Parallel Programming Lab1

...

2018/9/27

afg, zlsh80826

Conventions used in this slide

Descriptive text is serif.

Commands and codes are `monospaced`.

Codes that require modifications to work are *`italic`*.

You do not need to turn in “Practice”s.

Useful commands

- Login: `ssh mewtwo@140.114.91.183`
- File transfer: `rsync -ahP mydir/ mewtwo@140.114.91.183:mydir/`
- SLURM usage: `smap -i 1`
- Disk quota: `quota -s`
- Change password: `passwd`
- Navigation: `cd` `ls` `vim` `mv` `rm` `cp`
- Compilers: `gcc` `g++` `mpicc` `mpicxx`
- Download a file: `aria2c`

Outline

- Platform introduction - apollo
- Login to apollo
- Compile & execute program on apollo
- Lab - Approximate the value of π using MPI

Platform introduction

20 nodes:

- Intel X5670 2x6 cores @ 2.93GHz
- 96GB RAM
- 5.5TB shared RAID5 disk
- Infiniband network

workload scheduler: SLURM

Available resources

- 1 login node (200%CPU max)
- 19 compute nodes (1200% CPU max)
  - Use `smap -i 1` to view SLURM usage
  - Cluster monitor: http://140.114.91.183/monitor
- 48GB disk space per user
  - Use `quota -s` to view disk quota

Outline

- Platform introduction - apollo
- Login to apollo
- Compile & execute program on apollo
- Lab - Approximate the value of π using MPI

Login information

IP : 140.114.91.183

帳號 : 已經寄信給同學

密碼 : 已經寄信給同學

課程	平行程式Parallel Programming(10710CS542200)
寄件人	尤立宇, afg984@gmail.com (此信由系統信箱自動發
時間	2018-09-27 00:36
內容	尤立宇同學您好， 您在平行程式課程的工作站帳號及密碼如下： 帳號：mewtwo 密碼：aU3kjc4q 工作站入口：140.114.91.183 第一次登入時，請依系統指示重設密碼。 如果有任何疑問，請和助教聯絡。 明天晚上 Lab 會再說明如何使用本課程的工作站。

SSH with Unix systems {Linux,BSD,macOS}

- Open the terminal
- `ssh mewtwo@140.114.91.183`

File transfer

- `rsync -ahP mydir/ mewtwo@140.114.91.183:mydir/`

SSH from windows

- MobaXterm
  http://mobaxterm.mobatek.net/download-home-edition.html
- Putty
  https://www.putty.org/

Practice

- Login to the server

Notes:

- On first login, you will be forced to reset your password
- If you want to change your password again, you can use: `passwd`

Outline

- Platform introduction - apollo
- Login to apollo
- Compile & execute program on apollo
- Lab - Approximate the value of π using MPI

Compilers on apollo

- `gcc` -- C compiler
- `g++` -- C++ compiler
- `mpicc` -- MPI C compiler wrapper
- `mpicxx` -- MPI C++ compiler wrapper

Compile a hello world MPI program

To compile a program, use:

```
mpicc hello.c -O3 -o  hello
```

You can download the code at:

https://www.open-mpi.org/papers/workshop-2006/hello.c

Command: `aria2c https://www.open-mpi.org/papers/workshop-2006/hello.c`

Run a hello world MPI program

Use srun / sbatch / salloc:

```
$ srun -n 4 ./hello
Hello, World.  I am 1 of 4
Hello, World.  I am 3 of 4
Hello, World.  I am 0 of 4
Hello, World.  I am 2 of 4
```

Practice

- Run the MPI hello world program.

SLURM job queues: “partitions”

Two queues:

- debug (default) -- for quick debugging
- batch -- for benchmarking

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up       5:00     19   idle apollo[32-50]
batch        up      15:00     17   idle apollo[34-50]
```

Partition constriants (*: subject to change)

	debug	batch
max nodes per job	2	4
max cpu per node	4	12
Max time	5 min	15 min
Max running jobs	1	unlimited
Max submitted jobs	2	unlimited*

Job priority

SLURM will favor:

- **debug** partition
- **short running** 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 (srun)

`srun` *`options`* `./executable args…`

options	usage
-p *PARTITION*	*PARITION* should be either debug of batch
-N *NODES*	*NODES* is the number of nodes to run the job
-n *PROCESSES*	*PROCESSES* is number of **total processes** to launch
-c *THREADS*	*THREADS* is the number of thread **per process**
-t *TIME*	The time limit in "minutes" or "minutes:seconds"
-J *NAME*	The name of the job. Will be displayed on `squeue`

Job Submission (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
#SBATCH -p batch
srun ./hello_world
```

- After writing the script, run the command

```
$ sbatch job.sh
```

Job Submission (sbatch)

