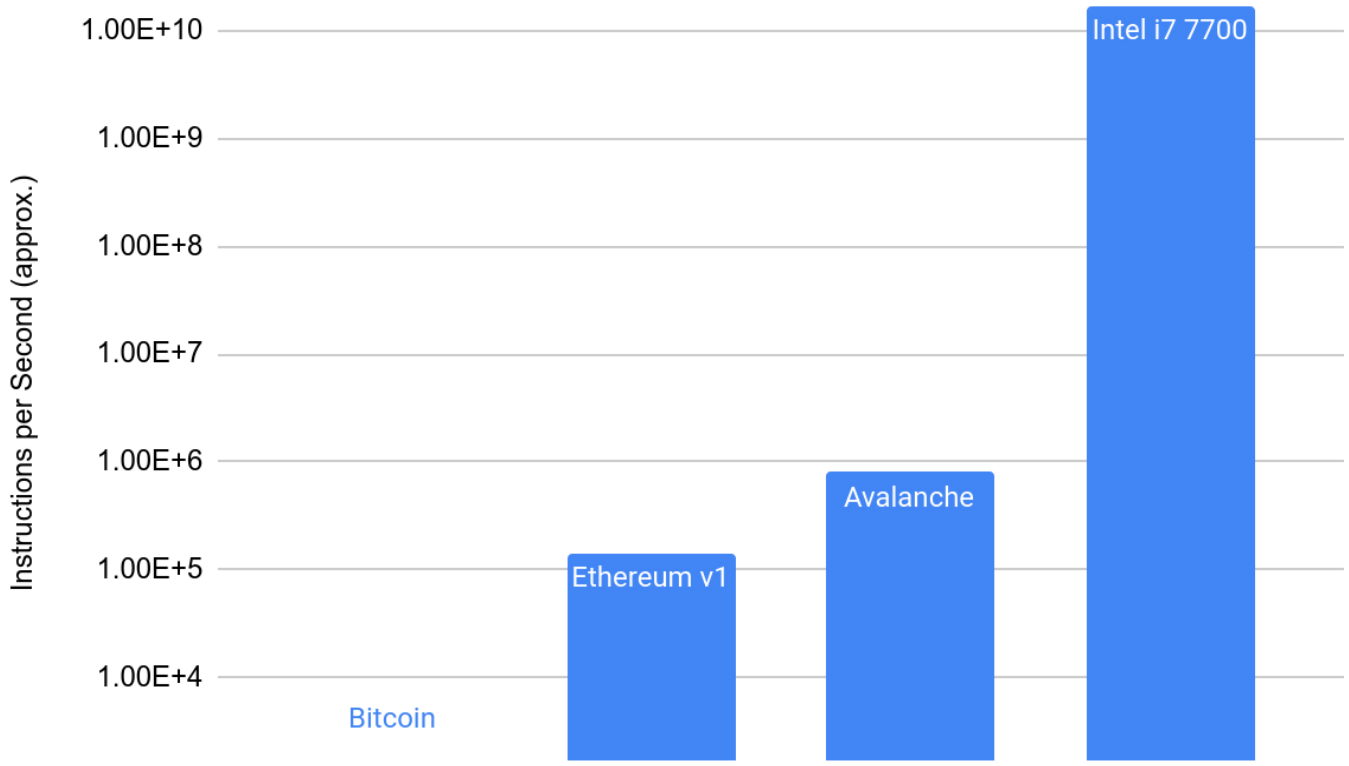
Arbitrum

Kai Mast – CS839

The Performance Gap

Numbers are based on approximations

- A simple instruction costs about 5 gas in ETH
- Bitcoin Script is not Turing complete but supports about 1000 instructions/per second
- For Avalanche, I assumed a block every 2 second with 8 million gas



Recap: Layer 2

- State and Payment Channels
  - Two parties exchange cryptographic commitments
  - State/payments can be routed through a network of channels

- Subchains, e.g., Plasma
  - A set of validators manages a chain
  - Chain is anchored to/secured by a parent chain

