

# 누리온(NURION) 지침서

2024. 07.



한국과학기술정보연구원  
Korea Institute of Science and Technology Information

# 목 차

I . 시스템 .....	1
I -1. 시스템 개요 및 구성 .....	1
I -2. 사용자 환경 .....	9
I -3. 사용자 프로그래밍 환경 .....	16
I -4. 스케줄러(PBS)를 통한 작업 실행 .....	42
I -5. 사용자 지원 .....	57
II . 소프트웨어 .....	58
II -1. ANSYS FLUENT .....	58
II -2. ANSYS CFX .....	64
II -3. Abaqus .....	69
II -4. NASTRAN .....	76
II -5. 가우시안16(Gaussian16) LINDA .....	81
II -6. 가우시안16(Gaussian16) .....	85
III . 부록 .....	89
III -1. 작업 스크립트 주요 키워드 .....	89
III -2. Conda .....	91
III -3. Singularity 컨테이너 .....	98
III -4. Lustre stripe .....	110
III -5. 데이터 아카이빙 .....	112
III -6. MVAPICH2 성능 최적화 옵션 .....	123
III -7. 딥러닝 프레임워크 병렬화 .....	124
III -8. 공통 라이브러리 목록 .....	128
III -9. 데스크톱 가상화(VDI) .....	130
III -10. 버스트버퍼(Burst Buffer) .....	138
III -11. 플랫 노드(Flat node) .....	143
III -12. DTN(데이터전송노드) .....	144

# I . 시스템

## I -1. 시스템 개요 및 구성

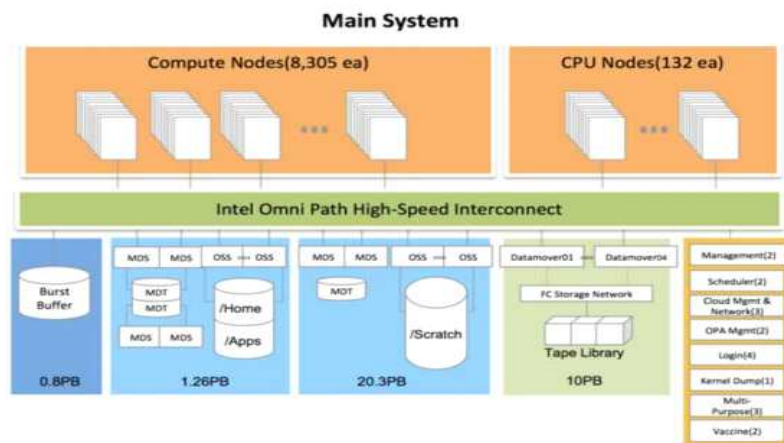
### 가. 개요

KISTI 슈퍼컴퓨터 5호기 누리온(NURIION)은 리눅스 기반의 초병렬 클러스터 시스템으로 이론최고성능(Rpeak) 25.7PFlops(4호기 Tachyon2의 약 70배)의 연산처리 성능을 갖고 있다.

누리온은 1소켓(CPU)당 성능이 3TFlops(4호기 Tachyon2 전체 성능의 약 1/100)인 매니코어 CPU(68개의 코어로 구성) 8,305개로 구성되어 수십만 코어를 동시에 활용할 수 있는 클러스터 방식의 초고성능 컴퓨팅 시스템이다. 그리고 고성능 인터커넥트(Interconnect), 대용량 스토리지, 버스트 버퍼(Burst Buffer)가 탑재되어 있다. 버스트 버퍼는 응용 프로그램에서 순간적으로 발생하는 대규모 I/O 요청을 원활히 처리할 수 있다.

누리온은 이러한 대규모 계산 능력을 통해 국가 연구개발(R&D)의 경쟁력 향상 기여에 목적을 두며, 기후 예측이나 신소재 개발 등 복잡한 문제나 신약 개발, 항공 실험 등 실험적으로만 해결하기에는 위험하고 비용이 많이 소요되는 문제를 해결하는 데 기여하고자 한다. 누리온은 이러한 과학기술 문제를 해결하기 위해 기획 단계부터 다양한 분야의 난제를 발굴하는 노력을 병행하고 있다.

또한 누리온은 이러한 연구 개발들을 지원하기 위한 다양한 종류의 소프트웨어를 탑재하고 지속적으로 업그레이드하고 있으며, 고성능 하드웨어와 유기적인 연동을 통해 최고의 생산성을 유지하기 위해 노력하고 있다. 아울러 최적·병렬화를 비롯한 다양한 사용자 지원과 거대 문제 및 난제 해결 등을 위한 대형 과제 발굴을 통해서 국가 과학기술 발전과 지능정보사회 구현의 핵심 인프라로서 역할을 수행하고자 한다.



[누리온 구성도]

구분		KNL 계산노드	Cpu-only 노드
제조사 및 모델		Cray CS500	
노드 수		8,305	132
성능최고성능(Rpeak)		25.3 PFlops	0.4 PFlops
프로세서	모델	Intel Xeon Phi 7250 (KNL)	Intel Xeon 6148 (Skylake)
	CPU당 이론성능	3.0464 TFLOPS	1.536 TFLOPS
	CPU당 코어 수	68	20
	노드당 CPU 수	1	2
	On-package Memory	16GB, 490GB/s	-
메인메모리	모델	16GB DDR4-2400	16GB DDR4-2666
	구성	16GB x6, 6Ch per CPU	16GB x12, 6Ch per CPU
	노드당 메모리(GB)	96GB	192GB
	CPU당 대역폭	115.2GB/s	128GB/s
	전체 크기	778.6TB	24.8TB
고성능 인터커넥트	토폴로지	Fat Tree	
	Blocking Ratio	50% Blocking	
	Switches 구성	274x 48-port OPA edge switches 8x 768-port OPA core switches	
	포트당 대역폭	100Gbps	
고성능 파일시스템 (Burst Buffer)	서버 수	DDN IME240 Server 48ea	
	서버당 디스크	16x 1.2TB NVMe SSD, 2x 0.45TB NVMe SSD	
	전체 가용 용량	0.8PB	
	대역폭	20GB/sec per server, 0.8TB/s total	
병렬파일시스템	파일 시스템	Lustre 2.7.21.3	
	전체 가용 용량	/scratch: 21PB, /home: 0.76PB, /apps: 0.5PB	
	대역폭	0.3TB/s	
	RAID 구성	RAID6(8D+2P)	

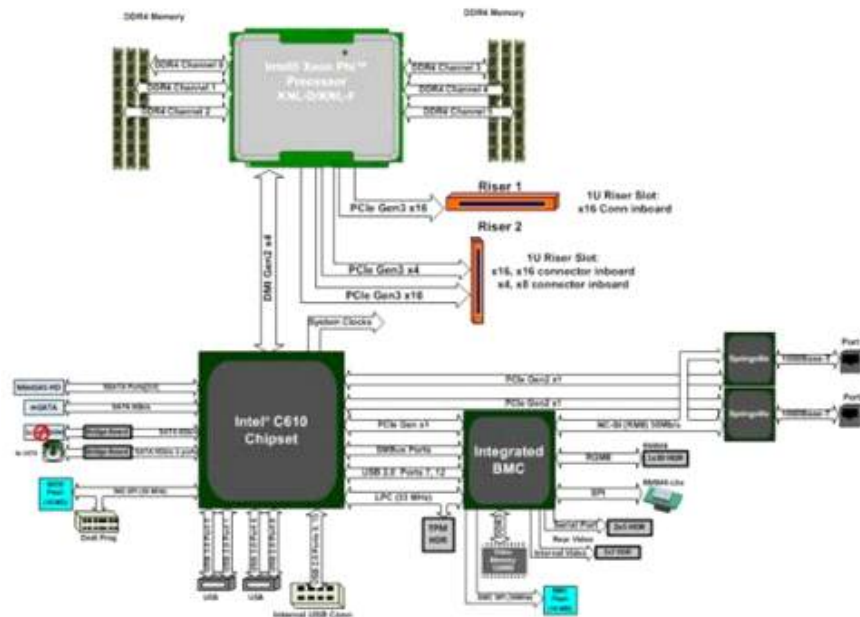
[누리온 시스템 사양 및 구성]

## 나. 계산 노드

누리온 시스템의 계산노드는 총 8,437개로 매니코어 기반의 Intel Xeon Phi 7250프로세서가 장착된 8,305개 계산노드와 Intel Xeon Gold 6148 프로세서가 장착된 132개 계산노드로 구성되어 있다.

### 1. KNL(매니코어) 노드

KNL 노드에 장착된 Intel Xeon Phi 7250 프로세서(코드명 Knights Landing)는 1.4GHz 기본 주파수에 68개 코어(hyperthreading off)로 동작한다. L2 캐시 메모리는 34MB이며, 온패키지 메모리인 MCDRAM (Multi-Channel DRAM)은 16GB(대역폭 490GB/s)이다. 노드 당 메모리는 96GB로 16GB DDR4-2400 메모리가 6채널로 구성되어 있다. 2U크기의 인클로저(enclosure)에는 4개의 계산노드가 장착되며, 42U 표준랙에 72개의 계산노드로 구성되어 있다.



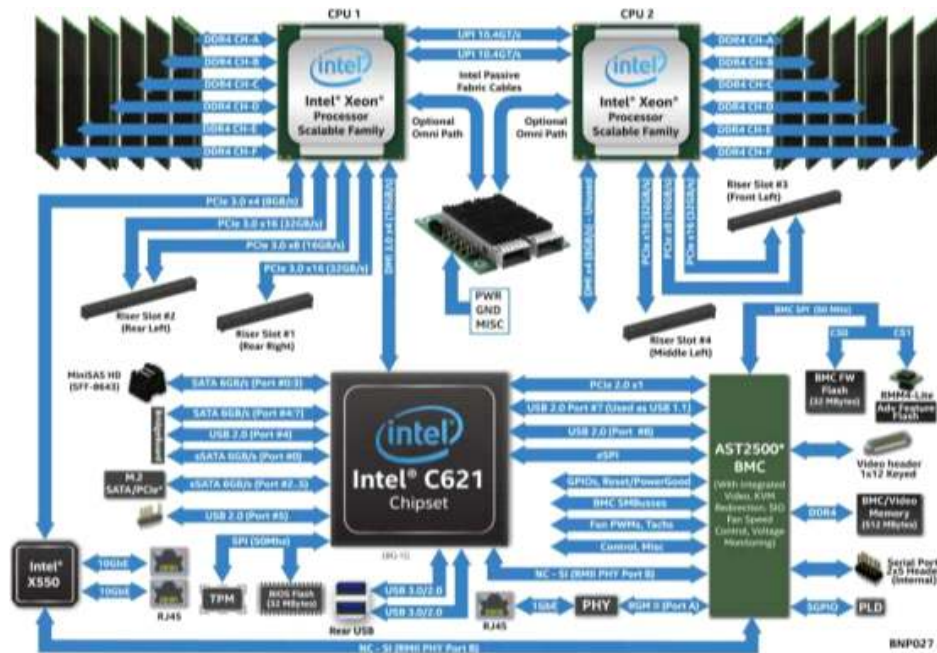
[KNL 기반 계산노드 블록 다이어그램]



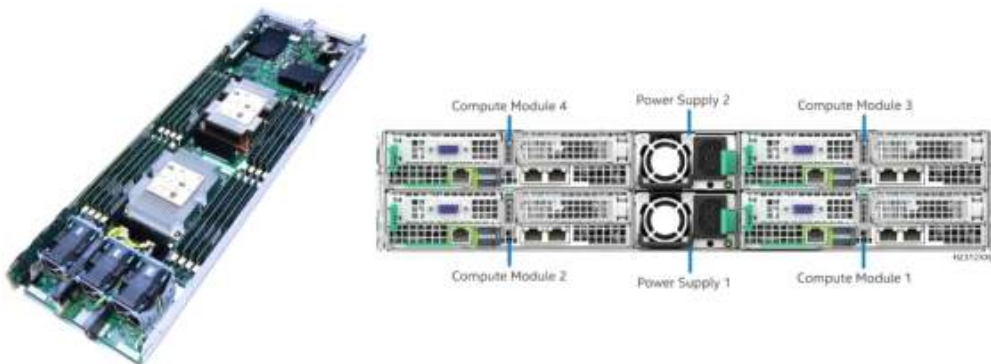
[KNL 기반 계산노드]

## 2. SKL(CPU-only) 노드

SKL(CPU-only) 노드에는 2개의 Intel Xeon Gold 6148 프로세서(코드명 Skylake)가 장착되어 있다. 기본 주파수는 2.4GHz이며 20개의 CPU 코어(hyperthreading off)로 구성된다. L3 캐시 메모리는 27.5MB이며, 각 CPU당 메모리는 96GB(노드당 192GB)로 16GB DDR4-2666 메모리가 6채널로 구성되어 있다. 2U 크기의 인클로저(enclosure)에는 4개의 계산노드가 장착되어 있다.



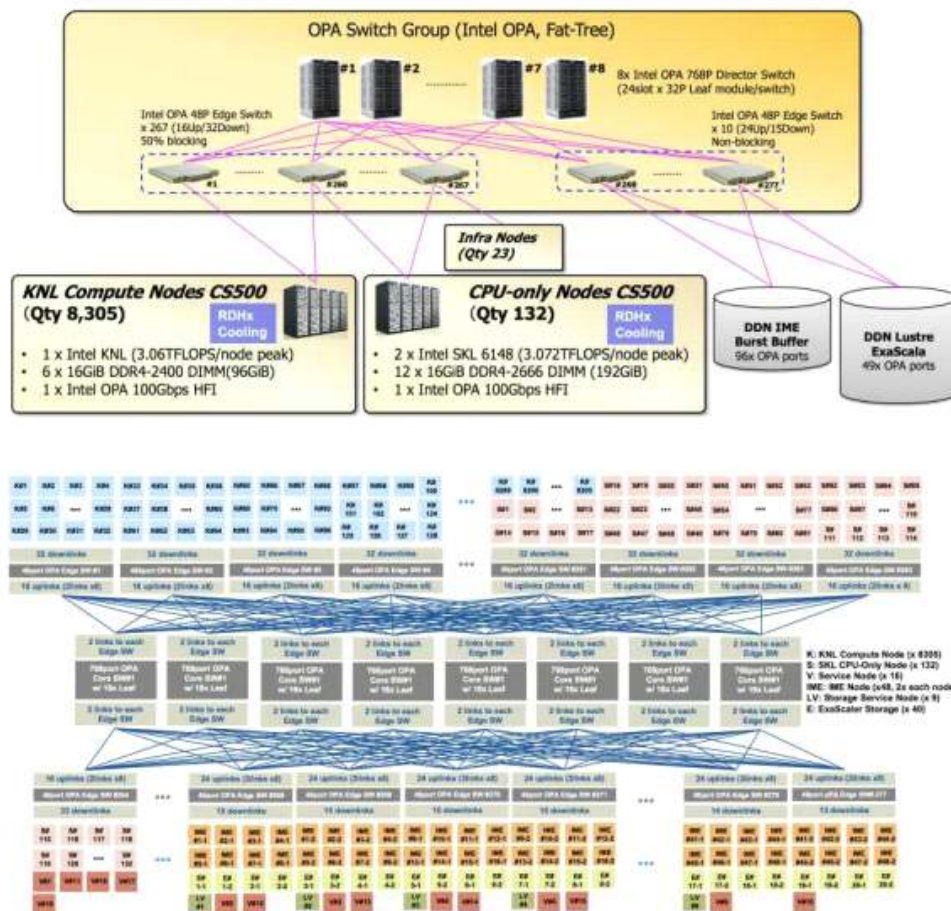
[SKL 기반 CPU-only 노드 블록 다이어그램]



[SKL 기반 CPU-only 노드]

## 다. 인터커넥트 네트워크

노드 간 계산 네트워크 및 파일 I/O 통신을 위한 인터커넥트 네트워크로 인텔 OPA (Omni-Path Architecture)를 사용하고 있다. 100Gbps OPA를 사용하여 Fat-Tree 구조로 계산노드 간 50% blocking, 스토리지 간 non-blocking 네트워크로 구성하였다. 계산노드와 스토리지는 277개의 48포트 OPA Edge 스위치에 연결되며 모든 Edge 스위치는 8대의 768포트 OPA Director 스위치에 연결하였다.



[인터커넥트 네트워크 구성도]

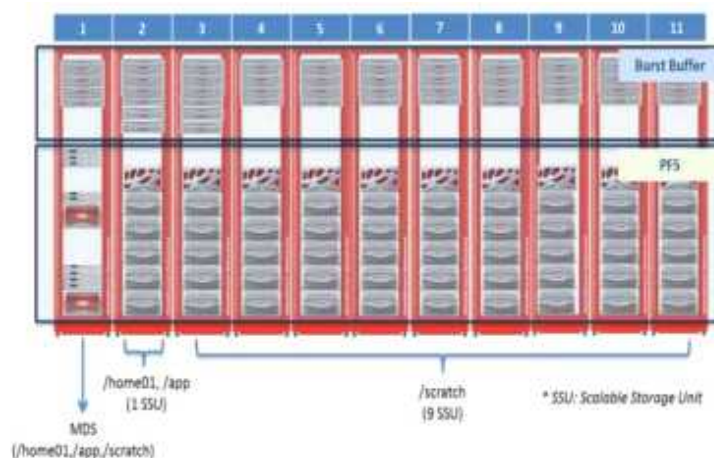


## 라. 스토리지

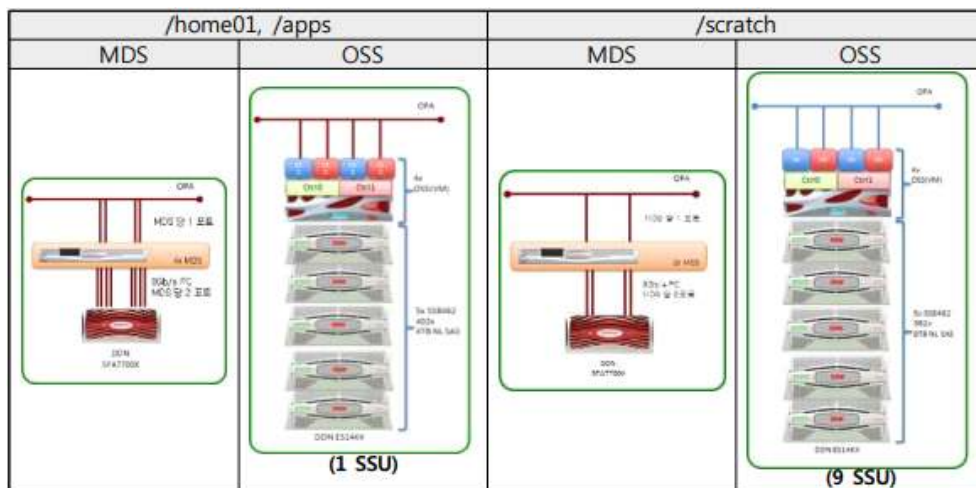
### 1. 병렬파일시스템 및 버스트버퍼 구성

Nurion은 IO 처리 및 데이터 보관을 위해 병렬파일시스템과 버스트버퍼(Burst Buffer)를 제공한다. 병렬파일시스템을 위해 스크래치(/scratch 21PB), 홈(/home01, 760TB), 어플리케이션(/app, 507TB) 세 개의 Lustre 파일시스템을 제공하고 있다.

각 파일시스템은 SFA7700X 스토리지 기반의 메타데이터 서버(MDS)와, ES14KX 스토리지 기반의 오브젝트 스토리지 서버(OSS)로 구성된다. 버스트버퍼는 계산 노드와 병렬파일시스템 사이에서 동작하는 NVMe 기반의 IO 캐시로 약 844TB의 용량을 제공한다. Lustre 파일시스템은 계산노드, 로그인노드, 아카이빙 서버(Data Mover)에 마운트되어 IO 서비스를 제공한다.



[병렬파일시스템 및 버스트버퍼 전체 랙 구성]



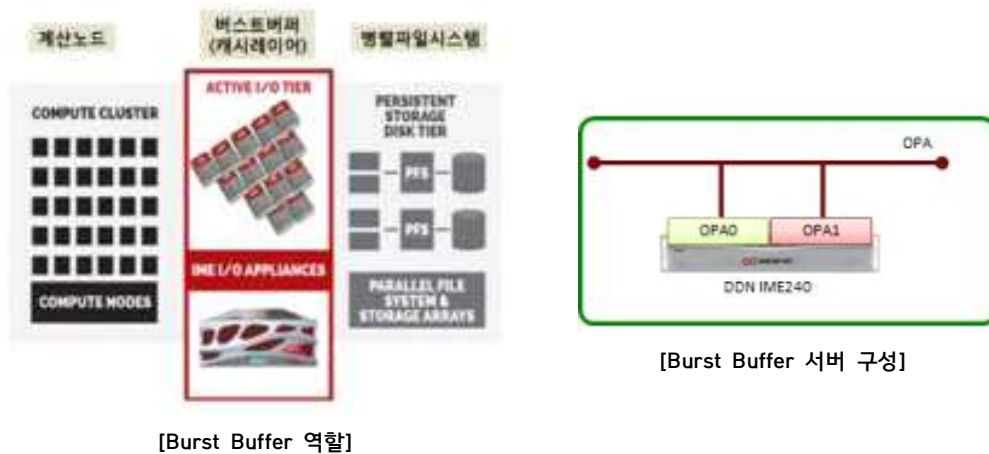
[PF5 서버 구성]



## 2. 버스트버퍼

### 1) 개요

버스트버퍼(Burst Buffer, 이하 BB)는 계산노드와 스토리지 (/scratch) 사이의 I/O 가속화를 위한 캐시 레이어로 병렬파일시스템의 small I/O 또는 random I/O에 대한 낮은 성능 보완 및 병렬 IO의 성능 극대화를 위해 5호기에 새로이 도입되었다. I/O 의존성이 높은 사용자 어플리케이션의 성능 향상을 목표로 하고 있으며, KNL 노드를 사용하는 모든 큐에 대해 지원하고 있다.



### 2) 시스템 구성

BB 구성을 위해 DDN사의 IME240 솔루션이 적용되었으며, 위 [Burst Buffer 서버 구성] 그림은 BB의 상세 구성을 나타낸다.

시스템	DDN IME240, Infinite Memory Engine Appliance
소프트웨어 버전	CentOS 7.4, IME 1.3
최대 IO 성능	20 GB/s
네트워크 인터페이스	2 x OPA
SSD 타입	1.2TB, NVMe 드라이브
SSD 수량	16ea(1.2TB NVMe) + 1ea(450GB SSD)
용량(RAW)	19.2TB

[IME 단일노드 구성]

IME240 x 48	IME240 x 48
총 용량(RAW)	979.2 TB
총 용량(패리티 적용 시)	816 TB (EC*, 10+2)
최대 IO 성능	800 GB/s

[버스트버퍼 전체 구성]

\* EC : Erasure Coding (설정은 변경될 수 있음)

## I -2. 사용자 환경

### 가. 계정발급

1. 누리온 시스템의 사용을 승인받은 연구자는 KISTI 홈페이지 (<https://www.ksc.re.kr>) 웹 서비스를 통해 계정을 신청한다.

#### 1) 신청 방법:

KISTI 홈페이지 웹사이트 접속, (상단) 사용신청 -> (상단) 신청 -> 신청서 선택

- 무료 계정 : 누리온 시스템 혁신지원 프로그램, 5호기 초보 사용자
- 유료 계정 : 5호기 일반사용자, 5호기 학생 사용자
- 계정 발급 완료 시 신청서에 기입한 이메일로 계정 관련 정보 발송

#### 2) OTP (One Time Password, 일회용 비밀번호) 인증코드 발급

수신하신 계정 정보 이메일을 참고하여 아래와 같이 작성하여 [account@ksc.re.kr](mailto:account@ksc.re.kr)을 통해 인증코드를 발급받는다.

메일 제목	OTP 인증코드 발송 요청 - 사용자 ID (예) OTP 인증코드 발송 요청 - x123abc
수신인	<a href="mailto:account@ksc.re.kr">account@ksc.re.kr</a>
메일 내용(예제)	로그인 ID: x123abc 휴대폰 번호: 010-1234-5678 이름: 홍길동 통신사: LG 유플러스(or SKT / KT)

#### 3) OTP 앱 설치

- 슈퍼컴퓨팅 보안 접속을 위해 OTP 스마트폰 앱이 제공된다.
- OTP 스마트폰 앱은 안드로이드 앱 스토어(Google Play)나 아이폰 앱 스토어(App Store)에서 “Any OTP”로 검색 후 미래기술(mirae-tech)에서 개발한 앱을 설치하여 사용할 수 있다.
- 슈퍼컴퓨터 로그인 시 “Any OTP” 앱의 OTP 보안숫자를 반드시 입력해야 한다.
  - 스마트폰을 사용하고 있지 않은 사용자의 경우,  
계정담당자([account@ksc.re.kr](mailto:account@ksc.re.kr))에게 문의
  - 자세한 OTP 설치 및 이용방법은 KISTI 홈페이지 > 기술지원 > 지침서에서 “OTP 사용자 매뉴얼” 참조
  - LG 유플러스의 경우에는 문자가 스팸처리되므로 이메일로 알려드립니다.

## 나. 로그인

- 사용자는 누리온 시스템 로그인 노드(nurion.ksc.re.kr)를 통해서 접근 가능하다.  
(하단, 노드 구성 참조)
  - 웹 브라우저를 통해 MyKSC (KISTI 슈퍼컴퓨터 웹 서비스 포털,  
<https://my.ksc.re.kr>)에 로그인하여 GUI 기반의 HPC 및 AI/데이터분석 서비스를  
활용할 수 있음(사용법은 MyKSC 지침서 참조)
- 기본 문자셋(encoding)은 유니코드(UTF-8)이다.
- 로그인 노드에 대한 접근은 ssh, scp, sftp, X11만 허용된다.

### 1. 유닉스 또는 리눅스 환경

```
$ ssh -l <사용자ID> nurion.ksc.re.kr [-p 22]
```

- 예) 사용자ID가 x123abc일 경우

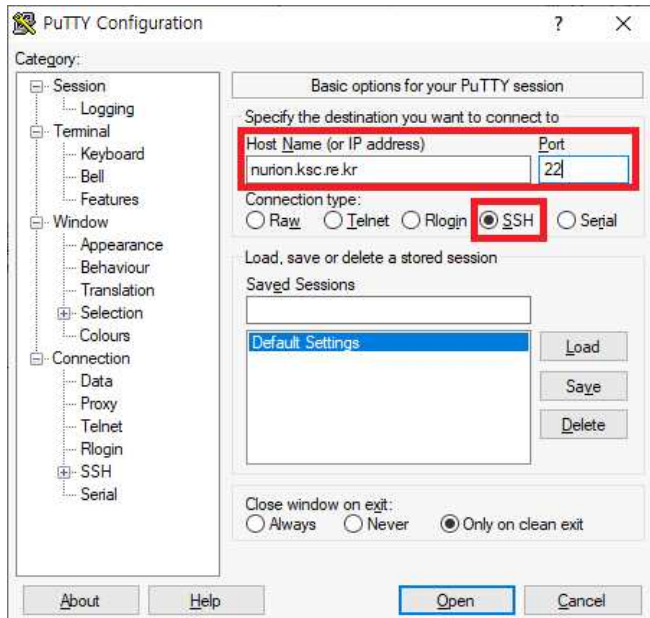
```
$ ssh -l x123abc nurion.ksc.re.kr  
혹은  
$ ssh -l x123abc nurion.ksc.re.kr -p 22
```

### 2. 윈도우 환경

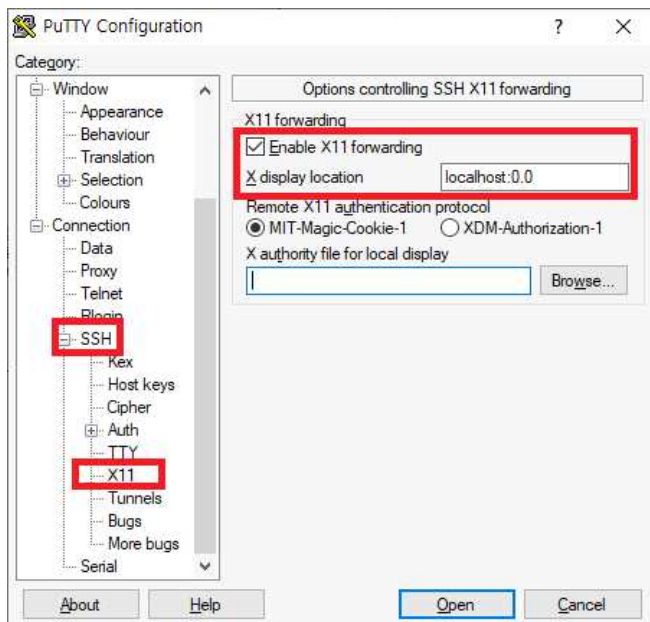
- X환경 실행을 위해 Xming 실행
  - 프로그램은 인터넷을 통해 무료로 다운로드 후 설치



- putty나 SSH Secure Shell Client 등의 ssh 접속 프로그램을 이용
  - Host Name : nurion.ksc.re.kr, Port : 22, Connection type : SSH
  - 프로그램은 인터넷을 통해 무료로 다운로드 가능



- ssh -> X11 tap -> check "Enable X11 forwarding"
- X display location : localhost:0.0



※ 만약, DNS 캐싱 문제로 접속이 안 될 경우에는 캐시를 정리 (명령 프롬프트에서 ipconfig/flushdns 명령어 수행)하고 재접속

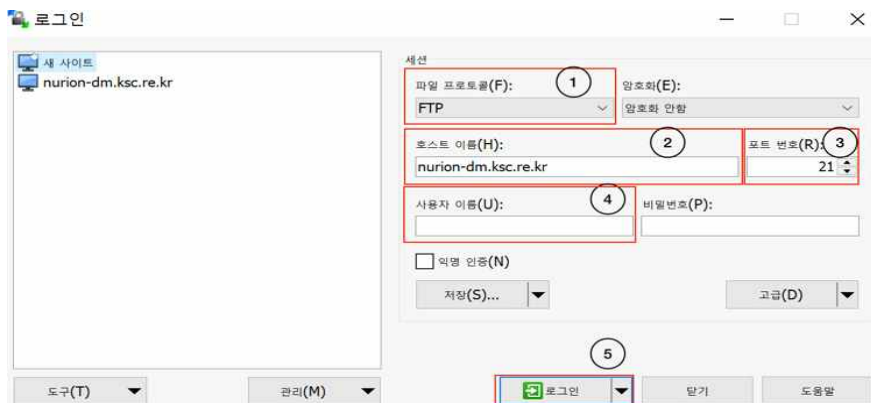
C: ipconfig /flushdns

### 3. 파일 송수신

- FTP 클라이언트를 통해 ftp나 sftp로 Datamover 노드(nurion-dm.ksc.re.kr)로 접속하여 파일을 송수신 한다(하단, 노드구성 참조)

```
$ ftp nurion-dm.ksc.re.kr
또는
$ sftp [사용자ID@]nurion-dm.ksc.re.kr [-p 22]
```

- 윈도우 환경에서는 WinSCP와 같이 무료로 배포되고 있는 FTP/SFTP 클라이언트 프로그램을 이용하여 접속한다.



- FTP (File Tranfer Protocal)을 이용하며, OTP를 입력하지 않고 파일 전송 가능
- SFTP(Secure-FTP) 을 이용하며, 파일 전송 시 OTP를 입력해야 함. (FTP보다 안전한 전송방식)

### 4. 노드 구성

	호스트명	CPU Limit	비고
로그인 노드	nurion.ksc.re.kr	20분	<ul style="list-style-type: none"> <li>ssh/scp 접속 가능</li> <li>컴파일 및 batch 작업제출용</li> <li>ftp/sftp 접속 불가</li> </ul>
Datamover 노드	nurion-dm.ksc.re.kr	-	<ul style="list-style-type: none"> <li>ssh/scp/sftp 접속 가능</li> <li>ftp 접속 가능</li> <li>컴파일 및 작업제출 불가</li> </ul>
계산 노드	KNL	node[0001-8305]	<ul style="list-style-type: none"> <li>PBS 스케줄러를 통해 작업 실행 가능</li> <li>일반사용자 직접 접근 불가</li> </ul>
	CPU -only	cpu0001-0132]	

- wget, git을 이용한 다운로드 및 대용량 데이터의 전송은 CPU Limit이 없는 Datamover 노드를 사용할 것을 권장함 (로그인 노드에서 수행 시에 CPU Limit에 따라 전송 중에 끊어질 수 있음)
- 파일 압축, 해제 또한 CPU Limit에 따라 중단될 수 있으며, 로그인 노드에서 프로세스가 강제로 중단되는 경우에는 Datamover 노드에서 압축/해제 진행



## 다. 사용자 셸 변경

- 누리온 시스템의 로그인 노드는 기본 셸로 bash이 제공된다. 다른 셸로 변경하고자 할 경우 chsh 명령어를 사용한다.

```
$ chsh
```

- 현재 사용 중인 셸을 확인하기 위해서 echo \$SHELL을 이용하여 확인한다.

```
$ echo $SHELL
```

- 셸의 환경설정은 사용자의 홈 디렉터리에 있는 환경설정 파일(.bashrc, .cshrc 등)을 수정하여 사용하면 된다.

## 라. 사용자 비밀번호 변경

- 사용자 패스워드를 변경하기 위해서는 로그인 노드에서 passwd 명령을 사용한다.

```
$ passwd
```

- 패스워드 관련 보안 정책
  - 사용자 패스워드 길이는 최소 9자이며, 영문, 숫자, 특수문자의 조합으로 이뤄져야 한다. 영문 사전 단어는 사용이 불가하다.
  - 사용자 패스워드 변경 기간은 2개월로 설정(60일)된다.
  - 새로운 패스워드는 최근 5개의 패스워드와 유사한 것을 사용할 수 없다.
  - 최대 로그인 실패 허용 횟수 : 5회
    - 5회 이상 틀릴 경우, 이 계정의 ID는 lock이 걸리므로, 계정담당자 (account@ksc.re.kr)에게 문의해야 한다.
    - 같은 PC에서 접속을 시도하여 5회 이상 틀릴 경우, 해당 PC의 IP 주소는 일시적으로 block 되므로 이 경우에도 계정담당자(account@ksc.re.kr)에게 문의해야 한다.
  - OTP 인증 오류 허용 횟수 : 5회
    - 5회 이상 틀릴 경우, 계정담당자(account@ksc.re.kr)에게 문의해야 한다.

## 마. 사용자 계정 SRU Time 사용량 확인

- 통합 슈퍼컴퓨터 계정 관리 시스템(ISAM)에 접속하여 사용자 계약 정보, 배치 작업 사용 상세 내역, 총 SRU time 사용량 등을 확인할 수 있다.

```
$ isam
```

## 바. 사용자 파일시스템 및 쿼터 정책

- 사용자에게는 홈 디렉터리(/home01/\$USER)와 스크래치 디렉터리(/scratch/\$USER) 2개의 파일시스템이 제공된다.

