*Journal of Marine Science and Technology, Vol. 25, No. 6, pp. 791-801 (2017)*
DOI: 10.6119/JMST-017-1226-21

# RESEARCH ON INTEGRATION AND SHARING SOLUTIONS FOR BIG STRUCTURED DATA FOR THE MARINE ENVIRONMENT

Wei Wang[1], Wenfang Cheng[2], Jing Chen[1], Ya-Nan Yu[1], and Jianxun Zhang[1]

Key words: marine big data, integration, sharing.

## ABSTRACT

The data-intensive science, a new paradigm in scientific discovery, demands an urgent solution to integrate and share big marine environment data. Through the modeling of marine environmental features (such as Temperature, Salinity, Density et al.), a novel solution is presented in this paper. This solution employs Hadoop and Hbase, based on the Bigtable algorithm first proposed by Google Inc. By using hierarchical technology, scalability of the data integration framework is realized. Additionally, an efficient execution of various queries is realized by using the customized auxiliary index algorithm. In the experimental cluster environment constructed from four ordinary personal computers, experiments involving a huge amount of structured marine environmental data importing, cost-effective storage, and query processing were carried out successfully. The experimental results indicate that this solution to storing and sharing big data for the marine environment has the property of infinite lateral expansion, and is able to execute efficient and complicated query operations.

## I. INTRODUCTION

Recent changes in science have led to a new paradigm in scientific discovery. This paradigm, which distinguishes data-intensive science from computational science, is called the fourth paradigm and has appeared in recent years (Bell et al., 2009; Chen et al., 2014; Merino et al., 2016; Wang et al., 2016). The aim of this paradigm is a world where all scientific literature and online scientific data can interact. Many scientific fields have already become highly data-driven with the development of computer sciences (Bryant et al., 2011; Szalay et al., 2011; Knoppers et al., 2017; Voyles et al., 2017). In the field of marine science, since the beginning of attempts to describe the complexity of the natural marine environment, long-term samplings of the oceans have been carried out by humans. As a result, marine environmental data has been accumulated rapidly. This is especially true in recent years, with the sizes of datasets expanding from gigabytes to petabytes. However, it is very difficult to utilize this precious data because marine data has a large number of features. This data can be summarized as multi-source, diverse, and multi-state, and can be defined as "marine big data". The lack of proper data management and analysis methods for marine big data means this data is not used to its full extent.

In addition, there is an increasing need for fast, reliable, accurate and up-to-date information from all marine data producers and users in China. Automated analysis is highly required for the enormous amount of data generated by oceanography. Moreover, a centralized repository is necessary as it is impractical to replicate copies of huge datasets for individual remote research groups. While many marine data databases were established on the basis of demand-driven, application-oriented, problem-solving, the data was not well-classified and end-users were not clearly identified. It is evident that there was a clear gap between the information produced and the information absorbed by end-users (Xu et al., 2014). Therefore, a solution that combines centrally managed data and distributed storage will undoubtedly drive the development of field of marine science in the future.

To adapt to these changing challenges and to promote the development of data-intensive marine science, methods to manage and share big marine data must be explored in depth using the latest computer technology.

The remainder of the paper is organized as follows. Section 2 summarizes related work in the field of data integration. Considering the characteristics of physical oceanographic data, a key-value based data model is proposed in Section 3. Section 4 introduces the implementation of the data model using Hadoop and Hbase. Section 5 gives the result of performance tests. The conclusions of this paper are presented in Section 6.

## II. RELATED WORK

Recently, a large body of work has addressed issues in data in-

*Paper submitted 04/25/17; revised 12/07/17; accepted 12/13/17. Author for correspondence: Wei Wang (e-mail: wangw@irsa.ac.cn).*

[1] *College of Information Engineering, Tianjin University of Technology and Education, Tianjin, China.*
[2] *Polar Research Institute of China, State Oceanic Administration, Shanghai, China.*

tegration, sharing and the manipulation of large datasets. We will summarize previous work in the following section.

### 1. Data Warehouse in China Digital Ocean Prototype System Framework

Digital Ocean is a new research domain of Digital Earth. The construction of the China Digital Ocean Prototype System (CDOPS) pushes Digital Ocean a step forward from a mere con-cept to a realistic system (Zhang et al., 2012). To realize all kinds of Marine data integration and sharing, data warehouse technology has been used in the CDOPS. Data from 22 different distributed nodes has been transferred through a data service bus which was developed from web service technology. The data stored in the data warehouse include geo-reference data, remote sensing data, ocean observation data, and ocean business data. While the data warehouse solves the problems of data integration to a certain extent, including multi-source and heterogeneous data, disadvantages appear with the rapid increase of the volume of marine-environment data. Firstly, the existing storage capacity of the CDOPS has difficulty supporting the rapid growth of marine data and the storage arrays used in this system are very expensive. Secondly, The SQL JOINs which were fast to run in the past are now slowing down and are simply not performing well enough at scale. Last but not least, once problems occur in the storage server system, the entire data warehouse system is paralyzed. Although the data warehouse in the CDOPS solves the problems of marine data integration and application to some extent, the disadvantages become obvious when data rapidly accumulates.

### 2. Marine SDI

The concept of Spatial Data Infrastructure (SDI) has emerged in the last two decades with the goal of realizing sustainable land management at different scales for specific user groups. SDI is an initiative that allows access and sharing of spatial data for decision-making, management and administration. In response to this development, many countries worldwide are developing SDI as a way to better manage and share their spatial data assets. It is becoming evident that efficient access to appropriately structured and spatially referenced data from many different sources is necessary to realize the full potential of information technology applications (Root et al., 1997). The SDI promotes geospatial data sharing at all levels (local, regional and national) of governmental, academic, private, and non-profit sectors. It provides a structured system of practices and relationships between data producers and users, which underpins the design, implementation and maintenance of mechanisms which facilitate standardization, data sharing and accessing at affordable costs. However, SDI is still in its infancy in China. Most of the data environment is still application-oriented. There are no consolidated and efficient systems to maintain, manage and utilize geospatial databases. The available legacy databases suffer from inconsistency of data storing, duplications and digital transformation (Xu et al., 2014).

