Improved Query Model for Rapidly Query Based on Distributed Hash Index

Qisen Zhou[1,2,a], Zhongguang Sun[1,2,b] and Yong Li[1,2,c]

[1]*China Coal Technology and Engineering Group Chongqing Research Institute, Chongqing, China*
[2]*China Coal Technology and Engineering Group Chongqing Smart City Technology Research Institute, Chongqing, China*
a. runciov5@hotmail.com, b. sunzhongguang126@126.com, c. ace2007@163.com

Keywords: Smart urban management platform, the Internet of things, distributed hash index, data dictionary, hash bucket.

Abstract: Aiming at the low efficiency of the Big data query of the internet of Things, the research based on the smart urban management platform, and on the premise of retaining the original platform data integration framework. The query model and process are redesigned and a new data improved query model is proposed by introducing hash coding, distributed index and data dictionary. Compared with the one by one comparison query method, the speed of the improved query model is increased by 106 times when processing millions of data, which significant improves the query efficiency of big data. The research results can help improve the data processing and analysis ability of Chongqing Liangjiang Smart Urban Management Platform in China.

1. Introduction

The Internet of Things (IOT) is now synonymous with the third wave of information, as the scale of data that needs to be stored and managed in the IOT becomes huge and complex [1-3], how to process data efficiently in the IOT has become an important research direction. Domestic and international experts and scholars have proposed a mass of optimization algorithms, such as dynamic programming methods for commercial systems, ant colony algorithms in the smart domain and particle swarm algorithms, etc. [4-7]. There are two main directions of the current search methods for massive data, namely, the search method based on tree structure and the search method based on Hash methods [8-10]. Compared to the tree structure search, the hash search method is dimension insensitive and can achieve dimensionality reduction through optimized hash coding, thus has more advantages in query speed and memory occupation[11].

To address the problem of inefficient querying in a large number of data processing sessions in Chongqing Liangjiang (China) Smart Urban Management Platform, this paper introduces local sensitive Nature Hash Ideas and Enhanced Hash Bucket Improvement Query Methods, to present New Data Based on Hash Distributed Indexes and Data Dictionaries Improve the query model, aimed at significantly improving the efficiency of the platform in handling mass data integration.

2. Research Overview

First, the data integration of the Chongqing Liangjiang Smart Urban Management Platform system is based on the mine Internet of Things. The area of the mine Internet of Things system is relatively limited. The existing smart urban management platform covers nearly one hundred thematic systems and the number of monitors reaches millions. As the amount of system integration increases, the original one-by-one comparison query method can no longer meet the application requirements of smart urban management platform data integration. For example: when the data query table length is 2000, it can be processed within 1s; when the data query table length reaches 100,000, the processing time is as high as hundreds of seconds, it can cause obvious lag to the system. The reasons for the analysis are as follows: 1. The monitoring parameter quantity in the mine IOT is generally in the order of 10^4, while the monitoring parameter quantity in the smart urban IOT platform is rapidly expanded to the order of 10^6~10^8; 2. The query is used to compare one-by-one in order Keyword mode, in theory, the time complexity [T(n)=O(n^2)] ,which is a very high consumption; 3. The keyword index[12] of the device object is a string type, and the time-consuming to match is significantly higher For integer values. Therefore, it is necessary to improve the original query method of comparing one-by-one on the platform.

3. Exploration of Quick Query Method

To solve the above problems, this paper uses distributed hash index [13], enhanced hash bucket and data dictionary to improve the query method and model to solve the problem of low efficiency of data processing.

3.1.Distributed Hash Index

The hash algorithm is a message digest function that compresses any length of the message to a fixed length. The algorithm uses the idea of function mapping to associate the record storage location with the record key so that it is capable of fast search. In the hash value algorithm, there is a probability that the value of a character string is repeated after hash calculation, and the probability depends on the optimization degree of the hash value calculation method. The hash value calculation process to be implemented in this paper refers to the hash method with Locality Sensitive Hashing (LSH) [14], hoping to reduce the probability of hash collision, the calculation steps are as follows：

Step 1: Suppose V is a characteristic point vector, construct a set of random variables a and a random real number b in the range of [0, w], where W is a larger real number.

