A Big Grid Data Manage Method and Its Implementation

Xiang-yu YU\(^1\) and Bo CAI\(^2,a,*\)

\(^1\) Hubei Subsurface Multi-Scale Imaging Key Laboratory, Institute of Geophysics and Geomatics, China University of Geosciences, Wuhan 430074, China
\(^2\) The Information Center of Wuhan University, Wuhan 430074, China
\(^a\)bo_cai@yeah.net
\(^*\)Corresponding author

Keywords: Grid data, Big data management, NoSQL.

Abstract. With the rapid development of the sensor and data acquisition technology, problems of data management and query on the large-scale grid are frequently emerged in applications about Geo-Information sciences. This paper proposes a distributed data manage method for the big data on the large-scale grid. The method designs a series of storage and indexing mechanism with the features of NoSQL to implement an efficiently data storing, extracting and query for big data on the grid. In addition, the implement system based on the method brings a good user experience in actual works.

Introduction

In many applications about Geo-Information science, the earth observation data are usually described and stored with the form of grid such as DEM (Digital Elevation Model) and various Geophysical data [1]. And the grid is used to represent the sampling distribution of observations value on/under the ground. That is to say, the adjacent sampling points can be connected into rectangles [2]. With the development of data acquisition technology, these grid data gradually have some characteristics as large amount, various types, and fast growth [3]. How to implement an efficient management for the big grid data become a very practical problem.

According to the grid form, the grid data which commonly used can be divided into the regular grid data (each grid unit has the same size in the sample space) and the irregular grid data (the space size of each grid unit is unequal) [4]. Compared to the irregular grid data, the regular grid data is easier to manage. And it is usually stored in some standard format files such as *.grd format file (ASCII) and Geotiff format file (Binary) [5]. Similar to the management of the big image data, we can also use the file and relation database for a mixed management to the regular grid data. For example, a typical manage method is to divide the whole grid regularly and indexing each divided grid then use the relation database to store the index of the divided grid and store values of the divided grid in the file which using the index as the file name.

As for the irregular grid data, it is usually difficult to manage with a suitable data organization form in practical applications. The most frequently used method is using the quad (x-y-z-data) to record each grid point value on the file (ASCII) [6]. The first three items (x-y-z) represent the position information of the data sampling point, and the last item (data) records the acquired value on the point. As lacking more strictly limited in this data form, we can record the quad with an arbitrary order. For example, a data recording order is recording values on the x-y plane with the increasing of coordinate z; another recording order is recording values on the x-z section with the
increasing of coordinate y. The irregular grid data is also unsuited to be managed with the relation database. The reason is, if we directly record the collection of quad (x-y-z-data) to the table of relation database, we will encounter the problem of big table which can cause the low efficiency of data query and other data operations with the rapid increasing of the data size [7].

However, in actual works we can find above manage methods for large amount grid data is unable to meet needs more and more. Take an earthquake monitoring data management for a large area as an example, if using the common grid data managing method it is difficult to execute an arbitrary granularity query for the observation data on a specific time or position. This paper will describe a big data manage method and its implementation for the grid data based on the technology of distributed data storage. In the rest paper, the authors first give a detail description about the method and its several important components: the data storage process, the data extraction method, the data schema definition and the data query implementation. Then the authors introduce the system GPMS which is developed based on the method and finally give the conclusion.

Methodology

Data Storage Process

As the traditional relational database is difficult to meet the storage needs for the big grid data, the solution based on the distributed architecture and the NoSQL (Not Only SQL) database is an important alternative [8]. In addition, some characteristics of the grid data are also very suitable to store in the NoSQL database. Firstly, almost all NoSQL databases are based on the distributed hash table (DHT) [9]. And if we use the form of key-value to represent the quad (x-y-z-data), the quad (x-y-z-data) is equivalent to the mapping (position-data). Secondly, the massive collection of (position-data) is distributed to computer nodes for storing, but the query mode to the collection is always singleness. That is, query the value by the position, and the NoSQL database is also good at such query model. So, if we can adopt an effective method to encoding and indexing the position of the grid point, we can take advantage of the big data management capabilities of the NoSQL database. Moreover, we can attach more semantic information which needs to be extracted from the additional description of the grid data such as data type to the position encoding. The process of data storing is shown in Figure 1.

