Appendix N: Understanding Gas

Return to Reference Architecture (RA) or Return to Appendices

Overview

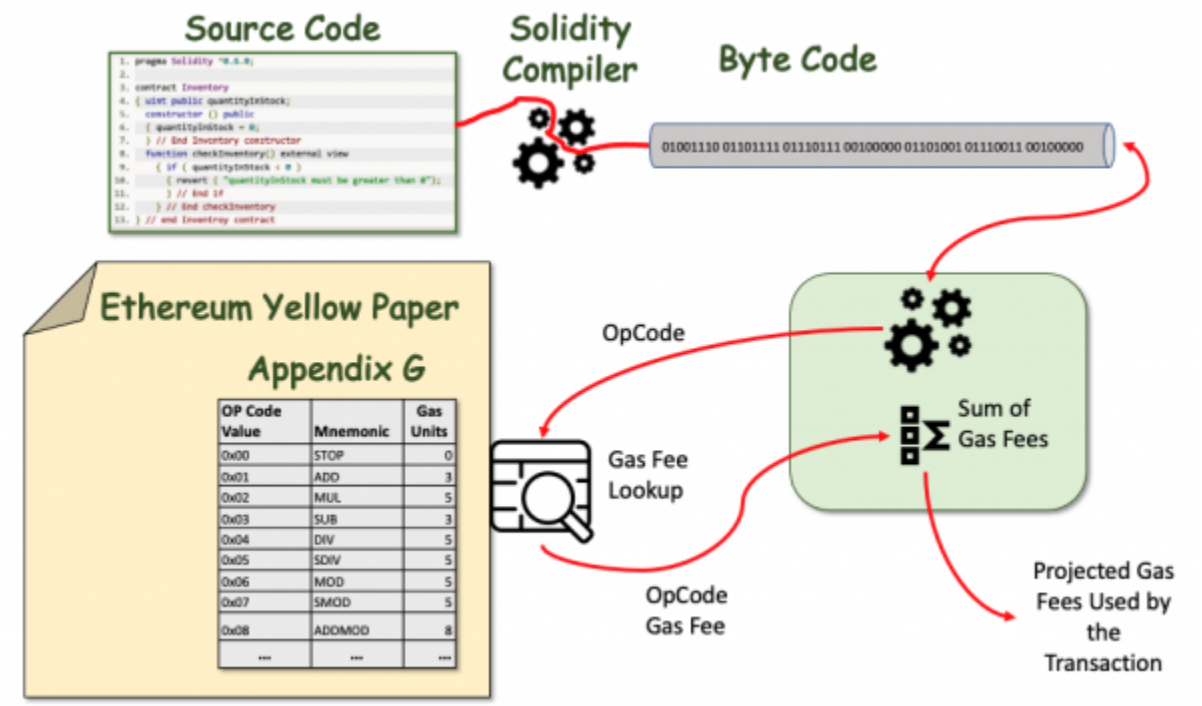
Return to Top

Some of the most important aspects of a DIDOs are:

- The data is immutable and every update takes more storage
- They are distributed and all the nodes in the system must ultimately get the same results from calculations
- The data is distributed across all the nodes and must be kept in sync across all the nodes
- There needs to be an assurance of data correctness
- The Data needs to be secure during Data-at-Rest, Data-in-Motion, and Data-in-Use
- **There is a cost to using the distributed resources**

Most of these can be achieved as part of a DIDO Platform such as Ethereum. In this section, the focus is on the last item, ***"There is a cost to using the distributed resources</color>*. This requires a method for remuneration for all the work required to make the DIDO function. In Ethereum and in many other DIDO Platforms, this is achieved using the concept of Gas. The purpose of gas is to provide a "value layer" providing a way to recapture the computational costs of running and maintaining the DIDO Platform and its Node Network (i.e., Ethereum Platform and Ethereum Network). Unfortunately, the analogy used to describe how such a system would work was also used as the name of the value only adding to the confusion and leading to ambiguity. The unfortunate analogy used is between the cost of running a gasoline car over a certain distance and the cost of running a transaction over a certain network. Here is the example often quoted: : *running a real-world car for X miles may require Y gallons of fuel, or moving X amount of money from your bank account to your friend's credit card account may cost you Y dollars in a processing fee. In both cases, X indicates the utility value, while Y indicates the cost of performing the process of the car trip or financial transaction.* [1] Although the car analogy is useful in explaining how simplistically the "value layer" operates, using the exact same terminology leads to inconsistencies and constraints in moving forward. Regardless of how we got here, or how it could have been done differently, we are where we are and for better or worse, for now, we must use the terms given to us by Ethereum. Hopefully, future "value layer" propositions won't make the same mistake. that solely indicates the consumption toward computational expenses on the Ethereum network. Having a separate unit for this purpose allows for a practical distinction between the actual valuation of the cryptocurrency (ETH), and the computational cost of using Ethereum's virtual machine (EVM). Here, gas refers to Ethereum network transaction fees, not the gasoline for your car. Gas fees are payments made by users to compensate for the computing energy required to process and validate transactions on the Ethereum blockchain. "Gas limit" refers to the maximum amount of gas (or energy) that you're willing to spend on a particular**

