AlterEgo: A Dedicated Blockchain Node For Analytics

Qi Guo
Max Planck Institute for Informatics
Saarbrücken, Germany
qiguo@mpi-inf.mpg.de

Mahdi Alizadeh[*]
Max Planck Institute for Software Systems
Saarbrücken, Germany
malizade@mpi-sws.org

Ali Falahati[*]
Max Planck Institute for Software Systems
Saarbrücken, Germany
afalahat@mpi-sws.org

Laurent Bindschaedler
Max Planck Institute for Software Systems
Saarbrücken, Germany
bindsch@mpi-sws.org

ABSTRACT

Blockchains today amass terabytes of transaction data that demand efficient and insightful real-time analytics for applications such as smart contract hack detection, price arbitrage on decentralized exchanges, or trending token analysis. Conventional blockchain nodes, constrained by their RPC APIs, and specialized ETL-based blockchain analytics systems grapple with a trade-off between materializing pre-calculated query results and analytical expressiveness. In response, we introduce AlterEgo, a blockchain node architected specifically for analytics that maintains parity with traditional nodes in ingesting consensus-produced blocks while integrating a robust analytics API. Our prototype supports efficient transactional and analytical processing while circumventing the rigidity of ETL workflows, offering a better trust model, enabling distributed and collaborative querying, and achieving significant performance improvements over the state-of-the-art.

CCS CONCEPTS

• **Computing methodologies** → *MapReduce algorithms*; • **Computer systems organization** → **Peer-to-peer architectures**; • **Networks** → *Peer-to-peer networks*; • **Information systems** → **Online analytical processing engines**; *MapReduce-based systems*; **Extraction, transformation and loading**.

KEYWORDS

Blockchain Analytics, Real-Time Analytics, Blockchain Node Architecture, HTAP, ETL, Decentralized Analytics

ACM Reference Format:
Qi Guo, Mahdi Alizadeh, Ali Falahati, and Laurent Bindschaedler. 2024. AlterEgo: A Dedicated Blockchain Node For Analytics. In *7th International Workshop on Edge Systems, Analytics and Networking (EdgeSys '24), April 22, 2024, Athens, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3642968.3654814

[*] Both authors contributed equally to this research.

This work is licensed under a Creative Commons Attribution International 4.0 License.

EdgeSys '24, April 22, 2024, Athens, Greece

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0539-7/24/04.
https://doi.org/10.1145/3642968.3654814

1 INTRODUCTION

Decentralized Ledger Technology (DLT) is rapidly gaining traction across various industries. As of December 2023, Ethereum, one of the leading blockchains, contains over one terabyte of transaction data, growing at a rate of over a million transactions per day [1]. As a result, the need for efficient and low-latency blockchain analytics systems has become increasingly paramount. While adept at handling transactional data, traditional blockchain nodes fail to provide comprehensive analytics capabilities. These nodes typically offer Remote Procedure Call (RPC) APIs, allowing clients to access detailed information about transactions, states, and events. However, these APIs cannot handle complex queries essential for in-depth analytics, such as filters, joins, and aggregation.

Due to the limitations of blockchain nodes, researchers and companies have recently introduced specialized systems to support analytics workloads [2, 13, 16, 21]. The prevailing approach in current systems for blockchain analytics involves executing Extract, Transform, and Load (ETL) processes to obtain data from a blockchain node. This paradigm first entails extracting fine-grained information from the blockchain node and processing the data in external analytics systems, which presents several challenges. First, extensive resources are required to extract and store the data. Second, it introduces considerable latency in processing data updates. Third, moving processing out of the blockchain nodes, often to the cloud, hurts the decentralization of the overall system. Fourth, it creates a dependency on the blockchain node serving the raw data as a single source of truth. Finally, it requires knowing the specific data to be queried in advance, making it impossible to respond to unexpected queries without initiating another ETL process to collect the missing data. Overall, these issues contribute to system inefficiency and suboptimal user experience.

