COCOA Beowulf Cluster FAQ

Anirudh Modi anirudh-modi@psu.edu 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 문 는 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 개요 **1**

제 2 절 COCOA제작에 관한 상세사항 **2**

2.1 하드웨어 설치 2

2.2 소프트웨어 설치 2

제 1 절 개요

무엇이 COCOA인가?

COCOA는 **CO**st effective **CO**mputing **A**rray의 약자로서 Beowulf 부류의 수퍼컴퓨터입니다. Beowulf는 병렬처리에 사용될 수 있는 다중 컴퓨터 구조를 갖고 있다. 이는 하나의 서버노드와 이더넷 카드나 다른 빠른 네트워크로 연결된 하나 이상의 클라이언트 노드들로 이루어져있다. 상용화되어 있는 표준 이더넷 어댑터와 스위치를 장착한 데스크탑 PC와 같은 상용 하드웨어를 이용해 구축할 수 있다. 상용화 되지 않은 하드웨어를 포함하지 않으며 쉽게 제작할 수 있다. <http://cocoa.ihpca.psu.edu/>

COCOA를 만드는데 어떠한 하드웨어가 필요한가?

Dell <http://www.dell.com>에서 나온 26 WS-410 워크스테이션들인데, 구성요소는 다음과 같다.

1. Dual 400 MHz Intel Pentium II Processors w/512K L2 cache
2. 512 MB SDRAM
3. 4 GB UW-SCSI2 Disk
4. 3COM 3c509B Fast Ethernet adapter (100 Mbits/sec)
5. 32x SCSI CD-ROM drive
6. 1.44 MB floppy drive
7. Cables

추가로 다음의 내용도 사용되었다:

1. 한개의 Baynetworks 450T 24-way 100 Mbits/sec switch
2. 두개의 12-way 모니터/키보드/마우스 switches
3. 4개의 500 kVa Uninterruptible Power Supplies from APC

4. 한개의 모니터, 키보드, 마우스와 서버 노드에 연결한 54 GB의 UW-SCSI2 하드 디스크

COCOA의 운영체제는 무엇인가?

Linux! 특별히 레드햇 리눅스 5.1을 사용했다. <http://www.redhat.com>리눅스는 자유롭게 사용할 수 있는 운영체제이고 모든 i386과 호환되는 컴퓨터에서 작동할 수 있다.(지금은 PowerPC, Alpha, Sparc, Mips, Ataris와 Amigas에서도 작동한다.) 리눅스 커널은 Linus Torvalds torvalds@transmeta.com과 다른 자원봉사자들에 의해 만들어 졌다. 리눅스에서 구동되는 여러 프로그램은 일반적으로 유닉스 프리웨어이며 대부분의 경우 GNU 프로젝트에서 만들어졌다.

COCOA에는 어떤 소프트웨어가 설치되었나?

서버에는 다음과 같은 소프트웨어가 설치되었다.

1. 레드햇 5.1에서 제공하는 기본 꾸러미들 <http://www.redhat.com>
2. GNU C/C++ 컴파일러와 펜티엄에서 최적화된 GNU C/C++ 컴파일러 (*gcc, pgcc*)
3. Fortran 77/90 컴파일러와 Portland 그룹이 제공하는 디버거
4. C/C++, Fortran77/90에서 병렬처리를 할 수 있게 만들어주는 **M**essage **P**assing **Interface** (MPI) 라이브러리
5. Amtec Corporation <http://www.amtec.com>에서 제공하는 TECPLOT라는 Scientific Visulaization 소프트웨어

COCOA는 얼마인가?

대략 100,000불이다.

제 2 절 COCOA제작에 관한 상세사항

2.1 하드웨어 설치

하드웨어를 설치하는 것은 간단히 알 수 있다. 주요과정은 다음과 같다:

1. 사용할 머신들을 박스에서 꺼내서 설치할 랙에 집어 넣고 번호를 붙인다.
2. 24-포트 네트워크 스위치를 설치하고 10Mbit 포트중에 하나를 서버의 두번째 이더넷 어댑터에 연결하여 사설 네트워크를 만든다. 나머지 23개의 포트는 클라이언트의 이더넷 어댑터에 연결한다. 2개의 포트를 갖고 있는 확장 카드를 남아있는 2개의 클라이언트를 연결할 스위치에 추가한다.
3. 16-way 모니터/키보드를 설치하고 화면출력 부분과 25개 머신의 각각의 키보드 케이블을 연결하고 서버를 연결한다.
4. 4개의 UPS에 파워를 연결한다.

2.2 소프트웨어 설치

여기서 부터 실제적인 작업에 들어갑니다! 주요과정:

1. 서버를 우선 설치한다. 레드햇 5.1을 CD-ROM을 이용하여 설치한다. 대부분의 하드웨어는 자동적으로 감지되며 (네트워크 카드를 포함하여) 중요한 점은 드라이브를 분할하고 설치에 필요한 꾸러미를 선택한다. 3 GB root 파티션은 시스템 파일 생성과 설치할 패키지를 위해 만들어 둔다. 두개의 128 MB의 스왑 파티션을 생성시키고 나머지 50 GB는 여러사용자를 위해 남겨둔다. /tmp를 위해 1 GB를 할당하는 것이 좋은 생각이라는 것을 나중에 깨달을 것이다.

2. 최신의 리눅스 커널(여기서는 #2.0.36)을 내려받은 후 펜티엄 GNU CC 컴파일러 pgcc <http://www.goof.com/pcg/> (펜티엄 II에 최적화된 코드를 생성시킴)를 이용하여 SMP 지원과 하드웨어에 알맞은 선택사항을 고려하여 커널을 컴파일한다. 다음의 최적화 선택사항들이 사용된다: pgcc -mpentiumpro -06-fno-inline-functions SMP 지원을 사용하는 것은 커널 설정 메뉴에서 볼 수 있는 *Processor type and features*에 있는 버튼을 사용함으로써 가능하다.(make xconfig를 실행하면서 시작된다.

3. 리눅스를 위한 새로운 커널-공간 NFS 서버 (*knfsd*) <http://www.csua.berkeley.edu/~gam3/knfsd/>가 향상된 NFS 성능을 얻기위해 그전에 설치되었던 사용자-공간 NFS 서버를 대체하여 설치되어 있다. 빠른 설치를 바라는 경우에는 레드햇 RPM 꾸러미를 <http://rufus.w3.org/linux/RPM/>에서 구하여 설치하면 된다. 디폴트 선택사항이 사용된다.

4. ssh는 <http://www.cs.hut.fi/ssh/>에서 구할 수 있고 컴파일하여 설치할 수 있다. (외부로 부터의 안전한 접속을 위해서) ssh-1.2.26는 ssh-2.0.11보다 선호된다. 왜냐하면 ssh v2.x는 역방향으로 호환성이 없을뿐더러 많이 느리기 때문이다. sshd 데몬은 /etc/rc.d/rc3.d에서 runlevel 3에서 시작한다. 최근에 ssh레드햇 RPM이 <http://rufus.w3.org/linux/RPM/>에서 제공되고 있고 다른 RPM 들도 설치하기가 편해지고 있다.

5. 두개의 3c905B 이더넷 어댑터들이 설정되었다; 한개는 eth1으로서 IP를 128.118.170.11로 설정되어있고 다른 하나는 사설 네트워크에 연결되어서 dummy IP 주소를 10.0.0.1로 할당되어 있다. Donald Becker에 의해 만들어진 3COM 3c905B 어댑터에 관한 최신 드라이버는 (3c59x.c v0.99H <http://cesdis.gsfc.nasa.gov/linux/drivers/vortex.html>) 커널내부로 컴파일되어 100Mbit/sec Full-duplex 연결을 보장한다. 이는 vortex-diag 유틸리티를 이용하여 <http://cesdis.gsfc.nasa.gov/linux/diag/vortex-diag.c>를 이용하여 확인할 수 있다. 설정을 위해서 다음 파일을 변경해야한다: /etc/sysconfig/network, /etc/sysconfig/network-scripts/ifcfg-eth0와 /etc/sysconfig/network-scripts/ifcfg-eth1. 여기에 변경후 모습이다:

   /etc/sysconfig/network:

```
NETWORKING=yes
FORWARD_IPV4=no
HOSTNAME=cocoa.ihpca.psu.edu
DOMAINNAME=ihpca.psu.edu
GATEWAY=128.118.170.1
GATEWAYDEV=eth1
NISDOMAIN=ihpca.psu.edu
```

   /etc/sysconfig/network-scripts/ifcfg-eth0:

```
DEVICE=eth0
IPADDR=10.0.0.1
NETMASK=255.255.255.0
NETWORK=10.0.0.0
BROADCAST=10.0.0.255
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
```

   /etc/sysconfig/network-scripts/ifcfg-eth1:

```
DEVICE=eth1
IPADDR=128.118.170.11
```

```
NETMASK=255.255.255.0
NETWORK=128.118.170.0
BROADCAST=128.118.170.255
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
```

6. 쉬운 설치를 위해, BOOT 프로토콜을 이용해서 네트워크로부터 PC 하나하나를 부팅하기로 결정했다. BOOTP 서버는 /etc/inetd.conf파일의 다음 줄의 주석을 없애줌으로써 가능하다.

```
bootps  dgram   udp     wait    root    /usr/sbin/tcpd  bootpd
```

리눅스 부팅 플로피는 3c905B 네트워크 어댑터를 지원하는 커널을 이용해서 준비할 수 있고 이 어댑터는 각 클라이언트 노드들이 부팅이 되고 각자 고유의 96-bit 네트워크 주소글 알려준다. (에를 들어 00C04F6BC052) 이 주소를 이용하면 /etc/bootptab파일에 다음과 같이 설정해야한다:

```
.default:\
        :hd=/boot:bf=install.ks:\
        :vm=auto:\
        :dn=hpc.ihpca.psu.edu:\
        :gw=10.0.0.1:\
        :rp=/boot/client/root:

node1:ht=ethernet:ha=00C04F6BC0B8:ip=10.0.0.2:tc=.default
node2:ht=ethernet:ha=00C04F79AD76:ip=10.0.0.3:tc=.default
node3:ht=ethernet:ha=00C04F79B5DC:ip=10.0.0.4:tc=.default
.
.
.
node25:ht=ethernet:ha=00C04F79B30E:ip=10.0.0.26:tc=.default
```

7. /etc/hosts파일은 다음과 같이 편집한다:

```
127.0.0.1       localhost       localhost.localdomain
# Server [COCOA]
128.118.170.11 cocoa.ihpca.psu.edu cocoa.aero.psu.edu cocoa

# IP address <--> NAME mappings for the individual nodes of the cluster
10.0.0.1        node0.hpc.ihpca.psu.edu node0           # Server itself!
10.0.0.2        node1.hpc.ihpca.psu.edu node1
10.0.0.3        node2.hpc.ihpca.psu.edu node2
.
.
.
10.0.0.26       node25.hpc.ihpca.psu.edu node25
```

/etc/hosts.conf파일에는 다음과 같은 내용을 포함하게 한다:

```
order hosts,bind
```

이는 DNS 서버로 부터 어떠한 요청이 있기전에 /etc/hosts파일에 있는 IP 주소를 강제로 찾게 해준다.

8. 전달할 파일시스템들은 /etc/exports에 다음과 같이 추가하면 된다:

```
/boot           node*.hpc.ihpca.psu.edu (ro,link_absolute)
/mnt/cdrom      node*.hpc.ihpca.psu.edu (ro,link_absolute)
/usr/local      node*.hpc.ihpca.psu.edu (rw,no_all_squash,no_root_squash)
/home1          node*.hpc.ihpca.psu.edu (rw,no_all_squash,no_root_squash)
/home2          node*.hpc.ihpca.psu.edu (rw,no_all_squash,no_root_squash)
/home3          node*.hpc.ihpca.psu.edu (rw,no_all_squash,no_root_squash)
/home4          node*.hpc.ihpca.psu.edu (rw,no_all_squash,no_root_squash)
```

9. 각 클라이언트 노드에 빠르고 안정적인 설치를 위해, 레드햇 5.1 KickStart 설치가 이상적이다. 본 저자의 kickstart 파일 /boot/install.ks은 다음과 같다:

```
lang en
network --bootproto bootp
nfs --server 10.0.0.1 --dir /mnt/cdrom
keyboard us
zerombr yes
clearpart --all
part / --size 1600
part /local --size 2048
part /tmp --size 400 --grow
part swap --size 127
install
mouse ps/2
timezone --utc US/Eastern
rootpw --iscrypted kQvti0Ysw4r1c
lilo --append "mem=512M" --location mbr
%packages
@ Networked Workstation
%post
rpm -i ftp://10.0.0.1/pub/CLUSTER/RPMS/wget-1.5.0-2.i386.rpm
rpm -i ftp://10.0.0.1/pub/CLUSTER/RPMS/xntp3-5.93-2.i386.rpm
/usr/bin/wget ftp://10.0.0.1/pub/CLUSTER/kernel/vmlinuz -O/boot/vmlinuz
/usr/bin/wget ftp://10.0.0.1/pub/CLUSTER/conf/lilo.conf -O/etc/lilo.conf
/sbin/lilo
/usr/bin/wget ftp://10.0.0.1/pub/CLUSTER/conf/hosts.equiv -O/etc/hosts.equiv
sed "s/required\(.*securetty\)/optional\1/g" /etc/pam.d/rlogin > /tmp/rlogin
mv /tmp/rlogin /etc/pam.d/rlogin
```

레드햇 KickStart 설치에 대해 더욱더 많은 정보는 <http://www.cache.ja.net/dev/kickstart/KickStart-HOWTO.html>에서 얻을 수 있습니다.위에서 언급한 설치명령중에서 /etc/pam.d/rlogin의 처음 라인은 다음과 같이 변경한다:

```
auth       optional     /lib/security/pam_securetty.soauth       optional     /lib/security
```

이것은 rlogin/rsh를 동작시켜 서버에서 클라이언트로 패스워드 없이 연결하게 한다. 이는 클라이언트 노드의 소프트웨어 관리에 좋은 방법이다. 또한 /etc/hosts.equiv파일은 다음과 같다:

```
node0
node1
node2
```

```
node3
.
.
.
node25
```

레드햇 리눅스 5.1 CD-ROM을 NFS를 각 클라이언트에 전달하는 서버에서 /mnt/cdrom으로 마운트 시키고 SMP를 지원하는 새로운 커널을 서버와 같은 방법으로 컴파일하고 레드햇 부트 디스켓에 있는 기존의 커널을 바꾼다. 이 커널은 클라이언트 노드에서 사용하기 때문에 특별히 많은 옵션이 필요하지 않다. 더우기 BOOTP를 이용해서 커널 레벨 자동설정은을 위한 옵션은 커널설정메뉴에서 네트워킹 선택사항에서 가능하게 할 수 있다. 이는 노드가 자동적으로 부팅과정중에 서버로 부터 ip를 부여받을 때 요구된다. 부트디스의 설정파일에 관한 지원은 KickStart 모드에서 직접적으로 부팅하기위해서 변형된다. 각 클라이언트 설정을 위해 필요한 모든 것은 부트 디스크를 집어넣고 워크스테이션에 전원을 키고 자동설정이 끝날 때까지 기다리면 된다.

10. 모든 클라이언트들이 설치후 부팅되자마자, 클러스터는 작동을 하게된다. `brsh`(<http://www.beowulf.org/software/RPMS/beobase-2.0-1.i386.rpm>)와 같은 유용한 유틸리티들은 각 클라이언트 노드들에게 `rsh`를 이용해 동일한 명령을 내리도록하기위해 설치된다. NIS는 각 클라이언트 노드에 사용자 로그인을 관리하기 위해 설치될 수 있지만 대신에 간단한 쉘 스크립트를 이용해서 서버의 `/etc/passwd`, `/etc/shadow`와 `/etc/group`파일을 분배할 수 있다.

11. 대부분의 서비스들은 각 클라이언트 노드들에 대해 `/etc/inetd.conf`에서 사용이 불가능하게 되고 각 클라이언트 노드들은 다음과 같이 보이게 된다:

```
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
auth    stream  tcp     nowait  nobody  /usr/sbin/in.identd in.identd -l -e -o
```

12. *automount* 꾸러미는 각 노드에 설치하여 요구할 때 다양한 사용자 파티션을 자동적으로 마운트 시켜준다. 이는 조금 향상된 NFS 성능을 보이지만, 버그가 많고 불안정하다. 마지막으로 리눅스를 위한 *automount*는 준비가 아직 되어있지 않고 일반적인 NFS 마운트를 위해 제거된다.

13. MPI-CH 소스 코드는 <http://www.mcs.anl.gov/mpi/>에서 받을 수 있으며 `pgcc`를 이용하여 컴파일하면 된다. `/usr/local/` 파티션에 설치하는 것은 큰 문제가 없을 것이다. `mpif77`스크립트는 사용자의 요구에 맞게 변형되었고 비슷한 `mpif90`또한 만들어 졌다. `/usr/local/mpi/util/machines/machines.LINUX`에 각 클라이언트에 대해 두가지 입력을 하면된다. (dual-processor SMP 노드를 사용하는 경우도 마찬가지다.) `mpirun`을 이용하여 프로그램을 수행한다.

14. queing 시스템, DQS v3.0은 <http://www.scri.fsu.edu/~pasko/dqs.html>에서 다운로드 할 수 있고 컴파일하여 `/usr/local/DQS/`에 설치되고 NFS를 이용하여 모든 클라이언트를 이용해서 접근가능 하게 한다. 적절한 서버와 클라이언트 변화들은 `/etc/services`의 서비스들을 첨가하고 서버에서 `qmaster`를 클라이언트에서는 `dqs_execd`를 시작하여 작동하게 할 수 있다. 하지만 몇몇 문제점들이 닥칠 수도 있다. 이는 DQS의 잘못된 문서화때문이다. 본저자가 DQS 설정을 하여 클라이언트 노드가 인식하도록 하기까지 너무 오랜 시간이 걸렸지만, 한번 설정이 끝나면 나머지는 쉽게 할 수 있다. Wrapper 쉘 스크립트는 DQS 결과를 잘만들어 내게하고 조금의 향상을 시키는 데 필요한 `qsub`, `qstat`와 `qdel`를 위해 본 저자가 변형되었다. 예를 들어 `qstat`은 변형을 하여 queue에 있는 기급한 작업에 의해 요청되는 노드의 수를 보여주게 만들었다. 또한 다른 세가지 `qinfo`,`qload`와 `qmem`는 각 노드들을 위해서 유용한 load 데이터를 주게 만들었고 클러스터 리소스 이용에 사용되게 만들어졌다.

15. COCOA는 완전히 기능적이고 벤치마킹과 많은 병렬처리를 위해 준비가 되어있다. 커널을 가지고 `pgcc`컴파일러를 이용하는 것이 C/C++ 코드를 위해 권장된다. 특히 전형적인 FPU intensive number crunching 노드를 위해 `pgcc`를 `-mpentiumpro -06 -funroll-all-loops`옵션를 이용하면 `gcc`를 이용하는 것 보다 실행시간이 30% 향상을 가져온다.

이 문서는 Anirudh Modi <htmlurl url="mailto:anirudh-modi@psu.edu" name="anirudh-modi@psu.edu">에 의해 유