RESEARCH OF VECTOR TILE CONSTRUCTION TECHNOLOGY BASED ON APACHE SEDONA

Zhang Hongping[1,2], Du Mingyi[1, *], Huang Wei[2], Ding Lei[2], Tang Dejin[2], Jiang Jie[1]

[1] School of Geomatics and Urban Spatial Information, Beijing University of Civil Engineering and Architecture, Beijing, China – (dumingyi, jiangjie) @ bucea.edu.cn

[2] National Geomatics Center of China, Beijing, China- (zhanghongping, huangwei, dinglei,tangdejin)@ngcc.cn

Commission IV

KEY WORDS: Web map, Vector tile, Pyramid model, Parallel processing, Apache Sedona

ABSTRACT:

With the development of spatial information technology, the amount of geographic information data shows an explosive growth, which puts forward higher demands on the time efficiency and visualization effect of geographic information data release. vector tile maps have become the main map service mode in the fields of Internet maps and GIS industries because of the advantages of its powerful interactive capabilities, efficient data transmission and lower storage costs. In order to construct massive vector tiles quickly and enhance the scalability of vector tiles application, this paper researched the vector tiles construction technology based on the distributed computing framework. Firstly, we introduced the construction process of vector tiles such as pyramid model, data organization and data generalization, Specifically, the data generalization methods of different geometric types were discussed. Second, Apache Sedona, the distributed computing framework were researched and the advantages for vector data processing is introduced. Then, the Sedona based parallel construction technology is proposed, and pipeline optimization of Spark is applied in this process. Third, a comparative experiment to evaluate the performance were conducted, the result showed that the parallel construction technology had obvious performance advantages, the greater the volume and extent of vector data, the greater the advantage.

1. INTRODUCTION

With the development of HTML5 and WebGL technologies, the vector tile service represented by Mapbox has gradually replaced raster tiles and become the mainstream service mode of web maps. Vector tiles are the segmentation and storage of vector data layers in the form of tiles. Compared with raster tiles, vector tiles have the advantages of small volume, high generation efficiency, dynamic interaction, and support for online editing and style modification, and the traditional raster tiles do not have the characteristics (Zhu, X. L., et al., 2016). At present, spatial data with high precision, large coverage, and many layers is exploding. For instance, the national land coverage and geographic national conditions data in a single year can reach TB scale, and the number of geometric objects can even reach more than one billion (Zhang, J., X., et al., 2016). In this context, how to construct vector tiles for map services quickly and efficiently with the massive spatial data has become one of the current research hotspots.

The raster tile is to render vector data into JPG/PNG format according to the pyramid model, and vector tile is to store data in binary format with the Mapbox MVT specification using Google Protocol Buffers (PBF) (Vladimir, et al., 2018). Therefore, the number of data layers, geometric complexity, and the amount of feature nodes have a greater impact on the efficiency of vector tile construction and map rendering. In recent years, researches on vector tile map have been conducted (Chen, J., P.& Ding J., X, 2017). To make the tile size more smaller, an improved Visvalingam algorithm for simplifying Linear Elements was proposed according to the application requirements of the vector tile map service (Jin C., et al., 2019). Usually, the distribution characteristics of spatial data are ignored, resulting in unbalanced data volume among vector tiles at the same level, some studies researched the method for dense and sparse vector tiles construction considering the spatial distribution characteristics of spatial data (Zhu, X., X, et al., 2017). With the rise of Hadoop and Spark, there are many studies on parallel construction technology for spatial data analysis and management (Shin,2021;). Geospark, a cluster computing framework for processing spatial data was introduced, and it is a combination of Spark and geo-information (Yu, J., et al.,2016, Huang Z, et al.,2017). A scalable geospatial data visualization framework (GeoSparkViz) is introduced in the apache spark ecosystem, but it focus on spatial big data visualization rather than vector tiles (Yu,.J., 2018). The parallel computing framework Spark and Hadoop are both used to build the vector tile pyramid model (Nie, P.,2020), however, spatialRDDs are not used in their research.

It can be seen from the above research that how to construct massive vector tiles quickly and efficiently has become the key part of vector data mapping and visualization. In this paper, we present the vector tile construction technology framework based on Apache Sedona. First, the vector tile construction process including pyramid model, data organization and data generalization is introduced. Second, Apache Sedona, the distributed computing framework is researched and the

* Corresponding author

advantages for vector data processing are analysed. Then, the Sedona based parallel construction technology is proposed. Third, a comparative experiment to evaluate its performance is conducted. Finally, we close with some concluding remarks.