This paper proposes a novel blockchain analytics solution, *AlterEgo*, representing a radical shift from the current ETL paradigm. AlterEgo is a blockchain node that mirrors the functionality of a standard node but is *inherently designed to support analytics workloads.* It maintains a copy of the entire blockchain, functioning as an archive node[1], and directly ingests new blocks produced by the consensus mechanism instead of extracting them from a single blockchain node via RPC. Aside from adding an analytics API, AlterEgo nodes are indistinguishable from traditional nodes from

[1] An archive node stores a copy of the blockchain since inception and allows querying the state at any block height.

the perspective of the blockchain system: they are globally distributed close to end-users, running in the same environments as traditional nodes, they can participate in consensus, relay transactions, and validate blocks. As a result, AlterEgo nodes provide additional redundancy and increase the decentralization of the underlying blockchain. What sets AlterEgo apart is its internal architecture, which revolves around a columnar-vectorized store that supports analytical query workloads and fast data ingestion. Moreover, this design enables a more potent model for blockchain analytics that supports decentralized query processing.

To validate our concept, we have developed a prototype of AlterEgo and conducted a comprehensive evaluation of its performance. We compare our approach against traditional blockchain RPCs and the leading solution in blockchain analytics, The Graph Protocol [2]. Our findings demonstrate that AlterEgo significantly outperforms current solutions while supporting low-latency data updates, more expressive querying capabilities, lower network overheads for distributed execution, and a superior trust model for data integrity and provenance.

This paper makes the following contributions:

- We introduce AlterEgo, a new kind of blockchain node for analytics that integrates seamlessly into existing blockchain environments.
- We present the design of AlterEgo and highlight its advantages over the state-of-the-art.
- We propose a novel distributed and collaborative query execution scheme where multiple AlterEgo nodes collectively answer queries.
- We implement a prototype of AlterEgo and demonstrate significant performance improvements and increased flexibility.

2 BACKGROUND & MOTIVATION

2.1 Blockchains & Smart Contracts

Blockchains are a type of distributed ledger that consists of cryptographically linked records. They are organized as peer-to-peer decentralized networks with nodes typically deployed close to end-users to enhance accessibility and reduce latency [22, 32]. This decentralized design democratizes data access and increases system robustness against single points of failure, ensuring higher levels of security and trust. Over time, blockchains have evolved from simple transactional exchange platforms to complex ecosystems supporting smart contracts with intricate states and sophisticated interactions [4, 7]. Smart contracts are written directly into code and reside on the blockchain, allowing them to interact with other contracts and user accounts [31].

In this paper, we focus on blockchain systems compatible with the widely popular Ethereum Virtual Machine (EVM), which include Ethereum [29], Polygon [17], and Arbitrum [6]. However, our ideas broadly extend to other systems [9, 26, 30]. Transactions in EVM blockchains pay a fee to node operators, known as the gas cost, to execute smart contracts. When a smart contract runs, it can emit events logged within the blockchain. These events are crucial for tracking the activity of smart contracts and state changes. Furthermore, each transaction generates a receipt, which provides essential details about the transaction, including its status, the gas used for its execution, and the log of emitted events.

2.2 Blockchain Analytics

The complexity and richness of blockchain interactions require advanced analytics to understand, monitor, and optimize the performance and security of blockchain applications and to facilitate decision-making. For example, smart contracts providing facilities such as trading, lending, or staking house millions to billions of dollars that must be continuously monitored to detect abnormal patterns in fund flow that may indicate exploitative actions or illicit activities [11]. Likewise, capitalizing on price discrepancies across decentralized exchanges promises substantial profits but demands comprehensive analyses of trade volumes and prices or fast identification of negative cycles [28]. Finally, numerous financial and social applications derive value from tracking trending tokens [25], which requires advanced window functions and aggregation methods that surpass the capabilities of the RPC APIs provided by traditional blockchain nodes.

Blockchains are fundamentally designed to support online transaction processing (OLTP) and, therefore, require integrating online analytics processing (OLAP) for effective analytics. Unsurprisingly, current blockchain analytics systems revolve around ETL processes that transfer blockchain data to an off-chain data management system for further processing. EtherQL [21] is a query infrastructure for Ethereum, offering a RESTful API that supports range and limit queries. Blockchain ETL [13] provides a collection of public datasets in relational format stored in Google BigQuery [27]. BlockSci [16] is an in-memory blockchain analytics database capable of importing data from multiple blockchains that offers a domain-specific language (DSL) for specifying graph queries. The Graph Protocol [2] is a commercial solution that lets programmers select a subset of the blockchain data to extract and query using a GraphQL interface.