구분	디렉터리 경로	용량 제한	파일수 제한	파일 삭제(퍼지) 정책	파일시스템	백업 유무
홈 디렉터리	/home01	64GB	200K	N/A	Lustre	O
스크래치 디렉터리	/scratch	100TB	1M	15일 동안 접근하지 않은 파일은 자동 삭제*		X

- 홈 디렉터리는 용량 및 I/O 성능이 제한되어 있어 모든 계산 작업은 스크래치 디렉터리인 /scratch의 사용자 작업 공간에서 이루어져야 한다.
- 사용자의 현재 사용량은 아래 명령어로 확인이 가능하다.

```
$ lfs quota -h /home01
$ lfs quota -h /scratch
```

- 파일 삭제 정책은 파일 속성 중 atime을 기준으로 적용된다.
  - atime(Access timestamp): 마지막 접근 시간을 나타냄
  - mtime(Modified timestamp): 파일의 마지막 변경 시간 의미
  - ctime(Change timestamp): 파일의 퍼미션, 소유자, 링크 등에 의한 inode 변경 시간 의미

※ 삭제 정책(15일 동안 접근하지 않은 파일은 자동 삭제함)에 따라 삭제된 파일의 복구는 기본적으로 불가능하나 파일 관리를 잊은 사용자에게 유예기간을 제공하기 위해 퍼지 정책을 일부 변경합니다. (2023.05. 시행)

- 삭제 정책 수행 후 대상이 되는 파일은 앞에 접두사(ToBeDelete\_)가 붙게 됩니다.
  - 예) ToBeDelete\_file1.txt, ToBeDelete\_file2.txt
  - 대상 파일이 필요한 경우 사용자는 직접 접두사가 붙은 파일명을 복구시키고 atime을 업데이트해야 합니다.
- 일정 기간(20일~30일) 후 ToBeDelete\_붙은 파일은 일괄 삭제됩니다. 이후 해당 파일은 복구가 불가능합니다.

- atime은 stat, ls -lu 명령 등으로 확인할 수 있다.

```
[login01 ~]# stat file1
File: 'file1' Size: 0 Blocks: 0 IO Block: 4096 regular empty file
Device: 805h/2053d Inode: 3221237903 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)
Access: 2022-04-05 08:52:30.113048319 +0900
Modify: 2022-04-05 08:52:30.113048319 +0900
Change: 2022-04-05 08:52:30.113048319 +0900
```

```
$ ls -lu test.file
-rw-r--r-- 1 testuser testgroup 58 Jan 1 17:06 test.file
```

## I -3. 사용자 프로그래밍 환경

### 가. 프로그래밍 도구 설치 현황

- 컴파일러 및 라이브러리 모듈
  - Abaqus는 2023년 8월 9일로 서비스 재개되었음

구분	항목(이름/버전)	
아키텍처 구분 모듈	• craype-mic-knl	• craype-x86-skylake
컴파일러	• intel/17.0.5 • intel/18.0.1 • intel/18.0.3 • intel/19.0.1 • intel/19.0.4 • intel/19.0.5 • intel/19.1.2 • intel/oneapi_21.2	• cce/8.6.3 • gcc/6.1.0 • gcc/7.2.0 • gcc/8.3.0 • pgi/18.10 • pgi/19.1
컴파일러 의존 라이브러리	• hdf4/4.2.13 • hdf5/1.10.2 • lapack/3.7.0 • ncl/6.7.0	• ncview/2.1.7 • NCO/4.7.4 • netcdf/4.6.1
MPI	• impi/17.0.5 • impi/18.0.1 • impi/18.0.3 • impi/19.0.1 • impi/19.0.4 • impi/19.0.5 • impi/19.1.2 • impi/oneapi_21.2	• mvapich2/2.3 • mvapich2/2.3.1 • openmpi/3.1.0 • mvapich-verbs/2.2.ddn1.4 • ime/mvapich-verbs/2.2.ddn1.4
MPI 의존 라이브러리	• fftw_mpi/2.1.5 • fftw_mpi/3.3.7 • hdf5-parallel/1.10.2	• netcdf-hdf5-parallel/4.6.1 • parallel-netcdf/1.10.0 • pio/2.3.1
상용 소프트웨어	• Abaqus/6.14-6 • Abaqus/2016 • Abaqus/2017 • Abaqus/2018 • Abaqus/2019	• cfx/v145 • cfx/v170 • cfx/v181 • cfx/v191 • cfx/v192

	<ul style="list-style-type: none"> <li>• Abaqus/2020</li> <li>• Abaqus/2021</li> <li>• Abaqus/2022</li> <li>• Abaqus/2023</li> <li>• gaussian/g16.a03</li> <li>• gaussian/g16.a03.linda</li> <li>• gaussian/g16.b01.linda</li> <li>• gaussian/g16.c01.linda</li> </ul>	<ul style="list-style-type: none"> <li>• cfx/v195</li> <li>• cfx/v201</li> <li>• cfx/v202</li> <li>• cfx/v212</li> <li>• cfx/v221</li> <li>• cfx/v222</li> <li>• cfx/v231</li> <li>• cfx/v232</li> <li>• fluent/v145</li> <li>• fluent/v170</li> <li>• fluent/v181</li> <li>• fluent/v191</li> <li>• fluent/v192</li> <li>• fluent/v195</li> <li>• fluent/v201</li> <li>• fluent/v202</li> <li>• fluent/v212</li> <li>• fluent/v221</li> <li>• fluent/v222</li> <li>• fluent/v231</li> <li>• fluent/v232</li> </ul>
응용 소프트웨어	<ul style="list-style-type: none"> <li>• cp2k/5.1.0</li> <li>• cp2k/6.1.0</li> <li>• ferret/7.4.3</li> <li>• forge/18.1.2</li> <li>• grads/2.2.0</li> <li>• gromacs/2016.4</li> <li>• gromacs/2018.6</li> <li>• gromacs/2020.2</li> <li>• gromacs/5.0.6</li> <li>• lammmps/12Dec18</li> <li>• lammmps/8Mar18</li> <li>• lammmps/3Mar20</li> </ul>	<ul style="list-style-type: none"> <li>• namd/2.12</li> <li>• namd/2.13</li> <li>• PETSc/3.8.4</li> <li>• python/2.7.15</li> <li>• python/3.7</li> <li>• python/3.9.5 <ul style="list-style-type: none"> <li>• qe/6.1</li> <li>• qe/6.4.1</li> <li>• qe/6.6</li> <li>• R/3.5.0</li> <li>• R/4.2.1</li> </ul> </li> <li>• siesta/4.0.2</li> <li>• siesta/4.1-b3</li> </ul>
가상화 모듈	<ul style="list-style-type: none"> <li>• singularity/3.6.4</li> <li>• singularity/3.9.7</li> </ul>	<ul style="list-style-type: none"> <li>• conda/pytorch_1.0</li> <li>• conda/tensorflow_1.13</li> <li>• conda/intel_caffe_1.1.5</li> </ul>

Intel 디버깅 모듈	<ul style="list-style-type: none"> <li>• advisor/17.0.5</li> <li>• advisor/18.0.1</li> <li>• advisor/18.0.3</li> <li>• advisor/18.0.3a</li> </ul>	<ul style="list-style-type: none"> <li>• vtune/17.0.5</li> <li>• vtune/18.0.1</li> <li>• vtune/18.0.2</li> <li>• vtune/18.0.3</li> </ul>
Cray 모듈	<ul style="list-style-type: none"> <li>• cdt/17.10</li> <li>• cray-ccdb/3.0.3</li> <li>• cray-cti/1.0.6</li> <li>• cray-fftw/3.3.6.2</li> <li>• cray-fftw impi/3.3.6.2</li> <li>• cray-impi/1.1.4</li> <li>• cray-lgdb/3.0.7</li> <li>• cray-libsci/17.09.1</li> <li>• craype/2.5.13</li> <li>• craypkg-gen/1.3.5</li> <li>• chklimit/1.0</li> </ul>	<ul style="list-style-type: none"> <li>• vapich2_cce/2.2rc1.0.3_noslurm</li> <li>• vapich2_gnu/2.2rc1.0.3_noslurm</li> <li>• papi/5.5.1.3</li> <li>• perftools/6.5.2</li> <li>• perftools-bas/6.5.2</li> <li>• perftools-lite/6.5.2</li> <li>• PrgEnv-cray/1.0.2</li> <li>• libfabric/1.7.0</li> <li>• pbs/trace</li> <li>• pbs/tools</li> </ul>
etc	<ul style="list-style-type: none"> <li>• cmake/3.12.3</li> <li>• cmake/3.17.4</li> <li>• cmake/3.26.2</li> <li>• common/memkind-1.9.0</li> <li>• git/1.8.3.4</li> <li>• git/2.35.1</li> <li>• IGPROF/5.9.16</li> </ul>	<ul style="list-style-type: none"> <li>• ImageMagick/7.0.8-20</li> <li>• perl/5.28.1</li> <li>• qt/4.8.7</li> <li>• qt/5.9.6</li> <li>• subversion/1.7.19</li> <li>• subversion/1.9.3</li> </ul>

- apps/Modules/modulefiles/test는 test 모듈임
- ANSYS 사용이 가능한 사용자 그룹은 대학교, 산업체(중소기업), 출연연구소에 한함.
  - 사용 가능 사용자 그룹이 아니거나 사용 신청을 하지 않은 타 기관 사용자에게 의한 ANSYS 사용은 ANSYS 사에 의해 법적 제재를 당할 수 있음에 유의하시기 바랍니다.
- Abaqus 사용이 가능한 사용자 그룹은 대학교, 산업체(중소기업), 출연연구소에 한함.
  - 사용 가능 사용자 그룹이 아니거나 사용 신청을 하지 않은 타 기관 사용자에게 의한 Abaqus 사용은 다쏘시스템코리아에 의해 법적 제재를 당할 수 있음에 유의하시기 바랍니다.
- Gaussian은 helpdesk 계정담당자(account@ksc.re.kr)를 통해 사용 권한 신청 후 사용 가능함
- 공통 라이브러리(예: cairo, expat, jasper, libpng, udunits 등) 설치 현황은 [별첨 5]을 참조



## 나. 컴파일러 사용법

### 1. 컴파일러 및 MPI 환경설정(modules)

#### 1) 기본 필요 모듈

사용할 계산 노드에 해당되는 모듈 하나를 반드시 추가해야 한다.

사용할 계산 노드	KNL	SKL
모듈명	craype-mic-knl	craype-x86-skylake

```
$ module load craype-mic-knl  
혹은  
$ module load craype-x86-skylake
```

#### 2) 모듈 관련 기본 명령어

- 사용 가능한 모듈 목록 출력

사용할 수 있는 컴파일러, 라이브러리 등의 모듈 목록을 확인할 수 있다.

```
$ module avail  
혹은  
$ module av
```

- 사용할 모듈 추가

사용하고자 하는 컴파일러, 라이브러리 등의 모듈을 추가할 수 있다.

사용할 모듈들을 한 번에 추가할 수 있다.

```
$ module load [module name] [module name] [module name] ...  
혹은  
$ module add [module name] [module name] [module name] ...  
  
예)  
$ module load intel/19.1.2 impi/19.1.2 python/3.7
```

- 사용 모듈 삭제

필요 없는 모듈을 제거한다. 이때 한 번에 여러 개의 모듈을 삭제할 수 있다.

```
$ module unload [module name] [module name] [module name] ...  
혹은  
$ module rm [module name] [module name] [module name] ...
```

- 사용 모듈 목록 출력

현재 설정된 모듈 목록을 확인할 수 있다.

```
$ module list
```

혹은

```
$module li
```

- 전체 사용 모듈 일괄 삭제

```
$ module purge
```

이 경우, 기본 필요 모듈도 일괄 삭제되므로, 모듈 재사용 시 기본 필요 모듈을 다시 추가해야 한다.

- 기타 module 활용 예시
  - module help 명령 (도움말 보기)

```
$ module help [module name]
```

- module show 명령 (설정 환경변수 확인)

```
$ module show [module name]
```

## 2. 순차 프로그램 컴파일

순 프로그램은 병렬 프로그램 환경을 고려하지 않은 프로그램을 말한다. 즉, OpenMP, MPI와 같은 병렬 프로그램 인터페이스를 사용하지 않는 프로그램으로써, 하나의 노드에서 하나의 프로세서만 사용해 실행되는 프로그램이다. 순차 프로그램 컴파일 시 사용되는 컴파일러별 옵션은 병렬 프로그램을 컴파일할 때도 그대로 사용되므로, 순차 프로그램에 관심이 없다 하더라도 참조하는 것이 좋다.

### 1) Intel 컴파일러

Intel 컴파일러를 사용하기 위해서 필요한 버전의 Intel 컴파일러 모듈을 추가하여 사용한다. 사용 가능한 모듈은 module avail로 확인할 수 있다.

```
$ module load intel/18.0.3
```

※ 프로그래밍 도구 설치 현황 표를 참고하여 사용 가능 버전 확인

#### • 컴파일러 종류

컴파일러	프로그램	소스 확장자
icc / icpc	C / C++	.C, .cc, .cpp, .cxx, .c++
ifort	F77/F90	.f, .for, .ftn, .f90, .fpp, .F, .FOR, .FTN, .FPP, .F90

#### • 컴파일러 옵션

컴파일러 옵션	설명
-O[1 2 3]	오브젝트 최적화. 숫자는 최적화 레벨
-ip, ipo	프로시저 간 최적화
-qopt_report=[0 1 2 3 4]	벡터 진단 정보의 양을 조절
-xCORE-AVX512	512bits 레지스터를 가진 CPU를 지원 (SKL 노드 사용 계산의 경우)
-xMIC-AVX512	512bits 레지스터를 가진 MIC를 지원 (KNL노드 사용 계산의 경우)
-fast	-O3 -ipo -no-prec-div -static, -fp-model fast=2 매크로
-static/-static-intel/ -i_static	공유 라이브러리를 링크하지 못하게 함
-shared/- shared-intel/ -i_dynamic	공유 라이브러리를 링크를 함
-g -fp	디버깅 정보를 생성

-qopenmp	OpenMP 기반의 multi-thread 코드 사용
-openmp_report=[0 1 2]	OpenMP 병렬화 진단 레벨 조절
-ax -axS	특정 프로세서에 최적화된 코드를 생성 SIMD Extensions4(SSE4) 벡터라이징 컴파일러와 미디어 가속 명령어들을 활용하는 특화된 코드를 생성
-tcheck	스레드 기반의 프로그램의 분석을 활성화 함
-pthread	멀티스레딩 지원을 받기 위해 pthread 라이브러리를 추가 함
-msse<3,4.1>,- msse3	Streaming SIMD Extensions 3 지원
-fPIC,fpic	PIC (Position Independent Code)가 되도록 컴파일
-p	프로파일링 정보를 생성(gmon.out)
-unroll	unroll활성화, 은 최대 횟수 (64비트만 지원)
-mcmmodel medium	2GB이상의 memory allocation이 필요한 경우 사용
-help	옵션 목록 출력

- Intel 컴파일러 사용 예제

다음은 **KNL 계산노드**에서 test 예제파일을 intel 컴파일러로 컴파일하여 실행파일 test.exe를 만드는 예시임

```
$ module load craype-mic-knl intel/18.0.3
$ icc -o test.exe -O3 -fPIC -xMIC-AVX512 test.c
혹은
$ ifort -o test.exe -O3 -fPIC -xMIC-AVX512 test.f90
$ ./test.exe
```

※ /apps/shell/home/job\_examples에서 작업제출 test 예제파일을 복사하여 사용 가능

- 권장 옵션

계산노드	권장옵션
SKL	-O3 -fPIC -xCORE-AVX512
KNL	-O3 -fPIC -xMIC-AVX512
SKL & KNL	-O3 -fPIC -xCOMMON-AVX512

## 2) GNU 컴파일러

GNU 컴파일러를 사용하기 위해서 필요한 버전의 GNU 컴파일러 모듈을 추가하여 사용한다.  
사용 가능한 모듈은 module avail로 확인할 수 있다.

```
$ module load gcc/7.2.0
```

- ※ 프로그래밍 도구 설치 현황표를 참고하여 사용 가능 버전 확인
- ※ 반드시 "gcc/6.1.0" 이상 버전을 사용

### • 컴파일러 종류

컴파일러	프로그램	소스 확장자
gcc / g++	C / C++	.C, .cc, .cpp, .cxx, .c++
gfortran	F77/F90	.f, .for, .ftn, .f90, .fpp, .F, .FOR, .FTN, .FPP, .F90

### • GNU 컴파일러 옵션

컴파일러 옵션	설명
-O[1 2 3]	오브젝트 최적화. 숫자는 최적화 레벨
-march=skylake-avx512 -march=knl	512bits 레지스터를 가진 CPU를 지원 (SKL 노드 사용 계산의 경우) 512bits 레지스터를 가진 MIC를 지원 (KNL 노드 사용 계산의 경우)
-Ofast	-O3 -ffast-math 매크로
-funroll-all-loops	모든 루프를 unrolling함
-ffast-math	fast floating point model 사용
-mline-all-stringops	더 많은 inlining 허용, memcpy, strlen, memsetdp 의존적인 코드의 성능을 향상 시킴
-fopenmp	OpenMP 기반의 multi-thread 코드 사용
-g	디버깅 정보를 생성
-pg	프로파일링 정보를 생성(gmont.out)
-fPIC	PIC(Position Independent Code)가 생성되도록 컴파일
-help	옵션 목록 출력

- GNU 컴파일러 사용 예제

다음은 KNL 계산노드에서 test 예제파일을 GNU 컴파일러로 컴파일하여 실행파일 test.exe를 만드는 예시임.

```
$ module load craype-mic-knl gcc/7.2.0
$ gcc -o test.exe -O3 -fPIC -march=knl test.c
혹은
$ gfortran -o test.exe -O3 -fPIC -march=knl test.f90
$ ./test.exe
```

※ /apps/shell/home/job\_examples에서 작업제출 test 예제파일을 복사하여 사용 가능

- 권장 옵션

계산노드	권장옵션
SKL	-O3 -fPIC -march=skylake-avx512
KNL	-O3 -fPIC -march=knl
SKL & KNL	-fPIC -mpku



### 3) PGI 컴파일러

PGI 컴파일러를 사용하기 위해서 필요한 버전의 PGI 컴파일러 모듈을 추가하여 사용한다. 사용 가능한 모듈은 module avail로 확인할 수 있다.

```
$ module load pgi/18.10
```

※ 프로그래밍 도구 설치 현황표를 참고하여 사용 가능 버전 확인

#### • 컴파일러 종류

컴파일러	프로그램	소스 확장자
pgcc / pgc++	C / C++	.C, .cc, .cpp, .cxx, .c++
pgfortran	F77/F90	.f, .for, .ftn, .f90, .fpp, .F, .FOR, .FTN, .FPP, .F90

#### • PGI 컴파일러 옵션

컴파일러 옵션	설명
-O[1 2 3 4]	오브젝트 최적화. 숫자는 최적화 레벨
-Mipa=fast	프로시저 간 최적화
-fast	-O2 -Munroll=c:1 -Mnoframe -Mlre -Mautoinline의 매크로
-fastsse	SSE, SSE2를 지원하는 최적화
-g, -gopt	디버깅 정보를 생성
-mp	OpenMP 기반의 multi-thread 코드 사용
-Minfo=mp, ipa	OpenMP관련 정보, 프로시저 간 최적화
-pg	프로파일링 정보를 생성(gmon.out)
-Mprof=time -Mprof=func -Mprof=lines	PGPROF output file 생성 - time에 기초한 명령어 단위의 프로파일링 정보를 생성, 많이 사용됨 - function 단위의 프로파일링 정보를 생성 - line 단위의 프로파일링 정보를 생성 (-Mprof=lines의 경우 overhead의 증가로 계산시간이 매우 느려질 수 있음)
-mcrmodel medium	2GB이상의 memory allocation이 필요한 경우 사용
-tp=skylake	Skylake 아키텍처 프로세서 전용 옵션
-tp=knl	KNL 아키텍처 프로세서 전용 옵션
-fPIC	PIC(Position Independent Code)가 생성되도록 컴파일
-help	옵션 목록 출력

- PGI 컴파일러 사용 예제

다음은 **KNL 계산노드**에서 test 예제파일을 PGI 컴파일러로 컴파일하여 실행파일 test.exe를 만드는 예시임

```
$ module load craype-mic-knl pgi/18.10
$ pgcc -o test.exe -fast -tp=knl test.c
혹은
$ pgfortran -o test.exe -fast -tp=knl test.f90
$ ./test.exe
```

※ /apps/shell/home/job\_examples에서 작업제출 test 예제파일을 복사하여 사용 가능

- 권장 옵션

계산노드	권장옵션
SKL	-fast -tp=skylake
KNL	-fast -tp=knl
SKL & KNL	-fast -tp=skylake,knl

#### 4) Cray 컴파일러

Cray 컴파일러를 사용하기 위해서 필요한 버전의 Cray 컴파일러 모듈을 추가하여 사용한다. 사용 가능한 모듈은 module avail로 확인할 수 있다.

```
$ module load cce/8.6.3
```

※ 프로그래밍 도구 설치 현황표를 참고하여 사용 가능 버전 확인

##### • 컴파일러 종류

컴파일러	프로그램	소스 확장자
cc / CC	C / C++	.C, .cc, .cpp, .cxx, .c++
ftn	F77/F90	.f, .for, .ftn, .f90, .fpp, .F, .FOR, .FTN, .FPP, .F90

##### • 컴파일러 옵션

컴파일러 옵션	설명
-O[1 2 3]	오브젝트 최적화. 숫자는 최적화 레벨
-hcpu=mic-knl	512bits 레지스터를 가진 MIC를 지원
-Oipa[0]	1
-hunroll[0 1 2]	Unrolling 옵션. Default 2인 경우 모든 루프를 unrollinig
-hfp[0 1 2 3 4]	Floating_Point 최적화
-h omp(default)	OpenMP 기반의 multi-thread 코드 사용
-g   -G0	디버깅 정보를 생성
-h pic	2GB이상의 static memory가 필요한 경우 사용 (-dynamic 함께 사용)
-dynamic	공유 라이브러리를 링크함

##### • Cray 컴파일러 사용 예제

다음은 KNL 계산노드에서 test 예제파일을 Cray 컴파일러로 컴파일하여 실행파일 test.exe를 만드는 예시임

```
$ module load craype-mic-knl cce/8.6.3 PrgEnv-cray/1.0.2
$ cc -o test.exe -hcpu=mic-knl test.c
혹은
$ ftn -o test.exe -hcpu=mic-knl test.f90
$ ./test.exe
```

※ /apps/shell/home/job\_examples에서 작업제출 test 예제 파일을 복사하여 사용 가능

- 권장 옵션

계산노드	권장옵션
SKL	기본값
KNL	-hcpu=mic-knl
SKL & KNL	기본값

- ※ 테스트를 위한 test.c 및 test.f90 등은 /apps/shell/home/job\_examples에서 확인 가능  
(사용자 디렉터리로 복사하여 테스트)
- ※ KNL 최적화 옵션을 사용할 프로그램은 KNL debug 노드로 인터랙티브 작업 제출을  
통해 접속한 후 컴파일하는 것을 권장  
(“스케줄러를 통한 작업실행 → 나. 작업제출 모니터링 → 2) 인터랙티브 작업제출” 참고)

### 3. 병렬 프로그램 컴파일

#### 1) OpenMP 컴파일

OpenMP는 컴파일러 지시자만으로 멀티 스레드를 활용할 수 있도록 간단하게 개발된 기법으로 OpenMP를 사용한 병렬 프로그램 컴파일 시 사용되는 컴파일러는 순차 프로그램과 동일하며, 컴파일러 옵션을 추가하여 병렬 컴파일을 할 수 있는데, 현재 대부분의 컴파일러가 OpenMP 지시자를 지원한다.

컴파일러 옵션	프로그램	옵션
icc / icpc / ifort	C / C++ / F77/F90	-qopenmp
gcc / g++ / gfortran	C / C++ / F77/F90	-fopenmp
cc / CC / ftn	C / C++ / F77/F90	-homp
pgcc / pgc++ / pgfortran	C / C++ / F77/F90	-mp

- OpenMP 프로그램 컴파일 예시 (Intel 컴파일러)

다음은 KNL 계산노드에서 openMP를 사용하는 test\_omp 예제파일을 intel 컴파일러로 컴파일하여 실행파일 test\_omp.exe를 만드는 예시임

```
$ module load craype-mic-knl intel/18.0.3
$ icc -o test_omp.exe -qopenmp -O3 -fPIC -xMIC-AVX512 test_omp.c
혹은
$ ifort -o test_omp.exe -qopenmp -O3 -fPIC -xMIC-AVX512 test_omp.f90
$ ./test_omp.exe
```

- OpenMP 프로그램 컴파일 예시 (GNU 컴파일러)

다음은 KNL 계산노드에서 openMP를 사용하는 test\_omp 예제파일을 GNU 컴파일러로 컴파일하여 실행파일 test\_omp.exe를 만드는 예시임

```
$ module load craype-mic-knl gcc/7.2.0
$ gcc -o test_omp.exe -fopenmp -O3 -fPIC -march=knl test_omp.c
혹은
$ gfortran -o test_omp.exe -fopenmp -O3 -fPIC -march=knl test_omp.f90
$ ./test_omp.exe
```

- OpenMP 프로그램 컴파일 예시 (PGI 컴파일러)

다음은 **KNL 계산노드**에서 **openMP**를 사용하는 test\_omp 예제파일을 PGI 컴파일러로 컴파일하여 실행파일 test\_omp.exe를 만드는 예시임

```
$ module load craype-mic-knl pgi/18.10
$ pgcc -o test_omp.exe -mp -fast test_omp.c
혹은
$ pgfortran -o test_omp.exe -mp -fast test_omp.f90
$ ./test_omp.exe
```

- OpenMP 프로그램 컴파일 예시 (Cray 컴파일러)

다음은 **KNL 계산노드**에서 **openMP**를 사용하는 test\_omp 예제파일을 Cray 컴파일러로 컴파일하여 실행파일 test\_omp.exe를 만드는 예시임

```
$ module load craype-mic-knl cce/8.6.3 PrgEnv-cray/1.0.2
$ cc -o test_omp.exe -homp -hcpu=mic-knl test_omp.c
혹은
$ ftn -o test_omp.exe -homp -hcpu=mic-knl test_omp.f90
$ ./test_omp.exe
```



## 2) MPI 컴파일

사용자는 다음 표의 MPI 명령을 실행할 수 있는데, 이 명령은 일종의 wrapper로써 .bashrc를 통해 지정된 컴파일러가 소스를 컴파일하게 된다.

구분	Intel	GNU	PGI	Gray
Fortran	ifort	gfortran	pgfortran	ftn
Fortran + MPI	mpiifort	mpif90	mpif90	ftn
C	icc	gcc	pgcc	cc
C + MPI	mpiicc	mpicc	mpicc	cc
C++	icpc	g++	pgc++	CC
C++ + MPI	mpiicpc	mpicxx	mpicxx	CC

mpicc로 컴파일을 하더라도, 옵션은 wrapping되는 본래의 컴파일러에 해당하는 옵션을 사용해야 한다.

- MPI 프로그램 컴파일 예시 (Intel 컴파일러)

다음은 **KNL 계산노드에서 MPI**를 사용하는 test\_mpi 예제파일을 Intel 컴파일러로 컴파일하여 실행파일 test\_mpi.exe를 만드는 예시임

```
$ module load craype-mic-knl intel/18.0.3 impi/18.0.3
$ mpiicc -o test_mpi.exe -O3 -fPIC -xMIC-AVX512 test_mpi.c
혹은
$ mpiifort -o test_mpi.exe -O3 -fPIC -xMIC-AVX512 test_mpi.f90
$ mpirun -np 2 ./test_mpi.exe
```

- MPI 프로그램 컴파일 예시 (GNU 컴파일러)

다음은 **KNL 계산노드에서 MPI**를 사용하는 test\_mpi 예제파일을 GNU 컴파일러로 컴파일하여 실행파일 test\_mpi.exe를 만드는 예시임

```
$ module load craype-mic-knl gcc/7.2.0 openmpi/3.1.0
$ mpicc -o test_mpi.exe -O3 -fPIC -march=knl test_mpi.c
혹은
$ mpif90 -o test_mpi.exe -O3 -fPIC -march=knl test_mpi.f90
$ mpirun -np 2 ./test_mpi.exe
```

- MPI 프로그램 컴파일 예시 (PGI 컴파일러)

다음은 **KNL 계산노드에서 MPI**를 사용하는 test\_mpi 예제파일을 PGI 컴파일러로 컴파일하여 실행파일 test\_mpi.exe를 만드는 예시임

```
$ module load craype-mic-knl pgi/18.10 openmpi/3.1.0
$ mpicc -o test_mpi.exe -O3 -fPIC -tp=kn1 test_mpi.c
혹은
$ mpifort -o test_mpi.exe -O3 -fPIC -tp=kn1 test_mpi.f90
$ mpirun -np 2 ./test_mpi.exe
```

- MPI 프로그램 컴파일 예시 (Cray 컴파일러)

다음은 **KNL 계산노드에서 MPI**를 사용하는 test\_mpi 예제파일을 Cray 컴파일러로 컴파일하여 실행파일 test\_mpi.exe를 만드는 예시임

```
$ module load craype-mic-knl cce/8.6.3 PrgEnv-cray/1.0.2
$ cc -o test_mpi.exe -hcpu=mic-knl test_mpi.c
혹은
$ ftn -o test_mpi.exe -hcpu=mic-knl test_mpi.f90
$ mpirun -np 2 ./test_mpi.exe
```

## 다. 디버거 및 프로파일러

5호기 누리온 시스템 베타서비스는 사용자의 프로그램 디버깅을 위하여 DDT를 제공한다. 또한 사용자의 프로그램 프로파일링을 위하여 Intel vtune와 CaryPat 두 가지의 프로파일러를 제공한다.

## 1. 디버거 DDT 사용예제

- 5호기 시스템에서 DDT를 사용하기 위하여 사용할 아키텍처, 컴파일러, MPI를 선택하고 DDT를 사용하기 위한 모듈까지 선택한다.

```
$ module load craype-mic-knl or craype-x86-skylake
$ module load intel/17.0.5 impi/17.0.5
$ module load forge/18.1.2
```

- 본 예제는 위와 같은 환경에서 테스트 되었다.
- DDT를 사용하기 전 사전 준비로 컴파일 시 -g -O0 옵션을 넣어 실행파일을 선택한다.

```
$ mpiicc -o test.x -g -O0 test.c
```

- 사용자의 데스크탑에서 xming 실행 및 ssh X환경 설정 완료한 후 ddt 실행 명령을 실행한다.

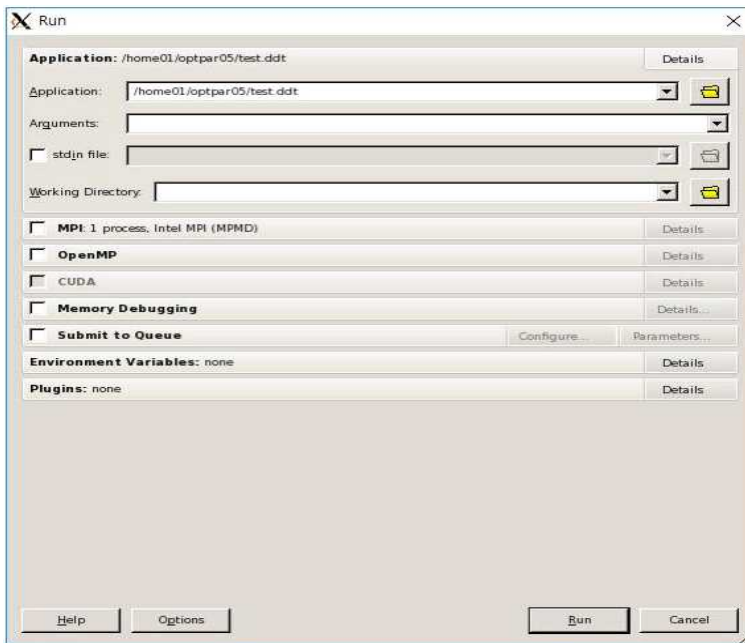
---

\$ ddt &

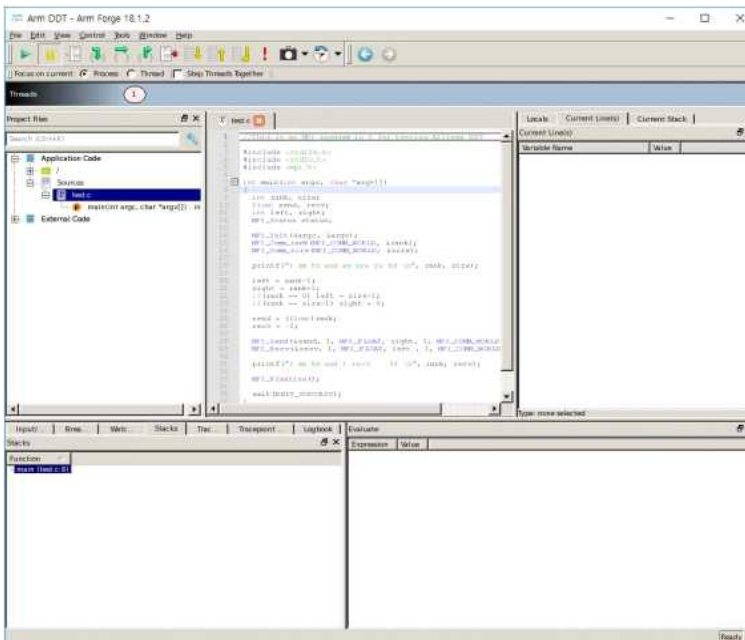
- 명령을 실행하여 다음과 같은 팝업 실행 창을 확인한다.



- 위 명령 중 "RUN" 을 선택하여 다음과 같이 디버깅할 파일 선택한 후 다시 새로운 팝업 창의 "RUN" 선택한다.



- 다음과 같이 선택된 실행 파일에 대한 디버깅 모드로 진입하여 디버깅을 진행할 수 있다.



## 2. 프로파일러 Intel vtune Amplifier 사용예제

- 본 시스템에서 프로파일러인 vtune을 사용하기 위하여 아키텍처, 컴파일러, MPI를 선택한 후 vtune을 선택하면 프로파일러를 사용할 수 있다.

```
$ module load craype-mic-knl or craype-x86-skylake
$ module load intel/17.0.5 impi/17.0.5
$ module load vtune/17.0.5
```

본 예제는 위와 같은 환경에서 진행되었다.

