A BigchainDB Primer

BigchainDB GmbH, Berlin, Germany

May 2017

Abstract

BigchainDB is a scalable blockchain database: a big-data database with blockchain characteristics including decentralization, immutability and built-in support for creation & transfer of assets. Decentralized protocols and technologies have always been a part of the internet, but the introduction of Bitcoin ignited a new wave of interest in their capabilities. In particular, Bitcoin showed that it's possible to create a trustable, immutable record of transactions (a ledger) without relying on a central, trusted third party. Many people realized that the ideas ("blockchain technologies") behind Bitcoin could be used, in theory, to store trustable records of any kind. Unfortunately, Bitcoin itself can't scale to handle all those records, at least not in its current form. There are many ideas and prototypes to scale Bitcoin and similar blockchains. BigchainDB takes a different approach to solving the scaling problem. It starts with a proven, scalable, big-data database and then adds blockchain characteristics.

1 Overview

BigchainDB is a scalable blockchain database: a big-data database with blockchain characteristics including decentralization, immutability and built-in support for creation & transfer of assets. This primer[1] explains why something like BigchainDB was needed, summarizes how it adds blockchain characteristics, explores how it fits into the rest of the decentralization ecosystem, and concludes with a section about how you can try it.

2 Bitcoin and Other Decentralized Protocols

One goal of the early internet (ARPAnet) was to create a communications network that could withstand nuclear war [2]. To meet that goal, the designers removed the need for a central owner or controller, a single point of failure. The core internet protocols are decentralized. Many other internet-based protocols are similarly decentralized, at least in part, e.g. email, DNS, and BitTorrent. Over time, however, many centralized services and APIs have become prevalent on the internet (e.g. Facebook and Google search). Bitcoin [3], introduced in 2009, is a decentralized protocol which sparked a new wave of interest in such protocols. The reason was that it demonstrated a working decentralized cryptocurrency requiring no central bank or other trusted third party. Trust

[1] The BigchainDB Whitepaper [1] goes into much more detail.

emerges from the protocol and its one lasting artifact: the long-term Bitcoin blockchain—a record of every valid/finalized bitcoin transaction, copied consistently and verifiably to thousands of computers all over the world.

3 Blockchains for Records of Other Things?

If Bitcoin can generate a trustable record of bitcoin transactions, without needing a central owner or controller, can Bitcoin or a similar technology be used to generate a trustable record of other trust-important information? Consider birth certificates, college transcripts, patent filings, land ownership transactions, art ownership transactions[2], supply-chain records, lab reports, and government transparency reports. Some of those examples can generate new records very fast. How much could Bitcoin handle?

4 The Scale of Bitcoin vs. Modern Databases

The Bitcoin network of today records less than ten transactions per second (on average), and adding more nodes won't change that: write throughput doesn't scale with node count. One *can* store arbitrary information in a Bitcoin transaction, but not much: dozens of bytes. The total storage capacity of the Bitcoin blockchain grows by about one megabyte every ten minutes, but that's *all* the data, not just the arbitrary data. Moreover, total storage capacity *doesn't* grow when more nodes are added to the network. It takes tens of minutes to hours before a Bitcoin transaction can be considered finalized (i.e. part of the long-term blockchain).

Now consider a typical modern "big-data" database. Write throughput can be in the millions of transactions per second (or more), and it can grow when you add more nodes. For example, Netflix tested Cassandra in 2011 and the results are shown in Figure 1. At the bottom left of the plot, we see that 50 distributed Cassandra nodes could handle 174,000 writes per second. Increasing to 300 nodes allowed for 1.1 million writes per second [4]. A follow-up study three years later showed a throughput of 1 million writes per second with just a few dozen nodes [5].

Consider other properties of modern databases: individual records can be fairly large (megabytes) and total storage capacity can grow as you add more nodes (to petabytes and beyond). Transaction latency can be hundreds of milliseconds in a globe-spanning distributed database (and milliseconds in a single data center, determined mostly by the speed of light). Moreover, all modern databases have support for rich querying and permissions.

