Arbitrum One

Executive summary

This report analyzes transaction fees on Arbitrum One, focusing on both revenue and costs to assess profitability. It covers the latest update, the Nitro protocol, and fees before and after EIP-4844 upgrade.

Arbitrum One, an Ethereum-compatible L2 solution, has significantly increased throughput and reduced costs through advanced calldata compression. They also introduce dynamic pricing mode based on “speed limit” for L2 gas adjusts based on demand, balancing efficiency and cost.

L1 gas costs are managed through an adaptive pricing algorithm, ensuring collected fees align with actual costs. Brotli compression approximates each transaction's contribution to the overall batch size, ensuring fair cost distribution. Total profit so far in ETH: 17,268.11.

Impact on our design: Understanding how Arbitrum manages L1 costs through adaptive pricing and compression can influence our design by implementing similar strategies to ensure fairness and charge transactions depending on their data size. The concrete speed limit and adjustment algorithms may not be viable on our sharded system but can serve as inspiration for handling overload and both L2 fee and L1 fee pricing.

Introduction

In this analytics work we are going to cover Arbitrum Nitro protocol fee mechanism as it is the latest update of the Arbitrum One optimistic rollup. Arbitrum One is an ethereum compatible L2 solution, which means that it has the same opcode costs in gas as well as supports the same API as common Ethereum nodes. We will not go over every design detail as it is out of scope for the goals of this research. However we will give some context in order to understand the

protocol better. At present the Sequencer sequences transaction on the first come first served basis. It is centralized and operated by Offchain Labs, however there are plans to transition to a committee-based sequencer. Before EIP-4844 upgrade on 03/14/2024 it used to post its compressed (by Brotli) transaction batches as Calldata, currently it submits blobs. As other L2s it alse has an inbox for direct L1 to Arbitrum One transactions. Let us take a look at how the transactions are processed in Arbitrum One:

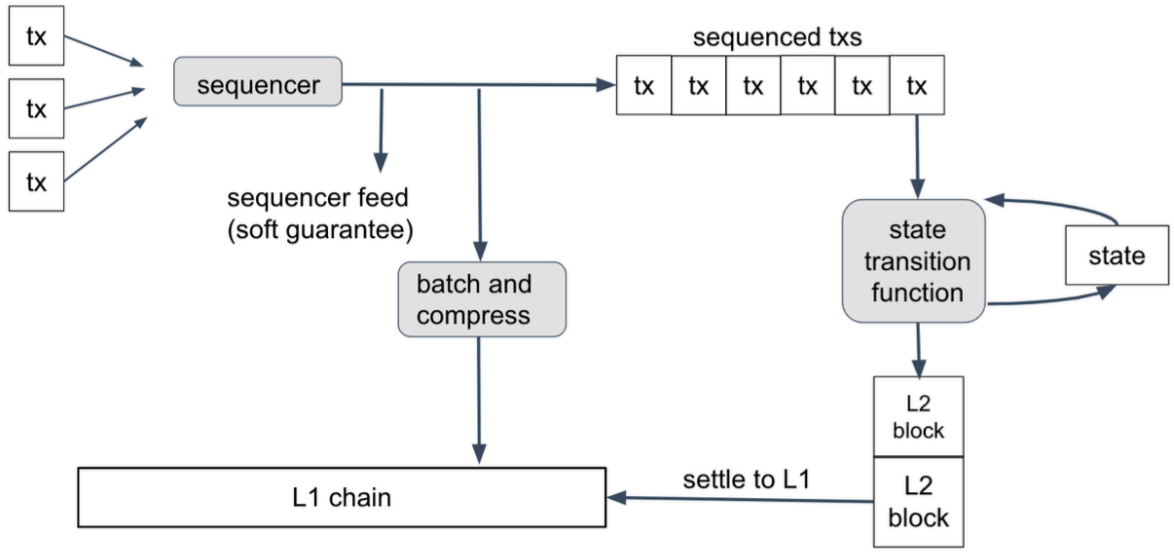


Figure 1: Processing of transactions in Nitro. The sequencer establishes an ordering on transactions, and publishes the order as a real-time feed and as compressed data batches on the L1 chain. Sequenced transactions are processed one at a time by a deterministic state transition function, which updates the chain state and produces L2 blocks. These blocks are later settled to the L1 chain.

Source: Nitro Whitepaper

Nitro upgrade brought these improvements to Arbitrum One:

- Increased throughput: 7x-10x higher compared to pre-Nitro.
- Advanced calldata compression: Reduced transaction costs by minimizing data posted to L1.
- Ethereum L1 gas compatibility: Aligned pricing and accounting for EVM operations with Ethereum.

- Additional L1 interoperability: Included tighter synchronization with L1 block numbers and full support for all Ethereum L1 pre-compiles.
- ...And other upgrades which are out of scope for this research effort.

Goals & Methodology

The main goal of the report is to analyse Arbitrum data and to determine:

1. Data Availability cost (ETH)
2. Before EIP-4844
3. After EIP-4844
4. Fee dynamics
5. Total revenue
6. Economics specifics

This research is performed by obtaining transaction fee data and DA transactions data using Dune Analytics. The Economy and other specifics are acquired by reading official documentation, blogs, and other materials on Arbitrum One network and its Nitro upgrade.

