Bassa - Scalable Blockchain Architecture for Smart Cities

Eranga Bandara*, Xueping Liang†, Peter Foytik*, Sachin Shetty*, Nalin Ranasinghe‡, Kasun De Zoysa‡
* {cmedawer, PFoytik, sshetty}@odu.edu
Old Dominion University, Norfolk, VA, USA
† {x_liang}@uncg.edu
University of North Carolina at Greensboro, NC, USA
‡ {dnr, kasun}@ucsc.cmb.ac.lk
University of Colombo School of Computing, Sri Lanka

***Abstract*—Smart cities will depend on reliable Internet of things to effectively provide residential and commercial services. Smart city applications generate vast amount of data and will need the infrastructure to support high transaction throughput. The data associated with smart cities can include sensitive information and there is a need to guarantee the security and privacy through a blockchain-based decentralized infrastructure. Current blockchain environments face several challenges, such as, lack of high transaction throughput, high scalability, real-time transaction processing, and back-end stress operations, etc. In this paper, we propose "Bassa", a blockchain platform that will meet the aforementioned challenges. Bassa employs a Apache Kafka based consensus and 'validate-execute-group" blockchain architecture to realize real-time transaction. Bassa's smart contract platform realizes concurrent transaction by leveraging actor-based concurrency.**

***Index Terms*—Blockchain, Smart City, Big Data, IoT, Distributed Systems**

1. Introduction

Smart cities deal with connected "things" and bring together an infrastructure to improve the quality of life of citizens and enhance their interactions with the urban environment. The deployment of smart city will involve the integration of technologies, such as, artificial intelligence, machine learning, sensor network and engineering systems. The plethora of sensor devices will generate a vast amount of data for the smart city infrastructure to use it for supporting residences and businesses. The ability to handle this large amount of sensor data will require a platform that can address scalability, high transaction throughput, security and privacy. The appropriate handling of sensitive data will be paramount for consumers to participate in a smart city environment.

To handle the security and privacy concerns of smart city applications, blockchain-based infrastructure can be adopted. Data sharing between different components, data aggregation, monitoring, decision-making functions in smart cities can be securely facilitated with blockchain-based architecture, as in Figure 1. However when integrating existing blockchain platforms with smart city applications, one encounter many challenges: 1) Existing blockchain platforms are not scalable, 2) Do not support real-time transaction processing, 3) Do not support high transaction write throughput, 4) Do not support rich query APIs, so cannot support data analytics/machine learning functions 5) Do not support back-end stress [1] operation handling 6). As such, current blockchains are not immediately suited for the highly scalable smart city applications.

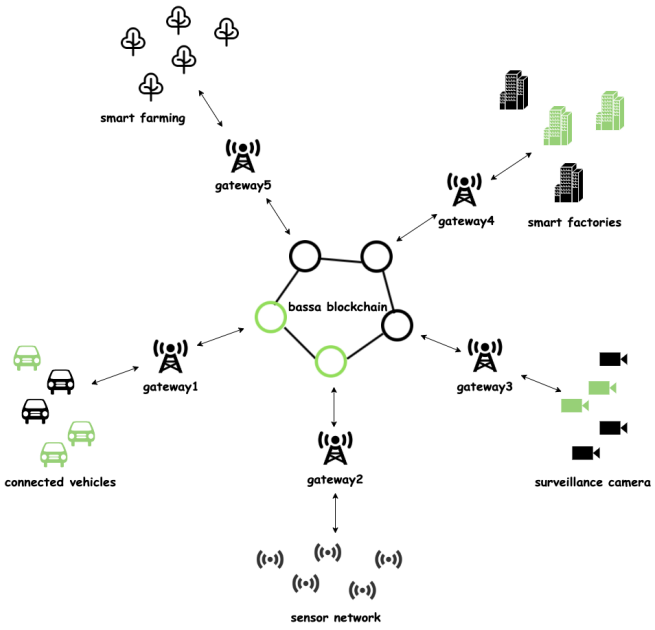

Figure 1: Blockchian-based smart city architecture. Different entities connects to the blockchain via gateways.

When we look at existing blockchains and their scalability issues, we have identified three main bottlenecks: "Storage model", "Order-Execute architecture" and "imperative-style smart contracts". In existing blockchains, each peer in the network maintains their own ledger. Data on the network

replicates throughout all nodes. Unlike distributed database, there is no sharding to improve the system performance [2]. With Order-Execute architecture, blockchains cannot execute transactions when a peer submits a transaction to the network. To validate and execute a transaction, it needs to wait until a block is created [3]. Most of the existing blockchain smart contract platforms [4] do not come with concurrency control. Therefore, concurrent transactions are not supported in existing blockchain platforms. All transactions execute sequentially and it is the same when updating the ledger. This results in low transaction throughput and latency [5], [6].