Research into the application of SDI in the marine domain is less advanced because of the complexity of marine environmental data. Much research proposes only concepts or theoretical models for ocean data integration and sharing. Although (Strain et al., 2006) proposed a seamless SDI which can include data from the land, coastal and marine environments and which will improve access and sharing of data between these environments, there is not yet any such running system based on SDI because of the intrinsic characteristics of ocean data.

### 3. Cloud Computing

Cloud Computing Technology has recently been applied to the field of Marine data integration and sharing because the service provide by cloud computing has the properties of efficiency, security, reliability, low cost and flexible organization and service. A cloud computing architecture for marine environmental data was proposed by Shi based on the processing requirements and features of marine environmental data (Shi et al., 2013). The architecture proposed in this paper is composed of three layers to realize the integration and sharing of marine environmental information. Taking advantage of Hadoop, an open source cloud computing platform, the prototype demonstration system with this architecture can achieve high performance and scalability when dealing with large-scale marine data.

The proposed architecture solves the problems of accessing a huge amount of marine environment data to some extent by using Hadoop, which was not the case with the CDOPS. However, it cannot model marine features in the real world which are described by data in their own means and provide a solution to integrate data coming from the land, coastal, and marine environments seamlessly, which was proposed in SDI. Moreover, the architecture does not offer a practical solution to the problems arising in the process of data integration, which can be summarized as data inconsistency, data redundancy and data integrity. These problems limit the further research and application of this architecture.

## III. MODELING MARINE ENVIRONMENTAL FEATURES

Marine environmental data is a digital description of the state of marine environmental features in the real world at a given moment. To achieve the effective integration and application of marine environmental data, various features must first be modeled in a multi-dimensional spatial-temporal environment.

### 1. Data Saving Model

To implement integration and sharing of marine environmental data, the first important step is to determine the items contained in the dataset. Marine environmental data relates to many subjects such as marine hydrology, marine meteorology, marine chemistry, marine geology, and marine physics. Table 1 and Table 2 list items in the datasets for marine hydrology and marine meteorology.

Through the description of data items (also fields) contained in the marine environmental dataset, it can be concluded that

**Table 1. Marine hydrology.**

| Field | Details | Data Type |
|---|---|---|
| Position | Geographic coordinates composed of Latitude and longitude | Double |
| Time | Date and time of data gathering which is accurate to minutes | Date time |
| Water Depth | Sampling or measuring depth | Float |
| Water Temperature | Temperature value of current spatial point | Float |
| Salinity | Salinity value of current spatial point | Float |
| Water Color | Description of water color | String |
| Transparency | Description of water transparency | String |
| … | | |

**Table 2. Marine meteorology.**

| Field | Details | Data Type |
|---|---|---|
| Position | Geographic coordinates composed of Latitude and longitude | Double |
| Time | Date and time of data gathering which is accurate to minutes | Date time |
| Water Depth | Sampling or measuring depth | Float |
| Cloud | Description of cloud at current position | String |
| Visibility | Description of visibility at current position | String |
| Wind Direction | The current value of wind direction at current position | Integer |
| Wind Speed | The current value of wind speed in current position | Float |
| Air Temperature | The current value of air temperature in current position | Float |
| … | | |

the value of each item is closely related to the spatial position, water depth, and time of data-gathering. If we define A as a collection of keys (one key can be composed of coordinates, date-time, depth and qualifier of a field), and B as a collection of all real values in a marine environment dataset, there must be some existing relationship between A and B, defined by *f*, which makes any element in B matches an element in A one by one. The relationship can be denoted by the following formula:

$$f : \mathrm{A} \rightarrow \mathrm{B} \tag{1}$$

That is to say, marine environmental data from any source, saved in any format, can be simplified by storing it as the combination of a key and a value. Multi-source, diverse, multi-state and huge amounts data can be transformed to this form and integrated together seamlessly.

Data formatted in the form of key-value can be saved into Bigtable, which was first presented by Google, Inc. A Bigtable cluster is a set of processes that run the Bigtable software. Each cluster serves a set of tables. A table in Bigtable is a sparse, distributed, persistent multidimensional sorted map (Chang et al., 2008). Data is organized into three dimensions: rows, columns, and timestamps.

(Row: string, column: string, time: int64) →string

Data can be obtained from a particular combination of row key, column key, and timestamp. Rows are grouped together to form the unit of load balancing, and columns are grouped together to form the unit of access control and resource allocation.

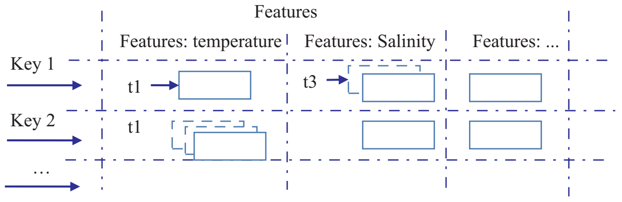


**Fig. 1. A slice of the marine environmental data table.**

## 2. Row Key Design

Getting large volumes of marine environmental data into an NoSQL database, which implements the data model of Bigtable, is only half the problem; once there, it needs to be retrieved as efficiently as possible for it to be useful. To realize fast data retrieval, a reasonable design of row key must be carried out before data is saved into the database. Information contained in an arbitrary water body (depicted in Fig. 2) can be determined by three tuples, namely, by depth, coordinates, and time. Thus, the row key must contain depth, coordinates, and time. Because Bigtable maintains data in lexicographic order by row key, to ensure that data relating to a particular water body remains together, we arrange these three parts into the sequence of depth, coordinates, and time. The structure of the row key can be depicted by Fig. 3.

