11-7-2013

Geospatial Data Indexing Analysis and Visualization via Web Services with Autonomic Resource Management

Yun Lu
Florida International University, yun@cs.fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

GEOSPATIAL DATA INDEXING ANALYSIS AND VISUALIZATION VIA WEB SERVICES WITH AUTONOMIC RESOURCE MANAGEMENT

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Yun Lu

2013

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Yun Lu, and entitled Geospatial Data Indexing Analysis and Visualization via Web Services with Autonomic Resource Management, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Malek Adjouadi

Tao Li

Ming Zhao

Naphtali Rishe, Major Professor

Date of Defense: November 07, 2013

The dissertation of Yun Lu is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2013

DEDICATION

I dedicate this dissertation to my family and my friends. Without their love, patience, understanding and support, the completion of this work would never have been possible.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advising professor, Dr. Naphtali Rishe, who has continuously given me guidance and excellent support throughout my entire Graduate and Doctoral studies at FIU.

I would like to thank my committee members, Professor Ming Zhao, Professor Tao Li, and Professor Malek Adjouadi for serving on my dissertation defense committee and giving enlightening comments.

I would like to thank all members who have been working in the HPDRC team, for their generous support, and always willing to offer suggestions for work and research,

Finally and most important, I would like to express my deepest thank to my family, who provides me with selfless support and generous encouragement during my dissertation writing.

This material is based in part upon work supported by the National Science Foundation under Grant Nos. MRI CNS-0821345, MRI CNS-1126619, CREST HRD-0833093, I/UCRC IIP-1338922, I/UCRC IIP-0829576, RAPID CNS-1057661, RAPID IIS-1052625, MRI CNS-0959985, AIR IIP-1237818, SBIR IIP-1330943, FRP IIP-1230661, III-Large IIS-1213026, SBIR IIP-1058428, SBIR IIP-1026265, SBIR IIP-1058606, SBIR IIP-1127251, SBIR IIP-1127412, SBIR IIP-1118610, SBIR IIP-1230265, SBIR IIP-1256641. Includes material licensed by TerraFly (http://terrafly.com) and the NSF CAKE Center (http://cake.fiu.edu).

ABSTRACT OF THE DISSERTATION

GEOSPATIAL DATA INDEXING ANALYSIS AND VISUALIZATION VIA WEB SERVICES WITH AUTONOMIC RESOURCE MANAGEMENT

by

Yun Lu

Florida International University, 2013

Miami, Florida

Professor Naphtali Rishe, Major Professor

With the exponential growth of the usage of web-based map services, the web GIS application has become more and more popular. Spatial data index, search, analysis, visualization and the resource management of such services are becoming increasingly important to deliver user-desired Quality of Service.

First, spatial indexing is typically time-consuming and is not available to end-users. To address this, we introduce TerraFly sksOpen, an open-sourced an Online Indexing and Querying System for Big Geospatial Data. Integrated with the TerraFly Geospatial database [1-9], sksOpen is an efficient indexing and query engine for processing Top-k Spatial Boolean Queries. Further, we provide ergonomic visualization of query results on interactive maps to facilitate the user's data analysis.

Second, due to the highly complex and dynamic nature of GIS systems, it is quite challenging for the end users to quickly understand and analyze the spatial data, and to efficiently share their own data and analysis results with others. Built on the TerraFly Geo spatial database, TerraFly GeoCloud is an extra layer running upon the TerraFly map and can efficiently support many different visualization functions and spatial data

analysis models. Furthermore, users can create unique URLs to visualize and share the analysis results. TerraFly GeoCloud also enables the MapQL technology to customize map visualization using SQL-like statements [10].

Third, map systems often serve dynamic web workloads and involve multiple CPU and I/O intensive tiers, which make it challenging to meet the response time targets of map requests while using the resources efficiently. Virtualization facilitates the deployment of web map services and improves their resource utilization through encapsulation and consolidation. Autonomic resource management allows resources to be automatically provisioned to a map service and its internal tiers on demand. v-TerraFly are techniques to predict the demand of map workloads online and optimize resource allocations, considering both response time and data freshness as the QoS target. The proposed v-TerraFly system is prototyped on TerraFly, a production web map service, and evaluated using real TerraFly workloads. The results show that v-TerraFly can accurately predict the workload demands: 18.91% more accurate; and efficiently allocate resources to meet the QoS target: improves the QoS by 26.19% and saves resource usages by 20.83% compared to traditional peak load-based resource allocation.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER 1 1
1 INTRODUCTION 1
1.1 Motivation 1
1.2 My work 3
1.2.1 sksOpen 3
1.2.2 GeoCloud 3
1.2.3 v-TerraFly 4
1.3 Dissertation Outline 5

CHAPTER 2 6
2 LITERATURE REVIEW 6
2.1 Geographic information retrieval systems 6
2.2 Spatial data analysis and visualization 9
2.3 Workload prediction and resource management 11

CHAPTER 3 16
3 sksOpen: Efficient Indexing, Querying and Visualization of Geo-spatial Data 16
3.1 Introduction 16
3.2 Background 17
3.2.1 TerraFly 17
3.2.2 Spatial data visualization 19
3.2.3 MapReduce 19
3.2.4 K-NN 20
3.3 Architecture of sksOpen 21
3.3.1 The index algorithm of sksOpen 21
3.3.2 The Structure of sksOpen 25
3.4 Visualization of sksOpen 26
3.5 Case study 28
3.5.1 Setup 28
3.5.2 Performance study 29
3.5.3 User experience study 29
3.6 Related work 32

CHAPTER 4 34
4 GeoCloud: Online Spatial Data Analysis and Visualization 34
4.1 Introduction 34
4.2 Background 36
4.2.1 TerraFly geospatial database 36
4.2.2 Visualizing spatial data 37
4.2.3 Map Rendering 38
4.3 TerraFly GeoCloud 39

4.4 Visualization in TerraFly GeoCloud 42
4.4.1 Spatial Data Visualization 42
4.4.2 Spatial Data Mining Results Visualization 43
4.4.3 Customized Map Visualization (Supported by MapQL) 48
4.5 Case Study 54
4.5.1 Setup 55
4.5.2 Case Study for realtor data analysis 55
4.5.3 A case study for Epidemiological Data Analysis 59
4.6 Related Work and Products 62

CHAPTER 5 65
5 v-TerraFly: Autonomic Resource Management for Virtualized Web Map 65
5.1 Introduction 66
5.2 Background 68
5.3 v-TerraFly 71
5.3.1 Architecture 71
5.3.2 Virtual Load balance cluster 73
5.3.3 Motivating Examples 73
5.4 Autonomic Resource Management in v-TerraFly 77
5.4.1 General Approach 77
5.4.2 Workload Prediction 78
5.4.3 Performance Profiling 81
5.4.4 QoS Model 84
5.5 Evaluation 85
5.5.1 Setup 85
5.5.2 Workload Prediction 86
5.5.3 Resource Management of *Reader Tier* 87
5.5.4 Resource Management of both *Reader* and *Loader Tier*s 90

