CSCI 3366 (Introduction to Parallel and Distributed Processing), Spring 2002

Homework 4

Assigned: April 9, 2002.

Due: April 18, 2002, at 5pm. *Not accepted later than April 19 at noon.*

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 **1**

2 Details **2**
- 2.1 Program input and output . . . 2
- 2.2 Choice of programming language/library . . . 2
- 2.3 A starter program . . . 2
- 2.4 Running the program . . . 3

3 Helpful hints, etc. **3**
- 3.1 Sample programs to look at . . . 3
- 3.2 Compiling programs that use POSIX threads . . . 3
- 3.3 Cautionary comments . . . 3

4 What to turn in **3**

1 Overview

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

$$(a_0 - avg)^2 + (a_1 - avg)^2 + \cdots + (a_{N-1} - avg)^2$$

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

[1] A student has pointed out that a more standard definition of variance is this sum divided by N. For this assignment, use the incorrect definition, since this was pointed out after some students had submitted solutions.

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.

It should process these arguments as follows.

- 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++ with the POSIX threads library functions 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.

- A C++ version is in `variance.cpp`[2]. To compile and run this program, you will also need include file `timer.h`[3].
- A Java version is in `Variance.java`[4]. To compile and run this program, you will need additional classes found in files `Timer.java`[5] and `SimpleReader.java`[6].

[2] `http://www.cs.trinity.edu/~bmassing/CS3366_2002spring/Homeworks/HW04/Problems/variance.cpp`
[3] `http://www.cs.trinity.edu/~bmassing/CS3366_2002spring/SamplePrograms/timer.h`
[4] `http://www.cs.trinity.edu/~bmassing/CS3366_2002spring/Homeworks/HW04/Problems/Variance.java`
[5] `http://www.cs.trinity.edu/~bmassing/CS3366_2002spring/SamplePrograms/Timer.java`
[6] `http://www.cs.trinity.edu/~bmassing/CS3366_2002spring/SamplePrograms/SimpleReader.java`

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).

3 Helpful hints, etc.

3.1 Sample programs to look at

You may find it useful to look at some of sample programs[7], in particular the various programs to compute the sum of N numbers.

3.2 Compiling programs that use POSIX threads

Note that when you compile (and link) a program that uses the POSIX threads library, you need the flag `-pthread`. The sample programs page has an example of a Makefile[8] that takes care of this.

3.3 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.cpp` or `Variance.java`), plus the text file containing your timing measurements, as described in the Guidelines for Programming Assignments[9], using a subject header of "cs3366 hw 4".

[7] `http://www.cs.trinity.edu/~bmassing/CS3366_2002spring/Homeworks/HW04/SamplePrograms/index.html`

[8] `http://www.cs.trinity.edu/~bmassing/CS3366_2002spring/Homeworks/HW04/SamplePrograms/pthreads/Makefile`

[9] `http://www.cs.trinity.edu/~bmassing/CS3366_2002spring/Notes/pgmguidelines/index.html`