International Journal of Database Theory and Application
Vol.8, No.3 (2015), pp.203-214
http://dx.doi.org/10.14257/ijdta.2015.8.3.17

An Efficient Distributed Data Management Method based key Columns Partition Preprocessing

Xu Tao[1], Zhang Wei [2,3*] and Li Baolu[2]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[2] School of Computer Science, Beijing Information Science & Technology University, Beijing 100101, China
[3] Beijing Key Laboratory of Internet Culture and Digital Dissemination Research, Beijing Information Science & Technology University, Beijing 100101, China
t-xu10@mails.tsinghua.edu.cn, zhwei@bistu.edu.cn, libaolu0821@163.com

Abstract

With the development of mobile internet and social network, the scale of structured data have been increasing to PB level and above rapidly, while the query performance is greatly reduce. The efficiency of query optimization on large-scale datasets is currently a research focus in both academia and industry. In this paper, we present a distributed data management method, designed to improve query performance, called KCSQ. KCSQ analyses historical SQL commands, deduces statistics using frequency and the coupling degree of tables and table columns, and confirms the key column based on statistical evidence. When importing new tables into the HDFS, the data are divided into different blocks according to their key column. Any query on these columns can reduce the amount of data to be queried and the number of working nodes and thus effectively improves the throughput rate of the system.

***Keywords**: HDFS, SQL, massive structured data, key column, data partition*

1. Introduction

With the rapid development of the internet, data are yielded at an increasingly high speed [1]. Statistics from IDC suggest that the amount of data created and replicated globally in 2011 reached 1.8 ZB [2], which is far more than the total data size of all printed material throughout history. Research concerning big data has received much attention. The larger the data scale, the more difficult the processing, and the higher the value of data explored. The major internet companies and open-source groups have developed, and shared, a number of technologies and systems, such as GFS [3], MapReduce [4], HDFS [5], Hive [6], *etc.*

With the rise of social networking and e-commerce, structured data grows rapidly to PB level or higher, which brings various problems and challenges for its management and analysis on such a large-scale. For structured data, relational database is undoubtedly the classical, and most popular, database system (*e.g.* Oracle, MySQL, SQLServer, and DB2). In 1970, Codd [7] first proposed a new model of the relationship, which sparked research into the relational method and theory of databases. After decades of development, relational database have come to be widely used in various types of information management systems and business application systems [8], and have become an effective storage and analysis tool for data warehousing [9]. However, the system expense of relational database for index construction and transaction mechanisms increases sharply when dealing with PB level data. Owing to the high cost and poor scalability, relational database appear weak when processing massive amounts of data and running large-scale concurrent applications.

In recent years, HDFS has become the most common big data storage system, and those analysis technologies and tools based on HDFS have also matured and are being constantly perfected. At present, for massive structured data, researchers mainly focus on the optimization of parallel database and HDFS-based technology improvement, such as Dremel [10], Impala [11], Spark [12], BlinkDB [13], *etc*. In specific processing scenarios, these systems show better performance querying on PB level structured datasets. However, these systems make use of the original data storage format of HDFS that is designed to deal with unstructured data, without optimization for massive structured data.

In this paper, we present a distributed data management method that is designated KCSQ. KCSQ analyses historical SQL commands, infers statistics from the frequency of use and degree of coupling of tables and their columns, and confirms the identity of the key column based on these statistical results. On this basis, it classifies the data in accordance with the key column to improve the query performance. We packaged KCSQ into middleware and embedded it into HDFS, which can work on other HDFS-based systems with minimal changes to achieve interactive query abilities.

2. Related Work

Massive structured data has gradually exceeded the processing scale of traditional database. This problem can be solved by research into parallel data, to some extent. The query performance of parallel data is closely related to the data distribution before processing.

Li proposed a dynamic multi-dimensional data distribution method for parallel database [14], which forcefully supported various parallel data operation algorithms in a dynamic database. Data warehousing is another massive structured data processing model. Li et al. proposed a multi-dimensional data model for a data warehouse [15]. This model could fully express the structure and semantics of the complex data in a data warehouse and effectively support on-line analytical processing (OLAP) applications.

