

About Me

How I became a Bitcoin Developer

- Joined the Seoul Bitcoin Meetup in 2017
([meetup.com/seoulbitcoin](https://www.meetup.com/seoulbitcoin))

About Me

How I became a Bitcoin Developer

- Joined the Seoul Bitcoin Meetup in 2017
([meetup.com/seoulbitcoin](https://www.meetup.com/seoulbitcoin))
- Started contributing in 2019

About Me

How I became a Bitcoin Developer

- 2017년 서울 비트코인 맷업 참여
([meetup.com/seoulbitcoin](https://www.meetup.com/seoulbitcoin))
- Starting contributing in 2019
- Got funding in 2020

*"The most important book about technology today,
with implications that go far beyond programming."*
—Guy Kawasaki

Revised & Expanded

THE CATHEDRAL & THE BAZAAR

MUSINGS ON LINUX AND OPEN SOURCE
BY AN ACCIDENTAL REVOLUTIONARY



ERIC S. RAYMOND

WITH A FOREWORD BY BOB YOUNG, CHAIRMAN & CEO OF RED HAT, INC.

- No single roadmap

- No single roadmap
- Differing views on how to scale Bitcoin

- No single roadmap
- Differing views on how to scale Bitcoin
- I'll be sharing my take

- Anthony Towns - [www.erisian.com.au/wordpress/
2023/06/21/putting-the-b-in-btc](http://www.erisian.com.au/wordpress/2023/06/21/putting-the-b-in-btc)
- James O'Berine - [jameso.be/2023/08/14/scaling-
bitcoin.html](http://jameso.be/2023/08/14/scaling-bitcoin.html)

Scaling Bitcoin with Utreexo

Current limitations and overcoming them

<https://github.com/kcalvinalvin/2023-12-02-Taipei-Meetup>



**“We very, very much need such a system, but
the way I understand your proposal, it does
not seem to scale to the required size.”**

<https://www.mail-archive.com/cryptography@metzdowd.com/msg09963.html>

What makes Bitcoin special

**Something that can't
be negotiated**

Trustlessness

What is trustlessness?

Only trusting myself

“Of Bitcoin’s many properties, trustlessness, or the ability to use Bitcoin without trusting anything but the open-source software you run, is, by far, king”

<https://bluematt.bitcoin.ninja/2017/02/28/bitcoin-trustlessness/>

- Decentralization
- P2P
- Self-custody

**Things that hurt
trustlessness is rejected**

Using Bitcoin Trustlessly

Run a full node

Trustless usage

- Full node = fully validating



Run a full node

Trustless usage

- Full node = fully validating
- Validating transactions and blocks



Validation Process

What all full nodes go through

- Download 548GB of blocks

Validation Process

What all full nodes go through

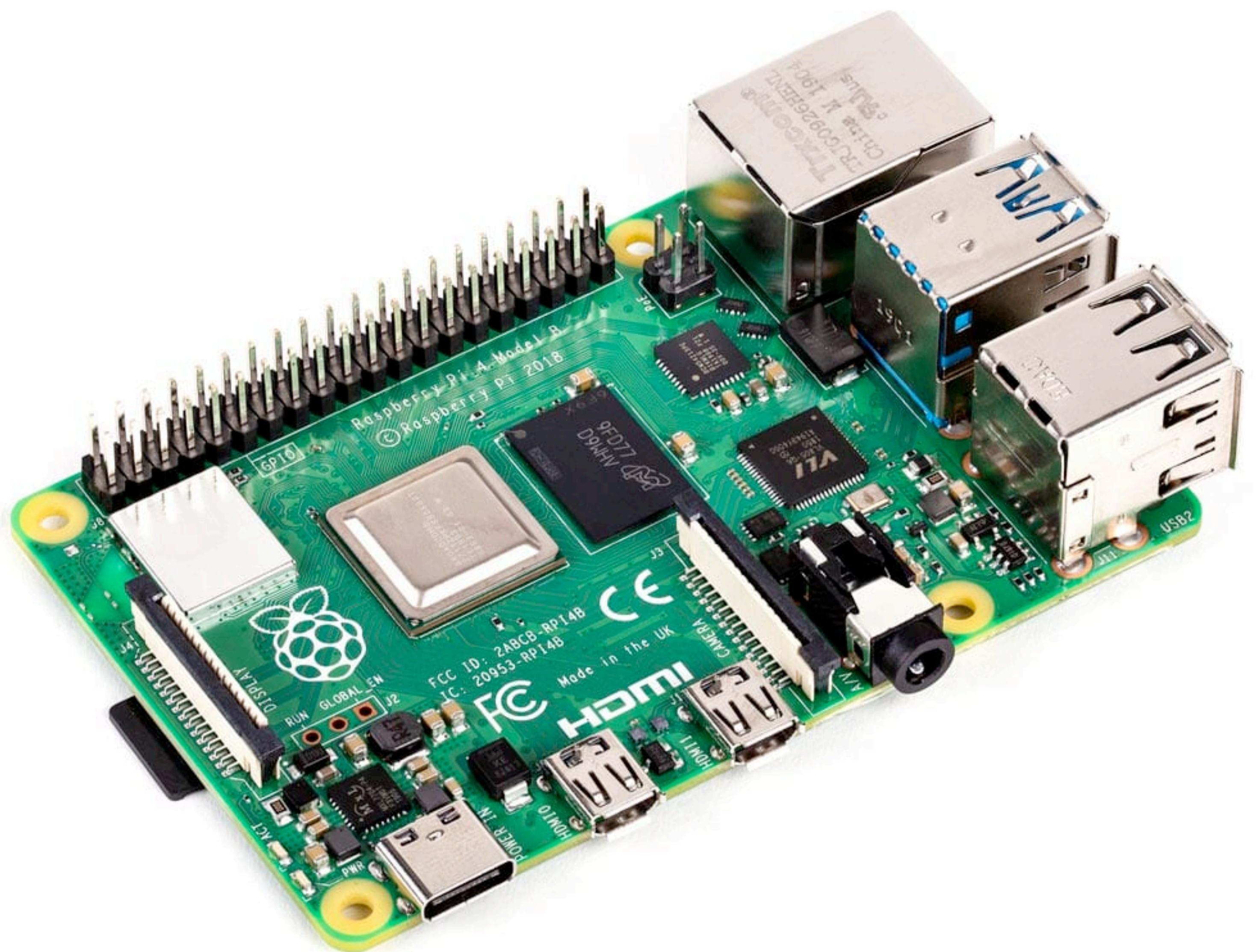
- Download 548GB of blocks
- Update the UTXO set every block

Validation Process

What all full nodes go through

- Download 548GB of blocks
- Update the UTXO set every block
- Validate the signature of the tx

Can everyone use it trustlessly?



Things limiting scalability

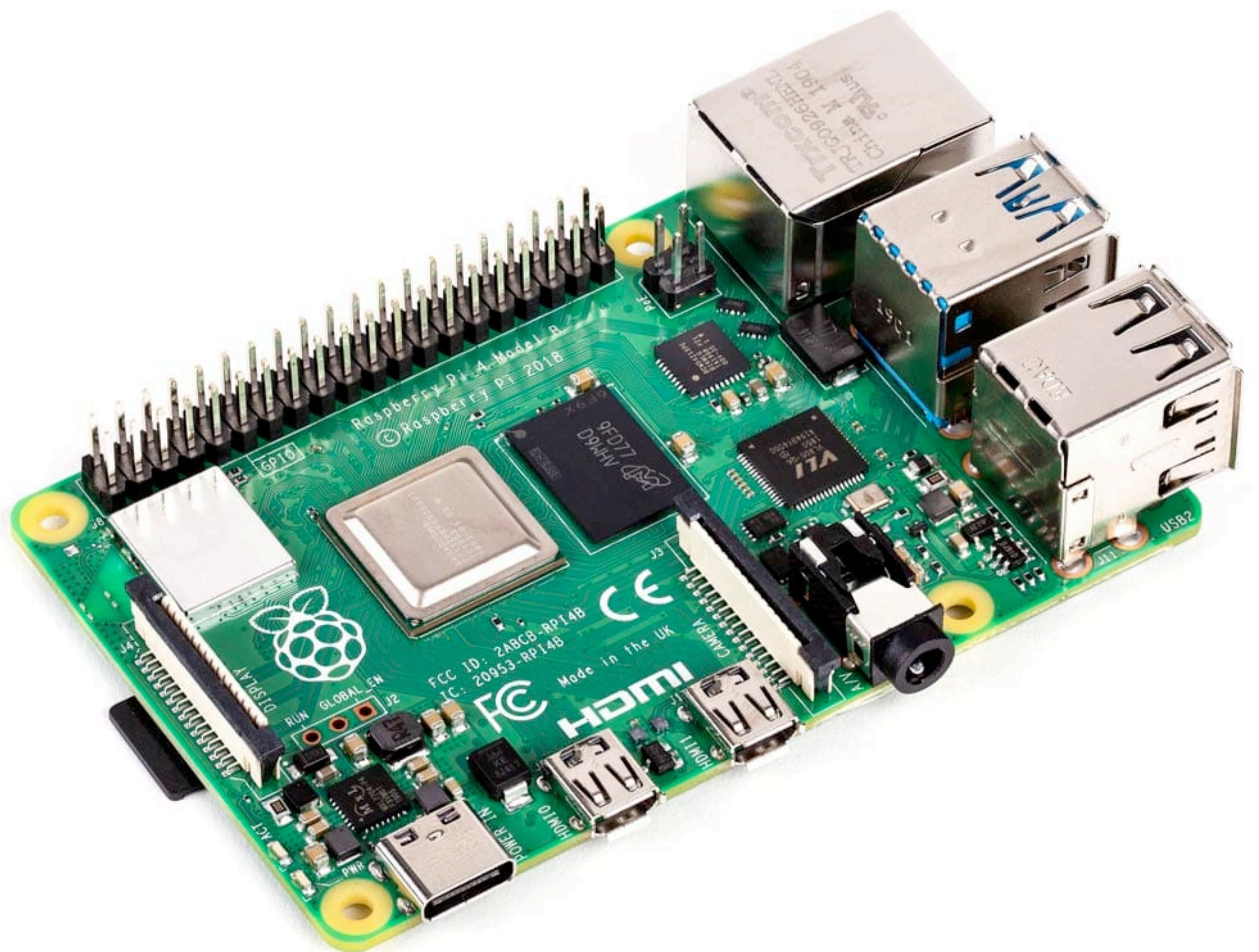
Scalability?

How much txs the system is able to process in a given time

**~2MB block every
10minutes**

2000-3000 transaction

**Limited by the slowest
computer**



How to scale Bitcoin

While keeping the trustlessness

- Use off-chain protocols

How to scale Bitcoin

While keeping the trustlessness

- Use off-chain protocols
- Increase the block size

Off-chain Protocols

The gist

- Increase the bang-for-the-buck

Off-chain Protocols

The gist

- Increase the bang-for-the-buck
- Multiple transfers within a single Bitcoin tx

Protocols with security tradeoffs

Off-chain Protocols

- Liquid Network
- Statechains
- Fedimint/cashu

Protocols without security tradeoffs

Off-chain Protocols

- Lightning Network
- Ark

Lightning Network

UX downsides during a high fee environment

- Need to make an expensive on-chain tx when getting started
- Need to make an expensive on-chain tx when there's liquidity issues

Ark

UX downsides during a high fee environment

- Exiting the Ark system is tricky when the fee is higher than the value of the VTXO

**High fees make L2s
suck**

My highly opinionated conclusion

**Eventually increase
the block size**

Currently blocked by the technical reasons

Why can't we increase the block size

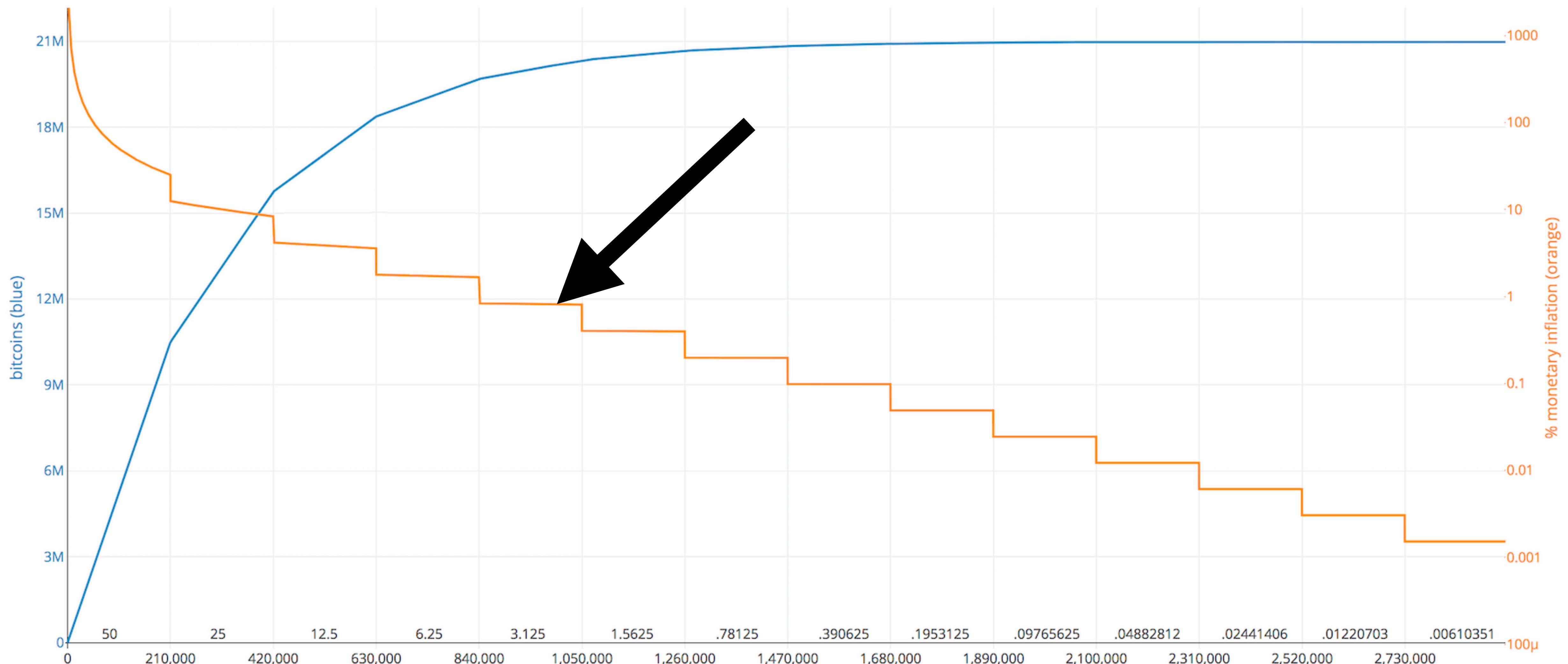
2 reasons

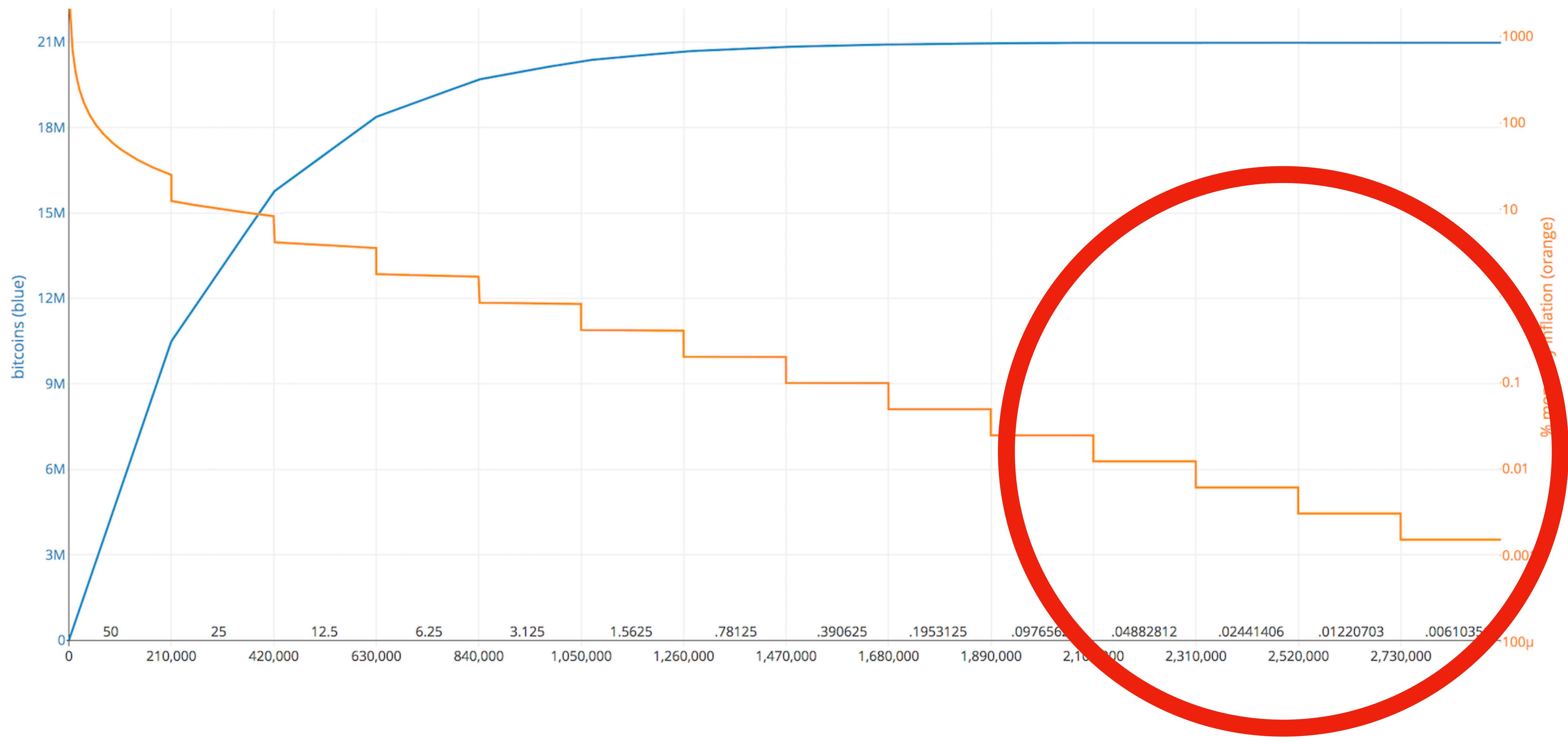
- Economical
- Technical

Economical reasons

Supply and demand

- Are there going to be enough fees?





Economical reasons

Supply and demand

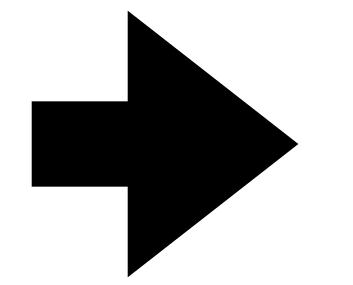
- Current double spend deterrents are mostly from the block rewards

Economical reasons

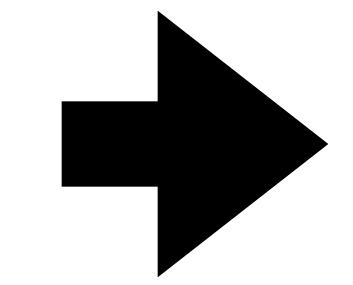
Supply and demand

- Current double spend deterrents are mostly from the block rewards
- As rewards go down, fees have to go up

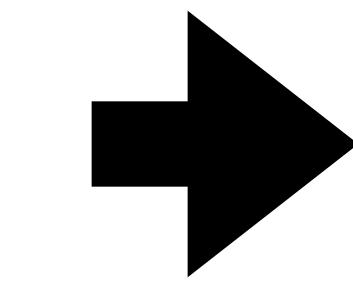
1



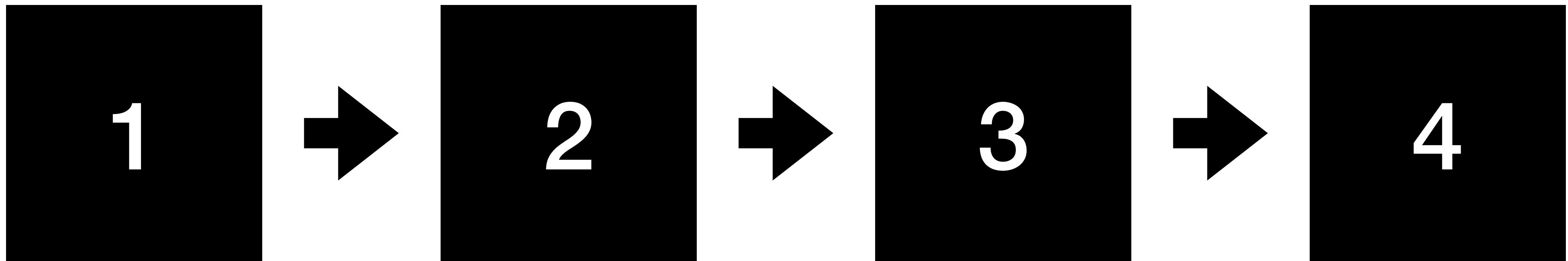
2



3



4



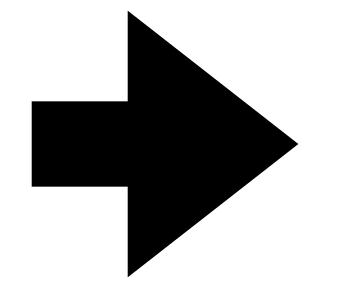
50 btc transaction

- Alice buys 50 btc with gold

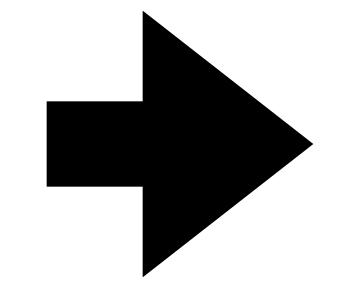
- Alice buys 50 btc with gold
- To revert the 50 btc tx, Alice bribes miner Bob with 10 btc

- Alice buys 50 btc with gold
- To revert the 50 btc tx, Alice bribes miner Bob with 10 btc
- Alice's profit - 40 btc + gold
- Bob's profit - 10 btc + block rewards

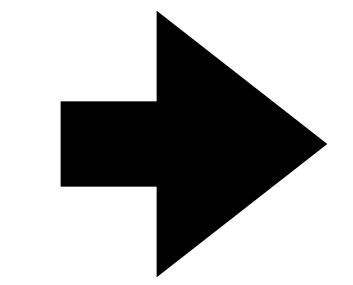
1



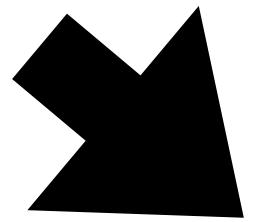
2



3

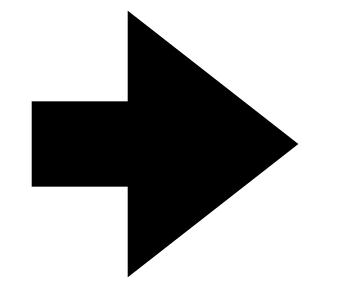


4

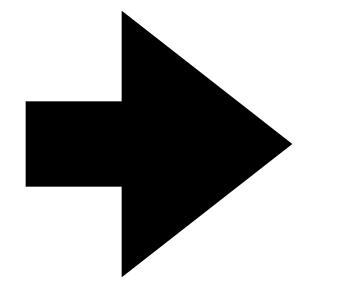


4b

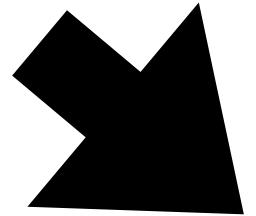
2



3

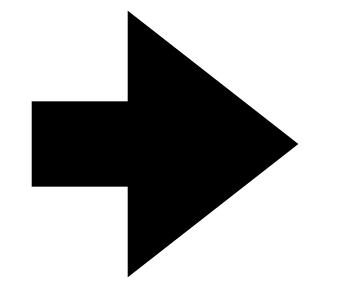


4

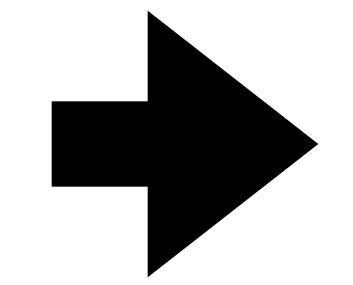


4b

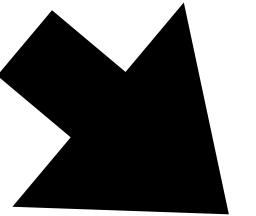
2



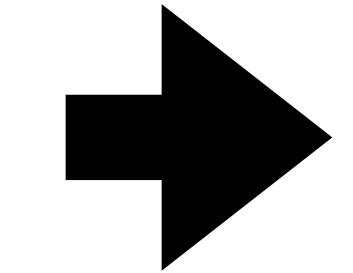
3



4



4b



5

Economical reasons

Supply and demand

- Enough fees need to exist to secure Bitcoin

Economical reasons

Supply and demand

- Enough fees need to exist to secure Bitcoin
- Blocksize ↑, Fees ↓

No research ongoing for
economic reasons

Technical reasons

Trustlessness

- Can everyone use Bitcoin trustlessly?

Technical reasons

Trustlessness

- Can everyone use Bitcoin trustlessly?
- Blocksize ↑, Cost of running full nodes ↑

Technical reasons

Process of Validation

- Download ~548GB of data
- Update the UTXO set per block
- Validate the signature of the tx

Technical reasons

Process of Validation

- Internet
- Disk
- CPU

Internet

Block / Transaction Validation

- Download ~500GB on the first start

Internet

Block / Transaction Validation

- Download ~500GB on the first start
- 100mbps Internet = ~666 minutes

Internet

Block / Transaction Validation

- Download ~500GB on the first start
- 100mbps Internet = ~666 minutes
- **1gbps = ~66 minutes**

Internet

Block / Transaction Validation

- Download ~500GB on the first start
- 100mbps Internet = ~666 minutes
- 1gbps = ~66 minutes
- 10gbps = ~6 minutes

Internet speeds are enough

Disk

UTXO set read/writes

- UTXO set - All the Bitcoin available to be spent

What's a UTXO set?

- TXO - Transaction Output

Typical TX

[8ad63210d059908229f00ad177469d9f16a5d37a7ea64cede9c92684ff7cf06e](#)

DETAILS



#0 [f758ad929fbefac99ef0d1f8b13990fa75efb517ef6c919178c8af7308f145](#) 0.04852489 BTC
07:1

#0 [17dSwJytZBszAafyWrK8Pue7rsRh9s5xeJ](#) 0.04033963 BTC

#1 [bc1qwqdg6squ38e46795at95yu9atm8azzmyvckulcc7kytlcckxswvvzej](#) 0.00758526 BTC

1 CONFIRMATION 0.04792489 BTC

Outputs of a TX

[8ad63210d059908229f00ad177469d9f16a5d37a7ea64cede9c92684ff7cf06e](#)

DETAILS +

#0 [f758ad929fbefac99ef0d1f8b13990fa75efb517ef6c919178c8af7308f145](#) 0.04852489 BTC
07:1

#0 [17dSwJytZBszAafyWrK8Pue7rsRh9s5xeJ](#) 0.04033963 BTC

#1 [bc1qwqdg6squ38e46795at95yu9atm8azzmyvckulcc7kytlcckxswvvzej](#) 0.00758526 BTC

1 CONFIRMATION 0.04792489 BTC

What's a UTXO set?

- TXO - Transaction Output
- UTXO - Unspent TXO. Not yet spent TXO

What's a UTXO set?

- TXO - Transaction Output
- UTXO - Unspent TXO. Not yet spent TXO
- UTXO set - All the UTXOs

Disk

UTXO set read/writes

- UTXO set is stored as a key-value database (leveldb)
- leveldb is random read/write

Disk

UTXO set read/writes

- Minimum 59B per UTXO
- Typical tx is 1 input, 2 output
- For 2000txs: ~2000 read, ~4000 write

기술의 경이로움이 시작된다

Platinum P41 SSD

科技的奇蹟開始了



SK하이닉스 Platinum P41 SSD는 업계 최고 수준의 성능으로
차세대 컴퓨팅 시스템을 완성하는 제품입니다. PCIe 4세대 NVMe 인터페이스와
176단 NAND의 한층 더 강화된 기술력을 토대로 최강의 데이터 전송 속도와
성능을 제공합니다. 이제 차원이 다른 성능을 경험해 보세요.

성능

2TB

순차성능
(최대)

읽기 : 7,000 MB/s
쓰기 : 6,500 MB/s

랜덤성능
(최대)

읽기 : 1,400K IOPS
쓰기 : 1,300K IOPS

TBW

최대 1,200TBW

1TB

읽기 : 7,000 MB/s
쓰기 : 6,500 MB/s

읽기 : 1,400K IOPS
쓰기 : 1,300K IOPS

최대 750TBW

500GB

읽기 : 7,000 MB/s
쓰기 : 4,700 MB/s

읽기 : 960K IOPS
쓰기 : 1,000K IOPS

최대 500TBW

- 1,400 IOPS = ~5.6MB/s read
- 1,300 IOPS = ~5.2MB/s write
- Per block, 0.11MB read, 0.23MB write

$$(1,400 * 4) / 1000. \text{ 4 is the OS page size (mb)}$$



- 73 IOPS = ~0.29MB/s read&write
- Per block 0.11MB read, 0.23MB write

Caching

Current way of reducing the disk burden

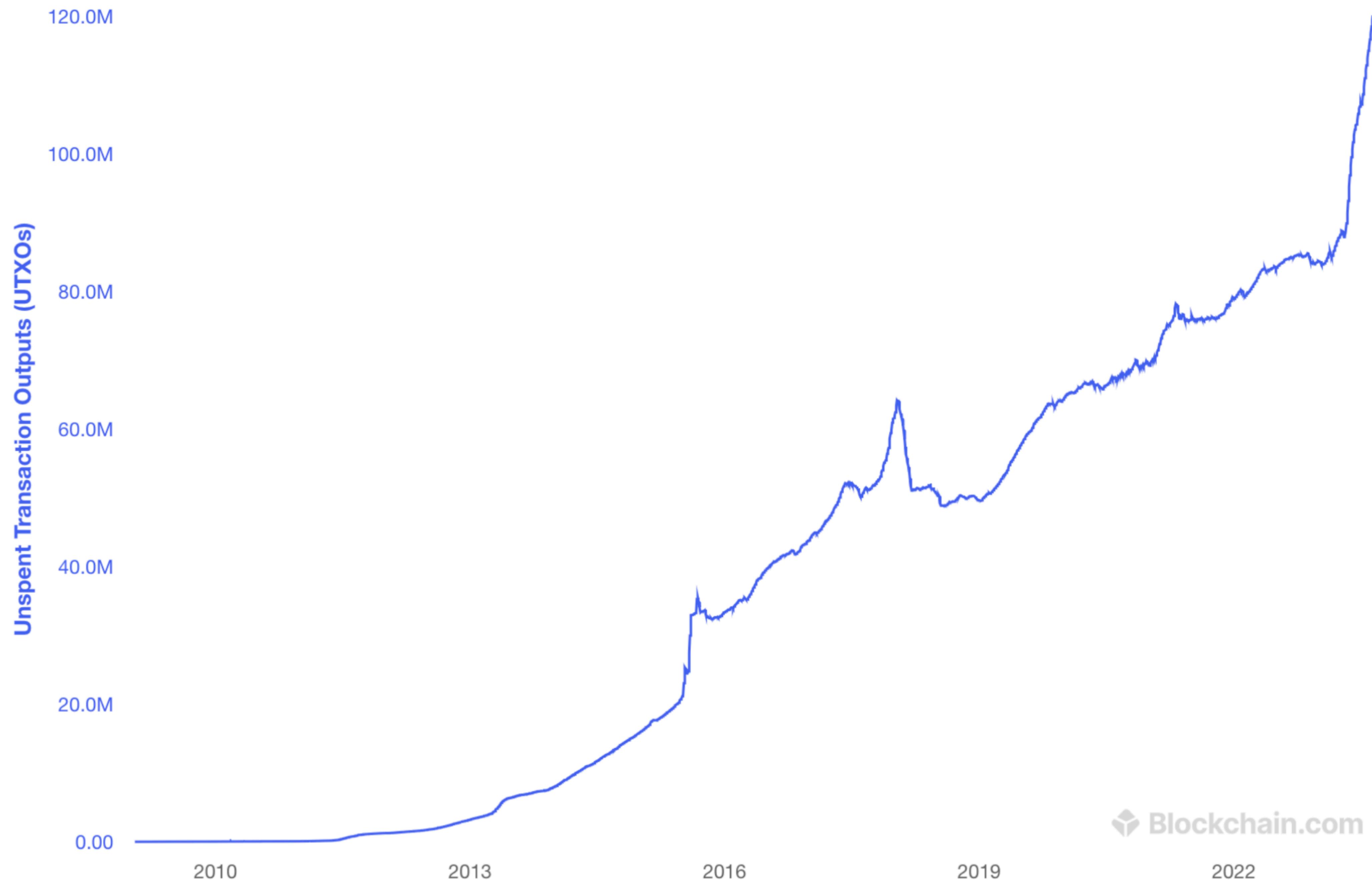
- Write is fast (only to RAM)
- Read is sped-up (if cached)

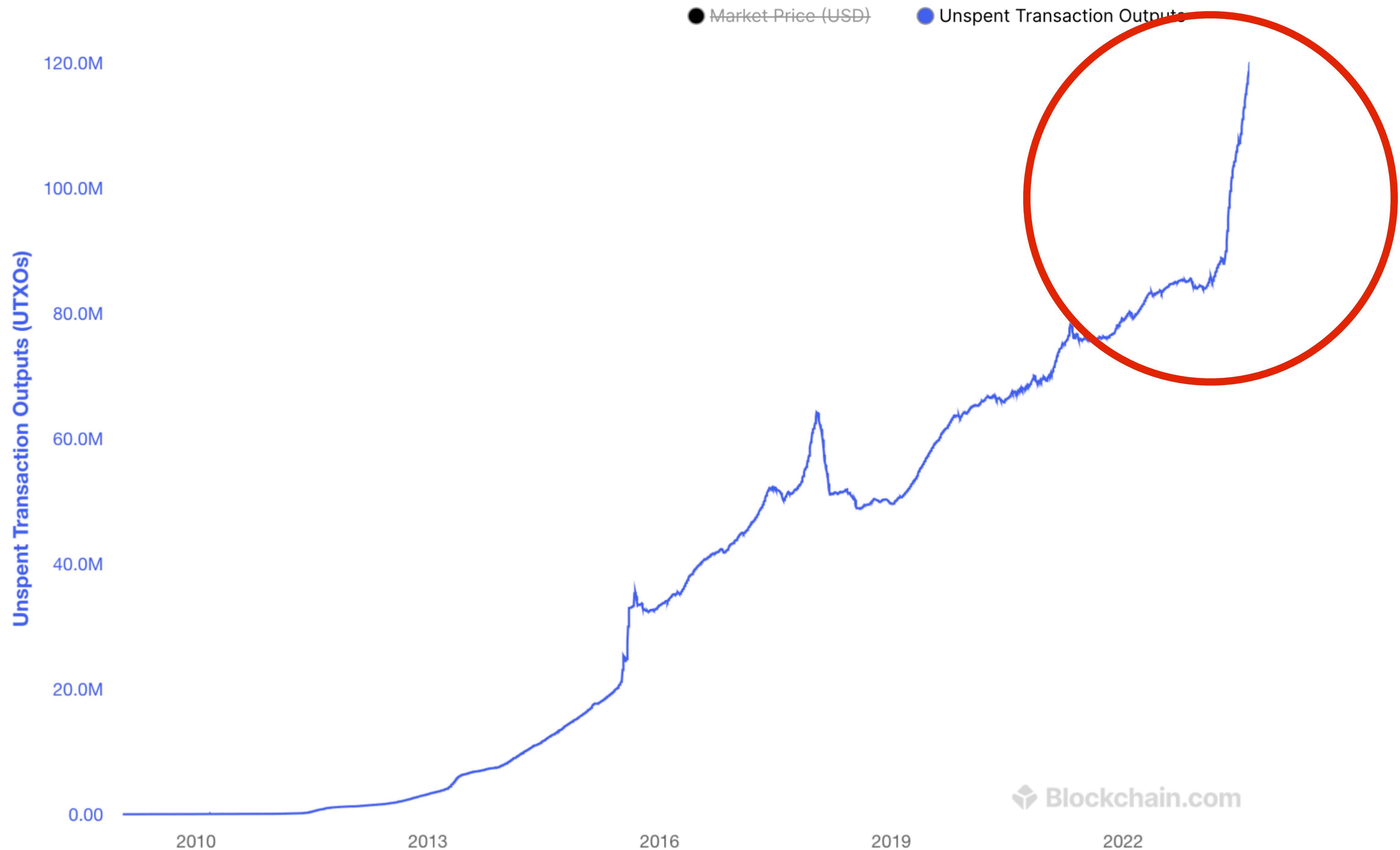
Caching

Current way of reducing the disk burden

- As the UTXO set grows, the RAM size limit will make optimization less effective

● Market Price (USD) ● Unspent Transaction Outputs





**As the UTXO set grows, it
gets slower**

**UTXO set size has no
bounds**

8 billion UTXO, 59B each

472GB

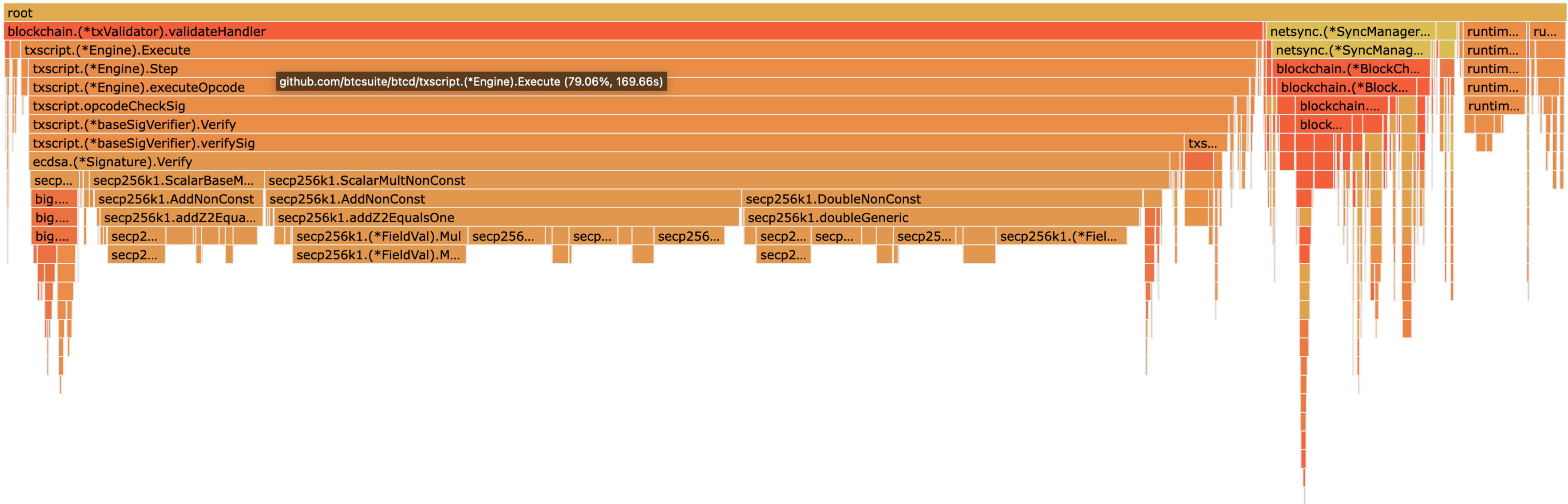
Already bottleneck for HDD

CPU

Transaction signature validation

- Signature operations take up a bulk of the block validation

github.com/btcsuite/btcd/txscript.(*Engine).Execute (79.06%, 169.66s)



**80% time on sig
validation**

For btcd

CPU

Transaction signature validation

- CPU cores increasing
- Taproot txs can be batch verified

libsecp256k1

[build](#) [passing](#) [dependencies](#) [none](#) [irc.libera.chat](#) [#secp256k1](#)

Optimized C library for ECDSA signatures and secret/public key operations on curve secp256k1.

This library is intended to be the highest quality publicly available library for cryptography on the secp256k1 curve. However, the primary focus of its development has been for usage in the Bitcoin system and usage unlike Bitcoin's may be less well tested, verified, or suffer from a less well thought out interface. Correct usage requires some care and consideration that the library is fit for your application's purpose.

Features:

- secp256k1 ECDSA signing/verification and key generation.
- Additive and multiplicative tweaking of secret/public keys.
- Serialization/parsing of secret keys, public keys, signatures.
- Constant time, constant memory access signing and public key generation.
- Derandomized ECDSA (via RFC6979 or with a caller provided function.)
- Very efficient implementation.
- Suitable for embedded systems.
- No runtime dependencies.
- Optional module for public key recovery.
- Optional module for ECDH key exchange.
- Optional module for Schnorr signatures according to [BIP-340](#).

**Already CPU bottlenecked
but rooms for improvement**

**What do we need to increase the
block size?**

Signature validation ↑
UTXO read/write ↑

How to improve read/write?

Utreexo

What I work on!

Merkelize the UTXO set

**~Same performance on
HDD and SSD**

disk read/write X

No more random reads

Testing the performance

Test methodology

- Use an option that does a lot of read/writes and compare

36% slower

On current nodes

3% slower

On Utreexo nodes

The writes that happen for Utreexo

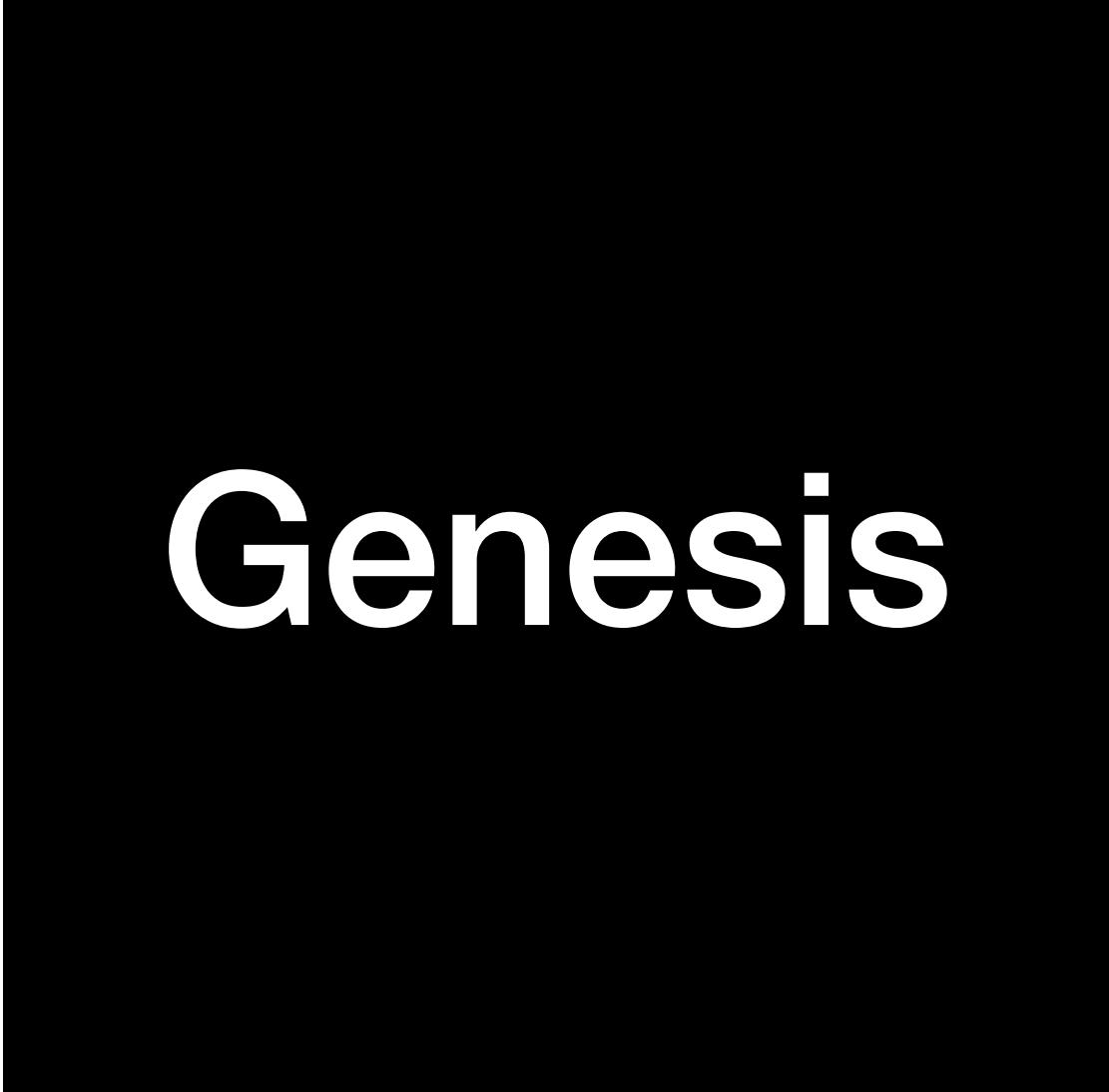
No reads

```
{443000, newHashFromStr("000000000000000047efc3c126c3398cfdef1609682d1492cd2909e3c0a17f"),  
    43243051, []*chainhash.Hash{  
        newLeafHashFromStr("df31c496a54a8021c38c77809da15a8bd3968970315f5a56c07f0fd9763262ff"),  
        newLeafHashFromStr("755f4cfbf5727c3e14aa6dcc6e5ab150c61101cead06fade5a886ee8d1dd377"),  
        newLeafHashFromStr("c572a92c7cd9e14ce5d7c94473ef90bee21856654eed5535314b3c117006bc83"),  
        newLeafHashFromStr("9e85567347ef0ea38c1797d8ed8e399ed0d99a07025388f6e06791ff6b88643c"),  
        newLeafHashFromStr("ab4efc738bd7b8281689e899979d44ed0ca8fe1489efa0f5a09f5b1c276fe792"),  
        newLeafHashFromStr("1c763d8f490baef0b756597dfc2da5a31d0aee324da6314b52f21707351cf51a"),  
        newLeafHashFromStr("29fcd9989939a17e5d113fb1f247f3bd05606488bcb48c9cfad55a57006f2248"),  
        newLeafHashFromStr("e79f840fe1bcd6b638a53b637885dda7ea3da2b2b5f1f5447ebf4d54fc0a2e"),  
        newLeafHashFromStr("02f33c3ce0d684e644410fa35fee4c7572454facc710389ea0f69a00e27c1780"),  
        newLeafHashFromStr("4e29ad940088ab42991bd99d7943a68ea38d498f8435c4bea21b8252788b5a7d"),  
        newLeafHashFromStr("feabfc6565e918827df8ade12d0885e7da28bb81410c71a0fc92da28e5538e"),  
        newLeafHashFromStr("42275481edcd150c0891f8f7486941e7ecc9dbec0d1554d1ae3027d8523571b4"),  
        newLeafHashFromStr("84dd84bbb564ab9e878ea774d66a2cb818c2b74680974e01a07dde68bd064bfd"),  
        newLeafHashFromStr("19a64f2f77403d4a1e3482b3a1801aa360120f6429f5ed47261fc8676920d9d"),  
    },  
},  
},
```

Out of Order Block Validation

Current Block Validation

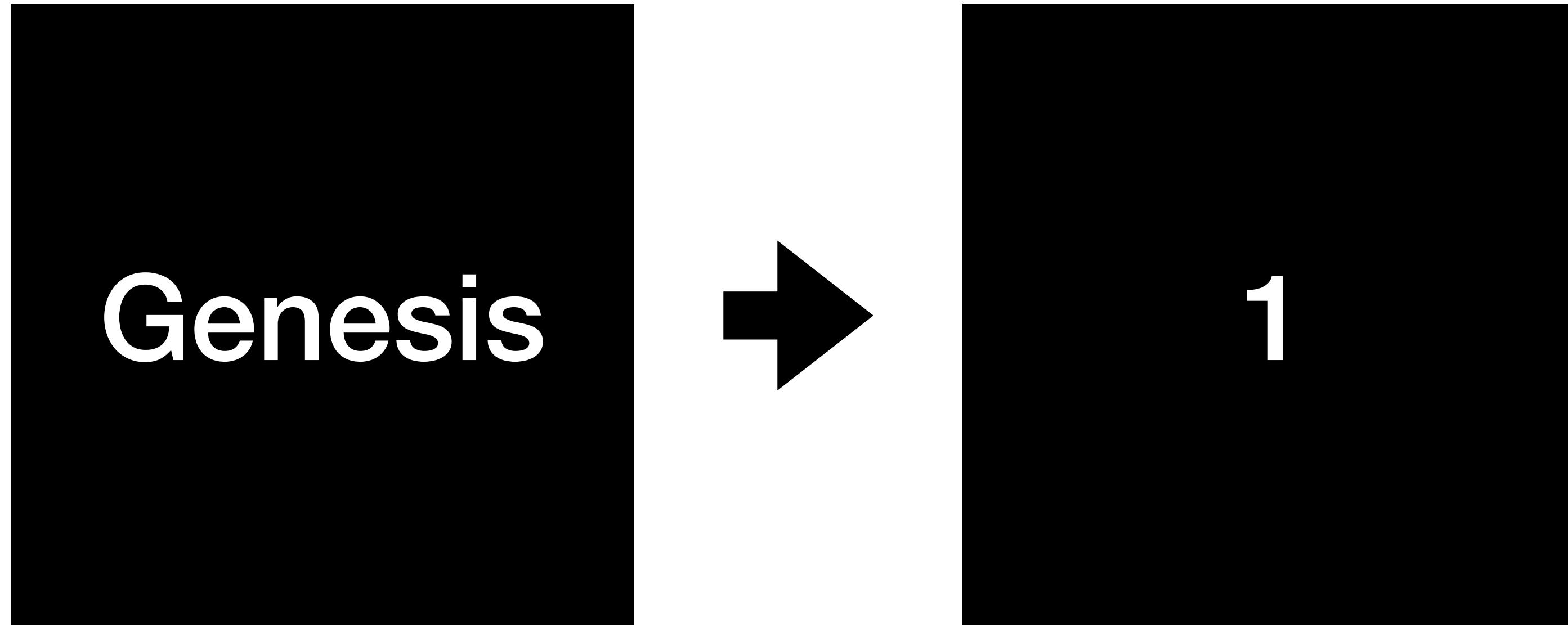
Sequential



Genesis

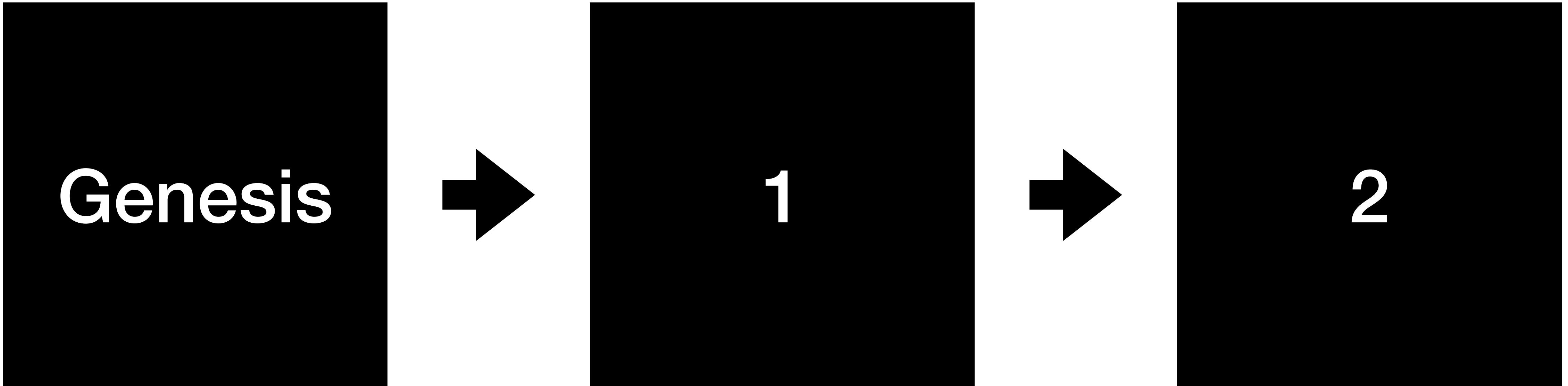
Current Block Validation

Sequential



Current Block Validation

Sequential



Out of Order Block Validation

Doesn't have to be sequential

- Provide the merkle roots of the Utreexo at each block
- Use them to validate the next block in line

Out of Order Validation

Genesis

100,000

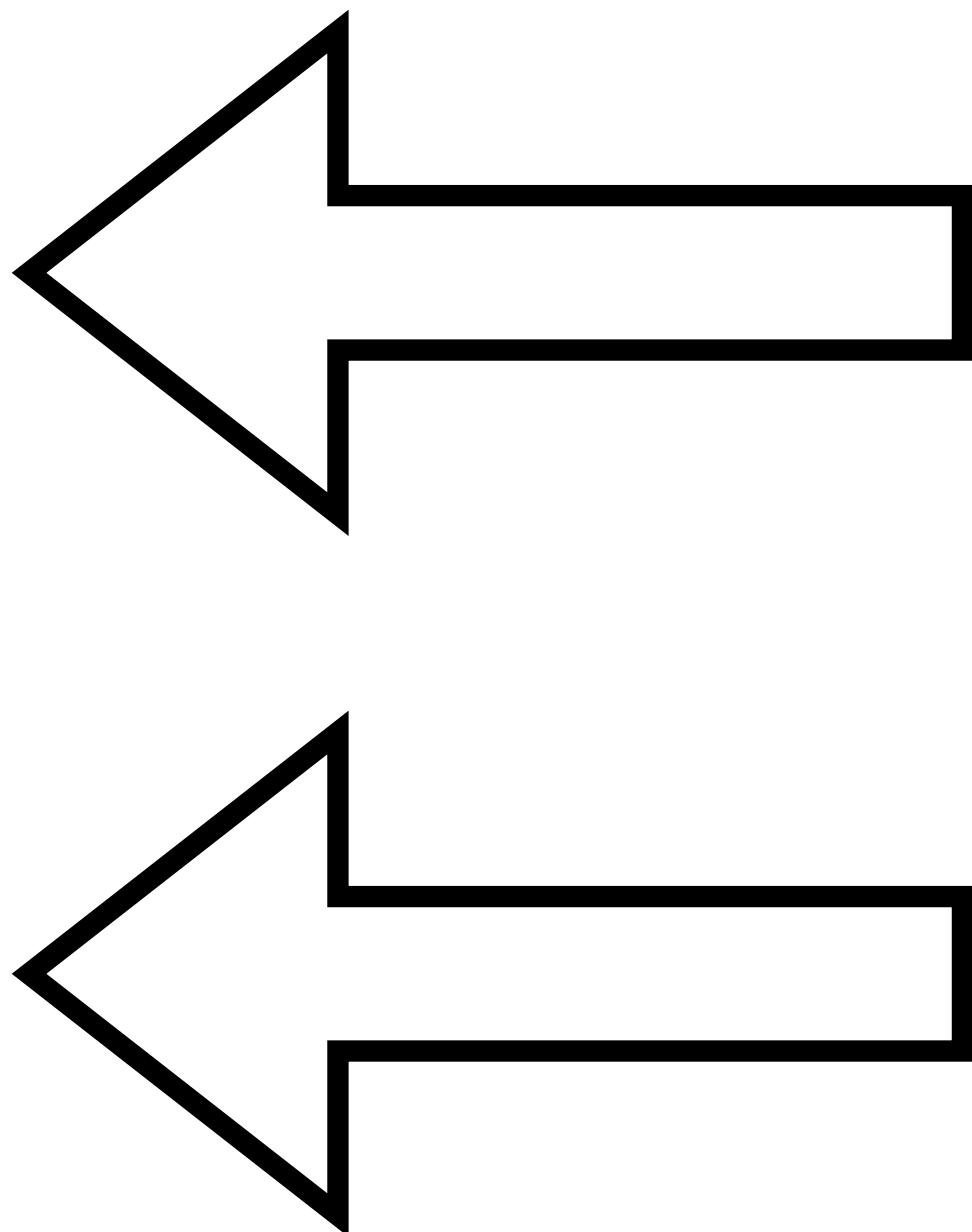
200,000

Out of Order Validation

Genesis

100,000

200,000



Roots committed in
binary

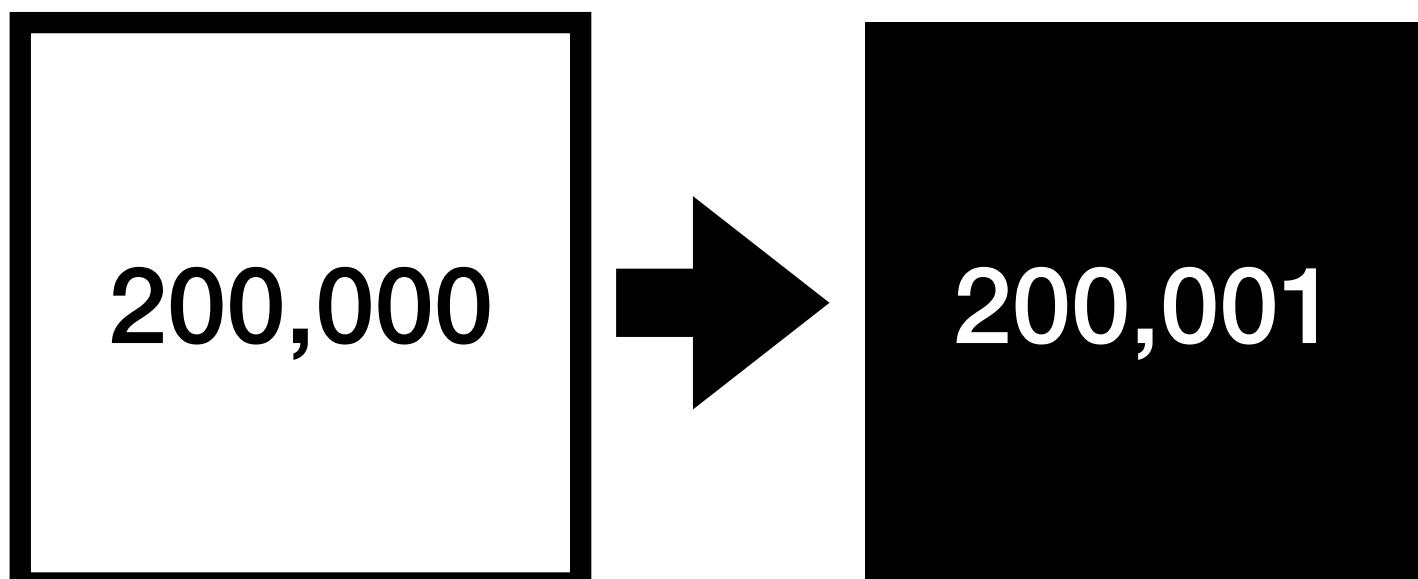
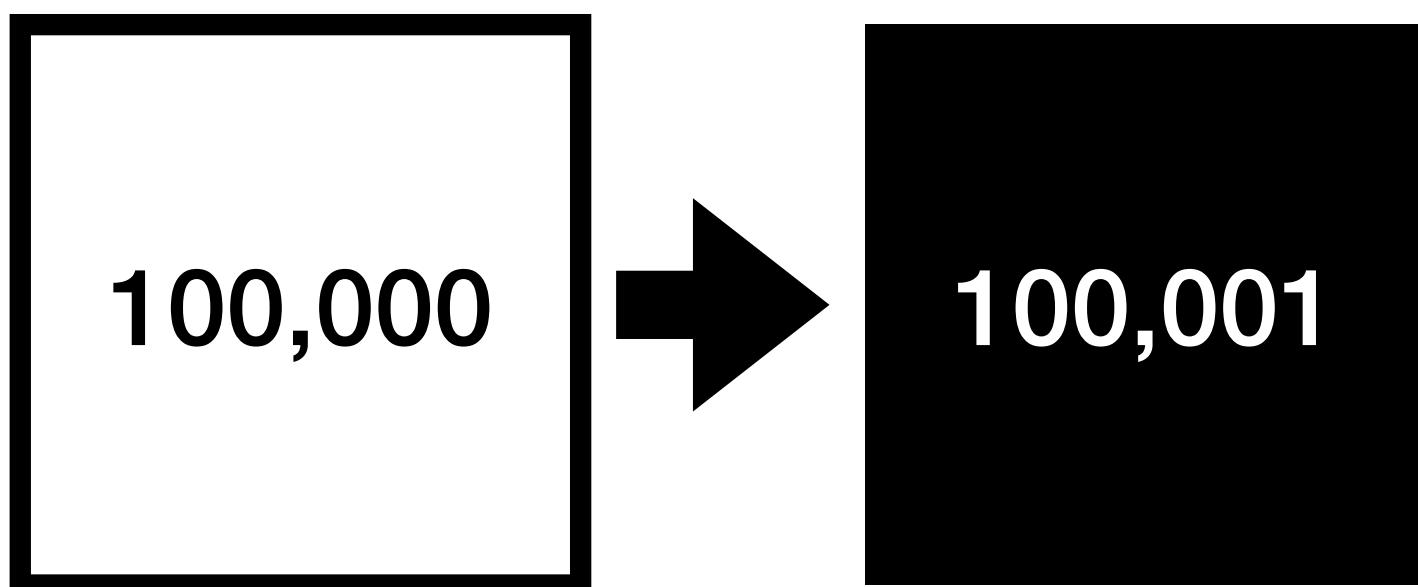
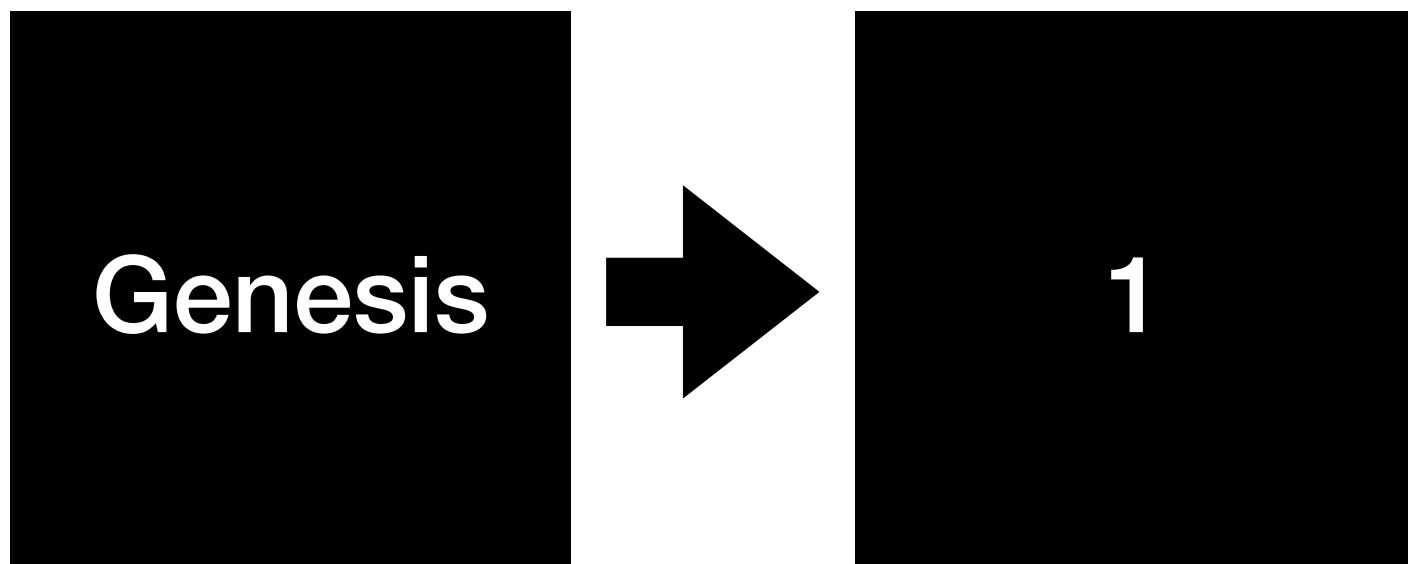
Out of Order Validation

Genesis

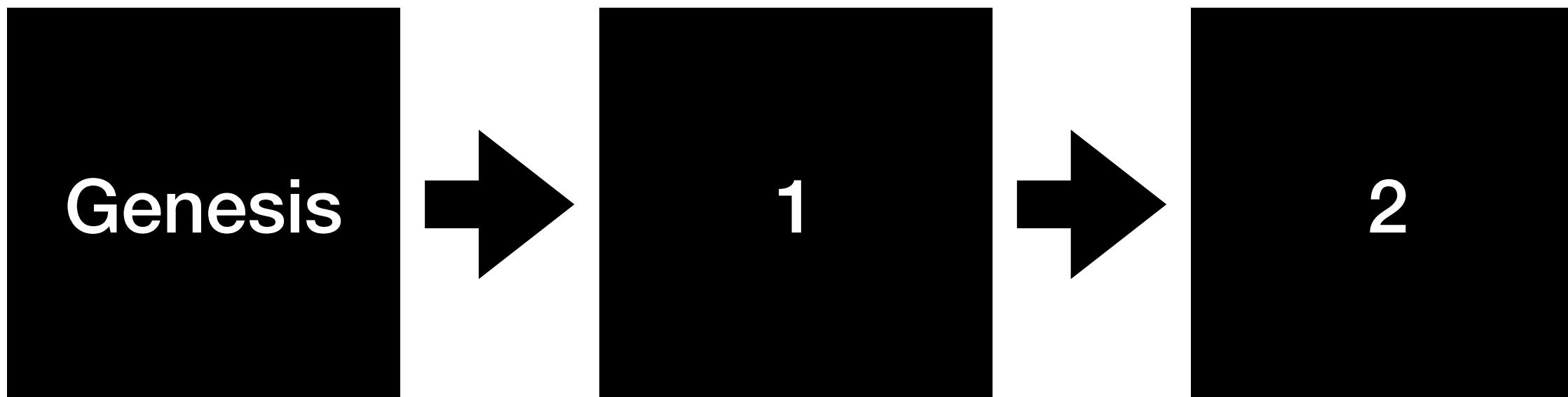
100,000

200,000

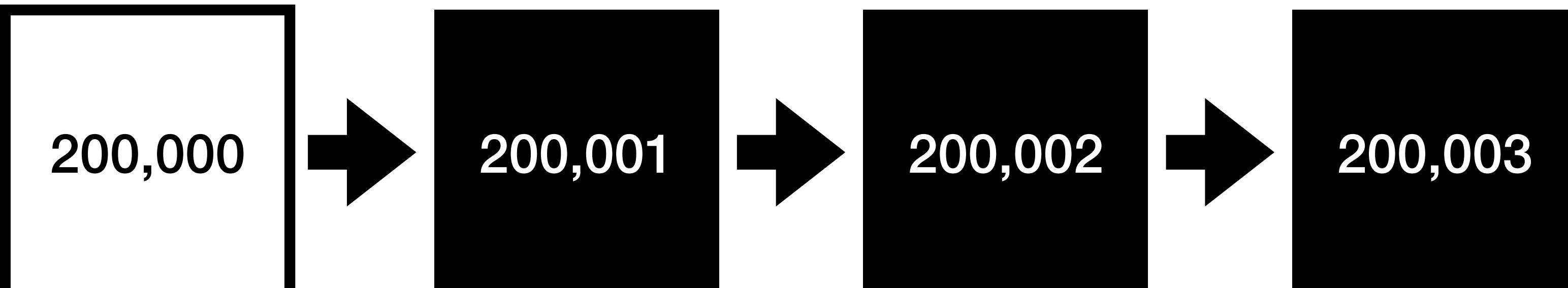
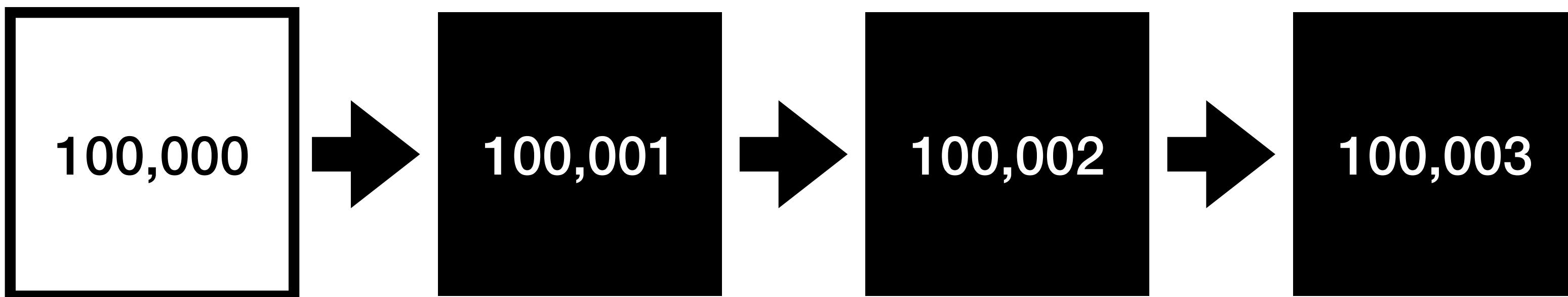
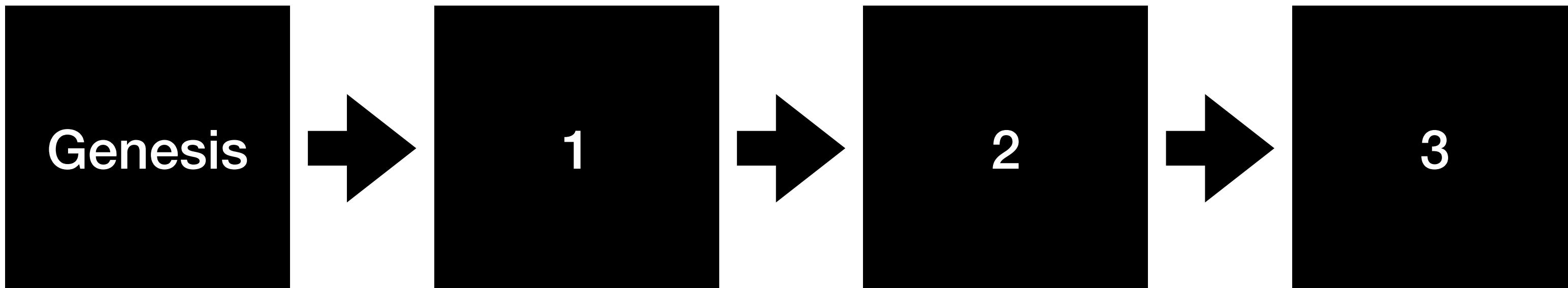
Out of Order Validation



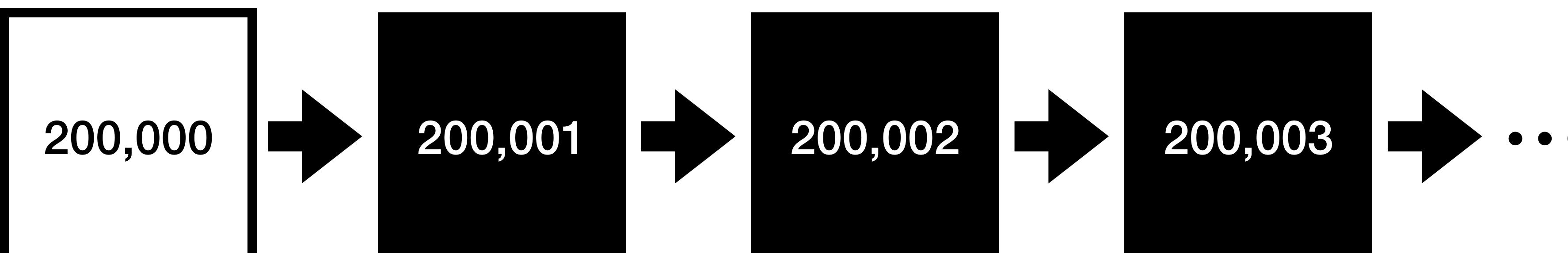
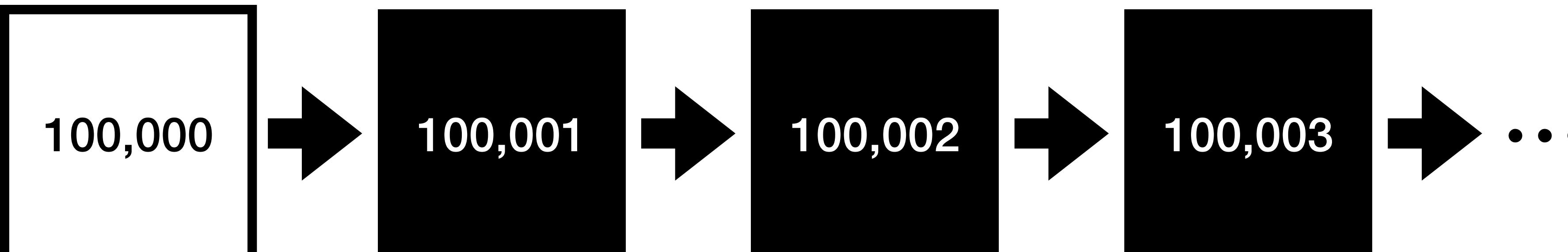
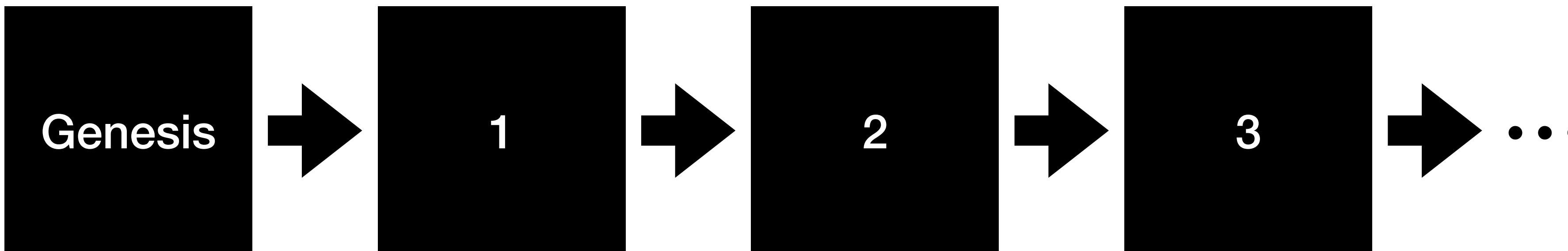
Out of Order Validation



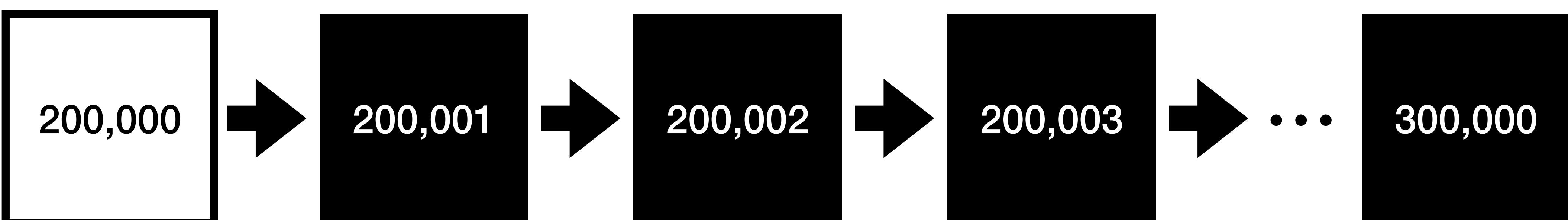
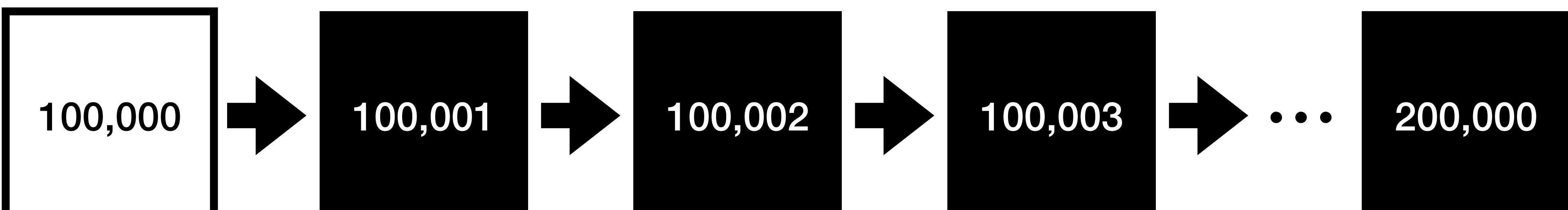
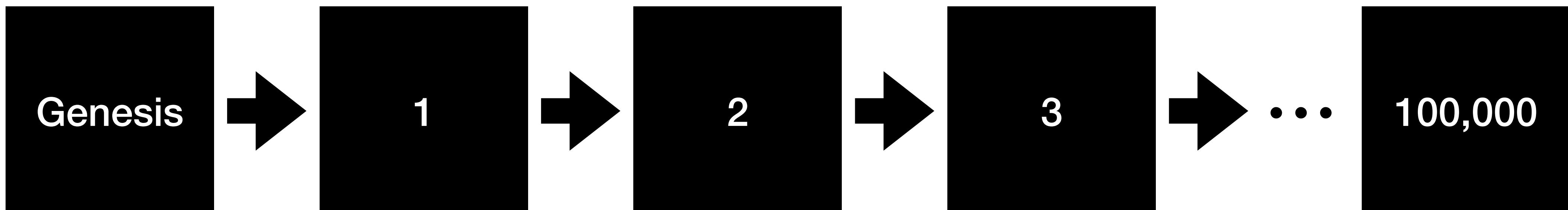
Out of Order Validation



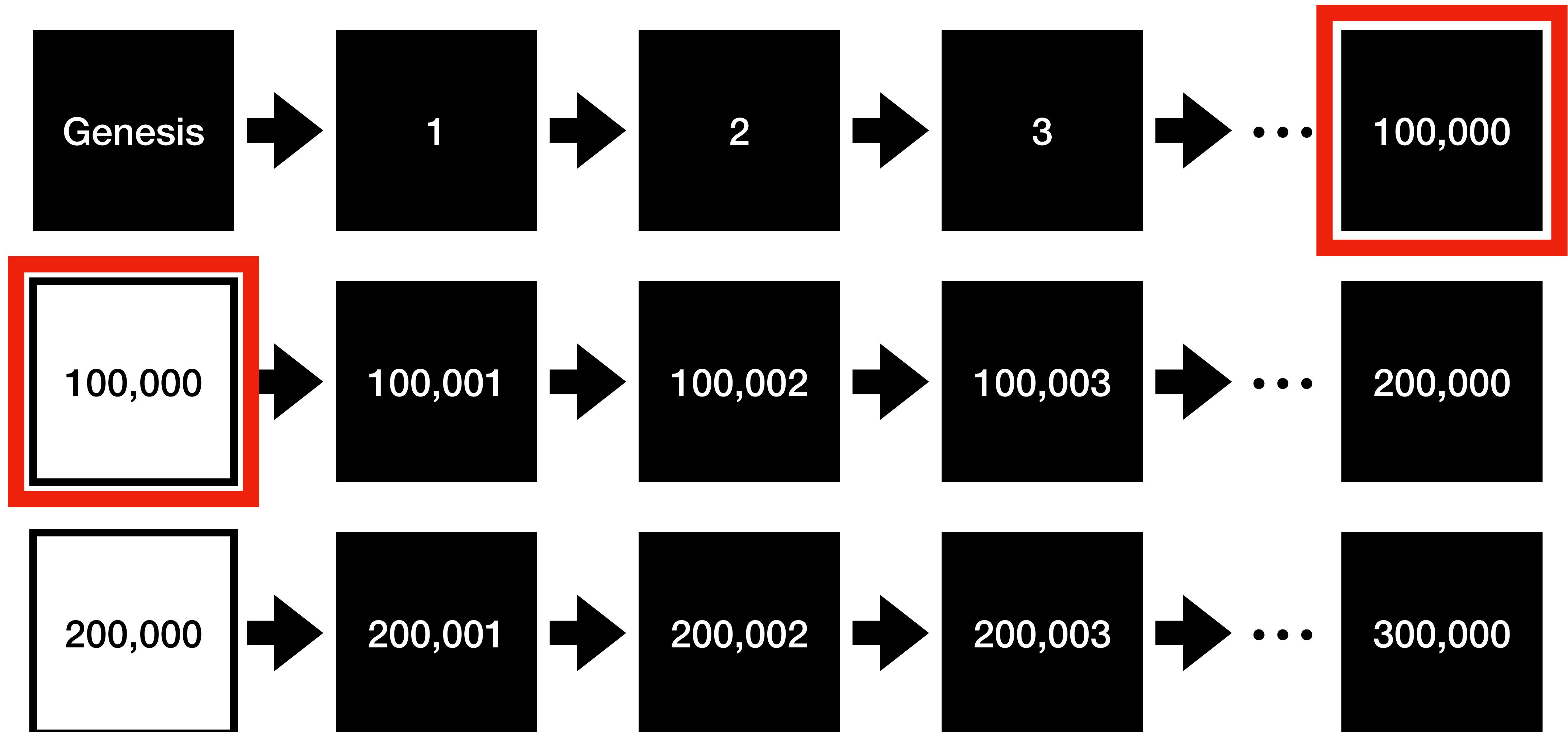
Out of Order Validation



Out of Order Validation

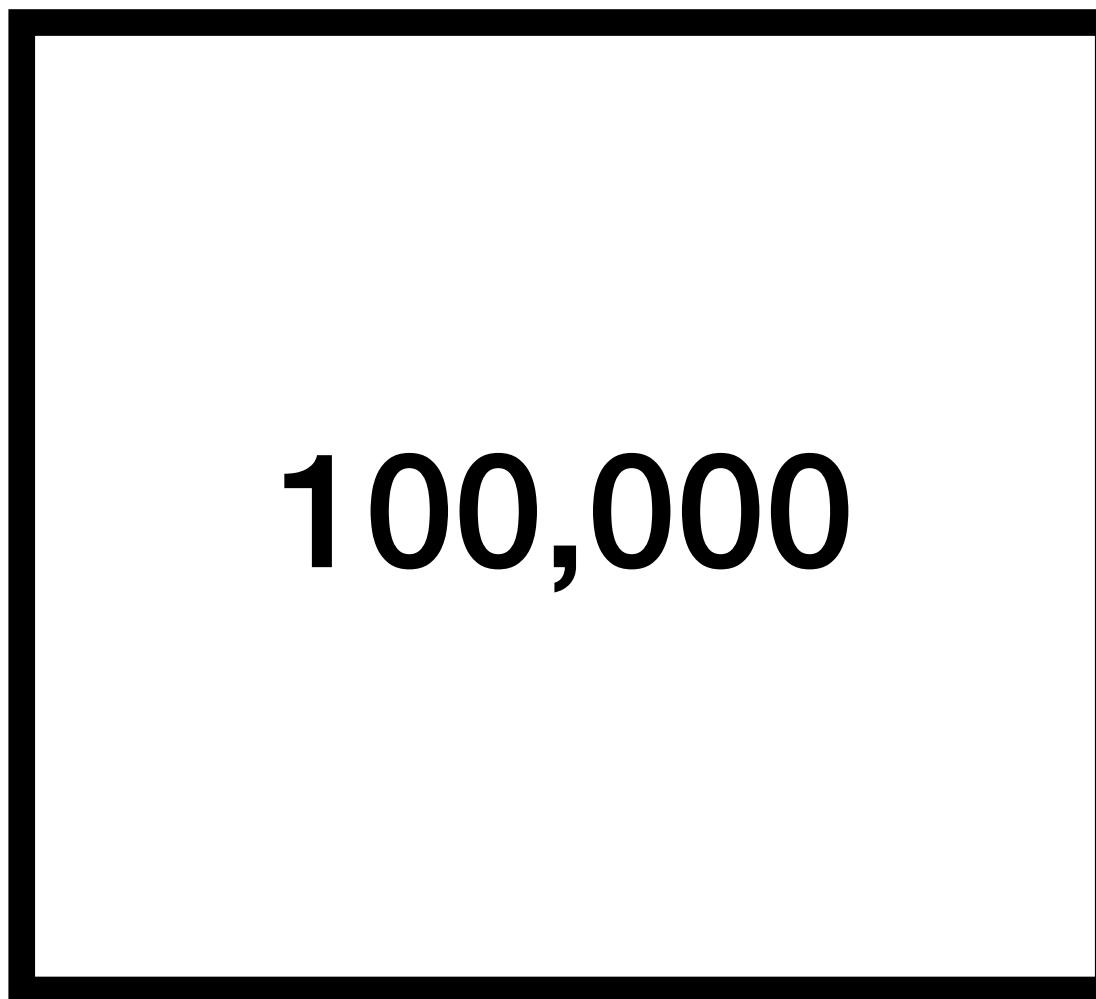


Out of Order Validation

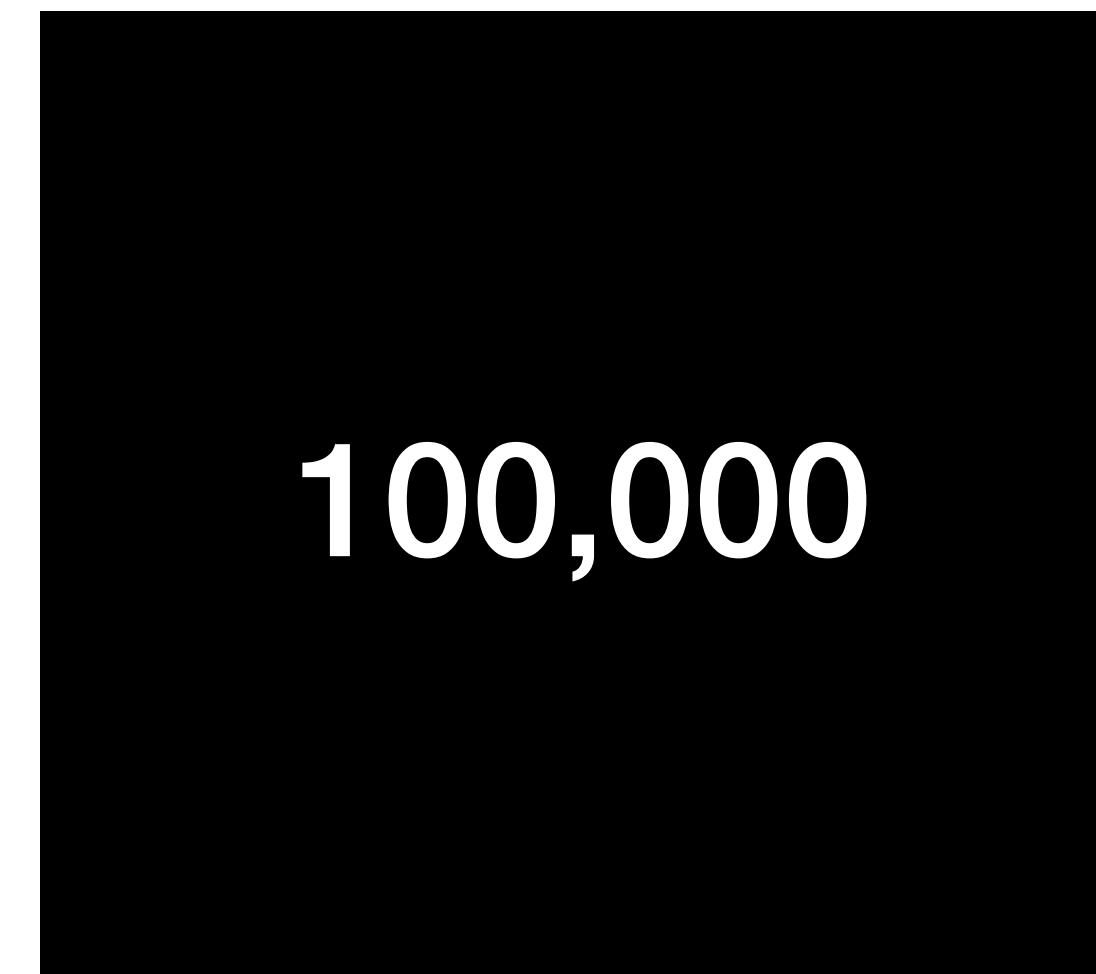


Compare Merkle Roots

Check if the generated roots at 100,000 is the same

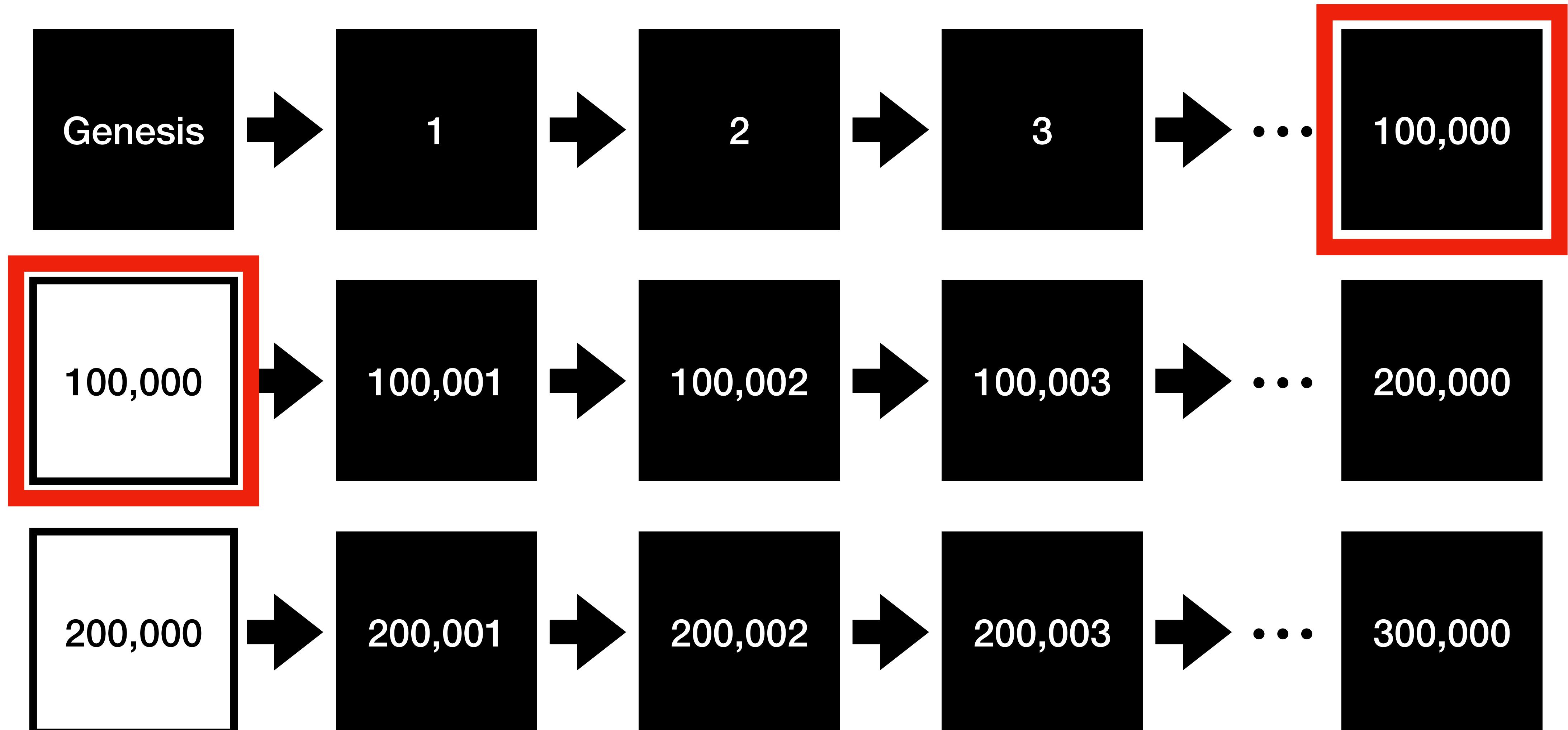


==



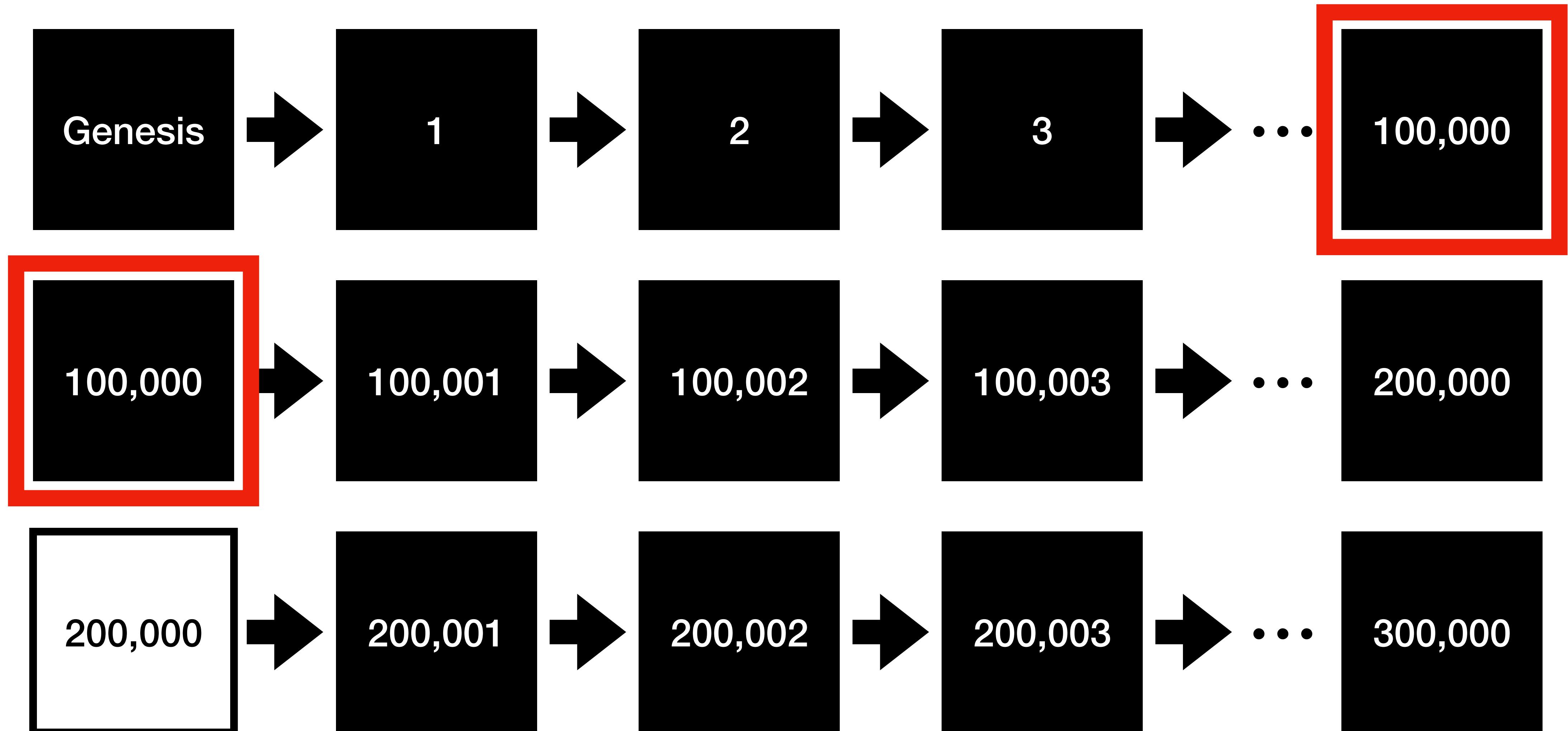
With Utreexo

Efficient parallel validation

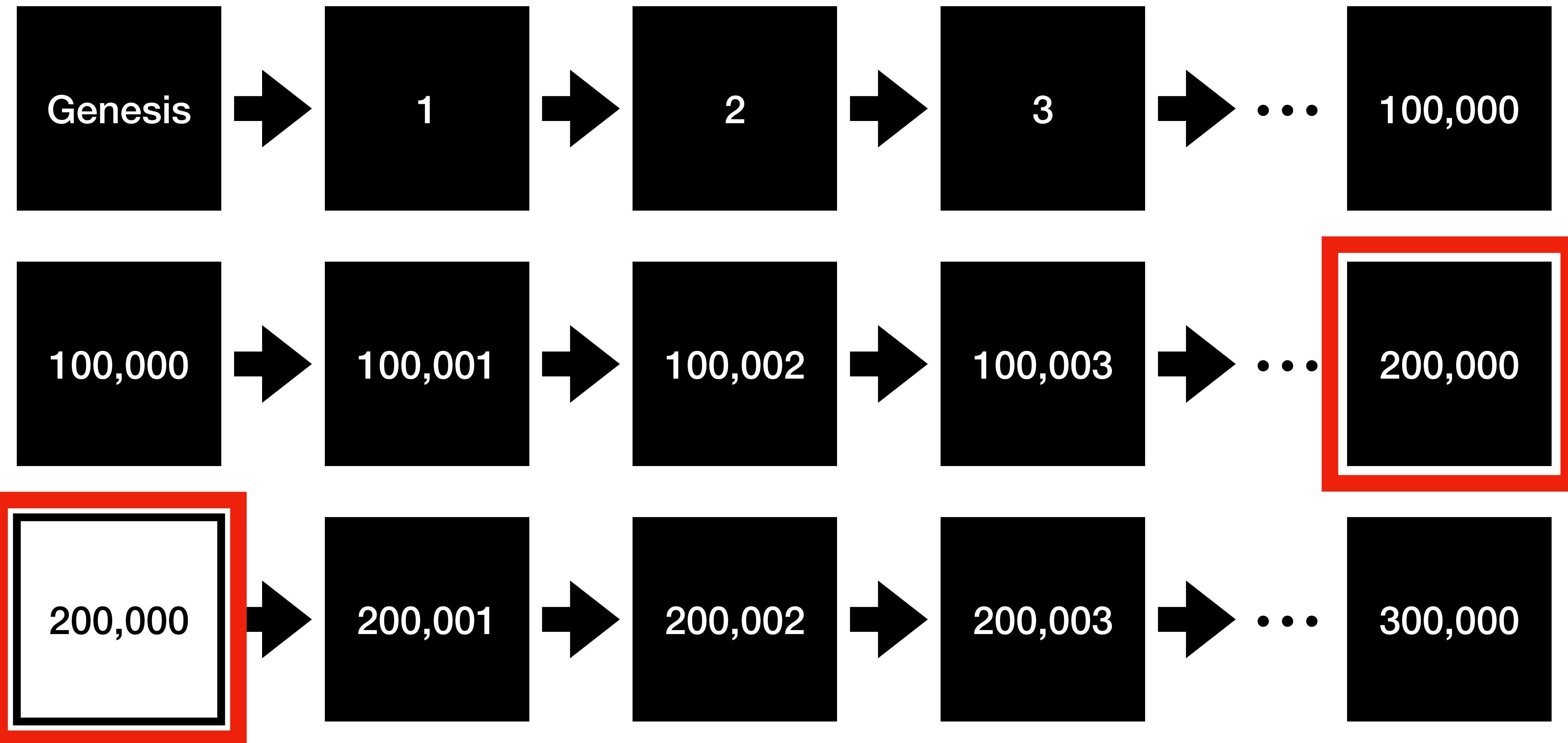


With Utreexo

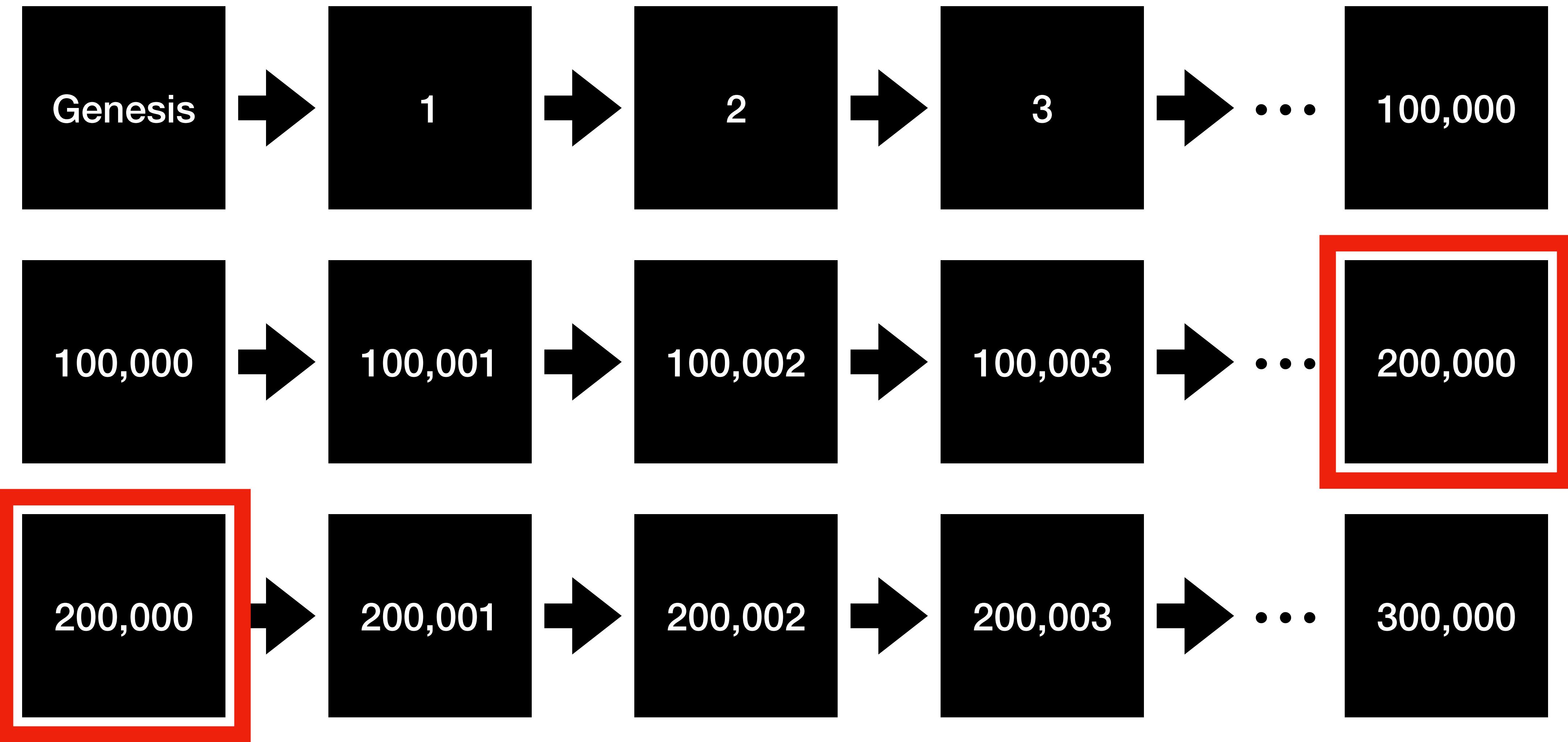
Efficient parallel validation



Out of Order Validation

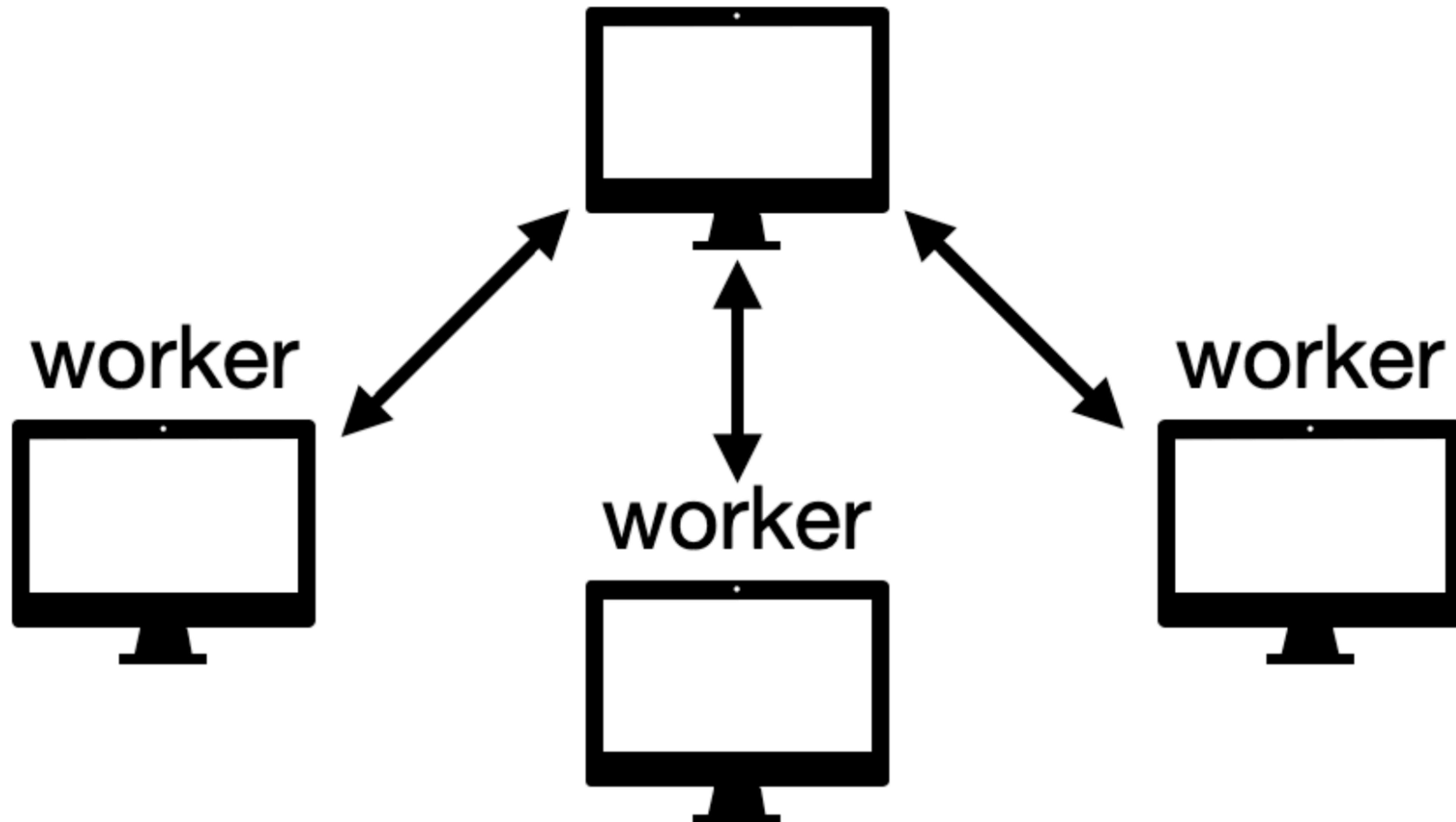


Out of Order Validation



**~62% faster block
validation**

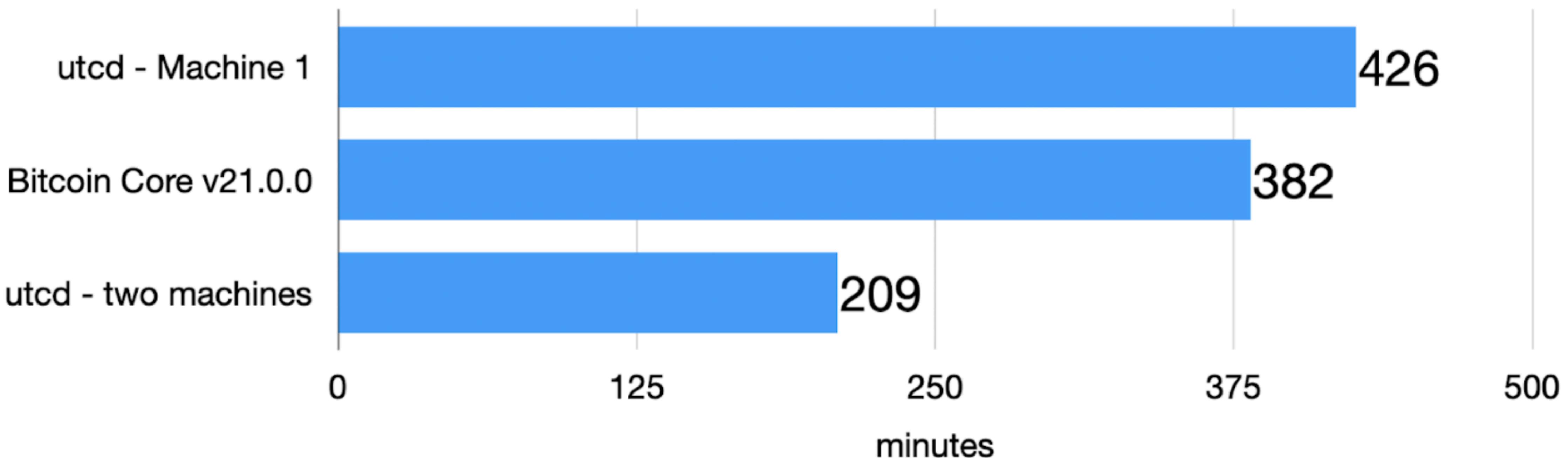
Coordinator/worker



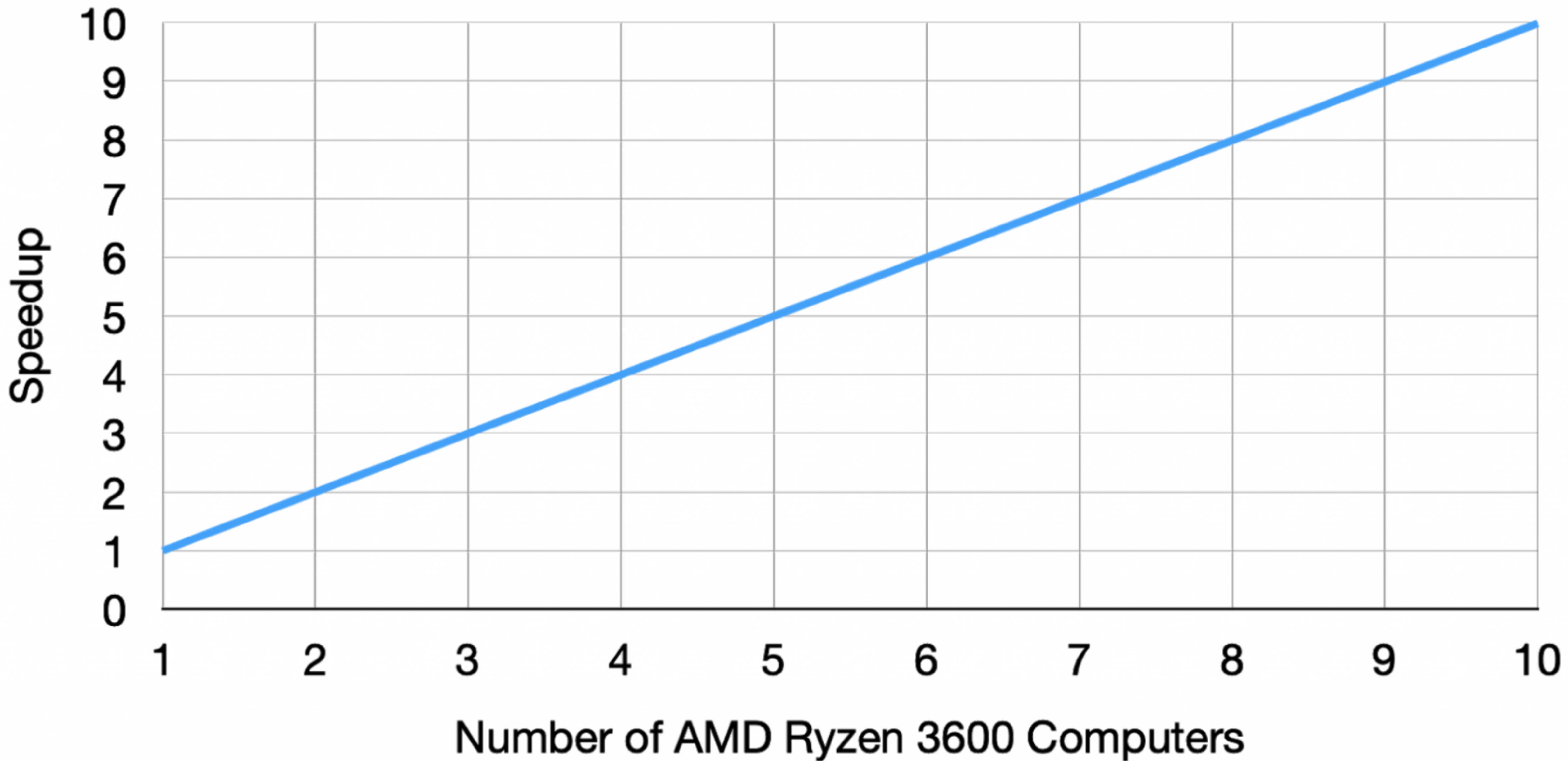
2 times faster block validation

When tested with 2 computers

Initial Block Download Speeds to block 671,000 (Full signature check)



Utreexo Parallel Node Speedups (based on Amdahl's Law)



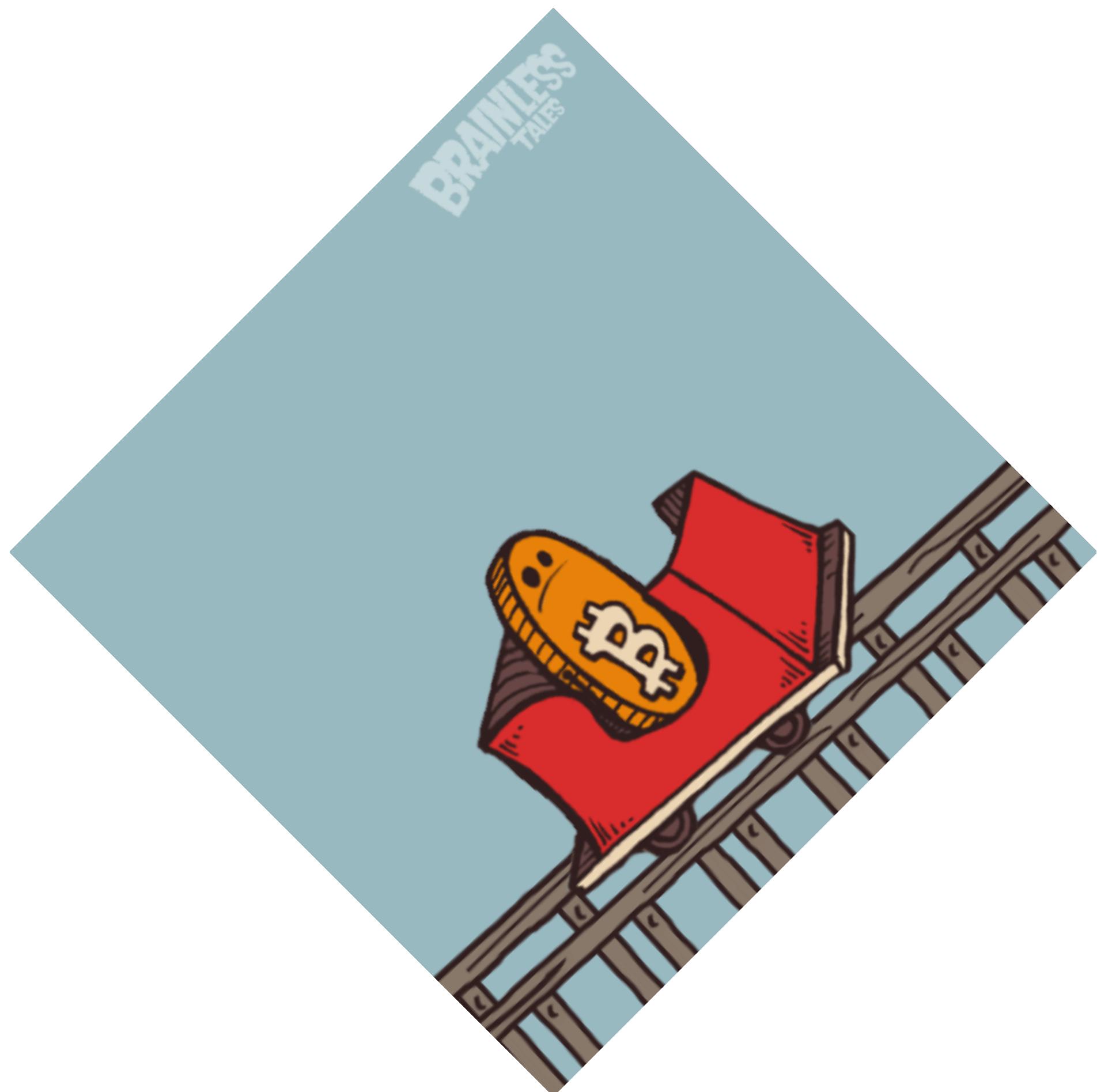
Trade offs

More internet usage

Extra data needed to be downloaded for Utreexo nodes

- Need to download 364GB Utreexo proofs (up until height 710,000)
- Working on caching to reduce the amount

Internet CPU



Disk read/write Memory Usage

