

A Comprehensive Study of Dencun Upgrade

All Things for Rollup and Staking

[Andy Yang](#)

Abstract

Dencun Upgrade, also known as the Cancun-Deneb Upgrade, continues the naming tradition that has been the hallmark of Ethereum hard forks: the execution layer uses the name of the place where Devcon was previously held (Cancun), and the consensus layer uses the name of the star (Deneb) in alphabetical order. The upgrade is scheduled to go live on 13 March 2024 and contains a set of Ethereum Improvement Proposals (EIPs). In this study, we describe the design of EIP-4844, the landscape of Rollups Post-Dencun, and its effects on Staking and Validators.

TL;DR

1. EIP-4844 creates a specialised storage space called "blob" for Rollup, with cheaper gas pricing than the original calldata; the rest of the transaction structure is inherited from EIP-1559.
2. Blob is parallel to blocks, like a "sidecar". Only reference to blobs are stored in the block, it will not be executed by EVM.
3. Blob only lasts ~18 days. Validators do not need to worry about the additional storage for blobs, since they only need to store about 99.53 GB, even in the worst case.
4. We estimate that the Dencun upgrade will reduce Rollup's fees by less than 10x, depending on Rollup's tuning of the parameters.
5. EIP-7044 lets the pre-signed exit validator message valid permanently, which leads to a more secure Staking-as-a-Service ecosystem.
6. EIP-7045 relaxed the requirements for target vote, validators may get more rewards.
7. EIP-7514 set a limit to activated validator per epoch, validator activation time will be longer, this ensures ETH staking rate will not increase to 50% at the end of 2024 and a profitable APR for validators.
8. EIP-4788 introduced "native oracle," liquid staking pools and restaking protocols are much safer now.

Rollup-Centric EIP-4844: World of Blob

Why do we need EIP-4844?

Name	Send ETH	Swap tokens
↔ zkSync Era	\$0.07	-
↳ Loopring	\$0.08	\$0.81
↔ zkSync Life	\$0.11	\$0.26
OP Optimism	\$0.15	\$0.30
B Boba Network	\$0.21	\$0.24
Arbitrum One	\$0.24	\$0.67
Polygon zkEVM	\$0.26	\$0.96
DeGate	\$0.28	\$0.26
StarkNet	\$0.36	\$1.14
ETH Ethereum	\$2.04	\$10.18

Source: l2fees.info

Based on the data in the chart above, the current Rollup transaction cost is about 10x ~ 30x lower than L1, but still not enough to support the high-frequency transaction scenarios such as derivatives, games, etc. Rollup's costs come mainly from posting data from L2 to L1. The top five gas users on Ethereum are all Rollups.

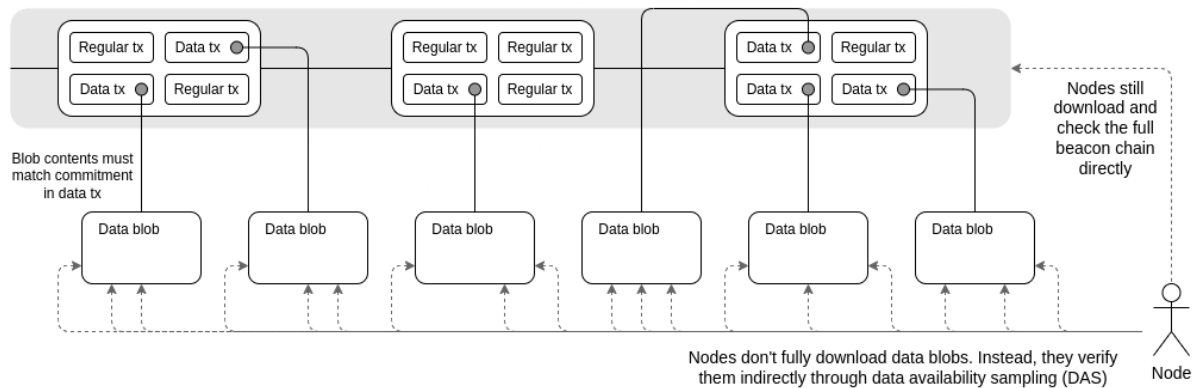
Top 50 Gas Spenders (Sending Accounts that pay a lot of Gas)					
Rank	Address	⚡ Fees Last 3hrs	⚡ % Spent 3hrs	⚡ Fees Last 24hrs	⚡ % Spent 24hrs
1	Arbitrum: Batch Submitter	\$95,649.16 (25.96 Eth)	2.78%	\$794,453.22 (215.59 Eth)	2.36%
2	Scroll: Batch Committer	\$62,651.37 (17.00 Eth)	1.82%	\$260,580.65 (70.71 Eth)	0.81%
3	Optimism: Batch	\$50,641.74 (13.74 Eth)	1.39%	\$683,551.70 (185.50 Eth)	1.93%
4	zkSync Era: Validator	\$40,833.16 (11.08 Eth)	1.09%	\$345,646.04 (93.80 Eth)	0.97%
5	Base: Batch Sender	\$37,213.63 (10.10 Eth)	1.04%	\$338,613.35 (91.89 Eth)	0.97%

Source: [Etherscan](https://etherscan.io)

To reduce Rollup's costs and empower Ethereum's rollup-centric blueprint to flourish, EIP-4844 creates an economically cheaper, dedicated storage space for Rollup: blob, an independent gas pricing market for blobs, and a Type 3 transaction: "blob-carrying" transaction.

