Received June 23, 2020, accepted July 4, 2020, date of publication July 8, 2020, date of current version July 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007942

Efficient, Customizable and Edge-Based WebGIS System

HONGLEI HE[1] AND WENMING ZHU[2]
[1]School of Information Engineering, Lianyungang Technical College, Lianyungang 222006, China
[2]Department of Public Courses, Shenzhen Institute of Information Technology, Shenzhen 518172, China

Corresponding author: Wenming Zhu (zhuwm@sziit.edu.cn)

This work was supported by the Lianyungang High-Tech Zone Science and Technology Project Plan under Project ZD201915.

ABSTRACT Nowadays there is a wide range of applications for WebGIS which can add great value to modern economic, and building WebGIS system for specific scenarios is the common requirement of the industry. While currently separate WebGIS systems are deployed at different sites and operated by different owners, each of which has the whole set of functionalities of WebGIS, and thus introduce high cost of development and maintenance, which is a waste of resources as most of the functionalities are the same or similar. An edge computing based WebGIS architecture is proposed in the paper to meet the need of customization by applying the idea of SaaS. In this distributed architecture, the resource load is reasonably balanced between the server and the browser, which improves the overall performance of the system. Also it utilizes edge computing to reduce the pressure on the server by sharing map tiles among WebGIS clients. The proposed WebGIS system can not only be well customized and personalized as it is edge computing base, but also be well usable for large number of visits due to its distributed feature. The experiments show 5 concurrent requests per second, as well as response speed increases by more than 38.6% against traditional deployment.

INDEX TERMS Edge computing, customizable, WebGIS, high performance, map tile sharing, device-enhanced MEC.

I. INTRODUCTION

WebGIS has achieved amazing development in recent years and is widely used in almost all fields of the nation, e.g. resource transportation, environmental assessment, disaster prediction, terrain management, urban planning, post and telecommunications, state security, water conservancy and power, public facilities management, agriculture, forestry and animal husbandry, statistics and so on [1]–[3].

In terms of resource management, WebGIS is mainly used in agriculture and forestry to solve 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, WebGIS is mainly used in regional ecological planning, environmental status assessment, environmental impact assessment. It also provides decision support for pollutant reduction, sustainable region development, environmental protection facility management, environment planning, etc [5].

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

In network analysis, WebGIS aims to establish computer models of transportation networks, underground pipeline networks for traffic flow study, traffic rule conduction, and emergency services.

In terms of visualization applications, WebGIS 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 and simulation, tourism and other fields.

Therefore, there is strong need to apply WebGIS system in different fields of economy and society. At present, such GIS system is commonly developed based on some standalone platform (such as ARCGIS, mapinfo) [6], [7],

The associate editor coordinating the review of this manuscript and approving it for publication was Rongbo Zhu.

and professional IT team is also required to complete the application-specific development, which introduces significant budget burden for most users, especially those non-core business users who does not frequently use it. Moreover, as most functionalities in different WebGIS products are the same or similar, investment to this kind of development loop is also a meaningless waste of resources.

To avoid resource waste, as well as reduce development and maintenance cost, new WebGIS architecture is required to meet the requirements of scalability, flexibility, customizability and availability.

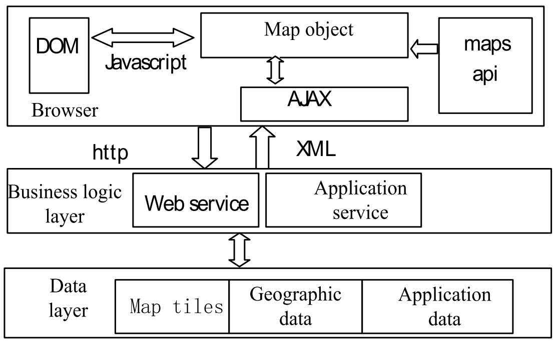


FIGURE 1. WebGIS System architecture.

II. RELATED TECHNOLOGIES

A. OVERVIEW OF WebGIS

The current mainstream WebGIS technology is a hierarchical tiled map which was represented by Google and Baidu. The architecture is composed 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 also responds to various user actions such as zooming, dragging, and marking [8]–[11]. The server provides services such as WMS, WTS, WFS. The Client-Server architecture was designed to reduce the server load and provide compatibility across browsers [12], [13].

1) MAP LAYER MODEL

The map displayed by WebGIS on the browser is divided into multiple levels, as shown in Figure 2. One can edit the designed map by adding HTML elements (e.g. text, pictures, links, or videos) in any of these layers, or locate specified geographic coordinates [14]–[17].

2) MAP TILES

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 [18], [19]. 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

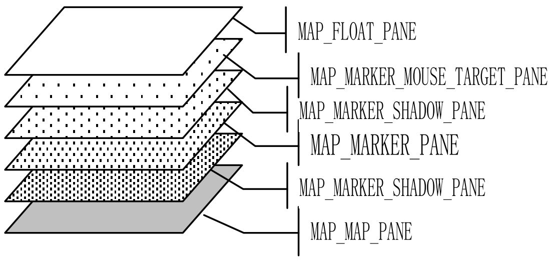


FIGURE 2. Map layer model.

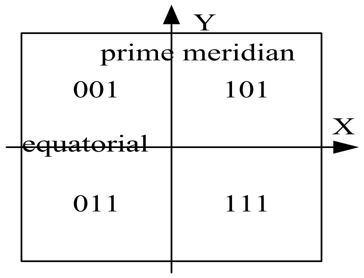


FIGURE 3. Tile numbering.

(z = 1) world map is divided into 4 pieces, and the tile numbers xyz are 001, 101, 011, and 111 (Figure 3).

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

$$\mathbf{X} = \left[2^{Z-1} \cdot \left(\frac{\lambda}{180} + 1 \right) \right] \tag{1}$$

$$\mathbf{Y} = \left[2^{Z-1} \cdot \left(1 - \frac{\ln \left[\tan \left(\frac{\pi \cdot \varphi}{180} \right) + \sec \left(\frac{\pi \cdot \varphi}{180} \right) \right]}{\pi} \right) \right] \tag{2}$$

B. SaaS MODEL OF CLOUD COMPUTING

There are three modes of cloud computing, namely IaaS, PaaS, and SaaS, among which SaaS model provides high level of self-customization or user-defined functions under the multi-tenancy architecture. In SaaS, users can customize fields, menus, reports, views, workflows, etc. according to their own business requirements, so that they can be tailored to SaaS software even though they do not have much programming skill [20], [21].

When WebGIS is concerned, applying SaaS on the system enables customization of the interface, business logic, data structure, as well as service level w.r.t. different user needs or membership.

In traditional SaaS model, all of the customization jobs are done in the cloud as all data from different users located in different places are transmitted to the cloud for batch processing, which leads to relatively heavy server load and heavy network traffic burden [22].

On the contrary, if all the user-centric process of WebGIS can be done near the user by utilizing edge computing, both the cloud load and traffic burden can be efficiently reduced [23]–[27]. This is also the basic idea of this paper: all the context-related processing modules are deployed as near the user side (edge node) as possible, while other system-related functionalities stay on the cloud side. The edge node collects and processes data directly from the users nearby, and also provides local services to local users, without unnecessary communication between with the cloud center. In this autonomic distribution system, each user terminal (edge node) can establish a communication channel to achieve resource sharing and improve the processing speed of the system.

C. WebRTC

The communication protocol stack of WebRTC is shown in Figure 4 [28]–[31]. 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. So choose UDP to transmit voice and video data, and choose TCP to transmit signaling data for control functions [32]–[35].

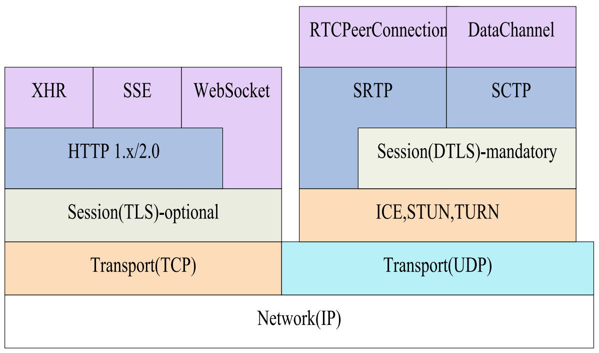


FIGURE 4. **WebRTC communication protocol stack.**

D. DEVICE-ENHANCED MEC

Any network entity on the path between data source and the cloud center can act as an edge computing node, which has the capability of computing, storage and resource sharing [36]–[40]. The edge node is able to perform data processing in near real-time as compared with cloud computing [41]–[43]. In a traditional distribution, the edge computing node is typically provided by the service operator, while more and more user terminals, which is closer to the data source and also has some level of computing capabilities, are used as edge nodes.

