Analysis of Blockchain Sustainability through the Comparison of Different Smart Contracts Programming Languages

Stefano Bistarelli[1,†], Marco Fiore[2,*,†], Antonio Ignazio Lazzizera[2,†], Ivan Mercanti[1,†] and Marina Mongiello[2,†]

[1]Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy
[2]Department of Electrical and Information Engineering, Polytechnic University of Bari, Bari, Italy

Abstract

This paper aims to analyze the correlation between programming languages used to implement smart contracts and the related Blockchain platforms, seeking an economically and environmentally viable solution. The research selects four programming languages commonly employed for smart contract creation, defining three standard functions that will be tested on the Ethereum, Tezos, Solana, and Polkadot Blockchains. The study investigates how these programming languages directly impact the amount of energy resources required and subsequently influence the cost of executing a smart contract.

Keywords

Blockchain, Sustainability, Smart contracts, Comparison

1. Introduction

The emergence of the digital revolution has brought about various transformative technologies, but none quite like Blockchain. This innovative technology can potentially revolutionize the market and security vision that has characterized modern civilization. Blockchain technology is known for its exceptional security, trust, decentralization, and community concepts, making it the most significant technology to fundamentally change the market and security vision that has marked modern civilization.

Blockchain technology has diverse applications in various fields, such as healthcare [1], agri-food [2, 3], election voting systems [4], and transactions [5]. Although Blockchain has significant potential and interest, programmers and project proponents must address environmental sustainability [6] and carbon footprint issues. They must keep the network running 24/7 with machines that harness energy resources while minimizing the consumption of raw materials and ensuring a low-energy impact solution for users while providing maximum efficiency [7].

One of the recent developments in Blockchain technology is the rise of smart contracts. These contracts are written in different programming languages, which determine their complexity and, consequently, the transaction's cost. In this paper, we analyze cost by considering the smart contracts based on different blockchains and various programming languages. This analysis will help identify which of the blockchains is the most economical and sustainable in terms of energy for those who execute them and the environment. The study will provide valuable insights into today's most economical and sustainable Blockchain technology.

6th Distributed Ledger Technology Workshop (DLT 2024), May 14-15, Turin, Italy
*Corresponding author.
†These authors contributed equally.
stefano.bistarelli@unipg.it (S. Bistarelli); marco.fiore@poliba.it (M. Fiore); a.lazzizera@studenti.poliba.it (A. I. Lazzizera); ivan.mercanti@unipg.it (I. Mercanti); marina.mongiello@poliba.it (M. Mongiello)
0000-0001-7411-9678 (S. Bistarelli); 0000-0003-0102-2394 (M. Fiore); 0000-0002-9774-1600 (I. Mercanti); 0000-0002-1477-1434 (M. Mongiello)

2. Background

In this section, we present an overview of the blockchain, and we define their consensus protocol.

2.1. Blockchain type

The vast world of blockchain does not present one universal model for all types of uses. However, it is divided into two different categories on which the significant blockchains are based: UTXO and Account-based.

A *UTXO*, which stands for Unspent Transaction Output, is the amount of cryptocurrency obtained by the recipient or saved by the sender as a result of a transaction on some blockchain; the amount of each UTXO can then be ”spent” on a new exchange. The balance may be viewable in a crypto wallet, a shared ledger with a unique address that reports information on all transactions in crypto.

An *account* is a structure used to identify users and smart contracts and verify the validity of transactions made. Each account, specifically, is associated with a wallet address and a status containing a set of information about the balance of crypto possessed by the user. When a transaction is initiated on an account-based blockchain, the system verifies the amount of crypto possessed by the user who initiated the transaction, ensuring that the wallet contains an equal or more significant number than the amount to be sent. The account model can be compared, very simplistically, to a banking-type service.

A *permissionless* blockchain is a decentralized and distributed ledger that enables secure, transparent, and tamper-proof transactions. It operates on a peer-to-peer network where all participants have equal access to the network and its data and can participate in transaction validation. Since these blockchains are public, anyone can join the network and validate transactions. The use of a public ledger, accessible to all participants, increases accountability, enhances security, and strengthens trust, a fundamental aspect of blockchains [8].

Private blockchains are not accessible to individual users without authorization. They are restricted and usually limited to enterprise networks, providing higher levels of error control and transaction validity.