Storage cost is a major limitation of using Bitcoin. By early 2017, if you wanted a short Bitcoin transaction confirmation time, you had to pay around 25 US cents in transaction fees [6], which is around 1 US cent per byte of arbitrary data stored, or ten million US dollars per gigabyte. Compare that to the cost of Amazon Glacier (used for long-term, secure, durable data storage): currently 0.004 US dollars per gigabyte-month, so 40 US dollars to store a

[2]BigchainDB began as a project within ascribe.io, which offers a service to record art attribution, transfer, consignment and loan records on the Bitcoin blockchain.

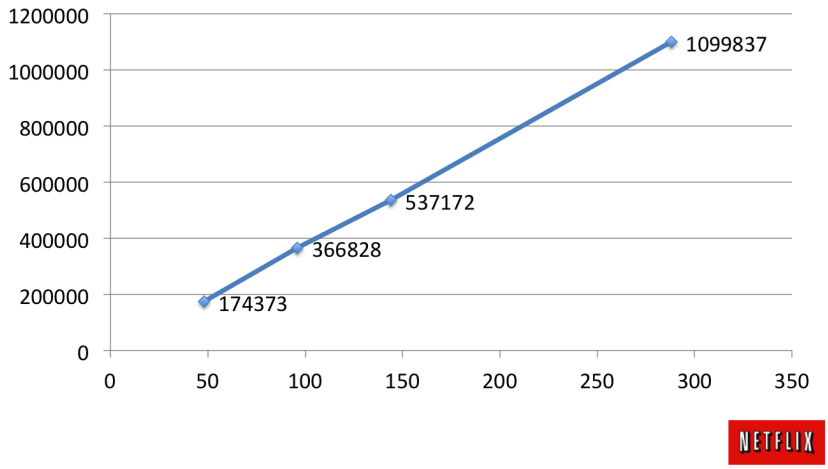

Figure 1: Netflix experimental data on write throughput of its Cassandra database (Client writes/s by node count; Replication Factor=3). The x-axis is number of nodes; the y-axis is writes per second. From [4].

gigabyte for over 800 years (assuming storage costs stay constant, whereas in fact they are likely to go down over time.)

There are many proposals to tweak or improve Bitcoin and similar blockchains, to increase throughput, increase scalability, increase storage capacity, or reduce latency. In short, they start with a blockchain and add database characteristics.

5 The BigchainDB Solution

BigchainDB takes a different approach. It's built by starting with a big-data database, then adding blockchain characteristics including decentralization, immutability and built-in support for creation & transfer of assets.

	Typical Blockchain	Typical Distributed Database	BigchainDB
High Throughput; increases with node count		✓	✓
Low Latency		✓	✓
High Capacity; increases with node count		✓	✓
Rich querying		✓	✓
Rich permissioning		✓	✓
Decentralization	✓		✓
Immutability	✓		✓
Built-in support for creation & transfer of assets	✓		✓

Currently, one can use MongoDB or RethinkDB as the database backend, but we recommend MongoDB: a trusted, enterprise-grade database with high perfomance, strong consistency and modern tooling [7]. Figures 2 and 3 illustrate the architecture of a BigchainDB federation/cluster. An external client can communicate with any instance of BigchainDB Server in the cluster. Each BigchainDB Server instance is associated with one node in a MongoDB database.