Impala [11] is an MPP (massive parallel processing) SQL query engine developed by Cloudera. It provides an interactive query interface directly on massive Hadoop data stored in HDFS or HBase. However, for join queries and complex queries involving the forwarding of intermediate results, Impala does not improve the query performance. The query processing will even fail if the size of the intermediate results goes beyond the memory capacity. Furthermore, Impala provides an interface for HiveQL (SQL-like) but does not support standard SQL. Dremel [10] is an interactive data analysis system proposed by Google. It can execute ad-hoc queries on PB level datasets in seconds. To achieve such high performance, Dremel depends mainly on three aspects: using a new model that supports nested data; combining multi-level execution trees and a columnar storage format; and an ability to manage large-scale clusters. As the report engine of Google BigQuery, Dremel makes up for the disadvantages of MapReduce. However, Dremel is mainly meant for nested data optimization, and does not perform very well when processing fact tables and dimension tables.

Although the storage and query of massive data are achieved using distributed parallel queries, invalid query tasks are prone to be generated, *i.e.*, when querying a few records from a table, all the data blocks of the table are queried in parallel using the current processing mode. The query on the data blocks excluding result information are invalid query tasks. Pan et al. proposed a data distribution strategy for multi-dimensional datasets in a cloud computing environment [16]. In this strategy, a table is divided into a base table (the truth table) and clustered tables and a query is distributed to each data node. If the query results have been stored to a clustered table in advance, the query is returned directly. Otherwise, equivalence sets are queried or a recursion algorithm is called. Wang et al. proposed a multi-dimensional indexing structure, RT-CAN [17]. By indexing, the nodes that possibly contain result information are obtained to reduce the amount of queried data and improve the throughput rate. Wen et al. proposed a query-oriented database data distribution strategy (SOD) [18]. By query statistics, the querying law of a given table is obtained. Moreover, the sheets that are frequently and jointly queried are saved together to reduce the amount of data transmission over the network during the query while improving query performance.

3. Key Column-based Split and Query (KCSQ)

Based on the application of Hadoop to cloud computing, and mass structured processing, we propose a key column-based data partition and an efficient query task generation method. The key column is found from statistics relating to the historical query operations. The column with the highest usage frequency is deemed to have been the key column. Efficient query task here refers to query tasks generated for data blocks containing results.

Data sheets are stored in HDFS in block form. In cases querying all of the data blocks in each query, invalid queries may be produced since some data blocks do not contain the result. To solve this problem, the importation of the database sheet to HDFS employs a key column-based data partition method. The specific process is as follows:

(1) Determine the key column according to historical query information and classify the key columns into *M* areas according to values or ranges, *i.e.*, classifying the universal sets of the values of the key column into *M* areas.

(2) Pre-process the records in the sheet according to the key column and classify the records into different areas according to the valuation condition of the key column.

(3) Cluster the data according to the number of the key columns meeting the area and recording metadata information after partitioning each area (*i.e.*, the ID, area, and key column numbers meeting the area of each data block).

(4) Generate efficient query tasks for the query containing key columns.

4. Design and Realization of KCSQ

4.1 Sqoop

As an open source data import tool, Sqoop [19] can import sheet data in a database into HDFS. The following command is used as an example here to explain the data import process:

--connect jdbc:mysql://IP:3306/db_name - username root - password 111111 - sheet sheetname - hive-import -m 2

(1) Generating a class corresponding to the structure of the sheet that needed to be imported by searching and linking mysql using jdbc. The class is mainly used for serialisation and deserialisation.

(2) According to the parameters followed by *-m* in the command, determining the MapReduce task generated (only map tasks are used in the importing process) and then writing data into HDFS.

(3) Establishing a new cloud hive script file, creating sheets, and loading data.

