A Distributed GIS for Managing Shanghai Landscape Resources

Yue Zhu [1,2], Chaowei Yang[1], David W. Wong[1], Menas Kafatos[1]

[1]Earth Systems and GeoInformation Sciences, School of Computational Sciences, George Mason University
E-mail:{yzhu, cyang3, dwong2, mkafatos}@gmu.edu
[2]Shanghai CityGIS Developing Corp., 75 Wanping Nan Road, Xuhui District, Shanghai, China

Abstract
Given the decentralized computing environment for managing landscape resources in Shanghai, China, this paper introduces a Distributed GIS application to support a more efficient and effective approach for resource management. Four critical computing issues related to such a Distributed GIS are addressed: (1) large image management, (2) time dimension management, (3) network communication of geospatial information within the computer network, and (4) spatial data access through a spatial data engine. This paper suggests possible solutions to these four issues and illustrates how general landscape management functions are implemented through a Distributed GIS. This paper also offers some insights on the design and development of a Distributed GIS.

Keywords
distributed GIS, time scaled spatial data model, image library, landscape

I. INTRODUCTION

Managing the natural landscape of a metropolitan area has attracted increasing attention, especially from the public. How to efficiently manage the landscape using information technology is becoming a very important issue. In the past decade or so, many local governments have started employing Management Information Systems (MIS) and/or Geographic Information Systems (GIS) to manage the urban landscape environment. However, both types of systems encounter a similar set of problems: How to collect the physically dispersed landscape data continuously over time and maintain the data available online during their lifetime? These difficulties are both administrative and technological issues. An operational distributed information system with typical GIS and MIS functions is needed to deal with these problems. The same set of problems has also posted difficulties to other government agencies, such as public works, environmental and sanitation management, and transportation. Therefore, finding solutions to efficiently manage the landscape resources using MIS and/or GIS can also lead to successes in other public management organizations. In this paper, we describe our activities in designing and developing an efficient and practical information system to manage the Shanghai landscape resources based on Distributed GIS.

Since its inception in 1960s, GIS has been widely used in numerous government projects. Most existing GIS applications in different government agencies manage large volumes of spatial data through traditional methods with spatial data classified by themes and layers, and stored in a local database or local files. These valuable spatial data are isolated from each other. In order to utilize the available spatial data efficiently and effectively, GIS has to go beyond just collecting and storing data to provide information services to potential users. However, modern GIS history is marked by the fusion of the Internet. The Internet, especially its applications of World Wide Web (WWW), provides an ideal platform to empower the public and government agencies with GIS technology through Distributed GIS (Plewe, 1997; Peng and Tsou, 2003; Yang, Yang and Kafatos, 2002). One of the most critical issues in developing a Distributed GIS is to decide where to allocate different processing components (Yang, 2000), such as data processing, analysis, and visualizing. The ultimate goal of this exercise is to obtain a satisfactory level of performance for users. Many researchers (e.g., Kenneth and Kirvan, 1997; Peng, 1999; Yang, 2000) suggest the Client/Server (C/S) computing model to organize or integrate different computing components, which may be connected by different networks (such as coaxial cable, optical fiber, phone line). A typical C/S distributed system includes three key elements: the client, the server and the network connecting them. The client interacts with users and server(s), and may also perform certain data processing functions. The server stores and provides data, and performs some geospatial computing processes, the result of which will then be returned to a client. The network hosts the transmission of data between the client and the server. Within such a distributed system, the data structure and spatial database organization for storing the distributed and heterogeneous data are very important technical issues (Gong et al., 1998; Wu and Zhang, 2002).

Among various types of network GIS, WebGIS is currently the most extensively developed and widely used. There are already many successful examples of WebGIS, such as Mapquest (http://www.mapquest.com/), Terraserver (http://www.terraserver.com/), the Weather Channel (http://www.weather.com/), and others that have been widely used for online route selection,city planning(Deng, 2000), environmental exploration (Dragicenvic, 2000), watersheds management (Kelly, 2001), land use planning (Bogner et al., 2001), road/rail construction (Barthello and Pollack, 2001;

1082-4006/05/1101-29$5.00

Slater, 2002), business analysis (Shen, 2001), airport construction (Galinao and Brennan, 2002), and data integration and dissemination (Eichelberger, 2001; Takatsuka and Gahegan, 2002). We follow the paths of these successful stories to adopt the Distributed GIS technology, specifically WebGIS, to develop the Shanghai Landscape Management Information System. Also, The C/S architecture for Distributed GIS is detailed here, as well as the Browser/Server(B/S) architecture in which the client side functions are implemented within a web browser.

II. BACKGROUND

There are 19 district bureaus and a central bureau that perform landscape management in Shanghai, China. Before 1998, statistical data were manually calculated and submitted to the central bureau by the end of each year in paper format. The process of collation and correction often lasted to March of the subsequent year, creating significant problems for all statistics staff, in addition to collecting detailed landscape data. In 1999, the central landscape management bureau set up the first GIS-based information system to collect spatial data and related attributes. At that time, there was no mature Distributed GIS solution to support the operation, the developed system did not satisfy some basic requirements, such as keeping data online all the time. Network connections play a vital role in this process, but different districts have highly heterogeneous network environments. For instance, some districts have optical fiber connections, but some have ISDN/ASDL connections. Some districts in the suburbs only have phone-line-based 56kbps connections. All data collected are subject to statistical analyses to provide information to decision makers. Daily operational landscape management tasks also rely on these collected data to provide pertinent information. Therefore, the desired Distributed GIS-based landscape management system should satisfy the following requirements:

- All landscape bureaus should be connected to the Distributed GIS.
- Data should be kept locally for maintenance and updates, but should be available online to the central bureau and other local bureaus.
- Detailed technical issues, such as where data are located, should be transparent to users.
- System performance should be acceptable to most, if not all, users.

