# 功能一:「pi-p7」

```
s407210013@workstation1:-/osdi/sharedFolder/hw05.pthread$ ./pi -p 7
lowerbdd is 3.14159266856
pi is 3.14159266356
s407210013@workstation1:-/osdi/sharedFolder/hw05.pthread$ ./pi -p 7
lowerbdd is 3.14159266856
pi is 3.14159266356
s407210013@workstation1:-/osdi/sharedFolder/hw05.pthread$
```

## 功能二:「pi-p7-P5」

```
s407210013@workstation1:-/osdi/sharedFolder/hw05.pthread$ ./pi -p 7 -P 5
lowerbdd is 3.14159265859
upperbdd is 3.14159266859
pi is 3.14159266359
s407210013@workstation1:-/osdi/sharedFolder/hw05.pthread$ ./pi -p 7 -P 5
```

功能三:「./pi」

```
### S407210013@workstation1:-/osdi/sharedFolder/hw05.pthread

File Edit View Search Terminal Help

$407210013@workstation1:-/osdi/sharedFolder/hw05.pthread$ ./pi

C

Lowerbdd is 3.14159265493

upperbdd is 3.14159265492

pi is 3.14159265448

$407210013@workstation1:-/osdi/sharedFolder/hw05.pthread$
```

#### 主題甲:

時間計算 (皆採用 48 核心去做計算):

Real time: 在 48 核心下需花費的總實際時間。

User time: user mode 處理時總共 CPU 花費的時間。

System time: printf 之類在 kernal mode 所花費的時間。

逼折到小數點第5位數

```
s407210013@workstation1:~/osdi/sharedFolder/hw05.pthread$ time ./pi -p 5

lowerbdd is 3.14159305723
upperbdd is 3.14159403284

pi is 3.14159354504

real    0m0.684s
user    0m2.227s
sys    0m0.991s
```

逼近到小數點第6位數

```
s407210013@workstation1:~/osdi/sharedFolder/hw05.pthread$ time ./pi -p 6
lowerbdd is 3.14159270259
upperbdd is 3.14159280259
pi is 3.14159275259
real  0m1.456s
user  0m18.766s
sys  0m1.837s
```

### 逼近到小數點第7位數

```
s407210013@workstation1:~/osdi/sharedFolder/hw05.pthread$ time ./pi -p 7
lowerbdd is 3.14159265856
upperbdd is 3.14159266856
pi is 3.14159266356
real  0m6.971s
user  3m15.146s
sys  0m2.897s
```

#### 逼近到小數點第8位數

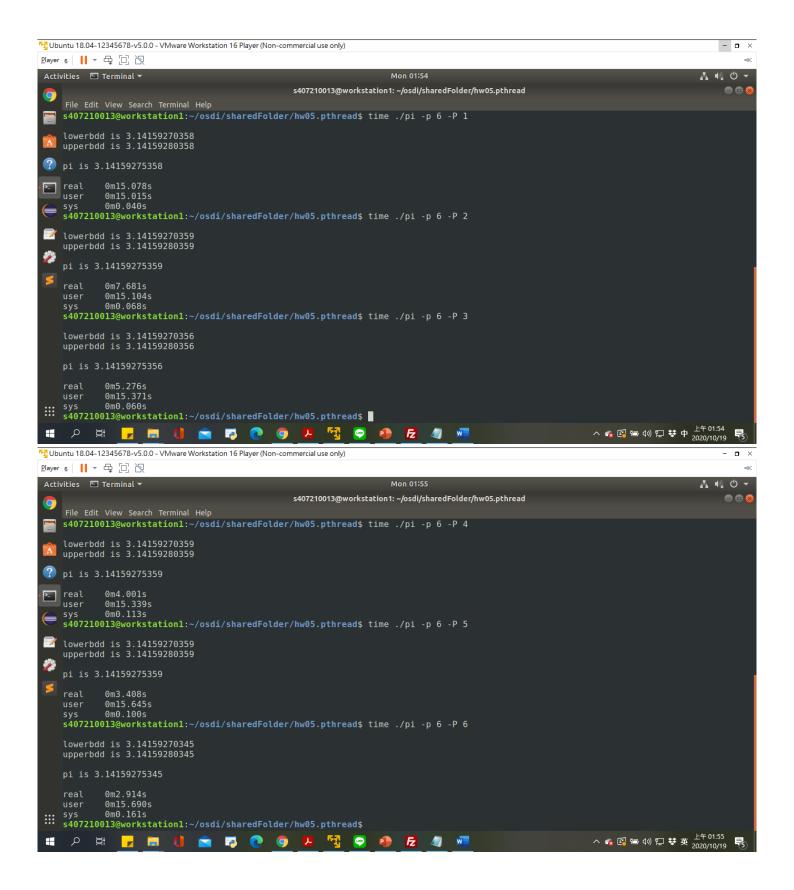
```
s407210013@workstation1:~/osdi/sharedFolder/hw05.pthread$ time ./pi -p 8
lowerbdd is 3.14159265407
upperbdd is 3.14159265505
pi is 3.14159265456
real  0m54.397s
user  33m55.730s
sys  0m3.619s
```

