Study of SaaS Mode WebGIS based on JavaScript Engine

Honglei He

School of Information Engineering, Lianyungang Technical College, Lianyungang, 222006, China

he_hong_lei@163.com

Abstract. WebGIS has a wide range of application requirements, but development and application require high labor and resource costs. In order to solve this contradiction, a SaaS mode WebGIS system is proposed. Through the statistical summary of various application scenarios, the system provides a variety of geographic information atomic services to choose from. Tenants can freely combine these services to implement a WebGIS system that meets their needs. The user interface is implemented by an online HTML editing system, and the data structure is self-made through a metadata service, and the business logic is self-made through service assembly and business rules. Business rules are ultimately done through event processing by client-side JavaScript. Through the establishment of an instance, the feasibility of SaaS mode WebGIS system is verified, and users can build WebGIS system according to their own needs

Keywords: JavaScript;SAAS; WebGIS; Customizable; load balancing.

1. Introduction

WebGIS refers to a geographic information system based on WWW (World Wide Web) technology running on the Internet. Through the World Wide Web technology, the use of WebGIS is very convenient and popular. Any user on the Internet can use geographic information services through browsers, such as browsing, querying, analyzing, and modifying spatial data. Compared with traditional GIS, WebGIS greatly improves the sharing range and operational efficiency of spatial information. As a result, this application has been rapidly popularized and rapidly developed in various industries, including transportation, weather, logistics, electricity, agriculture, tourism, community security, and campus management.

The most mainstream way today is a layered tile map. The client uses the scripting language Javascript supported by the browser by default, without any plugins installed, and is universal and easy to use. The server provides a Web Map Service (WMS) that returns the corresponding map tiles based on the user's request [1]. (see picture 1)

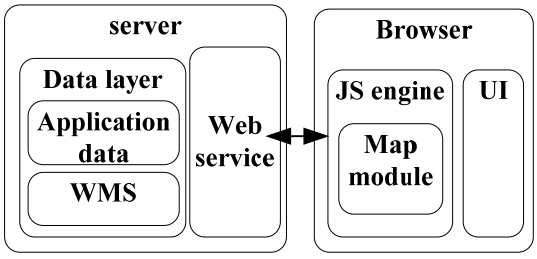

Fig.1 WebGIS System structure

Currently, many organizations hope to have their own WebGIS system, which requires the development of a WebGIS system. Software currently used in development systems: ARCGIS, mapinfo, Topmap, supermap, etc [2]. These development software's are expensive and require a professional developer. After the development is completed, it is also necessary to set up a special operating environment for the system. For most units, it is a big burden.

In order to solve the contradiction between this broad demand and high cost, this paper proposes a SAAS model WebGIS based on JavaScript engine. The system adopts the B/S mode, the user does not need to install the software, and only needs to use the browser to complete the customization and use of the WebGIS system. The huge cost of purchasing professional software is eliminated. In

addition, the SAAS model of the JavaScript engine proposed by the article greatly improves the stability and throughput of the system compared with the traditional method.

2. Design of SAAS Mode WebGIS

2.1 Model Structure

SAAS mode WebGIS systems need to be customizable. Because the system has to support a large number of users at the same time, and the needs of a large number of users will not be exactly the same, the system must provide customizable features. Tenants can customize business logic, user interface, data structure, etc. according to their needs. According to this requirement, the SAAS model of WebGIS system is designed (Figure 2).

The system uses SOA (Service Oriented Architecture), and the bottom layer is the service entity layer, which provides various atomic services. The service assembly layer implements system functions by combining services. Since services are independent of each other, different functions can be implemented in different combinations [3]. Tenants can choose their existing services to assemble or customize their own service assembly to implement their own business logic functions. At the same time, tenants can also customize the user interface and data structure. End users can choose to use a variety of tenant-customized instances.

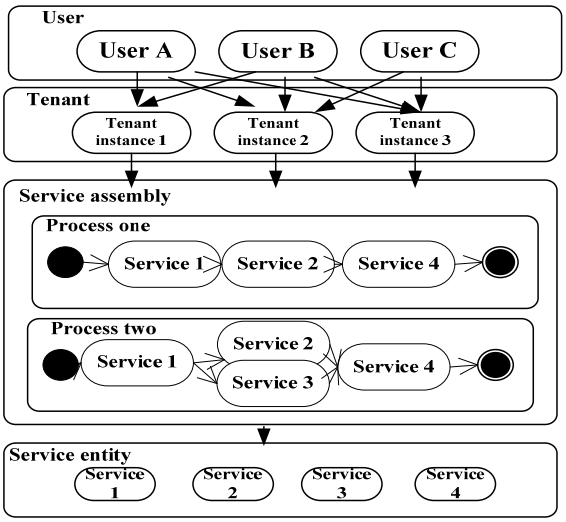

Fig. 2 Model structure

Therefore, the main points of SAAS mode WebGIS design are: user interface, user data structure, service assembly (business logic) customization.

