Hochschule für Technik Rapperswil

A Pruneable Approach for Bazo

by

Stefano Fontana

MAS SE-16

October 2018

Hochschule für Technik Rapperswil

Abstract

MAS SE-16

by Stefano Fontana

Bazo (Esperanto for “base” or “foundation”) is a cryptocurrency created by the University of Zürich for an external financial entity that facilitates the use of a customer bonus points program. Every purchase made by a customer allows him to collect points that can be spent with registered partners. The objective of the Bazo blockchain is to replace the need of having a bespoke contract for each of these partners by replacing the procedure with a decentralized system. The system also allows customers to perform direct payments to partners and other customers without the need of supervised and inefficient administrative procedures. Bazo was created with simplicity and scalability in mind since the beginning, however it faces the same limitations of many existing blockchains, namely scalability, speed and efficiency. Scalability and speed are being tackled by other students.

Storage space is also a limitation commonly faced by blockchains since there is a need to fetch and store all blocks and transaction ever created in order to validate new blocks. The goal of the thesis is to propose a solution for the storage problem by creating a snapshot of the system in a way that older blocks are no longer needed and can be safely discarded/pruned.

Acknowledgements

I would like to thank Dr. Thomas Bocek for giving me the opportunity of working on such an interesting topic and supporting me during these months. His continuous flow of refreshing ideas is quite remarkable.

Contents

Abbreviations

POS	**P**roof **O**f **S**take
POW	**P**roof **O**f **W**ork
POA	**P**roof **O**f **A**uthority
AccTx	Transaction to create/remove accounts
FundsTx	Transaction to transfer funds
StakeTx	Transaction to (un)stake miner
ConfigTx	Transaction to change system parameters
ConsolidationTx	Transaction to store snapshots of the system

Chapter 1

Introduction

During the last year blockchain related topics gained a lot of attention from news outlets and general public not only because of the economic gains/losses but also because of other potential applications that may be instrumental in solving problems derived from lack of trust and decentralization. One of the of most common and basic uses is to send tokens from one user to another in a secure and decentralized way. A financial service provided needs precisely that and, in collaboration with the Communication Systems Group of the University of Zürich created a blockchain called Bazo (meaning "base" or "foundation" in Esperanto) to realize the project of the financial provider while exploring the topic of blockchains [1]. The aim of this thesis is to increase the efficiency of Bazo blockchain.

1.1 Motivation

Many blockchains are trying to solve the same problems (reaching a trustworthy coherent global view of the system's state) by using different techniques (PoW, PoS, PoA, etc), however most of them face very similar limitations like speed (time to get a certain transaction approved) and the space on disk occupied by ever growing chain of blocks that needs to be stored. Addressing the aforementioned problems can help the widespread adoption of blockchain related applications. For instance, while it is feasible having the most popular blockchains on a local pc, it proves challenging if not impossible running them on mobile devices. At the time of writing a bitcoin node needs at least 180GB [2] to run, while an Ethereum full node needs more than 660GB [3].

Solving these limitations means that more people can join the network and benefit from the advantages of blockchains. Furthermore more participants can become miners

and contribute to lowering the risk of problems or attacks derived from the lack of decentralization.

1.2 Description of Work

The thesis will focus on the problem of the space required to store the blocks. The solution proposed needs to adapt to the already working system while keeping in mind that it can be further developed. Security is paramount for blockchain related applications so the changes adopted should maintain the same level of security for all Bazo components.

The desired outcome of the thesis is the Bazo blockchain that works exactly as before from a user's point of view while occupying less space. The difference is a built-in system that creates snapshots and prunes batches of blocks that are no longer needed in order to curb the space occupied on disk.

1.3 Thesis Outline

Chapter 2 familiarizes the user with the Bazo ecosystem developed by UZH [1] and the research currently in progress. Chapter 3 describes the solution adopted for pruning Bazo. Chapter 4 discusses the implementation of such solution. Chapter 5 analyses the solution implemented. Future possible improvements are proposed in Chapter 6 and Chapter 7 follows with a summary and a conclusion.

Chapter 2

Background and Related Work

The following section describes the Bazo blockchain including features that are being developed or have been integrated since the first version. The second part of the chapter looks at how other blockchains are handling the problem of storage space.

