Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.038330
Article

Real-Time Memory Data Optimization Mechanism of Edge IoT Agent

Shen Guo*, Wanxing Sheng, Shuaitao Bai, Jichuan Zhang and Peng Wang

China Electric Power Research Institute, Beijing, 100192, China
*Corresponding Author: Shen Guo. Email: 1119312172@qq.com
Received: 12 December 2022; Accepted: 24 February 2023

Abstract: With the full development of disk-resident databases (DRDB) in recent years, it is widely used in business and transactional applications. In long-term use, some problems of disk databases are gradually exposed. For applications with high real-time requirements, the performance of using disk database is not satisfactory. In the context of the booming development of the Internet of things, domestic real-time databases have also gradually developed. Still, most of them only support the storage, processing, and analysis of data values with fewer data types, which can not fully meet the current industrial process control system data types, complex sources, fast update speed, and other needs. Facing the business needs of efficient data collection and storage of the Internet of things, this paper optimizes the transaction processing efficiency and data storage performance of the memory database, constructs a lightweight real-time memory database transaction processing and data storage model, realizes a lightweight real-time memory database transaction processing and data storage model, and improves the reliability and efficiency of the database. Through simulation, we proved that the cache hit rate of the cache replacement algorithm proposed in this paper is higher than the traditional LRU (Least Recently Used) algorithm. Using the cache replacement algorithm proposed in this paper can improve the performance of the system cache.

Keywords: Disk resident database; real-time database; main memory database; internet of things; industrial process control

1 Introduction

With the popularity of computer applications in recent years, disk-resident database (DRDB) has been fully developed and widely used in business and transactional applications [1–6]. In long-term use, some problems of disk databases are gradually exposed. For applications with high real-time requirements, the performance of using a disk database is not satisfactory [7]. The main reason for these problems is that in the case of high real-time nature, a large number of transactions will be established at the same time [8–12]. These transactions interact with the disk data, and there will be a lot of input and output operations. With the continuous development of modern science and technology, the integration of memory chips has been continuously improved [13], and the main memory capacity

of computers has been significantly improved [14]. While the current computer system has a large main memory capacity, its cost has been continuously reduced [15]. Based on these conditions, putting the data operations in the database into memory has the value of realization, and the main memory database (MMDB) is born with the trend [16].

At the same time, the Internet of things has become a new economic industry and application ecology, and it is also an economic and technological reform that promotes industrial upgrading and transformation using Internet information technology and intelligent technology [17–22]. The Internet of things is inseparable from data, and relying only on the traditional relational memory database can not meet the needs of the storage, processing, and analysis of massive data generated in the process of industrial production control, as well as the high timeliness of data [23–25]. Therefore, as a new branch of database system development, real-time database (RTDB) technology is applied in the industrial field. RTDB is a new database produced by combining real-time processing technology and database technology [26–28]. It combines real-time tasks and traditional database transactions and takes real-time as its prominent feature. However, most RTDBS still have defects in the efficiency of transaction processing and data storage [29].

Therefore, this paper aims at the business needs of efficient data collection and storage of the Internet of things. We take the transaction pre-analysis method based on function substitution to analyze a transaction in advance and take corresponding measures to ensure its normal completion, improve the real-time performance of transaction processing, and use the concurrency control technology based on transaction clustering to alleviate the performance bottle and ensure the accuracy of concurrency control. Our method combines the memory database recovery mechanism of replica snapshot and writeable log file to maintain the integrity of the memory database. At the same time, we have built a real-time and efficient data storage model for data collection of the Internet of Things to achieve real-time and efficient storage of data collected by the Internet of Things. Our approach optimizes the transaction processing efficiency and data storage performance of the memory database, realizes a lightweight real-time memory database transaction processing and data storage model, and improves the reliability and efficiency of the database.

The rest of the paper is organized as follows. In Section 2 we introduce the related work. Section 3 introduces the real-time memory data management model of edge IoT agents. In Section 4 we introduce our real-time memory data optimization mechanism. Section 5 is the simulation and we conclude in Section 6.

2 Related Work

Currently, the real-time database has been used in industrial scenarios and distributed systems. Reference [30] proposes an architecture of real-time database in the distributed system, which can handle the service implementation of different types of data in distributed system without overloading the performance of the machine. At the same time, a distributed architecture based on Petri nets is proposed, which has flexible data processing methods and can avoid deadlock. However, the real-time database is based on the cloud platform, and will face unbearable delays in the actual application process.

Real-time database management focuses on the timeliness of user information transmission and the accuracy of real-time data, while avoiding potential hot issues. Reference [31] proposes a QoS management scheme considering the timeliness and reliability maintenance of thermal constraints. The scheme adopts a normal control method based on feedback to support the temperature set point. Based on the control signals calculated at each control time, the scheme adopts the updated version

selection and permission control to effectively improve the quality of data and transactions. The performance of the scheme is verified by experiments. However, the feedback control method of this scheme has certain limitations, and it is difficult to run stably in the scenario of a large number of concurrent data.

Reference [32] proposes a framework for analyzing large-scale database management systems, modeling transactions and transaction management mechanisms as random time automata networks, so that atomicity, isolation, and time correctness can be analyzed through UPPAAL SMC, and the correctness of anti-collision systems of multiple automatic construction vehicles has been verified. However, this scheme lacks a reliable data recovery mechanism and is difficult to deal with database failures in high-concurrency scenarios.

The comparison between our work and existing work is shown in Table 1.

