System and Performance Monitoring on Diskless PC Clusters

Chao-Tung Yang and Ping-I Chen

High-Performance Computing Laboratory
Department of Computer Science and Information Engineering
Tunghai University, Taichung City, 40704 Taiwan, R.O.C.
ctyang@mail.thu.edu.tw
g932834@student.thu.edu.tw

Abstract. In this paper, we introduce the experiments of monitoring on SLIM and DRBL diskless PC clusters. We constructed the performance experimentations by using 16 computing nodes and only one hard disk. We used LAM/MPI and PVM to run the matrix multiplication program in order to evaluate their performance. We also used spm2 and ntop to monitoring the system status during performance evaluation. As the result, we can find that the network loading of SLIM is less than DRBL even though the performance result of DRBL is better than SLIM. Maybe this is because the DRBL use NFS to mount everything which the nodes needed. We can realize that if we want to construct a high-performance diskless Linux cluster. We should take care of the free space of main memory and the mounting method of file systems.

Keywords: Diskless, System monitoring, Performance monitoring, PC clusters, DBRL, SLIM.

1 Introduction

The use of loosely coupled, powerful and low-cost commodity components (PCs or workstations, typically), especially without any hard disk drive, connected by high-speed networks has resulted in the widespread usage of a technology popularly called diskless cluster. Strictly speaking, it consists of one or more servers which provide not only bootstrap service but also related network services (such as DHCP, NIS, NFS servers, and etc) and many clients with no hard disk drive requesting for booting from the network. The availability of such clusters made maintain as easy as possible, and also reduced the waste in storage space. The diskless cluster differs from the traditional one in that a network is used to provide not only inter-processor communications but also a medium for booting and transmission for a live file system. Thus, each diskless node before booting can boot through a floppy disk or a NIC’s boot ROM with a small bootstrap program and even with a NIC’s PXE, which sends a broadcast packet to a DHCP server and is then assigned an IP address. After each node has been assigned a valid IP address, it sends a request to the TFTP server for getting the boot image, referred to the Linux Kernel, through TCP/IP protocol and starts the booting process. During the booting process, all the necessary system files are transmitted through the network. After the remote file system is mounted as root file system (NFS_ROOT), and the system initialization is done, the node is ready to work.

In this paper, we introduce the experiments of monitoring on SLIM and DRBL diskless PC clusters. We constructed the performance experimentations by using 16 computing nodes and only one hard disk. We used LAM/MPI and PVM to run the matrix multiplication program in order to evaluate their performance. We also used spm2 and ntop to monitoring the system status during performance evaluation. As the result, we can find that the network loading of SLIM is less than DRBL even though the performance result of DRBL is better than SLIM. Maybe this is because the DRBL use NFS to mount everything which the nodes needed. We can realize that if we want to construct a high-performance diskless Linux cluster. We should take care of the main memory's free space and the mounting method of file systems.

We can realize that if we want to construct a high-performance Linux diskless PC cluster. We should take care of the free space of main memory and the mounting methods of file systems. Using NFS to mount everything can ease the nodes' main memory loading. But it will cause the network loading very high. Using TFTP to download the whole server's Linux image file will waste a lot of main memory space. Maybe we can combine the advantage of them to construct the brand-new high performance diskless Linux cluster.

2 Background Review

2.1 PXE

PXE (Preboot eXecution Environment) is Intel's loosely defined standard for booting PCs over the network. A PXE-capable BIOS or boot ROM can download bootstrapping code and load an operating system over the network. Booting Linux with PXE is a straightforward way of starting a diskless workstation or appliance in a closed network. PXE defines a method for the BIOS or NIC ROM to fetch a booting code over the network. It does this via standard Internet protocols. When the appliance is powered on, the BIOS or ROM makes a DHCP request.

