Smart Bitcoin Cash: a Bitcoin Cash Sidechain with EVM & Web3 Compatibility

Abstract

While Bitcoin Cash aims to provide a decentralized, high-throughput, low-cost, and easy-to-use infrastructure for cryptocurrency; any changes to the mainnet require a high level of consensus, which hinders the trial-and-error process.

So, we decided to develop Smart Bitcoin Cash - a sidechain for Bitcoin Cash with an aim to explore new ideas and unlock possibilities. It will be compatible with Ethereum's EVM and Web3 API, for they are the de facto standards for blockchain DApps nowadays.

Ethereum is solving the issues related to low-throughput and high-cost by switching to ETH2.0, which, as we all know, still requires years of development to finish. Smart Bitcoin Cash attempts to tackle these issues differently, by optimizing the implementation of EVM and Web3 at a low-level in order to fully leverage the potential of hardware, especially its inherent parallelism. We believe that Smart Bitcoin Cash will provide the same benefits of ETH2.0 in a much shorter time.

Motivation

Different people want different new features from Bitcoin Cash.

Bitcoin Cash's block interval remains 10 minutes, which is too long by comparison, as there are other chains offering intervals of seconds. Although Bitcoin Cash supports secure zero-confirmation transactions, complex scenarios besides payments need short confirmation times for better user experience, such as DeFi.

Bitcoin Cash has a limited script system which is not Turing-complete, making it more difficult to use than Ethereum's EVM. Also, it is less capable than EVM and Solidity, as they offer the best ecosystem and encompass the most programmers among all the smart contract platforms. It's a pity they cannot be utilized in Bitcoin Cash's ecosystem.

Currently, Bitcoin Cash's capability has been proven to reach 14MB. Although at this moment its block size is only 0.8 MB on average, it is growing fast: more than three times since the beginning of 2021. If it keeps growing at such a rate, 14MB would not be enough in the near future. Since issues surrounding block size larger than 14MB remain unknown and have not been field-tested, it would be best to begin getting prepared for further scaling of Bitcoin Cash's throughput.

What can Smart Bitcoin Cash contribute to the Bitcoin Cash ecosystem? Well, it builds a new playground for testing new features: It has a short confirmation time, supports EVM and Web3 with novel optimizations to provide higher throughput, and also provides new channels for one to invite more users to join Bitcoin Cash's ecosystem.

As Smart Bitcoin Cash grows more mature, the developed libraries and the learned lessons will help improve Bitcoin Cash's mainnet, too.

Background

It has been 8 years since Vitalik Buterin proposed Ethereum in 2013. Smart contracts were born, bred, and have been blooming ever since. Ethereum is now the most successful smart contract platform, and we may examine its ecosystem as well as make observations from it:

The single chain's user experience is hard to approach through sharding or layer-2 solutions. Zero latency and atomic interoperation between smart contracts is only possible within the same chain. The cross-shard or layer-2-to-layer-1 interoperation must undergo an interchain communication process, which prompts latencies similar to depositing and withdrawing at centralized exchanges. Some popular mechanisms, such as flash loan and flash swap, would not work cross-chain.

Low-throughput deters ordinary users from interacting directly with DApps. The gas upper bound is fixed for each block, and miners pack the transactions with high gas fees first. Meanwhile, transactions with low gas fees must wait a long time to be packed into a block or never get packed at all. Naturally, only high-value transactions are worth a high gas fee. So, ordinary users who cannot afford high gas fees can only deposit their funds into centralized organizations, then delegate them to operate their funds. In the DeFi world, only a small number of accounts are sending transactions. These accounts have a lot of funds, either because they own a lot, or because they gather a lot from ordinary users. Ironically, today's decentralized finance is not so decentralized.

Storage costs more resources than computation when executing smart contracts. In the history of Ethereum, the storage's gas cost has been increased by two EIPs: EIP-1884 and EIP-2200. But that is not enough, as Research shows that Ethereum still underestimates certain storage operations, making it susceptible to DoS attacks. So, another EIP, EIP-2929, is about to increase the gas of high storage operations again in the upcoming Berlin hard fork. At the same time, the 256-bit arithmetic of EVM is speeded up substantially by new libraries from Martin Holst Swende and Paweł Bylica.

