Performance and Fault Tolerance Trade-offs in Sharded Permissioned Blockchains

Chunyu Mao
Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
c5mao@uwaterloo.ca

Anh-Duong Nguyen
Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
ad3nguyen@uwaterloo.ca

Wojciech Golab
Electrical and Computer Engineering
University of Waterloo
Waterloo, Canada
wgolab@uwaterloo.ca

***Abstract*—Blockchain has become a promising technology in distributed systems in recent years, but scalability remains a major problem. The traditional approach to scalability, namely sharding, does not solve the problem easily because the process of interleaving blocks stored in different shards to create a unified master ledger introduces overhead. This paper examines two techniques for interleaving the shards of permissioned blockchains, which we refer to as strong temporal coupling and weak temporal coupling. We implement these techniques in a prototype system with a Bitcoin-like transaction structure, using the EPaxos consensus protocol for transaction ordering. Our experimental results show that strong coupling can achieve lower latency as compared to weak coupling but same level of peak throughput. However, strong coupling requires all shards to grow at the same rate, and cannot tolerate any shard failure. In contrast, the higher latency of weak coupling is because of the consensus strategy it uses to order the blocks. However, if shard failure occurs, weak coupling can still make progress without stalling the whole system.**

***Index Terms*—Blockchain, Sharding, Interleaving, Scalablity, Fault Tolerance**

I. Introduction

Since Bitcoin was introduced by Nakamoto in 2008 [1], blockchain technology has been considered as a promising solution to the trust issue raised by the traditional distributed systems. The core of blockchain systems is a consensus protocol [2]. The FLP impossibility result [3] states that the traditional notion of consensus in computer science cannot be achieved in an asynchronous environment in the face of server crashes. This barrier can be overcome with randomization and partial synchrony [4]–[6], but implementing a secure and consistent blockchain system using consensus remains hard at scale. The two most sophisticated permissionless blockchains, Bitcoin [1] and Ethereum [7] solved the security and consistency issue using Proof of Work (PoW). However, Bitcoin can only process around 7 txns/sec, while Ethereum does about 30 txns/sec. In contrast to permissionless blockchains which do not require authentication to join the network, most permissioned blockchains, like BoscoChain [8] assume a high degree of control over the set of participants and try to employ Byzantine [9] consensus protocols to achieve security and consistency. However, due to the communication complexity of Byzantine agreement and the cost of crypto computation, such systems have difficulty scaling beyond roughly 10,000 txns/sec [10], [11].

Sharding is an intuitive approach to improve the scalability of the blockchain. The final ledger is derived in sharded systems by interleaving blocks from all the shards into one single blockchain, and checking carefully for cross-shard double-spending. Elastico [12], OmniLedger [13] and RapidChain [14] are three permissionless sharded blockchains. BoscoChain [8] is a permisssioned sharded blockchain. Although sharding is a conventional technique to achieve high scalability in database systems, the effect of sharding in blockchain systems is not well understood. This paper examines the scalability and fault tolerance trade-off of two methods for interleaving shards, which we refer to as strong temporal coupling and weak temporal coupling.

Authors supported by Ripple, and by the Natural Sciences and Engineering Research Council of Canada (NSERC).

II. Methods of Temporal Coupling

From a high-level point of view, interleaving in sharded blockchain systems means combining multiple sequences of blocks into a unique main sequence of blocks, and ensuring that each transaction remains valid in this main sequence.

Strong temporal coupling means the final ledger is obtained by interleaving the blocks of different shards in a static order. The advantage of this method is that each peer can do the interleaving independently without communicating with other peers, and still guarantee consistency if there is no Byzantine failure. The principal drawback of this method is that when one shard becomes unavailable (e.g., during a network partition), the entire system stalls. Multiple remedies to this problem are proposed in [15], all requiring that transaction commitment be delayed by one or more consensus cycles – a hazard to latency.

In contrast to strong coupling, weak temporal coupling interleaves the blocks dynamically using another layer of consensus. This method can also guarantee consistency, and there is no limitation on the progress of each shard. However, consensus must be achieved via additional communication among all the peers.

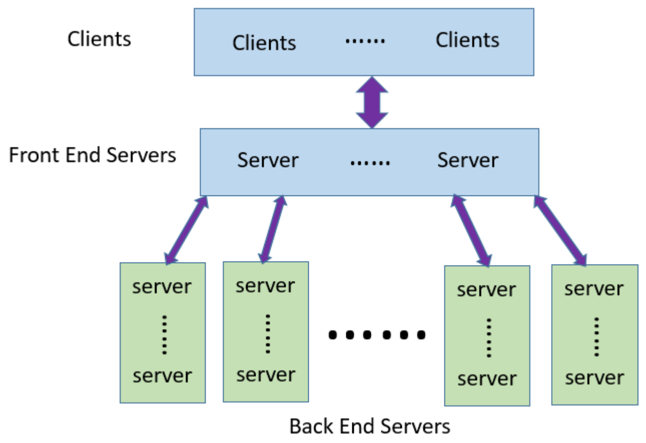

Fig. 1. System architecture.

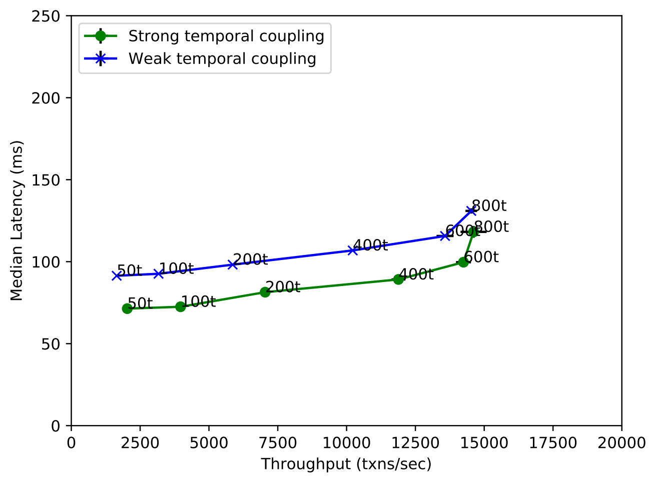

Fig. 2. Performance of 3 shards.

