

공대전산실 고속계산용 클러스터 시스템 이용안내서

담당자 : 이창성 (7429, hpcman@eng.snu.ac.kr)

1. 시스템(hpceng)개요

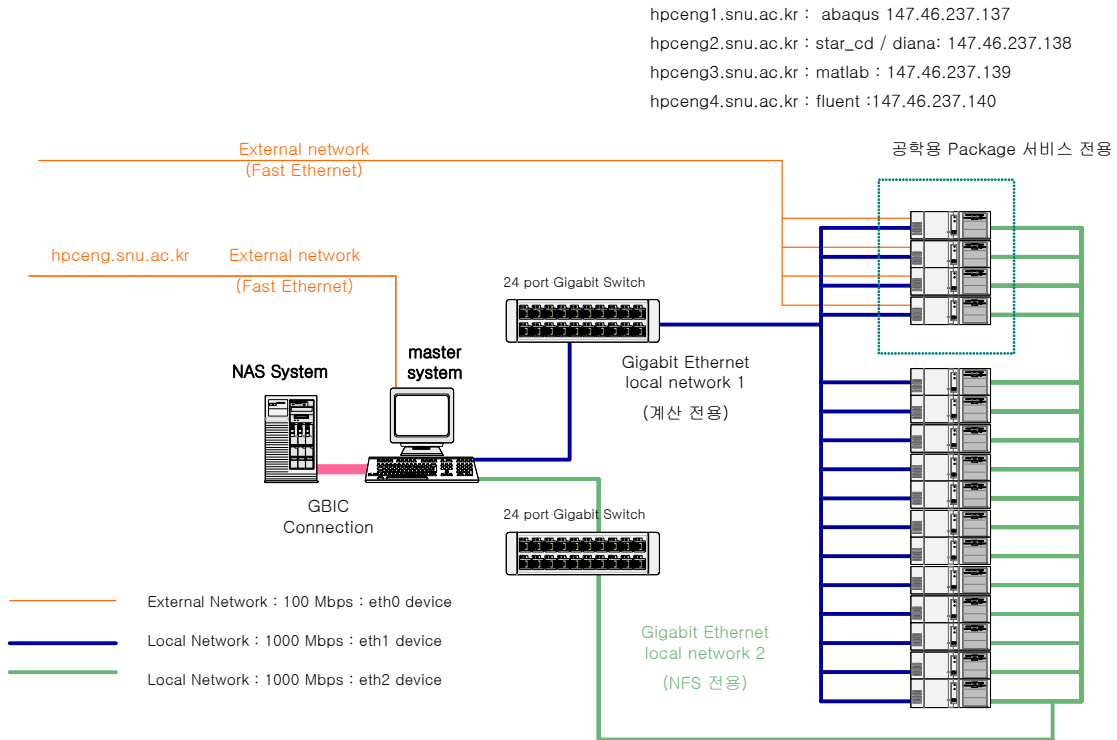


그림 1 hpceng 시스템 네트워크 구성도

- 병렬계산용 서버 Pool과 공학계산용 Package 서비스 Pool로 나누어서 구성
- 주요사양
 - 16 Node Cluster System + master node + NAS System
 - CPU : Intel Xeon 2.4 GHz 32개
 - RAM : 2GB (12 node/병렬계산용), 3GB (4 node/Package 서비스 서버)
 - SCSI : 36GB (12 node/병렬계산용), 54GB (4 node/Package 서비스 서버)
 - Storage Backup : 1.2TB (운영 : RAID 1 Mirroring, 600 GB)
 - Network : Gigabit Ethernet Network (Computing,NFS,Management),
Fast Ethernet Network (External)
 - OS : Linux (RedHat 9.0 기반, Kernel : 2.4.24)
- 주요 소프트웨어
 - Compiler : intel compiler(c/c++/f77/f90), gcc compiler(c/c++/f77),
Portland Group Compiler(c/c++/f77/f90)
 - Math Library : BLAS, LAPACK, IMSL (f90)
 - Scheduler : LSF(Platform)

MPI : MPICH, LAM-MPI

□ 공학용 Package

- Abaqus 6.4 : Standard / Explicit / Aqa / Design

- Fluent

fluent 6.0.12

TGrid 3.4.2

Gambit 2.0.4

tfilter 2.5, flpost 1.2.6

- LS-DYNA (Parallel Version - mpp970) : 8 CPUs

lspost

- Matlab

Simulink

Optimization

Statistics

Neural Network

Extended Symbolic Math

PDE

Mapping

Spline

Curve fitting

Signal Processing

System Identification

Wavelet

Filter design

Control system

Fuzzy logic

Robust Control

Mu-analysis and LMI Control

Stateflow

Nonlinear control design blockset

-Diana

2. 시스템 사용신청

공대전산실에서 계정신청서를 작성한다. 유의사항은 기존의 공대전산실 사용 아이디가 있다고 해도 새롭게 신청을 해서 발급을 받아야 사용가능.

3. 병렬시스템 이용방법

① 로그인 하기

기본적으로 hpceng 시스템은 SSH Secure Shell 로만 접속이 가능하다. ssh를 이용한 접속방법을 유닉스 / 리눅스 시스템과 윈도우즈 시스템으로 나누어서 살펴보면 다음과 같다.

- 유닉스/리눅스 시스템

현재, 많이 사용되고 있는 리눅스 시스템의 경우에는 대부분의 배포판에서 ssh client 프로그램이 설치되어 있다. 다음의 명령어로 시스템에 설치된 ssh 프로그램을 확인할 수 있다.

```
$ rpm qa |grep ssh
```

위와 같이 명령어를 실행시켰을 경우에 3.0 이상 버전의 ssh가 설치되어 있는 것이 바람직하다. 만약, 시스템에 ssh가 설치되어 있지 않는 경우에는 www.ssh.com 사이트에서 rpm 버전이나 소스를 다운받아 설치해야 한다. Unix 시스템인 경우에는 시스템 관리자에게 문의해서 ssh가 설치되어 확인해야 한다.

