학사 학위논문

고해상도의 수치모의를 위한 리눅스 병렬 시스템 구축

2001년 11월

경 북 대 학 교

천 문 대 기 과 학 과

이 정 훈 (starwave@hanmail.net)

지 도 교 수 : 박 명 구

<제목 차례>

1. 서론

MM5(Fifth generation Mesoscale Model)는 PSU/NCAR(Pennsylvania State University - National Center for Atmospheric Research)에서 개발된 중규모 기상 수치 모델로, 1970년대 개발된 이후로 많은 변화를 거쳐 현재 다양한 분야에서 기본 모델로 활용되고 있다. 고해상도의 수치 모의에 요구되는 시스템 사양과 수행시간 때문에, MM5는 주로 고가의 Cray 시스템에서 운영되었지만 최근 PC(Personal Computer) 기술의 급격한 발달과 PC 환경에서 Unix 환경을 제공해주는 운영체제인 Linux의 개발로 PC와 같은 저가 시스템에서도 이용할 수 있게 되었다. 그러나 단일 PC 시스템으로 MM5를 이용한 수치 모의를 하기에는 운영시간이 지나치게 길다는 단점이 여전히 존재한다. 이러한 단점을 보완할 수 있는 한 방법으로서 높은 사양의 시스템을 사용할 수도 있겠지만, 현재 전 세계적으로 각종 연구 분야에서 각광받고 있는 PC 클러스터링 기술이 보다 근본적인 대안이다.

MM5는 1998년 2.8버전부터 IBM SP, Cray T3E, Fujitsu VPP, PC(또는 워크스테이션) Beowulf 클러스터와 다중 프로세서의 분산 메모리 클러스터와 같은 분산 메모리(Distributed Memory - DM) 병렬 컴퓨터를 지원하고 있다. MM5의 분산 메모리 기능은 MPI(Message Passing Interface)를 이용해 구현되었다. 이에 저가 고효율 병렬 시스템인 리눅스를 이용하여 병렬 시스템을 구축하였고, 실제 성능을 테스트 해 보았다.

2. System 구성 및 소프트웨어

2.1 System 구성도

각 노드별로 하나의 CPU를 가진 컴퓨터로 4대를 구성하였다. Master Node에는 4개의 Ethernet Card를 가지고 있고, 나머지 Slave Node에는 각 3개의 Ethernet Card를 가지고 있다. 3개의 8Port Switching HUB를 사용하여 네트워크 병목현상을 줄였다. Master Node에는 SCSI HDD, IDE HDD, CD-ROM, Graphic Card등 모든 장치가 있지만, Slave Node에는 HDD, CD-ROM등을 제거하였다. (Fig 1)(Fig 2)(Fig3)

2.2 시스템 사양

시스템 선정시 많은 고민이 있었지만, 최근 펜티엄(Pentium) 4보다 뛰어난 성능을 발휘하는 AMD 애슬론(Athlon) 1.4를 사용하였다. 애슬론 CPU는 펜티엄 4에 비해 저렴한 가격과 더욱 뛰어난 계산 능력을 보여주고 있어 선택하였고, 매인보드(MainBoard)는 ASUS A7V133을 선택하였다. SCSI Controller는 Adaptec 7892A를, HDD는 IBM 10000rpm 18GByte하드를 선택하였으며, Slave Node와 함께 구입한 IDE 하드디스크는 시게이트 바라쿠다(Seagate Barracuda) ATA IV를 구입하였다. 랜 카드(Lan Card)는 3Com 10/100Mbps를 외부 인터넷용으로, Intel Express 10/100Mbps를 내부 NFS용으로, Realtek 8139 10/100Mbps 각2개씩을 계산용으로 사용하였다.(Fig 4)

2.3 사용 소프트웨어

O/S(Operation System)는 RedHat 7.1을 설치하였고, 최신 커널인 2.4.13을 사용하였다. 컴파일러(Compiler)는 Gnu Compiler와 PGI f77컴파일러, MPICH를 사용하였다. MM5는 MM5V3를 사용하였으며, Diskless를 위해서 etherboot를 사용하였다.(Fig 5)

