슈퍼컴 5호기 누리온
사용자 지침서

(베타서비스용)

2018년 11월

슈퍼컴퓨팅서비스센터

1 누리온 시스템 사양 및 구성 1

가. 계산 노드 2

나. 인터커넥트 네트워크 4

다. 스토리지 5

2 사용자 환경 10

가. 계정발급 10

나. 로그인 10

다. 사용자 쉘 변경 12

라. 사용자 비밀번호 변경 13

마. 작업 디렉터리 및 쿼터 정책 13

3 사용자 프로그래밍 환경 16

가. 프로그래밍 도구 설치 현황 16

나. 프로그램 컴파일 17

다. 싱귤레러티(singularity) 컨테이너 이미지 사용법 21

4 스케줄러(PBS)를 통한 작업 실행 28

가. 큐 구성 28

나. 작업 제출 및 모니터링 32

다. 작업 제어 33

5 사용자 기술 지원 34

별첨1 작업 스크립트 주요 키워드 35

별첨2 누리온 소프트웨어 설치 목록 37

별첨3 Lustre stripe 사용법 46

1. 누리온 시스템 사양 및 구성

KISTI 슈퍼컴퓨터 5호기 누리온(NURION)은 리눅스 기반의 초병렬 클러스터 시스템으로 이론최고성능(Rpeak) 25.7PFlops의 연산처리 성능을 갖고 있다.

구분		KNL 계산노드	CPU-only 노드
제조사 및 모델		Cray CS500	
노드수		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 코어수	68	20
	노드 당 CPU수	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
	노드 당 메모리(GB)	96GB	192GB
	대역폭	115.2GB/s	128GB/s
	전체크기	778.6TB	24.8TB
고성능 인터커넥트	토폴로지	Fat Tree	
	Blocking Ratio	50% Blocking	
	Switches 구성	278x 48-port OPA edge switches 8x 768-port OPA core switches	
	포트 당 대역폭	100Gbps	
고성능 파일시스템 (Burst Buffer)	서버수	DDN IME240 Servers 48ea	
	서버 당 디스크	16x 1.2TB NVMe SSD, 2x 0.45TB NVMe SSD	
	전체용량	0.92PB usable	
	대역폭	20GB/sec per server, 0.8TB/s total	
병렬파일시스템	파일시스템	Lustre 2.7.21.3	
	전체 용량	/scratch: 21PB, /home: 0.76PB, /apps: 0.5PB	
	대역폭	0.3TB/s	
	RAID 구성	RAID6(8D+2P)	

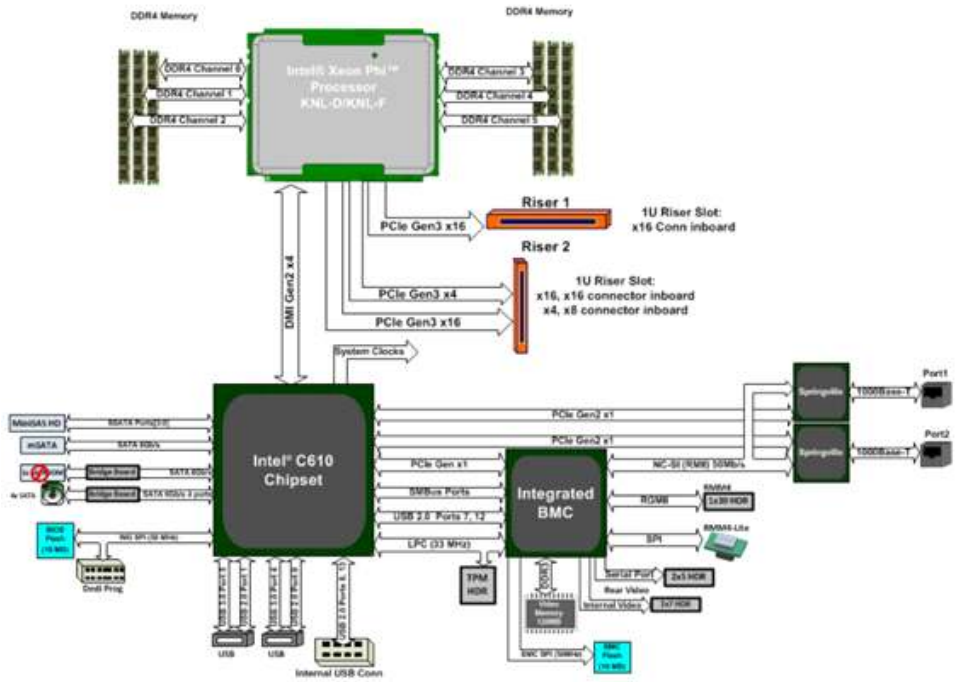
[누리온 시스템 사양 및 구성]

가. 계산 노드

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

○ KNL(매니코어) 노드

KNL 노드에 장착된 Intel Xeon Phi 7250 프로세서(코드명 Knights Landing)는 1.4GHz 기본 주파수에 68개 코어(쓰레드 272개)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.

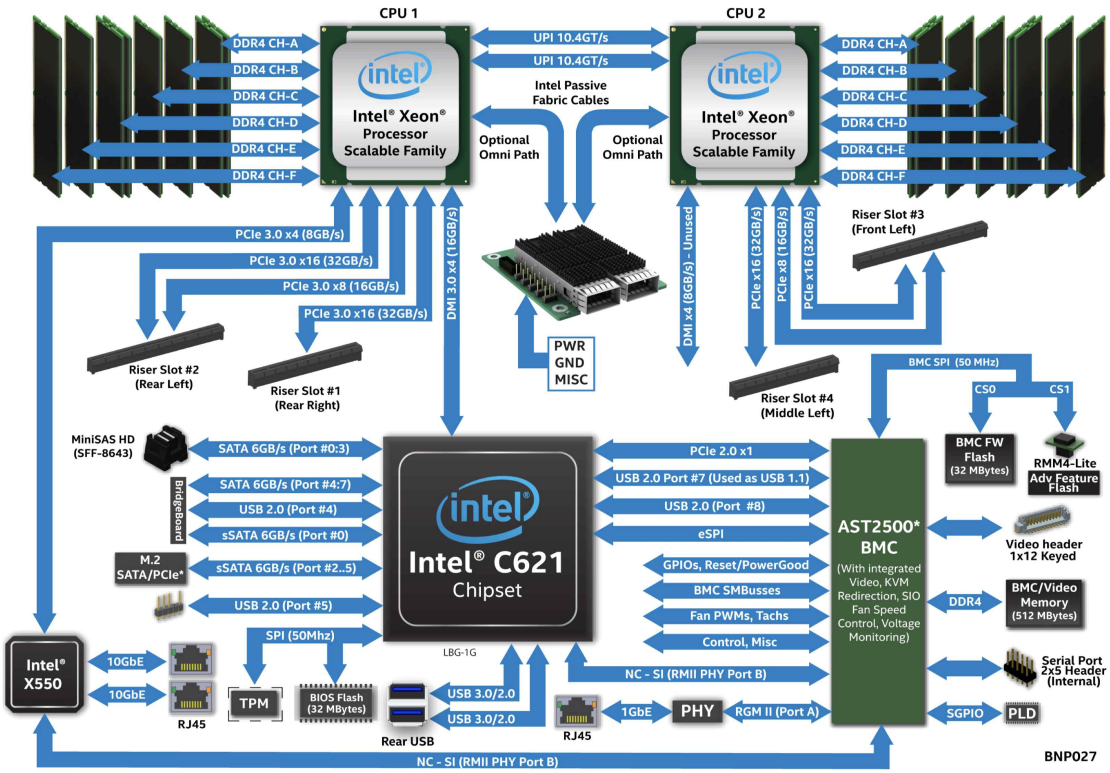


[KNL 기반 계산노드 블록 다이어그램]

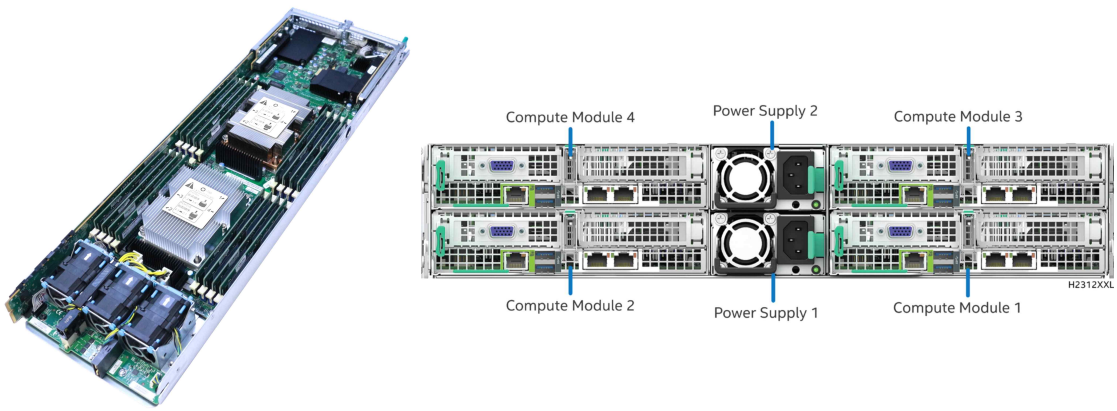
[KNL 기반 계산노드]

○ SKL(CPU-only) 노드

SKL(CPU-only) 노드에는 2개의 Intel Xeon Gold 6148 프로세서(코드명 Skylake)가 장착되어 있다. 기본 주파수는 2.4GHz이며 20개 코어(쓰레드 40개)로 동작한다. L3 캐시 메모리는 27.5MB이며, 각 CPU당 메모리는 96GB(노드 당 192GB)로 16GB DDR4-2666 메모리가 6채널로 구성되어 있다. 2U크기의 인클로저(enclosure)에는 4개의 계산노드가 장착되어 있다.



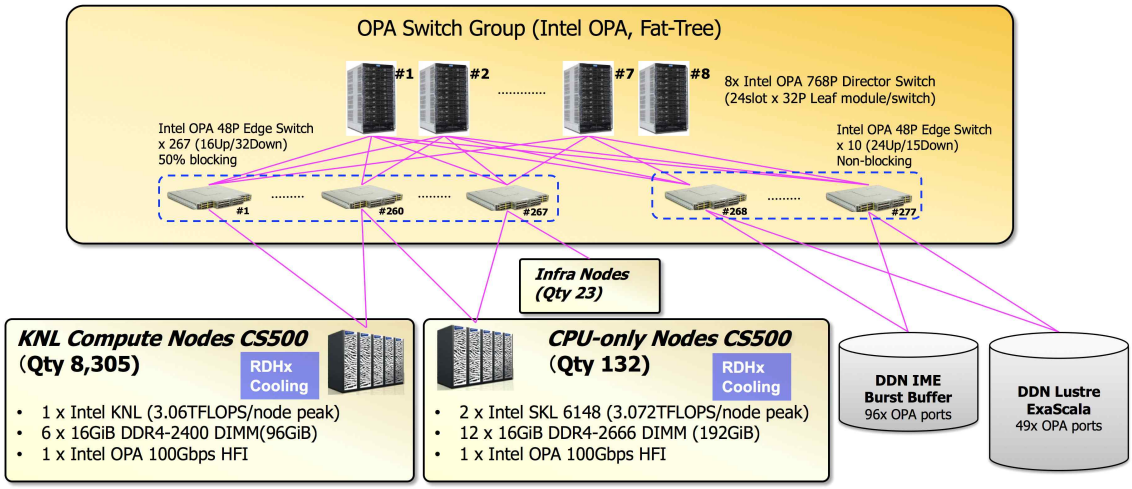
[SKL 기반 CPU-only 노드 블록 다이어그램]