## 3. Depth

Because the deepest part of the ocean has a depth of about 12000 m, the length of the Depth section is set to 5 bits. To arrange data in a sequence from shallow to deep, zero must be filled

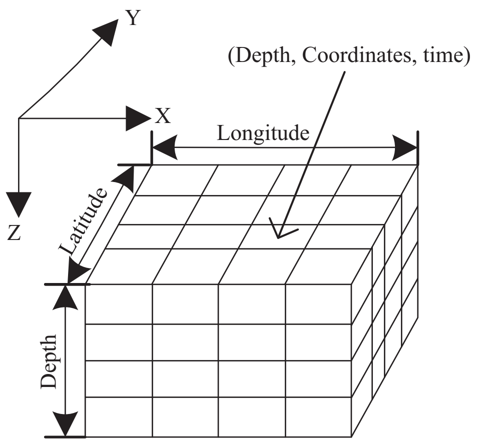


**Fig. 2. An arbitrary body of water descripted by depth, coordinates, and time.**

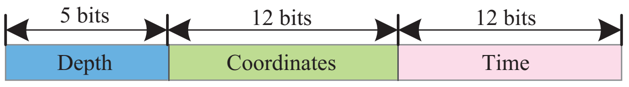


**Fig. 3. Structure of the row key.**

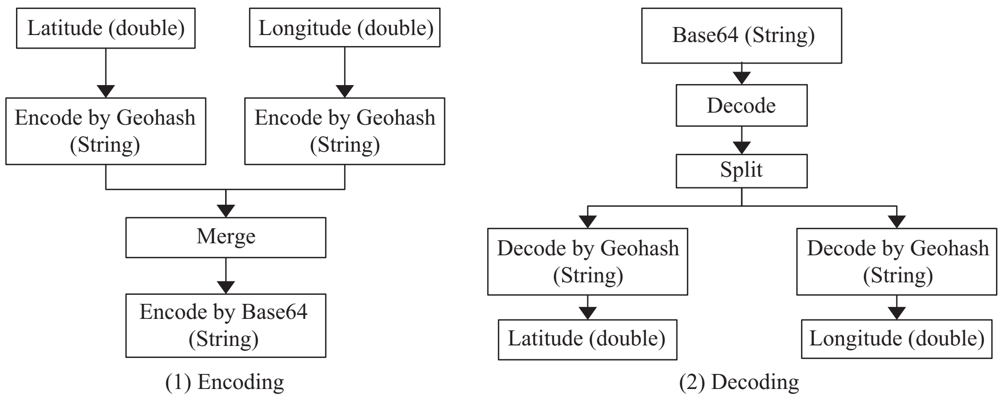

Latitude (double)
Longitude (double)
Encode by Geohash (String)
Encode by Geohash (String)
Merge
Encode by Base64 (String)
(1) Encoding
Base64 (String)
Decode
Split
Decode by Geohash (String)
Decode by Geohash (String)
Latitude (double)
Longitude (double)
(2) Decoding

**Fig. 4. Encoding and decoding of geographic coordinates using Geohash.**

if the value of depth is not long enough for 5 bits. Placing the Depth section at the start of the row key means all marine environmental data will be divided according to depth. Different parts of the dataset will be saved into different regions of the server and access hot points can be avoided effectively.

### 4. Coordinates

In order for the two-dimensional geographic coordinates to be arranged in linear manner, the algorithm of Geohash is used in this paper. Geohash is an algorithm for latitude/longitude geocodes, which was invented by Gustavo Niemeyer (Balkić et al., 2012). It is a hierarchical spatial data structure, which subdivides space into buckets with a grid shape. Using Geohash, the values of latitude and longitude are converted into two strings composed of the characters 0 and 1. The two strings are then merged into one string by the means of insertion, such that the even bits represent the longitude code and the odd bits represent the latitude code. Finally, this string is encoded using Base64. The values of latitude and longitude can be obtained easily by following these steps in reverse. The procedure of encoding and decoding is depicted in Fig. 4.

Because the precision of geographic coordinates used in marine environmental data is 0.0001, a length of 12 bits is sufficient for the Base64 string.

### 5. Time

Time continuity is one of the most important factors for marine environment data. In the row key, we use 12 bits to identify the time at which field value was gathered. The Time section contains five parts which represent the year (4 bits), month (2 bits), day (2 bits), hour (2 bits), and minute (2 bits).

## IV. ARCHITECTURE

The architecture of the integration and sharing solution for structured Marine environmental data consists of three distinct

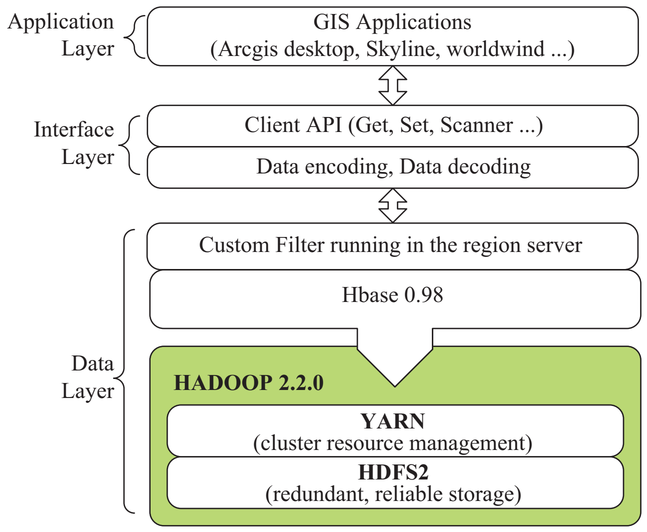

Application Layer
GIS Applications (Arcgis desktop, Skyline, worldwind ...)
Interface Layer
Client API (Get, Set, Scanner ...)
Data encoding, Data decoding
Data Layer
Custom Filter running in the region server
Hbase 0.98
HADOOP 2.2.0
YARN (cluster resource management)
HDFS2 (redundant, reliable storage)

**Fig. 5. Logical architecture of data integration and sharing solution.**

layers, as illustrated in Fig. 5. From the bottom up, these consist of the data, interface, and application layers, respectively. Hadoop (version 2.2.0) and Hbase (version 0.98), open-source projects developed by Apache, were adopted in the data layer to realize big marine environmental data saving. With the help of client API provided by Hbase, data can be sent or received using Hbase in the Interface layer. Data is finally obtained by the applications running in the client.

