World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:8, No:5, 2014

Some Issues with Extension of an HPC Cluster

Pil Seong Park

***Abstract*—**Homemade HPC clusters are widely used in many small labs, because they are easy to build and cost-effective. Even though incremental growth is an advantage of clusters, it results in heterogeneous systems anyhow. Instead of adding new nodes to the cluster, we can extend clusters to include some other Internet servers working independently on the same LAN, so that we can make use of their idle times, especially during the night. However extension across a firewall raises some security problems with NFS. In this paper, we propose a method to solve such a problem using SSH tunneling, and suggest a modified structure of the cluster that implements it.

***Keywords*—**Extension of HPC clusters, Security, NFS, SSH tunneling.

I. Introduction

THE desire to get better computing power and reliability by combining a number of low cost off-the-shelf computers has given rise to an architecture called a computer cluster, and it is widely used as a low-cost alternative, typically being much cost-effective than a single computer of comparable speed or availability [1].

Incremental growth is another benefit of a computer cluster. Many small labs first build their own homemade Linux cluster, and add more dedicated nodes later. However, instead of adding dedicated nodes to the cluster, if there are some other nodes that are being used for other purposes on the same local area network (LAN), we may try to utilize their idle times by extending the cluster to include them, as long as they are not always busy enough, especially during the night.

Incremental growth results in a heterogeneous cluster with nodes running possibly different Linux versions. On the other hand, a computer cluster normally consists of the dedicated nodes that reside on an isolated private network behind a firewall. To extend the cluster to include the non-dedicated nodes outside of the firewall, we will be confronted with some security problem in communication and data sharing between nodes.

In this paper, we deal with such issues arising when we extend an old homemade Linux cluster across a local area network, and propose a solution using SSH (Secure Shell) tunneling. It is not a state-of-the-art method adopting up-to-date technologies, assuming that we do not upgrade hardwares and softwares of old nodes.

II. Background

A. HPC Clusters

A computer cluster consists of a set of loosely connected computers that work together so that they can be viewed as a single system in many respects. The components of a cluster are usually connected to each other through a fast LAN, with each node running its own instance of an operating system.

Computer clusters may be configured for different purposes. Load-balancing clusters are configurations in which nodes share computational workload like a web server cluster. High-performance computing (HPC) clusters are used for computation-intensive purposes, rather than handling IO-oriented operations. High-availability (HA) clusters improve the availability of the cluster, by having redundant nodes, which are then used to provide service when system components fail. The activities of all compute nodes are orchestrated by "clustering middleware", a software layer that allows treating the cluster via a single system image concept.

Well-known HPC middlewares based on message passing are the Message Passing Interface (MPI) [11] and the Parallel Virtual Machine (PVM) [6], the former being the de facto standard. LAM/MPI [15], FT-MPI, and LA-MPI are some of widely used non-commercial MPI implementation libraries, and their technologies and resources have been combined into the on-going Open MPI project [18].

In this paper, we are concerned about extension of a small tightly-coupled asymmetric HPC cluster only, in which a master/login node (or master, for short) sits in front of compute nodes (or slaves, for short), administering the whole function of the cluster. In many cases, the master node can also have attached storage that is exported to the compute nodes using insecure NFS (Network File System) over UDP. All the nodes sit on a secure private LAN protected by a firewall.

The Berkeley NOW (Network of Workstations) project is one of early attempts to harness the power of clustered machines connected via high-speed switched networks on a building-wide scale [13]. However such an extension gives rise to difficulties in security, administering the cluster, and load forecasting for optimal performance.

Data partitioning and load balancing are important components in parallel computation. Since earlier works (e.g., see [12]), many authors have studied load balancing using different strategies on dedicated/non-dedicated heterogeneous systems [3]-[5], [9] but it is nearly impossible to find works on the security problems arising in cluster expansion, which is rather technical than academic.

