HIChain: A Hierarchical IoT Permissioned Blockchain with Edge Cloud Architecture

Tinghao Feng, Zeshun Peng(✉), Yanfeng Zhang, Xiaohua Li, Xiaomei Dong, and Ge Yu

Northeastern University, Shenyang, China
{fength,pengzeshun}@stumail.neu.edu.cn,
{zhangyf,lixiaohua,xmdong,yuge}@mail.neu.edu.cn

Abstract. The Internet of Things (IoT) is gaining popularity in smart cities, healthcare, and industry. To ensure secure and trusted data sharing, permissioned blockchains are emerging as crucial technology. When deployed in IoT scenarios, numerous edge nodes generate substantial sensor data. However, most existing blockchains are not designed for this, resulting in insufficient performance. 1) Traditional consensus protocols lack scalability, particularly when edge nodes are geographically distributed, which restricts overall performance. 2) Edge nodes receive transactions from multiple sensors at high rates, but their limited computing power makes providing or validating certificates of these transactions a bottleneck. 3) Edge nodes have limited storage capacity, making it challenging to manage the growing IoT data.

To address these issues, we propose `HIChain`, a IoT permissioned block-chain. `HIChain` adopts a three-layer architecture for better scalability in geo-deployment. To reduce computational costs, only a portion of transactions are provided with certificates, and additional measures are implemented to prevent Byzantine faults. Additionally, we use dictionary and delta encoding techniques, along with cloud storage, to reduce storage costs on edge nodes. Experimental results show that `HIChain` effectively reduces consensus overhead and storage overhead in distributed environment. Throughput reaches up to 126.49 KTPS, 120.94% higher than the baseline, with on-chain storage cost reduced by 89.9%.

Keywords: Permissioned blockchain · Consensus · Edge computing

1 Introduction

With the spread of the Internet of Things (IoT) and advancements in network connectivity technologies, IoT devices can be connected flexibly and efficiently, driving rapid progress in IoT fields including smart cities [1–3], IoT-healthcare [4], Internet of Vehicles [5] and industrial applications [6], the data generated by the sensors is used for monitoring, analysis and prediction [7]. With the development of these applications, the number of sensors and the amount of IoT data is increasing. For example, in the 85 Information IoT Platform [8], telemetry

M. Barhamgi et al. (Eds.): WISE 2024, LNCS 15438, pp. 253–266, 2025.
https://doi.org/10.1007/978-981-96-0570-5_18

data from photovoltaic sensors must support real-time access, processing, and storage for at least 50,000 devices with a total of 4 million measurement points. The daily increase of huge amount of sensor data presents significant challenges for data replication and storage.

Additionally, a critical requirement of IoT is ensuring that data are verifiable, secure, and trustworthy. For example, in supply chains, guaranteeing the authenticity and quality of products can reduce substantial intermediary costs [9]. Consequently, blockchain, as a specialized distributed ledger, provides an effective solution for conducting IoT transactions securely and verifiably. In recent years, blockchain has been rapidly applied in various IoT fields, including product traceability [10], supply chain [11], and other IoT scenarios [12,13].

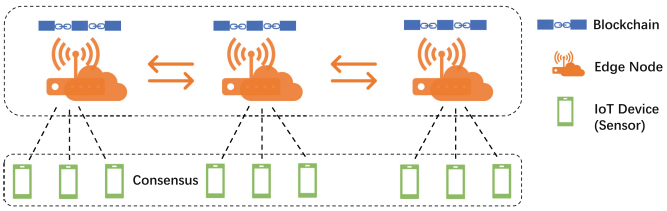


Fig. 1. Traditional IoT blockchain architecture.

However, these solutions that apply blockchains to IoT have poor performance due to the limited scalability of the underlying blockchains.

Firstly, achieving consensus on numerous widespread IoT devices leads to significant network overhead. As shown in Fig. 1, geo-distributed IoT devices (among cities and data centers) receive metric data from nearby sensors and form a block. These IoT devices then exchange the block using a consensus protocol (such as PBFT [14] or Raft [15]), and propagate the block to edge nodes. The edge nodes act as proxies to accelerate communication between IoT devices and store the entire blockchain. As the number of edge nodes and IoT devices increase, the performance of consensus protocols decreases and can be a bottleneck under high sending rates.