Based on these requirements, we designed and developed a logically unified, but physically distributed GIS to support the distributed spatial data collection and dissemination processes. The system can also support other landscape management functions. In section III, we describe the network structure and system architecture. Section IV is dedicated to four computing issues: image data management, time scaled spatial data models, communication implementation, and spatial data Input/Output (I/O) in the main server. These issues are critical to the implementation of such a Distributed GIS. We describe the implementations of the system in section V. Finally, we discuss the remaining issues on the implemented system in section VI.

III. THE NETWORK AND SYSTEM ARCHITECTURE

The design of the system architecture is constrained by the geographic distribution of involved districts and the location of the central bureau. Also, the system architecture has to satisfy the functionality requirements of the system as described in section II.

A. Network architecture

As illustrated in Figure 1, the physical architecture of the network includes the central landscape bureau and two types of districts: those with the governmental high speed optical fiber connections and those with ADSL/ISDN connections. In the central landscape bureau, five servers constitute the central computing facility. These five servers are Web Server, GeoService Server, Data Exchange Server, Office Automation (OA) Server, and Database Server. The Web Server is to support the system's Browser (client)/Server (B/S) structure. The GeoService Server provides distributed spatial data services supporting GIS operations, such as buffering and spatial selection. The Data Exchange server is to maintain the loosely connected databases. The Database Server serves as the central depository for all landscape data, including spatial and attributes data. The OA server is a Linux server, which controls transferring, tracking and storing of all documents.

Districts, such as Huangpu and Luwan, located close to the center of Shanghai city, have stable high speed government optical fiber connections to the central computing facility. Other districts, such as Chongming and Nanhui, are farther from the center of Shanghai city and have only ADSL/ISDN connections to the central computing facility. The ADSL/ISDN connections are less stable and degrade the system performance when real time data update is required. As a result, data from the local bureaus with ADSL/ISDN connections are collected periodically to be stored in the central Data Exchange Server. In each local district, because the management processing functions are not too demanding, only one local server is allocated to support the operation.

B. Logical architecture

Figure 2 illustrates the logical architecture of the Distributed GIS-based system. The left side in the figure represents the operational components at the central computing facility. On the right side are the components in the landscape bureaus of the local districts. Each GeoService in the figure represents the server that resides in a local district. The dotted lines highlight the components supporting the communication and data synchronization mechanisms between the local servers and the central server.

