Performance Benchmarking For Ethereum Opcodes

Amjad Aldweesh, Maher Alharby, Aad van Moorsel
School of Computing, Newcastle University
Newcastle upon Tyne, UK
Email: {a.y.a.aldweesh2, m.w.r.alharby2, aad.vanmoorsel}@ncl.ac.uk

***Abstract*—Ethereum is a public (permissionless) blockchain with a Turing complete execution machine for smart contracts. Miners that execute a smart contract receive a fee determined by the gas associated with the operation codes (opcodes) in the smart contract. It is important that the gas award is proportional to the computation resources required, to assure that incentives are aligned and denial of service attacks are avoided. Currently, the amount of gas awarded is set statically for each opcode, but it is unknown if these values are correct for various computer architectures. Therefore, we propose in this paper a benchmark approach to assess the computational resources required per opcode. We apply the benchmark approach to PC and MAC as a first illustration of the approach.**

***Index Terms*—Blockchain, Ethereum, Smart Contract, Benchmarking.**

I. Introduction

Bitcoin and its underlying technology blockchain have gained huge interest from industry and academia. Ethereum is a permissionless crypto-currency blockchain that distinguishes itself by a Turing-complete execution machine, the Ethereum Virtual Machine (EVM) [2], which executes distributed applications through smart contracts. Miners that execute smart contracts charge a fee to the submitter, as compensation for them contributing their computing resources. Gas costs are set per operational code (opcode) by the Ethereum foundation, accounting for computation overheads, storage, and network costs [3].

Inappropriate gas costs for opcodes might enable DoS attacks, if the cost is low but computational effort is high. The Ethereum foundation has had to adjust the cost for certain operational codes in response to DoS attacks, see, e.g., [4]. In addition, if the gas cost is not proportional to the computational effort, miners are not awarded fairly, thus potentially disincentivising miners to operate the blockchain in the best manner [6].

As mentioned, gas cost per opcode is currently set statically in [3], and it would therefore be useful to assess whether the set values are correct for different hardware architectures. Therefore, we propose a benchmark approach to investigate whether the gas costs of EVM opcodes are set properly or not. We run the benchmark for PC and MAC, and show that gas costs are not always proportional to computational effort.

II. Background

Ethereum blockchain is similar to Bitcoin's, in that it operates as a distributed ledger, in which all transactions history is recorded and stored in every node in the network. Additionally, in Ethereum every node also stores the most recent state of smart contracts. The native currency of Ethereum is Ether. In addition to transactions transfering Ether, Ethereum has two more transactions to support smart contracts: *contract-creation* and *contract-invoking*.

Ethereum has two types of accounts *externally owned accounts(e.g., users)* and *contract accounts*. The former is controlled by private keys to create and sign transactions, whereas the latter is controlled by their associated smart contract code. The associated code for the contract accounts is stored in the blockchain in a bytecode form and interpreted by the EVM. In the Ethereum blockchain, miners have a pool to maintain pending transactions(transactions that have not been executed yet). Each time a miner wants to create and append a new block to the blockchain, they select a number of transactions from their pool to be executed. If the transaction invokes an existing contract, miners execute the code associated with that contract using their local EVM. Upon successful execution, the state of the smart contact variables, storage and balance will be adjusted accordingly.

Gas is an abstraction that has been introduced in the Ethereum network to reward miners for their computation efforts invested to execute transactions. The submitter of a transaction pays miners a fee according to the amount of Gas used by the miner, which depends on the complexity of the transaction. In particular, each operation code in the transaction has associated a pre-defined amount of gas. Gas introduces a manner to mitigate DoS attacks, since attackers would have to pay the fee associated with a transaction that takes a long time (or run forever) to be executed. More in general, the correct operation of Ethereum relies on the fact that miners are awarded proportional to their investment of computational resources. It is therefore essential to have ways to assess and ensure that the gas cost associated with a transaction is proportional to the cost of executing that transaction [6]. For a complete solution, one wants to consider the actual cost of executing a smart contract to a miner, is dominated by energy usage. In this paper, we focus on CPU usage, as a first approximation to miner costs.

III. Opcode Benchmark Design

EVM is a stack-based virtual machine where each operation code is pushed onto the stack and executed. The Ethereum foundation [3] has proposed a gas cost for each EVM opcode (e.g., opcode for additiona costs 3 units of gas) depending on the computation overhead, storage and network

978-1-5386-9120-5/18/$31.00 ©2018 IEEE

costs required by the opcode.The storage and network costs have been determined based on some formulas, while the computation overhead has been determined based on the CPU time consumed to execute the opcode [2].

Extended to [7], to investigate the computational overhead of opcodes we analyzed in detail the internals of the EVM, to understand how opcodes are interpreted and how computational effort can be measured. In EVM's stack-based virtual machine, all opcodes are carried out by pushing and popping the instructions and the results into or from the stack. Since separate gas is awarded for PUSH and POP, we need to isolate the time used for the actual opcodes from that for PUSH and POP. For example, to test the LT (Less Than) opcode, two PUSH opcodes as inputs and one POP opcode as outputs are required. In our benchmark approach we set timers per opcode, and also for overall sets of opcodes, so that both the impact of PUSH and POP as well as the overhead associated with setting timers are accounted for.

