Computing and Informatics, Vol. 24, 2005, 1001–1016, V 2005-Oct-24

A HIERARCHICAL COMPONENT-BASED WEBGIS AND ITS KEY TECHNOLOGIES

Yingwei Luo, Xiaolin Wang, Wenjun Wang, Zhuoqun Xu

Department of Computer Science and Technology
Peking University, Beijing, China, 100871
e-mail: lyw@pku.edu.cn

Li Ding

Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County, USA

Manuscript received 2 September 2004; revised 5 August 2005
Communicated by Ladislav Hluchý

Abstract. A practical hierarchical component-based WebGIS model referred to as Geo-Union is presented. Geo-Union consists of four layers: storage layer, service layer, component layer and application layer. Service layer is partitioned into another two layers: Geo-Union client and Geo-Union server. The architectures and object diagram of each layer in Geo-Union are discussed in details. After that, four key technologies adopted in Geo-Union (spatial data model, ORDB, spatial index and spatial cache) are summarized and analyzed, especially the spatial cache framework of Geo-Union. At last, some future works in WebGIS, such as interoperability, security, distributed computing and intelligent computing, are indicated and simply explored.

Keywords: Component, WebGIS, hierarchical model, geo-union, spatial cache, spatial index

1 INTRODUCTION

Geographical Information System (GIS) is well known as software system specialized in capturing, representing, managing and visualizing spatial information. It has been widely used in governmental and commercial domain [1, 2].

Along with the fast growth of the Web, WebGIS, i.e. the deployment of GIS in the Internet, emerges with a growing impact on our daily life [3], e.g. GPS systems and web based digital map service. There are two categories of WebGIS on the current market:

- digital street map services, such as MapQuest (`http://www.mapquest.com`), Yahoo Maps (`http://maps.yahoo.com`), and Google Maps (`http://maps.google.com`);
- professional WebGIS platforms, such as ESRI ArcIMS (`http://www.esri.com`), Mapinfo MapXtreme (`http://www.mapinfo.com`), Autodesk MapGuide (`http://www.autodesk.com`), Intergraph GeoMedia Web Map (`http://www.intergraph.com`), and GeoStar GeoSurf (`http://www.geostar.com.cn`).

Although these commercial WebGIS applications and platforms are implemented using various technologies (e.g. CGI/Server API/Servlet, plug-in, CORBA/DCOM, Java Applet or mix of them), their design principles fall into two categories: *thin-client* and *fat–client*. The thin-client approach assigns all spatial data management tasks and most spatial computation tasks to the server, and leaves the client as simple as possible. A client simply collects users' requests, then forwards the requests to a central server, and then displays the result (usually in raster format) on screen. This principle minimizes the computational requirement on client side at the expense of heavy network traffic and server side computational load. On the contrary, the fat-client approach puts most of the computation on the client side and makes the server a data provider. A client needs to load raw spatial data (usually in vector format) from the server, to perform spatial computation and to render vector graphics. This principle reduces the interoperations between server and client but requires non-trivial computational resource on the client side.

The above two principles are in fact two design extremes, and a customizable approach is preferred in Web context due to various users' requirements. In addition, performance is also a critical issue in designing WebGIS since Web users expect the response within seconds. This paper presents a component-based approach to WebGIS, Geo-Union, which has been successfully used in several projects. Its multi-layer architecture makes it easy to customize WebGIS according to the distribution of computational resource. Its component based design enables portable spatial computation, i.e. spatial computation can be run on both client and server sides. Its spatial cache framework and spatial index mechanisms additionally enhances the performance of GIS for Web context.

2 GEO-UNION ARCHITECTURE

Component modeling has been widely used in providing customizable WebGIS [4, 5] due to its flexibility, extensibility and reusability. Beside component technologies, Geo-Union provides WebGIS service in multi-client/server mode through four consequent layers: storage layer, service layer, component layer and application layer [6, 7, 8] (see Figure 1). The rest of this section details these four layers.