With the rapid development of smart devices, IoT (the Internet of Things) and 5G technologies, edge computing, as well as the multi-access edge computing (MEC) technology, gains more and more attraction. And device-enhanced MEC, which makes full use of the computing power and storage capacity of the terminal, was proposed as traditional edge computing nodes lacks of resource computing efficiency and service flexibility [44].

The WebGIS system in this paper refers to the principle of device-enhanced MEC by enabling user terminals as edge nodes for connection establishment [45]–[49], local computing and resource (e.g. map tile) sharing with each other.

III. DESIGN OF WebGIS SYSTEM BASED ON EDGE COMPUTING

A. DESIGN GOALS

The goal is to propose a WebGIS architecture on the basis of edge computing (as shown in Figure 5). Users can customize WebGIS features through SaaS cloud without building another standalone system, and the availability and flexibility of the network can also be better achieved by using various distributed light-weighted edge nodes.

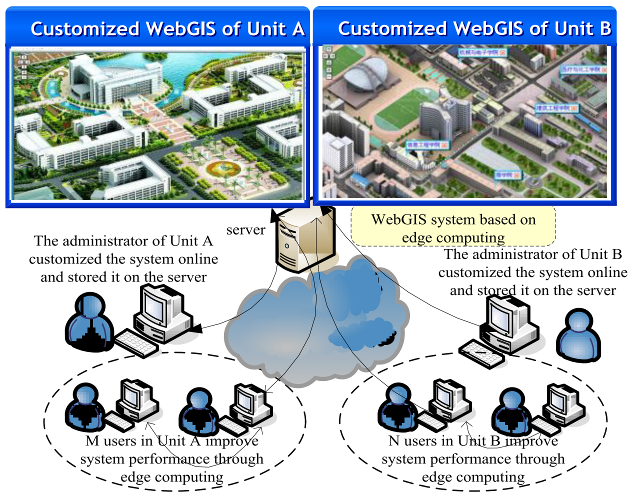

Customized WebGIS of Unit A
Customized WebGIS of Unit B
server
WebGIS system based on edge computing
The administrator of Unit A customized the system online and stored it on the server
The administrator of Unit B customized the system online and stored it on the server
M users in Unit A improve system performance through edge computing
N users in Unit B improve system performance through edge computing

FIGURE 5. **Design goal of WebGIS system based on edge computing.**

B. SYSTEM CUSTOMIZABLE DESIGN

Customizability is one aspect of the design. User interface, business logic, and data structure should be highly customizable, via graphical interface, according to different requirements.

Two kinds of service units, which are driven by common and personalized requirements respectively, are classified as shown in Figure 6, where one or more of the common service units are designed on the cloud server side, in the form of Web services, and the user specific units are implemented on the client side (edge node).

The functionality of common service units includes data access management, user authentication and authorization, etc., and the functions of user specific units include legend service, data statistics, and vehicle positioning etc.

All the service units are designed loosely coupled and are relatively independent of each other to achieve the goal of customization.

In practical applications (Figure 7), the users are able to select the service units to build their own WebGIS apps according to different requirements [50]–[52].

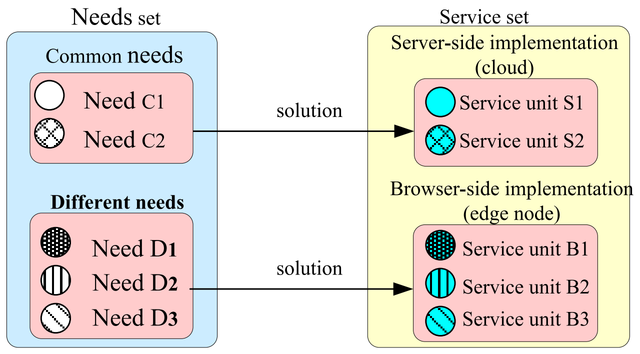


FIGURE 6. The principle of customization for differentiated needs.

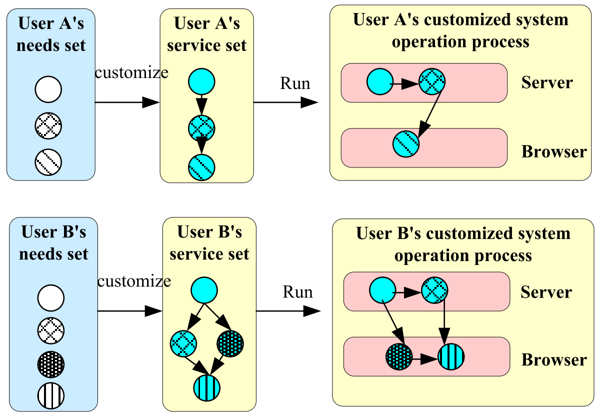


FIGURE 7. Customization and run of user systems.

C. TILE SHARING DESIGN

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

1) THE SIGNIFICANCE OF TILE SHARING

Assuming there are K (K>0) instances of customized WebGIS app running on the network.

Definition 1: N_i^{tile} (1<=i<=K) denotes the number of map tiles on each WebGIS instance.

Definition 2: N_i^{user} (1 <= i <= K) denotes the number of users who are updating map tiles on each instance.

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

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

As the number of users and WebGIS instances scales up, the load of the server will increase rapidly, and the loading speed of user maps decreases and the user experience degrades.

To solve the problem, edge based tile sharing is proposed to balance the server load and achieve cost efficiency without upgrading server configuration or increasing server number, which is shown in Figure 8.

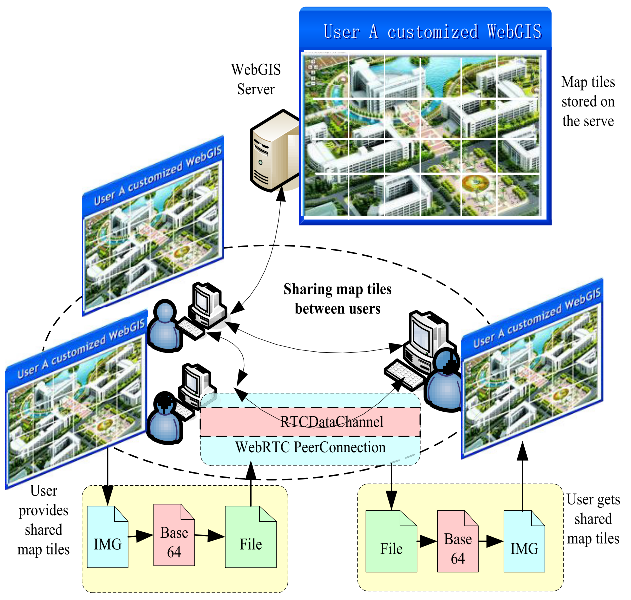


FIGURE 8. The significance and principle of sharing map tiles by edge computing.

2) PRINCIPLES OF MAP TILE SHARING

The principle of map tile sharing is shown in Figure8. Each user terminal can act as an edge node, and communication connections are established straightly between the nodes for map tile sharing, rather than each node downloading tiles from the server respectively.

For example, multiple WebGIS instance may share the map tiles with other simultaneous instances once it downloads tiles from the server, such that only one download session with cloud server is needed.

The sharing connection is established via peer-to-peer protocols, e.g. WebRTC PeerConnection for signaling channel and WebRTC Datachannel for tile data transmission.

As RTCDatachannel can only transmit text format [53], the IMG-format map tiles are converted to Base64 before essential transmission take place. The peer also need to convert the Base64 payload back to IMG format once it receives the packets.

D. ARCHITECTURE DESIGN

The overall architecture of edge computing based WebGIS is shown in Figure 9. It includes Presentation Layer, Service Layer and Data Layer.

The Presentation Lay responds of user interactions and graphical rendering on the browser side, and the Service Layer contains two components, i.e., server-side and browser-side client service, which enables full customizability of the services.

The customization, including user interface, service logic, and data structure, can be performed on browser, metadata

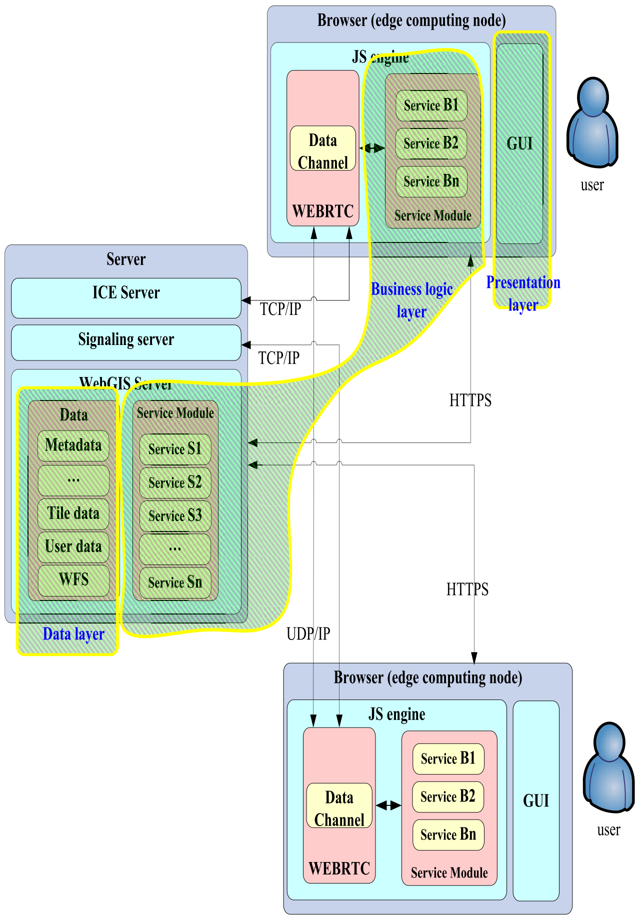


FIGURE 9. Architecture of WebGIS based on edge computing.

and server respectively. The customized WebGIS instance is stored as the server-side metadata for later maintenance.

