School of Artificial Intelligence and Advanced Computing
Xi'an Jiaotong-Liverpool University

Xi'an Jiaotong-Liverpool University 西交利物浦大學

DTS202TC Foundation of Parallel Computing

Lab 4: Shared-Memory Programming with OpenMP

This lab weight 35 marks of the A3 lab report, please save all screenshots of activities, source code for future reference.

Installation

You should be able to compile OpenMP hello world program on Windows Cygwin as follows:

```
gcc -g -Wall -fopenmp -o omp_hello omp_hello.c
```

Unfortunately, we can't do that on Mac. Mac users may find this https://stackoverflow.com/questions/35134681/installing-openmp-on-mac-os-x-10-11 helpful. Basically, if you have the Xcode command line tool installed. You can try the following:

```
clang -Xpreprocessor -fopenmp -o omp_hello omp_hello.c -lomp
```

Alternately, you can reinstall the gcc using Brew:

```
brew reinstall gcc --without-multilib
gcc -g -Wall -fopenmp -o omp_hello omp_hello.c
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

void Hello(void);  /* Thread function */

int main(int argc, char* argv[]) {
   /* Get number of threads from command line */
   int thread_count = strtol(argv[1], NULL, 10);

#  pragma omp parallel num_threads(thread_count)
   Hello();

   return 0;
}  /* main */

void Hello(void) {
   int my_rank = omp_get_thread_num();
   int thread_count = omp_get_num_threads();

   printf("Hello from thread %d of %d\n", my_rank, thread_count);

}  /* Hello */
```

Figure 1: OpenMP HelloWorld

OpenMP Hello World (5 marks)

Type the above HelloWorld source code manually in a editor,

Compile and run the program 3 times and observe the results. Search for the api that prints out the maximum number of threads available, put it in the code above so that it prints out the value.

Estimate π with OpenMP (20 marks)

Suppose we toss darts randomly at a square dartboard, whose bullseye is at the origin, and whose sides are 2 feet in length. Suppose also that there's a circle inscribed in the square dartboard. The radius of the circle is 1 foot, and it's area is π square feet. If the points that are hit by the darts are uniformly distributed (and we always hit the square), then the number of darts that hit inside the circle should approximately satisfy the equation

$$\frac{\text{number in circle}}{\text{total number of tosses}} = \frac{\pi}{4} \tag{1}$$

We can use this formula to estimate the value of π with a random number generator:

```
number_in_circle = 0;
for (toss = 0; toss < number_of_tosses; toss++) {
  x = random double between -1 and 1;
  y = random double between -1 and 1;
  distance_squared = x*x + y*y;
  if (distance squared <= 1) number_in_circle++;
}
pi_estimate = 4*number_in_circle/((double) number_of_tosses);
```

This is called a "Monte Carlo" method, since it uses randomness (the dart tosses). Write an OpenMP program that uses a Monte Carlo method to estimate π. Read in the total number of tosses before forking any threads. Use a reduction clause to find the total number of darts hitting inside the circle. Print the result after joining all the threads. You may want to use long long ints for the number of hits in the circle and the number of tosses, since both may have to be very large to get a reasonable estimate of π.

Comparison (10 marks)

Compare the performances of all the estimate π programs from lab 3 and 4 on both efficiency and speedup.