Covalent

The Block Specimen

Standardizing Data Provenance for Web3

A novel solution to an age-old problem

Last updated: 26 April 2022

Background

Since the inception of the Bitcoin blockchain, the blockchain space has been laser-focused on write scalability. That is, actually writing on the blockchain (confirming a transaction) and doing so efficiently. This is why we have multiple layer 1 projects fixated on achieving higher transactions per second among other core metrics. As a result, the space has certainly become more accommodating, leading to increased adoption as powerful and scalable applications are being developed atop blockchains.

However, an overlooked aspect of blockchains as a technology platform is read scalability. This is different to write as the focus with read is on extracting and reading the data on the blockchain, whether that be Ethereum, Avalanche, or Solana. By ignoring such and not innovating in this field, blockchain adoption will be negatively impacted. Our view is that fast, cheap, and accessible read capabilities will lead to more diverse and better-adapted blockchain technologies.

Covalent's mission is to bring transparency to blockchain-based assets. Since day 1, Covalent has been focused on the read scalability aspect of blockchains. The Block Specimen is a novel invention that addresses the lack of scaling technologies to read back from blockchains with the right data provenance model.

Reading from the JSON RPC Layer

The crux of the issue of reading from blockchains can be found when examining the JSON RPC layer; a uniform set of methods that applications can call using an API to pull data from, in this case, Ethereum nodes. A number of issues present themselves when doing so.

1) **Slow** - One needs to make a series of individual data queries to extract the block and its constituent elements like transactions and receipts. For example, an individual who wishes to query a group of users' ERC-20 token balances for a given token at a given block has to make multiple JSON RPC `eth_call` to the given token contract for each of the users individually. Here each call is a separate thread that returns a balance value that has to be recorded and stored. A batched `eth_call` can also be performed by averaging the same overhead as the last case. However, no special cases exist where this can be concurrently done for all the addresses one is interested in a single thread, for the token at the block height ("multi-eth-call") with a 10x lower overhead in terms of IOPS. Frankly, this does not scale.

2) **Not Multiversion** - In database management systems, multiversion concurrency control methods are employed. This ensures point-in-time consistent views if multiple parties are viewing or querying the database. Such methods do not exist in Web3.

While the data queries are in flight on the JSON RPC layer, the network itself is evolving. Therefore, if multiple independent actors were to extract a block and its children, there's no mathematical guarantee that the extraction process is precisely the same. As a result, achieving consensus for the extraction process is impossible because of the difference in proofs. Without consensus through the proofs, downstream use-cases of blockchain data are left unsecured. The opposite is the case when you examine the write aspect of blockchains where cryptography, signatures, and economic incentives ensure security.

3) **Expensive** - To access historical data at any point in time, you need to run your blockchain clients in a mode known as "full archive nodes" which requires specialized and expensive hardware to scale.

4) **The Purge** - For Ethereum specifically, Vitalik recently outlined an updated roadmap for its development which included a phase titled 'The Purge'. Once this phase is implemented, clients will no longer store historical data older than a year. Hence, alternatives will be needed to access Ethereum's full historical state.

> "Clients will stop storing history that's more than 1 year old and we shift the responsibility of storing and retrieving that history to other protocols." - Vitalik Buterin

There are solutions beyond calling the JSON RPC layer to extract data from blockchains. Other APIs beyond Covalent's focus primarily on DApps rather than presenting a canonical representation of a blockchains historical state. However, with these models, scalability issues once again arise when you consider the multitude of DApps that are currently scaling, in testing and development, and being incubated not only on Ethereum but other networks as well.

Introducing the Block Specimen

The Block Specimen is a technical specification designed to eradicate the aforementioned read scalability issues. The specification was defined and revised over multiple years of building with blockchain technology internally at Covalent.

The Block Specimen is a bulk export method that can be implemented on existing blockchain clients including Geth and Erigon. These patches are essentially an extract-and-normalize worker that create Block Specimens, which together form a canonical representation of a blockchains full historical state. With this, the Covalent Network's complete storage and efficient retrieval of historical data will be an obvious choice for developers who need it in their applications.

The Block Specimen is also a cryptographically secure representation of a block and its constituent elements. For every Block Specimen created, a block-specimen production-proof transaction is published to the Covalent chain. Given the use of proofs, any deviations in the data either accidentally or malicious will have mismatching proofs.

Finally, the Block Specimen is standalone. Covalent has separated out the data storage layer from the block execution and distributed consensus functionality. As a result, anyone can run full tracing on the block specimen and accurately recreate the blockchain without access to a blockchain client software.

