State growth and its relationship to Ethereum's gas limit are widely misunderstood. It is commonly believed that state growth is Ethereum's primary scaling bottleneck. However, discussions of state growth are often held back by imprecise terminology and a lack of detailed quantitative evidence.

Embracing a data-driven approach brings significant clarity to the state growth issue. In this article we leverage high-resolution datasets to understand the size and shape of state growth. In doing so, we reach the surprising conclusion that modern consumer hardware can sustain current rates of state growth for at least a decade. Furthermore, this runway is likely to be extended indefinitely by upcoming improvements to software and hardware.

We believe that Ethereum has a clear roadmap toward 1) complete elimination of state growth as a scaling bottleneck and 2) raising the gas limit to a level that supports a global-scale decentralized financial system. The goal of this blogpost series is to develop a scientific approach for understanding and enacting this scaling roadmap.

This article is part 1 in a blogpost series about Ethereum scaling. Part 1 is about state growth, part 2 is about history growth, part 3 is about state access, and part 4 is about the gas limit.

What is state growth?

The term "state growth" is commonly used as a catch-all for any Ethereum scaling bottleneck where data size exceeds the capacity of Ethereum node hardware. However, state growth should not be thought of in this monolithic way. There are multiple types of Ethereum data, each with its own relationship to a node's underlying hardware components. It is thus crucial to use precise terminology that disentangles each distinct scaling bottleneck.

State

is the set of data necessary for building and validating new Ethereum blocks. State is composed of contract bytecode, contract storage, account balances, and account nonces. History

is the set of data necessary for syncing a node from Genesis to the latest block. History is composed of blocks and transactions. State and history are non-overlapping datasets. From these definitions there are at least 3 distinct phenomena that put significant stress on a node's hardware:

1. State growth

: the accumulation of new accounts, new contract bytecode, and new contract storage.

1. History growth

: the accumulation of new blocks and new transactions.

1. State access

: the set of read and write operations used for building and validating blocks.

Each of these bottlenecks has a unique relationship to a node's hardware constraints. The four most relevant1

hardware constraints are:

1. Network IO

is the amount of upload and download speed that a node must maintain in order to reach stable consensus with peers.

1. Storage size

is the amount of data that a node must keep in permanent storage in order to build, validate, and distribute blocks.

1. Memory size

is the amount of data that a node must cache in memory in order to stay synchronized with the tip of the chain.

1. Storage IO

is the amount of read and write operations per second that a node must perform in order to stay synchronized with the tip of the chain.

The relationships between these bottlenecks and hardware constraints is illustrated in Figure 1

.

Figure 1: Ethereum Scaling Bottlenecks

Figure 1: Ethereum Scaling Bottlenecks

Starting at the top of the diagram, every time Ethereum executes a transaction, all resources used by that transaction are priced in terms of gas

. Ethereum's gas limit

is thus a single-dimensional quantity that rate-limits all forms of on chain activity2

. Downstream of the gas limit are block size

and operations per block

. The more bytes per block, the faster history will grow. The more IO operations per block, the greater the rate of state access and (usually) the greater rate of state growth.

Thus, the scaling bottlenecks are connected to the node's hardware constraints as follows3

:

- To support large amounts of state growth

, a node must have sufficient storage size and memory size. If the state grows too large, it will either not fit in storage, or the frequently-accessed portion of state will not fit in memory, degrading performance.

- To support large amounts of history growth

, a node must have sufficient network bandwidth to share large amounts of block data and enough storage capacity to store that data.

- To support large amounts of state access

, a node must have large amounts of memory to cache hot state and large amounts of storage IO to support sufficient read and write operations.

For state growth in particular, the main challenge is ensuring that state size does not grow at a faster rate than can be sustained by ongoing improvements to consumer hardware. Node memory and storage are finite resources, and so they will eventually reach capacity unless either the state stops growing or hardware is periodically upgraded. It is fortunate that memory and storage hardware have been improving for many years. Even so, the exact forecast of these improvements is not certain, and it should not be taken as a given that their rapid growth will continue indefinitely.

Note that data blobs introduced by the upcoming EIP-4844 will bring some changes to these scaling relationships. After EIP-4844, it is expected that much less history will accumulate on disk, network IO might increase significantly for transmitting large amounts of blob data.

In this article we will focus mainly on the state size and state growth rate rather than memory size and state access patterns. We will investigate these or other topics in future work.

What are the largest contributors to Ethereum state?

The next step in understanding state growth is examining the total size of state along with the relative size of each state contribution. Currently, Ethereum state takes up about 245.5 GiB on disk. These numbers were measured using a reth node, but the numbers for each node client are roughly comparable as shown in this comparison spreadsheet. Accounts, contract bytecodes, and contract storage occupy 14.1%, 4.3% and 81.7% of state respectively.

Figure 2

shows how much state is occupied by various categories of smart contract protocols. In this visualization, the size of each contract category represents the number of bytes occupied by its storage slots and bytecodes. Contract categories are hierarchical and can be navigated using mouse clicks. A category is also included to represent the total amount of state taken up by account balances and nonces.