시스템에 ssh가 설치되어 있는지 확인한 후에는 시스템에서 xterm이나 hanterm을 실행시키고 다음의 명령어를 수행한다.

```
$ xhost +
```

그리고 나서 hpceng 시스템에 접속한다.

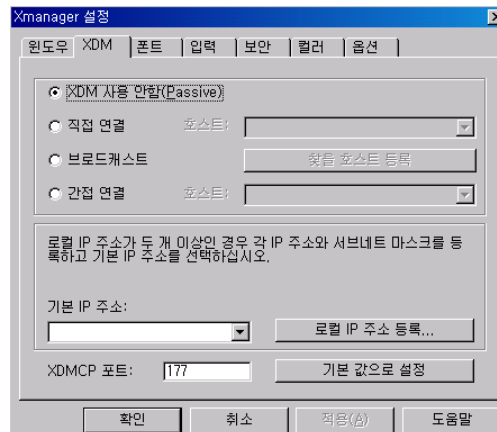
```
$ ssh -l userid hpceng.snu.ac.kr
```

패스워드 입력후에 성공적으로 로그인 되고 난 후에는 아래의 명령을 실행시킨다. hpceng 시스템은 기본적으로 Born Shell을 사용하므로, 다음과 같이 입력해서 DISPLAY 변수를 설정해 주면 GUI 기반 응용프로그램을 사용할 수 있다.

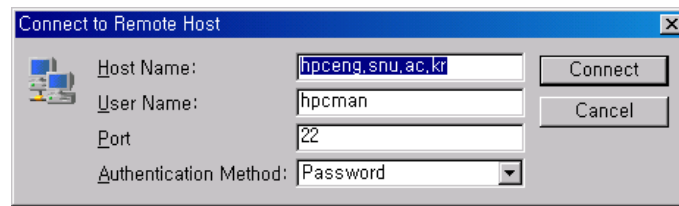
```
$ export DISPLAY=ip-address:0.0
```

- 윈도우즈 PC 시스템

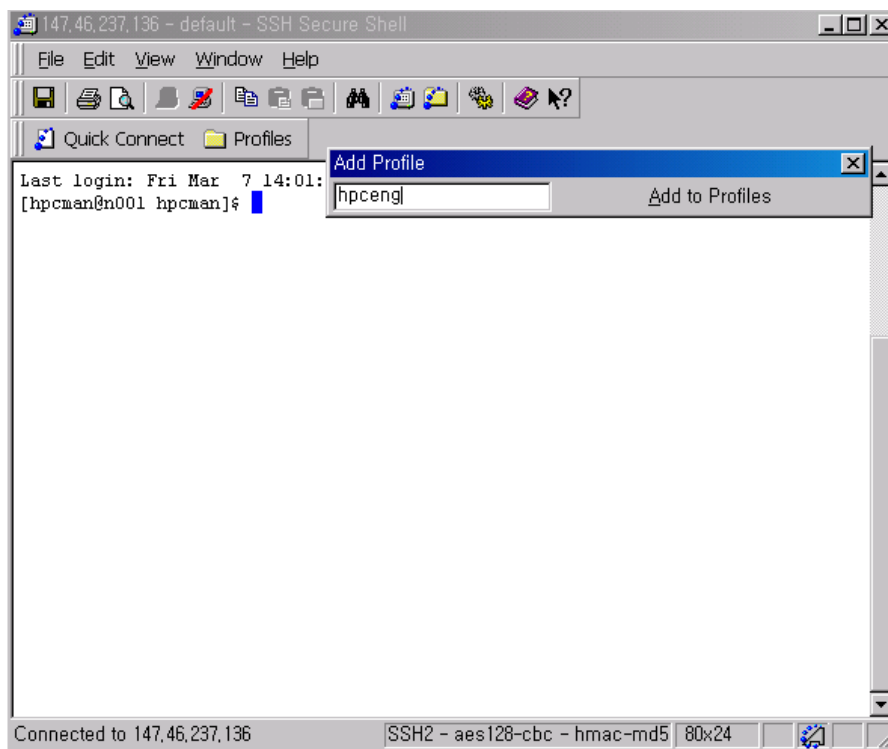
Windows인 경우에는 먼저, it4u.snu.ac.kr에 가서 SSHWinClient-31과 Xmanager 프로그램을 설치한다. 이 때 Xmanager 설치시에 “파일” 메뉴의 “설정”을 눌러서 xdm을 사용하지 않는 것이 좋다.



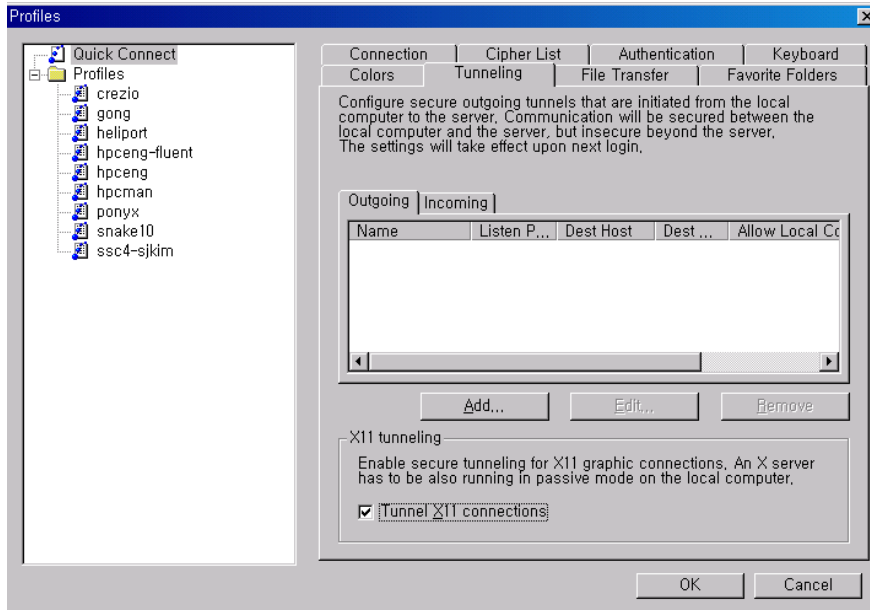
SSHWinClient-31을 설치한 후에, 실행시키고 나서 “Quick Connect”를 click 한 후에 다음과 같이 User name에 등록된 User ID를 입력한다.



