METECLOUD: A PRIVATE CLOUD PLATFORM FOR METEOROLOGICAL DATA STORAGE USING HADOOP

[1]Xue Shengjun, [*2]Xu Xiaolong, [3]Wang Delong, [4]Zhang Jie, [5]Ji Feng

[1,*2, 3, 4, 5] School of Computer and Software

Nanjing University of Information Science & Technology

[1] Jiangsu Engineering Center of Network Monitoring

Nanjing University of Information Science & Technology

Nanjing 210044, China

Emails: xlxu1988@gmail.com

Submitted: Dec. 26, 2012 ***Accepted: Mar. 19, 2013*** ***Published: Apr. 10, 2013***

Abstract- With the increasing popularity of open-source platform Hadoop, the meteorological industry is available to create a Meteorological Cloud (MeteCloud) platform to store and deploy applications. In this paper, we propose an idea to build the MeteCloud platform for meteorological departments using Hadoop. We also present a backup policy for meteorological data. In addition, one kind of storage process of the meteorological A-format file is presented. Furthermore, we experiment with one-year historical data on the platform by varying many related parameters such as the number of nodes, meteorological records and fields. Finally, the proposed MeteCloud platform proves to be efficient and suitable for the storage of meteorological data.

Index terms*:* MeteCloud, Hadoop, Hive, meteorological data, transfer storage

I. INTRODUCTION

Cloud computing has brought the telecom, transportation and other industries into prosperity, and the cloud platform of them has been used popularly [1]. It is necessary and potential for Chinese meteorological industry to build its private cloud platform. There is a large amount of historical meteorological data. Moreover, with the rapid increase in the scale of automatic weather stations, weather radar, rainfall calibration station and meteorological satellite, the data is growing at the same time. Currently, each province has its own independent data storage system, so meteorological industry lacks an internal information sharing platform. And a number of machines in the departments are standing idle which leads to a waste of resources.

Nowadays, a variety of commercial public cloud platforms are presented to provide storage and compute services. These companies have reliable storage structures such as Microsoft, Amazon, Google and Yahoo so that users do not need to worry about data loss or lack of storage space [2] [3]. Besides public cloud, private cloud is also applied by users who do not want to put their data in public environment to prevent data theft and viruses and the users can program their own applications with the definite scheduling policies [4] [5]. In this mode, users need to provide the infrastructures and build the platform by their own solutions.

Considering the data security and confidentiality, the meteorological industry needs to build their own cloud platform. Hadoop, inspired by Google File System, Google's MapReduce model and other projects, is an Apache open-source project that supports distributed storage and distributed applications. Hadoop is composed of several sub-projects such as Hadoop File System (HDFS), MapReduce, HBase, Hive and Pig. HDFS distributes data blocks and copies across nodes with high performance and reliability [6]. Hive is a NoSQL solution, and its storage schema is different from relational database. The tables with Hive are stored in the format of text file on HDFS. However, Hive provides a query language HQL (Hive Query Language) which uses the idea of MapReduce to execute query process [7].

In this paper, according to current storage status of Chinese meteorological departments in China, a cloud platform for meteorological industry will be built with Hadoop and we call it MeteCloud. In order to get and query text-format meteorological daily data more conveniently for research than before, it will be designed to restore in Hive. And the process of creating tables and query

data will be described in detail. Moreover, we will also need to experiment with variable scale of clusters and variable fields.

The rest of the paper is organized as follows: Section II describes related works. The construction and descriptions of MeteCloud platform is represented in Section III. In Section IV, meteorological data storage process is proposed. Section V presents the related experimental evaluation with proposed methods. Section VI concludes the paper and discusses the future works.

II. RELATED WORKS

Recently, a large amount of researches have been done in cloud storage with various solutions. This section provides a brief review of some these related works.

The cloud service renters typically stores and retrieves data from vendors via internet while using cloud storage services [8]. There is no need for users to buy, run, upgrade or maintain data storage systems and users do not need to worry about insufficient storage space [9]. Most cloud service providers such as Amazon [10], OpenStack [11], IBM Smart Business [12] and Oracle [13] use virtualization technology to enhance ability to share resources.

Hadoop has been used in many domains to store and process applications without virtualization [14]. The overview, architecture and components of Hadoop, HCFS (Hadoop Cluster File System) and MapReduce programming model are described in [15]. In [16], a standard bag-of-visual-words image indexing pipeline is scaled across hadoop clusters. The ever growing technology has resulted in the need for storing and processing excessively large amounts of data on cloud, and an efficient resource scheduling method using hadoop is proposed to tackle it [17].

