Maat: Analyzing and Optimizing Overcharge on Blockchain Storage

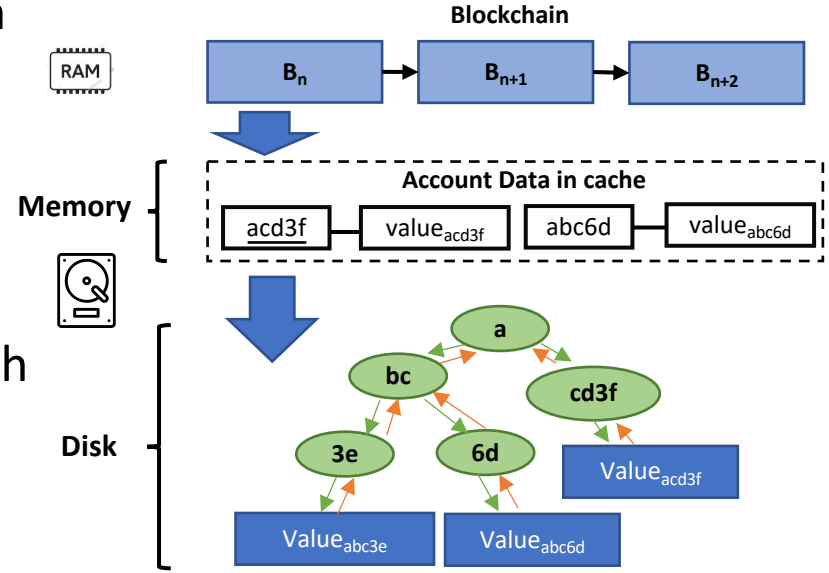
Zheyuan He[1], Zihao Li[2], Ao Qiao[1], Jingwei Li[1], Feng Luo[1], Gelei Deng[3], Shuwei Song[1], Xiaosong Zhang[1], Ting Chen[1], Xiapu Luo[2]

[1]University of Electronic Science and Technology of China
[2]The Hong Kong Polytechnic University
[3]Nanyang Technological University

23rd USENIX Conference on File and Storage Technologies (FAST 25)

Blockchain and its world state

- Blockchain is a distributed ledger consisting of a chain of blocks, each containing transactions that interact with the blockchain world state.
- Blockchain world state
  - Blockchain world state contains all account and contract data, and is organized as Merkle Patricia Tries, with nodes stored on disk.
  - To improve efficiency, a partial blockchain world state is stored in memory caches shared by several adjacent blocks (max 128 blocks).



Read and write on world state

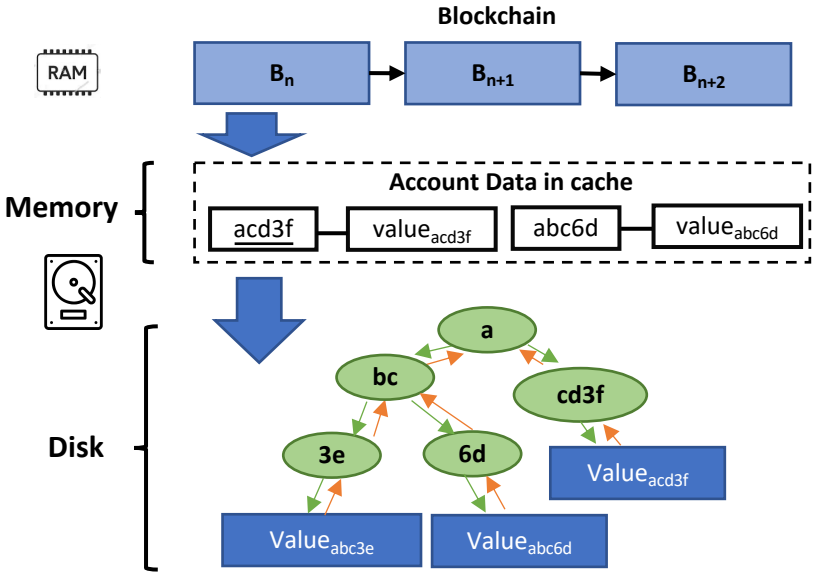
THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Read on world state:

Transactions first fetch data from the memory caches. If the caches miss, transactions will access the data stored on hard disk.

Write on world state:

Transactions first write the updated data to the memory caches. After the transactions in a block are finished, the updated data will be written to hard disk.



