Managing Data Storage with Blockchain and BigchainDB

By Chris Ward (https://www.sitepoint.com/author/cward/) April 06, 2016

There's no denying that whilst Bitcoin's future may be hazy right now, the underlying technology it relies upon — the Blockchain — has revolutionized many industries and projects, with more to come.

Ascribe (https://www.ascribe.io/) is a fascinating startup that uses Bitcoin's Blockchain to record a limited quantity of unique references to digital artworks. Thus making them traceable, accountable and (hopefully) more valuable, due to this finite amount of 'copies'.

Ascribe hit technological problems with this approach, and those problems were primarily due to Bitcoin's Blockchain itself. Writing everything to it is slow, costly (currently 80c each time) and has a maximum number of daily entries and total capacity for writes. It's also counter to typical scalable database technologies, adding nodes doesn't improve performance and has no real query language. This makes scaling a business that relies upon the Bitcoin Blockchain a challenge.

But the Blockchain concept is a strong one and the past years have seen an increasing rise in usage and legitimacy, with even major banks announcing development of technologies inspired by the concept.

Ascribe decided to combine the best of both worlds, taking a proven NoSQL database (RethinkDB (https://www.rethinkdb.com/)) and adding a Blockchain layer on top to add control, asset tracking and an additional level of security.

This combination of technologies is especially interesting to NoSQL database users, as traditionally, few of them support 'transactions (https://en.wikipedia.org/wiki/Database_transaction)' that help guarantee a database change has taken place. By writing to an underlying NoSQL database via a Blockchain layer, BigchainDB adds transactional support.

Thanks to the Blockchain layer, BigChainDB also claims to be fully decentralized. Whilst many distributed NoSQL databases claim this, there is often a pseudo master/slave setup.

Installing BigChainDB and Dependencies

There are couple of ways to install BigChainDB. First I tried the Docker images, but ran into some connection issues, finding the Python packages most reliable.

1. Install RethinkDB (http://rethinkdb.com/docs/install/), for other Mac users, there is also a Homebrew package available.
2. Install Python 3.4+ (https://www.python.org/downloads/).
3. Install BigChainDB with Pip – `sudo pip install bigchaindb`
4. Start RethinkDB with `rethinkdb`
5. Start BigChainDB with `bigchaindb start` which will also configure things for you.
6. Open the BigChainDB (actually the RethinkDB UI) admin UI at `http://SERVER_IP:58080/`

Simple Example – Message Allocation and Tracking

One of BigchainDB's prime use cases (and why Ascribe created it), is for tracking assets, so let's make a simple example in Python. First run the following commands in your terminal.

```
pip install bigchaindb
bigchaindb configure
bigchaindb show-config
```

Create a new file, *app.py* and add the following:

```
from bigchaindb import Bigchain
b = Bigchain()
print(b)
```

This imports the bigchaindb library, creates a new object and connects to it with the settings file just created.

Then run the Python application:

```
python app.py
```

You should see something like `<bigchaindb.core.Bigchain object at 0x1085b0dd8>`, this tells us that everything is well.

Add the following:

```
from bigchaindb import Bigchain
import time

b = Bigchain()

spuser_priv, spuser_pub = b.generate_keys()
print("User Created")

digital_asset_payload = {'msg': 'This is my special message just for you'}

tx = b.create_transaction(b.me, spuser_pub, None, 'CREATE', payload=digital_asset_payload)
print("Transaction Written")

tx_signed = b.sign_transaction(tx, b.me_private)
b.write_transaction(tx_signed)
print ("Transaction Written to BC, now waiting")

time.sleep(10)

tx_retrieved = b.get_transaction(tx_signed['id'])
print(tx_retrieved)
```

This creates a user and associated keys for access to the database — remember that extra level of security. Then a payload for writing to the database is created, assigning the required keys, and written.

