系統程式設計 System Programming 2021

Hw14 蒙特卡羅 Pi

電機四 406415089 宋 O 天

<1> 使用蒙特卡羅法計算 pi，應用程式可以有二個參數，第一個參數是總共打多少個點，第二個參數是使用多少 thread 做運算

完成，參考老師的 code 去改一些地方而已

```
void main(int argc,char*argv[]) {
    pthread_t id[16];
    int i;
    double pi = 0.0;
    double rand_d;

    total_loopcount=atol(argv[1]);
    num_thread=atoi(argv[2]);
    assert(num_thread < 16);
    para =(struct param*)malloc(sizeof(struct param)*num_thread);

    signal(SIGINT , sighandler);

    pthread_mutex_init(&mutex,NULL);
    for(i = 0 ;i < num_thread ;i++ ) {
        /*使用aligned_alloc分配記憶體，避免false sharing*/
        /*在這個例子中，「剛好」用malloc也會得到相同效果*/
        para[i].thread_index = i;
        para[i].now_loopcount= 0;
        para[i].rand_buffer = aligned_alloc(64, sizeof(struct drand48_data));
        struct param* para_ptr = para+i;

        srand48_r(rand(), para[i].rand_buffer);
        drand48_r(para[i].rand_buffer, &rand_d);

        printf("@buffer = %p\n",para[i].rand_buffer);
        printf("thread%d's seed = %f\n",i,rand_d);

        pthread_create(&id[i],NULL,(void *)thread,(void *)para_ptr);
    }

    for(i=0;i<num_thread;i++)
        pthread_join(id[i],NULL);

    pi = (double)4*(global_hit/total_loopcount);
    printf("pi = %.8lf\n",pi);
}
```

<2> 當按下 ctr-c 的時候，顯示截至目前 計算出來的 pi 是多少

先宣告了全域的 struct，並會利用裡面的 now_loopcount 來記錄現在是第幾次 loop 了

```
struct param{
    int thread_index;
    int now_loopcount;
    struct drand48_data* rand_buffer;
};

struct param* para;
```

for 迴圈裡面會更新 now_loopcount 是多少

```
void thread(struct param* para) {
    double point_x,point_y;
    double rand_d;
    long i;
    long local_loopcount = total_loopcount/num_thread;

    pthread_mutex_lock(&mutex);
    long* local_hit = &hit[idx++];
    pthread_mutex_unlock(&mutex);

    for(i = 0 ;i < local_loopcount ;i++){
        para->now_loopcount = i;
        drand48_r(para->rand_buffer, &rand_d);
        point_x = rand_d;
        drand48_r(para->rand_buffer, &rand_d);
        point_y = rand_d;

        if( (point_x*point_x + point_y*point_y) < 1.0)
            *local_hit+=1;
    }
    //printf("hit = %ld\n", *local_hit);

    pthread_mutex_lock(&mutex);
    global_hit += *local_hit;
    pthread_mutex_unlock(&mutex);
}
```

<3> 當在一秒內連續按下二次 ctr-c 時顯示截至目前的 pi 是多少，並結束程式

在 signal_handler 裡面利用 clock_gettime 來計算兩次按下 Ctrl+C 的時間，

如果大於一秒的話就 exit()，計算次數利用取餘數的方式

```
void sighandler(int signum){
    if (enter_handler_count%2 == 0)
        clock_gettime(CLOCK_MONOTONIC, &tt1);
    else
        clock_gettime(CLOCK_MONOTONIC, &tt2);

    for (int i=0; i<idx; i++){
        printf("now, thread %d hit = %ld\n",i ,hit[i]);
        printf("now, loopcount : %d\n",para[i].now_loopcount);
    }
    calculate_average();
    enter_handler_count++;

    long time = ts_to_long(tt2)-ts_to_long(tt1);
    if (time < 1000000000 && time > 0)
        exit(EXIT_SUCCESS);

}
void calculate_average(void){
    double total_sum = 0.0;
    for(int i = 0 ;i < num_thread ;i++ ){
        pi[i] = 4*((double)hit[i]/para[i].now_loopcount);
        printf("pi[%d] = %lf\n",i,pi[i]);
        total_sum = total_sum+pi[i];
    }
    printf("Now pi = %2.7lf\n",total_sum/num_thread);
}
```

<4> 請說明你的程式比老師所給的範例程式快或者是慢，具體說明原因

編譯時不加入-O3 參數

我

```
real    0m14.869s
user    0m29.489s
sys     0m0.036s
nash@SleepyCat:~/Desktop/sp_hw/14$ time ./pi 1000000000 8
```

老師

```
real    0m15.247s
user    0m30.104s
sys     0m0.088s
nash@SleepyCat:~/Desktop/sp_hw/14$ time ./pi2 1000000000 8
```

編譯時加入-O3 參數

我

```
real    0m13.100s
user    0m25.679s
sys     0m0.132s
nash@SleepyCat:~/Desktop/sp_hw/14$
```

老師

```
real    0m14.210s
user    0m28.174s
sys     0m0.024s
nash@SleepyCat:~/Desktop/sp_hw/14$
```

這樣看實際上差不多，兩個程式都是用 mutex，都沒有直接宣告 atomic 型態的變數，所以都偏慢

我覺得我的 code 主要在於更新 loop_count 的時候，每一次都會將變數更新，也是沒辦法提升速度的原因。

```
    for(i = 0 ;i < local_loopcount ;i++){
        para->now_loopcount = i;
        drand48_r(para->rand_buffer, &rand_d);
        point_x = rand_d;
        drand48_r(para->rand_buffer, &rand_d);
        point_y = rand_d;

        if( (point_x*point_x + point_y*point_y) < 1.0)
            *local_hit+=1;
    }
```

如果我利用 if 判斷式並將-O3 開啟，讓 CPU 去猜我下一次 branch 的結果，這樣稍快一點

但依然沒有課堂上那位同學來的好，他整整快了兩倍，可能 atomic operation 真的有差。

(我交的 code 是沒有 if 的版本)

```
real    0m12.639s
user    0m25.005s
sys     0m0.028s
```

```
 for(i = 0 ;i < local_loopcount ;i++){
     if (flag == 1)
         para->now_loopcount = i;
     drand48_r(para->rand_buffer, &rand_d);
     point_x = rand_d;
     drand48_r(para->rand_buffer, &rand_d);
     point_y = rand_d;

     if( (point_x*point_x + point_y*point_y) < 1.0)
         *local_hit+=1;
 }
```

<參考資料>

https://github.com/ray1422/System-Programming-Homework/blob/homework/hw14/calc_pi.c

老師的 code

<致謝>

摯友 博禕

這學期辛勞的助教、羅習五老師的教導