In this paper, we introduce a highly scalable blockchain platform, "Bassa" to address the requirements of smart city based applications. Bassa addresses the above mentioned issues on the existing blockchain platform and builds a scalable blockchain system as a proof-of-concept prototype. The real-time transaction enabled "Validate-Execute-Group" blockchain architecture is introduced using Apache Kafka based consensus, with Actor [7], [8] based Aplos smart contracts written with a functional programming language [9]. Aplos smart contract supports concurrent transaction execution on the blockchain. Bassa utilizes the eventually consistent distributed storage [10] as an underlying asset storage platform. The data maintained on the asset storage platform will be replicated with other nodes using sharding. With this approach, we can resolve the full node data replication issue which exists in traditional blockchains [2] on Bassa. Following are the main research contributions in this paper.

1) Real-time transaction enabled, scalable blockchain architecture is introduced with Apache Kafka-based consensus. The proposed architecture reduces the overhead of order-execute architecture.
2) Functional programming and Actor based smart contract platform is integrated to achieve concurrent transactions in the blockchain.
3) Instead of full-node data replication, sharding has been used to reduce the network and communication overhead in the blockchain.
4) Microservices-based architecture is introduced to build scalable blockchain applications. Backpressure operations on scalable applications are handled with Reactive Streaming based methodology.
5) Data analytics and machine learning functions are integrated into blockchain with full-text search support.

The rest of the paper is organized as follows. Section 2 and 3 discuss the overview, architecture and implementation of the Bassa blockchain. Section 4 presents the performance evaluation of the Bassa blockchain. Section 5 offers insights into the related work. Finally, Section 6 concludes the paper.

2. Bassa

2.1. Bassa Overview

Bassa is a highly scalable Blockchain system which is targeted for scalable and concurrent applications(e.g smart city applications). It utilizes Apache Kafka [11] as the consensus platform and eventually consistent distributed database system [10] as the underlying asset storage. Bassa adopts reactive streams based approach [12] to handle back pressure [1] operations on scalable, concurrent applications. Bassa utilizes the Apache Lucene index [13] based API [14] to perform full-text search on blockchain data. The storage integrated with the federated machine learning service which is capable of performing analytics on blockchain data.

2.2. Address Performance Bottlenecks

There are three main performance bottlenecks affecting the scalability and transaction throughput of the blockchain, a) full node data replication, b) imperative style smart contracts, c) order-execute architecture. Bassa proposes an alternative architecture to address these performance bottlenecks. Each peer in Bassa blockchain comes with two types of storage, off-chain storage and on-chain storage. Both of them are built on top of eventually consistent distributed database nodes [10]. Off-chain storage stores the actual data generated by the peers. The hash of these data is published to on-chain storage and shared with other peers. Bassa blockchain keeps all its transactions, blocks, and asset information on this on-chain storage. The on-chain storage in each peer is connected in a ring cluster architecture. After a transaction is executed each peer performs a state update on that is distributed and replicated using sharding using an underlying storage service. By using this approach, Bassa avoids the full node replication requirement that can cause issue with traditional blockchains [2].

To address the issues on imperative style smart contracts on existing blockchains, Bassa introduces a concurrent transaction enabled Aplos smart contract platform. This smart contract platform is built using functional programming [9] and actor-based concurrency handling [7]. By supporting concurrent transaction execution with smart contracts, Bassa produces high transaction throughput and high scalability in blockchain [4].

Bassa introduces real-time transactions enabled "Validate-Execute-Group" blockchain architecture to address the issues of Order-Execute architecture [3] in traditional blockchain platforms. The Order-Execute architecture based systems validate and execute transactions on each peer multiple times. But in Validate-Execute-Group based systems, transactions will be validated and executed only once in one peer when the client submits them to the blockchain. The client does not need to wait until creating a block to commit the transaction. In the validate phase it checks the double spend of the transactions when the client submits them to the blockchain. Execute phase, executes the transactions with Aplos smart contracts. With Aplos smart contracts, the transactions can be executed concurrently when the time peer submits them to the blockchain. Once smart contracts are executed, the transaction and state update will be stored in underlying asset storage in Bassa. Blocks will be created in the Group phase. This new architecture provides real-time transactions execution of

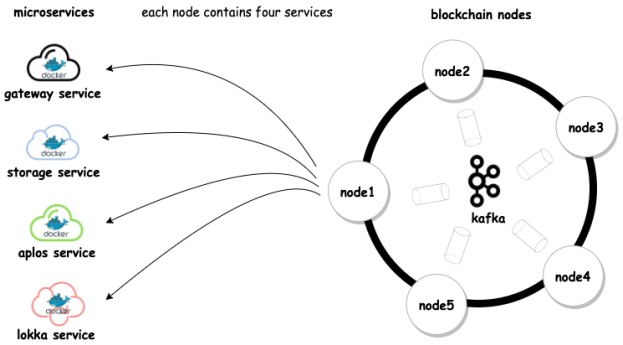

Figure 2: Bassa microservice-based architecture. Each blockchain node contains five services - Gateway service, Storage service, Aplos service and Lokka service.

the blockchain which reasons high scalability and high transaction throughput [15].

