A load balancing method to support spatial analysis in XML/GML/SVG-based WebGIS

Haosheng Huang, Yan Li & Georg Gartner

ABSTRACT: This chapter aims at introducing method for load balancing spatial analysis into XML/GML/SVG-based WebGIS. We propose that the decision on where to execute spatial query operations (server side or browser side) should be based on the network communication cost versus the computational cost. This chapter mainly focuses on the network communication cost. After analyzing the workflow of spatial analysis, we address the following issues: GML (server side)/SVG (browser side) -based spatial information representation, spatial query language, and load balancing middlewares for dispensing spatial queries to either server side or browser side. Finally, we design some case studies to evaluate the proposed method. The results prove that the method is feasible and operable to support spatial analysis in the Web environment, and has a better performance compared to the server side solution and the browser side solution. The method enables users to access spatial analysis functions simply with a web browser with SVG support.

Keywords: XML/GML/SVG, WebGIS, spatial analysis, load balancing, network communication cost, spatial data modeling, SVG-based Spatial Extended SQL

1 INTRODUCTION

Recent years have witnessed rapid advances in Web-based geographic information systems (WebGIS), which aim at providing GIS functionality and services (such as web mapping and spatial analysis) to users through a common web browser, such as Internet Explorer and Firefox. The ability to support spatial analysis is viewed as one of the key characteristics which distinguish GIS from other information systems. In order to meet the increasing need of spatial information applications in the Web environment, spatial analysis should be introduced into WebGIS.

Scalable Vector Graphics (SVG) was proposed by the World Wide Web consortium (W3C) as a XML-based standard for describing two-dimensional vector graphics and graphical applications in 2001 (W3C 2003). Since then, SVG, together with XML and GML (Geography Markup Language, proposed by the Open Geospatial Consortium) (OGC 2003), have been increasingly considered as a promising solution for solving the problems of spatial data integration and sharing from the syntactic (data format) level. In this solution, GML is used as a coding, storing and transmitting standard of multiple spatial data on the server side, while SVG is considered as a visualization tool for displaying and interacting with vector spatial data on the browser side.

In the literature, some researchers used SVG's graphic elements (such as line and path) and graphic styles (such as fill, stroke and opacity) for spatial information visualization and developed some prototype systems (Neumann & Winter 2010, Peng et al. 2004, Köbben 2007). Furthermore, there is also some research focusing on transformation of GML to SVG (Guo et al. 2003, Tennakoon 2003, Herdy et al. 2008), SVG for LBS (Location Based Services) applications (Jeong et al. 2006), etc.

However, most of the WebGIS applications, especially XML/GML/SVG-based WebGIS applications, have been designed for visualization (web mapping) only, and "avoid the access to spatial analysis functions" such as spatial topological queries, map overlay, and buffer analysis that are vital to spatial information applications (Lin & Huang 2001). When performing spatial analysis tasks, we often install corresponding GIS software (e.g., ESRI's ArcGIS) on our computer, and carry out the tasks in a stand-alone or Local Area Network (LAN) environment. In the meantime, these applications only support their own spatial data formats. There are also some WebGIS applications which carry out all the spatial analysis tasks on the server side and then send the results to the browser side for visualization. These server-side solutions, sometimes, become impractical, as the server cannot handle a large volume of concurrent users. In order to solve this "bottleneck" problem, some of them introduce load balancing technology to the server side, which dispenses users' spatial queries and analysis requests to different servers in a server cluster (Supermap 2010, Qin & Li 2007, Luo et al. 2003, Wang et al. 2004). However, spatial analysis is a complex process; users often have to try different queries before they are satisfied with the results. Since every query may result in a large volume of data (such as intermediate results which users may not need), there will be a high transmission overload between the server side and the web browser side when using this server-side solution. As such, in order to improve the performance, not all the spatial operations should be implemented on the server side. For example, as the "buffer" operation often results in more data output than input, it may be implemented on the browser side in order to reduce the network transmission load.

Recognizing these limitations, this chapter proposes that *in order to meet the increasing need of spatial information application in the web environment, spatial analysis based on load balancing technology should be introduced into XML/GML/SVG-based WebGIS*. Load balancing spatial analysis carries out spatial queries and analyses *either on the server side or the browser side* depending on the *network communication cost* versus the *computational cost*. However, little work has been done on this aspect.

The objective of this chapter is to introduce method for load balancing spatial analysis into XML/GML/SVG-based WebGIS. After analyzing the workflow of spatial analysis (defining the goal and evaluation criteria, representing the needed geospatial data, carrying out spatial query and analysis with GIS tools, and result appraisal and explanation), we focus on the following three key issues: *GML (server side)/SVG (browser side) –based spatial information representation*, *spatial operators and spatial query language* for both server side (for GML) and browser side (for SVG), and *load balancing middlewares* for dispensing spatial operations to the server side or the browser side. With the method, users can easily carry out spatial analysis tasks in the Web environment.

Having provided a background to load balancing spatial analysis, the rest of this chapter is structured as follows. In Section 2, we analyze the workflow of spatial analysis, and identify the key issues of introducing spatial analysis into XML/GML/SVG-based WebGIS. Section 3 uses the theory of spatial data modeling to design GML/SVG-based spatial information representation models. Based on the representation models, we design some spatial operators and a spatial extended SQL (SSESQL) to support spatial query and analysis directly on GML/SVG in Section 4. Section 5 describes the load balancing middlewares. Some case studies are implemented to evaluate the method in Section 6. Finally, Section 7 draws the conclusions and presents the future work.

2 WORKFLOW OF SPATIAL ANALYSIS

Spatial analysis is "a set of methods whose results change when the locations of the objects being analyzed change" (Longley et al. 2005, p.217). It plays a key role in GIS. According to Wu (2002), the workflow of spatial analysis includes four steps: 1) defining the goal and evaluation criteria; 2) representing the needed spatial dataset; 3) carrying out spatial query and analysis with GIS tools; and 4) result appraisal and explanation. Steps 1 and 4 require domain knowledge and are mainly carried out by domain experts. For step 2 and step 3, GIS tools are needed to support and assist human-computer interaction. Figure 1 depicts the workflow of spatial analysis.

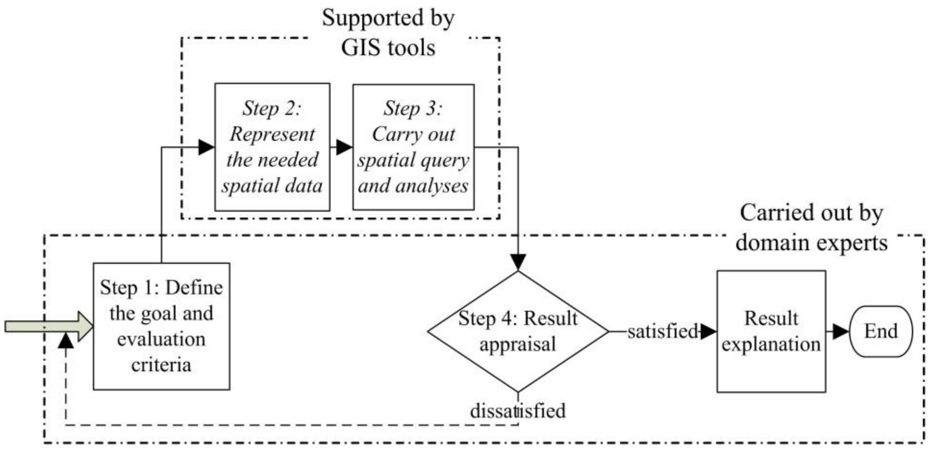

Supported by GIS tools
Step 2: Represent the needed spatial data
Step 3: Carry out spatial query and analyses
Carried out by domain experts
Step 1: Define the goal and evaluation criteria
Step 4: Result appraisal
satisfied
Result explanation
End
dissatisfied

Figure 1. Workflow of spatial analysis.

