GeoMapViz: a framework for distributed management and geospatial data visualization based on massive spatiotemporal data streams

Qi Xu, Longgang Xiang*, Haocheng Wang, Xuefeng Guan, Huaiyi Wu

State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China.

* Correspondence: geoxlg@whu.edu.cn; Tel.: + 027-6877-8751

Abstract. Spatiotemporal big data have multisource, heterogeneous, high-dimensional and spatiotemporal associations. Due to the limited computing and network resources, while the spatiotemporal data to be rendered are large and dynamic, efficient visual analysis has always been a popular topic and has had difficulty in the research of spatiotemporal big data. As one of the important means of big data visualization, thermal maps play an important role in expressing data flow, information flow, and trajectory flow. At the same time, the development of a distributed computing framework also provides technical support for the online calculation and visualization of spatiotemporal data streams. In response to the above problems, this paper designs and implements GeoMapViz, a distributed management based on massive spatiotemporal data streams and a multiscale geographic spatial visualization framework, which is oriented by the expression of thermal maps of massive point datasets. First, based on the concept of the tile pyramid model and spatiotemporal cube, we propose a thermal map sequential tile pyramid (TS_Tile) model, which realizes scalable storage and efficient retrieval of data flow. GeoMapViz adopts a high-performance Flink stream computing cluster to implement the large-scale parallel construction of hierarchical tile pyramids, implements distributed storage and index construction of data based on HBase and Geomesa, and uses GeoServer to manage the map service to provide a spatiotemporal range query interface. Finally, through using an open dataset as a system simulation test, the results show that the TS_Tile model can effectively organize large-scale, time-space and multidimensional thermal map data, and the query and visualization of the heatmap can reach a subsecond response. Furthermore, GeoMapViz supports the integration of the thermal map and original flow and provides a feasible solution for the visual analysis of large-scale spatiotemporal data.

Keywords: thermal map; tile pyramid; spatiotemporal query; distributed storage and index; data visualization

1. Introduction

With the rapid development of the global internet, the Internet of Things, cloud computing and mobile communication technology, ubiquitous human social behaviour is always accompanied by a large amount of data. These data show the complexity of multidimensional, semantic and spatiotemporal dynamic correlation because they contain temporal, spatial and attribute dimensional information [1]. Each data flow model can be defined by \( Object(S,T,V_n) \), where \( S \) represents the spatial location...
generated by the data flow, \( T \) is the temporal information of the data flow, and \( V_n \) is the \( n \) attribute mapping of the Object.

Currently, spatiotemporal data streams tend to diversify, and associative computation of spatiotemporal data streams is prevalent in many applications with high concurrent requests and large-scale data production. The traditional relational database gradually has bottlenecks in high concurrent reading and writing, efficient data storage, and scalability, which cannot meet the storage and management requirements of massive spatiotemporal data [2, 3]. Two schemes, file-level data management and nonrelational NoSQL database-level management, are often used in the storage management of data today. File-level data management [4-7] extends the local file system into a file system network with multiple data nodes, which achieves load balancing at the data storage level and relieves the pressure on computer storage by scaling horizontally. Many application studies have also been conducted by domestic and foreign scholars in data storage schemes for distributed file systems [8-10]. Unfortunately, file system-based storage solutions are usually suitable for offline data storage and computation and do not support real-time thermal map visualization with high temporal resolution and frequent updates.

A distributed nonrelational database represented by HBase shows great performance in practice because of its high reliability, high performance and scalability. In the practice of storing massive data based on HBase, there is a lack of research on the storage of temporal dimensions, while the storage of multiresolution tile data is mostly based on remote sensing images. For example, Liu et al. [11] used HBase to store massive image data and used MapReduce for scalable parallel processing. The data stream with spatiotemporal multidimensional information not only has spatiotemporal characteristics but also has multispatial resolution characteristics, which makes the storage scheme based on native HBase difficult to expand storage. Based on frequent updates of the hot map query requirements, HBase does not provide a secondary index mechanism, causing the multicondition application query requirements to have problems.

The storage scheme of data is also closely related to the spatiotemporal dissection scheme of data, and the use of a reasonable spatial data division method can solve the unified management and multiscale representation of massive spatiotemporal data at the global scale. The data-partitioning scheme includes the following: (1) spherical mesh subdivision based on polyhedron subdivision; and (2) spherical mesh generation based on geographic coordinates.

The spherical grid method based on polyhedron subdivision uses a regular polyhedron for data partitioning. For example, the quaternary triangular mesh (QTM, [12]) first divides the spherical surface into eight spherical triangles and then uses the form of a triangular quadtree for recursive subdivision to realize the discretization of data. Kimerling et al. [13] also extended the planar hexagonal grid to the spherical grid system to realize the hexagonal grid subdivision of the Earth projection plane. However, after the spherical subdivision of regular polyhedrons, there are inevitably deformations in size and shape, and the division of polyhedrons and the transformation of geographic coordinates are more complex.

Spherical grid subdivision based on geographic coordinate systems, such as Geohash, divides latitude and longitude information into two dimensions by dichotomy. This grid division method conforms to the law of people’s understanding of the Earth [14, 15]. The tile pyramid model is a multiresolution hierarchical structure, and the adopted idea is similar to the level of details (LOD) model, but its organizational object is map tiles. The division method of tiles is based on spatial position, and the division of the time dimension is ignored.

In contrast, in the study of visualization of data flow, data visualization technology is an important method in the field of big data analysis and data mining. It can show the original data flow through calculation and processing in a visual form and can help scholars uncover the laws hidden behind the data. For example, the point of interest (POI) of the taxi in New York can help the taxi company locate the hot area of the taxi. Traditional visualization technology stores and renders map resources through tile technology, which greatly improves the efficiency of webport access. However, its application is limited to image data and does not perform well in unstructured data applications. In the face of a big
data environment, classical geospatial map visualization (such as thermal maps) requires considerable time and cost to generate large-scale geospatial data maps, which is inefficient [16]. At the same time, after data preprocessing, most of them are deployed in different data platforms for decentralized storage. Due to the diversification of data types, the data standards of all walks of life are not uniform, which is not conducive to data sharing and publishing, and the efficiency of data integration is low. In the visualization of maps, such as Tableau, ArcGIS and Apache Zeppelin visualization tools, they work well on small- and medium-sized data, but they are difficult to extend to large-scale datasets. Tabula [17] used the sampling cube method to store the sample space and served the user-defined visualization tasks. MapD [18] used a GPU to parallelize, thus speeding up the query and pixelization in visualization, but it was limited to a single machine and had poor scalability. There are also many extended distributed visualization-processing frameworks, such as GeoSparkViz [19], HadoopViz [20], SHAHED [21], MAP_Vis [22], parallelized scatter maps, heatmaps and other map image-rendering pipelines, to realize the organization and visualization of spatiotemporal big data. Most implementations are concentrated in the distributed data-processing platform, using the batch-processing method of data, which has large data delay and does not support the visualization requirements of high real-time data.

