🧐

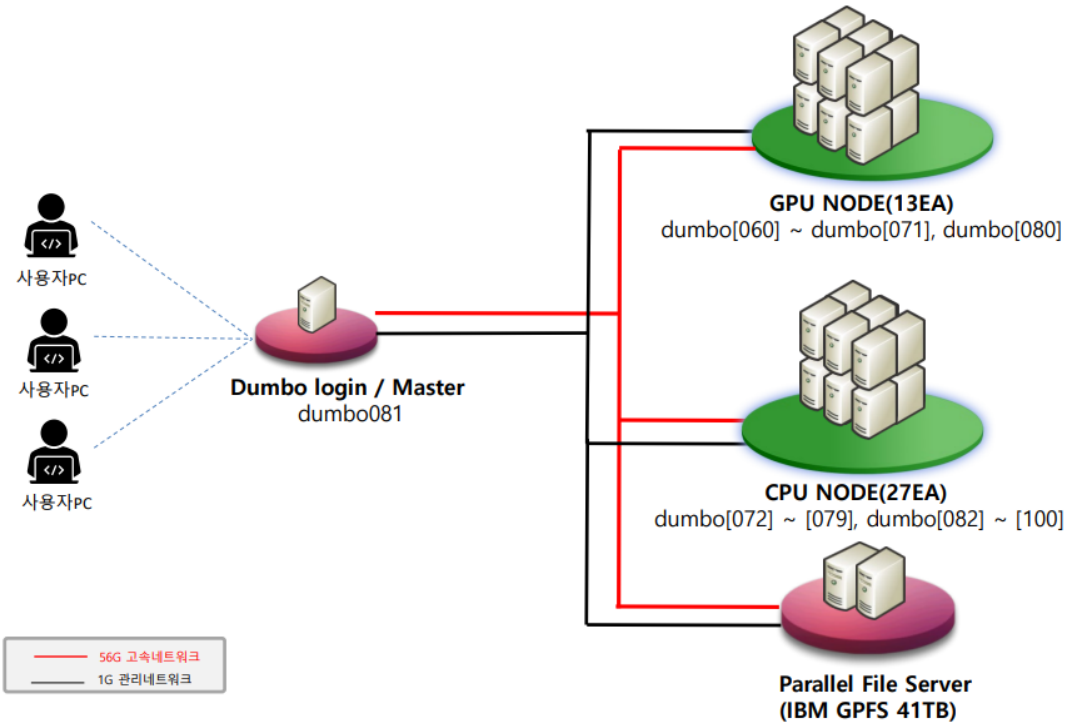
[AICP] GPU 사용법

UNIST HPC-GPU System User Guide

1. HPC(High Performance Computing)

유니스트 슈퍼컴퓨팅 센터(USC) 에서 HPC와 병렬 파일 시스템을 지원하고 있습니다.

- HPC x86_64 **Linux** 클러스터로 Tachyon, Leopard, Lion, Falcon, Eagle, Dumbo이 있으
  며 파일 시스템으로 Lustre parallel filesystem, Spectrum Scale(GPFS)을 보유
- Dumbo GPU Cluster 시스템은 40개의 계산 노드로 구성되어 있고, 각 node 당 16개의 코어를 장착
- CPU architecture는 Haswell 이고, GPU architecture는 nVidia K80 입니다. Dumbo 시스템들은 23개의 CPU 노드와 13개의 GPU 노드가 있습니다.
- 시스템들의 총 코어의 수는 640 코어이고, 노드 당 메모리는 CPU 노드는 32GB, GPU 노드는 64GB의 메모리가 꼽혀 있습니다.
- 각 노드 간 네트워크는 IB 4X FDR(56Gbps)로 구성되어있으며, 공유스토리는 병렬파일시스템인 GPFS를 통해 구성



로그인 방법 : 1개의 로그인 노드를 통해서 접근 가능합니다. SSH protocol(2123 port)를 사용하면 됩니다! **Hostname(DNS), dumbo(dlogin01.usc.unist.ac.kr)**

▼ Dumbo, SSH란 ?

