HBasechainDB – A Scalable Blockchain Framework on Hadoop Ecosystem

Manuj Subhankar Sahoo(✉) and Pallav Kumar Baruah

Sri Sathya Sai Institute of Higher Learning, Anantapur 515134, Andhra Pradesh, India
subhankar.3eblue@gmail.com, pkbaruah@sssihl.edu.in

Abstract. After the introduction of Bitcoin, blockchain has made its way through numerous applications and been adopted by various communities. A number of implementations exist today providing a platform to carry on business with ease. However, it is observed the scalability of blockchain still remains an issue. Also, none of the framework can claim the ability to handle Big Data and support to perform analytics, which is an important and integral facet of current world of business. We propose HBasechainDB, a scalable blockchain-based tamper-proofed Big Data store for distributed computing. HBasechainDB adds the blockchain characteristics of immutability and decentralization to the HBase database in the Hadoop ecosystem. Linear scaling is achieved by pushing computation to the data nodes. HBasechainDB comes with inherent property of efficient big data processing as it is built on Hadoop ecosystem. HBasechainDB also makes adaptation of blockchain very easy for those organizations whose business logic are already existing on Hadoop ecosystem. HBasechainDB can be used as a tamper-proof, decentralized, distributed Big Data store.

Keywords: Blockchain · HBase · Big Data · Tamperproof Immutability

1 Introduction

A Blockchain is a distributed ledger of blocks which records all the transactions that have taken place. It was first popularized by a person or a group under the pseudonym Satoshi Nakamoto, in 2008 by introducing Bitcoin [11]: A Peer-to-Peer Electronic Cash System. This technology revolutionized the decentralized paradigm by introducing and using a Consensus mechanism: Proof-of-Work (PoW). Proof-of-Work defines the requirement of an expensive calculation also called mining, to be performed so as to create a new trustless set of transactions, also called blocks on the blockchain. The major breakthrough for Bitcoin was the hash based blockchain which made the blocks of transactions tamper-proof, transparent and aversive to DoS attack.

Blockchains can support a variety of applications like decentralized financial services, Internet-of-Things [12], smart properties, etc. Several works have centered around the evaluation of potential use cases for the blockchain [3,9,13].

R. Yokota and W. Wu (Eds.): SCFA 2018, LNCS 10776, pp. 18–29, 2018.
https://doi.org/10.1007/978-3-319-69953-0_2

Blockchains can also be seen as a tamper-proof, decentralized data store of a variety of data including government deeds, land ownership records, stock market transactions, etc. However, the PoW consensus protocol enforces major performance bottlenecks on the blockchain. Transaction latencies are as high as an hour and the theoretical peak transactions throughput is just 7 transactions per second. Further, the full nodes in the Bitcoin network which are capable of validating blocks and transactions are expected to maintain copies of the entire Bitcoin blockchain. This places heavy storage demands on the participating nodes. The Bitcoin blockchain is about 136 GB at the time of writing [1]. The communication model also places heavy demands on network bandwidth. The Bitcoin blockchain is also not scalable in that an increase in the number of nodes in the Bitcoin network does not help in increasing the network throughput, latency or capacity. As pointed out by Croman et al. [7], with the increasing adoption of blockchains, we need to address concerns about their scalability.

Apart from digital currency, blockchain technology has taken its own way through many types of industries which includes Finance and Accounting, Supply Chain and Logistics, Insurance etc. For example, financial institutions can settle securities in minutes instead of days. Manufacturers can reduce product recalls by sharing production logs with original equipment manufacturers (OEMs) and regulators. Businesses of all types can more closely manage the flow of goods and related payments with greater speed and less risk. For this reason various industries are trying to adopt blockchain for running their business process smoother and faster. Hyperledger Fabric and BigchainDB are the most widely used framework for blockchainifying business processes.

