

OPTIMIZING A SPATIOTEMORAL INDEX TO SUPPORT GEOSS
CLEARINGHOUSE

by

Jizhe Xia
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Geographic and Cartographic Sciences

Committee:




Dr. Chaowei Yang, Thesis Director



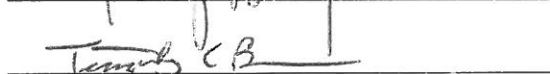
Dr. David Wong, Committee Member



Dr. Songqing Chen, Committee Member



Dr. Peggy Agouris, Department Chairperson



Dr. Timothy L. Born, Associate Dean for
Student and Academic Affairs, College of
Science



Dr. Vikas Chandhoke, Dean, College of
Science

Date: December 5, 2012

Fall Semester 2012
George Mason University
Fairfax, VA

Optimizing an Index with Spatiotemporal Patterns to Support GEOSS Clearinghouse

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Jizhe Xia
Bachelor of Science
The South China Normal University, 2010

Director: Chaowei Yang, Associate Professor
Department of Geography and Geoinformation Science

Fall Semester 2012
George Mason University
Fairfax, VA

Copyright: 2012 Jizhe Xia
All Rights Reserved

DEDICATION

This is dedicated to my parents.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Chaowei Yang for his mentorship during my graduate studies at George Mason University.

I would like to thank all committee members for their guidance in the dissertation.

I would like to thank all members of the Center for Intelligent Spatial Computing for Water/Energy Science (CISC) at George Mason University.

Finally, and most importantly, I would like to thank my parents. They have been always supporting me in my life.

TABLE OF CONTENTS

	Page
LIST OF TABLES.....	vi
LIST OF FIGURES	vii
ABSTRACT.....	viii
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 LITERATURE REVIEW	4
2.1 The R-tree Algorithm Optimizations.....	4
2.1 Distributed Indexing.....	5
2.2 Spatiotemporal Indexing.....	7
CHAPTER 3 METHODOLOGIES	9
3.1 Predefined Multiple Indices Mechanism (PMIM).....	9
3.2 Index Life Cycle and Maintenance Costs.....	12
3.3 User Behavior Identification Using Spatiotemporal Principles	14
3.4 Indexing with Known and Predicted Query	19
3.5 Access Possibility R-Tree (APR-tree).....	22
3.5.1 APR-tree Structure.....	22
3.5.2 APR-tree Algorithm.....	23
3.6 Index Updating Mechanism.....	26
CHAPTER 4 INDEXING ARCHITECTURE	27
4.1 Architecture and Components	27
4.2 Workflows	29
CHAPTER 5 EXPERIMENT & RESULT.....	32
5.1 Identified Spatiotemporal Patterns of Users Behavior	32
5.2 Performance Experiment	36
5.2.1 Experiment Data and Environment.....	36
5.2.2 PMIM vs. R*-tree	37
5.2.3 APR-tree query performance with different weights of access possibility... 38	
5.2.4 APR-tree with Different M Values	39
5.2.5 Performance Experiment for Index Life Cycle.....	40
CHAPTER 6 CONCLUSION.....	44
REFERENCES	50

LIST OF TABLES

Table	Page
Table 1 Table design for storing log information.....	24

LIST OF FIGURES

Figure	Page
Figure 1 GEOSS Clearinghouse can be distributed in different regions with the help of cloud computing services.....	10
Figure 2 All Users use the same context of index in a Spatial DBMSs	11
Figure 3 PMIM to support different categories of users.....	12
Figure 4 Life cycle of an index in PMIM	13
Figure 5 Visualization of the change of “public health” search from 2004 to 2011	17
Figure 6 present relation between user locations, time and interested regions and.....	19
Figure 7 Spatial features and their index without and with predictable queries.....	20
Figure 8 Spatial features and their index with(B) and without(A) predicted access	22
Figure 9 Tree structure of APR-tree	23
Figure 10 APR-tree algorithms and operations are affected by both spatial	25
Figure 11 Extended ChooseLeaf algorithm based on access possibility	25
Figure 12 Architecture of the optimized index system.....	29
Figure 13 Spatial query workflow	30
Figure 15 Population densities and end user distribution of the GEOSS	33
Figure 16 Bounding boxes of spatial queries from Europe users	33
Figure 17 Accumulated access of the GEOSS Clearinghouse at 4:00 A.M., 4:00 P.M. ..	34
Figure 18 The number of queries with the keyword “earthquake” in 2011.....	36
Figure 19 Query performance of PMIM vs. R*-trees with different number of	38
Figure 20 Query performance of APR-tree with different access possibility weights	39
Figure 21 Query performance of APR-tree with different M values.....	40
Figure 22 Tradeoff between PMIM maintenance cost and performance	42

ABSTRACT

OPTIMIZING AN INDEX WITH SPATIOTEMPORAL PATTERNS TO SUPPORT GEOSS CLEARINGHOUSE

Jizhe Xia, M.S.

George Mason University, 2012

Dissertation Director: Dr. Chaowei Yang

Big Data becomes increasingly important in almost many scientific domains, especially in geographic science where hundreds to millions of sensors are collecting data of the Earth continuously (Whitehouse News 2012). The data are managed and served through various Geospatial Cyberinfrastructure (GCI) components worldwide, and many GCI components are also developed to help discover and utilize the widely geographically dispersed data. In the Internet Era, users expect to receive responses in seconds for the discovery and it is a big challenge to achieve it with a proper index. For example, the R-tree (Guttman 1984) leverages spatial relationship among features and is widely used in spatial DataBase Management Systems (DBMSs) and different R-tree variants have been proposed to 1) improve data retrieval performance, 2) support temporal indexing, and 3) utilize multiple computers for indexing. However, it is hard to meet the seconds expectation because little research has included spatiotemporal patterns of user queries.

Traditionally, user behavior has rarely been considered in a spatial index and only one single index is used to support all users from different regions at different times. I propose a Predefined Multiple Indices Mechanism (PMIM) to support global user queries by predefining different indices for different categories of users who have similar query patterns. Access Possibility R-tree (APR-tree) is proposed to build an index based on spatiotemporal patterns of user queries. The new spatiotemporal indexing strategy provides a potential solution to leverage Big spatial Data indexing and enable seconds response to global users. Using metadata in the GEOSS Clearinghouse as an example, I conducted a series of performance experiments for PMIM implemented using APR-tree. Experiment results indicate that new indexing mechanism outperforms a regular R*-tree.

CHAPTER 1 INTRODUCTION

The rapid advancement of Earth observation technologies dramatically increases our capability in acquiring geospatial data. This explosive growth of data is also referred as data intensity or Big Data. The intergovernmental Group on Earth Observations (GEO) proposed Global Earth Observation System of Systems (GEOSS) Common Infrastructure to leverage the discovery, access, and usage of the Big Earth observation Data. The GEOSS Clearinghouse is the engine that drives the entire infrastructure by providing access to the descriptions of resources and search engine (Liu et al., 2011). By July 31, 2012, 29 catalogues and 167K metadata had been registered into the GEOSS Clearinghouse for sharing among over 140 countries (Huang et al., 2010). As a global operational system, the GEOSS Clearinghouse is required to provide high-speed metadata retrieval for global users. It is a big challenge to respond in seconds to users who are widely distributed in different world regions at different times. For example, users from high access density areas may receive very slow response at access rush hours. In addition, with the increasing number of metadata registered, query response time will increase linearly if no optimization were taken (Sardadi et al., 2008). Managing a massive number of spatial features and performing high-speed queries are also big challenges for the GEOSS Clearinghouse operation.

A spatial index is one of the most efficient mechanisms to improve the performance of spatial feature management and data retrieval. The R-tree (Guttman 1984) is one of the most popular spatial indices. Improvements and optimizations have been conducted for the R-tree and resulted in the R+-tree (Sellis, Roussopoulos, and Faloutsos 1987), the R*-tree (Beckmann et al., 1990) and the Hilbert R-tree (Kamel and Faloutsos 1994). These R-tree variants use different principles with different goals to optimize the R-tree. The R-tree family is widely used in spatial DBMSs such as Oracle Spatial, PostgreSQL, Informix, and others (Korthuri, Ravada, and Abugov 2002; Informix 2003; Kothuri et al., 2008; PostGIS 2011). Spatial features of the metadata in the GEOSS Clearinghouse are indexed using the R-tree supported by PostgreSQL. It takes 1-3 seconds to retrieve hundreds of metadata in a 1GB network supported by a server with 4 CPU cores and 16GB memory. Even if the traditional spatial index mechanism is adopted, query performance is still a big issue when the number of features increases to millions and billions. Moreover, current indexing solutions do not consider the spatiotemporal distribution of users and their behavior, which may significantly change the strategy for indexing features. For example, intensive queries against a specific region at a specific time window should be considered in the indexing process. A better index could be designed using spatiotemporal patterns of user queries. An index considering the spatiotemporal patterns of users are needed to index massive number of spatial features and support global users.

In this thesis, I propose a Predefined Multiple Indices Mechanism (PMIM) integrating the spatiotemporal patterns to support global user queries. In the PMIM,

different indices are predefined and stored at an index store. Each index is specially designed for one category of users who have similar query patterns. Different indices will be distributed at different locations based on the spatiotemporal distribution of users. And the entire PMIM will support different categories of users from different regions at different times. Finally, users will be supported by a proper index to speed up the data retrieval process. In the PMIM, the construction of each index will consider spatiotemporal patterns of user queries. A new index structure, Access Possibility R-tree (APR-tree), is proposed to build an R-tree based index using the access possibility of features. Spatiotemporal patterns of user queries are identified by studying the historical user behavior in the GEOSS Clearinghouse. The new spatiotemporal indexing mechanism can be used to support GEOSS Clearinghouse, GCI (Yang et al., 2010) components, Spatial DBMSs, and to address Big Earth observation Data management.

CHAPTER 2 LITERATURE REVIEW

2.1 The R-tree Algorithm Optimizations

The R-tree (Guttman 1984) extended the B-tree from one dimension to multi-dimension for static features. In the R-tree, spatial features are reorganized into different levels of entries based on their spatial relationship to support high-speed data retrieval. Generally, the process of constructing an R-tree is the process of inserting all features into the tree. This process includes 1) choosing a leaf-node to insert a feature based on the spatial relationship and 2) splitting a node and adjusting the tree when this node has too many features. Therefore, the key to optimize the R-tree is to choose the right node to insert and to reasonably split and adjust the tree.

Early research on the R-tree focused on local optimizations. For example, the R+-tree (Sellis and Faloutsos 1987) is a variation to the R-tree that the tree does not have overlapped nodes. However, good choice of each insertion does not mean a good choice for the entire R-tree construction. To achieve a good optimization in the entire construction process, some insertions have to be compromised. The R*-tree (Beckman et al., 1990) considers both local and global optimizations. Locally, the R*-tree adds multiple spatial factors to insert features. Beside the size of spatial overlap area, perimeters of feature, minimum bounding rectangle (MBR) and dead-zone area are also involved in the insertion process. Globally, a forced reinsert strategy is conducted. Before

splitting nodes, current level node structure will be examined and reorganized to postpone the split process as needed. The Hilbert R-tree, proposed by Kamel and Faloutsos (1993), adopts another optimization aspect. They used a “2D-c” method to sort MBR according to Hilbert value. This method maps spatial data from multi-dimension to one dimension. The Hilbert-tree yields up to 28% saving over the R*-tree according to experiments (Kamel and Faloutsos 1993).