2.2. Proof of work vs. Prof of stake

Like all technologies, blockchains are described and organized according to a set of rules defined within the so-called ”Protocol.” A protocol in Blockchain is a set of rules, standards and algorithms that govern the creation, validation and maintenance of a blockchain network. It defines the block structure, consensus mechanism and validation process for transactions within the Blockchain. There are several types, but the main distinction is made between the two main ones: Proof of Work and Stake.

Proof of Work is a protocol used to reach distributed consensus. It is based on finding a computationally tricky number, but once found, it becomes easy for all other nodes to verify its correctness. In a system using PoW, a block is valid only if it contains a valid solution to the PoW. Within this protocol, nodes in the network compete to solve a complex mathematical problem, an inverse hash. Solving this problem is a random process with very low probability, and the only way to find a valid PoW is to try all possible combinations until the right one is found. The first miner who solves the problem has the right to create the next block, earning a reward in cryptocurrency. Once the block is created, it is transmitted to the network, waiting for other nodes to check its validity. The difficulty of the resolution is updated periodically about the network of the network to keep the time required to generate a block as constant as possible. The main advantage of Proof of Work is the firm guarantee of immutability.

The disadvantages of this protocol are:

- *Massive energy consumption*: as Bitcoin is the largest project using Proof of Work, it currently consumes about 0.3% of the world’s electricity (19.69 GW per day[1]). About $1 million per day between electricity and mining hardware.

[1] https://ccaf.io/cbnsi/cbeci.

- *Scalability difficulties*: Proof of Work is one of the bottlenecks in scaling the system. However, it is possible to make this blockchain scalable without changing the consensus algorithm, either by adopting off-chain solutions or by changing the block size.
- *Vulnerability to a 51% attack*: If a miner reaches 51% of the network's total computing power, it could create blocks faster than the remaining miners. Together, the miner in question could reverse or modify some transactions.

Proof of Stake is another protocol used within blockchains. It serves the same purpose as PoW, but the process to achieve the end goal is different. In PoS, validators are alternated, chosen in advance based on the amount of cryptocurrency they hold for the relevant blockchain, also referred to as a stake. Users in possession of tokens can stake their tokens to have, in return, the right to confirm a blockchain transaction and receive a reward by dividing validators. The creator of a new block is then chosen in advance, using a combination of different parameters depending on the type of algorithm used. PoS is more efficient because complex calculations need not be followed for each new block, and attacks are more expensive. PoS is also vulnerable to a 51% attack. An attacker, in this case, will not need 51% of the total, but 51% of the total tokens. However, if an attacker tried to buy so many tokens, the market would react with a rapid increase in the price of the tokens. Finally, PoS is, therefore, much cheaper as there are no elevated electricity costs (680.33 kW per day on Ethereum[2]) and powerful hardware. A blockchain-based on Proof of Stake can efficiently run even on hardware with low computational capabilities, such as a Raspberry. This is the most widely used protocol in blockchains today.

2.2.1. Smart contract

A smart contract is a contract that governs the terms and conditions of an agreement between parties. However, unlike a traditional contract, the terms of a smart contract are executed based on a code programmed on a blockchain that mechanically complies with the terms of a legally binding agreement. The advantages over traditional contracts are reduced risks, lower administrative costs and more efficient business processes. A smart contract cannot be executed without payment of a fee, which is measured in terms of gas. Gas refers to the amount of computational work that miners or validators require in order to authorize transactions or execute smart contracts. Gas fees are valued in gwei on Ethereum, where one gwei is equivalent to a certain amount of ETH. The cost of gas fees is not fixed but depends on various defined parameters such as the blockchain protocol, functions implemented in the smart contract, and the programming language used to develop it.

Gas fees work on the laws of supply and demand: by offering a higher fee, validators will be incentivized to process the transaction faster. A portion of the gas fees can also be awarded as a premium to the validator who completes the transaction first. These fees are critical to the proper functioning of the network, much like gasoline in a car, ensuring the stability and safety of the system. When developing smart contracts and blockchain-based apps, it is essential to consider the unique limitations and features required by blockchain technology. Identifying and investigating all these changes is necessary to build a complete body of knowledge based on blockchain software engineering.

3. Proposed approach

Our research analyses smart contracts across various blockchains and uses different programming languages to evaluate their costs. This analysis will help identify the most cost-effective and energy-efficient blockchain for those who run it and the environment in which it operates. Moreover, we analyze how the use of different programming languages for writing smart contracts on various blockchains affects the transaction gas cost, defining which among those considered are more cost-effective. Regarding sustainability, we also consider the environmental impact of these blockchains in terms of electricity consumption and CO2 emissions. The chosen programming languages vary in type

[2] https://ccaf.io/cbnsi/ethereum.

to enable a comparison between high and low levels, and consequently, the blockchains that utilize them have been selected. For all programming languages used in smart contract writing, the analyzed blockchains are all permissionless.