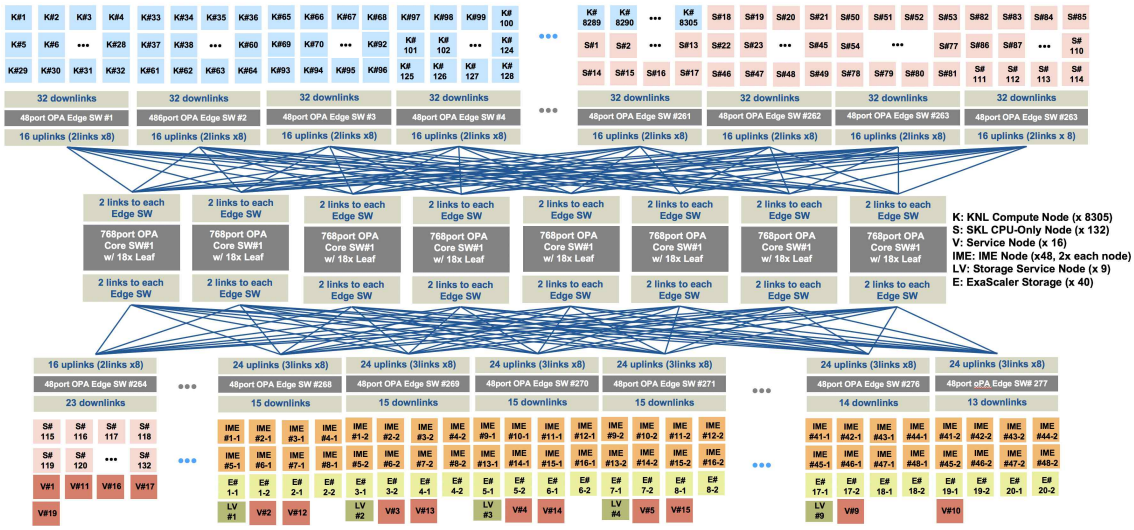


[SKL 기반 CPU-only 노드]

나. 인터커넥트 네트워크

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





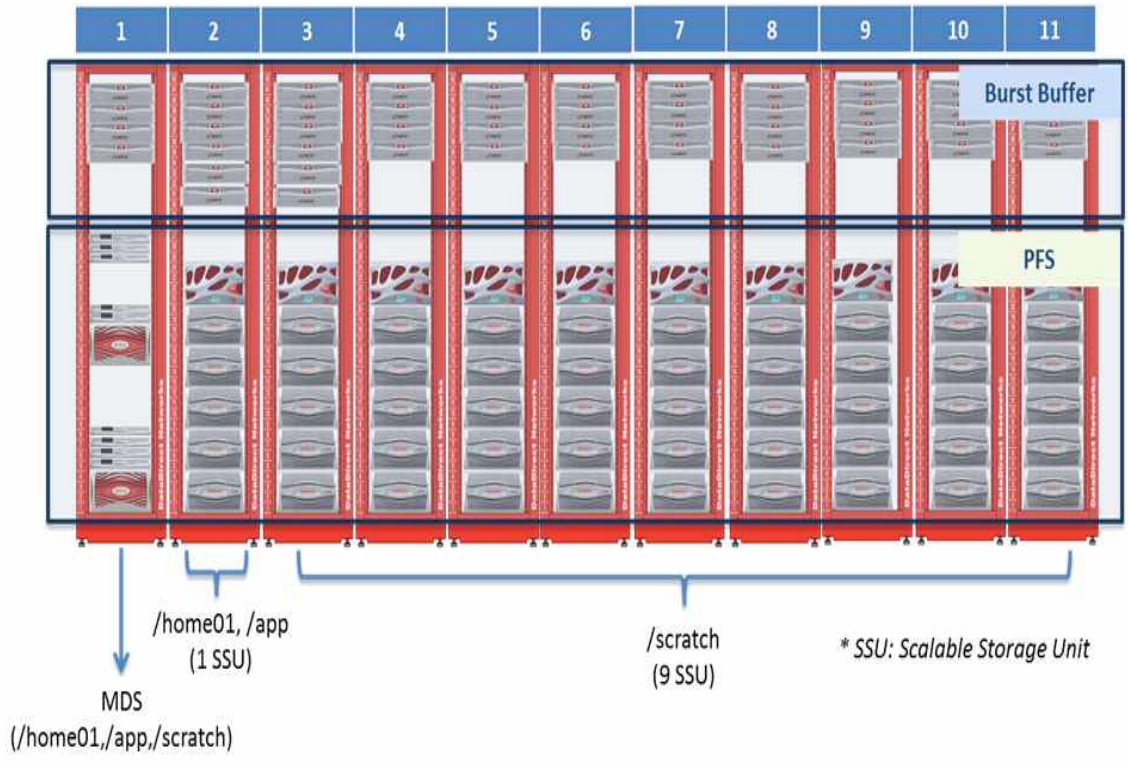
[인터커넥트 네트워크 구성도]

다. 스토리지

○ 병렬파일시스템 및 버스트버퍼 구성

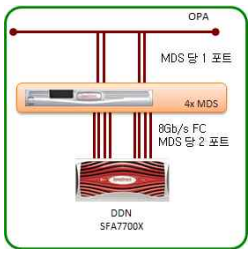
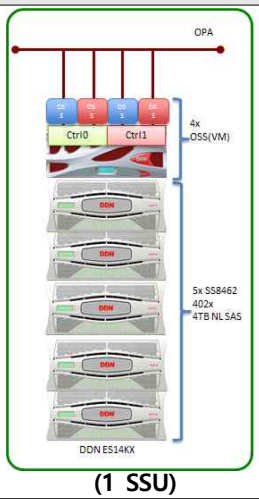
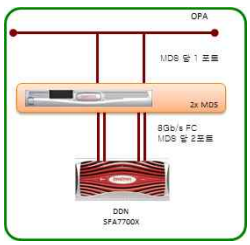
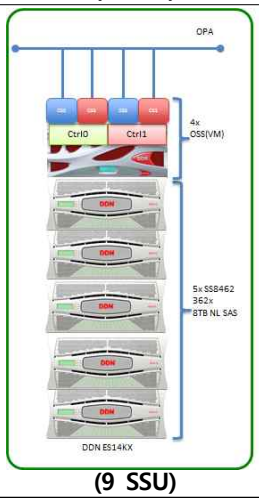
Nurion의 병렬파일시스템(Parallel File System, 이하 PFS)과 버스트버퍼(Burst Buffer, 이하 BB)는 DDN(Data Direct Network)사 IME(Infinite Memory Engine)와 Lustre 파일시스템 기반의 EXAScaler 솔루션을 사용하여 서비스된다. PFS는 DDN ES14KX 모델로 구성된 스크래치(/scratch), 홈(/home01), 어플리케이션(/app) 세 개의 파일시스템을 제공하고 있으며, 각 파일시스템은 SFA7700X 스토리지를 사용하는 두 대의 메타데이터 서버(MDS)를 포함한다.

BB는 계산 노드와 병렬파일시스템 사이에서 동작하는 NVMe 기반의 캐시로 약 829TB의 용량을 제공한다. PFS는 765TB 용량의 홈 디렉터리(/home01)와 510TB 용량의 사용자 어플리케이션이 제공 디렉터리(/applic), 21PB 글로벌 스크래치 디렉터리(/scratch)를 제공하고 있다. 해당 파일시스템은 계산노드, 로그인노드, 아카이빙 서버(Data Mover)에 마운트 되어 서비스 되고 있다.



[병렬파일시스템 및 버스트버퍼 전체 랙 구성]

다음 그림은 스크래치(/scratch), 홈(/home01), 어플리케이션(/app) 파일시스템의 상세 구성을 나타낸다.

	MDS	OSS
/home01 /apps		 **(1 SSU)**
/scratch		 **(9 SSU)**

파일 striping([별첨3] 참조)은 Lustre의 대표적인 특징이라 할 수 있으며 OST(Object Storage Target), 즉 물리적으로 분산되어있는 디스크에 하나의 파일을 분산시켜 저장함으로써 병목현상을 제거하고 I/O 성능을 향상시킬 수 있다. Nurion 파일 시스템의 striping 설정은 아래와 같다.

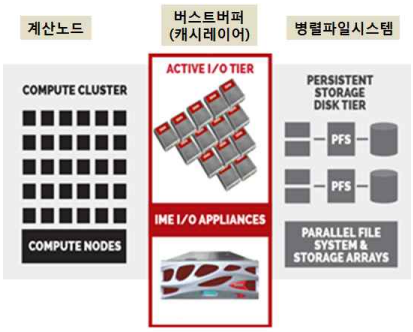
	stripe-count	stripe-size
/home01	1	1M
/apps	1	1M
/scratch	4	1M

[Nurion 파일시스템 striping 설정]

○ **버스트버퍼**

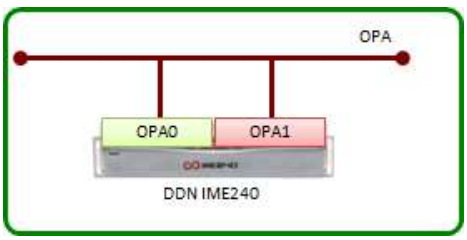
- 개요

버스트버퍼(Burst Buffer)는 계산노드와 스토리지 (/scratch) 사이의 I/O 가속화를 위한 캐시 레이어로 병렬파일시스템의 small I/O 또는 random I/O의 취약성을 보완하고자 5호기 Nurion에 최초 도입되었다. I/O 의존성이 높은 사용자 어플리케이션을 지원을 목표로 하고 있으며, 응용수행과제가 버스트버퍼 사용에 적절한지 사전 평가 후 전용큐(큐명, bb_cache, bb_falt)를 통해 지원하고 있다.



- 시스템 구성

버스트버퍼 구성을 위해 DDN사의 IME240 솔루션이 적용되었으며, IME는 계산 노드와 병렬파일시스템 사이에서 동작하는 NVMe 기반의 캐시를 제공한다. 아래 그림은 버스트버퍼의 상세 구성을 나타낸다.



시스템	DDN IME240, Infinite Memory Engine Appliance
소프트웨어 버전	CentOS 7.3, IME 1.2
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]

총 시스템 수	IME240 x 48대
총 용량(Raw)	921.6TB
총 용량(패리티 적용 시)	829.4TB (EC,9+1)
최대 IO 성능	0.8TB/s

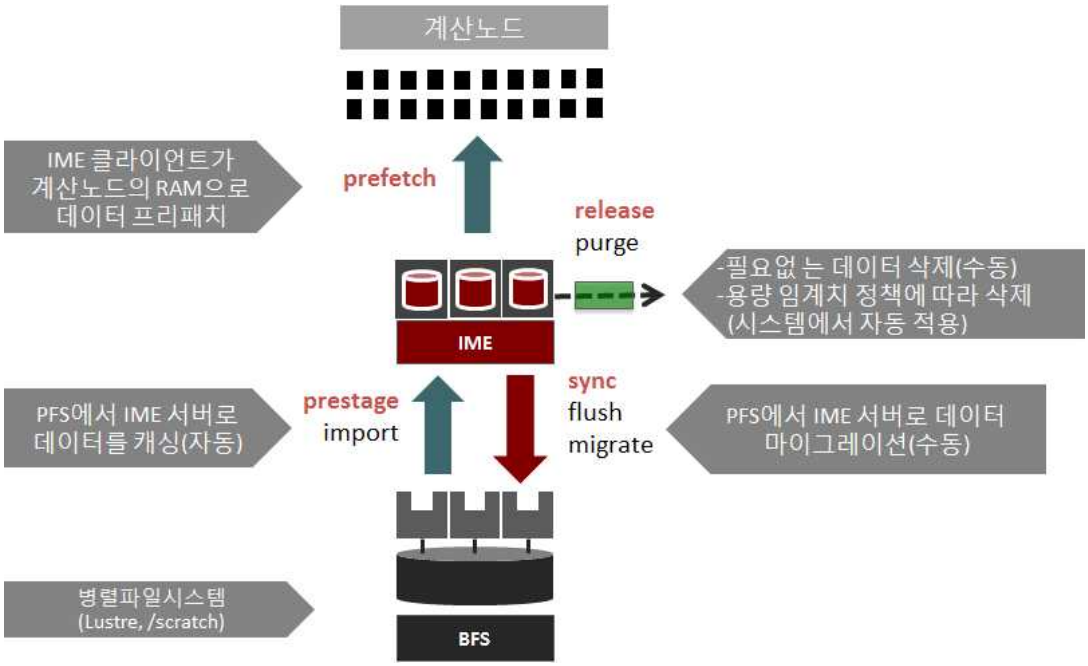
** EC: Erasure Coding (설정은 변경될 수 있음)*

[버스트버퍼 전체 구성]

- 사용방법

IME는 FUSE(**F**ile System In **USE**rspace)를 사용하여 IME-API를 제공하고 있다. 따라서 IME 클라이언트 서버는 FUSE를 통해 IME 디바이스를 마운트 하여 사용하게 된다. 주의할 것은 IME는 /scratch 파일시스템의 캐시 역할을 수행하기 때문에 /scratch 파일시스템이 사전에 반드시 마운트 되어 있어야 한다. FUSE를 사용하여 마운트 된 IME 디렉터리는 /scratch_ime이며 최초 사용자가 해당 디렉터리(/scratch_ime/$USER) 접속하면 스크래치(/scratch/$USER) 파일시스템의 모든 디렉터리와 파일들을 동일하게 확인할 수 있다. 이는 실제 IME 디바이스에 존재 않은 데이터이며 버스트버퍼를 이용하여 작업 수행 시 스크래치 파일시스템에서 IME로 캐싱하여 수행하게 된다.

