Blockchain Transaction Conflicts: A Historical Perspective

Supra Research

24 February 2025

Abstract

This paper presents a comprehensive analysis of historical data across two popular blockchain networks: Ethereum and Solana. Our study focuses on two key aspects: transaction conflicts and the maximum theoretical parallelism within historical blocks. By systematically examining block-level characteristics—both within individual blocks and across different historical periods—we aim to quantify the degree of transaction parallelism and assess how effectively it can be exploited.

Our findings contribute to a deeper understanding of blockchain scalability, efficiency, and its potential for optimizing transaction processing. In particular, this study is the first of its kind to leverage historical transactional workloads to evaluate the patterns of transactional conflicts in blockchain networks. By offering a structured approach to analyzing these conflicts, our research provides valuable insights and an empirical basis for developing more efficient parallel execution models across diverse blockchain ecosystems.

1 Introduction

Smart contract virtual machines (VMs) execute a *block* of user-defined *transactions*, each performing a sequence of *read* and *write* operations on the states of user accounts. A *block proposer*, a validator node in the network, takes as input a *block* consisting of n transactions and a *preset* total order $T_1 \to T_2 \ldots, \to T_n$. The node attempts to execute, in parallel, the n transactions such that the state resulting from the parallel execution of the n transactions must be the state resulting from the *sequential* execution of transactions $T_1 \cdot T_2 \cdots T_n$. The key requirement to ensure consistency across the network is that all nodes must execute transactions in the preset order.

Smart contract VMs utilize different execution strategies that directly affect transaction throughput. Ethereum [4] and Solana [13] represent two distinct architectures, each employing a different transaction execution approach. Ethereum processes transactions sequentially without prior knowledge of the states that they access. In contrast, Solana enables parallel execution by requiring clients to specify access information for each transaction, improving throughput. When block transactions are executed, they may modify the same account address or storage slot in a smart contract, potentially resulting in conflicts if one transaction accesses data modified by another.

A *conflict* occurs when two or more transactions access the same state and at least one of them performs an update operation. The transactions in the block must be executed sequentially in preset order, the order of the transactions in the block, or they must be serializable to the preset order if executed in parallel, to provide deterministic execution across the distributed network. The order in which conflicting transactions are executed directly affects the final state and must be executed in order, with one waiting for the other to finish. However, *independent transactions* can be executed in parallel or in any order, as their execution does not interfere with other transactions. Transactions are independent if they do not modify (write to) the same state and the outcome of one does not impact the other.

The *longest chain of conflicting transactions* is one of the parameters that determines the maximum theoretical limit for parallelization within a given block, including the number of *conflicting transactions* (resp. independent transactions), number of *conflicts* and different conflict sets (*conflict families*). A conflict family is a group of transactions that are mutually dependent on shared states. Identifying conflict families is essential for optimizing transaction execution, as it helps in detecting hotspots and employing the best suitable parallel execution strategy.

Transaction conflicts can be detected through smart contract bytecode analysis, read/write set analysis, and optimistic execution at runtime. Efficient conflict identification and resolution with minimal overhead, including abort and re-execution costs, are crucial for enhancing performance. The timing and cost of conflict detection and resolution significantly impacts parallel execution efficiency. However, the distribution of conflicts and state access patterns within historical blocks is the main focus of this study, along with the metrics we discussed above that directly or indirectly affect the parallel execution of transactions and overall throughput. These insights support the development of techniques that improve execution efficiency and enhance parallelization across blockchain networks. Previous studies [8, 11] analyzing historical Ethereum blocks have demonstrated promising potential for parallel execution. In this paper, we analyze two blockchain networks with different transaction execution models to better understand their conflict distributions and impact on performance.

- **Ethereum [4]**: Ethereum employs an account-based model with sequential execution through the Ethereum Virtual Machine (EVM). The state accessed by a transaction is determined only at the time of block execution. We refer to this as the *read-write oblivious* execution model, where read-write sets are not known a priori.
- **Solana [13]**: Implemented Sealevel [17] for parallel execution, enabling transactions to run in parallel within the Solana Virtual Machine (SVM). The state accessed by transaction is known in advance as read/write sets. We call it the *read-write aware* execution model.