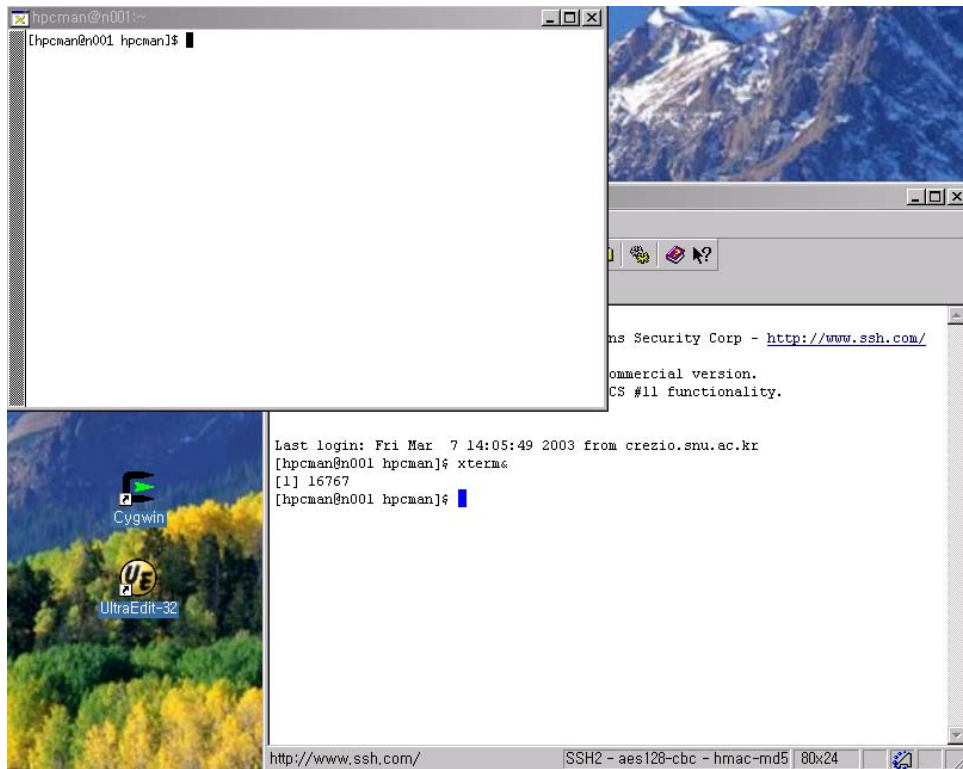
접속이 되면 패스워드를 입력한다. 그런 다음, 정상적인 로그인이 되면 “profiles” 를 click 해서 “Add profile”를 click 한 후에 아래 그림과 같이 profile 이름을 hpceng로 한 후에 “Add the current connections to profile”를 click 해서 현재 설정을 등록한다.



그런 다음, 일단 logout을 한다. 그런 다음, “edit profile” 명령을 실행시킨 후 아래 그림과 Tunneling 설정에서 Tunnel X11 Connections을 On 한 후에 save 시킨다.



위의 설정을 저장한 후에, profile에 저장되어 있는 hpceng를 click하면 hpceng 시스템에 저장된 설정으로 접속이 된다. 일단 이 상태에서는 vi 에디터를 이용한 파일 편집 등의 graphic interface를 요구하지 않는 Text 기반의 작업들은 모두 수행할 수 있다. 만약, GUI 기반 작업을 수행해야 할 경우에는 Xterm을 다음과 같이 실행시킨 후에 사용하면 된다. 먼저, 접속한 PC에서 Xmanager를 실행시킨다. 다음과 같이 xterm을 실행시키면 된다.

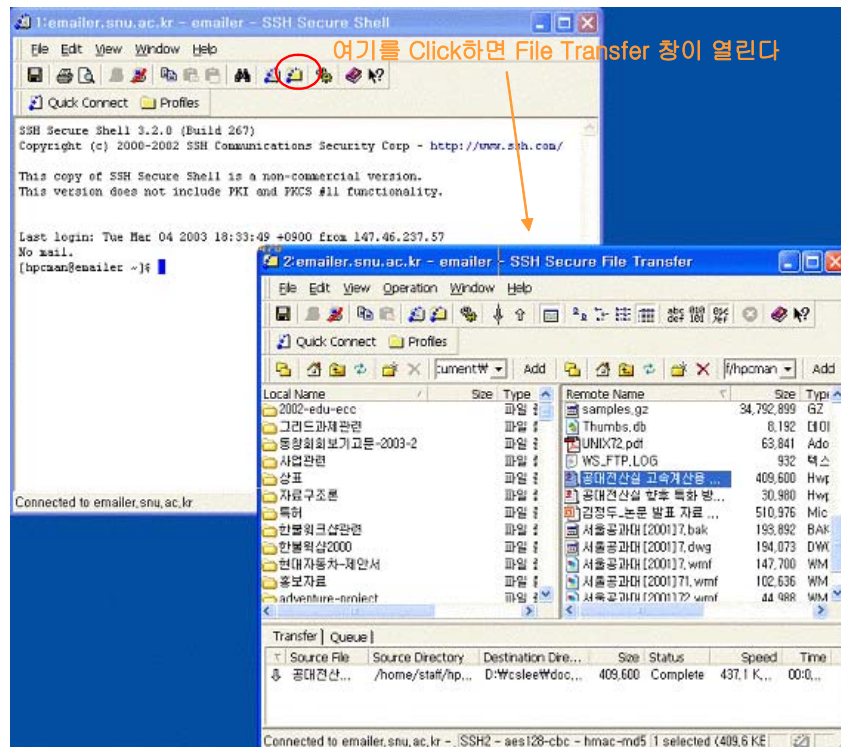


계산한 결과나 입력데이터를 전송해야 할 때에는 sftp를 이용할 수 있다. 유닉스나 리눅스 시스템의 경우에는 다음과 같은 명령어를 통해 사용한다.

```
$ sftp user_id@hpceng.snu.ac.kr
```

파일을 hpceng 시스템으로 Upload 할 때에는 put 이라는 명령어를 사용하고, download 할 때에는 get 이라는 명령어를 사용하면 된다.

