Trent McConaghy Follow
AI*blockchain. Founder & CTO @OceanProtocol | @BigchainDB | @IPDBFoundation. #AI #blockchain #data #Singularity #eResidency www.trent.st
Jul 10, 2016 · 14 min read

The DCS Triangle

Introduction

In this post, I introduce the DCS (Decentralized-Consistent-Scale) triangle as a way to compare tradeoffs among some of today's popular decentralized systems, including Bitcoin, Ethereum, IPFS, and BigchainDB. Each system has its own complementary uses. This is an engineering statement of technology today, not of fundamental boundaries like in the CAP theorem. As engineering progress is made, I envision systems that are fully DCS.

Reasoning in Ten Dimensions

With any technology, there are tradeoffs. With new technology, the tradeoffs are often harsher. The first iPhone barely functioned as a phone. But it went Blue Ocean: it optimized on new dimensions that weren't even considered before. So people bought it, and gave it space to mature. Same for the first VR (virtual reality), and the first AR (augmented reality).

BigchainDB is a new technology, with tradeoffs too. Internally, we've been using about ten dimensions to assess BigchainDB: throughput, latency, capacity, queryability, permissioning, decentralization, immutability, asset support / Turing-completeness, privacy, and security / fault tolerance (to crash and Byzantine faults).

But our ape brains aren't very good at contemplating ten dimensions at once. It's hard to explain to others. I've found myself in conversations trying to explain how BigchainDB is complementary to, say, Ethereum or IPFS, in an objective-as-possible way. Where on the Pareto Optimal Front (tradeoff) is Ethereum? IPFS? BigchainDB? Ten dimensions also complicates planning.

What can help is to reduce from ten to fewer dimensions, where the dimensions are still meaningful enough to reason about them, and they can be visualized. We can visualize one and two dimensions as a line and two-axis plot respectively. For three dimensions, we could use a 3-

D plot, though that's hard to use. Triangles are better. Beyond three dimensions, simple visualization gets much harder.

Reasoning With Triangles

Let's stick with three dimensions. Triangles are great for reasoning about tradeoffs in three dimensions. Triangles are simple, familiar, and visually flat.

A famous example is the Scope Triangle, which illustrates the tradeoff in cost, time and quality for managing projects.

Tradeoff triangles also show up in distributed systems:

- **Zooko's triangle**: in a network, names are ideally human-meaningful, secure, and decentralized. Pick any two.
- **Brewer's CAP theorem**: a database is ideally **c**onsistent (all nodes always in sync), fully **a**vailable (it's always up), and **p**artition tolerant. Pick any two. Another way of framing: if the network partitions, you get to remain available or consistent, but not both.

What's cool: at first glance these look like theories that you just can't get past. But smart engineers sometimes find a way around. For example, Aaron Swartz squared Zooko's triangle by leveraging a blockchain. Engineers sneak past the FLP impossibility theorem via randomization, additional timing assumptions, "eventually" really fast clocks (like Spanner), and other tricks.

The DCS Triangle

Back to BigchainDB. It turns out we can choose three dimensions that illustrate the tradeoffs made by BigchainDB compared to those by Ethereum or IPFS.

Those three dimensions are:

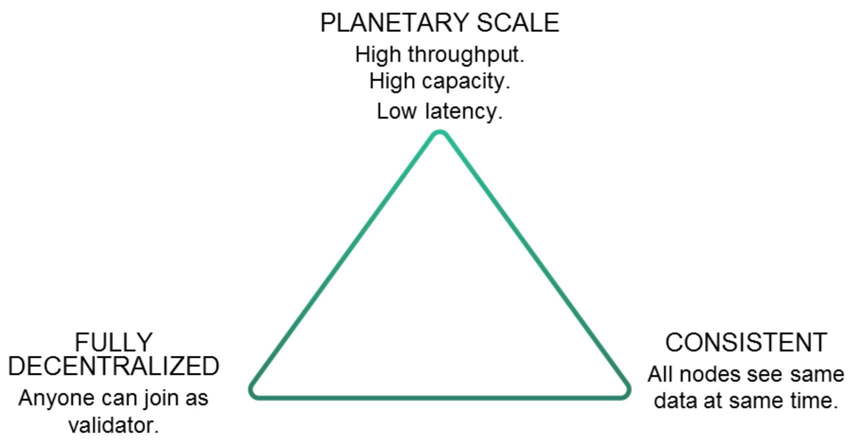
1. **Decentralized**, where no single entity controls the network. Big "**D**" means server-free (fully) decentralized; anyone can join the network as a validating node. Little "**d**" means server-based decentralized. If not D or d, the system is centralized.
2. **Consistent**, where the network aims to keep data in sync. Big "**C**" means all nodes see the same data at the same time. Being big-C consistent is a prerequisite to preventing double-spends, and therefore storing tokens of value. There are many models of