2.1 Description of Bazo

Bazo is an implementation of a decentralized public ledger that is maintained by entities called miners. The current state of the system is represented by all the transactions that were approved. Transactions are validated and grouped into a block that references the previous block, the result is a chain of blocks also known as blockchain. Unlike Bitcoin [4], which relies on unspent transaction outputs, Bazo is account based so all the information relevant to a particular account can be stored in a single entity.

It is important to note that Bazo is a private invite-only network and only root accounts can add/remove accounts and change configuration.

2.1.1 Implementation of Proof-of-Stake

The consensus protocol describes the steps that need to be followed by a rational miner in order to choose the single correct version of the history. Originally Bazo used a Proof-of-Work (POW) consensus protocol which allowed a miner to prove that a certain amount of resources (time and calculation power) were used to calculate a hash that respects certain conditions. Similarly to other blockchains, the amount of computation needed to maintain the network using a POW consensus protocol represented a limit and Bazo adopted Proof-of-Stake (POS) consensus protocol [5]. This change solved

numerous problems, e.g.: environmental issues, risk of centralization, high transaction fees [5, p. 12]. In the implemented protocol the next validator is elected among a pool of available validators and several measures have been taken in order to prevent attacks. A *configTx* transaction has been added to control the newly introduced parameters for the POS protocol.

2.1.2 Sharding

Given the limited number of transactions that can be approved per block, the implementation of sharding is actively being investigated to make Bazo more scalable and will eventually be integrated. See Chapter 6 for more information about the future integration with sharding.

2.1.3 Light Client

This addition to Bazo made it possible to check the balance of a particular account and send transactions without having to download the whole blockchain. This is achieved with the use of *Bloom filters* in the header that says whether may contain transactions affecting the state of the desired account [6].

2.1.4 Mobile Wallet

A mobile wallet was added to the Bazo ecosystem. This was designed as a Progressive-Web-Application and it allows to perform all the operations available in a full client. More details in [7].

2.1.5 Virtual Machine

Sending monetary transaction in a reliable way is important but the blockchain technology can also handle the execution of decentralized application an smart contracts. An implementation has been developed and is now part of Bazo [8].

2.1.6 Blockchain Explorer

The Bazo block explorer [9] is an application that makes it possible for anyone to check the details of the blocks without having to participate in the network. This brings more

transparency in a network that is managed by a central financial entity. It also allows the financial entity to easily send commands thanks to a user friendly web page.

In Bazo there are 4 types of transaction possible:

- AccTx: add or remove accounts
- FundsTx: move funds between accounts
- StakeTx: add or remove a miner from staking
- ConfigTx: change parameters of the system.

AccTx and ConfigTx are only available to Root accounts which means that only administrators are allowed to create or remove accounts and change system parameters.

2.2 Analysis of the Problem

In Bazo, a node must store all the blocks ever created with all the transactions since the genesis block. This allows a new participant of the network to download all the blocks since genesis up to the most current one and revalidate every single block (and reject part of the chain should a block be invalid). Having to store the whole blockchain leads to a couple of problems:

2.2.1 Storage Limitations

A rational miner would only hold the last few blocks and discard all the others in order to free up space. However if all participants in the blockchain are rational then no one would store the whole blockchain and a new miner wouldn't be able to download the whole chain and validate it, thus it wouldn't be able to join the network. Therefore blocks must be stored by at least one node of the network to keep the system running.

2.2.2 Bootstrapping Speed

When a new node joins the network it has to download all the blocks that were mined in order to get the latest state of the system and start mining new blocks. This can take a considerable amount of time that is proportional to the length of the blockchain.

In the same way, a client that wishes to know its balance must download all the block headers and if the bloomfilter field indicate that the searched account hash was part of

a transaction, the body must be also downloaded. This operation can also require a considerable amount of time and having to wait more than a few seconds just to know the balance can be frustrating for a customer.