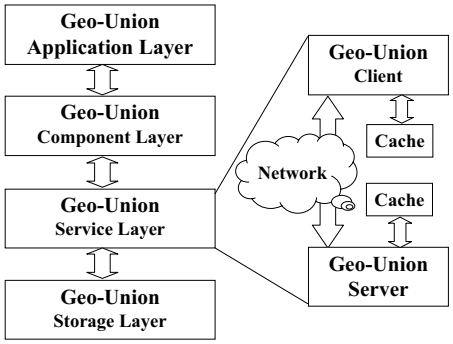


Fig. 1. Architecture of Geo-Union

2.1 Geo-Union Storage Layer

Storage layer, which focuses on GIS data modeling and storage, is the fundamental part in Geo-Union. The data model in Geo-Union consists of two categories:

- geospatial data model, which represents geospatial objects in vector format [8];
- rendering profile, which details how to render geospatial objects in a map [7].

Figure 2 overviews seven important data objects from both categories of the Geo-Union data model. 'Entity' and 'layer' belong to the geospatial data model. There are over twenty sub-classes of 'entity' from point, line, and polygon to complex entities (see Section 3.1). 'Entities' with the same attribute fields (semantic locality) and within a certain extent (spatial locality) are organized by a 'layer'. The remaining five objects in Figure 2 belong to rendering profile category.

Object-relational database (ORDB) has been chosen for storing GIS data in Geo-Union concentrates since it is a matured technology and it seamlessly stores GIS data with application specific attribute data together (see Section 3.2).

2.2 Geo-Union Service Layer

Geo-Union service layer is responsible for GIS data access. The service layer consists of two parts: *Geo-Union client* and *Geo-Union Server*.

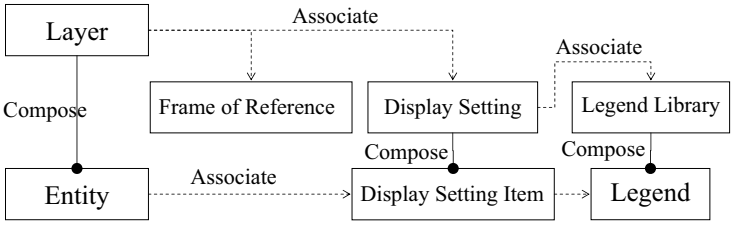


Fig. 2. E-R Diagrams in Geo-Union storage layer

Geo-Union server provides spatial data access services and spatial relation query services, while Geo-Union client provides component library for accessing Geo-Union server from client side. As shown in Figure 3, a Geo-Union server can serve multiple Geo-Union clients simultaneously through 'connection object'. Geo-Union server maintains a data source table for accessing multiple distributed spatial databases.

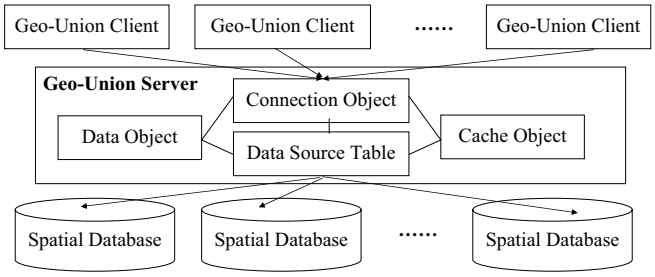


Fig. 3. Geo-Union server and Geo-Union client

Figure 4 shows the architecture of Geo-Union client. It is designed for accessing data services provided by Geo-Union Server. Besides, in-memory objects for storing and accessing elements in spatial data model, 'Connection' is introduced for managing data transfer between client and server. In addition, 'Algorithm' includes a library of spatial analysis algorithms and corresponding data structures.

In order to transfer GIS data efficiently in the Internet, distributed spatial cache (see Section 3.4) and multi-resolution spatial index (see Section 3.3) have been used. In addition, redundant information has been generated to reduce multi-table query, and redundant information remove-and-recover mechanisms can be used when network bandwidth is the bottleneck.

