MP2 Performance Study

Authors: Steve Huang, Asher Kang, Maria Ringes

I. Introduction

This program is an implementation of MP2, which demands a program that simulates and best imitates the real Bitcoin implementation. This program utilizes Go-channels and Go-routines to simulate user mining of a tamper-resistant log.

This report examines the effect of three variables on the time it takes for the first puzzle of the blockchain to be solved by a miner. These variables are the difficulty of the puzzle, the GOMAXPROCS (GMP) settings of the system, and the number of miners in the system. The last variable is not specified by the professor in the MP2 instructions, but was added to the experimentation because the authors were curious about its effect.

II. Methodology

A. Research

Our first step in research was understanding how blockchain technologies function. It was important for us to understand the structure of a blockchain in order to best and most accurately model a tamper resistant log. We researched, using videos, articles, and GitHub repositories, what each block within the blockchain should look like as well as which elements of the actual Bitcoin implementation were necessary to meet the given guidelines. This research allowed us to design our own block struct which would consist of a transaction string (as a Merkle Tree was not necessary) and our own BlockHeader struct. This BlockHeader struct would contain the version, the hash pointer of the previous block, the merkle root hash filler, the time, the bits (or difficulty value) as well as the nonce of the block.

Our second step in research was understanding how Bitcoin utilizes blockchain to function. From class lectures, as well as videos and other resources, we understood that there would exist one tamper-resistant log. Miners would then each try to solve a designated 'puzzle,' which consisted of guessing a nonce value of type *int* that, when combined with the field values of the last approved block on the blockchain, would yield a new hash value that is less than the target value, or the 'difficulty' level. Once a miner had solved this puzzle, a block would be created, verified, and appended to the blockchain. Subsequently, a new puzzle would be created and distributed for miners to work on in order to mine the next block on the blockchain. This research was the foundation for our structs Logger, Miner, and Message.

B. Data Collection

Difficulty vs. Puzzle Time

The tuning of difficulty is achieved by adjusting the difficulty array so that $difficulty[i] = j$, where a difficulty value of n will result in $difficulty[n/8] = n\%8$ if n is not a multiple of 8 and $difficulty[n/8 - 1] = 1$ otherwise. To collect data for each difficulty, the program is manually stopped once the first few puzzles are solved. The time taken for each solution is

recorded and put on the graph. Then the difficulty increases by 1 and the process repeats until no solution is given for five minutes under one difficulty.

GOMAXPROCS vs. Puzzle Time
For each GMP value, run the program and manually stop it once at least ten puzzles are solved. The first ten solving times are recorded and put on the graph. The process is repeated for all possible GMP values on this machine.

Number of Miners vs. Puzzle Time
To complete this data collection, the TimeLimit of the program was set to 10 minutes such that miners would continue to mine and add blocks to the tamper resistant log until 10 minutes had passed. Each time a miner had found a new block, the time elapsed since the last block had been verified would be added to a variable total time. Additionally, the program would keep track of how many blocks were verified and added to the log. The program, after 10 minutes, would output the average time it took for each block to be mined. The difficulty level stayed consistent throughout these trials. Results were then exported into a Google Sheets document[1] and graphed accordingly.

C. Design Decisions

As an inherent feature, our program prints out the time it took to solve the current puzzle and add the new block. We use this built-in feature to collect our Puzzle Time values for our experiment.

Difficulty vs. Puzzle Time
To measure difficulty vs. puzzle time, we chose to run the program from difficulty = 1 to difficulty = 30 (where a difficulty of n implies that the size of the target set is exactly 2^{256-n}). This range was chosen because it was the closest fit to the 5 minute limit established by the Professor in the original MP2 guidelines. We chose to use a constant value of 5 miners for each run of the experiment, due to the relative ease of generating and running 5 miners in terms of time.

GOMAXPROCS vs. Puzzle Time
To measure GOMAXPROCS vs. puzzle time, we chose to run the program with difficulty = 2 and 5 miners. Both of these values were chosen due to the relative ease of computation in terms of time.