$$H_{a,b}(V) = [\frac{a * V + b}{W}] \tag{1}$$

Step 2: Calculate the initial hash value $H_{a,b}(V)$ of vector V according to Equation (1), dot the mapped hash value with random real Numbers r, and take the modulus of the larger prime constant to get the hash value h.

$$h = (\sum_{i=1}^{k} r_i^{''} a_i) \bmod prime \tag{2}$$

Through the steps above, hash value of data keywords can be realized. On this basis, the distributed index architecture is set to store data objects into multiple data groups by hash grouping. The calculation formula of hash index of each data object is as follows:

$$I(V) = h \bmod size \tag{3}$$

As shown in formula (3), the module of hash value h is taken according to the number of nodes in the data group, and then the feature point v is allocated to the corresponding data group according to the result $I(V) \in [0, size-1]$. In order to solve the inevitable hash conflict problem [15-16], an enhanced hash bucket model (composed of hash bucket, standby hash bucket and overflow bucket) is introduced to store data objects in data packets.

3.2. Comparison of Hash Data Structures

Equation Hashtable, Sorted Dictionary and Dictionary are the three most commonly used and efficient data structures in hash algorithm. In order to select the optimal data structure which suitable for the system, the C# &. Net simulation program is developed to compare the storage and retrieval operation efficiency of the three structures. The simulation results are shown in Figure 1.

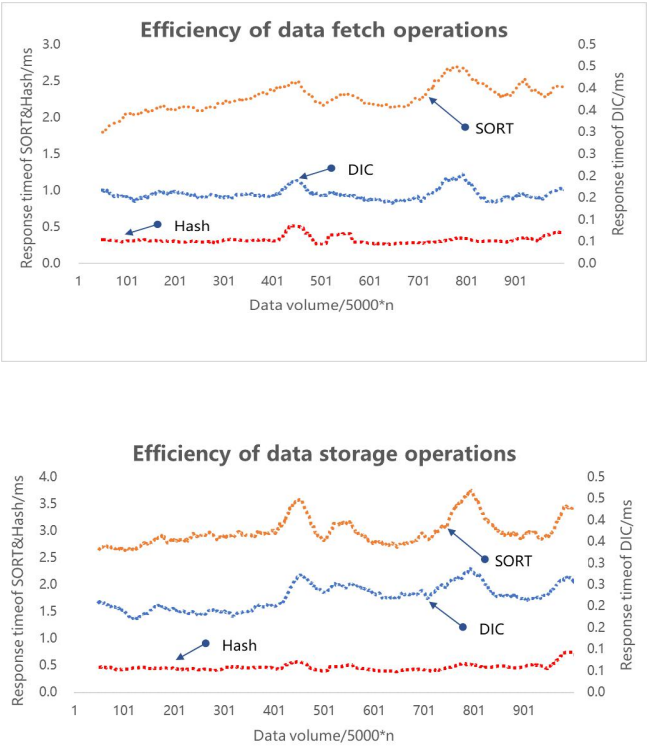

Figure 1: Comparison of data storage and retrieval efficiency of Hashtable, Dictionary and SortedDictionary.

Taking the results of 5.0×10^6 simulated data as an example for comparison, it can be seen that 5,000 data were stored by the data storage operation, Hashtable cost 0.520ms, Dictionary cost 0.304ms and SortedDictionary cost 3.767ms. During the data fetch operation, 5,000 data were

extracted, Hashtable cost 0.369ms, Dictionary cost 0.207ms and SortedDictionary cost 2.410ms. Overall, Dictionary takes less time to read and save than the other two data structures, so it is more appropriate with the platform's requirement for efficient processing of big data.

4. Improved Model of Quick Query

4.1. Improvement of Quick Query Model

To solve the problem of data query efficiency in Chongqing Liangjiang Smart Urban Management Platform, the distributed hash index and data dictionary are adopted to create the hash grouping model without modifying the original framework, and the query model is redesigned and improved. The main improvement directions include the following four points:

1) Taking the Code of the data object as the key word for query comparison, so as to directly compare integers and improve the efficiency.

2) Use LSH method to rewrite the calculation function GetHashCodeX() to reduce the probability of hash collision.

3) Use enhanced hash bucket structure to store data objects to solve the problem of conventional hash conflicts.