IME를 사용하게 위해서는 우선 아래 그림에 있는 데이터의 라이프 사이클을 파악해야 한다. IME이 데이터 처리 단계는 크게 prestage, prefetch, sync, releas 단계가 있으며 각각에 대해 IME-API를 제공하고 있다.



[데이터 라이프 사이클]

ime-ctl -i $INPUT_FILE	작업 데이터에 대해 Stage-In 실행 (/scratch에서 /scratch_ime로 데이터 캐싱)
ime-ctl -r $OUTPUT_FILE	버스트버퍼의 캐시 데이터를 병렬파일시스템과 동기화 (/scratch_ime 캐싱 데이터를 /scratch로 전송)
ime-ctl -p $TMP_FILE	버스트버퍼 캐시에 데이터 삭제 (/scratch_ime 퍼지)
ime-ctl -s $FILE	버스트버퍼에 캐싱된 데이터의 상태정보 제공

* ime-ctl --help를 통해 상세 옵션 확인 가능

- 작업제출을 통한 버스트버퍼 사용 예제

```
#!/bin/bash -l
#PBS -l select=1
#PBS -q bb_cache (or bb_
INPUT=/scratch/$USER/input.dat
FILE=/scratch/$USER/output.dat
echo "[ `date` ] PFS: start"
dd if=$INPUT of=$FILE bs=1M count=$(( 50 * 1024 ))
echo "[ `date` ] PFS: end"
INPUT=/scratch_ime/$USER/input.dat
FILE=/scratch_ime/$USER/output.dat
# stage in
/opt/ddn/ime/bin/ime-ctl -i $INPUT
/opt/ddn/ime/bin/ime-ctl -s $INPUT ①
echo "[ `date` ] IME: start"
dd if=$INPUT of=$FILE bs=1M count=$(( 50 * 1024 )) ②
echo "[ `date` ] IME: end"
/opt/ddn/ime/bin/ime-ctl -s $FILE ③
# release
/opt/ddn/ime/bin/ime-ctl -r $FILE
/opt/ddn/ime/bin/ime-ctl -s $FILE ④
# purge
/opt/ddn/ime/bin/ime-ctl -p $FILE ⑤
```

① ※ 출력값
File: /scratch_ime/$USER/input.dat'
Number of bytes:
Dirty: 0
Clean: 51200 (BB로 파일 stage-in 됨)
Syncing: 0

② ※ BB의 output.dat 파일이 변경

③ ※ 출력값
File: `output.dat'
Number of bytes:
Dirty: 53687091200 (PFS-BB의 데이터가 다름)
Clean: 0
Syncing: 0

④ ※ 출력값
File: `output.dat'
Number of bytes:
Dirty: 0
Clean: 4085252096 (BB 데이터 PFS로 release)
Syncing: 49601839104

⑤ ※ BB 데이터 캐쉬를 삭제 (purge)

* 주의) BB의 캐시 데이터가 PFS로 release(#ime-ctl -r) 완료되기 전 purge(#ime-ctl -p) 명령이 수행되면 해당 데이터가 손상됨.

- 주의사항

버스트버퍼는 초기 작업 수행 시 사용자 데이터를 버스트버퍼의 디바이스로 캐싱하는 단계와 주기적으로 캐시 데이터를 PFS(병렬파일시스템)로 flushing(sync) 해주는 작업의 부하가 발생한다. 그렇기 때문에 병렬파일시스템(Lustre)에서 취약한 대량의 small I/O가 많거나 checkpoint가 잦은 어플리케이션에 대해서 유리하다.

또한 버스트버퍼는 캐시의 용도로 사용되기 때문에 상대적으로 PFS(약 20PB)에 비해 BB(약 0.92PB)의 용량이 작다. 따라서 여러 사용자들이 사용 중에 버스트버퍼의 용량이 차게 되면 임계치 설정에 따라 데이터가 캐시에서 삭제 될 수 있으니 응용 수행에 대한 데이터 관리가 매우 중요하다.

2. 사용자 환경

가. 계정발급

- 누리온 시스템의 베타서비스 사용을 승인받은 연구자는 헬프데스크(https://helpdesk.ksc.re.kr) 웹서비스를 통해 계정을 신청한다.
  - 신청 방법 : 헬프데스크 웹사이트접속, (상단) 계정신청 -> (좌측) 계정신청 -> (하단) 누리온 시스템 베타서비스 메뉴를 통해 계정 신청 후 발급 받는다.

- OTP(One Time Password, 일회용 비밀번호) 설정
  - 슈퍼컴퓨터 접속 시에 OTP에서 제공하는 보안숫자를 입력하여야 하며, KISTI 슈퍼컴퓨팅 서비스센터에서는 OTP를 위한 스마트폰 앱을 제공한다.
  - OTP 스마트폰 앱의 다운로드 주소는 150.183.146.81 이며, OTP 앱 설정할 때 필요한 인증 코드는 계정 담당자(account@ksc.re.kr)에게 아래의 내용을 작성하여 메일로 요청한다.

메일 제목	OTP 요청 - 사용자 ID (예) OTP 요청 - x123abc
수신인	account@ksc.re.kr
메일내용(예제)	사용자 ID: x123abc 휴대폰번호: 010-1234-5678 이름: 홍길동

※ 스마트폰을 사용하고 있지 않은 사용자의 경우, 계정담당자(account@ksc.re.kr)에게 문의

나. 로그인

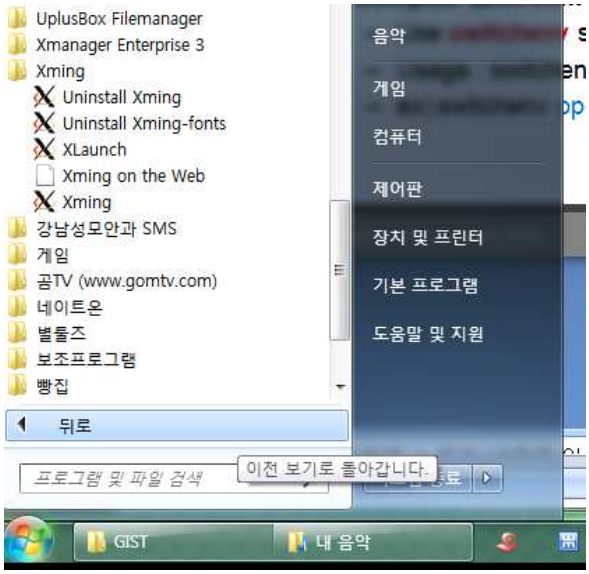
- 사용자는 누리온 시스템 로그인 노드(nurion.ksc.re.kr)를 통해서 접근이 가능하다(하단, 노드 구성 참조).
- 기본 문자셋(encoding)은 유니코드 (UTF-8)이다.
- 로그인 노드에 대한 접근은 ssh, scp, sftp, X11 만 허용된다.

① 유닉스 또는 리눅스 환경

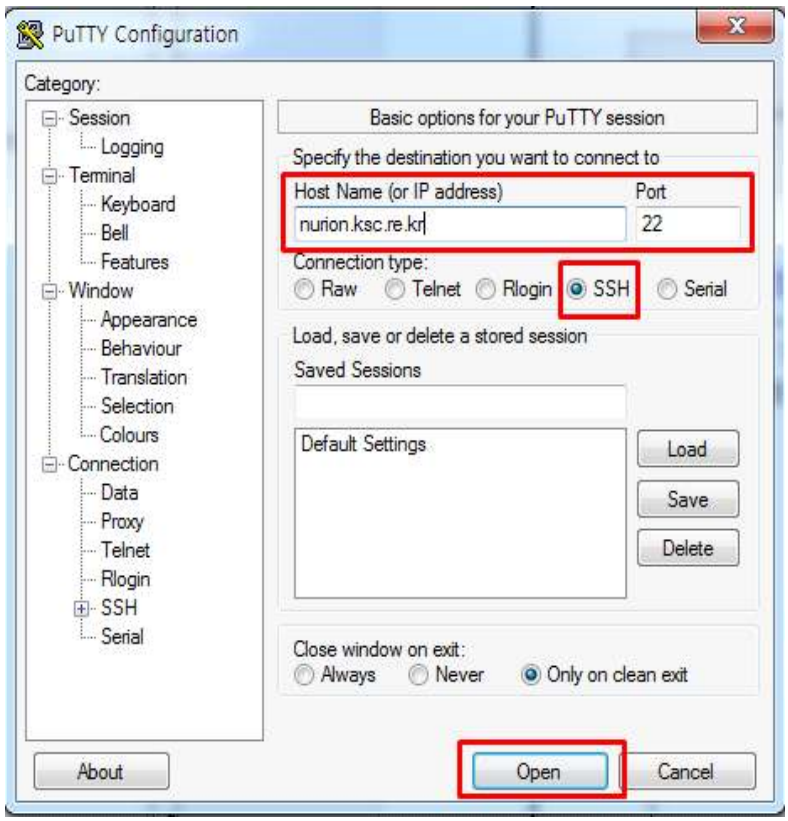
```
$ ssh -l <사용자ID> nurion.ksc.re.kr [-p 22]
```

② 윈도우 환경

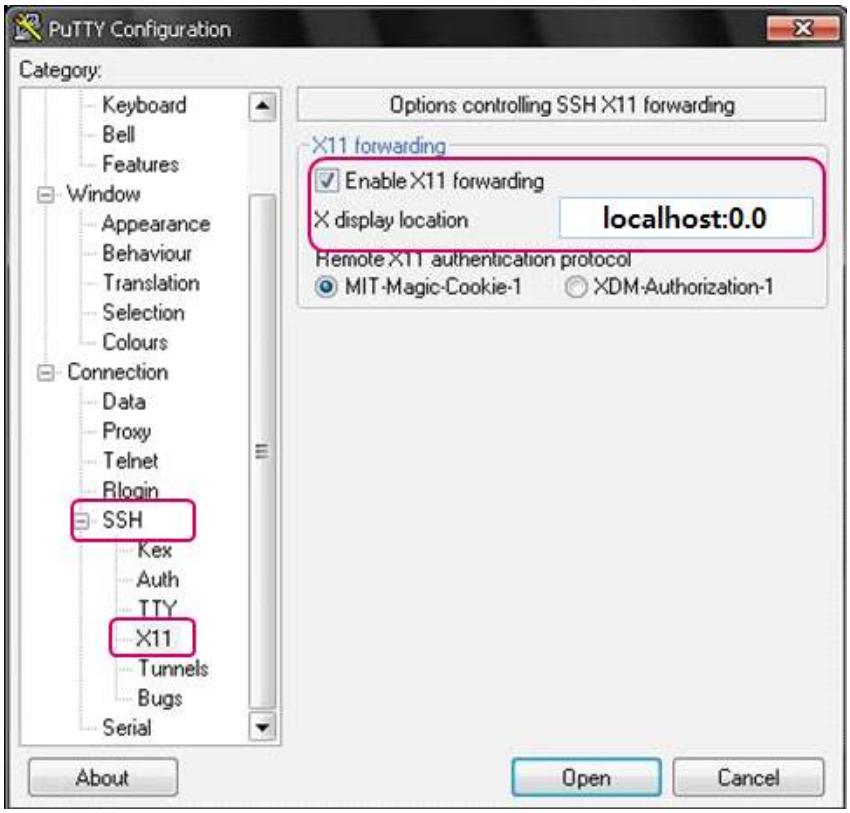
- X환경 실행을 위해 Xming 실행
  ※ 프로그램은 인터넷을 통해 무료로 다운로드 후 설치



