SSH 터널을 이용한 HPC 클러스터의 확장[1)]

박필성, 하쉿 쿠마
수원대학교 컴퓨터학과

Expansion of An HPC Cluster Over SSH Tunnel

Park, Pil Seong, Kumar, Harshit
University of Suwon
E-mail : {pspark, kumar}@suwon.ac.kr

요 약

실시간으로 데이터를 처리하여 빠른 서비스를 제공하기 위해 PC 클러스터가 널리 사용되고 있다. 본 논문에서는 PC 클러스터의 한 종류인 HPC 클러스터에 전용 노드를 추가하는 대신, 방화벽 외부의 네트워크 상에 존재하는 비전용 노드의 유휴시간을 활용하도록 클러스터를 확장하여 성능을 향상시키는 경우 발생하는 NFS 등의 보안 문제를 SSH 터널링을 사용하여 해결하는 방안을 제시하고 암호화된 NFS의 성능을 실험하였다.

1. 서론

실시간으로 데이터를 처리하여 고객에게 빠른 서비스를 제공하기 위해서는 강력한 컴퓨터의 사용이 필수적이다. 그러나 컴퓨터는 성능의 향상에 비해 가격이 기하급수적으로 높아지므로, 고성능 컴퓨터 대신 저렴하고 제작이 쉬운 PC 클러스터의 사용이 일반적이다. PC 클러스터의 한 종류인 HPC 클러스터의 경우, 모든 노드는 클러스터의 전용 노드로서 프론트엔드(front-end) 혹은 매스터(master) 노드 뒤의 사설 네트워크상에 존재하므로 프론트엔드 노드의 방화벽이 안전하면, 노드간의 데이터의 전송 및 파일 공유 등에 있어서 비록 암호화되지 않는다 하더라도 보안상 별 문제가 되지 않는다.

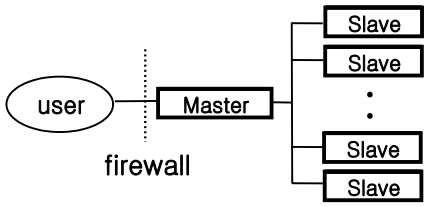
그러나 전용 노드들 외에, 외부 네트워크 상의 다른 목적으로 사용되는 비전용 노드의 유휴시간을 활용하기 위해 이들을 포함하도록 클러스터를 확장하면 추가적 비용 없이도 성능을 향상시킬 수 있을 것이다 [3]. 그러나 이 경우는 매스터 노드와 외부 네트워크 상의 비전용 노드간의 데이터 전송 및 데이터 공유에 따른 보안이 문제가 된다.

본 연구에서는 HPC 클러스터를 외부의 비전용 노드를 포함하여 확장할 경우, SSH 터널(tunnel)을 통해 데이터를 공유하도록 함으로써 이런 문제를 해결하고, 암호화된 NFS의 성능을 시험하였다.

2. HPC 클러스터의 확장에 따른 문제점

HPC 클러스터는 MPI(Message Passing Interface) [10]나 PVM(Parallel Virtual Machine) [12] 등의 미들웨어를 사용하여 구성되며, 그림 1과 같이 전체 연산을 총괄하는 매스터(master) 노드 아래 실제 작업을 수행할 슬레이브(slave) 노드들로 구성된다. 매스터 노드는 I/O를 담당하고 전체를 컨트롤하며 모든 노드들의 파일 공유를 위해 NFS 서버의 역할을 수행하므로 연산에 직접적으로 참여하지 않거나 적은 부분만을 담당하는 것이 일반적이다. 슬레이브 노드는 외부에 노출되지 않아 보안상 유리하고, 다른 부하가 걸리지 않아 안정적이며 고정적 부하 분산 정책을 사용할 수 있다는 장점이 있다.

