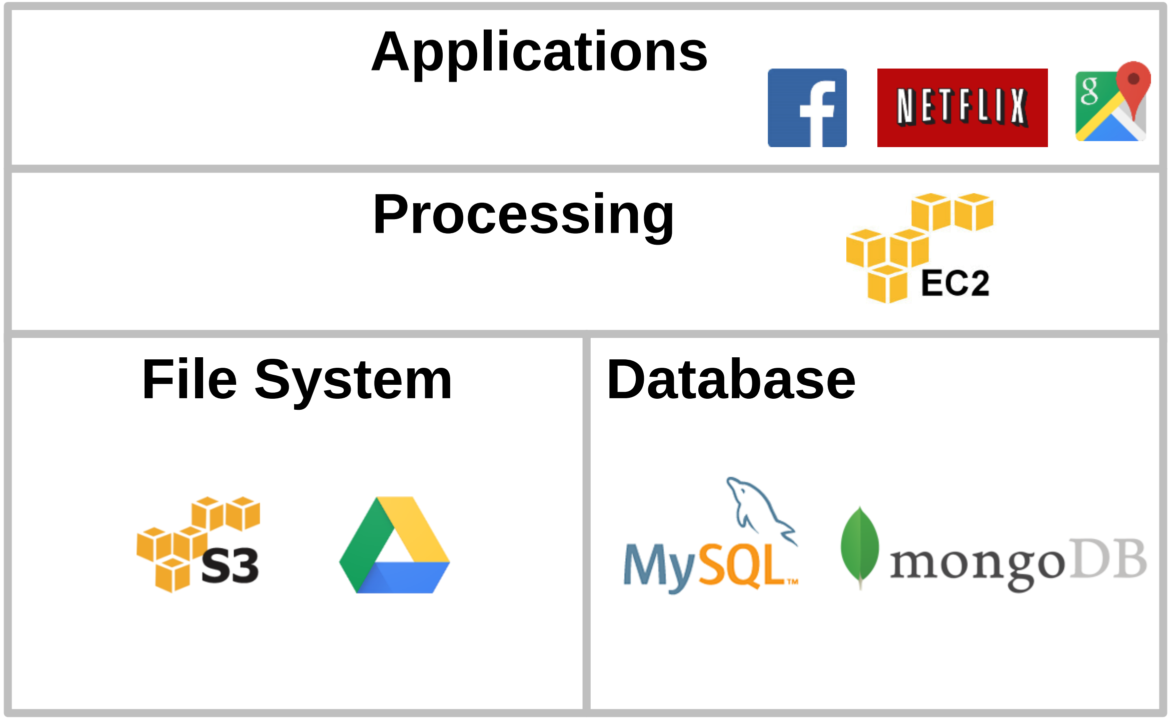
BigchainDB:
A Scalable Blockchain Database

Trent McConaghy

BIGCHAINDB

The modern cloud application stack



Applications

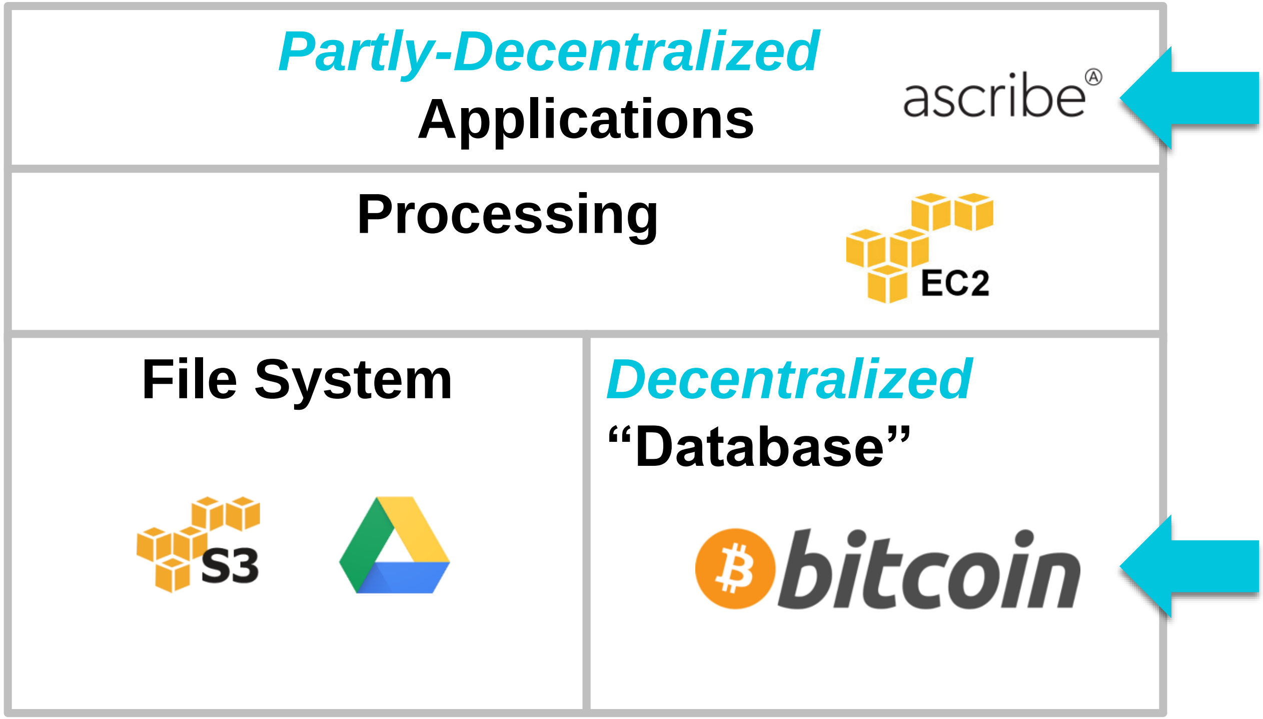
Processing

File System

Database

Along came Bitcoin…

The modern cloud application stack – with Bitcoin



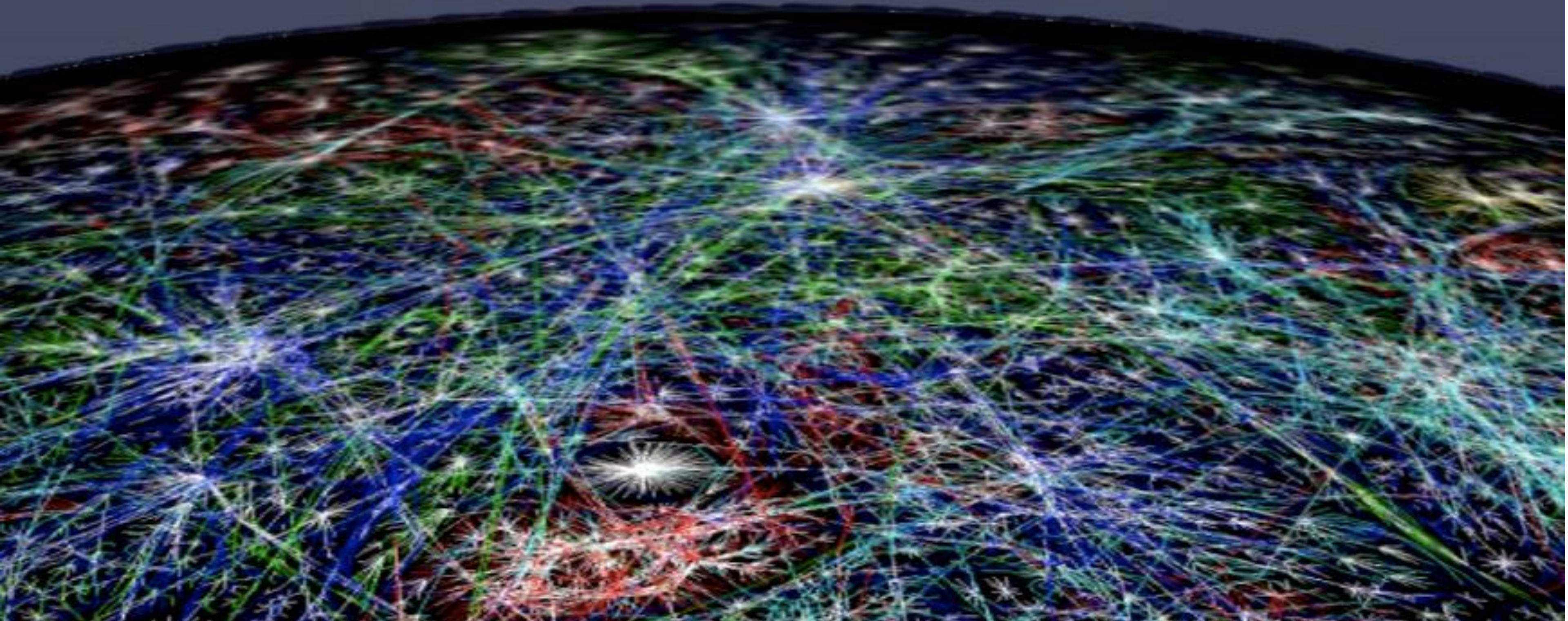
Partly-Decentralized
Applications

Processing

File System

Decentralized
“Database”

Planetary Scale?