Our cluster “Hydra” we want to extend has 10 nodes with Pentium 3 Xeon processors running Fedora Core 4, with LAM/MPI v.7.1.2 installed. The nodes are interconnected via a Gigabit LAN, and NFS is used for file sharing. For detailed information about LAM/MPI, see [10].

B. NFS and SSH Tunneling

NFS is a protocol created by Sun Microsystems in 1984.

Pil Seong Park is with the University of Suwon, Hwasung, Gyeonggi-do 445-743, Korea (phone: +82-31-220-2163; fax: +82-31-229-8281; e-mail: pspark@suwon.ac.kr).

NFS was developed to allow file sharing between systems residing on a LAN. The Linux NFS client supports three versions of the NFS protocol: NFSv2(1989), NFSv3(1995), and NFSv4(2000). However NFS as is has many problems to use in extending the cluster, since its packets are not encrypted and due to other shortcomings which will be discussed later.

Other alternatives to NFS include AFS (Andrew File System), DFS (Distributed File System), RFS (Remote File System), Netware, etc. [16]. There are also various clustered file systems shared by multiple servers [14]. However we do not adopt these new technologies since they may not supported old Linux versions and hardwares.

To securely extend the HPC cluster across a firewall, encryption of NFS traffic is required. One of the techniques that is ordinarily used is known as cryptographically protected tunneling. In this case, an IP-level or TCP-level stream of packets is used to tunnel application-layer segments [2]. A TCP tunnel is a technology that aggregates and transfers packets between two hosts as a single TCP connection. By using a TCP tunnel, several protocols can be transparently transmitted through a firewall. Under certain conditions, it is known that the use of a TCP tunnel severely degrades the end-to-end TCP performance, which is called TCP meltdown problem [7].

The SSH protocol allows any client and server programs to communicate securely over an insecure network. Furthermore, it allows the tunneling (port forwarding) of any TCP connection on top of SSH, so as to cryptographically protect any application that uses clear-text protocols.

III. Extension of an HPC Cluster

NFS itself is not secure since NFS relies on the inherently insecure UDP protocol (up to NFSv3), transactions between host and client are not encrypted, and IP spoofing is possible. Moreover, firewall configuration is difficult because of the way NFS daemons work, i.e., some ports used are not fixed.

We would like to extend the HPC cluster to include non-dedicated nodes Ex1 and Ex2 across the firewall, as shown in Fig. 1.

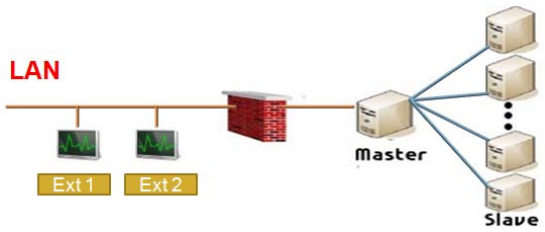

Fig. 1 Extension of an HPC cluster to include nodes Ext1 and Ext2 across the firewall

A. Fixing NFS Ports for SSH Tunneling

SSH tunneling, which makes use of SSH port forwarding, is widely used to encrypt some unencrypted packets or to bypass firewalls, e.g., see [2], [19].

But NFS as is has the following problems in the use with SSH tunnels.

1) NFS uses UDP protocols by default, and the ports of some daemons essential for the operation of NFS are variable.
2) SSH tunnels support only TCP protocols of fixed ports.

TCP protocols are also supported from the Linux kernel 2.4 and later on the NFS client side, and from the kernel 2.4.19 on the server side [17]. Since all the nodes satisfy this, all we need to do is just use the option "-o tcp" in the mounting command. The following is an example to specify the option when mounting the NFS server's directory.

mount –t nfs **–o tcp** *server*:/*nfs_dir* *mount_pt*

where *server* is the NFS server's name or its IP, *nfs_dir* is the NFS directory on the server, and *mount_pt* is the mount point on the client.

