Efficient and customizable WebGIS system based on edge computing

honglei he (✉ 695869107@qq.com)

Research

Keywords: edge computing, customizable, WebGIS, high performance, map tile sharing

Posted Date: June 1st, 2020

DOI: <https://doi.org/10.21203/rs.3.rs-31462/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Efficient and customizable WebGIS system based on edge computing

Honglei He ^{1,*}

¹ School of Information Engineering, Lianyungang Technical College, Lianyungang 222006, China;
695869107@qq.com

* Correspondence: 695869107@qq.com

Received: date; Accepted: date; Published: date

Abstract: WebGIS currently has a wide range of application requirements and can generate greater economic value. Therefore, there are many units that need to build their own WebGIS system. At present, the way to build a system is usually to purchase professional software and hire professionals to complete the development. Such a construction method requires a high cost, and many of the WebGIS system applications are the same or similar, and repeated construction is also a waste of resources. In order to solve this problem, a WebGIS system based on edge computing is proposed. Comprehensive use of SAAS mode and edge computing methods to solve the needs of users can customize the system. And the system load is reasonably distributed between the server and the browser, which improves the performance of the system. In addition, the system uses edge computing to share map tiles among WebGIS users, reducing the pressure on the server and increasing the speed of use of the WebGIS system. Therefore, the WebGIS system based on edge computing can not only be personalized and customized, but also has good usability and can deal with a large number of visits. After experimental verification, in the case of more than 5 concurrent user requests per second, the system response speed increased by more than 38.6%.

Keywords: edge computing; customizable; WebGIS; high performance; map tile sharing

1. Introduction

WebGIS has achieved amazing development in recent years and is widely used in resource survey, environmental assessment, disaster prediction, land management, urban planning, post and telecommunications, transportation, military public security, water conservancy and power, public facilities management, agriculture, forestry and animal husbandry, statistics, Commercial finance and almost all fields[1-3].

In terms of resource management, it is mainly used in the field of agriculture and forestry, and solves the problems of distribution, classification, statistics, and mapping of various resources in the field of agriculture and forestry[4].

In terms of ecology and environmental management and simulation, it is mainly used in regional ecological planning, environmental status assessment, environmental impact assessment, decision support for pollutant reduction and distribution, decision support for environmental and regional sustainable development, management of environmental protection facilities, environmental planning, etc[5].

In terms of geoscience research and application, it is mainly used in terrain analysis, watershed analysis, land use research, economic geography research, spatial decision support, spatial statistical analysis, mapping, etc.

In network analysis, it is mainly used to establish computer models of transportation networks, underground pipeline networks, etc., to study traffic flow, conduct traffic rules, and other emergency treatments.

In terms of visualization applications, it is mainly used to build a three-dimensional visualization model of cities, regions, or large-scale construction projects and famous scenic spots based on digital terrain models, to achieve multi-angle browsing, which can be widely used in

publicity, urban and regional planning, Engineering management and simulation, tourism and other fields.

Therefore, there are many users who need to build their own WebGIS system. At present, the way to build a system is usually to purchase professional software (such as ARCGIS, mapinfo)[6,7], and hire professionals to complete the development. Such a construction method requires a high cost, which is a burden for most users, especially some non-core business users who are not frequently used. Moreover, in many WebGIS system applications, many are the same or similar, repeated construction is also a waste of resources.

In order to solve this problem, a WebGIS system based on edge computing is proposed: through the SAAS method of cloud computing, users can customize the WebGIS system that meets their needs through online methods, without the need to purchase additional professional software; through edge computing , Lighten the load of the cloud server, so that the system customized by the user has better availability and can deal with a larger amount of access.

2. Related technologies

2.1. Overview of WebGIS

WebGIS is a geographic information system running on the Internet. Through the World Wide Web technology, the use of WebGIS is very convenient and popularized. Any user on the Internet can use geographic information services through a browser, such as browsing, querying, analyzing, and modifying spatial data. Compared with traditional GIS, WebGIS greatly improves the sharing range and operation efficiency of spatial information.

WebGIS uses World Wide Web technology to transmit information, so it has some characteristics of the World Wide Web. For example: with platform independence, users can use a browser to access various types of GIS services on various servers; with distributed source data, WebGIS data can be distributed on multiple servers, and WebGIS data can also be used with other Web Information integration; B / S mode, WebGIS decomposes geographic information functions into server and client to achieve reasonable load distribution.

The current mainstream method is a hierarchical tile map represented by Google and Baidu. The system function is decomposed into a combination of browser-side module and server-side module, as shown in Figure 1. The JavaScript engine on the browser side runs the map module, obtains the map tiles from the server side and organizes them into maps, and can also respond to various operations such as zooming, dragging, and marking of the user. The server provides WMS, WTS, WFS and various application services. This has the advantage of reducing the load on the server and improving compatibility with the browser.

Fig. 1 WebGIS System architecture

1. Map layer model

The map displayed by WebGIS on the browser is divided into multiple levels, as shown in Figure 2. You can add HTML elements in any layer, and locate to the specified geographic coordinates,

so as to achieve the goal of editing the design map. This HTML element can be text, pictures, links, or videos.

Fig. 2 Map layer model

2. Map tiles

In the layer model of WebGIS, the lowest layer is the map tiles. Map tiles are a very important part, the source of the tiles can be made by yourself, and also use open source commercial tile services, such as Google and Baidu maps. If you use a custom method, you need to cut the complete map picture into small pictures one by one, called tiles. Each tile is a 256 * 256 image file, and then uploaded to the server. Each tile has a number, which is determined according to three parameters, namely X coordinate Y coordinate Z zoom level.

The front end of the system uses the Web Mercator projected coordinate system, with the equator as the standard latitude and the prime meridian as the central meridian. The intersection of the two is the origin of the coordinate, positive from east to north and negative from west to south. The map tiles are distributed in a pyramid shape. The top-level ($z = 1$) world map is divided into 4 pieces, and the tile numbers xyz are 001, 101, 011, and 111 (Figure 3).

Fig. 3 Tile numbering

The initial numbers x and y can be calculated from equations (1) and (2), respectively, where λ is the longitude (unit: degree), and Z is the zoom level.

$$X = \left\lceil 2^{z-1} \cdot \left(\frac{\lambda}{180} + 1 \right) \right\rceil \quad (1)$$

$$Y = \left\lceil 2^{z-1} \cdot \left(1 - \frac{\ln \left[\tan \left(\frac{\pi \cdot \varphi}{180} \right) + \sec \left(\frac{\pi \cdot \varphi}{180} \right) \right]}{\pi} \right) \right\rceil \quad (2)$$

Therefore, the format of the map tiles is called: "server path" + "tile number".