However, while these systems provide efficient offline blockchain analytics, they face inherent limitations in real-time data analytics. Specifically, the ETL process introduces synchronization delays and requires trusting the blockchain nodes that provide the original data, raising concerns about the data's integrity and provenance. Moreover, most analytics systems do away with the decentralized nature of blockchains by moving the analytics storage and querying to powerful servers in the cloud, thereby gating access to the data behind proprietary and expensive APIs that limit free access to the contents of the public ledger.

2.3 An Integrated Blockchain Node

This paper advocates for a different approach to blockchain analytics that addresses the inherent limitations of current systems. The primary issue with existing solutions lies in using an explicit ETL process between two separate systems, creating a disconnect between them. Instead, we propose a unified system paralleling the Hybrid Transaction Analytics Processing (HTAP) architecture [15, 23], which combines blockchain and analytics functionalities into a single system. This integrated approach removes synchronization challenges and enables low update latency while reducing trust assumptions by sourcing transactions directly from the consensus layer. Supporting richer analytics also unlocks new opportunities for integrating blockchain nodes in multi-tier analytics systems where individual nodes can perform advanced filtering and aggregation, enabling efficient collaborative query processing.

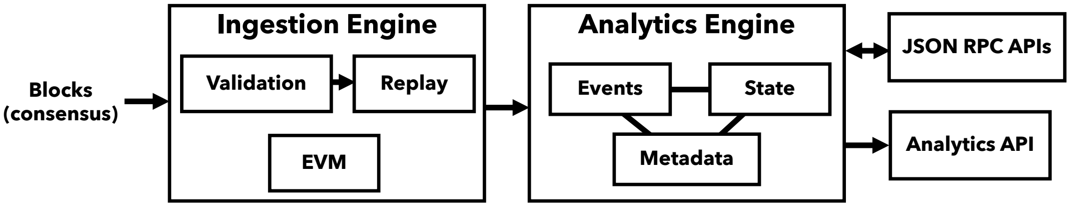

Figure 1: High-level architecture of an AlterEgo node.

3 DESIGN & IMPLEMENTATION

In this section, we describe the system design and details of AlterEgo's implementation. AlterEgo focuses on achieving two primary goals: low-latency synchronization of blockchain data to enable real-time analytics and efficient and expressive data analytics supporting arbitrary queries on any data stored in the node.

The architecture of an AlterEgo node, shown in Figure 1, follows from these design goals. The system efficiently ingests blocks, validates and replays transactions, and inserts the corresponding data into an analytics store that exposes a dedicated analytics API in addition to the standard JSON RPC APIs. In the rest of the section, we present the two main components of AlterEgo in more detail before discussing how multiple AlterEgo nodes can collectively process queries.

3.1 Ingestion Engine

AlterEgo receives blocks directly from the blockchain's consensus layer as it runs a complete blockchain node. Each block contains a set of transactions representing state changes within the blockchain, from value transfers to smart contract invocations. The contents of each block are verified and agreed upon by the network participants, providing data integrity.

Upon receipt of a block, the system proceeds, much like a conventional blockchain node, with the validation and state replaying of each transaction. Validation ensures that each transaction adheres to the network rules and does not violate any invariants, such as preventing double-spending. State replaying entails executing transactions to generate event logs and compute the new state of the blockchain. AlterEgo performs this replay in parallel for efficiency by determining the serialization order of conflicting transactions whenever possible.

Finally, AlterEgo stores the block and transaction metadata, the event logs, and the state changes in its embedded analytics database. Blocks and transactions can be batched together for efficient insertion. AlterEgo can dynamically adjust the batch size to balance performance and data freshness.

3.2 Analytical Engine

The analytical engine utilizes a columnar-vectorized store focused on efficient data retrieval to handle the vast and growing datasets in blockchain environments. We arrange the tables according to the snowflake schema [19] with event logs as the fact table, as shown in Table 1. The engine indexes and stores all the contents of the blockchain, akin to the functioning of an archive node, enabling arbitrary analytics involving any smart contract or state within the blockchain.

