클러스터 시스템 사용자 매뉴얼

2009. 09

IBM-KAIST Cluster Computing Center

차 례

제 1장 사용자 안내

1.시스템 소개

본 센터의 장비는 IBM 사의 클러스터 시스템입니다. 총 16 노드로 구성되어 있으며, 이중 1 노드는 관리노드, 15 노드는 계산 노드로 구성되어 있습니다.

구 분		내 용	수 량
HPC 서버 X3550	CPU	QC Intel Xeon 2.50GHz x 2 (8Core)	16SET
	Memory	32GB	
	HDD	146GB SAS 10K	
	OS	Open Linux	
	관리SW	CM Support	
STORAGE N3700 (NAS)		IBM SYSTEM STORAGE	1SET
	HDD	300GB, 10K FC HDD x 10EA	
	관리SW	CIFS, NFS, CFO	
기타	KVM Switch	IBM 2x16 Console Switch	1SET
	모니터	17" 랙 모니터	1SET
	NETWORK	Switchhub 48ports Gigabit	2SET

2.시스템 접속 정보

- 시스템 주소 : 143.248.136.39
- 접속도메인 : node.kaist.ac.kr, csnode001.kaist.ac.kr
- 접속방법 : SSH, SFTP

※필요시 VNC 를 이용하여 X-Window 사용도 가능합니다.

3.시스템의 구성

본 센터의 장비는 총 16 개 노드와, NAS 스토리지로 구성되어 있습니다. 또한 노드의 구성은 1 개의 관리노드와 15 개의 계산노드로 구분됩니다. 즉, 사용자는 1 개의 관리노드에 접속하여 15 개의 노드를 이용하여 계산을 수행하게되며, 계산시에는 최대 120 개의 Processor(1 노드당 8 개 Processor)를 사용하실 수 있습니다.

```
########## Network Access Storage ###########
172.20.101.100 nas
172.20.101.101 nas2
########## Management Node ####################
172.20.101.1 node001.kaist.ac.kr node001
########## Data and network Nodes ######################
172.20.101.1 node001.kaist.ac.kr node001
172.20.101.2 node002.kaist.ac.kr node002
172.20.101.3 node003.kaist.ac.kr node003
172.20.101.4 node004.kaist.ac.kr node004
172.20.101.5 node005.kaist.ac.kr node005
172.20.101.6 node006.kaist.ac.kr node006
172.20.101.7 node007.kaist.ac.kr node007
172.20.101.8 node008.kaist.ac.kr node008
172.20.101.9 node009.kaist.ac.kr node009
172.20.101.10 node010.kaist.ac.kr node010
172.20.101.11 node011.kaist.ac.kr node011
172.20.101.12 node012.kaist.ac.kr node012
172.20.101.13 node013.kaist.ac.kr node013
172.20.101.14 node014.kaist.ac.kr node014
172.20.101.15 node015.kaist.ac.kr node015
172.20.101.16 node016.kaist.ac.kr node016
########## Compute Nodes ######################
172.20.0.1 com001.kaist.ac.kr com001
172.20.0.2 com002.kaist.ac.kr com002
172.20.0.3 com003.kaist.ac.kr com003
172.20.0.4 com004.kaist.ac.kr com004
172.20.0.5 com005.kaist.ac.kr com005
172.20.0.6 com006.kaist.ac.kr com006
172.20.0.7 com007.kaist.ac.kr com007
```

제 2장 사용자환경

1.로그인

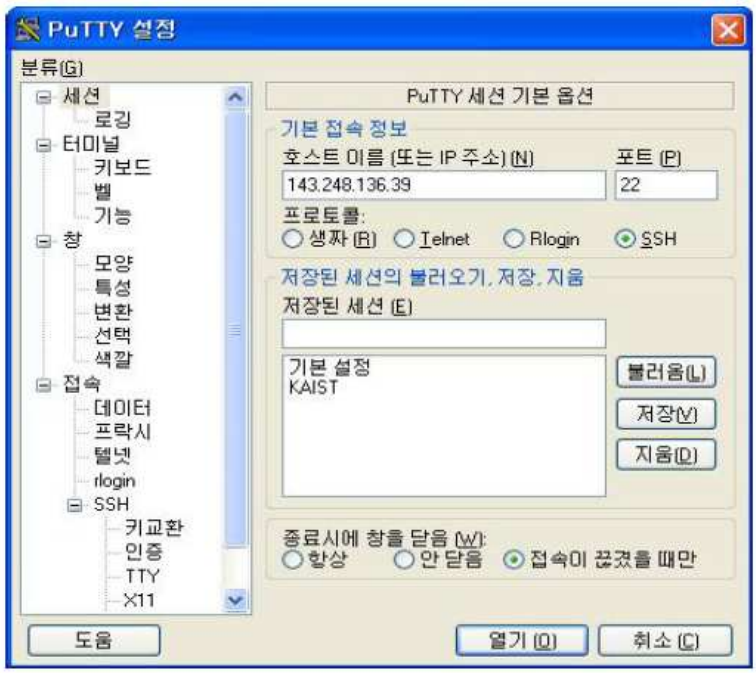
현재 로그인을 위해 SSH, SFTP, VNC 서비스를 제공하고 있습니다.

1)Unix, Linux 등에서

$ssh - l 사용자 ID 143.248.136.39

2)윈도우즈에서

-SSH Secure Shell Client 나 putty 등의 ssh 접속 프로그램 이용



[Putty를 이용한 접속 예]

3)VNC 를 통해 접속하기

vncserver 를 이용하여 원격에서 GUI 작업을 하실 수 있습니다.

아래와 같이 작업하시면 됩니다 (vncserver 명령 -> PW 를 입력)

```
[kaist@node001 simul]$ vncserver
You will require a password to access your desktops.
Password:
Verify:
xauth: creating new authority file /home/kaist/.Xauthority
New 'node001:6 (kaist)' desktop is node001:6 <- :6 번은 viewer 접속시 입력)
Creating default startup script /home/kaist/.vnc/xstartup
Starting applications specified in /home/kaist/.vnc/xstartup
```

위 작업이 완료된 후 클라이언트에서 VNC Viewer 프로그램을 이용하여 접속하시면 됩니다. (주의 : 143.248.136.39 번 IP 뒤에 해당 포트 번호를 정확히 입력해 주어야 함)

위 작업이 완료되면 기본적으로 TWM(Tab Window Manager) 환경이 실행되도록 되어 있습니다. 이에 실제 사용하는 GUI 와 동일하게 해주기 위해서는 다음과 같이 작업하셔야 합니다.

[kaist@node001 .vnc]$cd 사용자홈디렉/.vnc

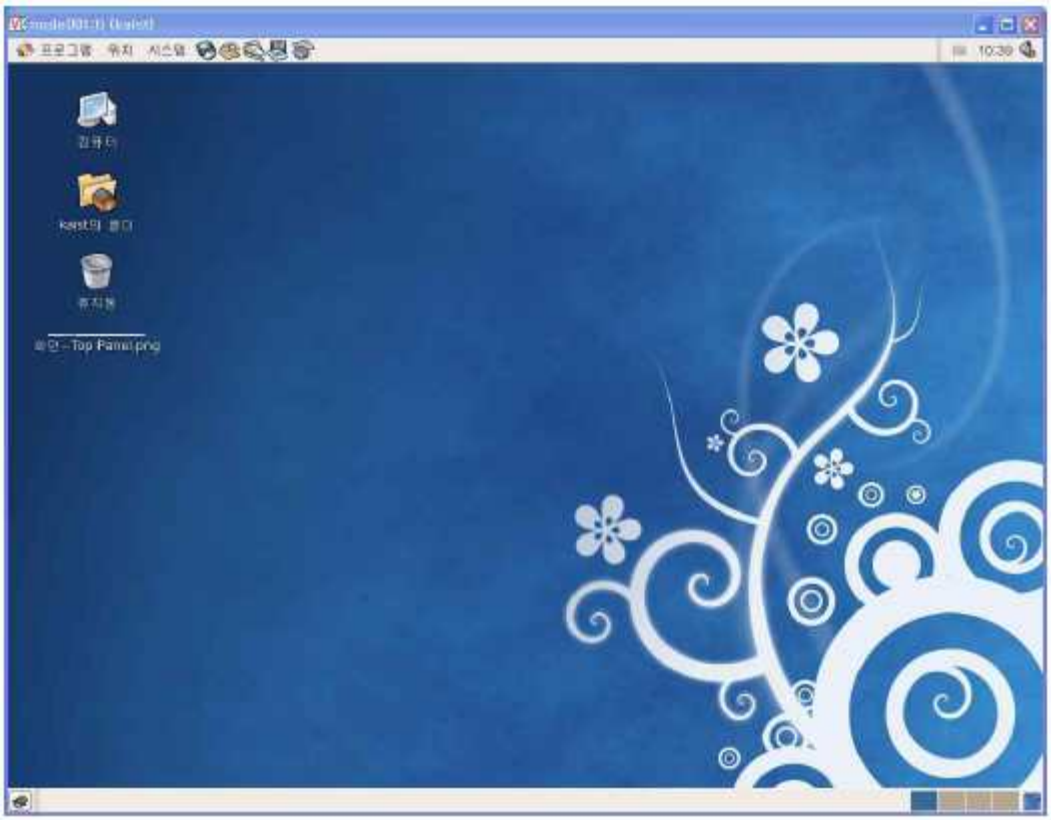
[kaist@node001 .vnc]$mv xstartup xstartup.bak

- TWM 환경 백업

[kaist@node001 .vnc]$ cp /etc/X11/xinit/xinitrc xstartup

- 현재 설정된 GUI 환경파일 복사

기존에 실행되던 vncserver 를 재시작 후에 다시 접속하시면 기존 TWM 환경에서 변경된 것을 확인할 수 있습니다.



