ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 7| Issue: 9| September 2021|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2021: 8.047 || ISI Value: 1.188

A REVIEW ON GAS CONSUMPTION AND GAS PREDICTION IN ETHEREUM SMART CONTRACTS

Kuldeep Nageshawar[1], Rupali chourey[2], Dr. Ritu Shrivastava[3]

[1]*Research Scholar, Department of Computer Science and Engineering, Sagar Institute of Research & Technology, Bhopal, India*

[2]*Asst. Professor, Department of Computer Science and Engineering, Sagar Institute of Research & Technology, Bhopal, India*

[3]*Dean & HOD, Department of Computer Science and Engineering, Sagar Institute of Research & Technology, Bhopal, India*

Article DOI: https://doi.org/10.36713/epra8349
DOI No: 10.36713/epra8349

ABSTRACT

This paper is a review of some real-time issues associated with the development of Ethereum smart contracts like out of gas exception and gas inefficient code patterns and focuses on the methods and solutions presented in recent years. With the help of this paper, we are trying to summarize the methods which can be used in future researches for gas prediction with the help of old transaction data and machine learning algorithms and have a look at old researches which are trying to predict with the help of regression algorithms and their efficiency.

KEYWORDS: ***Blockchain, Etherium, Gas Consumption, Gas Prediction, Transaction***

I. INTRODUCTION

Ethereum is a blockchain-based software platform that is primarily used to support the world's second-largest cryptocurrency by market capitalization after Bitcoin. Value exchange is the main use case of the Ethereum blockchain today, often via the blockchain's native token, ether. Ethereum uses smart contracts to leverage the benefits of blockchain. Unlike other blockchains, Ethereum is programmable using a Turing complete language[1], i.e., developers can code smart contracts that control digital value, run exactly as programmed, and are immutable. Every single operation in smart contracts requires some amount of gas to execute in an Ethereum virtual machine.

II. BACKGROUND

Blockchain

Blockchain is a tamper evident and tamper resistance digital ledger implemented in a distributed fashion and usually without a central authority. It enables a community of users to record transactions in a shared ledger within that community.

In 2008, the Blockchain idea was combined with several other technologies and computing concepts to create modern cryptocurrencies: electronic cash protected through cryptographic mechanisms instead of a central repository or authority. It allowed users to securely transfer crypto-currencies, known as "Bitcoins" without a centralized regulator. Besides, Ethereum, NXT, and Hyperledger Fabric were also proposed as blockchain-based systems used for the cryptocurrency [2].In Bitcoin the transfer of digital money takes place in a distributed system. Bitcoin users can digitally sign and transfer their rights to that information to another user and the Bit-coin blockchain records this transfer publicly, allowing all participants of the network to independently verify the validity of the transactions. Trust in the Bitcoin network is maintained by the 4 essential properties.

- Ledger – the technology uses an append-only ledger to provide full transactional history. Unlike traditional databases, transactions and values in a blockchain are not overridden.
- Secure – blockchains are cryptographically secure, ensuring that the data contained within

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 7| Issue: 9| September 2021|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2021: 8.047 || ISI Value: 1.188

the ledger has not been tampered with and that the data within the ledger is attestable.

- Shared – the ledger is shared amongst multiple participants. This provides transparency across the node participants in the blockchain network. [3]
- Distributed – the blockchain can be distributed. This allows for scaling the number of nodes of a blockchain network to make it more resilient to attacks by bad actors. By increasing the number of nodes, the ability for a bad actor to impact the consensus protocol used by the blockchain is reduced

Block-chain types are categorized based on who has the permission to publish the blocks in that network.

1. Public
2. Private
3. Consortium
4. Hybrid

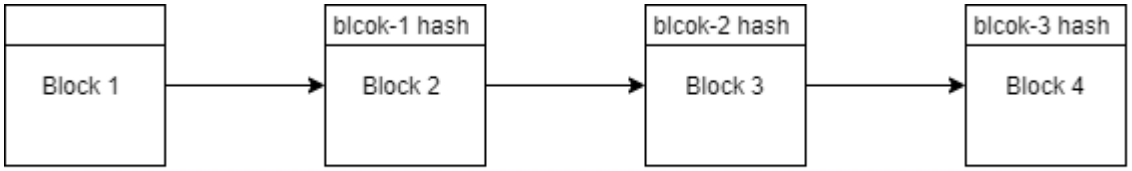

Fig-1 Blockchain

Blockchain Mechanism

A blockchain contains many blocks connected to its previous block. Every block contains the hash of the information of transactions for which that block is created (mined) and the hash of the previous block. The hash is a cryptographic term used to represent encrypted data of some real data encrypted by some encryption algorithm. Hash cannot be reversed into original form once it's encrypted. Every block has two parts one is block header and another one is block data.