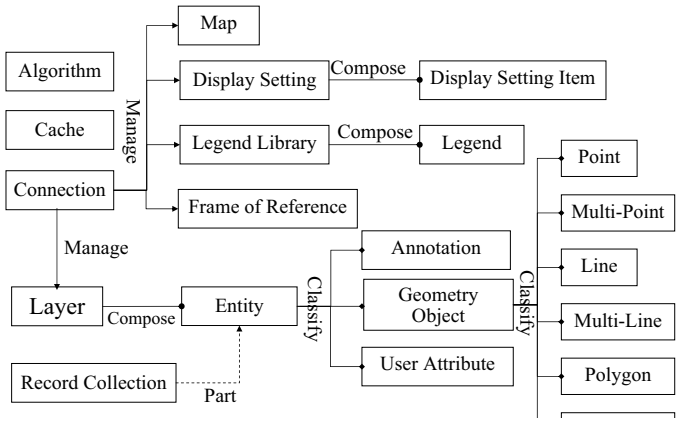


Fig. 4. Design architecture of Geo-Union client

2.3 Geo-Union Component Layer

The component layer is built on top of Geo-Union client and offers a library of components for customization. The components fall in the following categories (see Figure 5):

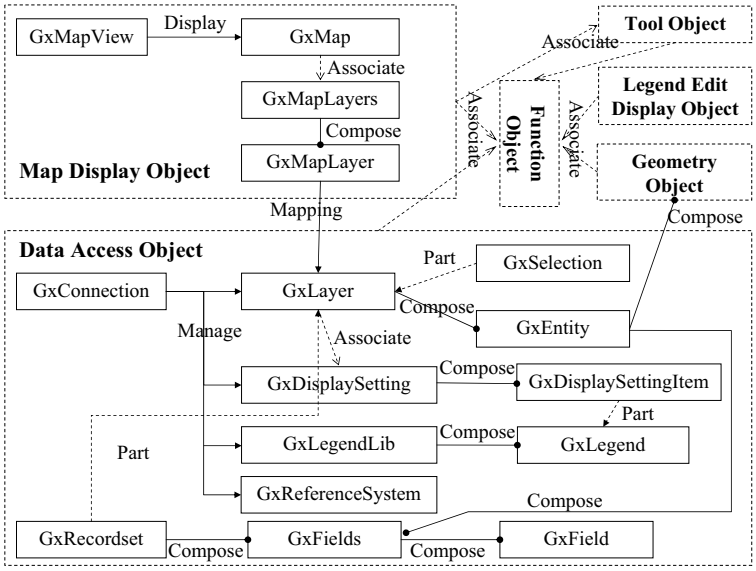


Fig. 5. Object diagram of the component layer

- data access objects, which wrap Geo-Union client's data access functions;
- map display objects, which manage users' map-display preference;
- tool objects, which process users' input via mouse and keyboard;
- legend edit display objects, which manage and render legend for a map;
- geometry objects, which let users manipulate spatial information in detail;
- function objects, which assembly components to complete certain tasks, e.g. compute and display a shortest path on map.

2.4 Geo-Union Application Layer

The application layer is designed for customizable GIS application. With the component based design, users may develop a conventional GIS application as well as a WebGIS application. Figure 6 overviews two alternative options for building a WebGIS application. A fat-client option could be achieved by deploying Geo-Union components on client side using DCOM/ActiveX technology, and GIS data is transferred to client side via DCOM communication between Geo-Union client and server at service layer. Alternatively, a thin-client option could be achieved by deploying Geo-Union components and building WebGIS application on server side; therefore, users could access GIS data directly from their web browsers and the web server will be responsible to interact with WebGIS application for GIS data and services as well as database for non-spatial application specific data.