CHAPTER 6 94
6 CONCLUSIONS AND FUTURE WORK 94
6.1 Conclusions 94
6.2 Future Work 95

LIST OF REFERENCES 97

VITA 108

LIST OF FIGURES

FIGURE PAGE

Figure 3.1 An super-node and leaf nodes 22

Figure 3.2 Hybrid Spatial-Keyword Index 23

Figure 3.3 MapReduce design of sksOpen 25

Figure 3.4 Loading Process 26

Figure 3.5 Visualization of a Hotels' query results 27

Figure 3.6 Interactive list Visualization 28

Figure 3.7 Visualization of Block-group Median Income query 30

Figure 3.8 Query result of properties data in an unfriendly database 31

Figure 3.9 TerraFly Visualization of Query results 32

Figure 4.1 Population Total (height) vs. Density (color) of US 37

Figure 4.2 The Architecture of TerraFly GeoCloud 39

Figure 4.3 The Workflow of TerraFly GeoCloud Analysis 40

Figure 4.4 Interface of TerraFly GeoCloud 41

Figure 4.5 Spatial Data Visualization: Point Data 42

Figure 4.6 Spatial Data Visualization: Polygon Data 43

Figure 4.7 Average properties price by zip code in Miami 45

Figure 4.8 Properties value in Miami 46

Figure 4.9 DBSCAN clustering on the crime data in Miami 47

Figure 4.10 Kriging data of the water level in Florida 48

Figure 4.11 MapQL implementation 49

Figure 4.12 Query data near the point 51

Figure 4.13 Query data along the line 52

Figure 4.14 Traffic of Santiago 53

Figure 4.15 Income at New York 54

Figure 4.16 Data Set Upload and Visualization 56

Figure 4.17 Properties in Miami 57

Figure 4.18 MapQL results 58

Figure 4.19 The flow path of Erik case 58

Figure 4.20 Datasets in TerraFly GeoCloud 59

Figure 4.21 Lung cancer disease map 60

Figure 4.22 P-value map of Local Moran I 61

Figure 4.23 Spatial auto-regression of lung cancer mortality and median income 62

Figure 5.1 Web enabled Map Service 68

Figure 5.2 v-TerraFly system 70

Figure 5.3 TerraFly System Workload Hourly Distribution 73

Figure 5.4 Performance and Resource cost comparison using different deployment schemes 74

Figure 5.5 Autonomic resource management system for v-TerraFly 77

Figure 5.6 Reader tier profiling 81

Figure 5.7 Reader tier profiling 82

Figure 5.8 Loader tier CPU usage profiling 83

Figure 5.9 Error and Standard deviation of different workload prediction approaches 87

Figure 5.10 Result: Response time 88

Figure 5.11 Result: VM nodes cost by hours and total VM nodes Cost 88

Figure 5.12 Result: response time improvement 90

Figure 5.13 Result: Throughput improvement..... 91

Figure 5.14 Node Allocation per Hour 91

Figure 5.15 Result: QoS value improvement..... 92

CHAPTER 1

INTRODUCTION

1.1 Motivation

With the exponential growth of the World Wide Web, there are many domains, such as water management, crime mapping, disease analysis, and real estate, open to Geographic Information System (GIS) applications. The Web can provide a giant amount of information to a multitude of users, making GIS available to a wider range of public users than ever before. Web-based map services are the most important application of modern GIS systems. For example, Google Maps currently has more than 350 million users. There are also a rapidly growing number of geo-enabled applications which utilize web map services on traditional computing platforms as well as the emerging mobile devices.

More people employ Web applications to update their geographical information via the process known as Geotagging. Geotagging can help users find a wide variety of location-specific information. For example, one can find images taken near a given location by entering latitude and longitude coordinates into a suitable image search engine [11]. Geotagging-enabled information services can also potentially be used to find location-based news, websites, and other resources. Geotagging can tell users the location of the content of a given picture or other media, and conversely on some media platforms, show media relevant to a given location [12].

However, it is quite challenging for users to manipulate spatial data. On one hand, typical geographic visualization tools do not offer spatial data index functions or application programming interfaces (API) to the public. On the other hand, even if users

have access to spatial data index services, it is very difficult to get the visualization of query results of their own spatial data.

At the same time, it is also challenging for the end users to quickly understand and analyze the spatial data, and to efficiently share their own data and analysis results with others. First, typical geographic visualization tools are complicated and fussy with a lot of low-level details, thus they are difficult to use for spatial data analysis. Second, the analysis of large amount spatial data is very resource-consuming. Third, current spatial data visualization tools are not well integrated for map developers, and it is difficult for end users to create the map applications on their own spatial datasets [10].

At last, virtual machines (VM) are powerful platforms for hosting web map service systems. But due to the highly complex and dynamic nature of web map service systems, it is challenging to efficiently host them using virtualized resources. First, typical web map services have to serve dynamically changing workloads, which makes it difficult to host map services on shared resources without compromising performance or wasting resources. Second, a web map service often consists of several tiers which have different intensive resource needs and result in dynamic internal resource contention. Third, for a typical web map service, both response time for requests and the freshness of the returned data are critical factors of the Quality of Service (QoS) required by users.

To address the above challenges, we need a search engine which opens to public and provide easy and clear visualization; and we need a spatial analysis platform to provide user with in needed analysis tools and data; and at last, we need powerful background support with virtualized hosted system which can automatically optimize the QoS while minimizing the resource cost [13][14].

1.2 My work

Motivated by the above challenges, my work concentrate in three directions: sksOpen, GeoCloud and v-TerraFly. Putting them together, we have a better solution of web based GIS system.

1.2.1 sksOpen

TerraFly sksOpen is an efficient online indexing, querying, and visualization system for Big Geospatial Data, which allows users to easily create indices of spatial objects and to query and visualize the results and share them via unique URLs. The TerraFly sksOpen Online Spatial Object Index and Visualization System is built using TerraFly Maps API, and JavaScript TerraFly API add-ons in a high performance cloud environment.

sksOpen, with MapReduce, improved a distributed disk-resident hybrid index for efficiently answering k-NN queries with Boolean constraints on textual content. With this algorithm, sksOpen have implemented efficient online indexing, querying, and visualization system for Big Geospatial data, to allow online users to index their own spatial data, and offer query visualization. Our experimental study showed an improved performance and scalability on large spatial datasets over alternate methods, and a better interactive user interface.

