# Merge Sort: Multi-thread

11610310 Lu Ning

## Compile

```
gcc ./OS_Bonus1_11610310.c -o <name> -pthread -lm
```

## Run

```
./<name of compiled file>  <power> <thread number>
```

- power : The array length will be $10^{power}$

### An example:

```
luning@luning-laptop ~/workspace/OS/lab4_bonus ./mergesort 2 4
array length : 100 thread number: 4
befor multithread sorting
split into : [0 24] [25 49] [50 74] [75 99]
Creating thread 0
Creating thread 1
Creating thread 2
Creating thread 3

 Start final merging
4 sub array(s)
merge [0, 24] and [25, 49]
merge [50, 74] and [75, 99]
2 sub array(s)
merge [0, 49] and [50, 99]
The merge time for 4 threads : 0.035000 ms

the multi-thread sorting cost: 0.601000 ms
2 12 25 31 33 50 50 51 52 58 59 60 62 75 78 106 111 118 143 178 203 208 211 228
229 239 256 267 285 288 289 322 355 369 377 399 399 406 406 416 417 419 465 467
474 482 485 524 526 528 530 543 544 581 587 614 628 628 629 639 644 649 662 664
670 685 699 706 708 710 716 721 724 725 734 737 750 752 758 774 797 805 812 816
822 831 836 864 884 895 897 899 908 911 927 936 943 954 981 992
```

## Experiment

### Test platform

- System : Ubuntu 18.04 64 bit
- Core: Intel Core i5-6300HQ CPU @ 2.30GHz × 4
- Hyper-Threading Technology : No

### Random Sequence

| thread num \ array length | $10^3$ | $10^4$ | $10^5$ | $10^6$ | $10^7$ | $10^8$ | $10^9$ |
|---|---|---|---|---|---|---|---|
| 1 | 0.772 ms | 5.52 ms | 25 ms | 280 ms | 3 s | 32 s | 495 s |
| 2 | 0.58 ms | 3.7 ms | 13.5 ms | 147 ms | 1.6 s | 18 s | 221 s |
| 4 | 0.72ms | 4.03ms | 11.4 ms | 107 ms | 1 s | 12 s | 137 s |
| 8 | 1.3ms | 3.96ms | 11.8 ms | 109 ms | 1.135 s | 13 s | 156 s |

## Observation

As array length times 10, the time cost is also multiplied by 10.

We can see that the time cost for 2-thread is half of 1-thread.

Time cost of 4-thread is about 1/3 of one thread.

8-thread time cost is a little bit larger than 4-thread.

## Analysis

The time complexity of merge sort is O(n log(n))

We can get

$$\frac{10n \cdot log(10n)}{n \cdot log(n)} = \frac{10n \cdot (log(n) + 1)}{n \cdot log(n)} = 10 + \frac{10}{log(n)}$$

When $n$ is becoming larger, the time will be multiplied by 10 no matter what the thread number is.

The N-thread (N < 4) program run the mergesort simultaneously and separately on 4 cores, so the time cost becomes 1/N. But for the final part, the merging process of the N sub-arrays, is not multi-threaded. So the time cost of 4-thread program is not precisely 1/4 of that of the 1-thread one.

And because my computer only has 4 cores and doesn't provide hyper-thread technology. So the 8-thread program is similar to 4-thread program. And more threads means more creating time and more final merging time.