The R-tree and its optimizations allow overlap of features and MBR, therefore, multiple query traces is unavoidable. With the increasing number of spatial features indexed, MBR overlap area increases, which significantly affects the performance of the data retrieval. With fixed features, inserting these features in different orders will construct very different R-trees. The key for the R-tree optimization is to put dynamically changeable spatial features to the right place in the tree structure. It requires finding the right organization of each level of nodes for insertions and deletions of features. Optimized algorithms were adopted by considering more spatial factors and reorganizing the tree structure frequently, but these optimized algorithms significantly increase the time complexity of constructing an R-tree.

2.2 Distributed Indexing

Distributed indexing technologies leverage high performance spatial indexing by utilizing parallel computational resources. Kamel and Faloutsos (1992) proposed a strategy to process the R-tree in parallel. Their approach is a hardware related architecture consisting only one processor with several disks attached to it. Therefore, nodes of the R-tree are stored in distributed disks and only one processor is used for

indexing. Wang et al., (1999) proposed an R-tree searching algorithm on Distributed Shared Virtual Memory (DSVM). This algorithm utilizes both distributed processors and disks. Also, they extended their parallel algorithm from one workstation with multi-processors to multiple workstations. Experiment was conducted based on Shusseuo which provides global persistent object management on DSVM (Wang et al., 1999). The Master-Client R-tree (Schnitzer and Leutenegger 1999) is a distributed R-tree structure in a shared nothing environment. In a Master-Client R-tree, nodes are stored in different machines. One dedicated machine that stores non-leaf nodes of the tree is the master server while leaf nodes are stored in client nodes. Client nodes also maintain complete R-tree structures for the portion of data assigned to it. Because both master and client nodes have complete R-tree structures, the entire distributed tree has redundant information. According to experiments, this redundant storage only adds 1% space overhead based on several synthetic and real data sets. MapReduce (Dean and Ghemawat 2008) provides a programming model that can be used to implement complex parallel processes. This model is used by Cary et al., (2009) to construct distributed R-tree. But a recent result shows no significant improvement when using MapReduce to construct distributed R-tree (Akdogan et al., 2010). The operations of the R-tree need cross-node communications, which is very time-consuming in a distributed system. Therefore, it is necessary to control and scale down the complexity of problem before leveraging large scale computing resources.

2.3 Spatiotemporal Indexing

Tayeb, Ulusoy, and Wolfson (1998) proposed a variant of the quad-tree (Samet 1984) to solve the problem of indexing dynamic point based on the idea of future movement prediction. This structure can only be used to index one dimension features. RT-tree (Xu, Han and Lu 1990) treats time as a new attribute in addition to a regular R-tree. Start and end times are added to spatial features. The main concern to RT-tree is spatial query, and temporal query is considered as a secondary issue. Since time dimension is stored as an attribute in R-tree structure, RT-tree support spatial queries almost as fast as a regular R-tree but time slice queries is inefficient. Theodoridis, Vazirgiannis and Sellis (1996) treat time as the third dimension in addition to spatial dimensions and utilize 3D R-tree to index multimedia applications. 3D R-tree treats temporal queries as important as spatial queries by extending the R-tree from two dimensions to three dimensions. However, 3D R-tree is inefficient in storage. Saltenis et al., (2000) proposed the TPR-tree, which extends the R*-tree indexing strategy to handle moving features. The indexed features and entries are augmented with velocity vectors. A snapshot of the index can be calculated for any query. The TPR*-tree (Tao, Papadias, and Sun 2003) adopts the same structure as TPR-tree but using new insertion and deletion algorithms that significantly improves the updating and query performance.

Spatiotemporal indexing structures combine the spatial index with temporal index. Distributed indexing technologies utilize distributed computing resources for the indexing process. Space and time are indexed using different principles with different goals. However, spatiotemporal thinking, which integrates space and time equally, has

not been considered. The construction of indices needs to consider spatiotemporal principles that govern the evolution of physical and social phenomena (Yang et al., 2011a), which can be described as features. Also, spatiotemporal indexing algorithms can be improved by considering the spatiotemporal pattern of index accesses. In this paper, I study the spatiotemporal patterns of user behavior and extend the R-tree indexing algorithm based on the patterns of index accesses. The PMIM and the APR-tree are proposed to build different indices for different categories of users with each category of users having similar spatiotemporal patterns.

CHAPTER 3 METHODOLOGIES

3.1 Predefined Multiple Indices Mechanism (PMIM)

GCI, Spatial DBMSs and related applications (e.g. GEOSS Clearinghouse) are usually replicated and distributed at multiple locations to balance the load from different regions. For example, multiple instances of an application should be deployed at different locations by considering the distributions of end users, data services and computing resources (Yang et al., 2011b). The GEOSS Clearinghouse is a global spatial metadata catalog which enables the sharing of Earth observation data among over 140 countries. It is important to distribute the GEOSS Clearinghouse at different locations over the world. The cloud computing provides an ideal platform to implement the distribution of GEOSS Clearinghouse replications. For example, Windows Azure and Amazon EC2 enable users to create a replication at all the locations shown in Figure 1. Multiple instances of GEOSS Clearinghouse can be distributed in different regions to support the queries from global user. However, simply replicating the application has many problems. For example, GEOSS Clearinghouse supports eight languages and we need to consider the local language when deploying a new instance at that location. In addition, users from different regions at different times may have very different demands and query patterns. Simply replicating the spatial index cannot best support different users. Therefore, I

propose the PMIM to support users who are dynamically distributed in different regions at different times.

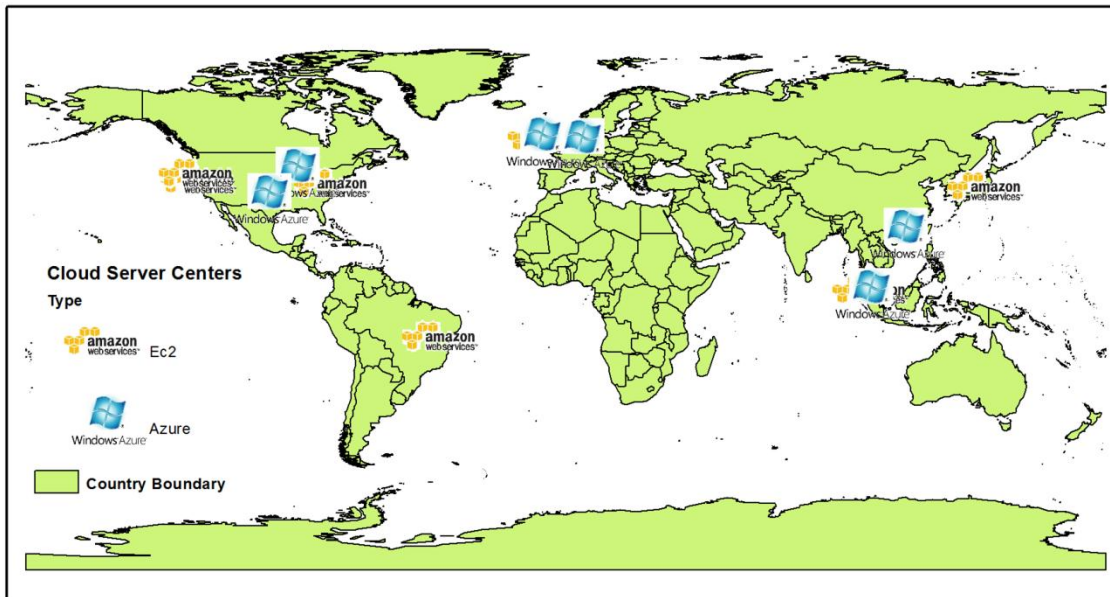


Figure 1 GEOSS Clearinghouse can be distributed in different regions with the help of cloud computing services

The single index mechanism uses only one index file to support spatial queries from all users (Figure 2). Different index files can be created and accessed but the content of these index files are the same. This mechanism does not consider the heterogeneity of user behavior. The optimization of this single index is very limited because it should support all different queries from different users. Instead of utilizing one single index, multiple-indices can be predefined to support different categories of users. Figure 3 illustrates a scenario of using PMIM. In this mechanism, Spatial DBMSs predefine and

maintain different categories of indices. Each index is specially designed for one category of users who have similar query behavior. For example, most users in Category One are from Western Europe and they query Europe data intensively during the Europe daytime. A specially designed index can be predefined to support users from Category One. And this specially designed index will support faster data retrieval process than the original single index. The entire PMIM will outperform the single index mechanism.

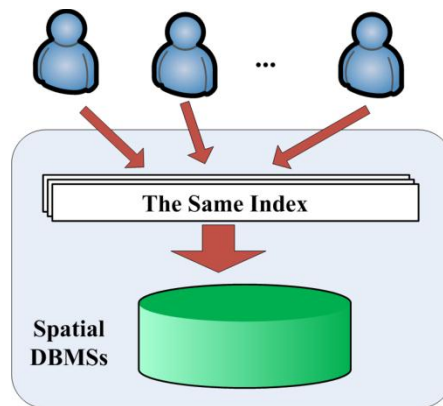


Figure 2 All Users use the same context of index in a Spatial DBMSs

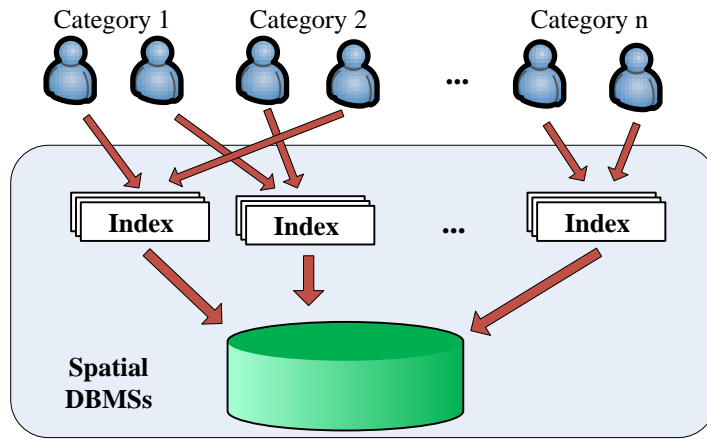


Figure 3 PMIM to support different categories of users

3.2 Index Life Cycle and Maintenance Costs

In the PMIM, multiple indices are created and stored in the index store, which will increase the maintenance costs of spatial DBMSs because the updating of increasing number of indices. Therefore, we need to comprehensively consider the index maintenance costs and index usage rate in an index life cycle. Figure 4 shows the life cycle of the PMIM. Multiple indices are stored in the index store. Users will use these indices for high-speed data retrieval. At the same time, the Index manager collects user behavior information and user feedbacks to analyze index access patterns. Index rebuilding events will trigger the index updating process. Indices in index store will be updated when triggered by events. Finally, users use the updated indices for data retrieval.

