Prog Hw4 Report

B09902032 資工二 彭昱齊

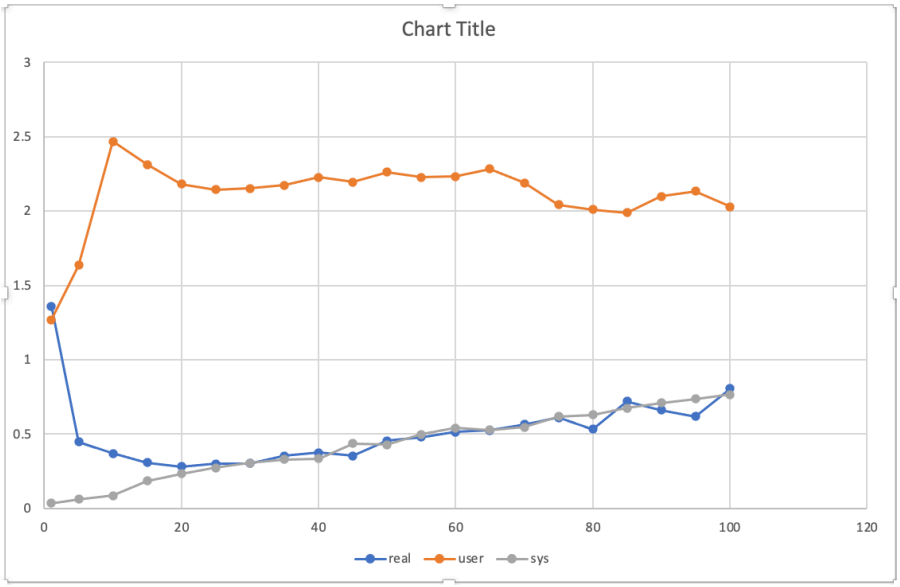
4.開較多thread會將工作切成較多份，每份工作量少，因此real time減少，但user time會將每個thread用到的user-mode時間加起來，因此越多thread導致user time越高，開thread時要呼叫system call因此開越多thread導致sys time增加

```
b09902032@meow2 [~/prog_hw4] time ./main -t 2 ./sample_input/largeCase_in.txt output.txt

real    0m1.505s
user    0m2.759s
sys     0m0.041s
b09902032@meow2 [~/prog_hw4] time ./main -t 20 ./sample_input/largeCase_in.txt output.txt

real    0m0.372s
user    0m3.393s
sys     0m0.228s
```

5.Real time: 剛開始因為thread增加而時間減少是因爲CPU有多個核心平行執行程式，但thread越開越多而核心不夠，且開thread也要花時間，所以後面呈現緩緩上升的趨勢
User time: 開越多thread就要把每個thread在各個核心的user-mode時間加起來，所以剛開始會上升，而thread開到一定數量後，因為CPU核心不夠，所以實際在用的核心數一樣，將所有核心user-mode花的時間加起來後會看到user time會呈現平平的趨勢
Sys time: 因為開thread要用到system call，所以開越多要花越多sys time



6.Real time: 因為都是2個thread或2個process都是平行的兩個東西在跑，所以real time差不多
Sys time: 可能mmap比較複雜花較多時間，fork時有些virtual memory的東西也要複製到child process，且OS在switch context要save 以及reload process狀態

```
b09902032@linux13 [~/prog_hw4] time ./main -t 2 ./sample_input/largeCase_in.txt output.txt

real    0m0.742s
user    0m1.326s
sys     0m0.031s
b09902032@linux13 [~/prog_hw4] time ./main -p 2 ./sample_input/largeCase_in.txt output.txt

real    0m0.755s
user    0m1.392s
sys     0m0.055s
```

程式重點部分

切開每個thread或process需要處理的區域：
判斷row跟col哪個較大，從較大的去除以process或thread的數量，例如：11cols 5 rows 5 threads，就從column切開，所以每個thread分到11/5=2個cols，而餘數的部分11%2=1就分給最後一個thread。

```
int cut;
if (row > col){
    cut = row / t;
    for (int i = 0; i < t-1; i++){
        jobs[i].row_s = cut * i;
        jobs[i].row_e = cut * (i+1) - 1;
        jobs[i].col_s = 0;
        jobs[i].col_e = col - 1;
    }
    jobs[t-1].row_s = cut * (t-1);
    jobs[t-1].row_e = row-1;
    jobs[t-1].col_s = 0;
    jobs[t-1].col_e = col - 1;
}else{
    cut = col / t;
    for (int i = 0; i < t-1; i++){
        jobs[i].col_s = cut * i;
        jobs[i].col_e = cut * (i+1) - 1;
        jobs[i].row_s = 0;
        jobs[i].row_e = row - 1;
    }
    jobs[t-1].col_s = cut * (t-1);
    jobs[t-1].col_e = col-1;
    jobs[t-1].row_s = 0;
    jobs[t-1].row_e = row - 1;
}
for (int i = 0; i < t; i++){
    jobs[i].r_size = row;
    jobs[i].c_size = col;
}
```

開thread:
每個epoch都要開thread，並join他們，這樣才能保證每個thread被分到的區塊都處理完了才進到下一個epoch

```
pthread_t tp[t];
//printf("t=%d\n", t);
for (int k = 0; k < epoch; k++){
    for (int i = 0; i < t; i++){
        //printf("before\n");
        pthread_create(&tp[i], NULL, survive, &jobs[i]);
        //printf("after\n");
    }

    for (int i = 0; i < t; i++){
        pthread_join(tp[i], NULL);
    }
    tmp = dots;
    dots = next_dots;
    next_dots = tmp;
}
```

開process：
每個epoch都開兩個processes並wait他們，這樣保證processes都處理完他們被分配到的區域才進入下一個epoch

```
if (fork() == 0){
    int lives;
    for (int x = row_l[0]; x <= row_r[0]; x++){
        for (int y = col_l[0]; y <= col_r[0]; y++){
            lives = 0;
            if (x-1 >= 0 && ddots[(x-1)*col+y] == 1)
                lives += 1;
            if (x+1 < row && ddots[(x+1)*col+y] == 1)
                lives += 1;
            if (y-1 >= 0 && ddots[x*col+(y-1)] == 1)
                lives += 1;
            if (y+1 < col && ddots[x*col+(y+1)] == 1)
                lives += 1;
            if (x-1 >= 0 && y-1 >= 0 && ddots[(x-1)*col+(y-1)] == 1)
                lives += 1;
            if (x-1 >= 0 && y+1 < col && ddots[(x-1)*col+(y+1)] == 1)
                lives += 1;
            if (x+1 < row && y-1 >= 0 && ddots[(x+1)*col+(y-1)] == 1)
                lives += 1;
            if (x+1 < row && y+1 < col && ddots[(x+1)*col+(y+1)] == 1)
                lives += 1;

            if (ddots[x*col+y] == 0)
                next_ddots[x*col+y] = transform[lives][0];
            else
                next_ddots[x*col+y] = transform[lives][1];
        }
    }
    exit(0);
}
```

2 processes share memory：
用mmap開一個virtual memory存一維陣列（從二維陣列轉換而來），之後fork出來的child就都可以存取到這個一維陣列

```
ddots = (bool *) mmap(NULL, sizeof(bool)*row*col, PROT_READ | PROT_WRITE, MAP_SHARED |
    MAP_ANONYMOUS, -1, 0);
next_ddots = (bool *) mmap(NULL, sizeof(bool)*row*col, PROT_READ | PROT_WRITE, MAP_SHARED |
    MAP_ANONYMOUS, -1, 0);
```