- CLI 사용법
  - Intel vtune Amplifier를 CLI 모드로 실행할 때 명령어는 아래와 같은 형식이다.
- \$ amplxe-cl 옵션 분석할 프로그램 실행

```
$ amplxe-cl -collect hotspots /home01/$USER/test/test.x

amplxe: Using result path '/home01/$USER/test/r000hs'
amplxe: Executing actions 75% Generating a report
Function CPUTime CPUTime:EffectiveTime CPUTime:EffectiveTime:Idle
CPUTime:EffectiveTime:Poor CPUTime:EffectiveTime:Ok CPUTime:EffectiveTime:Ideal
CPUTime:EffectiveTime:Over CPUTime:SpinTime CPUTime:OverheadTime Module Function(full)
SourceFile StartAddress

multiply1 21.568s 21.568s 0.176s 21.392s 0s 0s 0s 0s 0s martix.mic multiply1 multiply.c
0x401590
init_arr 0.020s 0.020s 0s 0.020s 0s 0s 0s 0s 0s matrix.mic init_arr matrix.c
0x402384
amplxe: Executing actions 100% done
```

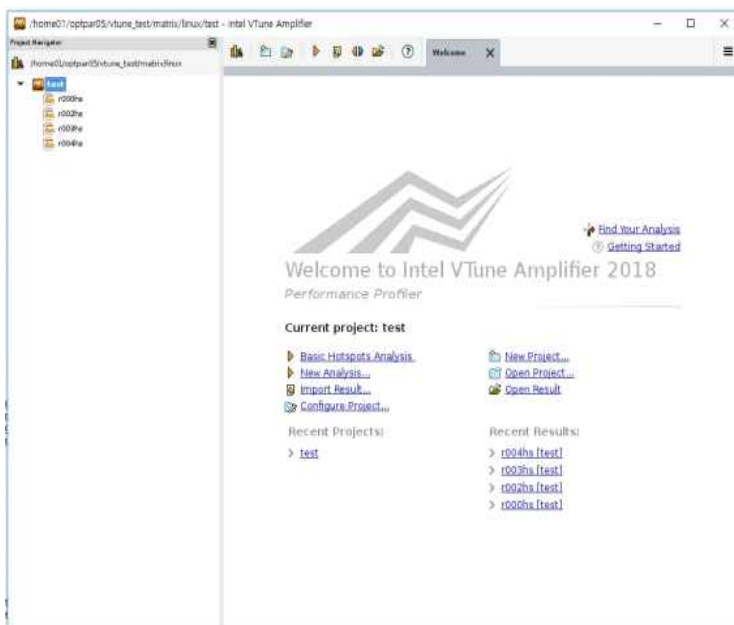
- 위와 같이 이미 컴파일 된 실행파일을 준비하여 명령어 형식에 맞게 실행 시 r000hs 디렉터리가 생성되는 것을 확인할 수 있다. 디렉터리가 생성된 것을 확인 후 리포트 생성을 위한 명령을 실행하면 아래와 같은 결과를 출력된다.

```
$ amplxe-cl -report hotspots /home01/$USER/debugger/test/r000hs
amplxe: Using result path `/home01/$USER/debugger/test/r000hs'
amplxe: Executing actions 75 % Generating a report Function CPU Time
CPU Time:Effective Time CPU Time:Spin Time CPU Time:Overhead Time
Module Function (Full) Source File Start Address
-----
amplxe: Executing actions 100 % done
```

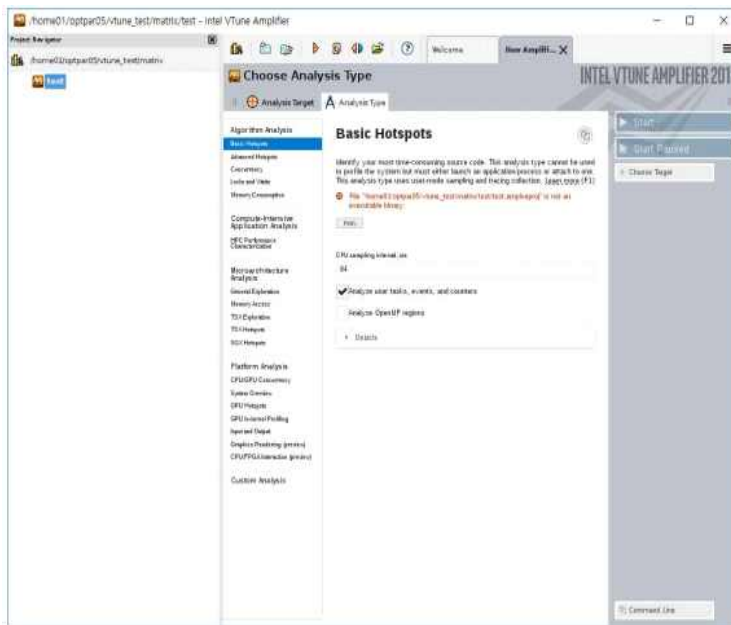
- GUI 사용 결과 확인 방법
  - Intel vtune Amplifier는 GUI 모드 역시 지원한다. 여기서는 GUI를 이용한 결과 확인 방법만 설명한다.
  - 사용자의 데스크탑에서 xming 실행

```
$ amplxe-gui
```

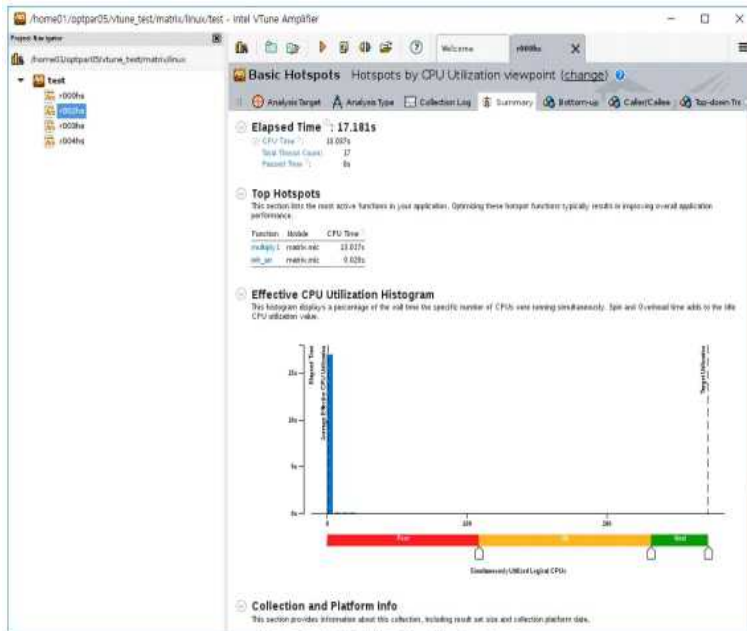
- 아래의 화면에서 New Analysis 버튼을 클릭한다.



- 아래와 같은 화면이 나타나면 cpu수와 체크 확인 후 Start 버튼을 클릭하면 분석이 시작된다.



- 완료되면 다음과 같이 분석 결과가 여러 개의 탭으로 요약되어 나타난다.



The screenshot shows the 'Basic Hotspots' window in VTune Amplifier, displaying a detailed table of CPU utilization analysis results. The table is organized into columns for 'Function', 'CPU Time', 'Module', 'Function ID', 'Callers', 'CPU Time', and 'CPU Time'. The data is filtered to show the top 100 results.

Function	CPU Time	Module	Function ID	Callers	CPU Time	CPU Time
main_thread	100.0%	libc.so.1	main_thread	main_thread	100.0%	100.0%
__clone	100.0%	libc.so.1	__clone	main_thread	100.0%	100.0%
ThreadFunction	100.0%	libc.so.1	ThreadFunction	main_thread	100.0%	100.0%
multiply_1	13.027s	matrix.mtc	multiply_1	main_thread	13.027s	13.027s
__start	0.027s	libc.so.1	__start	main_thread	0.027s	0.027s
__libc_start_main	0.027s	libc.so.1	__libc_start_main	main_thread	0.027s	0.027s
mat_get	0.027s	matrix.mtc	mat_get	main_thread	0.027s	0.027s
main	0.027s	matrix.mtc	main	main_thread	0.027s	0.027s



### 3. 프로파일러 Cary-Pat 사용예제

- 본 시스템에서 프로파일러인 CaryPat을 사용하기 위하여 아래와 같이 아키텍처 등 환경을 설정한 후 예제를 진행하였다.

```
$ module load craype-mic-knl or craype-x86-skylake
$ module load perftools-base/6.5.2 perftools/6.5.2
$ module load PrgEnv-cray/1.0.2 cce/8.6.3
```

- 먼저 예제에 사용될 test.c 파일을 컴파일한다.

```
$ cc test.c
```

- 위 실행 결과로 a.out이라는 실행파일이 생성된다.
- CaryPat으로 분석을 위하여 pat\_build를 이용하여 새로운 실행 파일을 생성한다.

```
$ pat_build -O apa a.out
```

- 위의 실행 결과로 a.out+pat 라는 파일이 생성된다.
- 생성된 실행파일은 MPI로 작성된 파일이므로 mpirun으로 실행한다.

```
$ mpirun -np 4 ./a.out+pat
```

- 실행을 완료하면 a.out+pat+378250-3s 디렉터리가 생성되고 디렉터리 안에 xf-files/002812.xf 파일이 생긴다.

```
$ pat_report a.out+pat+378250-3s
```

- 위와 같이 pat\_report를 실행하면 a.out+pat+378250-3s 디렉터리 안에 .ap2 파일과 .apa 파일이 생성된다.

```
$ pat_build -O a.out+pat+378250-3s/build-options.apa
```

- .apa 파일을 이용하여 다시 실행파일을 생성하면 a.out+apa 이름의 파일이 생성된다.
- 이렇게 생성된 a.out+apa 파일을 실행하면

```
$ mpirun -np 4 ./a.out+apa
```

- a.out+pat+378250-3t에 새로운 xf 파일이 생성된다.
- pat\_report를 다시 사용하여 새로운 데이터를 처리한다.

```
$ pat_report a.out+apa+378250-3t
```

- 위와 같이 실행하여 ap2 파일과 tracing 리포트를 생성한다.
- 위와 같이 수집된 데이터를 시각화하는 방법으로 app2를 제공한다.

```
$ app2 a.out+apa+378250-3t
```

- 아래와 같이 시각화가 가능하다.





## I -4. 스케줄러(PBS)를 통한 작업 실행

5호기 누리온 시스템의 작업 스케줄러는 Portable Batch System (이하 PBS)을 사용한다. 이 장에서는 스케줄러를 통해 작업 제출하는 방법 및 관련 명령어들을 소개한다. 사용자가 작업 제출 시 사용할 수 있는 큐는 정해져 있으며, 큐별로 사용자별 최대 제출할 수 있는 작업의 수는 제한이 있고, 이 값은 시스템의 부하 정도에 따라 변동될 수 있다.

누리온 시스템은 **배타적 노드 할당 정책**을 기본적으로 적용하여, 한 노드에 한 사용자의 작업만이 실행될 수 있도록 보장한다. 이는 공유 노드 정책을 적용할 경우 종종 발생할 수 있는 사용자 어플리케이션의 심각한 성능 저하를 예방하기 위함이다. 그러나 상용 SW를 사용할 수 있는 큐의 경우는 노드의 규모가 크지 않아서 효율적인 자원 활용을 위해 공유 노드 정책을 적용한다.

사용자 작업은 로그인 노드를 통해서만 제출이 가능하며 또한, 일반 사용자가 계산노드에 직접적으로 접근할 수 없다.

또한, 사용자 작업은 `/scratch/$USER`에서만 제출이 가능하다.

## 가. 큐 구성

- Commercial 큐 (상용SW 수행을 위한 큐)와 디버깅을 위한 debug큐는 공유 노드 정책이 적용되어 가용 자원(CPU core) 범위 내에서 노드 당 여러 개의 작업이 배치되며, 나머지 큐에서는 배타적 노드 정책으로 노드 당 하나의 작업만이 배치된다.
- 작업 큐
  - 일반사용자가 사용할 수 있는 큐와 사용자별 제출 가능 작업 개수는 다음 표와 같다. (2021년 4월 기준)

큐 구분	큐 이름	할당 노드 수	Total CPU core 수	Wall Clock Limit (시간)	작업제출개수제한*		리소스 점유제한**	
					사용자별 최대제출 작업개수	사용자별 최대실행 작업개수	작업별 노드 점유개수	
							최대	최소
KNL	exclusive	2600	176,800	unlimited	100	100	2600	1
	normal	4970	337,960	48	40	20	4970	
	long	300	20,400	120	20	10	300	
	flat	180	12,240	48	20	10	180	
	debug	20	1,360	12	2	2	2	
SKL	commercial	118	4,720	48	6	2	118	
	norm_skl	118	4,720		10	5		

※ 노드 구성은 시스템 부하에 따라 시스템 운영 중에 조정될 수 있음.

(showq 명령어와 motd를 통해 노드 구성과 최대 작업가능개수를 수시로 확인할 수 있음)

### 1. 큐별 설명

구분	큐명	특징
KNL	exclusive	R&D 혁신지원 프로그램 중 거대 도전연구 및 집단연구 분야 지원을 위한 전용 자원 큐
	normal	무상 서비스 사용자(창의연구 분야, 국가전략 분야, 혁신지원 분야)와 유상 서비스 사용자를 위한 일반 자원 큐
	long	장기간 작업 수행이 필요한 큐로 최대 120시간(5일) 동안 사용할 수 있는 일반 자원 큐
	flat	메모리 모드가 Flat으로 MCDRAM(16GB)과 DDR4를 지정하여 사용할 수 있으며 최대 102GB까지 메모리 사용이 가능한 일반 자원 큐(기타사용법 '다' 플랫폼노드 사용법 참조)
	debug	KNL 노드의 디버깅을 위한 큐로 공유노드 정책이 적용되어 있는 일반 자원 큐. 인터랙티브 작업을 통해 디버깅 가능
SKL	commercial	상용 어플리케이션 수행을 위한 일반 자원 큐로 공유노드 정책 적용 (*Gaussian을 제외한 상용SW는 commercial 큐 사용을 원칙으로 함)
	norm_skl	SKL 노드의 일반 자원 큐

## 2. 작업 제출 개수 제한

- 사용자별 최대 제출 작업 개수 : 초과하여 작업을 제출한 경우 제출 시점에 오류 발생함.
- 사용자별 최대 실행 작업 개수 : 초과하여 작업을 제출한 경우 이전 작업이 끝날 때까지 기다려야 함.

## 3. 리소스 점유 제한

- 작업별 노드 점유 개수 (최대|최소) : 단일 작업에서 점유 노드 수가 최소~최대 범위를 벗어나는 경우 제출 시점에 오류가 발생한다. 사용자의 대기 및 실행 중인 작업의 점유 노드 개수와는 무관함.

## 4. KNL 메모리 모드에 따른 큐 구분(Cluster 모드는 전부 Quadrant)

- exclusive, normal, long, debug 큐는 Cache 모드(MCDRAM을 L3캐시로 사용), flat큐는 Flat 모드(MCDRAM을 DDR4와 같이 RAM으로 사용)로 설정되어 있음.
- 시스템 보호를 위해 Cache 모드는 최대 가용 메모리 82GB, Flat 모드는 102GB 제한

## 5. Hyperthread off 설정으로 KNL 사용 시 노드 당 최대 68 스레드, SKL 사용 시 노드 당 최대 40 스레드 사용 가능

## 나. 작업 제출 및 모니터링

### 1. 배치 작업 제출

- 작업 스크립트 작성 및 예시
  - 작업 스크립트 작성 시 아래 필수 키워드, 아래 **작업 스크립트 예시**와 별첨1 **작업 스크립트 주요 키워드**를 참고하여 작성 후 제출한다. 또한, 작업 스크립트 예시 파일은 [/apps/shell/home/job\\_examples](/apps/shell/home/job_examples)에서 확인할 수 있다.

- PBS 작업 스케줄러 필수 옵션

필수 옵션	설명
#PBS -V	현재 환경변수 유지
#PBS -N	작업 이름 설정
#PBS -q	작업을 실행할 큐
#PBS -l	작업에 사용할 리소스 설정
#PBS -A	사용 프로그램 정보(통계 목적)

- 자원 할당 변수 키워드

사용하고자 하는 자원을 select, ncpus, ompthreads, miprocs, walltime 등의 키워드로 지정

키워드	설명
select	사용할 노드 수
ncpus	사용할 CPU 코어 수 ( $\geq$ 한 노드당 프로세스 수 * 스레드 수)
mpiprocs	사용할 한 노드당 프로세스 수
ompthreads	사용할 OMP 스레드 수
walltime	작업 실행 시간

※ 누리온 사용자 편익 증대를 위한 자료 수집의 목적으로, 아래와 같이 PBS 옵션을 통한 사용 프로그램 정보 작성을 의무화한다. 즉, 사용하는 어플리케이션에 맞게 PBS의 -A 옵션을 아래 표를 참조하여 반드시 기입한 후 작업을 제출해야 한다. (2019년 4월부터 적용)

※ 어플리케이션 구분 추가는 주기적으로 수집된 사용자 요구에 맞추어 진행됩니다. 추가를 원하시면 [consult@ksc.re.kr](mailto:consult@ksc.re.kr)로 해당 어플리케이션에 대한 추가 요청을 해주시기 바랍니다.

[Application별 PBS 옵션 이름표]

Application종류	PBS 옵션 이름	Application종류	PBS 옵션 이름
ANSYS (CFX, Fluent)	<b>ansys</b>	VASP	<b>vasp</b>
Abaqus	<b>abacus</b>	Gromacs	<b>gromacs</b>
Nastran(MSC One)	<b>nastran</b>	Amber	<b>amber</b>
Gaussian	<b>gaussian</b>	LAMMPS	<b>lammps</b>
OpenFoam	<b>openfoam</b>	NAMD	<b>namd</b>
WRF	<b>wrf</b>	Quantum Espresso	<b>qe</b>
CESM (CAM 포함)	<b>cesm</b>	QMCpack	<b>qmc</b>
MPAS	<b>mpas</b>	BWA	<b>bwa</b>
ROMs	<b>roms</b>	SIESTA	<b>siesta</b>
MOM	<b>mom</b>	in-house code	<b>inhouse</b>
TensorFlow	<b>tf</b>	Caffe	<b>caffe</b>
PyTorch	<b>pytorch</b>	Qchem	<b>qchem</b>
grims	<b>grims</b>	Charmm	<b>ramses</b>
cp2k	<b>cp2k</b>		<b>charmm</b>
그 외 applications	<b>etc</b>		

※ 예: VASP 사용자의 경우 PBS 프로그램 작업 스크립트에 #PBS -A vasp을 추가 작성

• 환경변수

환경변수	설명
PBS_JOBID	Job에 할당되는 식별자
PBS_JOBNAME	사용자에 의해 제공되는 Job 이름
PBS_NODEFILE	작업에 할당된 계산노드들의 리스트를 포함하고 있는 파일 이름
PBS_O_PATH	제출 환경의 경로 값
PBS_O_WORKDIR	qsub이 실행된 절대경로 위치
TMPDIR	Job을 위해 지정된 임시 디렉터리

PBS에서 배치 작업을 수행하기 위해서는 위에서 설명된 PBS 키워드들을 사용하여 작업 스크립트 파일을 작성해야 한다.

※ /apps/shell/home/job\_examples 에서 작업제출 스크립트 예제 파일을 복사하여 사용 가능



- Serial 프로그램 작업 스크립트 작성 예제(serial.sh)

```
#!/bin/sh
#PBS -N serial_job
#PBS -V
#PBS -q normal
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=1:ncpus=1:mpiprocs=1:ompthreads=1
#PBS -l walltime=04:00:00
#PBS -m abe # 작업 이메일 알림 옵션
#PBS -M abc@def.com # 수신할 메일 주소

cd $PBS_O_WORKDIR

module purge
module load craype-mic-knl

./test.exe
```

※ 1노드 점유 순차, 사용 예제

※ 위 예제와 같이 #PBS -m, #PBS -M 옵션을 사용하여 작업을 제출하는 경우 작업이 실행될 때와 완료 시, 그리고 작업이 중단되는 경우에도 abc@def.com로 이메일 발송

- OpenMP 프로그램 작업 스크립트 작성 예제(openmp.sh)

```
#!/bin/sh
#PBS -N openmp_job
#PBS -V
#PBS -q normal
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=1:ncpus=64:mpiprocs=1:ompthreads=64
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

module purge module load craype-mic-knl

./test_omp.exe
```

※ 1노드 점유, 노드 당 64 스레드(총 64 OpenMP 스레드) 사용 예제

- MPI (IntelMPI)프로그램 작업 스크립트 작성 예제(mpi.sh)

```
#!/bin/sh
#PBS -N IntelMPI_job
#PBS -V
#PBS -q normal#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=4:ncpus=64:mpiprocs=64
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

module purge
module load craype-mic-knl intel/18.0.3 impi/18.0.3

mpirun ./test_mpi.exe
```

※ 4노드 점유, 노드 당 64 프로세스(총 256 MPI 프로세스) 사용 예제

- MPI (OpenMPI)프로그램 작업 스크립트 작성 예제(mpi.sh)

```
#!/bin/sh
#PBS -N OpenMPI_job
#PBS -V
#PBS -q normal
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=4:ncpus=64:mpiprocs=64
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

module purge
module load craype-mic-knl gcc/7.2.0 openmpi/3.1.0

mpirun ./test_mpi.exe
```

※ 4노드 점유, 노드 당 64 프로세스(총 256 MPI 프로세스) 사용 예제

- MPI (Mvapich2) 프로그램 작업 스크립트 작성 예제(mpi\_mvapich2.sh)

```
#!/bin/sh
#PBS -N mvapich2_job
#PBS -V
#PBS -q normal
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=4:ncpus=64:mpiprocs=64:omphthreads=1
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

module purge
module load craype-mic-knl intel/18.0.3 mvapich2/2.3.1

TOTAL_CPUS=$(wc -l $PBS_NODEFILE | awk '{print $1}')

mpirun_rsh -np ${TOTAL_CPUS} -hostfile $PBS_NODEFILE ./test_mpi.exe
```

※ 4노드 점유, 노드 당 64 프로세스(총 256 MPI 프로세스) 사용 예제

※ mpirun 으로도 적은 규모의 노드에서는 실행이 가능하나, 많은 노드를 사용하는 작업의 경우 작업 전개가 정상적으로 진행되지 않을 수 있으니 위 예시와 같이 mpirun\_rsh 으로 실행 권장

- Hybrid(IntelMPI + OpenMP) 프로그램 작업 스크립트 작성 예제(hybrid\_intel.sh)

```
#!/bin/sh
#PBS -N hybrid_job
#PBS -V
#PBS -q normal
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=4:ncpus=64:mpiprocs=2:omphreads=32
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

module purge
module load craype-mic-knl intel/18.0.3 impi/18.0.3

mpirun ./test_mpi.exe
```

※ 4노드 점유, 노드 당 2 프로세스, 프로세스 당 32 스레드(총 8 MPI 프로세스, 256 OpenMP 스레드) 사용 예제

- Hybrid(openMPI + OpenMP) 프로그램 작업 스크립트 작성 예제(hybrid\_openmpi.sh)

```
#!/bin/sh
#PBS -N hybrid_job
#PBS -V
#PBS -q normal
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=4:ncpus=64:mpiprocs=2:omphreads=32
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

module purge
module load craype-mic-knl gcc/7.2.0 openmpi/3.1.0

mpirun --map-by NUMA:PE=34 ./test_mpi.exe
```

※ 4노드 점유, 노드 당 2 프로세스, 프로세스 당 32 스레드(총 8 MPI 프로세스, 256 OpenMP 스레드) 사용 예제

- Hybrid(Mvapich2 + OpenMP) 프로그램 작업 스크립트 작성 예제(hybrid\_mvapich2.sh)

```
#!/bin/sh
#PBS -N hybrid_job
#PBS -V
#PBS -q normal
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=4:ncpus=64:mpiprocs=2:ompthreads=32
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

module purge
module load craype-mic-knl intel/18.0.3 mvapich2/2.3.1

TOTAL_CPUS=$(wc -l $PBS_NODEFILE | awk '{print $1}')

mpirun_rsh -np ${TOTAL_CPUS} -hostfile $PBS_NODEFILE \
OMP_NUM_THREADS=$OMP_NUM_THREADS ./test_mpi.exe
```

※ 4노드 점유, 노드 당 2 프로세스, 프로세스 당 32 스레드(총 8 MPI 프로세스, 256 OpenMP 스레드) 사용 예제

※ mpirun 으로도 적은 규모의 노드에서는 실행이 가능하나, 많은 노드를 사용하는 작업의 경우 작업 전개가 정상적으로 진행되지 않을 수 있으니 위 예시와 같이 mpirun\_rsh 으로 실행 권장

- 작성한 작업 스크립트 제출 예시

```
$ qsub mpi.sh
```

※ mpi.sh 파일은 예시로, 작성한 작업 스크립트 파일을 이용하여 작업을 제출

- PBS 배치 작업을 수행하는 경우, 작업 중 STDOUT(표준 출력)과 STDERR(표준 에러)를 시스템 디렉터리의 output에 저장하였다가 작업 완료 후 사용자 작업 제출 디렉터리로 복사한다. 기본적으로, 작업 완료 시까지 작업 관련 내용을 확인할 수 없으나 다음 키워드를 추가하면 확인 가능하다.

- PBS에 의해 생성되는 STDOUT / STDERR를 작업 실행 중 확인할 수 있는 키워드 (/home01에 파일 생성)

```
#PBS -W sandbox=PRIVATE
```

- 리눅스의 Redirection 기능을 사용하여 작업 실행 확인

```
./test.exe 1>stdout 2>stderr
```

- 작업 이메일 알림 지정

```
$ qsub -m -M
ex) qsub -m abe -M abc@def.com hello_world.sh
```

옵션	설명
a	job 중단 시(기본값)
b	job 시작 시
e	job 실행 완료 시
n	이메일 알림을 받지 않음

## 2. 인터랙티브 작업 제출

인터랙티브 작업 제출의 경우 잡 스크립트 작성과는 달리 #PBS를 생략하고 -I -A 등의 옵션만 사용

※ 2시간 이상 미사용 시 타임아웃으로 작업이 종료되고 자원이 회수됨, 인터랙티브 작업의 walltime은 최대 12시간으로 고정됨

- 배치 스크립트 대신 "-I" 옵션 사용

```
$ qsub -I -l select=1:ncpus=68:omphthreads=1 -l walltime=12:00:00 \  
-q normal -A {PBS 옵션 이름}
```

- 인터랙티브 작업 제출 시 그래픽 환경 사용 (-X)

```
$ qsub -I -X -l select=1:ncpus=68:omphthreads=1 -l walltime=12:00:00 \  
-q normal -A {PBS 옵션 이름}
```

※ 여기서 -l select 이하 구문의 내용은 사용자 수요에 따라 변경하여 사용하면 되나, 위 구문들(리소스 점유, 큐 이름, PBS 옵션 이름)은 반드시 작성하고 작업을 제출해야 함

- 인터랙티브 작업 제출 시 기존 환경변수 상속 (-V)

```
$ qsub -I -V -l select=1:ncpus=68:omphthreads=1 -l walltime=12:00:00 \  
-q normal -A {PBS 옵션 이름}
```

※ 위 예제에서 소문자 l과 대문자 l를 유념하여 참조

### 3. 작업 모니터링

작업 모니터링 관련 명령어들은 로그인 노드에서만 사용 가능.

- 큐 조회

```
$ showq
```

- 큐별 노드 유휴 자원 조회

```
$ pbs_status
```

- 현재 사용 계정으로 사용 가능한 큐리스트 조회

```
$ pbs_queue_check
```

- 작업 상태 조회

옵션	설명
qstat -u	자신의 작업만 조회
qstat -T	Q 상태의 작업들의 잔여 대기시간 조회
qstat -i	Q 및 H 상태의 작업만을 조회
qstat -f	작업 상세 조회
qstat -x	종료된 작업 조회

```
$ qstat <-a, -n, -s, -H, -x, ...>
```

```
ex> qstat
Job id      Name      User      Time Use S Queue
-----
0001.pbs    test_01   user01     8245:43: R normal
0002.pbs    test_02   user02     8245:44: R flat
0003.pbs    test_03   user03     7078:45 R norm_skl
0003.pbs    test_04   user04     1983:11: Q long
```

※ Job Id: 작업번호.pbs

※ Name: 작업 스크립트의 #PBS -N 값

※ S: 작업의 동작 상태를 표시(R-수행 중/ Q-대기/ H-일시정지/ E-오류)



- 작업 속성 조회

```
$ qstat -f

ex> qstat -f 0000
Job Id : 0000.pbs
  Job_Name = test
  Job_Owner = user@login01
resources_used.cput = 8245:43:20
resources_used.mem = 33154824kb
resources_used.ncpus = 64
resources_used.vmem = 999899940kb
resources_used.walltime = 128:54:21
  job_state = R
<생략>
```

- 작업 대기 시간 조회 (Estimated Start Time, 작업 시작 시각 추정)

```
$ qstat -i -w -T -u
```

※ 여기서, i는 H나 Q상태의 작업 리스트를 보여주는 플래그이고, -w는 자세한 정보를 옆으로 길게 출력하는 플래그임 (-w 플래그 사용 시 터미널 창을 옆으로 확장하여 정렬을 맞춰주면 정보 확인이 용이함)

```
$ qstat -i -w -T -u user01
```

pbs:

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	S	Est Start Time
0000001.pbs	user01	long	test1	--	1	32	--	48:00:00	Q	Today 11:38
0000003.pbs	user01	flat	test3	--	1	32	--	48:00:00	Q	Today 11:39
0000004.pbs	user01	flat	test4	--	1	32	--	48:00:00	Q	Today 11:39

※ 사용자 작업스크립트의 walltime 정보 취합으로 계산된 예상치임

## 다. 작업 제어

- 작업 삭제

```
$ qdel
```

- 작업 suspend/resume

```
$ qsig -s
```

## I -5. 사용자 지원

사용 중 문제가 생기거나 의문 사항이 있으면 KISTI 홈페이지 > 기술지원 > 상담을 통해 문의한다.

분야	연락처
기술지원 안내 (상담 및 최적병렬화)	<a href="https://www.ksc.re.kr/gsjw/gsjwan">https://www.ksc.re.kr/gsjw/gsjwan</a>
교육 문의	<a href="https://kacademy.kisti.re.kr">https://kacademy.kisti.re.kr</a>
시스템 활용 정보	<a href="https://docs-ksc.gitbook.io/blog/">https://docs-ksc.gitbook.io/blog/</a>

## II. 소프트웨어

### II-1. ANSYS FLUENT

본 문서는 5호기 시스템에서 ANSYS FLUENT 사용을 위한 기초적인 정보를 제공하고 있습니다. 따라서, ANSYS FLUENT 소프트웨어 사용법 및 5호기 시스템/리눅스 사용법 등은 포함되어 있지 않습니다. 5호기 시스템/리눅스 사용법에 대한 정보는 KISTI 홈페이지 (<https://www.ksc.re.kr>)의 자료실 내에 누리온 사용자 지침서 등을 참고하시기 바랍니다. (updated: 2023. 11)

#### 가. 사용 정책

- 한 사용자 아이디 당 최대 40개 CPU 코어로 프로그램 실행 가능합니다.  
(mpiproc : 사용할 한 노드 당 프로세스 수 기준)
- 한정된 라이선스를 슈퍼컴 사용자들이 함께 사용하므로, 사용 정책 기준을 초과하여 사용할 경우 해당 작업은 관리자가 강제 종료합니다.
- 로그인 노드(login01~04)의 과부하 방지를 위해 로그인 노드에서의 pre/post 작업은 허가하지 않습니다. 다만, VDI를 통해서 일부 pre/post 작업이 가능합니다.  
(참고 : 누리온 VDI 지침서 - <https://docs-ksc.gitbook.io/nurion-user-guide/undefined-2/appendix-9-how-to-use-vdi>)
- 2019년 3월 PM 이후(3월14일)부터 작업제출 스크립트에 "#PBS -A ansys" 옵션을 사용해야 합니다.

#### 나. 소프트웨어 설치 정보

##### 1. 설치 버전

- v145, v170, v181, v191, v192, v195, v201, v212, v221, v222, v231, v232, v241

##### 2. 설치 위치

- /apps/commercial/ANSYS/(version)/fluent

##### 3. 실행 파일 경로

- /apps/commercial/ANSYS/(version)/fluent/bin

※ 위 version 의 위치에 사용하기 원하는 fluent 버전, 즉, v145, v170, v181, v191, v192, v195, v201, v202, v221, v222, v231, v232, v241 중의 하나로 교체하기 바랍니다.

## 다. 소프트웨어 실행 방법

- 명령어 실행 전 환경설정을 위한 스크립트를 먼저 실행합니다.
- (예) fluent v181을 사용하려면, 아래와 같이 수행합니다.

```
$ module load fluent/v222
```

※ module 환경설정 모듈 파일이 존재합니다. 위의 예제를 참고하여 알맞은 환경설정을 하시면 됩니다.

- 다음과 같이 batch mode로 job을 실행하기 위한 명령어 입력

형식	fluent [version] [-help] [option]
version	2d (2차원, single precision) 3d (3차원, single precision) 2ddp (2차원, double precision) 3ddp (3차원, double precision)
option	-g (GUI 없이 실행) -i journal (지정한 journal file을 읽음) -g -i journal ( background 모드로 작업을 수행) -tx (프로세서의 수를 x개로 지정)

- 로그인노드에서 Interactive 방식의 실행은 CPU time이 10분으로 제한되어 있습니다.
- 장시간의 계산 작업은 스케줄러를 이용하여 작업을 제출해야 합니다.

### 1. FLUENT input(journal) file 예제

/apps/commercial/test\_samples/ANSYS/wst.in 파일을 참조하여 적절히 수정하시기 바랍니다.

```
(set! *cx-exit-on-error* #t)
rc wst.cas                # read case file
/solve/init/init          # initialize the solution
it 100                    # calculate 100 iterations
wd wst.dat                # write data file
exit                       # exit FLUENT
yes
```

## 2. 스케줄러 작업 스크립트 파일 작성

5호기 시스템에서는 로그인 노드에서 PBS Professional 이라는 스케줄러를 사용하여 작업을 제출해야 합니다. 5호기 시스템에서 PBS Professional 을 사용하는 예제 파일들이 아래의 경로에 존재하므로 사용자 작업용 파일을 만들 때 이를 참고하시기 바랍니다.

