KISTI TACHYON
사용자 지침서

Ver. 1.0

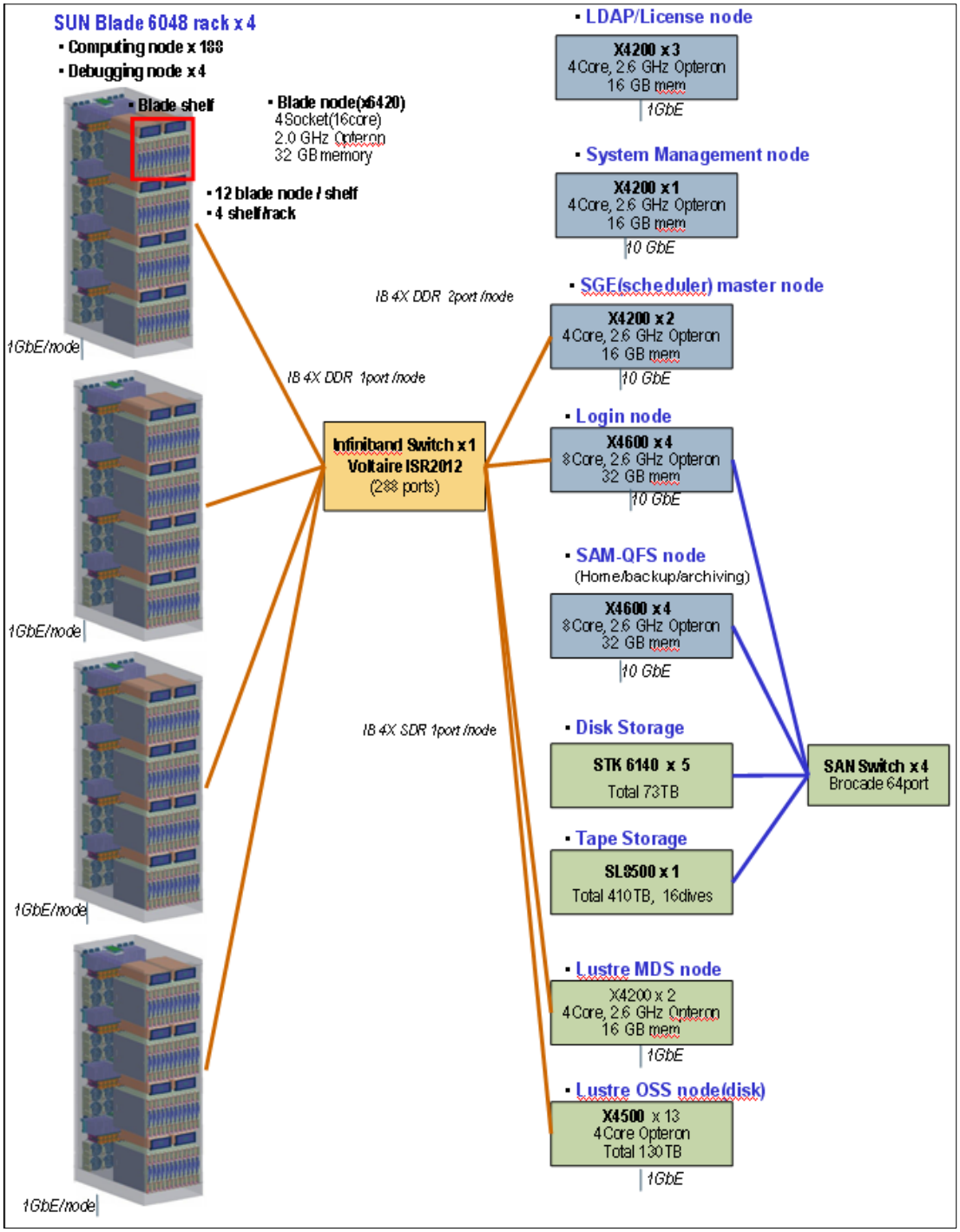
슈퍼컴퓨팅센터

KiSTi 한국과학기술정보연구원
www.kisti.re.kr
Korea Institute of Science and Technology Information

1. 시스템 사양 및 구성

슈퍼컴퓨터 4호기 초병렬컴퓨팅시스템(1차분)인 TACHYON(타키온)은 SUN의 Blade 6048 시스템으로 이루어져 있으며, 이론최고성능(Rpeak) 24TFlops 제공

구 분	내 용	비 고
제조사 및 모델	SUN Blade 6048	
아키텍처	클러스터	블레이드 타입
프로세서	AMD Opteron 2.0GHz(Barcelona)	시스템 버스 : HyperTransport (6.4GB/sec) L1/L2/L3 : 64KB/4*512KB/2MB on-die
노드수	**컴퓨팅 노드 188개**	로그인 노드 4개(X4600) 디버깅 노드 4개(Blade 6048)
CPU 코어수	**3,008개**	**16개/노드**
이론최고성능(Rpeak)	**24TFlops**	
메모리	DDR2/667MHz 6TB	32GB/노드, **2GB/코어**
디스크 스토리지	SUN X4500/STK6140	207TB
테이프 스토리지	SUN SL8500	422TB
Interconnection 네트워크	Infiniband 4X DDR	Voltaire ISR 2012 스위치
쿨링 방식	수냉식	Libert XDP/XDH
운영체제	**CentOS 4.6**	Kernel 2.6.9-67.0.4.ELsmp
파일시스템	Lustre 1.6.4.2 SAM-QFS 4.6	스크래치 디렉터리 홈 디렉터리
아카이빙 프로그램	SAM-QFS 4.6	
작업 관리 프로그램	SGE 6.1	

