

# A simple installation and administration tool for the large-scaled PC cluster system: DCAST

Tomoyuki HIROYASU<sup>1</sup>, Mitsunori MIKI<sup>1</sup>, Kenzo KODAMA<sup>2</sup>, Junichi UEKAWA<sup>2</sup>, and Jack DONGARRA<sup>3</sup>

<sup>1</sup> Department of Knowledge Engineering, Doshisha University, Japan,  
tomo@is.doshisha.ac.jp,

<sup>2</sup> Graduate Student, Doshisha University, Japan,

<sup>3</sup> Department of Computer Science, University of Tennessee, USA

**Abstract.** In this paper, a new setup/administration tool for PC cluster systems is proposed. Recently, in the high performance computing field, PC cluster systems are becoming popular. PC cluster systems consist of PCs connected via a network and are used for parallel and distributed computing. PC cluster systems achieve a good cost to performance ratio by using commodity hardware to construct the cluster. However, it is very hard to install and configure a PC cluster because many nodes exist and a large amount of knowledge is required for the installation and configuration of the cluster. In this paper, to solve this problem, a simple installation and administration tool for PC cluster called “Doshisha Cluster Auto Setup Tool: DCAST” is developed. DCAST has the following features: DCAST can be used for both diskless and diskfull clusters; DCAST is targeted for Linux; there is no interactive operation during the installation; slave nodes are booted over the network; and the whole system is reinstalled when reconfiguring the system. The targets, philosophy, and operations of DCAST are described.

## 1 Introduction to DCAST

PC clusters are parallel computers constructed from several PCs using network connection, and due to recent improvements in the speed of mass-produced CPU and advancement in high-speed networking such as Myrinet [1] and Gigabit Ethernet, it is now possible to attain similar computational ability to conventional supercomputers. Many PC clusters are ranked in to the Top500 list[2] which includes many of the World’s best supercomputers.

PC clusters provide high computational processing power for low cost, but it is not easy to construct.

Individual nodes that construct a PC cluster is each a computer that operates on its own. They require and Operating System each, and they each require software installation. In PC clusters, open source software are often used for the operating system and software. These software are often revised and corrected. When software revisions are released, all individual nodes that construct a PC cluster need to be revised.

It is possible to manually do software upgrades on all nodes of a PC cluster, but it is easy to make mistakes, and is a possible cause of trouble. Also, to perform such operations, knowledge of PC clusters and software are required, and it is a difficult task to perform for inexperienced people.

There are PC cluster installation tools such as Oscar[3], Rocks[4], Lucie[5], Scyld[6], and AlinkA[7]. They are advanced and polished installation tools, but when considering that PC cluster administrator is an inexperienced novice, the following problems need consideration:

1. A specification that allows a choice from several architectures may confuse the system administrator.
2. The administrator will not know which option to choose, even when there are interactive questions requiring a choice to be made.
3. Contrary to the technical level of the administrator, the desire to include new technology is high.
4. Want to consider security if possible.

In this research, targeting novice administrators as administrators, a tool to facilitate PC cluster construction and upgrading, Doshisha Cluster Auto Setup Tool (DCAST) was developed. DCAST aims at constructing a large-scale PC cluster with Debian GNU/Linux as an operating system. For upgrading a whole PC cluster, instead of upgrading individual nodes, all nodes that construct the PC cluster is reinstalled. By using DCAST, upgrades can be performed easily, without requiring the novice administrator to consider consistency throughout the PC cluster.

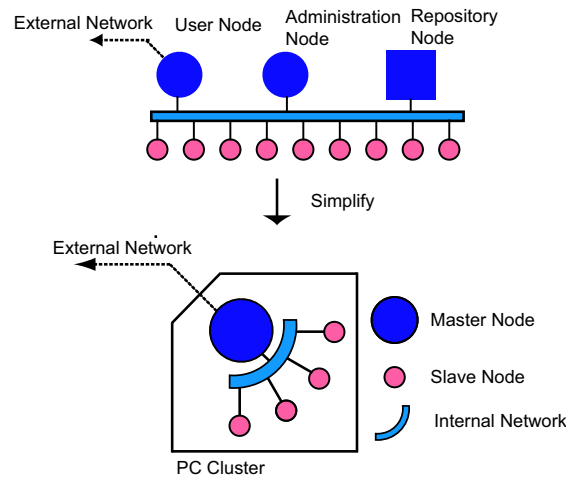
In this paper, problems of PC clusters are discussed, and then DCAST is proposed. After that, the design goals, operation from the users' viewpoint, DCAST internals when constructing a PC cluster, and PC cluster construction examples using DCAST are described.

## 2 PC clusters

### 2.1 Defining PC clusters

Beowulf-type cluster is a parallel computer which is a collection of workstations and PCs interconnected with an internal network, which operates as a single computational resource[8]. Recently, beowulf-type clusters which use PCs for individual nodes are attracting attention[9–17]. For example usage of such machines in computational simulation, protein tertiary structure prediction[9–11] and diesel engine fuel emission schedule designing[12, 13] are reported. In this paper, we call beowulf-type clusters consisting of PCs as PC clusters.

A basic construction diagram for PC cluster is shown in Fig.1. In this figure, a basic construction is consisted of a user node, an administration node, a repository node and computational nodes. As shown in the diagram, the basic construction is simplified into the model where the node which can communicate with the outside is called a Master node, and the other nodes are called Slave



**Fig. 1.** Basic PC cluster construct

nodes. In this model, a network that is independent from the outside network is constructed, and slave nodes cannot communicate to computers outside the PC cluster network.