consistency; we mean “consistent” in the same way that the CAP theorem does, except a little bit more loosely (for pragmatism). Little “**c**” means strong eventual consistency, such that when data gets merged there are no conflicts, but not consistent enough to prevent double spends. If not C or c, the system is not consistent.

3. **Scale.** Big “**S**” is planetary **s**cale. This means means sufficient performance characteristics to serve planet-scale or enterprise-scale needs, as typically seen in “big data” distributed databases. This includes throughput of 100,000 tx/s, 1M tx/s or more; capacity in the hundreds of TB, or PB, or more; and latency of <1 s (takes into account speed of light delays in a WAN setting). For reference, typical stock exchanges and ad networks run 100,000–500,000 tx/s. If not S, the system is not planetary scale.

We will elaborate on each of these dimensions later.

The three dimensions form a triangle:



Like the other triangles, there’s a tradeoff among the characteristics. This is what we have today. You can have any two, but not three. This is an engineering statement of technology today, not of fundamental boundaries like in the CAP theorem. Just as smart engineers pushed the boundaries of other triangles, we see that happening here as well.

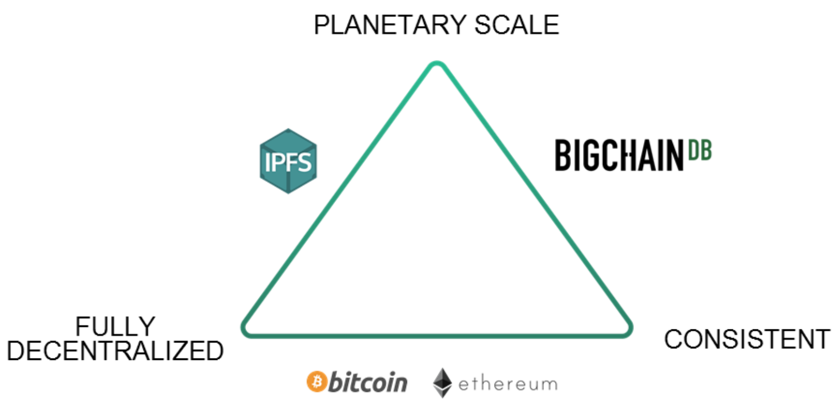
We aren’t able to *compress* all dimensions into three. Rather, we chose dimensions that highlight tradeoffs, to facilitate communication. The dimensions do not capture privacy or security, for example.

> *“All models are wrong, but some are useful”*

— *G.E. Box*

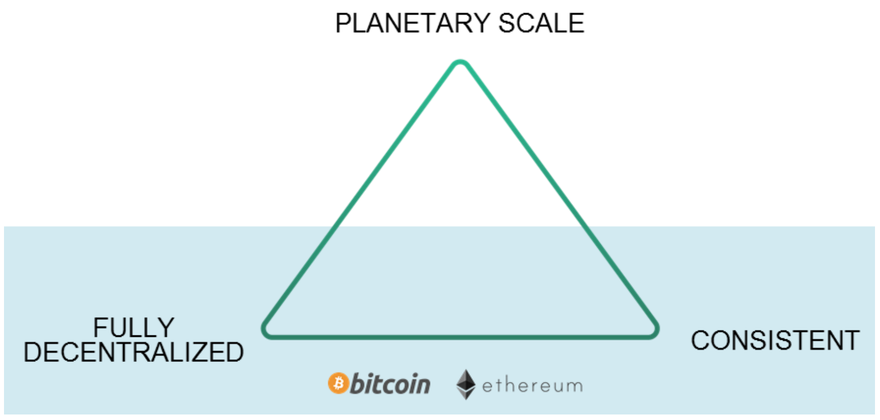
Systems on The DCS Triangle

Let’s explore representative systems, and how boundaries are getting pushed. First, all at once: below are systems that manifest each tradeoff. Bitcoin and Ethereum are DC. IPFS is DcS. BigchainDB is dCS. I’ll elaborate on each.



