多處理機平行程式設計 作業四 report

F74109032 陳均哲

Problem 1

1. 程式說明

Barrier實作方法

```
pthread_mutex_lock(&mutex);
if(counter[count % 2] == thread_num - 1){
    counter[(count + 1) % 2] = 0;
    swap(BMPSaveData, BMPData);
}
counter[count % 2]++;
pthread_mutex_unlock(&mutex);
while(counter[count % 2] < thread_num);
```

這邊是使用Busy-waiting and a Mutex的方式來實作Barrier。counter是個有兩個數值的陣列，然後會互相輪替使用。當其中一個thread執行到這邊的時候，counter會加一，然後會進入while迴圈等待，直到當counter的數值到達thread總數，代表所有thread都執行到相同的位置，所有thread才會跳出while迴圈繼續執行。而當最後一個thread執行到這邊的時候，會把像素資料與暫存指標做交換，並把下一輪要使用的counter歸零。

Pthread使用方法

```
void *child(void *data){
    int my_id = *(int*)data;
    ...
    pthread_exit(NULL);
}

int main(){
    ...
    pthread_t *t;
    t = (pthread_t*)malloc(thread_num * sizeof(pthread_t));

    int *data = (int*)malloc(thread_num * sizeof(int));
    for (int i = 0; i < thread_num; i++){
        data[i] = i;
        pthread_create(&t[i], NULL, child, &data[i]);
    }

    for (int i = 0; i < thread_num; i++){
        pthread_join(t[i], NULL);
    }
    ...
}
```

在pthread_create的地方會把前面設定的pthread分別執行child函式。而其中一個argument是要把每個thread各自代表的id傳進去，在main函式中用data一個陣列儲存然後傳進去。在child最後pthread_exit後會在main函式的後面pthread_join回去。

2. 時間比較

平滑次數 = 1000

thread_num =	1	2	4	8	16	24	32	40
time(sec)	44.9727	22.7152	11.9127	6.6721	6.5595	21.1213	26.3899	34.6756

3. 結果分析

從結果綜合觀察執行過程來看，當thread的數量為8到16之間會到達CPU使用率的極限，可能是因為核心數量有限的關係造成。因此當thread數量是1到8之間，能夠有效率的將程式平行化，使得執行時間與thread數量之間呈現成反比的關係；而當thread數量到達16以上，則是會因為核心數量有限，沒辦法所有thread平行化執行，造成context switch次數過多而thread之間搶資源的情形，又加上因為使用barrier的關係，所有thread都要等到所有thread都執行完一次平行化運算才能繼續執行下一輪的運算，造成執行時間暴增。