Fig 1 System 구성

Fig 2 Slave Node 내부

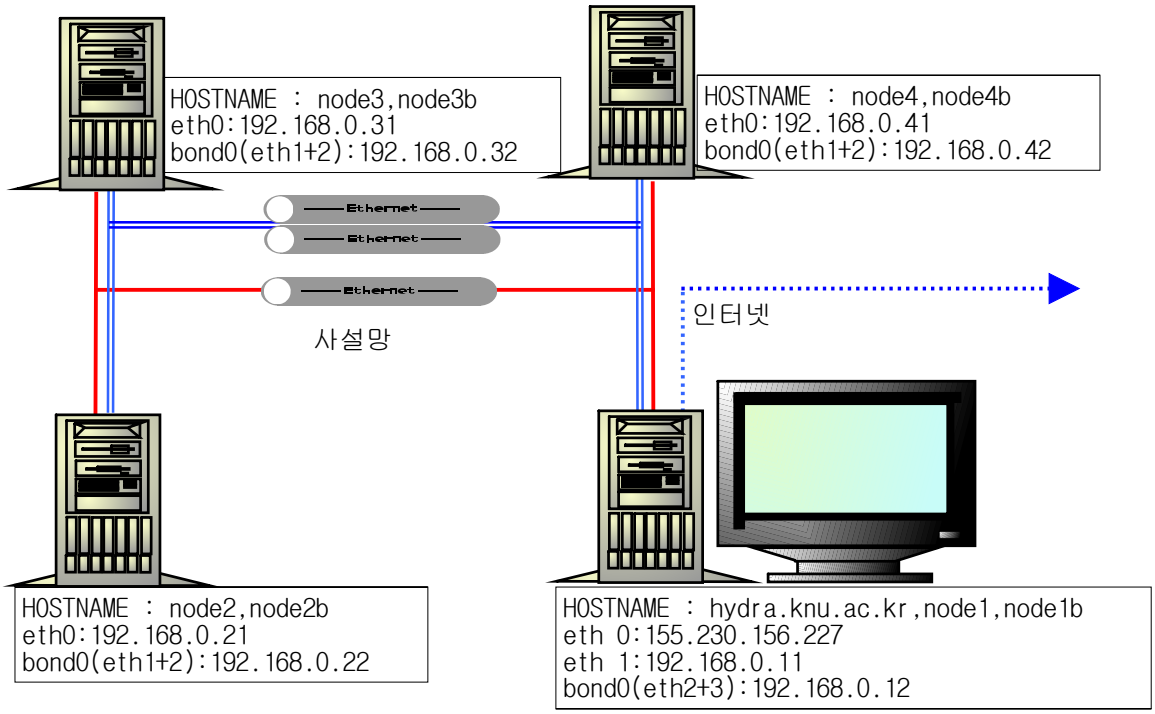


Fig 3 System 구성도

	Master Node	Slave Node
CPU	AMD Athlon(Thunderbird) 1.33@1.4Ghz	AMD Athlon(Thunderbird) 1.4Ghz
Memory	756MB(PC133 SDRAM)	512MB(PC133 SDRAM)
HDD	19GB SCSI 40GB IDE X 3 (40GB X 2=Software RAID)	
LAN	3Com 10/100Mbps X 1 Intel 10/100Mbps X 1 Realtek 10/100Mbps X 2	Intel 10/100Mbps X 1 Realtek 10/100 X 2
HUB	ReeNet 10/100Mbps Switching 8Port X 3	

Fig 4 각 Node별 System 사양

	Software	Web Site
O/S	RedHat 7.1 (Kernel : 2.4.13)	www.redhat.com
Compiler	Gnu compiler 2.95.3	bbs.hancom.com/databbs/pds15_files/gcc-2.95.3.tar.gz
	PGI Workstation 3.2	www.pgroup.com
	MPICH 1.2.2.2	www-unix.mcs.anl.gov/mpi/mpich/
MM5	MM5V3 + MMP MM5	www.mmm.ucar.edu/mm5/mm5-home.html
Diskless	etherboot-5.0.4	etherboot.sourceforge.net/doc/html/userman.html

Fig 5 사용 Software

3. MM5 설치

3.1 Linux 설치 및 환경설정

3.1.1 Linux 설치방법

Linux 설치 시 기본적인 프로그램을 설치하고, nfs, rsh, rlogin, Kernel Development, GCC 관련 프로그램도 함께 설치하여야 한다.

Network은 Master Node에서 eth0는 외부 인터넷을 eth1에는 내부 인터넷을 연결하고, eth1의 네트워크 환경을 아래와 같이 설정한다.

IP : 192.168.0.1

SUBNET : 255.255.255.0

네트워크 : 192.168.0.0

브로드케스트 : 192.168.0.255

3.1.2 GCC 설치

RedHat 7.1에는 GCC ver2.6버전이 들어있다. ver2.96은 GCC 정식 버전이 아닌 테스트 버전으로 MPICH test시에 에러가 발생한다. 따라서 안정 버전인 2.95.3으로 다시 설치하여야 한다.

먼저 http://bbs.hancom.com/databbs/pds15_files/gcc-2.95.3.tar.gz(한글과 컴퓨터)에서 GCC와 라이브러리가 함께 들어있는 프로그램을 다운로드 받는다.

rpm -Uvh --nodep --force [Package_Name]으로 컴파일러와 라이브러리를 설치한다.

3.1.3 rsh, rlogin 설정

/etc/xinetd.d/rsh 파일을 열어 disable = yes를 disable = no로 변경하고, 같은 방법으로 /etc/xinetd.d/rlogin 파일을 열어 disable = yes를 disable = no로 변경한다. 그런 후 /etc/rc.d/initd.d/xinetd restart하여 xinetd 데몬을 다시 실행한다.

/etc/hosts 파일을 열어 각 노드들을 등록해 준다.

example) /etc/hosts

127.0.0.1 hydra.knu.ac.kr hydra localhost.localdomain localhost

155.230.156.227 hydra.knu.ac.kr hydra

192.168.0.1 node1

192.168.0.2 node2

192.168.0.3 node3

192.168.0.4 node4

/etc/hosts.equiv 파일을 열어 각 노드들에게 rsh로 접속 할 수 있게 한다.

example) /etc/hosts.equiv

hydra

node1

node2

node3

node4

3.2 PGI Workstation(Trail Version) 설치

사용자 shell은 csh로 변경한다.

ftp://ftp.pgroup.com/x86/linux86-HPF-CC.tar.gz에서 PGI Workstation을 다운로드 받고, /tmp나 기타 디렉토리에 linux86-HPF-CC.tar.gz파일의 압축을 푼다.

install파일을 실행하여 프로그램을 설치한다.

설치 경로 : /usr/local/pgi

설치후, license.dat 파일을 출력을 하여 보관하도록 한다.

/etc/csh.cshrc에 경로를 추가한다.

setenv PGI /usr/local/pgi

set path = ($PGI/linux86/bin $path)

다시 로그인하거나, 현재 path에도 위의 경로를 추가해준다.

3.3 MPICH 설치

http://www-unix.mcs.anl.gov/mpi/mpich/index.html에서 mpich-1.2.2.2를 다운 받고, mpich.tar.gz파일의 압축을 푼 후 mpich-1.2.2.2디렉토리로 들어간다.

setenv FC /usr/local/pgi/linux86/bin/pgf77

setenv FLINKER /usr/local/pgi/linux86/bin/pgf77

./configure -prefix=/usr/local/mpich-1.2.2.2

을 한다.

그런 후 Makefile에 NOFF77=O인지 확인하고, O이 아니면 위의 과정을 다시 확인해 본다.

make

make testing -> 이 과정에서 에러가 나는지 잘 확인해야 한다. (GCC compiler를 2.96을 사용 하면 여기서 에러가 발생한다.)

make install PREFIX=/usr/local/mpich-1.2.2.2

/etc/csh.cshrc에 경로를 추가한다.

set path = (/usr/local/mpich-1.2.2.2 $path)

/usr/local/mpich-1.2.2.2/share/machines.LINUX파일을 열어, 각 노드들의 이름을 입력한다.

example) /usr/local/mpich-1.2.2.2/share/machines.LINUX

hydra

node2

node3

node4

3.4 MM5-MPP설치

http://www.mmm.ucar.edu/mm5/에서 MM5V3와 MPP를 다운로드 받는다.

MM5.tar.gz파일의 압축을 푼다.

