

國立台灣大學電機資訊學院資訊工程學系
碩士論文



Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

半中心化的區塊鏈智能合約：以太坊區塊鏈上的中心
化驗證及鏈下計算的智能合約

Semi-Centralized Blockchain Smart Contract : Smart
Contract of Centralized verification and Off-Chain
Execution on Ethereum Blockchain

林修平

Hsiu-Ping Lin

指導教授：廖世偉 博士

Advisor: Dr. Shih-Wei Liao

中華民國 106 年 7 月

July 2017



Acknowledgments

首先感謝廖世偉教授讓我有機會接觸到區塊鏈。當我一上研究所的時候，其實我還不太清楚自己未來的方向，直到我開始接觸比特幣及以太坊之後，我開始了解到去中心化的重要性、會對社會、經濟、權力結構產生的影響，並深深為此著迷，這也是我第一次這麼確定未來的計劃。

其次感謝ABCLAB的各位，水深火熱之中的夥伴，大家都過得很辛苦但還是撐過來了，感謝大陸同胞志峰、南部同胞德楷和天龍公子宜霖帶給我這麼多歡樂，當然還有論文的各種建議。

最後感謝我的家人與女友在這段期間對我的支持，很多時候都是靠著你們的鼓勵我才沒打算延畢。



摘要

部署在區塊鏈上的智能合約強調的是它的去中心化優勢。但在區塊鏈下一波的革新技術尚未成熟落地前，隨著加密貨幣日漸走紅，區塊鏈上的虛擬貨幣價值迅速攀升，造成以該鏈貨幣為計算支付單位的區塊鏈技術如以太坊面臨開發和執行成本暴增。

在這個情況下，我們提出一個半中心化的智能合約，將原本區塊鏈上去中心化的智能合約執行方式，搬至鏈下以中心化的方式來執行和驗證。目標是讓某些應用可以在不犧牲可驗證性及資料不可篡改性的前提下，藉由部分的取捨來提高執行效率和降低執行成本。

我們並非提出一個新的共識演算法或是分片技術來提高區塊鏈的每秒交易量或每秒計算量。我們提出的是一個不同的合約執行方式，讓合約使用者可以在中心化、去中心化、執行效率和每秒計算量限制之間權衡，選擇自己合約適合的執行方式。

關鍵字：區塊鏈、以太坊、Gcoin、去中心化、智能合約、鏈下計算



Abstract

Smart contract deployed on Blockchain has the advantage of decentralization. However, with the increasing popularity and surging market cap of Blockchain technology such as Ethereum, it has become much more expensive and difficult to deploy and execute a smart contract on Blockchain.

We propose a semi-centralized smart contract architecture to move the smart contract execution and verification away from the Ethereum blockchain. We hope applications can improve efficiency and save execution cost of their smart contract by balancing the trade-off between centralization and decentralization while retain computation verifiability and tamper-proof data.

We didn't propose a new consensus algorithm or sharding scheme to increase transaction-per-second or computation-per-second on Blockchain. What we proposed is a different approach to execute smart contract. User gets to decide between different property he desires such as centralization, decentralization, efficiency and computation-per-second and choose the best way to execute his smart contract.

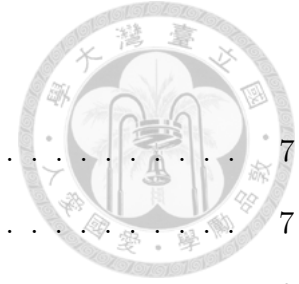
Key Words: Blockchain, Ethereum, Gcoin, Decentralization, Smart Contract, Off-Chain Computation





Contents

口試委員會審定書	i
Acknowledgments	i
摘要	ii
Abstract	iii
List of Figures	viii
List of Tables	ix
Chapter 1 Introduction	1
1.1 Research background	1
1.2 Research motivation	3
Chapter 2 Background	5
2.1 Blockchain	5
2.2 Bitcoin	6



2.3	Ethereum	7
2.3.1	Public key, Private key and Address	7
2.3.2	Gas, Gas Price and Gas Limit	8
2.3.3	Smart contract	8
2.4	Gcoin	9
2.5	Hyperledger	10
2.6	Docker	10
Chapter 3 Approach		11
3.1	Initiating Contract	13
3.2	Executing contract using Docker and submitting result	13
3.2.1	Save whole and submit	14
3.2.2	Save storage and submit	15
3.3	Gathering results and deriving final result	15
3.3.1	Random judge	16
3.4	Overall architecture	16
Chapter 4 Analysis		18
4.1	Gas cost	18
4.1.1	Batching transactions	18
4.2	Gas limit	19
Chapter 5 Conclusion		22

