Beowulf Installation and Administration HOWTO

Jacek Radajewski and Douglas Eadline Version 0.1.2 1999년 6월 2일

번역자: 양 유 성, yooseong@kldp.org 번역일: 2000년 3월 17일

이 문서는 Beowulf류의 수퍼컴퓨터를 만들고자 할 때 필요한 방향을 제시한다. 이 문서는 구조적인 몇몇 측면을 포함하지만 주로 운영체제의 설치와 유지에 초점을 맞추고 있다. 이 HOWTO는 diskless 클라이언트부분과 클러스터를 만들고 빠르게 실행시키고자 하는 목적이 있다. 이 HOWTO는 레드햇 리눅스 5.2와 커널 2.0.x의 내용을 포함하고 있다.

목 차

제 1 절 서문 **3**

1.1 Disclaimer . . . 3

1.2 저작권 . . . 3

1.3 이 HOWTO 문서에 관해 . . . 4

1.4 저자들에 관해 . . . 4

1.5 감사의 글 . . . 4

제 2 절 개요 **4**

제 3 절 Quick Start (disk-less clients) **4**

3.1 하드웨어의 설치와 점검 . . . 4

3.2 운영체제의 설치 . . . 5

3.3 Diskless-Clients의 점검 . . . 5

3.4 Diskless client 부팅 플로피 제작 . . . 5

3.5 Client를 위한 파일시스템 제작 . . . 5

제 4 절 Cluster 설계 **5**

4.1 디스크 . . . 6

4.1.1 Diskl-less Clients . . . 6

4.1.2 Fully local install . . . 6

4.1.3 표준 NFS 설정 . . . 6

4.1.4 분산 파일 시스템 . . . 7

4.2 메모리 . . . 7

4.2.1 Amount . . . 7

4.2.2 속도 . . . 7

4.3 CPU . . . 7

4.3.1 Type . . . 7

4.3.2 SMP (역자주: Symmetric MultiProcessing) . . . 8
4.4 Network . . . 8
4.4.1 Hypercube . . . 8
4.4.2 10/100 Mbps Switched Ethernet . . . 8
4.5 어떤 리눅스 배포판을 사용할 것인가? . . . 9
4.6 하드웨어의 구입 . . . 9

제 5 절 주노드의 설치 **10**
5.1 분할크기 . . . 10
5.2 레드햇 리눅스 설치 . . . 10
5.3 네트워크 설정 . . . 11
5.4 DNS 설정 . . . 11
5.5 z/etc/hosts . . . 12
5.6 z/etc/resolv.conf 만일 여러분이 서버를 DNS 서버로 사용하고 있다면 zresolve.conf파일은 지역 네임서버를 밝혀주어야한다. 이는 zTopcat에 있는 z/etc/resolv.conf에 있는 내용이다. . . . 12
5.7 z/etc/hosts.equivrsh을 클러스터의 어떠한 노드, 어떠한 사용자에게도 허용해주기 위해 여러분은 보안문제를 감수해야합니다. 그리고 모든 호스트의 리스트를 z/etc/hosts.equiv에 입력하여야 합니다. 보안문제는 11절을 참고하시기 바랍니다. . . . 13
5.8 Sample .cshrc . . . 13
5.9 시간동기화 . . . 13

제 6 절 클라이언트 설정 **14**
6.1 클라이언트 복제 . . . 14
6.2 disk-less 클라이언트 설정 . . . 14
6.2.1 문제해결 방안 . . . 16
6.3 클라이언트 콘솔접근 방법 . . . 16
6.4 각 클라이언트에 OS 설치 . . . 17

제 7 절 Compilers **17**

제 8 절 통신 소프트웨어 **17**
8.1 PVM VS MPI . . . 17

제 9 절 변환 소프트웨어: **18**

제 10 절 시스템 감시 소프트웨어 **18**
10.1 bWatch . . . 18
10.2 httpd와 CGI 스크립트 이용하기 . . . 18
10.3 Netpipe . . . 18
10.4 네트워크 퍼포먼스: netperf . . . 19

10.5 병렬 퍼포먼스: NASA 병렬 벤치마크 19

제 11 절 관리: CMS **19**

제 12 절 보안 **19**

12.1 서버 19

12.1.1 TCP wrapper 19

12.1.2 사용하지 않는 데몬 멈추기 - `z/etc/inetd.conf` 21

12.1.3 rc 스크립트를 이용하여 서버멈추기 21

12.1.4 `zipfwadm` 21

12.2 클라이언트 22

12.2.1 .rhosts versus hosts.equiv 22

12.2.2 root rlogin 접근: 22

12.2.3 root telnet 접근 23

12.2.4 root ftp 접근 23

제 13 절 이더넷 채널본딩: **24**

제 14 절 스위치 Configuration: **24**

제 15 절 참고문헌과 다른 정보출처들 **25**

제 16 절 `zsdct` 스크립트 **25**

제 17 절 `zadcn` 스크립트 **25**

제 18 절 `zfirewall`스크립트 **26**

제 19 절 변화들(버전 0.1.0에서 시작된) **28**

제 1 절 서문

1.1 Disclaimer

우리는 이 문서내의 어떠한 잘못된 정보에 대해서 그 것이 발생시킬 손실에 대해 책임을 지지 않는다

1.2 저작권

Copyright (C) 1997-1999 Jacek Radajewski. Copyright(C) 1999 Paralogic, Inc., 115 Bethlehem PA, 18015 (`<http://www.plogic.com>`)

이 문서는 GNU GENERAL PUBLIC LICENCE Version 2(1991 Copies of licence)에 의해 배포되며 이 라이센스는 `<http://www.fsf.org/copyleft/gpl.html>`에서 볼 수 있다.

1.3 이 HOWTO 문서에 관해

Jacek Radajewski는 1997년 11월에 Beowulf HOWTO를 쓰기 시작했고 Douglas Eadline이 가세를 했다. 몇달동안 Beowulf HOWTO는 많은 양의 문서가 되었고 1998년 8월에는 세부분으로 나누어졌다: Beowulf HOWTO, Beowulf Architecture Design HOWTO, Beowulf Installation and Administration HOWTO. Beowulf Installation and Administration HOWTO의 Version 1.0.0의 문서는 Linux Documentation Project에 곧 발표될 것이다.

1.4 저자들에 관해

- Jacek Radajewski는 네트워크 관리자로 일을 하고 있으며, 호주 Southern Queensland 대학에서 전산학 학위를 받기위해 공부하고 있다. Jacek이 리눅스를 처음 접한 것은 1995년 이었고 그 이후로 리눅스에 흠뻑 빠졌다. Jacek은 1997년 5월 처음으로 그의 Beowulf 클러스터를 제작했으며 그 이후로 꾸준히 이 클러스터 기술에 관해 연구하면서 클러스터를 위한 더 나은 방법을 찾고 있다.
- Douglas Eadline 박사는 미국 PA주 Bethlehem에 있는 Paralogic이라는 회사의 사장이자 수석 연구원이다. 물리/분석 화학자인 그는 화학 분석장비를 위해 1978년 single board 컴퓨터를 제작한 이후로 컴퓨터와 관련된 일을 하고 있다. Eadline 박사의 관심은 리눅스와 Beowulf 클러스터, 병렬 알고리즘이다. Eadline 박사는 deadline@plogic.com을 통해 연락을 주고 받을 수 있다.

1.5 감사의 글

우선 저는 이 HOWTO문서에 여러 도움을 주신 다음 분들께 감사를 드립니다: Rebecca Cox, Thomas Sterling, Donald Becker, Daniel Ridge, Thanh Tran-Cong, Christopher Vance, Ron Addie, Russell Waldron, David Smith와 나에게 많은 조언을 아끼지 않았던 Beowulf mailing list의 다른 많은 분들에게도 감사 드립니다.

제 2 절 개요

다음의 내용들은 다른 많은 분들의 작업한 내용을 바탕으로 Beowulf 구조와 설계, 제작, 성능확인에 대해 다룬다. 이 문서는 모든 설계와 성능확인의 측면을 다룰 수 없지만 Beowulf 클러스터를 처음 제작하여 사용하는 숙련되지 않은 리눅스 관리자에게는 충분한 정보를 줄 수 있다.

제 3 절 Quick Start (disk-less clients)

만일 여러분이 저만큼이나 참을성이 없다면, 여러분은 곧바로 시작하길 원할 것이고 이 섹션을 읽을 것이다. 이 섹션에서 다루는 내용은 모든 것을 빨리 하고자 할 때 필요한 내용을 기술하고 있다. 더 자세한 내용은 그 다음 섹션에서 발견할 수 있을 것이다.

3.1 하드웨어의 설치와 점검

우선 모든 컴퓨터(노드)와 스위치, 그 밖의 다른 하드웨어들을 박스에서 꺼낸다. 여러분은 모든 노드의 BIOS를 확인할 필요가 있다. 만일 여러분이 단지 하나의 비디오 카드와 한개의 키보드를 갖고 있다면, 각 노드를 분리하여 비디오 카드와 키보드를 연결하고 각 노드를 작동시켜 BIOS를 확인한다. 확인할 필요가 있는 일반적인 설치는 IDE 하드 디스크이며 키보드와 비디오 카드가 문제를 일으킬 때 멈추는지를 확인한다. 만일 여러분의 노드들이 키보드와 비디오 카드를 갖고 있지 않다면, BIOS에서 이를 감지하지 못했을 경우 멈추지 말아야한다. 모든 노드들에 전원을 연결하고 노드와 스위치 사이의 이더넷 케이블을 확인한다.

3.2 운영체제의 설치

서버 노드에 RedHat 5.2를 설치한다.(역자주: 원작자가 Beowulf를 사용할 당시는 RedHat 5.2 버전이 일반적으로 사용되었다.) RedHat 리눅스 5.2의 설치는 5.2절을 참조하면 된다. 모든 클라이언트 노드의 NFS-root 파일 시스템을 위해 root 파티션에는 충분한 공간을 확보한다. 물론 모든 클라이언트 노드들은 서버의 syslogd 서버에 그들의 log를 기록할 것이며 서버의 `/var/log`는 그 자신의 log뿐만 아니라 클라이언트의 log도 기록할 저장 공간을 필요로 하게 됩니다. `/var`,`/lib`,`/bin`, `/sbin`과 `/etc` 디렉토리는 disk-less 클라이언트 형성을 위해 분리되어서는 안되고 동일 파티션에 설치되어야 한다. 만일 여러분이 위의 것들중 하나를 다른 파티션에 설치하면 NFS-root 파일시스템의 템플릿을 형성하는데 사용되는 `sdct` 스크립트는 적절한 하드링크를 할 수 없게 된다. 모든 네트웍 디바이스들과 파일시스템은 지원된다. 여러분은 **RARP** 요청에 **RARP** (`CONFIG_INET_RARP`) 지원이 필요할 것이다. RARP는 RedHat 리눅스 5.2 배포본의 커널에서 지원된다. (역자주: NFS-root 파일시스템에 관해서는 `<http://kldp.org>`에서 관련 문서를 찾아보면 알 수 있음.)

3.3 Diskless-Clients의 점검

템플릿 디렉토리(대개는 `/tftpboot/Template`)를 만들기위해 서버에서 `sdct`(15절의 sdct) 스크립트를 실행한다. 이 템플릿은 클라이언트의 / 파일시스템으로 쓰일 것이다. 여러분의 필요에 의해서 템플릿에 약간의 수정을 가할 수도 있을 것이다. 스크립트를 변형하고서 그 변화된 내용을 쉽게 확인할 수 있을 것이다.