1.2.2 GeoCloud

TerraFly GeoCloud is an online spatial data analysis and visualization system, which allows end users to easily visualize and share various types of spatial data. First, TerraFly GeoCloud can accurately visualize and manipulate point and polygon spatial

data with just a few clicks. Second, TerraFly GeoCloud employs an analysis engine to support the online analysis of spatial data, and the visualization of the analysis results. Many different spatial analysis functionalities are provided by the analysis engine. Third, based on the TerraFly map API, TerraFly GeoCloud offers a MapQL language with SQL-like statements to execute spatial queries, and render maps to visualize the customized query results.

TerraFly GeoCloud online spatial data analysis and visualization system is built upon the TerraFly system using TerraFly Maps API and JavaScript TerraFly API add-ons in a high performance cloud Environment. The function modules in the analysis engine are implemented using C and R language and python scripts. Comparing with current GIS applications, our system is more user-friendly and offers better usability in the analysis and visualization of spatial data. The system is available at http://terrafly.fiu.edu/GeoCloud/.

1.2.3 v-TerraFly

Firstly, v-TerraFly can accurately predict the workload demands of a web map service online based on a novel two-way forecasting algorithm that considers both historical hourly patterns and daily patterns. Secondly, based on the predicted workload, v-TerraFly can automatically estimate the resource demands of its various tiers based on performance profiles created using machine learning techniques. Thirdly, v-TerraFly employs a new QoS model that captures the balance between response time and data freshness and uses this model to automatically optimize the resource allocation of a web map service system.

VMs support flexible resource allocation to both meet web map services system demands and share resources with other applications. Virtualization is also enabling technology for the emerging cloud computing paradigm, which further allows highly scalable and cost-effective web map services hosting, leveraging its elastic resource availability and pay-as-you-go economic model.

This proposed v-TerraFly system is realized on Hyper-V virtual machine environments and evaluated by experiments using real workloads collected from the production version of TerraFly system. The results show that the proposed two-level workload prediction method outperforms traditional exponential smoothing prediction by 18.91%, and the system improves the QoS by 26.19% compared to traditional statically node allocation. In the meantime, it saves resource usages by 20.83% compared to traditional peak-load-based resource allocation.

1.3 Dissertation Outline

The remainder of the dissertation is organized as follows. Chapter 2 discusses research problem and related works. Chapter 3 describes the sksOpen system which is an efficient Indexing, Querying and Visualization of Geo-spatial Data tool. Chapter 4 discusses TerraFly GeoCloud online spatial analysis and visualization system. Chapter 5 describes the v-TerraFly system, autonomic Resource Management for Virtualized Web Map. Chapter 6 summarizes the conclusions and provides recommendations for future research.

CHAPTER 2

LITERATURE REVIEW

In this thesis, we study and propose solution to three related research problems in geospatial databases:

1. Parallel construction of R-tree and inverted file index on large spatial databases, and provide efficient online querying, and visualization.
2. Efficient online spatial data analysis, visualization and sharing.
3. Accurately predict the workload demands of a web map service and automatically optimize the resource allocation of a web map service system

In this chapter we describe the most relevant literature. We first show existing approaches on K-NN search and visualization, and then we describe current and related problems on online spatial data analysis. Finally, we survey existing work in workload prediction and autonomic computing on web map service.

2.1 Geographic information retrieval systems

The R-tree traversal method in our work is inspired in Hjaltason and Samet's [16] incremental top-k nearest neighbor algorithm using R-trees [17]. Performance improvements on the original R-tree work have been proposed, e.g. R*-tree [18], R+-tree [19], and Hilbert R-tree [20]. Any of these variants can replace the Rtree index used in the proposed hybrid spatial keyword index without modifying our search algorithms. In information retrieval, inverted files are arguably the most efficient index structure for free-text search [21][22].

There has been lot of interest in building geographic information retrieval systems. The first work of this kind started in the context of digital library (DL) projects such as

GIPSY at UC Berkeley and Alexandria Digital Library Project at UC Santa Barbara [23]. In these projects, the main objective is to address the extraction of geographic references found in the text by using ontologies, gazetteers, thesaurus, etc., and convert them to coordinates for retrieving DL contents using geography.

In the context of geographic search engines, there are numerous academic projects. Most of them can be broadly classified under 1) work that focused on extraction of geographic references from documents and/or 2) efficient query processing. We will briefly describe a few of these. In GeoSearch System [24], the geographic scope ofWeb pages are extracted by analyzing the geographic references in text as well as the geographic location where the Web sites are registered. In [25], the focus is on improving the extraction techniques. In particular, after the relevant geographic references are extracted, ambiguities such as multiple place name references and alternate place names are resolved using techniques such as geo-matching and geo-propagation. Other relevant studies that addressed geographic search on the Web is [24].

In the context of query processing for GIR, indexing techniques for processing text and geographic data are the main focus. In [24], a simple inverted index structure for text and grid file for geographic data are used. They propose a hybrid index structure in which each keyword is combined with different partitions of space. In effect what they are proposing is similar to [26]. The other technique proposed in their work concatenates keyword with region identifier. For example, keyword earthquake is combined with spatial region “R1” and represented as spatial-textual key “earthquakeR1”. All the documents that are in “R1” and contains the text earthquake are attached as list to the key “earthquakeR1”. There are drawbacks of this approach. First, for large set of objects, this

approach will generate a large number of false positives. For query containing multiple keywords and spatial region, a number of such keys have to be looked up and filtered.

In a recent work, the authors propose to maintain individual indices for spatial and textual data. They propose various approaches to retrieve data from each index before the final merging of results. The spatial objects indexed in their applications are complex footprints that are extended regions in space. They approximate them by using MBRs and use memory-resident spatial index. Their approach does not scale well with increasing size of the dataset. To alleviate the problem, they propose to compress the MBRs, but the attempt generates large candidate set that needs to be fetched from the disk, with a high rate of false positives. This will become a major performance bottleneck for large scale GIR applications. In our work, we use disk-resident spatial index for GIR applications. Our data structure performs significantly better than their approach with respect to two aspects: 1) first it reduces the number of disk accesses in identifying the candidate objects and as a consequence 2) it reduces the overhead in merging the candidate objects [27].

In another much related work [26], the authors proposed a hybrid index by combining the spatial and inverted list structures. Their approaches either use multiple R*-trees to answer queries or generates more candidates for further filtering [27].

The problem of retrieving spatial objects satisfying non-spatial constraints has been studied in the recent past. Park and Kim [28] proposed RS-trees, a combination of R-trees and signature trees for attributes with controlled cardinality; signature chopping is suggested to mitigate combinatorial errors [29] (database overrepresentation) of superimposed signatures. Harinharan et al. [27] proposed to include a list of terms in every node of an R-tree. De Felipe et al. [30] augmented signature files in R-tree nodes

with similar constraints as [28]. Recently, Cong et al. [31] augmented an inverted file in every node of an R-tree, and used a ranking function that combines spatial proximity and text relevancy. Our work differs in that we assume distance as ranking score, and we focus on efficiently processing Boolean constraints on textual data. Further, none of the previous works offer efficient processing of the complement logical operator, which limits their applicability to the k-SB queries we considered in this work. Likewise, modern Web search engines, like Google and Yahoo!, offer Local Search services. Advanced querying options are provided to include and exclude certain terms from the search result. These are similar to the k-SB queries we consider. However, specific search algorithms are kept confidential by their owning companies [32]. Our approach combines modified versions of R-trees and inverted files to achieve effective pruning of the search space [32], with an extra quad-tree index to implement MapReduce.