![Figure 1. The process of data storing.](image-url)
Data Extraction

The main purpose of the data extraction is to extract the grid information and the data value on each grid point from the grid data. For the regular grid data, the grid information (the number of grid cells, the size of each grid cell) is usually easy to obtain by a simple calculation. Even some grid data head files have provided the information directly. As for the irregular grid data, it is difficult to separate the grid definition information and the value on each grid point from the collection of quads (x-y-z-data). Therefore, the authors design a two-step data reading strategy, which implements an automatic data extraction by reading the collection of quads (x-y-z-data) twice. In the first data reading process, the order of the quad record is found, and coordinate sequences of the grid data are recorded (the sequence value of duplicate coordinate needs to be removed); the number of each grid point is identified. In the second data reading operation, data values corresponding to the number of grid points are obtained one by one. Due to the collection of quads (x-y-z-data) has six record orders. If we set the number of each grid point by the X-Y-Z order (in other words, the data is recorded by x-y planes with the increasing of coordinate z), the number converting rule from other orders to the X-Y-Z order is shown in Table 1. While in this table, \( i \) represents the number of records order in the respective data grid and \( xSize, ySize, zSize \), respectively represent the size of three coordinates sequence x, y, z.

<table>
<thead>
<tr>
<th>Data Order</th>
<th>The Number Calculation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-y-z</td>
<td>( i )</td>
</tr>
<tr>
<td>x-z-y</td>
<td>( \frac{(i%(xSize<em>ySize))}{xSize}</em>(xSize<em>zSize)+i/(xSize</em>ySize)*xSize+i%(xSize) )</td>
</tr>
<tr>
<td>y-x-z</td>
<td>( \frac{(i%(ySize<em>xSize))}{ySize}</em>(ySize<em>xSize)+i/(ySize</em>xSize)*ySize+i%(ySize)*xSize )</td>
</tr>
<tr>
<td>y-z-x</td>
<td>( \frac{(i%(ySize<em>zSize))}{ySize}</em>(ySize<em>xSize)+i/(ySize</em>xSize)*ySize+i%(ySize)*xSize )</td>
</tr>
<tr>
<td>z-x-y</td>
<td>( \frac{(i%(xSize<em>ySize))}{xSize}</em>(xSize<em>zSize)+i/(xSize</em>ySize)*xSize+i%(xSize)*ySize )</td>
</tr>
<tr>
<td>z-y-x</td>
<td>( \frac{(i%(zSize<em>ySize))}{zSize}</em>(zSize<em>ySize)+i/(xSize</em>ySize)*xSize+i%(xSize)*ySize )</td>
</tr>
</tbody>
</table>

Data Schema Definition

Three data schemas need to be defined in the data manage method we designed. They are: the grid information, the data value on each grid point and the meta-data of the data value (mainly about the data type and its unit). In order to separate and store these different data schemas, the NoSQL product we choose is MongoDB [10] as it uses the “collection” to play a similar role with the “table” in relational databases. In addition, the MongoDB also provides a good spatial index function to support the spatial query. Then the data schema definitions are as following:

The data schema of the grid information is defined as three components. 1) The identification of the grid. It can be generated by the grid’s created time and other description. 2) The range of the grid. It can be represented by the maximum and minimum coordinates of the grid. 3) The spacing size of the grid unit. For the regular grid, the spacing size is a constant triple. And for the irregular grid, we need to record the spacing size by three collections (x y z coordinates) for example \{(30, 40, 60, 70, 90), (20, 30, 35), (-20, -5)\}.

