Hindawi
Complexity
Volume 2022, Article ID 6860915, 9 pages
https://doi.org/10.1155/2022/6860915

Research Article

Quick Compression and Transmission of Meteorological Big Data in Complicated Visualization Systems

He-Ping Yang,[1] Ying-Rui Sun,[1] Nan Chen,[1] Xiao-Wei Jiang,[1] Jing-Hua Chen,[1] Ming Yang,[2] Qi Wang,[1] Zi-Mo Huo,[1] and Ming-Nong Feng[1]

[1]*National Meteorological Data Center, Beijing, China*
[2]*Zhejiang Meteorological Information and Network Center, Zhejiang Meteorological Bureau, Hangzhou, China*

Correspondence should be addressed to Ying-Rui Sun; sunyr@cma.gov.cn

Received 24 November 2021; Revised 12 January 2022; Accepted 12 March 2022; Published 5 May 2022

Academic Editor: Ning Cai

The sizes of individual data files have steadily increased along with rising demand for customized services, leading to issues such as low efficiency of web-based geographical information system (WebGIS)-based data compression, transmission, and rendering for rich Internet applications (RIAs) in complicated visualization systems. In this article, a WebGIS-based technical solution for the efficient transmission and visualization of meteorological big data is proposed. Based on open-source technology such as HTML5 and Mapbox GL, the proposed scheme considers distributed data compression and transmission on the server side as well as distributed requests and page rendering on the browser side. A high-low 8-bit compression method is developed for compressing a 100 megabyte (MB) file into a megabyte-scale file, with a compression ratio of approximately 90%, and the recovered data are accurate to two decimal places. Another part of the scheme combines pyramid tile cutting, concurrent domain name request processing, and texture rendering. Experimental results indicate that with this scheme, grid files of up to 100 MB can be transferred and displayed in milliseconds, and multiterminal service applications can be supported by building a grid data visualization mode for big data and technology centers, which may serve as a reference for other industries.

1. Introduction

Currently, the development of information collection and storage technology has ushered in the era of big data in various industries, as the amounts of data being recorded, processed, and analyzed have exploded. In particular, meteorological data are among the most important types of data encountered in people's daily lives, playing an essential role in understanding the environment, natural resources, the economy, and other aspects of life [1]. To address society's need for refined meteorological data, grid data products for observations and predictions based on radar data, satellite data, and station observations have been extensively utilized [2]. On a global scale, the available meteorological grid data mainly include numerical forecasting products from the European Centre for Medium-Range Weather Forecasts (ECMWF) [3], the National Centers for Environmental Prediction (NCEP) Global Forecast System (GFS) model [4], the Global Regional Assimilation and PreEdiction System Global Forecast System (GRAPES_GFS) model of the China Meteorological Association (CMA) [5], and the real-time High-Resolution CMA Land Data Assimilation System (HRCLDAS) [6]. The most common storage formats for such grid data products are General Regularly-distributed Information in Binary form (GRIB) and Network Common Data Form (NetCDF). The former is a file format that was designed by the World Meteorological Organization (WMO) for storing and transmitting meteorological grid data, such as the outputs of numerical weather prediction models; this format is concise enough to be widely used in meteorology to store historical and forecast weather data [7]. The latter is an array-oriented and network sharing-based data description and coding standard proposed by scientists of the Unidata project at the University Corporation for Atmospheric Research (UCAR) [8].

Files in these commonly utilized grid data formats can only be opened by professional applications (apps) and can be used to obtain values at specific locations or to analyze the spatial distributions of variables such as temperature or rainfall [9]. For meteorological big data, which tend to have strong geographical spatial characteristics and location correlations, geographical information system (GIS) technology is usually combined with tools for visual expression and application provided by common services [10]. Panoply from the National Aeronautics and Space Administration (NASA) Goddard Institute for Space Studies, which is a viewing tool rather than a data extraction tool that supports multiple formats such as GRIB, requires the Java Runtime Environment [11]. MeteoInfo (i.e., MeteoInfoMap and MeteoInfoLab), which was developed by the Chinese Academy of Meteorological Sciences, is an integrated framework for both GIS applications and scientific computation environments that is utilized by the meteorological community to visualize and analyze spatial and meteorological big data in multiple data formats [12]. As the centerpiece of National Weather Service operations in China, the Meteorological Information Comprehensive Analysis and Process System (MICAPS) is a complicated computer system that combines meteorological, satellite, and radar data into one workstation and allows graphical and alphanumeric weather data in GRIB format to be read, analyze, combined, and manipulated [13].

All of the above methods have been used to display local meteorological files. However, with the emergence and popularization of cloud storage, new types of applications have arisen in which computing resources are no longer localized but rather distributed, heterogeneous, and dynamic. The Grid Analysis and Display System (GrADS) is a widely used drawing software tool in meteorology. It has two main functions, namely, data processing and image display, and plays a role in the meteorological research community similar to that of the World Wide Web in facilitating information exchange over the Internet. GrADS has unique characteristics, mainly for scientific research and business personnel involved in atmospheric and marine research. With its powerful data analysis capabilities, flexible environment setup, wide range of mapping types and variety of map projection methods, GrADS has greatly aided meteorological research [14]. The Integrated Data Viewer (IDV) from Unidata/UCAR is a Java-based software platform for analyzing and visualizing geoscience data [15]. The IDV combines the abilities to display and analyze satellite imagery, gridded data (such as numerical weather prediction model outputs), GIS data, and other data in a single interface [16]. It has been integrated with common scientific data servers, including Unidata's THematic Realtime Environmental Data Distributed Services (THREDDS) Data Server (TDS) [17], as data sources to enable easy access to a large number of real-time and archival datasets. The IDV is the main tool used in the computer laboratory portion of various meteorological courses at colleges and universities.