However, Hadoop is not suitable for handling massive small files which leads to the appearance of metadata-aware storage architecture for massive small files and it is presented in [6].

In addition to MapReduce, Hive is also applied to some applications. Hive, an open-source data ware-housing solution built on top of Hadoop is presented by Facebook. Hive supports queries expressed in a SQL-like declarative language HQL, which are compiled into map-reduce jobs executed on Hadoop [18]. A rigorous evaluation study for Hive use peta-byte scale data, and workload that needs to scan over entire dataset in [19]. It is to develop an approach for improving the performance of the HQL queries executed in the Hive framework in [20].

Meteorological data in China is stored in various information systems. And data backup copies need the tapes of heavy capacity. However, there is no a resource sharing platform of meteorological industry and there are many idle machines in the meteorological departments.

III. CONSTRUCTION AND DESCRIPTION OF METECLOUD PLATFORM

a. Architecture of Hive and Hadoop Cluster

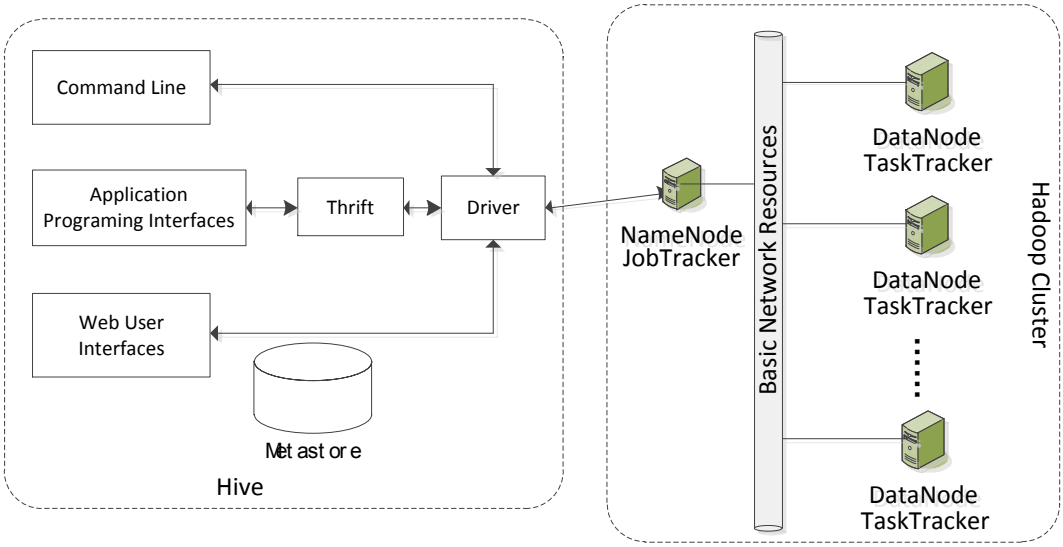


Figure 1. Architecture of Hive and Hadoop Cluster

Figure 1 shows the architecture of Hive and Hadoop Cluster. Hive, deployed on Hadoop, is a data warehouse system which promotes data summarization and analysis of large datasets.

(1) Hadoop cluster

Hadoop cluster consists of one NameNode running JobTracker and some DataNodes running TaskTracker. Both high performance computers and common servers can be set as nodes in clusters.

(2) Interfaces

Hive is running on the top of Hadoop, including many components. Command line, application programing interfaces such as JDBC and web user interfaces are external interfaces, which can be used to give the operation process of submitting their data and executing their application by users.

(3) Thrift

Thrift is a server to expose a very simple client API to execute HQL statements which allows traditional MapReduce programmers to plug in their custom mappers and reducers [18].

(4) Driver

Driver manages the life cycle of a HQL statement during compilation, optimization and execution [18]. When user query data from Hive, the process is referred to Hadoop cluster a job and it is executed by MapReduce. MetaStore

(5) Metastore

Metastore is the system catalog containing metadata specified during table creation and reused when the table is referenced in HQL about the tables stored in Hive [18].

