Annals of Emerging Technologies in Computing (AETiC)
Vol. 3, No. 5, 2019

Review Article

A Survey on Efficient Parallelization of Blockchain-based Smart Contracts

Alessio Meneghetti, Tommaso Parise, Massimiliano Sala, Daniele Taufer*

Department of Mathematics, University of Trento, Trento, Italy
alessio.meneghetti@unitn.it; tommaso.parise@studenti.unitn.it; maxsalacodes@gmail.com; daniele.taufer@unitn.it
*Correspondence: daniele.taufer@unitn.it

Received: 8th November 2019; Accepted: 24th November 2019; Published: 15th December 2019

Abstract: The main problem faced by smart contract platforms is the amount of time and computational power required to reach consensus. In a classical blockchain model, each operation is in fact performed by each node, both to update the status and to validate the results of the calculations performed by others. In this short survey we sketch some state-of-the-art approaches to obtain an efficient and scalable computation of smart contracts. Particular emphasis is given to sharding, a promising method that allows parallelization and therefore a more efficient management of the computational resources of the network.

Keywords: ***Blockchain (BC); Bitcoin; Crypto-currency; IoT; Proof of Work (PoW); Distributed Digital Ledger***

---

1. Introduction

Since the release of Bitcoin in 2008 [1] the blockchain technology has gained popularity and gathered the interest of many experts of different fields. Blockchain-based decentralization introduced a new paradigm in data management and reliable computing, which allows a wide range of applications such as smart contracts. This concept, which can be traced back to 1985 [2] and that was repeatedly refined in the following years [3], has found in the blockchain environment a natural setting to achieve automation of multi-step processes [4, 5], security [6] and privacy preservation [7].

One of the drawbacks of such a decentralized and public management of data is the huge amount of computational power required to reach consensus. This requirement heavily affects the network efficiency, notably when the number of nodes rises. To provide a real-life example, the main cryptocurrencies Bitcoin and Ethereum support few transactions per second (~7 and ~20 resp.) compared to those of a centralized payment circuit such as Visa (~2000).

To solve this issue much research has been lately carried on lately in different directions. On one side, there have been many attempts to improve current blockchain models, possibly preserving their smart contract capability [8, 9, 10]. On the other side, many new types of blockchains have been proposed [11, 12, 13, 14, 15, 16, 17, 18] but the research is still far from complete.

In this short survey we briefly report the proposals that we consider the most promising, with a special focus on sharding-like models.

In Section 2 we present some known methods to address the scalability problem, while Section 3 is devoted to examine novel ideas exploited by existing architectures. Finally, in Section 4 we summarize the discussed topics, highlighting those that we consider more fruitful.

Alessio Meneghetti, Tommaso Parise, Massimiliano Sala and Daniele Taufer, "A Survey on Efficient Parallelization of Blockchain-based Smart Contracts", Annals of Emerging Technologies in Computing (AETiC), Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 9-16, Vol. 3, No. 5, 15th December 2019, Published by International Association of Educators and Researchers (IAER), DOI: 10.33166/AETiC.2019.05.002, Available: http://aetic.theiaer.org/archive/v3/v3n5/p2.html.

2. The scalability problem

Any blockchain-based platform aiming at obtaining a world-wide diffusion needs to efficiently scale with the increasing number of users and transactions, while guaranteeing a feasible consumption of resources such as energy, computational power and allocated memory. Many solutions have been proposed to achieve scalability, which we discuss in this section. In our opinion only some of them are considered suitable for smart contracts execution, as we elaborate below.

SegWit: It consists of moving transactions signatures to a side chain, referred by a Merkle root in the main chain [19]. Since the signature data takes up to 70% of the transaction data, this operation allows to include more transactions per block. SegWit lowers transaction fees but increases their computational cost.

Block size increase: An increment in block size allows more transactions per block and therefore it reduces fees. However, this solution might not fit well smart contract platforms, such as Ethereum, because larger blocks may increase consensus time.