Moreover, the blockchain system needs a lot of computing resources to ensure security. In the traditional blockchain system, to provide the verifiability of data, the sensor signs each transaction before sending it to the edge node. Then the edge node independently verifies these data signatures locally. Signing and verifying the signature will generate a lot of calculation costs. In addition, an edge node receives a large number of transactions from multiple sensors at the same time, and it needs to verify each transaction signature, which leads to huge computational overhead. When the computing performance of the edge node is insufficient, it may become the bottleneck of the system.

In addition, due to the limited storage capacity of edge nodes, the complete storage of blockchain to edge nodes will generate huge storage pressure. In traditional permissioned blockchains that use consensus protocols such as

PBFT, nodes participating in the consensus need to save a complete copy of the blockchain. However, it is not realistic for edge nodes to store a complete blockchain. On the other hand, the blockchain will additionally store information such as block signatures and block Merkle trees for verification, further increasing the storage burden of edge nodes.

To solve the above problems, (2) we group edge nodes based on their geographical regions, where each group uses PBFT to prevent Byzantine edge nodes and IoT devices, reducing network overhead. (2) Drawing inspiration from the widespread use of batch processing, each sensor generates and signs data in batches rather than individually, reducing the signature and validation overhead on IoT devices and edge nodes. (3) To reduce the storage overhead, edge nodes store their blockchains on high-availability cloud storage, we also adopts online analytical processing (OLAP) compression [16] based on the characteristics of IoT data.

The contributions of this paper can be summarized as follows:

- We introduce a three-layer hierarchical permissioned blockchain architecture of IoT data to reduce the network overhead of edge nodes, supporting parallel consensus processing among groups.
- We design an periodic signature strategy to reduce the computational overhead of providing data signatures by IoT devices and verifying data signatures by edge nodes.
- We design a lightweight hierarchical storage architecture for IoT data to relieve the storage pressure of edge nodes and introduce OLAP compression strategy according to the characteristics of IoT data to reduce the overhead of on-chain storage and improve the utilization rate of cloud storage.

2 Related Work

Table 1 presents related works on the application of IoT blockchains. To achieve quality of service (QoS) for real-time task execution on mobile edge devices, COOPER-SCHED [17] proposes a collaborative scheduling framework. Tong et al. [18] propose a hierarchical edge-cloud architecture and a workload placement algorithm to prove performance. However, they just implement CFT(Crash Fault Tolerance), cannot guarantee the security and tamper-proof of IoT data.

To ensure the security of IoT data, Novo et al. [13] design a decentralized access control system that stores IoT data in a permissioned blockchain. Huang et al. [19] store the blockchain as a DAG to improve performance, and design a credit-based PoW algorithm to reduce the complexity of the PoW consensus protocol. However, these works do not use hierarchical consensus protocols. When a large number of IoT devices are deployed on a large scale, the network overhead between nodes can become a bottleneck.

CoopEdge+ [20] proposes a hierarchical systematic manner and a consensus algorithm SL-BFT (Secret Leader-based Byzantine Fault Tolerant) making cooperative multi-access edge computing have a superior ability to balance system throughput and defend against targeted attacks with low overheads. Blockene

Table 1. Summary of IoT blockchain systems.

System	Architecture	Consensus	Compressed	Year
COOPER-SCHED [17]	Flat	CFT	No	2019
Tong et al. [18]	Hierarchical	CFT	No	2016
Novo et al. [13]	Flat	PoW	No	2018
Huang et al. [19]	Flat	Credit-Based PoW	No	2019
CoopEdge+ [20]	Hierarchical	SL-BFT	No	2022
Blockene [21]	Hierarchical	Prioritized Gossip	No	2020
HIChain(ours)	Hierarchical	**PBFT + RAFT**	**Yes**	2024

[21] also adopt hierarchical architectures and propose the idea of Consensus by Committee, that is, not all IoT devices participate in each round of consensus, which is propagated and coordinated by the upper-layer edge nodes through the gossip protocol, further improving system performance. However, we adopt a different hierarchical design from the above systems. We also add periodic signature strategy for optimization, which greatly improved system performance.

