Parallel Programming WS15 HOMEWORK #2 (Solutions)

1 Basic concepts

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

   - Which computer is faster for this program?
   - What if Computer B required a 10% more instructions than Computer A?

   Solution.

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

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

   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 an application for a problem is 100 seconds for problem size 1. It 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 quadratic with growing problem size.

   - What is the speedup for the given application as a function of the number of processors p and the problem size n.
   - What is the execution time and speedup of the application with problem size 1, if it is parallelized and run on 4 processors?
   - What is the execution time of the application if the problem size is increased to 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 it 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 code 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: $\frac{15 \text{ TFlops/s}}{3600 \text{ s}} = 4.26$ GFlops/s.

   The achieved efficiency is: $\frac{4.26 \text{ GFlops/s}}{8 \text{ GFlops/s}} = 53\%$.

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.

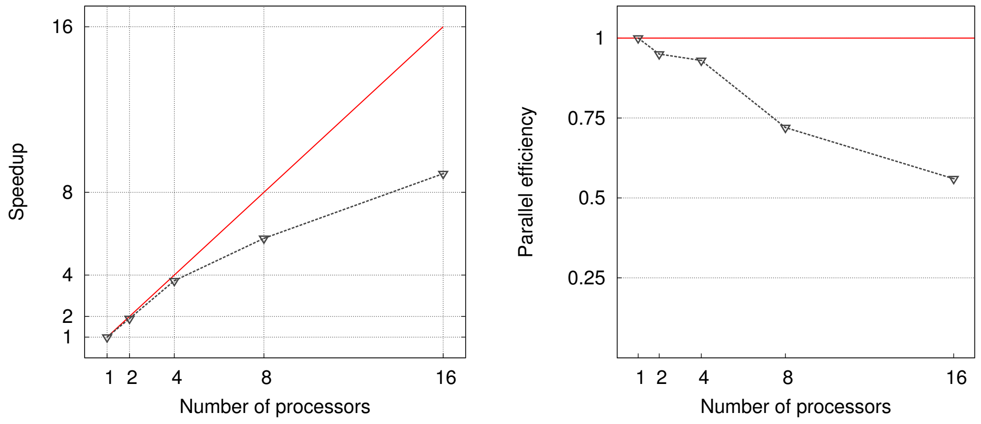

Figure 1: Scalability and parallel efficiency.