3. Bassa Architecture

3.1. Overview

Today, even the most advanced blockchain systems are built as monolithic systems [15]. All features of the blockchain in some cases are controlled from programs that are hosted by a single service [3], [15]. Example services include validating transactions, broadcasting, consensus handling, maintaining asset storage, etc. This could be modularized in a more distributed way to improve efficiency. In a monolithic system approach, everything is built in the same programming language to ensure easy integration. Since only one service is available, it is difficult to scale [16]. To improve scalability, we built Bassa using a Microservice architecture, providing a better way to distribute the processing. All functionalities are implemented as Microservices, allowing architects to appropriately partition different services to the appropriate hardware. The architecture of the Bassa blockchain shows in Figure 2. Each Bassa blockchain node contains the following services:

1) Gateway service (API service where clients interact with blockchain)
2) Aplos service(Smart contract service which implemented with Actor-based concurrency controlling)
3) Storage service (Eventually consistent distributed storage which used to store [10] blocks, transactions and assets)
4) Lokka service (Block creating service implemented using functional programming and reactive streaming [12])
5) Kafka Message broker (Distributed publisher-subscriber message broker used to order the messages and transmit the messages)

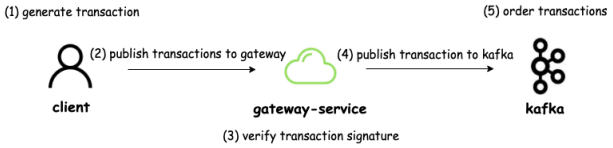

Figure 3: Bassa Gateway service architecture. Blockchain client(e.g LoRa hub in cities) connected to gateway service and publishes transactions. Then gateway service publishes them to kafka.

3.2. Gateway Service

Gateway is the client API of the Bassa blockchain. Blockchain clients connect to this service and publish their transactions. Once the transactions are received, it publishes the transactions into the Kafka message broker. Then the Kafka message broker orders the published transactions. Gateway service exposes HTTP based synchronous endpoint and Kafka message broker based asynchronous endpoint to the client, Figure 3. Each node in the cluster has a gateway service. The clients on each node publish transactions to their gateway service as JSON encoded objects(JSON message contains smart contract name and arguments) via the gateway API. The gateway service in Bassa blockchain is implemented with golang [17].

3.3. Aplos Service

Aplos is the smart contract service in Bassa blockchain. It is implemented using functional programming and actor-based concurrency controlling [4]. All blockchain-based software programs and the messages that pass between them are written as Aplos smart contracts [4]. Aplos consumes transactions from Kafka message broker as streams. After that, it validates and executes the transaction based on Validate-Execute-Group architecture. Bassa stores all the transactions on underlying asset storage. We have used eventually consistent distributed storage(e.g Apache Cassandra) with AP architecture in CAP theorem [18] as the asset storage. The default consistency model in AP based databases does not guarantee linearizable consistency [19], [20]. That means when querying the data it does not consider pending writes. To achieve linearizable consistency from AP based distributed storage, Bassa introduced a distributed cache-based mechanism. All the recent transaction execution information is added to the distributed cache. When validating it checks the double spends of the transaction in two steps, Figure 4. If both validations (two-step double-spend) are successful, the transaction is executed and the ledger asset is updated. The executed transaction ID is saved in the distributed cache. The actual transaction and asset update stored in the underlying storage service. Then storage service will distribute the state updates to other nodes in the cluster via sharding. Unlike other blockchain smart contracts, the Aplos smart contracts can execute transactions concurrently. The Aplos service has been implemented using the Scala

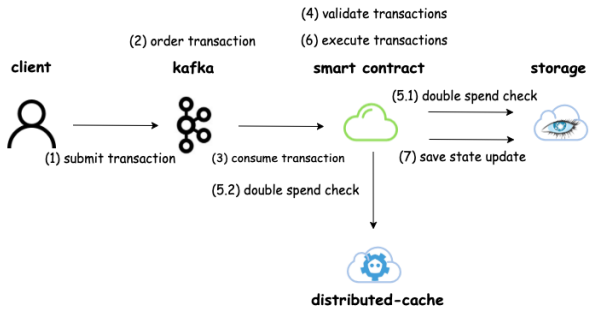

Figure 4: Bassa Aplos service architecture. Two step transaction validation happens with asset storage and distributed cache.