DC Systems

Bitcoin and Ethereum embody systems that are DC: big-D **Decentralized** and big-C **Consistent**, but not planetary Scale.



Both currently run a Proof of Work (PoW) based consensus algorithm, where every full node stores all the data. No one entity controls the network. Anyone can join the network and run a full node. In running a full node, one has a chance to validate a block of transactions proportional to one’s ability to run hashes. (These days one doesn't really have much chance of actually validating transactions because

ASIC farms have orders of magnitude more hash rate than a typical user; but it's still a theoretical difference.)

Both Bitcoin and Ethereum are consistent, in that all nodes see the same data at the same time. This actually isn't entirely true in a theoretical sense, because they never have a deterministic guarantee of a consistent order; they're only eventually consistent (in a probabilistic sense) based on longest chain rule as in Bitcoin, or the weighted-tree GHOST protocol as in Ethereum. But let's be generous and call them consistent, because in practice they are used that way, the workaround being higher latency as one waits for a sufficiently high probability of avoiding inconsistency.

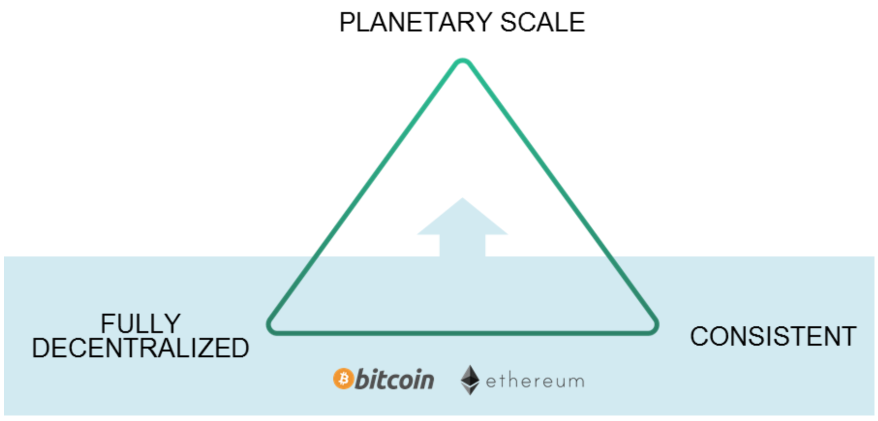
Neither Bitcoin nor Ethereum are planetary scale. The Bitcoin network has a theoretical maximum of 7 tx/s, and even lower in practice. Post-Segwit this will be 1.4x higher, but nowhere near 100K tx/s. The Lightning network does not fully solve this because it gives up other major advantages of blockchains for speed, namely transparency, auditability, storage, and a different trust model. (But of course it's useful for a series of value transfers between the same two actors.) Latency is 60 minutes: 10 minutes for one block confirmation; one typically waits for 3–6 blocks for sufficient probability the block remains stable. Capacity is limited: Bitcoin users worry about "bloat" despite holding just 70GB of data. Ethereum is roughly 10x better for each characteristic, but that's still far short of planetary scale.

DC Roadmap

Bitcoin and Ethereum's difficulties in reaching planetary scale are partly because each node must store all data; in database parlance this is a "replication factor of *n*". If we want the network to hold 1PB, each full node needs to hold 1PB. Is every full node user going to run their own datacenter?

