Asynchronous Blockchain Recording for Chain-of-Thought Tracing in Small Language Models

Sungmoon Park
Department of Healthcare IT
Inje University
Gimhae, Republic of Korea
qkrtjdans55@oasis.inje.ac.kr

Seowan Kim
Department of Healthcare IT
Inje University
Gimhae, Republic of Korea
gonstoro0521@oasis.inje.ac.kr

Hoansuk Choi
Gyeongnam Intelligence Innovation Center
Kyungnam University
Changwon, Republic of Korea
chs2024@kyungnam.ac.kr

Nam-Hyun Yoo
Gyeongnam Intelligence Innovation Center
Kyungnam University
Changwon, Republic of Korea
hyun43@kyungnam.ac.kr

Jinhong Yang
Department of Healthcare IT
Inje University
Gimhae, Republic of Korea
jinhong@inje.ac.kr
(Corresponding author)

***Abstract*—Small Language Models (sLMs) require transparency in their reasoning processes for deployment in critical domains. This paper presents an asynchronous blockchain recording system for tracking Chain-of-Thought (CoT) reasoning in sLMs. We implement a system using the Qwen2.5-1.5B model integrated with Hyperledger Fabric v2.5, introducing a hybrid asynchronous architecture with Write-Ahead Logging (WAL) to ensure data durability. Through experiments using an unbalanced design (N=30 for baseline and proposed method, N=6 for synchronous comparison), we measure the performance impact of blockchain integration with full disk persistence. The proposed Hybrid-BC mode achieves 11.1% latency overhead compared to the baseline, while providing crash recovery guarantees through WAL. In contrast, the synchronous mode results in 666.7% overhead ($p < 0.001$). The system validates that disk I/O costs for durability (574ms) are an order of magnitude smaller than blockchain consensus latency (10,404ms), effectively decoupling inference from network delays.**

***Index Terms*—small language model, chain-of-thought, blockchain, Hyperledger Fabric, asynchronous processing, traceability, durability**

I. Introduction

The deployment of Small Language Models (sLMs) in edge computing environments has increased due to their computational efficiency and privacy benefits. Models with 1.5 to 7 billion parameters, such as Qwen2.5 and Phi-3, achieve reasoning capabilities comparable to larger models on benchmarks like GSM8K [1]. However, a comprehensive evaluation revealed that 47.6% of sLMs are vulnerable to jailbreak attacks, necessitating robust audit trails for their Chain-of-Thought (CoT) reasoning processes [2].

Blockchain technology offers immutable logging for AI transparency. The EU AI Act mandates that high-risk AI systems "shall technically allow for the automatic recording of events (logs) over the lifetime of the system" to ensure traceability [3]. However, synchronous blockchain integration introduces prohibitive latency for real-time interactive AI. Research indicates that synchronous blockchain execution introduces latency overheads that increase with network complexity, potentially limiting scalability [4]. In high-frequency CoT generation, where a single prompt generates dozens of reasoning steps, this synchronous overhead accumulates and degrades user experience.

Furthermore, permissioned blockchains like Hyperledger Fabric face Multi-Version Concurrency Control (MVCC) conflicts in high-throughput scenarios [5]. Simply streaming CoT steps to the ledger results in transaction failures due to these conflicts.

This paper proposes a Hybrid Asynchronous Blockchain (Hybrid-BC) recording system that addresses these barriers. Inspired by FastFabric [6], we implement a parallelized submission mechanism that decouples AI inference from blockchain consensus, while incorporating Write-Ahead Logging (WAL) to guarantee data durability against system failures. Our contributions are:

- A Hybrid-BC tracing system achieving 11.1% overhead with full disk persistence, compared to 666.7% for the synchronous approach
- Statistical validation through 30 repeated experiments, confirming that WAL I/O overhead accounts for 8.57% of total inference time
- A case study demonstrating practical error backtracking via blockchain trace queries

II. Related Work

A. On-Device sLM Capabilities and Trustworthiness

Recent advancements enable language models to run locally on mobile hardware. Abdin et al. demonstrated that Phi-3-mini

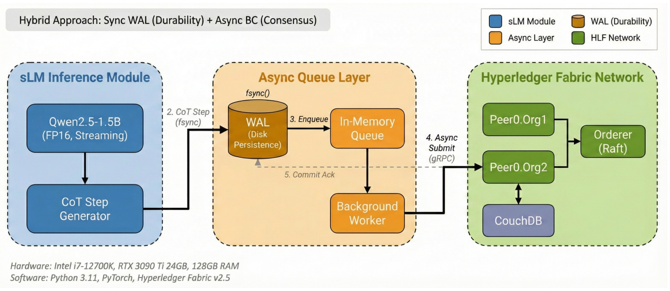

Fig. 1. Proposed Hybrid Asynchronous Blockchain recording system architecture. The async queue layer employs a non-blocking submission mechanism with Write-Ahead Logging (WAL) to ensure durability before decoupling inference latency from blockchain consensus delays.