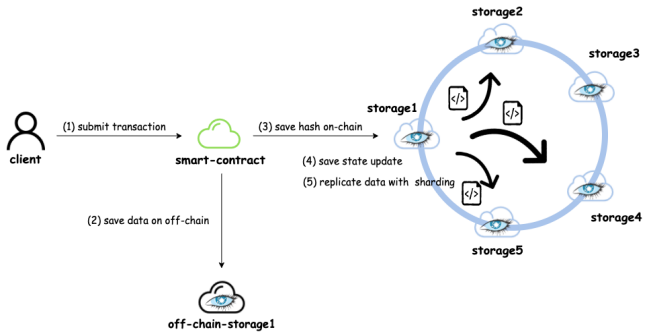

Figure 5: Bassa Storage service architecture. The Storage nodes are connected as a master-less ring cluster.

functional programming languages [21]. The Akka actor [8] framework is used to build smart contract programs.

3.4. Storage Service

Blocks, transaction, and assets on Bassa blockchain are stored in the Storage service. Apache Cassandra [10] is used by the Storage service providing a consistent distributed database for the Bassa system. In a scalable blockchain environment, every node in the cluster should have write capability and high transaction write throughput. Cassandra supports both of these requirements providing high-scalability and high-transaction write throughput up-to one million writes per second. Comparing to alternative approaches like master-slave architecture, Cassandra provides master-less ring architecture where any node in the cluster has the ability to write [22].

Each peer in Bassa blockchain comes with two types of storage, off-chain storage and on-chain storage. Off-chain storage stores the actual data generated by the peers. The hash of these data published to on-chain storage and shared with other peers. As shown in Figure 5, the on-chain Cassandra nodes are connected as a master-less ring cluster where each node has the write capability. Unlike other traditional blockchain systems, Bassa does not require the use of full node replication. The data is replicated with other nodes via sharding according to Cassandra's `replication_factor`. For instance, in a 10-node cluster, when the replication factor is set to 3, transactions will be distributed and replicated on 3 nodes. The on-chain storage comes with single Cassandra Keyspace and two main tables: `Transactions` and `Blocks`. Apart from these two tables, there are additional tables to store asset data. For an example in an account-based money transfer scenario, we can have an additional `Accounts` table on Cassandra to store user account balances. In the execution phase, valid transactions are added to the Transaction table. Based on the transaction outcome, the asset table(s) are updated. All the data in `Transactions`, `Blocks` and `Asset` tables will be indexed in Elasticsearch [14] to achieve full-text search capability of the Bassa blockchain. The storage integrated with the federated machine learning service which is capable of performing analytic on blockchain data in a privacy-preserving manner.

3.5. Lokka Service

The Lokka service creates blocks based on the transaction IDs in the cache. When creating a block, Lokka service generates the block hash and adds the transaction list to the block. Similar to transactions, this block is also stored in storage service and distributed to other nodes in the blockchain. Once the block is created it needs to be approved by other Lokka services in the network. The block approval process is done via federated consensus [23] which is implemented between Lokka services. When a block is approved, the newly created block information(e.g block ID) is broadcast to other Lokka services in the blockchain network via Kafka message broker. Other Lokka services take the block from their respective storage service and validate the transactions in the block. If all transactions in the block are valid, it gives a vote for the block (mark block as valid or invalid). To handle voting, Lokka service digitally signs the block hash and adds the signature into the block header. When the majority of Lokka services vote for a block, that block is considered as a valid block. The block ordering will happen with the majority vote from the federated consensus which is considered an endorsement policy defined among the Lokka services. This policy defines how many Lokka services need to vote to approve a block.

3.6. Kafka Message Broker

Apache Kafka used as the consensus platform and message broker in Bassa blockchain. All transactions published by the clients will be stored and ordered in a Kafka message broker. Aplos services take the ordered transactions from a Kafka message broker, execute them with smart contracts. Kafka message brokers in Bassa focus on two main scenarios. In the first scenario, the client publishes transaction messages to the gateway service via Kafka message broker. There is a separate Kafka message broker

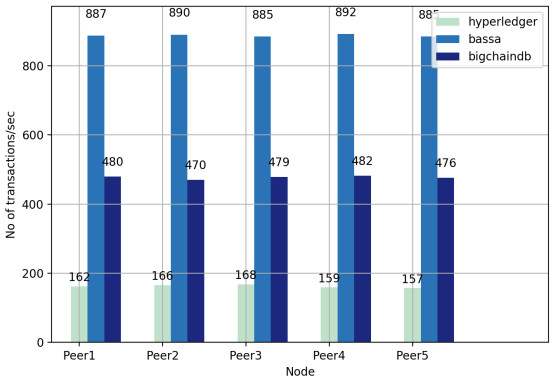

Figure 6: Invoke transaction results of Bassa, BigchainDB and HLF.