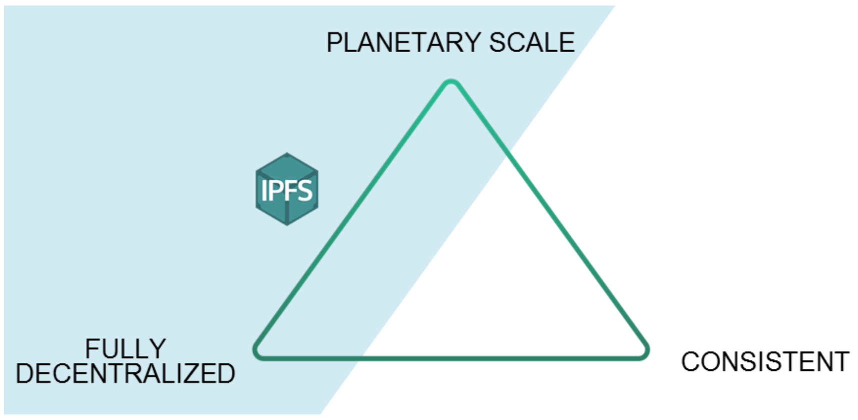
How might Bitcoin and Ethereum get closer to planetary scale? An obvious partial answer is sharding, where every node stores just a fraction of the data. There would be at least a couple backups too, for a replication factor 3 (and probably much higher). But neither Bitcoin nor Ethereum have that yet. It's coming, though. For example, Ethereum Foundation's Casper research is promising. Another somewhat-obvious answer is erasure coding, like what RAID arrays do. It takes less disk space than sharding, to get the same redundancy guarantees. Ethereum Swarm will be using it.

To summarize, shown below, Ethereum and Bitcoin are DC, but moving towards the S (planetary Scale) vertex.



DcS Systems

IPFS (InterPlanetary File System) is an example of a decentralized systems that is DcS: Big-D **Decentralized**, and big-S **Scale**. For those, they compromise Big-C consistency. However, they do have little-c consistency.



IPFS is big-D decentralized, in that no one controls the network, and anyone can join and be a node. Validation is simpler as it's just about checking hashes; there is no strict order of transactions to maintain. There is of course an implicit ordering that arises via hashes on previously hashed objects.

IPFS is planetary scale. Throughput and latency on a node is only limited by the node's bandwidth and ability to ingest the data. Capacity on a node is only limited by the node's capacity; a given node doesn't try to store the whole network. It's like a mashup of BitTorrent and Git; neither has shown significant scale issues. Part of what makes it possible to be planetary scale is that it compromises on consistency.

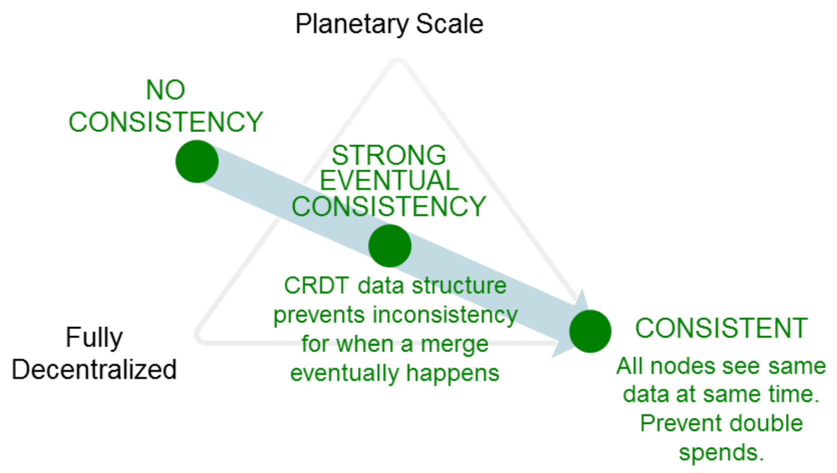
IPFS is not consistent in the CAP sense, because all nodes do not see the same data at the same time. And, it doesn't try to do this. You need consistency if you want to prevent double spends, which is a prerequisite if you want to store tokens of value. But there are plenty of applications that don't need tokens, such as media assets, documentation, and data. These are great use cases for IPFS, and why it's so nicely complementary to other systems.

Degrees of Consistency

IPFS does have a degree of consistency, however. For our FCS triangle, three levels are useful:

1. **No consistency**
2. **Strong eventual consistency, aka little-c**
3. **CAP consistency, aka big-C**

The figure below illustrates.


Planetary Scale
NO CONSISTENCY
STRONG EVENTUAL CONSISTENCY
CRDT data structure prevents inconsistency for when a merge eventually happens
Fully Decentralized
CONSISTENT
All nodes see same data at same time. Prevent double spends.

IPFS could have kept it simple and had no consistency, which would have still been pretty useful. But it went farther: it achieves **strong eventual consistency** via a **CRDT** (Conflict-free Replicated Data Type). CRDTs are a recent and super-cool computer science innovation.