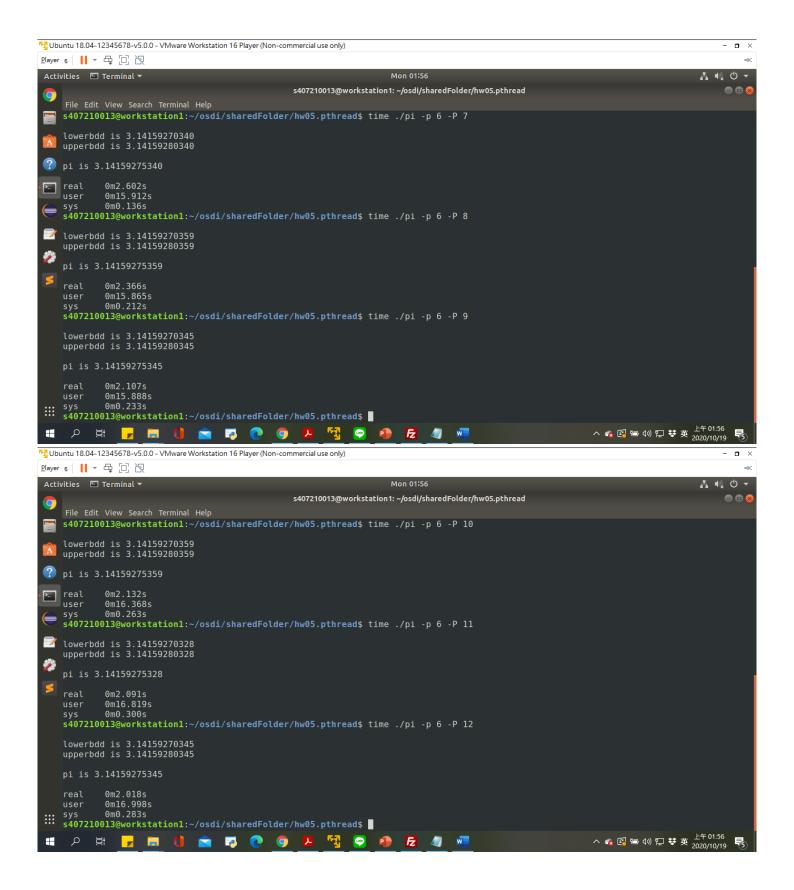
### 逼近到小數點第9位數

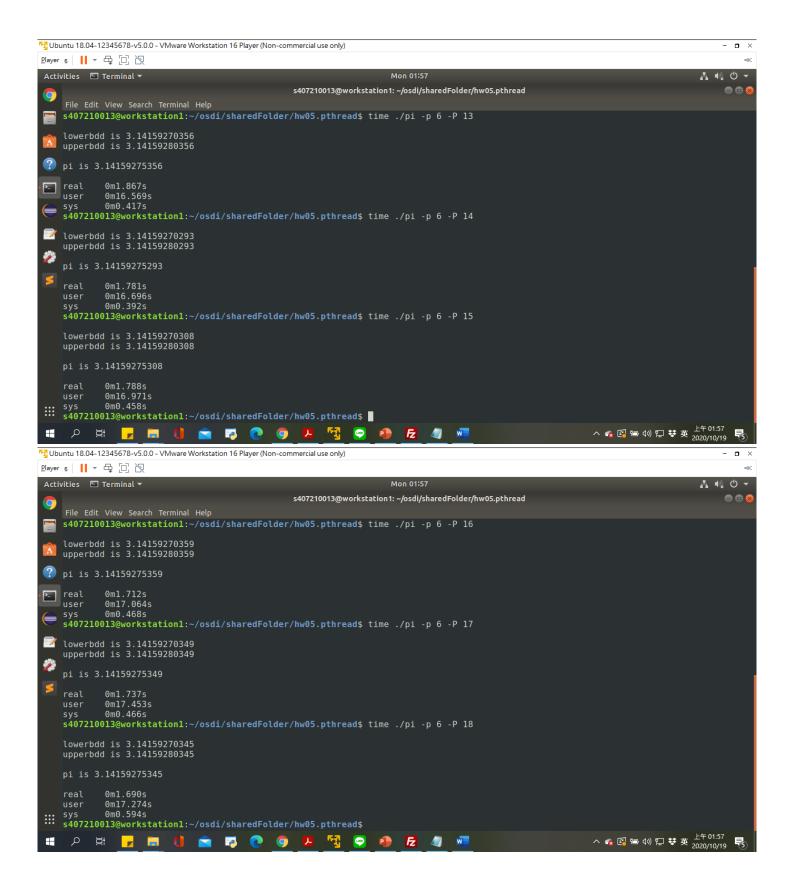
```
s407210013@workstation1:~/osdi/sharedFolder/hw05.pthread$ time ./pi -p 9