The chapter mainly focuses on designing method to support and assist step 2 and step 3. For step 2, we design GML/SVG-based spatial information representation models which can be used to represent the needed spatial datasets in GML and SVG (Section 3). For step 3, we design and implement some spatial operators and integrate them into an SVG-based Spatial Extended SQL (SSESQL) to support spatial query and analysis on spatial datasets represented in GML and SVG (Section 4). In order to improve the performance, we also introduce load balancing middlewares to reduce the network transmission load (Section 5). However, the load balancing middlewares are completely transparent to the end users.

3 SPATIAL DATA MODELING FOR SVG/GML-BASED SPATIAL INFORMATION REPRESENTATION

In this section, we employ the theory of spatial data modeling to design SVG/GML-based spatial information representation models for representing spatial data on both server side and browser side.

3.1 *SVG-based spatial information representation model*

In Huang et al. (2008), we set up a theoretical framework for SVG-based spatial information representation based on the theory of spatial data modeling, which includes three steps: 1) choosing a conceptual model which can abstract the real world most appropriately, 2) choosing a suitable data structure to represent the conceptual model, and 3) designing a file format, or an appropriate method to record or store the data structure in Step 2.

Based on that theoretical framework, Huang et al. (2008) adapted the OGC's Geometry Object Model (GOM) (OGC 1999), and developed an SVG-based spatial information representation model (Fig. 2).

In this model, we use *<svg>* element to represent *Map* (the dataset), and use *viewBox* attribute to represent its bounded range. *Layer* is represented as *<g>* element. *Point*, *Curve*, *Surface* are represented as *<circle>*, *<path>*, and *<path>*, respectively. *<g>* element is used to represent the *Multipoint*, *Multicurve*, *Multisurface* and *Multigeometry*. Both spatial and non-spatial attributes of spatial objects are represented as corresponding SVG element's attributes. In the model, if B is PART-OF A (i.e., composition/aggregation relationship), B is represented as a child element of A. For example, Layer is PART-OF Map, so <g> element which represents Layer is a child element of <svg> element which represents Map. With this model, users can use SVG to represent the needed spatial information. Figure 6 illustrates an example of such representation.

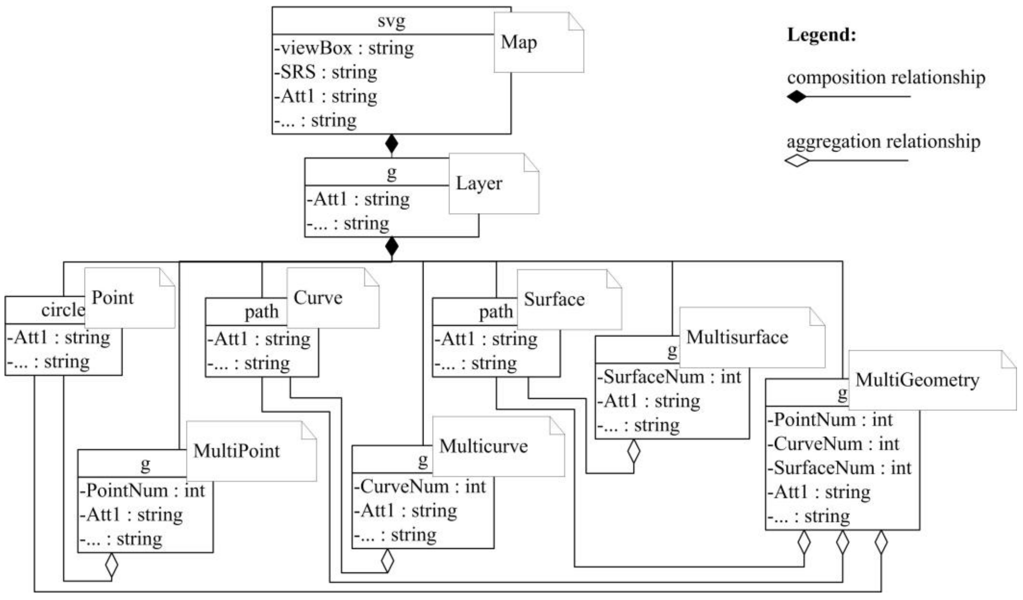


Figure 2. SVG-based spatial information representation model.

3.2 *GML-based spatial information representation model*

Similar to our SVG-based representation model, we developed a GML-based spatial information representation model (Fig. 3) (Huang & Li 2009).

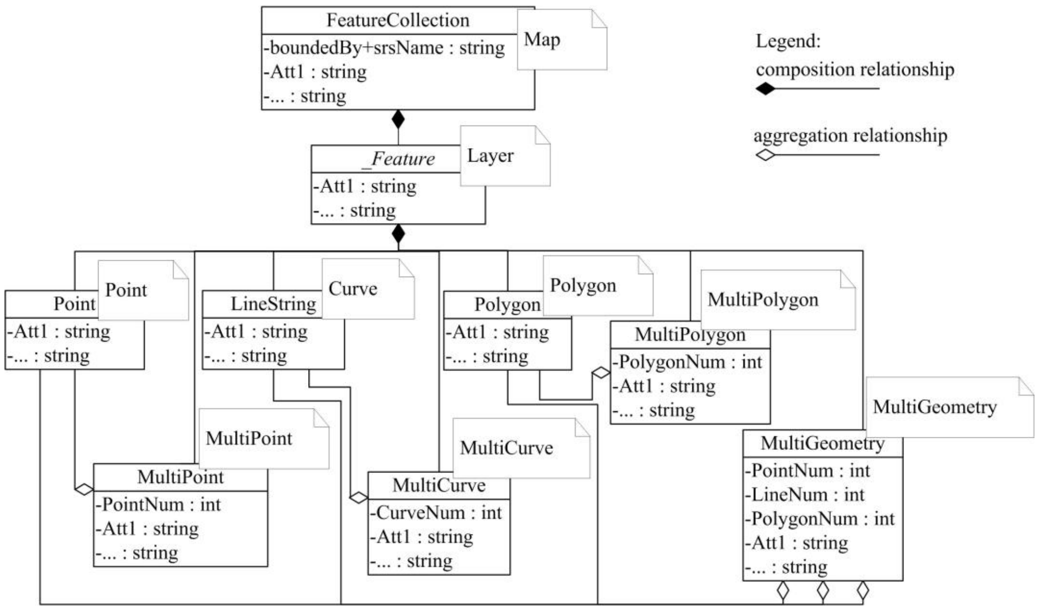


Figure 3. GML-based spatial information representation model.

In this model, we use GML "*FeatureCollection*" element to represent the *Map* (dataset), and use "*boundedBy*" and "*srsName*" to represent the bounded area and spatial coordinate system of the *Map*. *Layer* is represented as element which is inherited from "*gml:_Feature*". If B is PART-OF A (composition relationship in Fig. 3), B is represented as a child element of B. Both spatial attributes and non-spatial attributes are represented as the child elements of the corresponding GML elements.

3.3 *Transformation from GML to SVG*

By using the models depicted in Figures 2-3, users can easily use GML/SVG to represent the needed spatial data on both the server side and the browser side. Because they are based on the same conceptual model and spatial data structure (logical model), GML-based spatial datasets on the server side can be easily and losslessly converted to SVG-based datasets for the browser side. Table 1 compares the different elements/attributes in the GML and SVG-based spatial information representation models.

Table 1. Comparison of GML and SVG-based spatial information representation models.