These localized applications can read files directly from a local disk, efficiently download them to a local disk, or integrate them with common scientific data servers. However, the local installation process has relatively high system requirements and is therefore not suitable for public services.

As mobile terminal apps such as Wireless Application Protocol (WAP) browsers and WeChat have been developed for IOS and Android operating systems, rich Internet applications (RIAs), which are web-based applications designed to deliver the same features and functions normally associated with desktop applications, have become essential platforms that can run in web browsers without installation. One of the earliest attempts to make RIAs accessible was to use the World Wide Web Consortium (W3C) standard for Accessible Rich Internet Applications (ARIA) [18]. This technology has been used to visualize meteorological big data with web-based GIS (WebGIS) tools. Rain Viewer, an all-in-one weather radar and rain forecast app for predicting storm tracks, is available for 90 countries and offers the most comprehensive weather radar coverage on the market, displaying a single map with data from 1000+ Doppler radars with the option of viewing information about each radar on the map. To provide this functionality, all requests to Rain Viewer are routed through an online web service that overlays the Rain Viewer data on a map tile with 256 or 512 pixels centered on the user's current location and then resizes the image to match the screen size of the device [19]. In the WebGIS service OpenStreetMap, the layer overlay is displayed within milliseconds by using leaflet technology and considering the aging of single-slice requests. Thus, this service cannot meet the requirements of real-time interactions, such as changing the color range or filtering by value. Moreover, the layer is virtualized, and the user experience can be poor if the maximum image resolution is exceeded after the image is enlarged. Advancements in vector tile technology have offered solutions to the above problems, in which vector tile layers are saved as compressed files in the Protocolbuffer Binary Format (PBF) file format [20]. Such a compressed file, which contains vector map data in one or more layers, can be rendered and styled based on the style of each layer. The data in a vector tile include geographic features in the forms of points, lines, and polygons [21]. For weather radar, this solution goes beyond simply resampling the data and aims to generate vector contours based on the raw radar data. With the data in the form of vector polygons, the AerisWeather Mapping Platform (AMP) can render radar data at any zoom level without reducing the resolution or quality. This also allows the user to control how much smoothing is applied to the radar data, enabling clean and smooth radar imaging at the city and neighborhood levels [22].

Meteorological big data vary spatially and temporally, and any dynamic vector slicing scheme must include data preparation, slicing, and front-end visualization, with high-performance requirements for the server hosting the spatial database (PostGIS) [23]. Hence, complex visualization technology for meteorological big data is gradually developing in the direction of data file compression prior to transmission, followed by foreground decoding. The Null school designed a global visual display system (Nullschool.net) [24] for ECMWF forecast data, which converts a map

from the Natural Earth dataset into the TopoJSON format to serve as the base map, utilizes the EPAK format to transform and compress the grid data, and applies Node.js to rapidly render and display the foreground [25]. Since the launch of this system, the Tokyo Meteorological Bureau and other institutions have developed the Tokyo Wind Map based on this technical framework, which has been extensively promoted. Although this framework makes full use of the advantages of rapid visualization on a web terminal, there are still incompatibility problems on mobile terminals for base maps with coastline contour resolutions of 50 km or 110 km in the JSON format; hence, it is difficult to use this framework for fine-scale service applications. Lytvyn et al. [26] and Prastika et al. [27] have implemented multichannel support applications such as PC browser, WAP, and mobile applications based on the OpenStreetMap online map by integrating slices with data compression and transmitting only single slices within 30 kB for faster transmission and visualization. For data visualization based on the browser/server (B/S) architecture, the minimum resolution of meteorological grid data reaches more than 9 km worldwide, and the file size for a single transmission is between 700 kB and 2 MB. A single request can reach a second-level response, and dynamic rendering allows updating, which satisfies the requirements for large-scale services. However, various disaster prevention and emergency mitigation support applications need more refined grid data services. In 2021, the CMA released the HRCLDAS product [28], which covers East Asia with a resolution of 1 km. The grid size is $7000 * 4500$, and the data volume of a single file reaches 106 MB. The main applications for visualizing such data superimpose a transparent image directly on top of the map and reduce the zoom or resolution layer by layer, resulting in a loss of eigenvalues (e.g., the maximum and minimum values) and thus affecting the front-end rendering results. Therefore, it is extremely difficult to balance efficiency and data accuracy in page rendering based on the B/S architecture.

In conclusion, visualization systems based on the B/S architecture are the most user-friendly option; however, their compression and transmission methods have become increasingly complicated as the demand for refined services and individual file sizes have increased. Compression is a highly efficient method for files smaller than 10 MB, but the visualization process achieves better transmission and rendering pressure when the file size is approximately 100 MB or larger, and 100 MB is a common file size for refined meteorological services. Therefore, there is a need to design a fast transmission and display scheme for meteorological grid data on PCs, mobile browsers, WAP apps, WeChat applets, and other platforms based on an open-source WebGIS service platform.