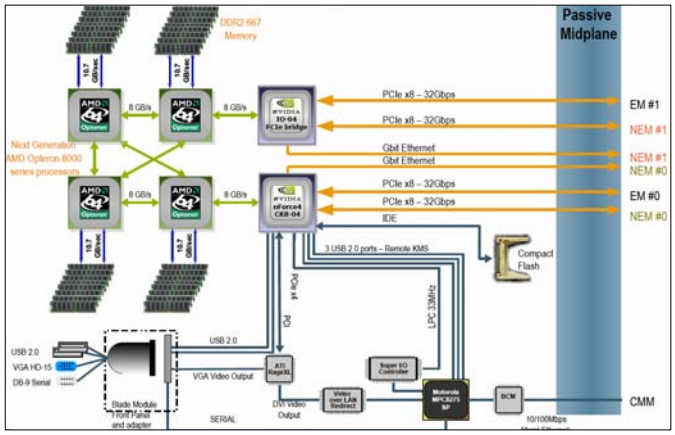


[TACHYON 시스템 구성도]

1) 계산 노드

188개의 계산 노드들은 높은 집적도를 가진 4개의 Sun Blade 6048 랙에 장착되어 있다. 각각의 Sun Blade 6048 랙은 4개의 Shelf로 구성되며, 각각의 shelf에는 12개의 x6420 블레이드 노드가 장착되어 있음. 각각의 x6420 블레이드는 AMD사의 쿼드코어 2.0GHz CPU(Barcelona) 4개와 32GB 메모리, 그리고 OS를 위한 8GB CF drive와 2개의 x8 PCI-e connection으로 이루어져 있음

[SUN Blade 6048]



[x6420 블레이드 노드 블록 다이어그램]

2) Interconnection 네트워크

Infiniband는 노드 간 통신을 위한 주 백본 네트워크로, 4x IB DDR을 사용한 non-blocking IB 네트워크로 구축되어 있으며 채널 당 2.5GB/sec(20 Gbps)의 대역폭을 가짐.

IB 네트웍 구성을 위하여 1대의 288 port Voltaire ISR 2012 IB 스위치에 모든 계산 노드와 그 외 IB 연결이 필요한 인프라 노드들을 4X IB로 연결하여 Non-Blocking IB 네트워크로 구성함

[Voltaire ISR2012 IB 스위치]

[IB 스위치 후면]

3) 스토리지

TACHYON의 스토리지 시스템은 IB기반의 스크래치 디렉터리 디스크(SUN X4500) 130.2TB와 SAN 기반의 홈 디렉터리 디스크(STK 6140) 73.2TB로 구성됨. 스크래치 디렉터리는 Lustre 기반의 공유파일시스템으로 구성되어 있으며, 홈 디렉터리는 SAM-QFS 기반의 공유파일시스템 및 아카이빙 방식의 백업시스템으로 구성되어 있음. SAM-QFS MDS 서버는 홈 디렉터리 파일시스템을 기반으로 총 4대(X4600)로 구성되며, HA(Active-Standby)로 구축되어 최적의 성능을 보장함.

홈 디렉터리에 대한 백업은 SAM-QFS 서버에 의해 실시간 아카이빙 방식의 백업이 수행됨. 1차 아카이빙은 SAN용 디스크 스토리지(STK6140)에 2차 아카이빙은 테이프 라이브러리에 저장되며, 실시간 아카이빙 방식으로 별도의 백업윈도우를 필요로 하지 않음.

HSM용 아카이빙은 로그인 노드에서 사용자에 의해 FTP 방식의 데이터 전송 후 자동으로 아카이빙 되는 방식으로 운영됨.

[SUN X4500 디스크 및 X4600 서버]

[SUN STK 6140 SAN 디스크]

2. 기본 사용자 환경

가. 로그인

- 사용자의 초기 접속은 총 4대의 로그인 노드로 한정됨
- 액세스 인터페이스는 ssh, sftp, ftp, X11 만 허용됨

① 유닉스 혹은 리눅스에서

```
$ ssh -l 사용자ID  hostname
     또는
$ ssh -l 사용자ID IP address
```

② 윈도우즈에서

- putty 나 SSH Secure Shell Client 등의 ssh 접속 유틸리티를 이용함
- 프로그램은 인터넷을 통해 무료로 다운 받을 수 있음

③ 노드 구성

비고	호스트 이름	IP 주소	CPU Limit (Interactive Process)
로그인 노드 (4노드)	tachyon.ksc.re.kr	DNS 대표 호스트 네임	10분
	tachyona.ksc.re.kr	150.183.147.213	
	tachyonb.ksc.re.kr	150.183.147.214	
	tachyonc.ksc.re.kr	150.183.147.215	
	tachyond.ksc.re.kr	150.183.147.216	
디버깅 노드 (4노드)	tachyon189	● 컴파일 및 디버깅 ● 컴퓨팅 노드와 동일한 시스템	30분
	tachyon190		
	tachyon191		
	tachyon192		
컴퓨팅 노드 (188노드)	tachyon001-188	● SGE(배치 스케줄러)를 통해서만 작업 실행 가능함	1분