- 예제 파일
  - /apps/commercial/test\_samples/ANSYS/fluent\_v181.sh  
(단일노드에서 수행)
  - /apps/commercial/test\_samples/ANSYS/fluent\_v181\_multinode.sh  
(멀티 노드에서 수행)

※ 아래는 누리온 시스템에서의 작업제출 예제입니다. (단일노드에서 수행)

```
#!/bin/sh
#PBS -V
#PBS -N fluent_job                # job 이름 지정
#PBS -q commercial                # 큐 지정
#PBS -l select=1:ncpus=40:mpiprocs=40:ompthreads=1  # MPI 태스크 및 Threads 수 지정
#PBS -l walltime=04:00:00         # 예상 작업 소요 시간 지정
#PBS -A ansys

cd $PBS_O_WORKDIR

##### Do not edit #####
TOTAL_CPUS=$(wc -l $PBS_NODEFILE | awk '{print $1}')
#####

module purge
module load fluent/v181

fluent 3d -pethernet -t${TOTAL_CPUS} -g -i wst.in > wst.output
```

- 위에서 내용을 사용자가 적절히 수정해야 합니다.
- 2019년 3월 PM 이후(3월14일)부터 작업제출 스크립트에 "#PBS -A ansys" 옵션을 사용해야 합니다.
- 작업 제출은 스크래치 디렉터리에서만 가능합니다.
- 사용자별 스크래치 디렉터리는 /scratch/\$USER입니다.
- 스크래치 디스크는 작업 종료 후 일정 시간이 지나면 삭제되기 때문에, 작업이 완료된 경우 빠른 시일 내에 백업하시길 권장합니다.
- 기타 스케줄러에 관련된 명령어 및 사용법은 누리온 사용자 지침서를 참조하여 주십시오.

## 라. 작업 모니터링

### 1. 큐 조회

```
$ showq
```

### 2. 노드 상태 조회

```
$ pbs_status
```

### 3. 작업 상태 확인

```
- 현재 실행/대기 중인 작업 조회
$ qstat -u $USER

- 종료된 작업까지 조회
$ qstat -xu $USER

$ qstat <-a, -n, -s, -H, -x, ...>
ex> qstat
Job id Name User Time Use S Queue
-----
0001.pbcm test_01 user01 8245:43: R commercail
0002.pbcm test_03 user03 7078:45: R commercail
0003.pbcm test_04 user04 1983:11: Q commercail
```

### 4. 작업 제출 방법

- 예제 : 스크립트 파일 이름이 fluent.sh인 경우

```
$ qsub fluent.sh
```

### 5. 제출된 작업을 강제로 종료

- 사용 방법 : qdel {작업ID}
- 작업ID는 qsub 명령어 실행 시 제일 왼쪽에 표시 되는 정보입니다.  
(ex. 0001.pbcm test\_01 user01 8245:43: R norm\_cache)
- 예제 : 작업ID 가 0001.pbcm 인 경우

```
$ qdel 0001.pbs
```



## 마. 기타

ANSYS Fluent 버전별 Start Guide 문서

## II -2. ANSYS CFX

본 문서는 누리온 시스템에서 ANSYS CFX 소프트웨어 사용을 위한 기초적인 정보를 제공하고 있습니다. 따라서, ANSYS CFX 소프트웨어 사용법 및 누리온/리눅스 사용법 등은 포함되어 있지 않습니다. 5호기 시스템/리눅스 사용법에 대한 정보는 KISTI 홈페이지 (<https://www.ksc.re.kr>)의 자료실 내에 5호기 시스템 사용자 지침서 등을 참고하시기 바랍니다. (updated : 2023.04.)

### 가. 사용 정책

- 한 사용자 아이디 당 최대 40개 CPU 코어로 프로그램 실행 가능합니다.  
(mpiproc : 사용할 한 노드 당 프로세스 수 기준)
- 한정된 라이선스를 슈퍼컴 사용자들이 함께 사용하므로, 사용정책 기준을 초과하여 사용할 경우 해당 작업은 관리자가 강제 종료합니다.
- 시스템의 로그인 노드의 과부하 방지를 위해 pre/post 작업은 허가하지 않습니다.
- 2019년 3월 PM 이후(3월14일)부터 "#PBS -A ansys " 옵션이 없는 경우 작업제출이 되지 않습니다.

### 나. 소프트웨어 설치 정보

#### 1. 설치 버전

- v145, v170, v181, v191, v192, v195, v201, v212, v221, v222, v231, v241

#### 2. 설치 위치

- /apps/commercial/ANSYS/(version)/CFX

#### 3. 실행 파일 경로

- /apps/commercial/ANSYS/(version)/CFX/bin

※ 위 version의 위치에 사용하기 원하는 CFX 버전, 즉, v145, v170, v181, v191, v192, v195, v201, v221, v222, v231, v241 중의 하나로 교체하기 바랍니다.

## 다. 소프트웨어 실행 방법

### 1. 실행 방법

- 명령어 실행 전 환경설정을 위한 스크립트를 먼저 실행합니다.
- (예) cfx v181을 사용하려면, 아래와 같이 수행합니다.

```
$ module load cfx/v181
```

※ module 환경설정 파일이 존재합니다. 위의 예제를 참고하여 알맞은 환경설정을 하시면 됩니다.

- 다음과 같이 batch mode로 job을 실행하기 위한 명령어 입력

형식	cfx5solve [option]
option	-def definition 파일 (또는 restart를 위한 result 파일) 지정 -parallel 병렬 모드로 실행 -par-local parallel run on the local host only -par-dist distributed parallel run -part <#partition> partitioning 모드로 solver를 실행 -parfile partitioning 정보 파일 지정 -help 사용가능 키워드 나열
예제	cfx5solve -def model.def cfx5solve -def model.def -par-local -partition 2 cfx5solve -def model.def -parallel -parfile model.par cfx5solve -def model.def -initial model_003.res -par-local -partition 2

- 로그인노드에서 Interactive 방식의 실행은 CPU time이 10분으로 제한되어 있습니다.
- 장시간의 계산 작업은 PBS Professional 이라는 스케줄러를 이용하여 작업을 제출해야 합니다.

### 2. 스케줄러 작업 스크립트 파일 작성

5호기 시스템에서는 로그인 노드에서 PBS Professional 이라는 스케줄러를 사용하여 작업을 제출해야 합니다. 5호기 시스템에서 스케줄러를 사용하는 예제 파일들이 아래의 경로에 존재하므로 사용자 작업용 파일을 만들 때 이를 참고하시기 바랍니다.

- 예제 파일
  - /apps/commercial/test\_samples/ANSYS/cfx\_v181.sh (단일노드에서 수행)
  - /apps/commercial/test\_samples/ANSYS/cfx\_v181\_multinode.sh (멀티 노드에서 수행)

※ 아래 예제는 5호기 시스템에서의 CFX에 대한 예제입니다. (단일노드에서 수행)

```
#!/bin/sh
#PBS -V
#PBS -N cfx_job                # job 이름 지정
#PBS -q commercial             # queue 지정
#PBS -l select=1:ncpus=40:mpiprocs=40:ompthreads=1    # 작업 청크 단위 지정
#PBS -l walltime=04:00:00      # 예상 작업 소요 시간 지정
#PBS -A ansys

cd $PBS_O_WORKDIR

##### Do not edit #####
TOTAL_CPUS=$(wc -l $PBS_NODEFILE | awk '{print $1}')
#####

module purge
module load cfx/v181

# CFX 명령 실행
cfx5solve -def StaticMixer.def -par-local -partition ${TOTAL_CPUS}
```

- 위에서 예제는 사용자가 적절히 수정해야 합니다.
- 2019년 3월 PM 이후(3월14일)부터 "#PBS -A ansys " 옵션이 없는 경우 작업제출이 되지 않습니다.
- 각 사용자별 홈디렉터리 disk quota 제한이 설정되어 있으므로, 스크래치 디렉터리에서 작업 수행을 권장합니다.
- 사용자별 스크래치 디렉터리는 /scratch/\$USER입니다.
- 스크래치 디스크는 작업 종료 후 일정 시간이 지나면 삭제되기 때문에, 작업이 완료된 경우 빠른 시일 내에 백업하시길 권장합니다.
- 기타 스케줄러에 관련된 명령어 및 사용법은 사용자 지침서를 참조하여 주십시오.

## 라. 작업 모니터링

### 1. 큐 조회

```
$ showq
```

### 2. 노드 상태 조회

```
$ pbs_status
```

### 3. 작업 상태 확인

- 현재 실행/대기 중인 작업 조회

```
$ qstat -u $USER
```

- 종료된 작업까지 조회

```
$ qstat -xu $USER
```

```
$ qstat <-a, -n, -s, -H, -x, ...>
```

```
ex> qstat
```

```
Job id Name User Time Use S Queue
```

```
-----  
0001.pbcm test_01 user01 8245:43: R commercail
```

```
0002.pbcm test_03 user03 7078:45: R commercail
```

```
0003.pbcm test_04 user04 1983:11: Q commercail
```

### 4. 작업 제출 방법

- 예제 : 스크립트 파일 이름이 cfx.sh인 경우

```
$ qsub cfx.sh
```

## 5. 제출된 작업을 강제로 종료

- 사용 방법 : qdel {작업ID}
- 작업ID는 qsub 명령어 실행 시 제일 왼쪽에 표시되는 정보입니다.  
(ex. 0001.pbcm test\_01 user01 8245:43: R norm\_cache)
- 예제 : 작업ID 가 0001.pbcm 인 경우

```
$ qdel 0001.pbs
```

## II -3. Abaqus

### Abaqus 지침서

본 문서는 누리온 시스템에서 ABAQUS 소프트웨어 사용을 위한 기초적인 정보를 제공하고 있습니다. 따라서, ABAQUS 소프트웨어 사용법 및 누리온/리눅스 사용법 등은 포함되어 있지 않습니다. 누리온/리눅스 사용법에 대한 정보는 KISTI 홈페이지 (<https://www.ksc.re.kr>)의 기술지원 > 지침서 내 누리온 사용자 지침서 등을 참고하시기 바랍니다.

### 가. 사용 신청 시 유의 사항

- 사용 가능 사용자 그룹 : 대학교/산업체(중소기업)/출연연구소 사용자
- 신규사용자의 경우 KSC 홈페이지 사용 신청서 작성 시 반드시 '사용할 애플리케이션'에 Abaqus를 명시하여야 합니다.
- 기존 사용자의 경우 account@ksc.re.kr로 계정, 소속, 아이디를 명기하고 신청을 해주십시오.
- 산업체(중소기업) 소속의 경우 account@ksc.re.kr로 관련 서류(중소기업 확인서, 벤처기업 확인서 등)를 제출해야 합니다.
- 사용 중 소속이 변경되어 해당 소속에 해당되지 않는 경우 반드시 account@ksc.re.kr로 알려주시기 바랍니다.

### 나. 사용 정책

- 사용자별 최대 40개 CPU 코어 수행 가능합니다.
- 한정된 라이선스를 슈퍼컴 사용자들이 함께 사용하므로, 사용정책 기준을 초과하여 사용할 경우 해당 작업은 관리자가 강제 종료합니다.
- 부득이하게 많은 라이선스가 필요할 경우, KISTI 홈페이지 (<https://www.ksc.re.kr>)를 통해 사전에 관리자와 협의해야 합니다.
- 작업 제출 전 "lic\_check" 명령에서 메뉴 선택을 통해 라이선스 상태를 확인한 다음 작업을 제출하시기 바랍니다.
- 로그인 노드(login01~04)의 과부하 방지를 위해 로그인 노드에서의 pre/post 작업은 허가하지 않습니다. 다만, VDI를 통해서 일부 pre/post 작업이 가능합니다.  
(참고 : 누리온 VDI 지침서 - <https://docs-ksc.gitbook.io/nurion-user-guide/undefined-2/appendix-9-how-to-use-vdi>)
- 작업제출 스크립트에 "#PBS -A abaqus" 옵션을 사용해야 합니다.

## 다. 소프트웨어 설치 정보

### 1. 설치 버전

- 6.14-6, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023

### 2. 설치 위치

- /apps/commercial/abaqus

### 3. 실행 파일 경로

- /apps/commercial/abaqus/Commands

## 라. 소프트웨어 실행 방법

### 1. 실행 방법

- 다음과 같이 batch mode로 job을 실행하기 위한 명령어 입력

형식	[command] [option]
Commands	abq6146 (6-14.6버전) abq2016hf19 (2016 버전) abq2017hf13 (2017 버전) abq2018hf5 (2018 버전) abq2019hf5 (2019 버전) abq2020hf4 (2020 버전) abq2021 (2021 버전) abq2022 (2022 버전) abq2023hf7 (2023버전)
예제 (6.14-6 버전)	abq6146 job=job_name [cpus=ncpus] [input=inp_file] [interactive] abq6146 job=job_name oldjob=old_job_name abq6146 help abq6146 information=environment abq6146 viewer [res=res_name]

- 파란 글씨 부분을 작업명, 사용 코어 수, 입력 파일명으로 수정 후 사용하시기 바랍니다.
- Interactive 방식의 실행은 CPU time이 20분으로 제한되어 있습니다.
- 장시간의 계산 작업은 PBS라는 스케줄러를 이용하여 작업을 제출해야 합니다.
- 작업 제출 전 "lic\_check" 명령을 실행하여 라이선스 상태를 확인 후 작업을 제출하시기 바랍니다.



## 2. 스케줄러 작업 스크립트 파일 작성

- 누리온 시스템에서는 로그인 노드에서 PBS라는 스케줄러를 사용하여 작업을 제출해야 합니다.
- 누리온 시스템에서 PBS를 사용하는 예제 파일들이 아래의 경로에 존재하므로 사용자 작업용 파일을 만들 때 이를 참고하시기 바랍니다.
- 예제 파일
  - /apps/commercial/test\_samples/abaqus/abaqus\_v6146.sh  
(단일노드에서 수행)
  - /apps/commercial/test\_samples/abaqus/abaqus\_v6146\_multinode.sh  
(멀티 노드에서 수행)

※ 아래 예제는 누리온 시스템에서의 ABAQUS 6.14-6 버전에 대한 예제입니다.  
(단일노드에서 수행)

```
#!/bin/sh

#PBS -V
#PBS -N abaqus_test # JOB의 이름을 지정
#PBS -q commercial # PBS 의 Queue 지정
#PBS -l select=1:ncpus=40:mpiprocs=40:ompthreads=1 # 연산에 사용할 자원 지정
#PBS -l walltime=04:00:00 # 예상 작업 소요시간 지정 (시:분:초)
#PBS -A abaqus
cd $PBS_O_WORKDIR
##### Do not edit #####
TOTAL_CPUS=$(wc -l $PBS_NODEFILE | awk '{
#####
cp /apps/commercial/abaqus/6146/6.14-6/SMA/site/abaqus_v6.env .
/apps/commercial/abaqus/Commands/abq6146 job=c2 cpus=${TOTAL_CPUS} int
```

- 위에서 파란색으로 표기된 부분은 사용자가 적절히 수정해야 합니다.
- 2019년 3월 PM 이후부터는 "#PBS -A abaqus" 옵션이 없는 경우 작업제출이 되지 않습니다.
- 작업 제출은 스크래치 디렉터리에서만 가능합니다.
- 사용자별 스크래치 디렉터리는 /scratch/\$USER입니다.
- 스크래치 디스크는 작업 종료 후 일정 시간(2023년 8월 현재 정책: 15일)이 지나면 삭제되기 때문에, 작업이 완료된 경우 빠른 시일 내에 백업하시길 권장합니다.
- 기타 PBS에 관련된 명령어 및 사용법은 누리온 사용자 지침서를 참조하시면 됩니다.

※ 아래 예제는 누리온 시스템에서의 ABAQUS 6.14-6 버전에 대한 예제입니다.  
(단일노드에서 threads mode 수행)

```
#!/bin/sh
#PBS -V
#PBS -N abaqus_test # JOB의 이름을 지정
#PBS -q commercial # PBS 의 Queue 지정
#PBS -l select=1:ncpus=40:mpiprocs=1:ompthreads=40 # 연산에 사용할 자원 지정
#PBS -l walltime=04:00:00 # 예상 작업 소요시간 지정 (시:분:초)
#PBS -A abaqus
cd $PBS_O_WORKDIR
cp /apps/commercial/abaqus/6146/6.14-6/SMA/site/abaqus_v6.env .
/applications/commercial/abaqus/Commands/abq6146 job=c2 cpus=${NCPUS} mp_mode=threads int
```

- 위에서 파란색으로 표기된 부분은 사용자가 적절히 수정해야 합니다.
- 2019년 3월 PM 이후 부터는 "#PBS -A abaqus" 옵션이 없는 경우 작업제출이 되지 않습니다.
- 작업 제출은 스크래치 디렉터리에서만 가능합니다.
- 사용자별 스크래치 디렉터리는 /scratch/\$USER입니다.
- 스크래치 디스크는 작업 종료 후 일정 시간(2023년 8월 현재 정책: 15일)이 지나면 삭제되기 때문에, 작업이 완료된 경우 빠른 시일 내에 백업하시길 권장합니다.
- 기타 PBS에 관련된 명령어 및 사용법은 누리온 사용자 지침서를 참조하시면 됩니다.

※ 아래 예제는 누리온 시스템에서의 ABAQUS 6.14-6 버전에 대한 예제입니다.  
(멀티노드에서 수행)

```
#!/bin/sh
#PBS -V
#PBS -N abaqus_test # JOB의 이름을 지정
#PBS -q commercial # PBS 의 Queue 지정
#PBS -l select=2:ncpus=40:mpiprocs=20:omphreads=1 # 연산에 사용할 자원 지정
#PBS -l walltime=04:00:00 # 예상 작업 소요시간 지정 (시:분:초)
#PBS -A abaqus
cd $PBS_O_WORKDIR
##### Do not edit #####
export MPI_IC_ORDER=TCP
TOTAL_CPUS=$(wc -l $PBS_NODEFILE
```

- 위에서 빨간색으로 표기된 부분은 수정하지 마세요.
- 2019년 3월 PM 이후 부터는 "#PBS -A abaqus" 옵션이 없는 경우 작업제출이 되지 않습니다.
- 위에서 파란색으로 표기된 부분은 사용자가 적절히 수정해야 합니다.
- 작업 제출은 스크래치 디렉터리에서만 가능합니다.
- 사용자별 스크래치 디렉터리는 /scratch/\$USER입니다.
- 스크래치 디스크는 작업 종료 후 일정 시간(2023년 8월 현재 정책: 15일)이 지나면 삭제되기 때문에, 작업이 완료된 경우 빠른 시일 내에 백업하시길 권장합니다.
- 기타 PBS에 관련된 명령어 및 사용법은 누리온 사용자 지침서를 참조하시면 됩니다.

### 3. 작업 제출 방법

- 예제 : 스크립트 파일 이름이 abaqus.sh인 경우

```
$ qsub abaqus.sh
```

### 4. 작업 상태 확인

```
$ qstat (또는 qstat -u $USER)
```

### 5. 제출된 작업을 강제로 종료

- 사용 방법 : qdel
- 작업ID는 qstat 명령어 실행 시 제일 왼쪽에 표시되는 정보입니다.  
(ex. 1771476.pbs)
- 예제 : 작업ID 가 1771476.pbs 인 경우

```
$ qdel 1771476.pbs
```

### 6. PBS 상에서 사용 가능한 자원 확인

```
$ pbs_status
```

## 라. abaqus 수행 오류 관련

- 6월 정기 점검 시 Lustre 파일시스템의 I/O 성능 향상을 위한 기능이 /scratch 파일시스템에 적용되었습니다.
- 이로 인해 현재 abaqus 수행 시 아래와 같은 **\*\*[관련 오류 메시지]\*\***가 출력되면서 파일 접근에 대한 오류가 발생할 수 있습니다. (다만, 특정 작업의 경우 오류 메시지가 출력되지 않고 작업이 R 상태로 진행되지 않을 수 있습니다.)
- 아래 오류 메시지와 유사한 파일 접근 오류가 발생하는 경우, 다음의 절차를 먼저 진행하시고 작업을 제출하시기 바랍니다.

### [stripe count 설정 방법]

```
$ mkdir
$ lfs setstripe -c 4
$ cp *.inp # .inp 파일은 abaqus의 input 파일을 지칭합니다.
$ cp *.sh # .sh 파일은 abaqus 작업제출 스크립트를 지칭합니다.
$ cd
$ qsub
```

- \* : 입력 데이터가 위치하고 있으며, 출력 데이터가 기록될 디렉터리
- \* 하위 디렉터리를 생성하는 경우 동일한 작업을 수행하셔야 합니다.

※ restart 작업과 같이 기존 파일을 활용해야 하는 경우, \ (stripe count 설정을 수행한 디렉터리)로 cp 명령어를 이용하여 **기존 파일을 복사하여** 작업을 수행해주시기 바랍니다. \ (mv 명령어 사용 시 기존의 속성을 유지하기 때문에 오류가 발생할 수 있습니다.)

### [관련 오류 메시지]

- (1) **\*\*\*ERROR: \*\*\* Error: Extend failed (utl\_File: read)**
- (2) **NOTE: DUE TO AN INPUT ERROR THE ANALYSIS PRE-PROCESSOR HAS BEEN UNABLE TO INTERPRET SOME DATA. SUBSEQUENT ERRORS MAY BE CAUSED BY THIS OMISSION**
- (3) **terminate called after throwing an instance of 'utl\_FileExceptionExtend'**

ABAQUS/pre rank 0 received signal 6 (Aborted)

## II -4. NASTRAN

본 문서는 누리온 시스템에서 NASTRAN 소프트웨어 사용을 위한 기초적인 정보를 제공하고 있습니다. 따라서, NASTRAN 소프트웨어 사용법 및 누리온/리눅스 사용법 등은 포함되어 있지 않습니다. 누리온/리눅스 사용법에 대한 정보는 KISTI 홈페이지 (<https://www.ksc.re.kr>)의 자료실 내에 누리온 사용자 지침서 등을 참고하시기 바랍니다.

\* updated: 2020. 2.

### 가. 사용정책

- 한 사용자 아이디 당 최대 32개 CPU 코어 수행 가능합니다.
- 한정된 라이선스를 슈퍼컴 사용자들이 함께 사용하므로, 사용정책 기준을 초과하여 사용할 경우 해당 작업은 관리자가 강제 종료합니다.
- 부득이하게 많은 라이선스가 필요할 경우, KISTI 홈페이지(<https://www.ksc.re.kr>)를 통해 사전에 관리자와 협의해야 합니다.
- 작업 제출 전 "lic\_check" 명령에서 메뉴 선택을 통해 라이선스 상태를 확인한 다음 작업을 제출하시기 바랍니다.
- 누리온 시스템 로그인 노드의 과부하 방지를 위해 pre/post 작업은 허가하지 않습니다.
- 2019년 3월 PM 이후(3월14일)부터 작업제출 스크립트에 "#PBS -A nastran" 옵션을 사용해야 합니다.

### 나. 소프트웨어 설치 정보

#### 1. 설치 버전

- MSC ONE (NASTRAN) 20182
- MSC ONE (NASTRAN) 20191
- MSC ONE (NASTRAN) 20213

#### 2. 설치 위치

- 20182 : /apps/commercial/MSK/Nastran
- 20191 : /apps/commercial/MSK/MSK\_Nastran
- 20213 : /apps/commercial/MSK/MSK\_Nastran/2021.3

### 3. 실행 파일 경로 및 config 파일 경로

<20182>

- 실행 파일 경로: /apps/commercial/MSNastran/bin
- Conf 파일: /apps/commercial/MSNastran/conf/nast20182rc

<20191>

- 실행 파일 경로 : /apps/commercial/MSNastran/2019fp1/bin
- Conf 파일 : /apps/commercial/MSNastran/2019fp1/conf/nast20191rc

<20213>

- 실행 파일 경로 : /apps/commercial/MSNastran/2021.3/bin
- Conf 파일 : /apps/commercial/MSNastran/2021.3/conf/nast20213rc

## 다. 소프트웨어 실행 방법

### 1. 실행 명령

- 사용자의 홈 디렉터리 혹은 서버 디렉터리에서 MD Nastran R2.1을 사용하는 경우 "nastran inputfile"의 형태로 입력합니다.

### 2. 환경 설정

- Nastran 실행 관련 설정을 변경하기 위해서는 환경설정 파일을 수정해 주어야 합니다.
- 20182 버전 환경 설정 파일 경로 :  
/apps/commercial/MSNastran/conf/nast20182rc
- 20191 버전 환경 설정 파일 경로 :  
/apps/commercial/MSNastran/2019fp1/conf/nast20191rc
- 20213 버전 환경 설정 파일 경로 :  
/apps/commercial/MSNastran/2021.3/conf/nast20213rc
- 20161 버전 기본 환경 설정을 변경하고자 하는 경우, 위 경로의 "nast20182rc"를 홈 디렉터리나 작업 디렉터리로 복사하여 마침표를 파일 이름에 추가하여 ".nast20182rc"로 파일 이름을 변경한 다음 그 파일에서 필요한 부분을 수정해서 사용하기 바랍니다.

### [ nast20182rc 파일 예제 ]

```
auth=27500@vaccine02.ext
mode=i8
sdir=/scratch/applic
buffsize=65537
memory=max
$
mpiimp=intelmpi
ishellpath=$MSC_BASE/msc20182/actran/linux64/Actran_18.0.b.107480/bin:$MSC_BASE/msc20182/nast:
j.env=ACTRAN_PATH=$MSC_BASE/msc20182/actran/linux64
j.env=ACTRAN_PRODUCTLINE=$MSC_BASE/msc20182/actran/linux64/Actran_18.0.b.107480
j.env=ACTRAN_MPI=$MSC_BASE/msc20182/actran/linux64/Actran_18.0.b.107480/mpi/intelmpi
j.env=ACTRAN_AFFINITY=reset
j.env=ACTRAN_MPI_OPTS='-pmi-connect nocache -pmi-noaggregate -genvnone -print-rank-map'
$
scr=yes
$ End
```

※ 사용자 홈디렉터리로 복사할 때 아래와 같이 사용하면 됩니다.

(예) cp 명령 사용 예제

```
$ cp /apps/commercial/MSC/Nastran/conf/nast20182rc ../.nast20182rc
```

## 3. 실행 방법

- Interactive 방식의 실행은 CPU time이 20분으로 제한되어 있습니다.
- 장시간의 계산 작업은 PBS라는 스케줄러를 이용하여 작업을 제출해야 합니다.
- 작업 제출 전 "lic\_check" 명령을 실행하여 라이선스 상태를 확인 후 작업을 제출하시기 바랍니다.



#### 4. 스케줄러 작업 스크립트 파일 작성

- 누리온 시스템에서는 로그인 노드에서 PBS라는 스케줄러를 사용하여 작업을 제출해야 합니다.
- 누리온 시스템에서 PBS를 사용하는 예제 파일들이 아래의 경로에 존재하므로 사용자 작업용 파일을 만들 때 이를 참고하기 바랍니다.
- 예제 파일 : /apps/commercial/test\_samples/MS\_C\_NASTRAN/nast\_20182.cmd

※ 아래 예제는 누리온 시스템에서의 NASTRAN에 대한 예제입니다.

```
#!/bin/sh
#PBS -V
#PBS -N Nastran_job
#PBS -q commercial
#PBS -l select=1:ncpus=40:mpiprocs=1:ompthreads=40
#PBS -l walltime=04:00:00
#PBS -A nastran

cd $PBS_O_WORKDIR

/apps/commercial/MS_C_Nastran/bin/nast20182 car_mod_freq.bdf smp=$NCPUS batch=no sdir="."
```

- module로 등록되어 있지 않으며 예시와 같이 경로를 입력해 주어야 합니다.
- 2019년 3월 PM 이후(3월14일)부터 "#PBS -A nastran" 옵션이 없는 경우 작업제출이 되지 않습니다.
- 작업 제출은 /scratch/\$USER 디렉터리에서만 가능합니다.
- 사용자별 스크래치 디렉터리는 /scratch/\$USER 이며, 사용자 scratch 디렉터리 내에서 작업별 디렉터를 생성하여 작업을 제출/실행할 수 있습니다.
- 스크래치 디스크는 작업 종료 후 일정 시간이 지나면 이름이 변경되고 추후 삭제되기 때문에, 작업이 완료된 경우 빠른 시일 내에 백업하시길 권장합니다. (사용자 환경 > 사용자 파일시스템 및 쿼터 정책 참고)
- 기타 PBS에 관련된 명령어 및 사용법은 누리온 사용자 지침서를 참조하시면 됩니다.

## 5. 작업 제출 방법

예제 : 스크립트 파일 이름이 nastran.sh인 경우

```
$ qsub nastran.sh
```

## 6. 작업 상태 확인

```
$ qstat (또는 qstat -u $USER)
```

## 7. 제출된 작업을 강제로 종료

- 사용 방법 : qdel {작업ID}
- 작업ID는 qstat 명령어 실행 시 제일 왼쪽에 표시되는 정보입니다.  
(ex. 1771476.pbs)
- 예제 : 작업ID 가 s1771476.pbs 인 경우

```
$ qdel 1771476.pbs
```

## 8. PBS 상에서 사용 가능한 자원 확인

```
$ pbs_status
```

## II -5. 가우시안16(Gaussian16) LINDA

본 문서는 누리온 시스템에서 가우시안 소프트웨어 사용을 위한 기초적인 정보를 제공하고 있습니다. 따라서, 가우시안 소프트웨어 사용법 및 누리온/리눅스 사용법 등은 포함되어 있지 않습니다. 누리온/리눅스 사용법에 대한 정보는 KISTI 홈페이지(<https://www.ksc.re.kr>)의 자료실 내에 누리온 사용자 지침서 등을 참고하시기 바랍니다.

\* updated: 2019. 3.

### 가. 가우시안 소개

가우시안은 에너지, 분자구조 및 진동주파수를 예측하는 분자 모델링 패키지이며, 화학, 물리, 생명과학, 공학 분야 연구자를 위한 프로그램입니다.

자세한 사항은 가우시안 사의 홈페이지를 통해 얻을 수 있습니다.

홈페이지 주소: <http://gaussian.com>

### 나. 설치 버전 및 라이선스

- KISTI 슈퍼컴퓨팅센터는 가우시안 16/LINDA의 사이트 라이선스를 보유하고 있으며, 누리온 시스템에는 가우시안16 Rev. A03이 설치되어 있습니다.
- 가우시안16을 사용하기 위해서는 사용자의 계정이 가우시안 그룹(gauss group)에 등록되어야 합니다. 가우시안 그룹 등록은 KISTI 홈페이지 또는 [account@ksc.re.kr](mailto:account@ksc.re.kr)로 문의하시기 바랍니다.
- 내 계정이 가우시안 그룹에 속해있는지 확인하는 방법은 다음과 같습니다.

\$ id 사용자ID
-------------

- ※ 가우시안 그룹에 포함되어 있으면 출력 결과에 "1000009(gauss)"가 포함되어 있어야 합니다.
- 보안 문제로 사용자는 프로그램의 소스 코드에는 접근할 수 없고, 실행 파일과 기저함수(basis function)에만 접근할 수 있습니다. 실제로 프로그램을 사용하는 데는 아무런 지장이 없습니다.
  - 가우시안에 연동하여 사용하는 프로그램을 사용하기 위해서는 사전에 일부 소스 코드 혹은 쉘 파일에 대한 접근 권한이 필요하며 (예, Gaussrate) 이 경우 KISTI 홈페이지 또는 [account@ksc.re.kr](mailto:account@ksc.re.kr) 메일을 통해 요청하셔야 합니다.
  - HF 계산과 DFT 계산은 병렬로 수행할 수 있습니다.

## 다. 소프트웨어 실행 방법

### 1. 환경설정

가우시안16은 module 명령을 통하여 환경을 로드할 수 있습니다.

```
$ module load gaussian/g16.a03.linda
```

### 2. 스케줄러 작업 스크립트 파일 작성

- 누리온 시스템에서는 로그인 노드에서 PBS Pro라는 스케줄러를 사용하여 작업을 제출해야 합니다.
- 누리온 시스템에서 PBS를 사용하는 예제 파일들이 아래의 경로에 존재하므로 사용자 작업용 파일을 만들 때 이를 참고하시기 바랍니다.

※ 아래 예제는 누리온 시스템에서의 가우시안16 LINDA에 대한 예제입니다.

- 파일 위치: /apps/commercial/test\_samples/G16/g16\_Linda.sh

```
#!/bin/sh
#PBS -V
#PBS -N gaussian_test
#PBS -q norm_cache                # PBS의 queue를 지정
#PBS -l select=2:ncpus=64:mpiprocs=1:omphreads=64
#PBS -l walltime=01:00:00         # 예상 작업소요시간 지정 (시:분:초)
#PBS -A gaussian
cd $PBS_O_WORKDIR

module purge
module load gaussian/g16.a03.linda

nodes=`cat $PBS_NODEFILE`
nodes=`echo $nodes | sed -e 's/ /,/g'`
export GAUSS_SCRDIR="/scratch/${USER}"
export GAUSS_WDEF=${nodes}
export GAUSS_PDEF=$NCPUS

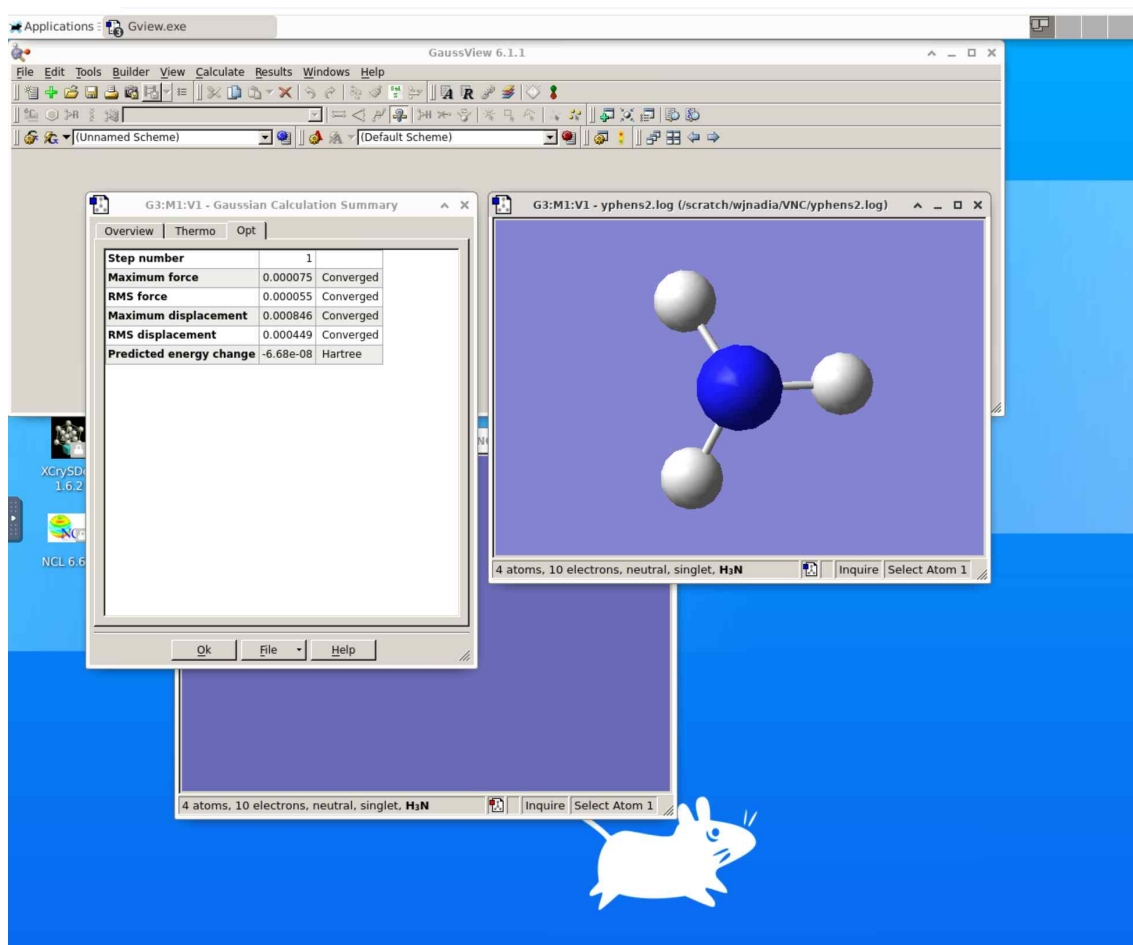
g16 test000.com

exit 0
```

- 2019년 3월 PM 이후(3월14일)부터 "#PBS -A gaussian" 옵션이 없는 경우 작업 제출이 되지 않습니다.
- GAUSS\_PDEF 변수는 %NProcShared 옵션과 동일하며, 입력파일에 %NProcShared 값이 있을 경우 해당 값이 적용됩니다.
  - 이때 GAUSS\_PDEF 또는 %NProcShared 옵션의 값은 누리온 KNL 계산노드는 68개 코어, SKL 계산 노드는 40개 코어가 장착되어져 있기 때문에 계산 노드에 맞게 기입하는 것이 안정적인 계산 성능이 발휘됩니다.
- 가우시안16 Rev. A03 버전에서 지원하는 최대 threads 수는 64개입니다. KNL 계산 노드를 이용하는 경우 64개까지만 사용 바랍니다.
- 작업 제출은 스크래치 디렉터리에서만 가능합니다.
- 사용자별 스크래치 디렉터리는 /scratch/\$USER입니다.
- 큐 이름은 누리온 사용자 지침서를 참조하여 설정하며, 일반적으로 normal로 설정해야 합니다.
- 가우시안 입력 파일을 PC에서 작성 후 FTP로 전송한다면, 반드시 **ascii mode로 전송해야만 합니다.**
- 기타 PBS에 관련된 명령어 및 사용법은 누리온 사용자 지침서를 참조하시면 됩니다.