This paper presents a detailed study of conflict relationships in these two most popular smart contract execution paradigms. Our findings offer insights into the *ground truth* for the maximum parallelism that can be extracted in historical block transactions, constrained by inherent conflicts within the blocks.

2 Extraction of Conflict Specifications

This section describes our data extraction and conflict analysis. The fundamental definition of a conflict remains unchanged; however, the granularity of available conflict information varies from one blockchain to another. For instance, on Ethereum, we get access to historical block data without distinguishing between reads/writes, whereas Solana provides more detailed transactional data with reads and writes.

Ethereum

For our analysis, we segregate Ethereum transactions into two types: ETH transfer transactions and smart contract transactions, to analyze and understand the access and conflict patterns. The ETH transfer transactions perform pure value transfers between externally owned addresses (EOAs) or to smart contract addresses. On the other hand, the smart contract transactions interact with sender addresses and contract address(es) to modify blockchain state via function calls and storage updates within the contract(s).

We trace the accessed states of all transactions within a block using the callTracer and prestateTracer [2], which provides a full view of the block's pre-state, the state required for the execution of current block. We identify all EOAs, contract addresses, and storage locations within contracts that are accessed by examining transaction data and the pre-state for it. This enables the detection of potential conflicts arising from overlapping state modifications by block transactions.

Ethereum Transaction Conflict: Two transactions T_i and T_j are in conflict if

- T_i and T_j both access a common EOA address.
- T_i and T_j both access a common storage location within a contract address.

It should be noted that two transactions are considered independent (non-conflicting) if they are initiated by separate EOAs and access different storage locations within all the contracts that they access. However, there is a special case that every transaction updates the coinbase account (block proposers account) for fee payment. As a result, all transactions are logically in conflict, unless the coinbase account is treated as an exception. For this reason, when we analyze conflicts, we remove the coinbase account from the transactions access set. In [10], a solution is proposed to collect the fee payment for each transaction independently and update the coinbase account cautiously at the end of the block, allowing the transactions to be executed in parallel.

We use an *exclusive-access* paradigm for our conflict analysis of Ethereum blocks. Since read set information is not explicitly provided with prestateTracer [2] and transactions accessList [1], we treat every operation as an exclusive update operation, which may result in overestimating conflicts. For example, if T_i and T_j access the same state, they will conflict in the current analysis; however, there could be no conflict if they both just read the state in practice. Therefore, a more detailed analysis that separates accesses into read and write operations would likely reduce the conflict numbers presented in the next section for Ethereum.

Solana

In contrast to Ethereum, Solana transactions are made up of the account access specification, a list of accounts to read from or write to [13], known as the read set and write set, respectively. This specification is added to the transactions upfront by the clients through RPC node interaction. The success or failure of a transaction depends on the freshness of its read-write set, from the moment it is added to the transaction by the client until its execution at the validator node. The read and write access set specification simplifies our analysis and improves the accuracy of conflict detection for Solana blocks.

We used the beta API of Solana's mainnet to obtain block details in JSON format with the max supported transaction version set to 0 [14]. The extracted details are then parsed to obtain the information required for our analysis.

Solana Transaction Conflict: Two transactions T_i and T_j are in conflict if both of them access a common account and at least one of them is a write.

3 Conflict Analysis

This section will present our findings from the historical analysis of Ethereum (in Section 3.1) and Solana blocks (in Section 3.2) over different time periods, focusing on the following key aspects:

- **Historical periods (HP)**: Different time periods exhibit varying transaction loads. We analyze peak and low-traffic periods to understand how congestion affects execution efficiency.

Table 1 Historical Blocks from Ethereum's mainnet: 1000 blocks from each historical period, where analysis is done based on exclusive access to accounts by transactions.