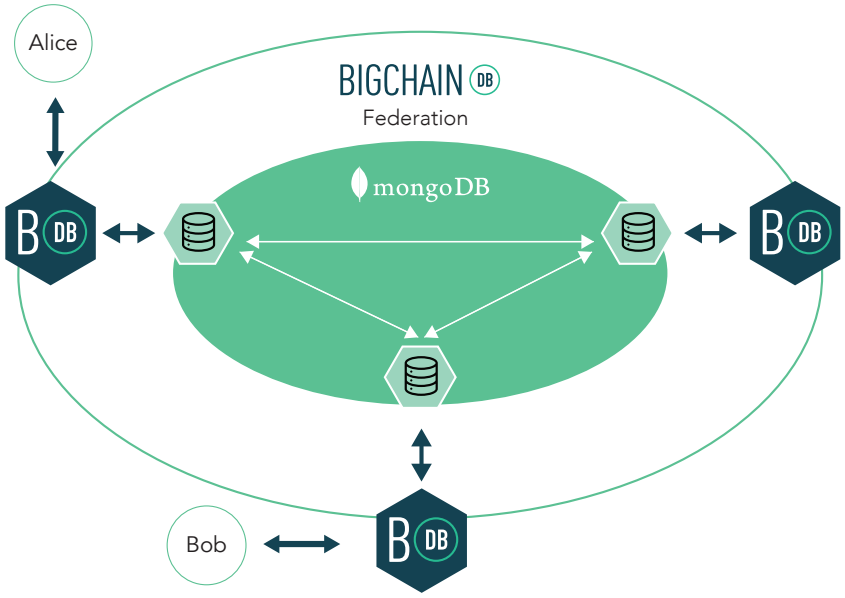

Figure 2: BigchainDB has a two-layer consensus. The lower level (MongoDB) handles transaction ordering and crash faults. The higher level (BigchainDB) federation handles double spends and malicious attacks.

Decentralization. One key requirement is that every node in the cluster should be owned and operated by a different person or organization. That's not handled by the code; it's something that must be enforced by the organization overseeing the BigchainDB cluster. That organization needs rules and processes, for example, to add and remove participants/nodes (i.e. governance is required). The degree of democracy/autocracy can vary from organization to organization. Ideally, the nodes should be located in many countries, legal jurisdictions and hosting providers, so an issue with one doesn't affect them all. As with other blockchains, the only people who can create a valid transfer transaction are the people with the necessary private keys. Nobody else, including node operators, can do that: the power to transact is decentralized. Plans for adding other decentralizing features are outlined in the BigchainDB roadmap [8].

Immutability. Blockchains are very tamper-resistant. Even if tampering does happen, it can be detected (and rejected), mostly because of extensive use of cryptographic signatures, hash chains and Merkle trees. In the blockchain com-

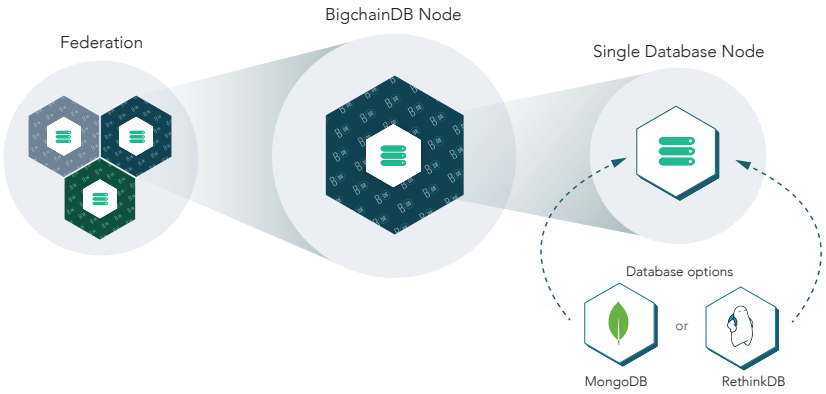

Figure 3: A BigchainDB federation/cluster contains several independent BigchainDB nodes. Each BigchainDB node contains one node of a MongoDB (or RethinkDB) database.

munity, these characteristics are often called "immutability" (which is different from the usual meaning of that word, but language is a living thing, and words often get overloaded with additional meanings). BigchainDB also makes extensive use of cryptographic signatures and hashes. For example, all transactions, blocks and votes are signed. Also, transactions and blocks are identified by their hashes. Tamper-resistance is also increased by having multiple copies of the data (i.e. a high replication factor). It's worth noting that tamper-resistance can also be augmented by standard techniques such as secure backups and secure cluster infrastructure (secure network, secure operating system, etc.). Plans for adding other tamper-resistance features are outlined in the BigchainDB roadmap [8].

Built-in support for creation & transfer of assets. While banks use databases for the transfer of assets all the time, the databases themselves don't usually have any built-in concept of assets: that functionality is added in the application layers. Like most blockchains, BigchainDB *does* have a built-in concept of assets, along with built-in transaction validity checking (e.g. checking signatures, checking for double-spending, checking that the output-being-spent is in a valid transaction, etc.).

Sybil tolerance. Some blockchain networks (such as Bitcoin) allow anyone to add their node to the network. That brings the concern that someone could add so many nodes that they effectively control the network. It's known as a Sybil attack. Bitcoin makes Sybil attacks unlikely by making them prohibitively expensive. In a BigchainDB network, the governing organization behind the network controls the member list, so Sybil attacks are not an issue.