Results & Discussion

Economics Specifics

Arbitrum One uses ARB token. It has two main utilities:

- Governance
- Incentivizing community and financing various operations via Arbitrum Foundation

According to the official documentation allocation looks as follows:

Percentage of initial supply	Number of tokens	Allocated to	Vesting

35.28%	3.528 billion	Arbitrum DAO treasury	Dependent upon the approval of the governance DAO through a voting process
26.94%	2.694 billion	Team and Contributors + Advisors	4-year lockups, with the first unlocks happening in one year and then monthly unlocks for the remaining three years
17.53%	1.753 billion	Investors	4-year lockups, with the first unlocks happening in one year and then monthly unlocks for the remaining three years
11.62%	1.162 Billion	Users of the Arbitrum platform (via airdrop to user wallet addresses)	Available at L1 block 16890400
7.5%	750 million	Arbitrum Foundation	The Arbitrum Foundation's allocation is subject to a lockup that began on 4/17/2023 and linearly unlocks over the course of four years
1.13%	113 million	**DAOs** building apps on Arbitrum (via airdrop to DAO treasury addresses)	Available after L1 block 16890400

Pricing Mechanism Specifics

We will separate gas into two separate portions:

- L2 gas - gas consumed by execution on Arbitrum One
- L1 gas - gas on Ethereum

Each transaction requires an amount of L2 gas depending on the resource used by the transaction itself. It specifies the gas limit (maximum gas allowed to

consume) and specifies the maximum basefee it is willing to pay per gas unit. If it tries to consume more L2 gas the transaction will fail but will have to pay for the L2 gas used, and if the maximum basefee is lower than the current basefee the transaction will not run. The price of L2 gas unit is essentially the base_fee. The price is denominated in ETH. We can notice that the user experience in terms of transaction bidding is the same as with Ethereum. Arbitrum One is also Ethereum compatible in terms of opcode costs as each EVM instruction costs the same number of gas units.

Current model (L2 Gas pricing):

Arbitrum One dynamically adjusts its basefee based on demand, so that when demand exceeds the sustainable capacity of the chain, the basefee increases. When the demand is lower than the sustainable capacity the basefee decreases to stimulate new demand.

The major difference compared to ethereum's adjustment is the speed limit parameter which represents the maximum throughput of the chain. Gas usage is allowed to exceed the speed limit only over a short period and the adjustment model is designed with that in mind. Another difference is that Nitro has variable time between blocks so the adjustment algorithm operates at a one second granularity. As the sequencer attaches timestamps to transactions the chain must handle large number of transactions with same timestamp.

Arbitrum One gas metering algorithm tracks a backlog B, which is updated as follows.

• If a transaction consumes G NitroGas, B ← B + G.
• If T seconds elapse, B ← max(B − TS, 0), where S is the speed limit.

Essentially B tracks how far behind the sustainable speed limit the chain has been during the current burst of usage.
Arbitrum basefee is then calculated as
$F(B) = F_0 e^{\max(0, \beta(B - B_0))}$

- where F_0 is the minimum basefee, and B_0 is a tolerance parameter.
- The scale factor β is chosen so that a 12-second period with gas usage double the speed limit would multiply the basefee by a factor of $\frac{9}{8}$, yielding $\beta \approx \frac{1}{102S}$.

This matches the rate of increase that Ethereum would see if it experienced a 12-second block
at double its speed limit.

Rationale from Offchain labs: “If the demand curve is unchanging, and if demand exceeds the speed limit at the minimum basefee, then the basefee will equilibrate at the level where demand equals the speed limit, and the backlog will be constant and log- arithmic in the equilibrium price.”

Current model (L1 Gas):

As with other L2 solution transaction pricing models transaction on Arbitrum One also uses resources on Ethereum. In order to cover these costs they must be included in the pricing model. These L1 charges are not included when tracking the backlog, because they do not reflect consumption of the Nitro chain’s own resources. The L1 costs are incurred by the Sequencer when it submits data batch transactions on Ethereum.

According to the whitepaper there are two challenges with pricing these resources:

- “It is not obvious how to apportion the costs of a batch among the transactions that comprise it. The posted data is compressed using a general-purpose compression algorithm whose effectiveness depends on patterns shared in common across the transactions in a batch. ” Since there is no obvious and efficient way to determine how much a particular transaction contributed to overall compressibility they approximate it.
- “L1 fee assessed to a transaction must be known to ArbOS when the transaction is sequenced to use information that is available only later would violate the determinism property of the State Transition Function. But at the time a transaction is sequenced, the cost of the batch eventual batch in which it will be posted is not known.”

  Essentially the actual fee depend on the future basefee on Ethereum.

Arbitrum solves the L1 fee assessment challenge in the following way:

- Apportioning Costs among individual transactions
  - They approximate compressibility by applying the Brotli compressor, at its lowest compression / cheapest computation level, to each transaction, and

multiplying the size of the result by 16 as Ethereum charges 16 gas per byte for most data. They use the cheapest computation level in order to be fast enough.

    - “The size of this compressed data is an approximation to the size of the same transaction if compressed with the more aggressive compressor used to build Sequencer batches.” - Whitepaper

- Determining cost per data unit
    - They do not trust the Sequencer to report the L1 basefee since its its earnings depend on it directly.
    - Because the units charged are only an approximation of the costs they might not be corresponding to the Sequencer’s costs.
    - To combat this the data is priced using an adaptive algorithm by the Pricer.

