Improving and Scaling Starknet RPC: A Research Perspective

Abstract:

Starknet, positioned as a decentralized Layer 2 solution within the Ethereum ecosystem, emerges as a pivotal framework delivering pronounced scalability and cost-efficiency for decentralized applications (dApps). At the core of this infrastructure, the Remote Procedure Call (RPC) mechanism assumes a critical role, orchestrating communication across diverse components inherent to the StarkNet ecosystem. Notwithstanding its fundamental role, the extant StarkNet RPC infrastructure confronts inherent limitations, particularly in terms of operational efficiency and scalability. This scientific endeavor to proffer innovative enhancements geared towards amplifying the performance and scalability metrics of StarkNet RPC, thereby fortifying its capacity to accommodate the escalating demands imposed by the burgeoning landscape of dApps and their associated user base. Through rigorous investigation and empirical insights, this research aspires to contribute to the ongoing discourse surrounding decentralized application architectures and their sustainable scalability solutions within the Ethereum framework.

Introduction:

Starknet, with its intricate architecture, stands at the forefront of decentralized applications. The seamless communication between its components, facilitated by RPC, is fundamental to its functionality. This paper sets out to elevate the efficiency, scalability, and overall performance of Starknet. By delving into the intricacies of RPC and its impact, our goal is to provide insights that pave the way for a more streamlined and responsive decentralized infrastructure.

In this paper, we will discuss the current state of RPCs and nodes. We will go through each node providers in the ecosystem today and will test it out- and find how reliable or fast it is.

Nodes in Starknet:

Nodes are essential components of the Starknet network, playing a crucial role in its operation and security. They are responsible for verifying transactions, maintaining the network's state, and providing access to Starknet data. Just like Ethereum, Starknet has different types of nodes, each with its own specific function.

In essence, blockchain nodes are divided into two main types—full and light. The key difference? One contains a copy of the blockchain’s history (full), while the other downloads block headers only to save the hard drive space for its users. There is more to nodes than just that, however, so let’s dig further down into the specifics.

Light nodes

As just mentioned, light nodes do not carry the entire weight of the blockchain but rather only the headers of blocks to allow portable interaction with Web3. This type of node is the most commonly seen throughout the space, as it is usually found on the user side, such as a wallet interface.

Light nodes, or Simple Payment Verification (SPV) nodes as they are technically referred to, rely on full nodes to provide them with the data they need to perform their operations. And considering they don’t carry a copy of the blockchain themselves, they can only query the status of the last block and broadcast transactions to the network for processing.

That being said, it becomes clear what makes light nodes such a portable implementation that requires a significantly lower number of resources than a full node would. Even so, this convenience does come at the cost of security.

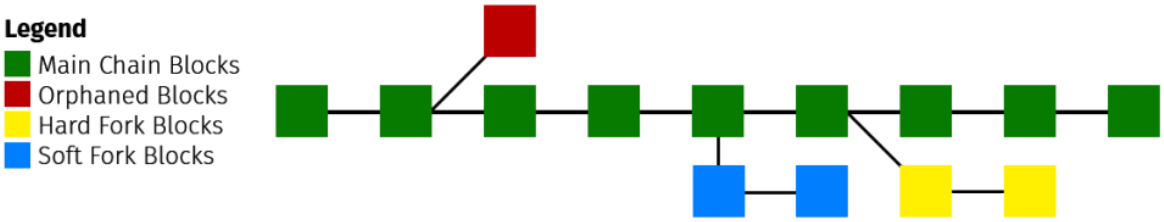
Full nodes

On the other hand, full nodes are the bread and butter of a blockchain network. They act as a server within the distributed network and are mainly responsible for the verification of transactions. This type of node carries the full weight of a blockchain network by holding a copy of it on its local storage.

Full nodes are also responsible for the consensus mechanism, which ensures only valid transactions are propagated on the network. In doing so, these nodes vote on proposals and make decisions for the future of a network, should 51% of them agree on a certain outcome.

If a particular outcome fails to hit the 51% voting mark, it is automatically rejected by the network, in turn creating what is called an “orphaned block.” In some cases, especially

when it comes to large networks with many full nodes involved, this can result in a network split called a hard fork.



Block propagation and chain splits; Source: ExtremeTech

The result of this is two separate networks supported by the full nodes that backed their proposals. One of the most well-known occurrences of this was the Bitcoin Cash Fork, which caused the creation of two new networks—ABC and SV.

Archive nodes and pruned full nodes

Full nodes come in two main forms—archive nodes and pruned full nodes. Both of these nodes synchronize blocks from the beginning of the blockchain's history. The main difference between them, however, is that pruned full nodes have a certain limit, after which old blocks begin being replaced.

