Query Optimization Method for Massive Heterogeneous Data of Internet of Things Based on Machine Learning

Yun-wei Li[1] and Lei Ma[2(✉)]

[1] Beijing Youth Politics College, Beijing 100102, China
liyunwei@bjypc.edu.cn
[2] Beijing Polytechnic, Beijing 100016, China
malei235@tom.com

Abstract. In view of the problem that the traditional query optimization method of massive heterogeneous data of the Internet of things can not describe the data characteristics clearly, which results in the long execution time of data query, a query optimization method of massive heterogeneous data of the Internet of things based on machine learning is designed. It divides the massive heterogeneous data query level of the Internet of things, and extracts the data characteristics according to the hierarchical structure and the Dirichlet smoothing method in machine learning. The feature data is transformed into a query tree, and a dynamic data dictionary is constructed. The data dictionary is referred to the traditional query optimization method of massive heterogeneous data in the Internet of things. At this point, the query optimization method for massive heterogeneous data of the Internet of Things based on machine learning is designed. The test link of the construction method shows that the use effect of this method is better than the original method and the method based on artificial intelligence technology.

Keywords: Machine learning · Dirichlet smoothing method · Heterogeneous data · Internet of things

1 Introduction

With the further development of the Internet of things technology, in the face of growing data, the traditional storage architecture due to poor scalability, in the long run, the storage environment will become increasingly complex, resulting in high energy consumption [1]. Unlike traditional storage systems, distributed cloud storage systems can store massive amounts of information, efficiently manage large-scale files, and provide good query efficiency. But because the cloud storage system is based on Internet technology, it mainly stores small file data, such as small picture streams and small video streams, but in the Internet of Things, it needs to repeatedly access massive picture streams and video stream data. Because the Internet of Things needs to collect various information such as sound, light, heat, electricity, chemistry, location, etc., and the content of information captured by different types of sensors and information receivers is very different.

S. Liu and L. Xia (Eds.): ADHIP 2020, LNICST 347, pp. 345–356, 2021.
https://doi.org/10.1007/978-3-030-67871-5_31

The distributed storage of massive data and the data processing between heterogeneous databases have become an inevitable trend. Through the integration of data information and hardware devices in various databases, a heterogeneous database system is formed which is logically unified and physically independent [2, 3]. This kind of cross-database query and multi-table connection needs to increase data redundancy to ensure the reliability and availability of heterogeneous database systems, but the physical distributed storage of data and the need for such data redundancy make heterogeneity become higher. The query processing of the database adds more content and difficulty. Therefore, query optimization of heterogeneous database system plays an important role.

This article first analyzes the structure of the heterogeneous database of the Internet of Things and the query processing process, and understands that in the query process of the heterogeneous database system, the query request should be converted into a global query tree and converted. Through the principle of equivalence, the query tree can be decomposed into a local query tree to further optimize and determine the final query execution plan. Such query optimization process needs to consider local response time and transmission cost. At present, the existing query optimization algorithms can only achieve the shortest local response time or the lowest transmission cost, and can not take into account the dual optimization of time cost and space cost. Moreover, when using dynamic algorithms to achieve query optimization, the computing power is insufficient, and the algorithm often falls into the local optimization and can not get the global optimal solution that conforms to the characteristics of heterogeneous databases. Therefore, in this study, a query optimization method for massive heterogeneous data in the Internet of Things based on machine learning is designed.

2 Query Optimization Method of Massive Heterogeneous Data in Internet of Things Based on Machine Learning

2.1 Hierarchical Division of Massive Heterogeneous Data Query in Internet of Things

Query processing, as the most important link in the multi data integration system, directly affects the query efficiency of the system. How to ensure the correctness of query processing, and how to answer query efficiently are all the tasks to be completed by query processing [4]. The query processing of the massive heterogeneous database of the Internet of Things is more complicated. A query in a global mode needs to be decomposed into query fragments that can be run on each site. After the corresponding operations of these query fragments are executed at the site, the results must be returned to the global query Mode, and summarize the execution results of all the fragments as the final output of the system. But for users, query fragmentation, local processing, result summary processing and other processes are transparent. Users only care about the execution results of global queries. In the query optimization of a heterogeneous database system, the communication cost and the time complexity of local response must be considered at the same time. Therefore, these two types of query optimization

standards are often used at the same time, each of which depends on the execution environment and data size. The standard weights will vary accordingly.