Today's business processes doesn't generate just data, they generates huge amount of data of wide variety with thrilling velocity. After putting these kinds of data on blockchain it is very important to process and analyze these data in an efficient way. Hadoop is a classic ecosystem which provides numerous functionalities with high efficiency for processing and analyzing these kind of data. A lot of business logic already exists in Hadoop ecosystem to process and analyze these data. Therefore it will be much more easier for the industries to adopt blockchain technology if there exists a scalable blockchain framework in Hadoop ecosystem. Towards this end, HBasechainDB is a first step towards providing a scalable blockchain framework in the Hadoop ecosystem. HBasechainDB is started by High performance Computing and Data (HPCD) Group, Department of Mathematics and Computer Science (DMACS), Sri Sathya Sai Institute of Higher Learning. This is achieved by imparting the blockchain characteristics of immutability and decentralization to the HBase database.

2 Background and Related Work

A lot of work has been underway for addressing the scalability of blockchains. Vukolic [14] has contrasted PoW-based blockchains to those based on BFT state machine replication for scalability. Eyal et al. [8] introduces Bitcoin-NG as a scalable blockchain protocol based on BFT consensus protocols. These approaches

are focused upon improving the consensus protocol. McConaghy et al. [10] adopted a different approach to scalability. They started with a distributed database, MongoDB, and added the blockchain features of decentralized control, immutability while supporting the creation and movement of digital assets to provide a scalable, decentralized database BigchainDB. The major contribution of BigchainDB that enables this scalability is the concept of blockchain pipelining. In blockchain pipelining, blocks are added to the blockchain without waiting for the current block to be agreed upon by the other nodes. The consensus is taken care of by the underlying database. The validation of blocks is not done during block addition but eventually by a process of voting among nodes. This has huge performance gains and BigchainDB has points to transaction throughputs of over a million transactions per second and sub-second latencies.

In creating HBasechainDB, we have adopted an approach similar to that of BigchainDB. Instead of using MongoDB as the underlying database, we use the Hadoop database, Apache HBase. Apache HBase is a distributed, scalable Big Data store. It supports random, real-time read/write access to Big Data. Apache HBase is an open-source, distributed, versioned, non-relational, column-family oriented database modeled after Google's Bigtable [6]. HBase provides both linear and modular scaling, along with strongly consistent reads/writes. HBase tables are distributed on the cluster via regions. HBase supports automatic sharding by splitting and re-distributing regions automatically as data grows. HBasechainDB is a scalable, decentralized data store akin to BigchainDB.

3 Terminology

- **Blockchain:** A chain of blocks where every block has a hash link to the previous block i.e. every block stores the hash of the previous block. An advantage is, just by storing the hash of the last block we can easily detect if any change has been made to any of the block.
- **Double spending:** It is an attack where the asset is spent in more than one transaction. To prevent double spending, blockchain framework needs to check whether a particular asset is spent in any of the previous transactions. For instance: user U2 wants to spend/transfer an asset A1, in transaction T2, to another user U3. Say, the asset A1 was transferred to U2 by user U1 in some previous transaction T1. U2 specifies T1's Id in T2, which shows that T1 was the transaction which contained asset A1, and U2 got it from U1. Now, U2 wants to spend/transfer it to U3. So before validating transaction T2 with asset A1, a Blockchain framework checks all the transaction with asset A1 that has occurred/lies in between T1 and T2, in order. If A1 does not occur in any of the transactions then A1 is not double spent else it's double spent.
- **Blockchain Pipeline:** In Blockchain pipelining, blocks are written to the underlying database without waiting for a vote which confirms the block's validity. Voting for a block and forming a chain happens as a separate layer.