For security purpose, ICE/STUN server may be involved when WebRTC sharing session is concerned [54], hence TCP protocol may be used for signaling and UDP protocol is used for map tile data transmission.

E. 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. As the performance of models relies on whether the service unit are implemented on server side or client side, which is compared as follows.

1) SERVICE UNITS ON SERVER SIDE

All client requests need to be queued for processing by 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 service rate is μ, 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 / ∞/∞/ FCFS.

Definition 4: E_i^c denotes the computing capability of i-th client, where $1 \le i \le D$.

Definition 5: S_y denotes the y-th service unit on the server, where $1 \le y \le K$.

Definition 6: w_i^y denotes the workload on the server generated by the call from the client i to the service unit S_y.

Definition 7: E_s denotes the computing capability of the server.

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

Denote G as the overall workload of all service requests arriving in a unit time, shown in equation (4).

$$G = \sum\nolimits_{y=1}^{K} \sum\nolimits_{i=1}^{D} \lambda_i^y \cdot W_i^y \tag{4}$$

Denote N as the overall number of service requests arriving per unit time, as equation (5).

$$N = \sum\nolimits_{y=1}^{K} \sum\nolimits_{i=1}^{D} \lambda_i^y \tag{5}$$

Denotes T as the average processing time of each service request, $T = G / (N * E_s\,)$.

Denote λ the arrival rate of the service request, $\lambda = N$.

Denote μ as the speed of service processing, $\mu = 1 / T$.

Denote ρ as the probability of at least one service request in the system, $\rho = \lambda / \mu$.

Denote L_s as the average queue length in the system is expressed as, $L_s = \rho/$ (1-ρ).

Denote L_q as the average number of customers waiting for service in the system, $L_q = L_s / \rho$.

Denote Wq as the average waiting time of each service request, Wq = Lq / λ, we can get equation (6).

$$\begin{aligned} W_q &= \frac{\lambda}{\mu^2 - \lambda\mu} \\ &= \frac{\left(\sum_{y=1}^{K} \sum_{i=1}^{D} \left(\lambda_i^y \cdot W_i^y\right)\right)^2}{E_s \cdot \sum_{y=1}^{K} \sum_{i=1}^{D} \lambda_i^y \left(E_s - \sum_{y=1}^{K} \sum_{i=1}^{D} \left(\lambda_i^y \cdot W_i^y\right)\right)} \end{aligned} \tag{6}$$

For the customized system of user i, denote N_i the number of requests sent per unit time, as shown in equation (7).

$$N_i = \sum\nolimits_{y=1}^{K} \lambda_i^y \tag{7}$$

Denote G_i as the requested workload per unit time i-th users instance, shown as equation (8).

$$G_i = \sum\nolimits_{y=1}^{K} \left(\lambda_i^y \cdot W_i^y\right) \tag{8}$$

Denote $\mathrm{T_i^s}$ as the total running time of the customized i-th user's instance, shown as equation (9).

$$T_i^s = N_i \cdot W_q + \frac{G_i}{E_s} \tag{9}$$

According to equations (6) (7 (8) (9), equation (10) can be obtained.

$$T_i^s = \frac{\sum_{y=1}^{K} \lambda_i^y \left(\sum_{y=1}^{K} \sum_{i=1}^{D} (\lambda_i^y \cdot W_i^y)\right)^2}{E_s \cdot \sum_{y=1}^{K} \sum_{i=1}^{D} \lambda_i^y \left(E_s - \sum_{y=1}^{K} \sum_{i=1}^{D} (\lambda_i^y \cdot W_i^y)\right)} + \frac{\sum_{y=1}^{K} (\lambda_i^y \cdot W_i^y)}{E_s} \tag{10}$$

2) SERVICE UNIT ON USER SIDE

The service units are all placed on the user side for implementation The system can be regarded as a queuing model of M / M / 1 / ∞/∞/ FCFS.

Denote T_i^c as the system running time of i-th user. According to equation (10), equation (11) can be obtained.

$$T_i^c = \frac{\sum_{y=1}^{K} \lambda_i^y \left(\sum_{y=1}^{K} (\lambda_i^y \cdot W_i^y)\right)^2}{E_i^c \cdot \sum_{y=1}^{K} \lambda_i^y \left(E_i^c - \sum_{y=1}^{K} (\lambda_i^y \cdot W_i^y)\right)} + \frac{\sum_{y=1}^{K} (\lambda_i^y \cdot W_i^y)}{E_i^c} \tag{11}$$

3) COMPARISON ON RESPONSE TIME

In the actual operation of user i's system, the transmission delay must also be considered. The transmission delay is expressed as $\mathrm{T}_i^{\mathrm{delay}}$. The network delay is expressed as $\mathrm{T}_i^{\mathrm{network}}$. The amount of data transferred is expressed as DTi. The transmission bandwidth is expressed as BW. Then there is equation (12).

$$T_i^{delay} = T_i^{network} + \frac{DT_i}{BW} \tag{12}$$

If the system is running on the server, the response time of the system is expressed as $\mathrm{T}_i^{\mathrm{rs}}$, $\mathrm{T}_i^{\mathrm{rs}} = \mathrm{T}_i^{\mathrm{s}} + \mathrm{K} \cdot \mathrm{T}_i^{\mathrm{delay}}$. $\mathrm{T}_i^{\mathrm{rs}}$ see equation (13).

$$T_i^{rs} = K \cdot T_i^{delay} + \frac{\sum_{y=1}^{K} \lambda_i^y \left(\sum_{y=1}^{K} \sum_{i=1}^{D} (\lambda_i^y \cdot W_i^y)\right)^2}{E_s \cdot \sum_{y=1}^{K} \sum_{i=1}^{D} \lambda_i^y \left(E_s - \sum_{y=1}^{K} \sum_{i=1}^{D} (\lambda_i^y \cdot W_i^y)\right)} + \frac{\sum_{y=1}^{K} (\lambda_i^y \cdot W_i^y)}{E_s} \tag{13}$$

If the system is running on the user side, the response time of the system is expressed as $\mathrm{T}_i^{\mathrm{rc}}$. Because the transmission delay is 0, $\mathrm{T}_i^{\mathrm{rs}} = \mathrm{T}_i^{\mathrm{c}}$. According to equation (11), equation (14) can be obtained.

$$T_i^{rc} = \frac{\sum_{y=1}^{K} \lambda_i^y \left(\sum_{y=1}^{K} (\lambda_i^y \cdot W_i^y)\right)^2}{E_i^c \cdot \sum_{y=1}^{K} \lambda_i^y \left(E_i^c - \sum_{y=1}^{K} (\lambda_i^y \cdot W_i^y)\right)} + \frac{\sum_{y=1}^{K} (\lambda_i^y \cdot W_i^y)}{E_i^c} \tag{14}$$

Suppose $Es = 5E_i^c$, Es = 1000W, $\lambda_i^y = 10$, k = 2, $\mathrm{T}_i^{\mathrm{network}} = 0.05$ (s), DTi = 2 KB, BW = 1 Mbps. The relationship between the number D of clients and the response time Trs of the system running on the server and the time Trc of the system running on the client is shown in the figure 10.

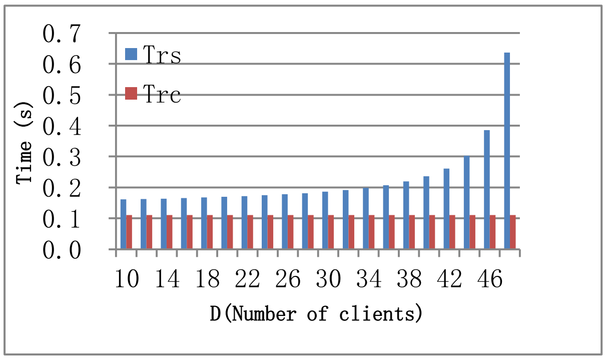


FIGURE 10. Relationship between the number of clients D and response time.