Table 1: Comparison between our work and existing work

	Year	Advantage	Disadvantage
Our work	2022	Low latency data processing, concurrency control, data recovery	Lack of practical application
[29]	2017	High concurrency data processing, Deadlock avoidance	High latency
[30]	2018	Low latency data processing, high data quality	The feedback control method has hysteresis
[31]	2014	Large-scale data management	Lack of data recovery

3 Real-Time Memory Data Management Model of Edge IoT Agent

The real-time memory data management model of the edge IoT agent is as Fig. 1.

At present, IoT has high requirements for data storage and efficient computing. To solve this problem, we built a lightweight real-time memory database transaction processing and data storage model, including a transaction pre-analysis method based on function replacement, a concurrency control technology based on transaction clustering, a recovery mechanism based on log-driven checkpoint algorithm, and a real-time efficient data storage model for data collection in the Internet of Things. We have improved the traditional database management method by introducing a series of technologies and put forward a new lightweight memory database management model, which can meet the high data processing requirements of the Internet of Things business and play an important role in the data collection and storage of the Internet of Things. The model proposed in this paper mainly adopts the following methods.

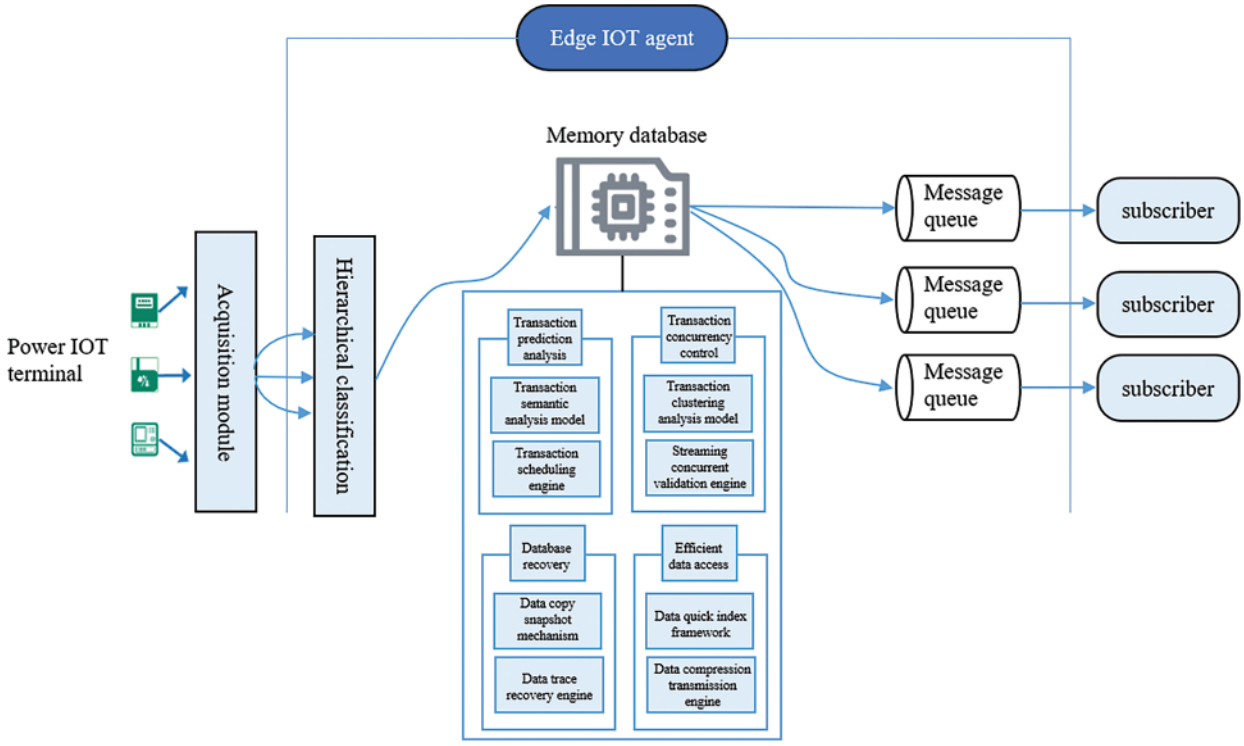

Figure 1: Real-time memory data management model of edge IoT agent

3.1 Transaction Pre-Analysis Method Based on Function Substitution

To improve the predictability of real-time database transaction processing, a transaction pre-analysis method based on function substitution is proposed. This method uses the tree transaction semantic model and combines the function substitution mechanism to propose the method of active real-time transaction pre-analysis processing and dynamic prediction. A transaction is analyzed in advance and corresponding measures are taken to ensure its normal completion and improve the real-time performance of transaction processing.

3.2 Concurrency Control Technology Based on Transaction Clustering

Aiming at the problem of complicated transactions caused by massive data collection and access in the Internet of things, a scheme called transaction clustering is proposed to automatically determine the best isolation mechanism for any given transaction pair. Based on transaction clustering, a clustering-based concurrency control algorithm is further constructed. This algorithm combines pessimistic and optimistic concurrency control algorithms, which not only alleviates the performance bottleneck but also realizes the accuracy of concurrency control and effectively ensures the stable and efficient operation of real-time memory databases.

3.3 Memory Database Recovery Mechanism Combining Replica Snapshot and Writability Log File

Given the volatile defects of memory, a memory database recovery mechanism combining replica snapshots and log files is proposed. This mechanism uses the existing memory database replica snapshots and logs to recover the memory database, which can effectively improve the speed and accuracy of data recovery and maintain the integrity of the memory database.

3.4 A Real-Time and Efficient Data Storage Model for Internet of Things Data Acquisition

According to the business requirements of massive data collection of the Internet of things, this paper proposes a real-time and efficient data storage model, uses the rabbitmq real-time communication service mechanism to complete the high-speed access of data, designs a data compression and dump mechanism to improve the quality and reliability of data, designs a data index structure based on CSB+ tree to improve the query efficiency, and finally realizes the real-time and efficient storage of the collected data of the Internet of things.