During importation, data are divided into slices according to the primary key (e.g. record ID). Firstly, Sqoop finds out the max (ID) and min (ID) from the sheet. By dividing the difference Val between max (ID) and min (ID), the number of entries contained in each slice can be obtained. For example, max (ID) = 2000 and min (ID)

= 1, the ranges are (1-1000) and (1001-2000). Then two map tasks are initialized for data importing, and the records belonging to the two ranges are taken as the Result Set and imported into HDFS respectively.

4.2 Key Column-based Data Partition

When importing the database sheet into the HDFS, multiple key column-based data partition is conducted as follows:

(1) Statistically analyzing the historical query on the database and sequencing the query requirements in descending order according to the column usage frequency.

(2) Determining the area number M for data storage.

(3) Dividing the value range of the column into M areas and determining the KeyColumn (simply recorded as K below) in descending order according to usage frequency.

(4) Assuming that N key columns are selected ($0 < N \leq M$); representing the value of the area of the key column of each record by K(x, y), where, x represents a key column and y represents its area. For instance, K(1,1) signifies that the first key column is valued in the first area.

(5) As for arbitrary record in the database sheet, we determined the first column as the standard key column and judged the area of the records in the sheet, as shown in Fig. 1.

5.1) Determining K(x, y) for each key column.

5.2) Conducting statistical analysis of the values of y obtained in Step 5.1 by and finding the frequency of appearance of each y value.

5.3) Given N values for y, determining the area of y according to the first column.

5.4) Setting the area of the y value at its maximum amount as the area of this record, if the value variety number of y is less than N and y merely has one maximum value.

5.5) Determining the area of the y value of the key column with a small number as the area of this record is needed, if the value variety number of y is less than N and y has more than one maximum value. For example, when N = 5, M = 6, the statistical results of a record are K(1,1), K(2,1), K(3,2), K(4,2), and K(5,6); *i.e.*, key columns 1 and 2 are valued in Area 1; key columns 3 and 4 are valued in Area 2; and key column 5 is valued in Area 6. Therefore, with y values of key columns 1 and 2 as standard, the record should be stored in Area 1.

Through the above method, any record can be divided into a single area. As shown in Fig. 2, data are clustered according to the conditions classifying them into each region. There are N possibilities for the classification condition of each region:

(1) The value of N key columns satisfies the area.

(2) The value of N - 1 key columns satisfies the area.

(3) The value of N - 2 key columns satisfies the area.

…

(N - 1) The value of 2 key columns satisfies the area.

(N) The value of 1 key column satisfies the area.

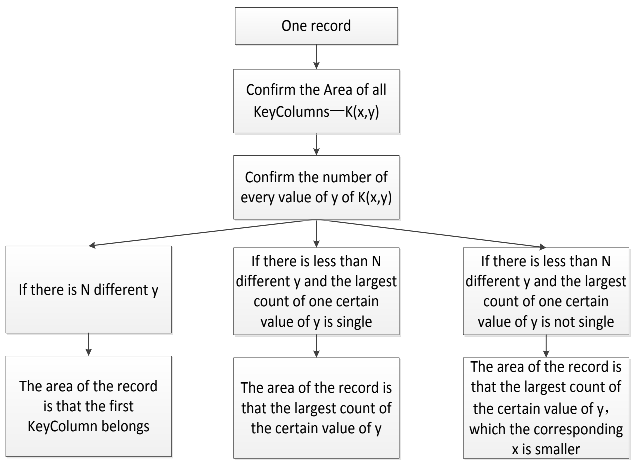

One record
Confirm the Area of all KeyColumns—K(x,y)
Confirm the number of every value of y of K(x,y)
If there is N different y
If there is less than N different y and the largest count of one certain value of y is single
If there is less than N different y and the largest count of one certain value of y is not single
The area of the record is that the first KeyColumn belongs
The area of the record is that the largest count of the certain value of y
The area of the record is that the largest count of the certain value of y，which the corresponding x is smaller