Suppose $Es = 5E_i^c$, Es = 1000W, $\lambda_i^y = 10$, D = 10, $\mathrm{T}_i^{\mathrm{network}} = 0.05$ (s), DTi = 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 11.

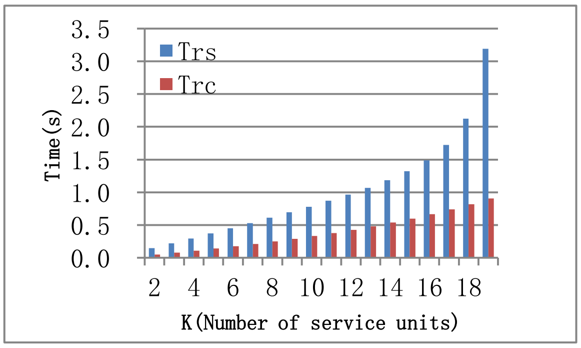


FIGURE 11. Relationship between the number of service units K and response time.

Suppose $Es = 5E_i^c$, Es = 1000W, K = 2, D = 10, $\mathrm{T}_i^{\mathrm{network}} = 0.05$ (s), DTi = 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 12.

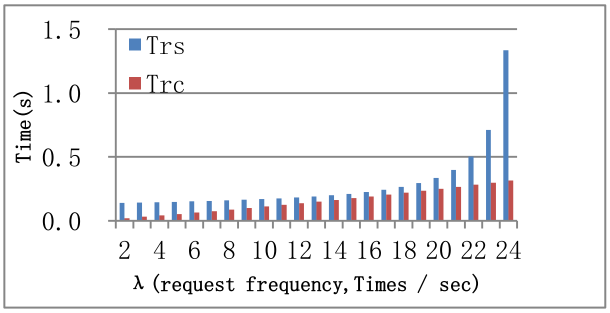


FIGURE 12. Relationship between request frequency λ and response time.

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.

IV. IMPLEMENTATION

A. IMPLEMENTATION OF CUSTOMIZABLE SYSTEM

1) IMPLEMENTATION OF THE SERVICE UNIT

Various use cases result in various requirements for WebGIS, including user management, geographic information management, map management, type switching, data editing, mapping, timer, data search, data customization, etc. Among these requirements some are common used for all services, and others are application specific.

User management, data access, etc. are the common requirements, and corresponding service units should be implemented on the server side (e.g. via ASP.net).

Requirements, such as map services, data editing, dashboard etc., are user specific, and should be implemented on the browser side (e.g. via JavaScript) [55]–[58].

As an example, the Map Unit and Data Editing Unit can be implemented as follows.

a: MAP SERVICE UNIT

The user specific Map Service Unit can be generated after the parameters are applied to the system (e.g. via a form submission). The customized parameters includes the tiles, controls, initial geographic coordinates, width and height of the map objects.

For example, the user may have the option to use public tils (e.g google map, baidu map) or private tiles. In private option the user specific tiles should be generated and numbered and upload to the server.

As the result of map tile customization, the front-end HTML and JavaScript code contains all of the map objects. The Google map is referenced in our code implementation, while other framework such as Openlayers may also be used in practical applications.

b: DATA EDITING UNIT

The user may define a service oriented Data Editing Unit via another form. Parameters such as ID, height, width and initial number of rows and columns of the data editing can be highly customized.

The corresponding HTML and JavaScript codes are automatically generated once the Data Editing Unit is created. The HTML code is an internal frame $<$ iframe $>$, and the JavaScript is a Handsontable plugin using Jquery.

2) CUSOMIZED VISUALISATION

As an example, suppose a WebGIS user interface includes map visualization, data editing, dashboard and data storage, as shown in Figure 13. A visual editor app is implemented to enable such customization shown in Figure 14.

The insertion of forms and other controls is done via the toolbar, and then the corresponding HTML/CSS code is generated in the editing area, which decides the layout of the user interface.

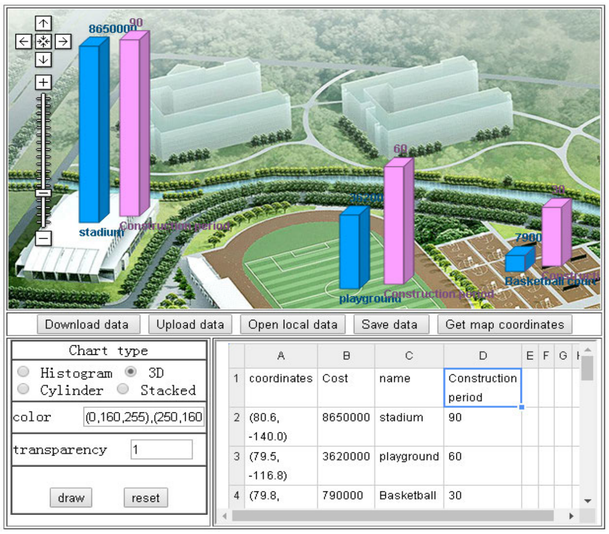
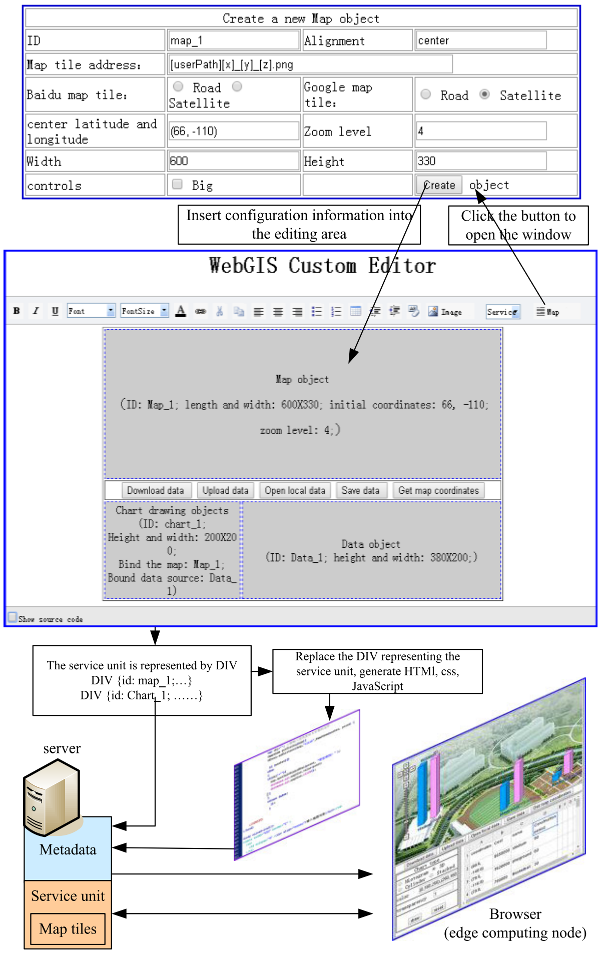


Chart type	
( ) Histogram [✓ 3D] ( ) Cylinder ( ) Stacked	
color	(0,160,255),(250,160
transparency	1

	A	B	C	D	E	F	G
1	coordinates	Cost	name	Construction period			
2	(80.6, -140.0)	8650000	stadium	90			
3	(79.5, -116.8)	3620000	playground	60			
4	(79.8,	790000	Basketball	30			

FIGURE 13. Examples of user needs.



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	(66, -110)	Zoom level	4
Width	600	Height	330
controls	☐ Big		Create object

WebGIS Custom Editor

Map object
(ID: Map_1; length and width: 600X330; initial coordinates: 66, -110; zoom level: 4;)

Chart drawing objects
(ID: chart_1;
Height and width: 200X200;
Bind the map: Map_1;
Bound data source: Data_1)

Data object
(ID: Data_1; height and width: 380X200;)

Insert configuration information into the editing area
Click the button to open the window
B I U Font FontSize Image Service Map
Download data
Upload data
Open local data
Save data
Get map coordinates
Show source code
The service unit is represented by DIV
DIV {id: map_1;…}
DIV {id: Chart_1; ……}
Replace the DIV representing the service unit, generate HTMl, css, JavaScript
server
Metadata
Service unit
Map tiles
Browser
(edge computing node)

FIGURE 14. The process of customization.

Other objects, such as map objects, database objects and charts etc., may also be inserted and bound with each other, so as to achieve service customization.

In our case, all the service units, and binding of units are implemented in JavaScript code, which is stored as server metadata later on. Related HTML and CSS is also stored after appropriate editing, and they will be retrieved by the browser for rendering during the active period of the running instance.

B. IMPLEMENTATION OF MAP TILE SHARING

The map tiles are shared among the nodes via WebRTC, including handshakes, information broadcasting and request/ response handling, data transmission etc.

1) ESTABLISH A COMMUNICATION CHANNEL BETWEEN NODES

Before the data sharing takes place, a signaling connection should be firstly established between the browsers (edge nodes), as shown in Figure 15.