## 2.2 PC Cluster characteristics

PC cluster have the following characteristics.

- Good cost-performance ratio  
It is possible to create a PC cluster cheaply by combining mass-produced parts. Also, by improving computational power of these parts, the cost/performance ratio is good.
- Can construct with arbitrary size  
It is possible to construct any size of PC cluster by changing the number of nodes that construct the PC cluster.
- Possible to construct with arbitrary combination  
It is possible to combine different CPUs in a PC cluster by placing different CPUs on different nodes. Also, it is possible to construct PC clusters with some nodes not having hard drives.
- Possible to introduce newer technologies  
It is possible to introduce new technologies that are brought into PCs, such as Gigabit Ethernet and Hyper-threading CPU. Also, it is possible to introduce latest software technology such as communication middleware and compilers.

PC cluster is a parallel computer which has many advantages as shown, but it has its problems.

### 2.3 Problems with PC Clusters

The problems with PC clusters are as follows:

- Requires knowledge in construction and maintenance  
Involved knowledge of operating systems and networking is required for PC cluster construction and maintenance, and it is a steep learning curve for novices.
- Installation effort  
PC cluster installation requires operating system installation on each node. Then, extra software and network configuration for operation as PC cluster is required. As PC cluster scales get larger, the configuration process alone becomes a large burden.
- maintenance cost  
Each nodes that constructs a PC cluster operates as one computer, and requires same amount of care as a standalone PC, such as software installation, hardware checking, software upgrading, and handling of hardware failures. As PC clusters get larger, the difficulty of finding the problem nodes and keeping consistency between nodes increase.

To solve these problems, PC cluster construction tool is proposed.

## 3 Design of PC cluster construction tool

### 3.1 Overview of proposed system

The proposed PC cluster construction tool is Doshisha Cluster Auto Setup Tool (DCAST). DCAST is a tool for efficiently constructing PC cluster. The tool is aimed at users who do not have specific skill about PC cluster, such as students. For those novice administrators, by using this tool it is possible to construct a large-scale PC cluster.

The target users of this tool have the following problems while constructing a PC cluster:

- Cannot handle the details of many interactive operations required for set ups while installing the operating system
- Cannot keep consistency of software between individual nodes
- Does not know which software to upgrade
- Wishes to improve security to a certain level, but does not know how
- Cannot provide information matching different computer architectures and characteristics

As a tool to solve these problems, DCAST was designed.

DCAST does a diskless booting, which involves using kernel and root filesystem from other nodes while booting a PC, so that a node that does not have a hard drive can start up. After that, without keyboard interaction, hard drive

partitioning, operating system installation, and other configurations for PC cluster node are performed, transforming the node to a state where it can function as a PC cluster node. DCAST can construct PC cluster efficiently using already existing software technologies.

Also, when problem occurs on some nodes and it is no longer possible to start up, using DCAST it is possible to perform a diskless boot, and rewrite the hard drive contents, to quickly solve the problem. Also when reinstallation of operating system is required, it is possible to reinstall using the same procedure as initial installation.

One of the characteristics of PC clusters is that it is possible to introduce latest technology to PC clusters. To introduce hardware with the latest technologies, a software which handles them is required. However, for large scale PC clusters, installing one software requires a large amount of work. Because DCAST does not require any keyboard interactivity, new technology can be installed with similar procedure to initial installation.

DCAST is a tool that is useful for PC cluster construction, maintenance and upgrading.

### 3.2 Design goals

The proposed system, DCAST is designed with the following design goals:

1. No interactivity at installation time  
Operating system installation requires a lot of interaction. A novice administrator cannot handle such interaction. Also, for constructing a large scale PC cluster, repeated interactive operations become require a large amount of human effort to process. For this reason, DCAST lacks interactive operation completely, and automates installation, to simplify the process of PC cluster construction and to reduce workload of the administrator.
2. For upgrading, reconstruct the whole PC cluster  
It is possible to introduce new technology to PC cluster. To do so, upgrading software is required. However, upgrading often makes keeping consistency between nodes of the PC cluster. To solve such problem, DCAST will reconstruct the whole PC cluster on upgrading.
3. Use Debian GNU/Linux[18]  
Debian GNU/Linux has an advanced package controlling mechanism, and it is possible to easily upgrade installed software and apply security updates on individual nodes. Also, it tracks conflicts and dependencies between packages it is easier to maintain consistency of software. This kind of package maintenance tool is specific to Debian GNU/Linux, and compared to other distribution, security updates and software upgrades can be performed with ease. With DCAST, Debian GNU/Linux is used.
4. Do not assume heterogeneous environment  
DCAST assumes novices such as students as a user, and does not consider heterogeneous cluster environment with computers of different architectures. All nodes are assumed to be of x86[19] architecture.

5. Allows both diskfull and diskless  
 Hard drives are moving parts, and they often break. Having many hard drives is not desirable in the view of PC cluster maintenance. To lighten the maintenance cost, there is a form of PC cluster where slave nodes do not have a local hard drive. Such node is called diskless node, and a PC cluster constructed by a master node with disk and diskless nodes is called diskless cluster. The nodes with hard drive are called diskfull nodes, and PC cluster which consists of diskfull nodes as diskfull cluster. DCAST is able to construct either type of PC cluster.
6. Integration with existing software  
 DCAST is constructed using several existing software. By using several software, DCAST software operation is divided, and results in a structure where applying improvements and bug fixes are easier.

By using this design, we aimed at constructing a tool that can construct and maintain a PC cluster.

