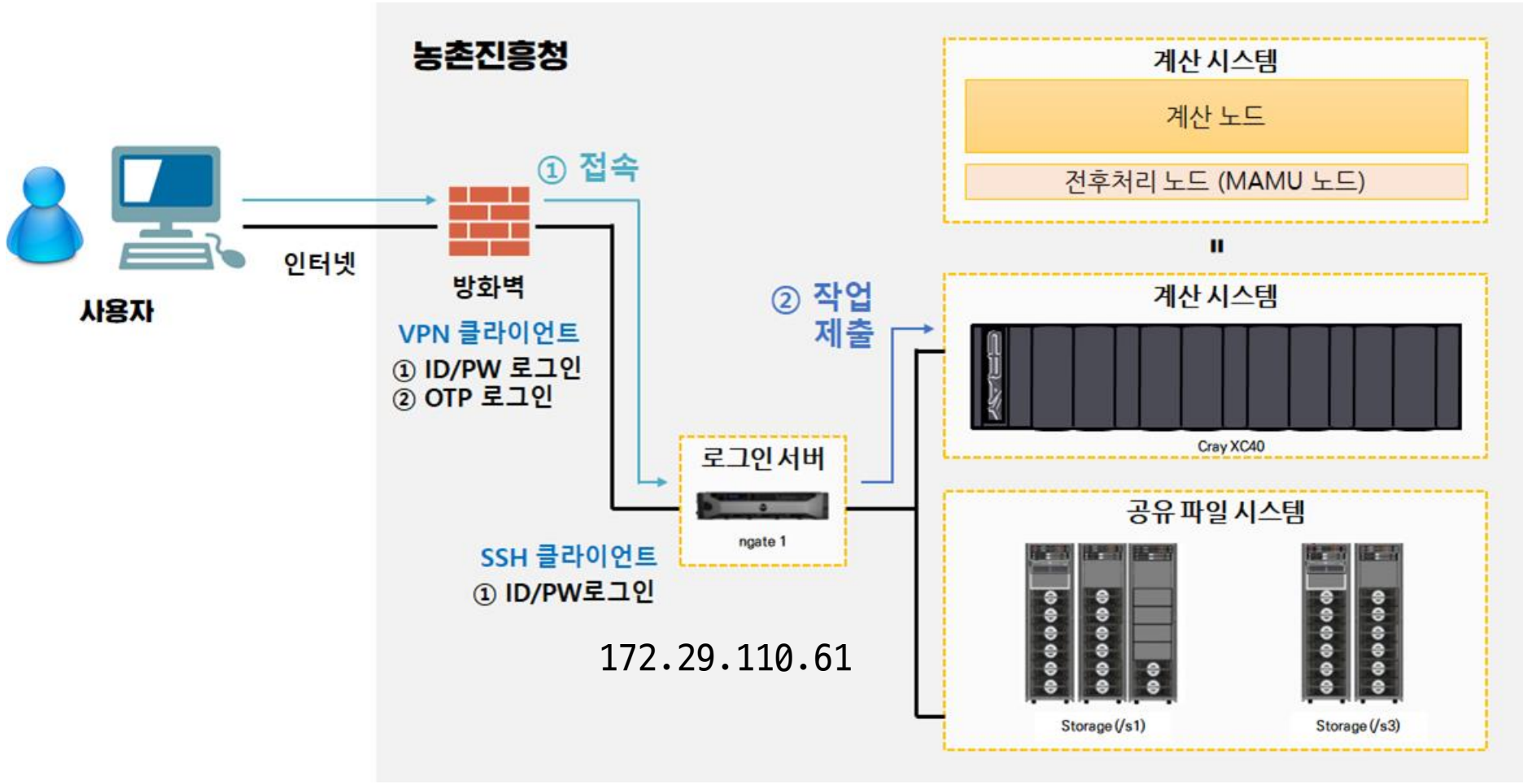
농촌진흥청 초고성능컴퓨터 2호기 기초 사용 방법

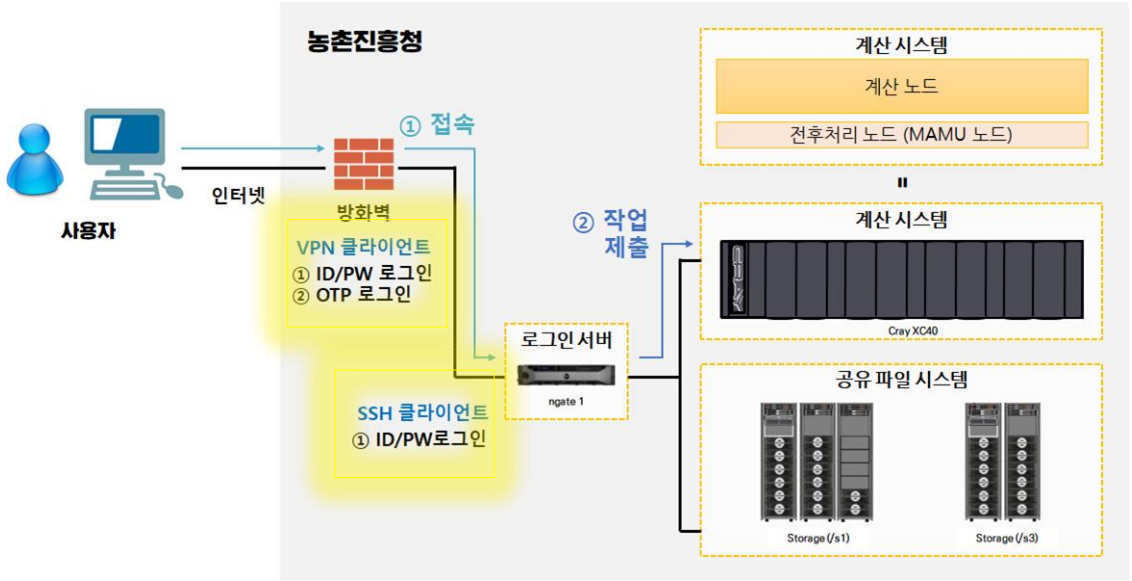
v2. 2025.06.

