Blockchain data API
design decisions and documentation

Marcin Pieczka

1 Deployment and packaging

Docker - for and against

The main things that can be accomplished with well chosen deployment and packaging are simplicity of use and cross-platform possibilities witch can positively influence adoption rate of this solution in perspective of both usage and further open source development.

Docker is known for achieving those goals, but not everything about Docker will be helpful. My main concern is running Bitcoin Core in Docker Image, it would be a great inconvenience to people who already have full node on their machine and this problem has to be dealt with.

My proposal is to have Bitcoin Core installed separately, and create FTP server that would provide access to its block data, the rest of the application would by in Docker. With that we can keep the advantages of using docker, and provide a new possibility to the user - keeping Bitcoin Core on separate machine witch when considering that both parts of the application, Bitcoin Core and database with API will be well over 100Gb in size might be a big advantage.

2 Database

Database operations

My least concern are operations that modify the data, and their performance will not be taken into consideration

Main goal is to enable fast querying of the blocks by block hash, time and other block attributes, and to be able to return specified amount of consecutive blocks starting or ending with certain block.

Additionally it might be necessary to enable fast querying for transactions or blocks containing transactions of certain addresses but further research of what users need in this regard is needed

2.1 Database choice

At the beginning lets simplify the choice between RDBMS and NoSQL databases. Out of many NoSQL possibilities I have chosen MongoDB database based on some quick research of different NoSQL systems strengths and weaknesses.

Lets lay out some facts that will help to decide whether to use relational database, or MongoDB

- To achieve fast querying, the data will be strongly denormalized
- You can achieve comparable performance from MongoDB and some RDBMS but Mongo seams to make storing denormalized data idiomatic and RDBMS with highly denormalized data just don't feel right
- MongoDB fully supports JSON, witch will be the format of data received by end user

Based on these facts I will use as my database MongoDB. This problem seams like a perfect usage for database of such type because of its denormalized nature and native support for JSON

2.2 Database structure

At this point I propose having one collection of blocks, each document containing all block attributes like hashes, time, transactions list and others

Indexing increases performance of querying the data and hinders the performance of operations like adding and removing data witch in this case looks like a great bargain. There might be additional memory cost associated with indexes, but this should not be a problem.

The indexes will be added to fields like block hash, time or height, adding indexing to transaction list is also a possibility and will be considered and tested. With indexes on transaction list it should be possible to quickly query for blocks containing transactions in witch given address receives or sends bitcoin, but its hard for me to speculate about this matter without thorough testing in live system.