The DHCP server, recognizing the appliance as a network-booting client, returns instructions on the location of a TFTP server and the name of the name of the file that it should download from the server. First, it checks it's configuration files (located on the TFTP server) and then downloads the Linux kernel, passing it the necessary kernel arguments, including the IP information received from DHCP. When the kernel loads, it uses the IP information provided for it. The kernel can either be download an initial ramdisk (initrd) and use that as a root filesystem, or it may just connect to an NFS server. Once the kernel has gotten the root file system setup, the operating system can complete the boot process, and the system is ready for use.[1]

2.2 TFTP

TFTP is a simple protocol to transfer files, and therefore was named the Trivial File Transfer Protocol or TFTP. It has been implemented on top of the Internet User Datagram protocol (UDP or Datagram), so it may be used to move files between machines on different networks implementing UDP. It is designed to be small and easy to implement. Therefore, it lacks most of the features of a regular FTP. The only thing it can do is read and write files (or mail) from/to a remote server. It cannot list directories, and currently has no provisions for user authentication. In common with other Internet protocols, it passes 8-bit bytes of data. Three modes of transfer are currently supported: netascii (This is ASCII as defined in "USA Standard Code for Information Interchange" with the modifications specified in "Telnet Protocol Specification".) Note that it is 8 bit ASCII. The term "netascii" will be used throughout this document to mean this particular version of ASCII.); octet (This replaces the "binary" mode of previous versions of this document.) raw 8 bit bytes; mail, netascii characters sent to a user rather than a file. (The mail mode is obsolete and should not be implemented or used.) Additional modes can be defined by pairs of cooperating hosts. [2]

2.3 NFS

The Network File System (NFS) was developed to allow machines to mount a disk partition on a remote machine as if it were on a local hard drive. This allows for fast, seamless sharing of files across a network. It also gives the potential for unwanted people to access your hard drive over the network (and thereby possibly read your email and delete all your files as well as break into your system) if you set it up incorrectly. So please read the Security section of this document carefully if you intend to implement an NFS setup. [3]

3 Two Types of Diskless PC Cluster

There are many ways to construct a Linux cluster. But dealing with differences with node file systems is time consuming and can be problematic. So, some research groups created this kind of cluster in order to construct the cluster in an efficient way.

It also costs less money and we can manage the nodes easily. In our experiment, we only use two kinds of diskless cluster. One is SLIM, which is designed by University of Hong Kong. Another is DRBL, which is designed by National Center of High-performance Computing.

3.1 The SLIM Cluster

SLIM stands for Single Linux Image Management. It holds the Linux OS image to be shared by all PCs via network booting. This solution is solely designed and implemented by people in Department of Computer Science, The University of Hong Kong.

The SLIM server holds pre-installed Linux system image for sharing across the network. The system image is exported as read only by the NFS to all client PC to build their local root file system during booting up. One SLIM server may serve as many as OS images made by different Linux distributions. It also provides TFTP service to allow client PC to download network boot loader. It also holds OS boot images which are Linux kernel and initrd for network boot loader to download. [4]

3.2 The DRBL Cluster

DRBL stands for Diskless Remote Boot in Linux. This solution is solely designed and implemented by people in National Center of High-performance Computing, Taiwan.

DRBL uses PXE/etherboot, NFS, and NIS to provide services to client machines. Once the server is ready to be a DRBL server, then the client machines can boot via PXE/etherboot (diskless). “DRBL” does NOT touch the hard drive of the clients, so other Operating Systems (for example, M$ Windows) installed on your client machines will not be affected. This may be important in a phased deployment of GNU/Linux, where users still want to have the option of booting to Windows and running Office. DRBL allows you to be flexible in your deployment of GNU/Linux [5].

4 Our System Architecture

Our SMP cluster is a low cost Beowulf-type class supercomputer that utilizes multi-computer architecture for parallel computations. The clusters consists of sixteen PC-based symmetric multiprocessors (SMP) connected by one 24-port 100Mbps Ethernet switches with Fast Ethernet interface. Its system architecture is shown in Figure 1. There are one server node and fifteen computing nodes. The server node has one Intel Pentium 4 2.8GHz (with hyper-threading) processor and 512MBytes of shared local memory. The other fifteen nodes are AMD MP 2000+ SMP machines with 1GBytes of shared local memory. Each individual processor is rated at 1.6GHz.

