Gebze Technical University - Computer Engineering Faculty

CSE-344 Homework-5 Report

Muhammed Bedir ULUCAY – 1901042697

- Task To Do:

Our task is implementing a matrix operation over the two-matrix using multi-thread program and all thread will calculate the matrix_size / (thread_number) number of row or column.

- Design:

Firstly, we are reading data from the two-matrix file and placing them into a matrix struct variable, then calculate the index number for thread calculation border and we determine these borders.

Then create all the threads and run them.

```
for(int i=0; i<m; ++i){

    int start = i * (pow(2, n) / m);
    int end = (i+1) * (pow(2, n) / m);

    thread_args *args = (thread_args*) calloc(sizeof(thread_args), 1);      You, 4 days ago • done multi th
    args->start = start;
    args->end = end;
    args->id = i;

    pthread_create(&threads[i], NULL, (void*) thread_start, args);
}
```

Calculation Of Matrix Multiplication:

```
for(int i=args->start; i<args->end; ++i){
    for(int j=0; j<8; ++j){

        row = get_nth_row(&matrix_A, i);
        col = get_nth_col(&matrix_B, j);

        matrix_C.matrix[i][j] = multiply_row_column(matrix_A.n, row, col);
  You, 23 hours ago • done except stdout …
        free(row);
        free(col);
    }
}
```

Firstly, all threads must finish their first task to calculate the second task of the calculated matrix.

To make a synchronization barrier after task 1. We are keeping two integer variables, one for the number of threads and the other one for the task1_done counter. Until the last thread arrive this point all thread call the pthread_cond_wait() method until the last thread calls the pthread_cond_broadcast() sending them a signal and all thread will keep running where they are left to run.

And we are waiting for all threads to finish task 1 using two methods

- pthread_cond_wait(&cond_task_1, &mutex_task_1);
- pthread_cond_broadcast(&cond_task_1);

And we are defining this area in mutex to avoid race conditions

```
pthread_mutex_lock(&mutex_task_1);

task_1_done++;
if(task_1_done < m){
    pthread_cond_wait(&cond_task_1, &mutex_task_1);
}
else{
    pthread_cond_broadcast(&cond_task_1);
}
pthread_mutex_unlock(&mutex_task_1);
  You, 3 days ago • syncronization barrier done …
```

After all the thread runs again. We are continuing to calculate the 2D Discrete Fourier Transform of the calculated matrix.

Using the following formulas.

$$F[k,l] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m,n] e^{-j2\pi\left(\frac{k}{M}m + \frac{l}{N}n\right)}$$

where $k = 0,1,\ldots,M-1$ and $l = 0,1,\ldots,N-1$

Euler's Formula

$$e^{i\phi} = \cos\phi + i\sin\phi$$

```
for(int row_index=args->start; row_index<args->end; ++row_index){
    for(int col_index=0; col_index<matrix_C.n; ++col_index){      You, 3 days ago • a little discreat
        int result = iter_over_matrix(&matrix_C,(double) row_index,(double) col_index);
        result_matrix.matrix[row_index][col_index] = result;
    }
}
```

```
double iter_over_matrix(Matrix* matrix, double row_index, double col_index){

    double result = 0, power, euler, real, imag;

    for(int i=0; i<matrix->n;++i){
        for(int j=0; j<matrix->n; ++j){

            power = 2 * PI * (((row_index / matrix->n) * i) + ((col_index / matrix->n) * j));

            real = cos(power);
            imag = -sin(power);
            euler =  CMPLX(real, imag);

            result += matrix->matrix[i][j] * euler;
        }
    }
    return result;      You, 3 days ago • a little discreate fourier transform …
}
```

Then the result matrix printing the output file.

- Speed Analysis:

In multi-thread programs can execute at the same time on a CPU. But according to the input thread number can change the speed size. Lets say we have so big input in this case use all the thread in the CPU's core can make sense but if we have a very small input in this case it is not make sense assigning each row with another

```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   39 bits physical, 48 bits virtual
CPU(s):                          8
On-line CPU(s) list:             0-7
Thread(s) per core:              2
Core(s) per socket:              4
Socket(s):                       1
```