	CryptoKitties Deployment (E_{ck})	Ethereum 2.0 Merge (E_{e2})	Ethereum Recent Blocks (E_{rb})
Block ID of Historical Event	4605167	15537393	21631500
Block Range Before Event	4604664 - 4605166	15536879 - 15537392	21631000 - 21631500
Block Range After Event	4605168 - 4605670	15537394 - 15537907	21631501 - 21632001
Average Number	**before event - after event**	**before event - after event**	**before event - after event**
Average Block Size	71 - 83	178 - 156	181 - 176
ETH Transfer Txs	35 - 42	66 - 42	65 - 62
Smart Contract (SC) Txs	37 - 41	113 - 114	116 - 114
ERC20 Transfer Txs	12 - 13	11 - 16	43 - 38
Independent Txs (%)	38 (61.87%) - 42 (55.70%)	87 (54.18%) - 55 (39.45%)	92 (51.73%) - 90 (51.82%)
Chain of Conflicting Txs	15 - 15	38 - 42	31 - 27
Independent ETH Transfer Txs (%)	21 (70.05%) - 24 (63.82%)	32 (62.57%) - 24 (67.72%)	49 (78.29%) - 47 (78.31%)
Independent SC Txs (%)	17 (57.20%) - 19 (52.96%)	56 (56.95%) - 32 (32.52%)	45 (39.82%) - 45 (40.32%)
Conflict Families	30 - 37	56 - 40	72 - 69

- **Access (read/write)-level conflicts**: Identify conflicting transactions based on their access levels or detect potential conflicts at a more granular level (reads/writes).
  - *Independent transactions (%)* that do not interfere with others and can be executed in parallel.
  - *Longest chain of conflicting transactions* in the block.
  - *Conflict families*, a family is a group of transactions that are mutually dependent on the shared state.
  - The *most dense conflict family*, a conflict family with most transactions.
  - *Total conflicts* and *write-write conflicts* between transactions.

3.1 Ethereum

As shown in Table 1, we selected three distinct historical periods (HPs) based on the major events that impact Ethereum's network congestion. Each of these periods allows us to assess the blocks in different HPs, giving insights into how major events like popular dApp launches and significant protocol upgrades affect transaction parallelism and conflicts. It also helps us understand the limitations of parallel execution approaches under different network conditions and historical periods.

- **Ethereum CryptoKitties Contract Deployment (E_{ck}):** CryptoKitties [3] game is among the first and the most popular dApps. The CryptoKitties was deployed in block 4605167, after which an unexpected spike in transactions caused Ethereum to experience never-before-seen congestion. The workload consists of 500 blocks each from before and after the deployment of the CryptoKitties smart contract. We can expect this period to receive a high volume of transactions for a contract, consequently leading to congestion at a specific contract, as observed by an earlier study in [11]. Hence, this period is an ideal workload for determining how well the parallel execution approach performs with a large influx of transactions for a contract.
- **Ethereum 2.0 Merge (E_{e2}):** Ethereum's transition from proof-of-work to proof-of-stake consensus took place at block 15537393, called the Ethereum 2.0 merge [5]. This event has changed Ethereum's consensus mechanism and could have optimized the transaction processing, block validation, and network traffic in general. So in this workload, we try to determine the direct impact of this upgrade on the parallel execution pattern that impacts the transaction throughput and network behavior by comparing blocks before and after the merge.

- **Ethereum Recent Blocks (E_{rb}):** In addition to the above historical periods, we analyze transactions from the 1000 most recent blocks, ranging from block number 21631000 to 21632001. We selected this range to better understand current transaction access patterns and parallelism in normal conditions when there are no major historical events impacting the network traffic. By examining this workload, we can establish connections between different historical periods—tracking Ethereum's evolution, user access patterns, network congestion over time, and the impact of optimizations on recent blocks. This analysis also helps in developing the approaches that exploit parallelism efficiently for future network upgrades.

Observation-1: The initial evaluation aims to understand the parallelism by distinguishing between dependent (conflicting) and independent (non-conflicting) transactions, identifying the longest chain of conflicting transactions, and examining conflict families both within and across historical periods.