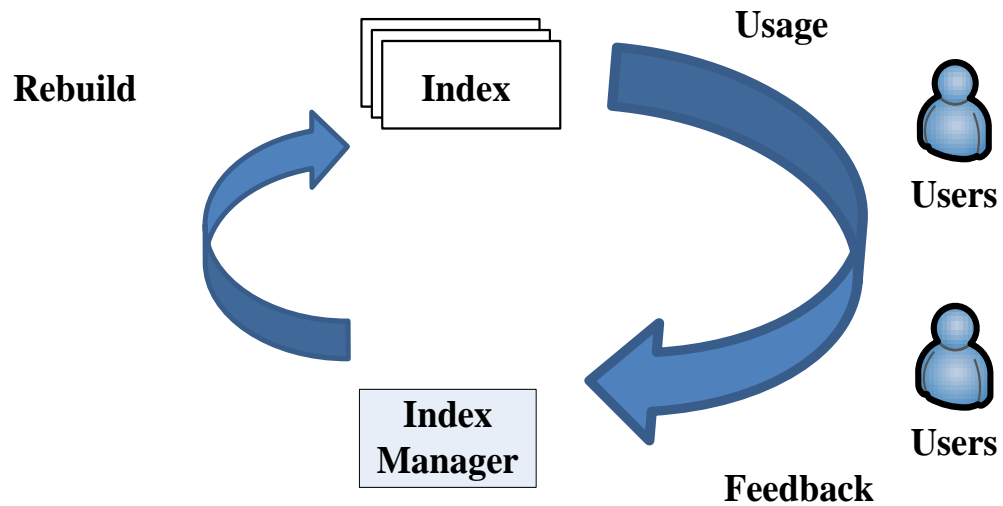


Figure 4 Life cycle of an index in PMIM

In the index life cycle, both index usage and index updating require significant computational resources. Since multiple indices are used, the PMIM can enhance index usage by supporting faster data retrieval, but with a higher maintenance cost. A good index updating mechanism can largely decrease the index maintenance costs and the system delay. But the maintenance may become unaffordable when too many updates are conducted. We need to consider the scale of data, index usage rate, updating frequency requirement, performance requirement and also the computational capability to balance the improvements for retrieval and maintenance cost. For example, we do not want to maintain an index in the PMIM if the index usage rate is very low.

3.3 User Behavior Identification Using Spatiotemporal Principles

Since PMIM is based on different categories of users and each category of users have similar query behavior. I made three assumptions: 1) Users have different behavior across space and time, 2) User behavior can be identified and categorized, and 3) User query patterns are predictable when user behavior is identified. Generally, user behavior is affected by spatiotemporal principles that govern the relationships of phenomena (Yang et al., 2011a). For example, closer things are more related than those farther away measured by spatiotemporal dimensions (Klemm 2006). User behavior is expected to have the following patterns:

- **Local interests:** Users who come from similar locations may tend to query the data that have similar characteristics. Users from a specific area may demand the data related to this area. For example, users from the United States may query the data related to the United States rather than the data related to other regions.
- **Temporal concentration:** Users who are very active in a specific time window may tend to query the data that have similar characteristics. For example, users who are very active in the morning may tend to query the weather data.
- **Background factors:** Users who have similar personal background tend to have similar query behavior. For example, Geologist may tend to query regions involving the lithosphere.
- **Combined pattern:** A comprehensive pattern that combines spatial, temporal and other factors of user behavior, data, applications and GCIs.

To identify user behavior patterns, I study the historical user behavior of the GEOSS Clearinghouse. The spatiotemporal patterns of user behavior are analyzed in three steps:

Step1: Data collection includes collecting user accesses in certain time period. We maintained rich operational information in the GEOSS Clearinghouse log files. We can retrieve from the log files the user behavior information including the access IP and time, spatial and keyword-based search parameters, search results, data access page language and others. The volume of the log file is approximately 88 gigabyte from Dec 22th 2010 to Jan 30th 2012.

Step2: Data preprocessing extracts and reorganizes the information related to user behavior. Related information that can be directly extracted from the log file includes:

- a) Access IP
- b) Access Time
- c) Session ID
- d) Operation Type (including main page access, search, CSW, metadata access and others)
- e) Search parameters
 - e1) text parameters
 - e2) spatial parameters

The longitude, latitude, country and city of an accessing IP are parsed using IP parsing web service¹. All the information are imported into a MYSQL database. Table 1 shows the design of the database table.

Table 1 Table design for storing log information

Field Name	Data Type	Description
Id	INT	Primary key
Ip	VARCHAR(50)	
Lon	FLOAT	Longitude of IP
Lat	FLOAT	Latitude of IP
City	VARCHAR(100)	City of IP
Country	VARCHAR(100)	Country of IP
Time	VARCHAR(50)	The time of access in a day
Date	VARCHAR(50)	The date of access
sessionID	VARCHAR(80)	Access session
accessType	VARCHAR(100)	Operation type
textWords	text	Text parameter of a query
Ifspatial	boolean	If or not a spatial query
spatialType	VARCHAR(80)	Spatial query type (e.g. intersect, within)
West	FLOAT	Query bounding box
East	FLOAT	Query bounding box
North	FLOAT	Query bounding box
South	FLOAT	Query bounding box

Step3: Data analysis can be used to identify user behavior patterns. Spatially, user behavior from different regions is analyzed to identify the distribution, density and pattern of user queries. A simple analysis is to visualize the distribution of users and request density that query against a specific topic or region. For example, the users can be

¹ api.ipinfodb.com

classified and visualized according to their spatial distribution. Also, user behavior can be aggregated at different spatial scales. For example, users from a specific continent, country or city can be aggregated as one group. In this case, diversity can be visualized according to different spatial scales.

Temporal analysis is used to identify patterns of user behavior across time. A simple way is to visualize the change of the search density across time. Figure 5 shows an example of visualizing the change of the search density with a keyword “public health” from 2004 to 2011. The first pattern can be found is that a hot event will significantly increase the search behavior against this event in a very short time. For example, health emergency in United States 2009 caused a dramatic increase of search behavior against “public health”. An index and mechanism that specifically optimized for this event should be built for the intensive usage in this short time. The second pattern is that the frequency of search behavior against a specific topic increases and decreases periodically. For example, at the end of each year, the frequency of search against “public health” drops.



Figure 5 Visualization of the change of “public health” search from 2004 to 2011 from Google Trends

User behavior in spatial DBMSs has both spatial and temporal patterns and user behavior analysis should consider both space and time. Ginsberg et al., (2009) estimated the current level of weekly influenza activity in each region of the United States by analyzing the search behavior of users. Since both space and time are variables, a traditional visualization method cannot present the dynamic change of user behavior across both space and time. One solution is to aggregate spatial and temporal attributes. For example, all user behaviors in January are considered as one behavior group, then presents the changes in each month. But this method put the importance of spatial over temporal aspect by cutting continuous time into different pieces. Another solution is to present user behavior using animation. User access can be reformatted into .KML and visualized in Google Earth. In this case, categorization of space and time can be avoided. Besides the accesses, the user query itself also has spatial and temporal attributes. Figure 6 could present relationships between user locations, time, query regions and query keywords. However, both spatial attributes and temporal attributes have to be aggregated.

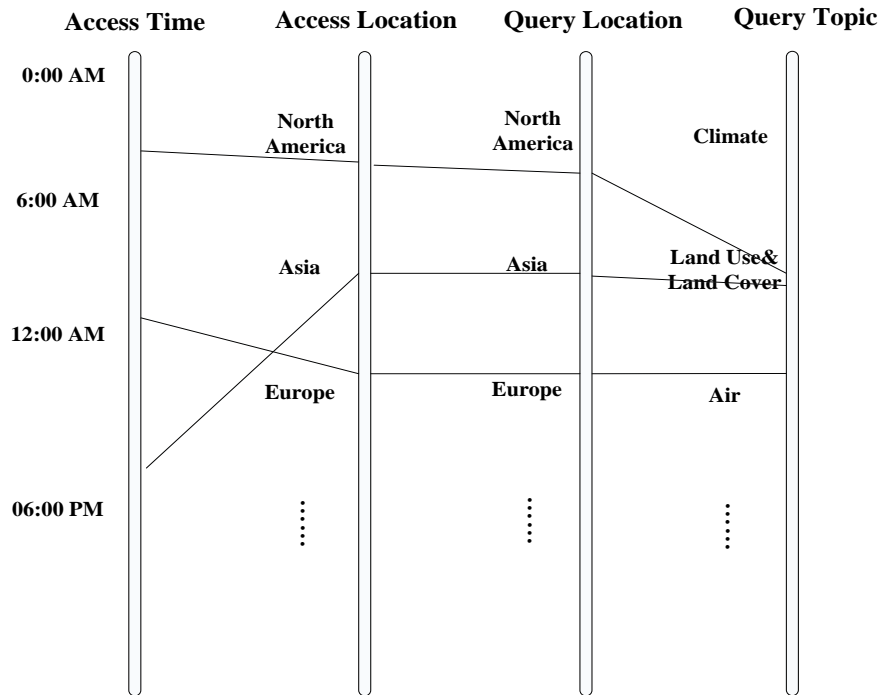


Figure 6 relation between user locations, time and interested regions and topics

3.4 Indexing with Known and Predicted Query

After user behavior and user query patterns are identified, we can design a new indexing strategy using the patterns. In the single index mechanism, index builder assumes that user queries are unknown. This single index needs to support all different queries. Therefore, the optimization of this single index is very limited. If a user query is predictable, the indexing strategy can be changed and the “best index” can be built for this known query. Figure 7 illustrates the change of indexing strategy with predictable query. Figure 7A shows a regular indexing process when a user spatial query is unknown.

Features F1, F2 and F3 are indexed into Node 1 (N1) and features F4, F5 and F6 are indexed into Node 2 (N2) because of the shorter spatial distance. This index can reduce I/O by tracing features using the tree structure instead of looping through all features. But we can build a better index if a user's spatial query is predictable. In Figure 7 B, a user query is to access feature F5, which is put in a single node. A more effective indexing structure is to separate queried features from other features and avoid the overlapping of nodes. This index can minimize I/O cost.

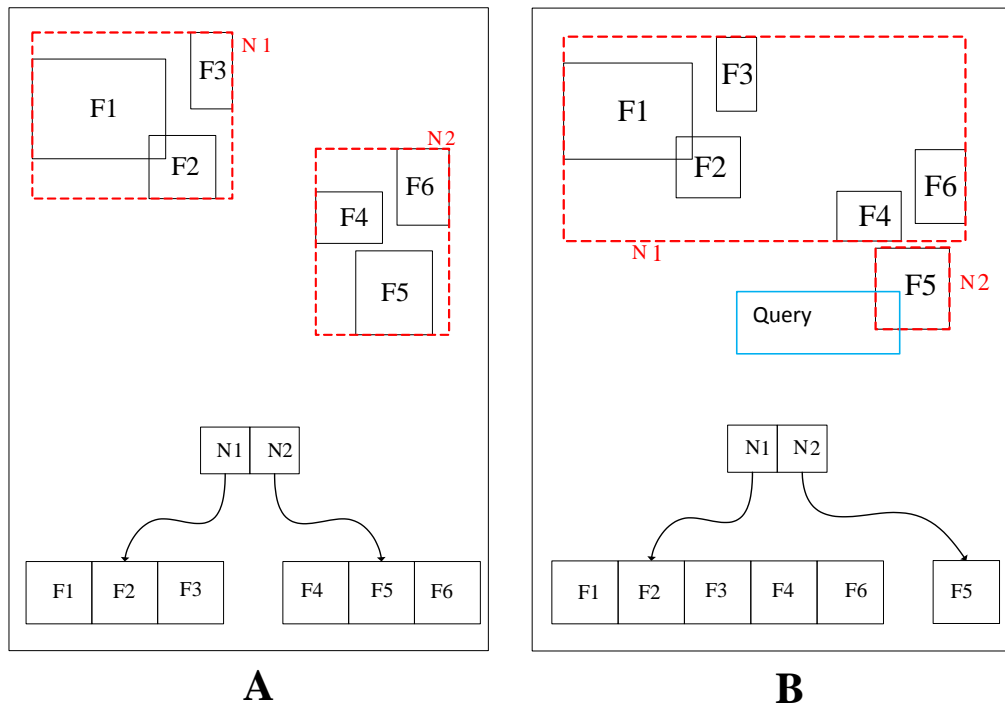


Figure 7 Spatial features and their index without and with predictable queries

Predictable feature access patterns can be utilized to optimize the indexing structure and algorithm. Figure 8 shows an example of an indexing strategy based on feature access prediction. Traditionally, features F1, F2 and F3, features F4, F5 and F6 are indexed into the same node because of the shorter spatial distance (Figure 8A). In Figure 8B, the access possibility of each feature is predicted based on user query patterns and each feature has different access possibilities. Figure 8B shows that feature F4 and F5 have very high possibility to be accessed. Features F4 and F5 can be indexed into node 1 (N1) and A, B, C and F can be indexed into the other node (N2). By using this index structure, features with high access possibility can be separated from features with low access features so that high access features can be put into computer cache. The index with access possibility (Figure 8B) outperforms the original index (Figure 8A) with this given query pattern. Different indices can be built for different user query patterns by using the PMIM proposed in 3.1.

