MEV on L2

FlashBabies (Ha, Vlachou, Kilbourn, De Michellis)

Dec 12 2021

Contents

1 Introduction

2021 is the year of roll ups and MEV for the blockchain world. The boom in DeFI has lead to an increasing dissatisfaction pertaining to low block space/rate and high gas fees on L1. This resulted in a large interest in L2 scaling solutions, which move computation and/or data off-chain, posting only a small summary of activity on L1 while still inheriting the security of Ethereum mainnet. Chains like Arbitrum have already attracted significant volume and metrics like total value locked (TVL) are expected to increase rapidly in coming months. In parallel to the growing emphasis on L2 technology, Maximal Extractable Value [1] (MEV) has become a central topic of discussion on blockchain forums and academic papers. Topics of debate range from whether MEV is inherently detrimental to the health of the blockchain ecosystem, to the feasibility of different solutions designed to mitigate MEV. Since value which can be extracted at the application layer provides incentive for consensus destabilization, it is important to render MEV extraction transparent and measurable, at the very least. Based on this observation, FlashBots, one of the foremost research groups in the space, developed and released mev_inspect.py, a tool which classifies transactions into different MEV extraction models.

In this project, we study the marriage of these two topics: MEV on L2s. In summary, our contributions are

1. Summarizing what is currently known about MEV, including main extraction techniques, quantification techniques, and open challenges.

2. Modifying an MEV quantifyer tool to work with GETH traces, deploying it on Optimism.

2 Background

2.1 Miner Extractable Value

Miner Extractable Value (MEV) [1] was first coined in 2019 by Daian et al as a measure of the profit miners can make by arbitrarily reordering, including, and excluding transactions within the blocks. In Eth1 miners are the only party that can guarantee the inclusion of a transaction or an MEV opportunity within a block, hence they are the ones that are supposed to collect the profit [2]. However, in practice specific users, called searchers, also accrue MEV by watching the mempool, analyzing blockchain data to detect MEV opportunities and running bots that automatically submit the respective transactions. These bots participate in Priority Gas Auctions (PGAs) in an attempt to frontrun each other by incentivizing miners to execute their transactions first, which subsequently increases the transaction fees that miners receive [1]. As a result, miners still accrue a portion of the MEV. After the realization that any participant (e.g., miners, searchers, bots) can extract MEV, MEV was renamed to Maximal Extractable Value.

A lot of research has been done on MEV strategies on L1 ETH, with some of the most common being the following:

1. Decentralized exchange (DEX) arbitrage: Searchers can leverage inefficiencies in DEXs that offer the same token at different exchange rates [2]. To do so, they can execute atomically in one transaction the following two orders: 1) an order to buy the token from the pool with the low exchange rate and 2) and order to sell the token on the pool with the high exchange rate. To make sure that the orders will both either succeed or fail, they can use a smart contract called proxy contract that executes multiple orders atomically within a single transaction.

2. Cyclic arbitrage: Starting with Uniswap2 and the usage of Wrapped Ether (WETH), it is possible to create pools between any two tokens with the exchange rates of the pools not being constant [3]. Searchers -or in this case arbitrageurs- can submit atomic cyclic transactions (a series of DEXs that start and end with the same asset) to leverage the unbalanced exchange rates of the pools [4]. For example, an arbitrageur can submit transactions to atomically exchange token A for token B, token B for token C, and then token C for token A to accrue MEV if the product of the exchange rates is larger than the total amount of transaction fees.

3. Sandwiching: the sandwiching technique is a combination of frontrunning and backrunning a DEX transaction [5]. Searchers are watching the mempool to identify transactions that present profitable MEV opportunities by having a significant impact on the price of an asset. After the identification of such a transaction, called a victim transaction, their bots submit a frontrunning transaction and a backrunning one to leverage the exchange rates variance. For example, suppose that the victim transaction increases the price of asset A. The bot will submit a frontrunning transaction to buy A -that will slightly increase the price-, the victim transaction will then move the price up even more and the backrunning transaction submitted by the bot to sell A will sell at a price better than the original. Since most clients order transactions within the blocks according to their gas fees, bots can do successful sandwich attacks with high probability by adjusting the gas price of the transactions they submit.

4. Liquidations: the existence of lending protocols on L1 provides searchers with MEV opportunities. Searchers are monitoring the mempool to detect borrowers that can be liquidated and run bots that submit transactions that aim to collect the liquidation fee [2].

MEV is a potentially consensus-destabilizing landscape. In particular, very profitable MEV opportunities can lead to two different types of attacks: 1) fee-based attacks [6], where miners fork blocks that include high MEV and 2) time-bandid attacks [5] where miners try to fork a large part of the chain in order to levy past MEV. For this reason, it is important to mitigate the

negative potential consequences of the known MEV strategies. Flahsbots [7] is a research organization that focuses on shedding light on MEV to enable searchers to capture MEV opportunities without being frontrun and mitigate the negative externalities of MEV. Eskandari et al [8] have identified a number of ways to mitigate front-running attacks on L1, which they classify as follows:

1. deprive miners of the ability to arbitrarily order transactions by enforcing a specific ordering (transaction sequencing)

2. leverage cryptographic techniques (e.g., commit/reveal) to reduce the visibility of transactions and ensure confidentiality so that the searchers do not have the necessary information to engage with frontrunning

3. redesign DApps to inherently eliminate ordering/time dependencies.

In the past months Layer 2s (L2s) have garnered a lot of attention since the network congestion problem and the high gas fees on Ethereum have pushed users to find more scalable and less expensive solutions. This growth in scalability solutions built on top of a L1 has brought new challenges and increased complexity in the potentially consensus-destabilizing MEV landscape. Many of the L2s have been running only for a few months (e.g., Optimism, Arbitrum) and hence, very little is known about the state of MEV on L2, the effectiveness of the MEV strategies, and the vulnerabilities they exploit.

In L2s, centralized operators or sequencers receive transactions and can accrue MEV by censoring, inserting their own, or modifying the true sequence of transactions. A known way to mitigate this are MEV Auctions (MEVA) [9] which redirect fees initially directed at miners back to the ecosystem by having miners compete to become sequencers and having the funds go to the community. However, this mitigating technique comes with its own shortcomings since sequencers will have the incentive to extract as much MEV as possible in the course of their short tenure. Chainlink has been developing Fair Sequencing Services (FSS) [10, 11], a decentralized transaction ordering service that mitigates MEV by decentralizing the sequencer and adding fairness and predictability in the time-ordering of transactions.

2.2 Layer 2s

Rollups [12] are a L2 scaling paradigm that aims to solve the network congestion problem by moving computation and state storage off-chain while keeping the bare minimum of transaction data on chain. There is an on-chain smart contract which stores the state root, i.e., the Merkle root of the state of the rollup, that enables fraud detection. In rollups, the transactions are published in batches. There are two types of rollups:

1. optimistic rollups that optimistically assume correctness until challenged by on-chain fraud proofs [13]. After a batch is published, any party can publish within a certain dispute period a fraud proof that shows that one

the of the batch's state transitions was invalid. If the proof is verified, the invalid batch as well as the following batches will be reverted. While a long dispute period results in a long finality window it can also act as a mechanism against censorship attacks.

2. ZK rollups [13, 14] where every batch includes a validity proof -called ZK-SNARK- that has been computed off-chain and proves the correctness of the new state. ZK-SNARKS enable instant batch verification and hence short finality time.

A sequencer [12, 13] is a party that is responsible for executing and storing on L2 transactions submitted by users, and for submitting the state root of the batch of the respective state transitions on L1. In L2, sequencers can be chosen in multiple ways with the most common being the centralized sequencer, i.e., a single operator that can submit batches. While this is an efficient solution, it relies on a single actor for liveness and hence introduces centralization in the L2s. One simple way to decentralize the sequencer would be to still use one sequencer at a time but use a mechanism to rotate the sequencers. As discussed in Section 2.1, sequencer auctions (e,g,, MEVA [9]) and FSS [11, 10] developed by Chainlink can decentralize the sequencer and increase trust-minimization.

In optimistic rollups sequencers play a major role and factoring in that they can introduce centralization into the system, it is important to understand how they work. Users send transactions to the sequencer, who checks their validity, executes them, and rolls the resulted state transitions up in a batch that will submit on L1 for finalization. Since the sequencers are the actors that choose the transactions that will be included in the next block, a concern that arises is whether the sequencer can censor transactions and how can the liveness of the system be ensured in case the rollup has a centralized sequencer that suddenly goes offline. Sequencers are required to include users' transactions in a block within a time window called "forced inclusion period" [15]. If the sequencer does not include some user transactions (called "forced transactions") within that period, the next L2 block will include only those forced transactions. This mechanism -that is enforced via fraud proofs- essentially allows the users to bypass the sequencer if needed and submit transactions directly to L1.

Optimism or Optimistic Ethereum is an EVM compatible L2 chain that uses optimistic rollup with single round fraud proofs to increase Ethereum's throughput and decrease transaction fees [16]. As a result, the transactions submitted by users are executed faster and with lower gas cost. In optimistic rollups, sequencers are responsible for rolling up transactions into batches that will then submit to Ethereum. Optimism uses a single centralized sequencer that as discussed cannot censor user transactions. A key characteristic of Optimism is that currently blocks include only a single transaction [17].

Another EVM compatible L2 solution that uses optimistic rollups -this time with multi-round fraud proofs- to increase Ethereum's capacity, scale dApps,

and decrease transaction costs is Arbitrum [18]. Abtitrum used a centralized sequencer, but it is currently exploring Chainlink's FSS [11], a decentralized transaction ordering solution that helps decentralize the sequencer and ensure fair ordering by using decentralized oracle networks to collect transactions from users, order them, and submit them to L1. An important characteristic of Arbitrum that is related to MEV opportunities is that there is no mempool for bots to frontrun. Of course, it is possible for mempools to arise in applications.

Polygon is an interoperability layer 2 solution that builds and connects Ethereum-based blockchains [19]. It was designed to solve the problem of high gas fees and increase transaction throughput while maintaining security. However, the reduced gas fees resulted in Polygon facing bot-spamming attacks [20]. In order for a frontrunning, backrunning, or sandwiching attack to succeed the bot's transaction needs to land right before or behind the victim transaction respectively. Since users or bots cannot choose the ordering of their transaction, they need to set the gas fees in such a way that the transaction will land in the right place. Since the transaction costs at Polygon are low, the bots can engage with the hit-and-trial method to spam the network with identical transactions in hope that one of them will land in the right place. If the transaction lands in another place, the bot will cancel the transaction to save gas. That means that the chain will include multiple cancelled transactions that consume gas and space but do not change its state.

3 MEV Inspector

In the introduction we argued for the importance to create tools that help us better understand and quantify MEV. Rollups are gaining traction this year and therefore it's important to make them as transparent as possible from the start. Ideally, they are where most of the application-layer volume will flow to in the Ethereum ecosystem and therefore making them as trustworthy as possible is imperative. This is in accordance with the current vision of making the Ethereum mainnet less congested and reducing it's main use to becoming the execution layer for scalability solutions.
To quantify the amount of MEV extracted on Ethereum mainnet Flashbots have built mev-inspect-py. Mev-inspect-py categorises each transaction by computing the transaction traces and then if the transaction pattern fits with one of the patterns identified as MEV it quantifies how large the extraction was and adds it to a SQL database. As, unfortunately, happens with most codebases we can't take mev-inspect-py out of the box and implement it on other chains. In order to understand the limitations and what to modify we will first provide some information on the different Ethereum clients and how they implement transaction traces. We will then discuss how mev-inspect-py works, the necessary modifications implemented by the Marlin Protocol team to get it to work on Polygon, and how we have gotten it to work on Optimism.

3.1 Preliminary Knowledge

In order to understand how mev-inspect-py works and what the problem with it's initial implementations were we will briefly discuss Ethereum clients and how they implement transaction traces.

3.1.1 Ethereum Clients

An Ethereum client is a piece of software built to communicate with other clients, signing and broadcasting transactions, deploying smart contracts and updating our local state.
Having multiple clients on which users can run nodes is central to keeping the network decentralised, as it makes the network healthier and more resilient to bugs on any one client. This is why recently, on December 14, 2021 the Ethereum Foundation launched a Client incentive program [21] to reward different client's continuing to build. Different clients offer different implementations of features, apis and storage of data which result in different tradeoffs and appeal to different users. Currently, approximately 80% of the nodes on Ethereum run go-ethereum [22], a client developed by a team within the Ethereum Foundation, which is developed entirely in Go. The second most used client is OpenEthereum (formerly Parity), which is run by 10% of nodes [22], and is written in Rust. Erigon (5% of nodes [22]), which also spawned from the Parity client, is a new client which has a more efficient implementation of its db structure.
To develop their clients most L2s have used forks of Geth. This is because Geth is the most popular client on Ethereum and it's codebase is transparent and more easily modifiable. Polygon, which launched about a year before the Optimistic L2s which we've looked at, forked Geth to develop its own client Bor. The client can support the higher transaction throughput and the different consensus algorithm that Polygon has. Arbitrum and Optimism have also developed forks of Geth tailoring them to their specific needs.

3.1.2 Transaction traces

There are two types of Ethereum transactions: plain value transfer and contract executions. Transactions that invoke an execution on a smart contract will cause a state change in the Ethereum Virtual Machine (EVM). When an Ethereum transaction is executed the clients will store the state change to the EVM and any emitted events in the transaction receipts. However, there is no way of getting more details on what contracts the transaction interacted with and what were the results of these interactions unless we reexecute the transactions, this is the information transaction traces give us.
Tracing a transaction requires requesting an Ethereum client to reexecute the transaction with different levels of data collection and return it aggregated in some way. Different clients will run transaction traces in different ways. The original version of mev-inspect-py was developed with transaction traces in the format that Parity clients (OpenEthereum, Erigon) follow. Parity traces return

every single call made by the transaction with the input parameters and recipient address of the call. This is, in fact, all that is needed to classify a transaction for MEV since by using ABI to decode calldata of known formats. We can get transaction traces by calling the 'track_block' api endpoint on a Parity client.
Unfortunately, L2 clients have been developed with forks of Geth which trace transactions slightly differently. The basic transaction trace that Geth produces are raw EVM opcode traces. For every VM instruction, the transaction execution returns an entry with what the stack, memory and storage are at that point in the execution. This is clearly a lot more information than we need. To solve this problem the Marlin Protocol implemented a filter of the traces to get an output that resembles that of Parity which, with some other necessary modifications, can then run in mev-inspect-py.
To summarise, mev-inspect-py was initially developed to work with Parity-like traces. However, the L2s which we considered have developed clients that are forks of Geth. This means that in order to run the software we need to modify the program to accept Geth traces by modifying the output of the filtered geth traces to being equivalent with Parity traces.

3.2 mev-inspect

mev-inspect-py is a software created by Flashbots with the aim of identifying, classifying and quantifying MEV on Ethereum with the aim of "illuminating the dark forest". Before they developed this there was only anecdotal information of how much MEV was being extracted on Ethereum and who the major players were. mev-inspect-py works by analysing the transaction traces of each Ethereum block, classifying them according to whether the transaction can be classified as MEV, so whether the transaction is an arbitrage or liquidation and, finally save all of these results to a sql database. We now give some clarifications on how mev-inspect works by looking at the function inspect_block() which classifies all transactions inside a block.

1. Along with the block we want to inspect we provide client endpoint and a sql db to the function.

2. We then fetch all the information from the block which we will require to classify. That is: transaction traces, transaction receipts and block metadata (timestamp, miner, base fee)

3. Using the traces classify the traces for each transaction. The developers created specs for each protocol / exchange that they wanted to analyse, they included the ABI (application binary interface) for all functions that could be called which could be included in a transaction with MEV (e.g swap() for a uniswap pool). Then the software iterates through each transaction and checks whether the calls we identified when tracing correspond to any of these specs. If the classifier is successful we save it.

4. The classified traces are then further refined to check for different types of MEV: arbitrages, liquidations, punk snipes. For each of these we can

calculate what the value generated by the transaction was and how much profit was paid to the miner.

As we discussed, mev-inspect-py was initially implemented with Parity-like traces. The modifications that the Marlin protocol implemented which we then piggy-backed on to run it on Optimism. These modifications were mainly at the tracing level, in order to make the Geth traces compatible with the mev-inspect format they used filtered tracing and modified the output to complement.

Another important modification was made by the Marlin Protocol team to be able to run mev-inspect-py on Polygon. The problem, as we mentioned in the background section, is that Polygon blocks due to their low cost of failed transactions are congested with transactions sent by bots wanting to backrun transactions. This means that it's very expensive to trace a block because not only there are more transactions per block but block rate is around 1 second. So to get around this problem they first filter using event logs from transaction receipts. Since most actions taken by a bot like a Uniswap swap emit an event, we can just filter for transactions which have that topic in their logs.

3.2.1 Running mev-inspect-py on Optimism

Although were able to piggyback on the work done by the Marlin Protocol team to use Geth traces, we still needed to make some changes to successfully run mev-inspect-py on Optimism. These were mainly due to the differences in the Optimism clients and some bugs that were introduced in this new, less battle-tested, version of mev-inspect-py with Geth traces. The changes we had to make to mev-inspect-py:

1. Made some modifications to web3.py to support parsing of traces from geth
2. Debugging of mev-inspect-py geth branch, especially due to incorrect use of the Python asynchronous framework to successfully await calls to the geth node

3.3 Running Nodes

In order to even attempt to run mev-inspect-py on Optimism and Arbitrum we had to run our own nodes to access the tracing api modules. Unfortunately, at this time, Arbitrum does not provide tracing of transactions. Since Arbitrum transactions are run in the AVM which has some structural differences compared to the EVM. We did run a node to test whether the event based method would work but realised that ultimately we needed traces and our work on Arbitrum was put on standby until the team Arbitrum team releases a tracing api.
We were, however, able to successfully ran an Optimism node and then run mev-inspect-py on it. The work we had to here required a lot of trial and error to be able to get the node to sync successfully.

4 Results

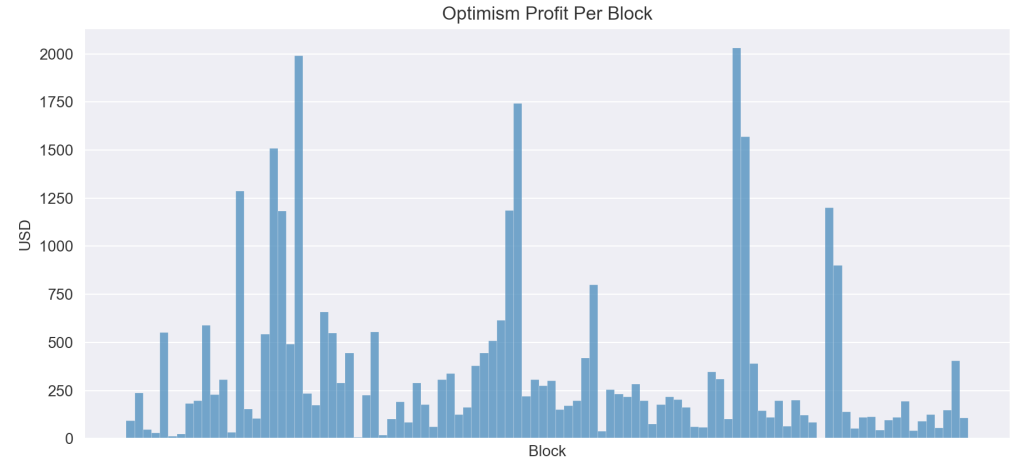
4.1 About The Data

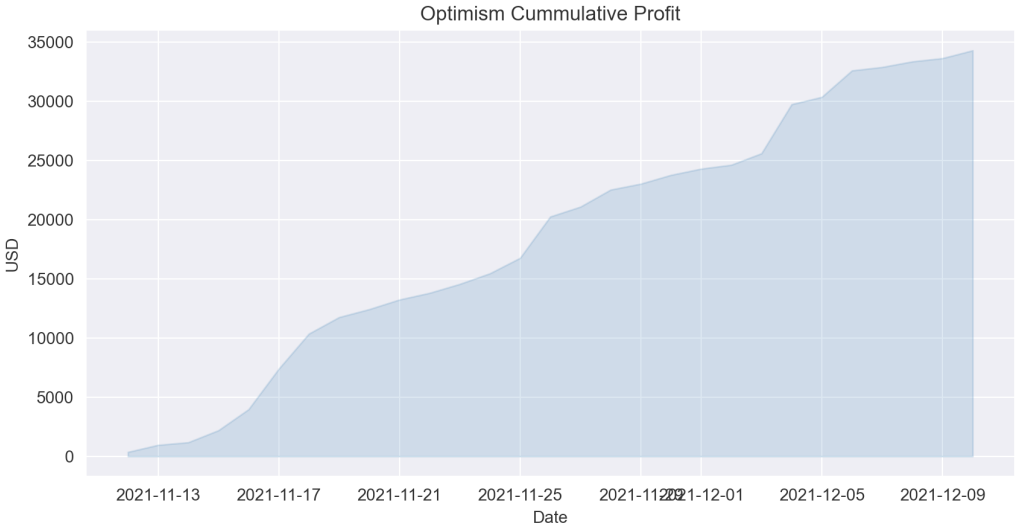
The measure of the value of each transaction must be considered as an approximation and not an exact value for the following reasons:

- We were only able to access hourly historical price data. Hence, in times of extreme volatility, the value at the exact moment of transaction may be quite far removed from the value we used.
- There is not objective value of a token at a given time. Leaving aside stablecoins, which we have assumed to have a 1:1 peg with USD, other common tokens like wrapped ETH do not have a clear market price. As such, we can only use an estimate for the "true" price. In our analysis we pulled price data from CryptoCompare who aggregate price data from several different sources. Again, in times of high volatility, there is certainly room for disagreement about the true price of an asset.

4.2 Optimism

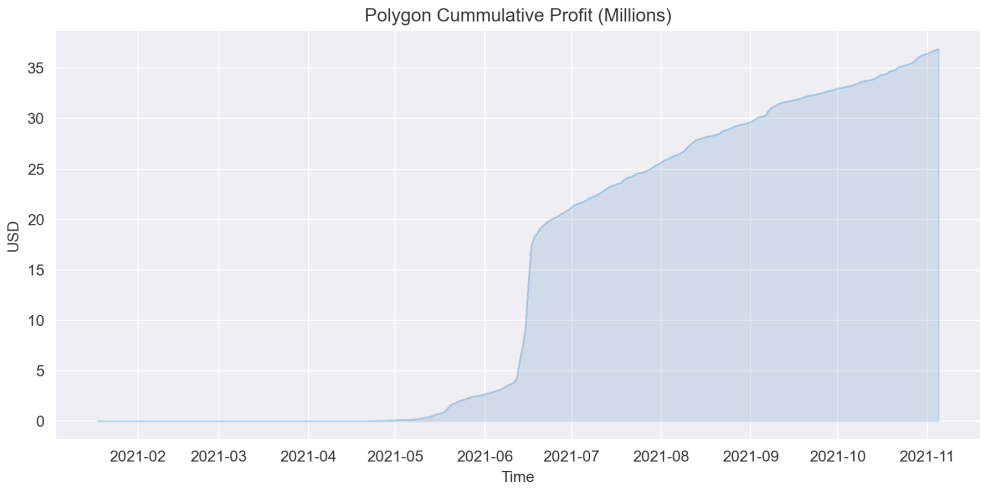
Optimism Geth tracing has been available since the 11 of November. We ran our implementation of mev-inspect over the entirety of the chain from that date (obtained from our own node). The results, at first, may be somewhat surprising. We only found four arbitrageurs and a meager total profit of $34K (almost exclusively using ETH).
Some potential explanation for such low MEV could be the immaturity of the network and the lack of volume and exchanges that come hand in hand with that. It is possible that our script captured a small percentage of the total arbitrage revenue. The strongest explanation, however, comes out when one digs a little deeper. Up until today (17 December '21), Optimism has reserved a whitelist of accounts allowed to launch smart contracts on the network. Since most arbitrage relies on an atomic group of swaps facilitated by a smart contract, traditional arbitrage was nigh impossible up until recently.
The question then becomes how, in the absence of the ability to create contracts for atomic arbitrage, any arbitrage took place. The answer lies in the contracts which were already launched by the whitelist. Uniswap for example, maintains a "router" contract which facilitates swaps between multiple pools. Traditional MEV searchers do not use the router because it is gas inefficient and conspicuous, rendering one vulnerable to frontrunning. All of the arbitrage found on optimism ran through such routers.
Since routers are specific to exchanges, no inter-exchange arbitrage was possible. This is traditionally where most of the opportunity lies so we anticipate a significant increase in arbitrage on Optimism in the weeks to come, as any EOA is now able to launch a smart contract.

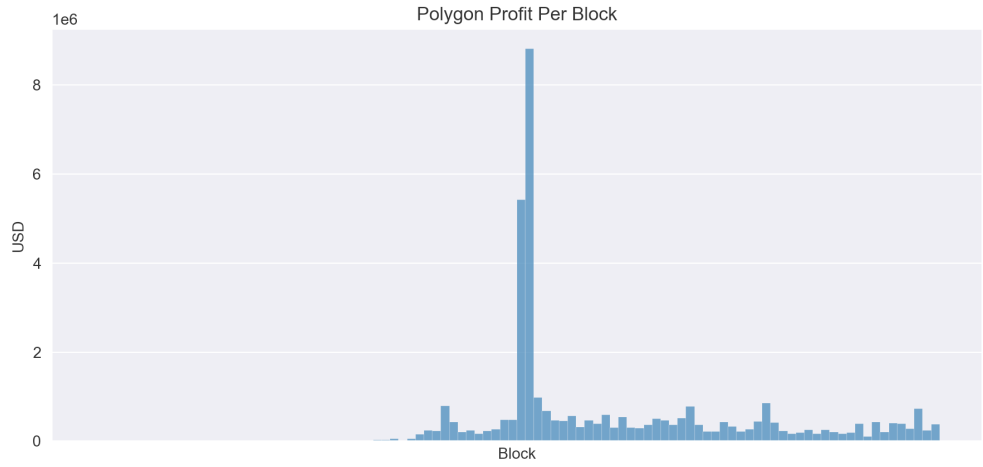




4.3 Polygon

Our analysis is based on work done by Marlin Protocol, who supplied us with a database of transactions identified as arbitrages along with the related increase in token. The main reason we sourced this data from another party is that, due to Polygon's block size and rate, simulating and tracing transactions can be executed at only slightly faster than the network speed, making this a very expensive process (note: event-based tracing significantly improves this process, but it still remains expensive). In order to complete the data supplied by the team from Marlin, we needed to collect data from several API providers to 1)


Polygon Cummulative Profit (Millions)
35
30
25
20
15
10
5
0
USD
2021-02
2021-03
2021-04
2021-05
2021-06
2021-07
2021-08
2021-09
2021-10
2021-11
Time


Polygon Profit Per Block
1e6
8
6
4
2
0
USD
Block

map token addresses to token symbols, 2) retrieve historical price data 3) map block numbers to block hashes and then block timestamps.

Interestingly, almost half of the 3.7 million total revenue was generated in a week, with 5.8 million coming in on a single day. After some digging, we found that this spike in arbitrage revenue correlates with the collapse of Titan Token. We hypothesise that the collapse of the token lead to a spike in price fluctuation, opening up the opportunity for arbitrage.

The largest arbitrage revenue on Polygon came in at $726K on the 16th of June. This is the same day in which 5.8 million profit was taken, the largest recorded day. The majority of arbitrages started and finished in MATIC (%63.7), followed by ETH (%20.6) and USDC (%15.3). It is unclear why these tokens are chosen over others. The attraction of USDC is clear as a stable asset. It may be that MATIC also has appeal for gas reasons as it is the native token on Polygon. Other factors may include volume and volatility of the respective tokens, but this remains an open question.

4.4 Comparison

Comparing the scale of arbitrage between Polygon and Optimism seems to be a somewhat uninformative venture. Even accounting for the longer time period over which arbitrage in Polygon was measured, arbitrage on Polygon is two orders of magnitude greater than Optimism.

Scale being out of the question, we consider the efficiency of the arbitrageurs:

- Revenue per arbitrage opportunity on Optimism is approximately $22.61
- Revenue per arbitrage opportunity on Polygon is approximately $4.09

this significant difference can be explained by the high ratio of failed ot succeeded arbitrage transactions on Polygon as explained in the background section. Since gas on Polygon is so cheap, a trial and error method with very little filtering for bad transactions leads to such a low return per transaction.

4.5 Conclusion

We set out to analyse MEV on L2 scaling solutions. In the process of doing so, we ran multiple versions of Optimism and Arbitrum's clients, spoke to multiple well-known founders, downloaded a very large fraction of the Polygon chain and accessed more API's than we would like to remember. The conclusions were not straightforward clean insights as one would hope for, but this is a feature of such a new field. In a sense, we feel we have been too early. Arbitrum's tracing functionality is set to come out in the next few weeks and arbitrage on Optimism has likely already picked up significantly in the last day since the whitelist has been removed.
What we have done is learned that MEV profits are concentrated in a handful of

tokens, that large events that cause extreme volatility like the collapse of Titan Token contribute a disproportionate amount to total arbitrage revenues and established a methodology through which future analysis of single-domain MEV can be conducted. We plan to continue our analysis, setting up dashboards, incorporating more forms of MEV and developing finer-grained insights as the space develops.

5 Future Work

We have ported the MEV Inspector tool to work with GETH traces, deployed it on Optimism, and analyzed results on Polygon (data courtesy of the Polygon guys) and Optimism. There are two main limitations. First, the tool only scrapes and classifies traces of a single blockchain, but we live in a multi-chain world. For as long as multiple self-contained domains exists, the ordering matters not only within chain but also between chains (Section 5.1). Second, only arbitrages and liquidations are currently quantified. As the community is reporting new forms of MEV which are long tail, more subtle, potentially application specific, theory is struggling to keep up. To better characterise existing MEV opportunities, predict novel ones, and design MEV mitigation strategies, we first need to agree on a formalization of MEV (Section 5.2).

5.1 Cross Domain MEV

As this report is being written, someone has successfully performed an arbitrage between Polygon and Ethereum, resulting in a profit of $16862.59 (Fig 3). It involved two chains, three pools, three tokens, and the profit token was the same one the trader started with but on a different chain. A quick browse through a https://westerngate.xyz/ shows that cross-domain MEV opportunities is already very much here, and will only become more abundant as the number of different domains as well as the volume/activity in each domain grows.

Cross domain MEV [23] exists due to the fact that different domains (e.g., L1s, L2s, side-chains, shards, CEXes) have different states, with exchanges whose token pricing is computed using the domain's states. The differing volume and activity levels on different domains containing pools of the same assets creates arbitrage opportunities.

Whether MEV is bad or not, whether we should aim to eliminate/mitigate it, and how much we can do so is still a topic of heated debate. However, most people can agree that if MEV is around, it is better for

1. MEV opportunities to have a low barrier of entry.
2. Engagement in MEV extraction and competition does not have negative externalities on the network.
3. MEV behaviour to be visible and accountable

The current state of affairs for cross domain MEV looks quite grim for 1) (Section 5.1.1), 2) (Section 5.1.2), and 3) (Section 5.1.3)

5.1.1 Higher Barrier of Entry

A single-domain arbitrage opportunity could be captured by a player with a small capital investment completely at a fee using flash loans or partially captured with their own capital. In contrast, extending flash loans to the multi-domain setting (i.e., lend on one domain, pay on another domain) is unclear. Further, players could risk failed execution due to non-atomicity of the trade and large cross domain delays.

Instead, the optimal MEV extractor would have a large reserve of the main tokens (e.g.: WETH, MATIC, USDC) on *multiple* domains. As soon as an arbitrage opportunity presents itself, the agent could directly execute the arbitrage path on their respective domains, bypassing the cost and delay of cross-domain communication completely. This clearly has a lower operation cost than relying on technologies like bridges along with a higher chance of success, which is why platforms offering cross-domain arbitrage paths suggest this strategy[1]. In short, cross domain MEV will have a higher barrier of entry.

5.1.2 Cross Domain Collusion/Sequencer Centralization

If cross domain arbitrage continues to be a highly profitable endeavor with lower competition due to its high barrier of entry, sequencers from different domains may become incentivized to collude with each other. While there are costs to collusion (e.g., set up trust, social norms against colluding sequencers, etc.), profits from cross domain arbitrage could outweigh these costs. Similarly, large potential payoffs could incentivize cross domain sequencer centralization. Controlled by a single entity, the centralized cross domain sequencer could experience zero collusion costs and maintain significant influence on the transaction ordering on multiple domains at once by a capturing large arbitrage profits with much higher success rates than competing single-domain/colluding multi-domain MEV extractors.

Currently, many domains with large volumes are controlled by a few (occasionally centralized) sequencers, with only future plans of decentralizing sequencing. With arbitrage profits in the order of magnitude of Western Gate leaderboard, large for-profit sequencers may already be incentivized to collude. Cross domain arbitrages are here to stay, so if cross domain MEV is inevitable, an important future direction is to see how we can make efficient and fair sequencer collusion trustless in order to avoid sequencer centralization.

5.1.3 Quantifying Cross-Domain MEV

MEV Inspector quantifies a lower bound on MEV on each chain by looking at block traces and counting the profits when the traces matches the defined liquidation or arbitrage models. This inherently limits the amount of MEV discoverable and quantifiable by the models defined. For instance, if some MEV

[1] See https://westerngate.xyz/ → About → Q: How do I profit from this?

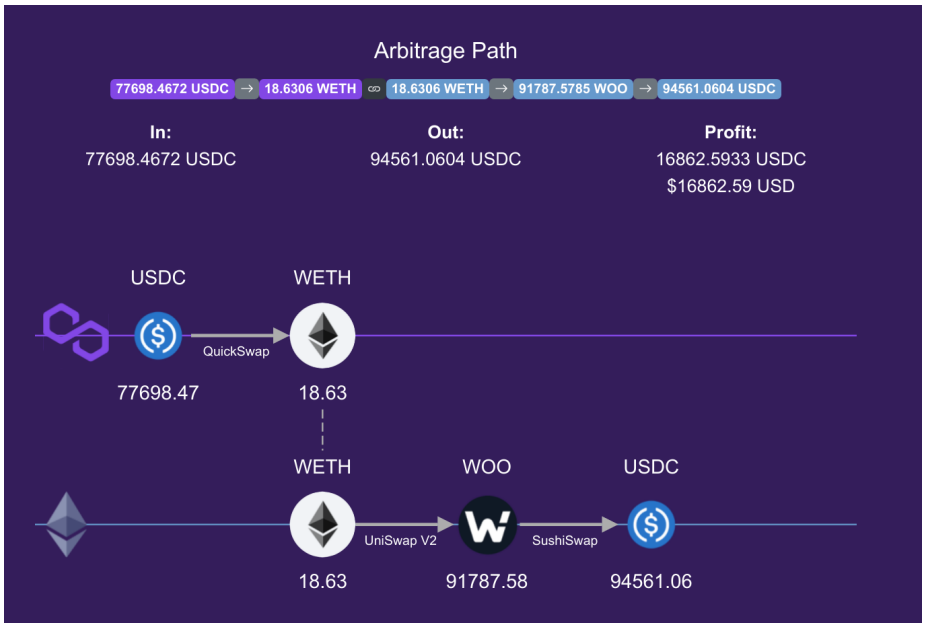


Figure 3: **Arbitrage between Polygon and Ethereum.**

extraction occurs over the span of multiple transactions or blocks, this will not be included in the final value we report.

Even if we kept expanding our set of models, there are still cases which can never be quantified, such as when one entity owns multiple wallets. Without making additional assumptions on how players use different wallets with different public keys, it is impossible (as well as computationally infeasible) to cluster public keys into players and track MEV based on players instead of public keys. Different public keys on different domains is exactly why cross domain MEV may also be more opaque than single-domain MEV.

5.2 Formalizing MEV

To have rigorous theory for MEV, we need to agree on a unifying definition for MEV. Candidates for definitions of MEV seem to emphasize the following properties:

1. **Permissionless**: MEV should be defined in terms of the least priviledge actor inside the system. This captures the notion that these opportunities can be extracted by anyone in the system, independent of their reputation.

2. **Investment Capital Dependent**: MEV should also be separated into different tiers depending on how much capital a player starts off with, where the difference in MEV between these tiers could give us a measure of how high the barrier of entry is. In its simplest form, arbitrages require

traders to provide some form of economic incentive to the miners (e.g., gas fees, bundle payments, etc.), and this requires upfront capital. Another level up in terms of costs as well as (potential) profits could be by running a mining node or locking up stake in order to influence transaction orders.

3. **Beyond Single-Block**: More subtle types of MEV could occur over the span of multiple blocks and/or domains.

4. **Non-miner MEV**: A blockchain ecosystem contains many different actor roles, each with their own action space, which can be used to influence the final block ordering (directly as a miner or indirectly otherwise) to extract MEV.

Starting from a reasonable definition proposal [24], we can attempt to iterate on it to achieve more properties which we want from the list above, similar to [25]. Within a chain, the extractable value (EV) in the primary token of the chain expresses the maximum value that can be extracted by a player p (with block proposing rights) from a valid sequence of blocks $(B_1, \ldots, B_k)$ of length k could be defined as

$$EV(p, \mathcal{B}, s) = \max_{(B_1,\ldots,B_k)\in\mathcal{B}} b\big(p, s(B_1, \ldots, B_k)\big) - b(p, s) \tag{1}$$

where $\mathcal{B}$ is a set of sequences of blocks, all of length k, s and $s(B_1, \ldots, B_k)$ are the state of the system before and after applying the k block sequence respectively, and $b(p, s)$ denotes the balance of p in the primary token in state s. Babel et al. then defined MEV of player p to be p's EV when $\mathcal{B}$ is the set of all valid k sequence of blocks p can propose in state s

$$k - MEV(p, s) = EV(p, validBlocks_k(p, s), s) \tag{2}$$

Note that Eq. 2 does not achieve all of the properties listed above. First, it depends on p, and therefore, p's reputation. Instead, to get a permissionless definition of MEV which does not depend on the player, we can take the minimum MEV over the set of all possible players to get the value extractable by the least priviledged player. Second, this definition lumps together MEV extractable by any player regardless of their initial balance. Third, while the definition accounts for multi-block MEV extraction, it does not account for multi-domain MEV, or actions other than block proposal for MEV extraction. Finally, the action space here is clearly in the block ordering, and doesn't account for MEV extractable by, say, traders.

Drawing from Obadia et al. and Salles, we can patch this definition up to achieve the properties we want. To achieve the permissionless property, the MEV should be a quantity independent of the player, which we can achieve by taking the minimum MEV over all players. Instead, MEV should be dependent on the investment capital of this least priviledged player to achieve the second property. To go beyond miner MEV, we can use a generalized notion of action spaces instead of just block sequences. Going beyond single block, we would

incorporate the sum of extractable values for some player over all domains as a function of the actions they can take on all of these different domains together. Putting it all together, a patched definition of MEV in some state s for any player with a balance of at least K, a combined action space $A(p) = A_1 \cup \ldots A_n$, and combined balance $B(p) = B_1 \cup \ldots B_n$ over domains $i \in [n]$ is

$$MEV(s; K) = \min_{\{p \in \mathcal{P} | \sum_{b \in B(p)} b(p,s) \geq K\}} \max_{a_1, \ldots, a_n \in A(p)} \Big\{ \sum_{b \in B(p)} ev_b(p, s, a_1, \ldots, a_n) \Big\} \tag{3}$$

This definition seems to align well with our current understanding of MEV. For instance, a larger combined action space A can only increase MEV. This makes sense in light of our discussion about sequencer centralization (Sec. 5.1.2). We can also understand fair consensus protocols or proposer-builder separation as mechanisms for reducing the action space of certain players. We can also articulate what is the ideal case for MEV if we can't completely prevent it. For example, we might want MEV to be close to a linear function of K, implying low barrier of entry (since players with any amount of initial capital can extract an amount of profit proportional to their investment).

References

[1] Philip Daian et al. "Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges". In: *CoRR* abs/1904.05234 (2019). arXiv: 1904.05234. URL: http://arxiv.org/abs/1904.05234.

[2] *MINER EXTRACTABLE VALUE (MEV)*. 2021. URL: https://ethereum.org/en/developers/docs/mev/ (visited on 12/15/2021).

[3] *Uniswap V2 Overview*. 2020. URL: https://uniswap.org/blog/uniswap-v2 (visited on 12/14/2021).

[4] Ye Wang et al. "Cyclic Arbitrage in Decentralized Exchange Markets". In: *Available at SSRN 3834535* (2021). arXiv: 2105.02784. URL: https://arxiv.org/abs/2105.02784.

[5] Zhou Liyi et al. "High-Frequency Trading on Decentralized On-Chain Exchanges". In: *2021 IEEE Symposium on Security and Privacy (SP). IEEE* (2020). arXiv: 2009.14021. URL: https://arxiv.org/abs/2009.14021.

[6] Carlsten Miles et al. "On the Instability of Bitcoin Without the Block Reward". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 154–167. ISBN: 9781450341394. DOI: 10.1145/2976749.2978408. URL: https://doi.org/10.1145/2976749.2978408.

[7] *welcome to flashbots*. URL: https://docs.flashbots.net/ (visited on 12/15/2021).

[8] Eskandari Shayan, Moosavi Mahsa, and Clark Jeremy. "SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain". In: *FC 2019: Financial Cryptography and Data Security, Available at SSRN* (2019). URL: https://ssrn.com/abstract=3369236.

[9] Karl Floersch. *MEV Auction: Auctioning transaction ordering rights as a solution to Miner Extractable Value*. 2020. URL: https://ethresear.ch/t/mev-auction-auctioning-transaction-ordering-rights-as-a-solution-to-miner-extractable-value/6788 (visited on 12/14/2021).

[10] Ari Juels, Lorenz Breidenbach, and Florian Tramèr. *Fair Sequencing Services: Enabling a Provably Fair DeFi Ecosystem*. 2020. URL: https://blog.chain.link/chainlink-fair-sequencing-services-enabling-a-provably-fair-defi-ecosystem/ (visited on 12/14/2021).

[11] *Creating a Highly Scalable and MEV-Resistant DeFi Ecosystem Using Arbitrum and Fair Sequencing Services*. 2021. URL: https://blog.chain.link/arbitrum-and-chainlink-fair-sequencing-services/ (visited on 12/14/2021).

[12] *An Incomplete Guide to Rollups*. 2021. URL: https://vitalik.ca/general/2021/01/05/rollup.html (visited on 12/15/2021).

[13] *(Almost) Everything you need to know about Optimistic Rollup*. 2021. URL: https://research.paradigm.xyz/rollups (visited on 12/15/2021).

[14] *LAYER 2 ROLLUPS*. 2021. URL: https://ethereum.org/en/developers/docs/scaling/layer-2-rollups/ (visited on 12/15/2021).

[15] *Optimistic Sequencing*. URL: https://community.optimism.io/docs/protocol/sequencing.html (visited on 12/15/2021).

[16] *OPTIMISTIC ETHEREUM*. URL: hhttps://www.optimism.io/ (visited on 12/15/2021).

[17] *Optimism API*. URL: https://docs.alchemy.com/alchemy/apis/optimism-api (visited on 12/15/2021).

[18] *ARBITRUM*. URL: https://arbitrum.io/ (visited on 12/15/2021).

[19] *polygon - Ethereum's Internet of Blockchains*. URL: https://polygon.technology/ (visited on 12/15/2021).

[20] Robert Miller. *MEV negative externalities*. URL: https://twitter.com/bertcmiller/status/1412579402345586696 (visited on 12/15/2021).

[21] *Tweet - Client Incentive Program*. 2021. URL: https://twitter.com/AyaMiyagotchi/status/1470654120180649986 (visited on 12/15/2021).

[22] *Ethereum Nodes Data*. 2021. URL: https://www.ethernodes.org/ (visited on 12/16/2021).

[23] Alexandre Obadia et al. *Unity is Strength: A Formalization of Cross-Domain Maximal Extractable Value*. 2021. arXiv: 2112.01472 [cs.CR].

[24] Kushal Babel et al. "Clockwork Finance: Automated Analysis of Economic Security in Smart Contracts". In: *CoRR* abs/2109.04347 (2021). arXiv: 2109.04347. URL: https://arxiv.org/abs/2109.04347.

[25] Alejo Salles. *On the Formalization of MEV*. 2021. URL: https://writings.flashbots.net/research/formalization-mev/#current-mev-definitions (visited on 12/13/2021).