Blockchain Performance Benchmarking: a VCG Auction Smart Contract Use Case for Ethereum and Tezos (Short Paper)

Lucas Massoni Sguerra[1] ✉
MINES ParisTech, PSL University, France

Pierre Jouvelot ✉
MINES ParisTech, PSL University, France

Emilio J. Gallego Arias ✉
Inria Paris, France

Gérard Memmi ✉
Télécom Paris, IP Paris, France

Fabien Coelho ✉
MINES ParisTech, PSL University, France

Abstract

The second generation of blockchains introduces the notion of "smart contract" to decentralized ledgers, but with each new blockchain system comes different consensus mechanisms or different approaches on how to assess the cost of computation inside the chain, both aspects that affect the efficiency of the systems as a decentralized computer. We present an experimental comparison of two blockchain systems, namely Ethereum and Tezos, from the perspective of smart contracts, centered around the same implementation of a VCG for Sponsored Search auction algorithm, respectively encoded in Solidity and SmartPy. Our analysis shows the feasibility of implementing an algorithm for sponsored search in such an environment while providing information on how useful these systems can be for this type of smart contracts.

2012 ACM Subject Classification Computing methodologies → Distributed computing methodologies

Keywords and phrases Blockchain, Smart contracts, Ethereum, Tezos, Vickrey–Clarke–Groves (VCG) auction

1 Introduction

The presence of "smart contracts", i.e., distributed code snippets, into recent blockchain architectures such as Ethereum [10], EOS [4] or Tezos [3] calls for a comparative performance analysis of their implementations, in terms of running time, storage requirements or cost. Even if some benchmarking work has been done previously, see for instance [7] and [6] , we are not aware of work performing a comparative analysis of smart contracts. We report here on preliminary results of such a comparison on two architectures, Ethereum and Tezos, using a real-life use case, the Vickrey-Clarke-Groves auction for sponsored search (VCG) algorithm.

Our motivation for designing this particular benchmark setting is two-fold. First, we opted to focus on blockchains adhering to different philosophies, the popular and proof-of-work-based Ethereum and the newer and proof-of-stake-based Tezos, to better assess the performance characteristics of the blockchain ecosystem. Second, our choice of the VCG algorithm is driven by its importance for search-engine and social-network companies, where

[1] Corresponding author

advertisements are the main source of revenues. Since these systems, like Facebook, have to mitigate their bottom line and their users' satisfaction, they often opt for this particular type of auction to sell their ads slots, since there advertisers can indeed bid according to their perceived value of each targeted user.

The importance of the reliability and openness of VCG-based ad bidding processes make them a potentially valuable application for smart contracts. In the rest of this paper, we introduce the Ethereum and Tezos approaches to smart contracts (Section 2), specify the VCG algorithm and its implementation as a smart contract (Section 3), present our test protocol (Section 4), detail our metrics and main results on both Ethereum and Tezos (Section 5), discuss our main findings (Section 6) and finally conclude (Section 7).

2 Ethereum and Tezos Smart Contracts

After its introduction in 2008 with Satoshi Nakamoto's Bitcoin paper, blockchain systems evolved and gained functionalities on top of the peer-to-peer exchange of value. The second generation of blockchains, kick-started by Ethereum, added Turing-complete computation to their offerings, making decentralized applications (dApps) a possibility, via so-called "smart contracts". A smart contract is an autonomous agent that runs on a blockchain and can implement a wide range of applications.

Ethereum is currently the second largest blockchain system, and the first choice for dApps. Its blocks are produced by miners, and the consensus on this blockchain is achieved via the Ethash proof-of-work protocol. Ethereum smart contracts are written in Solidity, a high-level language influenced by C++, Python and JavaScript. Its compiler targets the Ethereum Virtual Machine (EVM), generating EVM opcodes to be executed. Each of these opcodes has a gas cost associated, related to how much computation it requires or storage it manages. Whenever someone tries to execute a transaction to a smart contract, it is necessary to provide a gas limit and a gas price in ETH or Gwei (1.0 ETH $= 10^9$ Gwei) as parameters for the transaction. A miner will execute the transaction until its completion or it runs out of gas, while the user will be debited by the amount of gas used multiplied by the current gas price. Gas is important to insure that the system will not get bogged down by a single contract execution, or be vulnerable to denial-of-service attacks, as well as functioning as a reward system for the miners. There is a gas limit for each block mined, which is voted by the miners; currently, it is 1.25×10^7. As there is a limit to the amount of gas, miners will give preference to transactions with a high price of gas.

