JANUARY 14, 2016

AVALANCHAIN LTD
London

Contents

1 Enterprise applications rationales 3

2 Avalanchain 8

3 Glossary 14

4 Applications 16

5 Conclusion 17

1 Enterprise applications rationales

1.1 Centralization vs decentralization

1.1.1 Disaster Recovery

The first question any responsible manager usually asks before approving a new IT system for production use is "who shall I call if things go terribly wrong?"

In a case of a centralized system the answer is very obvious – call the 1st/2nd/3rd level support, wake them up if necessary, and let them deal with the issue.

In a case of a fully decentralized system this option is not easily available as decentralization assumes different parts of the system to be controlled by different entities from support perspective with unavoidably required coordination between them for system integrity level actions.

From this perspective even Bitcoin is not a fully decentralized system. At the end of the day it is *one* network.

1.1.2 Deployment, monitoring, production support

Same logic applies to other standard operations jobs.

Software releases should follow established procedures.

Users must have dedicated points of contact.

1.1.3 Data storage

Decentralized validation assumes decentralized data storage.

For a case of a large enough organization this require separate Database provisioning and management every node or cluster of distribution. With all the associated burden.

1.2 Consensus

1.2.1 Proof of Work vs Proof of Stake

Direct use of a separate Proof of Work implementation for an in-house system even in a global multinational organization looks like a massive overhead with quite limited business benefits.

That is actually one of the reasons why many Proof of Stake projects has arisen recently. Most of them are way less computationally intensive comparing to PoW based ones but the fact of their variety by itself tells us that none of them solves issues like "nothing to stake" completely. And we probably can expect to see more of "Proof of XXX" algorithms be announced in nearest future.

For more information, please read the Bytecoin article by Ray Patterson:

https://bytecoin.org/blog/proof-of-stake-proof-of-work-comparison/

As an alternative a combination of PoW and PoS can be used (e.g. Proof of Activity - https://bytecoin.org/blog/proof-of-activity-proof-of-burn-proof-of-capacity/) or combination of the Bitcoin and Sidechains (please see Avalanchain Bitcoin nailing section for further details).

1.2.2 Consensus in general

Existing Blockchain implementations select network level consensus mechanisms with common parameters for every application, user and scenario.

In reality this fact creates quite interesting situation when blockchain developers effectively making decisions about how End User's business should operate.

On these grounds it looks like a reasonable expectation for an Enterprise level Blockchain system which is expected to survive for years to have Consensus mechanism selected for it pluggable or swappable depending on actual business needs and scenarios.

1.3 Fees

Let's be honest, within a real organization it is quite difficult to find a rationale to make one department to pay another one some amount of virtual currency for validating their transactions.

Such applications are naturally limited to currency exchange/trading, and different types of order entry applications.

More commonly used patterns usually rather involve renting of real or virtual machines with time-based billing in real department budget currency.

1.4 Data replication

When talking about Blockchain in its original Bitcoin sense many authors using term "Distributed Database". This is a term misuse because of the transactions data is replicated (copied) to every participated node rather than distributed between some of them. So the term "Replicated Database" is more relevant.

"Replicated Database" means the Data Storage requirements grow at least linearly (and network traffic grow super-linearly) together with increasing amount of Validation Nodes regardless of actual feasibility of such a heavy replication. Effectively this doesn't allow effective data partitionability (sharding). (Google for "CAP theorem" for more details.)

Another consideration is the fact that modern Cloud and other infrastructure providers supplying customers with data storage solutions which are already powered by hardware data replication.

1.5 Write vs Read

Vast majority of existing Blockchain implementations supply users with a convenient way of writing data into the system (putting it into a replicated transaction log or distributed ledger). But with reading and especially querying data situation is way less straight forward.

Users presented with the current state of the system (in a case of Bitcoin via balances). But to find out how some specific account used to look like when say the difference between two other accounts was minimal and the fourth one had at least 100 BTC requires writing very non-trivial data processing logic. Additionally it may require to traverse through enormous amount of data as there is no guarantee which blocks contain transactions related to accounts in question.

If we put payments aside and try to create a more general purpose Blockchain based solution then we quickly realize that standard SQL databases simply not designed to work efficiently with data represented as a stream of facts in perfect tense (transactions, events, etc.) rather than in continues tense (balances, final states).

1.6 Blockchain as an Event Sourcing System

All Blockchains are actually Event Sourcing systems.