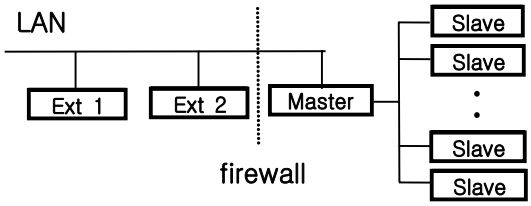
- 본 연구는 경기도의 경기도지역협력연구센터(GRRC) 사업의 일환으로 수행하였음 [GRRC수원2009-A2, 지식기반 상황인식 정보 시스템 연구]



(그림 1) 보통의 HPC 클러스터

한편 개방된 LAN상의 컴퓨터들의 유휴시간을 활용하는 클러스터는 1990년대 버클리 대학의 NOW (Network of Workstations)가 그 시초이나 [5], 이 방식은 보안상 취약하며 관리가 어렵고, 부하 예측이 불가능하여 동적 부하분산이 주요 이슈가 된다 ([13] 참고). 한편 실습실의 컴퓨터를 야간 또는 방학동안 클러스터로 활용하는 연구([1] 참고)도 있으나 이는 클러스터의 범위를 하나의 실습실 또는 건물 전체로 넓힌 것으로 근본적으로 보통의 HPC 클러스터와 같은 구조이다.

본 논문에서는 그림 2와 같이, 방화벽으로 보호된 전용 HPC 클러스터를 LAN 상의 비전용 노드(Ext 1, Ext 2 등)를 포함하도록 확장하고, 그에 따라 발생하는 다음과 같은 문제를 다루고자 한다.



(그림 2) 개방된 LAN으로 확장된 HPC 클러스터

- 모든 노드는 암호화되지 않는 NFS(network file system)를 통해 파일을 공유한다.
- 방화벽의 설정이 어렵다.
- 병렬 연산에는 노드간에 암호 없이 전달/실행이 가능한 “Berkeley R” 명령들(rlogin, rsh, rexec)이 주로 사용되는데, 이들은 암호화되지 않는다.

3. SSH 터널에 의한 NFS의 암호화

3.1 SSH 터널을 위한 NFS 포트 고정

NFS처럼 파일 공유 기능을 제공하는 것으로는 AFS(Andrew file system) [4], Coda(Coda file system) [6] 등이 있으며, 이들은 약간의 보안 등의 추가적인 기능을 제공한다. 그러나 NFS는 현재 충분히 성숙된 표준으로서 많은 플랫폼에서 지원되며 HPC 클러스터에서 널리 사용되므로 그대로 NFS를 사용토록 한다.

SSH의 포트포워딩(port forwarding) 기능을 이용한 SSH 터널링(tunneling)은 암호화되지 않은 프로토콜을 안전하게 암호화하거나 방화벽을 우회하는 데 널리 사용되며 그 방법은 널리 알려져 있다 [2,10,15].

그런데 NFS는 SSH 터널의 사용에 있어서 다음과 같은 문제가 있다.

1) NFS는 디폴트로 UDP를 사용하나, SSH 터널은 TCP만 지원한다.
2) NFS 작동에 필수적인 일부 서비스는 포트가 고정되지 않는데, SSH 터널링을 위해서는 이를 고정해야 한다.

NFS 클라이언트측은 Linux 커널 2.4 이후, 서버측은 2.4.19부터 TCP가 지원된다 [9]. NFS가 TCP를 사용토록 하기 위해서는, 클라이언트에서 NFS 서버가 제공하는 디렉토리를 마운트할 때 -o tcp 옵션을 주면 된다.

NFS 작동에 필수적인 것은 portmapper(111), rpc.nfsd(2049), rpc.mountd, rpc.lockd, rpc.statd, rpc.rquotad이다.(괄호 속의 숫자는 고정된 포트 번호이며, 번호가 지정되지 않은 것은 유동적이다). 유동적인 포트는 서버측에서 그림 3의 예와 같이 적절한 설정 파일에 지정함으로써 고정할 수 있다 [14].

```
/etc/sysconfig/nfs에 추가
  STATD_PORT=4001
  LOCKD_TCPPORT=4002
  LOCKD_UDPPORT=4002
  MOUNTD_PORT=4003
/etc/services에 추가
  rquotad 4004/tcp # rpc.rquotad tcp port
  rquotad 4004/udp # rpc.rquotad udp port
```

(그림 3) NFS 포트 고정을 위한 수정의 예

3.2 NFS를 위한 SSH 터널 설정

NFS 서버측의 설정

NFS 설정 파일 /etc/exports에 클라이언트에게

제공할 디렉토리를 자기 자신(즉 localhost)에게 제공하도록 다음과 같이 설정하고 NFS 서버를 시작한다. 다음은 /home을 제공하는 예이며, insecure는 1023보다 높은 포트에서의 접속을 허용하라는 것이다.