Column	Data Type	Description
`log_id`	BIGINT	Log identifier
`block_num`	BIGINT	Block number (sequential)
`contract`	CHAR(40)	Contract address
`signature`	CHAR(64)	Event signature
`block_ts`	TIMESTAMP	Block timestamp
`from_addr`	CHAR(40)	Sender address
`to_addr`	CHAR(40)	Receiver address
`data`	BLOB	Data field

Table 1: Fact table's schema. `log_id` is the primary key. We add indexes on all fields except `data`.

AlterEgo's analytics API is based on SQL, enabling users to execute complex queries with familiar SQL syntax. In addition to standard SQL, we support window functions and common table expressions to enhance data analysis capabilities and facilitate working with time, an essential requirement for blockchain analytics.

3.3 Distributed, Collaborative Query Execution

AlterEgo nodes are not only beneficial for simple, short queries, but they can also be integrated into more complex distributed blockchain analytics environments. In this mode, multiple AlterEgo nodes collectively execute sophisticated, long-duration queries. This collaborative effort reduces the computational load on individual nodes, boosting performance, reliability, and trustworthiness.

Since AlterEgo nodes support arbitrary SQL queries, we can design efficient dataflow analytics leveraging near-data processing at the edge in ways that traditional RPC nodes cannot due to the limited expressiveness of JSON-RPC queries. For example, AlterEgo easily supports pushdown filters or partial result aggregation that can reduce the network traffic overhead by orders of magnitude while achieving more efficient querying.

Collaborative query execution can work in two ways. In a fully partitioned fashion, the query is divided, often by time or block height, and each partition is processed by a different node. In a replicated fashion, the entire query is executed simultaneously on multiple nodes to verify the result's integrity. Figure 2 shows a simple collaborative query execution with time-based partitioning.

When processing distributed queries, it is crucial to consider that different nodes may not ingest blocks synchronously. Each node requires the specification of a block in addition to the query to ensure the overall consistency of the result. This block serves as the maximum block height to be considered for the query.

AlterEgo nodes support integrating custom analytical functions directly by leveraging the storage engine and ingestion pipelines, e.g., pattern recognition, outlier detection, or machine learning,

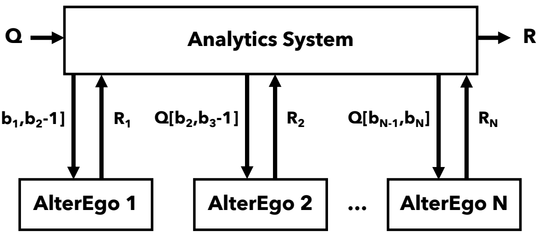

Figure 2: Collaborative query execution example. A query Q covering blocks between b_1 and b_N is partitioned by block ranges and executed on N AlterEgo nodes independently. The analytics system combines the partial results (R_i) to produce the final result R.

enabling the real-time identification of anomalies or potentially fraudulent activities. As a result, AlterEgo nodes can also serve as an effective early warning system for blockchain systems.

Efficiently coordinating multiple AlterEgo nodes for collaborative execution remains an area for further exploration. Our current approach proposes an open system where AlterEgo nodes accept queries from any user, similar to traditional blockchain RPCs. While this design is simple and flexible enough for many needs, we leave possible refinements, such as developing more secure, potentially permissioned coordination mechanisms for future work [18, 20, 24] or the integration of Byzantine Fault Tolerant query verification mechanisms, e.g., through cross-checking [8] or cryptographic proofs [5, 14], for future work. Finally, designing a sharded analytics node to address the growing size of blockchains remains an open area of exploration.

3.4 Implementation

We chose the official Ethereum node implementation in Golang, known as Geth [12], as the foundation for AlterEgo for prototyping efficiency, modifying ~270 lines of code. This decision enabled us to focus on specific enhancements, namely integrating our analytics storage engine and adding support for the analytics API, while relying on Geth's efficient transaction validation and state updates. While our prototype utilizes Geth, our approach can support other EVM-compatible blockchains with minimal adjustments. We build the analytical component of AlterEgo around DuckDB [10], an embedded OLAP database that aligns well with the needs of our system while providing satisfactory performance. We leave the design of a completely new node without a pre-existing codebase for future work.

