Turbo-Geth: optimising Ethereum clients

Alexey Akhunov

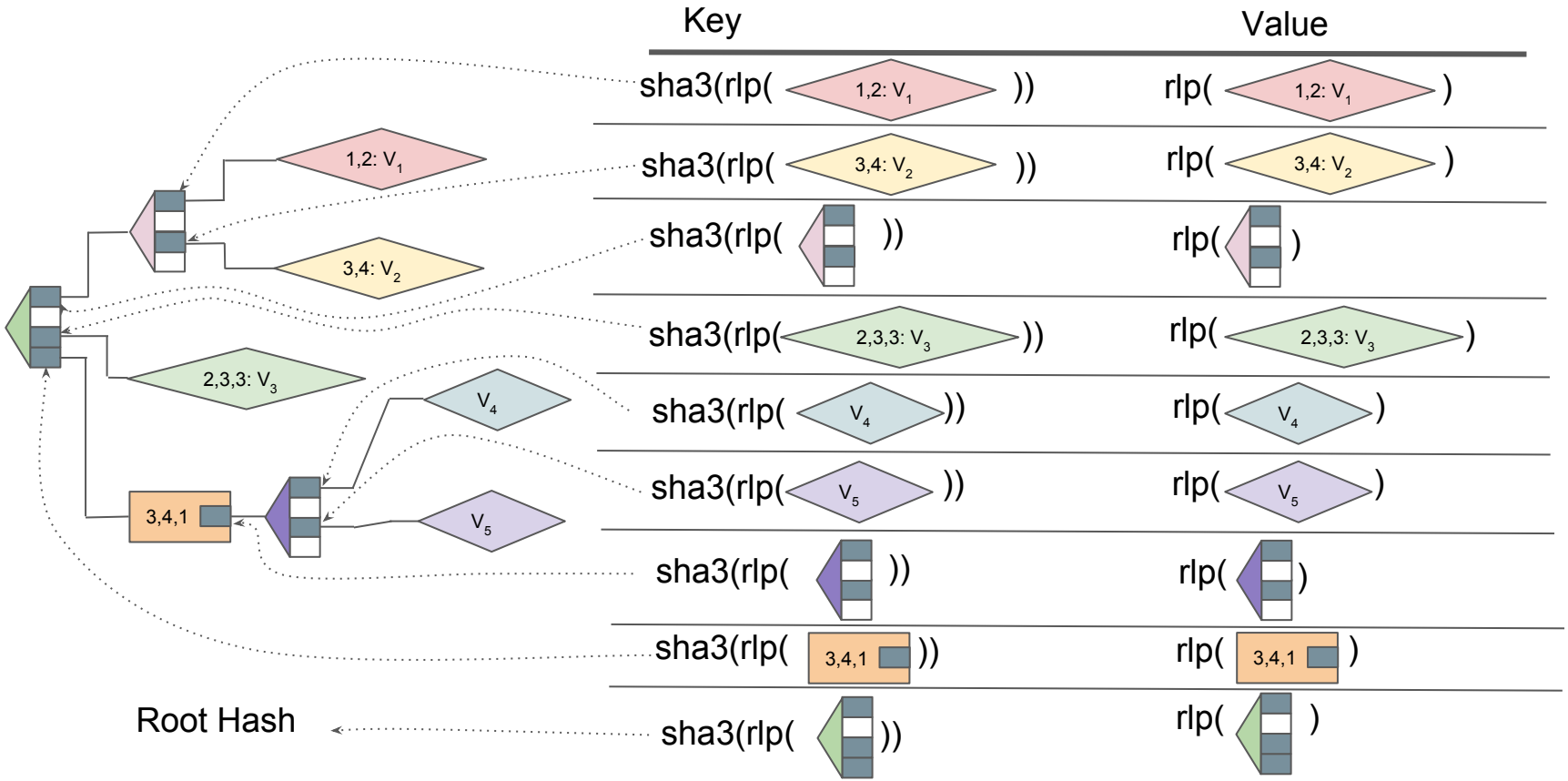
Supported by:

