System programming hw4 report

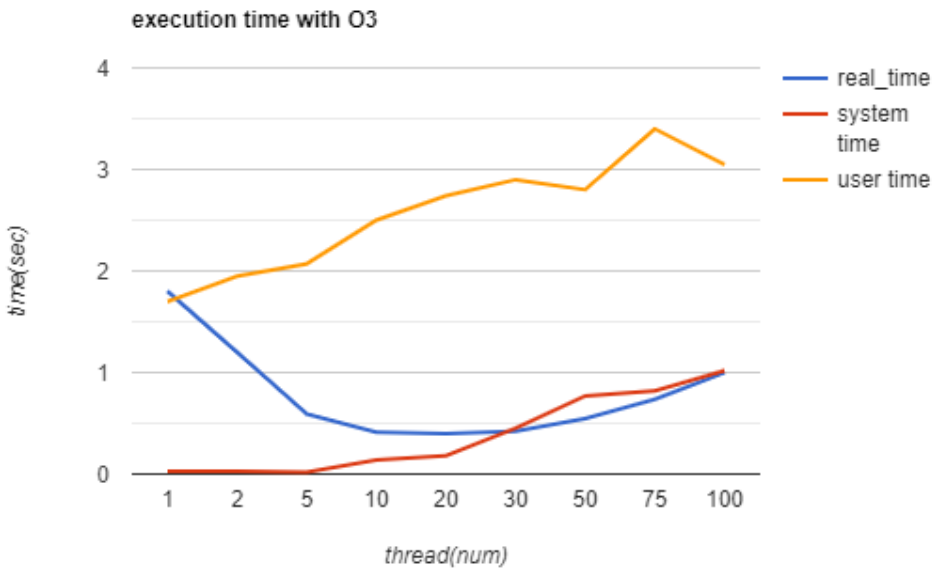
B07902094 廖榮運

(4)thread =20 時明顯比 thread=2 時更快，thread 越多就能更有效的利用 user 資源，但到一定程度後效能反而會變差，這次作業 thread 大概在 20~30 效能最好。

(5)**Real time**:thread 從 1 到 30 前都能加快 process 速度，但之後就逐漸上升，呈現凹狀的 line graph。

User time:逐漸升高，時間最高，因為 pthread 是 user thread , create 越多 user time 也越高

System time:逐漸升高，因為當 thread 越多，system 做一些無意義的 contextswitch 的可能性越高。



Code implement:

```
for(int i=0;i<q_num;i++){
    argu[i].how=row_or_col;
    argu[i].before_arr = init_before_arr;
    argu[i].after_arr = init_after_arr;
    argu[i].start = i * (bigger/q_num);
    if(i == q_num-1)
        argu[i].end = bigger-1;
    else
        argu[i].end = (i+1) * (bigger/q_num) -1;
}
for(int e=0;e<epoch;e++){
    for(int i=0;i<q_num;i++){
    pthread_create(&tid[i],NULL,run_epoch,&argu[i]);
    }
for(int i = 0;i<q_num;i++){
    pthread_join(tid[i], NULL);
}
copy_arr(init_before_arr,init_after_arr);
}
```

我的作法是先將每個 thread 要處理的部分在 main_thread 裡面先切分好，存在一個 structure 裡，內容包括處理的陣列、如何處理、以及處理範圍，再分別丟去給 thread 做，因為是所有 thread 做完再更新原本的 array，所以不需要用到 lock。如果 row 大於 col 就橫著去切分，不然就直著切，來處理極端矩陣資料。每個 pthread 照著原本的 array 去根據 rule 決定另一個 array 的值。全部 thread 都做完後才將 array copy 過去，再進行下一個 epoch。