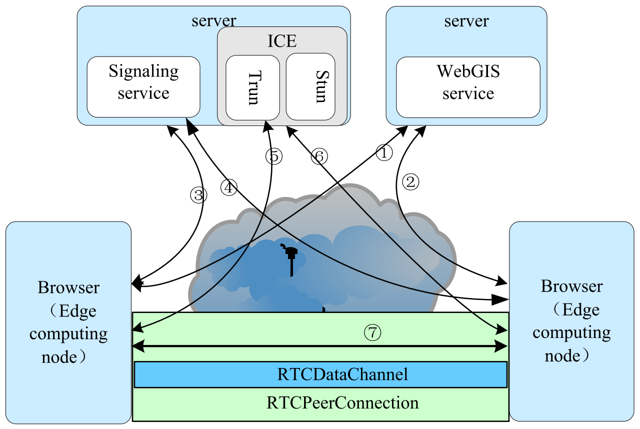


FIGURE 15. Establish a communication channel between nodes.

a: WORKING PROCESS OF COMMUNICATION MODULE

(1) Two browsers (edge nodes) are first registered (and sign in) to WebGIS server as steps ① ②.

(2) The browsers also sign in the signaling server to obtain their identification respectively, as steps ③④.

(3) The initiator send out a RTCPeerConnection Offer and the receiver will send back an Answer via the RTCPeer-Connection. The parties also negotiate ICE parameters, as steps ⑤ ⑥.

(4) The RTCPeerConnection connection is established, and then a RTCDataChannel instance is created for data transmission, as step ⑦.

b: CREATION OF RTCPeerConnection

When a browser initiates a real-time communication connection call, the sequence of establishing an RTCPeerConnection connection is shown in Figure 16.

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

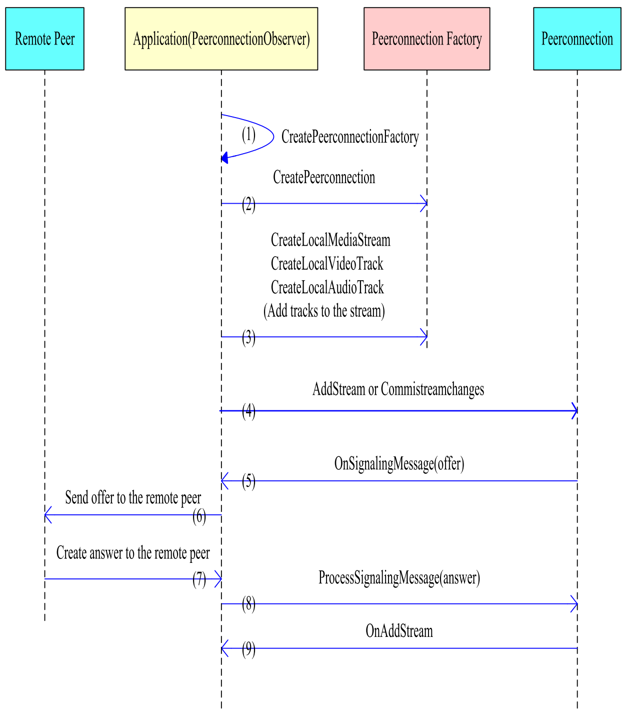

Remote Peer
Application(PeerconnectionObserver)
Peerconnection Factory
Peerconnection
(1) CreatePeerconnectionFactory
CreatePeerconnection
(2)
CreateLocalMediaStream
CreateLocalVideoTrack
CreateLocalAudioTrack
(Add tracks to the stream)
(3)
AddStream or Commistreamchanges
(4)
OnSignalingMessage(offer)
(5)
Send offer to the remote peer
(6)
Create answer to the remote peer
(7)
ProcessSignalingMessage(answer)
(8)
OnAddStream
(9)

FIGURE 16. RTCPeerConnection connection establishment timing.

(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, or submit stream changes.

(5)The PeerConnection instance connected to the calling party generates an Offer.

(6)The application sends the offer to the remote callee through a signaling service after receiving the Offer.

(7) After receiving the Offer, the remote callee generates Answer and sends it to the application program through the signaling service.

(8)The application transfers the Answer 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.

(9) When the local PeerConnection instance receives the media stream from the other party, it is handed over to the application for processing.

c: 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 17.

First, you must first create a RTCPeerconnection to establish a connection. Then create a DataChannel object on this RTCPeerconnection. Multiple DataChannel objects can

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

FIGURE 17. DATA_CHANNEL_OPEN datagram format.

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.

When we transfer map tiles between browsers, the established peerconnction 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 \ no =-8335070345506316053 2 IN IP4 127.0.0.1 \ r \ ns =-\ r \ nt = 0 0 \ r \ na = msid- semantic: WMS \ r \ nm = application 9 DTLS / SCTP 5000 \ r \ nc = IN IP4 0.0.0.0 \ r \ nb = AS: 30 \ r \ na = ice-ufrag: PxEn \ r \ na = ice-pwd: ib3DQhicOqJWFA9GyinV0SXn \ r \ na = ice-options: trickle \ r \ 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 \ na = setup: active \ r \ na = mid: data \ r \ na = sctpmap: 5000 webrtc-datachannel 1024 \ r \ n”}

2) TRANSMISSION OF MAP TILES

The transmission of map tiles includes downloading and sharing, as shown in Figure 18.

For instance, node A calculates the geographic coordinates and decides which tile to used, then it sends a tile download request to the server by default. If the server load is low enough (under some threshold), the server may directly reply to the node with the requested tile data, otherwise it notifies the node to retrieve data from other nodes.

In the latter case, node A sends a tile request broadcast to the whole network, the broadcast content contains the tile number. Any candidates that contain the requested tile will respond back to node A, and some node B will be selected

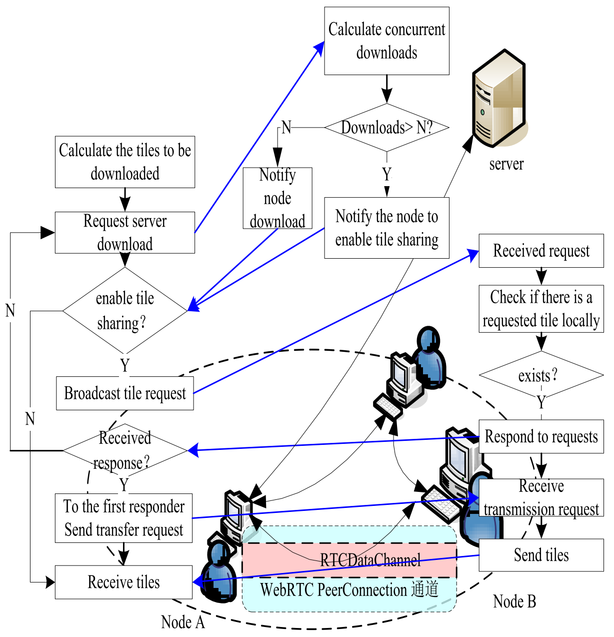

Calculate concurrent downloads
Downloads> N?
N
Y
server
Calculate the tiles to be downloaded
Request server download
Notify node download
Notify the node to enable tile sharing
enable tile sharing?
N
Y
Received request
Check if there is a requested tile locally
Broadcast tile request
exists?
Y
N
Received response?
Y
Respond to requests
Receive transmission request
To the first responder Send transfer request
Send tiles
RTCDataChannel
Receive tiles
WebRTC PeerConnection 通道
Node A
Node B

FIGURE 18. Flow chart of tile transmission.

for sharing as its response arrives first. Then node A setup a WebRTC session with node B and initiates data transmission.

If node A does not receive any response from the nodes in time (e.g. in 300 ms), it attempts to download from the server.

The map tile data should be converted from IMG format to text format for better RTCDataConncetion transmission.

V. RESULTS AND DISCUSSION

There are 2 Alibaba Cloud ECS Servers and 4 clients in our experimental environment. The configuration of the cloud servers are (respectively): 2-core CPU, 8G memory, 5Mbps downstream bandwidth, with Windows Server 2008 R2 64-bit Enterprise Edition OS.

One server is implemented as a WebGIS server, and the other is the WebRTC and STUN server.

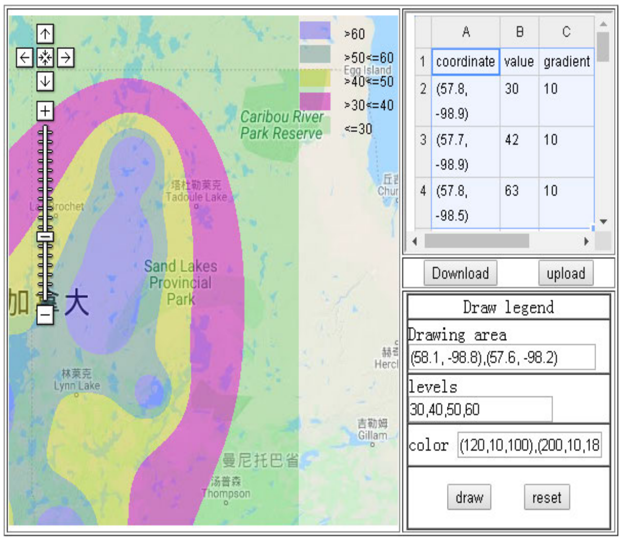
The clients are PCs, which has the configuration (respectively): Intel core i5-6200U 2.3GHz CPU, 4GB memory, 500G hard disk, and with Windows 7 OS. The browser used in the experiment is Chrome 71.0.3578.98. The bandwidth between each browser node is 10Mbps.