Figure 2: Distribution of Ethereum state

(click it)

Figure 2: Distribution of Ethereum state

(click it)

The numbers in Figure 2

represent the total bytes that a node client must store on disk. This includes data used by indexes and other types of storage overhead. Numbers were taken from a reth node, but values are roughly comparable across clients. The average size to store each account and each storage slot were 133.6 bytes and 191.3 bytes respectively.

There are many interesting patterns within Figure 2

, but here are some of the most important takeaways:

- Tokens are the biggest contributor to state.

The largest contributors to Ethereum state are ERC-20 and ERC-721 tokens, occupying 27.2% and 21.6% of state respectively. The reason why tokens take up so much state is that each user balance of each token must be separately stored in its own 32 byte storage slot. Thus, this half of Ethereum's state scales with the total number of Ethereum users and the total number of tokens held by each user.

- At least 7.4% of Ethereum's state is dormant.

Some of the largest contracts in Ethereum's state are no longer actively used. These protocols are from a time when blockspace and state space was much cheaper than it is today. This includes most of the protocols in the Game, Gambling, and Scam/Scheme categories. There are also many DEX's that are no longer actively used, including IDEX, Etherdelta, and Oasis. Collectively these protocols compose at least 7.4% of Ethereum state. The true level of dormancy is higher, as it would also include many entries from the longtail of ERC-20, ERC-721, and Other categories.

- L2 bridges occupy less than 2% of Ethereum state.

By utilizing techniques like compression, ZK proofs, and improved encodings, L2 transactions make much more efficient use of state than mainnet. L2's collectively achieve ~5x more transactions per second than mainnet, despite occupying only 2% of mainnet state (and 10% of state growth, see next section).

How fast is Ethereum state growing?

The most important aspect of state growth is the evolution of state growth rates over time. These rates reveal the severity of the problem and whether the severity is trending upward or downward.

Figure 3

shows state growth rates since Ethereum's inception in 2015. These rates are computed by summing the contract bytecodes and contract storage within each contract category. Subsets of these categories can be visualized by clicking and double clicking items in the figure legend.

Figure 3: Ethereum state growth over time

(double click the legend)

Figure 3: Ethereum state growth over time

(double click the legend)

There are many interesting patterns within Figure 3

, but here are some of the most important takeaways:

- State currently grows around 2.62 GiB per month, down from a peak of 5.99 GiB per month.

By linear extrapolation, these numbers project the total state size to be between 396 GiB and 606 GiB in 5 years. Although one might describe the current growth rate as 12.8% per year, the absolute growth rate has been declining while the state

continues to grow, so simple exponential growth may not be an appropriate model.

- Recent declines in state growth are mainly due to reduction in NFT activity.

(Hint: double click the “erc721” label on the figure legend to view this data in isolation). Although one might expect some degree of correlation between different types of network activity, there is a surprising amount of independence between individual state contributors. For example, the ERC-20 state growth rate has actually been increasing each year since 2020, despite the total state growth rate declining over the past couple years.

- State growth is the lowest it’s been since 2021.

This decline is rather surprising, but it makes sense given state is mostly proportional to the formation of new token balances. If state growth rates have been declining, some might suggest that this indicates that Ethereum is capable of supporting a higher gas limit. This may be true, but it is important to remember that 1) there’s nothing preventing a new surge in growth rate under the current gas pricing model, and 2) state is not the only type of bottleneck downstream of the gas limit.

How much state growth is acceptable?

We now know the Ethereum state’s 1) size, 2) composition, and 3) growth rate. How do we determine the range of acceptable state growth values? This question is complicated because it depends on both unpredictable market forces and philosophical choices about which tradeoffs Ethereum should make.

Let’s start with the simplest possible model of how long the current levels of state growth are sustainable on common consumer hardware, assuming no future hardware improvements. As shown in Figure 3

, in recent years the state has been growing at an annualized rate somewhere between 31GiB/year to 72GiB/year. Common consumer hardware currently tops out around 4TiB of storage and around 64GiB of memory. From this we can create a simple forecast of storage and memory requirements:

- Storage

: Nodes must currently store a total of around 1TiB (due to storing both state and history). In practice this means that many nodes are using disks of size at least 2TiB. For simplicity, let us ignore future history growth, as if we were in a post-EIP-4444 world. We can compute the amount of runway as runway = (remaining storage capacity) / (state growth rate), as shown in more detail in this spreadsheet. Thus, node storage hardware can support current rates of state growth for well over a decade without exhausting 2TiB of space. A 4TiB drive would suffice for almost half a century at current levels of state growth.

