Merge Sort algorithm

1. In a table summarize the duration of time for your single thread and multi-thread merge sort algorithms for the following unsorted array sizes of {10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7}

duration				
Array Size	Single thread	4 threads	8 threads(number of processors in my machine)	10 threads
10	00.00	00.03	00.09	00.14
10^2	00.00	00.04	00.14	00.14
10^3	00.00	00.07	00.12	00.14
10^4	00.00	00.04	00.11	00.18
10^5	00.04	00.06	00.14	00.21
10^6	00.43	00.20	00.22	00.26
10^7	04.83	01.50	01.39	01.49
10^8	50.64	16.08	17.15	17.54

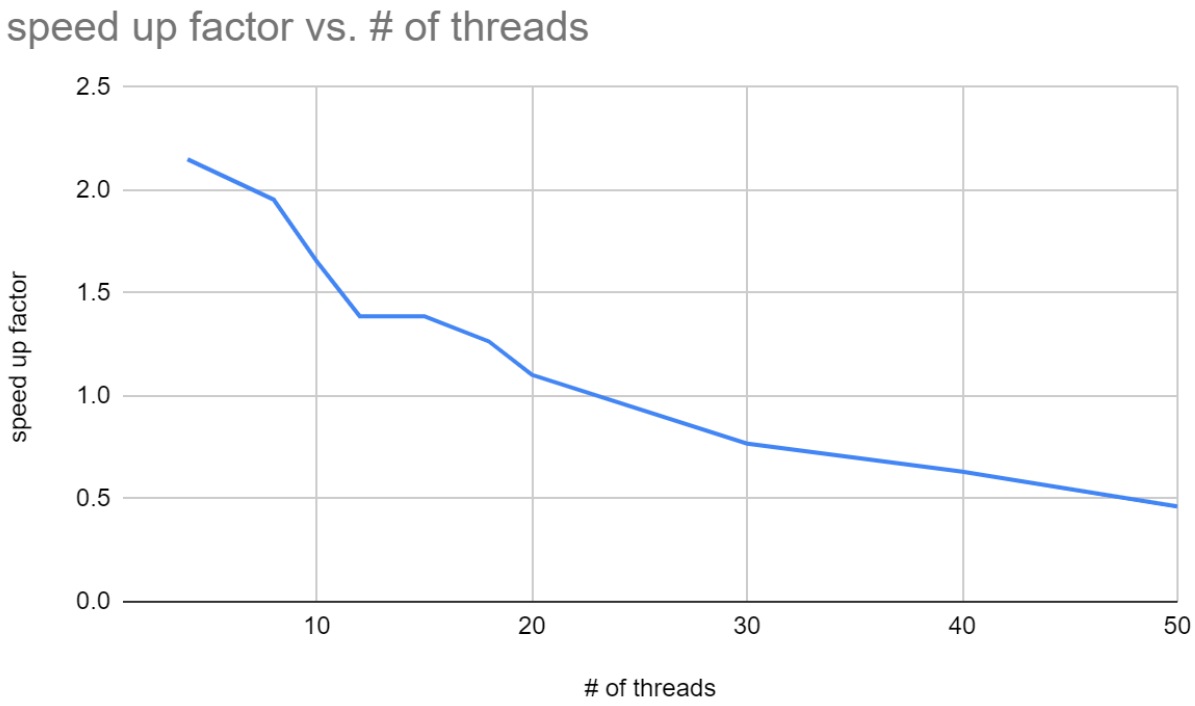
2. How much speed-up were you expecting based on the number of processors/cores on your machine?

I was expecting it to be 6 times faster since my computer has 8 cores. I imagined the other 2 cores were used to run other tasks for my computer.

3. Did you achieve the speed-up you expected? If not, what do you think might be interfering with this?

No, I think overhead, thread creation, memory allocation, and the fact that they are short calculations may cause the threads to appear to be sequential when they are in parallel. So by the time the next thread is started the last thread is close to being done or even done. Allocating memory also adds to the overhead cost of multithreading

4. In your parallel implementation, try different number of threads for an array size of million elements. Observe the speed up factor as a function of thread size, i.e. speed-up factor (Y-axis) for #threads increasing (X-axis). Summarize your results in a table in your README file in the repo.