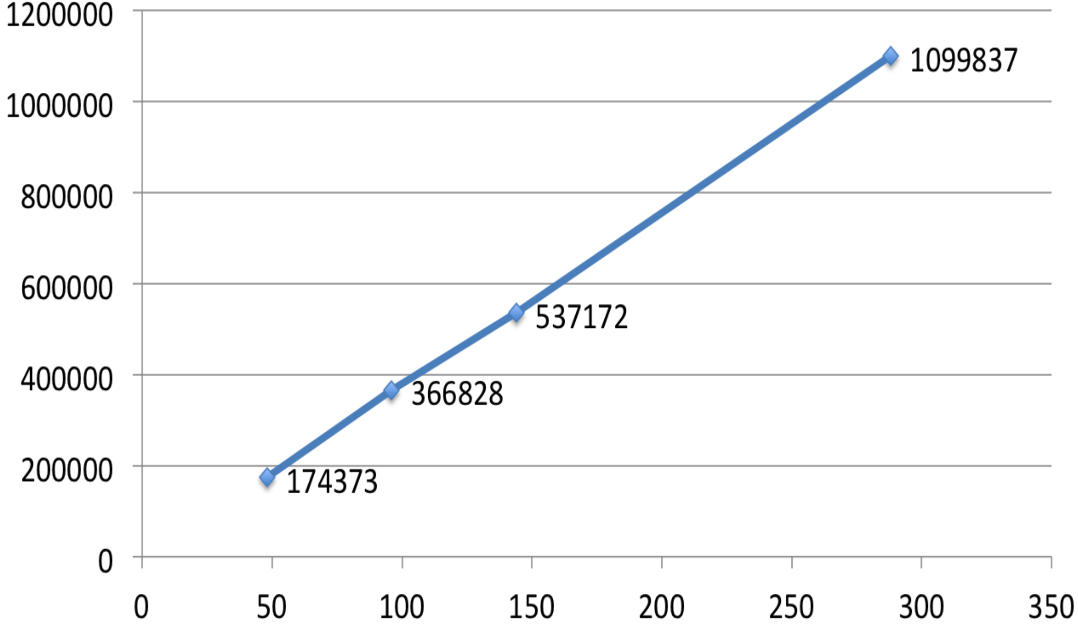
Netflix uses 37% of Internet bandwidth

Netflix uses 37% of Internet bandwidth

Using a modern distributed “big data” database

Netflix uses 37% of Internet bandwidth
Using a modern distributed “big data” database

Writes / s vs. # nodes



Two ways to scale up

Big data-fy blockchains

- Builds on man-decades of work
- Significant scalability hurdles

<or>

Blockchain-ify big data

- Builds on man-centuries (millennia?) of work
- Scalability challenges already resolved
- How to blockchain-ify? …

“Blockchain-ify”

Decentralization: no single entity owns or controls

Immutability: tamper-resistant

Assets: Can issue & transfer assets

Blockchain (noun): hashed-together chain of blocks (1991!)

Blockchain (noun): storage that is decentralized + immutable + assets

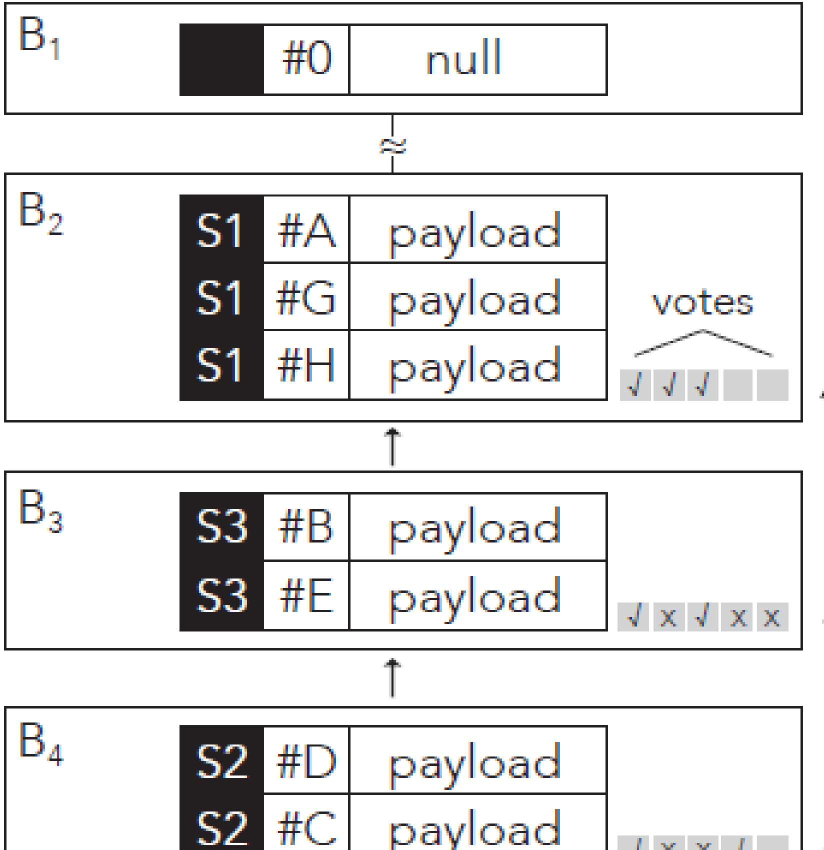
Blockchain (*adj*): decentralized + immutable + assets

How to Blockchain-ify Big Data

- **Decentralized:** each DB node is a federation node
- **Immutable:** hash on prev. blocks, append-only
- **Assets:** Interledger protocol

bigchaindb.com/whitepaper

github.com/bigchaindb (AGPL)



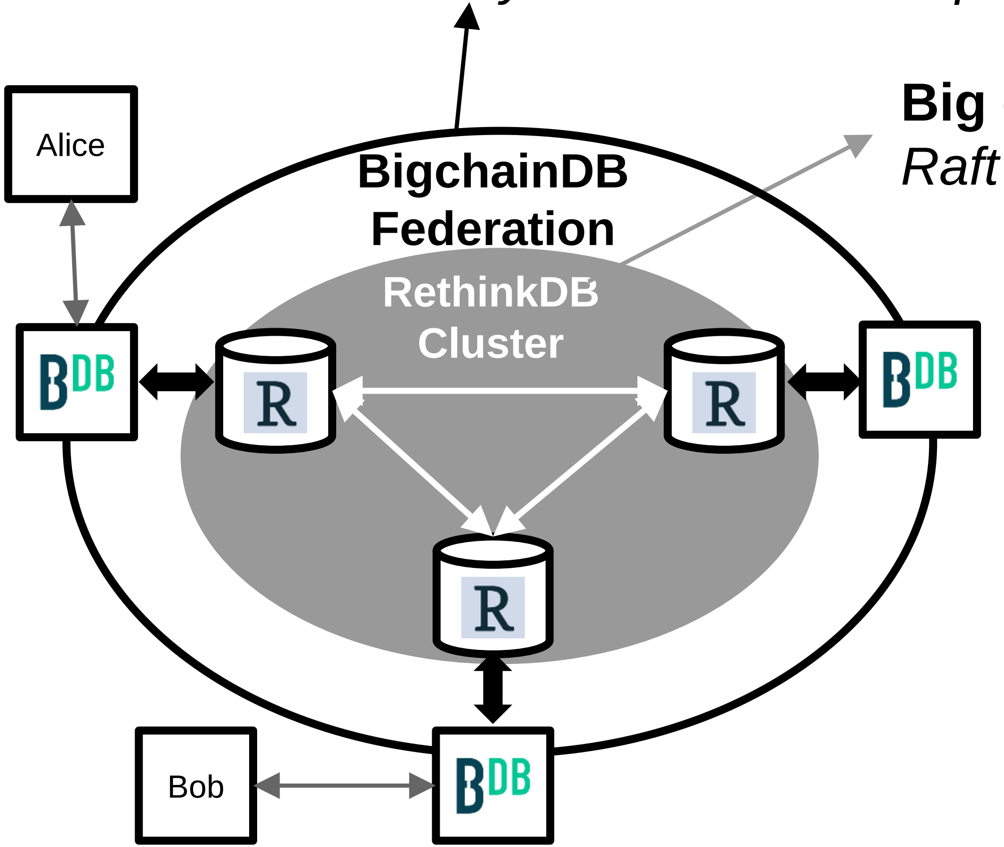
Architecture

Blockchain consensus

Byzantine actors -> quorum

Big data consensus

Raft -> strong consistency



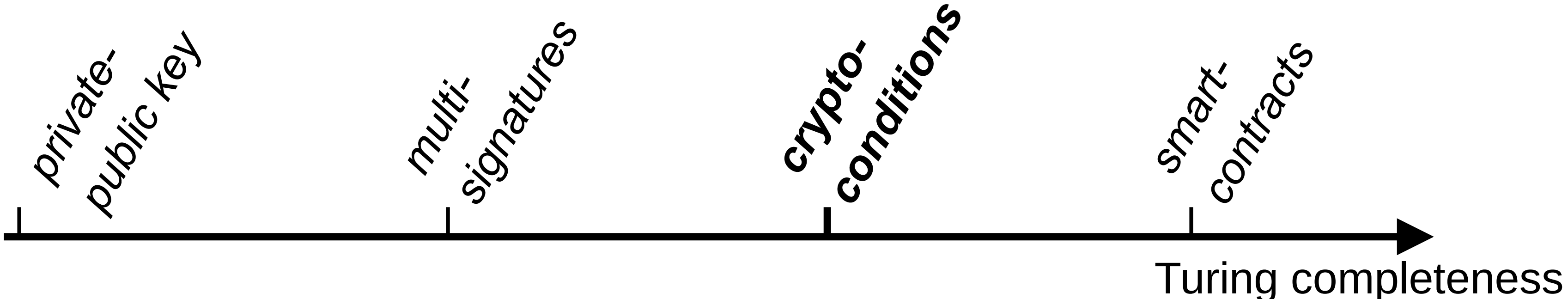
BigchainDB Interface

Database part : data

Via ReQL (JSON meets SQL)

+ Blockchain part : assets, transaction-style

Via Interledger Protocol (Crypto-conditions)



BigchainDB characteristics

Throughput
>1,000,000 writes/s
~100,000 transactions/s

Latency
<100 ms

Capacity
Petabytes with each node adding 48TB

Scalability
Performance increases as nodes are added

Query
Database is fully queryable

Decentralization
Federated
non-anonymous participation

Public version of BigchainDB

iPDB

INTERPLANETARY DATABASE

IPDB

INTERPLANETARY DATABASE

- A shared global database. For everyone, everywhere
- And, a nonprofit foundation, with decentralized governance
- Powered by BigchainDB, to start
- Free except for high-volume users
- Caretakers co-operate network & co-govern foundation

IPDB Caretakers (so far)