2.2. SAAS model of cloud computing

SaaS is a software application model that emerged in the early 21st century with the development of Internet technology and the maturity of application software.

It is a new software service mode provided through the Internet. Users can use the software through the Internet without the need to maintain the software. The platform has full authority to manage and maintain the software. For small and medium-sized enterprises, they can not only use the services they need, but also eliminate the capital investment such as system construction and maintenance, hardware purchase and so on.

The characteristics of the SaaS service method are:

1. The software is based on the Internet and uses a standard browser as the interactive interface.
2. Achieving mobile office, users can use the software anywhere on the Internet without adding any special software and hardware.
3. Have a quick and concise delivery, set-up and training process.
4. Adopt "one-to-many" mode, is a multi-subscriber system architecture, can support thousands of users at the same time.

Self-customization or customization under the multi-tenancy architecture is another core technology of SaaS software. The products of leading manufacturers have made self-customization very user-friendly. Customers can customize fields, menus, reports, views, workflows, etc. according to their company's business processes, so that they can be tailored to SaaS software without programming knowledge.

2.3. WebRTC

WebRTC is a combination product of the libjingle project and GIPS project acquired by Google in 2010. W3C has accepted the technology and is responsible for defining some standards of the browser interface. The IETF is responsible for defining the data format, protocol, security, and all the underlying technologies of WebRTC. libjingle is the part of the load network communication, and GIPS is the part of the media processing.[11]

Google will open source the WebRTC technology and provide a browser-oriented API. Developers only need to use these APIs to develop various types of voice and video communication systems[12,13]. And now that W3C has accepted WebRTC technology and formulated corresponding browser interface standards, all currently used popular browsers support WebRTC.

The communication protocol stack of WebRTC is shown in Figure 4. The bottom layer is the IP of the network layer, and the upper layer is the transport layer. There are two protocols: TCP and UDP. TCP is a reliable connectionless transmission service, and UDP is an unreliable connectionless transmission service. The overhead of TCP transmission is relatively large, all data packets are transmitted along the same network path, so they will arrive in order. The transmission overhead of UDP is relatively small, the data may be lost, the data arrives out of order, and the data packets sent later may arrive first, which is suitable for real-time transmission that does not require high data reliability. So choose UDP to transmit voice and video data, and choose TCP to transmit signaling data for control functions.

Above TCP is TLS, which is also used to ensure data security and integrity. Above is HTTP protocol and WebSocket protocol, which is used to transmit signaling. SSE (Server-sent Events) is a lightweight solution that uses HTTP protocol instead of WebSocket.

Above UDP are ICE (Interactive Connectivity Establishment), STUN (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT). These protocols are used to establish P2P links, especially for users who use NAT to access the Internet.

Fig. 4 WebRTC communication protocol stack

The above protocol is DTLS (Datagram Transport Layer Security), used to ensure data security and integrity. SRTP (Secure Real-Time Transport Protocol) and SCTP (Stream Control Transport Protocol) on top of DTLS are used to securely transmit voice and video streams and data, respectively.

3. Methods

3.1. Design goals

The design goal is to implement a WebGIS system for edge computing (as shown in Figure 5). Through the SAAS method of cloud computing, users can customize the WebGIS system that meets their needs online, without the need to purchase additional professional software; through edge computing, the load of the cloud server is reduced, making the user-customized system have better Availability, can deal with a large number of visits.

Fig. 5 Design goal of WebGIS system based on edge computing

3.2. System customizable design

One of the goals of the system is to realize the customizability of the WebGIS system. Users can customize the user interface, business logic, and data structure according to their own needs. And in order to achieve simple and easy operation, this customization process is realized through a graphical interface.

1 . The principle of customization of WebGIS system

In order to meet the customizability of WebGIS system, first analyze the needs of users. That is to say, the common and differentiated needs are statistically decomposed among the numerous user needs, and these needs constitute the system's needs set. Then give each solution a solution, and each solution is a service unit, as shown in Figure 6.

Fig. 6 The principle of customization for differentiated needs

In order to facilitate user customization, and also reasonably distribute system load. Put the common service unit on the server side and implement it in the form of Web services, and put the different service unit on the user side (edge node) and implement it in the edge computing mode. Common service units such as data access management, user management, etc. ; differentiated service units include legend service, data statistics, and vehicle positioning.

Since the service units are independent of each other, different functions can be realized through different combinations. Users can customize their own service assembly to achieve their own business logic functions.

Different users have different needs. When customizing the system, they select the corresponding service unit to assemble their own system according to their own needs[14]. As shown in Figure 7.

Fig. 7 Customization and run of user systems

3.3. Tile sharing design

Tile sharing refers to the use of edge computing without increasing server resources, sharing map tiles among various browsers (edge nodes) to improve the performance of the WebGIS system.

1. The significance of tile sharing

Assuming that K units customize their own WebGIS system and each unit customizes an instance, then there are K instances of WebGIS system stored on the server.

Definition 1: The number of map tiles of each WebGIS system instance is expressed as: N_i^{tile} ($1 \leq i \leq K$).

Definition 2: The number of map update users per instance is expressed as: N_i^{user} ($1 \leq i \leq K$).

Definition 3: The number of tiles that the server needs to provide for downloading per unit time is expressed as N_{total} . N_{total} is expressed as Equation 3.

$$N_{total} = \sum_{i=1}^K N_i^{user} \cdot N_i^{tile} \quad (3)$$

As the number of users and the number of WebGIS system instances increase, the load of the server will increase rapidly, so that the loading speed of user maps decreases and the user experience deteriorates.

In order to solve this problem, you can use the configuration to improve the performance of the server, but it also brings an increase in the cost of use. If edge computing is used to share map tiles, the system performance can be improved without increasing the server configuration, as shown in Figure 8.

Fig. 8 The Significance of Edge Computing for Sharing Map Tiles

2. Principles of map tile sharing

The principle of map tile sharing is shown in Figure 9. Each user's terminal can be regarded as an edge node, a communication connection is established between the nodes, and then map tiles can be shared without downloading the tiles from the server. For example, multiple users use a WebGIS instance at the same time. As long as one of the users downloads the map tiles from the server, they can share these tiles with other users.