SSH : Secure Shell Protocol, 데이터 전송 / 원격 제어 시에 사용

Dumbo : Linux cluster의 일종으로 다음과 같이 구성되었다고 합니다.

iv. HPC Dumbo CLuster

1) System Overview

A) GPU Nodes

Hostname	dumbo[060] – dumbo[71], dumbo[80]
Number of node(# of core)	13node (208 cores), 16 cores/node
Processor	Intel Xeon Haswell E5-2640 v3 2.60GHz
GPU Processor	Nvidia K80 2ea per node
Memory	64GB per node
OS	Linux CentOS 7.8
Interconnection	Infiniband(56G) network between computing nodes
Storage	/home in GPFS Parallel file system

B) CPU Nodes

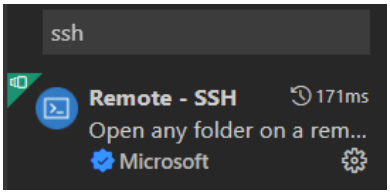
Hostname	dumbo[72] – dumbo[100]
Number of node(# of core)	27node (432 cores), 16 cores/node
Processor	Intel Xeon Haswell E5-2640 v3 2.60GHz
Memory	32GB per node
OS	Linux CentOS 7.8
Interconnection	Infiniband(56G) network between computing nodes
Storage	/home in GPFS Parallel file system

2. 유니스트 서버 접근 방법(VSCode ver.)

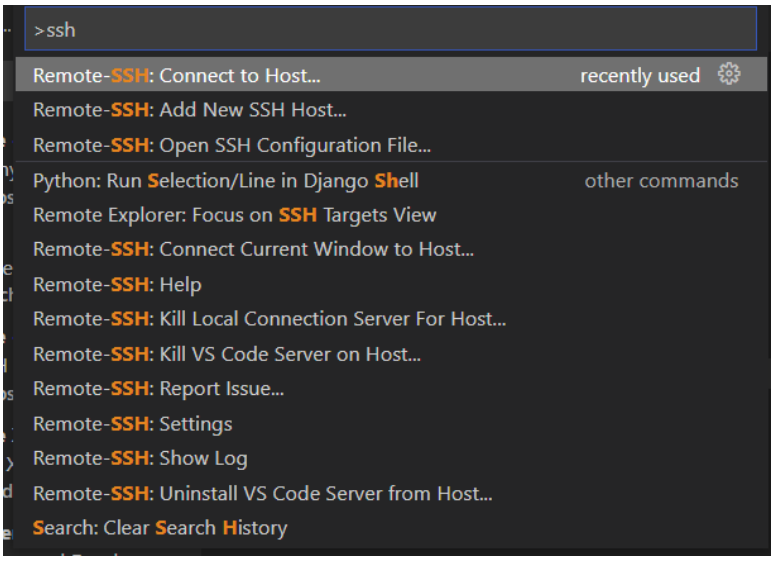
*학교 내부 인터넷으로만 접근 가능(VPN 사용시, AICP 관계자분을 통해서 슈퍼컴퓨팅 센터에 신청(VPN 프로필에서 ip, ssh port 매핑 필요)

▼ **VPN을 통해 USC 서버 접근 방법**(다음주 월요일 메일 넣어서 정리 예정)

1. vscode 확장자 메뉴에서 ‘ssh’ 를 검색하여 이 확장자를 설치합니다.



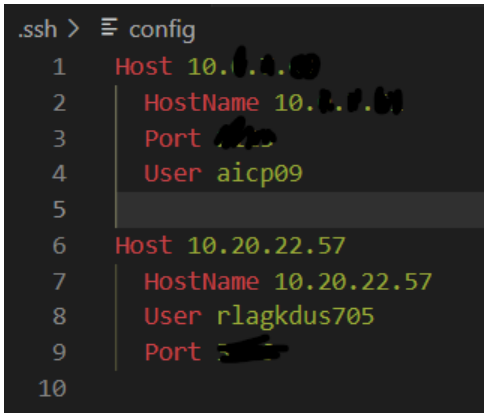
2. 설치 후, Windows는 F1, Mac은 Cmd-P키를 누른후 ‘ssh’를 검색하면 다음과 같은 창이 뜹니다.