In order to achieve the query effect required by users, the query process of massive heterogeneous database system is generally divided into four levels from the structure: query decomposition, data localization, global optimization and local optimization, as shown in the figure below (Fig. 1).

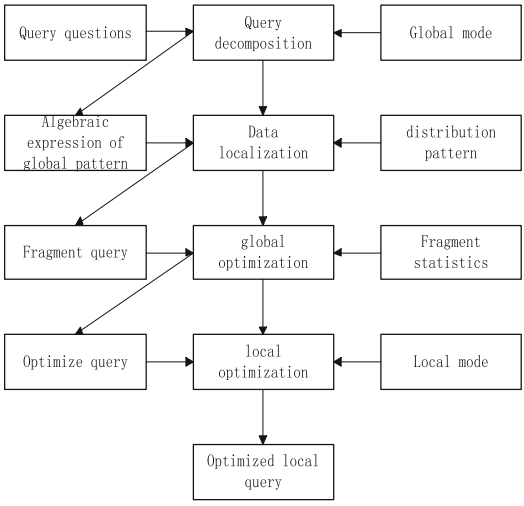


Fig. 1. Structure hierarchy

The most important work in query processing is query hierarchy. Query hierarchy refers to the decomposition of m-sql queries based on mediator mode into sub query sets based on wrapper mode [5]. In a multi-data integration system, the user query writes a query statement for the Mediator mode. There is no actual data in the Mediator mode. The query decomposition needs to convert the M-SQL query according to the specific situation according to the mapping relationship between the Mediator mode and Wrapper mode One or more queries based on Wrapper mode. Based on the above-mentioned query hierarchical division results of the massive heterogeneous data of the Internet of things, the massive heterogeneous data of the Internet of things is classified, and the classification results are used as the data base of the query optimization results.

2.2 Data Feature Extraction

After determining the query hierarchy of the Internet of things massive heterogeneous data, we need to generate a rich set of features for each query vocabulary, which represent the relationship between the candidate extension word and the original query, and the importance of the candidate extension word in the corpus. These features can be generally divided into two categories: the value of one type of feature depends on the specific input query, and the other type of feature is independent of the specific

query [6]. Since SVM is used for training, all features need to be normalized. In normalization, given a search topic and a set of candidate expansion words, the characteristics of each candidate expansion word are changed proportionally, and the characteristics of the current expansion word are normalized between 0 and 1 by the maximum value of the same characteristic in the training data.

Set in the physical network heterogeneous database, there are N query information $A : a_1, a_2, \ldots, a_n$; a_n candidate expansion word queue can be expressed as $U : ua_1, ua_2, \ldots, ua_n$; word frequency of u_n in document f_i is $g(u_n, f_i)$; word frequency of u_n in Y corpus Is $g(u_n, Y)$; the document frequency of the lexical item u_n in the corpus C is $g(u_n)$; the co-occurrence frequency of the lexical items u_n and u_m can be set to $co - ocur(u_n, u_m)$; the file length of the i th file f_i in the heterogeneous database is fl_i; $avgfl$ is the heterogeneous The average file length in the database. Use the above settings to obtain the various characteristics of IoT heterogeneous data.

The probability of the monolingual model is calculated by the Dirichlet smoothing method [7, 8] in machine learning. The parameter α used in the Dirichlet smoothing is set according to each evaluation data set, and α is set as the average document length of this test data set. The calculation formula is listed below. Here, do/n is used to model the probability P of the occurrence of word u_n in the corpus.

$$Unigram - fwature(t) = \sum_{f_i \in f} \frac{g(t, u_n) + \alpha \frac{do}{n}}{fl + \alpha} \tag{1}$$

This design mainly calculates the probabilities of the two unary language models, calculates the unary language probabilities on each feedback document, and then adds up these probability values. Consider all the feedback documents as a whole, and calculate the probability of the unitary model on this whole.