To put this into a better perspective, setting a size limit of 1GB for a pruned full node would result in a blockchain history of no more than 1GB being saved to local storage. If new blocks being propagated increase the size beyond that 1GB limit, the node begins to delete the history of the first blocks, in order to make room for the new ones. And should you wish to get the state of a block that was already replaced, the node needs to go through the entire chain to validate those blocks once again.

On the other hand, archive nodes can be quite heavy, when it comes to storage requirements, as they contain the entire chain state history that can be queried.

Currently there are 3 nodes:

Provider name	Description	More information
Juno	A Starknet full-node written in go-lang by Nethermind	github.com/NethermindEth/juno

Papyrus	A Starknet full-node written in Rust by StarkWare	github.com/starkware-libs/papyrus
Pathfinder	A Starknet full-node written in Rust by Equilibrium	github.com/eqlabs/pathfinder

When it comes to node providers, there are a few:

Provider name	**Provider site**	**Open API endpoint, where relevant**
Alchemy	www.alchemy.com/starknet	
Blast API	blastapi.io/public-api/starknet	https://blastapi.io/public-api/starknet
Chainbase	chainbase.com/chainNetwork/Starknet	
Chainstack	chainstack.com/build-better-with-starknet	
Infura	www.infura.io/networks/ethereum/starknet	
Lava Protocol	www.lavanet.xyz	https://www.lavanet.xyz/get-started/starknet
Nethermind	starknetrpc.nethermind.io	http://starknetrpc.nethermind.io

The Starknet network is still under development, and the role of nodes is likely to evolve in the future. However, nodes will continue to be essential components of the network, and they will play a critical role in ensuring its security and scalability.

Comparing nodes

	Pros	Cons
Light nodes	• Portable • Resource-efficient • User-friendly	• Do not validate the network • Do not propagate blocks • Do not maintain consensus • Less secure
Full nodes	• Validate the network • Propagate blocks • Maintain consensus • More secure	• Resource-heavy • Harder to maintain • Less user-friendly
Pruned	Flexible storage	Need to revalidate old blocks
Archive	Carry full history	Resource and storage heavy
Mining	• Easily trackable involvement • Can pool with others to increase reward rate	• High and wasteful energy consumption • High equipment cost and barrier to entry
Staking	• Low barrier to entry • Low energy consumption	• Reward system based on luck • Low transparency in staking pools
Masternodes	• Balanced network benefits and rewards • Lower maintenance costs	• High initial investment • Difficult setup process

The challenges of blockchain node setup

There are several challenges associated with the deployment of a blockchain network and its associated nodes. As mentioned, these issues are typically characteristic of

public ledgers such as Ethereum and Bitcoin. However, mega-consortia blockchains handling massive amounts of data succumb to similar pressures.

Slow synchronization

When synchronizing hundreds of GB of blockchain data, all while trying to keep up with new network activity, it’s going to take awhile. Typically this translates to hours or even days – as they say time is money.

Data corruption and network instability

As all the data items stored in blocks are downloaded and linked together, corruption in one block can corrupt the entire blockchain. This scenario may occur as a result of a poor network connection amongst a myriad of other reasons. As a result, you could wait days for node synchronization only to find all your efforts have been a waste.

The cost of hardware and network traffic

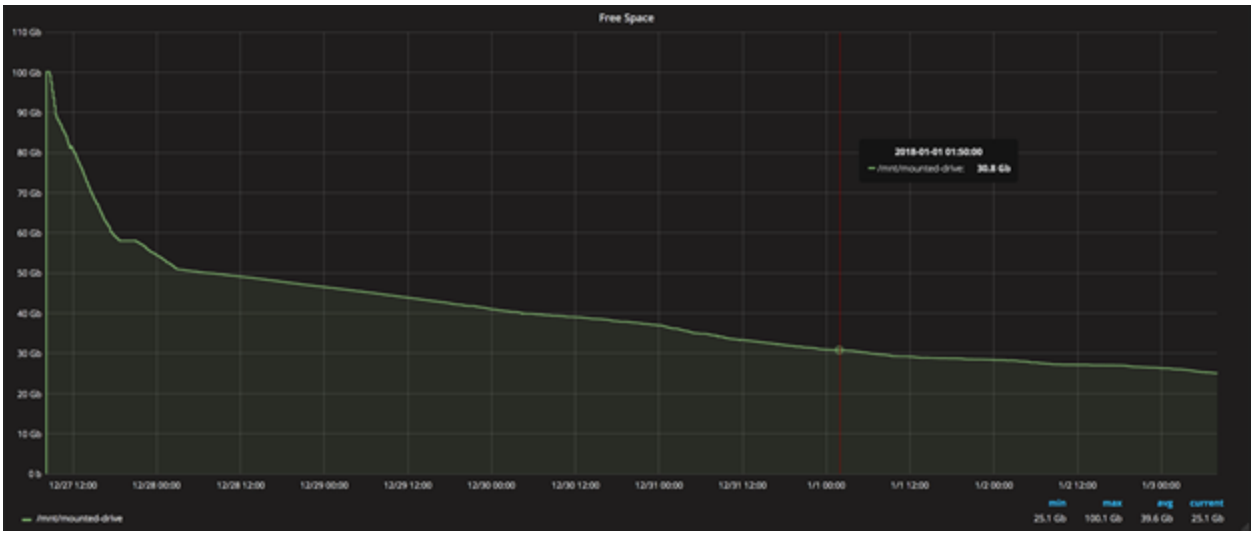
Depending on the cost of network traffic for a particular distributed ledger, there could be significant costs associated with initial node synchronization. There’s no guarantee of how much this may cost – the amount of data is immense, and the volume increases continuously.