Tezos is a third-generation blockchain that intends to address the cost, energy and scalability issues generated by the proof-of-work approach. It uses proof of stake as its consensus mechanism. Funded by the second largest Initial Coin Offering in 2017, Tezos is characterized by its self-amending properties and its proof-of-stake consensus mechanism. Tezos presents a particular case of proof of stake, in which the ability to produce blocks ("baking", in Tezos terminology) can be delegated to another entity, so the name "delegated proof of stake". Bakers do not need to perform work as in Ethereum, but rely on access rights linked to "coins" valued in XTZ. Tezos also has a different gas system than Ethereum. A user is charged for each transaction in two different ways: a fee, which is credited to the block baker, and a certain amount of burned coins, sent to an unreachable account. For performing a transaction, a user needs to provide a fee (in XTZ) and a gas limit; the transaction will then compete with other transactions to be added to a block, taking into account two limitations, namely hard block gas limit (10,400,000 gas) and hard operation gas limit (1,040,000 gas). Bakers then choose transactions, assuming that gas fits the block and fees respect a minimum. When the size of the blockchain storage increases due to a transaction, the sender must pay a "burn", in XTZ. This happens when a contract storage increases (storage burn) or when a new contract is put on the chain (allocation burn).

3 VCG for Sponsored Search Smart Contract

VCG for Sponsored Search (VCG) is a specialization of the Vickrey–Clarke–Groves auction mechanism dedicated to the sale of sponsored links. In this version, the goods being auctioned are "slots" in a web page, and the buyers are advertisers interested in putting one of their ads in a slot. Each slot is associated to a click-through rate ("ctr"), a measure of the number of clicks advertisers can be expected to receive on their ads per number of impressions.

The VCG algorithm, in which n bidders vie for k slots, each characterized with a ctr α_j, can be outlined as follows, where the ctrs α_j are assumed down sorted [8]:

1. accept a bid b_i from each bidder i, and relabel the bidders so that $b_1 \geq b_2 \geq \ldots \geq b_n$;
2. assign each bidder i of the k first bidders to the i-th slot (the others lose);
3. charge each such bidder a price $p_i = \frac{1}{\alpha_i} \sum_{j=i+1}^{k+1} b_j(\alpha_{j-1} - \alpha_j)$, with $\alpha_{k+1} = 0$.

Intuitively, the price (per click) p_i paid by the bidder i is designed to compensate the loss in "social welfare" suffered by all the other bidders by the mere presence of the bidder i. In the framework of search engines, such a VCG algorithm must be run each time a web page is about to be displayed on a computer, meaning billions of times per day.

Implementing VCG as a smart contract varies according to whether one targets Ethereum (Solidity) or Tezos (SmartPy). We strove to have similar code for both implementations to make the comparison as fair as possible, and use the Solidity version for reference here (the full implementation is available at `https://github.com/LucasMSg/VCG_SmartContracts`).

Storage

The following data structures are used to implement VCG (`unit` denotes unsigned integers):

- `owner(address)`, the Tezos address of the user who owns the auction smart contract;
- `isOpen(bool)`, a flag indicating if an auction is opened at the moment or not;
- `ctrs(uint[])`, the array of ctrs of the slots being auctioned;
- `bids(uint[])`, the bids sent by the advertisers;
- `agents(address[])`, the Tezos addresses of the advertisers;
- `prices(uint[])`, the prices computed at the end of the auction.

Public functions

Here are the main public functions (entry points, in SmartPy) of a VCG contract (only `bid` is not reserved to the contract owner):

- `transferOwnership` transfers the contract ownership;
- `updateCTRs` updates the `ctrs` array, if an auction is not under way;
- `openAuction` opens an auction, providing it an initial `ctrs` array argument;
- `bid` receives one bid from a participant (the bid and bidder's address are registered);
- `cancelAuction` cancels the auction (the bids and agents are erased);
- `closeAuction` closes the auction, sorts the bid list and computes the VCG `prices`.