### 3. GaussView 사용법

- GaussView는 GUI 기반 프로그램이기 때문에, MyKSC(웹 서비스 포털)의 VNC(원격 데스크톱)에서 실행할 수 있음 ([MyKSC VNC 사용법 참조](#))
- MyKSC VNC에서 GaussView는 할당된 계산 자원의 한계로 입력 데이터 생성 및 결과 분석 용도로 사용할 것을 권장하며, 본격적인 계산 작업은 배치 작업 스케줄러를 통해 실행해야 함



[MyKSC VNC에서 GaussView 실행화면]

## 라. 참고자료

- 가우시안을 처음으로 사용하고자 하는 사람은 다음의 책의 일독을 권합니다.
  - James B. Foresman and Aeleen Frisch,  
 "Exploring Chemistry with Electronic Structure Methods:  
 A Guide to Using Gaussian",  
[www.amazon.com](http://www.amazon.com), [www.bn.com](http://www.bn.com) 등의 온라인 서점에서 구매할 수 있고,  
<http://gaussian.com>에서도 직접 구매가 가능합니다.
- 가우시안에 관한 모든 정보는 Gaussian사의 홈페이지(<http://gaussian.com>)를 통해 얻을 수 있습니다.

## II -6. 가우시안16(Gaussian16)

본 문서는 누리온 시스템에서 가우시안 소프트웨어 사용을 위한 기초적인 정보를 제공하고 있습니다. 따라서, 가우시안 소프트웨어 사용법 및 누리온/리눅스 사용법 등은 포함되어 있지 않습니다. 누리온/리눅스 사용법에 대한 정보는 KISTI 홈페이지(<https://www.ksc.re.kr>)의 자료실 내에 누리온 사용자 지침서 등을 참고하시기 바랍니다.

\* updated : 2019. 3.

### 가. 가우시안 소개

가우시안은 에너지, 분자구조 및 진동주파수를 예측하는 분자 모델링 패키지이며, 화학, 물리, 생명과학, 공학 분야 연구자를 위한 프로그램입니다.

자세한 사항은 가우시안 사의 홈페이지를 통해 얻을 수 있습니다.

홈페이지 주소: <http://gaussian.com>

### 나. 설치 버전 및 라이선스

- KISTI 슈퍼컴퓨팅센터는 가우시안 16의 사이트 라이선스를 보유하고 있으며, 누리온 시스템에는 가우시안16 Rev. A03이 설치되어 있습니다.
- 가우시안16을 사용하기 위해서는 사용자의 계정이 가우시안 그룹(gauss group)에 등록되어야 합니다. 가우시안 그룹 등록은 KISTI 홈페이지 또는 [account@ksc.re.kr](mailto:account@ksc.re.kr)로 문의하시기 바랍니다.
- 내 계정이 가우시안 그룹에 속해있는지 확인하는 방법은 다음과 같습니다.

\$ id 사용자ID
-------------

- ※ 가우시안 그룹에 포함되어 있으면 출력 결과에 "1000009(gauss)"가 포함되어 있어야 합니다.
- 보안 문제로 사용자는 프로그램의 소스 코드에는 접근할 수 없고, 실행 파일과 기저함수(basis function)에만 접근할 수 있습니다. 실제로 프로그램을 사용하는 데는 아무런 지장이 없습니다.
  - 가우시안에 연동하여 사용하는 프로그램을 사용하기 위해서는 사전에 일부 소스 코드 혹은 쉘 파일에 대한 접근 권한이 필요하며 (예, Gaussrate) 이 경우 KISTI 홈페이지 또는 [account@ksc.re.kr](mailto:account@ksc.re.kr) 메일을 통해 요청하셔야 합니다.
  - HF 계산과 DFT 계산은 병렬로 수행할 수 있습니다.

## 다. 소프트웨어 실행 방법

### 1. 환경설정

가우시안16은 module 명령을 통하여 환경을 로드할 수 있습니다.

```
$ module load gaussian/g16.a03
```

### 2. 스케줄러 작업 스크립트 파일 작성

- 누리온 시스템에서는 로그인 노드에서 PBS Pro라는 스케줄러를 사용하여 작업을 제출해야 합니다.
- 누리온 시스템에서 PBS를 사용하는 예제 파일들이 아래의 경로에 존재하므로 사용자 작업용 파일을 만들 때 이를 참고하시기 바랍니다.

※ 아래 예제는 누리온 시스템에서의 가우시안16에 대한 예제입니다.

- 파일 위치: /apps/commercial/test\_samples/G16/g16.sh

```
#!/bin/sh
#PBS -V
#PBS -N gaussian_test
#PBS -q normal                # PBS의 queue를 지정
#PBS -l select=1:ncpus=64:mpiprocs=1:ompthreads=64
#PBS -l walltime=01:00:00      # 예상 작업소요시간 지정 (시:분:초)
#PBS -A gaussian

cd $PBS_O_WORKDIR

module purge
module load gaussian/g16.a03

export GAUSS_SCRDIR="/scratch/${USER}"
export GAUSS_PDEF=$NCPUS

g16 test000.com

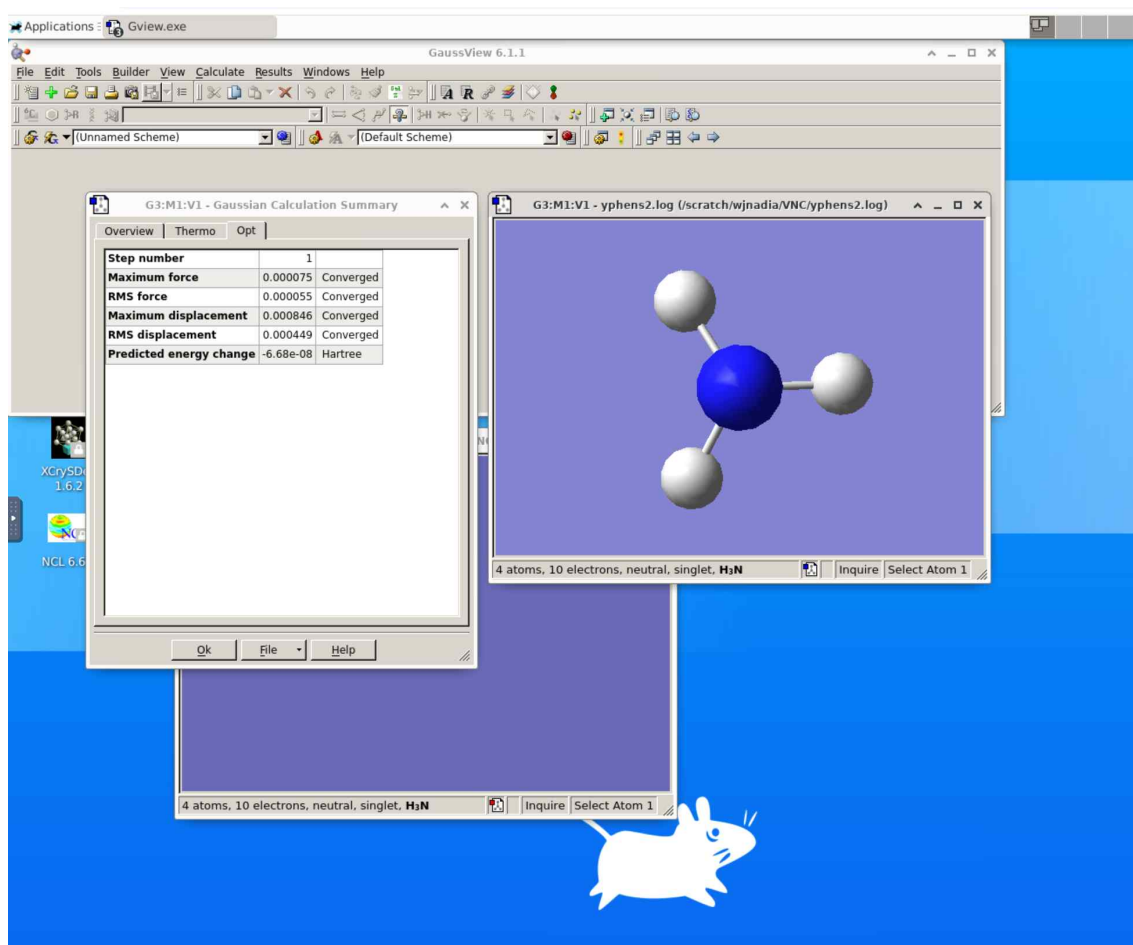
exit 0
```



- 2019년 3월 PM 이후(3월14일)부터 "#PBS -A gaussian" 옵션이 없는 경우 작업 제출이 되지 않습니다.
- GAUSS\_PDEF 변수는 %NProcShared 옵션과 동일하며, 입력파일에 %NProcShared 값이 있을 경우 해당 값이 적용됩니다.
  - 이때 GAUSS\_PDEF 또는 %NProcShared 옵션의 값은 누리온 KNL 계산 노드는 68개 코어, SKL 계산 노드는 40개 코어가 장착되어져 있기 때문에 계산 노드에 맞게 기입하는 것이 안정적인 계산 성능이 발휘됩니다.
- 가우시안16 Rev. A03 버전에서 지원하는 최대 threads 수는 64개입니다. KNL 계산 노드를 이용하는 경우 64개까지만 사용 바랍니다.
- 작업 제출은 스크래치 디렉터리에서만 가능합니다.
- 사용자별 스크래치 디렉터리는 /scratch/\$USER입니다.
- 큐 이름은 누리온 사용자 지침서를 참조하여 설정하며, 일반적으로 norm\_cache로 설정해야 합니다.
- 가우시안 입력 파일을 PC에서 작성 후 FTP로 전송한다면, 반드시 **ascii mode로 전송해야만 합니다.**
- 기타 PBS에 관련된 명령어 및 사용법은 누리온 사용자 지침서를 참조하시면 됩니다.
- Gaussian Multi-Node 사용을 위해서는 Gaussian16 LINDA 버전을 사용하세요.

### 3. GaussView 사용법

- GaussView는 GUI 기반 프로그램이기 때문에, MyKSC(웹 서비스 포털)의 VNC(원격 데스크톱)에서 실행할 수 있음 ([MyKSC VNC 사용법 참조](#))
- MyKSC VNC에서 GaussView는 할당된 계산 자원의 한계로 입력 데이터 생성 및 결과 분석 용도로 사용할 것을 권장하며, 본격적인 계산 작업은 배치 작업 스케줄러를 통해 실행해야 함



MyKSC VNC에서 GaussView 실행화면

## 라. 참고자료

- 가우시안을 처음으로 사용하고자 하는 사람은 다음의 책의 일독을 권합니다.
  - James B. Foresman and Aeleen Frisch,  
 "Exploring Chemistry with Electronic Structure Methods:  
 A Guide to Using Gaussian",  
 www.amazon.com, www.bn.com 등의 온라인 서점에서 구매할 수 있고,  
 http://gaussian.com에서도 직접 구매가 가능합니다.
- 가우시안에 관한 모든 정보는 Gaussian사의 홈페이지(http://gaussian.com)를 통해 얻을 수 있습니다.

### Ⅲ. 부록

#### Ⅲ-1. 작업 스크립트 주요 키워드

작업 스크립트 내에서 적절한 키워드를 사용하여 원하는 작업을 위한 자원 할당 방법을 명시해야 한다. 주요 키워드는 아래와 같으며, 사용자는 이들 중에서 몇 가지만 사용하여 작업 스크립트 파일을 작성할 수 있다.

옵션	형식	설명
-A		어카운팅 문자열 지정
-a	[[[YYYY]MM]DD]hhmm[.S]]	예약된 실행
-C	""	PBS 지시어의 접두어(ex>#PBS) 변경
-c	[c   c=numeric   w   w=   n   s   u]	Job의 체크포인트 주기 지정
-e		error 파일의 경로 지정
-h		Job 실행 지연(지연실행)
-I   -X		X11 포워딩 활성화된 Interactive Job
-J	]	Job 배열 정의
-j	[o   e]	output과 error 파일 병합
-k	[o   e]	실행 호스트에서 output과 error 파일 유지 여부
-l		Job 리소스 요청
-M		이메일 받는 사람 리스트 설정
-m		이메일 알람 지정
-N		Job 이름 지정
-o		output 파일의 경로 지정
-P		프로젝트 이름 지정
-p		Job 우선순위 설정
-q		서버나 큐의 이름 지정
-r	[Y   N]	재실행 가능여부 마킹
-S		사용할 스크립트 언어 지정
-u		Job의 사용자 ID 지정
-V		환경변수 내보내기
-v		환경변수들 확장
-W		Job의 속성 값 설정
-W block=		Job의 완료를 기다리는 qsub 요청

-W depend=		Job의 의존성 지정
-W group_list=		Job 소유자의 group ID 지정
-W pwd=	""	작업 당 password 방식
-W run_count=		작업이 실행되어지는 횟수 제어
-W sandbox=	[HOME   PRIVATE]	스테이징 디렉터리와 실행 디렉터리
-W stagein=		input 파일 스테이징
-W stageout=		output 파일 스테이징
-W unmask=		UNIX의 JOB umask 지정
-X		Interactive Job으로 부터의 X output
-z		함축된 작업 식별자

※ 상세 매뉴얼 : <https://pbsworks.com/pdfs/PBSUserGuide14.2.pdf> 참조

## III-2. Conda

아나콘다(Anaconda)는 PYTHON과 R 프로그래밍 언어로 된 과학 컴퓨팅(데이터 과학, 기계 학습 응용 프로그램, 대규모 데이터 처리, 예측 분석 등) 분야의 패키지들의 모음을 제공하는 배포판이다.

Anaconda 배포판은 1,200 만 명이 넘는 사용자가 사용하며 Windows, Linux 및 MacOS에 적합한 1400가지 이상의 인기 있는 데이터 과학 패키지를 포함한다.

Anaconda를 설치하기 위해서는 <https://www.anaconda.com> 웹사이트에서 자신의 OS(예: Windows, MacOS, Linux)에 맞는 배포판을 다운받아 설치하면 된다.

현재 Anaconda는 Python 3.7 기반의 버전과 Python 2.7 기반의 버전을 제공한다.

conda는 아나콘다에서 패키지 버전 관리를 위해 제공되는 어플리케이션이다.

Python 사용자들이 패키지 설치 시 가장 어려움을 겪는 의존성 문제를 conda를 활용함으로써 쉽게 해결할 수 있다.

본 문서는 KISTI 시스템에서 Python 사용자를 위하여 conda 패키지 활용하는 방법을 소개한다.

소개 페이지의 "/home01/userID" 는 테스트 계정의 홈 디렉터리로 자신에 맞는 경로로 적절히 변경해서 사용해야 한다.

### 가. Conda의 사용

- Miniconda는 <https://docs.conda.io/en/latest/miniconda.html> 사이트에서 각 OS에 맞는 버전을 다운받을 수 있고,
- Anaconda는 <https://www.anaconda.com/distribution/#download-section> 사이트에서 각 OS에 맞는 버전을 다운받을 수 있다.

명령어 모음	내용
clean	Remove unused packages and caches.
config	Modify configuration values in .condarc. This is modeled after the git config command.  Writes to the user .condarc file (/home01/userID/.condarc) by default.
create	Create a new conda environment from a list of specified packages.
help	Displays a list of available conda commands and their help strings.
info	Display information about current conda install.
init	Initialize conda for shell interaction. [Experimental]
install	Installs a list of packages into a specified conda environment.
list	List linked packages in a conda environment.
package	Low-level conda package utility. (EXPERIMENTAL)
remove	Remove a list of packages from a specified conda environment.
uninstall	Alias for conda remove.

run	Run an executable in a conda environment. [Experimental]
search	Search for packages and display associated information. The input is a MatchSpec, a query language for conda packages.  See examples below.
update	Updates conda packages to the latest compatible version.
upgrade	Alias for conda update

- Conda Initialize 방법

conda init 명령으로 홈 디렉터리의 .bashrc에 설정을 추가할 수 있다.

```
$ source /apps/applications/Miniconda/23.3.1/etc/profile.d/conda.sh
$ conda init
$ source ~/.bashrc
```

- Conda 경로 변경 방법

conda 환경, 패키지 경로는 기본적으로 홈 디렉터리로 설정되어 있으나, scratch와 같은 다른 경로로도 변경할 수 있다.

```
$ echo "export CONDA_ENVS_PATH=/scratch/$USER/.conda/envs" >> /home01/$USER/.bashrc
$ echo "export CONDA_PKGS_DIRS=/scratch/$USER/.conda/pkgs" >> /home01/$USER/.bashrc
$ source ~/.bashrc
```

## 나. Conda Environment 생성

- conda environment는 Python의 독립적인 가상 실행 환경을 만들어 패키지들의 버전 관리에 용이하다.
- "conda create -n [ENVIRONMENT]"를 이용하여 conda environment를 생성할 수 있다.
- 기본 값으로 conda path의 envs 아래 경로에 지정한 environment 이름으로 생성된다.

- 예제 -

```
※ conda initialize를 하지 않았다면 최초 1회 실행
$ source /apps/applications/Miniconda/23.3.1/etc/profile.d/conda.sh
$ conda init
$ source ~/.bashrc

$ conda create -n scikit-learn_0.21
Collecting package metadata: done
Solving environment: done

## Package Plan ##

  environment location: /home01/userID/.conda/envs/scikit-learn_0.21

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use:
# > conda activate scikit-learn_0.21
#
# To deactivate an active environment, use:
# > conda deactivate
#

$ conda activate scikit-learn_0.21
(scikit-learn_0.21) $
```

## 다. Conda Environment에 패키지 설치 및 확인

- `conda install [패키지명]`으로 패키지를 설치할 수 있다.
- `conda` 채널에 있는 패키지는 "`conda install -c [채널명] [패키지명]`"와 같이 설치할 수 있다.
- 위 항목("나. Conda environment 생성")에서 생성한 conda environment 경로 아래에 패키지들이 설치된다.

- 예제 -

```
$ source /apps/applications/Miniconda/23.3.1/etc/profile.d/conda.sh
$ conda activate scikit-learn_0.21
(scikit-learn_0.21) $ conda install scikit-learn
Collecting package metadata: done
Solving environment: done

## Package Plan ##

environment location: /home01/userID/.conda/envs/scikit-learn_0.21
added / updated specs:
- scikit-learn

The following packages will be downloaded:
package                        |          build          |
-----|-----
ca-certificates-2019.1.23      |          0          | 126 KB
...
wheel-0.33.1                   |        py37_0        | 39 KB
-----|-----
Total:                          277.6 MB

The following NEW packages will be INSTALLED:

blas                pkgs/main/linux-64::blas-1.0-mkl
...
zlib                pkgs/main/linux-64::zlib-1.2.11-h7b6447c_3

Proceed ([y]/n)? y
```



```
Downloading and Extracting Packages
setuptools-40.8.0 | 643 KB | ##### | 100%
...
openssl-1.1.1b | 4.0 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(scikit-learn_0.21) $ python -c "import sklearn"
(scikit-learn_0.21) $
```

## 라. Conda Environment 목록 확인

- "conda-env list" 또는 "conda env list"를 이용하여 목록을 확인할 수 있다.

- 예제 -

```
(scikit-learn_0.21) $ conda env list
# conda environments:
#
base /apps/applications/PYTHON/3.7
scikit-learn_0.21 * /home01/userID/.conda/envs/scikit-learn_0.21
(scikit-learn_0.21) $ conda deactivate
$
```

## 마. Conda Environment 내보내기

- 내보내기 전 conda-pack 패키지 필요

※ (참고) <https://conda.github.io/conda-pack>

- "conda pack -n [ENVIRONMENT] -o [파일명]"을 이용하여 conda environment를 다른 시스템에서 활용할 수 있다.

(예) 외부 인터넷이 연결되지 않는 다른 시스템에서 동일한 conda 환경을 이용하고자 할 때나, 다른 시스템에서 동일한 conda 환경이 필요할 때 재설치하지 않고 아래와 같이 conda 환경을 내보낼 수 있음

- 예제 -

```
$ conda activate scikit-learn_0.21
(scikit-learn_0.21) $ conda install -c conda-forge conda-pack
(scikit-learn_0.21) $ conda pack -n scikit-learn_0.21 -o scikit-learn_test.tar.gz
Collecting packages...
Packing environment at '/home01/userID/.conda/envs/scikit-learn_0.21' to 'scikit-learn_test.tar.gz'
[#####] | 100% Completed | 4min 18.8s
(scikit-learn_0.21) $ ls -l scikit-learn_test.tar.gz
-rw-----. 1 userID in0162 1459826406 Mar 28 15:03 scikit-learn_test.tar.gz
(scikit-learn_0.21) $
```

## 바. Conda Environment 가져오기

- conda pack을 이용하여 생성했던 conda environment를 아래 -예제-와 같이 가져와 환경설정 후 사용 가능.

- 예제 -

```
$ mkdir -p $HOME/.conda/envs/scikit-learn_test
$ tar xvfz scikit-learn_test.tar.gz -C $HOME/.conda/envs/scikit-learn_test
※ scratch 가 conda 경로인 경우, /scratch/$USER/.conda 로 입력

$ conda activate scikit-learn_test
(scikit-learn_0.21) $ conda-unpack
(scikit-learn_0.21) $ conda deactivate
$
```

## 사. Conda Environment 삭제

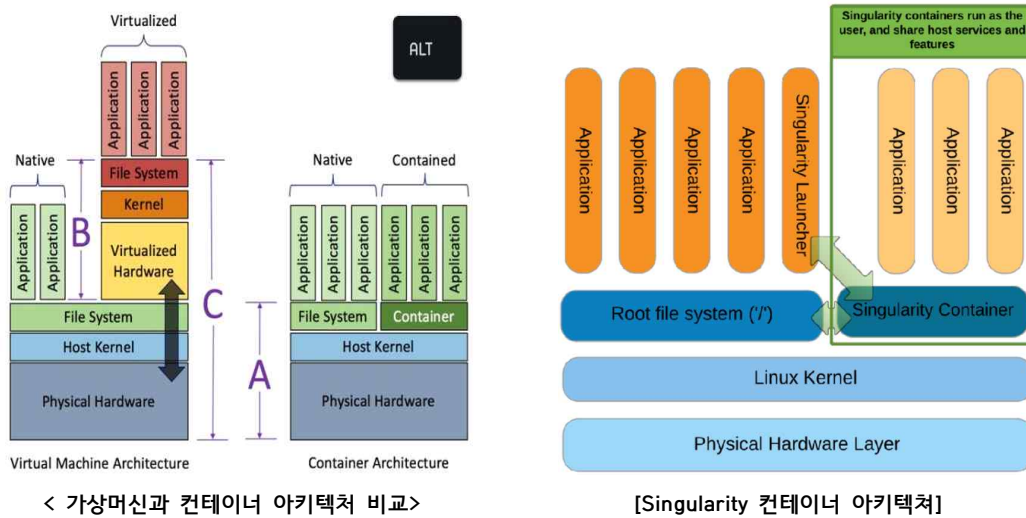
- "conda-env remove -n [ENVIRONMENT]" 또는 "conda env remove -n [ENVIRONMENT]"를 이용하여 삭제할 수 있다.

- 예제 -

```
$ conda env remove -n scikit-learn_test  
Remove all packages in environment /home01/userID/.conda/envs/scikit-learn_test:
```

### III-3. Singularity 컨테이너

싱귤러리티(Singularity)는 도커(Docker)와 같이 OS 가상화를 구현하기 위한 HPC 환경에 적합한 컨테이너 플랫폼이다. 사용자 작업 환경에 적합한 리눅스 배포판, 컴파일러, 라이브러리 등을 포함하는 컨테이너 이미지를 생성하고 컨테이너를 구동하여 사용자 프로그램을 실행할 수 있다.



< 가상머신과 컨테이너 아키텍처 비교 >

[Singularity 컨테이너 아키텍처]

※ 가상머신은 애플리케이션이 하이퍼바이저와 게스트 OS를 거쳐 올라가는 구조이나, 컨테이너는 물리적인 하드웨어에 더 가까우며 별도의 게스트 OS가 아닌 호스트 OS를 공유하기 때문에 오버헤드가 더 작음. 최근 클라우드 서비스에서 컨테이너의 활용이 증가하고 있음

## 가. 컨테이너 이미지 빌드하기

### 1. 싱귤러리티 모듈 적재 혹은 경로 설정

```
$ module load singularity/3.11.0  
or  
$ $HOME/.bash_profile  
export PATH=$PATH:/apps/applications/singularity/3.11.0/bin/
```

### 2. 로컬 빌드

- 누리온 시스템의 로그인 노드에서 컨테이너 이미지를 로컬 빌드하기 위해서는, 먼저 [KISTI 홈페이지](#) > [기술지원](#) > [상담신청](#)을 통해 아래와 같은 내용으로 fakeroot 사용 신청을 해야함
  - 시스템명 : 누리온
  - 사용자 ID : a000abc
  - 요청사항 : 싱귤러리티 fakeroot 사용 설정

```
$ singularity [global options...] build [local options...] <IMAGE PATH>  
<BUILD SPEC>
```

[주요 global options]

- d : 디버깅 정보를 출력함
- v : 추가 정보를 출력함
- version : 싱글레리티 버전 정보를 출력함

[관련 주요 local options]

- fakeroot : root 권한 없이 일반사용자가 가짜 root 사용자로 이미지 빌드
- remote : 외부 싱글레리티 클라우드(Sylabs Cloud)를 통한 원격 빌드(root 권한 필요 없음)
- sandbox : 샌드박스 형태의 쓰기 가능한 이미지 디렉터리 빌드

<IMAGE PATH>

- default : 읽기만 가능한 기본 이미지 파일(예시 : ubuntu1.sif)
- sandbox : 읽기 및 쓰기 가능한 디렉터리 구조의 컨테이너(예시 : ubuntu4)

<BUILD SPEC>

definition file : 컨테이너를 빌드하기 위해 recipe를 정의한 파일(예시 : ubuntu.def)

local image : 싱글레리티 이미지 파일 혹은 샌드박스 디렉터리(IMAGE PATH 참조)

URI

library:// 컨테이너 라이브러리 (default <https://cloud.sylabs.io/library>)

docker:// 도커 레지스트리 (default 도커 허브)

shub:// 싱글레리티 레지스트리 (default 싱글레리티 허브)

oras:// OCI 레지스트리

```

① Definition 파일로부터 ubuntu1.sif 이미지 빌드하기
$ singularity build --fakeroot ubuntu1.sif ubuntu.def*

② 싱귤러리티 라이브러리로부터 ubuntu2.sif 이미지 빌드하기
$ singularity build --fakeroot ubuntu2.sif library://ubuntu:18.04

③ 도커 허브로부터 ubuntu3.sif 이미지 빌드하기
$ singularity build --fakeroot ubuntu3.sif docker://ubuntu:18.04

④ 도커 허브로부터 intel 아키텍처에 최적화된 pytorch 이미지 빌드하기
$ singularity build --fakeroot pytorch1.sif
  docker://intel/intel-optimized-pytorch:2.3.0-pip-multinode

⑤ Definition 파일로부터 intel 아키텍처에 최적화된 pytorch 이미지 빌드하기
$ singularity build --fakeroot pytorch2.sif pytorch.def**

* ) ubuntu.def 예시
bootstrap: docker
from: ubuntu:18.04
%post
apt-get update
apt-get install -y wget git bash gcc gfortran g++ make file
%runscript
echo "hello world from ubuntu container!"

** ) pytorch.def 예시
# 로컬 이미지 파일로부터 콘다를 사용하여 새로운 패키지 설치를 포함한 이미지
  빌드
bootstrap: localimage
from: pytorch1.sif
%post
pip install scikit-image

# 도커 허브에서 사용하여 새로운 패키지 설치를 포함한 이미지 빌드
bootstrap: docker
from: intel/intel-optimized-pytorch:2.3.0-pip-multinode
%post
pip install scikit-image

```

### 3. 원격 빌드

```
$ singularity build --remote ubuntu4.sif ubuntu.def
```

(Sylabs Cloud에서 제공하는 원격 빌드 서비스 이용하여 Definition 파일로부터 ubuntu4.sif 이미지 빌드하기)

※ Sylabs Cloud(<https://cloud.sylabs.io>)에서 제공하는 원격빌드 서비스를 이용하려면 액세스 토큰을 생성하여 뉴론 시스템에 등록해야 함 [참조 1]

※ 또한, Sylabs Cloud에 웹 브라우저 접속을 통해서 싱글레리티 컨테이너 이미지의 생성·관리가 가능함 [참조 2]

### 4. 컨테이너 이미지 가져오기/내보내기

```
$ singularity pull tensorflow.sif library://dxtr/default/hpc-tensorflow:0.1
```

(Sylabs cloud 라이브러리에서 컨테이너 이미지 가져오기)

```
$ singularity pull tensorflow.sif docker://tensorflow/tensorflow:latest
```

(도커 허브에서 이미지를 가져와서 싱글레리티 이미지로 변환)

```
$ singularity push -U tensorflow.sif library://ID/default/tensorflow.sif
```

(Sylabs Cloud 라이브러리에 싱글레리티 이미지 내보내기(업로드))

※ Sylabs Cloud(<https://cloud.sylabs.io>)에 이미지를 내보내기(업로드) 위해서는 액세스 토큰을 생성하여 누리온에 등록해야 함 [참조 1]

## 다. 싱글레리티 컨테이너에서 사용자 프로그램 실행

### 1. 싱글레리티 모듈 적재 혹은 경로 설정

```
$ module load singularity/3.11.0
```

or

```
$ $HOME/.bash_profile
```

```
export PATH=$PATH:/apps/applications/singularity/3.11.0/bin/
```

### 2. 싱글레리티 컨테이너에서 프로그램 실행 명령어

```
$ singularity [global options...] shell [shell options...] <container>
```

```
$ singularity [global options...] exec [exec options...] <container>
```

```
<command>
```

```
$ singularity [global options...] run [run options...] <container>
```



```

① 싱귤러리티 컨테이너에서 셸 실행 후 사용자 프로그램 실행
$ singularity shell pytorch1.sif
Singularity> python test.py

② 싱귤러리티 컨테이너에서 사용자 프로그램 실행
$ singularity exec pytorch1.sif python test.py
$ singularity exec singularity exec
  docker://intel/intel-optimized-pytorch:2.3.0-pip-multinode python
  test.py
$ singularity exec library://dxtr/default/hpc-tensorflow:0.1 python test.py

③ 싱귤러리티 컨테이너에서 runscript(이미지 빌드 시 생성)가 존재하면 이 스크립트를 실행하고,
  runscript가 존재하지 않고 컨테이너 뒤에 사용자 명령어를 입력하면 해당 명령어가 실행됨
$ singularity run ubuntu1.sif
hello world from ubuntu container!

$ singularity run pytorch1.sif python test.py

```

### 3. 스케줄러(PBS)를 통한 컨테이너 실행 방법

#### 1) 작업 스크립트를 작성하여 배치 형태 작업 실행

- 실행 명령어 : qsub <작업 스크립트 파일>

```

[id@login01]$ qsub job_script.sh
14954055.pbs

```

- ※ 자세한 스케줄러(PBS) 사용 방법은 "[누리온 지침서-스케줄러\(PBS\)를 통한 작업실행](#)" 참조

#### 2) 작업 스크립트 파일 예시

- Serial 작업

- \* 실행 명령어 : qsub <작업 스크립트 파일>

```
#!/bin/sh
#PBS -N openfoam
#PBS -q normal
#PBS -A openfoam
#PBS -V
#PBS -j oe
#PBS -W sandbox=PRIVATE
#PBS -m e
#PBS -M wjnadia@kisti.re.kr
#PBS -r y
#PBS -l select=1:ncpus=1:mpiprocs=1:omphthreads=1
#PBS -l walltime=00:30:00

cd $PBS_O_WORKDIR
module load singularity/3.11.0
cd cavity
singularity run openfoam-default:2312.sif icoFoam
```

- **Serial 작업**

\* 실행 명령어 : mpirun singularity run <컨테이너> [사용자 프로그램 실행 명령어]

```
#!/bin/sh
#PBS -N openfoam
#PBS -q normal
#PBS -A openfoam
#PBS -V
#PBS -j oe
#PBS -W sandbox=PRIVATE
#PBS -m e
#PBS -M wjnadia@kisti.re.kr
#PBS -r y
#PBS -l select=2:ncpus=64:mpiprocs=64:omphthreads=1
#PBS -l walltime=00:30:00

cd $PBS_O_WORKDIR
module load singularity/3.11.0 gcc/8.3.0 openmpi/3.1.0
cd cavity
mpirun singularity run openfoam-default:2312.sif icoFoam
```

※ 2노드 점유, 노드 당 64 TASK(총 128개 MPI 프로세스) 사용 예제

### 3) 스케줄러가 할당한 계산 노드에서 인터랙티브 작업 실행

- 병렬프로그램(OpenMPI) 실행 예시

```
[id@login01]$ qsub -I -l select=2:ncpus=64:mpiprocs=64:omphthreads=1 -l  
walltime=00:30:00 -q normal -A openfoam  
qsub: waiting for job 14954204.pbs to start  
qsub: job 14954204.pbs ready  
  
[id@node1000]$  
[id@node1000]$ module load singularity/3.11.0 gcc/8.3.0 openmpi/3.1.0  
[id@node1000]$ cd cavity  
[id@node1000]$ mpirun singularity run openfoam-default:2312.sif icoFoam
```

- 텐서플로우 프로그램 실행 예시

```
$ qsub -I -V -l select=1:ncpus=68:omphthreads=68 \  
-l walltime=12:00:00 -q normal -A tf  
  
$ export OMP_NUM_THREADS=68; singularity exec tensorflow-1.12.0-py3.simg  
python convolutional.py
```

※ 예제 singularity 이미지 파일 위치:

/apps/applications/tensorflow/1.12.0/tensorflow-1.12.0-py3.simg

※ 예제 convolutional.py 위치:

/apps/applications/tensorflow/1.12.0/examples/convolutional.py

## 라. 참조

[참조 1] Sylabs Cloud 액세스 토큰 생성 및 누리온에 등록하기

## 웹 브라우저

### ① Sylabs Cloud 계정 등록 및 로그인 하기

 <https://cloud.sylabs.io/home> Sylabs Cloud



 Sign In

Singularity Container Services

### ② 새로운 토큰 생성하기



The screenshot shows the 'Account Management' page of Sylabs Cloud. The user is logged in as 'ID'. A dropdown menu is open under the 'ID' header, showing options: 'Access Tokens', 'My Containers', 'My Stars', and 'Sign Out'. The 'Access Tokens' option is highlighted. Below the header, the page says 'Welcome back ID' and 'My Tokens'. There is a text input field with 'new token' and a green button labeled 'Create a New Access Token'.




