

Pthread & OpenMP

Parallel Programming Lab2-2
Oct 20 2022

Lab2 Tasks

— — —

- We are going to **approximate pixels** using pthread, OpenMP and hybrid of MPI and OpenMP in this lab

- **Deadline of the Lab2 is 10/27 23:59**
- All sample codes and test cases are provided at `/home/pp22/share/lab2`
- `testcases_1` is for practice 1, `testcases_2` is for practice 2, and so on.
- Check your codes with `lab2_pthread-judge`, `lab2_omp-judge`, `lab2_hybrid-judge`
- Scoreboard: [pthread](#), [OpenMP](#), [hybrid](#)
- Hand in your code(three files) to eeclass. TA will check your code after deadline.

SLURM quick reference

```
srun [flags] ./prog
```

```
===== or =====
```

```
#!/bin/bash
```

```
#SBATCH [flags]
```

```
srun ./prog # (MPI)
```

```
./prog # (non-MPI)
```

```
----- run with: -----
```

```
sbatch job.sh
```



[flags]:

- N number of nodes
- n number of processes
- c CPUs per process**
- t additional time limit
- J name of job

Outline

— — —

- Pthread
 - Hello world
 - Mutex
- OpenMP
- MPI + OpenMP

Running pthread programs on apollo

SYNOPSIS

```
#include <pthread.h>
```

```
int pthread_create(  
    pthread_t *thread, const pthread_attr_t *attr,  
    void *(*start_routine) (void *), void *arg);
```

Type ``man pthread_create`` in terminal to see this

Compile and link with **-pthread**.

Running pthread programs on apollo

— — —

Example code

`/home/pp22/share/lab2/sample/hello_pthread.cc`

Compile

`g++ hello_pthread.cc -o hello_pthread -pthread`

NOT `-lpthread`

Execute

`srunk -c4 -n1 ./hello_pthread 4`

`-c4` means 4 CPUs per process

`-n1` means 1 process

You can use sbatch as well!

```
pp21t00@apollo31 ~/l/sample> srunk -c4 -n1 ./hello_pthread 4
In main: creating thread 0
In main: creating thread 1
Hello, thread #0!
In main: creating thread 2
Hello, thread #1!
In main: creating thread 3
Hello, thread #2!
Hello, thread #3!
```

Outline

— — —

- **Pthread**
 - Hello world
 - **Mutex**
- OpenMP
- MPI + OpenMP

Pthread Lock/Mutex Routines

- To use mutex, it must be declared as of **type pthread_mutex_t** and initialized with **pthread_mutex_init()**
- A mutex is destroyed with **pthread_mutex_destroy()**
- A critical section can then be protected using **pthread_mutex_lock()** and **pthread_mutex_unlock()**
- Example:

```
#include "pthread.h"
pthread_mutex_t mutex;
pthread_mutex_init (&mutex, NULL);
pthread_mutex_lock(&mutex);

    Critical Section

pthread_mutex_unlock(&mutex);
pthread_mutex_destroy(&mutex);
```

specify default attribute for the mutex

// enter critical section

// leave critical section

Mutex

— — —

man pthread_mutex_init

```
#include <pthread.h>
```

```
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

```
int pthread_mutex_trylock(pthread_mutex_t *mutex);
```

```
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

man pthread_mutex_lock

[Practice 1] Approximate pixels using pthread

— — —

★ Modify the sequential code `lab2_pthread.cc` with **pthread**

[example commands]

```
g++ lab2_pthread.cc -o lab2_pthread -pthread -lm  
srun -c4 -n1 ./lab2_pthread r k
```

You can also use **Makefile** to compile your code!



Outline

— — —

- Pthread
 - Hello world
 - Mutex
- **OpenMP**
- MPI + OpenMP

Running OpenMP programs on apollo

Example code

`hello_omp.cc`

Compile

`g++ hello_omp.cc -o hello_omp -fopenmp`

Execute

`srun -c4 -n1 ./hello_omp`

OpenMP automatically detects
number of CPUs from SLURM
(affinity)
So we don't have to specify
it again

Try different number of threads!

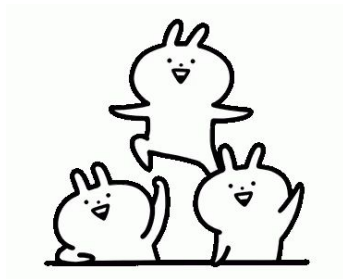
```
pp21t00@apollo31 ~/l/sample> srun -c4 -n1 ./hello_omp
Hello: thread 1/ 4
Hello: thread 0/ 4
Hello: thread 3/ 4
Hello: thread 2/ 4
```

[Practice 2] Approximate pixels using OpenMP

★ Modify the sequential code `lab2_omp.cc` with **OpenMP**

★ Try yourself to see the effect of changing

- ◆ dynamic/static scheduling
- ◆ chunk size
- ◆ number of threads



[example commands]

```
g++ lab2_omp.cc -o lab2_omp -fopenmp -lm
```

```
srn -c4 -n1 ./lab2_omp r k
```

Outline

— — —

- Pthread
 - Hello world
 - Mutex
- OpenMP
- **MPI + OpenMP**

Running Hybrid MPI and OpenMP programs on apollo

Example code

hello_hybrid.cc

Compile

We're using MPI

We're using OpenMP

mpicxx hello_hybrid.cc -o hello_hybrid **-fopenmp**

Execute

srun -c3 -n2 ./hello_hybrid

3 threads

2 processes

```
pp21t00@apollo31 ~/l/sample> srun -c3 -n2 ./hello_hybrid
Hello apollo32: rank 1/ 2, thread 0/ 3
Hello apollo32: rank 1/ 2, thread 2/ 3
Hello apollo32: rank 1/ 2, thread 1/ 3
Hello apollo32: rank 0/ 2, thread 2/ 3
Hello apollo32: rank 0/ 2, thread 1/ 3
Hello apollo32: rank 0/ 2, thread 0/ 3
```

[Practice 3] Approximate pixels using MPI and OpenMP

- — —
- ★ Modify the sequential code `lab2_hybrid.cc` with **MPI and OpenMP**
 - ◆ You can refer to your code in lab1!

[example commands]

```
mpicxx lab2_hybrid.cc -o lab2_hybrid -fopenmp -lm  
srun -N2 -n6 -c4 ./lab2_hybrid r k
```