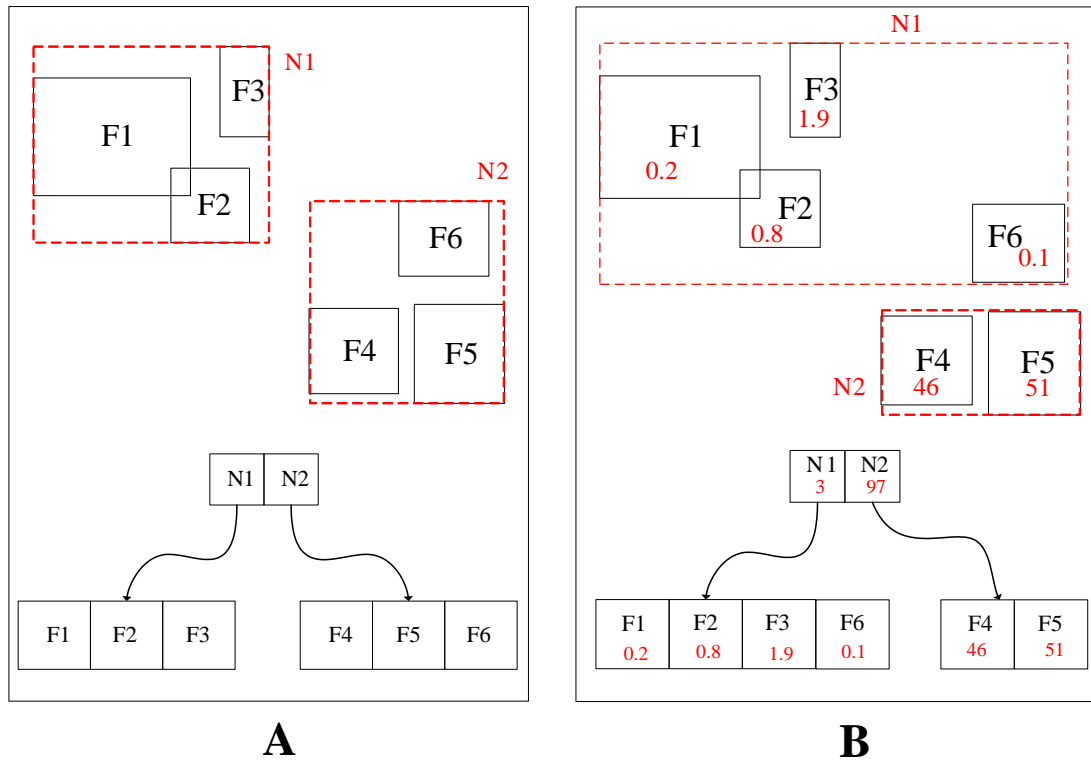


Figure 8 Spatial features and their index with (B) and without (A) predicted access frequency

3.5 Access Possibility R-Tree (APR-tree)

3.5.1 To implement the new index strategy introduced in 3.4, I designed and proposed APR-tree. APR-tree introduces an access possibility of features and nodes in addition to the structure and algorithm of the regular R-tree. APR-tree Structure

In the APR-tree, a new attribute is added to the structure of the regular R-tree. For each feature and data node, the access possibility of each feature will be considered when building the tree. The value of access possibility is calculated according to the

spatiotemporal patterns of user queries in the past and this value could be updated at certain frequency when more user behavior data are analyzed. For non-leaf nodes, the access possibility is also added as a new attribute. The access possibility of a non-leaf node represents the possibility that user will access features that belongs to this node. This value is an aggregation of access possibility value of leaf nodes or non-leaf nodes that belong to a certain node (Figure 9). By using this structure, the access possibility of leaf nodes and non-leaf nodes can be easily retrieved.

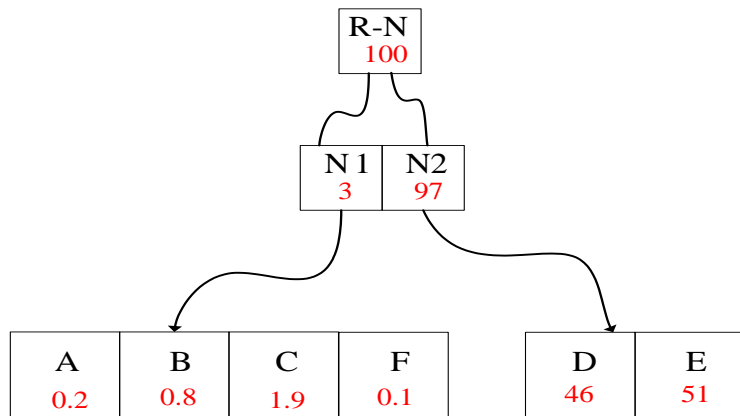


Figure 9 Tree structure of APR-tree

3.5.2 APR-tree Algorithm

Since APR-tree extends the regular R-tree data structure by adding access possibility as a new attribute, the R-tree algorithm needs to be extended. A regular R-tree algorithm includes tree sub-algorithms: Insertion, Deletion and Split. Each sub-algorithm includes several sub-operations. These sub-algorithms and sub-operations need several

inputs such as dimension size, maximum number of entries in one node (M value) and the bounding boxes of features. However, features are grouped merely according to spatial prosperities such as proximity. The access pattern of index is not considered. In the APR-tree, the access possibility will be the second factor that affects all the sub-algorithms and sub-operations related to space partition (Figure 10). For example, Figure 11 shows the “ChooseLeaf” algorithm in the APR-tree. The “ChooseLeaf” algorithm chooses a proper node to insert a new feature which is a key process in the R-tree construction. In the regular R-tree, a new feature is inserted to the node that can minimize the enlargement area of features and nodes. In the APR-tree, features and nodes with high access possibility tend to be inserted into the same nodes. And the access possibility of features and nodes will be calculated. Since two factors are used, the enlargement area and enlargement access possibility should be normalized. A comprehensive score will be calculated and the best Leaf for insertion will be chosen. Using the same method, the access possibility is added to the other sub-algorithms and sub-operations such as Quadratic Split, AdjustTree and others.

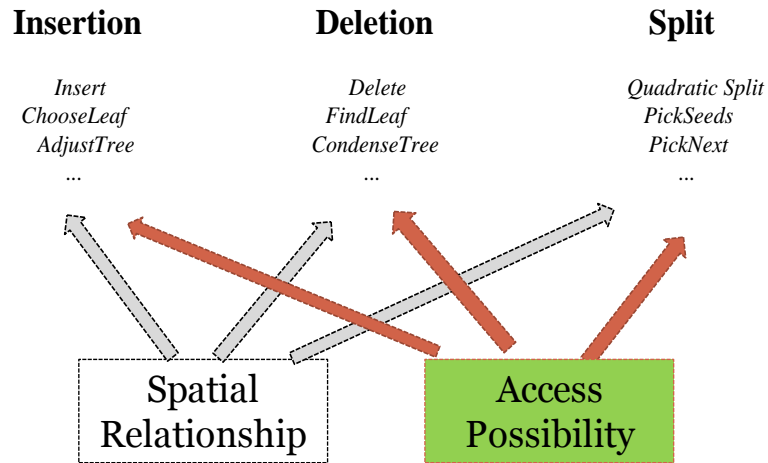


Figure 10 APR-tree algorithms and operations are affected by both spatial relationship and access possibility of features and nodes

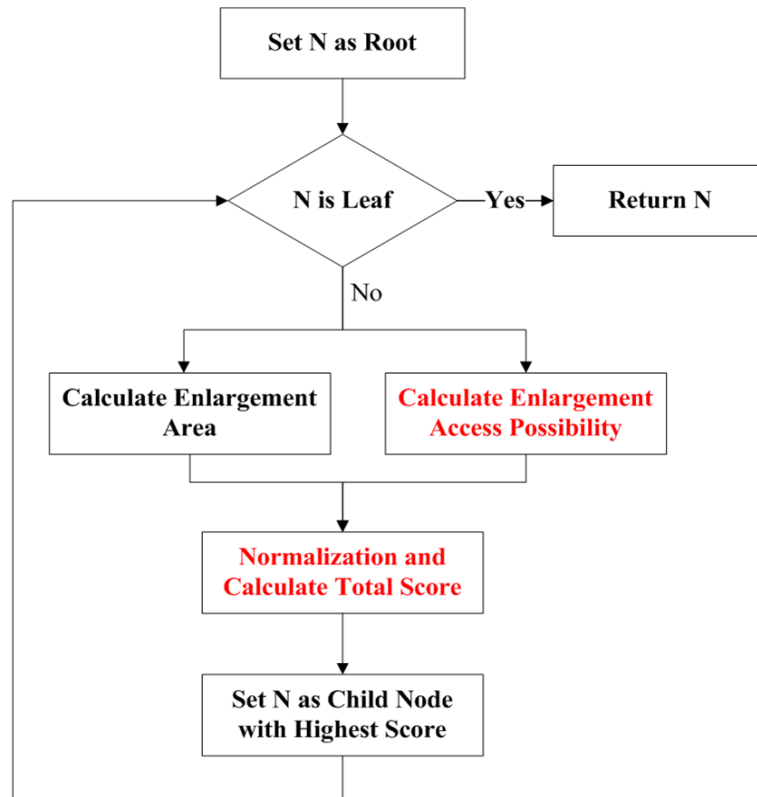


Figure 11 Extended ChooseLeaf algorithm based on access possibility

3.6 Index Updating Mechanism

The R-tree is a dynamic index structure (Guttman 1984). However, the R-tree updating process is still time-consuming, especially in a large scale spatial database. Currently, many spatial DBMSs and GCI components update their spatial index as soon as a feature is registered or removed. This updating mechanism is inefficient and will cause slow response when updating. A routine updating mechanism is needed for large spatial Database. Currently, we consider different rules for a routine updating mechanism. For example, the index will be updated according to time dimension or user patterns. More specific, an index can be updated every month, day, hour or every 100 new features are inserted. Different rules of routine updating mechanisms are widely used in different spatial DBMSs. However, almost no index updating mechanism considers spatiotemporal principle. For example, we can identify the user query patterns in a region to analyze the best time window for updating. This time period should be long enough to finish the index updating process and has a low access density. For natural disaster or other hot events, the spatial index needs to be updated instantly. For some specific topic related data (e.g. natural disaster data), the index need to be updated and optimized in real time to support fast response. This updating mechanism is critical to emergency response for saving lives.