Private blockchains are not included in the analysis because they are not accessible to individual users without authorization.

Chosen blockchains are summarized below:

- **Ethereum and Solidity**
  Ethereum, created by Vitalik Buterin in 2013, revolutionized blockchain with its programmable smart contracts. Operating as a permissionless public blockchain, Ethereum has transitioned to a Proof of Stake consensus protocol in 2023. Its native token, ETH, fuels platform activities and facilitates value exchange. Ethereum's pivotal role extends beyond cryptocurrency, powering the burgeoning decentralized finance (DeFi) sector. Advantages include a deflationary token supply, a robust community fostering demand, passive income opportunities through staking, and serving as foundational infrastructure for emerging projects like Non-Fungible Tokens (NFTs). Solidity, a high-level programming language primarily for Ethereum, facilitates smart contract development. Inspired by C++, Python, and JavaScript, Solidity interfaces seamlessly with the Ethereum Virtual Machine (EVM). Developers leverage Solidity to craft smart contracts governing various functionalities, including voting mechanisms and multi-signature wallets. Solidity programming necessitates Solidity compilers, such as Remix and npm, tailored for Linux and MacOS environments.
- **Tezos and SmartPy**
  Tezos, introduced in 2014, offers innovative features surpassing traditional blockchains like Ethereum. Its Liquid Proof-of-Stake (LPoS) consensus protocol allows for decentralized smart contract execution and governance through a voting mechanism. Tezos adaptability enables dynamic adjustments to its protocol parameters, enhancing security through formal methods. The native cryptocurrency, XTZ, fuels transactions within the ecosystem. Tezos architecture comprises multiple layers, with a peer-to-peer layer ensuring network connectivity and a subsequent layer treating the network as a distributed database. Blocks are extracted, modified, and passed to the economic protocol layer, responsible for consensus enforcement. SmartPy, a Python-like high-level language, simplifies smart contract development on Tezos. It offers a user-friendly syntax and a secure programming environment. Initially, smart contracts on Tezos were coded in Michelson, a low-level language focused on security and precision. SmartPy's compiler seamlessly translates code into Michelson for testing and deployment onto the blockchain, enhancing developer efficiency and contract reliability.
- **Polkadot and Rust**
  Polkadot, established in 2017, revolutionizes blockchain interoperability with its Nominated Proof of Stake (NPos) consensus mechanism. The network features a primary relay chain governing inter-chain communication and numerous user-created parachains operating autonomously within the Polkadot ecosystem. Polkadot boasts a throughput capability of 1,000 transactions per second. Rust, developed by Mozilla Research, epitomizes modern systems programming by seamlessly blending low-level performance control with high-level convenience and robust security assurances. It excels in speed and memory efficiency, making it suitable for critical performance services and embedded systems, while its advanced type system and ownership model ensure memory and thread safety, minimizing bugs at compile time. Renowned for its eco-friendly attributes, Rust stands out as one of the most environmentally sustainable programming languages in use today.
- **Solana and C++**
  Solana, introduced in 2018, stands out as a blockchain platform optimized for widespread adoption, catering to diverse sectors such as finance, NFTs, payments, and gaming. Operating as a global state machine, Solana is distinguished by its high-performance architecture, open nature, and decentralized ethos. Its native cryptocurrency, SOL, drives transactions within the ecosystem.

Solana's innovative Proof of History (PoH) consensus mechanism ensures verifiable event order and time intervals, fostering trust without reliance on external entities or synchronized clocks. C++, renowned for its versatility and efficiency, serves as an ideal programming language for Solana development. Combining low-level capabilities with high-level functionalities, C++ enables seamless integration with system libraries, making it a preferred choice for video game development, AR/VR applications, IoT devices, and operating systems. Notably, C++ boasts exceptional energy efficiency attributed to its minimal storage usage, further enhancing its appeal for Solana development.

3.1. Smart contracts definition

The selection of functions to analyze within smart contracts remains a critical aspect of the comparison. Due to the heterogeneous nature of the chosen programming languages, defining the programs to execute proved to be challenging, as not all languages support certain high or low-level functionalities, object-oriented programming, or other features. Therefore, the decision was made to opt for very basic projects to achieve a satisfactory overall understanding.