First, they need to establish a communication channel between them. Using WebRTC technology can achieve this goal, first establish a WebRTC PeerConnection connection, and then create an RTCDatachannel channel, using this channel, you can share data between various users.

Then, since the map tiles are in IMG format, the RTCDatachannel can only transmit text format. So first convert it to Base64 format, then convert it to File format, and transmit it via RTCDatachannel in text mode. After receiving, the receiving end converts it into map tiles in IMG format.

Fig. 9 The principle of edge maps sharing map tiles

3.4. Architecture design

The design of WebGIS structure based on edge computing is shown in Figure 10. The presentation layer is the user interface on the browser side, and the business logic layer includes two parts: the server-side web service and the browser-side business logic layer. The business logic layer is divided into a front end and a server end to facilitate customization of business logic users.

The customization of the user's WebGIS system includes customization of the user interface, business logic, and data structure. The user interface is the presentation layer on the browser side, which can be customized through HTML and CSS; the data structure can be customized through metadata; and the customization of business logic is the choice of each service unit.

After the customization of the user's WebGIS system is completed, the customized data is stored in the metadata on the server side. When the user uses the customized system, the system configuration information is read from the server-side metadata and executed.

In order to realize the sharing of map tiles between edge nodes, the WebRTC module is responsible for the communication between the two browsers. When users access the WebGIS server, they communicate through the https protocol. When WebRTC accesses the signaling server and ICE server, it uses TCP / IP protocol to communicate. When sharing map tiles between edge nodes, UDP / IP protocol is used for communication between browsers.

Fig. 10 Architecture of WebGIS based on edge computing

3.5. Performance analysis of WebGIS based on edge computing

One of the design goals of WebGIS based on edge computing is to speed up the operation of the system by placing some service units on the client. The following two models compare the performance of the service unit on the server and client.

1. The service unit is placed on the server side

The model of the system is shown in Figure 11. All client requests need to be queued for processing by the server.

Fig. 11 Queuing model where the service unit is placed on the server

Assuming that the service request arrival time of each user in the system follows Poisson distribution, the number of servers is 1, and the running time required by each service unit is a negative exponential distribution. In the case of not overloading, the queue space on the server side is unlimited. The system can be regarded as a queuing model of $M / M / 1 / \infty / \infty / FCFS$.

Definition 4: There are D clients, and the computer capability of the i -th client is expressed as $E_i^c, i \in D$.

Definition 5: There are K service units provided by the server, where service y is expressed as $S_y, y \in K$.

Definition 6: The workload generated by the call from the client i to the service unit S_y is expressed as w_i^y .

Definition 7: The computer capacity of the server is expressed as E_s .

Definition 8: Request rate of client i to service S_y : λ_i^y (req / s)

The amount of work requested for the service S_y per unit time is expressed as G^y , and G^y is shown in Equation (4).

$$G^y = \sum_{i=1}^D \lambda_i^y \cdot w_i^y \quad (4)$$

The total requested workload of all service units arriving in a unit time is expressed as G , and G is shown in equation (5).

$$G = \sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot w_i^y) \quad (5)$$

The total number of service requests arriving per unit time is expressed as N , N see equation (6).

$$N = \sum_{y=1}^k \sum_{i=1}^D \lambda_i^y \quad (6)$$

The average processing time of each service request is expressed as T , then $T = G / (N \cdot E_s)$, according to equations (5) (6), equation 7 can be obtained.

$$T = \frac{\sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot w_i^y)}{E_s \cdot \sum_{y=1}^k \sum_{i=1}^D \lambda_i^y} \quad (7)$$

The arrival rate of the service request is expressed as λ , then: $\lambda = N$, according to equation (6), equation (8) can be obtained.

$$\lambda = \sum_{y=1}^k \sum_{i=1}^D \lambda_i^y \quad (8)$$

The speed of service processing is expressed as μ , then $\mu = 1 / T$. According to equation (7), equation (9) can be obtained.

$$\mu = \frac{E_s \cdot \sum_{y=1}^k \sum_{i=1}^D \lambda_i^y}{\sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot w_i^y)} \quad (9)$$

ρ represents the probability of at least one service request in the system, $\rho = \lambda / \mu$, according to equations (8) (9), equation (10) can be obtained

$$\rho = \frac{\sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y)}{E_s} \quad (10)$$

The average queue length in the system is expressed as L_s , $L_s = \rho / (1 - \rho)$, and the equation (11) can be obtained.

$$L_s = \frac{\sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y)}{E_s - \sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y)} \quad (11)$$

The average number of customers waiting for service in the system is expressed as L_q , because $L_q = L_s / \rho$. The available equation (12).

$$L_q = \frac{\rho}{1 - \rho} - \rho = \frac{\rho^2}{1 - \rho} = \frac{\lambda^2}{\mu^2 - \lambda\mu} \quad (12)$$

The average waiting time of each service request W_q : $W_q = L_q / \lambda$, according to equations (8) (9) (12), equation (13) can be obtained.

$$W_q = \frac{\lambda}{\mu^2 - \lambda\mu} = \frac{\left(\sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y) \right)^2}{E_s \cdot \sum_{y=1}^k \sum_{i=1}^D \lambda_i^y \left(E_s - \sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y) \right)} \quad (13)$$

For the customized system of user i , the number of requests sent per unit time is expressed as N_i , then N_i see equation (14).

$$N_i = \sum_{y=1}^k \lambda_i^y \quad (14)$$

User i 's customized system, the requested workload per unit time is expressed as G_i , G_i see equation (15).

$$G_i = \sum_{y=1}^k (\lambda_i^y \cdot W_i^y) \quad (15)$$

The total running time of the customized system of user i is expressed as T_i^s , see equation (16).

$$T_i^s = N_i \cdot W_q + \frac{G_i}{E_s} \quad (16)$$

According to equations (13) (14) (15) (16), equation (17) can be obtained.

$$T_i^s = \frac{\sum_{y=1}^k \lambda_i^y \left(\sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y) \right)^2}{E_s \cdot \sum_{y=1}^k \sum_{i=1}^D \lambda_i^y \left(E_s - \sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y) \right)} + \frac{\sum_{y=1}^k (\lambda_i^y \cdot W_i^y)}{E_s} \quad (17)$$

2. The service unit is placed on the user side

The service units are all placed on the user side for implementation, and the system model is shown in the figure 12.