b. Construction of MeteCloud platform

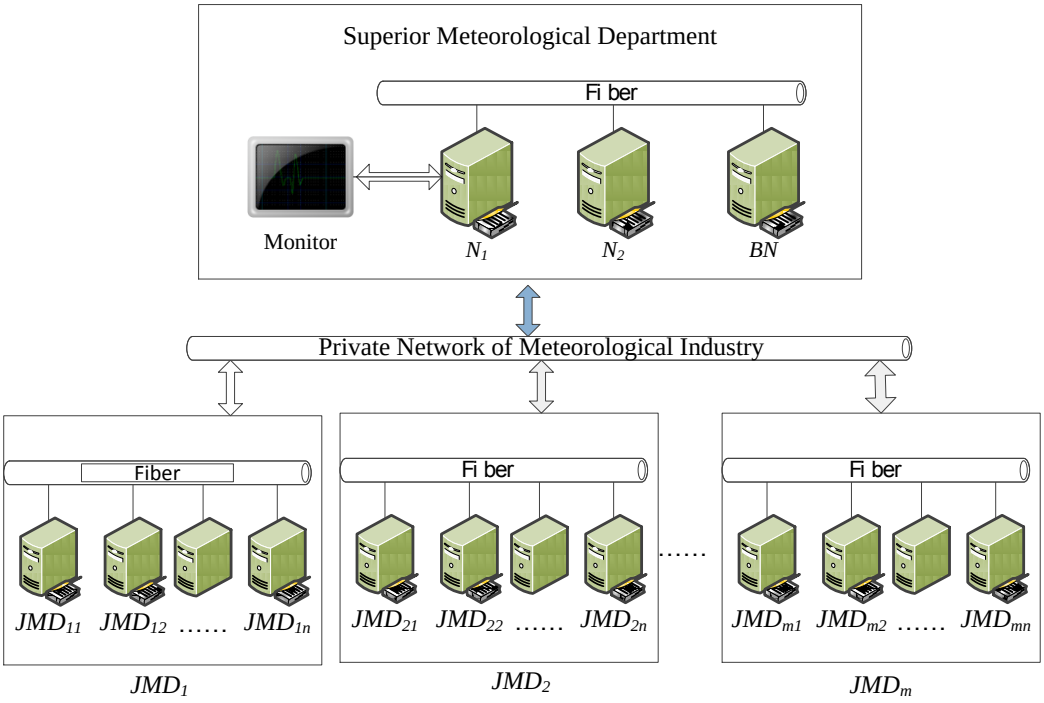


Figure 2. Construction of Meteorological Cluster

The construction of meteorological cluster is based on the aim of integrating compute, storage and transfer resources of meteorological industry. In superior meteorological department, 3 nodes $\{N_1, N_2, BN\}$ are loaded with Hadoop File System and Hive database. The cluster is designed as

master-slave mode, including one primary node N_1 and a large number of data nodes. The primary node N_1 presented is used to administrate the namespace of the file system and record the block locations and the status information.

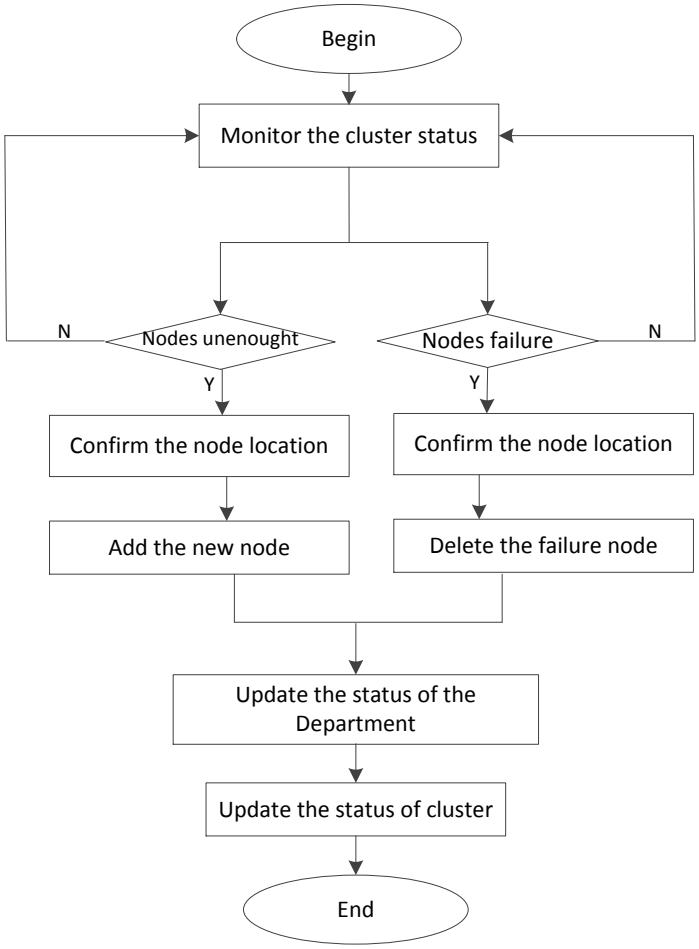


Figure 3. Monitor process of the cluster

However, MeteCloud platform also need to run in a reliable environment. Another primary node N_2 is necessary to merge and back up images and logs on N_1 in order to prevent the file from growing too large that it can lower the performance of the platform, and when failures happen, N_1 can recovery from N_2. Moreover, in the case of primary node failure situation, a backup node *BN* which synchronizes data from it in real time is designed and will be started after N_1 fails.