In this article, we propose a customized scheme for use in complicated visualization systems for meteorological big data. At the back end, a high-low 8-bit compression algorithm is adopted, and customized slice transmission is required to ensure high network transmission efficiency. Based on the HTML5 and Vue frameworks, the front-end uses Mapbox GL [29, 30] technology to satisfy the demands of dynamic meteorological big data visualization services while considering compression, slicing, display, and other factors. The proposed scheme offers a mid-platform support mode for visualizing meteorological grid data that have been published in the meteorological visualization column of the China Meteorological Data Service Center, via a mobile app, or on WeChat. This scheme provides a fast and convenient solution for rapidly visualizing and rendering grid data and can serve as a reference for grid data visualization applications in other industries.

2. Methodology

The proposed system includes big data processing, transmission, and page rendering and involves technologies such as data analysis, compression, browser transmission, page data restoration and splicing, as well as WebGL [31] rendering. The detailed design is shown in Figure 1.

Data processing and compression: Red-green-blue (RGB) channels are used to compress and store the data in high-low 8-bit PNG files to maintain data accuracy to the fullest extent possible during the transmission process.

Data slicing and transmission: Based on pyramid slicing technology, slicing is performed with a specified minimum scaling resolution, and distributed multithreading is used during transmission to increase the timeliness of transmitting the data from the server to the page terminal.

Data visualization rendering: The browser obtains slices and performs slice stitching, while the Mapbox GL component based on WebGL technology realizes fast dynamic rendering.

2.1. Data Processing and Compression. First, the source GRIB file is transformed into a float array by PYGRID [32] in Python and is generally stored as 4-byte data with a maximum legend display resolution of 0.1. The data can be retained to 1 significant digit before being stored, that is, the original value O is multiplied by 10 and rounded to obtain the integer C. The image is saved as an RGB compressed image with 2 bytes in the G and B channels in the range of [−32768, 32767]. To minimize data loss, the image is stored in the PNG format by adopting an LZ77-derived algorithm for file compression, thus resulting in a small data volume, a high compression ratio, and no data loss.

$$O = \begin{bmatrix} I_{11} & \cdots & I_{1W} \\ \vdots & \ddots & \vdots \\ I_{H1} & \cdots & I_{HW} \end{bmatrix}; \quad C = \text{round}(O \times 10). \tag{1}$$

The high 8-bit and low 8-bit values of the converted data C are stored in the G and B channels, respectively. The specific operations are shown as follows:

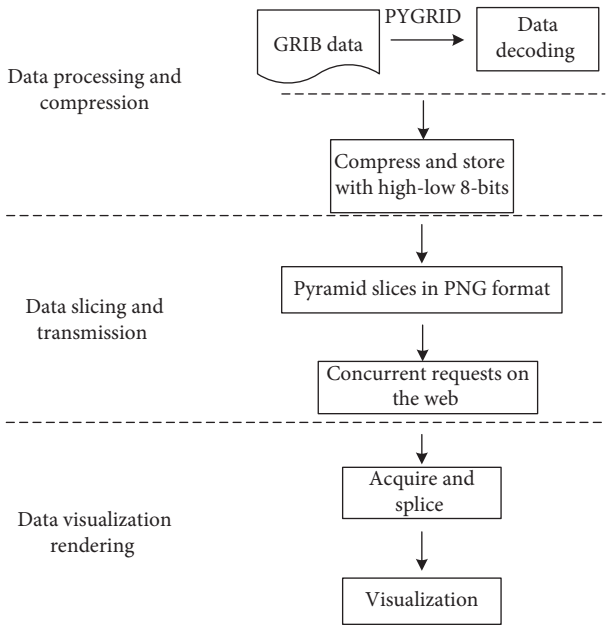

Figure 1: Design of the system.

$$U = \text{trunc}\left(\frac{C}{256}\right);$$

$$D = C\%256,$$

$$G\_N = \begin{bmatrix} [R_{11}, G_{11}, B_{11}] & \cdots & [R_{1W}, G_{1W}, B_{1W}] \\ \vdots & \vdots & \vdots \\ [R_{H1}, G_{H1}, B_{H1}] & \cdots & [R_{HW}, G_{HW}, B_{HW}] \end{bmatrix}, \quad (2)$$

$$G\_N[:][:][:, :, 1] = U;$$

$$G\_N[:][:][:, :, 2] = D.$$

where U and D represent the intermediate values, while G_N denotes the newly created PNG image used to store the corresponding compressed values.

To improve the efficiency of page data transmission, the data are processed in accordance with the image pyramid model. The compressed PNG file is scaled down using the image pyramid approach with bicubic interpolation. The tilemap pyramid model is a multiresolution hierarchical model. The resolution decreases from the bottom to the top of the tile pyramid; however, the geographical range of the representation remains constant. The image is first scaled and then filled with squares in accordance with the tilemap pyramid model.

2.2. Data Slicing and Transmission. The original image serves as layer 0 of the pyramid and is scaled by 2×, 4×, 8×, and 16× via bicubic interpolation [33]. In numerical analysis, bicubic interpolation is the most commonly applied interpolation method in two-dimensional space.

It is assumed that if the source image G has a size of $M \times N$ and the scaled target image g has a size of $m \times n$, the coordinates of g on G can be calculated using formula (3).

$$x\prime = x \times \frac{M}{m},$$

$$G\_N[:][:][:, :, 1] = U; \quad (3)$$

$$G\_N[:][:][:, :, 2] = D.$$

As shown in Figure 2, (x', y') denotes the location of a point P' in the original image, which corresponds to the position $g(x, y)$ in the compressed image; the value at this point is obtained by interpolating from the pixel values at the 16 neighborhood points (P00, . . ., P33). If the position of P11 is (x, y), then the position of P' can be expressed as $(x + u, y + v)$, where u and v represent the fractional parts of the pixel coordinates.