### 3.3 Parts that construct DCAST

DCAST operates as a whole using other software. The software used is as follows:

- bootp[20]  
 bootp is a protocol used by nodes of a PC cluster to boot up from other nodes on the network. Using this protocol, even when hard drive is empty, it is possible to start up a kernel from the network, and obtain an IP address to start up a node. DCAST uses bootpd software to provide this service.
- tftp[21]  
 tftp is a trivial file transfer protocol, and the protocol used to transfer boot images by the request of bootp. This is required for loading the kernel from the network, to boot individual nodes of the PC cluster. Using this protocol, it is possible to boot the kernel on nodes which do not have a kernel image on its own. DCAST uses tftpd software to provide this service.
- NFS[22, 23]  
 Network File System is a protocol to provide server filesystem to a client through a network, developed by Sun Microsystems[24]. NFS is a de-facto standard protocol which is available on most UNIX-compatible systems. Using this protocol, root file system can be mounted even on machines without a hard drive, and allows construction of diskless PC cluster. DCAST uses nfs-kernel-server software to provide this service.
- grub[25]  
 grub is a bootloader for loading the operating system, and can boot up operating system specified in the configuration file. As a functionality of grub, it can analyze file system, so the configuration file can be placed within the file system, and it is easy to change configuration. Also, it has functionality as a bootp and tftp client, and it can network boot from the server. DCAST uses a floppy disk with grub image for booting up the slave nodes.

- update-cluster[26–28]  
update-cluster is a tool to maintain node information in an uniform manner using XML[29]. In DCAST, by using update-cluster, DCAST configuration file and software configuration file are generated automatically from XML. For example, the configuration file for the popular parallel programming library MPICH[30] can be generated. Also, update-cluster is effective for recreating configuration files when structure of PC cluster have changed.

## 4 Using DCAST

### 4.1 Procedure for constructing PC Cluster using DCAST

To construct PC cluster using DCAST, configuration of master node and slave node as specified in Fig.1 is required. The procedure is as follows:

1. Create master node
2. Install DCAST
3. Create grub floppy disk for slave node
4. Create DCAST configuration file
5. Invoke DCAST commands

Next, I will explain the details of individual steps of PC cluster construction using DCAST.

**Construction of Master node** First, master node construction is done. This process requires similar steps to installing Debian GNU/Linux on a standalone system. However, to use DCAST the kernel needs to support bootp, NFS, and the NIC. DCAST requires two types of kernel to be present; a type of kernel for the master node and another type of kernel for the slave node. Master node kernel has features such as NFS server, is installed on the master node. The slave node kernel which has features such as bootp client is placed on the directory on the master node where tftpd uses.

**DCAST installation** DCAST is installed on the constructed master node. DCAST is available publicly from the PC Cluster Group software web pages[31], Intelligent Systems Design Laboratory, Doshisha University. To install DCAST, download the compressed archive file of DCAST, and extract the contents to an arbitrary place on a system which has Debian GNU/Linux installed. Then, in that directory, type `./configure && make && make install`. Then install softwares shown in 3.3 such as bootpd, tftpd, nfs-kernel-server, and update-cluster.

**Creating grub floppy for slave nodes** To start slave nodes, grub boot images in floppy disks are used. DCAST provides a command `dcast-grubfloppy` for creating floppy disks with grub boot images. The command is used as follows:

```

#Enter PARTITION size.
FPRT /dev/hda1 128 boot *
SPRT /dev/hda2 512 swap
TPRT /dev/hda3 - /
4PRT none - none
NISDOMAIN nis.org
LOCALETHCARD eth0
# NETWORK NETMASK BROADCAST
NET 192.168.1.1 255.255.255.0 192.168.1.255
#DCASTMASTER Master's name Master's IP
DCASTMASTER host01 192.168.1.11
NFMASTER host_master 192.168.1.2
GATEWAY 192.168.1.254
#slave's name slave's IP slave's MACaddress
#Autogenerated by update-cluster
host02 192.168.1.12 009027D0A80B
host03 192.168.1.13 004005A06C67
host04 192.168.1.14 004005A886A5
host05 192.168.1.15 004005A06427
host06 192.168.1.16 004005A40DEE
host07 192.168.1.17 004005A40DEF
#End update-cluster

```

Fig. 2. slave.lst

#### dcast-grubfloppy NIC-name diskfull

By using this command, a floppy disk with grub image for a diskfull node that supports the NIC. Specifying `diskless` instead will result in creating a grub floppy image that starts up a diskless node.

**Writing configuration file** In DCAST all configuration is written to `slave.lst`. An example is shown in Fig.2.

In this file, configuration details such as network configuration are written in space-delimited manner. The meaning of the entries are:

– Partition construction

Specify the partition settings when building diskfull cluster. In DCAST, it is possible to create a maximum of 4 different partitions. The format is:

```
FPRT device-name size(MB) mount-point *
```

FPRT stands for “first partition”, and second partition (SPRT), third partition (TPRT), and 4th partition (4PRT) follow. For the device-name, specify Linux device names such as `/dev/hda1`. For the size, specify the partition size in megabytes. Writing “-” the rest of the region is allocated for the partition. To specify swap region, specify `swap` as the mount-point. “\*” specifies a boot sector, and it needs to be specified on one of FPRT, SPRT, TPRT, or 4PRT. Also, when constructing diskless clusters, there is no hard drive, and there is no need to specify the partition information in that case.