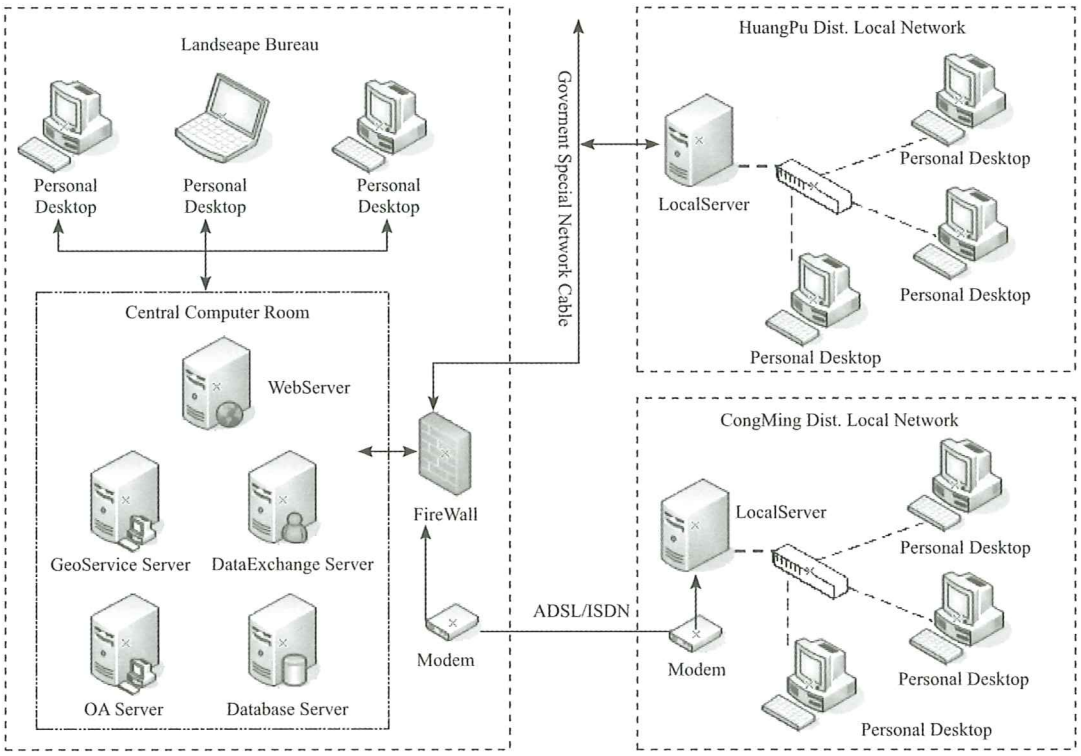

Figure 1. Network architecture

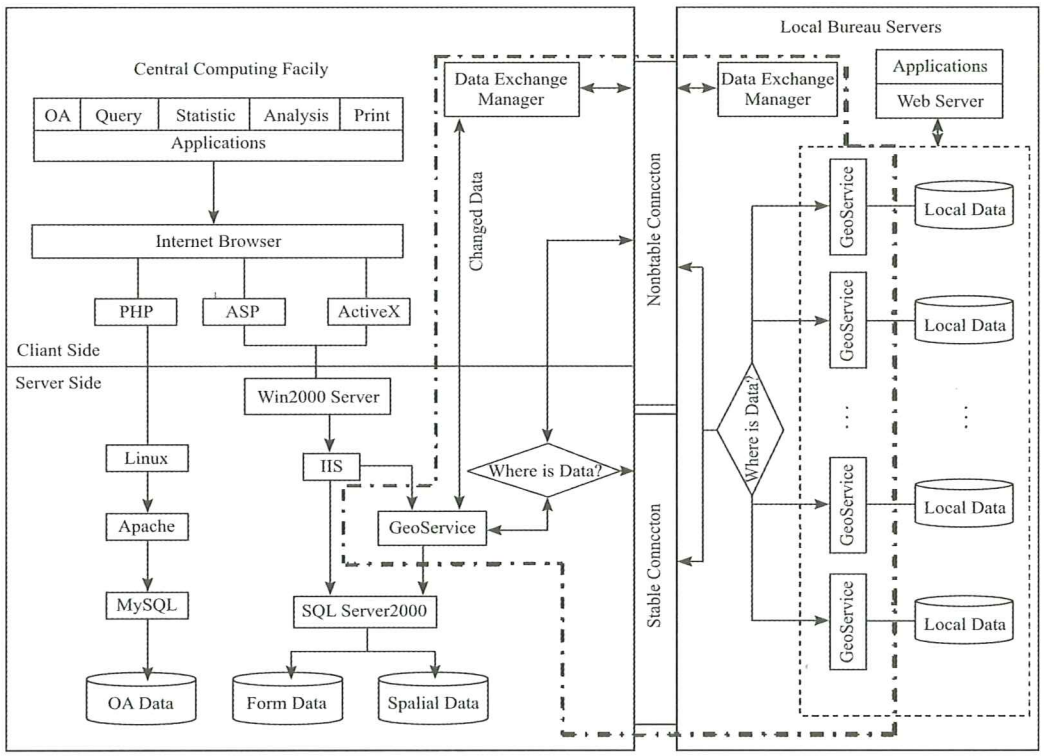

Figure 2. System logical architecture

The entire system is based on the B/S architecture and all system functions are integrated and interfaced using MS Internet Explorer (IE) 6.0 with ASP, PHP and ActiveX Controls. Different types of users are given specific priorities, and have different user environments, which are automatically configured when users log on to the system. Functions supported through the system include three categories: OA system, public website, and landscape business management. Traditionally, these three categories of functions operate independently and there was no need to set up communication protocols among them. However, in this Distributed GIS-based resource management system, data used by one category of functions could be useful to other categories and therefore, should be shared with other function categories. For example, annual statistics about landscape coverage rate produced by the landscape management function can serve as major data inputs for the OA system.

In our system, we attempted to integrate the three function categories (OA system, public website, and landscape management) using shared communication protocols and data storage formats, but focusing on the landscape management system because it is what the Distributed GIS is built for. The landscape management system requires the following major functions:

- Periodically collect landscape data to reflect changes. These spatial data include changes in gardens/parks, roads, tree cover, and other landscape information. ActiveX Control is used to implement the complex spatial data edit, collation, and error detection processes.
- Provide distributed spatial data input interface to ingest landscape data by city, district, or other spatial entities.
- Provide user-friendly tools to support standard and spatial queries, produce various statistical forms/reports and print-outs in standard styles
- Facilitate daily resource management processes such as park management, street green stripe management, and community greenbelt management.

The critical part for maintaining a successful data sharing environment and implementing the information distribution criteria is the communication and data synchronization mechanism highlighted by the dot-dashed frame in Figure 2. Several criteria for data sharing are:

- Each district bureau keeps its own data in the local database.
- Database servers in the central computing facility create a backup copy of the local district data.
- Each district must upload their data at least monthly to the central database through the Data Exchange Manager.
- When data are needed from a remote server, the central computing facility will first activate its GeoService to connect to the GeoService in the remote database server. If the connection attempt fails or there is no optical cable connection, the central database will be accessed. The connectivity information among different remote servers and the central server are maintained in a status table.
- When a local district application needs data from another district, the local application will connect to the GeoService running on the server at the central computing facility. And the central computing facility will coordinate the communication as illustrated in previous step.

IV. COMPUTING ISSUES

There are four critical computing issues to be addressed in the development of the proposed Distributed GIS-based management system: (1) How to manage the aerial images in order to facilitate the digitizing of landscape data? (2) How to organize data to minimize data communications among the local districts and the central server? (3) How to design communication protocols to comply with the criteria discussed in section III? (4) How to support spatial data processing on the server side?

A. Management of large images

There are at least two options to collect landscape data: (1) through ground surveys, using GPSs and PDA's (this option is rather expensive for a large city like Shanghai); and (2) through digitizing data from aerial images. The second option is a relatively inexpensive, yet reasonably accurate method. The issue, then becomes, how to manage and deliver the large images through the network to support the digitizing efforts. We adopted the pyramid architecture proposed by many scholars (including Yang et al., 1999). We tested and found that 500%500 pixels to be an optimal size for an image piece to be transferred in our network environment. With the optimal image size, we can also derive the layer count. Utilizing the image size and layer count parameters, we construct the image library to manage image data. All the pyramid information and the image are stored in a table structure as illustrated in Table 1.

Table 1. Pyramid image storage with a table

Field Name	Description
LevelID	The level this image block belongs to
ImageID	The id in this level
ImageData	Binary image data
SpatialIndexID	Link to external spatial index table
Left	Left X geographic coordinate
Right	Right X geographic coordinate
Top	Top Y geographic coordinate
Bottom	Bottom Y geographic coordinate
ImageWidth	Width of image block
ImageHeight	Height of image block

The process of accessing images from the image database is fairly straight-forward. Once a request for an image is received, a proper level will be calculated and relevant pieces of image at that level will be composed and a 500x500 image will be cut. The specific procedure includes the following steps:

- Calculate the geographic extent of current view and compute a scale ratio between the geographic extent and the screen size.
- Compute the geographic coordinates based on screen coordinates and scale ratio.
- Select an image level that is the closest to the required one.
- Select all image blocks within the level and fill in the geographic extent of the current view.
- Read out these image pieces and integrate them into an image.
- Resample the combined image to the scale of the current view.
- Send the image to the client and display on a screen.

