Parallel Programming WS16 HOMEWORK (with solutions)

Performance Metrics

1 Basic concepts

1. **Performance**. Suppose we have two computers A and B. Computer A has a clock cycle of 1 ns and performs on average 2 instructions per cycle. Computer B, instead, has a clock cycle of 600 ps and performs on average 1.25 instructions per cycle. Consider the program P, which requires the execution of the same number of instructions in both computers:

   - Which computer is faster for this program?
   - What if P, when executed in Computer B, performs 10% more instructions than when executed in Computer A?

   Solution.

   Computer A performs $\frac{\text{2 instructions}}{\text{1 cycle}} \times \frac{\text{1 cycle}}{10^{-9}\text{ seconds}} = 2 \times 10^9 \frac{\text{instructions}}{\text{second}}$

   Computer B performs $\frac{\text{1.25 instructions}}{\text{1 cycle}} \times \frac{\text{1 cycle}}{600\times 10^{-12}\text{ seconds}} = 2.08 \times 10^9 \frac{\text{instructions}}{\text{second}}$

   Computer B performs more instructions per second, thus it is the fastest for this program.

   Now, let's n be the number of instructions required by Computer A, and $1.1 \times n$ the number of instructions required by Computer B. The program will take $\frac{n}{2\times 10^9}$ seconds in Computer A and $\frac{n}{1.89\times 10^9}$ seconds in Computer B. Therefore, in this scenario, Computer A executes the program faster.

2. **Speedup**. Assume the runtime of a program is 100 seconds for a problem of size 1. The program consists of an initialization phase which lasts for 10 seconds and cannot be parallelized, and a problem solving phase which can be perfectly parallelized and grows quadratically with increasing problem size.

   - What is the speedup for the program as a function of the number of processors p and the problem size n.
   - What is the execution time and speedup of the program with problem size 1, if it is parallelized and run on 4 processors?
   - What is the execution time of the program if the problem size is increased by a factor of 4 and it is run on 4 processors? And on 16 processors? What is the speedup of both measurements?

Solution.

The application has an inherently sequential part (c_s) that takes 10 seconds, and a parallelizable part (c_p) that takes 90 seconds for problem size 1. Since, the parallelizable part grows quadratically with the problems size, we can model T_1 (execution time in 1 processor) as:

$$c_s + c_p \times n^2.$$

The function of the speedup is, thus,

$$S(p,n) := \frac{T(1,n)}{T(p,n)} := \frac{c_s + c_p \times n^2}{c_s + (c_p \times n^2)/p} \equiv \frac{10 + 90 \times n^2}{10 + (90 \times n^2)/p}.$$

For problem size 1 ($n = 1$) and 4 processors ($p = 4$), the execution time is 32.5 seconds. The achieved speedup is 3.08.

Finally, if problem size is increased to 4, the execution time and speedup using 4 and 16 processors is:

- 4 processors: 370 seconds, speedup of 3.92.
- 16 processors: 100 seconds, speedup of 14.5.

3. **Amdahl's law**. Assume an application where the execution of floating-point instructions on a certain processor P consumes 60% of the total runtime. Moreover, let's assume that 25% of the floating-point time is spent in square root calculations.

   - Based on some initial research, the design team of the next-generation processor $P2$ believes that they could either improve the performance of all floating point instructions by a factor of 1.5 or alternatively speed up the square root operation by a factor of 8. From which design alternative would the aforementioned application benefit the most?
   - Instead of waiting for the next processor generation the developers of the application decide to parallelize the code. What speedup can be achieved on a 16-CPU system, if 90% of the entire program can be perfectly parallelized? What fraction of the code has to be parallelized to get a speedup of 10?

Solution.

Amdahl's law: $S_p(n) := \frac{1}{\beta + (1-\beta)/p}$.

