High-Performance Scientific Computing (MATH-GA 2011/ CSCI-GA 2945)

Homework Set 2

Due: September 19, 2012 · Out: September 12, 2012

Just like last week, turn in your homework as a repository named like "`hpc12-hw2-netid123`" on `http://forge.tiker.net`.

Also, for this homework, please ensure that your virtual machine sees at least two (real) cores. If your personal machine is single-core, please get in touch with us about access to a parallel machine. If you have a CIMS account, try running on one of the machines on this page[1].

Problem 1: OpenMP Deadlock

Do problem 6.12 in Rauber/Rünger. Turn in your solution in text form as `problem-1.txt` in your repository.

Problem 2: OpenMP Bugs

In the repository at `https://github.com/hpc12/hw2-problem2`, you'll find six OpenMP programs, each of which has one or more bugs, detailed in the comments at the start of the source file.

Note that these are well-known programs, and it's super-simple to find solutions for them on the web. We're aware of that. To state the obvious, you'll get more out of this problem if you try to do them on your own.

Hint for program 4: How much stack space is needed for the array? What does the command '`limit`' tell you about the limit on the stack size? What does that command do? (No idea what I've just said? Check Wikipedia[2]. This problem is more complicated[3] on OS X.)

Turn in a fixed version of each program with the same file name as in the source repository above, all in a subdirectory '`problem-2`' of your repository. Try to make sure that the output of

```
diff -u original/omp_bugN.c hpc12-hw2-netid123/problem-2/omp_bugN.c
```

makes sense, because that's what I'll be looking at. (Check this article[4] to learn how to read the output.) In particular, make as few changes as possible, and don't change the formatting of the program.

Problem 3: Build a tree in parallel

In this problem, we will parallelize the sorted binary tree build from homework 1.

a) Add timing around the tree building part of your code. (Don't time the output or the generation of the random numbers.) Use the timing code from this repository[5] to output the insertion rate as millions of insertions per second.

[1]`http://cims.nyu.edu/webapps/content/systems/resources/computeservers`
[2]`https://en.wikipedia.org/wiki/Stack-based_memory_allocation`
[3]`http://linuxtoosx.blogspot.ch/2010/10/stack-overflow-increasing-stack-limit.html`
[4]`https://en.wikipedia.org/wiki/Diff`
[5]`https://github.com/hpc12/lec1-demo`

b) Parallelize the tree insertion, initially without regard for the necessary synchronization. Verify the output of your program.

   Answer the following questions:

   1) What types of failures do you observe?
   2) Explain the failures you are seeing. Can you draw any conclusions about how your machine performs pointer updates?
   3) What speedup do you observe on this *incorrect* code?
   4) What efficiency?
   5) Are there any synchronizations hidden in your code? Perhaps in any of the C library functions you are using? Which ones?

   (Use timing data for 10 million entries. If you know how caches work, you may check what happens on smaller/larger cases for your own entertainment.)

c) Now write a version of your code that uses a critical section for synchronization. Verify that the output of your program is correct.

   1) What speedup do you observe on your *correct* code?
   2) What efficiency?
   3) Explain the performance, relative to the previous parts.

   (Again use timing data for 10 million entries.)

d) Now write a version of your code that uses per-entry locks for synchronization. Verify that the output of your program is correct.

   1) What speedup do you observe on your *correct* code?
   2) Explain the performance, relative to the previous parts.

   (Again use timing data for 10 million entries.)

e) Now write a version of your code that uses atomic operations for synchronization. Verify that the output of your program is correct.

   1) What speedup do you observe on your *correct* code?

   (Again use timing data for 10 million entries.)

In a subdirectory '`problem-3`', turn in the following:

- An updated `tree-sort-wrong.c` with no synchronization.
- A version of your code called `tree-sort-critical.c` with critical sections.
- A version of your code called `tree-sort-locks.c` with locks.
- A version of your code called `tree-sort-atomic.c` with atomic operations.
- A plain text file `answers.txt` where you provide answers to the prompts in parts b), c), d), e).