They all work with streams of transactions coming from different sources into limited number of actual chains.

Unfortunately no available implementations (as of now) actually consider Blockchains as streaming systems, nor try to turn this natural feature of Blockchains into a business benefit.

1.7 Sidechains

The concept of Sidechains has arisen from the general understanding of the fact that Bitcoin doesn't fit it all use cases, in particular it cannot stretch indefinitely to satisfy all very different business scenarios.

One- and two-way pegs were introduced to let custom chains to communicate with Bitcoin and each other.

For the enterprise environment, the ability to physically split information between different chains whilst permitting referencing between them is extremely attractive, not least for security and regulatory reasons (e.g. "Chinese Walls", geographical restrictions, etc.).

1.8 Security

1.8.1 Anonymity

Anonymity and full traceability are a trade-off. Anonymity can create complications with KYC (and other regulations) whilst be desirable, indeed necessary for others (e.g. Swiss banking). From this perspective, anonymity mechanisms originally proposed in the Bitcoin community show promise in cases of controlled use in variety of environments.

1.8.2 Multi-signature transactions

Modern cryptography can now require cryptosystems requiring unavoidable collaboration of a specific number of separate parties (e.g. 3 of 12 company directors) to confirm the decision (or access to some information).

1.8.3 Role-based security

All the following operations in an Enterprise level system are expected to be role-based (and no existing implementation of Blockchains currently provide this).

1.8.3.1 Authentication

A user recently joined the organization should be able to get access to the system (and be able to get recognized by the system).

They should lose this access on leaving the organization.

1.8.3.2 Authorization

Depending on their role(s) within the organization they should have access to different parts of system's functionality. The set of functionality accessible should follow changes in the set of roles assigned to the user and their authorizations.

1.8.3.3 Data encryption

Ability to encrypt and decrypt some data based on requirements of a role covers a variety of data protection use-cases starting from simple "user lost his key" to more complex cases like "information available only for company Directors".

1.8.4 Cryptographically strong proofs

Blockchains naturally allow creation and validation of cryptographic proof of facts.

1.9 Scripting (aka Smart Contracts)

A legitimate expectation of any database user (for decades) is an ability to atomically validate a transactions input or rejection of the transaction if an input has a chance to put the system into a logically inconsistent state. Users need to be provided with ways of not only validating but also actively reacting to inputs and state changes.

Another legitimate expectation is an ability to move the code, or the logic, as close to the data as possible instead of moving data to the code. This ability in itself can create a huge performance

improvement, but if combined with the option of saving the result back to the database with full audit, performance improvements are dramatically enhanced.

Yes, we are talking about Triggers and Stored Procedures.

In Blockchain world these are addressed in two ways:

- by hardcoding processing logic (e.g. payment transaction processing in Bitcoin-like networks), or
- by adding scripting option to the protocol (rather functionally limited op_codes in Bitcoin and full Turing-complete languages in platforms like Ethereum and Codius).

There are of course legitimate reasons for using any of these options but an important point is that the enterprise level Blockchain systems that don't provide convenient and user-friendly triggers and stored procedure functionality will be at a massive disadvantage compared to any (distributed) database or calculation framework.

2 Avalanchain

2.1 Main purpose

2.1.1 Introduce “read friendly” Blockchains

2.1.2 Introduce “Enterprise Ready” Blockchains

2.1.3 Deal with super-linearly growing data volumes

2.1.4 Introduce automatically created chains

2.1.5 Process Transactions close to real-time

2.1.6 Intensify Blockchains with cascading updates

2.1.7 Introduce Blockchain backed CEP

2.1.8 Make consensus decision to be Business driven

2.1.9 Provide users with ultimate data explainability and entitlement

2.1.10 Enable Role Based ACLs on Blockchain

2.1.11 Lightweight asynchronous processing of Smart Contracts

2.2 Chains as Data Streams

2.2.1 Create as many Chains as you like

Each Stream is a separate Chain of digitally signed events.

2.2.2 Chains can be created, stopped or deleted on fly

No need to restart anything to start a new chain.

2.2.3 High volume and minimal latency on every Stream

2.3 Avalanchain as an Event Store

2.3.1 CQRS

Command Query Responsibility Segregation pattern is implemented in order to deal with limitation of the CAP theorem.

2.3.2 Projections

Projections allow you to react to events as they are written, and to create new events when interesting combinations occur.

2.3.3 Event pattern matching

Incoming events selected for processing by Streams using Pattern Matching API which is the most natural fit for streamed data.