※ Process Resource Limit 확인

```
$ ulimit -a
```

나. 사용자 쉘 변경

기본으로 설정되는 bash에서 다른 shell로 변경했을 경우 사용자의 홈디렉터리에 있는 해당 환경설정 파일을 적절히 수정하여 사용함. 원본이 필요한 경우 사용자가 직접 /applic/shell 디렉터리에서 필요한 쉘의 환경 설정 파일을 자신의 홈 디렉터리로 복사하여 적절히 수정하여 사용함.

```
$ ldapchsh
```

다. 패스워드 변경

사용자 패스워드를 변경하기 위해서는 passwd 명령을 사용함

```
$ passwd
```

※ 패스워드 관련 보안 정책

① 사용자 password 길이를 8 character 이상 설정

② 사용자 password 변경 기간을 2개월로 설정

③ 새로운 패스워드는 이전 패스워드와 비교하여 2문자 이상 달라야 함.

④ 최대 허용 로그인 재시도 회수 :10회

라. 사용자 계정 SRU Time 사용량 확인

통합 슈퍼컴퓨터 계정 관리 시스템(ISAM)에 접속하여 사용자 계약정보, 배치 작업 사용 상세내역, 총 SRU Time 사용량 등을 확인할 수 있음

```
$ isam
```

마. 작업 수행 시 유의 사항

- 로그인 노드에서는 CPU time을 10분으로 제한하고 있기 때문에 프로그램 수정, 디버깅 및 배치 작업 제출 등의 기본적인 작업만 수행함

- CPU time으로 10분 이상 소요되는 디버깅 및 기타 인터랙티브 작업은 디버깅 노드에서 수행해야 함

- 홈 디렉터리는 용량 및 I/O 성능이 제한되어 있기 때문에, 모든 계산 작업은 /work01 혹은 /work02 스크래치 디렉터리의 사용자 작업 공간에서 이루어져야함

바. 작업 디렉터리

구분	내용	용량 제한	파일 삭제 정책	파일시스템 종류	백업 유무	디렉터리 마운트 여부		
						로그인 노드	컴퓨팅 노드	디버깅 노드
홈 디렉터리	/home01	구좌 당 6GB	-	Lustre	○	○	○	○
스크래치 디렉터리	/work01 /work02	사용자 당 1TB	4일 이상 액세스 하지 않은 파일 자동 삭제		×	○	○	○
애플리케이션 디렉터리	/applic	-	-		○	○	○	○

■ 홈 및 스크래치 디렉터리 용량 제한 및 사용량 확인

```
$ quotaprint

          [ USER DISK USAGE IN THE HOME & SCRATCH DIR ]

================================================================
 ID/GROUP       DIR   QUOTA_LIMIT     USED_DISK    AVAIL_DISK
================================================================
   in1000   /home01       12573MB       10937MB        1636MB
     test   /work01     1073740MB       54905MB     1018835MB
     test   /work02     1073740MB           0MB     1073740MB
================================================================
```

사. SAM-QFS(데이터 아카이빙)

홈 디렉터리의 용량을 초과한 사용자 데이터를 보관하기 위해서 SAM-QFS 기반의 데이터 아카이빙을 지원하며, SAM-QFS 사용법은 별도의 「SAM-QFS 사용자 지침서」를 참조할 것.

3. 사용자 프로그래밍 환경

가. 프로그래밍 도구 설치 현황 (/applic 디렉터리 참조)

구분	항목
컴파일러 (/applic/compilers)	● PGI CDK 7.1 ● Intel Compiler 10.1 ● gcc 3.4.6.9 (/usr)
프로파일러 (/applic/lib.{compiler})	● TAU 2.17
디버거 (/applic/debuggers)	● TotalView 8.3
MPI 라이브러리 (/applic/mpi)	● MVAPICH 1.0 ● OpenMPI 1.2.5 ※ 향후 MVAPICH2 지원 예정
수학 라이브러리 (/applic/lib.{compiler})	● Aztec 2.1 ● ACML 4.0.1 ● ATLAS 3.6 ● BLAS ● BLACS ● FFTW 3.1.2 ● GotoBLAS 1.23 ● LAPACK ● Scalapack 1.8
기타 라이브러리 (/applic/lib.{compiler})	● Petsc 2.3.3 ● HDF4 4.2 ● HDF5 1.8 ● NCARG 5.0.0-9 ● NetCDF 3.6.2-4 ● VTK 5.0.4
분야별 응용소프트웨어 (/applic/Applications)	기존 보유 응용소프트웨어를 단계적으로 설치 예정 ※ http://www.ksc.re.kr 보유자원-S/W 정보 참조

나. 프로그램 컴파일

본 시스템에서는 GNU 컴파일러 이외에 Portland Group(PGI) 컴파일러, Intel 컴파일러를 지원함.

또한 이들 컴파일러를 사용하여 MPI 라이브러리를 사전에 컴파일하여 사용자에게 제공함. 모든 프로그램은 PGI, Intel, GNU 컴파일러를 사용하여 컴파일 가능하며 MPI 환경을 이용한 컴파일도 가능함. (예: gcc, pgcc, icc, mpicc). 컴파일러에서

사용한 옵션은 MPI 환경에서도 사용할 수 있음.

병렬프로그래밍 환경의 지원을 위해서, 본 시스템에는 MVAPICH, OpenMPI를 지원함.