There is an important aspect in Blockchain, who will publish the next block. This is solved by applying one of the possible consensus algorithms. The most widely used algorithms are Proof of Work, Proof of stake, Round robin, Proof of authority/Proof of Identity.

In proof of work, the Publisher needs to solve a computationally difficult puzzle to publish a new block. Bitcoin uses a proof of work consensus model.

Proof of stake is based on the idea that the more stake a publisher has invested into the blockchain is more likely to publish the block. Ethereum blockchain is the biggest block using proof of stake to get the next block into the chain.

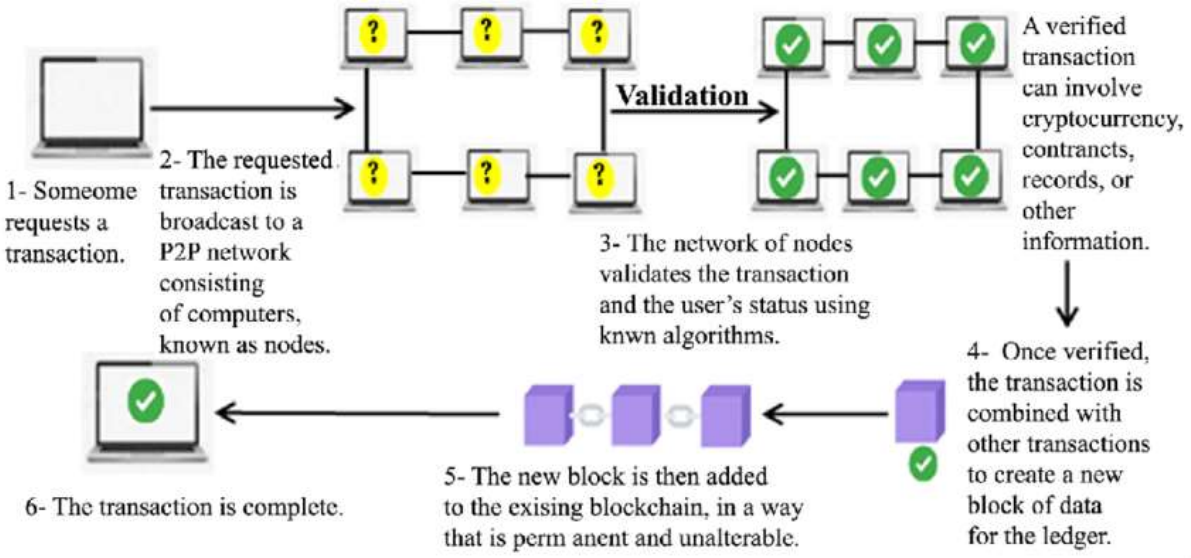

Fig 2 Blockchain mechanism [16]

Smart Contract

The smart contract is a technology used for maintaining a blockchain publically available to add transactions. Smart contracts are a block of codes that reside on multiple nodes distributed over the blockchain. Szabo (1994) defined a smart contract as a piece of computerized transaction protocol that satisfies contractual conditions such as payment terms,

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 7| Issue: 9| September 2021|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2021: 8.047 || ISI Value: 1.188

confidentiality, or enforcement, reduces exceptions, and minimizes the need for trusted intermediaries [4] . Smart contracts use to maintain the trust between two anonymous parties without any third party, using a distributed ledger of transactions reside over the network. Smart contracts are typically used to automate the execution of contracts. Etherium is the widest network using smart contracts for maintaining the blockchain network used for multiple purposes. Smart contracts provide security and transparency for a blockchain distributed ledger. Once a smart contract is deployed. It starts working and whenever the conditions are satisfied in a smart contract, a new is added to the network. Usually, only the smart contract owner can destruct the contract [2].