The chosen functions, ordered by increasing complexity, are as follows:

- Sum of two numbers: we test how different programming languages manage the sum of two integers number, being it a mathematical primitive in equations. This primitive is shown in [9].
- Conversion of an integer into a string: we aim to analyze the behaviour of each Blockchain to a cast operation, as indicated in [10].
- Creation of a personal data record for a hospital patient: the goal of this contract is to verify the reaction of tested platforms with the management of a structure composed by strings and integers. Healthcare is between the main applications for evaluating smart contracts, as highlighted in [11].

Each function has been implemented in a smart contract written for every analyzed Blockchain. A total of 12 smart contracts have been written and tested.

4. Tests and results

This Section will show all the tests conducted on the selected blockchain and the related results.

Ethereum - Solidity For the Ethereum blockchain, Remix IDE was utilized to conduct all test cases. Remix IDE is a configuration-free tool featuring a GUI for smart contract development. It is employed by both experts and beginners, facilitating a straightforward deployment process on any chosen chain. All tests were simulated using virtual cryptocurrencies within a testnet provided by the IDE. This approach ensured a controlled environment for experimentation and validation of smart contracts on the Ethereum blockchain.

Tezos - SmartPy For the Tezos Blockchain, SmartPy IDE was selected. SmartPy IDE provides a comprehensive online environment for developing, testing, and deploying smart contracts on the Tezos Blockchain, all within the familiar Python syntax. Simulation was conducted on a test network called Ghostnet, which is popular for testing contracts without spending real cryptocurrencies.

Polkadot - Rust For the analysis of Polkadot, a framework similar to Remix for Ethereum, Substrate, was employed. Substrate executes the development of Blockchain applications by providing developers with a versatile toolkit that overcomes the constraints typically encountered in other frameworks. Unlike rigid frameworks that impose predefined structures, Substrate offers a flexible and modular architecture, empowering developers to tailor the blockchain infrastructure precisely to their project's needs.

Table 1
Results of contracts execution, in gas, in each chosen Blockchain

Blockchain	Smart contract language	Integers sum	Cast operation	Healthcare scenario
Ethereum	Solidity	328	1910	11681
Tezos	SmartPy	367	**1033**	**9112**
Polkadot	Rust	903	2416	13006
Solana	C++	**299**	4591	17467

Substrate's agility stems from its design philosophy, which prioritizes adaptability and customization. Its modular nature allows developers to effortlessly integrate and customize components, enabling the seamless implementation of complex functionalities. By leveraging Substrate, developers can expedite the development process, reduce overhead costs, and enhance the scalability and interoperability of their Blockchain solutions.

Solana - C++ A local testing environment has been developed for Solana analysis. The tool used for testing was SolanaLabs, a local framework-like tool enabling the evaluation of various aspects of a contract for development and configuration. For the testnet, SolanaTestnet, provided by Solana, proved to be ideal for testing the smart contracts. Lastly, the Mathwallet, a local wallet, was utilized. It was populated with SOL using the SolanaDevnetFaucet.

4.1. Collected gas fees

We collect the spent gas for deploying each contract in a Blockchain. Results are summarized in Table 1.

Fig. 1 shows the gas fee comparison between the analyzed platforms. The trend depicted in the figure is nearly linear across all languages. Costs are significantly low when dealing with the sum smart contract. This can be attributed to the minimal gas required for the transaction since only a few low-level functions are used, resulting in low gas consumption. Upon closer inspection, intersections of graphs are visible, particularly in the case of the sum contract. This can be attributed to the nature of programming languages. C++, being a low-level language, enables operations directly on memory, thus requiring minimal gas for execution on the blockchain. The scenario changes when considering the cast smart contract. Here, high-level languages like SmartPy and PyTeal prove to be more cost-effective as they possess built-in syntax, commands, and functions for converting an integer to a string with just a single line of code. Conversely, low-level languages incur higher costs due to multiple and resource-intensive register operations. For the healthcare smart contract, the overall cost trend becomes clearer. Here, creating a Patient object creates some differences in different languages. Low-level and hybrid languages like Rust need to rely on external or more complex functionalities, leading to higher resource consumption on the chain.

4.2. Sustainability analysis

In this section, we present an analysis of various blockchain networks regarding their annual transaction volume, energy consumption, and energy cost per transaction. Table 2 shows the results of the analysis.