4 EVALUATION

In this section, we conduct an evaluation of AlterEgo using several representative query benchmarks executed over the Ethereum blockchain. We focus on providing answers to the following research questions.

RQ1: How expressive is AlterEgo compared to existing solutions?

RQ2: How does the performance of AlterEgo compare with traditional RPC-based analytics and state-of-the-art solutions such as The Graph Protocol?

RQ3: How much reduction in network traffic can AlterEgo achieve compared with traditional RPC nodes when processing distributed queries?

RQ4: How fast can AlterEgo apply updates?

RQ5: How much overhead does AlterEgo introduce compared with traditional blockchain nodes?

4.1 Experimental Setup

4.1.1 Benchmarks. To the best of our knowledge, there are currently no benchmarks for blockchain analytics. Therefore, we consider four representative queries inspired by different public subgraphs available in The Graph Protocol's explorer [3]:

Q1: *Token Activity*: Count the number of transfers of an ERC20[2] token (USDT) over a given period.

Q2: *Top Users*: List the top k address pairs with the most transactions of an ERC20 token (USDT) over a given period.

Q3: *Top Holders*: List the k addresses with the largest balance of an ERC20 token (USDT) at any given time.

Q4: *Trading Volume*: Calculate the total trading volume of a trading pair (USDT-ETH) on Uniswap, a decentralized exchange.

4.1.2 Baselines. We compare with the following baselines:

- **Golang+RPC**: Collect the data necessary from the query from a local Ethereum blockchain node (Geth version 1.12.1) using JSON-RPC and process it in golang (version 1.21.3). For each query, we execute `eth_getLogs` iteratively on successive block ranges with the appropriate event filter on the relevant smart contract(s) to extract the necessary data before processing it in a custom golang program.
- **The Graph**: Execute the query directly in The Graph Protocol's GraphQL query format. We run a dedicated subgraph for each query inside a local The Graph deployment (version 0.33.0) and synchronize it with a local Ethereum node (Geth version 1.12.1). We synchronize the subgraph before the start of each experiment and exclude this pre-processing time from the reported results.

4.1.3 Configuration. We perform all experiments using servers equipped with four Intel(R) Xeon(R) E7-8857 v2 CPUs with a total of 48 cores, 1.5 TB DRAM, and 10 TB of secondary storage running Debian with Linux kernel version 5.15.130.1.amd64-smp. We make sure that no other applications are executing on the servers to avoid interference and disable Dynamic Voltage and Frequency Scaling (DVFS). Note that we mainly use these powerful machines for their extensive secondary storage (the largest capacity available in our cluster). AlterEgo nodes require little extra resources compared to the corresponding traditional blockchain nodes (see §4.6 for details).

4.2 Expressiveness

AlterEgo, RPC nodes, and The Graph present distinct paradigms influencing their expressiveness. AlterEgo employs a schema-based query model akin to traditional blockchain nodes, offering flexibility to formulate queries across various parameters and supporting ad hoc querying for evolving analytical needs. The queries in §4.1 are readily expressible in AlterEgo and benefit from its efficient execution. In contrast, while sharing some of this expressive capacity, traditional RPC nodes incur a significant efficiency trade-off due to

[2] ERC20 is the standard for fungible tokens on Ethereum.

their inability to perform advanced filtering and aggregation. They also do not operate on timestamps but on block numbers, requiring cumbersome back-and-forth translation. Finally, The Graph utilizes a fixed computational model that materializes the results of predefined queries, which confines its query potential to the bounds of anticipated queries and limits exploratory and real-time analysis. For instance, accommodating the queries from §4.1 requires extensive pre-calculation for all block heights (i.e., timestamps), resulting in increased storage demands, performance overhead, and restrictions on data granularity.