The BM25 weight is a well-known formula in the field of information retrieval. It is used to express the importance of this word in pseudo-feedback documents. The higher the value of BM25, the more important this word is in this document. This value indirectly depends on the original query item. For a given candidate extension word, the BM25 weight formula is as follows:

$$BM25 - fwature(t) = \frac{(h_i + 1)g}{h_i(1 - j) + j\frac{fl}{avgfl}} \log \frac{n - do + 0.5}{do + 0.5} \tag{2}$$

Among them, $h_i \in [1.0, 2.0]$ and j are usually set to 0.75. The weight of BM25 is similar to the calculation of the previous unary language model. There are two cases to calculate the characteristics of the weight of BM2s, that is, calculate the BM25 weight of the word on each pseudo-feedback document, and then add these weights together; Or take all the pseudo-feedback documents as a whole and then calculate the BM25 weight.

In this study, the main features of the global corpus are the monolingual model of the candidate extension word in the whole corpus and the gdo value in the whole corpus. The calculation formula is:

$$g * do(a_i) = g(u_n, Y) * \log \frac{Corpus - Size}{g(u_n)} \tag{3}$$

In order to calculate the similarity between the candidate expansion word and the original query, for a given candidate expansion word, extract mutual information and Pearson's chi-square statistics from the pseudo-feedback document [9, 10]. The higher the value of these statistics, the closer the relationship between the extended word and the original query word. The original query submitted by the user is composed of multiple keywords, so the similarity between the extension words and each keyword is accumulated here. As shown below, both statistics need to calculate the co-occurrence between the expansion word w and the original query term a, using the following statistics on the co-occurrence between the two words:

$$MI - fwature(t) = \sum_{w \in a} MI(a, w) \tag{4}$$

$$\beta - fwature(t) = \sum_{w \in a} \frac{n(c_{11}c_{22} - c_{21}c_{12})^2}{(c_{11}c_{12})(c_{11} + c_{21})(c_{12} + c_{22})(c_{21} + c_{22})} \tag{5}$$

Among them, (a, w) represents the query word a and the expansion word w. The joint probability on the pseudo feedback document set. c_{11} represents the co-occurrence of the query term a and the expansion term w; c_{12} represents the frequency where only the query term a appears without the expansion term w; Qiu: c_{21} represents the frequency where the query term a does not appear and only the expansion term w appears; c_{22} represents both the query word a does not appear, nor does the frequency of the expansion word w appear. Obtain the characteristics of the query information and the corresponding generated information through the above calculation, and set the collected characteristics to the database storage format.

2.3 Building a Data Dictionary

Through the research on the original data query method, it can be seen that the user query request is converted into a global query tree, and by accessing the global data dictionary, the relevant database information, site information and other global description information are obtained, so as to perform the subsequent query decomposition operation. In the process of setting up the global data dictionary [11, 12], there are some points to be noted, such as when a field appears in multiple tables, multiple records should be set up, otherwise conflicts will occur. If the table of the same field is distributed in the database of different sites, multiple records need to be established according to the site conditions.

In view of the problems in the use of the original query method, the global data dictionary is converted into a dynamic data dictionary. The dynamic data dictionary is similar to the global data dictionary but different. It can take query commands as input, and can use dynamic data fields to describe the dynamic status of global sites and relational tables of heterogeneous databases. A dynamic data dictionary mainly includes the following contents (Table 1):