As shown in Figure 2, meteorological data nodes are designed to distribute in junior meteorological departments $JMD = \{JMD_1, JMD_2,\ldots, JMD_m\}$ ($\forall JMD_i \in JMD$), and each department has a number of nodes $JMD_i = \{JMD_{i1}, JMD_{i2},.., JMD_{in}\}$. These nodes communicate

with each other and connect to superior meteorological department by private network of meteorological industry.

Figure 3 shows the monitor process of adding and deleting nodes. The primary node has a monitor to view running status of these data nodes. When a subordinate meteorological department exists in the cluster is found paralyzed, cluster administrator can delete this node in the administration center. Then the administrator sends the maintenance information to the department according to the node location so that the failure node can be deal with in time. After the paralysis node is deleted, the data stored on the node will recover the data by its data copy or source data on the other data nodes of the cluster. In addition, when the cluster overall storage space is more than a pre-set value, the platform alarms and notifies the administrator to increase the data node. Both node delete operation and node add operation need to update the overall status and configuration files of the cluster after the operation is over.

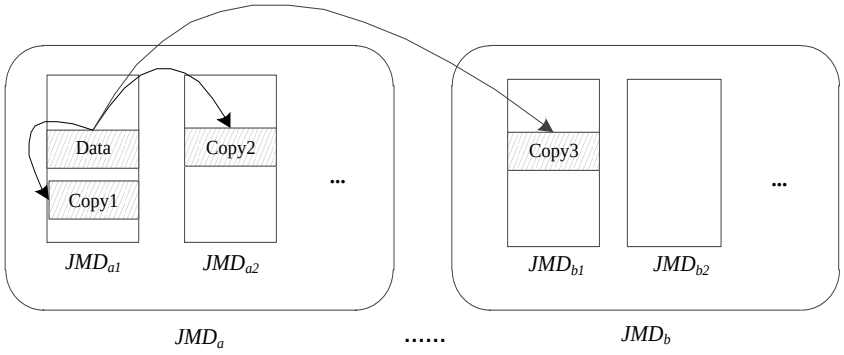


Figure 4. Backup Process of Meteorological Data Copies

The low tape backup capabilities and real-time ability to read and write meteorological data prompt the appearance of intelligent backup technology. The users can access the backup copies quickly using the technology when the data is lost caused by downtime or power outage. All nodes in the MeteCloud platform are loaded with HDFS. However, in order to store the meteorological data with high reliability, a new backup strategy referred to the original solution is proposed. When the meteorological data received from ground observation stations sends to the MeteCloud platform, the first copy will be stored in the same node with the source data, the second copy will be stored in a different node of the same junior meteorological department, and the third data copy will be stored in a different department. In Figure 4, *Data* is the original input

data, and $Copy_1$ is the first copy which will recover *Data* in real time when it is lost. $Copy_2$ is used to recover the data when JMD_{a1} fails and user can recover data from JMD_{b1} when the whole department *JMD* doesn't run.

IV. METEOROLOGICAL DATA STORAGE PROCESS ON METECLOUD PLATFORM

a. Meteorological data classification

Meteorological files are usually coded with a variety of classification methods due to the different observation methods and storage formats. In this paper, the meteorological data classification, based on the current storage of departments, includes Surface Meteorological Data (SMD), Upper-Air Observation (UAO), Agricultural Meteorological Data (AMD), Meteorological Disaster Data (MDD), Radar Data (RD) and Meteorological Satellite Data (MSD) as shown in Table 1. The above-mentioned types of meteorological files can be used in a variety of scientific researches, including thunderstorm forecasts, rainfall predictions as well as weather graphics drawings.

(1) SMD is one of the most important information in our Weather Watch collections, which is the base of the exploration of climate evolution and the predictions of climate trends. The Automatic Weather Station (AWS) data of A-format is a typical representative.

(2) The Observational data of wind profile is a form of UAO, which can be got from radiosonde, pilot balloon or weather satellites.

(3) AMD has a variety of data formats such as timing sequence diagram of microwave radiation and meter profile map of microwave radiation.

(4) Meteorological disasters are the direct or indirect damage caused by the atmosphere to human life and property, the national economic construction and national defense construction. Meteorological disaster data and radar raw data are both MSD-format data.

