Blockchains and Databases: Opportunities and Challenges for the Permissioned and the Permissionless

Divyakant Agrawal, Amr El Abbadi(✉), Mohammad Javad Amiri, Sujaya Maiyya, and Victor Zakhary

Department of Computer Science, University of California, Santa Barbara, USA
{agrawal,amr,amiri,sujaya_maiyya,victorzakhary}@cs.ucsb.edu

1 Introduction

Bitcoin [12] is a successful and interesting example of a global scale peer-to-peer cryptocurrency that integrates many techniques and protocols from cryptography, distributed systems, and databases. The main underlying data structure is blockchain, a scalable fully replicated structure that is shared among all participants and guarantees a consistent view of all user transactions by all participants in the system. In a blockchain, nodes agree on their shared states across a large network of *untrusted* participants. Although originally devised for cryptocurrencies, recent systems exploit its many unique features such as transparency, provenance, fault tolerance, and authenticity to support a wide range of distributed applications. Bitcoin and other cryptocurrencies use *permissionless* blockchains. In a permissionless blockchain, the network is public, and anyone can participate without a specific identity. Many other distributed applications, such as supply chain management and healthcare, are deployed on *permissioned* blockchains consisting of a set of known, identified nodes that still might not fully trust each other. This paper illustrates some of the main challenges and opportunities from a database perspective in the many novel and interesting application domains of blockchains. These opportunities are illustrated using various examples from recent research in both permissionless and permissioned blockchains. Two main themes unite the various examples: (1) the important role of distribution and consensus in managing large scale systems and (2) the need to tolerate malicious failures. The advent of cloud computing and large data centers shifted large scale data management infrastructures from centralized databases to distributed systems. One of the main challenges in designing distributed systems is the need for fault-tolerance. Cloud-based systems typically assume trusted infrastructures, since data centers are owned by the enterprises managing the data, and hence the design typically only assumes and tolerates crash failures. The advent of blockchain and the underlying premise that copies of the blockchain are distributed among untrusted entities has shifted the focus of fault-tolerance from tolerating crash failures to tolerating malicious failures. These interesting and challenging settings pose great opportunities for database researchers.

J. Darmont et al. (Eds.): ADBIS 2020, LNCS 12245, pp. 3–7, 2020.
https://doi.org/10.1007/978-3-030-54832-2_1

2 Permissionless Blockchains

The recent adoption of blockchain technologies and open permissionless networks suggest the importance of peer-to-peer atomic cross-chain transaction protocols. Users typically own assets in different crypto-currencies and should be able to atomically exchange their assets across different blockchains without depending on centralized intermediaries such as exchanges. Recent peer-to-peer atomic cross-chain swap protocols use hashlocks and timelocks to ensure that participants comply with the protocol [1,10,13]. However, an expired timelock could lead to a violation of the all-or-nothing atomicity property. An honest participant who fails to execute a smart contract on time due to a crash failure or network delays might end up losing her assets. Although a crashed participant is the only participant who ends up worse off, current proposals are unsuitable for atomic cross-chain transactions in asynchronous environments where crash failures and network delays are the norm.

An **A**tomic **C**ross-**C**hain **T**ransaction, AC^{2}T, is a distributed transaction that spans multiple blockchains. This distributed transaction consists of sub-transactions and each sub-transaction is executed on a different blockchain. An **A**tomic **C**ross-**C**hain **C**ommitment protocol is required to execute AC^{2}Ts. Such a protocol is a variation of traditional distributed atomic commitment protocols (e.g., 2PC [7,8]) and should guarantee both *atomicity* and *commitment* of AC^{2}Ts. **Atomicity** ensures the **all-or-nothing** property where either all sub-transactions take place or none of them are executed. **Commitment** guarantees that any changes caused by a cross-chain transaction are durable and once committed, these changes are permanent. Unlike in 2PC and other traditional distributed atomic commitment protocols, atomic cross-chain commitment protocols are also trust-free and therefore must **tolerate** maliciousness [10]. Existing solutions by Nolan [1,13] and generalized by Herlihy [10] do not guarantee the atomicity of AC^{2}Ts in asynchronous environments where crash failures, network partitioning, denial of service attacks and message delays are possible. In [14], we present an **A**tomic **C**ross-**C**hain **C**ommitment protocol that uses an open **W**itness **N**etwork. Events for redeeming and refunding transactions that exchange assets across permissionless blockchains are modeled as conflicting events. An open permissionless network of witnesses is used to guarantee that conflicting events could never simultaneously occur and either all smart contracts in an atomic cross-chain transaction are redeemed or all of them are refunded.

