슈퍼컴퓨터 사용자 안내서

KISTI 슈퍼컴퓨팅센터

목 차

1. IBM ····· 1
1.1 도입 개요 ····· 1
1.2 시스템 사양 ····· 1
1.3. 노드 구성 ····· 2
1.4. 하드웨어 특징 ····· 3
1.5. 시스템 접속 ····· 5
1.6. 패스워드 변경 ····· 5
1.7. 쉘 변경 ····· 6
1.8. 프로세스 자원 제한 ····· 7
1.9. 보안 상 제약 사항 ····· 7
1.10. 파일 시스템 ····· 8
1.11. 사용자 요금 정책 ····· 13
1.12. HPSS를 사용한 대용량 데이터 저장 관리 ····· 15
1.13. 소프트웨어 설치 내역 ····· 22
1.14. 프로그램 컴파일 방법 ····· 24
1.15. ESSL/PESSL[수학 라이브러리] 사용법 ····· 28
1.16. POE를 통한 병렬 프로그램 실행 ····· 29
1.17. LoadLeveler를 통한 배치 작업 실행 ····· 37
1.18. Checkpoint/Restart ····· 57

2. NEC ····· 59
2.1. 도입 개요 ····· 59
2.2. 시스템 구성 ····· 59
2.3. 사용자 환경 ····· 61

2.4. 파일시스템 ····· 63
2.5. HPSS ····· 65
2.6. Cross compiler ····· 66
2.7. 프로그래밍 ····· 67
2.8. 배치작업 환경 ····· 73
2.9. 요금 정책 ····· 85

3. HAMEL ····· 86
3.1. 시스템 개요 ····· 86
3.2. 자원할당 및 시스템 억세스 ····· 91
3.3. 사용자 계정관리 ····· 93
3.4. 프로그래밍 ····· 95
3.5. 작업관리 ····· 101
3.6. 대용량 스토리지(HPSS) ····· 113
3.7. 사용자 자원관리 ····· 114

부 록 ····· 117

1. IBM FAQ ····· 118
2. NEC ····· 125

표 차례

<표 1-1> IBM 시스템 사양 ····· 1
<표 1-2> 노드 구성 리스트 ····· 2
<표 1-3> POWER4 프로세서 구조 ····· 3
<표 1-4> 방화벽 접속 제한 리스트 ····· 7
<표 1-5> 홈디렉토리 구성 ····· 9
<표 1-6> 스크래치 디렉토리 구성 ····· 9
<표 1-7> 구좌당 자원제공 내역 ····· 13
<표 1-8> Machine Charging Factor ····· 14
<표 1-9> 사용자 지원 ····· 15
<표 1-10> 소프트웨어 설치 내역 ····· 22
<표 1-11> 응용 소프트웨어 지원 내역 ····· 23
<표 1-12> 컴파일러 명령어 리스트 ····· 24
<표 1-13> -qarch 또는 -qtune의 옵션값 ····· 26
<표 1-14> 클래스별 구성 내역 ····· 40
<표 1-15> p690 시스템 클래스별 가용 CPU 수 ····· 40
<표 1-16> 사용자 별 구성 내역 ····· 41
<표 1-17> Batch 작업관리 시스템 명령 비교 ····· 56
<표 1-18> LoadLeveler/NQS/LSF 변환 ····· 56
<표 2-1> NEC SX-5/8B 시스템 하드웨어 내역 ····· 60
<표 2-2> NEC SX-5/8B 시스템 소프트웨어 내역 ····· 60
<표 2-3> SX-5 NQS queue table ····· 76
<표 2-4> SX-6a NQS queue table ····· 76
<표 2-5> SX-6b NQS queue table ····· 77
<표 3-1> 하멜 시스템의 큐 ····· 102

그림 차례

<그림 1-1> IBM 시스템 구성도 ····· 2
<그림 1-2> POWER4 프로세서 구조 ····· 3
<그림 1-3> POWER4 시스템 기본 구성 단위 ····· 4
<그림 1-4> p690 시스템 백플레인 구성도 ····· 4
<그림 1-5> IBM 시스템 스크래치 파일 시스템 구성도 ····· 10
<그림 1-6> POE를 통한 병렬프로그램 수행 흐름도 ····· 30
<그림 1-7> LoadLeveler 작업 실행 흐름도 ····· 38
<그림 1-8> xloadl 실행화면 ····· 39
<그림 2-1> NEC Network Configuration ····· 61
<그림 2-2> NEC System Configuration ····· 65
<그림 2-3> NQS 구조 ····· 74
<그림 3-1> HAMEL 시스템 네트워크 구성도 ····· 89

1. IBM

1.1 도입 개요

IBM 시스템은 Cray 2S와 Cray C90에 이어 세 번째로 도입되는 슈퍼컴퓨터 3호기로써 2차에 걸쳐 분할 도입되었으며, 1차 시스템[p690 4대]은 2002년 1월경에, 그리고 2차 시스템[p690 17대]은 2003년 7월경에 도입 완료되었다.

1.2 시스템 사양

<표 1-1>에서 보는 바와 같이 1차 및 2차로 분할 도입된 p690 시스템의 전체 이론 성능은 4,364.8 GFLOPS이며, 특히 p690은 단위 컴퓨팅 시스템 당 169GB-256GB의 대용량 메모리를 장착하고 있다. 또한, Raid 구성 후 가용 외장 디스크 용량은 98TB이며, 테이프 백업 장치는 381TB에 이른다.

<표 1-1> IBM 시스템 사양

구 분		1차 시스템	2차 시스템
시스템명		IBM p690	IBM p690+
수 량		4대	17대
CPU	프로세서명	POWER4 [1.3GHz]	POWER4 [1.7GHz]
	전체 이론 최고성능 (GFLOPS)	665.6 [166.4/노드]	3,699.2 [217.6/노드]
	전체 개수(개)	128 [32/노드]	544 [32/노드]
주기억장치 용량(GB)		688 [208×1, 160×3]	3,712 [192×10, 256×7]
내장 디스크 용량		2,329.6 [36.4×16개/노드]	19,964.8 [73.4×16개/노드]
외장 디스크 용량(TB)*		98TB	
테이프 백업 장치		● 3584 LTO 라이브러리 2대 ● 드라이브수: 52개 ● 테이프용량 : 381TB	

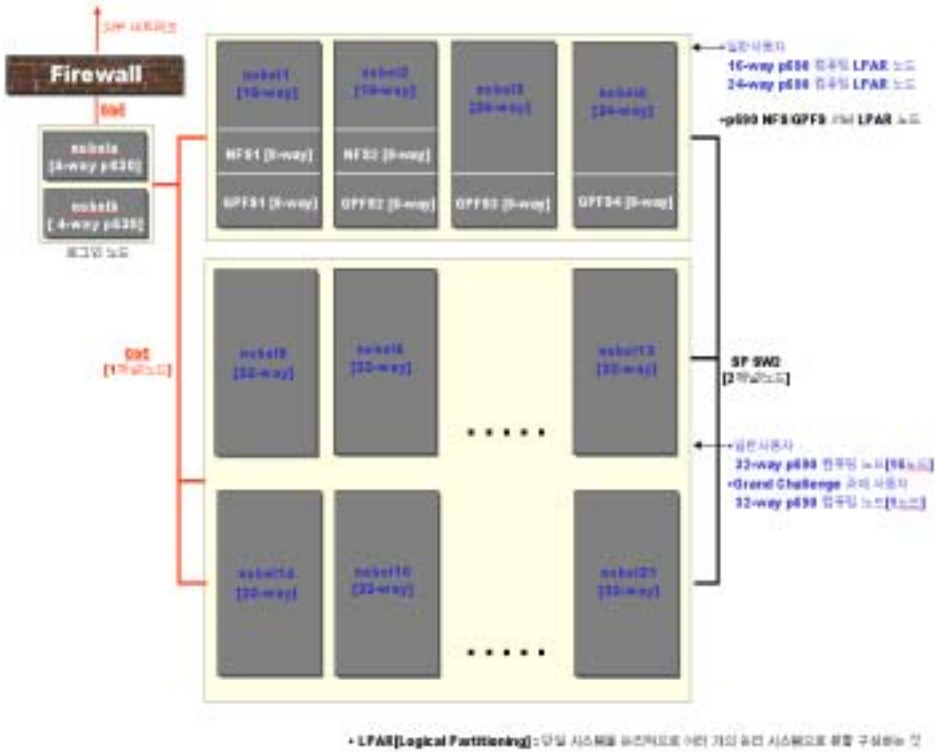
※ *) : Raid 구성 후 외장 디스크 용량임.

1.3. 노드 구성

<표 1-2> 노드 구성 리스트

구분	시스템 구성	노드 이름	설 명
로그인 노드	● p630 2노드	● nobela[nobel] ● nobelb	- 사용자 로그인 - 프로그램 편집/컴파일 - 배치 작업 제출
컴퓨팅 노드	● 16-way p690 LPAR 2노드 ● 24-way p690 LPAR 2노드 ● 32-way p690 16노드	● nobel1-20	- 유료 사용자 노드
	● 32-way p690 1노드	● nobel21	- Grand Challenge 과제 노드*

*) Grand Challenge 과제 노드 수는 과제의 규모에 따라서 유동적으로 변화될 수 있음.



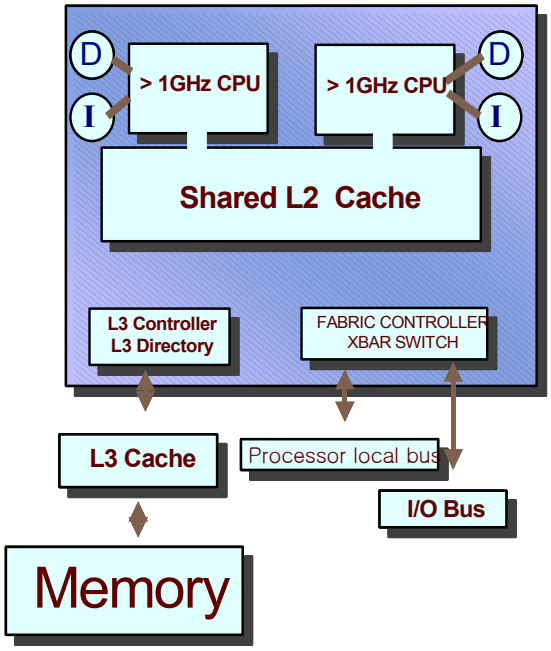
<그림 1-1> IBM 시스템 구성도

1.4. 하드웨어 특징

1.4.1 POWER4 프로세서 구조

<표 1-3> POWER4 프로세서 구조

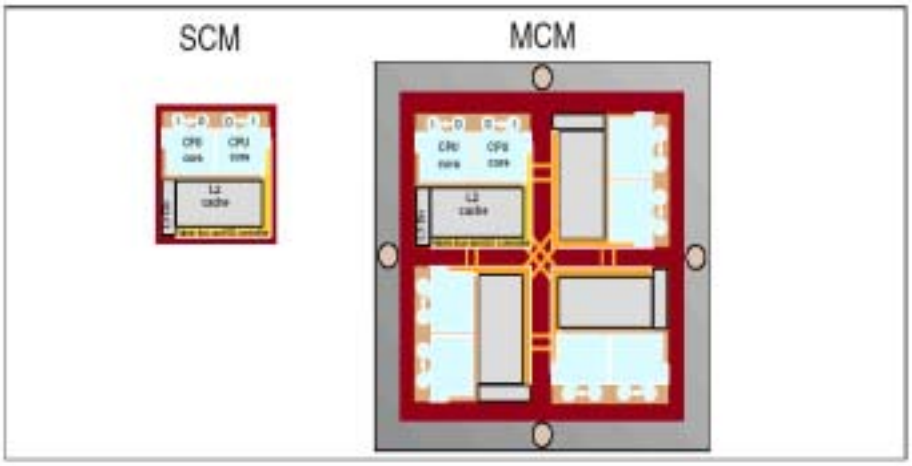
구분 \ 시스템 명		p690
-Clock speed		1.3GHz[1차]/1.7GHz[2차]
-단일 칩 당 내장 CPU 수		2개
-Cache 구조	+L1 Cache	data: 32KB, instr: 64KB
	+L2 Cache	1.44MB over 100GB/s
	+L3 Cache	32MB over 10GB/s
-기타		NUMA enabled



<그림 1-2> POWER4 프로세서 구조

1.4.2 POWER4 시스템 기본 구성 단위

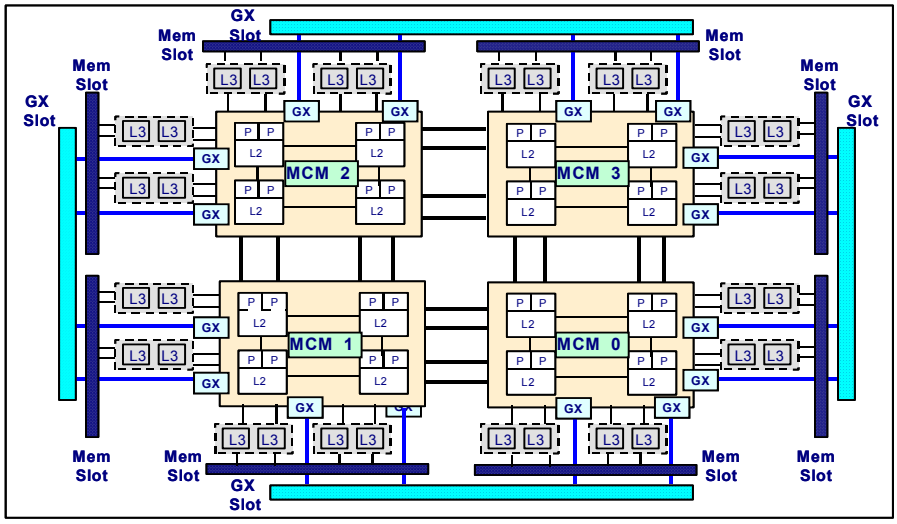
p690은 8개 CPU[4개 칩]로 구성된 Multi-Chip Module[MCM]이 기본 구성 단위이다. <그림 1-3> 는 본 슈퍼컴퓨팅센터에 도입된 p690 시스템에 장착된 프로세서 모듈을 보여주고 있다.



<그림 1-3> POWER4 시스템 기본 구성 단위

1.4.3 p690 시스템 백플레인 구성도

<그림 1-4>은 p690 시스템의 백플레인 구성도를 보여주고 있다.



<그림 1-4> p690 시스템 백플레인 구성도

1.5. 시스템 접속

로그인 노드는 2대로 구성되어 있으며, 사용자가 두 노드 중의 하나를 선택하여 다음 예제와 같이 접속할 수 있다. 현재 보안상 안전한 통신을 이루기 위해서 SecureShell 터미널 접속만을 지원하며, 기존 telnet 접속은 허용되지 않고 있습니다.

1.5.1 SSH 터미널 접속 방법

□ 로그인 노드 1)

```
ssh -l User_ID nobela.ksc.re.kr
                or
ssh -l User_ID nobel.ksc.re.kr
                or
ssh -l User_ID 150.183.5.101
```

□ 로그인 노드 2)

```
ssh -l User_ID nobelb.ksc.re.kr
                or
ssh -l User_ID 150.183.5.102
```

Note

- IBM 시스템 관련 공지 사항 확인
  #> cat /etc/motd
- SecureShell[SSH] 사용자 지침서
  Cyber HelpDesk 홈페이지[http://helpdesk.ksc.re.kr]에서 Technote 게시판의 "SSH 사용자 지침서"를 참조하기 바란다.
- 윈도우용 SSH 클라이언트 프로그램 다운로드
  Cyber HelpDesk 홈페이지[http://helpdesk.ksc.re.kr]에서 Technote 게시판의 "SSH Secure Shell Client" 혹은 "putty" 프로그램을 다운로드 하여 설치한다.

1.6. 패스워드 변경

사용자 패스워드를 변경하기 위해서는 passwd 명령을 사용하여야 한다.

```
nobel:/system/super/wjnadia> passwd
Changing NIS password for wjnadia
Old NIS password:
wjnadia's New password:
Enter the new password again:
nobel:/system/super/wjnadia>
```

1.7. 쉘 변경

"yppasswd -s" 명령을 사용하면 된다.

```
nobela:/system/super/wjnadia> yppasswd -s
yppasswd: 1831-167  Changing NIS shell for wjnadia
Old NIS password:
yppasswd: 1831-184 Current available shells:
                /bin/sh
                /bin/bsh
                /bin/csh
                /bin/ksh
                /bin/tsh
                /bin/ksh93
                /usr/bin/sh
                /usr/bin/bsh
                /usr/bin/csh
                /usr/bin/ksh
                /usr/bin/tsh
                /usr/bin/ksh93
                /usr/sbin/sliplogin
                /usr/sbin/uucp/uucico
                /usr/sbin/snappd
                /usr/local/bin/bash
yppasswd: 1831-185 wjnadia's current login shell:
                /bin/ksh
yppasswd: 1831-182 Change (yes) or (no)? >yes
yppasswd: 1831-186 To?> /bin/csh
yppasswd: 1831-165 NIS login shell changed on backup1
nobela:/system/super/wjnadia>
```

1.8. 프로세스 자원 제한

IBM 시스템에서 배치 작업을 제외한 인터랙티브 작업의 사용자 프로세스에 대해 아래와 같이 시스템 자원 항목에 대해 제한을 가하고 있다.

-time : CPU 사용 시간 10분

사용자에게 허용된 프로세스 자원은 아래의 예와 같이 ulimit 명령으로 확인할 수 있다.

```
$ ulimit -Ha
time(seconds)          600
file(blocks)           unlimited
data(kbytes)           unlimited
stack(kbytes)          unlimited
memory(kbytes)         unlimited
coredump(blocks)       2097151
nofiles(descriptors)   2000
```

1.9. 보안 상 제약 사항

최근 급증하고 있는 보안 침해사고 예방을 위하여 다음과 같은 보안정책이 적용되고 있사오니, 필독하시어 시스템 사용에 유의하시기 바랍니다. <표 1-4>은 슈퍼컴퓨터 접속 시 방화벽에서 허용되는 서비스를 설명하고 있다.

<표 1-4> 방화벽 접속 제한 리스트

접속 방향	허용 되는 서비스
사용자 시스템 → IBM 시스템[로그인노드]	SSH, FTP, X-Window
IBM 시스템 → 사용자 시스템	E-Mail

1.9.1 패스워드 정책

□ 패스워드 최소 8자 이상 입력

□ 패스워드 만료기간 : 2개월

□ 패스워드 변경 시 이전 패스워드와 비교 최소 2자 이상은 달라야함.
□ 10회 이상 로그인 실패 시 계정 자동 잠김 : 계정이 잠기면 관리자에게 연락 요망.

1.9.2 접속 정책

□ SSH 터미널 접속만 허용: 기존 telnet 접속은 허용하지 않음.
□ 사용자 시스템에서는 로그인 노드만 직접 접속 가능
[기타 컴퓨팅 노트는 로그인 노드에 로그인 후 접속 가능]
□ 로그인 노드[nobela/b]를 제외하고 X-GUI 사용 불가[텍스트 터미널 환경만 지원]

1.9.3 기타 사용자 권고 사항

□ 사용자 홈디렉토리의 .netrc, .rhosts, .sh_history 파일의 퍼미션 600 유지
예) chmod 600 .netrc .rhosts .sh_history
□ 홈디렉토리 및 스크래치 작업 디렉토리의 퍼미션 : 750 유지
예) chmod -R 750 /gpfs1/job_dir <- 하위 디렉토리까지 퍼미션 조정[-R 옵션]

1.10. 파일 시스템

1.10.1 파일 시스템 구성

홈디렉토리는 사용자가 시스템에 접속하여 로그인 하면 처음 위치하게 되는 디렉토리로써 사용자 프로그램 및 데이터 저장 용도로만 사용해야 한다. 또한 1 계정 당 일반 사용자는 3GB, 학생 사용자는 300MB를 제공하고 있으며, 이 제한 용량을 초과하면 더 이상 데이터를 저장할 수 없다.

그리고, 홈 디렉토리 용량을 초과하는 데이터 및 스크래치 디렉토리의 작업 결과 데이터를 저장하고자 하는 경우에는, 대용량 데이터 저장관리 솔루션인 HPSS를 사용하여 사용자 당 수백 GB 이상의 데이터를 저장하고 recall 하여 활용할 수 있다. [관련 내용 및 사용법은 5. 대용량 데이터 저장관리 부분 참조]

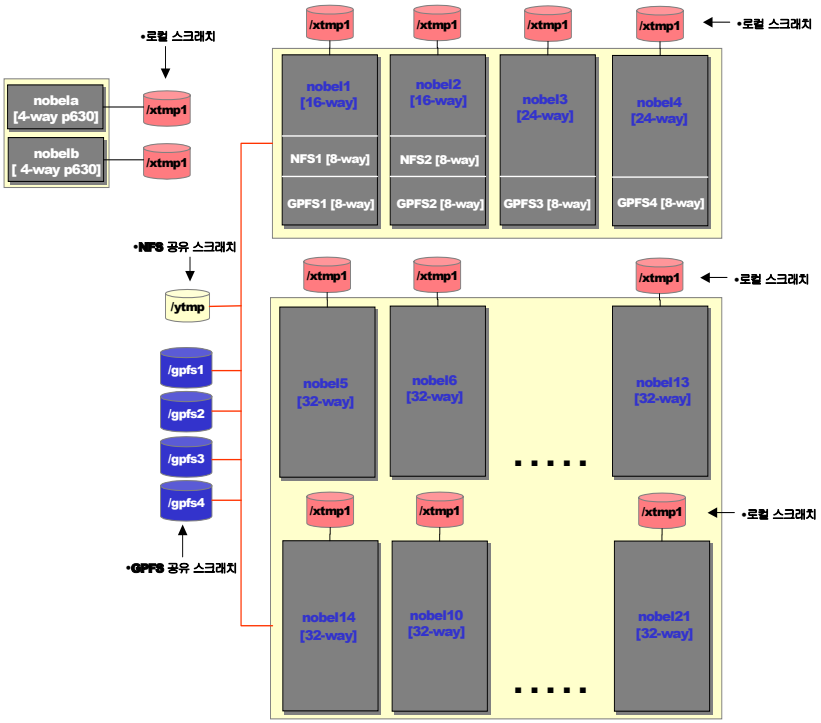
<표 1-5> 홈디렉토리 구성

구분	파일 시스템명	백업 유무	삭제 정책	용량 제한
홈디렉토리	/edun, /inst, /inst2	매일 백업함	계정 만료시 삭제	-일반 사용자:3GB/계정 -학생사용자:300MB/계정

특히, 계산 작업 시에는 반드시 스크래치 디렉토리를 사용해야 한다. 스크래치 디렉토리는 각 노드에 직접 연결된 로컬 스크래치[/xtmp1]와 공유 스크래치[/gpfs1-4, /ytmp]가 사용자 작업 수행 공간으로 제공된다. 일반적으로 I/O 성능을 고려하여 로컬 스크래치를 사용할 것을 권장하며, 특별히 대용량 파일에 대한 순차적인 I/O가 집중적으로 발생되는 작업은 /gfps1-4 공유 스크래치를 사용하는 것이 좋다. 단, 스크래치 디렉토리에서 작업 종료 후 최종 변경일로부터 1일 이상 경과된 파일들은 자동 삭제된다.

<표 1-6> 스크래치 디렉토리 구성

구분	파일 시스템명	용량	백업 유무	삭제 정책	용량 제한
로컬 스크래치	/xtmp1	1.2TB	백업하지 않음	작업 종료 후 최종변경일로부터 1일 이상 경과한 파일 삭제	제한없음
공유 스크래치	/ytmp[NFS]	400GB			20GB/사용자ID
	/gpfs1 /gpfs2 /gpfs3 /gpfs4	각 6.4TB			500GB/사용자ID 각 파일시스템별



<그림 1-5> IBM 시스템 스크래치 파일 시스템 구성도

※ /xtmp1 노드별 스크래치 디렉토리를 사용한 LoadLeveler 작업 실행 방법

/xtmp1 스크래치 디렉토리는 노드별 1.2TB 크기로 제공되는 I/O 성능이 우수한 대용량 스크래치 디렉토리이다. 이러한 /xtmp1 스크래치 디렉토리를 사용하여 배치 작업을 실행하기 위해서는 LoadLeveler Job Command File에서 작업 명령 실행 전에 작업 파일들을 홈디렉토리에서 실행 노드의 /xtmp1 작업 디렉토리로 복사하고, 작업 종료 후 결과 파일들을 실행노드의 /xtmp1 작업 디렉토리에서 홈디렉토리로 복사하는 스크립트를 작성해야 한다. 예제 파일은 /system/samples/
LoadL/xtmp1.cmd이다.

```
#!/bin/ksh
#@ job_type = serial
#@ class = normal
#@ resources = ConsumableCpus(1) ConsumableMemory(1gb)
#@ wall_clock_limit = 2:00:00
#@ queue
 rm -rf /xtmp1/wjnadia    # 작업 실행 노드의 /xtmp1 디렉토리에 기존 작업 디렉토리 삭제
 mkdir /xtmp1/wjnadia     # 실행노드의 /xtmp1 디렉토리에 새로운 작업 디렉토리 생성
 cp -f /edun/wjnadia/* /xtmp1/wjnadia # 작업 실행에 필요한 파일들을 실행노드의 /xtmp1
                                      # 디렉토리로 복사
 cd /xtmp1/wjnadia              # 실행 노드의 /xtmp1 작업 디렉토리로 작업 경로 변경
   ./long_serial_program
 cp -f /xtmp1/wjnadia/* /edun/wjnadia/output/  # 작업 결과 파일을 홈 디렉토리로 복사
 rm -rf /xtmp1/wjnadia          # /xtmp1의 작업 디렉토리 삭제
```

1.10.2 파일 시스템 구성 정보 확인

각 시스템에서 사용 가능한 파일시스템의 구성 정보와 사용 가능량은 아래와 같이 "df -k" 명령으로 확인할 수 있다.

```
$ df -k
Filesystem        1024-blocks      Free    %Used    Iused  %Iused   Mounted on
/dev/hd4            524288       505892    4%       2943     3%      /
/dev/hd2           4227072      1003660   77%      68420    23%      /usr
/dev/hd9var        1081344      1050024    3%        689     1%      /var
/dev/hd3           2097152      1132848   46%        394     1%      /tmp
/dev/hd1             32768        32400    2%          5     1%      /home
/proc                  -            -      -          -      -       /proc
/dev/hd10opt         32768        26040   21%        290     5%      /opt
/dev/xtmplv      419430400    412118496    2%        293     1%      /xtmp1
/dev/ytmplv     1048576000   1036464268    2%      27998     1%      /ytmp
/dev/acctwk        4063232      3923812    4%        181     1%      /var/adm/acct
backup1:/LoadL     3145728      3042780    4%        187     1%      /LoadL
backup1:/edun   1073741820   1049036144    3%       1332     1%      /edun
backup1:/account 104857600    104060224    1%       1257     1%      /account
backup1:/applic  104857600    104645092    1%         17     1%      /applic
backup2:/inst    524288000    522199388    1%         21     1%      /inst
backup2:/system  209715200    209282524    1%        113     1%      /system
backup2:/super   209715200    174062852   18%      75141     3%      /super
```

1.10.3 홈디렉토리 할당량 및 사용량 확인

현재 제공되는 사용자 홈 디렉토리는 NFS 형태로 제공되고 있어, 실제 사용자가 자신의 홈 디렉토리 할당량 및 사용량을 확인하기 위해서는 아래와 같은 명령을 사용해야 한다. 또한 사용자가 염두에 두어야 할 점은 홈디렉토리 용량은 사용자 ID가 아니라, 이러한 사용자 ID의 모임인 그룹 ID[계정번호] 단위로 제한을 받게 된다는 것이다.

- 홈디렉토리가 /edun 디렉토리 아래 존재하는 경우

```
/edun/i002eos > rsh nfs2 quota -g sf1200
Disk quotas for group sf1200 (gid 2080):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
        /edun  1872292 3000000 3072000               148       0       0
/edun/i002eos >
```

- 홈디렉토리가 /inst 디렉토리 아래 존재하는 경우

```
/inst/r004hwk > rsh nfs1 quota -g sf1290
Disk quotas for group sf1290 (gid 2090):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
        /inst    1872292 3000000 3072000               148       0       0
/inst/r004hwk >
```

위에서 blocks는 KByte 단위로 표시되는 그룹 ID sf1200 사용자의 홈디렉토리 사용량을 표시하며, 바로 limit이 그룹 ID sf1200 사용자에게 할당된 KByte 단위의 제한 용량을 의미한다.

홈 디렉토리가 /edun 파일시스템에 있는 사용자는 rsh nfs2 quota 명령을 사용해야 하고, /inst, /system 파일시스템에 있는 사용자는 rsh nfs1 quota 명령을 사용해야 한다.

위와 같이 rsh 명령을 사용하기 위해서는 사용자 홈 디렉토리의 .rhosts 파일에 현재 로그인 노드의 호스트 이름을 추가해야만 한다.

```
/system/super/wjnadia> cat >> .rhosts
nobela
nobelb
/system/super/wjnadia>
```

또한 사용자 별로 할당된 홈 디렉토리 내에 사용량을 파악하는 다른 방법으로 du -k 명령을 사용할 수 있다.
명령의 실행 결과 중 마지막 라인 “1872596“은 KByte 단위의 사용량을 나타내고 있다.

```
/system/super/wjnadia > du -k ~wjnadia
817468   data
  312       errpt
  12        mbox
  8         notice
  1052188  pilot
  4         pw
  817468   ./data
  1872596  .
  /system/super/wjnadia >
```

1.11. 사용자 요금 정책

1.11.1 구좌당 자원제공 내역

- 사용자가 isam을 통하여 확인할 수 있는 시간은 CPU Time에 3가지 Factor를 곱하여 환산한 SRU Time이다.
- CPU Time에 대해서만 요금을 부과하고 디스크와 테이프는 무료로 제공한다.

<표 1-7> 구좌당 자원제공 내역

구 분	회원구분	사용료	SRU Time	CPU Time	디스크 할당량 (홈디렉토리)	백업 테이프 할당량
p690	일반회원	1,000,000원/1구좌	100시간 [360,000초]	1,000시간 [3,600,000초]	3GB	200GB
	학생회원	100,000원/1구좌	10시간 [36,000초]	100시간 [360,000초]	300MB	20GB

※ 사용자 작업 수행 시간에 따른 CPU 사용 현황은 로그인 시스템[nobela/nobelb]에서 "isam" 명령을 수행하고, 안내에 따라 메뉴를 선택하는 것으로 손쉽게 파악할 수 있다.

1.11.2 사용료

사용료의 기본 서비스 단위는 SRU(Service Resource Units) Time으로서 다음과 같은 식으로 계산됨.

[단위: 초]

SRU Time=CPU Time×Machine Charging Factor×Service Charging Factor*×Industry Factor

- Time : Total CPU Time
- Machine Charging Factor

시스템의 구입가격, 성능, 운영비용 등을 고려한 수치로서 다음 <표 1-8>와 같이 NEC SX-5 시스템이 1이며, HP SMP는 0.05, IBM p690은 0.2로 책정됨.

<표 1-8> Machine Charging Factor

구분	NEC SX-5/8B	HP SMP	IBM		HAMEL
			p630 [로그인 시스템]	p690 [컴퓨팅 시스템]	
값	1	0.05	0.05	0.1	0.005

- Service Charging Factor

  사용자 요구수준(Service Level)에 따라 부과되는 수치이나, 현재는 적용되지 않고 있음.
- Industry Factor

  산업체 사용자의 경우는 2, 비 산업체 사용자의 경우는 1임.
  (단, 벤처기업이나 중소기업의 경우는 1)

1.11.3 사용자 지원

사용자는 시스템 사용 중 질의 사항이 발생하면 무료 전화, 웹을 통한 사이버 Helpdesk 및 E-Mail을 통해서 HelpDesk의 지원을 받을 수 있으며, 홈페이지의 게시판을 활용하여 질의할 수도 있다. 또한 사용자는 수시로 슈퍼컴퓨팅센터 홈페이지를 방문하여 공지사항 및 업그레이드된 지침서 등의 자료를 확인하는 것이 좋다.