- **Changefeed:** It is a mechanism by which any update on the Blockchain is notified to the nodes. This automatic change notifications on the Blockchain brings another benefit: they improve tamper-detection (beyond what a blockchain offers). If a hacker somehow manages to delete or update a record in the data store, the hashes change (like any blockchain). In addition, a data-store with automatic change notifications would notify all the nodes, which can then immediately revert the changes and restore the hash integrity.
- **HBase region-server:** These are the basic elements of availability and distribution for tables, and are comprised of a store per Column Family.
- **HBase Master:** Responsible for coordinating Regions in the cluster and execute administrative operations.

4 Architecture

4.1 Data Model of Transaction

The Transaction Model of all Blockchain Platforms has three important fields: Transaction Id, List of Inputs, List of Outputs. Apart from these, there are fields which are platform dependent. HBasechainDB's transaction model consists of; Transaction Id, Asset, List of Inputs, List of Outputs and Metadata.

1. **ID:** The transaction Id uniquely identifies the transaction. It is a SHA3-256 hash of the asset, list of inputs, list of outputs and metadata.
2. **Asset:** A JSON format document associated with a transaction.
3. **List of Inputs:** Each input in the list of a transaction is spendable/transferable if it has a link to the output of some previous transaction (in case of transfer transaction. Creation transaction doesn't have link to out of any previous transaction). This input is then spent/transferred by satisfying/fulfilling the crypto-conditions on that previous transaction output. A CREATE transaction should have exactly one input. A TRANSFER transaction should have at least one input (i.e. $\geq$1).
4. **List of outputs:** Each output in the list of a transactions indicates the crypto-conditions which must be satisfied by anyone who wishes to spend/transfer that output to some other transaction. It also indicates the number of shares of the asset tied to that output.
5. **Metadata:** User-provided transaction metadata. It can be any valid JSON document or NULL.

4.2 Design Details

HBasechainDB is a super peer-to-peer network operating using a federation of nodes. All the nodes in the federation have equal privileges which gives HBasechainDB its decentralization. Such a super peer-to-peer network was inspired by the Internet Domain Name System. Any client can submit or retrieve transactions or blocks, but only the federation nodes can modify the blockchain. The federation can grow or shrink during the course of operation of

HBasechainDB. Let us say there are n federation nodes N1, N2, ..., Nn. When a client submits a transaction t, it is assigned to one of the federation nodes, say Nk. The node Nk is now responsible for entering this transaction into the blockchain. Nk first checks for the validity of the transaction. Validity of a transaction includes having a correct transaction hash, correct signatures, existence of the inputs to the transaction, if any, and the inputs not having been already spent. Once Nk has validated a set of transactions, it bundles them together in a block, and adds it to the blockchain. Any block can only contain a specified maximum number of transactions. Let us say t was added in the block B.

When the block B is added to the blockchain its validity is undecided. Since the federation is allowed to grow or shrink during the operation of HBasechainDB, blocks also include a list of voters based on the current federation. All the nodes in the voter list for a block vote upon B for its validity. For voting upon a block, a node validates all the transactions in the block. A block is voted valid only if all the transactions are found to be valid, else it is voted invalid. If a block gathers a majority of valid or invalid votes, its validity changes from undecided to valid or invalid respectively. Only the transactions in a valid block are considered to have been recorded in the blockchain. The ones in the invalid blocks are ignored altogether. However, the chain retains both valid and invalid blocks. A block being invalid does not imply that all the transactions in the block are invalid. Therefore, the transactions from an invalid block are re-assigned to federation nodes to give the transactions further chance of inclusion in the blockchain. The reassignment is done randomly. This way, if a particular rogue node was trying to add an invalid transaction to the blockchain, this transaction will likely be assigned to a different node the second time and dropped from consideration. Thus, if block B acquires a majority of valid votes, then transaction t would have been irreversibly added to the blockchain. On the other hand, if B were invalid, then t would be reassigned to another node and so on until it is included in the chain or removed from the system.