INFURA

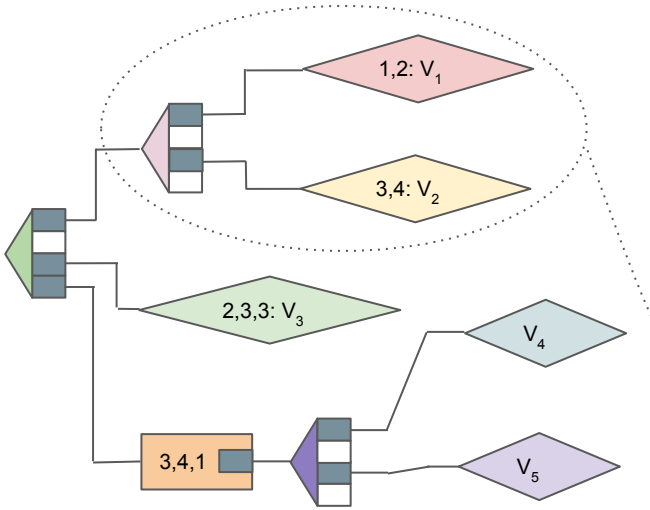
INTERCHAIN FOUNDATION

Ledgerwatch

Persistence of Patricia tree in geth



Persistence ~~of Patricia tree~~ in turbo-geth



Key	Value
1,1,1,2	V_1
1,3,3,4	V_2
3,2,3,3	V_3
4,3,4,1,1	V_4
4,3,4,1,3	V_5

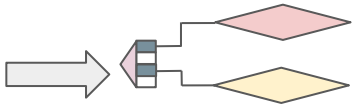
Goes here because it is sorted