(1) User interface customization

Customization of the user interface can be achieved through a metadata service. Tenants customize their own interfaces, such as icons, forms, text, backgrounds, table layouts, etc., and store customized interface data in metadata tables. When the user accesses the system, the system displays the interface according to the data of the metadata table.

(2) Customization of user data structure

If each user has a unique database or each user has a unique set of tables in a database, the customization of the user data structure is very simple, and the user-defined table structure is fine.

Considering the effective use of system resources, a multi-user shared a table set of a database, using a metadata service. The method is to keep the expandable field in the shared table set. This field does not directly determine the meaning and data type of the field, but requires a metadata table to determine. Because metadata can be used as data to describe data, users only need to customize the metadata to customize the data structure.

(3) Service assembly customization

The service assembly can be customized online by the tenant. The tenant first customizes the process control file and then passes it to the process control engine. The process engine parses the file generation process instance and controls the assembly of each atomic service [4-5]. The process engine is implemented by a Javascript program.

2.2 JavaScript-based System Control Engine

The system control engine is responsible for running the system in accordance with user-customized parameters, including the user interface and business logic. The user interface data (HTML file) is stored in the user's metadata table, so it can be loaded when the system is initialized. Business logic control is the core of the control module. Considering the reduction of server load and the interaction with the user, the control engine design runs on the browser side and is done by a JavaScript program. The control engine first loads the corresponding JavaScript library according to the service module added by the user. Then bind the corresponding DOM object and initialize it. Then, according to the business rule file, the relevant service module is called in order, and the user's activity is monitored, and the user activity is responded according to the conditions set by the business rule file [6-7], as shown in FIG. 3.

Listening for user activity refers to listening to the specified DOM object in the business rule. To prevent confusion in listening to browser DOM objects, use the listener method for DOM level2 events:

object.addEventListener("event", eventFunction, boolean);

Object is the DOM object specified in the business rule, obtained with getElementById (DomID). The DomID is the ID of the specified DOM object. To prevent event bubbling, boolean is set to false.

Business logic processing can be seen as a central process, running in a single-threaded asynchronous mode. It is determined according to the established conditions to call different service modules, which may be user activities or the return value of the last called service. Business logic handles loops based on conditions, declaring variables, copying and assigning values, and defining fault handling.

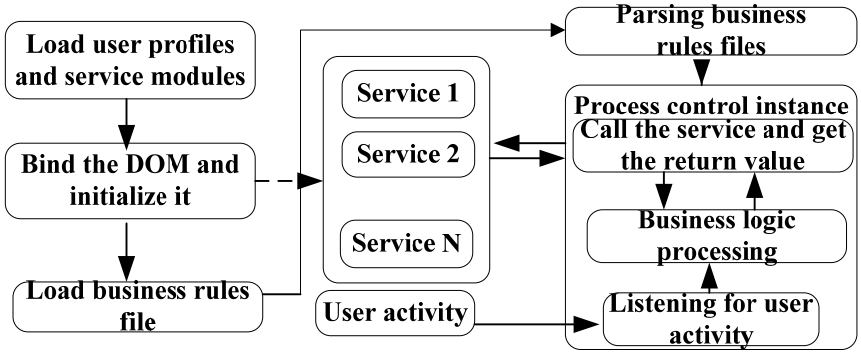

Fig.3 Control engine structure

Considering that JavaScript must be run on one page, it cannot be run continuously after replacing the page. So, you must keep a main page running the control engine, and other variable activities, run inside the iframe. Also, consider that the browser closes unexpectedly. The execution process of the control engine is recorded in the cookie local data. If you accidentally close the browser, etc., you can continue from the breakpoint.

3. Performance Evaluation

To further evaluate the SAAS model WebGIS based on the JavaScript engine. Compare the traditional SAAS model with the model presented in this paper.

In the traditional mode, all service instances are running on the server. In the mode proposed in this paper, only the generic service S_0 (initialization and data access) instances run on the server side, and other differentiated service instances run on the client.

The comparison of the response times of the various examples in the two modes is discussed below.

Definition 1: The server has P, the jth server is represented as V_j . and its computer capability is expressed as F_{vj} $(j \in P)$.

Definition 2: The client has Q, and the i-th client is represented as c_i. Its computer capability is expressed as E_{ci} $(i \in Q)$.

Definition 3: Service K, where service y is represented as S_y $(y \in K)$, the workload of service S_k: w_y^s.