컴파일러	위치
PGI	/applic/compilers/pgi
Intel	/applic/compilers/intel
GCC	/usr

각 컴파일러에 대한 자세한 내용은 다음 웹 링크 참조

- GCC : http://gcc.gnu.org/onlinedocs/gcc-3.4.6/gcc.pdf
- PGI : http://www.pgroup.com/doc/pgiug.pdf
- Intel : http://www.intel.com/cd/software/products/asmo-na/eng/346158.htm

.

1) 컴파일러 환경변수

각 컴파일러에 맞는 환경변수의 자동 설정을 위해 .bashrc와 .bashrc-[mpi]-[compiler] 파일을 제공함.

예를 들면, pgi 컴파일러와 mvapich를 사용하기 위해서는 .bashrc-mvapich-pgi를 사용해야 함. 각 컴파일러별 .bashrc_[mpi]_[compiler]에는 PATH, MANPATH, LD_LIBRARY_PATH, LICENSE_FILE 등이 적절히 해당 컴파일러환경에 맞게 지정되어 있음.

또한 select-mpi-[bash|csh]을 통해 쉽게 다른 컴파일러 환경으로 변경하여 사용할 수 있음.

select-mpi-bash을 이용한 컴파일러 및 MPI 지정 (다음 로그인부터 적용)	
사용법	**select-mpi-bash [MPI] [Compiler]**
예제	**$ select-mpi-bash mvapich pgi**
선택 가능한 MPI	mvapich, openmpi
선택 가능한 Compiler	gnu, intel, pgi

2) 순차 프로그램 컴파일

각 컴파일러로 컴파일 할 수 있는 프로그램은 다음과 같음.

벤더	컴파일러 명령	프로그램	소스 확장자
PGI	pgcc	C	.c
	pgcpp	C++	.c, .C, .cc, .cpp
	pgf77	F77	.f, .for, .fpp, .F, .FOR
	pgf90/pgf95	F90/95	.f, .for, .f90, .f95, .fpp, .F, .FOR, .F90, .F95
Intel	icc	C	.c
	icc	C++	.c, .C, .cc, .cpp, .cxx,.c++
	ifort	F90	.f, .for, .ftn, .f90, .fpp, .F, .FOR, .FTN, .FPP, .F90
GCC	gcc	C	.c
	g++	C++	.C, .cc, .cpp, .cxx

■ 컴파일러 별 주요 옵션

최적화 컴파일을 위해 다음의 옵션을 부여할 수 있음

① GNU 컴파일러

컴파일러 옵션	설명
-O[1|2|3]	오브젝트 최적화. 숫자는 최적화 레벨
-funroll-all-loops	모든 루프를 unrolling함
-ffast-math	fast floating point model 사용
-minline-all-stringops	더 많은 inlining 허용
-g	디버깅 정보를 생성
--help	옵션 목록 출력

.

② Intel 컴파일러

컴파일러 옵션	설명
-O[1|2|3]	오브젝트 최적화. 숫자는 최적화 레벨
-ip, -ipo	프로시저 간 최적화
-vec_report[0|1|2|3|4]	벡터 진단 정보의 양을 조절
-xW	타겟 아키텍처 : SSE, SSE2 인스트럭션을 위한 코드를 포함
-fast	-xT -O3 -ipo -no-prec-div -static의 매크로
-static	공유 라이브러리를 링크하지 못하게 함
-g -fp	디버깅 정보를 생성
-openmp	OpenMP 기반의 multi-thread 코드 사용
-openmp_report[0|1|2]	OpenMP 병렬화 진단 레벨 조절
-help	옵션 목록 출력

※ 권장 옵션 : -O3 -axT -xW -m64 -fPIC -i_dynamic

③ PGI 컴파일러

컴파일러 옵션	설명
-O[0|1|2|3|4]	오브젝트 최적화. 숫자는 최적화 레벨
-Mipa=fast	프로시저 간 최적화
-fast	-O2 -Munroll=c:1 -Mnoframe -Mlre -Mautoinline 의 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
-help	옵션 목록 출력

■ 컴파일러 사용 예제

컴파일러	예제
GNU	gcc -o test.exe -O3 test.c
PGI	pgcc/pgcpp/pgf95 -o test.exe -fast test.c/cc/f90
Intel	icc/ifort -o test.exe -O3 -xW test.c/cc/f90

3) MPI 병렬 프로그래밍

사용자가 다음 표의 MPI 명령을 실행하게 되면, .bashrc를 통해 지정된 컴파일러에 해당하는 wrapper들이 소스를 컴파일하게 됨

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 함.

※ 사용 예제
intel : mpicc/mpif90 -o test.exe -O2 -xW test.cc/f90
pgi : mpicc/mpif90 -o test.exe -fast test.f90

■ 컴퓨팅 노드에서 MPI 프로그램 실행

컴퓨팅 노드(tachyon001-188)에서 MPI 프로그램을 실행하기 위해서는 스크래치 디렉터리로 작업 실행에 필요한 파일들을 스크래치 디렉터리로 복사한 후 SUN Grid Engine(SGE : 배치 작업 스케줄러)을 사용하여 배치 모드로 프로그램을 실행해야 함.