The experiment has two phases: *Generating bytecode* and *Running the bytecode on PyEthApp client*. The first phase generates bytecode that contains the opcodes that need to be executed. The bytecode contains all required inputs and is in a ready form to go to the blockchain. The second phase creates two transactions, one to deploy the bytecode in the blockchain and the other to invoke the bytecode. All overhead time of creating transactions, pushing and popping opcodes are isolated from the result. Due to the stack size limit [2], each bytecode has 1024 opcodes copies to be tested.

IV. Experimental Results

EVM opcodes are classified into eleven categories (e.g., Stop and Arithmetic Operations, Comparison & Bitwise Logic Operations, Push operations, etc.). Here we report on initial experiments, for which we selected one category (Comparison & Bitwise Logic Operations) to be benchmarked. This category has eleven opcodes (LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, NOT and BYTE). The gas cost set by [3] for all these opcodes is 3 units of gas. Currently, eight different clients exist for EVM (e.g., Go-Ethereum and PyEthApp). We use the PyEthApp client to conduct the experiment. The experiment was conducted on two different machines, a MacBook Pro with a 2.8 GHz Intel i5 CPU and 8 GB RAM and a Desktop equipped with a 3.20 GHz Intel i7-4790s CPU and 8GB RAM. We execute 1000 runs for each opcode and calculate the average time as well as a confidence interval.

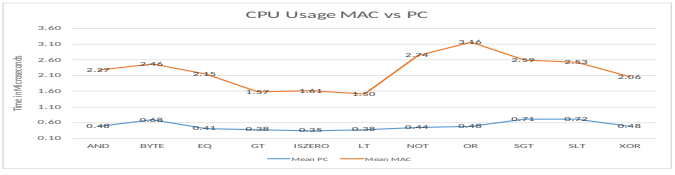

Fig. 1. CPU Usage MAC vs PC

Figure 1 presents a comparison of CPU usage for each opcode between the MAC and the PC. The x-axis gives the opcodes and y-axis the execution time in microsecond. As can be seen, the PC is faster than the MAC for executing all opcodes. This is due to the high hardware specification of the PC. The result of the PC shows that all opcodes take less than 1 microsecond, whereas it is between 1.5 to 3.16 microseconds on the MAC.

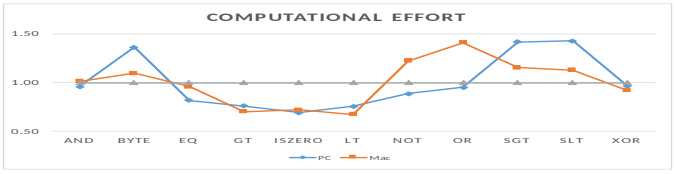

Fig. 2. Normalized CPU Usage per Unit of Gas, for MAC & PC.

Figure 2 shows the CPU used per unit of gas for each of the opcodes, where we normalized the results using the mean CPU time over all opcodes for PC and MAC, respectively. So, the results are scaled so they average to 1, as indicated by the gray line. If, for some opcode and platform, the result is above 1, it implies that this opcode is relatively expensive in terms of CPU use per unit of gas. If it is below 1, it is relatively cheap. From the Figure we see that for the PC and MAC, the most profitable opcodes are similar (GT, ISZERO and LT). However, the MAC is more costly for the NOT and OR opcodes, while the PC is costly for BYTE, SGT and SLT. From these early experiments we see that for different platforms (PC versus MAC) different opcodes are most cost-effective for miners.

V. Conclusion and Future Work

This paper presents a benchmark approach to assess the CPU usage required per EVM opcode and compare it with the set gas cost. We conducted a first experiment, comparing CPU consumption of EVM opcodes with their associated gas costs on PC and MAC. This provides us with some interesting insights about reward for invested CPU resources, with respect to opcodes and computer platform. In the future, we will extend our benchmark approach for CPU usage to cover all the EVM opcodes. Moreover, we aim to consider computing effort beyond CPU usage (e.g. storage), and relate computing resources more strongly to actual energy costs.

References

[1] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[2] Wood, G. (2014). Ethereum: *A secure decentralised generalised transaction ledger*. Ethereum Project Yellow Paper, 151, 1-32.

[3] Ethereum Foundation. (2013). *EVM Operation Codes Gas Cost*. `https://bit.ly/2nEwrEv`

[4] Atzei, N. and Bartoletti, M. and Cimoli, T. (2017). *A survey of attacks on Ethereum smart contracts (SoK)*. International Conference on Principles of Security and Trust (pp. 164-186). Springer.

[5] Ethereum foundation. (2013). *Pyethereum client*. `https://bit.ly/2siwj2g`

[6] Alharby, M. and Van Moorsel, A. (2018). *The Impact of Profit Uncertainty on Miner Decisions in Blockchain Systems*. In UK Performance Engineering Workshop.

[7] Aldweesh, A. and Alharby, M. and Van Moorsel, A. (2018). *Performance Benchmarking of Smart Contracts to Assess Miner Incentives in Ethereum*. In First International Workshop on Blockchain Dependability.