4 Test Protocol

Both blockchain development environments provide editors to test contracts, but these tests are being simulated in a sandbox blockchain [2] and are thus not present in an actual blockchain. To provide performance results more representative of actual blockchains, we tested the VCG contract in so-called “testnets”, i.e., Ropsten for Ethereum and Delphinet for Tezos. This way we can expect to experience the behavior of a full-fledged blockchain without having to pay transaction fees. To deploy and communicate with contracts, we use Truffle, a development environment for smart contracts initially developed for Ethereum, but for which a Tezos integration, though still under development, presents enough functionalities for our tests. Truffle's contract abstraction provides means to interact with contracts using JavaScript.

We implemented a unit test that performs the following transactions on each blockchain: deployment of a VCG smart contract, opening of an auction, sending of bids (to simulate participants) and finally auction closure, producing a table of winners. We opted to deploy a new contract each time the test is performed to better track the possible cost incurred by the addition of more storage to a contract (e.g., the burned XTZ for Tezos). Our full test consists then of a series of unit test auctions with increasing numbers of participants and slots, namely 10 participants with 4 and 8 slots, 20 participants with 4, 8 and 16 slots and 50 participants with 4 slots. We stopped our tests after 50 participants and 4 slots because it was already enough to reach the gas limit of a Ropsten block. We note n_m an auction with n participants and m slots.

We focused the collection of test data on the most important factors that generally characterize the dynamic performance of contracts. Our use of Truffle also limited the scope of metrics we could put our hands on. For Ethereum, we measured gas usage, setting a large value for both the gas limit and the price to ensure that our transactions would be chosen by the miners and also have enough resources to run our contract to completion (in particular, when closing auctions). For Tezos, Truffle automatically sets the fee and gas limit; at the end of the test, we get the actual gas used and the number of burned coins for the execution of the test contract.

5 Results

Metrics selection

Our comparison experiment focuses on programmability (see next subsection), performance and cost issues. In view of the data available through Truffle and the blockchains' APIs, we selected gas and burned, blocktime and cost in dollars as metrics for our comparison's parameters. Gas and burned are used for computation (CPU, storage) performance assessment; they also illustrate how both platforms units of gas are not directly comparable. “Wall-clock” execution time could be considered as the time performance parameter, but when working with smart contacts, since it is directly linked to each blockchain's block time, we consider the latter as our time comparison parameter. Finally, monetary considerations are strongly linked to blockchain technologies, so we look at cost issues. Cost is, in some sense, a better parameter for comparison than the previous ones, since it is a good indicator of the practical usability of smart contracts and blockchains.

Programmability

An important though somewhat subjective point of comparison between the two blockchain environments is the ease of programming and interacting with smart contracts. Ethereum's main language for writing smart contracts is Solidity, similar to Java and C++. Ethereum provides a good IDE, Remix, with the possibility to execute transactions broken down by EVM op-codes and follow the changes in storage and gas consumption. On the other hand, Tezos offers four languages for smart contracts, including Michelson, the stack-based language that is, in the end, executed inside its blockchain. We opted for SmartPy, a Python library; scripts are then regular Python scripts that use SmartPy constructs. SmartPy relies on meta-programming, which may present a steeper learning curve for developers that Solidity.

For deployment and interaction with Ethereum contracts, we used Truffle, a well-documented tool; thanks to its many tutorials for setting up configurations, it poses few problems. Tezos, however, proved more difficult to put to use. We started with Tezster-CLI, a specific tool for Tezos's contracts, which happened to be not adaptable to our serial unit tests. We ended up switching to Truffle for Tezos, which, while still experimental, proved resilient enough for this experiment.

Gas and Burned

The experimental data obtained vary significantly according to the phase of the VCG auction process and the blockchain on which they run.

Deployment. On both blockchains, the gas for each deployment of a VCG contract is always the same. Ethereum consumes 1,016,192 gas, while Tezos needs 24,017 gas while charging 1.183 XTZ for the allocation of 4,475 bytes.

