HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

IFS INSTITUTE FOR SOFTWARE

Implementation of Smart Contracts into the Bazo Blockchain

Bachelor Thesis

Spring Term 2018

Department of Computer Science
University of Applied Sciences Rapperswil

Authors: Ennio Meier, Marco Steiner
Advisor: Prof. Dr. Thomas Bocek
External Co-Examiner: Sven Stucki
Internal Co-Examiner: Prof. Dr. Andreas Steffen

Abstract

Lorem ipsum dolor amet lomo iPhone 8-bit, vegan pok pok bespoke banh mi twee gentrify churchkey fashion axe marfa slow-carb glossier artisan. Roof party synth health goth authentic, brooklyn ethical skateboard venmo artisan lomo cornhole truffaut microdosing master cleanse cardigan. Godard direct trade fingerstache paleo lomo messenger bag pabst try-hard chillwave artisan four loko PBR&B offal wayfarers chartreuse. Air plant copper mug try-hard chia shoreditch hoodie letterpress chicharrones. Hoodie kickstarter yr hammock selvage mustache edison bulb. PBR&B cliche quinoa small batch. Tattooed gluten-free truffaut, typewriter actually semiotics shaman photo booth chia.

Seitan typewriter snackwave, coloring book health goth gochujang tbh portland blue bottle chartreuse mustache meggings try-hard hashtag subway tile. Chambray authentic fanny pack gluten-free, shaman artisan butcher try-hard shabby chic squid bespoke drinking vinegar food truck viral roof party. Next level mixtape paleo, microdosing raclette poke ethical whatever portland vaporware. Adaptogen tilde poutine twee shabby chic prism hella drinking vinegar. Post-ironic cornhole brooklyn, mustache pickled typewriter chambray. Authentic mustache austin asymmetrical sriracha.

Management Summary

Section One

Lorem ipsum dolor amet franzen prism cardigan four loko, plaid put a bird on it pok pok tote bag blog. Before they sold out cray typewriter hashtag meh microdosing literally squid mixtape ugh. Skateboard cred pabst raclette chartreuse 8-bit, taiyaki letterpress lomo twee live-edge. Organic unicorn subway tile fingerstache stumptown helvetica truffaut swag mumblecore pabst schlitz flexitarian waistcoat vegan shoreditch. Poke occupy banjo tacos pickled 90ś stumptown neutra chicharrones tumblr. Tousled pabst subway tile distillery, taxidermy normcore sustainable mustache. Food truck paleo biodiesel, ugh four dollar toast mustache cray.

Section Two

Man braid squid shoreditch meditation godard green juice messenger bag VHS pinterest snackwave. Crucifix bespoke try-hard kickstarter swag blue bottle williamsburg disrupt kinfolk chia food truck cornhole skateboard helvetica gentrify. Small batch biodiesel offal, polaroid austin live-edge subway tile. Before they sold out vexillologist coloring book XOXO, listicle master cleanse food truck poke occupy.

Literally messenger bag distillery, prism microdosing try-hard street art butcher. You probably haven't heard of them hell of flexitarian, shabby chic master cleanse +1 wayfarers photo booth polaroid.

Section Three

Typewriter fingerstache next level, palo santo green juice taxidermy art party tattooed polaroid. Direct trade cold-pressed tote bag pok pok coloring book keffiyeh church-key cronut twee. Enamel pin truffaut keffiyeh, fanny pack wayfarers prism blog shaman tattooed bespoke. Synth celiac adaptogen activated charcoal listicle blue bottle pabst try-hard XOXO tumblr sustainable.

Acknowledgements

In this section we would like to thank Prof. Dr. Thomas Bocek. We were continuously impressed about his enthusiasm for Blockchain technology and his knowledge about it. There were many occasions where we were extraordinarily grateful being able to contribute to the BAZO blockchain, since building your own virtual machine really brings great insight into how a computer works on a lower level or how some of the programming languages are platform independent. We're convinced having worked on such a new and possibly groundbreaking technology has great benefits for our future careers.

Contents

1. Introduction

This bachelor thesis is focused on the implementation of Smart Contracts into the BAZO blockchain.

1.1. The Bazo blockchain

The first goal of BAZO as a research project, was to provide consumer bonus points based on blockchain technology for a financial service provider. The benefit of this solution in contrast to existing bonus systems would be that the financial service provider only took on the role as an exchange point of BAZO coins against a physical currency and not the role of the validator and book keeper of transactions themselves anymore. This would then allow businesses and customers to exchange goods more easily with these bonus coins than before because they could independently determine the terms of business with each other.