3. 처음 등록이니 'Remote-SSH : Add New SSH Host'를 누른 후, 다음과 같이 입력해주세요.

`ex) ssh -p 2222 airflow@localhost` (ssh -p 포트번호 아이디@ip)

4. config 파일 선택 (C:\Users**사용자명**.ssh\config) -> 필요한 정보가 다음과 같이 저장됩니다.



5. 이제 config파일이 등록되었으니 편하게 연결하실 수 있습니다!(F1 / Cmd-P 버튼을 누르고 'Remote-SSH:Connect to Host'를 선택한 후 해당 host를 클릭하면 간편하게 접속 가능합니다)

3. Module 사용법(코드 실행)

: GPU에서 모델을 학습하기 위해서 우리는 반드시 CONDA 환경을 이용하여야 합니다. 이때, tensorflow/pytorch는 따로 설치해야하는 번거로움을 느끼는데, 이를 덜기 위해서 필요한 모듈을 책장에서 책을 꺼내듯 사용할 수 있습니다.

1. 현재 시스템에 올려져있는 module을 보기 위해서 `module list` 명령어를 작성합니다. 이때, 아직 load한 module이 없기 때문에, 아무것도 존재하지 않습니다. 이후, 필요한 module을 둘러보기 위해서 `module avail`명령어를 실행합니다. 책장에 많은 책이 꽂혀있듯 여러 module을 확인할 수 있습니다.

```
[aicp09@dumbo ~]$ module list
No Modulefiles Currently Loaded.
[aicp09@dumbo ~]$ module avail

------------------------------------------------------------------------------------------------------------------------ /apps/Modules/modulefiles/compilers ------
gcc/10.3.0   gcc/4.8.5    gcc/8.4.0    intel/19.5   intel/20.4.9

------------------------------------------------------------------------------------------------------------------------ /apps/Modules/modulefiles/libraries ------
cmake/3.21.1   cuda/10.2      gromacs/2021.3 hdf4/4.2.13    hdf5/1.10.2    lapack/3.9.0   netcdf/4.6.1

------------------------------------------------------------------------------------------------------------------------ /apps/Modules/modulefiles/mpi ---------
cudampi/openmpi-3.1.6 mpi/impi-19.5         mpi/impi-20.4.9       mpi/mvapich2-2.3.4    mpi/openmpi-3.1.6

------------------------------------------------------------------------------------------------------------------------ /apps/Modules/modulefiles/conda_packages ----
conda/pytorch        conda/tensorflow-cpu conda/tensorflow-gpu
```

2. 이번에는 ‘module load [modulefile_name]’ 사용 가능한 모듈 중 필요한 모듈을 불러옵니다. pytorch와 tensorflow-cpu를 같이 부르면 충돌하기 때문에, 기존의 pytorch를 unload 한 후 tensorflow-cpu를 load해야 충돌이 일어나지 않습니다.(`module switch 현재파일 바꿀파일`)로 간단하게 module 변경도 가능합니다.)

```
[aicp09@dumbo ~]$ module load conda/pytorch
[aicp09@dumbo ~]$ module load conda/tensorflow-cpu
conda/tensorflow-cpu(9):ERROR:150: Module 'conda/tensorflow-cpu' conflicts with the currently loaded module(s) 'conda/pytorch'
conda/tensorflow-cpu(9):ERROR:102: Tcl command execution failed: conflict conda

[aicp09@dumbo ~]$ module unload conda/pytorch
[aicp09@dumbo ~]$ module load conda/tensorflow-cpu
```

3. ‘module show [modulefile_name]’ 으로 module의 설명도 자세히 볼 수 있습니다.

```
[aicp09@dumbo ~]$ module show conda/tensorflow-cpu
-------------------------------------------------------------------
/apps/Modules/modulefiles/conda_packages/conda/tensorflow-cpu:

conflict        conda
prepend-path    PATH /apps/application/miniconda3/envs/tensorflow-cpu/bin
-------------------------------------------------------------------
```

4. 새로운 파이썬 버전에서 pytorch / tensorflow 모듈을 사용 원하는 경우, 새로운 가상환경을 만든 후(`conda create -n “가상환경 이름”` → `conda create “가상환경 이름”`) 그 환경에서 원하

는 파이썬 버전을 설치합니다(`conda install python = 버전`). 이후 원하는 모듈(pytorch/tensorflow)를 load 합니다.(`module load conda/pytorch`) 그러면, 따로 새로운 가상환경에 pytorch를 설치할 필요 없이 사용할 수 있습니다.