4) 클러스터 하위 노드에 접속하기

클러스터 하위 노드에 접근하기 위해서는 node001 번에 접속한 후 SSH 를 이용하여 하위 노드에 접속하셔야 합니다.

```
$ssh node002 <- node002 번에 접속하기
$ssh node003 <- node003 번에 접속하기
```

※rsh 은 본 클러스터에서 지원하지 않습니다. rsh 대신 ssh 를 사용하시면 됩니다.

2.사용자 쉘 변경

-기본 쉘은 bash Shell

-쉘 변경은 다음과 같다.

```
$chsh
```

3.홈/작업 디렉터리

홈 디렉터리에는 용량에 제한이 있으므로 대용량 작업이 필요한 경우 /data 에서 작업하시기 바랍니다.

디렉터리 명	경로	용량제한
홈 디렉터리	/home/계정명	soft : 3G (3G 이후 경고) hard : 4G (4G 이후 사용제한) 유예기간 : 7day (3G 이상에 대해서는 7 일간 유예기간)
작업디렉터리(스크래치)	/data	무제한

※ node001 에 존재하는 홈 디렉터리는 전 하위노드에 공유되어 있어, 하위노드는 node001 과 동일한 홈 디렉터리 내용을 가지게 되어 있습니다.

4.사용자 프로그래밍 환경

연구진행 및 클러스터 사용에 필요한 라이브러리 및 SW 는 사용자 요청시 수시로 설치해 드리고 있습니다. 필요한 라이브러리 및 SW 가 있을 경우, 홈페이지 기술지원요청 및 이메일 등을 이용하여 요청 바랍니다.

구 분	항 목
gcc	4.1.2
mpi	mpich-1.2.7p1
java	java-1.6.0
perl	perl5
python	python 2.4
MVAPICH	mvapich-1.1.0-0.2931.3.el5 mvapich2-1.0.3-3.el5
수치해석	Matlab2009a
작업관리	Torque, Maui, OpenPBS
기타	PVM, R-package

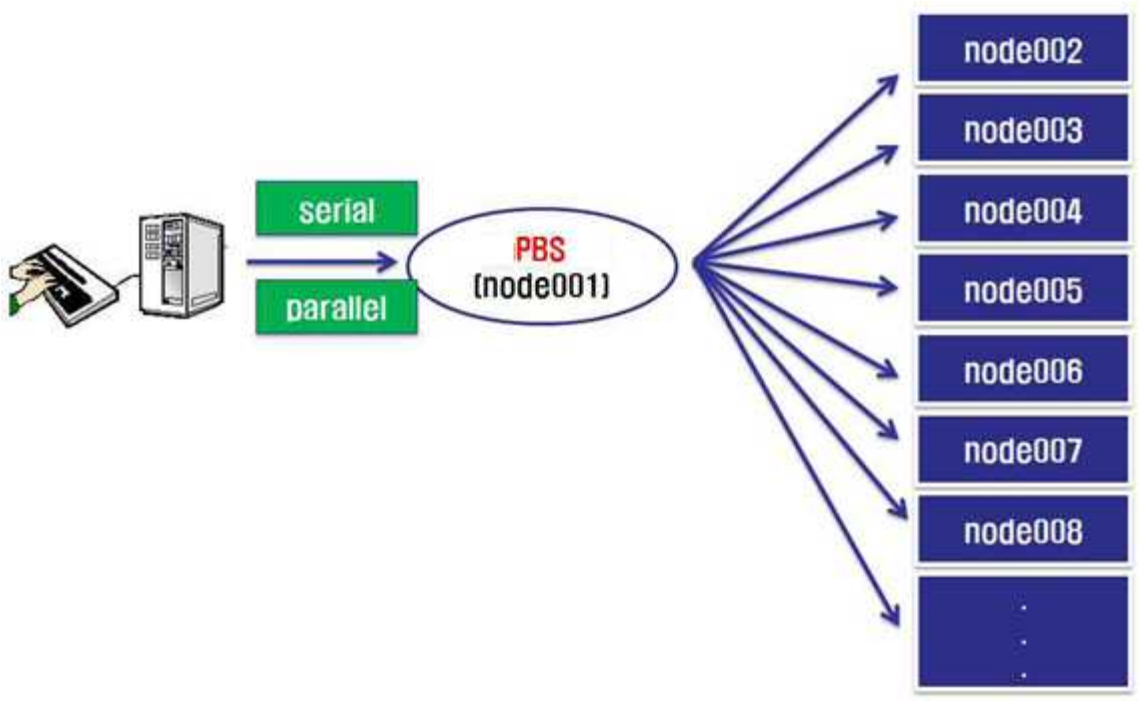
5.작업관리자

1)작업관리자 개요

ㅁ PBS(Portable Batch System)를 이용한 배치 작업의 실행

본 센터의 클러스터에는 PBS 로 Torque 를 사용하고 있으며 PBS 는 배치 작업 스케줄링 프로그램으로서 배치 작업이 네트워크로 연결된 여러 컴퓨터 자원들 중에서 최적의 자원을 사용할 수 있도록 할당해 주는 기능을 합니다. 이것은 여러 사용자가 배치 작업을 수행할 때 작업을 분산시킬 수 있어 전체 시스템의 성능을 향상시키고 사용자가 최적 상태의 시스템을 직접 확인하는 번거러움을 줄여줍니다.

본 클러스터에서는 쉘 스크립트 작업, 순차작업(Serial), 병렬작업(Parallel) 등 거의 대부분의 작업을 처리할 수 있습니다.



[PBS를 이용한 작업관리 예]

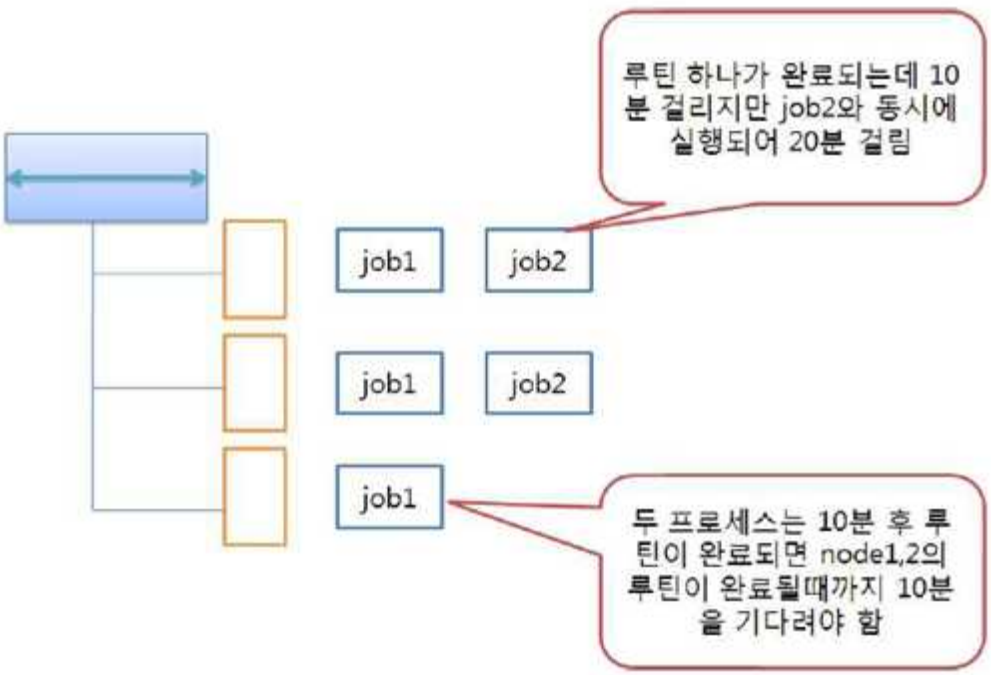
2) 작업관리자

클러스터는 단일 사용자가 사용하는 시스템이 아니므로 하나의 사용자가 작업을 실행하는 중에 다른 사용자도 클러스터를 사용하려고 할 것입니다. 이때 작업 하나가 실행되는 중에 다른 사용자가 로그인해서 작업을 실행하면, 현재의 운영체제는 모두 다중

실행을 지원하므로 이미 실행하던 작업의 자원을 빼앗아 새 작업이 같이 실행됩니다. 클러스터가 아닌 시스템에서는 먼저 실행한 작업의 시간이 늘어나는 것으로 끝나겠지만, 클러스터에서는 다른 문제가 하나 더 있습니다. 클러스터는 여러 노드가 하나의 작업을 나누어 실행합니다. 이때 노드에 배분할 작업의 양은 프로그래밍 단계에서 정해지기 때문에 여러 노드를 사용해 작업이 실행되는 상태에서 다른 사용자가 일부 노드만을 사용해 다른 작업을 실행시키면 그 작업이 실행되는 노드의 속도는 느려지고, 다른 노드는 자신의 프로세스를 실행한 후 그 노드의 프로세스가 끝날 때까지 대기해야 합니다. 이러한 식으로 클러스터에서는 심각한 효율의 저하를 가져올 수 있습니다. 그렇기 때문에 시스템을 효율적으로 사용할 수 있으려면, 임의의 순간에 어느 노드를 보아도 그 노드는 작업 하나만을 실행하고 있어야 합니다.

이때, 사용자가 일단 시스템에 로그인해서 먼저 실행중인 프로세스가 있는지 확인하고, 있으면 프로세스가 끝날 시간을 예측해서 그 때 다시 로그인해 작업을 실행시키는 것은 불편한 일이 아닐 수 없습니다. 따라서 이것을 자동화 시켜 주는 프로그램이 작업 관리자입니다.

