Abstract

Current GIS development pays more attention to the ability of distribution based on the client-server architecture. In a batch mode distributed GIS, a file server functions passively. Information sharing and integrating are far from concurrent, nor is the database integrity controlled. The dynamic Voronoi data model, equipped with a dedicated spatial server, presents a bright future in overcoming these shortcomings. In a dynamic distributed GIS, the spatial database is maintained in the server machine which provides database services to clients through different views. A view is an image of a partial map. Any access to the database through views must be examined and performed by the spatial server. Should the database is modified by an authorized user, the change could be reflected simultaneously on all relevant views. Spatial searches through the database could be achieved efficiently without additional indexing. As the spatial database gets bigger and more complex, some compact objects based on spatial proximity could be constructed, which brings a hierarchical representation of the Voronoi diagram. The architecture of the dynamic distributed GIS also provides feasible solutions towards large spatial problems with divide-and-conquer and parallel processing approaches.

Résumé

Le développement des SIG actuels prend plus en compte la capacité de distribution basée sur l'architecture client-serveur. Dans un SIG distribué en mode bloc, le serveur fonctionne passivement. Le partage et l'intégration de l'information sont loin d'être concurrents et, l'intégrité des bases de données contrôlée. Le modèle de données Voronoi dynamique équipé avec un serveur spatial dédié offre une voie future de dépassement de ces embuches. Dans un SIG dynamique distribué, la base de données spatiale est maintenue dans le serveur, qui fournit des services de bases de données aux clients à travers différentes vues. Une vue est une image d'une partie de la carte. Tout accès à la base de données à travers les vues doit être traité et réalisé par le serveur spatial. Si la base de données modifiée par un utilisateur autorisé, le changement pourrait être reflété immédiatement dans toutes les vues appropriées. La recherche spatiale dans la base de données pourrait être réalisée efficacement sans indexage additionnel. Quand la base de données devient plus grande et plus complexe, des objets compacts pourraient être construits sur la base de la proximité spatiale, ce qui amène une représentation hiérarchisée du diagramme Voronoi. L'architecture d'un SIG dynamique distribué fournit également des solutions concrètes aux problèmes spatiaux d'ordre supérieur avec les approches de recherche dichotomique et de traitement parallèle.
1. Introduction

In the era of ever more sophisticated computing and communicating technology, people are expecting more from it to facilitate their tasks. There is no exception for users of GIS. As the spatial database is getting bigger and bigger with integrated data, speed and efficiency become more and more matters to deal with in routine jobs for which interactive response from the system is often desired. One solution towards speeding up applications for a large spatial database is by distribution. In a distributed system, not only data, but also functionalities applicable to the data, could be distributed. That means that a client at a local site could work with a relatively small data obtained through centralized services.

In the current main stream of GIS technology, an application could obtain data from a central data bank, government agencies or from other sources. After the project is finished, organizations controlling the data could accept the updated data by authorized users back into the corporate databases. This standalone style of data communications not only delays the development of applications, complicates administration and conversion of data from different sources with different formats, but also precludes integration of different technologies to facilitate and expedite data integration, let alone a real-time exposure to and interactive control of current data at either donor or user site.

Recent advances in newer hardware and generic software technology permit the GIS community thinking about extending or re-engineering some part of existing GIS modules to an open system, on a client-server basis [Frosh 1993, Keene 1993, Dangermond 1993, Blue and Lee 1992, Lemkow 1992, Bannura 1991, Healey et al. 1991]. Discussions and prototyping designs of distributed GIS at the architecture level are also seen in literature [Milne et al. 1993, Abel et al. 1992, Webster 1988]. Concentrated on this new movement of GIS technology are mostly redefining user interfaces to adapt to multi-users at multi-sites with multi-platforms. Spatial, as well as aspatial, data accessibility is the central part of the whole story. Using a graphics user interface, an end-user could compose his query, pass it to a navigator or spatial data server or browser, which takes cares of interpreting user requests into meaningful GIS commands and accesses data from remote sites. If a whole map is not the main concern, some part of it could be selected, previewed, downloaded to application platforms, and worked with in different GIS environments. This approach leads to a significant reduction of time spent on transferring and processing irrelevant data.