- Memory

: Ethereum-on-arm users report that the minimum viable memory size for running an Ethereum node is currently around 16GiB. If we assume that memory requirements grow proportionately with state size (and this is a big assumption), then the 30GiB/year to 72GiB/year state growth rates translate into 2GiB - 4.7GiB additional memory needed per year. Thus, at current gas rates 32GiB RAM should be sufficient for anywhere from 3 years to 8 years. 64GiB of RAM should be sufficient for 10 years to 23 years.

This is a simplified model with many assumptions. Possible extensions to this model include 1) history growth, 2) nonlinear scaling of memory requirements, 3) decreasing hardware costs, 4) increases to gas limit, 5) opcode gas repricing, and 6) future Ethereum architecture improvements. Each of these factors can interact nonlinearly and evolve over time. We will explore these model extensions in future work.

It must be emphasized that long-term sustainability is a good thing. Even if modern hardware can support many years of runway, shortening this runway should never be taken lightly. Any plan that accelerates state growth should include a significant buffer for unforeseen changes to the hardware or software landscape.

How can state growth be solved?

Many different solutions have been proposed for addressing state growth. Three improvements to Ethereum architecture stand out: rollups

, Verkle tries

, and state expiry

. Taken together, these form a comprehensive roadmap for solving state growth in the short, medium, and long term.

Short term

: Rollups do not solve state growth, but they do ease the burden.

As shown in Figure 2

and Figure 3

, rollups are able to use state more efficiently than mainnet. Offloading activity to L2's does require some amount of state to be stored on mainnet in order to support user exits. However, the state footprint of L2 transactions is much lower than the footprint of transactions on mainnet. Rollups thus make it more sustainable to increase total activity in the ecosystem. The adoption of rollups is expected to grow with the upcoming EIP-4844, which will make rollups much cheaper through use of blobs.

Medium term

: Verkle tries solve state growth for validator nodes, but not for nodes that need to build new transactions

: Verkle tries are a new data structure for Ethereum state. They enable more efficient light clients and "stateless" nodes. These nodes will be able to validate new blocks without any knowledge of existing state values. This eliminates the state growth problem for validator nodes. The building of new transactions will still require storing and accessing state, but this would still be a more sustainable situation than we have today, because transaction construction is a task that can be easily distributed across many machines. In terms of scope, Verkle tries represent a significant engineering effort that could take years to implement.

Long term:

State expiry solves state growth for all nodes, but requires additional infrastructure

. State expiry allows nodes to discard inactive portions of the state, such as the dormant state shown in Figure 2

. Note that the term "state hibernation" might be a more appropriate name, as most existing proposals allow for recovering "expired" state via proofs. With regard to concerns around expired state being lost over time, as long as the history (block and transaction data) is available then the state can be reconstructed. Thus, whatever solution is developed for the history preservation problem of EIP-4444 will also solve the state preservation problem. It is possible that state expiry might be unnecessary in a world where Verkle Tries succeed at their goals.

These are not the only solutions proposed to address state growth. Others include state rent and sharding, but historically these have had concerns around UX or soundness. A combination of these solutions and others may be necessary for reaching an endgame solution in the more distant future.

Closing

Although state growth is a key challenge for scaling Ethereum, we believe it is a solvable problem using known technical solutions. By our reading of the data, Ethereum can sustain current levels of state growth for many years, with a comfortable buffer for experimenting with architectural upgrades.

We believe that empirical methods will be essential for engineering Ethereum's gas limit and steering Ethereum toward endgame scaling solutions. This article is only a single step toward that goal. There are other types of data beyond state, each imposing their own scaling burdens on an Ethereum node and on the Ethereum gas limit. We hope to explore these other bottlenecks in future work.

If you are excited about research in Ethereum scaling, reach out to storm@paradigm.xyz and georgios@paradigm.xyz. We'd love to hear about how you are thinking about the problem and potentially collaborate. The data and code used for this article can be found on Github here.

Further Reading

- Ethereum Data Structures
- State size management
- Statelessness and state expiry roadmap
- Why it's so important to go stateless
- Verkle Trie EIP
- Ethereum In Numbers
- The Limits to Blockchain Scalability

- Verkle Trees for Statelessness with Guillaume Ballet

Acknowledgments

Thank you to Tim Beiko, Péter Szilágyi, Guillaume Ballet, Banteg, Alex Stokes, Jesse Pollak, Toni Wahrstaetter, Patrick O'Grady, lightclient, Ansgar Dietrichs, Frankie, Dan Robinson, Matt Huang, Doug Feagin, and Arjun Balaji for review and feedback.

Thank you to Achal Srinivasan for the Figure 1

graphics.

1

There are additional hardware bottlenecks such as CPU. Here we focus on the four bottlenecks most relevant to Ethereum mainnet, but other hardware bottlenecks might become relevant in other contexts.

2

EIP-4844 will introduce a separate blob gas limit dimension for controlling the rate of Ethereum blob activity.

3

Note that Figure 1

only shows the primary relationships between these bottlenecks. There are secondary bottlenecks that are sometimes relevant. For example, when performing a snap sync rather than a full sync, there is a strong relationship between snap sync time, state growth, and network I/O.