Optimizing the computation power and cost of hosting blockchain nodes

Web3 application workload monitoring:

In the basic architecture of a web3 application, we have a front-end where the end-user interacts with the application. Then, we have a backend server and a blockchain network that manages user requests. Different web3 applications can vary significantly in terms of workload depending on factors such as the type of application, the size of the user base, the frequency of user interactions, the complexity of smart contract logic, and the volume of data stored on the blockchain. For example:

- A decentralized finance (DeFi) application that allows users to trade cryptocurrencies, lend or borrow assets, or earn interest on their holdings may experience high volumes of transactions, as well as high concurrency and throughput demands. These applications often require significant computational resources to perform complex calculations and execute smart contract logic securely and efficiently.
- On the other hand, a decentralized social media platform that allows users to share content and engage with one another may not require as much computational power, but may still experience high volumes of user interactions such as likes, comments, and shares.

Moreover, the number and complexity of queries can vary greatly depending on the application. Some applications may require frequent real-time updates to data, while others may primarily rely on historical data. Therefore, the choice of hosting a blockchain node needs to consider these requirements, as the application grows, the users' interaction query and the size of the user base might increase and vice versa. Additionally, provisioning cloud resources to a blockchain node requires proper analysis of the type of web3 application. Choosing the right computation power for a blockchain node will result in better utilization of cloud resources to avoid overprovisioning.

Furthermore, as application workloads change, monitoring application characteristics over time to provide dynamic allocation of resources can help optimize computation resources and associated costs in hosting a blockchain node. If the workload is less for a certain time, some idle nodes can be turned off to save costs.

Minimize the interaction with blockchain node as possible:

To optimize the computational efficiency of hosting a blockchain node, we need to think about how we can minimize interaction with the node. How can we control the requests or offload them to fulfill users' requests without directly requesting resources from the blockchain node? We can solve this problem by using caching or databases that convert blockchain data to high-performance query databases, such as **Astra block**. Here, blockchain data is stored in the database for better read performance. This leads to less load on the blockchain node, increasing computational efficiency and cost savings. However, it comes at the cost of managing a database. Additionally, memory caching can also be used to offload blockchain queries by running a caching layer in a Docker container, such as **Redis** and **PostgreSQL**. This is suitable for applications where some blockchain data does not change frequently (e.g., decentralized identifiers and verifiable claims for users can be stored, so we don't call functions on the blockchain node every time we want to see them) but is required frequently on the client-side. In such cases, information can be kept in an in-memory database.

Moreover, minimizing interaction with the blockchain node helps offload tasks to databases, which increases the performance of web3 applications. If the blockchain node is running on a resource-based pricing model, this can reduce costs and make the blockchain node computationally efficient since the user's applications do not directly query or interact with the blockchain nodes; rather, they receive the same data from up-to-date databases.

Trade-off between scalability and cost-effectiveness:

There is a trade-off between the scalability of a web3 application from a user's perspective (we are not referring to the scalability of core blockchain consensus here; we will discuss it later in this document) and the cost-effectiveness of hosting the blockchain node. For example, if the number of users of the application is increasing as the web3 application becomes more popular, we need to provision more resources to handle the application's workload. However, this increases the cost of providing more resources to the web3 application. Therefore, we need to actively monitor the workload of the web3 application to address this issue. Additionally, we need to analyze the specific use case and deployment scenarios for the web3 application. We can think of it as a dynamic workload scenario where the blockchain nodes will not always be serving user requests. If we have a proper analysis of the application workload, we can give sufficient VM resources to the blockchain node. One solution could be to allow developers to run multiple blockchain nodes rather than a single blockchain node with high computational power. As the workload is dynamic, and there can be times when the blockchain nodes are not receiving many requests, we can turn the idle blockchain nodes off. This brings two benefits: firstly, we can avoid overprovisioning of resources, and secondly, we can dynamically optimize the computation power of blockchain nodes. We can also use auto-scaling policies that automatically adjust the number of blockchain nodes. However, we need to understand that if we want to optimize the cost, the web3 application might experience some performance degradation.

Optimizing the scalability of a blockchain network

In order to understand the scalability of the core blockchain consensus, we first need to understand the difference between private and public blockchains.

In a public blockchain, the block time is predefined. For example, the block time for the Ethereum blockchain is around 12 seconds. This means that the block will be mined every 12 seconds. No matter how many blockchain nodes we introduce in the Ethereum network, we cannot increase the transactions per second or improve the block time. We also cannot change the inherent consensus algorithm to increase scalability. Imagine a public blockchain node is deployed in the cloud VM. We cannot control the write requests, as the transactions are processed according to the rules of the public blockchain. What we can do is scale the user side request. Similarly, if we are running validator nodes of the public blockchain, these nodes need to be run 24/7 to validate the blocks and transactions because the blockchain network is active 24/7 and does not stop making blocks.

In contrast, in private blockchains, we can deploy nodes and define the type of consensus mechanism, such as fast or slow, to control the scalability of the blockchain nodes. If we want to process transactions faster, we can devise our own consensus algorithm or create multiple blockchain nodes based on the web3 applications' needs.

In the subsection “Minimize the interaction with the blockchain node as much as possible,” we discussed how we can scale web3 applications from the user’s requests point of view. We can have caching or databases based on Ethereum blockchain or other blockchains to ensure the application scales as the number of users or requests increases. However, when we talk about scaling the core blockchain algorithm, we have different techniques that can be used, such as sharding, layer 2 solutions, and side chains.

Sharding is a technique that allows a blockchain network to scale by partitioning the network into smaller pieces, or shards, and processing transactions in parallel on each shard. This can greatly increase the capacity of the network and reduce transaction times.

Considering the case of a private blockchain, where the number of nodes and participants is limited and known, sharding can be a viable way to improve performance and scalability. We can define multiple blockchain nodes in the cloud VMs and by dividing these nodes into smaller networks, we can create more manageable shards. Each shard can handle a subset of the network's transactions, reducing the load on any individual node or shard. This can help the private blockchain to scale more effectively and handle more transactions per second. We can also introduce the concept of dynamic sharding, where we can monitor the transactions being generated by the web3 application and change the number of blockchain nodes and shards to ensure computational optimization.

Consensus algorithm optimization is another way to increase the transaction throughput, leading to improvements in the speed and scalability of the network. In private blockchains, we can tune the consensus algorithms to achieve this. For example, by reducing the block time or block size, we can process transactions faster in the blockchain network.

Finally, **cost-effective scalability** is possible by carefully considering the application dynamics and workload, and scaling up and down the resources to fulfill the application requirements.