2.2 Spatial data analysis and visualization

In the geospatial discipline, web-based GIS services can significantly reduce the data volume and required computing resources at the end-user side [33][34]. To the best of our knowledge, TerraFly GeoCloud is one of the first systems to study the integration of online visualization of spatial data, data analysis modules and visualization customization language.

The principles behind interactive spatial data analysis can be traced back to the work on dynamic graphics for data analysis in general, originated by the statistician John Tukey and a number of research groups at AT&T Bell Laboratories. An excellent review of the origins of these ideas is given in the collection of papers edited by Cleveland and

McGill (1988), and early discussions of specific methods are contained in the papers by, among others, Becker et al (1987), Becker and Cleveland (1987), and Stuetzle (1987). More recent reviews of methods for the dynamic analysis of high-dimensional multivariate data and other aspects of interactive statistical graphics can be found in papers by, among others, Becker et al (1996), Buja et al (1991, 1996), Cleveland (1993), and Cook et al (1995) [35].

Dynamic graphical methods started as enhancements to the familiar static displays of data, by allowing direct manipulation by the user that results in 'immediate' change in a graph. This had become possible by the availability of workstations with sufficient computational power to generate the statistical graphs without delays and to allow interaction with the data by means of an input device (light pen or mouse). The overall motivation was to involve the human factor more directly in the exploration of data (i.e. exploiting the inherent capabilities of the brain to detect patterns and structure), and thereby gain richer insights than possible with the traditional rigid and static display. This was achieved by allowing the user to delete data points, highlight (brush) subsections of the data, establish links between the same data points in different graphs, and rotate, cut through, and project higher-dimensional data. Furthermore, the user and not a preset statistical procedure determined which actions to perform. Interactive statistical procedures become particularly effective when datasets are large (many observations) and high-dimensional (many variables), situations where characterization of the data by a few numbers becomes increasingly unrealistic (for an early assessment see, for example, Andrews et al 1988: 75). While dynamic graphics for statistics were originally mostly experimental and confined to research environments, they have quickly become

pervasive features of the EDA capability in modern commercial statistical software packages [35].

Various GIS analysis tools are developed and visualization customization languages have been studied in the literature. ArcGIS is a complete, cloud-based, collaborative content management system for working with geographic information. But systems like ArcGIS and Geoda focus on the content management and share, not online analysis [36][37]. Azavea has many functions such as optimal Location find, Crime analysis, data aggregation and visualization. It is good at visualization, but has very limited analysis functions [38].

Various types of solutions have been studied in the literature to address the problem of visualization of spatial analysis [37]. However, on one hand, good analysis visualization tools like Geoda and ArcGIS do not have online functions. To use them, users have to download and install the software tools, and download the datasets. On the other hand, good online GIS systems like Azavea, SKE, and GISCloud have limited analysis functions. Furthermore, none of above products provides a simple and convenient way like MapQL to let user create their own map visualization [39][40]. Our work is complementary to the existing works and our system also integrates the data mining and visualization.

2.3 Workload prediction and resource management

In the geospatial discipline, web-based map services can significantly reduce the data volume and required computing resources at the end-user side [45]. To the best of our knowledge, v-TerraFly is the first to study the virtualization of typical web map

services and propose QoS-driven resource management for a virtualized web map service through workload forecasting and dynamic resource allocation [46].

Similarly to the birth of the Internet, one of the notable early self-managing projects was initiated by DARPA for a military application in 1997. The project was called the Situational Awareness System1 (SAS), which was part of the broader Small Units Operations (SUO) program. Its aim was to create personal communication and location devices for soldiers on the battlefield. Soldiers could enter status reports, for example, discovery of enemy tanks, on their personal device, and this information would autonomously spread to all other soldiers, which could then call up the latest status report when entering an enemy area. Collected and transmitted data includes voice messages and data from unattended ground sensors and unmanned aerial vehicles. These personal devices have to be able to communicate with each other in difficult environmental conditions, possibly with enemy jamming equipment in operation, and must at the same time minimize enemy interception to this end [47]. The latter point is addressed by using multihop ad-hoc routing; that is, a device sends its data only to the nearest neighbors, which then forward the data to their own neighbors until finally all devices receive the data. This is a form of decentralized peer-to-peer mobile adaptive routing, which has proven to be a challenging self-management problem, especially because in this project the goal is keep latency below 200 milliseconds from the time a soldier begins speaking to the time the message is received. The former point is addressed by enabling the devices to transmit in a wide band of possible frequencies, 20–2,500 MHz, with bandwidths ranging from 10 bps to 4 Mbps. For instance, when distance to next soldier is many miles, communication is possible only at low frequencies, which results in low

bandwidth, which may still be enough to provide a brief but possibly crucial status report. Furthermore, there may be up to 10,000 soldiers on the battlefield, each with their own personal devices connected to the network [47].

In 2001, IBM suggested the concept of autonomic computing. In their manifesto [48], complex computing systems are compared to the human body, which is a complex system, but has an autonomic nervous system that takes care of most bodily functions, thus removing from our consciousness the task of coordinating all our bodily functions. IBM suggested that complex computing systems should also have autonomic properties, that is, should be able to independently take care of the regular maintenance and optimization tasks, thus reducing the workload on the system administrators. IBM also distilled the four properties of a self-managing (i.e., autonomic) system: self-configuration, self-optimization, self-healing, and self-protecting.

Finally, we would like to mention an interesting project that started at NASA in 2005, the Autonomous Nanotechnology Swarm (ANTS). As an exemplary mission, they plan to launch into an asteroid belt a swarm of 1000 small spacecraft (so-called pico-class spacecraft) from a stationary factory ship in order to explore the asteroid belt in detail. Because as much as 60–70% of the swarm is expected to be lost as they enter the asteroid belt, the surviving craft must work together. This is done by forming small groups of worker craft with a coordinating ruler, which uses data gathered from workers to determine which asteroids are of interest and to issue instructions. Furthermore, messenger craft will coordinate communications between members of the swarm and with ground control. In fact, NASA has already previously used autonomic behavior in its DS1 (Deep Space 1) mission and the Mars Pathfinder [49]. Indeed, NASA has a

strong interest in autonomic computing, in particular in making its deep-space probes more autonomous. This is mainly because there is a long round-trip delay between a probe in deep space and mission control on Earth. So, as mission control cannot rapidly send new commands to a probe—which may need to quickly adapt to extraordinary situations—it is extremely critical to the success of an expensive space exploration mission that the probes be able to make certain critical decisions on their own.