2. CONSTRUCTION OF VECTOR TILES

2.1 Tiles construction model

The vector tile refers to a specific data format formed by dividing the original data into blocks according to the predefined tile model. The tile model generally needs to satisfy the user's quick access to data in different areas and different scales. Since the calculation and indexing method of the pyramid model is simple and fast, it is more suitable for high concurrent access scenarios on the Internet. Therefore, in the field of Internet maps, the pyramid level model is generally used to generate map tiles, and this model is also applied in this paper. The pyramid model is a multi-resolution hierarchical data model. Under the condition that the geographic range remains unchanged, from the top to the bottom of the tile pyramid, the scale value becomes smaller and smaller. It divides the map according to the quad-tree rule. As shown in Figure 1, the scale and resolution relationship between adjacent layers is 1:2, the number of tiles is 1:4, and the tile size is consistent at different layers. The size of vector tile refers to the resolution of the tile on the screen, that is, the length and width of the tile, generally according to the square tile of 512 × 512 pixels.

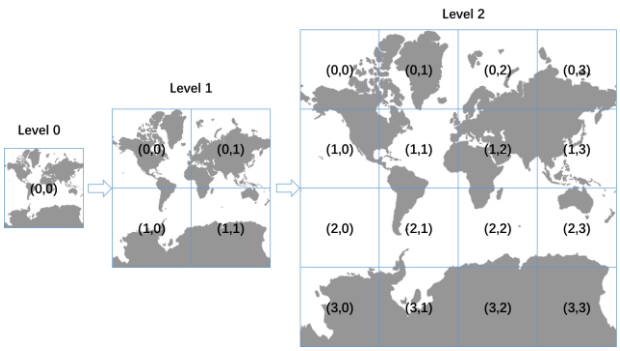

Figure 1. Quardtree spatial index of tiles from level 0 to level 2

As the level increases, the scale of the tile increases, the spatial range represented by a single tile decreases, and the spatial range relationship between adjacent levels of tiles is 1:4. The number of vector tiles at each level can be calculated according to

$$N = \sum_{i=0}^{n} 4^i \qquad (1)$$

where *i* represents the level of the pyramid model.

2.2 Vector tile data organization

Unlike raster tiles that only store image information, vector tiles store both geometric information and attribute information. For the organization of vector tile data, neither ISO nor OGC has issued a unified data standard. Currently, the open source Mapbox Vector Tile Specification (MVT) is the mainstream vector tile data organization file format, which is not only supported by the tools such as Tippecanoe from Mapbox, but also supported by many GIS softwares like GeoServer, mapnik, etc. Google Protocol Buffer（pbf）, a compact binary format for structured data, is used to encode vector tiles. This kind of vector tile format can reduce data redundancy, greatly save storage space, and improve the data transmission efficiency on the Internet. A vector tile does not contain information about its bounds and projection. The file format assumes that the decoder knows the bounds and projection of a vector tile before decoding it.

2.2.1 Layers: Like GeoJson, a vector tile consists of a set of named layers, such as POIs, road, water, etc, and each layer contains geometric features and their metadata. A vector tile should contain at least one layer, and a layer should contain at least one feature. The attributes of each feature are stored with one or more key-value pairs, which can be used for map styles and attribute query. In order to improve front-end rendering efficiency and reduce tile size, the vector data from the original projected coordinate system are converted to the screen coordinate system, and the float coordinates are converted to integer screen coordinates. The coordinate system of the vector tile is the screen coordinate system, the upper left corner of the tile is the origin of the coordinate system, the direction of the x-axis is positive to the right, and the direction of the y-axis is positive downward. An *extent* field should be included to describe the width and height of the tile. In this paper, we expanded the vector data of 0.5% of each tile *extent* as a buffer for rendering features that overlap multiple adjacent tiles.

