Assignment 2

LU Decomposition

Using Pthread and OpenMP

By

Ashutosh Mohanta

2019PH10620

Working:

Serial:

To compile and run serial.c:

```
gcc serial.c -lm
```

```
time ./a.out n thread_count
```

Where n*n is the size of matrix. For serial code the thread_count won’t matter as it will only use 1 core. The program takes it as the argument just for the sake of uniformity.

example:

```
gcc serial.c -lm
```

```
time ./a.out 7000 1
```

Pthread:

To compile and run pthread.c:

```
gcc pthread.c -lpthread -lm
```

```
time ./a.out n thread_count
```

Where n*n is the size of matrix. thread_count can be atmost 32.

OpenMP:

To compile and run pthread.c:

```
gcc openmp.c -fopenmp -lm
```

```
export OMP_NUM_THREADS=thread_count
```

```
time ./a.out n thread_count
```

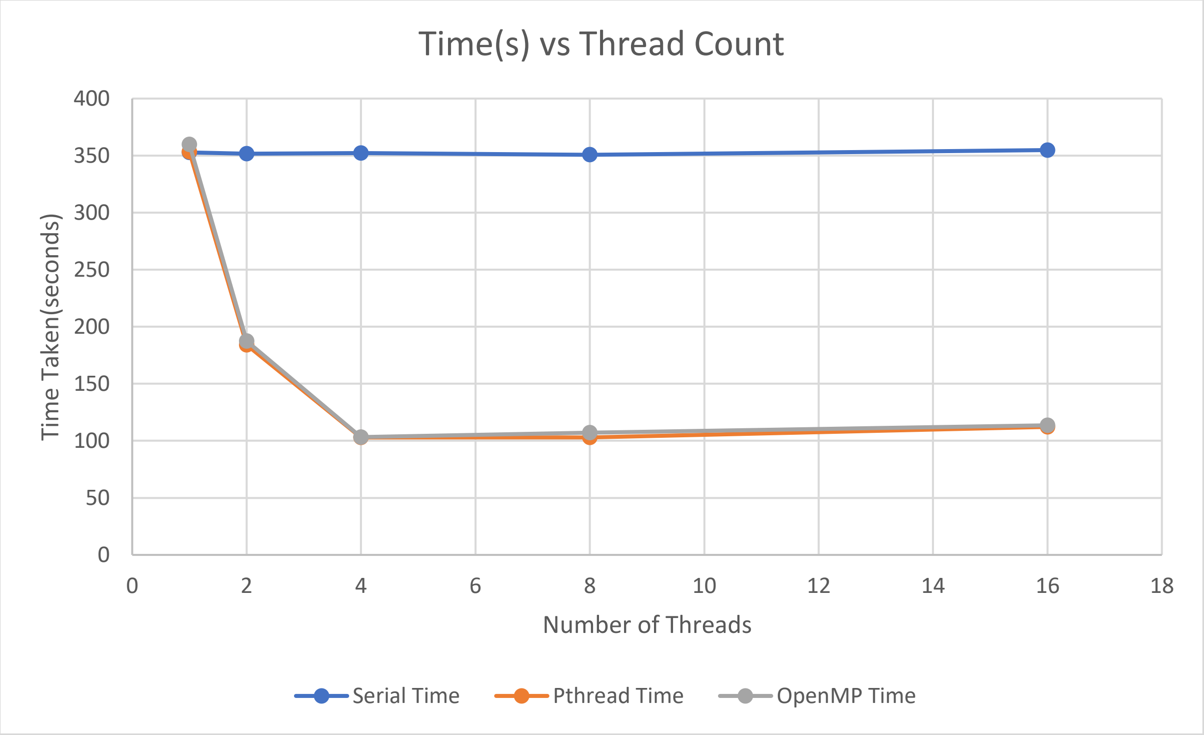
example:

```
export OMP_NUM_THREADS=8
```

```
time ./a.out 7000 8
```

Where n*n is the size of matrix. thread_count can be anything.

Graphs:



As we can see the minimum time takes is around 4 threads as I ran the code on 4 core laptop. After the 4 cores the speed tend to stay constant and/or decrease a bit.