(5) Meteorological radar is specialized for atmospheric sounding, and its storage format includes radar images and radar products.

(6) MSD is received from meteorological satellite, which contains 2E and 2D primary data and MICAPS data.

Table 1: Meteorological data types and some examples

Data Types	Examples
Surface Meteorological Data	Automatic Weather Station (AWS) data of A-format
Upper-Air Observation	Observational data of wind profile
Agricultural Meteorological Data	Timing sequence diagram of microwave radiation; Meter profile map of microwave radiation
Meteorological Disaster Data	Meteorological disaster data; Radar raw data
Radar Data	Radar images, Radar Products
Meteorological Satellite Data	2E and 2D primary data and Micaps data

b. Storage architecture of MeteCloud platform

The meteorological data is transferred as tables using Hive in HDFS. The main components of Hive have been introduced in Section III (a). Figure 5 shows the execution architecture of MeteCloud platform. The users of meteorological departments submit data to the MeteCloud platform from web interfaces. They input historical data from the interfaces and store them in Hive database with HDFS. Driver can get a plan from the compiler with HQL string while users want to execute query statements. Driver also can send a plan consisting of a directed acyclic graph (DAG) of map-reduce jobs to the execution engine, such as ‘select * from table’. In addition, users can operate data or execute applications with MapReduce directly or indirectly. All documents and tables are stored in HDFS with the backup policy presented in Section III (b).

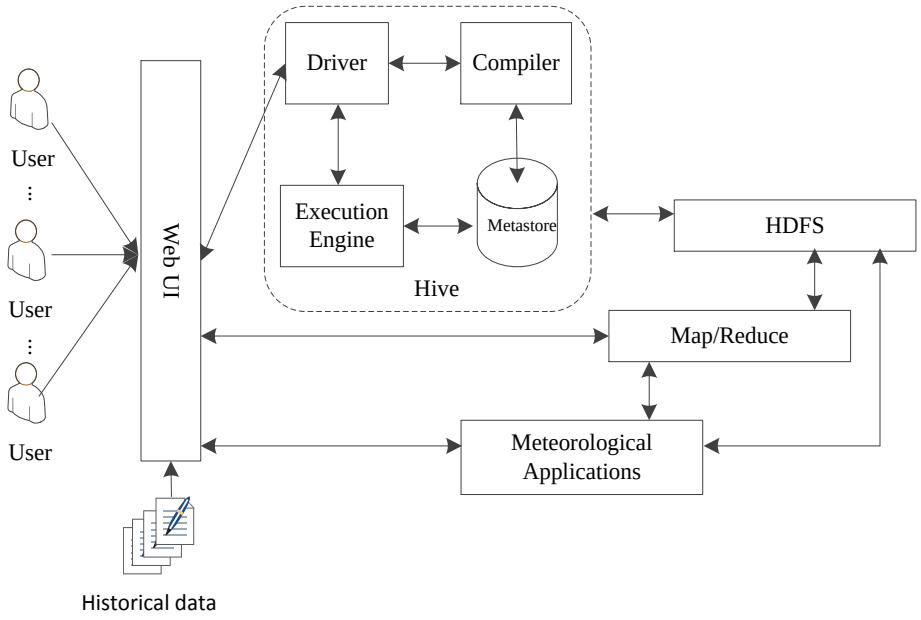


Figure 5. Execution architecture of MeteCloud platform

c. Meteorological data transfer process

The historical data researched in this paper is A-format collected from 756 Chinese Automatic Weather Stations, and we can storage it using the methods in Figure 6 and the main steps are as follows.

(1) Confirm the separator format of meteorological data

The A-format data contains multiple meteorological factors, such as station number, year, month, day, precipitation and air pressure. The separator of the data can be commas, tabs and spaces. Therefore, we need confirm the separator format at first.

(2) Confirm the number of fields

The fields we are going to create are equal to the number of factors. However, we need to create the tables according to the meteorological application requirements. The number of fields will affect the speed of the storage.

(3) Transfer storage operation using HQL

After the steps above, we store A-format file with HQL and the execution steps are as follows.

1) Connect to the Hive database. The users need to connect to the Hive database by JDBC programming interface.

2) Name the table we want to create. Before the table creation, we need to name it.

3) Create the table with HQL including the name and type of fields.

4) Confirm the specified separator. Use this separator in the process.

5) Confirm the location of input files. All input files have been stored in HDFS.

6) Load tables according to the path in step 5).

(4) Delete the table

There are many tables in the Hive database, and as times goes on, many tables may not need to use. As a result, we use the "drop" operation of HQL. When the tables are deleted, the copies are also deleted at the same time.