- NISDOMAIN  
Network Information Service(NIS) is a protocol to share information list databases such as host name and user information through computer network. Here, the domain name used by NIS is specified. It is not necessary to set this entry if NIS is not going to be used.
- LOCALETHCARD  
When there are more than one NIC on the master node, specify the NIC that is connected to the internal network of the PC cluster. Usually, eth0 is specified.
- NET  
Specify the network address, net mask, and broadcast address of the network inside the PC cluster.
- DCASTMASTER  
Specify the host name and IP address of the node which DCAST is being ran on.
- NFSMASTER  
Specify the host name and IP address of the node which provides the /home directory through NFS.
- GATEWAY  
Specify the gateway node used by the slave nodes.
- Configuration of slave nodes  
List for all slave nodes their host name, IP address, and MAC address. For MAC address, DCAST provides a functionality to automatically collect the information. The user can execute the following command as root.

```
tcpdump -e | getmac
```

When grub starts up on slave nodes, bootp request packet is sent from that node. By analyzing the packet header, it is possible to determine the MAC address of that specific slave node. `tcpdump` is the command used for monitoring packets on the network. Specifying the `-e` command-line option, `tcpdump` gives the MAC address.

By using the `getmac` command provided by DCAST, MAC addresses of the slave nodes are extracted, and written to a file called `macDB`. By booting all slave nodes it is possible to obtain the MAC address.

With above, all settings that the user needs to do is completed.

**Invoking DCAST** After completing the above operations and configuration, `dcast-setup` command, which is provided by DCAST will do the remaining task. After this, for diskless nodes, just starting up the slave node will make the nodes operate as diskless nodes. For diskfull nodes, powering the slave nodes will start the process of creating the root file system image on the hard drive. After this process, the nodes will reboot as diskfull nodes.

## 4.2 Maintenance after constructing PC cluster

**Services provided by the master node** On PC clusters that are constructed using DCAST, on the master node, services such as `bootpd`, `nfs-kernel-server` and `tftpd` are required after constructing the PC cluster. This is due to the requirement of these services for booting individual nodes on reinstalling to fix node failures, or on upgrading PC cluster. Also, these services are required for booting diskless slave nodes.

**Adding new nodes** To add new nodes to a PC cluster, the host name, IP address and the MAC address are added to the `slave.lst`. This operation allows adding of the new node.

**Maintenance for starting up the slave nodes** Slave nodes change boot-time operation according to grub configuration file. For diskless nodes, the grub configuration file for diskless nodes which is provided by DCAST (`menu.diskless`) are used.

For diskfull nodes, there is one grub configuration file for booting from the master node (`menu.diskfull`) and the one for booting from the local hard drive (`menu.local`).

When starting with `menu.diskfull`, the operating system will boot from the master node, and OS installation will start. For PC cluster upgrading, this configuration is used.

When starting with `menu.local`, the node will start up with the kernel available on the local hard drive, with no interaction with DCAST master node. For operation as diskfull node, start with `menu.local`.

As above there are some configurations required for booting for slave nodes of the PC clusters created with DCAST. There are commands to automate this task provided by DCAST.

- `exchangereboot`

Command which reboots after changing grub configuration file. Used like this:

```
exchangereboot diskfull
```

The command replaces grub configuration file of one node with `menu.diskfull`.

- `allexchangereboot`

Do `exchangereboot` on all nodes of a PC cluster.

Using these commands, grub configuration file can be modified easily, and thus it is possible to upgrade PC cluster without requiring much manual operation.

**Upgrading application software for the PC cluster** To upgrade software or install new software on a PC cluster, the operating system image is reinstalled on all nodes. The steps for reinstallation is described.

**Shutting down slave nodes** All slave nodes are shut down, then the respective root file system images on the master node are removed. DCAST provides the command `dcast-remove` command for facilitating the shutdown and removing of file system images.

**Upgrading master node** Master node is then upgraded. Software upgrading is done using Debian GNU/Linux package management system. For software that require different settings for master node and slave nodes, it is necessary to use module scripts, which is described later.

**Invoke `dcast-setup`** Invoke `dcast-setup` and create the root file system for slave nodes. After creating the root file system, boot the slave nodes. The upgrade process is then completed.

**Upgrading kernel of slave nodes** There are cases when kernel needs to be upgraded for handling new hardware. To upgrade the kernel of slave nodes, a new kernel is placed on the directory on the master node that tftpd uses. Then, rebooting the diskless node, the new kernel will be loaded. For diskfull nodes, the grub configuration needs to be modified to the one for booting from the master node (`menu.diskfull`), to load the new kernel from the master node. Using this procedure, slave node kernel upgrade can be done.

## 5 DCAST internals

### 5.1 DCAST internals when creating a PC cluster

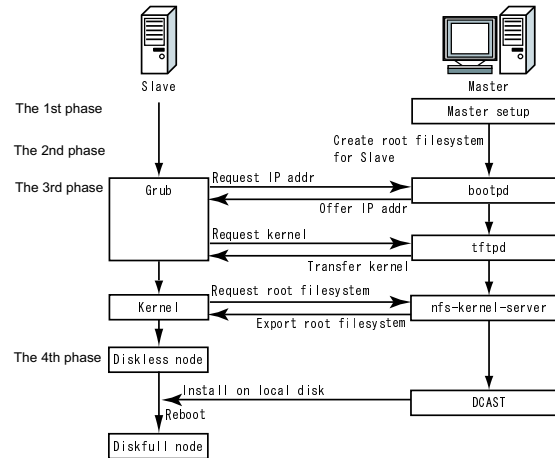
To construct a PC cluster using DCAST, one node for running DCAST on is required. Install the operating system on that one node. Use this node as the master node to create all the slave nodes. On the master node, the following software will run.