<표 1-9> 사용자 지원

구분		내 용
HelpDesk	무료 전화	080-041-1991
	웹 사이버 HelpDesk	http://helpdesk.ksc.re.kr
	E-Mail	helpdesk@ksc.re.kr
홈페이지		http://www.ksc.re.kr

1.12. HPSS를 사용한 대용량 데이터 저장 관리

1.12.1 HPSS(High Performance Storage System)의 개요

HPSS는 대규모 스토리지 환경을 위한 계층적 스토리지 관리 및 서비스 제공을 위한 소프트웨어이며 수백 테라 바이트에서 수 페타 바이트까지의 데이터를 처리할 수 있다. HPSS는 고성능 컴퓨터나 디스크, 테이프 라이브러리들 사이에 매우 큰 데이터 오브젝트들을 이동시키도록 고안되었으며 처리속도는 단지 컴퓨터나 네트워크, 스토리지 장치 등의 최대 속도에 의해서만 제한된다. HPSS는 초당 수백 메가바이트의 속도로 멀티 network-connected 디스크들로부터 병렬 데이터 전송 처리할 수 있다. HPSS는 미국 국립 슈퍼컴퓨터 센터들과 산업체들 간의 협력에서 나온 결과물이며 IBM에 의해 상업적으로 배포되고 있다. HPSS의 핵심적인 목표는 오늘날의 어떤 상업적인 저장 시스템들 보다 훨씬 빠르게 병렬 또는 클러스터링 된 시스템과 저장장치 사이에 대용량 파일들을 신속히 이동시키는 것이다. 현재 KISTI는 5대로 구성된 HPSS 서버 시스템을 운영 중에 있으며 3.6TB의 디스크 캐시와 60TB이상의 테이프 스토리지를 제공할 수 있다.

1.12.2 HPSS 기본 사용법

HPSS를 이용하여 대용량의 데이터를 저장하고자 하는 사용자는 다음의 순서에 따라 계정을 신청한 후에 적절한 명령어를 사용하여 HPSS 서비스를 이용할 수 있다.

□ STEP 1. HPSS 계정 신청

HPSS 서비스를 이용하려면 HPSS 시스템에 access 할 수 있는 계정이 필요하다. IBM 시스템의 계정을 가지고 있는 사용자는 HPSS 계정을 다음의 e-메일로 신청할 수 있다.

jangoq@kisti.re.kr

□ STEP 2. HPSS 로그인

가. IBM 1차 시스템의 계정을 가지고 있는 사용자는 각 시스템에 로그인 한 후 다음 명령어를 수행한다.

```
$ pftp_client core 4021
```

나. 다음과 같은 메시지를 볼 수 있다

```
Parallel block size set to 8388608.
PdataSockBufSize reset equal or below sb_max (1048576)
Connected to core.hpcnet.ne.kr.
220-
220-**************************************************
220-*                                                *
220-* This is the HPSS 4.3 system running at KISTI  *
220-*                                                *
220-**************************************************
220-
220 core FTP server (HPSS 4.3 PFTPD V1.1.1 Mon Dec 3 14:30:48 KORST 2001) ready.
Name (core:ykchoi):
```

다. "Name (core:ykchoi) : " 에 HPSS administrator로부터 받은 ID를 입력한다. 엔터를 친 후 다시 암호 입력 메시지가 나오면 암호를 입력한다.

```
Name (core:ykchoi): ykchoi98
331 Password required for /.../kisti_cell.hpcnet.ne.kr/ykchoi98.
Password:
```

라. 정확히 HPSS 계정 ID와 패스워드를 입력하면 다음의 메시지를 볼 수 있다.

```
230 User /.../kisti_cell.hpcnet.ne.kr/ykchoi98 logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
**** NOTE: Server supports Parallel Features     ****
****         Auto-Parallel Substitution Enabled.  ****
**** NOTE: Protocol set to PDATA_AND_MOVER       ****
Multinode is Disabled.
ftp>
```

마. 230으로 시작되는 첫 번째 줄을 보게 되면 자신의 HPSS 계정 ID를 볼 수

있다. (예제의 경우 "ykchoi98").

바. pwd 명령어를 이용하면 현재 자신의 디렉토리를 볼 수 있다.

```
ftp> pwd
257 "/home/ykchoi98" is current directory.
ftp>
```

작업 디렉토리는 사용자 계정에 따라 다를 수 있다. 사용자는 작업 영역에 따라"home/ykchoi98"의 서브디렉토리를 만들 수 있다(mkdir 명령어 사용). 예를 들어 "test"라는 서브디렉토리를 만들고자 할 때는 HPSS 내에서 작업 공간의 이동은 "cd dir_name" 명령어를 이용한다. ftp> 프롬프트 상태에서 로컬의 디렉토리를 이동하려면 "lcd dir_name" 명령어를 이용한다.

```
ftp> mkdir test
257 MKD command successful.
ftp> cd test
250 CWD command successful.
ftp> pwd
257 "/home/ykchoi98/test" is current directory.
ftp>
```

□ STEP 3. Parallel file store - pput

가. *pput* 는 로컬로부터 HPSS로 파일을 전송하는 명령어이다.

```
ftp> pput file_name
```

나. 파일을 전송하기 전에 자신의 로컬 디렉토리에 있는 파일을 확인해 본다. *!ls* 명령어를 이용한다

```
ftp> !ls
300.out                pput.ksh                babo.txt
pput.out               frontsmp_pput_1.txt      pput_final.txt
mksparse               pput_frontsmp.new.txt    pget.ksh
ftp>
```

다. HPSS로 전송할 파일을 정했으면 pput 명령어를 사용한다. 여기서는

300.out 파일을 전송해 보기로 한다.

```
ftp> pput 300.out
local: 300.out remote: 300.out
200 Command Complete.
200 PORT command successful.
150 Opening BINARY mode data connection for 300.out.
226 Transfer complete.
342635 bytes sent in 0.08 seconds (4.29 Mbytes/s)
ftp>
```

라. *ls* 명령어를 사용하면 HPSS로 파일이 전송된 것을 확인할 수 있다.

```
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
300.out
medium
medium_Merkava
test
226 Transfer complete.
39 bytes received in 0.01 seconds (4.88 Kbytes/s)
ftp>
```

마. 여러 개의 파일을 전송할 때는 *mpput* 명령어를 사용한다. 주의할 점은 interactive mode를 off 하기 위하여 prompt 명령어를 *mpput*을 사용하기 전에 사용한다. 이 모드가 on 상태를 유지하게 되면 파일이 한번씩 전송될 때마다 확인 메시지가 나와서 사용자는 계속 y 키를 눌러야 한다.

```
ftp> prompt
Interactive mode off.
ftp> mpput file_names1 file_name2 file_nameN
```

파일의 개수가 많지 않을 경우 파일의 이름을 위와 같이 나열할 수 있으나 와일드 카드를 이용하면 보다 편리하다

```
ftp> mpput *.dat
```

□ STEP 4. Parallel file retrieval - pget

가. pget 는 HPSS로부터 로컬로 파일을 전송하는 명령어이다.

```
ftp> pget file_name
```

나. 파일을 전송하기 전에 자신의 HPSS 디렉토리에 있는 파일을 확인해 본다. *ls* 명령어를 이용한다

```
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
300.out
226 Transfer complete.
9 bytes received in 0.01 seconds (1.29 Kbytes/s)
ftp>
```

다. 로컬로 전송할 파일을 정했으면 pget 명령어를 사용한다. 여기서는 300.out 파일을 전송해 보기로 한다.

```
ftp> pget 300.out
local: 300.out remote: 300.out
200 PORT command successful.
150 Opening BINARY mode data connection for 300.out (342635 bytes).
226 Transfer complete.
342635 bytes received in 1.62 seconds (205.91 Kbytes/s)
ftp>
```

라. !ls 명령어를 사용하면 자신의 로컬 디렉토리에 파일이 전송된 것을 확인

할 수 있다.

마. 여러 개의 파일을 전송할 때는 *mpget* 명령어를 사용한다.

```
ftp> prompt
Interactive mode off.
ftp> mpget file_names
```

□ STEP 5. 기타 명령어 - bye, del, mdel

가. *pftp_client* 종료할 때는 bye 명령어를 이용한다.

```
ftp> bye
421 Timeout (900 seconds): closing control connection.
frontsmp [/system/super/ykchoi] $
```

나. HPSS에 전송된 파일을 삭제할 경우 *del* 명령어를 사용한다.

```
ftp> del pput.txt
250 DELE command successful.
ftp>
```

다. 여러 개의 파일을 한꺼번에 삭제할 경우 mdel을 사용한다. interactive mode를 off 하기 위하여 *prompt* 명령어를 *mdel*을 사용하기 전에 사용한다. 이 모드가 on 상태를 유지하게 되면 파일이 한번씩 전송될 때마다 확인 메시지가 나와서 사용자는 계속 y 키를 눌러야 한다.

```
ftp> mdel *.txt
mdel babo.txt? y
250 DELE command successful.
mdel frontsmp_pput_1.txt? y
250 DELE command successful.
mdel pget_final.txt? y
250 DELE command successful.
mdel pget_frontsmp.txt? y
250 DELE command successful.
mdel put_1030.txt?
```

< prompt 사용 후 >

```
ftp> prompt
Interactive mode off.
ftp> mdel *.txt
250 DELE command successful.
250 DELE command successful.
250 DELE command successful.
250 DELE command successful.
250 DELE command successful.
250 DELE command successful.
ftp>
```

1.12.3 배치작업에서의 HPSS 사용

LoadLeveler를 사용한 배치 작업에서 HPSS를 사용하기 위해서는 Job Command File 내에 간단한 코드를 삽입하는 것으로 간편하게 이용할 수 있으며, 주요 코드는 다음과 같다

```
echo "pput filename" | pftp_client core 4021
또는
echo "pget filename" | pftp_client core 4021
```

또한 사용자는 자신의 홈디렉토리에 다음과 같이 .netrc 파일을 생성하여 HPSS 서버에 자동 로그인 하도록 설정해야 한다. 다음과 같이 HPSS 서버 이름 및 HPSS 계정 신청을 통해서 얻은 HPSS_ID, HPSS_PASSWORD를 입력해야 한다. 또한, .netrc 파일의 퍼미션을 600으로 변경하고 로그아웃 후 다시 로그인하면 자동 로그인이 가능하게 된다.

```
nobel : /edun/wjnadia > cat .netrc
machine core login HPSS_ID password HPSS_PASSWORD
nobel : /edun/wjnadia > chmod 600 .netrc
nobel : /edun/wjnadia > exit
현재 계정으로 다시 로그인
```

다음 예제는 “p41c.tar"을 HPSS 서버에서 다운로드하여 retrieve 후 작업을 실행하고, 작업 결과 파일(”p41c_out.tar")을 다시 HPSS 서버로 자동 업로드하는 스크립트이다.

```
#!/bin/ksh
#@ job_type = serial
#@ notification = always
#@ notify_user =  wjnadia@kisti.re.kr
#@ class = normal
#@ resources = ConsumableCpus(1) ConsumableMemory(1gb)
#@ wall_clock_limit = 10:00:00
#@ queue
rm -rf /xtmp1/wjnadia        # 기존에 있던 자신의 작업 디렉토리(/wjnadia) 삭제
mkdir /xtmp1/wjnadia         # 새로운 작업 디렉토리 생성
cd /xtmp1/wjnadia            # 새로운 작업 디렉토리로 이동
echo "pget p41c.tar"|pftp_client core 4021   # HPSS 서버에 접속해 작업 파일을 다운로드 함
tar xvf p41c.tar             # 가져온 작업 파일의 압축을 푼다
./p41c > p41c.out 2>&1          # 가져온 작업 파일을 실행시켜 output 생성
tar cvf p41c_out.tar p41c.out       # output 파일을 압축한다
echo "pput p41c_out.tar" |pftp_client core 4021  # 압축한 파일을 HPSS 서버로 업로드 함
rm -rf /xtmp1/wjnadia          # 작업 디렉토리 삭제
```

1.13. 소프트웨어 설치 내역

1.13.1 기본 소프트웨어

<표 1-10> 소프트웨어 설치 내역

구분	소프트웨어 이름 및 버전
운영체제	- AIX5L V5.1 ML08
클러스터 환경 지원	- PSSP V3.5 - GPFS V2.1
병렬 환경 지원	- PE V3.2
배치 작업관리	- LoadLeveler V3.1
성능 분석	- Performance Toolbox V3.0
프로그래밍 언어	- C for AIX V6.0 - VisualAge C++ Pro for AIX V6.0 - XL Fortran for AIX V8.1
수학 라이브러리	- ESSL V3.3, PESSL V2.3 - IMSL, NAG - SCALAPack, LAPack
HPSS	- HPSS pftp Client

1.13.2 응용 소프트웨어

대부분의 응용프로그램은 /applic 디렉토리 밑에 설치되어 있으며, /applic/samples 디렉토리에 주요 애플리케이션에 대한 초기화 파일 및 Job Command File의 예제가 있다.

<표 1-11> 응용 소프트웨어 지원 내역

분야	소프트웨어명	버전	License 현황	설치된 모듈
구조/고체	ABAQUS	6.5-1	6 Jobs[1cpu 기준] [모든 노드]	- STANDARD - Explicit - Viewer
	LS-DYNA3D	960,970	6 Jobs, 32CPU FLOATING [모든 노드]	- DYNA3D
	MSC/PATRAN	2003, 2005	1 job	
	MSC/NASTRAN	2005	nobel1	
	SYSNOISE	5.6		
열/유체	STAR-CD	3.15A, 3.20	4 Jobs & 4 CPU Parallel [모든 노드]	- Prostar - STAR v3.1x
	FLUENT	6.1.22	6 Jobs & 4 Node Parallel [모든 노드]	- Gambit 2.1.2 - T-Grid 3.6.8 - Fluent v6.1
	CFX	5.7	5 jobs(작업당 4 CPU)	
화학/생명	GAUSSIAN*	98, 2003	No Limit	
	NAMD	2.5		
	GAMESS	-		
	NWChem	4.5		
	GROMACS	3.1.4		
	APBS	0.2.6		
	AMBER	7.0		
	SuperCHARMM	29b,30b1		
	DS Modeling	1.1		
가시화	NCAR Graphics	4.2	No Limit	-무료 모듈만 지원

※ GAUSSIAN 2003 버전을 사용하기 위해서는 관리자에게 연락요망
[우준, Tel. 042)869-0595]

Note
가장 최신의 응용 소프트웨어 지원 현황을 확인하기 위해서는
http://www.ksc.re.kr/resource/resource_1_a.htm 을 참조하기 바란다.

1.14. 프로그램 컴파일 방법

1.14.1 순차 컴파일러 및 병렬 컴파일 스크립트

IBM 시스템에서는 아래 표와 같이 컴파일러 및 병렬 컴파일 스크립트가 지원된다.

<표 1-12> 컴파일러 명령어 리스트

언어	Serial(주1)	OpenMP(주1)	MPI(주1)	OpenMP+MPI(주1)
C	cc, xlc	cc_r	mpcc	mpcc_r
C++	xlC	없음(주2)	mpCC	없음(주2)
Fortran77	xlf	xlf_r	mpxlf	mpxlf_r
Fortran90	xlf90	xlf90_r	mpxlf90	mpxlf90_r
Fortran95	xlf95	xlf95_r	mpxlf95	mpxlf95_r

-주1: LoadLeveler에서 스케쥴링 방식으로 Gangscheduling을 사용하는 경우의 MPI 프로그램 혹은 Checkpoint를 할 경우에는 반드시 thread-enabled compile script(*_r)를 사용해야한다. [현재 Backfilling을 사용중임]
-주2: C++의 경우 현재는 OpenMP가 지원되지 않으나 pthread를 사용한 공유메모리 프로그램은 xlC_r 또는 mpCC_r을 사용하여 컴파일 할 수 있다.

아래는 Fortran을 위한 컴파일러 또는 컴파일 스크립트 사용 예이다:

```
$ xlf -o sample sample.f                    ----> Serial

$ xlf_r -o sample_omp  -qsmp=omp sample_omp.f    ----> OpenMP

$ xlf_r -o smple_smp -qsmp=auto sample_smp.f ----> 자동 병렬화

$ mpxlf -o para_sample  para_sample.f             ----> MPI

$ mpxlf_r -o hybrid_sample -qsmp=omp hybrid_sample.f ----> OpenMP+MPI
```

특기할 사항으로, OpenMP 프로그램은 '-qsmp=omp' 또는 '-qsmp=noauto' 옵션을 주어야하며, serial 프로그램의 자동 병렬 컴파일을 위해서는 '-qsmp =auto' 옵션을 준다.

C, C++ 컴파일러 사용법도 기본적으로 Fortran과 같으며, 컴파일 스크립트의 옵션 지정 순서는 순차 컴파일 사용법과 같다.

1.14.2 아키텍쳐 최적화 컴파일

시스템의 프로세서 종류에 따라 최적화하기 위해 다음과 같이 -qtune, -qarch, 그리고 -qcache 옵션을 사용할 수 있다.

-qtune=architecture

이 옵션은 이진 호환성을 유지하면서 프로세서 아키텍쳐에 최적화된 오브젝트 코드를 생성한다. 실행 프로그램은 원칙적으로 모든 IBM RS/6000 또는 pSeries 계열의 시스템에서 실행된다. 컴파일 시스템과 프로그램 실행 시스템이 같은 프로세서를 사용하는 경우에는 간단히 '-qtune=auto'를 지정해 주면된다. 컴파일 시스템과 실행 시스템이 다른 경우, 또는 일부러 이 옵션을 지정하고자 할 경우에는 <표 1-13>의 값을 참조하기 바란다.

-qarch=*architecture*

이 옵션은 프로세서 아키텍쳐에 의존하는 인스트럭션을 사용하도록 허용하며, 따라서 이진 호환성이 보장되지 않을 수 있다. 즉, 프로세서 아키텍쳐가 다른 시스템에서는 실행되지 않을 수 있다. -qtune과 마찬가지로 컴파일 시스템과 목적 시스템이 같은 경우에는 '-qtune=auto'를 사용할 수 있으며, 그 외의 경우에는 <표 1-13>를 참조하여 지정하기 바란다.

시스템의 아키텍쳐를 확인하려면 아래와 같이 lsattr 명령어를 사용하면 된다.

+ p690 컴퓨팅 시스템의 아키텍쳐 확인

```
 $ lsattr -EHl proc0
attribute     value                 description         user_settable
state         enable                Processor state     False
type          PowerPC_POWER4        Processor type      False
frequency     1300000000            Processor Speed     False
```

여기서 세 번째 행이 프로세서의 아키텍쳐를 나타내는 데, -qarch 또는

-qtune을 지정하지 않으면 모든 프로세서에 실행되는 공통 오브젝트 코드를 생성한다. 주요 프로세서 종류에 해당하는 -qarch 또는 -qtune의 옵션값은 아래의 표와 같다.

<표 1-13> -qarch 또는 -qtune의 옵션값

Type	-qarch 옵션	-qtune 옵션
컴파일 프로세서(주1)	auto	auto
공통(주2)	com	N/A
PowerPC	ppc	ppc
PowerPC_601	601	601
PowerPC_603	603	603
PowerPC_604	604	604
POWER	pwr	pwr
POWER2	pwr2, pwrx, pwrs(주3)	pwr2, pwrx, pwrs(주3)
PowerPC_POWER3	pwr3	pwr3
PowerPC_POWER4	pwr4	pwr4
RS64	rs64a	rs64a
RS64-II	rs64b	rs64b
RS64-III	rs64c	rs64c
기본 값	com(주4), ppc(주5)	pwr2(주4), pwr3(주5)

-주1: 컴파일하고 있는 시스템의 프로세서를 자동으로 감지하여 그에 맞게 컴파일 한다.
-주2: 모든 종류의 프로세서에서 지원되도록 컴파일 한다.
-주3: pwrs는 성능을 저하시킬 수 있는 고정밀도 인스트럭션이 포함되지 않게 한다.
-주4: 32-bit 모드인 경우
-주5: 64-bit 모드인 경우

-qcache

캐쉬의 구조에 최적화하는 옵션으로 POWER4 프로세서의 경우 아래와 같다:

-qcache=level=1:type=i:size=64:line=128:assoc=0:cost=13 ₩

-qcache=level=1:type=d:size=32:line=128:assoc=2:cost=11 ₩

-qcache=level=2:type=c:size=1440:line=128:assoc=8:cost=125만약 POWER4 시스템에서 컴파일하고 같은 시스템에서 실행하는 경우라면 간단히 '-qcache=auto'라고 지정해 주면된다. 주의할 것은 컴파일러 최적화 옵션 -O4

이상을 지정할 경우, '-qcache=auto'로 간주되므로 컴파일하는 시스템과 프로그램을 실행할 시스템의 프로세서가 다른 종류인 경우에는 목적 시스템에 맞는 -qcache 옵션을 명시해야 한다.

1.14.3 32bit/64bit 컴파일

기본적으로 AIX용 컴파일러는 프로그램을 32bit 모드로 컴파일한다. 원하는 bit 모드를 지정하려면 컴파일러, 아카이버, 링키지 에디터 등을 위해 아래와 같이 bit 모드 옵션이나 환경변수를 사용할 수 있다.

- 컴파일러 옵션: -q32/-q64
- OBJECT_MODE 환경변수: 32/64
- Archive(ar) 옵션: -X32/-X64/-X32_64
- 링키지 에디터(ld) 옵션: -b32/-b64

아래는 64-bit 오브젝트를 생성하도록 컴파일하고 확인하는 예이다:

```
$ xlf -c -q64 bt.f

$ dump -ov -X32_64 bt.o
bt.o:
                    ***Object Module Header***
# Sections      Symbol Ptr      # Symbols      Opt Hdr Len      Flags
        4      0x000026fe            81                0      0x0000
Flags=( )
Timestamp = "Feb 14 11:18:07 2002"
Magic = 0x1f7  (64-bit XCOFF)
```

위와 같이 dump 명령을 사용하면 실행 파일 또는 오브젝트 파일이 32-bit 모드인지 64-bit 모드인지 확인할 수 있는데, 위의 예에서는 '64-bit XCOFF'로 되어 있으므로 64-bit 오브젝트 파일임을 알 수 있다. 또한, dump 명령의 옵션인 -X32_64, -X32, -X64는 대상 파일의 bit 모드를 지정하는 것으로, 모를 때는 위와 같이 -X32_64를 사용하면 된다.

AIX 4.3 환경에서 컴파일한 64-bit 프로그램은 AIX 5L에서는 실행되지 않으므로 AIX 5L용으로 다시 컴파일 해주어야 하지만, 32-bit 프로그램은 AIX5L 이전의 버전에서 컴파일 된 실행파일을 그대로 사용할 수 있다.

운영체제의 32/64-bit 모드는 'ls -l /unix' 명령으로 확인할 수 있는데 /unix

가 /usr/lib/boot/unix_64에 링크되어 있으면 64-bit 커널이고, /usr/lib/boot/unix_mp에 링크되어 있으면 32-bit 커널이다.

1.14.4 대형 메모리 컴파일

32bit 프로그램의 주소 공간은 16개의 256MB의 세그먼트들로 나뉘어 지는데 기본적으로는 사용자 프로그램의 stack, heap, data 주소 공간을 위해서 세그먼트 하나만 할당된다. 그러므로 256MB이상의 메모리(주로 heap 메모리)를 사용하려면 아래의 예와 같이 링크 옵션 -bmaxdata를 사용한다.

```
$ xlf -o large_mem_program -bmaxdata:0x80000000  large_mem_program.f
```

-bmaxdata의 단위는 바이트이며 크기는 십진수, 8진수(0으로 시작), 16진수(0x로 시작)로 나타낼 수 있는데, 위의 예는 8개의 데이터 세그먼트를 허용하여, 최대 2GB까지 heap 메모리를 사용할 수 있게 하는 것이다. 64bit 프로그램 모드에서는 이와 같은 256MB 세그먼트 한계가 없으나 -bmaxdata를 지정하여 그 값을 명시할 수도 있다.

1.15. ESSL/PESSL[수학 라이브러리] 사용법

ESSL(Engineering and Scientific Subroutine Library)은 IBM pSeries와 IBM RS/6000 웍스테이션, 서버 그리고 SP 시스템을 위해 다양한 과학/공학용 고성능 수학 라이브러리를 제공하며, PESSL(Paralle ESSL)은 ESSL의 MPI 버전이다. 두 버전 모두 SMP 병렬 기능을 제공하며, MPI 프로그램이 아닌 경우에는 PESSL이 필요 없다. 또한 PESSL을 위해서는 ESSL이 먼저 설치되어야 하며, PESSL은 반드시 ESSL과 함께 사용해야한다.
ESSL 제품군이 지원하는 주요 분야는 아래와 같다.

가. 주요 기능

ESSL은 공개 라이브러리인 LAPACK 및 BLAS와 호환되며, PESSL은 ScaLapack 및 PBLAS와 호환된다. ESSL 제품은 이들 공개 라이브러리의 대부분의 루틴을 IBM 플랫홈에 최적화한 버전으로 제공하지만, 일부는 아직 지원되지 않는 것도 있다. 아래는 ESSL 제품군에서 제공하는 주요 기능이다:

▮Basic Linear Algebra Subprograms(BLAS)

- Linear Algebraic Equations
- Eigensystem Analysis
- Fourier Transforms

나. 사용예

ESSL 라이브러리는 아래의 예와 같이 컴파일 시 링크시켜주면 된다:

```
$ xlf -o sample -lessl sample.f
```

PESSL의 사용예는 아래와 같다:

```
$ mp_xlf -o sample -lessl -lesslp sample.f
```

다. CCI for LAPACK 사용법

사용자들의 프로그램 포팅을 용이하게 할 목적으로 기존에 개발된 소스를 수정하지 않고 이전의 루틴을 다른 목적 시스템에서 제공하는 루틴으로 대체해주는 도구를 wrapper라고 한다. LAPACK인 경우에도 http://www.netlib.org/lapack/essl에서 제공하는 CCI(Call Conversion Interface) wrapper를 사용하여 대부분의 LAPACK 기능을 ESSL로 대체하여 IBM 시스템에 최적화된 성능을 얻을 수 있다. 단, 컴파일을 할 때는 링크 단계의 오류가 발생하지 않도록 아래의 예와 같이 반드시 LAPACK이 먼저 링크되도록 한다:

```
$ xlf -o sample -L/usr/local/lapack -llapack -lessl
```

위의 예는 LAPACK이 이미 /usr/local/lapack 디렉토리에 설치되어 있고, CCI가 적용된 경우를 가정한 것이다.

1.16. POE를 통한 병렬 프로그램 실행

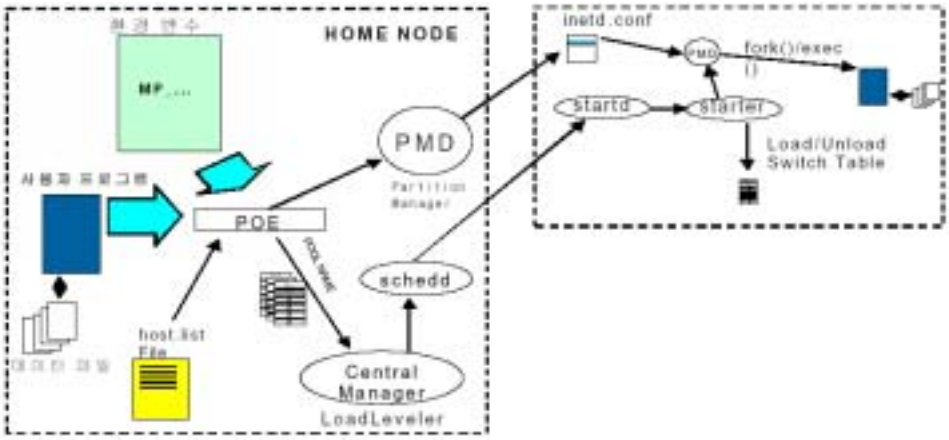
1.16.1 POE란 무엇인가?

POE(Parallel Operating Environment)는 IBM Parallel Environment(PE) 제품의 일부로서 병렬프로그램 실행 환경을 제공한다. PE는 POE 뿐만 아니라 아래와 같이 여러 가지 기능을 포함한다. 한 가지 사용자가 유의할 점은 현재 컴퓨팅 노드에서 병렬 작업은 POE를 통해서 Interactive하게 실행할 수 없으며, LoadLeveler를 통해서만 실행할 수 있다는 것이다.

- Message Passing 라이브러리: MPI, MPL
- 병렬 컴파일 스크립트: mpxlf, mpcc, mpCC
- 병렬 실행 환경: POE
- X-윈도우 병렬 런-타임 분석도구: Program Marker Array
- 병렬 파일 복사 유틸리티: mcp, mcpscat, mcpgath
- 병렬 디버거: pedb, pdbx
- 병렬 프로파일링 도구: xprofiler

1.16.2 POE 내부 구조

아래의 그림은 POE를 통한 병렬프로그램 수행 흐름도를 나타낸 것이다.



<그림 1-6> POE를 통한 병렬프로그램 수행 흐름도

1.16.3 POE 용어

- 노드(Node)

일반적으로 노드는 네트웍 상에서 유일한 호스트 이름(host name)을 가지며, 그 자체로서 독립적으로 운영될 수 있는 시스템을 말하는데, IBM SP 시스템 또는 클러스터 시스템을 구성하는 기본 단위 시스템이다. 사용자의 입장에서는 각 노드를 아래와 같이 두 가지로 구분할 수 있다.

+ 홈 노드(Home Node): 로그인하여 작업을 수행하는 노드. 로그인 노드(Login Node)라고도 한다.
+ 원격 노드(Remote Node): 홈 노드 이외의 노드를 말한다.
+ 계산 노드(Computing Node): 실제로 프로그램이 수행되는 노드를 말한다. 홈 노드와 원격 노드 모두가 계산 노드가 될 수 있으며, 각 사이트의 구성

환경에 따른다.

- 태스크(Task)

하나의 병렬프로그램이 시작되면 주어진 개수의 프로세스가 생성되는데, 이 각각의 프로세스를 태스크라고 한다. 노드 내부에서 각 태스크는 고유한 프로세스 ID를 가지며, 한 병렬프로그램에 속한 N개의 태스크들은 0, 1, 2, ..., N-1의 태스크 ID가 부여된다.

- 풀(Pool)

병렬 프로그램 수행 노드들을 쉽게 지정하기 위해 시스템 관리자가 미리 정의해 놓은 노드의 집합이다. 단순히 노드 이름(엄밀히 말하면 네트웍 어댑터의 이름)을 모아 놓은 것이다.

- 파티션(Partition)

파티션은 아래와 같이 두 가지 뜻으로 쓰인다.

+ POE 용어: 하나의 병렬프로그램에 속한 태스크 전체의 집합
+ PSSP 용어: IBM SP 또는 클러스터 구성을 위한 소프트웨어인 PSSP를 사용하여 서로 격리시킨 SP스위치 네트웍

- US(User Space) 프로토콜

SP 스위치 네트웍을 통한 고대역폭(high bandwidth)/저지연율(low latency) 메시지 통신 프로토콜이다. SP 스위치가 장착된 경우에만 사용이 가능하다.

- IP(Internet) 프로토콜

UDP/IP 메시지 통신 프로토콜이다. SP 스위치를 뿐만 아니라 일반 네트웍에서도 사용이 가능하다.

- 암시적 노드 할당(Implicit Node Allocation)

병렬작업을 위해 풀 또는 클래스만 지정하고, LoadLeveler가 자동으로 실제 노드를 할당해주는 방식을 말한다.

- 명시적 노드 할당(Explicit Node Allocation)

호스트 리스트 파일을 작성하여, 실제로 할당할 노드의 이름을 명시하는 방식이다.