Description	GML elements	SVG elements/ attributes
Map	FeatureCollection	svg
Bounded area of the map	boundedBy/ Envelope	viewBox
Point object	Point	circle
Curve object	LineString	path
Polygon object	Polygon	path (end with “Z”)
Compound object	MultiPoint, MultiLine, MultiPolygon, MultiGeometry	g

When transforming GML datasets to SVG datasets for visualization, we should also make some transformation to the coordinate systems in GML. Figure 4 compares the differences between the coordinate systems used in GML and SVG. Therefore, if we do not make any transformation to the coordinate systems, the map represented in SVG may become headstand.

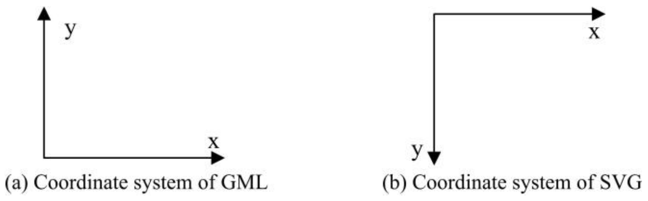


Figure 4. Comparison of the coordinate systems of GML and SVG.

We use SVG’s “*translate*” attribute to implement this transformation. The transformation parameter is set as “*translate (0, 2*min_y + height), scale(1,-1)*”, where *(min_x, min_y, width, height)* represents the bounded area of the map, and can be found in the “*viewBox*” attribute of the root “SVG” element. Because all the layers have to be translated to avoid headstand, we use <g> element to group all the layers together.

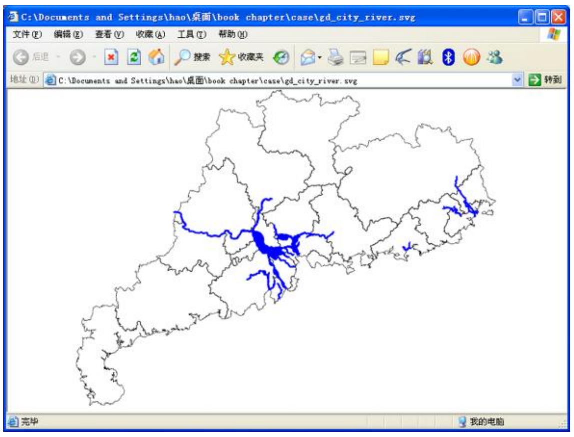
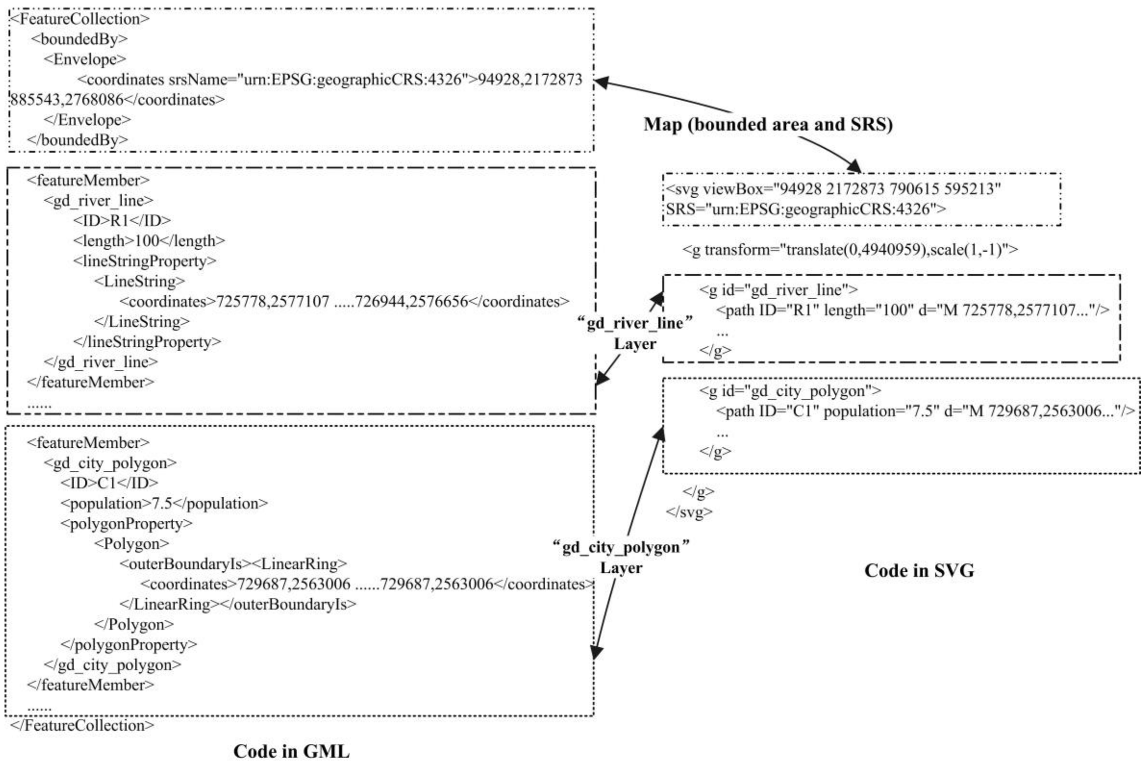

C:\Documents and Settings\hao\桌面\book chapter\case\gd_city_river.svg
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) C:\Documents and Settings\hao\桌面\book chapter\case\gd_city_river.svg 转到
完毕
我的电脑

Figure 5. An SVG map of Guangdong province with two layers (gd_river_line and gd_city_polygon)

Figure 5 shows a map of Guangdong province (China) represented in SVG. We use Internet Explorer with SVG plug-in to visualize the map. The map includes two layers: *gd_river_line* and *gd_city_polygon*. In Figure 6, we compare the GML and SVG codes which represent the map using our suggested models (Figs 2-3). The code "*<g transform="translate(0, 4940959), scale(1,-1)">*" in Figure 6 also gives an example about the transformation of the coordinate systems.



```
<FeatureCollection>
  <boundedBy>
    <Envelope>
        <coordinates srsName="urn:EPSG:geographicCRS:4326">94928,2172873
885543,2768086</coordinates>
    </Envelope>
  </boundedBy>

  <featureMember>
    <gd_river_line>
       <ID>R1</ID>
       <length>100</length>
       <lineStringProperty>
          <LineString>
             <coordinates>725778,2577107 .....726944,2576656</coordinates>
          </LineString>
       </lineStringProperty>
    </gd_river_line>
  </featureMember>
  ......

  <featureMember>
    <gd_city_polygon>
      <ID>C1</ID>
      <population>7.5</population>
      <polygonProperty>
          <Polygon>
             <outerBoundaryIs><LinearRing>
                <coordinates>729687,2563006 ......729687,2563006</coordinates>
             </LinearRing></outerBoundaryIs>
          </Polygon>
      </polygonProperty>
    </gd_city_polygon>
  </featureMember>
  ......
</FeatureCollection>
```

```
<svg viewBox="94928 2172873 790615 595213"
SRS="urn:EPSG:geographicCRS:4326">

  <g transform="translate(0,4940959),scale(1,-1)">

     <g id="gd_river_line">
        <path ID="R1" length="100" d="M 725778,2577107..."/>
        ...
     </g>

     <g id="gd_city_polygon">
        <path ID="C1" population="7.5" d="M 729687,2563006..."/>
        ...
     </g>

  </g>
</svg>
```

Figure 6. Comparison of codes in GML and SVG.