Table 2
Analysis of CO2 emission for each analyzed Blockchain

Blockchain	**Transactions per year**	**Power consumption [GWh]**	**CO2 emission [kg]**
Ethereum	4.9e10	7.210	2784143
Tezos	1,2e8	1.010	302461
Polkadot	3,1e6	0.473	122275
Solana	3,3e8	5.961	2228750

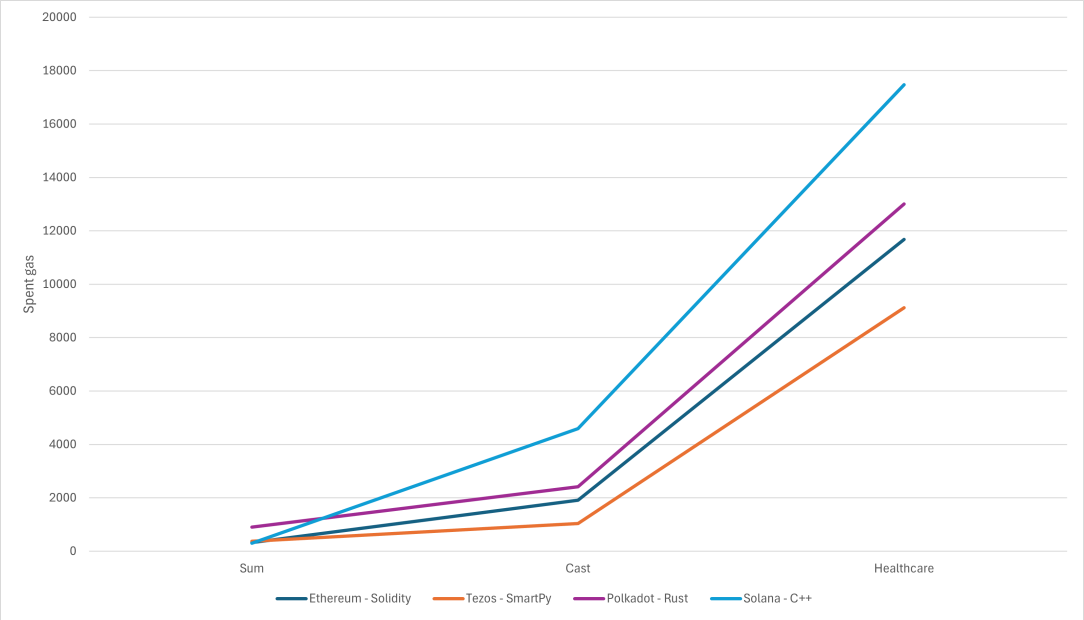

Figure 1: Gas spent for each Blockchain

All the parameters are strongly influenced by multiple factors, primarily the consensus protocol, which, even though based on PoS, has various variants that characterize emissions. Regarding power consumption, the values vary widely, which is justifiable due to each blockchain's popularity: the more users utilize them, the more energy consumption is affected. In direct proportion, data about CO2 emissions can be derived. However, this data significantly varies depending on the energy sources used to operate the nodes within the chain.

