00-UF

World Multiconference on Systemics, Cybernetics and Informatics

July 23-26, 2000
Orlando, Florida, USA

PROCEEDINGS

Volume II
Information Systems Development

EDITORS
Belkis Sanchez
Nader Nada
Awais Rashid
Timothy Arndt
Maria Sanchez

Organized by IIIS

International Institute of Informatics and Systemics

Member of the International Federation of Systems Research

IFSR

Co-organized by IEEE Computer Society (Chapter: Venezuela)

A USER-FRIENDLY MULTIMEDIA SYSTEM FOR QUERYING AND VISUALIZING OF GEOGRAPHIC DATA

Shu-Ching Chen, Naphtali Rishe, Xinran Wang, and Mark Allen Weiss

High-Performance Database Research Center
School of Computer Science
Florida International University, Miami, FL 33199, U.S.A.
{chens, rishen, wang01, weiss}@cs.fiu.edu

ABSTRACT

In this paper, we present a user-friendly multimedia system called TerraFly, which is used for Geographical Information System (GIS) applications. This system makes a large number of remotely sensed images available to the general public. TerraFly can let the user interact with the system and explore spatial data of their choosing without advanced knowledge of these data. A client/server architecture is used in the TerraFly system. This system implements a high-performance semantic multimedia spatial database for the storage and retrieval of all the data used by the system and allows users to fly over and manipulate the retrieved data. The information server is responsible for answering range queries and nearest neighbor queries. This server is implemented using a multithread environment with thread pools. The TerraFly client is implemented using Java. It has a communication interface to synchronize data retrieved by both the database server and information server. Detailed examples are presented to show the ease of use and the power of the TerraFly systems.

Keywords: TerraFly, GIS, Databases.

1. INTRODUCTION

The availability and usage of remotely sensed data has increased dramatically in recently years. Technological advances in computer science and remote data sensing make tremendous amounts of spatial information available [2]. Along with this increased availability comes the complication of extraction and storage of such data. There are a large number of different formats of remote sensed data such as Landsat, SPOT, AVHRR, etc. While spatial data has become more readily available nowadays, the variety of formats limits our ability to use the data. Different formats need different manipulation to retrieve and use such data. Modern GIS software should handle the integration of such data from various sources. A number of GIS software tools, such as Arc/Info, Imagine, and ENVI, are available to manipulate and view different spatial data. But these tools all have serious drawbacks: (1) They may suffer from inefficiency of entry, storage, and retrieval of spatial data; (2) They need experts who should have both experience in the use of GIS software and thorough knowledge of the spatial data they are dealing with; (3) They need some special hardware requirements, which either are expensive or limit the usage of such software – for example, ARC/INFO needs a number of UNIX-based workstations installed with complete GIS software; (4) The software applications are expensive. To address all these issues, the High Performance Database Research Center (HPDRC) [9] has developed a user-friendly multimedia system called Terrafly. The database system of Terrafly efficiently stores and retrieves multi-dimensional spaces such as image data as well as alphanumeric data. One goal of the Terrafly project is to make the large number of remotely sensed images available to the general public. This means that the Terrafly system does not constrain which computer systems users have and, additionally, Terrafly can let the user interact with the system and explore spatial data of their interest without advanced knowledge of these data. So Terrafly is designed based on a thin-client/fat-server architecture. As long as the users' computer systems satisfy some minimal requirements, Terrafly runs very well across different platforms. A user-friendly GUI is well-designed so that it does not require users to have advanced knowledge data, and the servers handle all complexities.

Terrafly is a multi-user real-time spatial data tool that allows users to view, query, and ana-

This research was supported in part by NASA (under grants NAGW-4080, NAG-5095, NAS5-97222, and NAG5-6830), NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409), ARO (DAAH04-96-1-0049 and DAAH04-96-1-0278), AFRL (F30602-98-C-0037), BMDO (F49620-98-1-0130 and DAAH04-0024) DoI (CA-5280-4-9044), and State of Florida.

Mark Allen Weiss was supported in part by NSF (EIA-9906600).

lyze remotely sensed images. It uses a variety of remote sensed data, some of which include several spectral sensors. Users can fly over the color composite image of their choice by selecting any of these sensor combinations and the system also gives the user control over the intensity of the colors in the imagery. A Geographical Names Information System (GNIS) is also included so that (1) the user can select the name of locale and be instantly transported to that area on the remotely sensed image; (2) users can do range queries to get more information of the area around the locale they selected; and (3) users can search the nearest neighbors. In addition, users can zoom into an area virtually by switching to a high-resolution data set.