DAG: the classical blockchain architecture is extended with a more sophisticated one, based on Directed Acyclic Graphs. This technology leads to parallel validation of transactions, while increasing the implementation complexity.

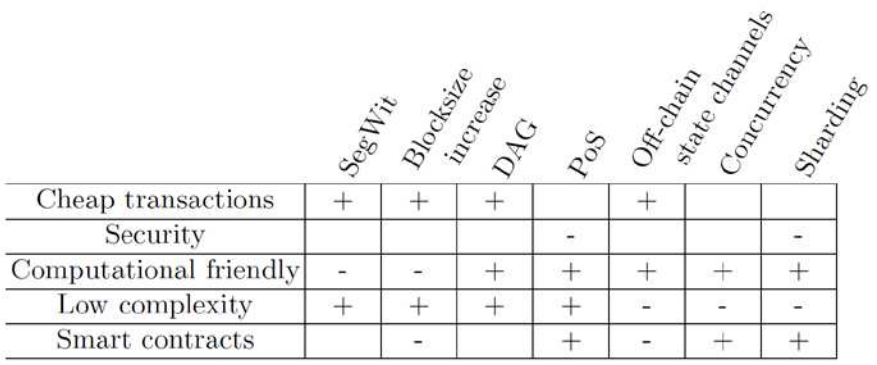
Proof of Stake (PoS): It is a consensus algorithm in which the probability of being elected as block producers depends on a "stake" (usually, wealth or age). Whilst having the advantage of being low energy-consumptive, the comparative security of PoS w.r.t. PoW is questioned [20, 21]. Similar properties are shared by alternative consensus algorithms (e.g., DPoS, PoA).

Off-chain state channels: The use of these channels [22] decreases transaction time exponentially, since most transactions are no longer dependent on someone else (e.g. a miner) to be validated. Indeed, participants can sign some transactions without notifying the entire blockchain. Therefore, nodes may benefit from private and instant transactions with lower fees. However, to perform smart contracts or multi-signatures, a segment of the blockchain state has to be locked and agreed by a set of participants, making smart contracts more difficult to implement.

Concurrency: As in software transactional memory [9], each miner can execute contract codes as speculative actions that are regulated by a specifically-built virtual machine, which assigns different "abstract locks" to primitive commuting operations on state variables, allowing different threads to work on distinct locks in parallel. Although in principle this approach promises to be extremely efficient [11], it is very difficult to implement.

Sharding: It consists of breaking the blockchain into small parts that are managed in parallel by node subsets, called shards. This speeds up the consensus process and augments throughput, since many transactions can be simultaneously validated. Inter shard communication has to be managed by its own protocols, forcing developers to handle an additional level in their code. We also note that shards are easier to be corrupted than the whole chain, possibly threatening the system security.

Remark: although all solutions have merits, in terms of applicability to smart contract-based platforms we recommend PoS, concurrency and sharding. In particular, concurrency and sharding allow efficient and scalable computation of smart contracts, but concurrency is very hard to implement.



	SegWit	Blocksize increase	DAG	PoS	Off-chain state channels	Concurrency	Sharding
Cheap transactions	+	+	+		+		
Security				-			-
Computational friendly	-	-	+	+	+	+	+
Low complexity	+	+	+	+	-	-	-
Smart contracts		-		+	-	+	+

Figure 1. Comparison between scalability solutions

3. Architectures

We now review the main blockchain architectures that aim at solving the scalability problem for the smart contract execution, employing the techniques discussed in the previous section.

3.1. Ethereum

Ethereum's development team plans to improve the platform both by replacing the current PoW-based consensus algorithm with the PoS-based Casper protocol and by implementing sharding [4]. This new architecture will comprise two main components: the "beacon chain", which will host the main consensus mechanism, and the "shard chains", multiple blockchains storing account data and transactions.

Participants to the new PoS-based consensus will be required to deposit 32 ETH's into the beacon chain, becoming *validators*. They will appear in a registry, stored and managed by the beacon chain, and take part in the consensus protocol.