Blob: A New Storage Space

Storage Structure

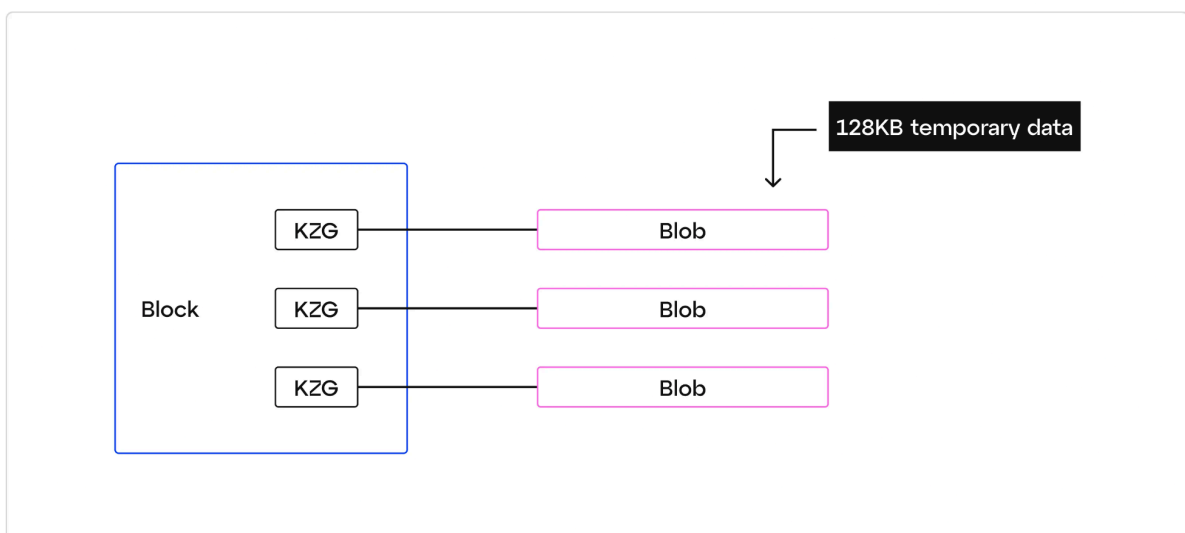


A blob is a container with 4096 fields, each storing 32 bytes. Transactions that use blobs are called "blob-carrying" transactions because blob data is not stored in the block. Only a reference to the blob is stored. This is why blobs can save gas: Since blobs are stored as references, EVM will not execute them, thus consuming no additional computational resources.

With the introduction of the Blob, two fields were added to the block header:

blob_gas_used : Total amount of blob gas consumed by the transactions within the block.

excess_blob_gas : A running total of blob gas consumed in excess of the target, prior to the block. Blocks with above-target blob gas consumption increase this value, and blocks with below-target blob gas consumption decrease it (bounded at 0).



Source: [Consensys](https://consensys.com)

The blob structure is like the idea used in traditional blockchains, where each block stores a hash of the previous block, and the hashes join the blocks together to form a chain. Similarly, storing a reference to a blob in each block makes blobs and blocks parallel. It's like a "sidecar": blocks are connected to blobs.

Blob-carrying Transaction

Blob-carrying Transaction, aka Type 3 transaction, introduces two new fields:

max_fee_per_blob_gas : Maximum fee a user is willing to pay per blob gas.

blob_versioned_hashes : A reference to a list of blobs.

A Blob-carrying Transaction can carry one or two blobs. Like other transactions, Blob-carrying transactions (Type 3 transactions) are stored in the mempool, waiting for the validator to pack. However, Type 3 transactions are not executed by the EVM but propagated in the consensus layer as sidecars via the gossip protocol.

Before 4844, data uploaded from L2 to L1 could only be mounted on calldata. Calldata is very expensive, 1 non-empty byte requires 16 gas, and an empty byte requires 4 gas. For blob-carrying Transactions, calldata portion is replaced by the purchase of the whole blob. Other operations require EVM pay the gas fee as usual.

Why blob, not less calldata consuming gas?

There have been two options to reduce the gas consumed for Rollup:

1. Reduce the gas used by calldata directly.
2. Full Sharding: Dividing the complete blockchain into many parallel mini-blockchains, thus greatly increasing the throughput.

If we want to reduce the gas fee by 10 times, adopting scheme 1 will cause the theoretical maximum block size to ~18MB, which is unbearable for the Ethereum network. Option 2 is still not fully technically achievable.

EIP-4844 is a compromise solution that is compatible with full sharding. At the same time (Rollup does not need to drastically change the data format when implementing a full sharding solution) reducing Rollup's gas costs in the medium to long term.

For this reason, it is also known as Proto-Danksharding (Protolambda and Dankrad Feist), which is a small step forward for sharding and a big step forward for Ethereum.

How long should a Blob be stored?

The original maximum size of a block was 30M gas / 16 gas per calldata byte = 1.875MB.

The current target number of blobs per block is 3 and the maximum number of blobs is 6.

The size of each blob is approximately 0.128 MB.

Target Case: All blocks store 3 blobs

Currently there is one block per slot (12s), 32 slots per epoch, and 82,181.25 epochs per year.

Ideally, we need to store $82181.25 * 32 * 0.384 \approx 1.01$ TB per year, and if we adjust the Target value to 6, we need to store 3.03 TB per year.

Worst case: All blocks store 6 blobs

If we max out the blobs in each block, we need to store an additional $1.01 * 2 = 2.02$ TB per year. if we adjust the Target value to 6, we need to store an additional 4.04 TB per year.

This is clearly node-unfriendly; choosing to store the blob continuously would make the requirements on the nodes higher, leading to less decentralization. Therefore, EIP-4844 employs a pruning function that clears the blob after 4096 epochs (~18d). Even for an Optimistic Rollup with a challenge period (avg. 7d), 18 days is sufficient to complete the challenge.

Dual Fee Markets: Blob and Block

Recalling EIP-1559 Gas pricing

Let's start with a brief review of the EIP-1559 fee structure:

When you initiate a transaction in your wallet, you must pay a certain gas fee. The gas fee is the cost necessary to use Ethereum computing resources. It is just like paying for using other public services in real life (e.g., Paying to take the underground). The difference is that Ethereum runs 24/7, and if the gas fee were not required, a malicious attacker could misuse the resources using infinite loops, resulting in a DoS attack on Ethereum.

To calculate the gas fee, we use **gas fee = gas unit * gasPrice**.

Different operations (send ETH, swap, etc.) need different gas units. It's like building a rocket and screws can't have the same price, since EVMs consume more compute resources to handle more complex operations, they require a larger amount of gas.

The pricing of each unit of gas is divided into:

- base fee: the basic fee that users need to pay, which is determined by the total amount of gas used in the previous block.
- priority fee: a small fee that users pay to process their transactions as quickly as possible.