2.3 Related Work

2.3.1 Rollerchain

Rollerchain [10] is a proposed consensus protocol that aims to solve problems of storage a bootstrapping in the Bitcoin system. It does so by giving the right to generate a block (and get the subsequent reward) to a participant providing non-interactive proof of storing a subset of the past state snapshots. This does not directly reduce the size of the blockchain but provides a way to reward participants for supporting the network.

2.3.2 Parity

Parity [11], a client for the Ethereum blockchain introduced a *warp sync* that significantly reduces the time needed to get the full copy of the state for a given block [12]. With a regular interval, one of the Parity nodes takes a snapshot of the systems that can be fetched by any other node over the network, allowing a much faster sync.

The missing blocks of the chain are then downloaded in the background.

2.3.3 Geth

Geth offers 3 options for syncing the blockchain:

- **full**: download block header, block data and validate the whole chain.
- **fast**: download block header, block data and validate the last few thousand blocks.
- **light**: download current state and download missing blocks as needed.

This approach brings a faster initial syncing but a full node will still have to obtain the whole blockchain resulting in a great amount of space needed.

Chapter 3

Design

This chapter considers different solutions proposed and then proceeds to describe why the selected one is the best for Bazo.

3.1 Solution adopted

Since Bazo is a blockchain aimed at transferring bonus points there is no need to keep track of all the transactions. Users only worry about the current balance and not about a single specific transaction that occurred in the past. This allows us to create a snapshot capturing the current state of the system with a few bytes per account and forget about the history of the transactions. The snapshot consists of account statuses that are encoded in transactions that are then added to a block.

3.1.1 Transaction vs Block

Creating a new transaction type is the most sensible choice because it can be simply added to the set existing ones. Adding a new block would inevitably entail large changes in order to support the new block type and would only save a few bytes of space without having any important advantages over a new transaction. Moreover the support for a new block would require changes to the whole Bazo ecosystem.

3.1.2 ConsolidationTx size

A Bazo transaction doesn't have a limit in terms of size so theoretically all accounts can be contained in a single consolidationTx. On the other hand a big transaction can

take a considerable amount of time to download so clients might time ou and fail to download the transaction. For this reason a block can contain more than consolidation transaction.

3.1.3 Validation of Consolidation Transaction

When a new block containing consolidation transactions is received the nodes check whether the snapshot is correct or not before adding it to the chain.

To check correctness of the snapshot at a specific height each node calculates the state between the previous and current consolidation point. Once the expected consolidation state is calculated the nodes check if the fields of each account match the expected ones. If correct the nodes add the block to their current branch of the blockchain.

The system parameters are checked in the same way.

3.1.4 Storing a lot of Accounts

The solution adopted can accommodate a very large number of accounts because only the hash of the consolidation transactions are stored in the block. Since the maximum size of the block is **5MB**, the number of consolidationTx hashes (each occupying *32Bytes* that can be stored is:

$$\frac{5MBytes}{32Bytes} = 150\ thousand\ hashes/block$$

Note that this is calculated excluding the space occupied by other fields since they account for few hundreds bytes.

If the limit is reached, for example if more fields are required to define an account, the snapshots may be split into multiple blocks.

Chapter 4

Implementation

In this chapter the details about the implementation of consolidation transaction are explained.

4.1 Anatomy of a Block

A block of the Bazo blockchain is made of a header and a body. A Bazo header contains the following fields:

```
package protocol
type Block struct {
        Header       byte
        Hash         [32]byte
        PrevHash     [32]byte
        NrConfigTx   uint8
        NrElementsBF uint16
        BloomFilter  *bloom.BloomFilter

        ....
        // Body
}
```

Header is used to recognise the type of block being used.

Hash is the hash value of the block calculated using the function *Block.Hash()*

PrevHash hash of the previous block.

NrConfigTx the number of configuration transactions containing changes to the system parameters.

NrElementsBF the number of elements in the Bloom Filter.

BloomFilter encoded Bloom Filter. This is used to determine whether the state of a specific account may be affected in any of the transactions encoded into the block body. This allows a client to skip wirth certainty blocks that don't contain any transactions affecting the account given.