Arbitrum: Scalable, private smart contracts

- Published in 2018 at USENIX Security by Harry Kalodner, Stephen Goldfeder, Edward Felten and other
- Now managed by a blockchain start, Offchain Labs
- Arbitrum provides an optimistic rollup mechanism
- "Arbitrum One" mainnet launched in August 2021
  - Note, there are some difference between the Arbitrum paper and Arbitrum One

Harry Kalodner
(Offchain Labs)

Stephen Goldfeder
(IC3, Offchain Labs)

Edward Felten
(Princeton University)

Arbitrum Governance

Verifier (the parent chain)

- Holds all metadata of the Arbitrum chain
- Servers as a neutral party to manage failure

Managers (or "Stakers" in Arbitrum One)

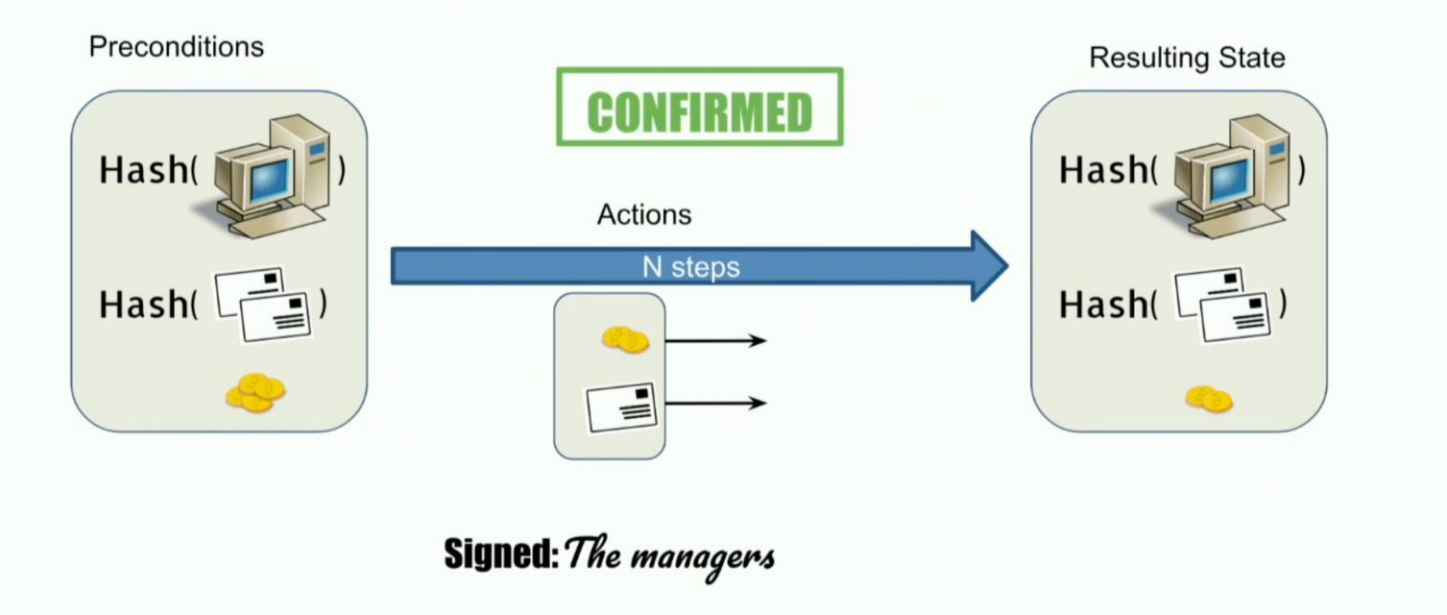
- Sign-off new changes to the Arbitrum blockchain
- Need to stake some of amount of money for accountability
- Disputes between managers are handled by the parent chain
- Any-trust assumption: At least one manager must be correct/honest to ensure safety

Keys (Clients)

- Issue new transactions
- Clients can also be managers themselves

Arbitrum Execution Model

- Parent chain holds account balance and hashes of the chain state
- Each Arbitrum chain has an "inbox" that manages transaction requests and messages from other Arbitrum chains
- Managers propose updates to the state
  - Each update contains many computational steps
- For a state transition to be valid it must be
  - Signed off by all managers
  - Or signed off by one manager and be uncontested