In view of the above problems, this paper designs and implements GeoMapViz, a distributed management and multiscale geospatial visualization framework based on massive spatiotemporal data streams. Through distributed parallel computing on large-scale POI data (taxi data, check-in data, and traffic information) to achieve rapid online organization, distributed storage to multiple nodes establishes a spatiotemporal data index and thermal map-oriented visualization of data stream rendering. Based on the concept of the tile pyramid model and spatiotemporal cube, GeoMapViz proposes a thermal map sequential tile pyramid (TS.Tile) model, which realizes scalable storage and efficient retrieval of data flow. GeoMapViz adopts a high-performance Flink stream computing cluster to implement the large-scale parallel construction of hierarchical tile pyramids, implements distributed storage and index construction of data based on HBase and Geomesa, and uses Geoserver to manage the map service to provide a spatiotemporal range query interface. Finally, this paper forms a set of multiscale massive data visualization frameworks, GeoMapViz, which integrates data aggregation, computing, storage, service and visualization. GeoMapViz provides an open source visualization interface that integrates dynamic visualization rendering of thermal maps, visualization of geofences and original flow, and a spatiotemporal query association module.

The main contributions of GeoMapViz are as follows:

- Based on the concept of the tile pyramid model and spatiotemporal cube, GeoMapViz proposes a thermal map sequential tile pyramid model, which is called the TS.Tile model. The organization method of vector tile maps is used to save massive spatiotemporal big data into block data with multiple resolutions. The minimum unit of the spatiotemporal cube in logic is used as the thermal map mapping of the temporal tile, which can effectively organize large-scale thermal map tiles with time information.
- GeoMapViz encapsulates the map visualization process of spatiotemporal data streams, including rasterized spatial objects, pixel aggregation and other operations. Through the distributed ability of clusters, a large-scale parallel-processing process of thermal tiles is quickly constructed, and a multilevel pyramid model of thermal tiles is constructed in the form of TileDataStream without hierarchical aggregation of thermal tiles, which improves the computational efficiency and real-time support of the system.
- GeoMapViz forms a data-processing pipeline that publishes the preprocessing results of data flow as a map service to achieve data integration and sharing. The Geoserver platform is used to perform multiclass queries, including spatial queries, attribute queries, and spatiotemporal queries, with higher readability and applicability.
- GeoMapViz forms a set of multiscale massive data visualization schemes that integrate data aggregation, calculation, storage, service and visualization. The system integrates the modules of dynamic visualization rendering of thermal maps, visualization of geographic fences and
original flow, and spatiotemporal query association and provides an open source user interface to realize the interactive experience of spatiotemporal data flow.

In view of this prospect, the rest of this paper is organized as follows: Section 2 describes the construction method for organizing large-scale spatiotemporal data streams through the TS_Tile model and illustrates the system framework of GeoMapViz. Section 3 tests and evaluates the system through taxi datasets as use case scenarios. Section 4 provides a summary.

2. Functions and framework of GeoMapViz

2.1. Thermal Map Sequential Tile Pyramid (TS_Tile) Model

Considering the structure of data flow (S, T, and V), it is necessary to effectively organize and manage its spatiotemporal information and attribute information. Here, through two classical graphic concepts, the tile pyramid model and spatiotemporal cube are effectively combined to quickly build the TS_Tile model of the thermal map. The tile pyramid model provides a spatial information reference for the TS_Tile structure and constructs spatial divisions at different levels by LoD. The spatiotemporal cube provides a multidimensional extension for the TS_Tile structure and maps and abstracts the time dimension and attribute dimension information through the cube structure, which can effectively store and apply the data flow information.

2.1.1. Tile Pyramid Model

The tile pyramid is a multiresolution hierarchical structure that can organize the area in the same spatial range according to the decreasing resolution from the bottom to the top of the pyramid and the decreasing number of tiles. In the field of geographic information, the tile pyramid model was initially applied to the image pyramid. Through tile-slicing technology, the world map was formed into a multilevel image dataset according to the recursive subdivision principle of a quadtree.

The tile pyramid model adopts a global unified subdivision framework and realizes the transformation of spherical coordinates to a two-dimensional plane through Web Mercator projection. Because of the equiangular projection property of Web Mercator, its spatial range is [85.01°S, 85.01°N], [180°W, 180°E]. When the spatial level = 0, it is a square with the geographic scope of the world map. On this basis, quadtree recursive subdivision is carried out. Each low-level tile corresponds to four high-level tiles, forming a stepped tile map dataset, as shown in Figure 1.

![Figure 1. Map projection and the tile pyramid model.](image)

The tile coordinate system needs to be defined in the tile pyramid model to uniquely identify each tile in the pyramid model. The upper left corner is used as the coordinate origin, the longitude direction is the X axis, the latitude direction is the Y axis, and the scaling level is Z. Therefore, each tile can be uniquely identified by (X, Y, and Z), as shown in Figure 2, where each tile contains 256*256 pixels.
The image pyramid slice based on the grid slice is the image format after completion: (1) the style is fixed, and the flexibility is poor. (2) There is also a lack of real time and cannot be updated in real time, and grid tiles need to be regenerated when data information changes. (3) Attribute representation information is limited, and grid slicing cannot obtain attribute information. Due to the needs of zooming and roaming, the thermal map can be organized by tile pyramid technology, and users can schedule and render on demand according to needs.

