Project #0 - Simple OpenMP Experiment

Junhyeok Jeong
jeongju@oregonstate.edu
CS 475 - 001- S2020

1. Tell what machine you ran this on
   - OS: Linux Ubuntu 18.04.4 LTS (64-bit)
   - RAM: 15.6 GiB memory
   - Processor: AMD® Ryzen 7 2700x eight-core processor × 16
   - Graphics: GeForce RTX 2070/PCIe/SSE2
   - GNOME: 3.28.2

2. What performance results did you get?
   - Array size: 16384, NUMTRIES: 100

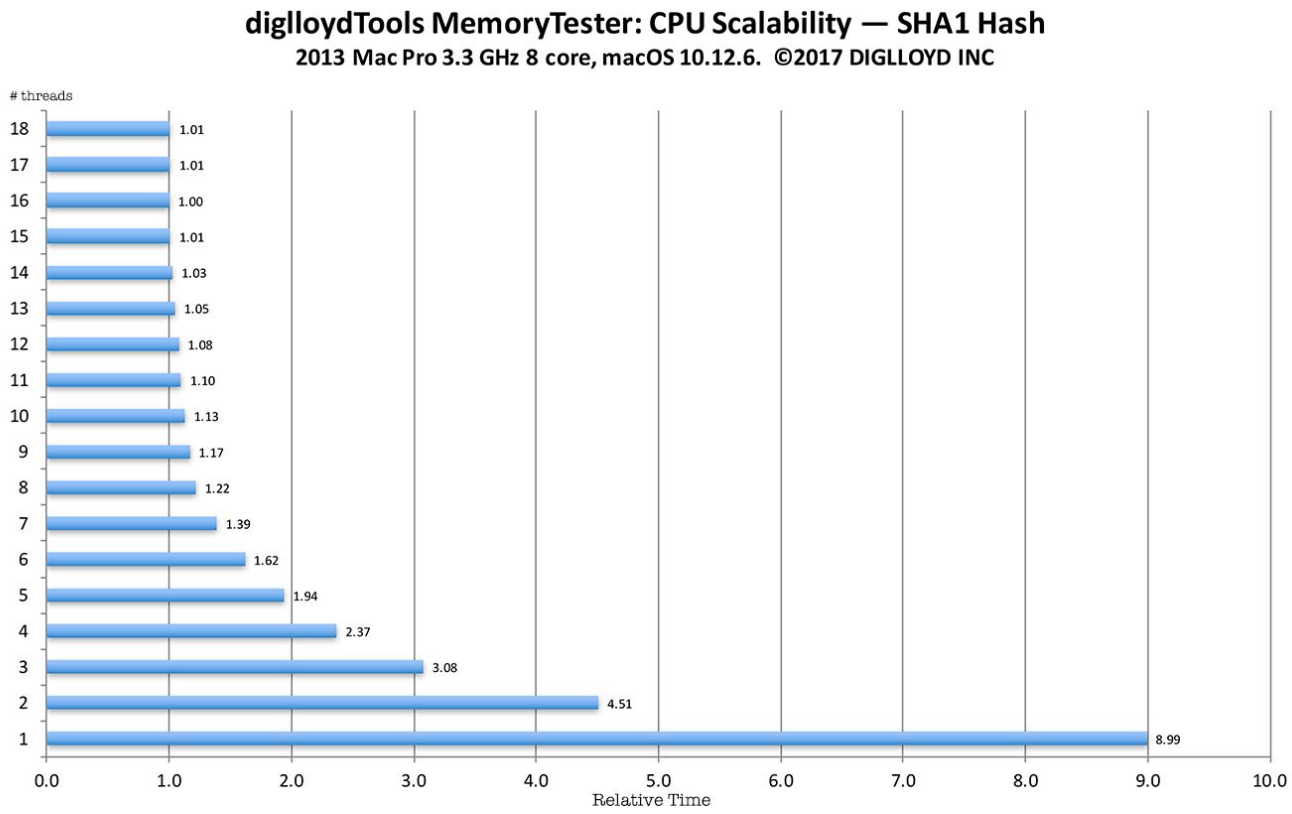
	1 thread	4 threads
1 time	234.69	454.38
2 times	234.56	481.68
3 times	235.13	505.04
4 times	234.72	486.27
5 times	172.68	454.38
6 times	172.72	465.11
7 times	234.93	465.38
8 times	172.99	480.69
9 times	172.96	520.31
10 times	234.76	460.91
Average	210.014	477.415

3. What was your 4-thread-to-one-thread speedup?

- **S** = (Performance with four threads) / (Performance with one thread) = 477.415 / 210.014 = 2.27

4. If the 4-thread-to-one-thread speedup is less than 4.0, why do you think it is this way?
   - At first, the fact is that multi-threads and good CPU scalability make the machine run faster than a single thread. However, it will not always reflect in proportion to the number of added threads. In other words, the machine will not be n times faster for adding n threads because that proportion would require the high precision to be partitioned into n independent thread parts. Furthermore, technically, all threads and CPU cores would not work (some threads and cores would take a break) if the inputted task is small.

   - Related data chart from other source (https://macperformanceguide.com/blog/2017/20171223_2311-why-more-cores-often-dont-help.html)



5. What was your Parallel Fraction, Fp?
   - float Fp = (4./3.)*(1. - (1./S)), where S = 2.27
     Fp = 0.75