(5) END

When all operations are accomplished, end the whole process of the operation.

Main process of the creating tables and created tables

1. Create the connection object;
2. Connect to the cluster;
3. Choose the operations, if it is the creating process, go to line 5, and if the process is to delete the table, go to line 11;
4. Upload the historical data to the HDF;
5. Name the creating table;
6. Name the fields of the table;
7. Transfer the uploaded files and store them in the table with doubtless fields using "load" operation of HQL;
8. If the storage process if accomplished, it will go to the line.
9. Delete the table with "drop" operation of HQL.
10. If the process is accomplished, go to line 11;
11. END

Figure 6. Main process of the creating tables and created tables

(4) Query operation using HQL

In order to show the data in the tables conveniently, we experiment in the eclipse environment. Figure 7 shows the main process of query process, where the mainly process is to use the "query" operation of HQL so as to accomplish the query goals of users.

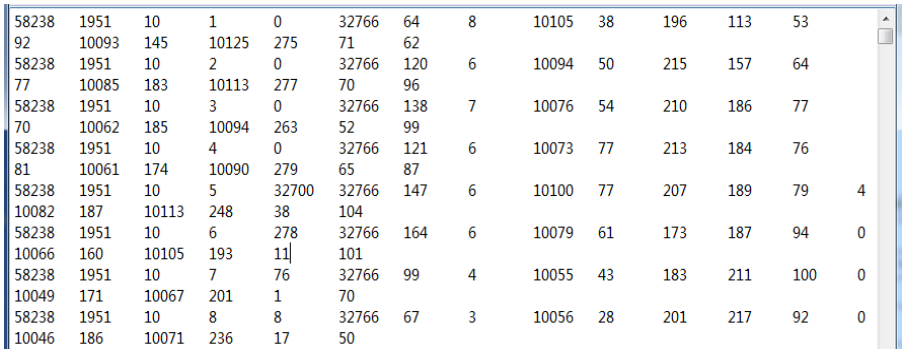
Main process of querying data of the created tables

1. Create the connection object;
2. Connect to the cluster;
3. Locate the position of the querying tables;
4. Confirm the fields you want to get;
5. Query the tables using "select" operation of HQL;
6. Name the fields of the table;
7. If the process is accomplished, go to line 8;
8. END;

Figure 7. Main process of querying data of the created tables

V. Experiments and analysis

a. Experimental datasets



58238	1951	10	1	0	32766	64	8	10105	38	196	113	53	
92	10093	145	10125	275	71	62							
58238	1951	10	2	0	32766	120	6	10094	50	215	157	64	
77	10085	183	10113	277	70	96							
58238	1951	10	3	0	32766	138	7	10076	54	210	186	77	
70	10062	185	10094	263	52	99							
58238	1951	10	4	0	32766	121	6	10073	77	213	184	76	
81	10061	174	10090	279	65	87							
58238	1951	10	5	32700	32766	147	6	10100	77	207	189	79	4
10082	187	10113	248	38	104								
58238	1951	10	6	278	32766	164	6	10079	61	173	187	94	0
10066	160	10105	193	11	101								
58238	1951	10	7	76	32766	99	4	10055	43	183	211	100	0
10049	171	10067	201	1	70								
58238	1951	10	8	8	32766	67	3	10056	28	201	217	92	0
10046	186	10071	236	17	50								

Figure 8. Experimental datasets

Figure 8 shows the experimental datasets. Each province in China produces about one million records. The data has a set of columns $Column = \{CL_1, CL_2, \ldots, CL_n\}$. Each column represents a meteorological factor, and the separators in the file are tabs. We store the files with different numbers of columns: $\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$. And the files with different numbers of records: $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ are also stored in the platform at first.

b. Analysis of experiment results

The number of nodes has a key impact on the time of transfer storage and query. We experiment with different number of nodes: {*1*, *2*, *3*, *4*, *5*}. Before the experiment, we choose the file of 100 records and 20 fields. Figure 9 shows the experimental results. We can confirm that the time of storage and query is being smaller when the number of nodes is increasing. And we also confirm that the query time is much lager than the transfer storage time.