Based on the above problems, vector tile data are more scalable, which is defined as:

- **Pixel**
  Each tile in the pyramid model is composed of $256 \times 256$ pixels, and the pixel is the smallest construction unit of the pyramid model, which represents a spatial partition area of the finest grain. $Z$ is the spatial level, and $X$ and $Y$ are the positions in the tile coordinate system, where $Z$-$X$-$Y$ constitutes the parent node key of the pixel set. Value is the attribute mapping of pixel units (aggregation expression of multiple attributes, such as sum, average, count, max, min, and exist).

  $$\text{Pixel} = \{Z, X, Y, \text{Value}\}$$  \hspace{1cm} (1)

- **Tile**
  Tile represents each map tile in the pyramid model, where $Z$, $X$, and $Y$ represent the position of the tile in the pyramid model, which can be used as the only specified tile area of the index, including a set of $256 \times 256$ pixels ($n = 256$).

  $$\text{Tile} = \{Z, X, Y, \bigcup_{j=1}^{n} \text{Pixel}\}$$ \hspace{1cm} (2)

- **TileMap**
  TileMap represents a map tile at a spatial level in a pyramid structure, which determines the map range at the $Z$ level, and the tile collection when $Z$ is the key.

  $$\text{TileMap}_i = \left\{Z, \sum_{m=1}^{Z} \sum_{n=1}^{N} X_m, Y_n, \bigcup_{j=1}^{256 \times 256} \text{Pixel}\right\}$$ \hspace{1cm} (3)

- **Pyramid**
  Pyramid is the overall structure of the tile pyramid, which is a set of TileMaps at multiple spatial levels.

  $$\text{Pyramid} = \left\{\sum_{i=0}^{Z} \text{TileMap}_i\right\}$$ \hspace{1cm} (4)

2.1.2. **Data Cube**

The data cube [23] is a multidimensional data model that allows users to analyse and query data from multiple perspectives and levels and is widely used in online analytical processing (OLAP) for big data management and organization. The data cube is built on top of a multidimensional data model, with each dimensional variable being able to be indexed for data access. The data cube has no heterogeneity, and the dimension range expressed in each cube is consistent. However, due to the uneven distribution of data, the data cube has a problem of data tilt. Many cube spaces are sparse, occupying a large amount of storage space. Therefore, the unit size of the cube dimension also
determines the storage efficiency of the cube. The data cube was first proposed by Grey et al. [24] to provide multidimensional analytical operations: drill-down, roll-up, slice, dice and pivot. Through the integration and subdivision of data, data can be divided from different angles and different levels to meet the analytical needs at different levels.

When the spatial and temporal dimensions are applied to the data cube, it becomes a spatiotemporal cube model. The three branches of the data cube are replaced by longitude, latitude and time, and the entire data space is divided into the same spatiotemporal unit. Each spatiotemporal unit has different partition granularity in each direction, and the obtained spatiotemporal unit size is also different. The dividing granularity of the spatiotemporal cube determines the dividing line position of the data in a certain dimension. The dividing position of the latitude and longitude grid of the thermal map object is closely related to the level of the scaling level, and the time dividing granularity is consistent among the levels. Each spatiotemporal unit represents the dataset with the same spatial and temporal range.

Since there is no upper bound on the time axis, the interval partition on the time dimension can be realized by the time bin and used as a coarse-grained time window. In a spatiotemporal cube, the time information of all data comes from the same time window (as shown in Figure 3); therefore, the multilevel temporal tile cube can be constructed.

![Figure 3. Spatiotemporal cube model.](image)

2.1.3. TS_Tile Model

Based on the concept of space division and hierarchy of tile pyramids and the multidimensional expansion of the data cube, the TS_Tile model can be constructed, which can effectively store and manage thermal tiles with multidimensional time and space information.

![Figure 4. TS_Tile model.](image)
As shown in Figure 4, the TS_Tile model divides the data stream information with spatial and temporal multidimensional information according to the global unified subdivision framework, forming a two-dimensional grid area on the plane and stretching the attribute dimension information longitudinally. At each level, combined with the time window mechanism of Flink stream processing, by designing the predetermined time window size, the spatial data are extended horizontally to form a single level of the spatiotemporal cube model. Between multiple levels, the space-time cubes still form a pyramid model. In the organization of the data model, the time information is divided into interval forms to form a multidimensional spatial dataset. However, in the data visualization, the deadline point of the time window is used. The time dimension and attribute dimension of the spatiotemporal cube need to be mapped to the two-dimensional data slice to show the visualization effect of the sequential continuous thermal map.

Similarly, the definition of the TS_Tile model is as follows:

- **ts_Pixel**
  
  We define the minimum construction unit in the TS_Tile model as a space-time pixel (ts_Pixel), which represents a space partition region with the finest granularity. It is the pixel unit of a pyramid structure existing in the window sequence bin at a certain time window length.

  \[
  ts\_Pixel = \{Bin_n (Z,X,Y,Value)\}
  \]  

- **ts_Tile**

  Spatiotemporal tiles (ts_Tile) represent the pyramid map tiles with time series identification, including the ts_Pixel set with the same time mark and attribute value.

  \[
  ts\_Tile = \{Bin_n (Z,X,Y, \bigcup_{i=1}^{n} ts\_Pixel)\}
  \]  

- **ts_TileMap**

  The spatiotemporal map (ts_TileMap) denotes the map under a certain spatial level in the pyramid structure with a certain time window.

  \[
  ts\_TileMap = \{Bin_n \left[Z, \sum_{n=1}^{256} \sum_{j=1}^{256} X_{m,j}, \bigcup_{i=1}^{n} ts\_Pixel \right] \}
  \]  

- **ts_Pyramid**

  The spatiotemporal pyramid (ts_Pyramid) is an extension of the tile pyramid in the attribute dimension. It is a set of ts_TileMap under multiple spatial levels and a time window unit in the TS_Tile model.

  \[
  ts\_Pyramid = \{Bin_n \left[ \sum_{i=0}^{Z} ts\_TileMap \right] \}
  \]  

- **TS_Tile**

  TS_Tile represents a pyramid set under multiple time windows and constitutes a pyramid sequence in the time dimension.

  \[
  TS\_Tile = \left\{ \sum_{i=0}^{N} Bin_i (ts\_Pyramid) \right\}
  \]

The advantage of the TS_Tile model is that it can effectively combine Flink’s time window-processing mechanism to construct the pyramid model of map tiles in parallel in a single time window by grouping the time window bin as the key. The traditional tile pyramid needs to be aggregated by high-level tiles to generate lower-level map tiles. This requires considerable time, affects the real-time visualization efficiency of the data stream, and enhances the delay of data. Through the distribution mechanism, the data stream is carried in parallel to the map network at all levels, which can realize the parallel construction of map tiles at all levels.
8

Figure 5. Parallel construction of the TS_Tile model.

As shown in Figure 5, when the data stream information enters the tile pyramid construction window, it is assigned to the tile area of the information on the level map from Level = 0 to Level = n according to its geographic location information. Time information determines the tile map of the data stream on the spatiotemporal cube. The tiles determined by the space dimension and the time dimension are used to encapsulate the thermodynamic data in this concept to determine the tile position of the data flow between different levels.