Definition 4: There are T instances, where instance b is represented as α_b and the service set of α_b is $S_0 + \sum_{y=m_b}^{n_b} S_y$ （$1 \leqslant m_b \leqslant n_b \leqslant k$）.

Definition 5: Client i request rate: R_i (req / s), then the server-side request arrival rate:

To simplify the calculation, assume that the services in all instances are executed sequentially. The workload of the service set of instances α_b is

$$W_0^S + \sum_{y=m_b}^{n_b} W_y^S \quad (1 \leqslant mb \leqslant nb \leqslant k)$$

In the traditional SAAS mode, all services are deployed on the server side. the service request workload arriving in unit time is shown in Equation 1:

$$G = \sum_{i=1}^{Q} R_i \left(W_0^S + \sum_{y=m_i}^{n_i} W_y^S \right) \quad (1 \leqslant mi \leqslant ni \leqslant k) \tag{1}$$

Average server processing speed = PF/G.

Assume that in the case of no overload operation, according to the Erlang C conclusion of queuing theory:

$$\text{Average waiting time} = \frac{E_c\left(P, \frac{G}{F}\right) * \frac{G}{F}}{H(1-\rho)} \tag{2}$$

Where E_c is the Erlang C formula and the expression is as in Equation 3:

$$E_c(H, V) = \frac{\frac{V^H}{H!}}{\frac{V^H}{H!} + (1-\rho)\sum_{K=0}^{H-1} \frac{H^K}{K!}} \quad \left(\rho = \frac{V}{H}\right) \tag{3}$$

Therefore, for the customer instance i overall service response time T_i^{Server} as Equation 4:

$$T_i^{Server} = \frac{E_c\left(P, \frac{G}{F}\right) * \frac{G}{F}}{H(1-\rho)} + \frac{\sum_{y=m_i}^{n_i} S_y}{\frac{PF}{G}} \quad (1 \leqslant m_i \leqslant n_i \leqslant k) \tag{4}$$

In the SAAS mode proposed in this paper, the initialization service S_0 is deployed on the server side, and other services are deployed on the client side.

The service request workload for reaching the server per unit time is shown in Equation 5:

$$G_2 = \sum_{i=1}^{Q} K_i \left(W_0^S\right) \quad (1 \leqslant m_i \leqslant n_i \leqslant k) \tag{5}$$

According to Equation 4, server-side processing time:

$$\frac{E_c\left(P, \frac{G_2}{F}\right) * \frac{G_2}{k*F}}{H(1-\rho)} + \frac{G_2}{PF}$$

Client i processing time T_i^c:

$$T_i^c = \frac{\sum_{y=m_i}^{n_i} W_y^S}{E_i} \quad (1 \leqslant m_i \leqslant n_i \leqslant k;\ i \in Q) \tag{6}$$

so, the overall response time is shown in Equation 7:

$$T_i^{Client} = \frac{E_c\left(P, \frac{G_2}{F}\right) * \frac{G_2}{k*F}}{H(1-\rho)} + \frac{G_2}{PF} + \frac{\sum_{y=m_i}^{n_i} W_y^S}{E_i} \quad (1 \leqslant m_i \leqslant n_i \leqslant k;\ i \in Q) \tag{7}$$

Using matlab 2014b for simulation evaluation, in order to simplify the calculation, it is assumed that the number of servers is P=2 (number), the computing power of each server is F_{vj}=100 (workload/second), and the computing power of each client is E_{ci}=20 (workload/ Seconds, each instance contains the number of services K = 5 (one), the workload of each service S_k is 1, all client i request rate R = 1 (times / second). Table 1 shows the changes in the service response times T_i^{Client} and T_i^{Server} of the two configurations as the number of clients Q changes.

Table 1. Response time comparison

Q	TClient	TServer
1	0.2100	0.0500
2	0.2100	0.0501
……	……	……
62	0.2103	0.2632
63	0.2104	0.3463
64	0.2104	0.5128
65	0.2104	1.0127

When the number of clients is small, the response time of the traditional SAAS mode is faster. As the number of users increases, the response time of the traditional SAAS mode increases rapidly, which is much higher than the SAAS mode response time proposed in this paper. The response time of the SAAS model proposed in this paper is very stable and basically does not change with the number of clients.

4. Instance Performance Study

In order to verify whether the developed SAAS mode WebGIS platform can support multiple tenants to execute different business process instances at runtime, and whether different process instances can be isolated, we simulate two tenants on the client, and the tenants set the configuration plan separately, and then The configuration scheme is saved as a different user profile. The experimental environment configuration is shown in Table 2. User A needs to customize a WebGIS system to draw a line chart. User B needs to customize a WebGIS system to map weather maps. The results are shown in Figures 4 and 5.