- Improvement 1 (all fp instructions sped up by a factor 1.5). Sequential part (β): 0.4. $p = 1.5$. The application would observe a total speedup of:

$$S_p(n) := \frac{1}{\beta + (1-\beta)/p} = \frac{1}{0.4 + (1-0.4)/1.5} = 1.25.$$

- Improvement 2 (square root instructions sped up by a factor of 8). Sequential part (β): 0.4 + 0.45. $p = 8$. The application would observe a total speedup of:

$$S_p(n) := \frac{1}{\beta + (1-\beta)/p} = \frac{1}{.85 + (1-0.85)/8} = 1.15.$$

Thus, the application would benefit the most from the first alternative.

Parallelization of code. The speedup achieved on a 16-CPU system is:

$$S_p(n) := \frac{1}{\beta + (1-\beta)/p} = \frac{1}{0.1 + (1-0.1)/16} = 6.4.$$

To attain a speedup of 10, a 96% of the code would need to be perfectly parallelizable. This value is obtained by solving the equation:

$$10 == \frac{1}{\beta + (1-\beta)/16}.$$

4. **Efficiency**. Consider a computer that has a peak performance of 8 GFlops/s. An application running on this computer executes 15 TFlops, and takes 1 hour to complete.

   - How many GFlops/s did the application attain?
   - Which efficiency did it achieve?

   Solution.

   The application attained: $\frac{15 \text{ TFlops/s}}{3600 \text{ s}} = 4.17$ GFlops/s.

   The achieved efficiency is: $\frac{4.17 \text{ GFlops/s}}{8 \text{ GFlops/s}} = 52\%$.

5. **Parallel efficiency**. Given the data in Tab. 1, use your favorite plotting tool to plot

   a) The scalability of the program (speedup vs number of processors)
   b) The parallel efficiency attained (parallel efficiency vs number of processors)

   In both cases plot also the ideal case, that is, scalability equal to the number of processors and parallel efficiency equal to 1, respectively.

# **Processors**	Best seq.(1)	2	4	8	16
# **GFlops/s**	4.0	7.6	14.9	23.1	35.6

Table 1: Performance attained vs number of processors.

Solution.

Table 2 includes the speedup and parallel efficiency attained. Figure 1 gives an example of the requested plots.

# **Processors**	Best seq.(1)	2	4	8	16
# **Speedup**	1	1.9	3.725	5.775	8.9
# **Par. Eff.**	1	0.95	0.93	0.72	0.56

Table 2: Performance attained vs number of processors.

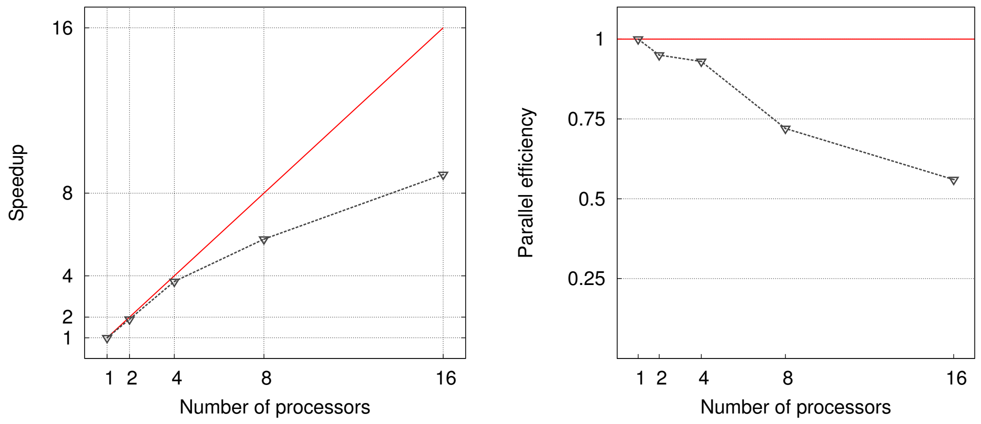

Figure 1: Scalability and parallel efficiency.

6. **Weak scalability**. Algorithm $\mathcal{A}$ takes an integer k as input, and uses $O(k^2)$ workspace. In terms of complexity, $\mathcal{A}$ is cubic in k. Let $T_p(k)$ be the execution time of $\mathcal{A}$ to solve a problem of size k with p processes. It is known that $T_1(8000) = 24$ minutes. Keeping the ratio workspace/processor constant, and assuming perfect scalability, what is the expected execution time for $k = 128000$?