Chapter 6 Future Work

Bibliography



24

26



List of Figures

3.1	Basic work flow	12
3.2	Essential information stored in on-chain contract	14
3.3	Recording hash of the storage	15
3.4	Overall architecture	17
4.1	Batching requests to save gas cost.	21



List of Tables

4.1	Gas cost of different operation in on-chain contract.	19
4.2	Gas cost of basic operation.	19
4.3	Gas cost of expensive computation.	21



Chapter 1

Introduction

1.1 Research background

As Blockchain technology becomes popular and mature over the past few years, more and more applications and use cases are being experimented and tested out on different Blockchain platforms. Because some use cases such as financial settlement[4], advertisement[6], regulation[1] and others[3] that required complex execution logic, a more capable and sophisticated-designed Blockchain platform is strongly needed. And this is where Blockchain platforms with Turing-complete characteristic like Ethereum[8] comes into play.

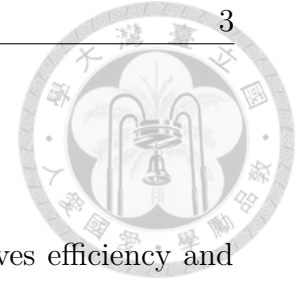
Founding members of Ethereum builds the Ethereum Virtual Machine from scratch. Programs running on Ethereum have no limits on either its program size or storage size. Compared with limited capability of Bitcoin scripts, Ethereum Virtual Machine makes Ethereum more than just a platform for store of value or simple

value transfer. Being a Turing-complete system means you can in fact simulate a computer on top of it. We can think of Ethereum as a distributed world computer.

Turing-completeness of Ethereum enables users to write smart contracts like a simple escrow, a vault or whatever business model desired. However, performance and scalability issues of Ethereum have become significant problems for both developers and users. Every node in the network has to verify transactions itself which means one's smart contract is executed not only by himself but also by rest of the network. This design has raised many concerns regarding its scalability and privacy.

Also, as public blockchains like Ethereum and Bitcoin are gaining enormous attention from investors, transaction fee continues to surge and reaches new height. As the time of writing, price of one bitcoin has tripled while price of one ether is now almost ten times higher than it was, even just three months ago. This has made executing a smart contract more costly both for developers and users. It also means that developers will have to spend more efforts on trimming the size of their program.

On the other hand, Gcoin[2] Blockchain uses a different approach to build its smart contract service. Instead of having contracts executed by every node in the network, users who want to start a smart contract select their counter parties to be the executors of their smart contracts. We will refer to these smart contract executors as validators in the following sections. All executions are recorded on Gcoin Blockchain so every interested individual can execute the contract and validate the results themselves by tracing and repeating transactions in order. This approach shifts the burden from every node to a constant number of nodes.



1.2 Research motivation

Smart contract validator architecture in Gcoin Blockchain improves efficiency and privacy by

- having only designated validators execute the smart contract and
- running the execution off-chain

Inspired by this design, we hope it can be implemented on Ethereum as well to increase processing speed of transactions, reduce on-chain computation and in the end have a overall faster and more stable platform for smart contract services.

Once we move computation off the blockchain, computation cost will drop and transaction fee cost will also decrease. In the benefit of this, we either save the cost or we can lift the transaction fee up to have our transaction committed into blockchain faster.

And that's what we propose to do - move the computation off the blockchain. But instead of using EVM to execute smart contracts like what Gcoin smart contract architecture does, We propose to use Docker to execute our smart contract. We do this for three reasons. First, in Ethereum there are only two programming languages to write a smart contract which are Solidity and Serpent. Solidity is under full development and is often updated in a rather fast pace. Serpent on the other hand has not been updated for a long time. Using Docker to execute smart contract, users get to choose which programming language they prefer to design their smart contract. This introduces a whole new space for smart contract development.

