천둥 사용법

멀티코어 컴퓨팅 2017 가을학기

서울대학교 매니코어 프로그래밍 연구단

김희훈

천둥

천둥

- 이종 슈퍼컴퓨터
  (heterogeneous supercomputer)
  - 서울대학교 매니코어 프로그래밍 연구단과 매니코어소프트(주)에서 개발
  - 소프트웨어(SnuCL) 성능 평가용
  - 2012년 10월 구축
  - 2012년 11월 TOP500 리스트 및 Green500 리스트 등재
- 2013년 4월부터 일반에 서비스 제공 중
  - http://chundoong.snu.ac.kr/

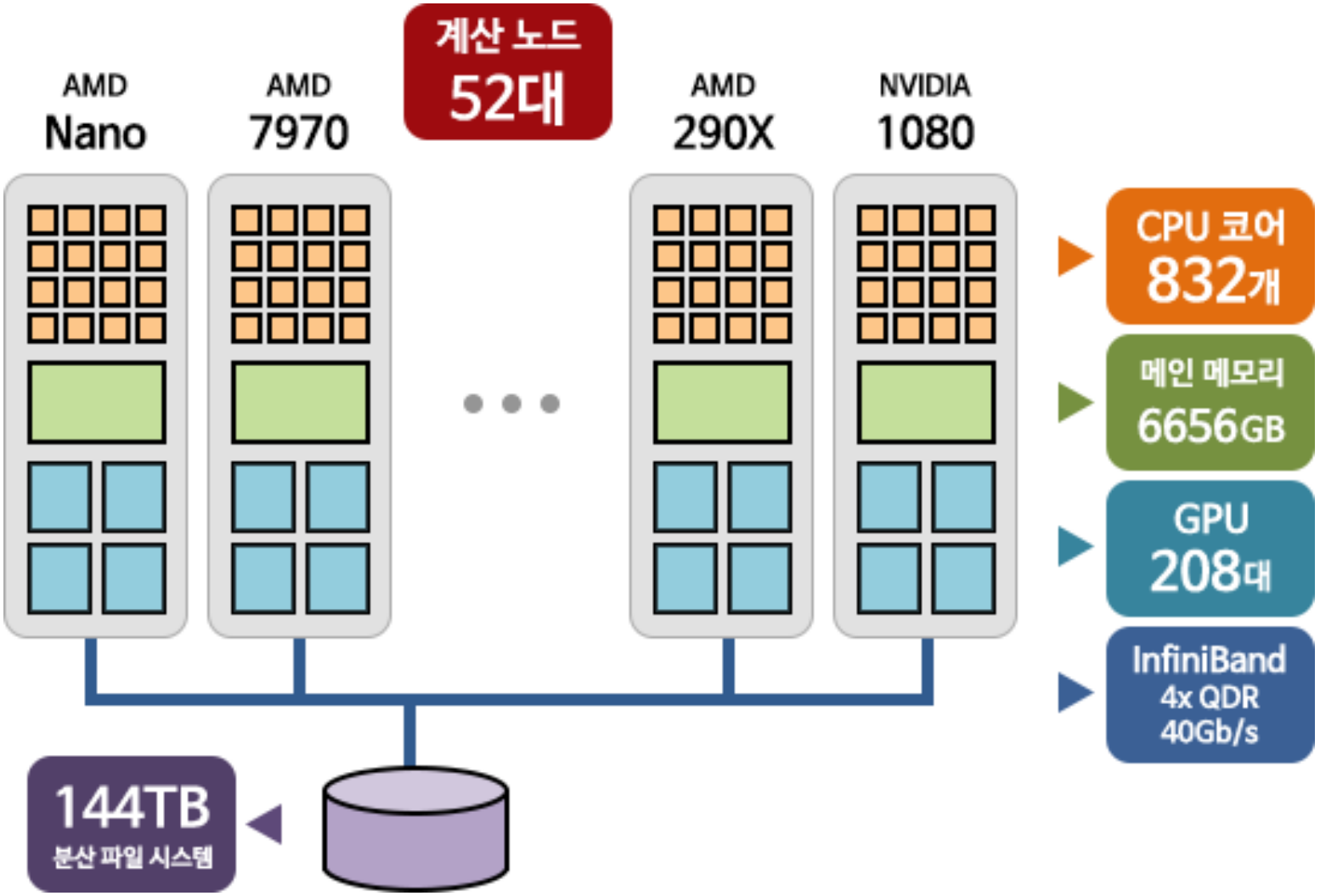
TOP500 리스트
277위
(2012년 11월)

Green500 리스트
32위
(2012년 11월)