4 SPATIAL OPERATORS AND SSESQL

This section focuses on how to carry out spatial data query on GML and SVG. It is important to note that XQuery is often employed to query XML-based data, and is also suitable for querying GML/SVG data. However, XQuery has a very special and complicated syntax, and has not become familiar to many people. In contrast, SQL (Structured Query Language) has a relative simple and intuitive syntax, and has been familiar to many technical users. More importantly, SQL is more powerful in querying data. Thus, in consideration of end users' acceptance and the processing capabilities of SQL, an extended SQL is employed for spatial data query on GML and SVG.

Several efforts have been made, attempting to make spatial extensions to SQL (Lin & Huang 2001, Egenhofer 1994). These extended SQL introduce spatial data types (e.g., point, line and polygon) and spatial operators, allowing users to inquire spatial features, primarily in terms of spatial relationships and metric constraints (Lin & Huang 2001). It is widely acknowledged that these spatial operators and SQL like languages can be used for spatial analysis (Frank 1984).

According to Section 3, spatial information is organized as "map - layer - spatial object" structure. A map represented by SVG/GML can be viewed as a database, the layers as tables of the database, the attributes (spatial and non-spatial) of spatial objects in a layer as columns of the corresponding table, and spatial objects as records of the corresponding table. Thus SQL like languages can be used for spatial query and analysis on SVG and GML.

In this section, we design some spatial operators and integrate them into an SVG-based Spatial Extended SQL (SSESQL). The SSESQL uses the basic spatial data types discussed in Section 3: Point, Curve, Surface, Multipoint, Multicurve, Multisurface and Multigeometry. It can be used on both server side and browser side for spatial query and analysis on GML and SVG.

4.1 *Spatial operators*

Spatial operators are mainly designed to access spatial attributes, calculate spatial relationships, and perform geometrical operations. We introduce five types of operators: *attribute access operators* (GeometryType, Centroid, Length, Area, and Envelope), *spatial topological operators* (Disjoint, Touch, Crosses, Within, Overlap, and Contain), *spatial order operators* (East, East_South, South, West_South, West, West_North, North, and East_North), *spatial metric operators* (Max_Dist, Min_Dist, and Mean_Dist), and *geometrical operators* (Intersection, Union, Difference, and Buffer).

These five types of operators can meet the basic needs of spatial analysis. For network analysis, we can use Touch operator and Length operator to find out the touched spatial object (e.g., road) and the distance. We can use Buffer operator and topological operators to carry out buffer analysis, such as "which cities are around 100km of the Danube River". Also, we can use Intersection, Union and Difference operators to carry out overlay analysis.

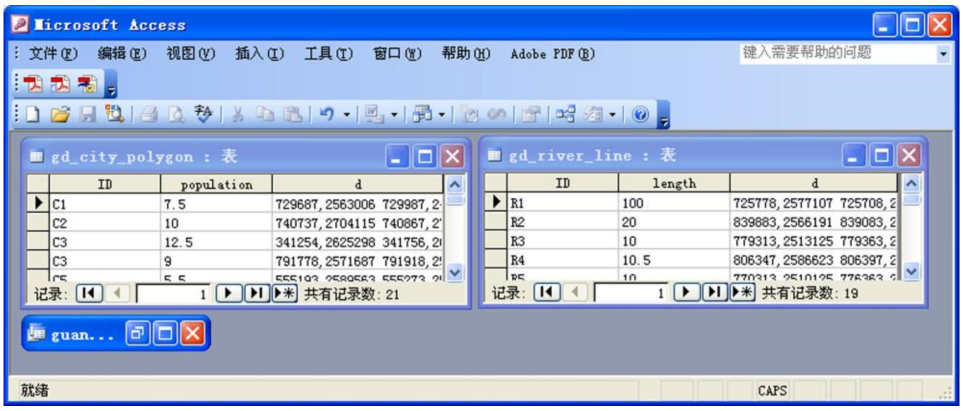
It is important to note that, the above spatial operators can be used as APIs (Application Programming Interface) and integrated into other programs. However, we integrate them into a SVG-based Spatial Extended SQL (SSESQL) to support end users' interaction.

More detail of the spatial operators can be found in Huang et al. (2008).

4.2 *SSESQL and some query examples*

As SSESQL is designed for spatial query, there is no need to consider data insert, update and delete. As a result, we integrate the above spatial operators to the original SELECT clause of SQL. The EBNF (Extended Backus-Naur Form) of SELECT clause of SSESQL can be found in Huang et al. (2008).

The following are some query examples. The two layers represented in the Figures 5-6 can be viewed as the following tables: gd_city_polygon (ID, population, d) and gd_river_line (ID, length, d) in Figure 7.



ID	population	d
C1	7.5	729687,2563006 729987,2
C2	10	740737,2704115 740867,2
C3	12.5	341254,2625298 341756,2
C3	9	791778,2571687 791918,2
C5	5.5	555193,2589563 555273,2

记录: 1 共有记录数: 21

ID	length	d
R1	100	725778,2577107 725708,2
R2	20	839883,2566191 839083,2
R3	10	779313,2513125 779363,2
R4	10.5	806347,2586623 806397,2
R5	10	779313,2510125 776363,2

记录: 1 共有记录数: 19

Figure 7. Tables of gd_city_polygon and gd_river_line shown in Microsoft Access.

1) Query example 1: Lists the neighbor cities of city "C1".

```
SELECT city1.ID AS Neighbors_of_C1
FROM gd_city_polygon city1, gd_city_polygon city2
WHERE Touch (city1.d, city2.d)=True AND city2.ID='C1'
```

2) Query example 2: Lists the cities which are crossed by the river "R1".

```
SELECT c.id AS cid
FROM gd_river_line r, gd_city_polygon c
```

WHERE r.id=”R1” AND Crosses(r.d, c.d)=True;

3) Query example 3: There is some toxic contamination throughout the River “R1”. This toxic contamination affects the cities, which are 100km around the river. Please list all affected cities.

SELECT c.id
FROM gd_river_line r, gd_city_polygon c
WHERE Overlap(c.d, Buffer(r.d, 100000))=True AND r.id=”R1”;

Since the GML-based spatial information representation model uses the same conceptual model and spatial data structure (logical model) as the SVG-based spatial information representation model, we can also apply the SSESQL to carry out spatial queries and analyses on GML-based spatial data on the server side. The only difference in applying SSESQL for spatial analysis on SVG and GML is the implementation of the SSESQL compiler.

5 LOAD BALANCING MIDDLEWARES

In this section, we focus on designing the load balancing middlewares to dispense a spatial query to either the server side or the browser side based on the cost of that spatial query.

5.1 *General principle*

The general principle of the load balancing algorithm is to compare the costs of server side solution (C_{server}) and browser side solution ($C_{browser}$). If C_{server} is less than $C_{browser}$, execute the spatial query on the server side, and send the result data to the browser. Otherwise, send the data (input data) to the browser, and carry out the spatial query on the browser side.

The cost of a spatial query includes the computational cost (execution of the spatial query) and network transmission cost (input data and output data of spatial operations) and both of them are often measured by delay. As a normal PC’s processor performance has been drastically improved, the difference between the server side and the browser side’s computational cost becomes minimal or less significant for most of the spatial queries. Therefore, the difference between C_{server} and $C_{browser}$ mainly depends on the network transmission cost. Our discussions below will focus on comparing the network transmission cost.

5.2 *Architecture*

In order to compare the network transmission costs, we design middlewares for both server side and browser side. Figure 8 shows the architecture.