4 Real-Time Memory Data Optimization Mechanism of Edge IoT Agent

4.1 Transaction Pre-Analysis Method Based on Function Substitution

The flow of transaction pre-analysis based on function substitution is as Fig. 2.

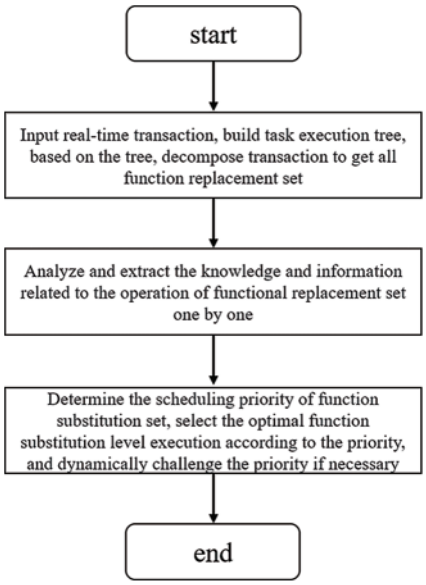

Figure 2: Flow chart of transaction pre-analysis based on function substitution

Transaction function substitution set means that real-time tasks can be decomposed into multiple different tasks, which are time limited. These tasks can be decomposed into a group of subtransactions, which have the same functions as the original task and are called task replacement sets. A collection of subtasks is called transaction function replacement. Even though the efficiency of these subtasks is different from that of the original task, the final function of this set is consistent with that of the original task. Therefore, the required functions can be successfully implemented by executing a set of subtasks, that is, a functional alternative. We define the real-time transaction T that can be decomposed into subtasks as:

$$T ::=< FA, R, <_t, tc > \tag{1}$$

$$FA ::= \{FX_i | 1 \leqslant i \leqslant m\} \tag{2}$$

$$FX_j ::= \{P_j | 1 \leqslant j \leqslant n\} \tag{3}$$

where FA is the function substitution set of transaction T, M is the number of function substitution sets of transaction T, P is the sub-transaction, and R is the various resources required by the transaction,

including CPU, memory, access objects, etc., $<_t$ refers to the temporal relationship of task execution; TC refers to the timing requirements and restrictions of transactions. Each time the system schedules real-time transactions, the object is a function substitution set contained in it. To improve the success rate of factual transactions and system operation efficiency, real-time transactions should be pre-analyzed [33]. Therefore, this paper proposes a transaction pre-analysis method based on function substitution.

Before the transaction arrives at the database processing module, build a task execution tree, and extract all the functional substitution sets of the transaction based on the tree model. The specific algorithm is as follows:

Algorithm 1: Function substitution tree construction algorithm

Input: real-time transactions T

Output: real-time transaction function substitution tree FAT

1. According to the temporal relationship of task execution <t, the task execution tree TT of transaction t is constructed. The nodes on the tree represent tasks, and the number of layers of nodes represents their execution order.
2. $FAT \leftarrow TT$
3. Starting from the root node, breadth-first traverse TT, view the transaction information table, obtain the sub transactions of the current node task, generate n flag (the complete method of the task) sub = transaction nodes, and add them to the corresponding position of the real-time transaction function replacement tree fat.
4. return FAT

Real-time transaction function substitution tree is as Fig. 3. Analyze and extract the knowledge and information related to the operation of the functional substitution set one by one, including the information of data set, operation logic (type and sequence), timing requirements, urgency and criticality, running time estimation, activities/transactions that may be triggered, as well as the structure, behavior, data, and timing between transactions.

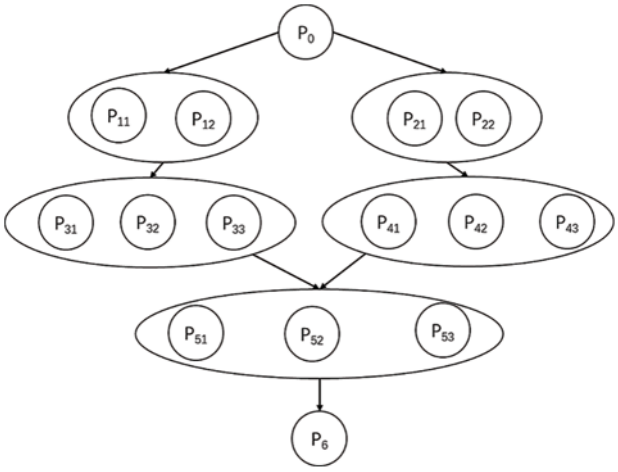

Figure 3: Real-time transaction function substitution tree

Before executing the transaction scheduling, the appropriate priority should be arranged for the functional substitution of the transaction, so that the optimal functional substitution can be put into

operation first [34]. When assigning priority to function substitution, the following three factors are mainly considered: CPU time consumed, resources occupied, and how many nesting levels.

Based on the above factors, the priority function of function substitution can be constructed:

$$P(FX_i) = \sum_{j=1}^{p} \left(\alpha_1 C_{ij} + \alpha_2 R_{ij} + \alpha_3 E_{ij}\right) \tag{4}$$

where, α_1, α_2, α_3 are the weights, P is the number of tasks of transaction T, and C_{ij}, R_{ij}, and E_{ij} respectively refer to the CPU execution time, resource requirements, and nesting level of the jth sub transaction of the ith function substitution set.