시스템

개요


농촌진흥청
계산 시스템
계산 노드
전후처리 노드 (MAMU 노드)
계산 시스템
Cray XC40
공유 파일 시스템
Storage (/s1)
Storage (/s3)
① 접속
② 작업 제출
인터넷
방화벽
사용자
VPN 클라이언트
① ID/PW 로그인
② OTP 로그인
로그인 서버
ngate 1
SSH 클라이언트
① ID/PW로그인
172.29.110.61

농촌진흥청 슈퍼컴퓨터

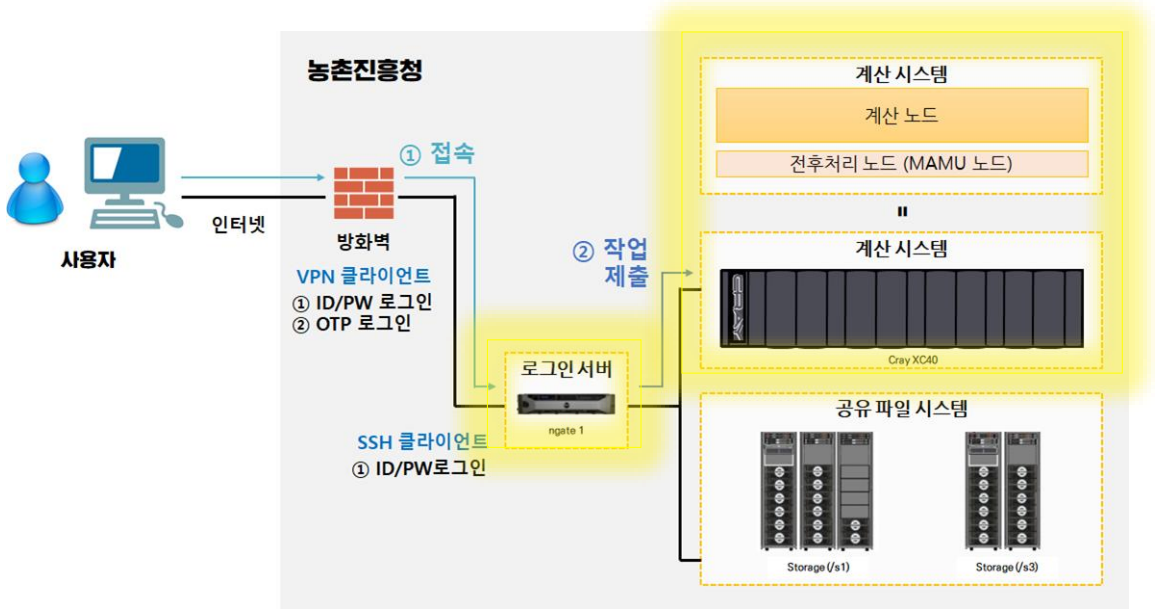


보안이 강화된 2가지 원격접속 방법

- VPN을 이용 (ID/PW + OTP)
  - 평일 9시~18시 접속 가능
- SSH 프로토콜을 이용한 원격 접속(ID/PW)

- 사용자 암호는 비대칭 암호화되어 보관

농촌진흥청 슈퍼컴퓨터

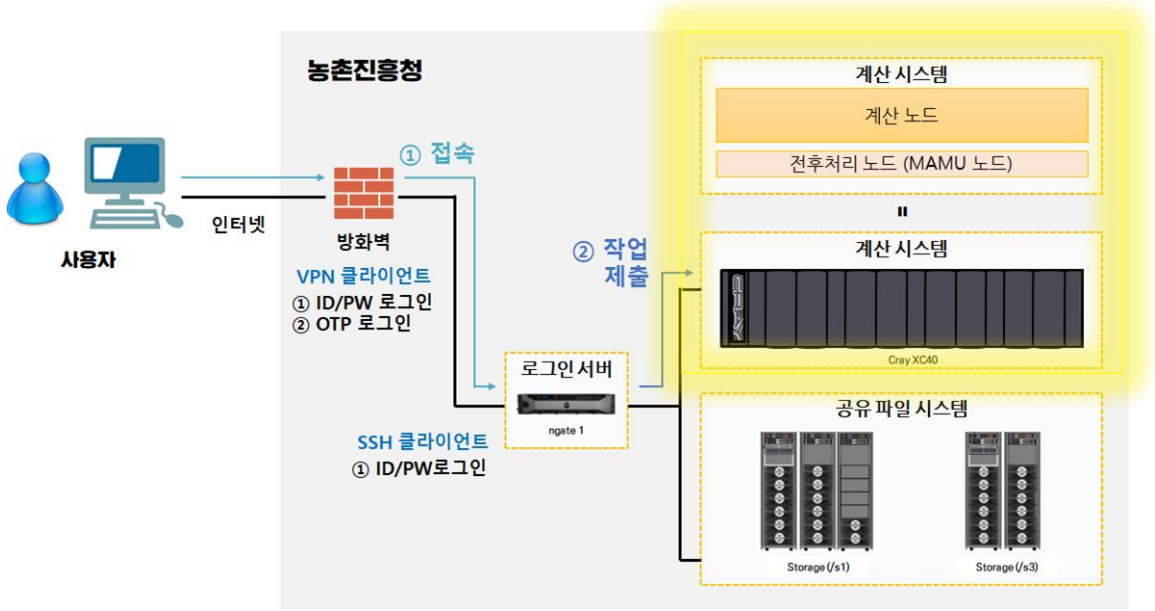


로그인서버

- 사용자가 네트워크를 통해 접속해 사용하는 컴퓨터
- Intel Xeon 2.6GHz 12 cores × 2
- 메모리 : 256GB
- OS : SUSE 리눅스 기반

압축을 푸는 정도의 아주 간단한 작업만 로그인서버에서 가능!