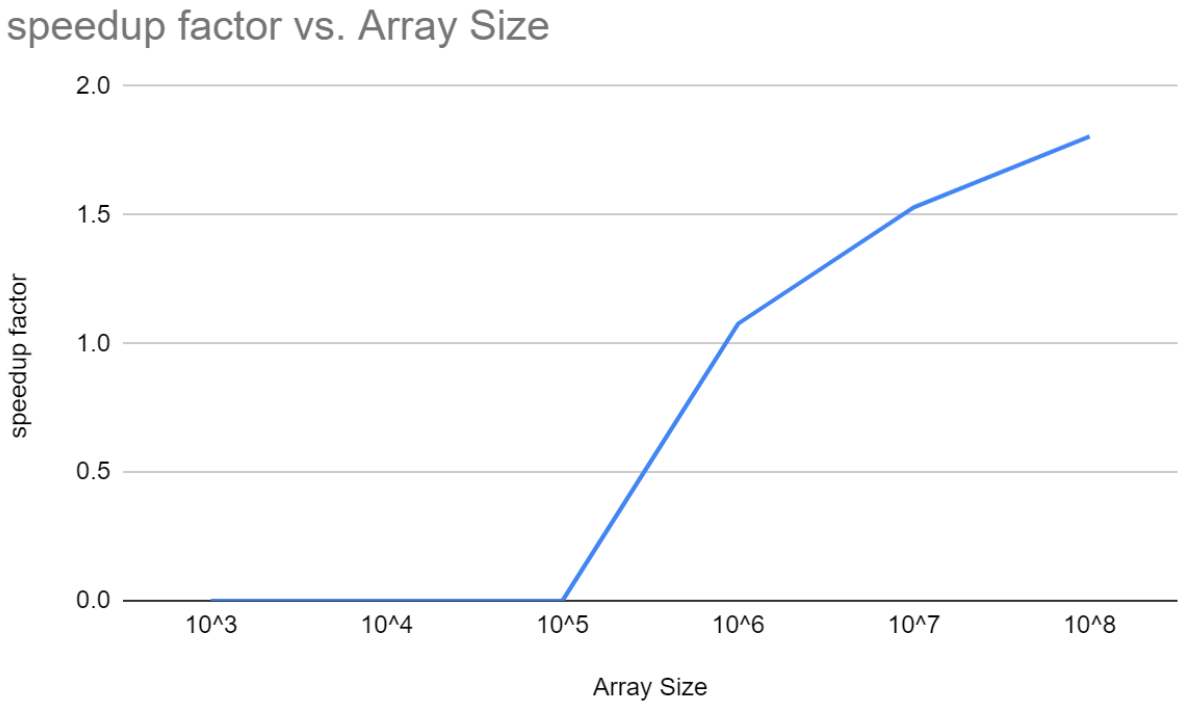


# of threads	1	4	8	10	12	15	18	20	30	40	50
speed up factor		2.15	1.9545 45455	1.6538 46154	1.3870 96774	1.3870 96774	1.2647 05882	1.1025 64103	0.7678 571429	0.6323 529412	0.4623 655914
duration	0.43	0.2	0.22	0.26	0.31	0.31	0.34	0.39	0.56	0.68	0.93

Monte Carlo Pi estimation

	duration		
	Single	Multi Thread	Speedup factor

Array Size	thread		
10^3	00.00	00.05	0
10^4	00.00	00.06	0
10^5	00.00	00.08	0
10^6	00.28	00.26	1.076923077
10^7	02.60	01.70	1.529411765
10^8	24.22	13.42	1.804769001



1. What have you learned in terms of splitting up work between threads?

Having more threads is not always a good thing, threads take up alot of overhead time to initiate and splitting up work may not always be the best option, but it helps with large numbers of repetitive calculations.

2. What implications does this have when designing concurrent code?

It is resource-costly and not always beneficial to use a large number of threads during multithreading, but optimizing the number of threads to the purpose you want to execute may be beneficial. And using a single thread for small computations is faster and less expensive in terms of memory and resources.

3. How many samples do you *think* you will need for an accuracy of 7 decimal places? Is the Monte Carlo simulation an efficient method to estimate Pi with high accuracy (feel free to research in the internet)?

I would say around 100 billion, i had a 5 digit accurate answer with 1 billion samples, i do not think Monte Carlo is an efficient method to estimate Pi since it is mainly based on probability and according to ACCURACY AND EFFICIENCY OF MONTE CARLO METHOD by Julius Goodman “The accuracy of the Monte Carlo method of assessment ...is significantly lower than what is widely believed.” The limit of Monte Carlo is the source of randomness in the results. By construction of these methods, it cannot be mathematically proved, but only “confidence interval” results. Monte Carlo is very slow if we want to obtain a fairly accurate result.