As discussed in the previous section, the chain is not formed when blocks are created. When a block is entered into hbasechain table, the blocks are stored in HBase in the lexicographical order of their ids. The chain is actually formed during vote time. When a node votes on a block, it also specifies the previous block that it had voted upon. Thus, instead of waiting for all the federation nodes to validate the current block before proceeding to the creation of a new block, blocks are created independent of validation. This is the technique of blockchain pipelining described earlier. Over time, the blockchain accumulates a mix of valid and invalid blocks. The invalid blocks are not deleted from the chain to keep the chain immutable. What we also note here is that while it would seem that different nodes could have a different view of the chain depending upon the order in which they view the incoming blocks, it is not seen in practice in HBasechainDB due to the strong consistency of HBase and the fact that the blocks to be voted upon are ordered based on their timestamp. Thus, each node sees the same order of blocks, and we have the same chain view for different nodes.

To tamper with any block in the blockchain, an adversary will have to modify the block, leading to a change in its hash. This changed hash would not match the vote information for the block in the votes table, and also in subsequent votes that refer to this block as the previous block. Thus an adversary would have to modify the vote information all the way up to the present. However, we require that all the votes being appended by nodes are signed. Thus, unless an adversary can forge a node's signature, which is cryptographically hard, he cannot modify the node's votes. In fact, he has to forge multiple signatures to affect any change in the blockchain preventing any chances of tampering. This way HBasechainDB provides a tamper-proof blockchain over HBase.

4.3 Exploiting HBase

In this section we describe the distinction between MongoDB and HBase. We also justify the means to achieve greater performance with the proposed system design.

MongoDB is a document store database. A document is a big JSON block with no particular schema or format. This gives an edge to dynamic use cases and ever-changing applications. MongoDB does not provide triggers. Although MongoDB has its own advantages, the document store characteristic of MongoDB degrades its performance for following operations:

1. Working with individual columns.
2. Performing join operations.

HBase is a wide column store database. It is a distributed, scalable, reliable, and versioned storage system capable of providing random read/write access in real-time. It provides a fault-tolerant way of storing large quantities of sparse data. HBase features compression, in-memory operation and Bloom filters on a per-column basis.

We use the following characteristics of HBase extensively to derive performance:

1. HBase is partitioned to tables, and tables are further split into column families. Column families must be declared in the schema, and we can group certain set of columns together. One of the major operations in blockchain transaction is checking for Double-Spending. In order to make the check for double spending more efficient, we can keep the input column of all these transactions in a separate column family. This will allow us to perform the check for double spending faster because the region server will need to load only one column family which contains the input of the transaction. In case of database such as MongoDB the database server needs to load the whole document before filtering out the input column and performing Double Spent check.
2. HBase is optimized for reads, supported by single-write master, which results in a strict consistency model. And use of Ordered Partitioning supports row-scans. In Blockchain we need one write and many read operation because

the transactions are written only once but read many times for various purposes like checking double spending and performing checks on whether any tampering took place.

3. HBase provides us with various ways in which we can run our custom code on the region-server. HBase co-processor and custom filters are two such ways. HBase co-processor can act as database triggers. In our implementation we use these features in following ways:
   (a) The check for double spending is generally done by loading the transactions to the federation nodes(i.e. the client system). Loading this many transactions from region-server to the federations node system is a major bottleneck for the system throughput. In our approach, instead of pulling the data required for double spending check on to the client-system, we push the computation check to the region-server using HBase custom filter. This approach improves the performance in two ways:
      i. Data does not move towards the computation node rather computation moves towards the Data node. Since the code size is exponentially lesser than data size, we improve the system by decreasing the communication time.
      ii. Computation for double spending is done in parallel on multiple region-server compared to the traditional approach of checking on a single Client node.
   (b) Changefeed brings a great benefit to the Blockchain framework. We use HBase co-processor to implement changefeed which will notify immediately whenever a hacker tries to change or delete the content of the database.