3 System Design

3.1 Overview and Key Components

HIChain is deployed across multiple geo-distributed trust domains, where each domain encompasses multiple edge servers and IoT devices. Figure 2 shows the overall architecture of HIChain.

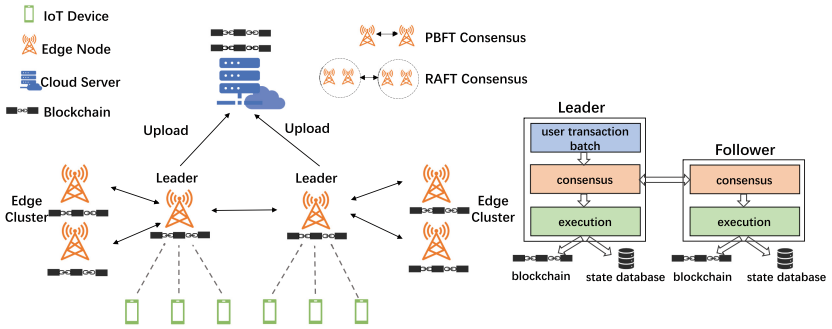


Fig. 2. Overall system architecture.

Since traditional consensus protocols incur significant network overhead when edge nodes are geo-distributed, HIChain employs a three-layer consensus architecture for processing and storing IoT data. There are three layers in HIChain: the IoT device layer, the edge server layer, and the cloud server layer.

IoT Device Layer. IoT devices are sensors that collect timestamped data, such as measurement points. IoT devices send sensor data to the closest edge nodes in the second layer. Sensor data is signed with the device's ECDSA signature to ensure authenticity and integrity.

Edge Server Layer. Edge servers are grouped into groups, receiving IoT data and establishing hierarchical Byzantine fault-tolerant (BFT) consensus across all groups. Edge nodes are responsible for receiving and processing data sent by IoT devices, verifying their signatures, and batching data into blocks for consensus. The consensus at the edge server layer consists of two parts: PBFT consensus within the group and Raft consensus between groups.

Cloud Server Layer. In the cloud server layer, cloud servers receive blocks after BFT consensus and store them in sub-chains, with each edge server group generating a sub-chain of blocks. Cloud servers are nodes with high storage capacity and computing power. They are responsible for the safe and reliable storage and provide blockchain queries for users. Additionally, sensor data storage is optimized through compression and encoding.

3.2 System Execution Flow

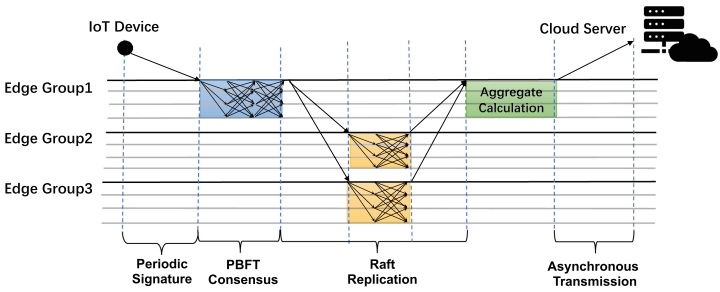


Fig. 3. System execution flow.

Figure 3 shows the execution flow of `HIChain`:

(1) IoT devices collect sensor data at fixed intervals and send it to the nearest edge group. Each device applies a digital signature to batches of data collected over a specific period, instead of signing each individual data point.

(2) Upon receiving the sensor data, the leader of the edge group performs PBFT within its group (to achieve local consensus) and Raft across all groups (to achieve global consensus). After achieving global consensus, the verified blocks are stored in the edge nodes, and their state databases are updated. These blocks are regularly transferred to the cloud server.

(3) Cloud servers receive the blocks sent by edge groups. The blocks are then encoded and compressed in the cloud server.