Thus, we get how to calculate the gas fee: **gas fee = gas unit * (base fee + priority fee)**

Blob Gas: EIP-1559 Derived Independent Pricing

EIP-4844 introduces a new Gas fee model for blobs. Previously, the gas base fee defined in 1559 was determined only by the total amount of gas used in the previous block:

- If the previous block's gas usage is less than the target value, the base fee is reduced by some value less than 12.5%.
- If the amount of gas used in the previous block is the target value (15M), the base fee for the next block remains unchanged.
- If the previous block's gas usage is greater than the target value but less than the maximum value, the base fee is increased by some value less than 12.5%.
- If the gas usage of the previous block reaches the maximum (30M), then the base fee of the next block will be increased by 12.5%.

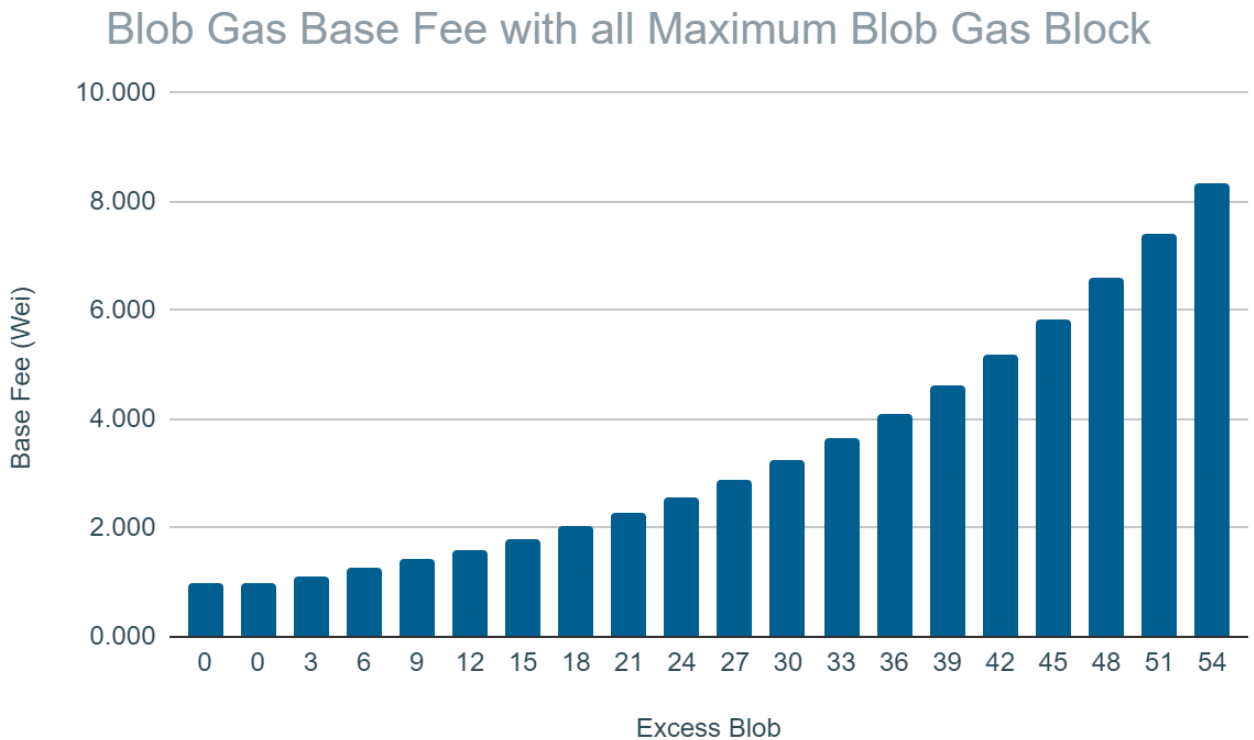
As for blob gas, the blob base fee increases exponentially as the excess blob increases, which reduces usage and eventually forces the excess blob to decrease. Unlike gas pricing, the base fee for blob gas starts at 1 wei and is determined by the number of blobs that have cumulatively exceeded the target value (6 blobs) up to the last block. Block space only considers short-term impacts, while blob considers more medium-term impacts. This may sound abstract, so let's take an example.

Block Number	Included Blob Gas	Fee Increase	Blob per Block	Excess Blob	Excess Blob Gas (Wei)	Current Base Fee (Wei)
1	393,216	0.0%	3	0	0	1.000
2	786,432	0.0%	6	0	0	1.000
3	786,432	12.5%	6	3	393,216	1.125
4	786,432	12.5%	6	6	786,432	1.266
5	786,432	12.5%	6	9	1,179,648	1.424
6	786,432	12.5%	6	12	1,572,864	1.602
7	786,432	12.5%	6	15	1,966,080	1.802

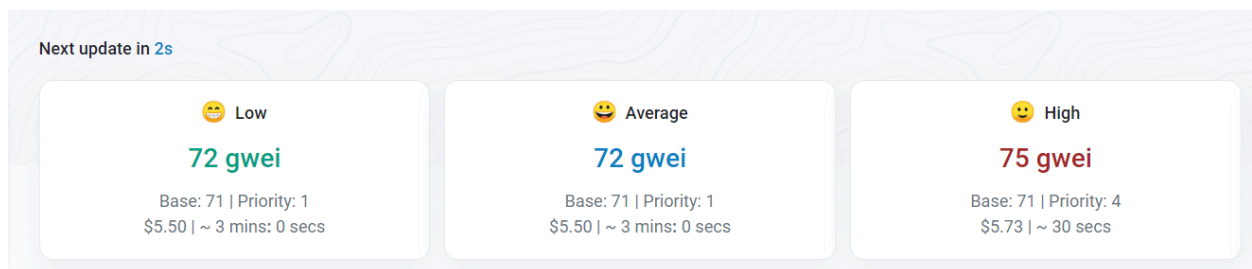
All the blobs in all the blocks in the table above are filled all the time so that when there are 6 blobs in the block, the base fee for Blob Gas in the next block grows by 12.5%.

From [our model](#), we observe that the base fee increases by ~6 times for every 50 excess blobs, and by ~49 times for every 100 excess blobs. This indicates that if the block is continuously filled with blob gas, the base fee will increase exponentially.

How much will Rollup's fees be reduced?



The gas fee reaches the level of the bull market in Ethereum mainnet (~120 Gwei) when the excess blob is 600~650. To quantify the blob gas fee in this interval, we created the above bar chart.