B. Time scaled spatial data model (TSSDM)

One of the most important goals of this system is to develop a mechanism to keep the landscape data online all the time. We designed a "Time Scaled Spatial Data Model" to handle the time dimension of the data. This model is illustrated in Figure 3. A traditional GIS uses a field in the attribute table to record time. It is difficult to extract data based on a specific time because of the lack of historical information of the data. Therefore, many researchers and scholars (Lin, 2002; Cai, 2002) have suggested some time based spatial data models, but most of these models are too complicated and difficult to be implemented. We introduced the "Time Scaled Spatial Data Model" (TSSDM) and used it in the map management engine. Figure 3 illustrates the model, in which each spatial feature has a field as a time stamp to record the date the data were last modified or changed.

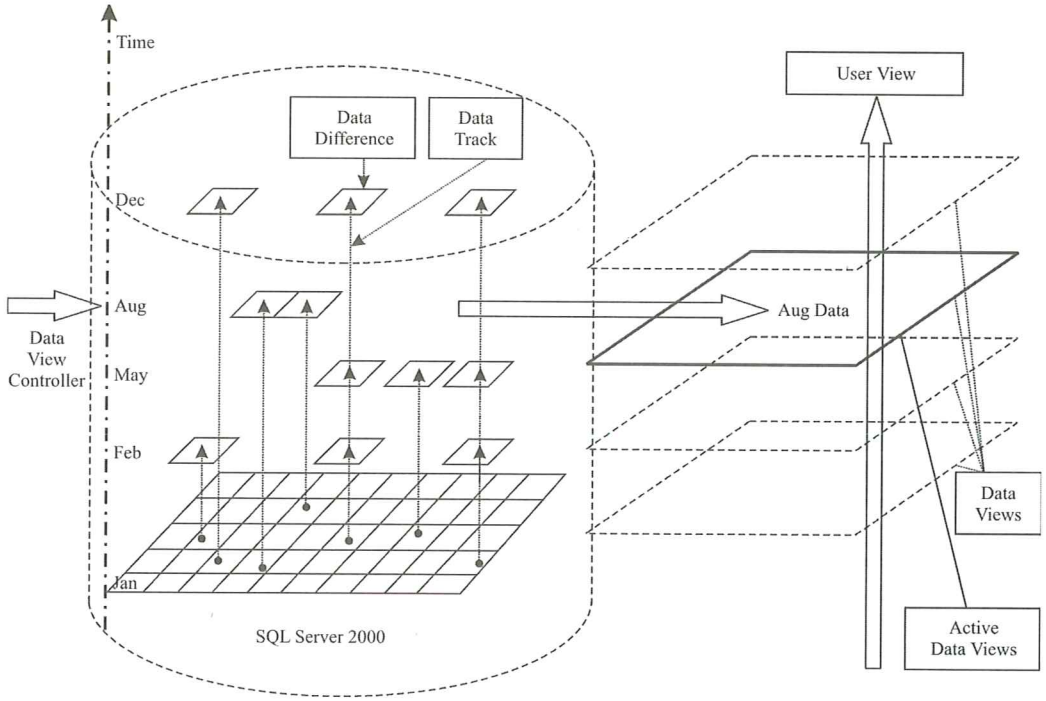

Figure 3. Time scaled spatial data model

During the editing time, data will not be submitted for recording until the changes are finished to avoid too many records and to reduce network traffic. The entire layer will be submitted for updating until all changes are made and the editing process is completed. This procedure is to minimize network traffic to transmit data that have not yet completed in the editing process. When the editing process is completed locally, only new or updated data are transferred to further reduce the network traffic.

As illustrated in Figure 3,

- Not all data records are changed at the same time; mostly only a few records are edited in a large database.
- The vertical time axis is used to illustrate when the data were changed.
- The data history is also recorded in this model by only recording data changes.
- Users can select current data for any given time on the vertical axis.
- By adjusting the time span along the time axis, a data view assigned by the time controller can be selected as the active data view.

We extend the spatial data attribute table to support TSSDM as illustrated in Table 2. The last two fields are the key attributes to support the TSSDM:

Table 2. Data table supporting TSSDM

Field Name	Description
ShapeID	Spatial object id
UserID	Field may link to other attribute table
Name	A quick access to object name
SpatialObject	Binary data of this shape object
Left	Left edge of this shape object
Right	Right edge of this shape object
Top	Top edge of this shape object
Bottom	Bottom edge of this shape object
EditType	What edit was made to this shape object Three values: EditAdd —Newly added shape EditModify —Newly modified EditDeleted —Deleted
TimeScale	The time when an edit was made

This data table supports the three major operations of TSSDM:
Add: A new record is inserted into the table to add an object; current time will be recorded into the TimeScale field.
Modify: When editing an object, such as moving, reshaping or stretching, a new record is also inserted with TimeScale to refer to the current time, and the EditType is set to EditModify.
Delete: When deleting an object, a new record will be put into the same data table if the current time is different from the last change time, and the record will be removed from the table if the current time is the same as the last change time.

Other editing options can be composed from these three general operations. For example, splitting an object can be composed by deleting an existing one and adding new ones.