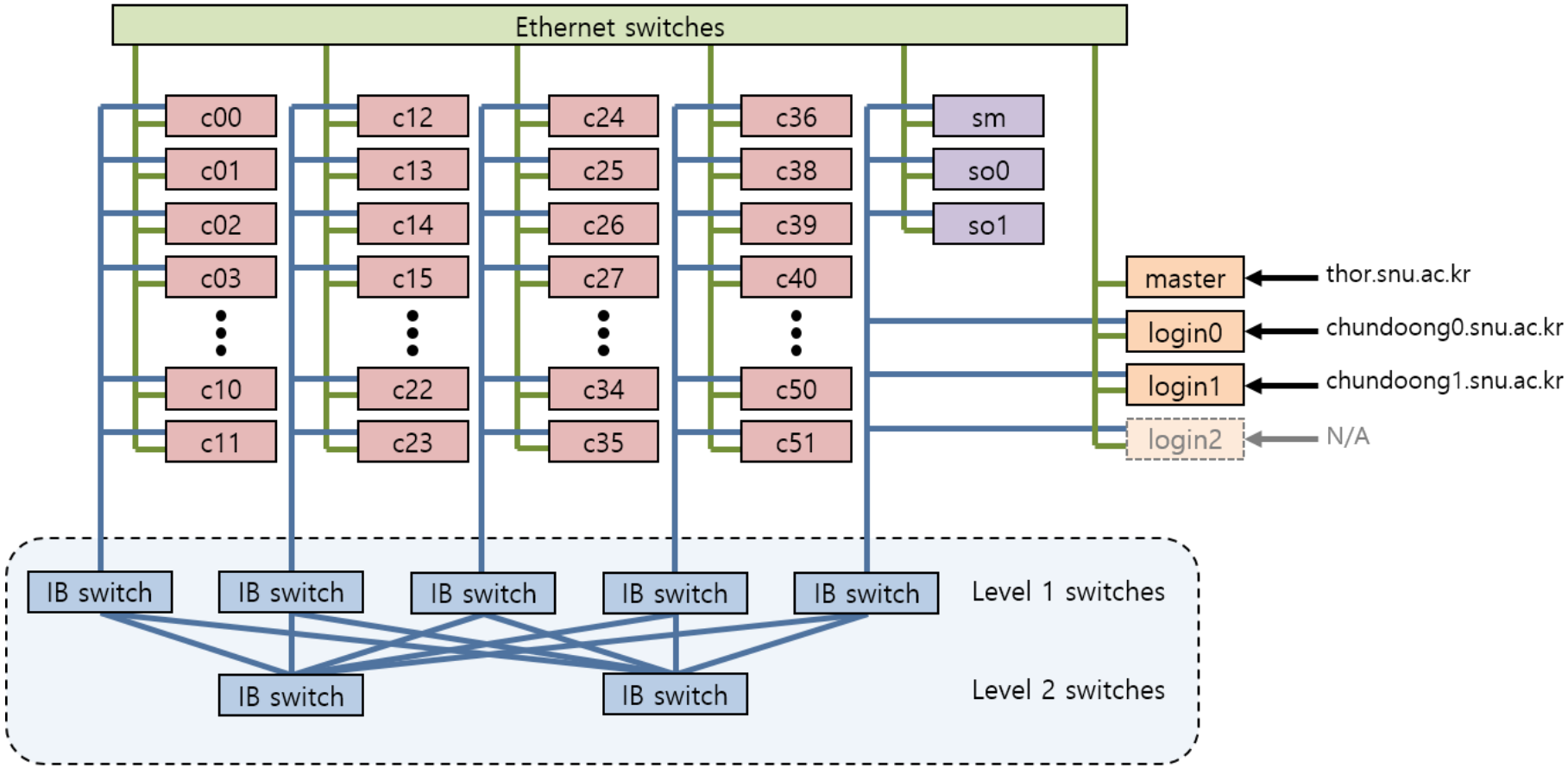
한국연구재단
기초연구 우수성과 50선 선정
(저비용, 고효율 이종 슈퍼컴퓨터 천둥의 연구개발 및 구축)

천둥 1.5

천둥 1.5


계산 노드
52대
AMD
Nano
AMD
7970
AMD
290X
NVIDIA
1080
CPU 코어
832개
메인 메모리
6656GB
GPU
208대
InfiniBand
4x QDR
40Gb/s
144TB
분산 파일 시스템