Source: [Etherscan](https://etherscan.io)

When the excess blob reaches 650, the base fee reaches the current peak of [Blobscan's](https://blobscan.io) average daily gas price: 121 Gwei. In this case, if the priority fee = 1 Gwei, a blob would use $122 \text{ Gwei} * 2^{17} \approx 0.016 \text{ ETH}$ of gas. However, the use of ETH is still much lower than the current cost of using calldata. As shown in the figure below, an Arbitrum transaction using calldata with normal gas fee on the mainnet (gas price between 60 Gwei ~ 70 Gwei) costs about 0.12 ETH, with ~113.5 KB of data. A blob can hold 128 KB costs 1/7th of that at ~120 Gwei.

* We did not use the same base fee for the calculation here, as blob gas for EIP-4884 and gas for 1559 are independent markets so the base fee may vary. This is only a rough comparison.

Transaction Action: Call Add Sequencer L2Batch From Origin Function by Arbitrum: Batch Submitter on Arbitrum: Sequencer Inbox

Sponsored:

From: 0xC1b634853Cb333D3aD8663715b08f41A3Aec47cc (Arbitrum: Batch Submitter)

To: 0x1c479675ad559DC151F6Ec7ed3FbF8ceE79582B6 (Arbitrum: Sequencer Inbox)
 Transfer 0.119215652069908626 ETH From 0xe64a54E2...80E2E4eb5 To Arbitrum: Batch Submitter

Value: 0 ETH (\$0.00)

Transaction Fee: 0.119227272785735981 ETH (\$439.35)

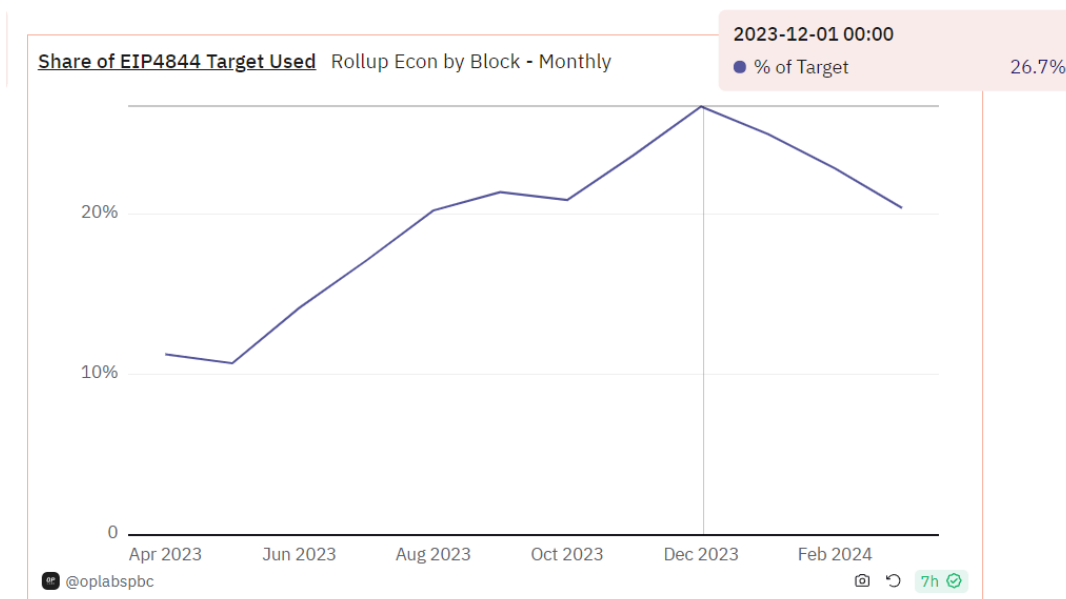
Gas Price: 65.629279626 Gwei (0.000000065629279626 ETH)

Source: [Etherscan](#)

With the significant reduction in L2 costs, we expect a less than 10x reduction in gas fees. However, since **L2 transaction costs = L2 transaction fees + the cost of posting L2 transaction batches into L1**, the exact amount of reduction will depend on L2's adjustments to its fee structure. We estimate that it probably be less than 10x lower.

Is Blob Enough For Rollup?

According to Dune's data from April 2023 ~ February 2024, all Rollups collectively only use equivalent to 26.7% of the target blob size, which means that 1 blob (~33.3%) is already enough for all Rollups. (Of course, it would be another story if Rollups adjusted the interval of posting L2 transactions to L1)





Source: [Dune](#)

Viewing Blob Information In Blobscan

Here, we will briefly describe how to view the information on blobscan:

Block

Block Details	
Overview	
Block Height	10645246
Hash	0x0c6c2a633f58987214deb9f13312be908c822f818b64740bb16f58709b321191
Timestamp	8 hours ago (Mar 7, 2024 9:57 AM+08:00)
Slot	7772389 
Blob Size	Total Included Blobs Size 768 KiB (6 blobs)
Blob Gas Price	Blob Base Fee + Priority Fee 0.000000019893400088 ether (19.893400088 Gwei)
Blob Gas Used	% of blobs used 786,432 (100%)  +100.00% Blob Gas Target
Blob Gas Limit	Max blobs per block 786,432 (6 blobs per block)
Blob As Calldata Gas	5,368,208 (6.83 times more expensive)

In the above figure:

- **Blob Size** indicates the total size of the blobs in the block
- **Blob Gas Price** indicates the price of Blob Gas
- **Blob Gas Used** indicates the percentage of blobs used compared to the Target value (3 blobs)
- **Blob Gas Limit** indicates the maximum number of blobs (6 blobs) in a block
- **Blob As Calldata Gas** indicates how much gas is consumed by using calldata for the same data.

This example transaction consists of 6 blobs, about 768 KB, and the price of Blob Gas is 19.89 Gwei, which is more than twice the target value of blobs used. Uploading data using calldata consumes 6.83 times more than using a blob.