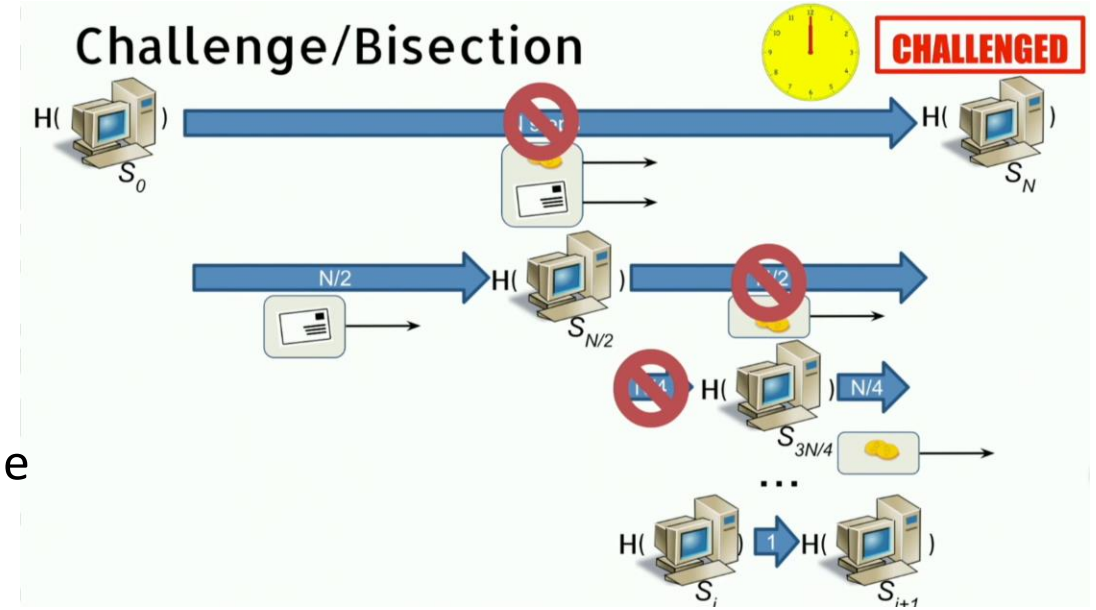


(From Harry Kalodner's Slides)

Improving Performance of Fraud Proofs

Problem: Main chain does not hold entire state of the Arbitrum chain, but needs to be able to verify the fraud proof.

- Main chain needs to re-execute everything
  - Remember, subchain might be "faster" than the parent chain
- Or challenger needs to provide a lengthy fraud proof



(From Harry Kalodner's Slides)

Solution: Narrow fraud proof down to the (first) point in the state transition where something went wrong

- Every round, the creator(s) of the state update split the update in half
- Each half is simply represented as a block header
- The challenger then picks which one they claim is incorrect