Off-chain QPS (queries per second) is as important as on-chain TPS (transactions per second). A DApp works by not only sending a transaction for execution but also querying the latest on-chain state and the chain's historical events. Though a transaction is executed only once, the corresponding events and state changes may be queried many times. So the total requirements for QPS are much higher than TPS. In Ethereum's ecosystem, the queries are made through Web3 API, the largest provider of which is Infura. Infura maintains an optimized Web3 implementation, which is better than standard full node clients, such as go-ethereum, but is not open-sourced. Many developers choose to use Infura's low-cost service instead of running their own full nodes. Consequently, if Infura experiences a severe service interruption, many DApps and exchanges would fail to work.

Users are quite tolerant of transaction latency and do not pay much attention to the exact order of transactions. Usually, we use a wallet such as MetaMask to sign transactions, for which a proper gas price is chosen before broadcast. To save gas fees, we often choose a lower gas price and expect the transaction to be confirmed in minutes or even hours, instead of in the next block. In some cases, some other transactions may be packed before yours, resulting in losses such as larger slippage. But most users can live with that.

We also observed that, since 2013, the most important trend in computers is **adding more cores to CPUs**. Let's compare a MacBook in 2013 and in 2021, and predict how it will be like in 2029:

	MacBook Pro in 2013	**MacBook Pro in 2021**	**MacBook Pro in 2029**
# CPU cores	2	8	32
Highest CPU Frequency	2.9GHz	3.1GHz	3.3GHz
Lithography for CPU	22nm	5nm	1nm

Integrated circuit technology can hardly boost frequency further after 28nm; however, it does provide more transistor budget in the new generations. Designers may use these transistors to implement more and more CPU cores. In the last decade, surrounding all the novel programming languages hype, is the easy leveraging of the potential of more CPU cores: channels and goroutines of Go, isolates of Dart, and fearless concurrency of Rust.

The above observations shall guide the design and implementation of Smart Bitcoin Cash.

Core Components of Smart Bitcoin Cash

Smart Bitcoin Cash's innovation lies in libraries. Instead of inventing fancy consensus and cryptographic algorithms, we decided to adopt another methodology: to develop low-level libraries with an aim to fully uncover the hardware's potential, especially its inherent parallelism. Ordinary users and developers are provided with a compatibility layer supporting EVM and Web3, so the optimized low-level "close to the metal" libraries themselves remain concealed by this layer of abstraction. During the implementation, we used the codename "Moeing", which is added to the libraries' names as prefix.

With these powerful components, Smart Bitcoin Cash aims to enlarge the gas consumption every 15 seconds to one billion gas in the medium term. In the long run, overall throughput will be boosted further by adopting sharding and rollups.

MoeingEVM

MoeingEVM is a parallelized execution engine that currently manages multiple EVM contexts and executes multiple transactions. Based on an optimized EVM implementation from evmone, it can be observed that there are several novel techniques adopted to maximize transaction parallelism.

As multi-core CPUs become more and more popular, scalability would be hindered in the context of Ethereum's single-thread execution semantics, because under its constraint speeding up is very hard. But if we switch to multi-thread execution semantics, a better overall result would be achieved as multi-core CPUs can be utilized more easily and in a more straightforward manner.

So MoeingEVM is developed following the multi-thread execution semantics.

To fully utilize the inherent parallelism in modern hardware, we attempt to leverage two kinds of parallelism:

1. The parallelism between consensus engine and transaction execution engine.
2. The parallelism among different transactions.

To make consensus engine and transaction execution engine work concurrently, MoeingEVM uses such a scheme: when a block is committed, the transactions in it would not be executed immediately. Instead, these transactions would be saved as a part of the world state. After they are saved, the Merkle root of the world state would be calculated and the next block will be proposed and determined; meanwhile, the saved transactions will be examined and executed.