DEPTH DOES NOT MATTER

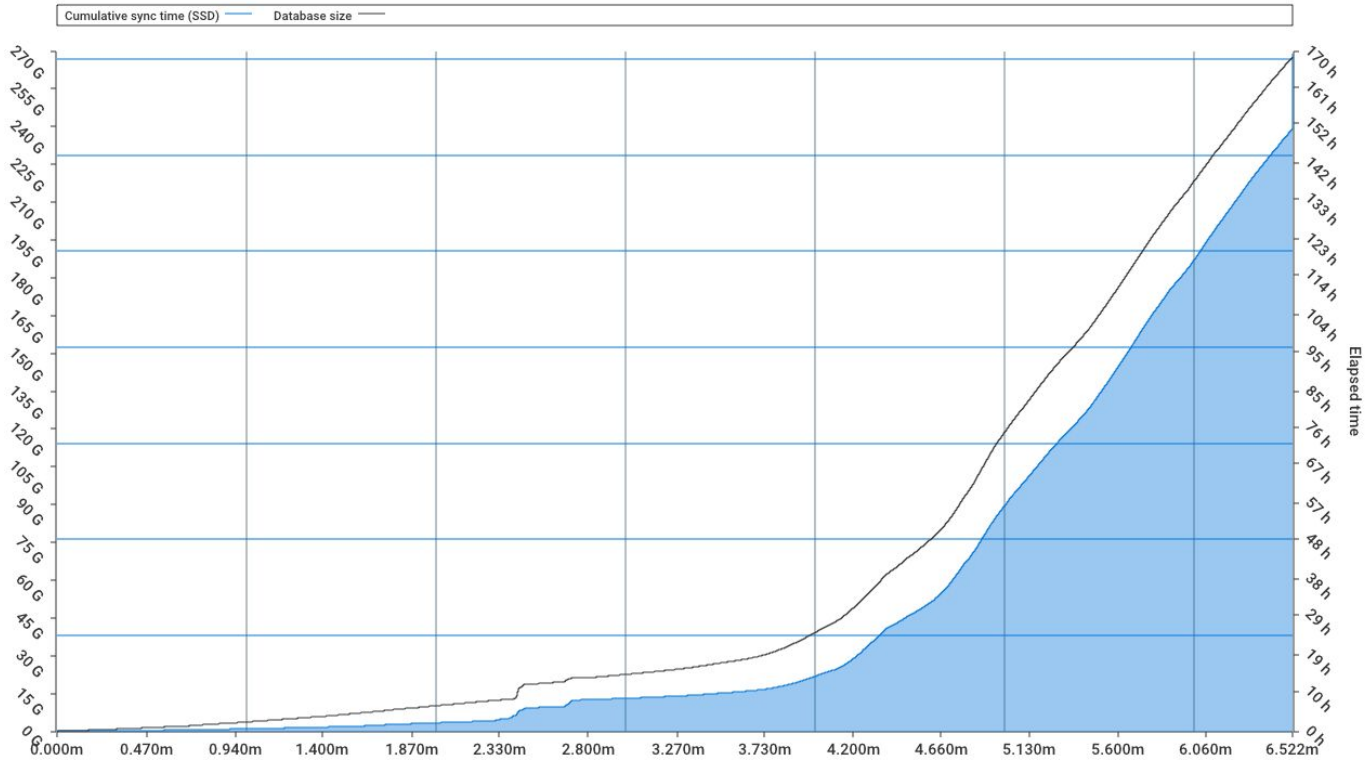
Range query:
1,*,*,*,>=-blockNr

1,1,1,2
1,3,3,4

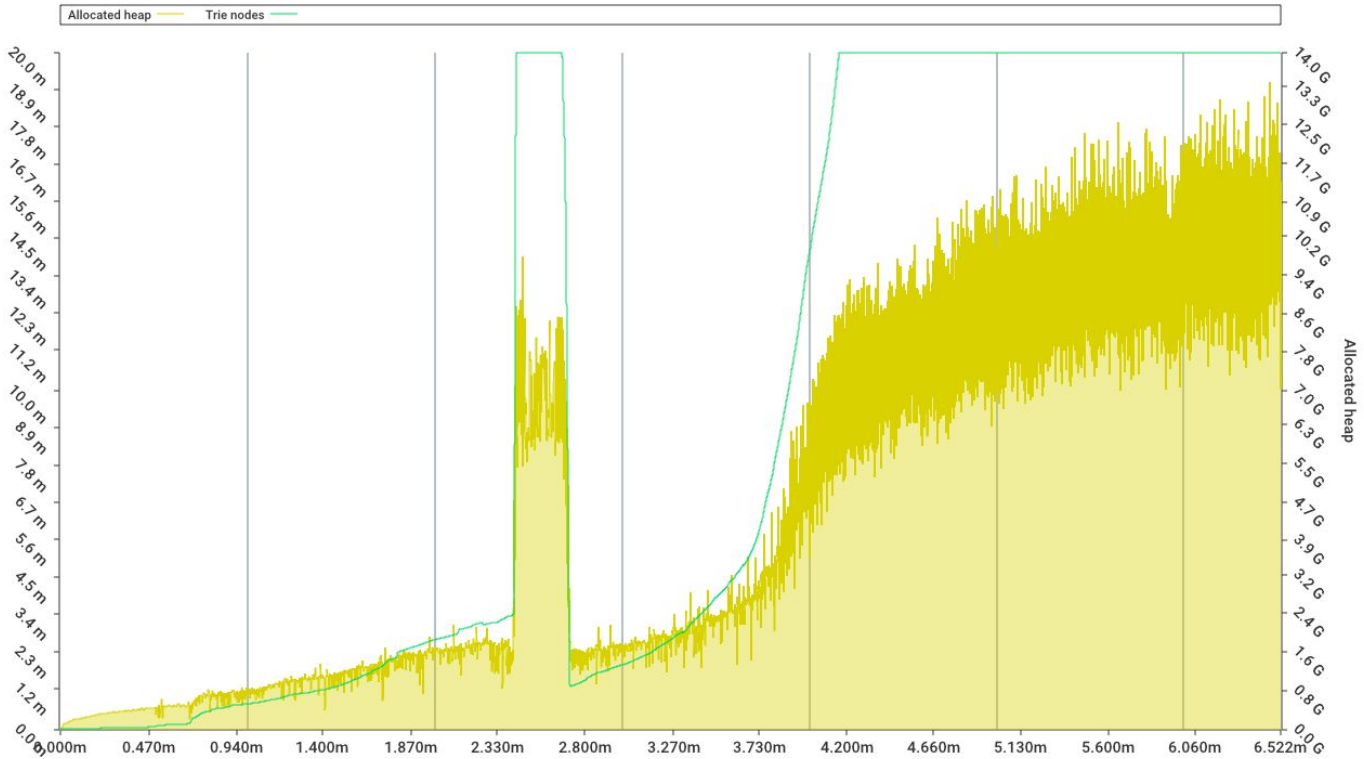
V_1
V_2



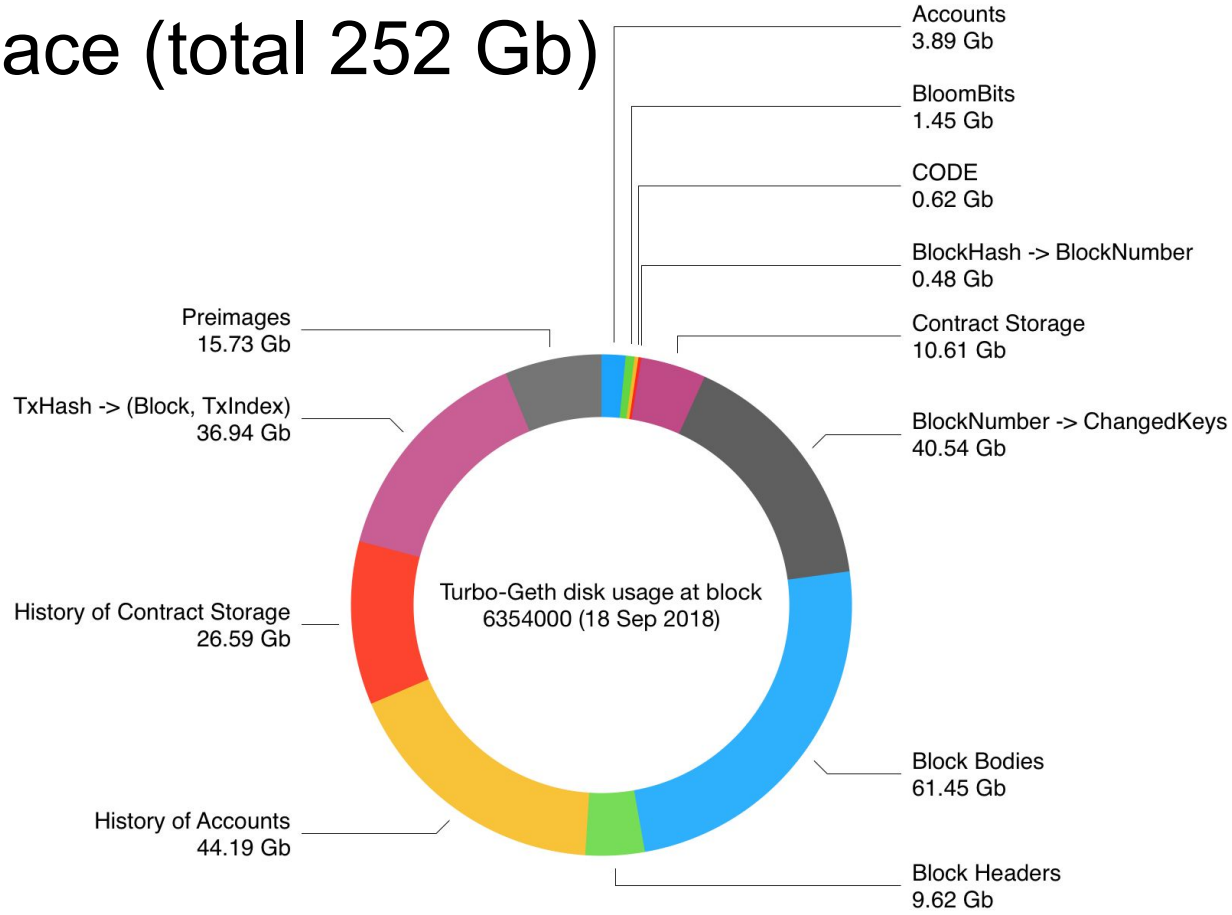
Sync on i3.4xlarge (122 Gb, 16 vCPUs, NVMe)



Sync on i3.4xlarge (122 Gb, 16 vCPUs, NVMe)



Disk space (total 252 Gb)