The block's body structure is as follows:

```
package protocol
type Block struct {
     // Header
        ....
        Nonce                  [8]byte
        Timestamp              int64
        MerkleRoot             [32]byte
        Beneficiary            [32]byte
        NrAccTx                uint16
        NrFundsTx              uint16
        NrStakeTx              uint16
        SlashedAddress         [32]byte
        Seed                   [32]byte
        Height                 uint32
        HashedSeed             [32]byte
        ConflictingBlockHash1  [32]byte
        ConflictingBlockHash2  [32]byte
        AccTxData     [][32]byte
        FundsTxData   [][32]byte
        ConfigTxData  [][32]byte
        StakeTxData   [][32]byte
}
```

Timestamp block creation UNIX time.

Merkle Root value of the merkle's root.

Beneficiary address of the account receiving the rewards of the transactions and the block.

NrAccTx, NrFundsTx, NrStakeTx number of transactions of each type encoded into the block.

AccTxData, FundsTxData, ConfigTxData, StakeTxData the hashes of the transactions included in the block.

Height height of the block in the blockchain.

Seed, HashedSeed fields used for POS.

SlashedAddress, ConflictingBlockHash1, ConflictingBlockHash2 fields used for slashing.

4.1.1 Solution adopted

In order to save the snapshot of the system a new transaction called **ConsolidationTx** is added.

Since a transaction may be limited in size, the block should be able to contain multiple consolidation transactions. For this reason, similarly to the other transactions, a field **NrConsolidationTx** is added to the header of the block to know how many consolidation transactions are part of the block.

The NrConsolidationTx field is part of the header so a block with consolidations can be immediately recognised without having to download the whole body. It is also consistent with ConfigTx which is also in the header. This is because a consolidationTx is always of interest so it should always be downloaded by the client since it contains all the accounts.

The ConsolidationTxData field contains the hashes of the consolidation transactions. Similarly to the other transaction types, the hash is requested in the network and then transaction is then fetched and decoded.

4.1.2 ConsolidationTx

The ConsolidationTx is represented by the following object

```
type ConsolidationTx struct {
        // Header
        Header     byte
        Reward         uint64
        PreviousConsHash [32]byte
        NrAccounts uint64

        // Body
        Accounts []ConsolidatedAccount
        ActiveParameters conf.Parameters
}
```

Header reserved for future uses.

Reward reward for including this transaction into the next block mined.

PreviousConsHash pointer to the previous block containing a consolidation transaction. Thanks to this field it's possible to reach the genesis block via the previous consolidation points and safely forget about the non consolidation blocks.

NrAccounts number of accounts consolidated.

Accounts list of consolidated accounts

ActiveParameters current system parameters, always included.

A consolidated account is represented through the following structure

```
type ConsolidatedAccount struct {
        Address [64]byte
        Balance uint64
        TxCnt   uint32
        Staking bool
}
```

New fields can be added without the need of a hard fork.

Address is the public key of the account.

Balance amount of coins held by account at the moment of consolidation.

TxCnt number of transactions executed by account at the moment of consolidation. This is used to avoid double spending.

Staking whether the account is taking at the moment of the consolidation.

This data model can be easily expanded should new needs arise in the future.

4.1.3 Validation

The transaction is valid if all the transactions belonging to blocks between the current and previous consolidation points have been included. The checking is done by simply redoing the calculation for the snapshot and making sure the numbers are correct.

4.1.4 Creation of a Consolidation point

The **ConsolidationInterval** defines the number of blocks between consolidation points is a configurable parameter that has been added to the set of system parameters.

A miner creates the consolidation point if:

- miner is staking
- miner has been elected to mine the next block
- the height of the next block modulo *consolidationInterval* is 0

4.1.5 Space Considerations

An empty block (a block without transactions) occupies *320 Bytes* of space. This means that if a block is mined every second, around **30 MegaBytes** of space a day will be needed. With an increase of the number of transactions the space consumed can easily increase to **50 MegaBytes** assuming a fundsTx is validated every second (a *fundsTx* occupies **213 Bytes**).

A consolidated account occupies just **77 Bytes** even with 10 thousand accounts, the size of all the consolidated accounts would still be below 1 MegaByte. So the space saved by deleting the non consolidation blocks is very considerable.