2.2. Distributed TS_Tile storage structure and index encoding

2.2.1. Distributed Storage in HBase

GeoMapViz data storage is based on a distributed HBase database. The basic unit stored in HBase is CELL, representing a set composed of RowKey, Column Family, Qualifier and Version, namely:

\[
CELL = \{\text{RowKey}, \text{Column Family}, \text{Qualifier}, \text{Version}\}
\]  

Since the RowKey screening ability is the strongest in HBase, the design of RowKey in HBase directly affects the data retrieval efficiency. There are two types of data tables in HBase; namely, high table and wide table. Considering the frequent updating of the tiles of the sequential thermal map, the time information is integrated into the design of RowKey, and the data-filtering function can be realized by sliding the time window. Therefore, the data structure presents the high table property, and a ts_Tile is represented by a row.

Table 1. Tiles Storage structure for the TS_Tile model.

| RowKey          | Column Family | PK     | Tile_id | WindowEndTime | Tile_result |
|-----------------|---------------|--------|---------|--------------|-------------|
| RowKey_1        |               | PK_1   | Tile_id_1| WindowEndTime_1| Tile_result_1|
| RowKey_2        |               | PK_2   | Tile_id_2| WindowEndTime_2| Tile_result_2|
| RowKey_3        |               | PK_3   | Tile_id_3| WindowEndTime_3| Tile_result_3|

Table 1 shows the logical structure of tile data storage in HBase, and the key value is still used in physical storage. PK is the main key of the data table, which is consistent with RowKey in composition. PK adopts a space-time (T-S) serial-coding scheme, and the main key design method can quickly filter out large-scale time series through time information to improve the retrieval efficiency of time series tiles. The T-S code can explain the temporal and spatial information of tiles so that only one thermal map tile is identified.

Tile_id appears as tile attribute information, representing tile spatial information coding (layer, row, and column). Geomesa can establish attribute indices for tiles through spatial attribute fields to improve the efficiency of data indices. WindowEndTime also appears as an attribute field because the thermal map tile is a spatiotemporal cube and represents section information in the time dimension. In
2D map visualization, it is necessary to map the time dimension and identify the time window in this range with the end time, representing the data falling in this time window.

Tile_result is the sum of ts_Pixel recorded in the spatiotemporal cube, which is divided into 256*256 pixel planes after mapping. Since the thermal map tiles have data aggregation and obvious data tilts in space, in many thermal map tiles, the pixel matrix shows sparse matrix properties. GeoMapViz only stores pixels with thermal value data to improve the spatial utilization of storage. Finally, the Avro data serialization tool is used to compress the pixel matrix, and Base64 is carried out, which can effectively improve the data transmission efficiency.

2.2.2. Space-time Coding and Index Construction

To map the abstract concept of map tiles into one-dimensional data coding, we need to introduce the space-filling curve to realize the conversion of space-time units to one-dimensional binary coding and store it as RowKey of HBase. The spatiotemporal unit contains the information of the time dimension and spatial dimension; therefore, the information of the time dimension and spatial dimension is mapped to the spatial-filling curve to form the RowKey of the data table.

(1) Space coding: The space-filling curve is used to connect the space-time units at each level to realize the mapping from high-dimensional spatial information to one-dimensional binary coding. The geohash-coding method is introduced in the space-coding part, and the geohash-coding method is applied in the tile coordinate system to form space coding.

(2) Time coding: Time granularity also affects the length of the time-coding sequence. Common time granularity includes year (Y), month (M), week (W), day (D), time (H), minute (M), and second (S). Different application requirements correspond to different time granularities, which accurately depict the time accuracy that you want to give attention to. Due to the characteristics of frequent updating of the thermal map, we define the time window as the time granularity by dividing (M), taking the end time of the first tile time window of the dataset as the starting point, compressing the time stamp information in the original YYYY-MM-DD HH: mm: ss format, and using the offset (s) of the first tile time stamp of the thermal map as the time encoding.

Different concatenations of space coding and time coding also affect the storage structure of data in the database. There is also spatiotemporal coding, such as Z3, but it is found that in most cases, due to the difference in spatiotemporal scale, the query efficiency of the Z3 index strategy is much lower in practice. GeoMapViz uses a T-S series connection, uses the filtering ability of T, can quickly filter the data in a non-time window, and improves the efficiency of data retrieval. The serial encoding method is shown in Figure 6.

![Figure 6. Space-time coding.](image)

Combined with the multiple indices provided by Geomesa, it can support the rapid retrieval of a dataset and return the spatiotemporal query results in a spatiotemporal query.
2.3. **Thermal Visualization Algorithm**

The thermal map data organization maps the spatiotemporal data flow into the data tiles in the spatiotemporal cube and logically forms the data partition. Thermal map visualization includes three processes: rasterizing the original data flow, pixel aggregation and visual rendering to form a map visualization pipeline, as shown in Figure 7. Its core lies in the process management and parallel processing of data flow processing, which can quickly assemble and render the spatiotemporal data flow in clusters and can present the distribution of data in the form of a thermal map.

![Figure 7. Thermal map visualization process.](image)

**2.3.1. Data Flow Mapping**

The rasterization of spatiotemporal data flow to pixels: this rasterization process is a mapping process that contains a large number of trajectory data flows and time windows as input and converts spatiotemporal objects into pixel space under multimap pixel resolution in parallel. In the tile coordinate system, there are not only tile coordinates but also pixel coordinates in the tile. The purpose of rasterization is to map the longitude and latitude of the data flow to the pixel of the corresponding tile so that each pixel mapped by the point object has a coordinate/position in the screen coordinate system. In this paper, the tile coordinates and pixel coordinates from the WGS1984 geographic coordinate system to the Web Mercator projection coordinate system are realized by Equation (11):

\[
\begin{align*}
    \text{Tile}_X &= \text{Math.floor} \left( \frac{\text{Longitude} + 180}{360} \times 2 \hspace{1mm} \text{Level} \right) \\
    \text{Tile}_Y &= \text{Math.floor} \left( \frac{1}{2} \ln \left( \frac{\tan(\text{latitude} \times \pi / 180) + \sec(\text{latitude} \times \pi / 180))}{2 \times \pi} \right) \times 2 \hspace{1mm} \text{Level} \right) \\
    \text{Pixel}_X &= \frac{\text{Longitude} + 180}{360} \times 2 \hspace{1mm} \text{Level} \times 256\%256 \\
    \text{Pixel}_Y &= \left( \frac{1}{2} \ln \left( \frac{\tan(\text{latitude} \times \pi / 180) + \sec(\text{latitude} \times \pi / 180))}{2 \times \pi} \right) \times 2 \hspace{1mm} \text{Level} \times 256\%256
\end{align*}
\]  

Through rasterization, massive data streams are distributed in parallel to the corresponding time window, map tiles under each resolution, and further mapped to pixel units. The pixel unit logically presents the key-value pair of position information with data flow, and the tile stores the set of pixel pairs <pixel, value> within its tile range. Because the data stream used in this paper is point information, we only need to consider the point-to-pixel mapping. If the data stream is linear and planar, we can map the pixel trajectory of the line through the Bresenham line-scanning algorithm [25].