Table 2. Experimental environment configuration

Configuration	server	Tenant
Cpu	Xeon	Core
Main frequency	3.4G	3.4G
RAM	8G	4G
operating system	Windows Server 2012	Windows7

User B's WebGIS system, its JavaScript-based process control engine. Code is shown in Figure 6.

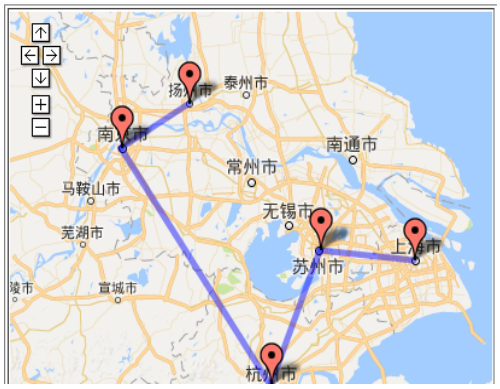

Fig.4 User A customized WebGIS

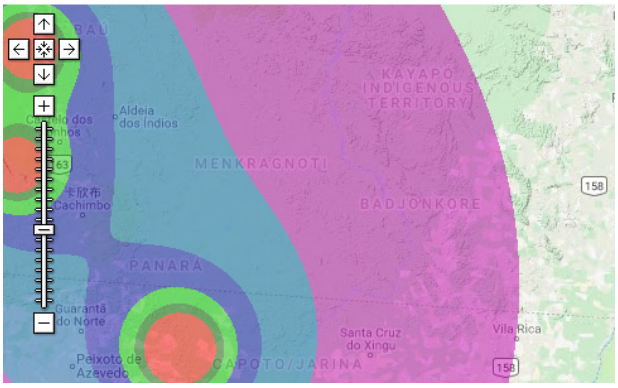

Fig.5 User B customized WebGIS

```
document.getElementById('btnJS').addEventListener('click', myFunction);
    function myFunction() {
    eval(document.getElementById('jsValue').value);
    }
document.getElementById('btnHtml').addEventListener('click', myFunction2)
    function myFunction2() {
    $('#mapDiv').html(document.getElementById('htmlValue').value);
    console.log(document.getElementById('mapDiv').innerhtml)
    }
document.getElementById('btn_map').addEventListener('click', myFunction3)
```

```
var map_1 = new GMap2(document.getElementById("map_1"));
map_1.setCenter(new GLatLng(26, -110), 3);
map_1.addControl(new GLargeMapControl());
addCanves(map_1,'(26,-110)');
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
var x1=20,y1=130,x2=30,y2=35,x3=150,y3=275;
var m1=60,m2=70,m3=98;

for(x=0;x<=500;x++){
```

Fig.6 The Code

The experimental results show that the developed SAAS mode WebGIS platform can derive different process instances according to the tenant configuration file, and the execution of each process instance is isolated and does not affect each other.

5. Summary

The article proposes a scheme for the SAAS-mode WebGIS system. With a service-oriented architecture, users' common functions are decomposed and modularized. Users can freely configure the system to distribute the load to the server and client for execution. The user system instance is controlled by the JavaScript engine. Through the establishment of two examples, it is verified that the solution of the key technology of the system is effective.

References

[1]. Y E Wang, H Liu.Research on WebGIS and Its Architecture. Surveying Engineering.vol. 18 (2009), p.70-73.

[2]. J Guo, P Yan.Analysis and Design of Urban Tourism Information System Based on WebGIS. Energy Procedia. vol. 13(2011). No.1, p. 3794-3799.

[3]. R Krishna,J Iqbal, A K Gorai.Groundwater vulnerability to pollution mapping of Ranchi district using GIS.Applied Water Science. vol. 5(2015) No.4, p. 345-358.

[4]. M Koning, C A Sun, M.Sinnema. Avgeriou P. VxBPEL: Supporting variability for Web services in BPEL. Information & Software Technology. vol.51(2009) No. 2, pp. 258-269.

[5]. M S El-Kady, M A ElMesmary. Creating spatial database of the foundation soil in Aljouf area using GIS. Innovative Infrastructure Solutions. vol. 3(2018)No.1, p. 1-6.

[6]. T T Wang, Q J Shen. JavaScript debugger software architecture. Journal of Natural Science of Hunan Normal University. vol.37(2014) No.6, p. 88-92.

[7]. M C Loring,M Marron, D Leijen. Semantics of asynchronous JavaScript. ACM SIGPLAN Notices.vol.52(2017) No. 11, p. 51-62.