The synchronization process is also extremely disk intensive, requiring a lot of writes. This can put excess stress on your valuable hardware while consuming substantial amounts of disk space. Ultimately, this requires resynchronization to clear disk space or the constant expansion of storage capacity.

This table shows the evolution of the free space on the disk at specific points:

```
| DATE          | SIZE(GB)    | DIFF (GB)   |
|---------------|-------------|-------------|
| 27/12/17 8:30 | 93.23209763 |             |
| 28/12/17 8:30 | 46.39630127 | 46.83579636 |
| 29/12/17 8:30 | 41.57238007 | 4.823921204 |
| 30/12/17 8:30 | 36.69866943 | 4.873710632 |
| 31/12/17 8:30 | 31.91919327 | 4.779476166 |
| 1/1/18 8:30   | 27.54598618 | 4.373207092 |
| 2/1/18 8:30   | 25.38702011 | 2.158966064 |
| 3/1/18 8:30   | 23.32711411 | 2.059906006 |
```

and this is the graph of the evolution:



Bolt technology by Chainstack

To speed up the syncing process, Chainstack created a regular snapshots of the ledgers they support. When a node creation request is initiated, the platform restores the ledger on the node from the latest snapshot. As a result, there is no need to sync all the way through the blockchain. Instead, only the data added since the latest ledger snapshot needs to be downloaded.

This functionality ensures you have a fully synced node in just minutes – not hours or days. Bolt is a simple and robust technology that helps you spawn any number of nodes quickly and without the need for technical know-how.

As mentioned, the cost of network traffic can be significant when syncing a node due to the immense volume of data being downloaded. And although a fully synced node still requires the appropriate hardware to perform optimally with Bolt, Chainstack takes care of this for you. As a platform user, you'll never worry about hardware costs or the high disk impact of syncing activities. You avoid investing in expensive hardware and potentially costly network traffic – that's a win.

Endpoints

StarkNet attempts to solve the scalability issue while preserving the composability and security
of Layer 1 Ethereum using a permissionless decentralized zero-knowledge rollup, leading to a
reduction of transaction costs by 100 times.

In this paper, we explore a simple protocol that can be built on StarkNet: a StarkNet smart
contract containing a machine learning dataset that is decentralized and collaborative. We will
use a labeled dataset of housing prices for supervised learning as an example. The contract
enables anyone who wishes to contribute to building a dataset to append data, offering a
continuous update to the machine learning model when needed.

Common Starknet endpoints provided by RPCs

1. Getting Contract and Block Information
2. Transaction Management
3. Querying Transactions
4. Account Information
5. Chain Information
6. Event and State Data
7. Fee Estimation
8. NFT API

Let’s start with a single or common endpoint call which could give an answer.