Once the influence weights of the 16 neighborhood points relative to point $P^{'}$ have been calculated, the value of $P^{'}$ can be obtained and mapped to the scaled image g. The basic function for bicubic interpolation is shown in Formula (2), where $a = -0.5$:

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^3 + 1, & \text{for } |x| \le 1, \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a, & \text{for } 1 < |x| < 2, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

When the rows and columns are separated, the distance between the pixel value to be calculated and the known pixel value P00 in Figure 2 can be expressed as $(1 + u, 1 + v)$; hence, the abscissa-coordinate weight of the P00 pair is $W(1 + U)$ and the ordinate weight is $W(1 + V)$, yielding a corresponding value contribution of $\text{Pix}_{00} \times W(1 + u) \times W(1 + v)$. The contributions from the other 15 points can be calculated similarly. Finally, the pixel value at the point in the map scaled to the image G can be calculated using formula (5).

$$G(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} \text{Pix}_{ij} \times W(i) \times W(j). \quad (5)$$

Notably, a browser can process only a limited number of concurrent requests for the same domain name, which restricts the number of simultaneous requests that can be served during page rendering, resulting in requests queuing or timing out. This process occurs on GIS service websites such as Google and Baidu Maps, which add subdomains and domain dashes to increase the number of concurrent requests that can be served [34]. However, given the increased difficulty of DNS resolution for an excessive number of domain names, the concurrency of each secondary domain name should be limited to 2–4.

The scheme proposed here, which is based on a B/S service framework, uses an Nginx server, which is a lightweight Web server/reverse proxy server, and an e-mail (IMAP/POP3) proxy server distributed under a BSD-like

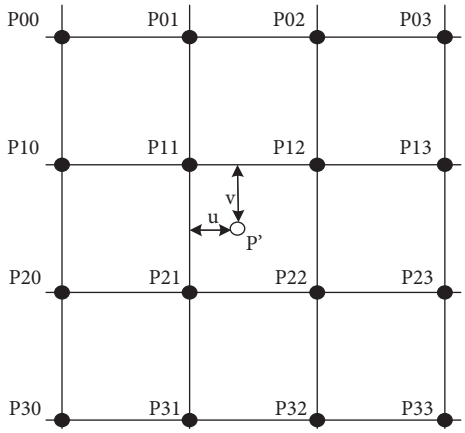

Figure 2: Bicubic interpolation diagram.

protocol, as proxies, thus occupying less memory and enabling high concurrency. Moreover, Nginx supports the gzip compression function, which can be used to compress the website CSS, JS, XML, and HTML files during transmission, thus boosting the access speed and optimizing Nginx performance.

2.3. Data Visualization Rendering. During the rendering process, the number of slices should be adjusted to account for different screen resolutions and map magnifications. The coordinates of the map are translated into screen coordinates to realize the transformation between the longitude and latitude coordinates of the map and the screen coordinates. Once the scale problem has been resolved, the spatial information of the latitude and longitude ranges on the earth corresponding to the GRIB file is transformed to the range of the screen. Based on the size of each layer of the image pyramid in the GRIB file, the pyramid slice that offers the best rendering is finally selected (Figure 3).

The distance between any two points expressed in terms of longitude and latitude can be calculated using the following flow formula:

$$\text{Distance}((\text{lons}, \text{lats}), (\text{lone}, \text{late})) = R * \cos^{-1}\left[\begin{array}{c}\cos(\text{lats}) * \cos(\text{late}) * \cos(\text{lons} - \text{lone}) \\ +\sin(\text{lats}) * \sin(\text{late})\end{array}\right], \tag{6}$$

where R is the radius of the earth, which is approximately equal to 6371.0 km.

The corresponding screen distance can be calculated from the Mapbox scale.

$$\text{screen}D = \text{Distance}((\text{lons}, \text{lats}), (\text{lone}, \text{late})) * \text{scale}. \tag{7}$$

Finally, the layer number of the selected slices can be determined based on the diagonal width of each layer image in the pyramid.

Here, orgSZ denotes the diagonal image size in the compressed resource file (Figure 4). The result of dividing "orgSZ" by "screenD" indicates how many times larger the visible range on the screen is than the size of the compressed file (layer 0). Because consecutive layers differ in size by a factor of 2, the indicated layer is determined by rounding up the value to a power of 2.

$$\text{Pyramid layer} = \sqrt[2]{\frac{\text{org}SZ}{\text{screen}D}}. \tag{8}$$

Thereafter, the latitude and longitude ranges on the screen are obtained, namely, the upper left (lu_x, lu_y) and lower right corner (rd_x, rd_y), and the tile index is obtained in accordance with the flow progress.

$$\begin{aligned} \text{dis}x &= \text{tile}_x - \text{lu}_x, \\ \text{col}_{\text{start}} &= \text{ceil}\left(\frac{\text{dis}x}{\text{interval}}\right), \quad \text{if dis}x > 0 \text{ else} = 0, \\ \text{col}_{\text{end}} &= \text{col}_{\text{start}} + \text{ceil}\left(\frac{rd_x - lu_x}{\text{interval}}\right), \end{aligned} \tag{9}$$

where tile_x is the upper left corner of the data in the x direction, dis$x$ denotes the distance between the upper left corner of the screen and the upper left corner of the data in the x direction, $\text{col}_{\text{start}}$ represents the index of the starting tile column, col_{end} represents the index of the ending tile row, interval denotes the length and width of a tile, and ceil denotes the operation of rounding up. Similarly, the row and column indices of the starting and ending tiles can be obtained. Because the tile coordinate range is greater than the screen coordinate range, all tiles need to be offset. Starting with the upper left corner of the screen, the position offsets in the CSS file can be obtained by calculating the difference between the pixel coordinates of the upper left corner of each tile and those of the upper left corner of the screen [35].