Figure 1. Process of Judging the Area of a Record

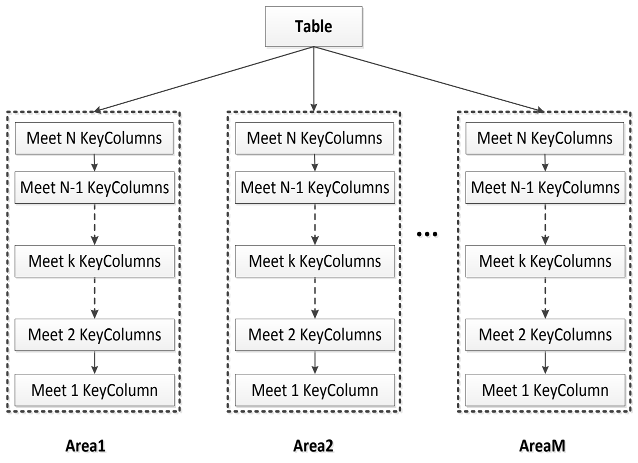


Figure 2. Result of Judging the Area of a Record

4.3 Storage and Application of Metadata

According to the storage and processing of the metadata [20][21] in Hadoop, the Area of each data block and the number of key columns that satisfied the current Area are recorded during importation: this information is stored by generating an attached table named DB_TABLENAME_META. Here, DB_TABLENAME_META mainly contained three fields: BlockID, Area, and KNum. BlockID represents the number of the data block in HDFS; Area represents the region of the data block; KNum is the number of key columns that satisfy the current region in the data block.

In case of receiving a query request from the client, the query request is firstly processed. If the query request does not contain key columns, we query the whole sheet; otherwise, we need to find the maximum number of key columns (KNum) with small number within the Area. Using Area and KNum, it is possible to obtain the BlockID containing the final result information from the table DB_SHEET_META and, in turn, generate efficient query tasks for those data blocks containing result information.

4.4 The Generation Process of Efficient Query Tasks

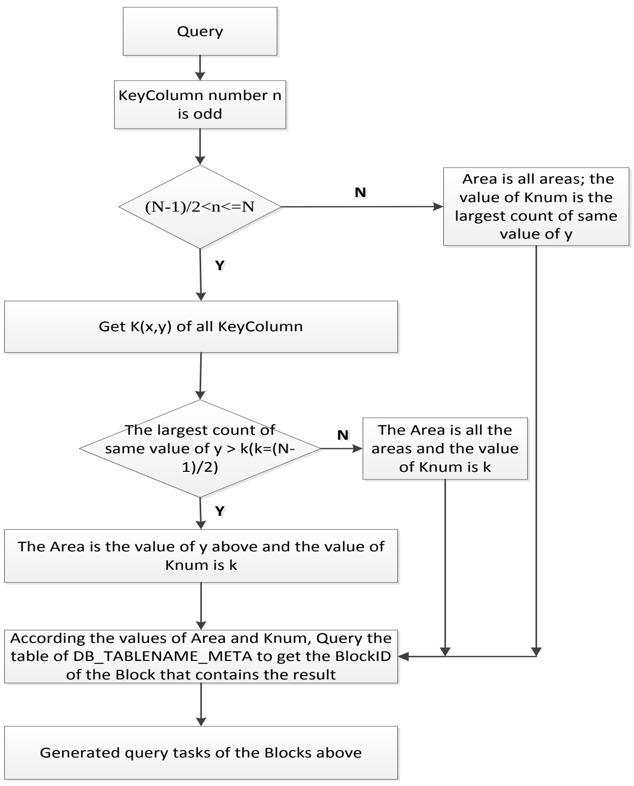

Query
KeyColumn number n is odd
(N-1)/2<n<=N
N
Area is all areas; the value of Knum is the largest count of same value of y
Y
Get K(x,y) of all KeyColumn
The largest count of same value of y > k(k=(N-1)/2)
N
The Area is all the areas and the value of Knum is k
Y
The Area is the value of y above and the value of Knum is k
According the values of Area and Knum, Query the table of DB_TABLENAME_META to get the BlockID of the Block that contains the result
Generated query tasks of the Blocks above

Figure 3. Generation of Efficient Query Tasks

After data partitioning using the method described above, efficient query tasks can be found from the query containing key columns according to the attached metadata sheet.