Not only can smart contract now be designed by a more mature and battle-tested programming language but also be developed in a more friendly environment where a great deal of SDK and IDE are provided. Second, Docker container provides an isolated environment for arbitrary smart contract execution. And third, it's conveniency for cross-platform development and tremendously lightweight virtual machine makes executing transactions a lot faster compared with a typical virtual machine.



Chapter 2

Background

2.1 Blockchain

Blockchain the word itself basically explains the essence which lies with in - blocks and chain. A blockchain is a series of records arranged in batches called blocks that use cryptographic method to link one to another. Each block references and identifies the previous block by a hash value created from hashing function. One after another, forms a chain, hence the name.

There are two important characteristics of blockchain: decentralization and immutability.

- Decentralization: Unlike most existing transactional systems, which require a centralized system run by a trusted authority, blockchain distributes the system of record (typically referred to as a distributed ledger. This ledger is not stored in a master location or managed by any particular entity. Instead,

it exists on thousands of computers across the world at the same time in such a way that anybody with an interest can maintain a copy of it. Decentralization makes it so that no single entity can tamper the record, commit a fraud without getting detected.

- **Immutability:** With the aid of hashing function, Blockchain can preserve the integrity of its record. Instead of altering a single data entity like traditional centralized database, old transactions are preserved forever and new transactions are added to the ledger irreversibly. Anyone on the network can check that ledger and see the same transaction history as everyone else.

Distributed ledgers hold the potential of streamlining many financial institution payment mechanisms. A blockchain would require a single payment to be sent from one institution to another, with no need for intermediary institutions that vouch for preceding institution. Compared with traditional payment system where user needs to pay every intermediary institution for its vouching service, employing a blockchain can significantly reduce the cost. It would, as another benefit, reduce the time required for an international payment from days to minutes.

2.2 Bitcoin

Introduced by whose true identity is still unknown - Satoshi Nakamoto, Bitcoin[5] is the world's first cryptocurrency whose value is not endorsed by any central bank, but is based on the perception of its users. It was introduced as a peer-to-peer electronic cash system that does not rely on a central authority to issue

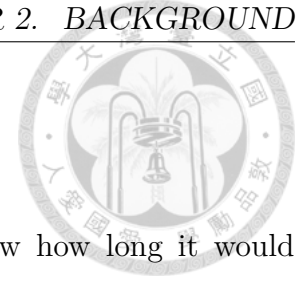
currency or authorize transactions. A decentralized network of peers can provide the infrastructure to maintain an immutable, censorship-resistant and public ledger that stores value on the network. It's also the stepping-stone of the Blockchain technology.

2.3 Ethereum

Ethereum is one of the world's most popular cryptocurrency. It was created by Vitalik Buterin, formalized by Gavin Wood and crowdfunded to kick-start development. Their vision was to introduce a global computer that can store and execute programs. Some of these programs are called smart contracts[7] as the conditions of an agreement between two or more parties are enforced using the same consensus that secures the blockchain. Similarly to Bitcoin, it relies upon a decentralized network of peers and permissionless governance to provide an immutable and censorship-resistant blockchain.

2.3.1 Public key, Private key and Address

Public key and private key come in pairs. With private key, one can sign a message and others can check the message signed against signer's public key to see if the signature is valid. Public key can be seen as one's pseudo identity, but usually it's encoded as an address which is much shorter and human-readable.



2.3.2 Gas, Gas Price and Gas Limit

As a consequence of Turing-completeness, one can not know how long it would take to run the program. It may be running forever in a for-loop. So in order to prevent users from running a program forever, each operation in Ethereum Virtual Machine is priced base on how many CPU cycles it takes and denominated in Gas. For example, writing to a storage slot costs you 20000 gas. And how fast your transaction will be part of the history depends on the gas price you choose, namely how much you pay for per unit of gas. Finally there 's the gas limit which prevents somebody rich from clogging the network by running a contract with mega for-loop.

2.3.3 Smart contract

This concept[7] was first envisioned by Nick Szabo back in 1994 and he provides the following definition:

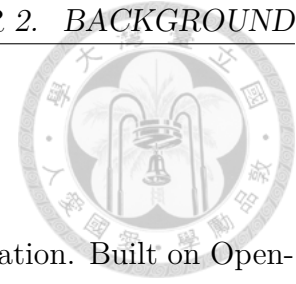
“A smart contract is a computerized transaction protocol that executes the terms of a contract. The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries. Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs.”