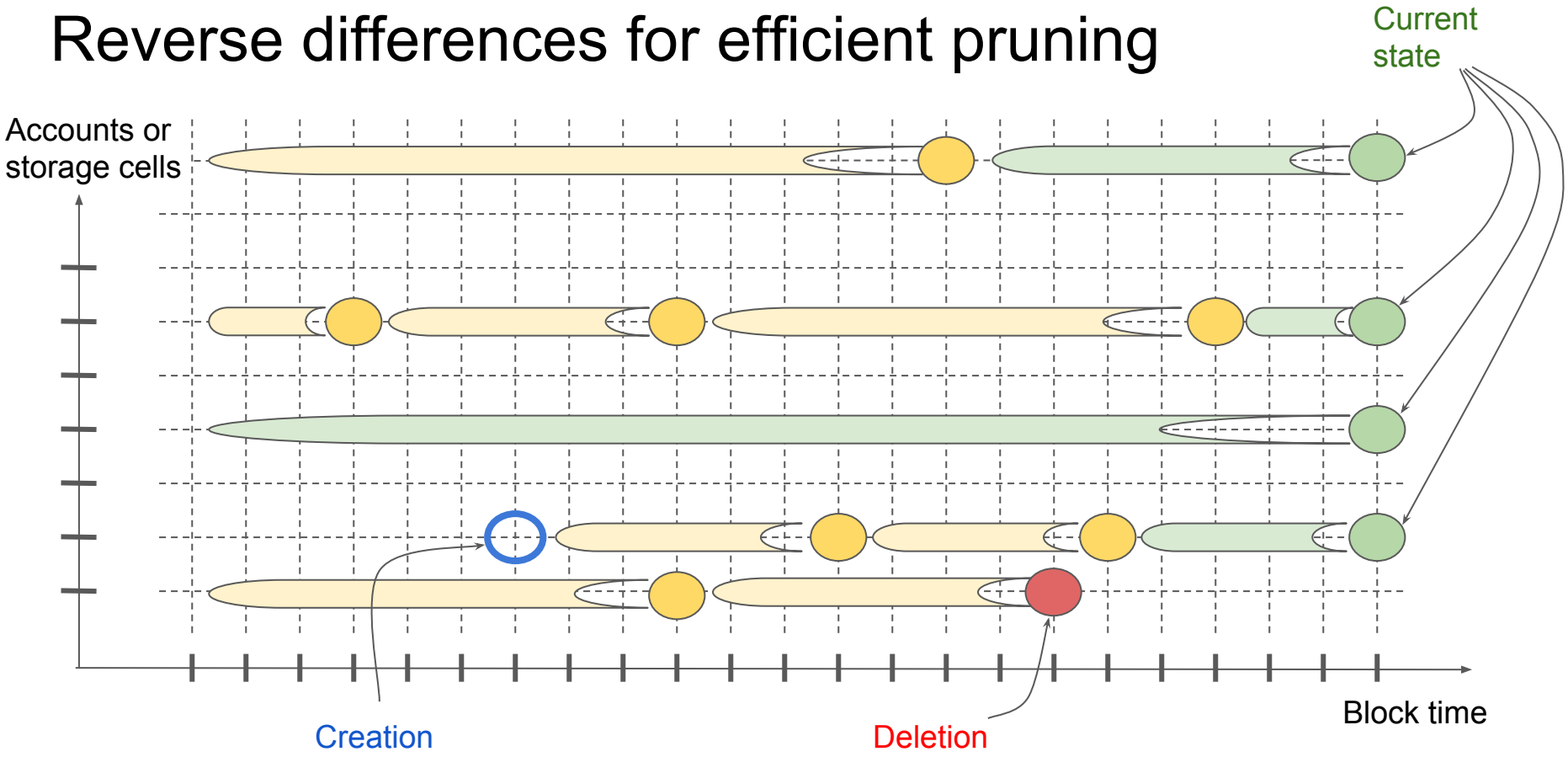


JSON RPC performance

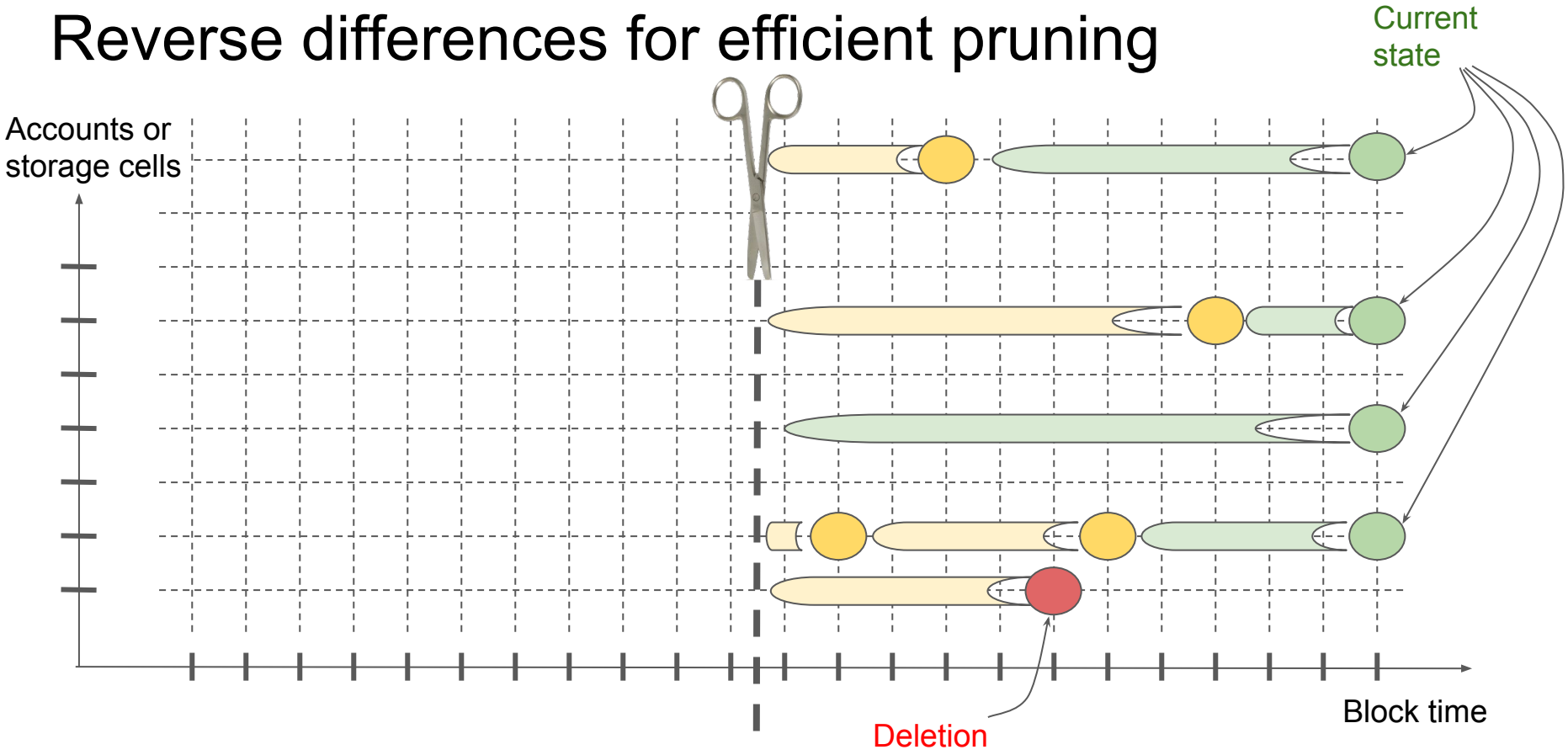
No rigorous benchmarks yet, these are very rough and preliminary numbers.RPC needs a lot more work

Type of RPC call	**Difference from geth archive node**
debug_traceTransaction	10x faster
debug_storageRangeAt	50x faster
debug_getModifiedAccountsByNumber	10x faster, also includes accounts created and destroyed within the block range
eth_getTransactionReceipt	up to 10x slower
eth_getBalance	5x faster

Reverse differences for efficient pruning



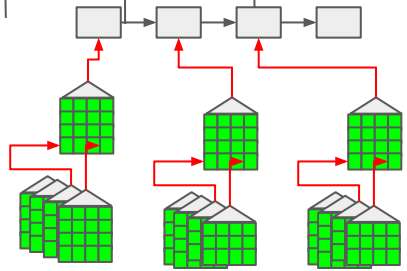
Reverse differences for efficient pruning



Light clients?

	eth/63	les/2	
Status	✓	✓	Handshake - negotiate version, network_id, genesis
[Get]Block(Headers|Bodies)	✓	✓	Get/Send headers/blocks by number or by hash
[Get]NodeData	✓	✓	Get/Send nodes of the patricia tree by hash
[Get]Receipts	✓	✓	Get/Send receipts by transaction hash
NewBlock[Hashes]	✓		Announce new block/block hash
Announce		✓	Announce new chain head
[Get]Proofs		✓	Get/Send merkle proof for given part of trie and block hash
[Get]ContractCode		✓	Get/Send code of given contract at block hash
SendTx	✓	✓	Add new transaction to the pool and relay
[Get]HelperTreeProofs		✓	Get/Send merkle proof of block hash/bloom filters