CHAPTER 4 INDEXING ARCHITECTURE

4.1 Architecture and Components

I designed a new indexing architecture based on the PMIM and the APR-tree (Figure 12). An index extension module, index store and index log are added to traditional Spatial DBMSs architecture. Red arrows represent the traditional interaction between access client, spatial DBMSs and spatial data resources. Gray arrows represent the new interaction between these components in new architecture. The index extension module includes search processor and index manager. The search processor performs fast spatial data retrieving and the index manager handles the management and construction of multiple indices. Important components include:

- **User parser** parses the information of an access. This component parses and provides user location, access time and user background information. This information is used as an input for index chooser component.
- **Index chooser** helps user choose a proper index to perform high-speed data retrieving. The mechanism of choosing a proper index is based on user behavior patterns. Also, this mechanism is continually adjusted with more user behavior patterns analyzed.

- **Log parser** provides historical system information. Historical user and system information can be retrieved by parsing the database log file. Information parsed by log parser include a) location, time and background information of historical user, b) spatial query parameters, c) query result, and d) time-consumed. These information are used as input of user prediction model and the trigger for index updating.
- **User prediction module** discovers and predicts user behavior patterns. Historical user behavior information are parsed by log parser and imported into the user prediction module. User behavior patterns will be analyzed and user query patterns can be simulated by existing models (Akaike 1974; Fred 1986; Ajzen 1991; Venkatesh and Fred 2000). User prediction model is also continually adjusted to make more accurate predictions with more user behavior patterns analyzed.
- **Index Trigger & Builder** constructs new spatial index for the index store. The index trigger initiates an index construction process. New user behavior pattern and slow data retrieval process are two important triggers to this process. Instead of using the traditional R-tree construction algorithm, index builder builds APR-tree based on user behavior patterns.
- **Indexes Store** stores predefined indices. Each index is specially optimized for one group of users who have similar query patterns.
- **Log** stores historical user and system operation information. Log file is the input of the log parser_component.

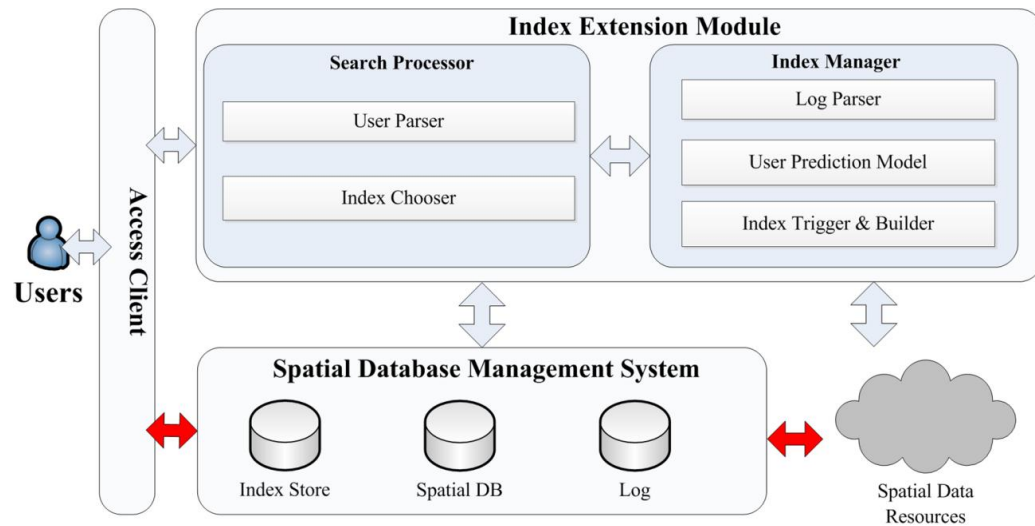


Figure 12 Architecture of the optimized index system

4.2 Workflows

Figure 13 illustrates the workflow of data retrieval process: 1) user information (e.g. location, time and background) are parsed by User Parser. 2) The index chooser selects a proper index for this user, based on user information from User Parser. 3) The selected index will be used to support high-speed data retrieval and results will be returned to users. 4) The summary information of this query will be stored into Database Log.

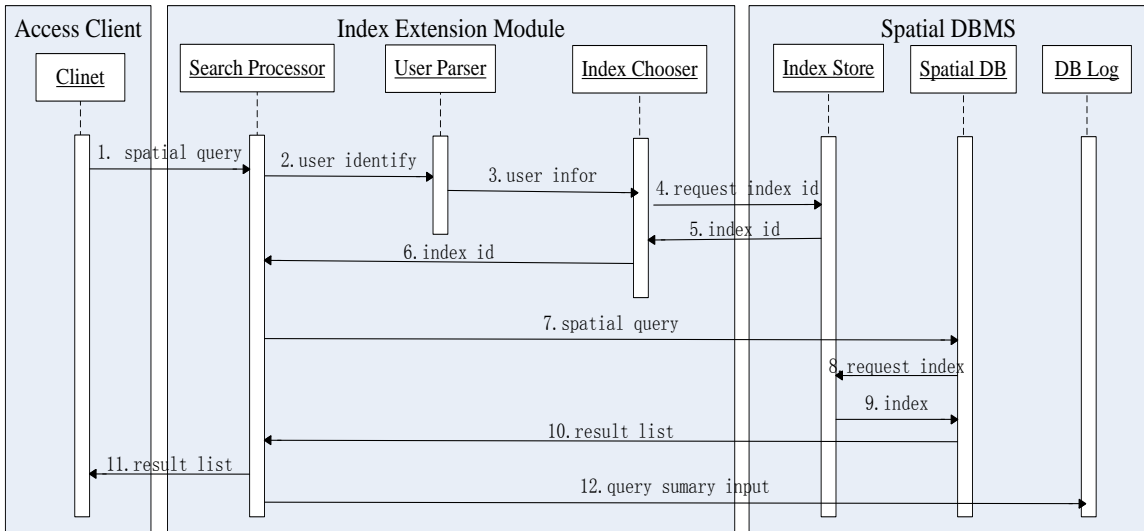


Figure 13 Spatial query workflow

Figure 14 shows the workflow for an index construction process: 1) the index manager is continuously evaluating the system performance. 2) The Index Trigger & Builder builds a new index when necessary. 3) The Database Log is pulled out and parsed by log parser. 4) User prediction model identifies user behavior patterns based on log information and spatiotemporal principles. 5) Finally, a new APR-tree is constructed and input to the index store. To update an index file when inserting new feature, current user behavior patterns need to be pulled out from user prediction model and this feature will be inserted to the APR-tree based on the feature access possibility.

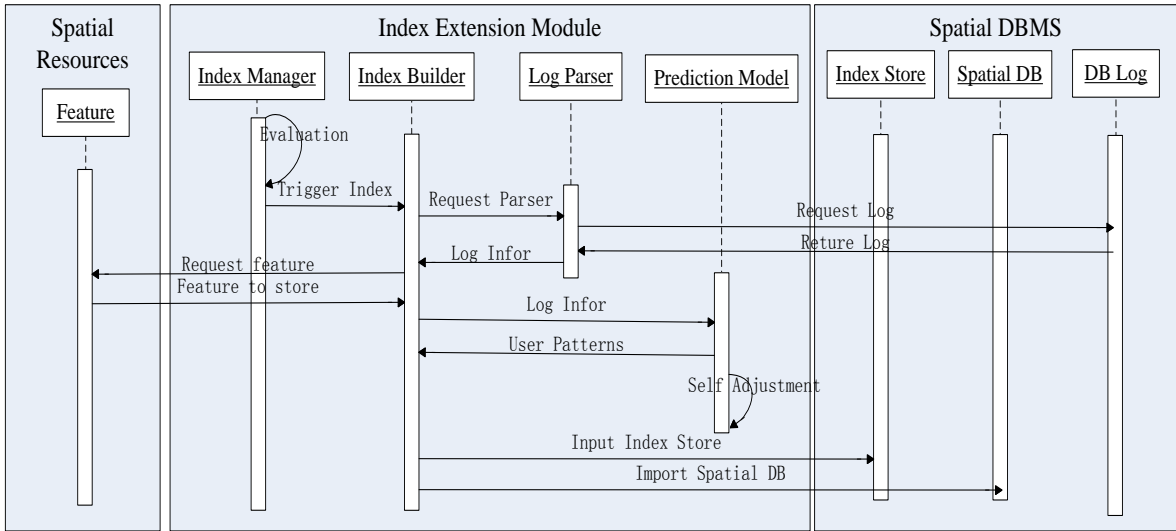


Figure 14 Indexing workflow

CHAPTER 5 EXPERIMENT & RESULT

5.1 Identified Spatiotemporal Patterns of Users Behavior

Based on the historical user behavior of the GEOSS Clearinghouse, different spatiotemporal patterns of user access are identified. From Dec 22th 2010 to Jan 30th 2012, a total number of 1,681, 604 user accesses and 505, 720 user queries have been recorded. The query protocols include OGC: CSW, RSS and queries from end users. Based on the user behavior identification methods introduced in section 3.3, identified spatiotemporal patterns include:

- **High Access Density Regions:** The GEOSS Clearinghouse has high access density from specific regions. Figure 15 illustrates the global distribution of end user request density and the density of global population. Large proportions of users came from the United States and Europe. Generally, regions that have high population densities tend to generate more accesses. More GEOSS Clearinghouse instances should be deployed in high access density regions. Also, the distribution of indices in the PMIM should consider this spatial pattern.

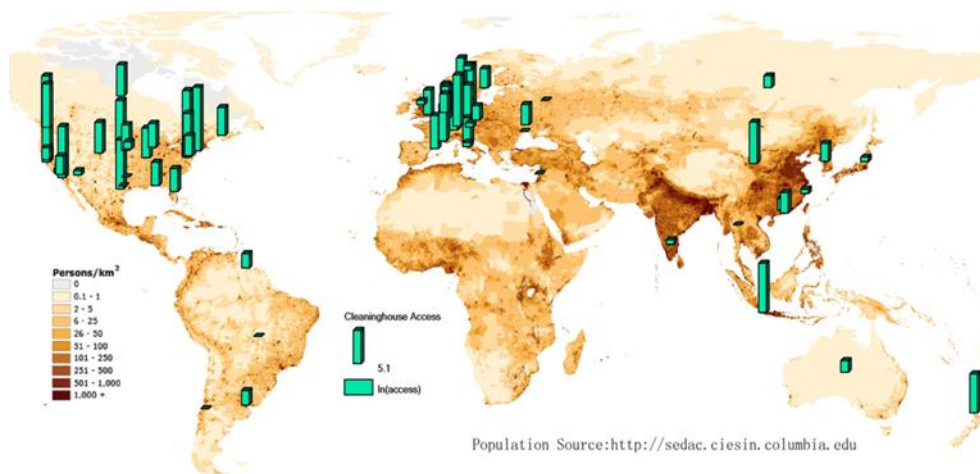


Figure 15 Population densities and end user distribution of the GEOSS Clearinghouse (based on $\log N$, N is access number)

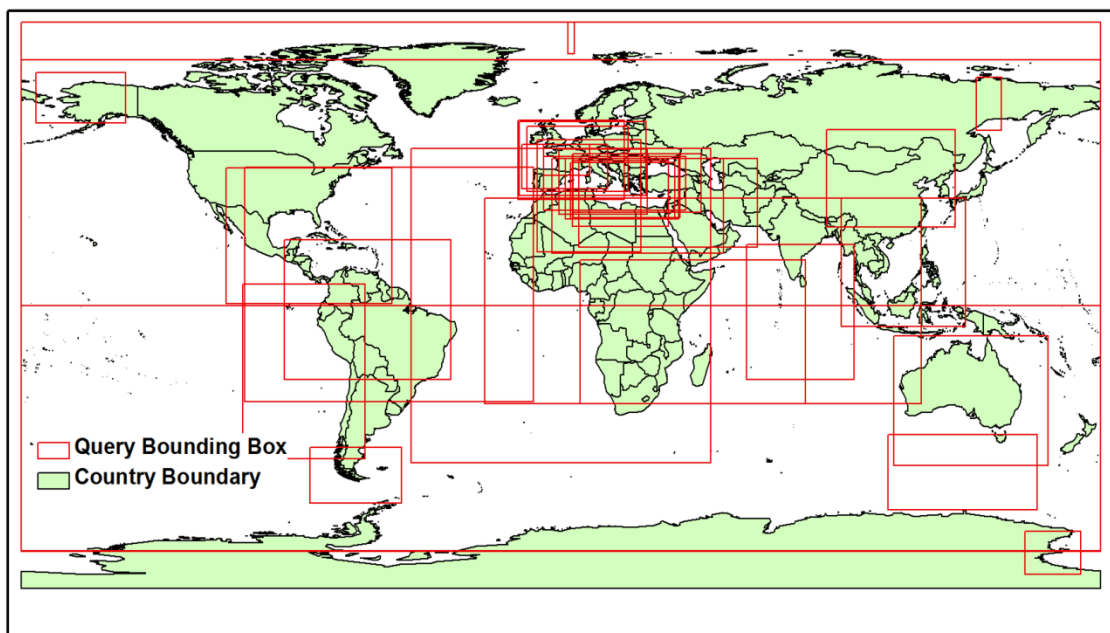


Figure 16 Bounding boxes of spatial queries from Europe users

- Local Interests:** Users from similar locations tend to query similar data. For example, users from a specific region tend to query the data related to this region. Figure 16 shows the bounding boxes of spatial queries from European users. The bounding boxes concentrate on European regions. European users tend to query and access data that are spatially closer to their location.