Smart contract is one of the important features in Ethereum. Effectively, all network nodes deterministically perform the contract 's computation using their copy of the Ethereum Virtual Machine in order to reach the same final state. This repe-

tition of computation permits the network to directly enforce the correct execution of a contract and results in publicly verifiable contracts. The smart contract code is compiled into byte code and is composed of Ethereum-specific operation-codes before it is stored in the Blockchain. Contracts are stateful and their constructor is called upon creation to set its initial state. A persistent memory area called storage is available to store the contract's data and maintain its state. Furthermore, contracts are event-driven and changing its state requires a user to invoke one of its functions.

2.4 Gcoin

Gcoin adopts permissioned blockchains and a multi-role structure that allows it to closely model real-world systems. Through the implementation of the Gcoin system, traditional financial instruments can be integrated without introducing centralized intermediary risk, so the multi-role structure can meet a variety of business needs and be used for major national public projects. Powered by blockchain technology, Gcoin system can support high frequency trading and contract versatility. Gcoin system is capable of supporting sophisticated financial market and e-commerce applications within a decentralized structure. Gcoin provides smart contract capability that is optionally extensible for the customer need. Most Bitcoin miners will accept only standard transactions while Gcoin creates more flexibility for its customers.



2.5 Hyperledger

Hyperledger is an open source project hosted by Linux Foundation. Built on Open-blockchain developed by IBM, it now has several branches in full swing. Instead of building a blockchain for all kinds of use cases, Hyperledger cooperates with leading companies in different industries to bring innovation and adequate architecture best suited for the underlying industry.

2.6 Docker

In essence, the Docker eliminates “works on my machine” problems when collaborating on code with co-workers. Using containers, everything required to make a piece of software run is packaged into isolated containers. Unlike VMs, containers do not bundle a full operating system - only libraries and settings required to make the software work are needed. This makes for efficient, lightweight, self-contained systems and guarantees that software will always run the same, regardless of where it’s deployed. Each docker image or docker container has it’s own identifier which is computed by hashing the whole image or the whole container.



Chapter 3

Approach

There are two pieces of code. One serves as a verifiable public record and is executed on-chain. The other implements the execution logic of user's contract and is executed off-chain. We will use the term "off-chain smart contract" to refer to the smart contract program being executed off-chain and the term "on-chain smart contract" to refer to the smart contract deployed and executed on Ethereum. An off-chain smart contract is executed in a docker container while an on-chain smart contract is executed in Ethereum Virtual Machine. Figure 3.1 illustrate a basic work flow.

In short, there is a user who wants to set up a contract and there are validators who will execute the contract and provide result. We use the on-chain smart contract to record some basic and related information of the contract. If validators agree to participate, they will send a transaction to the on-chain smart contract to confirm. After it's all set, user submits his request and validator execute the con-

tract according to the request. A period in which all validators execute according to a request is called one round. In each round, user will first send a transaction to the on-chain smart contract to submit his request. Then validators will fetch the request and execute the off-chain smart contract using Docker container. After that, validators submit the result back to the on-chain smart contract. Finally, after all results are submitted or after a predefined amount of time had passed, final result is determined by majority of the results and signify the end of this round. In each following section, we will have a walkthrough on each step along with an example.