Select the optimal function substitution set according to the priority, and take measures to adjust the priority set scheduling algorithm when necessary dynamically, to ensure the successful execution of transactions and improve the efficiency and reliability of real-time memory database

4.2 Joint Management Algorithm

At present, everyone has reached a consensus: when the conflict between transactions is serious, pessimistic concurrency control should be used to avoid deadlock, and when the conflict is relatively minor, an optimistic concurrency control algorithm should be used to minimize the locking cost and improve transaction processing efficiency [35]. The concurrency control method based on transaction clustering will divide the transactions into different clusters according to the data sets that need to be accessed by the running transactions, and cluster the transactions with serious conflicts into a cluster. After clustering, transactions in the same cluster have a high conflict rate with each other and will be handled by a pessimistic concurrency control algorithm. On the contrary, transactions between different centralizations will be handled by optimistic concurrency control. The concurrency control technology based on transaction clustering mainly includes three steps: transaction clustering, applying for data locks in the cluster, and conflict verification. The process is as Fig. 4.

First, the possibility of conflicts between transaction work sets is estimated through the Jaccard similarity between them, and these transactions are clustered. The work sets of transactions are the data sets they access. The clustering of real-time transactions in the database is incremental. Therefore, the incoming transactions are compared with all other running transactions to find transactions with similarity higher than the threshold, and finally complete the clustering of transactions. Minhash is a locally sensitive hash algorithm, which can be used to quickly estimate the similarity of two sets.

Secondly, a transaction needs to apply for obtaining the data lock of the cluster before accessing any data record. When other transactions in the cluster occupy the data lock, the access request will be blocked until the lock is authorized.

Finally, even if transactions in the same cluster use lock isolation, the access of transactions from different clusters is not controlled, which may lead to data inconsistency. This method requires any transaction to adopt the conflict verification mechanism in optimistic concurrency control when committing. If there is a conflict, the transaction must be blocked.

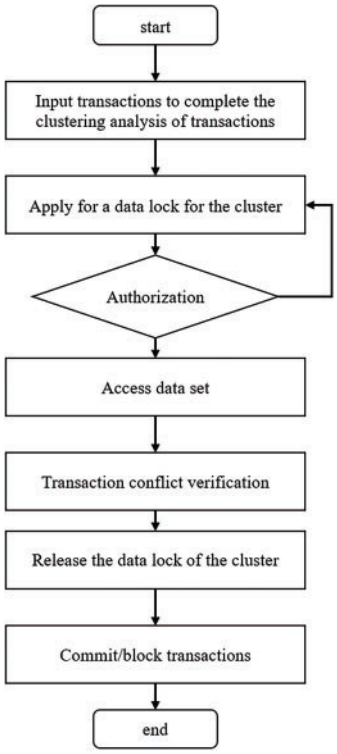

Figure 4: Process diagram of concurrency control technology based on transaction clustering

Algorithm 2: Transaction clustering algorithm

Input: incoming transaction T with K minhash vectors (k_1,k_2,...,k_K), and a minhash table m that stores the mapping from any minhash vector k to a set of transactions
Output: transaction set C in the same cluster as t
1. C ← ∅
2. for each k in (k_1, k_2, . . . , k_K) do
3. C_k ← The set of transactions that k maps to in M
4. C ← C ∪ Ck
5. return C

Algorithm 3: Transaction lock acquisition algorithm

Input: lock table t, transaction t applying for lock, mode m of lock, and access data set X
Output: true if the lock is successfully obtained, otherwise false
1. if x is not locked, then
2. Add transaction t with its MinHash vectors to M
3. return TRUE
4. Retrieve the MinHash table M and the current locking mode m' of x in T
5. Find the cluster C of t using Algorithm 1
6. if $C \setminus \{t\}$!= ϕ and m is not compatible with m' then

(Continued)

Algorithm 3: Continued

7. return FALSE
8. else
9. Add transaction t with its MinHash vectors to M
10. return TRUE

4.3 Memory Database Recovery Mechanism Combining Replica Snapshot and Writability Log File

On-chain and off-chain data verification process is as Fig. 5.

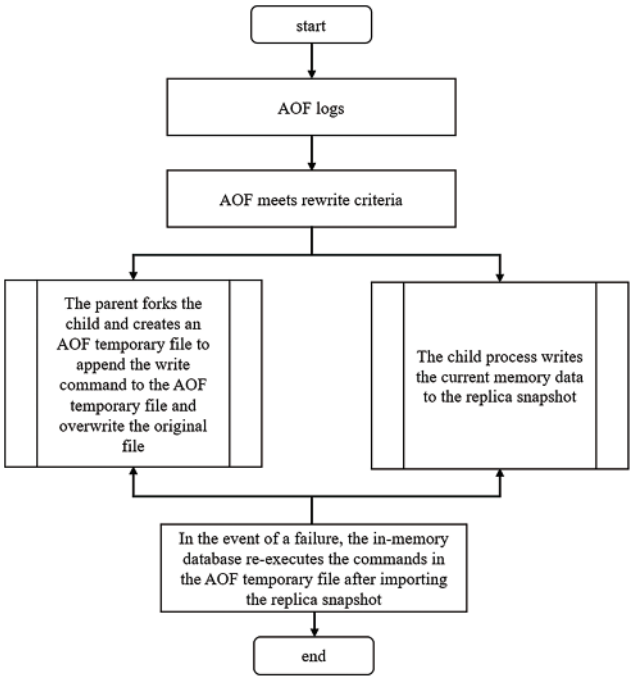

Figure 5: On-chain and off-chain data verification process

We mainly take two measures to recover the memory database. The first is data persistence. Data persistence can be realized mainly through the replica snapshot mechanism and log file query. Data snapshot means that in the memory database we designed, data will be stored on the corresponding disk according to certain policies. Log file query means that the memory database designed by us can write the write command to the traceable file at the same time, and the write order is the same as the file protocol order. When the memory database fails or the memory is powered off, we can use data snapshots and logs to ensure the consistency of the memory database before the failure and restore the database according to these files.

