Lab3
Pthread & OpenMP

22 Oct 2020

Parallel Programming

Lab3 Tasks

- Practice 1: Approximate pixels using pthread
- Practice 2: Approximate pixels using OpenMP
- Practice 3: Approximate pixels using MPI & OpenMP

- **Deadline of the 3 practices is 10/29 23:59**
- Check your codes with **lab3_pthread-judge、lab3_omp-judge、lab3_hybrid-judge**
- Hand in your code(three files) to ILMS. 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
- OpenMP + MPI

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);

       Compile and link with -pthread.
```

Type `man pthread_create` in terminal to see this

Running pthread programs on apollo

```
cp /home/pp20/share/lab3/sample/hello_pthread.c .
```

compile

```
    gcc hello_pthread.c -o hello_pthread -pthread
```

NOT
`-lpthread`

execute

```
    srun -c4 -n1 ./hello_pthread 4
```

-c4 means 4 CPUs per process

-n1 means 1 process

You can use sbatch as well

Outline

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

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"                          specify default
pthread_mutex_t   mutex;                      attribute for the mutex
pthread_mutex_init (&mutex, NULL);
pthread_mutex_lock(&mutex);                   // enter critical section

        Critical Section

pthread_mutex_unlock(&mutex);                 // leave critical section
pthread_mutex_destroy(&mutex);
```

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

Mutex:
[Practice 1] approximate pixels using pthread

```
g++ lab3_pthread.c -o lab3_pthread -pthread -lm
```

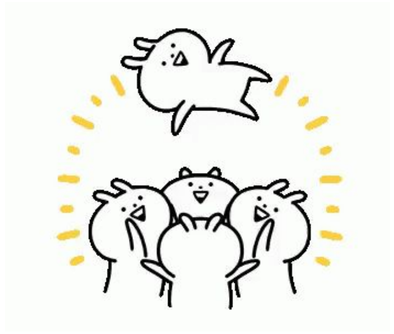
code filename

executable filename

we’re using pthread

```
srun -c4 -n1 ./lab3_pthread r k
```

number of threads



Sequential code at /home/pp20/share/lab3/sample/lab3_pthread.cc

Outline

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

Running OpenMP programs on apollo: example (/home/pp20/share/lab3/sample/hello_omp.c)

compile

```
    gcc hello_omp.c -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

-c4 means 4 CPUs per process

-n1 means 1 process

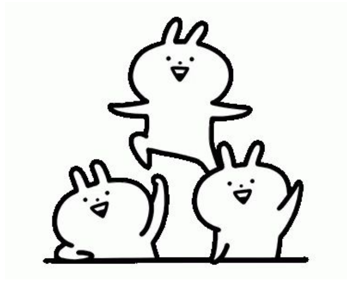
You can use sbatch as well

Try different number of threads!

[Practice 2] OpenMP

1. Modify the sequential lab3_omp.c with openmp

2. Try to see the effect of changing
   dynamic/static **scheduling**
   chunk size
   number of **threads**



[example commands]

```
g++ -lm lab3_omp.cc -o lab3_omp -fopenmp
srun -c4 -n1 ./lab3_omp r k
```

Outline

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

Hybrid MPI and OpenMP program

```
mpicc hello_hybrid.c -o hello_hybrid -fopenmp
```

We're using MPI

We're using OpenMP

```
srun -c3 -n2 ./hello_hybrid
```

3 threads

2 processes

Hybrid MPI and OpenMP program: Hello World

```
srun -c3 -n2 -N2 ./hello_hybrid
Hello apollo32: rank  0/ 2, thread  0/ 3
Hello apollo32: rank  0/ 2, thread  1/ 3
Hello apollo32: rank  0/ 2, thread  2/ 3
Hello apollo33: rank  1/ 2, thread  0/ 3
Hello apollo33: rank  1/ 2, thread  1/ 3
Hello apollo33: rank  1/ 2, thread  2/ 3
```

Hybrid MPI and OpenMP program: **[Practice 3]** Approximate pixels

Use MPI and OpenMP to approximate pixels

(You can refer to your code in lab1)

```
mpicxx hybrid_pi.cc -o lab3_hybrid -fopenmp -lm
srun -N2 -n6 -c4 ./lab3_hybrid r k
```