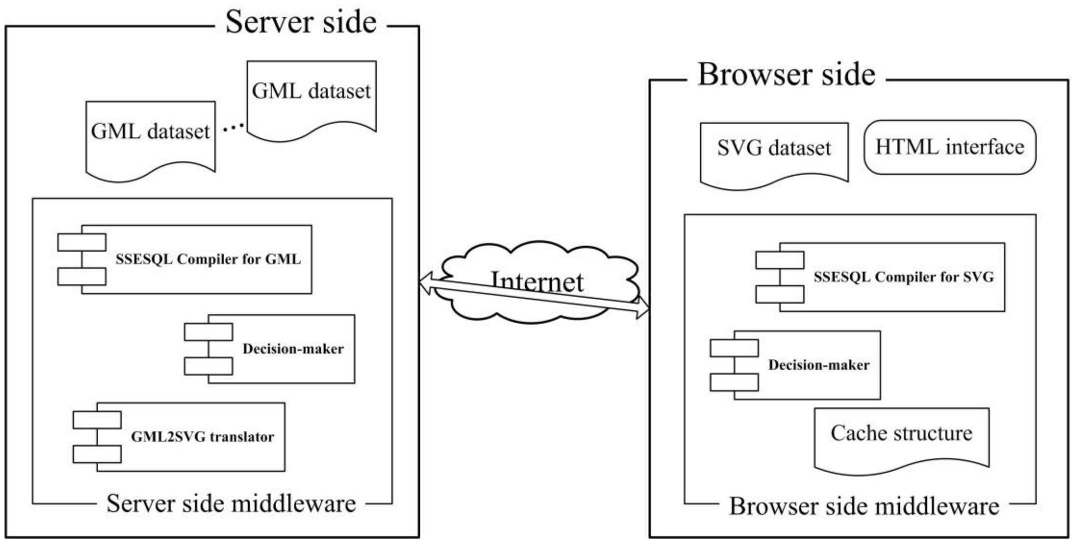


Figure 8. Architecture of the load balancing middlewares.

5.2.1 *Browser side's middleware*

The browser side's middleware provides a *SSESQL compiler for SVG*, a *cache structure*, and a *decision-maker*. The SSESQL compiler carries out syntax, sentence, and semantic analysis for users' SSESQL sentences. The cache structure records which spatial data have been delivered to the browser side. Currently, we choose a very coarse granularity for organizing and transmitting the spatial data: layer as a unit of organizing and transmitting spatial data. The following is the data structure of the cache structure.

```
Class cache_structure
{
   String objectID; // the ID of the layer
   Bool bExecuted; //True or False, whether the layer is created by browser side' execution,
                   // bExecuted = True means that the layer can only be found on the
                   //browser side.
   Bool bTransmitted; //True or False, whether the layer has been delivered to the browser
}
ArrayList< cache_structure > cache; // record the transmitted data
```

The decision-maker figures out the needed spatial data (layer) from the FROM clause of the SSESQL sentence (spatial query) by invoking the SSESQL compiler, and checks whether all the needed spatial data have been delivered to the browser side (by checking the cache structure). If necessary, the decision-maker sends the request to the server side's middleware.

5.2.2 *Server side's middleware*

The server side's middleware includes an *SSESQL compiler for GML*, a *GML2SVG translator*, and a *decision-maker*. The SSESQL compiler carries out syntax, sentence, and semantic analysis for users' SSESQL sentences to query GML data on the server side. The GML2SVG translator is employed to translate GML data into SVG data. The decision-maker is responsible for receiving the requests from the browser side's middleware, and employing some inference rules to dispense a spatial query to either the server side or the browser side based on the network transmission cost of that spatial query.

5.3 *Inference rules in the decision-makers*

Every time a user submit a spatial query (by SSESQL) through their browser (e.g., Internet Explorer), the browser side's decision-maker will figure out the needed input data (layer) from the FROM clause in SSESQL sentences, and check whether the needed spatial data have been delivered to the browser side. If all the needed input data are available on the browser side, this spatial query will be executed by the SSESQL compiler on the browser side. Otherwise, the spatial query sentence, the names (IDs) of the needed input data (only those which has not been delivered to the browser side, i.e., bTransmitted = False), and the data size $S_{browser}$ of the other input data whose bExecuted is True (the data can only be found on the browser side) will be sent to the server side.

The server side's decision-maker receives the information, and employs some inference rules for the dispensation. In order to identify these inference rules, we analyze the spatial operations provided for GML and SVG in Section 4. We find that only the "Buffer" and "Union" spatial operators often result in more data output than input. Therefore, we design the following inference rules:

1) All the needed input data are available on the server side: if the needed output data do not involve with "Buffer" and "Union" spatial operators (it can be figured out from the SELECT clause in SSESQL sentences), the spatial query will be carried out by the SSESQL compiler on the server side, and the result (output) data will be sent to the browser side. Otherwise, the needed input data will be translated into SVG, and sent to the browser side, the spatial query will be executed by the SSESQL compiler on the browser side.

2) The needed input data are located on different sides (browser side and server side): if the data size S_{server} of the needed input data (only those which have not been delivered to the browser side) is smaller than $S_{browser}$ (the data size of the needed input data which can only be found on the browser side), the needed input data will be sent to the browser side, and the spatial query will be executed by the SSESQL compiler on the browser side. Otherwise ($S_{server} > S_{browser}$), those needed input data which can only be found on the browser side will be sent to the server side, and the spatial query will be executed by the SSESQL compiler on the server side; after that, the result (output) data will be sent to the browser side.

For all the above cases, the cache structure on the browser side's middleware will be updated. It is important to note that the above inference rules are still very simple, and need to be improved further.

6 IMPLEMENTATION, CASE STUDIES AND DISCUSSIONS

In this section, we discuss how to implement the proposed method. In order to evaluate the method, we design three case studies, and compare different solutions (server side, browser side, and load balancing) for accomplishing these tasks.

6.1 *Implementation*

Most of the spatial operators in Section 4 involve geometrical computation. The field *computational geometry* provides rich collection of algorithms for solving pure geometrical problems. As well, Java already provides some basic computational geometry APIs (Java 2D API). We implement the spatial operators with algorithms of computational geometry and Java 2D API. For the SSESQL, a compiler is needed to carry out syntax, sentence, and semantic analysis for SSESQL sentences. Levine et al. (1992) discussed how to implement an SQL compiler and provided some c++ codes. As our SSESQL is based on the original SQL, we adapt their c++ codes, and implement our own SSESQL compiler by combining the spatial operators. Currently, we do not introduce *query optimization* into our compiler. However, as our SSESQL only introduces some spatial operators into the original SQL, all the technologies for query optimization can be used to improve the performance of our SSESQL compiler.

For the server side's middleware, spatial operators, SSESQL compiler, GML2SVG, and decision-maker are implemented as Java Servlets. For the browser side's middleware, spatial operators, SSESQL compiler, decision-maker and the cache structure are developed as Java Applets, and embedded in HTML. A user interface is also embedded in HTML for inputting SSESQL query sentences (Fig. 9). We use JavaScript to access SVG document and invoke the Java applets, which can help to receive users' input and execute the spatial query on the browser side. In order to facilitate the interaction between server side and browser side's middlewares, we use AJAX (Asynchronous JavaScript and XML) technology (mainly XMLHttpRequest Object). Users can access spatial analysis functions simply with a web browser (such as Internet Explorer) which has an SVG plug-in or SVG viewer (such as Adobe SVG Viewer). In the last several years, more and more web browsers (Firefox, Opera, Google Chrome, etc) start to provide native SVG support (Wikipedia 2010).