The following are the daemons essential for NFS operation:

- fixed ports: portmapper (port 111), rpc.nfsd (port 2049)
- variable ports: rpc.mountd, rpc.lockd, rpc.statd, rpc.rquotad

The ports of the latter four are randomly assigned by the operating system, and the ports can be fixed by specifying port numbers in the NFS configuration file (Fig. 2) and defining new port numbers in /etc/services (Fig. 3) which contains all port numbers used by Linux [21].

```
STATD_PORT=4001
LOCKD_TCPPORT=4002
LOCKD_UDPPORT=4002
MOUNTD_PORT=4003
```

Fig. 2 The configuration file /etc/sysconfig/nfs to be created to fix the port of the 4 daemons

```
rquotad 4004/tcp      # rpc.rquotad tcp port
rquotad 4004/udp      # rpc.rquotad udp port
```

Fig. 3 Defining the new port 4004 in the configuration file /etc/services

B. Setting up an SSH Tunnel

On the server side, the configuration file /etc/exports has to be modified so that its NFS directory to be exported to clients can be mounted by itself. The following is an example to export the NFS directory /home to itself.

/home **localhost** (sync,rw,insecure,root_squash)

where "insecure" means it allows connection from ports higher than 1023.

Then we need to set up an SSH tunnel from the client's side. For example, to forward the ports 11000 and 12000 on the client's side to the fixed ports 2049 (rpc.nfsd) and 4003 (rpc.mountd), respectively, we can use the command

```
# ssh nfssvr -L 11000:localhost:2049 \                    (1)
             -L 12000:localhost:4003 -f sleep 600m
```

where "*nfssvr*" is the IP or the name of the server registered in the configuration file /etc/hosts, and "-f sleep 600m" means that port forwarding is to last for 600 minutes in the background.

Once connected to the NFS server, an SSH tunnel will be

open if the correct password is entered. Then manually mount the NFS server's export directory. The following command is an example to mount the /home directory of the NFS server on the /mynfs directory of the client.

```
# mount -t nfs -o tcp,hard,intr,port=11000, \
  mountport=12000 localhost:/mynfs
```

C. The Suggested Structure of an Extended Cluster

Even though the NFS connection through an SSH tunnel is encrypted, it has a serious drawback if we cannot utterly and completely trust the local users on the NFS server [17]. For example, if some local user on the NFS server can login on the server and create an SSH tunnel, any ordinary user on the server can mount the file systems with the same rights as root on the client.

One possible solution might be prohibiting local users' direct login to the NFS server to prevent creating an SSH tunnel. One simple way is changing all local users' login shells from /bin/bash to /sbin/nologin in the file /etc/passwd.

Then it causes another problem. In general, the master server normally works as the NFS server too in small homemade HPC clusters, as shown in Fig. 1. However local users should be allowed to login the master server to use the cluster, which is dangerous when using NFS through an SSH tunnel, as was pointed out previously.

Fig. 4 is an alternative of the structure of an extended HPC cluster that takes everything into account. The structure has a separate NFS server which does not allow local users' login. An SSH tunnel can be created by the superuser on Ext1 or Ext2, and local users just login on the master node to use the cluster.

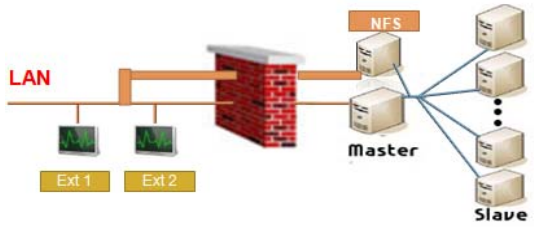

Fig. 4 The suggested structure of an extended HPC cluster

D. Some More Remarks

The firewall setting of the master node does not need any modification. However, since the NFS service is provided through SSH tunneling, it is required for the NFS server to open the port 22 only to the computation nodes outside of its firewall that participate in parallel computation.

The NFS server should be configured so that it releases as little information about itself as possible. For example, services like portmapper should be protected from outside. And it is suggested to specify explicitly the hosts that are allowed to access all the services on the NFS server. This can be done by setting ALL:ALL in the configuration file /etc/hosts.deny, and listing explicitly the hosts (or their IPs) together with the services which are allowed to access, in /etc/hosts.allow file.