Based on the Vue development framework, the proposed scheme comprehensively considers the spatiotemporal attributes of meteorological big data and the demands of fast rendering, utilizes Mapbox to support geographical information services, and applies WebGL high-performance front-end rendering technology for data visualization. Vue is a progressive and high-performance JavaScript framework for front-end page display that uses view layer rendering as its core. Vue utilizes a component mechanism, a routing mechanism, and a state management mechanism to quickly realize front-end high-frequency Document Object Model (DOM) operations and efficient page interactions. Due to the use of the NodeJS service and the utilization of Node Package Manager (NPM) to install the Vue command-line interface (Vue CLI), the framework can be built quickly. Mapbox, which has corresponding GIS engines for different platforms (e.g., PC and mobile), is an efficient WebGIS development framework. As a Mapbox component [36],

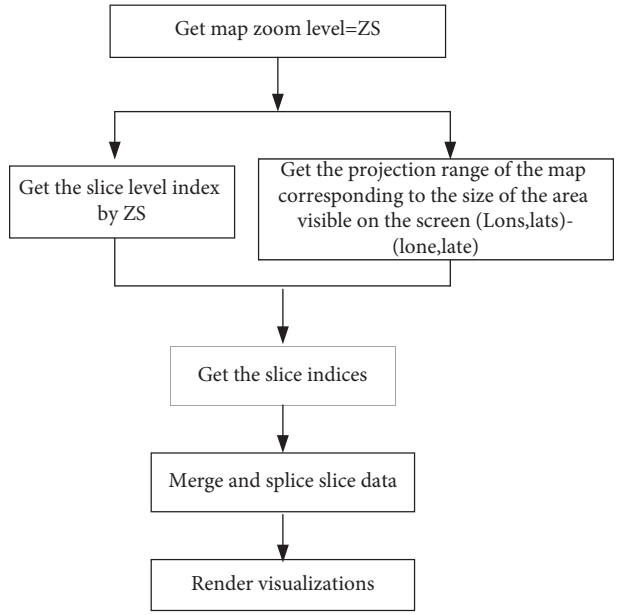

Figure 3: The rendering process.

Mapbox GL has been used for HTML5 web development. Mapbox GL is a JavaScript library that can render a large number of map elements while allowing for smooth interactions and animation.

3. Experiments and Analysis

To examine the performance and effectiveness of the whole proposed process, high-low 8-bit compression was developed in Python 3.7, and the grid data (GRIB2) were decoded into arrays by PYGRID. The web page was developed in Visual Studio XCODE, and the web server was deployed using Nginx 1.16.0. The progress server and web server were 64-bit Linux servers, and the CPU for the experiment was an eight-core Intel Core i5 @ 2.30 GHz with 16 GB of memory.

3.1. Data Processing. The data used in the experiments were obtained from the CMA Multisource Precipitation Analysis System (CMPAS) [37], which are available through the China Meteorological Data Service Center (http://data.cma.cn). The data include latitude and longitude ranges of 70–140° E and 15–60° N. The experiments included 24 hourly precipitation fusion products with a resolution of 1 km ∗ 1 km from 00 : 00 to 23 : 00 on July 20, 2021; these data were chosen as an example to evaluate and compare the data processing with the foreground display. The data included 7000 latitude points and 4500 longitude points, with over 300 million data points in a single file, and the single file size of the hourly precipitation fusion product was 101.3 MB.

Before image slicing, the image in each layer was transformed into a square. The longest side length was taken as the side length to create a new square canvas. The upper left corner of the original image was overlapped with the upper left corner of the square, and the remainder of the square was filled in white. Thus, layer 0 (7000 ∗ 7000), layer 1 (3500 ∗ 3500), layer 2 (1750 ∗ 1750), layer 3 (875 ∗ 875), and layer 4 (437 ∗ 437) were obtained.

Table 1 shows the minimum and maximum compression ratios of the five data layers, which reached 30 and 95, respectively, after PNG compression and conversion. The average file size of the 24 experimental files was calculated. The degree value equivalent to the pixel interval corresponding to each scale ratio in the five layers of the pyramid was determined based on the scaling coefficient. The pixel size in layer 4 was taken as the size of a single tile to slice the data from the other four layers. The maximum size of a single-slice file was less than 90.7 kB, sufficient to guarantee fast transmission. When the sizes of the slice files were compared, it was found that the slice files tended to be larger in areas with rainfall due to the data distribution in these locations.

3.2. Transmission Efficiency. In the experiments, the time it took for all slices in the compressed file (layer 0) to be transmitted from the server to the browser was a thousand times faster than the time it took for the original file to be transmitted when 256 concurrent channels were used. The compressed PNG transmission efficiency is compared between the full-size image and its slices (Figure 5).