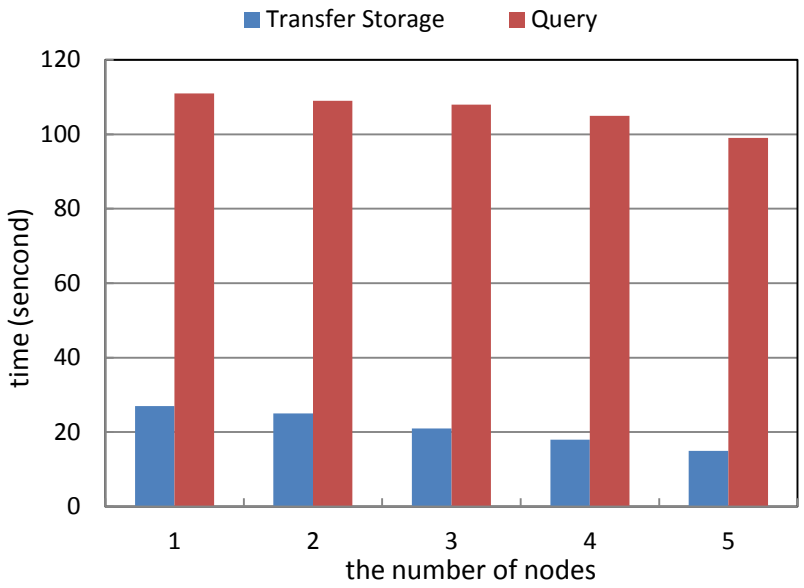


Figure 9. Transfer storage and query results with various numbers of nodes

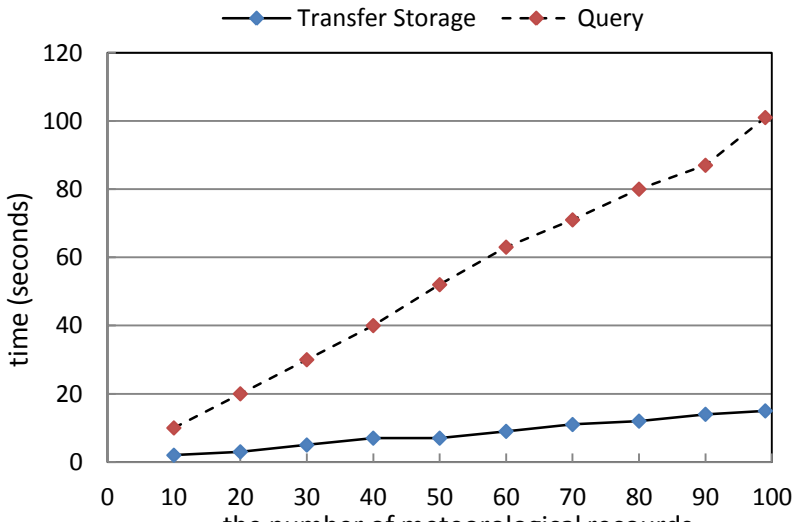


Figure 10. Transfer storage and query results with various numbers of meteorological records

We then experiment with different number of meteorological records. We fixed the number of nodes as 5 and the number of fields as 20. We choose the related files stored in HDFS, and the transfer storage and query results can be shown in Figure 10. We can see from it, the running time is growing when the records increase.

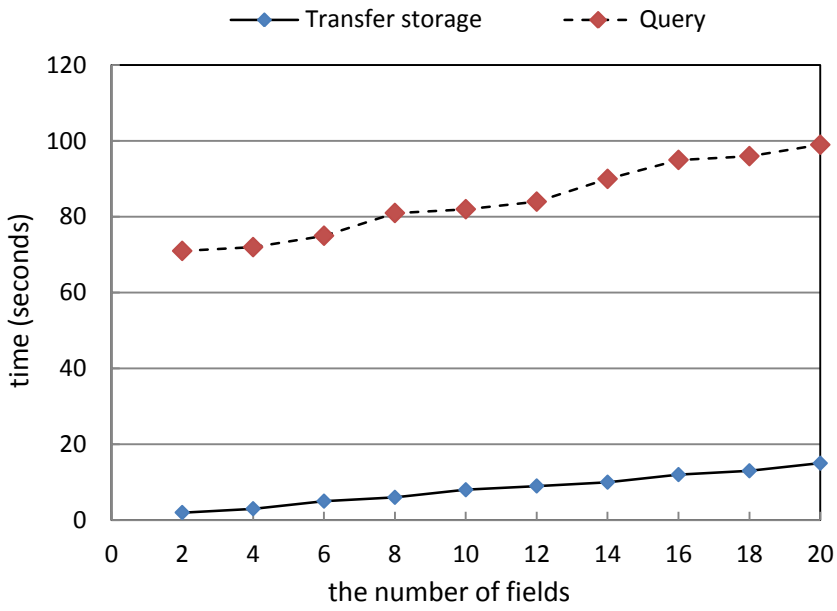


Figure 11. Transfer storage and query results with various numbers of fields

In addition, we experiment with different number of fields. We fixed the number of nodes as 5 and the number of records as 100,0000. We also choose the related files stored in HDFS, and the transfer storage and query results can be shown in Figure 11. We can see from it, the running time is growing when the number of fields increases. As a result, we need to choose a certain amount of fields so that we can store them with high efficiency and in this paper we set the value as 10 finally.

