Fall 2023 ME/CS/ECE759 Final Project Report
University of Wisconsin-Madison

Bitcoin Miner

Jiayi Liu

December 13, 2023

Abstract

Mining is at the core of Bitcoin's feature and security. It is the process of validating transactions in a Bitcoin block by generating a cryptographic solution that matches specific criteria. This project aims to build a mining program in C++ and leverage CUDA for optimized parallel processing in an attempt to increase mining efficiency.

Link to Final Project `git` repo: https://git.doit.wisc.edu/JLIU694/repo759

Contents

1. General information 4
2. Problem statement 5
3. Solution description....5
4. Overview of results. Demonstration of your project....6
5. Deliverables. Building & running your project....6
6. Conclusions and Future Work 8
References....9

1. General information

In this short section, please provide only the following information, in bulleted form (six bullets) and in this order:

1. Name: Jiayi Liu
2. Email: jliu694@wisc.edu
3. Home department: Applied Mathematics
4. Status: Undergraduate
5. If applicable: no teammate
6. Choose one of the following two statements (there should be only one statement here):
    - I am not interested in releasing my code as open-source code.

IMPORTANT NOTE: For bullet 6 above, your choice does not affect in any way the score for your Final Project. It will only tell me that sharing your code in the future is ok.

Other points:

- It is ok to not have a fully finished Final Project as long as you demonstrate good progress towards completing the work spelled out in your Final Project Proposal document. Your `git` commit history plays an important role in this
- The Final Project points, on a scale from 1 to 100, will be allocated as follows:
    - The code, and what it can accomplish: 50%
    - The quality of the final report: 45%
    - Sticking with the letter of this template for the final report and making sure a link to the git repo is provided as instructed: 5%
- Please post follow up questions on Piazza

2. Problem statement

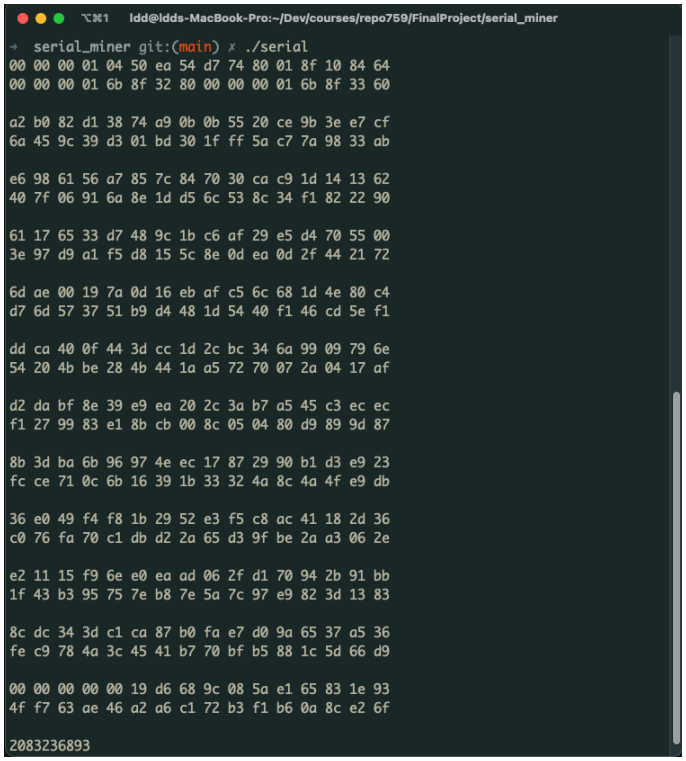
My initial goal was to write a Bitcoin miner from scratch in C++. I planned to build a serial miner that runs on the CPU as well as a CUDA-based miner that runs on the GPU. Then, I planned to compare the mining speed and tweak the CUDA parameters to increase the hash rate.

I have always been fascinated by the concept of a decentralized network, where information is permanent and uncontrollable by any single entity. I think Bitcoin is a good starting point in understanding such a network. This project will help me understand how a proof-of-work blockchain functions under the hood. It is also a good chance to apply what I've learned about CUDA and experiment with different optimization methods.