「4. SUN Grid Engine을 통한 작업 실행」 부분 참조

■ 디버깅 노드에서 MPI 프로그램 디버깅

디버깅 노드(tachyon189-192)에서 MPI 프로그램을 디버깅하기 위해서는 먼저 작업 실행에 필요한 파일들을 스크래치 디렉터리로 복사하고 디버깅 노드에 로그인 함.
MPI 프로그램의 실행은 MPI 종류와 관계없이 mpirun 명령을 사용함. -n 또는 -np 옵션으로 프로세스의 수를 지정함. -machinefile {filename} 옵션은 실행할 MPI 프로세스의 수 만큼 디버깅 노드의 호스트 이름을 지정함.

```
 [testuser01@tachyon189  /lustre1/testuser01]# cat hosts
tachyon189
tachyon189
tachyon190
tachyon190
tachyon191
tachyon191
tachyon192
tachyon192
```

```
[testuser01@tachyon189 /lustre1/testuser01]# mpirun -machinefile hosts -np 8 ./mpi_pi
Process 0 on tachyon189
pi is approximately 3.1416009869231254, Error is 0.0000083333333323
wall clock time = 0.000044
Process 0 on tachyon190
Process 1 on tachyon190
tachyon190
```

5) 라이브러리

본 시스템에는 수치 계산 및 기타 라이브러리들을 별도의 디렉터리에 제공함. 이러한 라이브러리들 각각은 컴파일러의 이름에 따라 /applic/lib.[compiler]에 설치되어 있음.

구분	내용
/applic/lib.pgi	pgi로 컴파일된 라이브러리들
/applic/lib.intel	intel로 컴파일된 라이브러리들
/applic/lib.gcc	gcc로 컴파일된 라이브러리들

또한 MPI를 사용하는 라이브러리들의 경우 각 MPI Library 별로 컴파일 되어 있음. 예를 들어 gcc로 컴파일된 BLACS의 경우 MPI 이름에 따라 다음과 같이 세 하위 디렉터리를 가짐.

구분	내용
/applic/lib.gcc/BLACS/mvapich	mvapich로 컴파일된 BLACS
/applic/lib.gcc/BLACS/openmpi	openmpi로 컴파일된 BLACS

컴파일러 환경 변수 파일 .bashrc-[mpi]-[compiler]에서, 사용자가 필요로 하는 MPI 및 컴파일러에 따라 적절한 라이브러리들의 경로가 LD_LIBRARY_PATH 변수에 지정됨.

6) 프로파일링

프로파일링 기능을 위해 TAU를 지원함.

- TAU의 기본적인 사용법

1. TAU API를 이용해 소스 코드를 instrument
2. instrumented 된 코드를 compile
3. 위의 과정을 통해 만들어진 바이너리를 실행
4. 작업이 종료 후 만들어진 performance file을 확인
5. 위의 단계에서 만들어진 performance file를 visualization등을 통해 분석

```
$ pprof // text-based
$ paraprof // GUI-based
```

7) 디버깅

디버거를 사용하기 위해서, 컴파일 시 -g 옵션을 추가함.

```
$ icc -g -lm -o simple simple.c
```

다음의 형식으로 Totalview를 실행시켜 디버깅을 수행.

```
$ totalview executable -a command_line_args
```

totalview의 자세한 사용방법은 다음의 웹 링크 참조.

- http://www.totalviewtech.com/Documentation/latest/pdf/User_Guid.pdf

4. SUN Grid Engine(SGE)을 통한 작업 실행

Sun Grid Engine (이하 SGE)은 Batch 작업과 Interactive 작업을 효율적으로 처리해 주는 작업 스케줄러임.

가. 큐 구성

큐이름	Wall Clock Limit (시간)	작업 실행 노드	작업별 CPU수	Priority	SU Charge Rate	비고
small	48	tachyon001-018	1-16	Low	1	16 CPU 이내 작업
normal	48	tachyon001-188	17-1536	normal	1	
long	168	tachyon001-188	1-128	Low	1	Long running 작업
strategy	TBD	tachyon001-188	32-3008	High	1	Grand Challenge 작업
special	12	tachyon001-188	1537-3008	-	2	대규모 자원 전용 (사전 예약)

※ 사용자별 최대 Runinng 작업수 : 10개(작업 부하에 따라 수시로 조정될 수 있음)

■ 큐 구성 정보 확인하기

```
$ showq
```

나. 작업 제출 및 모니터링

1) 작업 제출

SGE를 사용하여 배치 작업을 제출하기 위해서는 job script 파일을 작성하여 qsub 명령을 사용해야함. 예제 job script를 /applic/shell/job_examples/job_script 디렉터리에서 복사하여 사용할 수 있음

```
$ qsub job_script
```

■ Serial 프로그램(1CPU) 작업 스크립트 작성 예제(serial.sh)

```
#!/bin/bash
#$ -V                        # 작업 제출 노드의 쉘 환경변수를 컴퓨팅 노드에도 적용 (default)
#$ -cwd                      # 현재 디렉터리를 작업 디렉터리로 사용
#$ -N serial_job             # Job Name, 명시하지 않으면 job_script 이름을 가져옴
#$ -q small                  # Queue name
#$ -R yes                    # Resource Reservation
#$ -wd /work01/<user01>/serialtest # 작업 디렉터리를 설정. 현재 디렉토리(PWD)가
                                   # /work01/<user01>/serialtest가 아닌 경우 사용,
                                   # 그렇지 않으면 cwd로 충분함
#$ -l h_rt=01:00:00          # 작업 경과 시간 (hh:mm:ss) (wall clock time), 누락 시 작업 강제 종료
#$ -M myEmailAddress         # 작업 관련 메일을 보낼 사용자 메일 주소
#$ -m e                      # 작업 종료 시에 메일을 보냄
serial.exe
```

