Bloom Filter Variations for Enhanced Log Search in Ethereum

George Drakoulis, Andreas Sendros, Periklis Kostamis, Pavlos Efraimidis

Electrical and Computer Engineering Dept., Democritus University of Thrace

#Blockchain
#Data_Structures

Problem Statement

Ethereum's exponential log growth creates computational bottlenecks in data retrieval.

• Current Bloom Filters in Ethereum [1, 2] produce false positives - unnecessary block scans.
• Standard implementation doesn't exploit Ethereum-specific log characteristics.
• Need for optimized data structures [3] as blockchain scales.

Background & Motivation

What are Bloom Filters [4]?

Probabilistic data structure for fast set membership testing
Use k hash functions to map elements to bit array positions
• Space-efficient: Much smaller than storing complete sets.
• Key property: No false negatives, but allows false positives.
• Uses k independent hash functions h_1, h_2, ...
• Bit array of size m, initially all bits set to 0.
• False Positive Probability:
P(false positive) ≈ (1 - e^(-kn/m))^k

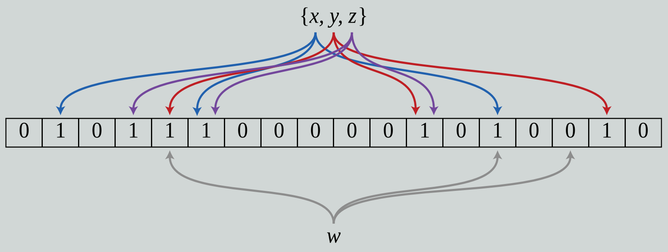

Figure 1. Bloom Filter Hash Function Mapping [5]

Bloom Filters in Ethereum

• 2048-bit filters in block headers summarize log addresses and topics.
• Hash functions, k = 3.
• Enable fast membership testing to filter irrelevant blocks.
• Reduce computational overhead by avoiding full block scans.
• Current limitation: False positives still require additional verification.

Bloom Filter Variations [6]

• One Hashing BFs: Use single hash function, reduced computational overhead but increased false positives [7].
• Ultra-Fast BFs: Leverage SIMD parallelization, faster operations but require specialized hardware [8].
• Compressed BFs: Optimize storage through compression, reduced space but additional computational overhead.
• Elastic BFs: Dynamically adjust filter structure, lower false positives but require more storage/processing.

Two Types of False Positives Identified

• Compressed False Positives: Inherent to Bloom Filter probabilistic nature.
• Justified False Positives: When log elements exist separately from different.

Dataset & Experimental Setup

Block Period	Total Logs	% Logs with 1 Topic	% Logs with 2 Topics	% Logs with 3 Topics	% Logs with 4 Topics
All Blocks	4,207,764,482	12.1%	12.6%	62.7%	13%
2019	247,999,603	10.9%	12.3%	66.1%	14%
2024	843,504,299	13.3%	12.6%	65.2%	8.9%
Our Dataset	1000	11.3%	14.7%	67.1%	6.4%

Table I: General Distribution of Topics in the Ethereum Blockchain

Dataset

• 1,000 logs from Ethereum blocks.
• Representative of current log topic distribution.
• Logs categorized by number of topics (1-4).

Infrastructure

• Intel i9-13900K, 128GB RAM, 2TB M.2 storage
• Python 3.6 with eth_bloom, web3, CryptoHash libraries Data accessed via Infura.

Proposed Variations

Proposed Variations

• DTI (Different Topic Insertion): Combines address+topics as single elements to reduce justified false positives.
• Compressed Bloom Filter: Increases filter size, reduces hash functions, enables efficient compression.
• Compressed DTI: Merges both techniques for optimal false positive reduction.

Method	Key Innovation	Technical Details	Main Benefit	Trade-off
DTI Bloom Filter	Insert address+topics as single element	2 bits/element (vs. 3 standard)	Reduces justified false positives	Less efficient for batch queries
Compressed BF	Larger filter + compression	5,000 bits → k=1 → compress to ~2,048	Lower false positive rate	Decompression overhead
Compressed DTI	Combines both techniques	8,000 bits, single element, k=1	Lowest false positive rate	Highest computational cost