Transaction fee mechanism

- To prevent unlimited access and modification on world states, blockchain charges users with some transaction fee based on the usage of on-chain resource, which is measured in gas units.
- Gas cost
  - Every transaction/contract operations (e.g., data access on disk) will be assigned a *gas cost.*
  - Gas cost is designed to reflect the actual resource consumption
    - Reading contract data from memory caches: 100 gas units
    - Reading contract data from disk: 2100 gas units

Table 2: Charges for opcodes on blockchain storage.

opcode	$g_{memload}$ (gas)	$g_{memstore}$ (gas)	$g_{diskload}$ (gas)	$g_{diskstore}$ (gas)
sload	100	N.A.	2,100	N.A.
balance	100	N.A.	2,600	N.A.
extcodesize	100	N.A.	2,600	N.A.
extcodehash	100	N.A.	2,600	N.A.
extcodecopy	100	N.A.	2,600	N.A.
call/callcode	100	N.A.	2,600	9,000
delegatecall	100	N.A.	2,600	N.A.
staticcall	100	N.A.	2,600	N.A.
$call_{external}$	N.A.	N.A.	N.A.	9,000
sstore	100	100	2,100	2,900
create/create2	N.A.	N.A.	N.A.	L_{code}*200

$call_{external}$ indicates external transactions (i.e., EOA to EOA), L_{code} indicates the length of the bytecode.

Our focus

In this work, we focus on *those storage related gas cost*

- Blockchain, like Ethereum, has been suffering from high transaction fees as a huge problem. For example, in 2023, users paid 2.4 billion USD in transaction fees on Ethereum.
- The storage related gas cost accounted for more than 70% of these transaction fees.

"High fees are a huge problem and the main thing that prevents the platform from being used for cool stuff today is just the fees." — Vitalik Buterin, Ethereum co-founder

Overcharge issues

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

- Overcharge issues
  - By investigating the specifications and implementations of Ethereum clients, we uncover ***three overcharging issues***.
  - The gas cost for account/contract operations is higher than the actual resource usage

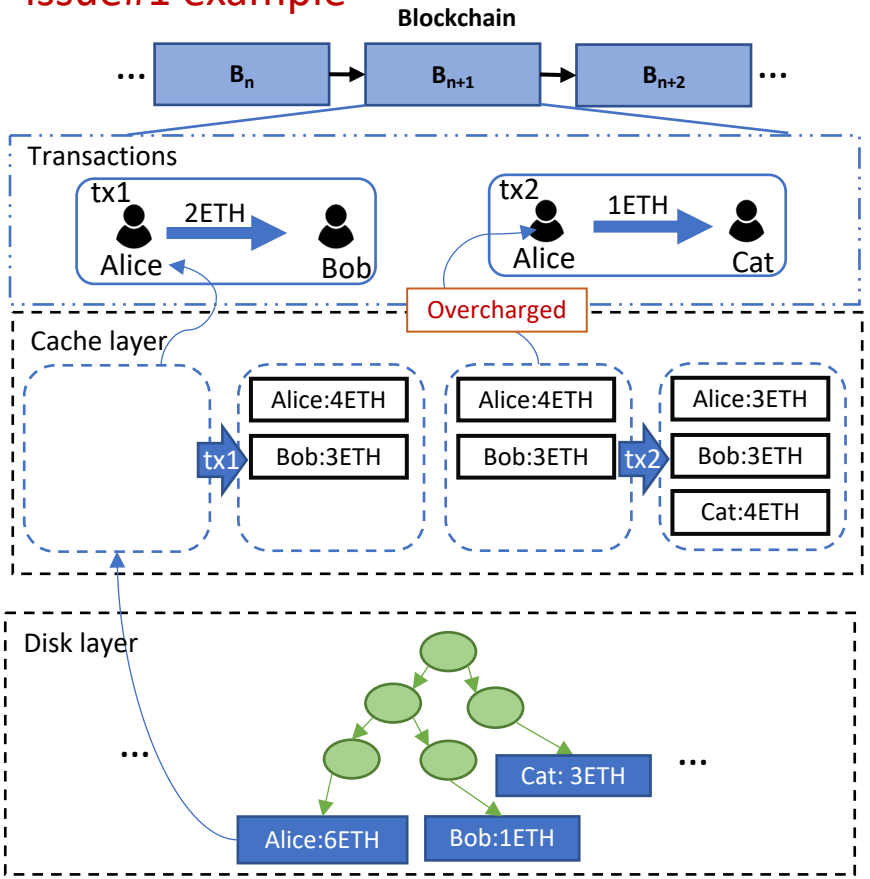
  E.g., charging a memory load operation as the fee of a disk load operation.