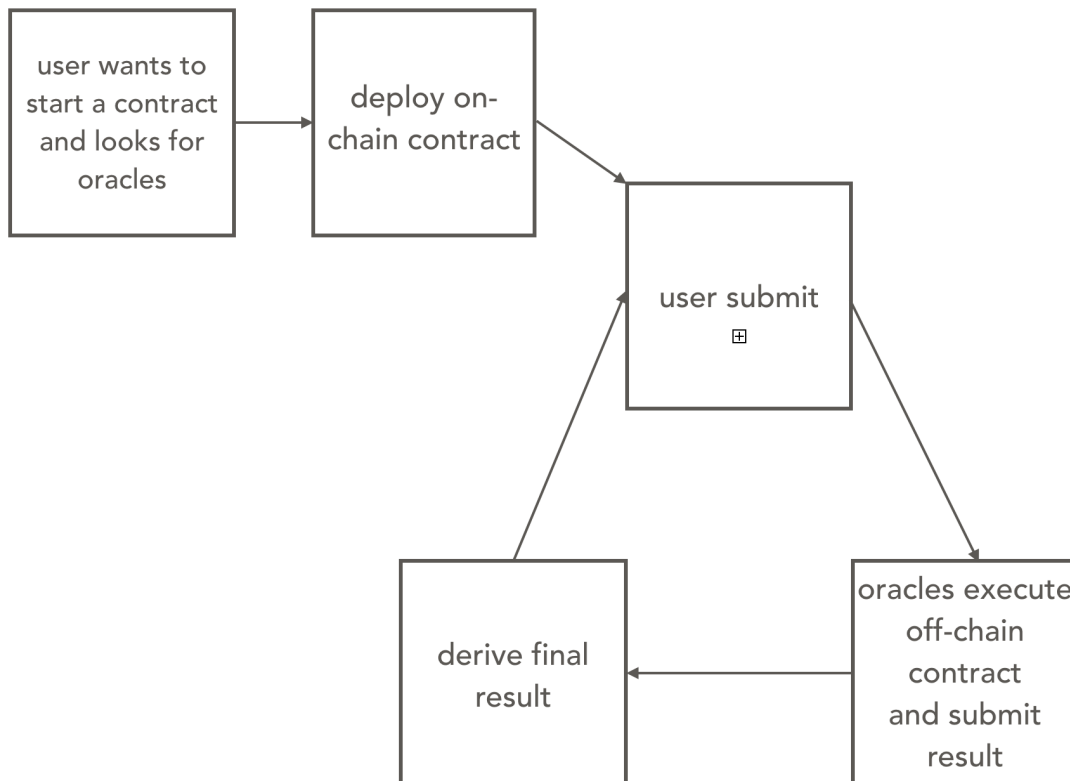
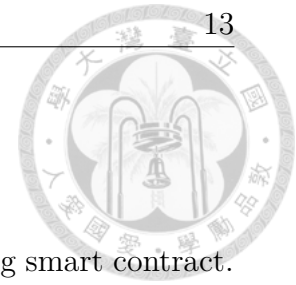


Figure 3.1: Basic work flow




3.1 Initiating Contract

We can begin with an example of a user trying to make a will using smart contract. He first looks for related entities like friends, banks or representatives from Government and ask them to be the executors of his will. And of course user himself can be an executor as well. After the deal is made, together user and executors come up with one off-chain smart contract and one on-chain smart contract. The on-chain smart contract is deployed onto Ethereum blockchain with their public keys included in the contract. We also record a hash value as the identifier of docker image of the off-chain smart contract. We will be calling the executors using the term “validators” . See line No.2 to line No.4 in Fig. 3.2.

And we need to record every request from user and every execution result from validators. See line No.6 to line No.12 in Fig. 3.2. The variable “currentRound” is used to keep track of which round of execution are we at.

3.2 Executing contract using Docker and submitting result

Validators should have their blockchain client software watching for events triggered by the on-chain smart contract. Every time user submits a request, validators will be notified. Then they fetch the request and start the docker image up with the request as input. There are two different approaches to execute the off-chain smart contract. First one is saving the whole docker container as a new image after execution and



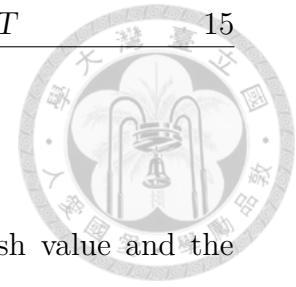
```
1 contract on_chain_contract {
2     address user;
3     address[] oracles;
4     bytes32 imageHash;
5
6     uint currentRound;
7     string[] requests;
8     struct result {
9         mapping(address => bytes32) from;
10    }
11    result[] results;
12    bytes32[] finalResults;
13
14
15    function submitRequest(string _request) onlyUser { ... }
16    function submitResult(bytes32 _result) onlyOracles { ... }
17 }
```

Figure 3.2: Essential information stored in on-chain contract

submits the image identifier as result. Second one is updating the storage of the off-chain contract, hashing the storage and submits the hash value as result.

3.2.1 Save whole and submit

From the on-chain contract, validator fetches the new request and the final result from last round. Then uses the final result as an identifier to retrieve the docker image from remote repository. Then he executes the off-chain smart contract along with the request as input, saves the whole container as a new image and submits the identifier of the new image as result. This approach has the advantage of easy implementation, but at the cost of creating new image on every request.