The use "rsh" command, which is not encrypted, is common on old HPC clusters with old middlewares in parallel computation. Fortunately LAM/MPI v.7.1.2 allows SSH login with an authentication key but without a password, e.g., see [20].

Note that the original HPC cluster may be homogeneous, i.e., the performances of all slave nodes may be the same, until some new nodes with different performance are added. However, the extended HPC cluster may be heterogeneous or may act like a heterogeneous one even if all the nodes have the same power, since the communication speeds between nodes are variable here and there depending on various factors: the types of network, the existence of firewall, and necessity of encryption. Moreover the workload of the non-dedicated servers outside of the firewall may change continually. Hence we need to use a dynamic run-time load balancing strategy [8], while assigning equal amount of work to the original slaves of the cluster.

IV. Performance Tests

The performance of the NFS through an SSH tunnel will inevitably drop due to encryption overhead. We compare the performances of NFS with or without SSH tunneling. The separate NFS server and the non-dedicated servers Ext1 and Ext2 are all 1.6 GHz Pentium 4 machines with 1GB memory, equipped with Intel Pro/100 fast Ethernet card. They all run Fedora Core 4.

Tests were performed between the NFS server and Ext1 machine, using UDP or TCP, and with or without an SSH tunnel across the firewall. The times it took to read or write a file of size 1GB from Ext1 were measured, at varying NFS block sizes. Since NFSSVC_MAXBLKSIZE (maximum block size) of the NFS in Fedora Core 4 is 32*1024 (see /usr/src /kernels/2.6.11-1.1369_FC4-i686/include/linux/nfsd/const.h), tests were performed at 4k, 8k, 16k, and 32k, respectively, 3 times for each and they are averaged. In addition, to delete any remaining data in cache, NFS file system was manually unmounted and mounted again between tests.

The following shows example commands that measure the time taken to create the file /home/testfile of size 1GB on the NFS server and read it using block size 16KB.

```
# time dd if=/dev/zero of=/home/testfile bs=16k \
  count=65536
# time dd if=/home/testfile of=/dev/null bs=16k
```

The results of the NFS performance test using the above commands, with or without SSH tunneling are given in Table I. For the common NFS without tunneling, the figures in parentheses are the times it took when TCP is used, and others are when UDP is used.

TABLE I
PERFORMANCE OF NFS, WITH OR WITHOUT SSH TUNNELING (SEC)

Block size	Common NFS		SSH tunneled	
	Write	Read	Write	Read
4K	118.07 (122.69)	96.20 (95.22)	132.79	101.37
8K	117.57 (120.47)	95.10 (94.52)	123.59	98.86
16K	114.96 (117.29)	93.96 (92.58)	125.22	95.09
32K	112.46 (115.44)	92.74 (91.85)	117.46	94.03

As we see, the larger the NFS block size, the faster in all cases. For the common NFS without SSH tunneling, write operation using UDP is slightly faster than TCP, but it is not the case for read operation. Moreover the NFS with SSH tunneling takes 4.5%-12.5% more time for write and 1.4-5.4% more for read, than the common NFS using UDP

As long as the NFS block size is taken as large as possible, the tunneling overhead may not be large even though NFS service is done through SSH tunneling, since the non-dedicated nodes outside of the firewall need not read or write so often through NFS, which is common in high performance parallel computing.

V. CONCLUSION

HPC clusters are widely used in many small labs, because they are easy to build, cost-effective, and easy to grow. Instead of adding new nodes, we can extend clusters to include some other servers on the same LAN, so that we can make use of their idle times. However, unlike a tightly-coupled HPC cluster behind a firewall, the resulting system suffers a security problem with NFS which is vital for HPC clusters.

Of course there are many new good solutions using recent technologies. However we do adopt such solutions, because they require upgrades of hardwares and/or softwares including an operating system. Instead we devise a solution using SSH tunneling, which can be applied to the old system as is. Probably this approach may be helpful to many of small homemade cluster systems.