To solve the cost per data unit, the Pricer tracks:

- an amount owed to the Sequencer,
- a reimbursement fund, which receives all of the funds charged to transactions for L1 fees,
- a count of recent data units, to which the number of data units in each transaction is added, and
- the current L1 data unit price, in wei.

The pricer varies the L1 data unit price adaptively, based on this data.

When the Sequencer posts a batch to the L1 inbox, this causes the L1 inbox to insert a ”batch posting report” transaction into the chain’s delayed inbox. After a delay, this transaction will be processed by the pricer in the following way according to the whitepaper:

- The pricer computes the cost of posting the reported batch, and adds that amount to the amount owed to the Sequencer.

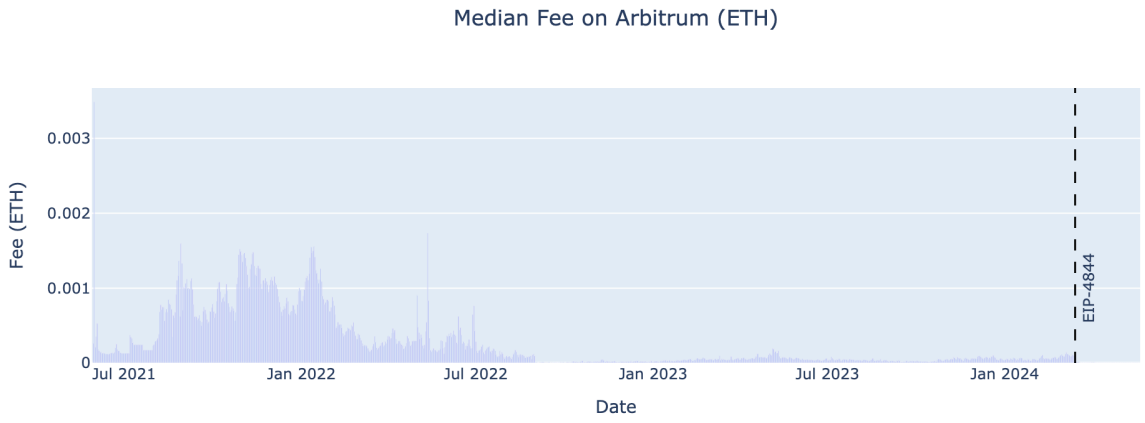
- Then the pricer computes a number of data units assigned to this update, as $U_{\text{upd}} = U \frac{T_{\text{upd}} - T_{\text{prev}}}{T - T_{\text{prev}}}$, where U is the count of recent data units, T is the current time, T_{upd} is the time when the update occurred, and T_{prev} is the time when the previous update occurred. U_{upd} is subtracted from U.
- The pricer pays the Sequencer, from the reimbursement fund, the minimum of what the Sequencer is owed and the balance of the reimbursement fund. The amount paid is deducted from the reimbursement fund and from the amount owed to the Sequencer.
- The pricer computes the current surplus S, which is the reimbursement fund balance, minus the amount owed to the Sequencer. (The surplus may be negative.) It then computes the derivative of the surplus as $D = \frac{S - S_{\text{prev}}}{U_{\text{upd}}}$.
- The pricer computes the "derivative goal" as $D' = -\frac{S}{E}$, where E is an equilibration constant. This is the derivative that must hold on average in order for the surplus to reach zero after E more data units are processed.
- The pricer updates the price to $P = \max(0, P_{\text{prev}} + \Delta P)$

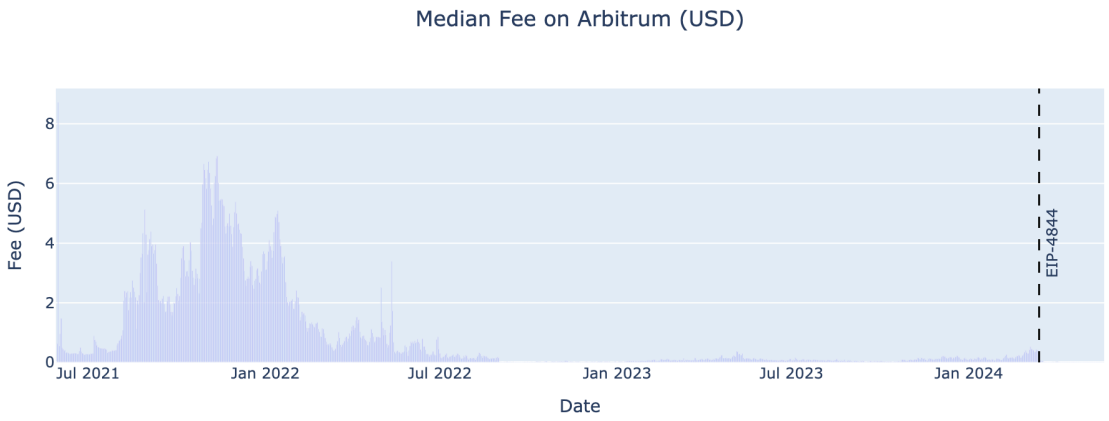
This algorithm ensures the Sequencer’s reimbursement to be nearly equal to its long term costs.