transaction. A higher gas limit means that you must do more work to execute a transaction using ETH or a smart contract. To draw an analogy, running a real-world car for X miles may require Y gallons of fuel, or moving X amount of money from your bank account to your friend's credit card account may cost you Y dollars in a processing fee. In both cases, X indicates the utility value, while Y indicates the cost of performing the process of the car trip or financial transaction. Similarly, a contract or transaction on Ethereum may be worth 50 ETH (X), and the gas price to process this transaction at that particular time might be, say, 1/100,000 ETH (Y). Ethereum miners, who perform all the important tasks of verifying and processing transactions on the network, are awarded this particular fee in return for their computational services. If the gas price limit is too low, miners can choose to ignore such transactions. As such, the price of gas fluctuates (priced in ETH) with supply and demand for processing power. ===== Gas Units ===== On Ethereum, each low-level operation available in the Ethereum Virtual Machine (EVM) is called an Operational Code (OPCODE). There are OPCODES for the following categories[2]: * Stop and Arithmetic Operations * Comparison and Bitwise Logic Operations * Accessing Environmental Information, * Accessing Block Information * Accessing Stack, Memory, Storage, and Flow Operations * Executing Push Operations * Performing Duplication Operations * Conducting Exchange Operations * Performing Logging Operations * Executing System Operations A spreadsheet is available documenting the Ethereum OPCODDES and the cost of each OPCODE: See EVM-OPCODE-GAS-COSTS. Each OPCODE (i.e, ADD) within each category has an associated cost that is expressed in terms of Gas units. Gas is an abstract number that represents the relative complexity of operations. For example, the integer ADD (i.e., addition) operation uses 3 gas units while the Integer MUL (i.e.m multiplication) uses 5 gas units. Therefore the MUL OPCODE is more complex than ADD OPCODE. ===== Ethereum Virtual Machine (EVM) ===== Return to Top The Ethereum Virtual Machine (EVM) is the runtime environment used by the Ethereum platform. There are several reasons why Ethereum uses a Virtual Machine (VM) for its runtime on every node within the Node Network. - Platform Independence - Hardware - Operating System - Runtime System - Security - Correctness It allows smart contract code to run by compiling to EVM bytecode. EVM plays a core role in the Ethereum blockchain to ensure a trustless mechanism without having any central administrator. EVM keeps each node systemically isolated from others in order to avoid security risks. Even if one node is compromised, it does not influence any other nodes and the blockchain network. EVM ensures the Ethereum blockchain is secured. ===== Gas Units ===== Return to Top The Smart Contract written in Solidity is compiled into a stream of EVM Bytcodes. EVM Bytecode The Ethereum Yellow Paper Appendix G, provides a cost for processing each Gas Units are the calculated cost of every operation the EVM can perform. <figure OpCodeCalc>



<caption>Schematic of how the Gas Fees for a Transaction are calculated.</caption> </figure> ===== Gas Price ===== Return to Top The number of Gas units each operation consumes remains constant. However, the cost of the gas is dynamic and dictated by market conditions and reflects the amount each user is willing to pay to get their operations executed (i.e., the gas price expressed in Gwei which equals 0.000000001 ETH). The Gas value is attached to the user's transaction when the Transaction is submitted. Note, that as the price of an ETH varies, so will the gas price, but the amount of gas remains constant. In other words, the gas is associated with the relative cost of the operation and the gas price reflects how valuable the end-user thinks their operation is. The actual cost to the end-user is market-driven. The total fee the end-user pays is equal to **`gas_price * gas_used`**. Miners are paid from the transaction fee and consequently, the miners to maximize profits prioritize transactions with a higher gas price. So, in essence, the higher the gas price you are willing to pay, the faster your transaction will be processed. The following is a summary of an excellent article by Danny Ryan [3]. : *A miner executes the computation associated with each transaction being included in a block, resulting in an updated state. Upon successfully mining a block, a miner broadcasts the block to the network. Each of the other miners and non-mining nodes verifies the validity of the transactional computation and resulting state change before accepting the block as valid, incorporating the block into their copy of the blockchain, and moving on to the next block.*[4] ===== Processing Costs ===== Return to Top The sample data shown in the first part of Table ##REF:gasCosts## rows 1-2, demonstrates the cost of Gas for performing simple **add** operations on the Ethereum blockchain. The first row in the table shows the cost of performing a single addition on the Ethereum Blockchain. Note: the same would occur for a single subtraction. : **Line 1:** The cost to perform a simple addition once is about \$0.000002655 U.S. Dollars. This seems pretty inexpensive until the problem is expanded to do the same calculation 1 Million times. : **Line 2:** The cost to perform a simple addition one million times is about \$26.550000000 U.S. Dollars. This becomes significant and should be of concern for most projects. As a comparison, using the Amazon Web Services (AWS) cloud and a Python program, adding the two numbers together one Million times takes about 0.04 seconds at a CPU cost of \$0.0059/hour making the calculations \$0.000001639/second or \$0.000000066 for the entire operation on AWS. The results are stunning: \$0.000000066 on AWS versus \$26.55 on Ethereum. The Ethereum calculation is about 400 Million times more expensive (or 40 Million if you are willing to pay a low gas price). ===== Data Storage ===== Return to Top The sample data shown in