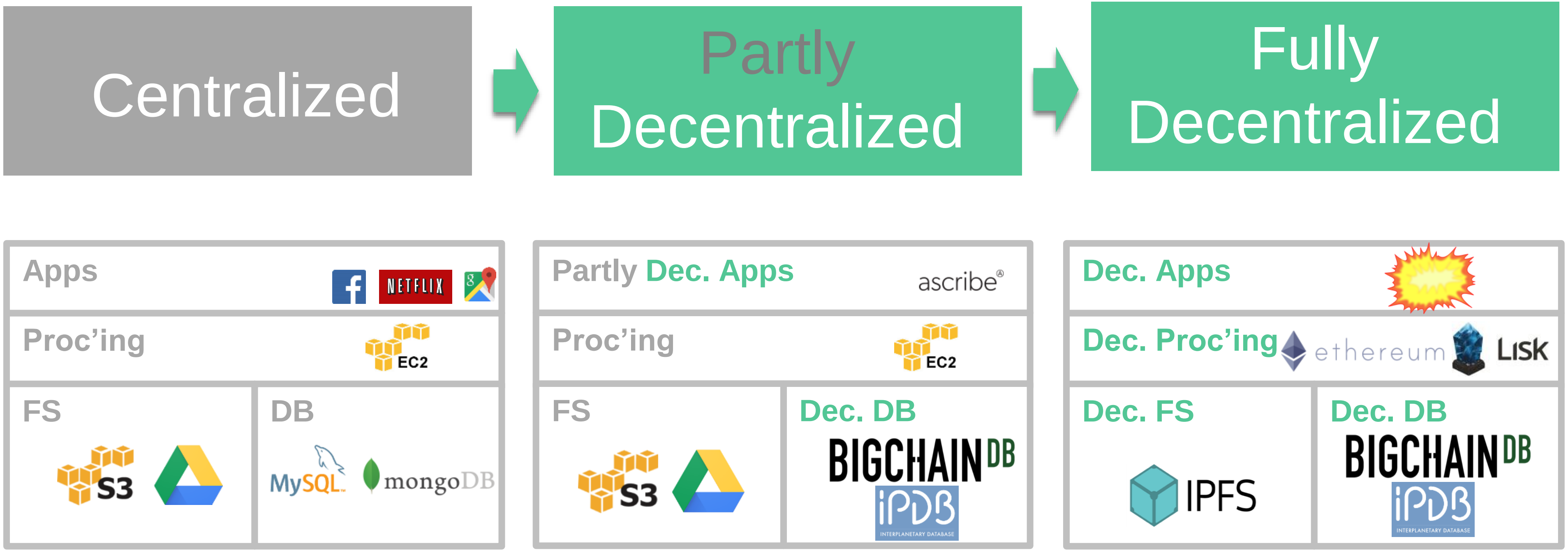
Not-for-profit

Blockstack
B.SAFE
COALA
Dyne.org
Internet Archive
OpenMedia
UnMonastery

For-profit

BigchainDB
Consensys
Eris Industries
Protocol Labs (IPFS)
SmartContract.com
Synereo
Tendermint

Decentralization of the Cloud



Centralized → Partly Decentralized → Fully Decentralized

Centralized

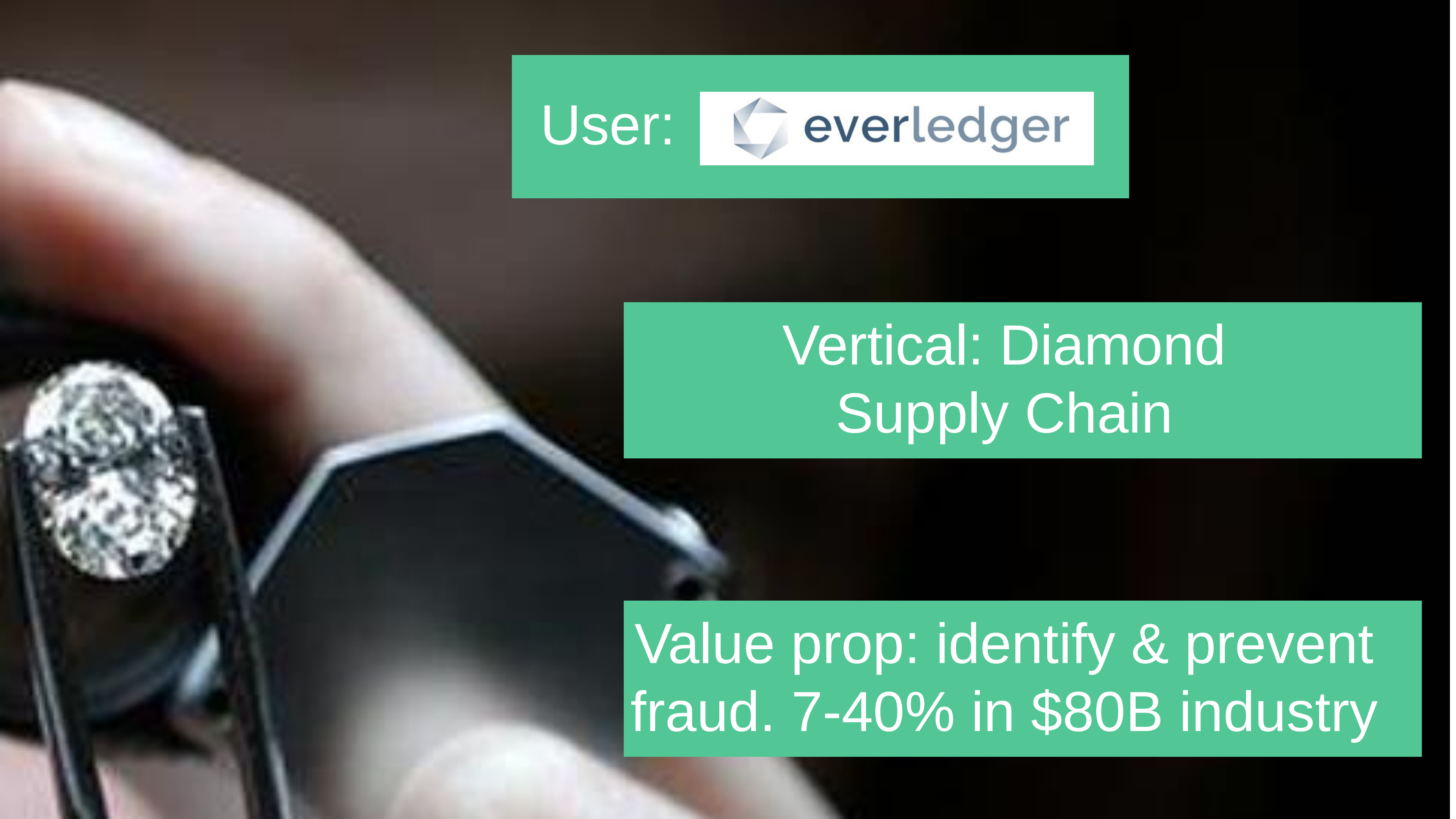
- Apps
- Proc'ing
- FS
- DB

Partly Decentralized

- Partly Dec. Apps
- Proc'ing
- FS
- Dec. DB

Fully Decentralized

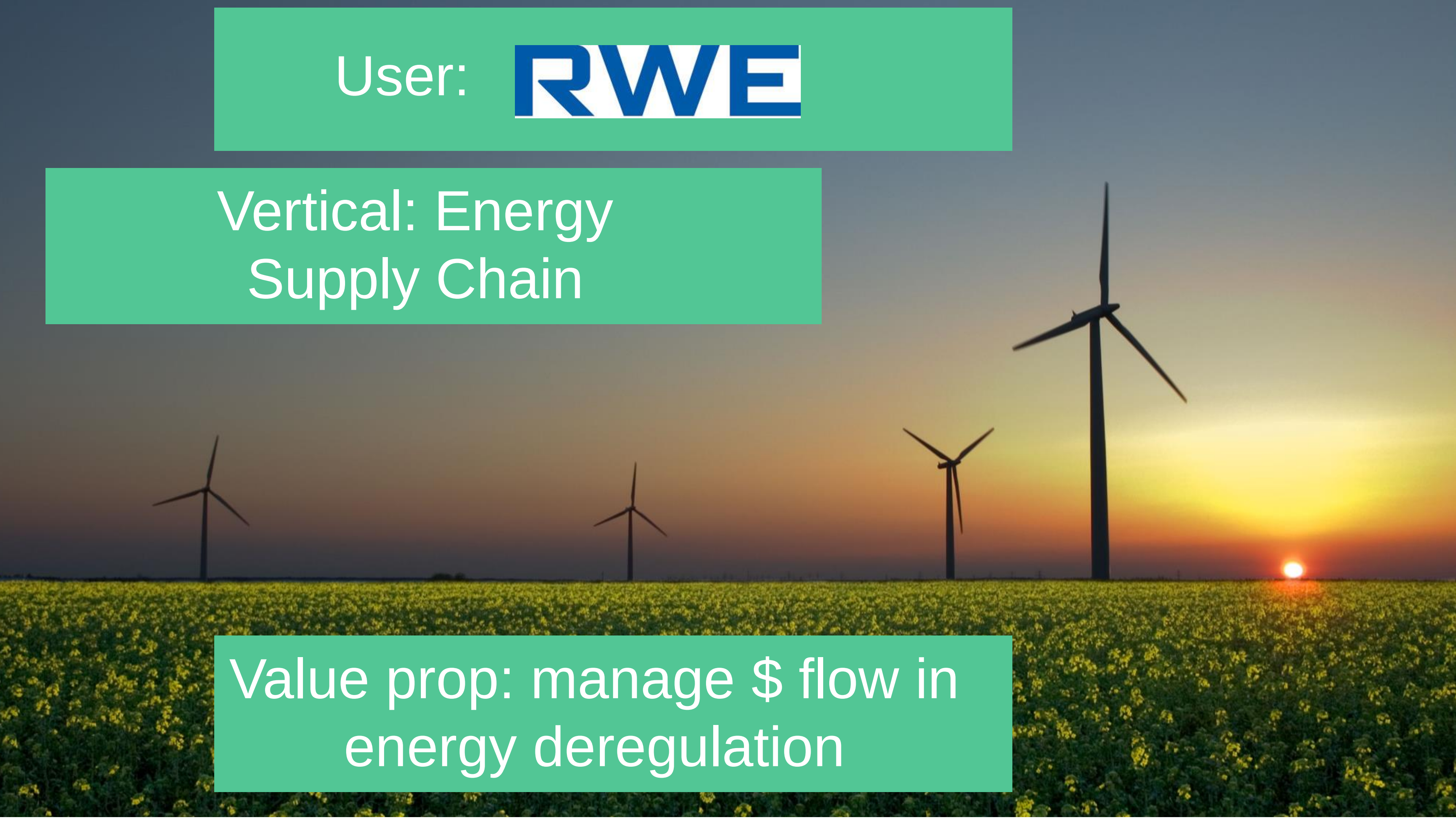
- Dec. Apps
- Dec. Proc'ing
- Dec. FS
- Dec. DB