- bootpd for giving IP addresses to the slave nodes
- tftpd for giving kernel images to the slave nodes
- nfs-kernel-server for giving root file system images to the slave nodes

The configuration of these software for DCAST are done by `dcast-setup` command automatically.

The operation of DCAST can be divided into 4 steps. DCAST operation is shown in Fig.3.

Next, the details of operation on each step is explained.



**Fig. 3.** Operation of DCAST

**Step 1. Setting up the master node** Prepare and configure so that DCAST will work on the master node.

The template for the root file system for slave nodes is created. The template is created as follows:

1. copy all directories under `/var` except for `tftpboot`, `cache`, `tmp` directories, and remove `/var/lib/apt` and `/var/lib/dpkg`
2. Create `/home` top-level directory
3. Copy all other top-level directories, and remove `/lib/modules`

**Step 2. Creating root file system image for slave nodes** `dcast-setup` will create slave node root file system image from the template using the contents of `slave.lst` (Fig.2). Slave nodes use the corresponding directory as the root file system. The root file system for slave nodes that is used here is the same as root file system for the master node, except for changes that `dcast-setup` made for diskless nodes.

**Step 3. Booting slave nodes** Slave node will initially boot up grub, and send a request for an IP address to the master node. The `bootpd` running on master node will provide an IP address to the slave node. Slave node, after receiving the IP address, will send a request for a kernel image to the master node. Receiving this request, `tftpd` on the master node will give a kernel image to the slave node. Slave node will boot up using the kernel, and mount the root file system via NFS, which was previously prepared by `dcast-setup` on the master node, and start up as a diskless node. For diskless clusters, the process is complete here.

**Step 4. Constructing diskfull clusters** The next step is required for constructing a diskfull node. DCAST provides the `init` program used for starting up the operating system in step 3. The `init` provided by DCAST partitions the slave node hard drive, and creates root file system image from master node template. Template is provided via NFS. After creating the root file system, configuration necessary to create a diskfull node is done. For example, grub configuration is changed from network boot setting to booting from the local hard drive. After that, the node is rebooted.

By doing these processes, diskless node and diskfull nodes are constructed. To create a PC cluster, this process is repeated on each node.

## 5.2 Additional module scripts

For construction parts of PC cluster, many different kinds of hardware can be used. However, hardware addition usually requires software addition. Installation process for software requires an effort that is proportional to the number of nodes of a PC cluster. Therefore, it is effective to have a system to automatically install and configure applications on all the nodes of a PC cluster when constructing a PC cluster. DCAST has a system where different software can be installed when PC cluster is constructed.

**Module script operations** For installing software on PC clusters, different settings are sometimes done on each node. For example, IP address of the local host is different on each node. Therefore, there are two types of DCAST module scripts; the ones that are ran on master node, and the one which is ran on each slave node. On the master node, all scripts for the master node are ran when `dcast-setup` is invoked. On the slave node, slave node scripts are ran at boot time. By operating in this way, different configuration can be done on each node.

For software that does not require different settings on each node, only the module script for master node is required.

**Adding module scripts** Here, the method for creating and adding a DCAST module script is described. To signify that it is a module script for master node, the script file name ends with `.masterconfig`. Similarly module script for slave nodes have file names ending with `.slaveconfig`. The module scripts are placed in `/usr/lib/dcast`.

Here, the following module scripts are created and added to DCAST:

- `lm-sensors`[32]
- Myrinet[1] driver

Below, examples of adding module scripts are shown.

```

lm87-i2c-0-2e
Adapter: SMBus PLIX4 adapter at 0580
Algorithm: Non-I2C SMBus adapter
2.5V:    +0.00 V (min = +2.36 V, max = +2.63 V)  ALARM
Vccp1:   +1.68 V (min = +1.60 V, max = +1.77 V)
3.3V:    +3.28 V (min = +3.12 V, max = +3.47 V)
5V:      +5.10 V (min = +4.73 V, max = +5.26 V)
12V:     +12.25 V (min = +11.37 V, max = +12.62 V)
Vccp2:   +0.00 V (min = +1.60 V, max = +1.77 V)  ALARM
fan1:    4530 RPM (min = 3000 RPM, div = 2)
fan2:    4560 RPM (min = 3000 RPM, div = 2)
temp1:   +29.0°C (min = +10°C, max = +60°C)
CPU_Temp: +27.0°C (min = +10°C, max = +60°C)
vid:     +1.70 V

```

Fig. 4. Output of sensors command

**Module scripts for lm-sensors** As an example of software that does not require different set up on individual nodes, lm-sensors[32] exists. lm-sensors is a free hardware monitoring software that works on linux. It can detect CPU temperature, fan speed, and power supply voltage from the hardware sensors. lm-sensors source code is available from lm-sensors site[32]. An example of lm-sensors output is shown in Fig.4.

To install lm-sensors, the lm-sensors-source Debian package which is the lm-sensors source file, and i2c-source Debian package which is the i2c source file for BIOS sensor chip driver. lm-sensors installation consists of 4 steps:

1. Install i2c
2. Install lm-sensors
3. Use `sensors-detect` included in lm-sensors to detect the sensor chip on the motherboard
4. install the sensor chip driver into the kernel

`lm-sensors.masterconfig` is a module script that automates this process. The created script is placed under `/usr/lib/dcast`.

By adding a module script like this, it is possible to use a software that requires same setup on all machines on all nodes.