농촌진흥청 슈퍼컴퓨터



계산시스템

- 실제로 작업이 돌아가는 곳
- Intel Haswell 2.6GHz 12 cores, 노드 당 2 소켓

  = 노드 당 24core

  * 슈퍼컴 속도에 영향을 끼치는 원인은 다양하여, 데스크탑과 단순한 속도 비교는 힘듬

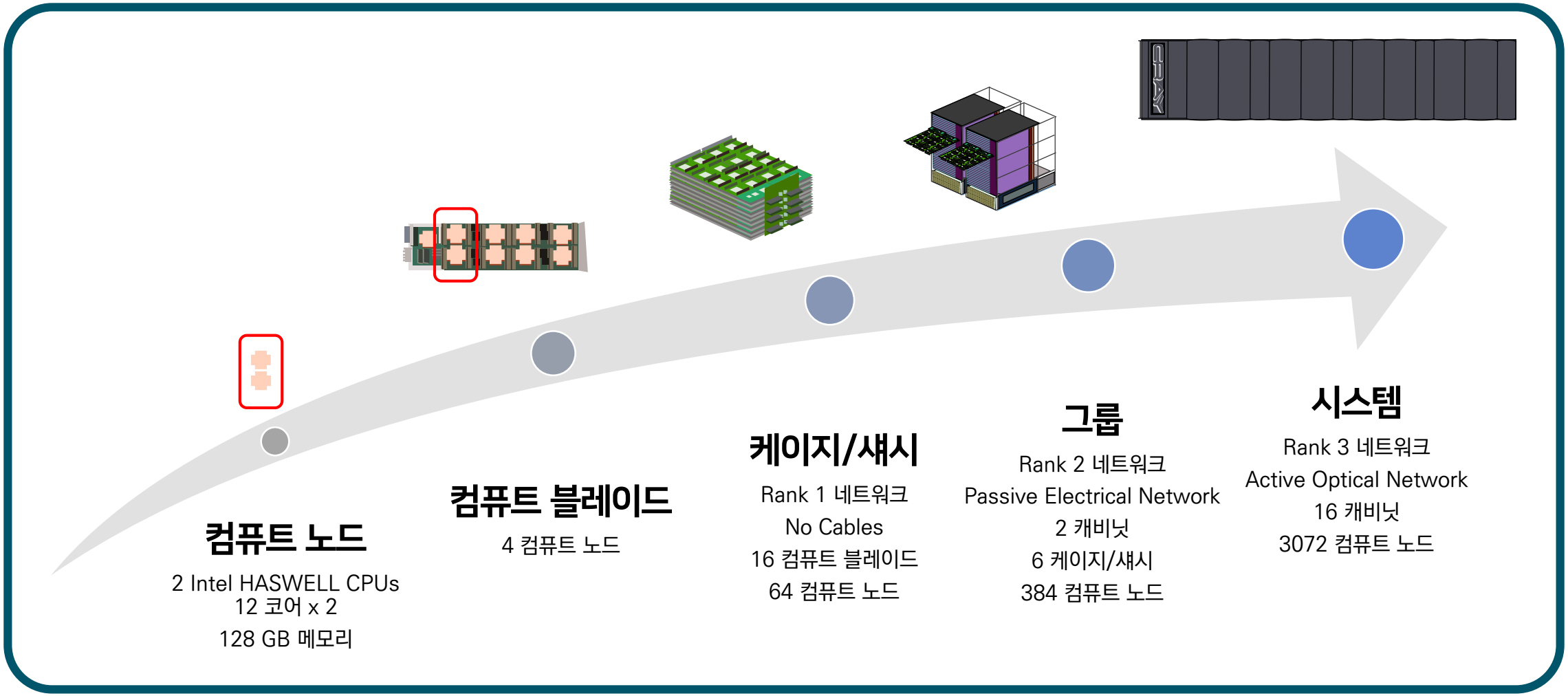
- 노드 당 128GB 메모리
- 2,904개의 계산노드

  * TIP. 2,904대의 컴퓨터가 있다고 볼 수 있음

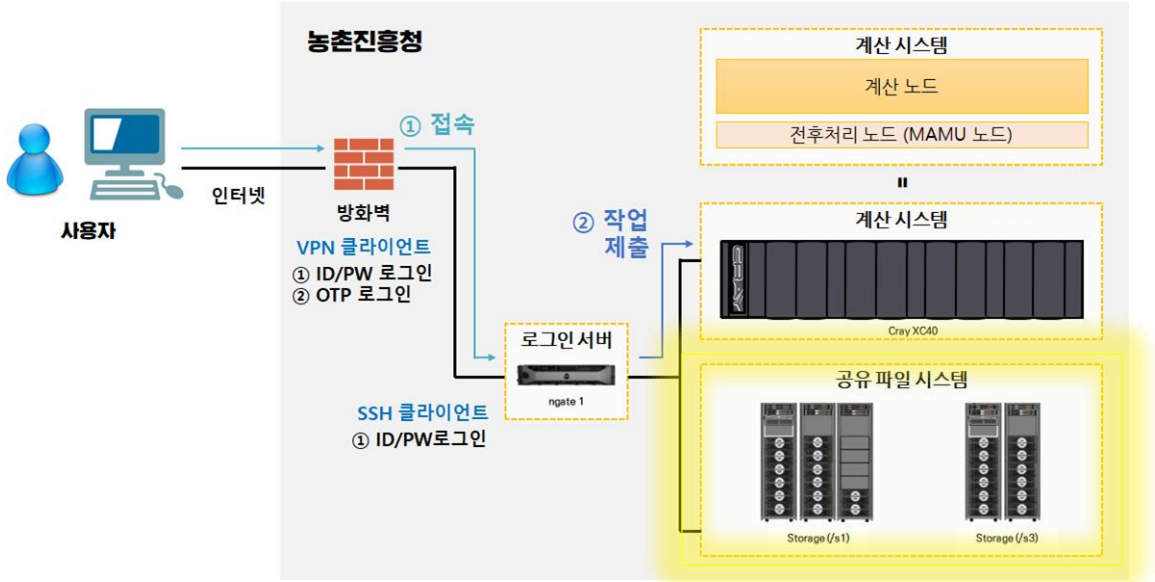
농촌진흥청 슈퍼컴퓨터

계산시스템

- Cray XC40



농촌진흥청 슈퍼컴퓨터

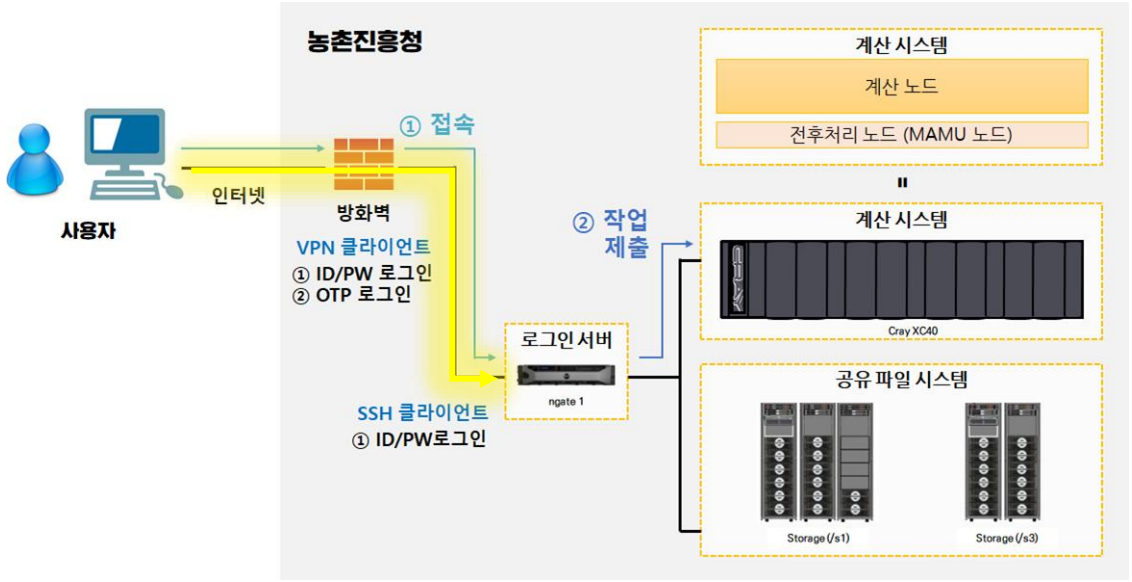


스토리지

- 러스터(Lustre) 파일 시스템 사용
- /s1과 /s3가 있음
  - 사용자 home 디렉토리: /s1/home/<rda or exRDA>/<account>
- 계정별 저장 용량 한계, 파일 보관기간은 향후 안내 예정

- 현재 작업 제출용 scratch 디렉토리 설정 없음. Home 디렉토리에서 작업 제출

농촌진흥청 슈퍼컴퓨터



파일 업로드/다운로드

- 현재 자료전송서버 없음
- 파일 업로드, 다운로드 방법 : SFTP 사용
  * 파일질라 사용 불가, WinSCP 사용 가능, MobaXterm 사용 가능
  * 한 디렉토리에 너무 많은 파일/디렉토리가 담기지 않도록 조절 필요

- 대용량 파일 업/다운로드: 농촌진흥청 슈퍼컴퓨팅센터로 SATA 하드 또는 외장하드 배송

- 인터넷을 통한 파일 다운로드(예: NCBI SRA) : 보안 이슈로 슈퍼컴센터에서 다운로드 진행
  * 신청방법 : 파일 위치, 종류, 예상 크기, 다운로드 방법을 적어서 제출

작업환경 설정
(모듈, 싱귤러리티)

모듈 사용 방법

- **모듈** : 환경변수를 쉽게 변경 가능
- 모듈을 사용하는 이유 : 컴파일러, 라이브러리, 유틸리티 등의 종류, 버전의 변경을 용이하게 함.
- 모듈 명령어와 모듈 파일로 구성됨