5 Implementation Details

The Federation Nodes in HBasechainDB are initialized with a key-pair; Ed25519 [2,4] signing system. SHA3-256 [5] hashing scheme is used for hashing the transactions and blocks. The current implementation of HBasechainDB uses six HBase tables. A critical issue in the current design of HBase tables is that of designing the row key, since the region splits and the scans on HBase tables are done in the lexicographical order of the row key. The row key pattern depends upon the access pattern for the data in the Hbase table.

Following is the description of the HBase tables:

1. backLog: When a transaction is submitted to the Federation nodes, the transaction is randomly assigned to one of the nodes. All such assigned transactions are stored in the backlog table with each transaction stored in a single row. A node scanning the backlog table should only have to read the transactions assigned to itself. Thus, the first segment of the row key for backlog table is the public key of the node to whom the transaction was assigned, to ensure that a node can scan the backlog table with the row prefix being its own public key. The last segment of the row key contains the transaction reference id. So the row key looks like: <publicKey>_<transactionId>

2. block: This is the table that contains all the blocks in the blockchain. Each block is a logical block which contains only the id's of the transaction which are present in the block. The actual transaction details are stored in `"hbasechaindb"` table. Since the access pattern for this table is looking up blocks based on block id, the row key for this table is just the block id: <blockId>
3. hbasechaindb: This is the table where all the transaction details are stored after a transaction is put on the blockchain. In this table each row corresponds to a single transaction. Since the access pattern for this table is looking up transaction based on transaction link id, the row key of this table is <transaction link id>. The transaction link id consists of <block_id>_<transaction_id>. This transaction link id which is of previous output is used in inputs of current transaction while spending an asset
4. toVote: Every new block created has to be voted upon by the Federation nodes. For this, we need to inform the Federation nodes of their need to vote upon a newly created block. To this end, every block created is added to this table to signal the node for voting. It is removed from the table once the node has finished voting on it. The row key of this table is : <federation node's signing key>_<block id>
5. vote: This is the table in which all the votes are recorded. There has to be an entry for every federation node which votes for their respective blocks. The row key of the table is: <block_id>_<decision>_<Fed. Node public key>
6. reference: This is the table which stores the map between transaction link id and transaction id. This table acts as an index when the details of a transaction is queried. Since the access pattern of the table is transaction reference id, the row key of this table is just the transaction reference id: <transacation_link_Id>

When a transaction is submitted to HBasechainDB, it is first put in the *backLog* table. Federation nodes picks the transactions from *backLog* table in certain time interval, checks the validity of the transactions, bundles them into blocks and adds those blocks to the Blockchain. As show in Fig. 1, when a federation node forms a block, it updates 3 HBase tables. In *block* table, the transaction_Id of all the transactions are made as separate blocks and stored. In the *hbasechaindb* table, all the transaction details are stored. In the *toVote* table the information about newly created block is stored. The federation nodes refers this *toVote* table to vote for the block. All the Federation nodes, in certain time interval checks the *toVote* table and cast their vote after checking the validity of the block. All the votes are stored in the *vote* table. After the validity of a block, entries corresponding to all the transactions are made in the *reference* table.

The complete implementation of HBasechainDB is done using Java since the performance of HBase API for Java is best among the HBase API's present for different languages. HBase API for Java also gives advantage of writing custom filters and coprocessors.