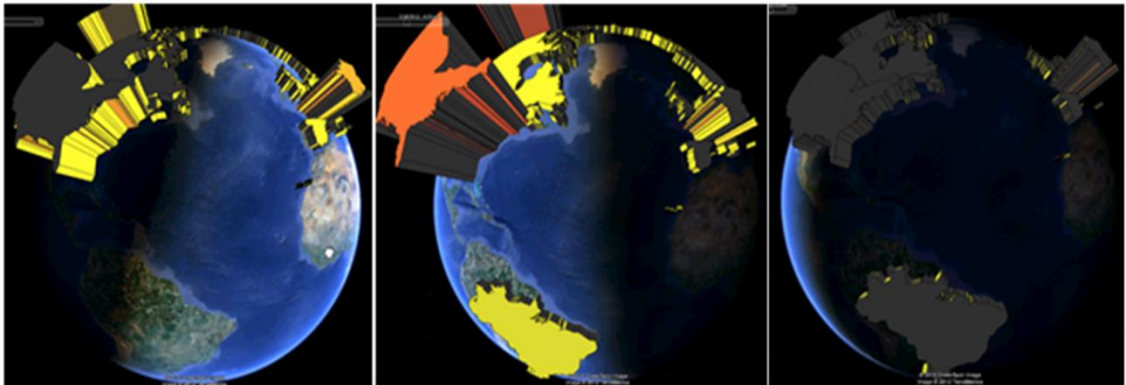


Figure 17 Accumulated access of the GEOSS Clearinghouse at 4:00 A.M., 4:00 P.M. and 8:00 P.M.

- Periodical Accesses & Access Time Windows:** The access density increases and decreases periodically in each day. In most regions, day time tend to generate more user queries than night time. Afternoon and morning are two access rush hours. Figure 17 shows three screenshots of an animation presenting the access density aggregated by counties in 24 hours. Three screenshots represent the accumulated user accesses at 4:00 A.M., 4:00 P.M. and 8:00 P.M. (Eastern Standard Time). Generally, the user request frequency increases from 4:00 a.m. to

4:00 p.m. at local time while the access density drops after sunset. In some regions (e.g. Australia and New Zealand), about 80% of the total requests are generated at day time. More computing resources need to be utilized at day time especially at access rush hours. The index updating time window needs to include this periodical access pattern in different regions.

- **Spike on demand:** A hot event triggers a significant growth of the public interests in a specific topic. The query density for this topic may increase significantly in a very short time. Figure 18 shows the query frequency for the keyword “earthquake” in 2011. The query frequency increase significantly in October. The earthquake in Van, Turkey may have contributed to this significant increase. This 7.2 magnitude earthquake killed more than 600 people on October 23th. In the geographic science domain, hot events are usually related to natural disasters such as earthquakes, floods, hurricanes and others. Different from regular hot events, fast response is critical for natural disasters related queries, because it is important for emergency responses. A specially designed index is needed for intensive queries and fast responses. Also, the spatial index needs real-time updating instead of routine updating.

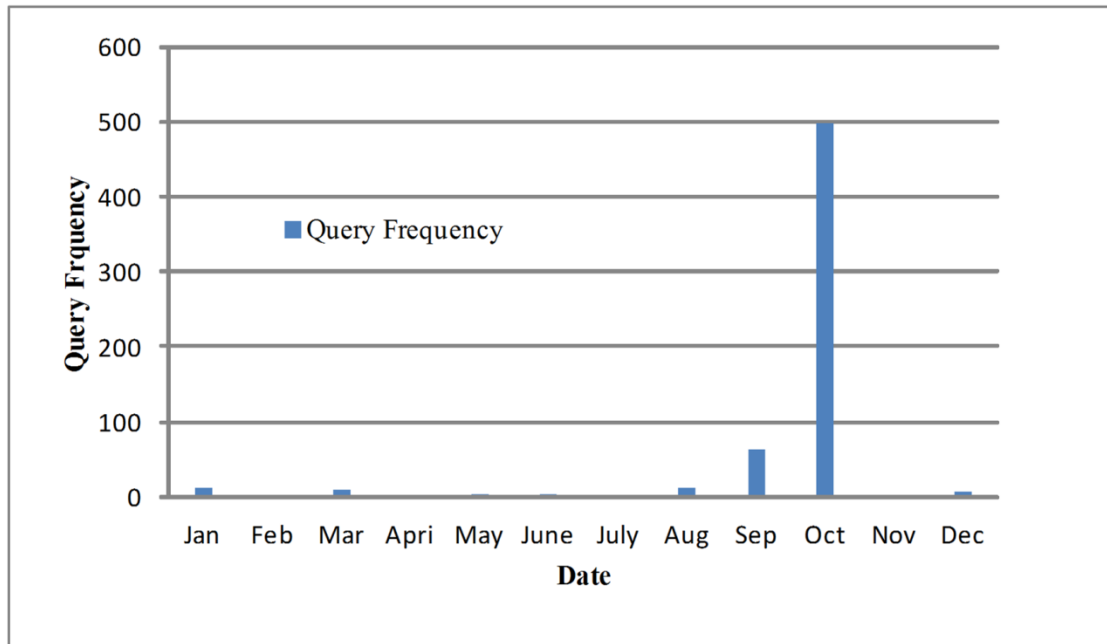


Figure 18 The number of queries with the keyword “earthquake” in 2011

5.2 Performance Experiment

5.2.1 Experiment Data and Environment

167K metadata records in the GEOSS Clearinghouse are used as experiment data. Another 350 k metadata are simulated. The text contents of simulated data are replications. But I randomly shift the bounding box of each simulated data from -10 to 10 degree to make sure the spatial features are not duplicated. I ran the performance comparison on local desktop with the Window 7 Operation System. The CPU is Intel Q8300 with 8 cores and 2.5GHz. The RAM is 4 GB and the storage seeking speed is 7200 rpm. The regular R*-tree and APR-tree are implemented using java based on the

code provided by Dimitris Papadias from Hong Kong University of Science and Technology (HKUST)².

5.2.2 *PMIM vs. R*-tree*

Based on the new indexing strategy, I implemented the PMIM to build three ARP-trees for users from the United States, Europe and China. The weight of access possibility is 0.5 which means that access possibility has the same importance as the spatial factor. The maximum number of entries in one node (M value) is 125. In each region, 50 users are simulated and each user performs 10 queries according to their interests. The query performance is tested using different number of features. The same number of features are also indexed using the regular R*-tree for comparison. Figure 19 shows the query response time of 500 queries from different regions with different feature number. The PMIM outperforms the R*-tree in almost all three regions with different number of indexed features. But the performance improvement is not significant especially when the number of indexed features are 10 k and 60 k. Queries in the United States consume more computing time than other two regions, because the United States contains more features.

² <http://www.rtreeportal.org/code.html>

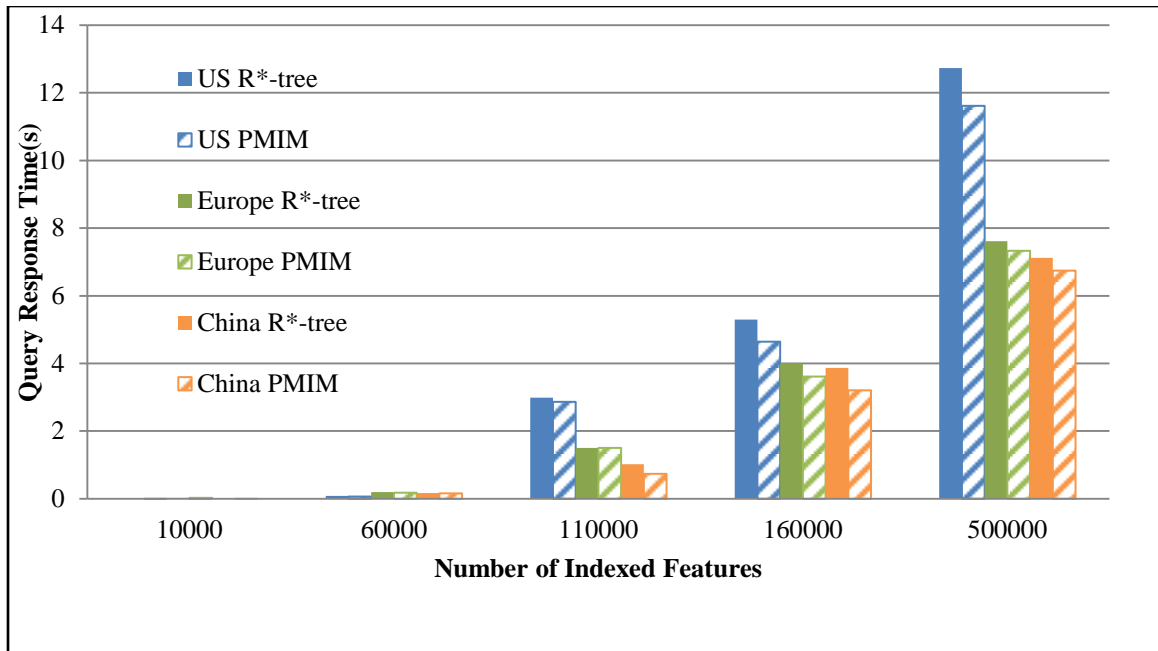


Figure 19 Query performance of PMIM vs. R*-trees with different number of indexed features

5.2.3 APR-tree Query Performance with Different Weights of Access Possibility

Since two factors (access possibility and spatial properties) are used to build an APR-tree, these two factors need to be normalized and weighted. In this experiment, five APR-trees are built using five different weights of access possibility. The M value is 125 and the number of indexed features is 160k. A regular R*-tree is built using the same configuration for comparison. Figure 20 shows the response time of 500 queries for five APR-trees with different access possibility weights. The APR-tree outperforms the R*-tree when the weight of access possibility is less than 0.7. The query response time increases when the weight of access possibility is too high or too low. A larger weight of possibility increases the preservation of feature access but decrease the spatial

discrimination capabilities of the index. The APR-tree may have worse performance than the regular R*-tree if the weight of the access possibility is high. A weight of 0.5 is a good balance of weight for the experiment data.