As shown in Table 1, transactions per block have increased since E_{ck} HP, with contract transactions rising ∼4× and ETH transfers ∼2×. This implies an increased demand for computational resources and an increased adoption of blockchain technology for broad smart contract applications (dApps).

The percentage of independent transactions per block has decreased, particularly post-Ethereum merge, though over 50% remain independent on average. The longest conflict chain comprises ∼19–20% of the block size, peaking at ∼22% post-merge and stabilizing at ∼16–17% in recent blocks. This suggests that even with perfect parallelization of other transactions, including scheduling of conflicting transactions, speedup is limited to ∼16-17% of transactions, the theoretical upper bound on speedup over sequential execution in recent blocks.

Compared to previous HPs, the percentage of independent ETH transfer transactions in recent blocks has increased, whereas the number of independent smart contract transactions has decreased, indicating an upsurge of transactions for specific contracts and diverse user transactions for ETH transfer. However, the rise in conflict families and block sizes from the earlier period to the more recent one suggests that there is a lot of parallelism. This can make Ethereum's throughput better if parallel transaction execution is employed.

Observation-2: Calculating the ratio of ETH transfers to smart contract transactions in Table 1, comprising the historical period from E_{ck} to E_{e2} and E_{rb}, shows an increased user engagement with contract applications. In E_{ck} HP, the ratio of $\frac{ETH\ transfer}{SC\ Txs}$ is $\frac{38.5}{39} = 0.99$, while it is ~ 0.47 in E_{e2} and ~ 0.55 in E_{rb}. This indicates a surge in computational costs over time and the need for parallel transaction execution to improve network throughput.

Observation-3: To understand how many blocks in each HP have a certain percentage (>40%, >50%, ... >80%) of independent transactions and which HP has a higher parallelism compared to others. As shown in Figure 1, we analyzed 1000 blocks of each HP (500 before and 500 after the event).

The number of blocks with more than 40% independent transactions has increased in recent blocks, more than 94% of blocks have at least 40% independent transactions in E_{rb}. However, there was a notable decline in independent transactions after each historical event, suggesting a spike in conflicts. Note that more than 50% of the blocks in each HP had more than 50% of independent transactions, while post Ethereum merge E_{e2} there is a significant drop. The reasons could be increased congestion for a specific contract (the longest conflict chain increased), a decrease in the number of ETH transfer transactions, and a slight decrease in block size compared to pre-merge, as observed in Table 1. The number of blocks with a higher percentage of independent transactions could increase if false conflicts are removed

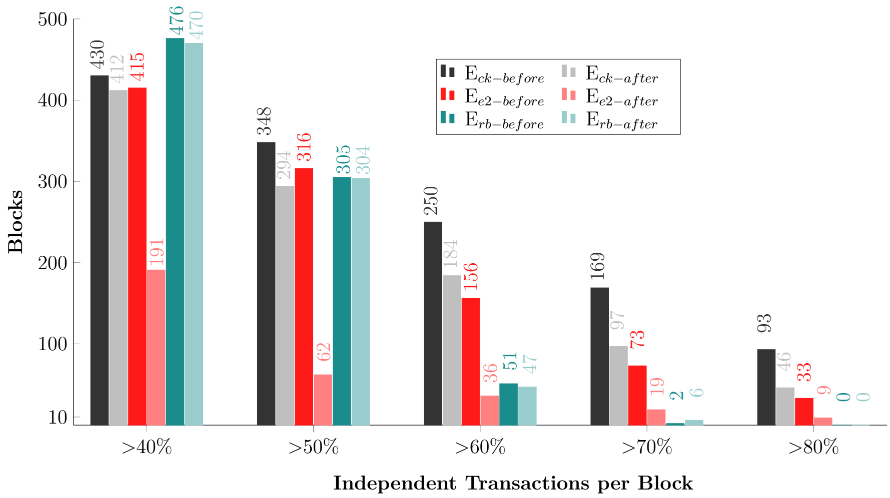

Figure 1 Ethereum: number of blocks where the percentage of independent transactions exceeds the threshold before and after historic event.

