CONTENTS

Very Short Primer on Blockchain Technology for Database Researchers

Zsolt István
IT University of Copenhagen, Denmark
zsis@itu.dk

ABSTRACT

Blockchain is an emerging technology, considered increasingly often beyond the cryptocurrency world for business-to-business use-cases. In contrast to public blockchains such as Bitcoin, that are open systems in which anyone can participate, in business-to-business scenarios the membership of the service is controlled (permissioned blockchain). This permits the use of Byzantine fault tolerant (BFT) consensus protocols at the core of the service to establish a total order of transactions, instead of the more expensive Proof-of-Work-based consensus protocols. Permissioned blockchains typically set out to solve problems in the space where databases have traditionally resided, with the main difference being that the former decentralizes trust. There are numerous research proposals in the intersection of databases and blockchains. Sadly, there are still many misconceptions about this technology which leads to confusion in the community. The main goal of this primer is to give an overview of the relevant topics and provide pointers for further reading.

1 BLOCKCHAIN BASICS

Blockchains have entered the "spotlight" after the seminal Bitcoin paper published by Nakamoto [11] more than a decade ago. Even though many think of Blockchains as being part of the data management field, the Nakamoto paper was addressing a financial issue: its goal was to provide a decentralized currency that does not require trust in a central authority for its functioning. It has been only later that discussions emerged about "transaction processing", that is, smart contracts on top of blockchains, perhaps most notably, in the Ethereum White Paper [4].

In the years since their proposal, Blockchains have lived through a hype and this resulted in a dizzying number of different systems. At their core, however, all blockchains are quite similar. They aim to *decentralize trust* and are composed of two parts: *1) a verifiable data structure* that allows participants to determine whether the data in the blockchain has been tampered with and *2) a consensus algorithm* that defines how participants can add new data to the data structure. The choice for data structure is typically an append-only log with cryptographic hashes linking entries (i.e., blocks, see Figure 1). Nonetheless, there are also blockchains, such as IOTA, that order data into a directed acyclic graph (DAG) [10] instead. Put in database terms, we can think of the former as establishing global total order on all transactions and keeping a single logical shard of the data, and the latter as data sharding with total order within a shard and infrequent cross-shard transactions.

The choice of the consensus algorithm is determined by the assumptions the blockchain makes on the trustworthiness of third parties and the participation model. We can split blockchains into two categories based on who is allowed to participate in the consensus algorithm and, as a result, add data to the data structure: First, *permissionless blockchains* where anyone is allowed to participate and, second, *permissioned blockchains* where only selected entities can participate (this also assumes that the nodes in the system have verifiable identities, which is not the case in permissionless blockchains).

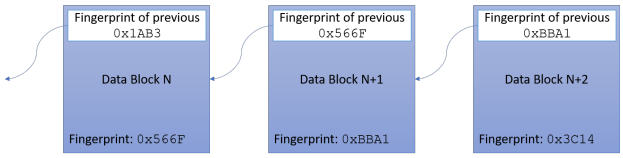

Figure 1: At the core of all blockchains is a verifiable data structure, typically in the form of an append-only log. Cryptographic hashes (fingerprints) included in blocks recursively tie them to the previous blocks.

Permissionless blockchains are, by definition, *public*, since anyone can access the data structure and read all previous transactions. Permissioned blockchains can be either *private*, in which case the governing consortium or members of the blockchain restrict access based on internal rules (e.g., business partners in specific areas), or they can be *public*, in which case anyone fulfilling some non-restrictive condition can join (e.g., Alastria[1] implements a country-wide blockchain network that relies on national tax register numbers as a pre-condition to joining). It is important to note that, while permissionless blockchains are typically used for dealing with fully virtual assets, permissioned blockchains are more similar to databases, in that the data they store typically represents real-world assets and events.

What is there for database researchers to explore? In the permissionless blockchain space most current and future challenges are related to scalability of the network, to economic incentives, as well as, to consensus algorithms with better guarantees for tolerating malicious actors. In terms of data management and transaction processing, there is no clear need for inventing new approaches because these systems can benefit from already existing best practices. In contrast, permissioned blockchains are considered for use-cases which are much closer to the database community, such as supply-chain management, banking or notaries, and therefore face challenges closer in nature to the database community. As we will discuss in the last section, there is significant overlap in database and blockchain research, with many database ideas/techniques being well applicable in the blockchain space. The fundamental difference, however, between permissioned blockchains and traditional distributed databases is the decentralization of trust in the former, that is, the assumption that not all nodes belong to the same enterprise and the ability to function even if not all members of a consortium are trusting each other. From a database research perspective, hence, most

[1] https://alastria.io/en/

of the interesting future work is in understanding how the performance of transaction processing (and perhaps analytics) can be increased while benefiting from the trust decentralization of blockchains. Additionally, the standardization of smart contracts and data models, as well as their integration with traditional databases is an open challenge.