Various automatic forecasting algorithms have been studied in the related work [50], including different kinds of exponential smoothing. The work of Brown (1959) and Gardner (1985) led to the use of exponential smoothing in automatic forecasting (e.g., Stellwagen &Goodrich, 1999) [42][43][44]. Hyndman (2002) developed a more general class of methods with a uniform approach to calculate the prediction interval [41][42]. The workload prediction algorithm proposed in this paper is based on exponential smoothing, but it is novel in the use of two levels of double exponential smoothing to capture both hourly pattern and daily pattern in the workload, which achieves much higher accuracy than traditional exponential smoothing methods.

In particular, Dinda *et al*. studied prediction-based best-effort real-time service to support distributed, interactive applications in shared computing environments. Two of the examples are an earthquake visualization tool and a GIS map display tool, which were shown to benefit from the service [51]. However, the workload prediction is based on linear prediction which is often not sufficient for real-world dynamic workloads. In this paper, we proposed a two-level exponential smoothing algorithm which shows good prediction accuracy for real TerraFly workloads.

Various types of solutions have been studied in the literature to address the problem of autonomic VM resource management. Different machine learning algorithms have been considered to model VM resource usages [52][53][54][55]. Feedback control theory has also been used to adjust VM resource allocations, which are often based on models trained to identify the system and build the controller [56][57][58][59]. These various solutions are complementary to this paper's work which focuses on the management of virtualized web map services. Meanwhile, this paper proposes a unique QoS model to capture multiple important objectives and a new method to optimize resource allocation across multiple competing tiers, which has not been studied in the related work and can be applied to manage other multi-tier applications with similar characteristics.

CHAPTER 3

sksOpen: Efficient Indexing, Querying and Visualization of Geo-spatial Data

With the fast growing use of web-based map services, the performance of indexing and querying of location-based data is becoming a critical quality of service aspect. Spatial indexing is typically time-consuming and is not available to end-users. To address this challenge, we have developed and open-sourced an Online Indexing and Querying System for Big Geospatial Data, sksOpen. Integrated with the TerraFly Geospatial database [1], TerraFly sksOpen is an efficient indexing and query engine for processing Top-k Spatial Boolean Queries. Further, we provide ergonomic visualization of query results on interactive maps to facilitate the user's data analysis.

3.1 Introduction

With the exponential growth of Internet applications, there are many domains open to Geographic Information System (GIS) applications. Massive amounts of spatial information become available to a wide range of public uses [10]. More and more people employ Web applications to update their geographical information via the process known as Geotagging. For example, Google Maps currently has more than 350 million users. There are also a rapidly growing number of geo-enabled applications, which utilize web map services on traditional computing platforms as well as on emerging mobile devices.

Geotagging can help users find a wide variety of location-specific information. For example, one can find images taken near a given location by entering latitude and longitude coordinates into a suitable image search engine [11]. Geotagging-enabled information services can also potentially be used to find location-based news, websites,

and other resources. Geotagging can tell users the location of the content of a given picture or other media, and conversely on some media platforms, show media relevant to a given location [12].

However, due to the highly complex and dynamic nature of GIS systems, it is quite challenging for users to manipulate spatial data. On one hand, typical geographic visualization tools do not offer spatial data index functions or application programming interfaces (API) to the public. On the other hand, even if users have access to spatial data index services, it is very difficult to get the visualization of query results of their own spatial data.

To address the above challenges, we have developed TerraFly sksOpen, an efficient online indexing, querying, and visualization system for Big Geospatial Data, which allows users to easily create indices of spatial objects and to query and visualize the results and share them via unique URLs.

The TerraFly sksOpen Online Spatial Object Index and Visualization System is built using TerraFly Maps API, and JavaScript TerraFly API add-ons in a high performance cloud environment.

3.2 Background

3.2.1 TerraFly

TerraFly is a system for querying and visualizing geospatial data developed by the High Performance Database Research Center (HPDRC) lab at Florida International University (FIU) [1-9]. The TerraFly system serves worldwide web map requests to over 125 countries and regions, providing users with customized aerial photography, satellite

imagery, and various overlays, such as street names, roads, restaurants, services and demographic data [60][61].

The TerraFly API allows rapid deployment of interactive web applications, and has been used to produce systems for disaster mitigation, ecology, real estate, tourism, and municipalities. TerraFly's web-based client interface is accessible from anywhere, via any standard web browser, with no client software to install [62][63].

TerraFly allows users to virtually ‘fly’ over enormous geographic information simply via a web browser with several advanced functionalities and features, such as user-friendly geospatial querying interfaces, map display with user-specific granularity, real-time data suppliers, demographic analysis, annotation, route dissemination via autopilots, API for web sites, etc [64][65].

TerraFly's server farm ingests geo-locates, cleanses, mosaics, and cross-references 40TB of base map data and user-specific data streams. The 40TB TerraFly data collection includes, among others, 1-meter aerial photography of almost the entire United States, and 3-inch to 1-foot full-color recent imagery of major urban areas. TerraFly’s vector collection includes 400 million geo-located objects, 50 billion data fields, 40 million polylines, 120 million polygons, including: all US and Canada roads, US Census demographic and socioeconomic datasets, 110 million parcels with property lines and ownership data, 15 million records of businesses with company stats and management roles and contacts, 2 million physicians with expertise detail, various public place databases (including the USGS GNIS and NGA GNS), Wikipedia, extensive global environmental data (including daily feeds from NASA and NOAA satellites and the USGS water gauges), and hundreds of other datasets [66][67].

3.2.2 Spatial data visualization

Information visualization (or data visualization) techniques are able to present the data and patterns in a visual form that is intuitive and easily comprehensible, allowing users to derive insights from the data, and support user interactions.

Visualizing the objects in geo-spatial data is as important as the data itself. The visualization task becomes more challenging as both the data dimensionality and richness in the object representation increases. In TerraFly data querying we have addressed the visualization challenge, including the interactive map visualization spatial data and interactive list visualization [68].

3.2.3 MapReduce

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without much experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

Hadoop, a version of MapReduce, is an open-source software framework that supports data-intensive distributed applications [69]. It is this programming paradigm that

allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce concept is fairly simple to understand for those who are familiar with clustered scale-out data processing solutions [70].

The term MapReduce denotes the two main tasks that Hadoop programs perform. The first task, Map, takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The Reduce task takes the output from a Map as input and combines those data tuples into a smaller set of tuples [71].

3.2.4 K-NN

In pattern recognition, the k-nearest neighbor algorithm (k-NN) is a non-parametric method for classifying objects based on closest training examples in the feature space. K-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification [72]. The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor [72].

Nearest neighbor rules in effect implicitly compute the decision boundary. It is also possible to compute the decision boundary explicitly, and to do so efficiently, so that the computational complexity is a function of the boundary complexity [73].

3.3 Architecture of sksOpen

TerraFly sksOpen is implemented in Java, and is a web service easily accessible from anywhere. In this section, we will introduce the algorithm and software structure of sksOpen.

3.3.1 The index algorithm of sksOpen

We improved the spatial object index algorithm developed by Cary, Rishe et al in 2010 [32]. The algorithm creates spatial object indices as a hybrid index; it includes both an R-Tree spatial index and an inverted text file index. We have added a new "map" algorithm to split the data set in order to speed up the index to fit large-scale spatial data index [74].

By employing this hybrid index, we attained fast retrieval, even when matching objects were located far away from one another, efficiently filtering-out of objects not satisfying the query Boolean constraints on keywords, and maintained low storage requirements while keeping high query performance.

The challenge is reducing the computations to eliminate as many non-candidate objects as possible. In particular, NOT-semantics constraints may substantially shrink the output size and lead to unnecessary scans.

The indexing approach leverages the strengths of R-trees in spatial search, and modifies an inverted file for efficient processing of Boolean constraints. The combination of indexing techniques yields the hybrid data structure: Spatial-Keyword Index (SKI) [32].

Next, we define the principal terminology in SKI [32], which invented by Cary, A., Wolfson, O., & Rishe, N. (2010, January). Efficient and scalable method for processing top-k spatial boolean queries:

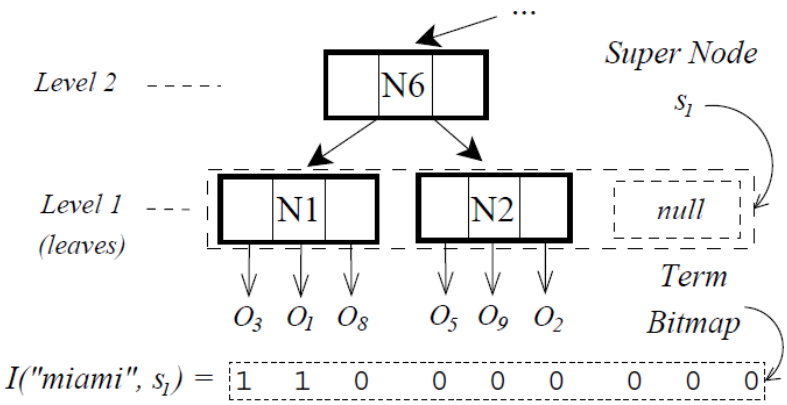


Figure 3.1 An super-node and leaf nodes [32]

R-tree Index (*R*): A modified R-tree built with spatial attributes. Entries in *R*'s inner nodes are augmented with index ranges [*a, b*], where S_a and S_b are the left-most and right-most, respectively, super nodes contained in the sub tree rooted at node entry. Ranges in leaf-node entries contain a single value, the index of the super node containing the leaf node.

Spatial Inverted File (SIF): A modified inverted file constructed on a vocabulary *V*. The Lexicon contains terms in *V* and their document frequencies (*df*). Posting lists are modified to include spatial information from *R*. Specifically, the posting list of a term *t* contains all its term bitmaps sorted by the super node index as follows:

Posting(t) = [$I(t, s_1)$, $I(t, s_2)$, ...] where S_i Belongs to $S(R)$

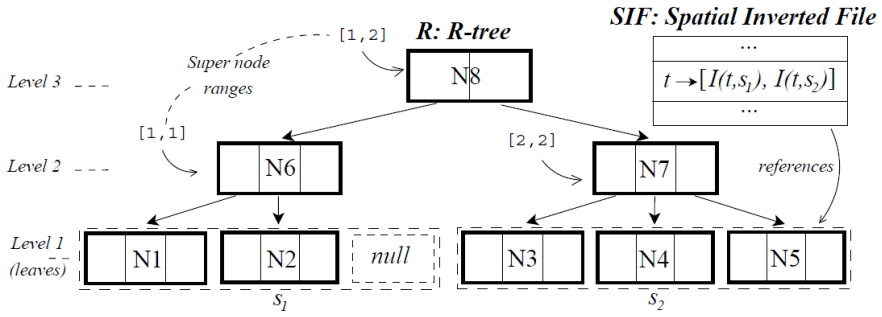


Figure 3.2 Hybrid Spatial-Keyword Index [32]

We organized posting elements in a B+tree to allow fast random and range retrieval. Figure 3.2 shows the structure of the hybird index [75][76].

However, this algorithm whichi invited by Cary, A is indexing the tuples in a particular sequence, which is not feasible in a large-scale index enviorment. To address this challege, my work is added a new componet of the algorithm, named *split*, and merge modules to split the input data set quickly into different parts, and finally merge all indices, to facilate multi-core or multi-machine index loading, to significantly increase the performance of the algorithm [77][78].

We employed a Z-order value to quickly set the split points. In mathematical analysis, Z-order, Morton order, or Morton code, is a function that maps multidimensional data to one dimension, while preserving locality of the data points. It was introduced in 1966 by G. M. Morton [79]. The Z-value of a point in a multidimensional space is calculated by interleaving the binary representations of its coordinate values. Once the data is sorted into this ordering, any one-dimensional data structure can be used such, as binary search trees, B-trees, skip lists, or hash tables. The resulting ordering can equivalently be described as the order one would get from a depth-first traversal of a quadtree; because of its close connection with quadtrees, the Z-

ordering can be used to efficiently construct quadtrees and related higher dimensional data structures.

The Split Algorithm works as follows:

1. Get the Z-order Value of a tuple of coordinate to get the Split points
2. Create a Split point array
3. For each entry, perform a binary search in the Split point array to find out the partition index
4. Write the entry into corresponding partition file
5. Send the partitioned files to a thread or a loading machine to start index loading

The Index Merge Algorithm works as follows:

1. Save Split point array and load it when querying
2. For each search point, perform a binary search in Split point array to find the partition index
3. Perform the query procedure in the corresponding spatial keywords index
4. Find the eligible entry and return a list of the results

With this improvement, we have added a Quadtree at the top of the R-Tree index to improve the performance of multi-task loading [81]. Because the partitions of the data file are easy to control, the depth of the Quadtree is usually short, wich means the binarySearch in Split point runs quickly and takes O(log(m)).

3.3.2 The Structure of sksOpen

With the improvement of methods for processing top-k spatial Boolean queries by introducing the Split and Merge modules, we can utilize the MapReduce model to create the sksOpen indices for Big Data. The performance of the loading of indices is significantly improved.