**2.3.2. Data flow convergence**

The aggregation of spatiotemporal data streams: through the rasterization operation of data streams, many overlapping pixels can be mapped at the same position in the screen coordinate system. The reasons for this are as follows. 1. Some spatial objects essentially overlap/intersect with each other and
have spatial proximity characteristics. 2. Maps have different resolutions at different zoom levels; therefore, many objects overlap/intersect at low resolutions. However, in map visualization, each pixel can only correspond to an exact thermal value and display the colour of the pixel. Therefore, this paper aggregates the spatiotemporal data flow within the pixel and calculates the weight of overlapping pixels to determine the final weight of the pixel.

The aggregation strategy of pixels is different due to the different application scenarios of data. In the visualization requirements of the thermal map, the data flow is mapped to the pixel through the location. Since there is no numerical difference in thermal values between data, the cumulative strategy is used to record the number of data streams in pixels. After Flink’s FlatMap conversion operation, the tiles (Z, X, and Y) are grouped by the Keyby operation, and each pixel is placed in a key value pair of the HashMap. The key is the pixel, and the value is the current weight of this pixel. When traversing a pixel, if a pixel already exists in the hash map, the aggregation strategy is used to update the weight stored in the HashMap.

The calculation methods of topics can be divided into two types: full calculation and incremental calculation. In some complex calculation logics, a full calculation method is needed. In some simple logical calculations, such as maximum, minimum and count, the incremental calculation method can effectively save the processing time cost of data, and the cost is to require additional storage space to save the intermediate results of data. In the pixel aggregation operation of the thermal map, the incremental calculation method can be used to calculate the thermal value in each pixel of the tile.

### 2.3.3. Visualization of thermal values

**Thermal map visualization algorithm:** After determining the final weight of each pixel by a pixel aggregation operation, the colouring operation can render each pixel according to its weight value. The relationship between colour and weight is usually defined by mathematical equations. After the thermal value is linearly stretched to the unidirectional value of [0 - 255], the linear Equation (12) and the piecewise colour Equation (13) are often used for pixel visualization.

\[
\text{Color}(\text{pixel}, \text{value}) = \text{Color}(\text{value}, 255, 255)
\]

\[
\text{Color} = \begin{cases}  
\text{Red} & \text{value} \in [0, v_0) \\
\text{Green} & \text{value} \in [v_0, v_1) \\
\text{Blue} & \text{value} \in [v_1, 255]
\end{cases}
\]

Through the experiment, it is found that due to the sparse data of tiles in a single time sequence, the visualization effect achieved by pixel colour mapping is not good; therefore, the gradual circle method is used for visualization. The effects achieved include:

1. Each pixel presents a radial gradual circle, and this gradual circle represents the radiation effect of the thermal value data from strong to weak. The central data are concentrated, and the data in the radial circular direction are dispersed.
2. Multiple circles can be superposed with each other, and they are linearly superposed, which shows the superposition of data strength in the radial range of data.
3. Colour mapping with intensity.

It is worth mentioning that when the circularity is filled with intensity chromatography, because the colour obtained in this way is three-dimensional, it is nonlinear when superimposed, and it is prone to data incompatibilities such as colour mutation. Therefore, the single colour channel in the RGBA colour channel is selected as the dimension representing the strength, the data value in the pixel is linearly mapped to a vector, and alpha is used as the measurement index of the outward radiation from the centre of the circle. After colouring all pixel values in the current view, the vector is mapped to the RGBA value through Equation (14). The colouring method is shown in Figure 8.
\[ \text{limit} = \ln(\text{value} + 1) / \ln(v_0) \]
\[ \text{Color}(r, g, b, a) = \begin{cases} 
  r = \text{Math.floor}(255 \times \min(1, \text{limit})) \\
  g = \text{Math.floor}(255 \times \min(1, \max(0, \text{limit} - 1))) \\
  b = \text{Math.floor}(255 \times \min(1, \max(0, \text{limit} - 2))) \\
  a = \text{Math.min}(1, \text{limit}) 
\end{cases} \]  

(14)

where \( v_0 \) is the controllable parameter, which can adjust the truncation value \( \text{limit} \) of colouring, and \( \text{value} \) is the thermal value reflected by the vector.

**Figure 8.** Gradient circle visualization algorithm.

The original data stream is rasterized mapping, pixel aggregation and visualization rendering to form a map visualization pipeline and to finally present the form of a thermal diagram. The method is illustrated in Algorithm 1.

**Algorithm 1:** Visualization of heatmap

Input: &lt;coordinate,datetime&gt; \( \text{dataFlow} \)
- \( \text{highLevel} = L0 \)
- \( \text{lowLevel} = L1 \)
Output: \&lt;pixel,rgba&gt; \( \text{tileList} \)
1. **Function** \( \text{Vizheatmap}(\text{dataFlow}, \text{highLevel}, \text{lowLevel}) \)
2. Create an empty \( \text{tileList} \);
3. Create an empty \&lt;Pixel, weight&gt; HashMap \( \text{HM} \);
4. **for** data in \( \text{dataFlow} \) **do**
5. **for** lowLevel &lt; Level &lt; highLevel **do**
6. \( \text{datebin} = \text{getBin(data.datetime)} \);
7. \( \text{Tile} = \text{createTile(Level, datebin)} \);
8. \( \text{pixel} = \text{rasterize(data.coordinate)} \);
9. **If** pixel in \( \text{HM} \) exist **do**
10. Aggregate the weight
11. **end if**
12. \( \text{Tile}.\text{add(pixel)} \)
13. \( \text{tileList}.\text{add(Tile)} \);
14. **end for**
15. **end for**
16. **for** Tile in \( \text{tileList} \) **do**
17. render \&lt;pixel,value&gt; \( \text{Tile in R vector} \)
18. **end for**
19. for Tile in tileList do
20. render <pixel,R>Tile in Rgba
21. end for
22. Return <pixel,rgba> tileList;

2.4. GeoMapViz visualization framework and other APIs

2.4.1. The GeoMapViz Framework
According to the proposed TS_Tile data organization model, we finally formed a set of multiscale massive data visualization frameworks, GeoMapViz, that integrate data aggregation, calculation, storage, service and visualization. The system integrates dynamic visualization rendering of thermal maps, visualization of geographic fences and original flow, and spatiotemporal query association modules.

As shown in Figure 9, the GeoMapViz visualization framework includes three parts: background server cluster, middleware and client.