4.1.6 Bootstrapping Speed

The consequent advantage is that participants joining the network only need to download all the blocks until the latest consolidation block included, plus all the previous consolidation blocks until the genesis block. This allows a client to get the current state of the system in a few seconds.

4.1.7 Consolidation Interval

The frequency of the snapshots can be adjusted by a *Root account* using the following command:

go clientMain configTx 0 11 new_consolidation_interval fee txCnt RootPublicKey

This commands tells the system to take a snapshot with a regular interval. The interval depends on the traffic of the network, for instance, if there are only few transactions, creating too many snapshots may be counter-productive. Likewise, taking a snapshot infrequently won't bring the benefits listed above.

4.2 Deletion of non consolidation blocks

After the creation of a consolidation point, the previous blocks shouldn't be deleted immediately. The system should wait long enough to take into account possible rollbacks, this can be at a specific height or after a number of sufficient snapshots taken. For instance, if the deletion point is set to 5 consolidation points, after the 6th snapshot the system can delete the non consolidation blocks between the 1st consolidation block

and the genesis block, after the 7th snapshot, all the blocks between the 2nd and 1st consolidation block and so on.

The deletion interval is hardcoded and cannot be changed although future revisions of the protocol can easily add this as a system parameter that can be adjusted with a **configTx**.

Chapter 5

Evaluation and Tests

This section describes the testing of the consolidation transaction.

5.1 Security evaluation

5.1.1 Safety of pruning

Having the possibility to download the whole blockchain allows a node to independently validate every single block every created. It can be argued that creating a consolidation block is as safe as creating a *normal* block. This can be claimed because a consolidation block is effectively a normal block (plus the consolidation transactions) and starts acting as snapshot point once the intermediate blocks are removed. Since the intermediate blocks are not deleted immediately, the miners always have the whole recent history (the length is configurable) so it's always possible to fully validate the non-pruned part of the chain.

5.1.2 Genesis block

All blockchains have an initial block called *genesis block*. Clients like *geth* for Ethereum or Bitcoin [13] usually contain the parameters needed for generating the genesis block [14] which will have a non zero hash [15]. For simplicity reasons the Bazo blockchain generates a genesis block with a null hash. This may represent a problem for the first consolidation snapshot since the hash of the transaction is calculated only with the previous consolidation hash. Since intermediate blocks between the genesis block and the first snapshot are not deleted, first snapshot may be forged and a malicious miner may trick another miner in bootstrapping mode into building a completely different

chain. This is known as *eclipse attack*, that is a node if fooled into accepting false data that appears to originate from other miners and the result is the attacked node wasting time approving forged transactions. A simple solution can be not deleting the first interval so that the first snapshot block can be verified.

5.2 Unit and Integration Tests

The was already a very good test coverage. More unit tests have been added to cover the new *consolidationTx*. Furthermore integration tests have been added to make sure that the individual modules work as expected. The integration test first spawns a miner (root account) that receives commands sent (account creation, fund transfers, change of system parameters, etc..) from the test clients and then it starts a new miner that joins the network and downloads the consolidation blocks. The test then checks that balance and other parameters in the current state are the ones expected.

This test can be easily expanded to cover other cases.

Chapter 6

Future Work

The consolidation transaction can be extended in the future to accommodate more account details if necessary. The newly designed transaction should be tested to make sure it doesn't introduce new weak points.

6.1 Block header

NrConsolidationTx which is *uint8* can be moved to the body and be replaced with a simple bool to save 7 bytes in the block header. The same can be done for *NrConfigTx*. The speed of the system can be improved with changes of this type given that block header is the entity being exchanged the most in the network .

6.2 Multiple Back-References to Previous Snapshots

Right now consolidation blocks with the snapshot of the system are created but are never deleted. In the long run consolidation blocks can occupy a lot of space even though data saved in older blocks has no use other than providing a link to the previous consolidation block. This problem can be solved by saving references to multiple consolidation blocks instead to just a single one. This would made some older consolidations blocks redundant and a simple "garbage collector" may look for past referenced blocks and delete them (along with the consolidations transactions).

6.3 Consolidation Speed

Further optimizations can be introduced, for example instead of starting to create the snapshot exactly at the moment needed, a snapshot that is continuously updated could be used so that there are no slowdowns during the creation of the consolidationTx and the mining of the block that should contain the transaction created.