**Myrinet module script** Myrinet[1] is an example of requiring different configuration on each node. Myrinet is a networking system that is developed by Myricom[33]. It is possible to use TCP/IP networking over GM driver for Myrinet, and when using that, different IP address needs to be assigned for each node. In Table 1, a list of theoretical throughput values for Fast Ethernet, Myrinet, and Myrinet-2000 is given.

To use Myrinet, the Myrinet-GM driver needs to be prepared. Myrinet-GM is publicly available from Myricom web site [34].

**Table 1.** Performance of network

Network	Theoretical bandwidth (Mbit/s)	Network layer
Fast Ethernet	100 Mbit/sec	TCP/IP
Myrinet	1280 Mbit/sec	Myrinet-gm
Myrinet-2000	2 Gbit/sec	Myrinet-gm

Installation is possible by downloading the archive, extracting it, and executing the script.

To use Myrinet, the following operations are required.

1. Install Myrinet-GM
2. Obtain the MAC address and hostname of the node that is connected via Myrinet, and execute `mapper` program to initialise the route.

A script that does the above, `myrinet-gm.masterconfig` was created.

Next, IP address that is used by Myrinet is configured on each slave nodes. A script that does it on each slave node `myrinet-gm.slaveconfig` was created.

The created scripts are placed under `/usr/lib/dcast`.

By using module scripts, it is possible to use software that requires different settings on each node.

## 6 Examples of PC cluster construction using DCAST

In this section, real examples of PC cluster construction using DCAST are explained. The target PC clusters are as follows:

- A small PC cluster that consists of 4 diskfull nodes and 5 diskless nodes, with total of 9 nodes. (section 6.1)
- A large PC cluster that consists of 16 clusters of 16-node diskless nodes, ordered hierarchically (section 6.2)

By constructing the two types of PC clusters, it will demonstrate that DCAST can be used on:

- PC cluster where there are both diskfull nodes and diskless nodes
- PC cluster that constructs an hierarchical structure (hierarchical clusters)

### 6.1 PC cluster that has a mix of diskless node and diskfull node

In this section, a method of using DCAST to create a PC cluster that has a mix of diskless nodes and diskfull nodes is introduced. The target PC cluster, called Mill, is a PC cluster that consists of one master node and 8 slave nodes. Slave nodes consist of 3 diskfull nodes and 5 diskless nodes. The node structure is shown in Fig.5.

Here, as DCAST master node, PC cluster master node was used. The procedure is shown below:

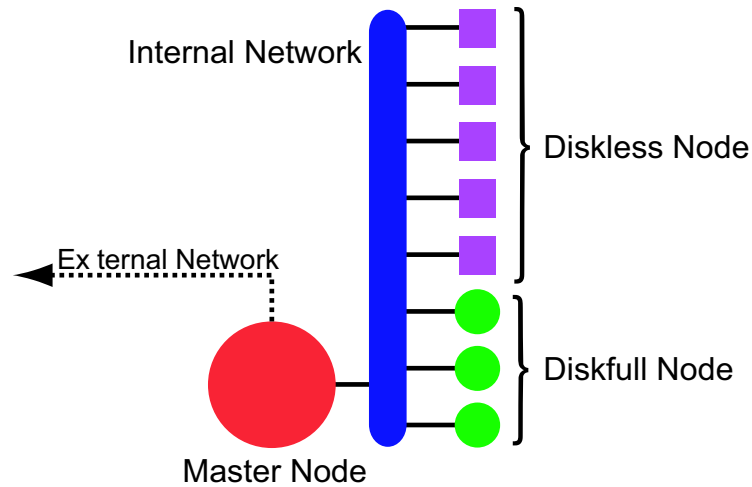


Fig. 5. Mill system node structure

**Step 1. Installing the operating system to the master node** Install Debian GNU/Linux on the master node.

**Step 2. Install DCAST** According to the procedure in 4.1, install DCAST.

**Step 3. Create grub floppy disk** According to 4.1, execute `dcast-grubfloppy` on the master node, and create grub floppy. The target PC cluster uses Intel PRO/100S Desktop Adapter[35] as the network interface card. Therefore, `dcast-grubfloppy` is invoked with `eeepro` as an argument, like:

```
dcast-grubfloppy eeepro diskfull
```

The second argument will be `diskless` when creating grub floppy disk for diskless nodes. The created grub floppy is then inserted to the slave node floppy disk drive.

**Step 4. Create slave.lst** According to 4.1, create `slave.lst`. The created `slave.lst` is shown in Fig.6.

Even when diskless node and diskfull node coexist, `slave.lst` is written in a similar manner to Fig. 2.

**Step 5. Invoke dcast-setup** According to 4.1, invoke `dcast-setup`. After this process finishes, all slave nodes are rebooted. When all slave nodes finish the process of rebooting, PC cluster construction completes.

```

#Enter PARTITION size.
FPRT /dev/hda1 128 boot *
SPRT /dev/hda2 512 swap
TPRT /dev/hda3 - /
4PRT none - none
NISDOMAIN nis.org
LOCALETHCARD eth0
# NETWORK NETMASK BROADCAST
NET 192.168.0.1 255.255.255.0 192.168.0.255
#DCASTMASTER Master's name Master's IP
DCASTMASTER mixed01 192.168.0.11
NFSMASTER mixed01 192.168.0.11
GATEWAY 192.168.0.11
#slave's name slave's IP slave's MACaddress
#Autogenerated by update-cluster
mixed02 192.168.0.12 009027D0A80B
mixed03 192.168.0.13 004005A06C67
mixed04 192.168.0.14 004005A886A5
mixed05 192.168.0.15 004005A06427
mixed06 192.168.0.16 004005A40DEE
mixed07 192.168.0.17 004005A40D37
mixed08 192.168.0.18 004005A40D75
mixed09 192.168.0.19 004005A40D0E
#End update-cluster

```

Fig. 6. slave.lst for Mill

By invoking DCAST like above, it is shown that it is possible to create a PC cluster where diskless nodes and diskfull nodes coexist.