3.4 Diskless client 부팅 플로피 제작

클라이언트를 위해서 NFS-root 부트 플로피를 컴파일한다. 가장 손쉬운 방법은 클라이언트를 위해 단일 커널(monolithic kernel)을 만드는 것이다. 그리고 나서 NFS-root 파일 시스템을 컴파일한다.(이다음번의 HOWTO에서는 net-booting을 위한 모듈 커널의 사용방법에 대해 기술할 것이다.) 다음의 선택사항에서 'y'를 입력한다:`CONFIG_ROOT_NFS`, `CONFIG_RNFS_BOOTP`, `CONFIG_RNFS_RARP`

커널을 컴파일하고 난 후, 루트 디바이스 NFS-root를 변화시키고 나서 dd를 이용 플로피 디스크에 저장한다.

```
mknod /dev/nfsroot b 0 255
cd /usr/src/linux/arch/i386/boot
rdev zImage /dev/nfsroot
```

3.5 Client를 위한 파일시스템 제작

`adcn`스크립트를 클러스터의 새로운 노드로 서버에 넣기 위해 실행시킨다. 가장 쉬운방법은:

```
adcn -i 10.0.0.2 -c node2 -d my.beowulf.domain -l -D eth1
```

`eth1`는 클러스터에 연결된 인터페이스를 의미한다. 이제 NFS-root 커널 플로피를 클라이언트 드라이프에 집어넣고 시스템을 다시 시작한다.

제 4 절 Cluster 설계

이 문서의 목적이 Beowulf 클러스터의 설치와 관리를 다루고 있지만, 설치하기 전에 클러스터 설계와 관계된 몇가지 점들을 살펴보고 몇몇 설계에 관한 결정을 하는 것이 중요하다. 여러분은 CPU의 선택과 속도 서버와 클라이언트 노드들을 위한 메모리 크기, 디스크 크기 등을 살펴보아야 할 것이다. 본 저자는 저자가

생각하고 있는 것이 최상이라고 여기는 것을 상세히 서술할 것이고 변형된 설계에 대해서도 간략히 살펴볼 것이다.

4.1 디스크

Beowulf 클러스터에서 디스크 공간의 설정방법에는 적어도 네가지가 있다. 이러한 설정들은 가격과 성능, 관리면에서 차이를 보이고 있다. 이 문서는 저자가 선호하는 disk-less 클라이언트 설정을 다룰것이다.

4.1.1 Diskl-less Clients

이러한 disk-less 클라이언트 설정에서는 서버가 모든 파일들을 disk-less 클라이언트들에게 제공한다. disk-less 클라이언트 시스템의 가장 큰 장점은 새로운 노드를 추가하고 클러스터를 관리하는데 편하다는 점이다. 클라이언트 노드들이 어떠한 정보들도 갖고 있지 않기 때문에 새로운 노드를 추가할 경우, 여러분은 서버에서 몇몇 파일들을 변형시키고 이러한 것들이 작동되게 하면 될 것이다. 여러분은 클라이언트 노드들에 운영체제나 그밖의 다른 소프트웨어도 설치할 필요가 없다. disk-less 시스템의 단점이라고 말할 수 있는 것은 증가된 NFS traffic을 자동적으로 처리하게 하는 스크립트를 만든다 하더라도 초기 셋업을 더욱 복잡하게 만든다는 것이다. 만일 여러분이 disk-less 클라이언트 시스템을 선택한다면 여러분은 플로피 디스크나 부트롬(boot-ROM) 이더넷카드가 필요할 것이다. 이번 절에서 기술되는 다른 관리설정으로부터 이 disk-less 클라이언트 설정은 Beowulf HOWTO에서 정의된 Beowulf 구조에 가장 가까운 것이다. 만일 여러분이 여분의 돈이 있고 각 클라이언트 노드를 위한 디스크를 구입하길 원하다면, 여러분은 기존의 disk-less 클라이언트 디스크 저장 설정을 사용할 수 있지만, 클라이언트 노드의 디스크에 운영체제를 복사하는데 필요한 스크립를 사용해야한다. 이는 disk-less 클라이언트 설정에 유연성을 줌과 동시에 NFS traffic과 지역 스왑영역을 줄일 수 있다.

어떻게 disk-less 클라이언트 노드가 부팅되는가?

disk-less 클라이언트 설정에서 클라이언트 노드들은 자기자신에 대해서 어떠한 것도 알지 못한다. 그러면 클라이언트 노드가 작동을 하고 무엇을 해야하는지 어떻게 인식을 하는가? 그럼 여기서 Beowulf 클러스터에서 작동되는 새로운 노드를 가지고 예를 들어보자. 먼저 전원을 켰을 때, 클라이언트 노드는 플로피 디스크나 이더넷 카드의 EPROM에서 구동된다. 그리고나서 자신의 누구인지를 말하는 IP주소를 요청할 때 필요한 RARP (Reverse Address Resolution Protocol)를 보낸다. 그러면 서버노드는 IP 주소를 알려주거나 "당신의 이름은 node64이고 주소는 10.0.0.64입니다."라고 알려준다. 그 새로운 노드는 계속해서 부팅과정을 실행하고 자신의 이더넷 인터페이스 설정을 하며 서버에서 제공되는 경로를 통해 NFS 파일시스템을 마운트한다. 루트의 파일시스템은 RAM 디스크에 마운트될 수 있지만, 대부분의 경우 NFS 파일시스템으로 마운트된다. 클라이언트 노드가 부팅과정에서 해야할 마지막 작업들중에 하나는 서버노드에게 자신이 작동을 할 수 있다는 것을 알려주는 것이다. 서버노드는 이러한 정보를 기록하고 새로운 클라이언트를 계산에 사용할 수 있게된다. 이때부터는 클라이언트 노드가 서버에 의해 제어되며 실행해야할 것을 시행한다.

4.1.2 Fully local install

또다른 극단적인 방법은 각 클라이언트에 모든 것을 저장하는 것이다. 이러한 설치의 장점은 어떠한 NFS traffic도 일어나지 않는다는 것이고, 단점은 매우 복잡한 설치와 유지가 필요하다. 이러한 설정의 유지는 복잡한 쉘 스크립트와 모든 파일 시스템을 향상시킬 수 있는 rsync와 같은 유틸리티를 사용하면 좀더 쉽게 할 수 있다.

4.1.3 표준 NFS 설정

세번째 방법은 disk-less 클라이언트와 fully local install 설정의 중간이라고 할 수 있는데 클라이언트는 자기자신의 운영체제를 갖는 하드디스크를 갖고 있으며 스왑또한 설정 되어있고 단지 서버노드에 `/home`과 `/usr/local/`를 마운트 시키면 된다. 클라이언트 노드에 운영체제를 설치하는 방법은 여러가지가 있다. 이것에 관해서는 나중에 상세히 설명할 것이다.

4.1.4 분산 파일 시스템

분산 파일 시스템은 많은 노드에 존재하는 파일 시스템이다. 여러형태의 분산 파일 시스템이 있으며 몇몇은 리눅스로 포팅되어있다. 이러한 파일 시스템에서의 작업은 매우 실험적이어서 나는 여기서 이것에 관해 상세히 다루지 않겠다. 만일 여러분이 여러분의 Beowulf 시스템에서 분산 파일 시스템을 사용하고자 한다면 *Implementation and Performance of a Parallel File System for High Performance Distributed Applications* <http://ece.clemson.edu/parl/pvfs/pvfshpdc.ps>*의 자료를 읽어보고 해보면 될 것이다. (역자주: Parallel Virtual File System에 관한 내용은* <http://www.beowulf-underground.org>*에서도 확인할 수 있다.)*

4.2 메모리

4.2.1 Amount

적절한 메모리 용량의 선택은 Beowulf 시스템의 설계에 있어서 가장 중요한 작업중의 하나이다. 만일 여러분이 여러분이 실행할 작업들을 저장할 공간이 충분히 없다면, 여러분은 확장된 스왑핑으로 인해 여러분 시스템의 성능을 저하시킬 것이다. 스왑핑은 여러분이 원하는 것이 아니다. 하드디스크에서 읽히는 모든 페이지는 여러분에게 상당한 실행시간을 요구할 것이다. 하드디스크에서 읽어들이는 것은 RAM으로 부터 읽어드리는 것에 비해 상당히 느리다. Sparc 서버에서 돌아가는 아주 커다란 작업을 본 적이 있는데 wall clock time의 99.5%가 하드디스크에서 읽고 저장하는데 사용되고 나머지 0.5%만이 실제 계산에 사용되었다. 이상적으로는 여러분이 스왑핑을 원하지 않지만, 여러분은 계획보다 큰 작업을 실행할 경우를 대비하여 어느 정도의 스왑 공간을 마련하여 두는 것이 나을 것이다.

4.2.2 속도

여러분의 메모리 속도는 매우 중요하다. 만일 빠른 버스에서 작동하는 빠른 CPU를 선택했다면, 노드간에 메모리 병목현상의 가능성이 아주 많다. 여기서 16ns SDRAM 쓰기를 권장한다.

4.3 CPU

4.3.1 Type

CPU의 선택은 두가지 부류중에서 이루어져야한다: Intel x86 호환과 DEC Alpha 시스템. 다른 CPU들도 리눅스가 지원을 하지만 인텔과 알파 시스템을 제외한 다른 시스템을 이용, Beowulf 시스템을 만드는 것은 나로서는 알지 못한다. 일반적으로, Intel기반의 시스템들은 확장시스템으로 여겨지는데 이는 다양한 소스(Intel, AMD, Cyrix)가 있고 쉽게 구할 수 있기 때문이다. 이와는 달리 DEC Alpha는 성능면에서는 탁월하지만 한정된 소스(DEC)여서 경제적인 가격으로 구하기가 조금 힘들다.

Intel의 "slot 기반" 시스템들이 제한된 소스라는 말이 나오긴 하지만, 아직 시장에서 이 시스템에 관한 것은 아직 결정된 것이 없다.

Intel 기반의 시스템들중에서, Pentium Pro와 PII는 최상의 부동소수 연산능력을 보이고 있고 SMP motherboard를 지원하는 유일한 것이다. 어떤 CPU를 사용할 것인지에 관한 논쟁(1998년 말경에 끝난)-PII는 최상의 클럭 스피드의 반으로 작동을 하고 Pentium Pro는 최상의 클럭 스피드로 작동-이 있다. 대개는 SDRAM를 갖는 PII가 Pentium Pro와 동일한 클럭 스피드로 동작한다고 알려졌다. 여러분의 평가는 달라질 수 있지만, PII의 클럭 스피드가 333MHz로 다가감에 따라 대부분이 PII를 택하였다. 여기에 따른 논쟁은 다음 사이트에서 확인할 수 있다: <http://www.tomshardware.com/iroadmap.html> <http://www.compaq.com/support/techpubs/whitepapaers/436a0597.html>

4.3.2 SMP (역자주: Symmetric MultiProcessing)

대칭 다중프로세서 보드는 Beowulf 클러스터에서 일반적으로 많이 쓰인다. 주된 장점으로는 가격 대 성능 면에서 앞서고 동일한 보드에서 두 프로세서간의 빠른 통신이 이루어진다. 만일 여러분이 아주 큰 클러스터를 제작하고 싶다면 이는 매우 중요한 점이다. 전체 클러스터에서 이중 CPU를 사용함으로써 여러분은 네트웍 카드와 케이스, 전력공급기, 보드의 수를 반으로 줄일 수 있다. 유일하게 비싼것은 SMP 보드인데 다른 비용의 절감이 이를 극복한다.