   Solution.

   We have:

   - Workspace complexity: $O(n^2)$
   - Computational complexity: $O(n^3)$
   - $T_1(8000) = 24$ minutes

   We have to find p and t in:

   - $T_p(128000) = t$ minutes

   We start by finding the necessary number of processors. The problem size k is 16 times larger (because $\frac{128000}{8000} = 16$). Since the workspace complexity is quadratic, we will need 256 times more memory. To keep the workspace/processor constant, we will need 256 times more processors. Therefore, $p = 256$.

   Now, to compute $T_{256}(128000)$ we can start from $T_1(8000)$ and work our way towards the solution. First we compute $T_1(128000)$ as

$$T_1(128000) = 16^3 \times T_1(8000) = 16^3 \times 24 \text{ minutes},$$

   because we have cubic computational complexity. Then, since we have perfect scalability, $T_{256}(128000)$ is simply

$$T_{256}(128000) = T_1(128000)/256 = 16 \times 24 = 384 \text{ minutes}.$$

2 Real world scenarios

1. A given program was run on a node consisting of 4 multi-core processors, each of which comprises 10 cores. The peak performance of the node is 320 GFlops/s. To study the scalability of the program, it was run using 1, 2, 4, 8, 16, 24, 32 and 40 of the cores, for fixed problem size. For this problem size, the program executes 22×10^{12} flops. The execution time for each number of cores is collected in Tab. 3. Complete the table with the performance, speedup, efficiency, and parallel efficiency for each case.

# Cores	**Time (s)**	**Perf. (GF/s)**	**Eff.**	**Speedup**	**Par. Eff.**
1	34566.23	0.64	0.08	1.00	1.00
2	17133.64	1.28	0.08	2.02	1.01
4	8535.92	2.58	0.08	4.05	1.01
8	4326.95	5.08	0.08	7.99	1.00
16	2206.02	9.97	0.08	15.67	0.98
24	1531.88	14.36	0.07	22.56	0.94
32	1133.53	19.41	0.08	30.49	0.95
40	943.30	23.32	0.07	36.64	0.92

Table 3: Study of performance, speedup and efficiency.

2. A program P consists of parts A, B and C. Part A is not parallelized while parts B and C are parallelized. The program is run on a computer with 10 cores. For a given problem size, the execution of the program in 1 of the cores takes 10, 120, and 53 seconds, for parts A, B, and C, respectively. Answer the following questions:

   - Which is the minimum execution time we can attain for each part if we now execute the same program with same problem sizes on 5 of the cores? What is the best speedup we can expect for the entire program?
   - What if we run it using the 10 cores?

   In practice, the execution of the program using 5 and 10 cores to the time shown in Tab. 4. What is the speedup attained in each case for each part?

# Cores	**Part A(s)**	**Part B (s)**	**Part C(s)**
1	17	120	53
5	17	33	18
10	17	17	9

Table 4: Timings for program P using 1, 5 and 10 cores.

Solution.

The minimum possible execution time for each part, for 5 and 10 cores, is given in Tab. 5.

# Cores	Part A(s)	Part B (s)	Part C(s)
1	17	120	53
5	17	24	10.6
10	17	12	5.3

Table 5: Minimum possible timings for program P using 1, 5 and 10 cores.

The best speedup we can achieve in each case is:

- For 5 cores: $S_5 = \frac{T_1}{T_5} = \frac{190}{17+173/5} = 3.68$
- For 10 cores: $S_{10} = \frac{T_1}{T_{10}} = \frac{190}{17+173/10} = 5.54$

The speedup achieved in practice is:

- For 5 cores: $S_5 = \frac{T_1}{T_5} = \frac{190}{68} = 2.79$
- For 10 cores: $S_{10} = \frac{T_1}{T_{10}} = \frac{190}{43} = 4.42$