---

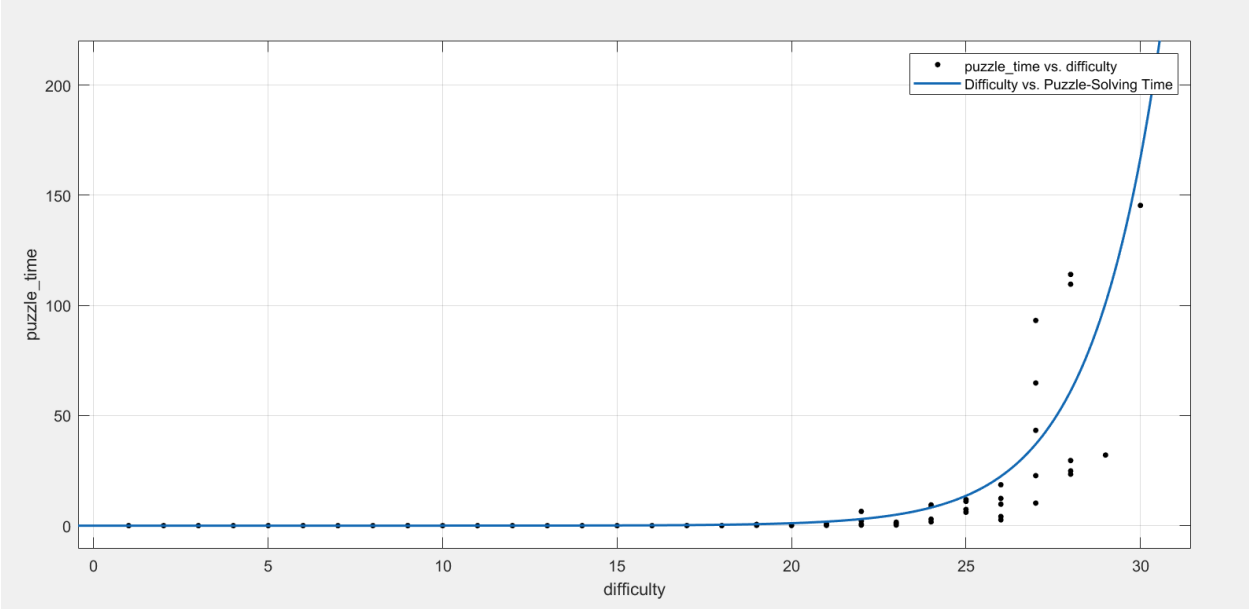
[1] https://docs.google.com/spreadsheets/d/1E_Wm5R7iB743qcsS-SAVg8p5YxdU9J9giB1OqS4Y_jQ/edit?usp=sharing

Number of Miners vs. Puzzle Time

To measure the number of miners vs. puzzle time, we chose to run the program with difficulty = 2 and default GOMAXPROCS value of 8. Both of these values were, again, chosen due to the relative ease of computation in terms of time.