2.2.2 Features: Features are used to describe the geometric object with geometry and type fields. Each feature should contain at least one type, and the type field must be a value in the enumerable types, such as Point, Linestring, Polygon, and Unknown. The tag and id fields are optional, if there are attributes for the feature, the data should be saved in the tag fields, and the values of id field should be unique among the other features in the layer.

```
{
  "type":
"FeatureCollection",
  "features": [
    {
      "geometry": {
        "type": "Point",
        "coordinates":    [-
8247861.1000836585,49702
41.327215323
          ]
      },
      "type": "Feature",
      "properties": {
        "aa": "world",
        "bb": "world",
        "cc": 2
      }
    }]
}
```

(a)GeoJson struct

```
layers {
 version: 2
 name: "points"
 features: {
  id: 1
  tags: 0
  tags: 0
  tags: 1
  tags: 0
  tags: 2
  tags: 1
  type: Point
  geometry: 9
  geometry: 2410
  geometry: 3080
 }
 keys: "aa"
 keys: "bb"
 keys: "cc"
 values: {
  string_value: "world"
 }
 values: {
  double_value: 2
 }
extent: 4096
}
```

(b) MVT struct

Figure 2. Examples of GeoJson and MVT struct

Figure 2 shows a geojson example was converted to the mvt. The name of the layer is points, and there is only one geometric object of Point type. The coordinates, type and attribute index values are stored in the feature. The extent of the tile only specifies the range of the tile, not the size of the tile after rendering. For example, the mvt with an extent of 4096 does not mean that the final rendered image is an image of 4096 × 4096 pixels. The size of the rendered image is not determined by the extent.

2.2.3 Metadata: Due to the vector tiles are rendered on the browser, the metadata such as format, bounds, layers and should be provided to the JavaScript API firstly. On the basic of Mapbox metadata, we expanded some other fields like producer and release date according the actual production. All the fields considered for vector tiles in this paper are in Table 1:

Field	Description
name	name of the vector tile dataset
type	map type, such as basic map or thematic layers
version	version of vector tiles
projection	vector tiles projection, such as EPSG:3857
tilesize	size of vector tiles, such as 512 pixels
minlevel	the min level of the vector tiles
maxlevel	the max level of the vector tiles
center	the center point of the vector layers
format	vector tile format, such as geojson, mvt or customed pbf format
bounds	the max Bbox of the vector dataset, all the tiles of different levels are included
json	layers, layer id, layer zoom levels, descriptions, etc, in the format of json
url	the url to access vector tiles
producer	vector tiles production personnel or department
release_date	release date of vector tiles
description	descriptions of the vector tiles

Table 1. Metadata for vector tiles

2.3 Vector data generalization

The size of each vector tile determines the transfer efficiency on the Internet and rendering efficiency on the canvas. Therefore, the generalization of vector data is the most important part in the process of vector tiles construction. In fact, the problem of vector data generalization has always been a difficult problem in the field of map synthesis, and it has not been well solved so far. In this paper, we combined attribute filtering and feature simplification for different kinds of layers to reduce the tile since and improve rendering efficiency.

For the Attribute filtering, vector tiles are multi-scale and do not render all vector features at once, and all the features are displayed according to the cartographic synthesis theory and pyramid model. In the case of road layer, the expressway is displayed from level 6, national and provincial highway is displayed from level 7, small and dense urban and rural roads are displayed from level 16. The attributes filtering conditions are set before generating the tiles, then each level only contains the needed features that should be displayed, and the others are not included in the tiles, which can greatly reduce the number of features contained in low-level tiles and improve the rendering efficiency of tiles.

As to the POIs, the points of each tile should be filtered according to the level and classification firstly, if the number of POIs does not exceed the threshold, the generalization process will not be performed, otherwise the POIs will be simplified based on the grids of each tiles. Only one POI is reserved for all categories in each grid, and if the category and importance of POIs in a grid are the same, the random deletion method is adopted.

As shown in Figure1, the Douglas–Peucker algorithm that takes into account topological relationships is used to simply the line and polygon features. The overall process is as follows: 1) reading all the features of the same layer; 2) extracting the adjacent edges; 3) using two object lists to save independent edges and adjacent edges, and establishing the relationship between them; 4) simplifying the edges separately, and the endpoints should not be removed; 5) reorganizing the edges and forming the feature.

