International Journal of Geographical Information Science
Vol. 19, No. 3, March 2005, 319–342

Research Article

Performance-improving techniques in web-based GIS[#]

CHAOWEI (PHIL) YANG*†‡, DAVID W. WONG†‡, RUIXIN YANG†‡,
MENAS KAFATOS†‡ and QI LI§

†Earth Systems and GeoInformation Sciences, School of Computational Sciences, George Mason University, Fairfax, VA 22030, USA
‡Center for Earth Observing and Space Research, George Mason University, Fairfax, VA 22030, USA
§Institute of RS & GIS, Peking University, Beijing, 100871, PR China

(*Received 3 January 2003; in final form 12 November 2004*)

WebGIS (also known as web-based GIS and Internet GIS) denotes a type of Geographic Information System (GIS), whose client is implemented in a Web browser. WebGISs have been developed and used extensively in real-world applications. However, when such a complex web-based system involves the dissemination of large volumes of data and/or massive user interactions, its performance can become an issue. In this paper, we first identify several major potential performance problems with WebGIS. Then, we discuss several possible techniques to improve the performance. These techniques include the use of pyramids and hash indices on the server side to handle large images. To resolve server-side conflicts originating from concurrent massive access and user interactions, we suggest clustering and multithreading techniques. Multi-threading is also used to break down the long sequential, layer-based data access to concurrent data access on the client side. Caching is suggested as a means to enhance concurrent data access for the same datasets on both the server and the client sides. The technique of client-side dynamic data requests is used to improve data transmission. Compressed binary representation is implemented on both sides to reduce transmission volume. We also compare the performance of a prototype WebGIS with and without these techniques.

Keywords: WebGIS; performance; pyramid; hash index; multithread; cluster; cache; dynamic data request

1. Introduction

The fast-paced development of GIS has triggered some researchers (Sui and Goodchild 2001) to reconsider the fundamental essence of GIS and its social implications. GIS has been widely used in various types of business, government and university projects. For instance, in North American alone, the value of the GIS market will increase from \$1.4 billion in 2001 to \$2.0 billion in 2004 even in a slow economy (Rogul 2003). Rogul also points out that one of the reasons for this continued expansion is that GIS is finding new markets on the Internet. Today, GIS serves as a means of communication, conveying information and knowledge to the

*Corresponding author. Email: cyang3@gmu.edu
[#]Methods described and relevant implementation are patent – pending.

public. Sui and Goodchild (2001) suggest using the term 'media' to describe GIS. This booming trend of GIS is potentially attributable to many, but at least two primary factors: the building of the Spatial Data Infrastructure (SDI) worldwide, and the dazzling development of computing technology and information technology in general.

During the past decade or so, the construction of SDI has proliferated geographically across all levels, from the National SDI in the US (www.fgdc.gov/) to the Global SDI (www.gsdi.org/), and the local SDI, including the states (e.g. http://www.vgin.state.va.us/, for the Virginia Geographic Information Network) and counties (e.g. www.loudoun.gov/omagi/index.htm, for the Loudoun County in Virginia). In the US, the building of the NSDI has involved almost every department of the federal government (www.whitehouse.gov/omb/circulars/a016/a016_rev.html). Large volumes of geographic data, which are valuable to various organizations, have been accumulated mostly in a traditional hierarchical manner: objects and their related attributes are collected and classified according to different themes or layers, and different layers are overlain to produce a specific map, as illustrated in figure 1a. In order to fully utilize the available spatial data efficiently and effectively, GIS has to play a critical role, not just in disseminating the raw data, but also in providing information and offering value-added services to potential users.

However, in order to utilize and access the valuable data, GIS-enabled environments have to be available to the public. The fast development of the Internet, especially the World Wide Web (WWW) and wireless communication, provides an ideal platform to empower the general public with the GIS technology through WebGIS (Plewe 1997, Peng and Tsou 2003) and Location Based Service

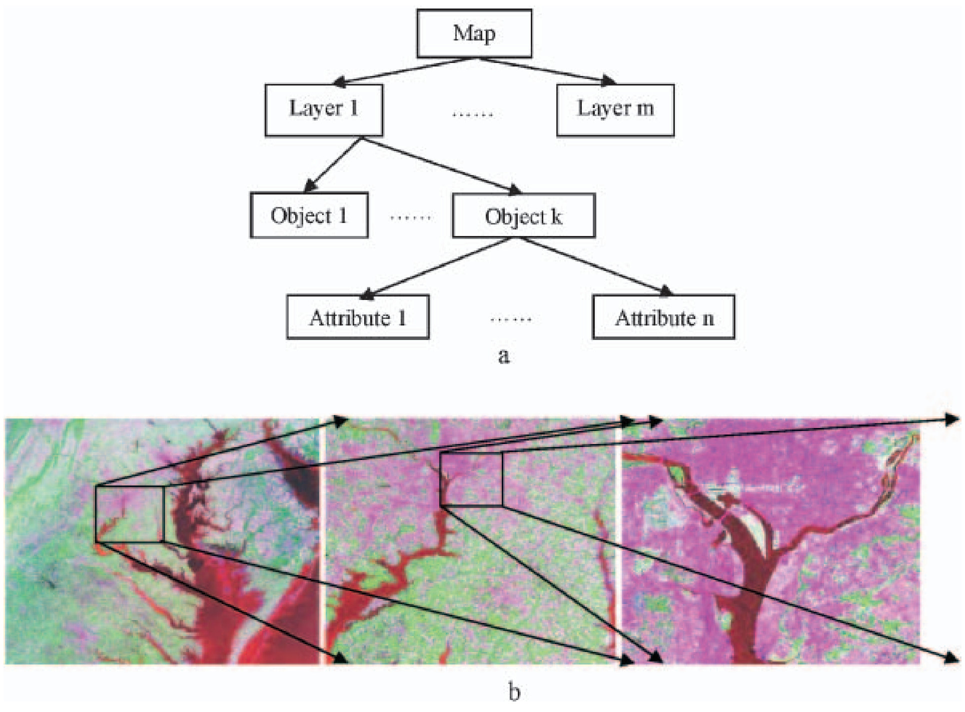

Figure 1. Hierarchical organizations of data. (a) Hierarchical organization of geographic data. (b) A pyramid structure of image data.

(LBS) (http://www.openls.org/). The geospatial enablement of our everyday tools, e.g. cars and phones, has provided the general public channels to access GIS environments almost anywhere and anytime (Cowen 1994, Koeppel 2001). These developments, facilitated by the general advancement of computing technology, have equipped us to share data and information through SDI and different types of distributed systems. In all diverse systems, the common vital units are the computing components, which handle the access, processing, and visualization of the geographic information, and the interactions between users and data. Therefore, a distributed GIS can be abstractly organized into different computing units, which are themselves connected through various types of networks (coaxial cable, optical fiber, satellite), and represented by the client/server computing model suggested by many scholars (Kenneth and Kirvan 1997, Peng 1999, Yang 2000). According to different roles, a typical system broadly consists of three parts: the client, server and network. The client interacts with the users and performs certain computing functions on spatial data. The server supplies data and information, and performs some value-added services to a client. The network hosts the transmission of information between the client and server.

Among various types of network GIS, including the LBS, WebGIS is the most extensively developed and widely used. It has accompanied the rapid development of the WWW during the past decade. Many Internet users have experience using WebGIS. Mapquest (http://www.mapquest.com/), Terraserver (http://www.terraserver.com/), weather forecast (http://www.weather.com/), and many other WebGISs 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, Slater 2002), business analysis (Shen 2001), airport construction (Galinao and Brennan 2002), and data integration and dissemination (Eichelberger 2001, Takatsuka and Gahegan 2002), just to name a few applications. To give the user a better view of the data or information, 3D visualization (Coors and Flick 1998), Virtual Reality Markup Language (VRML) (Huang and Lin 2001), and multimedia (Grønbæk 2002) have also been integrated into certain WebGIS.

Some scholars suggest that WebGIS should include spatial analysis capability, not just web mapping and data delivery. Following this argument, for example, ArcIMS may be regarded as a WebGIS, while TerraServer may not. However, the simple web mapping function contributes to the popularization of WebGIS. All these systems require similar techniques to improve performance. Therefore, we also refer to MapQuest, TerraServer and other web-mapping systems and data portals (e.g., www.geodata.gov) as WebGISs in this paper.

While WebGIS is gaining in popularity, dissemination of voluminous and heterogeneous data becomes a challenge, as the Internet bandwidth is not limitless. To handle this challenge, two important issues should be considered: (1) share and interoperate the heterogeneous data among different systems, different communities, and different users (Buehler and McKee 1998); and (2) improve the system performance so that data are delivered to the users within a reasonable time span. The OpenGIS Consortium (OGC) (http://www.opengis.org/) and the Technical Committee 211 of the International Organization of Standards (http://www.iso.org/iso/en/stdsdevelopment/tc/tclist/TechnicalCommitteeDetailPage.TechnicalCommitteeDetail?COMMID=4637) were established to address the first issue by providing a series of standardized interface specifications to allow different components of the Performance-improving techniques in web-based GSI system, including data, to support interoperability. Although the second issue has been addressed by various suggestions, research on the performance of WebGIS still has two major limitations: (1) most methods focus on only one aspect of the performance problem; and (2) most methods do not consider how the hierarchical structure of map, layer, object, and attribute may affect performance. In this paper, we propose a set of solutions to provide an improved treatment on WebGIS performance.