■ mpi 프로그램 작업 스크립트 작성 예제(mpi.sh)

① select-mpi-[shell] 명령어를 이용하여 job 실행환경 선택

```
$ select-mpi-bash [mvapich | openmpi] [pgi | intel | gnu]
$ exit
( exit 후 다시 로그인 해야 선택한 환경으로 설정됨)
```

② MPI task(CPU) 수 명시

```
#$  -pe mpi_fu {Total_MPI_task(CPU)}
#$  -pe mpi_fu 32
```

```
#!/bin/bash
#$ -V                        # 작업 제출 노드의 쉘 환경변수를 컴퓨팅 노드에도 적용 (default)
#$ -cwd                      # 현재 디렉터리를 작업 디렉터리로 사용
#$ -N mvapich_job            # Job Name, 명시하지 않으면 job_script 이름을 가져옴
#$ -pe mpi_fu 32             # selec-bash-mpi에서 선택한 mvapich로 실행되며 각 노드의 가용 cpu를
                             # 모두 채워서(fu : fill_up) 총 32개의 MPI task가 실행됨.
#$ -q normal                 # 큐 이름(17개 이상의 CPU를 사용하는 경우에는 normal 큐를
                             # 16개 이하 CPU를 사용하는 경우 small or long 큐 사용)
#$ -R yes                    # Resource Reservation
#$ -wd /work01/<user01>/mvapich # 작업 디렉터리를 설정. 현재 디렉토리(PWD)가
                                  # /work01/<user01>/mvapich가 아닌 경우 사용,
                                  # 그렇지 않으면 cwd로 충분함
#$ -l h_rt=01:00:00      # 작업 경과 시간 (hh:mm:ss) (wall clock time), 누락 시 강제 작업 종료
#$ -l normal                 # normal queue에 job실행 시 다른 queue보다 높은 priority를
                             # 얻기 위해 반드시 명시, 누락 시 작업 강제 종료
#$ -M myEmailAddress         # 작업 관련 메일을 보낼 사용자 메일 주소
#$ -m e                      # 작업 종료 시에 메일을 보냄
mpirun -machinefile $TMPDIR/machines -np $NSLOTS /work01/<user01>/mvapich/mpi.exe
```

■ 많은 메모리를 사용하는 mpi 프로그램 작업 스크립트 작성 예제(mpi_mem.sh)

```
#!/bin/bash
#$ -V
#$ -cwd
#$ -N mvapich_job
#$ -pe mpi_fu 32
#$ -q normal
#$ -R yes
#$ -l h_rt=01:00:00
#$ -l normal
#$ -M myEmailAddress
#$ -m e

#unset existing MPI affinities
export          MV2_USE_AFFINITY=0
export      MV2_ENABLE_AFFINITY=0
export      VIADEV_USE_AFFINITY=0
export VIADEV_ENABLE_AFFINITY=0
mpirun -np $NSLOTS -machinefile $TMPDIR/machines ./numa.sh
```

※ numa.sh

```
#!/bin/bash

#socket numbers in a compute node
SPN=4

#get my MPI rank
[ "x$PMI_RANK" != "x" ] && RANK=$PMI_RANK
[ "x$MPI_RANK" != "x" ] && RANK=$MPI_RANK
[ "x$MPIRUN_RANK" != "x" ] && RANK=$MPIRUN_RANK
[ "x$OMPI_MCA_ns_nds_vpid" != "x" ] && RANK=$OMPI_MCA_ns_nds_vpid

socket=$(( ($RANK + 3)  % $SPN  ))
echo "myrank: $RANK, mysocket: $socket, hostname: $(hostname)"

/usr/bin/numactl --cpunodebind=$socket --membind=$socket  ./mpi.exe
```

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

```
#!/bin/bash
#$ -V                        # 작업 제출 노드의 쉘 환경변수를 컴퓨팅 노드에도 적용 (default)
#$ -cwd                      # 현재 디렉터리를 작업 디렉터리로 사용
#$ -N openmp_job             # Job Name, 명시하지 않으면 job_script 이름을 가져옴
#$ -pe openmp 4              # OpenMP thread 수
#$ -q small                  # Queue name(OpenMP 작업은 small or long 큐 사용 가능)
#$ -R yes                    # Resource Reservation
#$ -wd /work02/<user01>/openmp   # 작업 디렉터리를 설정. 현재 디렉토리(PWD)가
                                 # /lustre1/<user01>/openmp가 아닌 경우 사용,
                                 # 그렇지 않으면 cwd로 충분함
#$ -l h_rt=01:00:00          # 작업 경과 시간 (hh:mm:ss) (wall clock time), 누락 시 작업 강제 종료
#$ -M myEmailAddress         # 작업 관련 메일을 보낼 사용자 메일 주소
#$ -m e                      # 작업 종료 시에 메일을 보냄
export OMP_NUM_THREADS=4
/work02/<user01>/omp.exe
```

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

Hybird 작업 수행 시에는 아래와 같은 옵션 및 환경변수 설정에 유의해야 함

① select-mpi-[shell] 명령어를 이용하여 job 실행환경 선택

```
$ select-mpi-bash [mvapich | openmpi] [pgi | intel | gnu]
$ exit
( exit 후 다시 로그인 해야 선택한 환경으로 설정됨)
```

② 노드 당 / 전체 MPI task(CPU) 수 명시

```
#$  -pe mpi_{MPI_task(CPU)_per_node}cpu {Total_MPI_task(CPU)}
#$  -pe mpi_4cpu 16
```