User: everledger

Vertical: Diamond Supply Chain

Value prop: identify & prevent fraud. 7-40% in $80B industry



User: RWE

Vertical: Energy Supply Chain

Value prop: manage $ flow in energy deregulation

User: BenBen

Vertical:
Land registry

Value prop: less-corruptible land titles; land ownership as stepping stone

Vertical: Medical Journals / Supply Chain

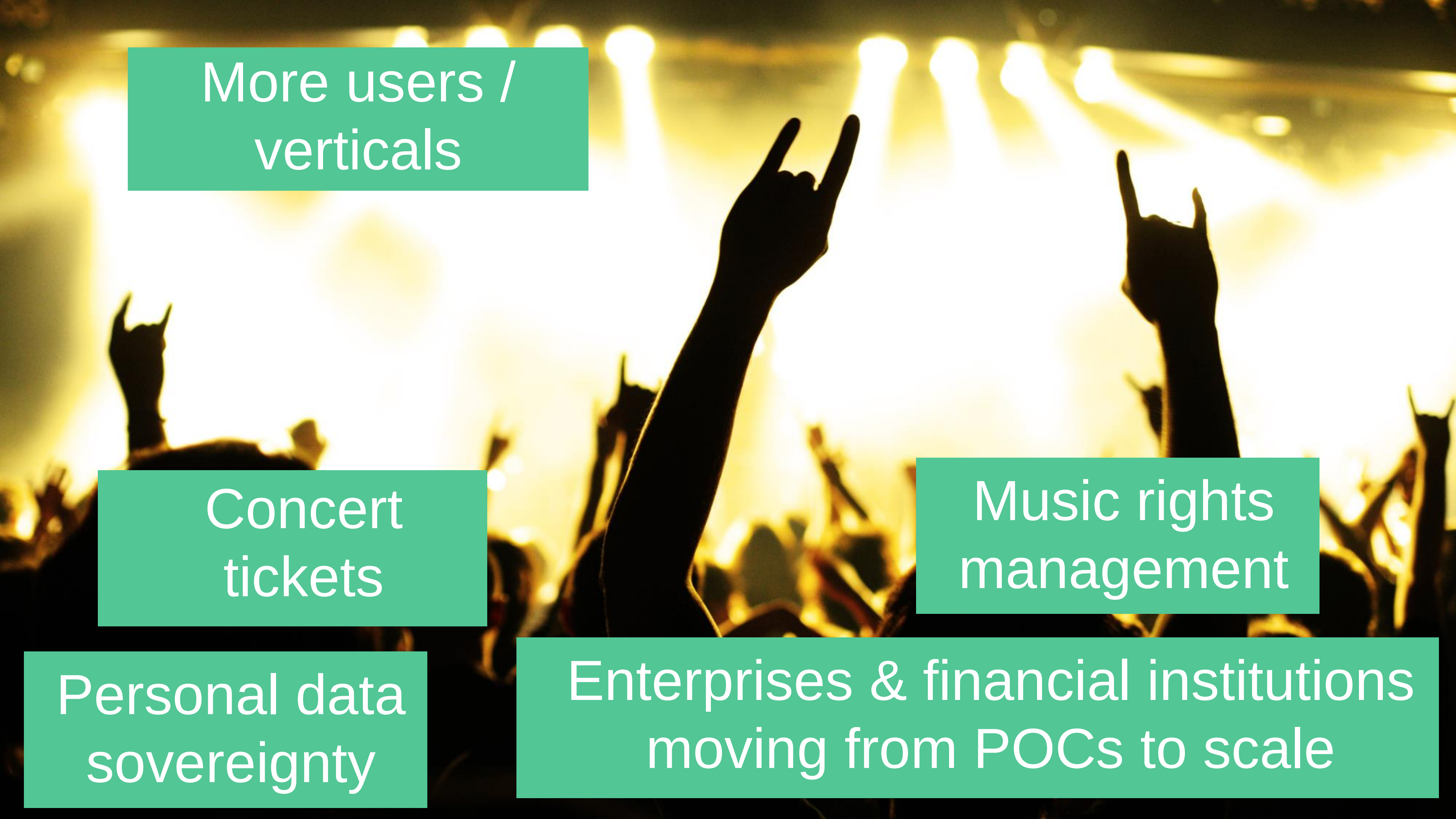
User: Tangent90

Value prop: government-mandated transparent $ flow

User: ascribe.io
(incl. 5000 artists, 25 orgs)

Verticals: Art Supply Chain, Intellectual Property

Value Props: secure provenance, IP mgmt.



More users / verticals

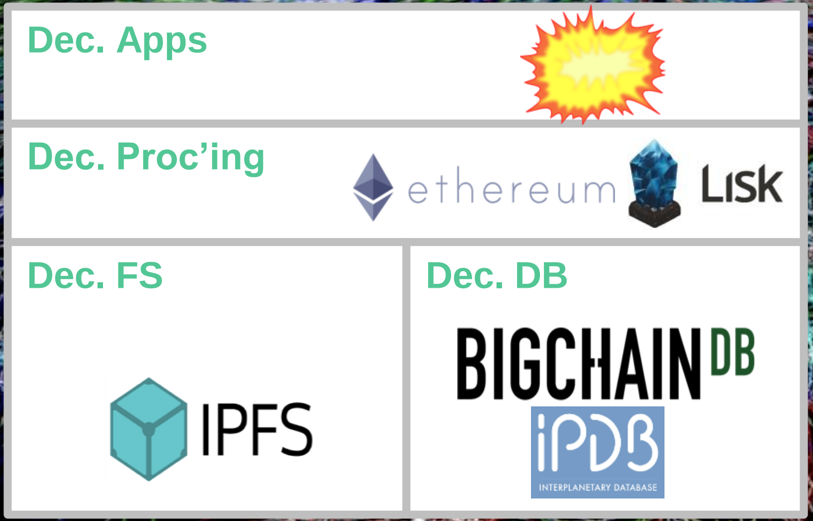
Concert tickets

Music rights management

Personal data sovereignty

Enterprises & financial institutions moving from POCs to scale

BigchainDB:
A Scalable Blockchain Database
For Enterprises, and the Planet



Dec. Apps

Dec. Proc'ing

Dec. FS

Dec. DB

bigchaindb.com

github.com/bigchaindb

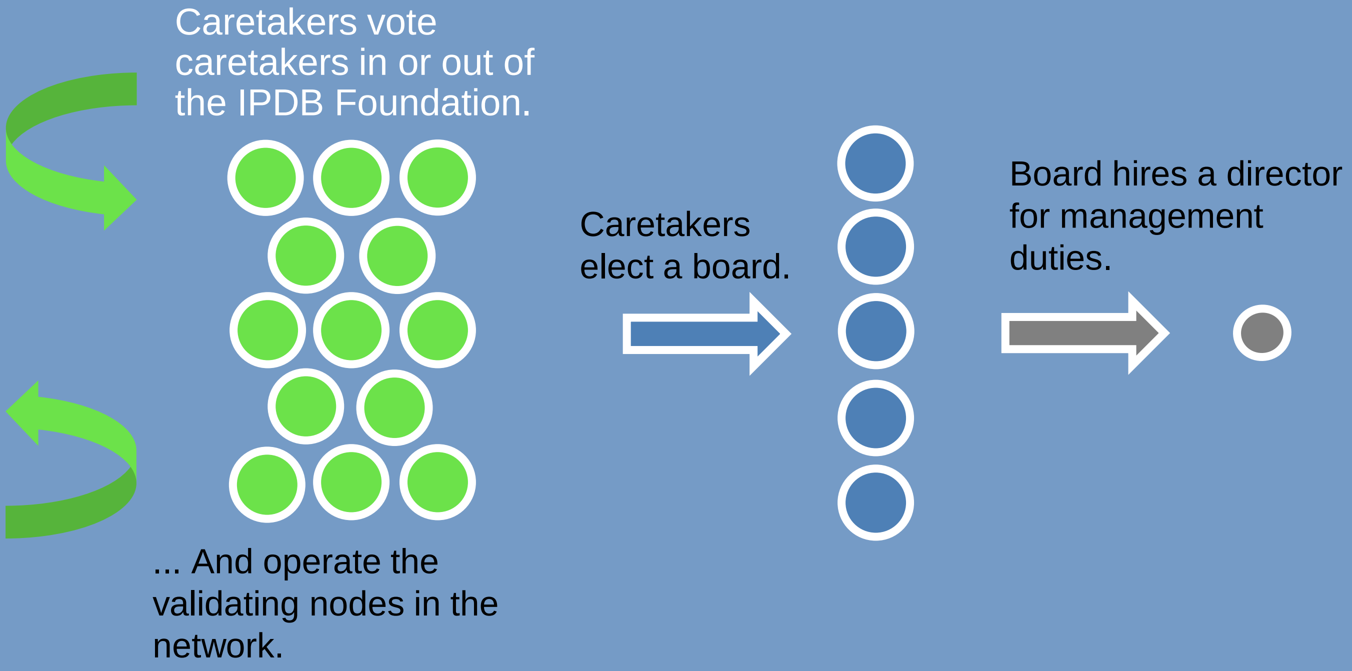
../bigchaindb-examples

trent@bigchaindb.com

NETWORK ILLUSTRATION
BY OPTE PROJECT

APPENDIX:
IPDB

IPDB Governance: caretakers at the heart



Caretakers vote caretakers in or out of the IPDB Foundation.

... And operate the validating nodes in the network.

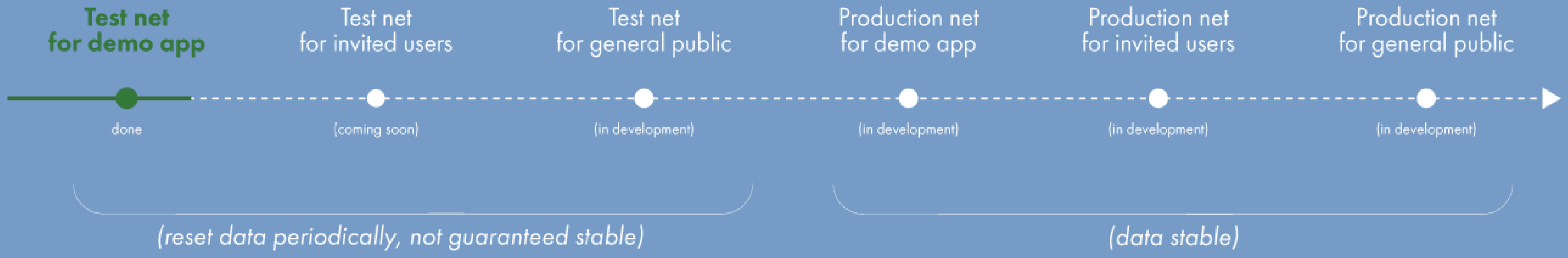
Caretakers elect a board.

Board hires a director for management duties.

Yes, this could be a DAO.

But not yet. Walk before we run.