- 호스트 리스트 파일(Host List File)

명시적 노들 할당을 위해 노드명 또는 네트웍 어댑터 주소를 명시한 파일을 말한다. 아래와 같이 호스트의 이름을 프로세스 수 만큼 적어주면 된다.

```
$ cat ./host.list
nobel1
nobel1
nobel1
nobel1
```

- SPMD(Single Program Multiple Data Parallel Programming Model)

파티션의 병렬 태스크들이 동일한 실행파일을 수행하되, 서로 다른 데이터를 조작하는 병렬프로그램 방식.

- MPMD(Multiple Program Multiple Data Parallel Programming Model)

파티션의 병렬 태스크들이 각각 다른 실행파일을 수행하는 병렬프로그램 방식.

1.16.4 POE를 통한 프로그램 실행 절차

- 프로그램 작성
- **컴파일 및 링크**: mpxlf, mpcc등의 병렬 컴파일 스크립트를 사용하여 컴파일한다.
- **파일 복사/공유**: 실행 파일 및 데이터 파일을 각 노드에서 접근할 수 있게 복사 또는 공유한다.
- **원격 실행 권한 부여**: 아래의 예와 같이 $HOME/.rhosts에 원격 노드들을 명시하여 rsh 권한을 부여한다.

```
$ cd $HOME
$ cat ./rhosts
nobela
nobelb
nobel1
nobel2
nobel3
nobel4
........
nobel20
nobel21
```

단, 이렇게 하더라도 root 사용자는 poe 작업이 허용되지 않는다.

- **실행 환경 변수 설정**: 태스크 수, 노드할당 방법 등 POE 환경 변수를 지정한다.
- **프로그램 실행**: POE를 사용하여 직접 대화식으로 병렬 프로그램을 실행하거나, LoadLeveler를 사용하여 일괄처리 스케쥴링에 의해 실행한다. 프로그램 모델에 따라 아래와 같이 실행파일을 로드한다.
  + SPMD 프로그램: 전체 노드에 같은 실행 파일의 복사본을 로드한다.
  + MPMD 프로그램: 노드마다 정해진 실행 파일을 개별적으로 로드한다.

1.16.5 POE 환경 변수

POE를 사용하여 병렬프로그램을 수행하기 위해 환경변수를 설정하거나 그에 동등한 poe 옵션을 지정해야 한다. POE 환경변수는 "MP_변수명"의 형식을 가지며, 그에 해당하는 명령행 옵션은 MP_를 제외한 변수명을 소문자로 바꾸어주면 된다. 단, 명령행 옵션 값 지정에는 '='를 사용하지 않는다. 또한 POE 명령행 옵션을 사용한 경우에는, 이전에 지정된 동일한 POE 환경변수는 일시적으로 무시된다.

환경 변수 지정은 아래와 같이 한다.

Type of Shel	To Set an environment variable	To set a command-line flag
Korn Shell	export VARIABLE=*value*	command -flag *value*
C Shell	setenv VARIABLE *value*	command -flag *value*

아래는 POE 환경변수를 지정하여 my_program이라는 병렬프로그램을 수행하는 예이다.

```
$ export MP_PROCS=10
$ export MP_HOSTFILE=./host.list
$ export MP_EUIDEVICE=csss
$ export MP_EUILIB=us
$ poe my_program
```

또는 아래와 같이 명령행 옵션을 지정하여 수행할 수도 있다.

```
$ poe my_program -procs 10 -hostfile ./host.list -euidevice csss -euilib us
```

사용자는 작업별로 자주 쓰이는 환경변수 설정 Shell script를 작성해 두는 것이 편리할 것이다.

다음은 주로 많이 쓰이는 POE 환경변수를 설명한다.

- MP_PROCS(-procs)

태스크의 수를 나타내며, 지정하지 않을 경우 기본값은 1이다. CPU 효율을 높이기 위해서는 가용한 프로세서의 개수 이하로 지정하는 것이 좋다. 또한, Hybrid(OpenMP+MPI) 프로그램인 경우에는 (MP_PROCS x OMP_NUM_THREADS) 또는 (MP_PROCS x XLSMPOPTS=parthds)의 값이 가용한 프로세서의 개수를 넘지 않게 지정하는 것이 바람직하다. 예를 들어, CPU 16개 짜리 노드 4개가 남아 있는데, 태스크당 쓰레드 8개의 효율이 가장 좋다면, 64개 쓰레드로 작업을 수행할 때 아래의 식과 같이 MP_PROCS=(# of tasks/node x # of nodes)=(2 x 4)=8이 가장 최적의 성능을 낼 것으로 기대할 수 있다.

64 threads = (8 threads/task) x (2 tasks/node) x (4 nodes)

- MP_RESD(-resd)

POE가 직접 노드를 할당하지 않고 LoadLeveler를 통해 자동으로 할당할 것인지의 여부를 지정한다. 지정하지 않은 경우 기본값은 아래와 같다.

+ no: MP_EUILIB=ip이고 MP_HOSTFILE에 호스트 리스트 파일이 지정된 경우
+ yes: 그 외의 경우

MP_RESD=yes인 경우, 노드는 LoadLeveler의 기본 Class(대개의 경우 inter_class) 안에서 선택된다. MP_EUILIB=us인 경우에는 스위치 자원을 할당해야 하므로 반드시 yes값을 가져야한다.

- MP_HOSTFILE(-hostfile, -hfile)

POE가 직접 노드를 할당하도록 호스트 리스트 파일명을 명시한다. 절대 경로가 아닌 경우에는 현재 작업 디렉토리에 대한 상대 경로명으로 간주하며, 지정하지 않을 경우 기본값은 ./host.list이다. 만약 “”(빈 문자열) 또는 NULL로 지정된 경우, 호스트 리스트 파일을 무시한다.

- MP_RMPOOL(-rmpool)

POE가 직접 노드를 할당하지 않고 지정된 풀에서 LoadLeveler가 자동으로 노드를 선택한다. 단, MP_RMPOOL 환경변수가 유효하려면 아래의 예와같이 MP_HOSTFILE=NULL이 지정되어야한다. 즉, 호스트 리스트 파일을 사용하도록

설정되어 있으면 풀 사용 옵션은 무시된다.

```
$ export MP_HOSTFILE=NULL
$ export MP_RMPOOL=1
```

풀 구성 정보는 LoadL_admin 파일에 정의되어지며, 사용자는 'llstatus -l' 명령을 실행하여 각 노드에 지정된 'Pool'의 이름을 확인할 수 있다. 단, 풀은 기본적으로는 구성되지 않으며 관리자가 구성해 주어야한다.

- MP_EUIDEVICE(-euidevice)

메시지 패싱 통신을 위한 네트웍 인터페이스명(en0, csss 등)을 지정한다. 단, MP_EUILIB=us인 경우에는 항상 MP_EUIDEVICE가 SP 스위치(csss 등)인 것으로 간주된다.

- MP_EUILIB(-euilib)

통신 모드를 지정한다. ip 또는 us가 가능하다. 단, LAPI 프로그램을 수행하려면 반드시 MP_EUILIB=us이어야 한다.

- MP_MSG_API(-msg_api)

메시지 패싱 라이브러리를 설정한다. MPI, LAPI, MPI LAPI가 가능하다. 지정하지 않으면 MPI로 간주된다.

- MP_LLFILE(-llfile)

대화형으로 POE 작업을 수행하되 LoadLeveler Job command file에 명시된 키워드 설정에 따라 환경변수를 지정하고 싶을 때, 그 파일명을 나타낸다. 단, MP_RESD=yes가 지정되어야만 지원된다. 아래는 사용 예이다.

```
$ export MP_RESD=yes
$ export MP_LLFILE=./lljob.cmd
$ poe my_mpi_program
```

- MP_SHARED_MEMORY(-shared_memory)

같은 노드 내의 태스크끼리의 메시지 패싱 통신에 공유메모리를 사용할 것인가, 실제 네트웍 어댑터를 사용할 것인가 지정한다. 가능한 값은 yes와 no이다. MP_SHARED_MEMORY=yes인 경우, 서로 다른 노드의 태스크간의 메시지 패싱은

네트웍을 거쳐 일어나지만 같은 노드 내의 태스크 간의 메시지 패싱은 실제 네트웍을 거치지 않고 공유메모리를 사용하여 일어나므로 훨씬 빠른 통신이 가능하다.

- MP_PGMMODEL(-pgmmodel)

  병렬 프로그램 모델을 명시한다. spmd와 mpmd가 가능하다. 지정하지 않은 경우 기본값은 spmd이다.

- MP_CMDFILE(-cmdfile)

  MP_PGMMODEL=mpmd인 경우, 각 태스크가 수행할 프로그램명을 명시한 파일명을 나타낸다.

- MP_INFOLEVEL(-infolevel)

  수행 상태 메시지 보고의 수준을 지정한다. 기본값은 1이며 가능한 값은 아래와 같다:

  + 0 : 에러
  + 1 : 경고, 에러
  + 2 : 참고, 경고, 에러
  + 3 : 참고, 경고, 에러 및 IBM 지원센터에서 참고할 진단 메시지
  + 4, 5, 6 : 참고, 경고, 에러 및 IBM 지원센터에서 참고할 상위 및 하위 레벨의 진단 메시지

1.16.6 POE 작업 실행

POE를 통한 작업 실행 방법은 프로그램 모델에 따라 아래와 같이 구분된다.

- 순차프로그램 작업 또는 명령어

  poe serial_program/command_name [POE_option_flags]

```
$ poe hostname  -procs 4  -rmpool 1 -hostfile NULL
```

- SPMD 프로그램 작업

  [poe] program_name [POE_option_flags] [executable_arguments]

```
$ poe mpi_program -procs 8 -hostfile ./host.list
$ mpi_program -procs 8  -hostfile ./host.list
```

위에서 두 번째의 경우처럼, MPI 프로그램인 경우에는 poe를 생략해도 무방하다.

- MPMD 프로그램 작업

아래의 예처럼 poe를 실행한 후 각 프로그램을 순서대로 명시해준다. MP_PROCS로 지정된 수만큼 프로그램 명이 입력된 후 자동으로 작업이

```
$ export MP_PGMMODEL=mpmd
$ export MP_PROCS=4
$ poe [Enter]
0:nobel1> master [Enter]
1:nobel2> worker [Enter]
2:nobel3> worker [Enter]
3:nobel4> worker [Enter]
$
```

또는 아래와 같이 미리 프로그램 리스트 파일을 작성하여 MP_CMDFILE 옵션으로 지정할 수도 있다.

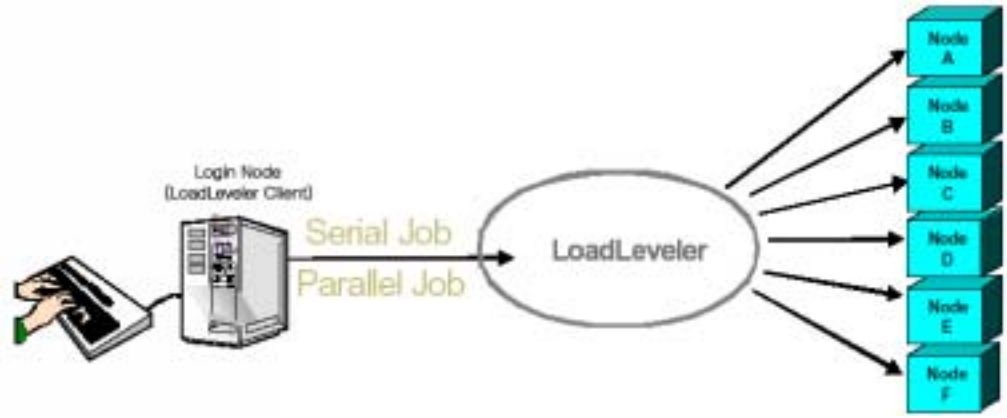
```
$ cat  pgm.list
master
worker
worker
worker
$ poe  -pgmmodel mpmd  -procs 4  -hostfile host.list -cmdfile pgm.list
```

단, 프로그램 리스트 파일의 각 행에는 단 하나의 프로그램 명을 명시해야한다.

1.17. LoadLeveler를 통한 배치 작업 실행

1.17.1 LoadLeveler 개요

LoadLeveler는 Batch Job 스케쥴링 프로그램으로서 Batch job이 네트웍으로 연결된 여러 컴퓨터 자원들 중에서 최적의 자원을 사용할 수 있도록 할당해 주는 기능을 한다. 이것은 여러 사용자가 Batch 작업을 수행할 때 Job을 분산 시킬 수 있어 전체 시스템의 성능을 향상시킬 수 있을 뿐만 아니라 사용자 가 최적 상태의 시스템을 직접 찾을 필요가 없다는 장점이 있다. 또한 쉘 스크립트, Serial Job, Parallel Job 등 슈퍼컴퓨터에서 실행할 수 있는 대부분의 작업을 처리할 수 있다.



<그림 1-7> LoadLeveler 작업 실행 흐름도

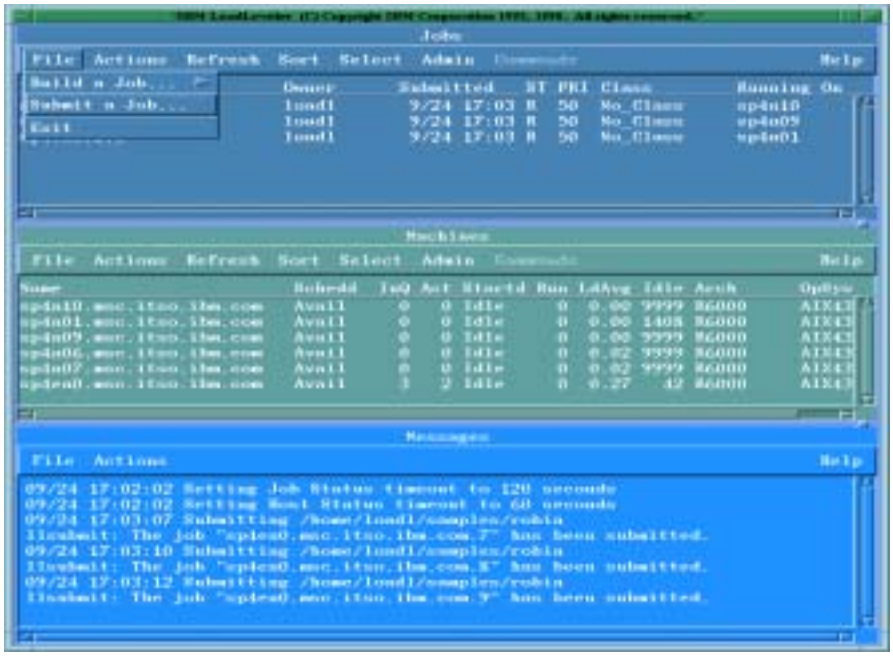
1.17.2 LoadLeveler의 작업 수행 절차

일반적으로 LoadLeveler를 사용하여 사용자 작업을 수행하는 절차는 아래와 같다.

- Job Command File 작성
- 작업 제출
- 대기 또는 수행중인 작업 취소
- 대기중인 작업 Holding
- 대기중인 작업 우선 순위 변경
- 작업 수행/완료 상태 조회

작업관리 절차는 명령 행으로 처리할 수도 있으며, 아래의 그림과 같이 xloadl을 수행하여 GUI 환경에서 처리할 수도 있다.

$ xloadl &



<그림 1-8> xloadl 실행화면

1.17.3 LoadLeveler 클래스 구성

프로그램 편집, 컴파일, 디버깅 등의 작업은 로그인 노드에서 Interactive로 실행 가능하지만, 계산 작업은 반드시 LoadLeveler를 통해서 배치로 실행해야 한다.

이에 따라 <표 1-14>에서와 같이 LoadLeveler에서 클래스는 32-way nobel5-6 노드에서 단일 CPU를 사용하는 normal 클래스, 1.3GHZ 16/24-way 노드에서 작업을 수행하는 p_normal_1.3 클래스, 1.7GHZ 32-way 노드에서 작업을 수행하는 p_normal_1.7 클래스, p_normal_1.3 및 p_normal_1.7 클래스 노드 전체에 걸쳐 작업을 수행하는 p_normal 클래스, Grand Challenge 사용자를 위한 grand 클래스로 구분된다.

<표 1-15> 및 <표 1-16>에서와 같이 각 사용자 작업 실행 시간, 클래스 대기 및 실행 작업 수에 제약을 가하고 있다. 또한 이러한 제약 사항은 작업의 부하에 따라서 수시로 변경될 수 있다.

<표 1-14> 클래스별 구성 내역

클래스 이름	Priority	Max Proc	Max Jobs	Wall_clock_limit	설 명
normal	1	1	no limit	720시간	일반 1CPU 작업
p_normal	2	572	no limit	720시간	p_normal_1.3 및 p_normal_1.7 클래스 중 가용 CPU가 있는 쪽으로 작업을 자동 제출해줌.
p_normal_1.3	2	60	no limit	720시간	1.3GHz POWER4 CPU 노드에서 실행되는 일반 병렬 작업
p_normal_1.7	2	512	no limit	720시간	1.7GHz POWER4 CPU 노드에서 실행되는 일반 병렬 작업
grand	1	32	no limit	960시간	Grand Challenge 과제 작업

<표 1-15> p690 시스템 클래스별 가용 CPU 수

클래스 이름	총 가용 CPU 수	실행 노드	최대 노드수
normal	124	nobel1/3-6	5노드
p_normal	572	nobel1/3-20	19노드
p_normal_1.3	60	nobel1/3-4	3노드
p_normal_1.7	512	nobel5-20	16노드
grand	32	nobel21	4노드

Note

- 현재 클래스 구성은 IBM 시스템에 로그인하여 “llinfo” 명령을 수행하여 확인할 수 있다.
  - ❖ Priority : 작업 대기열에서 클래스별 우선 순위를 나타낸다.
  - ❖ Max Proc : 작업 실행 시 할당받을 수 있는 최대 프로세서 수.
  - ❖ Max Jobs : 최대 실행 작업 수, 최대 값을 넘으면 더 이상 작업이 submit 되지 않고 바로 종료됨.
  - ❖ 로그인 노드인 nobela와 nobelb는 작업의 submit만 허용됨.

<표 1-16> 사용자 별 구성 내역

항목	User Limit	비 고
maxidle	2	큐에서 idle 상태인 작업 수
maxjobs	10	running 작업 수
maxqueued	unlimited	maxidle+maxjobs

1.17.4 Job Command File 작성

Job Command File은 LoadLeveler를 사용하여 작업을 실행할 때 사용하는 배치 작업 실행 스크립트이다.

가. Job Command File 문법

Job Command File을 작성할 때는, 다음과 같은 규칙을 따라야한다.

- 키워드문은 #@로 시작한다. #와 @ 사이와 전후의 공백은 무시된다.
- #로 시작하되 키워드문이 아닌 행(#@로 시작하지 않은 행)은 주석문이다.
- 키워드는 대소문자 구별을 하지 않는다. 그러나 일반 Shell Script와 같이 변수 또는 상수명은 대소문자를 구별하므로 주의해야한다. 즉, 키워드문에서 =의 좌측은 대소문자 구별을 하지 않고, =의 우측은 대소문자를 구별한다.
- 행연속은 ₩(역 슬래쉬)로 표현되며, 연속된 행은 #@로 시작해서는 안 된다.
- Job Step마다 맨 끝에 “#@queue" 선언행이 있어야한다.

Job Command File은 일반 텍스트 편집기를 통해 직접 작성하거나 xloadl이라는 LoadLeveler GUI를 통해 작성할 수 있다. 처음 LoadLeveler를 사용하는 경우에는 키워드 사용법이나 Job Command File의 문법에 익숙지 않아 어려움을 겪을 수 있는데, 이때는 xloadl이 제공하는 Job Command File 작성 기능을 이용하거나, 예제 Job Command File을 활용하는 것이 바람직하다.

나. Job Command File 주요 키워드

Job command file 내에서 적절한 키워드를 사용하여 원하는 작업을 위한 자원 할당 방법을 명시해야 한다. 주요 키워드는 아래와 같으며, 사용자는 이들 중에서 몇 가지만 사용하여 Job command file을 작성할 수 있다.

- job_type

  작업의 유형을 지정하며, serial, parallel, pvm3가 가능하다. 명시하지 않으면 serial로 간주된다. 단, 앞에서 설명한 대로 LoadLeveler의 정의에 따르면 공유메모리 병렬프로그램(OpenMP, Pthread 프로그램 등)의 job_type도 serial job이다. 사실은 job_type을 parallel로 지정하여 필요한 CPU 개수만큼 태스크 수를 요청하는 방식도 사용할 수 있으나, IBM 시스템에서는 사용자가 ConsumableCpus라는 변수를 지정하도록 강제하고 있다. 즉, LoadLevler Job Command File의 resources 키워드를 통해 사용할 CPU 개수를 명시하여, 각 노드마다 가용한 CPU 자원을 관리하도록 한다. 이때 ConsumableCpus는 각 태스크마다 적용되므로 OpenMP 프로그램의 job_type은 serial로 해야 원하는 쓰레드 수만큼만 CPU를 요청하게 된다. 아마도 초보자에게는 이 부분이 어려울 것으로 예상되는데 혼동하지 않기를 바란다.

- job_name

  작업의 이름을 지정하며, 명시하지 않으면 LoadLeveler가 임의로 부여한다.

- step_name

  작업 파일 내에서 Job Step 이름을 지정하며, 명시하지 않으면 LoadLeveler가 0, 1, 2,.와 같이 Job Command File내의 순서대로 부여한다.

- input

  Standard input을 지정한다.

- output

  Standard output을 지정한다.

- error

  Standard error를 지정한다.

- initialdir

  초기 작업 디렉토리를 지정한다.

- node

  Job Step을 위해 할당할 노드 수를 지정하며, 각 Job Step 마다 지정해 주어야 한다. 노드 수를 하나의 숫자로 지정하거나 범위로 지정할 수 있다. 지정하지 않으면 1로 간주된다.

  예 1) # @ node = 6, 12
  # @ tasks_per_node = 2

  예 2) # @ node = 6
  # @ total_tasks = 12

위의 예에서처럼, tasks_per_node 또는 total_tasks 키워드와 함께 사용한다.

- blocking

일정한 개수씩 태스크를 분배하는 노드 할당 방법으로, 한번에 분배할 태스크의 개수를 나타낸다. 카드놀이에서 패를 돌리는 카드의 장수를 연상하면 이해가 될 것이다. 값은 양의 정수 또는 unlimited가 가능하다. unlimited인 경우는 한 노드를 가득 채우고 나서 다음 노드를 가득 채워나가는 방식으로 태스크를 분배한다.

예 1) # @ blocking = unlimited
예 2) # @ blocking = 1
예 3) # @ blocking = 4

- tasks_per_node

노드당 태스크 수를 지정한다.

- total_tasks

수행할 전체의 태스크 수를 지정한다.

- class

작업 수행을 위한 LoadLeveler class를 지정한다. 지정하지 않을 경우 일반적으로 No_Class라는 이름의 클래스로 간주된다.

- node_usage

해당 Job Step이 수행될 때, 같은 노드에서 다른 Job의 수행을 허용할 것인지 지정한다. shared와 not_shared가 가능하다.

- network

통신 프로토콜, 네트웍 어댑터, 통신 특성 등을 지정하며 각 Job Step마다 명시해야 한다. 지정 형식은 아래와 같다.

#@ network.protocol = network_type [, [usage] [, mode [, comm_level]]]

+ protocol : MPI, LAPI, PVM가 가능하다. 단, LAPI는 SP 스위치 network_type과 US mode를 함께 지정해야한다.
+ network_type : 반드시 명시되어야 하며, 어댑터 이름(Etherchannel, csss 등) 또는 인터페이스 종류(Etherchannel, ethernet, switch 등)로 나타낼 수 있다. IBM 시스템은 SP 스위치를 통해서 노드간 통신이 이루어지므로 여기에 csss를 사용하면 된다. 어댑터 구성 정보는 'llstatus -l'을 실행하여 각 노드의 'Adapter'를 확인하기 바란다.
+ usage : shared 또는 not_shared. 지정하지 않으면 shared로 간주된다.
+ mode : IP 또는 US. 지정하지 않으면 IP로 간주된다. 단, US mode는 SP 스위치 network_type에서만 지원된다.

+ comm_level : US mode에서만 지원되며, 태스크간의 통신량의 수준을 나타낸다. LOW, AVERAGE, HIGH가 가능하다. 즉, 이 옵션은 스위치 사용량을 자원화하여 노드 할당을 최적화하기 위한 것이다.

예 1) #@ network.MPI = csss,shared,US

←현재 IBM 시스템의 노드간 병렬 작업 실행 시 사용 가능한 network_type, usage, mode임.

- notification

알림 메일을 보낼 조건. always, error, start, never, complete가 가능하며 일반적으로 아래와 같이 complete를 많이 사용한다. 지정하지 않으면 complete로 간주된다.

예 1) #@ notification = complete

- notify_user

알림 메일을 보낼 주소. 지정하지 않을 경우, 작업 제출 노드의 작업 제출자 주소로 메일이 보내진다. 사용자가 자주 사용하는 메일 주소를 사용하면, 직접 시스템에 접속하지 않더라도 자동적으로 작업의 수행 상태를 알려주므로 편리할 것이다.

예 1) #@ notify_user = test@kisti.re.kr

- resources

Consumable resource 종류 및 요구량을 지정한다. IBM 시스템 사용을 위해 가장 중요한 옵션은 ConsumableCpus와 ComsumableMemory 인데 반드시 지정하여야 한다.

예) #@ resources = ConsumableCpus(16) ConsumableMemory(1gb)

위의 예는 태스크 당 16개 CPU 및 1GB의 메모리를 요청하는 것이다.

ConsumableCpus는 serial, Pthread 또는 OpenMP 프로그램인 경우에는 실제로 사용할 CPU 개수를 뜻하며, MPI 작업인 경우에는 각 태스크당 쓰레드 개수라고 보면 된다. 특히, OpenMP 혹은 OpenMP+MPI 프로그램인 경우에는 OMP_NUM_THREADS 또는 XLSMPOPTS=parthds 변수의 값과 ConsumableCpus 값이 같아야하고, serial 혹은 순수 MPI 프로그램인 경우에는 ConsumableCpus(1)을 지정하면 된다.

ConsumableMemory는 1CPU, Pthread 또는 OpenMP 프로그램인 경우에는 전체 메모리 용량을 뜻하며, MPI 프로그램인 경우에는 각 태스크 당 메모리 용량이라고 보면 된다.

※ ConsumableMemory 단위

❖ 100MB : ConsumableMemory(100mb) 혹은

ConsumableMemory(100) : default 단위 MB

❖ 1GB : ConsumableMemory(1gb)

이와 같은 Consumable reosurce의 구성 정보는 'llstatus -R' 명령으로 확인할 수 있다.

- requirements

할당될 노드가 만족해야할 필수 요구 조건들을 지정한다. 아래의 예는 반드시 nobel5에서, 혹은 nobel5나 nobel6에서 수행해야함을 지정한 것이다.

예 1) #@ requirements = (Machine == "nobel5")
or
#@ requirements = (Machine == {"nobel5" "nobel6"})

- wall_clock_limit

모든 사용자는 반드시 예상 작업 소요 시간(Elapsed time)을 지정해야만 한다. 초단위 또는 시:분:초 형식으로 지정한다. Backfill 스케쥴링에서 지원되며, 앞에서 대기 중인 작업의 수행에 영향을 미치지 않으면서 현재 가용한 자원만 사용하여 wall_clock_limit 시간 내에 작업을 마칠 수 있는 경우, 먼저 수행하도록 허용한다. 단, wall_clock_limit가 지나도 완료되지 않은 경우에는 LoadLeveler가 강제로 종료시키므로 안전하게 여유 시간을 주어 지정해야한다.

- checkpoint

Checkpoint 옵션을 설정하며, interval, yes, no가 가능하다.

· no: 어떤 경우에도 checkpoint를 수행하지 않음.
· yes: 수동 및 프로그램 내의 checkpoint 함수 호출에 의한 checkpoint를 수행함.
· interval: LoadLeveler에서 정해진 일정한 간격으로 자동적으로 checkpoint를 일으킴. 또한 yes인 경우와 마찬가지로 수동 및 프로그램 내의 checkpoint 함수 호출에 의한 checkpoint도 수행함.

- restart_from_checkpoint

  Checkpoint된 프로그램의 restart 옵션을 설정하며, yes, no가 가능하다.

- ckpt_dir

  Checkpoint 파일의 디렉토리명을 지정한다.

  예) /ckpt_restart →컴퓨팅 노드별로 제공되는 로컬 체크포인트 디렉토리로써 특별한 경우가 아니라면 이 전용 디렉토리를 사용해야 한다.

- ckpt_file

  Checkpoint 파일의 경로명을 지정한다. 절대 경로명으로 지정한 경우 ckpt_dir 키워드는 무시된다.

- restart_on_same_nodes

  수행도중 미완료 상태로 중단된 작업이 restart 또는 restart_from_checkpoint 키워드에 의해 재실행될 경우, 반드시 이전과 같은 노드에서 수행할 것인지의 여부를 설정한다. yes, no가 가능하다. Checkpoint 파일을 로컬 디렉토리에 생성한 경우에는 반드시 이 옵션을 yes로 설정해야한다. 그 외에도 필요에 따라 이 값을 yes로 설정할 수 있다.

다. Job Command File 작성 예제

우선 아래와 같이 몇 가지 유용한 주의사항을 숙지하기 바란다.

> Note
>
> Job Command File 작성 시 다음 사항에 유의하자
>
> ■ Serial 및 OpenMP 프로그램인 경우, job_type=serial 또는 생략
>
> ■ 반드시 ConsumableCpus, ConsumableMemory 및 wall_clock_limit을 지정해야함.
>
> ■ MPI 통신 네트웍 옵션: Network.MPI=csss, shared, US
>
> ■ 노드 내 MPI 통신 옵션: MP_SHARED_MEMORY=yes
>
> ■ OpenMP busy-wait threads 옵션: XLSMPOPTS="spins=0:yields=0"

아래에 설명된 파일의 예제는 /system/samples/LoadL 디렉토리 아래에서 참조할 수 있다.

- Serial 프로그램 : /system/samples/LoadL/serial.cmd

```
#!/bin/ksh
#@ job_type = serial
#@ class = normal
#@ resources = ConsumableCpus(1) ConsumableMemory(1gb)← 총 1GB 메모리 요구
#@ notification = complete
#@ notify_user = user_e-mail@ksc.re.kr
#@ output = $(step_name).out
#@ error = $(step_name).err
#@ wall_clock_limit = 1:00:00
#@ queue
cd /ytmp/test
./long_serial_program
```

Note
작업 수행 종료 시 사용자 메일 주소로 작업 종료 정보를 송신하도록 다음과 같이 옵션을 설정하면 편리하다.

```
#@ notification = complete
#@ notify_user = user_e-mail@ksc.re.kr
```

- Pthread 프로그램 : /system/samples/LoadL/abaqus.cmd

일반 pthread 프로그램을 위한 Job command file의 예이다. 예제는 abaqus.cmd이다.

```
#!/bin/ksh
#@ job_type = serial
#@ job_name = bmt
#@ step_name = abaqus04
#@ wall_clock_limit = 2:00:00
#@ input = /dev/null
#@ output = $(step_name).out
#@ error = $(step_name).err
#@ initialdir = /ytmp/abaqus04
#@ notify_user = user_e-mail@ksc.re.kr
#@ class = p_normal
#@ resources = ConsumableCpus(4) ConsumableMemory(1gb) ← 총 1GB 메모리 요구
#@ notification = complete
#@ queue
cd /ytmp/abaqus04
/apphome/abaqus/abaqus58-15/abaqus j=abaqus_test cpus=4 inter
```

job_type을 serial로 하였으나 ConsumableCpus(4)로 지정하였으므로 쉘스크립트 행에서 cpus=4가 의도하는 대로 4개의 CPU를 요청하게된다. 실제로는 LoadLeveler는 CPU를 할당할 때, Job command file 내의 키워드문에 지정한 job_type, ConsumableCpus, total_tasks 등의 조건들만 고려하며, Shell script 행의 cpus와 같은 옵션은 인식하지 못한다. 단지, 쉘스크립트 행의 프로그램이 수행될 때, 실행 프로그램의 입력행 옵션으로 cpus 값이 전달되어 실제로 cpus 만큼의 쓰레드를 사용하여 실행되는 것이다. 즉, LoadLeveler는 다른 LoadLeveler 작업이 한 작업에 예약된 CPU 개수 만큼을 사용하지 못하도록 할 뿐이며, 실제로 응용 프로그램이 꼭 그 CPU 개수 만큼 사용하도록 할 수는 없다. LoadLevler와 응용 프로그램은 서로 별개이며 아무런 상호 작용도 하지 않으므로 사용자가 위에서 처럼 LoadLeveler ConsumableCpus와 응용프로그램 입력행 옵션인 cpus가 같게 지정해 주어야 하는 것이다.