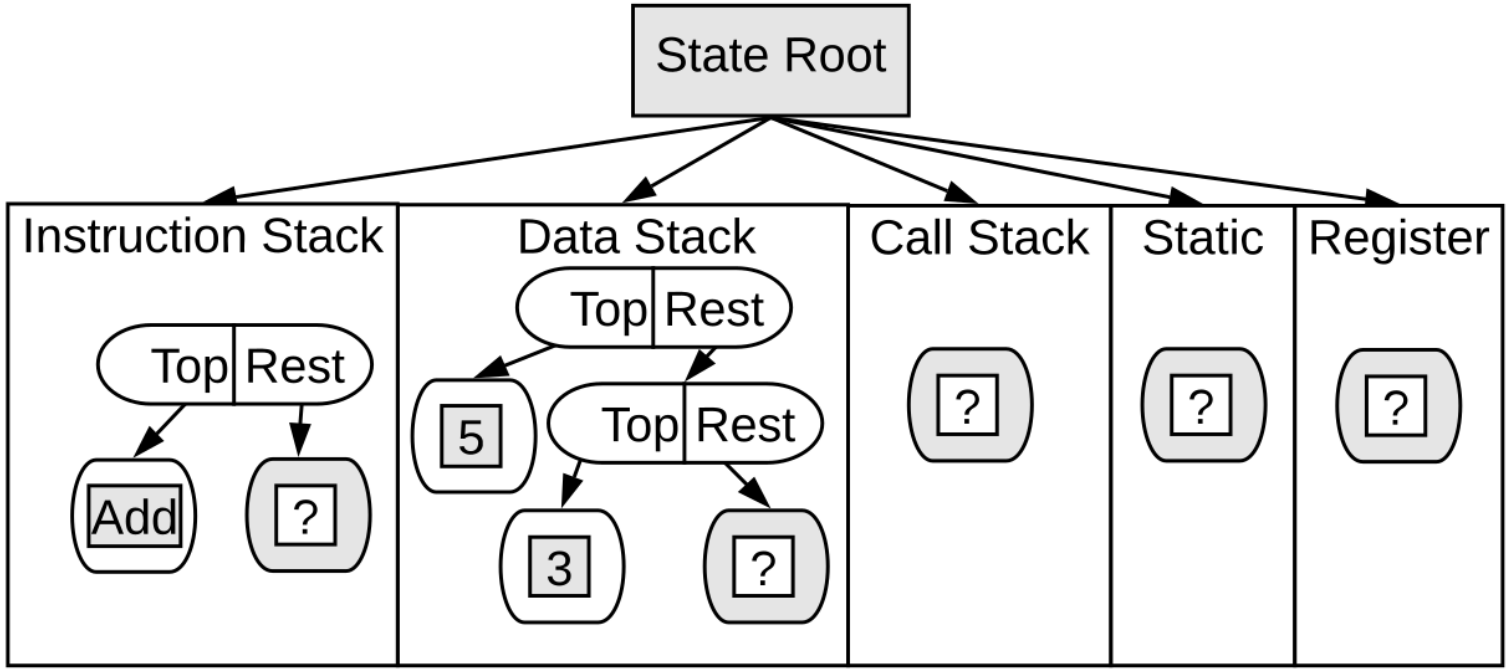
The Arbitrum Virtual Machine

Goal: Keep state at any point in time small

1. Instruction Stack (instead of program counter)
2. Data Stack (no heap)
3. Call stack
4. Static Data (e.g., program code and constants)
5. One Register (instead of many)

One-Step Proofs

- We only need to show find first computation step where "things went wrong"
  - all preceding state has been proven correct by the bisection protocols
- Proofs should be as small as possible to not overload the main chain
  - Arbitrum claims proof size of only about 100bytes



Privacy in Arbitrum

- By default VM state is only held by the managers and not revealed unless there is a dispute
- Disputes only reveal a small portion of the state
- Optionally: Implement one-step proofs as Zero-Knowledge proofs

Evaluation

- Initial implementation in 6800 lines of Go code
- Benchmarks were run on a 2013 Apple MacBook Pro with a 2.7GHz Intel Core i7
- Arbitrum VM can generate about 970,000 SHA2 hashes per second
  - Approx. 57% of native performance

Arbitrum Rollups

- The version that is actually deployed on Ethereum
  - Each Arbitrum chain has an on-chain contract (or "EthBridge") associated with it
- This version differs somewhat from the mechanisms described in the paper
- See full documentation here: https://developer.offchainlabs.com/docs/rollup_basics

Recap: Optimistic Rollups

- Smart contract is stored on the parent chain
- Each transaction/call to the contract is stored on the parent chain
- An off-chain aggregator executes transaction and updates the state on-chain

Advantages:

- Can run any kind of smart contract
- No availability problem; state can always be recomputed on chain

Disadvantage:

- Does not (easily) allow for batching

Rollup Chain Blocks

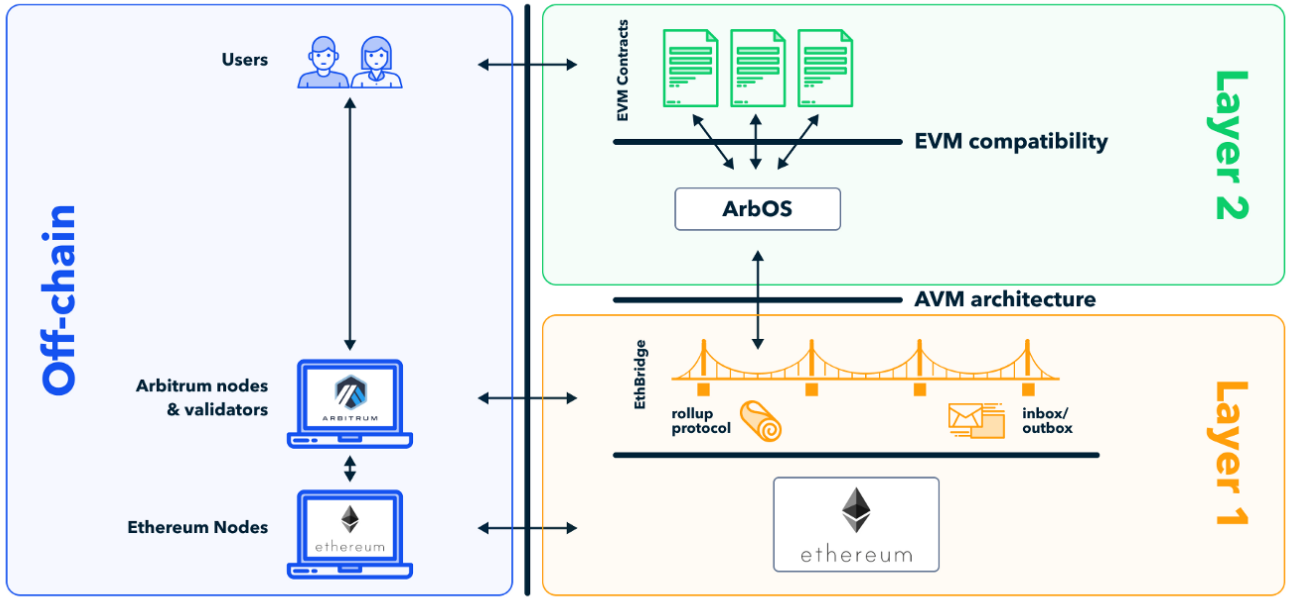
Each block on the rollup chain contains:

- the rollup block number
- the predecessor block number: rollup block number of the last block before this one that is (claimed to be) correct
- the amount of computation the chain has done in its history (measured in ArbGas)
- the number of inbox messages have been consumed in the chain's history
- a hash of the outputs produced over the chain's history
- a hash of the chain state

Executing Ethereum Contracts on Arbitrum

ArbOS

- ArbOS is an "operating system" that allows running multiple contracts on the same Arbitrum VM
- Provides isolation between the contracts
- ArbOS supports executing EVM bytecode and, thus, unmodified