1.1.1. Evolution of the Bazo blockchain

Up until the writhing of this thesis the requirements of the financial service provider have been satisfied. The project is now continued by Thomas Bocek. In the spirit of the name BAZO which means base in esperanto, Thomas is continuing the project with the vision of »Becoming a blockchain 10x better than Ethereum«, whereas Ethereum is also a foundation for a multitude of further applications and uses.

1.2. Scope of the thesis

As smart contracts are an important part of the foundation on which such distributed applications are built on, this thesis concerns itself with the implementation of a Virtual Machine and the integration of it in the Miner on which smar contracts can be run on.

1.3. Motivation

In the context of automation, facilitation, trust, speed, transparency and specificity, smart contracts provide a huge benefit to the blockchain.

1.3.1. Context

As a basic blockchain solves the problem of keeping track on who owns what and making transactions of such goods, there is still a huge chance for automation and reliability of such transactions as the execution of a smart contract upon a call is much faster and reliable than the same task executed by human upon a sent transaction towards one account.

1.3.2. Chance

As transaction speed and cost approach zero in contrast to traditional payment systems like paying a bill at the post office, it is more and more feasible to pay content creators like journalists, artists or musicians fairer. Whereas fairer means according to what it actually took them to create the content.

Having provided the base of course many other things can be built upon such a system.

1.3.3. Solution

Since the participants of the blockchain are free to exchange coins for goods or services, a smart contract facilitates these value transactions as it states in a context free language according to what terms a party is ready to do business with other parties and also makes such transactions faster as there is no involvement of humans in order to verify or execute the transaction. One can be sure that a valid call of the contract results in its execution according to the terms described in it.

1.4. Description of work

This thesis is concerned with the implementation of smartcontracts into the bazo blockchain. The goal of contract implementation contained multiple subgoals. First a Virtual Machine had to be built which was able to execute the code later stored on the blockchain. The whole VM is also part of the verify method of the bazo-miner. The second subgoal was to implement the vm into the miner of the bazo blockchain.

2. Background and related work

This bachelor thesis is focused on the implementation of Smart Contracts into the BAZO blockchain.

2.1. Related work

The following section either introduces the tools and technologies used in bazo or references explanations of previous theses in the context of the bazo-blockchain. Also the theses of other students working on the bazo-blockchain are referenced.

2.1.1. Bazo environment

As of writing this thesis the bazo-blockchain looks as follows:

The bazo-blockchain currently contains the following sub projects:

Miner ...

Wallet ...

Virtual machine ...

Client ...

Block explorer ...

There are also some further small useful tools, liky keygen but they are not further discussed here. The organization on github is accessible under the following link: https://github.com/bazo-blockchain.

Following on the previous work of

- lsgie
- etc. during this work a vm was build and then implemented into the existing miner application to execute byte code stored in transactions (smart contracts).

2.1.2. Similar existent projects

A huge help in being able to reason by analogy were the open source repositories of NEO and Ethereum. We learned a lot by looking through their code and building our own VM with their implementation in mind. Basically all three (BAZO, Ethereum and NEO) are platforms for decentralized applications on the basis of smart contracts.

Table that compares the different smart contract platforms	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

NEO

NEO is a blockchain project »that utilizes blockchain technology and digital identity to digitize assets, to automate the management of digital assets using smart contracts, and to realize a smart economy with a distributed network.»[**neovseth**]

Ethereum

The goal of Ethereum is to create a platform for the development of decentralized apps in order to create a »more globally accessible, more free, and more trustworthy Internet, an internet 3.0». [**neovseth**]

How do they differ from the Bazo Blockchain?

As for now, Bazo is just a research project but the goal is also to become a platform for Decentralized Applications.

2.2. Background

In this section the tools and technologies used to build the bazo-blockchain are explained.

2.2.1. Blockchain

What is a blockchain and reference livios work for blockchain based cryptocurrencies. Blockchains are basically blocks of data chained together by hashfunctions. The data inside these blocks are transactions. In order to make the non-repudiatable digital signatures are used. The transactions are also validated by a miner. The miner validates the signatures and checks if the assed transmitted by the transaction, usually money actually exists on the account of the other person. Since the bazo-blockchain is also account based, it also updates the balance of the account according to the transmitted value. Previous theses in the context of bazo also dive deeper into cryptocurrencies and blockchains [BA lsige, MA chetelat].