Note
입출력 및 에러를 다음과 같은 옵션으로 특정 파일에 저장할 수 있다.

```
#@ input = /dev/null
#@ output = $(step_name).out
#@ error = $(step_name).err
```

- OpenMP 프로그램 : /system/samples/LoadL/openmp.cmd

```
#!/bin/ksh
#@ job_type = serial
#@ class = p_normal
#@ wall_clock_limit = 1:00:00
#@ notification = complete
#@ notify_user = user_e-mail@ksc.re.kr
#@ output = $(step_name).out
#@ error = $(step_name).err
#@ resources = ConsumableCpus(16) ConsumableMemory(1gb) ← 총 1GB 메모리 요구
#@ step_name = mm5_omp.16
#@ queue
export LANG=en_US
export XLSMPOPTS="parthds=16:yields=0:spins=0"
/u/workshop/MM5/Run/mm5.exe > mm5.16.stdo 2>&1abaqus_test cpus=4 inter
```

ConsumableCpus(16)과 parthds=16이 서로 일치함을 확인하기 바란다. OpenMP 쓰레드 개수는 위의 예에서 처럼 IBM XL runtime 변수인 XLSMPOPTS="parthds="로 지정하거나 OpenMP runtime 변수인 OMP_NUM_THREADS로 지정해 준다. 위의 XLSMPOPTS 행을 아래와 같이 바꿀 수 있다:

```
export XLSMPOPTS="yields=0:spins=0"
export OMP_NUM_THREADS=16
```

XLSMPOPTS 환경변수에서 'yields=0:spins=0'은 쓰레드 수행 시 busy wait 상태를 유지하여 쓰레드가 sleep 상태로 빠지지 않게 하여 수행 성능은 좋아지지만 그만큼 전체 CPU 사용 시간은 증가한다. 프로그램 성능과 시스템 사용료 산정 기준을 고려하여 이 옵션을 적절히 사용해야 할 것이다.

- MPI 프로그램 : /system/samples/LoadL/mpi.cmd

```
#!/bin/ksh
#@ job_type = parallel
#@ notification = complete
#@ notify_user = user_e-mail@ksc.re.kr
#@ class = p_normal
#@ wall_clock_limit = 1:00:00
#@ Network.MPI = csss,shared,US
#@ total_tasks = 16
#@ blocking = unlimited
#@ resource = ConsumableCpus(1) ConsumableMemory(1gb)←총 16GB 메모리 요구(MPI 태스크 당 1GB)
#@ output = $(step_name).out
#@ error = $(step_name).err
#@ queue
export LANG=en_US
export MP_SHARED_MEMORY=yes
/system/super/loadl/mpi_pi_reduce
```

MPI 프로그램을 수행하는 예로서, job_type=parallel로 지정하였다. network.MPI 키워드와 total_tasks와 같은 노드 할당 키워드를 지정하였다. node 키워드를 지정하지 않았으므로 node=1로 간주된다. csss는 IBM 시스템의 MPI 통신용 네트웍 종류이며, resourcses 옵션에서

ConsumableCpus와 ConsumableMemory를 지정해야함을 주의하기 바란다. executable 키워드가 없으므로 Job command file 자체가 실행파일로 간주될 것이다.

- Hybrid(OpenMP+MPI) 프로그램 : /system/samples/LoadL/omp_mpi.cmd

```
#!/bin/ksh
#@ job_type = parallel
#@ class = p_normal
#@ wall_clock_limit = 2:00:00
#@ notification =complete
#@ notify_user = user_e-mail@ksc.re.kr
#@ tasks_per_node = 2
#@ node = 4
#@ resources = ConsumableCpus(8) ConsumableMemory(1gb)
#@ step_name = mm5_hybrid.64
#@ network.MPI = csss,shared,US
#@ output = $(step_name).out
#@ error = $(step_name).err
#@ queue
export XLSMPOPTS="yields=0:spins=0"
export OMP_NUM_THREADS=8
/u/workshop/MM5/Run/mm5.exe
```

위에서 OMP_NUM_THREADS=8이므로 각각의 MPI 태스크는 8개의 쓰레드로 수행하겠다는 것이며, tasks_per_node=2이므로 노드당 2개의 태스크, 즉 노드당 총 16개의 쓰레드가 필요하게 된다. node=4이므로, 전체 태스크 개수는 8개가 되며, 전체 쓰레드 수는 64개가 된다. 또한, 태스크당 1GB씩 총 8GB의 메모리를 필요로 한다.

Note

/xtmp1 스크래치 파일 시스템을 사용하여 배치 작업 수행하기

```
#!/bin/ksh
#@ job_type = serial
#@ class = normal
#@ resources = ConsumableCpus(1) ConsumableMemory(100mb)
#@ wall_clock_limit = 2:00:00
#@ queue
rm -rf /xtmp1/wjnadia        #작업 실행 노드의 /xtmp1 디렉토리에 기존 작업 디렉토리 삭제
mkdir /xtmp1/wjnadia         #실행노드의 /xtmp1 디렉토리에 새로운 작업 디렉토리 생성
cp -f /edun/wjnadia/* /xtmp1/wjnadia #작업 실행에 필요한 파일들을 실행노드의 /xtmp1 디렉토리로 복사
cd /xtmp1/wjnadia                    #실행 노드의 /xtmp1 작업 디렉토리로 작업 경로 변경
  ./long_serial_program
cp -f /xtmp1/wjnadia/* /edun/wjnadia/output/  #작업 결과 파일을 홈 디렉토리로 복사
rm -rf /xtmp1/wjnadia                         #/xtmp1의 작업 디렉토리 삭제
```

1.17.5 LoadLeveler 명령어

가. Job Submit

Job Command File을 사용하여 작업을 제출하기 위해 llsubmit 명령을 사용하거나 xloadl의 작업 제출 메뉴를 사용할 수 있다. 명령행을 이용할 경우의 예는 다음과 같다.

```
$ llsubmit  test.cmd
```

위에서 test.cmd는 현재 작업 디렉토리의 Job Command File 이름이다.

나. 작업 상태 조회

- **일반적인 작업 상태 조회 : llq**
  + Machine name : 작업을 스케쥴링한 노드 이름. 일반적으로, 실제 작업이 수행되고 있는 노드와 꼭 같지는 않다.
  + Job ID : Job Command File이 제출될 때마다 새로 부여되는 번호로서, 하나의 Job Command File에 속한 Job Step은 공통의 Job ID에 속한다.
  + Step ID : 같은 Job Command File에 속한 각각의 Job Step을 구분하기 위한 번호이다.

이렇게 부여된 작업 번호는 작업 상태의 조회 또는 변경을 위해 제출된 작업의 상태는 아래와 같이 llq 명령으로 조회할 수 있다.

```
$ llq
Id                    Owner     Submitted    ST PRI Class       Running On
---------------------- ----------- ------------ -- --- ------
nobel.2740.0          hsyi        12/5  09:30 R  50  normal       nobel2
nobel.2739.0          esim        12/5  09:22 R  50  normal       nobel3

nobel.2741.0          hsyi        12/5  09:34 R  50  normal       nobel2
nobel.2738.0          hsyi        12/5  09:21 R  50  normal       nobel3
nobel.2727.0          hsyi        12/2  13:15 R  50  express      nobel2
nobel.2734.0          yhkim       12/4  18:19 R  50  express      nobel2

6 job steps in queue, 0 waiting, 0 pending, 6 running, 0 held
```

위에서 Id는 작업 번호, ST는 작업 상태, PRI는 대기열 우선 순위, Running On은 현재 작업이 수행중인 노드를 나타낸다. 자세한 -l 옵션을 사용하면 상세한 작업 상태를 볼 수 있다.

- 사용자 작업이 idle인 이유 알아보기 : llq -s Job_ID

나열된 리스트의 하위 부분의 "EVALUATIONS FOR JOB STEP nobel1.supercomputing.re.kr.8291.0"에 기술된 내용을 통해서 이 작업이 요청한 자원이 부족한 것을 알 수 있다.

```
nobela:/system/root/bin> llq -s nobel1.8298.0
============ EVALUATIONS FOR JOB STEP nobel1.supercomputing.re.kr.8298.0 ============
SUMMARY
This LoadLeveler cluster does not have sufficient resources at the present time to run this job step.
.........................
```

Note
1) 위 예에서와 같이 idle 상태인 이유가 사용자가 요구한 CPU 자원을 제공할 수 있는 노드가 현재 없는 경우라면, 가용 CPU 자원이 확보될 때까지 대기하여야 한다.
2) 단, MPI 작업인 경우에 일부 노드들의 가용 CPU 자원의 합이 요구 CPU 자원량 이상을 제공할 수 있다면, Job Submit File에 **#@ blocking = unlimited** 옵션을 추가하여 다수의 노드 간에 걸쳐서 MPI 작업을 수행할 수 있다. 물론 단일 노드 내에서보다는 작업 수행 성능이 저하될 수 있음.

다. 노드별 가용 CPU 수 알아보기 (llstatus -R)

현재 각 노드별 전체 CPU 수 및 가용 CPU 수를 확인할 수 있다.

ConsumableCpus(**가용CPU수,**전체CPU수)

```
$ llstatus -R
Machine                        Consumable Resource(Available, Total)
------------------------ -----------------------------------
nobel1.supercomputing.re.kr    ConsumableCpus(16,16)
nobel10.supercomputing.re.kr   ConsumableCpus(12,32)
nobel11.supercomputing.re.kr   ConsumableCpus(7,32)
   :(생략)
nobel7.supercomputing.re.kr    ConsumableCpus(11,32)
nobel8.supercomputing.re.kr    ConsumableCpus(10,32)
nobel9.supercomputing.re.kr    ConsumableCpus(15,32)
```

라. 클래스 상태 조회

- 클래스별 최대 타스크 수 및 가용 타스크 수 알아보기 : llclass

아래 예에서 Free Slots은 해당 클래스에서 추가로 수행할 수 있는 타스크 개수이며, Max Slots는 할당된 최대 타스크 개수이다. 특정 클래스의 자세한 정보를 보고 싶으면 아래와 같이 -l 옵션을 사용한다.

```
$llclass

Name          MaxJobCPU   MaxProcCPU   Free  Max  Description
              d+hh:mm:ss  d+hh:mm:ss   Slots Slots
---------- ---------- ----------  ---- ---- ---------------------
p_normal_1.7   undefined    undefined    207   512   Normal parallel job - 1.7 GHz CPU
normal         undefined    undefined    106   288   Normal serial job
p_normal_1.3   undefined    undefined     32    48   Normal parallel job - 1.3 GHz CPU
p_interact     undefined    undefined     32    32   Interactive parallel job - nobel1 and nobel2
grand          undefined    undefined     27    32   Grand challege job - 1.7 GHz CPU
--------------------------------------------------------------------------------
"Free Slots" values of the classes "normal", "p_normal_1.7" are constrained by the MAX_STARTERS limit(s).
```

- 클래스별 Wall_clock_limit 알아보기 : llclass -l normal |grep Wall

아래 예제에서 wall_clock time 30일로 정의되어 있음. 일수+시간:분:초

```
nobela:/system/root/bin> llclass -l normal |grep Wall
   Wall_clock_limit: 30+00:00:00, undefined (2592000 seconds, undefined)
```

마. 작업 일시 정지

LoadLeveler 큐에 대기중인 작업을 일시 정지시키거나 다시 대기 상태로 환원하기 위해 llhold 명령을 사용한다. 아래는 실행 예이다.

```
$ llhold nobel.2740.0
$ llhold -r nobel.2740.0
```

바. 작업 취소

수행중이거나 대기중인 작업을 취소하려면, llcancel 명령을 사용하거나 xloadl의 취소 메뉴를 사용한다. 취소할 작업은 작업 번호로 지정하여 아래와 같이 취소 명령을 실행한다.

```
$ llcancel nobel.2740.0
```

사. 대기열 우선 순위 변경

llprio 명령으로 작업의 대기열 우선 순위를 변경할 수 있다. 아래는 사용자 우선 순위를 10만큼 높이는 예이다:

```
$ llprio +10 nobel.2740.0
```

직접 User Priority를 지정하려면 아래의 예와 같이 +, -를 사용하여 표시한다.

```
$ llprio 100 nobel.2740.0
```

단, 동일 사용자의 동일 클래스 내에서만 적용된다.

아. Checkpointing

LoadLeveler 작업의 강제 체크 포인트는 아래와 같이 llckpt 명령으로 지원된다.

```
$ llckpt nobel.20.0
```

위와 같이 아무 옵션 없이 Jobstep ID만 지정한 경우에는 checkpoint 후에도 작업은 계속 수행된다. llckpt 명령어의 옵션은 아래와 같다:

-r: Checkpoint 완료를 기다리지 않고 바로 작업을 계속 수행
-k: Checkpoint 완료 후 작업 중단
-u: Checkpoint 완료 후 대기열에 Holding 상태로 전환

자. 작업 시간 정보

LoadLeveler Notification Mail은 작업 수행 결과의 시간 정보를 전해준다. 작업이 끝난 후 미리 지정한 수신자(notify_user)의 e-mail을 확인하면 해당 작업에 대해 아래의 내용을 확인할 수 있다.

- Submitted at : 작업 제출 시간
- Started at : 작업 개시 시간
- Exited at : 작업 종료 시간. (Exited at - Started at)으로 작업 수행 시간(Elapsed time)을 계산할 수 있다.
- Real Time : 제출 시간부터 종료 시간까지의 소요 시간. 사용자 입장에서는 실제로 느끼는 작업 소요 시간이 된다.
- Job Step User Time : 작업이 수행되는 동안 사용자 영역에서 소요한 CPU 사용 시간
- Job Step System Time : 작업이 수행되는 동안 그 작업을 위해 시스템 영역에서 소요한 CPU 사용 시간
- Total Job Step Time : 총 CPU 사용 시간. (Job Step User Time + Job

Step System Time)으로서 사용자의 작업이 사용한 CPU 시간을 말한다. 일반적으로 시스템 사용료를 부과할 때 참조하는 CPU 사용 시간이다.

- Starter User Time : LoadLeveler Starter 프로세스가 작업을 관리하기 위해 사용자 영역에서 소요한 CPU 사용 시간. 이 시간이 사용자 작업에 직접 영향을 미치는 부하로 유일한 부분으로 볼 수 있으며, 상대적으로 무시할 정도로 작다.
- Starter System Time : LoadLeveler Starter 프로세스가 이 작업을 위해 시스템 영역에서 소요한 CPU 사용 시간. 일반적으로 매우 작다.
- Total Starter Time : LoadLeveler Starter 프로세스가 사용한 총 CPU 시간.

1.17.6 LoadLeveler 용어

- **노드** : 자체의 운영체제와 CPU 및 메모리 등의 하드웨어를 가지고 운영되는 독립된 시스템으로서 LoadLeveler 클러스터 구성의 최소 단위를 말한다.
- **클러스터** : 작업 부하 분산을 위해 LoadLeveler가 한 묶음으로 관리하는 노드들의 집합을 말한다. 하나의 글로벌 구성 파일에 의해 관리된다.
- **풀(Pool)** : LoadLeveler 클러스터 내에서 편의상 특정 노드들을 모아서 관리하기 위한 단위로서 Pool 이름을 부여하여 사용한다. 단순히 노드의 이름(엄밀히 말하면 통신 어댑터의 네트웍상의 이름)들을 모아놓은 것이다.
- **클래스(Class)** : Pool이 단순히 노드의 집합이라면, 클래스는 가용한 자원의 종류 및 필요량, 스케쥴링 선택사항 등 구체적인 특성을 부여한 노드들의 집합이다. 사용자가 작업을 제출할 때 작업의 형태와 필요한 자원, 시스템 사용료 등을 고려하여 선택해야 한다.
- Job : LoadLeveler를 통해 한 번에 제출하여 수행하는 작업 단위이다. 하나의 Job은 하나의 Job Command File에 의해 정의된다.
- Job Step : 하나의 Job 안에서 독자적으로 수행되는 작업의 단위로서, 각 Job Step은 순차 또는 병렬 작업이 될 수 있다. 하나의 Job Command File 안에서 여러 개의 Job Step이 정의될 수 있다.
- **태스크(Task)** : 병렬 작업에서 같이 수행되는 각각의 프로세스 단위를 말하며, 병렬 Job Step은 여러 개의 태스크로 이루어진다.
- **순차 작업(Serial Job)** : 하나의 노드 내에서만 수행될 수 있는 작업을 말하며, LoadLeveler는 특별히 명시하지 않는한 기본적으로 하나의 작업을 순차 작업으로 간주한다. 한가지 주의할 점은 공유메모리 병렬프로그램도 LoadLeveler에서는 순차 작업으로 정의한다.
- **병렬 작업(Parallel Job)** : LoadLeveler에서 말하는 병렬 작업은 MPI, PVM 등을

통해 여러 노드에 걸쳐 병렬로 수행될 수 있는 작업을 말한다.

- Consumable Resource : LoadLeveler 클러스터 내에서 임의로 정의한 자원을 말하며, 수량으로 표시된다. Machine resource와 Floating resource의 두 가지가 있으며, Machine resource는 각 노드마다 정의될 수 있는 임의의 자원을 말하고, Floating resource는 특정 노드에 상관없이 전체 클러스터에 대해 정의될 수 있는 자원을 말한다.

1.17.7 배치 작업관리 프로그램 비교

- Batch 작업관리 시스템 명령 비교

<표 1-17> Batch 작업관리 시스템 명령 비교

구분	LoadLeveler	NQS	LSF
status	llq	qstst	bjobs
submit	llsubmit	qsub	bsub
cancel	llhold llcancel	qdel	bstop bkill
info	llclass	qstat -b qstat -f name	bqueues

- LoadLeveler/NQS/LSF 변환

<표 1-18> LoadLeveler/NQS/LSF 변환

option	LoadLevler	NQS	LSF
comment /commands	#@	#QSUB	#BSUB
run-time	wall_clock_limit=hh:mm:ss	-lt hh:mm:ss -l mpp_t=sec	-c hh:mm (Max CPU) -W hh:mm (Max wallclock)
memory		-lm Xmw -lM Ymw	-M Xkb
queue	class=name	-q name	-q name
runCPUs	node=N tasks_per_node=M	-l mpp_p N	-n minpe,maxpe
parallel job	job_type=parallel		
send mail	notification=always (complete, start, never, error)	-mb -me	-B -N
redirect stdout	output=file error=file	-o file -e file -oe	-o file -e file

1.18. Checkpoint/Restart

1.18.1 Checkpoint 기능 개요

Checkpoint 기능은 사용자의 작업 수행 중 적당한 시점에 작업의 정보를 디스크에 저장하여, 프로그램이 완료되지 않은 상태로 중단된 경우에도 다시 처음부터 수행하지 않고 정보가 저장된 상태 이후부터 수행할 수 있도록 해준다. 현재 순차 프로그램과 병렬 프로그램 모두 지원된다.

1.18.2 Checkpoint 준비

MPI 프로그램은 반드시 _r 버전 컴파일 스크립트를 사용해야하며, 아래와 같이 Job Command File에 관련 옵션을 추가해 주어야 한다.

```
#!/usr/bin/ksh
#@ job_type=parallel
#@ class=p_normal
#@ resources=ConsumableCpus(1) ConsumableMemory(100mb)
#@ total_tasks=32
#@ blocking=unlimited
#@ restart_on_same_nodes=no
#@ checkpoint=interval or yes
#@ restart=yes
#@ restart_from_ckpt=no
#@ ckpt_dir=/사용자 작업 디렉토리 or /ckpt_restart[노드별 로컬 checkpoint 전용 디렉토리]
#@ ckpt_file=mpi_pi.ckpt
#@ error=mpi_pi.err
#@ output=mpi_pi.out
#@ Network.MPI=csss,shared,US
#@ queue
cd /user/user01/PI
export MP_SHARED_MEMORY=yes
./mpi_pi_send > mpi_pi.$$ 2>&1
```

1.18.3 Checkpoint 수행

- 수동 checkpoint

사용자가 원하는 시기에 'llckpt -옵션 jobstep_id' 명령어를 사용하여 checkpoint 시킬 수 있다. Job Command File에서 #@ checkpoint=yes 또는 interval 옵션을 설정해야 한다.

```
$ llckpt nobel.job01.0     → Checkpoint를 완료하고 계속 프로그램 수행
$ llckpt -r nobel.job02.0 → Checkpoint 완료를 기다리지 않고 프로그램 계속 수행
$ llckpt -u nobel.job03.0 → Checkpoint를 완료하고 대기열로 전환
$ llckpt -k nobel.job04.0 → Checkpoint를 완료하고 작업 중단
```

- 자동 checkpoint 수행

LoadLeveler에 의해서 정해진 시간(현재 2시간)을 주기로 자동 체크포인트 된다. Job Command File에서 #@ checkpoint=interval 옵션을 설정해야 한다.

1.18.4 Restart 방법

Job command file에서 '#@ restart_from_checkpoint=yes'를 지정해주고 다시 작업을 제출해야 한다. 아래는 Job command file의 일부 예이다:

```
#!/usr/bin/ksh
#@ job_type=parallel
#@ class=p_normal
#@ resources=ConsumableCpus(1) ConsumableMemory(100mb)
#@ total_tasks=32
#@ blocking=unlimited
#@ restart_on_same_nodes=no or yes     # ckpt_dir을 /ckpt_restart로 지정한
                                          경우에는 반드시 yes로 설정해야 한다.
#@ checkpoint=interval or yes
#@ restart=yes
#@ restart_from_ckpt=yes
#@ ckpt_dir=/사용자 작업 디렉토리 or /ckpt_restart
#@ ckpt_file=mpi_pi.ckpt
#@ error=mpi_pi.err
#@ output=mpi_pi.out
#@ Network.MPI=csss,shared,US
#@ queue
cd /user/user01/PI
export MP_SHARED_MEMORY=yes
./mpi_pi_send > mpi_pi.$$ 2>&1
```

Note

작업이 정상적으로 restart 되는지 확인하는 방법 한 가지

1) 작업이 checkpoint 후 종료되면 기존 출력 파일을 새로운 이름으로 변경 저장하고, 기존 출력 파일과 동일한 빈 파일을 새롭게 생성한다.
   # mv mpi_pi.out mpi_pi.out .org
   # touch mpi_pi.out
2) 작업을 restart 시킨 후, mpi_pi.out .org 파일의 작업 결과 이후로 mpi_pi.out 파일에 작업 결과가 이어서 생성되는 지 비교해 본다.

2. NEC

2.1. 도입 개요

○ 1993년에 도입 설치되어 2001년 5월까지 서비스 해온 Cray C90 시스템의 대체 기종으로 NEC 시스템과 IBM 시스템이 선정되었으며, 이들 시스템들은 각각 1, 2차로 나뉘어 도입됨.

○ NEC 시스템은 main 시스템이 설치되기 전에 pilot 시스템으로서 NEC SX-4 시스템을 이미 2001년 5월부터 사용자에게 무상으로 서비스하여 Cray C90 시스템의 철수에 따른 준비로서 사용자 불편을 최소화하였으며, main 시스템은 1차로 NEC SX-5/8B 시스템(80Gflops)을 2001년 6월 중순에 도입하였고, 2차로 NEC SX-6 시스템(160Gflops)을 2003년 2월에 도입하였음

○ 2001년 8월부터 사용자 서비스를 개시한 NEC SX-5/8B 시스템은 8개의 CPU로 구성되어 있으며, 최대 전체 이론성능은 80Gflops(CPU 당 이론성능은 10Gflops), 메모리는 128GB, 디스크 용량은 2.85TB로서, 전체 이론성능은 Cray C90 시스템에 비해 최대 5배(CPU 당 이론성능은 10배), 메모리는 32배, 디스크 용량은 14배나 확대되었기 때문에 대규모 프로그램 실행이 가능함.

○ 1차 시스템인 SX-5에 이어 2차 시스템인 SX-6는 1차 시스템과 성능이 비슷한 시스템으로서 2노드가 추가 도입됨에 따라 NEC 시스템은 전체 3노드로 구성되고 전체 이론 성능은 240Gflops, 전체 메모리는 256GB, 전체 디스크 크기는 5.7TB임

2.2. 시스템 구성

2.2.1. 시스템 사양

○ NEC SX-5/8B 시스템의 운영체제는 <표 2-1>에 기술되어 있는 것처럼, SUPER-UX로서 현재 버전은 R13.1이며, SX-5 시스템은 단일 노드로 SX-6 시스템은 두개의 노드가 IXS로 연결되어 있음

<표 2-1> NEC SX-5/8B 시스템 하드웨어 내역

구분	내역	
	SX-5	SX-6
O.S명	SUPER-UX R13.1	SUPER-UX R13.1
CPU 수	8개	16개
최대 이론성능	80Gflops	160Gflops
메모리 용량	128GB	128GB
디스크 용량	2.85TB	2.85TB

○ NEC SX-5/8B 시스템의 프로그래밍 환경은 <표 2-2>에 기술되어 있는 것처럼, C, C++, FORTRAN을 기본적으로 제공하고 컴파일러는 C 및 C++ 컴파일러는 각각 C/SX와 C++/SX FORTRAN은 FORTRAN90/SX가 제공되고. 병렬화 지원도구로는 MPI와 OpenMP가 지원됨. 또한, 사용자 작업의 원활하고 효율적인 자원배분을 위하여 NQS가 제공되고 있음

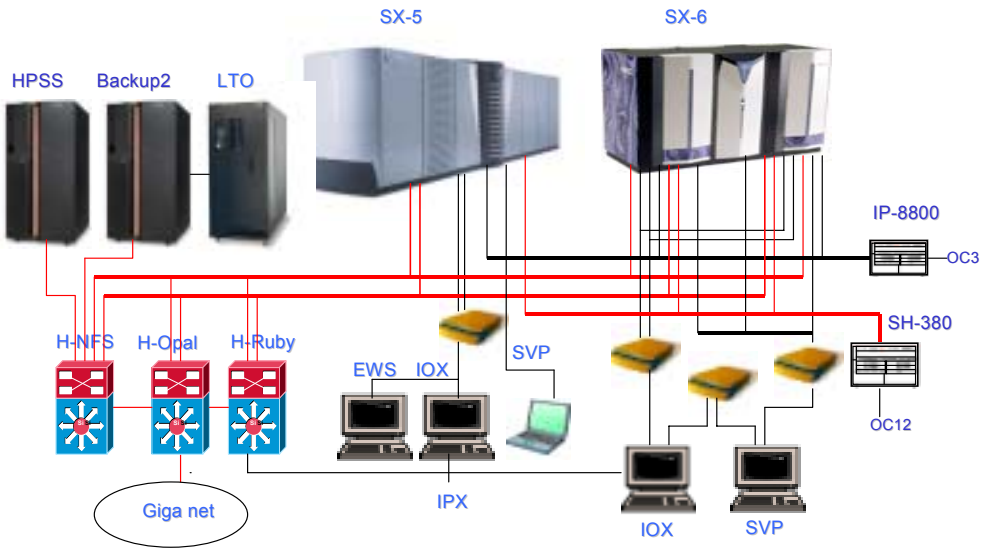
<표 2-2> NEC SX-5/8B 시스템 소프트웨어 내역

구분	내역
운영체제	SUPER-UX13.1
Batch Job Scheduler	NQS
컴파일러	C/SX, C++/SX, Fortran77/SX, Fortran90/SX, HPF
병렬화 도구	MPI/SX, OpenMP Fortran
디버거	PDBX
Math 라이브러리	NAG, IMSL, SCALAPACK, LAPACK

2.2.2 네트워크 구성

○ 사용자는 1Gbps Gigabit 또는 100Mbps Fast Ethernet을 통하여 NEC SX-5/8B 시스템에 접속이 가능하며, NEC SX-5와 SX-6 시스템 간에는 1Gbps Gigabit로 이중으로 연결되어 있고 SX-6의 두 노드 간에는 Internode Crossbar Switch를 통하여 8Gbps Network으로 연결됨 .

NEC Network Configuration



<그림 2-1> NEC Network Configuration

2.3. 사용자 환경

2.3.1. Login

○ 사용자는 자신의 local terminal에서 '*telnet hostname*' 명령어를 이용하여 NEC SX-5/8B 시스템에 login 할 수 있음.

○ NEC 시스템은 모두 SX-5, SX-6의 2node로서 모두 3개의 node로 구성되어 있으며 그 중 SX-5가 login node가 되고 모든 작업은 여기서 준비하여 자신 및 다른 노드에 수행시킴.

```
% ssh -l userid necsx6a.ksc.ne.kr(혹은 ssh -l userid 150.183.5.73)
necsx6a%
```

2.3.2. 패스워드

○ 패스워드를 변경하려면 '*passwd*' 명령어를 이용하며, 패스워드는 2개 이상의 알파벳과1개 이상의 숫자나 특수 문자를 포함하여 적어도 6자 이상으로 구성되어야 하고, 기존의 패스워드에서 3자 이상이 변경되어야 함.

```
necsx6a% passwd
Changing password for user_id
Old password: ********
New password: ********
Re-enter new password: ********
```

2.3.3. 셸

○ 사용자는 POSIX shell:/bin/sh, Korn shell:/bin/ksh 및 C shell:/bin/csh 중의 어느 하나의 셸을 사용할 수 있으며, tcsh:/usr/bin/tcsh, bash:/usr/bin/bash도 사용 가능하나 이 셸은 third part로서 NEC에서 사용할 수 있도록 지원함. 사용 중에 사용자가 셸을 변경하려면 '*chsh*' 명령어를 이용하고 login 시 변경하려면 profile 또는 .login에서 정의하여 주면 됨.

```
necsx6a% chsh
Changing login shell for taro.
Old shell:  /bin/sh
New shell:  csh
```

2.3.4. 터미널 타입 설정s

○ 사용자가 자신의 터미널 타입을 변경하려면 셸에 따라 다음의 명령어를 사용함

```
necsx6a% setenv TERM vt100 (csh일 경우) 혹은
necsx6a# export TERM=vt100 (ksh일 경우)
```

2.3.5. CPU time

○ 사용자의 Interactive session은 CPU time이 1,000초를 초과하면 종료되며, CPU time이 1,000초 보다 많은 시간을 요구하는 작업은 배치로 수행하여야 함

2.3.6. 기타 유틸리티

○ 사용자들의 편의를 위하여 다음과 같은 유틸리티 등이 설치되어 있으며 사용자의 요구가 있으면 필요 시 이를 설치함.

vim, gzip, xemacs

2.4. 파일시스템

○ NEC SX-5/8B 시스템에서는 사용자의 데이터 저장과 프로그램 실행을 지원하기 위해서 디스크 공간에 홈 디렉토리, 스크래치 디렉토리 등으로 구성된 파일시스템을 구축하여 사용자에게 제공하고 있음

2.4.1. 홈 디렉토리

○ 홈 디렉토리는 사용자의 중요한 데이터나 프로그램을 저장하는 디스크 영역이며, 학계 사용자의 경우에는 /edun/*user-name*, 산업계 사용자 경우에는 /inst/*user-name*내부 사용자는 /system/*user-name*이란 디렉토리로 구성됨.