Fig. 12 The queuing model of the service unit placed to the client

If all service units are running on the client, the system running time of user i is expressed as T_i^c . According to equation (17), equation (18) can be obtained.

$$T_i^c = \frac{\sum_{y=1}^k \lambda_i^y \left(\sum_{y=1}^k (\lambda_i^y \cdot W_i^y) \right)^2}{E_i^c \cdot \sum_{y=1}^k \lambda_i^y \left(E_i^c - \sum_{y=1}^k (\lambda_i^y \cdot W_i^y) \right)} + \frac{\sum_{y=1}^k (\lambda_i^y \cdot W_i^y)}{E_i^c} \quad (18)$$

3. Comparison of system response time

In the actual operation of user i 's system, the transmission delay must also be considered. The transmission delay is expressed as T_i^{delay} . The network delay is expressed as T_i^{network} . The amount of data transferred is expressed as DT_i . The transmission bandwidth is expressed as BW . Then there is equation (19)

$$T_i^{\text{delay}} = T_i^{\text{network}} + \frac{DT_i}{BW} \quad (19)$$

If the system is running on the server, the response time of the system is expressed as T_i^{rs} , $T_i^{\text{rs}} = T_i^{\text{s}} + K \cdot T_i^{\text{delay}}$. T_i^{rs} see equation (20).

$$T_i^{\text{rs}} = K \cdot T_i^{\text{delay}} + \frac{\sum_{y=1}^k \lambda_i^y \left(\sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y) \right)^2}{E_s \cdot \sum_{y=1}^k \sum_{i=1}^D \lambda_i^y \left(E_s - \sum_{y=1}^k \sum_{i=1}^D (\lambda_i^y \cdot W_i^y) \right)} + \frac{\sum_{y=1}^k (\lambda_i^y \cdot W_i^y)}{E_s} \quad (20)$$

If the system is running on the user side, the response time of the system is expressed as T_i^{rc} . Because the transmission delay is 0, $T_i^{\text{rs}} = T_i^{\text{c}}$. According to equation (18), equation (21) can be obtained.

$$T_i^{\text{rc}} = \frac{\sum_{y=1}^k \lambda_i^y \left(\sum_{y=1}^k (\lambda_i^y \cdot W_i^y) \right)^2}{E_i^c \cdot \sum_{y=1}^k \lambda_i^y \left(E_i^c - \sum_{y=1}^k (\lambda_i^y \cdot W_i^y) \right)} + \frac{\sum_{y=1}^k (\lambda_i^y \cdot W_i^y)}{E_i^c} \quad (21)$$

Suppose $E_s = 5 E_i^c$, $E_s = 1000W$, $\lambda_i^y = 10$, $k = 2$, $T_i^{\text{network}} = 0.05$ (s), $DT_i = 2$ KB, $BW = 1$ Mbps. The relationship between the number D of clients and the response time T_{rs} of the system running on the server and the time T_{rc} of the system running on the client is shown in the figure 13.

Fig. 13 Relationship between the number of clients D and response time

Suppose $E_s = 5 E_i^c$, $E_s = 1000W$, $\lambda_i^V = 10$, $D = 10$, $T_i^{\text{network}} = 0.05$ (s), $DT_i = 2$ KB, $BW = 1$ Mbps. The relationship between the number K of service units included in each user system and the response time Trs of the system running on the server and the time Trc of the system running on the client are shown in figure 14.

Fig. 14 Relationship between the number of service units K and response time

Suppose $E_s = 5 E_i^c$, $E_s = 1000W$, $K=2$, $D = 10$, $T_i^{\text{network}} = 0.05$ (s), $DT_i = 2$ KB, $BW = 1$ Mbps. The relationship between each user's access request frequency λ and the response time Trs of the system running on the server and the time Trc of the system running on the client are shown in Figure 15.

Fig. 15 Relationship between request frequency λ and response time

From the comparison of the above three situations, we can see that with the increase in the number of users, the frequency of access requests, and the number of service units included in the user system, the response time of the service unit running on the client is faster. Therefore, compared with the traditional server model, the WebGIS system based on edge computing has better performance.

4. Implementation of the system

4.1. Implementation of customizable system

4.1.1. Implementation of the service unit

First analyze the common and different needs of WebGIS, and then build the corresponding service unit. Through analysis, commonly used WebGIS system requirements include: user management, geographic information management, map management, type switching, data editing, mapping, timer, data search, data customization, etc.

◎ Common requirements: including user management, data access, etc. The corresponding service unit is implemented by ASP.net on the server side.

◎ Different needs: including map services, data editing, chart drawing, etc. The corresponding service unit is implemented in JavaScript on the browser side[16,17].

Different needs, such as map services and data editing, the corresponding service unit implementation methods are as follows.

1. Map Service Unit

Through a form (Figure 16), users can customize the tiles, controls, initial geographic coordinates, width and height of the map objects in the WebGIS system to generate corresponding map service units.

Create a new Map object			
ID	map_1	Alignment	center
Map tile address:	userPath\[x] [y] [z].png		
Baidu map tile:	☐ Road ☐ Satellite	Google map tile:	☐ Road ☒ Satellite
center latitude and longitude	(78, -131)	Zoom level	4
Width	500	Height	400
controls	☐ Big		<input type="button" value="Create"/> object

Fig. 16 Create a map service unit

Users can choose to use public map tiles or custom tiles. Common map tiles such as google map tiles or Baidu tiles. If users choose to use their own map tiles, they need to number the tiles (Figure 17) and upload them to the server.

Fig .17 Map tile making

When a user creates a map service unit, the corresponding HTML and JavaScript codes are generated according to the information in the form. as follows:

```
<div id='map_1' style=' width:700px; height:500px;'></div>
<script>
var tileServer='[userPath][x]_[y]_[z].png';
var map_1 = new GMap2(document.getElementById('map_1'));
map_1.setCenter(new GLatLng(78, -131),4);
map_1.addControl(new GSmallMapControl());
</script>
```

The final generated front-end code also contains the JavaScript code of the map object. In this article, the Google map code is used for reference, and Openlayers can also be used. as follows:

```
<script src=" maps.js" type=text/javascript> </script>
```

The result of running the map service unit is shown in Figure 18.

Fig. 18 Map service unit operation result

2. Data editing unit

Through a form (Figure 19), the user defines a data editing unit at the front end. The ID, height, width and initial number of rows and columns of data editing can be customized.