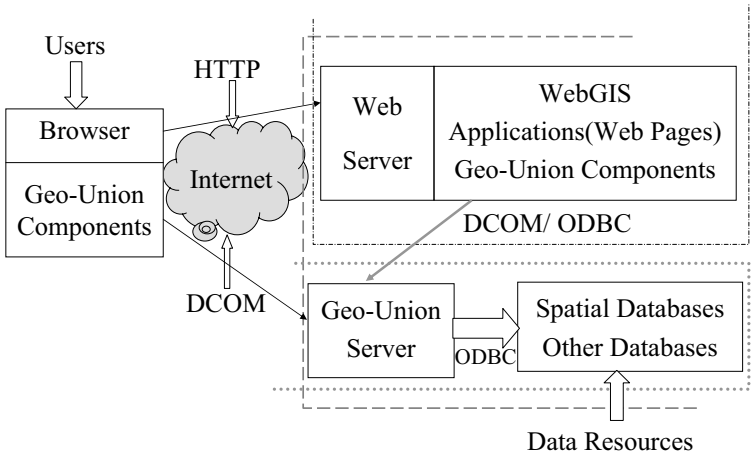


Fig. 6. Building WebGIS using Geo-Union

3 KEY TECHNOLOGIES IN GEO-UNION

3.1 Geography Spatial Data Model in Geo-Union

Geometry and attribute factor are the most fundamental characters of spatial entity. These two factors need to be considered to classify a spatial entity because of its complexity and diversity. Therefore, in Geo-Union, spatial entity can be classified into four kinds, which are 0-dimension, 1-dimension, 2-dimension and complex entity, and more than twenty sub-kinds [8].

In Geo-Union, spatial data are organized as map layers. Map layer is an organic collection of spatial entities with the same attribute factor within some special extent. However, map layer is not simply assembled by spatial entities. It is compounded by the spatial entities in some special application domain and has intimate relationship among the entities. In Geo-Union spatial data model, map layer is the container of spatial entities and the basic unit of spatial data organization.

Due to huge spatial data and complex spatial access and analysis algorithms, efficient process of spatial data is a hinge. Considering that, we have some ideas about spatial data model:

- First, spatial data relations including topology relation, distance relation etc. are so complex that they require large computation. Therefore, when designing the spatial data structure, we must use redundant information to store various relations, thus improving data access and analysis efficiency.
- Second, network data transfer usually becomes a bottle-neck when accessing spatial data, because spatial data is distributed in the network and distributed spatial computation inevitably results in flowing of spatial data in the network. Therefore, to reduce network transfer of spatial data, spatial data is classified into basic and derived data. Basic data requires network transfer, but derived data can be generated through computing on basic data.

Of course, a suitable compromise should be made among storage efficiency, analysis efficiency and access efficiency.

3.2 Store Spatial Data in ORDB

In an ORDB, users can define complex non-relative objects including data structures and operations, so as to support various complex data type and data process, such as multimedia, map, text and so on. Moreover, ORDB often has the optimized capability to operate large data block, thus helping to store huge complex data. These characteristics facilitate the integration of GIS and database system to develop integrative GIS [9].

Generally, map layer and spatial entity are basic management units of spatial data. In ORDB, a map layer can be stored as a table. A spatial entity is a record in the table, the geometry information of the spatial entity is a large data block of the

record together with its attribute information. In this way, spatial database system has consummate data security, integrity, consistency control mechanism as well as high concurrency and huge data management capability. Openness, maintenance means, management means, etc. of ORDB are also consummate.

Using ORDB to store spatial data can enable to take full advantage of various services provided by ORDB. Hence we can focus mainly on the problems with most GIS characteristics.

Certainly, we should take full advantage of ORDB and should not be confined by ORDB. Special GIS data access service component is used here to achieve this goal as a bridge between database client and database server.

3.3 Grid Spatial Index for Geo-Union

When designing spatial database, spatial index is often constructed to enhance the efficiency of data access and management. Different spatial index structure and management technique directly affect a GIS system performance. The complexity of spatial data determines the complexity of spatial index structure. In Geo-Union, two simple but practical spatial index techniques are adopted: layered grid spatial index and 3-ary tree grid spatial index [10].

The basic concept of layered grid spatial index can be described as follows.