A. CUSTOMIZABLE TEST

The WebGIS system is highly customized due to the flexibility nature of service units.

For example, a WebGIS legend instance can be built by using {map service, data editing service, legend drawing service, data access service} configurations, as shown in Figure 19.

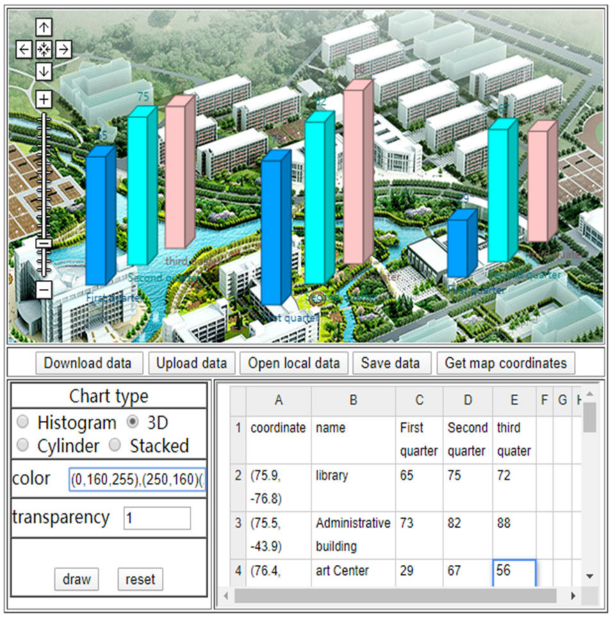
Another example is the WebGIS chart instance with configurations of {map service, data editing service, chart service, data access service, coordinate reading service}, as shown in Figure 20.



	A	B	C
1	coordinate	value	gradient
2	(57.8, -98.9)	30	10
3	(57.7, -98.9)	42	10
4	(57.8, -98.5)	63	10

Drawing area (58.1, -98.8),(57.6, -98.2)
levels 30,40,50,60
color (120,10,100),(200,10,18

FIGURE 19. Customized legend drawing WebGIS system.



Histogram [✓ 3D]
Cylinder Stacked
color (0,160,255),(250,160)(
transparency 1

	A	B	C	D	E	F	G	H
1	coordinate	name	First quarter	Second quarter	third quater			
2	(75.9, -76.8)	library	65	75	72			
3	(75.5, -43.9)	Administrative building	73	82	88			
4	(76.4,	art Center	29	67	56			

FIGURE 20. Customized chart drawing WebGIS system.

B. PERFORMANCE TEST

In our experiment, service units of data editing, chart drawing, legend drawing are selected for testing, each of which includes server-side and browser-side performance 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 21.

It is obvious that as the access frequency increases, the response time will increase rapidly when the server is of almost full load.

On the other way, JavaScript test is performed when the data editing unit runs on the browser side. and the response time of each data edit (add, delete) is less than 1ms.

Assume each data editing operation includes 20 atomic operations such as data addition, deletion, and modification etc., then server-side service unit relates 20 requests while client-side service unit only relates 1 request.

Also s uppose there are 60 clients. According to Table 1, the frequency of user data editing Ru (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 The relationship of time Trs is shown in Figure 22.

According to the table, 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:

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

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

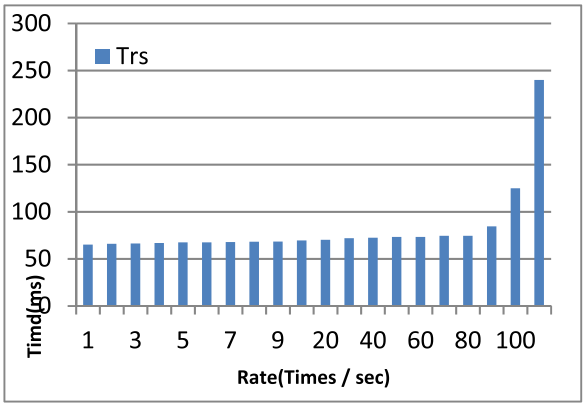


FIGURE 21. Response time of the data editing service unit running on the server.

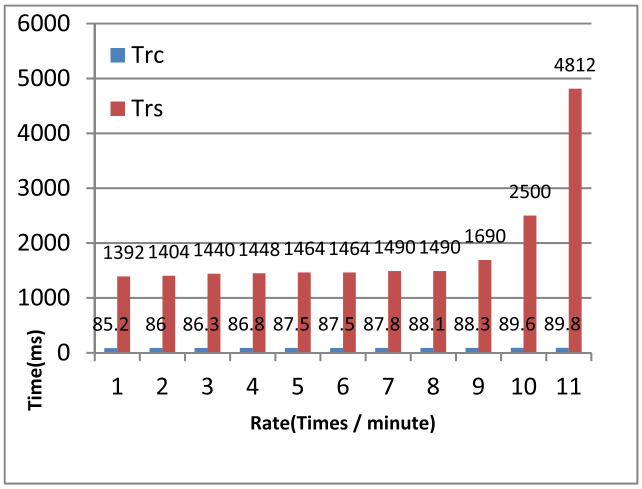


FIGURE 22. Data editing unit, comparison of Trs and Trc at different access frequencies.

2) CHART DRAWING UNIT

Http_load test is performed when the charting service unit is on the server side, which includes 10000 data accumulation calculations and 500K data transmission (simulating the file size of the drawn picture). The response time is expressed in Trs.

JavaScript test is performed when the charting unit runs on the browser side, which includes 10000 data accumulation calculations. The response time is expressed in Trc.

Suppose there is only 1 client. The relationship between Trs and Trc is shown in Figure 23 under different access frequencies R (times / sec).

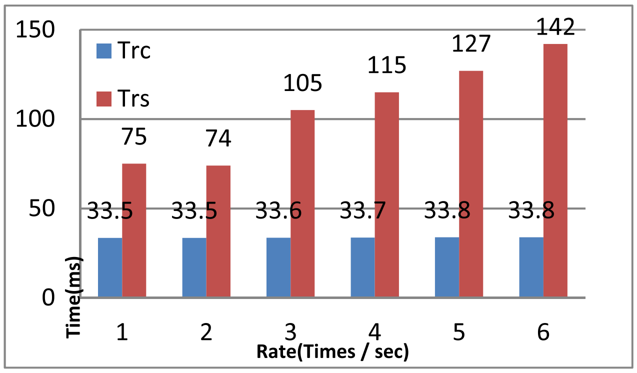


FIGURE 23. Chart drawing unit, the relationship between Trs and Trc at different access frequencies.

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

Http_load test is performed when the legend drawing unit is on the server side, which includes 1000000 data accumulation calculations and 500K data transmission (simulation drawing 1000 $*$ 1000 pixels legend). The response time is expressed in Trs.

JavaScript test is performed when the legend drawing unit runs on the browser side, which includes 1,000,000 data accumulation calculations. 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 24.

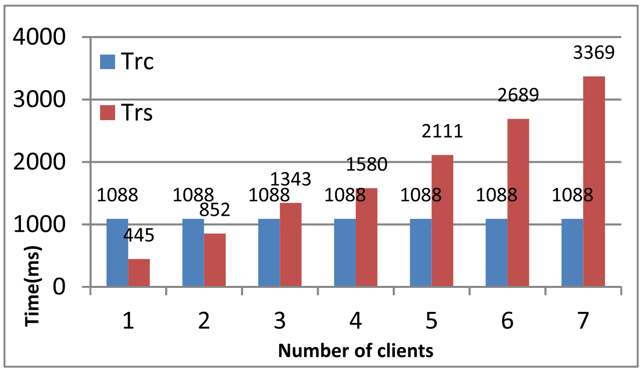


FIGURE 24. Legend drawing unit, the relationship between the number of clients D and Trs, Trc.

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.

C. 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, the time for new client to download tiles is significantly reduced when multiple WebGIS instances are online, i.e., more client participation means faster speed, as shown in Figure 25.

From Table 2, it is obvious that our tile sharing method takes full advantages and is more bandwidth efficiency when the server bandwidth is constrained, as shown in Figure 34.