To make EVMs run in parallel, we allow transactions from several blocks to be mixed and reordered. In each round, a bundle of independent transactions is picked and executed in parallel to achieve a higher degree of concurrency. After several rounds, all or part of the saved transactions get executed, whereas the remaining unexecuted transactions are saved back to the world state for later execution. Since transactions are always enforced into bundles and each bundle is executed in parallel, this scheme is named "**enforced-bundle parallelism** (EBP)".

EIP-2930 makes a transaction explicitly containing a list of addresses and storage keys that it plans to access. This list helps analyze the interdependence of transactions. In the future, MoeingEVM will utilize it to further boost the parallelism of transaction execution.

MoeingADS

Why are storage operations so expensive? Well, Ethereum's storage engine MPT is absolutely the root cause.

Is it okay to skip MPT? Yes, many blockchains (including Bitcoin Cash) work well without it. However, as an authenticated data structure capable of proving what states do or do not exist in the world state, it remains very important for trustlessness and is the cornerstone for light clients and chain-crossing.

So we move forward to develop MoeingADS — another authenticated data structure capable of replacing MPT.

Ethereum's storage engine has a two-layer architecture. The first is LevelDB and the second is MPT. On the other hand, blockchains such as Bitcoin Cash adopt a single-layer architecture for storage — using LevelDB to store UTXOs directly. MPT works on top of LevelDB to be an authenticated data structure, at the cost of a lower read & write throughput. Every time the EVM reads or writes the world state, MPT must perform several LevelDB operations, prompting multiple operations on the SSD, resulting in the slowness of MPT.

MoeingADS uses a single-layer architecture, accessing the file system directly without having to use any other databases. It is a KV database that can provide existence and non-existence proof. With MoeingADS, reading a KV pair requires one read to disk, overwriting a KV pair requires one read and one write, inserting requires two reads and two writes, and deleting requires two reads and one write. What's more, the writes are appending, which is very SSD-friendly.

Experiments show that MoeingADS is even faster than LevelDB. The cost is the larger consumption of DRAM: each key-value pair demands about 16 bytes.

MoeingDB

Besides supporting MPT, LevelDB is also commonly used to store historical data such as blocks, transaction receipts, and logs. However, it is not optimized for blockchain workloads, which carry the following characteristics:

1. Read volume is much larger than write volume.
2. No need to support Read-Modify-Write atomic transactions.
3. Simple read/write locks are better than MVCC, as modifications are large batched writes for blocks.
4. Poor spatial locality for efficient caching, which results from most keys being Hash IDs.
5. Susceptible to DDoS attack, unless cold data's read latency has a reasonable upper bound.

MoeingDB is an application-specific database that stores blockchain history, and we developed it with the above characteristics in mind to suit blockchain's workload best. Based on its features, an open-source high QPS Web3 API can be built, benefiting both Smart Bitcoin Cash and Ethereum. We hope that it will facilitate the Web3 API provider market to be more decentralized.

MoeingKV

MoeingKV is a KV storage much faster than LevelDB in reading and writing, at the cost of removing iteration support.

To support iterators, LevelDB involves a lot of trade-offs and optimizations. But in most cases, blockchain storage engines can work without iterators, with examples like Ethereum's MPT and Bitcoin Cash's UTXO storage.

In underlying data structure design and code implementation, MoeingKV produces trade-offs and optimizations to speed up normal read and write operations. So, it can replace LevelDB as a better "first layer" for MPT.

While MPT cannot be replaced by MoeingADS due to compatibility, one can use MoeingKV to support MPT. MPT backed by MoeingKV might not be as fast as MoeingADS, but it is much faster than one backed by LevelDB.

Though MoeingKV is not used in Smart Bitcoin Cash, its key ideas come from MoeingADS and MoeingDB. We particularly hope other projects may benefit from MoeingKV.

MoeingAOT

MoeingAOT is an ahead-of-time compiler for EVM.

EVM, which is adopted more commonly than other VMs like WebAssembly, is a de facto standard for smart contracts.