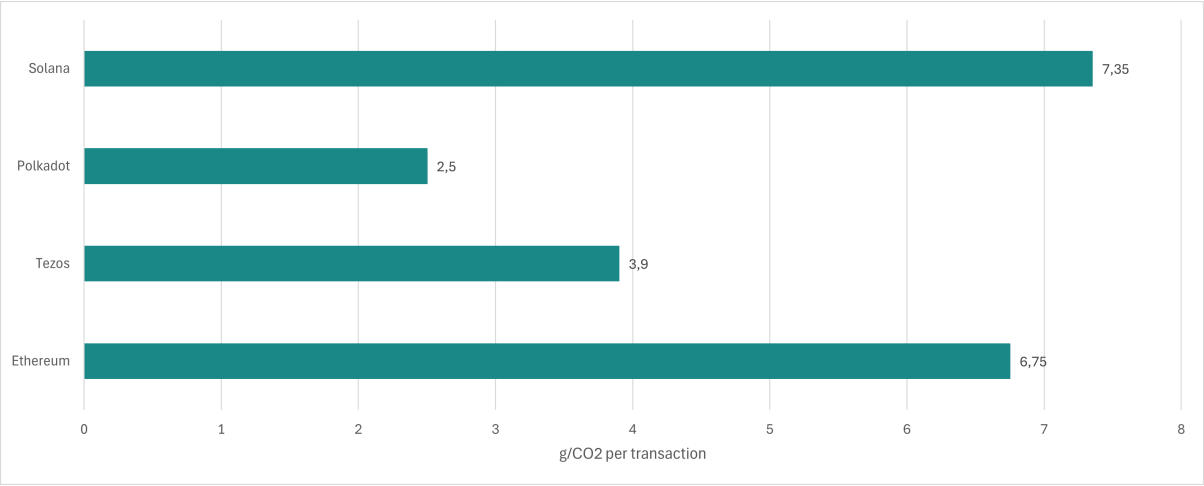

Figure 2: CO2 emission per transaction for each analyzed Blockchain

Thanks to official data gathered from different sources[3], it is possible to create a graph regarding emissions per transaction, as shown in Fig. 2. Some values, instead, were calculated by knowing the number of transactions and the total estimated emission for the year 2022. A clear difference in ecological impact data can be observed. As blockchain with lower CO2 emissions, Tezos stands out. Tezos' low carbon footprint means that developers and users can prioritize innovation without compromising sustainability. Being a platform designed to evolve through its on-chain governance

[3]https://etherscan.io/, https://indices.carbon-ratings.com/,https://greenpolkadot.io/, https://www.theblock.co/data/crypto-markets/spot,https://indices.carbon-ratings.com/

mechanism, Tezos' efficiency is not accidental but designed. This flexible design allows the blockchain to adapt to future needs and grow based on users' requirements on the platform. In fact, Tezos' blockchain has increased energy efficiency per transaction by at least 70%, with an estimated energy requirement per transaction being less than 30% in 2021 compared to that of 2020[4]. Regarding Polkadot's consumption, this was possible to derive by calculating, through multiple block explorers, the number of daily transactions, averaging them annually. The total includes transactions executed on the relay-chain and parachains. Polkadot is designed to power the next big wave of web innovation without the high energy consumption of traditional proof-of-work blockchains. The innovative consensus protocol consumes a small fraction of the energy consumed by conventional blockchains[5]. In conclusion, the last two blockchains, Solana and Ethereum, have very similar consumption levels. Ethereum has long assessed and analyzed consumption, thus deciding to undergo a transition called the 'Merge', radically modifying its consensus protocol, which was very energy-intensive and consequently also carbon emissions: from PoW to PoS. This reveals itself as one of the most intelligent choices given the blockchain's vast capabilities and popularity, which, behind the famous Bitcoin, serves as a model for all new projects. The Solana Foundation, on the other hand, is committed to studying the impact of the Solana blockchain, making data public and adopting measures to reduce the chain's footprint to zero. Solana is the first Layer 1 blockchain with smart contract capability with real-time energy emissions monitoring, allowing anyone worldwide to examine the network emissions down to the validator or RPC level[6].

5. Related Work

BLOCKBENCH [10] was introduced in 2017 as the first evaluation framework for analyzing private blockchains in a fair and in-depth manner. The framework enables a better understanding of various system design choices and enables an objective comparison between different platforms. Integrating any private blockchain into BLOCKBENCH is a simple process which can be accomplished via its APIs. BLOCKBENCH provides performance evaluation, measuring overall and component-wise performance in terms of throughput, latency, scalability and fault tolerance. The authors used BLOCKBENCH to comprehensively evaluate three major private blockchains: Ethereum (as a solidity blockchain), Parity, and Hyperledger Fabric. The results suggest that there is still significant room for improvement before these systems become viable replacements for current database systems in traditional data processing workloads. Furthermore, the assessment highlights performance gaps among the three systems attributed to the design choices at different blockchain software stack levels.

In [12], the authors have used a benchmarking approach in their research and shared the preliminary results for the Python Ethereum client running on a Mac. The study reveals that there can be significant differences in the reward per CPU second for functions in Ethereum's most popular contracts, which can result in misaligned incentives that impact the dependable operation of the blockchain. Additionally, the research highlights that contract creation, done once for each new contract, can be more lucrative than the regular execution of contract functions.

In 2019, [13] presents a comparative analysis of various distributed ledger technology (DLT) platforms in a diplomatic and unbiased manner. The selection of platforms is based on their popularity, current market share, and evolving trends and approaches. The platforms selected for the analysis are Ethereum, EOS, Hyperledger Sawtooth, and NEO. The comparison is done from both development and performance perspectives to provide a comprehensive understanding of their strengths and limitations. The analysis reveals that Sawtooth offers significant customization capabilities that may affect performance, while EOS maintains stable throughput under varying network scales and loads.