[참조 2] 웹 브라우저에서 리모트 빌더에 의한 싱귤러리티 컨테이너 이미지 빌드하기

## ① 웹 브라우저에서 컨테이너 이미지 빌드하기

https://cloud.sylabs.io/builder Sylabs Cloud

Home Singularity Library Remote Builder Keystore

 Remote Builder

Singularity 3 introduces the ability to build your containers in the cloud, so you can easily and securely create containers for your applications without special privileges or setup on your local system. The Remote Builder can securely build a container for you from a definition file entered here or via the Singularity CLI.

Build a Container

Use the editor below to provide a definition file to build. You can drag & drop definition files onto the editor, or use the upload button to select a file from your computer. See the [Singularity User Guide](#) for more information about writing definition files.

Upload a definition file

Valid definition file

```
1 bootstrap: library
2 from: ubuntu:18.04
3 %runscript
4 echo "hello world from ubuntu container!"
5
```











(Optional) Specify a location for the built image:

wjnadia/default/ 

Select location

Build

## ② 빌드한 컨테이너 이미지 목록 보기

My Builds						
Build ↕	Status ↕	Submit Time ↕	Duration ↕	Recipe	Library Image	
<a href="#">5fbdfc449a193de1c6105256</a>	Completed	2020-11-25T15:40:04+09:00	13s	<a href="#">library:ubuntu:18.04</a>	<a href="#">52.09 MB</a>	
<a href="#">5fbdfb1b1cf9a73d6be9497a</a>	Completed	2020-11-25T15:35:07+09:00	13s	<a href="#">library:ubuntu:18.04</a>	<a href="#">52.09 MB</a>	
<a href="#">5fbdf7e91cf9a73d6be94979</a>	Completed	2020-11-25T15:21:29+09:00	14s	Unknown	<a href="#">24.67 MB</a>	
<a href="#">5fbdf35871bec9ed56678526</a>	Completed	2020-11-25T15:02:00+09:00	13s	Unknown	<a href="#">52.09 MB</a>	
<a href="#">5fbdec6b71bec9ed56678523</a>	Completed	2020-11-25T14:32:27+09:00	12s	Unknown	<a href="#">52.09 MB</a>	
<a href="#">5fb350549a193de1c6104ec4</a>	Completed	2020-11-17T13:23:48+09:00	52s	Unknown	<a href="#">81.23 MB</a>	
<a href="#">5fb33ba41cf9a73d6be945bc</a>	Completed	2020-11-17T11:55:32+09:00	23s	Unknown	<a href="#">48.61 MB</a>	
<a href="#">5fb264539a193de1c6104e5c</a>	Completed	2020-11-16T20:36:51+09:00	49s	Unknown	<a href="#">81.23 MB</a>	
<a href="#">5fb2466571bec9ed56678131</a>	Completed	2020-11-16T18:29:09+09:00	55s	Unknown	<a href="#">81.23 MB</a>	
<a href="#">5fb228db71bec9ed56678125</a>	Completed	2020-11-16T16:23:07+09:00	49s	Unknown	<a href="#">81.23 MB</a>	
<div> <span>⏪</span> <span>⏴</span> <span>1</span> <span>2</span> <span>⏵</span> <span>⏩</span> </div>						

※ 누리온에서 *singularity* 명령어로 리모트 빌드한 이미지 목록도 포함됨

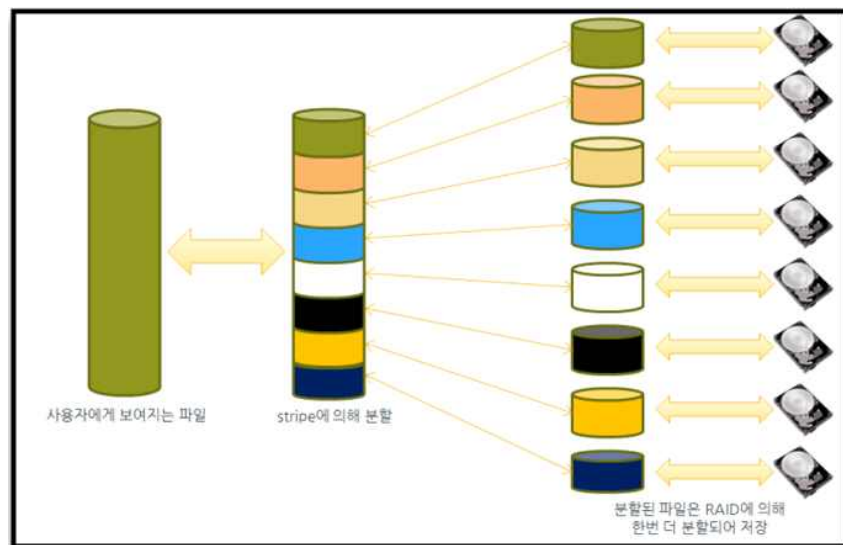
### Ⅲ-4. Lustre stripe

#### 가. 5호기 Lustre Striping 기본 설정

5호기 Lustre File System은 File Striping을 지원하며, 이를 위해 복수 개의 OST(Object Storage Target, 즉 물리적으로 분산되어 있는 여러 디스크)에 하나의 파일을 분산시켜 저장함으로써 병목을 줄이고 I/O 성능을 향상시킬 수 있다. 특히, Lustre 2.10부터 지원되는 PFL(Progressive File Layout)이 /scratch 파일시스템에 적용되어 있다. 이 기능은 사용자 별도의 striping 설정 없이 파일의 크기에 따라 stripe-count 개수가 자동으로 적용되어 I/O의 성능을 향상시킬 수 있다. Nurion 파일 시스템의 striping 설정은 아래와 같다.

		stripe-count	stripe-size
/home01		1	1MB
/apps		1	1MB
/scratch (PFL)	file size < 4MB	1	1MB
	4M < file size < 512MB	2	1MB
	512MB < file size < 1G	4	1MB
	1G < file size < 10G	8	1MB
	10G < file size < 100G	16	1MB
	file size > 100G	32	1MB

#### 나. Lustre Striping 개념



Lustre는 각 OST별로 자료를 분할하여 대용량 파일에 대한 I/O 성능을 최대화할 수 있으며, 병렬화가 유효한 최대 분할 수는 OST 숫자와 같다. 단일 파일 역시 위 그림과 같이 Lustre Striping 기능을 사용하여 OST에 병렬로 저장함.



## 다. stripe 설정 및 확인

```
$ lfs setstripe [--stripe-size|-s size] [--stripe-count|-c count] filename|dirname
```

- 파일 또는 디렉터리에 striping 설정을 적용시키는 명령어. 위 명령으로 생성된 파일이나 위 명령이 적용된 디렉터리에서 생성되는 모든 파일은 striping 설정 적용
  - --stripe-size
    - 각 OST에 저장할 데이터의 크기를 설정
    - 지정된 크기만큼 저장하면 다음 OST에 데이터를 저장
    - 기본값은 1MB이며 stripe\_size를 0으로 설정하면 기본 값을 사용함
    - stripe\_size는 반드시 64KB의 배수로 설정해야 하며 4GB보다 작아야 함
  - --stripe-count
    - Striping에 사용할 OST 개수를 설정
    - 기본 값은 1이며 stripe\_count를 0으로 설정하면 기본 값을 사용
    - stripe\_count가 -1이면 가능한 모든 OST들을 사용

```
$ lfs getstripe filename|dirname
```

- 파일 또는 디렉터리에 적용된 striping 설정값을 확인하는 명령어

## 라. 권장사항 및 팁

- 작업 스크립트 내에서 모델의 결과 파일이 저장될 디렉터리에 대해 setstripe를 지정하면, 이후 생성되는 하위 디렉터리 및 파일은 모두 해당 설정 값 상속
- --stripe-count는 파일 사이즈가 1GB 이상인 파일에 대해 4로 설정 시 대부분 성능 향상. 더 큰 값 사용 시 테스트 필요
- --stripe-size는 파일 사이즈가 수 TB 이상인 파일인 경우에만 유효하므로 대부분 default 값을 사용해도 문제없음

### Ⅲ-5. 데이터 아카이빙

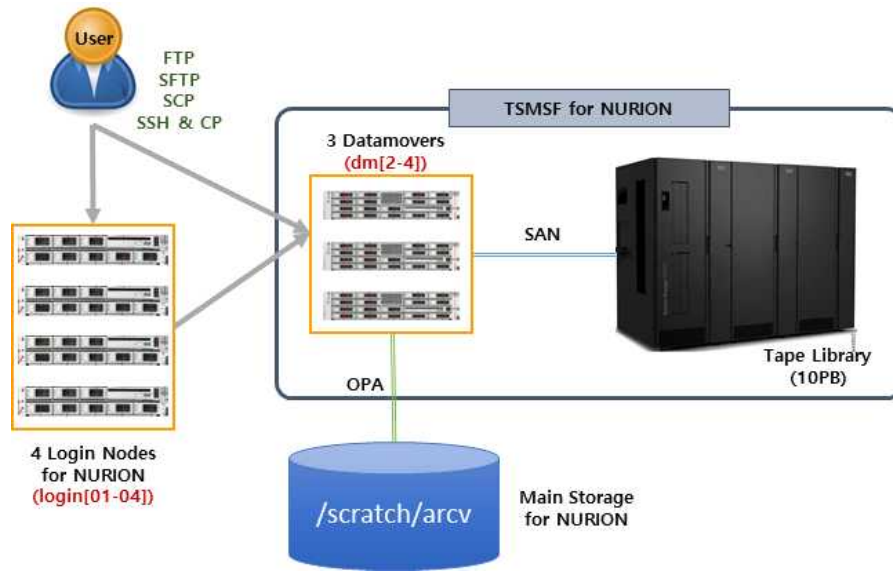


#### 가. 시스템 환경 및 사용 신청

누리온은 사용자의 중요 파일 및 대용량 파일의 장기간(계약 만료 후 1년) 보관을 지원하기 위하여 10PBytes의 테이프(Tape) 라이브러리(IBM TS4500)를 운영합니다.

TSMSF(Tiered Storage Management Script Facility)는 사용자의 데이터 백업 및 아카이빙을 지원하기 위한 소프트웨어로 누리온의 Datamover 노드에 설치되어 운영되고 있습니다. TSMSF는 사용자 데이터를 자동으로 아카이빙하고 사용자가 데이터를 필요로 할 때 수동으로 복원(restore)할 수 있도록 지원을 합니다. 이는 관리자에 의해 정기적으로 이루어지는 사용자 홈 디렉터리(/home01) 백업과는 달리 사용자가 직접 파일을 특정 디렉터리(/scratch/arcv/[\$USER])에 업로드(upload)하는 방식으로 이루어집니다. 현재 TSMSF는 그림과 같이 테이프(Tape) 라이브러리, Datamover(DM) 노드로 구성됩니다.

- 누리온 시스템 TSMSF : 테이프(10PB)
- 누리온 시스템 Datamover 노드: [nurion-dm.ksc.re.kr](http://nurion-dm.ksc.re.kr)



[ TSMSF 시스템 구성도 ]

사용자가 파일을 Datamover에서 지정된 디렉터리(/scratch/arcv/\$USER)로 업로드(upload)하고 7일(추후 변경 가능)이 지나면 데이터는 자동으로 아카이빙(Archiving)이 진행되게 됩니다. Datamover로 파일을 업로드 할 경우에는 FTP, SCP, SSH, SFTP 등을 이용하여 외부에서 직접 파일을 업로드 할 수 있고, 누리온 로그인 노드(단, 로그인 노드의 경우 FTP 사용 불가)를 통해서도 가능합니다. 이 때, FTP를 제외한 서비스는 일회용 패스워드(OTP)를 입력해야 합니다. 또한 FTP를 제외한 서비스는 제한된 자원(CPU 시간 10분)을 소진하게 되면 자동으로 파일 업로드가 중단되니 FTP를 이용하여 파일을 업로드하여 주시길 권고 드립니다.

## 나. 사용자 정책 및 신청 방법

TSMSF에 저장할 수 있는 용량은 계정(ID) 당 100TB(파일수는 최대 1,000,000개)이며, 보존기간은 사용자의 계정 사용 종료 후 1년까지입니다.

TSMSF는 기본적으로 테이프 미디어를 이용하므로 **테이프 미디어의 장애 시 복구가 어려울 수 있습니다.** 따라서 **중요 데이터는 사용자 로컬 시스템에 반드시 보관해 주시는 것이 좋습니다.**

또한 테이프 미디어는 접근(Access)을 위하여 테이프 라이브러리로 로딩>Loading>할 때, 일반적으로 많은 시간이 소요되므로 백업/아카이빙 하고자 하는 파일들이 source 파일과 같이 작은 파일들이 많은 경우 tar 및 gzip 등을 통해 압축하여 보관하시는 것이 좋습니다.

**TSMSF를 사용하기 위해서는 별도의 사용 환경 구성이 필요하므로 account@ksc.re.kr 메일로 신청하시기 바랍니다.(별도의 양식은 필요하지 않습니다. 계정 정보(ID)를 보내 주시면 됩니다.)**

그 외의 일반적인 기술 지원 관련 문의는 consult@ksc.re.kr 메일을 이용해주시기 바랍니다.

## 다. 아카이빙 방법

### 1. 로그인 노드 및 Datamover 노드 접속

사용자 데이터 아카이빙을 위해서는 누리온 시스템의 로그인 노드나 Datamover 노드를 활용하시기 바랍니다. 주의하실 점은 최초 로그인 시 디렉터리 위치는 '/home01/\$USER'로 이는 TSMSF 내의 홈 디렉터리가 아니라 누리온 시스템의 일반적인 홈 디렉터리입니다. 따라서 cd 명령어(cd /scratch/arcv/\$USER)로 경로를 변경하여 TSMSF를 위한 사용자 디렉터리로 이동하시기 바랍니다.

(방법1) 로그인 노드(nurion.ksc.re.kr)나 Datamover 노드(nurion-dm.ksc.re.kr)에 접속(ssh)하여 cp 사용

```
[user01@dm2:/home01/user01]$ ssh $USER@nurion-dm.ksc.re.kr (또는  
ssh $USER@nurion.ksc.re.kr)  
Password(OTP):  
Password:  
[user01@dm02:/home01/user01]$ cp -r [mydir | myfile] /scratch/arcv/$USER/
```

(방법2) 원격에서 scp 사용

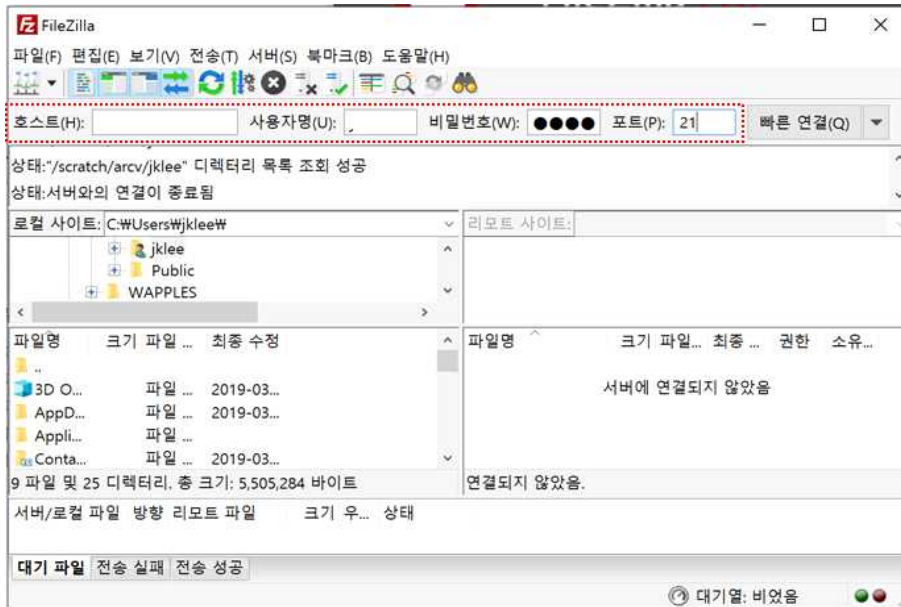
```
$ scp -r [mydir | myfile] $USER@nurion-dm.ksc.re.kr:/scratch/arcv/$USER/  
Password(OTP):  
Password:
```

(방법3) FTP / SFTP 클라이언트 프로그램을 이용하여 Datamover 노드(nurion-dm.ksc.re.kr) 접속하여 파일 업로드

## 1) 파일질라(FileZilla) 클라이언트를 이용한 예제(FTP)

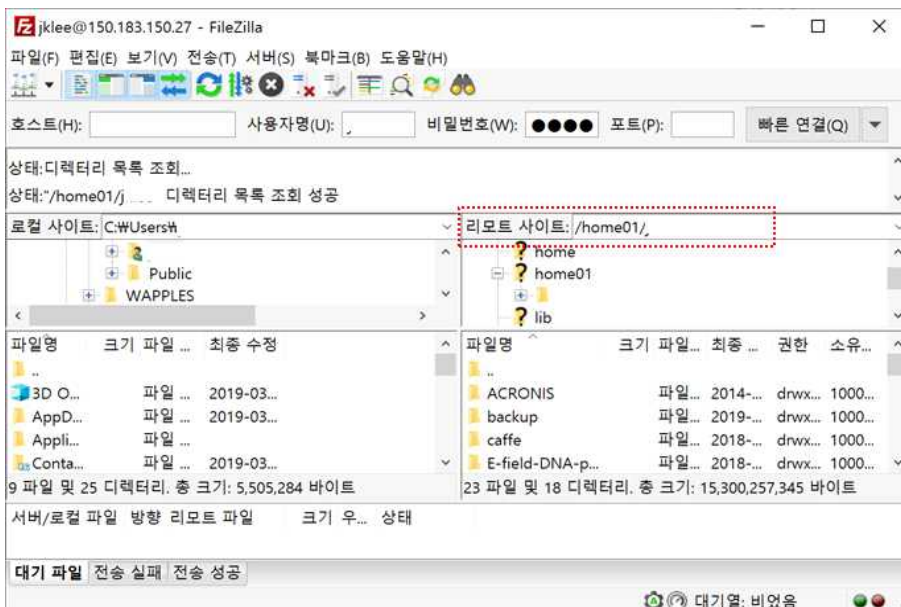
- Datamover 노드에 접속하기 위한 정보를 입력하고 빠른 '연결 버튼'을 클릭

호스트	사용자명	비밀번호	포트
nurion-dm.ksc.re.kr	USER ID(사용자 계정)	사용자 비밀번호	21

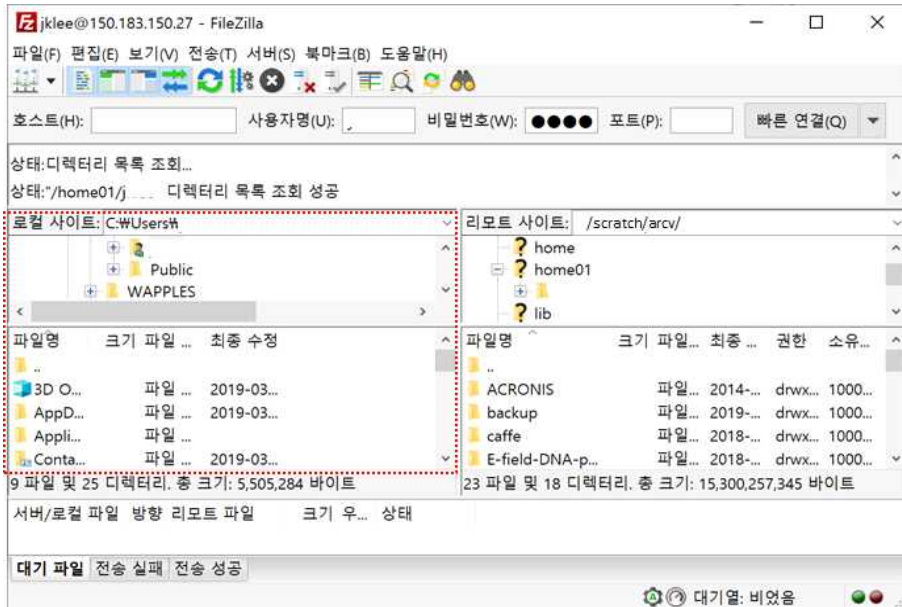


- 리모트 사이트의 경로가 사용자 홈 디렉터리(/home01/\$USER) 이므로 지정된 디렉터리(/scratch/arcv/\$USER)로 이동

※ 리모트 사이트 경로에 직접 절대경로를 입력하면 쉽게 이동 가능



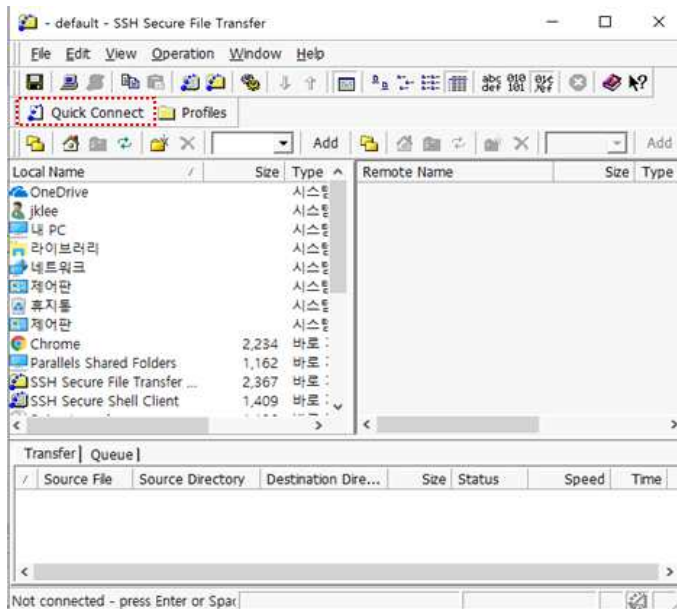
- 왼쪽 로컬 사이트에서 파일이나 디렉터리를 선택하여 파일 업로드



※ 보다 상세한 파일질라 매뉴얼은 사이트(<https://filezilla-project.org/>) 참조

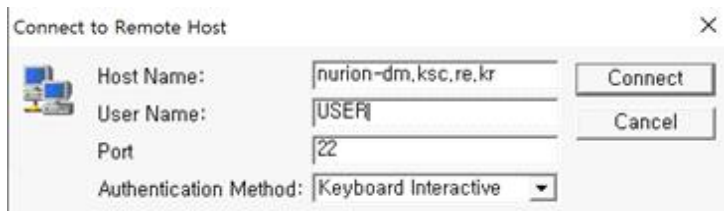
## 2) SSH Secure Shell Client를 이용한 예제(SFTP)

- Datamover 노드에 접속하기 위해서 'Quick Connect' 버튼 클릭



- Datamover 노드에 접속하기 위한 정보를 입력하고 '연결(Connect)' 버튼 클릭

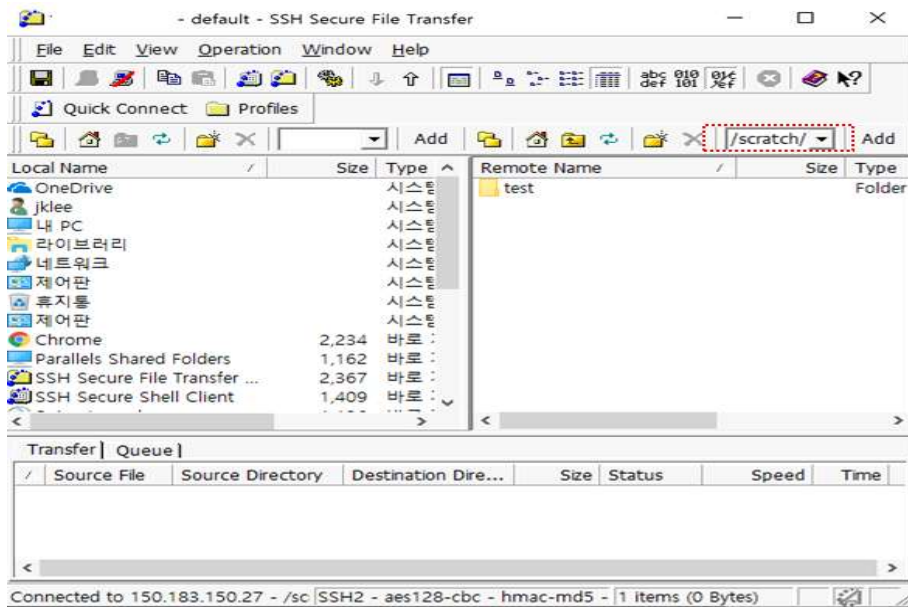
호스트	사용자명	포트	인증방법
nurion-dm.ksc.re.kr	USER ID(사용자 계정)	22	Keyboard Interactive



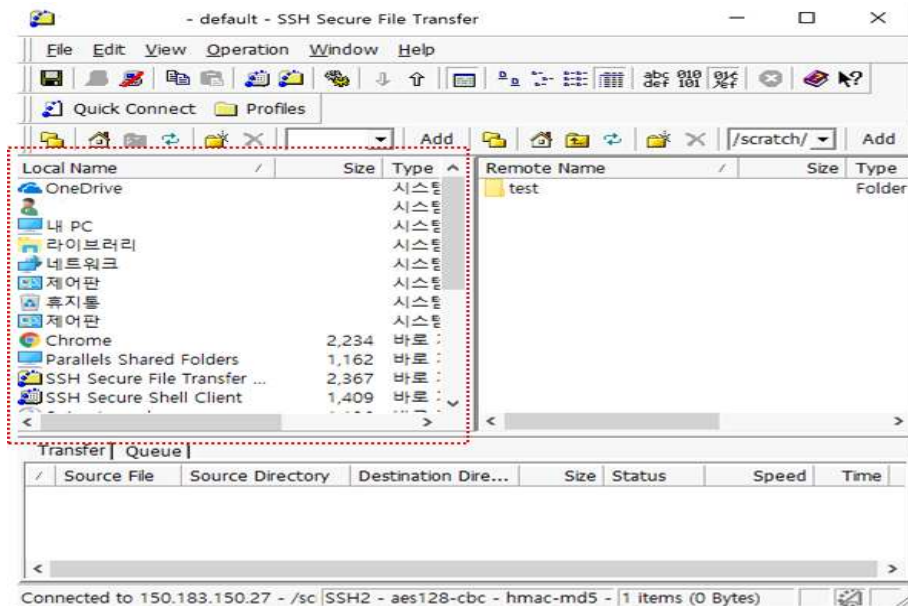
- 일회용 패스워드(OTP)와 비밀번호(Password)를 순차적으로 입력하고 'OK' 버튼 클릭



- 오른쪽 사이트의 경로가 사용자 홈 디렉터리(/home01/\$USER) 이므로 지정된 디렉터리(/scratch/arcv/\$USER)로 이동
- ※ 해당 경로에 직접 절대경로를 입력하면 쉽게 이동 가능



- 왼쪽 로컬 사이트에서 파일이나 디렉터리를 선택하여 파일 업로드





## 2. 데이터 아카이빙

지정된 디렉터리(/scratch/arcv/\$USER) 내 파일 중 크기가 10Mbytes 이상이고 3일 동안(추후 변경 예정) 접근하지 않은 파일은 자동으로 아카이빙이 진행되며, 아카이빙이 진행된 파일은 다음과 같이 파일 사이즈 0인 임시파일(chunk file)만 해당 디렉터리에 존재하고 실제 데이터는 테이프 라이브러리에 존재하게 됩니다. 파일의 개수가 많아지면 테이프 라이브러리로 아카이빙을 진행할 때나 파일 사용을 위해 디스크로 복구하는 데 시간이 더 걸리게 되므로 가능하면 압축(tar 등)하여 파일 개수를 줄일 것을 권장합니다.

```
[user01@dm2 ~] ls -lh /scratch/arcv/$USER/*  
... 생략 ...  
-rw-r--r-- 1 root root 0 Feb 5 18:51 2019-2-19-18-51.[File Name].archived  
... 생략 ...
```

### ◦ 예제

```
# ls -l /scratch/arcv/testdir/  
total 2744320  
-rw-r--r-- 1 root root 0 Feb 25 20:48 2019-2-25-20-48.test.220M.archived  
-rw-r--r-- 1 root root 0 Feb 25 20:48 2019-2-25-20-48.test.230M.archived  
-rw-r--r-- 1 root root 104857600 Sep 27 14:54 test.100M  
-rw-r--r-- 1 root root 10485760 Sep 27 14:54 test.10M  
-rw-r--r-- 1 root root 115343360 Sep 27 14:54 test.110M  
... 생략 ...
```

### 3. 아카이빙된 파일의 세부 정보 확인

아카이빙 된 자료는 `arc_ls` 명령어를 통해 다음과 같이 파일의 세부 정보를 확인할 수 있습니다.

```
# arc_ls /scratch/arcv/testdir/  
DIR: /scratch/arcv/testdir  
ARC: /scratch/arcv/testdir/2019-2-25-20-48.test.460M.archived  
- File to restore : /scratch/arcv/testdir/test.460M  
- File owner : 0  
- File group : 0  
- File size : 482344960  
- Archived Name : 3_7_4_6_scratcharcvtestdirtest.460M  
- Archived time : 1551095210  
NRM: /scratch/arcv/testdir/test.10M  
... 생략 ...
```

`arc_ls` 명령어는 디렉터리나 파일의 상대경로 또는 절대경로를 인자로 줄 수 있고 필요 시 `-r` 옵션을 통해 지정된 디렉터리 내 하위 경로를 탐색할 수 있습니다. 출력 결과에서 DIR은 디렉터리를 의미하며 NRM은 일반 파일, ARC는 아카이빙 된 파일을 의미합니다.

## 라. 복원(Restore) 방법

아카이빙 된 데이터를 사용하기 위해서는 복원을 진행해야 합니다. 파일 복원을 위해서 제공하는 명령어는 arc\_restore입니다. 이때 **테이프 라이브러리에서 데이터 파일을 가져오므로 많은 시간이 소요됩니다.** 다음과 같이 아카이빙 된 파일을 복원할 수 있습니다.

```
[user01@dm2 ~] arc_restore /scratch/arcv/$USER/[*.archived]
... 생략 ...
-rw-r--r-- 1 root root 200M Feb 5 18:51 [original file]
... 생략 ...
```

◦ 예제

```
# arc_restore arc_restore /scratch/arcv/testdir/2019-2-25-20-49.test.580M.archived
ARCRESTORE : start to find archived files.
Processing restore queue...
[207249:207265] resRunner : filecheck : ADD : 1/0/0 : /scratch/arcv/testdir/test.580M
[207265:207269] Process ID 207269 started for restoring...
[207265:207269] File to restore : /scratch/arcv/testdir/test.580M
[207265:207269] File owner : 0
[207265:207269] File group : 0
[207265:207269] File size : 608174080
[207265:207269] Archived Name : 3_7_4_6_scratcharcvtestdirtest.580M
[207265:207269] Archived time : 1551095339
[207265:207269] Restore started at : Thu Feb 28 12:16:43 2019
[207265:207269] restoreFile : sh PTLGet -p 3_7_4_8_scratcharcvtestdirtest.580M
-d /scratch/arcv/testdir -t 1551095339
[207265:207269] restoreFile : move /scratch/arcv/testdir/.working/3_7_4_6_scratcharcvtestdirtest.580M
to /scratch/arcv/testdir/test.580M
[207265:207269] restoreFile : unlink /scratch/arcv/testdir/2019-2-25-20-49.test.580M.archived
[207265:207269] restoreFile : dbupdate( ARCHIVED_FILES /
last_mod=1551323868,restore_count=1,status_id=1,result_id=1 Where
file_id=341 )
[207265:207269] Restoring was done : /scratch/arcv/testdir/test.580M
[207249:207265] resRunner : CProc was forked : CProc count=1, CProc
tCount=1
[207249:207265] resRunner : [ success=1 / fail=0 ] 207269 was finished :
Success
resRunner : REPORT : resresult : Success=1 / Fail=0
Main thread runtime = 00:01:05
```

arc\_restore 명령어는 arc\_ls 명령어와 마찬가지로 디렉터리나 파일의 상대경로 또는 절대경로를 인자로 줄 수 있고, 필요시 -r 옵션을 통해 지정된 디렉터리 내의 하위 경로를 탐색할 수 있습니다. 또한, 파일 지정 시 chunk 파일명(timestamp.[File Name].archived)과 원본 파일명을 모두 지정 가능하며, 두 가지 중 하나를 선택하여 사용할 수 있습니다.

## 마. 삭제(Delete) 방법

지정된 디렉터리(/scratch/arcv/\$USER) 내 파일 중 일부, 혹은 전체를 삭제하기 원한다면 Linux 기본 명령어인 rm 명령어를 사용할 수 있습니다.

임시파일(chunk file, 확장자 .archived) 역시 rm 명령어를 사용하여 삭제하면 되며, 임시파일 삭제 시 테이프 라이브러리로 아카이빙 된 데이터가 실제 삭제되어 동기화되기까지 약 3일 정도 소요될 수 있습니다. 즉, 임시 파일이 원래 경로에 존재하지 않다면 “사용자가 해당 파일을 삭제”했다고 판단하므로 직접 임시 파일의 파일명을 변경하거나 이동시키지 말고, 꼭 복원 후 변경/이동을 해야합니다. (임시 파일 삭제 시 파일 복원이 불가능하므로 유의하시기 바랍니다.)