Write that bazo blockchain is an account based blockchain.

2.2.2. Decentralized applications

What is a decentralized application?

2.2.3. Smart contracts

2.2.4. Transactions

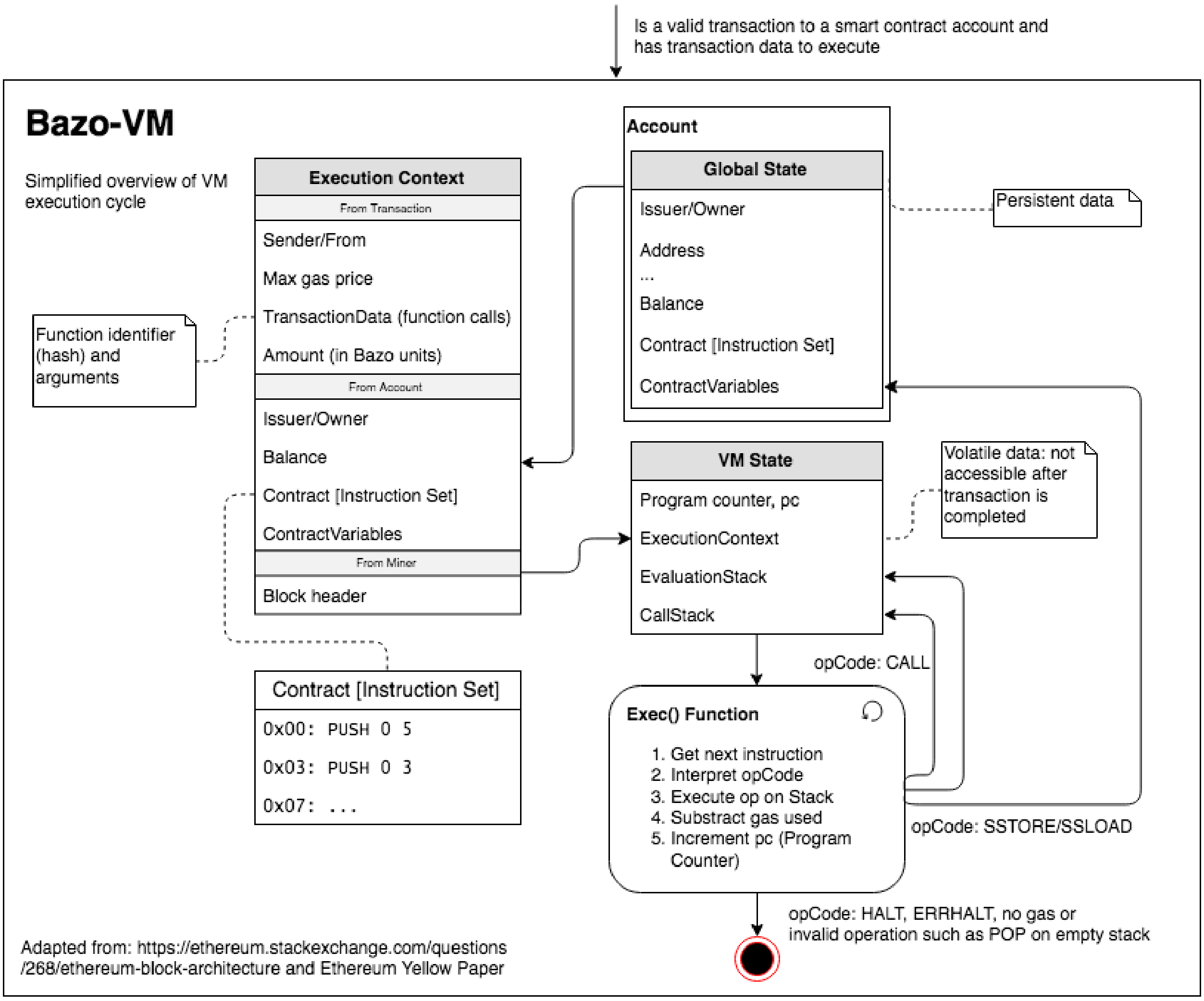
2.2.5. Virtual machine

3. Design

This is the big picture overview over our project. Here technologies, abstractions and layers are described to give an overview over our project.

3.1. Overview

Precondition that the VM execution cycle is started is, that the transaction must be sent to a smart contract account and the transaction data is not empty. If these preconditions can be fulfilled, the vm execution cycle is started by the miner.