Validators are pseudo-randomly divided into committees, which are active parts in the beacon chain consensus and attest for a certain shard chain. The flowing of time is marked by *slots*, 6 seconds windows in which a validator, called *proposer*, can create a beacon chain block while the others, called "attesters", may perform Attestations, which are used to validate shard blocks and to reach consensus on beacon chain blocks.

A set of consecutive slots during which every validator has had the opportunity to make exactly an attestation is called an *epoch*.

If a certain shard block reaches a sufficient number of attestations, it will create a "crosslink". A crosslink is a set of signatures from a committee attesting to a block in a shard chain, which can be included in the beacon chain. When a crosslink is included in the beacon chain block, the corresponding shard chain fragment is considered confirmed. Crosslinks can also be used as a mean for asynchronous cross-shard communication.

As previously mentioned, the shard chains are where all the account data will be stored. Transactions will be managed inside the shards and added to the shard blocks. Shard state executions have still not been precisely defined, since presently the effort of the community is on providing construction, validity and consensus for the data of shard chains.

3.2. Zilliqa

Zilliqa [17] is a blockchain platform designed to securely scale using sharding. Its network comprises shards and a Directory Service (DS) committee, where the role of each node is assigned by PoW. To be part of the network, a node periodically solves a PoW puzzle, which is validated by the existing nodes. Afterwards, the DS committee is elected through a PoW competition. At regular intervals the oldest member of the committee is pushed out and a new participant is elected. Finally, all the remaining nodes participate in another PoW-based contest, validated by the committee, to be assigned to shards.

Shards have a minimum size of 600 nodes in order to ensure that the probability of having 1/3 of malicious nodes is below 10^{-6} [14]. Shards work on distinct states, which together compose a "Global state". Each new transaction is assigned to a single shard, so that different shards validate distinct set of transactions obtaining an overall improvement in throughput.

Transactions are assigned to shards based on the sender address, so that transactions from the same address are processed by the same shard. Double spending is avoided by using nonces: every time an address sends a transaction, the nonce in the account and the global state are updated. A nonce lower or equal to the current global state causes the immediate rejection of the transaction.

Within each shard, Zilliqa employs Practical Byzantine Fault Tolerance (PBFT) to reach consensus [23]. In PBFT a piece of record, proposed by a "leader", is valid as soon as a supermajority of nodes in the shard agreed on it.

Once a shard has reached consensus on a block of transactions, the block is sent to the DS committee. The DS leader assembles blocks with the received transactions and runs a final consensus round. Once the committee agrees on the final block, its header and signature are sent to the shards that proceed to recompose the block data, add it to their local blockchain and update the global state.

PBFT is not computationally intensive: once identities are established, the network can proceed to build several blocks before a new PoW is needed. The main issue related to PBFT is that the efficiency of the protocol falls as the number of nodes grows, since every node has to communicate with every other node of the same shard.

3.3. EOS

The EOS infrastructure [13] was designed specifically for being a blockchain platform with limited transaction fees and demand of computational capacity. Its approach is to extend the large-scale high-performance blockchain experiences of Bitshares [24] and Steem [25], reassembling their components to build distributed applications.

First, EOS switches from the classic view of a deterministic state machine described by a state to a description based on events, called "Actions". The reconstruction of the state is obtained by using Actions as aid, so that users are able to verify transactions and updating the state more efficiently.

The main strength of this project is the usage of Distributed Proof of Stake (DPOS) as consensus method. There is a two-level structure of decision/governance: in the first level the actual PoS decision is taken by a restricted group of 21 producers, while all the users who have "stakes" on the chain participate, in the second level, in the election of the producers. With this system if a node of the producers is thought to be behaving incorrectly will be voted out of the producers from the stakeholders.