(3.8B parameters) achieves 82.5% on GSM8K, surpassing GPT-3.5 [7]. However, Nakka et al. found that quantized models are less trustworthy than server-based counterparts, with optimization for mobile deployment exacerbating bias and privacy breaches [8]. This trust gap necessitates external audit mechanisms.

B. *Blockchain Architectures for AI Integrity*

The BC4LLM framework conceptualizes blockchain for recording training processes [9]. Federated TrustChain integrates Hyperledger Fabric with federated learning [10]. However, these approaches focus on static artifacts rather than dynamic reasoning steps during inference. Systems using off-chain storage introduce variable latency that renders real-time CoT tracking impractical [11]. More advanced cryptographic approaches, such as VerifyNet [12], impose higher overheads for proof generation, making them unsuitable for interactive sLM applications.

C. *Hyperledger Fabric Optimization*

Thakkar et al. identified state validation and MVCC checks as primary bottlenecks [13]. Approaches like Fabric++ build conflict graphs during ordering [14], while EMVCC implements local caching [15]. Our work adopts a lightweight asynchronous pattern inspired by FastFabric [6], eliminating the need for complex reordering mechanisms.

D. *Blockchain vs. Centralized Logging*

While centralized logging systems such as Elasticsearch or cloud-based solutions provide lower latency (typically 5-10ms), they lack several critical properties required for audit trails in regulated domains. Centralized mechanisms are vulnerable to threats where logs "secured improperly in storage or in transit might also be susceptible to intentional and unintentional alteration and destruction" [16]. Blockchain provides Byzantine fault tolerance, cryptographic non-repudiation, and decentralized trust without single points of failure. Furthermore, the eIDAS 2.0 regulation establishes that data records in a qualified electronic ledger "shall enjoy the presumption of their unique and accurate sequential chronological ordering and of their integrity" [17], providing a stronger legal presumption of integrity than centralized logs. These properties are essential for compliance with regulations such as GDPR Article 32, which requires demonstrable data integrity. The 11.1% overhead represents an acceptable trade-off for applications requiring these guarantees.

III. System Architecture

The system consists of three components: the sLM inference module, the async queue layer with WAL, and the Hyperledger Fabric network, as shown in Fig. 1.

The inference module uses the Qwen2.5-1.5B-Instruct model loaded with half-precision (FP16). Each inference session generates multiple CoT steps stored independently in the blockchain.

The async queue layer implements a non-blocking asynchronous submission mechanism. To address the data loss risks of purely in-memory queues, we implement a **Hybrid-BC approach**:

1) **Write-Ahead Logging (WAL):** Before submission, each CoT step is synchronously appended to a local log file using `os.fsync()` to ensure physical disk persistence.
2) **Asynchronous Submission:** Once persisted, the transaction is added to an in-memory queue for background processing by the blockchain client.

This design ensures that even if the system crashes before blockchain commitment, the reasoning steps can be recovered from the WAL, providing step-level durability with minimal latency impact.

The Hyperledger Fabric network comprises two organizations with two peers each, using Raft consensus with a single orderer. Block generation occurs every 2 seconds or upon reaching 10 transactions.

Chaincode Design: Each trace is stored independently with the following structure:

```
{
  "inference_id": "UUID",
  "step_sequence": 1,
  "content": "reasoning step text",
  "model_id": "Qwen2.5-1.5B",
  "timestamp": "ISO8601"
}
```

This schema eliminates MVCC conflicts by avoiding shared state updates during trace creation.