윈도우즈 시스템의 경우에는 SSHWinClient에서 제공하는 File Transfer Window를 이용하면 윈도우즈의 탐색기와 같은 편리한 유저 인터페이스를 이용해서 자유롭게 파일을 전송할 수 있다.



② 디스크 사용 관련

hpceng 시스템에 로그인 한 후에 `df -h` 라는 명령을 수행하면 아래와 같이 현재 사용할 수 있는 하드디스크 용량과 파일시스템에 대해 알 수 있다.

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/sda2	8254272	2828848	5006128	37%	/
/dev/sda1	124427	14996	103007	13%	/boot
/dev/sda5	25861692	1984	24545996	1%	/scratch
none	1034716	0	1034716	0%	/dev/shm
/dev/sdb1	141122196	29471024	104482572	23%	/system-bak
n000s:/home	345354160	15661976	312149208	5%	/home
n000s:/data	460470760	8	437080120	1%	/data

위의 파일 시스템들을 살펴보면 `/system-bak` 파티션은 시스템 이미지 백업용이라 관리자만 접근 가능한 파티션이다. 유저들의 홈디렉토리는 `/home` 파티션이다. `/home` 파티션은 NFS(Network File System)으로 연결되어 있어서 모든 클러스터 노드들에서 동일한 이미지를 유저에게 제공한다. 그러나 `/home` 파티션은 유저들에게 1GB 이상의 저장공간을 허락하지 않는다. 따라서 대용량 계산 결과 파일등의 저장은 `/data` 파티션을 이용하여야 한다. `/data` 파티션 역시 NFS로 연결되어 있다. `/data` 파티션의 경우에 유저별 사용량의 제한이 없기 때문에 사용가능한 용량까지 자유롭게 사용가능하지만, 매일 새벽 2시에 용량이 부족할 경우 관리자가 모든 데이터를 삭제할 수 있기 때문에 대용량 결과 데이터의 임시적인 저장장소로 활용하고 중요한 데이터는 유저의 개인 PC로 항상 옮겨 놓아야 한다. 이 때 조심해야 할 것은 계산 중에 임시로 열어서 사용하거나 대용량의 파일에 결과를 적어야 할 경우에는 반드시 `/scratch` 디렉토리 밑에서 읽고 써야 한다. 유저의 홈디렉토리는 NFS(Network File System) 이므로 자신의 홈디렉토리에 그냥 파일을 열고 쓰게 되면 굉장히 계산이 느려질 수 있다. `/scratch` 파티션은 모든 클러스터 노드에 로컬의 별도 하드디스크로 연결되어 있기 때문에 이미지는 서로 다르지만 입출력 속도는 `/home`이나 `/data` 파티션에 비해서 빠르다.

③ 병렬 프로그램 수행하기

기본적으로 hpceng 시스템은 최근에 거의 사용되지 않는 PVM 라이브러리는 지원하지 않고, MPI 라이브러리만 지원한다. MPI 라이브러리 중에서는 앞에서 설치한 것처럼 MPICH와 LAM-MPI를 설치하였다. 시스템에서 기본(Default)로 설정된 값은 MPICH와 Intel Compiler를 이용하여 LSF Scheduler를 이용하여 병렬처리 job을 Submit 하게 되어 있다.

- 병렬코드 컴파일하기

mpicc, mpiCC, mpif77, mpif90 명령어를 이용하여 컴파일한다. 이 명령어는 기본적으로 intel compiler에 MPICH 라이브러리를 자동링크되게 한 것이므로 컴파일 기본 옵션은 intel 컴파일러와 동일하다. intel 컴파일러에 대한 자세한 자료는 support.intel.com에서 참조할 수 있다.

그런데, 자신이 개발한 코드가 intel 컴파일러에서 간혹 에러를 발생시키는 경우가 있다. 유닉스 시스템의 표준 컴파일러인 GNU gcc 컴파일러로 컴파일할 때는 에러가 발생하지 않는 코드인데 intel 컴파일러로 컴파일할 시에는 에러를 발생시킬 수 있다. 특히, Fortran 의 경우가 그런 경우가 많이 발생한다. 이럴 경우에는 GNU gcc 컴파일러로 컴파일할 수 있고 명령어는 gmpicc, gmpiCC, gmpif77을 이용하면 된다.

병렬 전용 컴파일러로 많이 사용되는 또 하나의 컴파일러가 Portland Group Compiler이며 hpceng 시스템에 설치되어 있다. 명령어는 pmpicc, pmpiCC, pmpif77, pmpif90을 사용하면 된다. pgc 컴파일러에 대한 자세한 자료는 www.pgroup.com에서 참조할 수 있다.