3.2.2 Save storage and submit

From the on-chain contract, validator first fetches the imageHash value and the final result from last round. Then he respectively retrieves the docker image using imageHash value and the storage using final result as an identifier from remote repository. In this case, we need to store the hash of storage in on-chain smart contract. See Fig. 3.3

Then validator executes the off-chain smart contract along with the request as input. After the execution, he saves the storage and submits the hash of the storage as result. This approach has the advantage of a reusable image which could potentially spare every participants enormous amount of space.

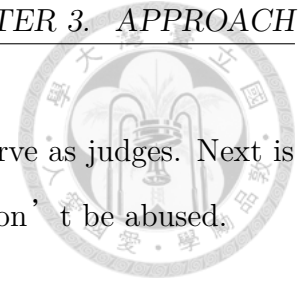
```
address user;  
address[] oracles;  
bytes32 imageHash;  
bytes32 storageHash;
```

Figure 3.3: Recording hash of the storage

3.3 Gathering results and deriving final result

After validators have submitted their result respectively, the final result is decided by majority result. This approach is rather intuitive and simple, but problem will rise if there' s a disagreement on the final result. One way to handle the disagreement is that we make the contract open to public and allow anybody to run the off-chain

contract and verify the result. This way, the outsiders can serve as judges. Next is to provide with right incentives so that the judicial system won't be abused.



3.3.1 Random judge

First user must provide an amount of money as bounty for outsiders to verify the results. And outsiders also have to pay a certain amount of money as a stake in order to join and verify. The stake paid by outsiders will be confiscated if they misbehave or present with wrong results. And if no other results are submitted after a given amount of time the bounty goes to the judge. But the trade-off here is that no privacy is preserved for user's contract.

3.4 Overall architecture

Figure 3.4 shows the overall architecture of our design. First, there are on-chain execution and off-chain execution and two types of role, user and validator. Second, validator will be watching for updates from their server. Every time user submits a new request validator will be notified. Then validator will fetch the request and execute the off-chain contract with input from request. Finally, validators submit their result back to on-chain contract and derive a final result.