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

NFS 클라이언트측의 설정

클라이언트 측에서는 다음과 같이 SSH 터널을 설정한다. 다음은 클라이언트의 11000번 및 12000번 포트를 각기 서버의 2049(rpc.nfsd) 및 앞에서 고정한 4003(rpc.mountd)번 포트로 포워딩하는 명령이다. “nfssvr”는 /etc/hosts에 등록된 NFS 서버의 이름이며, "-f sleep 60m"은 포트포워딩이 백그라운드에서 600분 동안 수행되도록 한다.

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

원격 서버에 접속되면 암호를 입력하고 SSH 터널이 열리는데, 그러면 클라이언트의 다른 접속을 통하여 NFS 마운트를 한다. 다음은 NFS 서버의 /home을 클라이언트의 /home2에 마운트하는 명령이다.

```
# mount -t nfs -o tcp,hard,intr,port=11000, \
   mountport=12000 localhost:/home /home2
```

4. 추가적으로 보완된 HPC 클러스터

4.1 SSH 터널에 의한 NFS의 심각한 문제

이상과 같이 SSH 터널을 통한 NFS는 암호화되기는 하나 포트포워딩이라는 관점에서 NFS 서버의 사용자를 완전히 믿지 못하는 상황에서는 심각한 문제가 발생할 수 있다 [9].

예를 들어, NFS 서버에 일반 계정을 가진 어떤 사용자가 자신이 관리자인 클라이언트 머신으로부터 SSH로 NFS 서버로 로그인하며 SSH 터널을 만듦으로써 자신의 클라이언트 머신의 포트를 2049번과 4003번(위의 예에서) 포트로 포워딩하면 NFS 서버의 관리자와 같은 권한으로 공유된 파일 시스템에 접근할 수 있다는 점이다.

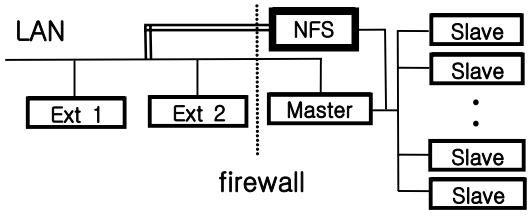
4.2 보완된 HPC 클러스터의 구조

한 가지 해결책은, NFS 서버에는 파일 공유를 위해 HPC 클러스터를 사용하는 일반 사용자의 계정은 만들어두되 로그인은 허용하지 않는 것이다. 여러 가지 방법이 있을 수 있겠으나, 단순한 한 가지 방법은 /etc/passwd 파일에서 일반사용자에 대해서는 로그인 셸 /bin/bash 대신 /sbin/nologin으로 지정하는 것이다. 만일 일반 사용자가 직접 NFS 서버로 로그인하려고 하면 다음과 같이 옳은 암호를 입력해도 로그인이 불가능하게 되므로 SSH 터널을 생성할 수 없을 것이다.

```
$ ssh nfssvr
pspark@nfssvr's password:
Last login: Wed Oct 10 15:50:43 2009 from
211.221.225.179
This account is currently not available.
Connection to nfssvr closed.
$
```

한편 그림 2와 같은 보통의 HPC 클러스터의 구조에서는 매스터가 NFS 서버의 역할을 겸한다. HPC 클러스터의 사용을 위해서는 매스터로의 접속이 필요하므로 NFS 서버의 역할을 매스터로부터 분리하여 그림 4와 같이 별도의 NFS 서버를 두어야 할 것이다.

NFS 서버는 외부 네트워크상의 노드로는 SSH 터널을 위한 22번 포트만 열어두면 된다. 일반 사용자는 이전과 같이 매스터로 로그인하여 사용하면 된다.



(그림 4) 별도의 NFS 서버를 가진 확장된 HPC 클러스터

한편 NFS 서버는 일반사용자의 로그인이 불가능하므로 병렬 연산에는 참여할 수 없다.(LAM MPI는 일반사용자로서의 사용을 권장하고 있다

[8]) 따라서 공유된 파일시스템의 접근이 많지 않은 보통의 HPC 클러스터에는 성능이 낮은 컴퓨터를 NFS 서버로 사용할 수도 있을 것이며, 또한 NFS 서버의 역할을 더 이상 수행할 필요가 없는 매스터는 더 많은 부하를 담당하도록 할 수 있을 것이다.