Algorithm 1: Local Consensus Process

```
1  RequestReceive:
   Input: IoT deivce ID i, sensor data d and signature sig(optional).
   Output: sender group Cj, sensor data d.
2  if request is signed then
3      VerifyResult = Verify(sig);
4      PBFTConsensus(Cj, d);
5      CacheData = CacheList[i];
6      for data in CacheData do
7          InsertToBlock(data);
8      InsertToBlock(d);
9      SendToGroup(Ck, block); (k ≠ j)
10 else
11     CacheList[i].add(d)
12     PBFTConsensus(Cj)
```

Local Consensus Process. We use PBFT for local consensus. Since we adopt the periodic signature strategy, not all sensor data has a signature. If sensor data lacks a signature, it is cached in a queue because it cannot be immediately verified. The edge node optimistically sends all sensor data to PBFT. Therefore, after consensus is reached, the sensor data in the block cannot be marked as valid until its signature is verified. In detail, if the current sensor data is signed, unverified sensor data (i.e., its predecessors) from the cache queue is retrieved and processed: (1) If the signature is valid, this group of data is collectively marked as valid in the block, and Raft is performed across edge server groups. (2) If the signature is invalid, the sensor data is discarded from the block.

In Algorithm 1: *RequestReceive* is the process of handling the periodic signature after the leader of the edge group receives a request; *Verify* checks that the signature is valid; *CacheList* is the cache queue for unsigned data in edge nodes; *SendToGroup* sends data to other edge groups; *PBFT consensus* includes three stages: Pre-Prepare, Prepare, and Commit.

Global Consensus Process. After the leader node of the edge group completes local PBFT consensus, it sends the block to other groups using Raft consensus, as shown in Fig. 3. Once Raft consensus is achieved, the block is considered verified and persistent, ready for upload to cloud servers. After global consensus, `HIChain` performs aggregation calculations on the sensor data in the block, computing values such as maximum, minimum, and average, which are then stored in the block header.

4 Periodic Signature Strategy

Since edge nodes and IoT devices lack sufficient computing resources to validate and sign each piece of sensor data, we propose periodic signature strategy to

reduce the total number of signatures. In the naive approach, each IoT device collects and signs a set of data instead of signing each transaction separately, then transmits it to the edge nodes. However, this increases transaction latency. In contrast, our periodic signature strategy optimistic assumes all sensor data are correct, allowing immediate transmission to edge nodes upon generation. Sensors periodically package and sign previously unsigned data, linking the signatures to ensure system security.

The periodic signature strategy we propose is inspired by batch processing techniques used to expedite transaction processing in the blockchain field. Batch processing treats a group of transactions as a single unit. Similarly, after a sensor generates a set of data, it can be signed and sent to the edge node in a batch, thereby increasing the granularity of data signatures and significantly reducing the computational overhead associated with signature verification. However, while batch transmission reduces computational overhead, it also increases the average processing delay, making it less suitable for IoT scenarios that demand low-latency data processing.

The periodic signature algorithm adopts optimistic thinking, without blocking the transmission of sensor data, generates data at a fixed acquisition frequency and sends it to the edge node for consensus immediately, but signs the data at a certain interval. At the same time, in order to prevent the transaction from being forged, in the edge group layer, the unsigned data completed by consensus will be temporarily stored in the block because the validity cannot be determined. According to its own computing power, the sensor regularly packages and signs the unsigned data it sends, and chains the signature to ensure system security. In this way, the signature overhead of sensors and the verification signature overhead of edge nodes are further reduced. This periodic signature method solves the blocking problem of batch signatures and greatly reduces the computational overhead of verifying signatures in the process of data consensus. See function *RequestReceive* in Algorithm 1 for the detailed algorithm description.