2 PERMISSIONLESS BLOCKCHAINS

In permissionless blockchains there are no node identities and, as a result, a consensus mechanism is required that cannot be subverted by creating a large number of "fake" nodes to achieve majority in the system (Sybill attack). Therefore, Bitcoin and many other cryptocurrencies use *Proof of Work* (PoW) consensus, that requires participants to the consensus to solve a cryptographic puzzle before they can add to the log. The puzzle consists of finding a hash value that fulfills a specific condition, for instance, a number of leading zeros. The input to the hashing is the block that the participant would like to record on the blockchain. Inside the block are 1) transactions collected from clients of the system and other participants, 2) the cryptographic hash (signature) of the previous block in the blockchain and 3) a field holding a random number. The randomness is added to the block before being hashed, and unless the resulting hash fulfills the condition, the last step is repeated (this is the process of *mining*). Finally, when a miner discovers a combination for which the condition holds, it appends it to the blockchain by broadcasting it to all other participants. These can *validate* whether the added block is legitimate simply by computing its cryptographic hash and checking whether this hash fulfills the condition[2]. Hence, even though mining is very expensive, clearly limiting the throughput of the system, validation itself is cheap. In exchange for "finding" the next block of the blockchain, the miner will receive a payment in the underlying cryptocurrency, typically deducted as fees from the transactions in the just-appended block.

While PoW consensus has benefits in that it limits participants' power in the blockchain to their relative compute power (which is much more costly to increase than creating new IP addresses for instance), it has also many drawbacks. Apart from the obvious energy efficiency concerns resulting from the repeated hashing computations and the issue that all miners are fundamentally competing with each other for the next block, it has an important implication on *transaction finality*. In systems relying on PoW, competing miners could create forks in the log, working towards two "alternate realities". In practice, blocks are considered committed once enough additional blocks (e.g., 5) have been added after them. The expectation is that, at that point, the economic incentives are driving all participants to continue building on the longest subchain of the system. Nonetheless, transactions never reach *finality* because there is a small probability that any block, no matter how old, could have been forked and it does not, in fact, lie on the longest subchain.

It has to be noted that the common practice of composing *blocks* from a large number of transactions is rooted in the space of PoW blockchains in an effort to amortize the exuberant cost of mining, as well as, to simplify global broadcasts. Overall, the idea of a blockchain and its underlying mechanisms could just as well work for appending single transactions into a tamper-proof log, albeit, with much less efficiency for PoW consensus-based blockchains, such as Bitcoin.

To overcome the fundamental efficiency issues of PoW, other forms of consensus are becoming wide-spread, most notably, variations of Proof of Stake (PoS), e.g., in Tezos, Peercoin, and in newer iterations of Ethereum. PoS consensus is, at its core, a majority-based consensus algorithm that requires the voting participants to stake part of their cryptocurrency holding behind each consensus round. In case irregularities happen, e.g., forks, or conflicts, they are liable to lose their stake. This creates a strong economic incentive for participants to adhere to the rules.

In addition to PoS, there are other exciting proposals, such as Proof of Storage, Proof of Elapsed Time and Proof of Personhood, to name only a few (see more in this survey [14]). In addition to consensus protocols that provide a "proof" of owning some information or property, there are other proposals that embrace stochastic behavior and can even tolerate 51% attacks temporarily [13].

Even though non-PoW consensus solves many of the efficiency issues of PoW blockchains and, in principle, allows for lower latencies, these blockchains still typically suffer from the lack of finality in transactions: their throughput and latency are not determined only by the choice of underlying consensus protocol but also by the economic assumptions the blockchain makes.

3 PERMISSIONED BLOCKCHAINS

Permissioned Blockchains, such as Hyperledger Fabric [2], Corda R3 [3] and IOTA [12], introduce the requirement for a mechanism to associate identities with participants of the blockchain. Identities could be either issued by a trusted third party or by a consortium of the blockchain nodes themselves. Identities allow the use of more traditional, and significantly cheaper, forms of consensus algorithms, typically those from the family of Byzantine Fault Tolerant (BFT) algorithms (e.g., PBFT [5]), to append to the shared data structure. Using BFT consensus has the important benefit that transactions can be committed with *finality*. As a result, these permissioned systems can reach latencies and throughputs more similar to those of traditional databases. It has to be noted, however, that many of these systems run in widely geodistributed setups without the availability of dedicated, high bandwidth, networking, resulting in lower performance than what we typically see in RDBMSs.