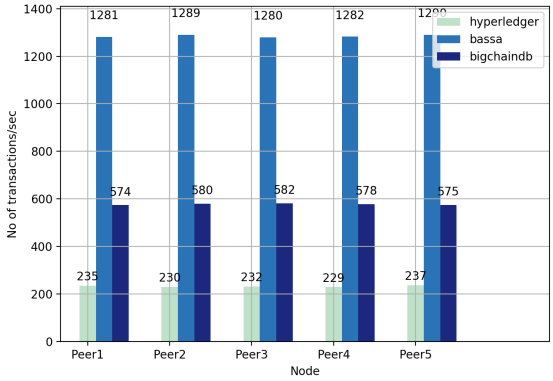

Figure 7: Query transaction results of Bassa, BigchainDB and HLF.

channel(e.g. Kafka topic) for each Gateway service. Clients publish transaction messages to these Kafka message broker channels, Figure 3. By using Kafka for client-to-gateway service communication, we can handle back-pressure operations [1] with handling a high transaction load in the scalable application. The second scenario is communication between Lokka services. When a Lokka service generates a block and saves it in the Blocks table, the block ID is broadcast to other Lokka services via Kafka for approval.

4. Bassa Performance Evaluation

Performance evaluation of Bassa is completed and discussed in comparison with Bigchaindb and Hyperledger Fabric blockchains. To obtain the results, Bassa, Hyperledger Fabric(HLF) and BigchainDB blockchains deployed on AWS cluster(AWS 2xlarge instance with 16GB RAM and 8 CPUs). Bassa blockchain is run with 4 Kafka nodes, 3 Zookeeper and Cassandra as the state database. HLF runs with Kafka based consensus with 3 Orderer nodes, 4 Kafka nodes, 3 Zookeeper nodes and LevelDB [3] as the state database. BigchainDB blockchain runs with Tendermint consensus [23] and MongoDB [24] as the state database. The evaluation results are obtained for the following, with the varying number of blockchain peers (1 to 7 peers) used in different evaluations.

1) Invoke Throughput (Invoke transactions)
2) Query Throughput
3) Scalability (of Transactions)
4) Transaction execution time

4.1. Invoke Transaction Throughput

In this evaluation, the number of "Invoke transactions" that are executed by a single blockchain peer is recorded. An Invoke transaction creates a transaction on the ledger and updates the status of the asset. Concurrent invoke transactions are issued for each blockchain peer and the number of executed transactions are recorded. As shown in Figure 6, the invoke transaction throughput of Bassa is compared to BigchainDB and HLF. Bassa performs with a higher transaction throughput than BigchainDB and HLF. The real-time transaction enabled Validate-Execute-Group blockchain architecture and concurrent transaction enabled smart contract platform are the main reasons for high transaction invoke throughput on Bassa blockchain.

4.2. Query Transaction Throughput

In this evaluation, the number of query transactions that can be executed in each Tikiri blockchain peer is recorded. Query transaction just retrieve the status from the ledger. They neither create transactions or update the ledger. Concurrent query transactions for each blockchain peer are performed and the number of completed queries are recorded. Similar to invoke transactions, the Query operation throughput of Bassa is compared with BigchainDB and HLF, Figure 7. Bassa serves its query API via Apache Lucene index-based search API. BigchainDB facilitates the query API with MongoDB and HLF with CouchDB. Query transaction throughput of Bassa is higher than both BigchainDB and HLF. Concurrent transaction execution of smart contracts, and Akka streams based back pressure handling are the main reasons for the improved results.

4.3. Transaction Scalability

For the scalability evaluation, the number of invoke transactions (per second) against the number of blockchain peers in the network are recorded. Concurrent invoke transactions are issued to each blockchain peer and the number of executed transactions are recorded. As shown in Figure 8 the transaction scalability of Bassa is compared with BigchainDB and HLF. Bassa has higher scalability than BigchainDB and HLF blockchains. When the number of peers increases, the rate of executed transactions increase relatively. The real-time transaction enabled blockchain architecture and concurrent transactions enabled smart contract platform are the main reasons for the higher scalability in Bassa.