### 1. Data Layer

Hadoop is the infrastructure in the data layer, and is an Apache Software Foundation project which became top-level in January 2008. There have been many contributors, both academic and commercial (Yahoo being the largest such contributor), and Hadoop has a broad and rapidly growing user community (http://wiki.apache.org/hadoop/PoweredBy). As one of the main components of Hadoop, The Hadoop Distributed File System (HDFS) is designed to store very large datasets reliably, and to stream those datasets at high bandwidth to user applications. With the help of YARN, a framework for job scheduling and cluster resource management, thousands of servers both host directly attached storage and execute user application tasks to make up a large cluster. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at any size (Taylor, 2010) (Tabaa et al., 2012).

Hbase, an Apache Software Foundation project (http://hadoop.apache.org/Hbase/), is another vital component in the data layer of this solution. Having looked at the Bigtable model above, we could simply state that Hbase is to some extent a faithful, open-source implementation of Google's Bigtable. Hbase adds a distributed, fault-tolerant scalable database, built on top of the HDFS file system, with random real-time read/write access to data. Each Hbase table is stored as a multidimensional sparse map with rows and columns, with each cell having a time stamp. A cell value at a given row and column is by uniquely identified by (Table, Row, Column-Family: Column, Timestamp) $\rightarrow$ Value.

To extract data from Hbase as quickly as possible, a custom filter program running on the regional server has been developed based on the structure of the lexicographic order. With the filter string transferred from client, the location of the data required by the client can be found accurately without scanning the whole data array.

### 2. Interface Layer

The main component in the Interface layer is a Java client API owned by Hbase. The API contains methods such as Get, Set, Scanner and so on. With the help of these methods, data can be saved or retrieved to or from Hbase. In our solution, we use Set to save data to the Hbase and Scanner to get data from Hbase. Through the Interface layer, centralized data management can be realized.

### 3. Application Layer

After data is retrieved from the Interface layer, it is sent to the application layer. Applications in this layer read the data and visualize it in real time.

## V. IMPLEMENTATION

Based on the features of the marine environmental data model and the architecture presented above, the implementation of this solution consists of two main aspects: data input and retrieval.

### 1. Data Input

Saving data into Hbase is very simple with the help of Hbase's client API. In this solution we used data of the marine water temperature and salinity from Argo. The data ran from 2010 to 2013 and originated from all over the globe. A large amount of data is saved in the format of single textual files. Information on the temperature and salinity hosted in these files was extracted and inputted to Hbase in lexicographic order after being encoded. The program code for data input is shown in Fig. 6.

Through these operations, a large amount of multi-source heterogeneous data was integrated together. Inconsistency of source data is effectively avoided during the process of data input.

### 2. Data Retrieval

The fastest manner of data retrieval in Hbase is using Scan with the start and end range. Therefore, the key problem of data retrieval in Hbase is to know the exact ranges of data that you want to capture before the retrieval operation executes. Analyses of the data distribution range for query scenarios under different condition are presented in the following sections.

### 3. Only Depth Known

If user wants to retrieve all data pertaining to a special depth

```
public void SaveDataToHbase(byte[] DataSource){
onfiguration conf = HBaseConfiguration.create();
HTable table = new HTable(conf, "Marine_Environment");
table.setAutoFlush(false,false);
int count=0;
for(byte row: DataSource){
tring[] value = Byte.toString(row).split(‘,’);
Put put = new Put(value[0]); //rowkey
put.add(Bytes.toBytes("Features"), Bytes.toBytes("Temperature"), value[1]); // Temperature
put.add(Bytes.toBytes("Features"), Bytes.toBytes("Salinity"), value[2]); // Salinity
table.put(put);
count++;
if(count % 1000 == 0) {
            table.flushCommits();
}
}
table.flushCommits();
}
```

**Fig. 6. Saving Data to Hbase.**

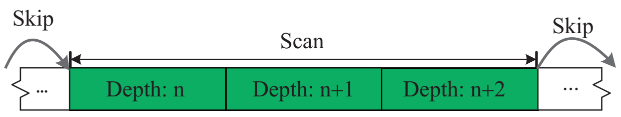


**Fig. 7. Data distribution range in the only depth known scenario.**

```
public void RetrieveDataFromHbase(){
…
Scan scan = new Scan();
 scan.setStartRow(Bytes.toBytes(“00002”));
 scan.setStopRow(Bytes.toBytes(“00003”));

ResultScannerresultScanner = table.getScanner(scan);
For(Result result: resultScanner)
{
System.out.println(result.getValue);
}
resultScanner.close();
…
}
```

**Fig. 8. Data retrieval with only the depth known.**

over the whole globe at any time, the parameter of water depth range only can be specified. Because the depth parameter is located in the first section in the key row, data with depth *n* is arranged together, which is depicted by Fig. 7.

The query implementation in this scenario is very simple. Through setting the starting and end range parameters, data in this special range can be retrieved easily. The Java code used to retrieve this range of data is shown in Fig. 8.

## 4. Only Spatial Area Known

In many cases, users, such as researchers, hope to retrieve all data for a particular area, which can be denoted by left-bottom and right-up points in their research activities. Based on the structural design presented earlier in this paper, the encoded coordinate string located in the second part of the row key must contain every depth value for a particular area. Each depth value contains the values for every set of coordinates in the entire globe. The alignment of the row-key necessary to extract the desired information in this scenario is shown in Fig. 9.