-병렬코드 실행하기

병렬코드는 LSF Scheduler를 이용하여 submit 하여 실행한다. 우선, 시스템의 상황을 살펴보는 명령은 다음과 같다.

```
[hpcman@n001 hpcman]$ bhosts
```

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
n001	ok	-	2	0	0	0	0	0
n002	ok	-	2	0	0	0	0	0
n003	ok	-	2	0	0	0	0	0
n004	ok	-	2	0	0	0	0	0
n005	ok	-	2	0	0	0	0	0
n006	ok	-	2	0	0	0	0	0
n007	ok	-	2	0	0	0	0	0
n008	ok	-	2	0	0	0	0	0
n009	ok	-	2	0	0	0	0	0
n010	ok	-	2	0	0	0	0	0
n011	ok	-	2	0	0	0	0	0
n012	ok	-	2	0	0	0	0	0
n013	unavail	-	2	0	0	0	0	0
n014	unavail	-	2	0	0	0	0	0
n015	unavail	-	2	0	0	0	0	0
n016	unavail	-	2	0	0	0	0	0

위의 상황은 현재 n001 ~ n012 시스템까지 이용가능하며, MAX 라는 것은 각 노드별 수행할 수 있는 작업의 최대 수이다. 각 노드별로 2개의 CPU가 설치되어 있으므로 모든 노드의 MAX 값은 2로 설정되어 있으며, RUN 이라는 것은 현재 수행되고 있는 작업의 수이다. NJOBS는 현재 수행하여야 하는 작업의 수이다. n013부터 n016까지는 Package 서비스 노드이므로 병렬작업 수

행에는 이용할 수 없다. 특별히 32개의 CPU 작업을 수행해야 하는 경우에는 담당조교에게 연락해서 상의해야 한다. 그리고 각 노드별 Resource의 현황을 살펴 보기 위해서는 lload라는 명령을 수행한다. 각 노드별 CPU 가동율과 이용가능한 RAM 메모리 현황 등을 볼 수 있다.

```
[hpcman@n001 hpcman]$ lload
```

HOST_NAME	status	r15s	r1m	r15m	ut	pg	ls	it	tmp	swp	mem
n001	ok	0.0	0.0	0.0	0%	1.1	2	3	5748M	1028M	1771M
n006	ok	0.0	0.0	0.0	0%	0.3	0	213	6260M	1027M	1859M
n003	ok	0.0	0.0	0.0	0%	0.3	1	52	6092M	1027M	1858M
n002	ok	0.0	0.0	0.0	0%	0.3	0	220	5996M	1027M	1860M
n007	ok	0.0	0.0	0.0	0%	0.3	0	214	6260M	1027M	1859M
n004	ok	0.0	0.0	0.0	0%	0.3	0	53	6260M	1027M	1859M
n010	ok	0.0	0.0	0.0	0%	0.3	0	214	6260M	1027M	1859M
n011	ok	0.0	0.0	0.0	0%	0.3	0	214	6096M	1027M	1859M
n008	ok	0.0	0.0	0.0	0%	0.3	0	214	6260M	1027M	1859M
n005	ok	0.0	0.0	0.0	0%	0.3	0	52	6100M	1027M	1859M
n012	ok	0.0	0.0	0.0	0%	0.3	0	214	6152M	1027M	1861M
n009	ok	0.0	0.0	0.0	0%	0.3	0	213	6316M	1027M	1861M
n013	unavail										
n014	unavail										
n015	unavail										
n016	unavail										

그런 다음에는 Job submit 파일을 작성한다. 간단한 submit 파일 예제는 다음과 같다.

```
#BSUB -n 24
#BSUB -o 1.out
#BSUB -e 1.err
mpijob ./xhpl
```

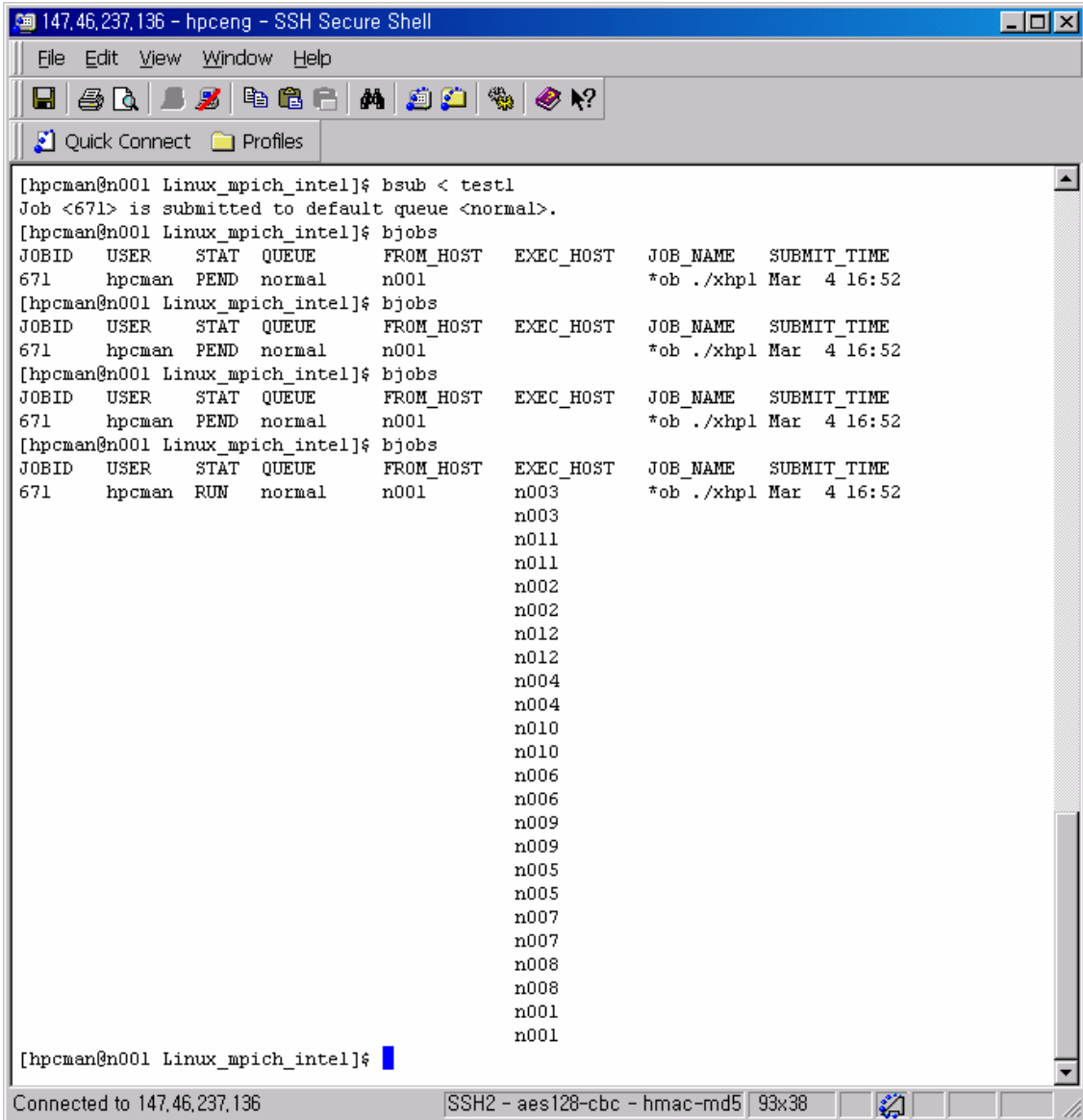
첫 번째 라인은 CPU를 24개 활용하겠다는 것이며, -o 옵션은 코드의 stdout 출력을 저장할 파일이며 -e 옵션은 코드의 stderr 출력을 저장할 파일명이다. 마지막 라인이 수행할 명령어를 적어주는 라인인데, **실제로 유저가 컴파일한 실행파일 앞에 mpijob 이라는 keyword를 반드시 적어 주어야 한다. 그리고 병렬 프로그램 컴파일시에 사용한 명령어와 반드시 일치시켜 주어야 하는데 규칙은 다음과 같다.**

```
mpijob : mpicc/mpicc/mpif77/mpif90 으로 컴파일했을 시
mpijob_gcc : gmpicc/gmpicc/gmpif77으로 컴파일했을 시
mpijob_pgc : pgmpicc/pmpicc/pmpif77/pmpif90으로 컴파일했을 시
```