The Terrafly system implements a high performance semantic multimedia spatial database for the storage and retrieval of all data used by the system and allows users to fly over and manipulate the retrieved data. This database server is capable of storing a huge amount of binary data such as images and other multimedia data distributed over TCP/IP networks. The system includes a multithreaded information server based on a GNIS to provide functionality of range queries and nearest neighbor queries. To achieve this functionality, we implemented an R-tree representing the spatial data structure and its query algorithms. The Terrafly client is a data-less thin client working across the Internet.

In this paper, we discuss Terrafly's client/server architecture and its system components. We also discuss multithreading and a thread pool model since they are very important features of Terrafly. We briefly describe the R-tree and other data structures and study a range query algorithm and a nearest neighbor query algorithm used in Terrafly.

This paper is organized as follows. Section 2 introduces the system architecture and its components of the Terrafly multimedia system. In Section 3 we present the Terrafly information server and discuss R-tree and query algorithms. Conclusions are presented in Section 4.

2. SYSTEM ARCHITECTURE AND COMPONENTS OF TERRAFLY

Terrafly uses client/server computing. The Terrafly client is a Java-based web application, and it is a data-less graphical user interface (GUI). The servers provide service: namely, they retrieve data requested by the clients and answer queries for the clients. Together, the client/server

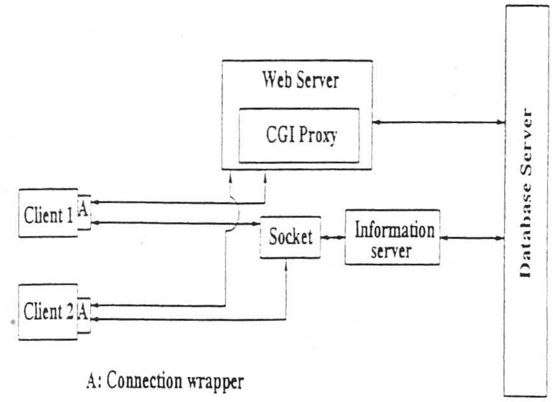

Figure 1: TerraFly system architecture.

architecture forms a complete computing system with a district division of responsibility. Terrafly is based on the three-tier client/server model. This model is more advanced and flexible than the traditional two-tier model in that (1) the separation of the application logic from client and server gives a new level of application logic processes. This new level of logic processes becomes more robust because these processes operate independently of clients and servers, and (2) this new application logic process make system integration and evolution more easier and stable. The Terrafly system architecture is shown in Figure 1.

Java clients use a connection wrapper to synchronize data transferring. The clients send requests to either a web server or information server, and receive data (image data and textual data) from different servers. When the web server receives a request from a client, the proxy server will parse the request, retrieve image and textual data related to the fly-over request from the database server, and send the data back to the client. When the information server receives a request from a client, it will search the R-tree, retrieve information related to range queries and nearest neighbor queries from the database server, and send the answers to the client.

2.1. Java Client

The Terrafly client is written using Java to achieve platform independence. A snapshot of the client is shown is Figure 2. This snapshot represents a fly-over trace (the white dotted line) from NE (NorthEast) to SW (SouthWest). The main features of the client include:

1. Capability to fly over the Landsat TM data, Digital Orthophoto Quad (DOQ) at different directions by positioning the mouse

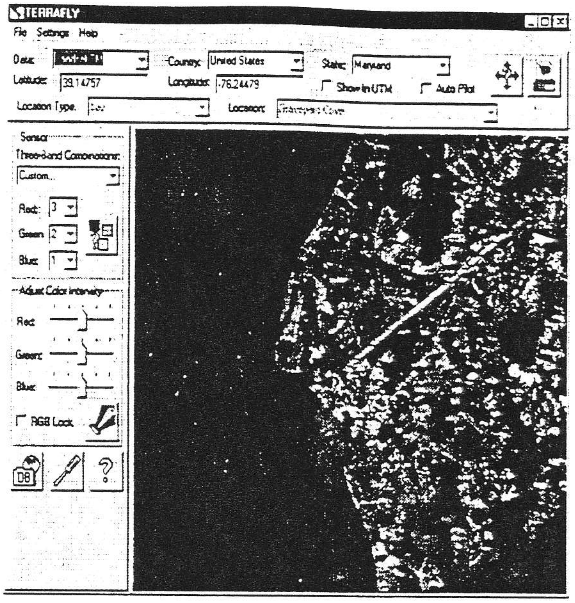

Figure 2: The white dotted line represents the fly over trace from NE to SW.

within the image.