Smart contract development

A smart contract is likely to be a class that includes state variables, functions, function modifiers, events, and structures that are intended to execute and control relevant events and actions according to the contract terms. Besides, it can even call other smart contracts. Each smart contract includes states and functions. The former are variables that hold some data or the owner's wallet address [2].smart contract can be developed and deployed in different languages and Platforms

Bitcoin [5] is a public blockchain platform that can be used to process cryptocurrency transactions, but with a very limited computing capability. it uses a stack-based bytecode scripting language[2]. The ability to create a smart contract with rich logic using the Bitcoin scripting language is very limited. Major changes would need to be made to both the mining functions and the mining incentivization schemes to enable smart contracts proper on Bitcoin's blockchain.

NXT is a public blockchain platform that includes built-in smart contracts as templates [6]. It relies entirely on a proof-of-stake consensus protocol. I allow us to choose the already developed template to use for creating a smart contract. Any type of customization is not allowed due to the lack of Turing completeness in its scripting language.

It includes a selection of smart contracts that are currently living. However, it is not Turing-complete, meaning only the existing templates can be used and no personalized smart contract can be deployed [2]

Ethereum [7] is the first blockchain platform for developing smart contracts. It supports advanced and customized smart contracts with the help of a Turing complete virtual machine, called the Ethereum virtual machine (EVM)[6]. EVM is the runtime environment for smart contracts, and every node in the Ethereum network runs an EVM implementation and executes the same instructions. Solidity, as a high-level programming language, is used to write smart contracts, and the contract code is compiled down to EVM bytecode and deployed on the blockchain for execution[2]. The Ethereum platform can support withdrawal limits, loops, financial contracts, and gambling markets[6].

Hyperledger Fabric [8] Is a private blockchain rather than a public blockchain listed above. Permissioned with only a collection of business-related organizations can join in through a membership service provider, and its network is built up from the peers who are owned and contributed by those organizations. Hyperledger Fabric is an open-source enterprise-grade distributed ledger technology platform, proposed by IBM and supports smart contracts. It offers modularity and versatility for a broad set of industry use cases. The modular architecture for Hyperledger Fabric accommodates the diversity of enterprise use cases through plug-and-play components [1].

In Ethereum, the state is made up of objects called "accounts", with each account having a 20-byte address and state transitions being direct transfers of value and information between accounts[7]. An Ethereum account contains four fields

- The nonce, a counter used to make sure each transaction can only be processed once
- The account's current ether balance
- The account's contract code, if present
- The account's storage (empty by default)

III. DISCUSSION

A smart contract is a computer program executed on virtual machines which use its resources to calculate blockchain blocks like memory and computational power. So programming smart contracts correctly and analysis of resources consumption in creating smart contracts is an important area of research. Some newly proposed programming languages such as Solidity, SmaCoNat [9], and Flint [10]. For instance, Regnath and Steinhorst [9] proposed a human-readable, security, and executable programming language called SmaCoNat. The authors converted programming language grammar into natural language sentences in order to improve program readability[2].

Transaction and Gas

Ethereum provides a decentralized Turing complete machine, namely the Ethereum Virtual Machine (EVM), to execute scripts using an international network of public compute nodes [19]. On Ethereum, people can use programming languages, e.g., Solidity11 and Viper12, to develop complex smart contract applications.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 7| Issue: 9| September 2021|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2021: 8.047 || ISI Value: 1.188

The term "transaction" is used in Ethereum to refer to the signed data package that stores a message to be sent from an externally owned account[7].

Ethereum uses a pricing system i.e. gas [7] for all transactions running on it. Gas is a measure of how much computing resources a transaction would cost. People need to pay a gas fee (in Ethers) for each transaction they make, and a transaction would fail if it runs out of gas[11].

Whenever the user wants to make a transaction into the Ethereum blockchain, Transaction should contain two important parameters START GAS and GAS PRICE to start execution. START GAS is the limit and the GAS PRICE is a fee to pay to the miner per computational step[7]. If the transaction runs out of gas, all state changes revert except for the payment of the fees and if traction execution halts with some gas remaining then the remaining proportion of the fees is refunded to the sender.