However, EVM lacks certain important speedup methods, such as ahead-of-time (AOT) compilers and just-in-time (JIT) compilers. In the software industry, almost every important VM has its AOT and/or JIT compilers, for example, JVM, ART VM, Javascript V8, DartVM, WebAssembly, LuaJIT, and GraalVM. We believe that it is about time that EVM has its compiler. And implementing an AOT compiler for it would be reasonable, since, unlike Javascript and Lua, it has static semantics.

MoeingAOT can compile EVM bytecode into native code, which would consequently be saved as a dynamically linked library. When the EVM interpreter starts running a smart contract and finds its corresponding compiled library file, the library will be loaded and run, and bytecode interpretation wouldn't be necessary.

For the frequently used contracts, such as USDT and UniSwap, ahead-of-time compilation is valuable because native code is much faster than interpretation, which reduces the execution time drastically.

MoeingRollup

Rollup is an offload methodology to scale up a chain's throughput. Different projects, such as optimistic rollup and arbitrum rollup, have different implementations. Generally speaking, rollup means rolling up a whole set of states into one commitment, meaning one state root. Usually, one rollup extension resides inside a smart contract and a sequencer maintains its states, packs users' transactions into blocks, and submits the respective state roots into the smart contract. The transition between the state roots of two adjacent blocks can be validated with some proof data.

An honest sequencer must reliably maintain the states and neutrally packs users' transactions, which means without any censorship. It must also provide the states and blocks to anyone in need; otherwise, users may evict it and find a new replacement using some staking mechanism predefined in the smart contract.

Furthermore, the submitted state roots sequence must be linked with valid transitions between adjacent roots. If an invalid one is located, the sequencer may be challenged to provide the proof data. And when the sequencer fails to do so, he will be slashed and evicted.

The detailed running rules are defined in smart contracts and may vary in different rollup extensions. Anyway, their common purpose is to prove valid state transition with proof data. Unfortunately, the proving task is quite heavy and hard to implement in EVM.

MoeingRollup implements this proving task natively. Using a primitive for smart contracts, all the rollup extensions can perform the proving tasks easily and efficiently. Proof data are required to encompass these three parts:

1. A block of transactions
2. The transactions' input KV pairs and their existence proof against starting state root
3. The transactions' output KV pairs and miscellaneous data for calculating the ending state root

At the same time, MoeingRollup also eases the sequencer's job by providing utilities, including proof data generation.

MoeingLink

Smart Bitcoin Cash will start as a single-shard chain. But in the long run, it is possible to include more shards and transform them into a multi-shard chain.

MoeingLink is a protocol enabling different shards to interact directly without executing any transactions on Bitcoin Cash's mainnet.

Currently, all major sharding solutions require an intermediate chain. In ETH2.0, it's the beacon chain, and in Polkadot, it's the relay chain. As more and more shards are created, the inter-shard transactions will produce a huge pressure on the crowded layer-1.

To avoid that, MoeingLink allows shards to prove self-state to others, utilizing MoeingADS's state roots committed on layer-1. Once they have acknowledged each other's state, they can interact directly without the help of layer-1.

The Consensus Algorithm

Smart Bitcoin Cash adopts tendermint as its consensus engine. The quorum of validators are elected by both hash power and BCH owners, and they take on duties in epochs.

An epoch contains 2,016 blocks (takes about two weeks). During an epoch, BCH owners prove their ownerships of time-locked UTXOs and use the values of these UTXO to vote for a validator; whereas mining pools use coinbase transactions to vote. This is a hybrid consensus model: proof of hash power and stakes. The voting process is performed on Bitcoin Cash's mainnet and totally permissionless because a new validator only needs endorsements from miners and/or holders.

An epoch's end time is the largest timestamp of its blocks, and its duration time is the difference between the end times of adjacent epochs. The quorum elected during an epoch will stay in a stand-by state for about 5% of the epoch's duration time. Then it takes its turn to be on duty, until the next quorum leaves its stand-by state, which is necessary because any Bitcoin Cash reorganization may alter the blocks in an epoch.

Each validator must pledge some BCH as collateral, which would be slashed should it misbehaves during its duty.

