DESIGN SPATIAL CACHE FOR WEBGIS

LUO Yingwei *, WANG Xiaolin and XU Zhuoqun

Dept. of Computer Science and Technology, Peking University, Beijing, P.R.China, 100871 – lyw@pku.edu.cn

KEY WORDS: Component, WebGIS, Geo-Union, Spatial Cache Framework

ABSTRACT:

A spatial cache framework is designed and adopted in a component based WebGIS system Geo-Union to improve its performance in network environment. The spatial cache framework includes three typical cache modes: database cache, network cache and cache server (data proxy server).

1. INTRODUCTION: GEO-UNION

Geo-Union is a GIS platform developed by spatial information lab in dept. of computer science and technology, Peking University. Geo-Union has a multi-level Client/Server architecture, which is implemented by principle of ORDB and component techniques. Geo-Union provides an object-oriented, extensible GIS component library for further GIS application developers. Geo-Union can be used in both stand-alone environment and network environment.

Component model is a primary approach to deepen the functions of WebGIS (Bin Li, 2001). To make the system more clear, Geo-Union can be divided into four layers: Geo-Union application layer, Geo-Union component layer, Geo-Union service layer and Geo-Union storage layer, where service layer has different units to provide both client services and server services. Figure1 shows the architecture (Dept. of Computer Science and Technology, Peking University, 2002). Hierarchical spatial component object model can distribute GIS functions in network reasonably and make the system reusable, as well as provide efficiency approach for further development and integration with other systems.

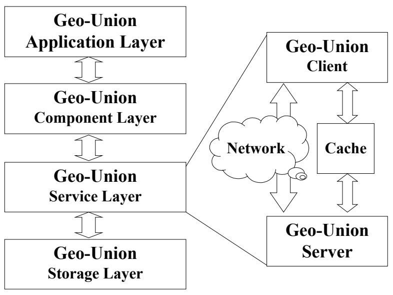

Figure 1 The Architecture of Geo-Union

(1) Storage layer is the ground of Geo-Union. Storage layer is responsible for storage and management of both spatial data and non-spatial data based on ORDB. The main problems solved at this layer are how to represent and store spatial data, and how to maintain relationships among spatial data.

(2) Service layer is in charge of spatial data access and process, which can be divided into another two parts: Geo-Union client provides data access and process services to component layer, and Geo-Union server provides data access and process services to Geo-Union client through interacting with storage layer. Geo-Union server can manage different spatial data resources, and also can reply to different spatial data requests from different clients. Geo-Union client and Geo-Union server are two independent parts but have close relationship with each other. Geo-Union server provides the services of data access, spatial index, basic spatial query, transaction process, data share and so on. Geo-Union client provides different GIS tools and further development functions to component layer based on the services from Geo-Union server. Cache is an important unit of Geo-Union client, which is imported to reduce network load and improve response speed of the system. Using Geo-Union client, we can develop a simulation server, which can reduce network load and improve response speed through its cache.

(3) Component layer provides a rich set of services (components) to develop domain-oriented GIS application systems for further developers. Component layer provides interface of GIS functions to users, but the implementation details are completed in service layer. Component layer exists as a component library, and servers as a bridge between users and service layer. Component layer provides function-explicit and reusable interface components for users according to the functions of service layer.

(4) The work in application layer is to exploit application systems for different special domains by assembling and integrating Geo-Union components. These application systems can be running both in desktop and network environment.

2. SPATIAL CACHE FOR GEO-UNION

Cache is an important technique for improving the performance of system. In Geo-Union, there are two aspects affecting the efficiency of data access: one is the access to database, especially when storing spatial data in ORDB; the other is the transmission of spatial data in network. We take different spatial cache modes to solve these two problems in Geo-Union.

2.1 Spatial Cache Framework

Figure 2 is the spatial cache framework in Geo-Union. The spatial cache framework includes three typical cache modes: database cache, network cache and cache server (data proxy server).

* Corresponding author: LUO Yingwei, lyw@pku.edu.cn.