작업 관리자는 일단 사용자가 시스템에서 직접 작업을 실행하지 못하도록 하고, 사용자의 작업 명령을 받아 자신의 대기열(queue)에 쌓아 둡니다 다음 시스템이 작업을 마치고 대기 상태가 되면 즉시 대기열에 있던 작업을 순서대로 실행시킵니다. 따라서 여러 작업을 중복해서 실행하거나 작업을 사용자 스스로 순차적으로 실행시킬 때 작업간의 시간 공백을 없애 주므로 시스템을 효율적으로 사용할 수 있습니다.



[PBS를 이용한 작업관리 예]

작업관리자는 여러 사용자의 작업을 단일 프로그램이 관리해야 하는 특성상 root 또는 지정된 관리자 계정만이 관리할 수 있고, 평소에는 데몬으로 시스템에서 대기하고 있습니다.

단일 시스템에서는 하나의 작업이 끝나면 바로 대기열에 있는 다음 작업을 실행하면 되지만, 클러스터는 프로세스를 노드 단위로 배분하기 때문에, 한 단계를 더 고려해야 합니다. 클러스터에서 사용자가 반드시 모든 노드를 사용하지는 않습니다. 16 개 노드를 가진 클러스터에서 한 사용자가 10 개를 사용할 경우 다른 사용자가 5 개를 사용하려고 하면 그 사용자의 작업은 다른 노드에 배분해서 즉시 실행합니다. 하지만 또 다른 사용자가 5 개를 사용하려고 하면 그 사용자의 작업은 대기하고 있어야 합니다.

※ Backfilling : 노드를 사용하는 작업이 먼저 대기열에 들어 있더라도 4 개를 사용하

리눅스 클러스터에 많이 사용되는 작업 관리자는 condor, maui, OpenPBS 등이 있으며 모두 사용법과 개념은 거의 같음.

3) PBS(torque) 데몬의 구성

pbs_server 데몬 : 사용자가 입력하는 작업을 받고 관리하는 데몬.

pbs_sched 데몬 : 전체 노드의 자원을 모니터링하고 작업을 언제 어디에서 실행시킬지 결정하는 데몬 (※ 본 클러스터 시스템에서는 psb_sched 데몬 대신 maui 스케줄러를 사용하고 있음.)

pbs_mon : 노드에서 작업을 실행시키는 데몬

pbs_server, pbs_sched 데몬은 사용자가 직접 사용하는 노드에서만 실행되면 되고 pbs_mon 데몬은 작업이 실행될 모든 노드에서 실행됨.

PBS 에 작업을 넣을 때에는 실행 파일의 이름을 직접 넣는 것이 아니라 파일이 실행될 때 적용받는 여러 속성을 정의한 스크립트를 작성해서 집어넣습니다. 이때 작업마다 일일이 속성을 정의하지는 않고 미리 여러 가지의 속성을 정의한 속성 템플릿을 정의합니다. 작업을 넣을 때에는 '이 작업은 이러한 속성 템플릿의 적용을 받는다'의 식으로 스크립트를 작성합니다. 따라서 사용자의 작업마다 일일이 여러 속성을 정의하는 것보다는 간편합니다.

PBS 에서는 이러한 속성 템플릿을 작업 큐라고 합니다. 따라서 본래의 큐의 의미와 혼동하지 않도록 합니다. 여기서는 특별히 지적하지 않는 한 큐는 PBS 가 가리키는 개체를 지칭하는 것으로 사용하겠습니다.

큐에서 지정할 수 있는 속성에는 최대시간, 최대 사용 노드 수, 우선순위, 최대로 사용할 수 있는 resource, 동시에 최대로 이 큐를 사용할 수 있는 사용자의 수 등이 있습니다. PBS 는 클러스터 전체(server), 큐, 작업(jobs) 세가지에 대해 속성을 정의할 수 있습니다.

지정 가능한 속성
최대시간
최대사용 노드 수
우선순위
최대치 resource
큐를 사용할 수 있는 최대 사용자 수

□ 속성을 설정하는 명령

```
[root@node001 sysconfig]# qmgr
Max open servers: 4
```

이 명령은 qmgr 명령 프롬프트에서 속성을 설정할 수도 있고 스크립트를 작성해서 직접 입력할 수도 있습니다. 또는 qmgr -c “명령 구문”으로 쉘에서 할 수도 있습니다. qmgr 은 여러 사용자에 적용되는 설정을 관리하기 때문에, root 또는 지정된 계정만이 사용할 수 있고 일반 사용자는 설정된 속성을 확인할 수만 있습니다.

□ qmgr 명령의 형식

```
qmgr {명령} {적용받을 개체} {속성}
```

명령은 다음과 같은 것이 있습니다.

명 령	내 용
active	활성화된 객체를 지정
create	새로운 개체를 만듬
delete	지정한 객체를 제거
set	객체의 속성을 설정
unset	설정된 객체의 속성을 제거
list	객체에 설정되어 있는 속성을 보여줌 list {개체}의 형식으로만 사용됨
print	객체의 특정한 속성을 보여줌 print {개체} {속성}의 형식으로 사용

※명령과 개체는 줄여 쓸 수 있음. ex) set server = s s, print queue = p q

PBS 에서 실행되는 모든 작업은 큐에 속해 있어야 하므로, 작업을 넣을 때 큐 이름을 정해주지 않으면 기본으로 사용할 큐가 반드시 하나 있어야 합니다.
큐는 자신이 직접 작업을 실행하는 실행(execution)큐와 자신이 작업을 갖고 있다가 다른 큐에 넘겨주고 실행하도록 하는 전달(route) 큐가 있습니다. 전달 큐는 작업을 넘겨받을 실행 큐를 속성으로 따로 지정해야 합니다.

□ 큐 만들기

```
#
#Set server attributes.
#
set server scheduling = True
set server default_queue = defaultq
set server log_events = 511
set server mail_from = adm
```

관리자 속성은 큐에 들어온 작업을 PBS 가 실행하게 하는 것으로 당연히 True 로 설정합니다. query_other_jobs 는 qstat 명령으로 실행되고 있거나 대기하고 있는 큐를 확인할 때 다른 계정의 작업도 확인할 수 있도록 합니다. (나머지는 기본값으로 두면 됩니다)

□ root 외에 다른 계정이 qmgr 로 서버와 큐 속성을 바꿀 수 있게 하려면 managers 속성을 지정하기

```
set server manager = user1@node1.kaist.ac.kr, user2@node001.kaist.ac.kr
```

□ node_pack 속성은 SMP 노드에서 한 노드에 하나의 작업만 들어가도록 하기.

```
set server node_pack = {true,false }
```

정의되어 있지 않으면 nodes 파일에 있는 순서에 따라 작업이 할당

□ node001 에서만 작업제출 하는 qsub 명령을 사용할 수 있도록 하려면

```
set server acl_hosts = node001@kaist.ac.kr
다른노드를 추가하기 위해서는
```

□ 속성에서는 *를 사용할 수 있습니다. 따라서 모든 노드에서 qsub 를 사용할 수 있도록 하는 명령은 다음과 같습니다.

```
set server acl_hosts =*.kaist.ac.kr 또는 *
```

□ 자원을 설정할 수 있는 속성 키워드는 resources_available, resources_cost, resources_defaults, resources_max 등이 있는데 대개 최대값을 지정하는 resources_max 만 쓰면 됩니다.

예를 들어 서버에서 실행되는 작업 하나의 최대 시간을 1 시간 20 분으로 하는 설정은

```
set server resources_max.walltime=1:20:00
```

자원 속성뒤에 . {속성을 설정할 자원 이름}을 붙이고 정할 값을 지정해 주면 됨.
$man pbs_resources 에 나와 있습니다.

□.{속성을 설정할 자원 이름} 항목

자원명	내 용
cput	Maximum amount of CPU time used by all processes in the job.
file	The largest size of any single file that may be created by the job. Units: size.
nice	The nice value under which the job is to be run. Units: uni-tary.
pcput	Maximum amount of CPU time used by any single process in the job. Units: time.
pmem	Maximum amount of physical memory (workingset) used by any single process of the job. Units: size.
pvmem	Maximum amount of virtual memory used by any single process in the job. Units: size.
vmem	Maximum amount of virtual memory used by all concurrent pro-cesses in the job. Units: size.
walltime	Maximum amount of real time during which the job can be in the running state. Units: time.
arch	Specifies the administrator defined system architecture requried. This defaults to whatever the PBS_MACH string is set to in "local.mk". Units: string.
host	Name of host on which job should be run. This resource is provided for use by the site's scheduling policy. The allowable values and effect on job placement is site depen-dent. Units: string.
nodes	Number and/or type of nodes to be reserved for exclusive useby the job. . To ask for 12 nodes of any type: -l nodes=12 . To ask for 2 "server" nodes and 14 other nodes (a total of 16): -l nodes=2:server+14 The above consist of two node_specs "2:server" and "14". . To ask for (a) 1 node that is a "server" and has a "hippi" interface, (b) 10 nodes that are not servers, and (c) 3 nodes that have a large amount of memory an have hippi: -l nodes=server:hippi+10:noserver+3:bigmem:hippi . To ask for three nodes by name: -l nodes=b2005+b1803+b1813 . To ask for 2 processors on each of four nodes: -l nodes=4:ppn=2

	. To ask for 4 processors on one node: -l nodes=1:ppn=4 . To ask for 2 processors on each of two blue nodes and three processors on one red node: -l nodes=2:blue:ppn=2+red:ppn=3
host	Allows a user to specify the desired execution location.
other	Allows a user to specify site specific information. This resource is provided for use by the site's scheduling policy.
software	Allows a user to specify software required by the job. This is useful if certain software packages are only available on certain systems in the site.
qsub -l nodes=15,walltime=2:00:00 script or in a qsub script as a directive: #PBS -l nodes=15,walltime=2:00:00 qsub -l cput=1:00:00,walltime=2:00:00,file=50gb,mem=15mb script qalter -lcput=30:00,pmem=8mb 123.jobid or in a qsub script as a directive: #PBS -l cput=1:00:00,walltime=2:00:00,file=50gb,mem=15mb	

ㅁ서버 속성을 설정 후 작업을 받을 큐를 만듬

```
#
#Create queues and set their attributes.
#Create and define queue
#
create queue pbsq_day
set queue pbsq_day queue_type = Execution
set queue pbsq_day enabled = True
```

enabled, started 속성은 기본 속성이므로 true 로 놓아야 함

클러스터를 쓰는 사용자는 자신의 작업을 실행하는데 저마다 다른 조건이 필요할 수 있습니다. 어떤 사용자는 하루 단위로 작업을 완료할 수 있지만 다른 사용자는 한달까지 계속 실행될 수도 있고, 어떤 사용자는 노드 4 개만으로도 충분하지만 다른 사용자는 모든 노드를 다 써야 할 수도 있습니다.