- sbatch also works without a script with the --wrap option:

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

```
$ sbatch -N 2 -n 4 --warp=”srun ./hello_world”
```

Job control

- `sinfo`
  view status of nodes
- `squeue`
  view submitted jobs in queue
- `scancel` *`JOBID`*
  cancel a submitt job with its JOBID
- `smap -i 1`
  view the jobs and partitions

sinfo

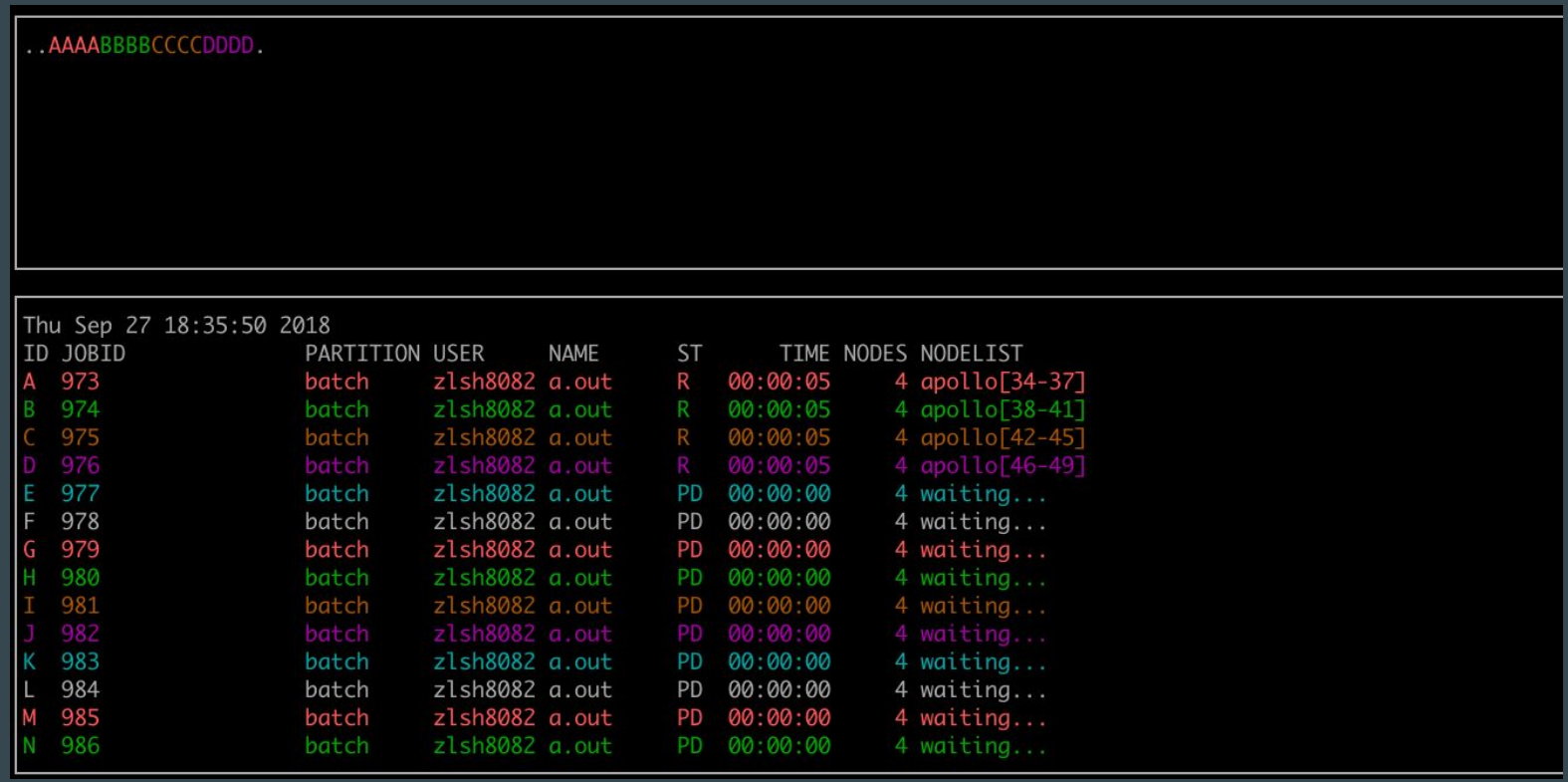
- If you see the STATE are drain or down, please contact TA

```
[18:30:07] zlsh80826@apollo31 /home/zlsh80826 (130)
> sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up       5:00     19   idle apollo[32-50]
batch        up      15:00     17   idle apollo[34-50]
```