[11] objectively evaluates and compares different smart contract functions. The article suggests using multi-criteria analysis (MCA) to assess and compare functions based on multiple criteria to ensure a

[4] https://tezos.com/carbon
[5] https://polkadot.network/features/technology
[6] https://solana.com/environment

fair evaluation process. The study provides a comprehensive review of the current state-of-the-art in the field by considering various criteria used in the selection process, such as security, scalability, and performance. It also highlights the challenges associated with critical criteria for smart contract selection, including security, privacy, efficiency, scalability, and regulatory concerns. However, the article stresses that the process is complicated by challenges such as the lack of standardization and difficulty comparing platforms. The article emphasizes the need for further research on smart contract platforms' long-term performance and scalability and more comprehensive and objective evaluation methods for MCA of smart contract selection while acknowledging the complexity of the subject matter.

The authors of [9] have proposed a multi-objective test selection technique for smart contracts that aims to balance three crucial objectives: time, coverage, and gas usage. They comprehensively evaluated their approach using five solidity smart contracts based on data collected from GitHub and State of the DApps[7]. They compared their results with various test selection methods in traditional software systems. Through statistical analysis of their experiments, which utilized benchmark Solidity smart contract case studies, the authors have demonstrated that their approach significantly reduces the testing cost while maintaining acceptable fault detection capabilities. These findings have been compared to random search, mono-objective search, and the traditional re-testing method that does not employ heuristic search.

With respect to current state of the art, this paper analyzes different public Blockchains and programming languages, with the final goal to understand the impact of a language on energy consumption requirements.

6. Conclusion

The in-depth analysis conducted on smart contract dynamics reveals a complex landscape where programming languages play a significant role, especially when it comes to low-level languages. These languages can directly influence the efficiency and optimization of operations performed by smart contracts on a blockchain. Although they are faster, do not require compilers, and use minimal memory, they rely on external functionalities that compromise the final cost. Another significant contribution comes from the efficiency of the blockchains on which smart contracts operate. Regarding the actual costs incurred for executing a smart contract on a blockchain, these are still influenced by the blockchain itself: governance policies and resources employed by the operating logic make prices vary significantly. The intrinsic characteristics of the blockchain and the adopted consensus protocol play a predominant role in determining the environmental footprint. The decentralized and diversified nature of consensus protocols can substantially impact energy consumption and CO2 emissions associated with smart contract execution. Although careful programming can contribute to optimizing smart contract efficiency, it is the fabric of the blockchain itself and the consensus rules that predominantly influence carbon emissions.

In summary, while programming languages play a significant role in the internal optimization of smart contracts, the broader context of the blockchain structure and the consensus protocol largely dictates the environmental impact of smart contract-related activities. Additionally, considerations must be made regarding the cryptocurrency market and its daily fluctuations, influenced by global stock markets. Prices and consumption are just some of the aspects to consider when developing a decentralized project or application. For a comprehensive overview, perspectives on the project's blockchain growth, upcoming implementations, and community support should also be evaluated to ensure foresight and backing from the community. Nearly all platforms analyzed, regardless of the programming language for smart contracts, are exploring and testing substantial changes to align with global emission reduction programs.

The conducted analysis is a first step in making a deep analysis on smart contracts execution and Blockchain platforms sustainability in different scenarios. Future developments regard the inclusion

[7] https://www.stateofthedapps.com.

of more Blockchains and smart contracts with different programming languages, together with the development of an automated tool that calculates Blockchain consumptions with a smart contract given by the user.

Acknowledgments

S. Bistarelli and I. Mercanti are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM). This work has been partially supported by:

- GNCS-INdAM, CUP_E53C23001670001;
- European Union - Next Generation EU PNRR MUR PRIN - Project J53D23007220006 EPICA: "Empowering Public Interest Communication with Argumentation";
- University of Perugia - Fondo Ricerca di Ateneo (2020, 2021, 2022) - Projects BLOCKCHAIN4FOOD-CHAIN, FICO, AIDMIX, "Civil Safety and Security for Society";
- European Union - Next Generation EU NRRP-MUR - Project J97G22000170005 VITALITY: "Innovation, digitalisation and sustainability for the diffused economy in Central Italy";
- Piano di Sviluppo e Coesione del Ministero della Salute 2014-2020 - Project I83C22001350001 LIFE: "the itaLian system Wide Frailty nEtwork" (Linea di azione 2.1 "Creazione di una rete nazionale per le malattie ad alto impatto" - Traiettoria 2 "E-Health, diagnostica avanzata, medical devices e mini invasività").

