NEAR Indexer Framework

GitHub repo https://github.com/near/nearcore/tree/master/chain/indexer You might be looking for NEAR Lake Framework NEAR Lake Framework is a lightweight alternative to NEAR Indexer Framework that is recommended for use when centralization can be tolerated.

Description

NEAR Indexer Framework is a Rust package (crate) that embedsnearcore , and abstracts away all the complexities of collecting every bit of information related to each produced block in NEAR network. The crate name isnear-indexer , and it is part of thenearcore repository .

near-indexer is a micro-framework, which provides you with a stream of blocks that are recorded on NEAR network. It is useful to handle real-time "events" on the chain.

Rationale

As scaling dApps enter NEAR's mainnet, an issue may arise: how do they quickly and efficiently access state from our deployed smart contracts, and cut out the cruft? Contracts may grow to have complex data structures and querying the network RPC may not be the optimal way to access state data. The NEAR Indexer Framework allows for streams to be captured and indexed in a customized manner. The typical use-case is for this data to make its way to a relational database. Seeing as this is custom per project, there is engineering work involved in using this framework.

Limitations

NEAR Indexer Framework embeds the full NEAR node and thus requires to sync with the peer-to-peer network and store all the network data locally thus it is subject to the storage requirements, which is hundreds of GBs on SSD if you only need to extract the data that is not older than ~2.5 days, and thousands of GBs on SSD if you want to be able to go over the whole history of the network. Also, the network sync process is known to be extremely slow (while the block production is 1 block per second, while the block sync usually reaches 2 blocks per second, which means that it is capable to catch up with the live network at a speed of 1 block per second, so if your node was offline for one hour, it will take one hour to catch up to the tip of the network that keeps getting freshly produced blocks).

NEAR Indexer Framework only exposes the blocks that were finalized. In NEAR Protocol, it takes 3 consecutive blocks to get the block finalized which means that there is at least a 3-second delay between the time when some transaction hits the network, and the time it is finalized and streamed from NEAR Indexer Framework. If we measure the delay between the moment when a transaction gets submitted from the client device to the moment Indexer Framework-based indexer would receive it, we can see the following timings:

- A serialized transaction being transferred over the Internet to NEAR node (most commonly, throughNEAR JSON RPC broadcast_tx_commit
-): around 50ms (it is not measured precisely as it is mostly network latency of TCP handshake + HTTPS handshake)
- The transaction is routed to thevalidation node
- : around 50ms (again, mostly network latency between the peer nodes)
- The transaction arrives in the mempool on the validation node and will be delayed at least until the next chunk/block is produced, so if the transaction was received right at the moment when transactions for the current block were selected, it would take 1.2 seconds on mainnet to get the next block produced
- Once the transaction is included in a block, it will produce a receipt which often will be executed in the next block (another 1.2-second delay) - learn more about the NEAR Protocol data flowhere
- Given that block finalization takes 3 blocks (1.2 seconds * 3), Indexer Framework will only get the opportunity to start collecting the information about the block where the transaction was included 3.6 seconds later, but we should also include at least a 50ms delay that is introduced by the network latency when produced blocks propagate back from the validation nodes back to the regular nodes
- Indexer Framework then collects all the bits of information for the produced block and streams it: around 50-100ms
- Custom indexer implementation receives the block and there could be additional delays down the line, but that is outside of our scope here

Ultimately, it takes at least 3.8 seconds from the moment one submits a transaction to the network, and Indexer Framework-based indexer picks it up, where the finalization time contributes the most of the delay. In real life scenario, dApps usually need to know the result of the execution, and so it will take a couple of blocks after the transaction is included to get all the receipts executed (read more about the data flowhere), so the delay between the transaction submission and the result being observed by an indexer could be 5-7 seconds.

Current Status

Indexer Framework is a tool that provides a straightforward way of getting a stream of finalized NEAR Protocol blocks as

soon as possible operating over a decentralized NEAR Protocol peer-to-peer network.

However, in our experiments with the Indexer ecosystem, we realized that we need a lightweight foundation to build micro-indexers instead of maintaining a full nearcore node. We considered various solutions to deliver events (Kafka, RabbitMQ, etc), but ultimately we decided to dump all the blocks as is to AWS S3 bucket. This is where NEAR Lake ecosystem was born, learn more about it here .

These days, we use NEAR Indexer Framework to implement NEAR Lake Indexer and from there we build micro-indexers based on NEAR Lake Framework . Said that, Indexer Framework plays a crucial role in the ecosystem even though most of the indexers these days are implemented without using it directly.

Applications

See the example for further technical details.

- near-examples/indexer-tx-watcher-example
- NEAR Indexer example that watches for transaction for specified accounts/contracts

NEAR Indexer Framework usage The most famous project build on top of NEAR Indexer Framework is NEAR Indexer for Explorer Edit this page Last updated on Oct 19, 2022 by Damián Parrino Was this page helpful? Yes No

Previous Intro to Indexers Next Lake Framework