A rectangle geospatial extent will be partitioned n times. In the first partition, the rectangle geospatial extent is partitioned into M_1 rows and N_1 columns ($M_1 \times N_1$ grids), with length L_1 and width W_1 of each unit. In the second partition, the rectangle geospatial extent is partitioned into units with length L_2 and width W_2 (here, $L_2 > L_1$, $W_2 > W_1$, generally, $L_2 : L_1 = W_2 : W_1 \geq 2$). As above, the extent will be partitioned into 1 row and 1 column in the n^{th} step, only one grid with the whole rectangle geospatial extent. Then each dividing grid is given a unique number, corresponding to a dividing grid. The first $M_1 \times N_1$ dividing grid is called layer 1, and the last 1×1 grid is layer n, where:

$$n = Integer(\log_{\frac{W_2}{W_1}} \max(M, N))$$

The index number of each entity is assigned as the following rules: if the entity's extent is within a grid's rectangle after some steps' partition, the index number will be the unique number of that partitioned grid.

The key problem of layered grid spatial index is how to divide the grid as well as how many entities are contained in each grid. The number of entities contained in each index is the main effect of index efficiency. If the number is too large, many useless entities will also be involved in each access, thus making index meaningless; if the number is too small, spatial and time spending will increase significantly so that performance will also be affected. Therefore, it is the best way to specify the number of entities contained in each grid to be ideal. Definition of this perfect number is relative with memory size, memory page size, exchange policy between

memory and disk and network performance of a system, as well as the storage size of the map layer and the average storage size of all entity in the map layer.

Because distribution of entities in a map layer is not well-proportioned, in other words, entities in one sub-extent of the map layer are dense while entities in another sub-extent are rare. Fixed grid partition in layered grid spatial index technique results in the fact that entities in some grids are dense while entities in other grids are rare or even empty. Furthermore, the values of M, N and step size are difficult to define for non-professional users. The ideal grid algorithm can surely equipoise the entities in a grid. That is to say, grid size will be small if the entities' distribution is dense, while grid size will be large if the entities' distribution is rare. To satisfy this requirement, a 3-ary tree grid spatial index is implemented. Its core thought can be described as follows.

According to the number of spatial entities in layered grid index, each grid is self-partitioned to generate a layered grid partition tree. After each partition of a grid, original grid becomes three non-balanced parts (sub-grids): Partition1 (left or top part), Partition2 (right or bottom part) and Middle (entities that span Partition1 and Partition2 are contained here). Thus, the generated partition tree is a 3-ary tree. Each node (grid) in 3-ary tree is described as a 7-tuple:

<
Parttition1 (left or top part),
Partition2 (right or bottom part),
Middle (span partition part. it will be a NULL or point to a leaf node),
Scope (partition extent, in a non-leaf node, the scope is the union of all its child nodes' extents),
CutLine (if positive real number, it means X coordinate [left/right] partition. If negative real number, it means Y coordinate [top/bottom] partition),
EntityList (in a leaf node, EntityList is the array of entities contained in the node),
EntityNumber (number of entities in the node)
>

Beside spatial data management and access, the above two spatial index techniques are also used in various spatial queries, spatial analysis algorithms and map display algorithms. This technique not only improves the efficiency of spatial data access, but also enhances the efficiency of spatial query and analysis as well as map display speed in the client.

3.4 Spatial Cache Framework for Geo-Union

Cache is an important technique for improving data access performance in software systems. In Geo-Union, there are two time consuming data access operations:

(i) accessing spatial data in databases at storage layer, and
(ii) transferring spatial data in network at service layer.

In order to improve their performance, a spatial cache framework has been designed and implemented in Geo-Union.

3.4.1 Spatial Cache Framework

Figure 7 overviews the design of spatial cache framework in Geo-Union. The framework includes three types of cache, namely database spatial cache, network spatial cache and spatial data proxy server.

