Connected Chains DAGs (ccDAGs)

Panagiotis Vryonis @ WeatherXM, 2023. Draft 0.3

Abstract

While Merkle Direct Acyclic Graphs (DAGs) are not limited to a single root, having more than one is usually considered an anomaly that can be remedied by creating a new root node that links to each one of them. In this article, we introduce Connected Chains DAGs (we will call them ccDAGs) which is are a special case of a multi-root DAG.

Problem statement

At WeatherXM we have a large number of weather stations producing a flow of weather data that need to be stored on IPFS. We want

1. Data writes to happen in parallel to allow easy scaling of the system and also to avoid introducing a point of centralisation.
2. There must be a way for a third party that wants to retrieve all data from IPFS to do so.
3. There must be an efficient way for a third party that frequently traverses our IPFS data structures to retrieve deltas (new data).

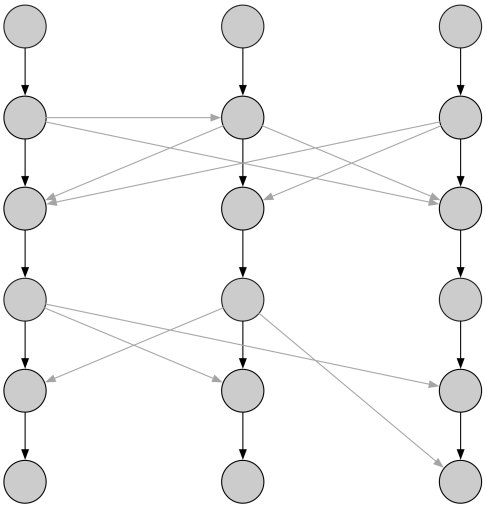
The answer to #1 is obvious: maintain X servers that will receive data from stations and store them in parallel. However this makes #2 and #3 a challenge. To solve this problem, we introduce **ccDAGs**, the *"Connected Chains DAGs"*.

ccDAGs

One can visualise ccDAGs as a number of blockchains growing in parallel, where every X blocks contain a block linking to the heads of the rest of the chains.

More specifically, a *C-N-ccDAG* (for example, a 3-5-ccDAG) has the following properties:

- Consists of *C* chains growing in parallel
- In each one of the chains, every *N* blocks, there is a block linking to the current roots of the other *C-1* chains.



While ccDAGs pose a number of challenges when the data of one block depend on the data of other blocks (which is the case of blockchains), they provide an attractive alternative to Merkle trees when storing data that have no dependencies to past states of the system.

Definitions and Properties

Block Height

We define *block height* as the length of the longest path starting from a block.
More formally:

- If a block links to no blocks, then its block height is 0.
- If block *B* links to blocks *C1, C2,..., Cn*, then *height(B) = max{ height(Ci), i=1..n } + 1*

Eventual Convergence

An important property of a C-N-ccDAG is that all *C* chains converge after *R* blocks, in the sense that they contain the same blocks.