만일 여러분이 보드당 한개의 CPU만을 사용한고자 결정한다 하더라도, SMP 서버를 구입하는 것이 가치있는 일일 것이다. 우리의 *Topcat* 시스템은 세사람이 사용한는데 주노드(master node)에서 사용자들이 그들이 만든 코드를 편집하고 컴파일하고 시험해본다. 주노드의 두개의 CPU를 이용하여 로드평균이 2이상에서 작동하는 것이 일반적이다. (역자주: top명령을 이용하면 로드평균이 나온다.) 주노드는 클라이언트 노드들에게 파일 시스템을 제공해야하기 때문에, NFS 서버는 충분한 CPU 주기를 가지고 그 작업을 수행하는 것이 중요하다. 만일 여러분이 여러분 서버노드가 사용자들에의해 부하가 걸린다면, 여러분은 빠른 SMP의 선택을 고려해야한다.

4.4 Network

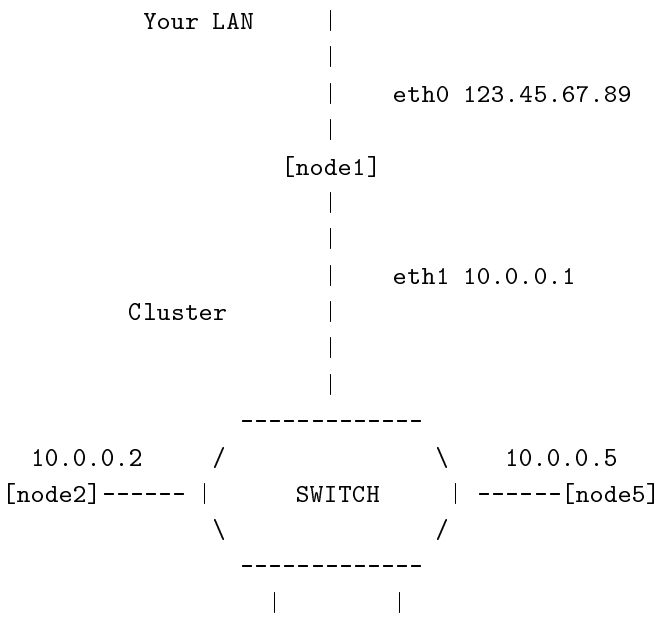
4.4.1 Hypercube

Hypercube는 노드와 그 경계를 연결한 네트워크 위상(topology)이다. 100Mbps 네트워크 스위치의 가격하락으로 인해 hypercube는 더이상 경제적인 네트워크 위상이 아니라고 할 수 있다.

4.4.2 10/100 Mbps Switched Ethernet

100 Mbps로 변경된 완전한 2중 이더넷은 Beowulf 시스템의 네트워크로 가장 일반적으로 사용되고, 완전한 망상조직의 네트워크와 거의 동일한 성능을 발휘한다. 네트워크에 연결된 모든 컴퓨터들이 외부와의 접속을 위해 경쟁하고 정보 패킷의 충돌을 일으키는 것과는 달리, 스위치 이더넷은 스위치에 연결된 어떠한 두개의 노드 사이에서도 충분한 대역폭을 제공한다. 여러분의 클러스터를 위해 빠른 이더넷 네트워크 카드를 구입하기 전에, 여러분은 먼저 리눅스 네트워크 드라이버를 다음 사이트에서 확인을 해보는 것이 좋을 것이다. `<http://www.alternic.net/nic/rfcs/1900/rfc1918.txt.html>`

간단한 예로, 다섯개의 노드로 이루어진 Beowulf 클러스터는 다음과 같이 나타낼 수 있다.



```
                |         |
                |         |
        10.0.0.3  |         |  10.0.0.4
              [node3]      [node4]
```

4.5 어떤 리눅스 배포판을 사용할 것인가?

Beowulf 시스템에서 많이 쓰이는 배포판은 레드햇 리눅스이다. 설치가 간단하고 레드햇 FTP 서버 `<ftp://ftp.redhat.com>` 이나 다른 미러사이트에서 쉽게 구할 수 있다. 현재의 이 문서는 레드햇 5.2 배포판에 기준하고 있다. 만일 여러분이 데비안이나 슬랙웨어 또는 다른 배포본을 사용하고 있다면 본 저자와 다른 방법으로 할 수 있다.

레드햇 리눅스의 중요한 장점중의 하나는 RPM (RedHat Package Manager)를 이용하여 모든 꾸러미를 설치하고 업그레이드하고 제거하는데 있어서 편리하게 사용할 수 있다는 점이다. PVM (역자주:Parallel Virtual Machine) 과 MPI (역자주: Message Passing Interface)와 같은 소프트웨어와 Beowulf 커널과 같은 것도 RPM 형태로 얻을 수 있다.

NOTE: Extreme 리눅스 CD의 원본은 매우 오래되었다. 그 CD에 들어있는 문서들을 제외한 RPM들은 사용해서는 안된다.

4.6 하드웨어의 구입

다음 내용은 어떠한 하드웨어의 구입에도 적용된다. Beowulf를 제작할 때에는 하드웨어 구입이 중복되지 않게 하여야 한다.

상용 하드웨어의 문제점:

비용을 절감하는 좋은 방법 - 166MHz CPU를 구입해서 233MHz로 표시를 바꾼 후, 몇백 달러를 더 받고 팔아라. 또는 낮은 품질의 DRAM을 좋은 품질의 DRAM으로 팔아라. 사업이 상업적 성격 때문에, 사용자들은 "plug and play" 구성을 기대한다. 불행하게도, 233MHz로 변경을 해서 벌 수 있는 돈은 부정직한 하드웨어 판매업자에게는 커다란 기회인 것이다. 돈을 벌 수 있는 또한가지 방법은 품질이 낮은 축전기를 포함하는 메인보드를 제작하는 것이다. 이름이 없는 보드들은 종종 이러한 부품들을 포함하고 있다. 제작은 보드 당 가격을 20-30 달러를 아낄 수 있지만 1, 2년 후면 이 보드는 쓸모없게 된다.

CPU를 오버클럭을 할 수 있고 싼 RAM으로 동작할 수 있고 싼 보드가 잠시동안 작동할 수 있지만, 이 부품들은 작동하지 않을 수 없고 문제를 일으킬 수 있다. 이러한 종류의 부품들의 구입은 여러분이 제품을 다시 하드웨어 판매업자에게 되돌려 주었을 때 판매업자가 유지해야하는 전체 비용의 15%가 여러분에게 할당될 수도 있으며 하드웨어 중복되는 하드웨어 문제를 야기시킬 수 있다.

해결책:

우선 타당한 하드웨어 가격인지를 살펴본다. 소규모 하드웨어 판매업자들이 수년동안 사업을 하지 않았고 다른 모든 요구사항들을 만족시키지 않는 다면 그러한 제품들을 구입하지 않는 것이 좋다. 둘째로, 하드웨어 판매업자에게 적어도 다음 세가지의 사항을 강조하라.

1. CPU와 DRAM에 대해서 3-4년의 품질보증을 확인받으라. 영원한 품질보증이 더욱 좋지만 실제로는 3-4년 후엔, 부품이 생산될지도 모르거나 여러분이 신경을 쓰지 않을지도 모르기 때문에 3-4년이 적당하다. 좋은 품질의 제품을 판매하는 업자라면 품질보증을 할 것이다.

2.만일 하드웨어에 문제가 발생하였을 때 수리비용의 15%를 여러분이 부담해야하는 품질이 나쁜 하드웨어를 판매하는 판매상과 거래하지 마라.

3.하드웨어에 어떠한 문제가 발생하였을 때 그 문제를 해결할 수 있는 기술자가 있는지 판매업자에게 물어봐라. (권위있는 Intel 기술자는 각각의 개인번호를 갖고 있다.) 만일 그렇지 않다면 판매상들은 부품의 정확한 출처를 모르는 것을 판매하게 될 것이다.

마지막으로, 이름없는 메인보드나 비디오카드 네트워크 제어기등의 복제품들을 구입하지 마라. 몇푼의 돈을 아끼면 나중에 수리비가 더 들어갈 것이다. 사실, Beowulf 클러스터의 경우 비용을 절약할 수 있기 때문에 굉장한 매력을 갖고 있다.(즉, 이름없는 NIC 카드를 55달러에 구입할 것인가? 아니면 이름있는 NIC카드를 75달러에 구입할 것인가?)

하드웨어를 구입하는 경우 고려해야할 몇가지 것들이 있다. PC 시장은 표준화된 제품을 생산하고 경쟁을 하지만, 이는 또한 질낮은 부품들이 잘못 알려지고 팔리게 할 수 있다. 구입자여 현명해져라

제 5 절 주노드의 설치

여러분이 직접묻는 질문으로는 무엇이 주노드 서버인가 하는 것이다. 대부분의 Beowulf 시스템들은 단지 하나의 서버와 클러스터 외부로 접속하는 게이트웨이 하나를 갖지만 몇몇의 경우는 성능과 신뢰도의 차원에서 다중 서버를 갖고 있다. 큰 disk-less 클라이언트 클러스터의 경우, 클라이언트 노드들에게 시스템 파일을 제공하기 위해 다중 NFS서버를 사용하고자 할 것이다. 더욱더 분산된 환경에서는 모든 노드가 클라이언트와 서버로 작동하는 것이 가능하다. 만일 여러분이 단지 하나의 서버를 사용하고자 한다면, 여러분은 '주(master)'라는 말을 생략할 수 있고 주서버를 그냥 서버라고 생각할 수 있다.

주서버는 여러분의 Beowulf 시스템에서 가장 중요한 노드일 것이다. 클라이언트 노드들에게 NFS 파일시스템을 제공하고 소스코드를 컴파일하며 병렬처리를 시작하고 외부로부터의 연결을 가능하게 한다. 다음의 내용들은 주서버를 설치하고 설정하는 단계를 설명하고 있다.

5.1 분할크기

설정과정중 가장중요한 부분은 분할크기를 선택하는 것이다. 여러분의 클러스터가 작동을 시작하기 시작할 때에는 분할설정을 하기가 매우 힘들기 때문에 여러분이 얼마만큼의 분할이 필요한지 선택하는 것이 중요하다. 본 저자는 이 문서를 갱신할 때마다 아래 열거한 분할영역을 변경하였다. 여러분이 직접 시험해보았음에 틀림없겠지만, 다음의 크기는 4GB의 HDD와 레드햇 5.2, 16 노드, disk-less 클라이언트 클러스터에는 문제가 없다. 다음의 목록은 여러분의 파일을 저장할 수 있는 `/home` 는 제외한다.

