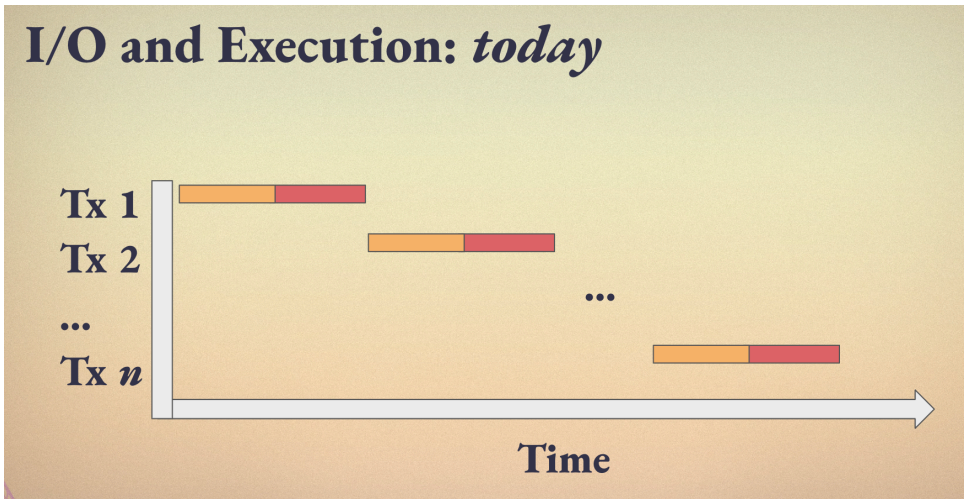
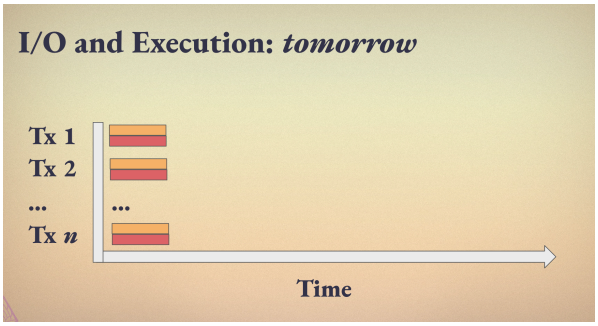
Block level access list

Motivation

- Current transaction validation is sequential.

