CSCI 3366 (Introduction to Parallel and Distributed Processing) Spring 2001

Homework 4

Assigned: April 12, 2001.

Due: April 24, 2001.

Credit: 20 points.

> *Note:* The HTML version of this document may contain hyperlinks. In this version, hyperlinks are represented by showing both the link text, formatted like this, and the full URL as a footnote.

Contents

1 Overview

The *variance* of a set of N numbers $a_0, a_1, \ldots, a_{N-1}$ is defined to be the sum

$$(a_0 - avg)^2 + (a_1 - avg)^2 + \cdots + (a_N - avg)^2$$

where avg is the average of $a_0, a_1, \ldots, a_{N-1}$.

For this assignment, you are to write a multi-threaded program to compute the variance of a set of numbers. The textbook discusses using multiple threads to speed up the calculation of the sum of a set of numbers; it is not difficult to extrapolate from this discussion to an approach for using multiple threads to speed up calculating the variance of a set of numbers, if you break down the calculation into two steps:

1. Compute the average of the input numbers, using multiple threads to speed up the required calculation of the sum of the numbers.

2. Compute the variance as defined above, using multiple threads to speed up the calculation.

2 Details

2.1 Program input and output

Your program should take two command-line arguments:

1. The number of threads to use (call this P). This argument is required.
2. The number of input numbers to generate (call this N). This argument is optional.

If the program is started with one command-line argument (P), it should read its input numbers from standard input, continuing to read until end-of-file is encountered. It should accept any number of input numbers. Floating-point inputs are okay.

If the program is started with two command-line arguments (P and N), it should generate N input numbers using any reasonable technique for generating random numbers.

In either case, once it has read or generated its input numbers, the program should compute their variance and print the following output:

- The computed variance.
- The time required to compute the variance, from just after the input number are read/generated until after the variance has been computed.

2.2 Choice of programming language/library

You can write your program either (i) in C++ or C, using the POSIX threads library functions as we have been doing in class, or (ii) in Java using Java's built-in support for multi-threading. Whichever language you use, be sure your program compiles and executes correctly on the department's Linux machines.

2.3 A starter program

So that you do not have to write the tedious and non-parallel parts of this program, I am providing a sequential program that performs the required calculations. You can find it in variance.cc[1]. To compile and run this program, you will also need the `timer()` function, which is in file threads-timer.h[2].

(Obviously this program will not help you much if you write your program in Java. Such is life.)

2.4 Running the program

Once you have confirmed that your program is operating correctly (for small numbers of inputs), try running it for a large number of generated inputs and varying values of P (number of threads). Record at least half a dozen observations (different combinations of N and P) to see how running time varies with these two variables. Also record which machine you performed these experiments on. You may find it interesting to see whether multi-threading can help even if you have more threads than processors. FYI, machines known to have multiple processors include `SnowWhite.CS.Trinity.Edu` (4 processors) and the `Dwarf`n`.CS.Trinity.Edu` machines (2 processors each).

[1] `http://www.cs.trinity.edu/~bmassing/CS3366_2001spring/Homeworks/HW04/Problems/variance.cc`
[2] `http://www.cs.trinity.edu/~bmassing/CS3366_2001spring/SamplePrograms/threads-timer.h`

2.5 Collaboration

For this assignment, please work individually. As always, discussion of the assignment is allowed (encouraged, even), but for this assignment each person should do the actual coding independently.

3 Helpful hints, etc.

3.1 Sample programs to look at

You may find it useful to look at some of the example programs using multi-threading; see the sample programs page[3]. In particular you may find it useful to look at the two programs that compute the sum of N numbers. Both programs take the same approach to parallelizing the computation, but they implement it in slightly different ways: threads-sum-1.cc[4] makes use of global variables, while threads-sum-2.cc[5] takes a very C++-ish approach to passing the required data to the threads via parameters. Included files other than those from the standard library (e.g., `threads-timer.h`) should also be available linked from the sample programs page.

3.2 Cautionary comments

P (the number of threads) might not evenly divide N. Your code should be prepared to cope with this. At the very least, it should print an error message and stop.

4 What to turn in

Submit your completed program (`variance.c`, `variance.cc`, or `variance.java`), plus a text file containing your timing observations (as described in the "Details" section above), by e-mail as described in the Guidelines for Programming Assignments[6], using a subject header of "cs3366 hw 4". Please submit the timing observations as a plain text file.

[3] `http://www.cs.trinity.edu/~bmassing/CS3366_2001spring/SamplePrograms/index.html`
[4] `http://www.cs.trinity.edu/~bmassing/CS3366_2001spring/SamplePrograms/threads-sum-1.cc`
[5] `http://www.cs.trinity.edu/~bmassing/CS3366_2001spring/SamplePrograms/threads-sum-2.cc`
[6] `http://www.cs.trinity.edu/~bmassing/CS3366_2001spring/Notes/pgmguidelines/index.html`