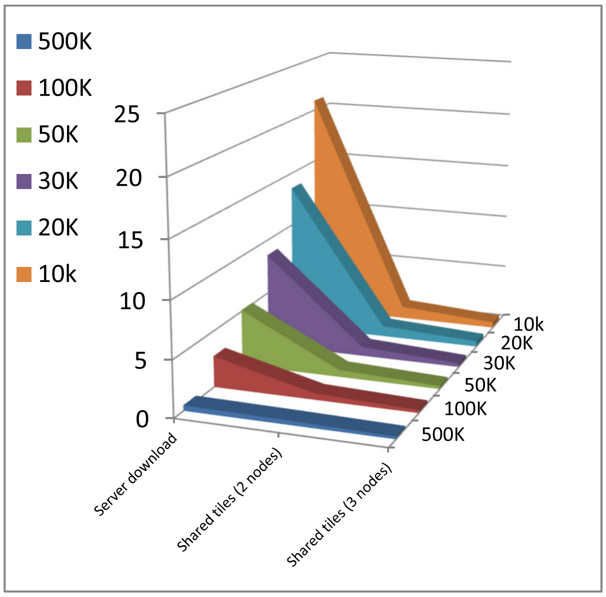


FIGURE 25. Comparison of server download mode and tile sharing mode.

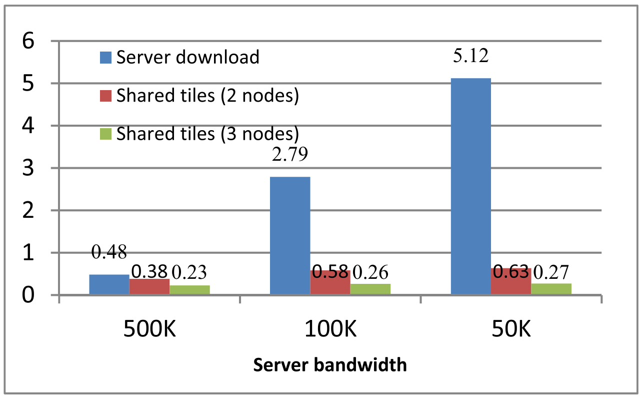


FIGURE 26. Comparison of server download and download sharing under different server bandwidths.

Figure 26 shows that even in the case of 500Kbps bandwidth constraint, the download speed of the tile sharing increases by more than 50% if there are more than 3 sharing clients.

It can be seen from the experimental results that the map tile sharing technology greatly improves the performance of the system as it is very similar to device-enhanced MEC and ad-hoc cloud computing. Since the client is closer to data source and also have computing and storage capabilities, it can act as nodes for edge computing, is an important supplement to MEC or cloud computing resources. When the cloud is overloaded, all nearby clients establish connections through ad-hoc cloud and share computing and storage resources.

VI. CONCLUSION

Based on the requirements from the industry, an edge computing based WebGIS architecture is proposed and implemented aiming the goal of high customizability, usability, flexibility and scalability compared to traditional vertical WebGIS systems.

The customizability is obtained by classified requirement analysis and decomposition, from which the common service units and user-specific service units are derived and designed carefully on the server side or client side with the consideration of server load and traffic optimization.

Also the flexibility and scalability of the system are satisfied by utilizing peer-to-peer protocol between clients for map tile sharing, which avoids single point of failure on the server side. From experimental testing, it is concluded that the overall response speed on distributed edge infrastructure increases by at least 38.6% against traditional cloud computing.

As for the usability, various visualization technologies are taken into account when the WebGIS system is designed and implemented, so as to enable the user to customize the instance through a graphical interface.

The last but not the least, map tile sharing among the clients by utilizing device-enhanced MEC technology not only reduces the server burden, but also supports large concurrent visits. Experiments also shows the download speed for map tiles improves significantly in the case of multiple client participation.

ACKNOWLEDGMENT

The testing is done in Information Center of Lianyungang Vocational and Technical College. The authors would like to thank colleague Yuxuan He from Xuzhou University of Technology for her hospitality, time and opinions during the research.

REFERENCES

[1] R. Lathrop, L. Auermuller, J. Trimble, and J. Bognar, "The application of WebGIS tools for visualizing coastal flooding vulnerability and planning for resiliency: The New Jersey experience," *ISPRS Int. J. Geo-Inf.*, vol. 3, no. 2, pp. 408–429, 2014.

[2] R. G. Howell, S. L. Petersen, C. S. Balzotti, P. C. Rogers, M. W. Jackson, and A. E. Hedrich, "Hedrich,using WebGIS to develop a spatial bibliography for organizing, mapping, and disseminating research information: A case study of quaking aspen," *Rangelands*, vol. 41, no. 6, pp. 68–76,2019.

[3] V. Velasco, R. Gogu, E. Vázquez-Suñè, A. Garriga, E. Ramos, J. Riera, and M. Alcaraz, "The use of GIS-based 3D geological tools to improve hydrogeological models of sedimentary media in an urban environment," *Environ. Earth Sci.*, vol. 68, no. 8, pp. 2145–2162, Apr. 2013.

[4] R. Liu and Y. Ge, "WebGIS architecture research and design for forestry resources application," in *Proc. 8th Int. Conf. Comput. Sci. Edu.*, Apr. 2013, pp. 1427–1430.

[5] F. Peker, Y. Kurucu, H. H. Tok, E. Saygili, and E. Tok, "An application of GIS-supported analytic hierarchy process to determine the ecological thresholds in the Edirne province," *J. Environ. Protection Ecol.*, vol. 14, no. 2, pp. 713–722, 2013.

[6] M. Yang, T. Liu, X. Wang, Y. Yan, R. Hu, and Q. Zhu, "Design of WebGIS system based on javascript and ArcGIS server," in *Proc. Int. Conf. Smart Grid Electr. Autom. (ICSGEA)*, May 2017, pp. 709–712.

[7] H. Lu, W. Nihong, W. Chang, and C. Yujia, "The research on the WebGIS application based on the J2EE framework and ArcGIS server," in *Proc. Int. Conf. Intell. Comput. Technol. Autom.*, May 2010, pp. 942–945.

[8] W. Yin, D. Zou, C. Bao, Z. Zhou, L. Li, P. Lei, Q. Li, and J. Wu, "An integrated design for geospatial analysis based on WebGIS," in *Proc. IEEE 3rd Adv. Inf. Manage., Communicates, Electron. Autom. Control Conf. (IMCEC)*, Oct. 2019, pp. 1890–1894.

[9] C. Zhou, Z. Du, L. Gao, W. Ming, J. Ouyang, X. Wang, Z. Zhang, and Z. Liu, "Key technologies of a large-scale urban geological information management system based on a browser/server structure," *IEEE Access*, vol. 7, pp. 135582–135594, 2019.

[10] P. Bergougnoux, "Editorial: A perspective on dynamic and multi-dimensional GIS in the 21st century," *GeoInformatica*, vol. 4, no. 4, pp. 343–348, Dec. 2000.

[11] D. P. B. and G. R. Dodagoudar, "An integrated geotechnical database and GIS for 3D subsurface modelling: Application to chennai city, india," *Appl. Geomatics*, vol. 10, no. 1, pp. 47–64, Mar. 2018.

[12] D. L. R. ong, J. G. Shang, and D. Gan, "Unified process for the building of 3D urban geological information system," (in Chinese), *Geolo. Sci. Technol. Inf.*, vol. 35, no. 1, pp. 212–217, 2016.

[13] Y. Yang, Y. Sun, S. Li, S. Zhang, K. Wang, H. Hou, and S. Xu, "A GIS-based Web approach for serving land price information," *ISPRS Int. J. Geo-Inf.*, vol. 4, no. 4, pp. 2078–2093, Oct. 2015.

[14] J. Wang, T. C. Stein, T. Heet, D. M. Scholes, R. E. Arvidson, and V. Heil-Chapdelaine, "A WebGIS for Apollo Analyst's Notebook," in *Proc. 2nd Int. Conf. Adv. Geographic Inf. Syst., Appl., Services*, Feb. 2010, pp. 88–92.

[15] J. Harbschleb, "Ingeo-base an engineering geological database," in *Proc. 6th Int. IAEG Congr.*, 1990, pp. 47–53.

[16] Z. Zhinong, J. Ning, C. Lwo, and W. Qiuyun, "Spatial data management for WebGIS," *J. Syst. Eng. Electron.*, vol. 15, no. 4, pp. 760–765, Dec. 2004.

[17] Istikmal, T. A. Wibowo, and L. V. Yovita, "Polygon WebGIS of distric level for development and monitoring of PUSKESMAS in health care services," in *Proc. 1st Int. Conf. Wireless Telematics (ICWT)*, Nov. 2015, pp. 1–6.

[18] D. Taniar, M. Safar, Q. T. Tran, W. Rahayu, and J. H. Park, "Spatial network RNN queries in GIS," *Comput. J.*, vol. 54, no. 4, pp. 617–627, Apr. 2011.