It is supposed that there are n key columns in the query conditions:

If N is odd and $(N - 1)/2 < n \leq N$, the generation process of an efficient query task is shown in Fig. 3:

(1) Determining K(x, y) for each key column.

(2) Conducting statistical analysis of values of y according to value.

(3) If the y values of n key columns are identical and all belonged to Area m, directly query the data that simultaneously satisfied n key columns in Area m.

(4) If the y values of n - 1 key columns are identical (at most), and all belonged to Area m, directly query the data that simultaneously satisfied n - 1 key columns in Area m.

(5) If the y values of k ($k > (N - 1)/2$) key columns are identical and all belonged to Area m, we directly query the data that simultaneously satisfied k key columns in Area m.

(6) If the y values of k ($1 < k \leq (N - 1)/2$) key columns are identical (at most) and all belonged to Area m, we directly query the data that simultaneously satisfied k key columns in all regions.

If N is odd and $1 < n \leq (N - 1)/2$, the generation of an efficient query went as follows: if the y values of k ($1 < k \leq (N - 1)/2$) key columns are identical (at most) we directly query the data that simultaneously satisfied k key columns in all regions.

If N is even and $N/2 < n \leq N$, the efficient query task is generated as follows:

(1) Determining K(x, y) for each key column.

(2) Conducting statistical analysis of values of y according to value.

(3) If the y values of n key columns are identical and all belonged to Area m, directly query the data that simultaneously satisfied n key columns in Area m.

(4) If the y values of n - 1 key columns are identical (at most), and all belonged to Area m, directly query the data that simultaneously satisfied n - 1 key columns in Area m.

(5) If the y values of k ($k > N/2$) key columns are identical and all belonged to Area m, we directly query the data that simultaneously satisfied k key columns in Area m.

(6) If the y values of k ($1 < k \leq N/2$) key columns are identical (at most) we directly query the data that simultaneously satisfied k key columns in all regions.

If N is even and $1 < n \leq N/2$, the generation process of an efficient query tasks went as follows: if the y values of k ($1 < k \leq N/2$) key columns are identical (at most) we directly query the data that simultaneously satisfied k key columns in all regions.

5. Case Verification

In the verification, we assume that N = 3 and M = 3, that is, there are three key columns and the value, or value range, of each column is classified into three areas. As shown in Table 1, it is supposed that the datasheet Record has four columns: Id, City, Age, and Month, respectively. A total of 27 data points are generated. Each data represents a type of record.

City, Age, and Month are selected as key columns. According to KCSQ, the data in three areas are generated, as shown in Table 2-4. In each area, data are clustered according to the number of key columns that satisfied the area. In Area1, the three key columns are valued by Beijing, 1-9, and Jan. respectively; in Area2, the three key columns are valued by Tianjin, 10-19, and Feb. respectively; in Area3, the three key columns are valued by Shanghai, 20-29, and Mar. respectively.

The query containing key columns is firstly pre-processed to obtain Area and KNum, which are then used to determine the data blocks containing the information and generate an efficient query task. It is supposed that there are six queries:

Table 1. Typical Record

Id	City	Age	Month	Id	City	Age	Month
1	Beijing	7	Jan	15	Tianjin	12	Mar
2	Beijing	8	Feb	16	Tianjin	20	Jan
3	Beijing	9	Mar	17	Tianjin	21	Feb
4	Beijing	12	Jan	18	Tianjin	22	Mar
5	Beijing	15	Feb	19	Shanghai	9	Jan
6	Beijing	16	Mar	20	Shanghai	8	Feb
7	Beijing	21	Jan	21	Shanghai	7	Mar
8	Beijing	22	Feb	22	Shanghai	15	Jan
9	Beijing	23	Mar	23	Shanghai	16	Feb
10	Tianjin	5	Jan	24	Shanghai	18	Mar
11	Tianjin	6	Feb	25	Shanghai	22	Jan
12	Tianjin	7	Mar	26	Shanghai	23	Feb
13	Tianjin	10	Jan	27	Shanghai	26	Mar
14	Tianjin	11	Feb				