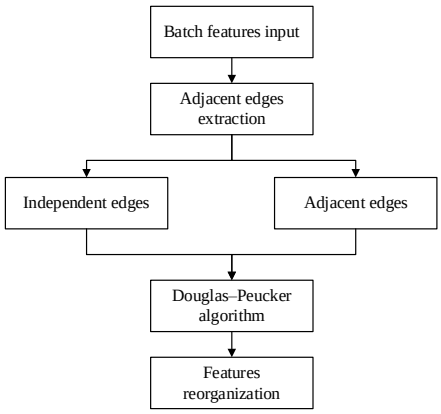

Figure 3. Vector data generalization method

3. SEDONA BASED PARALLEL CONSTRUCTION TECHNOLOGY

3.1 Apache Sedona framework

Apache Sedona (incubating) is a cluster computing system for processing large-scale spatial data. Sedona extends Apache Spark / SparkSQL with a set of out-of-the-box Spatial Resilient Distributed Datasets / SpatialSQL that efficiently load, process, and analyze large-scale spatial data across machines (Apache Sedona, 2021). The predecessor of Senona is geospark, it is the combination of GIS and Spark. RDD (Resilient Distributed Dataset), which is called a distributed dataset, is the most basic data abstraction in Spark. It extends RDDs to form spatial RDDs (SRDDs) and efficiently partitions SRDD data elements across machines and introduces parallelization of spatial transformations and operations to provide a more intuitive interface for users to write spatial data analysis programs. Figure 4 shows the architecture of Apache Sedona, which contains 3 layers (Apache Sedona, 2021).

3.1.1 Spatial RDD layer: It supports various spatial data input formats, such as text and wkt text. The data is transformed and processed and stored in SRDDs, and the JTS topology suite is integrated to support spatial objects. Depending on the type of spatial object, spatial RDDs are defined as PointRDD,

RectangleRDD, PolygonRDD and LineString RDD. Some geometry operations for SRDDs are built-in, and hence geometric operations can interact with Spark Layer through RDD operators such as Map, Sort, Filter, and Reduce.

,

3.1.2 Spatial Query Processing Layer: Based on the Spatial RDDs Layer, it supports spatial queries (such as range queries and join queries) of large-scale spatial datasets. Users can call the spatial query provided by the spatial query processing layer after the geometric objects are stored and processed in the SRDD layer, and all the process is in the memory.

3.1.3 Spatial SQL/Python API Layer: SQL interface follows SQL/MM Part3 Spatial SQL and OGC Standards. All SedonaSQL functions (list depends on SedonaSQL version) are available in Python API.

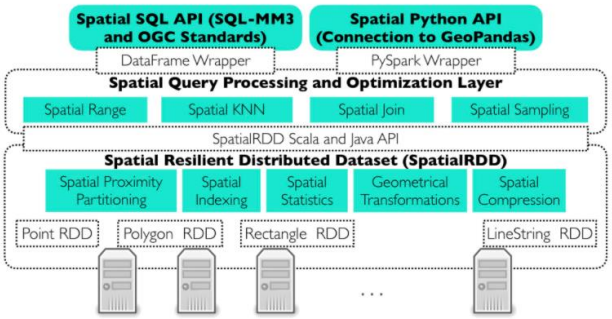

Figure 4. Architecture of Apache Sedona

3.2 Vector tiles parallel construction process

The original vector layers with necessary mapping attributes are stored in the PostgreSQL according to the different categories, such as POIs, roads, water, buildings and boundary. Figure 5 shows the overall parallel construction method adopted in this paper.

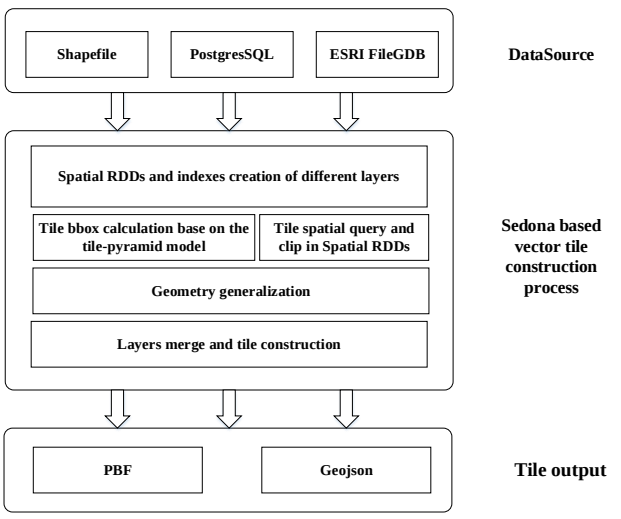

Figure 5. Sedona based vector tile construction process

3.2.1 Data preprocessing: Before importing the data though SparkSQL, the problems such as coordinate system, self-intersection and multicurve should be preprocessed. Due to the size of different features is not equal, for example, the size of some water and vegetation features are nearly 100M, and most of the other features are less than 1M, this will result in data skew while parallel construction in Sedona. The size of the features is determined by the points, therefore, we split the features with the points more than 10 thousand to keep features balancing.