The Writability log continuously records the write operations of the in-memory database.

When the amount of trace log data is too large and reaches the rewriting standard, the memory database starts the child process to write the current memory database to the copy snapshot in the disk. At the same time, the parent process creates an AOF temporary file, and then appends the write command to the AOF temporary file and overwrites the original file.

When the transaction execution fails or the memory is powered off. The memory database imports the latest copy snapshot, and then simulates the client to execute the commands in the AOF file once, to restore the database to the state of the last shutdown or failure, and finally realize the recovery of the database.

4.4 A Real-Time and Efficient Data Storage Model for Internet of Things Data Acquisition

High-speed data access method based on RabbitMQ real-time communication service mechanism.

The process is as follows:

The real-time memory database subscribes to the messages published by the Internet of things data convergence point.

The data convergence point of the Internet of things uses TCP connection to establish an AMQP communication channel. It sends messages to the server, that is, the summarized collected data, and the server passes the messages through the corresponding queue bound.

After receiving the message, the queue establishes a channel through a network connection and pushes the message to the real-time memory database.

The real-time memory database sends ack confirmation information to the queue after receiving the message. When the queue receives ACK, the message in the queue is deleted to avoid message accumulation.

The real-time memory database analyzes and processes the data and writes it into the database to complete the task of collecting and storing data.

Query algorithm of data index structure based on CSB+ tree

CSB+ tree combines the ideas of B+ tree and CSS tree and uses chain structure to store in memory [36]. Unlike the B+ tree, there is only one pointer in the node of the CSB+ tree. Each parent node of the CSB+ tree reserves a pointer to the child node, so that all child nodes of the same node are stored in an array, which is called a node group. To improve the utilization efficiency of the cache and realize cache sensitivity, the traditional CSB+ tree controls the nodes of the tree to about one cache block, generally between 64 bytes and 128 bytes. This will cause the height of the index tree to be very high, resulting in a series of problems: with the increase of the height of the index tree, the number of TLB failures will also increase significantly because the number of parent nodes flowing to child nodes increases during the query processing of the entire index tree. More memory page switching. Aiming at the problems of the traditional CSB+ tree, this paper proposes an improved CSB+ tree, which expands the capacity of nodes, reduces the height of the index tree, reduces the number of TLB mismatches, and improves query performance. Its structure is shown in Fig. 6.

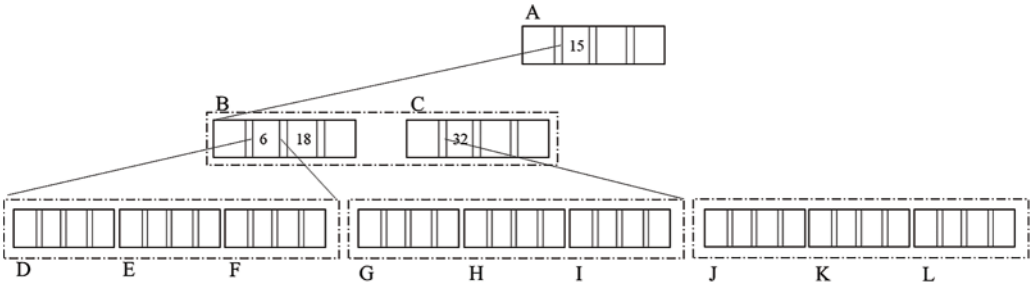

Figure 6: Data index structure based on improved CSB+ tree

Suppose that the node of the improved CSB+ tree is composed of two parts, that is, the index A [n] in the node (where n is the number of intervals), the node header m including control information, and all partition sets n. To ensure the efficiency of the cache operation, the size of the node header m must be controlled within a cache block size, generally between 64 and 128 bytes. Let s be the cache size of the current system, and the size of M be equal to s. According to the partition design in the node, n is a group of N intervals of equal size. Since the size of each interval is s, the size of n is $n \times x$. Assuming that all index entries in this node are integers, the length of the pointer and the integer number are 4 bytes, the number of index entries that can be stored in each interval is w. The node header m includes an index entry counter in the node with a size of 2 bytes and a maximum index entry value in the current node with a size of 4 bytes.

Algorithm 4: Query algorithm of data index structure based on CSB+ tree

Input: key value k

Output: if the query is successful, the index entry pointer P corresponding to the K value is returned; otherwise, null is returned

1. access the header information partition of the node pointer and record the starting position of the partition in the node
2. access the index in the node and find the possible interval n of K value
3. Visit the interval and do binary search for K in the interval
4. if find(k) = null
5. if this_node is middle, then
6. n = n_next
7. goto 4
8. else
9. return NULL
10. else
11. P = find(k)
12. return P

5 Simulation

In the experiment, we designed a lightweight real-time memory database as follows:

As shown in Fig. 7, the lightweight real-time memory database is mainly composed of a transaction processing model and a data storage model. The transaction processing model includes the transaction pre-analysis method based on function substitution, the concurrency control technology based on transaction clustering, the memory database recovery mechanism combined with copy snapshots and trace log files, etc. The data storage model includes the data high-speed access method based on the rabbitmq real-time communication service mechanism, the data index structure based on the improved CSB+ tree, the data processing method based on the five-point cubic smoothing method, and the data compression and dump mechanism based on the LZW algorithm. These two models can effectively improve the transaction processing performance and reliability of the database, accelerate the warehousing of the collected data of the Internet of things, and improve the efficiency of data analysis and processing, laying a strong foundation for the efficient application of the Internet of things data. Enterprise managers can analyze the production situation of the enterprise according to the historical data dumped by the real-time memory database, find the weak links in the operation

of the enterprise, and adopt corresponding adjustment measures to promote the further sustainable development of the enterprise.