Table 2 Ethereum Recent Historical Period (21631001–21631020): analysis based on exclusive access to accounts by transactions.

Block ID	Block Size	ETH Transfer Txs	SC Txs	ERC20 Transfer Txs	Independent Txns (%)	Chain of Conflicts	Independent ETH Transfer Txs	Independent SC Txs	Conflict Families
21631001	337	130	207	75	146 (43.32%)	56	82	66	116
21631002	148	39	109	46	69 (46.62%)	28	34	38	51
21631003	82	27	55	35	47 (57.32%)	24	23	27	38
21631004	191	65	126	59	92 (48.17%)	50	53	42	73
21631005	233	75	158	55	125 (53.65%)	52	69	57	99
21631006	154	64	90	34	86 (55.84%)	20	47	40	74
21631007	192	67	125	46	101 (52.60%)	28	57	46	75
21631008	163	60	103	47	78 (47.85%)	27	46	34	67
21631009	177	68	109	55	81 (45.76%)	39	53	31	70
21631010	207	86	121	52	81 (39.13%)	52	37	45	68
21631011	148	46	102	31	78 (52.70%)	20	41	39	61
21631012	134	46	88	32	83 (61.94%)	16	43	40	66
21631013	175	51	124	58	93 (53.14%)	34	49	45	72
21631014	200	66	134	41	109 (54.50%)	21	54	55	91
21631015	138	42	96	40	82 (59.42%)	22	40	42	68
21631016	180	68	112	58	76 (42.22%)	34	43	33	61
21631017	119	38	81	43	70 (58.82%)	26	34	38	61
21631018	230	100	130	46	113 (49.13%)	41	62	52	94
21631019	145	47	98	30	84 (57.93%)	19	36	48	67
21631020	166	55	111	39	84 (50.60%)	26	45	41	68
Average	**170**	**60**	**110**	**45**	**86 (51.70%)**	**31**	**46**	**42**	**72**

using complete read-write access information.

Observation-4: Table 2 highlights block-wise trends of the recent historical period (E_{rb} HP) where we selected 20 blocks. As shown, most blocks have more than 50% independent transactions, with the highest parallelism in block 21631012, contains 134 transactions out of which 61.94% are independent. The conflict chain is the shortest in this block, with 16 transactions (11.94% of the block size). In particular, most conflicts are from contract transactions; out of 88 contract transactions, 48 are conflicting, while only 3 are conflicting from ETH transfer transactions. Block 21631010, on the other hand, has the least parallelism, with 210 transactions, only 39.13% of which are independent, and the longest conflict chain involving 25.12% of the block. These blocks have an average of 170 transactions, of which 51.70% are independent. The longest conflict chain takes up 18.23% of the block size.

Table 3 Historical Blocks from Solana's mainnet: 1000 blocks from each historical period and analysis based on read-write sets of non-voting transactions.

	Old Historical Period (S_{ob})	Mid Historical Period (S_{mb})	Recent Historical Period (S_{rb})
Block Range / Average	61039000 - 61040210	205465000 - 205466007	293971000 - 293972009
Block Size (including Voting Txs)	519	1972	1249
Non-Voting (NV) Txs	252	113	334
Successful NV Txs	215	78	182
Conflicting NV Txs (%)	252 (100%)	101 (87%)	310 (93%)
Independent NV Txs (%)	0 (0%)	12 (13%)	23 (7%)
Chain of Conflicting NV Txs	214	52	119
Conflict Families of NV Txs	3	19	39
Dance Conflict Family of NV Txs	233	60	184
Total Conflicts of NV Txs	3664	886	3917
W-W Conflicts of NV Txs	3664	676	3751

In Ethereum's historical blocks, over 50% of the blocks in each HP contained more than 50% independent transactions. The theoretical upper bound on maximum speedup is constrained by the longest conflicting chain, which accounts for approximately 16–17% of block transactions. The change in independent transaction percentages over time and block by block, longest conflict chains, and conflict families indicates that no single parallel execution strategy is optimal for all blocks. Moreover, historical periods show significant shifts in conflict patterns, with smart contract transactions being the primary source of contention. Our observations highlight that we need an adaptive scheduling technique that dynamically chooses the best possible execution strategy and also optimizes overall execution based on real-time block characteristics to maximize throughput and efficiency. Alternatively, a hybrid parallel execution model that leverages conflict information available with transactions with minimum added overhead can maximize the performance of speculative parallel execution and minimize aborts and re-execution overhead.