To retrieve data at close to real-time speed for a given spatial area, the starting and stopping row-keys must be known accurately prior to the query operation. The key to solving this prob-

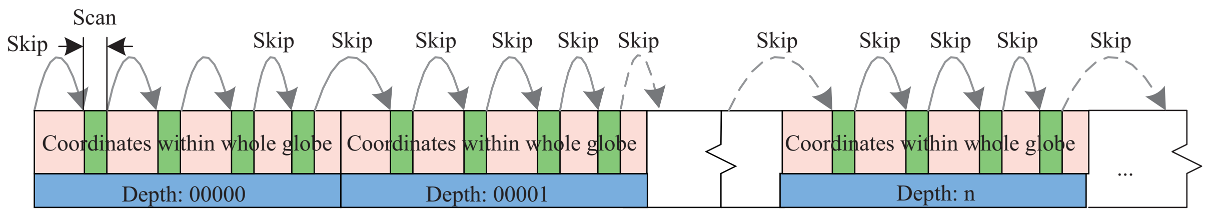
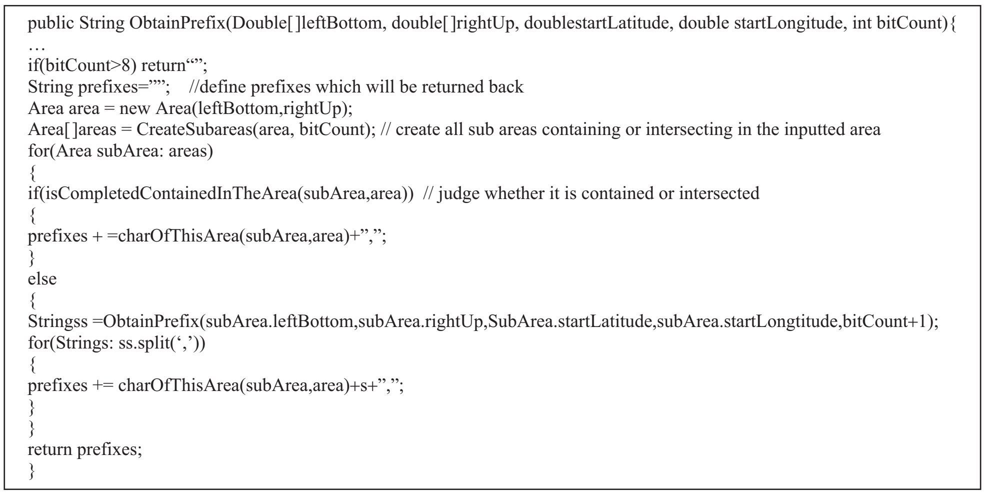
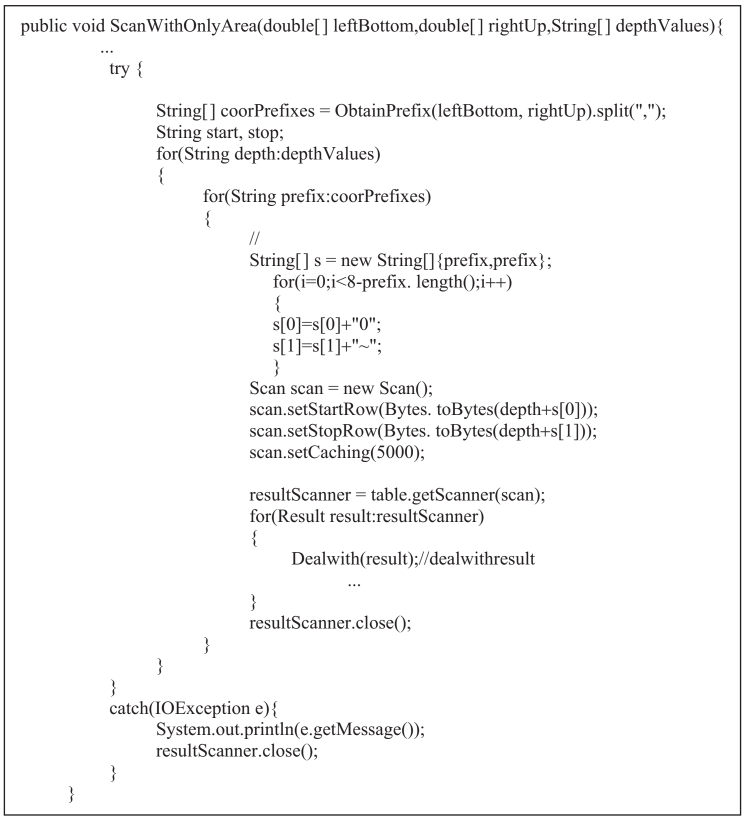

Scan
Skip
Skip
Skip
Skip
Skip
Skip
Skip
Skip
Skip
Skip
Skip
Skip
Coordinates within whole globe
Coordinates within whole globe
Coordinates within whole globe
...
Depth: 00000
Depth: 00001
Depth: n

**Fig. 9. Data distribution range in the only spatial area known scenario.**



```
public String ObtainPrefix(Double[]leftBottom, double[]rightUp, doublestartLatitude, double startLongitude, int bitCount){
…
if(bitCount>8) return“”;
String prefixes=””;    //define prefixes which will be returned back
Area area = new Area(leftBottom,rightUp);
Area[]areas = CreateSubareas(area, bitCount); // create all sub areas containing or intersecting in the inputted area
for(Area subArea: areas)
{
if(isCompletedContainedInTheArea(subArea,area))  // judge whether it is contained or intersected
{
prefixes + =charOfThisArea(subArea,area)+”,”;
}
else
{
Stringss =ObtainPrefix(subArea.leftBottom,subArea.rightUp,SubArea.startLatitude,subArea.startLongtitude,bitCount+1);
for(Strings: ss.split(‘,’))
{
prefixes += charOfThisArea(subArea,area)+s+”,”;
}
}
return prefixes;
}
```

**Fig. 10. Program of obtaining prefixes among encoded coordinates.**



```
public void ScanWithOnlyArea(double[] leftBottom,double[] rightUp,String[] depthValues){
        ...
        try {

                String[] coorPrefixes = ObtainPrefix(leftBottom, rightUp).split(",");
                String start, stop;
                for(String depth:depthValues)
                {
                        for(String prefix:coorPrefixes)
                        {
                                //
                                String[] s = new String[]{prefix,prefix};
                                    for(i=0;i<8-prefix. length();i++)
                                    {
                                    s[0]=s[0]+"0";
                                    s[1]=s[1]+"~";
                                    }
                                Scan scan = new Scan();
                                scan.setStartRow(Bytes. toBytes(depth+s[0]));
                                scan.setStopRow(Bytes. toBytes(depth+s[1]));
                                scan.setCaching(5000);

                                resultScanner = table.getScanner(scan);
                                for(Result result:resultScanner)
                                {
                                        Dealwith(result);//dealwithresult
                                                ...
                                }
                                resultScanner.close();
                        }
                }
        }
        catch(IOException e){
                System.out.println(e.getMessage());
                resultScanner.close();
        }
}
```