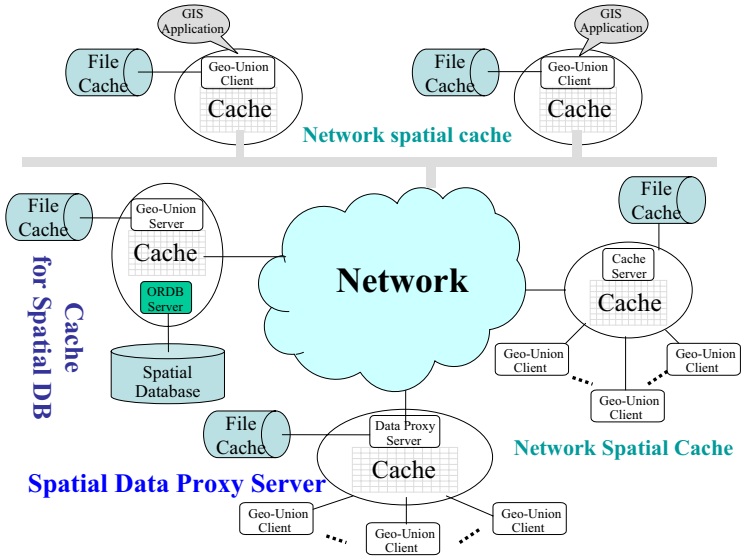


Fig. 7. Spatial cache framework

Database spatial cache. Geo-Union server is the bridge between Geo-Union client and ORDB, so cache for spatial database is established and maintained by Geo-Union server. The database cache is maintained in both memory and file system; therefore, users' data access request will be processed by looking up memory cache, file cache and database in sequence. Maintaining such a three-level cache is especially useful for GIS applications, which usually load a group of spatial entities in bulk based on their spatial locality. While Geo-Union server can support multiple Geo-Union clients simultaneously through 'connection' object, the clients are in fact accessing the same set of spatial data sources managed by Geo-Union server; hence the database cache is maintained globally in Geo-Union server rather than one instance per user connection so as to enable better.

Network spatial cache. Since network bandwidth is quite different in different context and lots of web users are not equipped with high-speed network, net-

work remains to be a critical bottleneck in WebGIS applications. To address this issue, Geo-Union supplies two approaches to build spatial cache on client side. First, the network spatial cache could be maintained on the end-users' machine, it maintains a partial cache for downloaded spatial data. Second, the network spatial cache could be maintained on a high capacity/performance machine in a high-speed LAN serving a group of end-users (cache server). These two approaches could be further combined for better performance.

Spatial data proxy server. Spatial data proxy server is created by an overloaded Geo-Union server to serve as an alternative data service provider. It can be viewed as the extension of cache server (the structure and implementation of spatial data proxy server is the same as cache server); however, it differs from cache server in that (i) it is deployed in the Internet and open to all public users while cache server only serves privileged users within LAN; (ii) spatial data proxy server is an initiative cache server, it has its own strategies to cache all or part of spatial data in spatial database, not according to client's requests; and (iii) it aims to share the load of an overloaded Geo-Union Server while cache server aims at reducing requests to Geo-Union server.

3.4.2 Organization of Spatial Cache

In Geo-Union, data in spatial cache is organized as three levels of granularity: layer, slot and entity.

Layer is the top level granularity and it is identified by *layereID*, i.e. a Global Universal Identification (GUID) which is generated when the layer was created in spatial database. Each layer in spatial cache occupied an exclusive space. A *layerVersion* attribute is attached to each layer so as to determine whether a cached layer is still valid, and a cached layer will be flushed if its *layerVersion* is different from its *layerVersion* obtained from the original spatial database.

Slot is the second granularity that organizes a set of entities within a layer since

(i) the amount of entities in a layer could be very large and

(ii) a spatial query/operation usually affects only a small set of entities in a layer.

Appropriately dividing entities into slots turns out to be an important task. Usually, entities are organized by their spatial locality. For instance, entities along a railway could be put into one slot; and entities within a certain scope could be put into one slot. It is notable that slots are exclusive, i.e. an entity can not appear in two slots of a layer. An additional benefit brought by a slot is that the update of a layer could be bounded in several affected slots. A *slotVersion* is used to track the change of a slot.

One update operation may modify only one or several entities in a layer, so *layerVersion* and *slotVersion* of the layer are too coarse for tracking such change, e.g. it is not worthy the reload the entire layer when only one entity in a layer has been changed in spatial database. Hence Geo-Union assigns a versionNumber

to every entity in a layer. An entity's versionNumber increments when it has been updated. Meanwhile, a layer keeps an *entityVersion* attribute which is the maximum versionNumber of its entities. When the *entityVersion* of a cached layer is less than that in spatial database, the cached layer could be updated by loading the changed entities via comparing the versionNumber of each cached entities with that in spatial database. The strategy minimizes the entities to be transferred through network at the expense of a small overhead for exchanging entities versionNumber.