3 Permissioned Blockchains

Permissioned blockchains consist of a set of known, identified nodes that still might not fully trust each other. To be practical in real-life settings permissioned blockchains face multiple challenges regarding the *confidentiality*, *verifiability*, *performance*, and *scalability* requirements of distributed applications.

Confidentiality of data is required in many collaborative distributed applications, e.g., supply chain management, where multiple enterprises collaborate

with each other following Service Level Agreements (SLAs) to provide different services. To deploy distributed applications across different collaborating enterprises, a blockchain system needs to support the internal transactions of each enterprise as well as cross-enterprise transactions that represent the collaboration between enterprises. While the data accessed by cross-enterprise transactions should be *visible* to all enterprises, the internal data of each enterprise, which are accessed by internal transactions, might be *confidential*. In Caper [2], each enterprise orders and executes its internal transactions locally while cross-enterprise transactions are public and visible to every node. In addition, the blockchain ledger of Caper is a directed acyclic graph that includes the internal transactions of every enterprise and all cross-enterprise transactions. Nonetheless, for the sake of confidentiality, the blockchain ledger is not maintained by any node. In fact, each enterprise maintains its own *local view* of the ledger including its internal and all cross-enterprise transactions. Since ordering cross-enterprise transactions requires global agreement among all enterprises, Caper introduces different consensus protocols to globally order cross-enterprise transactions.

Besides confidentiality, in many cross-enterprise systems, e.g., crowdsourcing applications, participants need to verify transactions that are initiated by other enterprises to ensure the satisfaction of some predefined global constraints on the entire system. Thus, the system needs to support *verifiability* while preserving the confidentiality of transactions. To address this problem, we have introduced Separ [6], a multi-platform blockchain-based crowdsourcing system that uses a token-based technique to ensure verifiability. A token can be seen as an entity that represents some property based on a global constraint, which need to be verified. For example, if a global constraint declares that a particular participant cannot initiate more than 20 transactions in a week, the system will assign 20 tokens to that participant and the participant consumes a token whenever it initiates a transaction. Depending on the requested global constraints, the tokens might need to satisfy different properties. First, tokens need to be non-exchangeable, i.e., different participants cannot exchange their tokens. Second, a token should expire after some predetermined amount of time, and third, a token cannot be consume more than once.

In addition to confidentiality and verifiability, distributed applications, e.g., financial applications, require high performance in terms of throughput and latency, e.g., while the Visa payment service is able to handle more than 10000 transactions per second, Multichain [9] can handle at most 200 transactions per second. SharPer [3,5] supports the concurrent processing of transactions by clustering nodes into clusters and sharding the data and the blockchain ledger. SharPer supports both intra-shard and cross-shard transactions and introduces flattened consensus protocols for ordering cross-shard transactions among the involved clusters.