MPP.tar.gz파일을 MM5/디렉토리 밑에 푼다.

configure.user를 configure.user.unix로 변경하고, configure.user.linux를 configure.user로 변경한다.

configure.user.unix에서 「7. MPP options」 부분과 「7g. Linux PCs. Need Portland Group pgf77 and MPICH.」 부분을 복사해서 configure.user의 「6. Physics options」 다음 부분에 추가한다.

configure.user의 「7g. ...」에서

LOCAL_LIBRARIES = -L$(LINUX_MPIHOME)/build/LINUX/ch_p4/lib -lfmpich -lmpich 를

LOCAL_LIBRARIES = -L$(LINUX_MPIHOME)/lib -lfmpich -lmpich 로 수정하고,

CFLAGS = -DMPI -I/usr/local/mpi/include 를

CFLAGS = -DMPI -I/usr/local/mpich/include 로 수정한다.

/home/mm5/MM5/MPP/RSL/RSL/로 이동 후,

makefile에서 다음과 같이 수정한다.

linux :

$(MAKE) -f makefile.linux LINUX_MPIHOME=$(LINUX_MPIHOME) $(MAKE_OPTS) all 를

linux :

$(MAKE) -f makefile.linux LINUX_MPIHOME=/usr/local/mpich $(MAKE_OPTS) all 로 수정한다.

MM5 최상위 디렉토리로 이동한 다음 *make mpp*를 하여 Run/mm5.mpp를 생성하고, make mm5.deck을 하여 mm5.deck을 생성한후, *./mm5.deck*을 실행하여 mm5.deck을 적용시킨다.

모델 초기 자료(MM5 HomePage에서 다운 받는 test 자료 - input2mm5.tar.gz)를 Run/ 디렉토리로 옮기거나 링크 시킨다.

3.5 Test

~mm5/MM5/Run에서

mpirun -np 1 mm5.mpp 로 MM5를 테스트 해 볼 수 있다.

실행시간을 측정해 보려면,

time sh -c 'mpirun -np 1 mm5.mpp' 로 테스트 해 볼 수 있다.

example) $ time sh -c 'mpirun -np 1 mm5.mpp'

hydra.knu.ac.kr -- rsl_nproc_all 1, rsl_myproc 0

619.870u 2.680s 10:53.27 95.2% 0+0k 0+0io 4728pf+0w

*총 소요시간은 10분 53초이고, 평균적으로 95.2%의 CPU를 사용하였다.

3.6 Slave Node 설정

3.6.1 Linux기본 설치

Slave Node에는 kernel Development와 GCC 컴파일러는 필요치 않지만, nfs, rsh, rlogin은 반드시 포함시켜야 한다.

3.6.1.1 rsh, rlogin

Master Node와 같이 설정한다.

/etc/hosts, /etc/hosts.equiv파일도 동일하게 한다.

mm5계정으로 로그인하여 rsh node2와 같이 각 노드들에게 rsh로 접속이 가능한지, 확인한다.

3.6.1.2 NFS

nfs를 설치하기 위해서는 Master Node에서 /etc/exports 파일을 열어 각 노드들에게 nfs로 마운트 할 수 있게 한다.

example) /etc/exports

/home node2(rw,no_root_squash,no_all_squash)

/home node3(rw,no_root_squash,no_all_squash)

/home node4(rw,no_root_squash,no_all_squash)

/usr/local node2(rw,no_root_squash,no_all_squash)

/usr/local node3(rw,no_root_squash,no_all_squash)

/usr/local node4(rw,no_root_squash,no_all_squash)

Slave Node에서는 /etc/fstab파일에서 Master Node의 /home /usr/local 디렉토리를 마운드 한다.

example) /etc/fstab

192.168.0.1:/home /home nfs hard,intr,rw

192.168.0.1:/usr/local /usr/local nfs hard,intr,rw

3.6.2 계정관리

Master Node의 mm5계정과 Salve node의 계정의 UID와 GID가 같은지 확인하고, 기본 SHELL을 역시 csh로 변경하고, /etc/csh.cshrc을 동일하게 만든다.

3.7 TEST

Master Node에서 nfs데몬은 다시 실행하고, Slave Node는 /home과 /usr/local을 마운트 하거나 리부팅 한다.

Master Node에서 ~mm5/MM5/Run에서

MM5을 실행한다.

mpirun -np 4 mm5.mpp

example) $ time sh -c 'mpirun -np 4 mm5.mpp'

[mm5@hydra Run]$ time `mpirun -np 4 mm5.mpp`

hydra.knu.ac.kr -- rsl_nproc_all 4, rsl_myproc 0

node3 -- rsl_nproc_all 4, rsl_myproc 2

node2 -- rsl_nproc_all 4, rsl_myproc 1

node4 -- rsl_nproc_all 4, rsl_myproc 3

208.500u 7.350s 4:38.59 77.4% 0+0k 0+0io 9410pf+0w

*총 소요 시간은 4분 38분이고, 평균적으로 77.4%의 CPU를 사용하였다.

4. Channel Bonding 하기

4.1 Channel Bonding이란?

2개 이상의 Ethernet을 하나로 묶어 네트워크의 대역폭을 늘리는 방법이다.

4.2 Lan card 설치하기

인터넷에 연결된 랜카드를 eth0으로, bonding을 할 Lan을 eth1과 eth2로 설정하였다.

*A7V133의 경우 같은 랜카드를 2개 설정하면 AGP슬롯에 가까운 쪽부터 먼저 인식한다. 예를 들어 AGP슬롯에 가까운 쪽에 3Com 랜카드가 그 다음에 Realtek Lan이 있다면, 3Com 랜카드를 eth0으로 Realtek 랜카드를 eth1로 인식한다.

HUB를 2개 사용하여 HUB1에는 Master Node의 eth1과 Slave Node의 eth0을 연결하고, HUB2에는 Master Node의 eth2과 Slave Node의 eth1을 연결하였다.

4.3 Kernel 컴파일

Channel Bonding을 하기 위해서는 커널을 컴파일 해야 한다.

Network device support --->

<*> Bonding driver support

4.4 설정하기

먼저 iputil RPM Package가 설치되어 있는지 확인한다.

4.4.1 /etc/sysconfig/network

기존 설정은 그대로 두고,

GATEWAYDEV=bond0

를 추가한다.