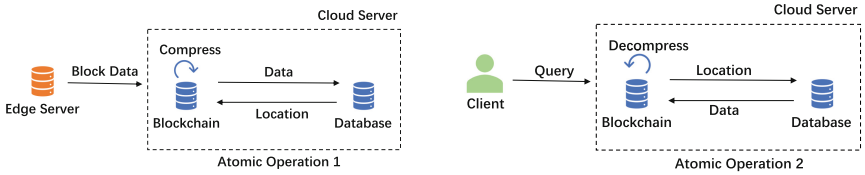
Correctness. The data received by the edge node can be categorized into two scenarios: 1) If the data is not signed, the edge node will initially perform a PBFT consensus on the data. Although the validity of the data cannot be immediately ascertained, it will be temporarily cached in the edge node. The validity of this data will be determined based on the validity of subsequent data signatures from the same device. If no signed data from the same device is received before the edge node synchronizes with the cloud server, the current cache will be cleared, and synchronization with the cloud server will not occur; 2) If the data is signed, its consensus process aligns with the existing blockchain system. The validity of the data is directly established through PBFT consensus. Additionally, data from the same device that is present in the cache will be retrieved and assigned the same validity.

5 Storage Design

5.1 Trusted Storage Process

The cloud server layer combines the blockchain technology with the database system. The blockchain stores the index and hash summary of data, and stores complete information such as data, timestamp and signature in the database. The blockchain ensures the verifiability of data storage and query, but users are not allowed to directly query the database, and users can only access data through the query interface of the blockchain system.

In addition, the cloud server will compress the data regularly to improve the utilization rate of cloud storage. The specific compression strategy is described in detail in the Sect. 5.2. Each edge group corresponds to a different table in the database. During data storage, the storage location will be recorded and then returned to the blockchain system. When the user inquires, the compressed data will be temporarily restored to complete data, and the storage location and hash summary will be obtained. This method greatly reduces the storage pressure on the chain of the system.



(a) Store Operation. (b) Query Operation.

Fig. 4. Operations in cloud server.

Security Assurance. Figure 4 shows the process of the system when the system stores data and users query. If the storage and query operations of the on-chain and off-chain separate, the database operation process presents security challenges, and if there are security problems during the execution of the database off-chain, the overall security of the system cannot be assured.

The database in `HIChain` only supports the remote call of blockchain, provides query interface and storage interface, and does not provide deletion service and modification service. The query interface of the database provides the hash summary of the query data when it returns the data, and the data integrity can be verified by comparing it with the hash summary stored in the chain. `HIChain` considers that the storage and query process needs to be an atomic process, that is, the process from complete block data to data compression and storage in the database is a complete transaction, which ensures the atomicity of the server storage process; Similarly, when querying data from the database, it is necessary

to restore the original data according to the differences, calculate the hash digest and return it. This process is also structured as a complete transaction to ensure the atomicity of the user's query operation.

5.2 Block Storage Compression

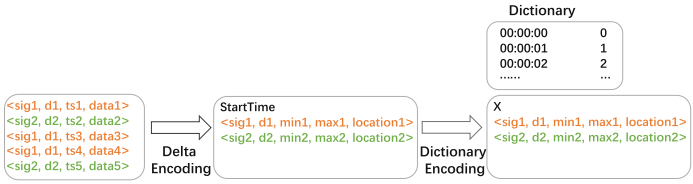


Fig. 5. On-chain storage compression.

When the data is directly stored in the blockchain, each record will contain the signature, device ID, timestamp and data in the interval signature group. In addition, data such as hash digest need to be stored in the blockchain. Due to the huge amount of data generated by IoT devices in the scene, it will consume a lot of additional storage space.

Given the application of interval signature technology to data, it is logical to compress and store data groups with the same correctness and interval signature, and the data timestamp is increasing in the IoT scene, so this paper considers introducing OLAP [16] dictionary encoding and delta encoding technology to compress data, which can significantly save storage space. The data compression process is shown in Fig. 5.

Dictionary Encoding. Dictionary encoding is commonly used in databases like Oracle [22] to optimize storage and improve query performance. The idea is to replace repetitive data values with shorter codes, reducing the space to store the data. This technique is particularly effective for columns with many repeated values.