- putty나 SSH Secure Shell Client 등의 ssh 접속 프로그램을 이용
  ※ 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

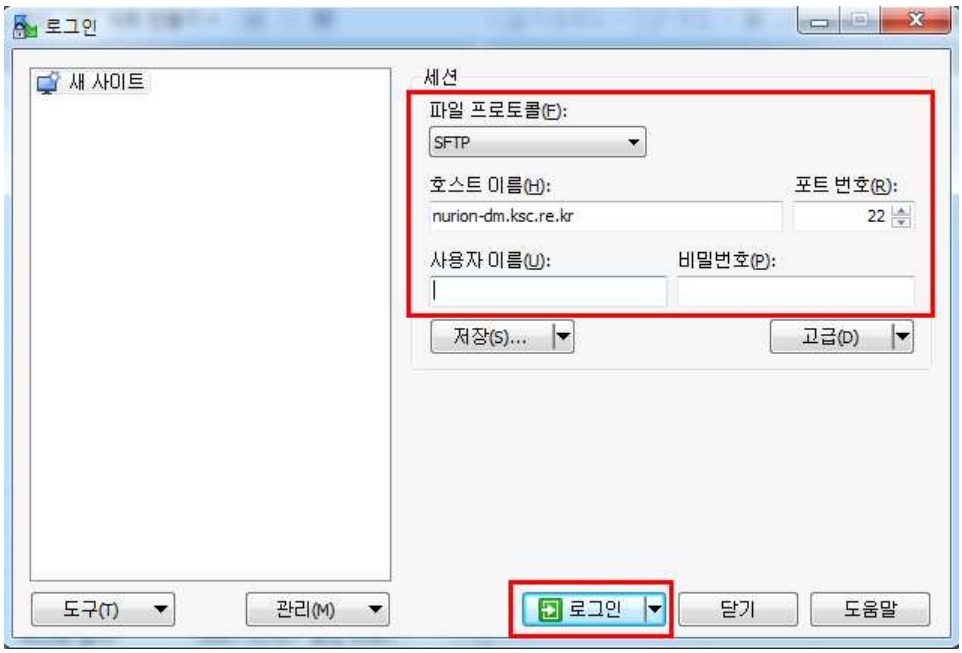
```
C:₩ ipconfig /flushdns
```

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 SFTP 클라이언트 프로그램을 이용하여 접속한다.



④ 노드구성

		호스트명	CPU Limit	비고
로그인 노드		nurion.ksc.re.kr	20분	• ssh/scp/sftp 접속 가능 • 컴파일 및 batch 작업제출용 • ftp 접속 불가
Datamover 노드		nurion-dm.ksc.re.kr	-	• ssh/scp/sftp 접속 가능 • ftp 접속 가능 • 컴파일 및 작업제출 불가
계산노드	KNL	node[0001-8305]	-	• PBS 스케줄러를 통해 작업 실행 가능 • 일반사용자 직접 접근 불가
	CPU-only	cpu[0001-0132]	-	

다. 사용자 쉘 변경

- 누리온 시스템의 로그인 노드는 기본 쉘로 **bash**이 제공된다. 다른 쉘로 변경하고자 할 경우 chsh 명령어를 사용한다.

```
$ chsh
```

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

```
$ echo $SHELL
```

- 쉘의 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

o 사용자 패스워드 길이는 최소 9자이며, 영문, 숫자, 특수문자의 조합으로 이뤄져야 한다. 영문 사전 단어는 사용이 불가하다.
o 사용자 패스워드 변경 기간은 2개월로 설정(60일) 된다.
o 새로운 패스워드는 최근 5개의 패스워드와 유사한 것을 사용할 수 없다.
o 최대 로그인 실패 허용 횟수 : 5회
  - 5회 이상 틀릴 경우, 이 계정의 ID는 lock이 걸리므로, 계정담당자(account@ksc.re.kr)에게 문의해야한다.
  - 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.
o OTP 인증오류 허용 횟수 : 5회
  - 5회 이상 틀릴 경우, 계정담당자(account@ksc.re.kr)에게 문의해야 한다.

마. 작업 디렉터리 및 쿼터 정책

○ 홈 디렉터리 및 스크래치 디렉터리에 대한 정보는 아래와 같다.

구분	디렉터리 경로	용량제한	파일 수 제한	파일 삭제 정책	파일시스템	백업유무
홈 디렉터리	/home01	64GB	10K	N/A	Lustre	X
스크래치 디렉터리	/scratch	100TB	1M	15일 동안 접근하지 않은 파일은 자동 삭제		X

- 홈 디렉터리는 용량 및 I/O 성능이 제한되어 있기 때문에, 모든 계산 작업은 스크래치 디렉터리인 /scratch의 사용자 작업 공간에서 이루어져야 한다.
- 사용자 작업 디렉터리는 용도에 따라 다른 정책을 적용받는다.
- **베타서비스 기간 동안은 백업을 지원하지 않는다**.

3. 사용자 프로그래밍 환경

가. 프로그래밍 도구 설치 현황

○ 컴파일러 및 라이브러리 모듈

구분	항목	
아키텍처 구분 모듈	· craype-mic-knl · craype-network-opa	· craype-x86-skylake
Cray 모듈	· cdt/17.10 · cray-ccdb/3.0.3 · cray-cti/1.0.6 · cray-fftw/3.3.6.2 · cray-fftw_impi/3.3.6.2 · cray-impi/1.1.4 · cray-lgdb/3.0.7 · cray-libsci/17.09.1 · craype/2.5.13	· craypkg-gen/1.3.5 · msr-tools/1.3 · mvapich2_cce/ 2.2rc1.0.3_noslurm · papi/5.5.1.3 · perftools/6.5.2 · perftools-bas/6.5.2 · perftools-lite/6.5.2 · PrgEnv-cray/1.0.2
컴파일러	· intel/17.0.5 · intel/18.0.1	· intel/18.0.3
컴파일러 의존 라이브러리	· hdf4/4.2.13 · hdf5/1.10.2 · lapack/3.7.0	· ncl/6.5.0 · ncview/2.1.7 · netcdf/4.6.1
MPI 라이브러리	· impi/17.0.5 · mvapich2/2.3	· openmpi/3.1.0
MPI 의존 라이브러러	· fftw_mpi/2.1.5 · hdf5-parallel/1.10.2 · parallel-netcdf/1.10.0	· fftw_mpi/3.3.7 · netcdf-hdf5-parallel/4.6.1
Intel 패키지	· advisor/18.2.0 · vtune/18.2.0	· advisor/18.3.0 · vtune/18.3.0
응용소프트웨어	· cmake/3.12.3 · forge/18.1.2 · grads-2.2.0 · gromacs/2016.4 · lammps/8Mar18 · namd/2.12	· python/2.7.15 · python/3.7 · qe/6.1 · R/3.5.0 · siesta/4.0.2 · siesta/4.1-b3
상용소프트웨어	· cfx/v145 · cfx/v170 · cfx/v181 · cfx/v191 · gaussian/g16.a03	· fluent/v145 · fluent/v170 · fluent/v181 · fluent/v191

○ 상용소프트웨어 정보

분야	소프트웨어	버전	라이센스	디렉터리 위치
열유체역학	ANSYS CFX	V145 V170 V181 V191	4개 (HPC 128)	/apps/commercial/ANSYS
	ANSYS Fluent			/apps/commercial/ANSYS
화학/생명	Gaussian	g16-a03	작업 수 제한없음 단일노드 내 CPU 수 제한없음	/apps/commercial/G16/g16

※ Gaussian은 helpdesk 계정담당자(account@ksc.re.kr)를 통해 사용권한 신청 후 사용 가능함

나. 프로그램 컴파일

5호기 누리온 시스템 베타서비스는 Intel 컴파일러를 지원한다. 또한, Intel 컴파일러를 사용하여 사전에 컴파일한 MPI 라이브러리를 사용자에게 제공한다.

병렬 프로그래밍 환경 지원을 위해, 본 시스템은 IMPI, MVAPICH2, OpenMPI를 지원한다.

1) 컴파일러 및 MPI 환경 설정 (modules)

5호기 누리온 시스템은 module 명령어를 이용하여 쉽게 환경변수를 설정한 후 소프트웨어 사용이 가능하다. 주로 쓰이는 module 명령어들은 다음과 같다.

○ **기본 필요 모듈**

※ 모듈 craype-mic-knl 이나 craype-x86-skylake 둘 중 사용할 시스템에 해당하는 모듈 하나를 **반드시** load 해야 함

```
$ module load craype-mic-knl
```

혹은,

```
$ module load craype-x86-skylake
```

○ 사용가능한 모듈 목록 출력

```
$ module avail|av
```

○ 모듈 적재

```
$ module add|load [module name]
```

○ 적재된 모듈 삭제

```
$ module rm|unload [module name]
```

○ 적재된 모듈 목록 출력

```
$ module list
```

○ 적재된 모든 모듈 삭제

```
$ module purge
```

※ 적재된 모든 모듈 삭제 시 기본 적재 모듈도 삭제

※ 자주 사용하는 환경은 .bashrc 등 startup 스크립트에 적어주면 로그인 할 때마다 기본적으로 적용된다.

2) 순차 프로그램 컴파일

순차 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 컴파일 할 때도 그대로 사용되므로, 순차 프로그램에 관심이 없다 하더라도 참조하는 것이 좋다.

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

■ **Intel 컴파일러 주요 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 -xMIC-AVX512	512bits 레지스터를 가진 CPU를 지원 512bits 레지스터를 가진 MIC를 지원
-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<processor> -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 추가함.
-msse<3,4.1>,-mssse3	Streaming SIMD Extensions 3 지원
-fPIC,-fpic	PIC (Position Independent Code)가 되도록 컴파일
-p	프로파일링 정보를 생성(gmon.out)
-unroll<n>	unroll 활성화. <n> 은 최대 횟수 (64비트만 지원)
-mcmodel medium	2GB이상의 memory allocation이 필요한 경우 사용
-help	옵션 목록 출력

※ 권장 옵션 : -O3 -fPIC -xCORE-AVX512 (Skylake)

※ 권장 옵션 : -O3 -fPIC -xMIC-AVX512 (KnightsLanding)

※ 권장 옵션 : -O3 -fPIC -xCOMMON-AVX512 (Skylake & KnightsLanding)

■ **Intel 컴파일러 사용 예제**

```
$ icc/ifort -o test.exe -O3 -xCORE-AVX512 test.c/cc/f90
```

3) 병렬 프로그램 컴파일

○ **OpenMP 컴파일**

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

컴파일러	프로그램	옵션
icc / icpc	C / C++	-qopenmp
ifort	F77/F90	-qopenmp

- Intel 컴파일러 사용시 OpenMP 컴파일 예제

```
$ icc/ifort -o test.exe -qopenmp -O2 -xCORE-AVX512 test.c/cc/f90
```

○ **MPI 컴파일**

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

컴파일러	프로그램	소스 확장자
mpicc	C	.c
mpicxx/mpiCC	C++	.cc, .c, .cpp, .cxx
mpif90	F77/F90	.f, .for, .ftn, .f90, .f95, .fpp

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

- Intel 컴파일러 사용 시 MPI 컴파일 예제

```
$ mpicc/mpif90 -o test.exe -O2 -xCORE-AVX512 test.c/cc/f90
```

- Intel 컴파일러 사용 시 impi 컴파일 예제

```
$ mpiicc/mpiifort -o test.exe -O2 -xCORE-AVX512 test.c/cc/f90
```

■ **디버거 DDT** 사용예제

5호기 시스템에서 DDT를 사용하기 위하여는 로그인노드에서 인터랙티브 작업제출을 하여 노드로 접속해야 한다.