작업 Submit을 작성하고 적절한 이름으로 저장한다. 예를 들어 위의 Submit 파일예제를 test1이라고 저장했다면 다음의 명령어를 사용해서 submit 한다.

```
$ bsub < test1
```

submit 한 작업이 정상적으로 수행되는지를 알아보기 위해서는 bjobs라는 명령어를 수행시켜 본다. submit을 한 직후에는 stat가 PEND 상태로 있다가 정상적으로 필요한 Resources를 할당받으면 RUN 으로 바뀌어 진다. 그러면 bjobs 명령어를 수행시키면 EXEC_HOST에 현재 작업을 수행하고 있는 시스템의 리스트가 표시된다.



```
[hpcman@n001 Linux_mpich_intel]$ bsub < test1
Job <671> is submitted to default queue <normal>.
[hpcman@n001 Linux_mpich_intel]$ bjobs
JOBID  USER  STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
671    hpcman  PEND  normal  n001
[hpcman@n001 Linux_mpich_intel]$ bjobs
JOBID  USER  STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
671    hpcman  PEND  normal  n001
[hpcman@n001 Linux_mpich_intel]$ bjobs
JOBID  USER  STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
671    hpcman  PEND  normal  n001
[hpcman@n001 Linux_mpich_intel]$ bjobs
JOBID  USER  STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
671    hpcman  RUN   normal  n001      n003      *ob ./xhpl Mar  4 16:52
                                     n003
                                     n011
                                     n011
                                     n002
                                     n002
                                     n012
                                     n012
                                     n004
                                     n004
                                     n010
                                     n010
                                     n006
                                     n006
                                     n009
                                     n009
                                     n005
                                     n005
                                     n007
                                     n007
                                     n008
                                     n008
                                     n001
                                     n001
[hpcman@n001 Linux_mpich_intel]$
```

만약에 submit된 작업이 잘못 수행된 것이어서 사용자가 취소하고 싶은 경우에는 kill 명령어를 실행시킨다. bjobs 명령어로 JOBID를 확인한 다음 실행한다.

```
$ kill 671
```

한편, 병렬 Pool에는 모두 4개의 Queue가 설정되어 있다. 이것은 24개 CPU를 이용한 병렬작업이 Queue에 쌓여 있을 경우에 다른 작업들이 실행이 되지 않는 것을 방지하기 위해서이다. 새벽 2시부터 오전 8시까지는 16개 CPU 작업이 가장 높은 우선권을 가지며 실행되고, 나머지 시간대에는 4개 이하의 CPU 작업이 가장 우선권이 높다. 24개의 CPU를 요구하는 작업은 항상 낮은 우선권을 가지며, Background 환경으로 실행이 된다.

```
=====
Q NAME : cpu_4
Q priority :
  새벽 2시 ~ 8시까지 : LOW
  기타시간 : VERY HIGH
cpu limit : 4개까지
=====
Q NAME : cpu_8
Q priority : 새벽 2시 ~ 8시 : LOW
  기타시간 : HIGH
cpu limit : 8개까지
=====
Q NAME : cpu_16
Q priority : 새벽 2시 ~ 8시 : HIGH
  기타시간 : LOW
cpu limit : 16개까지
=====
Q NAME : cpu_24
Q Priority : low
cpu limit=24
```

유저의 경우에 job을 submit 할 때, 특별히 Queue를 지정하지 않아도 유저가 요구한 CPU 개수를 보고 자동으로 시스템에서 판단하여 Queue를 지정하도록 되어 있다. 아래는 cpu 6개를 요구한 유저 작업을 submit 했을 시의 실행 모습이다.

```
[hpcman@n001 Linux_mpich_intel]$ bsub < test1
No running job found in queue <cpu_4>
No pending job found in queue <cpu_4>
No running job found in queue <cpu_8>
No pending job found in queue <cpu_8>
No running job found in queue <cpu_16>
No pending job found in queue <cpu_16>
No running job found in queue <cpu_24>
No pending job found in queue <cpu_24>
... hpcman 's Job submitted to cpu_8 Queue
Job <1880> is submitted to queue <cpu_8>.
```

만약 유저가 특정 Queue를 지정하고자 할 경우에는 -q 옵션을 줄 수 있다. 이러한 경우는 다음에 설명할 Parametric Study의 경우에 유용하게 사용할 수 있다.