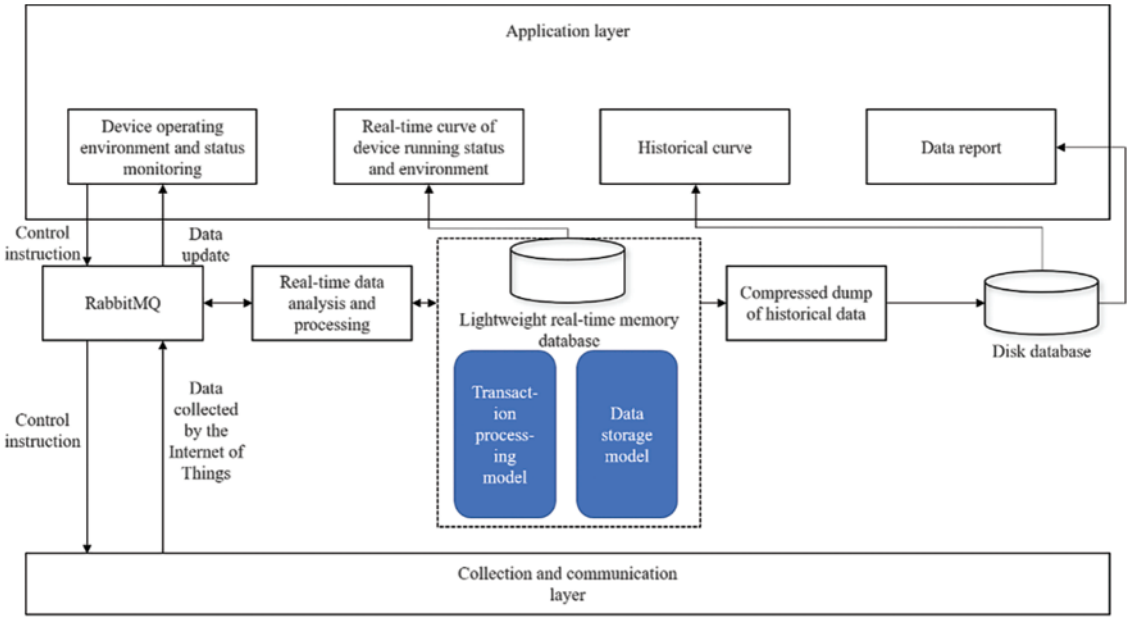

Figure 7: Lightweight real-time memory database use case

To determine the optimal value of the calculated cache item, this paper finds the parameter value that makes the cache hit rate the highest through experiments. Take u1 as 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 respectively, and the corresponding u2 as 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1. For the value of each pair of parameters, send 100 simulated query requests in the system, calculate the cache value V corresponding to each pair of cache items in the process of cache item storage and replacement, and finally calculate the total cache hit rate. The experimental results are shown in Fig. 8.

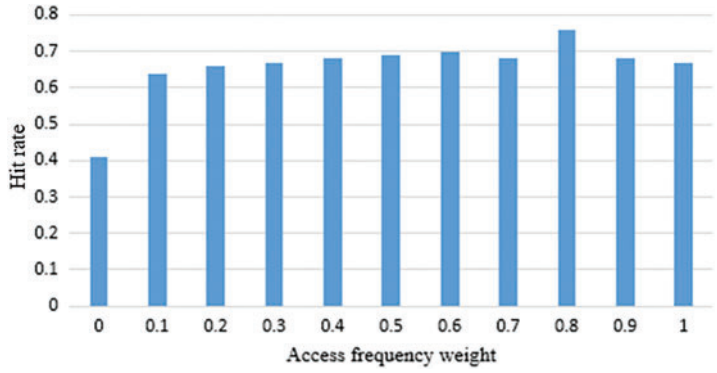

Figure 8: Different hit rates corresponding to different parameter values

The abscissa in the figure above indicates the value of the access frequency weight, and the ordinate indicates the cache hit rate when different values are taken. It can be seen from the experimental results that when the access frequency weight is set to u = 0.8 and the value weight is set to 0.2, the cache hit

rate is the highest. It can be seen that in our system, the access frequency of cache items has a greater impact on cache replacement.

According to the results of the above experiment, take the value of U1 as 0.8 and the value of U2 as 0.2 to continue the experiment. 10000 random numbers are used to simulate the query request sequence sent by the user, and 500 data tables are queried and accessed. According to Zipf's law, 8000 (10000 $*$ 80%) requests in the 10000 request sequence are accesses to 100 (500 $*$ 20%) of the 500 data tables, and the remaining 2000 requests are accesses to the remaining 400 data tables. In the experiment, the random number is read through the code every 10ms, and the database command corresponding to the random number is sent to the server. If the cache hits, the number of hits of the cache item corresponding to the database statement is increased by 1. If it misses, the query result needs to be obtained by connecting to the database, and the query result set is stored in the cache space. In 10000 accesses, the hit times of 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000 and 10000 requests are recorded respectively, and the cache hit rate is calculated. For the traditional LRU cache replacement algorithm, when the cache capacity limit is reached, the cache item with the earliest access time is moved out of the cache according to the access time. The experimental results are shown in Fig. 9.