노드로 접속한 후 사용할 아키텍처, 컴파일러, MPI를 선택하고 DDT 를 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.

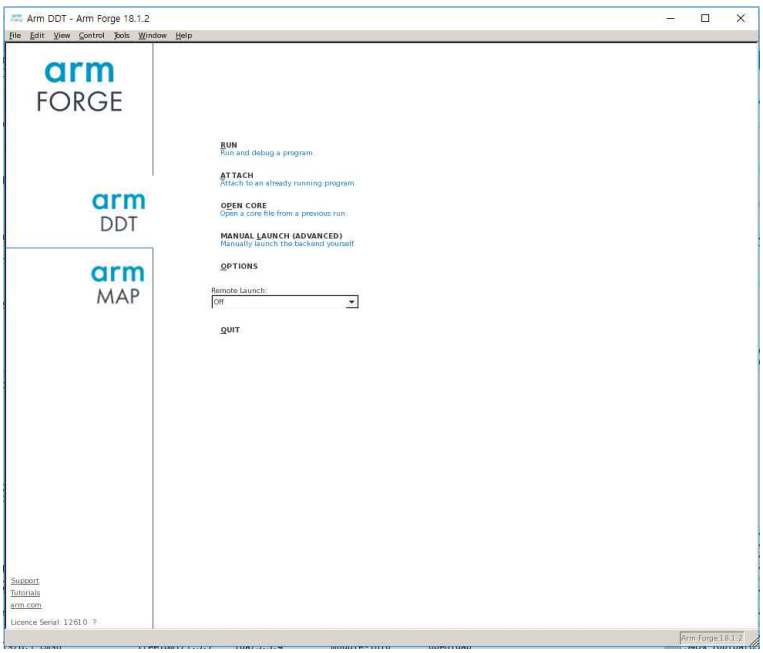
DDT를 사용하기 전 사전 준비로 컴파일 시 -g -O0 옵션을 넣어 실행파일을 생성한다.

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

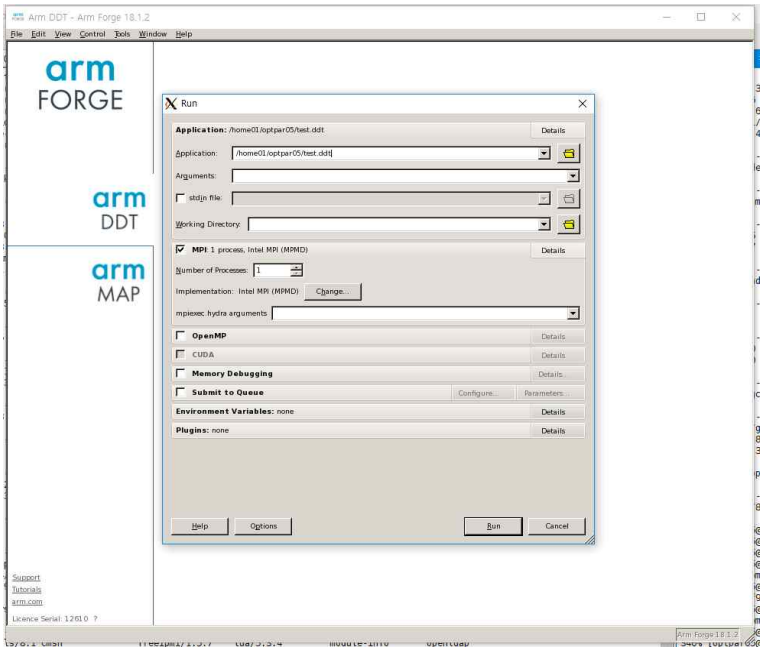
사용자의 데스크탑에서 xming 실행 및 ssh X환경 설정 완료한 후 ddt 실행 명령을 실행한다.

```
$ ddt &
```

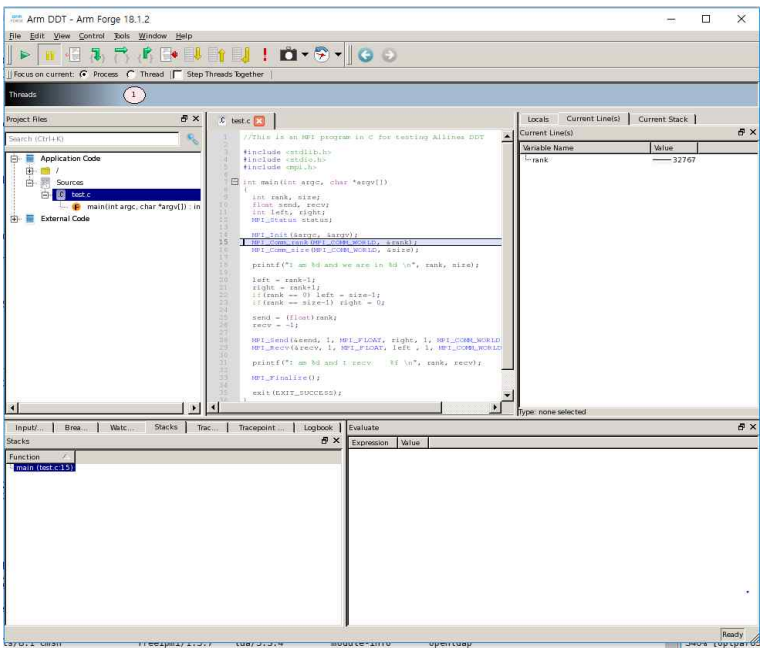
명령을 실행하여 아래와 같은 팝업 실행 창을 확인한다.



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



아래와 같이 선택 된 실행 파일에 대한 디버깅 모드로 진입하여 디버깅을 진행할 수 있다.



■ **프로파일러 Intel vtune** Amplifier 2018.3.0 사용예제

본 시스템에서 프로파일러인 vtune을 사용하기 위하여 인터랙티브 작업제출하여 노드로 진입 후 사용하여야 한다.

노드에서 아키텍처, 컴파일러, MPI 를 선택한 후 vtune을 선택하면 프로파일러를 사용 할 수 있다.

$ module load craype-mic-knl or craype-x86-skylake

$ module load intel/18.0.3 impi/18.0.3
$ module load vtune/18.3.0
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
```

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

```
$ amplxe-cl -report hotspots /home01/$USER/test/r000s
amplxe: Using result path `/home01/$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
matrix.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
```

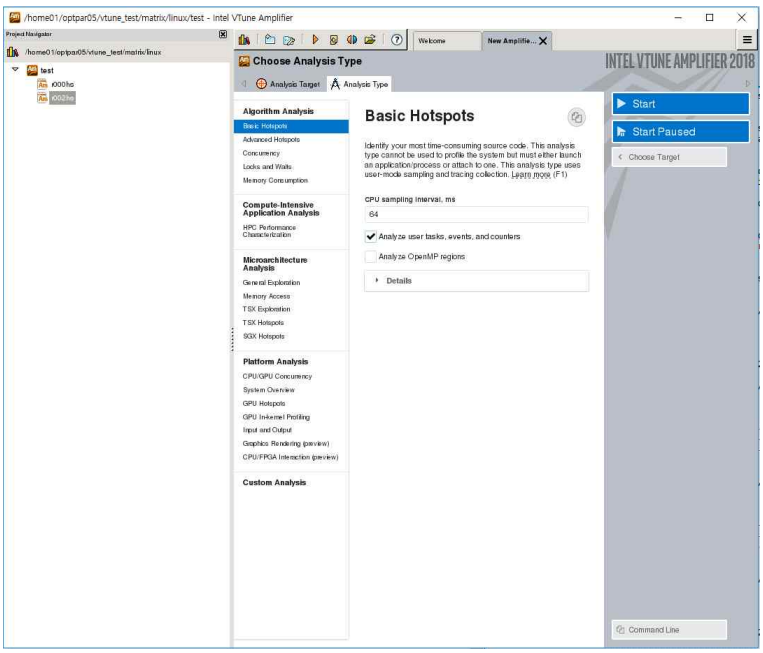
○ **GUI 사용 결과 확인 방법**

Intel vtune Amplifier 는 GUI 모드 역시 지원한다. 여기서는 GUI를 이용한 결과 확인 방법만 설명한다.

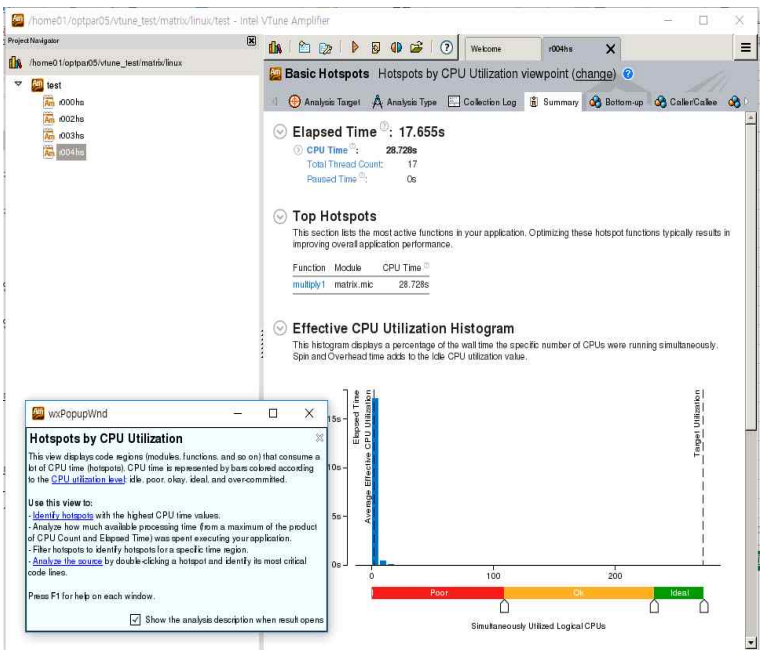
사용자의 데스크탑에서 xming 실행

```
$ amplxe-gui
```

아래의 화면에서 New Analysis 버튼을 클릭하여 아래와 같은 그림에서 cpu 수와 체크 확인 후 Start 버튼을 클릭하면 분석이 시작된다.



완료되면 아래와 같이 분석 결과가 요약되어 나타난다.



■ 프로파일러 Cray-Pat 사용예제

본 시스템에서 프로파일러인 CrayPat을 사용하기 위하여 인터랙티브 작업제출을 통해 계산노

드에 접속하여 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 컴파일 한다.

```
$ cc test.c
```

위 실행 결과로 a.out 이라는 실행파일이 생성된다.
CrayPat으로 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 MPI 로 작성된 파일이므로 mpirun 으로 실행한다.

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

실행을 완료하면 a.out+pat+114026-2s 디렉터리가 생성되고 xf-files/002812.xf 파일이 생긴다.

```
$ pat_report a.out+pat+114026-2s
```

위와 같이 pat_report를 실행하면 .ap2 파일과 .apa 파일이 생성된다.

