Monad for developers

Monad is an Ethereum-compatible Layer-1 blockchain with 10,000 tps of throughput, 1-second block times, and single-slot finality.

Monad's implementation of the Ethereum Virtual Machine complies with theShanghai fork ; simulation of historical Ethereum transactions with the Monad execution environment produces identical outcomes. Monad also offers full Ethereum RPC compatibility so that users can interact with Monad using familiar tools like MetaMask and Etherscan.

Monad accomplishes these performance improvements through the introduction of several major innovations:

- MonadBFT
- (pipelined HotStuff consensus with additional research improvements),
- Deferred Execution
- (pipelining between consensus and execution to significantly increase the execution budget)
- Parallel Execution
- MonadDb
- (high-performance state backend)
-

Although Monad features parallel execution and pipelining, it's important to note that blocks in Monad are linear, and transactions are linearly ordered within each block.

Transaction format

Address space matches Ethereum 20-byte addresses using ECDSA Transaction formatmatches Ethereum complies withEIP-2718 , encoded withRLP . Access lists (EIP-2930) are supported but not required. Wallet compatibility Monad is compatible with standard Ethereum wallets such as Metamask. The only change required is to alter the RPC URL and ChainId.

Smart contracts

- Monad supports EVM bytecode, and is bytecode-equivalent to Ethereum. Albpcodes
- (as of the Shanghai fork) are supported.
-

Consensus

Sybil resistance mechanism Proof-of-Stake (PoS) Delegation Allowed (in-protocol) Consensus mechanism and pipelining MonadBFT is a leader-based algorithm for reaching agreement about transaction order and inclusion under partially synchronous conditions. Broadly characterized, it is a derivative of HotStuff with additional research improvements.

MonadBFT is a pipelined 2-phase BFT algorithm with linear communication overhead in the common case. As in most BFT algorithms, communication proceeds in phases. At each phase, the leader sends a signed message to the voters, who send back signed responses. Pipelining allows the quorum certificate (QC) or timeout certificate (TC) for blockk to piggyback on the proposal for blockk+1. Timeouts incur quadratic messaging. Block time 1 second Finality Single-slot. Once 2/3 of the stake weight has voted YES on a block proposal, it is finalized. Mempool There is amempool . Transactions are erasure-coded and communicated using a broadcast tree for efficiency. Spam resistance Users pay for inclusion in blocks ("carriage cost ") and transaction execution ("execution cost"). Consensus participants Direct consensus participants vote on block proposals and serve as leaders. To serve as a direct participant, a node must have at leastMinStake staked and be in the topMaxConsensusNodes participants by stake weight. These parameters are set in code. Transaction hashing For efficiency, block proposals refer to transactionsby hash only . If a node does not have a transaction, it will request the transaction by hash from a neighbor. Deferred execution and carriage costs In Monad, consensus and execution occur in a pipelined fashion. Nodes come to consensus on the official transaction orderprior to executing that ordering (Deferred Execution); the outcome of execution isnot a prerequisite to consensus.

In blockchains where executionis a prerequisite to consensus, the time budget for execution is a small fraction of the block time. Pipelining consensus and execution allows Monad to expend the full block time onboth consensus and execution.

Block proposals consist of an ordered list of transaction hashes and a state merkle root fromD blocks ago. The delay parameterD is set in code; it is expected thatD = 10 initially.

The user must pay to have a transaction included in a block (the "carriage cost "). For each address, nodes maintain two balances:

- a reserve balance, used to pay for this carriage cost
- the execution balance, used to pay for transaction execution
-

Carriage cost is charged to the reserve balance when the transaction is included in a block (consensus); it is deducted from the execution balance at execution time (double charge), and repaid to the reserve balance after the delay period ofD blocks passes.

An account's reserve balance is effectively a budget for "in-flight" orders; it exists to ensure that only transactions that are paid for are included in blocks.

Each account has a target reserve balance which can be altered by interacting with an enshrined smart contract, e.g. for EOAs that anticipate sending a large number of inflight orders. State determinism Finality occurs at consensus time; the official ordering of transactions is enshrined at this point, and the outcome is fully deterministic for any full node, who will generally execute the transactions for that new block in under 1 second.

TheD -block delay for state merkle roots is only for state root verification, for example for allowing a node to ensure that it didn't make a computation error.

Execution

The execution phase for each block begins after consensus is reached on that block, allowing the node to proceed with consensus on subsequent blocks.

Parallel Execution

Transactions are linearly ordered; the job of execution is to arrive at the state that results from executing that list of transactions serially. The naive approach is just to execute the transactions one after another. Can we do better? Yes we can!

Monad implementsparallel execution :

- An executor is a virtual machine for executing transactions. Monad runs many executors in parallel.
- An executor takes a transaction and produces aresult
- . A result is a list ofinputs
- to andoutputs
- of the transactions, where inputs are (ContractAddress, Slot, Value) tuples that were SLOADed in the course of execution, and outputs are (ContractAddress, Slot, Value) tuples that were SSTOREd as a result of the transaction.
- Results are initially produced in a pending state; they are then committed in the original order of the transactions. When a result is committed, its outputs update the current state. When it is a result’s turn to be committed, Monad checks that its inputs still match the current state; if they don’t, Monad reschedules the transaction. As a result of this concurrency control, Monad’s execution is guaranteed to produce the same result as if transactions were run serially.
- When transactions are rescheduled, many or all of the required inputs are cached, so re-execution is generally relatively inexpensive. Note that upon re-execution, a transaction may produce a different set of Inputs than the previous execution did;
-

MonadDb: high-performance state backend

All active state is stored inMonadDb , a storage backend for solid-state drives (SSDs) that is optimized for storing merkle trie data. Updates are batched so that the merkle root can be updated efficiently.

MonadDb implements in-memory caching and usesasio for efficient asynchronous reads and writes. Nodes should have 32 GB of RAM for optimal performance.

Comparison to Ethereum: User's Perspective

Attribute Ethereum Monad Transactions/second (smart contract calls and transfers) ~10 ~10,000 Block time 12s 1s Finality 2 epochs (12-18 min) Single-slot (1s) Bytecode standard EVM (Shanghai fork) EVM (Shanghai fork) RPC API Ethereum RPC API Ethereum RPC API Cryptography ECDSA ECDSA Accounts Last 20 bytes of keccak-256 of public key under ECDSA Last 20 bytes of keccak-256 of public key under ECDSA Consensus mechanism Gasper (Casper-FFG finality gadget + LMD-GHOST fork-choice rule) MonadBFT (pipelined HotStuff with additional research improvements) Mempool Yes Yes Transaction ordering Leader's discretion (in practice, PBS) Leader's discretion (default behavior: priority gas auction) Sybil-resistance mechanism PoS PoS Delegation allowed No; pseudo-delegation through LSTs Yes Hardware requirements (full node) 4-core CPU 16 GB RAM 1 TB SSD 25 Mbit/s bandwidth 16-core CPU 32 GB RAM 2 TB SSD 100 Mbit/s bandwidth

Previous Monad for users Next Technical discussion Last updated3 months ago