This paper is organized into several sections. In section 2, we review related research on improving the performance of WebGIS, and discuss different methods to improve the performance of individual components, as well as the overall system performance. In section 3, we discuss the use of pyramid and hash indices for handling large image data management and transmission problems. In section 4, we outline multithread and cluster techniques, which are used to deal with the problem of concurrent access and to improve client-side interactive performance. In section 5, we propose using the methods of caching and dynamic transmission to improve the interactive performance. In section 6, we suggest using binary data compression to improve the transmission performance by reducing data transmission volume. Finally, in section 7, we compare the performance of a prototype WebGIS with and without the methods proposed in this paper.

2. Related works

WebGIS focuses on how to allocate spatial data, both raster and vector (Goodchild 1992), in a client–server-based web platform, as well as on how to allocate functions to different system components in processing these data to satisfy users' needs (Yang 2000). Our review includes three parts: raster data transmission, vector data transmission, and other performance-related computing techniques used in the client–server model.

2.1 *Raster data*

To handle raster data transmission, useful solutions can be borrowed from research and applications in image transmission in computer science. For instance, the progressive raster transmission method has been frequently suggested (Rauschenbach and Schumann 1999, Srinivas *et al.* 1999). The basic idea of progressive transmission is to use image compression techniques to gradually extract and transmit raster data. After the compressed image is transmitted to the client, the image is gradually reconstructed on the client side. The simplest progressive raster transmission method is to randomly extract and transmit the image without following a systematic algorithmic process. More sophisticated methods for progressive raster transmission could be based on image-compression techniques (Rauschenbach and Schumann 1999, Srinivas *et al.* 1999), such as the widely used Joint Photographic Experts Group (JPEG) method (Kern and Carswell 1994), various wavelet methods (Morlet and Grossman 1984, Wu *et al.* 2002), fractal methods (Barclay,1989), or a combination of the above methods (Zhao and Yuan 1996, Davis 1998). Because of their complexity and their computing requirements, the progressive methods do not have the flexibility to handle the transmission of large volume and variable size images efficiently in the WebGIS environment, but are ideal for transmitting fixed-size images on the Internet. However, the fundamental methods of image compression can still be used to reduce the overall image transmission size.

A relatively large image can first be extracted into different levels of detail to construct a hierarchical or pyramid structure. In each hierarchical layer, the image is further cut into pieces or tiles, which are logically connected through their respective coordinates. Then, the image data are transmitted in a load-on-demand manner, i.e. only the data of interest, as requested by the user, are combined from the respective tiles and transmitted to the client side (Coors and Flick 1998, Barclay *et al.* 1999, Wei *et al.* 1999, Chen *et al.* 2000, Yang *et al.* 2000, Tu *et al.* 2001). TerraServer uses this method to assist the SQL Server to manage images (Barclay *et al.* 1999), but this approach is not widely applicable because most WebGISs are not specifically designed with SQL Server. ArcGIS also adopt the pyramid technique in handling big images, but the pyramid is built on the fly every time a large image is accessed. This approach of constructing an instant pyramid is not suitable for managing images on WebGIS because the response time will be too long if the pyramid is constructed for every access. Therefore, a strategy for managing a permanent pyramid is needed. In this paper, we further develop this method and propose a hash index method to manage a pyramid to improve performance.

Some scholars suggest adopting tile pre-fetching and caching techniques to improve raster data transmission (Chen *et al.* 2000, Kang *et al.* 2001, Tu *et al.* 2001). To cache is to temporarily keep loaded data for future re-uses, while pre-fetching is to obtain and cache data in advance, when the data are expected to be used later. Cached data are used at least once, but pre-fetched data may never be used. Unfortunately, this combined method is effective for raster data only, and the complex nature of WebGIS involves the handling of both raster and vector data, as well as the support of spatial analysis, in the ideal situation. Moreover, users will request raster images in a relatively random manner and therefore, the pre-fetching technique will not be efficient and effective in handling random requests. Therefore, caching, rather than pre-fetching or both, will be one of the performance-improving techniques discussed in this paper.

2.2 *Vector data*

The progressive transmission technique could also be used for vector data, but the process is different from that applied to raster data in the hierarchical pyramid structure. Vector data are extracted using cartographic principles to construct the multi-level or multi-layer structure, instead of using the simple resampling process for raster data. Hoppe (1996) proposed a mesh scheme that marked a milestone for vector progressive transmission. By slightly modifying the topology of the input mesh, Baja *et al.* (1999) achieved a higher compression ratio for transmitting Triangulated Irregular Network (TIN) data. Floriani *et al.* (1998) proposed an encoding structure to improve the efficiency of progressive transmission. Bertolotto and Egenhofer (2001) proposed a model to generate multiple map representations and a set of generalization operators. Buttenfield (2002) introduced an algorithm that transmits vector coordinates at progressive levels of resolution, preserving geometric and topologic properties. These processes perform atomic topological changes on the vector map to achieve a better transmission performance, preserve the topology of the spatial data, and are best for transmitting single-layer maps. However, they do not take into account the de facto hierarchical geographic data organization of map, layer, object, or attribute. Therefore, they cannot be used generically to handle the heterogeneous data sets in a WebGIS environment.

Another technique for improving the performance of vector data transmission is indexing. Indexing techniques have been widely studied mainly from two perspectives: spatial object, attribute-based thematic indexing and spatial indexing (Worboys 1995). Thematic indexing is the process of indexing attributes such as address, postcode, phone number, feature name and other attributes, such that attribute data can be efficiently processed using popular commercial databases (McCurley 2001). The R-tree index method has also been used for thematic indexing (Chen *et al*. 2000). Spatial indexing is more complex than thematic indexing, and can be classified into two general categories: hierarchical access indexes, such as R-tree (Guttman 1984) and Quad-tree (Samet 1984), and hash indexes, such as Grid-files and R-files (Kwon *et al*. 2002). R-tree utilizes the concept of a feature's Minimum Bounding Box (Guttman 1984), the minimum rectangle which contains the feature. There are several extensions of R-tree, such as R*-tree, suggested by Beckmann *et al*. (1990) to deal with points and rectangles, and R+-tree, proposed by Sellis *et al*. (1984) to allow dynamic indexing. To support the multi-level data structure, Reactive-tree (Oosterom 1991), PR-file (Becker *et al*. 1991) and Multi-scale Hilbert R-tree (Edward and Kevin 2002) have been proposed.

These spatial and thematic index research efforts provide a basis for implementing vector data indexing. The indexing methods mentioned above could be adopted on the client and/or server side to improve the access to vector data. Thus, the primary focus of this paper is on dynamically allocating vector data between the server and client, and related computing methods.

2.3 *Other computing techniques*

Kang *et al*. (2001) and Tu *et al*. (2001) have used pre-fetching and caching techniques for raster data transmission in systems exclusively handling raster data. In a WebGIS, which involves both raster and vector data, caching could be used not only for transmitting data, but also for allocating data between client and server, especially for metadata and fundamental layer information. Multithreading techniques have also been suggested to improve the performance by processing more than one task simultaneously (Chen *et al*. 2000). This technique, in fact, is very useful for handling server-side concurrent access, as well as to improve client-side interactive capabilities. When a server is opened to the public, many users may access the server simultaneously. In this potentially massive access situation, the system may also be required to adopt cluster techniques for multiple servers to serve the users to ensure reliability and to improve the overall performance (Yang 2000).

2.4 *A system solution*

Based upon the client–server environment in WebGIS, we propose a set of techniques that can be used together to improve the system-wide performance of a WebGIS. The techniques of pyramid and hash indexing are used for managing large image datasets on the server side, while well-developed R-tree indexing techniques, details of which are beyond the scope of this paper, could be used for indexing vector data. Multithread and dynamic request methods are used for handling users' concurrent access, as well as client-side concurrent data requests. A caching technique is used for keeping some information on the client side based on the submitted dynamic requests. Binary compression technique is used to reduce the transmission volume of data. The following sections provide a detailed description of each of these techniques, except the R-tree indexing methods, and demonstrate how they can enhance the performance of WebGIS.

3. Pyramid technique and hash index

On the client side of a WebGIS, hardware specifications often restrict the size of an image viewed by users. For example, modern desktops and laptops use 1024×768 or 1280×1024 resolution with 15–21 inches of display in general, and MapQuest even restricts the size to 500×500 and 356×250 for a better performance (http://www.mapquest.com). It is almost always the case that the entire image is much larger than these restricted display sizes. For instance, if the data set is a 1 m resolution image of a county with an area of approximately $50\,\mathrm{km} \times 50\,\mathrm{km}$, the image size should be $50\,000 \times 50\,000$, which cannot be displayed by an ordinary monitor. In order to provide faster access to different parts of the entire image from the restricted window, we propose using both the pyramid and cut-and-hash indexing technique.

A pyramid, also called a hierarchy (Tu *et al.* 2001), is constructed by generating different abstraction levels of the original data through resampling (figure 1b). For example, we can generate an image of $10\,000 \times 10\,000$ by resampling the original $50\,000 \times 50\,000$ image based on a 5×5 resampling scheme. By further resampling the reduced image based on a 5×5 reduction, we can obtain an image with only 2000×2000 pixels. It is much faster to display and transmit this reduced 2000×2000 image for an overview of the original image than to process the original $50\,000 \times 50\,000$ pixels.