- 프로그래밍 환경 모듈 (Programming Environment Modules, PrgEnv-vendor/vers)
  - 컴파일러, 라이브러리, 그 외 프로그래밍 환경에 필요한 여러 구성 요소들에 대한 모듈의 집합
  - 각 구성 요소에 대한 모듈을 각각 로드할 수 있지만, "PrgEnv-" 모듈을 로드해서 사용하는 것을 권장함
  - 예) gnu 컴파일러 사용 : module load PrgEnv-gnu
  - Cray, gnu 컴파일러가 있음

- 대표적인 명령어
  - `module avail` : 사용 가능한 모듈 보기. module avail + 접두사를 입력하면 해당 접두사로 시작되는 모듈만 보임.
  - `module list` : 현대 로드된 모듈 목록
  - `module [ add ¦ load ] [모듈이름]` : 모듈 추가. 뒤에 버전을 안 붙이면 default로 지정된 모듈이 로드됨.
  - `module [ unload ¦ rm ¦ remove ] [모듈이름]` : 사용하지 않을 모듈을 환경에서 제거
  - `module [switch ¦ swap] [모듈이름 1] [모듈이름 2]` : 모듈의 버전을 변경할 때 사용
  - `module purge` : 로드된 모든 모듈파일을 언로드

모듈 사용 방법

- **사용 예시**
  - `java –version` : 자바 버전 확인
  - `module use -a /s3/opt/modulefiles` (TIP. 절대경로로 모듈파일을 로드하면 언로드가 안 됨)
  - `module avail` : 사용 가능한 모듈 확인
  - `module add java/17` : java/17 모듈 로드
  - `java –version`
  - `module swap java/17 java/22` : java/17 모듈을 java22 모듈로 전환
  - `java –version`
  - `module unload java/22` : java/22 모듈을 언로드
  - `java –version`

