SF2568 Parallel Computing

Sebastian Myrbäck

February 2024

1. **Describe the difference between a process and a processor!**

   A process is a computer program that executes instructions, using the processor as aid for doing computations. The process occupies a certain memory space and a set of resources, that are relatively isolated from other processes occuring.

   A processor is a Central Processing Unit (CPU), which is a hardware component in computers that deals with executing instructions, such as arithmetic operations and data movement. In essence, the processor reads instructions from the program, decodes them and executes them accordingly.

2. **A multiprocessor consists of 100 processors, each capable of a peak execution rate of 2 Gflops (i.e floating point operations per second). What is the performance of the system, as measured in Gflops when 10% of the code is sequential and 90% is parallelizable?**

   Using Gustafson's law, we have that the scaled speedup is given by

   $$S'_P = f' + (1 - f')P$$

   where f' is the fraction of serial time of the program, and P is the number of processors.

   From the question, we have $f' = 0.1$, $P = 100$, which results in $S'_P = 0.1 + 0.9 \cdot 100 = 90.1$.

   The performance is given by the speed up times the peak execution rate of each processor, so the total performance of the system is given by $2\text{Gflops} \times 90.1 = 180.2\text{Gflops}$. The result would differ slightly if one chose to follow Amdahl's law.

3. **Is it possible for a system to have a system efficiency (η_P) of greater than 100%? Discuss.**

   We define parallel efficiency as

   $$\eta_P = \frac{S'_P}{P} = \frac{f' + (1 - f')P}{P}.$$

We can write $\eta_P = (f'(1-P)+P))/P$ and since $f' \in [0,1]$ and $P \geq 1$, we can estimate it from above and below as:

$$\frac{f'(1-P)+P}{P} \leq \frac{0+P}{P} = 1$$
$$\frac{f'(1-P)+P}{P} \geq \frac{1-P+P}{P} = \frac{1}{P}$$

and we conclude that $\eta_P \in [1/P, 1]$, i.e. η_P can *not* be greater than 100%.

4. **In the Parallel Rank Sort method presented in the lecture, the number of processors must be the same as the number of elements in the list. Assume that the number of elements in the list is actually ten times larger than the number of processors in the computer. Describe a modification to handle this situation.**

   Denote the list by A such that $size(A) = 10P$, where P denotes the number of available processors. Since P is clearly a positive integer, the size of A is divisible by 10, and we can thus divide A into ten sublists A_i, $i = 1, 2, \ldots, 10$ such that $size(A_i) = P$ for all i. Now we can construct a program which iterates through i, that is, the number of sublists. For each sublist, one can apply Parallel Rank Sort as in the lecture, since the size of the sublist to be sorted is the same size as the number of available processors.

   Once all sublists has been sorted internally, one can do a final global sort using e.g. priority queue or min-heap sorting.

5. **You are given some time on a new parallel computer. You run a program which parallelizes perfectly during certain phases, but which must run serially during others.**

   (a) **If the best serial algorithm runs in 64s on one processor, and your parallel algorithm runs in 22s on 8 processors, what fraction of the serial running time is spent in the serial section?**

   (b) **Determine the parallel speedup.**
   The speedup is given by $S_P = 64/22 = 32/11$ (answer to (b)). Using Amdahl's law, we have

   $$\frac{32}{11} = \frac{P}{1+(P-1)f} = \frac{8}{1+7f}.$$

   Solving for f gives $f = 1/4$ (answer to (a)).

   (c) **You have another program that has three distinct phases. The first phase runs optimally on 5 processors; this is, performance remains constant on 6 or more processors. The second phase runs optimally on 10 processors, and the third on 15. The pases consume 20%, 20% and 60% of the serial running time, respectively. What is the speedup on 15 processors?**

Let T_1 denote the total serial running time. Let f_i and P_i^{opt} denote the fraction of the serial time spent in phase i and the number of processors that runs phase i optimally, respectively, for $i = 1, 2, 3$. Then, assuming the program has no serial fractions, the time of the parallel program is given by

$$T_P = \frac{f_1 T_1}{P_1^{\text{opt}}} + \frac{f_2 T_1}{P_2^{\text{opt}}} + \frac{f_3 T_1}{P_3^{\text{opt}}},$$

and consequentially, the speedup is given by

$$S = \frac{T_1}{T_P} = \frac{1}{\frac{f_1}{P_1^{\text{opt}}} + \frac{f_2}{P_2^{\text{opt}}} + \frac{f_3}{P_3^{\text{opt}}}} = \frac{1}{\frac{0.2}{5} + \frac{0.2}{10} + \frac{0.6}{15}}.$$