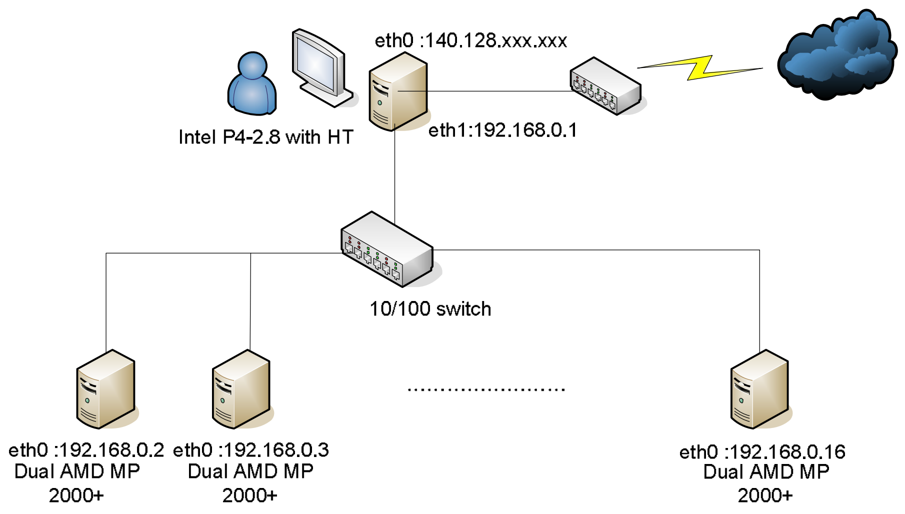


Fig. 1. Diskless system architecture.

At first, we used 16-port switch to handle all of the machines. It had no trouble on SLIM diskless system. But when we changed the system to DRBL, there was something wrong. The whole network loading was very high, so that we could not SSH to another machine and could not startup PVM and LAM/MPI services which were in our private network. So, we used another 24-port switch on DRBL diskless system, and the network returned to normal.

We thought that it might be caused by NFS service. We also had some trouble with SLIM. At the start, we used a PC which has only 256MB memory. Sometimes, the boot-up process would fail or could not start some daemon. So, we added more memory and then the whole process returned to normal. We thought it might be that we fully installed the Linux system, so that the image is too large for a PC which has only 256MB memory to handle the boot-up process and start the daemon. Table 1 shows the equipments and software about our diskless cluster.

Table 1. Equipments ans software on each machine.

Machines / Equipments	Diskless server	Nodes
CPU	Intel P4-2.8G with HT	AMD Dual MP2000+
OS	SLIM: Fedora Core 2; DRBL: Fedora Core 1	
Kernel	SLIM: 2.6.5; DRBL: 2.4.22	
Disk	1	none
RAM	512MB	1G
SWITCH	PCI FX-32n 10/100	
Network Interface Card	2	1

5 Performance Monitoring

5.1 Experimental Environments

5.1.1 PVM

PVM (Parallel Virtual Machine) is a portable message-passing programming system, designed to link separate host machines to form a “virtual machine’’ which is a single, manageable computing resource.

The virtual machine can be composed of hosts of varying types, in physically remote locations. PVM applications can be composed of any number of separate processes, or components, written in a mixture of C, C++ and FORTRAN. The system is portable to a wide variety of architectures, including workstations, multiprocessors, supercomputers and PCs. [6]

5.1.2 LAM/MPI

LAM/MPI is an implementation of the Message Passing Interface (MPI) parallel standard that is especially friendly to clusters. It includes a persistent runtime environment for parallel programs, support for all of MPI-1, and a good chunk of MPI-2, such as all of the dynamic functions, one-way communication, C++ bindings, and MPI-IO. [7]

5.1.3 SPM2

Compaq Storage Performance Monitor for Linux (CSPM) has evolved into System Performance Monitor 2 by hp. Spm2 now provides monitoring of cpu, mem, storage, network, and irq utilization This monitor is being released as open source to get community involvement. [8]

5.1.4 ntop

ntop is a network traffic probe that shows the network usage, similar to what the popular top Unix command does. ntop is based on libpcap and it has been written in a portable way in order to virtually run on every Unix platform and on Win32 as well.
ntop users can use a a web browser (e.g. netscape) to navigate through ntop (that acts as a web server) traffic information and get a dump of the network status. In the latter case, ntop can be seen as a simple RMON-like agent with an embedded web interface. The use of:

- a web interface
- limited configuration and administration via the web interface
- reduced CPU and memory usage (they vary according to network size and traffic)

make ntop easy to use and suitable for monitoring various kind of networks. [9]

5.2 Startup time of SLIM and DRBL

First, we use SPM2 to monitor the status of CPU, Memory, Network, and Storage when the system starup. Figure 2 shows the SPM2 monitoring result.