References

[1] M. Attaran, Blockchain technology in healthcare: Challenges and opportunities, International Journal of Healthcare Management 15 (2022) 70–83. URL: https://doi.org/10.1080/20479700.2020.1843887. doi:10.1080/20479700.2020.1843887. arXiv:https://doi.org/10.1080/20479700.2020.1843887.

[2] S. Bistarelli, F. Faloci, P. Mori, *-chain: A framework for automating the modeling of blockchain based supply chain tracing systems, Future Gener. Comput. Syst. 149 (2023) 679–700. URL: https://doi.org/10.1016/j.future.2023.07.012. doi:10.1016/J.FUTURE.2023.07.012.

[3] M. Fiore, M. Frem, M. Mongiello, F. Bozzo, C. Montemurro, G. Tricarico, A. Petrontino, Blockchain-based food traceability in apulian marketplace: Improving sustainable agri-food consumers perception and trust, Internet Technology Letters e503 (2024) 1–6. doi:https://doi.org/10.1002/itl2.503.

[4] S. Bistarelli, I. Mercanti, F. Santini, Enforcing confidentiality in tornado cash-based e-voting systems, in: P. Mori, I. Visconti, S. Bistarelli (Eds.), Proceedings of the Fifth Distributed Ledger Technology Workshop (DLT 2023), Bologna, Italy, May 25-26, 2023, volume 3460 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3460/papers/DLT_2023_paper_7.pdf.

[5] S. Bistarelli, I. Mercanti, F. Santini, An analysis of non-standard transactions, Frontiers Blockchain 2 (2019) 7. URL: https://doi.org/10.3389/fbloc.2019.00007. doi:10.3389/FBLOC.2019.00007.

[6] A. Parmentola, A. Petrillo, I. Tutore, F. De Felice, Is blockchain able to enhance environmental sustainability? a systematic review and research agenda from the perspective of sustainable development goals (sdgs), Business Strategy and the Environment 31 (2022) 194–217.

[7] M. Fiore, M. Mongiello, G. Acciani, A context-aware multiple blockchain architecture for managing low memory devices, in: 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech), 2023, pp. 1–6. doi:10.23919/SpliTech58164.2023.10193670.

[8] T. Bayan, R. Banach, Exploring the privacy concerns in permissionless blockchain networks and potential solutions, IEEE International Conference on Smart Information Systems and Technologies (SIST), IEEE, Astana, Kazakhstan, 2023, pp. 567–572.

[9] B. Alkhazi, A. Alipour, Multi-objective test selection of smart contract and blockchain applications, PeerJ Comput. Sci. 9 (2023) e1587. URL: https://doi.org/10.7717/peerj-cs.1587. doi:10.7717/PEERJ-CS.1587.

[10] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, K.-L. Tan, Blockbench: A framework for analyzing private blockchains, in: Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 1085–1100. URL: https://doi.org/10.1145/3035918.3064033. doi:10.1145/3035918.3064033.

[11] N. M. Alshahrani, M. M. Kiah, B. B. Zaidan, Smart contract evaluation by multi-criteria analysis: Selection challenges and open issues, a review, in: 2023 3rd International Conference on Emerging Smart Technologies and Applications (eSmarTA), 2023, pp. 1–8. doi:10.1109/eSmarTA59349.2023.10293363.

[12] A. Aldweesh, M. Alharby, E. Solaiman, A. van Moorsel, Performance benchmarking of smart contracts to assess miner incentives in ethereum, in: 14th European Dependable Computing Conference, EDCC 2018, Iaşi, Romania, September 10-14, 2018, IEEE Computer Society, 2018, pp. 144–149. URL: https://doi.org/10.1109/EDCC.2018.00034. doi:10.1109/EDCC.2018.00034.

[13] S. Benahmed, I. Pidikseev, R. Hussain, J. Lee, S. M. A. Kazmi, A. Oracevic, F. Hussain, A comparative analysis of distributed ledger technologies for smart contract development, in: 30th IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2019, Istanbul, Turkey, September 8-11, 2019, IEEE, 2019, pp. 1–6. URL: https://doi.org/10.1109/PIMRC.2019.8904256. doi:10.1109/PIMRC.2019.8904256.