2. Shortcomings of Current GIS in Distribution

What has been missing so far in the literature is a somewhat detailed description about controlling concurrent transactions of distributed data and maintaining data integrity. If all transactions do not violate any consistency constraints, the database would be internally consistent. This could be achieved by not allowing any modification of spatial data or by executing each transaction alone, one after another. It is obvious that neither approach can be implemented in any practical system, since it minimizes the system throughput. On the other hand, since most current GISs require a batch process to build topological
relationships among spatial objects, any transactions that lead to altering the spatial relationships would be classified as long-life transactions. This disadvantage would either leave outdated copies of the database to be used in application platforms, or let clients who want to work with latest data wait -- jeopardizing reliability and efficiency of the system. It is therefore not feasible for such a system to be extended to spatial decision supporting systems or spatial simulating systems where on-line transactions are desirable.

3. A Solution -- Dynamic GIS

The shortcomings of current GIS in implementing a distributed system could be overcome by designing a dynamic spatial database. In a dynamic environment, topological relationships of existing spatial objects always preserved, even during the execution of a transaction. Users do not have to worry about making network and rebuilding topology after any modification of the map, as they usually do while using a batch mode GIS. One of the obvious advantages of such a GIS in a distributed system is that any database updating transaction becomes a short-life or on-line one. With a spatial server, users working in other sites could receive the update information and couple it into their applications if necessary. This will be explained in a following section.

A dynamic GIS could be constructed around a dynamic data model. One of such models, one being developed at the Laval university is call the Voronoi dynamic data model. The Voronoi data model [Gold 1992a] lends itself to be a hybrid structure, a way of viewing vector and raster representations of spatial data. The Voronoi model comes from a new definition of “neighbours” in the space [Gold 1992b]. By the Voronoi data model, every spatial object (point or line so far) is assigned a “piece of space” which is closer to this object than to anything else. This associating space is called Voronoi polygon. Two objects are neighbours if they share a Voronoi boundary. In two dimensional space, a graphical representation of the Voronoi model is called a Voronoi diagram. A Voronoi diagram consists of point and line objects is shown in Figure 1, where the thicken lines and dots representing map objects and thin lines depicting Voronoi polygons for each map object. It can be noticed that a line object is actually composed of two “half lines”, each associated with a Voronoi polygon. A computerized representation of the Voronoi data model, the Voronoi data structure, is actually maintained by the Delaunay triangulation, the dual graph of the Voronoi diagram.

The dynamic feature of this data model is enforced by a few “event driven operations” plus an “intelligent pen navigator” [Gold 1993]. In an interactive environment, the map is displayed, and the position of the mouse is captured at any time. Meanwhile, the nearest object to the mouse is found and highlighted. The operator could issue Voronoi macro commands to digitize and edit the map. The point object could be selected and moved to some given location. As the pen (moving point) moves, the shape of its Voronoi polygon changes. A topological event occurs as the moving point separates from an once-neighboured object or includes another object as its new neighbour. The change of the Voronoi polygon of the moving point affects only its immediate neighbours. This makes the
modification of the Voronoi diagram purely local actions. To add a point at a given location, the system first splits a new point and then moves this point to the destination. Deleting a selected point performs a reverse process. The point is first moved and then merged with the nearest point. Operations for adding and deleting lines are analogous to point operations, except that the moving point either drags a new line behind or compress an existing line in front. Since the pen is part of the data structure, it has the intelligence of knowing where it is and the kind of map object with which some topological event is going to happen. The macro commands could be programmed such that the operator is consulted before some specific topological actions be taken. For example, if the pen is near a line, before it gets across it, the operator could decide whether he means this to happen. With the instructions given, the pen therefore can stop where it is, or join with the line, or else continue moving after a new node is created at the intersection. A direct advantage of this option is that it results in a relatively clean map right after the digitizing, saving the cost of repetitive manual editing and topology rebuilding.

4. A Spatial Server for Distribution

A distributed GIS could have a centralized spatial database which serves access requests from local sites through a proper communication channel. Through the network, each site works with a view of a map. A view is some part of the spatial database. It is desirable that any changes to the spatial database could be seen almost concurrently by relevant sites such that these sites can incorporate the new data. It is also desirable that local offices can continue most of their work without having to wait to refresh the entire spatial database. The justification for this situation is that quite often local modifications, for simulations or real changes, have little impact on distant areas. Except when some global operations or information about the whole map is needed, local activities could just suffice. These features are difficult to implement for any batch mode GIS. With the dynamic Voronoi data model, however, it is achievable with the help of a dedicated spatial server.