Singularity 사용 방법

- **싱귤러리티**
  - HPC(고성능 컴퓨팅) 환경에 최적화 된 컨테이너 플랫폼. 도커와 비슷함

- **컨테이너**
  - 애플리케이션 실행에 필요한 환경(라이브러리, 바이너리 등)을 하나로 묶은 독립적 실행 단위
  - 호스트 시스템과 격리되며, 일관된 실행 환경 제공
    - 분석에 사용한 소프트웨어 환경을 그대로 보존하여, 쉽게 해당 환경 재현 가능
  - 컨테이너를 다양한 시스템에 쉽게 배포하고 실행할 수 있음
  - 싱귤러리티 버전이 낮아 (v2.6.1) 도커 이미지를 싱귤러리티 컨테이너로 변환 시 오류가 생길 수 있음
  - 슈퍼컴센터 제공 샌드박스 형태 컨테이너: Ubuntu 24.04 LTS 사용 중

    * 보안 이슈로 사용자가 직접 컨테이너 내 소프트웨어 등 설치 불가

Singularity 사용 방법

- **singularity 사용을 위한 모듈 로드 방법**
  - `module use /s3/opt/modulefiles/`
  - `module load singularity/2.6.1`

  * `module load /s3/opt/modulefiles/singularity/2.6.1` 입력 시 모듈 언로드가 안 됨

- **singularity 컨테이너에 들어가지 않고 컨테이너 내 프로그램을 사용하고 싶은 경우**
  - `singularity [global 옵션] exec [exec 옵션] [컨테이너 이름]`
    - 예) `singularity exec -e [컨테이너 경로] fasterq-dump`
  - TIP. job script 안에 singularity 컨테이너 내 프로그램을 사용하려는 경우 이 방법 사용

- **singularity 컨테이너에 들어가고 싶은 경우**
  - `singularity [global 옵션] shell [shell 옵션] [컨테이너 이름]`
    - 예) `singularity shell [컨테이너 경로]`

- **singularity 컨테이너** : `/s3/simg`에 위치

소프트웨어

소프트웨어, 모듈, 패키지 등

- 소프트웨어/패키지/모듈 설치: 보안 이슈로 슈퍼컴퓨팅센터에서 진행
  - 신청방법 : 프로그램 이름, 버전, 설치방법을 적어서 제출

- 설치된 소프트웨어 사용 방법 : `/s3/program_list/program_list.txt` 참고

- `/s3/opt/bin` 에 설치된 소프트웨어를 쉽게 사용하는 방법
  - `module use /s3/opt/modulefiles`
  - `module load s3_opt/1.0`

작업 제출법 (PBS, ALPS)

PBS, ALPS 사용 방법

큐[Queue]

- 시스템의 구성에 따라 아니면 특정 연구 목적에 따라 구분해서 계산노드를 논리적으로 구분
- 논리적으로 각각 구분한 계산 노드의 그룹: Queue[큐]
- 각각의 큐는 각기 다른 이름을 가짐

농생명 슈퍼컴퓨터 2호기 큐[Queue]

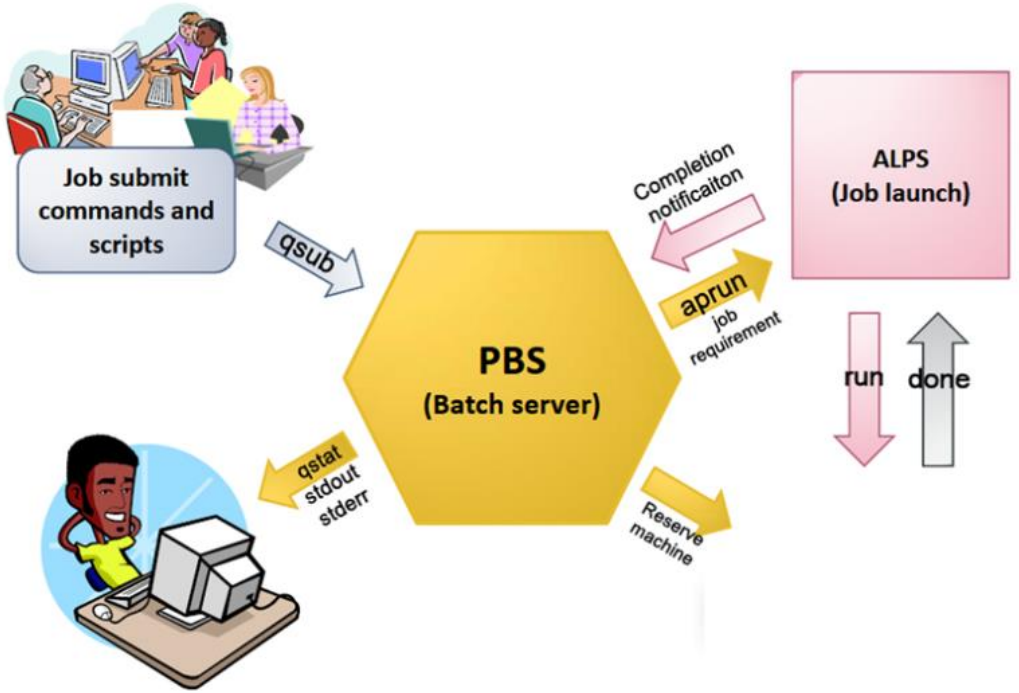
- nabic : 계산노드용, 기본 큐 (사용자가 별도로 지정하지 않으면 nabic으로 작업이 제출됨)
- edu : 계산노드용, 교육용 큐
- nabic_m : 전후처리용

PBS, ALPS 사용 방법

작업 스케쥴러 (Job Scheduler)

- 계산 자원의 상태를 인지하여 순서에 맞게 효율적으로 작업을 할당하는 역할을 수행하는 소프트웨어
- 작업이 제출됨 → 대기 → 수행
- PBS, Slurm 등이 있음. 농진청 슈퍼컴 2호기는 **PBSPro** 사용