Queries 1-3 contain three key columns. By pre-processing, it is found that Query 1 corresponded to the data block for which KNum = 3 in Area1; Query 2 corresponded to the data block for which KNum = 2 in Area1; Query 3 corresponded to the data block for which KNum = 1 in Area1. When generating query tasks, it is only necessary to query corresponding data blocks.

Queries 4 and 5 contain two key columns. By pre-processing, it is found that Query 4 corresponded to the data block for which KNum = 4 in Area1; Query 5 corresponded to the data block for which KNum = 1 in Area1. When generating query tasks, it is only necessary to query corresponding data blocks.

Query 6 contains one key column. By pre-processing, it is found that Query 6 corresponded to the data block for which KNum = 1 in all three areas. When generating query tasks, it is only necessary to query corresponding data blocks.

This method effectively reduces the amount of data to be queried and the number of working nodes in a given query: it thus effectively improves the throughput rate of the system.

Table 2. Area1

		Area1 (Beijing, 1-9, Jan.)		
	Id	City	Age	Month
Knum = 3	1	Beijing	7	Jan.
Knum = 2	2	Beijing	8	Feb.
	3	Beijing	9	Mar.
	4	Beijing	12	Jan.
	7	Beijing	21	Jan.
	10	Tianjin	5	Jan.
	19	Shanghai	9	Jan.
Knum = 1	6	Beijing	16	Mar.
	8	Beijing	22	Feb.

Table 3. Area2

		Area2 (Tianjin, 10-19, Feb.)		
	Id	City	Age	Month
Knum=3	14	Tianjin	11	Feb
Knum=2	13	Tianjin	10	Jan
	15	Tianjin	12	Mar
	11	Tianjin	6	Feb
	17	Tianjin	21	Feb
	5	Beijing	15	Feb
	23	Shanghai	16	Feb
Knum=1	12	Tianjin	7	Mar
	16	Tianjin	20	Jan

Table 4. Area3

		Area3 (Shanghai, 20-29, Mar.)		
	Id	City	Age	Month
Knum=3	27	Shanghai	26	Mar
Knum=2	25	Shanghai	22	Jan
	26	Shanghai	23	Feb
	21	Shanghai	7	Mar
	24	Shanghai	18	Mar
	9	Beijing	23	Mar
	18	Tianjin	22	Mar
Knum=1	20	Shanghai	8	Feb
	22	Shanghai	15	Jan

6. Conclusion

KCSQ classifies data according to some key columns and records related metadata information when importing the database sheet files into HDFS. It pre-processes query requests containing key columns to generate an efficient query task, and effectively improves the processing efficiency and system throughput rate for massive structured data operations. Moreover, it is packaged into middleware and embedded within HDFS, which can work on other HDFS-based systems with minimal changes to achieve an ability to conduct interactive queries.

Acknowledgements

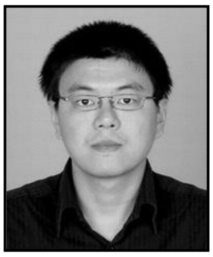
Funding for this research was provided in part by the Program for the Top Young and Middle-aged Innovative Talents of Higher Learning Institutions of Beijing, the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research (ICDD201306), and The Project of Construction of Innovative Teams and Teacher Career Development for Universities and Colleges Under Beijing Municipality (IDHT20130519). We like to thank anonymous reviewers for their valuable comments.