이때에는 사용자가 자신에 맞는 조건을 가진 큐를 골라 쓸 수 있도록 큐를 여러개 만들어 놓고 쓰일 특성에 따라 속성을 주면 됩니다.

시간에 제한을 두거나, 노드 수에 제한을 둘 수도 있고 큐를 쓸수 있는 사용자나 소프트웨어를 제한할 수도 있습니다. 그리고 큐가 여러개 있으면 그들 사이에 우선순위를 두어 순위가 높은 큐를 쓴 작업은 나중에 대기 큐에 들어와도 먼저 실행되도록 할 수 있습니다.

pbsq1 이 pbsq2 보다 우선순위가 높도록 지정하려면 다음과 같이 합니다.

```
set queue pbsq1 priority=1
```

priority 속성이 지정되지 않은 큐의 값은 0 입니다. 따라서 다른 큐보다 순위가 높으려면 1 에서 1023 사이에서 임의로 아무 값이나 지정합니다. 물론 다른 큐의 값이 지정되어 있으면 그보다 높은 값을 지정해주면 그 큐를 먼저 실행하도록 할 수 있습니다. 우선순위 값은 -1023 에서 1023 까지 쓸수 있으므로 다른 큐보다 우선 순위를 낮추는 설정도 가능합니다.

큐의 자원을 설정할 수 있는 속성 키워드는 resources_available, resources_min, resources_defaults, resources_max 등으로 서버의 자원 속성에서 cost 가 빠진 것 외에는 비슷합니다. 자원에 따라 큐가 실행될 최대시간, cpu 점유율과 메모리 크기, 노드 수, 큐를 사용할 수 있는 소프트웨어도 제한할 수 있습니다.

큐에 자원 속성을 지정하면 그것은 다른 개체에 지정한 속성보다 우선합니다. 앞에서 서버에 최대실행시간을 지정했다면 그것은 큐에 최대 시간 속성이 지정되지 않았을 때에만 적용됩니다.

ex) pbsq_day 큐로 실행되는 작업은 node2~node5 에서만 실행되도록 하려면 다음 속성을 설정

```
set queue pbsq_day resource_available.host = node2 node3 node4 node5
```

특정 노드를 지정하지 않고 4 개 노드 이상 쓰지 못하게 하려면 다음과 같이 합니다.

```
set queue pbsq_day resource_max.nodes = 4
```

benchmark 라는 소프트웨어만 실행하는 큐를 따로 만들려면 benchmark 큐를 만든 후 다음과 같이 속성을 정합니다.

```
[root@node001 ~]# qstat
Job id Name User Time Use S Queue
------------------------- ---------------- ----------------------
```

```
set queue benchmark resource_available.software = {디렉토리경로}/benchmark
```

기타 :

-작업단위로 속성 부여하기 : $man pbs_job_attributes

-작업에 대한 속성 : qsub -W

-큐, 서버, 작업에 대한 속성은 이 순서대로 우선함. 작업을 작업 큐에 넣을 때 작업에 대해 설정한 속성이 작업이 속한 큐의 속성과 충돌하면 설정한 작성 속성은 적용되지 않음.

```
[kimjw@node001 ~]# qstat
Job id Name User Time Use S Queue
------------------------- ---------------- ---------------------
```

작업 상태

E(exting), H(held), Q(queued), R(running), S(suspend), W(waiting)

qstat 명령 옵션

-Q,q	어떤 큐를 사용한 작업이 대기하거나 실행되고 있는지 확인
-f	작업 큐에 있는 작업의 속서을 자세하게 보여줌.

작업을 쌓아 놓는 명령은 qsub 입니다. 이 명령은 root 로 사용할 수 없음. 쉘에서 여러 작업을 한번에 실행시키려면 스크립트 파일을 만든 후 sh script.sh 와 같이 쉘이 스크립트 파일을 읽어 실행하게 하는 것과 비슷하게, qsub 는 미리 만들어진 작업 스크립트 파일을 받아 그 안에 지시되어 있는 대로 작업을 실행합니다.

PBS 가 실행하는 작업은 큐에 속해서 그 큐의 속성을 받고 또 작업 속성도 지정되어 실행되기 때문에 스크립트에 job 속성을 정의하는 부분이 추가됩니다.

```
#PBS -S /bin/sh
#PBS -q pbsq_day
##PBS -q pbsq_week
##PBS -q pbsq_month
#PBS -l nodes={노드수}:ppn=
#PBS -N{작업이름}
#PBS -r n
#PBS -e {작업이름}
#PBS -o {작업이름}
#PBS -V
cd $PBS_O_WORKDIR
set 'wc -l $PBS_NODEFILE|cut -d" " "-f 1';NNODE=$1
${MPICH_HOME}/bin/mpirun - machinefile $PBS_NODEFILE -np $NNODE{실행파일 1}
```

-q : 작업이 사용할 큐를 지정

-l : 사용할 노드와 프로세스 수를 지정

(사용할 노드는 PBS 가 빈 노드가 생길때마다 자동으로 정해주므로, 사용자는 사용할 프로세스 수만 정의) 노드가 CPU 를 여러개 갖고 있는 SMP 노드에서는 ppn(processors per node)인자로 프로세스 수를 지정해주고 노드수만 바꾸면 됨.

그 밖의 인자

-e{path/name}	실행중인 프로세스가 오류를 출력하면 오류의 내용을 정해진 파일에 출력
-N{name}	qstat 에 나타날 작업의 이름
-o{path/name}	실행되는 작업이 출력하는 내용을 정해진 파일에 출력
-p{priority}	작업의 우선 순위를 결정함. 기본값은 0 이고 -1024 에서 +1023 까지 정할 수 있음
-r y|n	qrerun 명령으로 실행중에 처음부터 재실행될 수 있을지를 정의
-S {path_list}	스크립트를 실행시킬 쉘을 정의
-v{variable_list}	qsub 변수 외 환경변수를 스크립트 안에서 정의해서 사용
-V	외부에서 선언된 환경변수를 qsub 스크립트에서도 쓸수 있도록 함
-W{작업속성}	qsub 인자에서 설정되지 않는 작업 속성을 정의할 때 사용

-스크립트 안에 qsub 인자는 #PBS 키워드로 선언되며, 이때 앞의 #는 주석 표시가 아니므로 생략하지 말아야함. PBS 키워드에서 주석으로 처리할때는 ##으로 해야함.

PBS 에서 제공하는 환경변수

변수명	내 용
PBS_O_HOST	qsub 명령이 실행된 노드 이름을 나타냄
PBS_O_QUEUE	스크립트에서 사용한 큐의 이름을 나타냄
PBS_O_WORKDIR	qsub 명령이 실행된 디렉토리 경로를 나타냄
PBS_ENVIROMENT	batch 작업이면 PBS_BATCH, interactive 작업이면 PBS_INTERACTIVE 값을 가짐
PBS_JOBID	작업 관리자에서 할당한 작업 ID 번호를 나타냄
PBS_JOBNAME	사용자가 스크립트에서 써 준 작업 이름을 나타냄
PBS_NODEFILE	작업이 실행되는 노드 목록이 있는 임시 파일 이름을 나타냄
PBS_QUEUE	작업이 실행될 때 사용하는 큐의 이름을 나타냄. 스크립트에 전달 큐를 사용했으면 PBS_O_QUEUE 에는 전달 큐 이름이, PBS_QUEUE 에는 작업을 넘겨받은 실행 큐 이름이 저장

mpirun 실행 명령은 쉘 실행과 같음. 다음과 같이 여러 명령을 순차적으로 작업 스크립트 하나로 한번에 실행할 수 있음.

```
set 'wc -l $PBS_NODEFILE'; NNODE=$1
${MPICH_HOME}/bin/mpirun -v -machinefile $PBS_NODEFILE -np $NNODE{실행파일}
${MPICH_HOME}/bin/mpirun -v -machinefile $PBS_NODEFILE -np $NNODE{실행파일}
```

PBS_NODEFILE 변수에는 노드 이름을 나열한 문자열이 아니라 노드 이름이 나열되어 있는 가상 파일의 이름이 있음. -machinefile 을 $PBS_NODEFILE 로 해서 PBS 가 지정한 노드에서 프로그램이 실행되도록 합니다.

-np 에는 프로세스 수를 사용합니다. 여기서는 스크립트에서 PBS 변수의 노드 수와 -np 의 수를 일일이 바꾸기 번거러우므로 노드의 수를 세어 NNODE 변수에 저장하고 -np 가 이 변수 값을 사용하도록 했음. 그러므로 스크립트에서 PBS-l 의 값만 바꾸면 됩니다.