6.4 Compression of Snapshots

Given the highly structured format of snapshots, it's probably worth investigating the possibility of compressing them. This would help to further reduce the space needed by the Bazo blockchain and reduce the size of data to be transmitted.

6.5 Election Consolidator Node

Right now the consolidator node is the one that happens to be the POS elected node for the block of height x where $x > 0, x \ mod \ cons\_interval \ = \ 0$

6.6 Integration with Sharding

Scalability is another of the problems afflicting blockchains [16], more specifically Bazo cannot process more transaction than a single node can, and in its current status the system can have maximum one node at a time mining blocks.

At the time of writing a solution is being actively researched and will be implemented in the future. The fundamental idea of sharding is to partition the state and history into smaller chunks called *shards*. Having parallel histories at the same will allow to multiply the number of transactions approved per block, however the consolidation protocol will need to be adjusted once sharding gets implemented.

Chapter 7

Summary and Conclusions

The purpose of the thesis was to find a way to reduce the number of blocks that need to be stored to keep the Bazo blockchain functioning. Creating inter-linked snapshots of the global state of the system proves to be a very effective way to curb the size of the Bazo blockchain. The safety is guaranteed by the fact the system always works on the current branch of the blockchain which is represented by the last n non-pruned blocks of the chain. n depends on the current parameters of the system and should be big enough to account for possible chain rollbacks.

A snapshot is saved in one or more consolidation transactions which is a new type of transaction created to accommodate the consolidated global state. The structure created adapts well to the current Bazo ecosystem, it doesn't require radical changes and can be easily expanded in the future. Possible future improvements have been suggested and integrations with forthcoming developments have been discussed.

Appendix A

Executing the test

To start the integration test simply type ”go test bazo-miner/integration”.

The test will

- Start a miner using a newly created root account.
- Send command to create a new non-root account.
- Transfer funds to the newly created account.
- A system parameter is changed.
- Stake the new account.
- Spawn a new miner that uses the new account.

After this this the state of the non-root miner is checked to make sure the state of the blockchain is the one expected.

Bibliography

[1] Livio Sgier. Bazo – a cryptocurrency from scratch, 2017. URL https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Livio-Sgier.pdf.

[2] 2018. URL https://www.blockchain.com/charts/blocks-size?

[3] 2018. URL https://bitinfocharts.com/ethereum/.

[4] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL https://bitcoin.org/bitcoin.pdf.

[5] Simon Bachmann. Proof of stake for bazo, 2018. URL https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Simon-Bachmann.pdf.

[6] Marc-Alain Chetelat. A mobile light client for bazo, 2018. URL https://github.com/bazo-blockchain/bazo-client.

[7] Jan von der Assen. A progressive web app (pwa)-based mobile wallet for bazo, 2018. URL https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Jan-von-der-Assen.pdf.

[8] M. Steiner E. Meier. Virtual machine for bazo, 2018. URL https://github.com/bazo-blockchain/bazo-smartcontract.

[9] Luc Boillat. A blockchain explorer for bazo, 2018. URL https://files.ifi.uzh.ch/CSG/staff/bocek/extern/theses/BA-Luc-Boillat.pdf.

[10] A. Ojiganov A. Chepurnoy, M. Larangeira. Rollerchain, a blockchain with safely pruneable full blocks, 2016. URL https://arxiv.org/abs/1603.07926?context=cs.

[11] 2018. URL https://github.com/paritytech/parity-ethereum.

[12] November 2016. URL https://paritytech.io/announcing-parity-1-4/.

[13] https://github.com/bitcoin, 2009. URL https://github.com/bitcoin/bitcoin/blob/78dae8caccd82cfbfd76557f1fb7d7557c7b5edb/src/chainparams.cpp.

[14] Ethereum foundation, 2015. URL https://github.com/ethereum/go-ethereum/blob/9e24491c65120d3fb36da45e57a0cb87c549f621/core/genesis.go#L301.

[15] 2015. URL https://www.etherchain.org/block/0.

[16] 2018. URL https://en.wikipedia.org/wiki/Bitcoin_scalability_problem.