A pyramid consists of multi-scale replicated images of the original one. When a client requests data at a given scale, the server will search the required data from the level which has a scale closest to that requested, instead of searching the original high-resolution data directly. For example, in figure 1b, suppose the three images in the pyramid are in the scale of 1:100 000, 1:20 000, and 1:4000, respectively. When the client requests an image of 1: 25 000 in scale, the server generates the requested image from the 1:20 000 image instead of from the original 1:4000 image, which is 25 times larger. The use of the pyramid technique in this manner can reduce access time by extracting data from a smaller cartographic scale image already stored in the pyramid instead of the original one, especially when there is a large difference between the requested scale level and the original scale. When more detailed data are needed, or when it becomes necessary to access the original image, a better access speed can be achieved by accessing the smaller piece of the original data, if the original data are cut into smaller pieces (Yang *et al.* 2000). In this circumstance, a restricted area of the image, instead of the entire image, is accessed. If the smaller-piece images are managed and accessed efficiently with effective strategies, the performance of the system will be improved tremendously.

Implementation strategies for image cutting and pyramid construction have to deal with three issues: determining the size of the cut pieces, finding the related pieces efficiently, and identifying a proper number of levels that should be used for constructing the pyramid layers. To derive solutions to these issues, the procedure for accessing a pyramid is critical. Given that a pyramid is built for an image, four steps are involved in accessing the data. The first step is to determine the target level at which the data will be extracted by comparing the requested scale with different scales already stored in the pyramid. The second step is to find related pieces at the target level by using coordinate information. The third step is to combine these related pieces to form an image, which may have a spatial extent larger than the requested area. The final step is to cut the combined image to match the requested area and, if necessary, resample the cut image to the requested scale level. These steps are illustrated in figure 2.

Suppose the requested data have dimensions $H \times W$ with scale S, and bounded by (X_1, Y_1) and (X_2, Y_2), as illustrated in figure 3; a pyramid is constructed with a uniform distribution of scale levels (e.g. 1:1 000 000, 1:100 000, 1:10 000, 1:1000, etc.). Each piece of the image has a dimension of $H' \times W'$. The original image has a scale S_0 and is bounded by the coordinates (X, Y), (X', Y'). Suppose the pyramid has n levels, and let S_i denote the scale at the ith level. Images at different scale levels

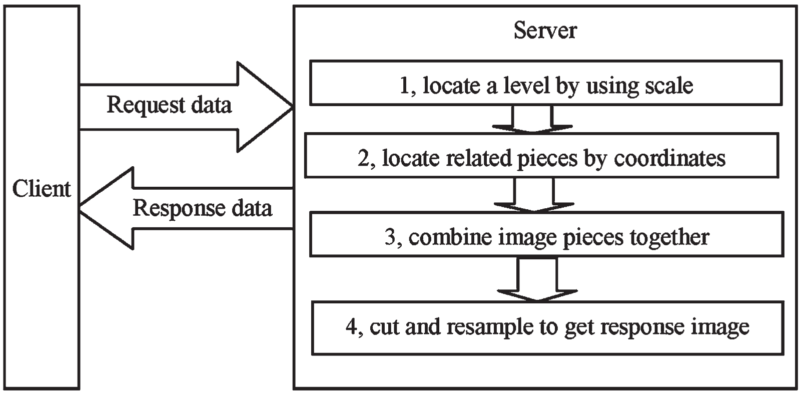
Server
1, locate a level by using scale
2, locate related pieces by coordinates
3, combine image pieces together
4, cut and resample to get response image
Request data
Response data
Client

Figure 2. Procedure for data access using the pyramid and cut method.

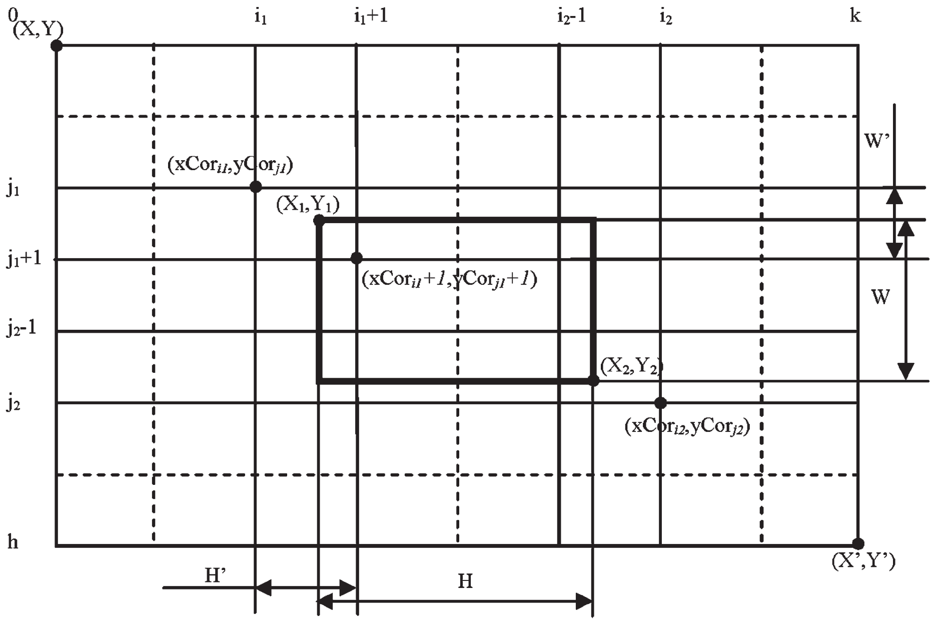
0
(X,Y)
i1
i1+1
i2-1
i2
k
j1
j1+1
j2-1
j2
h
(xCor$_{i1}$,yCor$_{j1}$)
(X1,Y1)
(xCor$_{i1}$+1,yCor$_{j1}$+1)
(X2,Y2)
(xCor$_{i2}$,yCor$_{j2}$)
(X',Y')
W'
W
H'
H

Figure 3. Locating pieces from a pyramid level.

in the pyramid could be arranged in a descending or ascending order. The first step in the data access procedure is to find the appropriate level m with scale S. This can be accomplished by a bisectional search procedure based on the descending or ascending order. It is possible that no layer in the pyramid has the scale of exactly S, and therefore the goal is to identify the two layers such that their scales are in the relation of $S_m < S \leqslant S_{m+1}$. In this case, the larger scale image will be used to provide a better display of the data.

In the second step, related pieces will be located by their coordinate information. The coordinates of all pieces need to be compared at level m in the pyramid with coordinates of the requested data. When the number of image pieces is not very large, the coordinates of all pieces can be evaluated, and thus the related pieces can be identified. When the number of pieces at each level becomes very large (and unfortunately, this is often the case), searching through all pieces with every request from the clients will be too time-consuming and unmanageable. A Hash index method can be used to handle this problem.

Assume that the uniformly cut data pieces have the same dimension $H' \times W'$ and different coordinates ($x\mathrm{Cor}_{i1}$, $y\mathrm{Cor}_{j1}$) and ($x\mathrm{Cor}_i + 1$, $y\mathrm{Cor}_j + 1$), where i and j are integers referring to the horizontal and vertical sequence of pieces, and the maximum values of i and j in a certain level are k and h, respectively. Therefore, there are $k \times h$ pieces at a level, as illustrated in figure 3. Each of these pieces is assigned the name *i–j*.ext, where i and j refer to the location of the piece at the given level, and ext could be jpg, tiff, and bmp, etc. Therefore, the location of the piece and its coordinates are coded into the name of the data file. The coordinates of each piece could be computed according to equation 1. Given the requested data coordinates (X_1, Y_1) and (X_2, Y_2), we can compute (i_1, j_1) and (i_2, j_2) according to equation (3), which itself is derived from equation (2). Equation (2) is the formal expression for identifying a spatial overlap condition. After deriving (i_1, j_1) and (i_2, j_2) from equation (3), the results are used to locate data pieces, which intersect with the requested area. These data pieces or files are selected for future processing.

$$\begin{cases} x\mathrm{Cor}_{i1} = X + (X' - X)/k \times i \\ y\mathrm{Cor}_{j1} = Y + (Y' - Y)/h \times j \\ x\mathrm{Cor}_{i1+1} = X + (X' - X)/k \times (i+1) \\ y\mathrm{Cor}_{j1+1} = Y + (Y' - Y)/h \times (j+1) \\ i = 0, 1, 2, \ldots k-1 \\ j = 0, 1, 2, \ldots h-1 \end{cases} \tag{1}$$

$$\begin{cases} x\mathrm{Cor}_{i2+1} = X + (X' - X)/k \times (i_2+1) > X_2 > x\mathrm{Cor}_{i2} = X + (X' - X)/k \times i_2 \\ y\mathrm{Cor}_{j2+1} = Y + (Y' - Y)/h \times (j_2+1) > Y_2 > y\mathrm{Cor}_{j2} = Y + (Y' - Y)/h \times j_2 \\ x\mathrm{Cor}_{i1} = X + (X' - X)/k \times i_1 < X_1 < x\mathrm{Cor}_{i1+1} = X + (X' - X)/k \times (i_1+1) \\ y\mathrm{Cor}_{j1} = Y + (Y' - Y)/h \times j_1 < Y_1 < y\mathrm{Cor}_{j1+1} = Y + (Y' - Y)/h \times (j_1+1) \end{cases} \tag{2}$$

$$\begin{cases} i_1 = \lfloor (X_1 - X) \times k/(X' - X) \rfloor \\ j_1 = \lfloor (Y_1 - Y) \times h/(Y' - Y) \rfloor \\ i_2 = \lceil (X_2 - X) \times k/(X' - X) \rceil \\ j_2 = \lceil (Y_2 - Y) \times h/(Y' - Y) \rceil \end{cases} \tag{3}$$

Instead of searching through the entire level and comparing each coordinate of the cut pieces, we could find the required pieces directly from i_1, j_1, i_2, j_2, and the associated file names with equation (3), which serves as the Hash function for locating the appropriate pieces. After all required pieces are identified, they can be combined and then cut according to the requested area. Resampling may be required too if the combined image does not have the same scale as the required data. The process of combining and cutting images can be performed efficiently with existing methods provided by many image-processing modules.

