MP2 Performance Study
Matt, Liam, Roland

Introduction/Background Information

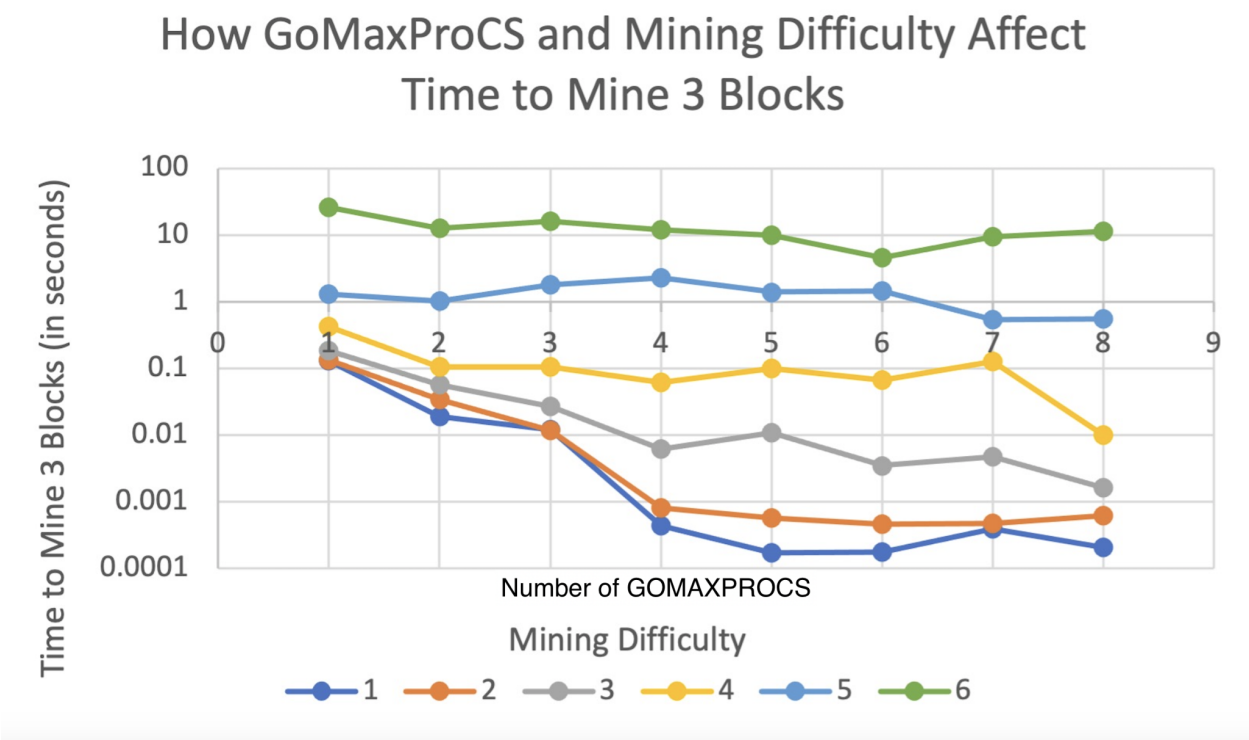
Bitcoin mining is an important process through which new Bitcoin are added to circulation and new transactions are marked as valid on the blockchain. In order to "mine", a node must solve a computationally intensive math problem which requires a lot of processing power to solve. This solution is hard to find but is very easy to validate by other nodes. We have created a simulation of this mining process.

Methodology:

The program was written to use Go-channels and Go-routines to simulate mining and a tamper-resistant log. The program stops after 5 blocks have been mined but for our trials we stopped it after 3 for timing purposes. Our mining puzzles were completed in a similar manner to bitcoin, where a hash of the transaction data, the name of the new block, and a nonce value were concatenated and hashed with sha256. This hash was then checked to confirm it has at least the number of leading zeros specified by the difficulty value. This was done by turning the hash value (in hex form) into a string, creating a string of zeros equal to the number specified by the difficulty, and checking if the hash contained the string of zeros as a prefix (the first characters in the string).

Packages Used:

crypto/sha256: used for hashing transactions and blocks
encoding/hex: used for turning sha256 results into strings
fmt: used for printing in terminal
runtime: used for modifying GOMAXPROCS
strconv: used for converting ints to strings for hashing
strings: used for building prefix of zeros and checking against string version of sha256 value
sync: used for generating waitgroups, which simplifies concurrency
time: used for timing the creation of new blocks



Discussion:

The data graphed above supports the hypothesis that the more GOMAXPROCS used the less time it takes to mine a certain number of blocks. Additionally, lowering the mining difficulty (or number of leading 0s required for the mining hash) also lowers the time it takes. Both of these observations make complete sense because with more threads running our program there is extra computational power to solve the mining puzzles. Also, it makes sense requiring less leading 0s for the mining hash means it can be done faster because there are more possibilities for a valid number resulting from the hash. For example, if a valid mining hash can be any number from 0-1,000,000 and the only requirement is 1 leading 0, then the number of valid numbers is 0-999,999. However if the mining difficulty is 5 meaning there needs to be 5 leading 0s, the number of valid numbers drops to 0-9,999 which is exponentially less options. This also helps explain why the difficulty affects the time in an exponential way rather than a linear way (note the log scale on the y-axis in order to fit the data). Changing the difficulty from 1 to 2 means there are exponentially less options for the result hash so it makes sense that this is reflected in the data.