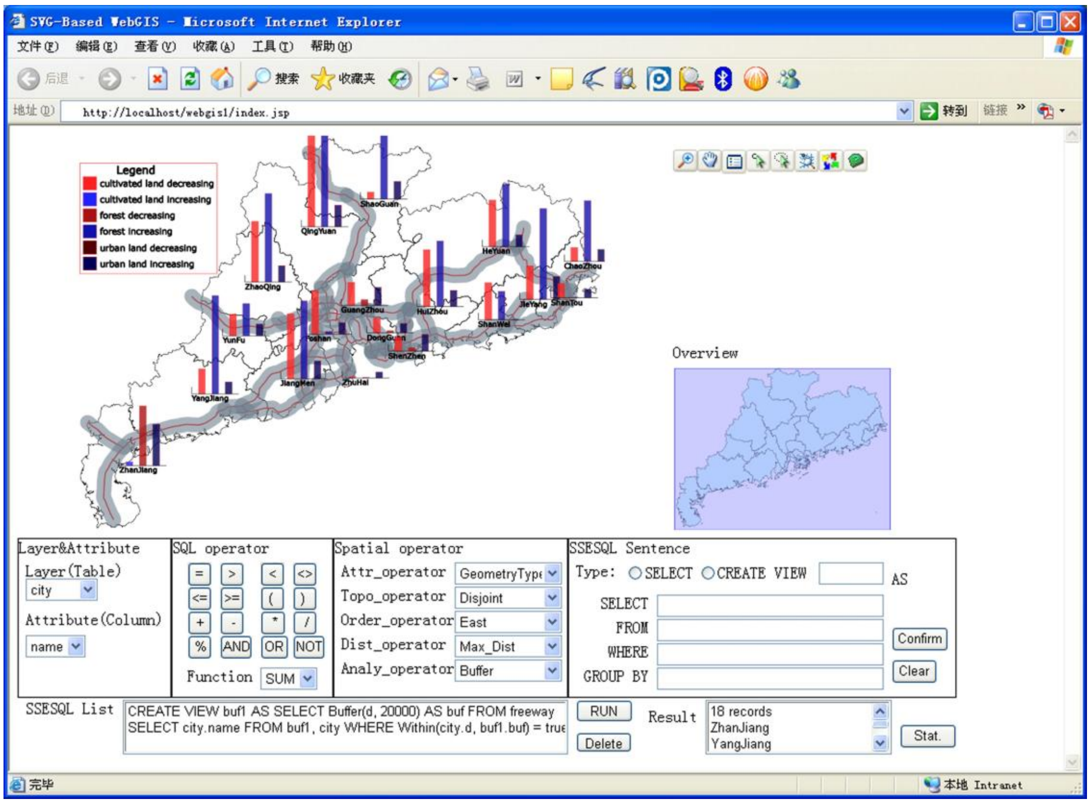
6.2 *Case studies*

We design several case studies to evaluate the proposed method. It is important to note that a good design of spatial analysis procedure is vital to a spatial analysis task. This means that users (domain experts) do have to plan the analysis task carefully, even if they have the best spatial analysis tool.

The cultivated lands are very important in a highly populated area like Guangdong Province in China. Our case studies focus on the problem of whether the land uses change with the growth of transportation networks.

Our first case study tries to find out *how the land uses along the freeway change in Guangdong Province between 1987 and 1996*. In order to investigate this issue, we have to define a buffer for the freeway, and find out which cities are located in this buffer, and then analyze the changes of land uses for these cities. We carry out this task based on the workflow described in Figure 1. First, we identify the needed data and the evaluation criteria by carefully analyzing this case study. And then based on the suggested model in Figure 3, we use GML to represent the needed spatial data (freeway and city layers) on the server side. We also represent the district boundary layer in SVG and deliver it to the browser side as the initial User Interface (UI). We then submit SSESQL sentences on the browser side to carry out the spatial queries by the following steps: 1) Calculate a buffer of freeway (using Buffer operator). In this case, we use 20km as the buffer size. 2) Find out which city centers are located in this buffer (using Within operator). 3) Use the statistics function to generate the bar graphs of changes of land uses for every identified city. While working completely transparent to the end user, the load balancing middlewares dispense the SSESQL sentences (requests) to either the server side or the browser side based on the cost of those requests automatically.

Figure 9 shows the user interface and the result of this case study. Functions of zoom in/out, pan, layer control, query, and statistics are also added to this interface. The names of cities involved in the calculation are listed in the box shown at the right-bottom corner.



Layer(Table): city
Attribute(Column): name
Function: SUM
Attr_operator: GeometryType
Topo_operator: Disjoint
Order_operator: East
Dist_operator: Max_Dist
Analy_operator: Buffer

```
CREATE VIEW buf1 AS SELECT Buffer(d, 20000) AS buf FROM freeway
SELECT city.name FROM buf1, city WHERE Within(city.d, buf1.buf) = true
```

```
18 records
ZhanJiang
YangJiang
```

Figure 9. The land uses change along the freeway (Legend for bar graphs: red for cultivated lands decreasing; blue for cultivated lands increasing; dark red for forest decreasing; dark blue for forest increasing; brown for urban land decreasing; dark grey for urban land increasing).

As can be seen from the bar graphs, there is a decreasing trend of cultivated lands and an increasing trend of urban land during 1987 to 1996 in all related cities. These trends are coincident with the situation of Guangdong Province between 1987 and 1996. At that time, with the growth of its transportation networks, especially freeway, all the cities were greatly expanded by converting cultivated lands into urban lands. However, there is an increasing trend of the forest between 1987 and 1996. An explanation for this would be that the Guangdong Province government announced a policy for forest in 1985: “Create a Green Guangdong in 10 Years” (Guangdong Forest 2010). From this case study, we can understand that land uses change with the growth of transportation networks. In order to draw some quantitative conclusions, more case studies should be done by using our proposed method.

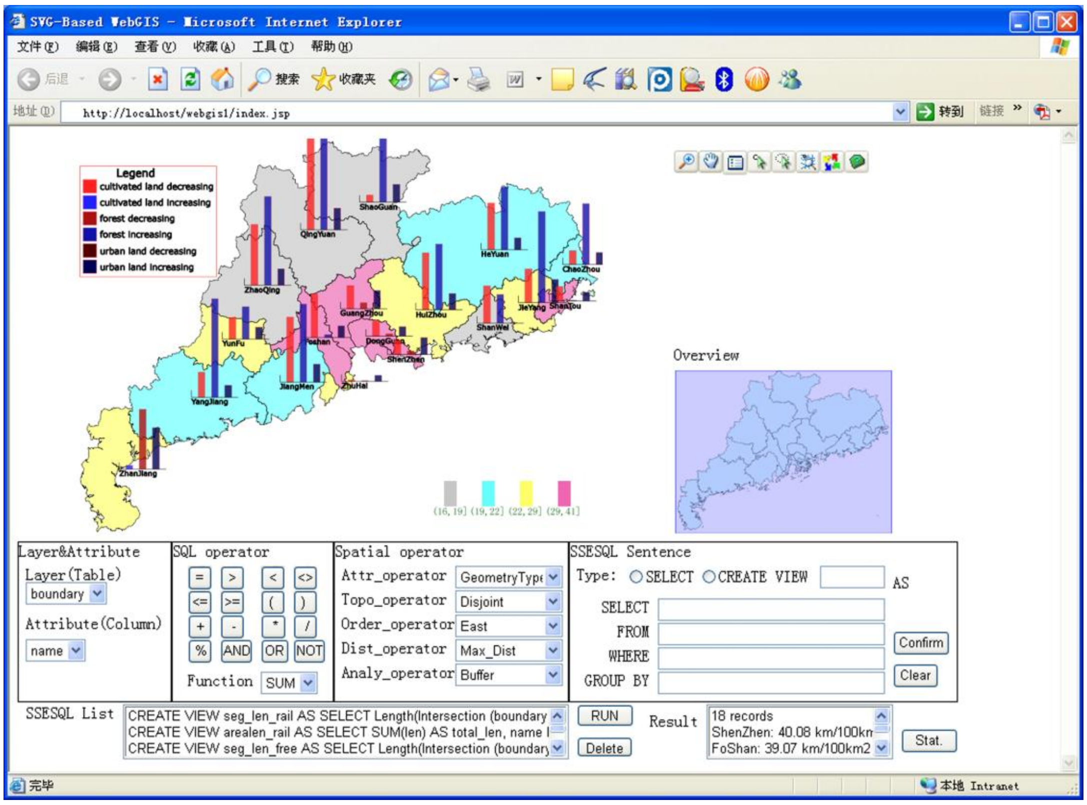
In order to illustrate the benefit of introducing load balancing technology, we compare different solutions (server-side solution, client-side solution, and load balancing solution) for carrying out this task. We mainly compare the data amount of the network transmission (excluding the request/response sentences) between the server side and the browser side. Table 2 depicts the result. As can be seen from the table, our load balancing solution has a smaller network transmission load between server side and browser side.