**Fig. 11. Code for data retrieval with only the spatial area known.**

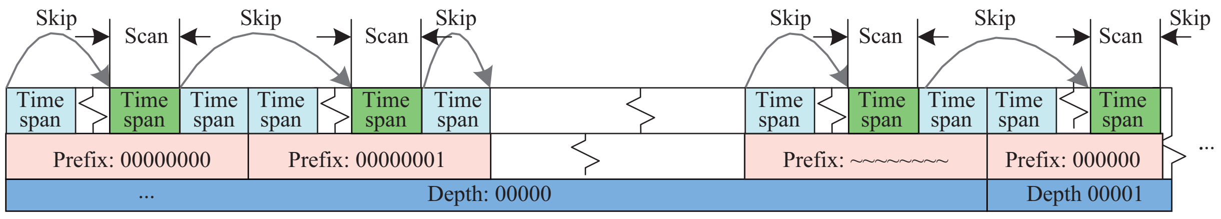


**Fig. 12. Data distribution range in the only time-span known scenario.**

```
public void ScanWithOnlyTimeSpan(String startTime, String endTime, String[ ] depthValues){
        ...
        try {
            HTable table = new HTable(conf, "Marine_Environment");
            HTable tableIndex = new HTable(conf, "Marine_Environment_SecondaryIndex");
            Scan scan = new Scan();
            for(Result result:tableIndex.getScanner(scan)){
                String coordinate = Bytes.toString(result.getRow());
                for (Stringdepth:depthValues)
                {
                    String startKey = depth + coordinate + startTime;
                    String endKey = depth + coordinate + endTime;
                    Scan scanData = new Scan();
                    scanData.setStartRow(Bytes.toBytes(startKey));
                    scanData.setStopRow(Bytes.toBytes(endKey));
                    for (Result rlt:table.getScanner(scanData)){
                       DealWith(rlt); //deal with query result
                    }
                }
            }
            resultScanner.close();
            resultScannerIndex.close();
        }
        catch (IOException ∈){
            System.out.println(e.getMessage());
            resultScanner.close();
        }
    }
```

**Fig. 13. Code of data retrieving with only time-span known.**

lem is to compute the prefixes of coordinate strings that contain 8 characters. According to the Geohash algorithm and Base 64 coding, prefixes in the given spatial area can be obtained through the algorithm represented by the code snippet depicted in Fig. 10.

Prefixes that correspond to each depth value are not unique. On the contrary, there usually may be many prefixes in a given spatial area. Through the operation of joining each depth-value string and prefix together, the starting or stopping row-key can be known. The procedure of row-key making and data retrieval is represented by the code snippet depicted in Fig. 11.

## 5. Only Time-Span known

The requirement of retrieving global marine environment data for a certain time period from the central database is urgent in marine science research. Because the time value is located in the third section of the row-key, every possible depth value and coordinate prefix must be known before constructing starting or stopping row-key in order to compute the data distribution in this scenario. Data distribution ranges in only time-span known scenario are depicted in Fig. 12. However, the efficiency of the scan operation based on only setting the starting and stopping row-keys will be not high because the number of possible coordinate prefixes is very large and the number of possible combinations of depth value and coordinate prefixes is huge.

To avoid scanning the whole lexicographical order from start to end, other methods must be introduced to realize data retrieval at close to real-time speed. A simple option is to introduce a filter provided by the Hbase client API to solve this problem. However, this approach cannot prevent scanning of the whole lexicographical order and the performance is therefore very poor. As such, it is necessary to find another method to improve query performance.

In this solution, we used a secondary index technology to solve the problem. Distinct and encoded coordinate values were simultaneously saved into another table in Hbase when data was saved into the main data table. All coordinate values should be brought out by a full scan of the index table before the query operation is executed in the scenario that only the time-span is known. Starting or stopping row-key prefixes are formed through composition of each depth value, each coordinate, and the starting or end time value together.

Realization of this approach in Java is depicted by Fig. 13.

**Table 3. Settings of software and hardware environment.**

| NO. | Nodes of cluster | | Hardware environment | | | Software environment | | |
|---|---|---|---|---|---|---|---|---|
| | Role | Process running | CPU | RAM | Hard Disk | OS | Software | JDK |
| 01 | Master Node | NodeManager, ResourceManager, WebAppProxyServer | 2.4HZ | 2G | 500G/7200r/s | CentOS 6.5 | Hadoop 2.2.0, Hbase 0.2 | Java development kit Version 1.7 |
| 02 | Data Node | DataNode, NodeManager | 2.4HZ | 2G | 500G/7200r/s | | | |
| 03 | Data Node | DataNode, NodeManager | 2.4HZ | 2G | 500G/7200r/s | | | |
| 04 | Data Node | DataNode, NodeManager | 2.4HZ | 2G | 500G/7200r/s | | | |

**Table 4. Only depth known scenario.**

| Condition (Depth) | Results (rows) | Retrieval time at different size (Millisecond) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0.5G | 1.5G | 2.5G | 3.5G | 4.5G | 6G |
| 00000 | 83803 | 2501.38346 | 2250.62 | 2342.26 | 2224.95967 | 2291.3395 | 2212.115459 |
| 00001 | 125807 | 3887.33597 | 3294.67 | 3325.96 | 3252.8644 | 3314.2976 | 3228.181752 |
| 00002 | 183469 | 3985.30368 | 10235.8 | 16717.7 | 22304.5275 | 29349.519 | 38696.98035 |
| 00003 | 39919 | 781.89113 | 757.637 | 741.974 | 751.158132 | 784.86321 | 787.107105 |
| 00004 | 41747 | 721.931034 | 709.6 | 753.208 | 729.53404 | 712.75555 | 726.244634 |
| 00005 | 39770 | 1891.94916 | 1799.44 | 1805.08 | 1795.98648 | 1843.9445 | 1805.700454 |
| 00006 | 100532 | 1586.2548 | 1509.58 | 1503.26 | 1490.88982 | 1502.0994 | 1531.917398 |
| 00007 | 83444 | 2130.99634 | 2050.68 | 2047.63 | 2027.65037 | 2058.2228 | 2042.29831 |
| 00008 | 113563 | 4863.555 | 4124.08 | 3729.86 | 3863.312 | 3700.066 | 3451.255 |

