# Holochain / Holo Training & Empowerment

Perry Kundert

2022-07-09 13:00:00

**What if you could train your team, and gently transition to a fully decentralized Holochain architecture?**

Owning/managing centralized databases, compute and networking is a business risk:

- Physical compute and networking equipment is expensive and unresponsive to growth.
- Hosted capacity is expensive, and you may be deplatformed at the whim of any politician (or mob).
- Crypto-based decentralized applications (dapps) are crushingly expensive to deploy and operate, and cannot scale past a few transactions per second.

Holochain is a completely novel (Agent-centric, not Data-centric) re-imagination of reliable distributed and decentralized system architecture, and addresses these problems.

It is available to begin prototype development, and deploy fully operational test systems at Internet scale. In the coming months (late 2022), Beta public deployments will be possible.

Begin your team's empowerment with customized on-site training in Holochain / Holo, and the Rust, Nix, etc. technology stack used to build decentralized solutions. (PDF)

## Contents

# 1 Why Agent-centric vs. Data-centric Structures?

A massive, centralized database – even if it is sharded over many distributed hosts – still presents a central point of failure, and a choke-point for application scaling. All read/write traffic must eventually move from every client to/from one (or a set) of hosts, who must (eventually) agree on a single, global state.

## 1.1 Data-centricity: Finality and Single "Total Order"

Traditional and "NoSQL" databases, and global-consensus blockchains primarily differ on how they enforce agreement on this global single state. But, agreement there is. This takes time, and massive coordinated effort and energy. The current theoretical maximum appears to have been achieved by systems like Hashgraph, which can achieve this with cryptographically assured finality at a rate of around 50,000 TPS for global-scale systems (primarily limited by the speed of light, and the separation of the systems agreeing).

Other blockchains generally do not provide assured finality at all (only statistical finality), and do so at rates far lower than Hashgraph. Bitcoin's PoW algorithm for sybil resistance provides strong assurances against centralized hijacking of total-order consensus, but at the cost of huge energy consumption and a transaction rate of less than 10 TPS. Other blockchains provide other features, but generally are characterized by various attempts to improve thruput by weakening various guarantees, or by inflicting other "costs" on attackers, to weaken the financial viability of various attacks.

### 1.1.1 So What?

Step back and ask the big question, though: why?

Why does it *matter* whether Fred's Lightning-network transaction buying a coffee in El Salvador came immediately before or after Yegor's transaction in Abu Dhabi selling his yacht for Bitcoin?

Does it even really makes sense to compare these transactions? Their light-cones don't even intersect for a fraction of a second – the concept of "simultaneity" between them doesn't exist. Why are all data-centric systems in the world needing to frantically (and expensively) beat out some globally agreed-upon "total order" between these events?

## 1.2 Agent-centricity: Validity Always, Consensus When You Need It

With Holochain, a key breakthrough came by accepting the fact that most decentralized transactions don't really **need** a global "total order" over every other transaction in a decentralized system.

Blades of grass only "duke it out" over H2O and CO2 when their blades are touching – most of the time, your lawn (and its ants, ...) only need to agree on who did what first in very localized instances of competition over resources. Most transactions in distributed systems are similar.

### 1.2.1 Inductive Proof of Validity

Every state change is established by an agent committing a chain of Actions referencing Records to an individual blockchain (we call it a Source Chain). The Action (and often the Record) are published to a DHT (Distributed Hash Table).

Every node collaborating in the decentralized computation hosts an "arc" of the DHT hash space – but **only** if it can independently **validate** it! This requires 2 things, which form the foundation of an inductive proof of correctness:

1. That the Action and/or Record *N* is verified to have been computed correctly by the claimed DNA/Zome code, and
2. That each predecessor state (Action and possibly Record *N-1*) has been hosted by the DHT, and not "Warranted" as incorrect by any node (and the warrant was correct).

If both of these things are true for state *N*, then by induction, all prior states *N-1*, *N-2*, ... are also valid, back to the origin of each agent involved .