Table II: Overview of the three main Bloom Filter variations

Results

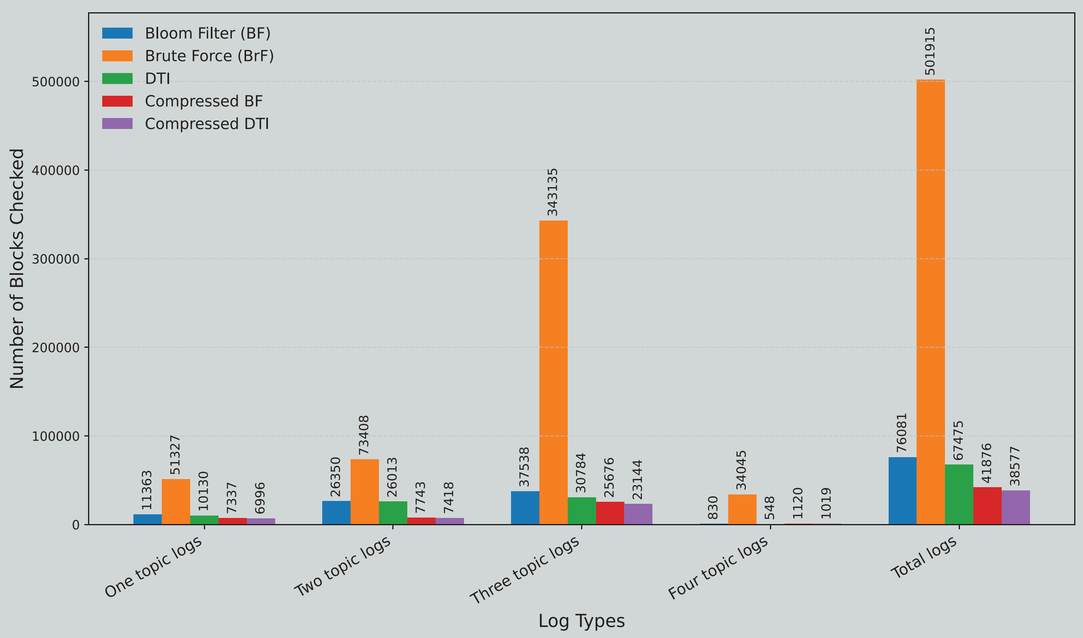

Figure 2. Comparison of Blocks Checked Across Bloom Filter Variations

Evaluation Metrics

• Number of Blocks Checked per query.
• Computational Overhead.
• False Positive Rate.

False Positive Reduction:

• DTI method effectively eliminate justified false positives.
• Compressed method achieve lower false positive rates through optimized parameters.
• Combined approaches (Compressed + DTI) provide maximum benefit.

Trade-offs:

• DTI: Better for specific queries, less efficient for batch operations.
• Compressed: Require decompression overhead but maintain 2048-bit compatibility.
• Transaction hash inclusion: Eliminates all justified false positives at computational cost.

Conclusions

All proposed methods outperform Ethereum's default BF in false positive reduction.

• DTI: Simple and effective, best for targeted queries.
• Compressed BF: Balanced improvement, slight overhead.
• Compressed DTI: Lowest false positives, but requires decompression.

Future Directions

• Testing in private Ethereum networks for real-world validation.
• Advanced compression techniques exploration and untested DTI combinations.
• Integration with existing Ethereum client implementations.

References

[1] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, 2014.
[2] V. Buterin et al., "A next-generation smart contract and decentralized application platform," white paper, vol. 3, no. 37, pp. 2–1, 2014.
[3] P. Kostamis, A. Sendros, and P. S. Efraimidis, "Data management in ethereum dapps: A cost and performance analysis," Future Generation Computer Systems, vol. 153, pp. 193–205, 2024
[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, no. 7, pp. 422–426, 1970.
[5] "Bloom Filter," Wikipedia Commons, Available: https://upload.wikimedia.org/wikipedia/commons/a/ac/Bloom_filter.svg
[6] L. Luo, D. Guo, R. T. Ma, O. Rottenstreich, and X. Luo, "Optimizing bloom filter: Challenges, solutions, and comparisons," IEEE Communi- cations Surveys & Tutorials, vol. 21, no. 2, pp. 1912–1949, 2018.
[7]J. Lu, T. Yang, Y. Wang, H. Dai, L. Jin, H. Song, and B. Liu, "One-hashing bloom filter," in 2015 IEEE 23rd international symposium on quality of service (IWQoS). IEEE, 2015, pp. 289–298.
[8] J. Lu, Y. Wan, Y. Li, C. Zhang, H. Dai, Y. Wang, G. Zhang, and B. Liu, "Ultra-fast bloom filters using simd techniques," IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 4, pp. 953–964, 2018.

euclid.ee.duth.gr
@sendrosandreas
EUCLID
Algorithms & Privacy Research Unit @DUTH

This research is partially supported by "Cascade Funding" received by the Horizon Europe project entitled "TRUSTCHAIN - Fostering a Human-Centred, Trustworthy and Sustainable Internet" (Grant Agreement N°101093274 with the European Commis-