IPDB Roadmap



Test net for demo app

done

Test net for invited users

(coming soon)

Test net for general public

(in development)

Production net for demo app

(in development)

Production net for invited users

(in development)

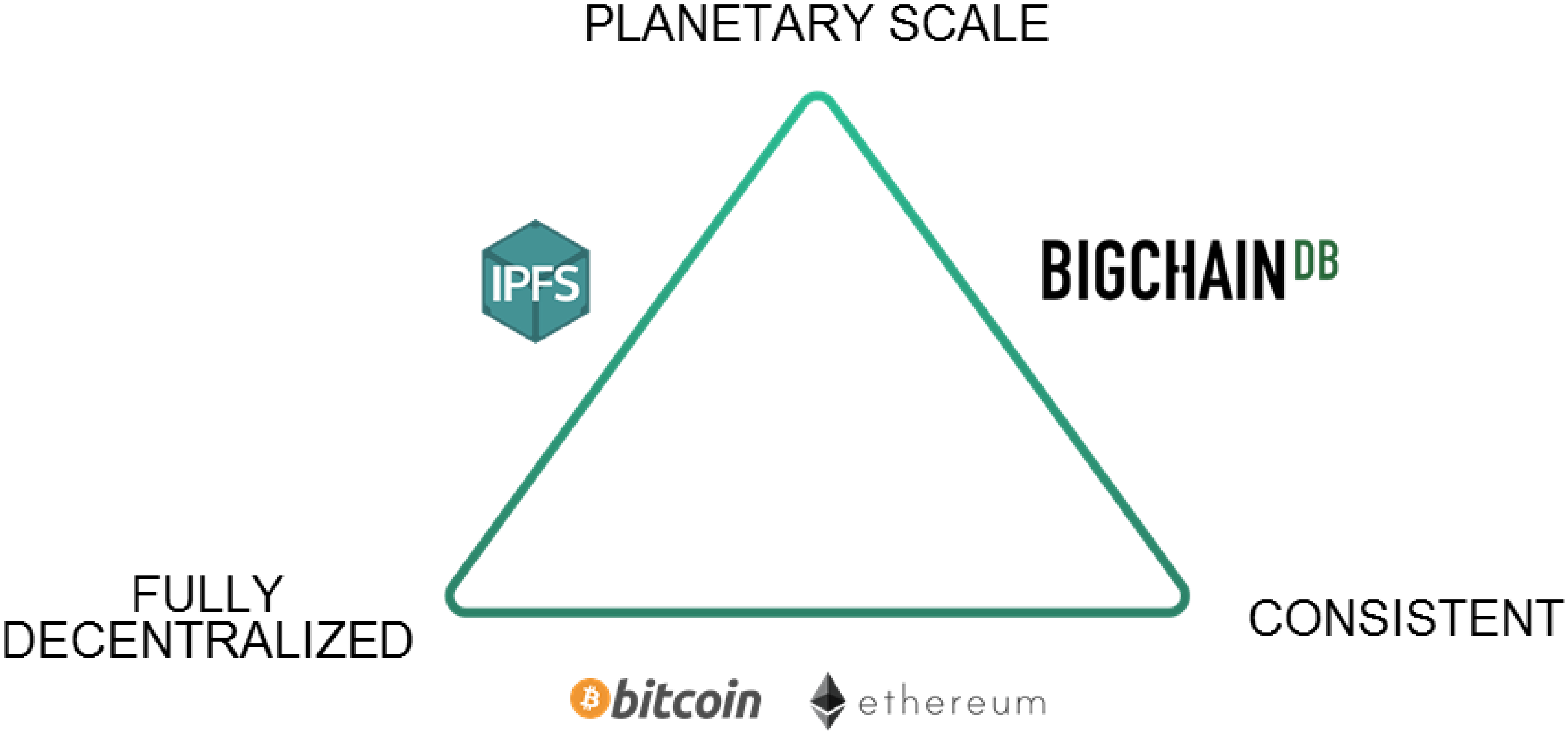
Production net for general public

(in development)

(reset data periodically, not guaranteed stable)

(data stable)

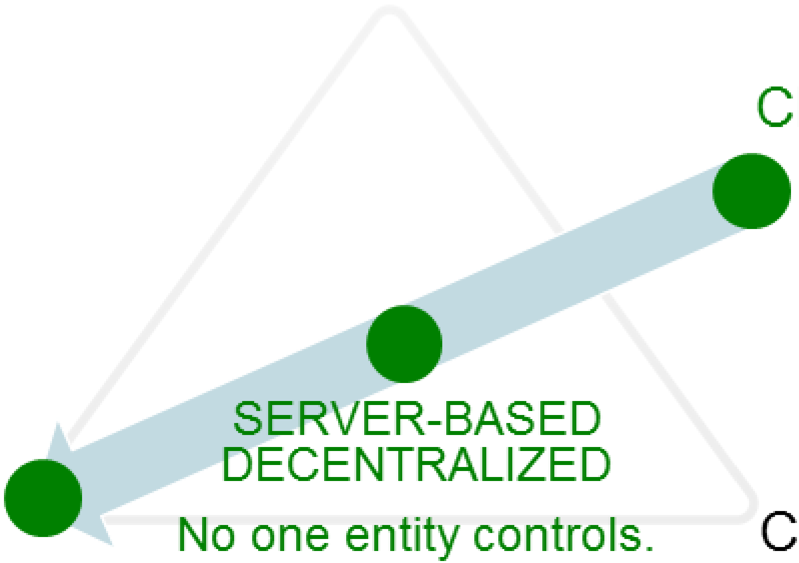
APPENDIX: TRADEOFFS



PLANETARY SCALE

FULLY DECENTRALIZED

CONSISTENT



Planetary Scale

CENTRALIZED

Single entity controls

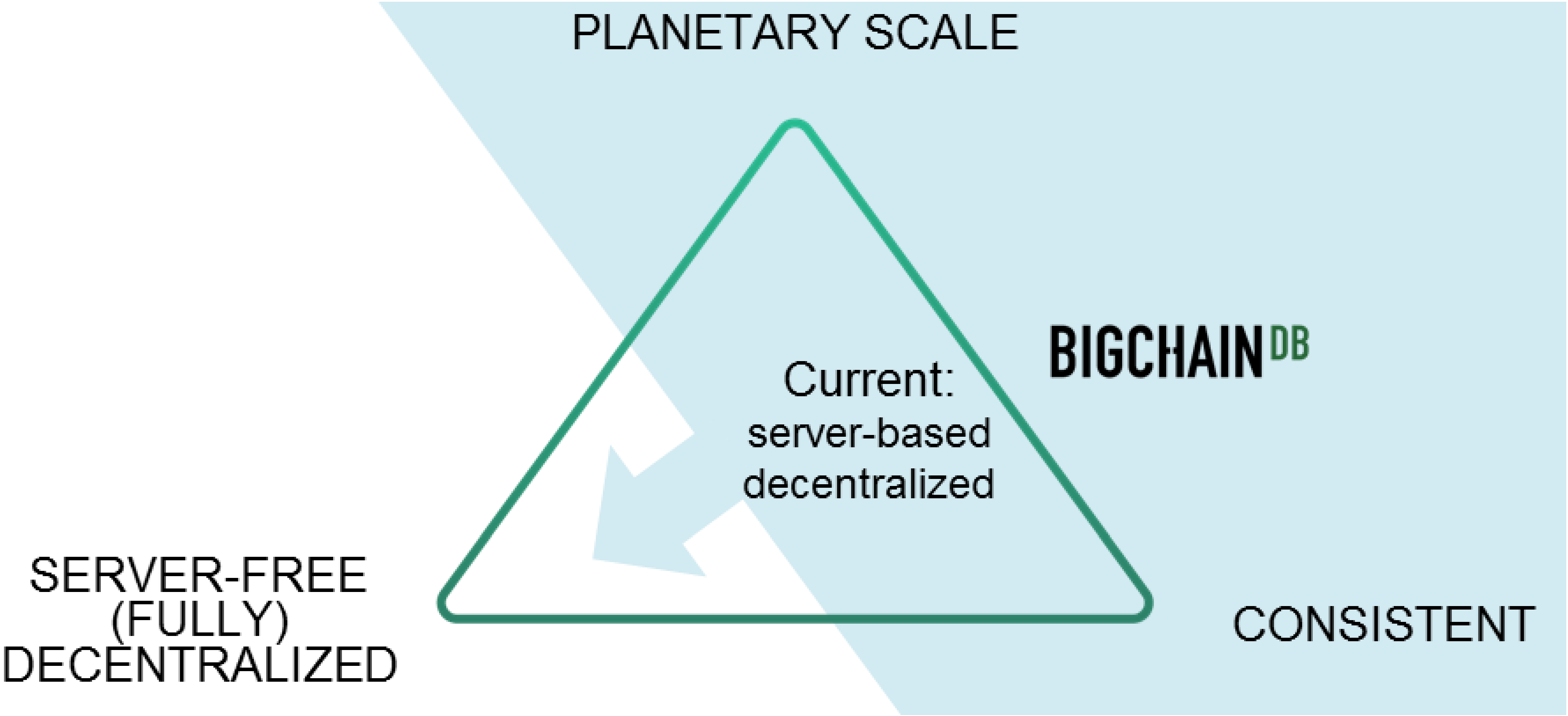
SERVER-BASED DECENTRALIZED

No one entity controls.
Anyone can write.
Anyone can read.
Anyone voted in by federation can be validator

SERVER-FREE (FULLY) DECENTRALIZED

No one entity controls.
Anyone can write,
Anyone can read.
Anyone* can be validator.
(*need CPU power)