MPICH_HOME 같은 외부에서 선언된 환경 변수를 사용할 때에는 #PBS -V 인자를 먼저 붙여주어야 합니다.

mpirun 예)

```
[root@node001 ~]# cat /usr/local/CMSupporter/script/parallel.sh
#!/bin/csh -f
#PBS -l nodes=4:ppn=2
#PBS -l walltime=04:00:00
#PBS -V
set NCPUS=`wc $PBS_NODEFILE|awk '{print $1}'`
echo '#CPUs is:' $NCPUS
cd $PBS_O_WORKDIR
echo "Current working directory is `pwd`"
echo "Node file: $PBS_NODEFILE :"
cat $PBS_NODEFILE
echo "----------------------"
########################################
setenv LD_LIBRARY_PATH /usr/local/mpich-1.2.7p1/lib
set MPIRUN='/usr/local/mpich-1.2.7p1/bin/mpirun'
#define you command and opnions
```

제 3장 작업실행 및 관리

1.Parallel 프로그램

1) Parallel 프로그램 예제 개요

Parallel 프로그램 예제를 위해 다음과 같은 4X4 행렬과 1X4 행렬의 곱에 대해 병렬처리 및 프로그램 실행을 하도록 하겠습니다.

$$\begin{pmatrix} 1 & 15 & 14 & 4 \\ 12 & 6 & 7 & 9 \\ 8 & 10 & 11 & 5 \\ 13 & 3 & 2 & 16 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \end{pmatrix} = \begin{pmatrix} 90 \\ 80 \\ 80 \\ 90 \end{pmatrix}$$

[Parallel 프로그램 예제]

결과인 1X4 행렬의 값 하나는 4X4 행렬과 1X4 행렬의 숫자들의 곱을 합한 값입니다.

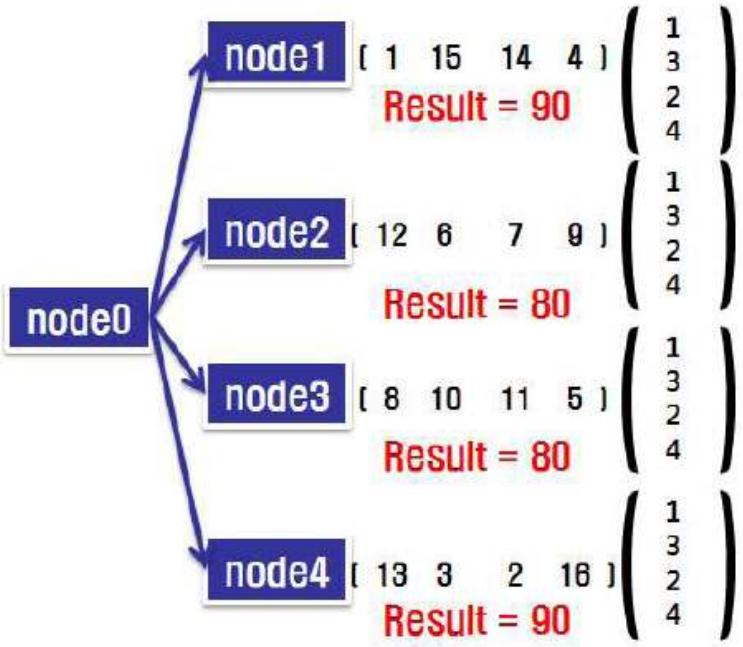
$$\begin{bmatrix} 1 & 15 & 14 & 4 \end{bmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \end{pmatrix} \quad \text{Result} = 90$$

$$\begin{bmatrix} 12 & 6 & 7 & 9 \end{bmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \end{pmatrix} \quad \text{Result} = 80$$

$$\begin{bmatrix} 8 & 10 & 11 & 5 \end{bmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \end{pmatrix} \quad \text{Result} = 80$$

$$\begin{bmatrix} 13 & 3 & 2 & 16 \end{bmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \end{pmatrix} \quad \text{Result} = 90$$

이것을 클러스터에서 계산하기 위해서는 4개 노드를 사용하여 4X4 행렬의 각 행을 4개로 나누어주고, 1X4 행렬은 모든 노드가 알아야 하므로 각 노드에 그대로 분배해 주면 됩니다.



[각 노드별 분배 및 연산 예제]

위 계산을 프로그램 코드로 작성하면 다음과 같습니다.

[주요 코드설명]

◇ MPI_Scatter(&m1,4,MPI_INT,&m1_n,MPI_INT,0,MPI_COMM_WORLD);

- 배열 값을 노드에 나눠주는 함수

MPI_Scatter(나눌변수,변수크기,변수형,보낸 값 받는 수,크기,변수형,노드순번,통신자)

◇MPI_Bcast(&m2,4,MPI_INT,0,MPI_COMM_WORLD);

- 배열 값을 나누지 않고 그대로 전달하는 함수

MPI_Bcast(전달할변수,크기,변수형,루트노드,통신자)

◇MPI_Gather(&m3_n,1MPI_INT,&m3,1,MPI_INT,0,MPI_COMM_WORLD);

- 0 번 노드로 값을 다시 모으는 함수

MPI_Gather(모을변수,크기,변수형,받을변수,변수형,루트노드,통신자)

[Parallel 프로그램 예제]

```
#include <mpi.h>
#include <stdio.h>
int main(int argc,char *argv[])
{
int m1[4][4],m2[4],m1_n[4],m2_n[4],m3[4];
int i,m3_n=0,iproc,nproc;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&nproc);
MPI_Comm_rank(MPI_COMM_WORLD,&iproc);
if(nproc!=4)
{
if(iproc==0)printf("Error! excuted node have to be 4. ₩4");
MPI_Abort(MPI_COMM_WORLD,1);
}
if(iproc==0)
{
m1[0][0]=1;
m1[0][1]=15;
m1[0][2]=14;
m1[0][3]=4;
m1[1][0]=12;
m1[1][1]=6;
m1[1][2]=7;
m1[1][3]=9;
m1[2][0]=8;
m1[2][1]=10;
m1[2][2]=11;
m1[2][3]=5;
m1[3][0]=13;
```

2)프로그램 컴파일

위의 예제 프로그램은 MPI 를 이용하여 작성된 코드이므로 mpicc 명령을 이용하여 컴파일 합니다.

```
$mpicc -o paral parallel.c
```

3)Parallel 프로그램 실행

(1)parallel.sh 파일 수정

parallel.sh 원본 파일의 경로 /usr/local/CMSupporter/script

```
#!/bin/csh -f
#PBS -l nodes=4
#PBS -l walltime=04:00:00
#PBS -V
set NCPUS=`wc $PBS_NODEFILE|awk ' (print $1 5'`
echo '#CPUs is:' $NCPUS
cd $PBS_O_WORKDIR
echo "Current working directory is `pwd`"
echo "Node file: $PBS_NODEFILE :"
cat $PBS_NODEFILE
echo "----------------------"
########################################
setenv LD_LIBRARY_PATH /usr/local/mpich-1.2.7p1/lib
set MPIRUN='/usr/local/mpich-1.2.7p1/bin/mpirun'
#define you command and opnionsv
```

(2)parallel.sh 파일 실행

```
$qsub parallel.sh
```

(3)작업모니터링

구 분	명령어	설 명
큐 구성정보	showq	큐 구성에 대한 정보
작업 모니터링	qstat	사용자 자신의 작업 정보
	옵션 -f	작업 세부정보
	옵션 -Q	작업 큐별 실행 정보

(4)결과확인

```
[kaist@node001 simul]$ cat parallel.sh.<작업번호>
#CPUs is: 4
Current working directory is /home/kaist/simul
Node file: /usr/local/CMSupporter/spool/aux//2733.node001.kaist.ac.kr :
node005
node004
node003
node002
----------------------
( 1 15 14 4)( 1) = (90)
```

(5)작업삭제

```
해당 jobid 를 가지는 작업 삭제
$qdel <jobid>
```

2. Serial 프로그램

1) Serial 프로그램 예제 개요

1~1,000,000 까지 더하는 예제를 통해 Serial 프로그램의 실행 방법과 과정을 알아 보도록 하겠습니다.

2) 예제 프로그램 소스코드

```
#include <stdio.h>
#include <time.h>
int main(int argc,char *argv[]) {
double i,sum_t=0;
for(i=0;i<=1000000;i++)
{
sum_t+=i;
}
```

3) 예제 프로그램 컴파일하기

```
$gcc -o serial serial.c
```

4) 작업 스크립트 작성하기

serial.sh 원본 파일의 경로 /usr/local/CMSupporter/script

□Serial.sh 파일 수정

```
#!/bin/csh -f
# simple TORQUE/PBS job script to run program my_prog
# set default resource requirements for job
# - these can be overridden on the qsub command line
#PBS -l walltime=2:00:00
#PBS -l nodes=1:ppn=8
# export all my environment variables to the job
#PBS -V
# Change to directory from which job was submitted
cd $PBS_O_WORKDIR
echo "Current working directory is `pwd`"
echo "Node file: $PBS_NODEFILE :"
echo "----------------------"
cat $PBS_NODEFILE
   '    "                                "
```

5) serial.sh 파일 실행

```
$qsub serial.sh
```

6) 작업모니터링

구 분	명령어	설 명
큐 구성정보	showq	큐 구성에 대한 정보
작업 모니터링	qstat	사용자 자신의 작업 정보
	옵션 -f	작업 세부정보
	옵션 -Q	작업 큐별 실행 정보

7)작업삭제

해당 jobid 를 가지는 작업 삭제

※jobid 는 qstat 명령을 사용하면 확인 가능

제 4장 작업정책

1.큐 우선순위 및 작업정책

작업시간에 따라 우선순위가 다릅니다. 10 분 이내의 작업이 가장 높은 우선 순위를 가지며, 1 시간>12 시간>48 시간>720 시간>무제한 순으로 우선순위가 조정됩니다.
qsub 를 실행시 기본적으로 실행되는 작업큐는 batch 큐이며, 720 시간(약 30 일) 정도의 walltime(CPU 제한시간)을 가지며, 그 외의 특별한 제약 사항은 없습니다.