The maximum amount of data that could be requested by the browser includes the map and data slices. In these experiments, the actual maximum amount requested was less than 60 because the visible slices were determined according to the range of the map. Sixty-four concurrent channels with eight subdomains were opened for the requests.

3.3. Rendering Efficiency. Currently, there are three main online WebGIS rendering methods for processing GRIB, NetCDF, and other file formats: GeoJSON processing, binary compression, and grayscale compression. Table 2 lists

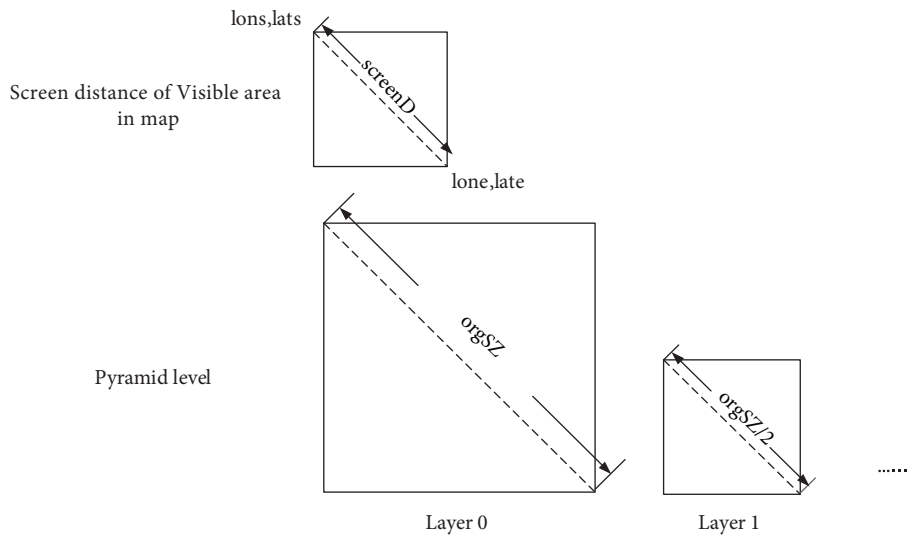

Figure 4: The area relation between the visible screen range and the diagonal width of each layer.

Table 1: Slices after compression.

Layer	Resolution	Compression to PNG (kB)	Range of the single-slice file size (kB)	Number of slices	Distance between adjacent grid points
0	7000×7000	1611.9	[0.6, 79.3]	256	0.01° (1 km)
1	3500×3500	543.0	[0.6, 57.3]	64	0.02° (2 km)
2	1750×1750	174.3	[0.9, 35.8]	16	0.04° (4 km)
3	875×875	56.4	[9.5, 20.4]	4	0.08° (8 km)
4	437×437	17.6	[17.6, 17.6]	1	0.16° (16 km)

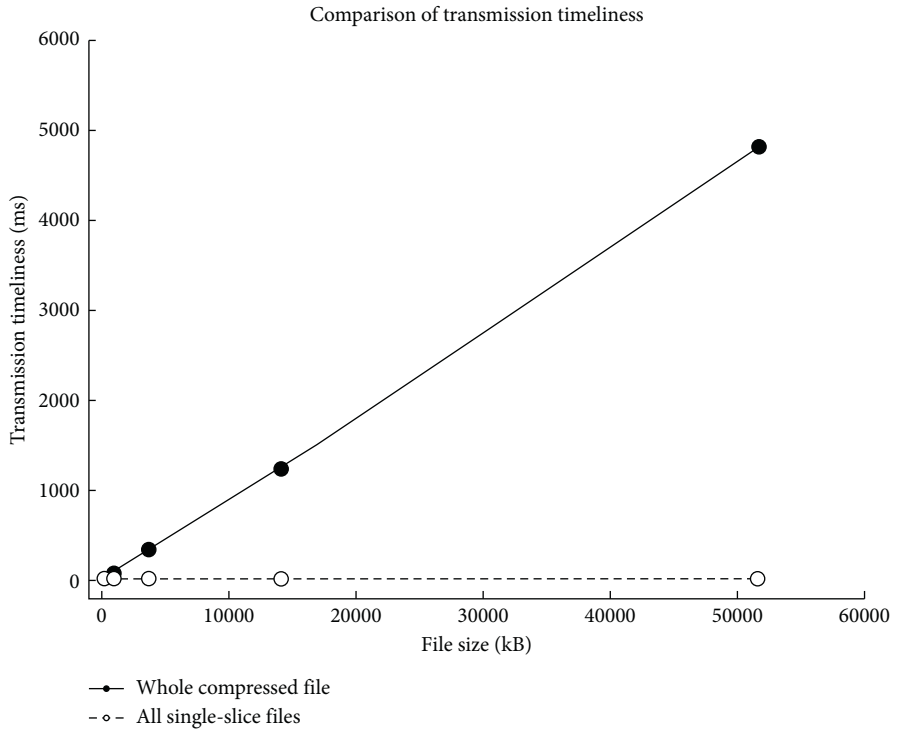

Figure 5: Comparison of transmission timeliness between the whole compression file and the slices.

Table 2: Comparison of the various treatment results and the rendering efficiency.

