Assignment 1

ECSE 420 – Parallel Computing – Fall 2015

Released: September 14, 2015
Due: September 21, 2015 at 11:59 pm

Assignment 1 is to be completed individually. You should submit a single PDF containing all the required solutions and textual answers.

1. (15%) Describe the following terms, their cause, and the work-around the industry has undertaken to overcome their consequences: (i) memory wall, (ii) frequency wall, and (iii) power wall.

2. (40%) Suppose we have a program that can be divided into 5 tasks which do not depend on each other. 4 of these tasks take T seconds to run, and one task takes 2T seconds to run. We redesign this program to run on a multicore computer using threads.

   (a) What is the speedup obtained by running the 5 tasks as 5 threads on 5 separate cores? (Assume there is no OS overhead, and all 5 threads start at the same time.)

   (b) What is the maximum speedup obtained when there are 5 threads but only 4 cores? (Assume there is no OS overhead, and 4 of the threads start at the same time.)

   (c) Now assume that one of the 5 threads is a "master thread" which spawns the other threads. The master thread requires X seconds of OS overhead to spawn a worker thread. After those X seconds, the spawned thread begins to run. Once the master thread is finished spawning the 4 other threads, it runs the final task. What is the maximum speedup as a function of X and T? Make a diagram of the sequence of threads to back up your answer.

3. (20%) Write a C program using Pthreads which takes as command line input an integer N. The program should spawn N threads and report the order in which the threads finish their "task" (the task is to wait a random number of microseconds). For example, for N = 3, if thread 0, thread 1, and thread 2 finish their task in the order {2,0,1}, the program should print "2, 0, 1". You don't need to submit any extra files- just copy the body of your program into your PDF.

4. (25%) 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 spends while busy and in performing data access?

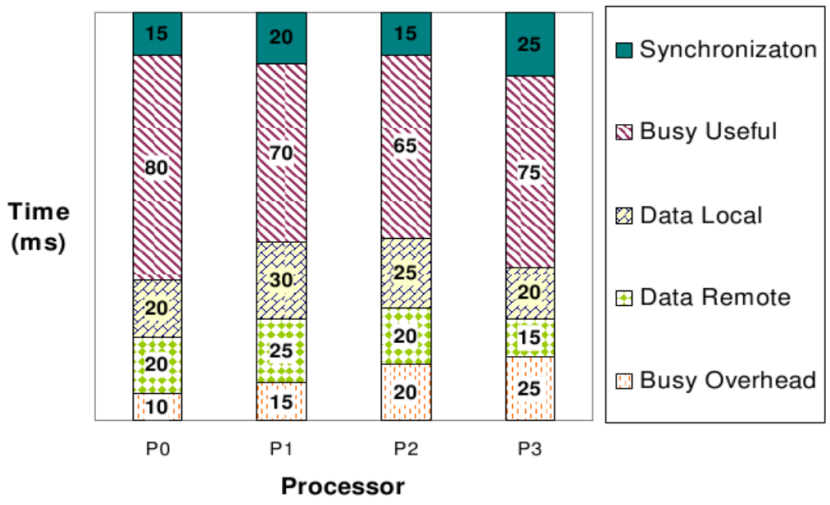

Figure 1: Time breakdown of the various functions