  * TIP. 일반적인 리눅스 서버와 다르게, PBS mom이 계산노드에 있지 않고, XC40 내의 서비스노드에 있음(nuri1, nuri2)



PBS, ALPS 사용 방법

ALPS(Application Level Placement Scheduler)

- 작업이 Cray 아키텍처에서 효율적으로 실행될 수 있도록 **계산 노드**에 배치, 실행, 관리하는 Cray 특화 프로그램
  - ALPS는 **계산노드에서 실행되는 응용프로그램에만 적용**되며, 전후처리 노드에서는 제공되지 않음
  - **계산노드에서 작업 시행 시 반드시 aprun을 사용 필요**

- Cray 시스템에서 응용프로그램 스케줄링을 위해 작업관리 스케줄러(PBS)와 연동
  - **PBS** : 사용자 작업 관리, 큐(queue) 및 스케줄링을 결정
  - **ALPS** : PBS에 의해 제출된 작업이 효율적으로 시행되도록 리소스를 배정하고 노드를 관리

TIP. 개념 이해하기

- PBS로 작업을 제출했을 때, 작업스크립트 내 ALPS 실행 명령어를 만나면 작업이 계산노드에서 실행됨

PBS, ALPS 사용 방법

PBS 사용 방법 예시

① 수행하려는 작업 작성 : 컴파일이 필요하면 컴파일!

② 작업 스크립트(job script) 작성 : test_job.sh(예시) 이라는 파일을 만듬

- PBS를 통해 작업을 실행하는지에 대한 내용이 담겨있어야 함. (≒ 작업명세서)
  예) 몇 개의 노드, 몇 개의 cpu 등을 사용해 몇 시간동안 작업을 수행할 것인가
- ①을 ALPS 명령어(aprun)를 사용해 실행하는 내용이 담겨있어야 함

Job script

```
#!/bin/sh
#PBS -N serial_test
#PBS -q normal
#PBS -l select=1:ncpus=1
#PBS –j oe
#PBS -l walltime=00:02:00
cd $PBS_O_WORKDIR
aprun  -n 1 ./serial
```

③ 작업 스크립트 실행 : qsub test_job.sh 이라는 명령어로 작업 스크립트를 실행

PBS 명령어 중 작업 제출에 관한 명령어

PBS, ALPS 사용 방법

- PBS 주요 명령어

PBS	User Commands
qsub	작업 제출
qstat	작업 상태
qdel	작업 취소
qalter	명령줄에서 작업 식별자로 지정된 작업의 속성(날짜, 시간, 리소스, 간격, 작업 이름 등)을 수정
qhold	서버에 작업에 대해 하나 이상의 보류를 요청
qrls	배치 작업에 존재하는 릴리스 보류
qrerun	가능한 경우 지정된 작업을 다시 실행하도록 지시
qsig	일괄 작업 실행을 위한 신호 전송

PBS, ALPS 사용 방법

- qsub : 작업(job) 제출, 예) qsub + 작업스크립트
  - 단일 명령어를 사용해도 되지만, 작업 스크립트 작성 권장
  - **I/O가 많은 작업을 여러 개 던지는 경우 슈퍼컴센터와 상의 후 진행**
  - **최대 사용 가능한 노드 수 : 100노드 (그 이상 사용 필요 시 슈퍼컴센터 문의)**

- qsub 환경변수

Environment variables	Description
PBS_O_HOST	qsub 명령이 실행되는 호스트
PBS_O_QUEUE	작업이 제출된 원래 대기열
PBS_O_SYSTEM	qsub가 실행된 운영 체제(OS) 이름
PBS_O_WORKDIR	**qsub가 실행된 디렉토리의 절대 경로**
PBS_ENVIRONMENT	작업 유형 표시: **PBS_BATCH** or **PBS_INTERACTIVE**
PBS_JOBID	배치 시스템에서 작업에 할당한 작업 식별자
PBS_JOBNAME	사용자가 제공한 작업 이름
PBS_NODEFILE	작업에 할당된 노드 목록이 포함된 파일

PBS, ALPS 사용 방법

작업 스크립트 예시 - serial

Job script	설명
#!/bin/sh	스크립트의 쉘 종류 지정
#PBS -N serial_test	PBS –N 옵션으로 작업명 설정
#PBS -q normal	PBS –q 옵션으로 사용할 큐 설정
#PBS -l select=1:ncpus=1:mpiprocs=1:ompthreads=1 #이 앞에 붙어야 함	PBS –l 옵션으로 사용할 자원 수 설정 (4개 동시 사용 필요) - select=1 : 사용하고자 하는 노드수가 1 - ncpus=1 : 노드 당 사용할 코어 수가 1 (최대 24) - mpiprocs=1 : chunk(노드) 당 사용할 MPI 프로세스 개수가 1 - ompthreads=1 : chunk(노드) 내에서 프로세스 당 사용할 쓰레드 수가 1
#PBS –j oe	Standard error와 standard output이 한 파일에 저장됨.
#PBS -l walltime=00:02:00	PBS –l 옵션으로 walltime 제한 (총 예상 작업시간 설정) 반드시 설정해주세요. (에러 났을 때 유용)
cd $PBS_O_WORKDIR	qsub 가 실행된 디렉토리의 절대 경로로 이동
aprun -n 1 ./serial	aprun으로 실행 파일 serial.e 실행 ☆ 반드시 aprun을 사용해야 함!