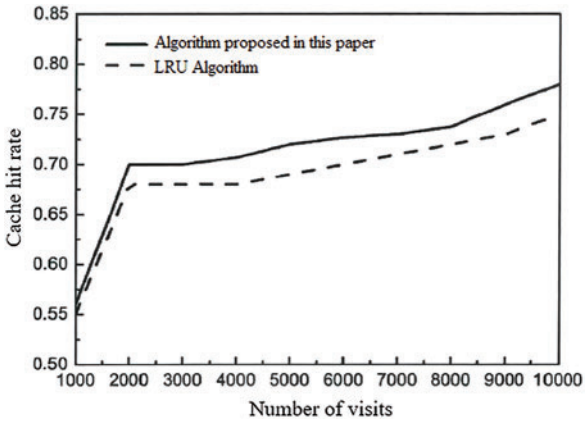

Figure 9: Comparison of hit rate of the cache replacement algorithm

The abscissa in the figure above indicates the number of access requests issued by users, a total of ten groups of data, and the ordinate indicates the cache hit rate corresponding to different access numbers. It can be seen from the figure that with the increase of the number of accesses, the hit rate of the two cache replacement algorithms is improving. When the access size is small, the hit rate of the two cache replacement algorithms is almost the same. However, with the increase of accesses, the cache hit rate of the cache replacement algorithm proposed in this paper is slightly higher than the traditional LRU algorithm. The higher the cache hit rate, the more times the target data is obtained by using the cache, the fewer times the results are obtained by connecting to the database, and the shorter the average query response time. Therefore, using the cache replacement algorithm proposed in this paper can improve the performance of the system cache.

6 Discussion

The experimental results show that our algorithm can effectively improve the cache hit rate, which verifies the effectiveness of our data storage model and cache replacement algorithm. In addition, from the experimental results, we can see that when the number of visitors is small, our algorithm

and the baseline algorithm are not much different, but as the number of visitors increases, the gap between our algorithm and the baseline algorithm is becoming larger and larger, which shows that the transaction pre-analysis method and concurrency control technology we adopted in the algorithm have achieved certain results. However, the method in this paper is lack of research on fast processing of real-time data, and further research is needed in the future to make the system better. At the same time, the data indicators observed in the experiment are relatively single, and it is difficult to observe the difference between the algorithm in this paper and the existing work from multiple perspectives. More experiments need to be carried out in future work.

7 Conclusion

Facing the business needs of efficient data collection and storage of the Internet of things, this paper optimizes the transaction processing efficiency and data storage performance of the memory database, realizes a lightweight real-time memory database transaction processing and data storage model, and improves the reliability and efficiency of the database, which has broad prospects. Enterprise managers can accurately obtain the operating status and environmental status of the equipment in real-time through the lightweight real-time memory database, so as to timely and accurately issue the corresponding control instructions, ensure the stable and efficient operation of the production system, and improve the efficiency of the enterprise. On the other hand, enterprise managers can analyze the production situation of the enterprise according to the historical data dumped by the real-time memory database, find the weak links in the operation of the enterprise, and adopt corresponding adjustment measures.

Acknowledgement: This paper is supported by the National Key R&D Program of China "Key technologies for coordination and interoperation of power distribution service resource" [2021YFB13 02400]; "Research on Digitization and Intelligent Application of Low-Voltage Power Distribution Equipment" [SGSDDK00PDJS2000375].

Funding Statement: This paper is funded by the National Key R&D Program of China "Key technologies for coordination and interoperation of power distribution service resource" [2021YFB1302400]; "Research on Digitization and Intelligent Application of Low-Voltage Power Distribution Equipment" [SGSDDK00PDJS2000375].

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] W. G. Choi, D. Kim, H. Roh and P. Sanghyun, "OurRocks: Offloading disk scan directly to GPU in write-optimized database system," *IEEE Transactions on Computers*, vol. 70, no. 11, pp. 1831–1844, 2021.

[2] X. Zhou, W. Liang, K. Yan, W. M. Li, K. I. Wang *et al.,* "Edge enabled two-stage scheduling based on deep reinforcement learning for Internet of everything," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3295–3304, 2022.

[3] W. Liang, Y. Hu, X. Zhou, Y. Pan and K. I. Wang, "Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 8, pp. 5087–5095, 2022.

[4] K. Yan, X. Chen, X. Zhou, Z. Yan and J. Ma, "Physical model informed fault detection and diagnosis of air handling units based on transformer generative adversarial network," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 2192–2199, 2022.