The collection of data value on each grid point is the core of the storage. And it not only needs to store large amount records but also needs to ensure a high efficiency of
data query. So, the data schema of this collection is designed by three fields: the identification of the grid point, the value on this grid point and the geographic spatial object of the grid point. We define the identification of the grid point as a code which combined with three elements: the grid ID of the grid point, the meta-data code of the data value and the number of the grid point recorded in the grid. Thus, in the identification of the grid point, it mixes the other two identification information which defined in their data schemas. The identification is expressed as an instance: “201509021730/D0/385”. The geographic spatial object of the grid point is stored for the spatial index and spatial query. It is described by the GeoJson [11], as an example “loc:{ type: "Point", coordinates: [73.97, 40.77] }”. If we use the longitude and latitude to represent the spatial coordinate, the spatial index we created should be specified as the “2dSphere” [12] just like the command “createIndex( { loc : "2dsphere" } )”.

For the meta-data of the data value, we use a collection to define several data types and their units. The meta-data collection is defined as Table 2. Then we use a char to represent the data type and a number to represent a kind of unit in the data type. And on the contrary, we can get the semantic information from a short code by searching in the collection. As in the above instance, the code D0 is used to represent the density data and its unit is g/cm³ referenced from the example of Table 2.

Table 2. The example of meta-data collection.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description of Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Char</td>
<td>D</td>
</tr>
<tr>
<td>name</td>
<td>Data type description</td>
<td>density</td>
</tr>
<tr>
<td>cells</td>
<td>Units</td>
<td>g/cm³, kg/m³</td>
</tr>
</tbody>
</table>

**Data Query**

Based on the data storage method mentioned above, we can further provide the data query method for the grid data. In common practical applications, there are two data query patterns for the grid data: the point query and the range query. While the point query is generally obtained the data value by a given point coordinates which users can pick the point by click their mouse on the digital map. And the range query result is usually acquired by the user’s defined range which users can use their mouse to pull a box. Of course, in order to achieve a better interactive effect, we can match a digital map with the grid data based on the range information of the grid. The digital map can be provided by the user or download from WMS endpoints as Bing, Google.

There are two ways to handle the request of the point query: the one is find the closest point to the query point on the grid and return the data value of the closest point; the other is firstly generate a buffer rectangle with the center of the query point, then compute the value by a distance weighted composition with all grid point in the range of the rectangle. Due to the implementation to the latter point query processing is similar to the range query which will be introduced below, here we give the processing about the implementation of the first point query handle.

1. Search data in the collection of grid information to find the grid which contains the query point and save the grid’s ID.
2. Search data in the collection of data value to find the record which ID is start with the saved grid’s ID and its spatial index is closest to the query point. As in the MongoDB, we can use the spatial query statement “$geoNear”.
(3) Return the query records.

The range query must be supported by the spatial index defined in the collection of data value. So, before the data query, we need to use the GeoJson object which type is "polygon" to describe the query range pulled by users. Then following steps will be performed.

1. Search data in the collection of grid information to find the grid which contains or intersects the query range and save the grid’s ID.
2. Search data in the collection of data value to find records which ID is start with the saved grid’s ID and its spatial index is in the query range. As in the MongoDB, we can use the spatial query statement "$geoWithin".
3. As the range query result is usually a collection, we need to return the records after some data organization according to application needs.

Results and Conclusion

The GPMS is a big grid data management system which the authors developed based on the above method. And its English version is shown as Figure 2. It uses java and Eclipse RCP (http://www.eclipse.org) technologies and has been applied in some practical geological survey works. In theory, it can run on any number of computer nodes as users need. Besides the data query, functions like grid data import, matching and joint are also implemented comprehensively.

![Figure 2. The main interface of GPMS](image)

From the use effect of GPMS, the authors’ data manage method is especially suitable for characteristics of large amount, fragmentation and decentralization in actual grid data management works. So, the GPMS can bring a great convenience to the data mapping and archiving, information extraction and interpretation in many earth exploration and observation works. But the GPMS still has some works to improve which are mainly included following two points.

1. Combines the data visualization technology with the big grid data management to make users achieve the data more intuitively.
(2) Supports more interpolation operations for the grid data based on rules which users defined.

References