PBS, ALPS 사용 방법

사용 예시

- **GATK4를 사용하기 위해서는 버전이 높은 java를 사용하기 위해 모듈을 로드해야 함**
  - `module use /s3/opt/modulefiles/`
  - `module load java/22`

  `* 잘 로드 되었는지 확인: /s3/opt/soft/gatk/gatk SelectVariants --help > SV_help.01.txt 2>&1`

- **로그인서버에서 모듈을 로드해도, job script 안에 모듈을 로드하지 않으면 제대로 작동하지 않음**
  - Job script (jobscript.GATK.help.sh) 만들기
  - PBS를 이용해 작업 제출하기
    - `qsub jobscript.GATK.help.sh`

Job script (jobscript.GATK.help.sh)

```
#!/bin/sh
#PBS -N gatk_help_01
#PBS -l select=1:ncpus=1:mpiprocs=1:ompthreads=1
#PBS -l walltime=00:02:00
#PBS -j oe

cd $PBS_O_WORKDIR

aprun -n 1 /s3/opt/soft/gatk/gatk SelectVariants --help >
gatk_help_01.txt 2>&1
```

PBS, ALPS 사용 방법

사용 예시

- job script 안에서 모듈을 로드하면 제대로 작동 됨
  - Job script (run.jobscript.SV.help.sh) 만들기
  - PBS를 이용해 작업 제출하기
    - qsub jobscript.module.GATK.help.sh
  - **틀린 예** : job script 내에서
    - aprun -n 1 /s3/opt/soft/gatk/gatk SelectVariants --help > gatk_help_02.txt 2>&1
  - **TIP**. 모든 명령어를 별개의 shell script 안에 넣어 사용
    - aprun –n 1 bash script.GATK.sh
    - 이 땐 모듈을 job script 내가 아니라 shell script 안에서 로드해도 작동함

Job script (jobscript.module.GATK.help.sh)

```
#!/bin/sh
#PBS -N gatk_help_02
#PBS -l select=1:ncpus=1:mpiprocs=1:ompthreads=1
#PBS -l walltime=00:02:00
#PBS -j oe

cd $PBS_O_WORKDIR

module use /s3/opt/modulefiles/
module load java/22

aprun -n 1 bash –c ‘/s3/opt/soft/gatk/gatk SelectVariants
--help > gatk_help_02.txt 2>&1’
```

script (script.GATK.sh)

```
s3/opt/soft/gatk/gatk SelectVariants --help >
gatk_help_03.txt 2>&1
```

PBS, ALPS 사용 방법

대표적인 PBS 명령어

- qstat : 작업, 대기열 또는 배치 서버의 상태를 요청하는 데 사용

- qstat 결과 해석

```
Job id            Name             User             Time Use     S Queue
----------------  ---------------- ---------------- --------     - -----
4029705.nuri      작업이름         작업제출계정명   cpu 사용시간 R nabic
```

Job ID	PBS에서 할당한 작업 식별자	**Time Use**	사용된 CPU 시간
Name	제출자가 지정한 작업 이름	**queue**	작업이 있는 대기열
User	작업 소유자	**S**	작업 상태 Q: 대기, 실행 가능 또는 라우팅됨 R: 실행 중 E: 실행 후 종료 → E가 떠있는 상태로 지속 되면 슈퍼컴센터 문의! H: 보류 S: 일시 중단 T: 새 위치로 이동 중 W: 실행 시간에 도달하기를 기다리는 중

PBS, ALPS 사용 방법

대표적인 PBS 명령어

- qstat –a : 지정된 양식의 전체 작업 상태를 자세히 표시

Username	작업 소유자 유저 이름	**TSK**	작업에서 요청한 CPU 수
Jobname	제출자가 지정한 작업 이름	**Req'd Memory**	작업이 요청한 메모리 양
SessID	세션 ID, 작업이 실행 중인 경우에만 나타남	**Req'd Time**	Walltime을 보여줌
NDS	작업에서 요청한 노드 수	**Elap Time**	walltime 또는 cputime을 보여줌

- qstat –u [사용자 리스트] : 사용자 리스트에 있는 사용자의 작업의 상태를 표시

- qstat –B : 배치 서버의 상태 표시

Server	서버 이름	**Hld**	홀딩된 작업 수
Max	서버에서 동시에 실행 가능한 최대 작업 수	**Wat**	실행 대기 중인 각업 수
Tot	현재 서버에서 관리하고 있는 총 작업 수	**Trn**	전환 중인 작업 수
Que	대기열에 있는 작업 수	**Ext**	종료 중인 작업 수
Run	실행 중인 작업 수	**Status**	서버 상태

PBS, ALPS 사용 방법

대표적인 PBS 명령어

- qstat –Q : 큐 상태 표시

Queue	큐 이름	**Run**	실행 중인 작업 수
Max	큐에서 동시에 실행될 수 있는 최대 작업 수	**Hld**	홀드 중인 작업 수
Tot	큐에 있는 총 작업 수	**Wat**	실행 대기 중인 작업 수
Ena	큐의 활성화 여부	**Trn**	Number of jobs being moved
Str	큐의 시작 또는 중지 여부	**Ext**	종료 중인 작업 수
Que	대기열에 있는 적업 수	**Type**	큐 타입 (executing or routing)

PBS, ALPS 사용 방법

대표적인 PBS 명령어

- qdel : 제출한 작업 취소, 예) qdel + 작업아이디(Job id)
  - -W 옵션을 써야 하는 경우, 해당 명령어를 입력하지 말고 슈퍼컴센터에 연락!