Data Analysis

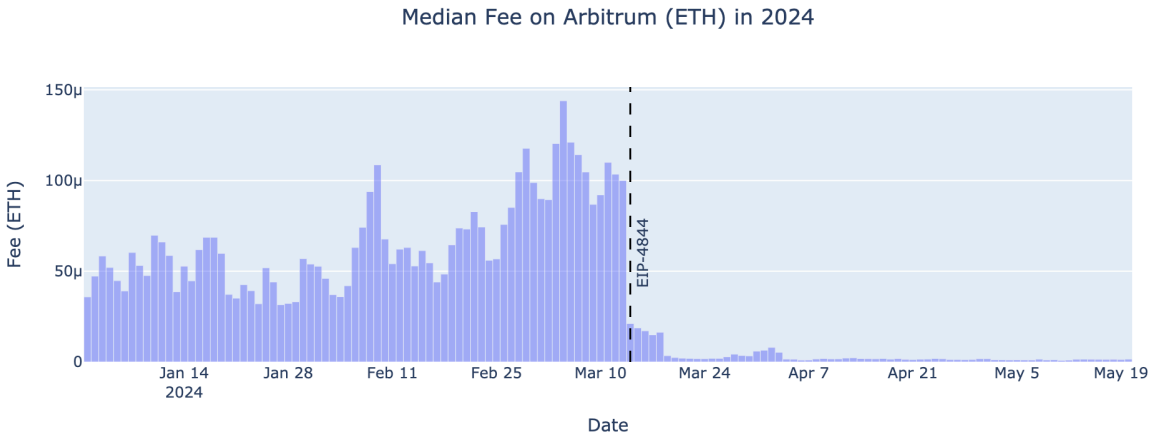
Revenue (Transaction Fees paid by users of Arbitrum)

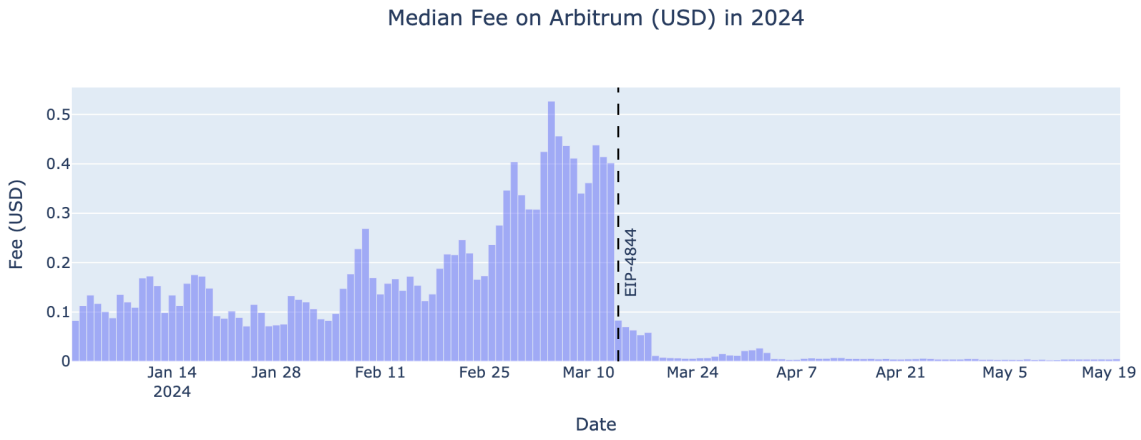
Median fee on Arbitrum:



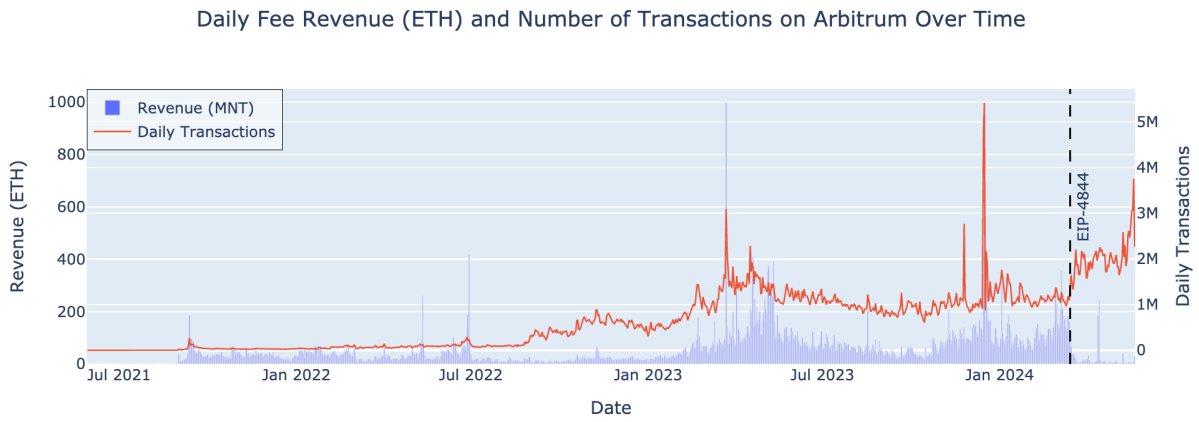


We can see that Arbitrum One started with higher fees. Since the recent fees are multiple times lower let us focus on the 2024 fees:



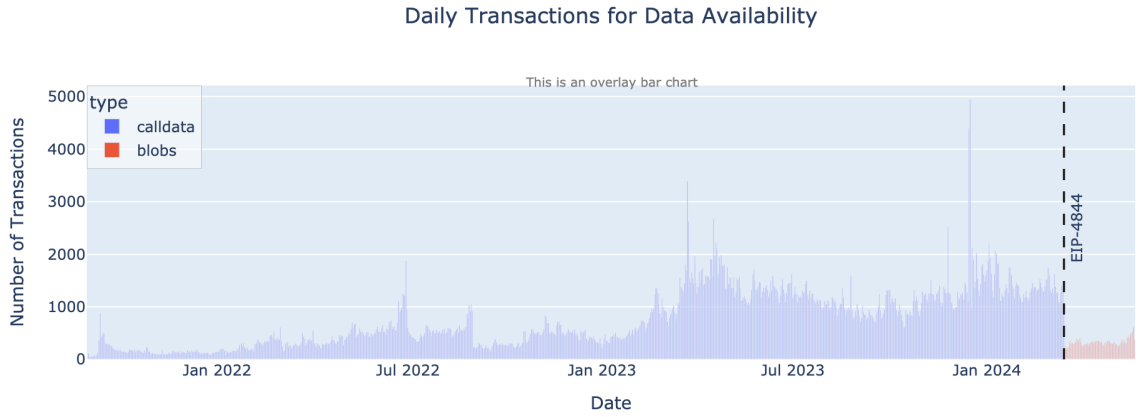


We can see that the EIP-4844 lowered the median fee significantly.

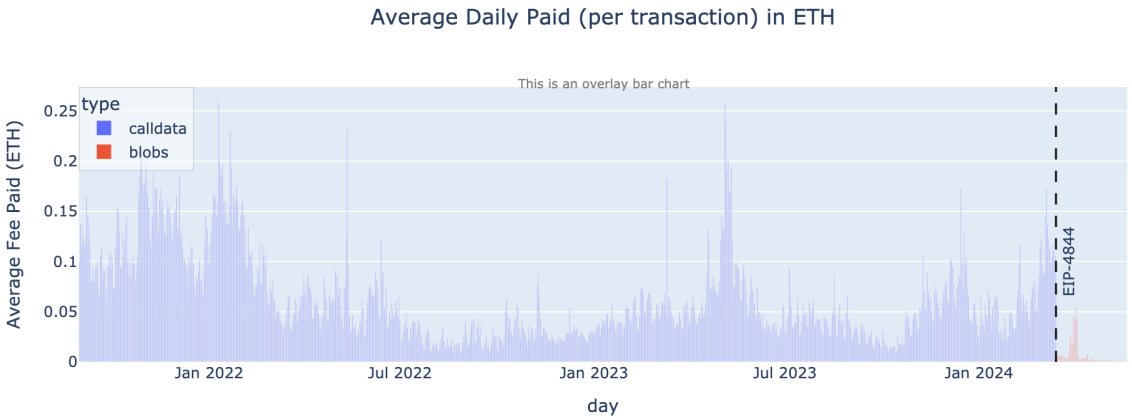


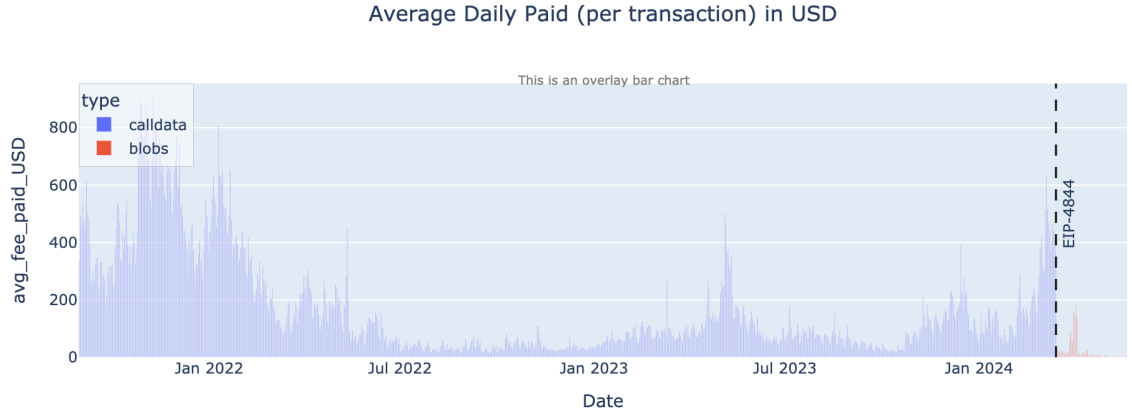
We can see the influence of the EIP-4844 on the amount of transactions and revenue. The lower revenue is expected as the l1 portion of the reduced.

Costs (Transaction Fees paid for Ethereum Data Availability)

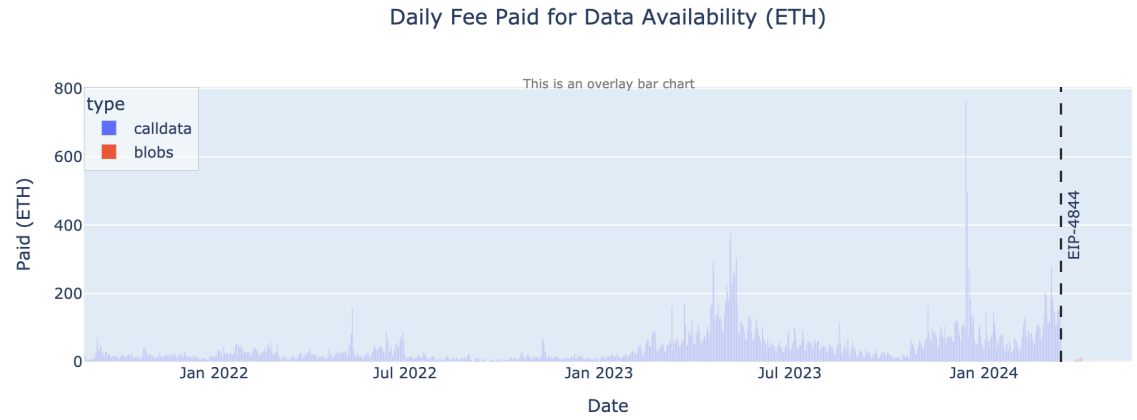


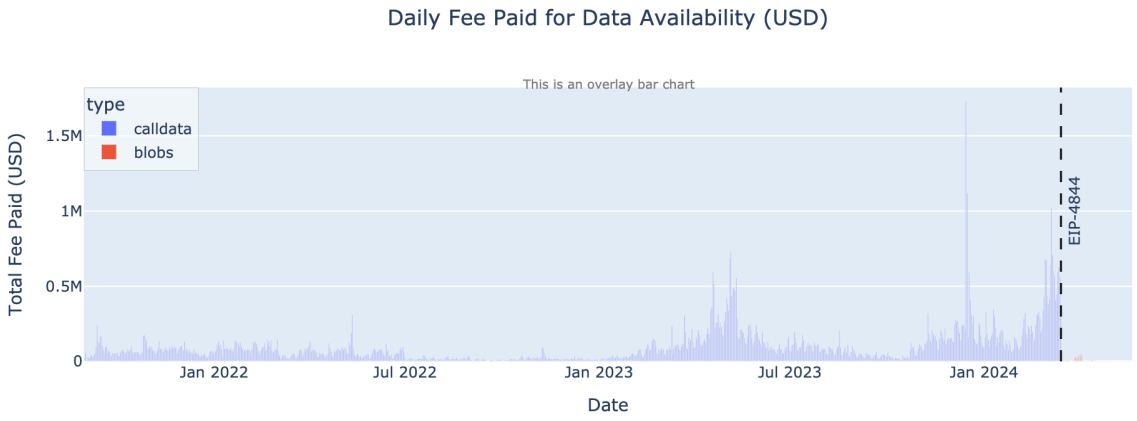
We can see the daily number of transactions stabilize after EIP-4844 upgrade. Let’s take a look at the average fee paid per DA transaction:



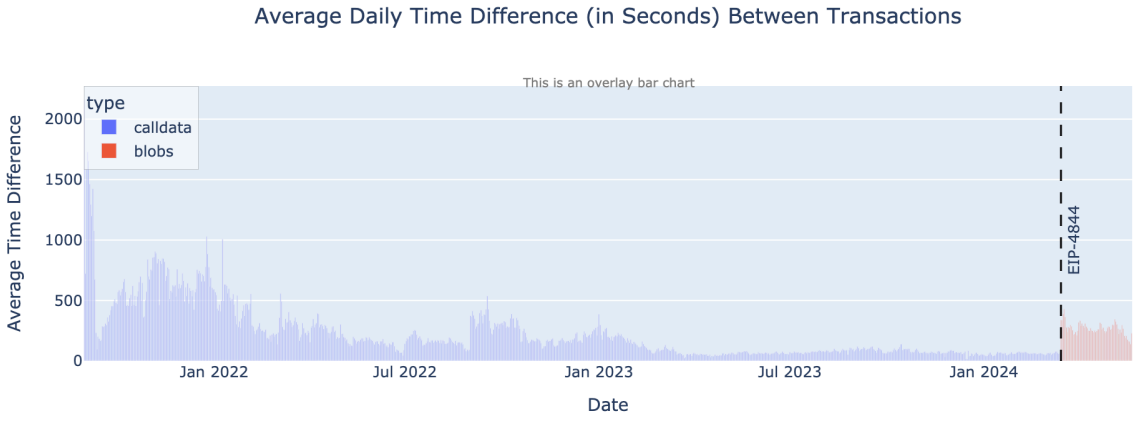


As expected the average fee is reduced. Lets take a look at total fees paid for Data Availability:



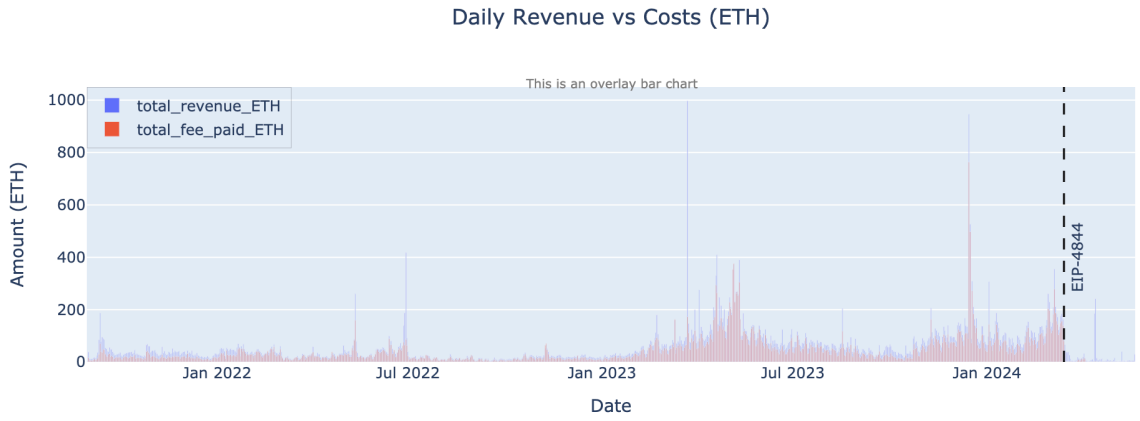


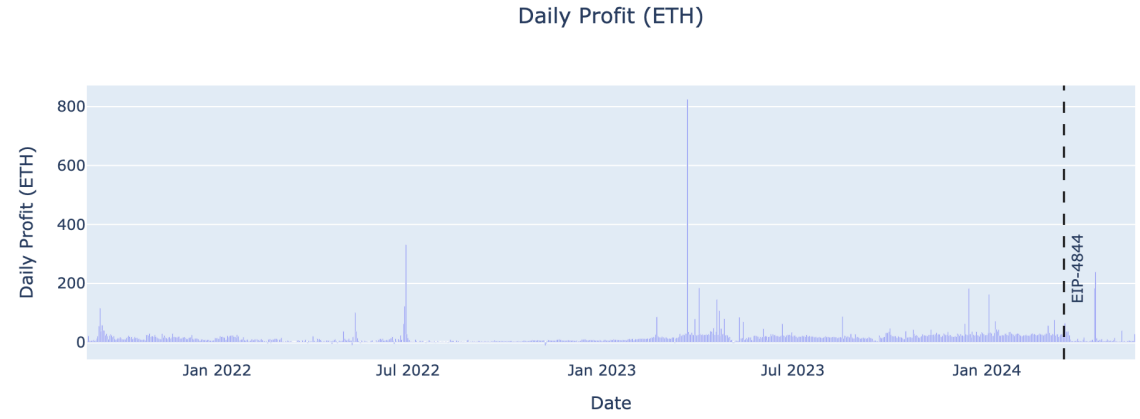
The reduction in costs is significant.



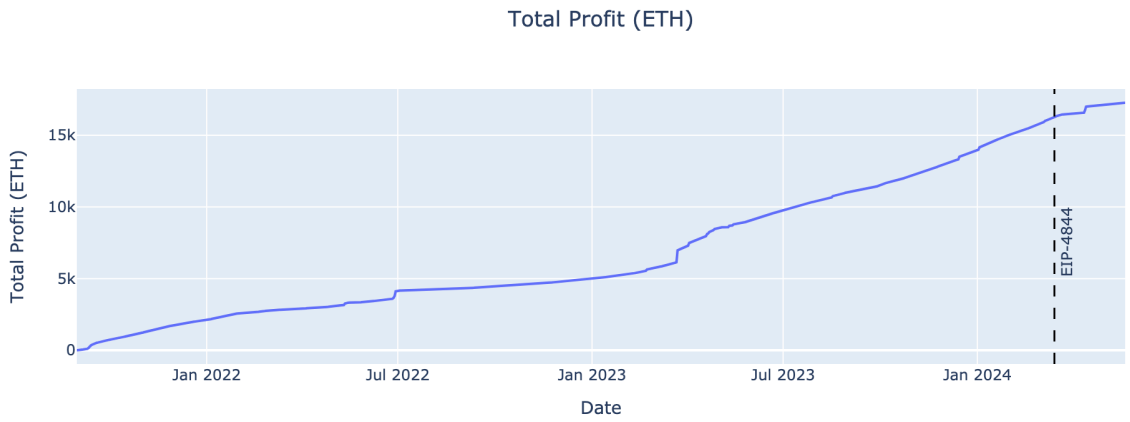
We can see that the introduction of blobs increased the average time difference between transactions as less transactions are needed.

Profitability (Transaction Fees paid by users of Arbitrum - Transaction Fees paid for Data Availability)*





We can see that Arbitrum One is generally profitable. We can see that the daily profit on average is less than pre EIP-4844.



Conclusion

The analysis of Arbitrum One's fee mechanism highlights its pricing model and design principles. The dynamic pricing model for L2 gas adjusts the basefee based on demand, ensuring network responsiveness. The incorporation of a speed limit parameter balances performance and cost, providing a consistent user experience even during high demand periods.

The Nitro upgrade's calldata compression significantly reduces transaction costs by minimizing data posted to L1. Additionally, L1 gas costs are managed through an adaptive pricing algorithm, aligning collected fees with actual costs to maintain financial sustainability.

To allocate L1 costs fairly among transactions, Arbitrum One uses Brotli compression to approximate each transaction's contribution to the overall batch size. This approach ensures a fair distribution of L1 costs among users.

These findings show that Arbitrum One's fee mechanism effectively balances user experience and profitability. The use of dynamic pricing, advanced compression, and adaptive algorithms ensures the platform remains efficient and cost-effective. This approach is interesting and worth considering in our own design to achieve similar fee responsiveness and competitiveness.

Data Tables

Profit table per month:

Month	Total Profit (ETH)
2021-08	3.31168
2021-09	715.321
2021-10	484.839
2021-11	535.634
2021-12	351.401
2022-01	474.016
2022-02	198.398
2022-03	126.882
2022-04	160.737
2022-05	313.819
2022-06	777.126
2022-07	99.6544
2022-08	81.4227
2022-09	113.06
2022-10	151.254
2022-11	186.233
2022-12	223.279
2023-01	265.939
2023-02	481.196
2023-03	1506.38
2023-04	1281.78
2023-05	505.136
2023-06	737.658
2023-07	605.838
2023-08	642.979
2023-09	479.43
2023-10	662.657
2023-11	773.274

2023-12	999.605
2024-01	1086.89
2024-02	790.353
2024-03	685.085
2024-04	612.667
2024-05	154.852

Median Fee Info Before EIP-4844 Implementation (ETH)

Statistic	median_arbitrum_fee_ETH	total_fee_ETH
mean	0.0002646015501256103	56.29552101810677
std	0.0003723682962027095	70.70456033530391
min	0	0.0023458133999999
25%	0.00003797353470914383	18.544270127950103
50%	0.00007678567373064855	36.07016113133942
75%	0.0003196005805165 75	73.02905620188292
max	0.0034874993708496	997.7531751207272

Median Fee Info After EIP-4844 Implementation (ETH)

Statistic	median_arbitrum_fee_ETH	total_fee_ETH
mean	0.000003198234876705 3887	18.90881608102154
std	0.00000435832476060 2384	38.56298093896501
min	9.180444873662738e-7	2.964552163759499
25%	0.00000138296712709 6234	5.027207831979385
50%	0.00000171892429763 85185	6.760460283443033
75%	0.00000210788907700 47424	14.321519124697382
max	0.00002119568401594 9912	242.15224126144463

Median Fee Info Before EIP-4844 Implementation (USD)

Statistic	median_arbitrum_fee_usd	total_fee_usd
mean	0.8393507040514332	125397.27257375159
std	1.412841457451403	159573.1263800298
min	0	5.635522652306454
25%	0.0674486348298034	29744.015557401235
50%	0.15204245861701654	92505.10586403993
75%	0.8107607954569009	159685.94347092087
max	8.726427343707487	2144887.479257358

Median Fee Info After EIP-4844 Implementation (USD)

Statistic	median_arbitrum_fee_usd	total_fee_usd
mean	0.011080294737499316	64073.3381799468
std	0.01642695941158188	130872.17689182133
min	0.0026792706955863	8651.909408752903
25%	0.0043085685977072245	15730.44425286386
50%	0.0054427872843996	21524.14833716511
75%	0.0073667278820025005	46700.17706395616
max	0.0832336708403699	776017.0644840236

Descriptive Statistics for DA Before EIP-4844

statistic	metric	calldata
total_transactions	mean	762.6306695464363
total_transactions	std	550.902953820936
total_transactions	min	11
total_transactions	25%	310.25
total_transactions	50%	575.5
total_transactions	75%	1191
total_transactions	max	4951
avg_time_diff	mean	222.72407904118222
avg_time_diff	std	230.5584016540715
avg_time_diff	min	17.37830741264391
avg_time_diff	25%	72.22383610541505
avg_time_diff	50%	149.80715579710147
avg_time_diff	75%	277.2456876879992
avg_time_diff	max	2159.818181818182
avg_fee_paid_ETH	mean	0.06357822448350475
avg_fee_paid_ETH	std	0.044152023517700735
avg_fee_paid_ETH	min	0.0086387630409903
avg_fee_paid_ETH	25%	0.03232110660256063
avg_fee_paid_ETH	50%	0.04901544925833775
avg_fee_paid_ETH	75%	0.08085017145537904
avg_fee_paid_ETH	max	0.2601504918675889
total_fee_paid_ETH	mean	44.43855560156067
total_fee_paid_ETH	std	53.3446381698396
total_fee_paid_ETH	min	1.203294013529142
total_fee_paid_ETH	25%	15.057069912343
total_fee_paid_ETH	50%	26.877129339316532
total_fee_paid_ETH	75%	55.71986539186777
total_fee_paid_ETH	max	763.8449128898897

Descriptive Statistics for DA After EIP-4844

statistic	metric	blobs
total_transactions	mean	333.14925373134326
total_transactions	std	70.59407085827914
total_transactions	min	200
total_transactions	25%	291.5
total_transactions	50%	325
total_transactions	75%	358
total_transactions	max	613
avg_time_diff	mean	268.1155679025729
avg_time_diff	std	50.1764453178103
avg_time_diff	min	140.51549755301795
avg_time_diff	25%	240.92324617869434
avg_time_diff	50%	265.62461538461537
avg_time_diff	75%	294.8836530540191
avg_time_diff	max	429.84
avg_fee_paid_ETH	mean	0.0064885891166030765
avg_fee_paid_ETH	std	0.011014052995040857
avg_fee_paid_ETH	min	0.0007297111516465
avg_fee_paid_ETH	25%	0.00147277030732695
avg_fee_paid_ETH	50%	0.0027868535738723
avg_fee_paid_ETH	75%	0.0053582163742945
avg_fee_paid_ETH	max	0.0561694830529159
total_fee_paid_ETH	mean	2.0128378914483194
total_fee_paid_ETH	std	3.185419472852842
total_fee_paid_ETH	min	0.2690106540261748
total_fee_paid_ETH	25%	0.49494140290752686
total_fee_paid_ETH	50%	0.8472034864571848
total_fee_paid_ETH	75%	1.5562488571764472
total_fee_paid_ETH	max	16.51382801755728