Any data source that respects the BSP specification can be seamlessly used by the Covalent API.

The Covalent Network & Block Specimen

The Covalent Network moves the platform beyond a centralized blockchain data API, enabling token holders and developers to engage with the network in new ways. All while eradicating the limitations of centralization.

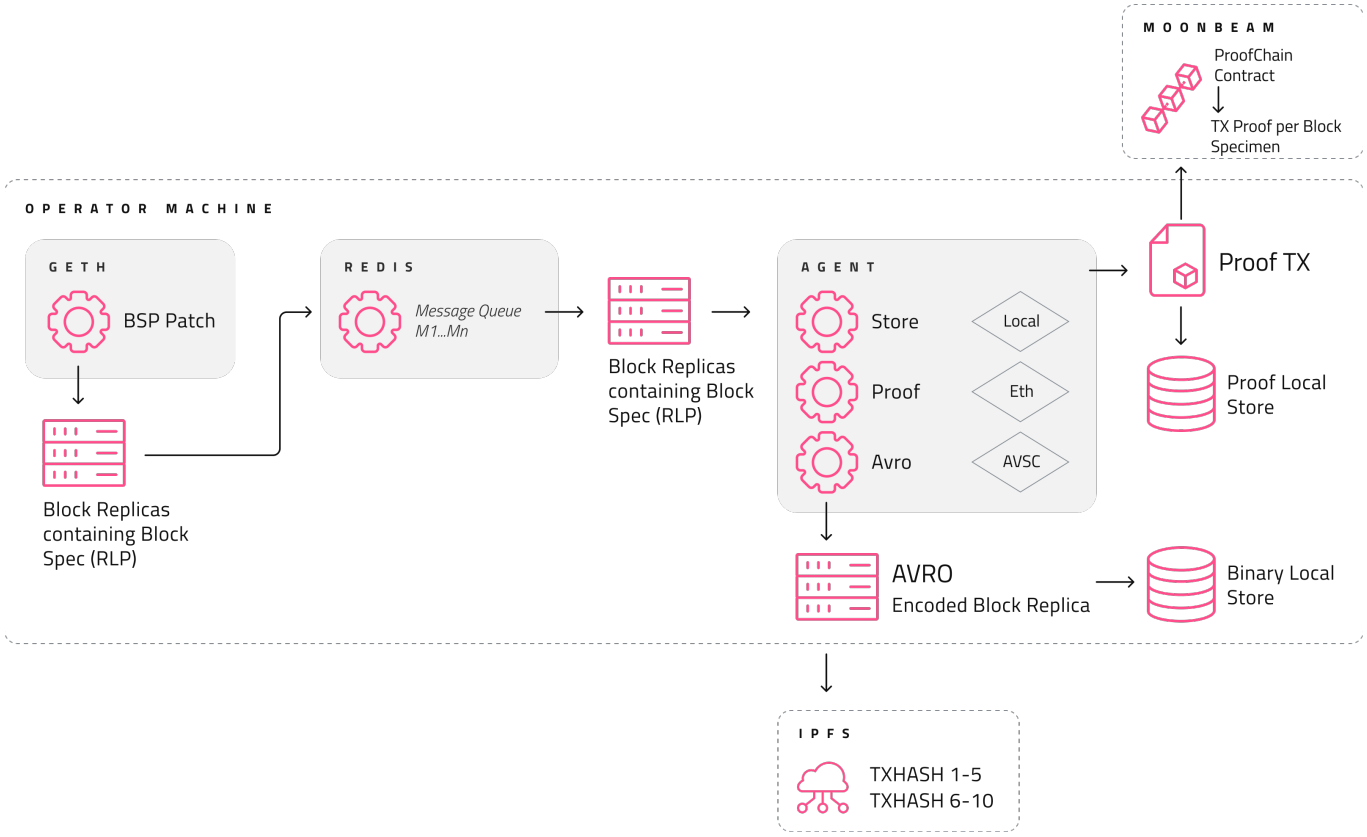
While Block Specimens are being created internally at Covalent for each respective blockchain indexed, the Covalent Network shifts this responsibility to operators (a node on the network performing a role). Any operator can opt to act as a Block Specimen Producer (BSP). Similar to the specifications discussed previously, this will involve running a full or archive node (Ethereum mainnet initially) with a Covalent extract-and-normalize worker patch sitting on top.

It is the responsibility of the BSPs to consume blocks from external blockchains and publish both the Block Specimen along with a production proof to the Covalent virtual chain.