Blob-Carrying Transaction

Transaction Details

Overview

Hash	0xf43494a3c557126c818b5ecf67d7ab6d1c400ad16c843a3a273bd7bdcf4b7b0a
Block	10645246
Timestamp	8 hours ago (Mar 7, 2024 9:57 AM+08:00)
From	0xa83c816d4f9b2783761a22ba6fad0eb0606d7b2
To	0x11e9ca82a3a762b4b5bd264d4173a242e7a77064
Total Blob Size	256 KiB
Blob Fee	Base: 0.005214935472668672 ether (5,214,935.472668672 Gwei) Max: 0.01572864 ether (15,728,640 Gwei)
Blob Gas Price	0.000000019893400088 ether (19.893400088 Gwei)
Blob Gas Used	262,144
Blob As Calldata Gas	1,797,020 (6.86 times more expensive)

Since a blob is 128 KB, the transaction carries two blobs. Each blob consumes a fixed amount of gas of 2^{17} , so two blobs consume 262,144 gas. Where $\text{Blob Fee (Base)} = \text{Blob Gas Price} * \text{Blob Gas Used}$. $\text{blob Fee (Max)} = \text{Max Blob Fee (Max)} = \text{Max Blob Gas Price} * \text{Blob Gas Used}$.

Blob

Blob Details

Overview

Versioned Hash	0x01c8447587a5792b7c6fa36b44381263e9c306a29a99aa72108a9b97cb73abcd
Commitment	0xa981bcbab880c745102abb5cc7229f5b3eac25005942a5bff95b65975677afc2d81c7a6b8872065be210f52b8ed2499
Proof	0xaf3ccb65b4d34c9d52096d0571bce6c1f4dac7a67a914a40348dc56756f4e36edac3e3c8cd9e13fdcebf254cff128a7
Size	128 KiB

Data

View as: Original

```
0x001611aa00000000418ff00ff0001feedd4cde6b7f53e99f3998ca9fe74118700630213dfa7a6b269e193c42c3b8e13647c007b80234cfd948bf37d0e4b9487006fdbca40c45899b75198659ac1b0178fd6393e0ac8a65368c00bf49cf9289f00e08daefaf88e09564b26c2ce5cc22cfd8636d46e1810ee30d5cbf3cfb5c4d600e2e31faa370f9dd4263ef192ba64505fd2a1fbf5b7fcd13741cc826c2dfb200f4e3f33abd4a46647dcc22e346607007f3efca1e9db51274e1bc6e4fba6c900e0e6e9c42e51eef2a1fe1face1d03b340d77e70fa4279ecae20fb0627ae0bc00aef9d3caed163ac18a373e3ae0f753e2d687af783d2548e2bcc28a510e1a1002fa41ea53e398a95243310944bde248d3ad6057b6d170408f8d874f4c2641c0060e686421bdc5e03c7677eeb6ce42f6a08e8178f5871203006e23b8e540896007886da07e0929ff531764ea0ecd387cbccdd8dcc55322df13d4c66f6b952d10084a74682d1574acb57493c6d3492dba4bf66130e8472218ace8a97c245db30059ec44ae18d6f37dc8ea5bcb
```

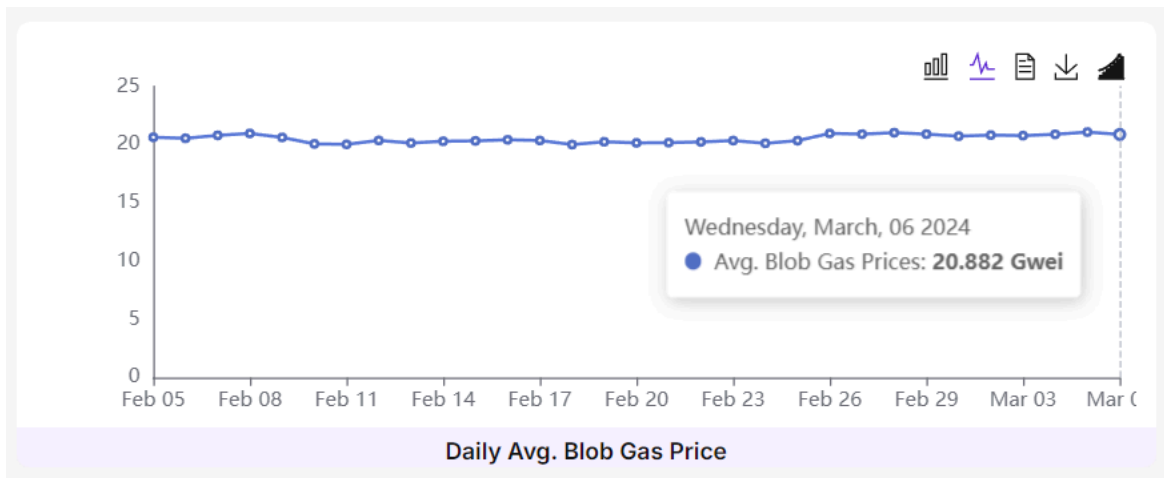
- **Versioned Hash** here is a reference to blobs, consisting of a 0x01 byte representing the version and the last 31 bytes of the SHA256 hash of the KZG proof. You may wonder why the KZG proof is not stored. This is because EVM handles data in 32 bytes, which is easy to migrate if the cryptographic primitives are changed later for

other purposes (e.g. post-quantum security). The process is as follows: **Blob** → **KZG Commitment** → **Versioned Hash**.

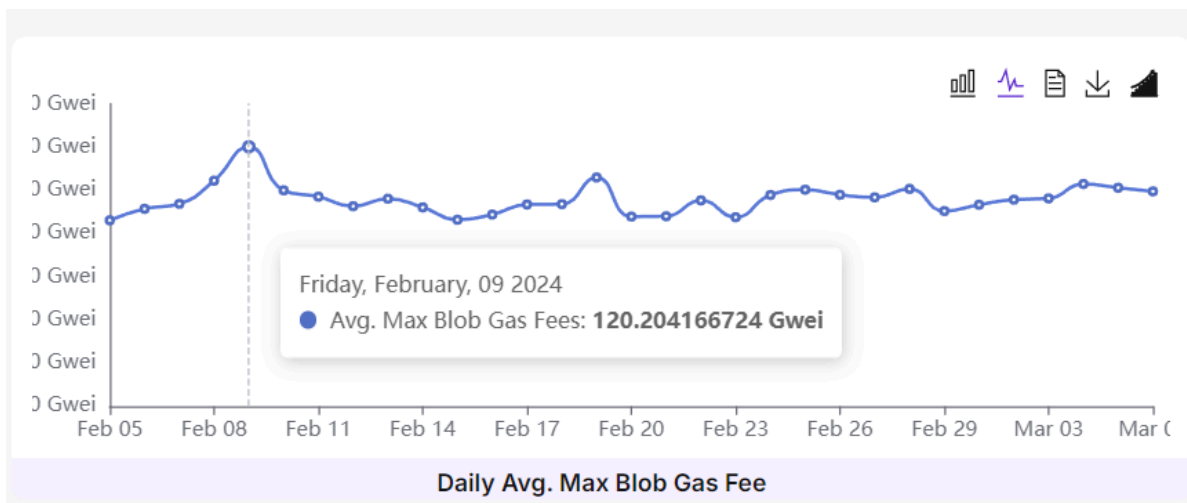
- **Proof** aims to prove that each KZG Commitment corresponds to a blob.
- **Size** represents the size of a blob.
- **Data** below the figure is the raw data of the transaction posted by Rollup.