3.4.3 Updating and Pre-fetching Spatial Cache

In order to keep spatial cache up to date efficiently, three kinds of version information are used, namely *layerVersion*, *slotVersion*, *entityVersion*. When accessing entities in a layer, we can determine whether spatial data in cache is valid or not by comparing version information in cache with that in spatial database according to the following rules in sequence:

- When *layerVersion* of a layer in cache is smaller than that in spatial database, the entire layer in spatial cache is invalid and need to be reloaded from spatial database.
- When *layerVerion* remains the same but *slotVersion* of a layer in cache is smaller than that in spatial database, some slots in the layer are invalid and needs to be reloaded.
- When *layerVersion* and *slotVersion* remain the same but *entityVersion* of a layer in cache is smaller than that in spatial database, some of entities of a layer are invalid.
- When all the three version numbers remain the same, the spatial cache is said valid and no cache update operation is needed.

In addition to update strategies, cache space management and pre-fetching are also important for improving data access efficiency. It is well known that spatial data access exhibits both temporal locality (i.e. users are highly likely revisiting recently visited spatial entities) and spatial locality (i.e. users are likely visiting spatial entities near their recently visited spatial entities). Hence temporal locality based cache space management (e.g. LRU) and spatial-locality based pre-fetch mechanisms in spatial cache turn out to be useful in implementing Geo-Union spatial cache.

3.4.4 Object Diagram of Spatial Cache

As shown in Figure 8, Geo-Union spatial cache has six types of objects: CGuCacheMgr, CGuCache, CGuSlots, CGuSlot, CGuEntries and CGuOpList.

Every spatial cache has exactly one CGuCacheMgr object that manages all objects stored in the cache. A cached layer is managed as an instance of CGuCache, which is identified by the corresponding layerID. Cached spatial entities are managed as an instance of CGuEntries, which provide random access to spatial entities via entityID. The slotID of an entity indicates the status of an entity:

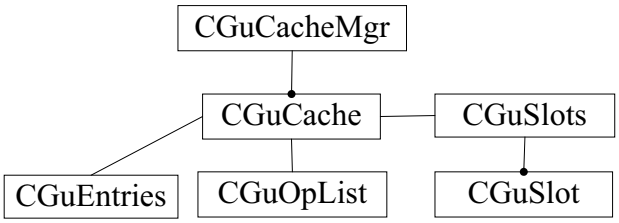


Fig. 8. Object diagram of spatial cache

(i) the entity belongs to a slot ($slotID > 0$);

(ii) the entity is not yet assigned to any layer ($slotID = 0$); or

(iii) the entity is not yet in cache ($slotID < 0$).

CGuOpList is used for transaction management. It keeps a list of update operations made but not committed by the client. The end-user may choose either committing these operations to spatial database or canceling these operations on the client side without affecting the spatial database.

4 CONCLUSIONS

Geo-Union system has been used in many real projects in China, such as Pipe Network Integration Information System in Wafangdian City, Electric Power Information System in Yantan City, Water Environment Information System in Daqing City and Fire Emergency Information System in Haikou City. At present, these systems are running in stable status and handle large numbers of concurrent users requests without failing.

While Geo-Union has explored critical technologies in building WebGIS and provided a customizable WebGIS platform, there are still many issues to be addressed for improving its performance in real world applications:

- The growth of the web will surely bring more spatial information available for all web users. Usually, the spatial information is encoded in various formats with different quality; hence interoperability issue should be addressed appropriately. Although considerable progress has been achieved through standardization in the past years (e.g. by OpenGIS consortium), this issue will continue to be critical in the future.
- As long as GIS data is distributed on the Web, copyright protection and data access security/privacy should be considered, especially for spatial data in vector format.
- The performance of WebGIS is critical to users' satisfaction adoption; for example, most web users will give up if a map query will cost more than one minute.

