An Efficient Design and Implementation of Blockchain Architecture

Anish Gupta (Student-3rd year)
Suryansh Rohil (Student-3rd year)
B.Tech - Computer Science and Engineering
SoE, Shiv Nadar University
2310110047 (Anish), 2310110314 (Suryansh)
ag801@snu.edu.in, sr738@snu.edu.in

Dr. Sweta Kumari (Faculty Advisor)
Assistant Professor
Dept. of Computer Science and Engineering
Shiv Nadar University
sweta.kumari@snu.edu.in

***Abstract*—Blockchain technology has revolutionized how we manage secure, decentralized transactions across various industries, from finance to supply chain management. However, traditional blockchains, which use basic structures like linked lists (used in bitcoin [12]) and Merkle trees (used in hyperledger [11]), face issues with transaction speed, scalability, and energy efficiency. Directed Acyclic Graph–based blockchains mark a shift from these designs, enabling significantly higher throughput through concurrent storage and execution. In our work, we refine MorphDAG [10]'s transaction classification by replacing its coarse operation-count threshold with a per-account read/write frequency analysis. Transactions touching only low-frequency ("cold/read dominated") accounts are executed fully in parallel , while those involving high-frequency ("hot/write dominated") accounts incur minimal dependency checks. Integrated into the MorphDAG prototype and tested on a 5 iterations of the test applications with a randomly generated set of transactions based on synthethic workload of smallbank benchmark, this refinement yields a 12% reduction in block-processing latency (819ms→722ms) and a 13% increase in throughput, with no data inconsistencies over a variable number of transactions.**

I. Introduction

Blockchain technology underpins a wide range of decentralized applications (dApps), from permissionless cryptocurrencies to permissioned ledgers for supply chain and healthcare. Its core innovations are immutability, transparency, and decentralized consensus derive from chaining cryptographically linked blocks, as in Bitcoin's linked-list structure [12], or Merkle tree based designs such as Hyperledger Fabric [11]. However, these traditional architectures suffer from inherent limitations:

- **Throughput bottlenecks**: Sequential block creation and strict ordering limit transactions per second (TPS).
- **Energy inefficiency**: Proof of Work (PoW) mining consumes vast electricity, even in permissioned contexts.
- **Poor scalability**: Network and storage overheads grow linearly with chain length and transaction volume.

Directed Acyclic Graph (DAG) based blockchains address these concerns by allowing multiple blocks (or "events") to be appended concurrently, yielding higher parallelism and throughput. Notable DAG systems include IOTA's Tangle, Nano's block-lattice, and hybrid approaches such as LightDAG [14] and MorphDAG [10]. In particular, MorphDAG combines an *elastic storage concurrency* model dynamically tuning the number of parallel block producers via PoS-based VRF sortition with a dual-mode transaction processor as made by MorphDAG developers that handles "hot" and "cold" data differently to mitigate conflicts under skewed workloads.

In its original form, (old) MorphDAG classifies transactions as hot or cold based on a simple threshold of total read/write operations per transaction. While this approach improves over uniform execution, Transactions with both reads and writes can end up on the "hot" path simply because their total operation count is high even if they touch only rarely used accounts and would never actually conflict. This causes them to be serialized unnecessarily. Our per-account frequency approach fixes this by tagging a transaction as "hot" only if it really uses a busy (high-frequency) account, and otherwise running it fully in parallel., we propose a *per-account read/write frequency* refinement:

- We compute `writeFreq` and `readFreq` for each account over the current block's transaction set.
- We designate high-frequency accounts (*hot*) and low-frequency accounts (*cold*) via a tunable percentile threshold.
- Transactions that touch only cold accounts (read-dominated) execute fully in parallel ; those involving any hot accounts incur lightweight dependency checks.

We integrate this refinement into the MorphDAG prototype and evaluate it using a Smallbank synthetic benchmark tested on a local machine, Across five iterations and a variable set of transactions, our method achieves a **12% reduction in block-processing latency** and a **13% increase in throughput**, with no observed state inconsistencies. These results demonstrate that per-account frequency based classification can further unlock the parallelism inherent in DAG-based

blockchains while preserving security and correctness.