the first part of Table ##REF:gasCosts## rows 3-5, demonstrates the cost of Gas for storing data on the blockchain. This could be a simple, single value such as the number of days until a Smart COntract expires (i.e., simple integer) or it could be something more ambitious such as a short story, a music file, or a video. Storing data on the blockchain is extremely expensive. The reason for the cost is the storage of data on a blockchain is immutable and is replicated across tens of thousands of Ethereum Nodes in the Node Network. Therefore, uploading a short story, a music file, or a video onto the blockchain is cost-prohibitive: : **Line 3:** The cost of storing 256-bit word is about \$0.171 U.S. Dollars. requires about 20,000 gas. This is about 6,000 times more expensive than adding two numbers together (see **Line 1**) : **Line 4:** The cost of storing 1MB is about \$18.75 : **Line 5:** The cost of storing 1GB is about \$18750.00 Another bottleneck in storing large amounts of data is the current Block Gas Limit of approximately 4700000 gas/block. At this cap of gas per block, it would take over 132 blocks to write 1 MB of data to the blockchain, and that is assuming you can manage to hog all of the gas per block and that there are no other operations required! ===== Analysis of Cost Data ===== Return to Top <table gasCosts> <caption>The gas costs on Ethereum for calculation and storage on the blockchain.[5] </caption> Given: Median Gas Price on **23 August 2017** * 1 Gwei equals 0.000000001 ETH) * median gas price (28 Gwei) * USD/ETH exchange rate ($295/ETH) |< 100% 20% >| ^ Processing ^^^^^^^^^^ ^ Row Number * Task ^ Gas required ^ Cost (ETH) ^ Cost (USD) ^ Ops per ETH ^ Ops per USD ^ Ops per Block ^ Blocks to complete OP | ^ 1 ^ Add or subtract two integers | 3 | 0.000000009 | 0.000002655 | 11111111.11 | 37664.78343 | 1566666.667 | 0.0000006382978230 | ^ 2 ^ Add two Integers, 1 Million times | 3000000 | 0.09 | 26.550000000 | 11.1111111 | 0.037664783 | 1.566666667 | 0.638297872 | ^ Storage ^^^^^^^^^^ ^ Task ^ Gas required ^ Cost (ETH) ^ Cost (USD) ^ Ops per ETH ^ Ops per USD ^ Ops per Block ^ Blocks to complete OP | ^ 3 ^ Save a 256-bit word to Storage | 20000 | 0.0006 | 0.171 | 1666.666667 | 5.649711751 | 243 | 0.004255319 | ^ 4 ^ Save 1MB to Storage (31250 256-bit words) | 625000000 | 18.75 | 5531.25 | 0.053333333 | 0.000180791 | 0.00752 | 132.9787234 | ^ 5 ^ Save 1GB to Storage (31250 256-bit words) | 6250000000000 | 18750.00 | 5531250 | 0.00005333333 | 0.00000018079 | 0.00000752 | 132978.7234 | ^ **Note:** The data mentioned above are only true for a public Ethereum blockchain. The data could be totally different for private or permissioned blockchains. ^^^^^^^^^^ </table> ~~DISCUSSION:on|Outstanding Issues~~ ~~DISCUSSION:off~~

[1)]
Jake Frankenfield, Investopedia, Gas (Ethereum), 20 May 2022, Accessed 2 June 2022
https://www.investopedia.com/terms/g/gas-ethereum.asp
[2)]
Dr. Gavin Wood, ETHEREUM: A Secure Decentralized Gernerised Transaction Ledger, H.2 Instruction Set, BERLIN VERSION b2d0dbf Page: 30, 6 May 2022, Accessed: 11 May 2022,
https://ethereum.github.io/yellowpaper/paper.pdf
[3)], [4)], [5)]
Danny Ryan, Hackernoon, , 29 May 2017, Accessed: 11 May 2022,
https://hackernoon.com/ether-purchase-power-df40a38c5a2f

From:
https://omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:n_gas&rev=1654190313

Last update: **2022/06/02 13:18**