VI. CONCLUSIONS

We propose a meteorological cloud platform. The construction idea is also given in this paper. We model the MeteCloud platform considering the idle resources in the meteorological

departments through the private network. We then demonstrate the process of transfer storage and query. Moreover, we experiment in the platform using A-format files. As a result, the Mete-Cloud platform is effective to store and query the meteorological data.

In this paper, the platform is only used to store the data. In the future, we will execute meteorological applications in this platform, and the platform can also need to be optimized.

VII. ACKNOWLEDGMENT

This work is partly supported by National Natural Science Foundation of China (Grand Nos. 41275116) and Jiangsu Economic and Information Technology Commission project of China (Grand Nos.{2011}1178).

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffth, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: A berkeleyview of cloud computing. Tech. rep, 2009.

[2] S. Kamara and K. Lauter, "Cryptographic cloud storage", Proceedings of Financial Cryptography: Workshop on Real-Life Cryptographic Protocols and Standardization 2010, January 2010.

[3] X. Shengjun, Z. Jie, X. Xiaolong, "An improved algorithm based on ACO for cloud service PDTs scheduling", Advances in Information Sciences and Service Sciences, Vol. 4, no.18, pp. 340-348, 2012.

[4] R. Zhang, C. Zhao, "Automated Deployment Mechanism for Computational Chemistry Services in Cloud", Advances in Information Sciences and Service Sciences, Vol. 4, no. 3, pp. 83-90, 2012.

[5] R. Grossman, "The case for cloud computing", IT Professional, Vol.11, no. 2, pp. 23–27, 2009.

[6] X. Zhao, Y. Yang, L. Sun, H. Huang, Metadata-Aware Small Files Storage Architecture on Hadoop, Web Information Systems and Mining, Lecture Notes in Computer Science Vol. 7529, 2012, pp. 136-143

[7] G. Makrai and Z. Prekopcsák, Scaling out data preprocessing with Hive. In Proceedings of the 15th International Student Conference on Electrical Engineering, 2011.

[8] H. Jeong, J. Park, "An Efficient Cloud Storage Model for Cloud Computing Environment", In Advances in Grid and Pervasive Computing, vol. 7296 pp. 370-376, 2012.

[9] A. Joint, E. Baker, E. Eccles, "Hey, you, get off of that cloud?", Computer Law & Security Review, Vol. 25, no. 3, pp. 270–274, 2009.

[10] "About AWS and solutions", what is AWS? (http://aws.amazon.com/what-is-aws/), 2012.

[11] "OpenStack®: the open alternative to cloud lock-in Invented by Rackspace and NASA", OpenStack | Open cloud Operating System supported by Rackspace (http://www. rackspace.com/ cloud/ openstack/), 2012.

[12] "IBM SmartCloud", IBM Cloud Computing Overview (http://www.ibm.com/cloud-computing /us/en/?lnk=msoST-ccom-usen), 2012.

[13] "Platform Services", Oracle | Cloud Computing (http://www.oracle.com/us/solutions/cloud/ over-view/ index.html),2012.

[14] "Welcome to Apache Hadoop", Hadoop (http://hadoop.apache.org/), 2012.

[15] R. P. Padhy, "Big Data Processing with Hadoop-MapReduce in Cloud Systems", International Journal of Cloud Computing and Services Science, Vol. 2, no. 1, pp. 16-27, 2013.

[16] J. S. Hare, S. Samangooei and P. H. Lewis, "Practical scalable image analysis and indexing using Hadoop", Multimedia Tools and Applications, 2012.

[17] Y. Pingle, V. Kohli, S. Kamat and N. Poladia, "Big Data Processing using Apache Hadoop in Cloud System", National Conference on Emerging Trends in Engineering & Technology, 2012.

[18] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff and R. Murthy, "Hive: a warehousing solution over a map-reduce framework", The Proceedings of the VLDB Endowment, Vol. 2, no. 2, pp. 1626-1629, 2009.

[19] N. Pansare and Z. Cai, "Using Hive to perform medium-scale data analysis" (http://www.cs.rice.edu /~np6/Papers/), 2010.

[20] A. Gruenheid, E. Omiecinski and L. Mark, Query optimization using column statistics in hive, IDEAS' 11 Proceedings of the 15th Symposium on International Database Engineering & Applications, USA 2011, pp. 97-105, 2011.