[5] L. Qi, W. Lin, X. Zhang, W. Dou, X. Xu *et al.,* "A correlation graph based approach for personalized and compatible web APIs recommendation in mobile APP development," *IEEE Transactions on Knowledge and Data Engineering*, 2022.
[6] M. Azrour, J. Mabrouki, A. Guezzaz and Y. Farhaoui, "New enhanced authentication protocol for Internet of Things," *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 1–9, 2021.
[7] V. P. K. Anne, V. S. Ponnam and G. Praveen, "A significant approach for cloud database using shared-disk architecture," in *Proc. CSI Sixth Int. Conf. on Software Engineering*, Indore, India, pp. 1–4, 2012.
[8] X. Zhou, Y. Hu, J. Wu, W. Liang, J. Ma *et al.,* "Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 570–580, 2022.
[9] K. Yan K and X. Zho, "Chiller faults detection and diagnosis with sensor network and adaptive 1D CNN," *Digital Communications and Networks*, vol. 8, no. 4, pp. 531–539, 2022.
[10] L. Qi, C. Hu, X. Zhang, M. Khosravi, S. Sharma *et al.,* "Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4159–4167, 2021.
[11] T. Zhou, W. Liu, N. Li, X. Yang, Y. Han *et al.,* "Secure scheme for locating disease-causing genes based on multi-key homomorphic encryption," *Tsinghua Science and Technology*, vol. 27, no. 2, pp. 333–343, 2022.
[12] C. Hu, W. Fan, E. Zeng, Z. Hang, F. Wang *et al.,* "Digital twin-assisted real-time traffic data prediction method for 5G-enabled internet of vehicles," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2811–2819, 2022.
[13] M. Murugesan, H. Hashimoto, J. Bea, M. Koyanagi and T. Fukushima, "Chip-to-chip/wafer three-dimensional integration of 2.5 mm-sized neuron and memory chips by via-last approach," in *Proc. 7th Int. Workshop on Low Temperature Bonding for 3D Integration*, Nara, Japan, pp. 28, 2021.
[14] X. Zhou, X. Yang, J. Ma and K. I. -K. Wang, "Energy-efficient smart routing based on link correlation mining for wireless edge computing in IoT," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14988–14997, 2022.
[15] K. Yan, X. Zhou and J. Chen, "Collaborative deep learning framework on IoT data with bidirectional NLSTM neural networks for energy consumption forecasting," *Journal of Parallel and Distributed Computing*, vol. 163, no. 8, pp. 248–255, 2022.
[16] G. Zhou, C. Pan, H. Ren, K. Wang, K. K. Chai *et al.,* "User cooperation for IRS-aided secure MIMO systems," *Intelligent and Converged Networks*, vol. 3, no. 1, pp. 86–102, 2022.
[17] X. Xu, H. Tian, X. Zhang, L. Qi, Q. He *et al.,* "DisCOV: Distributed COVID-19 detection on X-Ray images with edge-cloud collaboration," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1206–1219, 2022.
[18] H. Liu, "Application analysis of artificial intelligence technology in computer network based on big data era," in *Proc. Int. Conf. on Intelligent Transportation, Big Data & Smart City*, Dalian, China, pp. 34–38, 2020.
[19] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu *et al.,* "Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9310–9319, 2022.
[20] K. Yan, "Chiller fault detection and diagnosis with anomaly detective generative adversarial network," *Building and Environment*, vol. 201, no. 2, pp. 107982, 2021.
[21] J. Ren, J. Li, H. Liu and T. Qin, "Task offloading strategy with emergency handling and blockchain security in SDN-empowered and fog-assisted healthcare IoT," *Tsinghua Science and Technology*, vol. 27, no. 4, pp. 760–776, 2022.
[22] D. Anuradha, N. Subramani, O. Khalaf, Y. Alotaibi, S. Alghamdi *et al.,* "Chaotic search-and-rescue optimization-based multi-hop data transmission protocol for underwater wireless sensor networks," *Sensors*, vol. 22, pp. 2867, 2022.
[23] D. Yu, L. Zhang, Q. Luo, X. Cheng, J. Yu *et al.,* "Fast skyline community search in multi-valued networks," *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 171–180, 2020.

[24] J. Wang, Z. Duan, X. Han and D. Yang, "Efficient top/bottom-k fraction estimation in spatial databases using bounded main memory," *Tsinghua Science and Technology*, vol. 27, no. 2, pp. 223–234, 2022.
[25] X. Zhou, X. Xu, W. Liang, Z. Zeng and Z. Yan, "Deep-learning-enhanced multitarget detection for end-edge–cloud surveillance in smart IoT," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12588–12596, 2021.
[26] K. Yan, A. Chong and Y. Mo, "Generative adversarial network for fault detection diagnosis of chillers," *Building and Environment*, vol. 172, no. 8, pp. 106698, 2020.
[27] A. K. Sandhu, "Big data with cloud computing: Discussions and challenges," *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 32–40, 2022.
[28] H. Niu, Z. Chu, Z. Zhu and F. Zhou, "Aerial intelligent reflecting surface for secure wireless networks: Secrecy capacity and optimal trajectory strategy," *Intelligent and Converged Networks*, vol. 3, no. 1, pp. 119–133, 2022.
[29] C. Zhang, "Intelligent Internet of things service based on artificial intelligence technology," in *Proc. IEEE 2nd Int. Conf. on Big Data, Artificial Intelligence and Internet of Things Engineering*, Nanchang, China, pp. 731–734, 2021.
[30] S. Pandey and P. Astya, "Real time database management in mobile computing," in *Proc. Int. Conf. on Computing, Communication and Automation*, Greater Noida, India, pp. 825–828, 2017.
[31] T. Bai, H. Xu and Y. Pan, "Thermal-aware QoS management for real-time databases," in *2018 IEEE 9th Int. Conf. on Software Engineering and Service Science (ICSESS)*, Beijing, China, pp. 1–4, 2018.
[32] S. Stoja, S. Vukmirovic, B. Jelacic, D. Capko and N. Dalcekovic, "Architecture of real-time database in cloud environment for distributed systems," in *Int. Conf. on Artificial Intelligence, Modelling and Simulation*, Madrid, Spain, pp. 258–263, 2014.
[33] S. Cai, B. Gallina, D. Nyström and C. Seceleanu, "Statistical model checking for real-time database management systems: A case study," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, pp. 306–313, 2019.
[34] Y. Yang, X. Yang, M. Heidari, M. A. Khan, G. Srivastava *et al.,* "ASTREAM: Data-stream-driven scalable anomaly detection with accuracy guarantee in IIoT environment," *IEEE Transactions on Network Science and Engineering*, 2022.
[35] F. Wang, G. Li and Y. Wang, "Privacy-aware traffic flow prediction based on multi-party sensor data with zero trust in smart city," *ACM Transactions on Internet Technology*, 2022.
[36] L. Qi, Y. Yang, X. Zhou, W. Rafique and J. Ma, "Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6503–6511, 2022.