Overcharge issues

➢Issue#1

- Scenario: two transactions continuously read/write the same object in a block.
- Overcharged cases: Charging the gas cost of a disk read/write operation (2,100 gas) for a memory read/write operation on intra-block memory cache.

Issue#1 example

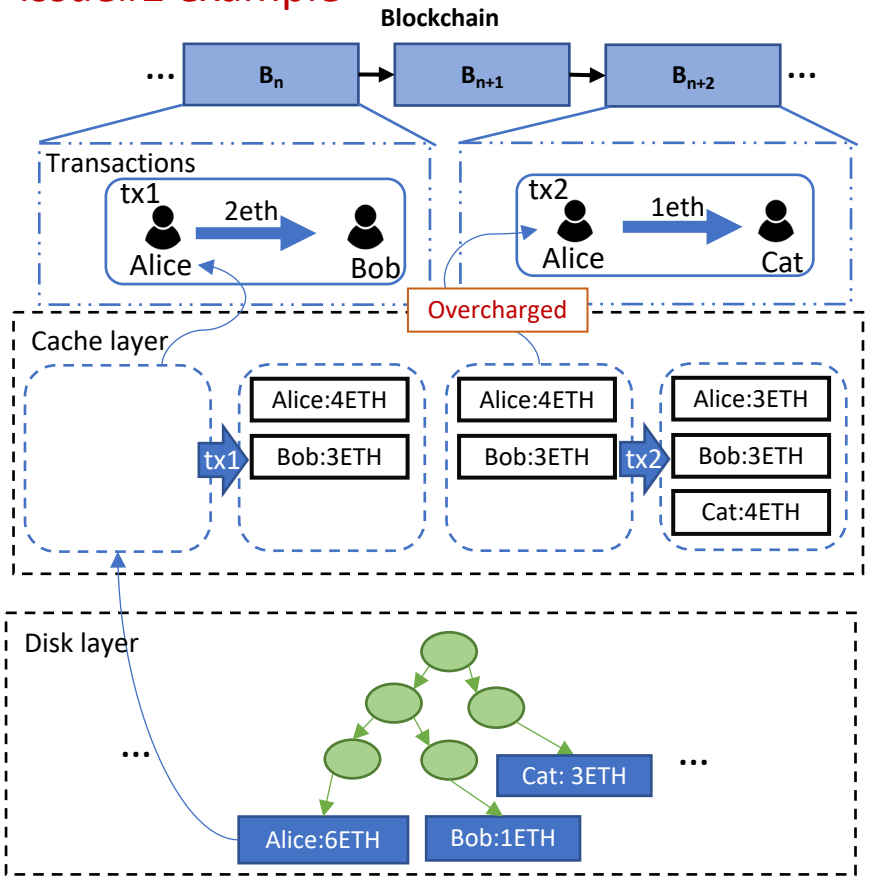


Overcharge issues

➢Issue#2

- Scenario: Two transactions continuously read the same object within 128 blocks.
- Overcharged cases: Charging the gas cost of a disk read operation (2,100 gas) for a memory read operation on cross-block memory cache.