3.2 Solana

Solana was the first read-write aware blockchain to support parallel execution. Transactions include the specification of state components that are read or written during execution. Solana Sealevel [17] execution engine makes parallel execution feasible by using locking-based techniques (read and write locks) to determine which transactions could be executed in parallel in a number of iterations [16]. The longest chain of conflicting transactions determines the minimum number of iterations required for a block, given that a sufficient number of cores are available to fully exploit the parallelism.

To understand the distribution of conflicts in historical Solana blocks, we analyzed 1000 non-empty blocks from three distinct periods: the old historical period (S_{ob}) from block number 61039000 to 61040210, the mid historical period (S_{mb}) from block 205465000 to 205466007, and the recent historical period (S_{rb}) from block 293971000 to 293972009. The Solana block consists of voting and non-voting transactions; we consider non-voting transactions for our analysis.

Analysis

Observation-1: As shown in Table 3, the average block size has increased more than 2× from old HP to recent HP; however, note that this increase is contributed by voting transactions. Non-voting transactions have increased with miner margin, which has seen a deep mid-historical period with increased voting transactions.

Observation-2: The percentage of successful non-voting transactions has decreased over time, with the success rate of ∼85.32% in the old historical period to ∼69.03% in the mid-historical period and to ∼54.50% in the most recent historical period. Shows that increasing network congestion must have contributed to transaction failures, potentially due to inaccuracies in transaction specifications; the time when specifications are generated by users to the time when executed may differ due to intermediate ongoing execution at the validator nodes. The exact reasons for the increased transaction failures require further analysis; however, it could be due to increased network congestion or inaccuracies in transaction specifications. However, it indicates the limitations and efficiency of read-write aware execution models in high-contention periods.

Observation-3: The percentage of independent transactions in Solana blocks is considerably lower than in Ethereum blocks. However, there is a noticeable upward trend, with independent transactions increasing from 0% in the old historical period (Sob) to ∼7% in the recent historical period (Srb), while the mid historical period (S_{mb}) recorded ∼13%. This suggests a gradual shift toward greater parallelism over time.

Since Solana employs a locking-based multi-iteration parallel execution strategy, the number of conflict families has surged, from just 3 in Sob to 39 in Srb, suggesting that despite high conflicts, multiple independent subsets of transactions can still be executed in parallel. Each subset was executed in parallel with others, enhancing execution efficiency.

The longest chain of conflicting transactions, relative to the total number of non-voting transactions in a block, has seen a substantial decline. Specifically, the longest conflict chain has reduced by $\sim 2.3\times$. The longest chain of conflicting transactions decreased from 84.92% in Sob HP to 46.01% in Smb HP and further to 35.62% in the most recent S_{rb} HP. The number of transactions within the most densely conflicted family has also seen downward trends. It suggests more distributed conflicts and the possibility of improved parallel execution with more granular bottlenecks in transaction execution. Showing that transaction access patterns have changed over time consequently improved throughput of Solana's read-write aware execution model.

Observation-4: Note that the majority of conflicts are from write sets in historical blocks, accounting for nearly 100% in the old historical period, which decreased by ∼4.24% (∼95.76%) in recent blocks. This suggests that any approach that could minimize write-write conflicts could significantly enhance Solana's throughput. A potential solution could be to adopt a multi-version data structure, similar to the one employed in Block-STM [6], which allows parallel execution while minimizing write-write contention.