To calculate the transaction fee in ether, need to calculate the START GAS * GAS PRICE, Before initializing the transaction sender should have START GAS * GAS PRICE [7] much ether into the Ethereum account. If the transaction is completed successfully then the consumed gas * GAS PRICE amount of ether automatically passed to the minor account and the remaining gas from the START GAS will be returned to the sender.

Due to the unique gas mechanism in smart contract development where the execution of smart contracts would cost gas and users need to pay the gas fee as a result, developers need to pay special attention to gas consumption during smart contract development.

Gas is money

On public blockchain platforms like Ethereum, all the resources that a smart contract uses would translate into actual direct costs that need to be paid by users in terms of gas. In other words, “Gas is money for users' ' (P1), thus developers need to be much more conscious of resource consumption [11]. A contract under Ethereum has to be executed under very right constraints. All The Resources the user would translate into actual direct costs.

Every single operation in Ethereum, be it a transaction or a smart contract instruction execution, requires some amount of gas. The gas consumption of the Ethereum Virtual Machine (EVM) instructions is spelled out in [26]; importantly, instructions that use replicated storage are gas-expensive.

Performance issues and gas optimization

Ethereum smart contracts are typically developed through solidity[12]. before being compiled into byte codes that can be executed by the Ethereum Virtual Machine (EVM). Programmatically gas is money, optimized smart contracts can lead to unnecessary gas leaks and, thus, to money losses.Q

Every single operation in Ethereum, be it a transaction or a smart contract instruction execution, requires some amount of gas.

Gas leaks in transaction

The programming and coding choice depend on developers, data structures used, the number of cycles, the kind of instructions, the types of variables used, where and how they are initialized or valued, which may affect the gas consumption of a smart contract.

Although Research outlines design patterns and guidelines for developing optimized code still developers feel pain to write gas-optimized code.

GASMET SUITE [12] provides many gas smells which leads to gas leaks in the form of metrics. V, Functions Returning Local Variables (RLV), Global Variables (GV), Number of Loops (NLF6)., Number of non-32-bytes variables (NU), Indexed Parameters (IP), Number of Mappings (NM), Mappings and Arrays (MA), External Calls (EC), Boolean Variables (BV), Number of Events (NE) and Defined Functions (DF)[12]. These all factors of coding have a great impact on the development of cost-effective smart contract development that will reduce gas leaks.

GASOL[1] is a gas optimizer tool that can be used while developing a smart contract as a plugin in eclipse IDE. This tool detects potential sources of optimization [1] and feeds them to the optimizer to generate an optimized Solidity function within a new file

GASCHEKER[13] is another tool for automatically identifying gas-inefficient code. It Identifies ten gas-inefficient programming patterns, some coding practices and space sequence then uses Symbolic execution to detect them in the Byte code. Patterns are like Opaque Predicate (comparison only single output), dead code (code will not be executed in practice), some In-efficient loop operations, and Wasted Disk Space[13].

GASPER[14] is another tool like GASCHEKER Identify 7 costly patterns to detect gas leaks by analyzing smart contract bytecode.

Out-of-Gas Conditions and Gas Prediction

Out of Gas is a vulnerability when using EVM for a smart contract this condition arises when the gas limit provided by the sender is exhausted in the middle of a transaction. Whenever an out of gas exception occurs the sender has to bear the losses. If the transaction does not exceed the amount of gas set by the initiator, then the latter will get back the (GASPRICE × GASLIMIT) − Gas Cost; while the

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 7| Issue: 9| September 2021|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2021: 8.047 || ISI Value: 1.188

Gas Cost is paid to the miner, Otherwise, the amount is lost[15]. Transaction History summarizes can be used to predict the amount of gas used by transaction for a given address. Being able to predict gas consumption has practical implications. The initiator set an appropriate Gas Limit, preventing transactions from running out of gas the initiator set an appropriate Gas Limit, preventing transactions from running out of gas. By using some predictive machine learning models trained on old transactional data some researchers try to predict the gas consumption of the transaction.