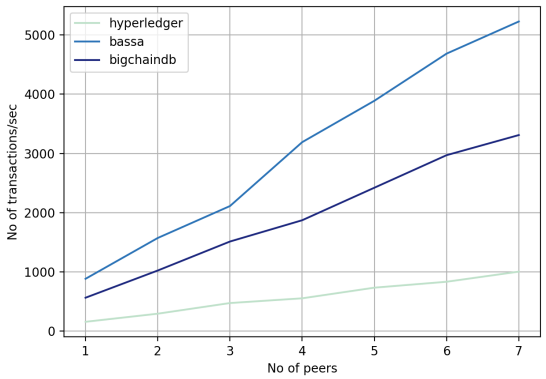

Figure 8: Transaction scalability results of Bassa, BigchainDB and HLF.

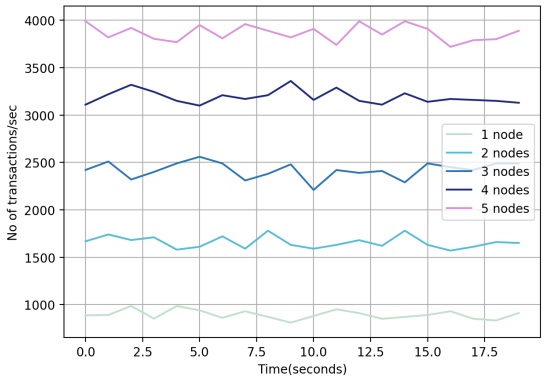

Figure 9: Transaction execution rate comparison with number of nodes in Bassa blockchain.

4.4. Transaction Execution Rate

Next, we evaluate the transaction execution rate in the Bassa platform. We tested the number of submitted transactions and executed transactions in different blockchain peers recording the time. Figure 9 shows how transaction execution rate varies when having a different number of blockchain peers in the Bassa. When the number of peers increases, the rate of executed transactions is increased relatively. Figure 10 shows the number of executed transactions and submitted transactions in a single blockchain peer. There is a back-pressure operation [1] between the rates of submitted transactions and executed transactions. We have used a reactive streaming-based approach with Apache Kafka to handle these backpressure operations in the Bassa blockchain.

5. Related Work

Research has been conducted for supporting blockchain with highly scalable, concurrent application requirements. In this section, we outline the main features and architecture of these research projects.

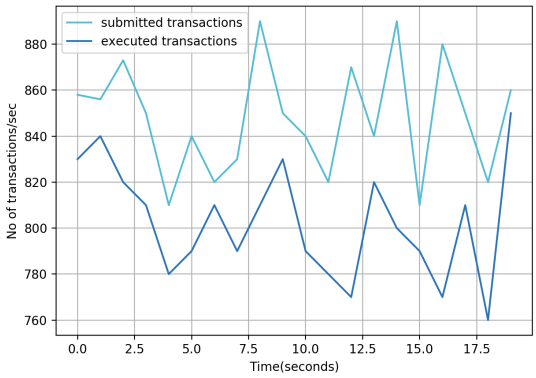

Figure 10: Transaction execution rate and transaction submission rate in a Bassa single blockchain node.

BigchainDB [24], HBasechainDB [25] and Hyperledger Fabric [3] blockchains are trying to increase the scalability and transaction throughput of the blockchain by providing new blockchain architectures. They are providing alternatives for traditional Order-Execute(O-E) blockchain architecture which is the major performance bottleneck in existing blockchains. BigchainDB and HbasechainDB blockchains are mainly targeted for big data applications. Built on top of MongoDB, Bigchaindb is built as an enterprise blockchain database. MongoDB has the ability to handle and perform the consensus within the database. HBasechaindb is a similar kind of blockchain database to BigchainDB. Instead of MongoDB, Apache HBase with Hadoop is used. Both Bigchaindb and HbasechainDB use blockchain pipelining architecture and majority vote federated consensus to generate the blocks. Hyperledger Fabric is a permissioned blockchain system using a modular design approach that allows scalability, extensibility and flexibility. It comes in different consensus algorithms which can be configured `Kafka`, `RBFT`, `Sumeragi`, `and PoET`. HLF introduced Execute-Order-Validate architecture to support high transaction throughput on the blockchain.