천둥 1.5



InfiniBand 4x QDR (40Gb/s) interconnection network

천둥 1.5: 시스템 구성

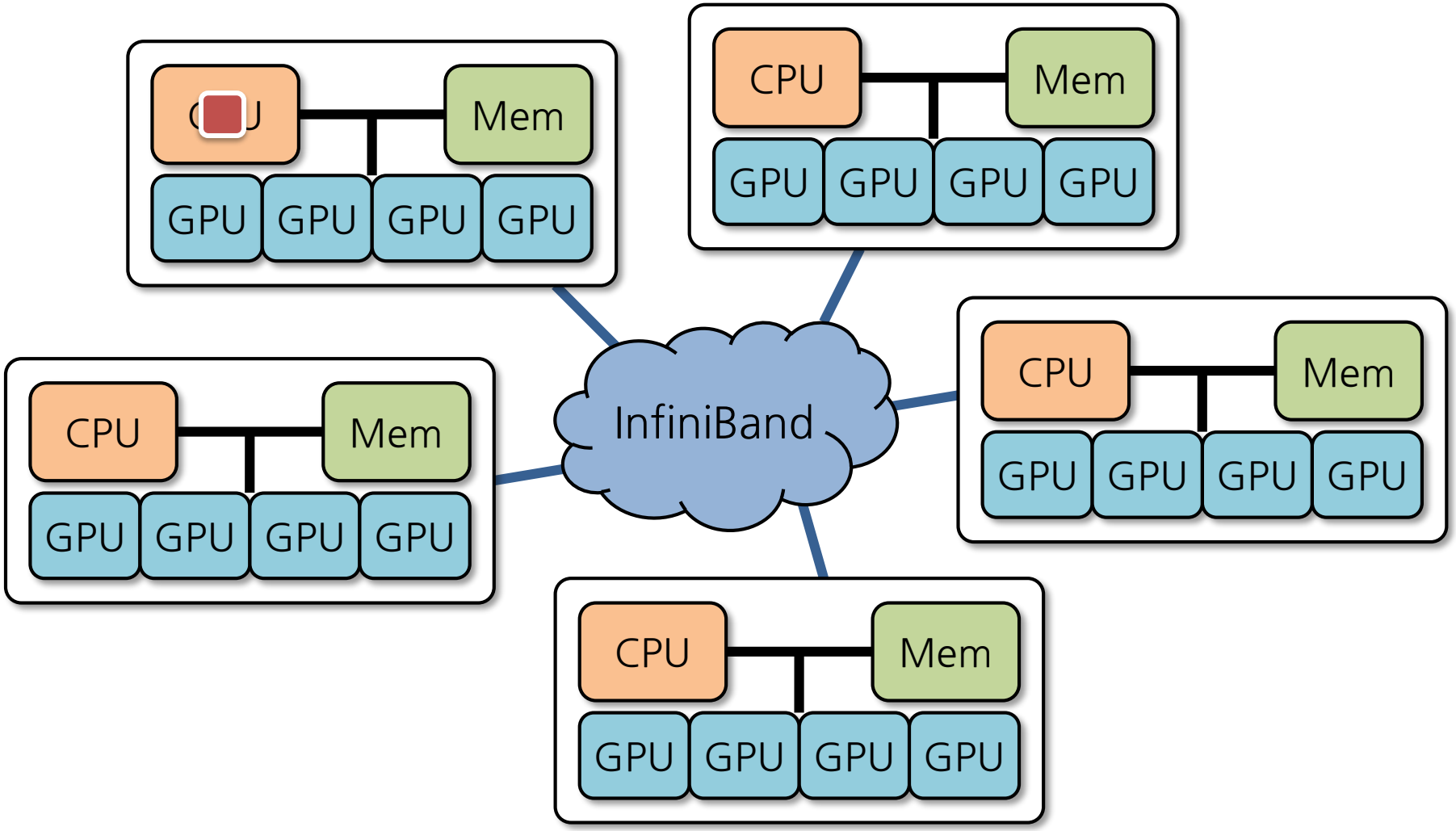
클러스터 구성

항목	내용
계산 노드	52개
스토리지	물리 192 TB, 가용 144 TB Lustre 파일 시스템
관리 노드	1대
로그인 노드	2대
인터커넥션	InfiniBand QDR
총 CPU 개수	104개, 832코어
총 GPU 개수	208개 - **AMD HD 7970** 96개 - **AMD R9 Nano** 48개 - **AMD R9 290X** 16개 - **NVIDIA GTX 1080** 64개
총 메인 메모리	6,656 GB

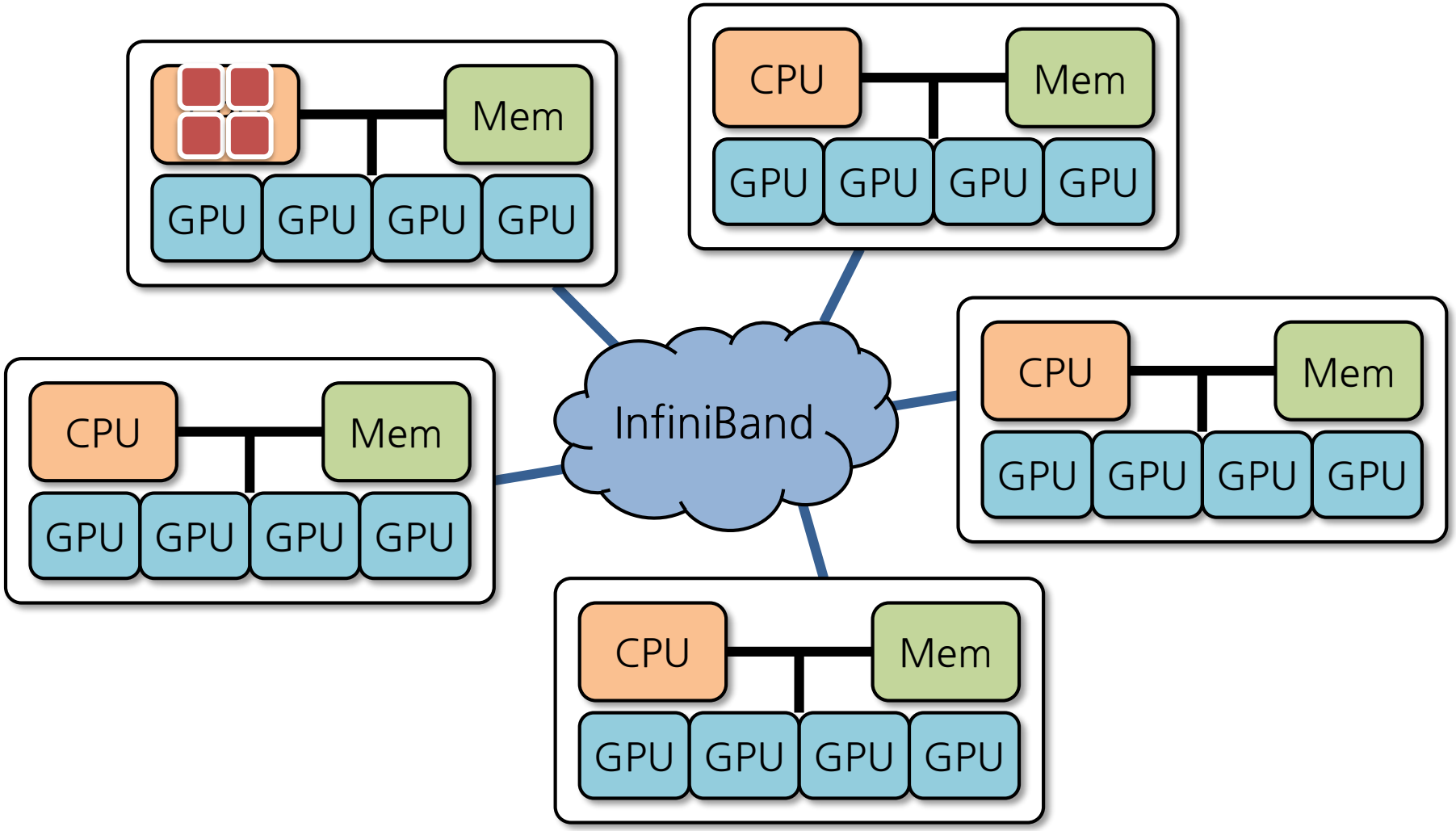
계산 노드 구성

항목	내용
CPU	2 × Intel Xeon E5-2650 (총 16코어, 2.0 GHz)
GPU	다음 중 하나: 4 × AMD Radeon HD 7970 4 x AMD Radeon R9 Nano 4 x AMD Radeon R9 290X 4 x NVIDIA GeForce GTX 1080
메인 메모리	128 GB (1,600 MHz DDR3)
HCA	Single-port Mellanox InfiniBand QDR HCA
운영체제	CentOS 7.2