CONSISTENT



PLANETARY SCALE

BIGCHAINDB

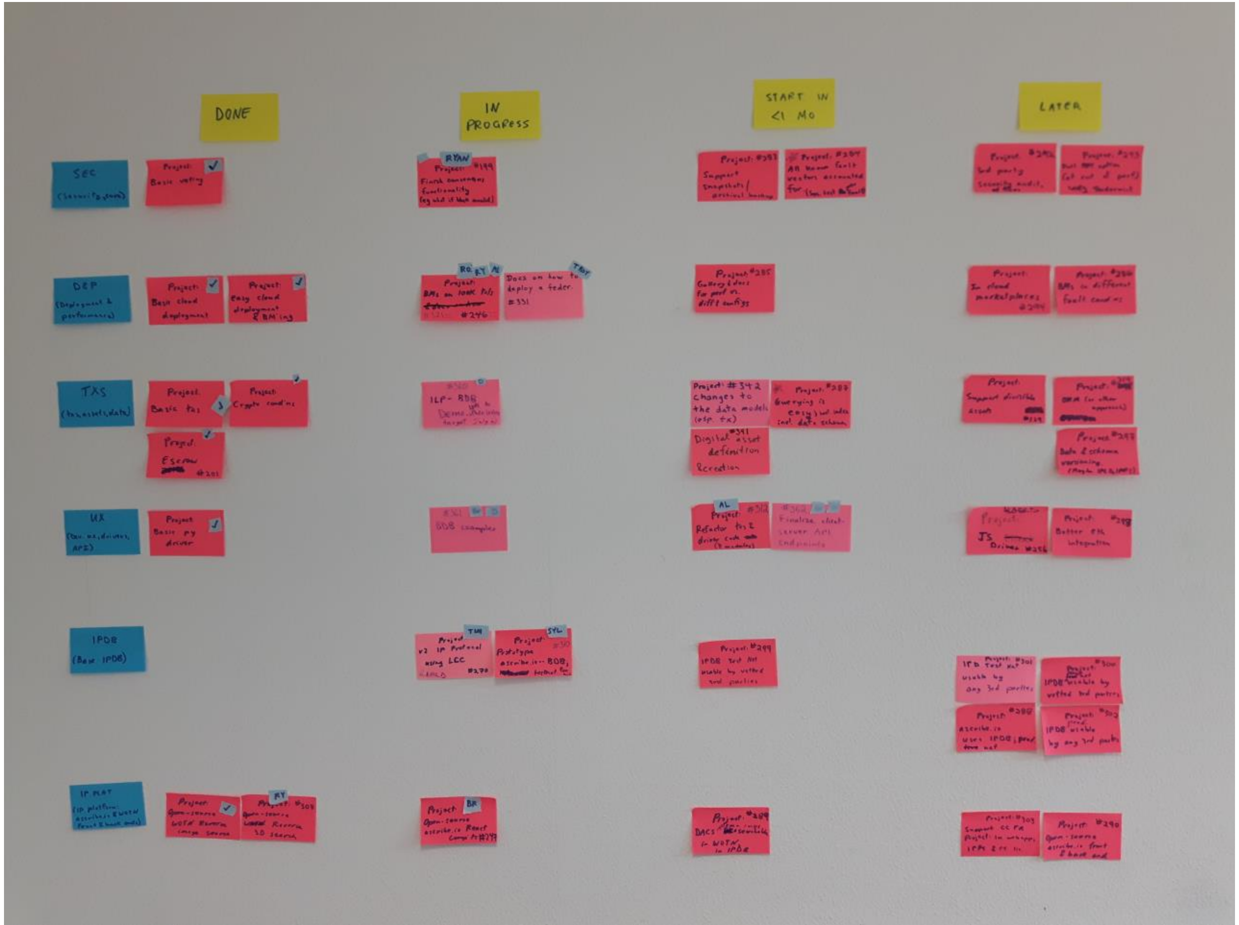
Current:
server-based
decentralized

SERVER-FREE
(FULLY)
DECENTRALIZED

CONSISTENT

APPENDIX: ROADMAP

https://github.com/bigchaindb/org/blob/master/ROADMAP.md


DONE
IN PROGRESS
START IN <1 MO
LATER
SEC (security, ...)
Project: Basic voting ✓
RYAN Project: #199 Finish consensus functionality
Project: #292 3rd party security audit
Project: #293
DEP (Deployment & performance)
Project: Basic cloud deployment ✓
Project: Easy cloud deployment & BM'ing ✓
Project: BMs on 100K ... #246
TROY Docs on how to deploy a feder. #331
Project #285
Project: In cloud marketplaces #294
Project: #296 BMs in different fault conditions
TXS (txs, assets, data)
Project: Basic txs ✓
Project: Crypto conditions ✓
Project: Escrow #201 ✓
ILP - BDB
Project: #342 changes to the data models (esp. tx)
#341 Digital asset definition & creation
Project: #287
Project: Support divisible assets
Project: #297 Data & schema versioning
UX (Dev. ux, drivers, API)
Project Basic py driver ✓
BDB examples
AL Project: #312 Refactor py & driver code
#362 Finalize client-server API endpoints
Project: JS Driver #256
Project: #298 Better eth integration
IPDB (Base IPDB)
TIM Project v2 IP Protocol using LCC #270
SYL Project: Prototype ascribe.io -- BDB
Project: #299 IPDB Test Net usable by vetted 3rd parties
Project: #301 IPDB Test Net usable by any 3rd parties
Project: #300 IPDB usable by vetted 3rd parties
Project: #288 ascribe.io uses IPDB
Project: #302 IPDB usable by any 3rd parties
IP PLAT (IP platform: ascribe, WOTN (front & back ends))
Project: Open-source WOTN Reverse image search ✓
RY Project: #303 Open-source Reverse 3D search
BR Project: Open-source ascribe.io React Components #297
Project: #289 DACs ... WOTN ... IPDB
Project: #303
Project: #290 Open-source ascribe.io front & back end

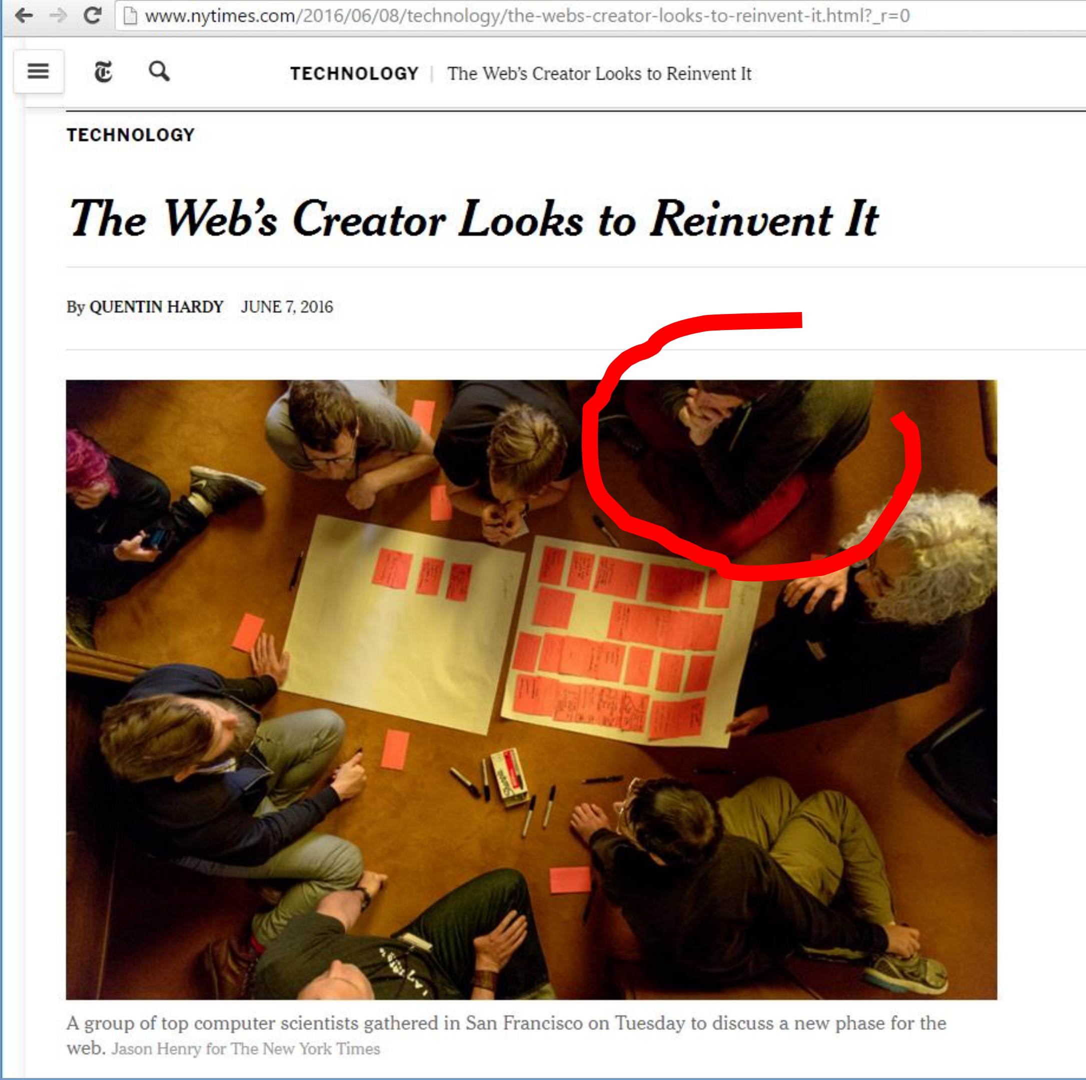
APPENDIX: INTERNET

The internet is getting upgraded, driven by the winds of blockchain.

Old + new guard are joining forces!

How to have lasting upgrade? New protocols.

W3C Blockchain, Coala IP, Copyright Hub / LCC, OMI, Interledger, IPLD, Web of Trust, Estonia e-identity


www.nytimes.com/2016/06/08/technology/the-webs-creator-looks-to-reinvent-it.html?_r=0
TECHNOLOGY | The Web's Creator Looks to Reinvent It