Pattern Matching API allows:

- Filter streams for specific events
- Split events in a stream into number of streams based on the value of a property
- Combine events from different streams into one
- On-fly aggregations

2.3.4 Historical queries as First Class Citizens

You can use the same model for writing temporal correlation queries that run over historical data and on into the future.

2.4 Cascading Stream updates

2.4.1 Streams can depend on other Streams

States (calculation results) of parent Streams automatically considered as input events for dependent Streams.

2.4.2 Blockchain backed Complex Event Processing (CEP)

Convenient CEP API is provided for a number of commonly used languages and platforms (please see below). CEP API is optimized by its design to use with streaming data and utilizing current best practices in the space.

2.5 Versioned Streams and Projections

System consistency is maintained by cryptographic guarantee that all Nodes execute exactly the same code processing events of the Stream.

Increasing version of the Projection or Stream Definition will cause either forking or full stream recalculation depending on User's preference.

2.6 Cheap indexes

2.6.1 Indexes are Streams of Spines

2.6.2 Use of indexes is endorsed

2.6.3 Indexes enables high level of parallel processing

2.6.4 Increase performance and lowering IO

2.7 Elastic Scalability

2.7.1 Designed to run on commodity hardware

Just add more machines to the required named Execution Group.

2.8 Hierarchical Referencing and Scoping

2.8.1 Stream are organized in a natural tree-like structure

2.8.2 Permissions are propagated from parent to children

2.8.3 Permissions assigned to both Streams and Execution Groups

2.8.4 Each Execution Group represents execution isolation

2.9 Avalanchain as a Key-Value Store

Every entity in the system (transactions, states, proofs, stream definitions, etc.) can be retrieved in a standard way from the network as long as access is permitted by corresponding stream ACL.

The system supports entity retrieval via Hash, Spine or any number of user-defined Aliases.

The Distributed Data Storage is organized in a way similar to BigTable and Torrent network but redundancy level is finely adjustable and controlled via Stream-level Consensus settings.

2.10 Avalanchain as a functional Database

2.10.1 Projection is a "map"

Map is the most standard and common operation in Functional Programming.

2.10.2 State is a "left fold"

Left fold is a standard operation in FP and equivalent to *"reduce with initial state"*

2.10.3 Stream State is the first derivative of Stream Events

State for any Stream at any point in time can be rebuilt by replaying events from the start of the Stream till that point in time through Stream's Projection.

2.11 Access Control Lists

2.11.1 Role based actions

2.11.2 Roles can match organization's org chart

2.11.3 Roles are based on Group and Ring Signatures

Use Case: Execution Policy (Consensus) requires Transactions in group "My Trades" to be also signed by someone from group "FCA". I shouldn't be able to recover who exactly in "FCA" signed my transaction but for "FCA" the fine-grained audit should be maintained.

2.11.4 Role based Event Data (Content) Encryption

2.12 Stream-level Consensus settings

2.12.1 Real-time transaction processing

For many use cases no need to wait for the block to be signed.

2.12.2 Tunable redundancy guarantees

2.12.2.1 *Guaranteed N nodes*

Prescribed redundancy.

2.12.2.2 *Guaranteed percentage with weighted nodes*

Proof of Stake in all its variations.

2.12.2.3 *Guaranteed named Execution Group*

Every Transaction is validated and approved by "FCA" and "HMRC" Execution Groups before processing any subsequent event.

2.12.3 Natural sharding

Sharding can be achieved by filtering, partitioning or aggregating Events from a Stream. And/or by writing Events into separate Streams in first place and manipulating them using CEP API (Pattern Marting etc).

2.12.4 High Availability

High Availability is available with *Guaranteed N nodes* and *Guaranteed percentage* Consensus settings.

2.13 Unforgeable Audit

2.13.1 Immutability

Events/transactions accepted by the system are digitally signed, places into a Chain and receive their Spine which is their unique and irreversible proof of the place in the Chain.

2.13.2 Explainability

Explainability is the ability to back-trace all the calculations involved in the derivation of any output value. Simply speaking it is an ability to take the output value and see all the calculations (projections, streams) data has got through to build this output. Or just an ability to see a cryptographic proof that it has happen (maybe more preferred for security reasons).

2.13.3 Data Entitlement

This is a direct derivative from the explainability feature. When you can explain every single data output you ultimately can proof if some specific data was or wasn't used during this output derivation.