IV. Experimental Methodology

A. *Hardware and Software Environment*

Experiments were conducted on:

- CPU: Intel Core i7-12700K (12th Gen)
- GPU: NVIDIA GeForce RTX 3090 Ti (24GB VRAM)
- RAM: 128GB DDR4
- OS: Windows 11 Home
- Software: Python 3.11, PyTorch, Hyperledger Fabric v2.5

B. Experimental Design

We utilized an unbalanced experimental design to optimize resource usage while maintaining statistical rigor:

- **Baseline & Hybrid-BC:** $N = 30$ runs each (satisfying Central Limit Theorem requirements).
- **Sync-BC:** $N = 6$ runs. Due to the prohibitive latency of the synchronous mode (mean 58.76s per run), a smaller sample size was sufficient to demonstrate the statistically significant performance gap ($p < 0.001$).

Protocol:

- **Warm-up:** 1 iteration discarded before data collection.
- **Randomization:** Mode execution order shuffled per iteration.
- **GPU Memory Cleanup:** `torch.cuda.empty_cache()` called between modes to prevent interference.
- **Inter-mode Interval:** 1 second delay between modes; 2 second delay between runs.

Ten questions requiring multi-step reasoning were used, processed cyclically across runs.

Three modes were compared:

- **Baseline:** No blockchain tracing.
- **Hybrid-BC:** Asynchronous submission with WAL durability (Proposed).
- **Sync-BC:** Waits for block commitment for each step.

C. Evaluation Metrics

Latency Overhead:

$$\text{Overhead} = \frac{T_{mode} - T_{baseline}}{T_{baseline}} \times 100\% \qquad (1)$$

WAL Analysis: Separate measurement of disk I/O time (`fsync`) versus blockchain network latency.

Throughput: Tokens per second (TPS) measured for baseline mode.

V. Results

A. Performance Comparison

Table I presents the latency measurements across synchronization modes.

TABLE I
PERFORMANCE COMPARISON ACROSS SYNCHRONIZATION MODES

Mode	N	Mean (s)	Std (s)	Overhead
Baseline	30	7.66	3.67	-
Hybrid-BC	30	8.52	3.92	+11.1%
Sync-BC	6	58.76	37.54	+666.7%

The Hybrid-BC mode achieves an overhead of 11.1%, which represents an acceptable trade-off for the added value of traceability and durability. In contrast, the Sync-BC mode increases latency by 666.7%, rendering it impractical for real-time applications. The difference between Hybrid-BC and Sync-BC is statistically significant ($p < 0.001$), validating the architectural advantage of decoupling consensus from inference.

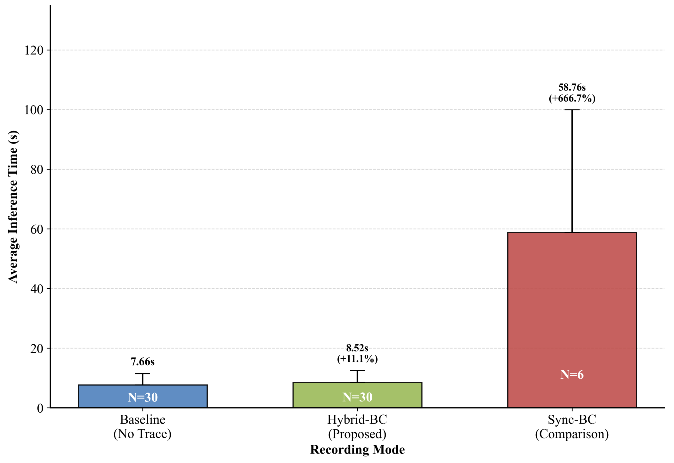

Fig. 2. Performance comparison. Hybrid-BC shows 11.1% overhead despite including full disk persistence costs, whereas Sync-BC exhibits 666.7% overhead due to network blocking.

The baseline mode achieved a mean throughput of 43.6 tokens per second (TPS) with a standard deviation of 4.5 TPS.

B. WAL Durability Analysis

To understand the cost of durability, we analyzed the specific overhead introduced by the Write-Ahead Logging.

TABLE II
WAL DURABILITY OVERHEAD ANALYSIS (HYBRID-BC)[1]

Metric	Mean	Std
WAL Write (ms)	573.9	270.2
WAL Commit (ms)	403.2	200.9
WAL Overhead Ratio (%)	8.57	3.84
BC Latency (ms)	10,403.5	4,914.1

As shown in Table II, the average time spent on disk I/O (WAL Write plus Commit marking) is approximately 977ms per inference session. This accounts for 8.57% of the total inference time when considering the baseline plus WAL overhead. The blockchain network latency (10,404ms) is approximately 10.6 times higher than the disk I/O time. This confirms that the primary bottleneck in decentralized systems is network consensus, which our asynchronous architecture effectively masks from the user.