○ Disk는 Quota를 적용하고 1차 시스템인 SX-5에서는 홈 디렉토리에 1구좌 당 1GB까지의 데이터 저장이 가능하였으나 2차 시스템인 SX-6가 추가로 도입됨에 따라 홈 디렉토리에 1구좌 당 6GB의 공간을 제공함. Quota는 디스크 사용량을 제한하는 기능으로, 하드웨어 제한, 소프트웨어 제한 및 시간제한의 3가지가 있음. 2차 시스템부터는 하드웨어 제한은 1구좌 당 6GB, 소프트웨어 제한은 사용자 disk quota의 95%이상, 시간제한은 7일로 세트되어 있음. 소프트웨어 제한을 초과한 사용자가 login하면 경고메시지가 나타나고, 3번 이상 login하게 되면 하드웨어 제한을 초과하지 않은 경우라도 홈 디렉토리에 write를 못함. 마찬가지로, 소프트웨어 제한을 초과한 사용자가 시간 제한(7일)이 경과한 후에 login하면 하드웨어 제한을 초과하지 않은 경우라도 홈 디렉토리에 write를 못함

○ 사용자가 quota가 적용된 홈 디렉토리의 디스크 사용현황을 알아보려면 다음과 같이 quota 명령어를 이용함

```
necsx6a% quota -g user_id
Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
  /edun     963968  996148 1048576           5243     0       0
```

2.4.2. 스크래치 디렉토리

○ xtmp

- 사용자 작업 결과의 임시 저장을 위한 local 디스크 공간으로 557GB이고 SX-5, SX-6a, SX-6b의 각각에 local로서 사용자의 작업을 위하여 제공되고, 백업하지 않음

- 해당 디렉토리의 파일은 사용자 작업 종료 후 2일이 지나면 자동으로 삭제되기 때문에. 사용자는 작업이 끝난 후 2일이 지나기 전에 파일을 다른 곳으로 옮겨야 함

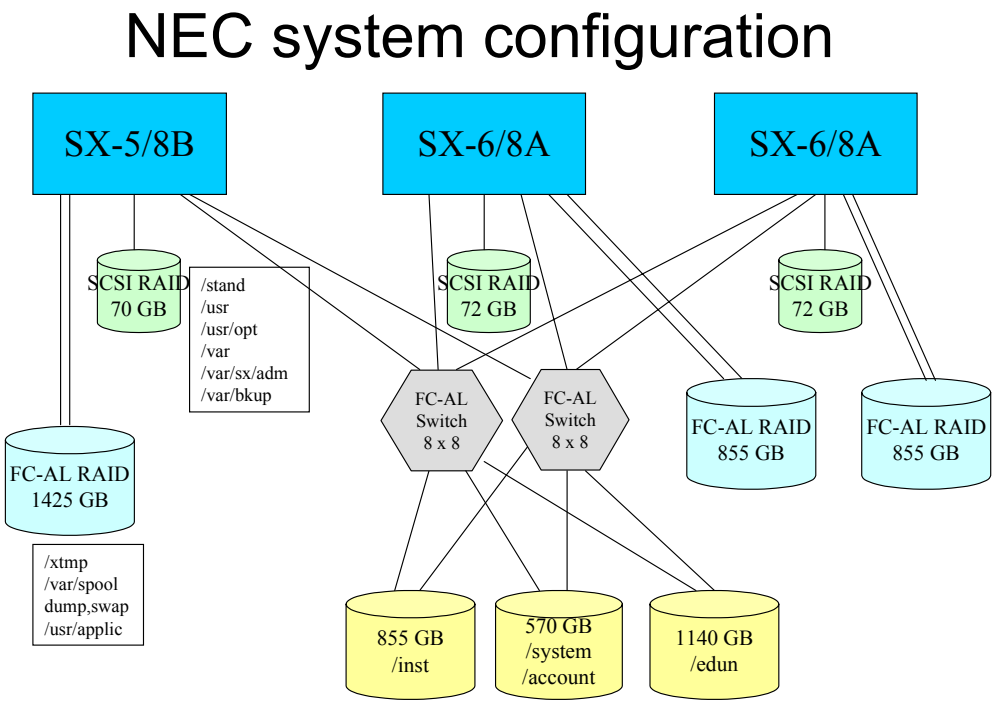
○ ytmp
- 사용자 작업 결과의 임시 저장을 위한 local 디스크 공간으로 347GB이고 SX-6a, 에 local로서 연결되어 있고, SX-5와 SX-6b는 GFS로 마운트되어 있고, 백업하지 않음
- 해당 디렉토리의 파일은 사용자 작업 종료 후 1일이 지나면 자동으로 삭제되기 때문에 사용자는 작업이 끝난 후 2일이 지나기 전에 파일을 다른 곳으로 옮겨야 함. 해당 디렉토리는 다른 노드에서 서로 공유되어 있기 때문에 사용시 부하가 많이 발생하고 속도 저하가 나타날 수 있음.

○ 고속 I/O 디렉토리
- 사용자 프로그램의 고속 I/O를 지원하기 위하여, 메인 메모리의 일부를 할당하여 제공하고 있으며 /work란 이름으로 크기는 4GB 임.
- 이 디렉토리는 크기가 비교적 작기 때문에 사용자가 작업이 끝난 후 파일들은 자신이 즉시 삭제해야 함.

2.4.3. GFS 구성

○ GFS(Global File System)은 NEC에서 제공하는 NFS와 같은 기능으로서 빠른 I/O를 제공하며 주로 데이터가 128KB 이상에서 좋은 I/O 성능을 나타냄.

○ 사용자의 홈디렉토리는 아래의 그림과 같이 /edun. /inst. /system의 3개로 나뉘어 구성되어 있으며 이들은 모두 SX-6a에 연결되어 있고 SX-5, SX-6b에서는 이들의 디스크를 GFS로 공유하고 있음.



<그림 2-2> NEC System Configuration

2.5. HPSS

2.5.1. Tape 저장장치

○ 사용자들의 안전한 자료 저장을 위한 Tape 저장 장치로 360TB를 보유하고 있으며 이 중 사용자 백업용으로 50TB 사용자 Data Migration 용으로 310TB가 할당되어 있으며 사용자에게 200GB/1구좌당을 제공하고 있음.

○ HPSS로 저장된 자료는 다른 시스템에서도 공유가 가능함으로 어느 시스템에서나 필요 시 HPSS에 저장된 데이터를 이용할 수 있음.

2.5.2. 사용법

○ 사용방법은 다음에 예로 보여주고 있으며 상세한 내용은 Web에 공지되어 있음

```
SX-6a [/system/adm/yjlee] $ pftp_client core 4021
Parallel block size set to 8388608.
        PdataSockBufSize reset equal or below sb_max (524288)
Connected to core.
220-
220-**************************************************
220-*                                                *
220-* This is the HPSS 4.3 system running at KISTI  *
220-*                                                *
220-**************************************************
220-
220 core FTP server (HPSS 4.3 PFTPD V1.1.1 Mon Dec 3 14:30:48
KORST 2001) ready.
Name (core:yjlee): yjlee
331 Password required for /.../kisti_cell.hpcnet.ne.kr/yjlee.
Password:
230 User /.../kisti_cell.hpcnet.ne.kr/yjlee logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
**** NOTE: Server supports Parallel Features     ****
****        Auto-Parallel Substitution Enabled.  ****
**** NOTE: Protocol set to PDATA_AND_MOVER       ****
Multinode is Disabled.
ftp>
```

2.6. Cross compiler

○ 크로스 컴파일러란 컴파일러가 실행되는 시스템과 다른 아키텍처를 가진 시스템을 위한 실행파일을 생성해내는 컴파일러를 말하는 것으로서 KISTI 슈퍼컴퓨팅센터의 경우 벡터형 시스템인 NEC SX-5 및 SX-6에서 컴파일 속도가 느린 관계로 이들 시스템에서의 실행파일을 생성하기 위한 리눅스 기반 시스템을 따로 제공하고 있다.

2.6.1. 구성

○ Cross compiler의 hostname은 neccross이고, 해당 시스템에 별도로 접속할 필요는 없다.

○ cross compiler sever는 SX-6a에 NFS로 연결되어 있으며 cross compler server에서 compile 된 프로그램은 SX-5, SX-6에서 모두 사용할 수 있음.

2.6.2. 사용법

○ 크로스 컴파일러의 사용법은 NEC 시스템의 어느 노드에서나 일반 컴파일 하는 것처럼 크로스 컴파일 명령어를 사용하여 컴파일 하면 된다. 이렇 경우 자동적으로 해당 프로그램이 크로스 컴파일을 수행한 다음 목적 파일을 생성하여 준다.

○ 크로스 컴파일 명령어

- 컴파일러의 옵션은 동일하며 기존의 컴파일 명령을 다음과 같이 바꿔주면 된다.

f90 → sxf90
cc → sxcc
c++ → sxc++
ld → sxld

○ 성능비교

- 크로스 컴파일 시스템을 사용하는 경우 이 시스템에 대한 사용료가 따로 부과되지 않을 뿐 아니라 아래 표에서와 같이 컴파일 시간도 10배 이상 단축할 수 있다. 특히 컴파일할 소스 파일이 많고 시간이 오래 걸리는 경우 매우 큰 이점이 있다.

Program	Language	Compile time (sec)		ratio
		necsx5	neccross	
Couple	FORTRAN	85	7	12.1
YONU AGCM	FORTRAN	1465	116	12.6
QCD_vector	FORTRAN	103	7	14.7
FEM_LSA	C++	975	60	16.3

2.7. 프로그래밍

2.7.1. 컴파일러 옵션

○ NEC SX-5/8B 시스템의 FORTRAN 컴파일러인 FORTRAN90/SX 컴파일러를 수행하는 명령은 'f90'으로 Cray C90의 FORTRAN 컴파일러(CF90)와 동일하며 FORTRAN90/SX의 컴파일러 옵션은 CF90과 거의 비슷한 개수가 있음. 잘 쓰이지 않는 옵션의 경우 각 컴파일러의 매뉴얼을 비교해서 비슷한 기능을 하는 것을 찾아낼 수밖에 없는데 이 경우는 NEC 기술지원 인력의 지원을 받을 수 있

음. 그러나 그렇지 않은 대부분의 경우는 사용자가 직접 간단한 수정만으로 컴파일해서 바로 실행이 되도록 할 수 있으며 이때 참고해야 할 사항들은 다음과 같음

1) 표준 컴파일 옵션

-c, -D, -g, -I, -l, -L, -o, -O 등과 같은 표준 컴파일 옵션은 FORTRAN90/SX에서 그대로 사용할 수 있음

2)「-dp」옵션(Disable Double Precision Arithmetic)

CF90에서 가장 흔히 쓰이는 옵션 중 하나로 FORTRAN90/SX에서는「-ew」와 같은 역할을 하며,「-dp」옵션을 사용하지 않고 컴파일 한 경우(Cray의 default data size를 사용한 경우)는 FORTRAN90/SX에서 컴파일 할 때「-Wf "-w double16"」옵션(빈칸에 주의)을 추가해야 함. 결론적으로 Cray 시스템에서 명시적으로 DOUBLE PRECISION을 선언하여 사용한 경우는「-Wf "-w double16"」옵션을, 그렇지 않은 경우는「-ew」옵션을 추가해야만 같은 정확도의 결과를 기대할 수 있음.

컴파일러 옵션과 각 데이터 형의 크기의 관계(Default 크기에 유의)

Compiler	Cray		NEC		
Option	(default)	-dp	(default)	-ew	-Wf"-wdouble16"
INTEGER	8byte	8byte	4byte	8byte	8byte
REAL	8byte	8byte	4byte	8byte	8byte
DOUBLE	16byte	8byte	8byte	8byte	16byte

3) Loader

segldr .... -l lib1 -l lib2

Object 파일의 링크 시에 사용되는 CF90의 Loader인 'segldr'은 일반적인 Loader 프로그램인 'ld'와 달리 라이브러리를 순환적으로 검색하여 필요한 루틴을 찾는데 이 때문에 'segldr'을 사용하여 프로그램을 링크하는 경우 사용되는 라이브러리는 순서에 상관없이 나열할 수 있음. 이 기능을 쓰지 않는 대부분의 경우는 일반적인 Loader인 'ld'를 그대로 사용해도 무방하나 라이브러리의 순환적 검색이 필요한 경우는 'ld' 명령에「-h lib_cyclic」이라는 옵션을 추가해야 함. 대부분의 경우 컴파일, 링크 모두 'f90' 명령 하나만으로 충분한 경우가 많

이 있으나 'f90' 명령을 통해 Loader로 「-h lib_cyclic」옵션을 보내고자 하는 경우는 -Wl 옵션 이용

예) necsx6a% f90 -Wl "-h lib_cyclic" .....

4) 성능 최적화 옵션

성능 최적화를 위해 CF90에서는 -Oxxxx」와 같은 형태의 옵션을 사용하나, FORTRAN90/SX에서는 다른 형태를 사용함. FORTRAN90/SX에서 사용되는 최적화 옵션들은 매우 다양하기 때문에 이것들을 일일이 조절하기보다는 컴파일 모드를 지정해서 거기에 맞게 최적화 옵션들이 적절히 설정되도록 해주게됨. 최적화 관련된 컴파일 모드를 지정하는 옵션은 단일 프로세서를 효율적으로 활용해 성능을 높이기 위한 벡터화 관련 옵션과 둘 이상의 프로세서를 사용함으로써 성능을 높이는 병렬화 관련 옵션이 있음

- 벡터화 관련 옵션

-C [hopt,vopt,sopt,vsafe,ssafe,debug]

Compiler Option	Sub Option	의 미	관련 CF90 옵션
-C	hopt	최적화(optimization)와 벡터화를 최대로 수행	-O*n* -Ovector*n* -Oscalar*n* (*n*=0,1,2,3)
	vopt(default)	최적화 및 벡터화 보통 정도로 수행	
	sopt	보통 정도의 스칼라 코드 최적화만 수행	
	vsafe	안전한 최적화 및 벡터화만을 수행	
	ssafe	안전한 스칼라 코드 최적화만 수행	
	debug	최적화 및 벡터화를 수행하지 않고 디버깅이 가능하도록 컴파일	

예) necsx6a% f90 -C vsafe prog.f

- 병렬화 관련 옵션

-P [auto,openmp,static]

Compiler Option	Sub Option	의 미	관련 CF90 옵션
-P	auto	컴파일러에 의한 자동병렬화	-O*n* -Otask*n* (*n*=0,1,2,3)
	openmp	OpenMP 디렉티브를 이용한 병렬화	
	static(default)	병렬화하지 않음	

예) necsx6a% f90 -P auto prog.f

2.7.2. 내장 서브루틴 및 라이브러리

1) CRAY Scientific Library(SCILIB)

○ NEC SX-5/8B시스템에서 제공하는 MathKeisan이라는 라이브러리가 SCILIB과 거의 유사한 기능을 제공하며, 특히 FFT와 같은 주요 루틴의 인터페이스는 동일하므로 소스 프로그램의 수정 없이도 컴파일이 가능함. MathKeisan에는 고도로 최적화된 BLAS 루틴도 포함되어 있으며, 또한 SCILIB과 마찬가지로 LAPACK이 포함되어 있음. 따라서 BLAS나 LAPACK을 쓰는 경우에도 소스 프로그램의 수정 없이 바로 사용 가능하나 SCILIB의 서브루틴 중 MathKeisan 등에서 지원되지 않는 것이 있으므로 이 경우 직접 기술지원을 받아야 함.

※ Cray SCILIB의 FFT 루틴을 사용하는 경우 컴파일 시「-lfft」혹은「-lfft_64」옵션 추가. BLAS 및 LAPACK을 사용하는 경우「-lblas」,「-llapack」또는 64bit 모드의 경우「-lblas_64」, 「-llapack_64」옵션 추가

2) 내장 서브루틴 및 함수

○ CF90에서 FORTRAN90/SX로의 프로그램 이식 시 다음 표에 나열된 것을 포함한 몇몇 내장 서브루틴들은 해당하는 FORTRAN90/SX의 내장 서브루틴으로 수정해 주어야 함

CF90 Intrinsic	SX Meaning	FORTRAN90/SX Intrinstic
t1=second()	Get CPU time	call clock(t1)
call rtc call timef	Get elapsed time	call etime
call date call time	Get current date and time	call datim(a,b[,c])
Shiftr	Logically shifts a1 by a2 to the right.	irl(a1,a2)
Shiftl	Logically shifts a1 by a2 to the left	ils(a1,a2)
HPALLOC and HPDEALLOC	Allocates necessary area size i. This function is for word pointer mode.	Mallocw() with -ew option

3) NAG 및 IMSL 라이브러리 사용법

○ NAG 사용법 : 링크 시 '-lnag' 옵션 추가

○ IMSL 사용법

① 먼저 다음 shell scipt를 수행해서 필요한 환경변수를 설정함

csh의 경우: $ source /usr/applic/vni/CTT3.0/ctt/bin/cttsetup.csh
sh의 경우: $ /usr/applic/vni/CTT3.0/ctt/bin/cttsetup.sh

② 컴파일 및 링크 시 설정된 환경 변수를 command line에 추가
예1) $ f90 $F90FLAGS -o imsltest imsltest.f $LINK_F90
예2) $ f90 $F90FLAGS -c imsltest.f
$ f90 -o imsltest imsltest.o $LINK_F90

2.7.3. 환경변수

○ NEC 시스템에서 프로그램 실행과 관련된 주요 환경변수로는 다음과 같은 것들이 있음

1) F_RSVTASK

Cray C90에서의 'NCPUS'라는 환경변수와 같은 의미로 프로그램 수행 시 사용될 microtask(또는 thread)의 수를 지정함. 병렬화 옵션을 주고 컴파일 한 프로그램의 경우에만 적용이 되며, 다음과 같이 지정함(C shell 기준)

setenv F_RSVTASK 4

2) TMPDIR

Cray C90의 'TMPDIR'과 같은 의미로 임시 파일(작업 중간에 임시로 생성된 파일)이 생성될 디스크 공간(디렉토리)을 지정함

setenv TMPDIR /xtmp

3) F_PROGINF

프로그램 수행 시간, 수행 성능(MFLOPS), 평균 Vector length, 사용 메모리 등 프로그램 수행과 관련된 정보들을 화면으로 출력할지 여부를 지정함. Main 프로그램이 C 언어(또는 C++)로 된 경우 F_PROGINF 대신 C_PROGINF 사용.

setenv F_PROGINF {NO | YES | DETAIL}

4) F_SETBUF

입출력 버퍼의 크기를 지정하며, 큰 값을 지정할수록(시스템 자원이 허락하는 한도 내에서) 더 좋은 입출력 성능을 얻을 수 있음

setenv F_SETBUF[u] m
u : Unit number. 생략 시 모든 입출력 Unit에 적용됨
m : 버퍼 크기. KB 단위로 지정

5) F_FF*n*

포트란 입출력 Unit에 파일을 지정하며, Cray의 'assign' 명령어와 같은 역할을 함

예) Cray C90에서 'input'이라는 파일을 포트란 입출력 Unit 10번에 할당하는 경우

```
$ assign -a input fort.10
```

예) NEC SX-5/8B에서 'input'이라는 파일을 포트란 입출력 Unit 10번에 할당하는 경우

```
$ setenv F_FF10 input
```

2.7.4. 바이너리 데이터 변환(Cray C90 시스템 → NEC SX-5/8B 시스템)

○ Cray C90 시스템에서 만들어진 바이너리 데이터를 NEC SX-5/8B 시스템에서 읽기 위해서는 다음과 같은 방법으로 Cray format의 binary 데이터를 IEEE 포맷으로 변환하면 됨

1) Compile a program with "-ew" option.
   ex) f90 -ew read.f

2) Conversion from "Cray binary data file" to "SX5 binary data file" using "cos2f77" command.
   ex) cos2f77 cray.dat nec.dat

3) Set run-time options, F_UFMTFLOAT2 and F_UFMTADJUST(Unit number of the file 'nec.dat' is assumed to be 1)

```
ex) setenv F_FF1 nec.dat
    setenv F_UFMTFLOAT2 1
    setenv F_UFMTADJUST NO
```

4) Execute.
   데이터를 읽을 프로그램은 -ew 옵션으로 컴파일 되어야 하고 읽을 데이터는 cos2f77 프로그램으로 변환해 주어야 한다.

2.7.5. Namelist I/O

○ Namelist I/O를 사용하는 경우 Cray 시스템에서 표준 포맷이 아닌 형태로 파일을 만드는 수가 있기 때문에 다음과 같은 표준 포맷으로 그 내용을 수정할 필요가 있음

```
&variable group name ..................... /
```

예) Cray에서 만들어진 Namelist I/O 파일이 다음과 같은 경우

```
$bcinp itype=4,iblk1=1,ist1=  1, 2, 2,imd1=  1, 2,33,ien1=  1,33,33,
iblk2=1,ist2=130, 2, 2,imd2=130, 2,33,ien2=130,33,33 $
$bcinp itype=1,iblk1=1,ist1=  2, 1, 2,imd1=129, 1, 2,ien1=129, 1,18 $
$bcinp itype=4,iblk1=1,ist1=  2, 1,19,imd1= 65, 1,19,ien1= 65, 1,33,
iblk2=1,ist2=129, 1,19,imd2= 66, 1,19,ien2= 66, 1,33 $
```

다음과 같이 바꿔 줌

```
&bcinp itype=4,iblk1=1,ist1=  1, 2, 2,imd1=  1, 2,33,ien1=  1,33,33,
iblk2=1,ist2=130, 2, 2,imd2=130, 2,33,ien2=130,33,33 /
&bcinp itype=1,iblk1=1,ist1=  2, 1, 2,imd1=129, 1, 2,ien1=129, 1,18 /
&bcinp itype=4,iblk1=1,ist1=  2, 1,19,imd1= 65, 1,19,ien1= 65, 1,33,
iblk2=1,ist2=129, 1,19,imd2= 66, 1,19,ien2= 66, 1,33 /
```

2.7.6. 프로그램의 memory limit과 CPU time 예측

○ 'size' 명령어를 사용하여 정확하지는 않지만 소요되는 대략적인 메모리 양을 산출하는 것이 가능함. 병렬 프로그램의 경우에는 static 영역과 stack 영역을 포함할 수 있음. stack 영역을 보려면 -l 옵션을 사용하며, 크기는 byte로 나타남

- 사용법
  size [-l CPU 수] 실행파일 이름
- 사용 예
  necsx6a% size -l 4 a.out
  42856 + 158744 + 417760 + 89798152 * 4 =360711968

○ 실행시간을 조사하기 위하여 먼저 소규모 프로그램을 실행시킨 후에 좀 더 큰 프로그램을 실행시킴. 문제의 크기(예: grid와 time steps의 수)를 기반으로 하여 실행시킬 프로그램이 얼마나 시간이 소요될 것인가를 예측할 수 있음

2.8. 배치작업 환경

2.8.1. 개요

○ NEC SX-5/8B 시스템에서는 사용자 작업(job 혹은 request라고도 함)의 배치처리를 위하여 NQS(Network Queuing System)를 지원하고 있으며, NQS에 의해 지원되는 주요 특성은 다음과 같음

- 대규모 프로그램 실행
  Interactive session에서 처리할 수 없는 대규모 프로그램(많은 CPU time을

요하는 프로그램)을 실행시킴

- Checkpoint restart
  시스템이 계획적이거나 혹은 일시적인 장애에 의하여 다운되었을 때, 실행 프로그램은 저장되고, 시스템이 reboot 되었을 때, 그 프로그램은 정지된 시점부터 다시 실행됨
- 표준 출력파일과 표준 에러파일을 디렉토리에 생성시킬 수 있음
- Logout 후에도 작업은 계속 실행됨
- 동시에 실행할 수 있는 작업 개수를 제한하고, 작업별 CPU time이나 메모리 제 한 등의 기능을 수행함

○ NEC SX-5/8B 시스템에서 제공되는 queue는 pipe queue와 batch queue가 있음

- pipe queue는 사용자가 보내 온 작업을 CPU time limit을 조사하여 사용자 작업들을 적정의 배치 queues로 보내는 역할을 함
- batch queue는 pipe queue에서 보내 온 작업이나 사용자가 직접 보내 온 작업들이 CPU 자원을 기다리면서 머무는 곳임

1) Batch queue 설명

가) 큐의 종류

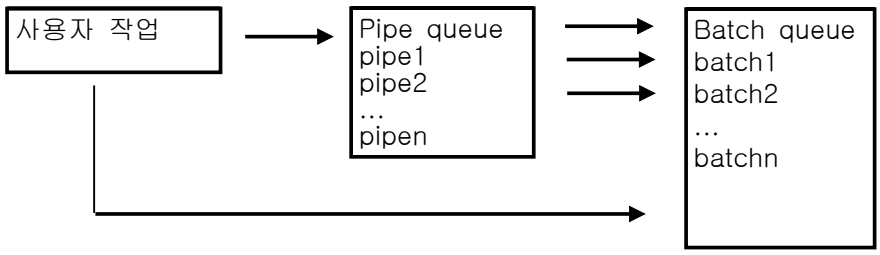
○ Pipe queue

사용자가 보낸 작업을 우선 저장하였다가 queue의 정책과 작업의 특징에 따라서 적당한 Batch queue로 할당함

○ Batch queue

사용자의 작업을 처리하는 queue

아래의 <그림 2-3>에서 보는 바와 같이 사용자는 작업을 pipe queue 또는 batch queue로 직접 submit 할 수 있음. pipe queue로 보낸 작업은 해당 pipe queue에서 작업의 특성에 따라 적당한 batch queue로 할당됨



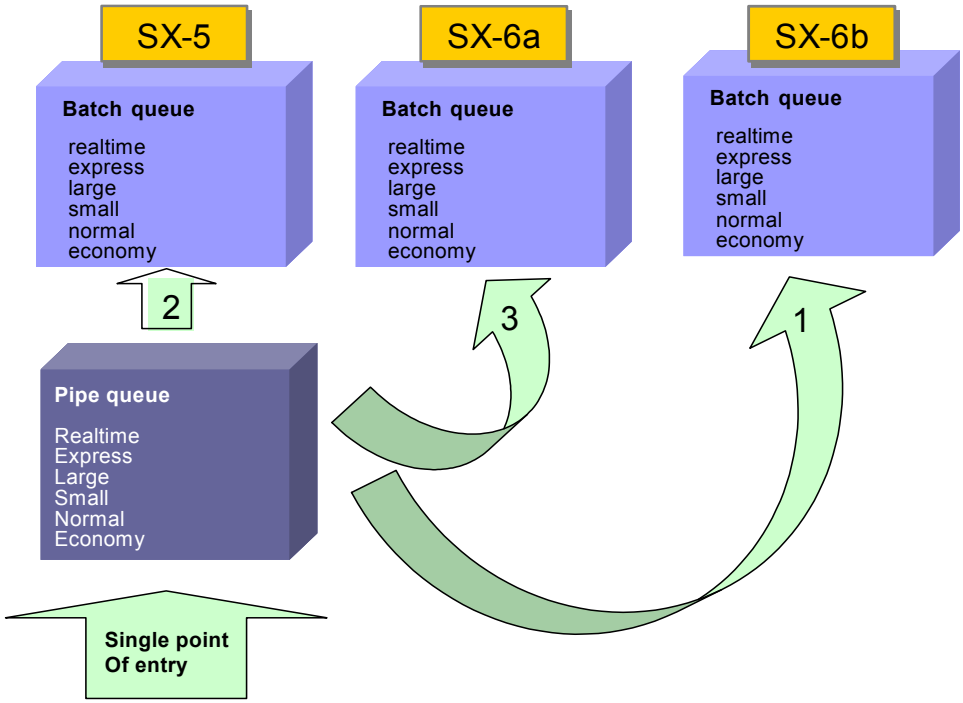
<그림 2-3> NQS 구조

2.8.2. NQS 시스템 구성

○ SX-5, SX-6a, SX-6b의 각각 시스템에 pipe queue와 batch queue가 있으며, 작업을 보낼 때는 원칙적으로 pipe queue로만 가능하게 하였다. 이는 pipe queue에서 작업을 각각의 노드로 load balancing하기 위함이다.

○ Pipe queue에서 batch queue로의 job 분배는 DDM(Demamd Delivery Methord)방식으로서 사용자가 pipe queue를 통하여 작업을 보내면 그 작업은 pipe queue에 임시로 저장되어 있다가 그 pipe queue에 속한 batch queue 중에서 비어 있는 queue를 찾아서 할당됨.

○ 작업을 NQS로 보낼 때 환경 변수 등으로 인하여 에러가 발생하면 디폴트 큐인 small에서 수행됩니다. 작업이 도중에서 중단되었다면 먼저 small 큐에서 수행되었는지 확인하여 보고, 다른 큐를 이용하시던지 아니면 장애가 발생하여 small 큐에서 수행되었는지 확인해야 합니다.



2.8.3. NQS queues 특성

○ 현재 NEC 시스템에서의 NQS queue는 다음 표에서 보인 것처럼 서비스 수준에 의해 작업의 실행 우선을 정의하는 Service Charging Factor에 따라 dedicated, realtime, express, normal 및 economy의 5개의 queue로 구분되며, Dedicated, Realtime, Express, Small, Large, Normal, Economy는 pipe queue이고, dedicated, realtime, express, small, large, normal, economy는 batch queue이다. normal queue는 어느 특정 노드를 지정하여 작업할 수 있는 큐임. Service Charging Factor에 따라 요금이 다르게 부과됨. 즉, realtime queue의 작업들은 normal queue의 작업에 비해 2배의 요금이 부과됨. 모든 queue에서 메모리 제한은 두지 않고 있음

<표 2-3> SX-5 NQS queue table

queue name	CPU time	Mem	Service Charging Factor	Job limit			complex	global
				run	group	user		
dedicated	unlimited		2	∞	∞	∞		13
realtime	unlimited		2	4	2	2		
express	unlimited		1.5	4	2	2		
large	unlimited		1	6	4	4	(10 6 6)	
small	180 min		1	6	4	4		
normal	unlimited	unlimited	1	6	4	4		
economy	unlimited		0.5	6	4	4		

<표 2-4> SX-6a NQS queue table

queue name	CPU time	Mem	Service Charging Factor	Job limit			complex	global
				run	group	user		
dedicated	unlimited		2	∞	∞	∞		13
realtime	unlimited		2	3	2	2		
express	unlimited		1.5	3	2	2		
large	unlimited		1	4	2	2	(10 6 6)	
small	180 min		1	4	2	2		
normal	unlimited	unlimited	1	6	4	4		
economy	unlimited		0.5	4	2	2		

<표 2-5> SX-6b NQS queue table

queue name	CPU time	Mem	Service Charging Factor	Job limit			complex	global
				run	group	user		
dedicated	unlimited		2	∞	∞	∞		13
realtime	unlimited		2	1	1	1		
express	unlimited		1.5	1	1	1		
large	unlimited		1	3	2	1	(8 6 6)	
small	180 min		1	2	2	1		
normal	unlimited	unlimited	1	6	4	4		
economy	unlimited		0.5	2	2	1		

- dedicated queue는 벤치마크 시험용으로 제공되며 예약을 받아서 이 queue에 있는 작업들은 정해진 시간대(주로 금요일 저녁부터 월요일 새벽 시간대)에 실행되고, CPU 수는 1개부터 최대 8개까지 할당될 수 있음
- realtime queue는 거의 실시간이 요청되는 작업을 위한 것이며. 4개의 CPU를 전용으로 할당하되, 이 queue에 작업이 없는 경우에는 다른 queue의 작업들이 이 CPUs를 사용하며 queue에 작업이 대기할 수 없음.
- express queue는 긴급을 요하는 작업을 위한 queue임
- 일반 큐는 small과 large queue가 있으며, CPU time이 180분 이하의 작업 경우에는 small queue를 이용함
- economy queue에 있는 작업들은 CPU가 idle 상태인 경우에만 실행됨

2.8.4. 배치작업 관리

1) 배치작업 스크립트 예

○ 배치작업을 실행시키기 위해서는 UNIX 명령어를 포함한 셀 스크립트에 NQS 옵션이 첨가된 셀 스크립트를 준비해야 하며, 다음은 컴파일을 하는 셀 스크립트(comp.bat)의 예임

○ 배치 스크립트를 작성할 때는 queue name만 지정하면 되고 job에 대한 CPU time을 적을 때에는 queue name과 대치되는지 확인 해야함.