2. Customized three-band (sensor) combination: users can select some predefined and useful three-sensor combinations to view false color images from a drop down menu. When users select a new combination, a different set of images/bands is retrieved from the database and the selected false color image is computed and displayed within a window.

3. Advanced three-band color composite: this application allows scientific users to enter any three-band combinations (RGB) that the user is interested in studying or analyzing.

4. RGB intensity control: this option allows the users to increase or reduce the intensity of any of the bands that represent the colors.

5. Capability to issue range queries and nearest neighbor queries.

6. Capability to obtain feature extraction of the image.

7. Capability to display online information (latitude, longitude, regions, etc) of the images that users are viewing.

2.2. Database Server and Proxy Server

The Terrafly database system is a multimedia spatial database system built by our group using the Semantic Object-Oriented Database Management System (Sem-ODB) based on Semantic Binary Model [7]. In the Semantic Binary Model, information is represented by logical association (relations) between pairs of objects and by the classification of objects into categories. The Semantic Binary Model is a natural and convenient way of specifying the logical structure of information and defining the concepts of an application's world. Unlike the traditional database systems which consist only of alphanumeric data, the Sem-ODB not only has alphanumeric data, but also has data that cover multidimensional spaces such as image data (maps). Currently, the database contains semantic/textual, spatial/ remote sensed (Landsat) and digital data including Digital Orthophoto Quad (DOQ) (Arial photograph) data. The Sem-ODB provides very efficient data storage and manipulation.

Terrafly uses a proxy server to relay data requested by clients to the database server, and to transmit the data retrieved by Sem-ODB back to clients. This proxy needs to use two different protocols, one to interface with clients, and the other one to interface with Sem-ODB server. We use Common Gateway Interface (CGI) as the first protocol and Sem-ODB API as the second protocol. CGI is a well-established method for web-based client applications to communicate with a remote server. The web server can handle requests from multiple clients simultaneously. It can spin off as many CGI processes as needed, and will perform the mapping of the communication between the client and its corresponding process. Overall, the use of CGI has simplified the coding of both the server process and the client process.

Upon receiving a request from a Terrafly client, the proxy server must decode the request and retrieve approximate data from the database server. The proxy process accomplishes this by using Sem-ODB API. As documented in the programmer's reference [10], all that the proxy server needs to do is to find where the database server is located. If the database is to be opened locally (i.e., the proxy process will also act as the database manager), it only needs to specify the complete path where the database is located. If there is a separate database manager, the proxy server needs to specify the complete IP address and port number to reach this database manager. After this database connection is established, all queries can be performed using APIs provided by Sem-ODB.

3. TERRAFLY INFORMATION SERVER

Terrafly provides an integrated view of spatial and associated data along with both the capability to display and manipulate spatial images. Also, being a GIS system, Terrafly provides the capabilities to issue range queries and nearest neighbor queries to satisfy particular interests of scientists and public users. Range queries are to find spatial objects in a specific area around a location specified by users. A sample query is "Find all rental car companies around Miami International Airport within six miles." Nearest neighbor queries are used to find the nearest spatial object to the object the user specifies; for instance, "Find a car rental company that is nearest to Miami International Airport."

The information server is a multithreaded application. It receives requests from clients through a UNIX socket. The configuration of the information server is shown in Figure 1. When the Terrafly system is booted up initially, the information server reads information from a GNIS database to extract essential information for spatial objects. Based on the information such as latitude and longitude, an R-tree is created. The information server answers queries by searching this R-tree.

3.1. Multithreaded Information Server Design

The Terrafly Information Server is implemented in a multithreaded environment (POSIX threads in Solaris or Linux). Multithreading is a technique that allows one program to do multiple tasks concurrently. Although the basic concepts and research of multithreading programming have existed for several decades, the emergence of this concept in industry as an accepted standardized programming technique started in 1990s. The greatest push is from the emergence of shared memory symmetric multiprocessors (SMP). Multithreading provides exactly the right programming paradigm to obtain vastly greater performance by making the maximal use of these new machines. Nowadays, the ever-increasing clock speed of the CPUs allows us to obtain better and better performance from processors. However, the speeds of bus, memory, and peripheral devices in computer systems limit the overall performance, and it is exactly where the SMP comes into play. That is, the SMP allows us to increase our overall system performance. A thread is a stream of control that executes its instructions independently, and allows a multithreaded process to perform numerous tasks concurrently. Threads are program states that get scheduled on a CPU, and they share the same process structure and most of the operating system states: the address space, file descriptors, etc.