④ 반복적 작업 수행 : Parametric Study

클러스터 시스템은 병렬 응용 프로그래밍의 개발 및 수행에도 많이 활용되지만, 많은 공학 분야에서 널리 활용되는 Parametric Study와 같이 동일한 작업을 여러 개의 입력데이터를 이용해서 수행해야 하는, 이른바 High-throughput Computing 환경에도 매우 적합하다. hpceng 시스템의 Scheduler인 LSF에서도 High-throughput Computing을 쉽게 구현할 수 있다.

예) p4c 라는 실행코드를 이용하여 10개의 서로 다른 입력데이터를 이용해서 계산을 수행한 후 Stdout 출력을 서로 다른 파일에 저장할 경우.

먼저 수행할 프로그램을 컴파일한다. 인텔 컴파일러, GNU gcc 컴파일러 등 적합한 컴파일러로 컴파일한다. 그런 다음, 입력데이터 파일을 첨자를 이용해서 구별되도록 작성한다. 10개의 입력데이터일 경우 다음과 같이 작성하고, 실행코드와 같은 위치에 옮겨 놓는다.

```
input.1 input.2 ..... input.10
```

그런 다음, bsub 명령을 통해서 작업을 submit을 하는데, 문법은 다음과 같다.

```
bsub -J "arrayname[indexlist, ...]" myjob
```

위의 예제의 경우에는 다음과 같이 수행할 수 있다.

```
$ bsub -J "testarray[1-10]" -i"input.%" -o"output.%" -e"err.%" ./p4c
```

(주의 %i에서 i는 대문자 i입니다)

위의 명령을 통해서 실제로 수행하는 작업은 다음과 같다.

```
./p4c input.1 : stdout => output.1 , stderr => err.1
```

```
./p4c input.2 : stdout => output.2 , stderr => err.2
```

```
.....
```

```
./p4c input.10 : stdout => output.10 , stderr => err.10
```

그런데 이러한 Parametric Study의 경우에는 동시에 요구되는 cpu는 1개 이므로 앞에서 설명한 것처럼 가장 우선권이 높은 cpu_4 Queue를 사용하게 된다. 이럴 경우의 문제는 우선권이 높아서 작업이 빨리 실행되기는 하지만, 동시에 4개의 cpu밖에 활용할 수 없게 된다. 따라서 16개 이상의 경우의 수를 보다 빨리 수행시키기 위해서는 -q 옵션으로 cpu_24 Queue를 지정해 주는 것이 바람직하다.

```
$ bsub -q cpu_24 -J "testarray[1-10]" -i"input.%" -o"output.%" -e"err.%" ./p4c
```

submit 한 작업을 상황을 알아보기 위해서는 앞에서 설명한 bjobs 명령을 통해서 살펴볼 수 있다. array를 이용해서 submit 한 작업은 모두 동일한 JOBID를 가지게 된다. bjobs 명령을 통해서 array를 통해 submit된 작업의 전체 정보를 보려면 다음과 같이 사용한다.

bjobs -A JOBID

```
[hpcman@n001 hp-through]$ bjobs
JOBID  USER  STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIM
798    hpcman  PEND  normal  n001              *tarray[1] Mar  8 14:
798    hpcman  PEND  normal  n001              *tarray[2] Mar  8 14:
798    hpcman  PEND  normal  n001              *tarray[3] Mar  8 14:
798    hpcman  PEND  normal  n001              *tarray[4] Mar  8 14:
798    hpcman  PEND  normal  n001              *tarray[5] Mar  8 14:
798    hpcman  PEND  normal  n001              *tarray[6] Mar  8 14:
798    hpcman  PEND  normal  n001              *tarray[7] Mar  8 14:
798    hpcman  PEND  normal  n001              *tarray[8] Mar  8 14:
798    hpcman  PEND  normal  n001              *tarray[9] Mar  8 14:
798    hpcman  PEND  normal  n001              *array[10] Mar  8 14:
[hpcman@n001 hp-through]$ bjobs -A 798
JOBID  ARRAY_SPEC  OWNER  NJOBS  PEND  DONE  RUN  EXIT  SSUSP  USUSP  PSUSP
798    testarra   hpcman  10     0    0    10   0     0     0     0
[hpcman@n001 hp-through]$ bjobs
JOBID  USER  STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIM
798    hpcman  RUN   normal  n001        n006       *tarray[1] Mar  8 14:
798    hpcman  RUN   normal  n001        n006       *tarray[2] Mar  8 14:
798    hpcman  RUN   normal  n001        n008       *tarray[3] Mar  8 14:
798    hpcman  RUN   normal  n001        n008       *tarray[4] Mar  8 14:
798    hpcman  RUN   normal  n001        n010       *tarray[5] Mar  8 14:
798    hpcman  RUN   normal  n001        n010       *tarray[6] Mar  8 14:
798    hpcman  RUN   normal  n001        n009       *tarray[7] Mar  8 14:
798    hpcman  RUN   normal  n001        n009       *tarray[8] Mar  8 14:
798    hpcman  RUN   normal  n001        n012       *tarray[9] Mar  8 14:
798    hpcman  RUN   normal  n001        n012       *array[10] Mar  8 14:
[hpcman@n001 hp-through]$ bjobs
```

submit 한 작업을 취소할 때도 array 전체 작업을 한번에 취소할 수 있고 각각의 단위 작업 별로 취소할 수 있다.