High performance and scalable WebGIS platforms that could serve huge amount of web users at the same time are in great need. From literature, multi-agent systems [11], peer-to-peer system, and service-oriented architecture are potential driving technologies. Existing works [12, 13] have investigated how to build WebGIS using agent technologies. Other technologies should be investigated as well. Separation of data service and computing service in WebGIS applications and platforms could be a good strategy for collaborative computation in the internet.

Acknowledgement

This work is supported by the National Grand Fundamental Research 973 Program of China under Grant No. 2006CB701306; the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20020001015; the National Science Foundation of China under Grant No. 60203002; the National High Technology Development 863 Program under Grant No. 2002AA134030; the Doctoral Foundation of Xinjiang Bingtuan.

REFERENCES

[1] Longley, P. A.—Goodchild, M. F.—Maguire, D. J.—Rhind, D. W. (Editors): Geographical Information Systems. Volume 1, Principles and Technical Issues, Second Edition. John Wiley & Sons, Inc, 1999.

[2] Longley, P. A.—Goodchild, M. F.—Maguire, D. J.—Rhind, D. W. (Editors): Geographical Information Systems. Volume 2, Management Issues and Applications, Second Edition. John Wiley & Sons, Inc, 1999.

[3] Li, Z. et al.: Geographic Information System in the Internet Age. Acta Geodaetica et Cartographica Sinica, Vol. 27, 1998, No. 1, pp. 9–15, in Chinese.

[4] Szyperski, C.: Component Software: Beyond Object-Oriented Programming. MA: Addison-Wesley Press, 1998.

[5] Bin, L.: A Component Perspective on Geographic Information Services. Cartography and Geographic Information Science, Vol. 27, 2001, No. 1, pp. 75–86.

[6] Luo, Y. et al.: The Components Design for WebGIS. Chinese Journal of Image and Graphics, Vol. 4, 1999, No. A (suppl), pp. 79–84, in Chinese.

[7] Li, M.: Research and Implementation of the Componentization of Distributed WebGIS. Master Dissertation. Beijing: Peking University, 2000, in Chinese.

[8] Wu, J.: A Study on Spatial Data Management in Component-Based Distributed WebGIS. Master Dissertation. Beijing: Peking University, 2000, in Chinese.

[9] Ding, L. et al.: Survey on Distributed Spatial Information Management. Chinese Journal of Image and Graphics, Vol. 6, 2001, No. 11, pp. 1101–1106, in Chinese.

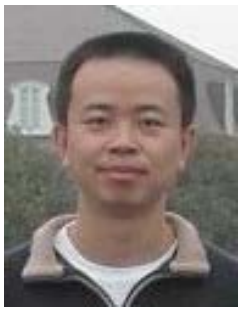
[10] Shengri, C.: Key Issues on ORDB-Based Component GIS. Ph. D. Dissertation. Beijing: Peking University, 1999, in Chinese.

[11] Nwana, H. S.: Software Agent: An Overview. Knowledge Engineering Review, Vol. 11, 1996, No. 3, pp. 205–244.

[12] Yingwei, L. et al.: The Research on Geo-Agents. Journal of Computer Research and Development, Vol. 37, 2000, No. 12, pp. 1504–1512, in Chinese.

[13] Tsou, M. H.—Buttenfield, B. P.: An Agent-Based Communication Mechanism For Distributing Geographic Information Services On The Internet. GIScience 2000: First International Conference on Geographic Information Science, October 28–31, Savannah, Georgia.

Yingwei Luo is now an associate professor in the Department of computer science and technology, Peking University. His research interests include geographic information systems, software agent and mobile computing.



Li Ding is now a Ph. D. candidate in the Department of computer science and electrical engineering, University of Maryland, Baltimore County, USA. His research interests include semantic web and ontologies.

Xiaolin Wang is now an associate professor in the Department of computer science and technology, Peking University. His research interests include geographic information systems and virtual server.

Wenjun Wang is now a postdoctoral researcher in the Department of computer science and technology, Peking University. His research interests include geographic information systems and e-government.

Zhuoqun Xu is now a full professor in the Department of computer science and technology, Peking University. His research interests include geographic information system, artificial intelligence and parallel computing.