SpeedUp (2 Cores) = 1.94

SpeedUp (4 Cores) = 3.40

SpeedUp (8 Cores) = 3.33

Algorithm Details:

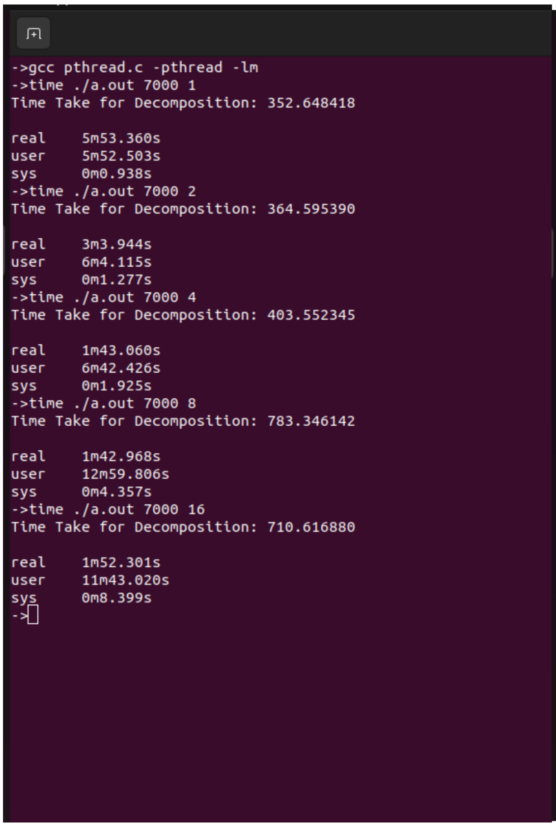
Pthread:

I used row division method to divide chunks of rows among threads. For Synchronization I used pthread_join to wait for synchronization. I also used mutex wherever necessary to ensure there is no race condition.

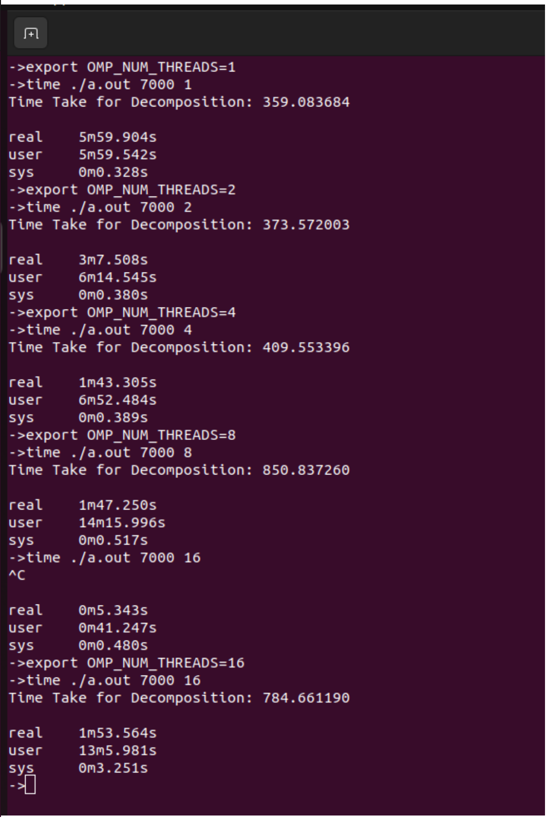
OpenMP:

Using #pragma to guide the compiler to divide the for loops among the threads. Four of the for loops are parallelized in this manner.

Timing Details:



```
->gcc pthread.c -pthread -lm
->time ./a.out 7000 1
Time Take for Decomposition: 352.648418

real    5m53.360s
user    5m52.503s
sys     0m0.938s
->time ./a.out 7000 2
Time Take for Decomposition: 364.595390

real    3m3.944s
user    6m4.115s
sys     0m1.277s
->time ./a.out 7000 4
Time Take for Decomposition: 403.552345

real    1m43.060s
user    6m42.426s
sys     0m1.925s
->time ./a.out 7000 8
Time Take for Decomposition: 783.346142

real    1m42.968s
user    12m59.806s
sys     0m4.357s
->time ./a.out 7000 16
Time Take for Decomposition: 710.616880

real    1m52.301s
user    11m43.020s
sys     0m8.399s
->
```



```
->export OMP_NUM_THREADS=1
->time ./a.out 7000 1
Time Take for Decomposition: 359.083684

real    5m59.904s
user    5m59.542s
sys     0m0.328s
->export OMP_NUM_THREADS=2
->time ./a.out 7000 2
Time Take for Decomposition: 373.572003

real    3m7.508s
user    6m14.545s
sys     0m0.380s
->export OMP_NUM_THREADS=4
->time ./a.out 7000 4
Time Take for Decomposition: 409.553396

real    1m43.305s
user    6m52.484s
sys     0m0.389s
->export OMP_NUM_THREADS=8
->time ./a.out 7000 8
Time Take for Decomposition: 850.837260

real    1m47.250s
user    14m15.996s
sys     0m0.517s
->time ./a.out 7000 16
^C

real    0m5.343s
user    0m41.247s
sys     0m0.480s
->export OMP_NUM_THREADS=16
->time ./a.out 7000 16
Time Take for Decomposition: 784.661190

real    1m53.564s
user    13m5.981s
sys     0m3.251s
->
```

I considered “real time” instead of “Time Take for Decomposition”, as the latter sums the times of all the cores and doesn’t reflect the real time that the user has to wait.