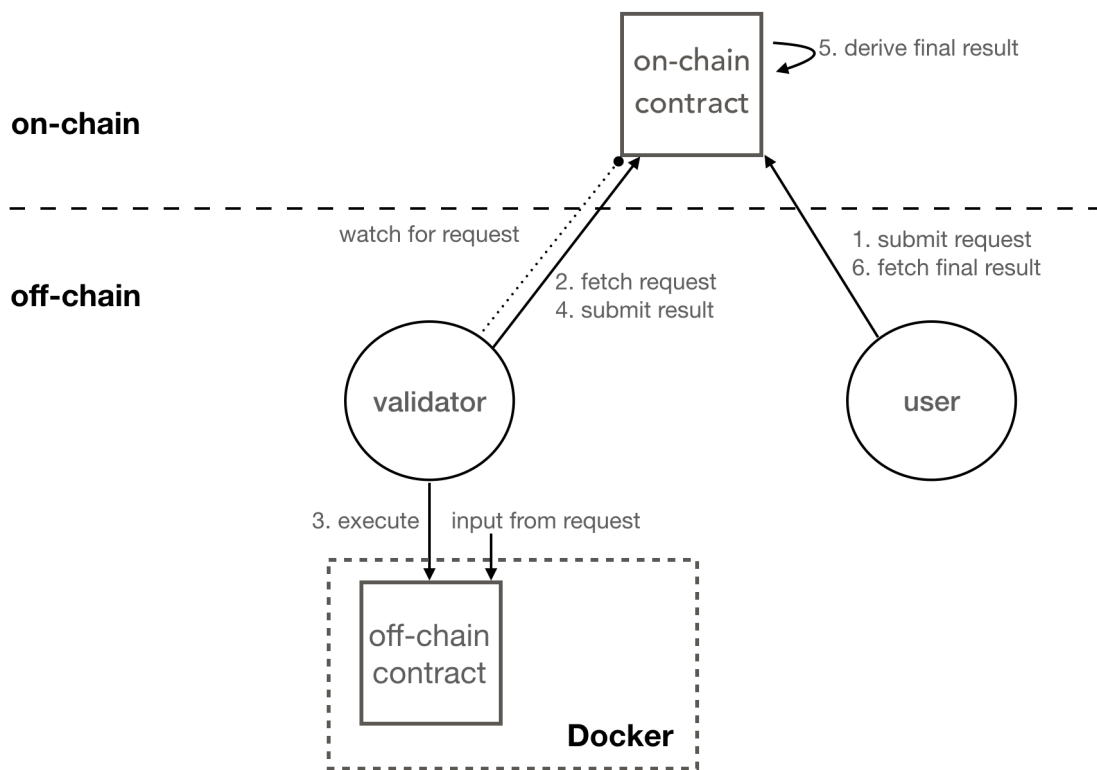


Figure 3.4: Overall architecture



Chapter 4

Analysis

4.1 Gas cost

One of the key features that makes our design advantageous is the gas cost saving. The most expensive operation in Ethereum Virtual Machine is the storage operation. And most operations in our on-chain contract is exactly storage operation and this is inevitable because we need to record essential informations on-chain.

As shown in Table 4.1, gas cost for user to submit his request is around 60000 and gas cost for validators to submit their result is around 80000. Note that there could be a slight improvement on gas cost of result submitting as shown below.

4.1.1 Batching transactions

Note that this technique is not related to gas cost saving by the off-chain execution. It's simply a way to reduce gas cost regarding on-chain transactions.



Table 4.1: Gas cost of different operation in on-chain contract.

Operation	avg. gas cost
Deploy contract	1019197
Submit request	63691
Submit result	82307

Table 4.2: Gas cost of basic operation.

Operation	avg. gas cost
basic cost/tx	21000
SSTORE	20000
ecrecover	5176

The idea is that we can batch these results along with validators' signatures together out-of-band and send in a single transaction and have these signatures verified on-chain.

In Table 4.2, we can see that gas cost of verifying elliptic signature using a library is around 5000, and basic gas cost of launching one transaction is around 21000, hence we can save about 16000 gas per validator. See Figure 4.1.

4.2 Gas limit

Besides the gas cost saving, there is another key advantage in our design - no gas limit. Total off-chain computation is not bounded by block gas limit of the un-

derlying blockchain. This is useful when execution of your contract require huge computation resource. For example, frequently updating a bunch of storage data when you are a token issuer. You either exceed the gas limit or divide the routine into sub-routine and pay a significant amount of money in extra. Another example is expensive computation like RSA verification, zero knowledge proof computation or any state-of-the-art cryptographic technologies. See Table 4.3.

But there's a potential problem hiding behind this advantage. Once the block gas limit is removed, validators will be facing the unsolvable "halting problem". Either caused by a naive mistake in a loosely verified program or maliciously fabricated program. Validators will be wasting possibly infinite amount of computation and time on a never-ending execution of the off-chain contract.

This could be mitigated in two way.

- One quick and easy solution is to put a cap on time used in each execution, but then it would be just another synonym of block gas limit.
- A better solution is to statically analyze the code of the off-chain smart contract beforehand.

This would of course requires the help of the still developing technology. Something worth noting is that Hyperledger is confronted with the same problem and they try to work it out in another way in which they introduce a role called 'endorser' to endorse user's transaction. The solution somewhat relies on trusting the endorsers and hence further increases the potential centralization risk.



Table 4.3: Gas cost of expensive computation.

Operation	avg. cost
SHA-512	200000gas
EC addition	30000 CPU cycles
EC multiply	940000 CPU cycles
EC pairing(k points)	$60000 * k + 4000$ gas

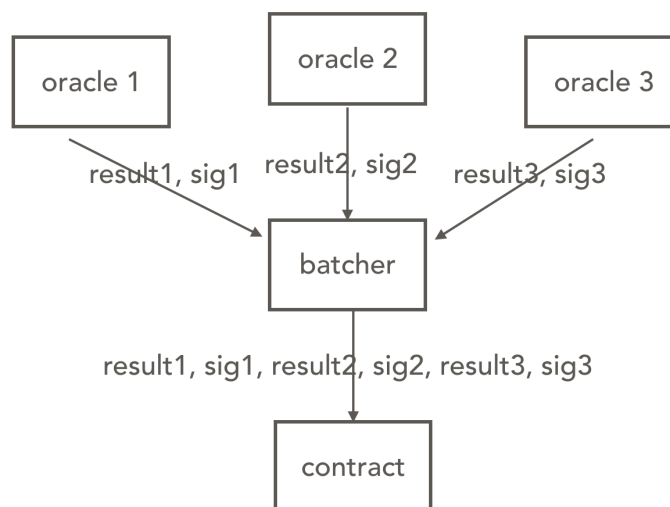


Figure 4.1: Batching requests to save gas cost.