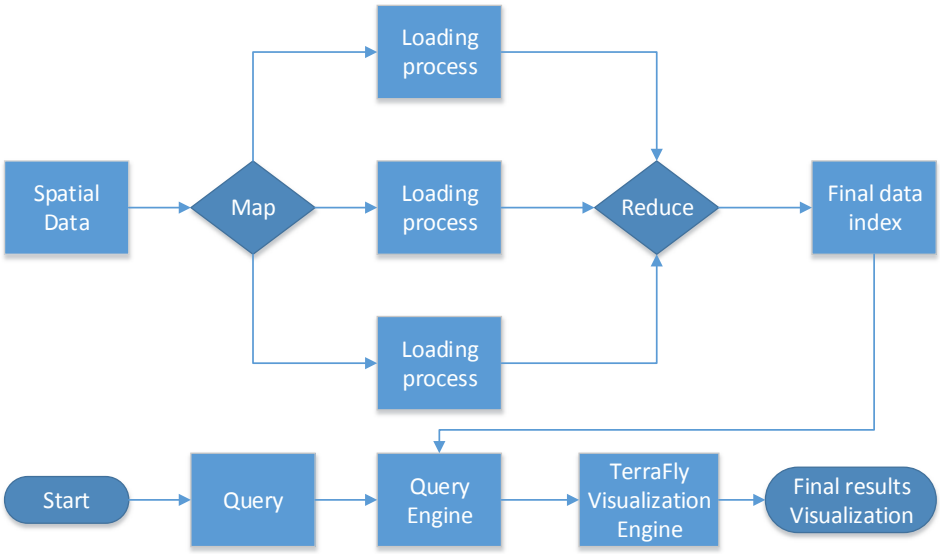


Figure 3.3 MapReduce design of sksOpen

As shown in Figure 3.3, the Map module splits the spatial data into partitions depending on how many hardware resources will be used for index loading. After each loading process is finished, the Reduce module will automatically merge the indices of the data partitions to produce the final data index. After that, the database can be efficiently queried [80]. When a query comes, the Query Engine will examine the final data index, and then produce the query results list. With the results list, the TerraFly visualization engine will offer visualization with a unique URL, which can be shared with other users.

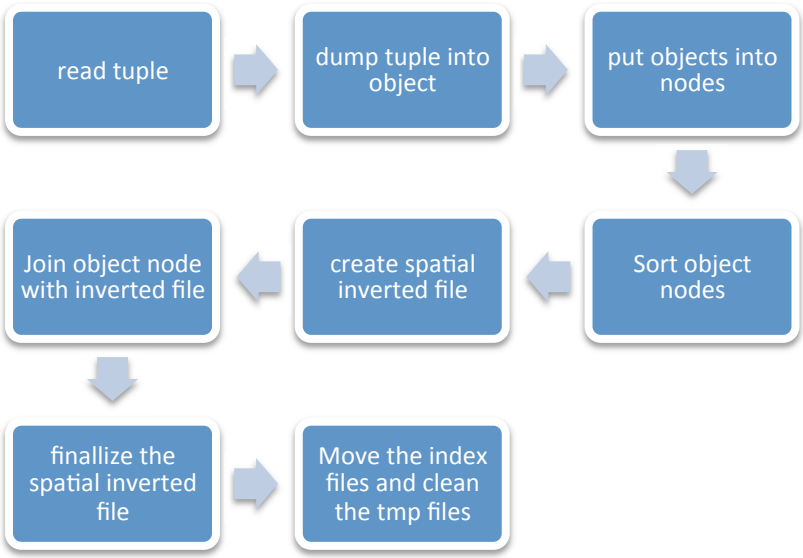


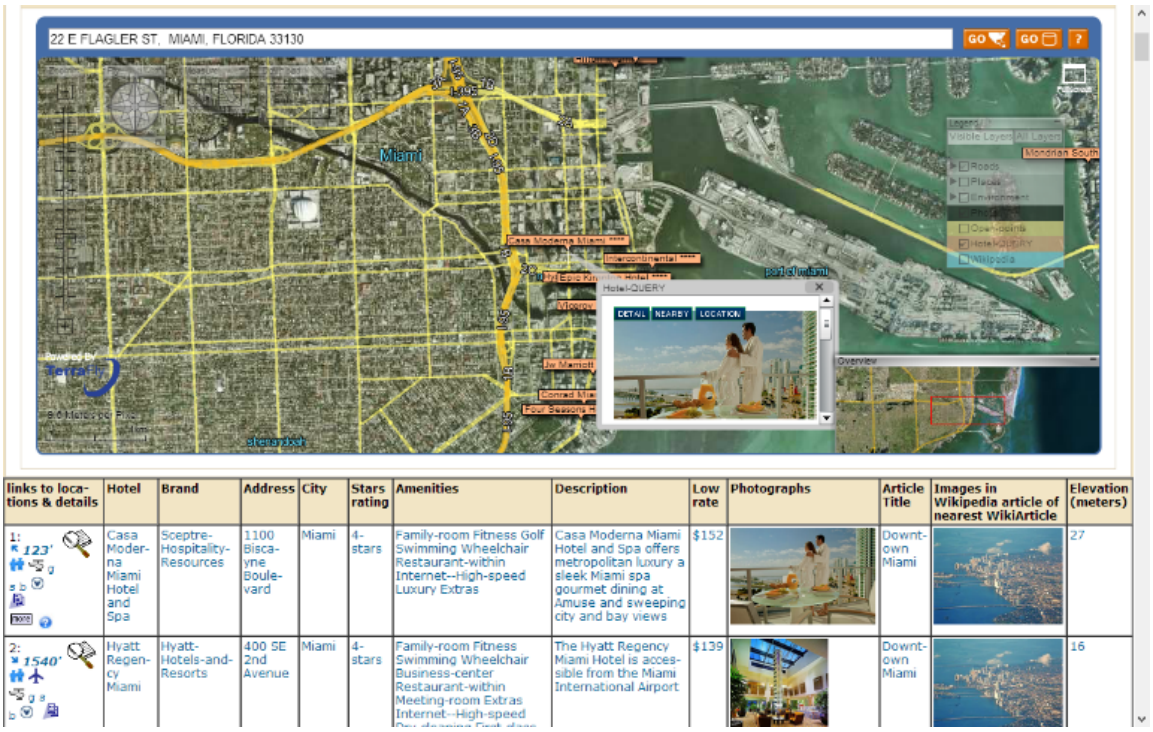
Figure 3.4 Loading Process

The loading process of sksOpen is one of the key modules. Figure 3.4 shows details of the loading process.

3.4 Visualization of sksOpen

For spatial object visualization, the system supports both map object visualization and data list object visualization. T visualization is dynamic and interactive.