II. Literature Review

Here are a few things I learnt from online resources and research papers available in this field (briefing some of them) to explore more about this topic :-

A. Basic Blockchain terminologies and concepts

Before diving into the world of Blockchain I had the opportunity to learn about basic blockchain working, terminologies and what are all the types of blockchain available in the market.

B. Research Paper-Implementation of directed acyclic graph in blockchain network to improve security and speed of transactions

This paper explores how DAGs type structures improve blockchain networks by enhancing transaction speed and security. Unlike traditional blockchains, they allow parallel transaction processing, reducing delays and fees. This structure eliminates mining, making transactions faster, scalable, and fee-less, ideal for IoT and financial applications.

C. Research Paper-A Comparative Analysis of DAG-based Blockchain Architectures

Since we knew how DAG is very promising from previous paper, this one analyzes DAG-based blockchains like IOTA, Nano, and Byteball, highlighting their scalability, speed, and efficiency over traditional blockchains. It compares their consensus mechanisms, transaction validation, and security to identify best practices and proposes an optimized hybrid DAG model for improved performance.

D. The Merkle Tree structure

A Merkle tree is a data structure used in blockchains to organize and verify transaction data efficiently. it follows a binary tree format, where each leaf node contains a cryptographic hash of a transaction, and each non-leaf node stores the hash of its two child nodes. this process continues recursively until a single hash, known as the Merkle root, is formed at the top. The Merkle root is stored in the block header, allowing efficient verification of transaction integrity without requiring the entire dataset. bitcoin and ethereum use this structure to secure transactions.

E. Why DAG, and not Merkle tree

- **Higher scalability** – DAG enable parallel transaction processing, whereas merkle trees follow a sequential validation approach.
- **Greater throughput**– DAG efficiently process high transaction volumes, while merkle trees may create bottlenecks in large networks.
- **Faster confirmations** – Transactions in a DAG validate each other, reducing delays compared to merkle tree-based proof-of-work systems.
- **No reliance on mining** – Some DAG-based networks remove the need for mining, making them more energy-efficient.

F. Research Paper-Jointgraph: A DAG-based efficient consensus algorithm for consortium blockchains

This paper helped us to understand and learn how combining consensus with DAG strucutre can be a optimal solution as unlike traditional blockchains, JointGraph allows parallel transaction processing, reducing confirmation delays and improving efficiency. It addresses Byzantine faults by isolating malicious nodes and optimizes memory through periodic snapshots.

G. Research paper- Tangle the Blockchain:Towards Connecting Blockchain and DAG

Tangle is a hybrid Blockchain-DAG integration to improve scalability and efficiency, particularly for IoT. It introduced a connector component that facilitates seamless interaction, using Blockchain for data storage and Tangle for fast, scalable transactions.This system enhances flexibility, reliability in decentralized networks.

H. Research paper- MorphDAG: A Workload-Aware Elastic DAG-Based Blockchain

This is the technology I am currently exploring. I am breifly explaining it here:-

- MorphDAG is a workload-aware DAG-based blockchain designed to improve scalability and storage efficiency.
- It dynamically adjusts its DAG structure based on transaction patterns, optimizing performance for hotspot and cold transactions.
- By implementing elastic storage techniques, MorphDAG reduces memory usage while maintaining fast consensus.
- The system is tested under real-world workloads, demonstrating high throughput, low latency, and adaptive resource allocation, making it well-suited for large-scale decentralized applications.

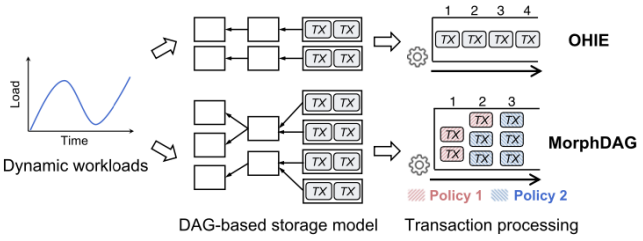

Fig. 1: Comparison of DAG-based blockchains (MorphDAG versus OHI)