- `/` - 500MB. / 분할은 `/bin`, `boot`, `/dev`, `/etc`, `/lib`, `/root`, `/sbin`, `/var`와 `/tftpboot` 디렉토리와 그 내부의 파일들을 포함한다. 대부분의 경우 `/tmp`또한 /에 있다. disk-less 클라이언트 설정에 있어서는 `/tftpboot`가 /와 동일 분할영역에 있는 것이 매우 중요하다.만일 우리가 분리된 분할영역에 두개의 디렉토리를 마운트시키고 싶다면, 우리는 작업할 NFS root 설정에 필요한 하드링크를 생성할 수 없다.

- `/usr` - 1.5GB. 이는 양이 많이 보일지 모르지만 대부분의 rpm이 `/usr/local`이 아닌 `/usr`에 저장된다는 것을 상기하자. 만일 대부분의 꾸러미르를 설치하고 싶다면, `/usr`에 더 많은 영역분할을 하라. 프로그램 실행시 디스크 공간 부족보다 더욱 나쁜 것은 없다.

- `/usr/local` - 500MB에서 2GB까지. 정확한 크기는 배포본에 있는 프로그램이 아닌 다른 곳에서부터 첨가할 소프트웨어가 얼마나 큰지에 달려있다.

- `swap` - 스와핑은 여러분 시스템 성능에 좋지 않다. 불행히도, 서버가 아주 커다란 작업을 수행하고 있고, 충분한 메모리 확보가 힘든 경우만 사용하도록 한다. 여러분은 아마도 RAM 크기의 2배 이상 크게 잡지 않아야한다. 예를 들어서 저자의 시스템의 경우, 노드1에 384MB의 RAM과 4개의 128MB 스왑을 설정했다.

5.2 레드햇 리눅스 설치

여기서는 레드햇 리눅스 5.2 설치를 `<http://www.redhat.com/support/docs/rhl/>`과 같이 상세하게 기술하지는 않을 것이다. 모든 레드햇 5.2 꾸러미를 다 설치할 것을 권한다. 만일 디스크 공간이 충분치 않다

면, 개개의 꾸러미를 선택하는데 많은 시간을 할애하지 말고 사용하지 않을 것같은 꾸러미는 설치하지 않고 넘어간다. 마치 Linux HOWTO 문서 번역처럼 말이다.

5.3 네트워크 설정

우선 이더넷 카드 설정이 필요하다. 여러분의 이더넷카드중의 하나가 진짜 IP 주소를 여러분에게 할당해 주어야 하며, 다른 하나는 클러스터 내부의 노드들에게만 보이는 사설 IP (예를 들면 10.0.0.1)를 갖고 있어야 한다. 여러분은 레드햇 배포본에 들어있는 GUI를 이용한 네트워크 설정 방법을 이용할 수도 있고 `/etc/system/network-scripts/ifcfg-eth*` 파일들을 만들고 편집하여 설정할 수도 있다. (역자주: 레드햇 배포본의 경우 `/etc/sysconfig/network-scripts/`에서 위의 파일들을 발견할 수 있다.) 단순한 Beowulf 시스템의 경우는 10/8 사설 IP 주소 범위를 사용하는데, 10.0.0.1은 서버에게 할당되고 나머지 10.0.0.2부터 10.0.0.254까지는 클라이언트 노드에 할당된다. 이러한 IP 주소를 사용하기로 결정한 후, netmask는 255.255.255.0로 10.0.0.255는 broadcast 주소로 사용한다. 저자가 사용하는 *Topcat* 시스템에서는 `eth0`를 외부와 연결된 인터페이스로 사용하고 `eth1`를 클러스터 내부의 네트워크와 연결하는 인터페이스로 사용했다. 라우팅 테이블은 다음과 같다.

```
[jacek@topcat jacek]$ /sbin/route
Kernel IP routing table
Desitination    Gateway         Genmask         Flags   Metric  Ref     Use     Iface
10.0.0.0        *               255.255.255.0   U       0       0       9       eth1
139.x.x.0       *               255.255.248.0   U       0       0       7       eth0
127.0.0.0       *               255.0.0.0       U       0       0       2       lo
default         139.x.x.1       0.0.0.0         UG      0       0       18      eth0
```

5.4 DNS 설정

본 저자의 Beowulf 시스템인 *Topcat*에서는 DNS를 작동시키지 않는다. 단순화된 Beowulf 클러스터 관리를 위해 DNS를 도메인을 갖는 것이 좋다고 생각했지만, DNS없이 *Topcat*을 설정하고 나서 작동이 더 잘되는 것 같았다. 설정은 전적으로 여러분에게 달려있다. 저자는 DNS에 대한 이부분을 참고적으로만 남겨놓을 것이고 더이상 관리는 하지 않을 것이다. 저자는 저자의 DNS 설정 파일들이 `named`의 최신 버전을 가지고 작동하지 않는다.

DNS 설정은 매우 직접적이다. 여러분의 서버(node1)이 DNS 서버가 될것이다. 서버가 이름들을 결정할 것이고 전체 Beowulf 클러스터를 위한 IP주소를 결정할 것이다. DNS 설정 파일들은 `<ftp://ftp.sci.usq.edu.au/pub/jacek/beowulf-utils>`에서 얻을 수 있다. 설정파일들은 *Topcat*시스템에서 사용했던 목록이고 여러분이 여러분의 시스템에 동일 이름을 사용하고 싶다면 그것들을 포함시킬 수 있다. 여러분이 봐서 알다시피 저자의 경우 사설IP 주소의 범위를 10.0.0.0/8로 잡고 있고 서브넷마스크를 255.255.255.0으로 설정해 놓았다. 누군가가 node1을 네임서버로 사용하지 않는다면 저자의 도메인은 외부에서 볼 수 없으며 이것이 우리가 원하던 바이다. 본 저자의 경우 도메인 이름을 `beowulf.usq.edu.au`로 결정하였고 여러분이 DNS 설정을 위해 변경해야할 설정파일들은 거의 없을 것이다. 앞서 언급한 `<ftp://ftp.sci.usq.edu.au/pub/jacek/beowulf-utils>`에서 얻을 수 있다. 설정파일들을 설치하고 `/etc/rc.d/init.d/named restart`를 실행함으로써 `named`를 재시작 할 수 있다.

여러분 DNS 서버:

```
[root@node1 /root]# nslookup node2
Server: node1.beowulf.usq.edu.au
Address:        10.0.0.1
<p>
Name:   node2.beowulf.usq.edu.au
```

```
Address:        10.0.0.2
<p>
[root@node1 /root]# nslookup 10.0.0.5
Server: node1.beowulf.usq.edu.au
Address:        10.0.0.1
<p>
Name:   node5.beowulf.usq.edu.au
Address:        10.0.0.5
```

5.5 /etc/hosts

만일 여러분이 DNS 서버를 사용하고 싶지 않다면 여러분은 모든 노드와 각 노드의 IP 주소를 /etc/hosts에 입력하여야 한다. 만일 disk-less 클라이언트 설정의 경우, sdct 스크립트와 adcn 스크립트를 이용 이 파일에 하드링크를 만들 수 있고 이를 모든 노드가 사용할 수 있다. 더욱이 adcn스크립트는 root 파일 시스템을 만들어 놓으려는 클라이언트를 위해 /etc/hosts에 입력시켜준다. 다음의 예는 *Topcat*에 나타난 /etc/hosts파일의 예이다.

```
127.0.0.1               localhost       localhost.localdomain
139.x.x.x               topcat.x.x.x    topcat
10.0.0.1                node1.beowulf.usq.edu.au        node1
10.0.0.2                node2.beowulf.usq.edu.au        node2
10.0.0.3                node3.beowulf.usq.edu.au        node3
10.0.0.4                node4.beowulf.usq.edu.au        node4
10.0.0.5                node5.beowulf.usq.edu.au        node5
10.0.0.6                node6.beowulf.usq.edu.au        node6
10.0.0.7                node7.beowulf.usq.edu.au        node7
10.0.0.8                node8.beowulf.usq.edu.au        node8
10.0.0.9                node9.beowulf.usq.edu.au        node9
10.0.0.10               node10.beowulf.usq.edu.au       node10
10.0.0.11               node11.beowulf.usq.edu.au       node11
10.0.0.12               node12.beowulf.usq.edu.au       node12
10.0.0.13               node13.beowulf.usq.edu.au       node13
```

5.6 /etc/resolv.conf 만일 여러분이 서버를 DNS 서버로 사용하고 있다면 resolve.conf파일은 지역 네임서버를 밝혀주어야한다. 이는 Topcat에 있는 /etc/resolv.conf에 있는 내용이다.

```
search beowulf.usq.edu.au eng.usq.edu.au sci.usq.edu.au usq.edu.au
nameserver      127.0.0.1
nameserver      139.x.x.2
nameserver      139.x.x.3
```

만일 DNS 서버를 갖고 있지 않다면 여러분은 다른 네임서버를 지정해주어야한다. 이는 저자의 현재 /etc/resolv.conf파일이다.

```
search eng.usq.edu.au sci.usq.edu.au usq.edu.au
nameserver      139.x.x.2
nameserver      139.x.x.3
```

5.7 /etc/hosts.equiv rsh을 클러스터의 어떠한 노드, 어떠한 사용자에게도 허용해 주기 위해 여러분은 보안문제를 감수해야합니다. 그리고 모든 호스트의 리스트를 /etc/hosts.equiv에 입력하여야 합니다. 보안문제는 11절을 참고하시기 바랍니다.

5.8 Sample .cshrc

```
#Assume LAM-MPI, PVM and MPICH are installed
setenv  LAMHOME /usr/local/lam61
setenv  PVM_ROOT /usr/local/pvm3
setenv  PVM_ARCH LINUX
setenv  MPIR_HOME /usr/local/mpich

set path = (. $path)
# use egcs compilers first
set path = (/usr/local/bin $path)
set path = ($path /usr/local/pvm3/lib/LINUX)
set path = ($path /usr/local/lam61/bin)
set path = ($path /usr/local/mpich/lib/LINUX/ch_p4)
```

5.9 시간동기화

2.0.x의 SMP와 시간변환에 몇가지 문제점들이 있다. 이는 몇몇 인터럽트 문제 때문에 발생한다. 가장 좋은 해결방법은 xntp를 사용해서 외부와의 시간을 일치시키도록한다. 어떠한 경우라도, 여러분의 클러스터의 시간을 동기화하라. xntp 설정방법은 다음과 같다.

1. 모든 시스템의 시간을 현재시각으로 설정하라.
2. `clock -w`명령을 이용하여 CMOS RTC (Real Time Clock)을 현재시각으로 변경하라.
3. 각 시스템에서 cdrom을 마운트하라. (`mount /mnt/cdrom`, 만일 실행되지 않으면 5절을 참조하라.)
4. `/mnt/cdrom/RedHat/RPMS`로 이동하라
5. root계정에서 `rpm -i xntp3-5.93-2.i386.rpm`을 실행하라.
6. `/etc/ntp.conf` 모든 시스템에서 다음 부분에 주석을 달아라.

   ```
   #multicastclient               # listen on default 224.0.1.1
   #broadcastdelay 0.008
   ```

   호스트를 제외한 나머지 시스템에는 다음과 같이 파일을 편집한다.

   ```
   server  HOSTNODE # local clock
   #fudge  127.127.1.0 stratum 0
   ```

   여기서 HOSTNODE라 함은 호스트노드의 이름이다.

   각 노드에서 `/etc/ntp.conf`를 닫고 나온다.
7. "/sbin/xntp" 명령을 통해서 xntpd를 실행한다.(역자주: 보통의 경우 /usr/sbin에 xntpd가 있다.