Create a new Data sheet			
ID	<input type="text"/>	Rows	<input type="text"/>
Width	<input type="text"/>	Cols	<input type="text"/>
Height	<input type="text"/>	<input type="button" value="Great"/>	

Fig. 19 Create a data editing unit

When a user creates a data editing unit, the corresponding HTML and JavaScript codes are generated based on the information in the form. The HTML code is an internal frame <iframe>, and JavaScript is the Handsontable plugin using JQuery. The result of the operation of the data editing unit is shown in Figure 20, and the user can freely edit the data.

	A	B	C	D	E	F	G	H
1	coordinates	Cost(yuan)	name					
2	(80.6, -140.0)	8650000	stadium					
3	(79.5, -116.8)	3620000	playground					
4	(79.8, -102.0)	790000	Basketball court					
5								

Fig.20 Operation result of data editing unit

4.1.2. The process of visual customization

For example, suppose a user's needs are to customize the WebGIS system shown in Figure 21. The upper part is the display area of the map, the middle row is 5 function buttons, the lower left is the chart type option, and the lower right is the data table. The user edits the data in the data table, and the geographic coordinate values can be obtained from the map by clicking (through the "get map coordinates" button), and then set the chart type, color, etc. on the left, click the "draw" button, it will be Generate a three-dimensional histogram on the map.

Fig.21 Examples of user needs.

So the user's demand set is: {map display, data display and editing, chart drawing, data access, coordinate reading}

Corresponding service set: {map service, data editing service, chart service, data access service, coordinate reading service}

In order to realize the customization of the visual WebGIS system and make the customization process simple and easy to operate, you can use the "editor" to customize the system. The WebGIS editor is above the toolbar, and below is the editing area, as shown in Figure 22.

The customization process using WebGIS editor is as follows:

- ◎ Insert the table and other elements through the toolbar, and generate corresponding HTML and CSS codes in the editing area, thus realizing the customization of the user interface.
- ◎ Insert service objects (such as maps, data tables, drawings, etc.) at the set position through the toolbar and bind their relationships, so that business logic customization is achieved.

The service unit and the relationship binding between the service units are all implemented by JavaScript code. These JavaScript code and the HTML and CSS code of the customized user interface constitute the user's front-end code, and the user metadata is stored in the server after editing. When users use their own customized system, they will read the front-end data from the server to the browser and execute it, and cooperate with the server-side service unit (tile service, etc.) to complete the operation of the customized system.

Fig.22 The process of customization

4.2. Implementation of map tile sharing

When the new node accesses the WebGIS system, it establishes a communication connection with other edge nodes through WebRTC. When you need to download the tiles, send a request broadcast, other nodes respond after receiving the broadcast, and then download to achieve sharing.

4.2.1. Establish a communication channel between nodes

To achieve tile sharing between browsers (edge nodes), a communication connection must be established between the browsers. This goal can be achieved through WebRTC technology, as shown in Figure 23.

Fig .23 Establish a communication channel between nodes

1. Working process of communication module

- (1) Two browsers (edge nodes) are connected to the WebGIS server, see steps ①②.
- (2) The two browsers are respectively connected to the signaling server to obtain their own signaling identification. See steps ③④.
- (3) The communication initiator creates an instance through the RTCPeerConnection interface and issues an Offer. After receiving the Offer, the communication receiver will create a local instance through the RTCPeerConnection interface. Then it will reply to the initiator of the communication. Both parties negotiate ICE (interactive connection establishment), see steps ⑤⑥.
- (4) Both communication parties establish an RTCPeerConnection connection, and then create an RTCDDataChannel instance. RTCDDataChannel can transmit data in any format, see step ⑦.

2. The creation process of RTCPeerConnection

When a browser initiates a real-time communication connection call, the sequence of establishing an RTCPeerConnection connection is shown in Figure 24.

Fig. 24 RTCPeerConnection connection establishment timing

- (1) First, the application (PeerConnection listening service) creates a Peerconnection Factory, which is a factory class used to generate PeerConnection.

- (2) Then, create a PeerConnection instance.
- (3) Create a local media stream, and add the video stream and audio stream to the corresponding track.
- (4) Add this media stream to the PeerConnection instance (AddStream), or submit stream changes.
- (5) The PeerConnection instance connected to the calling party generates an Offer, and the application (monitoring PeerConnection service) sends the offer to the remote callee through a signaling service (such as WebSocket) after receiving the Offer.
- (6) After receiving the Offer, the remote callee generates Answer and sends it to the application program (listening PeerConnection service) through the signaling service, and then transfers it to the local PeerConnection instance. After the negotiation between the two parties is completed, the local and remote SDP are set up and the connection is established (through the ICE mechanism).
- (7) When the local PeerConnection instance receives the media stream sent by the other party, it is handed over to the application (listening to the PeerConnection service), and then played or other processing through the browser.

3. The creation process of RTCDataChannel

DataChannel can transmit binary data or text data. DataChannel can be set to an ordered or unordered transmission method, or it can be set to a reliable or unreliable method. The setup message is transmitted through the DATA_CHANNEL_OPEN datagram, the format of the DATA_CHANNEL_OPEN datagram is shown in Figure 25.

First, you must first create a RTCPeerconnection to establish a connection. Then create a DataChannel object on this RTCPeerconnection. Multiple DataChannel objects can be created on an RTCPeerconnection, and each DataChannel object can be given a name (label) and configuration options (options).

In the JavaScript interface of WebRTC, the above datagram configuration options can be achieved by setting parameters when creating the DataChannel object.

Bit	+0..7	+8..15	+16..23	+24..31
0	Message Type(0x3)	Channel Type	Priority	
32	Reliability			
64	Label length		Protocol length	
...	Label			
...	Protocol			

Fig. 25 DATA_CHANNEL_OPEN datagram format

When we transfer map tiles between browsers, the established peerconnection has no audio and video, and no stream is added. Only datachannel. Therefore, there is no video or audio in the sent offer (sdp), the content is as follows:

```
{
  "type": "offer",
  "sdp": "v=0 \r\n o=-8335070345506316053 2 IN IP4 127.0.0.1 \r\n ns=-\r\n nt=0 0 \r\n na=msid-semantic: WMS \r\n nm=application 9 DTLS / SCTP 5000 \r\n nc=IN IP4 0.0.0.0 \r\n nb=AS: 30 \r\n na=ice-ufrag: PxEn \r\n na=ice-pwd: ib3DQhicOqJWFA9GyinV0SXn \r\n na=ice-options: trickle \r\n na=fingerprint: sha-256 08: 3A: 3A: FA: 09: 26: 35: 74: F7: 81: 6F: D5: CE: AB: B2: DF: 87: CF: FE: 45: 1B: AB: C4: 87: A2: 5E: 52: E2: A5: 45: A1: 90 \r\n na=setup: active \r\n na=mid: data \r\n na=sctpmap: 5000 webrtc-datachannel 1024 \r\n n"}

```

4.2.2. Transmission of map tiles

The flow chart of map tile transmission is shown in Figure 26. Node A calculates the tiles to be used according to the geographic coordinates of the current map, and by default, sends a tile download request to the server. After receiving the download request, the server calculates the

current number of concurrent downloads. If the download volume exceeds the threshold N (N is customized according to the bandwidth and tile size), the node A is notified to enable the tile sharing method to obtain tiles from other nodes. If the download volume does not exceed the threshold N , the node is allowed to download tiles.

