Deferred Execution

Pipelined consensus-execution staging One of the novel aspects of the Monad blockchain is that execution is decoupled from consensus.

To recap, consensus is the process where Monad nodes come to agreement about the official ordering of transactions, while execution is the process of actually executing those transactions and updating the state.

In Monad consensus, nodes come to agreement about the official ordering of transactions, but without either the leader or validating nodes having to have executed those transactions yet.

That is, the leader proposes an ordering without yet knowing the resultant state root, and the validating nodes vote on block validity without knowing (for example) whether all of the transactions in the block execute without reverting.

How can this be? And why does Monad do this?

The answer is a cornerstone of Monad's design, and it allows Monad to unlock significant speedups that allow a single-shard blockchain to scale to millions of users.

Interleaving execution and consensus is inefficient

In Ethereum, execution is a prerequisite to consensus, so when nodes come to consensus on a block, they are agreeing on both (1) the list of transactions in the block, and (2) the merkle root summarizing all of the state after executing that list of transactions. As a result, the leader must execute all of the transactions in the proposed blockbefore sharing the proposal, and the validating nodes must execute those transactionsbefore responding with a vote.

In this paradigm, the time budget for execution is extremely limited, since it has to happen twiceand leave enough time for multiple rounds of cross-globe communication for consensus. Additionally, since execution will block consensus, the gas limit must be chosen extremely conservative, ensuring that the computation completes on all nodes within the budget even in the worst-case scenario.

Determined ordering implies state determinism

Here is an obvious but crucial insight: given an official ordering of transactions, the true state is completely determined. Execution is required to unveil the truth, but the truth is already determined.

Monad takes advantage of this insight by removing the requirement that nodes execute prior to consensus. Node agreement is purely about the official ordering; each node executes the transactions in blockN independently while commencing consensus on blockN+1 .

This allows for a gas budget corresponding to the full block time, since execution merely needs to keep up with consensus. Additionally, this approach is more tolerant of variation in exact compute time, since execution only needs to keep up with consensus on average.

Delayed merkle roots still ensure state machine replication

The main objections one might have to the above are:

- What happens if one of the nodes is malicious, so doesn't execute the exact transactions specified in consensus? (For example, say it omits certain transactions, or just sets a state variable to an arbitrary value of its choice.)
- What happens if one of the nodes makes a mistake in execution?
-

To address these concerns, in Monad the block proposals include a merkle root delayed byD blocks, whereD is a systemwide parameter (currently anticipated to be 10). As a result of this delayed merkle root:

1. After the network comes to consensus (2/3 majority vote) on blockN
2. , it means that the network has agreed that the official consequence of blockN-D
3. is a state rooted in merkle rootM
4. . Light clients can then query full nodes for merkle proofs of state variable values at blockN-D
5. .
6. Any node with an error in execution at blockN-D
7. will fall out of consensus starting at blockN
8. . This will trigger a rollback on that node to the end state of blockN-D-1
9. , followed by re-execution of the transactions in blockN-D
10. (hopefully resulting in the merkle root matching), followed by a re-execution of the transactions in blockN-D+1
11. ,N-D+2
12. , etc.
13.

Ethereum's approach uses consensus to enforcestate machine replication in a very strict way: after nodes come to consensus, we know that the supermajority agrees about the official ordering and the state resulting from that ordering. However, this strictness comes at great cost - extremely limited throughput. Monad loosens the strictness slightly, to great effect.

On finality

In MonadBFT, finality is single-slot (1 second), and execution outcome will generally lag by less than 1 second for those using a full node. Let's unpack this a bit.

Finality in Monad is single-slot (1 second) . If you submit a transaction, you will see the transaction's official order (among all other transactions) after a single block. There is no potential for reordering, barring malicious action from a supermajority of the network. This makes Monad's finality significantly faster thanEthereum's (2 epochs, aka 12.8 minutes).

The execution outcome of a transaction (did it succeed or fail? what are the balances afterward?)will generally lag finality by less than 1 second on full nodes. Anyone who needs to know the outcome of a transaction quickly (for example, a high-frequency trader wanting to know the status of an order) can run a full node. Monad is designed to minimize the cost of full nodes; see Hardware Requirements for further information.

Anyone who wants to securely query the outcome of a transaction without running a full node can run a light client while querying a full node for balances with merkle proofs. In this case, queries will be lagged by the merkle root delay (D=10 blocks, i.e. 10 seconds). Note that most users currently view the state of the blockchain using software (browser) wallets or via a block explorer. Neither of these usage patterns involve a light client.

Some readers might mistakenly conflate the merkle root delay (D=10 blocks) with finality and mistakenly assume that finality is 10 blocks. That's not true. The official transaction order is determined after 1 block, after which there will be no reorgs without Byzantine behavior from a supermajority.

Previous Shared Mempool Next Carriage Cost and Reserve Balance Last updated5 months ago