6 BigchainDB in the Decentralization Ecosystem

BigchainDB is complementary to decentralized processing / smart contracts (e.g. Ethereum [9, 10] or Enigma [11, 12]), decentralized file systems (e.g. IPFS [13]),

and communication building blocks (e.g. email). It can be included in higher-level decentralized computing platforms (e.g. Monax/Tendermint [14, 15]). It can be used side-by-side with identity protocols, financial asset protocols (e.g. Bitcoin [3]), intellectual property asset protocols (e.g. SPOOL [16]), and glue protocols (e.g. pegged sidechains [17], Interledger [18]). Scalability improvements to smart-contract blockchains will help fully decentralized applications better leverage the scalability properties of BigchainDB.

BigchainDB works with more centralized computing systems as well. One use case is where decentralizing just storage brings the majority of the benefit. Another use case is where scalability needs are greater than the capabilities of existing decentralized processing technologies; in this case BigchainDB provides a bridge to an eventual fully-decentralized system.

Figure 4 illustrates how BigchainDB can be used in a fully decentralized setting, or as a mild extension from a traditional centralized computing context.

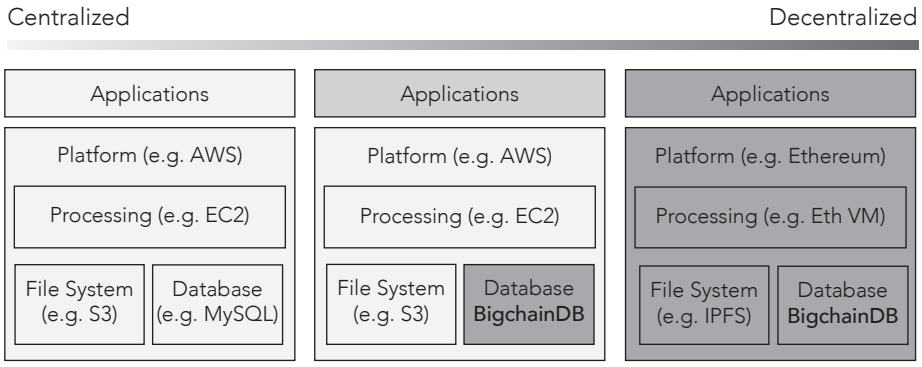

Figure 4: From a base context of a centralized cloud computing ecosystem (left), BigchainDB can be added as another database to gain some decentralization benefits (middle). It also fits into a fully-decentralized system (right).

7 How to Try BigchainDB

If you want to try BigchainDB, then you need a BigchainDB cluster to connect to. One option is to install a single-node BigchainDB "cluster" on your local machine. The Quickstart page in the BigchainDB Server docs [19] has installation instructions. Another option is to connect to an exsiting BigchainDB cluster, such as the IPDB Test Network (IPDB being the Interplanetary Database, which is managed by the IPDB Foundation[20]).

Once you have a BigchainDB cluster to connect to, you can communicate with it via the BigchainDB HTTP API [21]. BigchainDB also provides a WebSocket Event Stream API [22]. There are many tools and libraries to help craft a valid BigchainDB transaction: a CLI, a Python driver, a JavaScript transaction builder, a Java driver, a Go driver, etc. Links can be found in the Drivers & Clients page of the docs [23]. Listing 1 shows some example example Python code using the BigchainDB Python driver [24].

Listing 1: Example Code Using the Python Driver

```
1  from bigchaindb_driver import BigchainDB
2  from bigchaindb_driver.crypto import generate_keypair
3  bdb = BigchainDB('http://example-bdb-cluster.net:9984')
4  alice = generate_keypair()
5  new_asset = {
6      'data': {
7          'piece_name': 'Avocado Green',
8          'artist_name': 'Parker Millson',
9          'edition_number': 9
10     }
11 }
12 prepared_creation_tx = bdb.transactions.prepare(
13     operation='CREATE',
14     signers=alice.public_key,
15     asset=new_asset)
16 fulfilled_creation_tx = bdb.transactions.fulfill(
17     prepared_creation_tx,
18     private_keys=alice.private_key)
19 sent_creation_tx = bdb.transactions.send(fulfilled_creation_tx)
```

Line 4 generates a public/private keypair for a user named `alice`. Lines 5–11 create a Python dict named `new_asset`. Lines 12–15 create a raw (unsigned) BigchainDB CREATE transaction, to create an asset based on the `new_asset` dict, for `alice`. Lines 16–18 sign that transaction using the private key (signing key) of `alice`. Line 19 sends that to the BigchainDB cluster.