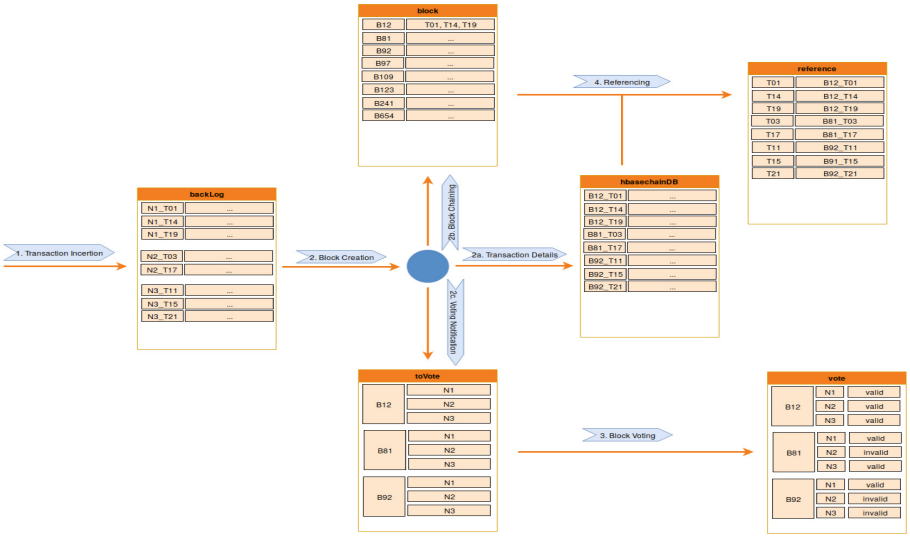

Fig. 1. Transaction flow of HBasechainDB

6 Performance

6.1 Experimental Setup

We have used three nodes for the initial performance testing of HBasechainDB with the following configurations:

- 3 nodes with Intel Core i5-4670 CPU @ 3.40 GHz*4 processor and 16 GB of memory, with Ubuntu 16.04 OS.
- Each of the 3 nodes runs HBase region-server. There is a HBase master running in one of the system.
- There is a Replication factor of 3 for the underlying HDFS.
- The HBase is backed by 3 quorum zookeeper.
- We consider only creation of transactions for our case.

6.2 Results

We have tested HBasechainDB for scalability over three nodes. There are two parameters that we describe the performance of HBasechainDB with:

- **Transaction Latency:** This is defined as the time elapsed since the submission of a transaction to HBasechainDB until the block in which it has been recorded is validated. The transaction latency is found for streaming transactions.
- **Throughput:** This is the number of transactions that are recorded in the blockchain per second. To find the peak throughput the blockchain is capable of, we store the transactions in the backlog beforehand and then run the nodes. The throughput observed then is the peak throughput.

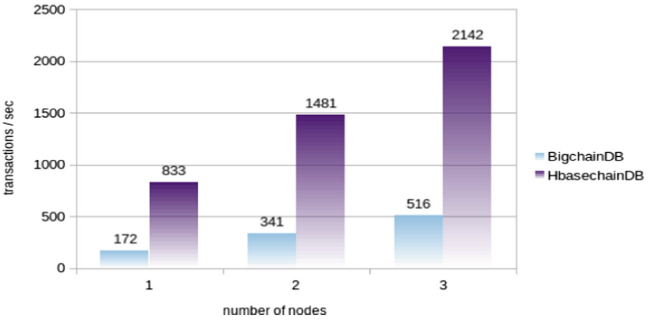

Fig. 2. Performance of BigchainDB and HBasechainDB

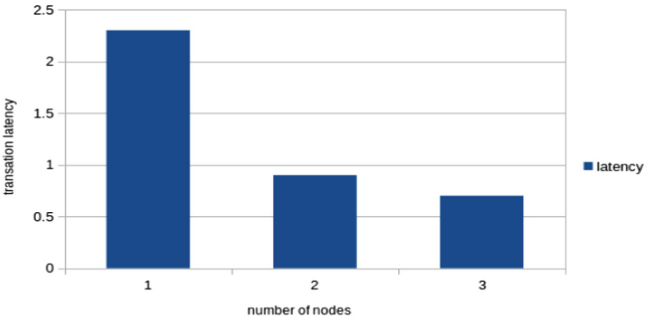

Fig. 3. Latency of HBasechainDB in sec.