squeue

```
[18:32:00] zlsh80826@apollo31 /home/zlsh80826 (0)
> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               972     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               971     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               970     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               969     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               968     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               967     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               966     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               965     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               964     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               963     batch    a.out zlsh8082 PD       0:00      4 (Priority)
               962     batch    a.out zlsh8082 PD       0:00      4 (Resources)
               958     batch    a.out zlsh8082  R       0:03      4 apollo[34-37]
               959     batch    a.out zlsh8082  R       0:03      4 apollo[38-41]
               960     batch    a.out zlsh8082  R       0:03      4 apollo[42-45]
               961     batch    a.out zlsh8082  R       0:03      4 apollo[46-49]
```

smap -i 1


..AAAABBBBCCCCDDDD.
Thu Sep 27 18:35:50 2018
ID JOBID PARTITION USER NAME ST TIME NODES NODELIST
A 973 batch zlsh8082 a.out R 00:00:05 4 apollo[34-37]
B 974 batch zlsh8082 a.out R 00:00:05 4 apollo[38-41]
C 975 batch zlsh8082 a.out R 00:00:05 4 apollo[42-45]
D 976 batch zlsh8082 a.out R 00:00:05 4 apollo[46-49]
E 977 batch zlsh8082 a.out PD 00:00:00 4 waiting...
F 978 batch zlsh8082 a.out PD 00:00:00 4 waiting...
G 979 batch zlsh8082 a.out PD 00:00:00 4 waiting...
H 980 batch zlsh8082 a.out PD 00:00:00 4 waiting...
I 981 batch zlsh8082 a.out PD 00:00:00 4 waiting...
J 982 batch zlsh8082 a.out PD 00:00:00 4 waiting...
K 983 batch zlsh8082 a.out PD 00:00:00 4 waiting...
L 984 batch zlsh8082 a.out PD 00:00:00 4 waiting...
M 985 batch zlsh8082 a.out PD 00:00:00 4 waiting...
N 986 batch zlsh8082 a.out PD 00:00:00 4 waiting...

Practice

- Run MPI with sbatch
  with sbatch, using 4 nodes and 8 processes per node,
  submit the job to batch partition

Outline

- Platform introduction - apollo
- Login to apollo
- Compile & execute program on apollo
- Lab - Approximate the value of π using MPI

MPI_Send

```
int MPI_Send(const void *buf,
             int count,
             MPI_Datatype datatype,
             int dest,
             int tag,
             MPI_Comm comm)
```

MPI_Recv

```
int MPI_Recv(void *buf,
             int count,
             MPI_Datatype datatype,
             int source,
             int tag,
             MPI_Comm comm, MPI_Status *status)
```

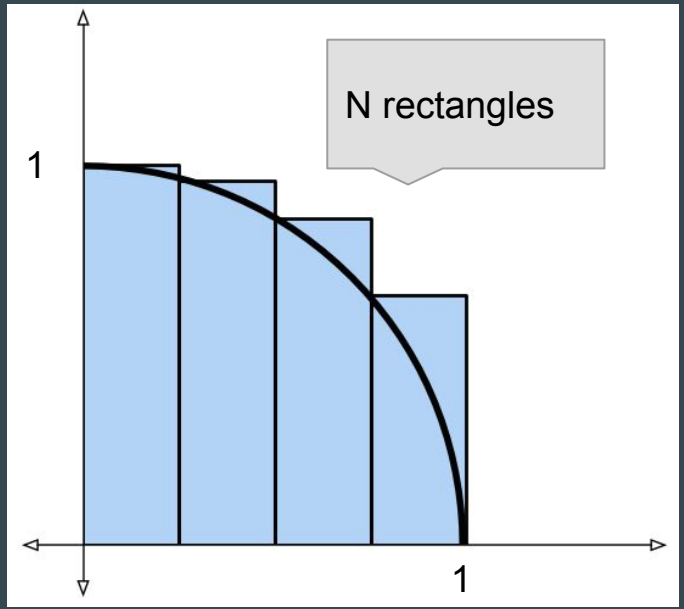
MPI_Reduce

```
int MPI_Reduce(const void *sendbuf,
               void *recvbuf, int count,
               MPI_Datatype datatype, MPI_Op op, int root,
               MPI_Comm comm)
```

Approximate the value of π using MPI

- We can approximate the value of π by Riemann sum

$$Area = \frac{1}{4}\pi \approx \sum_{x=0}^{N-1} \frac{1}{N}\sqrt{1-\left(\frac{x}{N}\right)^2}$$



Lab spec

- Your pi program should locate at ~/homework/lab1
- You can write with C/C++ (pi.c or pi.cc)
- Your program should accept an argument N, means the number of slices

  `$ srun` *`options...`* `./pi` *`N`*
- Output your value of π with only one process
- Use `lab1-judge` to submit your code
- Visit http://140.114.91.183/scoreboard/lab1/ for results