We were concerned only about the NFS security, but not the security in the communication between the non-dedicated nodes outside of a firewall and the master node, because we configured LAM/MPI to use the secure SSH protocol.

ACKNOWLEDGMENT

This work was supported by the GRRC program of Gyeonggi province [GRRC SUWON2013-B1, Cooperative CCTV Image Based Context-Aware Process Technology].

REFERENCES

[1] M. Baker and R. Buyya, "Cluster Computing: the commodity supercomputer", *Software-Practice and Experience*, vol.29(6), 1999, pp.551-576.
[2] M. Dusi, F. Gringoli, and L. Salgarelli, "A Preliminary Look at the Privacy of SSH Tunnels", in *Proc. 17th International Conference on Computer Communications and Networks(ICCCN '08)*, 2008.
[3] M. Eggen, N. Franklin, and R. Eggen, "Load Balancing on a Non-dedicated Heterogeneous Network of Workstations." in *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2002)*, June 2002.
[4] J. Faik, L. G. Gervasio, J. E. Flaherty, J. Chang, J. D. Teresco, E.G. Boman, and K. D. Devine, "A model for resource-aware load balancing on heterogeneous clusters", *Tech. Rep. CS-03-03*, Williams College Dept. of Computer Science, http://www.cs.williams.edu/drum/, 2003.
[5] I. Galindo, F. Almeida, and J. M. Badia-Contelles, "Dynamic Load Balancing on Dedicated Heterogeneous Systems", *Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science*, vol.5205, 2008, pp 64-74
[6] G. A. Geist, A. Beguelin, J. J. Dongarra, W. Jiang, R. Manchek, and V. S. Sunderam, *PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, MA, USA 1994.
[7] O. Honda, H. Ohsaki, M. Imase, M. Ishizuka, and J. Murayama, "Understanding TCP over TCP: Effects of TCP tunneling on end-to-end throughput and latency," in *Proc. 2005 OpticsEast/ITCom*, Oct. 2005.
[8] L. V. Kale, M. Bhandarkar, and R. Brunner, "Run-time Support for Adaptive Load Balancing", in *Lecture Notes in Computer Science, Proc. 4th Workshop on Runtime Systems for Parallel Programming (RTSPP)* Cancun – Mexico, Vol. 1800, 2000, pp.1152-1159.
[9] K. Lu, R. Subrata, and A. Y. Zomaya, "On the performance-driven load distribution for heterogeneous computational grids", *J. of Computer and System Sciences* vol.73, 2007, pp.1191-1206.
[10] A. I. Margaris, "Local Area Multicomputer (LAM-MPI)", *J. Computer and Information Science*, Vol.6(2), 2013, pp.1-8.
[11] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. J. Dongarra, *MPI: The Complete Reference*. MIT Press, Cambridge, MA, USA, 1996.
[12] M. Zaki, W. Li, and S. Parthasarathy, "Customized Dynamic Load Balancing in a Heterogeneous Network of Workstations", in *1996 Proc. 5th IEEE Int. Symposium on High Performance Distributed Computing*.
[13] Berkeley NOW Project, http://now.cs.berkeley.edu/
[14] Clustered file system, http://en.wikipedia.org/wiki/Clustered_file_system
[15] LAM/MPI Parallel Computing, http://www.lam-mpi.org/
[16] Linux NFS-HOWTO, http://nfs.sourceforge.net/nfs-howto/
[17] Linux NFS Overview, FAQ and HOWTO, http://nfs.sourceforge.net/
[18] Open MPI, http://www.open-mpi.org
[19] Port Forwarding Using SSH Tunnel, http://www.fclose.com/b/linux/818/port-forwarding-using-ssh-tunnel/
[20] ssh-keygen: password-less SSH login http://rcsg-gsir.imsb-dsgi.nrc-cnrc.gc.ca/documents/internet/node31.html
[21] http://www.linuxquestions.org/questions/linux -security-4/firewall-blocking-nfs-even-though-ports-are-open-294069/