Figure 1 in the MorphDAG paper illustrates the impact of a static, workload-agnostic storage concurrency setting versus MorphDAG's elastic adjustment under dynamic transaction loads. On the left, both OHIE and AdaptChain are shown with a fixed number of parallel block producers: when the incoming transaction rate surges, many transactions are delayed because the fixed parallelism cannot keep up, and when the rate falls, resources are wasted propagating redundant empty or duplicate blocks. On the right, MorphDAG continuously measures its

TABLE I: Different DAG Technologies

Comparing Factor	DAG Technologies		
	MorphDAG	***Tangle***	***JointGraph***
Scalability	High (Dynamically adjusts DAG structure)	High, but may suffer from orphaned transactions	Moderate (designed for consortium blockchains)
Transaction Speed	Very High (adaptive transaction processing)	Fast, but can slow down in low-participation scenarios	High (but limited by supervisory node bottlenecks)
Storage Efficiency	Elastic storage reduces redundant data	Prone to unbounded DAG growth	Snapshots reduce memory load
Latency	Lowest (adaptive consensus process)	Low, but varies based on network activity	Low (due to direct voting mechanism)
Security Model	Stronger (adaptive conflict resolution)	Secure, but susceptible to network attacks	Strong (malicious nodes removed)
Ideal Use Case	Large-scale decentralized applications	IoT and microtransactions	Consortium blockchains

pending transaction pool size and adjusts its degree of storage concurrency (ω) in real time. During peak load, ω increases to include more transactions in parallel blocks, reducing queuing delay; when load drops, ω decreases to avoid unnecessary block generation. The result is a consistently higher effective throughput and lower end-to-end latency across load fluctuations.

MorphDAG [10] is one of the first DAG-based blockchain systems to combine a *workload-aware* approach with a *dual-mode* execution engine. It adjusts the number of blocks created in parallel depending on the current load, and it also tries to execute transactions more efficiently by splitting them into two categories: *hot* and *cold*.

To decide whether a transaction is hot or cold, MorphDAG counts the total number of read and write operations in the transaction. If this number is higher than a certain threshold (T_{op}), the transaction is marked as **hot**. Otherwise, it is treated as **cold**.

- **Hot transactions** go through more careful processing. They are grouped by access type—Read-Only (R), Non-Incremental Write (NIW), and Incremental Write (IW)—and are checked for conflicts at the account level. Incremental writes only apply changes (not full values), which helps reduce conflicts.
- **Cold transactions** are assumed to have very low chances of conflict. So, they are run in parallel without any conflict detection, which makes execution much faster.

This method helped MorphDAG improve its performance, showing up to a **2.4×** increase in throughput compared to earlier DAG-based systems like OHIE [4] and AdaptChain [5]. However, this approach is still limited. Since it only looks at the number of operations in a transaction, it can't tell the difference between transactions that touch busy (hot) accounts and those that touch rarely used (cold) ones. As a result, some low-risk transactions may be treated as hot and processed more slowly than necessary.

In our work, we improve this by analyzing how frequently each account is read or written to, instead of just counting operations per transaction. This helps us more accurately identify truly hot accounts. With this better classification, we can safely run more cold transactions in parallel, improving speed and efficiency under realistic conditions.

III. Implementation: Refining Hot/Cold Account Classification

A key contribution of this work is the refinement of the mechanism used to classify "hot" and "cold" accounts within MorphDAG. The original implementation relied on a coarse-grained, per-transaction metric based only on the total number of operations, which could lead to performance bottlenecks. Our change introduces a more precise, block-wide analysis of account access patterns. Here is an analysis of the old method and our proposed one:-

A. Old Method: Per-Transaction Operation Count

Previously, a transaction was flagged as "hot" if the total number of read and write operations within that single transaction exceeded a predefined static threshold. While simple, this approach often misclassified transactions involving non-contentious accounts as hot, leading to unnecessary serialization.

```
// Old logic evaluated each transaction in isolation.
// A transaction was 'hot' if its total R/W operations
// exceeded a fixed threshold (e.g., Top).

opCount := 0
for _, rwsets := range tx.Payload.RWSets {
    opCount += len(rwsets) // Sum all operations in the transaction
}

if opCount > Top_threshold {
    // Mark the entire transaction as hot, even if the
    // underlying accounts were not frequently used.
}
```

Listing 1: Old Method: Simplified Per-Transaction Logic

B. Our Method: Per-Account Frequency Analysis

Our updated approach, analyzes the access patterns across all transactions in the current block. Instead of a simple operation count, it calculates the **write frequency** for each individual account. It then designates the top percentile of most frequently written-to accounts as "hot." This ensures that only accounts that are genuine points of contention are processed with stricter dependency checks.