III. System Architecture

To investigate the impact of temporal coupling in sharded blockchains, we implement a permissioned blockchain prototype in Golang. This section provides an overview of the prototype's system architecture, which is presented in Figure 1.

Clients are at the top layer, which proposes transactions to the front end (FE) servers. Clients can only communicate with FE servers. The FE servers in the middle layer are used for coordinating between clients and shards, and also work as interleaving servers. There are three functions that the FE server needs to carry out. First, collecting transactions from clients. Then, gathering the transactions into blocks and distributing the blocks to shards. Also, FE servers continuously request blocks from each shard and interleave them into the final ledger through either strong coupling or weak coupling. The interleaving process involves a double-spending check of transactions in each block against the interleaved blockchain. At last, each front end server stores a full copy of the interleaved final ledger, which is called a full node in Bitcoin.

The bottom layer consists of back end (BE) servers, which are arranged into multiple shards. The main function of the BE servers is collecting blocks from FE servers and arranging them into a consistent chain of blocks as a shard, which involves consensus. Each shard is maintained by a quorum of BE servers working together to tolerate crash failures. Besides the main function, BE servers are also responsible for serving blocks from the local shard to FE servers.

IV. Evaluation and Discussion

We run the experiments in a simulated environment, hosted on a multi-core server that supports up to 160 parallel threads (quad-socket Intel Xeon Gold 6230). Each process is assigned 4 cores, and 5ms round-trip network latency between processes is simulated using the traffic control subsystem of the Linux kernel. Block size is capped at 1000 transactions, and is further limited by the number of parallel threads per client.

The latency is measured at the client on an end-to-end basis as the time from when a user sent a transaction to the time when that user received a reply confirming whether or not the transaction was appended to the blockchain. We measure the latency of each confirmed transaction and the total number of confirmed transactions over 20s. Then we calculate the median latency and the number of confirmed transactions per second. We run the experiment 3 times and take the average to plot the figure.

Scalability. In the experiment, we investigate scalability by measuring the performance for 3 shards. Figure 2 shows the performance of strong and weak coupling. From the figure we can see that the latency of strong coupling beats weak coupling consistently due to the lower overhead of interleaving, while achieving almost the same peak throughput. We expect the difference in latency to grow as inter-process network latency increases, such as in a geo-distributed scenario.

V. Conclusion

In this paper, we compared two methods for interleaving transactions from different data partitions or shards in permissioned blockchains. We refer to these as strong temporal coupling and weak temporal coupling, reflecting the restriction they impose on the rate of progress at differnet shards. We implemented a blockchain prototype to evaluate the relative merits of both methods. Our experimental results suggest that strong coupling has slightly lower latency but similar peak throughput. On the other hand, strong coupling is inherently less robust against network partitions and shard failure, which can stall the entire system in some situations where weak coupling is able to maintain progress. We will consider Byzantine fault-tolerant implementations in future work.

VI. Acknowledgment

We are grateful to Srinivasan Keshav, Sergey Gorbunov, Robbert van Renesse, and Bernard Wong for stimulating discussions on the topic of consensus and blockchain scalability.

References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash system.” 2008.

[2] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, “Consensus in the age of blockchains,” *in CoRR*, vol. abs/1711.03936, 2017.

[3] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” *in J. ACM*, vol. 32, no. 2, Apr. 1985.

[4] M. Ben-Or, “Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols,” *in Proc. of the 2nd ACM Symposium on PODC*, pp. 27–30, 1983.

[5] G. Bracha and S. Toueg, “Asynchronous consensus and broadcast protocols,” *in J. ACM*, vol. 32, no. 4, pp. 824–840, Oct. 1985.

[6] C. Dwork, N. Lynch, and L. Stockmeyer, “Consensus in the presence of partial synchrony,” *in J. ACM*, vol. 35, no. 2, pp. 288–323, Apr. 1988.

[7] V. Buterin, “Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform.” 2014.

[8] R. van Renesse, “BoscoChain: Keeping Byzantine consensus for bockchains simple and flexible,” *Presentation at University of Waterloo on Fri, 17 Nov*, 2017.

[9] L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals problem,” *in ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[10] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” *OSDI. USENIX Assmiation, Co-sponsoed by IEEE TCOS and ACM SIGOPS*, 1999.

[11] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, D.-A. Seredinschi, and A. T. Orr Tamir, “SBFT: a Scalable and Decentralized Trust Infrastructure,” *arXiv: 1804.01626v2 [cs.DC]*, 2018.

[12] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, “A secure sharding protocol for open blockchains,” *in Proc. of ACM SIGSAC CCS*, pp. 17–30, 2016.

[13] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, “OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding,” *2018 IEEE Symposium on Security and Privacy (S&P)*, pp. 19–34, 2018.

[14] M. Zamani, M. Movahedi, and M. Raykova, “RapidChain: Scaling Blockchain via Full Sharding,” *in Proc. of ACM SIGSAC CCS*, pp. 931–948, 2018.

[15] S. Keshav, W. Golab, B. Wong, S. Rizvi, and S. Gorbunov, “RCanopus: Making Canopus Resilient to Failures and Byzantine Faults,” *arXiv:1810.09300 [cs.DC]*, 2018.