A spatial server is built around a spatial database and resides in the central site machine (Figure 2). It is a specially softwareed processor running in the central machine, which processes and dispatches messages between the server and client machines. It is active if there is a message in the network and idle otherwise. As a common interface connecting to clients, it preprocesses view generating requests including password and privilege checking, registering the view and book-keeping relevant site and display information, and more importantly, assigning a unique handle to this view. Through the handle, the spatial server obtains control over the spatial database and performs data access tasks on behalf of the client. Once a window displaying the view is created, instead of sending all the index files and then data files to the client, the server searches through the spatial database and sends back the information that satisfies the client’s constraints. The spatial server might then return to idle or act on other views that have issued event messages.

With a dynamic data model, the spatial server is able to preserve data controls at different levels with different features. Sometimes, the flexibility of choosing any type of view should
be left to the end user. For an environmental researcher, for example, he may prefer working with a stable experiment data set and does not want to change his map scenario without his agreement. As a decision maker using environmental data, on the other hand, he may wish to access the most recent data to enhance the validity and reliability of his plan. Furthermore, an environmental statistics reporter may best integrate the view as an OLE (object linking and embedding) object into his document, such that whenever there is some change to the view, a refreshed report is generated automatically. We therefore could have static and dynamic views depending on whether they are downloaded to the local site or simply linked.

A static view is a copy of a specific part of a map. The most challenge task for a spatial server to generate this view is how to extract the content of the view and their spatial relationships out of the spatial database. Once the view is wired to the client machine, the owner of the view has full rights to alter it. The spatial server could maintain a loose linkage to this view to suggest to its user the current status of the region correspondent to the view. The user decides whether he wants an updated copy of the view. When a view is statically embedded in one client site, no other client has direct access to the data in this view. They can, however, obtain information in this view via the spatial server.

A dynamic view is a linked object. The data of this object is not stored in local site. What an end user see is an image of the partial spatial database confined by the view specification. Any messages for the operation of the view are captured and examined by the spatial server, and possibly executed by the central GIS or other local site that holds the spatial data and distributed with GIS functions. If an owner of a linked view has only limited authority over the database, the challenging task for the spatial server is to obtain the knowledge on whether the operation request from the user is permitted.

How to extract and return the contents of a static view and its topology in the central spatial database, as well as to confine any illegal access to the database from linked views is an open question. Some preliminary approaches which creates different “spatial windows” are discussed by the authors [Yang and Gold 1993]. An example of one spatial window is shown in Figure 3a where physical boundaries for the window (the rectangle in the central part of the map) are temporarily generated. This spatial window corresponds to a view which can be operated by a client. It is illustrated in Figure 3b that adding two polylines inside the view could be incorporated into the central database. The topology updating involved is strictly confined within the view. The spatial relationships outside the spatial window are kept untouched.

5. Spatial Indexing and Searches

Searching through the spatial database could be enhanced by designing an additional spatial indexing system over the data structure and implementing efficient search algorithms. Search structures for Delaunay triangulated data (the dual of the Voronoi diagram) have been extensively discussed in the area of computational geometry. For an incremental
construction approach with random input order, it is accepted that an irregular search tree
can be built while the triangulation is processing [Watson 1994]. This paper will not attempt
to survey all the indexing structures applicable to the Voronoi diagram. It is our experience
that for an interactive and event driven system, a simple walk through the triangles can
locate the nearest neighbour in a hardly noticeable time. An algorithm for the range search
can also be developed without additional data structures. The property of the Delaunay
triangulation suggests that given a viewpoint, a systematic scan can be performed through
the whole triangulation. The result of this scan is a partial ordering of the space. The range
search could be constrained that only the interested area be scanned and that relevant spatial
objects be reported only once.

The Voronoi diagram contains sufficient topology among spatial objects from which any
spatial queries can be answered at run time. This information rich structure comes with an
extra overhead of storage space, representation complexity and computation time because
each point and line segment needs to be fully structured in the diagram. For a heavily
populated map composed of complex curves, a flat organization of spatial objects could
result in an occupation of large memory and awkward performance. This shortcoming has
motivated research on a compact Voronoi diagram for realistic GIS arcs and polygons
[McAllister and Snoeying 1994].