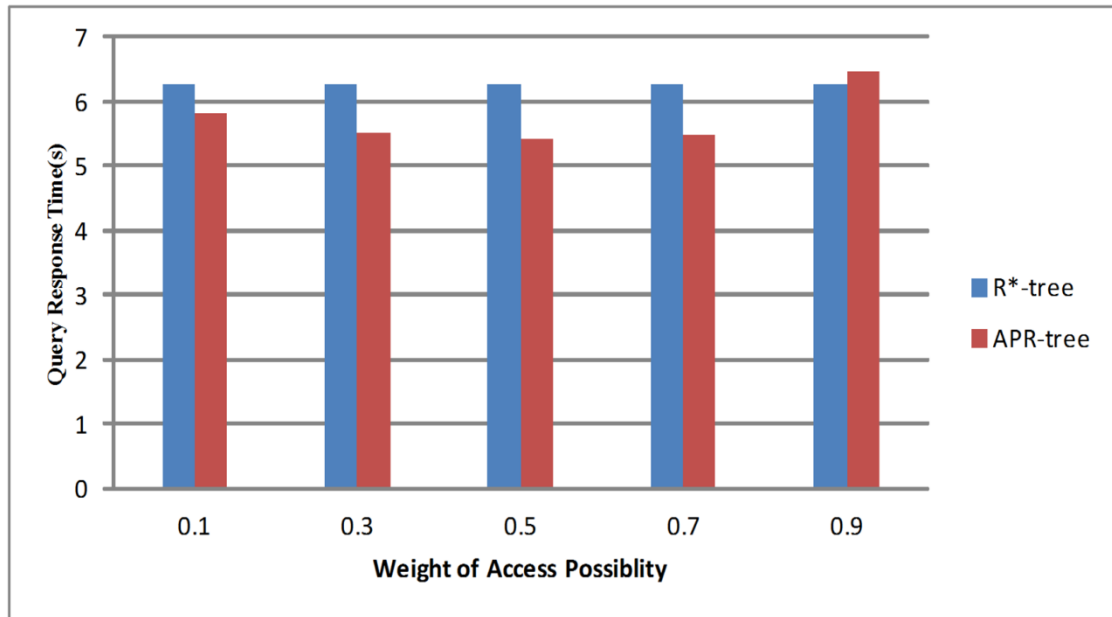


Figure 20 Query performance of APR-tree with different access possibility weights

5.2.4 APR-tree with Different M Values

The maximum number of entries in one node (M value) is an important parameter in R-tree-based index structure by changing the height and width of the tree. Five APR-trees are built using five different M values. The weight of access possibility is 0.5. And the number of indexed features is 160k. Five regular R*-trees are built using the same M values as APR-trees for comparison. Figure 21 shows the response time of 500 queries

for five APR-trees with different M values. The APR-tree outperforms the R*-tree when the M value is larger than 15. The query response time increases when a node has a too large or too small number of entries. When M is too low, the APR-tree may have worse performance than the regular R*-tree. The best M value usually depends on the computing environment and experiment data. The number 125 is a good balance of M values for this experiment data and computing configuration.

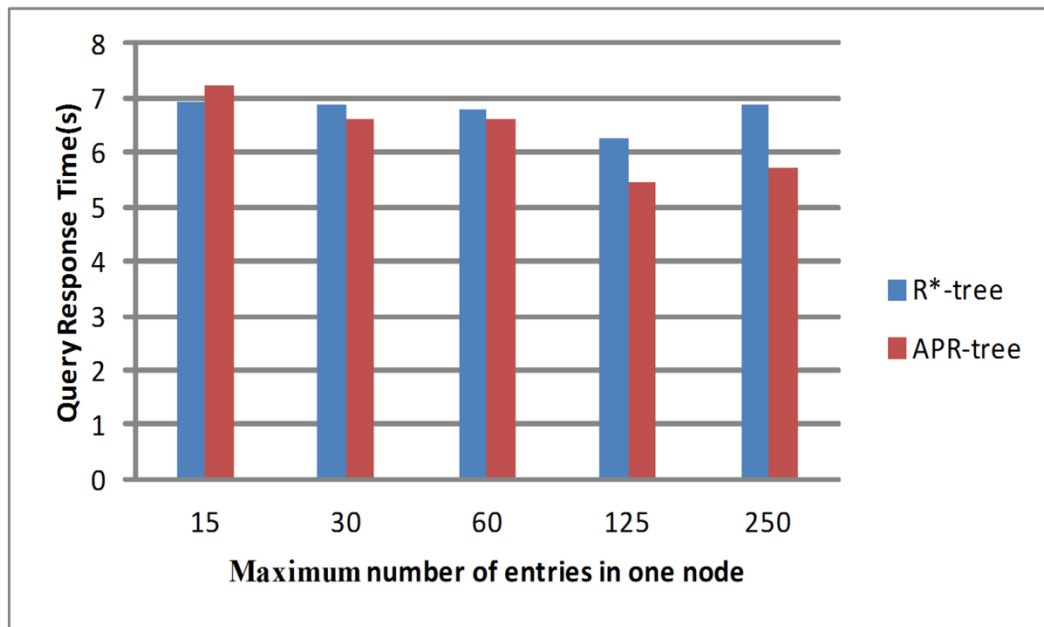


Figure 21 Query performance of APR-tree with different M values

5.2.5 Performance Experiment for Index Life Cycle

The PMIM leverages multiple indices to support different categories of users. Since index updating is a time-consuming process and increasing number of indices

needs to be updated, the PMIM will significantly increase the index maintenance costs. To simulate the index life cycle using the PMIM, we designed an experiment scenario based on the operation of the GEOSS Clearinghouse. In this scenario, I implemented the PMIM to build three different indices for users from the United States, Europe and China. The number of indexed features in the GEOSS Clearinghouse is 167k. I adopt daily routine index updating mechanism, and it normally takes about 4.622 seconds to update the index. Since three indices (United States, Europe and China) are built, the updating cost will be increased by 3 times in an updating cycle. For the APR-tree construction, I used 0.5 for the weight of access possibility and 125 for M value. I assume three different indices are distributed at the United States, Europe and China respectively, and users from each region use the index in that region for queries. User query frequencies in three different regions are calculated according to historical user access data. The United States, Europe and China took 15.5%, 66% and 0.15% of the total accesses. Also, I assume the user prediction module can successfully predict 80% of the user queries. Successfully predicted user queries can be supported by PMIM and will have different levels of performance gains according to different regions. Based on the aforementioned indexing configuration and assumptions, I explored the tradeoff between PMIM maintenance cost and performance improvement in the index life cycle.

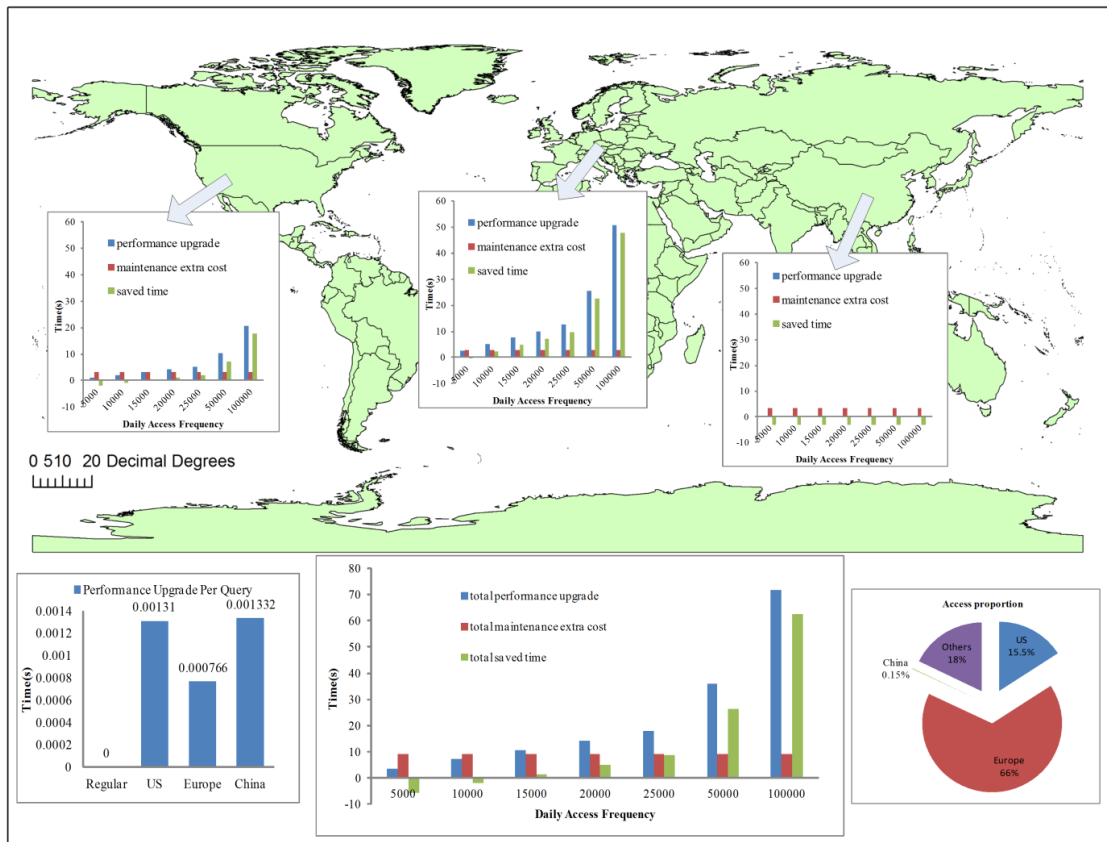


Figure 22 Tradeoff between PMIM maintenance cost and performance improvement in the index life cycle

Figure 22 shows the performance gains of the PMIM and the index maintenance cost with different frequencies of user access. By using the PMIM, users will have different levels of query performance improvements in three different regions. Performance improvements per query are 0.0013, 0.00077 and 0.0013 seconds respectively for users from the United States, Europe and China. However, PMIM needs 9.24 more seconds to update three indices per day. Therefore, there is a tradeoff between

maintenance cost and performance improvement. Here, we introduced a new parameter called B value. B value equals the total time of performance improvement minus the extra index updating time. When a daily index access number is less than 10k, the B value is less than 0, which means performance improvement is less than extra maintenance cost. When index usage rate is low, the PMIM may cost extra time and computational resources. When daily index access is larger than 15k, the B value is large than 0 and increase significantly with the increase of the daily access frequency. The PMIM will save more time and computational resources for the GEOSS Clearinghouse when index usage rate is high. For example, when the daily access frequency is 100k, the PMIM will save up to 62 seconds' computational resources.

According to historical data, different regions have different index access frequencies. The GEOSS Clearinghouse should maintain a new index for high access frequency regions. For example, the user access from Europe take 66% of the total accesses and maintaining a European index will be beneficial when the user access is larger than 10k per day. While the United States needs 15k daily accesses to maintain a local index although this local index have more performance improvement than Europe. China only takes 0.15% of the total accesses and maintaining the index in China is a burden to the GEOSS Clearinghouse in this experiment. Although multiple indices in the PMIM can support faster data retrieval process, we need to evaluate both query performance improvement and maintenance cost according to different regions and time when using the new indexing mechanism.

