Efficiently fetch data from a complex data structure in a solidity smart contract on the Ethereum blockchain.

**Arno Verbeke**

## Abstract

Fetching data from a complex data structure like a mapping or a list that is stored on the Ethereum blockchain can be challenging. For example it is impossible to directly fetch from a smart contract all the keys a mapping has without storing extra data (our keys) on the blockchain. By storing this extra data an overhead of data is created which leads to high gas fees. To avoid this there are multiple techniques that don’t require us storing an overhead of data. The best technique is a decentralized network of multiple computers storing the transactions that modify this complex data structure separately from the Ethereum network. These transactions can then be queried based on indexes configured. Further on in the examples the stored transactions are used to retrieve the keys a mapping has.

## Introduction

The rise of popularity of the Ethereum blockchain the last decade has let to a lot of development and improvements of the blockchain. One of these improvements has let us to solving a common problem on the Ethereum blockchain. It is sometimes difficult to fetch data from a complex data structure, for example the keys of a mapping. The reason is that the blockchain doesn’t store all the data, for example the keys of a mapping.

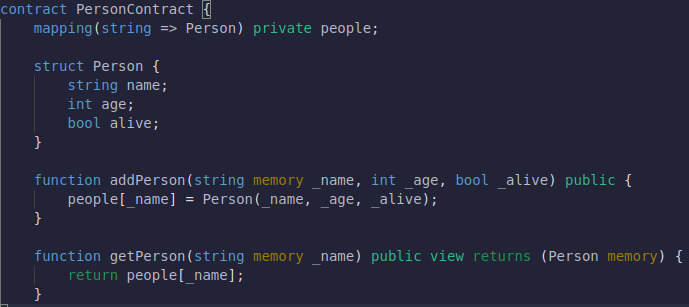
* 1. Further on we go over the possible solutions, the last solution could also be called a major improvement of the Ethereum blockchain usability. In each solution a mapping is used for our complex data structure.

## Solutions

### Simple solution

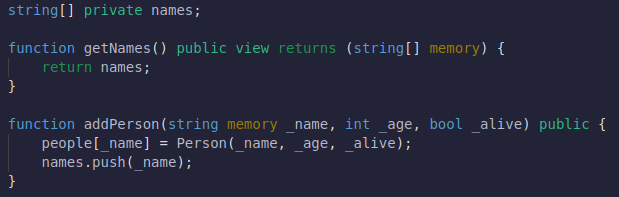
For our simple solution the data that needs to be more accessible is stored into another data structure. For this example the storing of keys into an array.

First a simple mapping is created in solidity. A function is added to update and remove from this mapping. To fetch the data from the mapping another function is added.

Two persons are created in the tests: Person1 (name: Arno, Age: 23, alive: true), Person2 (name: Jan, Age: 23, alive: true). The transaction gas fees are as follows after testing:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Creation Of Contract | Adding Person | Adding Second Person |
| Gas Fees | 454867 gas | 90510 gas | 90498 gas |

When creating mappings in solidity the keys are not stored on the blockchain. Therefore the keys of the mapping is stored into an array. The array will be updated every time a person is added. To be able to fetch all the keys a function is created to return the created array. The following has been changed:

Two persons are created as stated before. The transaction gas fees are as follows after testing:

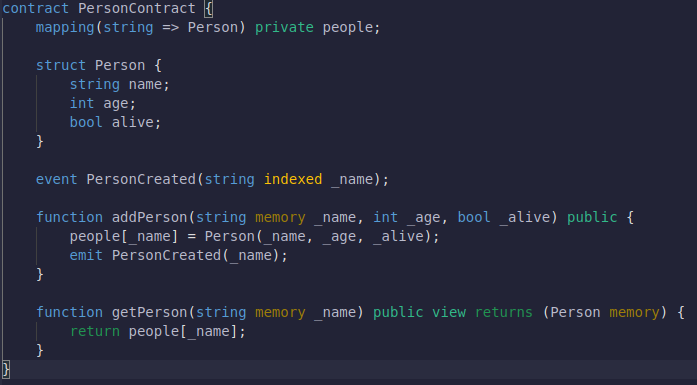
|  |  |  |  |
| --- | --- | --- | --- |
|  | Creation Of Contract | Adding Person | Adding Second Person |
| Gas Fees | 572670 gas | 135258 gas | 118146 gas |

In conclusion storing more data to save the keys leads to more gas fees in the creation of the contract and the transactions in the contract. It is also not very flexible as the smart contract can’t be modified if other data needs to be tracked.