In order to appreciate the advantage of a compact Voronoi diagram, let’s examine a simple
case (Figure 4) where the clustering of map objects produces a spatial proximity of four
rectangles. No matter how complex the spatial representation inside each rectangle, their
inter-relationships become regular and simple. In brief, the Voronoi boundaries (thin lines)
in map DIAG3 indicates that rectangle B, C and D are neighbours of rectangle A, A and D
are neighbours of B, A and D are neighbours of C, A, B, and C are neighbours of D. These
relationships is simplified as map DIAG3 1 where four condensed points A, B, C and D
represents four rectangles respectively. DIAG3 1 implies a hierarchical Voronoi indexing
structure. Should a condensed object be found in a spatial query, its lower level diagram is
searched until the leaf level is reached. The hierarchical structure fits well with the view
architecture in that a condensed point could be expanded as a partial view of the map.

6. Applications of a Distributed Dynamic GIS

The Voronoi dynamic data model, plus a dedicated spatial server, becomes an ideal choice
for many applications using spatially referenced data. With the spatial database being
managed by the server and the central machine, end users do not need to worry about the
locations and versions of a map and, even more significantly, its integrity. They can
therefore concentrate on developing their applications at an accelerated pace. The short
transaction over modification of the spatial database provided by the dynamic system
guarantees that multiusers have an identical view of a map. This ability helps a task team in
an enterprise working concurrently to resolve a problem involving spatial data, an
important feature seen in business applications with a relational database but not in a spatial
one. Furthermore, the multiview access, equipped with proper controls, makes divide-and-conquer solutions to and parallel processing of a large spatial problem into an easy task. Using the following artificial example (Figure 5) it is attempted to illustrate how a multi-intelligent digitizing process could be paralleled. The “intelligent” modifier is used here to mean that the topology is constructed during digitizing.

Figure 5a shows that as an initial process, the spatial server creates four empty views which are controlled by hard boundary spatial windows. Doing this builds relationships among four views and ensures that spatial operations inside each view are managed by the integrated spatial database. Once the clients have received the handles to the views, four view operators can start digitizing concurrently. The spatial server monitors the four views at the central site to make sure that no actions can cross the spatial windows. It is clear from Figure 5b that at any time, not only the topology inside each view, but also in the whole map is complete and that the inter-relationships among four views are unchanged. After all the operators have finished digitizing their part of the map, one authorized operator obtains the entire view of the map. He or she then removes the hard boundaries of spatial windows and digitizes remaining lines around individual views (Figure 5c). It is worth mentioning that if one operator finishes digitizing with his or her view, it is not necessary for this site to wait for others to finish. Other operations can proceed in this view.

7. Conclusions

Distributed spatial databases are getting more and more attention due to the demand for data sharing and data integration from applications such as multi-expertise spatial decision support systems. A spatial server must be included in the design of the system architecture for distribution. In a dynamic distributed GIS, the spatial server is not just a passive file server for selecting and checking out a map from a centralized spatial database, as it is in a batch GIS. It is an active view manager and coordinator which controls multi accesses and concurrent view updating. The event driven dynamic Voronoi data model provides a fast searching structure without additional spatial indexing. The problem with large complex maps can be overcome with the introduction of compact objects along with a hierarchical indexing structure. The architecture of the dynamic distributed GIS makes team co-operation for a project feasible and comfortable. It also provides easy solutions towards large spatial problems using divide-and-conquer approach and parallisation.

References


Bannura, Ramzi K. [1991]: A Step beyond Desktop Mapping: An Integration of Personal


Watson, Dave [1994]: Natural Neighbour Tree Structures. From e-mail communication within the group of Voronoi and Computational Geometry Applications <VORAPP-L@LAVALVM.l.BITNET>

Figure 1. A Voronoi diagram with line and point objects

Figure 2. Client-Server Architecture
Figure 3a. A spatial window confining a view

Figure 3b. The topology outside the spatial window untouched after the change of the view

Figure 4. A simple hierarchical representation of a Voronoi diagram. DIAG3 containing four compact objects is simplified by DIAG3 1

Figure 5a. Initiation of four related views

Figure 5b. Parallel digitizing

Figure 5c. A merged map from parallel digitizing