3.2.2 Data import: Use SparkSQL to read the data of specified extent in the PostgreSQL, for example, the Bbox of some administrative division. There is a geom field in PostGIS, which stores the geometric information in wkb format. For sparksql, the geom field is a varchar field and it is needed to convert it to Geometry. And then all the spatial features are converted into SRDDs in memory before parallel processing.

3.2.3 Tiles calculation: According to the pyramid model and the needed levels of tiles construction, the Bbox of each tile can be calculate by spatial query operation. Then clip the geometry objects in the SRDDs based on the tile's Bbox, and the tile's buffer is considered to prevent cracks between adjacent tiles. Due to the uneven spatial distribution of vector data, there are a lot of invalid queries and intersection calculations based on in the traditional tile construction technology, the problems are very obvious in the west province such as Tibet and Xinjiang. For instance, as shown in Figure 6, the polyline only crosses the grid 0,1 and 2. If we need to produce the vector tiles from level 1 to 18, there is no need to do any operation from level 2 to 18 in grid 3. By that analogy, grids 00,02,11 and13 are the same in the next level process. We used the pipeline optimization of Spark to solve the problem.

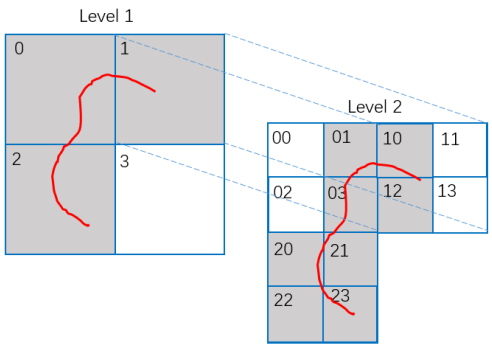

Figure 6. Tile construction process after optimization

3.2.4 Geometry object identification and simplification: Each geometry is divided into different tiles and tagged with (row, column, level), and then the generalization algorithm introduced in 2.3 is used to simply the point, line and polygon features of different layers and levels based on the threshold values. While clipping a geometric object, all the related attributes such as type, category, and some other values are reserved.

3.2.5 Geometric objects merge: The coordinates of geometric objects should be transferred to screen coordinates, and the tile data is formed by merging all geometric objects in the same tile tag (row, column and level), and the tile of PBF format can be produced and stored in the MongoDB database.

4. EXPERIMENT

This section uses a real application case to validate the construction method proposed in this paper.

4.1 Experimental data

As shown in Figure 7, the Beijing dataset from the National geographic information service platform (Tianditu) is choosed as the experimental case, which is located at 115.7 °-117.4 °E and 39.4 °-41.6 °N.

Figure 7. Experimental data and extent

The experimental data set includes multiple layers such as POIs, water, rail, road, buildings, vegetation, area of interest (AOI), annotation of AOI, etc. The size of the dataset is about 1GB, layers, number of features, nodes are shown in Table 2.

No.	Layer	Type	Number of Features	Nodes
1	POIs	Point	94,169	94,169
2	Water	Polygon	43,098	807,614,448
3	Water	Multiline	14,165	3,150,299
4	Rail	Multiline	955	39,166
5	Road	Multiline	1,057,013	5,809,320
6	Buildings	Polygon	5,635,766	40,038,553
7	Vegetation	Polygon	351,876	15,700,962
8	Area of Interest (AOI)	Polygon	6,068	141,326
9	Annotation of AOI	Point	604	604

Table 2. Layers and features of the experimental dataset

4.2 Comparison of vector tiles construction

The experiment compares the standalone and Sedona mode for constructing vector tiles based on Beijing's dataset. Geoserver 2.20.3 (hereinafter called Geoserver) and PostgreSQL12 + Postgis2.2 (hereinafter called Postgis) are selected as the standalone mode experiment softwares because they are both open source and many commercial GIS platforms which support vector tiles are developed based on them. The test server is a virtual machine with 48vCPU, 96GB memory, and 500GB storage space. The Sedona mode is used with 96 parallelisms to test the performance.