## 6.2 Hierarchical cluster

In this section, a PC cluster where diskfull cluster and diskless cluster are connected hierarchically is constructed. The target PC cluster structure has 16-node diskless cluster which has a diskfull node as a master node each, and the master nodes of the diskless clusters construct a 16-node diskfull cluster. The structure of target PC cluster is shown in Fig. 7. Each diskless cluster slave nodes have their root file systems served from the master nodes of the diskless clusters, and the /home directory is served from the master node of the PC cluster (Fig. 7)

The following is the procedure to create the PC cluster.

### Step 1. Install operating system to one master node of diskless cluster

Install Debian GNU/Linux on one of the master nodes of the diskless cluster.

**Step 2. Install DCAST** According to 4.1, install DCAST to the node which was installed on step 1.

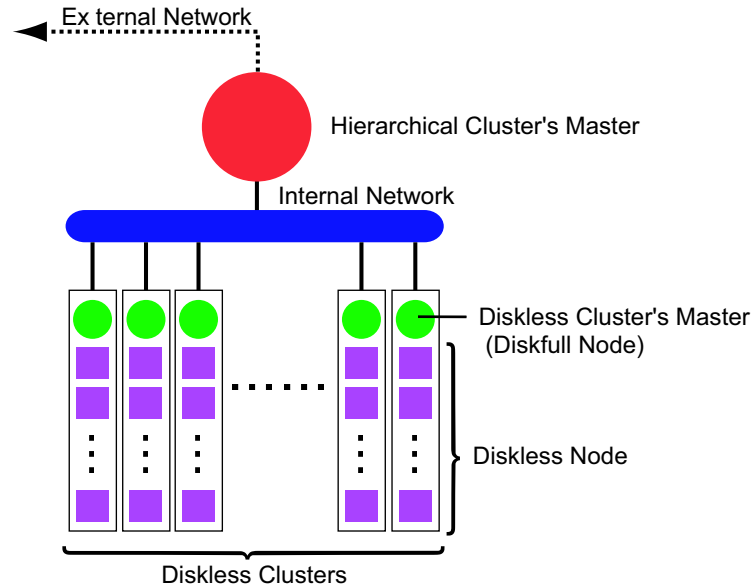


Fig. 7. cambria system node structure

**Step 3. Creating grub floppy disk** According to 4.1, run `dcast-grubfloppy` on the master node, and create grub floppy disk. The NIC used was DEC tulip chipset, so the command-line used is as follows:

```
dcast-grubfloppy tulip diskfull
```

Here, 256 grub floppy disk is created.

**Step 4. Create `slave.lst`** According to 4.1, create `slave.lst`. The created `slave.lst` is written similarly to one shown in Fig. 6.

Here, write the configuration required to install on the master nodes of the diskless clusters. As NFSMASTER, the master node of the PC cluster is set.

**Step 5. Invoke `dcast-setup`** According to 4.1, invoke `dcast-setup`. After that, boot up all remaining nodes of the diskfull cluster.

Then, the installation of the diskfull cluster of master nodes is completed.

**Step 6. Write `slave.lst`** According to 4.1, create `slave.lst`. Here, configuration for installing nodes on diskless nodes of the diskless clusters. As in step 4, NFSMASTER is the master node of the PC cluster.

**Step 7. Invoke dcast-setup on each master node of diskless cluster**  
 Invoke `dcast-setup` on 16 diskfull nodes, which are to serve as master nodes of diskless clusters. After that, start up all diskless nodes.

By using DCAST as described above, it is shown that it is possible to build a PC cluster with hierarchical structure with DCAST.

## 7 Conclusion

In this paper, we proposed **Doshisha Cluster Auto Setup Tool** (DCAST) and discussed the design, usage, internal implementation, and examples.

DCAST was designed with the following goals:

- Target at novices without specific knowledge to PC clusters
- Avoid interactive operations, and used Debian GNU/Linux for ease of software upgrades
- Software upgrading is done in the same manner as initial install
- Allow creation of both diskless cluster and diskfull cluster
- Combine existing software to create as much of the final product

The procedure to create a PC cluster using DCAST was shown, and the method to maintain the PC cluster after creation using DCAST was described. The operation of master node and slave nodes when creating PC cluster using DCAST was described. Also, the operation and method of adding module scripts for arbitrary software installation was explained.

In this paper, examples of PC clusters which diskfull nodes and diskless nodes coexist are shown.