4) Use distributed hash index and data dictionary to store data objects and reduce the time complexity of query.

4.2. Implementation of Quick Query Model

In view of the above four improvement directions, the specific design of the fast query model is proposed as follows:

1) The data object type is DataItem class, including key information Code, Hashkey, and other data properties. Among them, Hashkey is the hash value calculated by encoding through GetHashCodeX() function.

2) The hash bucket which is type of DataItem object designed to store hash conflicts that is HashBucket class, which contains a list of list type PList (used to store DataItem objects) and an integer encoded attribute ID (HashKey of all DataItem objects in PList is equal to this ID).

3) The distributed index grouping type is PointHashGroup class, which contains a BDcit object of dictionary < int, Hashbucket > type. The number of data dictionaries is unlimited. The key type is integer and the type of value is Hashbucket.

4) Design a PDict object of Dictionary < int, PointHashGroup > whose length is Maxlen. Maxlen can be configured according to the data size during system initialization. The key type is integer in [0, maxlen-1], and the value type is PointHashGroup.

5) When a DataItem is to be inserted into the model, it should follow the process below (as shown in Figure 2）.

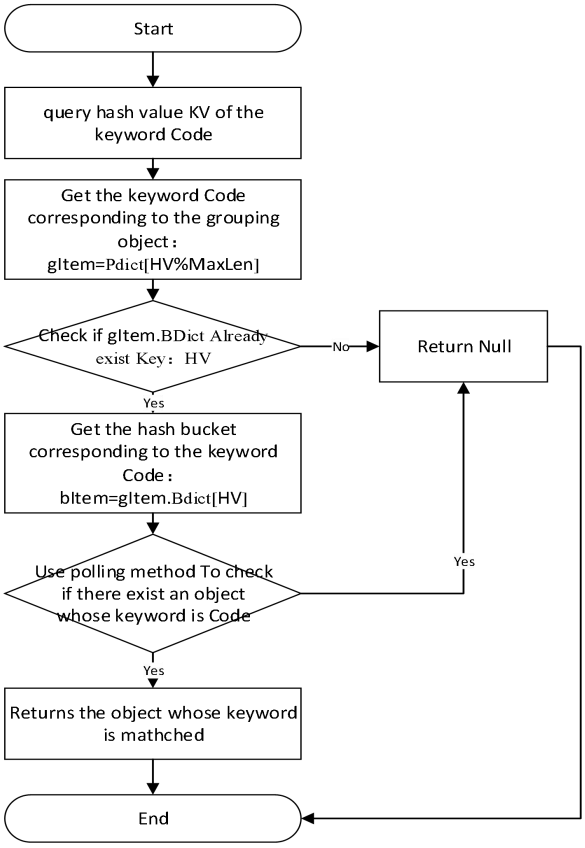

Figure 2: Improved query model flowchart.

5. Experimental Results and Analysis of Numerical Simulation

To show the optimization of "the improved searching model", C#. Net development simulation program is used to test the difference between "the unitary searching model" and the one-by-one comparison query method . Setting the value of Maxlen as 100, and the data size starts from 1000. 1000 identical data will be added to the data sheet of each model every time until the total data size reaches 87 thousand. The elapsed time is also recorded and shown on Figure 3.

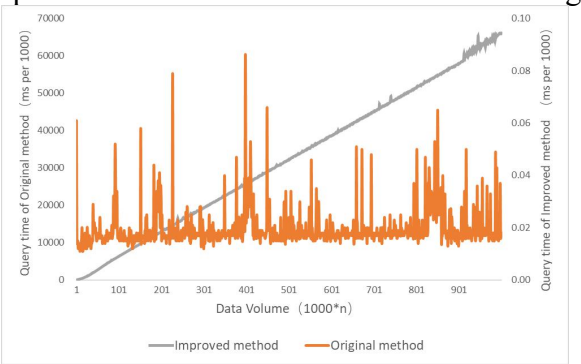

Figure 3: Comparison of numerical simulation results.

From Figure 3, the searching efficiency highly increased by using "The improved searching model" compared with the one-by-one comparison query method. For instance, when it processes

10^6 data set, it takes 66,064ms by using one-by-one method, and it only takes 0.018ms by using "The improved searching model". The searching speed is 3 x 10^6 times faster.