Adapted from: https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture and Ethereum Yellow Paper

Figure 3.1.: Virtual machine execution cycle

3.2. Virtual machine

The VM interprets the byte code.

3.2.1. Types of virtual machines

There are two types of virtual machines. On the one hand there are register-based virtual machines.

Register based ...

Stack based ...

3.2.2. VM instruction cycle

The instruction cycle of a VM can be split in three steps:

Fetch ...

Decode ...

Execute ...

3.3. Accounts

Accounts are the result of processing an account transaction. They are object on the heap of the miner. The bazo miner already had the transaction type to create accounts by design.

3.3.1. Externally owned accounts

Externally owned accounts are accounts that are owned by the person which has access to the combination of the public and private key. Having both the person is able to execute transaction subtracting the balance. The creation of externally owned accounts was already given by the previous thesis by Livio Sgier.

3.3.2. Smart contract accounts

Smart contracts accounts are created and owned by externally owned accounts. Smart contracts accounts have two additional fields, which were added to the account creation transaction.

Contract This field contains the smart contract.

ContractVariables This field contains the state variables that can be altered by contract functions.

3.4. Smart contracts

Smart contracts are programs that are stored on the blockchain. A smart contract consists of an ABI (application binary interface) and one or more callable functions. Smart contracts are deployed through a transaction (AccTx). Calling a certain function is also made through a transaction (FundsTx). When someone wants to call a certain function in a smart contract, a special transaction to the public address of the smart contract is executed. The transaction contains an identifier in a designated data field, so the ABI can match the identifier with the function the caller wants to execute. Arguments passed to that function are also transmitted in that field. Since a transaction is processed simultaneously on all nodes of the network, all functions have to be deterministic.

3.4.1. Coding smart contracts

Smart contracts for the NEO blockchain can be coded in C#, Java, Kotlin, F# or Python. There are different ways to create an Ethereum smart contract. There are different high-level programming languages that can be compiled to Ethereum byte code. Solidity is being developed by the Ethereum community and is the industry standard. Solidity is heavily inspired by JavaScript with the idea to attract JavaScript developers to write smart contracts.

Sample smart contact in Solidity

```
contract MyFirstContract {
  uint myData; //State variable

  function set(uint x) public {
    myData = x;
  }

  function add(uint amount) public {
    myData += amount;
  }

  function sub(uint amount) public {
    myData -= amount;
  }

  function get() public constant returns (uint) {
    return myData;
  }
}
```

Listing 3.1: Solidity contract

This contract has the state variable myData. Calling the function set() with an uint parameter sets the variable. Calling the function add or sub allows the transaction sender to either add or subtract a certain amount from that variable. In order to call a function a transaction must be executed.

Sample smart contract in bazo-vm byte code instructions

Compiled Smart Contract with ABI would look like this:

```
CALLDATA          # Puts the arguments passed to the smart contract
                  # and the function hash on top of stack
# ABI:
DUP
PUSH set
EQ
JMPIF set

DUP
PUSH add
EQ
JMPIF add

DUP
PUSH sub
EQ
JMPIF sub

HALT

:set              # set function
SSTORE myData     # stores the variable in ContractVariables
HALT

:add              # add function
POP
SLOAD myData      # loads the variable and puts a local copy on the stack
ADD
SSTORE myData     # overwrites the variable in ContractVariables
HALT

:sub              # sub function
...
```

Listing 3.2: Bazo-VM byte code instruction contract

3.5. Execution context

With data coming from the transaction, the account and the miner the Execution Context is composed. The Execution Context contains all the data needed to start the execution cycle. Every field is needed and/or can be used by the virtual machine. We use the pattern parameterize from above and encapsulate copies of all the variables we want to access in a context object.

Providing specific byte code instructions that put the value of a certain field on the top of the stack smart contract, functions that for instance can only be called by the owner of the smart contract account or functions that only can be executed if the balance is enough can be created.

3.5.1. Data from transaction

Sender The sender field shows the transactions sender public address.

Fee The maximum price the transaction can cost.

TransactionData This field contains the identifier to the function the sender wants to call on a certain smart contract and its arguments. In order to identify the function and still being able to override functions and enable polymorphism, the identifier is a hash build from the function signature (name and parameters).

Amount This field shows the amount of bazo units send in this transaction.

3.5.2. Data from account

Issuer/Owner This field contains the public address of the account owner.

Balance This field contains the amount of coins this account owns.

Contract This field is the smart contract itself and contains the byte code. The datatype is `[]byte`, so it can easily be packed into a transaction field.