lowerbdd is 3.14159265364
upperbdd is 3.14159265374
pi is 3.14159265369
real 8m9.803s
user 342m59.332s
sys 0m5.688s
```

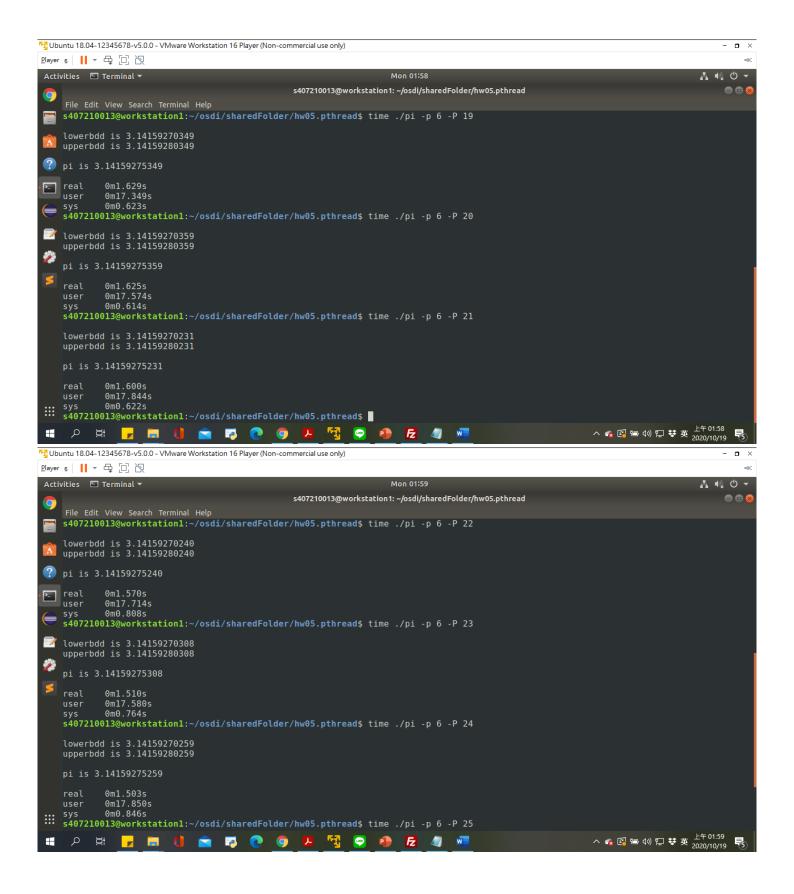
#### 主題乙:

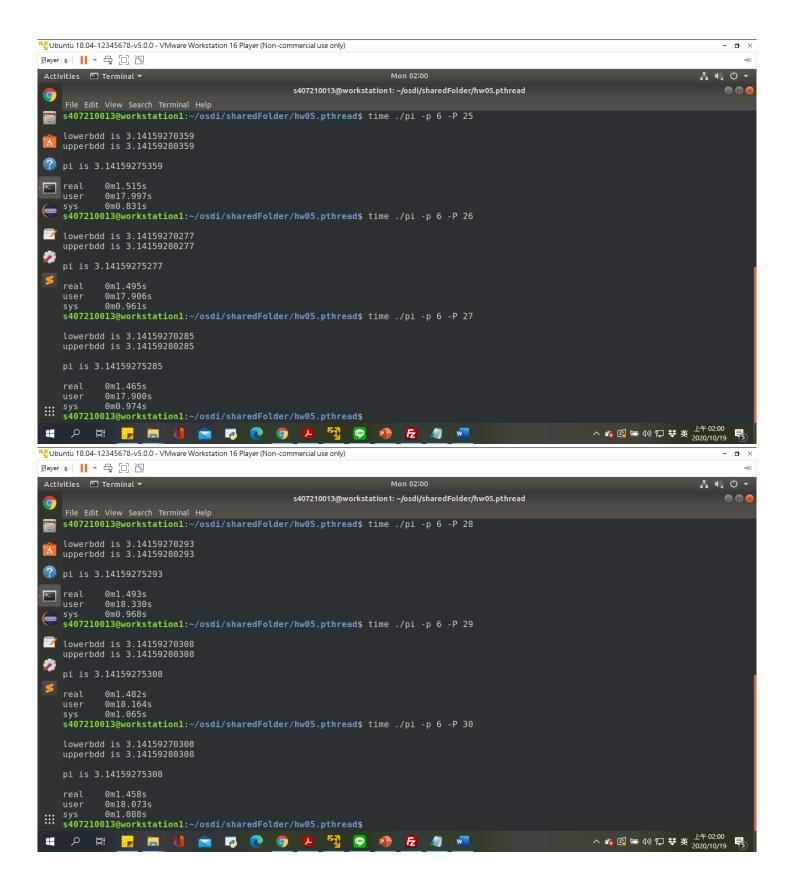
統一選擇逼近到小數點第6位數,探討在同樣的精準度下,觀察核心數與時間的關係。

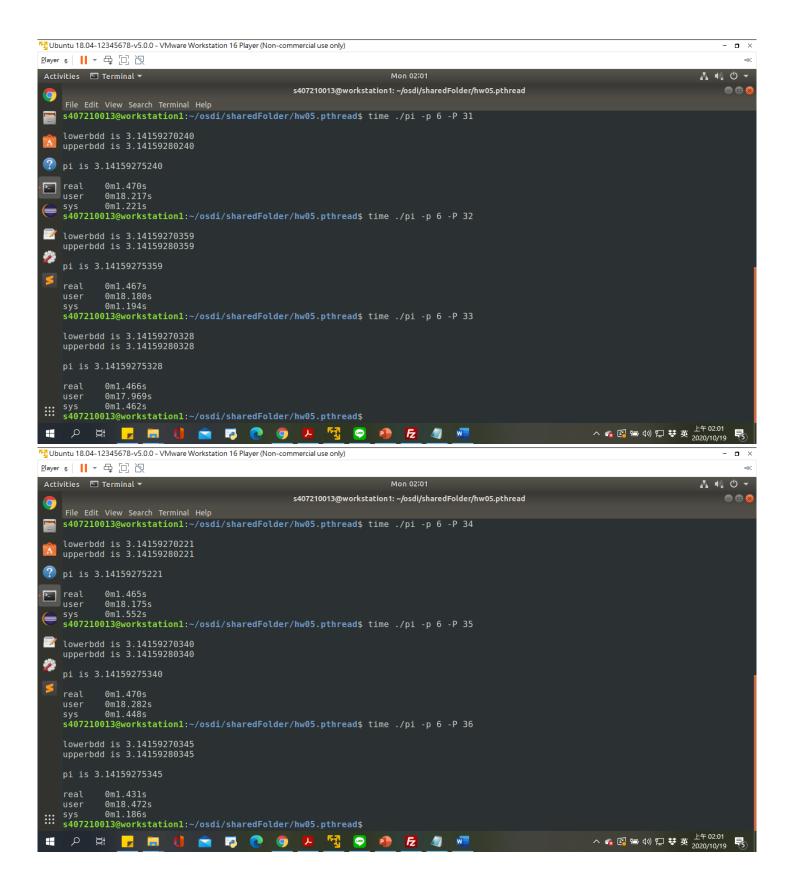


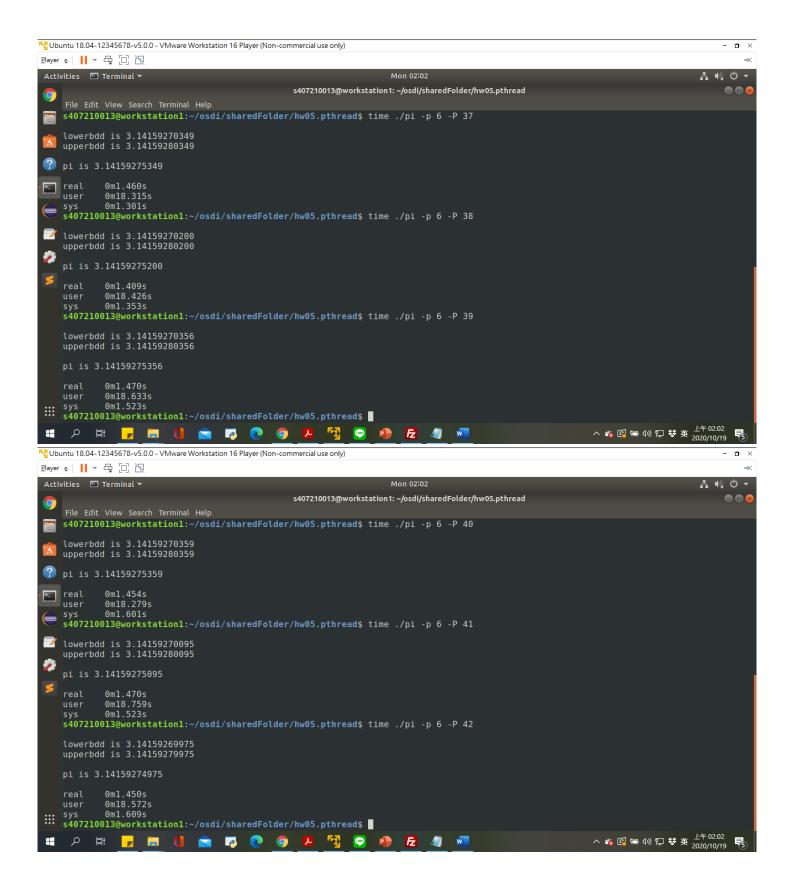


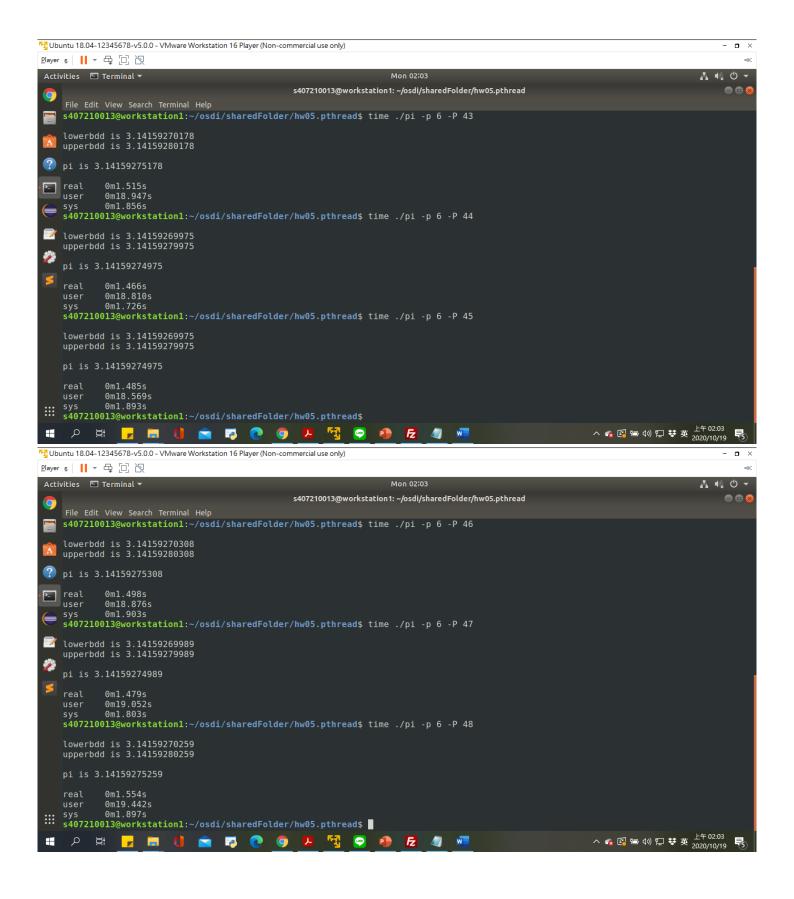






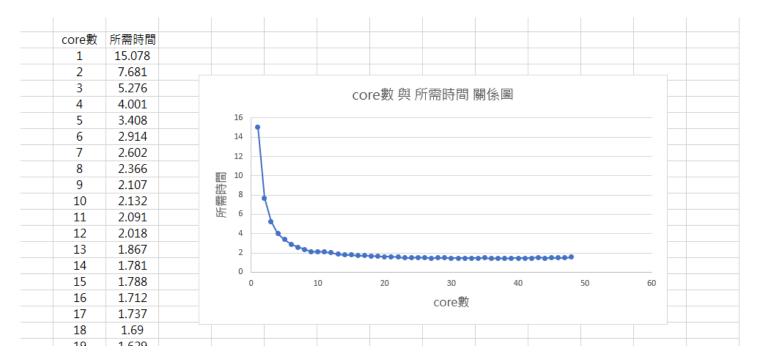






以上為各核心數下統計出來的所需時間。

以下為Core數與所需時間之關係圖,發現可得到非比尋常的線性加速,執行上減少許多 花費的時間。



#### 主題丙:

對於上界與下界的取法,我們只對下界進行累加,由於上界值與下界值只少了一塊最左 邊以圓半徑為長,等分切格的值為寬,所構成之矩形。因此讓下界加上此矩形,便為上 界的值,讓時間減少不少。

```
27 //注意,我使用了「volatile」
28 volatile double score[100];
29