The Terrafly information server is a classic software application suitable to use multithreads – a server needs to handle numerous overlapping requests simultaneously. Terrafly clients send large amount of requests that require the server to do some I/O, process the results, and return the answers. Every time, upon receiving a request, the server uses the shared R-tree, rather than replicates one as a process will do. Thus, we have exploited threads to the full extent resulting in improved performance. To further improve performance, we design the information server based on thread pool model [6]. The main thread of the server creates a fixed number of worker threads up front during the time of server startup, and all these worker threads survive for the duration of the program. When the main thread receives a new request, it places it on a job queue. Worker threads receive requests from the queue and process them. When a worker thread completes a request, it removes another request from the queue.

Compared to the creating of threads on demand (as in a boss/worker model [5]), the thread pool technique improves performance significantly. We can reuse the idle threads to handle new requests. Each time the main thread receives a new request, the server program uses one thread that is ready waiting to process the work. In a boss/worker model, the server would create a new thread to handle a new request. After a worker finishes the request, the server destroys the thread. From the client's point of view, the request's processing time (latency) is shorter without the time of creating a thread in the server side. Figure 3 shows the structure of the threads pool in Terrafly.

3.2. R-trees

R-trees [4] were proposed as a natural extension of B-trees with index records in its leaf nodes containing pointers to data objects. R-trees are designed to organize a collection of arbitrary spatial objects by representing them as d-dimensional rectangles. We briefly describe the R-tree data

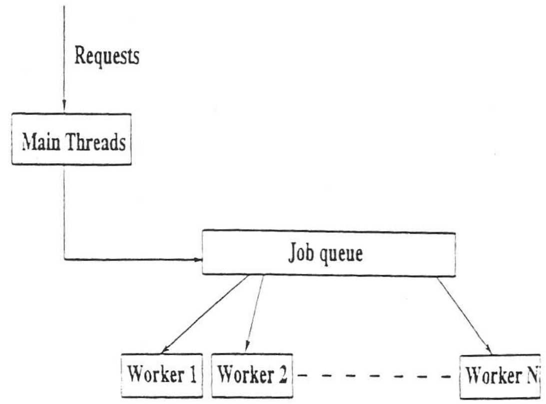

Figure 3: The structure of the thread pool in the information server.

structure here for the purpose of completeness. There are a few important features of R-trees: (1) like B-trees, they remain balanced, but they maintain dynamic adjustable index structure that deals with "dead space" on the pictures, as quadtrees do; (2) all leaf nodes store full non-atomic spatial objects and thus we can do a natural and high-level object-oriented search; (3) often the nodes corresponds to disk pages and thus the parameters defining the tree are chosen so that a small number of nodes is visited during a spatial query. Based on these features, we can see that R-trees can be used to do a direct spatial search for some non-atomic spatial objects, can be used to represent a medium such as quadtrees, and can be used as a storage structure. Following is the brief description of the R-tree structure.

Let M be the maximum number of entries of one node and $m \leq M/2$ be a parameter specifying the minimum number of entries in a node. An R-tree structure specifies the following properties:

1. All leaf nodes appear at the same level;

2. Each entry in a leaf node is a 2-tuple of the form (MBR, oid) such that MBR is the minimum boundary rectangle that spatially contains object pointed by *oid*;

3. Each entry in a non-leaf node is a 2-tuple of the form (MBR, ptr) such that MBR is the minimum boundary rectangle that spatially contains the MBR in the child node pointed by ptr.

4. Parameters m and M indicate that each node in the tree contains between m and M entries, with the exception of the root, which has at least two entries unless it is a leaf node.

The drawback of R-trees is that they do not result in a disjoint decomposition of space, and an object is only associated with one boundary rectangle. This property makes zero overlap between two (or more) leaf $MBRs'$ unachievable, unless we know the data points in advance. The performance of R-tree searching depends heavily on non-overlap and obviously efficient R-tree searching demands that overlap be minimized. But it is hard to control the overlap during the dynamic splits of R-trees, and efficient search degrades. Several tree structures have been developed, such as R+ trees [11], cell trees [3], and R* trees [1] to achieve non-overlap at the expense of space. These trees are based on the decomposition of the rectangles representing objects into smaller sub-rectangles representing partitions of sub-objects in order to avoid overlap among minimal bounding rectangles. If a given rectangle covering a spatial object at the leaf level overlap with another rectangle, we decompose it into several non-overlapping sub-rectangles, all of which make up the original rectangle. All pointers of such sub-rectangles point to the same spatial object. This splitting mechanism is propagating up to the nonleaf nodes, and thus, the non-overlap is achieved. These structures exhibit very good search performance, especially for point querying, at the expense of extra space. However, they have drawback that the decomposition is data-dependent. This means that it is difficult to perform tasks that require composition of different data sets.