In the entire process, the most time-consuming components are to identify the appropriate scale level and to find related pieces. The efficiency of these two processes can be improved tremendously by using bisection search and hash indexing as described above. The response time then becomes a function of the number of pieces required and the scale difference between the combined piece and the required image. The ideal situation is that we can store the data at any level the client requests. However, we do not have infinite storage space to maintain data from all possible scale levels. Therefore, to achieve a better performance as well as a balance on storage capability and data volume, we suggest that the cut pieces should have a dimension one to two times the size of the image or data display on the client side (Yang *et al.* 2000). In terms of the number of layers to be constructed in the pyramid, we have experimented with different scale ratios and found that using a scale ratio of 1:9 to extract layers to form the pyramid can achieve a reasonable performance. More formal analysis is required in the future to determine the most optimal scale ratio to construct the layers in the pyramid for raster data.

4. Cluster and multithread

Centralized or desktop GIS software includes many complex functions. To implement and integrate these GIS functions in a distributed environment, Component Technologies, such as Common Object Request Broker Architecture (CORBA) and Component Object Model (COM), can be used (Yang 2000). A WebGIS should provide an efficient environment for clients and servers to communicate intensively such that services requested by the users can be completed within a reasonable time. The communication between servers and clients, and related GIS function components in each side of the network are illustrated in figure 4.

Traditionally, the procedure for processing a user's request involving multiple data layers, raster and/or vector, takes the following steps: (1) When the client

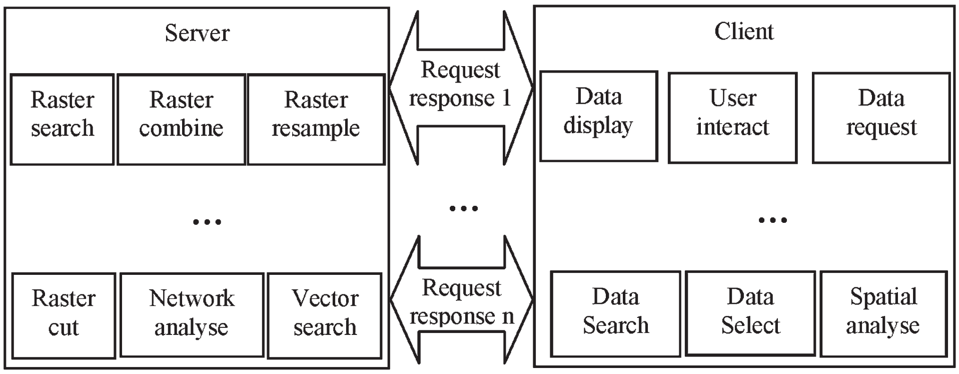

Figure 4. Processing functions in WebGIS.

receives a request from the user, the client performs a spatial search on the client side to identify which spatial datasets are required. (2) Within the selected data extent, a thorough search by data layer is performed to check whether the client side has the requested data layer cached. If the client side does have the data layer, then there is no need to fetch the data from the server. (3) If the client side does not have the requested data layer, the client will then submit a data request to the server, asking for the requested data. (4) The server will conduct a spatial search to locate the data. (5) After identifying the data layer, the server sends the data back to the client. 6) The system returns to the second step until all layers are found and transmitted. If a traditional single-thread process is adopted, a user cannot initiate a subsequent layer-data request before the former-layer data are received. If the request involves many searches and data layers, the process will become very time-consuming and may not be completed within a reasonable response time. This in turn will be a great frustration to users. Furthermore, when many clients access the same server simultaneously, the sequential process described above will be too slow to respond adequately to requests from multiple clients. The increasing number of clients may rapidly reach the bottleneck of the system with any given system architecture.

These problems related to responding to requests from multiple clients can be handled more efficiently by using cluster and multithread techniques (Solomon and Rankins 1997). A major advantage of using the multithread technique is illustrated in figure 5a. When a single thread server is used to process m processes, the time required to finish the entire process is

$$\sum_{i=1}^{m} t_i$$

where t_i is the time required for process i. If multithreading is used, these m processes can be concurrently performed in m threads, and the time required to finish all m processes will be $t=\max\ \{t_1,\ t_2,\ \ldots,\ t_m\}$.

Some computing processes, such as locating a disk, reading a file, numerical computation, transferring memory, etc., are executed in different parts of a computer system. These different processes can be executed concurrently if different processes are not interlaced, i.e. different threads could be processed in a parallel fashion. But when the client requests reach a certain level, the multithread technique may reach its limit. One simple option is to increase the computing capability for

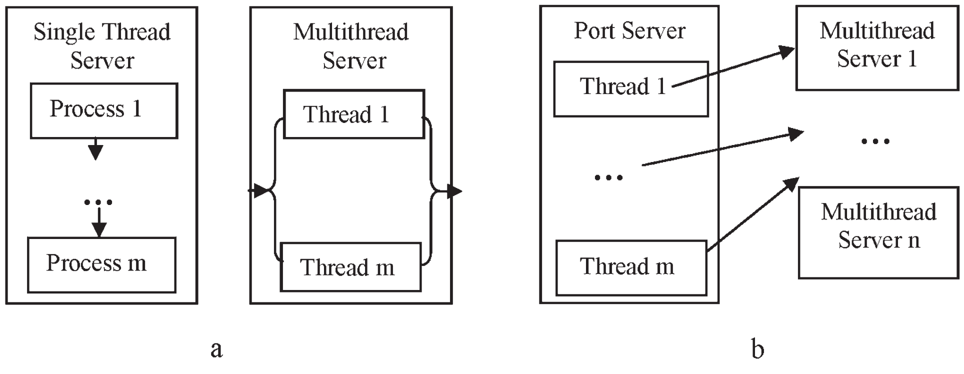

Figure 5. Adoption of multithread and cluster in WebGIS. (a) Multithread server in WebGIS. (b) Cluster in WebGIS.

that single server. However, this option will be dependent upon the current scientific and technological status of computing. Another viable option is to adopt the cluster technique.

Instead of using a single server, a cluster adopts multiple servers to support concurrent client accesses, which are distributed to different servers within a cluster. In addition, each server can use multithread techniques to process concurrent client requests, as illustrated in figure 5b. The National Geographic's Map Machine (http://plasma.nationalgeographic.com/mapmachine/index.html) uses this technique. When the client sends requests to the port server connected to the Internet, it channels all the process requests and responses. The port server can respond to more than one request at a time by allocating one thread to one request with multiple threads handling multiple requests. The port server will redirect requests to another multithread server for processing when the first multithread server reaches its capacity. A WebGIS server should be equipped with flexible capacity to serve concurrent requests by using cluster and multithread system configurations on the server side. Given this system architecture, performance of the WebGIS can be maintained or enhanced by upgrading the computers or adding a number of servers to meet the growth of concurrent accesses as needed. Therefore, this is a scalable solution that can accommodate the future growth in demand.

On the client side, a user often issues a request that involves many processes and accesses, and they can be accomplished sequentially. However, some of these processes or accesses could be performed independently without affecting each other. Then, the multithread technique can also be used to improve the performance on the client side. Given a request of 'zoom in' to a 10-layer map, the request may invoke 10 independent server accesses to the 10 layers. These 10 data access requests can be sent to the server at the same time through multithreading. As a result, using the multithreading technique can reduce the access time to possibly 1/10th of the required sequential access time in this specific example. Therefore, we adopt multithreading technique to send requests for different raster and vector layers simultaneously to the server.

5. Caching and dynamic data request

As illustrated in figure 6, when a request is issued, a WebGIS client will first examine whether the requested data are on the client. Then, the data request will be sent to the server only if the data are not on the client side. If the data are on the client side, the access will, of course, be much faster compared with the access across the

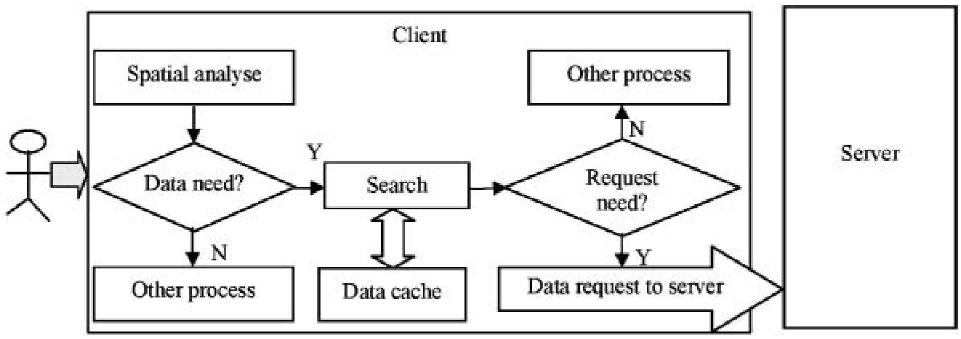

Figure 6. Cache and dynamic data request.