Algorithm 1 Hot Account Detection (Per-Account Write Frequency)

Input: Transactions T, Ratio threshold r
Output: Set of hot accounts $HotSet$

Initialize empty maps: `writeFreq`, `readFreq`
Initialize empty set: `HotSet`
for each transaction tx in T **do**
    for each account acc accessed in tx **do**
        if operation is a `read` **then**
            `readFreq[acc]++`
        else if operation is a `write (iw)` **then**
            `writeFreq[acc]++`
        end if
    end for
end for
Extract all `writeFreq` values into a list F
Sort F in descending order
Calculate number of hot accounts: $k \leftarrow \lfloor |F| \times r \rfloor$
if $k = 0$ and $|F| > 0$ **then**
    $k \leftarrow 1$ ▷ Ensure at least one hot account
end if
Set `hotThreshold` = $F[k-1]$ ▷ The k-th highest write frequency
for each account acc in `writeFreq` **do**
    if `writeFreq[acc]` $\geq$ `hotThreshold` **then**
        Add acc to `HotSet`
    end if
end for
return `HotSet`

This change from a transaction-level assessment to a block wide, account-level frequency analysis is fundamental. It allows the system to more intelligently parallelize the execution of transactions that, despite having many operations, only touch ”cold” or low-contention accounts. This targeted approach is directly responsible for the observed reductions in latency and increases in throughput. “‘

IV. Performance Evaluation and Results

The performance of the refined MorphDAG implementation (”MorphDAG (Updated)”) was benchmarked against the original MorphDAG. The evaluation focused on two key metrics: scalability with increasing transaction volume and performance stability across different read/write workload ratios. The results, averaged over five iterations, are presented below and conclusively demonstrate the effectiveness of our per-account frequency-based classification.

A. Scalability Analysis: Transaction Volume vs. Time

This test measures the time taken to process a block as the number of transactions within it increases. Figure 2 illustrates the results, comparing the processing time of the original MorphDAG against our updated version.

Primary Observation: The ”MorphDAG (Updated)” line (green) is consistently below the original ”MorphDAG” line (red). This indicates that for any given number of transactions, our refined implementation processes the block faster, resulting in lower latency.

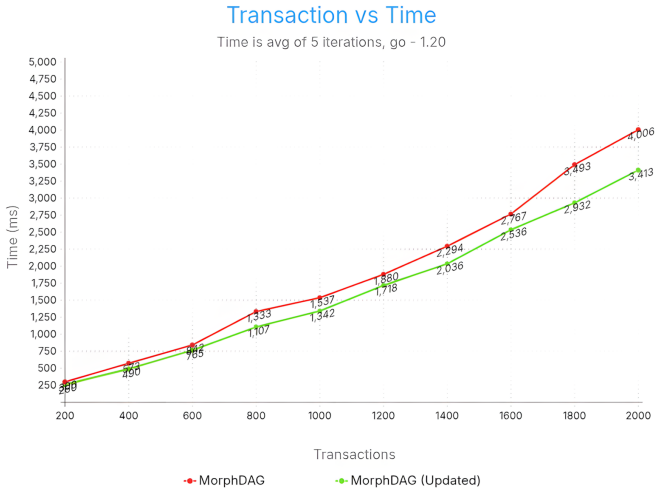

Fig. 2: Processing time vs. increasing transaction volume. Our updated version (green) shows consistently lower latency.

Quantitative Insights:

- At a load of **400 transactions**, the updated system processes the block in approximately **490ms** compared to the original’s **573ms**.
- As the load increases to **1600 transactions**, the performance gap widens. The updated version takes **2,536ms**, while the original takes **2,767ms**.