○ pipe queue를 이용하지 않고 직접 batch queue를 이용하고자할 때는 queue name은 pipe queue name를 destination은 host에 대한 batch queue name을 적어줌.

```
#@$-s /bin/csh : Executed in a C shell environment
#@$-r job_name : Job Request Name
#@$-q Large : Pipe Queue Name
#@$-ds : batch_queue_name@host_name 특정 host의 batch queue를 지절 할 때
#@$-lT 1:45:00 : Job에 대한 시간을 지정하고자 할 때
#@$-eo : 표준 출력과 표준 에러 파일을 단일 파일로 보냄
#@$ : NQS 옵션의 끝
#
cd $HOME/src
f90 prog.f
```

- NQS 옵션은 #@$으로 시작되며, 처음에 #가 나타나기 때문에 코멘트로 인식됨
- '#@$'과 '-' 사이는 띄우면 안됨(띄우면 NQS 옵션의 끝으로 인식)
- 상기 셸 스크립트를 queue로 보내기 위해서는 다음과 같이 'qsub' 명령어를 사용하며, queue로 보내진 각 작업은 고유한 request ID가 할당됨. request ID는 시스템으로 보내어진 순서에 따른 번호와 hostname으로 이루어 짐

```
necsx6a% qsub comp.bat
Request 34.necsx6a submitted to queue: small
```

2) 배치작업 실행 옵션

○ 배치작업을 queue로 보내 실행시키기 위해서는 'qsub' 명령어를 사용하며, 이 명령어는 다음과 같은 여러 가지 옵션이 있음

qsub [option] [script-file]

- script-file이 명시되어 있지 않으면 배치작업으로 실행되는 명령어는 표준 입력(stdin)으로부터 받아들이며, 옵션은 다음과 같음
  - -a date-time: 작업을 지정된 date-time까지는 실행시키지 않음
  - -c CPU-count: 작업에서 이용할 CPU수 명시
  - -eo [path/]file 입출력파일이 저장될 파일 이름 명시
  - -lT CPU-time 작업 당 최대 CPU time 명시
  - -lt CPU-time 프로세스 당 최대 CPU time 명시
  - -mb message 작업의 실행 시작 시에 사용자에게 메일을 보냄
  - -me message 작업의 실행 종료 후에 사용자에게 메일을 보냄

-q queue name 작업을 보내고자 하는 배치 큐 이름 명시
-ds host name 작업을 보내고자 하는 host 이름 명시
-r request name: 작업 이름을 request name으로 지정

3) 배치작업 진행경과 확인

○ 전송된 배치작업의 상태를 알아보기 위해서는 다음과 같이 'qstatr' 명령어를 사용함

```
necsx6a% qstatr
==========================================
NQS (R11.10) BATCH REQUEST HOST: necsx6a
==========================================
REQUEST ID   NAME   OWNER   QUEUE   PRI   NICE   MEMORY   TIME   STT   JID   R
-----------------------------------------------------------------------------
1859.necsxr   go.sh   taro   small   31   0   11199   1.865   RUN   1032   -
-----------------------------------------------------------------------------
 - REQUEST ID        Request ID
 - NAME              Name of the submitted scripts
 - OWNER                     User login name
 - QUEUE                     Name of the submitted queues
 - PRI               Priority
 - NICE              Nice value
 - MEMORY            Available memory
 - TIME              CPU time
 - STT               Status
 - JID               Job ID
 - R                 Restart
```

○ 상기에서 STT는 다음과 같이 작업 상태를 나타냄

- RUN : The request is being executed
- QUE : The request is queued for execution.
- WAT : The request waits until execution is started.
- HLD : The request is held.
- SUS : The request execution is temporarily interrupted.
- ARI : The request is being received from the pipe queue.
- RUT : The request is being transmitted from the pipe queue.
- EXT : The request's output file(stdout, stderr, JOR) is being sent to the specified host.

4) 배치작업 queue 상태 확인

○ 얼마나 많은 작업들이 queue에 있거나 실행되는가를 알기 위해서 다음과 같이 'qstatq -b' 명령어를 사용함

```
necsx6a% qstatq -b
==================================================
NQS (R11.10) BATCH QUEUE SUMMARY HOST: necsx6a
==================================================
QUEUE NAME  ENA   STS   PRI/BPR/  TMS /MPR RLM TOT  QUE RUN WAI HLD SUS ARR EXT
-----------------------------------------------------------------------------
small       ENA   RUN   30/ 85/   1000/ 0   8   4    0   4   0   0   0   0   0
large       ENA   RUN   30/ 85/   1000/ 0   8   5    1   4   0   0   0   0   0
express     ENA   RUN   30/ 70/    500/ 0   4   7    5   2   0   0   0   0   0
realtime    ENA   RUN   30/ 60/   1000/ 0   1   0    0   0   0   0   0   0   0
-----------------------------------------------------------------------------
<TOTAL>                                    21  16    6  10   0   0   0   0   0
-----------------------------------------------------------------------------
```

○ queue 상태를 확인해보기 위해서는 두 가지 특성을 조사해 보아야 함. 첫 번째 특성은 queue가 작업을 받아들일 수 있는 가를 조사하는 것이고, 두 번째 특성은 queue에서 작업이 실행될 수 있는가를 조사하는 것임

- queue의 작업수용 능력 조사는 ENA 컬럼에서 나타나는 내용으로 판가름 함

ENA 컬럼	queue 상태	작업 수용능력
ENA	ENABLED	가 능
DIS	DISABLED	불가능
CLS	CLOSED	불가능

▶ CLOSED State

작업이 queue에 의해 받아들일 수 있으나, NQS는 정지되어 있고 작업들이 queue에 쌓이지는 않음

▶ ENABLED State

queue가 작업을 받아들일 수 있으며 NQS가 동작 중임

▶ DISABLED State

queue가 작업을 받아들일 수 없음

- queue에서 작업 실행여부는 STS 컬럼에서 나타나는 내용으로 판가름 함

STS 컬럼	queue 상태	작업 실행여부
STP	STOPED/STOPPING	실행 안됨
RUN	RUNNING	실행됨
INA	INACTIVE	실행됨
SHT	SHUTDOWN	실행 안됨

▶ STOPPED State
실행되지 않은 상태에서 queue에 있는 작업들은 실행되지 않으며, 현재 queue에 있는 어떤 작업도 실행되지 않고 있음

▶ STOPPING State
실행되지 않은 상태에서 queue에 있는 작업들은 실행되지 않으나, 현재 queue에 있는 작업은 적어도 하나가 실행되고 있음

▶ RUNNING State
queue에 있는 작업들은 실행준비가 되어 있으며, queue에 있는 한 개 이상의 작업이 실행되고 있음

▶ INACTIVE State
queue에 있는 작업들은 실행준비가 되어 있으나, 현재 실행되고 있지는 않음

▶ SHUTDOWN State
NQS가 정지되어 있어서 작업들이 실행되지 않음

○ queue에 보낸 모든 작업들의 상태를 알아보기 위하여 'qstatr -ab' 명령어를 이용하거나, 사용 편의를 위하여 스크립트로 만들어 놓은 qra나 qr을 사용하면 됨.

```
necsx6a% qra
necsx6a% qstatr -ab

==========================================
NQS (R09.20) BATCH REQUEST HOST: vector
==========================================
REQUEST ID    NAME     QUEUE   PRI  NICE  MEMORY  TIME  STT  JID   R
------------------------------------------------------------------
1852.vector   spce_60* small   31   0     6083    880   RUN  1025  -
1857.vector   spce_40* small   31   0     6083    282   RUN  1030  -
```

1858.vector	spce_70*	small	31	0	6083	271	RUN	1031	-
1859.vector	go.sh	small	31	0	11199	1.865	RUN	1032	-
1779.vector	go4	small	31	0	45693	9561	RUN	993	-
1776.vector	go3	small	31	0	45693	5139	RUN	1006	-
1840.vector	jobb.sh	small	31	0	23773	3433	RUN	1014	-
1848.vector	jobin.ss	large	31	0	37138	1553	RUN	1022	-
1849.vector	jobin.ss	large	31	0			QUE		-
1310.vector	go5	large	31	0	45708	32185	RUN	924	-
1309.vector	go6	large	31	0	45708	13864	RUN	981	-
1464.vector	check.b*	express	31	0			QUE		-
1562.vector	subtri	express	31	0			QUE		-
1633.vector	go2	express	31	0			QUE		-
1652.vector	go4	express	31	0			QUE		-
1777.vector	go1	economy	31	0			QUE		-

○ queue에서 실행 중인 작업에 대하여 처리 상태에 대한 내용을 알고자 할 때는 다음과 같은 명령어를 사용함

necsx6a% qstatr -f <request_id>

○ queue에서 실행 중인 전체 노드에 대한 자신만의 작업에 대한 상태를 알고 할 때는 다음과 같은 명령어를 사용함

necsx6a% qstatm

```
necsx6a [/system/adm/yjlee/nqs/pgm] $ qstatm
Please wait ......
REQUEST ID        NAME  OWNER  QUEUE  PRI NICE  MEMORY   TIME   STT   JID   R
------------ -------- --------- ------- ---- --- ------  -----  --- ---- --
necsx6a:22.necsx6b test.yj*  yjlee   Small   31   0      -       -    RUT  4415  -
necsx6a:21.necsx6b test.yj*  yjlee   small   31   0    1826M   55.263 RUN   73  -
-------------------------------------------------------------------------------
```

○ queue에서 실행 중인 전체 노드에 대한 작업의 상태를 알고 할 때는 다음과 같은 명령어를 사용함

necsx6a% qstatma

```
REQUEST ID          NAME     OWNER   QUEUE    PRI NICE  MEMORY  TIME  STT  JID  R
----------------    -----    -------  -------  ---- ---  -------  ------ ---  ----- -
necsx6a:1361.necsx5 re140_3d  e739ydh  economy  31   0    238M     4806   RUN   7  -
necsx6a:1453.necsx5 Dth180k*  e737hsh  economy  31   0    1683     1.614  RUN  81  -
necsx6a:1526.necsx5 amip2.c*  e227kis  express  31   0    565M    62794   RUN 145  -
necsx6a:1624.necsx5 g2_movi*  r536juy  express  31   0    105M    12007   RUN 214  -
necsx6a:1629.necsx5 g2_v      r536juy  express  31   0    105M     7969   RUN 221  -
necsx6a:1509.necsx5 CFC       r567pyg  large    31   0    410M    59894   RUN 128  -
necsx6a:1523.necsx5 run_adj*  i002frl  large    31   0    1121M   41091   RUN 142  -
necsx6a:1524.necsx5 run_adj*  i002frl  large    31   0    1121M   40786   RUN 143  -
necsx6a:1535.necsx5 test.82   e227kis  large    31   0    351M    20742   RUN 155  -
necsx6a:1547.necsx5 w-PMSULC  k360pbk  large    31   0    146M     9907   RUN 188  -
necsx5:1627.necsx5 arps-ru*   i002hdj  large    31   0    583M     5697   RUN 219  -
necsx5:1592.necsx5 kdj_200    kdj      realtime 31   0    694M    24472   RUN 195  -
necsx5:1642.necsx5 hqnb       e167jyy  realtime 31   0    3608M    4187   RUN 234  -
necsx5:1628.necsx5 mm5_fdda   r106kaf  small    31   0    164M     5325   RUN 220  -
necsx5:1643.necsx5 compile    r568kjt  small    31   0    89060    1196   RUN 235  -
necsx6a:21.necsx6b test.yj*   yjlee    smal     31   0    1826M     121   RUN  73  -
necsx6b:22.necsx6b test.yj*   yjlee    smal     31   0    1826M   54.819  RUN  50  -
-----------------------------------------------------------------------------------
necsx6a [/system/adm/yjlee/nqs/pgm] $
```

○ Pipe queue에 대한 정보를 알고자 할 때는 다음과 같은 명령어를 사용함

```
necsx5% qstatq -p

=====================================================
NQS (R13.10) PIPE QUEUE SUMMARY     HOST: necsx6a
=====================================================

QUEUE NAME    ENA  STS PRI RLM TOT QUE ROU WAI HLD ARR
--------------- ----------------- ---------------------
Dedicated     ENA  INA  20  10   0   0   0   0   0   0
Economy       ENA  INA  20  10   0   0   0   0   0   0
Express       ENA  INA  20  10   0   0   0   0   0   0
```

```
   Large          ENA   INA   20   10     0    0     0     0     0     0
   Normal         ENA   INA   20   10     0    0     0     0     0     0
   Realtime       ENA   INA   20   10     0    0     0     0     0     0
   Small          ENA   INA   20   10     0    0     0     0     0     0
   normal         ENA   INA   30   12     0    0     0     0     0     0
---------------- ----------------- ---------------------
     <TOTAL>                   50     0    0     0     0     0     0
---------------- ----------------- ---------------------
```

5) 배치작업 취소

○ 실행중인 작업이나 queue에 있는 작업은 'qstatr -ab'로 해당 작업의 id를 확인한 다음에 다음과 같이 'qdel' 명령어를 사용하여 작업을 취소할 수 있음

```
necsx6a% qdel -k 1849
necsx6a% Request 1849 is running, and has been signaled.

necsx6a $ qdel -h necsx6a -k 21.necsx6b
Request 21.necsx6b is running, and has been signalled.
```

6) 배치작업 Hold 및 Release

○ 실행중인 작업이나 queue에 있는 작업은 'qhold' 명령어를 사용하여 hold 하고 'qrls' 명령어로 release 할 수 있음

```
- necsx6a% qhold 1849
  necsx6a% Request 1849 has been gotten checkpoint, and held.
- necsx6a% qrls 1849
  necsx6a% Request 1849 has been release.
```

7) 배치작업 Suspending 및 Resume

○ 실행중인 작업이나 queue에 있는 작업은 'qspnd' 명령어를 사용하여 suspend 하고 'qrsm' 명령어로 resume 할 수 있음

```
- necsx6a% qspnd 1849
  necsx6a% Request 1849 has been suspended.
- necsx6a% qrsm 1849
  necsx6a% Request 1849 has been resumed.
```

2.9. 요금 정책

○ 요금 부과 알고리즘

- 사용료의 기본 서비스 단위는 SRUs(Service Resource Units)로서 다음과 같은 식으로 계산됨

  SRUs = Time × Machine Charging Factor × Service Charging Factor

- Time은 CPU 사용시간으로 다수 사용자가 CPUs를 공유하는 Timeshared 경우에는 총 사용 CPU 시간, 단일 사용자가 CPUs를 독점해서 사용하는 Dedicated 경우에는 wall clock 시간이며 다음 식에 의해 계산됨

  Time = Total CPU Hrs (Timeshared)
  = Total Wall Clock Hrs × No of CPUs (Dedicated)

- Machine Charging Factor는 시스템의 구입가격, 운영비용, 성능 등을 고려한 수치 로서 NEC SX-5/8B 시스템의 경우는 1로 간주됨
- Service Charging Factor는 사용자 요구수준(Service Level)에 따라 부과되는 수치 로서 <표 2-3>에서와 같이 주어짐

  ※ 산업체 사용자의 경우에는 2배의 요금이 부과됨

3. HAMEL

3.1. 시스템 개요

3.1.1 하드웨어 사양

본 클러스터 시스템은 Intel Pentium IV Xeon DP(Dual Processor) 프로세서를 기반으로 하고 있습니다. 각 계산노드는 Gigabit Ethernet과 System Area Network의 일종인 Myrinet을 이용하여 2중으로 연결되어 있으며, Gigbit Ethernet은 파일 I/O 및 시스템 관리용으로, Myrinet은 계산 전용 네트워크로 설계되었습니다.

○ 계산 노드

계산 노드는 인텔 Pentium IV Xeon DP 2.8 GHz 프로세서 2개를 장착한 IBM x335 서버 256대를 Myrinet2000 Network와 Gigabit Ethernet Network를 연결하여 구성하였습니다.

○ 스토리지 노드

공유 파일 서비스를 위하여 SAN(Storage Area Network) 스토리지 서버와 FC RAID 장비그리고 SAN 스위치가 사용되고 있으며, 8대의 파일 서버가 SAN장비를 이용하여 공유하고 있는 10 TB의 FC RAID를 각 계산노드 및 로그인 서버에게 병렬 파일시스템인 GPFS(General Parallel File System)로 서비스하고 있습니다.

○ 로그인 노드

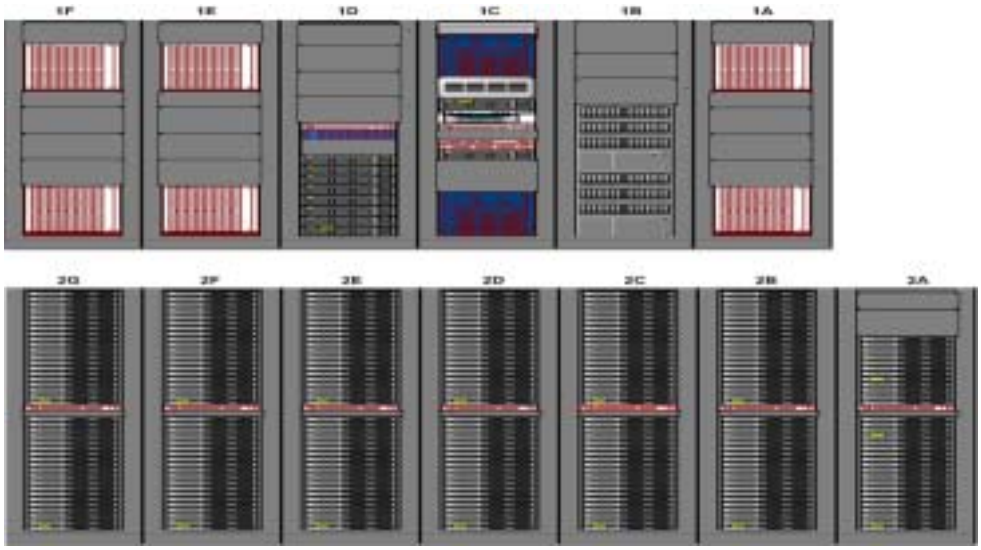
Login 시스템은 4대의 서버를 사용하여 사용자들에게 SSH을 통한 시스템 접근을 지원합니다. 4대의 서버들은 Load Balancing지원을 목표로 하고 있으며, 각각의 Login 서버들은 동일한 사용자 환경 및 데이터 자료를 제공합니다. 각각의 서버에는 사용자들이 사용할 Compiler, Library등이 포함되어 있습니다. 또한 Batch 작업 시스템이 포함되어 있어 사용자들이 작업 큐를 제출하고 확인할 수 있습니다.

○ 디버깅 노드

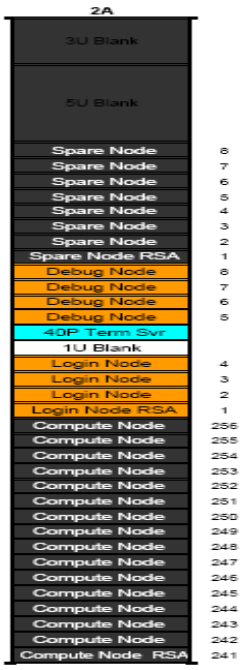
또한 Login 서버와는 별도로 4 대의 디버깅 노드를 두어 사용자가 compile 한 코드를 Totalview 및 SvPavlo등을 사용하여 디버깅 및 프로파일링 할 수 있도록 했습니다. 이 디버깅 노드들은 별도의 8 포트 미리넷 스위치에 연결되어 있습니다. 이 노드에서도 batch 작업 명령을 사용할 수 있습니다.

○ H/W 구성

다음은 본 클러스터 시스템의 랙 구성도입니다.



2B~2G에 있는 랙에는 각각 40개의 계산노드와 랙 가운데에 관리용 terminal server가 위치합니다. 따라서 총 40 X 6 =240개의 노드가 위치합니다.



2A 랙은 다음과 같은 구조를 가지고 있습니다(옆 그림 참조). 이 랙에는 16개의 계산노드와 로그인 노드 4개(lg01 - lg04) 그리고 4개의 디버깅 노드(db01 - db04)와 8개의 예비노드가 위치하고 있습니다.

두 번째 열의 구성을 살펴보면 1A, 1E, 1F 랙에는 각각 2개의 미리넷 스위치(M3-E128)가 들어 있으며, 1B 랙에는 스토리지 서버(FAStT900)와 디스크 박스(EXP 700) 3개로 구성된 스토리지 장비 2대가 위치합니다.

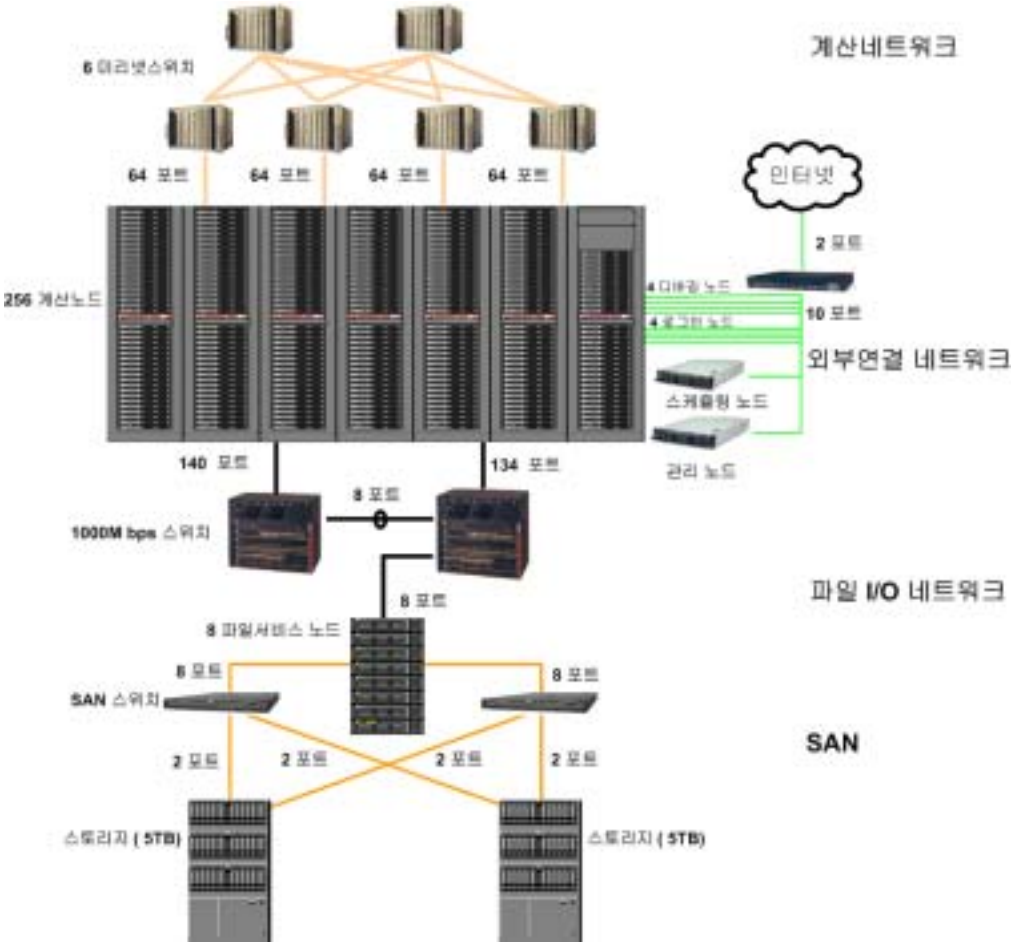
1D 랙에는 8개의 파일서비스 노드와 2개의 Fibre channel 스위치가, 1C 랙에는 파일서비스 I/O를 담당할 Cisco 4006 스위치 2개와 스케쥴링노드, 관리노드가 위치하고 있습니다.

기타 위의 랙구성도에는 보이지만 여기에서 언급 안 된 장비들은 관리 네트워크용 스위치 장비 및 모니터 그리고 콘솔 스위치입니다.

○ 각 구성노드의 세부사양은 다음과 같습니다.

	장비타입	세부 사양	수량	비고
노드	계산노드	IBM eServer x335 2 * Xeon DP 2.8 GHZ w/ 533 FSB/512K L2 2 * 1 GB PC2100 DDR SDRAM 2 * 512 MB PC2100 DDR SDRAM 36.4GB 10K RPM, Ultra320 SCSI HDD Myrinet Fiber/PCI-X Dtype Interface	256	
	예비노드	계산노드와 동일	8	
	디버깅 노드	IBM eServer x335 2 * Xeon DP 2.8 GHZ 4GB SDRAM 2 * 73.4GB SCSI HDD Myrinet Fiber/PCI-X Dtype Interface	4	
	로그인 노드	디버깅 노드와 동일 단, 미리넷카드는 없음	4	
	스케쥴링 노드	IBM eServer x345 2 * Xeon DP 2.8 GHZ 4 * 1 GB SDRAM 2 * 73.4GB SCSI HDD NetXtreme 1000 SX Fiber Ethernet Adaptor Redundant Hot Swap Power Supply	1	
	관리 노드	스케쥴링 노드와 동일 단 2 * 36.4GB SCSI HDD임	1	
	스토리지 노드	스케쥴링 노드와 동일 단, 1000 SX Fiber 이더넷 어댑터는 없으며 2 * FC2-133PCI 2GB/s FC HBA가 있음	8	

○ 시스템의 네트워크 구성도는 다음과 같습니다.



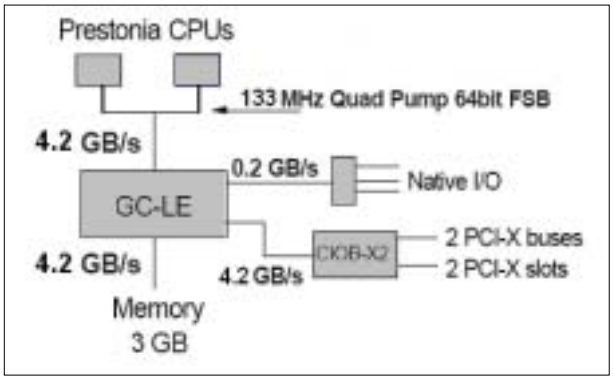
<그림 3-1> HAMEL 시스템 네트워크 구성도

○ SAN 및 시스템 네트워크의 사양은 다음과 같습니다.

SAN	스토리지 서버	FAStT900	2	총 10 TB
	디스크박스	FAStT EXP700	6	
	HDD	146KB 10K RPM 2Gbps FC HDD	70	
	FC 스위치	TotalStorage SAN FC 스위치 F16	2	
계산 네트워크	미리넷Clos 스위치	M3-E128 8 * M3-Spine-8F 라인카드 8 * M3-SW16-8F 라인카드	4	256 노드 Fat-tree 구조
	미리넷Spine 스위치	M3-E128 16 * M3-Spine-8F 라인카드	2	
파일서비스 네트워크	기가비트 스위치	Cisco Catalyst 4006 기가비트 스위치 4 * Cisco 4000 Gigabit 48-port 라인카드 Supervisor Engine III	2	
외부 네트워크	기가비트 스위치	2 GBIC + 8 Gigabit port	1	

계산노드로 사용하고 있는 x335는 ServerWorks의 메인보드를 사용하고 있으며 내부구조는 다음과 같습니다. CPU는 코드네임이 Prestonia인 2.8 GHz Xeon dual processor(DP)을 사용하고 있으며, 20KB의 L1 Cache(cache line: 64B)와 512 KB의 L2 Cache on chip(cache line: 128B)이 내장되어 있습니다.

FSB는 533MHz 클럭으로 작동하며 north bridge 칩셋과 south bridge 칩셋으로 각각 broadcom사의 Grand Champion Low End chipset(GC-LE)과 CIOB-X2를 사용하고 있습니다. 메모리는 1 GB PC2100 DDR SDRAM 두개와 512 MB PC2100 DDR SDRAM 두개로 구성되며, 2-way interleaved 방식으로 작동하여 최대 4.2 GB/s(2 * 266 GHz * 8B) 의 bandwidth을 가집니다.



3.1.2 소프트웨어 환경

본 시스템이 갖는 개발환경 및 user 환경은 다음과 같습니다.

운영체제	Redhat-Linux7.3 w/ kernel 2.4.20-28.7smp
파일시스템	ext3, GPFS 1.3 (/home2, ytmp)
병렬 파일시스템	GPFS 1.3
미리넷 통신 Layer	GM 2.0.12
Batch Job Scheduler	PBSPro 5.4.0
컴파일러	GNU compiler (gcc, g77) 2.96 Portland Group Compiler 5.2(pgf77, pgf90, pgcc, pghpf) Intel Compiler 8.1 (icc, ifc) Absoft Pro fortran 9.2(f77, f90, f95)
병렬 프로파일러	SvPablo, MPE Jumpshot, VTune(serial)
병렬 라이브러리	MPICH-GM 1.2.5..10 MPICH over IP 1.2.5.2 ROMIO
Math 라이브러리 및 기타 라이브러리	SCALAPACK, LAPACK, Atlas, PAPI, BLAS, BLACS, Petsc, Aztec, FFTW, HDF, VTK, NCAR, GOTO Lib, NetCDF, GSL, MKL
병렬 명령어	xCAT(psh, prcp, pping) 1.2.0

3.2. 자원할당 및 시스템 억세스

3.2.1 자원할당 종류 및 규모

슈퍼컴퓨터 사용자는 제공된 SRU(Service Resource Unit) 사용시간 내에서 1년간 KISTI 슈퍼컴퓨터(벡터/병렬)를 사용할 수 있으며, 일반 회원제와 학생 회원제로 구분합니다.

-일반 회원제: 100 SRU 시간(20,000 CPU 시간)/구좌
-학생 회원제: 10 SRU 시간(2,000 CPU 시간)/구좌

SRU와 CPU time과의 관계는 7.2 요금 부과 알고리즘을 참고하십시요.

3.2.2 보안 & 시스템 접근

※ SSH(Secure SHell) 사용을 원칙으로 함
※ telnet을 이용한 접속은 지원하지 않습니다.

사용자가 시스템의 접근하기 위해서는 4대의 로그인 노드와 4대의 디버깅 노드를 이용하여야 합니다. SSH을 통해서 시스템 접근이 가능하며, 각 서버의 설정된 IP를 통하여 시스템의 접속이 가능하나, 특정 로그인 서버의 편중된 접속을 방지하기 위하여, 다음의 대표 IP 주소의 사용을 권장합니다. 이를 통해 로그인 노드 4대중 한대에 접속할 수 있습니다.