Except for the data structure for storing time in spatial database, two COM components named ITimerRuler and ITimerScale as illustrated below are utilized.

```
ITimerRuler{
    STDMETHOD(AddTimeScale); /* add a time scale */
    STDMETHOD(get_CurrentTimeScale);/* get current time scale*/
    STDMETHOD(put_CurrentTimeScale);/*set current time scale*/
    STDMETHOD(Insert);   /* insert a timescale into timescale array*/
    STDMETHOD(get_TimeScaleCount); /* count the number of timescale object included in this time ruler object*/
    STDMETHOD(MoveTo); /* move current timescale to some position*/
    STDMETHOD(MoveLast); /* move to the last scale of current time scale object*/
    STDMETHOD(MovePrev); /* move up the scale of current time scale object*/
    STDMETHOD(MoveNext); /* move to next scale of current time scale object*/
    STDMETHOD(MoveFirst); /*jump to the first scale of current time scale object*/
}
```

```
ITimerScale{
    STDMETHOD(get_TimeScaleName);/*get time scale name*/
    STDMETHOD(put_TimeScaleName);/*set time scale name*/
    STDMETHOD(get_BeginDate); /* get the begin date of this time scale object*/
    STDMETHOD(put_BeginDate); /* set the begin date of this time scale object*/
    STDMETHOD(get_EndDate); /* get the end date of this time scale object*/
    STDMETHOD(put_EndDate); /* get the end date of this time scale object*/
    STDMETHOD(get_AvgSpan); /* get the current average scale resolution*/
    STDMETHOD(put_AvgSpan); /* set the current average scale resoltution*/
    STDMETHOD(MoveTo);/* jump to any scale of this scale object*/
}
```

A Time Scale object is a timer for measuring the time span, and a Time Ruler object includes different types of timers which measure the time elapse in different ways. Time scale can be adjusted to fit into different requirements, for example, month for landscape data collection and hours for temperature. The system will utilize these two interfaces to manipulate time control.

To extract spatial objects at a given date, it can be processed with a simple SQL string as follows:

```
strSQL.Format("Select * From %s as t1 where timescale in "
              "(select max(timescale) from %s as t2 where TimeScale <= '%s' "
              "and t1.ShapeID=t2.ShapeID) order by Shapeid",
              TableName, TableName, TIME);
```

where, Shapeid is the index number of a spatial object, TableName is the name of a table storing spatial data; and TIME is the designated date.

C. Communication

Three types of communication are required within the system as illustrated in Figure 4. The left side of the figure illustrates the communication logic inside the Central Computing Facility and between the Central Facility and its users. The right side illustrates the communication between the local servers and the clients. The middle section represents the communication between the Central Computing Facility and local servers.

(i) Communications between a local server and the client
Communications between the local server and client are executed through a web browser embedded with ActiveX and ISAPI. A communication trail is executed in the following process:

- Format a request string inside the ActiveX control with a predefined standard layer name, and proper SQL expressions to filter data (which will be detailed later).
- Send the request to an ISAPI program, which will interpret

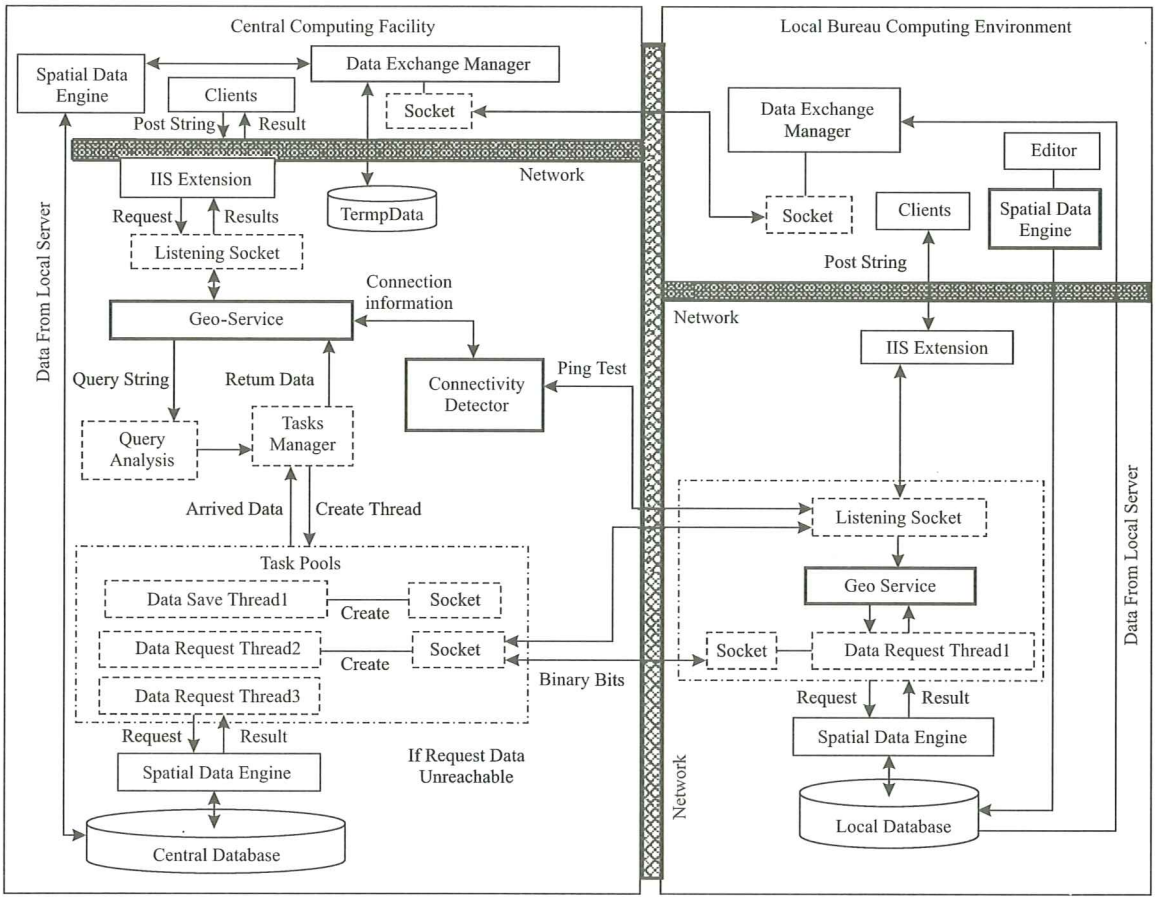

Figure 4. System communication

the request and execute the request with GeoServices and return results in the binary format.

- ActiveX map manager accepts the response in binary format and renders visual products based upon the data received.