-작업큐 표

큐이름	short_10	short_60	normal_12	normal_48	batch	long
큐설명	10 분이내작업	60 분이내 작업	12 시간이내작업	48 시간이내 작업	기본큐 한달(30 일)	무제한 작업
우선순위 (Priority)	500	400	300	200	100	10
CPU 제한시간 (Walltime)	00:10:00	01:00:00	12:00:00	48:00:00	720:00:00	없음
run 제한수 (max_run)	30	30	30	30	없음	없음

```
[root@node001 qmgr]# cat 20090729
create queue defaultq
set queue defaultq queue_type = Route
set queue defaultq route_destinations = batch
set queue defaultq route_destinations += short_10
set queue defaultq route_destinations += short_60
set queue defaultq route_destinations += normal_12
set queue defaultq route_destinations += normal_48
set queue defaultq route_destinations += long
set queue defaultq enabled = True
set queue defaultq started = True
# Create and define queue batch
create queue batch
set queue batch queue_type = Execution
set queue batch Priority = 100
set queue batch resources_default.neednodes = batch
set queue batch enabled = True
set queue batch started = True
# Create and define queue short_10
create queue short_10
set queue short_10 queue_type = Execution
set queue short_10 Priority = 500
set queue short_10 resources_max.walltime = 00:10:00
set queue short_10 enabled = True
set queue short_10 started = True
# Create and define queue short_60
create queue short_60
set queue short_60 queue_type = Execution
set queue short_60 Priority = 400
set queue short_60 resources_max.walltime = 01:00:00
set queue short_60 enabled = True
set queue short_60 started = True
```

```
#
# Create and define queue normal_12
#
create queue normal_12
set queue normal_12 queue_type = Execution
set queue normal_12 Priority = 300
set queue normal_12 resources_max.walltime = 12:00:00
set queue normal_12 enabled = True
set queue normal_12 started = True
#
# Create and define queue normal_48
#
create queue normal_48
set queue normal_48 queue_type = Execution
set queue normal_48 Priority = 200
set queue normal_48 resources_max.walltime = 48:00:00
set queue normal_48 enabled = True
set queue normal_48 started = True
#
# Create and define queue long
#
```

- 작업큐를 사용하기 위한 작업스크립트 변경하기

작업스크립트에 #PBS -q {작업큐명}을 추가해 주시면 됩니다.

```
#!/bin/csh -f
#PBS -l nodes=14:ppn=4
#PBS -l walltime=00:10:00 <-작업큐의 walltime 과 동일한 시간을 넣어주어야 함.
#PBS -p 10
#PBS -V
#PBS -q short_10 <-원하는 큐 이름을 넣습니다.
set NCPUS=`wc $PBS_NODEFILE|awk '{print $1}'`
echo '#CPUs is:' $NCPUS
cd $PBS_O_WORKDIR
echo "Current working directory is `pwd`"
echo "Node file: $PBS_NODEFILE :"
cat $PBS_NODEFILE
```

2.디스크 사용 정책

디스크 사용 제한은 Quota 가 이용되고 있으며, /home 디렉터리의 경우 soft 3G, hard 4G 입니다. 3G 초과된 데이터에 대해서는 7 일간 유예기간이 설정되어 있으니, 7 일 만료전에 필요한 데이터는 백업 받으시기 바랍니다.

대용량의 파일, 작업의 경우는 /data 영역을 (NAS 스토리지) 이용바랍니다.

디렉터리 명	경로	용량제한
홈 디렉터리	/home/계정명	soft : 3G (3G 이후 경고) hard : 4G (4G 이후 사용제한) 유예기간 : 7day (3G 이상에 대해서는 7 일간 유예기간)
작업디렉터리(스크래치)	/data	무제한

2.1)Quota 적용 용량 확인하기

$quota {userid} 명령

```
[root@node001 data]# quota {유저 ID}
Disk quotas for user kaist (uid 513):
Filesystem blocks quota limit grace files quota limit grace
/dev/mapper/VolGroup00-LogVol00
```