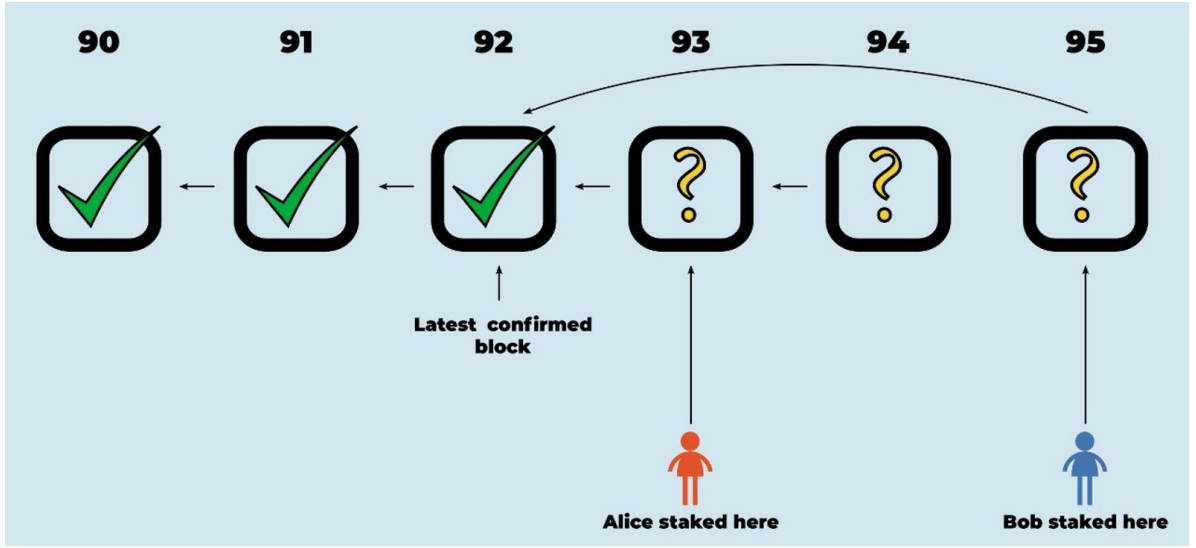


Staking in Arbitrum One

- Staking is permissionless: Anyone can join an Arbitrum chain as a staker (or manager in the old terminology)
  - The staking amount is defined by the on-chain contract (the EthBridge)
- Stakers "move" their stake to a specific block to indicate that they verified a block and all its predecessors
- Any staker can create new blocks
  - Block creation is defined by the Arbitrum code/ArbOS (e.g., create a block every k instructions)
- If two stakers back different blocks, and the blocks are siblings, a challenge must be started to resolve the conflict

Staking in Arbitrum One: Example

- Blocks 93 and 95 both have block 92 as their predecessor
- There needs to be a challenge between 93 or 95, and Alice or Bob respectively
- Note that only one block can be correct
  - Input is defined by the state stored on the Ethereum chain
  - All execution is deterministic



(Source: https://developer.offchainlabs.com/docs/inside_arbitrum)

Aggregators in Arbitrum One

Problem: For rollup chains, all inputs and outputs must be stored on chain

- Issuing an on-chain transaction might be too expensive

Solution: Special "Aggregator"-nodes accumulate many transaction requests/messages and issue them as a batch

- Aggregators are (mostly) trustless
- Aggregators may compress function calls to fit them in a single on chain transaction
- Multiple aggregators can exist at the same time for a single Arbitrum chain

Accepting or Rejecting Blocks in Arbitrum One

Blocks are accepted if (all of the below)

- the block’s predecessor is the latest confirmed block
- the block’s deadline has passed
- there is at least one staker
- all stakers are staked on the block

Blocks are rejected if

- the block’s predecessor has been rejected
- Or all of the following are true:
  - the block's deadline has passed
  - there is at least one staker
  - no staker is staked on the block
- Or the block has been proven faulty by a challenge

Finality in Arbitrum One

- An Arbitrum One block can be disputed within some deadline
  - Usually a week, like Plasma
  - Similar problems (and solutions) to exiting the chain as in Plasma
- However, one can treat a transaction as "final" as soon as all stakers, or one staker they trust, have signed off on a block

Connection parent chain (L1) and subchain (L2) contracts

Problem: Contracts on two different chains cannot invoke each other in a single transaction

- L1 Contracts can issue to a calls to an Arbitrum chain's EthBridge, but results will not be available immediately

Solution: "Tickets" allow saving the on-chain state and redeeming it later

- Tickets contain the sender’s address, a destination address, a callvalue, and calldata
- Tickets are stored in the Arbitrum-chain's EthBride
- Once the off-chain execution is done and finalized the ticket is redeemed and on-chain execution resumes

That's all for today

- Conclusion:
  - Arbitrum allows executing contracts much faster than Ethereum
  - Trade-offs are either in safety (the original paper requires one honest manager) or latency (Arbitrum One takes a week to finalize transactions)

- Please try to meet with me at least once a week to touch base about your project status
- Next week: Ethereum 2.0 (probably)