The background server cluster is built in the distributed environment of multiple data nodes, supporting the pretreatment of the original data stream, completing the construction of the TS_Tile model, and providing data storage and index construction. The middleware realizes the management of large-scale thermal diagram tiles, provides a data access interface, and analyses the client’s request. The client is the Web port to display the heatmap and the original stream, to visualize the geographic fence and to provide a dynamic query visualization method.

2.4.2. Thermal Visualization API
In the provided real-time thermal map service, we transform the data streams based on Flink, a parallel data-processing framework with a time window, and map each data stream by the TS_Tile model. In the heatmap calculation, the tile coordinates map the original individual latitude and longitude points into different pixels to obtain the pixel stream and then forms the heatmap tiles by pixel aggregation, which solves the problem of stacking visualization of large-scale data.

In this paper, the computation is organized in tiles, and the data streams are partitioned and assigned time windows based on Tile_id, enabling the rapid construction of a large-scale parallel-
processing process of thermal map tiles on a Flink cluster, forming a multilevel thermal map time-series tile pyramid model without aggregating thermal map tiles level by level and improving computational efficiency.

2.4.3. Geofence Visualization API
Geofence is a new application of location-based services (LBS). By constructing a virtual fence, a virtual geographic boundary is surrounded to monitor the area of interest in real time.

In GeoMapViz, we define some regions of interest in advance and store them in the database permanently in GeoJSON format. After defining the original data stream as SpatialDataStream, it is necessary to register the geographic fence information as a broadcast stream to apply the joint operation of the data stream and the geographic fence to all concurrent instances when the data flow enters. A fence label is added to each data stream after the data flow enters, and then it is stored permanently.

GeoMapViz provides an integrated display of the thermal map and original flow and sets the visualization of spatiotemporal data flow points when the scaling level is greater than LEVEL. The original stream point information within the fence geographic scope is highlighted.

2.4.4. Spatiotemporal Query Association API
GeoMapViz provides a user-defined spatiotemporal query scheme. Users can customize the mapping of geographic fence boundaries and give a time range for spatiotemporal querying of geographic fences. The returned information includes the minimum bounding rectangle range of the geographic fence, the number of points in the geographic fence and the highlight display of the original flow point.

In addition, GeoMapViz also provides the optional function of dynamic query because GeoMapViz is a process of stream reading, processing and visualization of real-time data, which requires real-time display of data calculation results and high real-time data. After defining the geographic fence, when GeoMapViz follows the data flow processing for dynamic visualization, the optional method of dynamic query can be synchronized with the system time to realize the real-time follow-up of the system evolution time and to realize the continuous dynamic query effect.

2.5. External Map Service
Due to the diversification of data types, the data standards of all walks of life are not uniform, resulting in an information island and low efficiency of data integration. Therefore, it is necessary to achieve the unified management and organization of big data through technical means and to realize a standard format for the sharing of data. GeoMapViz uses Geoserver as a platform for data service publishing and management to support OGC map service standards and realize the sharing of spatial geographic information.

Web access to the published map service through the WFS service standard can quickly and accurately obtain the thermal map tile data in the corresponding spatiotemporal unit. The WFS service access is shown in Algorithm 2.

**Algorithm 2:** WFS service request for heatmap tile data

| Input: | address | map | layerName | epsg | filter |
|--------|---------|-----|-----------|------|--------|
| Output: | <object> heatmapData |
| 1. | Function loadWFS(map, layerName, epsg, cql_filter) |
| 2. | urlString = address; |
| 3. | param = { service: ‘WFS’, |
| 4. | version: ‘1.0.0’ |
request: 'GetFeature',
typeName: layerName,
outputFormat: 'application/json'
cql_filter: filter,
srsName: epsg
}
}
$\text{ajax}(\$
url = urlString + getParamString(param, urlString)
$.ajax({
  url: url,
  dataType: 'json',
  success: loadWfsHandler,
});
function loadWfsHandler(heatmapdata) {
  Return heatmapData;
}

The spatiotemporal range query returns all data in the spatiotemporal range according to the specified spatial range, such as the range of the view window \([X_1, Y_1], [X_2, Y_2]\), the determined scaling level and a given time point \(T_0\) or time range \([T_1, T_2]\). Geoserver provides CQL/ECQL filter language that supports spatial queries and attribute queries. In the integrated visualization of the thermal map and original data stream, all tiles in the view are obtained, and a set of ID serial encodings is formed to query the thermal diagram data. In the query of raw data flow, a time range is customized for the data flow through the concept of a time window. In the query, only the data flow information falling in the view space and time window is retrieved.

The spatiotemporal query is shown in Algorithm 3.

**Algorithm 3:** The spatiotemporal query of heatmap and original flow

**Input:** latLonBox \(=(X_1, Y_1, X_2, Y_2)\)
zoomLevel \(= L\)
queryTime, windowTime

**Output:** queryResult

1. Function spatioTemporalQuery(latLonBox, zoomLevel, queryTime)
2. If zoomLevel < L0
3.  tileBox = getTile(latLonBox);
4.  timecode = getTime(queryTime);
5.  foreach tile in tileBox do
6.    zOrder = getZCoding(tile);
7.    tileID = stCoding(timecode, zoomLevel, zOrder)
8.    idBox.add(tileID);
9.  end for
10. While idBox.hasNext() do
11.    result = getTileData(idBox.next());
12.    queryResult.add(result);  
13. end while
14. else
15.  pointBox = getGeo(latLonBox);
16.  timeRange = getRange(queryTime, windowTime);
17.  result = getPointData();
18.  If result.box in pointBox and result.t during timeRange do
19.    queryResult.add(result);
20. end if
21. \textbf{end if}
22. \textbf{Return} queryResult;

3. \textbf{Experiments and Discussion}

3.1. Environment and Dataset Description

The GeoMapViz framework is now deployed in the 4-node open source Tencent Cloud Linux cluster, node configuration and environment, as shown in Table 2.

\begin{table}[h]
\centering
\caption{Cluster configuration information.}
\begin{tabular}{lll}
\hline
Node & Configuration & Environment \\
\hline
1 & 4 Core, 8GB & Geoserver, Node \\
2 & 8 Core, 16GB & Flink TaskManager, HBase RegionServer, Kafka \\
3 & 8 Core, 16GB & Flink TaskManager, HBase RegionServer, Kafka \\
4 & 8 Core, 16GB & Flink JobManager, HBase RegionServer, HMaster, Kafka \\
\hline
\end{tabular}
\end{table}