C. Case Study: Error Backtracking

To demonstrate practical utility, we conducted a case study using a question adapted from GSM8K-style arithmetic reasoning. The blockchain stored 52 reasoning steps with Inference ID `103d80fb...`. Querying the blockchain identified the error at Step 23 (“3 eggs/morning x 8 mornings/day”), allowing for precise root cause analysis of the hallucination.

[1]WAL Overhead Ratio represents the percentage of WAL operations time relative to the baseline inference time, calculated as: $(WAL_{write} + WAL_{commit})/(T_{baseline} + WAL_{total}) \times 100\%$

D. Storage Efficiency

Analysis of blockchain storage reveals:

- Average storage per step: 0.21 KB
- Total storage (N=30, Hybrid-BC): approximately 123 KB

For deployment at 1,000 inferences per day (15 steps average): 3.15 MB/day, 1.13 GB/year.

VI. Discussion

A. Durability and Performance Trade-off

A key critique of asynchronous systems is the potential for data loss during crashes. Our Hybrid-BC approach addresses this by enforcing synchronous disk writes (WAL) before adding tasks to the memory queue. Our results show that this durability mechanism costs approximately 11% in additional latency. Given that this mechanism protects against process failures while maintaining 88.9% of the baseline speed, it represents an acceptable trade-off for sLM deployments requiring audit capabilities.

B. Clarification on Asynchronous Performance

A potential concern is whether the performance advantage of Hybrid-BC arises merely from deferring blockchain operations to the background. This section clarifies the architectural rationale.

The WAL mechanism ensures that durability is achieved at the time of commit, not deferred. The ARIES protocol established that transaction durability requires log records to be written to stable storage before returning acknowledgment to the user [18]. In our implementation, each CoT step is synchronously written to disk using `os.fsync()`. This ensures immediate durability by forcing modified data to persistent storage before returning [19]. Only after this disk write completes is the transaction added to the asynchronous queue for blockchain submission. Therefore, even if the system crashes before blockchain commitment, all reasoning steps remain recoverable from the WAL.

The asynchronous architecture fundamentally separates user-facing latency from consensus latency. This design pattern is consistent with Hyperledger Fabric's Execute-Order-Validate model, where transaction simulation occurs before ordering [13]. Our measurements confirm that blockchain consensus latency (10,404ms) exceeds local WAL operations (977ms) by a factor of 10.6. The asynchronous design masks this consensus latency from end-users while preserving data integrity through WAL.

Regarding the experimental environment, the use of a single physical machine follows established methodology for isolating computational overhead from network variability [13]. This controlled setting enables identification of software bottlenecks that would be obscured by network noise. In distributed deployments with network latency, the performance gap between Hybrid-BC and Sync-BC would widen, as network delays affect synchronous operations but not local WAL writes.

C. The Order-of-Magnitude Principle

Our measurements reveal that the cost of local persistence (WAL) is roughly one order of magnitude smaller than the cost of remote consensus. Specifically, total WAL overhead (977ms, comprising write and commit operations) versus blockchain latency (10,404ms) yields a ratio of approximately 1:10.6. By moving the blocking operation from the network layer (Sync-BC) to the local disk layer (Hybrid-BC), we eliminate the major component of latency while retaining the integrity guarantees.

D. Scalability to Larger Models

The 11.1% overhead was measured on a 1.5B-parameter model with relatively short inference times (mean 7.66s). For larger models (7B, 70B parameters) where inference takes significantly longer, the relative impact of the fixed WAL overhead (977ms) would decrease further, likely dropping below 5%. This suggests the approach scales favorably with model complexity.

E. Limitations

The Sync-BC experiments used $N = 6$ due to time constraints imposed by the prohibitive latency. While sufficient for demonstrating the performance gap, future work should consider automated long-running experiments for more comprehensive statistical analysis of the synchronous mode.

Additionally, although the blockchain network was configured with production-like parameters (2 organizations, 4 peers, Raft consensus), all components ran on a single physical machine. This approach is methodologically valid for measuring computational overhead in isolation [13]. However, future work should validate these findings in geographically distributed deployments to quantify the impact of network latency on both synchronous and asynchronous modes.