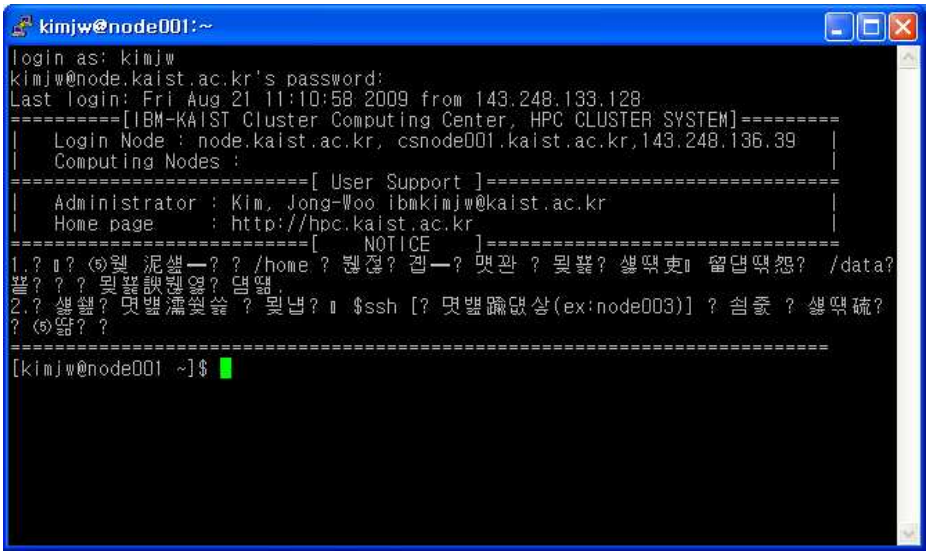
FAQ

1.SSH 접속시 한글이 깨질 경우

◇ PUTTY 사용시

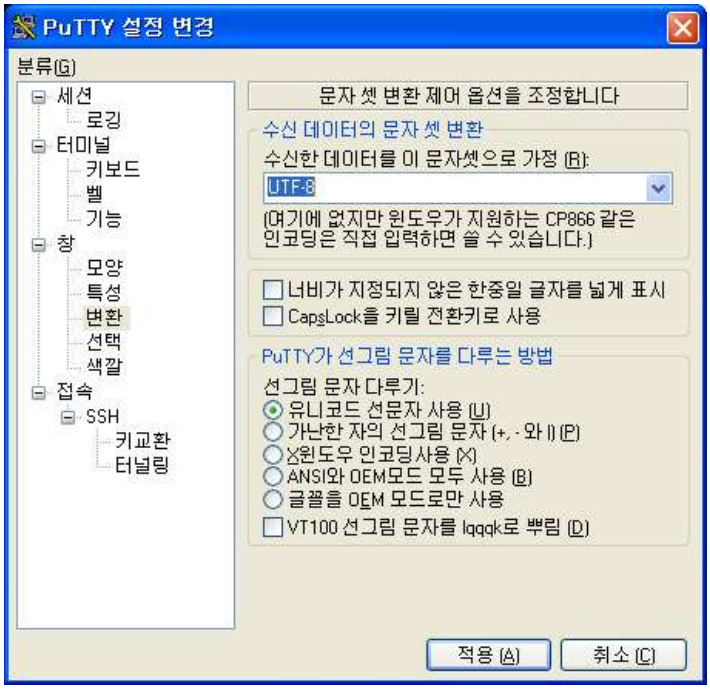
(1)아래와 같이 한글깨짐 현상이 발생할 경우

원인 : 클라이언트와 서버간의 인코딩 불일치 문제



(2)조치방법

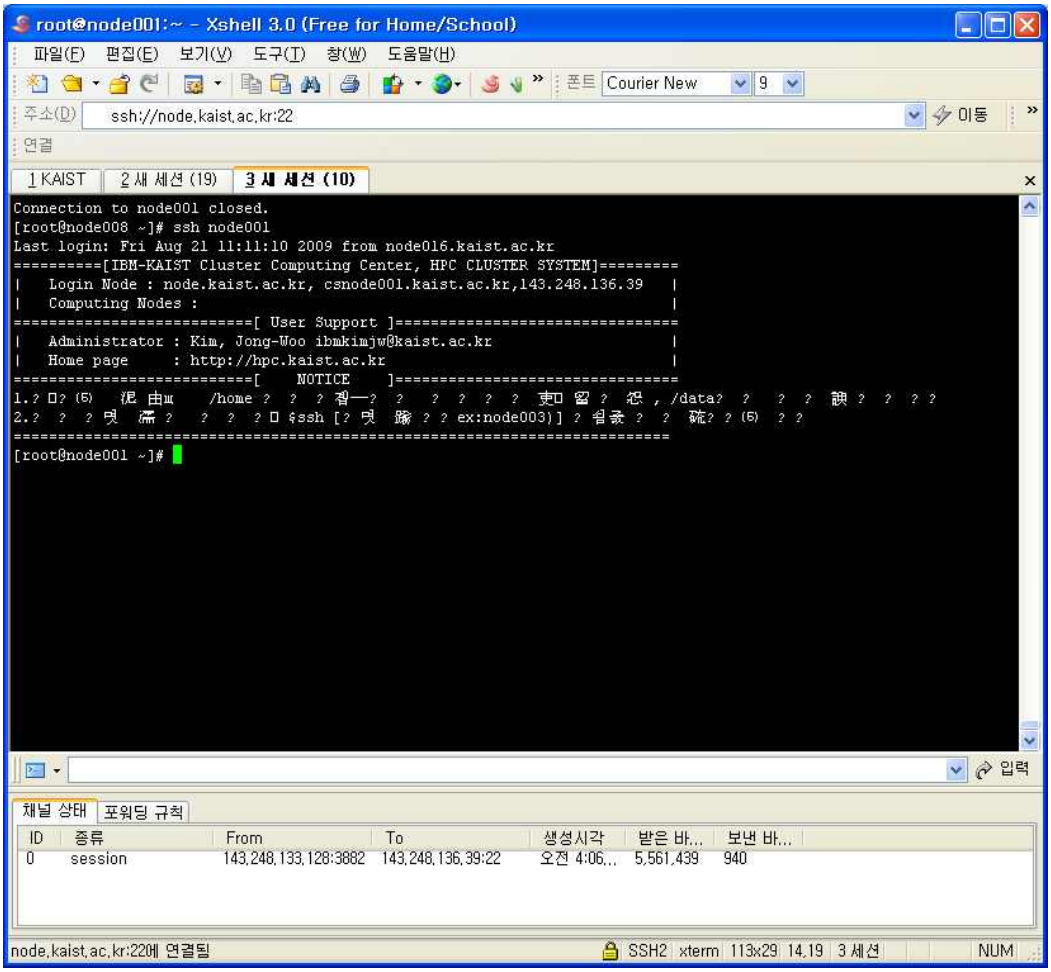
PUTTY 설정 창-변환-수신데이터의 문자 셋 변환에서 UTF-8 로 설정



◇ Xshell 사용시

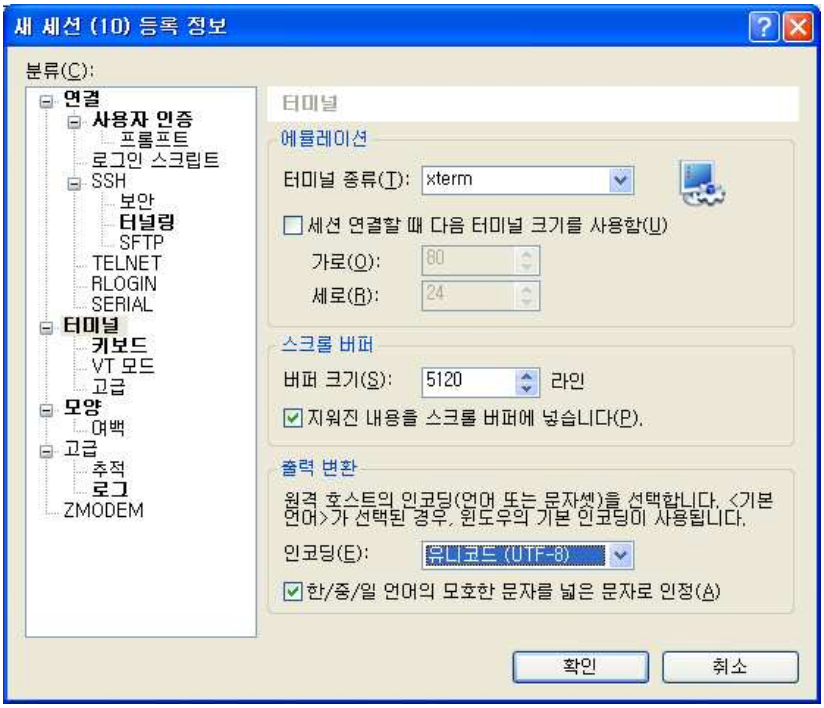
(1)아래와 같이 한글깨짐 현상이 발생할 경우

원인 : 클라이언트와 서버간의 인코딩 불일치 문제



2)조치사항

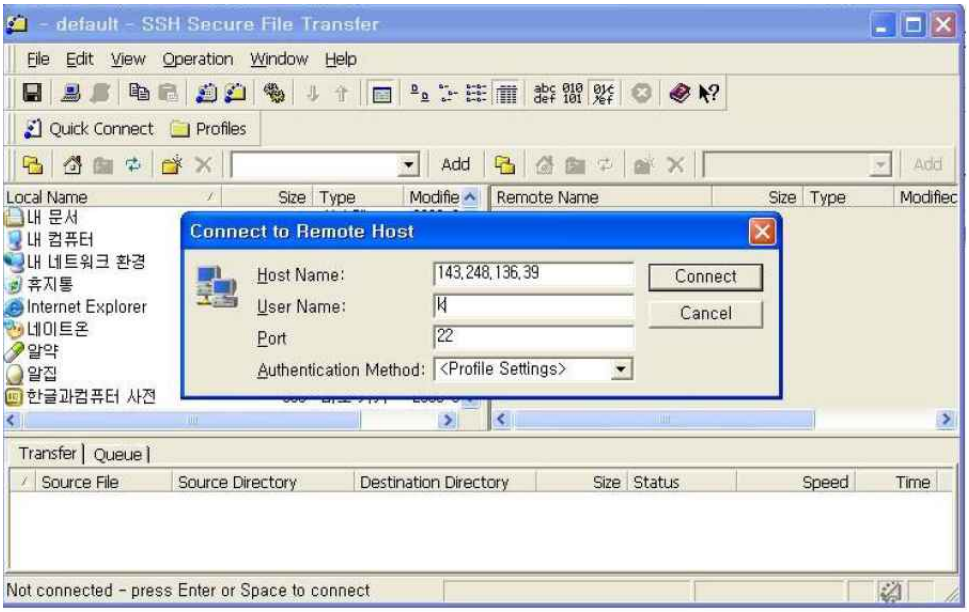
[파일] – [등록정보] – [터미널]에서 출력변환 부분의 인코딩 항목을 UTF-8 로 변경

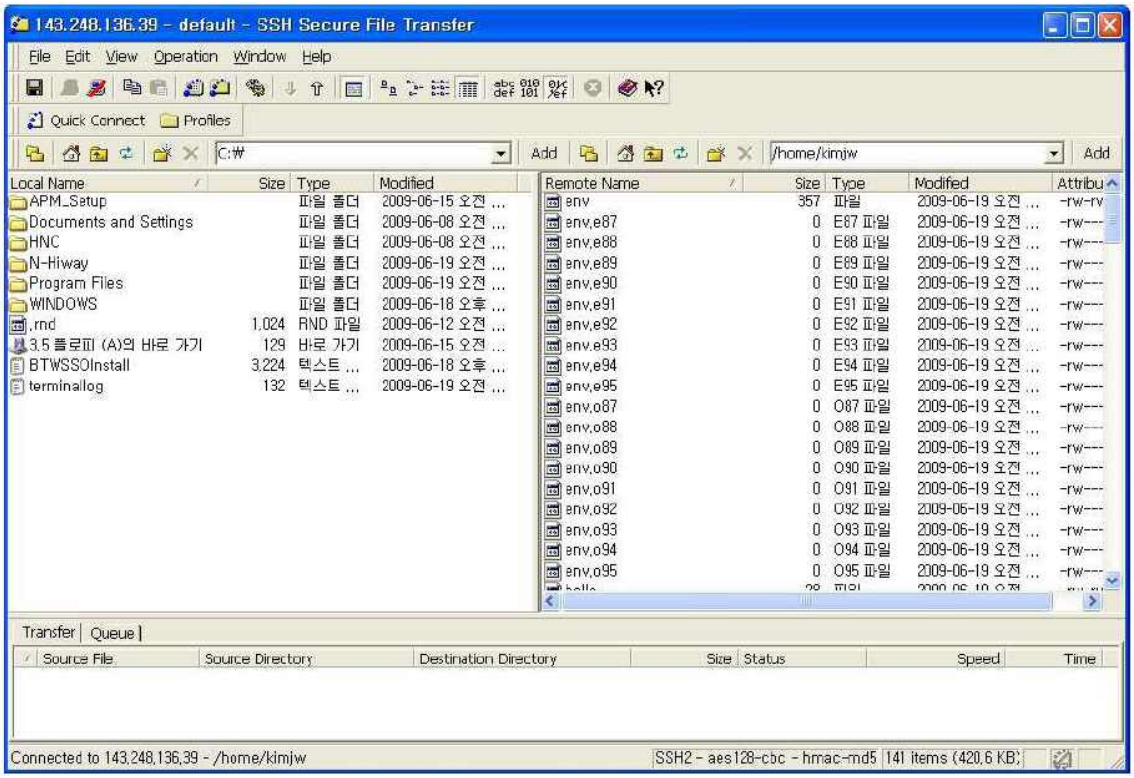


2.서버에 파일을 올릴 경우

SSH Secure File Transfer Client 프로그램을 이용하여 보안 FTP(SFTP) 서비스를 이용하시면 됩니다.

1. SSH Secure File Transfer Client 프로그램 실행
2. Quick Connect 버튼 클릭 – Connect to Remote Host 정보 입력
3. 접속완료


- default - SSH Secure File Transfer
File Edit View Operation Window Help
Quick Connect
Profiles
Add
Local Name
Size
Type
Modifie
Remote Name
Size
Type
Modifiec
내 문서
내 컴퓨터
내 네트워크 환경
휴지통
Internet Explorer
네이트온
알약
알집
한글과컴퓨터 사전
Connect to Remote Host
Host Name:
143.248.136.39
User Name:
k
Port
22
Authentication Method:
<Profile Settings>
Connect
Cancel
Transfer | Queue |
Source File
Source Directory
Destination Directory
Size
Status
Speed
Time
Not connected - press Enter or Space to connect



3.원격에서 X-window 를 사용하고 싶습니다. 어떻게 해야하나요?