[19] A. G.-O. Yeh, T. Zhong, and Y. Yue, "Angle difference method for vehicle navigation in multilevel road networks with a three-dimensional transport GIS database," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 1, pp. 140–152, Jan. 2017.

[20] G. R. Gangadharan, "Open source solutions for cloud computing," *Computer*, vol. 50, no. 1, pp. 66–70, Jan. 2017.

[21] G. Kulkarni, P. Khatawkar, R. Shelke, V. Solanke, and R. Waghmare, ""Multi-tenant SaaS cloud," in *Proc. Africon, Pointe-Aux-Piments*, 2013, pp. 1–4.

[22] W. Liu, B. Zhang, Y. Liu, D. Wang, and Y. Zhang, "New model of SaaS: SaaS with tenancy agency," in *Proc. 2nd Int. Conf. Adv. Comput. Control*, 2010, pp. 463–466.

[23] C.-K. Tham and B. Cao, "Stochastic programming methods for workload assignment in an ad hoc mobile cloud," *IEEE Trans. Mobile Comput.*, vol. 17, no. 7, pp. 1709–1722, Jul. 2018.

[24] W. Shen, B. Yin, X. Cao, Y. Cheng, and X. Shen, "A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 415–430, Apr. 2019.

[25] V. Balasubramanian, F. Zaman, M. Aloqaily, S. Alrabaee, M. Gorlatova, and M. Reisslein, "Reinforcing the edge: Autonomous energy management for mobile device clouds," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 44–49.

[26] Q. Qi and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing," *IEEE Access*, vol. 7, pp. 86769–86777, 2019.

[27] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, IoT, edge, and fog," *IEEE Access*, vol. 7, pp. 150936–150948, 2019.

[28] (Jul. 2020). *WebRTC 1.0: Real-Time Communication Between Browsers*. [Online]. Available: https://www.w3.org/TR/webrtc/

[29] (Jul. 2020). *Real-Time Communication in WEB-Browsers (RTCWeb)*. [Online]. Available: https://datatracker.ietf.org/wg/rtcweb/documents/

[30] J. Jang-Jaccard, S. Nepal, B. Celler, and B. Yan, "WebRTC-based video conferencing service for telehealth," *Computing*, vol. 98, nos. 1–2, pp. 169–193, Jan. 2016.

[31] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, "Performance analysis of the janus WebRTC gateway," in *Proc. 1st Workshop All-Web Real-Time Syst. AWeS*, 2015, pp. 1–7.

[32] M. Moulay and V. Mancuso, "Experimental performance evaluation of WebRTC video services over mobile networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 541–546.

[33] G. Bakar, R. A. Kirmizioglu, and A. M. Tekalp, "Motion-based rate adaptation in WebRTC videoconferencing using scalable video coding," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 429–441, Feb. 2019.

[34] A. Bergkvist, D. C. Burnett, C. Jennings, A. Narayanan, and B. Aboba, "WebRTC 1.0: Real-time communication between browsers," W3C Work. Draft, W3C, American, Tech. Rep., 2016. [Online]. Available: https://www.w3.org/TR/2016/WD-webrtc-20160913/

[35] E. Bertin, S. Cubaud, S. Tuffin, N. Crespi, and V. Beltran, "WebRTC, the day after: What's next for conversational services?" in *Proc. 17th Int. Conf. Intell. Next Gener. Netw. (ICIN)*, Oct. 2013, pp. 46–52.

[36] K. H. Abdulkareem, M. A. Mohammed, S. S. Gunasekaran, M. N. Al-Mhiqani, A. A. Mutlag, S. A. Mostafa, N. S. Ali, and D. A. Ibrahim, "A review of fog computing and machine learning: Concepts, applications, challenges, and open issues," *IEEE Access*, vol. 7, pp. 153123–153140, 2019.

[37] H.-J. Hong, "From cloud computing to fog computing: Unleash the power of edge and end devices," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2017, pp. 331–334.

[38] O. Zhdanenko, J. Liu, R. Torre, S. Mudriievskiy, H. Salah, G. T. Nguyen, and H. P. Frank Fitzek, "Demonstration of mobile edge cloud for 5G connected cars," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–2.

[39] J. Cui, L. Wei, J. Zhang, Y. Xu, and H. Zhong, "An efficient message-authentication scheme based on edge computing for vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1621–1632, May 2019.

[40] Z. Haitao, D. Yi, Z. Mengkang, W. Qin, S. Xinyue, and Z. Hongbo, "Multipath transmission workload balancing optimization scheme based on mobile edge computing in vehicular heterogeneous network," *IEEE Access*, vol. 7, pp. 116047–116055, 2019.

[41] E. Baccarelli, M. Scarpiniti, and A. Momenzadeh, "EcoMobiFog–design and dynamic optimization of a 5G Mobile-Fog-Cloud multi-tier ecosystem for the real-time distributed execution of stream applications," *IEEE Access*, vol. 7, pp. 55565–55608, 2019.

[42] E. E. Ugwuanyi, S. Ghosh, M. Iqbal, and T. Dagiuklas, "Reliable resource provisioning using Bankers' deadlock avoidance algorithm in MEC for industrial IoT," *IEEE Access*, vol. 6, pp. 43327–43335, 2018.

[43] P. Shantharama, A. S. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein, and A. Scaglione, "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, vol. 6, pp. 57545–57561, 2018.

[44] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. P. Fitzek, "Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166079–166108, 2019.

[45] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, and M. Imran, "Heterogeneity-aware task allocation in mobile ad hoc cloud," *IEEE Access*, vol. 5, pp. 1779–1795, 2017.

[46] T. Dbouk, A. Mourad, H. Otrok, H. Tout, and C. Talhi, "A novel ad-hoc mobile edge cloud offering security services through intelligent resource-aware offloading," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1665–1680, Dec. 2019.

[47] Y. Gong, C. Zhang, Y. Fang, and J. Sun, "Protecting location privacy for task allocation in ad hoc mobile cloud computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 110–121, Jan. 2018.

[48] F. Chi, X. Wang, W. Cai, and V. C. M. Leung, "Ad-hoc cloudlet based cooperative cloud gaming," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 625–639, Jul. 2018.

[49] L. Ritchie, H. S. Yang, A. W. Richa, and M. Reisslein, "Cluster overlay broadcast (COB): MANET routing with complexity polynomial in source-destination distance," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 653–667, Jun. 2006.

[50] C.-A. Sun, T. Xue, and M. Aiello, "ValySeC: A variability analysis tool for service compositions using VxBPEL," in *Proc. IEEE Asia–Pacific Services Comput. Conf.*, Dec. 2010, pp. 307–314.

[51] A. Charfi and M. Mezini, "AO4BPEL: An aspect-oriented extension to BPEL," *World Wide Web*, vol. 10, no. 3, pp. 309–344, Aug. 2007.

[52] M. Aiello, P. Bulanov, and H. Groefsema, "Requirements and tools for variability management," in *Proc. IEEE 34th Annu. Comput. Softw. Appl. Conf. Workshops*, Jul. 2010, pp. 245–250.

[53] V. Beltran, E. Bertin, and N. Crespi, "User identity for WebRTC services: A matter of trust," *IEEE Internet Comput.*, vol. 18, no. 6, pp. 18–25, Nov. 2014.

[54] K. Fai Ng, M. Yan Ching, Y. Liu, T. Cai, L. Li, and W. Chou, "A P2P-MCU approach to multi-party video conference with WebRTC," *Int. J. Future Comput. Commun.*, vol. 3, no. 5, pp. 319–324, Oct. 2014.

[55] J. Raigoza and R. Thakkar, "Browser performance of JavaScript framework, SAPUI5 & jQuery," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vegas, NV, USA, Dec. 2016, pp. 1420–1421.

[56] S. Delcev and D. Draskovic, "Modern JavaScript frameworks: A survey study," in *Proc. Zooming Innov. Consum. Technol. Conf. (ZINC)*, May 2018, pp. 106–109.

[57] H. Park, S. Kim, and S. Moon, "Work-in-progress: Advanced ahead-of-time compilation for Javascript engine," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst. (CASES)*, Seoul South Korea, Oct. 2017, pp. 1–2.

[58] S. López, J. Silva, and D. Insa, "Using the DOM tree for content extraction," *Electron. Proc. Theor. Comput. Sci.*, vol. 98, pp. 46–59, Oct. 2012.

HONGLEI HE received the Master of Engineering degree in computer technology from the Nanjing University of Technology, in 2009. He is currently an Associate Professor with the School of Information Engineering, Lianyungang Technical College, China. His research interests include WebGIS, edge computing, machine learning, and multimedia communication.

WENMING ZHU received the Ph.D. degree from the University of Science and Technology of China, in 2006. He is currently a Senior Engineer with the Shenzhen Institute of Information Technology. His main research interests include multimedia communication, network intelligence, and machine learning.