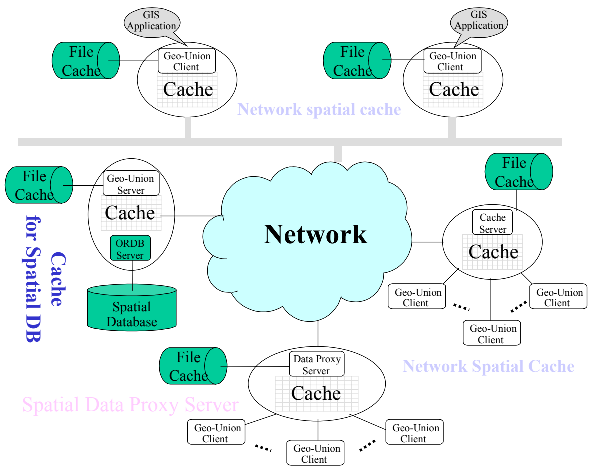

Figure 2 Spatial Cache Framework

(1) Cache for spatial database. Geo-Union server is the bridge between Geo-Union client and ORDB, so cache for spatial database is maintained by Geo-Union server. Cache is stored in both local file and memory. Because Geo-Union server can manage different spatial data resources, and also can reply different spatial data requests from different clients, so cache for spatial database is a global cache.

(2) Network spatial cache. The bandwidth of different users in network is different, but public users of Internet always have low bandwidth. So when people use WebGIS applications, it is a lethal delay when spatial data is transferred from Geo-Union server. In Geo-Union client, we adopt a two-level spatial cache mode to relief transmission bottleneck of the network:

The first network cache is used to help a single client access remote data: building a spatial cache in local place. This kind of cache is a partial cache, which is also a popular method in today's browser.

The second network cache is used to help many clients in a LAN access remote data: building a common spatial cache for a LAN (cache server). Once a local client in the LAN accessed some spatial data, other clients can reuse the spatial data in cache server. Cache server is still a partial cache.

Cache server can realize massive spatial data cache by means of the shared resources, which will speed up the hit rate of cache greatly, so as to improve efficiency of all local clients, and save resource of all local clients. Cache server solves the speed conflict between local disk data access and remote data access, and the speed conflict between the high-speed LAN and WAN with narrow bandwidth.

(3) Spatial data proxy server. Because the distribution of users in Internet is not well-proportioned, so different Geo-Union servers are unbalanced. Some Geo-Union server and its communication may overload. Aiming to this problem, we design spatial data proxy server for those Geo-Union servers to improve performance.

Spatial data proxy server is an initiative cache server. Overloaded Geo-Union server selects a suitable Geo-Union client and builds a spatial data proxy server there to response a special group of users.

In Geo-Union, spatial data proxy server serves as special server in Internet to provide spatial data access services for public users. The structure and implementation of spatial data proxy server is same as cache server, but they play a different role in Geo-Union.

Cache server is private of a LAN, and clients in the LAN have to get an authorization before accessing to cache server. Spatial data in cache server is changing with different requests of clients in the LAN.

Spatial data proxy server is public to all clients. Spatial data proxy server may serve as a peer of Geo-Union server. Spatial data proxy server can be built anywhere in Internet if needed. If spatial data in a spatial database is unchanged for a long time, spatial data proxy server can cache all spatial data of that spatial database.

Building spatial data proxy servers properly in Internet will make Geo-Union applications more effective.

2.2 Organization of Spatial Cache

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

When creating a layer in spatial database, a GUID (Global Universal Identification) is generated to identify the layer, which is named as *layerID*. When reading a layer into cache, the system will allocate a separate space for the layer according to its *layerID*. Whether a layer is valid in cache is determined by the *layerVersion* of the layer both in cache and in spatial database.

Entities in a layer are always separated into different slots according to a certain rule. When reading a layer into cache, we do not read the whole layer, but read some slots of the layer. Slot brings two benefits: almost all spatial queries do not need a whole layer but only a certain scope in the layer, so when the layer is massive, reading relative slots can satisfy the requirement and will reduce network load greatly; less data will exhaust less computing resource and storage resource. The rules to organize slot are various. We can organize slots in a layer according to a correlativity of geographical location or a neighborhood geographical location. A correlativity of geographical location means we can put entities along a railway into a slot, and a neighborhood geographical location means we will put entities in a certain spatial scope into a slot. Every entity in a layer belongs to a slot. When entities in any slots changed, the *slotVersion* of the layer will change too.