RapidChain [2] and RSCoin [26] blockchains are trying to increase the scalability requirements by providing sharding based blockchain protocols. They are trying to get rid of full-node data replication which is another major bottleneck in existing blockchains. Rapidchain claims to be the first Byzantine-Fault sharding-based public blockchain protocol [27]. It partitions the data and distributes them into multiple `committer` nodes(sharding). RapidChain is able to avoid gossiping transactions to the entire network by using a cross-shard transaction verification technique that ends up being more efficient. Another sharding based blockchain system and protocol is RSCoin that is used for the scalability of centrally-banked crypto-currencies. It utilizes a centralized monetary supply along with a distributed transaction ledger. The validation of transactions on the distributed ledger is controlled by a set of authorities referred to as `mintettes`. By having a centralized monetary authority,

RSCoin addresses scalability issues in decentralized cryptocurrencies.

Lightchain [28] and LSB(Lightweight Scalable Blockchain) [29] are mainly targeted for IoT application. They are trying to increase the scalability and transaction throughput of the blockchain to support IoT application scenarios. Lightchain is a lightweight blockchain system which is resource-efficient and suitable for power-constrained IIoT scenarios. Lightchain mainly considers three significant aspects of the blockchain: computing power consumption, storage space usage, and network resources usage. It presents a green consensus mechanism named Synergistic Multiple Proof(SMP) for stimulating the cooperation of IIoT devices. To avoid broadcasting entire blocks through the network, Lightchain utilizes a lightweight data structure called LightBlock (LB) to streamline broadcast content. LSB is a lightweight and scalable blockchain storage that is optimized for IoT requirements. It solves five main challenges when integrating existing blockchain platforms within the IoT environment. These challenges are complex consensus algorithms, Scalability and overheads, Latency, Security overheads, and Throughput. To meet the requirements of blockchain for IoT, LSB incorporates several optimizations which include a Lightweight Consensus(LC) algorithm, a distributed trust method, a distributed throughput management strategy, and a separation of the transaction traffic from the data flow.

SpeedyChain [30] and MASSB(Microservice-enabled Architecture for Smart Surveillance using Blockchain Technology) [31] proposed blockchain-based platforms for smart city applications. SpeedyChain is a blockchain-based framework for smart cities that focuses on decentralizing the the use and storage of smart city data. By doing this, the data exists in an immutable managed way that ensures greater resilience to the data used by all. Smart vehicles are intended to use and share their data on this system providing a more trust in the data while also maintaining privacy. The proposed framework uses a blockchain design which separates the data from the block headers giving it a faster performing system. MASSB proposed a blockchain-based secure smart surveillance system for smart cities. Object detection, tracking and feature extraction are achieved for the surveillance systems using cooperative micro-services running independently. Video analysis is secured and maintains synchronicity with the blockchain based database throughout the distributed micro-services network environment.

The comparison summary of these blockchain platforms and Bassa platform presented on Table 1. It compares, Blockchain type(public, private), Architecture, Consensus, Scalability level, Smart contract support, Full-text search support, Concurrent transaction support and Sharding details.

6. Conclusions and Future Work

With Bassa, we have developed a blockchain for highly scalable, concurrent applications such as smart cities. We introduced Validate-Execute-Group blockchain architecture to achieve real-time transaction on the blockchain and exhibits high transaction throughput and high scalability. The functional programming and actor based smart contract platform in Bassa supports concurrent execution of transactions. We added full-text search capability into Bassa by indexing transactions and blocks on Apache Lucene index-based search API. The machine learning service in Bassa enables to build supervised/unsupervised ML models by using the federated-learning approach. It makes smart city data more secure and meaningful where real-time data analytic and anomaly detection can be easily performed. The microservices-based architecture and Docker/Kubernetes based deployments enable fast deployment and easy scalability of Bassa blockchain. The evaluation has been proven the high scalability and transaction through features on Bassa blockchain toward the Smart City based scalable applications. For future developments, we are planning to support Self Sovereign Identity [32] with Zero-Knowledge Proof in Bassa blockchain.

Acknowledgements

This work is funded by the Department of Energy (DOE) Office of Fossil Energy (Federal Grant #DE-FE0031744).

References

[1] A. Destounis, G. S. Paschos, and I. Koutsopoulos, "Streaming big data meets backpressure in distributed network computation," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[2] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: A fast blockchain protocol via full sharding," *IACR Cryptology ePrint Archive*, vol. 2018, p. 460, 2018.

[3] 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*. ACM, 2018, p. 30.

[4] E. Bandara, W. K. NG, K. De Zoysa, and N. Ranasinghe, "Aplos: Smart contracts made smart," *BlockSys'2019*, 2019.

[5] R. O'Connor, "Simplicity: A new language for blockchains," in *Proceedings of the 2017 Workshop on Programming Languages and Analysis for Security*. ACM, 2017, pp. 107–120.

[6] E. Eykholt, G. Meredith, and J. Denman, "Rchain architecture documentation," 2017.

[7] C. Hewitt, "Actor model of computation: scalable robust information systems," *arXiv preprint arXiv:1008.1459*, 2010.