There's more example code in the Python driver docs [24].

8 Going to Production

As noted earlier, a production BigchainDB cluster should be operated by several different people or organizations. For example, within a single enterprise, each node could be operated by a different business unit; or within a consortium, each participating organization could run a node. Each operator must install and run a "BigchainDB node" containing an instance of BigchainDB Server, an instance of MongoDB (server), storage for MongoDB, and potentially other components. The BigchainDB docs give more details [25].

9 Conclusion

Many applications could use something decentralized like Bitcoin, but capable of storing large amounts of arbitrary data quickly, immutably, and with low latency. BigchainDB is our answer to that challenge: a big-data database with blockchain characteristics including decentralization, immutability and built-in support for creation & transfer of assets.

For more information, see the BigchainDB website at bigchaindb.com.

References

[1] BigchainDB Whitepaper. `https://www.bigchaindb.com/whitepaper/`.

[2] Cade Metz. Paul Baran, the link between nuclear war and the internet. *WIRED*, September 2012. https://www.wired.co.uk/article/h-bomb-and-the-internet.

[3] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf, 2009.

[4] Adrian Cockcroft and Denis Sheahan. Benchmarking Cassandra Scalability on AWS – Over a million writes per second. 2011. http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html.

[5] C. Kalantzis. Revisiting 1 Million Writes per second. http://techblog.netflix.com/2014/07/revisiting-1-million-writes-per-second.html.

[6] Justin Connell. Bitcoin Transaction Fees Are Up More Than 1200% in Past Two Years. https://news.bitcoin.com/bitcoin-transaction-fees-1200-past-two-years/, February 2017.

[7] MongoDB. https://www.mongodb.com.

[8] BigchainDB Roadmap. https://github.com/bigchaindb/org/blob/master/ROADMAP.md.

[9] Ethereum. https://ethereum.org/.

[10] Vitalik Buterin. Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform. http://blog.lavoiedubitcoin.info/public/Bibliotheque/EthereumWhitePaper.pdf.

[11] Enigma. http://enigma.media.mit.edu/.

[12] Guy Zyskind, Oz Nathan, and Alex Pentland. Enigma: Decentralized Computation Platform with Guaranteed Privacy. 2015. http://enigma.media.mit.edu/enigma_full.pdf.

[13] J. Benet. IPFS – Content Addressed, Versioned, P2P File System. http://static.benet.ai/t/ipfs.pdf, 2014.

[14] Monax. https://monax.io.

[15] Tendermint. http://www.tendermint.com.

[16] Dimitri De Jonghe and Trent McConaghy. SPOOL Protocol. https://github.com/ascribe/spool.

[17] A. Back. Enabling blockchain innovations with pegged sidechains. Technical report, October 2010. http://www.blockstream.com/sidechains.pdf.

[18] S. Thomas and Schwartz E. A Protocol for Interledger Payments. https://interledger.org/interledger.pdf, 2015.

[19] BigchainDB Server Quickstart. https://docs.bigchaindb.com/projects/server/en/latest/quickstart.html.

[20] IPDB Foundation. https://ipdb.foundation/.

[21] BigchainDB HTTP API. https://docs.bigchaindb.com/projects/server/en/latest/http-client-server-api.html.

[22] BigchainDB WebSocket Event Stream API. https://docs.bigchaindb.com/projects/server/en/latest/websocket-event-stream-api.html.

[23] BigchainDB Drivers & Clients. https://docs.bigchaindb.com/projects/server/en/latest/drivers-clients/index.html.

[24] BigchainDB Python Driver. https://docs.bigchaindb.com/projects/py-driver/en/latest/index.html.

[25] BigchainDB Production Nodes (Documentation). https://docs.bigchaindb.com/projects/server/en/latest/production-nodes/index.html.