VII. Conclusion

This paper presented a Hybrid Asynchronous Blockchain recording system for CoT tracing in sLMs. Through experiments ($N = 30$ for proposed and baseline, $N = 6$ for synchronous comparison), we demonstrated:

1) Hybrid-BC achieves 11.1% overhead with full disk durability, compared to 666.7% for the synchronous mode.
2) The analysis confirms that WAL disk I/O accounts for 8.57% of inference time, while effectively decoupling the 10-second blockchain latency from user-perceived response time.
3) Blockchain traces enable practical error backtracking and auditability with minimal storage footprint (1.13 GB/year at 1,000 daily inferences).

These results indicate that the hybrid asynchronous architecture successfully bridges the gap between the real-time requirements of sLMs and the immutability requirements of blockchain auditing.

Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP)-Innovative Human Resource Development for Local Intellectualization program grant funded by the Korea government (MSIT)(IITP-2025-RS-2024-00436773). This work was supported by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea government (Ministry of Trade, Industry and Energy) through the International Cooperation in Industrial Technology program (Project Number: P0026190).

References

[1] A. Yang et al., "Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement," *arXiv preprint arXiv:2409.12122*, 2024.

[2] W. Zhang, H. Xu, Z. Wang, Z. He, Z. Zhu, and K. Ren, "Can Small Language Models Reliably Resist Jailbreak Attacks? A Comprehensive Evaluation," *arXiv preprint arXiv:2503.06519*, 2025.

[3] European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)," *Official Journal of the European Union*, L 1689, Jul. 2024. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2024/1689/oj

[4] V. Drungilas, E. Vaičiukynas, M. Jurgelaitis, R. Butkienė, and L. Čeponienė, "Towards Blockchain-Based Federated Machine Learning: Smart Contract for Model Inference," *Appl. Sci.*, vol. 11, no. 3, p. 1010, 2021.

[5] A. Stoltidis, K. Choumas, and T. Korakis, "Performance Optimization of High-Conflict Transactions within the Hyperledger Fabric Blockchain," *arXiv preprint arXiv:2407.19732*, 2024.

[6] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, 2019, pp. 455–463.

[7] M. Abdin et al., "Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone," *arXiv preprint arXiv:2404.14219*, 2024.

[8] K. Nakka, J. Dani, and N. Saxena, "Is On-Device AI Broken and Exploitable? Assessing the Trust and Ethics in 'Small' Language Models," *arXiv preprint arXiv:2406.05364*, 2024.

[9] H. Luo, J. Luo, and A. V. Vasilakos, "BC4LLM: Trusted Artificial Intelligence When Blockchain Meets Large Language Models," *arXiv preprint arXiv:2310.06278*, 2023.

[10] X. Zuo et al., "Federated TrustChain: Blockchain-Enhanced LLM Training and Unlearning," *arXiv preprint arXiv:2406.04076*, 2024.

[11] Y. Zhang, J. Zhao, J. Jiang, Y. Zhu, L. Wang, and Y. Xiang, "Recording Behaviors of Artificial Intelligence in Blockchains," *IEEE Trans. Artif. Intell.*, vol. 4, no. 6, pp. 1437–1448, Dec. 2023.

[12] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and Verifiable Federated Learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 911–926, 2020.

[13] P. Thakkar, S. N. Nathan, and B. Viswanathan, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform," in *Proc. IEEE Int. Symp. Model. Anal. Simul. Comput. Telecommun. Syst. (MASCOTS)*, 2018, pp. 264–276.

[14] A. Sharma et al., "Blurring the Lines Between Blockchains and Database Systems: The Case of Hyperledger Fabric," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2019, pp. 105–122.

[15] H. Trabelsi and K. Zhang, "Early Detection for Multiversion Concurrency Control Conflicts in Hyperledger Fabric," *arXiv preprint arXiv:2301.06181*, 2023.

[16] K. Kent and M. Souppaya, "Guide to Computer Security Log Management," National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, Spec. Publ. 800-92, Sep. 2006. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-92

[17] European Parliament and Council of the European Union, "Regulation (EU) 2024/1183 amending Regulation (EU) No 910/2014 as regards establishing the European Digital Identity Framework," *Official Journal of the European Union*, Apr. 2024. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1183

[18] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz, "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," *ACM Trans. Database Syst.*, vol. 17, no. 1, pp. 94–162, Mar. 1992.

[19] T. S. Pillai, V. Chidambaram, R. Alagappan, S. Al-Kiswany, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "All File Systems Are Not Created Equal: On the Complexity of Crafting Crash-Consistent Applications," in *Proc. USENIX Symp. Oper. Syst. Des. Implement. (OSDI)*, 2014, pp. 433–448.