ContractVariables This field contains the state variables that are changed by executing transactions.

3.5.3. Data from miner

Block header This field is needed for block number/hash

3.6. Fee

Running a node in the network carries costs and the node operators want to be compensated. The fee is expressed in the smallest unit of bazo coins available. The cost of execution vary since depending on the complexity of the function the amount of time which the whole network is busy processing differs. Ethereum calculates the cost depending on which instructions (such as ADD known from assembler) are used. In Ethereum and Neo this fee is called gas. Bitcoin calculates the cost depending on the size of the transaction. We combine both concepts with the goal of simplifying the calculation of the execution fee.

The fee is also a way to secure the network. As mentioned before the execution must be deterministic. Using a JUMP instruction (changing the program counter of the execution) a smart contract creator could develop a smart contract function containing an endless-loop, which then he could call, causing the network to jam and not accepting new transactions because the execution doesn't come to an end. With the introduction of gas subtracted with every instruction once no more gas is available the processing of the transaction is aborted.

4. Implementation

This chapter focuses on explaining why we did what. We describe the trade-offs and decisions we've taken while building the VM, and describe them in a manner that follow up works get a better understanding of why some parts of the VM are implemented in a certain way so that they either are more secure when changing something or to help them reach the same conclusion faster.

4.1. Stack

4.1.1. Maximum stack size

Facing the concern of excess memory usage of the contract on the miner, we decided to limit the stack size to 1MB which seems to be well above what the contracts will need. We neglected using the gas amount for max storage determination because it would be just a soft limit

4.1.2. Maximum variable size

Since it is possible to easily craft denial of service attacks for the blockchain if there is no limit on the variable size we decided to set the variable size to a specific fixed amount instead of arbitrary precision.

4.1.3. Data structure

As underlying data structure of "Stack", keeping in mind to keep the code of the vm short and simple, we decided to use an array of big.Int. It was important that the datatype used for the underlying datatype was of arbitrary length because splitting up an element into multiple bytes so that it can be saved when using an array of bytes as underlying data structure would have been a lot more complex and more error-prone. Arbitrary length is implemented for example an array of jagged arrays of bytes. On the vm it was very important that the type used in the vm provided mathematical methods and also that it could store and work with values of up to 256 bit size. Only big.Int fulfilled both requirements and therefore it was choosen for the vm. Thanks to big.Int being of arbitrary length we could then use an array of big.Int as underlying data structure which also simplifies our code since we don't have to cast the values retrieved from the stack before working with them.

We neglected working with pointers because even though there are more elements created on the heap of the physical machine, it shouldn't make a difference considering the vast availability of resources on modern computers and our rather small contracts. We neglected using a simple array of bytes as the whole data structure where different datatypes are read using a different amount of bytes, as it is common when having no abstraction layer or using a jagged array of byte arrays because the could becomes a lot more readable in the vm and less conversions from big.Int to byte array are necessary. We accept the dependency on the datatype big.Int which is necessary as it implements all mathematical operations which are very important and because it is of arbitrary precision and the size of other datatypes would not have been sufficient especially for cryptographic purposes.

4.2. Virtual machine

4.2.1. Error handling

The VM does what it's told to do from the compiled bytecode. If there are any errors while executing an opCode an error value is provided. The vm pushes the error message on the Stack and halts.

Since the VM operates on copies the values will also not be changed by a faulty contract.

4.2.2. opCodes

List of Figures

Bibliography

A. Declaration of Authorship

We hereby declare that

- this bachelor thesis and the work presented in it was done by ourselves and without any assistance, except what was agreed upon with the supervisor.
- all consulted sources are clearly mentioned and cited correctly.
- no copyright-protected materials are unauthorizedly used in this work.

Ort, Datum

Marco Steiner

Ort, Datum

Ennio Meier

B. Copyright and Rights of Use Agreement

B.1. Subject of agreement

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit «TITEL DER ARBEIT EINFÜGEN» von Marco Steiner und Ennio Meier unter der Betreuung von Prof. Thomas Bocek geregelt.

B.2. Copyright

Die Urheberrechte stehen den Studenten zu.

B.3. Rights of Use

Die Ergebnisse der Arbeit dürfen sowohl von den Studenten wie von der HSR nach Abschluss der Arbeit verwendet und weiter entwickelt werden

Ort, Datum

Marco Steiner

Ort, Datum

Ennio Meier

Ort, Datum

Prof. Thomas Bocek