Internet from the server. Having some data residing temporarily on the client may also meet the need of subsequent data requests. Therefore, it will increase the efficiency of data access if some data are kept on the client by caching them in order to reduce the number of client requests sent to the server. Whenever data are needed, the client will first check whether the data are already available on the client side before sending a new request to the server. In a dynamic request and caching system, two issues need to be considered: what content should be cached and for how long?

Caching data on the client side can reduce the load on network transmission and server processing time, and thus improve the overall system performance. Ideally, data are required to be sent only once if all the data are duplicated on the client side. But this is impossible in most cases because the data sets may be too large to be completely cached on the client side. Therefore, based upon the user's interest or the frequently accessed data as reflected by coordinates, spatial data defined by these coordinates are cached for a certain duration before they are replaced by other data, which meet the more current needs of the user.

Frequently needed data should be cached to allow the system to respond as quickly as possible. Whether this technique will be effective or not is largely dependent on the specific applications. In general, basic data layers with a relatively small volume, such as the county boundary data for a state, should always be cached as a static layer. It should be loaded first and only once. Some frequently used data which are larger in volume, such as the district boundary data of the entire world, could be cached as needed. Some data which are not frequently used and have a small volume, such as the school data inside a county, could be cached once they are loaded. If a data set is not important (which has to be defined by the specific application environment) and with a large volume such as image background data, these data should not be cached. These scenarios are summarized in table 1 and can be implemented as options in a flexible function in the setup of a WebGIS. The system administrator or developer can customize the caching strategies based upon the data volume and characteristics of the data in the specific applications.

When many users access the system concurrently, the system should also cache certain frequently accessed data in the RAM on the server side to reduce the data access time from the permanent devices every time a data request is received. The first two strategies in table 1, therefore, can also be used for server-side caching. The third strategy also be used but requires an intelligent management of the server caching process.

6. Binary format and compression

The volume of transmitted data on the network greatly affects the performance of a distributed system. Our discussion in section 3 on pyramid and Hash indexing techniques can speed up primarily the handling of raster or image data on a WebGIS, and OGC has suggested a set of standards for the transmission of images

Table 1. Cache strategy at client side for different datasets.

Cache strategy	Use	Volume	Cache Architecture
Load at start-up and always cache	Frequent	Small	Static storage
Load once and always cache	Infrequent	Small	Semi-static cache
Load as needed and cache as needed	Frequent	Large	Dynamic cache
Load as needed but never cache	Infrequent	Large	No cache

in its Web Mapping Testbed (WMT) (http://www.opengis.org/datasheets/Dat04WMT2.000127.htm). In the WMT specification, OGC allows developers to decide which format to use for raster data. Users have many options, including the various compression technologies for raster data, such as gif and jpg file formats. In respect to vector data, OGC has proposed the Geography Markup Language (GML) for the transmission of data on the network. GML is a milestone for the interoperability of feature-based GIS data because it explicitly represents geographical data by using tags to mark objects and related information. But its performance is rather inferior because the data volume may increase by two- to threefold whenever GML is used.

In general, the two de facto data transmission formats are text-based and binary-based, and the transmission of compressed data can be regarded as the third method for comparison purposes. We take GML3.0 from OGC (2003) as a text-based data format, and the shapefile (ESRI 1998) as a binary-based format. Shapefile is a vector data file format used by ESRI primarily for its ArcView software package. The data in a shapefile are organized by records, each of which represents an object or feature in a GIS data layer or theme. All vector coordinate data are stored as binary data in a .shp file. To compress the binary shapefile, we can use the relative-value-compression method (ESRI 1997), which is based on the principle that vector data are stored as points, and consecutive points have minor differences in coordinates. Thus, the compression process records the first point's coordinate and stores only the difference in coordinates between the previous and subsequent points.

A set of data sets, which include points, lines and polygons, are randomly chosen to compare the data volume of these three different data storage and transmission methods. The results are shown in figure 7. The volume of the compressed format is the smallest as expected. Therefore, the compressed format is ideal and logical to be used to reduce the transmission volume over the Internet. We implemented the

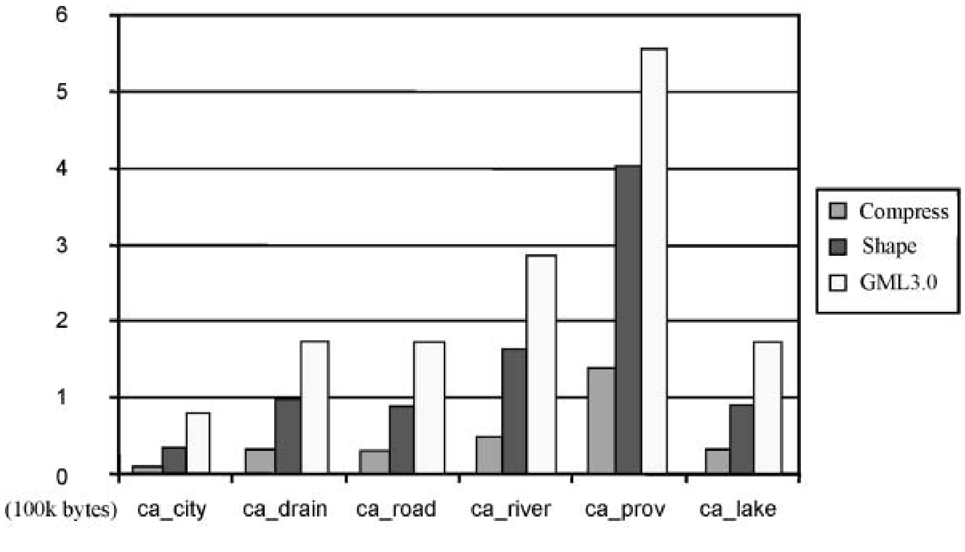

Figure 7. Data volume of three different formats from a set of Canadian Data (the 'ca' prefix stands for Canada) (Source: ArcGIS Data&Maps CD).

compression component on the server side and the decompression component on the client side to improve the performance of WebGIS.

7. Performance comparison

When various performance-improving techniques are adopted, the simple client–server distributed architecture can then be modified to become a relatively complex architecture (Yang and Li 2001) as illustrated in figure 8. In this modified and improved architecture, the cache, dynamic data request, and multithread techniques are implemented on the client side. The multithread, cache, pyramid, and cluster techniques are implemented on the server side. In addition, the data compression technique is used to compress data at the server and decompress data at the client to reduce the volume of data transmission. More information about the techniques and related problems, as well as implementation locations, is given in table 2.

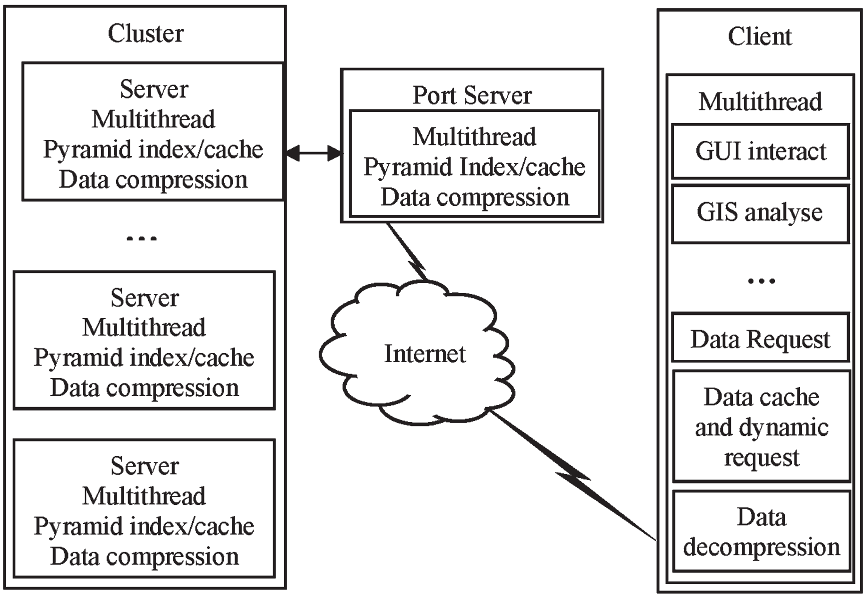

Figure 8. Performance-improving architecture of WebGIS.

Table 2. Performance-improving techniques.

Techniques	Problem addressed	Implementation
Pyramid, Cut, Hash index	Large image manage and publish	Server
Cache	Client user interact time and server data access time	Client/Server
Dynamic request	Client user data access time	Client
Cluster	Concurrent user access conflict	Server
Multithread	Concurrent user access and client user interaction	Server/Client
Binary compression	Network transmission load: Compress at Server and decompress at client	Server/Client

This relatively complex architecture is used in developing a prototype WebGIS (The CyberGIS Studio 1999, Yang 2000). The prototype is COM-based software developed using Visual C++ and Visual Basic (VB). The client GUI of the prototype is illustrated in figure 9. The performance of this WebGIS software is significantly improved using the relatively complex architecture. Figure 10 shows the detailed architecture used to develop the prototype: the client is an ActiveX control based upon VB. The server is a COM-based windows service using VC++. Each rectangle with small circle/line pairs attached is a component or ActiveX control. A small circle denotes the interface provided by the component to be called by other components. An arrow line denotes the interface–call relationship. In figure 10, User Interaction, Client Coordination, and Data Request on the client side adopt the multithread technique to ensure user interaction and data request parallelization. Client Coordination also controls the Cache, Data Decompression, and dynamic data request in the Map Management component. On the server side, Image Organization adopts the pyramid/cut/hash-index method. Server Coordination and Storage Data Access are multithread-enabled to ensure fast responses to concurrent users. The server also adopts the cluster technique to facilitate concurrent massive access.