As shown in Figure 7, the execution time of Sedona only costed 22 minutes, but the time of geoserver and Postgis spend are both more than one hour. From the results of the experiment, the execution time is less and the algorithm efficiency is higher by using the construction method presented in this paper. As the volume of data and the map extent increase, the advantages of Sedona mode will be more significant.

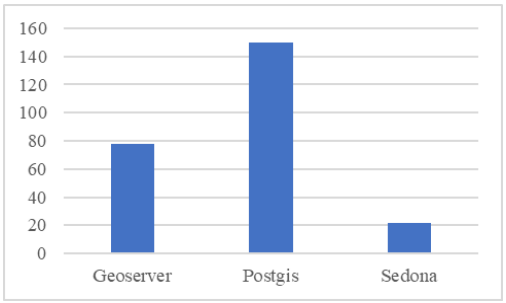

Figure 7. Total Execution Time of different construction modes (in minute)

5. CONCLUSION

Vector tiles have become the mainstream service mode of the current Internet map, and the construction efficiency of vector tiles is an important factor affecting the update of vector data. This paper proposed the vector tiles construction technology based on Sedona. The parallel tile construction process and the pipeline optimization method of Sedona are introduced. We can draw the following several conclusions:(1) The parallel vector tile construction based on Sedona in this paper is significantly faster than the traditional standalone mode. Comparative experiments show that the execution time of the parallel construction algorithm reduced about 85% than PostGIS. (2) It has more efficiency in large scale vector data, especially in the area of uneven spatial distribution. (3) It is feasible to construct vector tiles based on the high-performance distributed computing framework, and it has great advantages in the fields of large-scale vector data visualization and mapping applications.

Next, further reduction of the vector tile size and improvement of the data security will be our next research emphases. In addition, we hope to extend the framework to support the raster tile construction with the same source data.

ACKNOWLEDGMENT

This work was supported by National Geo-information Service Platform "Tianditu" and Technology innovation center for Geo-information common service.

REFERENCES

Apache Sedona, 2021. https://sedona.apache.org/

Chen, J., P., Ding J., X, 2017. Research on key technology of vector tile map. Geospatial Information, 15(8):44-47.

Huang, Z., Chen, Y., Wan, L., & Peng, X.., 2017. GeoSpark SQL: An effective framework enabling spatial queries on spark. ISPRS International Journal of Geo-Information, 6(9), 285.

Jin, C., An.,X.,Y., Cui, H., F.,& Zhao., Y., J., 2019 An Algorithm for Simplifying Linear Elements of Vector Tile Maps. Journal of Geo-Information Science, 21(10):1502-1509.

Nie, P., Chen, G., S. & Jing W., P., 2020. Parallel construction and distributed storage for vector tile. Journal of Geo-information Science, ,22(7):1487-1496.

Shin, H., Lee, K., & Kwon, H. Y., 2021. A comparative experimental study of distributed storage engines for big spatial data processing using GeoSpark. The Journal of Supercomputing, 1-24.

Vladimir A, John F, Eric F, et al. Mapbox vector tile specification[EB/OL]. https://github.com/mapbox/vector- tile-spec, 2018-05.

Yu, J., Wu, J., & Sarwat, M., 2016. A demonstration of GeoSpark: A cluster computing framework for processing big spatial data. In 2016 IEEE 32nd International Conference on Data Engineering (ICDE) (pp. 1410-1413). IEEE.

Yu, J., Zhang, Z., & Sarwat, M., 2018, July. Geosparkviz: a scalable geospatial data visualization framework in the apache spark ecosystem. In Proceedings of the 30th International Conference on Scientific and Statistical Database Management (pp. 1-12).

Zhang, J., X., Gu, H., Y., Lu, X., J., & Hou, W., 2016. Research framework of geographical conditions and big data. Journal of Remote Sensing, 20(5), 1017-1026

Zhu, X., L., Zhou, Z. W., Li, J., Zhao, Y., &Peng Y., L., 2016. Research for Web Map Vector Tiles Technology. Bulletin of Surveying and Mapping,11, 106-109.

Zhu, X., X., Zhang, F., & Du., Z., H., 2017. A method of the dense-sparse vector tile generation accounting for the spatial distribution of features. Journal of Zhejiang University(Science Edition), 44(5):591-598.