Goerli Testnet Blob Data Analytics

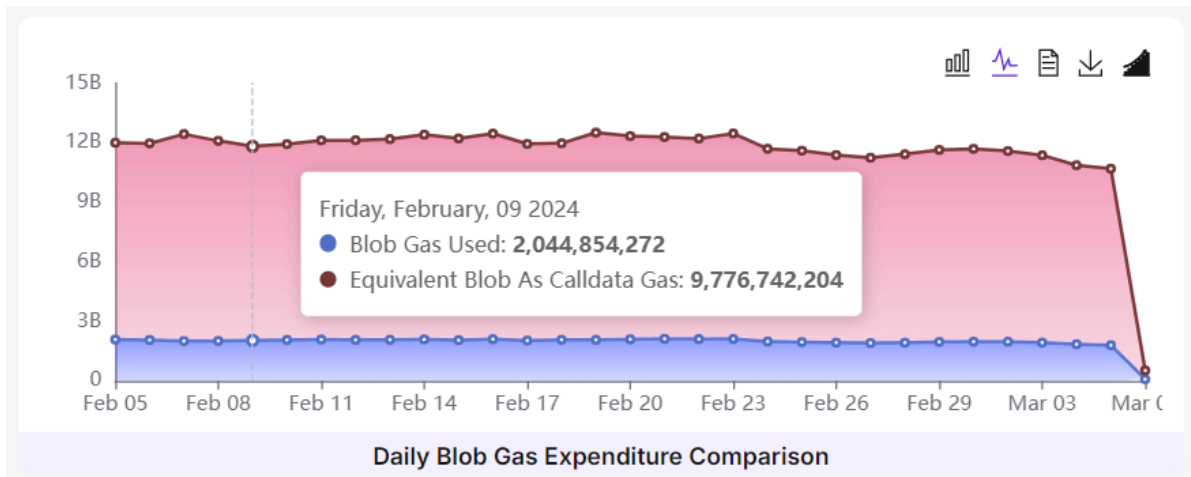
In this section we have analysed data from the Goerli testnet:



According to Blobscan data from 7 March, the Goerli test network, which will be deprecated after the Dencun upgrade, is currently averaging around 20 Gwei for Blob Gas, and has cumulatively exceeded its target (three blobs) by around 650 blobs.



Blob gas is currently averaging a daily high of 120.2 Gwei, cumulatively exceeding Target (3 blobs) by about 650 blobs.



We observe that the average amount of gas consumed using calldata is 3-6 times higher than using a blob.

How Does Rollup Use Blob Post-Dencun?

It is worth noting that if you choose to use a blob to upload data from Rollup to L1, you must purchase the full blob. This means that even if the uploaded data is less than 1 blob, you must use the full 1 blob. This naturally gives rise to two strategies:

1. Wait until the data can fill a blob and then buy the blob.
2. Choose a frequency to release the blob quickly.

For Rollup users, they want to **confirm their transactions as soon as possible** while maintaining **acceptable costs**. For Rollup, this creates a **cost and latency tradeoff**: Is it better to wait until the data has accumulated to the point where a blob can be filled before purchasing it (**low cost**), or is it better to purchase blobs frequently and release them quickly (**low latency**)?

We believe that which strategy Rollup adopts depends on the price of the blob. As we mentioned earlier, blob pricing is similar to EIP-1559, so the gas fee increases exponentially when demand increases dramatically. This then leads to the optimal strategy for rollup becoming:

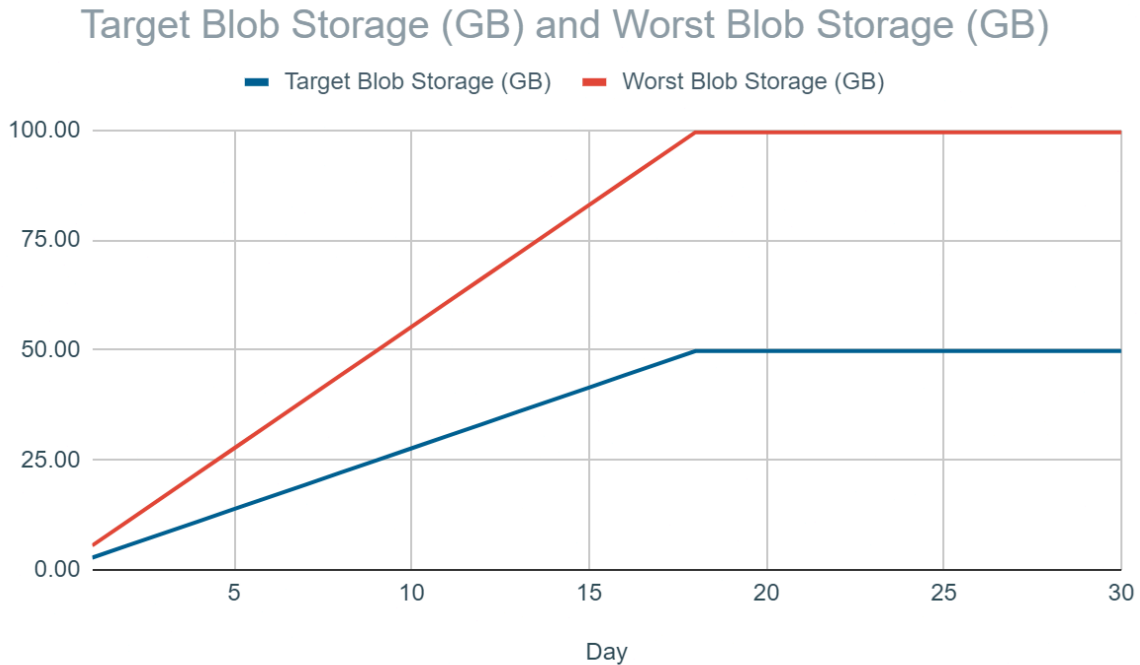
- If the blob is expensive, wait longer then post the full blob.
- If the blob is cheap, quickly buy and post mostly empty blob.