Finally, scalability is one of the main obstacles to business adoption of blockchain systems. To support a distributed application, e.g., large-scale database, a blockchain system should be able to scale efficiently by adding more nodes to the system. While database systems use the sharding technique to

improve the scalability of databases in a network of crash-only nodes, the technique cannot easily be utilized by blockchain systems due to the existence of malicious nodes in the network. ParBlockchain [4] introduces a new paradigm to support distributed applications that execute concurrently for workloads with some degree of contention. In ParBlockchain a disjoint set of nodes (orderers) establishes agreement on the order among transactions of different enterprises, constructs blocks of transactions, and generates a *dependency graph* for the transactions within a block. A dependency graph gives a partial order based on the conflicts between transactions and enables the parallel execution of non-conflicting transactions. Transactions are then executed following the generated dependency graph. While ParBlockchain supports contentious workloads, any non-deterministic execution of transactions will decrease its performance.

4 Back to Databases

As increasing amounts of data are currently being stored and managed on third-party servers. It is impractical for small scale enterprises to own their private datacenters, hence renting third-party servers is a viable solution. But the increasing number of malicious attacks, both internal and external, as well as buggy software on third-party servers may cause clients to lose their trust in these external infrastructures. While small enterprises cannot avoid using external infrastructures, they need the right set of protocols to manage their data on untrusted infrastructures. Fides [11], introduces a novel atomic commitment protocol, TFCommit, that executes transactions on data stored across multiple untrusted servers. This novel atomic commitment protocol executes transactions in an untrusted environment without using expensive Byzantine replication. Using TFCommit, we propose an *auditable* data management system, Fides, residing completely on untrustworthy infrastructure. As an auditable system, Fides guarantees the detection of potentially malicious failures occurring on untrusted servers using blockchain inspired tamper-resistant logs with the support of cryptographic techniques. Fides is scalable and incurs relatively low overhead that allows executing transactions on untrusted infrastructure.

Acknowledgement. This work is partially funded by NSF grants CNS-1703560 and CNS-1815733.

References

1. Atomic cross-chain trading (2018). https://en.bitcoin.it/wiki/Atomic_cross-chain_trading
2. Amiri, M.J., Agrawal, D., El Abbadi, A.: Caper: a cross-application permissioned blockchain. Proc. VLDB Endow. **12**(11), 1385–1398 (2019)
3. Amiri, M.J., Agrawal, D., El Abbadi, A.: On sharding permissioned blockchains. In: Second International Conference on Blockchain. IEEE (2019)

4. Amiri, M.J., Agrawal, D., El Abbadi, A.: Parblockchain: leveraging transaction parallelism in permissioned blockchain systems. In: 39th International Conference on Distributed Computing Systems (ICDCS), pp. 1337–1347. IEEE (2019)
5. Amiri, M.J., Agrawal, D., El Abbadi, A.: Sharper: sharding permissioned blockchains over network clusters. arXiv preprint arXiv:1910.00765 (2019)
6. Amiri, M.J., Duguépéroux, J., Allard, T., Agrawal, D., El Abbadi, A.: Separ: a privacy-preserving blockchain-based system for regulating multi-platform crowdworking environments. arXiv preprint arXiv:2005.07850 (2020)
7. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency control and recovery in database systems (1987)
8. Gray, J.N.: Notes on data base operating systems. In: Bayer, R., Graham, R.M., Seegmüller, G. (eds.) Operating Systems. LNCS, vol. 60, pp. 393–481. Springer, Heidelberg (1978). https://doi.org/10.1007/3-540-08755-9_9
9. Greenspan, G.: Multichain private blockchain-white paper. http://www.multichain.com/download/MultiChain-White-Paper.pdf (2015)
10. Herlihy, M.: Atomic cross-chain swaps. In: ACM Symposium on Principles of Distributed Computing (PODC), pp. 245–254. ACM (2018)
11. Maiyya, S., Cho, D.H.B., Agrawal, D., Abbadi, A.E.: Fides: managing data on untrusted infrastructure. In: 40th International Conference on Distributed Computing Systems (ICDCS). IEEE (2020)
12. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
13. Nolan, T.: Alt chains and atomic transfers (2013). https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949
14. Zakhary, V., Agrawal, D., El Abbadi, A.: Atomic commitment across blockchains. Proc. VLDB Endow. **13**, 1 (2020)