Table 1. Dynamic data font information table

Information no. Name content	Information no. Name content	Information no. Name content
1	Serial number	
2	Site number	
3	Table number	Table numbers in order
4	Table name	
5	Number of records in the table	
6	Field name	The fields of the same table are arranged in order; if a field appears in multiple tables at the same time, these fields are matched to different tables and added to the dictionary
7	Whether the field has an index	
8	Field data length	
9	Select field means that the value of a field must meet a certain condition in a query	Select field means that the value of a field must meet a certain condition in a query
10	Is it the result field? In the query, the value of the field is the content displayed after the query	Is it the result field? In the query, the value of the field is the content displayed after the query
11	Whether it is a contact field means that in the query, the value of the field is the connection condition between the table and the table	Whether it is a contact field means that in the query, the value of the field is the connection condition between the table and the table
12	Number of values allowed for the field	
13	Network performance matrix The performance ratio of the site itself to other network connections and network transmission speeds	Network performance matrix The performance ratio of the site itself to other network connections and network transmission speeds
14	Computational Performance Array Computational performance indicators of sites in the network	Computational Performance Array Computational performance indicators of sites in the network

The dynamic data dictionary stores the dynamic state information of each site and the table in the site. When the site works normally, the network performance and computing performance status can be stored normally. If the site is abnormal or loses connection, it will be represented by infinity or a negative number. The data dictionary is introduced into the original data query method, and the rest use the original data query method to design content. At this point, the query optimization method for massive heterogeneous data of the Internet of Things based on machine learning is designed.

3 Method Application Test

3.1 Test Platform Construction

In order to effectively simulate the massive heterogeneous data network of the Internet of things, HBase is used to build the distributed data storage network of the Internet of things.

This article uses a completely distributed operating mode to build an experimental platform. The experimental platform built in this article is based on Hadoop-1.0.4 version, HBase uses a version based on HBase-0.94.8 modified source code and recompiled, the JDK version is jdk-6u45-linux-i586. The specific hardware and software configuration is as follows (Table 2):

Table 2. Database cluster configuration

Cluster distribution Cluster composition Parameters	Cluster distribution Cluster composition Parameters	Cluster distribution Cluster composition Parameters
operating system	UbuntuKylin 14.04	32-bit
Hardware parameters	CPU	Intel E7000(Binuclear 、2.66 GHz)
	Memory	Master node 2 GB, Slave node 1 GB
	hard disk	250 GB(5400 RPM)
HBaseConfiguration	hbase.hregion max.filesize	10 GB
	hbase.hregion memstore.flush.size	128 MB
	hbase.client write.buffer:	2 MB
MySQL Configuration	Cluster mode	Standalone
	default-storage-engine	INNODB
	innodbes buffer-pool size	1230 MB
	transaction-isolation:	READ-COMITTED

After completing the above configuration, start Hadoop and HBase in turn. Before starting Hadoop, you need to format the HDFS file system, and then start Hadoop and HBase on the cluster through the start all. SH and start HBase. Sh scripts respectively. When viewing the process through JPS after startup, if there are JPS, namenode, secondarynamenode, jobtracker, hmaster, hquorumpeer on the master node, and JPS, datanode, tasktracker, hregionserver, and hquorumpeer on the slave node, it indicates that the startup is successful. You can view the status of HDFS through http: / / masterhost: 50070 /, and HBase through http:! / masterhost: 60010 /. So far, the establishment and start-up of the experimental environment has been completed.

3.2 Test Methods

In order to ensure the validity of the method test, this paper studies the effects of these three methods by comparing the design method with traditional optimization methods and optimization methods designed by artificial intelligence technology. In this test, the test data set will be set as the test data sample. The specific data is shown below (Table 3).

Table 3. Test data set

Data set	Dataset a dataset B	Dataset a dataset B
Data set size	2048	10240
Path expression	1315	6560
Suffix array	14887	74380
Remove redundant data items	9311	44540

Use the design method and the other two methods to query the above data set, and set the query statement as shown below (Table 4).