[Linux/unix 사용자]

```
대표 로그인 주소 : hamel.ksc.re.kr

$ ssh -l userid hamel.ksc.re.kr 혹은
$ ssh userid@hamel.ksc.re.kr
```

[windows 사용자]
사용법은 다음의 주소에 있는 "ssh 사용법"을 참조하십시오.
http://helpdesk.ksc.re.kr의 technote 게시판

문의 사항 : 보안 담당자 최윤근(ykchoi@kisti.re.kr) 042-869-0524

4 대의 디버깅 노드의 주소는 다음과 같습니다. 이 노드는 interactive 작업, 디버깅, 그리고 프로파일링 작업을 위해 사용할 수 있으며, 사용자가 임의로 선택하여 아래의 네 개의 노드 중 하나에 접속할 수 있습니다.

```
디버깅 노드1 : hamela.ksc.re.kr
디버깅 노드2 : hamelb.ksc.re.kr
디버깅 노드3 : hamelc.ksc.re.kr
디버깅 노드4 : hameld.ksc.re.kr
```

3.3. 사용자 계정관리

3.3.1 패스워드

○ 패스워드를 변경하려면 클러스터 병렬 명령어인 'ppasswd' 명령어를 이용해야 하며, 패스워드는 2개 이상의 영문자와 1개 이상의 숫자나 특수 문자를 포함하여 적어도 6자 이상으로 구성되어야 하고, 기존의 패스워드에서 3자 이상 변경되어야 합니다.

```
[hamel@lg01 hamel]$ ppasswd
Enter New password:
```

3.3.2 쉘(shell)

○ 기본 쉘은 /bin/bash이며, 추가로 POSIX shell:/bin/sh, Korn shell:/bin/ksh, C shell: /bin/csh 등의 쉘을 사용할 수 있으며, 사용자가 쉘을 변경하려면 'pchsh' 명령어를 이용해야 합니다. 사용가능한 shell을 보기 위해서는 pchsh명령을 실행하면 됩니다.

```
[hamel@lg01 hamel]$ pchsh  [shell]
```

3.3.3 홈 디렉토리

○ 본 시스템에서는 사용자의 데이터 저장과 프로그램 실행을 지원하기 위해서 8개의 파일서비스 노드와 스토리지를 사용하고 있으며, GPFS 파일시스템을 적용하여 스토리지 디스크 공간에 3.2 TB의 /home2로 이루어진 홈 디렉토리를 구성하였습니다. 이 홈 디렉토리는 8개의 파일 서비스 노드의 GPFS NSD 서비스를 통하여 256대의 각 노드에 제공되고 있습니다.

○ 홈 디렉토리는 사용자의 데이터와 프로그램을 저장하는 사용자 고유의 디스크 영역이며, /home2/'사용자 ID명'이란 디렉토리 이름을 가집니다. 사용할 수 있는 디스크 량은 5GB를 넘을 수 없습니다. 디스크 쿼터 정책은 구좌당 5GB입니다. 일반적인 경우 /home2 디렉토리의 하부에 사용자 디렉토리가 존재하게 되나, 시스템 운영 관리상 특정 사용자의 홈 디렉토리를 다른 경로에 위치시킬 수 있습니다.

○ 홈 디렉토리의 데이터는 정기적인 백업정책에 의하여 백업되어 집니다. 백업정책은 월 1회 Full Backup과 일일 Incremental Backup입니다.

사용자 홈 디렉토리의 데이터는 사용자가 다른 장소의 저장장치나 KISTI에서 제공하는 장기저장장치인 HPSS에 정기적으로 직접 저장하는 것을 권장합니다.

3.3.4 스크래치 디렉토리

○ 사용자 작업의 수행도중 일시적으로 생성되는 임시파일의 보관을 위하여 전체 계산노드에 GPFS 파일 시스템으로 공유되는 /ytmp(약 2.2TB)가 있으며, 그리고 각 계산노드의 로컬에는 16GB 크기의 /tmp 디렉토리도 제공하고 있습니다.

- 전체 계산노드 공유: /ytmp(2.2TB)
- 계산노드 로컬 : /tmp(16GB)

○ 스크래치 디렉토리는 홈 디렉토리와 달리 여러 사용자가 공용으로 사용하는 디렉토리로서 데이터의 보존에 대한 책임은 전적으로 사용자에게 있습니다. 그리고, 사용자는 자신의 작업이 종료된 후 스크래치 디렉토리의 데이터를 정리하여야 합니다. 그렇지 않으면, 관리자 혹은 관리 프로그램에 의하여 자동적으로 삭제될 수 있습니다.

○ 임시공간의 파일들은 다음에 의하여 삭제될 수 있다.

- 삭제정책(작업종료 후 3시간 보관)에 의하여 삭제될 수 있음
- 시스템이나 타사용자에게 영향을 미칠 경우 관리자에 의하여 삭제될 수 있음

3.3.5 환경변수 설정

○ 각 컴파일러에 맞는 환경변수의 자동 설정을 위해 .bashrc와 .bashrc-[compiler]

- [ip,gm] 파일을 제공합니다. 사용자의 홈 디렉토리에 만들어져 있는 이 파일들을 적절히 변경하여 각 컴파일러의 환경변수를 설정할 수 있으며, 또한 쉽게 다른 컴파일러 환경으로 변경하여 사용할 수 있습니다. 이에 관한 자세한 내용은 4장의 프로그래밍 환경을 참조하기 바랍니다.

3.3.6 사용자 자원 관리

○ 자원 사용내역 확인

- 다음과 같이 옵션없이 usage 명령어를 입력하면 클러스터 시스템에서의 자원 사용 내역에 대한 누적 정보가 나타납니다.

```
[jwsung@lg01 jwsung]$ usage
```

```
     ID     Project  JobNum  JobName        Queue  StartTime  CPUs  WallTime(s)
=====================================================================
   jwsung   ks0229  12757     testing    N16 2004-11-19 08:52   4         1
   jwsung   ks0229  12759     testing    N16 2004-11-19 08:54   8         1
   jwsung   ks0229  12790  cpitesting    N16 2004-11-19 14:51   4         1
   jwsung   ks0229  12791  cpitesting    N16 2004-11-19 14:52   4         1
=====================================================================
                                                                        4
```

- -w 옵션과 -m 옵션을 함께 사용한다면, 지난주(week) 혹은 지난달(month)의 정보를 확인할 수 있습니다. usage 명령어에 의해 나타나는 데이터는 전일(혹은 지난주, 지난달)의 정보를 나타냅니다. 실제적인 과금에 관한 정보는 isam 명령어를 이용하여 확인하시기 바랍니다.

3.4. 프로그래밍

3.4.1 컴파일러

○ 본 시스템에서는 GNU 컴파일러이외에 Portland Group(PGI) 컴파일러와 Intel 컴파일러 그리고 Pro Fortran 컴파일러를 지원하고 있으며, 이들 컴파일러를 사용하여 MPI Library를 사전에 컴파일하여 사용자에게 제공하고 있습니다.

○ 모든 프로그램은 GNU, PGI, Intel 및 Pro Fortran 컴파일러를 사용하여 컴파일 가능하며 mpi 환경을 이용한 컴파일도 가능합니다.(예: gcc, pgcc, icc, mpicc, pgi_mpicc등). 컴파일러에서 사용한 옵션은 mpi환경에서도 사용할 수 있습니다.
시스템의 FORTRAN 컴파일러인 PGI 컴파일러를 수행하는 명령은 'pgf90', 'pgf77'로 대부분의 컴파일러 옵션과 거의 비슷한 형식으로 사용합니다.

○ 병렬프로그래밍 환경의 지원을 위해서, 본 시스템에는 MPICH를 지원하고 있으며 각각의 디렉토리의 위치는 다음과 같습니다.

Compiler & MPICH	PATH
Intel compiler & MPICH	/opt/intel/ /opt/mpich/1.2.5..10/gm-2.0.12-2.4.20-28.7smp-i686/smp/intel32/ssh/ /opt/mpich/1.2.5.2/ip/smp/intel32/ssh/
GNU compiler & MPICH	/usr/ /opt/mpich/1.2.5..10/gm-2.0.12-2.4.20-28.7smp-i686/smp/gnu/ssh/ /opt/mpich/1.2.5.2/ip/smp/gnu/ssh/
PGI compiler & MPICH	/opt/pgi/linux86/5.1/ /opt/mpich/1.2.5..10/gm-2.0.12-2.4.20-28.7smp-i686/smp/pgi/ssh/ /opt/mpich/1.2.5.2/ip/smp/pgi/ssh/
Absoft compiler & MPICH	/opt/absoft/ /opt/mpich/1.2.5..10/gm-2.0.12-2.4.20-28.7smp-i686/smp/absoft/ssh/ /opt/mpich/1.2.5.2/ip/smp/absoft/ssh/

보다 자세한 내용은 다음의 관련 사이트를 참조하시기 바랍니다.

- Intel 컴파일러 : http://www.intel.com/software/products/compilers/linux/
- GNU 컴파일러 : http://www.gnu.org/software/gcc/gcc.html
- PGI 컴파일러 : http://www.pgroup.com/
- MPICH : http://www-unix.mcs.anl.gov/mpi/mpich/

3.4.2 컴파일러 환경변수

○ MPICH의 경로설정은 사용자의 홈디렉토리에 있는 “.bashrc” 파일과 “.bashrc
- [compiler]-[gm,ip]" 파일을 이용하시기 바랍니다. 사용자의 필요에 따라 이 파일을 ".bashrc"의 source .bashrc_[compiler]-[gm,ip] 부분에서 [compiler] 이름을 적절히 수정하여 사용하시기 바랍니다(아래 예제 참조). csh, ksh 등 다른 쉘을 사용하는 사용자는 이 파일들을 적절히 변경하여 사용해야 합니다.

○ 예를 들면, intel 컴파일러를 myrinet 기반에서 사용하기 위해서는 .bashrc-icc
- gm을 사용해야 합니다. 각 컴파일러별 .bashrc_[compiler]-[gm,ip]에는 PATH, MANPATH, LD_LIBRARY_PATH, LICENSE_FILE 등의 부분을 적절히 해당 컴파일러환경에 맞게 수정합니다.

○ 기타 사용자의 특정한 환경변수는 이 파일이나 .bashrc_profile을 이용하여 변경하기 바랍니다. 아래의 예제는 myrinet 환경에서 인텔컴파일러를 사용하기 위해 MPICH등의 환경을 설정한 경우입니다.

```
[hamel@lg01 hamel]$ ls -al .bashrc*
-rw-r--r--    1 root     root        325 Apr 23 09:27 .bashrc
-rw-r--r--    1 root     root        981 Apr 22 10:50 .bashrc-absoft-gm
-rw-r--r--    1 root     root        813 Apr 22 10:51 .bashrc-absoft-ip
-rw-r--r--    1 root     root        713 Apr 22 10:40 .bashrc-gcc-gm
-rw-r--r--    1 root     root        545 Apr 22 10:41 .bashrc-gcc-ip
-rw-r--r--    1 root     root       1362 Apr 22 10:39 .bashrc-icc-gm
-rw-r--r--    1 root     root       1373 Apr 22 10:40 .bashrc-icc-ip
-rw-r--r--    1 root     root       1024 Apr 22 11:02 .bashrc-pgi-gm
-rw-r--r--    1 root     root        856 Apr 22 11:01 .bashrc-pgi-ip
```

```
# .bashrc
# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
cd ~
source .bashrc-icc-gm
#source .bashrc-pgi-gm
#source .bashrc-gnu-gm
#source .bashrc-absoft-gm
... ...
cd -
```

```
#.bashrc-icc-gm
if [ -z "${PATH}" ]
then

PATH="/opt/intel/cc_81/bin:/opt/intel/fc_81/bin:/opt/mpich/1.2.5..10/gm-2.0.12-2.4.2
0-28.7smp-i686/smp/intel32/ssh/bin" ; export PATH
else

PATH="/opt/intel/cc_81/bin:/opt/intel/fc_81/bin:/opt/mpich/1.2.5..10/gm-2.0.12-2.4.2
0-28.7smp-i686/smp/intel32/ssh/bin:$PATH" ; export PATH
fi

if [ -z "${LD_LIBRARY_PATH}" ]
then
    LD_LIBRARY_PATH="/opt/intel/cc_81/lib:/opt/intel/fc_81/lib" ; export
LD_LIBRARY_PATH
else
    LD_LIBRARY_PATH="/opt/intel/cc_81/lib:/opt/intel/fc_81/lib:$LD_LIBRARY_PATH"
    export LD_LIBRARY_PATH
fi
… …
```

3.4.3 표준 라이브러리

○ MPICH

MPI 표준으로 Myrinet GM 프로토콜 기반으로는 MPICH 1.2.5..10을, Ethernet TCP/IP 프로토콜 기반으로는 MPICH 1.2.5를 지원하고 있습니다.

○ Intel Math Kernel Library

MKL(Math Kernel Library)은 기본적인 선형 대수 서브프로그램들(BLAS), 확장된 BLAS(sparse) 및 완전한 LAPACK 루틴들의 세트를 포함하고 있습니다. 라이브러리는 /opt/intel/mkl61에 설치되어 있으며, 다음과 같이 라이브러리를 링크 할 수 있습니다.

- blas인 경우: -lmkl -lguide
- lapack인 경우: -lmkl_lapack -lmkl -lguide

자세한 사항은 다음의 인텔 홈페이지를 참조하시기 바랍니다.

※ http://www.intel.com/software/products/compilers/

3.4.4 Third-party S/W

○ Hamel 시스템에는 수치 계산용 라이브러리들을 별도의 디렉토리에 제공하고 있습니다. 이러한 라이브러리들은 GNU Compiler와 Intel Compiler로 컴파일 되어 있으며 각각은 컴파일러의 이름에 따라 /opt/ksc/[gnu|intel]/에 설치되어 있습니다. 다음 표는 컴파일러 종류에 따라 설치되어 있는 라이브러리의 목록입니다. 자세한 설치 경로는 별도의 문서("KISTI_하멜클러스터시스템_math라이브러리 path")를 참조하시기 바랍니다.

Library list	gnu version	intel version
	설치 경로: /opt/ksc/gnu	설치 경로: /opt/ksc/intel
	설치 여부	설치 여부
gsl	o	o
svpablo 관련	o	
globus 관련	o	
blas	o	o
atlas	o	
blacs	o	o
lapack	o	o
fftw	o	o
scalapack	o	o
netcdf	o	
vtk	o	
hdf	o	
ncarg	o	o

위의 라이브러리를 사용하기 위해 다음과 같이 환경 변수를 설정할 수 있습니다.

```
[hamel@lg01 hamel]$ echo $LD_LIBRARY_PATH
[hamel@lg01 hamel]$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/ksc/gnu/lib
[hamel@lg01 hamel]$ export PATH=$PATH:/opt/ksc/gnu/bin:/opt/ksc/gnu/sbin
```

위의 예에서 intel compiler로 컴파일된 라이브러리를 사용하고자 할 경우에는 gnu 대신에 intel을 입력하시면 됩니다. 혹은 사용자의 기본 환경 설정 파일인 .bashrc 혹은 .bash_profile 등에 위의 내용을 추가하여 사용할 수도 있습니다.

3.4.5 타이밍

○ 사용자 프로그램의 수행 시간을 측정하기 위해서는 MPI 라이브러리에서 기본적으로 사용되는 MPI_Wtime()의 사용을 권장합니다.

3.4.6 프로파일링

○ 프로파일링 기능을 위해 SvPablo를 지원합니다. 이 프로그램은 디버깅 노드, 즉 db01 ~ db04 에 설치되어 있습니다. 실행프로그램인 runSvPablo는 /opt/ksc/gnu/bin에 있으며, data capture 라이브러리인 libsv_DCL.a, libsv_SEQ.a, libsv_DCL_AP.a 등은 /opt/ksc/gnu/lib에 있습니다. 자세한 사용 방법은 아래의 게시판을 참조하세요.

http://helpdesk.ksc.re.kr의 technote 게시판->응용소프트웨어->SvPablo 메뉴얼

```
svpablo quick reference
1. svpablo 프로젝트를 위한 임시디렉토리를 만듭니다.
   여기서는 svpablo2라는 이름을 사용했습니다.
   $ mkdir ~/svpablo2
2. 사용자 어플리케이션의 source code를 앞에서 만든 디렉토리에 복사합니다.
   cp mpich_example3/cpi.c svpablo2
3. runSvPablo를 실행합니다. 다음의 과정을 통해 소스 코드를 instrument합니다.
    1. project-> select Project base directory
    2. project-> new project
    3. configure the location of source and its dir, context dir, instrument dir
    4. instrument your source code as your need
    5. instrument -> save instrumentation
    6. project -> exit SvPablo
4. instrumented 된 코드를 compile 합니다.
   $ ls -al
   total 216
  drwxrwxr-x    3 hamel    hamel        4096 Jun  1 11:10 .
  drwx------   19 hamel    hamel        4096 Jun  1 10:58 ..
  -rw-r--r--    1 hamel    hamel        1988 Jun  1 10:47 cpi.c
  -rw-rw-r--    1 hamel    hamel      219148 Jun  1 10:59
cpi.defaultContext.inst.c
  drwxr-xr-x    3 hamel    hamel        4096 Jun  1 10:58 svTempProject
   $ mpicc -c cpi.defaultContext.inst.c
   $ mpicc -o cpi-intel-gm-svpablo cpi.defaultContext.inst.o
```

```
/opt/ksc/gnu/lib/libsv_DCL.a
5. 위의 과정을 통해 만들어진 바이너리를 일반 mpirun을 통해 실행합니다.
 $ cp mpich_example3/qsub_dual .
 $ vi qsub_dual
 $ qsub qsub_dual
6. mpi 작업이 종료후 만들어진 performance file을 결합합니다.
 $ SvPabloCombine -o c_SDDF* -h defaultContextDir/projectHistory
7. 위의 단계에서 만들어진 performance data를 visualization등을 통해 분석합니다.
 $ runSvPablo
 project -> open project
```

3.5. 작업관리

○ 본 시스템에서는 사용자 작업(job 혹은 request라고도 함)의 배치(Batch)처리를 위하여 PBS(Portable Batch Sytem) Pro를 지원하고 있으며, PBS의 일반적인 특성은 다음과 같습니다.

- 대규모 프로그램 실행 : Interactive session에서 처리할 수 없는 대규모 프로그램 (많은 CPU time을 요하는 프로그램)을 실행시킴
- 표준 출력파일과 표준 에러파일을 디렉토리에 생성시킬 수 있음
- Logout 후에도 작업의 계속 실행 가능
- 기타 시스템에서 동시에 실행할 수 있는 작업 개수를 제한하고, 작업별 CPU time이나 메모리 제한 등의 기능을 수행함

자세한 사항은 다음의 OpenPBS 홈페이지를 참조하시기 바랍니다.
※ http://www-unix.mcs.anl.gov/openpbs/

3.5.1 큐(Queue)

○ 본 시스템에서는 다음과 같은 5종류의 큐를 제공하고 있습니다. 특수한 경우가 아닌 평상시에는 N256 큐의 사용을 제한하고 있으므로 작업에 참조하시기 바랍니다. 사용 가능한 queue의 리스트는 다음과 같습니다.

큐	Walltime Limit (hour)	Max # Node	Max User Run	Max Queue Run	비고
N1	168	1	2	10	
N16	720	16	3	15	
N32	720	32	1	2	
N128	720	128	1	1	
N256	12	256	-	-	(1)

<표 3-1> 하멜 시스템의 큐

주:(1) N256큐는 시스템 관리자에게 요청 후, PM종료 후 혹은 사전 협의된 기간 동안 사용할 수 있으며, Accounting 요금은 요청한 전용시간의 2배로 계산됨(Service Factor=2). 사용자는 N256큐를 이용할 시에 12시간 이상을 사용할 수가 없다.

※ 위의 큐 구성은 변경될 수 있습니다. 시스템 및 사용자의 작업의 특성에 따라 큐의 구성은 변경될 수 있습니다. motd나 mail을 통하여 정보를 얻을 수 있습니다.

○ 배치작업을 실행시키기 위해서는 UNIX 명령어를 포함한 쉘 스크립트에 PBS 옵션과 미리넷을 위한 특별한 스크립트를 반드시 첨가하여 실행하여야 합니다. 위의 구성에서 Serial Job을 위한 큐의 이름은 N1 입니다. 또한 병렬작업을 실행시키기 위해서는 노드 수는 2~128개를 사용하게 되며 cpu 시간의 제한은 최대 720시간(30일)까지입니다.
다음 예제는 16노드(32 cpu)를 사용하여 병렬 프로그램을 수행한 간단한 PBS 작업 실행 script의 예입니다. 보다 다양한 설정을 위해서 **5.3 배치스크립트 샘플**을 참조하시기 바랍니다.

```
#!/bin/bash ----------------------------------------------------------------①
#PBS -l nodes=16:cpp=2 ------------------------------------------------------ ②
#PBS -l walltime=16:00:00 ---------------------------------------------------- ③
#PBS -N bt.A.4 --------------------------------------------------------------- ④
#PBS -q N32 ------------------------------------------------------------------ ⑤
#PBS -j oe ------------------------------------------------------------------ ⑥
#PBS -r n ------------------------------------------------------------------- ⑦
cd $PBS_O_WORKDIR ----------------------------------------------------------- ⑧
/opt/mpich/1.2.5..10/gm-2.0.12-2.4.20-28.7smp-i686/smp/intel32/ssh/bin/mpirun.ch_gm --gm-kill 5
--gm-no-shmem -np 32 -machinefile $PBS_NODEFILE PBS_JOBID=$PBS_JOBID bt.A.4 -------- ⑨
```

① 일반 shell script를 시작하는 방법과 동일합니다.
② PBS옵션은 #PBS로 시작되며, PBS 옵션중 사용할 노드수와 CPU의 수를 기술합니다. 이 노드 수는 사용할 Queue에서 허용하는 최대 노드 수 내에서 선택해야 합니다. 병렬 작업을 위한 스크립터는 반드시 cpp=2로 하여 주시기 바랍니다.
③ 작업이 수행될 최대 예상 시간을 기록합니다.
④ PBS 옵션 중 user가 수행할 작업의 이름을 지정합니다.
⑤ PBS 옵션 중 사용할 queue를 지정합니다.
⑥ 프로그램의 정상적인 output과 error 발생시 생성되는 message를 합치도록 하는 옵션입니다. output 파일의 이름은 ③에서 정한 작업 명칭과 job id를 이용해서 [작업명칭].o[Job ID]와 같은 형태가 되며 $PBS_O_WORKDIR(기본적으로 qsub 명령을 실행할 때 사용자가 있던 디렉토리임)에 output 파일이 만들어집니다.
⑦ 작업이 비정상 종료시(system fail) 작업의 자동 restart를 방지하는 옵션
⑧ 작업을 수행할 프로그램이 있는 곳으로 이동합니다. 기본적으로 사용자가 설정을 하지 않으면 홈 디렉토리입니다. 설정을 위해 ⑧ 앞줄에 사용자가 실행할 프로그램 바이너리가 있는 경로를 지정할 수 있습니다.
⑨ 실제 작업을 적는 부분으로, 마지막에 위치한 내용(bt.A.4)은 수행할 프로그램 바이너리의 이름입니다. 만일 ⑧ 작업을 수행하지 않았다면 프로그램이 위치한 절대 경로를 적어 주어야 합니다.
"--gm-kill 5" 옵션은 프로그램이 종료 후 5초 후에 해당 프로그램이 실행시킨 모든 프로세스를 종료하도록 하는 옵션입니다.

○ 현재 설치된 mpich-gm의 경우 한 노드 내에 있는 두개의 cpu는 서로 shared memory를 이용한 통신을 하도록 기본적으로 되어 있습니다. 하지만 안정성과 성능 면에 여러 가지 문제가 발생하여, 적절한 패치가 이루어질 때까지 사용하지 않을 것을 권합니다. 따라서 위의 경우처럼 "--gm-no-shmem" 옵션으로 사용할 것을 권합니다.

○ PBS_JOBID=$PBS_JOBID의 경우 MPI 프로세스로 pass될 환경변수를 설정한 것입니다. 이 PBS_JOBID 환경변수는 사용자가 qdel을 이용하여 작업을 중지시키고자 할 경우 PBS가 각 job을 구별할 때 사용되는 변수이며, 이를 위해 반드시 mpirun 실행 시 위와 같은 형태로 추가되어 있어야 합니다.

-machinefile $PBS_NODEFILE 의 경우, 필요에 따라 임의의 대체 노드 파일을 qsub 스크립트 내에서 만들어 사용하는 경우도 있습니다. '5.3 배치 스크립트 샘플'을 참조하시기 바랍니다.

위의 script 파일을 qsub명령어를 사용하여 Queue에 올리게 됩니다.

```
[hamel@lg01 hamel]$ qsub test.qsub
```

3.5.2 배치 명령어

3.5.2.1 qsub

○ 배치작업을 queue로 보내어 실행시키기 위해서는 'qsub' 명령어를 사용합니다. qsub를 통한 작업의 제출은 모든 로그인 노드와 디버깅 노드에서 실행할 수 있습니다.
qsub 명령어는 목적에 따라 다음과 같은 여러 가지 옵션이 있습니다.

Option	비 고
-a	작업의 수행을 시작하는 시간 지정
-A	작업을 수행시키는 작업자의 이름 설정
-I	Interactive job을 실행
-j	작업의 output으로 표준에러와 표준출력을 하나의 파일에 출력
-l	사용할 Resource 지정 ex) 작업의 최대 CPU time
-N	작업의 이름을 지정. 입출력파일이 저장될 파일 이름 명시
-p	작업의 priority를 지정

예를 들면 작업을 Interactive하게 작업을 수행하기 위한 queue 요청을 하기 위해서는 다음과 같이 명령을 내릴 수 있습니다.

```
qsub -I nodes=16:cpp=2 -q N16 -I
```

3.5.2.2 qstat

○ 전송된 배치작업의 상태를 알아보기 위해서는 다음과 같이 'qstat -a' 명령어를 사용합니다.

```
[hamel@lg01 hamel]$ qstat -a
                                                   Req'd  Req'd   Elap
 Job ID          Username  Queue    Jobname   SessID  NDS TSK Memory Time S  Time
----------- -------- ------- -------- ------- --- --- ------ ---- - -----
13124.sched     z079rds    N16      Z0512     10553   16   32    --    144:0 R 97:28
```

- Job ID ID + PBS 서버 이름
- Username 제출한 작업의 소유자 이름
- Queue 제출한 작업한 이름
- SessID Session ID
- NDS 작업에서 요청한 노드수
- TSK 작업에서 요청한 cpu의 수(즉, task의 수)
- Req'd Memory 작업에서 요청한 메모리 크기
- S 작업의 상태
- Req'd Time 작업에서 요청한 cpu time 혹은 wall time
- Elap Time 경과 시간

상기에서 작업의 상태는 다음과 같은 정보를 나타냅니다.

E - 작업이 실행 후 종료하고 있음
H - 작업이 hold되어 있음
Q - 작업이 queue에서 대기하고 있음
R - 작업이 실행중임
T - 작업이 새로운 위치로 이동하고 있음.
W - 작업이 실행시간이 될 때까지 기다리고 있음(qsub -a 옵션 사용시)

얼마나 많은 작업들이 queue에 있거나 실행중인가를 알기 위해서 다음과 같이 'qstat -Q' 명령어를 사용합니다.

```
188% [hamel1@lg01 hamel1]# qstat -Q
 Queue          Max   Tot Ena Str  Que Run  Hld   Wat   Trn   Ext Type
 ----------- ----- ----- --- --- --- --- ---- ----- ----- ----- -----
 routeq          0     0 yes yes    0   0    0     0     0     0 Rout
 N256            1     0  no  no    0   0    0     0     0     0 Exec
 N1             25    20 yes yes    1  19    0     0     0     0 Exec
 N16            25    14 yes yes    0  14    0     0     0     0 Exec
 N32             2     1 yes yes    0   1    0     0     0     0 Exec
 N128            1     1 yes yes    0   1    0     0     0     0 Exec
```

Queue : queue의 이름
Max : 동시에 실행 가능한 작업의 최대 수
Tot : 전체 작업의 수(실행+대기)
Ena : queue의 상태(enable or disabled)
Str : queue의 상태(started or stopped)
Que, Run, Hld, Wat, Trn, Ext : queue에 있는 작업의 status별 수
Type : queue의 타입(execution 또는 routing)

3.5.2.3 qdel

○ 다음과 같이 qdel [job_id] 명령어를 사용하여 작업을 취소할 수 있습니다.

```
[hamel@lg01 hamel]$ qstat -a
                                                          Req'd    Req'd    Elap
Job ID     Username  ueue Jobname  SessID  NDS TSK Memory  Time S Time
-------- -------  ---- -------- ------  --- --- ------ ---- - ----
137.sched   hamel    N16   bt.A.4    8840    16   32    --     --   R 00:22
138.sched   hamel    N32   bt.A.4    9589    32   64    --     --   R 00:12
```

```
[hamel@lg01 hamel]$ qdel 138
```

3.5.2.4 qhold

○ 실행중인 작업이나 queue에 있는 작업은 'qhold' 명령어를 사용하여 hold 하고, 'qrerun' 명령어로 release 할 수 있습니다.

```
[hamel@lg01 hamel]$ qhold Job_ID
[hamel@lg01 hamel]$ qrls Job_ID
```

3.5.2.5 qinfo

○ 전송된 배치작업의 상태를 알아보기 위해서는 'qstat'명령어와 유사한 'qinfo' 명령어를 사용할 수 있습니다. 'qstat -a'명령어에서 경과시간(Elapsed time)이 정상적으로 보이지 않을 경우에는 qinfo명령어를 사용하십시오.

```
[hamel@lg01 hamel]$ qinfo
Job ID          User ID     Job Name    S   Queue    Wall Time  NDS  TSK
----------  -------  ---------  -  ------  -------  ---  ---
4850.sched      hsyi        LOG.Nb     R   normal    37:35:49   48   48
4851.sched     x124kca    temp_sub    R   normal    37:16:12   16   16
```

- Job ID　　ID + PBS 서버 이름
- User ID　　제출한 작업의 소유자 이름
- Job Name　　제출한 작업한 이름
- S　　작업의 상태
- Wall Time　　CPU 사용 시간
- NDS　　작업에서 요청한 노드 수
- TSK　　작업에서 요청한 cpu의 수(즉, task의 수)

3.5.2.6 monnode

○ monnode명령어를 사용하면 node와 작업의 정보를 알 수 있습니다.

```
[jwsung@lg01 jwsung]$ monnode
[Job] Run:41 Queue:0  [CPU] Total:512 Run:199 Down:0 Idle:313
[Node]
Myrinet: ..............................................................................
info    : ..............................................................................
        : ...................F...............................
 Nodes : ---------1---------2---------3---------4---------5---------6---------7---------8---------9---------
   1-100: 2222222222222222222222221222222221.221222222212222222222222222222222222222222222222222222222222222.
 101-200: ................................221.1.............1.............................................
 201-256: ......................................................
[Help] F:Myrinet Fault, x:Node Down
```

위의 예에서 2와 1은 노드에서 수행되는 작업의 수를 의미하며, '.'은 작업이 없는 idle한 노드를 나타냅니다.

3.5.3 배치 스크립트 샘플

○ 사용자의 홈 디렉토리에는 시스템에서 제공하는 mpich_example 디렉토리가 있으며, 이곳에는 기본적인 MPI용 큐스크립트 예제가 포함되어 있습니다. 만약 예제 파일이 존재하지 않는 경우에는 /home2/scripts의 mpich_example 디렉토리를 참조하시기 바랍니다.

- MPICH용 Script (intel compiler와 GM을 사용하는 경우)

```
#!/bin/bash
#PBS -l nodes=16:cpp=2      ------------------------------------------------①
#PBS -l walltime=00:30:00
#PBS -N testing
### queue list: N1 (for serial job), N16, N32, N128, N256
#PBS -q N16
#PBS -j oe
#PBS -r n
```

```
#cd into the directory where I typed qsub
cd $PBS_O_WORKDIR

# make temporary nodefile for machinefile
NEW_NODEFILE=`mktemp nodefile.XXXXXX`

# update machinefile and the number of cpus
# (ex: node001, node001, node002, node002, ...., node016, node016 ; NP=32)
/bin/cat $PBS_NODEFILE | awk '{printf("%s₩n%s₩n",$1,$1)}'>& $NEW_NODEFILE    -----②
NP=`/usr/bin/wc -l $NEW_NODEFILE | awk '{print $1}'`

#program name and arguments
PROG="myMPI-intel"
PROGARGS=""

#mpirun
MPIRUN="/opt/mpich/1.2.5..10/gm-2.0.12-2.4.20-28.7smp-i686/smp/intel32/ssh/bin/mpirun.ch_gm"

#run it
$MPIRUN ₩
      -machinefile $NEW_NODEFILE ₩
      --gm-kill 5 ₩            ----------------------------------------------③
      --gm-no-shmem ₩          ----------------------------------------------④
      -np $NP ₩
      PBS_JOBID=$PBS_JOBID ₩   ----------------------------------------------⑤
      $PROG $PROGARGS

#exit
cleanipcs                 -----------------------------------------------------⑥
rm $NEW_NODEFILE
exit 0
```

※ 이 문서는 과거의 문서일수가 있습니다. 그러므로 이 문서에서 제공하는 예제를 복사하여 사용하지 마시고, 시스템의 디렉토리에 있는 예제 파일을 사용하십시오.