It will take a few seconds for the new transaction to pass from the Blockchain layer to the database. The code waits for ten seconds and then retrieves and prints the record. You should see something like:

```
{
  "signature": '304502205",
  "id": "0f442bcf4a42",
  "transaction": {
      "timestamp": "1457104938.430521",
      "data": {
        "hash": "b32779e57",
        "payload": {
          "msg": "This is my special message just for you"
        }
      },
      "operation": "CREATE",
      "current_owner": "hFJKYk2",
      "new_owner": "26pdiQTTx",
      "input": None
    }
  }
}
```

You now have one special message that you would like one person to have access to:

```
...
print("Now to transfer")

spuser2_priv, spuser2_pub = b.generate_keys()
print("Second User Created")

tx_transfer = b.create_transaction(spuser_pub, spuser2_pub, tx_retrieved['id'], 'TRANSFER')
print("Transfer Created")

tx_transfer_signed = b.sign_transaction(tx_transfer, spuser_priv)
b.write_transaction(tx_transfer_signed)
print ("Transaction Written to BC, now waiting")

time.sleep(15)

tx_transfer_retrieved = b.get_transaction(tx_transfer_signed['id'])
print("Transferred")
print(tx_transfer_retrieved)
```

This creates a second user and then takes the transaction ID of the special message and transfers it to the second user. The Blockchain layer of BigChainDB will prevent users and your code from executing the same action twice. If you tried running the code above again, a `double spend exception` will be thrown.

This examples shows a small set of the methods that BigChainDB adds to RethinkDB, find the full list here (https://bigchaindb.readthedocs.org/en/develop/developer-interface.html).

HTTP Endpoint

Currently, the only client library available for BigChainDB is Python, more may follow, but in the meantime, a limited HTTP endpoint is available for querying existing transactions:

http://localhost:5000/api/v1/transactions/tx_id

Or write a new transaction with:

http://localhost:5000/api/v1/transactions

(/)

Adding the following payload, where `operation` can be changed to suit the different types of transaction that can be written :

```
{
  "id": , ""
  "signature": "",
  "transaction": {
    "current_owner": "",
    "data": {
      "hash": "",
      "payload": null
    },
  "input": null,
  "new_owner": "",
  "operation": "",
    "timestamp": ""
  }
}
```


Part of a Decentralized Future

More from this author

- Developing Add-ons for Enterprise Apps like JIRA (https://www.sitepoint.com/developing-add-ons-for-enterprise-apps-like-jira/?utm_source=sitepoint&utm_medium=relatedinline&utm_term=&utm_campaign=relatedauthor)
- Exploring the Evive: A Book-Sized IoT Device (https://www.sitepoint.com/building-an-all-in-one-book-sized-iot-device-with-evive/?utm_source=sitepoint&utm_medium=relatedinline&utm_term=&utm_campaign=relatedauthor)
- Building Your First Blockchain App with Eris (https://www.sitepoint.com/getting-into-blockchain-with-eris/?utm_source=sitepoint&utm_medium=relatedinline&utm_term=&utm_campaign=relatedauthor)

Ignoring its Blockchain heritage for a moment, BigChainDB offers to supply a lot of functionality missing from current NoSQL and distributed databases. That fact alone may be a reason to try it and may provide a valid business/use case.

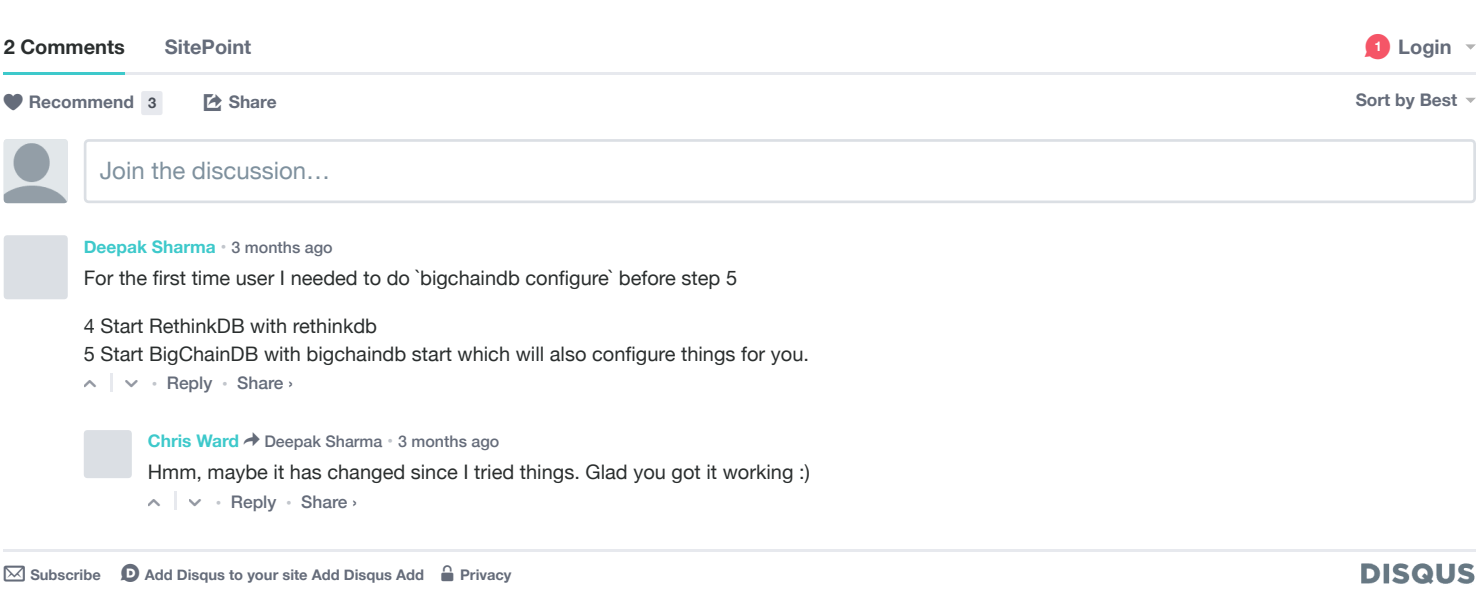
For the Blockchain aficionados amongst you, it also completes the puzzle for a complete decentralized application stack. In theory there is now Ethereum (https://www.ethereum.org/) for applications, IPFS (https://ipfs.io/) as a file system and now BigChainDB for data storage. The pieces are in place for a very different way of developing, deploying and maintaining applications, leading to a fascinating future that I would love to hear your opinions on in the comments below.