```
$ pat_build -O a.out+pat+114026-2s/build-options.apa
```

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

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

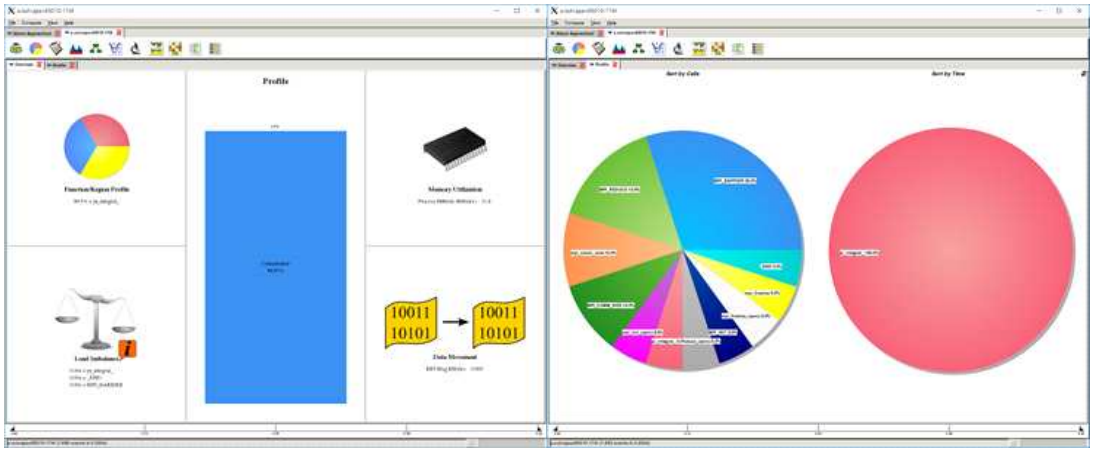
a.out+pat+114026-2s 에 새로운 xf 파일 생성된다.
pat_report를 다시 사용하여 새로운 데이터를 처리한다.

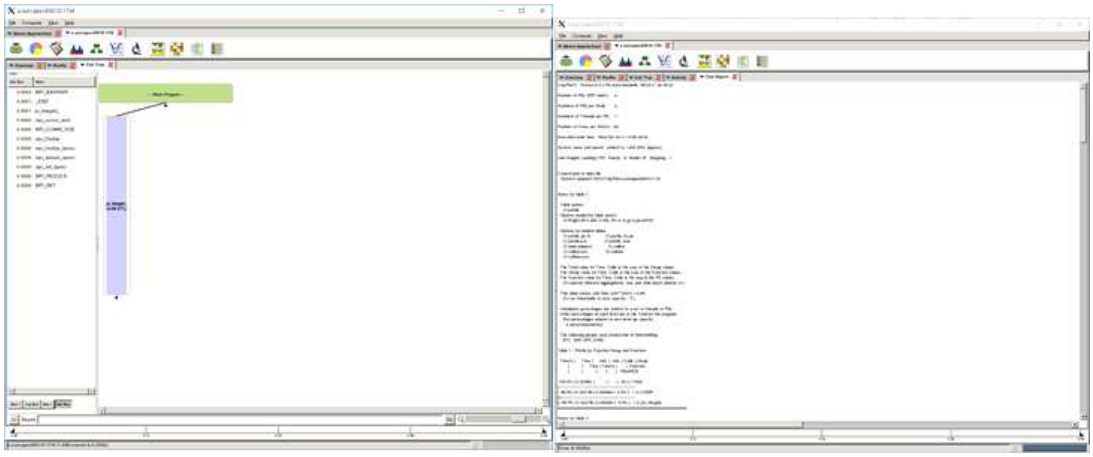
```
$ pat_report a.out+apa+114026-2s
```

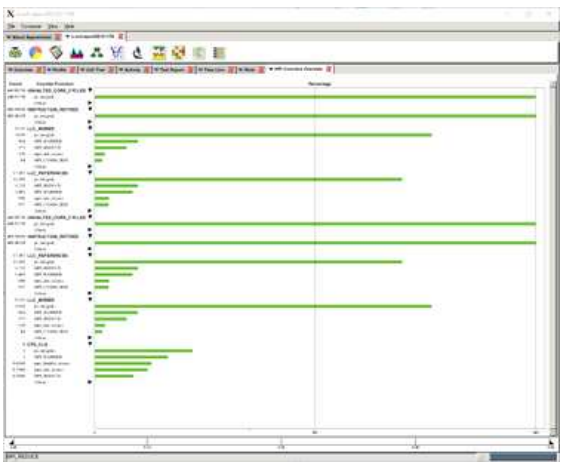
위와 같이 실행하여 ap2 파일과 tracing 리포트를 생성한다.
위아 같이 수집된 데이터를 시각화 하는 방법으로 app2를 제공한다.

```
$ app2 &  #
$ app2 a.out+apa+114026-2s
```

아래와 같이 시각화가 가능하다.


Profile
10011
10101
10011
10101





다. 싱귤레러티(singularity) 컨테이너 이미지 사용법

싱귤레러티는 OS 가상화를 구현하는 HPC 환경에 특화된 컨테이너 플랫폼이다. 기계학습을 위한 텐서플로우(Tensorflow) 등에서 요구하는 리눅스 배포판, 컴파일러, 라이브러리, 애플리케이션 등을 포함하는 컨테이너 이미지를 구동하여 사용자 프로그램을 실행할 수 있다. 향후 기계학습 및 빅데이터 분석 프레임워크 등을 위한 다양한 사용자 요구 기반의 컨테이너 이미지를 제공할 예정이다.

○ 싱귤레러티 모듈 적재

```
$ module load singularity/2.5.1
              or
$ $HOME/.bash_profile
  export PATH=$PATH:/opt/singularity/2.5.1/bin
```

○ 싱귤레러티 컨테이너에서 쉘 실행

```
$ singularity shell [이미지 이름]

$ singularity shell tensorflow-1.9.0-py3.simg
Singularity: Invoking an interactive shell within container...

Singularity tensorflow-1.9.0-py3.simg:tensorflow>
```

○ 싱귤레러티 컨테이너에서 사용자 프로그램 실행

```
$ singularity exec [이미지 이름] 실행명령어

$ singularity exec tensorflow-1.9.0-py3.simg python convolutional.py
```

※ 계산 노드에서 스케줄러(PBS)를 통한 컨테이너 실행은 p.23-24 예제 참조

소프트웨어 (프레임워크)	컨테이너 이미지 파일
tensorflow 1.9.0	/apps/applications/tensorflow/1.9.0/tensorflow-1.9.0-py3.simg

※ 제공되는 컨테이너 이미지는 사용자 작업 디렉터리로 복사하여 실행 권장

※ 파일럿 시스템에서는 사용자가 컨테이너 이미지를 직접 빌드하거나 수정할 수 없음

※ /applic/applications/tensorflow/1.9.0/examples 디렉터리에서 convolutional 모델 예제 프로그램(convolutional.py)과 데이터 디렉터리(data)를 사용자 작업 디렉터리로 복사하여 테스트할 수 있음

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, 큐 별로 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, 상용SW를 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.

가. 큐 구성

○ commercial 큐 (상용SW 수행을 위한 큐)에서만 공유 노드 정책이 적용되고, 나머지 큐에서는 배타적 노드 정책이 적용된다.

○ 작업 큐

- 일반사용자가 사용할 수 있는 큐는 다음과 같다. (2018년 10월 기준)

큐 구분	큐 속성	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
Compute	normal	**norm_cache**	4.5k	306k	48	40	20	4.5k	1
		norm_flat	500	34k		20	10	500	
	exclusive	**excl_cache**	2.1k	142,800	24	2	2	2.1k	512
	burst_buffer	**bb_cache**	700	47,600				300	1
		bb_flat	300	20,400					
	long	**long_cache**	180	12,240	120	20	10	180	1
		long_flat	20	1,360		2	2	20	
CPU_only		**commercial**	132	5,280	48	20	10	132	
		norm_skl	132	5,280					

① 작업 제출 개수 제한

- 사용자별 최대 제출 작업 개수 : 초과하여 작업을 제출한 경우 제출 시점에 오류가 발생함.

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

② 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.

③ KNL 메모리 모드에 따른 큐 구분(Cluster 모드는 전부 Quadrant)

- '_Cache'는 Cache 모드(MCDRAM을 L3캐시로 사용), '_Flat'는 Flat 모드(MCDRAM을 DDR4와 같이 RAM으로 사용)로 설정되어 있음.

④ 디버그 노드를 제공하지 않고, 인터렉티브 작업 제출로 디버깅 수행을 권장함. (인터렉티브 작업 제출법은 아래 나-2) 내용 참조)

⑤ Gaussian을 제외한 상용SW 사용자는 commercial 큐 사용을 원칙으로 함.

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

나. 작업 제출 및 모니터링

1) 배치 작업 제출

작업 유형별로 아래에 제시된 job script와 같이 사용자 작업에 대한 script 파일을 작성하여 qsub 명령을 사용하여 작업을 제출한다.

```
$ qsub <job attributes/resources> <job script>

ex)qsub -l select=1:ncpus=1 -q norm_cache hello_world.sh
```

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

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

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

```
#!/bin/sh
#PBS -N serial_job
#PBS -V
#PBS -q norm_cache
#PBS -l select=1
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

./a.out
```

※ 1노드 점유 순차, 사용 예제

■ OpenMP 프로그램 작업 스크립트 작성 예제(omp.sh)

```
#!/bin/sh
#PBS -N openmp_job
#PBS -V
#PBS -q norm_cache
#PBS -l select=1:ncpus=68:ompthreads=68
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

./a.out
```

※ 1노드 점유, 노드 당 68스레드(총 68 OpenMP 스레드) 사용 예제

■ MPI 프로그램 작업 스크립트 작성 예제(mpi.sh)

```
#!/bin/sh
#PBS -N mvapich2_job
#PBS -V
#PBS -q norm_cache
#PBS -l select=4:ncpus=68:mpiprocs=68
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

mpirun ./a.out
```

※ 4노드 점유, 노드 당 68 프로세스(총 272 MPI 프로세스) 사용 예제

■ Hybrid(MPI+OpenMP) 프로그램 작업 스크립트 작성 예제(hybrid.sh)

```
#!/bin/sh
#PBS -N hybrid_job
#PBS -V
#PBS -q norm_cache
#PBS -l select=4:ncpus=68:mpiprocs=2:ompthreads=34
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

mpirun ./a.out  # impi나 mvapich2 사용할 경우 or
mpirun --map-by NUMA:PE=34 ./a.out  # openmpi 사용할 경우
```

※ 4노드 점유, 노드 당 2 프로세스, 34 스레드 (총 8 MPI 프로세스, 136 OpenMP 스레드) 사용 예제

■ 상용SW Gaussian 사용그룹 스크립트 작성 예제

```
#!/bin/sh
#PBS -N gauss_job
#PBS -V
#PBS -q norm_skl
#PBS -l select=1:ncpus=40:mpiprocs=1:ompthreads=40
#PBS -l walltime=04:00:00

cd $PBS_O_WORKDIR

export g16root="/apps/commercial/G16"
export g16BASIS=${g16root}/g16/basis
export GAUSS_EXEDIR=${g16root}/g16
export GAUSS_LIBDIR=${g16root}/g16
export GAUSS_ARCHDIR=${g16root}/g16/arch
export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:${g16root}/g16"
export PATH="${PATH}:${g16root}/g16"
export GAUSS_SCRDIR="/scratch/${USER}"

g16 test.com
```

■ 싱귤레러티 컨테이너에서 텐서플로우 프로그램 실행 예제 (단일노드-KNL)

```
#!/bin/sh
#PBS -N tensorflow_job
#PBS -V
#PBS -q norm_cache
#PBS -l select=1:ncpus=68
#PBS -l walltime=1:00:00

cd $PBS_O_WORKDIR
module load singularity/2.5.1
export KMP_BLOCKTIME=1
export KMP_AFFINITY=granularity=fine,compact,1,0
export OMP_NUM_THREADS=68

singularity exec tensorflow-1.9.0-py3.simg python convolutional.py
```

○ 작업 이메일 알림 지정

```
$ qsub -m <options>

ex) qsub -m abe hello_world.sh
```

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

2) 인터렉티브 작업 제출

○ 배치 스크립트 대신 "-I" 옵션 사용