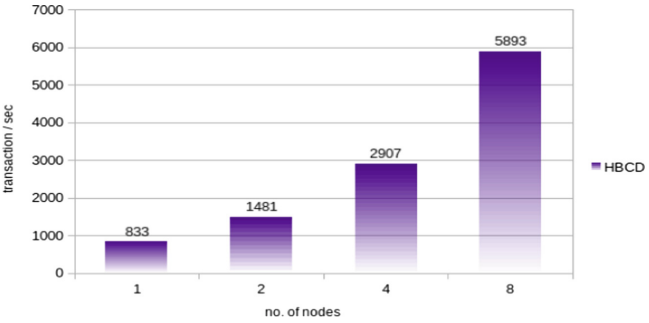

Fig. 4. Scalability of HBasechainDB upto 8 nodes

Figure 2 compares the transaction throughput of HBasechainDB and BigchainDB, using systems with 1, 2 and 3 nodes and Fig. 3 shows the latency of HBasechainDB. Figure 4 shows the scalability of HBasechainDB till 8 nodes. The result shows, as we add the nodes the transaction throughput of HBasechainDB scales linearly.

The main reason behind the linear scale of HBasechainDB is, almost all the computation which includes *computation for transaction's validity* and *check for double spending* is pushed to server side. Therefore if we increase the HBase nodes keeping the federation node constant, the system scales linearly.

7 Conclusion

Blockchain technologies can be very useful in the Big Data scenario by helping us immutably record data and decentralizing data services. However, current blockchain implementations with their extremely low transaction throughputs and high transaction latencies do not lend themselves to Big Data. Discussions on improving blockchain scalability have largely focused on using better consensus protocols as against the PoW protocol used by Bitcoin. BigchainDB provides an alternative idea where instead of scaling blockchains to provide scalable data stores, they implement a blockchain over an existing scalable distributed database. Such an implementation inherits the scalability of the underlying database, while adding the immutability and decentralization offered by blockchains. While BigchainDB was implemented upon the MongoDB and RethinkDB database, with our work we provide an alternate implementation over HBase. HBasechainDB is an hitherto unavailable blockchain implementation integrated with the Hadoop ecosystem. It supports very high transaction throughputs with sub-second latencies and the creation and movement of digital assets. HBasechainDB scales linearly and also is good platform for analyzing data that are present on blockchain.

Acknowledgments. Our work is dedicated to Bhagawan Sri Sathya Sai Baba, Founder Chancellor of Sri Sathya Sai Institute of Higher Learning. We acknowledge Adarsh Saraf from IBM Research, Bengaluru, India, who has initiated this work. We thank him for his inspiration and motivation. We also acknowledge Maestro Technology, USA for their help and support.

References

1. https://blockchain.info/charts/blocks-size
2. https://github.com/str4d/ed25519-java/tree/master/src/net/i2p/crypto/eddsa
3. Aron, J.: Automatic world (2015)
4. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. J. Cryptographic Eng. **2**(2), 1–13 (2012)
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak specifications. Submission to NIST (Round 2) (2009)
6. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. ACM Trans. Comput. Syst. (TOCS) **26**(2), 4 (2008)
7. Croman, K., et al.: On scaling decentralized blockchains. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 106–125. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_8

8. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-NG: a scalable blockchain protocol. In: NSDI, pp. 45–59 (2016)
9. Liebenau, J., Elaluf-Calderwood, S.M.:. Blockchain innovation beyond bitcoin and banking (2016)
10. McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., Henderson, R., Bellemare, S., Granzotto, A.: BigchainDB: a scalable blockchain database. White paper, BigChainDB (2016)
11. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
12. Panikkar, S., Nair, S., Brody, P., Pureswaran, V.: Adept: an IoT practitioner perspective. IBM Institute for Business Value (2014)
13. Swan, M.: Blockchain: Blueprint for a New Economy. O'Reilly Media Inc., Sebastopol (2015)
14. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In: Camenisch, J., Kesdoğan, D. (eds.) iNetSec 2015. LNCS, vol. 9591, pp. 112–125. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39028-4_9