At the first phase after Smart Bitcoin Cash's launch, only hash power is used for electing validators. Locking BCH at mainnet for staking will be implemented later and take effect in a future hard fork.

Token and Gas

Smart Bitcoin Cash will not introduce new tokens. Its native token is BCH, and its gas fees are paid in BCH.

At the end of a quorum's tenure, half of the collected gas fees will be rewarded to the validators and the other half will be burned. In this sense, BCH will become a deflationary currency. A validator must pledge enough collateral to get its gas fee reward, and the reward must undergo a lock period before it can be spent.

The buyback-and-burn mechanism is very common for exchange tokens (BNB, HT, FTT, OKB, etc) and DeFi governance tokens. And Filecoin has a similar mechanism to burn part of the gas fees, which will be followed by Ethereum in EIP-1559. This mechanism has proven effective, so we decided to use it.

BCH can be transferred bidirectionally between Bitcoin Cash's mainnet and Smart Bitcoin Cash, which means we can lock certain coins on the mainnet, and unlock the same amount of coins on Smart Bitcoin Cash, and vice versa. To bootstrap Smart Bitcoin Cash, we are inviting the major players in Bitcoin Cash's ecosystem to run a federated two-way pegged gateway, which bridges

the mainnet and Smart Bitcoin Cash to transfer BCH bidirectionally, just like how RSK and Liquid work. These players are not necessarily validators.

Nowadays the cryptocurrency communities have adapted to the "Same Symbol on Different Chains" scheme. For example, when mentioning USDT, it can refer a token on Bitcoin's Omni, Bitcoin Cash's SLP, Ethereum, Tron, etc. So we do not use another symbol to refer the BCH on Smart Bitcoin Cash, to avoid misunderstanding.

We are aware that Bitcoin Cash's scripting language is capable of implementing a non-custodial trustless gateway by using a lock script to trace the voting process carried out inside coinbase transactions. However, this scheme has not been field-proven. We will write up dedicated proposals to describe this scheme and it will be implemented in CashScript upon passing. Afterwards, Smart Bitcoin Cash will switch to this new scheme in a hard fork.

Interoperation with other Layer-2 solutions on Bitcoin Cash

There are many layer-2 extensions on Bitcoin Cash to allow issuers to mint fungible and non-fungible tokens. Among them, the most successful and important one is Simple Ledger Protocol (SLP). In an SLP token's ecosystem, its issuer plays a central role, who can be helpful in the transfer of tokens across Smart Bitcoin Cash.

For example, if Alice wants to transfer 10 XYZ coins from SLP to Smart Bitcoin Cash, she can send the coins to XYZ's issuer using SLP, then XYZ's issuer will send her 10 coins on Smart Bitcoin Cash, and vice versa. To step up security for this process, Alice can use an atomic swap to ensure the transactions on each side both happen or neither happen.

Roadmap

MoeingADS, MoeingEVM and MoeingDB are almost finished. Regardless, some throughput tests are necessary before Smart Bitcoin Cash can be officially launched.

MoeingAOT will be ready and take effect after a hard fork by the end of 2021. MoeingKV will also be developed in 2021 with the hope to meet potential demands from Bitcoin Cash's mainnet.

MoeingRollup and MoeingLink will be developed in 2022. By then, if the traffic of Smart Bitcoin Cash is congested, they will be deployed in a hard fork for further scaling.

Conclusion

Smart Bitcoin Cash provides an EVM & Web3 compatible sidechain for Bitcoin Cash, staking its hash power while utilizing BCH as gas. What’s more, by incorporating hardware-friendly components, scalability is unlocked. We believe that it will provide the same benefits of ETH2.0 in a much shorter time, achieving a block gas limit of one billion.

The Smart Bitcoin Cash, to a large extent, can be viewed as a demo and an experiment of the novel and aggressive techniques we have been developing, which aim to optimize storage and execution engines for extreme throughput. Just like other open-source projects, there might be bugs and vulnerabilities in its design and implementation. So please be aware of the potential risks and make sure that possible losses are affordable when transferring your assets (including BCH) onto the Smart Bitcoin Cash.