4.4.2 /etc/sysconfig/network-scripts/ifcfg-bond0

기존의 ifcfg-eth0의 설정을 유지하면서

DEVICE=eth0를 DEVICE=bond0로 변경한다.

example) /etc/sysconfig/network-scripts/ifcfg-bond0

DEVICE=bond0

USERCTL=no

ONBOOT=yes

BOOTPROTO=none

BROADCAST=192.168.0.255

NETWORK=192.168.0.0

NETMASK=255.255.255.0

IPADDR=192.168.0.12

4.4.3 /etc/sysconfig/network-scripts/ifcfg-[eth0,eth1]

DEVICE=[eth0,eth1] -> Master Node = eth1, Slave Node = eth0

USERCTL=no

ONBOOT=yes

MASTER=bond0

SLAVE=yes

BOOTPROTO=none

4.4.4 /etc/sysconfig/network-scripts/ifcfg-[eth1,eth2]

DEVICE=[eth1,eth2] -> Master Node = eth2, Slave Node = eth1

USERCTL=no

ONBOOT=yes

MASTER=bond0

SLAVE=yes

BOOTPROTO=none

4.5 TEST

Channel Bonding을 하기 전에 network데몬을 정지시켰으면, network데몬을 다시 시작하면 되고, 그렇지 않으면 리부팅한다.(다른 방법도 있다.)

Master Node와 Salve Node 모두 Channel Bonding이 되어 있는지 확인한다.

ifconfig을 실행하여 bond0가 있는지 확인하고, Slave와 ftp등을 이용해서 파일을 다운로드 받아본다.

FTP로 파일 전송한 결과

Channel Bonding 전 : 1.1e+03 Kbytes/sec

Channel Bonding 후 : 2.2e+03 Kbytes/sec

5. Diskless 하기

5.1 Diskless란?

Slave Node에 디스크를 두지 않고, Network을 이용하여 부팅 하는 것을 말한다.

5.2 필요 프로그램

기존에 NFS, DHCP, tftp가 설치되어 있는지 확인한다.

etherboot, mknbi (http://http://etherboot.sourceforge.net/)

*kernel컴파일만으로 Diskless를 구현 할 수 있지만, Lan Card의 EPPROM을 사용할 경우 EPPROM의 한계인 512K에 커널 크기를 맞춰야 하기 때문에 제약사항이 많다. 따라서 etherboot을 이용하였다.

5.3 etherboot 설정

http://sourceforge.net/projects/etherboot에서 etherboot와 mknbi을 다운받는다.

etherboot의 압축을 풀고, mknbi는 RPM으로 설치한다.

cd etherboot/src

make

make bin/boot1a.bin

Floppy Disk를 넣고,

cat bin/boot1a.bin bin32/eepro100.lzrom > /dev/fd0 #FloppyDisk에 이미지 생성

*eepro100.lzrom은 INTEL Ethernet Pro 100 - 82559 Chip을 위한 이미지다.

*realtek8139의 경우 rtl8139.lzrom으로 한다.

5.4 kernel 컴파일

기본적인 설정과 함께

Networking options

-> IP:kernel level autoconfiguration

-> IP:DHCP support

-> IP:BOOTP support

-> IP:RARP support

Network device support

-> Ethernet (10 or 100Mbit)

-> 사용하는 네트워크 카드를 선택한다.(Slave Node는 반드시 커널에 포함하여야 한다.)

File systems

-> Network File System

-> NFS에 관련된 모든 기능을 선택

커널 컴파일 후 bzImage를 특정 디렉토리로 복사한다.(여기서는 /tftpboot/)

mknbi를 실행하여 kernel에게 root 디렉토리가 nfs라는 것을 알려준다.

mknbi-linux bzImage --output=vmlinuz_node2.nb --ip=192.168.0.2:192.168.0.1:192.168.0.1:255.255.255.0:node2

*--ip=ClientIP:NFS-ServerIP:gatewayIP:Netmask:ClientHostname

5.5 DHCP(Dynamic Host Configuration Protocol)설정하기

/etc/dhcpd.conf파일을 열어 편집한다.

example) /etc/dhcpd.conf

```
option broadcast-address 192.168.0.255;              #Broadcast
option domain-name-servers 192.168.0.11;             #DNS
option routers 192.168.0.11;
subnet 192.168.0.0 netmask 255.255.255.0 {
        host node2 {                                 #node2에 대한 설정
                hardware ethernet 00:03:47:7a:XX:XX; #node2에 있는 LAN의 MAC
                fixed-address 192.168.0.21;          #node2의 IP
                option root-path "/tftpboot/node2";  #node2의 root directory
                filename "vmlinuz_node2.nb";         #node2의 kernel이 있는 위치
        }
        host node3 {
                hardware ethernet 00:03:47:7a:XX:XX;
                fixed-address 192.168.0.31;
                option root-path "/tftpboot/node3";
                filename "vmlinuz_node3.nb";
        }
        host node4 {
                hardware ethernet 00:03:47:7a:XX:XX;
                fixed-address 192.168.0.41;
                option root-path "/tftpboot/node4";
                filename "vmlinuz_node4.nb";
        }
}
```

subnet 155.230.156.0 netmask 255.255.252.0 {

}

*node들의 MAC address를 얻기 위해서는 node들이 disk를 가지고 있다면 ping 테스트를 한 후 arp -a를 해보면 MAC address를 알 수 있고, diskless라면 앞에서 etherboot를 이용해 만든 floppy disk로 부팅하면 MAC address를 알 수 있다.

dhcpd데몬을 다시 시작한다.

5.6 tftp설정

/etc/xinetd.d/tftp에서

disable=yes를 disable=no로 변경한 후 xinetd데몬을 다시 시작한다.

root(/)디렉토리에 /tftpboot디렉토리를 만들고, 각 노트의 이름으로 디렉토리를 만든다.

example) ls -al /tftpboot

drwxr-xr-x 17 root root 4096 11월 5 11:15 node2

drwxr-xr-x 17 root root 4096 11월 3 12:01 node3

drwxr-xr-x 17 root root 4096 11월 3 12:01 node4

/tftpboot/nodeX/에 Master Node의 모든 파일을 권한(Permission)과 함께 복사한다.

cp -Rp / /tftpboot/nodeX

*/home과 /usr등은 제외하고 복사해도 된다. 나중에 다시 nfs로 묶어도 된다. 단, link로 연결하면 Slave들이 부팅 후 인식하지 못한다.