Imagine a network partitioned into two, where parties on both sides kept adding data. Now imagine trying to merge the changes. With a code-versioning system you might have merge conflicts. But with CRDTs, the data structures guarantee that there are no merge conflicts. Amazingly, the space of possible data structures is still quite broad. For example in a messaging system, it basically interleaves messages.

Similar in a shared Google Doc, which is why you've never had to do a version-control style merge in Google Docs.

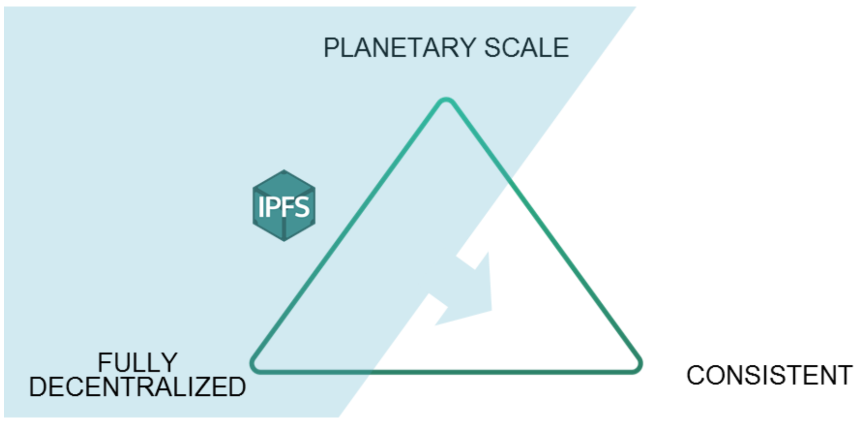
DcS Roadmap

But CRDTs do not do everything. All IPFS nodes do not see the same data at the same time. Therefore in IPFS, one cannot prevent double-spends, and therefore one cannot have tokens of value in it. As mentioned before, this is fine because IPFS has many other uses.

But, IPFS is moving towards greater consistency. Its protocol stack has a place for consensus algorithms to achieve CAP-style strong consistency; and the IPFS team is working on consistency algorithms. Though, this may not be truly "CRDT" style when done.

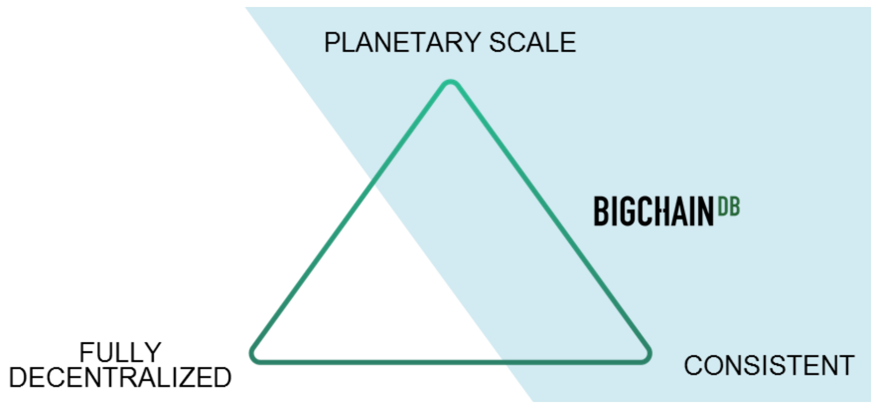
Related, Victor Grishchenko proposed a "Swarm Coin" built on top of a CRDT. The idea is not to make double spends theoretically impossible, but to
make them expensive, and cheap & fast to detect. If a double spend *does* happen, an "Entanglement matrix" manages the split.

The picture below illustrates the trend.



dCS Systems

BigchainDB is a decentralized system that is dCS: big-S **Scale**, and big-C **Consistent**. For these characteristics, it compromises being big-D decentralized; though it is little-D decentralized.



BigchainDB is built on top of a traditional “big data” distributed database, with three characteristics added: decentralization, immutability, and assets. It achieves decentralization via a federation of server nodes. Client nodes can talk to the server nodes, to read transactions, register assets, transfer assets, and create more complex transactions like multisig and escrow. Immutability (greater tamper-resistance) is via hashing, backups, and more.