Delta Encoding. Delta encoding reduces storage space by storing the differences between data. This method is particularly effective for handling continuous data or time-series data, as the differences between consecutive data items are often small. By storing differences, the required storage space is typically smaller

Because each block is generated in sequence, the timestamp in each block is not necessarily strictly increasing, but the generation time of the block must be increasing, so delta encoding technology is applied to encode the data according to the increasing nature of the timestamp. As shown in Fig. 5, we designs that each block stores a *StartTime*, and each piece of data in the block does not store

a complete timestamp, but only stores the deviation value between the current data and the start time. Taking every interval signature group as a unit, *Min* represents the minimum deviation value of this group of data, *Max* represents the maximum deviation value of this group of data, and *Location* represents the storage location of this group of data in the database, that is, the table and index where the data is located, and the data of a signature group is compressed into one piece of data.

In addition, because a large number of IoT devices generate data at the same time, there will be a large number of data with the same initial data. Aiming at this kind of repeated data, we adopts dictionary encoding technology, that is, the starting time value is stored in the dictionary, and the starting time in the block does not take the minimum time of the whole block, but takes a fixed time. As shown in Fig. 5, the server adds a mapping time to the dictionary every second, that is, a time value is mapped into a shorter integer. Taking a fixed time interval can ensure the efficiency of dictionary encoding, that is, each time can be mapped many times to avoid too many elements in the dictionary, and further compress the block data through this dictionary encoding technology.

When users query data from the cloud server, the server will restore the compressed data to the original data. The data restoration process is the reverse process of compressed storage. First, the start time value is restored according to the dictionary, then the data is restored to the original time according to the timestamp difference, and the signatures are copied to each piece of data in the interval signature group in turn, which is restored to the original block data.

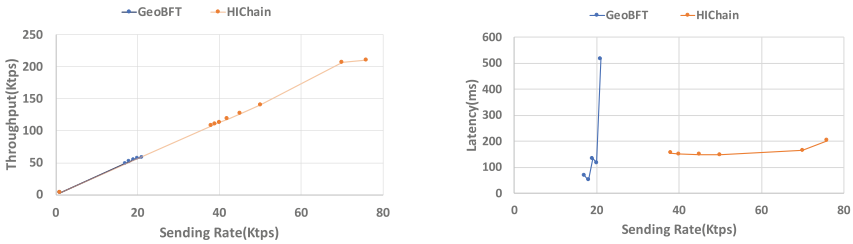
6 Experimental Results and Analysis

We implement a permissioned blockchain prototype of `HIChain` using C++, and use ResilientDB [23] as the baseline. In the Alibaba Cloud [24] distributed experimental environment, we verify the effectiveness of `HIChain`.

6.1 Experimental Setup

The experiment is based on the YCSB benchmark [25], using YCSB-A (read /write = 0.5/0.5) and YCSB-B (read/write = 0.95/0.05). We also measure the impact of storage compression on block data to verify the performance of `HIChain` storage compression design.

The experimental configuration is as follows: 12 Alibaba Cloud ECS c6 instances (8-core CPU, 16GB RAM), running Ubuntu 22.04. The environment includes 4 nodes in Chengdu (Western China), 4 nodes in Zhangjiakou (Northern China), and 4 nodes in Shenzhen (Southern China), with 1 client in each region. The network between nodes across data centers is 100 Mbps, while nodes within a data center have 10 Gbps. The RTT between any two nodes is 33.5ms, 36.1ms, and 38.6ms respectively. We set up experiments with signature intervals ranging from 0 to 4. ResilientDB [23] uses GeoBFT as a baseline for comparison.



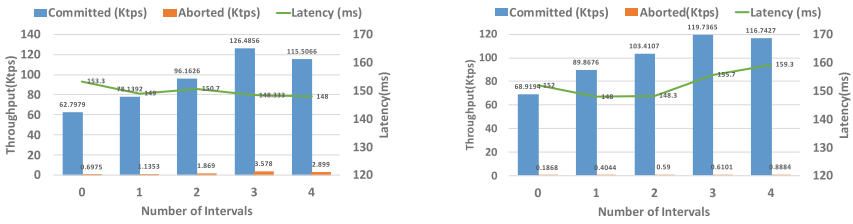
(a) Throughput. (b) Latency.

Fig. 6. System performance.