**Table 5. Depth and time known scenario.**

| Condition (Depth + Year + Month) | Results (rows) | Retrieval time at different size (Millisecond) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 3.2G | 4.2G | 5.2G | 6.2G | 7.2G | 8.2G |
| 000000201301 | 8472 | 931.31 | 903.452 | 918.6194 | 179.3 | 168.9 | 168.69 |
| 000000201302 | 7664 | 217.39 | 218.51 | 224.8583 | 161.21 | 156.16 | 156.185 |
| 000000201303 | 7245 | 137.53 | 136.913 | 139.4174 | 139.4 | 147.2 | 140.674 |
| 000000201304 | 7632 | 150.13 | 147.32 | 145.769 | 143.98 | 140.61 | 133.183 |
| 000000201305 | 5477 | 101.8 | 99.5339 | 99.08158 | 99.003 | 101.11 | 102.271 |
| 000000201306 | 7109 | 127.27 | 136.682 | 125.1851 | 134.73 | 130.73 | 138.185 |
| 000000201307 | 8419 | 155.35 | 153.771 | 149.3497 | 157.5 | 158.4 | 156.001 |
| 000000201308 | 8406 | 166.35 | 151.285 | 165.3672 | 157.81 | 153.41 | 157.757 |
| 000000201309 | 7028 | 142.29 | 135.159 | 130.8299 | 132.03 | 131.01 | 139.135 |
| 000000201310 | 7081 | 131.21 | 137.208 | 135.1174 | 128.75 | 143.05 | 130.478 |
| 000000201311 | 2076 | 47.998 | 47.1995 | 47.13052 | 47.018 | 46.994 | 42.205 |
| 000000201312 | 7194 | 137.77 | 136.647 | 139.4563 | 139.48 | 142.46 | 136.638 |

### 6. Other Scenarios

In addition to the above basic scenarios, other possible scenarios include depth and spatial area known, depth and time-span known, spatial area and time-span known, and all three known. Data retrieval in these scenarios can be easily implemented by comprehensive application of the above three basic scenarios, and this code implementation is not explained here.

## VI. PERFORMANCE EVALUATIONS

To verify the efficiency of accessing massive structured ocean environment data on a big data platform, we used 4 personal computer machines to form the experimental cluster and carry out the performance test.

### 1. Experimental Environment

The experimental environment is described in Table 3.

### 2. Results of Experiment

Using global reanalysis data of temperature and salinity coming from the CMOC/CHINA website (http://www.cmoc-china.cn), different scenarios of data retrieval time, for either depth known, or depth, year, and month known, were recorded in this experiment. The results are presented in Table 4 and Table 5.

Curve graphs of the data retrieval performance are for the data provided in Table 4 and Table 5.