In this section, we compare the performance of the prototype with and without different aforementioned techniques in several operations, including large image handling, client multi-layer accessing, client caching, two-computer clustering, and multi-user concurrent accessing. We use the same server and client configurations in the comparison. The servers have the following configuration: a 1.7 GHz Pentium 4 CPU, 1 Gb of RAM, and a 7200 rpm hard disk. The client configuration has the following parameters: 1.0 GHz Pentium III CPU, 256 Mb of RAM. The network speed is 10 Mbps.

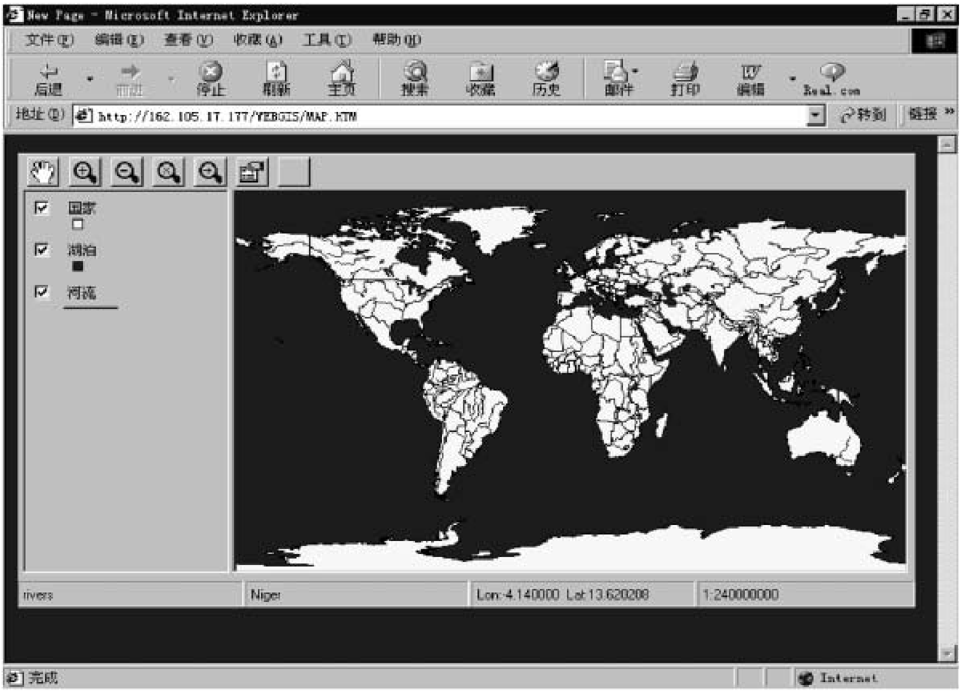
New Page - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 前进 停止 刷新 主页 搜索 收藏 历史 邮件 打印 编辑 Real.com
地址(D) http://162.105.17.177/WEBGIS/MAP.HTM
转到 链接
国家
湖泊
河流
rivers
Niger
Lon:4.140000 Lat:13.620208
1:240000000
完成
Internet

Figure 9. Client GUI of the WebGIS prototype.

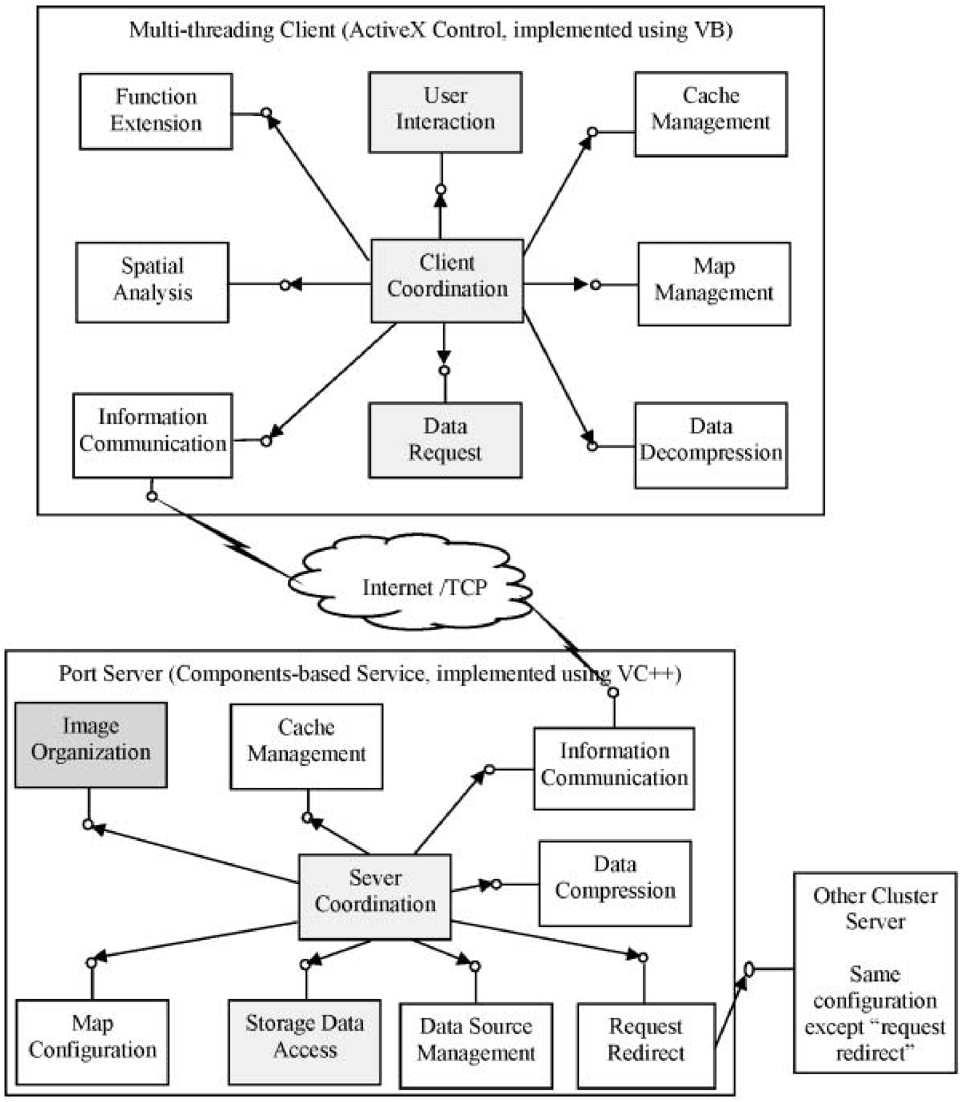

Figure 10. Architecture overview of the WebGIS prototype.

7.1 *Large image handling*

Different types of image data with different sizes are used in this part of the comparison. These data are 100 kbyte Moderate Resolution Imaging Spectroradiometer data (MODIS), 2.4 Mbyte Landsat Thematic Mapper data (TM), 79.3 Mbyte IKONOS data, 329 Mbyte Digital Orthophoto Quadrangles data (DOQ), and 1.2 Gbyte airborne photos. We organized these data in two different ways: (1) a single image and (2) the pyramid, hash-indexed organization introduced in section 3. The response times for accessing different images organized according to the two different ways are recorded. As illustrated in figure 11a, the second data organization method (pyramid with Hash index and cut) outperforms the first as the size of the data increases. In fact, the response times of the second method in handling all images are relatively stable for images of different sizes. Even with the

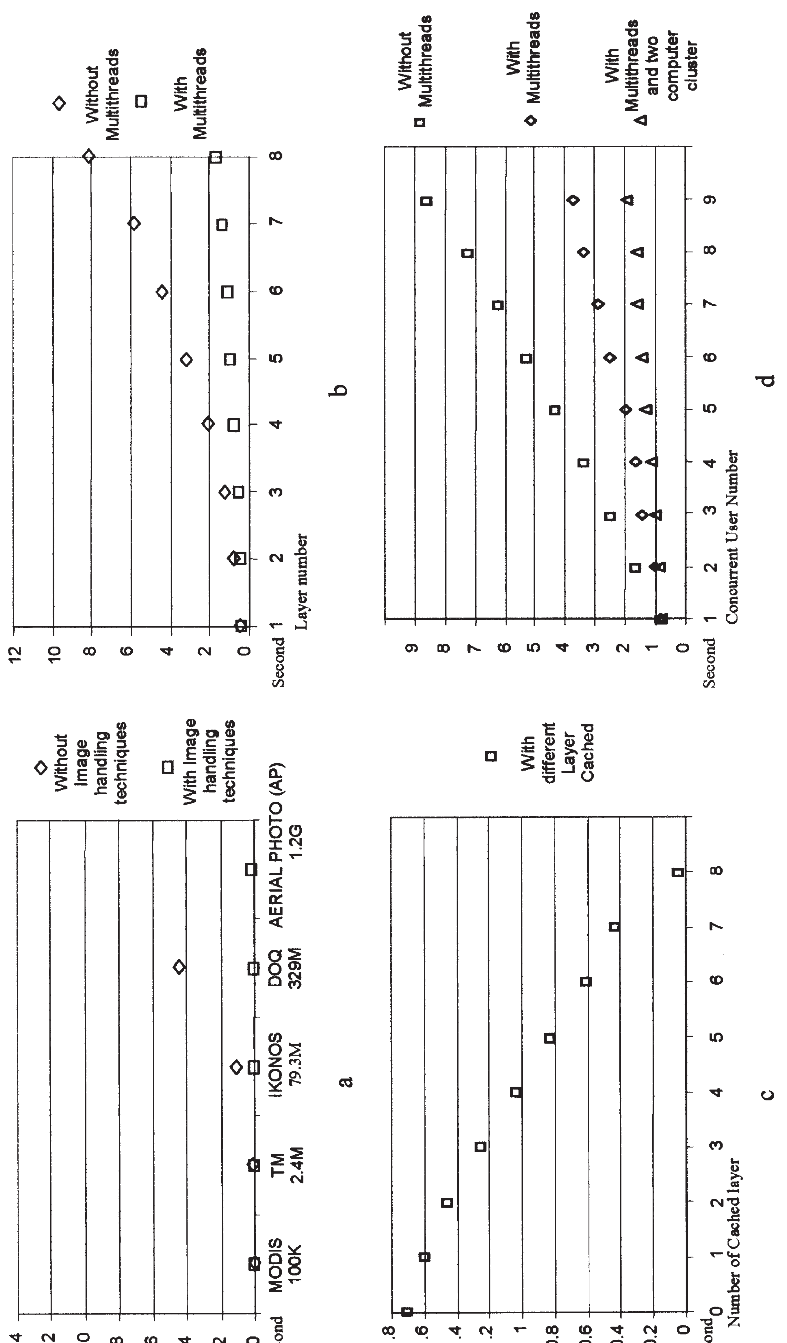

Figure 11. Response time of the WebGIS prototype with or without performance improving techniques. (a) Response time for handling different images of different data volume with or without the image-handling techniques. The response time of AP is too long to display on the same chart. (b) Client response time of multilayer accessing with or without multithread. (c) Client response time of multilayer access with different number of caching and without caching. (d) Server response time to concurrent users' access with three different settings: without multithreads, with multithreads, with multithreads and a two-computer cluster.

1.2 Gbytes of aerial photo data, the performance of the second method does not degrade significantly when compared with the handling of small images. Clearly, the first method has difficulty in handling the large images, especially the aerial photos within a reasonable time. Therefore, figure 11a does not show the result of using the first method to handle the aerial photos. Apparently, the performance of the second method is relatively independent of the data size, while the performance of the first method is inversely related to the size of the data.

7.2 *Multi-layer vector data accessing*

Geographic information is organized as layers in a hierarchical framework as shown in figure 1a. Normally, a WebGIS will organize the data into different layers, access these layers individually on the server side, and present the data layers on the client side. Whenever the client changes the display area or resolution, the operation will invoke requests to the multi-layer data sets. This common operation requires frequent access to each layer residing on the server. Therefore, the response time of the server will greatly affect the performance of WebGIS. We compare the performance of this multi-layer accessing process in the prototype with the multithread enabled and disabled. The data sets used in the comparison for this section and section 7.3 are found in the world data set of ArcGIS Data & Maps CDs provided by ESRI. They are geogrid—55 kbyte, cities—70 kbyte, world—30–141 kbyte, lakes—177 kbyte, drainage—299 kbyte, country—391 kbyte, rivers—464 kbyte, and latlong—547 kbyte, a total of eight layers. As illustrated in figure 11b, the response time for the multithread-enabled system increases much slower than that without the multithread technique, when the number of layers requested increases. Cache can also be used to reduce transmission time to improve the system performance. When one layer is cached, no transmission through the network is required when the layer is requested subsequently. The more layers are cached, the more transmission time will be saved, as shown in figure 11c.

7.3 *Multi-user concurrent accessing*

It is likely that many users will access the same WebGIS application when it is opened to the public. The ability to respond to many users simultaneously within a reasonable time is critical from a pragmatic perspective. We compare the performance of multi-user concurrent access between the multithreads-enabled server and the server with a single thread. The multi-user concurrent access is generated at the client side automatically by issuing multiple data requests, and only four layers are requested for each user access. Results are shown in figure 11d. When the number of concurrent users increases, the response time of the single-thread server increases dramatically, while the response time of the multithread-enabled server increases at a much slower pace. When the number of concurrent users exceeds eight, the single-thread server response exceeds 7 s, which may not be acceptable to some users, while the multithreads server still functions very well. When the cluster technique is added onto the multithreads server, as show in figure 11d, it further improves the performance of the system especially when the number of concurrent users increases. Moreover, for a given time-out value set for a system, adopting the cluster technique can help to support a larger number of users. For example, the single thread server can support two users with a time-out of two seconds, while the multithread server can support five users, and the two-computer cluster can support up to nine users, as illustrated in figure 11d.

8. Conclusions

This paper discussed the use of several methods in enhancing the performance of WebGIS. We also demonstrated that using these techniques increases system performance substantially. The pyramid-hash index improves the efficiency of publishing large images. Cluster and multithread methods can increase the efficiency to handle massive concurrent accesses to the servers. Cache and dynamic data management are used to improve client-side interactive performance. The binary and compression techniques are suggested to reduce the data transmission volume. Using the prototype system, this paper has also demonstrated that the system adopting these techniques has significantly out-performed the same system without these techniques in handling raster and multi-layer vector data, and in accommodating massive access.

Therefore, the techniques discussed in this paper can be used in the design and development of WebGIS to enhance performance. These techniques will be especially critical in the deployment of large-scale data dissemination projects. Adopting these techniques and handling those implementation issues appropriately can ensure that the system meets the current demand (CEOSR 2002), as well as future demand, which most likely will increase due to the proliferation of the use of spatial data in all aspects of our 'digital' or 'electronic' society. These techniques can also be used for constructing geospatial information service and interoperable systems for sharing vast amount of geospatial data, such as the services provided by the Geospatial One-Stop Portal (http://www.geodata.gov/).

Our emphases in this paper so far are on the significant performance problems one may encounter during the process of building a WebGIS. These problems are mostly software- or hardware-oriented. Performance of a WebGIS is also related to other issues, such as data contents and formats. Multimedia data, for instance, will require special treatments in order to be transmitted efficiently. Research and development in computer science in general, and in the distributed computing environment in particular, will provide some innovative methods to improve the performance of WebGIS. For example, the newly developed wavelets compression method and JPEG 2000 format (http://www.jpeg.org/CDs15444.html) can be used to further reduce data volume.

While our discussions focused mainly on spatial data handling in a client–server environment, 'true' GIS on the web should also support more sophisticated GIS functions and spatial analytical functions (Yang *et al.* 1999). The performance of such full-scale GIS will definitely be a concern. Specific geospatial methods and functions developed by geographic information scientists will be required for implementation in a distributed manner efficiently and with superior performance. For instance, real-time traffic information is required for intelligent transportation systems. One may select optimized routes based upon the real-time traffic information. In order to improve the performance of the real-time systems, road network and related traffic information have to be organized in an efficient manner to support the complex spatial analytical methods. In a distributed and cooperative spatial analysis environment, such as the 'data to models' and 'data to interpretation' analyses in the mid-ocean ridge research (Wright *et al.* 2003), it will be necessary to investigate how to improve the performance to support distributed model-based spatial analysis. Some issues to be considered include where the model should reside and operate, where the model data are stored, and in what formats. The performance issue will become even more complicated when heterogeneous systems, such as remote sensing, GIS, and GPS are included (Xue *et al.* 2002). We expect these issues to be be on the future research agenda of WebGIS.

Acknowledgements
Comments provided by Professor Harvey Miller, the North American Editor of *IJGIS*, and the three anonymous reviewers are gratefully acknowledged.

References

Baja, C.L., Pascucci, V. and Zhuang, G., 1999, Single resolution compression of arbitrary triangular meshes with properties, In *Proceedings of IEEE Data Compression Conference'99* (Snowbird, UT: IEEE), pp. 247–256.

Barclay, T., Gray, J. and Slutz, D., 1999, *Microsoft TerraServer: A Spatial Data Warehouse, Microsoft Technical Report*, http://research.microsoft.com/users/Tbarclay/MicrosoftTerraServer_TechnicalReport.doc

Barnsley, M., 1989, *Fractals Everywhere* (San Diego, CA: Academic Press).

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

Becker, B., Six, H.-W. and Widmayer, P., 1991, Spatial priority search: An access technique for scaleless maps. In *Proceedings of ACM SIGMOD* (Denver, CO: ACM), pp. 128–137.

Beckmann, N., Kriegel, H.P., Schneider, R. and Seeger, B., 1990, The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (Atlantic City, NJ: ACM), pp. 322–331.

Bertolotto, M. and Egenhofer, M.J., 2001, Progressive transmission of vector data over the World Wide Web. *GeoInformatica*, **5**, pp. 345–373.

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

Buehler, K. and McKee, L., 1998, *The OpenGIS Guide*, 3rd ed., http://www.opengis.org/techno/guide/guide/Guide980629.rtf.

Buttenfield, B.P., 2002, Transmitting vector geospatial data across the Internet. In *Geographic Information Science—Second International Conference GIScience 2002*, M. Egenhofer and D. Mark (Eds.), Lecture Notes in Computer Science Vol. 2478 (Berlin: Springer), pp. 51–64.

CEOSR, 2002, *VAccess-MAGIC (Virginia Access & Mid-Atlantic Geospatial Information Consortium) Annual Report* (Fairfax, VA: George Mason University).

Chen, S., Wang, X., Rishe, N. and Weiss, M.A., 2000, A high-performance Web-based system design for spatial data access. In *Proceedings of the Eighth ACM Symposium on Advances in Geographic Information Systems* (Washington, DC: ACM), pp. 33–38.

Coors, V. and Flick, S., 1998, Integrating levels of detail I a Web-based 3D GIS. In *Proceedings of the Sixth ACM International Symposium on Advances in Geographic Information Systems* (Washington, DC: ACM), pp. 40–45.

Cowen, D.J., 1994, The importance of GIS for the average person. In *Proceedings of First Federal Geographic Technology Conference* (Washington DC: National Research Council), pp. 7–11.

Davis, G., 1998, A wavelet-based analysis of fractal image compression. *IEEE Transactions on Image Processing*, **7**, pp. 141–154.

Deng, W., 2000, WebGIS—A new way for public participation in city planning. In *GIS'2000*, 13–16 March 2000, Toronto, Ontario.

Dragicenvic, S., 2000, Environmental data exploration: The web GIS approach. In *GIS'2000*, 13–16 March 2000, Toronto, Ontario.

Edward, P.F.C. and Kevin, K.W.C., 2002, On multi-scale display of geometric objects. *International Journal on Data and Knowledge Engineering*, **40**, pp. 91–119.

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

ESRI, 1997, *Getting Started with SDE*, An ESRI White Paper.

ESRI, 1998, *ESRI Shapefile Technical Description*, An ESRI White Paper, http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

Floriani, L.D., Magillo, P. and Puppo, E., 1998, Efficient implementation of multi-triangulations. In *Proceedings of IEEE Visualization '98* (Research Triangle Park, NC: IEEE), pp. 43–50.

Galinao, B. and Brennan, C., 2002, Power to the people! A Web-based GIS provides a public-involvement tool for airport development. *GeoWorld*, **15**, pp. 32–35.

Goodchild, M.F., 1992, Geographical data modeling. *Computers and Geosciences*, **18**, pp. 401–408.

Grønbæk, K., Vestergaard, P.P. and Ørbæk, P., 2002, Towards geo-spatial hypermedia: Concepts and prototype implementation. In *Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia* (College Park, MD: ACM), pp. 117–126.

Guttman, A., 1984, R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Boston, MA: ACN), pp. 47–54.

Hoppe, H., 1996, Progressive meshes. In *Proceedings of SIGGRAPH'96* (New York: ACM), pp. 99–108.

Huang, B., Jiang, B. and Lin, H., 2001, An integration of GIS, virtual reality and the Internet for visualization, analysis and exploration of spatial data. *International Journal of Geographic Information Science*, **15**, pp. 439–456.

Kang, Y.-K., Kim, K.-C. and Kim, Y.-S., 2001, Probability-based tile pre-fetching and cache replacement algorithms for web geographical information systems. In *Proceedings of ADBIS'2001* (Vilnius, Lithuania: ACM), pp. 127–140.

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

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

Kern, P. and Carswell, J.D., 1994, An investigation into the use of JPEG image compression for digital photogrammetry: Does the compression of images affect measurement accuracy? In *Proceedings of EGIS'94*, Paris, pp. 694–701.

Koeppel, I., 2001, *GIS extended to the wireless & internet world*, http://www.esri.com/news/arcnews/winter0001articles/gisextended.html

Kwon, J.-H. and Yoon, Y.-I., 2002, Efficient access technique using levelized data in web-based GIS. In *Proceedings of WAIM'2002* (Beijing: Springer), pp. 64–71.

McCurley, K.S., 2001, Geospatial mapping and navigation of the Web. In *Proceedings of ACM 10th Conference on World Wide Web* (New York: ACM), pp. 221–229.

Morlet, J. and Grossman, A., 1984, Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM Journal on Mathematical Analysis*, **15**, pp. 723–736.

OGC, 2003, *Geography Markup Language (GML) v3.0*, OGC Document Number: 00-029, http://www.opengis.org/techno/documents/02-023r4.pdf

OOSTEROM, P.V., 1991, The reactive-tree: A storage structure for a seamless, scaleless geographic database. In *Proceedings of Auto-Carto'10* (Baltimore, MD: ACSM-ASPRS), pp. 393–407.

PENG, Z.R., 1999, An assessment framework for the development of Internet GIS. *Environment and Planning B—Planning & Design*, **26**, pp. 117–132.

PENG, Z.R. and TSOU, M.H., 2003, *Internet GIS: Distributed geographic information services for the Internet and wireless networks*, pp. 2–5 (Hoboken, NJ: Wiley).

PLEWE, B., 1997, *GIS Online: Information Retrieval, Mapping, and the Internet*, pp. 6–14 (Santa Fe, NM: OnWord Press).

RAUSCHENBACH, U. and SCHUMANN, H., 1999, Demand-driven image transmission with levels of detail and regions of interest. *Computers and Graphics*, **23**, pp. 857–866.

ROGUL, D., 2003, *Geographical Information (GIS) Market Trends*, Faulkner Information Services, http://www.faulkner.com/products/faulknerlibrary/pdf/00016293.pdf

SAMET, H., 1984, The quadtree and related hierarchical data structure. *ACM Computing Surveys*, **16**, pp. 187–260.

SELLIS, T., ROUSSOPOULOS, N. and FALOUTSOS, C., 1984, The R+-tree: A dynamic index for multidimensional objects. In *Proceedings of the 13th International Conference on VLDB* (Singapore: Morgan Kaufmann), pp. 507–518.

SHEN, G., 2001, WebGIS tools for online spatial data exploration and analysis in business. *WebNet Journal (United States)*, **2**, pp. 16–53.

SLATER, S., 2002, Queensland a princely GIS. *Geoworld*, **15**, pp. 42–45.

SOLOMON, D. and RANKINGS, R., 1997, *Microsoft SQL Server 6.5 Unleashed*, pp. 1–15 (Indianapolis, IN: Sams).

SRINIVAS, B.S., LADNER, R., AZIZOGLU, M. and RISKIN, E.A., 1999, Progressive transmission of image using MAP detection over channels with memory. *IEEE Transactions on Image Processing*, **8**, pp. 462–475.

SUI, D.Z. and GOODCHILD, M.F., 2001, GIS as media. *International Journal of Geographic Information Science*, **15**, pp. 387–390.

TAKATSUKA, M. and 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.

The CyberGIS Studio, 1999, *WebGIS System Development Document*, Research Report 97-759-04, Peking University.

TU, S.R., HE, X., LI, X. and RATCLIFF, J.J., 2001, A systematic approach to reduction of user-perceived response time for GIS Web services. In *Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information System* (Washington, DC: ACM), pp. 47–52.

WEI, Z., OH, Y., LEE, J., KIM, J., PARK, D., LEE, Y. and BAE, H., 1999, Efficient spatial data transmission in Web-based GIS. In *Proceedings of the 2nd WIDM* (Kansas, MO: ACM), pp. 38–42.

WORBOYS, M.F., 1995, *GIS: A Computing Perspective* (London: Taylor & Francis).

WRIGHT, D.J., O'DEA, E., CUSHING, J.B., CUNY, J.E. and TOOMEY, D.R., 2003, Why Web GIS may not be enough: A case study with the virtual research vessel. *Marine Geodesy*, **26**, pp. 73–86.

WU, J., AMARATUNGA, K. and CHITRADOH, R., 2002, Design of distributed interactive online geographic information system viewer using wavelets. *Journal of Computing in Civil Engineering*, **16**, pp. 115–123.

XUE, Y., CRACKNELL, A.P. and GUO, H.D., Telegeoprocessing: the integration of remote sensing, Geographic Information System (GIS), Global Positioning System (GPS) and telecommunication. *International Journal of Remote Sensing*, **23**, pp. 1851–1893.

YANG, C., 2000, *Theory, Techniques and Implementation Methods of WebGIS*, Unpublished Ph.D. dissertation (Beijing: Peking University) (in Chinese).

Yang, C. and Li, Q., 2001, Research on Web publishing of spatial information. *Acta Scientiarum Naturalium Universitatis Pekinensis*, **37**, pp. 413–420 (in Chinese).

Yang, C., Li, Q., Cheng, J., Qi, R., Huang, L. and Zhang, D., 2000, Research and implementation on Web publishing of remotely sensed images. *Journal of Remote Sensing*, **4**, pp. 71–75 (in Chinese).

Yang, C., Li, Q. and Wang, J., 1999, Research on distribution of spatial object relation computation. *Chinese Journal of Image and Graphics*, **4**, pp. 331–335 (in Chinese).

Zhao, Y. and Yuan, B., 1996, A hybrid image compression scheme combining block-based fractal coding and DCT. *Signal Processing: Image Communication*, **8**, pp. 73–78.