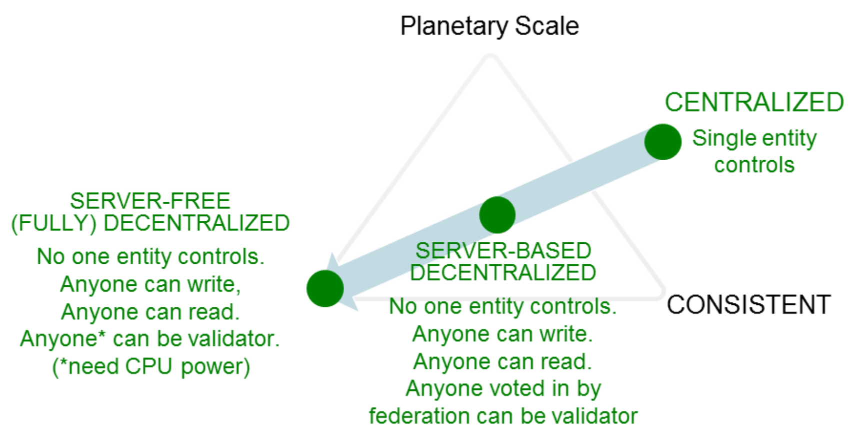
BigchainDB is consistent in the CAP sense: all nodes see the same data. BigchainDB leverages its lower-level distributed DB (RethinkDB) to order transactions deterministically, using its consensus algorithm. BigchainDB’s higher-level federation gives each server node one vote whether each transaction is valid.

BigchainDB inherits the planetary scale characteristics of its lower-level distributed DB. It’s designed to “get out of the way” of the raw performance of RethinkDB. The design assumes that bandwidths and writes are the main constraints, and wherever there are processing bottlenecks, to parallelize it. RethinkDB’s raw performance is 1M writes/s, and BigchainDB with full end-to-end transactions is >100K tx/s (running on just python!) and soon to be significantly higher. WAN latency is <1 s; the major bottleneck is speed of light as messages cross the planet. Capacity is a linear function of the number of shards; for example if one uses an Amazon XL instance with 48TB shards then one can get 1 PB capacity with 64 shards and replication factor 3. (Large shards like this do however increase the database point-in-time recovery.)

BigchainDB is decentralized: no single entity owns or controls a given network deployment. But it is not fully decentralized, where anyone can be a validator. This compromise was needed to achieve planetary scale. Let’s elaborate.

Degrees of Decentralization

There are three degrees of decentralization:



1. **Centralized.** A single entity controls the network. One could have a centralized distributed system: distributed because the resources are spread across more than one physical machine, but centralized because control is still in the hands of one entity. The consensus algorithm only needs to handle crash faults because all nodes are altruistic. This is the model of how Google, Facebook, and so forth deploy their networks.

2. **Server-based decentralized, aka "little-d".** Compute resources are distributed, but no single entity controls the network; to be a validator one must be voted in by the existing federation. The consensus algorithm needs to handle both crash and Byzantine faults. Clients may be anonymous, but not validators which must have known identities. One can deploy a public network (like IPDB) where anyone can write to or read from the network. This is also known as a federation, or a Super-Peer P2P network.

3. **Server-free (fully) decentralized, aka "big-D".** Compute resources are distributed, but no single entity controls the network; anyone can be a validator. Anyone can write to or read from the network. Clients and validators may be anonymous. It must handle crash and Byzantine faults. And because anyone can be a validator, it must handle "attack of the clones" Sybil faults. This is also known as a (pure) peer-to-peer network.

Each point on the decentralization spectrum has its uses. The first— centralization— is what most systems have been built on until recently. It's usually easier to optimize for performance, especially latency which does best where things are spread minimally. Many distributed, centralized systems are planetary scale. It's easier to control a network which can be useful as its builders evolve the technology. It's also easier for a company to reap the rewards of the network effect, as we've seen with companies like Facebook amass tremendous wealth. But there's a glaring negative: it's power in the hands of a single entity, which leads to all sorts of issues. Decentralization is a path to reduce these negative impacts.