CHAPTER 6 CONCLUSION

In this thesis, I propose a new indexing strategy to address the challenges of global data discovery by global users. This indexing strategy considers both space and time in that:

- **The Predefined Multiple Indices Mechanism (PMIM)** considers the spatiotemporal pattern of user queries. It predefines and maintains different indices and each index is special designed for one category of users who have similar spatiotemporal behavior. This specially designed index can perform faster data retrieval process compared to a regular spatial index for one specific category of users. And the PMIM will perform higher speed data retrieval for different categories of users.
- **The Access Possibility R-tree (APR-tree)** implement an R-tree based spatial index using access possibility of features. The APR-tree extends the structure and algorithm of a regular R-tree by adding the access possibility of features as a new attribute in tree structure and a new factor in space partition algorithm.

In order to study the spatiotemporal patterns of user behavior, historical user query behavior is extracted and parsed from the GOESS Clearinghouse log file. User behavior is analyzed and represented. The user behavior patterns include local interests,

intensive access regions or time windows, periodic accesses and others. Based on these patterns, metadata in the GEOSS Clearinghouse are indexed by the APR-tree using the PMIM. Performance experiments indicate that the new spatiotemporal indexing mechanism generally outperforms the regular R*-tree based on different M value and access frequency weight. However, we need to consider the balance of performance improvement and index maintenance cost. Nevertheless this indexing mechanism gives a potential indexing solution that can support global users and index large amounts of spatial features.

More research and advancements in relevant fields could greatly enhance the spatiotemporal indexing and my future works include the following research aspects:

- **User query modeling/ prediction** plays a key role in the proposed indexing mechanism because PMIM is trying to predefine multiple APR-trees based on prediction. In this paper, I collected and parsed historical user behavior data from the log file of the GEOSS Clearinghouse. Spatiotemporal patterns of user behavior are analyzed and represented. The prediction of feature access possibilities are calculated manually. To implement proposed indexing mechanism in the operation the GEOSS Clearinghouse, we need a user prediction model that can automatically model and predict user queries. The regression and multiple regression models can simulate the linear relationship between dependent variables and independent variables. In this case, user query results will be the dependent variables while user access location, time and other factors will be the independent variables. However, user behavior prediction is a

complex process and we cannot make comprehensive simulation and prediction by only historical behavior data. We may need to simulate the process of user decision. The technology acceptance model (TAM) (Fred 1986) and the theory of planned behavior (TPB) (Ajzen 1991) are two important models that predict an individual's intention in an information system (IS).

- **Database Caching** is to save frequently-used data into the computer cache. Usually, cached data are stored into computer memory. Using a database cache can reduce disk access, computational usage and most important reduce the time of data retrieval. Suppose 100 users in the GEOSS Clearinghouse are all making the same query, database caching mechanism can increase efficiency enormously by saving query results or parts of query process into the memory so that the computer does not have to repeat the entire query process 100 times. However, database caching costs large memory size. Therefore, we need to consider how many data and what kind of data should be cached. Normally, we preserve certain volume of lately frequently-used query result sets into the cache. But with limited volume of memory size, we can only maintain a small number of result sets if each result set has large number of data. One solution is that database cache can store only parts of the intermediate result in the index query process instead of storing the entire query result set. For example, we could store the tree node that has high access frequency (e.g. the “N2” node in figure 7) into cache instead of the entire result set. In this case, we do not have to repeat the process of accessing some nodes. Moreover, data with high access

frequency in the past may not have high access frequency at current and next time window. As discussed in 5.1, user queries in spatial DBMSs have different patterns and users may frequently access certain dataset periodically. More specifically, users may frequently access similar dataset at certain time windows of the day, week or month. This pattern requires us to change the cache data according to the time period.

- **Extensible indexing framework:** By using extensible indexing framework, a spatial index can be implemented in commercial and open source DBMSs. Extensible indexing frameworks provide templates of indexing structure for the implementation. For example, Generalized Search Trees (Hellerstein, Naughton and Pfeffer 1995) is an extensible indexing framework to support balanced trees (e.g. the B-tree and the R-tree). PostgreSQL uses the Generalized Search Trees for the implementation of spatial indexing extensions. Commercial databases also use different indexing frameworks such as IBM DB2 Spatial Extender and Microsoft SQL Server 2008 Spatial Module. In this research, the implementation of the new indexing structure did not use indexing framework. With the same index structure (R*-tree), the PostgreSQL outperforms our implementation that did not use the indexing framework. Also, the query performances of different index structures based on our implementation are relatively unstable. The java environment might be a potential reason. Therefore, it is critical to implement the new indexing strategy based on extensible indexing frameworks.

- **Spatial Cloud Computing (SCC):** SCC provides dynamically scalable and often virtualized resources over the Internet to support geospatial science and applications (Yang et al., 2011b). SCC is a convenient platform to distribute predefined indices to different regions (Figure 1). As a computing and data intensive application, spatiotemporal indexing could leverage a large computing and storage pool provided by SCC. In addition, certain patterns of user behavior in an index, such as hot events and sudden increase of access rate, can be handled by the elastic computing capacity provided by SCC.

REFERENCES

REFERENCES

1. Ajzen I 1991 The theory of planned behavior. *Behavior and Human Decision Processes* 50:179-211
2. Akaike H 1974 A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19: 716-723
3. Akdogan A, Demiryurek U, Kashani F, and Shahabi C 2010 Voronoi-Based geospatial query processing with MapReduce. In *Proceedings of the IEEE Second International Conference on Cloud Computing Technology and Science*, Indianapolis, Indiana: 9-16
4. Beckmann N, Kriegel H, Schneider R, and Seeger B 1990 The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Atlantic City, New Jersey: 322—331
5. Cary A, Sun Z, Hristidis V, and Rishé N 2009 Experiences on processing spatial data with MapReduce. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, New Orleans, Louisiana: 302 – 319
6. Dean J and Ghemawat S 2008 MapReduce: Simplified data processing on large clusters. *Communications of the ACM - 50th Anniversary Issue* 51: 137-150
7. Fred D 1986 A technology acceptance model for empirically testing new end-user information systems: theory and results. Doctoral dissertation, Sloan School of Management, Massachusetts Institute of Technology (MIT)
8. Ginsberg J, Mohebbi M, Patel R, Brammer L, Smolinski M, and Brilliant L 2009 Detecting influenza epidemics using search engine query data. *Nature* 457:1012–1014
9. Guttman A 1984 R-Trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD international conference on Management of data*, Boston, Massachusetts: 47-57

10. Hellerstein J, Naughton J, and Pfeffer A 1995 Generalized search trees for database systems. In Proceedings of 21th International Conference on Very Large Databases, Zurich, Switzerland: 562-573
11. Huang Q, Yang C, Doug N, Liu K, and Wuhuayi, 2010 Cloud computing for geosciences: deployment of GEOSS Clearinghouse on Amazon's EC2. In Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems, San Jose, California: 35-38
12. Informix 2003 IBM Informix R-tree Index User's Guide. WWW document, www.informix.com.ua/doc/9.40/ct1tana.pdf [Accessed 5 May 2012]
13. Kamel I and FaJoutsos C 1994 Hilbert R-tree: an improved R-tree using Fractals. In Proceedings of the 20th International Conference on Very Large Databases, Santiago de, Chile: 500–509
14. Kamel I and Faloutsos C 1992 Parallel R-trees. In Proceedings of the ACM SIGMOD international conference on Management of data, San Diego, California: 195-204
15. Klemm R 2006 Principles of Space-Time Adaptive Processing. London, Institute of Engineering and Technology
16. Korthuri R, Ravada S, and Abugov D 2002 Quadtree and R-tree indexes in oracle spatial: a comparison using GIS data. In Proceedings of the ACM SIGMOD international conference on Management of data, Madison, Wisconsin: 546-557
17. Kothuri R, Hanckel R, and Yalamanchi A 2008 Using Oracle extensibility framework for supporting yemporal and spatio-temporal applications. In Proceedings of the 15th International Symposium on Temporal Representation and Reasoning, Montr éal, Canada: 15-18
18. Liu K, Yang C, Li W, Li Z, Wu H, Rezgui A, and Xia J 2011 The GEOSS Clearinghouse high performance search engine. In Proceedings of the 19th International Conference on Geoinformatics, Shanghai, China
19. PostGIS 2011 PostGIS 2.0.0 Manual. WWW document, <http://postgis.refractor.net/documentation/manual-2.0> [Accessed 5 May 2012]
20. Saltenis S, Jensen C, Leutenegger S, and Lopez M 2000 Indexing the positions of continuously moving objects. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data, Indianapolis, Indiana: 331-342

21. Samet H 1984 The Quadtree and related hierarchical data structures. *Journal of ACM Computing Surveys (CSUR)* 16: 187 – 260
22. Sardadi M, Rahim M, Jupri Z and Baman D 2008 Choosing R-tree or quadtree spatial data indexing in one Oracle spatial database system to make faster showing geographical map in mobile geographical information system technology. *World Academy of Science Engineering and Technology* 46: 249-257
23. Schnitzer B and Leutenegger S 1999 Master-Client R-trees: A new parallel R-tree architecture, In *Proceedings of the 11th International Conference on Scientific and Statistical Database Management*, Cleveland, Ohio: 68 - 68
24. Sellis T, Roussopoulos N, and Faloutsos C 1987 The R+-tree: a dynamic index for multidimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases Conference*, San Francisco, California: 507-518
25. Tao Y, Papadias D, and Sun J 2003 The TPR*-Tree: an optimized spatio-temporal access method for predictive queries. In *Proceedings of the international conference on Very Large Databases*: Berlin, Germany: 790-801
26. Tayeb J, Ulusoy O, and Wolfson, O 1998 A Quadtree-based dynamic attribute indexing method. *The Computer Journal* 41(3): 185-200
27. Theodoridis Y, Vazirgiannis M, and Sellis T 1996 Spatio-temporal indexing for large multimedia applications. In *Proceedings of the IEEE Conference on Multimedia Computing and Systems*, Hiroshima, Japan: 441-448
28. Venkatesh V and Fred D 2000 A theoretical extension of the technology acceptance model: four longitudinal field studies. *Management Science* 46: 186-204
29. Wang B, Horinokuchi H, Kaneko K, and Makinouchi 1999 A parallel R-tree search algorithm on DSVM. In *Proceedings of 6th International Conference on Database Systems for Advanced Applications*, Kyoto, Japan: 237-245
30. Whitehouse News 2012 Obama administration unveils “BIG DATA” initiative: announces \$200 million in new R&D investments. WWW document, http://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_press_release_final_2.pdf [Accessed 5 May 2012]
31. Xu X, Han J, and Lu W 1990 RT-Tree: an improved R-Tree indexing structure for temporal spatial databases. In *Proceedings of International Symposium on Spatial Data Handling*, Zurich, Switzerland:1040–1049

32. Yang C, Raskin R, Goodchild M, and Gahegan M 2010 Geospatial Cyberinfrastructure: Past, Present and Future. *Computers, Environment and Urban Systems* 34: 264-277
33. Yang C, Wu H, Huang Q, Li Z, and Li J 2011a Using spatial principles to optimize distributed computing for enabling physical science discoveries. *Proceedings of National Academy of Sciences(PNAS)* 106: 5498-5503
34. Yang C, Goodchild M, Huang Q, Nebert D, Raskin R, Bambacus M, Xu Y, and Fay D 2011b Spatial Computing - How can geospatial sciences use and help to shape cloud computing. *International Journal of Digital Earth* 4: 305-329

CURRICULUM VITAE

Jizhe Xia received his Bachelor of Science in Geographic Information Science from the South Chin Normal University in 2010.