All about Validator and Staking

EIP-4844: Will Blob Cost More for Validators?

According to the previous calculation, the target case for blob is 384 KB per block, and the worst case for blob is 768 KB per block. Block is proposed per slot, each epoch has 32 slots,

so there are 12.288 MB per epoch in the target case, 24.576 MB per epoch in worst case. Since we have 225 epochs per day, there is 12.288 MB per epoch in the target case, 24.576 MB per epoch in the worst case. Recall that blobs are only stored for about 18 days, so the target case is 49.77 GB, and the worst case is 99.53 GB.



From the figure above, nodes only need to store 99.53 GB of data, even in the worst case.

Therefore, blob does not bother validators much in storage.

EIP-7044: Perpetually Valid Signed Voluntary Exits

More Secure and Automatic Staking-as-a-Service

For native ETH stakers, there are two keys:

- **Validator Key:** Performs validator duties like attestation, block proposal, and participation in synch committees.
- **Withdrawal Key:** Similar to keys in a wallet (e.g. MetaMask).

As shown in the table below, there are two ways to keep two keys.

HASKEY CLOUD

	Staking-as-a-Service Provider	User
Non-Custodial	Share/Hold Validator Key	Hold Withdrawal Key
Custodial	Hold Validator Key and Withdrawal key	N/A

Provider usually pre-signs an exit message offer to the user. Thus, when a user wants to exit, he can submit it to the consensus layer anytime. However, the pre-signed exit message valid up-to only two upgrades. For example, the pre-signed message from the previous version of the Capella fork is no longer valid in the deneb fork. This EIP can make the pre-sign message permanent, thus providing a smooth experience for the user.

In addition, according to [Consensys' blog](#): *"A permanently valid pre-signed exit message could be stored in some smart contract or automated mechanism that submits it to the consensus layer when validator is inactive. This would provide additional protection against inactivity penalties in case the validator settings are lost or otherwise taken offline."*

EIP-7045: Increase Max Attestation Inclusion Slot

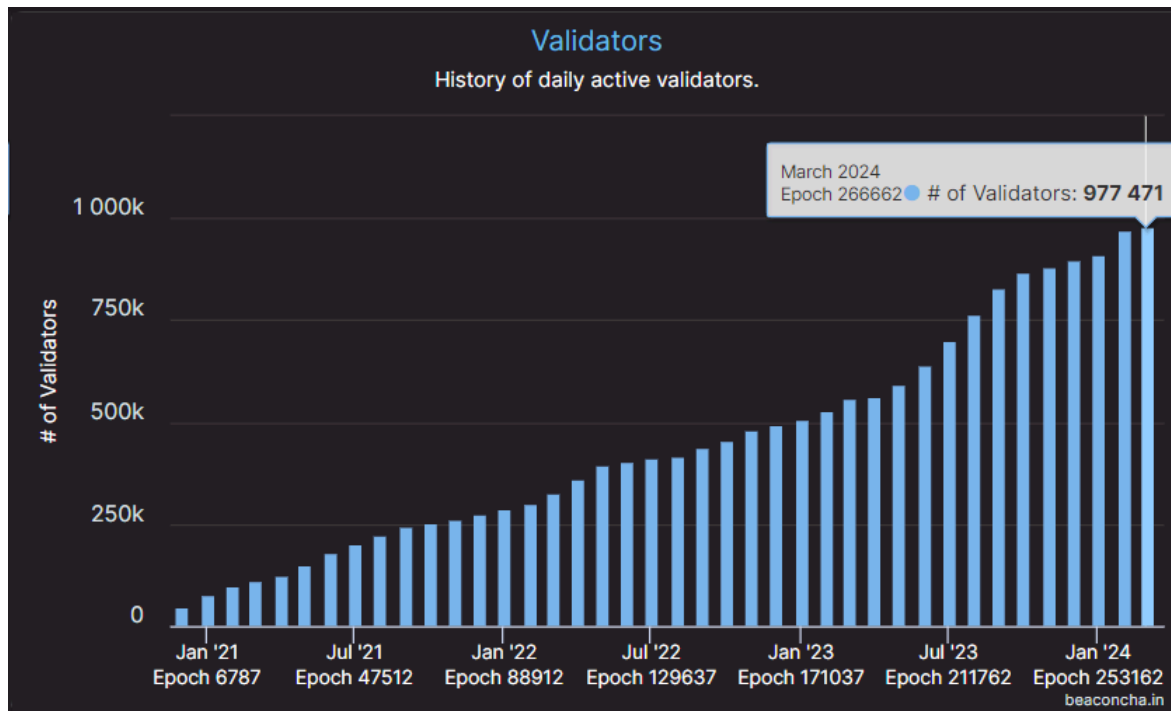
More Relaxed Attestation Requirements for Validators

Previously, attestation to a Target Vote (see details [here](#)) would only receive a corresponding reward if it was contained within 32 slots. EIP-7045 removes this restriction by allowing attestation to a target vote to be included in both the current and the next epoch, meaning that attestations created at the beginning of an epoch have more potential inclusion time slots than attestations created at the end of an epoch.

This would incentivize validators to try still to pack such validations into blocks. The new validation rules allow block validation in 3~4 time slots under normal mainnet conditions.