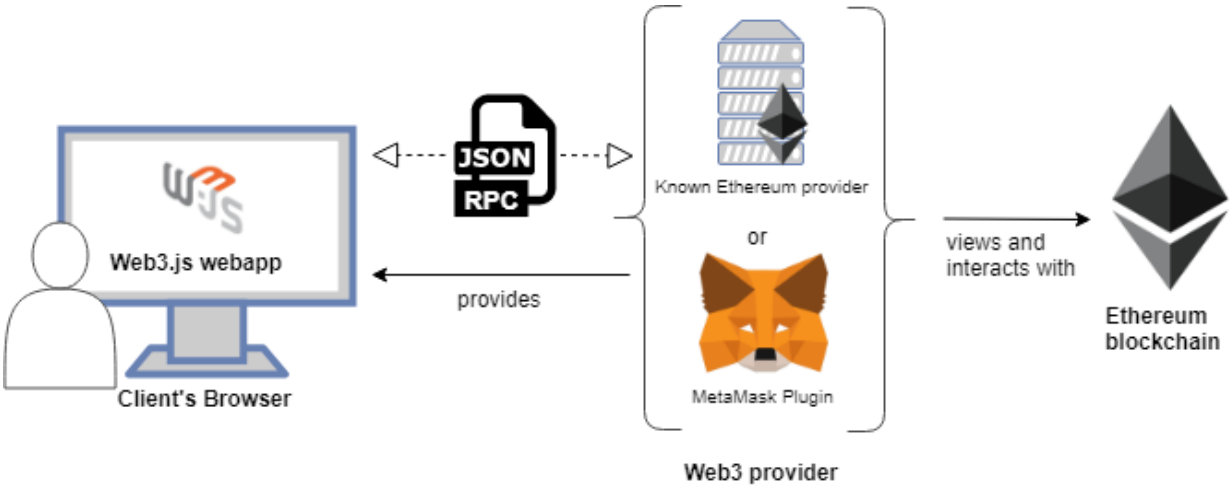
```
const options = {
  method: 'POST',
  headers: {accept: 'application/json', 'content-type': 'application/json'},
  body: JSON.stringify({id: 1, jsonrpc: '2.0', method: 'starknet_estimateFee'})
};

fetch('rpc-url', options)
  .then(response => response.json())
  .then(response => console.log(response))
  .catch(err => console.error(err));
```

The 3 basic options configurations are as follows:

- `method`: Specifies the HTTP method for the request, in this case, 'POST'.
- `headers`: Defines the headers for the request. It includes 'accept' and 'content-type' headers, both set to 'application/json'.
- `body`: Contains the payload of the request. It is a JSON string representing an object with properties such as 'id', 'jsonrpc', and 'method'. This adheres to the JSON-RPC 2.0 specification, where 'starknet_estimateFee' is the method to be invoked, and 'id' serves as a unique identifier for request-response correlation.

In summary, this code initiates an RPC to estimate fees on the Starknet Mainnet by sending a structured JSON payload to the specified URL. The subsequent processing of the response and error handling enhances the robustness of the client-side interaction with the StarkNet API.



Challenges and Limitations:

The legacy world of Web2 (think Facebook, Google, Amazon, Microsoft, Apple) is made of centralized, controlled, commercial, proprietary servers. The vast majority of the world's information is kept in centralized servers, which is kind of scary when you sit down and consider the implications. Even Web3 firms suffer heavily when they use a centralized server model:

- Metamask restricted user access due to its reliance on an Infura API.
- Binance and other exchanges halted withdrawals when an Infura API went down.

- Twitter's NFT features suffered due to its reliance on OpenSea, which relies on Alchemy servers.

In other words, a lot comes back to the simple fact of server ownership; where are you storing your information, and how safe is it? Time and time again, reliance on centralized servers leads to multiple inefficiencies.

A Web3 company that claims to be fully decentralized isn't if they use a centralized node provider like Infura. Infura does make building on Ethereum a lot easier, but the price of centralization is too high.

If all the information runs through a single centralized node provider, then it does not matter how many thousands of "decentralized" Web3 firms are changing the world. The fact is that the power lies with the person/firm who holds the data.

Conclusion:

In conclusion, this research provides a multifaceted exploration of Starknet RPC, focusing on scalability improvements, node provider comparisons, challenges in node setup, and innovative solutions like Bolt technology. The findings contribute to the ongoing evolution of Starknet, offering insights for developers, stakeholders, and researchers in the decentralized application space. The importance of decentralization, data ownership, and resilient node infrastructure is underscored as fundamental principles for the sustainable growth of decentralized ecosystems.

References:

https://docs.starknet.io/documentation/starknet_versions/juno_versions/

https://www.starknet.io/en/posts/developers/papyrus-an-opensource-starknet-full-node

https://docs.starknet.io/documentation/starknet_versions/pathfinder_versions/

https://ethereum.stackexchange.com/questions/34979/geth-disk-and-memory-performance-analysis

https://chainstack.com/introducing-bolt-the-chainstack-technology-made-for-simple-node-synchronization/#1-the-challenges-of-blockchain-node-setup

https://docs.alchemy.com/reference/starknet-api-endpoints#starknet-api-endpoints-by-use-case

https://github.com/ethereumbook/ethereumbook/issues/376

https://hackernoon.com/apis-rpcs-and-node-infrastructure-the-backbone-of-web3-development