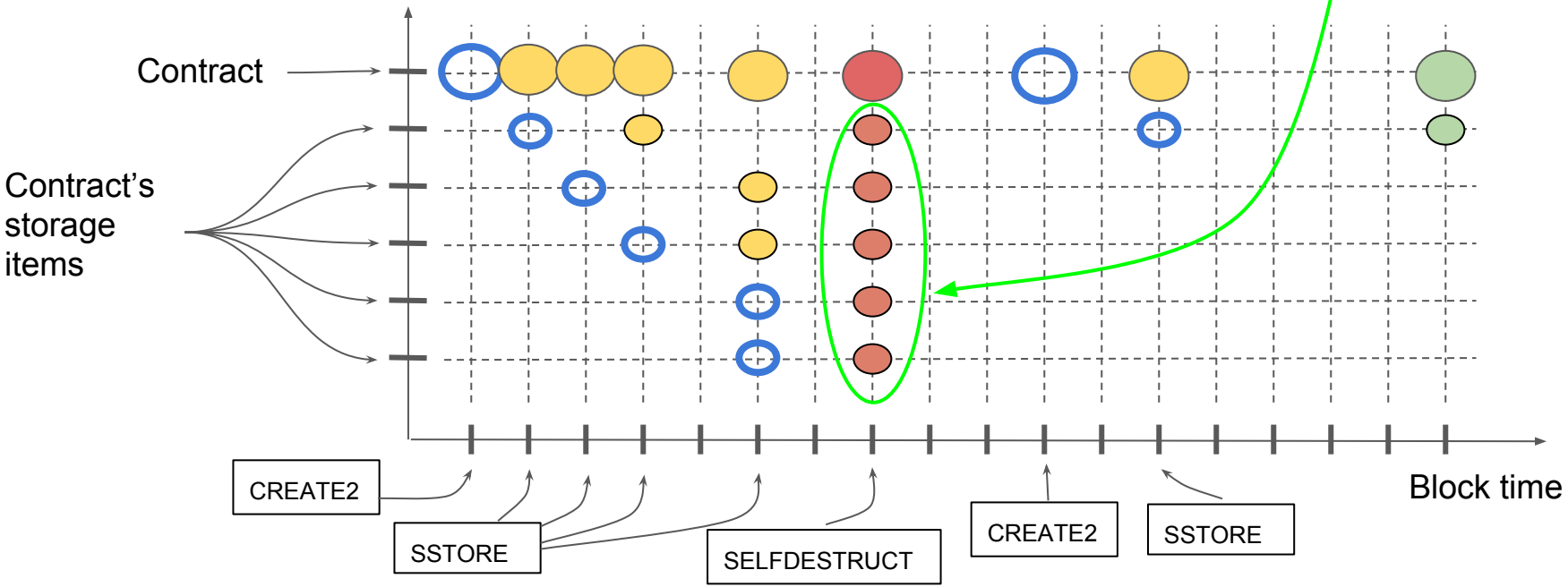
Require “materialised” Patricia tree



CREATE2 revival problem

In EIP-1014, address of contract is calculated as:

```
keccak256( 0xff ++ address ++ salt ++ keccak256(init_code)))[12:]
```



Morus

Codename for **family** of databases for storing of mutating authenticated state. Designed to:

1) Compactly store and prune state history

2) Allow efficient access to history

3) Support various authentication methods: Patricia trees, Sparse Merkle trees, IAVL trees, Weight-Balanced trees

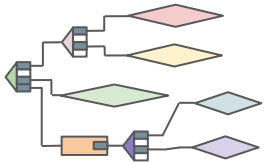
4) Help application with state caching

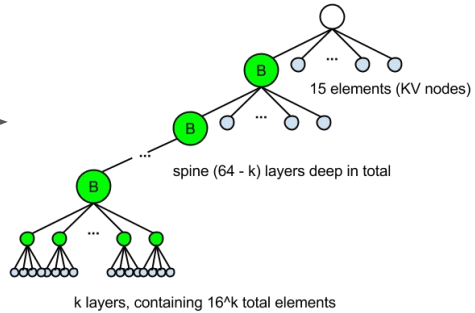
Will allow solving **light client** and **CREATE2 revival** for Turbo-Geth

Can become a backend store for **Ethermint**

Modelling the state - current approaches

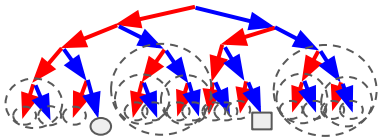
Patricia Tree





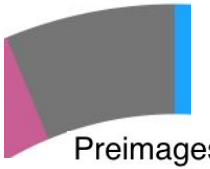
Andrew Miller (Least Authority), Ethereum security audit