Conclusion: The updated MorphDAG not only has lower latency but also demonstrates better scalability. The widening gap between the two lines suggests that our optimization becomes even more effective as the network load increases, allowing for significantly higher throughput.

B. Workload Robustness Analysis: R/W Ratio vs. Time

This test measures the system’s performance under varying workload compositions, from read-heavy (80% Reads, 20% Writes) to write-heavy (20% Reads, 80% Writes). The results are shown in Figure 3.

Primary Observation: The ”MorphDAG (Updated)” line is consistently below the original ”MorphDAG” line across all R/W ratios, showing our system is faster regardless of the workload type.

Key Finding - Performance Stability: The original MorphDAG (green line) shows considerable performance fluctuation, with latency varying from a high of **729ms** (at 80R 20W) down to **683ms**. In contrast, our updated MorphDAG (red line) is remarkably stable, with its performance curve remaining much flatter and within a tight band of **604ms to 631ms**.

Quantitative Insights:

- The most significant improvement is seen in **read-heavy workloads (80% Read, 20% Write)**, where latency is

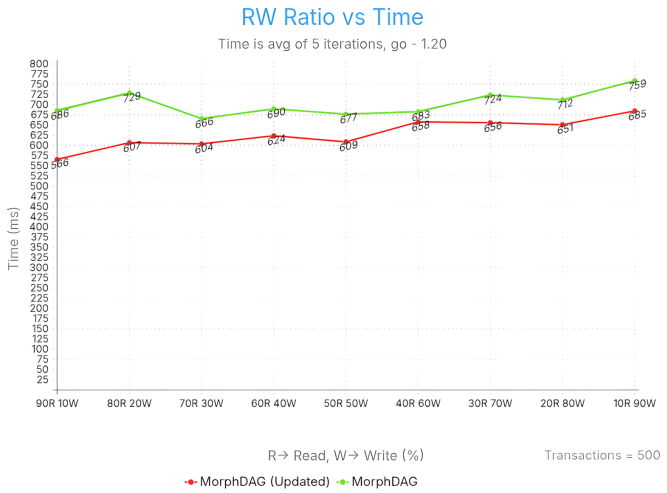

Fig. 3: Processing time vs. varying Read/Write (R/W) ratios. Our updated version (red) shows superior performance and stability.

reduced from **729ms** in the original to just **607ms** in our updated version—a reduction of over **16%**.

- Even in balanced (**50R 50W**) and write-heavy (**20R 80W**) workloads, our system is more efficient, clocking in at **609ms** and **631ms** compared to the original's **677ms** and **712ms**, respectively.

C. Verification of State Integrity

A critical aspect of our evaluation was to confirm that the performance enhancements did not compromise the transactional integrity or correctness of the ledger. The goal is to achieve faster processing without altering the final state of the blockchain.

To validate this, we inspected the output of the `stateTransfer()` function, which is responsible for applying the final state changes to the database as part of the Smallbank benchmark execution. We logged the resulting account balances and operational values after a series of transactions for both the original MorphDAG and our optimized version.

Our tests confirmed that the final state values are identical in both implementations. For example, for a given set of operations, both versions resulted in the exact same final account balances (e.g., `Balance For Read Operations: 100000`) and transaction values (e.g., `Value used in RW operations: 5`). This was achieved even while our updated system recorded lower processing times.

This result provides conclusive evidence that our refinement correctly preserves data consistency and produces deterministic outcomes. The increased parallelism achieved through per-account frequency analysis does not introduce race conditions or data loss, Maintaining integrity of the ledger.

Conclusion: The per-account frequency analysis is fundamentally more accurate than a simple operation count. It correctly identifies truly "cold" transactions even if they have many read operations, allowing for greater parallelism. This results in not only a faster system overall but one that is significantly more robust and predictable under changing application behavior.