Issue#2 example

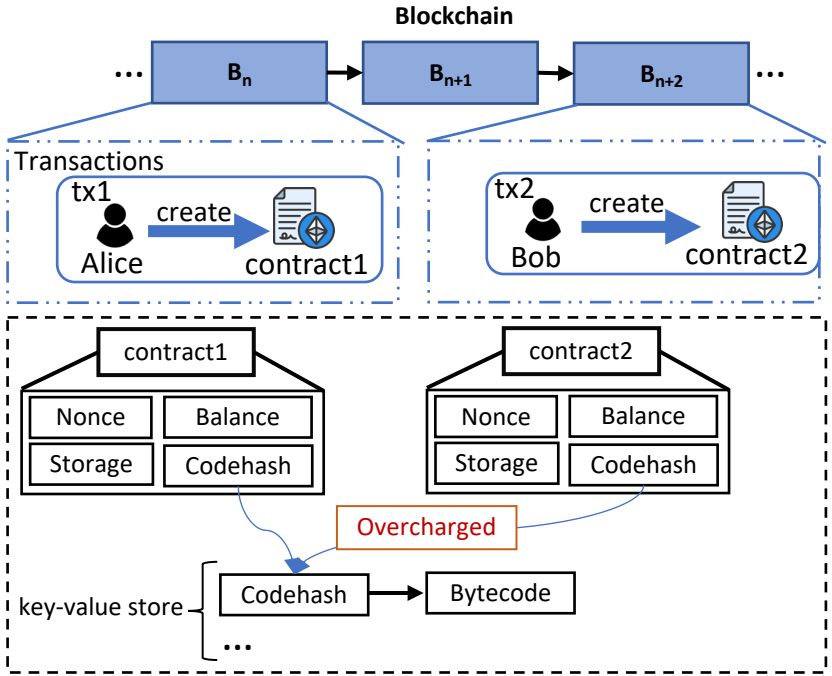


Overcharge issues

➢Issue#3

- Scenario: two transactions deploy contracts with duplicated bytecode.
- Overcharged cases: Charging an unnecessary disk write fee for storing the duplicated bytecode on disk (200 gas per byte).

Issue#3 example



Impact of three overcharging issues

- We measure the overcharging issues on Ethereum and BSC within the block range from #18M (Aug 26, 2023) to #19M (Jan 13, 2024) and the block range from #34.15M (Dec 07, 2023) to #35.15M (Jan 11, 2024)

	Ethereum	BSC
Transactions (Issue#1)	43.7% (69M)	63.1% (106M)
Fees (Issue#1)	25.6% (90M USD)	23.7% (6M USD)
Transactions (Issue#2)	42.4% (67M)	62.0% (104M)
Fees (Issue#2)	12.9% (45M USD)	15.3% (4M USD)
Transactions (Issue#3)	0.5% (1M)	0.3% (0.5M)
Fees (Issue#3)	3.5% (12M USD)	2.7% (1M USD)
Transactions (Total)	70.4% (117M)	92.8% (156M)
Fraction of fees (Total)	42.0% (147M USD)	41.7% (11M USD)

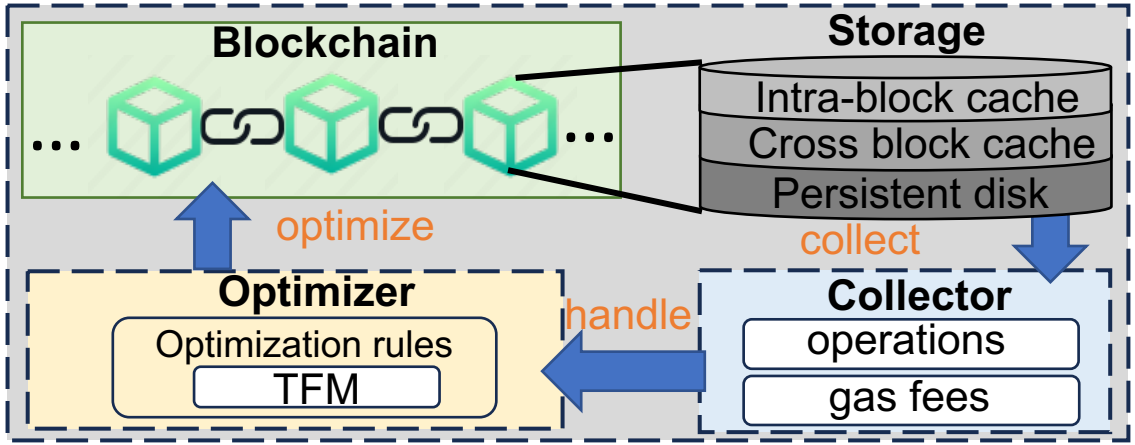
Maat Overview

THE HONG KONG POLYTECHNIC UNIVERSITY 香港理工大學

✓Target: Optimize the identified overcharging issues in blockchains

➢Methodology: Align the actual workloads of storage operations with gas fees.

- Collector: Gather storage operations and the fine-grained gas fees of the storage operations from the block and transaction execution in clients.
- Optimizer: Apply four optimization rules to optimize the gas fees in the actual cases of overcharging issues collected by Collector.



Four optimization rules

- O1/O2 target Issue#1, i.e., repeated memory reads/writes are incorrectly charged as disk reads/writes within the intra-block cache.
  - Replacing the incorrect charges of disk read/write with the fee of memory read/write.
- O3 addresses Issue#2, i.e., repeated memory reads are mistakenly charged as disk reads in the cross-block cache.
  - Replacing the incorrect charges of disk read as the fee of memory read.
- O4 mitigates Issue#3, i.e., deploying contracts with duplicated bytecode, where redundant disk fees are charged.
  - Eliminating the charges of deploying contracts with duplicated bytecode.