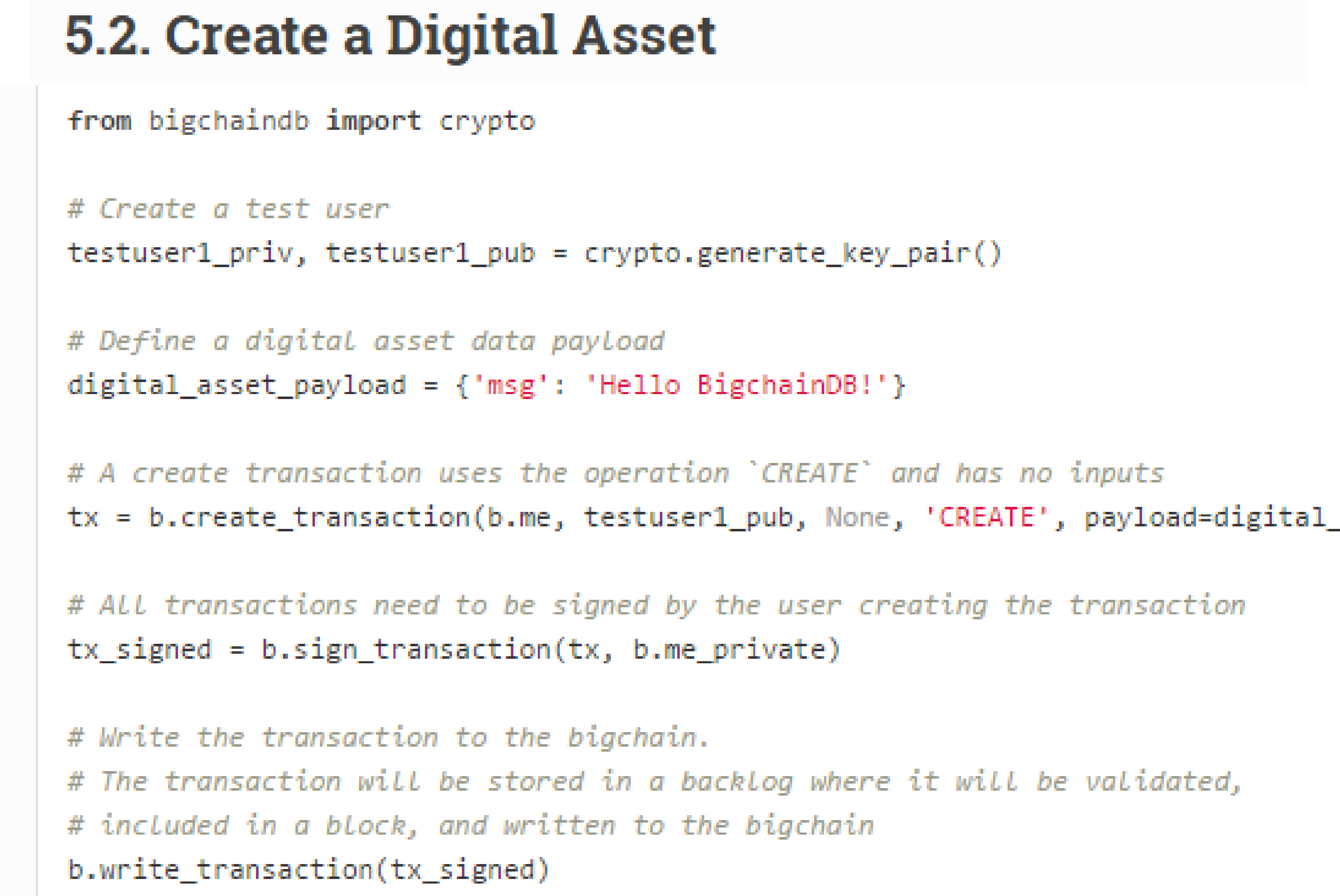
TECHNOLOGY

The Web's Creator Looks to Reinvent It

By QUENTIN HARDY JUNE 7, 2016

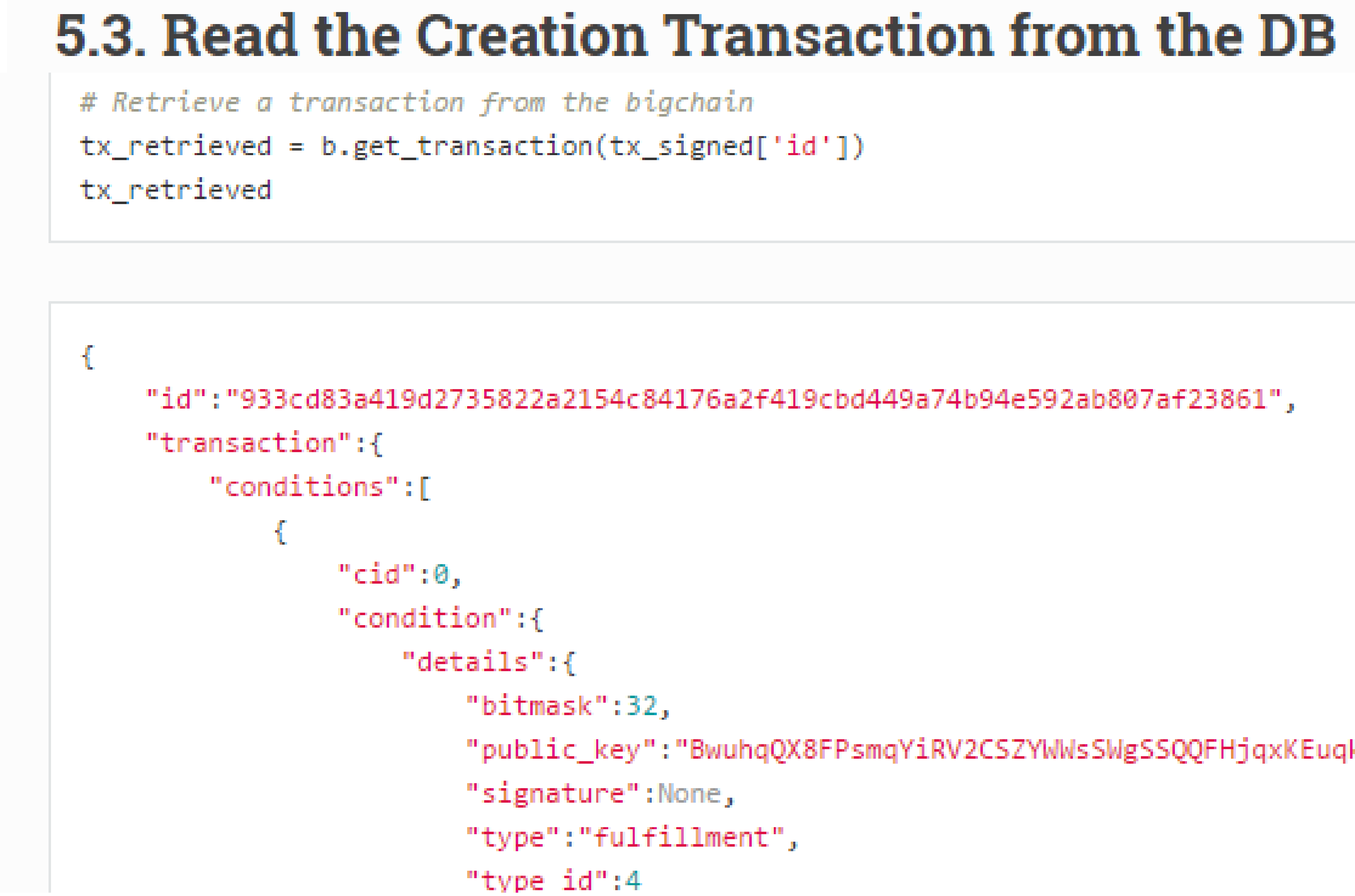
A group of top computer scientists gathered in San Francisco on Tuesday to discuss a new phase for the web. Jason Henry for The New York Times

APPENDIX: USAGE



5.2. Create a Digital Asset

```
from bigchaindb import crypto

# Create a test user
testuser1_priv, testuser1_pub = crypto.generate_key_pair()

# Define a digital asset data payload
digital_asset_payload = {'msg': 'Hello BigchainDB!'}

# A create transaction uses the operation `CREATE` and has no inputs
tx = b.create_transaction(b.me, testuser1_pub, None, 'CREATE', payload=digital_

# All transactions need to be signed by the user creating the transaction
tx_signed = b.sign_transaction(tx, b.me_private)

# Write the transaction to the bigchain.
# The transaction will be stored in a backlog where it will be validated,
# included in a block, and written to the bigchain
b.write_transaction(tx_signed)
```

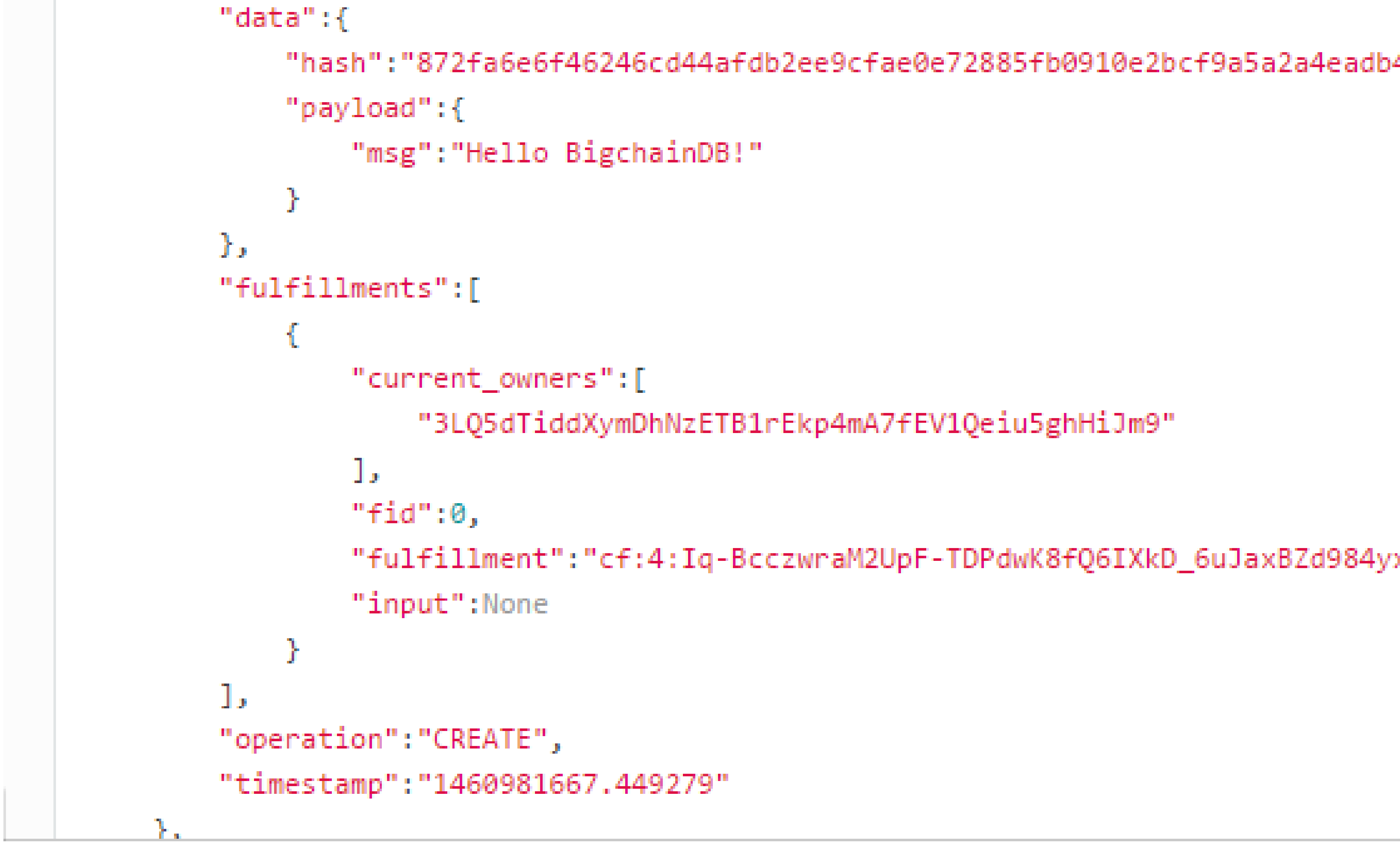


5.3. Read the Creation Transaction from the DB

```
# Retrieve a transaction from the bigchain
tx_retrieved = b.get_transaction(tx_signed['id'])
tx_retrieved
```