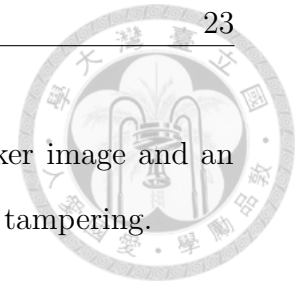
Chapter 5

Conclusion

It's not cheap to execute smart contract on a public blockchain like Ethereum and doing so is also met with the problem of low throughput because executions per second are bounded by block gas limit. Despite the problems mentioned, community has long been yearning for using the existing programming language like JAVA, C or GO to develop their smart contract.

Hence we propose the idea of moving smart contract execution away from blockchain and into a Docker container. We can benefit from the many great features Docker provided which includes portability and fast deployment across different host machines. Also we don't have to write our smart contract using limited tools provided by the underlying blockchain ecosystem, but with other more well developed programming languages.

In our design, the on-chain smart contract only stores user requests and the hash of docker images or the hash of storage depending on which approach in chapter



3 you implement. The hash value is used as an identifier of docker image and an integrity check of the image in case of incomplete transmission or tampering.

These gains do not come free, however, but at the cost of

- potential threat of non-stopping execution by not carefully designed or ill-intended program
- centralization risk

Blockchain technology starts from Bitcoin as a digital currency with scripts limited in capability, to all kinds of altcoins such as Gcoin which stands out with its innovation such as multi-color-currency, to Ethereum as a platform capable of running Turing-complete program.

People start with some simple contracts like Lottery or Casino. Further there's insurance or prediction market built as a smart contract and even a Decentralized Automated Organization (DAO) which took away all the attention in June 2016.

As people build bigger and fancier smart contracts, size and complexity of the code also grows significantly. In spite of many innovations and works done on scalability and privacy issues, there's no sign of any perfect solution in the short term. We envision eventually these issues will be solved. But still what we propose is, with some tradeoff, one can get rid off these shackles and focus on building more complex platform and contract at the moment.



Chapter 6

Future Work

There are a few challenges to the current implementation. First one is the privacy of smart contract. Since it's fair to assume that what's put on the blockchain is no secret at all, we should not assume the hash of docker images or the hash of storages recorded in the on-chain smart contract will not lead outsiders to the actual image or storage. One circumvention is to restrict the access to these resources.

However, restricting the access to docker images or storages means keeping these resources private and therefore pose a centralization risk of losing these resources due to datacenter or connection failure.

Next is the big challenge of verifying correctness of the off-chain contract. There are several ways to improve the verification process. For example, designing off-chain smart contract in functional programming languages like OCaml to make it more suitable for doing a formal verification on the program. Static analysis of program code helps making sure that the program executes as intended and searching for



potential loop hole like reentrancy problem.

There is also the need for light client friendliness. If no light client techniques implemented, every participant will need to become a full node and retrieve the docker images and the storage in order to verify the results. Thus enabling clients with light-weight device to engage with smart contract and expand the coverage of users in our design.



Bibliography

- [1] J. S. CERMEO. Blockchain in financial services: Regulatory landscape and future challenges for its commercial application, 2016. [Online; accessed 20-July-2017].
- [2] Digi. Gcoin white paper, 2016. [Online; accessed 22-July-2017].
- [3] A. T. Don Tapscott. Realizing the potential of blockchain - a multistakeholder approach to the stewardship of blockchain and cryptocurrencies, 2017. [Online; accessed 20-July-2017].
- [4] M. Hearn. Corda: A distributed ledger, 2016. [Online; accessed 20-July-2017].
- [5] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. [Online; accessed 22-July-2017].
- [6] B. Software. Basic attention token (bat) - blockchain based digital advertising, 2017. [Online; accessed 20-July-2017].
- [7] N. J. Szabo. Smart contracts, 1994. [Online; accessed 22-July-2017].
- [8] G. Wood. Ethereum: A secure decentralised generalised transaction ledger, 2015. [Online; accessed 20-July-2017].