Treatment	Result (MB)	Rendering time
GeoJSON transformation	253	2.5 s
Binary compression	76	485 ms
Grayscale image compression	89	287 ms
High-low 8-bit compression	1.3	145 ms

the rendering efficiencies of the above methods and the method in this article for a 120 MB file. The same WebGL and WebGIS rendering technology was used for all processes. The results indicate that the rendering efficiency of the high-low 8-bit compression method reduced the time consumption to the order of milliseconds, thus making this the optimal method.

4. Conclusion and Future Work

Due to climate and weather phenomena such as global climate change and the frequent occurrence of extreme weather, the demands for meteorological services have been increasing in relation to various social activities and industries. Accordingly, there is a need for methods of visualizing the spatiotemporal characteristics of meteorological big data for disaster prevention and mitigation for various social activities and industries. Based on the Vue architecture of HTML5, the scheme proposed in this article can be used to quickly visualize grid data of approximately 100 MB in size with PC and mobile browsers as carriers. The multiterminal rendering of technical data is accomplished by combining Python, Node.js, HTML5, Mapbox, and other technologies. The proposed process can support efficient WebGIS rendering of various kinds of large grid data files, thus providing a solution for quickly visualizing industrial data and spatial big data after fusion and improving the efficiency of installation-free visualization of grid big data in browsers.

Data compression: Various compression algorithms, including binary compression, grayscale map compression, and high-low 8-bit compression, were compared in terms of the compression ratio and the loss ratio. High-low 8-bit compression was selected because it enables the visual display of meteorological values accumulated over periods such as years, months, and days. However, the use of this compression algorithm is limited due to the large scale of the accumulated values and the need for visualization accuracy up to three significant digits after the decimal point.

Slicing: Pyramid slicing met the efficiency requirements of this study. The visualization efficiency could be further improved by adopting other algorithms, such as the quadtree algorithm, based on specific requirements.

Transmission and rendering: In this study, the image rendering data were separated from the original data. The data of layer 0 were original, while the data in the other layers were rendered on the web page, thus preserving the eigenvalues in the visualization, which is preferable to grid pumping and visualization. Furthermore, the rendering method based on the WebGL technical framework fully uses the capabilities of the browser, reducing the pressure on the server and taking advantage of combining cloud computing with sliding windows.

Data Availability

The data were saved as Grib2 format, which can be downloaded from http://image.data.cma.cn/test/Z_SURF_C_BABJ_P_CMPA_RT_CHN_0P01_HOR-PRE-20210720.rar, and the datafile can be decoded by the software of Panoply (https://www.giss.nasa.gov/tools/panoply/).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Authors' Contributions

He-Ping Yang and Ying-Rui Sun conceptualized the study and developed the original research program; Nan Chen and Jing-Hua Chen analyzed the data; Qi Wang and Ming Yang handled the data; He-Ping Yang and Xiao-Wei Jiang wrote the original draft; Zi-Mo Huo and Ming-Nong Feng compared the results of the different methods. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (2016YFA0600301); a Major Project of Zhejiang Province, China (Grant 2021C02036); and the Key Research and Development Project of China (Grant 2018YFC1505601). The authors are thankful to Profs. Hui Gao, Qingyun Zhang, and Xia Xu for helpful discussions.

References

[1] M. B. G. Martín, "Weather, climate and tourism a geographical perspective," *Annals of Tourism Research*, vol. 32, no. 3, pp. 571–591, 2005.

[2] L. Bright and D. Maier, *Deriving and Managing Data Products in an Environmental Observation and Forecasting System," the 2005 CIDR Conference*, pp. 162–173, 2005.

[3] T. N. Palmer, J. Barkmeijer, R. Buizza, and T. Petroliagis, "The ECMWF ensemble prediction system," *Meteorological Applications*, vol. 4, no. 4, pp. 301–304, 1997.

[4] Y. Ota, J. C. Derber, E. Kalnay et al., "Ensemble-based observation impact estimates using the NCEP GFS," *Tellus A: Dynamic Meteorology and Oceanography*, vol. 65, no. 1, p. 20038, 2013.

[5] R. Zhang and X. Shen, "On the development of the GRAPES——A new generation of the national operational NWP system in China," *Science Bulletin*, vol. 53, no. 22, pp. 3429–3432, 2008.

[6] J He, K Yang, W Tang et al., "The first high-resolution meteorological forcing dataset for land process studies over China," *Scientific Data*, vol. 7, no. 1, pp. 25–11, 2020.

[7] S. E. Haupt and B. Kosović, "Variable generation power forecasting as a big data problem," *IEEE Transactions on Sustainable Energy*, vol. 8, no. 2, pp. 725–732, 2016.

[8] C. S. Zender, "Analysis of self-describing gridded geoscience data with netCDF Operators (NCO)," *Environmental Modelling & Software*, vol. 23, no. 10-11, pp. 1338–1342, 2008.

[9] P. Berrisford, R. Brugge, L. Steenman-Clark, and D. Li, "The UGAMP use and diagnosis of the ECMWF meteorological analyses," *Systems Analysis Modelling Simulation*, vol. 42, no. 11, pp. 1615–1621, 2002.

[10] M. Ninyerola, X. Pons, and J. M. Roure, "A methodological approach of climatological modelling of air temperature and precipitation through GIS techniques," *International Journal of Climatology*, vol. 20, no. 14, pp. 1823–1841, 2000.

[11] R. B. Schmunk, *Panoply netCDF, HDF and GRIB Data Viewer*, NASA Goddard Institute for Space Studies, 2017.