Fig.26 Flow chart of tile transmission

If node A receives the notification from the server, the tile sharing method is enabled. Will send a tile request broadcast, the broadcast content contains the tile number. After receiving the broadcast, other nodes query whether they have the tile, and if there is a tile, they send a response. Node A selects the node that responds first (such as Node B) and sends a transmission request. Then start receiving tiles.

If node A does not receive a response from another node within time T (for example, 300 ms), it attempts to download from the server.

The function code of the tile request broadcasting is as follows:

```

function searchTile (url, callback) { // Send request according to tile number url and return tile
  rtc.sendSearch (rtc.connections [connId], url); // rtc is a WebRTC instance
  connId = connId + 1;
  if (connId >= rtc.connections.length)
    connId = 0;
  rtc.on ('receive_file', function (sendId, data, name) {
    callback (data, name);
  });
  ...
}
  
```

Since the map tiles are in IMG format, RTCDatChannel can only transmit text format. So first convert it to Base64 format, then convert it to File format, and transmit it via RTCDatChannel in text mode. The corresponding conversion function is as follows:

```

function getImgToBase64 (url, callback) { // Convert the picture to Base64
  var canvas = document.createElement ('canvas'),
    ctx = canvas.getContext ('2d'),
    img = new Image;
  img.crossOrigin = 'Anonymous';
  img.onload = function () {
  
```



```

        canvas.height = img.height;
        canvas.width = img.width;
        ctx.drawImage (img, 0,0);
        var dataURL = canvas.toDataURL ('image / png');
        callback (dataURL);
        canvas = null;
    };
    ...
}

function dataURLtoFile (dataurl, filename) { // Convert base64 to File format
    var arr = dataurl.split (','), mime = arr [0] .match (/: (. *?); /) [1],
        bstr = atob (arr [1]), n = bstr.length, u8arr = new Uint8Array (n);
    while (n-) {
        u8arr [n] = bstr.charCodeAt (n);
    }
    return new File ([u8arr], filename, {type: mime});
}

```

5. Results and Discussion

The experimental environment is 2 servers and 4 clients. 2 servers use Alibaba Cloud ecs server, the specific configuration is CPU: 2 cores; memory: 8G; bandwidth: 5M; operating system Windows Server 2008 R2 64-bit Enterprise Edition. One of them is a WebGIS server, and the other is a signaling and STUN server.

The client uses Intel core i5-6200U 2.3GHz CPU, 4GB memory, 500G hard disk, and the operating system is windows7. The browser uses chrome, version 71.0.3578.98. The bandwidth between each browser node is 10M.

5.1. Customizable test

Through the assembly of service units, the WebGIS system can be customized freely. Use {map service, data editing service, legend drawing service, data access service} several service units to make the WebGIS legend drawing system shown in Figure 27.

Fig. 27 Customized legend drawing WebGIS system

Use {map service, data editing service, chart service, data access service, coordinate reading service} several service units to make the WebGIS chart drawing system shown in Figure 28.

Fig .28 Customized chart drawing WebGIS system

5.2. Performance test

Selected data editing, chart drawing, legend drawing service unit for testing. Compare their running performance on the server side and browser side respectively.

Test method: browser-side performance is tested with JavaScript, and server-side performance is tested with http_load software.

1. Data editing unit

If the data editing unit is on the server side, the test code includes a data query and data addition, using http_load test. At different access frequencies R (times / sec), the response time Trs of the data editing unit is shown in Table 1. The corresponding histogram is shown in Figure 29.

Table 1. Response time of data editing unit under different access frequency R (server-side operation)

R	1	2	3	4	5	6	7	8	9	10
Trs(ms)	65.2	66	66.3	66.8	67.5	67.5	67.8	68.1	68.3	69.6
R	20	30	40	50	60	70	80	90	100	110
Trs(ms)	70.2	72	72.4	73.2	73.2	74.5	74.5	84.5	125	240

Fig .29 Response time of the data editing service unit running on the server

It can be seen from the figure that as the access frequency increases, the response time will increase rapidly when the server is close to full load.

If the data editing unit runs on the browser side. Using JavaScript test, the response time of each data edit (add, delete) is less than 1ms.

Assume that the user needs to perform 20 operations such as data addition, deletion, and modification for each data edit. Then if the data editing unit runs on the server side, 20 service requests are required. If the data editing unit runs on the browser side, only one service request is required between the user and the server.

Suppose there are 60 clients. According to Table 1, the frequency of user data editing R_u (times / minute), the total response time of the data editing unit running on the browser side Trc , the total response of the data editing unit running on the server side Trs is shown in Figure 30.

Fig. 30 Data editing unit, comparison of Trs and Trc at different access frequencies

According to the table calculation, the average value of Trc / Trs is 4.8%. So the total response time of the data editing unit running on the browser side is 95% faster than the total response time running on the server side.

2. Chart drawing unit

If the chart drawing unit is on the server side, the test code includes 10000 data accumulation calculations and 500K data transmission (simulating the file size of the drawn picture), using `http_load` test. The response time is expressed in Trs .

If the chart drawing unit runs on the browser side, the test code uses 10000 data accumulation calculations. Test with JavaScript. The response time is expressed in Trc .

Suppose there are 1 clients. The relationship between Trs and Trc is shown in Figure 31 under different access frequencies R (times / sec).

Fig. 31 Chart drawing unit, the relationship between Trs and Trc at different access frequencies

The average value of Trc / Trs is 31.6%. So the total response time of the chart drawing unit running on the browser side is 68.4% faster than the total response time running on the server side.

3. Legend drawing unit

If the legend drawing unit is on the server side, the test code includes 1000000 data accumulation calculations and 500K data transmission (simulation drawing 1000 * 1000 pixels legend), using http_load test. The response time is expressed in Trs.

If the legend drawing unit runs on the browser side, the test code uses 1,000,000 data accumulation calculations. Test with JavaScript. The response time is expressed in Trc.

In the case of access frequency 1 (times / sec), the relationship between the number of clients D and Trs and Trc is shown in the figure 32.

Fig. 32 Legend drawing unit, the relationship between the number of clients D and Trs, Trc

The average value of Trc / Trs is 61.4%. So the total response time of the legend drawing unit running on the browser is 38.6% faster than the total response time running on the server.

5.3 .Tile sharing experiment

The experiment simulates adding a new user to the WebGIS system at a certain moment, and then compares the time for the new user to download the map tiles. The number of tiles downloaded each time is 12 and the experimental data is shown in Table 2.

Table 2. Server download and tile sharing download time (seconds)

Server bandwidth	Server download time	Shared tiles (2 users)	Shared tiles (3 users)
10K bps	20.91	1.114	0.519
20K bps	13.56	0.982	0.487
30K bps	8.60	0.851	0.336
50K bps	5.12	0.633	0.271
100K bps	2.79	0.581	0.262
500K bps	0.483	0.381	0.227

In the case of the same server bandwidth, when using shared tiles, if multiple users already use the WebGIS system, the time for new users to download tiles is significantly reduced, which means that the more users participating in sharing, the more users use the system The faster the speed, as shown in Figure 33.

Fig. 33 Comparison of server download mode and tile sharing mode

As can be seen from Table 2, under different server bandwidths, the smaller the average available bandwidth of the server, the more obvious the advantage of tile sharing, as shown in Figure 34. Therefore, in the case of limited server bandwidth, the tile sharing method will greatly improve the efficiency of the system.

As can be seen from Figure 34, when the average available bandwidth of the server is less than 500K, and there are more than 3 shared users. The download speed of the tile sharing method has been increased by more than 50%.

Fig. 34 Comparison of server download and download sharing under different server bandwidths

6 .Conclusion

The design goals of the WebGIS system based on edge computing are: easy to use, rich in functions, free to personalize, and have high-speed and stable performance.

In order to meet the customizability of WebGIS system, first analyze the needs of users. Statistically decompose the common requirements and differentiated requirements from user

requirements. These requirements constitute the system's requirement set. Then give each solution a solution, and each solution is a service unit. Different users have different needs. When customizing the system, they select the corresponding service unit to assemble their own system according to their own demand set.

In order to improve the performance of the system, the common service unit is placed on the server side, and the differentiated service unit is placed on the user side to achieve edge computing. After experimental testing, the service unit is put on the client to run, and the response speed is increased by at least 38.6%.

In order to simplify the customization process, the WebGIS system is customized through a graphical interface.

For the same WebGIS system, all map tiles are stored on the server side. If multiple users access at the same time, the server needs to provide tiles to these users separately, which is a heavy load. If edge computing technology is used, the server only needs to provide map tiles to one of the users, and then these tiles are transmitted between these users, which not only improves the speed, but also reduces the pressure on the server, and the more users participate in the sharing, the faster the system The sooner. After experimental testing, sharing among multiple users improved the download speed.

Abbreviations

TCP: Transmission Control Protocol;UDP: User Datagram Protocol; HTTP: HyperText Transfer Protocol;NAT: Network Address Translation;GIS: Geographic Information System;WebRTC: Web Real-Time Communication ;DTLS:Datagram Transport Layer Security; SRTP :Secure Real-Time Transport Protocol; SCTP :Stream Control Transport Protocol;

Acknowledgements

The author would like to thank Yuxuan He from Xuzhou University Of Technology for her help. Thanks to the Information Center of Lianyungang Vocational and Technical College for providing the experimental environment.

Author Contributions

Honglei He wrote the entire article. The author read and approved the final manuscript.

Funding

This research was supported by the Lianyungang High-tech Zone Science and Technology Project Plan (Project Number: ZD201915).

Availability of data and materials

The relevant analysis data used to support the findings of this study are included in the article.

Competing interests

The authors declare no conflict of interest.

References

1. Ryan, G. H.; Steven, L. P.; Christopher, S. B.; Paul, C. R.; Mark, W. J.; Anne, E. H. Using WebGIS to Develop a Spatial Bibliography for Organizing, Mapping, and Disseminating Research Information: A Case Study of Quaking Aspen. *Rangelands*, **2019**, 41(6).
2. Mu, W.; Gong, S.; Stevan, S.; Zhang, X.; Radoslava, K.; Feng, J. WebGIS-based suitability evaluation system for Chinese table grape production. *Computers and Electronics in Agriculture*, **2019**, 165.
3. Zhang, B.; Ye, Y.; Shen, X.; Mei, G.; Wang, H. Design and implementation of levee project information management system based on WebGIS. *Royal Society open science*, **2018**, 5(7).
4. Shaoming, P.; Yanwen, C.; Zhengquan, X. An enhanced active caching strategy for data-intensive computations in distributed GIS. *The Journal of Supercomputing*, **2017**, 73, pp. 4324-4346.
5. Wang, Y.; Liu, H. Research on WebGIS and Its Architecture. *Surveying Engineering*, **2009**, 18, pp. 70-73.
6. Guo, J.; Yan, P. Analysis and Design of Urban Tourism Information System Based on WebGIS. *Energy Procedia*. **2011**, 13, pp. 3794-3799.
7. Krishna, R.; Iqbal, J.; Gorai, A. K. Groundwater vulnerability to pollution mapping of Ranchi district using GIS. *Applied Water Science*. **2015**, 12, pp. 345-358.
8. Okay, F. Y.; Ozdemir, S. A fog computing based smart grid model. In Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, 11-13 May 2016.
9. Kumari, A.; Tanwar, S.; Tyagi, S.; Kumar, N.; Obaidat, M. S.; Rodrigues, J. J. P. C. Fog Computing for SmartGrid Systems in the 5G Environment: Challenges and Solutions. *IEEE Wirel. Commun.* **2019**, 26, 47-53. [[CrossRef](#)]
10. Yan, Y.; Su, W. A fog computing solution for advanced metering infrastructure. In Proceedings of the 2016 IEEE/PES Transmission and Distribution Conference and Exposition (T D), Dallas, TX, USA, 3-5 May 2016.
11. WebRTC. Available online: <https://webrtc.org/> (accessed on 16 February 2020).
12. Atwah, R.; Iqbal, R.; Shirmohammadi, S.; Javadtalab, A. A Dynamic Alpha Congestion Controller for WebRTC. In Proceedings of the 2015 IEEE International Symposium on Multimedia (ISM), Miami, FL, USA, 14-16 December 2015; pp. 132-135.
13. Kilinc, C.; Andersson, K. A Congestion Avoidance Mechanism for WebRTC Interactive Video Sessions in LTE Networks. *Wirel. Pers. Commun.* **2014**, 77, 2417-2443. [[CrossRef](#)]
14. Koning, M.; Sun, C. A.; Sinnema, M. VxBPEL: Supporting variability for Web services in BPEL. *Information & Software Technology*. **2009**, 51, pp. 258-269.
15. El-Kady, M. S.; ElMesmary, M. A. Creating spatial database of the foundation soil in Aljouf area using GIS. *Innovative Infrastructure Solutions*. **2018**, 3, pp. 1-6.
16. Wang, T. T.; Shen, Q. J. JavaScript debugger software architecture. *Journal of Natural Science of Hunan Normal University*. **2014**, 37, pp. 88-92.
17. Loring, M. C.; Marron, M. D. Semantics of asynchronous JavaScript. *ACM SIGPLAN Notices*. **2017**, 52, pp. 51-62.
18. Amirante, A.; Castaldi, T.; Miniero, L.; Romano, S. P. On the seamless interaction between WebRTC browsers and SIP-based conferencing systems. *IEEE Commun. Mag.* **2013**, 51, 42-47. [[CrossRef](#)]
19. Johnston, A. B.; Burnett, D. C. WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web; DigitalCodex LLC: Lilburn, GA, USA, 2012.
20. Carlucci, G.; De Cicco, L.; Holmer, S.; Mascolo, S. Analysis and design of the google congestion control for web real-time communication (WebRTC). In Proceedings of the 7th International Conference on Multimedia Systems, Klagenfurt, Austria, 10-13 May 2016; pp. 1-12.