References

[1] R. Maggiani, "Cloud Computing Is Changing How We Communicate", IEEE International Professional Communication Conference, IPCC, **(2009)**.
[2] F. Yingxun, L. Shengmei and S. Jiwu, "Secure cloud storage system and summary of key technologies", Computer Research and Development, **(2013)**, vol. 50, no. 1.
[3] S. Ghemawat, H. Gobioff and S. T. Leung, "The Google file system", ACM SIGOPS Operating Systems Review, vol. 37, no. 5, **(2003)**, pp. 29-43.
[4] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters", Commun. of ACM, vol. 51, no. 1, pp. 107-113, **(2008)**.
[5] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System", Proceedings of IEEE Conference on Mass Storage Systems and Technologies (MSST), **(2010)**, pp. 1-10.
[6] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive - a petabyte scale data warehouse using Hadoop", IEEE 29th International Conference on Data Engineering (ICDE), **(2010)**, pp. 996-1005.
[7] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, vol. 13, no. 6, **(1970)**, pp. 377-387.

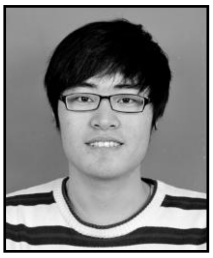
[8] L. Bellatreche and K. Y. Woameno, "Dimension table driven approach to referential partition relational data warehouses", Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP, **(2009)**; New York, NY, USA.
[9] J. Han , J. Y. Chiang , S. Chee , J. Chen , Q. Chen , S. Cheng , W. Gong , M. Kamber , K. Koperski , G. Liu , Y. Lu , N. Stefanovic , L. Winstone , B. Xia , O. R. Zaiane , S. Zhang , and H. Zhu, DBMiner, "a system for data mining in relational databases and data warehouses", Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, **(1997)**, pp. 326-336.
[10] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, Dremel, "Interactive analysis of webscale datasets", Proceedings of the VLDB Endowment, vol. 3, n. 1-2, pp. 330-339, **(2010)**.
[11] Impala project, http://impala.io/.
[12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing", Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, **(2012)**.
[13] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, BlinkDB, "Queries with Bounded Errors and Bounded Response Times on Very Large Data", Proceedings of the 8th ACM European Conference on Computer Systems, **(2013)**, New York, NY, USA.
[14] J. Li, "A Dynamic and Multidimensional Declustering Method for Parallel Databases, Journal of Software, 1999, 10(9):909-916.
[15] J. Li and H. Gao, Multidimensional Data Model for Data warehouses", Journal of Software, vol. 11, no. 7, **(2000)**, pp. 908-917.
[16] H. Pan, L. Gao, Y. Liu, "Research on Query and Data Distribution Strategy of Data Cube in Cloud Computing Environment", Microelectronics & Computer, vol. 29, no. 8, **(2012)**.
[17] J. Wang, S. Wu, H. Gao and B. C. Ooi, Indexing Multi-dimensional Data in a Cloud System, in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, ACM, **(2010)**, pp. 591-602.
[18] M. Wen and Z. Ding, "Selection Oriented Database Data Distribution Strategy for Cloud Computing", Computer Science, **(2010)**, vol.37, no. 9.
[19] Apache Sqoop project, http://sqoop.apache.org/.
[20] A. Chandrasekar, K. Chandrasekar, H. Ramasatagopan, A. R. Rafica and J. Balasubramaniyan, "Classification Based Metadata Management for HDFS", Proceeding of High Performance Computing and Communication, **(2012)**, pp. 1021-1026.
[21] C. L. Abad, H. Luu, N. Roberts, K. Lee, Y. Lu and R. H. Campbell, "Metadata Traces and Workload Models for Evaluating Big Storage Systems", Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, **(2012)**, pp. 125-132.

Authors



Xu Tao, he is currently a PhD candidate in Department of Computer Science and Technology of Tsinghua University. He received his MS degree from Tsinghua University in 2005. His research interests include massive storage, large-scale distributed system, and stream computing.

Zhang Wei, he is now an associate professor in School of Computer Science at Beijing Information Science & Technology University. He is a senior member of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research. His research interests include big data storage and security, software hardware co-design.



Li Baolu, he is currently a master student in School of Computer Science at Beijing Information Science & Technology University. His research focuses on big data storage.