In addition to the limitations imposed by networking bottlenecks, many permissioned blockchains inherit other limitations from the PoW blockchain space. They often rely, for instance, on batching a large number of transactions into blocks, which unsurprisingly results in artificially high latencies. These limitations, however, are not fundamental. We have demonstrated, as an example, that by rethinking the block-based processing of Hyperledger Fabric [9], its latency can be lowered from the hundreds of milliseconds to the millisecond range without negatively impacting its throughput or requiring significant changes to its code base. Therefore, it is reasonable to assume that permissioned blockchains that are being deployed in production systems will eventually "shed" the most obvious inefficiencies.

Given the decades-long of research in distributed consensus and fault tolerant systems, it is reasonable to predict that the work of database researchers will not necessarily be most useful in improving consensus but, instead, in enhancing the data management and data processing aspects of these systems. For instance it is an open challenge how to make Smart Contract

[2]Depending on the nature of the transactions in a block, of course, additional checks might be required that ensure that the underlying state remains consistent at all times.

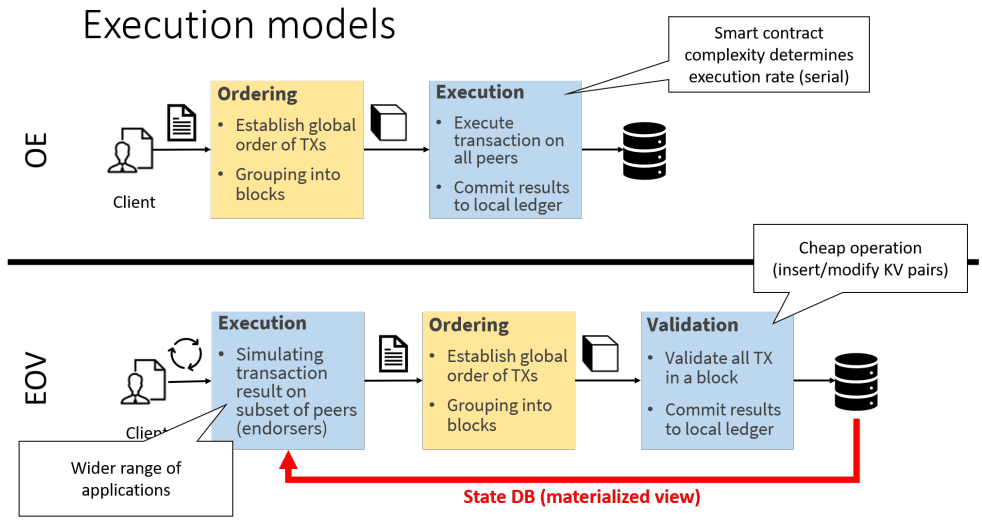

Figure 2: Blockchains can be classified into two groups depending on their smart contract execution model (nomenclature might differ across systems)

execution more efficient and integrated with RDBMs, which, in some cases, might have different schemas at different companies.

4 SMART CONTRACT EXECUTION

Most practical solutions for Smart Contracts (e.g., in Ethereum, Tezos, Fabric, etc.) treat the "contents" of the blockchain as a multi-versioned key-value store. As such, it is possible to think of smart contracts as small programs that can only interact with a key-value interface.

Blockchains adopt one of two execution models: *Order-Execute* (OE), which is similar to active replication in database terminology, and *Execute-Order-Validate* (EOV), which can be thought of as passive replication. As shown in Figure 2, the OE model first establishes the total order of transactions (that is, the order of smart contract invocations), then broadcasts these to all participants, each participant executing the transactions locally, typically in a serial manner. For this approach to work, smart contracts cannot include non-deterministic operations, otherwise the results could diverge. Blockchains using the OE model ensure this by relying on carefully designed DSLs for writing smart contracts.

Systems implementing the EOV model start by simulating (also called executing or endorsing) smart contracts on a subset of the nodes, chosen by a user-defined policy. The results of these executions are recorded in a terms of read/write modifications they would perform on the key-value store (state DB). Once the client receives enough simulation results, and these are all identical, it can submit the transaction for ordering (that is, the R/W modifications and proofs of execution). Once the order of transactions has been established, all participants will receive the list of transactions to record in their ledgers. At this point, the smart contracts are not re-executed but instead their read/write modifications are made directly on the ledger. The benefit of this solution is that it allows executing a large number of smart contracts in parallel but, since all simulations happen on a "view" of the ledger state that can change by the time validation happens, transactions can fail in the validation phase due to data staleness in the state DB in the first phase. In the EOV model it is feasible to use general purpose programming languages, such as Go or Java, to write smart contracts since (most) non-deterministic behavior will be detected in the simulation phase.