The attribute-based query is structured in SQL format, and the spatial query is structured using spatial search with feature ID's. Returned data are formatted in binary that can be interpreted by ActiveX map manager. HTTP Post is used for the communication and the basic request commands are listed below:

R: rID: cmdKeyword {Parameters}
R is a signal to alert the beginning of a post string.
rID is a command id to identify request type.
cmdKeywork has two values: "request" represents spatial query, and "select" represents an attribute query.
{Parameters} is parameters corresponding to rID and cmdKeyword

The following is a list of examples:

r:10:REQUEST layername
r:11:REQUEST layername &left&right&top&bottom
r:12:REQUEST layername objectscount &ID1&ID2...
r:51:REQUEST layername WHERE sqlexpression
r:52:REQUEST layername BY SpatialOperationID &objectcount &objectid &typeid&x1,y1,x2,y2.... DISTANCE distance WHERE sqlexpression
r:13:SELECT fieldname 1 fieldname 2 ... AT startat TOTAL recordscount FROM tablename
SELECT fieldname 1 fieldname 2 ... AT startat TOTALrecordscount FROM tablename WHERE conditionstring
r:14:SELECT fieldname 1 fieldname 2 FROM tablename
SELECT tablename fieldname & fieldname & fieldname 2 &...

For example, to request all the gardens located in Xuhui and Huangpu districts with garden area is greater than 3000m^2, the following string can be formed to process:

R:52:REQUEST : garden BY 0 &2 &0 &2 &x1,y1,x2,y2... &1 &2 &x21,y21, x22,y22... DISTANCE 0 WHERE "area">3000
Where SpatialOperationID =0 means "Included in", and objectcount=2. The two districts' boundaries are formatted in text.

(ii) Communication between ISAPI and GeoServices

When the GeoService in the central computing facility receives a request, a query analysis object will be created to interpret the parameters of the request and generate the relevant binary structure. The structure is submitted to the Task Manager. The Task Manager will create threads to process the request. In the former query example, two threads will be created to process data located in two districts separately. The threads will connect the relevant districts to process the request, or process the data in the central database if a connection cannot be established. The result will be in binary format and sent

back to ISAPI. The following two data structures are used to facilitate this communication:

```
typedef struct CG_RequestStructure
{
    short    nRequestType;     //Request Type
    short    nReturnType;      //the type of returned data,0is
                                 binary,1 is text
    char     pSubmitLayerName[MAX_LENGTH]; //Requested
                                 Layername or Tablename
    BOOL     bPostfix;         //whether followed by data
    long     nSQLPostfixLen; //the length of sql expression:
    long     nObjPostfixLen; //the length of data
    long     nObjectCount;     //requested objects total count;
    long     nBeginAt;         //the start position of records
    short    nSpatialOperateCode; //spatial operation id;
    short    nShapeTypeID;     //type of spatial query object;
    double   dDistance;        //buffer distance
    EXTENT   sExtent;          //the rectangle region of map;
}SUBMIT_DATA;

typedef struct CG_ReturnStructure
{
    long     nTotalLength;     //totoal length of returned data;
    short    nErrorCode;       //error code
    short    nReturnType;      //type of returned data
    long     nShapeCount;      //total object counts returned;
    long     nPointsCount;     //vector objects coordinate
                                 counts;
    long     nPartOffset;      //offset value of parts;
    long     nPartCountOffset;//offset value of partscount;
    long     nPointOffset;     //offset value of points;
    long     nBlockCount;      //image block counts;
    long     nFieldCount;      //fields count returned;
    long     nRecordsCount;//records count returned;
} RETURN_DATA;
```

(iii) Data exchange between the central server and local servers

The data synchronization is performed through the data exchange between the central server and the local servers. The data exchange is performed as follows:

- The central Data Exchange Manager dials up the local Data Exchange Manager with an unstable connection.
- Once the connection is established, the local Data Exchange Manager will read out the requested data directly from the local database without using Geoservices to ensure a reasonable level of efficiency.
- When the central Data Exchange Manager receives the requested data, a collation process will validate the uploaded data with data located at the central database. Any problems discovered during this validation process will be highlighted.
- Once validated, the data will be written to the central database directly by the Data Exchange Manager.

D. Server side spatial data processing(SSDP) implementation

The SSDP is based upon SQL Server 2000, and all the tables described before are managed by SQL Server 2000. The SSDP provides a framework to understand the spatial data structure in the relevant databases, and supports the read/write functions of spatial data by TSSDM. The spatial data engine includes the following eight COM components as illustrated in Figure 5.

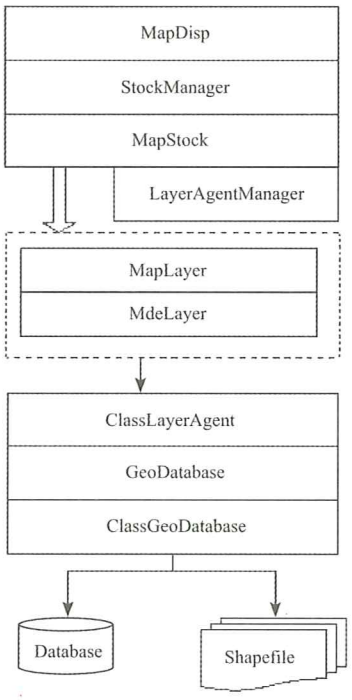

Figure 5. Spatial data engine components and relevant program classes

The eight kernel components or classes are: (1) MapStock: a project manager to record different types of data layers, relevant symbols, order, and style. (2) StockManager: the manager of the MapStock object coordinating storage of different types of spatial data. (3) MapDisp: controlling the display of spatial data stored in the MapStock objects. It also supports map operating functions such as zoom-in, zoom-out, pan and refresh. It also captures mouse movements and keyboard messages, and translates them into proper behaviors. (4) classLayerAgent: a C++ class to access spatial data from a database or a local file such as shapefiles. This agent supports several inherited classes that are used to process features in points, lines, polygons, and other objects in text, image, and cell objects. (5) LayerAgentManger: it accepts the request from the MapStock object to create or delete a classLayerAgent object to load or unload a designated layer. (6) MapLayer and MdeLayer wrap classLayerAgent to facilitate callings of the object from scripts. (7) classGeoDatabase is used to communicate with databases to access binary fields in tables through SQL commands. (8) GeoDatabase wraps the function of classGeoDatabase for classLayerAgent.

V. SYSTEM IMPLEMENTATION AND CLIENT SUPPORT FOR APPLICATIONS

The development of the system required using several

programming languages and the COM, ISAPI, ASP, Windows Service technologies as illustrated below: Communication modules - Windows Services, Visual C++ 6.0; Spatial data management – COM, ATL template library; Data Exchange Manager & client modules - ASP and PHP, and Visual Basic 6.0. The spatial data management COM components were developed to integrate the testing of the landscape data collection system. After testing the system in a local district, the software was deployed at the central computing facility and 19 local landscape bureaus with different types of network connections as illustrated in Figure 6. OA and business management are built on the basis of the landscape central database.

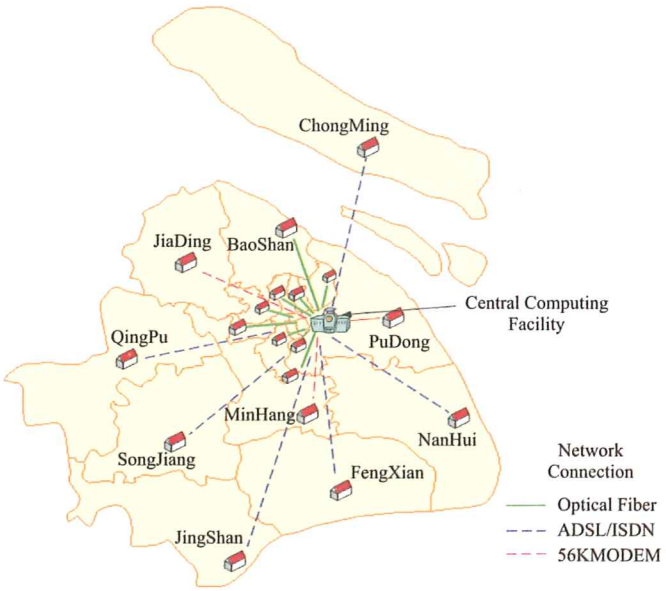

Figure 6. Deployment map of the distributed GIS

The client side is implemented as ActiveX Control within an Internet Explorer. Users will need to download the Control the first time when they access the server. According to different priorities, different functions will be provided to different types of users. For example, users with editing priority will be provided an editing toolbar with the Control. This section illustrates several client-side functions.

A. Management system interface

The initial interface (Figure 7) displays an overview of the Shanghai landscape spatial data, which are collected from the backend of different servers. A click on a district will lead the user to the specific district and relevant data. Although datasets physically reside in different districts, users will not notice the physical dispersion of the datasets and think that the data are all on a local computer.

B. Distributed query and statistic

Figure 8 illustrates a statistical summary of the landscape data. The list box on the top right with the date information allows

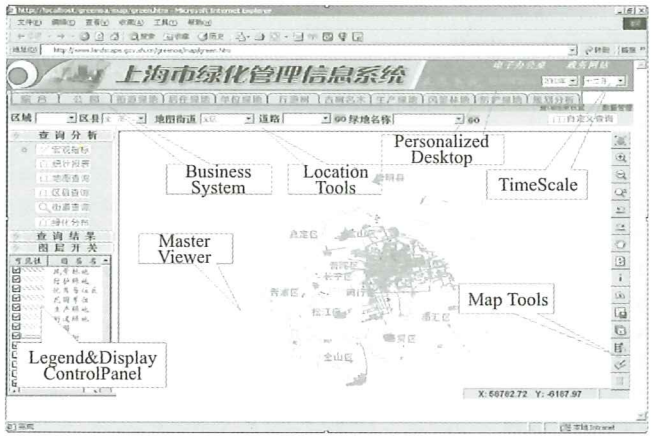

Figure 7. Landscape business management main interface

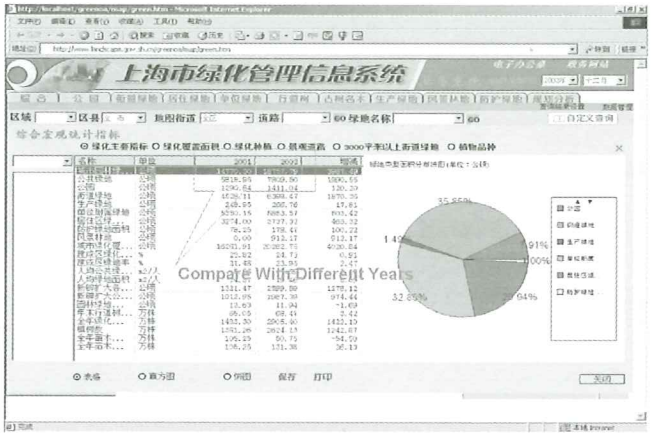

Figure 8. Landscape marco-statistic index

Figure 9. Query detailed park information by identify tool

users to change time and/or date and view the data based on the information stored in TSSDM. Spatial data and attribute data are linked through the ID field, which facilitates cross-access between spatial data and attribute data. Figure 9 shows the attributes of a park when spatial information was used in the query process.

C. Spatial analysis

One of the primary functions of GIS is to support and perform spatial analysis. The Distributed GIS-based landscape resource management system also supports various types of spatial analytical procedures, including distance-based spatial query and selection, and different types of buffering functions. For instance, if one wants to find out the total park area or green space in the vicinity of a neighborhood, the street segment representing that neighborhood should be selected first (Figure 10). Then, the user has to provide a distance to define the "neighborhood" or "vicinity". Given the distance and the selected street segment, a buffer will be created.

Figure 10. Buffer creation

Landscape data within this buffer will be extracted from one database or multiple databases, but in a seamless manner. Outputs in the form of statistical tables and charts in easily understandable formats will be generated as in Figure 11.

Figure 11. Landscaping statistical analysis

D. Image-based spatial data editing

Aerial photos and satellite images are important sources of landscape data. Landscape data can be derived from these sources through heads-up digitizing. To support this task, images are managed in the GeoService module as image services and are delivered using the pyramid method with image library/database to improve performance. All filled polygons are transparent for editing and can be updated to local and central databases through TSSDM (Figure 12).

Figure 12. Image as background to collect data

VI. CONCLUSION

This paper reviews in detail the four computing challenges to design and develop a Distributed GIS supporting the landscape resource management of Shanghai, China. The system has proven to be well designed and robust since its initial operation. The solutions provided through the system also meet management requirements.

A pyramid-based image library was successfully built and has been supporting the landscape spatial-data collection and update maintenance. TSSDM is used to handle the data update issues related to unstable network connections and to reduce the demand of data communication for data updates and edits. Transparency of distributed system access and issues on data integrity were achieved through our architecture, the data model, the communication approach, and the GeoServices we have adopted.

The experience we learned here can be used in many other relevant areas as mentioned in section I. However, system performance can be further improved by designing more efficient database structure, communication protocols, and parallel algorithms (Yang et al., 2005). The size of image library has to be reduced to a reasonable size and system approach should support standards in order to build open solutions (Yang and Tao, 2005). We envision that these remaining research issues will be addressed in future research and development efforts.

ACKNOWLEDGEMENTS

We thank Mr. Henry Wolf of CEOSR for his editorial comments.

REFERENCES

[1] Barthello M., Pollack J., 2001, WebGIS for road/rail conditions and rapid routing, 14th Annual Geographic Information Science Conference (Baltimore, MD: AAG).

[2] Bogner D., Daberning M., Koren G., 2001, Assessment of agricultural land use in urban regions as a decision support system using WebGIS, 4th AGILE conference (Brno, Czech Republic: AGILE).

[3] Deng W., 2000, WebGIS-A new way for public participation in city planning, GIS2000 (Toronto, Ontario, March 13–16, 2000).

[4] Dragicenvic S., 2000, Environmental data exploration: The web GIS approach, GIS2000 (Toronto, Ontario, March 13–16, 2000).

[5] Eichelberger P., 2001, Chester County, Pennsylvania WebGIS: An improved data approach, 14th Annual Geographic Information Science Conference (Baltimore:AAG).

[6] Galinao B., Brennan C., 2002, Power to the people! A web-based GIS provides a public-involvement tool for airport development, GeoWorld, 15(6): 32–35.

[7] Gong J., et al., 1998, Cross platform distributed geographic information organization and process (in Chinese). Journal of Wuhan technical university of surveying and mapping, 23(4): 364–369.

[8] Guangyou, Lin, 2002, Event based object oriented time spatial data model, Survey Transcation (in Chinese), 4, pp.71–76.

[9] Kelly M. 2001. WebGIS for watersheds: Developing applications and resources for watershed planning and conservation, Coastal Geotools 2001 (Charleston, SC: NOAA).

[10] Kenneth E. F., Kirvan A. P., 1997, WebGIS, NCGIA Core Curriculum in GIScience, http://www.ncgia.ucsb.edu/giscc/units/u133/u133.html.

[11] Yang C., 2000, Theory, Techniques and Implementation Methods of WebGIS (in Chinese), Unpublished Ph.D dissertation (Beijing: Peking University).

[12] Plewe B., 1997, GIS Online: Information retrieval, Mapping, and the Internet (Santa Fe: OnWord Press), pp. 6–14.

[13] Peng Z. R., Tsou M. H., 2003, Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks (Hoboken, New Jersey: John Wiley & Sons, Inc.), pp.2–5.

[14] Peng Z. R., 1999, An assessment framework for the development of Internet GIS, Environment and Planning B-Planning & Design, 26:117–132.

[15] Slater S., 2002, Queensland a princely GIS. Geoworld, 15(5): 42–45.

[16] Shen G., 2001, WebGIS tools for online spatial data exploration and analysis in business. WebNet Journal (United States), 2(3): 16–53.

[17] Takatsuka M., Gahegan M. N., 2002, Exploratory geospatial analysis using GeoVISTA Studio: From a Desktop to the Web, In Proceedings of the Second International Conference on Web Information Systems Engineering (Kyoto,Japan: IEEE), pp.92–101.

[18] Wu L., 2002, Integration technology of distributed spatial database. Geographic and Territorial Research, 18(1):6–10.

[19] Yang C., Kafatos M., Yang R., Wong D., Cao Y., 2005, GridGIS: A next generation GIS. *Proceeding of International Conference on Cybernetics and Information Technologies, Systems and Applications: CITSA 2004*, Jul. 21-25, Orlando, FL. 3: 16–21.

[20] Yang C., Li Q., Cheng J., 2000, Research and Implementation on Web Publishing of Remotely Sense Image, *Chinese Journal of Remote Sensing*, 4(1):71–75. (in Chinese).

[21]Yang C., Tao V., 2005., Chapter 5: Distributed Geospatial Information Services. Eds by S, Rana and Sharma. New Frontier of Geospatial Information Technology, Springer, in press.

[22] Yang C., Wong D., Yang R., Kafatos M., Li Q., 2005, Performance Improving Techniques for Web-based GIS. *International Journal of GIS*, 19(3): 319–341.

[23] Yang R., Yang C., Kafatos M., 2002, A distributed framework and its application in supporting automatic cooperative work in NASA environmental research, Nov. 10–14, Denver, CO, USA, *Proceedings of Integrated Remote Sensing at the Global, Regional and Local Scale*: 293–297.

[24] Zhi Cai, 2002, Compacted time spatial model based on kevin-Jan3D model (in Chinese), Survey Transaction, 1, pp.77–81.