## III-6. MVAPICH2 성능 최적화 옵션

MPI 라이브러리로 MVAPICH2를 활용하는 경우 작업 스크립트 내에서 환경변수를 사용하여 프로세스 간 통신 알고리즘을 설정할 수 있다. 주요 키워드는 아래와 같으며, 사용자는 코드 내에서 많이 활용되는 collective communication에 대하여 환경변수 설정을 활용하여 실행 성능을 최적화할 수 있다.

**MV2\_INTRA** 환경변수는 노드 내 MPI 프로세스 통신에서 활용되며 **MV2\_INTER** 환경변수는 노드 간 MPI 프로세스 통신에서 활용된다. MPI\_Alltoall의 경우 INTRA 및 INTER 두 가지 경우에 대하여 동일한 알고리즘이 적용된다. 크기가 다른 데이터 전송을 위한 루틴(예: MPI\_Alltoallv) 또는 non-blocking 기반 루틴(예: MPI\_Ibcast)에도 동일하게 적용 가능하며 설정 방법은 아래와 같다.

- MVAPICH2 라이브러리 collective communication 주요 환경 변수

MPI 통신	환경변수	설정범위
MPI_Gather	<b>MV2_INTRA</b> _GATHER_TUNING	0 ~ 3
	<b>MV2_INTER</b> _GATHER_TUNING	
MPI_Bcast	<b>MV2_INTRA</b> _BCAST_TUNING	0 ~ 10
	<b>MV2_INTER</b> _BCAST_TUNING	
MPI_Scatter	<b>MV2_INTRA</b> _SCATTER_TUNING	1 ~ 5
	<b>MV2_INTER</b> _SCATTER_TUNING	
MPI_Allreduce	<b>MV2_INTRA</b> _ALLREDUCE_TUNING	1 ~ 6
	<b>MV2_INTER</b> _ALLREDUCE_TUNING	
MPI_Reduce	<b>MV2_INTRA</b> _REDUCE_TUNING	1 ~ 6
	<b>MV2_INTER</b> _REDUCE_TUNING	
MPI_Allgather	<b>MV2_INTRA</b> _ALLGATHER_TUNING	1 ~ 11
	<b>MV2_INTER</b> _ALLGATHER_TUNING	
MPI_Alltoall	MV2_ALLTOALL_TUNING	0 ~ 4

※ 사용 예시 (bash 스크립트 작성 경우) : export MV2\_INTER\_ALLREDUCE\_TUNING=2

※ 각 환경변수에 대한 자세한 설명은 <http://mvapich.cse.ohio-state.edu/userguide/> 참조

## III-7. 딥러닝 프레임워크 병렬화

### 가. Tensorflow에서 Horovod 사용법

다중노드에서 CPU를 활용할 경우 Horovod를 Tensorflow와 연동하여 병렬화가 가능하다. 아래 예시와 같이 Horovod 사용을 위한 코드를 추가해주면 Tensorflow와 연동이 가능하다. Tensorflow 및 Tensorflow에서 활용 가능한 Keras API 모두 Horovod와 연동이 가능하며 우선 Tensorflow에서 Horovod와 연동하는 방법을 소개한다.  
(예시: MNIST Dataset 및 LeNet-5 CNN 구조)

※ Tensorflow에서 Horovod 활용을 위한 자세한 사용법은 Horovod 공식 가이드 참조 (<https://github.com/horovod/horovod#usage>)

#### 1. Tensorflow에서 Horovod 사용을 위한 import 및 메인 함수에서 Horovod 초기화

```
import horovod.tensorflow as hvd
...
hvd.init()
```

- ※ horovod.tensorflow: Horovod를 Tensorflow와 연동하기 위한 모듈
- ※ Horovod를 사용하기 위하여 초기화한다.

#### 2. 메인 함수에서 Horovod 활용을 위한 Dataset 설정

```
(x_train, y_train), (x_test, y_test) = \
keras.datasets.mnist.load_data('MNIST-data-%d' % hvd.rank())
```

- ※ 각 작업별로 접근할 dataset을 설정하기 위하여 Horovod rank에 따라 설정 및 생성한다.

### 3. 메인 함수에서 optimizer에 Horovod 관련 설정 및 broadcast, 학습 진행 수 설정

```
opt = tf.train.AdamOptimizer(0.001 * hvd.size())
opt = hvd.DistributedOptimizer(opt)
global_step = tf.train.get_or_create_global_step()
train_op = opt.minimize(loss, global_step=global_step)
hooks = [hvd.BroadcastGlobalVariablesHook(0),
         tf.train.StopAtStepHook(last_step=20000 // hvd.size()), ... ]
```

- ※ Optimizer에 Horovod 관련 설정을 적용하고 각 작업에 broadcast를 활용하여 전달함
- ※ 각 작업들의 학습 과정 step을 Horovod 작업 수에 따라 설정함

### 4. Inter operation 및 Intra operation의 병렬처리 설정

```
config = tf.ConfigProto()
config.intra_op_parallelism_threads = int(os.environ['OMP_NUM_THREADS'])
config.inter_op_parallelism_threads = 2
```

- ※ config.intra\_op\_parallelism\_threads: 연산 작업에서 사용할 thread 개수를 설정하는 데 사용되며 job script에서 설정한 OMP\_NUM\_THREADS를 불러와서 적용함(본 예시의 경우 OMP\_NUM\_THREADS를 32로 설정함)
- ※ config.intra\_op\_parallelism\_threads: TensorFlow의 작업을 동시에 실행하는 thread 개수이며 예시와 같이 2로 설정할 경우 두 개의 작업이 병렬적으로 실행됨

### 5. Rank 0 작업에 Checkpoint 설정

```
checkpoint_dir = './checkpoints' if hvd.rank() == 0 else None
...
with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
                                       hooks=hooks,
                                       config=config) as mon_sess:
```

- ※ Checkpoint 저장 및 불러오는 작업은 하나의 프로세스에서 수행되어야 하므로 rank 0번에 설정함

## 나. Intel Caffe에서 다중노드 사용법

Caffe의 다중노드 병렬화는 Horovod에서 공식적으로 지원하지 않으며 Intel에서 KNL에 최적화하여 개발한 Intel Caffe를 활용하여 병렬처리가 가능하다. Intel Caffe의 경우 병렬처리를 위한 작업들이 모두 코드 개발과정에서 적용되어 기존 Caffe에서 개발된 deploy.prototxt, solver.prototxt, train\_val.prototxt들을 그대로 사용할 수 있다.

※ Intel Caffe 활용을 위한 자세한 사용법은 Intel Caffe 공식 가이드 참조  
(<https://github.com/intel/caffe/wiki/Multinode-guide>)

딥러닝 개발자에 의해 수정된 Caffe의 코드에 대하여 병렬처리를 수행할 경우 Intel Caffe 소스코드에 해당 부분을 업데이트하여 컴파일 후 실행하여야 한다.

- Intel Caffe 병렬처리 수행 방법 (작업 스크립트 예제)

```
#!/bin/sh
#PBS -N test
#PBS -V
#PBS -l select=4:ncpus=68:mpiprocs=1:ompthreads=68
#PBS -q normal
#PBS -l walltime=04:00:00
#PBS -A caffe

cd $PBS_O_WORKDIR

module purge
module load conda/intel_caffe_1.1.5
source /apps/applications/miniconda3/envs/intel_caffe/mlsl_2018.3.008/intel64/bin/mlslvars.sh

export KMP_AFFINITY=verbose,granularity=fine,compact=1
export KMP_BLOCKTIME=1
export KMP_SETTINGS=1

export OMP_NUM_THREADS=60
mpirun -PSM2 -prepend-rank caffe train \
--solver ./models/intel_optimized_models/multinode/alexnet_4nodes/solver.prototxt

# 혹은
```



```
./scripts/run_intelcaffe.sh --hostfile $PBS_NODEFILE \  
--caffe_bin /apps/applications/miniconda3/envs/intel_caffe/bin/caffe \  
--solver models/intel_optimized_models/multinode/alexnet_4nodes/solver.prototxt \  
--network opa --ppn 1 --num_omp_threads 60  
  
exit 0
```

- ※ Network 옵션: Intel Onmi-Path Architecture (OPA)로 설정
- ※ PPN: Process per node의 약자로 노드 당 작업 수를 뜻함 (기본값: 1)
- ※ Script를 이용하지 않고도 MPI 실행을 통해 기존 Caffe 수행 방법과 동일하게 수행 가능

## Ⅲ-8. 공통 라이브러리 목록

### 가. 공통 라이브러리

이름	버전	경로
antlr	2.7.7	/app/common/antlr
bison	2.7	/apps/common/bison
boost	1.68.0	/app/common/boost
cairo	1.14.6	/apps/common/cairo
CGAL	4.13 4.9.1	/app/common/CGAL
cnvgrib	3.1.1	/app/common/cnvgrib
curl	7.59.0	/apps/common/curl
expat	2.2.5	/apps/common/expat
flex	2.6.4	/app/common/flex
fontconfig	2.13.0	/apps/common/fontconfig
freetype	2.9.1 2.9.5	/apps/common/freetype
g2clib	1.5.0	/apps/common/g2clib
g2lib	3.1.0	/app/common/g2lib
gdal	2.1.1	/apps/common/gdal
glib	2.50.3 2.52.3	/app/common/glib
gmp	6.1.2	/app/common/gmp
gperf	3.1	/apps/applications/gperf
gsl	2.5	/app/common/gsl
grib_api	1.26.1	/apps/common/grib_api
gv	3.7.4	/apps/common/gv
iconv	1.15	/apps/common/iconv
jasper	1.900.29	/apps/common/jasper
jpeg	9c	/apps/common/jpeg
libffi	3.2.1	/app/common/libffi
libpng	1.2.56	/apps/common/libpng
libtiff	4.0.9	/app/common/libtiff
libtool	2.4.3	/app/common/libtool
memkind	1.9.0	/app/common/memkind
motif	2.3.8	/app/common/motif
mpc	1.1.0	/app/common/mpc
mpfr	4.0.1	/app/common/mpfr

ncurses	6.1	/apps/common/ncurses
pixman	0.34.0	/apps/common/pixman
pkg-config	0.29.2	/apps/common/pkg-config
proj	4.9.2	/apps/common/proj
readline	7.0	/apps/common/readline
scons	2.3.0	/app/common/scons
swig	3.0.12	/apps/common/swig
szip	2.1.1	/apps/common/szip
termcap	1.3.1	/app/common/termcap
udunits	2.2.24	/apps/common/udunits
voro++	0.4.6	/app/common/voro++
VTK	7.1.1	/apps/common/VTK
w3lib	2.0.6	/app/common/w3lib
zlib	1.2.11	/apps/common/zlib

### Ⅲ-9. 데스크톱 가상화(VDI)

#### 가. VDI 서비스 개요

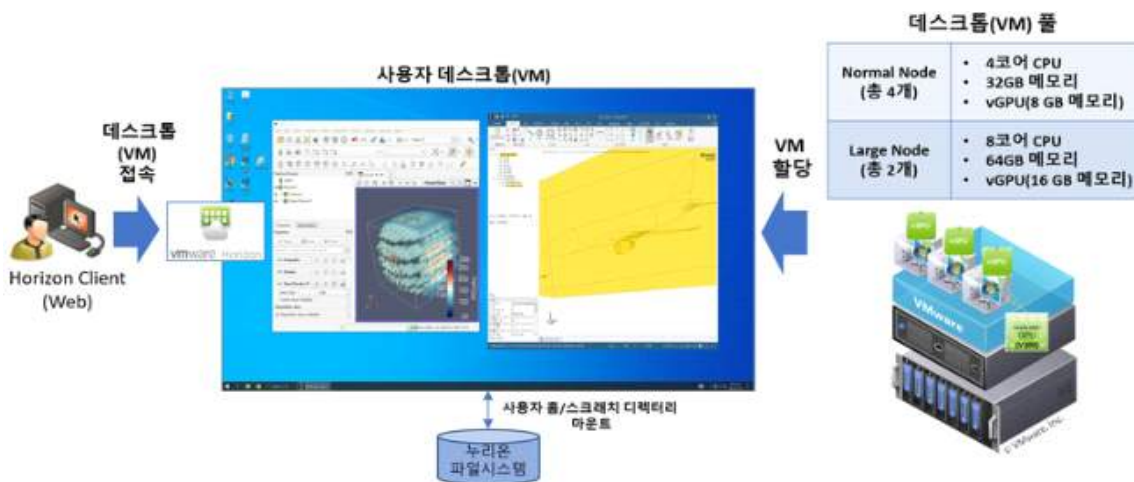
데스크톱 가상화(VDI)는 MS Windows 10 환경에서 고성능 그래픽 처리 기능이 필요한 애플리케이션 실행을 위한 원격 데스크톱 서비스다. VDI는 누리온의 파일시스템(사용자홈 및 스크래치 디렉터리)을 마운트하여 별도 파일 이동 없이 데스크톱(VM)에서 공학해석 전·후처리가 가능하다. 데스크톱(VM)에서 해석(solving)은 지원되지 않으며, 데스크톱(VM) 내에서 터미널 접속 프로그램(putty 등)을 통해 누리온 시스템으로 해석을 수행하면 된다.

데스크톱(VM) 풀은 아래와 같이 Normal Node와 Large Node로 구분되며 사용자에게 적절한 데스크톱(VM) 풀을 선택하면 된다. 한정된 데스크톱(VM)으로 다수의 사용자 지원을 위해 데스크톱(VM) 구동 후 idle 상태로 2시간이 경과하면 연결이 해제되고 자동 종료된다.

- Normal Node (총 8개\*) : 4코어 CPU, 32GB 메모리, 8GB GPU 메모리 할당  
\* VM 수는 조정될 수 있음

현재 공학해석 전·후처리를 위해 아래와 같은 소프트웨어를 지원하고 있으며, 향후 사용자 요구에 따라 지원 애플리케이션을 확대할 예정이다.

- Ansys (Workbench, SpaceClaim, CFD-Post 등) v2020 R2, v2023 R2
- Abaqus (CAE, Viewer) v2020, v2023
- Paraview v5.11.1, VAPOR v3.9.0
- Salome-Meca v2023



[VDI 서비스 구성도]

## 나. VDI 사용 방법

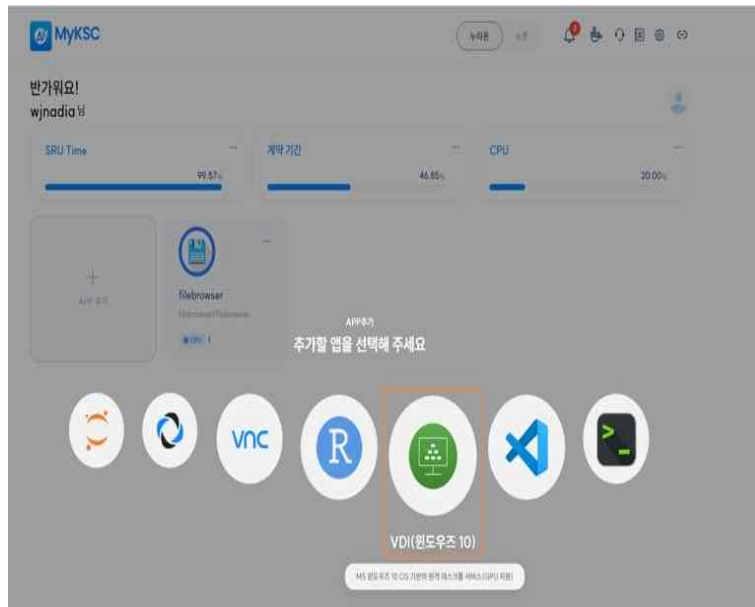
### 1. MyKSC 접속

- 웹 브라우저를 사용하여 MyKSC (KISTI 슈퍼컴퓨터 웹 서비스 포털, <https://my.ksc.re.kr>)에 접속한다.
- 슈퍼컴퓨터 사용자 인증을 위해 ID, 비밀번호, OTP 번호를 입력하고 Nurion 시스템을 선택하여 로그인한다.
- 사용자 인증이 성공하면 MyKSC 대시보드에 접속된다.



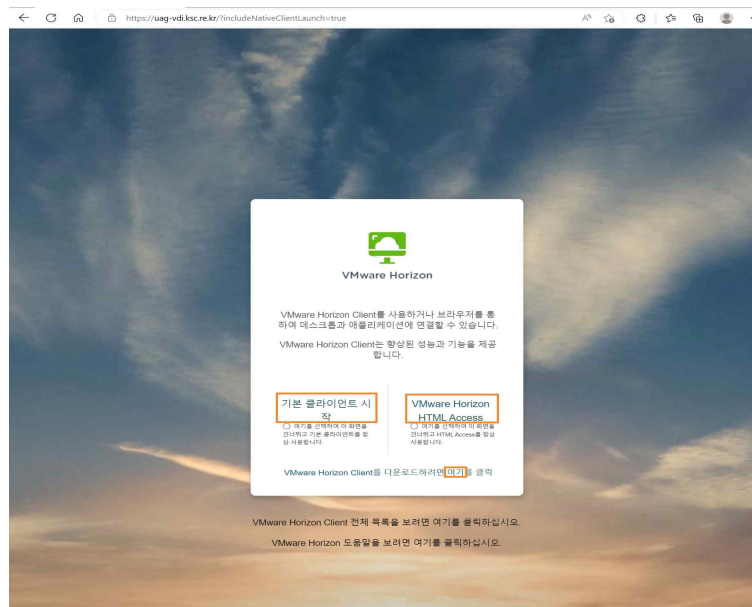
### 2. 데스크톱 가상화(VDI) 서버 접속

- MyKSC 대시보드에서 APP 추가(+) 버튼을 클릭하고 VDI(윈도우즈클10)를 선택하면 별도 창에서 VMware Horizon 페이지가 구동된다.



[대시보드에서 VDI 서비스 시작]

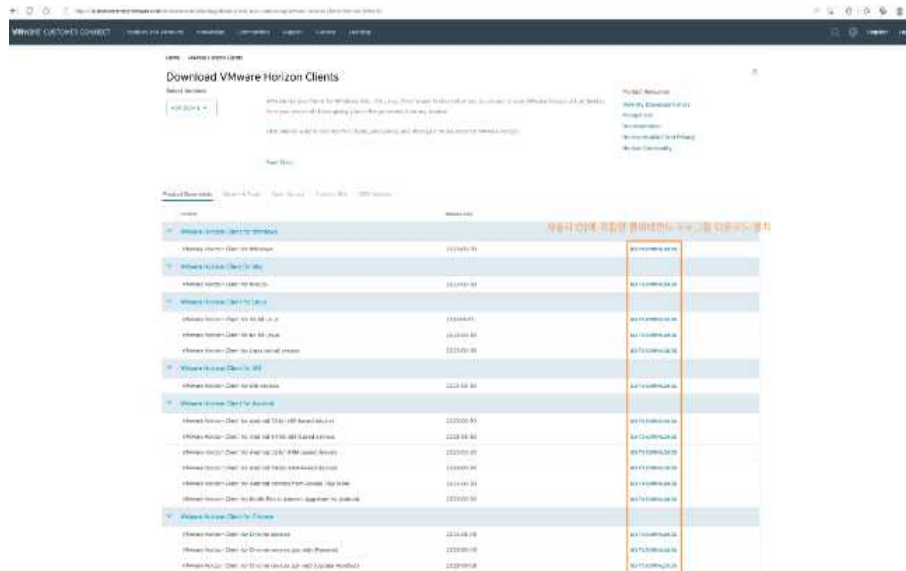
- VDI 서비스는 'VMware Horizon 클라이언트 프로그램' 또는 '웹 브라우저'를 통해 사용할 수 있다.
  - \* 안정성 측면에서 웹 브라우저보다는 클라이언트 프로그램 사용을 권장함



[웹 브라우저 또는 VMware Horizon Client 프로그램을 통한 VDI 서버 접속]

## 1) VMware Horizon Client 프로그램

- 처음 사용자는 'VMware Horizon Client를 다운로드하려면 여기를 클릭'을 클릭하여 프로그램을 설치 후 '기본 클라이언트 시작' 클릭
- Horizon Client의 '서버 추가'를 클릭하여 '연결 서버의 이름'으로 'https://uag-vdi.ksc.re.kr' 입력 후 연결
- 기존 VDI 사용자의 Horizon Client에서 연결 서버로 사용되었던 https://nurion-vdi.ksc.re.kr은 더 이상 사용할 수 없음



[VMware Horizon Client 프로그램 다운로드]



[VMware Horizon Client 프로그램에서 VDI 서버 추가]

## 2) 웹 브라우저

- 'VMware Horizon HTML Access' 클릭

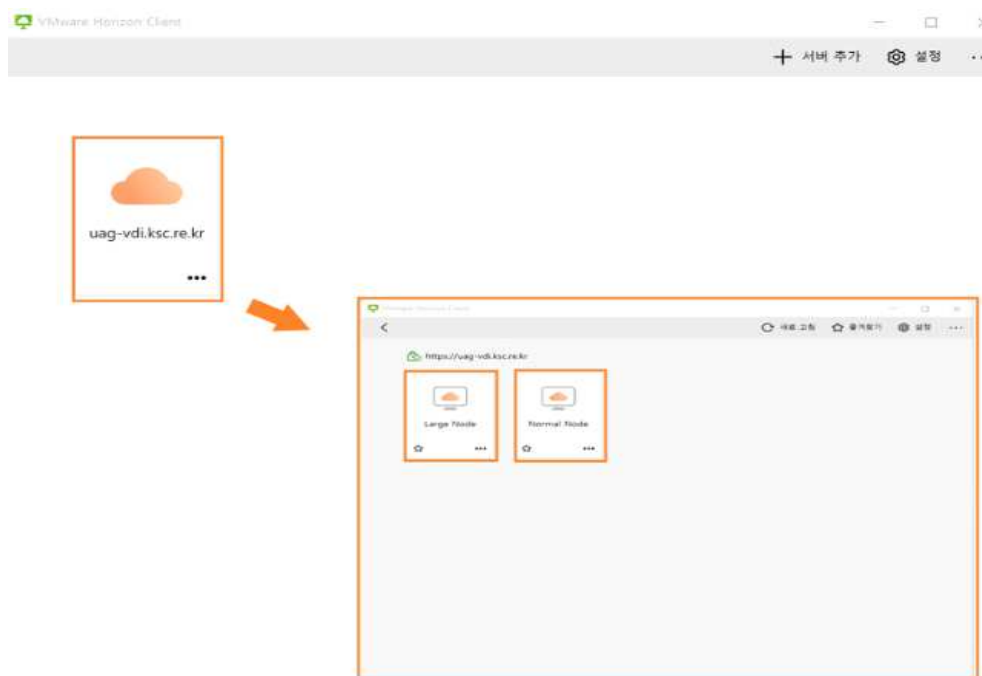
### 3. 사용자 데스크톱 연결 로그인

Horizon 웹 또는 프로그램에서 Large Node 또는 Normal Node 데스크톱 풀을 선택하여 VM 기반의 사용자 데스크톱에 연결한다. 가용 VM이 모두 사용 중인 경우 VM이 할당되지 않을 수 있다.

- Normal Node (총 8개\*) : 4코어 CPU, 32GB 메모리, 8GB GPU 메모리 할당  
\* VM 수는 조정될 수 있음



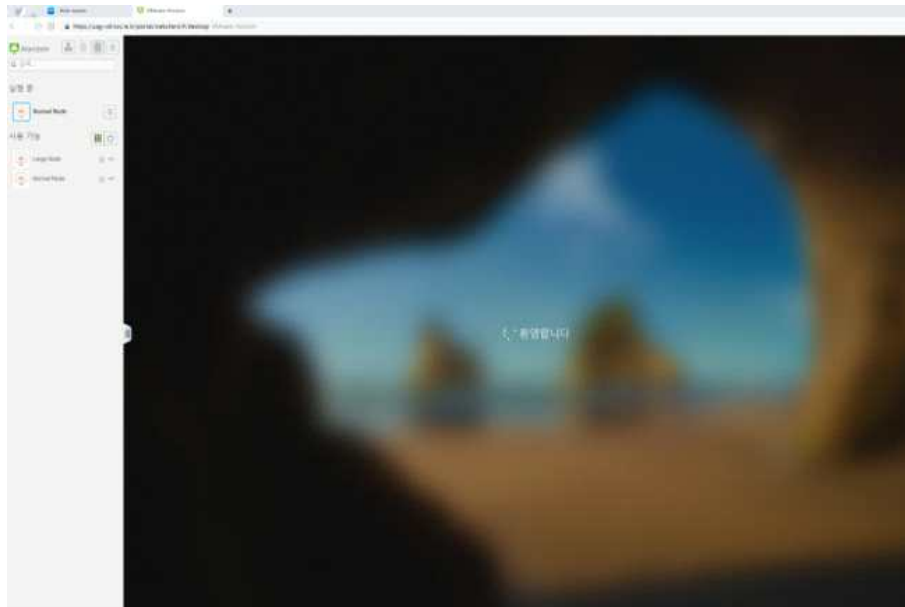
[웹 브라우저를 통한 VDI 서버 접속 화면]



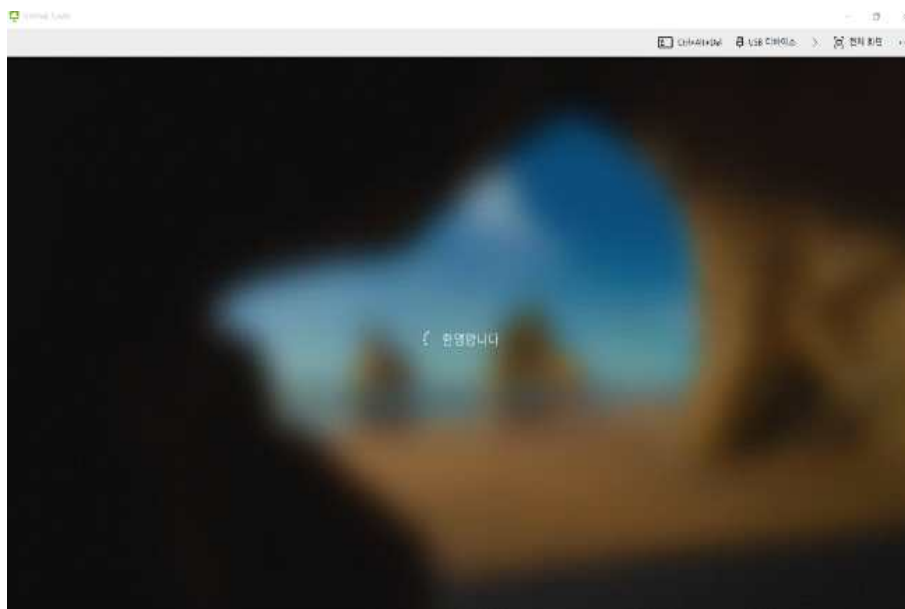
[VMware Horizon Client 프로그램을 통한 VDI 서버 접속 화면]



- 데스크톱(VM)이 할당되면 윈도우즈 10 OS에 연결 로그인되며 약 30초 이내 소요됩니다.

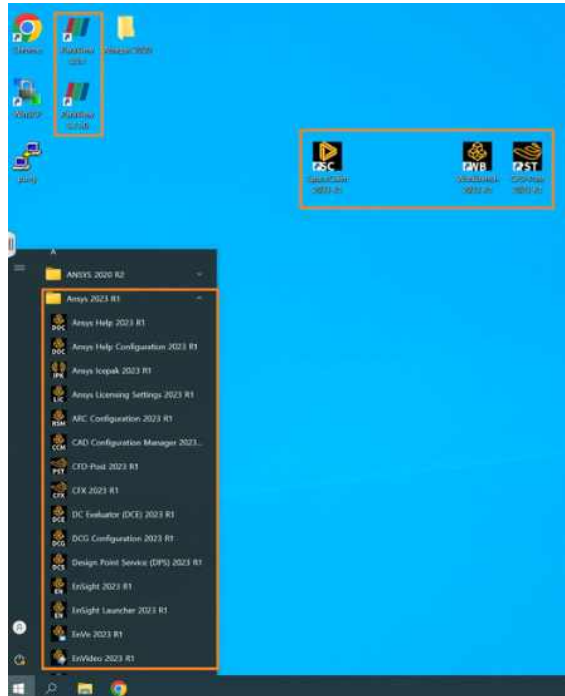


[웹 브라우저를 통한 사용자 데스크톱(VM) 연결 로그인 화면]



[VMware Horizon Client 프로그램을 통한 사용자 데스크톱(VM) 연결 로그인 화면]

- 바탕화면 혹은 시작-프로그램 목록에서 설치된 소프트웨어 목록을 확인할 수 있다.
  - Ansys (Workbench, SpaceClaim, CFD-Post 등) v2020 R2, v2023 R2
  - Abaqus (CAE, Viewer) v2020, v2023
  - Paraview v5.11.1
  - Salome-Meca v2023

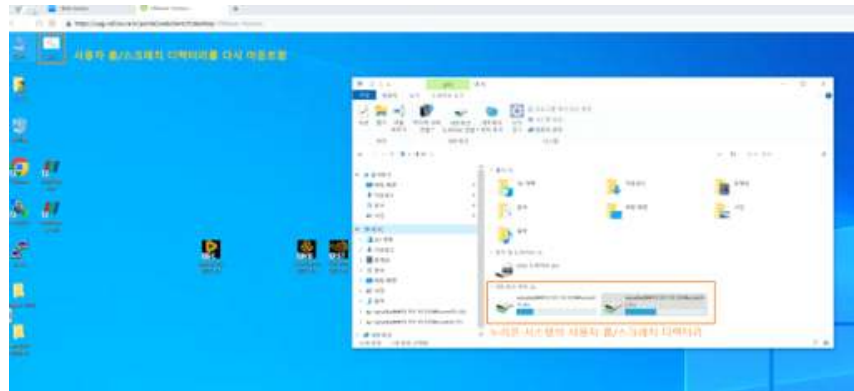


[사용자 데스크톱(VM) 설치 소프트웨어 목록]

사용자 데스크톱의 로그인 과정에서 누리온 시스템의 home01과 scratch의 사용자 디렉터리가 자동으로 마운트되며, 마운트가 되지 않거나 오류가 있을 경우 바탕화면의 'NFS' 아이콘을 실행한다.

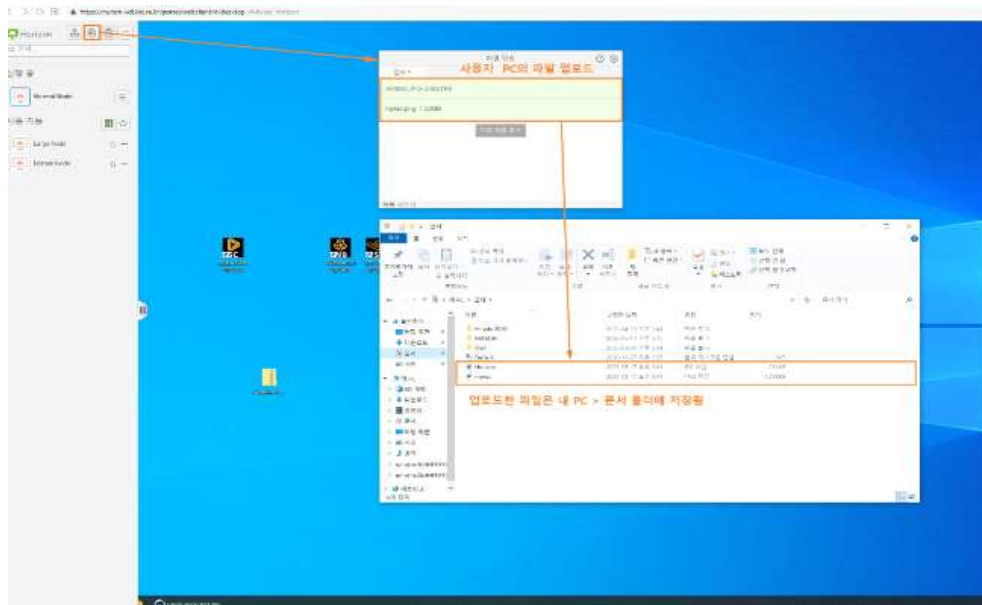
보안을 위해 C:\ 드라이브, 제어판 등은 접근이 차단되어 있으며, 사용자가 생성 및 저장한 파일들은 VM 종료 시 모두 초기화 된다. 따라서, 계속 보관이 필요한 파일들은 누리온 시스템의 사용자 홈 및 스크래치 디렉터리에 저장해야 한다.

데스크톱(VM) 구동 후 idle 상태로 2시간이 경과하면 연결이 해제되고 자동 종료된다.



[사용자 데스크톱(VM)에서 누린 시스템 사용자 홈/스크래치 디렉터리]

웹 브라우저를 통해 접속된 데스크톱에서는 왼쪽 메뉴 패널의 ‘파일 전송 패널 열기’를 실행하여 사용자 PC의 파일을 업로드할 수 있음

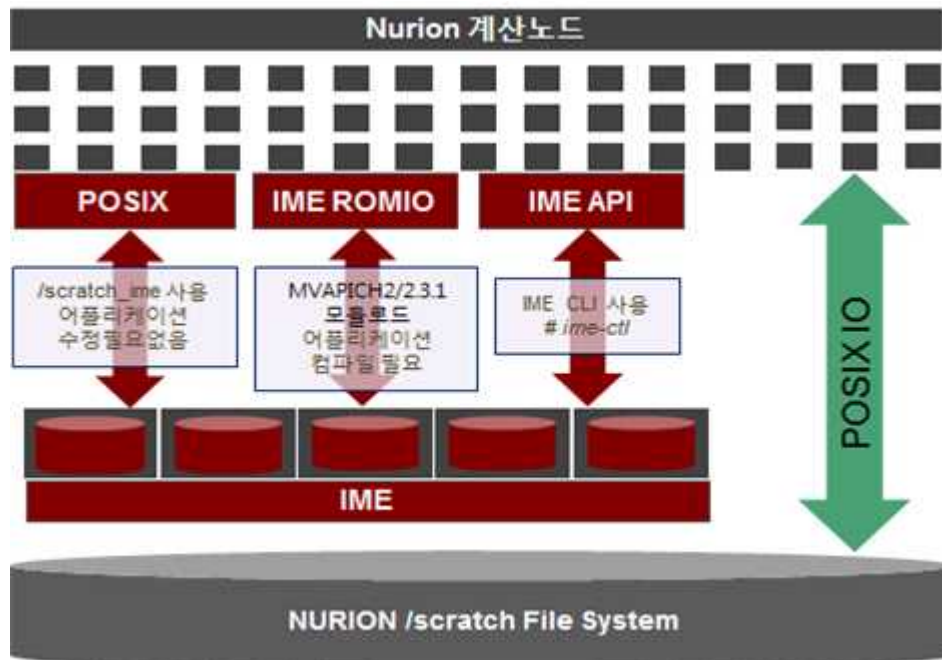


[웹 브라우저를 통해 연결된 사용자 데스크톱(VM)에서 PC의 파일 업로드하기]

### III-10. 버스트버퍼(Burst Buffer)

#### 가. 개념

버스트버퍼 IME는 Nurion /scratch 파일시스템의 캐시 역할을 수행한다. IME를 통한 데이터 접근 방법은 아래 그림과 같다.