③ MPI 타스크 당 OpenMP 쓰레드 수를 의미하는 옵션으로 OMP_NUM_THREADS 리소스 지정

```
#$  -l OMP_NUM_THREADS={OpenMP_threads_per_MPI_task}
#$  -l OMP_NUM_THREADS=4
```

④ MPI task 당 OpenMP thread 수를 OMP_NUM_THREADS 환경변수로 명시

```
export OMP_NUM_THREADS=4
```

```
#!/bin/bash
#$ -V                        # 작업 제출 노드의 쉘 환경변수를 컴퓨팅 노드에도 적용 (default)
#$ -cwd                      # 현재 디렉터리를 작업 디렉터리로 사용
#$ -N hybrid_job             # Job Name, 명시하지 않으면 job_script 이름을 가져옴
#$ -pe mpi_4cpu 16           # 전체 MPI task(CPU) 16개, 노드 당 MPI task(CPU) 4개
#$ -q normal                 # Queue name
#$ -R yes                    # Resource Reservation
#$ -wd /work02/<user01>/hybrid   # 작업 디렉터리를 설정. 현재 디렉토리(PWD)가
                                 # /lustre1/<user01>/hybrid가 아닌 경우 사용,
                                 # 그렇지 않으면 cwd로 충분함
#$ -l h_rt=01:00:00          # 작업 경과 시간 (hh:mm:ss) (wall clock time), 누락 시 강제 작업 종료
#$ -l normal                 # normal queue에 job실행 시 다른 queue보다 높은 priority를
                             # 얻기 위해 반드시 명시, 누락 시 작업 강제 종료
#$ -l OMP_NUM_THREADS=4      # MPI 타스크 당 OpenMP 쓰레드 수를 의미하며, 아래
                             # OMP_NUM_THREADS 환경변수 에서 명기한 OpenMP
                             # 쓰레드 숫자와 동일한 값 지정. 누락 시 작업 강제 종료
#$ -M myEmailAddress         # 작업 관련 메일을 보낼 사용자 메일 주소
#$ -m e                      # 작업 종료 시에 메일을 보냄
export OMP_NUM_THREADS=4
mpirun -machinefile $TMPDIR/machines -np $NSLOTS /work02/<user01>/hybrid.exe
```

※ 작업 스크립트 옵션 리스트

옵션	Argument	기 능
-q	queue_name	작업을 수행할 queue 명시
-pe	pe_name min_proc[-max_proc]	Parallel Environment를 선택하고, min_proc-max_proc 개수만큼의 병렬 process를 수행 ● **mpi_rr : 라운드 로빈 방식으로 노드의 CPU 할당** ● **mpi_fu : 각 노드의 비어있는 CPU를 꽉 채워서 할당** ● **mpi_[1-16]cpu : 정해진(범위 : 1-16) 숫자 만큼 노드의 CPU 할당** ● **openmp : 순수한 openmp 프로그램의 쓰레드를 위한 CPU 할당** ※ mpi의 종류[mvapich,openmpi]는 select-mpi-[bash,csh,ksh] 스크립트로 미리 선택함.
-N	job_name	Job의 이름을 정해줌
-S	shell (absolute path)	Batch 작업의 shell을 지정. 미 지정 시 SGE가 지정한 shell로 수행(/bin/bash)
-M	Email address	사용자의 email address를 명시
-m	{b|e|a|s|n}	언제 email notification을 보낼 지 명시 ● b: Mail is sent at the beginning of the job. ● e: Mail is sent at the end of the job. ● a: Mail is sent when the job is aborted or rescheduled ● s: Mail is sent when the job is suspended. ● n: No mail is sent. (default)
-V		사용자의 현재 shell의 모든 환경변수가 qsub시에 job에 적용되도록 함
-cwd		현재 디렉터리를 job의 working directory로 사용.(default)
-o	output_file	Job의 stdout 결과를 output_file 로 저장
-e	error_file	Job의 stderr 결과를 error_file 로 저장
-l	resource=value	**Resource limit을 지정** ● **h_rt** : 작업경과 예상시간 (hh:mm:ss) (wall clock time) ● **normal** : normal 큐에 작업 제출 시 Job이 높은 우선순위를 얻기 위해 반드시 명시 (-l normal 혹은 -l normal=TRUE) ● **strategy :** strategy 큐에 작업 제출 시 작업이 높은 우선순위를 얻기 위해 반드시 명시 (-l strategy 혹은 -l strategy=TRUE) ● **OMP_NUM_THREADS** : MPI 타스크 당 쓰레드 수를 의미하며, hybrid[MPI+OpenMP] 병렬 작업 실행 시 반드시 명기 (-l OMP_NUM_THREADS=[MPI 타스크 당 OpenMP 쓰레드 수]) ※ normal, strategy 큐를 제외한 다른 큐는 -l 옵션으로 기본 priority이기 때문에 큐 이름을 명시할 필요 없음. 추후 변경 시 공지 예정

■ dependancy가 있는 다수 작업 제출 예제

① Job_A 가 끝난 후 Job_B 가 실행되어야 하는 경우

```
# qsub Job_A.sh  (Jobname은 Job_A라고 가정)
   Your job 504 ("Job_A") has been submitted
# qsub -hold_jid Job_A job_B.sh
   혹은
# qsub -hold_jid 504 job_B.sh
```