4.3 기타 추가적 보완책

매스터의 외부로의 방화벽 설정은 이전과 별 달라질 것이 없다. 다만 NFS 서버는 외부의 계산에 참가하는 노드들에게는 22번 포트를 열어두고 불필요한 것은 모두 차단한다.

NFS 서버는, 보안을 위해서는 가능한 한 적은 정보를 노출하도록 설정한다. 예를 들어, portmapper 등은 외부로부터 철저히 차단한다. /etc/hosts.deny 파일을 ALL:ALL로 일단 모든 호스트의 모든 서비스로의 접근을 차단하고, /etc/hosts.allow에 접근을 허용할 호스트(즉 전용 클러스터 내부의 매스터와 슬레이브 노드들)들을 허용할 서비스와 함께 명시적으로 나열한다.

한편 LAM MPI를 사용하는 HPC 클러스터에서, 일반적으로 많이 사용하는 rsh 대신, 암호없이 SSH(secure shell)를 사용하도록 설정하는 방법은 이미 [3]에서 언급하였다. 다만 LAM MPI를 사용한 병렬 연산은 각 사용자가 자신의 병렬 프로세스를 띄워 연산을 수행하므로 각 사용자가 설정하는 것이 원칙이나, 시스템 관리자가 /etc/bashrc에 위 두 줄을 추가하면 각 사용자는 별도의 설정이 불필요하다.

4.4 SSH 터널을 통한 NFS의 성능 실험

HPC 클러스터를 외부로 확장하는 경우, SSH 터널을 통한 NFS의 성능을 내부 네트워크의 전통적 NFS와 비교 실험하였다. NFS 서버는 Pentium 4, 1.6GHz, 512MB 메모리를 가지며, 내부의 슬레이브 노드 및 외부 노드는 Pentium 4, 1.6GHz, 1GB 메모리를 가진 것이다. 모든 노드는 Fedora Core 4 Linux를 운영체제로 사용하며, Intel Pro/100 fast Ethernet 카드를 장착하고 있다. 모든 노드는 1Gbps까지 지원하는 3Com Switch 2824(3C16479)로 연결하였다.

실험은 보통의 암호화되지 않는 NFS(각기 UDP 및 TCP), 그리고 SSH 터널을 통한 NFS의 3가지 방식에 대하여, NFS의 전송 버퍼의 크기(즉 rsize와 wsize)를 달리하면서, 클라이언트로부터 서버로 1GB의 파일을 쓰고 읽는 데 걸리는 시간을 측정하였다. Fedora Core 4 NFS의 최대 데이터 전송 버퍼의 크기 NFSSVC_MAXBLKSIZE는 32*1024 이므로(/usr/src/kernels/2.6.11-1.1369_FC4-i686/include/linux/nfsd/const.h 참고) rsize와 wsize를 4K, 8K, 16K, 32K에 대하여 각기 3회씩 실시하여 평균치를 구하였다. 또한 캐쉬를 지우기 위해, 매번 NFS 파일시스템의 마운트를 해제하고 다시 마운트하여 테스트하였다.

다음은 각기 블록 크기 16K로 1GB의 파일을 생성하고 읽는 명령의 수행시간을 구하는 명령의 예이며, 이런 명령을 사용하여 NFS의 성능을 테스트한 결과는 표 1과 같다.

```
# time dd if=/dev/zero of=/home2/testfile \
  bs=16k count=65536
# time dd if=/home2/testfile of=/dev/null \
  bs=16k
```

(표 1) NFS의 성능 측정 결과(단위:초)

Block size	Normal 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

※ 괄호 속은 TCP를 사용한 경우임

일반적으로 세 경우 모두 Write/Read 경우 NFS의 블록 크기가 커질수록 시간이 적게 걸리므로 더 효율적임을 알 수 있다. 한편 SSH 터널을 통한 경우는 전통적인 UDP를 사용하는 NFS에 비해, Write는 4.5-12.5%, Read는 1.4-5.4% 정도 시간이 더 걸렸으며, 블록 크기가 클수록 그 차이가 감소함을 보였다. 한 가지 이상한 점은, 보통의 NFS Read의 경우, TCP를 사용한 경우가 UDP를 사용한 경우보다 시간이 더 걸릴 것이 예상되었으나 실제의 실험 결과는 오히려 반대로 대략 1%

정도 감소한 결과가 얻어졌는데, 그 차이가 미미한 것으로 보아 오차 범위 안에 드는 것으로 보인다.