\$ bkill JOBID : 전체 작업 취소
(Ex. bkill 798)

\$ bkill "JOBID[Index]"
(Ex. bkill "798[2]" 798의 JOBID 작업 중 2번째 작업 취소)

hpceng 시스템에서 작업의 dependency를 설정하는 방법은 `-w "done(Jobname)"`이라는 옵션을 사용하는 것입니다.

한가지 명심할 것은 jobname을 이용하기 위해서 `-J` 옵션으로 모든 Q 작업에 Jobname을 설정해 주는 것입니다.

예를 들어서 p4c라는 serial job과 test1 파일에 명시된 병렬 job을 이용해서 돌리는 것을 살펴 보겠습니다.

먼저 test1 파일에 병렬 job을 설정해 줍니다.

```
#BSUB -n 8
#BSUB -o 1.out
#BSUB -e 1.err
mpijob ./xhpl
```

그런 다음, jobexec 라는 파일을 만들어서 다음과 같이 적습니다.

```
bsub -J Job1 ./p4c
bsub -J mpijob1 -w "done(Job1)" < test1
bsub -J Job2 -w "done(mpijob1)" ./p4c
bsub -J mpijob2 -w "done(Job2)" < test1
```

저장하고 나와서 `chmod +x jobexec`라는 명령을 통해서 실행권한을 설정해 줍니다.

jobexec 파일을 살펴 보면 두번째 라인의 의미는

Job1 이라는 이름의 Jobname을 가지는 작업이 완료되면 test1에 설정된 병렬작업을 수행하라는 의미입니다. Job1은 첫번째 라인에 명시된 p4c serial 작업입니다.

세번째 라인의 의미는 두번째 라인에 정의된 mpijob1이 완료되면 p4c 코드를 돌리라는 의미입니다.

jobexec를 수행하실때는 `./jobexec`라고 하시면 되고

각 단계별로 `bjobs`를 살펴 보면 다음과 같습니다.

```
[hpcman@n009 Linux_mpich_intel]$ ./jobtest3
Job <1099> is submitted to default queue <normal>.
Job <1100> is submitted to default queue <normal>.
Job <1101> is submitted to default queue <normal>.
Job <1102> is submitted to default queue <normal>.
```

```
[hpcman@n009 Linux_mpich_intel]$ bjobs
JOBID  USER   STAT  QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
1099   hpcman  PEND  normal     n009              Job1       Mar 25 11:10
```

1100	hpcman	PEND	normal	n009		mpijob1	Mar 25 11:10
1101	hpcman	PEND	normal	n009		Job2	Mar 25 11:10
1102	hpcman	PEND	normal	n009		mpijob2	Mar 25 11:10

[hpcman@n009 Linux_mpich_intel]\$ bjobs

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
1099	hpcman	RUN	normal	n009	n002	Job1	Mar 25 11:10
1100	hpcman	PEND	normal	n009		mpijob1	Mar 25 11:10
1101	hpcman	PEND	normal	n009		Job2	Mar 25 11:10
1102	hpcman	PEND	normal	n009		mpijob2	Mar 25 11:10

[hpcman@n009 Linux_mpich_intel]\$ bjobs

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
1100	hpcman	RUN	normal	n009	n005 n012 n012 n007 n007 n006 n006	mpijob1	Mar 25 11:10
1101	hpcman	PEND	normal	n009		Job2	Mar 25 11:10
1102	hpcman	PEND	normal	n009		mpijob2	Mar 25 11:10

[hpcman@n009 Linux_mpich_intel]\$ bjobs

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
1101	hpcman	RUN	normal	n009	n011	Job2	Mar 25 11:10
1102	hpcman	PEND	normal	n009		mpijob2	Mar 25 11:10

[hpcman@n009 Linux_mpich_intel]\$ bjobs

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
1102	hpcman	RUN	normal	n009	n003 n003 n008 n008 n004 n004 n010 n010	mpijob2	Mar 25 11:10

4. 공학용 Package 이용하기

현재, 설치된 공학용 소프트웨어들과 해당 시스템은 다음과 같다.

fluent : hpceng4.snu.ac.kr (147.46.237.140)
diana : hpceng3.snu.ac.kr (147.46.237.139)
matlab : hpceng2.snu.ac.kr (147.46.237.138)
abaqus : hpceng1.snu.ac.kr (147.46.237.137)
ls-dyna: hpceng.snu.ac.kr (147.46.237.136)

3절 1항의 로그인 하기 항목을 참조하여 SSH를 이용하여 위의 해당 시스템으로 접속하면 사용할 수 있다. 각 소프트웨어별로 이용가능한 On-line 매뉴얼 등은 공대 홈페이지 (<http://eng.snu.ac.kr>)에서 전산 환경을 클릭한 다음 소프트웨어 설명을 보면 이용가능하다.

① fluent

hpceng4.snu.ac.kr로 접속한 후에 xterm을 실행시키면 fluent, gambit 등의 명령어를 통해서 사용할 수 있다.

② matlab

hpceng2.snu.ac.kr로 접속한 후에 xterm을 실행시키고 matlab이라는 명령어로 이용가능하다. matlab 자체적으로 help 명령어를 통해서 각 Command 별로 상세한 정보를 얻을 수 있다. GUI 인터페이스를 쓰지 않고 싶으면 -nodesktop 옵션을 주고 실행시키면 text 모드로 쓸 수 있다.

③ abaqus

hpceng1.snu.ac.kr로 접속한 후에 abaqus 라는 명령어로 사용할 수 있다. abaqus viewer 등을 실행하기 위해서는 xterm을 실행 한 후에 수행한다.

④ ls-dyna

hpceng.snu.ac.kr로 접속한 후에 "lsdyna ncpu=<cpu 개수> i=<input file>"의 명령으로 실행 시킬 수 있으며 stdout은 <input file>.out으로, stderr는 <inputfile>.err로 각각 저장된다. ls-dyna 작업을 위해 할당 될 수 있는 최대 CPU 개수는 8개 이다. lspost를 실행하기 위해서는 xterm을 실행 한 후에 수행한다.