Table 2. Comparison of the first case study (data amount is measured by Byte).

	Server-side solution	Client-side solution	Load balancing solution (our solution)
Step1: Buffer	256,347B	92,122B	92,122B (executed on the browser)
Step2: Within	741B	2,043B	2,043B (executed on the browser)
Step3: Stat.	2,168B	11,839B	2,168B (executed on the server)
Total	259,256B	106,004B	94,293B

Our second case study tries to address *the relationship between changes of land use and road (railway, freeway, and province level road) density in Guangdong Province*. Similar to the first case study, at the beginning, we only deliver the district boundary layer to the browser side as the initial UI. The task is carried out in the following steps: 1) Calculate the total road length for every city (using Interaction and Length operators). 2) Calculate the area for every city (using the Area operator). 3) Calculate the road density for every city and color every city accordingly (using SQL's "/" operator). 4) Use the statistics function to generate bar graphs of changes of land use for every city. The load balancing middlewares dispense the SSESQL sentences to either the server side or the browser side based on the cost of those requests automatically.

Figure 10 shows the result from this case study. The result lists the name of cities and their road density in the right-bottom box. Every city is marked with different color according to its road density, and with a statistic bar graph showing changes of land use. The legend of the bar graph is the same as that in Figure 9. As a result, we can make the following qualitative conclusion: the changes of land usediffer with different road density.



CREATE VIEW seg_len_rail AS SELECT Length(Intersection (boundary
CREATE VIEW arealen_rail AS SELECT SUM(len) AS total_len, name I
CREATE VIEW seg_len_free AS SELECT Length(Intersection (boundary

18 records
ShenZhen: 40.08 km/100km
FoShan: 39.07 km/100km2

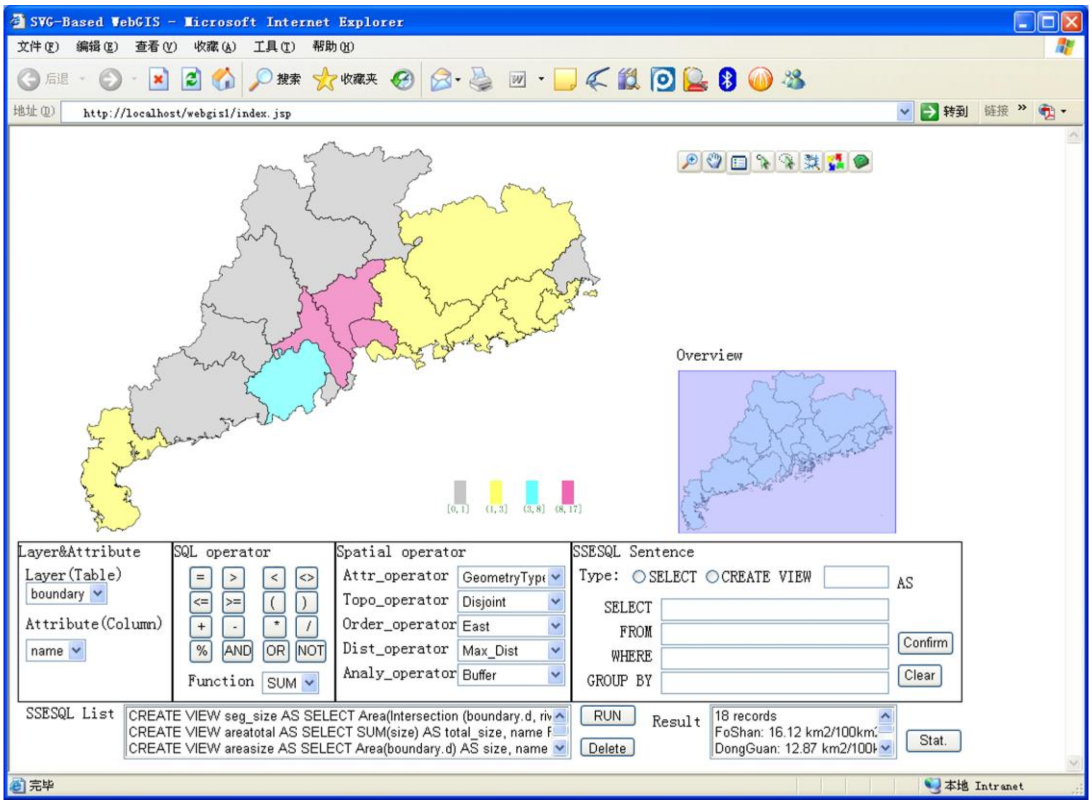
Figure 10. The relationship between changes of land uses and road density.

We also compare the amount of transmitted data between different solutions for accomplishing this task. Table 3 depicts the result.

Table 3. Comparison of the second case study (data amount is measured by Byte).

	Server-side solution	Client-side solution	Load balancing solution (our solution)
Step1: length	1,336B	1,125,073B	4,381B
Step2: area	1,345B	0B	0B (executed on the browser)
Step3: density	1,774B	0B	0B (executed on the browser)
Step4: statistics	2,168B	11,839B	2,168B (executed on the server)
Total	6,623B	1,136,912B	6,549B

We also use our proposed method to calculate *the river density (double line river) in Guangdong Province*. Figure 11 shows the result. In this case study, the city with the highest river density is Foshan city, located in the Pearl River Delta which is the most economically dynamic region in China, whose river density is *16.12 km^2/100km^2*. Other cities (Guangzhou, Dongguan, and Zhuhai) in the Pearl River Delta also have very high river density. Table 4 depicts the similar comparison among different solutions for accomplishing this task.


SVG-Based WebGIS - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退
搜索
收藏夹
地址(D)
http://localhost/webgis1/index.jsp
转到
链接
Overview
[0, 1]
(1, 3]
(3, 8]
(8, 17]
Layer&Attribute
SQL operator
Spatial operator
SSESQL Sentence
= > < <> <= >= ( ) + - * / % AND OR NOT
Type: SELECT CREATE VIEW AS
SELECT
FROM
WHERE
GROUP BY
Confirm
Clear
RUN
Delete
Stat.
完毕
本地 Intranet

Layer(Table): boundary
Attribute(Column): name
Function: SUM
Attr_operator: GeometryType
Topo_operator: Disjoint
Order_operator: East
Dist_operator: Max_Dist
Analy_operator: Buffer

SSESQL List

```
CREATE VIEW seg_size AS SELECT Area(Intersection (boundary.d, riv
CREATE VIEW areatotal AS SELECT SUM(size) AS total_size, name F
CREATE VIEW areasize AS SELECT Area(boundary.d) AS size, name
```

Result

```
18 records
FoShan: 16.12 km2/100km
DongGuan: 12.87 km2/100k
```

Figure 11. The river density (double line river) in Guangdong Province.

Table 4. Comparison of the third case study (data amount is measured by Byte).

