DUE DATE: **By the end of your lab session**

High Performance Computing
(CSE 443/543)
Exercise #16
Points: 20

Name: Maciej Wozniak

Objective of this exercise is to understand the concepts related to analytical performance modeling and analysis of parallel programs. You are expected to complete these exercises after specific topics have been covered in the class.

For each problem in this exercise, wait until your instructor indicates that you may commence working on a specific problem.

You are permitted to discuss any aspect of the exercise with your instructor or neighbors in the class.

1. The serial runtime (`Ts`) of a given program was measured to be 25 seconds. The parallelized version of the program was run using 5 threads/processes. The start and end runtimes of each of the 5 parallel process is shown in the adjacent table. What is the effective parallel runtime (`Tp`)? Is the parallel version faster or slower than the serial version?

Rank	Time (mm:ss)	
	Start	Stop
0	00:01	00:14
1	00:01	00:10
2	00:00	00:12
3	00:02	00:15
4	00:03	00:16

   Note: In this problem you are intentionally ignoring confidence interval values to make the problem easier. In practice you will have multiple values from which average and 95% CI value must be derived and used for analysis.

   The parallel runtime (`Tp`) of the program is: *16 seconds (we look at the earliest start and latest stop)*

   Is the parallel version *faster* or *slower* than the serial version of the program? *Faster*

2. What is the total overhead (`To`) for the parallel algorithm from the previous question?

   The total overhead (`To`) = 55 sec (16*5 – 25 sec) – Tp * number of runtimes

3. The runtime of a parallel searching algorithm (called Par. Search) that was run using 10 cores is being pitched against 3 serial searching algorithms. The timings recorded for searching a set of 1,000,000 entries is tabulated in the table below:

Algorithm	**Sequential algorithms**			Par. Search
	Linear	Binary Search	Block search	
Runtime (sec)	125±0.5	15.5±0.25	50.5±1.5	10.5±1.5
Time complexity	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(\log n)$

What is the actual speed up realized by `Par. Search`:

S = Ts/Tp = 15.5/10.5 = 1.48

What is the asymptotic speedup achieved by `Par. Search`:

O(log(n)/log(n)) = 1
Since the complexity is the same, we wont see the particular speed up

4. Using the timing and computational complexity of the four searching algorithms presented in the previous question, compute the actual Efficiency of the parallel searching program.

Actual efficiency of `Par. Search` is:

10logn?4

5. An image search program takes 2 seconds to read, 1 second to write, and 5 seconds to process an image. If the processing is parallelized, then what is the maximum speed-up that can be realized using 4 cores?

Amdahl's Law
Ts = 8s
Tp =2+1+(5/4)
Tp/Ts
1.88x

6. Briefly describe the two reasons for superlinear speed-up?

We use enough cores that problem can fit in cache. Then speed up can be more than number of cores. Exploratory decomposition

7. Asymptotic comparison of the computational complexity metrics of four different algorithms (namely: A1, A2, A3, and A4) for sorting a given list of numbers is shown in the table below. *n* is number of values in the list. The number of processors (***p***) needed for the algorithms is shown in terms of *n*.

	A1	**A2**	**A3**	**A4**
p (#processors)	n^2	$\log n$	n	$\sqrt{n}$
Tp (runtime)	1	n	$\sqrt{n}$	$\sqrt{n} \log n$
S (speedup)	$n \log n$	$\log n$	$\sqrt{n} \log n$	$\sqrt{n}$
E (efficiency)	$\log n / n$	1	$\log n / \sqrt{n}$	1
pTp (cost)	n^2	$n \log n$	$n^{1.5}$	$n \log n$

   a. Given unlimited compute nodes and need to rapidly solve problems, which algorithm would you choose?

      A1 -const runtime at n^2 compute nodes

   b. If the objective is to maximize effective utilization of CPU time, which algorithm would you choose?

      A2 – efficiency of 1, linear runtime. Speedup is faster. Number of processors grows slower

   c. If the objective is to obtain the "greenest" (most energy efficient) algorithm, which algorithm would you choose?

      A2 – perfect efficiency and linear runtime

8. What is scalability?

   Efficiency remains constant, when then number of inputs and cores increases

9. The following timing values (all timings are in seconds) were empirically measured to compare a serial and a parallel algorithm:

N	Serial Timing (sec)	Parallel Timing (sec)	
		p=1	p=10
64	48	64	7
128	96	120	12.5

Is the parallel algorithm scalable? Support your inference using suitable quantitative metric(s).

Yes, when number of inputs is the same, parallel solution is around 7x faster. Efficiency is almost the same that means that the algorithm is scalable.

Solution Tip: In order to effectively answer the above question you need to determine the efficiency of the parallel program for each configuration by filling in the following table and then determine if Efficiency is preserved as number of processors is increased.

n	p=1		p=10	
	Speedup	Efficiency	Speedup	Efficiency
64	.75	.8	6.85	0.685
128	.8	.8	7.86	0.786