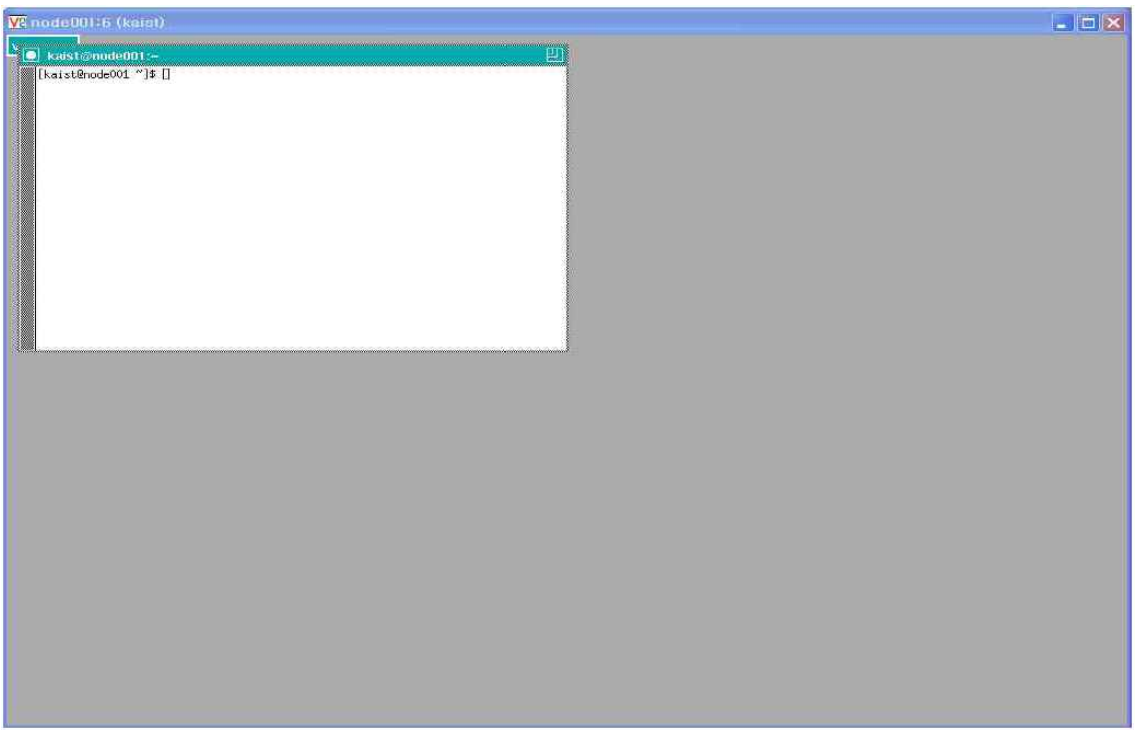
vncserver 를 이용하여 원격에서 GUI 작업을 하실 수 있습니다.

아래와 같이 작업하시면 됩니다 (vncserver 명령 -> PW 를 입력)

```
[kaist@node001 simul]$ vncserver
You will require a password to access your desktops.
Password:
Verify:
xauth: creating new authority file /home/kaist/.Xauthority
New 'node001:6 (kaist)' desktop is node001:6 <- :6 번은 viewer 접속시 입력)
Creating default startup script /home/kaist/.vnc/xstartup
Starting applications specified in /home/kaist/.vnc/xstartup
Log file is /home/kaist/.vnc/node001:6.log
```

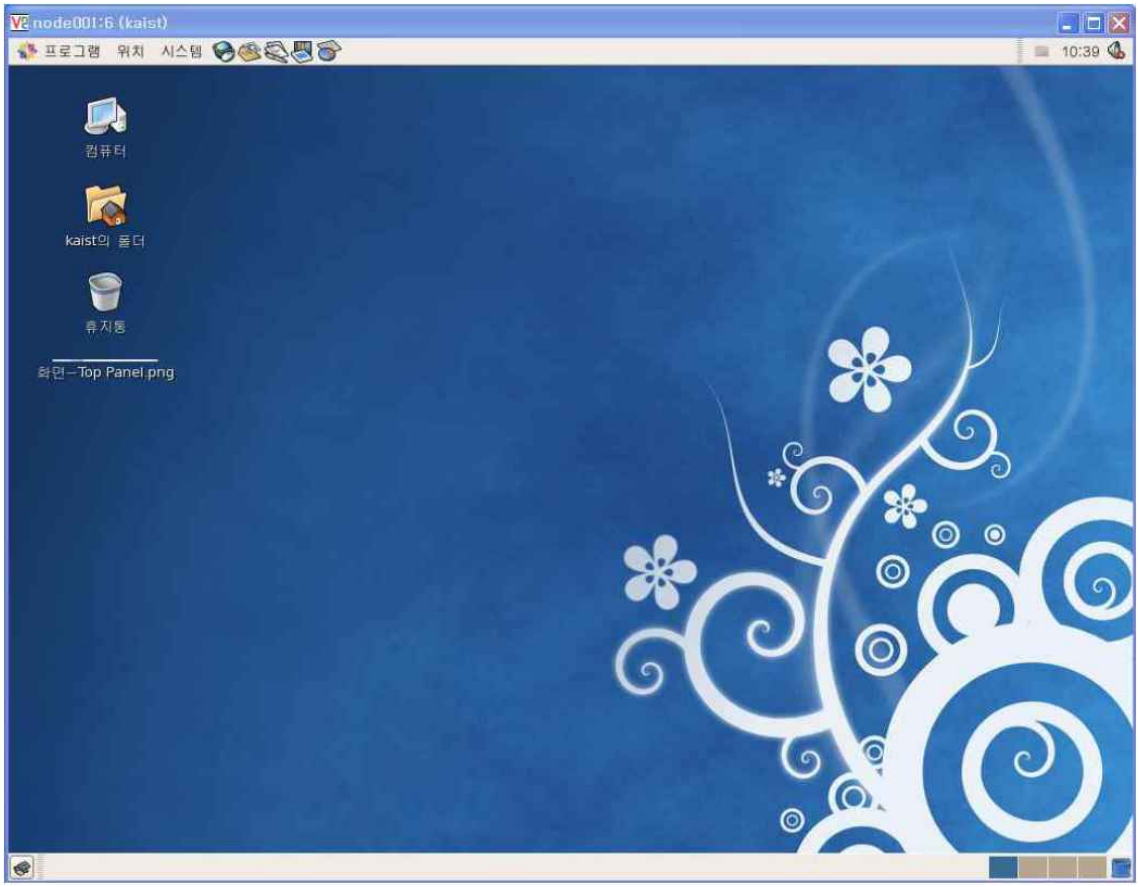
위 작업이 완료된 후 클라이언트에서 VNC Viewer 프로그램을 이용하여 접속하시면 됩니다. (주의 : 143.248.136.39 번 IP 뒤에 해당 포트 번호를 정확히 입력해 주어야 함)

위 작업이 완료되면 기본적으로 TWM(Tab Window Manager) 환경이 실행되도록 되어 있습니다. 이에 실제 사용하는 GUI 와 동일하게 해주기 위해서는 다음과 같이 작업하셔야 합니다.




```
[kaist@node001 .vnc]$cd 사용자홈디렉/.vnc
[kaist@node001 .vnc]$mv xstartup xstartup.bak
- TWM 환경 백업
[kaist@node001 .vnc]$ cp /etc/X11/xinit/xinitrc xstartup
- 현재 설정된 GUI 환경파일 복사
```

기존에 실행되던 vncserver 를 재시작 후에 다시 접속하시면 기존 TWM 환경에서 변경된 것을 확인할 수 있습니다.



4.qsub 로 실행시킨 job 이 어느 노드에서 실행되고 있는지 어떻게 확인하나요?

$ qstat -f 명령을 해보시면 실행노드를 알 수 있습니다.

Job Id: 118193.node001.kaist.ac.kr

Job_Name = script_51.sh

Job_Owner = sbmoon@node001.kaist.ac.kr

job_state = E

queue = batch

server = node001.kaist.ac.kr

Checkpoint = u

ctime = Fri Jul 31 16:41:37 2009

Error_Path = node001.kaist.ac.kr:/data/sbmoon/script_51.sh.e118193

exec_host = node002/2 <- 실행노드/프로세서

5.하위노드 접속시 Permission denided 및 Public key 에러가 발생합니다.

[에러내용]

```
Permission denied, please try again.^M
Permission denied, please try again.^M
Permission denied (publickey,gssapi-with-mic,password).^M
```

ssh-keygen 명령을 이용해 각 노드로 접근시 인증하는 문제를 다시 셋팅해야 합니다.

[각 하위노드로의 SSH 접속시 인증 셋팅]

1 번 노드에서 다음과 같이 작업을 수행합니다.

```
$ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/{사용자디렉터리}/.ssh/id_rsa):
/home/{사용자디렉터리}/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase): 엔터 혹은 값 입력
Enter same passphrase again: 확인값
Your identification has been saved in /home/{사용자디렉터리}.ssh/id_rsa.
Your public key has been saved in /home/{사용자디렉터리}/.ssh/id_rsa.pub.
The key fingerprint is:
```

위와같이 작업 후

/home/{사용자디렉터리}/.ssh 디렉터리로 이동

```
$cp id_rsa.pub authorized_keys
cp: overwrite `authorized_keys'? y
```

작업이 완료되면

하위노드에 정상적으로 접근이 가능합니다.

```
$ssh node002
Last login: Wed Jul 29 11:43:46 2009 from node001.kaist.ac.kr
```