6. Conclusions

This paper aims at the inefficiency of parsing and querying the massive monitoring data of the Internet of Things encountered in the Chongqing Liangjiang Smart Urban Management Platform. The query method is improved by introducing local sensitive qualitative hashing and enhanced hash buckets. The distributed hash index and data Based on the dictionary, a new data improvement query model is proposed, which replaces the original one-by-one comparison query method in the platform. Through the comparison of simulation experiments, the improved model can greatly improve the query efficiency and effectively solve the bottleneck problem of data processing in Chongqing Liangjiang Smart Urban Management Platform.

Acknowledgments

This study is financially supported by Science innovation and entrepreneurship special funded projects of China coal technology & engineering group (2018ZD007), the Natural Science Foundation of Chongqing, China (cstc2019jcyj-msxmX0633) and Science innovation and entrepreneurship special funded projects of Tiandi science & technology co. LTD (2018-TD-QN062).

References

[1] Pan Yunhe, "Data is king in the Internet of things Era," Internet of Things Technologies, pp. 11, May 2011. (references).

[2] Yu Shao-hua, "Advanced Stage after Industrial Internet Interconnection: Enterprise-AI Agent," Study on Optical Communications, pp.1-8, January 2019.

[3] Zhao Xiaoyu, Li Futong and Mo Qiong, "Research on mass Heterogeneous Data Retrieval in Internet of Things," ELECTRONIC TECHNOLOGY & SOFTWARE ENGINEERING, pp. 204, April 2014.

[4] Shan Xue, Biao Luo, Derong Liu and Yueheng Li, "Adaptive dynamic programming based event-triggered control for unknown continuous-time nonlinear systems with input constraints," Neurocomputing, vol. 396, pp. 191-200, 2020.

[5] FENG Binbin, LI Li, BAI Yunchao and LI Congbo, "Design of laser intelligent fire-fighting evacuation system based on ant colony algorithm," Journal of Chongqing University, vol. 43, pp. 31-39, May 2020.

[6] Liu Mingming, Wang Quan and Ma Shou, "Well placement optimization of coalbed methane based on hybrid particle swarm optimization algorithm," LITHOLIGIC RESERVOIRS, in press.

[7] HUANG Song and QIU Jian-lin, "Improvement of k-means clustering algorithm based on genetic algorithm and its application," Computer Engineering and Design, vol. 41, pp. 1617-1623, June 2020.

[8] LI Zhi-wei, "Study on distributed database query optimization based on greedy strategy," Computer Engineering and Design, vol. 31, pp. 3838-3840+3875, 2010.

[9] WANG Yan, YIN Biao, LIU Genghao, SONG Baoyan and WANG Junlu, "Skyline Query of Massive Incomplete Data Based on Combinational Dimensions," Journal of Frontiers of Computer Science and Technology, vol. 10, pp. 495-503, April 2016.

[10] Liu Peizeng, "The Design and Implementation of Multilevel Storage System Oriented to Multidimensional Data," Beijing University of Posts and Telecommunications, 2018.

[11] Zhongming Jin, "Research on Large Scale Multimedia Data Retrieval Based on Hashing Algorithm," Zhejiang University, 2015.

[12] Du Ruizhong, Li Mingyue and Tian Junfeng, "Multi-keyword Ranked Ciphertext Retrieval Scheme Based on Clustering Index," Journal of Computer Research and Development, vol. 56, pp. 555-565, March 2019.

[13] LIN Chaohui, YU Junqing, HE Yunfeng, GUAN Tao asnd AI Lief, "High-Dimensional Distributed Indexing Based on Locality-Sensitive Hashing," Egamer, vol. 7, pp.811-818, July 2013.

[14] Ji Jianqiu, "Research on Hashing Methods towards Large-Scale Similarity Computation and Search," Tsinghua University, 2015.

[15] Pan Changxin and Cao Lina, "Principle of Communication," Beijing: National Defense Industry Press, 2009.
[16] Wang Xiaoling and Lu Peng, "Research on balanced learning-based enhanced hash bucket model," Study On Optical Communications, pp. 30-32, March 2014.