An update operation may modify only one or several entities in a layer, so *layerVersion* and *slotVersion* of the layer cannot reflect the latest modification of entities. We set a *versionNumber* for every entity, and when a entity changes, its *versionNumber* changes too. The *entityVersion* of a layer is the max *versionNumber* in the layer. When the *versionNumber* of a layer in cache is less than that in spatial database, some entities in cache is invalid and those entities that have larger *versionNumber* should be reloaded from spatial database into cache.

2.3 Refresh and Pro-load Spatial Cache

Refreshment of spatial cache can be done online or offline. Online refreshment means updating spatial data in cache at the same time as updating spatial entities in a layer in spatial database. Offline refreshment means updating invalid spatial data of a layer in cache if their versions are invalid when accessing to them.

Information in spatial cache includes *layerVersion*, *slotVersion*, *entityVersion* and the corresponding relations between all entities and slots of a layer. When accessing to entities in a layer, we can determine whether spatial data in cache is valid or not through comparing version information in cache and spatial database.

(1) If *layerVersion* of a layer in cache is smaller than that in spatial database, it means that all entities of the layer are changed, and the whole layer should be refreshed in cache. Otherwise,

(2) If *slotVersion* of a layer in cache is smaller than that in spatial database, it means that all slots of the layer are rearranged, and the corresponding relations between all entities and slots of the layer should be refreshed in cache. Otherwise,

(3) If *entityVersion* of a layer in cache is smaller than that in spatial database, it means that some entities of the layer are changed, and those entities should be refreshed in cache.

(4) If above three conditions are all equal to those in spatial database, it means spatial data of a layer in cache is same as that in spatial database, and no refreshment is required.

When accessing to spatial data, if we can pre-load some spatial data into cache, it will make spatial cache more effective. But how to predict what kind of spatial data will be accessed to?

There are two rules of accessing to memory: there is every probability of accessing to just being accessed memory, and there is every probability of accessing to neighbors of just being accessed memory. There are same rules in accessing to spatial data: there is every probability of accessing to just being accessed spatial data, and there is every probability of accessing to neighbors of just being accessed spatial entities. According to those rules, we can arrange slots by spatial scope, and a slot is an accessing unit of spatial data. When the network is idle, we can pre-load some neighbors of spatial data in cache from spatial database.

3. CONCLUSIONS

Geo-Union has finished a preliminary component-based model for distributed WebGIS, and has got into use in many fields with sound effects. Although spatial cache and other techniques are adopted in Geo-Union to improve its performance, a lot of works still wait us ahead to make Geo-Union more practicable and effective:

(1) Dynamic load balancing policy. In Geo-Union, most works are completed at client side, and server is only responsible for data access and simple data query. Therefore it is not well balanced between client and server. Especially there will exists a lot of transmission for massive data between client and server. Although, spatial index and spatial cache techniques can improve the performance to a certain extent, we still want to take full advantage of the computing capability of GIS server, so as to lighten load at client side and decrease the transmission of redundant data in network. This needs a more reasonable component design for the system.

(2) System concurrency. WebGIS is open to millions of users. How to ensure the correctness, validity, stability and scalability of Geo-Union to satisfy users' requests is another key problem for practicable WebGIS.

4. ACKNOWLEDGEMENT

Supported by the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312000; 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. 2002AA135330 and No. 2002AA134030; the Beijing Science Foundation under Grant No.4012007.

5. REFERENCES

Li Bin, 2001. A Component Perspective on Geographic Information Services. *Cartography and Geographic Information Science*, 27(1), pp.75-86.

Dept. of Computer Science and Technology, Peking University, 2002. Operation and Component Guide for Geo-Union Enterprise (in Chinese). Technology Material, http://gis.pku.edu.cn. (accessed 20 Oct. 2002).