3. Solution description

I started with reading the Bitcoin white paper and watching a series of videos explaining Bitcoin in an attempt to further understand what "mining" really is.
After a brief understanding of what is involved in "mining", I set out to understand how SHA256 is implemented.
Then, with the help of several open-source projects, I built a serial miner that successfully verified the genesis block of bitcoin.
My attempt to build the miner based on CUDA failed but came pretty close.

4. Overview of results. Demonstration of your project



To the left is a screen shot of simulating the serial miner to verify Bitcoin's genesis block in 2009.

What's printed are 10 iterations of verifications, each one incrementing the value of nonce (number used once) with other information remaining unchanged (the blocks version, transactions, timestamps, previous block's hash value, etc.)

The miner considers a block verified once it finds a nonce that produces a hash value that has a certain amount of zeros at the beginning (i.e. meeting a certain "difficulty")

Finally, it prints out the nonce that produces the required hash.

5. Deliverables. Building & running your project

Discuss what is delivered for this Final Project. Important points:

- Talk about the structure of your code: what files you generated, where input files are (if any), etc.
- IMPORTANT: tell us how we should compile and run your code to get the results you report

- Code structure:
    - Serial miner:
        - sha256.c: here I took Brad Conte's standard implementations of SHA256, with all the functions and variables needed to hash a block header encapsulated in the file. I read several explanations of SHA256 and it's implementation and eventually understood the code and the process of generating a hash
        - util.c: in this file there are some utility functions that are 60% borrowed from Matt Beton's C++ serial implementation. I combined Brad's SHA256 with Matt's utility

functions (because Brad's SHA256 made more sense to me), made lots of tweaks and comments on the functions.

    - main.c: this is where I test the serial miner with bitcoin's genesis block. Here I simply ran the function `mineBlock` with a nonce starting at 2083236883. This will give the first hash that satisfies the genesis block's difficulty after exactly 10 hashes.

- CUDA miner:
    - Everything needed is in `main.c`. Since I was facing a lot of problems linking different files and running out of time, I combined everything into one file.
    - The tweak I made here is the function `mine_kernel`. Each thread computes one hash. All the threads share the same bitcoin block header (block version, previous block's hash, etc.), but the difference is the headers are combined with different nonces.
    - Each thread figures out its nonce based on its `threadID` and the starting nonce of the block.
    - The thread then performs a hash, and compares the hash value to the difficulty. If the hash is smaller than the difficult, it writes the current nonce to a global variable `correct_nonce` and writes the current hash to a global array `correct_hash`.
    - Since I ran out of time, I failed to implement a loop on the host that keeps calling the kernel if no correct nonce is found by the block of threads in the current kernel call. The loop will also require a global flag that stops the host from launching another block of threads if the correct nonce is found.

6. Conclusions and Future Work

I made a similar mistake in this project as most of my previous projects: I spent too much time up front trying to understand the theoretical parts without doing anything. The thing that helped me understand how bitcoin works is actually looking at existing code and trying to build a miner that makes sense to myself.
I think a mining program is a decently large project, and I did learn a lot from it. Although 50-60% of the code are borrowed from existing implementations, I have to understand everything to use the functions correctly to my advantage and build a program that is most intuitive to me.
In terms of time management, I failed again, this time worse than ME459. I failed to get the CUDA-based miner working correctly, not to mention timing the program and comparing the hash rate of the two implementations.

Future plans:
This is my last semester. My future plan is to realize some of my mobile app ideas. I have been putting them off because of school, and I'm excited to put in time where my passion is.
In terms of HPC, I'm selling my PC (with a 3070) right after this project (to quit gaming). Now I only have an M1 Pro MacBook. However, I have found some interesting articles online talking about how to utilize Metal (Apples GPU API) for scientific computing and graphics acceleration. I will learn more about that and hopefully apply it in one of my apps.

References

https://github.com/MattyAB/BitcoinMiner
https://github.com/B-Con/crypto-algorithms