Sparse Merkle Tree



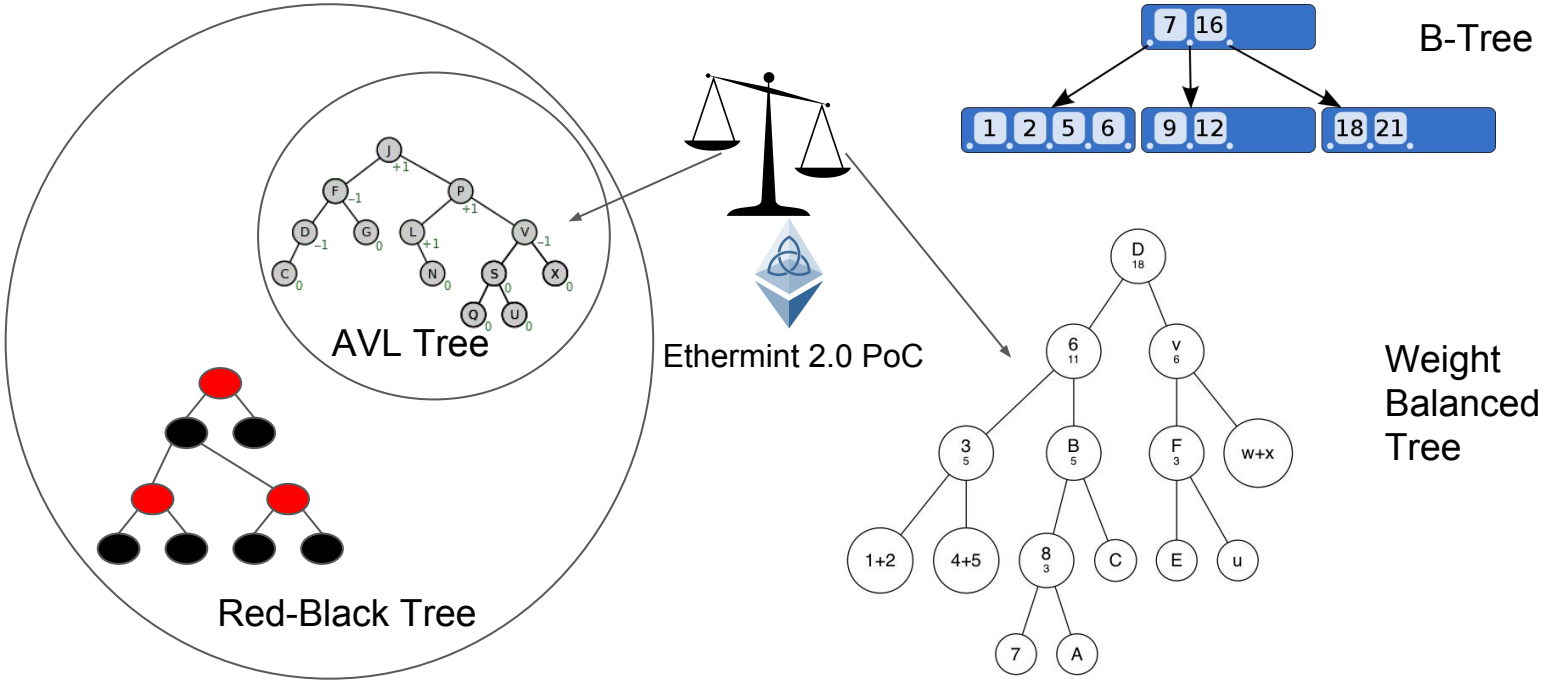
put(K, V)
get(K) -> V
delete(K)

put(sha3(K), V)
get(sha3(K)) -> V
delete(sha3(K))