MY-CPU

because thread creation can take more time according to the fake context switch. But if we use same input which is very big data. The more threads increase up to a certain point, the faster they get, but after a certain point it will decrease the speed according to the miss rate, page fault or thread creation is take more time.

```
mbulucay@mbulucay-GP72M-7REX:~/Desktop/sys
mbulucay@mbulucay-GP72M-7REX:~/Desktop/system/hw/5/19010426975 make run_n5n2
./hw5 -i data1 -j data2 -o output.csv -n 5 -m 2
Thread 0 will start from 0 to 16
Thread 1 will start from 16 to 32
22:14:06:750 Thread 0 has reached the rendezvous point in 0.001030 seconds.
22:14:06:750 Thread 1 has reached the rendezvous point in 0.001084 seconds.
22:14:06:750 Thread 1 is advancing to the second part
22:14:06:750 Thread 0 is advancing to the second part
22:14:06:800 22:14:06:800 Thread 0 has has finished the second part in 0.049210 seconds.
22:14:06:800 22:14:06:800 Thread 1 has has finished the second part in 0.049405 seconds.
The process has written the output file. The total time spent is 0.060987 seconds.
mbulucay@mbulucay-GP72M-7REX:~/Desktop/system/hw/5/19010426975 make run_n5n4
./hw5 -i data1 -j data2 -o output.csv -n 5 -m 4
Thread 0 will start from 0 to 8
Thread 1 will start from 8 to 16
Thread 2 will start from 16 to 24
Thread 3 will start from 24 to 32
22:14:11:020 Thread 1 has reached the rendezvous point in 0.000211 seconds.
22:14:11:020 Thread 3 has reached the rendezvous point in 0.000265 seconds.
22:14:11:020 Thread 0 has reached the rendezvous point in 0.000390 seconds.
22:14:11:020 Thread 2 has reached the rendezvous point in 0.000396 seconds.
22:14:11:020 Thread 1 is advancing to the second part
22:14:11:020 Thread 0 is advancing to the second part
22:14:11:020 Thread 2 is advancing to the second part
22:14:11:020 Thread 3 is advancing to the second part
22:14:11:031 22:14:11:031 Thread 1 has has finished the second part in 0.011280 seconds.
22:14:11:031 22:14:11:031 Thread 3 has has finished the second part in 0.011500 seconds.
22:14:11:036 22:14:11:036 Thread 0 has has finished the second part in 0.016135 seconds.
22:14:11:036 22:14:11:036 Thread 2 has has finished the second part in 0.016133 seconds.
The process has written the output file. The total time spent is 0.026645 seconds.
mbulucay@mbulucay-GP72M-7REX:~/Desktop/system/hw/5/19010426975 make run_n5n8
./hw5 -i data1 -j data2 -o output.csv -n 5 -m 8
Thread 0 will start from 0 to 4
Thread 1 will start from 4 to 8
Thread 2 will start from 8 to 12
Thread 3 will start from 12 to 16
Thread 4 will start from 16 to 20
22:14:17:160 Thread 0 has reached the rendezvous point in 0.000134 seconds.
Thread 5 will start from 20 to 24
22:14:17:160 Thread 2 has reached the rendezvous point in 0.000117 seconds.
Thread 6 will start from 24 to 28
22:14:17:160 Thread 1 has reached the rendezvous point in 0.000161 seconds.
22:14:17:160 Thread 3 has reached the rendezvous point in 0.000125 seconds.
Thread 7 will start from 28 to 32
22:14:17:160 Thread 4 has reached the rendezvous point in 0.000066 seconds.
22:14:17:160 Thread 5 has reached the rendezvous point in 0.000062 seconds.
22:14:17:160 Thread 6 has reached the rendezvous point in 0.000116 seconds.
22:14:17:160 Thread 7 has reached the rendezvous point in 0.000101 seconds.
22:14:17:160 Thread 7 is advancing to the second part
22:14:17:160 Thread 0 is advancing to the second part
22:14:17:160 Thread 4 is advancing to the second part
22:14:17:160 Thread 6 is advancing to the second part
22:14:17:160 Thread 2 is advancing to the second part
22:14:17:160 Thread 1 is advancing to the second part
22:14:17:160 Thread 5 is advancing to the second part
22:14:17:167 22:14:17:167 Thread 2 has has finished the second part in 0.006323 seconds.
22:14:17:167 Thread 3 is advancing to the second part
22:14:17:168 22:14:17:168 Thread 6 has has finished the second part in 0.008000 seconds.
22:14:17:169 22:14:17:169 Thread 0 has has finished the second part in 0.008375 seconds.
22:14:17:169 22:14:17:169 Thread 1 has has finished the second part in 0.008386 seconds.
22:14:17:169 22:14:17:169 Thread 7 has has finished the second part in 0.008441 seconds.
22:14:17:169 22:14:17:169 Thread 4 has has finished the second part in 0.008648 seconds.
22:14:17:170 22:14:17:170 Thread 5 has has finished the second part in 0.008147 seconds.
22:14:17:173 22:14:17:173 Thread 3 has has finished the second part in 0.006384 seconds.
The process has written the output file. The total time spent is 0.023641 seconds.
mbulucay@mbulucay-GP72M-7REX:~/Desktop/system/hw/5/19010426975
```