Assignment 1 - Solutions

ECSE 420 – Parallel Computing – Fall 2014

November 2, 2014

1. (15%) Describe briefly the following terms, expose their cause, and work-around the industry has undertaken to overcome their consequences: (i) Memory wall, (ii) Frequency wall, and (iii) Power wall.

(i) Memory wall: CPU performance in the past has increased too fast for the matching system memory, which yields memory which is below the system performance. Therefore this creates a bottleneck while a CPU is waiting for memory to be transferred to/from memory. To overcome this issue, industry has introduced caches (L1, L2, L3) between the CPU and system memory which are much faster, however small. Other methods created are things such as out-of-order execution and instruction-level parallelism.

(ii) Frequency wall: The frequency wall is the limit at which CPU clock cycles can be shortened. The reason for this is the transistors inside the CPU have a defined maximum switching time before heat dissipation as well as frequency interference becomes a problem. These limits also include such things as power consumption and the physical size of the transistors and in relation the chips) To overcome the frequency wall, systems have added enhanced cooling techniques as well as delegating multiple cores of lower frequency allowing for greater parallelism rather than simply higher clock speed.

(iii) Power wall: This term describes CPU heat dissipation problems. As transistors increase their frequency, heat dissipation increases. Approaches to overcome the power wall is to use enhanced cooling methods. However these cooling systems can end up being quite expensive.

NB: The above answers may not cover all possibillities, thus may differ slightly compared to yours. As long as the main idea for each problem is properly addressed and described, your response is accepted as valid answer.

2. (15%) A uniprocessor application is parallelized for 4 processors, yielding a 3.8× speedup. Given the time breakdown of the various functions seen in the graph, what is the minimum total time that the uniprocessor application spent while Busy and in performing Data Access?

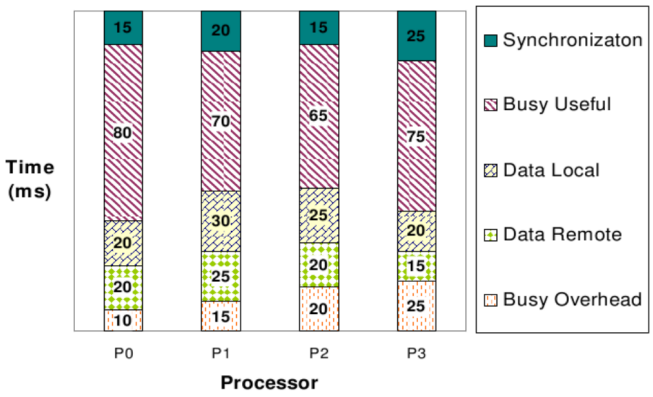

Figure 1: Time breakdown of the various functions

Let T_i be the total time spend per processor P_i. The speedup can be expressed as:

$$\text{Speedup} \leq \frac{\text{Sequential Time}}{\text{Max}(T_0, T_1, T_2, T_3)} \Rightarrow$$

$$3.8 \leq \frac{\text{Sequential Time}}{160ms} \Rightarrow$$

$$\text{Sequential Time} \geq 3.8 \cdot 160ms = 608ms$$

3. (20%) For a machine with the communication overhead, assist occupancy and network delay (message start-up time) of 200 ns and the asymptotic peak bandwidth of 5 GB/s, calculate the message lengths for reaching the 1/3rd and 2/3rd of the peak bandwidth. Assuming that the contention at these two points adds 300 and 500 ns to the start-up time, respectively, what message lengths would then be obtained?

From lecture slides, the total Transfer time is:

$$T(n) = T_0 + \frac{n}{B}$$

where n is the message length, B is the asymptotic peak bandwidth, and T_0 the message start-up time. Let B' be the actual bandwidth of our system:

$$B' = \frac{n}{T(n)} = \frac{n}{T_0 + \frac{n}{B}}$$

The actual bandwidth is a fraction of the asymptotic peak bandwidth $B' = \lambda B$. For the message length to be computed, we have:

$$B' = \frac{n}{T_0 + \frac{n}{B}} \Rightarrow \lambda B = \frac{n}{T_0 + \frac{n}{B}} \Rightarrow \lambda B T_0 + \lambda n = n \Rightarrow n(1-\lambda) = \lambda B T_0 \Rightarrow n = \frac{\lambda}{1-\lambda} B T_0$$

Therefore, for the two asked scenarios we have:

- 1/3rd of the peak bandwidth and $200 + 300 = 500ns$ start-up time:

$$n = \frac{\frac{1}{3}}{1-\frac{1}{3}} B T_0 = \frac{1}{2} \times (500 \cdot 10^{-9} s) \times (5 \cdot 10^9 \frac{bytes}{s}) = 1250 bytes$$

- 2/3rd of the peak bandwidth and $200 + 500 = 700ns$ start-up time:

$$n = \frac{\frac{2}{3}}{1-\frac{2}{3}} B T_0 = 2 \times (700 \cdot 10^{-9} s) \times (5 \times 10^9 \frac{bytes}{s}) = 7000 bytes$$

NB : These answers may differ slightly if you do a conversion with 1024^3 for bytes/bits stuff. Both are accepted as valid answers here.