[8] M. Gupta, *Akka essentials*. Packt Publishing Ltd, 2012.

[9] J. Hughes, "Why functional programming matters," *The computer journal*, vol. 32, no. 2, pp. 98–107, 1989.

[10] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[11] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, 2011, pp. 1–7.

[12] A. L. Davis, "Akka streams," in *Reactive Streams in Java*. Springer, 2019, pp. 57–70.

TABLE 1: Blockchain platform comparison

Platform	Public/Private	Architecture	Consensus	Scalability	Smart contract	Full text search	Concurrent transactions	Sharding
Bassa	Private	Validate-Execute-Group	Paxos	High	Yes	Yes	Yes	Yes
BigchainDB [24]	Both	Blockchain-Pipeline	Tendermint	High	Yes	Yes	No	Yes
HbasechainDB [25]	Private	Blockchain-Pipeline	ZAB	High	No	Yes	No	Yes
Hyperledger [3]	Private	Execute-Order-Validate	Kafka/ZAB	Mid	Yes	No	No	No
RapidChain [2]	Public	Order-Execute	Federated	Mid	No	No	No	Yes
RSCoin [26]	Private	Order-Execute	2PC variant	Mid	Yes	No	No	Yes
Lightchain [28]	Public	Order-Execute with Light Block	SMP	Mid	No	No	No	Yes
LSB [29]	Private	Order-Execute with Cluster Heads	LC	High	No	No	No	No
SpeedyChain [30]	Private	Order-Execute With Append Block	PoW	Mid	N/A	N/A	No	No
MASSB [31]	Private	Order-Execute	PoW	Mid	N/A	N/A	N/A	N/A

[13] A. Białecki, R. Muir, G. Ingersoll, and L. Imagination, "Apache lucene 4," in *SIGIR 2012 workshop on open source information retrieval*, 2012, p. 17.

[14] C. Gormley and Z. Tong, *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. " O'Reilly Media, Inc.", 2015.

[15] E. Bandara, W. K. Ng, K. D. Zoysa, N. Fernando, S. Tharaka, P. Maurakirinathan, and N. Jayasuriya, "Mystiko - blockchain meets big data," in *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, 2018, pp. 3024–3032.

[16] J. Thönes, "Microservices," *IEEE software*, vol. 32, no. 1, pp. 116–116, 2015.

[17] "The go programming language." [Online]. Available: https://golang.org/

[18] E. Brewer, "Towards robust distributed systems," in *PODC*, 01 2000, p. 7.

[19] I. L. Traiger, J. Gray, C. A. Galtieri, and B. G. Lindsay, "Transactions and consistency in distributed database systems," *ACM Transactions on Database Systems (TODS)*, vol. 7, no. 3, pp. 323–342, 1982.

[20] N. S. Barghouti and G. E. Kaiser, "Concurrency control in advanced database applications," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 269–317, 1991.

[21] "The scala programming language," https://www.scala-lang.org/, (Accessed on 06/21/2020).

[22] J. Lourenço, B. Cabral, P. Carreiro, M. Vieira, and J. Bernardino, "Choosing the right nosql database for the job: a quality attribute evaluation," *Journal of Big Data*, vol. 2, p. 18, 08 2015.

[23] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, p. 11, 2014.

[24] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto, "Bigchaindb: a scalable blockchain database," *white paper, BigChainDB*, 2016.

[25] M. Subhankar Sahoo and P. K. Baruah, "Hbasechaindb – a scalable blockchain framework on hadoop ecosystem," 03 2018, pp. 18–29.

[26] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," *arXiv preprint arXiv:1505.06895*, 2015.

[27] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[28] Y. Liu, K. Wang, Y. Lin, and W. Xu, "Lightchain: A lightweight blockchain system for industrial internet of things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3571–3581, 2019.

[29] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Lsb: A lightweight scalable blockchain for iot security and privacy," *arXiv preprint arXiv:1712.02969*, 2017.

[30] R. A. Michelin, A. Dorri, M. Steger, R. C. Lunardi, S. S. Kanhere, R. Jurdak, and A. F. Zorzo, "Speedychain: A framework for decoupling data from blockchain for smart cities," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018, pp. 145–154.

[31] D. Nagothu, R. Xu, S. Y. Nikouei, and Y. Chen, "A microservice-enabled architecture for smart surveillance using blockchain technology," in *2018 IEEE International Smart Cities Conference (ISC2)*. IEEE, 2018, pp. 1–4.

[32] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, "A survey on essential components of a self-sovereign identity," *Computer Science Review*, vol. 30, pp. 80–86, 2018.