Assignment 1

PPCP

Due 9/10/2010

September 7, 2010

1. Amdahl's Law

   A new program was developed to better approximate the weather for the next eight days across the entire United States. This program employs parallelism in the hopes that it can calculate the weather in a reasonable amount of time. The meteorology company that developed this software has a cluster of 200 nodes at their disposal. The program was run on the cluster, but unfortunately finished in 6 days, 5 hours, and 45 minutes (using all 200 nodes). The company needs the calculation to take approximately one day so that the results are relevant to the news viewers (if a 7-day forecast comes out 6 days too late, what good is it?). For this program to be useful, it should finish calculating in a day or less.

   The company wants to get an idea of the size of computer cluster that they would need to obtain the results in one day. After they know the size of cluster needed, they will determine if they can afford to create such a cluster. You told them that they need to time the program on their cluster one more time using less nodes. They ran the program using 100 nodes and it finished in 11 days, 20 hours, and 38 minutes.

   Assuming that the parallelized portion of the code is perfectly parallelized (i.e. scales perfectly), and assuming that the code can be modeled according to Amdahl's Law, calculate the following:

   (a) How much time would this program take if it was run on only one process?

   (b) What percentage of the time of the original program was parallelized?
   (This refers to the percentage of time when run on one process)

   (c) What percentage of the time of the original program was sequential?
   (This refers to the percentage of time when run on one process)

   (d) With an infinite number of processors, how long would this program take to finish? (i.e. what is the limit to the time optimization?)
   (Note that parallelism can not get this program to finish in less than this amount of time.)

   (e) How many processors are needed to complete the computation within one day?

   Note: we are making some assumptions about this program without knowing any of the implementation details. Amdahl's law is only an approximation since there is no such thing as "perfect parallelism". The answer to part (e) will be an underestimation of the actual amount of processors needed to complete the computation in one day.

2. Alpaca

   This portion of the assignment is designed to give you some familiarity with the testing program Alpaca. Alpaca is a tool produced by Microsoft Research that puts several important tools at your fingertips, including the CHESS code level verifier. Before working on this section please be sure that complete the basic tutorial on Alpaca, which is located at http://www.eng.utah.edu/ cs5955/PPCP-Course-Material/ppcp/alpaca/Alpaca%20Introduction.pdf.

NOTE: To avoid the issue of frequently 'unblocking' the contents of your homework zip file, be sure, with Windows Vista and Windows 7, to right-click, select 'properties', and then click the 'unblock' button

For this exercise, you are provided with a Visual Studio 2010 solution, which has two (mostly) empty classes that you will use to perform some analysis on a class library, 'FlawedFibonacci', which is already referenced by your project (the .dll is included as well). The file 'Problem2a.cs', which contains the class you will use for the first part of the exercise, includes a method 'UsageExample()' that illustrates how to use the 'FlawedFibonacci' class. You may write as many test methods as necessary to solve the problems, but you must, for each subproblem, include the correctly named test method (specified below). You must also properly instrument your code so that the problem is illuminated by using Alpaca.

(a) Find the Flawed Fibonacci Number

For this sub-exercise, you must find the 'first' flawed number in the Fibonacci sequence produced by FlawedFibonacci. You may use any approach you like, but you must have a test method with this signature: 'public void Answer2a' that reveals the flawed number with an assertion that writes the following string: "Bad fibonacci number is X", where X is the bad number. When checking your work by running your test in Alpaca, we should see the required assertion string.

(b) Which is faster: recursive or iterative?

For this sub-exercise, you must determine which method of generating Fibonacci numbers, recursive or iterative (i.e. FlawedFibonacci.GetIterative() or GetRecursive()) is the fastest. You may have any number of helper functions, but the test method that you must use to generate the timing results (with Alpaca, using 'PerformanceTestMethod' attribute to decorate your method) will have this signature: 'public void Answer2b(int n)', where n is the number to generate. In other words, this method will execute tests for both methods and time them using the TaskMeter class. Another requirement of this exercise is that you need to direct Alpaca to run the test with the values of n: 10, 50 and 120. One other requirement: you should, through attributes, make the performance tests run with a WarmupRepetitions value of 2. You are expected to take snapshots of TaskoMeter GUI for each value of n.

3. Marty/Hill Paper Problem

For this assignment, you will use an online calculator to view various facets of performance scaling and Amdahl's Law. You will find the calculator here:
http://www.cs.wisc.edu/multifacet/amdahl/.

Using the online calculator, plot the following curves:

(a) Ratio of asymmetric to symmetric for the full range of r (x on the webpage; basically x BCEs are glommed on to make a large core. So at $x = 256$, there is one core, as you can see from the graph)

(a) For $f = 0.999, 0.995, 0.99, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5$

(b) Ratio of dynamic to symmetric for the above f's

(c) Ratio of dynamic to asymmetric for the above f's

Hint: their online calculator gives you absolute speedups; I want speed-up ratios. Use the same range of BCEs

(d) *Extra Credit*

Since you have the formulas, draw 3-D plots of BCEs on the x axis, f on the y axis, and speed-up on the z axis. Do this for symmetric, asymmetric, and dynamic.

Extra Credit is due one week later than the rest of the assignment

Handin Instructions

Your assignment should consist of the following files:

1. hw1-1.jpg or hw1-1.doc This is either the handwritten answer to problem1, scanned into .jpg format or typewritten in Microsoft Word format

2. hw1-2.zip This file is your cleaned and zipped Visual Studio project for problem 2

3. hw1-2-tm10.jpg, hw1-2-tm50.jpg, hw1-2-tm120.jpg These files are the snapshots of TaskoMeter for each required test of problem 2b.

4. hw1-3.jpg This file should have screen shots and drawings for problem 3.

These files must be handed in with the CADE labs handin tool. For the command line:
handin cs5955 hw1 [filename]
You may also use their web interface:
https://cgi.eng.utah.edu/webhandin/
***