Table 4. Query statement

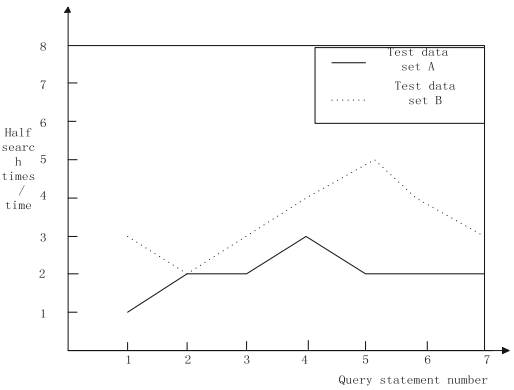
Query statement Matching method Category	Query statement Matching method Category	Query statement Matching method Category
X1	Backward	Resources
X2	Backward	Attribute
X3	Backward	Resources
X4	Backward	Resources
X5	Backward	Resources
X6	Backward	Attribute
X7	Forward	Attribute

Set the above query statement as the design method and the other two methods in the article, and query the test data set.

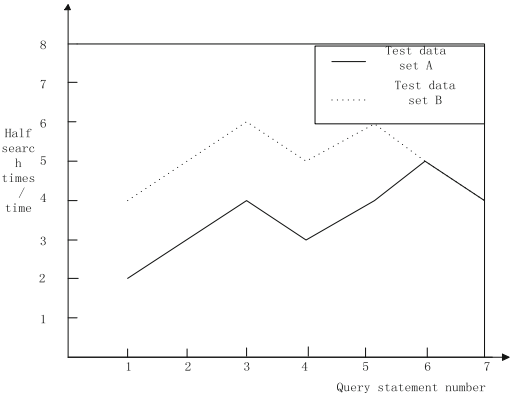
3.3 Test Indicators

In this test, the test index is set as follows: the search times in half and the execution time are used as the test index, and this index reflects the difference between the design method in the article and the other two methods.

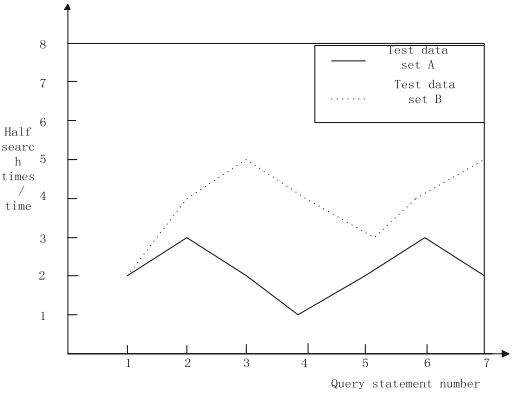
Half search times and execution time are the important embodiment of query ability of massive heterogeneous data. Through this index, we can directly understand the use effect of optimization method through data.