Opening. When opening an auction, the ctrs are stored in the blockchain. As can be expected, the gas and burned increase linearly with the number of slots being auctioned.

Bidding. Bids behave differently on each blockchain. Ethereum is more homogeneous, with the first `bid` transaction always needing more gas, since the first `push` sets up the storage for the array of bids and agents. The first bid consumes 105,917 gas, while the subsequent bids, having only to insert a `uint` and an address, always consume 75,917 gas.

For Tezos, gas consumption increases with a mean of 208.2 $\pm$ 1.1 (s.d.) with each subsequent bid, while the amount of coins burned is constantly 0.00925 XTZ or 0.0095 XTZ, depending on the size of the bid.

Closing. The close auction function/entry-point is the most relevant for our comparison, since the bulk of the VCG algorithm is performed here. The array of bids is sorted (we implemented a simple insertion-sort algorithm), and this sorted array is used in the third step of the VCG algorithm in order to compute the prices for the winners. Figure 1 is a graph of the closing gas for each of our tests. Note that it was not possible to close the 50 bids auction in Ethereum, the gas surpassing what the Ropsten network is accepting as gas limit for a single auction.

Block time

For Ethereum's main network, using Etherscan, we measured the block time at 14.82 $\pm$ 1.63 (in seconds), while the Ropsten network clocks at 14.5 $\pm$ 1.2 seconds. Measuring Ropsten directly from Truffle, we got a mean of 14.16 $\pm$ 7.72. For Tezos, its main network is advertised as providing a constant block time of 60 seconds, while Delphinet uses half of it, i.e., 30 seconds. Using Truffle, our tests on Tezos showed a block time of 43.07 $\pm$ 14.63 seconds. Note that we are not waiting for the suggested confirmation blocks.

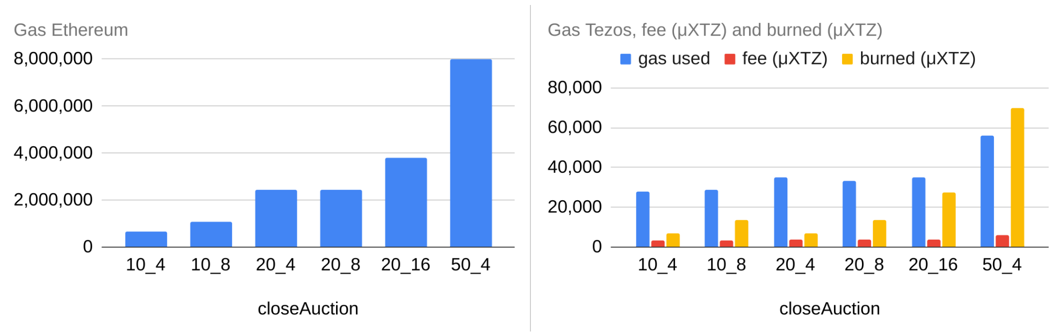

Figure 1 For each VCG contract n_m closing transaction, gas consumption on Ethereum (left) and gas, fee and burned for Tezos (right, where the Y axis scale is in gas and μXTZ).

Price

For our experiment, coins on testnets are free, so the ETH and XTZ amounts that were spent for this benchmarking had no actual value. Yet, an approximate prediction of the prices one would have to pay to run our VCG test can be obtained by taking the main network prices for both of these coins. At the time of this writing (Mar 10 2021, 10:24 UTC), one ETH is valued at $1,827, and one XTZ is $4.25.

For Ethereum, we used ETH Gas Station (`ethgasstation.info`) to get a quote for gas prices. For test purposes, we used the price category "Standard" (91 Gwei/gas, at the time of test), which led to the following prices for the deployment and bidding phases: $168.94, and $17.6 (first bid) and $12.6 (subsequent ones). The varying closing phase prices can be deduced from Figure 1; for instance, the price for a 10_4 auction was $128.26.

For Tezos, we used as transaction fees the ones automatically suggested by Truffle, while the burned costs, related to storage increments, are 0.00025 XTZ for 1 byte at the time of test, for both the main and testnets. The prices for the deployment phase are $0.03 for fee and $5.02 for burned. For the bidding phase, the fee paid by each bidder increases by $0.000088 ± 0.0000029 each time, while the burned remains somewhat constant, between $0.039 and $0.04. For the closing phase, one can refer to Figure 1 to get an estimate, where, for a 10_4 auction, the fee paid by the auctioneer would be $0.133 and the burned, $0.028.