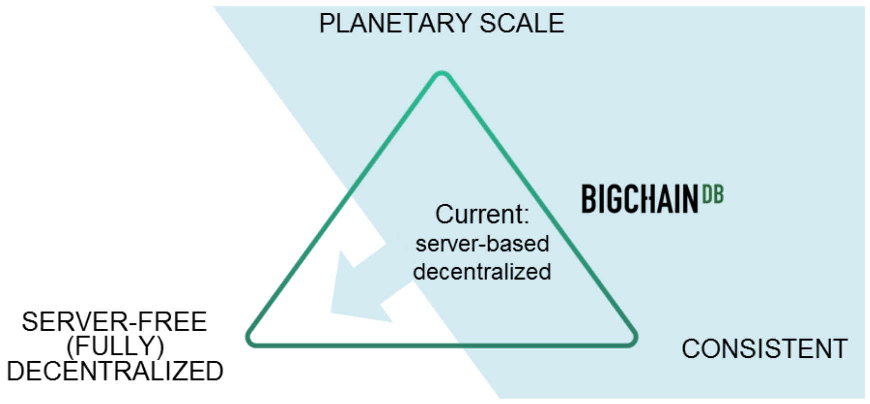
The last point on the spectrum— server-free decentralized—is an ideal because it spreads validating power among clients who is wishing to participate. And, that validating role remains anonymous. To allow anyone to validate yet be anonymous, these systems must be Sybil tolerant, that is, prevent an "attack of the clones" where some entity makes thousands of copies of themselves and takes over voting control. Bitcoin (and current Ethereum) solves this by making an entity vote with electricity. Others make an entity vote based on amount of stake in the system (Proof of Stake). Another way is Stellar Consensus Protocol, where each node selects who its validators are, though that isn't sharded yet either. This means, so far, we haven't seen a way to solve Sybil attacks while still achieving planetary scale. This is a real problem: there are countless efforts to build decentralized applications, where the toy proof-of-concept goes fine, but there is no way to follow that up with a production system at planetary or enterprise scale. We encountered this ourselves: we built ascribe.io on Bitcoin, but we were unable to serve customers having 100,000 tx/day. If we continued using the Bitcoin network we would swamp it and incur thousands of dollars in daily transaction fees.

The middle point on the decentralization spectrum—server-based decentralized—balances the desire for fully decentralized with the need for planetary and enterprise scale. Voting power is spread among the validation nodes, whether it be 15 or 50 or more. Each node gets one vote. One can roll a public version of this, such as IPDB, where client nodes are able to read any transaction, and write what they like. This gives most of the benefit of decentralization—control spread out; and with it transparency, auditability, and so forth—while getting the crucial characteristic of planetary scale. The middle paradigm is also useful for deployment as a "consortium DB" to ecosystems and enterprises, who may not want nor be able to disclose all their information to the broader public, such as with financial regulations.

To summarize: if you want anonymous participation and Sybil tolerance, then server-free decentralized makes sense. Otherwise, server-based decentralized makes sense, using either an existing public network (like IPDB) or in rolling your own network.

dCS Roadmap

It’s this middle paradigm—server-based decentralized—that we aimed for with BigchainDB, knowing that we had to compromise on server-free (fully) decentralized to start with. We also recognize that for public deployments, server-free is an ideal worth working towards. In our roadmap we aim to build a big-D server-free version. (There will always be use for the server-based version, for enterprises.) The figure below illustrates.


PLANETARY SCALE
BIGCHAIN DB
Current: server-based decentralized
SERVER-FREE (FULLY) DECENTRALIZED
CONSISTENT

Conclusion

This post introduced the DCS triangle to show how the characteristics of **decentralized (D)**, **Consistent (C), and** planetary **Scale (S).** They get traded off in modern decentralized systems such as Bitcoin, Ethereum, IPFS, and BigchainDB. Bitcoin and Ethereum are DC, IPFS is DcS, and BigchainDB is dCS. Each system has its own complementary uses.

This is an engineering statement of technology today, not of fundamental boundaries. (Contrary to, for example CAP, which is a fundamental boundary). As engineering progress is made, I envision systems that are fully DCS.

Update

I wrote this in July 2016. A few months later, Vitalik Buterin and Greg Slepak each discovered the same thing, here and here. Cool! As of Feb 2017, all three of us are aware of the others’ work.

Acknowledgements

Thanks to the following folks for discussions that led to this post, and for reviews of this post: Dimitri de Jonghe, Bruce Pon, Troy McConaghy, Juan Benet, Ryan Henderson, Gavin Wood, Jae Kwon, Victor Grishchenko, Alberto Granzotto, Tim Daubenshütz, and

Wojciech Hupert (and almost certainly several more that I missed-sorry).