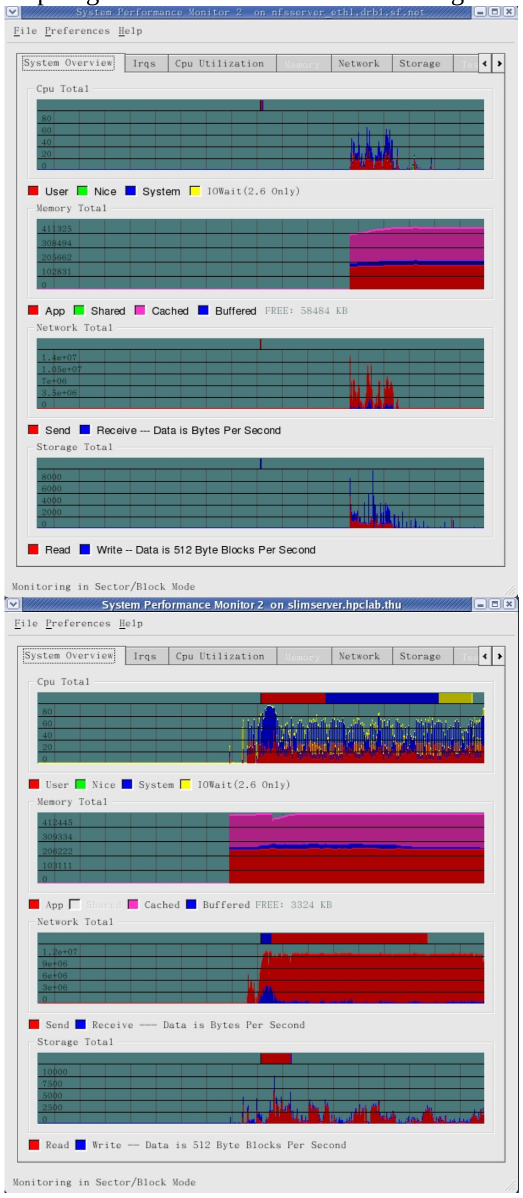
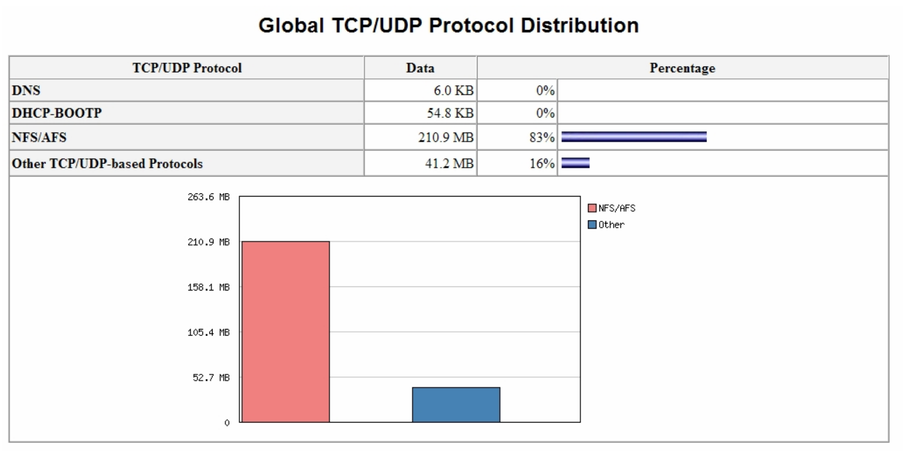


Fig. 2. SPM2 monitoring results on DRBL (upper-side) and SLIM (lower-side) during the system startup.

We can find that among CPU, Memory, Network, and Storage on SLIM are busier than on DRBL. SLIM also has a larger I/O wait so that the startup time is more than DRBL. We also want to know what makes the result so different. We think that it might because SLIM and DRBL use different kind of method to boot-up the node site. So, we use another monitoring tool which called ntop to get more detail information about the network status. Figure 3 shows the monitoring result of DRBL using ntop.

