Lab1
Platform Introduction

Introduction to Parallel Computing
2022/02/22

Outline

- Platform introduction - Apollo
- Login to Apollo
- Linux command
- Job submission
- Lab1

Platform instruction - Apollo

- 19 nodes for this course (apollo31 - 48, 50)
- Intel X5670 2x6 cores @ 2.93GHz
- 96GB RAM (each node)
- 5.5TB shared RAID5 disk
- QDR Infiniband

Software

- OS: Arch Linux
- Compilers: GCC 10.2.0, Clang 11.0.1
- MPI: Intel MPI Library, Version 2019 Update 8
- Scheduler: Slurm 20.02.5

Available resources

- 1 login node (apollo31) (200%CPU max)
- 18 compute nodes (1200% CPU max)
- Use `squeue` to view SLURM usage
- Cluster monitor: http://apollo.cs.nthu.edu.tw/monitor
- 48GB disk space per user
- Use `quota -s` to view disk quota

Login to Apollo

- Address: apollo.cs.nthu.edu.tw
- Username: check email
- Password: check email
- MINING IS PROHIBITED. Also, do not attack the server.

SSH - Linux and Mac

- Open terminal
- `ssh ipc22sXX@apollo.cs.nthu.edu.tw`
- Enter password
- You'll be ask to change your password on first login

SSH - Windows

- Tools
  - MobaXterm
  - Putty
  - Cmd or Powershell (Windows 10)
- `ssh ipc22sXX@apollo.cs.nthu.edu.tw`
- Enter password
- You'll be ask to change your password on first login

Optional: Make SSH Login Easier - Part 1

1. Create a SSH key pair on your computer if you don't have any:

   `ssh-keygen -t ed25519`

   This command creates:

   a. a **private key** at `~/.ssh/id_ed25519`

   b. and the corresponding **public key** at `~/.ssh/id_ed25519.pub`
   (the contents should be a single line)

<hint>
~ means your home directory

2. Then you can **either**:

   a. Run `ssh-copy-id` locally, and follow the instructions

   b. Login to the server and paste the contents of the **public key** into `~/.ssh/authorized_keys`

3. Future logins will be based on keys and without passwords

Optional: Make SSH Login Easier - Part 2

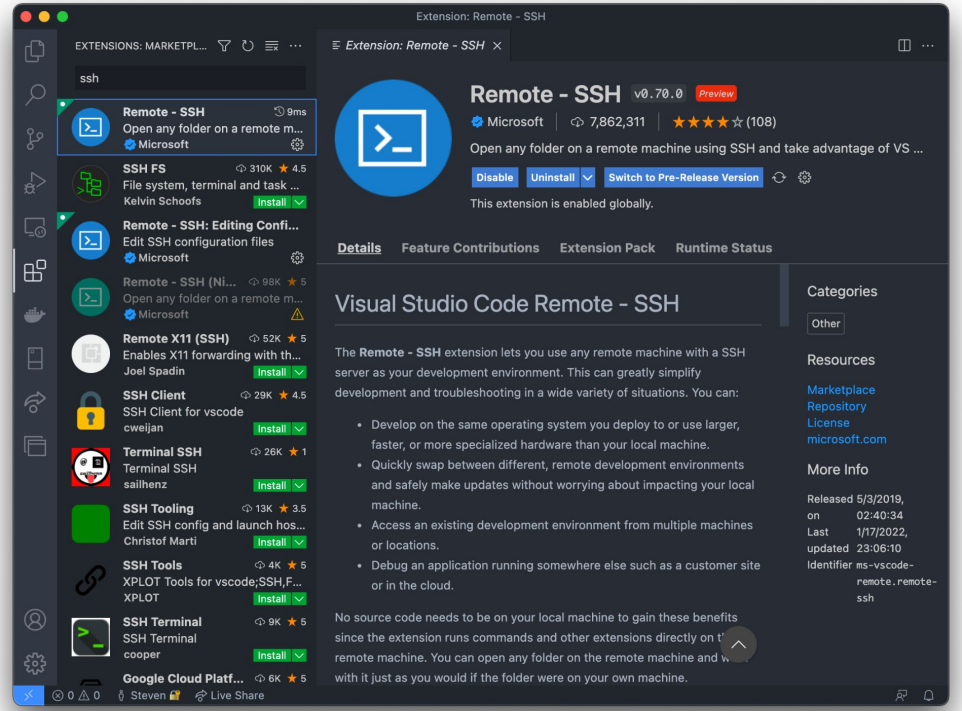
`ssh ipc22sxx@apollo.cs.nthu.edu.tw` is very long.

You can create a file named `~/.ssh/config` (if it does not exist yet) and add the following content:

```
Host apollo
Hostname apollo.cs.nthu.edu.tw
User ipc22sxx
```

Then you can instead type `ssh apollo`. ssh will replace it with the actual hostname and user for you.

Optional: VSCode Plugin



Some useful command

- Login: `ssh ipc22sXX@apollo.cs.nthu.edu.tw`
- Editors: `vim` `emacs` `nano`
- SLURM usage: `squeue`
- Disk quota: `quota -s`
- Change password: `passwd`
- Download file: `wget` `aria2c`
- Code syntax highlighting: `pygmentize`

Using rsync or scp (run these on your computer!)

`rsync -ahP apollo:code.c .`
➜ Download code.c from apollo to the current directory

`rsync -ahP apollo:homework backup` (assuming homework is a directory)
➜ Download the entire homework directory on apollo to the local directory named backup

`scp -r apollo:homework backup`
➜ Same as above, but you must start over if connection fails

`rsync -ahP ~/Documents/hw1.cc apollo:homework/hw1/`
➜ Upload the source code hw1.cc which is in the local Documents folder to the directory ~/homework/hw1 on apollo

Job submission

SLURM workload scheduler: On a cluster system, there are multiple users and multiple nodes. SLURM schedules jobs submitted by users across different nodes, so that the same resource is not used by two jobs at the same time (to ensure accuracy of performance-critical experiments), and also ensure the utilization of the cluster.

SLURM prefer the following jobs:

- short jobs (you can set time limit)
- less resource demanding jobs
- jobs queued for a long time
- users that haven't run a lot of jobs recently

Job submission using srun

- `srun [options] ./executable [args]`
- Options:
  - `-N NODES`: NODES is the number of nodes to run the job
  - `-n PROCESSES`: PROCESSES is number of total processes to launch
  - `-c CPUS`: CPUS is the number of cpus available to each process
  - `-t TIME`: The time limit in "minutes" or "minutes:seconds"
  - `-J NAME`: The name of the job. Will be displayed on squeue

srun - submit jobs

```
srun ./a.out
```
➜ run a.out via Slurm

```
srun -c4 ./a.out
```
➜ Give 4 CPUs to a.out

```
srun -n3 ./a.out
```
➜ Run 3 a.out processes

```
srun -n4 -c2 ./a.out
```
➜ Run 4 a.out processes, with 2 CPUs allocated to **each process**. Totally 8 CPUs.

Job submission using sbatch

- Using sbatch command to submit jobs in the background
- You can write a simple script to do that

```
#!/bin/bash
#SBATCH -n 4
#SBATCH -N 2
srun ./hello
```

- `$ sbatch script.sh`

Job control

- `sinfo`: view status of nodes
- `squeue`: view submitted jobs in queue
- `scancel JOBID`: cancel a job with its JOBID

Lab Spec

- Finish: https://forms.gle/47G2ModGYUgPzKkd9
- Due 2022/03/01 23:59