4.3 Performance Comparison

We begin with a performance comparison of AlterEgo with The Graph and Golang+RPC on each of the four queries (Q1-4) for different time ranges from 1k blocks to 1M blocks, as shown in Figure 3. To complete the comparison, we also include a version of AlterEgo where the query results are materialized (AlterEgo Mat) similarly to The Graph. Each block range starts from the 10,000,000th Ethereum block. Figure 3 shows the averages and standard deviations over ten runs for each query.

The results show that using Golang+RPC for complex queries across extensive time ranges is impractical due to the round-trip overheads and high intermediate data. In contrast, AlterEgo consistently delivers fast responses of under 5 seconds to all queries and shows better scalability as the queried time range expands. The Graph excels in handling such predefined queries, achieving impressive performance by pre-computing all results. Finally, AlterEgo Mat, which adopts a strategy of materializing query results similar to the Graph, achieves response times of ~1 ms, surpassing The Graph. Overall, the results underscore AlterEgo’s efficiency and indicate a potential to optimize performance for known queries.

4.4 Traffic Reduction

We now evaluate the network traffic reduction in AlterEgo to answer distributed queries. We compare AlterEgo with Golang+RPC when the entity issuing queries is remote. This simplified scenario illustrates a distributed querying example where different nodes are queried in parallel before aggregation.

Table 2 shows the total traffic in bytes involved when using a remote vanilla Ethereum blockchain node with JSON-RPC to answer different queries (Golang+RPC) versus a remote AlterEgo node with SQL to answer the same queries (AlterEgo). In both cases, we consider a series of increasing time ranges, starting from a 1k block range to 1M blocks. For Golang+RPC, we split queries into batches of 1k blocks at a time, as that is the maximum block range supported by RPCs in a single call. AlterEgo supports arbitrarily large ranges and executes a single SQL query.

Blocks	Golang+RPC				AlterEgo			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
1k	13 MB	13 MB	13 MB	0 B	8 B	840 B	420 B	64 B
10k	103 MB	103 MB	103 MB	0 B	8 B	840 B	420 B	64 B
100k	1.27 GB	1.27 GB	1.27 GB	1.20 KB	8 B	840 B	420 B	64 B
1M	5.76 GB	5.76 GB	5.76 GB	1.63 KB	8 B	840 B	420 B	64 B

Table 2: Total network traffic for Golang+RPC and AlterEgo with different time ranges (defined by block numbers).

AlterEgo achieves minimal network traffic overhead to answer queries, thanks to the node’s ability to perform advanced filtering, joins, and aggregations locally, requiring only the communication of queries and results. On the other hand, the traditional RPC node (Golang+RPC) suffers from heavy network overhead as it only supports basic filtering locally and must ship all intermediate transactions to the remote machine issuing the query. In all evaluated queries, AlterEgo returns a constant amount of data. Also, as the queried time range increases to 1M blocks, AlterEgo achieves up to six orders of magnitude reduction in network traffic over the traditional approach.

4.5 Update Latency

Next, we consider the data update latency (or analytical latency), i.e., the time between the receipt of a transaction and the point at which the new data is visible to queries in the different systems. We use different batch sizes by grouping blocks and applying them together to update the state of the analytics system.

Table 3 compares the average per-block update latency for AlterEgo and The Graph across various batch sizes. Notably, The Graph’s functionality is confined to indexing a single smart contract. To establish a comparison under equivalent conditions, we propose a similar workload in AlterEgo, referred to as ’AlterEgo Single,’ which contrasts with ’AlterEgo Full,’ which indexes the entire blockchain. Since The Graph’s batch size is dynamic and can only be configured with a maximum value, we report the per-block update latency average in each scenario. We perform this experiment starting from the 10,000,000th Ethereum block.

Batch Size	AlterEgo Full	AlterEgo Single	The Graph
1	91.70 ms	14.49 ms	31.16 ms
10	71.30 ms	12.64 ms	28.18 ms
100	66.85 ms	11.01 ms	30.87 ms
1,000	65.30 ms	10.73 ms	31.17 ms
10,000	64.00 ms	10.61 ms	31.20 ms

Table 3: Average Update Latency (in milliseconds) for AlterEgo and The Graph with different batch sizes.

AlterEgo Single ingests new blocks up to 3× faster than The Graph. Despite the increased workload undertaken by processing entire blocks, AlterEgo Full applies all updates in a timeframe of only 2-3× longer than The Graph.