6.2 Overall Performance

As shown in Fig. 6a and Fig. 6b, we measured system performance by adjusting different client sending rates. In YCSB-A, GeoBFT achieves the highest throughput of 57.25 KTPS when the sending rate in each region is 20k (60K total), with a transaction abort rate of 0.933% and latency of 117.3 ms. `HIChain` achieves the highest throughput of 206.63 KTPS with a latency of 164.33 ms when the sending rate in each region is 70k (210K total). Since the latency begins to rise sharply beyond this point, the throughput at an 80k sending rate is not considered the highest value.

6.3 Effects of Periodic Signature Strategy



(a) YCSB-A workload. (b) YCSB-B workload.

Fig. 7. Effects of periodic signature strategy.

As shown in Fig. 7a and Fig. 7b, the throughput of `HIChain` increases proportionally with the number of signature intervals until it reaches stability or decreases. In this experiment, under the YCSB-A workload, when the number of signature intervals is 3, the peak throughput of `HIChain` is about 126.49 KTPS, while the GeoBFT throughput is about 57.25 KTPS, indicating an improvement of 120.94%; under the YCSB-B workload, when the number of signature intervals

is 3, the peak throughput of `HIChain` is about 119.7 KTPS, while the GeoBFT throughput is about 59.8 KTPS, indicating an improvement of 100.1%. When the number of signature intervals is 3 and 4, the performance is very close, because the performance of `HIChain` theoretically increases linearly until the sending rate of the YCSB client becomes a bottleneck. As the number of signature intervals increases, the processing latency of `HIChain` for both workloads remains relatively stable, fluctuating between 117.3 and 159.3 ms.

The experimental results demonstrate that the hierarchical architecture and periodic signature strategy we propose significantly enhance system performance under both workloads. The periodic signature strategy greatly reduces the computational cost of the consensus process, effectively alleviating the performance bottleneck of the blockchain system during the consensus stage.

6.4 Storage Compression Performance

To verify the effect of the storage compression algorithm we propose, the experiment is conducted on the transformer dataset ETT [26], which is commonly used in the field of data analysis and prediction. The selected dataset has a total of 17,421 sensor data, and each sensor data row has 7 columns of sensor data values. The storage compression effect is related to the number of intervals between the signatures. We conducted simulation experiments with signature intervals of 1, 3, 5, and 10. Specifically, when the interval is set to 1, every 2 rows are compressed into a single row, and the columns are compressed using OLAP data compression techniques (Fig. 8).

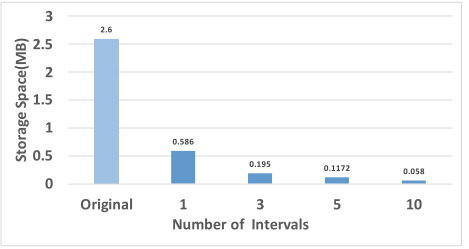


Fig. 8. On-chain storage compression effect.

Experimental results show that the storage compression strategy effectively reduces the storage space required for sensor data in blocks. The compression effect increases linearly with the number of periodic signatures.

As the number of signature interval increases, more sensor data is compressed into a single row, thereby naturally reducing storage space. However, this also means that more users will need to query the storage location of this compressed row, concentrating their queries on this group of compressed data. The number of decompressions during queries can be reduced through caching and other means.

Therefore, for data storage, the greater the number of intervals, the better the compression effect.

7 Conclusion

We design an edge cloud collaborative architecture for IoT data based on permissioned blockchain, addressing the performance challenges posed by the limited computational and storage capabilities of edge nodes in real-world IoT scenarios. A three-layer system architecture is employed to mitigate the network overhead resulting from the geographical distribution of edge nodes and IoT devices. The proposed periodic signature strategy reduces the computational complexity of traditional consensus protocols. Additionally, OLAP encoding is introduced in the cloud server to compress storage based on `HIChain` data characteristics. During distributed experimentation, latency, throughput, and compression effect of `HIChain` met the expected performance metrics. Future work will focus on designing and optimizing efficient query mechanisms for IoT data to further enhance practical value.