In the Terrafly system, we use R-trees because one goal of Terrafly is to deal with a number of different data sets, and other data structures are not sufficient to do this job. Furthermore, based on our research work, we found that when we deal with 2-dimensional spatial point objects of GNIS, we do not encounter a significant overlapping problem. The R-tree is well-suited to our queries and can save large amounts of space.

3.3. Range Query and Nearest Neighbor Query in TerraFly

TerraFly provides users with range query and nearest neighbor query capabilities. A range query searches spatial objects in a specific covering area around a specific spatial object that a user has selected. Given a user-specified point (latitude, longitude), a distance r, and an R-tree whose root node is RT, a range query finds all index records whose rectangles overlap the area S specified by (latitude, longitude) and r. The range query search algorithm descends the tree from the root in a manner similar to B-tree. However, if zero-overlap is not attainable, then more than one subtree under a node may need to be searched, and it is not possible to guarantee good worst-case performance. Carefully designed R-tree insertion and update algorithms will well maintain the tree so as to minimize the searching of irrelevant regions of indexed space, and examine only data near the range query area [4].

Nearest neighbor queries are to find the K nearest neighbor (K-NN) objects to a given point in a space. Such queries are substantially different from the search algorithm of a range query. In Terrafly, we use an efficient algorithm based on the branch-and-bound algorithm for processing exact K-NN queries for the R-trees [8].

4. CONCLUSION

In this paper, a GIS system called Terrafly is introduced. The Terrafly system is a multimedia application that allows users to view image, manipulate the retrieved data, and issue range queries and nearest neighbor queries. An R-tree data structure is implemented to handle the queries. The Terrafly information server uses multithreading to achieve better performance. A semantic object-oriented database management system is developed to meet the database requirements. Spatial data such as the maps can be stored to and retrieved from this database. Unlike existing GIS applications that only provide limited functions and require advanced knowledge of spatial data, the Terrafly multimedia system allows any user who has basic knowledge of computers to interact with the system and explore spatial data of his/her interest.

5. REFERENCES

[1] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R^*-tree: An Efficient and Robust Access Method for Points and Rectangles," *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 322-331, 1990.

[2] Yue-Hong Chou, *Exploring Spatial Analysis in Geographic Information Systems*, ONWORD press, 1997.

[3] O. Gunther, *Efficient Structures For Geometric Data Management*, Lecture Notes in Computer Science 337, Springer-Verlag, Berlin, 1988.

[4] A. Guttman, "R-tree: A Dynamic Index Structure for Spatial Search," *in Proc. ACM SIGMOD*, pp. 47-57, June 1984.

[5] Bil Lewis, Daniel J. Berg, *Multithreaded Programming With Pthreads*, Sun Microsystems Press, 1998.

[6] Bradford Nichols, Dick Buttlar, Jacqueline Proulx Farrell, *Pthreads Programming*, O'Reilly & Associates, Inc. 1996.

[7] Naphtali Rishe, *A Semantic Approach to Database Design: the Semantic Modeling Approach*, McGraw-Hill, 1992.

[8] N. Roussopoulos, C. Faloutsos, and T. Sellis, "Nearest Neighbor Queries," *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 71-79, 1995.

[9] Naphtali Rishe, Wei Sun, David Barton, Yi Deng, Cyril Orji, Michael Alexopoulos, Leonardo Loureiro, Carlos Ordonez, Mario Sanchez, Artyom Shaposhnikov, "Florida International University High Performance Database Research Center," *SIGMOD Record*, 24(1995), 3, pp. 71-76.

[10] *Semantic Object-Oriented Database Management System: C++ Interface Programmer's Guide*, Ciotek Version, High Performance Database Research Center, Florida International University, Miami, FL, 1998.

[11] Timos Sellis, Nick Roussopoulos, and Christos Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimentional Objects," *Proc. 13th Int. Conf. on Very Large Data Bases*, pp. 507-518, 1987.