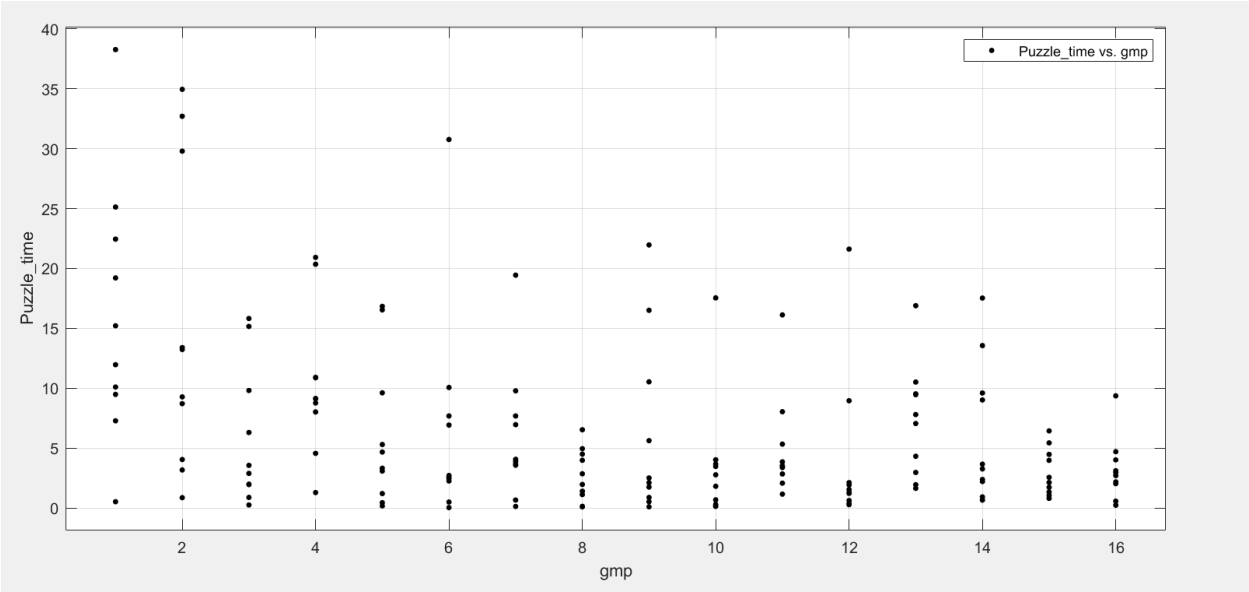
III. Results

Difficulty vs. Puzzle Time



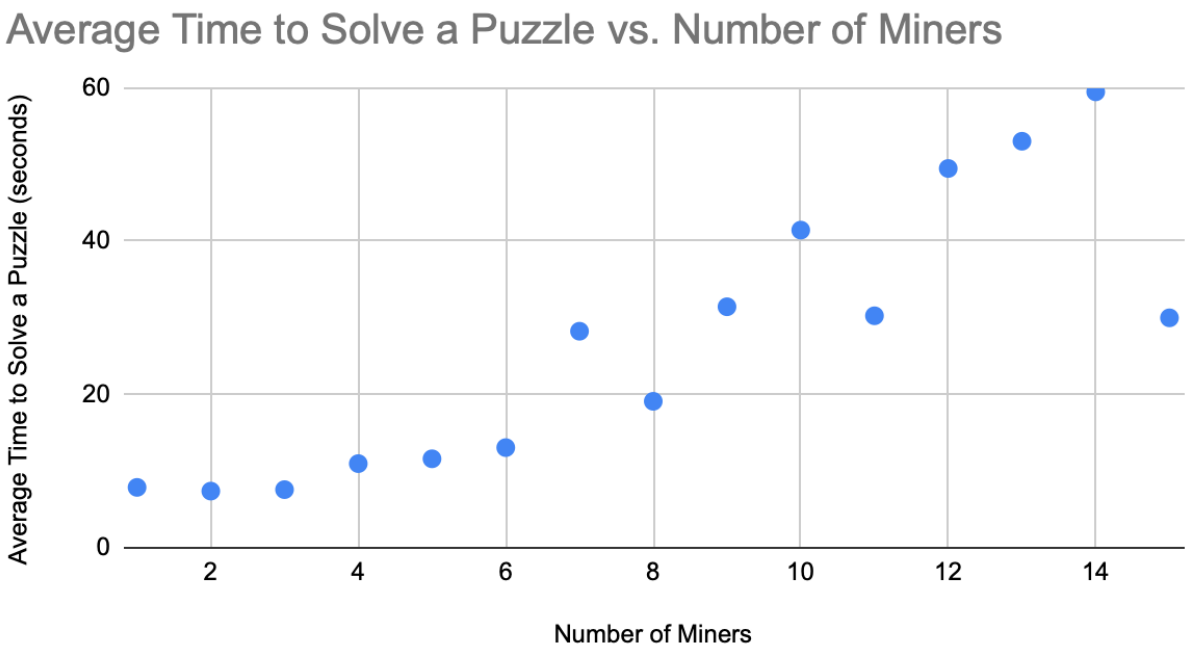
The line of best fit has the following equation: $f(x) = 4.78 \times 10^{-5} e^{0.5022x}$. The exponential growth of puzzle-solving time matches the exponential shrinking of the target set's size.

GOMAXPROCS vs. Puzzle Time



Even with the same difficulty, GMP, and number of miners, the puzzle-solving time can still vary drastically due to the fact that the first occurrence of puzzle solution varies for different puzzles. As a result, the data points are largely spread out in the graph above and no line of best fit can be obtained to truly represent the data. However, it is still observable that, at large GMP values, data points are more clustered toward the lower end, suggesting that a higher GMP value likely reduces puzzle-solving time overall.

Number of Miners vs. Puzzle Time



From this data, we can conclude that when run locally, as the number of miners increases, so does the average time to solve a puzzle. This is because, when run locally, the computation power of one machine begins to be split up among more entities, or miners, thus causing less computation power per miner. Ultimately, this decrease in computational power will lead to a delay in the time needed to solve a puzzle.

In the true implementation of Bitcoin, where each miner node has its own set computational power, the average time to solve a puzzle should not change with just the changing of the number of miners.

IV. Discussion

Our data analysis suggests that as the difficulty of the puzzle increases, the time it takes to solve the first puzzle also increases. This comes as no surprise, as the very definition of an increase in difficulty of a puzzle connotes that it would take more effort and more time for miners to solve it.

Our data also suggests that as GOMAXPROCS value settings increase, the Puzzle Time decreases. Intuitively, this might be because more processing power is available for use for the miners, and so it would take less time for the first puzzle to be solved.

Finally, our data suggest that as the number of miners in the system increases, the Puzzle Time also increases. On the surface, this result seems surprising, as one would expect that having more miners working on the puzzle would decrease the time it takes for the puzzle to be solved. More miners means more resources working towards cracking the puzzle, after all. However, our findings may be based on the fact that the computational power of the computer that runs these experiments is limited--a factor that is unique to this experiment, and not translatable to the real world. Limited computational power combined with an increasing number of miners means that each miner has an increasingly smaller and smaller share of the total computational power. As such, this partitioning of the total computational power may be the cause of the increase in puzzle solving time.