### 1.2.2 Localized Consensus

Once operating in a substrate of **validated** Actions and Records in a DHT, 2 or more agents can agree on some state change that must occur *if and only if* a quorum of the agents agree on it. For example,

1. They establish some base agreement about something (eg. commit some *Proposal* to their respective *source chain* and the DHT).
2. Then once they all observe that a sufficient quorum of counterparties has agreed (by examining the DHT and perhaps looking at their counterparties source chains, and observing each agreeing *Proposal*) – they each commit a bit-identical *Transaction* record to each of their individual *source chains*.
3. At any time before committing a *Transaction*, any counterparty can back out, and just leave the process, or commit a *Retraction*. If so, they cannot (or will not) ever commit a *Transaction* commit.
4. Once one agent collects a sufficient quorum of Ed25519-signed *Transaction* commits, they have non-repudiable proof of Consensus. Even if they received the commit out-of-band (eg. not from the DHT, but from private communications with one or more counterparty agents) – the Ed25519-signed commit is adequate for any agent to publish the Action (and Record) on behalf of the committed agent(s), even if the agent has been destroyed the instant after it produced the signed agreement.
5. Once published to the DHT and propagated, the *Transaction* represents a **global** change in state to the distributed system, that can be depended on by any other agent in computing its state changes. For example, if the *Transaction* represent a

change in a "mutual credit" account balance between multiple agents, other agents unrelated to the *Transaction* can depend on it: the agent's balance is valid, and it is inductively proven that all predecessor "mutual credit" balances leading to it were also valid.

# 2 Learn to Reframe your Data-centric Architecture

It takes years to learn to implement and operate Data-centric algorithms at scale.

It may take some time to re-frame your problem domain in terms of Agent-centric algoriths and implement them on Holochain.

## 2.1 Incremental Transition

Fortunately, many traditional systems are implemented using a layered approach; a client application (Web or traditional) which accesses centralized resource(s) via APIs.

These are very suitable to an incremental transition to Holochain.

### 2.1.1 Re-architect Some Subset of Shared Data

A set of rules are implemented for **creating** and **storing** data (used by every agent collaborating in the application), and other rules to **validate** the data (used by every agent hosting DHT Action/Record data). These are call the "DNA" of the application, and are written in Rust. They are often quite small, but can be arbitrarily large and complex, if required.

This DNA code is deployed to a constellation of Holochain agent nodes. You can run these all yourself, or (if desired) contract agents to host the DNA on your behalf (eg. via something like `https://holo.host`).

This decentralized fleet of Holochain agents accesses each-other as defined by your DNA code, and provides a decentralized API available to any client who wishes to execute functionality or access data available via your API.

### 2.1.2 Deploy Some Front-End Code Accessing The API

Standard Javascript for authenticating with the Holochain-implemented API is available.

Holochain clients do not typically use centralized credentials (eg. API keys or passwords), but instead create an agent under the cryptographic control of the user. This Holochain agent is operated by the user, who *always* holds the Ed25519 signing keys, and interacts with other agents to execute the user's will in the system.

1. What? My Users Aren't Crypto Experts!

   Of course, users don't usually realize they are doing this. They're just "logging in", as usual. But in actuality, their login credentials produce the Ed25519 private signing key which unlocks both the history of their agents actions, and holds the authority for them to execute future actions.

Systems are provided by eg. `https://holo.host` to help manage the transition from centrally managed authentication, to user-owned agency. The reduction in risk surface (eg. hacking and loss of user credentials) is a significant potential business benefit. Your users personally benefit from guaranteed individual agency (they cannot be de-platformed) and non-repudiability of action (they can prove and cannot disavow their personal actions).

### 2.2 Who Wants This?

Not everyone wants the things made possible by systems like Holochain. But many organizations and individuals are awakening to the benefits of individual liberty, autonomy, authority and responsibility afforded by cryptographically secured distributed systems.

## 3 How To Get Started

Reach out to Perry Kundert at Dominion R&D Corp. (perry@dominionrnd.com). We can do a one-week session with a handful of your system architects for USD$5,000 (plus lodging and airfare from Calgary or Edmonton, Canada).