Design consideration

- Q1: How does Maat ensure that optimized gas charges are consistent across all clients?
  - We harness the inherent data structures of the blockchain to formalize the conditions for the four optimization rules.
  - The inherent data structures include block, transaction, account, contract variables, and contract opcodes. The execution over these structures is deterministic across clients, as guaranteed by blockchain consensus.

Table 3: Optimization rules against overcharging issues.

Optimization rules	Issues
$O1: \dfrac{^{1}\left(tx_i, tx_j \in B\right) \wedge ^{2}\left(ins.load\left(loc_i\right) \in tx_i\right) \wedge ^{3}\left(ins.load\left(loc_j\right) \in tx_j\right) \wedge ^{4}\left(loc_i == loc_j\right)}{^{5}G\left[ins.load\left(loc_j\right)\right] := G_{memload}}$	Issue#1

Design consideration

- Q2: How does Maat guarantee the storage operations are consistent across all clients?
  - ? O1-3 depend on the intra-block and cross-block caches to conduct optimization. However, the different configurations and clients may vary the contents in caches.
  - We propose a resource pre-allocation technique to cache the data accessed in previous 128 blocks to assign the cache contents across clients in advance.

Table 3: Optimization rules against overcharging issues.

Optimization rules	Issues
$O1: \frac{{}^{1}(tx_i, tx_j \in B) \wedge {}^{2}(ins.load(loc_i) \in tx_i) \wedge {}^{3}(ins.load(loc_j) \in tx_j) \wedge {}^{4}(loc_i == loc_j)}{{}^{5}G[ins.load(loc_j)] := G_{memload}}$	Issue#1

Account: $\frac{30M}{2.6K\ gas} \times 164\ bytes = 1.8\ MiB$

$1.8MiB \times 128\ blocks =$ 230 MiB

Slot: $\frac{30M}{2.1K\ gas} \times 96\ bytes = 1.3\ MiB$

$1.3MiB \times 128\ blocks =$ 167 MiB

Evaluation

- Baseline: EIP-2929, which is designed to address overcharging issues, where repeated memory read is incorrectly charged as disk read in the intra-block cache (partial Issue#1).
- We conduct the evaluation on 1M blocks on Ethereum and BSC blockchains.
- Ethereum: Maat achieves a significant optimization of 112M USD in transaction fee, and outperforms baseline about $3\times$.
- BSC: Maat makes optimization of 7.71M USD in transaction fee, and outperforms baseline about $3\times$.

Table 4: Optimization effect on Ethereum overcharging issues.

Metrics	Optimized gas (gas)	Optimized ether (ETH)	Optimized fee (USD)
1M blocks (Maat)	2.01×10^{12}	58,358.52	1.12×10^{8}
1M blocks (Baseline)	0.67×10^{12}	21,221.28	0.39×10^{8}
Per block (Maat)	2.01×10^{6}	0.058	112.09
Per block (Baseline)	0.67×10^{6}	0.021	39.02
Per transaction (Maat)	12,730	3.69×10^{-4}	0.71
Per transaction (Baseline)	4,247	1.34×10^{-4}	0.25
Per opcode (Maat)	463.08	1.34×10^{-5}	0.026
Per opcode (Baseline)	155.54	4.93×10^{-5}	0.009
Per operation (Maat)	342.67	9.95×10^{-6}	0.019
Per operation (Baseline)	113.66	3.60×10^{-6}	0.006
Optimized rate (Maat)	33%	32%	32%
Optimized rate (Baseline)	11%	12%	11%

1M blocks represents Ethereum block #18M (Aug-26-2023) to #19M (Jan-13-2024)

Takeaways

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

- We identify three overcharging issues in blockchain implementations, which affect more than 70% of transactions and 40% of transaction fees.
- We propose a novel tool, Maat, to address the three overcharging issues.
- We assess the effectiveness of Maat on Ethereum and BSC, and it achieves 5.6M USD fees in weekly savings, and outperforms the baseline by 3×.
- **What's more?**
  - ✓ Effect of distinct optimization rules and their combinations.
  - ✓ Overhead of Maat at space overhead (5.6%) and time overhead (1.4%).
  - ✓ Scalability of Maat on 50 blockchains in Ethereum ecosystems.

About me: https://zzzihao-li.github.io/

Thank you!