References

1. Xiaolong, X., Liu, X., Zhanyang, X., Dai, F., Zhang, X., Qi, L.: Trust-oriented IoT service placement for smart cities in edge computing. IEEE Internet Things J. **7**(5), 4084–4091 (2019)
2. Wang, F., et al.: Privacy-aware traffic flow prediction based on multi-party sensor data with zero trust in smart city. ACM Trans. Internet Technol. **23**(3), 1–19 (2023)
3. Wu, H., Chen, J., Nguyen, T.N., Tang, H.: Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems. IEEE Trans. Ind. Inf. **19**(2), 2117–2128 (2022)
4. Rathee, G., Sharma, A., Saini, H., Kumar, R., Iqbal, R.: A hybrid framework for multimedia data processing in IoT-healthcare using blockchain technology. Multimedia Tools Appl. **79**(15), 9711–9733 (2020)
5. Ulm, G., Smith, S., Nilsson, A., Gustavsson, E., Jirstrand, M.: OODIDA: on-board/off-board distributed real-time data analytics for connected vehicles. Data Sci. Eng. **6**, 102–117 (2021)
6. Lane Thames and Dirk Schaefer. Cybersecurity for industry 4.0. Springer (2017)
7. Silvia Liberata Ullo and Ganesh Ram Sinha: Advances in smart environment monitoring systems using IoT and sensors. Sensors **20**(11), 3113 (2020)
8. TDengine. https://www.taosdata.com/tdengine-user-cases/
9. Azizi, N., Malekzadeh, H., Akhavan, P., Haass, O., Saremi, S., Mirjalili, S.: IoT-blockchain: harnessing the power of Internet of Thing and blockchain for smart supply chain. Sensors **21**(18), 6048 (2021)
10. Jain, D., Dash, M.K., Kumar, A., Luthra, S.: How is blockchain used in marketing: a review and research agenda. Int. J. Inf. Manag. Data Insights **1**(2), 100044 (2021)
11. Dutta, P., Choi, T.-M., Somani, S., Butala, R.: Blockchain technology in supply chain operations: applications, challenges and research opportunities. Transp. Res. Part E: Logistics Transp. Rev. **142**, 102067 (2020)

12. Wang, Q., Zhu, X., Ni, Y., Li, G., Zhu, H.: Blockchain for the IoT and industrial IoT: a review. Internet Things **10**, 100081 (2020)
13. Novo, O.: Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Internet Things J. **5**(2), 1184–1195 (2018)
14. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OsDI, pp. 173–186 (1999)
15. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: 2014 USENIX Annual Technical Conference (USENIX ATC 14), pp. 305–319 (2014)
16. Fang, W., He, B., Luo, Q.: Database compression on graphics processors. Proc. VLDB Endowment **3**(1–2), 670–680 (2010)
17. Liu, C., Li, K., Liang, J., Li, K.: COOPER-SCHED: a cooperative scheduling framework for mobile edge computing with expected deadline guarantee. IEEE Trans. Parallel Distrib. Syst. (1), 1–1 (2019)
18. Tong, L., Li, Y., Gao, W.: A hierarchical edge cloud architecture for mobile computing. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9. IEEE (2016)
19. Huang, J., Kong, L., Chen, G., Min-You, W., Liu, X., Zeng, P.: Towards secure industrial IoT: blockchain system with credit-based consensus mechanism. IEEE Trans. Industr. Inf. **15**(6), 3680–3689 (2019)
20. Yuan, L., et al.: CoopEdge+: enabling decentralized, secure and cooperative multi-access edge computing based on blockchain. IEEE Trans. Parallel Distrib. Syst. **34**(3), 894–908 (2022)
21. Satija, S., et al.: Blockene: a high-throughput blockchain over mobile devices. In: 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), pp. 567–582 (2020)
22. Pöss, M., Potapov, D.: Data compression in oracle. In: Proceedings 2003 VLDB Conference, pp. 937–947. Elsevier (2003)
23. Gupta, S., Rahnama, S., Hellings, J., Sadoghi, M.: ResilientDB: global scale resilient blockchain fabric. arXiv preprint arXiv:2002.00160 (2020)
24. Alibaba cloud: cloud computing services (2023). https://www.alibabacloud.com/
25. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 143–154 (2010)
26. Zhou, H., et al.: Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 11106–11115 (2021)