More: Ascribe (https://www.sitepoint.com/tag/ascribe/), BigChainDB (https://www.sitepoint.com/tag/bigchaindb/), blockchain (https://www.sitepoint.com/tag/blockchain/), Emerging Tech (https://www.sitepoint.com/tag/emerging-tech/)

Meet the author

Chris Ward (https://www.sitepoint.com/author/cward/) (https://twitter.com/chrischinch) (https://plus.google.com/u/0/101661055974776035046/posts) (https://www.facebook.com/chrischinchilla) (http://www.linkedin.com/in/chrischinchilla) (https://github.com/ChrisChinchilla) (https://medium.com/@ChrisChinchilla)

Developer Relations, Technical Writing and Editing, (Board) Game Design, Education, Explanation and always more to come. English/Australian living in Berlin, Herzlich Willkommen!



LATEST COURSES ›
(/premium/courses/)

PREMIUM COURSE

5h 37m

User Interface Design with Sketch 4 (https://www.sitepoint.com/premium/courses/user-interface-design-with-sketch-4-2927)

PREMIUM COURSE

1h 1m

Diving into ES2015 (https://www.sitepoint.com/premium/courses/diving-into-es2015-2924)

PREMIUM COURSE

2h 15m

Wrapping Your Head Around Python (https://www.sitepoint.com/premium/courses/wrapping-your-head-around-python-2916)

LATEST BOOKS ›
(/premium/books/)

PREMIUM BOOK

Rails: Novice to Ninja (https://www.sitepoint.com/premium/books/rails-novice-to-ninja)

PREMIUM BOOK

Designing UX: Forms (https://www.sitepoint.com/premium/books/designing-ux-forms)

PREMIUM BOOK

Bootstrap: A SitePoint Anthology #1 (https://www.sitepoint.com/premium/books/bootstrap-a-sitepoint-anthology-1)

(/)

Get the latest in Front-end, once a week, for free.

Enter your email

Subscribe

Facebook LinkedIn Twitter

About

Our Story (/about-us/)
Advertise (/advertise/)
Press Room (/press/)
Reference (http://reference.sitepoint.com/css/)
Terms of Use (/legals/)
Privacy Policy (/legals/#privacy)
FAQ (https://sitepoint.zendesk.com/hc/en-us)
Contact Us (mailto:feedback@sitepoint.com)
Contribute (/write-for-us/)

Visit

SitePoint Home (/)
Forums (https://www.sitepoint.com/community/)
Newsletters (/newsletter/)
Premium (/premium/)
References (/sass-reference/)
Shop (https://shop.sitepoint.com)
Versioning (https://www.sitepoint.com/versioning/)

Connect

(https://www.sitepoint.com/feed/) (/newsletter/) (https://www.facebook.com/sitepoint) (http://twitter.com/sitepointdotcom) (https://plus.google.com/+sitepoint)

(/)