In the future, the following enhancements are considered:

- Improve the speed in which root file system template is copied from mater node to slave node
- Addition of more module scripts
- A cluster description markup language for more verbose configuration that allows creation of a hierarchical PC cluster with one `dcast-setup` invocation

## References

1. Myricom <http://www.myri.com/myrinet/index.html>: Myrinet index page. (2002)
2. : TOP500 Supercomputer Sites. <http://www.top500.org/>. (2002)
3. Open Cluster Group: Open Cluster Group. OSCAR: A packaged cluster software stack for high performance computing. (2003)
4. Papadopoulos, P.M., Katz, M.J., Bruno, G.: NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters. IEEE Cluster 2001 (2001) 258–267
5. TAKAMIYA, Y., MANABE, A., SHIRASUNA, S., MATSUOKA, S.: Lucie: A fast installation and administration tool for large-scaled clusters(japanese). SWoPP Yuhuin 2002 (2002) 131–136
6. : SCYLD COMPUTING CORPORATION. <http://www.scyld.com/>. (1998)

7. ALINKA <http://www.alinka.com/>: ALINKA Cluster Solutions Home Page. (1999)
8. Sterling, T., Savarese, D., Beeker, D.J., Dorband, J.E., Renawake, U.A., Packer, C.V.: BEOWULF: A parallel workstation for scientific computation. In Proceedings of the 24th International Conference on Parallel Processing (1995) 11–14
9. Ogura, M., HIROYASU, T., MIKI, M.: Models For Distributed Memory Architecture Of Parallel Simulated Annealing Using Genetic Crossover. Proceedings of the EUROGEN2001 Conference (2001) 121–126
10. HIROYASU, T., MIKI, M., OGURA, M.: Parallel Simulated Annealing using Genetic Crossover . IASTED Proceedings of the IASTED International Conference PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS (2000) 139–144
11. YOSHIDA, T., HIROYASU, T., MIKI, M., OGURA, M., OKAMOTO, Y.: Energy Minimization of Protein Tertiary Structure by Parallel Simulated Annealing using Genetic Crossover . Proceedings of 2002 Genetic and Evolutionary Computation Conference (GECCO 2002) Workshop Program (2002) 49–51
12. HIROYASU, T., MIKI, M., KAMIURA, J., HIROYASU, S.W.H.: Multi-Objective Optimization of Diesel Engine Emissions and Fuel Economy using Genetic Algorithms and Phenomenological Model . SAESae, Technical Paper Series (2002)
13. KAMIURA, J., HIROYASU, T., MIKI, M., WATANABE, S.: MOGADES: Multi-Objective Genetic Algorithm with Distributed Environment Scheme . Computational Intelligence and Applications (2002) 143–148
14. HIROYASU, T., MIKI, M., WATANABE, S.: The New Model of Parallel Genetic Algorithm in Multi-Objective Optimization Problems(Divided Range Multi-Objective Genetic Algorithm). IEEE Proceedings of the congress on Evolutionary Computation 2000, Vol. 1 (2000) 333–340
15. WATANABE, S., HIROYASU, T., MIKI, M.: Parallel Evolutionary Multi-Criterion Optimization for Block Layout Problems . Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Volume 2 (2000) 667–674
16. MIKI, M., HIROYASU, T., KANEKO, M.: A Parallel Genetic Algorithm with Distributed Environment Schme . Proceedings of the Genetic and Evolutionary Computation Conference (2000) 376–376
17. HIROYASU, T., MIKI, M., HAMASAKI, M., TANIMURA, Y.: A New Model of Distributed Genetic Algorithm for Cluster Systems: Dual Individual DGA . Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Volume 1 (2000) 477–483
18. : Debian GNU/Linux – The Universal Operating System. <http://www.debian.org/index.en.html>. (1997)
19. Intel <http://www.intel.com/design/intarch/datashts/24018708.pdf>: Intel Datasheets: Intel386 SX Microprocessor Datasheet. (1994)
20. J. Reynolds <http://rfc.sunsite.dk/rfc/rfc1497.html>: RFC 1497: BOOTP Vendor Information Extensions. (1993)
21. K. Sollins <http://www.ietf.org/rfc/rfc1350.txt>: RFC 1350: THE TFTP PROTOCOL (REVISION 2). (1992)
22. Sun Microsystems, Inc <http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1094.html>: RFC 1094: NFS: Network File System Protocol Specification. (1989)
23. Sun Microsystems, Inc <http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1813.html>: RFC 1813: NFS Version 3 Protocol Specification. (1995)
24. : Sun Microsystems Home Page. <http://www.sun.com/>. (1994)
25. Free Software Foundation (FSF) <http://www.gnu.org/software/grub/>: GNU GRUB - GNU Project - Free Software Foundation (FSF). (1999)

26. Uekawa, J., Hiroyasu, T., Miki, M.: Mac address collection mechanism of update-cluster, the host listing integration system (japanese) (2002)
27. Uekawa, J.: update-cluster. Debian GNU/Linux, <http://www.netfort.gr.jp/~dancer/software/update-cluster.html>. (2003)
28. Powell, A.: Getting Started with Debian Beowulf, <http://lyre.mit.edu/~powell/debian-howto/beowulf-start.html>. (2003)
29. w3c <http://www.w3.org/XML/>: Extensible Markup Language (XML). (1996)
30. : MPICH-A Portable Implementation of MPI. (<http://www-unix.mcs.anl.gov/mpi/mpich/indexold.html>)
31. Cluster Group, Intelligent System Design Lab, Doshisha University <http://mikilab.doshisha.ac.jp/dia/research/cluster/index-e.html>: Cluster Group Webpage. (2001)
32. The Lm-sensors Group <http://www2.lm-sensors.nu/lm78/index.html>: Lm-sensors Site. (1998)
33. : Myricom Home Page. <http://www.myri.com/>. (2002)
34. Myricom <http://www.myri.com/scs/>: Myrinet Software and Documentation. (2003)
35. Intel <http://www.intel.com/network/connectivity/products/>: (Intel(r) PRO/100 S Desktop Adapter)