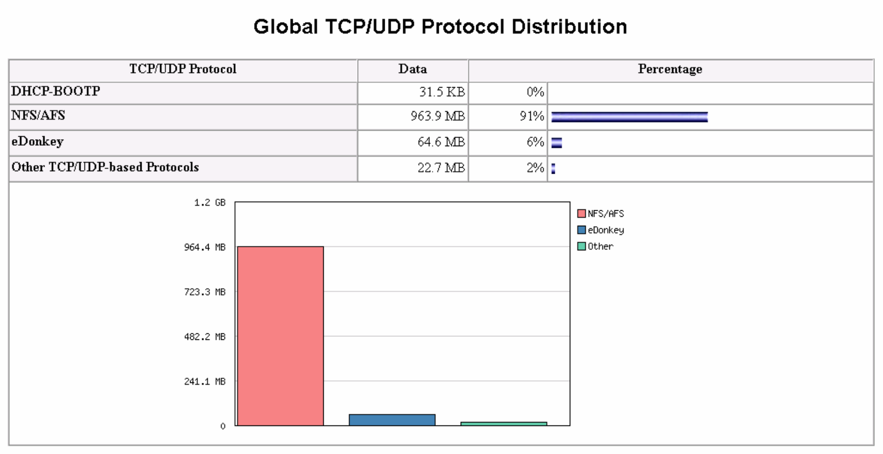


Global TCP/UDP Protocol Distribution

TCP/UDP Protocol	Data	Percentage
DNS	6.0 KB	0%
DHCP-BOOTP	54.8 KB	0%
NFS/AFS	210.9 MB	83%
Other TCP/UDP-based Protocols	41.2 MB	16%

Fig. 3. DRBL monitoring result using ntop.

As the result, we can find that the network loading is mainly result in NFS which is about 83% of total network flow. Figure 4 shows the monitoring result of SLIM using ntop.



Global TCP/UDP Protocol Distribution

TCP/UDP Protocol	Data	Percentage
DHCP-BOOTP	31.5 KB	0%
NFS/AFS	963.9 MB	91%
eDonkey	64.6 MB	6%
Other TCP/UDP-based Protocols	22.7 MB	2%

Fig. 4. SLIM monitoring result using ntop

We can find that the main network flow is caused by NFS. But the data transmitted by NFS on SLIM is almost four times than DRBL.

5.3 Performance Evaluation Monitoring

After the system startup, we used LAM/MPI and PVM to run matrix multiplication program and bioinformatics software in order to evaluate the system performance. The matrix multiplication problem sizes were 256×256, 512×512, 1024×1024, and 2048×2048 in our experiments and were running by different CPU numbers [7].

The biggest price we had to pay for the use of a PC cluster was the conversion of an existing serial code to a parallel code based on the message-passing philosophy. The main difficulty with the message-passing philosophy is that one needs to ensure that a control node (or master node) is distributing the workload evenly between all the other nodes (the compute nodes). Because all the nodes have to synchronize at each time step, each PC should finish its calculations in about the same amount of time. If the load is uneven (or if the load balancing is poor), the PCs are going to synchronize on the slowest node, leading to a worst-case scenario. Another obstacle is the possibility of communication patterns that can deadlock. A typical example is if PC A is waiting to receive information from PC B, while B is also waiting to receive information from A.

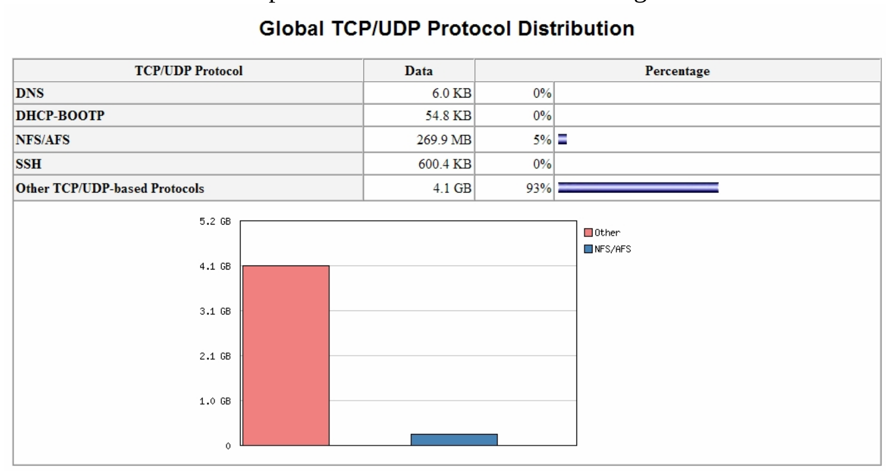
The matrix operation derives a resultant matrix by multiplying two input matrices, **a** and **b**, where matrix **a** is a matrix of *N* rows by *P* columns and matrix **b** is of *P* rows by *M* columns. The resultant matrix **c** is of *N* rows by *M* columns. The serial realization of this operation is quite straightforward as listed in the following:

```
for(k=0; k<M; k++)
    for(i=0; i<N; i++){
          c[i][k]=0.0;
          for(j=0; j<P; j++)
                c[i][k]+=a[i][j]*b[j][k];
    }
```