(a) Test results of design method in this paper

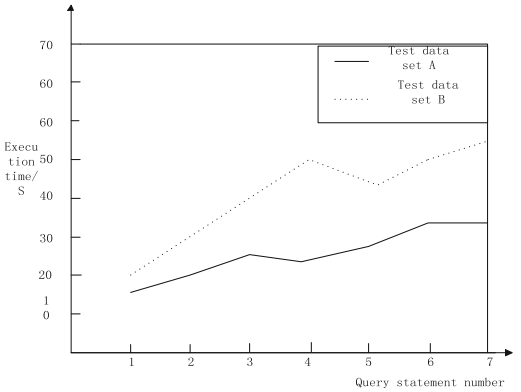


(b) Test results of traditional methods

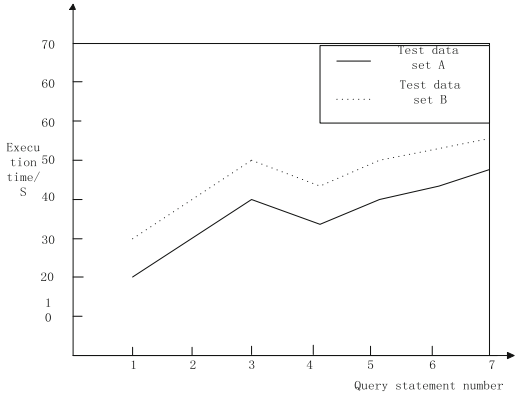


(c) Test results based on artificial intelligence method

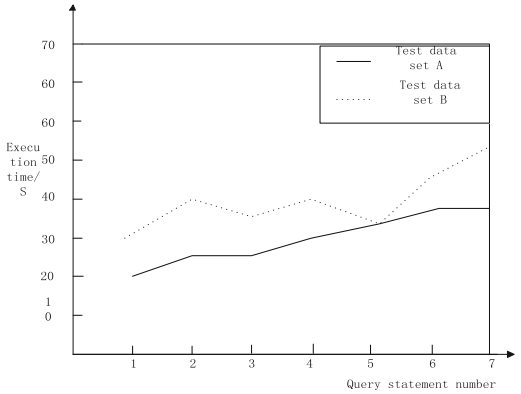
Fig. 2. Half-find test results



(a) Test results of design methods in the article



(b) Test results of traditional methods



(c) Test results based on artificial intelligence methods

Fig. 3. Execution time test results

3.4 Half-Find Test Results

According to the above test results, among the seven query statements, the method proposed in this paper needs the least number of half search. The reason is that the method proposed in this paper builds a data dictionary, which can reduce the number of search in half by indexing. It can be found from the data in the table that the method proposed in this paper is able to maintain the minimum number of halving searches even when the data set is significantly larger than the other two methods (Fig. 2).

3.5 Execution Time Test Results

In traditional methods and artificial intelligence methods, query processing time is almost the same, but in the design method of this article, it is half of the first two time. It can be seen from the data that under the premise that the query result is equal to the left and right comparison times, machine learning can greatly shorten the query time and improve the query efficiency. It can be seen that the method proposed in this paper has a good balance between query efficiency and accuracy (Fig. 3).

4 Conclusion

This paper studies the massive heterogeneous data processing of the Internet of things from a new perspective. While some achievements have been achieved, there are still some deficiencies and areas to be improved. In the future research, we should improve them. Finally, I hope that the work of this paper can play a reference role for the development of distributed heterogeneous data query processing and other related fields [13].

References

1. Hu, F., Li, C., Wang, M., et al.: SQL injection detection scheme based on machine learning. Comput. Eng. Des. **40**(06), 1554–1558 (2019)
2. Guo, S., Guo, Z., Hu, N., et al.: IoT-oriented heterogeneous data conversion model. Periodical Ocean Univ. China **49**(06), 140–146 (2019)
3. Li, G., An, J.: Internet of Things-oriented heterogeneous entities relation service model. Internet Things-Oriented Heterogen. Entities Relat. Serv. Model **46**(02), 131–140 (2019)
4. Zhang, P., Wang, Y., Jiang, M., et al.: Spatio temporal data retrieval and prediction system based on K-D tree and machine learning. Comput. Eng. Softw. **39**(08), 215–218 (2018)
5. Sang, H., Guo, W.: Research on key model of index semantic query based on massive heterogeneous data. J. Fuzhou Univ. (Nat. Sci. Ed.) **46**(03), 324–329 (2018)
6. Xia, R.: Design of machine learning web service engine based on spark. Command Control Simul. **40**(01), 113–117 (2018)
7. Wang, H., Dai, B., Li, C., et al.: Query optimization model for blockchain applications. Comput. Eng. Appl. **55**(22), 34–39+171 (2019)
8. Liu, S., Li, Z., Zhang, Y., et al.: Introduction of key problems in long-distance learning and training. Mob. Netw. Appl. **24**(1), 1–4 (2019)

9. Le, Y.: Design and research of query optimization algorithm for large scale database. Bull. Sci. Technol. **35**(09), 66–69+74 (2019)
10. Shuai, L., Gelan, Y.: Advanced Hybrid Information Processing, pp. 1–594. Springer International Publishing, Heidelberg (2019)
11. Xing, B., Lv, M., Jin, P., et al.: Energy-efficiency query optimization for green datacenters. J. Comput. Res. Dev. **56**(09), 1821–1831 (2019)
12. Fu, W., Liu, S., Srivastava, G.: Optimization of big data scheduling in social networks. Entropy. **21**(9), 902 (2019)
13. Liu, S., Lu, M., Li, H., et al.: Prediction of gene expression patterns with generalized linear regression model. Front. Genetics **10**, 120 (2019)