BSPs play a pivotal role in the network given that the data in the Block Specimens will feed the entire network with the data needed to answer user queries.

The Inner Workings

A lot happens from the client node to the block-specimen production-proof transaction being uploaded to the Covalent virtual chain. Let's break down each step, assuming that the operator is using go-ethereum as the underlying EVM-based blockchain node that houses the BSP modifications.



Once go-ethereum is run with the correct configurations, Block Specimens are created as the node catches up historically from block 0 or even if it just starts syncing from the block number where it left last off after the Geth process was stopped (with or without the patch). The operators have the ability to decide from which block they would like to start producing specimens.

Once the producer is correctly started, Block Specimens are created for every block that is executed and consequently synced with Geth. These specimens are then sent off to a Redis streaming service (that the validators need to run on the same system as the Geth process) at the particular topic key where the Agent can listen for receiving these messages.

Within the Agent, this topic key can be set along with a number of other configurations. These include how to pack incoming streamed messages so that they can be stored in any storage device. The "how" refers to the given size (contents), encoding used (AVRO), and location (GCP, AWS, or IPFS for example) of the block specimens for storage.

As each message is streamed to the Agent, the Agent firstly makes sure that the message follows the format it expects to receive from a BSP. If for example, it receives messages from another topic that does not conform to the expected schema, the application stops any further processing. Secondly, once the messages are verified to be in the correct format matching the Block Specimen schema, it proceeds to pack multiple stream messages into an array of block specimens within a single binary encoding using AVRO (a .avsc schema). The size of packing is specified at the initialization of the Agent.

The AVRO encoding for packing multiple Block Specimens is basically the process of segmenting the messages into batches and processing them for proving and storing them in those exact batches. Thirdly, the Agent keeps track of every stream message that's been verified and segmented into a batch and then encodes (compressing) them using this codec converting them into a single binary array of multiple specimens. Finally, after evaluating an entire segment the Agent has the capability to talk to a proof-chain contract deployed on the earlier specified blockchain node. It takes the binary specimen object that's ready for storage, does a SHA256 sum over its contents, and calls a payable function on the BSP-Proof-chain contract that is pre-deployed to the corresponding node that it is talking to.

The Agent effectively makes a provable (or dis-provable) commit regarding the work it's done so far on the Block Specimen segment object itself. Once this proof-of-creation transaction is mined, the Agent proceeds to upload the object it's already made the commit for to the specified object storage service location provided in the initialization configuration.

Between reading each stream message and storing a Block Specimen object representing an evaluation (validating and proving) process, the Agent performs a bunch of processing expensive operations including a single gas expending operation of writing the proof-of-creation of Block Specimens to an(y) EVM based blockchain. The chain that the agent writes to is also specified during the initialization of the Agent.

The process of evaluating a message, segmenting multiple messages into a single AVRO encoding, making a statement regarding the work done by making a transaction on-chain, and finally uploading the segment is an entirely atomic process. All of the above will either go through or none of them will. Anyone section failing in this chain at any point leads to the failure of the entire process within the Agent. In case of such failure, the exact reason is made as explicit as possible in the logs from where corrections can be applied during initialization. The messages in the Redis stream are persisted in the queue, in case a failure in the processing chain happens thereby making sure that if at all - only fully evaluated, encoded, and proved (processed) segments get uploaded to any storage service.

How to Get Involved

Covalent is a critical infrastructure component of the partner blockchains we support. We work closely with the technical and business teams at these blockchains to ideate, plan and execute a turn-key solution for developers building on top of their blockchains.

To learn more about the Covalent indexing process and implementing the Block Specimen standard to better serve your developers, reach out to a core team member on the Covalent Discord.

Furthermore, as Covalent progressively decentralizes the network, a number of operators will be onboarded. The initial whitelist, which is made of 10-12 professional infrastructure providers, will eventually be expanded as the network is developed. If you are interested in running a node on the Covalent Network, keep an eye on the Covalent social channels.

About Covalent

Covalent provides the industry-leading Unified API bringing visibility to billions of Web3 data points. Developers use Covalent to build exciting multi-chain applications like crypto wallets, NFT galleries, and investor dashboard tools utilizing data from 32+ blockchains. Covalent is trusted by a community of 27,000+ developers and powers data for 1000+ applications including 0x, Zerion, Rainbow Wallet, Rotki, Bitski and many others.

Covalent

One unified API. One billion possibilities.

@Covalent_HQ covalenthq.com/telegram covalenthq.com/discord covalenthq.com