Its algorithm requires n^3 multiplications and n^3 additions, leading to a sequential time complexity of $\boldsymbol{O}(n^3)$. Let's consider what we need to change in order to use PVM. The first activity is to partition the problem so each slave node can perform on its own assignment in parallel. For matrix multiplication, the smallest sensible unit of work is the computation of one element in the result matrix. It is possible to divide the work into even smaller chunks, but any finer division would not be beneficial because of the number of processor is not enough to process, i.e., n^2 processors are needed.

The matrix multiplication algorithm is implemented in PVM using the master-slave paradigm. The master task is named `master_mm_pvm`, and the slave task is named `slave_mm_pvm`. The master reads in the input data, which includes the number of slaves to be spawned, *nTasks.* After registering with PVM and receiving a *taskid* or *tid,* it spawns *nTasks* instances of the slave program `slave_mm_pvm` and then distributes the input graph information to each of them. As a result of the spawn function, the master obtains the *tids* from each of the slaves. Since each slave needs to work on a distinct subset of the set of matrix elements, they need to be assigned instance IDs in the range (0... *nTask-1*). The *tids* assigned to them by the PVM library do not lie in this range, so the master needs to assign the instance IDs to the slave nodes and send that information along with the input matrix. Each slave also need to know the total number of slaves in the program, and this information is passed on to them by the master process as an argument to the spawn function since, unlike the instance IDs, this number is the same for all *nTasks* slaves.

We also use ntop to get the information about the system's network status. Figure 5 shows the result of DRBL performance evaluation monitoring.



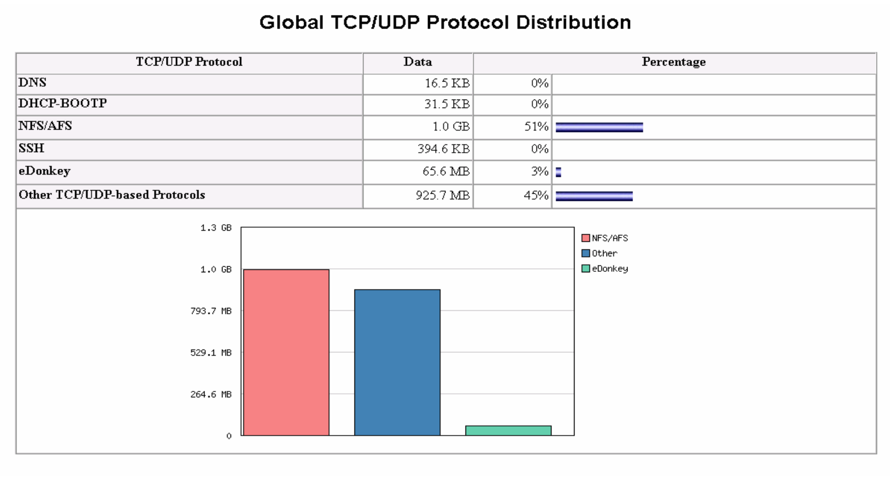
Global TCP/UDP Protocol Distribution

TCP/UDP Protocol	Data	Percentage	
DNS	6.0 KB	0%	
DHCP-BOOTP	54.8 KB	0%	
NFS/AFS	269.9 MB	5%	
SSH	600.4 KB	0%	
Other TCP/UDP-based Protocols	4.1 GB	93%	

Fig. 5. DRBL monitoring result using ntop (after performance evaluation).