Observation-5: Table 4 highlights recent block-by-block analysis in the recent historical period (S_{rb} HP). As shown, the size of the block (transactions per block) varies with significant margin. While the number of non-voting transactions remains constant, ranging from ∼200 to ∼550 per block. Indicating increased voting activity in the network with more participating validator nodes. On the other hand, the independent transaction percentage varies from a minimum of ∼3.27% in block 293971006 to a maximum of ∼25.82% in block 293971014, with an average of ∼9.27%, which is considerably lower compared to Ethereum blocks. Additionally, in all these blocks, the majority of conflicts originate from write sets. The longest conflict chain consists of 109 transactions, accounting for ∼30.96% of the non-voting transactions in the block, further emphasizing the possibility of write-write conflict optimization for efficient parallel execution.

From our observations, we can conclude that independent transactions remain very low

Table 4 Solana Recent Historical Period (293971001–293971020): analysis based on read-write sets of transactions.

Block ID	Block Size	Non-Voting (NV) Txs	Successful NV-Txs	Independent NV-Txs (%)	Chain of Conflicts	Conflict Families	Dance Conflict Family	Total Conflicts	W-W Conflicts
293971001	1179	332	167	31 (9.34%)	81	56	119	1717	1713
293971002	964	324	196	40 (12.35%)	87	61	138	1496	1488
293971003	1098	351	143	25 (7.12%)	129	47	159	2369	2367
293971004	1714	324	266	37 (11.42%)	67	57	147	3119	3096
293971005	1259	413	239	17 (4.12%)	188	28	362	3498	3386
293971006	1656	489	350	16 (3.27%)	242	27	367	15412	14717
293971007	675	392	147	17 (4.34%)	68	32	133	5366	5309
293971008	1830	308	200	52 (16.88%)	109	75	157	1489	1474
293971009	1528	298	165	21 (7.05%)	49	42	86	1984	1984
293971010	1438	401	331	14 (3.49%)	150	28	214	9111	9100
293971011	1129	431	250	20 (4.64%)	166	33	369	7892	7455
293971012	1692	375	222	33 (8.80%)	72	64	97	1496	1484
293971013	1556	218	91	44 (20.18%)	53	55	56	1572	1572
293971014	1458	213	116	55 (25.82%)	36	77	37	820	820
293971015	1425	240	135	24 (10.00%)	79	38	105	1392	1392
293971016	1850	541	313	45 (8.32%)	151	72	162	5225	4831
293971017	886	360	250	24 (6.67%)	128	44	128	9532	9516
293971018	1280	368	137	19 (5.16%)	106	32	298	5962	5440
293971019	993	314	119	22 (7.01%)	123	41	241	5492	5289
293971020	1822	349	243	33 (9.46%)	97	59	146	1637	1636
Average	**1372**	**352**	**204**	**29 (9.27%)**	**109**	**48**	**176**	**4329**	**4203**

in the Solana network in all three historical periods, with an average of ∼9.27% in recent blocks, while write-write conflicts dominate and contribute to ∼95.76% of all conflicts in the block. The longest chain of conflicting transactions has seen downward trends, from 84.92% in the old historical period to 35.62% in the recent historical period, but the number of conflict families has upward trends from 3 to 39, indicating more granular bottlenecks (conflicts) and increased parallelism. Further, the success rate of non-voting transactions has dropped from ∼85.32% to 54.50%, which highlights the need for adaptive or hybrid execution strategies that exploit the access specifications efficiently to improve throughput and reduce failure rates.

4 Discussion from Ethereum to Solana

It is important to understand the conflict ratio of block transactions in blockchains for efficient parallel execution. Both Ethereum and Solana exhibit significant parallel execution potential; they differ in key aspects of available concurrency and potential blockers for parallel execution. Ethereum, with lower conflict rates, allows for higher percentages of independent transactions, which could result in higher parallelism, showing potential for execution efficiency and lower abort rates in optimistic execution. While Solana's high conflict rates, particularly due to write-write conflicts, have resulted in more granular congestion or transaction conflict families (subsets of conflicting transactions), they also highlight the limitations of its current execution model. While showing scope for further optimizations.