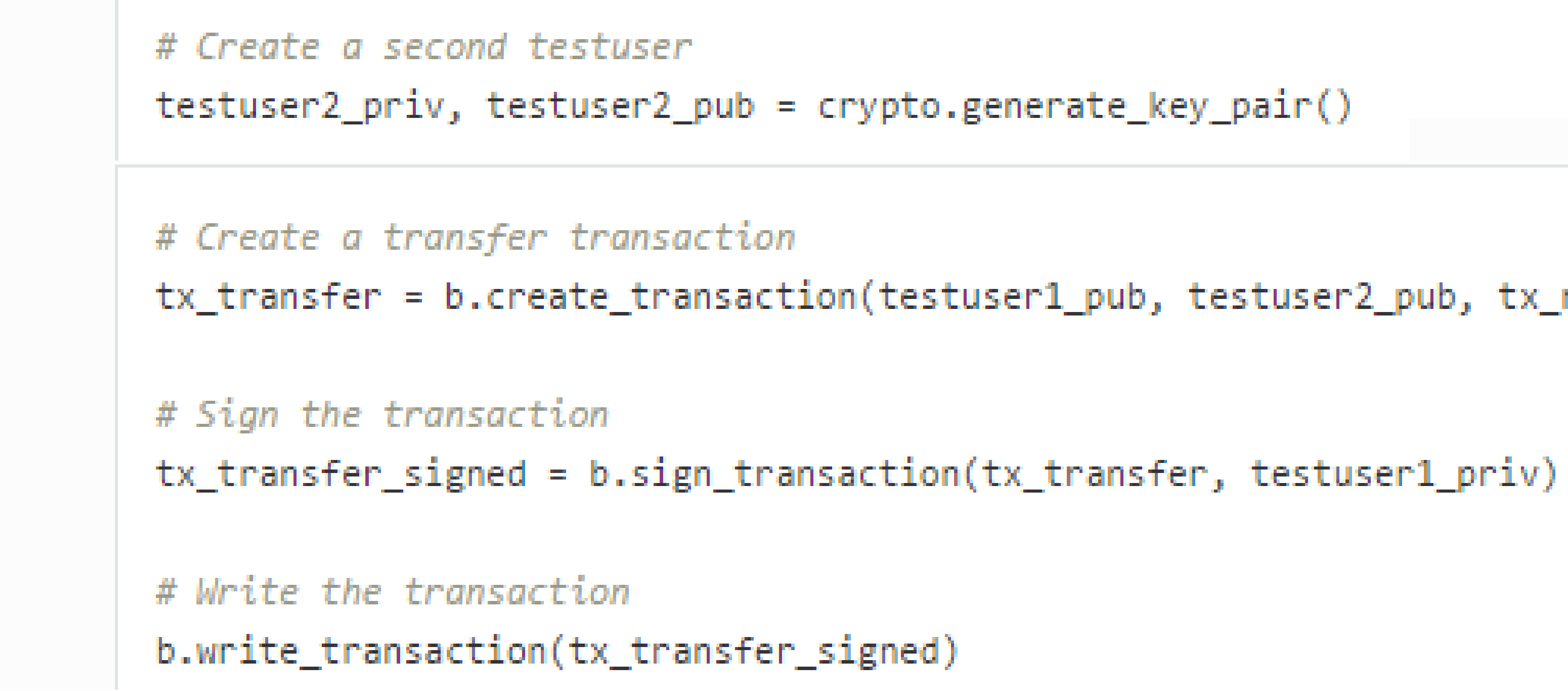
```
{
    "id":"933cd83a419d2735822a2154c84176a2f419cbd449a74b94e592ab807af23861",
    "transaction":{
        "conditions":[
            {
                "cid":0,
                "condition":{
                    "details":{
                        "bitmask":32,
                        "public_key":"BwuhqQX8FPsmqYiRV2CSZYWWsSWgSSQQFHjqxKEuq
                        "signature":None,
                        "type":"fulfillment",
                        "type_id":4
```

5.3. Read the Creation Transaction from the DB



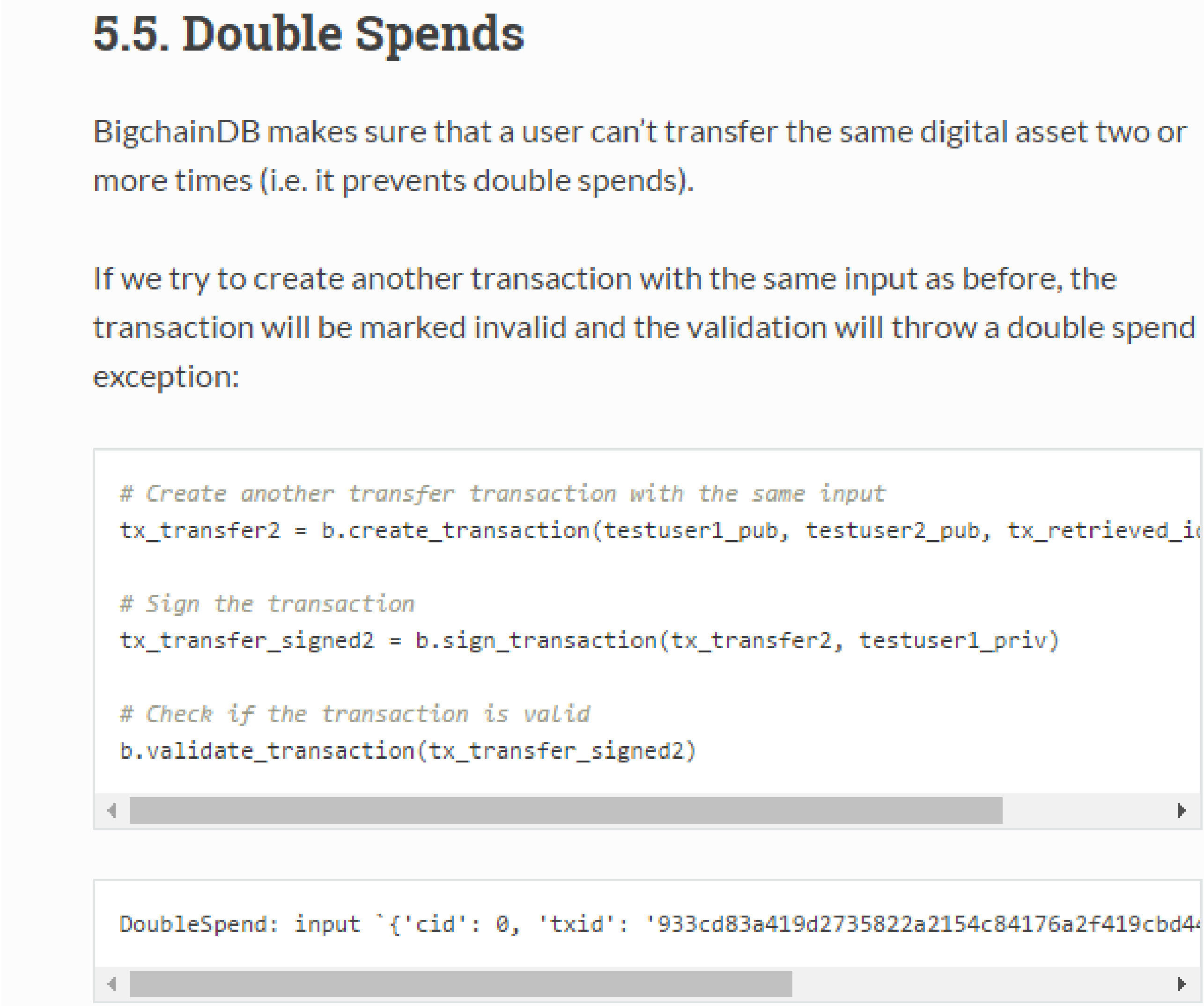
```
            "data":{
                "hash":"872fa6e6f46246cd44afdb2ee9cfae0e72885fb0910e2bcf9a5a2a4eadb4
                "payload":{
                    "msg":"Hello BigchainDB!"
                }
            },
            "fulfillments":[
                {
                    "current_owners":[
                        "3LQ5dTiddXymDhNzETB1rEkp4mA7fEV1Qeiu5ghHiJm9"
                    ],
                    "fid":0,
                    "fulfillment":"cf:4:Iq-BcczwraM2UpF-TDPdwK8fQ6IXkD_6uJaxBZd984y
                    "input":None
                }
            ],
            "operation":"CREATE",
            "timestamp":"1460981667.449279"
        },
```

5.4. Transfer the Digital Asset



```
# Create a second testuser
testuser2_priv, testuser2_pub = crypto.generate_key_pair()
```

```
# Create a transfer transaction
tx_transfer = b.create_transaction(testuser1_pub, testuser2_pub, tx_

# Sign the transaction
tx_transfer_signed = b.sign_transaction(tx_transfer, testuser1_priv)

# Write the transaction
b.write_transaction(tx_transfer_signed)
```



5.5. Double Spends

BigchainDB makes sure that a user can't transfer the same digital asset two or more times (i.e. it prevents double spends).

If we try to create another transaction with the same input as before, the transaction will be marked invalid and the validation will throw a double spend exception:

```
# Create another transfer transaction with the same input
tx_transfer2 = b.create_transaction(testuser1_pub, testuser2_pub, tx_retrieved_i

# Sign the transaction
tx_transfer_signed2 = b.sign_transaction(tx_transfer2, testuser1_priv)

# Check if the transaction is valid
b.validate_transaction(tx_transfer_signed2)
```

```
DoubleSpend: input `{'cid': 0, 'txid': '933cd83a419d2735822a2154c84176a2f419cbd44
```

	Traditional blockchains	Big Data	BIGCHAIN DB
Decentralized	☑		☑
Immutable	☑		☑
Assets	☑		☑
Scale: Throughput, Capacity, Latency		☑	☑
Query Capabilities		☑	☑