From the data listed in Tables 4 and 5, and the curves depicted

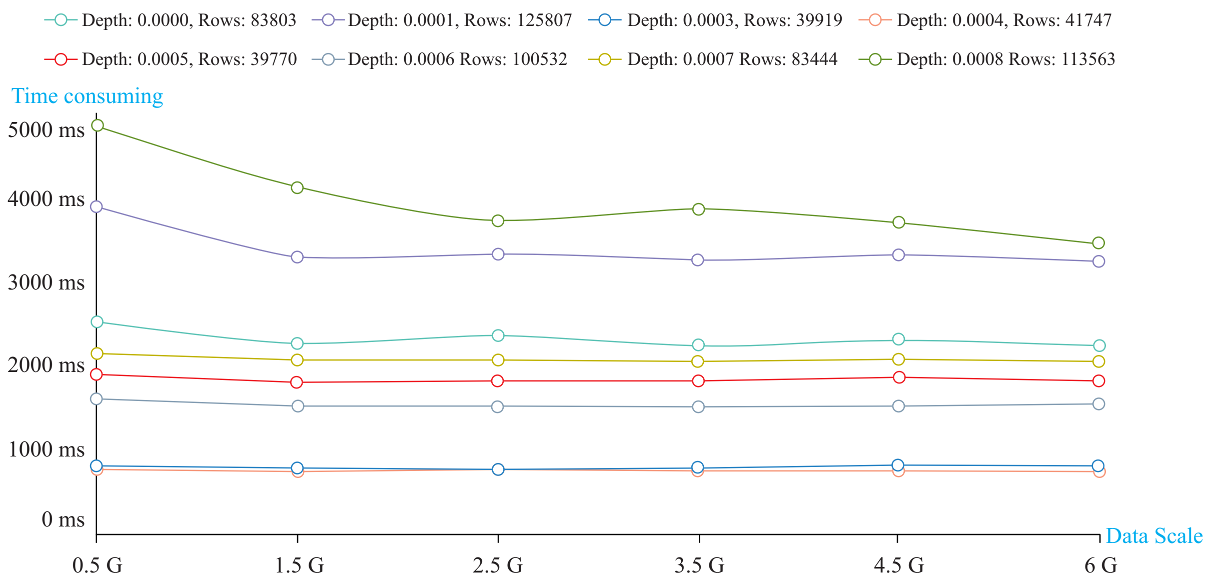


**Fig. 14. Data retrieval performance under the conditions of only depth known.**

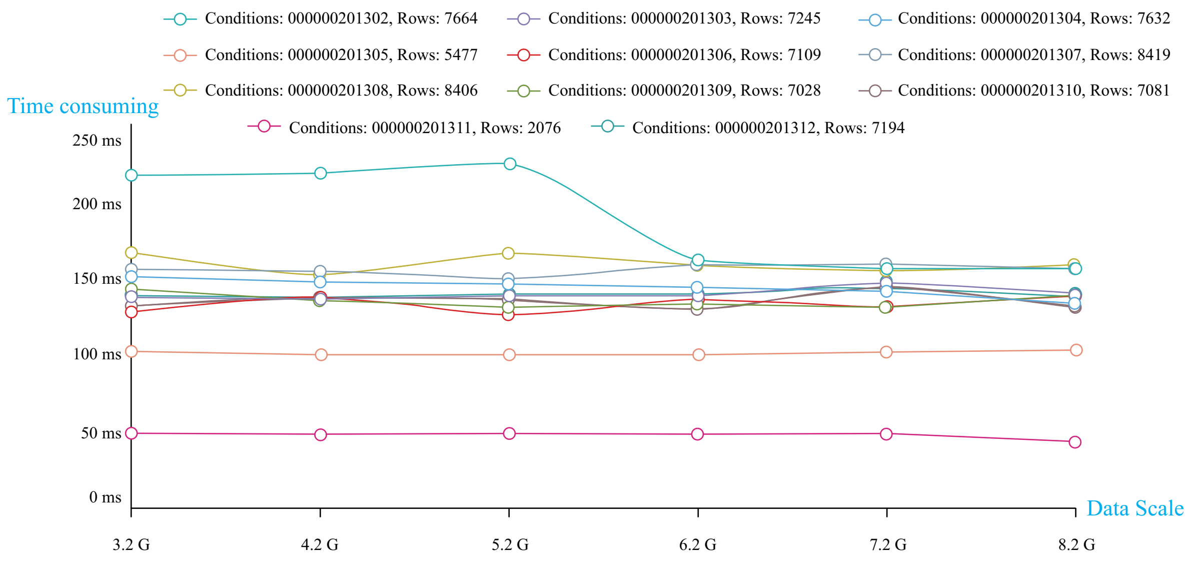


**Fig. 15. Data retrieval performance under the condition of depth and time known.**

in Figs. 14 and 15, it can be concluded that retrieval efficiency of big structured marine environmental data is not affected by an increase in the amount of data in the big data platform constructed by Hadoop and Hbase. At the same time, the speed of data acquisition is very high.

## VII. CONCLUSIONS

In this paper, we described an integration and sharing solution that can integrate and manage a large amount of structured marine environmental data efficiently and flexibly. This solution takes big marine environmental data containing features such as temperature, salinity, density, and marine current, and stores it in a form based on Bigtable technology, which is a distributed storage system for structured data proposed by Google. Through modeling big marine environmental data, this solution can integrate a variety of data formats from a variety of sources. The solution differs from related previous work in that it achieves centralized management of data through a unified interface and distributed storage using Hadoop and Hbase. The performance of infinite lateral data expansion provided by this solution meets the needs of marine big data storage and sharing. Efficient and complicated query operations can be executed using this platform, and real-time visualization of ocean features can be implemented.

## ACKNOWLEDGEMENTS

We acknowledge support from National Natural Science Foundation of China (No. 61403282). The authors would like to thank Suixiang Shi, Director of National Marine Data & Information Service, who provided the initial idea to develop a marine data integration system that can store and share big marine data. The first author also thanks Feng Zhang, Vice-Director at the Key Laboratory of Digital Ocean, who substantially supported the deployment of the test system. Lastly, we would like to thank Qinge Wu at the College of Electric and Information Engineering, who provided professional proof-reading for this paper.

## REFERENCES

Balkić, Z., D. Šoštarić and G. Horvat, (2012). GeoHash and UUID identifier for multi-agent systems. In Agent and Multi-Agent Systems. Technologies and Applications (pp. 290-298). Springer Berlin Heidelberg.

Bell, G., T. Hey and A. Szalay (2009). Beyond the data deluge. Science 323 (5919), 1297-1298.

Bryant, R. E. (2011). Data-intensive scalable computing for scientific applications. Computing in Science & Engineering 13(6), 25-33.

Butler, D. (2006). Virtual globes: The web-wide world. Nature 439(7078), 776-778.

Chang, F., J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes and R. E. Gruber (2008). Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS) 26(2), 4.

Chen, C. L. and C. Y. Zhang (2014). Data-Intensive Applications, Challenges, Techniques and Technologies: A Survey on Big Data. Information Sciences.

Gewin, V. (2004), Mapping opportunities. Nature 427 (6972), 376- 7.

Groot, R. (1997). Spatial data infrastructure (SDI) for sustainable land management. ITC journal 3(4), 287-294.

Knoppers, B. M. and A. M. Thorogood (2017). Ethics and Big Data in Health. Current Opinion in Systems Biology.

Merino, J., I. Caballero, B. Rivas, M. Serrano and M. Piattini (2016). A data quality in use model for big data. Future Generation Computer Systems 63, 123-130.

Shi, S., Y. Liu, H. Wei, B. Qiao, G. Wang and L. Xu (2013). Research on cloud computing and services framework of marine environmental information management. Acta Oceanologica Sinica 32(10), 57-66.

Strain, L., A. Rajabifard and I. Williamson (2006). Marine administration and spatial data infrastructure. Marine Policy 30(4), 431-441.

Szalay, A. (2011). Extreme data-intensive scientific computing. Computing in Science & Engineering 13(6), 34-41.

Tabaa, Y., A. Medouri and M. Tetouan (2012). Towards a next generation of scientific computing in the cloud. International Journal of Computer Science 9(6), 177-183.

Taylor, R. C. (2010). An overview of the Hadoop/MapReduce/Hbase framework and its current applications in bioinformatics. BMC bioinformatics 11 (Suppl 12), S1.

Voyles, P. M. (2017). Informatics and data science in materials microscopy. Current Opinion in Solid State and Materials Science 21(3), 141-158.

Wang, H., Z. Xu, H. Fujita and S. Liu (2016). Towards felicitous decision making: An overview on challenges and trends of Big Data. Information Sciences 367, 747-765.

Xu, B, S. Yan, Q. Wang, J. Lian, X. Wu and K. Ding (2014). Geospatial data infrastructure: The development of metadata for geo-information in China. IOP Conference Series: Earth and Environmental Science. IOP Publishing 17(1), 012259.

Zhang, X., W. Dong, S. Li, J. Luo and T. Chi (2011). China digital ocean prototype system. International Journal of Digital Earth 4(3), 211-222.