```
$ qsub -q [큐 이름] -I
```

○ 인터렉티브 작업 제출 시 그래픽 환경 사용

```
$ qsub -q [큐 이름] -I -X
```

○ 인터렉티브 작업 제출 시 기존 환경변수 상속

```
$ qsub -q [큐 이름] -l -V
```

※ 디버그 노드를 제공하지 않고, 인터렉티브 작업 제출로 디버깅 수행을 권장함

■ 인터렉티브 모드로 계산 노드에서 텐서플로우 프로그램 실행 예제

```
$ qsub -I -l select=1:ncpus=68 -q norm_cache

$ export KMP_BLOCKTIME=1;
export KMP_AFFINITY=granularity=fine,compact,1,0;
export OMP_NUM_THREADS=68;
singularity exec tensorflow-1.9.0-py3.simg python convolutional.py
```

3) 작업 모니터링

○ 큐 조회

```
$ showq
```

○ 노드 상태 조회

```
$ pbsnodes -ajS
```

컬럼	설 명
mem	기가바이트(GB)단위의 메모리 양
ncpus	이용 가능한 총 CPU 수
f/t	f=free, t=total

○ 작업 상태 조회

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

 ex> qstat
Job id           Name            User             Time Use S Queue
-----------------------------------------------------------------
0001.pbcm        test_01         user01           8245:43: R knlc
0002.pbcm        test_02         user02           8245:44: R knlf
0003.pbcm        test_03         user03           7078:45: R cpu
0003.pbcm        test_04         user04           1983:11: Q knlc
```

○ 작업 속성 조회

```
$ qstat -f <job ID>

ex> qstat -f 0000.pbcm
Job Id: 0000.pbcm
    Job_Name = test
    Job_Owner = user0@pbcm.cm.cluster
    resources_used.cpupercent = 6416
    resources_used.cput = 8245:43:20
    resources_used.mem = 33154824kb
    resources_used.ncpus = 64
    resources_used.vmem = 99989940kb
    resources_used.walltime = 128:54:21
    job_state = R
<생략>
```

다. 작업 제어

○ 작업 삭제

```
$ qdel <job ID>
```

○ 작업 suspend/resume

```
$ qsig -s <suspend/resume> <job ID>
```

5. 사용자 지원

사용 중 문제가 생기거나 의문사항이 있으면 헬프데스크를 통해 문의한다.

분야	연락처
기술지원 일반 (계약/계정/시스템/ 최적화/병렬화)	**슈퍼컴퓨팅본부 헬프데스크 : http://helpdesk.ksc.re.kr**
교육 문의	슈퍼컴퓨팅 교육: https://webedu.ksc.re.kr - 교육지원

[별첨1] 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	<alphanumeric>	어카운팅 문자열 지정
-a	[[[[YYYY]MM]DD]hhmm[.S]]	예약된 실행
-C	"<string>"	PBS 지시어의 접두어(ex>#PBS) 변경
-c	[c | c=numeric | w | w=<minutes> | n | s | u]	Job의 체크포인트 주기 지정
-e	</path/filename>	error 파일의 경로 지정
-h		Job 실행 지연(지연실행)
-I | -X		X11 포워딩 활성화된 Interactive Job
-J	<X-Y[:z>]	Job 배열 정의
-j	[o | e]	output과 error 파일 병합
-k	[o | e]	실행 호스트에서 output과 error 파일 유지 여부
-l	<resource_list>	Job 리소스 요청
-M	<id@domain.xxx>	이메일 받는 사람 리스트 설정
-m	<string>	이메일 알람 지정
-N	<alphanumeric>	Job 이름 지정
-o	</path/filename>	output 파일의 경로 지정
-P	<alphanumeric>	프로젝트 이름 지정
-p	<numeric>	Job 우선순위 설정
-q	<queue_name>	서버나 큐의 이름 지정
-r	[Y | N]	재실행 가능여부 마킹
-S	</path/shell_name>	사용할 스크립트 언어 지정
-u	<user_name>	Job의 사용자 ID 지정
-V		환경변수 내보내기
-v	<alphanumeric>	환경변수들 확장

옵션	형식	설명
-W	<attr=value>	Job의 속성 값 설정
-W block=	<Boolean>	Job의 완료를 기다리는 qsub 요청
-W depend=	<dependency type>	Job의 의존성 지정
-W group_list=	<group@host...>	Job 소유자의 group ID 지정
-W pwd=	"<alphanumeric>"	작업 당 password 방식
-W run_count=	<numeric>	작업이 실행되어지는 횟수 제어
-W sandbox=	[HOME | PRIVATE]	스테이징 디렉터리와 실행 디렉터리
-W stagein=	<list>	input 파일 스테이징
-W stageout=	<list>	output 파일 스테이징
-W umask=	<numeric>	UNIX의 JOB umask 지정
-X		Interactive Job으로 부터의 X output
-z		함축된 작업 식별자

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

[별첨2] 소프트웨어 설치 목록

- 컴파일러 및 응용 소프트웨어

구분	이름	버전	Modulefile	CPU 구분	설치 경로
컴파일러	INTEL	17.0.5	intel/17.0.5	-	/cm/shared/apps/intel/compilers_and_libraries/2017.5.239/linux
	INTEL	18.0.1	intel/18.0.1	-	/apps/compiler/intel/18.0.1
	INTEL	18.0.3	intel/18.0.3	-	/apps/compiler/intel/18.0.3
응용 SW (공개)	gromacs	2016.4	gromacs/2016.4	knl	/apps/applications/GROMACS/2016.4/intel/17.0.5/impi/17.0.5/mic-knl
				skl	/apps/applications/GROMACS/2016.4/intel/17.0.5/impi/17.0.5/x86-skylake
	lammps	8Mar18	lammps/8Mar18	knl	/apps/applications/LAMMPS/8Mar18/intel/17.0.5/impi/17.0.5/mic-knl
				skl	/apps/applications/LAMMPS/8Mar18/intel/17.0.5/impi/17.0.5/x86-skylake
	namd	2.12	namd/2.12	knl	/apps/applications/NAMD/2.12/intel/18.0.3/impi/18.0.3/mic-knl
				skl	/apps/applications/NAMD/2.12/intel/18.0.3/impi/18.0.3/x86-skylake
	Open FOAM	4.1	-	knl	/apps/applications/OpenFOAM/4.1/intel/17.0.5/impi/17.0.5/x86-skylake
				skl	/apps/applications/OpenFOAM/4.1/intel/17.0.5/impi/17.0.5/mic-knl
	qe	6.1	qe/6.1	knl	/apps/applications/QE/6.1/intel/17.0.5/impi/17.0.5/mic-knl
				skl	/apps/applications/QE/6.1/intel/17.0.5/impi/17.0.5/x86-skylake
	siesta	4.0.2	siesta/4.0.2	knl	/apps/applications/SIESTA/4.0.2/intel/17.0.5/impi/17.0.5/mic-knl
				skl	/apps/applications/SIESTA/4.0.2/intel/17.0.5/impi/17.0.5/x86-skylake
		4.1-b3	siesta/4.1-b3	knl	/apps/applications/SIESTA/4.1-b3/intel/17.0.5/impi/17.0.5/mic-knl
				skl	/apps/applications/SIESTA/4.1-b3/intel/17.0.5/impi/17.0.5/x86-skylake
	grads	2.2.0	grads/2.2.0	-	/apps/applications/GRADS/2.2.0
	python	2.7	python/2.7	-	/apps/applications/PYTHON/2.7
		3.7	python/3.7	-	/apps/applications/PYTHON/3.7
	R	3.5.0	R/3.5.0	-	/apps/applications/R/3.5.0
	cmake	3.12.3	cmake/3.12.3	-	/apps/applications/cmake/3.12.3
응용 SW (상용)	gaussian16	g16.a03	gaussian/g16	skl	/apps/commercial/G16/g16

- 공통 라이브러리

구분	이름	버전	설치 경로
공통 라이브러리	cairo	1.14.6	/apps/common/cairo/1.14.6
	curl	7.59.0	/apps/common/curl/7.59.0
	expat	2.2.5	/apps/common/expat/2.2.5
	fontconfig	2.13.0	/apps/common/fontconfig/2.13.0
	freetype	2.9.1	/apps/common/freetype/2.9.1
	g2clib	1.5.0	/apps/common/g2clib/1.5.0
	gdal	2.1.1	/apps/common/gdal/2.1.1
	gperf	3.1	/apps/applications/gperf/3.1
	grib_api	1.26.1	/apps/common/grib_api/1.26.1
	iconv	1.15	/apps/common/iconv/1.15
	jasper	1.900.29	/apps/common/jasper/1.900.29
	jpeg	9c	/apps/common/jpeg/9c
	libpng	1.2.56	/apps/common/libpng/1.2.56
	pixman	0.34.0	/apps/common/pixman/0.34.0
	pkg-config	0.29.2	/apps/common/pkg-config/0.29.2
	proj	4.9.2	/apps/common/proj/4.9.2
	swig	3.0.12	/apps/common/swig/3.0.12
	szip	2.1.1	/apps/common/szip/2.1.1
	udunits	2.2.24	/apps/common/udunits/2.2.24
	readline	7.0	/apps/common/readline/7.0
	ncurses	6.1	/apps/common/ncurses/6.1
	zlib	1.2.11	/apps/common/zlib/1.2.11

- 컴파일러 의존 라이브러리

구분	이름	버전	Modulefile	compiler	CPU구분	설치 경로
라이브러리 (applib1)	lapack	3.7.0	lapack/3.7.0	intel-17.0.5	knl	/apps/compiler/intel/17.0.5/applib1/mic-knl/lapack/3.7.0
					skl	/apps/compiler/intel/17.0.5/applib1/x86-skylake/lapack/3.7.0
				intel-18.0.1	knl	/apps/compiler/intel/18.0.1/applib1/mic-knl/lapack/3.7.0
					skl	/apps/compiler/intel/18.0.1/applib1/x86-skylake/lapack/3.7.0
				intel-18.0.3	knl	/apps/compiler/intel/18.0.3/applib1/mic-knl/lapack/3.7.0
					skl	/apps/compiler/intel/18.0.3/applib1/x86-skylake/lapack/3.7.0
	hdf4	4.2.13	hdf4/4.2.13	intel-17.0.5	knl	/apps/compiler/intel/17.0.5/applib1/mic-knl/hdf4/4.2.13
					skl	/apps/compiler/intel/17.0.5/applib1/x86-skyl

						ake/hdf4/4.2.13
				intel-18.0.1	knl	/apps/compiler/intel/18.0.1/applib1/mic-knl/hdf4/4.2.13
					skl	/apps/compiler/intel/18.0.1/applib1/x86-skylake/hdf4/4.2.13
				intel-18.0.3	knl	/apps/compiler/intel/18.0.3/applib1/mic-knl/hdf4/4.2.13
					skl	/apps/compiler/intel/18.0.3/applib1/x86-skylake/hdf4/4.2.13
	hdf5	1.10.2	hdf5/1.10.2	intel-17.0.5	knl	/apps/compiler/intel/17.0.5/applib1/mic-knl/hdf5/1.10.2
					skl	/apps/compiler/intel/17.0.5/applib1/x86-skylake/hdf5/1.10.2
				intel-18.0.1	knl	/apps/compiler/intel/18.0.1/applib1/mic-knl/hdf5/1.10.2
					skl	/apps/compiler/intel/18.0.1/applib1/x86-skylake/hdf5/1.10.2
				intel-18.0.3	knl	/apps/compiler/intel/18.0.3/applib1/mic-knl/hdf5/1.10.2
					skl	/apps/compiler/intel/18.0.3/applib1/x86-skylake/hdf5/1.10.2
	netcdf4	4.6.1	netcdf/4.6.1	intel-17.0.5	knl	/apps/compiler/intel/17.0.5/applib1/mic-knl/netcdf/4.6.1
					skl	/apps/compiler/intel/17.0.5/applib1/x86-skylake/netcdf/4.6.1
				intel-18.0.1	knl	/apps/compiler/intel/18.0.1/applib1/mic-knl/netcdf/4.6.1
					skl	/apps/compiler/intel/18.0.1/applib1/x86-skylake/netcdf/4.6.1
				intel-18.0.3	knl	/apps/compiler/intel/18.0.3/applib1/mic-knl/netcdf/4.6.1
					skl	/apps/compiler/intel/18.0.3/applib1/x86-skylake/netcdf/4.6.1
	ncarg	6.4.0	ncl/6.4.0	intel-17.0.5	knl	/apps/compiler/intel/17.0.5/applib1/mic-knl/ncl/6.5.0
					skl	/apps/compiler/intel/17.0.5/applib1/x86-skylake/ncl/6.5.0
				intel-18.0.1	knl	/apps/compiler/intel/18.0.1/applib1/mic-knl/ncl/6.5.0
					skl	/apps/compiler/intel/18.0.1/applib1/x86-skylake/ncl/6.5.0
				intel-18.0.3	knl	/apps/compiler/intel/18.0.3/applib1/mic-knl/ncl/6.5.0
					skl	/apps/compiler/intel/18.0.3/applib1/x86-skylake/ncl/6.5.0
	ncview	2.1.7	ncview/2.1.7	intel-17.0.5	-	/apps/compiler/intel/17.0.5/applib1/x86-skylake/ncview/2.1.7
				intel-18.0.1	-	/apps/compiler/intel/18.0.1/applib1/x86-skylake/ncview/2.1.7
				intel-	-	/apps/compiler/intel/18.0.3/applib1/x86-skyl

