CS 4000

Homework # 5: CS Coin and Parallel Processing

due Friday April 16th, 2021 11:59 pm

(50 pts.)

Introduction

Bitcoin, and other digital currencies use a concept called "proof of work" to award digital money to those who perform computational work. The concept is somewhat straightforward, as we explain next. To demonstrate proof of work, we assume that we have a function f that is *one-way* in the sense that computing $y = f(x)$ is easy, given x, but computing the inverse, i.e., $x = f^{-1}(y)$ is hard to do. So, I could give you a number z and a one-way function f, and you could "prove that you did some work" by giving me the x such that $f(x) = z$. For such an x, it is easy to verify that $f(x) = z$ because you can simply compute $f(x)$ and compare it to z.

The function that bitcoin uses as f is the SHA256 cryptographic function. The output of SHA256 is always a 256-bit unsigned integer. For bitcoin, the *proof of work* for a number z is finding a number x such that $SHA256(x) < z$. This is, in principle, a bit easier than actually finding a number x such that $f(x) = z$, but the best known approaches are basically brute force (i.e., try all possibilities).

CS Coin

For this assignment, we have decided to create our own digital currency called "CS Coin." A CS coin miner can request a mining task from the World CS government. This mining task is a pair of large integers (256 bit) $\langle l, r \rangle$ where $l < r$. The miner's task is to find an integer z such that $l < SHA256(z) < r$. Once the miner discovers such a number z, they inform the World CS government and are awarded a CS Coin. CS coin's are currently valued at \$0.50 a piece, but their value is expected to skyrocket during the next global pandemic. (There is an internet rumor by CSanon, the brother of Qanon, that during the next pandemic that CS coin will be the only valid currency for buying toilet paper. So, people who frequent certain Facebook forums are stocking up on CS coin at an amazing rate.)

Base Code

I have implemented two sequential C++ programs: `cs_miner.cc` and `coin_verify.cc`, and I have provided a Makefile that compiles both programs into executables called `miner` and `verify`, respectively. Each program use the `openssl` and `boost` libraries. You may need to install these on your local machines if you want to run your code at home.

The program `miner` reads in two large integers l and r and produces an integer z such that $l < sha256(z) < r$. I have also provided a number of coin challenges (e.g., `coin_try14`). To run the program, enter

```
./miner <coin_try14 > coin14
```

This will redirect the input from the file `coint_try14` to the program and stores the result in a file `coin14`. The program `verify` can test whether you successfully found a *coin*. To test if your program runs correctly, enter

```
cat coin_try14 coin14 | ./verify
```

If you successfully found a coin, the verify program will print "true." Otherwise, it prints "false" and fails. The verify program is always fast. The miner is slow.

Part 1:OpenMP or C++ Threads (30 pts.)

For part 1, use OpenMP or C++ Threads to build a parallel version of the miner program. Your program should be race condition free and should achieve a speedup of at least 3 on a 4 core machine on sufficiently hard coin challenges.

Part 2:OpenMPI (20 pts.)

For part 2, use OpenMPI to build a parallel version of the miner program. Your program should be race condition free and it should achieve a speed up of at least 6 when run on a sufficiently hard coin challenge on 10 machines.

Extra Credit (10 pts.)

I will post a difficult CS coin challenge. Your task will be to mine a coin as fast a possible. To get the extra credit points, you will submit your solution to this coin challenge.