### Track the transactions ourself

In this solution events are added to the transactions that update the complex data. Every time the data structure is updated an event will be emitted. A separate server is created that will store the data that needs to be more accessible. The created server listens to these events and updates the the data it stored accordingly.

First a standard smart contract is created as described in the second paragraph of the simple solution. Further on this smart contract is expanded with events that will trigger every time we update the data.

Now a simple server with an array that will store the keys should be created. Afterwards event listeners for the events the smart contract emits should be added. Every time an event is triggered the array with the keys stored on the server is updated.

Two persons are created as stated in our simple solution. The transaction gas fees are as follows after testing:

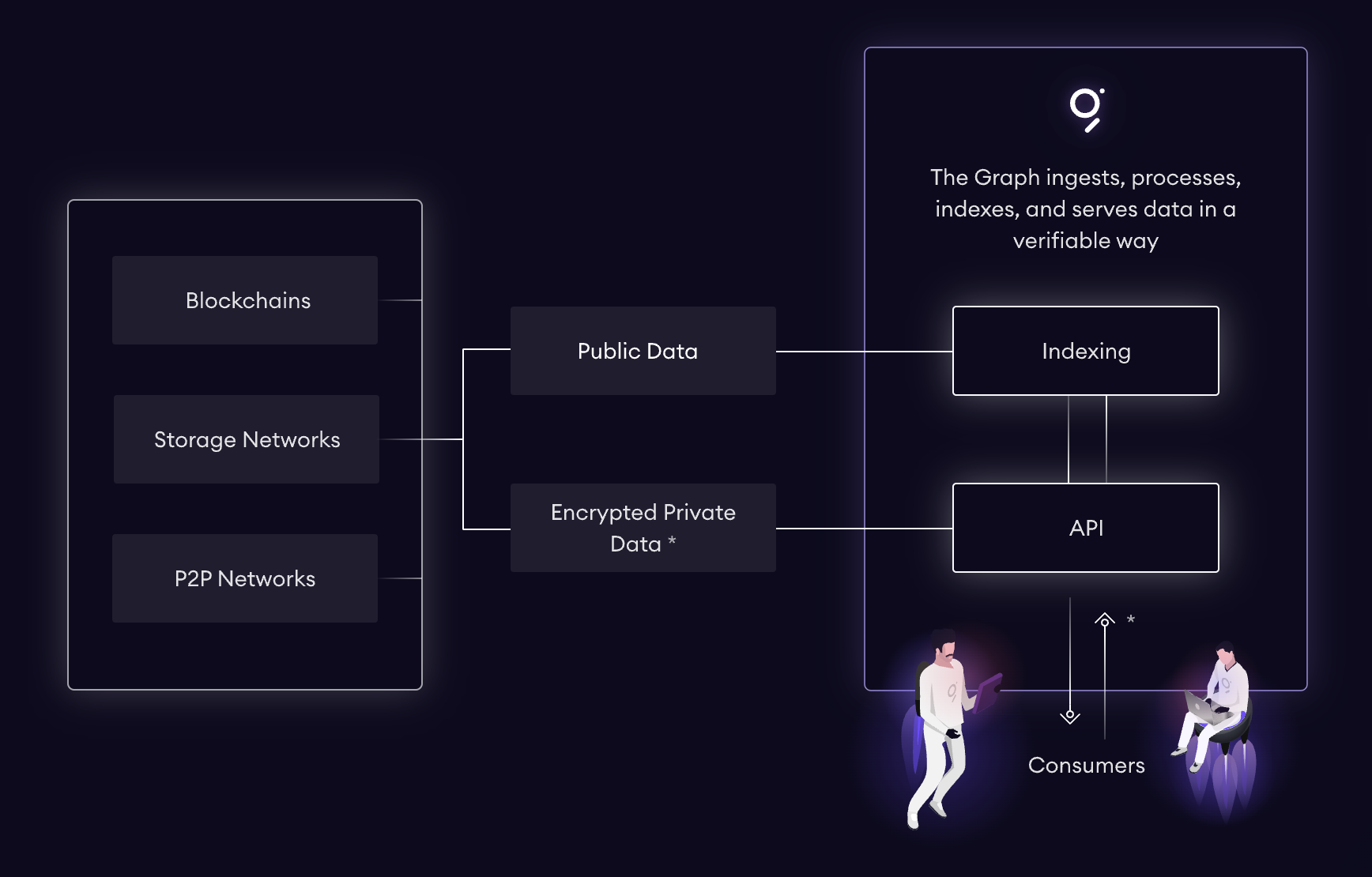
|  |  |  |  |
| --- | --- | --- | --- |
|  | Creation Of Contract | Adding Person | Adding Second Person |
| Gas Fees | 469160 gas | 92138 gas | 92126 gas |