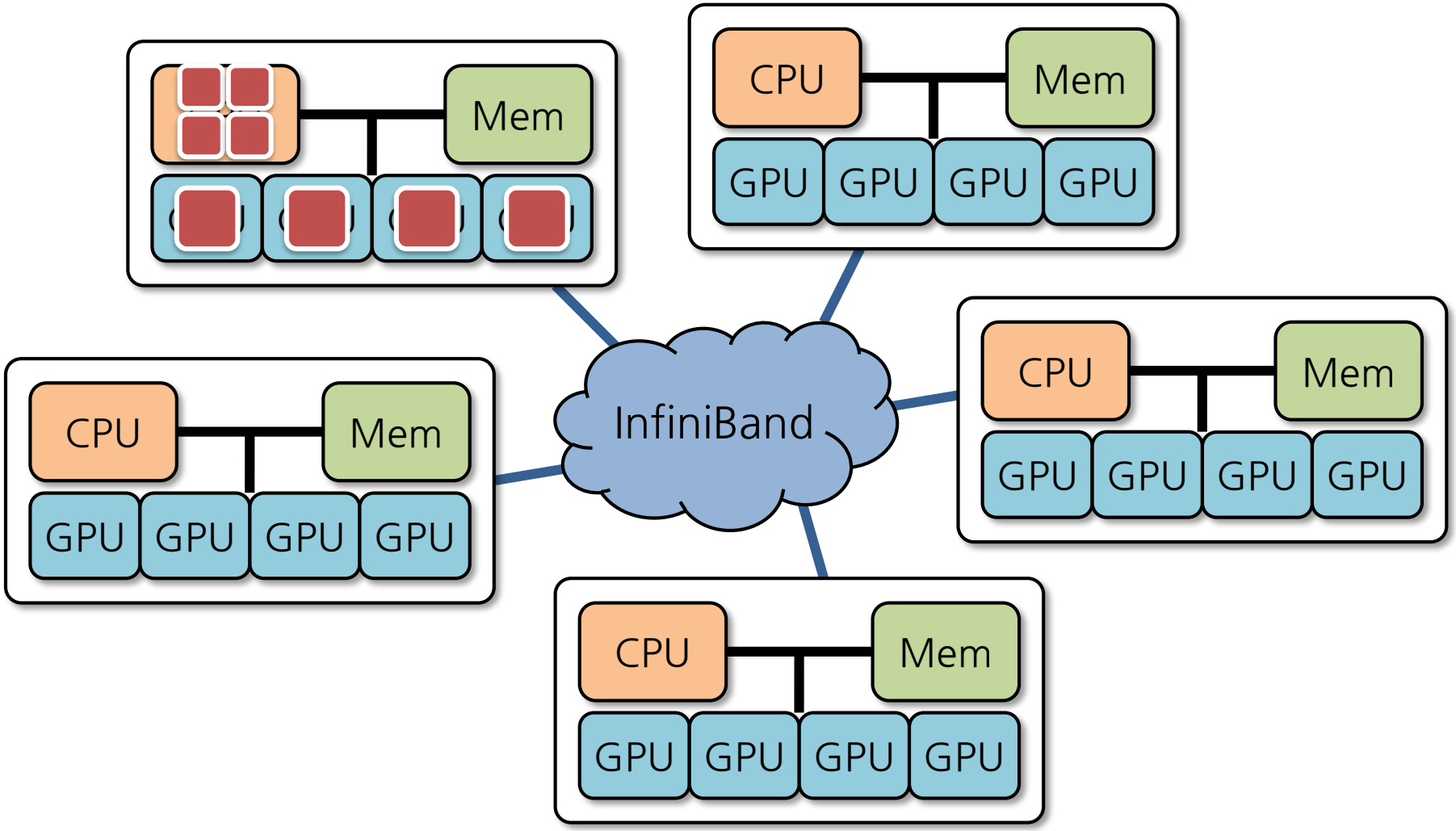
병렬 프로그래밍을 하지 않은 경우



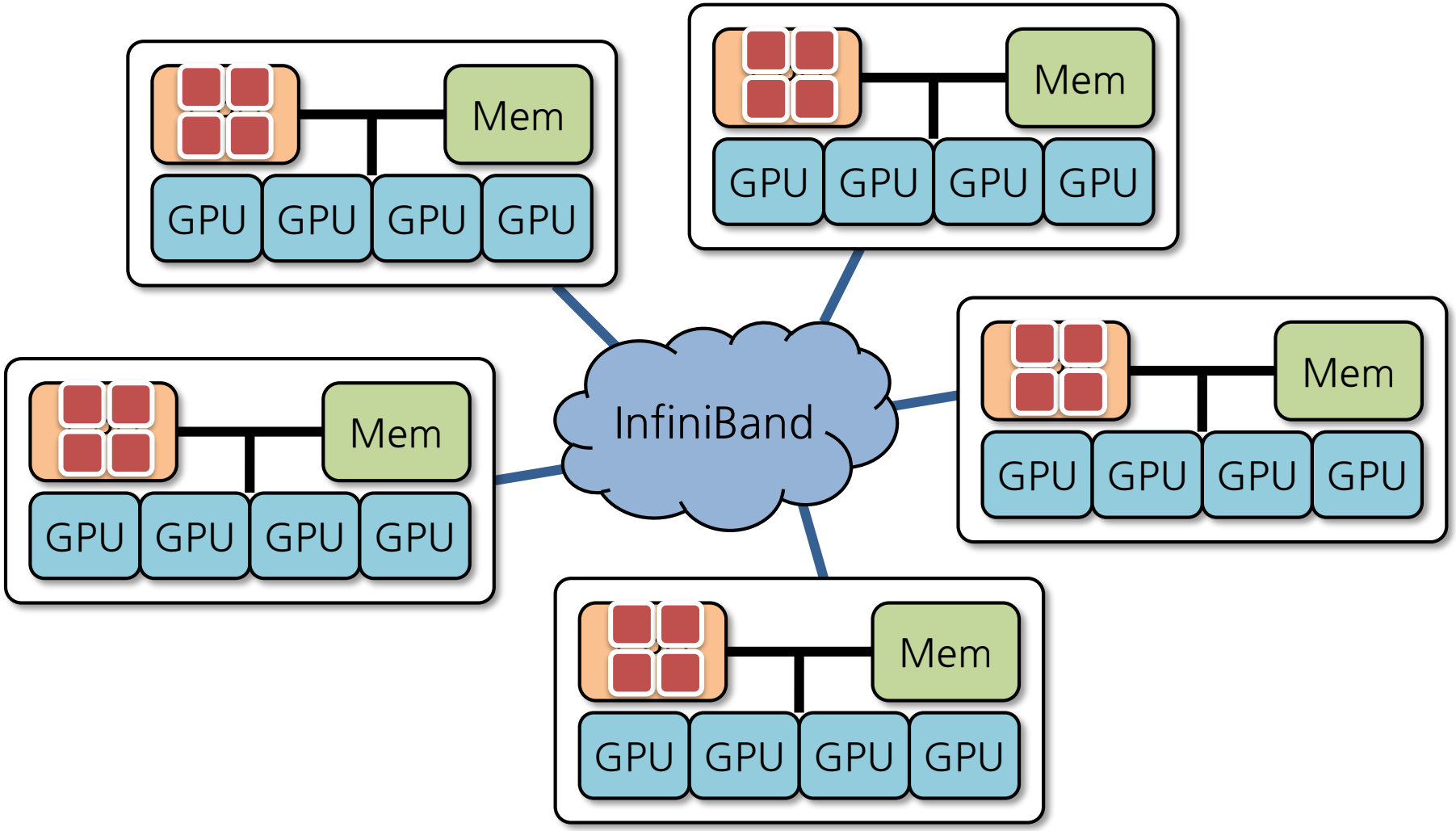
Pthreads, OpenMP



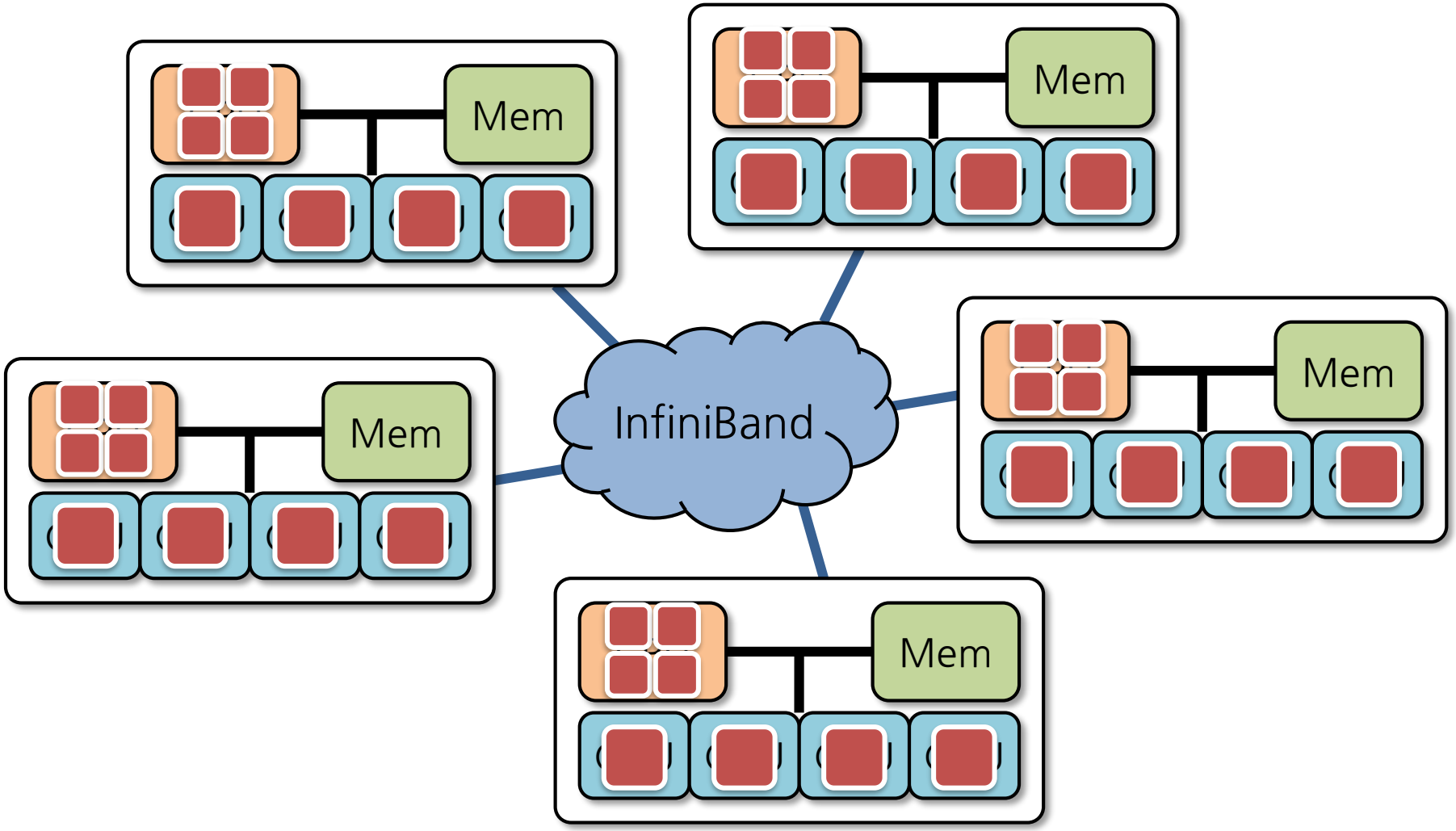
OpenCL



MPI


Mem
GPU
GPU
GPU
GPU
Mem
GPU
GPU
GPU
GPU
Mem
GPU
GPU
GPU
GPU
InfiniBand
Mem
GPU
GPU
GPU
GPU
Mem
GPU
GPU
GPU
GPU

MPI + OpenCL, SnuCL



천둥 계정: mc01 ~ mc30

번호	이름	계정
1	박대영	mc01
2	이현준	mc02
3	조용익	mc03
4	박종빈	mc04
5	박준모	mc05
6	박재유	mc06
7	위민복	mc07
8	최성준	mc08
9	김현규	mc09
10	강성진	mc10
11	박호진	mc11
12	이한규	mc12
13	김동주	mc13
14	박서홍	mc14
15	정모두하리	mc15

번호	이름	계정
16	정연규	mc16
17	박종훈	mc17
18	손혜원	mc18
19	이동규	mc19
20	윤지학	mc20
21	김종범	mc21
22	최원우	mc22
23	조승현	mc23
24	김진연	mc24
25	박진용	mc25
26	이준영	mc26
27	최남재	mc27
28	오평석	mc28
29	정희수	mc29
30	민우영	mc30

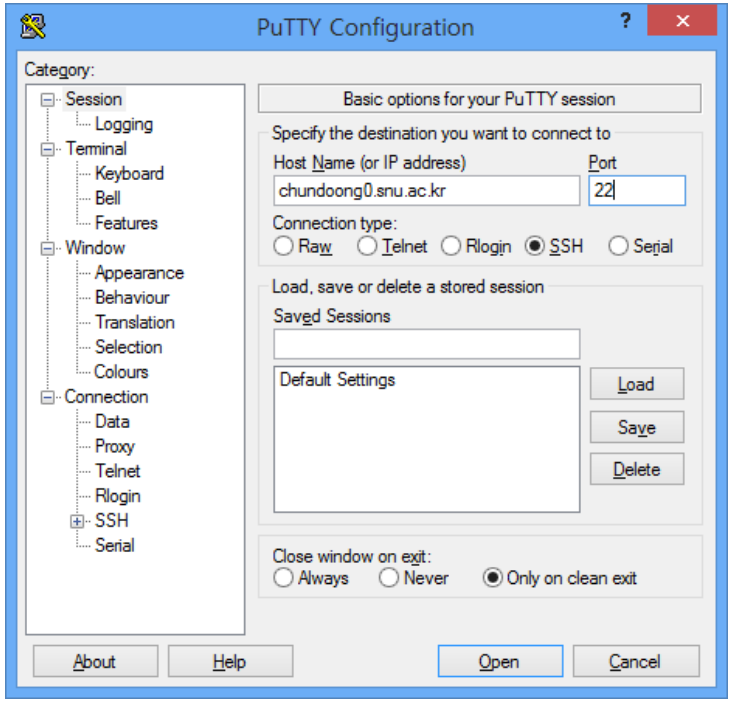
로그인

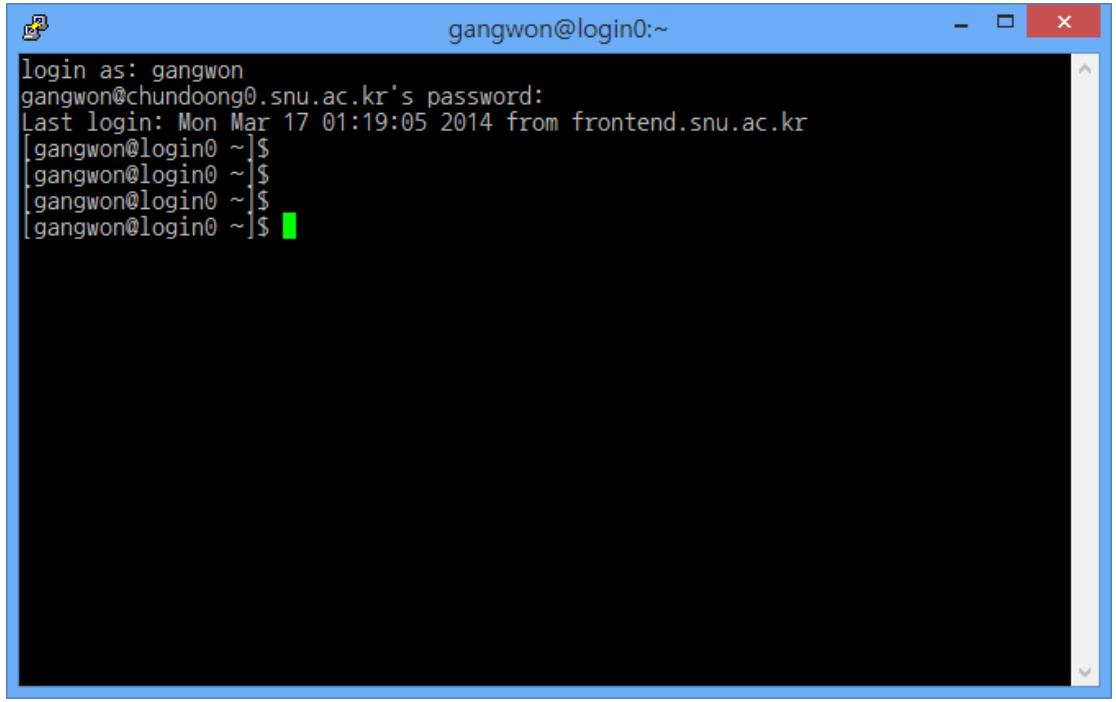
- SSH 클라이언트 사용
  - Windows의 경우: PuTTY, Xshell, …
  - Mac OS X의 경우: 터미널에서 ssh 명령 사용
- 발급받은 아이디와 패스워드를 이용
  - 기본 패스워드: multicore2017
- 로그인한 후 passwd 명령으로 패스워드 변경할 것

로그인 노드	주소	포트
login0	**chundoong0.snu.ac.kr**	**22 혹은 2222**
login1	**chundoong1.snu.ac.kr**	

로그인

- Windows의 경우





- Mac OS X의 경우
  - `ssh mc00@chundoong0.snu.ac.kr`

리눅스 쉘 사용법

- 홈 디렉터리(~)
  - 처음 로그인하면 /home/mcXX 디렉터리에 위치
  - 이 안에서 프로그램 작성, 컴파일 등의 작업 수행
- pwd: 현재 디렉터리 위치를 보여 줌
- ls: 현재 디렉터리의 파일 목록을 보여 줌
- cd: 다른 디렉터리로 이동
  - cd /home/mc00/my_dir (절대 경로)
  - cd my_dir (상대 경로: (현재 디렉터리)/my_dir로 이동)
  - cd ~/my_dir (/home/mcXX/my_dir 디렉터리로 이동)
  - cd .. (상위 디렉터리로 이동)

리눅스 쉘 사용법

- mkdir: 디렉터리 만들기
  - mkdir aaa
  - mkdir aaa/bbb
  - mkdir /home/mc00/ccc
- cp: 파일 복사
  - cp file_1 file_2 (file_1 ⇒ file_2)
  - cp file_1 my_dir (file_1 ⇒ my_dir/file_1)
  - cp file_1 my_dir/file_2 (file_1 ⇒ my_dir/file_2)
- cp -r: 디렉터리 전체를 복사
  - cp -r dir_1 dir_2 (dir_1 ⇒ dir_2)
  - cp -r dir_1 my_dir (dir_1 ⇒ my_dir/dir_2)
  - cp -r /scratch/share/some_example .
    (/scratch/.../some_example ⇒ some_example)

리눅스 쉘 사용법

- mv: 파일 혹은 디렉터리 이동
  - mv file_1 file_2 (file_1 ⇒ file_2)
  - mv file_1 my_dir (file_1 ⇒ my_dir/file_1)
  - mv file_1 my_dir/file_2 (file_1 ⇒ my_dir/file_2)
- rm: 파일 삭제
  - rm file_1
- rm -r: 디렉터리 삭제
  - rm -r dir_1

리눅스 쉘 사용법

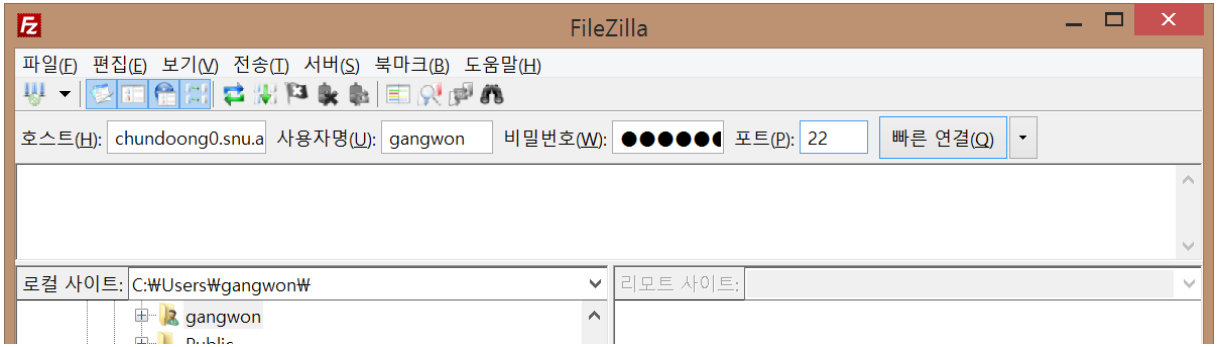
- `cat`: 파일 내용 출력
  - `cat result.txt`
- `diff`: 파일 내용 비교
  - `diff a.txt b.txt`
  - 다른 점이 없으면 아무 것도 출력되지 않음

리눅스 쉘 사용법

- gcc -o (실행 파일 이름) (소스 파일 이름) (옵션)
  - gcc -o my_program my_program.c
  - gcc -o my_program my_program.c -O3 -lOpenCL
  - (주의) gcc -o my_program.c my_program.c
- ./(실행 파일 이름)
  - ./my_program
  - ./my_program 10
    - argc, argv로 받을 수 있음

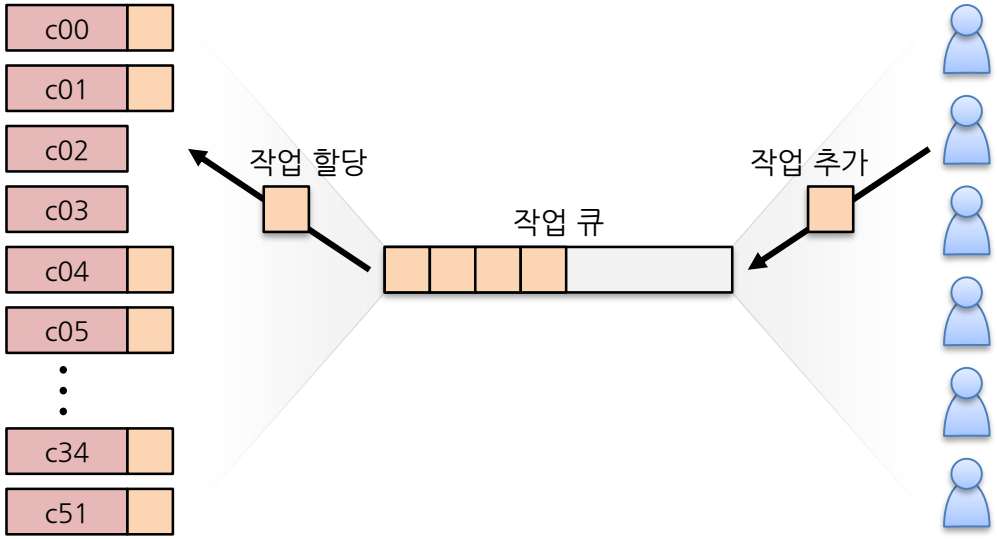
파일 업로드/다운로드

- SFTP 클라이언트 사용
  - Windows의 경우: FileZilla, Xftp, WinSCP, …
  - Mac OS X의 경우: 터미널에서 `scp` 명령 사용
    - `scp file mc00@chundoong0.snu.ac.kr:/home/mc00`
    - `scp mc00@chundoong0.snu.ac.kr:/home/mc00/file .`
- 주소, 포트, 아이디와 패스워드는 SSH 접속 때와 동일
  - 오류 발생시 프로토콜이 sftp로 되어 있는지 확인