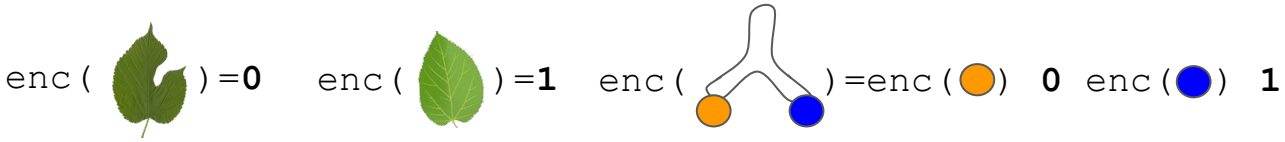


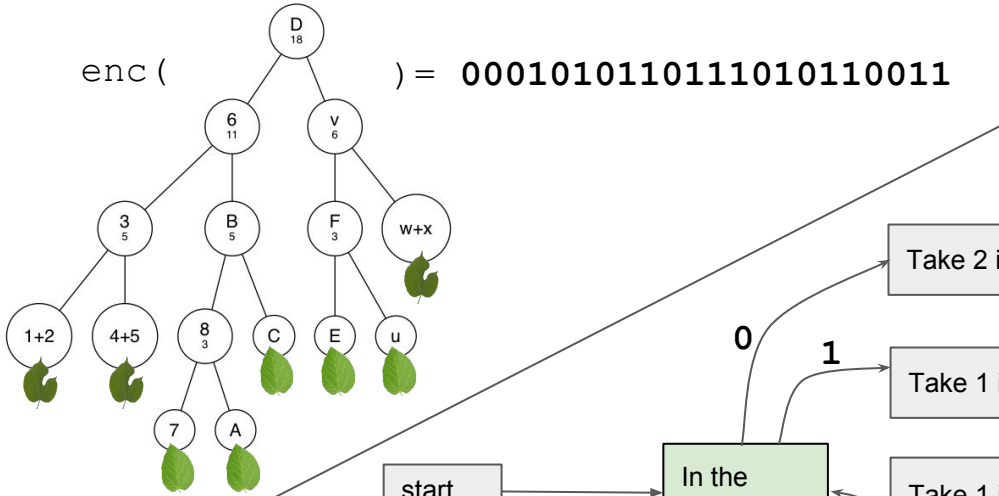
Preimages
15.73 Gb

Modelling the state - self balancing trees

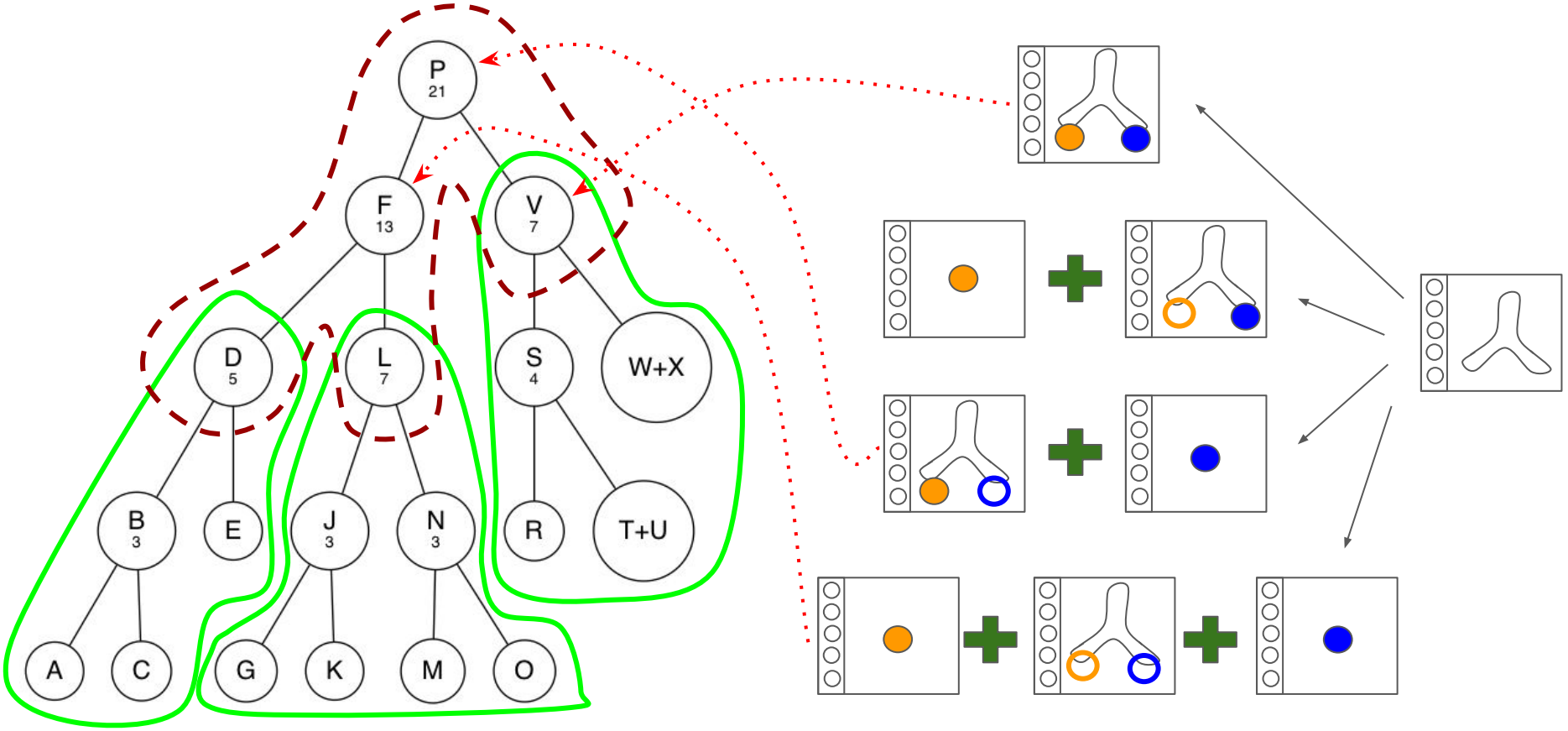


Encoding structure of a binary tree

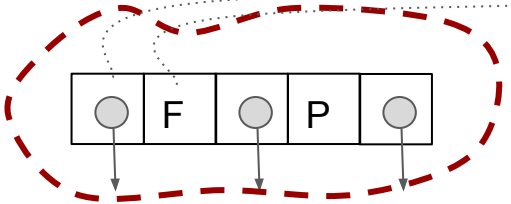


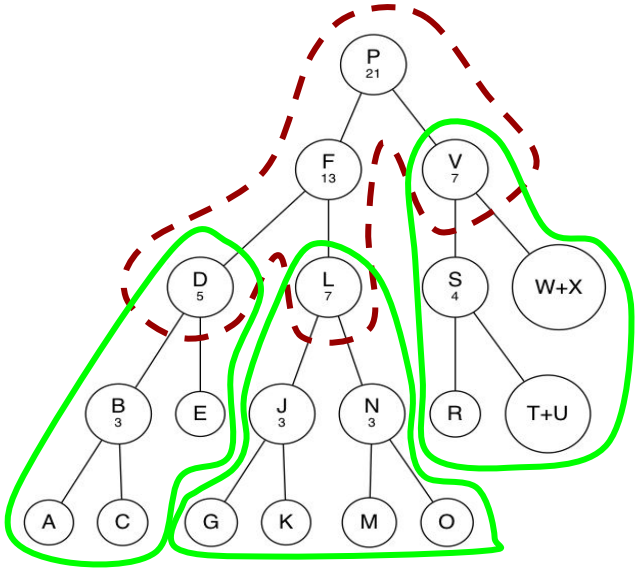


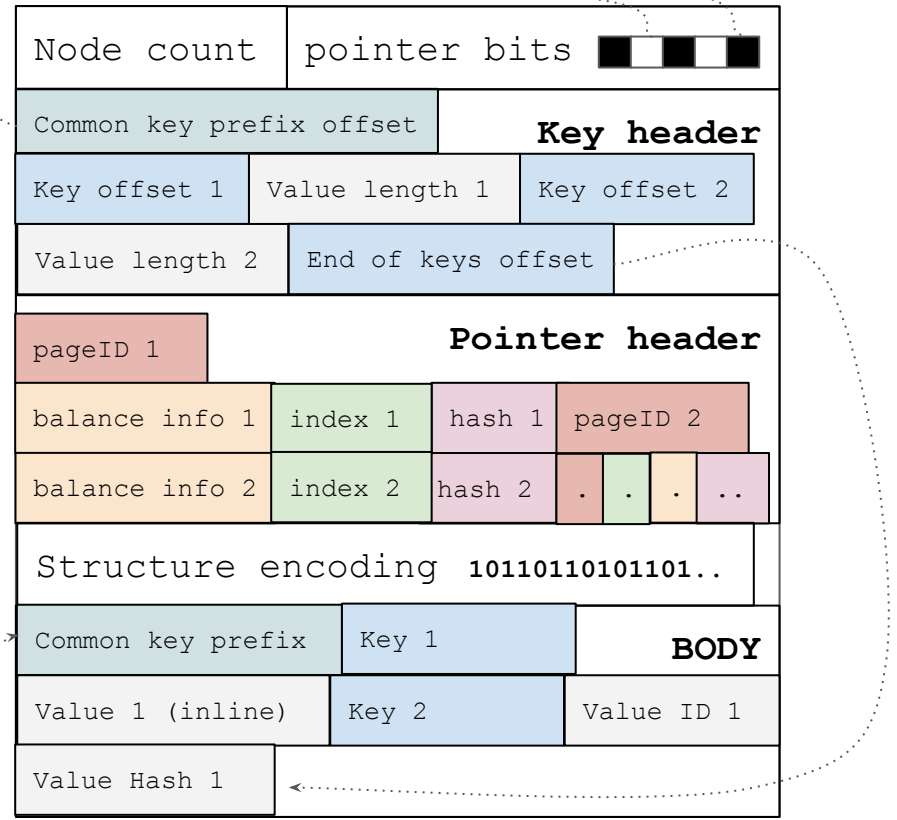
Splitting binary tree into pages of fixed size