4. (20%) Suppose we have a program that can be divided into 5 concurrent tasks. One of these tasks requires twice the execution time than the other 4.

   (a) Based on the Amdhals law compute the speed up (S) of using 5 processors (one processor per task)

   (b) Extend Amdahls law by taking into consideration the fixed overhead O in the communication and the setup of parallel processes. Derive an expression for Amdahls bound that properly takes the overhead into account.

   (c) Given this new expression, which would be a more realistic value for the speed-up S computed above?

(a) From the lecture slides, the speed-up S is:

$$S = \frac{N}{SP \times N + (1 - SP)}$$

where N is the number of processors and SP the sequential part of our program. One of our tasks requires twice the execution time ($t_5 = 2 \times T$) than the other 4 ($t_1 = t_2 = t_3 = t_4 = T$). The total time for these tasks to be executed sequentially is $t_{1processor} = 6 \times T$ Using the 5 processors, we can execute the 4 tasks and the first half of the 5th task in parallel. In other words, the second half of the 5th task is the part of our program that cannot be executed in parallel:

$$SP = \frac{T}{6 \times T} = \frac{1}{6}$$

Thus, we have the asked speed-up:

$$S = \frac{5}{\frac{1}{6} \times 5 + (1 - \frac{1}{6})} = \frac{5}{\frac{5}{6} + \frac{5}{6}} = 3$$

(b) Taking the fixed overhead O into consideration, we have the running time of a program on N processors:

$$T_N = SP \times T_1 + \frac{(1 - SP) \times T_1}{N} + O$$

where N is the number of processors,SP the sequential part of our program and T_1 the running time of the program on 1 processor.
Thus, the speed-up is

$$S = \frac{T_1}{T_N} = \frac{T_1}{SP \times T_1 + \frac{(1-SP) \times T_1}{N} + O} = \frac{1}{SP + \frac{(1-SP)}{N} + \frac{O}{T_1}} \Rightarrow$$

$$\lim_{N \to \infty} \left(\frac{1}{SP + \frac{(1-SP)}{N} + \frac{O}{T_1}} \right) = \frac{1}{SP + \frac{O}{T_1}}$$

We must therefore keep O and SP as low as possible in order to achieve maximal speedup as N increases.

(c) Because of the fixed overhead O, a more realistic result would be a smaller value compared to the speed-up S we computed above.

5. (30%) Suppose we have a machine with 100 cores that access a grid of ($N * N = 10^5 * 10^5$) elements. Each one of these elements is of double floating point precision. We can partition the grid in two different ways:

- 1D - blocks of contiguous columns
- 2D - Two-dimensional domain decomposition (blocking)

Note that in both cases we have equally sized partitions but the message size between adjacent partitions changes. The message start-up time is 4 us (We assume that it includes all SW and HW overhead, accessing the network interface and cross the network it can be thought of as the time to send the zero-length message). In addition, the asymptotic peak bandwidth is 400 MB/s.

Compute the communication overhead for each case. Which one of the two partitions would you select, for this total overhead to be reduced?

We are asked to compute the total communication overhead per case:

$$T(n) = T_0 + \frac{n}{B}$$

where n is the total message length, B is the asymptotic peak bandwidth, and T_0 the message start-up time. We therefore have to evaluate the total amount of data (n bytes) that should be communicated between adjacent partitions.

- 1D - blocks of contiguous columns:

  $$n = 98 \times (2N \times 8\text{bytes}) + (N \times 8\text{bytes}) + (N \times 8\text{bytes}) = 198 \times (8 \cdot 10^5\text{bytes})$$

  So, we have the total transfer time:

  $$T(n) = 198 \times T_0 + \frac{n}{B} = 4 \cdot 10^{-6}s + \frac{198 \times (8 \cdot 10^5\text{bytes})}{400 \cdot 10^6 \frac{\text{bytes}}{\text{s}}}$$

  $$\Rightarrow T(n) = 198 \times 4 \cdot 10^{-6}s + 0.396s$$

- 2D - Two-dimensional domain decomposition (blocking):

  $$n = 64 \times (4 \cdot 10^4 \times 8\text{bytes}) + 32 \times (3 \cdot 10^4 \times 8\text{bytes}) + 4 \times (2 \cdot 10^4 \times 8\text{bytes}) = 360 \times (8 \cdot 10^4\text{bytes})$$

  So, we have the total transfer time:

  $$T(n) = 360 \times T_0 + \frac{n}{B} = 4 \cdot 10^{-6}s + \frac{360 \times (8 \cdot 10^4\text{bytes})}{400 \cdot 10^6 \frac{\text{bytes}}{\text{s}}}$$

  $$\Rightarrow T(n) = 360 \times 6 \cdot 10^{-6}s + 0.072s$$

We can easily observe that the second 2D partition is the most efficient choice for the total communication overhead to be reduced.

NB: These answers may differ slightly if you do not multiply the number of messages to be sent by the message start-up time T_0. As long as the communication scheme between adjacent partitions is properly described, both are accepted as valid answers here.