2.14 Unforgeable Proofs

2.14.1 ECC based Signatures

Digital signature are based on Elliptic Curve Cryptography.

Avalanchain supports a number of curves out of the box and they can be easily swapped or new ones plugged in.

2.14.2 Multisignature Proofs

Some critical operations may require collaborated effort from people with different Roles and/or from different Execution Groups.

Avalanchain provides the Streams with option to have Role or Execution Group based Multisignature Stream Definition.

2.14.3 Merkle Tree based Spines

Transaction Spine is a Merkle Tree based immutable data structure uniquely and unmodifiable identifying Transaction in a Stream. Each Transaction is uniquely identified by its Spine and Spine provides enough information to retrieve all previous events (or their proofs) in the Stream with a minimal number of network requests.

2.14.4 System automatically rejects entities failed Hash or Signature validations

The caller will receive a graceful error message in case his transaction was rejected by executors during Consensus resolution.

All failures are logged and actively audited.

2.14.5 Bitcoin pegging

Event history can optionally be additionally secured via anchoring proofs of Events and States logged by Avalanchain directly on Bitcoin transactions.

This provides immutability equal to the most adopted and powerful PoW consensus and powered by its enormous computational strength.

There is no technical limitation for creating similar pegging to any other public chains if this will be viable in future.

2.15 Consistency

The system guarantees:

2.15.1 Immutability

Events/transactions accepted by the system will never change.

Calculation results once agreed will never change.

2.15.2 Eventual Consistency

Read-optimized consistency.

2.15.3 Bi-temporality

Bi-temporality is a method of allowing logical backdated amendments while maintaining consistency of historical queries.

The method supports two types of historical queries:

2.15.3.1 *How my balance for the 1st of last month look like*

2.15.3.2 *How my balance for the 1st of last month would look like if I ran the report on that day (before all the backdated amendments)*

2.16 Client Interfaces

2.16.1 .Net (C#, F#)

2.16.2 JVM (Java, Scala, Kotlin, Akka, Reactive Streams)

2.16.3 JavaScript

2.16.4 RX (Observables)

2.16.5 GraphQL

2.16.6 SQL

3 Glossary

Avalanchain combines Blockchain world with Event Sourcing and as a result uses terminology from both domains. In this section word "or" in paragraphs name mean the terms are functionally equivalent.

3.1 Event or Transaction

A fact which has been accepted by the system as something that "has happened".

Events are cryptographically signed by the first Node accepted them and by the User or Stream (for projection emitted events) who submitted them.

Derived Events (projection results) are back-linked to the previous State and previous Event using Proofs and Spines.

3.2 Stream or Chain

Cryptographically signed sequence of cryptographically signed messages (Events, Transactions) with specific metadata and properties.

3.3 Projection or Smart Contract

A transformation function (*f(s, e)*) associated with each Stream. Its job is to translate a tuple of previous State and a new Event into a new State'.

$$State' = f(State, Event)$$

3.4 State or Balance

State is a first derivative of all the events in the stream till the selected point it time. Effectively it is the value (account balance, version of a document, snapshot) a Stream represents at given point of time.

3.5 Event Reference or Spine

Transaction Spine is a Merkle Tree based immutable data structure uniquely and unmodifiable identifying Transaction in a Stream. Each Transaction is uniquely identified by its Spine and Spine provides enough information to retrieve all previous events (or their proofs) in the Stream with a minimal number of network requests.

3.6 Execution Policy or Consensus

A pluggable rule associated with a Chain dictating condition required to be met in order for the Chain to confirm that the fact represented by a newly added event has happened.

Execution Policy is an indefeasible property of any Chain definition, it is hashed and cryptographically signed together with it.

3.7 Execution Group

A named set of Nodes.

3.8 Node

Server application responsible for doing validations and executions in the Avalanchain network. A Node can run a number of Execution Groups simultaneously.

3.9 Proof

Easily verifiable cryptographic evidence of some fact.

4 Applications

Please find below examples of the real world use cases and areas Avalanchain technology fits best.

4.1 Streamed Data providers
4.2 Market and Reference Data processing
4.3 Complex Event Processing
4.4 Semi-decentralized Systems
4.5 Any kinds of activity logs
4.6 Regulatory reporting
4.7 Financial derivatives
4.8 Distributed calculation results caches
4.9 Identity, KYC and Reputation Systems
4.10 Supply Chain Management
4.11 Goods tracking

5 Conclusion