결론적으로 rsize와 wsize를 최대한 크게 설정하면 외부의 비전용 노드들이 자주 NFS를 통해 파일을 읽거나 쓰지 않는 한 SSH 터널을 이용한다 하더라도 터널링 오버헤드가 그리 크지 않을 것으로 생각된다.

5. 결론 및 향후 연구 방향

본 논문에서는 방화벽 외부의 네트워크 상에 존재하는 비전용 노드의 유휴시간을 활용하기 위해, 기존의 HPC 클러스터를 확장하는 경우 발생하는 NFS 등의 보안 문제를 SSH 터널링을 사용하여 해결하는 방안을 제시하고 암호화된 NFS의 성능을 실험하였다.

그러나 SSH 터널링을 통한 NFS의 경우는 HPC 클러스터의 사용자를 완전히 신뢰할 수 없는 경우 또 다른 심각한 보안문제를 야기할 수 있는데, NFS 서버를 별도로 두되 일반 사용자의 로그인을 허용하지 않음으로써 해결하는 방안을 보이고 보완된 확장 HPC 클러스터의 구조를 제안하였다.

응용 예 및 프로그래밍 방법에 따라 다를 수 있으나, 저자의 경험상 HPC 클러스터를 이용한 실제의 연산에서는 연산 시작 초기 이외에는 슬레이브 및 외부 노드가 직접 NFS 서버의 파일을 읽는 경우가 드물 것이며 대부분의 데이터는 매스터와 직접 주고 받고 매스터가 디스크에 기록하는 것이 일반적이다. 따라서 SSH 터널을 통한 약간의 지연은 그리 문제가 되지 않을 것으로 생각된다.

본 연구에서는 HPC 클러스터에서 주로 사용되고 많은 플랫폼에 의해서 지원되는 NFS를 그대로 사용하는 경우에 대한 것을 다루었다. 그러나 NFS 자체는 여전히 문제가 많으므로 Linux 상에서의 IPSec [11] 등을 이용하는 방안은 좋은 대안으로 생각되며, IOzone [7]과 같은 벤치마킹 테스트를 사용하는 것이 추후의 연구과제이다.

[참고문헌]

[1] 김영균, 오길호, “LAN 환경에서 유휴시간 예약에 기반한 PC Cluster 설계”, 「한국정보과학회 2003년도 가을 학술발표논문집」, 제30권 2호(III), 2003.

[2] 김원일, Open Ssh Tunneling, http://wiki.kldp.org/wiki.php/OpenSshTunneling

[3] 박필성, “LAN상의 비전용 노드를 포함한 HPC 클러스터의 확장에 의한 성능 향상”, 「한국IT서비스학회지」 제7권 제4호(2008), pp.209-219.

[4] AFS Frequently Asked Questions, http://www.angelfire.com/hi/plutonic/afs-faq.html

[5] Berkeley NOW Project, http://now.cs.berkeley.edu/

[6] Coda File System, http://www.coda.cs.cmu.edu/

[7] IOZone File System Benchmark, http://www.iozone.org/

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

[9] Linux NFS Overview, FAQ and HOWTO documents, http://nfs.sourceforge.net/

[10] Open MPI, http://www.open-mpi.org

[11] Openswan, http://www.openswan.org/

[12] PVM, http://www.csm.ornl.gov/pvm/

[13] Zaki, M., W. Li, and S. Parthasarathy, “Customized Dynamic Load Balancing in a Heterogeneous Network of Workstations”, In *5th IEEE Int. Symposium on High Performamce Distributed Computing,* 1996.

[14] http://www.linuxquestions.org/questions/linux-security-4/firewall-blocking-nfs-even-though-ports-are-open-294069/

[15] http://kr.blog.yahoo.com/thisrule1/857793 .html