In the framework of GeoMapViz, we utilize two experimental datasets: Xiamen and New York. Each dataset includes geospatial, temporal and attribute information. One of the datasets for the Xiamen region is from the Digital China Innovation Contest, DCIC 2020. The geometric feature type is point type, and the field features are listed in Table 3. The New York data were collected by a Taxicab and Livery Passenger Enhancement Programs (TPEP/LPEP) licenced technology provider and provided to the New York Taxi and Limousine Commission (TLC), with fields including pick-up and drop-off date/time, pick-up and drop-off locations, trip distance, line-by-line fare, rate type, payment type, and number of passengers reported by the driver. We preprocessed the dataset to ensure data quality and data validity. Since we focus only on the spatiotemporal representation of the trajectory data streams, we focus on extracting and computing the information in the temporal and spatial dimension fields of the data streams.

\begin{table}[h]
\centering
\caption{Xiamen trajectory fields.}
\begin{tabular}{lll}
\hline
Field & Type & Description \\
\hline
RUNNING\_STATUS & Int & Cab operation status, such as empty (1), carrying passengers (2) \\
SPEED & Double & GPS measurement of car driving speed \\
DIRECTION & Int & Driving direction angle \\
GPS\_DATE & Date & Satellite positioning time \\
CARNO & String & Car Track ID \\
PID & String & Car Record ID \\
\hline
\end{tabular}
\end{table}

Table 4 shows the summary of all datasets, including the number of records, the size of the original data, the time span and the number of aggregation tiles.

\begin{table}[h]
\centering
\caption{Dataset Information.}
\begin{tabular}{lllll}
\hline
Dataset & Time Span & Record & Data Size & Tile Number \\
\hline
Xiamen & 2019.06.07-2019.06.10 & 70 million & 5.9 G & 12 million \\
New York & 2009.01.01-2010.01.01 & 190 million & 6.3 G & 40 million \\
\hline
\end{tabular}
\end{table}

From Table 3, it is not difficult to find that the original records from Xiamen and New York are not the same as the final number of aggregated tiles. This difference can be attributed to the different sparse densities of the dataset, and the time window size defined by the two datasets due to different time ranges is also different, resulting in different proportions of aggregated tiles.

3.2. Data-Processing Experiment

GeoMapViz aims to use the streaming processing framework to form a set of real-time distributed data-processing and visualization frameworks. Therefore, it is necessary to coordinate the interaction parameters between the server and client to achieve synchronous access to the front and back ends. In the thermal map, the size of the time window affects the visualization effect of the data. If the time
window is too short, the data may be sparsely distributed due to fewer data. If the time window is too long, it will take a long time to obtain the thermal diagram of the period of interest. It is necessary to select the appropriate time window size according to the range of data. The relationship between parameters is shown in Equation (15).

\[
\begin{align*}
W &= \frac{24}{T} \times V \\
S &= \frac{W \times 60}{V}
\end{align*}
\]  

where V is the display time (s) of a thermal diagram at the front end, T is the total display time (m) of the actual 1-day data volume, W is the size of the time window (m), and S is the ratio of the actual time to the display time.

Figure 10 shows the visualization interface of the GeoMapViz framework, which includes three related views: dynamic visualization rendering of the thermal map, visualization of the geographic fence and original flow, and the spatiotemporal query association module. The system integrates the original flow on the thermal map, forming an integrated display of the thermal map and original flow, which is determined by the scaling level.
3.3. Verification of Spatiotemporal Query Efficiency

There are many ways to access thermal tiles. Geomesa can establish an ID index for the main key field of thermal tiles and can also construct an attribute index. Experiments show that compared with the attribute index, using the ID index can improve the efficiency of data retrieval. GeoMapViz’s query of thermal map tiles is a spatiotemporal query; that is, the window view is a spatial range and a time window. Since the time window is mapped to the end time of the time window to represent the entire spatiotemporal range, it can be queried by obtaining all data tile IDs in the window to form the ID set.

In this experiment, a variety of spatiotemporal thermal map query cases are defined, and Apache JMeter is used to test the query scheme. The experimental results are shown in Table 5 (total time includes WFS data acquisition, network transmission, thermal map data deserialization, map-rendering visualization, etc.).

**Table 5.** Thermal map query experiment.

| Dataset  | Window size | Level | Tile number | Data query(ms) | Total time(ms) |
|----------|-------------|-------|-------------|----------------|----------------|
| Xiamen   | 1 minute    | 12    | 15          | 15             | 280            |
| Xiamen   | 1 minute    | 16    | 24          | 17             | 160            |
| New York | 1 hour      | 12    | 15          | 16             | 260            |
In the verification of the spatiotemporal query of geofences, this experiment defines several fence regions with different sizes in advance and a variety of time spans (hours/days/months, taking the New York dataset as an example) to test the efficiency of the spatiotemporal query. The results are shown in Table 6 (total time includes WFS data acquisition, network transmission, geographic fence joining, and primitive flow visualization).

Table 6. Spatiotemporal query experiment of the geofence.

| Geofence Group | Bounding Box               | Time Span | Point Number | Total Time (ms) |
|----------------|---------------------------|-----------|--------------|-----------------|
| 1              | [-73.9914, 73.9901]       | 1 hour    | 18           | 120             |
|                |                           | 1 day     | 566          | 280             |
|                |                           | 1 month   | 9671         | 2500            |
| 2              | [-73.9955, 73.9950]       | 1 hour    | 4            | 110             |
|                |                           | 1 day     | 102          | 220             |
|                |                           | 1 month   | 1360         | 1350            |
| 3              | [-73.9806, 73.9796]       | 1 hour    | 19           | 115             |
|                |                           | 1 day     | 267          | 170             |
|                |                           | 1 month   | 2976         | 1200            |

It can be seen from the experimental results that in the query experiment of the thermal map, with the continuous change of time and space range, the time for WFS service to obtain data is constant at a level of 10 ms, and the overall visualization time fluctuates slightly at a level of 100 ms according to the number of tiles in the query. This shows that the TS_Tile model can adapt well to different spatiotemporal query sizes and can also ensure a subsecond response. In the spatiotemporal query experiment of geofences, the combined experiment based on different sizes of geofences and time intervals is finally attributed to the statistics of the number of original flow points falling in the spatiotemporal range. With the increase in the number of points, the visualization time of the system increases proportionally. Therefore, GeoMapViz can fit the real-time calculation of spatiotemporal information flow well and can realize the integration of dynamic visualization rendering of thermal maps, visualization of geographic fences and original flows, and spatiotemporal query association modules.