Figure 1 WebGIS System architecture

Figure2 Map layer model

Figure3 Tile numbering

Figure4 WebRTC communication protocol stack

Figure5 Design goal of WebGIS system based on edge computing

Figure 6 The principle of customization for differentiated needs

Figure 7 Customization and run of user systems

Figure 8 The Significance of Edge Computing for Sharing Map Tiles

Figure 9 The principle of edge maps sharing map tiles

Figure10 Architecture of WebGIS based on edge computing

Figure 11 Queuing model where the service unit is placed on the server

Figure 12 The queuing model of the service unit placed to the client

Figure 13 Relationship between the number of clients D and response time

Figure 14 Relationship between the number of service units K and response time

Figure 15 Relationship between request frequency λ and response time

Figure 16 Create a map service unit

Figure17 Map tile making

Figure 18 Map service unit operation result

Figure 19 Create a data editing unit

Figure20 Operation result of data editing unit

Figure21 Examples of user needs.

Figure22 The process of customization

Figure23 Establish a communication channel between nodes

Figure 24 RTCPeerConnection connection establishment timing

Figure 25 DATA_CHANNEL_OPEN datagram format

Figure26 Flow chart of tile transmission

Figure 27 Customized legend drawing WebGIS system

Figure28 Customized chart drawing WebGIS system

Figure29 Response time of the data editing service unit running on the server

Figure 30 Data editing unit, comparison of Trs and Trc at different access frequencies

Figure 31 Chart drawing unit, the relationship between Trs and Trc at different access frequencies.

Figure 32 Legend drawing unit, the relationship between the number of clients D and Trs , Trc

Figure 33 Comparison of server download mode and tile sharing mode

Figure 34 Comparison of server download and download sharing under different server bandwidths

Figures

Figure 1

WebGIS System architecture

Figure 2

Map layer model

Figure 3

Tile numbering

Figure 4

WebRTC communication protocol stack

Customized WebGIS of Unit A

Customized WebGIS of Unit B

Figure 5

Design goal of WebGIS system based on edge computing

Figure 6

The principle of customization for differentiated needs

Figure 7

Customization and run of user systems

Figure 8

The Significance of Edge Computing for Sharing Map Tiles

Figure 9

The principle of edge maps sharing map tiles

Figure 10

Architecture of WebGIS based on edge computing

Figure 11

Queuing model where the service unit is placed on the server

Figure 12

The queuing model of the service unit placed to the client

Figure 13

Relationship between the number of clients D and response time

Figure 14

Relationship between the number of service units K and response time

Figure 15

Relationship between request frequency λ and response time

Create a new Map object			
ID	map_1	Alignment	center
Map tile address:	userPath[x] [y] [z].png		
Baidu map tile:	☐ Road ☒ Satellite	Google map tile:	☐ Road ☒ Satellite
center latitude and longitude	(78, -131)	Zoom level	4
Width	500	Height	400
controls	☐ Big		Create object

Figure 16

Create a map service unit

Figure 17

Map tile making

Figure 18

Map service unit operation result

Create a new Data sheet

ID

Rows

Width

Cols

Height

Great

Figure 19

Create a data editing unit

	A	B	C	D	E	F	G	H
1	coordinates	Cost(yuan)	name					
2	(80.6, -140.0)	8650000	stadium					
3	(79.5, -116.8)	3620000	playground					
4	(79.8, -102.0)	790000	Basketball court					
5								

Figure 20

Operation result of data editing unit

Figure 21

Examples of user needs.

Figure 22

The process of customization

Figure 23

Establish a communication channel between nodes

Figure 24

RTCPeerConnection connection establishment timing

Bit	+0..7	+8..15	+16..23	+24..31
0	Message Type(0x3)	Channel Type	Priority	
32	Reliability			
64	Label length		Protocol length	
...	Label			
...	Protocol			

Figure 25

DATA_CHANNEL_OPEN datagram format

Figure 26

Flow chart of tile transmission

Figure 27

Customized legend drawing WebGIS system

Figure 28

Customized chart drawing WebGIS system

Figure 29

Response time of the data editing service unit running on the server

Figure 30

Data editing unit, comparison of Trs and Trc at different access frequencies

Figure 31

Chart drawing unit, the relationship between Trs and Trc at different access frequencies

Figure 32

Legend drawing unit, the relationship between the number of clients D and Trs, Trc The average value of Trc / Trs is 61.4%. So the total response time of the legend drawing unit running on the browser is 38.6% faster than the total response time running on the server.

Figure 33

Comparison of server download mode and tile sharing mode

Figure 34

Comparison of server download and download sharing under different server bandwidths