위의 **굵은 색**으로 표시된 글자를 적절히 변경하여 사용할 수 있습니다.

① cpp=2로 설정이 되어 있습니다. 여기서 cpp=2를 ppn=2로 바꾸지 마시기 바랍니다. 이것은 아래의 ②번에서 아래의 표에 보이는 바와 같이 적절한 nodefile을 만들기 위해 일부러 이렇게 설정한 것입니다.

② 기본 노드파일인 $PBS_NODEFILE로부터 적절히 노드파일을 수정합니다.
③,④ 5.1 큐 섹션의 하단에 설명된 내용을 참조하시기 바랍니다. 이 두 줄은 ethernet 기반에서 mpirun 실행 시에는 필요 없는 부분으로 제거되어야 합니다.
⑤ MPI 프로세스로 pass될 환경변수를 설정한 것입니다. 이 PBS_JOBID 환경변수는 사용자가 qdel을 이용하여 작업을 중지시키고자 할 경우 PBS에 의해 사용되는 변수이며 이를 위해 반드시 mpirun 실행 시 위와 같은 형태로 추가되어 있어야 합니다.
⑥ 부득이 --gm-no-shmem옵션을 쓰지 않을 경우 이 스크립트를 실행하여 shared memory에서 정상적으로 반환하지 않은 데이터를 지우도록 이 스크립트를 실행하는 것이 좋습니다.

machinefile의 형태별로 qsub스크립트의 내용이 달라집니다. 이를 위해 qsub 예제 스크립트가 각 계정의 mpich_example 디렉토리에 만들어져 있습니다. 혹은 /home2/scripts/mpich_ example 디렉토리를 참조할 수 있습니다.

N16 큐를 사용하여 8노드를 사용할 경우 노드파일은 다음과 같은 형태가 됩니다.

qsub_dual	node001, node001, node002, node002, ..., node016, node016
qsub_single	node001

앞의 스크립트를 실행하는 예제를 보면 다음과 같습니다.

1) 홈 디렉토리의 내용을 확인합니다. 아래와 같은 별도의 예제 파일 디렉토리가 존재하지 않는 경우에는 /home2/scripts/mpich_examples 디렉토리를 홈 디렉토리로 복사하시면 됩니다.

2) MPI 사용자 환경을 점검합니다. bash 쉘인 경우를 예로 들면 각 컴파일러별로 환경변수를 정의하는 .bashrc-icc-gm, .bashrc-icc-ip, .bashrc-absoft-gm, .bashrc-absoft-ip 등의 파일을 만들었습니다. 사용자는 .bashrc 파일에서 source 명령어를 사용하여 필요한 컴파일러 환경을 불러 올 수 있습니다. 사용자는 .bashrc, .bash_profile, 그리고 위의 파일들을 수정하여 적절한 사용자 환경으로 변경시킬 수 있습니다. 이에 대한 자세한 내용은 본 문서의 4.2절 컴파일러 환경변수 부분을 참조하시기 바랍니다.

3) 해당 예제 디렉토리로 이동합니다.

```
[hamel@lg01 mpich_example]# ls -al
-rwxr-xr-x    1 root     root       255603 Apr 17 10:16 cpi-absoft-gm
-rwxr-xr-x    1 root     root       254631 Apr 17 10:16 cpi-gcc-gm
-rwxr-xr-x    1 root     root       280284 Apr 17 10:16 cpi-icc-gm
-rwxr-xr-x    1 root     root       415662 Apr 17 10:16 cpi-pgi-gm
-rw-r--r--    1 root     root         1698 Apr 17 10:13 cpi.c
-rw-r-----    1 root     root         1555 Apr 22 08:14 qsub_dual
-rw-r-----    1 root     root         1521 Apr 22 07:54 qsub_normal
-rw-r-----    1 root     root         1520 Apr 22 08:10 qsub_single
-rw-r-----    1 root     root         1558 Apr 22 08:13 qsub_smp
```

4) 컴파일을 실행합니다. 이미 MPI환경이 설정이 되어 있으므로 별도 경로 지정 없이 MPI명령을 내리면 됩니다. 혹시 위의 2)단계에서 MPI 환경을 변경하였다면, 환경 변화의 반영을 위하여 재 로그인 하시거나, source 명령을 수행하십시오.

```
[hamel@lg01 mpich_example]$ mpicc ./cpi.c -o cpi
```

5) 5.3절에서 설명한 배치 스크립트 파일을 적절히 변경하여 큐에서 실행 시킵니다.

```
[hamel@lg01 mpich_example]$ qsub qsub-test
12212.lg02.kisti.re.kr
[hamel@lg01 mpich_example]$ qstat -a

sched:
                                                        Req'd Req'd      Elap
Job ID      Username  Queue   Jobname    SessID NDS TSK Memory Time S  Time
-------- -------- ------ -------- ------ --- --- ------ ---- - -----
121.sched   bslee     N128    mm5        4332   128 128   --     --  R 00:08
125.sched   bslee     N32     dpsgs-c    4719    64  64   --     --  R 00:02
131.sched   hamel     N16     testing    3116     8  16   --     --  R 00:01
[hamel@lg01 mpich_example]$
```

3.5.4 병렬 명령어

3.5.4.1 psh

IBM의 xCAT에서 지원하는 가장 일반적인 병렬명령어로서 계산노드에 대해 어떤 종류의 쉘 명령어도 실행할 수 있습니다. 아래와 같이 “psh”을 입력하면 간단한 사용방법을 볼 수 있습니다. 아래의 예제에 나와 있는 바와 같이 계산 노드의 node range를 지정하여 명령을 실행할 수 있습니다.

```
[hamel@lg01 hamel]$ psh

Usage: psh [-s] [-i interface] [noderange|me|pbs job id] [command]

e.g.

psh node001,node003 uptime
·psh node014-node056,node070-node203 uptime
psh node001,node003,node014-node056,node070-node203 uptime
psh all,-node129-node256 uptime
```

또한 PBS의 JOB ID를 이용하여, 현재 작업이 실행중인 노드에 대해서만 명령을 실행할 수도 있습니다. 아래의 예는 현재 작업의 각 노드에서의 pcpu를 보기 위한 명령의 예입니다.

```
[hamel@db03 hamel]$ qstat -a
sched:
                                                        Req'd   Req'd    Elap
Job ID        Username Queue   Jobname     SessID NDS TSK Memory Time   S Time
---------- -------- ----- ---------- ----  --- --- ------ ---- - -----
1897.sched      hamel     N256    TES256      2357  256  256    --     256:0 R 00:28
1898.sched      hamel     N256    TES256-02    --   256  512    --     256:0 Q  --

[hamel@db03 hamel]$ psh 1897 'ps -eo pcpu,cmd | grep [program_name] | grep -v grep'
node005:          94.6 /home2/hamel/[program_name]
node003:          95.8 /home2/hamel/[program_name]
node004:          95.9 /home2/hamel/[program_name]
node006:          95.8 /home2/hamel/[program_name]
node002:          95.9 /home2/hamel/[program_name]
node001:          95.9 /home2/hamel/[program_name]
```

```
node011:          95.9  /home2/hamel/[program_name]
node007:          96.0  /home2/hamel/[program_name]
node009:          96.0  /home2/hamel/[program_name]
....
```

3.5.4.2 prcp

하나의 파일 혹은 디렉토리를 여러 개의 노드에 병렬로 복사하고자 할 때 사용할 수 있습니다.

```
[hamel@lg01 hamel]$ prcp

Usage: prcp [files] [noderange:destination directory]
Usage: prcp [-r] [files] [directories] [noderange:destination directory]

e.g.

prcp -r /usr/local node001,node003:/usr/local
prcp -r /usr/local node014-node056,node070-node203:/usr/local
prcp -r /usr/local node001,node003,node014-node056,node070-node203:/usr/local
prcp -r /usr/local all,-node129-node256:/usr/local

[hamel@lg01 hamel]$
```

3.6. 대용량 스토리지(HPSS)

3.6.1 HPSS 개요

HPSS는 고속 병렬 처리를 위한 기존의 슈퍼컴퓨터 및 클러스터 컴퓨터에게 높은 가용성과 범용성을 제공하는 스토리지 서비스입니다. HPSS는 연구소, 정부, 의료기관 등에서 많은 양의 대용량 데이터로 인하여 발생될 수 있는 운영상의 문제를 고속 네트워크와 계층적 스토리지 시스템을 이용하여 해결함으로써 애플리케이션들의 높은 효율성을 제공하고 있습니다. 사용 가능한 용량은 구좌당 200GB로 쿼터가 적용되어 있습니다.

3.6.2 사용법

HPSS 계정을 발급받으신 분들은 HPSS 서버에 접속하기 위해서 시스템에 로그인하여 들어와 있는 상태에서 다음의 명령어를 입력합니다.

```
% pftp_client core 4021
  =>pftp_client [hpss_서버] [서버_포트]
```

위의 명령어를 입력한 후세 다음의 화면을 보실수 있으며 발급받으신 HPSS의 아이디와 패스워드를 입력합니다.

```
Parallel block size set to 8388608.
Connected to core.
220-
**************************************************
*                                                *
* This is the HPSS 4.3 system running at KISTI   *
*                                                *
**************************************************

220 core FTP server (HPSS 4.5 PFTPD V1.1.2 Wed Sep 1 04:02:05 KORST 2004) ready.
Name (core:root):
```

파일의 store와 retriee와 delete를 위한 명령어 사용법은 다음과 같습니다.

ftp>pput a.dat	# 1개의 파일 store
ftp>mpput *.dat	# 여러 개의 파일을 동시에 store/ 와일드 카드 사용 가능
ftp>pget a.dat	# 1개의 파일 retrieve
ftp>mpget *.dat	# 여러 개의 파일을 동시에 retrieve/ 와일드 카드 사용 가능
ftp>del a.dat	# 1개의 파일 delete
ftp>mdel *.dat	# 여러 개의 파일을 동시에 delete/ 와일드 카드 사용 가능
ftp>quot	# 현재 사용 중인 사용량 정보 표시

※자세한 사용법은 다음의 주소에 있는 "HPSS 시스템 사용 지침서"를 참조하십시오.(http://helpde나.ksc.re.kr)

※문의 사항은 HPSS 담당자(ykchoi@kisti.re.kr)에게 메일을 보내십시오.

3.7. 사용자 자원관리

3.7.1 사용자 등록/삭제

○ 클러스터 시스템을 사용하려는 사용자는 먼저 지정된 양식을 작성하여 슈퍼컴퓨터 계정관리자(jangoq@kisti.re.kr 042-869-0540)에게 제출해야 합니다.

○ 계정관리자는 계정 신청자에게 user ID와 패스워드를 e-mail이나 전화로 통지합니다.

○ 추가의 user ID가 필요할 경우와 삭제할 user ID가 발생할 경우에는 계정담당자에게 연락을 하시면 됩니다.

3.7.2 요금 부과 알고리즘

요금은 SRU(Servic Resource Unit)를 단위로 하여 부과되며, SRU는 다음과 같이 요청한 CPU의 수와 wall clock time을 기본으로 하고 있습니다.

```
SRU = CPUs × Time × Service Charging Factor × Machine Charging Factor
      CPUs: 요청한 CPU의 수
      Time: Wall Clock Hours
      Service Charge Factor: 1
      Machine Factor:  0.005
```

3.7.3 자원 사용내역 확인

isam 명령어를 입력하면 모든 시스템에서의 자원 사용 내역에 대한 누적 정보가 나타나게 됩니다.

3.7.4 보상(Refund)

○ 보상을 요구할 수 있는 경우는 다음과 같습니다.
- 시스템 충돌 혹은 하드웨어 고장 때문에 출력이 없어짐.
- 시스템 소프트웨어의 버그로 부정확한 출력이 발생함.
- KISTI의 잘못된 상담을 하여 부정확한 출력이 발생함.

○ 다음의 경우에는 보상 요구가 인정되지 않게 되오니 참조하시기 바랍니다.
- 2주 이후에 요청하였을 때
- 작업내역에 대한 기록이 없을 때
- 스크래치 파일 시스템을 사용하지 않았기 때문에 출력이 분실된 경우
- 작업 종료 후 스크래치 파일 시스템에 장기간 결과 파일을 방치한 경우

○ 보상을 요청하기 위하여 다음 정보를 제출하여야 합니다.
- 사용자 ID, 시스템 이름, Batch 작업 ID, 보상 이유, 작업이 끝난 날짜

3.7.5 계정 종료, 갱신, 연장

○ 사용자 계정의 할당 받은 자원이 소진되었거나 계정기간이 만료되었을 때, 계정(로그인 아이디)은 통지 후 삭제됩니다.

○ 사용자들은 계정이 종료된 이후 홈 디렉토리의 파일들을 가져가기 위하여 약 1주일간의 접속은 가능하며, HPSS 사용자 계정들은 시스템 계정이 삭제된 후 1년 기간 동안은 읽기전용 모드로 계정이 존재함 끝.

부 록

1. IBM FAQ

1.1.시스템 일반

1.1.1 각종 시스템 관련 참고 자료들은 어디에서 찾을 수 있나요?

슈퍼컴퓨팅센터 홈페이지(http://www.ksc.re.kr)의 “슈퍼컴퓨터 사용자 자료실“에 가시면 다운로드 받을 수 있습니다.

1.1.2 X 응용프로그램을 실행시키고자 할 때는 어떻게 해야 하나요?

가. 사용자 측에 방화벽이 존재한다면, 먼저 방화벽에 다음과 같은 포트가 개방되어 있는지 확인해야 합니다.

접속 방향			포트 번호	프로토콜
사용자 PC IP	⇒	150.183.5.101 150.183.7.101 150.183.5.102 150.183.7.102	22, 23	TCP
			177	UDP
150.183.5.101 150.183.7.101 150.183.5.102 150.183.7.102	⇒	사용자 PC IP	6000-6010	TCP

나. localhost 측에서 “xhost + nobela.supercomputing.re.kr[로그인 노드]”를 실행하여 서버에서 로컬 호스트로의 X 접속을 허용합니다.

다. 로그인 시스템[nobela.supercomputing.re.kr]에 로그인 합니다.

라. DISPLAY 환경 변수를 설정합니다.

csh : setenv DISPLAY "Local Host IP":0.0

sh, bash, ksh : export DISPLAY="Local Host IP":0.0

마. X 응용프로그램을 실행합니다.

1.1.3 로그인 노드[nobela/nobelb]의 명령 프롬프트 상에서 interactive[대화식]로 일정 시간 이상 작업을 수행하면 작업이 종료되는 데, 무엇이 문제인가요?

현재 Interactive 작업은 CPU Time을 10분으로 제한하고 있습니다. 병렬 작업에 대한 디버깅을 위한 Interactive 모드로 수행하기 위해서는 nobel1/nobel2에 로그인하여 사용할 수 있습니다. 이외의 작업은 LoadLeveler를 이용하여 배치 작업으로 수행시켜야 합니다.

1.1.4 프로그램을 튜닝하기 위한 지원은 받을 수 있나요?

응용 지원을 담당하는 각 분야의 전문가들이 있기 때문에, Helpdesk (T. 080-041-9712)에 전화하여 지원 요청을 하면 됩니다.

1.1.5 프로그램에서 더 많은 메모리를 사용하기 위해서는 어떻게 해야 하나요?

프로그램 컴파일 시에 -q64 옵션을 설정하여 64비트로 컴파일 하면, 메모리 사용에 대한 제약이 없습니다.

```
$ xlf -q64 -o large_mem_program large_mem_program
```

하지만, 위 옵션을 설정하지 않고 프로그램을 컴파일 하셨다면 기본으로 32비트로 컴파일 되기 때문에, 256MB의 세그먼트 하나만 할당된다. 그러므로 256MB 이상의 메모리(주로 heap 메모리)를 사용하려면 아래의 예와 같이 링크 옵션 -bmaxdata를 사용해야 합니다.

```
$ xlf -o large_mem_program -bmaxdata:0x80000000  large_mem_program
```

-bmaxdata의 단위는 바이트이며 크기는 십진수, 8진수(0으로 시작), 16진수(0x로 시작)로 나타낼 수 있는데, 위의 예는 8개의 데이터 세그먼트를 허용하여, 최대 2GB까지 heap 메모리를 사용할 수 있게 하는 것입니다.

1.1.6 슈퍼컴퓨터 사용자 교육 일정은 어떻게 확인할 수 있습니까?

슈퍼컴퓨팅 교육센터 홈페이지(http://wededu.ksc.re.kr)에 나와 있습니다.

1.1.7 터미널 타입을 변경하고자 합니다. 어떻게 하여야 하나요?

- csh : setenv TERM type
- sh, bash, ksh : export TERM=type

1.1.8 현재 사용 가능한 쉘의 종류는 어떤 것들이 있습니까?

sh, ksh, csh, bash, tcsh가 사용가능 합니다.

1.2.시스템 접속

1.2.1 시스템에 로그인 한 후 외부로의 ssh와 ftp 접속 시도가 거부되는데, 무엇이 문제인가요?

방화벽 정책상 외부에서 내부로 ssh와 ftp는 허용하지만, 내부에서 나가는 것은 E-Mail을 제외하고는 허용되지 않습니다.

1.2.2 nobela[nobel]/nobelb 로그인 시스템을 제외한 기타 컴퓨팅 시스템에 직접 로그인이 거부되는데, 무엇이 문제인가요?

nobel 시스템은 로그인 노드로 설정되어 있고, p690 시스템은 컴퓨팅 시스템으로 설정되어 있기 때문에 컴퓨팅 시스템에 대한 직접적인 외부접근은 차단되어 있습니다. 컴퓨팅 노드에 로그인 하기 위해서는 rsh 혹은 ssh 명령으로 로그인 노드에서 다시 접속해야 한다.

1.2.3 XDM/CDE 세션으로 시스템에 접속할 수 있나요?

윈도우즈 환경에서 X-manager 및 Exceed 등의 X 서버 에뮬레이션 프로그램을 사용하면 접속할 수 있습니다. 접속과 관련된 설정은 FAQ의 1.시스템 일반 부분을 참조하면 됩니다.

1.3. 계정

1.3.1 IBM 시스템의 계정 등록은 어떻게 하나요?

HelpDesk 무료 전화 080-041-1991에 전화하여 계정 등록 신청서를 제공받아 작성·제출하고, 슈퍼컴퓨터 회비를 온라인/지로로 입금하면 됩니다. 또한, 계정 등록은 사이버 Helpdesk 홈페이지[http://helpdesk.ksc.re.kr]에서 온라인으로도 가능합니다.

1.3.2 일반 유료 사용자와 전략과제 사용자는 어떻게 다른가요?

일반 유료 사용자는 구좌당 일정한 사용료를 지불하고 있는 대부분의 슈퍼컴퓨터 사용자들이며, 전략과제 사용자는 국가적 지원의 필요성이 인정되는 응용 연구과제를 선정하여 슈퍼컴퓨팅 자원을 무상으로 지원 받는 사용자입니다.

1.3.3 계정의 유효기간은 언제까지인가요?

계정 등록 후 1년입니다.

1.3.4 왜 제 계정이 비활성화 또는 접근이 차단되게 되나요?

할당된 CPU 시간을 전부 사용한 경우이거나 계정의 유효 기간이 경과한 경우입니다.

1.3.5 자원사용 내역은 어떻게 확인 할 수 있나요?

로그인 후 명령 프롬프트 상태에서 “isam” 명령을 실행하고, 메뉴 방식에 의하여 메뉴를 선택하게 되면 필요한 정보를 찾아볼 수 있습니다.

1.4. 파일 시스템

1.4.1 홈 디렉토리에서 삭제된 파일을 복구할 수 있나요?

사용자 홈 디렉토리는 주기적으로 백업을 받고 있습니다. 복구를 원하시는 시점을 알려주면 복구가 가능합니다.

1.4.2 기본 할당량 보다 더 많은 홈 디렉토리 공간을 할당받을 수 있나요?

정책에 따른 기본 할당량을 초과하여 홈 디렉토리를 할당할 수 없으며, 이를 초과하는 데이터는 HPSS를 사용하여 저장할 수 있습니다. 사용자 지침서의 “12. HPSS를 사용한 대용량 데이터 저장 관리“ 부분을 참조하세요.

1.4.3 내장 스크래치와 공유 스크래치 디렉토리는 어떻게 다른가요?

로컬 스크래치 /xtmp1은 노드별로 디렉토리 이름은 동일하지만 물리적으로 서로 다른 내장 스크래치 파일 시스템입니다. 그리고, 공유 스크래치는 모든 노드에서 접근이 가능하며, 대용량 파일에 대한 순차 I/O 성능이 우수한 GPFS 파일 시스템[/gpfs1-4] 및 NFS 파일 시스템[/ytmp]입니다.

1.4.4 배치 작업을 종료 시켰을 때, 스크래치 디렉토리도 같이 정리해 주어야 하나요?

정리하여야 합니다. 스크래치 디렉토리에 있는 파일들은 현재 작업 종료 후 최종 액세스[읽기/쓰기] 시간으로부터 1일 이상 경과되면 자동 삭제되기 때문에, 반드시 홈 디렉토리, HPSS, 사용자 로컬 사이트 등에 이전 보관하여야 합니다.

1.5. 배치작업

1.5.1 작업을 submit 하고 계속해서 idle 상태에 머무는 데 이유는 무엇인가요 ?

일반적으로 시스템 자원이 다른 작업에 의해서 점유되어 할당할 자원의 여유가 없거나, 사용자 ID별로 할당되는 최대 작업 개수를 초과한 경우입니다. 자신의 작업이 pending된 이유를 알려면 “llq -s JOB_ID“ 명령어를 사용하여 확인할 수 있습니다. 또한, 노드별 가용 CPU수를 확인하기 위해서는 ”llstatus -R" 명령을 사용하여 사용자 작업이 실행될 수 있는 대상 노드들의 ConsumableCpus 및 ConsumableMemory의 가용 자원을 확인하여 CPU 및 메모리 자원의 부족으로 idle 상태에 머무는지 파악할 수 있다.

1.5.2 backfill scheduling이란 무엇입니까?

현재 LoadLeveler에서 제공되고 있는 작업 스케쥴링 기법으로 앞에서 대기 중인 작업의 수행에 영향을 미치지 않으면서 현재 가용한 자원만 사용하여 wall_clock_limit 시간 내에 작업을 마칠 수 있는 경우, 먼저 수행하도록 허용하는 기법입니다. 따라서, Job Command File에 반드시 wall_clock_limit값을 설정하여야 합니다.

1.5.3 submit script의 resources 옵션에서 반드시 지정해야 하는 ConsumableCpus 및 ConsumableMemory가 무엇을 의미합니까?

예) #@ resources = ConsumableCpus(16) ConsumableMemory(1gb)

위의 예는 태스크 당 16개 CPU 및 1GB의 메모리를 요청하는 것이다.

- ▶ ConsumableCpus

  serial, Pthread 또는 OpenMP 프로그램인 경우에는 실제로 사용할 CPU 개수를 뜻하며, MPI 작업인 경우에는 각 태스크당 쓰레드 개수라고 보면 됩니다. 특히, OpenMP 혹은 OpenMP+MPI 프로그램인 경우에는 OMP_NUM_THREADS 또는 XLSMPOPTS=parthds 변수의 값과 ConsumableCpus 값이 같아야하고, serial 혹은 순수 MPI 프로그램인 경우에는 ConsumableCpus(1)을 지정하면 됩니다.

- ▶ ConsumableMemory

  1CPU, Pthread 또는 OpenMP 프로그램인 경우에는 전체 메모리 용량을 뜻하며, MPI 프로그램인 경우에는 각 태스크 당 메모리 용량이라고 보면 된다.

  ※ ConsumableMemory 단위

- ▶ 100MB : ConsumableMemory(100mb) 혹은 ConsumableMemory(100) :

default 단위 MB

▶ 1GB : ConsumableMemory(1gb)

1.5.4 작업에 대한 checkpoint/restart가 가능한가요?

Checkpoint 기능은 사용자의 작업 수행 중 적당한 시점에 작업의 정보를 디스크에 저장하여, 프로그램이 완료되지 않은 상태로 중단된 경우에도 다시 처음부터 수행하지 않고 정보가 저장된 상태 이후부터 수행할 수 있도록 해주며, 순차 프로그램과 병렬 프로그램 모두 지원됩니다. 자세한 사용법은 사용자 지침서 18장의 "Checkpoint/Restart" 부분을 참고하세요.

1.5.5 특정 노드에서만 작업을 실행하도록 하는 방법은 무엇인가요?

Submit script 파일에서 아래와 같이 옵션을 지정하면 됩니다.

```
#@ requirements = (Machine=={"nobel1"  "noble2"})
```

1.5.6 LoadLeveler 관련 명령어를 수행하면, 'command not found' 메시지가 나오고 실행되지 않습니다. 어떻게 해야 하나요?

LoadLeveler 관련 명령어의 path가 설정되어 있지 않기 때문입니다. 사용자 초기화 파일[.cshrc 혹은 .kshrc]의 path 변수에 "/usr/lpp/LoadL/full/bin" 경로를 추가하면 됩니다.

1.5.7 로컬 스크래치[/xtmp1]를 사용하여 작업을 수행하기 위해서는 어떻게 해야 하나요?

/xtmp1 로컬[내장] 스크래치를 사용할 경우에는 다음과 같이 LoadLeveler Job Command File에서 작업 명령 실행 전에 작업 파일들을 홈디렉토리에서 실행노드의 /xtmp1 작업 디렉토리로 복사하고, 작업 종료 후 결과 파일들을 실행노드의 /xtmp1 작업 디렉토리에서 홈디렉토리로 복사하는 스크립트를 작성해야 합니다.

```
#!/bin/ksh
#@ job_type = serial
#@ class = normal
#@ resources = ConsumableCpus(1) ConsumableMemory(1gb)
#@ wall_clock_limit = 10:00:00
#@ queue
rm -rf /xtmp1/wjnadia     # 작업 실행 노드의 /xtmp1 디렉토리에 기존 작업 디렉토리 삭제
mkdir /xtmp1/wjnadia      # 실행노드의 /xtmp1 디렉토리에 새로운 작업 디렉토리 생성
cp -f /edun/wjnadia/* /xtmp1/wjnadia # 작업 실행에 필요한 파일들을 실행노드의 /xtmp1
                                     # 디렉토리로 복사
cd /xtmp1/wjnadia                # 실행 노드의 /xtmp1 작업 디렉토리로 작업 경로 변경
  ./long_serial_program
cp -f /xtmp1/wjnadia/* /edun/wjnadia/output/  # 작업 결과 파일을 홈 디렉토리로 복사
rm -rf /xtmp1/wjnadia              # /xtmp1의 작업 디렉토리 삭제
```

2. NEC

Cray C90의 프로그램을 NEC 시스템으로 옮기는 과정에서 발생했던 주요 문제점들 중 본문에서 언급되지 못한 부분에 대한 해결책을 FAQ 형식으로 정리하였음

2.1. 컴파일시 나타나는 벡터화 관련 메시지 없애기

컴파일 시 매 루프마나 벡터화 여부를 화면으로 출력하게 되는데 이 경우 컴파일러 메시지가 너무 길어져서 정작 중요한 메시지를 확인하지 못하는 수가 있으며, 이 경우 다음 옵션을 사용하여 벡터화 관련 메시지를 없앨 수 있음

-Wf "-pvctl nomsg" 또는 -Wf ,-pvctl nomsg

컴파일 시 이 옵션을 추가하면 되고, 아니면 다음과 같이 환경변수를 설정하거나 각자 initial shell script에 이 줄을 추가해 놓으면 됨

csh의 경우: $ setenv F90_USER_OPTIONS "-Wf,-pvctl nomsg" 또는

sh의 경우: $ export F90_USER_OPTIONS= "-Wf,-pvctl nomsg"

2.2. '-ew' 옵션과 64bit 라이브러리 링크

'-ew' 옵션 등 single precision을 8byte로 처리해주는 옵션을 사용할 경우 라이브러리도 이에 맞는 것으로(64bit) 링크를 해 주어야 함

4byte real로 컴파일시: $ f90 -o myprog myprog.f -llapack -lblas

8byte real로 컴파일시: $ f90 -o myprog myprog.f -ew -llapack_64 -lblas_64

※ 모든 경우에 대하여 64bit(8byte real)라이브러리가 지원되는 것은 아님 (libxxx.a, libxxx_64.a와 같이 같은 이름의 _64로 이름이 붙은 라이브러리 파일이 있는 경우만 지원...)

2.3. NEC에서 C/C++ 프로그램 컴파일 및 실행

2.3.1 NEC 시스템에서 C 언어 컴파일러와 C++ 컴파일러는 다른가요?

C++의 경우 c++ 명령어를 이용하여 컴파일을 수행해야 하며, c++을 컴파일러로 사용하는 경우 최적화 옵션으로 -C hopt/vopt/sopt/ssafe 등을 사용할 수 있음

2.3.2 C/C++에서 성능분석을 위한 환경변수 설정 방법 :

프로그램 수행성능 등을 확인하기 위한 F_PROGINF(setenv F_PROGINF DETAIL/YES/NO) 환경변수는 포트란 프로그램(즉, main이 포트란으로 작성된)에 대해서만 적용되며, C/C++ 프로그램의 경우는 C_PROGINF 환경변수를 다음과 같이 사용함

```
$ setenv C_PROGINF DETAIL/YES/NO (csh 경우)
```

2.4. NEC 시스템에서 난수발생 루틴 처리

○ Cray 시스템에서는 포트란 프로그램에서 난수발생을 위해 ranset이라는 서브루틴과 ranf라는 함수를 사용하며 그 예는 다음과 같음

```
      CALL RANSET(1)
         DO 10 K=1,N3M
            ANXFXO(I,J,K)=(RANF()-RANF())*AMPP
            ANYFYO(I,J,K)=(RANF()-RANF())*AMPP
            ANZFZO(I,J,K)=(RANF()-RANF())*AMPP
         10 CONTINUE
```

○ NEC 시스템에서의 사용 예는 다음과 같음

```
      CALL RJUFSP(N3M*6,ix,iy,rr,ierr)
        DO 10 K=1,N3M
           ANXFXO(I,J,K)=(rr(k,1)-rr(k,2))*AMPP
           ANYFYO(I,J,K)=(rr(k,3)-rr(k,4))*AMPP
           ANZFZO(I,J,K)=(rr(k,5)-rr(k,6))*AMPP
         10 CONTINUE
```

※ 8byte 실수(real) 경우는 사용하려면 RJUFSP 대신 DJUFSP를 사용함

2.5 Sparse solver

유한요소해석 등에서 나타나는 Sparse 행렬에 의한 연립방정식을 고속으로 풀기 위한 Sparse solver가 Cray C90 시스템에서는 SSPOTRF, SSPOTRS 등의 이름으로 제공 되었으나, NEC 시스템에서는 이 루틴은 제공되지 않고 이와 거의 비슷한 기능을 하는 LDLT_FACT, LDLT_SOLVE 등의 루틴이 제공되므로 이

를 이용하면 유한요소해석 등에서 나타나는 연립방정식을 고속으로 풀 수 있음. 사용법은 다음과 같이 On-line 매뉴얼 참조

$ man solver

2.6 사용자 Q&A 게시판

NEC 시스템 사용 중 문제점이나 의문사항, 혹은 시스템 사용과 관련된 주요 정보 및 최신 정보들을 다음 게시판을 통해서 얻을 수 있음

http://www.ksc.ne.kr/helpdesk/index_nec.html