6 Discussion

Our goal with this benchmarking study was to compare the performance of two very similar smart contracts on Ethereum and Tezos. Translating a Solidity contract to the Tezos blockchain environment proved to be quite difficult, even though this could be somewhat expected since Ethereum is the most popular dApps platform, with thus a lot of support from its community, while Tezos is much less used for now. From our experience, most complications with Tezos are inherent to its design philosophy. In particular, the self-amending property of this blockchain translates into testnets being abandoned every time there is a new protocol upgrade (every 4 months or so, based on our experience), which led to temporary complications for our study, either because of bugs or because some tools were not adapted to the new testnet as fast as expected.

Ethereum's scalability is a big drawback for our VCG implementation. The gas limit for blocks implies a very small limit for the number of bidders, especially when compared to standard VCG auctions in industry. Adding the system's popularity to the scalability problems is rising gas prices, which results in a high average transaction fee of $39.49 (recorded in February 23, 2021). These values could be considered acceptable for a transfer-value system, but, for a dApps platform, they could lead to users abandoning the system. However, the EIP 1559 [1] proposal to reform the Ethereum fee market and the introduction of a proof-of-stake approach within Ethereum 2.0 are two welcoming changes that could positively impact the Ethereum results in our benchmarking.

The idea of implementing VCG as a smart contract, though initially appealing due to the archival nature of blockchains and the transparency of its data processing, had some less positive implications. The main one is that all data in a public blockchain is public, which goes against the sealed-bid requirement of VCG. We intend to address this issue in the future, via the inclusion of cryptography contracts similar to [5]. But even if one assumes that bidders are not able to access the blockchain to see the other bids, the bid transaction receipt automatically returned by Ethereum and Tezos could still be used to inform the bidder about the current status of the auction, since, for instance, a big gas consumption for Ethereum means that one has been the first bidder, while the always increasing prices for bids in Tezos can help subsequent bidders in figuring out better strategies.

Another hindrance of blockchains for auctions is the total time being linked to the block time. In an actual VCG for sponsored search setting, auctions are made in matters of seconds, which means that a smart contract is not viable for such an application, except maybe in very limited domains (high-value auctioned items among few participants, for instance as in a country-level energy market).

7 Conclusion

We present a comparative bench-marking use case for smart contracts on the proof-of-work Ethereum and proof-of-stake Tezos blockchains. Our test is based on the VCG auction mechanism widely used in the search-engine industry for advertisement placement, an application that might be thought to be able to profit from the trust and good governance practices blockchains bring to computations. Our experimental data suggest however that, currently, time and space performances (and price, mostly on Ethereum) prevent this type of application to be put most of the time to practical use.

References

1 Ethereum improvement proposal 1559. URL: `github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md`.

2 Ethereum. *Remix Ethereum IDE Deploy & Run.* URL: `remix-ide.readthedocs.io/en/latest/run.html`.

3 L.M Goodman. Tezos — a self-amending crypto-ledger.

4 Daniel Larimer. Eos.io technical white paper v2. 2018.

5 Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. 2017. `doi:10.1007/978-3-319-70972-7_20`.

6 Daniel Perez, Jiahua Xu, and Benjamin Livshits. Revisiting transactional statistics of high-scalability blockchains. 2020. URL: `https://arxiv.org/pdf/2003.02693.pdf%C3%82%C2%A0`.

7 Michel Rauchs, Apolline Blandin, Keith Bear, and Stephen McKeon. 2nd global entreprise blockchain benchmarking study. 2019. URL: `https://www.jbs.cam.ac.uk/wp-content/uploads/2020/08/2019-10-ccaf-second-global-enterprise-blockchain-report.pdf`.

8 Tim Roughgarden. *Twenty Lectures on Algorithmic Game Theory.* Cambridge U. Press, 2016.

9 Tezos. *Tezos Proof of stake.* URL: `tezos.gitlab.io/008/proof_of_stake.html`.

10 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger.