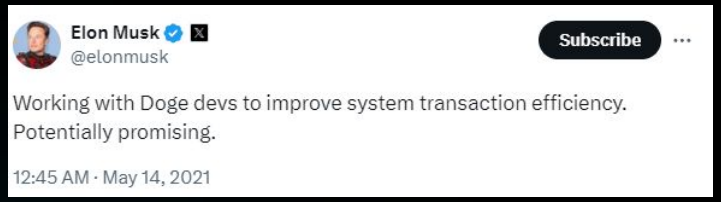
NETHERMIND

Gas limit and block execution

@M25Marek MarekM25

NETHERMIND

Gas Limit overview



Source: https://x.com/elonmusk/status/1392974251011895300

Source: https://x.com/elonmusk/status/1393738154889338884

Trade-offs

- **Disk usage:** Includes both history and state growth.
- **Networking:** Larger blocks and potentially more mempool usage.
- **Other considerations:** Syncing time, RPC availability, proof sizes for light clients, archive nodes.
- **Block execution:** How fast your EL can process blocks. This is one of the key factors impacting other hardware requirements, such as CPU, disk speed, and memory.

NETHERMIND

Who has a control over gas-limit?

The block's gas limit cannot increase/decrease by more than 1/1024 of the parent block's gas limit.

This property was utilized in [EIP-7783: Add Controlled Gas Limit Increase Strategy] by Giulio Rebuffo.

Locally produced blocks (~9%):

- Execution client teams can set their defaults.
- Every validator can override these defaults and set the desired gas limit.

Externally produced blocks(~91%):

- Consensus client teams can set their defaults.
- Every validator can override these defaults and set the desired gas limit.
- External builders are not required to follow these defaults. Ultimately, **builders** have the final control over the gas limit.

Consensus Client (externally produced blocks)

```
# Lighthouse (validator client)
--gas-limit 40000000

# Lodestar (validator client)
--defaultGasLimit 40000000

# Nimbus
--suggested-gas-limit=40000000

# Prysm (validator client)
--suggested-gas-limit 40000000

# Teku (modify teku.yaml)
validators-proposer-default-gas-limit: 40000000
```

Execution Client (locally produced blocks)

```
# Besu
--target-gas-limit 40000000

# Erigon
--miner.gaslimit 40000000

# Geth
--miner.gaslimit 40000000

# Nethermind
--Blocks.TargetBlockGasLimit 40000000

# Reth
--builder.gaslimit 40000000
```

Source: https://pumpthegas.org/

NETHERMIND

Block Execution - Mini Computer

Hardware: Intel NUC 11 Intel Core i7-1165G7, SSD Samsung 980 PRO 2TB NVMe,
Crucial SODIMM, DDR4, 32 GB
Very close to the recommended staking hardware by EthStakers.

Cost: ~715$ (475$ NUC + 240$ memory/disk). As of April 2023.

Network: Ethereum Mainnet

Avg Execution Times: around 50ms

Avg Execution Throughput: above 330MGas/s

NETHERMIND

Block Execution - High-End Consumer Machine

Mainnet Gas Target: 1.25MGas/s, **Mainnet Slot Time:** 12s

Hardware: AMD Ryzen 9 9950X, Samsung 990 PRO 4TB PCIe 4.0 x4 NVMe, Corsair Vengeance 192GB (4x48GB), MSI MPG X670E Motherboard
High-end consumer machine.

Cost: ~3000$, 280$ ~per month in cloud providers

Network: Ethereum Mainnet

Avg Execution Times: around 17ms

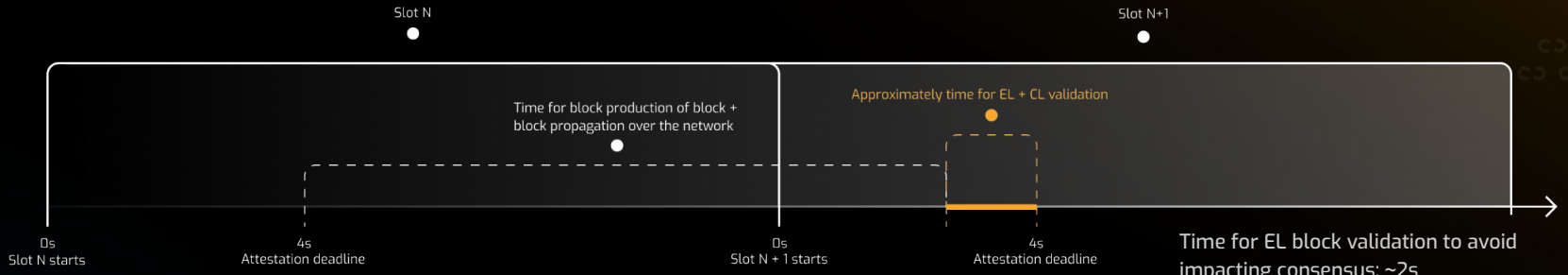
Avg Execution Throughput: around 950MGas/s

Current Mainnet Gas Target: 1.25MGas/s

Almost 3x faster than Intel NUC!

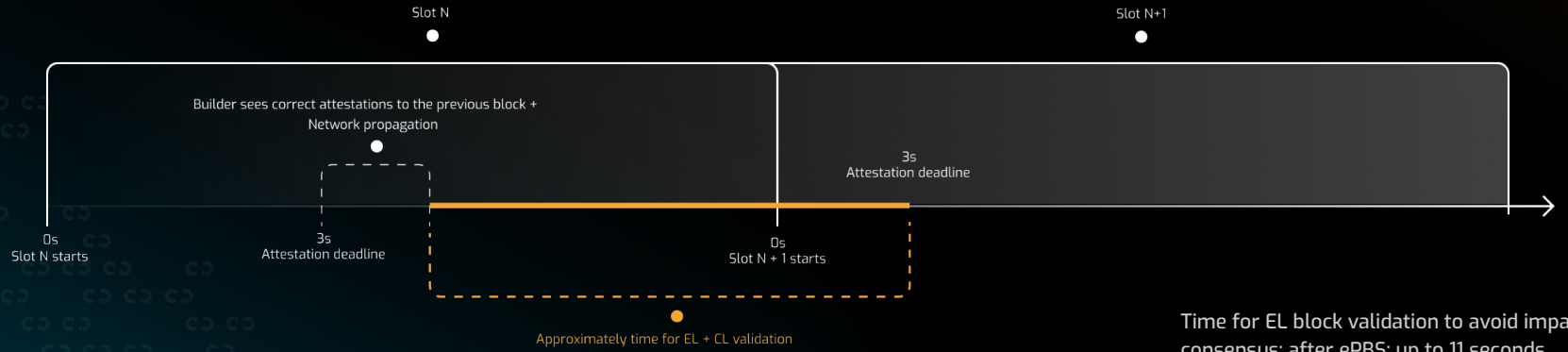
NETHERMIND

Ethereum Slot



Time for EL block validation to avoid impacting consensus: ~2s

Ethereum Slot after ePBS/APS

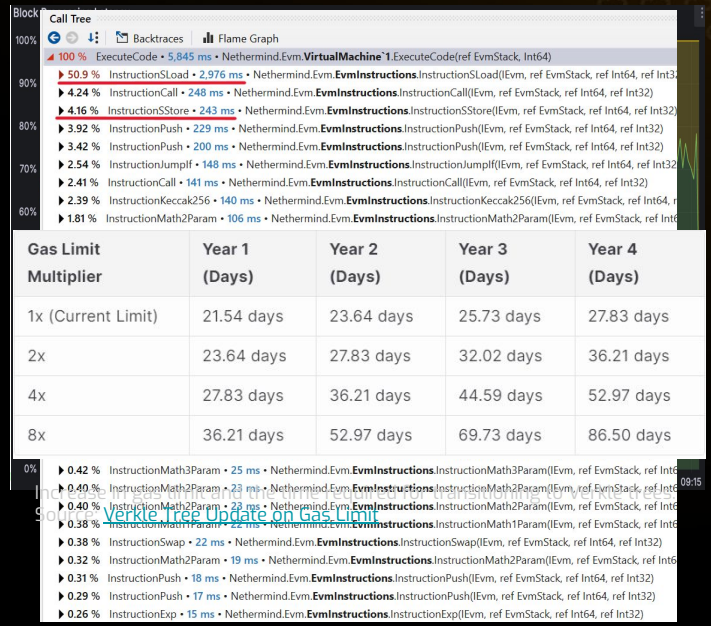


Time for EL block validation to avoid impacting consensus: after ePBS: up to 11 seconds

NETHERMIND

State access & state growth

- With the current gas limit, the state doesn't grow quickly. In the last 3 months, it grew by around 4 GB in the Nethermind database. You can read more about state growth in an excellent article from the Reth team.
- In practice, state-related operations take the most time during block execution. This is the primary area of optimization for EL clients.
- Ethereum will move to other trie in the next few years. The smaller the trie is the faster the transition will be. Data from Giulio from Erigon team shows that it doesn't have to be problematic.
- Long-term solution: state expiry, Vitalik blog post: The Purge



Gas Limit Multiplier	Year 1 (Days)	Year 2 (Days)	Year 3 (Days)	Year 4 (Days)
1x (Current Limit)	21.54 days	23.64 days	25.73 days	27.83 days
2x	23.64 days	27.83 days	32.02 days	36.21 days
4x	27.83 days	36.21 days	44.59 days	52.97 days
8x	36.21 days	52.97 days	69.73 days	86.50 days

Nethermind traces show that most of the execution time is spent on SLOAD/SSTORE operations.

NETHERMIND

EVM Gas Pricing

The factors that should be taken into account in gas pricing include:

- Time required for calculation and the resources consumed to compute a given operation.
- Contribution to state growth and history growth.
- Blocks that are heavy on hash function executions take significantly longer to ZK-SNARK prove than average blocks. It should be reflected in gas pricing and is highlighted in EIP-7667.

Gas Pricing differences	
Ripemd-160	1000 MGas/s
Simple EthTransfers	700 MGas/s
SHA-2 precompile	580 MGas/s
Keccak256	140 MGas/s
EcRecover	80 MGas/s
Point evaluation precompile	65 MGas/s

Gas pricing differences: The table shows the speed of each operation if the entire block were filled with it. Benchmarks were performed on an Intel NUC 11. For the source and more benchmarks, see: Nethermind Gas Benchmarks

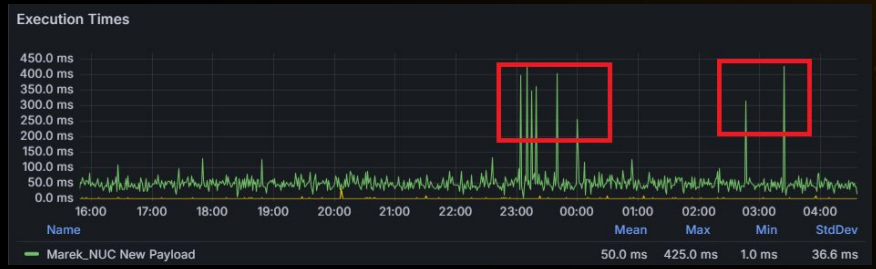
NETHERMIND

Worst-cases

Clients are continuously improving, becoming faster and faster at processing average blockchain transactions. However, we must ensure that Ethereum remains secure in worst-case scenarios.

Slow blocks can result in:

- **Chain health impact**: Blocks that take longer than around 2 seconds could cause stakers to miss attestations or proposals.
- **Liveness issues**: Blocks that take more than a few seconds to process may cause timeouts in EL clients, leading to block production failures.
- **DDoS vulnerabilities:** potentially impacting different parts of the infrastructure, such as RPC providers and relays.



Blocks are not executed in a consistent amount of time. Spikes (highlighted in red rectangles) in block execution times need to be monitored and minimized.

Solutions:

- Rigorous benchmarking and worst-case scenario analysis
- Repricing/optimizing for worst-case scenarios
- Continuous production monitoring
- Implementing EIP-7778: Prevent Block Gas Smuggling
- Implementing multidimensional gas pricing

NETHERMIND

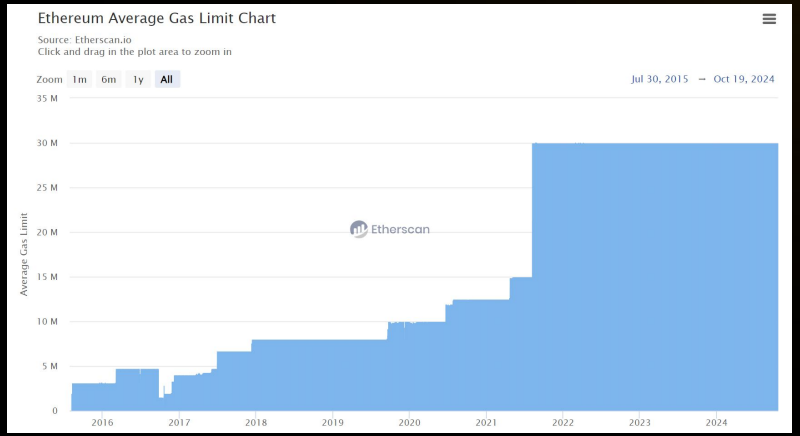
Changes in Gas Limits Through the Years

The gas limit has not been increased in a long time.
The last update to the gas target was in 2021, when it was raised from 12.5 million to 15 million gas.

The upgrade from 15 to 30 million gas was related to the introduction of the EIP-1559 base fee mechanism, but the gas target has remained at the 15 million level.

Since then:

- Hardware has improved.
- The performance of Ethereum clients has improved significantly.



Source: https://etherscan.io/chart/gaslimit

NETHERMIND

Summary

Challenge	Solution
Strict timing in Ethereum slots	Reorganization of the slot - ePBS
Migration to the new state tree & State growth	Verkle, State expiry
Gas pricing issues & worst cases	Rigorous benchmarking and worst-case scenario analysis Repricing/optimizing for worst-case scenarios Continuous production monitoring Implementing multidimensional gas pricing
History growth	EIP-4444, Portal network
Networking	EIP-7623, consolidations of validatiors, engine_getBlobs

- We need to ask ourselves what data we need and what challenges are left to solve.
- We haven't given up on L1 execution.
- Execution is not a bottleneck for scaling L1.
- There is a huge potential to scale Ethereum L1 and even bigger potential to scale with L2s.

"There is a near-unlimited number of blockchain projects aiming for the niche of "we can be super-fast, we'll think about decentralization later". I don't think Ethereum should be one of those projects. Ethereum L1 can and certainly should be a strong base layer for layer 2 projects that do take a hyper-scale approach, using Ethereum as a backbone for decentralization and security."
Vitalik Buterin

NETHERMIND

Thank you

Resources:

1. How to raise the gas limit, Part-1: State Growth by Reth Team
2. How to raise the gas limit, Part-2: History Growth by Reth Team
3. Advantage of Pipelining Consensus and Execution: Delayed Execution by Terence
4. Builder Reveal Timing Game in ePBS by Terence
5. ePBS specification notes by Potuz
6. Verkle Tree Update on Gas Limit by Giulio Rebuffo
7. EIP-7783: Add Controlled Gas Limit Increase Strategy by Giulio Rebuffo
8. Gas Benchmarks by Nethermind team
9. Timing Games: Implications and Possible Mitigations by Caspar and Mike
10. Timing.pics by Toni Wahrstätter

Special thanks to: *Gajinder, Potuz, Melannie (Nethermind), Tobi Oke (Nethermind) World Foundation for the grant that kicked off the benchmarking project.*