				18.0.3		ake/ncview/2.1.7
MPI	impi	17.0.5	impi/17.0.5	intel-17.0.5	-	/cm/shared/apps/intel/compilers_and_libraries/2017.5.239/linux/mpi
	impi	18.0.1	impi/18.0.1	intel-18.0.1	-	/apps/compiler/intel/18.0.1/impi/2018.1.163
	impi	18.0.3	impi/18.0.3	intel-18.0.3	-	/apps/compiler/intel/18.0.3/impi/2018.3.222
	Mvapich2	2.3	mvapich2/2.3	intel-17.0.5	-	/apps/compiler/intel/17.0.5/mvapich2/2.3
				intel-18.0.1	-	/apps/compiler/intel/18.0.1/mvapich2/2.3
				intel-18.0.3	-	/apps/compiler/intel/18.0.3/mvapich2/2.3
	OpenMPI	3.1.0	openmpi/3.1.0	intel-17.0.5	-	/apps/compiler/intel/17.0.5/openmpi/3.1.0
				intel-18.0.1	-	/apps/compiler/intel/18.0.1/openmpi/3.1.0
				intel-18.0.3	-	/apps/compiler/intel/18.0.3/openmpi/3.1.0

- MPI 의존 라이브러리

구분	이름	버전	Modulefile	compiler	CPU구분	MPI	설치 경로
	fftw	2.1.5	fftw_mpi/2.1.5	intel-17.0.5	knl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/mic-knl/fftw_mpi/2.1.5
					knl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/mic-knl/fftw_mpi/2.1.5
					knl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/mic-knl/fftw_mpi/2.1.5
					skl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/x86-skylake/fftw_mpi/2.1.5
					skl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/x86-skylake/fftw_mpi/2.1.5
					skl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/x86-skylake/fftw_mpi/2.1.5
				intel-18.0.1	knl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/mic-knl/fftw_mpi/2.1.5
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/mic-knl/fftw_mpi/2.1.5
					knl	openmpi-3.	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/mic-knl/fftw_mpi/2.1.5

						1.0	
					skl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/x86-skylake/fftw_mpi/2.1.5
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/x86-skylake/fftw_mpi/2.1.5
					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/x86-skylake/fftw_mpi/2.1.5
				intel-18.0.3	knl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/mic-knl/fftw_mpi/2.1.5
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/mic-knl/fftw_mpi/2.1.5
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/mic-knl/fftw_mpi/2.1.5
					skl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/x86-skylake/fftw_mpi/2.1.5
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/x86-skylake/fftw_mpi/2.1.5
					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/x86-skylake/fftw_mpi/2.1.5
		3.3.7	fftw_mpi/3.3.7	intel-17.0.5	knl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/mic-knl/fftw_mpi/3.3.7
					knl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/mic-knl/fftw_mpi/3.3.7
					knl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/mic-knl/fftw_mpi/3.3.7
					skl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/x86-skylake/fftw_mpi/3.3.7
					skl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/x86-skylake/fftw_mpi/3.3.7
					skl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/x86-skylake/fftw_mpi/3.3.7
				intel-18.0.1	knl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/mic-knl/fftw_mpi/3.3.7
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/mic-knl/fftw_mpi/3.3.7
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/mic-knl/fftw_mpi/3.3.7
					skl	impi-	/apps/compiler/intel/18.0.1/impi/2018.1.163/a

						18.0.1	pplib2/x86-skylake/fftw_mpi/3.3.7
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/x86-skylake/fftw_mpi/3.3.7
					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/x86-skylake/fftw_mpi/3.3.7
				intel-18.0.3	knl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/mic-knl/fftw_mpi/3.3.7
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/mic-knl/fftw_mpi/3.3.7
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/mic-knl/fftw_mpi/3.3.7
					skl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/x86-skylake/fftw_mpi/3.3.7
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/x86-skylake/fftw_mpi/3.3.7
					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/x86-skylake/fftw_mpi/3.3.7
	phdf5	1.10.2	hdf5-parallel/1.10.2	intel-17.0.5	knl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/mic-knl/hdf5-parallel/1.10.2
					knl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/mic-knl/hdf5-parallel/1.10.2
					knl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/mic-knl/hdf5-parallel/1.10.2
					skl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/x86-skylake/hdf5-parallel/1.10.2
					skl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/x86-skylake/hdf5-parallel/1.10.2
					skl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/x86-skylake/hdf5-parallel/1.10.2
				intel-18.0.1	knl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/mic-knl/hdf5-parallel/1.10.2
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/mic-knl/hdf5-parallel/1.10.2
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/mic-knl/hdf5-parallel/1.10.2
					skl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/x86-skylake/hdf5-parallel/1.10.2
					skl	mvapi	/apps/compiler/intel/18.0.1/mvapich2/2.3/app

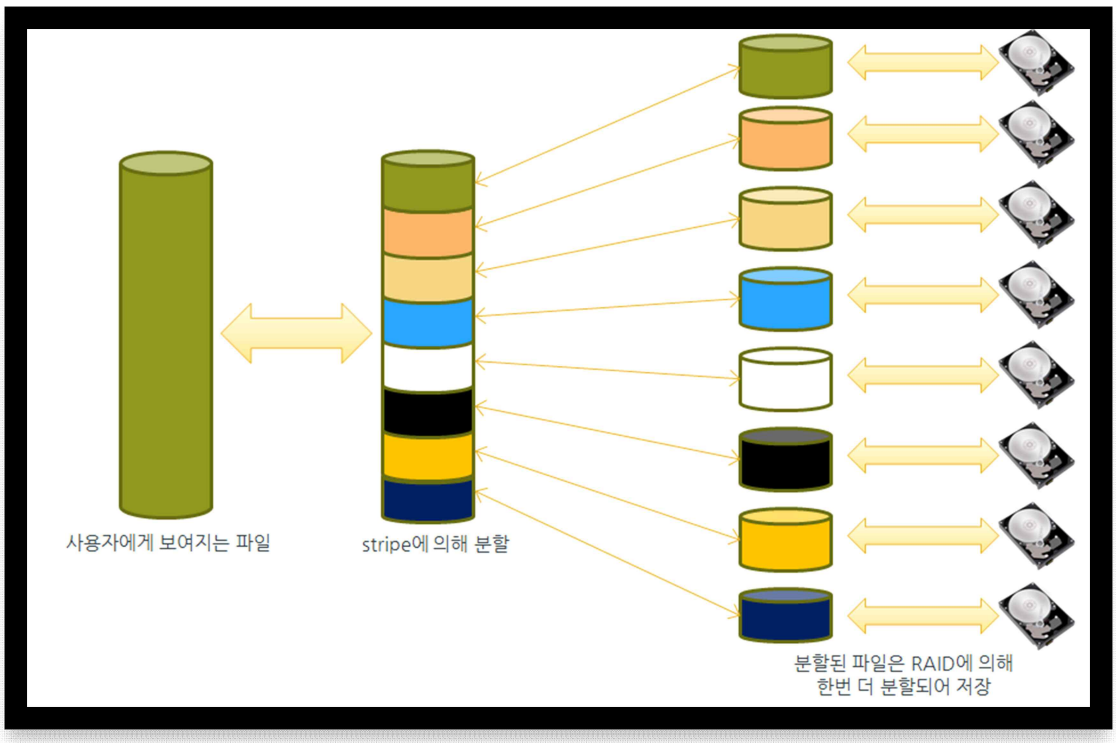
						ch2-2.3	lib2/x86-skylake/hdf5-parallel/1.10.2
					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/x86-skylake/hdf5-parallel/1.10.2
				intel-18.0.3	knl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/mic-knl/hdf5-parallel/1.10.2
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/mic-knl/hdf5-parallel/1.10.2
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/mic-knl/hdf5-parallel/1.10.2
					skl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/x86-skylake/hdf5-parallel/1.10.2
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/x86-skylake/hdf5-parallel/1.10.2
					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/x86-skylake/hdf5-parallel/1.10.2
	pnetcdf4	4.6.1	netcdf-hdf5-parallel/4.6.1	intel-17.0.5	knl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					knl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					knl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					skl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1
					skl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1
					skl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1
				intel-18.0.1	knl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					skl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1

					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1
				intel-18.0.3	knl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/mic-knl/netcdf-hdf5-parallel/4.6.1
					skl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1
					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/x86-skylake/netcdf-hdf5-parallel/4.6.1
	parallel-netcdf	1.10.0	parallel-netcdf/1.10.0	intel-17.0.5	knl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/mic-knl/parallel-netcdf/1.10.0
					knl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/mic-knl/parallel-netcdf/1.10.0
					knl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/mic-knl/parallel-netcdf/1.10.0
					skl	impi-17.0.5	/apps/compiler/intel/17.0.5/impi/17.0.5/applib2/x86-skylake/parallel-netcdf/1.10.0
					skl	mvapich2-2.3	/apps/compiler/intel/17.0.5/mvapich2/2.3/applib2/x86-skylake/parallel-netcdf/1.10.0
					skl	openmpi-3.1.0	/apps/compiler/intel/17.0.5/openmpi/3.1.0/applib2/x86-skylake/parallel-netcdf/1.10.0
				intel-18.0.1	knl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/mic-knl/parallel-netcdf/1.10.0
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/mic-knl/parallel-netcdf/1.10.0
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/mic-knl/parallel-netcdf/1.10.0
					skl	impi-18.0.1	/apps/compiler/intel/18.0.1/impi/2018.1.163/applib2/x86-skylake/parallel-netcdf/1.10.0
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.1/mvapich2/2.3/applib2/x86-skylake/parallel-netcdf/1.10.0
					skl	openmpi-3.	/apps/compiler/intel/18.0.1/openmpi/3.1.0/applib2/x86-skylake/parallel-netcdf/1.10.0

						1.0	
				intel-18.0.3	knl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/mic-knl/parallel-netcdf/1.10.0
					knl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/mic-knl/parallel-netcdf/1.10.0
					knl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/mic-knl/parallel-netcdf/1.10.0
					skl	impi-18.0.3	/apps/compiler/intel/18.0.3/impi/2018.3.222/applib2/x86-skylake/parallel-netcdf/1.10.0
					skl	mvapich2-2.3	/apps/compiler/intel/18.0.3/mvapich2/2.3/applib2/x86-skylake/parallel-netcdf/1.10.0
					skl	openmpi-3.1.0	/apps/compiler/intel/18.0.3/openmpi/3.1.0/applib2/x86-skylake/parallel-netcdf/1.10.0

[별첨3] Lustre stripe 사용법

□ Lustre Striping



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

□ 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
· 기본 값은 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 설정 값을 확인하는 명령어

□ 권장사항 및 팁

- ◦ 작업 스크립트 내에서 모델의 결과파일이 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