IME는 사용자 레벨 파일시스템인 FUSE(File System In **USER**space)를 사용하여 클라이언트 노드(전체 계산노드와 로그인노드)에 마운트 되어 있다. 이때 유의할 것은 IME는 캐시 역할을 수행하므로 /scratch 파일시스템이 사전에 반드시 마운트 되어 있어야 한다. IME 디렉터리 경로는 **/scratch\_ime**이며 최초 사용자가 해당 디렉터리(/scratch\_ime/\$USER)에 접속하면 스크래치(/scratch/\$USER) 파일시스템의 모든 디렉터리와 파일의 구조를 동일하게 확인할 수 있다.

이는 실제 IME 디바이스에 존재하지 않는 데이터이며, 버스트버퍼를 이용하여 작업 수행 시 /scratch에서 IME에 캐싱하는 방식으로 사용된다. IME를 사용하기 위해서는 작업 스크립트에 버스트버퍼 프로젝트명(#PBS -P burst\_buffer)을 명시해야 한다. 어플리케이션을 수행하는 방법은 크게 아래와 같이 두 가지 방식이 존재한다.

1. IME의 마운트 포인트인 /scratch\_ime를 입출력 디렉터리로 지정하여 사용하는 방식으로 별도의 컴파일 없이 기존 POSIX 기반의 I/O를 수행할 수 있다. 즉, 사용자는 기존의 방식대로 작업을 제출하되, 자료 입출력 자료 경로만을 /scratch\_ime/\$USER/ 하단에 설정하면 된다.

- ex) INPUT="/scratch\_ime/\$USER/input.dat",  
OUTPUT="/scratch\_ime/\$USER/output.dat"

```
#!/bin/sh
#PBS -N burstbuffer
#PBS -V
#PBS -q normal # 모든 큐 사용가능
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -P burst_buffer # 버스트 버퍼 사용을 위해서 반드시 명시
#PBS -l select=2:ncpus=16:mpiprocs=16
#PBS -l walltime=05:00:00

cd $PBS_O_WORKDIR

OUTFILE=/scratch_ime/$USER/output.dat

# 이하 해당 작업 관련 실행명령어 작성 (4장 "스케줄러를 통한 작업실행" 나절 예제 참조)
```

2. MPI-IO 기반의 I/O를 사용하기 위해서는 IME를 지원하는 mvapich2/2.3.1 모듈을 이용해야 수행이 가능하다. 응용 프로그램 또한 이 MPI 라이브러리를 이용하여 다시 컴파일 해야 한다. 또한 파일 또는 디렉터리의 경로는 아래 예시와 같이 IME 프로토콜을 사용하여 경로를 지정한다.

- ex) OUTFILE=ime:///scratch/\$USER/output.dat (아래 작업스크립트 예제 참조)

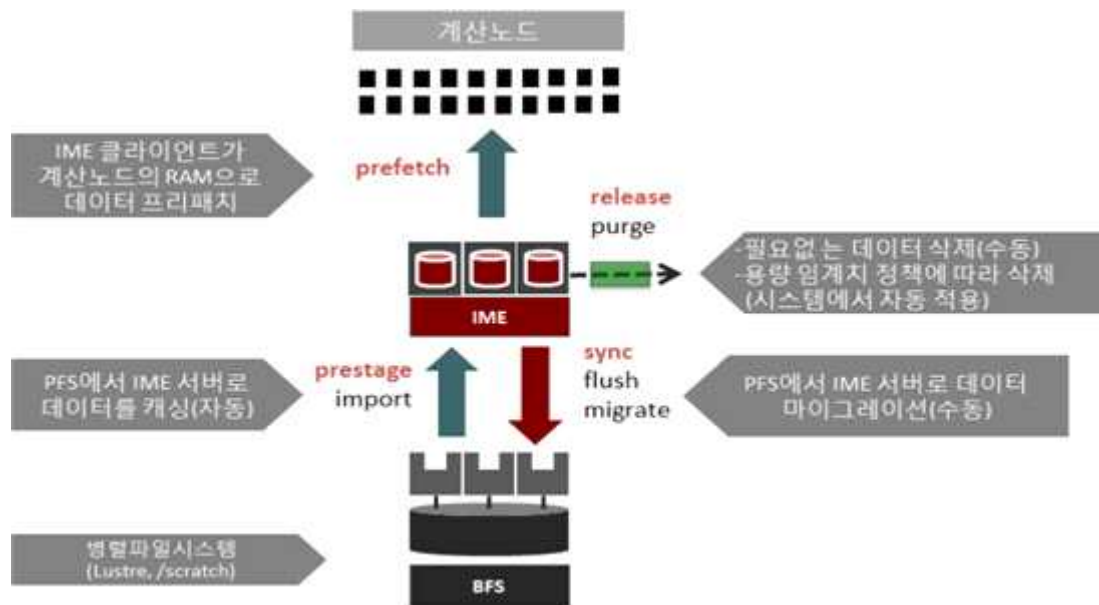
```
$ module load mvapich2/2.3.1
```

위와 같이 mvapich2/2.3.1 모듈을 로드하고 아래와 같이 작업 스크립트를 작성한다.

```
#!/bin/sh
#PBS -N mvapich2_ime
#PBS -V
#PBS -q normal # KNL에 해당하는 모든 큐 사용 가능 (전용큐, normal, long, flat, debug)
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -P burst_buffer # 버스트 버퍼 사용을 위해 반드시 명시
#PBS -l select=2:ncpus=16:mpiprocs=16
#PBS -l walltime=5:00:00
cd $PBS_O_WORKDIR
TOTAL_CPUS=$(wc -l $PBS_NODEFILE
```

- ※ 지원컴파일러: gcc/6.1.0, gcc/7.2.0, intel/17.0.5, intel/18.0.1, intel/18.0.3, intel/19.0.4, pgi/18.10
- ※ IME의 MPI-IO는 별도의 ROMIO 인터페이스를 사용하여 구현되었으나, MVAPICH2/2.3.1 버전부터 공식적으로 IME를 지원하는 ROMIO 기능이 포함됨. (OpenMPI는 미지원)
- ※ 버스트버퍼 IME는 NURION 전체 계산노드(SKL, KNL)에서 사용이 가능함.

- IME의 데이터 관리를 위해서는 아래 그림에 있는 데이터의 라이프 사이클을 파악해야 한다. IME 데이터 처리는 크게 Prestage, Prefetch, Sync, Release 단계가 있으며 각각에 대해 IME-API(#ime-ctl) 명령을 제공하고 있다.



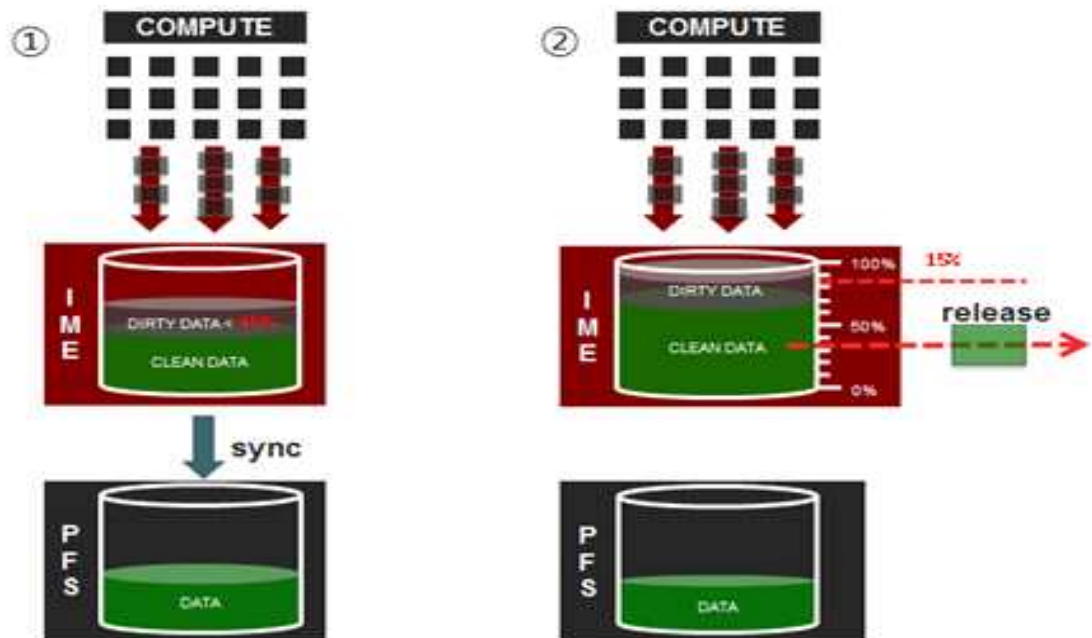
ime-ctl -i \$INPUT_FILE	작업 데이터를 IME로 Stage-In 실행 (/scratch에서 /scratch_ime로 데이터 캐싱)
ime-ctl -r \$OUTPUT_FILE	IME 데이터를 병렬파일시스템과 동기화 (/scratch_ime의 데이터를 /scratch로 전송)
ime-ctl -p \$TMP_FILE	IME에서 데이터 삭제 (/scratch_ime의 데이터를 삭제(purge))
ime-ctl -s \$FILE	IME 데이터의 상태정보 제공

※ #ime-ctl --help를 통해 상세 옵션 확인 가능

## 나. 데이터 처리

IME의 총 용량은 약 900TB이며 사용량에 따라 자동으로 /scratch 파일시스템에 플러싱 되거나 삭제된다. IME에서는 아래와 같이 2가지의 임계치 설정에 따라 자동으로 캐시 공간을 확보한다.

1. 새로 생성된 데이터(Dirty Data)의 총 용량이 45% 이상일 경우
2. 전체 여유 공간이 15% 이하일 경우



## 다. 유의 사항

IME는 초기 작업 수행 시 PFS의 데이터를 IME 디바이스로 캐싱하는 단계와 캐시 데이터를 다시 PFS로 flushing 또는 sync 해주는 작업의 부하가 발생한다. 따라서 PFS(Lustre)에서 상대적으로 취약한 대량의 small I/O, 잦은 checkpointing 또는 I/O 수행 빈도가 높은 어플리케이션에서 성능 향상을 기대할 수 있다.

또한 IME(약 0.9PB)는 PFS(약 20PB)의 캐시의 용도로 사용되기 때문에 상대적으로 용량이 작다. 따라서 사용 중에 IME의 용량이 차게 되면 임계치 설정에 따라 데이터가 캐시에서 제거될 수 있으므로 데이터 관리에 유의해야 한다.

※ 주의: IME에 캐시된 데이터를 삭제하기 위해서는 제공하는 IME-API 명령어를 사용해야 한다. 만약, /scratch\_ime에서 rm 명령을 통해 삭제할 경우 실제 /scratch에 저장된 데이터가 삭제되므로 반드시 주의가 필요하다.



### III-11. 플랫 노드(Flat node)

누리온 KNL 노드 중 MCDRAM (16GB)을 DDR4와 같이 사용할 수 있는 플랫 모드 (Flat mode)를 지원하는 노드를 일반 사용자에게 서비스하고 있으며, 해당 노드를 사용하기 위해서는 Flat 큐를 지정하여 작업을 제출해야 한다.

Flat 모드를 사용하기 위해서는 “numactl”이라는 명령어를 반드시 사용해야 하며, 이는 선호하는 메모리 모드 혹은 기본 설정(default) 메모리 모드를 특정하는 명령어이다. 예를 들어, “my\_app.x”라는 실행파일을 flat 모드에서 실행하고자 할 경우 numactl 명령어와 함께 “-m” 옵션으로 해당되는 NUMA 노드를 아래와 같이 명시할 수 있으나, my\_app.x 실행이 MCDRAM 16G 이상의 메모리를 필요로 할 경우, 메모리 부족으로 프로그램이 종료된다.

따라서, 아래와 같이 MCDRAM을 우선 사용한다는 옵션인 “-p”옵션을 활용하여 사용자 실행파일이 16G 이상의 메모리를 필요로 한 경우에도 작업이 종료되지 않도록 사용하는 방법을 권장한다.

- Flat 모드 작업 스크립트 작성 예제

```
#!/bin/sh
#PBS -N flat_job
#PBS -V
#PBS -q flat
#PBS -A {PBS 옵션 이름} # Application별 PBS 옵션 이름표 참고
#PBS -l select=1:ncpus=68:mpiprocs=32:ompthreads=1
#PBS -l walltime=12:00:00

cd $PBS_O_WORKDIR

mpirun numactl -m 1 my_app.x
또는
mpirun numactl -p 1 my_app.x
```

※ pbs option으로 제출할 큐는 반드시 flat 선택 (즉, -q flat)

## Ⅲ-12. DTN(데이터전송노드)

### 가. 시스템 환경 및 사용 신청

대용량 데이터를 활용한 초고성능컴퓨터의 사용이 증가하고 있는 환경에서 누리온에서는 대용량 데이터의 효율적인 전송을 위하여 Science DMZ 기반의 데이터 전송 전용 노드(DTN)를 구축하여 운영합니다. 누리온 DTN을 사용하기 위해서는 먼저 Globus(<https://globus.org>) Web App에 로그인 가능하여야 합니다. 이때 KAFE(<https://www.kafe.or.kr>)에 회원으로 가입되어 있다면 별도의 회원가입 없이 KAFE 계정을 활용하여 접속할 수 있습니다.

### 나. 사용자 정책 및 신청 방법

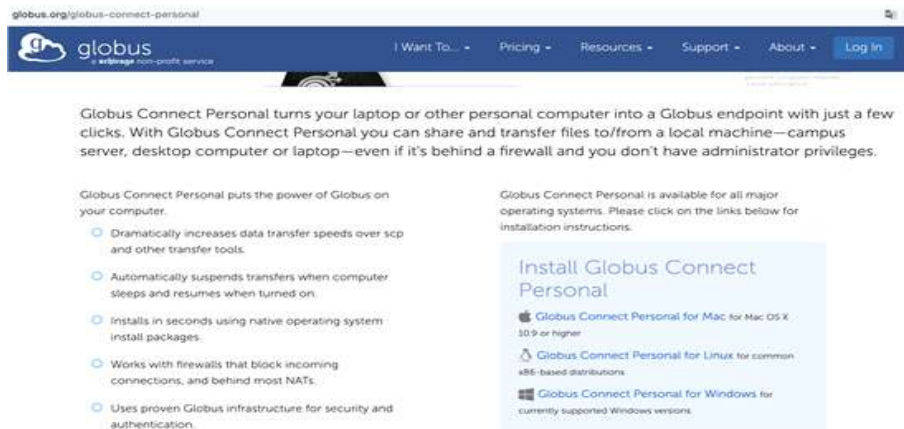
DTN 사용은 대표 홈페이지(<https://www.ksc.re.kr>)의 "기술지원 -> 상담" 메뉴 하단의 "상담신청"(상담 카테고리: DTN, 상담분류 DTN 계정발급)을 이용하여 신청하신 후 사용이 가능하며 /scratch 디렉터리에 직접 접근할 수 있습니다. 따라서 모든 사용 정책은 누리온의 /scratch 정책을 따라 갑니다. (15일 이상 접근하지 않은 파일은 삭제

※ 중요 데이터는 사용자 로컬 시스템에 반드시 보관해 주시는 것이 좋습니다. 일반적인 기술 지원 관련 문의는 [consult@ksc.re.kr](mailto:consult@ksc.re.kr) 메일이나 센터 홈페이지(<https://www.ksc.re.kr>) 상담 게시판을 이용해주시기 바랍니다.

### 다. 사용 방법

#### 1. Globus Web App 접속

Globus Web App을 보다 간편하게 접속하기 위하여 먼저 Globus Connect Personal (<https://www.globus.org/globus-connect-personal>)을 다운로드 받아 설치하시길 바랍니다. 해당 사이트에 접속하여 운영체제에 맞는 프로그램을 다운로드 받아 설치하시면 됩니다. (설치 관련하여서는 본 지침서에서 제공하지 않습니다. 해당 사이트를 참조하셔서 설치하시기 바랍니다.)



[Globus Connect Personal 설치 안내 페이지]

※ Globus Connect Personal 설치 및 설정 참조 :

<https://docs.globus.org/how-to/globus-connect-personal-windows/>

※ 로그인 방법 참조 : <https://docs.globus.org/how-to/get-started/>

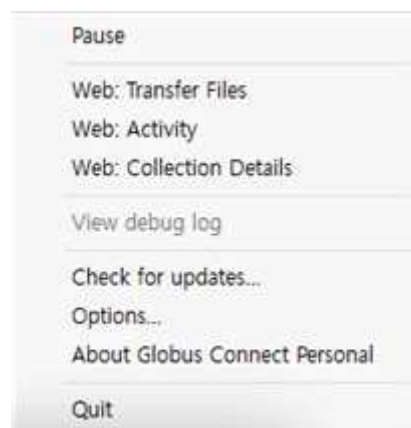
## 2. Globus Connect Personal 실행

- 프로그램을 설치하고 Globus Connect Personal을 실행하면 윈도우즈의 작업표시줄에 그림과 같이 아이콘이 나타나게 됩니다.



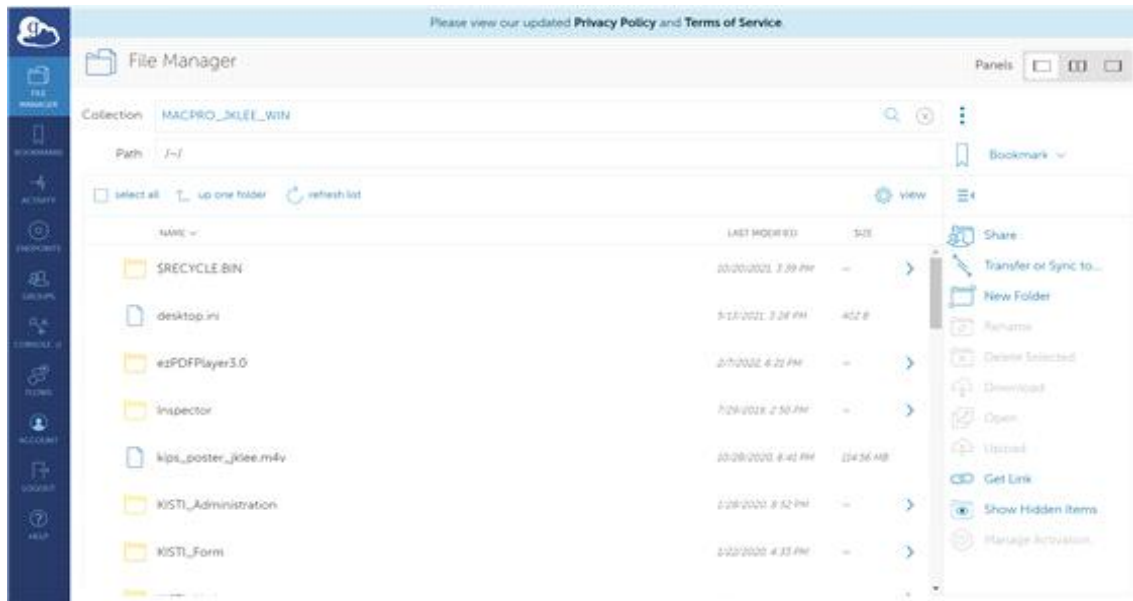
[작업표시줄의 Globus Connect Personal 아이콘]

- 아이콘에서 마우스 오른쪽 클릭을 하면 간단한 메뉴를 볼 수 있습니다.



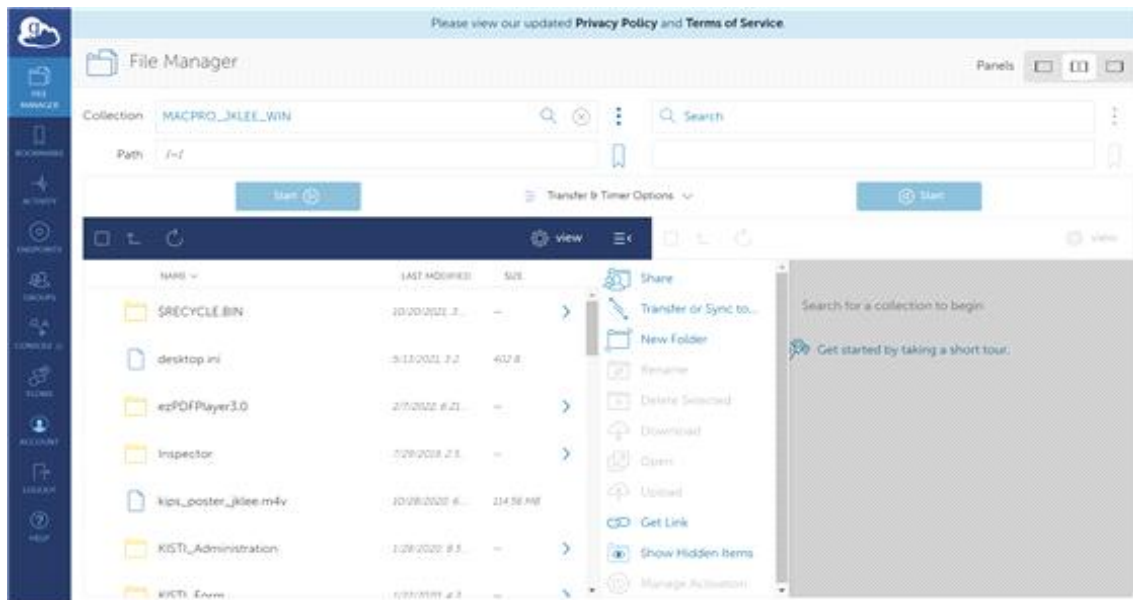
[Globus Connect Personal 간단 메뉴]

- 이 메뉴에서 “Web: Transfer Files”를 클릭하면 브라우저를 이용해서 파일을 송수신할 수 있습니다.



[Web: Transfer Files 실행화면(기본보기)]

- 브라우저에 보이는 패널의 모양은 “Web: Transfer Files” 실행 화면의 우측 상단에서 변경할 수 있습니다.



[Web: Transfer Files 실행화면 패널 보기 변경]

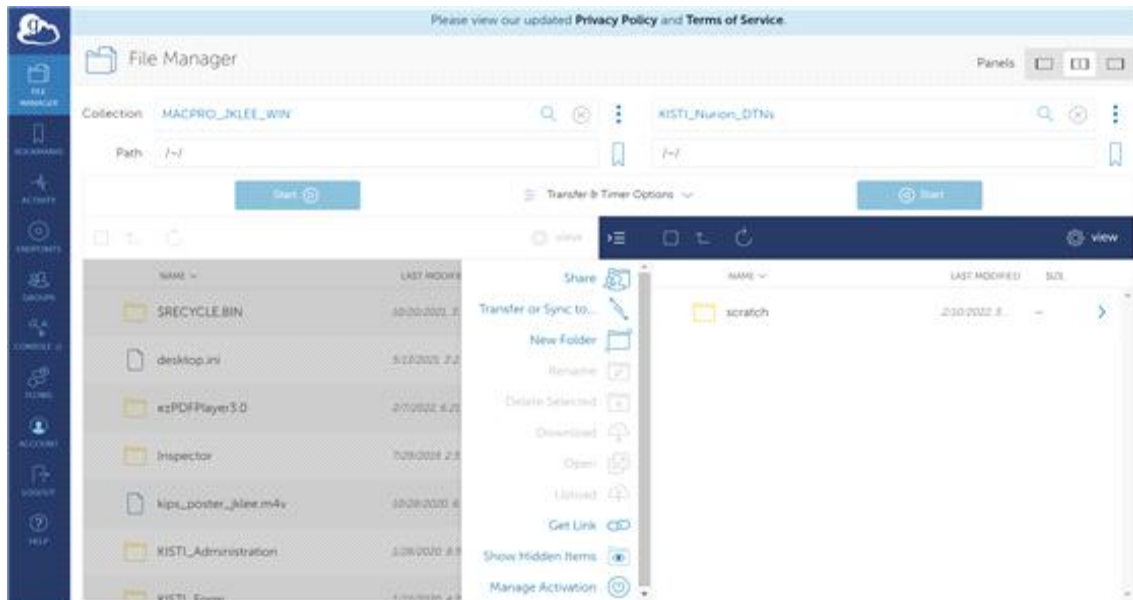
### 3. 누리온 DTN 검색 및 접속

- “Web: Transfer Files” 실행 화면의 검색창에 “nurion”이라는 키워드를 입력하고 검색을 하면 “KISTI\_Nurion\_DTNs”가 검색됩니다.



[KISTI\_Nurion\_DTN 검색 (검색키워드: nurion)]

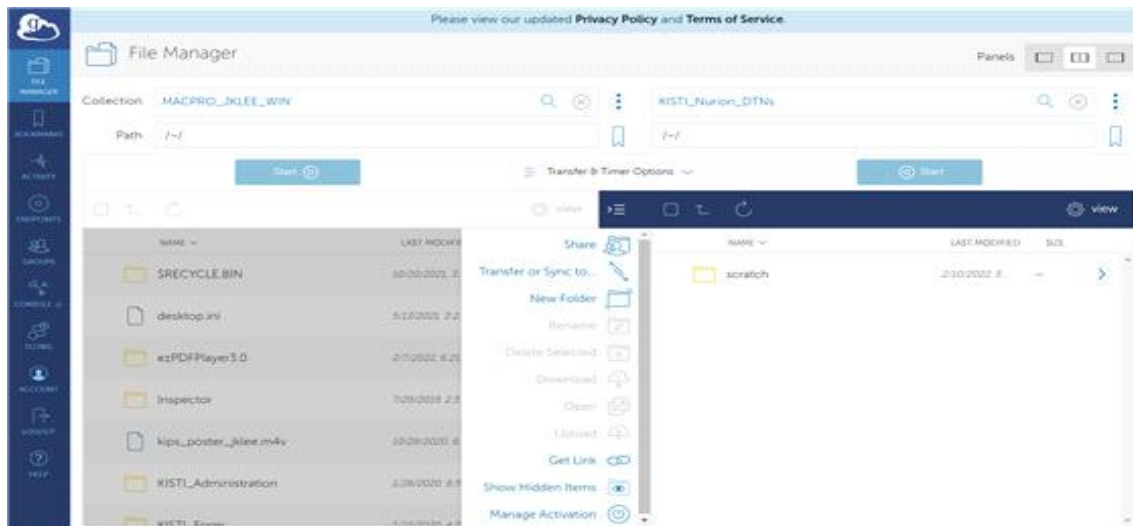
- 검색된 결과를 클릭하면 누리온의 /scratch 디렉터리에 접속됩니다.



[KISTI\_Nurion\_DTN 접속]

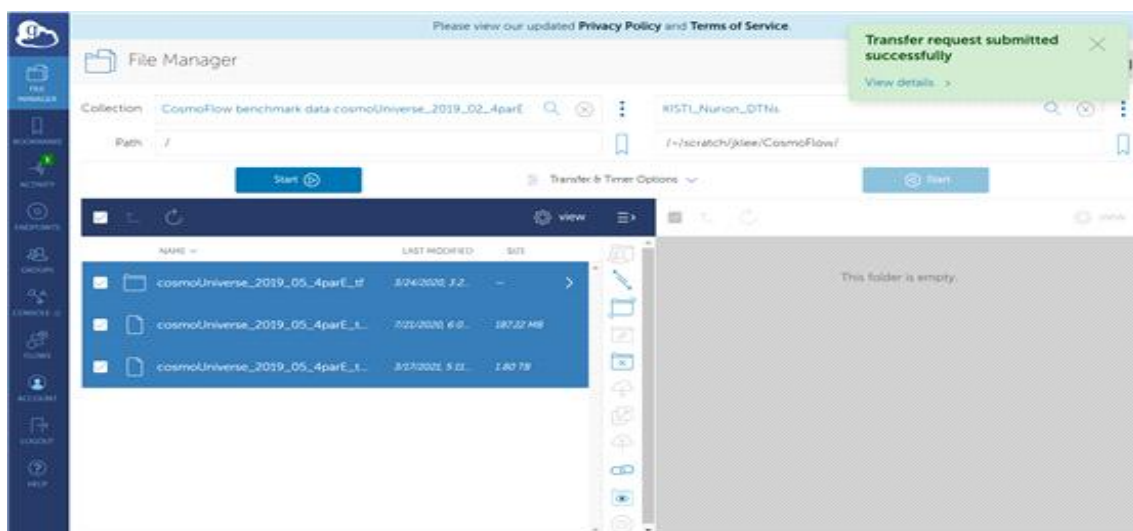
#### 4. 파일 전송

- Globus Web App에서는 개인 PC에 전송하려는 데이터를 선택한 후 상단에 있는 “Start” 버튼을 클릭하면 파일이 전송됩니다. 그리고, Globus에 가입된 해외 센터의 데이터를 누리온으로 직접 전송할 수도 있습니다.
- 예를 들어 NERSC에서 제공하는 “Climate segmentation benchmark data”나 “CosmoFlow benchmark data”도 누리온으로 바로 전송이 가능합니다. 누리온 DTN을 검색했던 것과 같은 방법으로 검색창에 키워드(ex) climate, cosmoflow, etc.)를 입력하고 검색한 후 전송을 원하는 데이터를 찾아서 전송(상단에 있는 “Start” 버튼 클릭 or 마우스 드래그 앤 드롭) 하면 바로 고속으로 누리온에 전송됩니다.



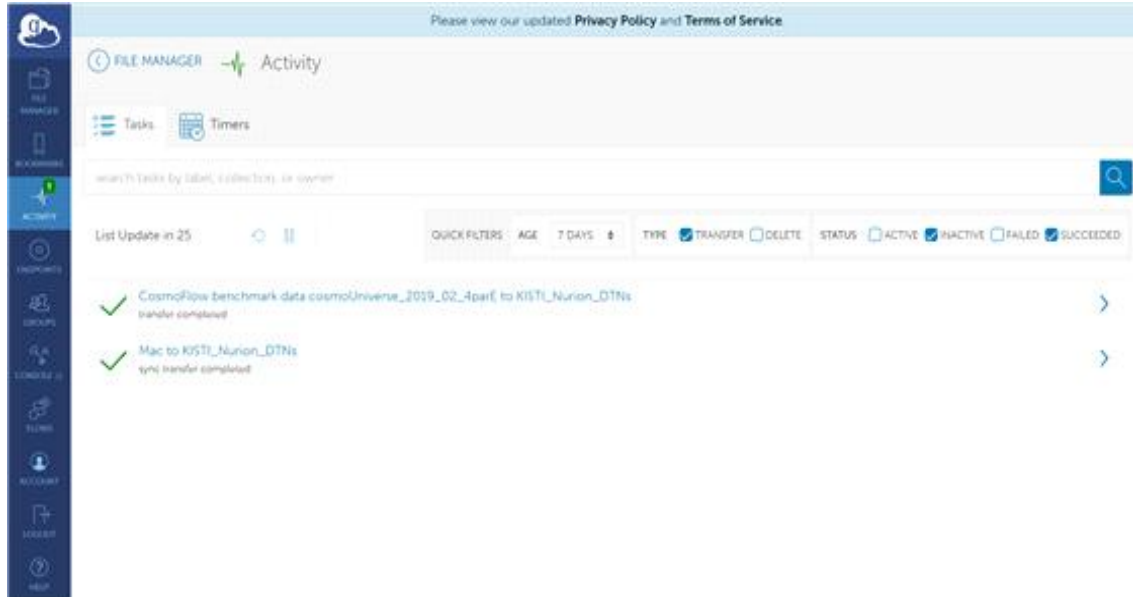
[CosmoFlow benchmark data(from NERSC to Nurion) 전송 예제]

- 파일 전송을 시작하면 창의 오른쪽 상단에 전송 요청이 성공되었다고 메시지 (Transfer request submitted successfully)가 나타납니다.



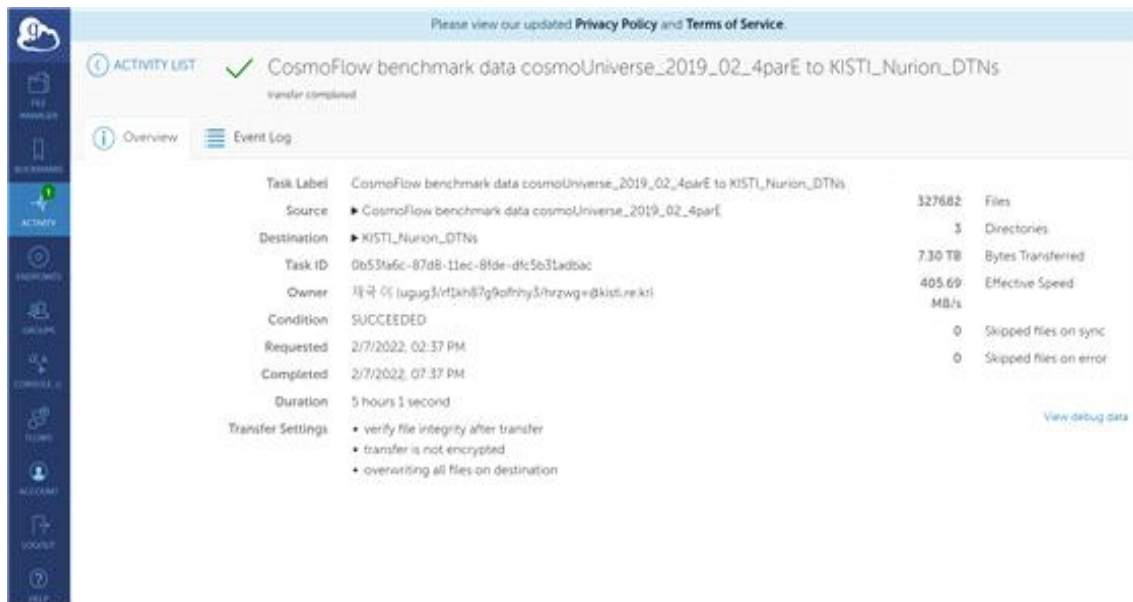
[데이터 전송 요청 성공 메시지 출력 화면]

- 파일이 전송되는 진행 상황은 왼쪽에 있는 “ACTIVITY” 탭에서 확인할 수 있습니다.



[데이터 전송 진행 현황 보기]

- “ACTIVITY” 탭에서 현재 진행 중인 데이터 전송 요청을 클릭하면 보다 상세한 진행 상황을 확인할 수 있습니다.



[데이터 전송 진행 현황 상세 보기]

- 창이 닫히더라도 데이터 전송은 정상적으로 진행이 되며 전송이 완료되면 메일로도 확인이 가능합니다.