그 외 PBS 명령어

- qalter : 명령줄에서 작업 식별자로 지정된 작업의 속성(날짜, 시간, 리소스, 간격, 작업 이름 등)을 수정
- qhold : 서버가 작업에 하나 이상의 보류를 설정하도록 요청
- qrls : 배치 작업에 존재하는 보류를 해제
- qrerun : 가능한 경우 지정된 작업을 다시 실행하도록 지시
- qsig : 실행 중인 배치 작업에 신호를 보냄

PBS, ALPS 사용 방법

작업 스크립트 예시 - serial

Job script	설명
#!/bin/sh	스크립트의 쉘 종류 지정
#PBS -N serial_test #PBS -q normal **#PBS -l select=1:ncpus=1:mpiprocs=1:ompthreads=1** #이 앞에 붙어야 함	PBS –N 옵션으로 작업명 설정 PBS –q 옵션으로 사용할 큐 설정 PBS –l 옵션으로 사용할 자원 수 설정 (4개 동시 사용 필요) - **select=1** : 청크 수.(노드 수랑 같다고 보면 편함) - **ncpus=1** : 노드 당 사용할 코어 수가 1 (최대 24) - **mpiprocs=1** : chunk(노드) 당 사용할 MPI 프로세스 개수가 1 - **ompthreads=1** : chunk(노드) 내에서 프로세스 당 사용할 쓰레드 수가 1
#PBS –j oe	Standard error와 standard output이 한 파일에 저장됨.
#PBS -l walltime=00:02:00	PBS –l 옵션으로 walltime 제한 (총 예상 작업시간 설정) 반드시 설정해주세요. (에러 났을 때 유용)
cd $PBS_O_WORKDIR	qsub 가 실행된 디렉토리의 절대 경로로 이동
aprun -n 1 ./serial	aprun으로 실행 파일 serial.e 실행 ☆ 반드시 aprun을 사용해야 함!

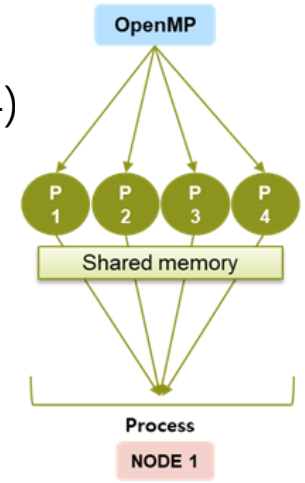
PBS, ALPS 사용 방법

작업 스크립트 예시 (OpenMP)

Job script (OpenMP)

```
#!/bin/sh

#PBS -N openmp_test
#PBS -l select=1:ncpus=12:mpiprocs=1:ompthreads=12
#PBS -l walltime=00:02:00
#PBS -j oe

export OMP_NUM_THREADS=12

cd $PBS_O_WORKDIR

aprun -n 1 -d 12 ./test3_openmp
```

- select=1 : 노드 1개 사용
- mpiprocs = 1 : 노드 당 하나의 프로세스 사용
- ompthreads= 12 : 하나의 프로세스 안에 12개 쓰레드
  → 12개의 코어 사용
- ncpus=12 : 노드 당 cpu 사용 수 12개
  ≤ 노드 당 코어 수 (24)



- -n : select × mpiproc 수와 같음
  - 해당 애플리케이션이 사용한 총 PE (Processing Elements) 수
- -d : 프로세스 당 쓰레드 수. ompthreads 수와 일치

PBS, ALPS 사용 방법

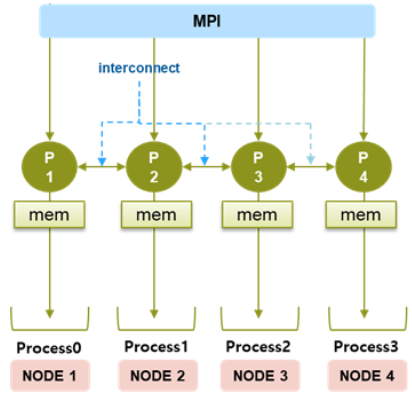
작업 스크립트 예시 (MPI)

Job script (MPI)

```
#!/bin/sh

#PBS -N mpi_test

#PBS -l select=3:ncpus=24:mpiprocs=24:ompthreads=1
#PBS -l walltime=00:02:00

#PBS -j oe
cd $PBS_O_WORKDIR



aprun -n 72 ./mpi
```

- select=3 : 노드 3개 사용
- mpiprocs = 24 : 노드 당 24개의 프로세스 사용
- ompthreads= 1 : 하나의 프로세스 안에 1개 쓰레드
- ncpus=24 : 노드 당 cpu 사용 수가 24개

  ≤ 노드 당 코어 수(24)



- -n : select × mpiproc 수와 같음 (총 72개)

PBS, ALPS 사용 방법

작업 스크립트 예시 (MPI + OpenMP)

Job script (MPI)

```
#!/bin/sh

#PBS -N hybrid_test

#PBS -l select=4:ncpus=24:mpiprocs=6:ompthreads=4
#PBS -l walltime=00:02:00

#PBS -j oe
cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=4



aprun -n 24 -d 4 ./hybrid
```

- select=4 : 노드 4개 사용
- mpiprocs = 6 : 노드 당 6개의 프로세스 사용
- ompthreads= 4 : 하나의 프로세스 안에 4개 쓰레드
- ncpus=24 : 노드 당 cpu 사용 수가 24개

  ≤ 노드 당 코어 수 (24)

- -n : select × mpiproc 수와 같음 (총 24개)
- -d : 프로세스 당 쓰레드 수. ompthreads 수와 일치(4개)

PBS, ALPS 사용 방법

대표적인 ALPS 사용 방법

- aprun : 작업 제출
  - -n PEs : 어플리케이션이 사용하는 총 PE의 수
  - -N pes_per_nod : 계산노드 당 총 PE의 수
    - ex) –N 1 : 계산노드 하나 당 프로세스 하나만 돌아감
  - -d depth : PE 당 쓰레드 숫자

- 예시 : OpenMP 방식이 아닌 다른 공유 메모리 방식을 사용해 멀티 쓰레딩이 가능한 소프트웨어 softwareA 가 있다고 가정
  - run_A.sh 내 softwareA 가 24개의 쓰레드를 사용하는 내용 포함
  - 작업 스크립트 예시

```
#!/usr/bin/sh
#PBS -N softwareA
#PBS -q edu
#PBS -l select=200:ncpus=24              200개의 노드, 노드 당 24 개의 코어 사용
cd $PBS_O_WORKDIR
aprun -n 200 -N 1 -d 24 run_A.sh         200개의 프로세스, 계산노드 당 하나의 프로세스, 계산노드 당 24개의 쓰레드
```