We can find that almost 93% of the network flow is TCP/UDP-based communication. Only 5% is caused by NFS. Then, we use the same way to proceed the performance evaluation on SLIM. Figure 6 shows the result of SLIM performance evaluation monitoring.

We can see that the network flow of TCP/UDP communication is about 45%, and the other 51% is caused by NFS. The total network flow is about 2GByte. But the total network flow of DRBL is about 4.3G. It is almost twice of SLIM. Although the network flow of NFS on DRBL is less than SLIM. But the network flow of TCP/UDP transmission is four times the result of SLIM. The network loading is too high for DRBL. It might because of DRBL's node boot-up method is different from SLIM.


Global TCP/UDP Protocol Distribution

TCP/UDP Protocol	Data	Percentage
DNS	16.5 KB	0%
DHCP-BOOTP	31.5 KB	0%
NFS/AFS	1.0 GB	51%
SSH	394.6 KB	0%
eDonkey	65.6 MB	3%
Other TCP/UDP-based Protocols	925.7 MB	45%

Fig. 6. SLIM monitoring result using ntop (after performance evaluation).

6 Discussions

Diskless PC clusters offer several advantages over traditional workstations in the area of manageability and security, such as:

- No moving parts, so they are less susceptible to dust, noise and vibration,
- Hard disk or floppy failure is no longer an issue,
- Less security risk since no data is stored locally,
- Easier to replace than traditional workstation - no OS and/or software re-installation is required.

After the experimentation, we can find out that the network loading of DRBL is very high although the performance result is better than SLIM. We also realize that if we choose SLIM as our diskless system, we may have to add more main memory. If we choose DRBL as our diskless system, we really need a good network environment, or it will not be worked.

7 Conclusions and Future Work

In this paper, we introduce our experiments on SLIM and DRBL diskless clusters. We constructed them by using 16 machines and only one disk. We also used LAM/MPI and PVM to run the matrix multiplication program in order to evaluate their performance. Then, we use spm2 and ntop to monitor the system's status. Finally, we got the result that the system loading of DRBL may be less than SLIM. We think it is because the DRBL use NFS to mount everything which the nodes needed. We can realize that if we want to construct a high-performance diskless Linux cluster. We should take care of the main memory's free space and the mounting method.

Referencs

[1] PXE, http://www.linuxdevices.com/articles/AT5834950453.html,

[2] TFTP, http://www.faqs.org/rfcs/rfc1350.html,

[3] Network file system, http://nfs.sourceforge.net/nfs-howto/intro.html#WHAT

[4] SLIM, http://slim.csis.hku.hk/

[5] DRBL, http://drbl.sourceforge.net/

[6] PVM – Parallel Virtual Machine, http://www.epm.ornl.gov/pvm/

[7] LAM/MPI Parallel Computing, http://www.lam-mpi.org

[8] SPM2, http://cspm.sourceforge.net/

[9] ntop, http://www.ntop.org/ntop.html

[10] R. Buyya, *High Performance Cluster Computing: System and Architectures*, Vol. 1, Prentice Hall PTR, NJ, 1999.

[11] R. Buyya, *High Performance Cluster Computing: Programming and Applications*, Vol. 2, Prentice Hall PTR, NJ, 1999.

[12] T. L. Sterling, J. Salmon, D. J. Backer, and D. F. Savarese, How to Build *a Beowulf: A Guide to the Implementation and Application of PC Clusters*, 2nd Printing, MIT Press, Cambridge, Massachusetts, USA, 1999.

[13] Gregory R. Watson, Matthew J. Sottile, Ronald G. Minnich, Sung-Eun Choi, Erik A. Hendriks, "Pink: A 1024-node Single-System Image Linux Cluster," *Proceedings of the Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region* (HPCAsia'04)

[14] B. Wilkinson and M. Allen, *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, Prentice Hall PTR, NJ, 1999.

[15] M. Wolfe, *High-Performance Compilers for Parallel Computing*, Addison-Wesley Publishing, NY, 1996.

[16] C. T. Yang, S. S. Tseng, M. C. Hsiao, and S. H. Kao, "A Portable parallelizing compiler with loop partitioning," *Proc. of the NSC ROC(A)*, Vol. 23, No. 6, pp. 751-765, 1999.