[12] Y. Q. Wang, "MeteoInfo: GIS software for meteorological data visualization and analysis," *Meteorological Applications*, vol. 21, no. 2, pp. 360–368, 2014.

[13] H Wu, W Zheng, B Luo et al., "Decision making meteorological services system based on geographic information system," *International Joint Conference on Computational Sciences and Optimization*, vol. 2, pp. 53–55, 2009.

[14] F. Berman, A. Chien, K. Cooper et al., "The GrADS project: software support for high-level grid application development," *International Journal of High Performance Computing Applications*, vol. 15, no. 4, pp. 327–344, 2001.

[15] D. Murray, J. McWhirter, Y. Ho et al., "The IDV at 5: new features and future plans," *Proceedings of 25th Conference on International Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology*, vol. 7, no. 2, 2009.

[16] R. Nogueira and E. M. Cutrim, "Applications of "integrated data viewer" (IDV) in the classroom," *Advances in Geosciences*, vol. 8, pp. 63–67, 2006.

[17] J. Gaigalas, L Di, and Z. Sun, "Advanced cyberinfrastructure to enable search of big climate datasets in THREDDS," *ISPRS International Journal of Geo-Information*, vol. 8, no. 11, p. 494, 2019.

[18] M. Linaje, J. C. Preciado, and F. Sanchez-Figueroa, "Engineering rich internet application user interfaces over legacy web models," *IEEE internet computing*, vol. 11, no. 6, pp. 53–59, 2007.

[19] S. Popelka, A. Vondrakova, and P. Hujnakova, "Eye-tracking evaluation of weather web maps," *ISPRS International Journal of Geo-Information*, vol. 8, no. 6, p. 256, 2019.

[20] S. Popić, D. Pezer, B. Mrazovac et al., "Performance evaluation of using protocol buffers in the internet of things communication," in *International Conference on Smart Systems and Technologies (SST)*, pp. 261–265, 2016.

[21] X. Shang, *A Study on Efficient Vector Mapping with Vector Tiles Based on Cloud Server Architecture*, Graduate Studies, 2015.

[22] N. Yap, M. Gong, R. K. Naha, and A. Mahanti, "Machine learning-based modelling for museum visitations prediction," in *Proceedings of the 2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–7, Shenzhen, China, 2020.

[23] L. S. Hsu and O. Obe, *PostGIS in Action*, Manning Publications Co., 2015.

[24] B. N. Hill, *An Analysis of the Factors that Influence the Sargassum Migratory Loop*, Diss., 2016.

[25] K. S. Yarygin, B. A. Kovarsky, T. S. Bibikova, D. S. Melnikov, A. V. Tyakht, and D. G. Alexeev, "ResistoMap-online visualization of human gut microbiota antibiotic resistome," *Bioinformatics*, vol. 33, no. 14, pp. 2205-2206, 2017.

[26] V. Lytvyn, O. Pashchetnyk, O. Klymovych et al., "Assessment of the hydro-meteorological conditions impact on the combat troops operations preparation and conduct in the geo-information subsystem of the automated battlefield management system," *CEUR Workshop Proceedings*, vol. 1, pp. 1063–1076, 2021.

[27] R. S. Prastika, A. N. Afandi, and D. Prihanto, "Mitigation of the alternative energy for the wind farm center considering temperature and wind speed," *MATEC Web of Conferences*, vol. 204, p. 04013, 2018.

[28] Y. Jiang, S. Han, C. Shi, T. Gao, H. Zhen, and X. Liu, "Evaluation of HRCLDAS and ERA5 datasets for near-surface wind over hainan island and south China sea," *Atmosphere*, vol. 12, no. 6, p. 766, 2021.

[29] B. Kastanakis, *Mapbox Cookbook*, Packt Publishing Ltd, 2016.

[30] G. Qiu and J. Chen, *Web-based 3D Map Visualization Using WebGL," 2018 13th IEEE Conference On Industrial Electronics And Applications (ICIEA)*, pp. 759–763, 2018.

[31] T. Parisi, *WebGL: Up and Running*, O'Reilly Media, Inc., 2012.

[32] M. Samir, M. Azab, M. R. M. Rizk, and N. Sadek, "PYGRID: a software development and assessment framework for grid-aware software defined networking," *International Journal of Network Management*, vol. 28, no. 5, p. e2033, 2018.

[33] S. Gao and V. Gruev, "Bilinear and bicubic interpolation methods for division of focal plane polarimeters," *Optics Express*, vol. 19, no. 27, pp. 26161–26173, 2011.

[34] N. Nikiforakis, M. Balduzzi, L. Desmet, F. Piessens, and W. Joosen, "Soundsquatting: uncovering the use of homophones in domain squatting," *Lecture Notes in Computer Science*, vol. 8783, pp. 291–308, 2014.

[35] W. Busch, B. T. Moore, B. Martsberger et al., "A microfluidic device and computational platform for high-throughput live imaging of gene expression," *Nature Methods*, vol. 9, no. 11, pp. 1101–1106, 2012.

[36] P. Halliday, *Vue. Js 2 Design Patterns and Best Practices: Build enterprise-ready, Modular Vue. Js Applications with Vuex and Nuxt*, Packt Publishing Ltd, 2018.

[37] Y. Wang, K. Dai, Z. Zong et al., "Quantitative precipitation forecasting using multi-model blending with supplemental grid points: experiments and prospects in China," *Journal of Meteorological Research*, vol. 35, no. 3, pp. 521–536, 2021.