The production of blocks occurs every 0.5s in round of 126 (6 blocks per producer at every round) and when a producer fails to propose a block it is excluded from block creation for the rest of the round, to minimize missed blocks. A Byzantine Fault Layer is added on top to ensure finality, and once 15 out of the 21 block producers have signed a block as valid, it is considered irreversible.

Furthermore, EOS uses Transaction as a Proof of Stake (TAPOS), that means that in every transaction there must be the hash of the recent block header, preventing replay attacks on forks not including the reference block. A side effect of this process is security against long-range attacks that attempt to generate alternative chains. Individual stakeholders directly confirm the blockchain every time they add a transaction. Over time all blocks are confirmed by stakeholders and this is something that cannot be easily replicated in a forged chain.

The first release of EOS is single threaded but the data structure to support parallel execution has already been included. The idea would be to apply a sharding-like division of work when processing blocks. The block producer will organize the transaction/action delivery into independent shards so that evaluation can happen in parallel. Every block will be divided into cycles, every cycle will be divided into shards, which will contain transactions and so actions. The resulting structure is a tree with alternating layers that require sequential (cycles, transactions, actions) or parallel (shards, account updating) executions.

3.4. RapidChain

RapidChain [18] is a recent research project whose purpose is to use sharding for solving performance and security issues of the Bitcoin model. The structure is similar to Zilliqa's: the main committee is created and it is responsible of shard assignment, transaction distribution, randomness generation and driving periodic reconfiguration events. Then every shard ("sharding committee") manages autonomously the transactions received, keeping their own ledger.

Time is divided into *epochs* and each of them has a unique randomness seed that may be used in a PoW to obtain identity in the network.

The members of every shard batch together the received transactions (distributed in a deterministic way) into a big (~2MB) block. Before the block can be appended, the committee has to validate every transaction in the block, communicating with the corresponding shard committees to ensure the input transactions exist in their shards.

Then inside shards, by using the epoch randomness, at every iteration the committee elects a leader. The leader "gossips" the created block to every member of the shard, i.e. it broadcasts the block using an information dispersal algorithm. This amounts to breaking the block into pieces, which are encoded with an erasure code scheme and sent to a different node in the committee that proceeds to send its part to neighbor nodes, and so on. Afterwards, the leader computes the block header, which contains the iteration number and the Merkle root computed from the block pieces obtained above. Finally, the leader starts the consensus protocol using the computed header, reaching consensus on the transmitted block and adding it to the shard ledger.

At the end of every epoch the network undergoes a reconfiguration phase. The main committee publishes a reconfiguration block containing the new epoch randomness and a new list of participants and their committee memberships. This protects the network from join-leave attacks and corrupted nodes.

3.5. Blockclique

Another line of work exploits DAG-based chains to reduce the consensus demand, as for example in SPECTRE [16] and IOTA [26]. Their transactions are not required to be sharded as incompatibility issues are solved either by an extra voting process or by a centralized node, respectively.

A more recent work shows that this approach, which has been designed to avoid sharding, may work well even paired with it. This model, called *blockclique* [27], prescribes a division of nodes into "threads", which work in parallel, producing blocks that are organized in a DAG. This graph has to respect two internal rules: the blocks produced by a single thread constitute a proper blockchain ("thread incompatibility") and new blocks take into accounts blocks recently produced by other threads ("grandpa incompatibility") by referring to their parent's hashes.

New transactions are divided among threads depending on their input addresses, dodging double spend, while the consensus rule ensures by construction the compatibility of valid transactions from different threads. This consensus may be achieved by looking at the DAG structure and is autonomously obtained by each node determining the best compatible block clique, i.e. the one of maximum work sum. In order to efficiently compute this clique the balance of each address is stored by each node and the graph is kept updated with only the most recent blocks. Specifically, a block is removed either if it belongs only to cliques of small size or if it has had many descendants inside maximal cliques. Both these conditions are regulated by a "finality" parameter, which is responsible for the trade-off between security and efficiency.