여러분이 이 명령을 /etc/rc.d/rc.local에 저장함으로써 시스템을 시작할 때마다, 실행시킬 수 있다.

시간동기화는 시간이 좀 걸리는 일이지만, /var/log/messages에서 xntpd로부터 나온 메세지들을 볼 수있다.

여러분이 방금한 작업은 호스트 노드에게 xntp를 실행시키고 지역 시간 시스템을 표준으로 삼는 것이다. 클러스터의 다른 노드들은 호스트로 부터 시간을 조정할 수 있다.

xntp가 시스템의 시간을 유지시키고 RTC를 동기화 시킨다. 하루에 한번씩 시간을 동기화 시키는 것이 중요하다. 이러한 작업은 관리자 계정에서 /etc/cron.daily를 통해 수행할 수 있으며 다음의 내용을 갖는 "sync_clocks"라는 파일을 만들어 놓는다.

```
#Assume ntp is running, so sync the CMOS RTC to OS system clock
/sbin/clock -w
```

여러분의 클러스터의 모든 시간을 동기화 시켜야하며 호스트를 항상 표준으로 삼아야 한다. 만일 여러분이 더알고 싶다면 xntpd 문서를 참고하라.

제 6 절 클라이언트 설정

클라이언트 노드를 설정하는데는 크게 세가지 방법이 있다. 우선 dd명령을 이용하여 노드들을 복제한다. 두번째 방법으로는 저자의 topcat 시스템에서 처음단계에서 사용했던 방법으로, 각각의 클라이언트에 운영체제를 따로 설치하고 나머지 설치를 할 수 있는 스크립트를 서버에서 실행시킨다. 세번째 방법으로는 서버에서 모든 설치과정이 끝난 disk-less 클라이언트 방법이다. 저자의 경우는 topcat 시스템에서 뒤의 두가지 방법을 사용했기 때문에 이 두가지 방법에 대해서 자세히 설명할 것이다.

6.1 클라이언트 복제

복제의 기본적인 개념은 한개의 드라이브에 있는 하나의 파티션을 정확하게 다른 드라이브에 복사하는 것이다. 하나의 클라이언트를 설치, 설정할 수 있고 디스크의 정확한 복사를 할 수 있다. 이 디스크 이미지를 다른 클라이언트에 사용할 수 있으며 여러분은 IP 주소와 호스트 네임과 같은 몇개의 부분만 변화시키면 된다. 만일 여러분의 클라이언트가 운영체제를 갖고 있는 자신만의 디스크를 갖고 있다면, 이러한 방법은 아주 손쉽게 할 수 있다. 복제는 Jan Lindheim에 의해 *Building a Beowulf System* <http://www.carc.caltech.edu/beowulf/tutorial/beosoft/>에 자세히 기술되어 있다. 하나의 디스크에서 한개의 파티션을 다른 디스크에 복사하는 것이다.

6.2 disk-less 클라이언트 설정

이 방법은 모든 클라이언트 설정을 서버에서 해야하기 때문에 앞선 두가지 방법과는 다르다. 이는 클라이언트들이 자신의 물리적 디스크를 갖고 있지 않기 때문에, 모든 파일을 서버노드에 저장해야한다. 만일 여러분이 disk-less 클라이언트 부팅에 관한 더 많은 정보를 원한다면, *NFS Root mini howto* <metalab.unc.edu/LDP/HOWTO/mini/NFS-Root.html>와 NFS Root Client HOWTO를 읽어보면 된다.

클라이언트 상에서 모든 시스템 파일들이 서버에 존재하기 때문에 클라이언트 설정은 모두 서버에서 이루어진다. 시스템에 약간의 변화를 주어 시스템을 설정할 때 NFS-root howto를 따라서 했다.

- 각 클라이언트들을 위해 우선 하나의 커널을 갖고 있는 플로피가 필요하다. 저자는 단순한 커널을 갖고 시도했지만 모듈커널이 작동하지 않는 이유를 알 수가 없었다. 한가지 기억해야할 일은 여러분의 네트워크 카드에 대한 지원내용을 커널에 넣어 컴파일해야 한다는 것이다. 파일 시스템을 마운트 하기전에 네트워크 드라이버가 필요할 것이다. 클라이언트에서 사용할 커널을 우선 컴파일하라. 다음과 같은 설정을 이용하면 된다:

```
make menuconfig
```

NFS-root를 지원하는 것을 컴파일하라: CONFIG_ROOT_NFS, CONFIG_RNFS_BOOTP, CONFIG_RNFS_RARP
커널에 있는 모든 선택사항을 설정한 후에 커널을 컴파일할 수 있다. 다음과 같은 명령을 실행한다.

```
make dep && make clean && make zImage
```

이제는 커널의 root 장치를 NFS-root로 변경할 것이다. *NFS-root Mini-howto*에서 알 수있는 dummy 장치를 형성하는 방법을 채택하였다.

```
makenod /dev/nfsroot b 0 255
cd /usr/src/linux/arch/i386/boot
rdev zImage /dev/nfsroot
```

이번에는 커널 이미지를 플로피에 복사를 하는 것이다.

```
dd if=zImage of=/dev/fd0
```

만일 여러분의 클라이언트가 동일하다면 모든 시스템을 시작하는 동일한 이미지를 사용할 수있다. 저자의 경우 두개의 다른 플로피를 준비했는데, 하나는 CPU가 하나인 경우이고 다른 하나는 SMP를 위한 것이다.

- 클라이언트를 위한 부트 디스크를 만든 후의 과정은 클라이언트의 root 디렉토리를 만드는데 사용되는 템플릿을 설정하는 것이다. 서버를 설치하고 운영체제의 패치를 한 직후 이 템플릿을 설치하는 것이 좋다. 그리고 /var와 /etc내의 파일들을 변경하기 전에 하는 것이 좋다. 단순히 15 (sdct) 스크립트를 잘라서 파일에 붙이면 되고 실행시키면 된다. 그 스크립트는 모든 필요한 디렉토리를 형성하고 모든 필요한 파일을 복사하는데 사용된다. 이 스크립트는 어떤 클라이언트에 대해서도 root 디렉토리를 형성하지 않으며단지 이 root 디렉토리를 만드는데 필요한 다른 스크립트를 이용할 수 있는 템플릿을 만드는데 이용된다. 여러분은 16 (adcn) 스크립트를 실행하여 각 클라이언트에 대해 NFS-root 파일 시스템을 만든다.

- NFS-root 디렉토리 템플릿을 만든 후, 각 클라이언트에 대해 NFS-root 파일 시스템을 만들어야 한다. 이는 16 (adcn) 스크립트를 실행하여 할 수 있고 이 스크립트는 /tftpboot아래에 파일시스템을 형성한다. 이 스크립트를 실행하는 일반적인 방법은:

  ```
  adcn -n node2 -i 10.0.0.2 -d beowulf.my.domain. -l -D eth1
  ```

  실행명령 옵션을 살펴보자:

  -n node2클라이언트의 이름을 의미한다. 도메인이름은 쓰지 않는다.

  -i 10.0.0.2 클라이언트의 IP주소를 정한다.

  -d beowulf.my.domain는 클러스터의 DNS를 의미한다. 이 선택사항이 정해지지 않으면 서버의 DNS 도메인이 사용된다. 여러분 서버의 도메인이 클러스터의 도메인과 다른경우에 사용하자. 저자의 경우, 클라이언트의 완전한 이름은 node2.beowulf.my.domain이다.

  -l는 RARP 요청을 듣는다는 의미이다. 이 선택사항이 쓰여지면, adcn는 -D 선택사항과 함께 정해지는 인터페이스에 있는 RARP 요청을 듣게 되고 클라이언트 하드웨어 주소로서 첫번째 "sniffed" RARP 요청으로부터 MAC 주소를 사용한다. 이 선택사항은 MAC 주소를 갖고 오기 위해 tcpdump를 사용한다. 우선 tcpdump가 설치되어있는지 확인하라.

  -D는 클러스터에 연결된 디바이스를 정하는데 필요한 선택사항이다.만일 여러분이 여러분의 클러스터에 연결된 디바이스가 한개 이상이면 여러분은 disk-less 클라이언트가 연결되어 있는 네트워크 인터페이스를 이용해야한다. 이 선택사항은 /etc/sysconfig/network-scripts/ifcfg-*에서 디바이

스 정보를 읽고 네트워크와, 넷마스크, 게이트웨이를 읽어드린다.(서버의 IP는 게이트웨이로 사용된다.) 디바이스 정보는 -l선택사항에 의해 나타나고 그 정보를 tcpdump를 통해 전달한다.

만일 -D 선택사항이 정해지지 않으면 adcn 스크립트는 default 값을 사용할 것이다. 다른 많은 선택사항은 adcn -h를 이용하면 알 수 있다. 대부분 여러분이 필요한 내용은 위에 명시되어 있다. 여러분은 스크립트에 중복명령을 실행할 수 있으며 하나의 명령을 이용하여 disk-less 클라이언트 전체를 설치할 수 있다. 예를 들어 클러스터에 연결된 서버의 인터페이스 eth1를 이용 16 node disk-less 클라이언트를 설치하기 위해 여러분은 다음과 같은 스크립트를 실행하면 된다:

```
#!/bin/bash
adcn -n node2 -i 10.0.0.2 -d beowulf.my.domain -l -D eth1
adcn -n node3 -i 10.0.0.3 -d beowulf.my.domain -l -D eth1
adcn -n node4 -i 10.0.0.4 -d beowulf.my.domain -l -D eth1
adcn -n node5 -i 10.0.0.5 -d beowulf.my.domain -l -D eth1
adcn -n node6 -i 10.0.0.6 -d beowulf.my.domain -l -D eth1
adcn -n node7 -i 10.0.0.7 -d beowulf.my.domain -l -D eth1
adcn -n node8 -i 10.0.0.8 -d beowulf.my.domain -l -D eth1
adcn -n node9 -i 10.0.0.9 -d beowulf.my.domain -l -D eth1
adcn -n node10 -i 10.0.0.10 -d beowulf.my.domain -l -D eth1
adcn -n node11 -i 10.0.0.11 -d beowulf.my.domain -l -D eth1
adcn -n node12 -i 10.0.0.12 -d beowulf.my.domain -l -D eth1
adcn -n node13 -i 10.0.0.13 -d beowulf.my.domain -l -D eth1
adcn -n node14 -i 10.0.0.14 -d beowulf.my.domain -l -D eth1
adcn -n node15 -i 10.0.0.15 -d beowulf.my.domain -l -D eth1
adcn -n node16 -i 10.0.0.16 -d beowulf.my.domain -l -D eth1
```

6.2.1 문제해결 방안

- Disk-less 클라이언트가 서버로 부터 RARP 응답을 얻지 못한다만일 여러분이 시스템을 시작하고 나서 "Sending BOOTP and RARP requests..."라고 메세지가 뜨는 경우 여러분은 다음의 내용을 살펴보아야한다.

  네트워크 케이블, 스위치 설정등을 확인하고 서버에 있는 인터페이스가 정확히 설정되어 있는지 확인하다.

  rarp가 서버 커널에서 지원이 되는지 확인한다.

  문제가 되는 클라이언트에 대한 rarp입력이 있는지 확인한다. 이는 'rarp -a'명령으로 알 수 있다. 클라이언트 하드웨어 주소가 정확한지 확인한다.

  'tcpdump -i eth1 rarp'를 서버에서 실행시키고 disk-less클라이언트를 부팅한다.(eth1이 클러스터에 연결된 인터페이스라고 가정한다.) 클라이언트가 부팅되고 rarp 요청을 내보낼 때, 여러분은 tcpdump 결과에서 이를 보아야한다. 만일 모든 설정이 정확히 되었다면, 여러분은 서버의 rarp 응답을 보면 될 것이다. 여러분이 요청을 보지 못했다면, 문제의 대부분의 원인은 잘못된 연결이다; 이는 케이블, 스위치, NIC가 문제로 작용했을 수 있다. 클라이언트의 rarp 요청을 볼 수 있지만, 서버가 응답하지 않는다면, 대부부의 문제는 rarp 입력이 잘못되어 나오는 결과이다.

6.3 클라이언트 콘솔접근 방법

여러분의 클라이언트들이 비디오 카드나 키보드등이 없는 경우 여러분이 서버에서 직접적으로 그들에게 연결할 수 있는 방법이 없다. 설정변화도중 네트워크에 문제가 생길경우와 클라이언트에 telnet 이나 rlogin을 할 수 없으면 여러분은 다른 방법으로 접속해야한다. 클라이언트 콘솔로 접속하는 몇가지 방법이 있다. 첫번째 방법은 Jan Lindheim이 만든 *Building a Beowulf System* <http://www.cacr.caltech.edu/beowulf/

tutorial/building.html>에 나온 모니터와 키보드 스위치를 이용하면 된다. 다른 한가지 방법은 serial terminal을 사용하는 것이다.

6.4 각 클라이언트에 OS 설치

만일 CD-ROM에서 설치하고 단지 하나의 드라이브가 있는 경우, 여러분은 설치할 때마다 각 CD-ROM 드라이브를 옮겨야한다. 그렇지 않은 경우는 NFS 설치를 하면된다. 여러분이 하나의 플로피 드라이브를 갖고 있다면 마찬가지로 해야한다. 저자의 경우 지역 ftp 서버를 이용하여 모든 노드에 설치하였다. 플로피 드라이브도 옮겨야했다. 설치시간을 줄이기 위해 저자는 완전한 설치를 권장한다. 설치할 꾸러미를 선택하는 것은 고통스러운(?) 일이며 16노드의 경우 정말 힘들다. 최근에는 가장 작은 크기의 하드가 2GB이므로 설치공간은 걱정하지 않아도 된다.

제 7 절 Compilers

저자는 egcs(g77을 포함하는) 사용을 권장한다. 소스는 <http://egcs.cygnus.com>에 있고 버전:egcs-1.1.1 gzip 형태이다.

한번 컴파일하고 설치하면 egcs 컴파일러는 /usr/local에 있게된다. 이러한 방법으로 사용자들은 적절한 버전으로 경로를 지정할 수 있다.(즉, 표준 egcs는 /usr/bin에 있고 egcs gcc는 /usr/local/bin에 있다.)

Note: 커널을 형성할 때는 gcc를 사용하라.(egcs gcc가 아니고) gcc -v 와 which gcc는 버전을 보여준다.

g77은 egcs FORTRAN 컴파일러이다.

제 8 절 통신 소프트웨어

다음이 완전한 목록은 아니다. 단지 많이 사용되는 패키지이다. 클러스터는 지역메모리 머신의 집합이다. node A가 node B와 통신하는 유일한 방법은 네트워크를 통해서이다. 이러한 메세지 전달 구조의 맨위에 소프트웨어는 코드는 개념상으로는 간단하지만, 작동이나 디버깅은 매우 복잡할 수 있다.

두가지 많이 쓰이는 소프트웨어로는 PVM과 MPI가 있다.

8.1 PVM VS MPI

PVM과 MPI 모두 메세지 전달을 도와주는 적용이 간편한 소프트웨어이다. 역사적으로 볼 때 PVM이 먼저 개발 되었고 워크스테이션의 네트웍에 맞게 설계되었다.(Parallel Virtual Machine) 이는 분산 저장을 하거나 하지 않거나에 관계없이 많은 병렬 수퍼컴퓨터에 적용되어져 왔다. PVM의 여러제반 사항에 관해서는 그것을 만든이들이 주로 관여하고 있다.

MPI는 그와는 달리 많은 하드웨어 판매자에 의해 지원되고 있으며 PVM보다 더 많은 기능을 제공한다. 클러스터를 위한 버젼이 있다. MPI에 관계된 사항은 표준위원회에서 결정한다.

많은 경우 PVM과 MPI 둘 중 어느하나를 써야한다는 규칙은 없다. MPI의 경우 표준이 정해져 있기 때문에 많은 사람들이 MPI를 선호한다. 하지만 PVM도 사용되고 있다. 이 문서는 각 소스에 관한 정보를 포함하고 있다.

MPI:

자유롭게 사용할 수 있는 두가지의 MPI 버젼이 있다. (역자주: 그 이외에도 여러가지가 있으며 <http://kluster.kaist.ac.kr> 등에서 확인할 수 있다.)

MPICH(역자주: MPI Chameleon의 약자):

Source: <http://www-unix.mcs.anl.gov/mpi/mpich/> Version: mpich.tar.gz (역자주: 최근 1.2 버젼까지 나왔음.) Notes: 우리를 포함한 사람들이 리눅스 버젼에 관해 몇가지 문제점들을 제시하고 있음.

LAM-MPI:

Source: <http://www.mpi.nd.edu/lam/> Version: lam61.tar.gz (역자주: 최근 6.4버젼까지 나왔음.) Notes: 패치(lam61-patch.tar)을 설치한다. LAM의 경우 -c2c 모드를 사용할 경우 좋은 결과를 나타냄. (역자주: -c2c는 옵션임)

PVM:

Version: pvm3/pvm3.4.beta7.tgz Source: <http://www.epm.ornl.gov/pvm/> Notes: 많은 PVM 코드와 예제들이 나와있음.

제 9 절 변환 소프트웨어:

기존의 소프트웨어를 병렬처리에 알맞게 변환한다는 것은 시간이 오래걸리는 작업이다. 자동변환은 매우 힘들다. 자동변환은 FORTRAN 변환에만 적용되고 있다. C를 변환하는 것은 포인터 때문에 매우 힘듦.

FORTRAN 코드의 변환방법을 BERT라고 불리우며 리눅스 시스템 에서 작동한다. <http://www.plogic.com/bert.html>에서 자유롭게 얻을 수 있다.

제 10 절 시스템 감시 소프트웨어

10.1 bWatch

bWatch는 GUI Beowulf 클러스터 모니터이다. 이는 load average와 메모리, 스왑, 프로세스수, 단일창에 있는 노드에 대한 사용자들. bWatch는 <http://www.sci.usq.edu.au/staff/jacek/bWatch>에서 구할 수 있다.

NOTE: bwatch.rpm은 SuSE 리눅스 배포본에 있으며 /usr/X11R6/bin에 설치되고 wish interpreter도 동일한 디렉토리에 있다. Red Hat 리눅스 배포본은 /usr/bin에 wish가 있기에 bWatch가 실행되지 않을 수 있다. 여러분은 /usr/X11R6/bin/bWatch의 첫번째 줄을 #!//usr/X11R6/bin/wish를 /usr/bin/wish로 변경할 수 있다.

10.2 httpd와 CGI 스크립트 이용하기

여러분의 beowulf 클러스터로부터 통계를 얻는 방법중의 하나는 서버노드에서 httpd와 CGI 스크립트를 실행시켜서이다. CGI 스크립트가 원격 shell을 여러분이 알고자 하는 노드에 실행시켜 서버가 httpd를 이용하여 여러분의 브라우저에 보내는 내용을 HTML 형태로 정보를 바꾼다. 이는 브라우저가 있고 인터넷에 연결만 되어 있다면 쉽게 클러스터의 성능을 알아볼 수 있는 방법이다. 이방법의 예는 <ftp://ftp.sci.usq.edu.au/pub/jacek/beowulf-utils>에 있는 index.html파일에 있다. 이 파일은 getinfo.cgi이다.

10.3 Netpipe

Netpipe는 TCP의 결과, 다른 크기의 MPI, PVM 패킷을 확인할 수 있는 네트워크 작동 도구이다. 여러분은 gnuplot이나 spreadsheet으로 Netpipe를 통해 나온 결과를 그래프로 나타낼 수 있다. 여러분은 NetPIPE를 <http://www.scl.ameslab.gov/Projects/ClusterCookbook/nprun.html>에서 구할 수 있다.

10.4 네트워크 퍼포먼스: netperf

Source: <http://www.netperf.org/netperf/NetperfPage.html>

```
Run Script:
./netperf -t UDP_STREAM -p 12865 -n 2 -1 60 -H NODE -- -s
          65535 -m 1472
./netperf -t TCP_STREAM -p 12865 -n 2 -1 60 -H NODE
```

NODE는 원격 노드 이름이다.

10.5 병렬 퍼포먼스: NASA 병렬 벤치마크

Source: <http://www.nas.nasa.gov/NAS/NPB>

제 11 절 관리: CMS

CMS (Cluster Management System)라고 불리는 꾸러미가 있다. 이는 <http://smile.cpe.ku.ac.th/software/scms/index.html>에서 정보를 얻을 수 있다. 새로운 버젼에 관해서는 테스트 해볼 시간 이 없었다. 그 전에 나온 버젼에 관해서는 실시간 원격 모니터링 을 제외하고 잘 작동하였다. 이는 시스템의 재시작과 중지에 관한 내용을 포함하고 있다.

제 12 절 보안

Beowulf 클러스터의 일반적인 보안정책은 클러스터내의 모든 노드가 서로를 신뢰할 수 있어야한다는 것이다. 클러스터내의 보안에 여러분이 안심할 수 있는 이유는 클라이언트 노드 어떠한 것도 외부와 직접연결이 되어 있지 않고 모든 노드들이 기본적으로 동일하기 때문이다. 만일 누군가가 게이트웨이를 해킹(역자주: 크래킹이 정확한 명칭)하려 한다면 크래커들은 클라이언트 노드에 대한 정보는 전혀 얻을 수 없어서 이러한 수준에서는 보안문제를 걱정할 필요가 없다. 누군가가 여러분의 노드의 콘솔에 있지 않고 서버노드를 거치지 않고 노드에 접속을 한다는 것은 불가능한 일이다. 클러스터내의 보안을 완화시키는 가장 큰 장점은 유연성이고 사용하기 쉽고 관리하기 쉽다는 점이다. 이와는 달리 서버노드는 클라이언트 노드를 믿어야 하지만 외부세계는 믿어서는 안된다. 클러스터내의 보안을 완화시키고 외부로부터 여러분 자신을 지키는 방법은 몇가지 있다.

12.1 서버

12.1.1 TCP wrapper

일반적으로 TCP wrapper로 알려진 tcpd 데몬은 방어의 제1선이고 여러분의 머신에 접속을 제한하는 가장 간단한 방법이어서 시스템의 보안성을 높인다. 이는 Red Hat 배포본의 일부분으로 나와있고 설정이 간단하다. 세가지 설저파일들이 있다: `/etc/hosts.allow`는 연결을 허락하는 호스트를 확인한다. `/etc/hosts.deny`는 `/etc/hosts.allow`에 나타나지 않은 목록이 있으면 읽어드린다. 연결을 거부하는 호스트를 확인한다. `/etc/inetd.conf`는 tcpd를 설정할 때 변경할 필요없는 것들이 있다. `host_access(5)` man 페이지는 `/etc/hosts.allow`와 `/etc/hosts.deny`의 문법에 관한 내용에 대한 좋은 정보를 준다.

Allowing access with `/etc/hosts.allow` 아래의 예는 IP 주소가 10.0.0.x, 10.1.x, 10.0.2.x에서 들어오는 어떠한 포트도 연결을 허락한다는 의미이다. 또한 `myworkstation.usq.edu.au`의 호스트로 부터오는 것도 접속을 허락한다는 의미이다. 모든 다른 접속은 `/etc/hosts.deny`파일에 의해서 막히고 서비스들은 `/etc/inetd.conf`에 목록화되어 있으며, `tcpd`를 통해서 설정이 시작된다.

```
#
# hosts.allow   This file describes the names of the hosts which are
#               allowed to use the local INET services, as decided
#               by the '/usr/sbin/tcpd' server
#

# we fully trust ourself and all the other nodes within the cluster

ALL : localhost, 10.0.0., 10.0.1., 10.0.2.
in.telnetd : myworkstation.usq.edu.au
```

/etc/hosts.deny를 이용한 접근거부 /etc/hosts.deny파일은 /etc/hosts.allow파일에서 일치하지 않는 호스트를 확인한다. TCP wrapper를 이용하는 가장 좋은 방법은 /etc/hosts.allow에서 허락하지 않는 모든 호스트를 거부하는 것이다. 저자의 경우 /etc/hosts.deny와 일치하지 않는 것은 물론 이거니와 모든 것의 접근을 거부시켜 놓았다. 모든 거부된 연결에 대해서는 관리자에게 자세한 내용을 이메일로 보낸다.

```
ALL: ALL: spawn ( \
echo -e "\n\
TCP Wrappers\:  Connection Refused\n\
By\:                    $(uname -n)\n\
Process\:               %d (pid %p)\n\
User\:                  %u\n\
Host\:                  %c\n\
Date\:                  $(date)\n\
" | /bin/mail -s "From tcpd@$(uname -n).  %u@%h -> %d." root)
```

만일 연결이 /etc/hosts.allow에서 나와있지 않는 호스트로부터 시도된다면 /etc/hosts.deny에서 연결을 거부할 것이며 거기에 따른 이메일을 저자는 받게 될 것이다. 그러한 이메일의 내용은 다음과 같다.

```
From root       Fri Apr 16 23:33:50     1999
Return-Path: <root>
             by topcat.beowulf.usq.edu.au (8.8.7/8.8.7) id XAA19278
             for root; Fri, 16 Apr 1999 23:33:50 +1000
Date: Fri, 16 Apr 1999 23:33:50 +1000
From: TOPCAT Admin <root@topcat.beowulf.edu.au>
Message-Id: <199904161333.XAA19278@topcat.beowulf.usq.edu.au>
To: root@topcat.beowulf.edu.au
Subject: From tcp@topcat.beowulf.usq.edu.au.   jacek@lamport.comp.usq.edu.au -> in.telnetd.
Status: 0

TCP Wrappers: Connection Refused
By:             topcat.beowulf.usq.edu.au
Process:        in.telnetd (pid 19270)
User:           jacek
Host:           jacek@lamport.comp.usq.edu.au
Date:           Fri 16 Apr 1999 23:33:50 EST 1999
```

12.1.2 사용하지 않는 데몬 멈추기 - /etc/inetd.conf

아주 단순 하지만 효과적인 서버보안의 한가지 방법은 사용하지 않는 데몬을 멈추는 것이다. 일반적으로 여러분이 사용하지 않는 것은 멈추는 것이 좋다. 대부분의 데몬이 inetd에 의해 작동이 되고 /etc/inetd.conf의 내용에서 사용하지 않는 데몬을 주석처리 해주면 작동하지 않는다. 다음의 예는 /etc/inetd.conf의 login, exec, talk과 ntalk의 예를 보여주는 것이다.

```
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
#login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rexecd
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  in.comsat
#talk   dgram   udp     wait    nobody.tty      /usr/sbin/tcpd  in.talkd
#ntalk  dgram   udp     wait    nobody.tty      /usr/sbin/tcpd  in.ntalkd
```

설정파일을 변경한 후에 다시 inetd데몬을 시작한다. 리눅스에서 가장 손쉬운 방법은 설정파일을 다시 읽어 드리게 데몬에게 신호를 주는 것이다.

```
[root@topcat root]# killall -HUP inetd
```

다른 유닉스 시스템에서는 kill을 잘 읽고 수행하라!

여러분은 모든 포트의 목록을 가지고 어떤 데몬이 수행되고 있는지를 확인할 수 있다. 다음 명령으로 이 목록을 얻을 수 있다.

```
[root@topcat root]# netstat -a | grep "LISTEN" | grep -v "^unix"
```

12.1.3 rc 스크립트를 이용하여 서버멈추기

웹서버(http)와 같은 서버와 삼바(smbd)는 rc 스크립트로서 작동한다. 보통 각각은 /etc/rc.d/rc3.d에 있는 각각에 해당하는 링크를 제거함으로써 작동을 멈추게된다. 예를 들어 httpd, samba와 sendmail(또다른 보안프로그램)dms run level 3과 5을 작동시킴으로써 다음과 같이 할 수 있다.

```
[root@topcat samba]# rm -f /etc/rc.d/rc/3d/S*httpd
[root@topcat samba]# rm -f /etc/rc.d/rc/5d/S*httpd
[root@topcat samba]# rm -f /etc/rc.d/rc/3d/S*smb
[root@topcat samba]# rm -f /etc/rc.d/rc/5d/S*smb
[root@topcat samba]# rm -f /etc/rc.d/rc/3d/S*sendmail
[root@topcat samba]# rm -f /etc/rc.d/rc/5d/S*sendmail
```

12.1.4 ipfwadm

ipfwadm프로그램은 특정 IP 주소에서부터 특정 포트까지 패킷을 막는역할을 한다. 이는 보안을 제어하는 가장 유연한 방법이다. 예를 들어 firewall (17절 (firewall_script)) rc 스크립트는 반드시 시스템이 시작할 때 같이 시작되어야 하고 다음과 같이 하면된다.

```
[root@topcat init.d]# cp /home/jacek/firewall /etc/rc.d/init.d
[root@topcat init.d]# chmod u+rx firewall
[root@topcat init.d]# ln -s /etc/rc.d/init.d/firewall /etc/rc.d/rc3.d/S05firewall
[root@topcat init.d]#ln -s /etc/rc.d/init.d/firewall /etc/rc.d/rc5.d/S05firewall
```

NOTE: 여러분은 여러분의 환경에 맞게 저자의 스크립트를 바꾸어야한다.

12.2 클라이언트

12.2.1 .rhosts versus hosts.equiv

사용자들이 하고싶어 하는 것중의 하나는 패스워드없이 노드간에 접속을 하고 원격명령을 내리는 것이다. 대부분의 Beowulf 소프트웨어와 유틸리티들은 여러분이 rsh로 작동하게 만들어 패스워드 없이 작업하게 만든다.

클러스터내의 패스워드를 없애는 두가지 방법이 있는데 하나는 /etc/hosts.equiv에 입력하는 것이고, 다른 하나는 사용자 각자의 디렉토리에 .rhosts를 첨가하는 것이다.

/etc/hosts.equiv가 모든 노드에 .rhosts에 있는 내용을 모아서 하나의 파일로 적용될 수 있기에 많이 선호된다.

다음의 형태는 .rhosts에 있는 호스트의 목록이다:

```
# must be read/writable by user only!
node1
node2
node3
node4
node5
node6
```

/etc/hosts.equiv의 형태는:

```
#node name      optional user name
node1
node2
node3
node4
node5
node6
```

12.2.2 root rlogin 접근:

root가 클러스터내의 어떠한 노드에도 rlogin하기 위해서는 각노드의 root 디렉토리에 .rhosts를 첨가해야합니다. .rhosts파일은 클러스터내의 모든 노드들을 명기하고 있어야합니다. 중요한점: .rhosts는 반드시 사용자만이 읽고 쓸 수 있어야합니다. (chmod go-rwx .rhosts) 이는 게이트웨이 노드에서는 해서는 안됩니다.

추가로 /etc/pam.d/rlogin:의 처음 두줄을 바꿔줍니다.

```
#orginal /etc/pam.d/rlogin
auth     required       /lib/security/pam_securetty.so
auth     sufficient     /lib/security/pam_rhosts_auth.so
auth     required       /lib/security/pam_pwdb.so shadow nullock
auth     required       /lib/security/pam_nologin.so
account  required       /lib/security/pam_pwdb.so
password required       /lib/security/pam_cracklib.so
password required       /lib/security/pam_pwdb.so shadow nullock
                                                  use_authtok
session  required       /lib/security/pam_pwdb.so
```

```
#first two lines are swapped /etc/pam.d/rlogin
auth     sufficient     /lib/security/pam_rhosts_auth.so
auth     required       /lib/security/pam_securetty.so
auth     required       /lib/security/pam_pwdb.so shadow nullock
auth     required       /lib/security/pam_nologin.so
account  required       /lib/security/pam_pwdb.so
password required       /lib/security/pam_cracklib.so
password required       /lib/security/pam_pwdb.so shadow nullock
                                                  use_authtok
session  required       /lib/security/pam_pwdb.so
```

NOTE: 더 나은 방법이 있는지는 모르겠지만 작동은 한다.

12.2.3 root telnet 접근

게이트웨이 노드를 제외한 모든 노드에 /etc/securetty 파일에 다음과 같은 내용을 첨가한다:

```
ttyp0
ttyp1
ttyp2
ttyp3
ttyp4
```

이러한 변화는 remote telnet을 이용 클러스터내의 어떠한 노드로 연결 이 가능케하는 것이다.

12.2.4 root ftp 접근

root의 ftp 접근이 필요한 시스템의 경우, /etc/ftpusers 파일에 다음과 같이 root 부분에 주석을 단다.

```
#Comment out root to allow other systems ftp access as root
#root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody
```

제 13 절 이더넷 채널본딩:

채널 본딩에 관한 내용은 <http://www.beowulf.org/software/software.html>

요구사항: 시스템당 두개의 이더넷 NIC 각 채널당 두개의 허브 또는 각 채널당 두개의 스위치 또는 버추얼 LAN으로 분리될 수 있는 스위치

수행과정: (리눅스 커널 2.0.36)

1. ifenslave.c 프로그램을 다음 사이트에서 받는다. (<http://beowulf.gsfc.nasa.gov/software/bonding/html>) 35라인에 주석처리 "#include " 그리고 "gcc -Wall -Wstrict-prototypes -O ifenslave.c -o ifenslave" 를 실행한다.

2.커널패치를 한다.(<ftp://ftp.plogic.com>에서 얻은 linux-2.0.36-channel-bonding.path를 커널 패치한다.)그리고 xconfig를 실행시키고 Beowulf Channel Bonding을 가능케 한다.