프로그램 실행

- 작업 스케줄러 “thor”
  - 북유럽 신화에 나오는 천둥의 신
  - thor를 통해 계산 노드에서 프로그램 실행 가능
    - thor를 사용하지 않고 계산 노드에 직접 접근하는 것은 불가능



작업 추가

```
[mc00@login0 ~]$ thorq --add --mode single ./test 10

Enqueue a new job:
  Mode: single
  Device: cpu
  Base directory: here
  Timeout: 259200
  Task: /home/mc00/test 10
  Path: /home/mc00
  Command string: thorq --add --mode single ./test 10

Job 400000 is Enqueued.
```

작업 상태 확인: 대기 중

```
[mc00@login0 ~]$ thorq --stat 400000
============================================================
ID              : 400000
Name            : task_400000
Status          : Enqueued
Enqueued        : 2014-03-17 01:23:45
Executed        :
Finished        :
Assigned nodes:
Device          : CPU only
# of nodes      : 1
# of slots      : 1
Command string: thorq --add --mode single ./test 10
============================================================
```

작업 상태 확인: 실행 중

```
[mc00@login0 ~]$ thorq --stat 400000
============================================================
ID              : 400000
Name            : task_400000
Status          : Running
Enqueued        : 2014-03-17 01:23:45
Executed        : 2014-03-17 01:24:11
Finished        :
Assigned nodes: c00
Device          : CPU only
# of nodes      : 1
# of slots:     : 1
Command string: thorq --add --mode single ./test 10
============================================================
```

작업 상태 확인: 실행 완료

```
[mc00@login0 ~]$ thorq --stat 400000
============================================================
ID              : 400000
Name            : task_400000
Status          : Finished (success)
Enqueued        : 2014-03-17 01:23:45
Executed        : 2014-03-17 01:24:11
Finished        : 2014-03-17 01:24:13
Elapsed time    : 1.442581 s
Assigned nodes: c00
Device          : CPU only
# of nodes      : 1
# of slots      : 1
Command string: thorq --add --mode single ./test 10
============================================================
```

실행 결과 확인

```
[mc00@login0 ~]$ ls
task_400000.stderr  task_400000.stdout  test  test.c
[mc00@login0 ~]$ cat task_400000.stdout
arg = 10
[mc00@login0 ~]$
```

thorq --add 옵션

- thorq --add와 프로그램 이름 사이에 씀

옵션	설명
--mode mpi	MPI 프로그램 실행
--mode snucl	SnuCL 프로그램 실행
--nodes [n]	n개 노드에서 MPI (혹은 SnuCL) 프로그램 실행
--slots [s]	각 노드마다 s개의 MPI 프로세스 실행
--device gpu/7970 --device gpu/nano --device gpu/290x --device gpu/1080	GPU 사용 (옵션이 없으면 CPU만 사용)
--timeout [t]	t초 후에 강제 종료 (기본값: 259,200초)
--name [name]	작업 이름 지정

GPU 사용하기

```
[mc00@login0 ~]$ thorq --add --mode single --device gpu/7970 ./foo

Enqueue a new job:
  Mode: single
  Device: gpu/7970
  Base directory: here
  Timeout: 259200
  Task: /home/mc00/foo
  Path: /home/mc00
  Command string: thorq --add --mode single --device gpu/7970 ./foo

Job 400001 is Enqueued.
```

GPU 사용하기

- 숙제, 프로젝트
  - 특별한 언급이 없으면 AMD Radeon HD 7970 사용
  - `--device gpu/7970` 옵션으로 실행
  - 다른 GPU를 사용해야 할 경우 스펙에 명시 될 예정

Timeout 설정

```
[mc00@login0 ~]$ thorq --add --mode single --timeout 100 ./foo

Enqueue a new job:
  Mode: single
  Device: cpu
  Base directory: here
  Timeout: 100
  Task: /home/mc00/foo
  Path: /home/mc00
  Command string: thorq --add --mode single --timeout 100 ./foo

Job 400002 is Enqueued.
```

이름 설정

```
[mc00@login0 ~]$ thorq --add --mode single --name thorq_test ./foo

Enqueue a new job:
  Name: thorq_test
  Mode: single
  Device: cpu
  Base directory: here
  Timeout: 259200
  Task: /home/mc00/foo
  Path: /home/mc00
  Command string: thorq --add --mode single --name thorq_test ./foo

Job 400003 is Enqueued.
```

이름 설정

```
[mc00@login0 ~]$ ls
foo  thorq_test.stderr  thorq_test.stdout
[mc00@login0 ~]$
```

기타

```
[mc00@login0 ~]$ thorq --kill 400000
Job 400000 is killed.
[mc00@login0 ~]$ thorq --stat-all
Job 400003 (thorq_test): Running
Job 400004 (task_400004): Running
Job 400005 (task_400005): Enqueued
(end of the list)
[mc00@login0 ~]$ thorq --kill-all
Job 400003 is killed.
Job 400004 is killed.
Job 400005 is killed.
(end of the list)
```

웹 페이지

- 서비스 안내: http://chundoong.snu.ac.kr/
  - 천둥 소개
  - 사용자 매뉴얼

- 웹 인터페이스: http://thor.snu.ac.kr:8080/
  - 실행 중인 작업 확인

기타

- 타 과목 과제에는 사용하지 말아주세요...
  - ML 등등
- 쉘은 bash만 지원
- 소프트웨어 버전이 필요한 버전보다 낮은 경우, 직접 컴파일하거나 로컬 인스톨러로 깔아서 쓸 것
  - pip, cmake 등등
- 기타 문의는 조교에게 (heehoon@aces.snu.ac.kr)