Simulations suggest that the architecture parameters may be chosen so that it profitably scales to realistic network assumptions without a significant loss in performance. However, this model imposes strong constraints on the types of smart contracts that can be processed, such as storing data of old blocks and resolving potential incompatibilities caused by the multithreaded environment.

3.6. Quorum

Quorum [15] is a permissioned blockchain, in which only pre-approved node can take part, designed for financial use cases. Quorum blockchain is a fork of Ethereum, which focuses on increased privacy for smart-contract execution.

From a smart-contract perspective, the novelty of Quorum is the division of transactions in two types: private and public, jointly with off-chain communication. Public transactions act like the usual transactions on Ethereum, while private transactions contain only digests of off-chain communication between nodes, that perform some smart contract in private. Both the public and private transaction are recorded in the same public blockchain, while the off-chain communication remains hidden. However, the peers who have and can decrypt the off-chain "transactions" will actually execute the relevant contracts. Thanks to the digests present in the private transactions, the integrity of off-chain "transactions" is guaranteed.

In addition to the public state, every node hosts a private state Patricia Merkle trie (a special type of Merkle tree). Off-chain transactions affect the private state, while public transactions affect the public trie.

Other interesting approaches, which would deserve a separate study, are listed below:

RSCoin [28], which works in a permissioned network, Elastico [29], which claims an almost linear scaling of transaction throughput w.r.t. computational power, Chainspace [30], which avoids global replications of state.

4. Conclusions

Introducing parallelism in a blockchain-based infrastructure is a challenging task. If we want the blockchain also to support execution of smart contracts, the task is so difficult that no practical solution has clearly emerged so far. Solutions tackling only the *block formation*, such as SegWit, DAG or consensus algorithms, do not show, in our opinion, a clear potential to address the parallel execution of smart contracts in an efficient and secure way. This goal may actually be achieved, at least in principle, by approaches that affect more drastically the *state management*. Among these, we see two promising approaches. The first yields a clear theoretically advantage, since it adapts, to a blockchain-based system, well-established techniques investigated in concurrency theory, such as abstract locks. Unfortunately, at the moment we do not have evidence that such an approach can be implemented in practice on a large scale. The second is sharding, which is less ambitious but better tailored to a blockchain environment.

Some other systems are promising, but with questionable feasibility in a real-world scenario, such as Ziliqa, RapidChain, Blockclique and Quorum.

Considering the architectures sketched in this short survey, we consider promising:

- the planned improvement of ETH, where sharding and PoS will be used jointly,
- the announced new releases of EOS,

Since the structure complexity is not increased dramatically and they adopt a sharding more suitable for a blockchain-based platform aiming at executing smart contracts.

Acknowledgments

This work has been carried on within the EU-ESF activities, call “PON Ricerca e Innovazione 2014-2020”, project “Distributed Ledgers for Secure Open Communities”. The authors would also like to thank the Quadrans Foundation for its support.

References

[1] S. Nakamoto, Bitcoin: "A Peer-to-Peer Electronic Cash System", 2008, Available: https://bitcoin.org/bitcoin.pdf

[2] N. H. Minsky, A. D. Lockman, "Ensuring integrity by adding obligations to privileges", IEEE Computer Society Press, 1985, Available: https://dl.acm.org/citation.cfm?id=319589

[3] N. Szabo, "Formalizing and Securing Relationships on Public Networks", First Monday, 1997, Available: https://ojphi.org/ojs/index.php/fm/article/view/548/469

[4] V. Buterin, "A next-generation smart contract and decentralized application platform", 2014, Available: http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[5] K. Christidis, M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things", 2016, Available: https://ieeexplore.ieee.org/document/7467408

[6] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, P. Saxena, "A Secure Sharding Protocol For Open Blockchains", CCS ‘16 Proceedings, 2016, Available: https://dl.acm.org/citation.cfm?doid=2976749.2978389

[7] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts", 2016 IEEE Symposium on Security and Privacy, 2016, Available: https://ieeexplore.ieee.org/document/7546538