4. Conclusion and Prospects

Spatiotemporal big data have multisource, heterogeneous, high-dimensional and spatiotemporal correlations. Large and dynamic spatiotemporal data to be rendered have always been a popular and difficult topic in the research of spatiotemporal big data. Classical geospatial map visualization requires considerable time and cost to generate large-scale geospatial data maps, which is inefficient. To solve the above problems, based on the concept of the tile pyramid model and spatiotemporal cube, we propose the TS_tile model to achieve scalable storage and efficient retrieval of data flow. On this basis, we designed and implemented GeoMapViz, a distributed management based on massive spatiotemporal data streams and a multiscale geographic spatial visualization framework. GeoMapViz adopts a high-performance Flink stream computing cluster to implement the large-scale parallel construction of hierarchical tile pyramids, implements distributed storage and index construction of data based on HBase and Geomesa, and uses Geoserver to manage the map service to provide a spatiotemporal range query interface. According to the proposed TS_Tile data organization model, we finally form a set of multiscale massive data visualization frameworks, GeoMapViz, that integrate data aggregation, computing, storage, service and visualization. The system integrates dynamic visualization rendering of the thermal map, visualization of the geofence and original flow, and a spatiotemporal query association module.

To a certain extent, this paper solves the problems of management, organization and access of large-scale sequential thermal maps and provides a visualization scheme for thermal maps, which can be optimized from the following aspects in subsequent studies:
(1) GeoMapViz is organized through a time bin in the time dimension, which lacks data aggregation in the time dimension and lacks query support for different time window sizes in the query application scenarios of long-term sequences.

(2) Current visualization schemes, including rasterized pixels of spatiotemporal data streams and pixel aggregation operations, are all aimed at spatiotemporal points. In the future, a multidata-type thermal map tile pyramid structure will be formed to realize the visualization scheme of multithermal map themes.

(3) GeoMapViz focuses on the realization of real-time thermal map organization and visualization. HBase is used as a public data storage framework for historical and current moments, and more methods to improve query efficiency will be considered in the follow-up. Two sets of data storage schemes are formed to support multiple types of query requirements.

References
[1] Li D R, Ma J, Shao Z F 2015 On Spatio-temporal Big Data and Its Application Satellite Application 9 7-11.
[2] Jatana N, Puri S, Ahuja M, et al. 2012 A survey and comparison of relational and non-relational database International Journal of Engineering Research & Technology 1 1-5. [Linkout]
[3] Gyorödi C, Gyorödi R, Sotoc R 2015 A comparative study of relational and non-relational database models in a web-based application International Journal of Advanced Computer Science and Applications 6 78-83. [Linkout]
[4] McKusick M K and Quinlan S 2009 GFS: Evolution on Fast-forward: A discussion between Kirk McKusick and Sean Quinlan about the origin and evolution of the Google File System Queue 7 10-20. [Linkout]
[5] Borthakur D 2008 HDFS architecture guide Hadoop apache project 53 2. [Linkout]
[6] dos Santos M N and Cerqueira R 2006 Gridfs: Targeting data sharing in grid environments vol 2 (Singapore: IEEE) pp 17. [Linkout]
[7] Lei T 2009 Building Low-Cost Electronic Records Management System Based on MogileFS Modern Computer 0 170-1.
[8] Dittrich J and Quiané-Ruiz J A 2012 Efficient big data processing in Hadoop MapReduce Proceedings of the VLDB Endowment 5 2014-15. [Linkout]
[9] Fang J, Li C, Zhao Y, et al. 2018 Research on Distributed Storage Method for Big Data of Geological and Mineral Resources Based on Hadoop (Kunming: IEEE) pp 1-8. [Linkout]
[10] Tian C, Zhou H, He Y, et al. 2009 A dynamic mapreduce scheduler for heterogeneous workloads (Lanzhou: IEEE) pp 218-224. [Linkout]
[11] Liu Y, Chen B, He W, et al. 2013 Massive image data management using HBase and MapReduce (Kaifeng: IEEE) pp 1-5. [Linkout]
[12] Dutton G 1996 Encoding and handling geospatial data with hierarchical triangular meshes (Netherlands: Talor & Francis) pp 43. [Linkout]
[13] Kimerling J A, Sahr K, White D, et al. 1999 Comparing geometrical properties of global grids Cartography and Geographic Information Science 26 271-288. [Linkout]
[14] Liu J, Li H, Gao Y, et al. 2014 A geohash-based index for spatial data management in distributed memory (Kaohsiung: IEEE) pp 1-4. [Linkout]
[15] Xiang L, Wang D, Gong J 2017 Organization and efficient range query of large trajectory data based on geohash Geomatics Inf. Sci. Wuhan Univ. 42 21-27. [Linkout]
[16] Sarwat M and Nandi A 2017 On designing a geoviz-aware database system-challenges and opportunities (Arlington: Springer Cham) pp 384-7. [Linkout]
[17] Yu J and Sarwat M 2020 Turbocharging geospatial visualization dashboards via a materialized sampling cube approach (Dallas: IEEE) pp 1165-76. [Linkout]
[18] Root C and Mostak T 2016 MapD: A GPU-powered big data analytics and visualization platform ACM SIGGRAPH 2016 Talks pp 1-2. [Linkout]
[19] Yu J, Zhang Z, Sarwat M 2018 Geosparkviz: a scalable geospatial data visualization framework in the apache spark ecosystem (the 30th International Conference on Scientific and Statistical Database Management) pp 1-12. [Linkout]

[20] Eldawy A, Mokbel M F, Jonathan C 2016 HadoopViz: A MapReduce framework for extensible visualization of big spatial data (Helsinki: IEEE) pp 601-612. [Linkout]

[21] Eldawy A, Mokbel M F, Alharthi S, et al. 2015 Shahed: A mapreduce-based system for querying and visualizing spatio-temporal satellite data (Seoul: IEEE) pp 1585-96. [Linkout]

[22] Guan X, Xie C, Han L, et al. 2020 MAP-Vis: A Distributed Spatio-Temporal Big Data Visualization Framework Based on a Multi-Dimensional Aggregation Pyramid Model Applied Sciences 10 598. [Linkout]

[23] Gatalsky P, Andrienko N, Andrienko G 2004 Interactive analysis of event data using space-time cube (London: IEEE) pp 145-152. [Linkout]

[24] Gray J, Chaudhuri S, Bosworth A, et al. 1997 Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals Data mining and knowledge discovery 1 29-53. [Linkout]

[25] Bresenham J 1977 A linear algorithm for incremental digital display of circular arcs Communications of the ACM 20 100-106. [Linkout]

Acknowledgements
Thank you for all author contributions and the support of the project.

Funding
This research was funded by the Hubei Science and Technology Major Project - Key R & D Project, grant number 2020AAA004.