5 RECENT RELATED WORK IN DATABASES

In the following, we present a handful of related works published within the database community targeting shortcomings of permissioned blockchains mentioned above (the list is not exhaustive and is intended as a sample of the space).

In an effort to increase transaction processing throughput in the EOV model, Sharma et al. [16], show that by relying on database techniques for concurrency control, it is possible to reorder transactions within a block during the ordering step (i.e., when establishing their total order) in a way that minimizes the number of failing transactions due to R/W conflicts. They implemented their prototype on Fabric and it is one example of how ideas from the database world can be used to improve existing blockchain platforms without fundamental redesigns. Other examples include FastFabric [7], that brings various optimizations, inspired by databases, to Fabric which result in an unprecedented 20,000 ops/s throughput.

There are also proposals which aim to increase throughput and lower latencies by implementing sharding at different levels in permissioned blockchains. CAPER [1], for instance, allows multiple applications to share one blockchain and leverages the fact that they operate on disjoint parts of the dataset to increase the overall throughput. This is achieved by separating the ordering of operations inside an application from that of ordering across applications. In a similar vein, ResilientDB [8] proposes a permissioned blockchain that incorporates a hierarchical consensus protocol design that relies on locality, both in terms of dataset and physical proximity of the nodes, to boost performance.

Other lines of work treat blockchains as a fault-tolerant and highly available storage layer and build traditional database capabilities on top, e.g., as in BlockchainDB [6]. This direction of research provides one possible answer the question of how to bridge the gap between SQL-based processing (and the amassed expertise in this space by developers) and the fairly exotic field of smart contracts. Other work, e.g., ChainifyDB [15], explores a similar

question and provides a different possible answer. Instead of replacing the storage engines of databases, ChainifyDB re-designs the distributed transaction processing protocol and utilizes a blockchain for BFT fault tolerance and transparency/auditing for both local and distributed transactions.

ABOUT THE AUTHOR

Zsolt István is an Associate Professor at the IT University of Copenhagen, working in the area of databases, distributed systems, and FPGA programming. Earlier, he was an Assistant Research Professor at the IMDEA Software Institute in Madrid, Spain. He holds a PhD and MSc in Computer Science from ETH Zurich, Switzerland. His personal website is at: https://zistvan.github.io.

REFERENCES

[1] M. J. Amiri, D. Agrawal, and A. E. Abbadi. Caper: a cross-application permissioned blockchain. *Proceedings of the VLDB Endowment*, 12(11):1385–1398, 2019.

[2] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[3] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn. Corda: an introduction. *R3 CEV, August*, 1:15, 2016.

[4] V. Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37), 2014.

[5] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[6] M. El-Hindi, C. Binnig, A. Arasu, D. Kossmann, and R. Ramamurthy. Blockchaindb: A shared database on blockchains. *Proceedings of the VLDB Endowment*, 12(11):1597–1609, 2019.

[7] C. Gorenflo, S. Lee, L. Golab, and S. Keshav. FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second. In *IEEE ICBC*, 2019.

[8] S. Gupta, S. Rahnama, J. Hellings, and M. Sadoghi. Resilientdb: Global scale resilient blockchain fabric. *Proceedings of the VLDB Endowment*, 13(6).

[9] L. Kuhring, Z. István, A. Sorniotti, and M. Vukolić. Streamchain: Rethinking blockchain for datacenters. *arXiv preprint arXiv:1808.08406*, 2018.

[10] Y. Li, B. Cao, M. Peng, L. Zhang, L. Zhang, D. Feng, and J. Yu. Direct acyclic graph-based ledger for internet of things: Performance and security analysis. *IEEE/ACM Transactions on Networking*, 28(4):1643–1656, 2020.

[11] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, 2008.

[12] S. Popov. The tangle. *White paper*, 1:3, 2018.

[13] T. Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies. *Available [online].[Accessed: 4-12-2018]*, 2018.

[14] L. S. Sankar, M. Sindhu, and M. Sethumadhavan. Survey of consensus protocols on blockchain applications. In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 1–5. IEEE, 2017.

[15] F. M. Schuhknecht, A. Sharma, J. Dittrich, and D. Agrawal. Chainifydb: How to blockchainify any data management system. *arXiv preprint arXiv:1912.04820*, 2019.

[16] A. Sharma, F. M. Schuhknecht, D. Agrawal, and J. Dittrich. Blurring the lines between blockchains and database systems: the case of hyperledger fabric. In *Proceedings of the 2019 International Conference on Management of Data*, pages 105–122. ACM, 2019.