These findings underscore the practicality of ingesting blocks within a 100-ms window, given that this time frame is shorter than the network propagation delay for blocks on most blockchains. We conclude that AlterEgo can maintain low update latency, even when processing data on a block-by-block basis.

4.6 Overhead Analysis

Finally, we briefly quantify AlterEgo’s overheads in terms of additional memory and storage requirements to support analytics. We measure the worst-case memory overhead (Q3) at an additional 6.4 GB of main memory, primarily used for the page cache and intermediate results. AlterEgo requires roughly the same storage as a traditional blockchain node. We measured the storage requirements of ingesting 1M blocks from Ethereum at 57 GB. While these overheads are non-negligible, they remain reasonable in the context of executing analytics on large amounts of data.

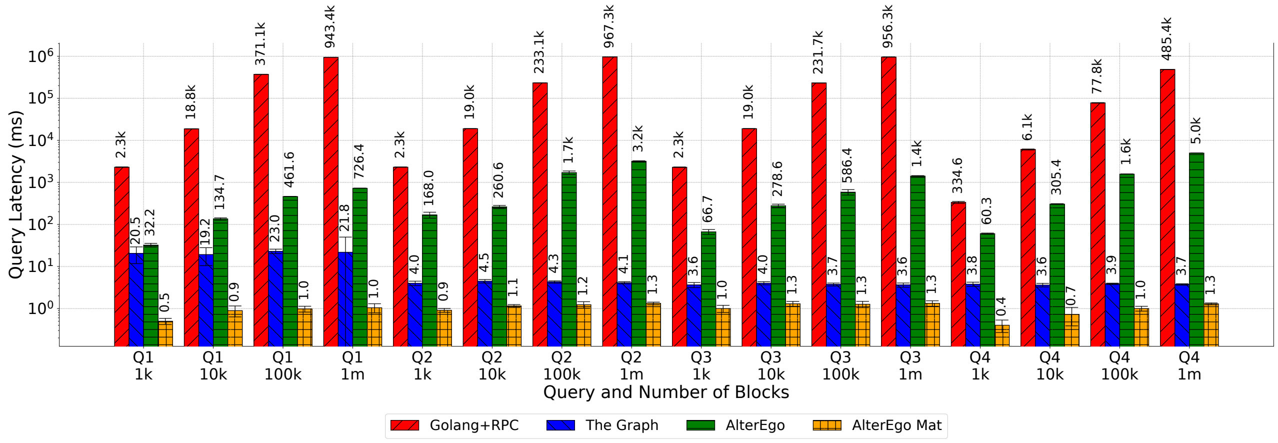

Figure 3: Comparison of query latency for the different systems. Note that the y-axis is in logscale for plot readability.

5 CONCLUSIONS AND FUTURE WORK

We introduced AlterEgo, a specialized blockchain node with advanced analytics capabilities that maintains the core functionalities of traditional blockchain nodes, offering substantial performance improvements, better user experience, collaborative query execution, and higher trustworthiness than the state-of-the-art. We plan to open-source AlterEgo upon publication of this article.

The encouraging outcomes from this initial prototype motivate the exploration of further research avenues. One key direction is reducing and balancing the load on individual nodes for long-running queries to ensure efficient operation. We plan to explore query offloading, dynamic query decomposition, and data sharding to that end. Another research opportunity involves leveraging the vast body of edge AI and analytics work to provide efficient real-time analytics primitives at the node level, enabling even more use cases. A third area for future work is connecting analytics nodes for different blockchains and non-blockchain systems to analyze their interactions. Our ultimate objective is the design of a comprehensive framework for decentralized blockchain analytics.

REFERENCES

[1] [n. d.]. https://etherscan.io. Accessed: 2024-03-29.

[2] [n. d.]. https://thegraph.com. Accessed: 2024-03-29.

[3] [n. d.]. https://thegraph.com/explorer. Accessed: 2024-03-29.

[4] Andreas M Antonopoulos and Gavin Wood. 2018. *Mastering ethereum: building smart contracts and dapps.* O'reilly Media.

[5] László Babai. 1985. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing.* 421–429.