3. 커널을 재형성하고 컴파일한다. 각 채널은 각기 다른 스위치 또는 허브(또는 분리된 스위치)에 있 어야 하며 두번째 네트웍 인터페이스는 IP 주소를 부여할 필요가 없다. 하지만 그 인터페이스를 분리된 네트웍으로 사용할 것이다. (채널 본딩없이) 이는 몇가지 응용에 이점이 있다.

채널 본딩을 위해 각 시스템이 root로 로그인하여 다음과 같은 명령을 실행한다.

./ifenslave -v eth0 eth1

이는 eth1과 eth0을 연결시켜 준다. 물론 eth0는 이미 시스템에서 받아들여져 있고 클러스터 네트웍으로 사용하고 있다. eth1은 단지 시스템 시작시 OS(Linux)에 의해서 감지된다.

여러분은 반드시 호스트 노드전에 모든 노드들을 슬레이브화함으로 써 이러한 작동을 시킬 수 있다. 각 노드는 다음 과정을 수행한다.

a. 창을 연다. b. 노드2에 로그인 한다. c. root계정으로 위 명령을 수행한다. d. 다른 창을 열어 노드1에 대해서 위의 명령을 수행한다.

그러면 여러분 클러스터는 채널본딩이 된 것이다. netperf나 비슷한 벤치마크를 해봄으로써 이러한 것을 시험해 볼 수 있다.

채널본딩의 멈춤은 쉬운 문제가 아니다. 우리는 이를 잘 살펴보아야 하며 채널본딩이 자동적으로 형성되고 멈추는 명령행을 입력해야한다. 하나의 채널 퍼포먼스를 저장하기 위한 가장 안전한 방법은 각 시스 템을 다시 시작하는 것이거나 네트웍 메니저(제어판의 일부)을 이용 각 인터페이스를 재시작하고 멈추게 할 수 있다.

기억할 점: 채널 본딩이 된 노드들과 그렇지 않은 노드간의 통신은 가능하지만 매우 느리다. 따라서 전체 클러스트가 채널 본딩을 해야 만한다.

제 14 절 스위치 Configuration:

```
BayStack 350T Main Menu

      IP Configuration...
      SNMP Configuration....
      System Characteristics...
      Switch Configuration...
      Console/Service Port Configuration...
      Spanning Tree Configuration...
      TELNET Configuration...
      Software Download...
      Display Event Log...
      Reset
```

```
Reset to Default Settings
Logout
```

화살표를 이용 원하는 옵션에 마킹을 하고 옵션을 선택한다.

제 15 절 참고문헌과 다른 정보출처들

- Jacek Radajwski and Douglas Eadline, *Beowulf HOWTO*, Linux Documentation Project <http://metalab.unc.edu/LDP/HOWTO/Beowulf-HOWTO.html>
- Jan Lindheim *Building a Beowulf System* <http://www.cacr.caltech.edu/beowulf/tutorial/building.html>
- *Cluster Cookbook* <http://www.scl.ameslab.gov/Projects/ClusterCookbook/nprun.html>
- Beowulf Papers <http://www.beowulf.org/papers/papers.html>
- Beoulf Homepage <http://www.beowulf.org>
- Jacek's Beowulf Page <http://www.sci.usq.edu.au/staff/jacek/beowulf>
- *Linux Parallel Processing HOWTO* <http://yara.ecn.purdue.edu/~pplinux/PPHOWTO/>
- *NFS-Root mini howto* <http://metalab.unc.edu/LDP/HOWTO/mini/NFS-Root.html>
- *NFS-Root-Client mini howto* <http://metalab.unc.edu/LDP/HOWTO/mini/NFS-Root-Client.html>
- Extreme Linux Homepage <http://www.extremelinux.org>
- *NFS-Root HOWTO*, Linux Documentation Project <http://metalab.unc.edu/LDP/HOWTO/NFS-HOWTO>
- *NIS-Root HOWTO*, Linux Documentation Project <http://metalab.unc.edu/LDP/HOWTO/NIS-HOWTO>
- *DNS HOWTO*, Linux Documentation Project <http://metalab.unc.edu/LDP/HOWTO/DNS-HOWTO>
- W.B. Ligion III and R. B. Ross *Implementation and Performance of a Parallel File SYstem for High Performance Distributed Applications* <http://ece.clemson.edu/parl/pvfs/pvfshpdc.ps>

제 16 절 sdct 스크립트

disk-less 클라이언트 템플릿을 설치 NOTE: 이 스크립트는 "setup_template"라 불리우고 몇몇 문서들이 이 오래된 이름을 참조하고 있다. 또한 이 스크립트 버전이 1.0.0 미만인 것은 실험적인 것이며 여러분운 위험을 감수해야한다.

<ftp://ftp.sci.usq.edu.au/pub/jacek/beowulf-utils/disk-less/sdct>

제 17 절 adcn 스크립트

Disk-less 노드를 첨가하라.

NOTE: 이 스크립트는 "add_node"라고 불리워졌으며 몇몇 문서들은 이 오래된 이름을 참조하고 있다. 또한 이 스크립트 버전이 1.0.0 미만인 것은 실험적인 것이며 여러분운 위험을 감수해야한다.

<ftp://ftp.sci.usq.edu.au/pub/jacek/beowulf-utils/disk-less/adcn>

제 18 절 firewall스크립트

```
#/etc/rc.d/init.d/firewall
#
# This file sets up the firewall rule
# for topcat.eng.usq.edu.au Beowulf class supercomputer
# version 1.0.0
# 18/08/1998
#
# author : Jacek Radajewski jacek@usq.edu.au
#
# this is our third line of defence
# 1. most of the services are disabled in inetd
# 2. secondly we use tcpd
# 3. we filter packets at the kernel level (this rc script)
#

# the ipfwadm program
IPFWADM="/sbin/ipfwadm"

 case "$1" in
  start)
  echo -n "Inserting firewall rules ... "
  export MODE="-i"
  # default policies
  export IN_POLICY="accept"
  export OUT_POLICY="accept"
  # if you have machines outside the cluster connected to
  # the main system via IP tunnel as described at
  # http://www.sci.usq.edu.ay/staff/jacek/topcat then you will
  # have to allow forwarding
  export FORWARD_POLICY="deny"
          ;;
  stop)
  echo -n "Deleting firewall rules ... "
  export MODE="-d"
  # default policies
  export IN_POLICY="accept"
  export OUT_POLICY="accept"
  export FORWARD_POLICY="accept"
          ;;
    *)
  echo "Usage: firewall {start|stop}"
  exit 1
  esac

  # source eth0 configuration
  # we assume that eth0 is our interface to the outside world
  # most firewall rules will be based on this
```

```
 . /etc/sysconfig/network-scripts/ifcfg-eth0
# this must be set to the host's IP address
 export MYIP=$IPADDR

 # we want to allow administrator to telnet in
 export ADMINIP=139.x.x.x

 #-----------------------------------------------------------------------
 # we first set default policies
 #-----------------------------------------------------------------------

 $IPFWADM -I -p $IN_POLICY
 $IPFWADM -O -p $OUT_POLICY
 $IPFWADM -F -p $FORWARD_POLICY

 #-----------------------------------------------------------------------
 # forwarding rules
 # deny all TCP and UDP
 #-----------------------------------------------------------------------

 $IPFWADM -F $MODE deny -S 0.0.0.0/0 -D 0.0.0.0/0 -P tcp
 $IPFWADM -F $MODE deny -S 0.0.0.0/0 -D 0.0.0.0/0 -P udp

 #-----------------------------------------------------------------------
 # We go through the normal services and deny everything we don't need
 # from outside.
 #-----------------------------------------------------------------------

 # ftp
 #$IPFWADM -I $MODE deny -D $MYIP/32 ftp -S 0.0.0.0/0 -P tcp
 #$IPFWADM -I $MODE accept -D $MYIP/32 ftp -S $ADMINIP/32 -P tcp

 # telnet
 #$IPFWADM -I $MODE deny  -D $MYIP/32 telnet -S 0.0.0.0/0 -P tcp
 #$IPFWADM -I $MODE accept -D $MYIP/32 telnet -S $ADMINIP/32 -P tcp

 # we block other known services ... well most of them

 $IPFWADM -I $MODE deny -D $MYIP/32 echo -S 0.0.0.0/0 -P tcp
 $IPFWADM -I $MODE deny -D $MYIP/32 echo -S 0.0.0.0/0 -P udp
 $IPFWADM -I $MODE deny -D $MYIP/32 discard -S 0.0.0.0/0 -P tcp
 $IPFWADM -I $MODE deny -D $MYIP/32 discard -S 0.0.0.0/0 -P udp
 $IPFWADM -I $MODE deny -D $MYIP/32 systat -S 0.0.0.0/0 -P tcp
 $IPFWADM -I $MODE deny -D $MYIP/32 daytime -S 0.0.0.0/0 -P tcp
 $IPFWADM -I $MODE deny -D $MYIP/32 daytime -S 0.0.0.0/0 -P udp
 $IPFWADM -I $MODE deny -D $MYIP/32 netstat -S 0.0.0.0/0 -P tcp
 $IPFWADM -I $MODE deny -D $MYIP/32 finger -S 0.0.0.0/0 -P tcp
 #$IPFWADM -I $MODE deny -D $MYIP/32 http -S 0.0.0.0/0 -P tcp
 $IPFWADM -I $MODE deny -D $MYIP/32 pop -S 0.0.0.0/0 -P tcp
 $IPFWADM -I $MODE deny -D $MYIP/32 pop-3 -S 0.0.0.0/0 -P tcp
```

```
$IPFWADM -I $MODE deny -D $MYIP/32 imap -S 0.0.0.0/0 -P tcp
$IPFWADM -I $MODE deny -D $MYIP/32 exec -S 0.0.0.0/0 -P tcp
$IPFWADM -I $MODE deny -D $MYIP/32 login -S 0.0.0.0/0 -P tcp
$IPFWADM -I $MODE deny -D $MYIP/32 syslog -S 0.0.0.0/0 -P udp
$IPFWADM -I $MODE deny -D $MYIP/32 shell -S 0.0.0.0/0 -P tcp
$IPFWADM -I $MODE deny -D $MYIP/32 talk -S 0.0.0.0/0 -P udp
$IPFWADM -I $MODE deny -D $MYIP/32 ntalk -S 0.0.0.0/0 -P udp
$IPFWADM -I $MODE deny -D $MYIP/32 cfinger -S 0.0.0.0/0 -P tcp
$IPFWADM -I $MODE deny -D $MYIP/32 nfs -S 0.0.0.0/0 -P udp

# we stop all connections to our X server (if running)
# comment out the line below if you require X access

#$IPFWADM -I $MODE deny -D $MYIP/32 6000 -S 0.0.0.0/0 -P tcp

echo "firewall"
```

DNS HOWTO의 최신버전은 bind8을 포함하고 있지만 많은 배포본이 bind version 4를 포함하고 있다.

제 19 절 변화들(버전 0.1.0에서 시작된)

- 02/06/1999 v0.1.2 - 문제해결부분 시작. 이 절은 disk-less 클라이언트 설정에서 일어날 수 있는 문제점들을 포함하고 있고 거기에 따른 해결점도 갖고 있다.
- 25/04/1999 v0.1.1 - `adcn`과 `sdct`를 레드햇 5.2에 맞게 변형했으며 이스크립트는 이문서의 양을 줄이기 위해 생략했음.
- 12/04/1999 v0.1.0 - Douglas Eadline의 *Cluster Quick Start*문서를 포함했음.