② Job_A와 job_B 가 끝난 후 Job_C 가 실행되어야 하는 경우

```
# qsub Job_A.sh  (Jobname은 Job_A라고 가정)
   Your job 504 ("Job_A") has been submitted
# qsub Job_B.sh  (Jobname은 Job_B라고 가정)
   Your job 505 ("Job_B") has been submitted
# qsub -hold_jid Job_A,Job_B Job_C.sh
   혹은
# qsub -hold_jid 504,505 Job_C.sh
```

2) 작업 모니터링

작업 제출 후에, 사용자는 **qstat** 명령을 이용하여 job의 상태를 모니터링 할 수 있음

■ 기본 작업 정보

```
$ qstat              (사용자 자신)
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
254    0.55500 work6      user1          r     04/02/2008 10:13:09 bmt.q@tachyon087               1
253    0.55500 work5      user1          r     04/01/2008 03:44:20 bmt.q@tachyon035               1
252    0.55500 work7      user1          r     04/01/2008 11:54:34 bmt.q@tachyon035               1

$ qstat -u '*'      (모든 사용자)
```

■ 상세 작업 정보

```
$ qstat -f -u "*"
queuename                      qtype used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@davinci02                BIP   1/4       0.14     lx24-amd64
    257 0.55500 sleep      root         r     04/01/2008 10:49:54     1
---------------------------------------------------------------------------------
all.q@davinci03                BIP   1/4       0.13     lx24-amd64
    258 0.55500 sleep      root         r     04/01/2008 10:49:54     1
---------------------------------------------------------------------------------
all.q@davinci04                BIP   2/4       0.01     lx24-amd64
    256 0.55500 sleep      root         r     04/01/2008 10:49:54     1
    261 0.55500 sleep      sgeadmin     r     04/01/2008 10:50:09     1
---------------------------------------------------------------------------------
all.q@grid01                   BIP   1/4       0.36     lx24-amd64
    259 0.55500 sleep      sgeadmin     r     04/01/2008 10:50:09     1
```

■ Pending 작업에 대한 상세 정보[Pending 이유] 출력

```
$ qstat -j
$ qstat -j job_id
```

Option	Result
no option	명령을 실행한 사용자 job의 상세 list를 보여줌
-f	명령을 실행한 사용자에 대한 queue와 job의 상세 리스트를 보여줌
-u *user_id*	명시한 user_id에 대한 상태를 보여줌. -u "*"는 전체 사용자의 상태를 보여줌. 주로 -f 옵션과 함께 쓰임.
-r	Job의 resource requirement를 display
-ext	Job의 Extended information을 display
-j <jobid>	Pending/running job에 대한 information을 보여줌
-qs {a|c|d|o|s|u|A|C|D|E|S}	명시한 status에 있는 job을 display
-t	Job의 subtask에 대한 추가 정보 display

[qstat 옵션]

3) 노드 상태 모니터링

```
$ showhost

HOSTNAME              ARCH          NCPU(AVAIL/TOT)    LOAD   MEMTOT MEMUSE SWAPTO SWAPUS
-------------------------------------------------------------------------------
tachyon001            lx24-amd64    0/16               16.03  31.4G  2.7G   0.0    0.0
tachyon002            lx24-amd64    0/16               16.00  31.4G  3.7G   0.0    0.0
tachyon003            lx24-amd64    0/16               16.04  31.4G  3.7G   0.0    0.0
tachyon004            lx24-amd64    0/16               16.03  31.4G  3.7G   0.0    0.0
tachyon005            lx24-amd64    1/16               15.53  31.4G  3.6G   0.0    0.0
tachyon006            lx24-amd64    0/16               16.00  31.4G  3.7G   0.0    0.0
tachyon007            lx24-amd64    0/16               16.01  31.4G  3.7G   0.0    0.0
tachyon008            lx24-amd64    2/16               14.03  31.4G  3.4G   0.0    0.0
tachyon009            lx24-amd64    0/16               16.00  31.4G  3.7G   0.0    0.0
tachyon010            lx24-amd64    0/16               16.03  31.4G  3.7G   0.0    0.0
tachyon011            lx24-amd64    0/16               16.03  31.4G  3.7G   0.0    0.0
tachyon012            lx24-amd64    0/16               16.02  31.4G  3.7G   0.0    0.0
tachyon013            lx24-amd64    0/16               16.03  31.4G  3.7G   0.0    0.0
```

4) 작업 제어

■ 작업 삭제

사용자는 qdel 명령을 이용하여 pending/running job을 queue로부터 삭제할 수 있음

```
$ qdel <jobid>           : 해당 <jobid>를 가지는 작업 삭제
$ qdel -u <username>  : <username>의 모든 작업 삭제
```

■ 작업 suspend/resume

사용자는 qmod 명령을 이용하여 running 상태의 job을 suspend/resume할 수 있음

```
$ qmod -sj <jobid>          # suspend job
$ qmod -usj <jobid>         # unsuspend(resume) job
```

■ 사용자 지원

- 일반 기술지원 : 홍태영, 042) 869-0667, tyhong@kisti.re.kr
- 응용 기술지원 : 정민중, 042) 869-0632, jeong@kisti.re.kr
- 시스템 계정 : 김성준, 042) 869-0636, sjkim@kisti.re.kr
- 사용자 교육 : 이홍석, 042) 869-0579, hsyi@kisti.re.kr
- 홈페이지 : http://www.ksc.re.kr