Despite Solana's greater transaction throughput on the mainnet, it faces the challenge of increasing transaction failures. On the other hand, Ethereum still executes transactions sequentially; there is ongoing research in parallel execution approaches [7, 9, 10, 12, 15] for EVM inspired by software transactional memory that could handle such contention more effectively for Ethereum. Given the current trends, we believe that both networks (Ethereum and Solana, including other popular EVM and SVM-based chains) would benefit from adaptive and hybrid scheduling techniques to exploit parallelism for higher throughput. Solana, in particular, requires innovations to mitigate write-write conflicts, potentially through the adoption of multi-version data structures.

References

1 Chainstack. Ethereum getTransactionByBlockNumberAndIndex API Reference. https://docs.chainstack.com/reference/ethereum-gettransactionbyblocknumberandindex, 2025. [Accessed: 12 February 2025].

2 Chainstack. Ethereum traceBlockByNumber API Reference. https://docs.chainstack.com/reference/ethereum-traceblockbynumber, 2025. [Accessed: 12 February 2025].

3 CryptoKitties. CryptoKitties Website. https://www.cryptokitties.co/, 2024. Accessed: 2024-10-30, Contract Address: 0x06012c8cf97BEaD5deAe237070F9587f8E7A266d, Creation Transaction: 0x691f348ef11e9ef95d540a2da2c5f38e36072619aa44db0827e1b8a276f120f4.

4 Ethereum (ETH): open-source blockchain-based distributed computing platform. https://www.ethereum.org/. [Online: accessed 15 January 2024].

5 Ethereum Foundation. Ethereum 2.0 Merge. https://ethereum.org/en/upgrades/merge/, 2022. [Accessed: 06 December 2024].

6 Rati Gelashvili, Alexander Spiegelman, Zhuolun Xiang, George Danezis, Zekun Li, Dahlia Malkhi, Yu Xia, and Runtian Zhou. Block-stm: Scaling blockchain execution by turning ordering curse to a performance blessing. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP '23, page 232–244, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3572848.3577524.

7 Monad Labs. Parallel execution & monad. https://medium.com/monad-labs/parallel-execution-monad-f4c203cddf31. [Online: accessed 10 January 2024].

8 SEI Research. 64-85% of ethereum transactions can be parallelized. https://blog.sei.io/research-64-85-of-ethereum-transactions-can-be-parallelized, December 2024. [Accessed: 12 February 2025].

9 Supra Research. Access specification aware software transactional memory techniques for efficient execution of blockchain transactions. Technical report, https://supra.com, February 2025. https://supra.com/documents/Supra_Specification_Aware_STM_whitepaper.pdf.

10 RISE Labs. PEVM: Parallel Ethereum Virtual Machine. https://github.com/risechain/pevm, 2023. [Accessed: 30 October 2024].

11 Vikram Saraph and Maurice Herlihy. An empirical study of speculative concurrency in ethereum smart contracts. In *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*, pages 4:1–4:15, Dagstuhl, Germany, 2019. OpenAccess Series in Informatics (OASIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2020/11968, doi:10.4230/OASIcs.Tokenomics.2019.4.

12 Sei- the layer 1 for trading. https://docs.sei.io/advanced/parallelism. [Online: accessed 22 Sep 2023].

13 Solana documentation. https://docs.solana.com/. [Online: accessed 10 January 2024].

14 Solana. Solana getBlock RPC API Reference. https://solana.com/docs/rpc/http/getblock, 2025. [Accessed: 12 February 2025].

15 Polygon Technology. Innovating the main chain: A polygon pos study in parallelization, December 2023. [Accessed: 12 February 2025]. URL: https://polygon.technology/blog/innovating-the-main-chain-a-polygon-pos-study-in-parallelization.

16 Umbraresearch. Lifecycle of a solana transaction. https://www.umbraresearch.xyz/writings/lifecycle-of-a-solana-transaction. [Online: accessed 15 January 2024].

17 Anatoly Yakovenko. Sealevel - parallel processing thousands of smart contracts. https://medium.com/solana-labs/sealevel-parallel-processing-thousands-of-smart-contracts-d814b378192, September 2019. [Online: accessed 23 January 2024].