5.7 IP변경하기

Master Node에는 4개의 Lan Card, Slave Node에는 각 3개의 Lan Card가 장착된다. 2개의 Lan Card는 Channel Bonding을 하므로 Master Node에는 외부 IP를 포함하여 3개의 IP가 Slave Node에는 각 2개의 IP가 필요하다. Diskless를 하기 전에는 192.168.0.1, 192.168.0.2, 192.168.0.3, 192.168.0.4로 주었지만, 각 2개의 사설 IP를 주기 위해 Fig 6 과 같이 IP를 변경하였다.

Node	변경전	변경후		이름	용도
Master(node1)	155.230.156.227	eth1	155.230.156.227	hydra	외부 인터넷
	192.168.0.1	eth2	192.168.0.11	node1	NFS-ROOT
		bond0	192.168.0.12	node1b	MM5
node2	192.168.0.2	eth0	192.168.0.21	node2	NFS
		bond0	192.168.0.22	node2b	MM5
node3	192.168.0.3	eth0	192.168.0.31	node3	NFS
		bond0	192.168.0.32	node3b	MM5
node4	192.168.0.4	eth0	192.168.0.41	node4	NFS
		bond0	192.168.0.42	node4b	MM5

Fig 6 IP 변경 Table

5.8 Slave Node 설정하기

앞에서 설명한 것과 같이 Slave Node의 root 디렉토리는 Master Node의 /tftpboot/nodeX 이다.

/tftpboot/nodeX/etc 밑에 있는 파일들을 편집한다.

exports파일을 삭제한다.

5.8.1 /tftpboot/node2/etc/fstab

root를 비롯하여 /usr, /home 디렉토리를 마운트 한다.

example) /tftpboot/node2/fstab

```
192.168.0.11:/tftpboot/node2 /          nfs    defaults        0 0
192.168.0.11:/usr            /usr       nfs    defaults        0 0
192.168.0.11:/usr/local      /usr/local nfs    defaults        0 0
192.168.0.11:/home           /home      nfs    defaults        0 0
none                         /proc      proc   defaults        0 0
```

5.8.2 etc/sysconfig/network

각 Node에 맞게 네트워크에 관련된 파일들은 편집한다. 예제는 첨부1에 나온다.

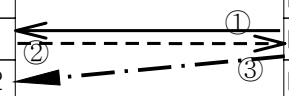
5.9 문제점들

위와 같이 Channel Bonding과 Diskless를 하면 문제가 발생한다.

만약, 같은 Lan Card를 3개를 사용하면, Fig 7의 Slave Node의 PCI 4에 있는 Lan Card를 기본으로 하여 etherboot가 Master Node를 찾기 때문에 부팅하지 못한다. 따라서 PCI 3에 다른 종류(여기서는 Intel)의 Lan Card로 변경하여, etherboot에서 Floppy Disk로 이미지 만들 때 해당되는 Lan Card의 이미지를 넣으면, 원하는 Lan Card를 이용하여 부팅을 할 수 있다.

SLOT	Device	Lan	
AGP	VGA		
PCI 1	SCSI		
PCI 2	3Com Lan	eth0	
PCI 3	Intel Lan	eth1	
PCI 4	Realtek Lan	Channel Bonding	eth2
PCI 5	Realtek Lan		eth3

Master Node의 SLOT구성



SLOT	Device	Lan
AGP	VGA	
PCI 1		
PCI 2		
PCI 3	Intel Lan	eth0
PCI 4	Realtek Lan	eth1
PCI 5	Realtek Lan	eth2

Slave Node의 SLOT구성

Fig 7 3개의 Lan을 구성하여 나타나는 문제점

① : Slave Node가 부팅되어 DHCP서버를 찾아간다.

② : Master Node가 DHCP정보를 준다.(기본적으로 eth2의 정보를 준다.)

③ : Slave Node가 Master Node의 eth2에게서 kernel을 받아오려 한다.

그리고 Channel Bonding을 할 때 설명한 Lan card의 위치를 주의해야 한다. Fig 7에서 보는 것과 같이 mknbi-linux에서 사용하는 디바이스는 eth0(①)이다. 하지만, Master Node의 dhcpd가 Slave Node에게 넘겨주는 Master Node의 IP는 Channel Bonding된 Device의 IP(②)이다 따라서 Slave Node에서 Master Node에 있는 Kernel을 가져올 수 없다(③). 원인은 정확히는 알 수 없었지만, dhcpd를 실행할 때 네트워크 디바이스 중 가장 먼저 나오는 ip를 사용하는 것 같았다. 따라서 이것을 해결하기 위해서 /etc/rc.d/init.d/route.sh라는 shell script를 만들었다. 작동 원리는 Channel Bonding을 해제하고, eth2와 eth3을 중지시킨 후 dhcpd를 실행한다. 그런 후 네트워크을 Channel Bonding을 한 후 다시 실행한다.

그리고 각 디바이스별로 보내는 데이터의 종류를 지정하기 위해서는 route명령을 사용하여 지정하였다.

example) /etc/rc.d/init.d/route.sh

```
/etc/sysconfig/network-scripts/setOrgNetwork.sh      #ChannelBonding 해제 script
/etc/sysconfig/network-scripts/ifdown eth2           #eth2 정지
/etc/sysconfig/network-scripts/ifdown eth3           #eth3 정지
/etc/rc.d/init.d/dhcpd restart                       #dhcpd 재 시작
/etc/sysconfig/network-scripts/setBonding.sh         #ChannelBonding로 설정
route add 192.168.0.21 eth1   #192.168.0.21로 가는 데이터는 eth1로가게함
route add 192.168.0.22 bond0  #192.168.0.22로 가는 데이터는 bond0로가게함
route add 192.168.0.31 eth1   #192.168.0.31로 가는 데이터는 eth1로가게함
route add 192.168.0.32 bond0  #192.168.0.32로 가는 데이터는 bond0로가게함
route add 192.168.0.41 eth1   #192.168.0.41로 가는 데이터는 eth1로가게함
route add 192.168.0.42 bond0  #192.168.0.42로 가는 데이터는 bond0로가게함
```

6. software RAID

6.1 RAID(Level 0) 란?

하나의 데이터를 기록할 때 여러 하드 디스크에 분산저장을 함으로써 빠른 입출력을 가능하게 한다.