[6] Lee Bousfield, Rachel Bousfield, Chris Buckland, Ben Burgess, Joshua Colvin, Edward W Felten, Steven Goldfeder, Daniel Goldman, Braden Huddleston, Harry Kalodner, et al. 2018. Arbitrum Nitro: A Second-Generation Optimistic Rollup. (2018).

[7] Vitalik Buterin. 2016. Ethereum: platform review. *Opportunities and Challenges for Private and Consortium Blockchains* 45 (2016).

[8] Miguel Castro, Barbara Liskov, et al. 1999. Practical byzantine fault tolerance. In *OsDI*, Vol. 99. 173–186.

[9] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems.* 34–50.

[10] DuckDB Authors. [n. d.]. DuckDB. https://duckdb.org/. Accessed: 2024-03-29.

[11] Bogdan Dumitrescu, Andra Băltoiu, and Ştefania Budulan. 2022. Anomaly detection in graphs of bank transactions for anti money laundering applications. *IEEE Access* 10 (2022), 47699–47714.

[12] Ethereum Foundation. [n. d.]. Geth (go-ethereum). https://geth.ethereum.org/. Accessed: 2024-03-29.

[13] Evgeny Medvedev. [n. d.]. Blockchain ETL. http://blockchainetl.io. Accessed: 2024-03-29.

[14] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. 2019. The knowledge complexity of interactive proof-systems. In *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali.* 203–225.

[15] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, et al. 2020. TiDB: a Raft-based HTAP database. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3072–3084.

[16] Harry Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. 2020. {BlockSci}: Design and applications of a blockchain analysis platform. In *29th USENIX Security Symposium (USENIX Security 20).* 2721–2738.

[17] Jaynti Kanani, Sandeep Nailwal, and Anurag Arjun. 2021. Matic whitepaper. *Polygon, Bengaluru, India, Tech. Rep., Sep* (2021).

[18] Donald Kossmann. 2000. The state of the art in distributed query processing. *ACM Computing Surveys (CSUR)* 32, 4 (2000), 422–469.

[19] Mark Levene and George Loizou. 2003. Why is the snowflake schema a good data warehouse design? *Information Systems* 28, 3 (2003), 225–240.

[20] Haoran Li, Chenyang Lu, and Christopher D Gill. 2021. RT-ZooKeeper: Taming the Recovery Latency of a Coordination Service. *ACM Transactions on Embedded Computing Systems (TECS)* 20, 5s (2021), 1–22.

[21] Yang Li, Kai Zheng, Ying Yan, Qi Liu, and Xiaofang Zhou. 2017. EtherQL: a query layer for blockchain system. In *Database Systems for Advanced Applications: 22nd International Conference, DASFAA 2017, Suzhou, China, March 27-30, 2017, Proceedings, Part II 22.* Springer, 556–567.

[22] Satoshi Nakamoto. 2008. Bitcoin whitepaper. *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)* (2008).

[23] Fatma Özcan, Yuanyuan Tian, and Pinar Tözün. 2017. Hybrid transactional/analytical processing: A survey. In *Proceedings of the 2017 ACM International Conference on Management of Data.* 1771–1775.

[24] Dennis Przytarski, Christoph Stach, Clémentine Gritti, and Bernhard Mitschang. 2021. Query processing in blockchain systems: Current state and future challenges. *Future Internet* 14, 1 (2021), 1.

[25] John Silberholz and Di Andrew Wu. 2021. Measuring utility and speculation in blockchain tokens. *Available at SSRN 3915269* (2021).

[26] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. 2022. Bullshark: Dag bft protocols made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security.* 2705–2718.

[27] Jordan Tigani and Siddartha Naidu. 2014. *Google bigquery analytics.* John Wiley & Sons.

[28] Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. 2022. Cyclic arbitrage in decentralized exchanges. In *Companion Proceedings of the Web Conference 2022.* 12–19.

[29] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[30] Anatoly Yakovenko. 2018. Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper* (2018).

[31] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. 2020. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems* 105 (2020), 475–491.

[32] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. 2018. Blockchain challenges and opportunities: A survey. *International journal of web and grid services* 14, 4 (2018), 352–375.