DATA LAKES

AND THE BLOCKCHAINS

prepared by

neuroware

Our understanding of a data-lake is as follows

- An intelligent repository for storing large quantities and varieties of data
- With a constant stream of data being fed to it from different sources
  - There is an **expectation** to preserve the original data
  - As well as the **precise** lineage of data transformation
- Requires unstructured data & structured data with embedded schema
- Leads to a stockpile of archived anonymous yet actionable events
- **However**, the more routes that lead to and from this lake of data, the more susceptible it becomes to presenting misleading information…

Where micro-services distribute workloads

- Like individual departments within one large organization - the task of generating & verifying information is then managed by different entities
- Each procedural component specializes in just one thing…
- But it can be utilized by other teams from within the eco-system
- Like components in platforms, teams are independent of each other
- Teams have their own internal hierarchal policies and procedures
- This requires a trusted immutable distributed historical log of events
- Until now, this has not yet been completely possible to achieve…

The problem with independence is TRUST

- **How can we trust the data coming into the lake?**
  - Is it going to the right place & is it coming from who it says it is?
  - If the application or service is independent what happens when we need to change either of these parameters?
- **Can we trust the data does not get altered once in the lake?**
  - Risks associated with centralized truth for distributed workloads
- **How can we trust the data coming out of the lake?**
  - Do the right team members have the data specific privileges?

Distributed ledger protocols can help with trust

dnkey

- Our **DNKey** protocol could be used to manage hierarchal public key abstraction so that external input routes could be controlled remotely

blockauth

- Our **BlockAuth** protocol could be used to manage network-wide, inter-internal hierarchal authentication and administrative privileges

EVERSTORE

- Our **Everstore** protocol could be used to track and publicize data archives with hashed payloads and other important meta information

Especially as data flows between these points

COLLECT DATA

Collect RAW schema-less data using MongoDB - can use GridFS for file storage

Document-Driven Design

This allows you to store and query anything

ARCHIVE DATA

Hash batches of data prior to archiving and publicize the meta data to the blockchains

Relational and Replicated

Mixing some of the oldest technologies with the latest

ANALYZE DATA

Use APIs to get data from the archives and prove its validity by checking the blockchains

Polyglot Persistance Prevails

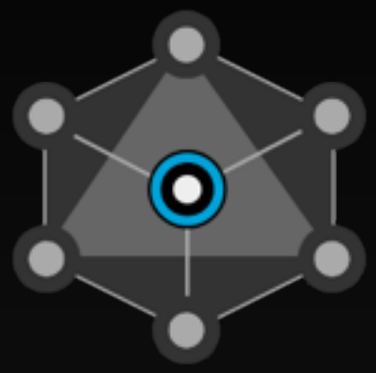
Use the right database for the right job at the right time!

Thinking forward, we could go much further

- Start storing raw results for specific metrics or transparent contents
- Abstracting individual micro-service instructions to the blockchains
- Using individual blockchain transactions for separate events
- Maintaining individual hierarchal key-chains for different users
- Storing query maps and reporting schemas on the blockchains
- Using blockchain-based messaging for encrypted job queues
- We could even start developing a new customized blockchain

But should first identify the next steps

- **Let's first ensure that everyone is on the same page…**
- Sign mutual NDAs and discuss partnership potential in more detail
- Define a project scope within a proposal and obtain an MOU / LOI
- Design a storyboard with an appropriate user experience
- Generate a working MVP



neuroware

BEYOND THE FINANCIAL BOUNDARIES OF BLOCKCHAINS

EMAIL THE TEAM DIRECTLY

founders@neuroware.io