*새로운 가상환경에서 load module을 한 후, root로 돌아오면 module이 잘 작동하지만, root에서 load module을 한 후, 새로운 가상환경을 다시 들어가면 module이 잘 작동하지 않는 문제가 발생합니다.(이런 경우, module을 다시 unload한 후 load 해주세요!)

4. Slurm(Job script) 사용법

: USC에서는 사용자가 code를 실행하기 위해서 queue system을 사용해야 합니다. 간단하게 말하면, 서버에 '이런 일을 해주렴~' 하고 할 일을 적어서 보내주는 일입니다.

1. slurm에서 GPU할당을 확인하기 위해서 예제로 다음과 같은 코드를 실행할 것입니다.

```
import torch

print(torch.cuda.is_available()) #GPU를 사용한다면 True 출력
```

2. [파일명].sh로 job shell script를 작성합니다.

```
#!/bin/bash
#SBATCH -J GAN #Job이름
#SBATCH -p gpu_edu #gpu 할당
#SBATCH -N 1 #노드1개 (고정)
#SBATCH -n 4 #cpu4개(고정)
#SBATCH -o %x.o%j #-o = output, x = GAN(Job이름), j = JobID
#SBATCH -e %x.e%j #-e = error
#SBATCH --time 48:00:00
#SBATCH --gres=gpu:1 #GPU 개수

module purge
module load intel/20.4.9 cuda/10.2 cudampi/openmpi-3.1.6 conda/pytorch #필요 모듈

python ./test.py #파일 실행 명령어
```

▼ ***작업 스크립트 예제(학교 예시)***

v. **sbatch** (submit a job)

$sbatch ./job_script.sh

```
[작업 스크립트 예제, GPU MPI 기준]
#!/bin/sh
#SBATCH -J thin_lfts          ## Job Name
#SBATCH -p gpu_k80            ## Partition name
#SBATCH -N 5                  ## node count 총 필요한 컴퓨팅 노드 수
#SBATCH -n 20                 ## total number of tasks across all nodes 총 필요한 프로세스 수
#SBATCH -o %x.o%j             ## filename of stdout, stdout 파일 명(.o)
#SBATCH -e %x.e%j             ## filename of stderr, stderr 파일 명(.e)
#SBATCH --time 120:00:00      ## 최대 작업 시간(Wall Time Clock Limit)
#SBATCH --exclusive=user      ## Only Use GPU Partition, GPU 파티션 사용시에만 적용한다.
#SBATCH --gres=gpu:4          ## number of GPU(s) per node

source /etc/profile.d/modules.sh
module load intel/20.4.9 mip/impi-20.4.9    ##필요한 Module 로드
mpirun ./lfts_cuda_single.out inputs        ##해석 실행 코드 입력
```

3. `sbatch ./[Job이름]` 명령어를 실행하면 다음과 같이 jobID(6601)를 부여해주고, job을 제출합니다.

```
[aicp09@dumbo GAN-pytorch]$ sbatch ./GAN
Submitted batch job 6601
```

4. `squeue` 명령어를 사용하면 현재 Job Schedule을 확인할 수 있고, `scontrol show job [JobID]` 명령어를 사용하면 Job에 대한 상세 정보를 확인할 수 있습니다.

5. job의 실행이 끝나면 [Job이름].e[JobID], [Job이름].o[JobID] 파일이 생겨서 각각 에러코드와 결과를 확인할 수 있습니다.

```
GAN-pytorch > GAN.o6601
1    True
2
```