In conclusion it adds extra gas fees because of the events that will be created and triggered however this is negligible. The biggest downside is when the server goes offline it stops tracking the events which makes the data stored invalid. Another downside is the setup and costs for hosting a separate server.

### The Graph

In our previous solution a server keeps track of every emitted transaction event. In this solution multiple servers keep track of the transaction events. The saving of these events is called indexing and the saved data is called a sub-graph. The servers are called nodes and are ran in a decentralized network where everyone can start his own node.

When retrieving data a query is send through their API. The query language is based on GraphQL. The people that query for this data are called consumers.

To ensure the retrieved data is valid The Graph network uses proof of stake as consensus mechanism. A consensus mechanism is a way to ensure the data that is stored and retrieved is valid. With proof of stake the actor that provides the data bets a fixed amount of tokens on the validity of the respective data. If the data is invalid the actor loses his bet and some of the tokens. The Graph uses their own token (GRT) for staking.

The Graph decentralized network has multiple actors to implement this proof. The actors that are discussed here are not the only actors but just the most importance ones for this solution.

1. The Consumers: Query data through a query engine and pay Indexers a fee. The Graph teams expect a fee of $0.00001 per query.
2. Indexers: Provide indexing and querying services as node operators and earn indexing rewards and collect query fees, respectively. They also need to stake tokens, which will be slashed if they conduct malicious practices that harm the network.
3. Fishermen: The fishermen verify the work of the indexers. When an indexer work has been deemed malicious the fishermen that disputed the work earns a fee.
4. Arbitrators: Assigned through decentralized governance, will decide if an Indexer has been malicious.

Because The Graph only works on the main-net of Ethereum we couldn’t test it out and show exact numbers. We can still conclude based on our previous results that gas fees will be higher because of the events that will be created and triggered. The transaction fee is also really low as stated in their Q&A

Furthermore the graph is convenient solution which easily allow the query of all blockchain data. It is cost-effective as the query fee is really low and there are almost no extra gas fees. It runs on a decentralized network which makes it a very reliable solution.

## Conclusion

When trying to retrieve data from the blockchain it can be very difficult especially when this data is stored in a complex data structure like a mapping. To get around this problems we found multiple possible solutions.

In the first solution we conclude that just adding the data to track to a different data structure it leads to more gas fees. It also isn’t very flexible when suddenly we want to keep track of other data.

In the second solution the problem of extra gas fees is solved by adding events to each transaction that updates the relevant data. These events are tracked by a separate server and the tracked data is retrieved from this server. While it removes the gas fees it is very unreliable because there is a single point of failure. If this server is offline the data on the server is out of sync and so invalid.

The third solution builds further on the second solution. Instead of saving the data on one server it saves it in a decentralized network of servers. This way the single point of failure is removed. The only downside is the small fee every time a query to the network is made.

Although the third solution adds a query fee it is much lower than the potential hosting costs for a server and it doesn’t add a lot of extra gas fees. Therefore it is the best solution as it also very reliable and doesn’t have a single point of failure.

## References

Jakub (2021). *The Graph – Google Of Blockchains?* Finematics. <https://finematics.com/the-graph-explained/>

*About the graph.* https://thegraph.com/docs/en/about/

Bybit Learn (2022). *The Graph (GRT): Search Engine of the future.* Bybit Learn. https://learn.bybit.com/altcoins/what-is-the-graph-crypto-grt/

Stefan (2021). *The Graph FAQs & Everything You Need to Know.* <https://forum.thegraph.com/t/the-graph-faqs-everything-you-need-to-know/850>