V. Future Work

While this work demonstrates significant performance improvements, several avenues for future research remain to further enhance the MorphDAG framework. We have identified the following key directions:

- **Dynamic Threshold Adjustment:** Develop an adaptive mechanism to dynamically adjust the 'ratio' threshold for hot account classification based on real-time network load and observed conflict rates, moving beyond a static percentile.
- **Broader Workload Evaluation:** Evaluate the refined model against a wider range of real-world application workloads, such as those from Decentralized Finance (DeFi) or NFT marketplaces, to assess its effectiveness under more diverse and complex transaction patterns.
- **Large-Scale Distributed Testing:** Conduct experiments on a distributed, multi-node network to analyze the system's performance, scalability, and resilience under a more realistic heavy workload.

Acknowledgment

I Would like to thank my friend **Suryansh Rohil** (Third year CSE) for working with me on this project, I would also like to thank my mentor **Ms Sweta Kumari** for letting me work on this topic I have learned a lot from this opportunit, I would also like to thank the OUR team at SNU for giving me the wonderful opportunity to take up this research in my second year.

References

[1] P. S. Anjana, S. Kumari, S. Peri, S. Rathor and A. Somani, "An Efficient Framework for Optimistic Concurrent Execution of Smart Contracts," 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pavia, Italy, 2019, pp. 83-92

[2] Fu, Xiang et al. "Jointgraph: A DAG-based efficient consensus algorithm for consortium blockchains." Software: Practice and Experience 51 (2019): 1987 - 1999.

[3] H. Hellani, L. Sliman, A. E. Samhat and E. Exposito, "Tangle the Blockchain:Towards Connecting Blockchain and DAG," 2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Bayonne, France, 2021, pp. 63-68

[4] H. Pervez, M. Muneeb, M. U. Irfan and I. U. Haq, "A Comparative Analysis of DAG-Based Blockchain Architectures," 2018 12th International Conference on Open Source Systems and Technologies (ICOSST), Lahore, Pakistan, 2018, pp. 27-34

[5] F. M. Benčić and I. Podnar Žarko, "Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph," 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2018, pp. 1569-1570

[6] Kotilevets, I. D. et al. "Implementation of directed acyclic graph in blockchain network to improve security and speed of transactions." IFAC-PapersOnLine 51 (2018): 693-696.

[7] Tokhmetov, A., Lee, V. ., and Tanchenko, L. . (2023). DEVELOPMENT OF DAG BLOCKCHAIN MODEL. Scientific Journal of Astana IT University, 16(16).

[8] Qin Wang, Jiangshan Yu, Shiping Chen, and Yang Xiang. 2023. SoK: DAG-based Blockchain Systems. ACM Comput. Surv. 55, 12, Article 261 (December 2023), 38 pages.

[9] Dai, Xiaohai and Wang, Guanxiong and Xiao, Jiang and Guo, Zhengxuan and Hao, Rui and Xie, Xia and Jin, Hai. (2024). LightDAG: A Low-latency DAG-based BFT Consensus through Lightweight Broadcast. 998-1008. 10.1109/IPDPS57955.2024.00093.

[10] S. Zhang et al., "MorphDAG: A Workload-Aware Elastic DAG-Based Blockchain," in IEEE Transactions on Knowledge and Data Engineering, vol. 36, no. 10, pp. 5249-5264, Oct. 2024

[11] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference (EuroSys '18). Association for Computing Machinery, New York, NY, USA, Article 30, 1–15. https://doi.org/10.1145/3190508.3190538

[12] (2009). Bitcoin: A Peer-to-Peer Electronic Cash System.

[13] Wang, Tianyu, et al. "Understanding characteristics and system implications of DAG-based blockchain in IoT environments." IEEE Internet of Things Journal 9.16 (2021): 14478-14489.

[14] Park, Seongjoon, Seounghwan Oh, and Hwangnam Kim. "Performance analysis of DAG-based cryptocurrency." In 2019 IEEE International Conference on Communications workshops (ICC workshops), pp. 1-6. IEEE, 2019.

[15] Yang, Wenhui, Xiaohai Dai, Jiang Xiao, and Hai Jin. "LDV: A lightweight DAG-based blockchain for vehicular social networks." IEEE Transactions on Vehicular Technology 69, no. 6 (2020): 5749-5759.