f_{TX} is Frequency, i.e. the number of all transactions per day author trained some models with the help of these machine learning algorithms (Ordinary Least Squares (OLS), Ridge, Bayesian Ridge, Lasso, and Stochastic Gradient Descent Linear Regressions, Linear Support Vector Regression, and Decision Tree Regressor)[15] with the default hyperparameters, using 5-fold cross-validation on the train set. Using R-Squared (R2) as the performance measure, which indicates how well the selected independent variables explain the variability in the dependent variable, the author selected and fine-tuned the best models (OLS and Lasso) using a Grid Search[15]. In the end, with a R^2= 0,729373, the Lasso model performs best and gives the following equation:

$$gas_{used} = 14382 - 944 * r_{rec} - 1,55e{-}11 * r_{rec\ ETH} - 776 * r_{sent\ ETH} + 81 * r_{contract} + 9310952 * \mu_{gasUsed} - 48 * N_{TX} \quad (1)$$

Where gas is consumed by the sender (μgas used), an account engages in transactions transferring ether (rec ETH and sent ETH), the less that account is predicted to consume gas in a transaction; and the more an account actively uses Ethereum (NTX), the less it is predicted to consume gas[15].

There are more prediction models and machine learning algorithms like deep learning and neural networks that can be used to predict gas consumption in the future.

IV. CONCLUSION

With the help of this review on gas consumption and tools available for DApps development we can conclude that Blockchain is an emerging technology creating DApps using Ethereum with help of Ethereum virtual machine to create smart contracts are in its middle age there is a large gap in researches and researchers need to research more on gas prediction. There are some tools available in the market for writing gas-efficient code development but minimal research is available on gas predictions. Tools are required to predict the gas required for the next transaction in the Ethereum based smart contract.

V. REFERENCES

1. *E. Albert, P. Gordillo, and G. Rom, "Ethereum Smart Contracts," no. i, pp. 1–8.*
2. *S. N. Khan, "Blockchain smart contracts: Applications, challenges, and future Blockchain smart contracts: Applications, challenges, and future trends," no. April, 2021.*
3. *D. Yaga and D. Yaga, Blockchain Technology Overview Blockchain Technology Overview. .*
4. *L. Ante, "Smart Contracts on the Blockchain – A Bibliometric Analysis and Review," no. 10, pp. 1–48, 2020.*
5. *S. Nakamoto, "Bitcoin : A Peer-to-Peer Electronic Cash System," pp. 1–9.*
6. *M. Alharby and A. van Moorsel, "Blockchain Based Smart Contracts : A Systematic Mapping Study," pp. 125–140, 2017, doi: 10.5121/csit.2017.71011.*
7. *B. V. Buterin, "A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM," no. January, pp. 1–36, 2009.*
8. *E. Androulaki et al., "Hyperledger Fabric : A Distributed Operating System for Permissioned Blockchains."*
9. *S. Steinhorst, "SmaCoNat : Smart Contracts in Natural Language," doi: 10.1109/FDL.2018.8524068.*
10. *F. Schrans, "Writing Safe Smart Contracts in Flint," no. June, 2018.*
11. *W. Zou et al., "Smart Contract Development : Challenges and Opportunities," no. March, pp. 1–20, 2018.*
12. *G. Canfora, A. Di Sorbo, S. Laudanna, A. Vacca, and C. A. Visaggio, "Profiling Gas Leaks in Solidity Smart Contracts," 2020.*
13. *T. Chen et al., "GasChecker : Scalable Analysis for Discovering Gas-Inefficient Smart Contracts," vol. 14, no. 8, 2020, doi: 10.1109/TETC.2020.2979019.*
14. *T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," SANER 2017 - 24th IEEE Int. Conf. Softw. Anal. Evol. Reengineering, pp. 442–446, 2017, doi: 10.1109/SANER.2017.7884650.*
15. *S. Bouraga, "An Evaluation of Gas Consumption Prediction on Ethereum based on Transaction History Summarization," 2020 2nd Conf. Blockchain Res. Appl. Innov. Networks Serv. BRAINS 2020, no. September, pp. 49–50, 2020, doi: 10.1109/BRAINS49436.2020.9223288.*
16. *R. A. Salha, M. A. El-Hallaq, and A. I. Alastal, "Blockchain in Smart Cities: Exploring Possibilities in Terms of Opportunities and Challenges," J. Data Anal. Inf. Process., vol. 07, no. 03, pp. 118–139, 2019, doi: 10.4236/jdaip.2019.73008.*