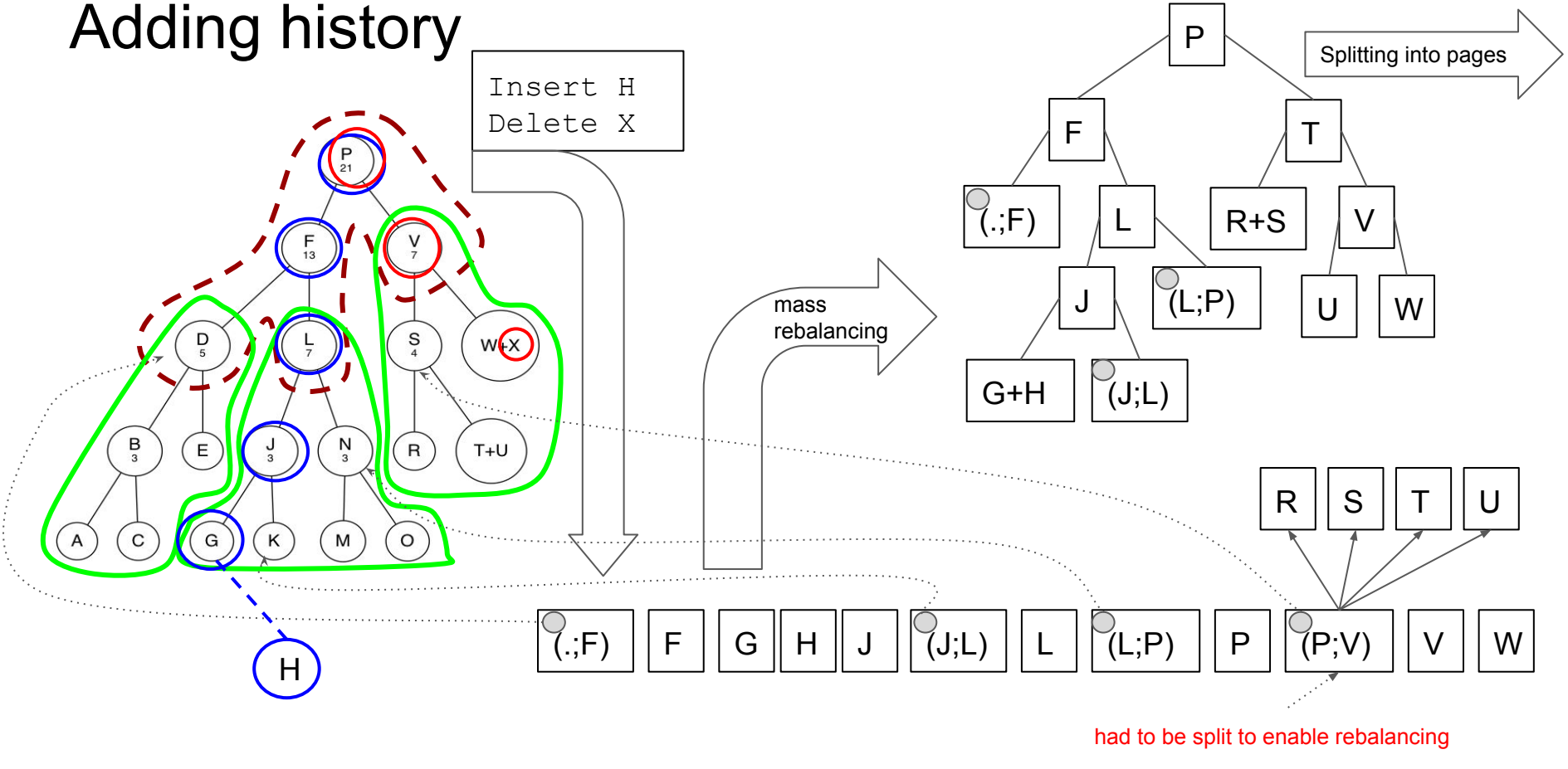
Splitting binary tree into pages of fixed size



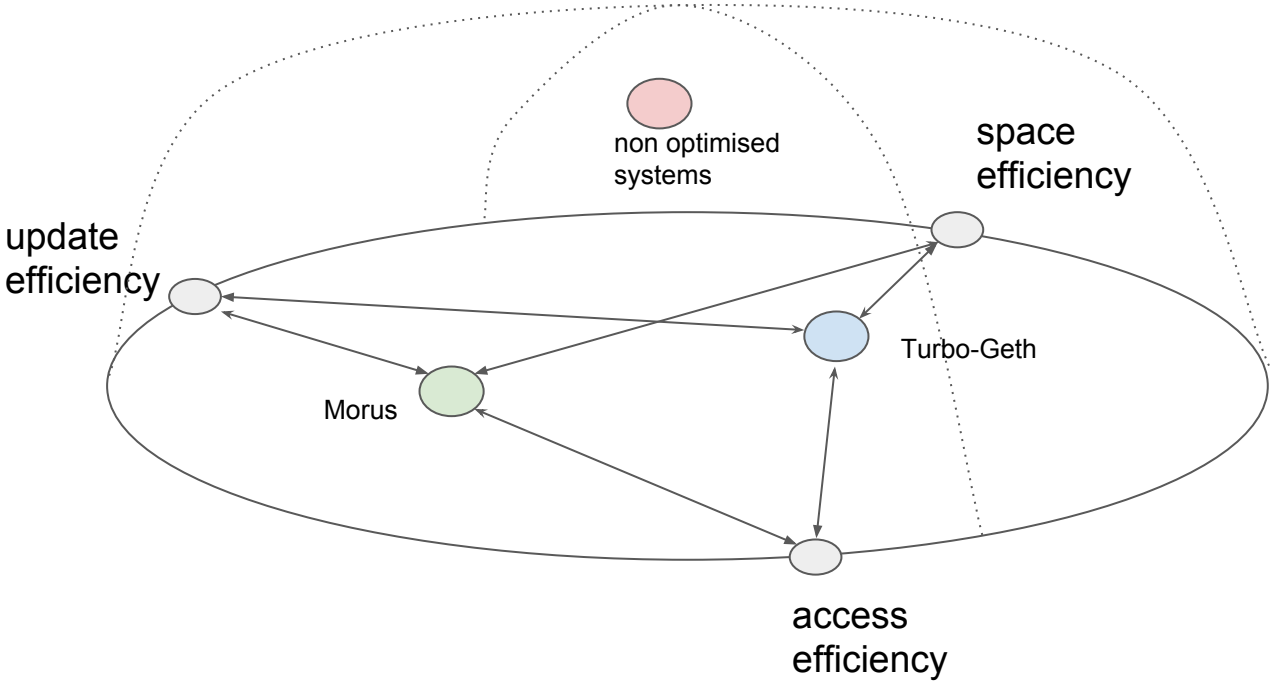




Adding history



Trade offs


non optimised
systems
space
efficiency
update
efficiency
Turbo-Geth
Morus
access
efficiency