[8] K. Croman et al, "On Scaling Decentralized Blockchains (a position paper)", 2016, Available: https://www.initc3.org/files/Scaling2016.pdf

[9] T. Dickerson, P. Gazzillo, M. Herlihy, E. Koskinen, "Adding Concurrency to Smart Contracts", 2017, Available: https://dl.acm.org/citation.cfm?doid=3087801.3087835

[10] A. Van Moorsel, "Blockchain Based Smart Contracts: A Systematic Mapping Study", IEEE, 2018, Available: https://ieeexplore.ieee.org/document/8756390

[11] E. Eykholt, L. G. Meredith, J. Denman, "RChain Architecture Documentation", 2017, Available: https://media.readthedocs.org/pdf/rchain-architecture/stable/rchain-architecture.pdf

[12] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gaill, E. Syta, B. Ford, "OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding", IACR Cryptology ePrint Archive, 2017, Available: https://eprint.iacr.org/2017/406.pdf

[13] D. Larimer et al., "EOS.IO Technical White Paper v2", 2017, Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[14] L. Luu, D. Chu, H. Olickel, P. Saxena, A. Hobor, "Making Smart Contracts Smarter", IACR, 2016, Available: https://eprint.iacr.org/2016/633.pdf

[15] The Quorum Team, "Quorum White Paper", Available: https://github.com/jpmorganchase/quorum/

[16] Y. Sompolinsky, Y. Lewenberg, A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol", IACR Cryptology ePrint Archive, 2016, Available: https://www.cs.huji.ac.il/~yoni_sompo/pubs/16/SPECTRE_complete.pdf

[17] The Zilliqa team, "The ZILLIQA Technical Whitepaper", 2017, Available: https://docs.zilliqa.com/whitepaper.pdf

[18] M. Zamani, M. Movahedi, M. Raykova, "RapidChain: Scaling Blockchain via Full Sharding", IACR, 2018, Available: https://eprint.iacr.org/2018/460.pdf

[19] E. Lombrozo, J. Lau, P. Wuille, "Segregated Witness (Consensus layer)", Available: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

[20] A. Kiayias, "Proving the security of blockchain protocols: from PoW to PoS", 2017, Available: https://blockchain-summer.epfl.ch/talks/blockchain-security-kiayias.pdf

[21] W. Li, S. Andreina, J. Bohli, G. Karame, "Securing Proof-of-Stake Blockchain Protocols", Data Privacy Management, Cryptocurrencies and Blockchain Technology, Springer, 2017, Available: https://link.springer.com/chapter/10.1007/978-3-319-67816-0_17

[22] J. Poon, T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments", 2016, Available: https://lightning.network/lightning-network-paper.pdf

[23] M. Castro, B. Liskov, "Practical Byzantine fault tolerance", Proceeding OSDI '99, 1999, Available: http://pmg.csail.mit.edu/papers/osdi99.pdf

[24] Bitshare Team, "Bitshare White Paper", Available: https://bitshares.org

[25] The Steem Team, "Steem White Paper", Available: https://steem.com/steem-whitepaper.pdf

[26] S. Popov, "The Tangle, Version 1.4.3", 2018, Available: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf

[27] S. Forestier, D. Vodenicarevic, "Blockclique: Scaling Blockchains through Transaction Sharding in a Multithreaded Block Graph", ArXiv, 2018, Available: https://arxiv.org/abs/1803.09029

[28] George Danezis, Sarah Meiklejohn, "Centrally Banked Cryptocurrencies", arxiv: 5005.06895, 2015

[29] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, Prateek Saxena, "A Secure Sharding Protocol For Open Blockchains", Proceedings 2016 ACM SIGSAC.

[30] Mustafa Al-Bassam, Alberto Sonnino , Shehar Bano , Dave Hrycyszyn and George Danezis, "Chainspace: A Sharded Smart Contracts Platform", arxiv: 1708.03778, 2017.