6.2 설정방법

6.2.1 Kernel 컴파일

Kernel 컴파일 시 RAID와 관련된 것을 선택해 준다.

```
CONFIG_BLK_MD=y             #Raid 구성을 위한 멀티플 디바이스 지원
CONFIG_AUTODETECT_RAID=y    #Raid 자동 인식
CONFIG_MD_LINEAR=y          #Raid 선형적인 블럭 디바이스 접근 모드
CONFIG_MD_STRIPED=y         #Raid 0
CONFIG_MD_MIRRORING=y       #Raid 1
CONFIG_MD_RAID5=y           #Raid 5
```

커널 컴파일 후 리부팅 해야 한다.

6.2.2 프로그램 설치

raidtools RPM package를 받아 설치한다.

6.2.3 raidtab파일을 작성

/usr/share/doc/raidtools-0.90/에 각 Level별로 예제 파일이 있다.

example) /etc/raidtab

```
raiddev                 /dev/md0
raid-level              0    # it's not obvious but this *must* be
                             # right after raiddev
persistent-superblock   1    # set this to 1 if you want autostart,
                             # BUT SETTING TO 1 WILL DESTROY PREVIOUS
                             # CONTENTS if this is a RAID0 array created
                             # by older raidtools (0.40-0.51) or mdtools!
chunk-size              4
nr-raid-disks           2
nr-spare-disks          0
device                  /dev/hde1
raid-disk               0
device                  /dev/hdg1
raid-disk               1
```

6.2.4 fdisk

fdisk를 이용하여 설정할 하드디스크를 파이션을 raid로 생성한다.

example) fdisk /dev/hde

*Command (m for help): **n***

*Partition number (1-8): **1***

*First cylinder (0-1965): **0***

*Last cylinder or +size or +sizeM or +sizeK (0-1965, default 1965): **1965***

*Command (m for help): **p***

Device	Flag	Start	End	Blocks	Id	System
/dev/hde1		0	1965	1336200	83	Linux native

*Command (m for help): **t***

*Partition number (1-8): **1***

*Hex code (type L to list codes): **fd***

*Command (m for help): **w***

각 값을 하드디스크의 용량에 맞게 입력을 한다.

2개(또는 그 이상)의 하드디스크의 용량은 같이 않아도 되며, 종류 또한 달라도 된다.

6.2.5 Raid Disk 만들기

Raid 파티션으로 설정된 하드 디스크에 File System을 만든다.

mkraid /dev/md0

mkfs /dev/mo0

Raid생성이 완료되었다.

mount /dev/md0 /MODEL

6.3 성능 TEST

간단하게 성능을 테스트 하기 위해서 약 2Gbyte용량의 파일을 복사해 보았다. (Fig 8)

Source Disk	Target Disk	Time
SCSI HDD	IDE RAID	1분34.510초
SCSI HDD	IDE HDD	1분19.301초
IDE RAID	IDE RAID	1분34.034초
IDE HDD	IDE HDD	2분55.269초
SCSI HDD	SCSI HDD	3분19.867초

Fig 8 HDD 성능 테스트

7. Benchmark

Channel Bonding이나 Diskless를 하기 전의 결과는 MM5에서 제공하는 TEST데이터에 관련된 결과만 있다. 하지만 이 Test데이터로 나온 결과는 실제 한반도 범위(Fig 9)를 MM5에 입력할 자료와는 계산시간에 많은 차이가 있고, 그 결과 역시 상당한 차이를 보인다. 한반도(Fig 9) 데이터를 입력하면 Channel Bonding을 전▪후의 계산시간에는 별 차이가 없었지만, Test데이터를 사용하여 계산을 하면 시간이 20%정도 더 걸렸다. 이 문제는 데이터범위차이에 의해서 계산양이나 계산 방식에 많은 차이가 있는 것으로 보여 진다.

계산 범위	위도	경도	참고
30Km	25N~45N	106E~143E	Fig 11
10Km	32N~44N	118E~133E	Fig 12

Fig 9 입력되는 데이터 범위

	소요 시간
Master Node만 실행	약 18시간 10분
2Node에서 실행	약 9시간 15분
4Node 모두에서 실생	약 5시간 12분

Fig 10 노드 개수별 실행 시간

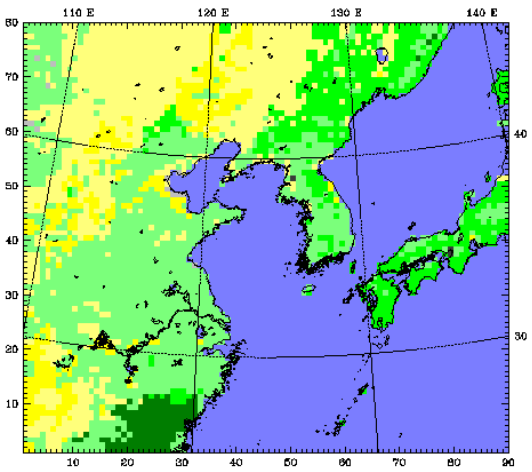


Fig 11 격자거리 30Km

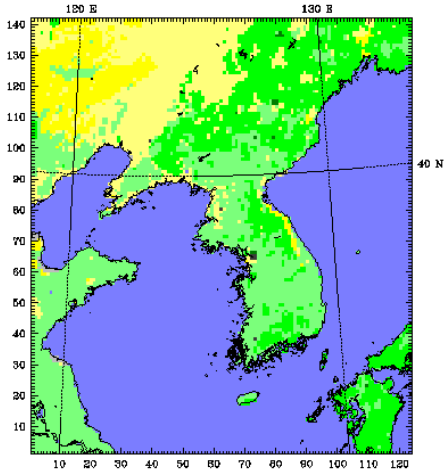


Fig 12 격자거리 10Km

8. 결론

하나의 Node에서만 MM5를 실행했을 때 약18시간 10분이 걸렸으나, Cluster를 구성한 후에는 5시간 12분으로 350% 계산 속도가 향상되었다.

Channel Bonding을 하기전과 후의 정확한 소요시간과 Diskless를 하기전과 후의 정확한 소요시간은 그 당시 시스템 불안정으로 측정하지 못했지만, Channel Bonding으로 인한 성능 향상은 거의 없었고, 또한 Diskless로 인한 성능 저하 또한 크게 나타나진 않았다. 4Node로는 아직 Network 대역폭을 100Mbps를 넘지 못하는 것으로 생각되어 진다. 하지만 차후 Node가 늘어나면 Channel Bonding으로 인한 성능 향상이 있을 것으로 예상된다.

한 가지 아쉬운 것은 시간 부족으로 좀 더 많은 벤치마크를 하지 못한 점과 모니터링 프로그램으로 네트워크 사용량 등을 확인하지 못한 것이 아쉽다.

참고 자료

http://linux.sarang.net

http://kldp.org

Linux@Work 2001년9월호 P190-202

http://www.redhat.com

http://bbs.hancom.com/databbs/pds15_files/gcc-2.95.3.tar.gz

http://www.pgroup.com

http://www-unix.mcs.anl.gov/mpi/mpich/

http://www.mmm.ucar.edu/mm5/mm5-home.html

http://etherboot.sourceforge.net/doc/html/userman.html

http://www.metri.re.kr/~tkjang

첨부 1

각종 설정 파일들

1. Master Node

◎/etc/hosts

```
127.0.0.1       hydra.knu.ac.kr hydra localhost.localdomain localhost
155.230.156.227 hydra.knu.ac.kr hydra
192.168.0.11    node1
192.168.0.12    node1b
192.168.0.21    node2
192.168.0.22    node2b
192.168.0.31    node3
192.168.0.32    node3b
192.168.0.41    node4
192.168.0.42    node4b
```

◎/etc/exports

```
/tftpboot/node2         node2(rw,no_root_squash,no_all_squash)
/tftpboot/node3         node3(rw,no_root_squash,no_all_squash)
/tftpboot/node4         node4(rw,no_root_squash,no_all_squash)
/usr                    node*(rw,no_root_squash,no_all_squash)
/usr/local              node*(rw,no_root_squash,no_all_squash)
/home                   node*(rw,no_root_squash,no_all_squash)
```

◎/etc/hosts.equiv

```
hydra
node1
node1b
node2
node2b
node3
node3b
node4
node4b
```

◎/etc/dhcpd.conf

```
option broadcast-address 192.168.0.255;
```

```
option domain-name-servers 192.168.0.11;
option routers 192.168.0.11;
subnet 192.168.0.0 netmask 255.255.255.0 {
        host node2 {
                hardware ethernet 00:03:47:7a:eb:f8;
                fixed-address 192.168.0.21;
                option root-path "/tftpboot/node2";
                filename "vmlinuz_node2.nb";
        }
        host node3 {
                hardware ethernet 00:03:47:7a:ec:57;
                fixed-address 192.168.0.31;
                option root-path "/tftpboot/node3";
                filename "vmlinuz_node3.nb";
        }
        host node4 {
                hardware ethernet 00:03:47:7a:ea:fc;
                fixed-address 192.168.0.41;
                option root-path "/tftpboot/node4";
                filename "vmlinuz_node4.nb";
        }
}
subnet 155.230.156.0 netmask 255.255.252.0 {
}
```

◎/etc/sysconfig/network

```
NETWORKING=yes
#HOSTNAME=hydra.knu.ac.kr
HOSTNAME=node1b
GATEWAY=bond0
GATEWAY=155.230.156.5
```

◎/etc/sysconfig/network-scripts/ifcfg-bond0

```
DEVICE=bond0
USERCTL=no
```

ONBOOT=yes

BOOTPROTO=none

BROADCAST=192.168.0.255

NETWORK=192.168.0.0

NETMASK=255.255.255.0

IPADDR=192.168.0.12

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

DEVICE=eth0

BOOTPROTO=static

BROADCAST=155.230.156.255

IPADDR=155.230.156.227

NETMASK=255.255.252.0

NETWORK=155.230.156.0

ONBOOT=yes

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

DEVICE=eth1

BROADCAST=192.168.0.255

IPADDR=192.168.0.11

NETMASK=255.255.255.0

NETWORK=192.168.0.0

ONBOOT=yes

BOOTPROTO=static

◎/etc/sysconfig/network-scripts/ifcfg-eth2

DEVICE=eth2

USERCTL=no

ONBOOT=yes

MASTER=bond0

SLAVE=yes

BOOTPROTO=none

◎/etc/sysconfig/network-scripts/ifcfg-eth3

DEVICE=eth3

USERCTL=no

ONBOOT=yes

```
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

◎/etc/sysconfig/network-scripts/setBonding.sh

```
/etc/rc.d/init.d/network stop
cd /etc/sysconfig
rm -f network
ln -s network_bonding network
cd /etc/sysconfig/network-scripts
rm -f ifcfg-eth2
rm -f ifcfg-eth3
ln -s ifcfg-bond0_bonding ifcfg-bond0
ln -s ifcfg-eth2_bonding ifcfg-eth2
ln -s ifcfg-eth3_bonding ifcfg-eth3
/etc/rc.d/init.d/network start
echo 'if you run manually "setBonding.sh", you must run "/etc/rc.d/init.d/route.sh"'
```

◎/etc/sysconfig/network-scripts/setOrgNetwork.sh

```
/etc/rc.d/init.d/network stop
cd /etc/sysconfig
rm -f network
ln -s network_org network
cd /etc/sysconfig/network-scripts
rm -f ifcfg-bond0
rm -f ifcfg-eth2
rm -f ifcfg-eth3
ln -s ifcfg-eth2_org ifcfg-eth2
ln -s ifcfg-eth3_org ifcfg-eth3
/etc/rc.d/init.d/network start
```

◎/etc/rc.d/init.d/route.sh

```
/etc/sysconfig/network-scripts/setOrgNetwork.sh
/etc/sysconfig/network-scripts/ifdown eth2
/etc/sysconfig/network-scripts/ifdown eth3
/etc/rc.d/init.d/dhcpd restart
```

/etc/sysconfig/network-scripts/setBonding.sh

route add node2 eth1

route add node2b bond0

route add node3 eth1

route add node3b bond0

route add node4 eth1

route add node4b bond0

2. Slave Node (node2)

◎/tftpboot/node2/etc/hosts

Master Node와 같음

◎/tftpboot/node2/etc/hosts.equiv

Master Node와 같음

◎/tftpboot/node2/etc/fstab

192.168.0.11:/tftpboot/node2	/	nfs	defaults	0 0
192.168.0.11:/usr	/usr	nfs	defaults	0 0
192.168.0.11:/usr/local	/usr/local	nfs	defaults	0 0
192.168.0.11:/home	/home	nfs	defaults	0 0
none	/proc	proc	defaults	0 0

◎/tftpboot/node2/etc/sysconfig/network

NETWORKING=yes

HOSTNAME=node2b

GATEWAY=192.168.0.11

GATEWAY=bond0

◎/tftpboot/node2/etc/sysconfig/network-scripts/ifcfg-bond0

DEVICE=bond0

USERCTL=no

ONBOOT=yes

BOOTPROTO=none

BROADCAST=192.168.0.255

NETWORK=192.168.0.0

NETMASK=255.255.255.0

IPADDR=192.168.0.22

◎/tftpboot/node2/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.0.255
IPADDR=192.168.0.21
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
```

◎/tftpboot/node2/etc/sysconfig/network-scripts/ifcfg-eth1

```
DEVICE=eth1
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=no
```

◎/tftpboot/node2/etc/sysconfig/network-scripts/ifcfg-eth2

```
DEVICE=eth2
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=no
```

◎/tftpboot/node2/etc/rc.2/init.d/route.sh

```
route add node1 eth0
route add node1b bond0
route add node3 eth0
route add node3b bond0
route add node4 eth0
route add node4b bond0
```

첨부 2 (2001년 12월 4 Node 추가)

	Master Node	Slave Node
CPU	AMD Athlon MP 1600+ (1.4Ghz) X 2	AMD Athlon XP 1600+ (1.4Ghz)
M/B	TYAN S2460 (Thunder K7)	SUMA SUPERIOR 8KHA+
Memory	1GB(PC2100 DDR RAM)	512MB(PC2100 DDR RAM)
HDD	19GB SCSI 80GB IDE X 1	
LAN	3Com 10/100Mbps X 1 Intel 10/100Mbps X 1 Realtek 10/100Mbps X 2	Intel 10/100Mbps X 1 Realtek 10/100 X 2
HUB	NETGEAR 10/100Mbps Switching 8Port X 1 (변경)	

Fig 1 추가된 Node들의 System 사양

Master Node는 Fig 1에 보듯이 Dual Athlon MP로 변경하였고, 추가되는 3 Node는 Athlon XP로 하였다.

2001년 12월 10일 현재 100%설치가 끝난 것은 아니지만, Slave Node들은 모두 테스트 중이다.

추가 사항 & 문제점들

1. HUB

기존에 사용하던 ReeNet HUB를 NFS용으로 사용하면, 추가된 Slave Node들은 부팅 되지 않았다(DHCP를 이용하여 IP는 가져오지만, Kernel을 가져오지 못한다).

따라서 NFS용 HUB를 NETGEAR 10/100Mbps Switching HUB를 변경하여 해결하였다. 무엇이 문제인지는 알 수 없었으나, MainBoard와 HUB간의 충돌이라고 밖엔 설명 할 수 없었다(CPU, RAM, Mainboard를 제외한 랜카드 등 다른 장비는 모두 기존 Node와 동일하다). 그리고 현재 ReeNet HUB를 계산용으로만 사용하지만 이 역시 컴퓨터에 전원을 넣지 않았는데도, 몇몇 Node에서만 HUB의 LINK램프가 들어오는 등 부품들의 조화가 완벽하다는 느낌을 받지는 못했다(어떻게 보면 MainBoard문제일지도... 처음 구입한 Asus MainBoard를 사용하는 Node에서는 문제가 없었다). 다음에 Node를 추가하기 위하여 HUB를 구입하거나 또는 다른 Cluster를 구성할 때는 성능에는 이상이 없다고 하여도 어느 정도 괜찮은 HUB를 사용해야 할 것으로 생각된다(실제 계산용으로 사용하기에는 전혀 이상이 없다).

2. Node 추가 시 잊어 버렸던 부분

Node를 추가할 때 hosts, hosts.equiv, exports등 RedHat의 기본적인 설정파일뿐만 아니라, Master Node와 각 Node의 /etc/rc.d/init.d/route.sh도 변경해 주어야 한다.

example) 변경된 Master Node의 /etc/rc.d/init.d/route.sh

```
route add node2 eth0
route add node2b bond0
route add node3 eth0
route add node3b bond0
route add node4 eth0
route add node4b bond0
route add node5 eth0
route add node5b bond0
route add node6 eth0
route add node6b bond0
route add node7 eth0
route add node7b bond0
route add node8 eth0
route add node8b bond0
```

3. Dual Athlon MP 1600+ 에서 Test한 결과

계산은 종료하는데 7시간 12분 걸렸다.

2 Node에서는 9시간 15분이 소요 된 것에 비해 2시간 3분 빠른 결과이다.

평균 CPU 점유율이 약 96%에 이르는 등 2대의 컴퓨터를 연결하는 것 보다 빠른 결과는 보여주었다. 물론 CPU와 RAM의 성능이 향상된 점도 있지만, 생각했던 것보다 많이 향상된 결과를 보여주었다. 하지만, 1CPU Node를 8대 연결하여 Cluster를 구성하는 것과 2CPU Node를 4대 연경하여 Cluster를 구성하는 것은 어느 쪽이 빠르다고 추정할 수 없었다. 실제 테스트가 있어야만 알 수 있을 것으로 생각되어 진다.

4. HomePage

http://hanl.knu.ac.kr/~mm5

매일 04시, 16시 마다 실행한 MM5의 결과를 위 홈페이지에 실시간으로 보여준다.

(Design & Program : 원덕진, 장태규)

5. 8Node Benchmark 결과

(Dual Athlon System을 사용하지 않고, 기존 Master Node사용)

8 Node를 모두 사용하여 MM5 모델을 실행하면 3시간 15분이 걸렸다.

정확한 시간은 제시 할 수 없으나, 7 Node를 사용하여 MM5 모델은 동작시킨 결과와 약 1시간 15분정도 차이가 난다. 그리고 평균 CPU 점유율 또한 약 5~10%정도 더 상승한 것으로 생각된다. 2, 4, 8, 16, ... 2^n으로 실행하였을 때 성능이 더욱 향상되는 것으로 생각 된다.

마지막으로 아주 단순히 생각 했을 때(전처리 과정과 후처리 과정을 8 Node일 경우만 포함 했을 때) 1 Node에서만 계산 했을 때 18시간 10분이 소요되었고, 이 시간은 단순히 8로 나누면 137분이 된다. 이 시간을 실제 8 Node로 계산하였을 때의 시간(3시간 15분)과 비교하면 각Node당 평균 가동률은 약 70%가 된다.