30 void thread(void *givenName) {
    int id = (intptr_t)givenName;
    double pi = 0.0;
32    double width = (double)1 / loopCount;
34    for(size_t i = loopCount / numCPU * id ; i < loopCount / numCPU * (id + 1) ; i++){
        double x = (double)i / loopCount;
        pi = pi + width * sqrt(1 - x * x);
37    }
38    score[id] = pi;
39 }
40</pre>
```

```
double lowerbdd = 0;
double upperbdd = 1;
double tol = pow(10 , -(x + 1));
int y = 3;
while((upperbdd - lowerbdd) >= (tol / 4) && keepRunning){
    lowerbdd = 0;
    pthread_t* tid = (pthread_t*)malloc(sizeof(pthread_t) * numCPU);
    for (long i=0 ; i < numCPU ; i++)
        pthread_create(&tid[i], NULL, (void *) thread, (void*)i);

for (int i=0 ; i < numCPU ; i++)
        pthread_join(tid[i], NULL);

for (int i = 0; i < numCPU ; i++)
        lowerbdd += score[i];

upperbdd = lowerbdd + (double)1 / loopCount;
if(loopCount / 10 < pow(10 , y)){
        loopCount = loopCount + pow(10 , y - 1);
}
else{
        loopCount = loopCount + pow(10 , y);
        y = y + 1;
}
</pre>
```