	Server-side solution	Client-side solution	Load balancing solution(our solution)
Step1: river area	1,333B	439,454B	2,206B
Step2: city area	1,345B	0B	0B (executed on the browser)
Step3: density	1,771B	0B	0B (executed on the browser)

Total	3,449B	439,454B	2,206B

The implementation of the above case studies shows that our suggested method for SVG-based spatial analysis is feasible and operable. Although we use some very simple and static inference rules for dispensing a spatial query to either the server side or the browser side, and a quite coarse granularity (by layer) for organizing and transmitting spatial data, our load balancing spatial analysis solution decreases the amount of data transmitted over the network between the server side and the browser side, and accordingly improves the overall performance. Our method enables users to access spatial analysis functions simply with a web browser (such as Internet Explorer and Firefox) with SVG support. This greatly improves the functions of current WebGIS applications, most of which have been only employed for web mapping.

To summarize, our solution has a good support for cross-platform (e.g., Microsoft Windows and Linux) and cross-browser (e.g., Internet Explorer, FireFox and Opera). As a result, our proposed method is especially suitable for providing spatial analysis functions to heterogeneous Web clients, such as to the public users who may use different platforms and different web browsers and to a company with a lot of subsidiary companies distributing on different places. At this moment, we are cooperating with some municipal Land and Resource Administration Bureaus in China which are interested in sharing spatial data and spatial analysis functions with their branches. The method can also be extended to support spatial analysis on mobile devices (e.g., PDA and smart phones), which would be very useful for field work.

7 CONCLUSIONS AND FUTURE WORK

Currently, most WebGIS applications in general and SVG-based spatial applications in particular only offer web mapping function, and do not provide spatial analysis functions that are vital to spatial information applications. This chapter focuses on introducing load balancing spatial analysis into XML/GML/SVG-based WebGIS. We proposed that the decision on where to execute spatial query operations (server side or browser side) should be based on the network communication cost versus the computational cost, mainly focusing on the network communication cost.

The contributions of this chapter are: 1) identification of the key issues of load balancing spatial analysis in WebGIS, 2) a lossless GML2SVG translator according to the GML/SVG based spatial information representation model, 3) design of load balancing middlewares to dispense spatial operations to the server side or the browser side, which reduce the network transmission load, 4) improved functionality of current WebGIS applications, especially XML/GML/SVG based WebGIS which have been mostly employed for web mapping only.

Our next step is to make more investigation on the granularity of organizing and transmitting spatial data, and to develop more flexible and precise inference rules for dispensing spatial queries to the server side or the browser side. We are also interested in introducing query optimization to improve the performance of our SSESQL compiler. Additionally, more complex case studies will be carried out to evaluate our suggested method.

ACKNOWLEDGMENT

This work has been supported by the National Natural Science Foundation of China (project no. 60842007), and the Science and Technology Department of Guangdong Province (project no. 2002B32101, 2004B32501001, 2005B30801006). Haosheng Huang and Georg Gartner also thank the support from the project UCPNavi funded by the Austrian FWF. We are also grateful to our anonymous reviewers and editors for their truly helpful comments.

REFERENCES

Chen, S. Lu, X. & Zhou, C. 2001. *Introduction of GIS* (in Chinese). Beijing: Science Publish.

Egenhofer, M. 1994. Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 6(1): 86-95.

Frank, A. 1982. Mapquery-database query languages for retrieval of geometric data and its graphical representation. *ACM Computer Graphics* 16(3): 199-207.

Guangdong Forest 2010. Guangdong Forest's 30 years. http://www.gdf.gov.cn/index.php?controller=front&action=view&id=10005572 (in Chinese), Accessed in Mar. 2010.

Guo, Z. Zhou, S. Xu, Z. & Zhou, A. 2003. G2ST: A novel method to transform GML to SVG. *Proc. the 11th ACM GIS,* New Orleans, 7-8 November 2003. New York: ACM.

Herdy, K. Burggraf, D. & Cameron, R. 2008. High performance GML to SVG transformation for the visual presentation of geographic data in web-based mapping systems. *Proc. the 6th International Conference on Scalable Vector Graphics,* Nuremberg, 26-28 August 2008. http://www.svgopen.org/2008/?section=abstracts_and_proceedings, Accessed in Mar. 2010.

Huang, H. Li, Y. & Gartner, G. 2008. SVG-based spatial information representation and analysis. *Proc. W2GIS 2008*, Shanghai, 11-12 December 2008. Berlin Heidelberg: Springer.

Huang, H. & Li, Y. 2009. Load balancing spatial analysis in XML/GML/SVG based WebGIS. *Proc. ESIAT 2009*, Wuhan, 4-5 July 2009. Los Alamitos: IEEE Computer Society.

Jeong, C. Chung, Y.,Joo, S. & Lee, J. 2006. Tourism guided information system for Location-Based Services. In H.T. Shen et al. (ed.), *APWeb Workshops 2006*, Harbin, 16-18 January 2006. Berlin Heidelberg: Springer.

Levine, J. Mason, T. & Brown, D. 1992. *Lex & Yacc (2nd)*. O'Reilly & Associates.

Lin, H. & Huang, B. 2001. SQL/SDA: A query language for supporting spatial data analysis and its web-based implementation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 13(4): 671-682.

Longley P.A. M. F. Goodchild, D. J. Maguire & D. W. Rhind. 2005. *Geographic information systems and science (second edition)*. Chichester: John Wiley.

Luo, Y. Wang, X. & Xu, Z. 2003. A dynamic load balancing policy for agent-based DGIS. *Proc. Asia GIS Conference 2003, Wuhan, 16-18 October 2003*.

Köbben, B. 2007. RIMapperWMS: a web map service providing SVG maps with a built-in client. In S.I. Fabrikant & M. Wachowicz (ed.), *The European Information Society: Leading the Way with Geo-information*. Berlin Heidelberg: Springer.

Neumann, A. & Winter, A. 2010. Cartographers on the net. http://www.carto.net, Accessed on Mar. 2010.

OGC. 1999. OpenGIS simple features specification for SQL (Revision 1.1). http://www.opengeospatial.org/standards/sfs, Accessed in Mar. 2010.

OGC. 2003. Geography Markup Language, http://www.opengeospatial.org/standards/gml, Accessed in Mar. 2010.

Peng, Z. & Zhang, C. 2004. The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS). *Journal of Geographical Systems* 6(2): 95-116.

Qin, G. & Li, Q. 2007. Dynamic resource dispatch strategy for webgis cluster services. *Proc. CDVE 2007,* Shanghai, 16-20 September 2007. Berlin Heidelberg: Springer.

Shekhar, S. Coyle, M. Goyal, B. Liu, D. & Sarkar, S. 1997. Data models in geographic information systems. *Communications of the ACM* 40(4): 103-111

SuperMap. 2010. SuperMap IS.NET 2008. http://www.supermap.com.cn/gb/products/fwskf.htm, Accessed in Mar. 2010.

Tennakoon, W.T.M.S.B. 2003. Visualization of GML data using XSLT. Master thesis of International Institute for Geo-Information Science and Earth Observation.

W3C. 2003. Scalable Vector Graphics (SVG) 1.1 Specification, http://www.w3.org/TR/SVG11/, Accessed in Mar. 2010.

Wang, P.Yang, C. Yu, Z. & Ren, Y. 2004. A load balance algorithm for WMS. *Proc. Geoscience and Remote Sensing Symposium 2004 (IGARSS'04)*, Anchorage, 20-24 September 2004. Piscataway: IEEE.

Wikipedia. 2010. Scalable Vector Graphics. http://en.wikipedia.org/wiki/Scalable_Vector_Graphics, Accessed in Mar. 2010.

Wu, X. 2002. *Principles and methods of GIS* (in Chinese), Beijing: Publishing House of Electronics Industry.