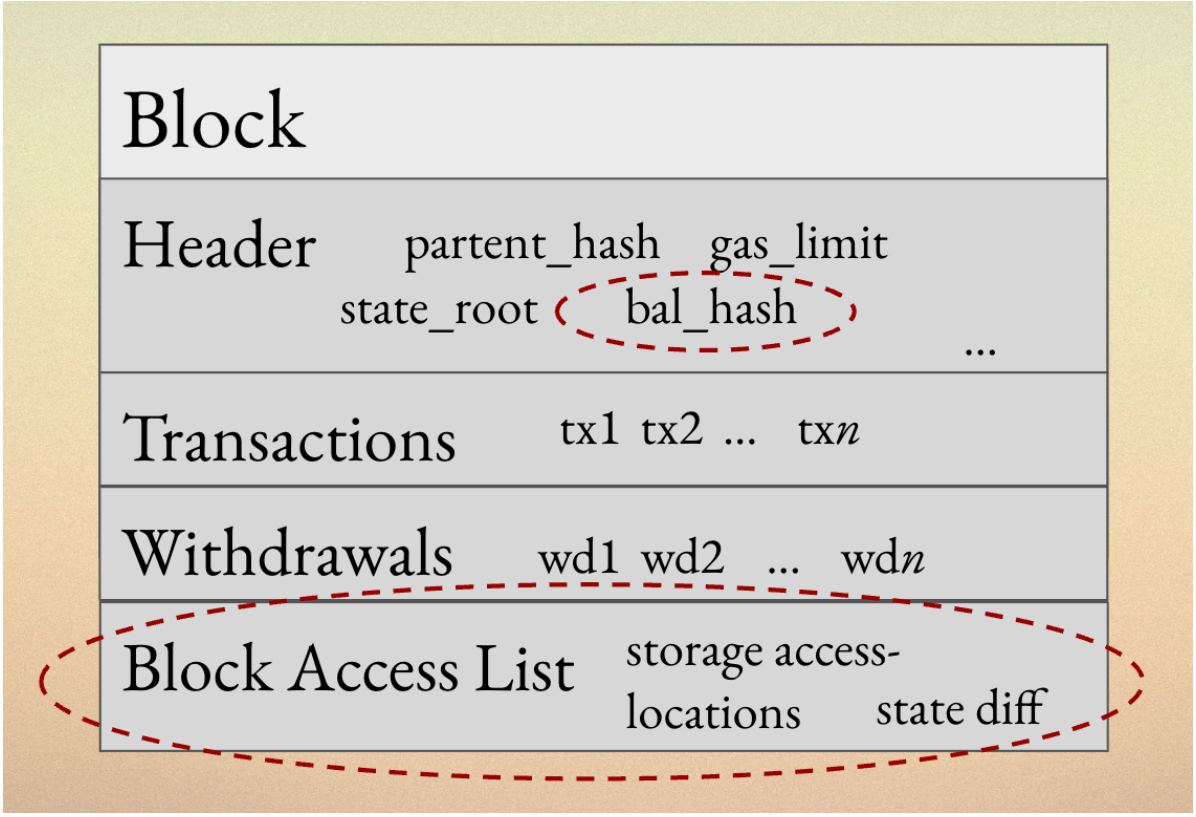




- Two tasks involved in transaction validation:
  - IO
  - Execution
- Goal: Introduce parallelization to Ethereum through an explicit access list.

BAL structure - EIP 7928

- Modified block structure



- Block contains:
  - A hash of the access list in the block header
  - The access list itself in the block
    - Storage locations
    - State diffs
- Question : Should the access list be included in the block?
  - Pruning or Sidecar can be candidate solutions
- Enables parallelization of both disk reads and transaction execution

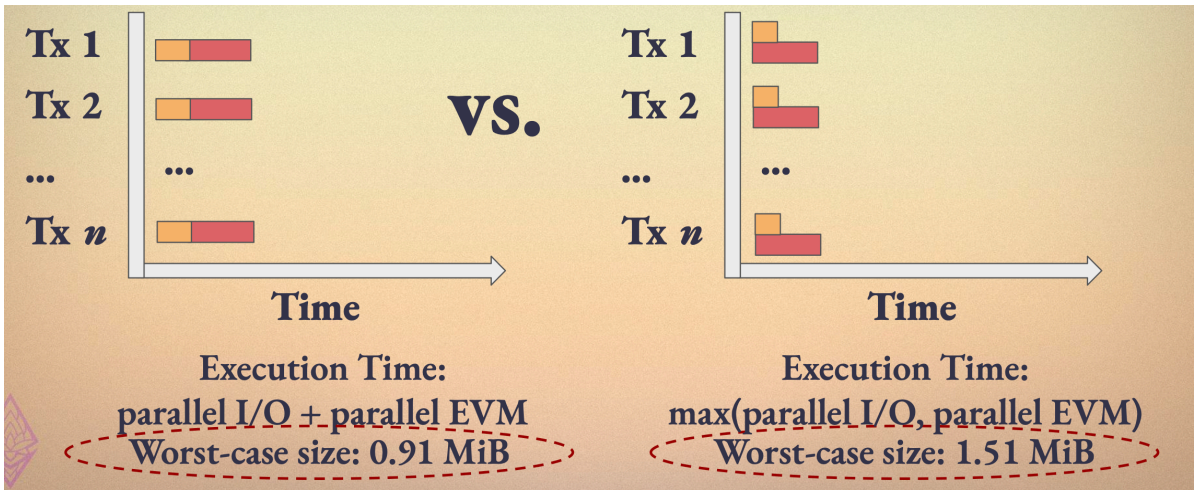
BAL Design Space

- BAL can contain

- Storage locations
  - Storage location tuple (address, storage key)
  - Worst-case: no parallelization
- Storage values
  - Pre-tx / Post-tx values
  - Pre-block / Post-block values
- State diffs (Including balance, code, nonce diffs)

Design Trade-offs

- Pre-tx valus vs Post-tx values



  - Pre-tx
    - Enables IO/execution parallelization
      - Question about skipping IO
      - Result of IO will be still needed for validation of access list
    - Makes BAL bigger
  - Post-tx
    - Better BAL size
    - With state diff, can enable IO/execution parallelization

Design	Execution Times	Max Size	State diff
Storage Locations	Parallel I/O + sequential EVM	0.93 MiB	❌
Pre-tx values[1]	max(parallel I/O, parallel EVM)	1.51 MiB	❌
Post-tx values[1 2]	Parallel I/O + parallel EVM	0.93 MiB	✅
Pre-block + Post-tx values[1 2]	max(parallel I/O, parallel EVM)	1.51 MiB	✅

[1] with storage locations
[2] with state diffs