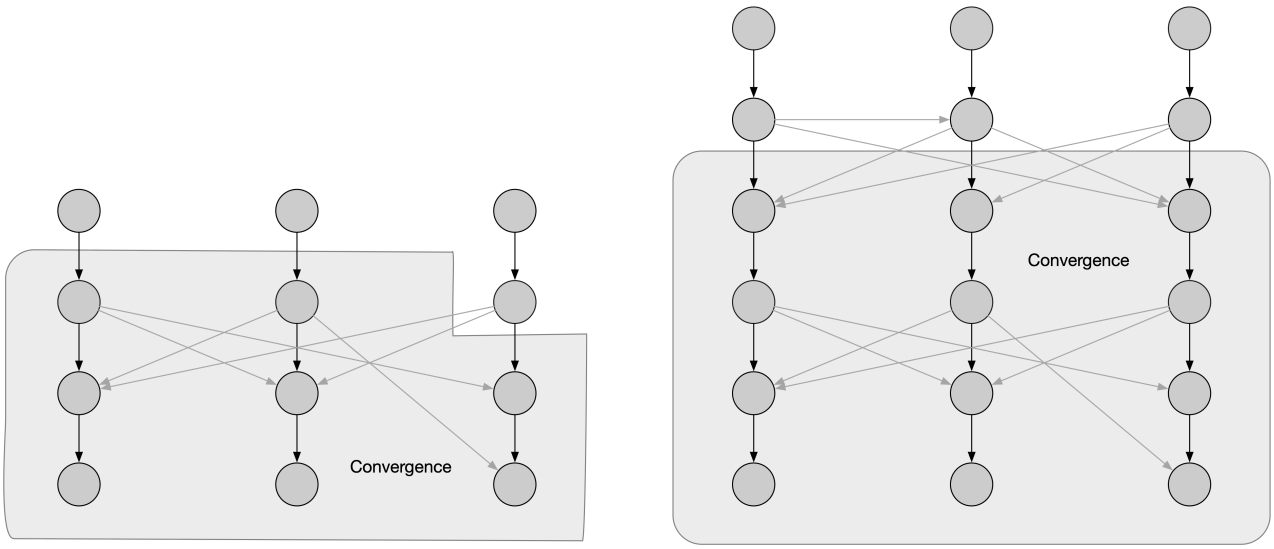
It is easy to prove that for any ccDAG

> *There is a natural number R, such that for any two roots r1, r2 and a block b linked from r1,*
> `height(b) < height(r1)−N => there is a path from r2 to b`

We define as **Convergence Rate** of a ccDAG, the minimum number R that satisfies the above statement. In a typical C-N-ccDAG, R=N, but this is not always the case as we will see later.

In other words, for chains A and B that are part of a C-N-ccDAG, every block we find after N steps while traversing chain A, can also be found while traversing chain B.

Thanks to eventual convergence, a third party can traverse a ccDAG starting from any one of the root nodes and they will get all data, except a few (up to N) blocks at the root of each chain. A new traversal after some time will fetch all the data missed during the previous one and so on.



Parallelism

Thanks to **Eventual Convergence**, all chains in a ccDAG can grow in parallel, making ccDAGs a structure that can provide high bandwidth for write/store operations.

Flexibility

C and *N* do not have to stay constant throughout a ccDAG's lifetime. It is easy to add/remove chains and even change the frequency of link blocks.

However, when *C* or *N* change, one should be aware that the *Convergence Rate* may change too.

Implementation considerations

Optimising ccDAG traversal

Storing the *block height* in each block, provides an easy way for anyone who is regularly traversing a ccDAG to know where to stop: Consecutive reversals can stop at blocks with height max*_prev_height_parsed-R*, where R is the convergence rate of the ccDAG, knowing that all new blocks (in the convergence area) have been parsed.

Also, when traversing the same ccDAG frequently, the reader should make sure that the root it starts reading from has a higher block height than the one it started from during the last traversal. If the block height is equal or lower than the last root used, a new root should be picked. This should protect from reading from a chain that has not been updated for long.

Optimnising ccDAG creation

For a C-N-ccDAG, it is possible to have an implementation where each link block links to X<C-1 chains (not all other chains), as long as the implementation ensures that eventually there is a path to *block_i_j* that belongs to chain *i* from all other chains.

Such an implementation could be as simple as using a round robin algorithm to pick the chains to link to, or more complex based on physical network topology and geographic distribution.

One must keep in mind that in the case where link blocks link to X<C-1 chains, the Convergence Rate will probably be higher than N and must be calculated based on the implementation details.

Considerations when adding a new chain

When we add a new chain to a ccDAG (going from C-N-ccDAG to D-N-ccDAG, where D>C), it is recommended that the first block of the new chain links to all the other currently active chains.

Considerations when removing a chain.

Of course a chain can not be actually removed, as its root will be eventually linked from other chains. However, we can stop updating one of the chains.

In this case, it is recommended that before shutting down the instance(s) that maintained this chain:
a. The instances stop updating its chain.
b. Makes sure that its last root block is linked to from other chains