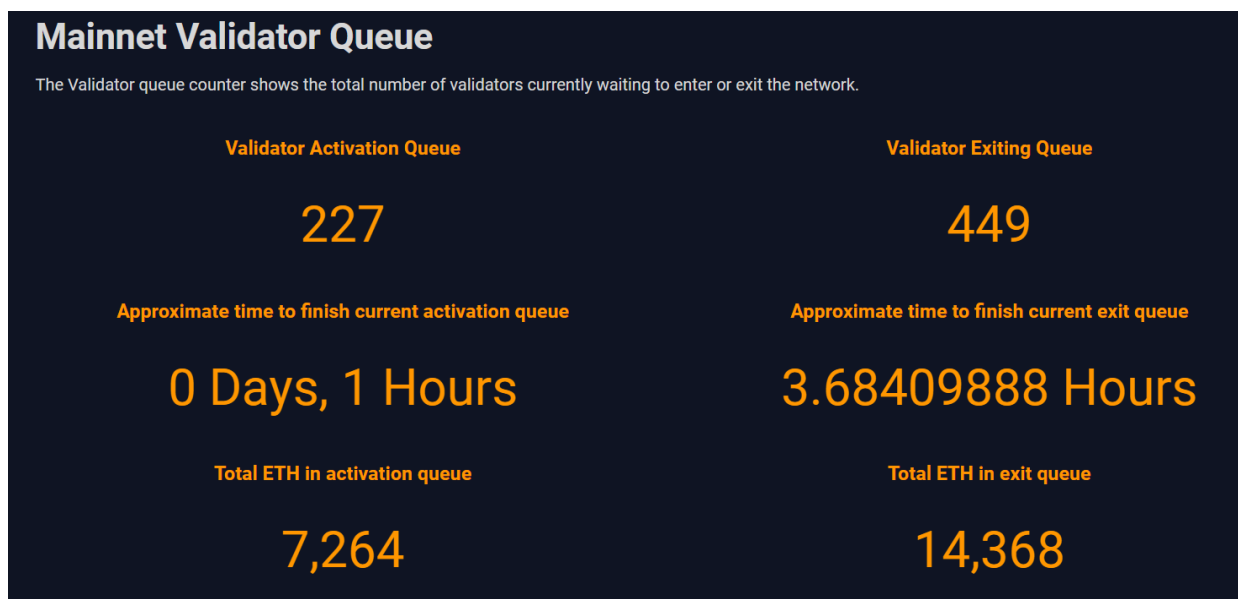
EIP-7514: Add Max Epoch Churn Limit

Slower Validator Growth Rate and Lower Network Capacity



Source: beaconcha.in

According to beaconcha.in, there are currently 977,471 validators, and each epoch can activate about $977,471/65636 \approx 14$ validators, which means that the maximum number of validators that can be activated in a day (225 epochs) is 3150. If more than 14 validators need to be activated per epoch, they must be queued.



Source: [Wen Merge](https://wenmerge.com)

When Merge's data shows current demand for validators remains hot. As we mentioned in a previous post, the consensus layer reward is inversely square rooting with the number of validators. When 100% of ETH is staked, the consensus layer reward APR is still 1.6%, however, this may make it unprofitable for validators other than large node operators who have low costs.

Meanwhile, as validators are increased, the consensus layer is put under more pressure. A large number of validators leads to an increase in gossip messages and beacon state size. Validator has to do more work in such cases.

To incentivize validators and reduce network capacity, EIP-7514 limits the maximum number of validators that can be activated per epoch to 8. In other words, if more than 8 validators are waiting to be activated, they need to be queued.

EIP-4788: Beacon Block Root in the EVM

Native Trustless Oracle: More Secured Liquid Staking and Restaking

Ethereum has two independent chains, also known as the consensus layer (PoS beacon chain) and the execution layer (PoW chain before merge). Informations related to validators (slashed, balance, rewards, etc.) are stored in the consensus layer. Thus, staking-related applications like liquid staking pools and restaking protocols need to use a centralized oracle to get validator data from the consensus layer. Meanwhile, centralized oracle adds more trust assumptions.

EIP-4788 embeds the parent beacon block root into every execution block. In other words, we have native Oracle now. Information from the protocol level is more secure than a centralized Oracle committee, right? After all, **Oracle may lie, but the protocol itself will never lie.**

Liquid Staking Pool

With the native Oracle, liquid staking pools like Lido and Rocket Pool do not need to rely on Oracle to send rewards to users. Whether the Liquid Staking Token (LST) is rebasing or reward-bearing, they can use protocol-embedded Oracle to update the reward information and distribute it to users.

Besides, since they can now trustlessly get rewards and penalties data, they can also build some metrics to rate node operator performance and measure the risk of LSTs.

Restaking Protocol

Restaking protocols like EigenLayer relies on the security of Ethereum validators. At the application level, they let validators willing to join restaking change their withdrawal key to EigenLayer's smart contract. With the native oracle, if a validator got slashed, they can get confirmation and slash the corresponding validator at their DAPP directly, not rely on external oracle information. This significantly improves the security of the EigenLayer: not by an external source, but by Ethereum itself.

Full EIP List

- [EIP-1153: Transient storage opcodes](#)
- [EIP-4788: Beacon block root in the EVM](#)
- [EIP-4844: Shard Blob Transactions](#)
- [EIP-5656: MCOPY - Memory copying instruction](#)
- [EIP-6780: SELFDESTRUCT only in same transaction](#)
- [EIP-7516: BLOBBASEFEE opcode](#)
- [EIP-4788: Beacon block root in the EVM](#)
- [EIP-7044: Perpetually Valid Signed Voluntary Exits](#)
- [EIP-7045: Increase Max Attestation Inclusion Slot](#)
- [EIP-7514: Add Max Epoch Churn Limit](#)

Appendix

[EIP-4844 Blob Gas Model](#)

Reference

1. [Multidimensional EIP 1559](#)
2. [A note on data availability and erasure coding](#)
3. [Proto-Danksharding FAQ](#)
4. [EIP 4844: What does it mean for L2 users?](#)
5. [ETH Global: Blob Merger](#)
6. [EIP-4844 Fee Market Analysis](#)
7. [EIP-4844: Shard Blob Transactions Discussion](#)
8. [EIP-4844 Official Website](#)
9. [Ethereum Cat Herders \(ECH\) Dencun](#)
10. [CL: Deneb -- The Beacon Chain](#)

11. [EL: Cancun Network Upgrade Specification](#)
12. [Ethereum Evolved: Dencun Upgrade Part 2, EIP-7044 & EIP-7045](#)
13. [Ethereum Evolved: Dencun Upgrade Part 3, EIP-4788](#)
14. [Ethereum Evolved: Dencun Upgrade Part 4, EIP-7514 & EIP-1153](#)
15. [Ethereum Evolved: Dencun Upgrade Part 5, EIP-4844](#)
16. [Blobspace 101](#)
17. [Ethereum: Breaking down EIP-7514, and how it could Impact Stakers](#)
18. [🗨 Peep-an-EIP: 4788](#)