Integrated with TerraFly map API and JavaScript, the query results of spatial object can be shown on a much better interface, including both map and object lists.



links to locations & details	Hotel	Brand	Address	City	Stars rating	Amenities	Description	Low rate	Photographs	Article Title	Images in Wikipedia article of nearest WikiArticle	Elevation (meters)
1: 123'	Casa Moderna Miami Hotel and Spa	Sceptre-Hospitality-Resources	1100 Biscayne Boulevard	Miami	4-stars	Family-room Fitness Golf Swimming Wheelchair Restaurant-within Internet--High-speed Luxury Extras	Casa Moderna Miami Hotel and Spa offers metropolitan luxury a sleek Miami spa gourmet dining at Amuse and sweeping city and bay views	$152		Downtown Miami		27
2: 1540'	Hyatt Regency Miami	Hyatt-Hotels-and-Resorts	400 SE 2nd Avenue	Miami	4-stars	Family-room Fitness Swimming Wheelchair Business-center Restaurant-within Meeting-room Extras Internet--High-speed	The Hyatt Regency Miami Hotel is accessible from the Miami International Airport	$139		Downtown Miami		16

Figure 3.5 Visualization of a Hotels' query results

Figure 3.5 shows visualization of a query of hotel information in Miami. When users query, for example, search for hotels of 4 stars or above and less than $200 per night near downtown Miami, the visualization of results will be shown as in Figure 3.5. The map on the top shows the location of the hotel results. When the mouse hovers over a hotel location, a popup appears with more detailed information. Below the map visualization, there is a table of results hyperlinked to further querying.

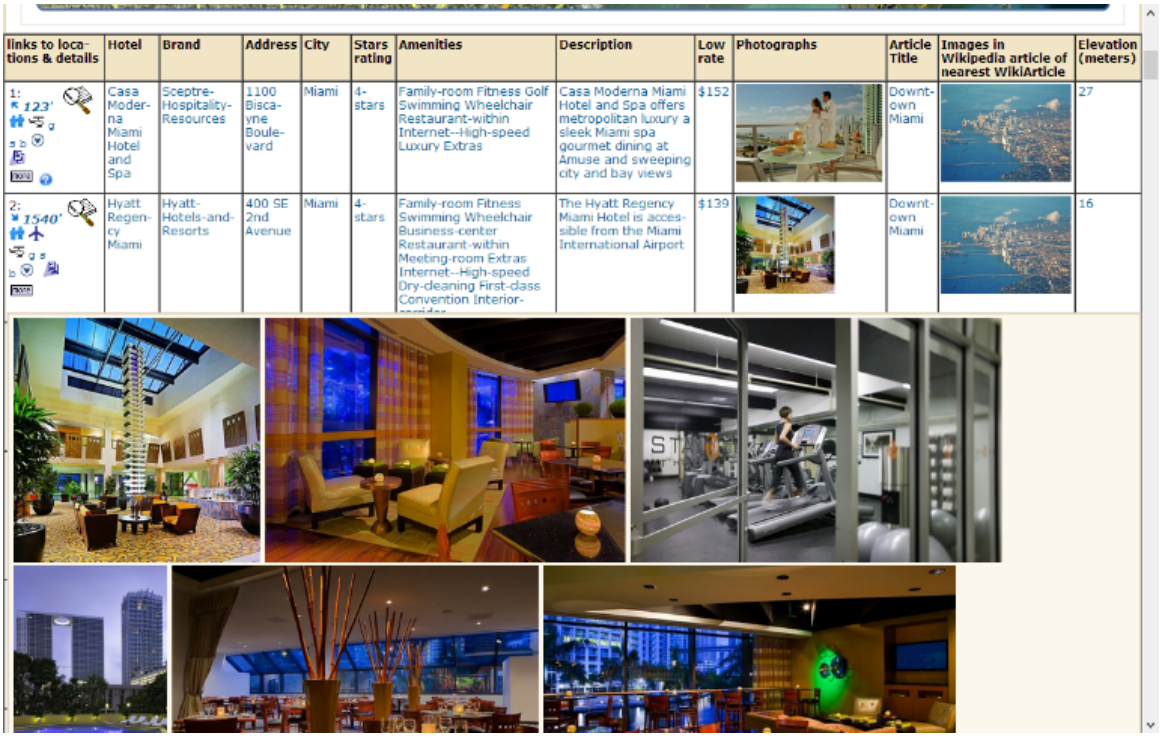


Figure 3.6 Interactive list Visualization

As shown in Figure 3.6, if the mouse hovers over any object, more data appear as a layer over the page.

3.5 Case study

3.5.1 Setup

This section evaluates the proposed sksOpen service system. As a typical web application, sksOpen provides a variety of web services via Apache Tomcat to serve online web requests. The test bed is set up on a Dell PowerEdge servers, each with XEON Intel E5520 2.27 GHz, 16GB (4x4GB) ECC -- DDR3 1066MHz, and one 1TB 7.2 RPM SAS disk. CentOS 5.6 are installed to provide the environment for sksOpen.

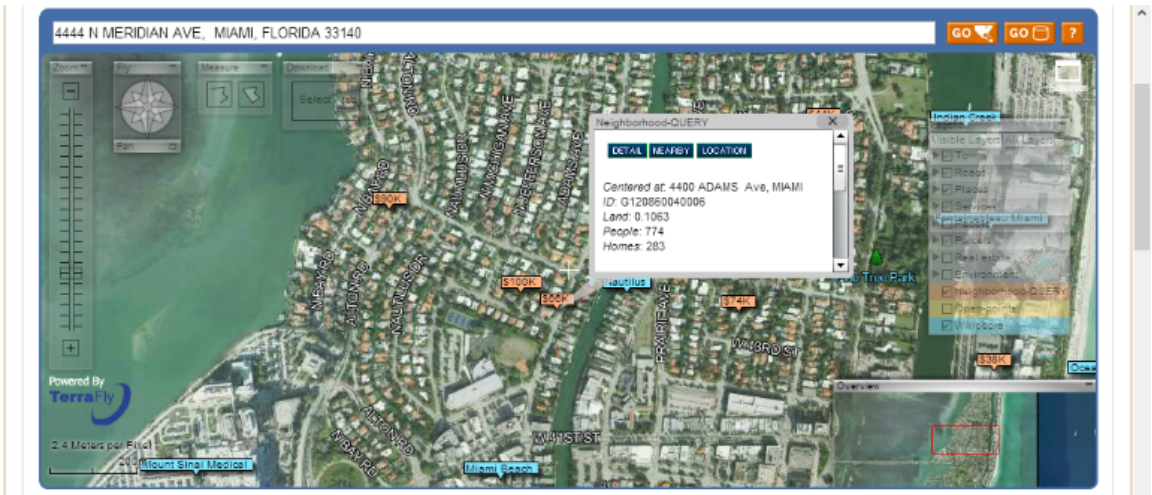
3.5.2 Performance study

The Test data file is “us_consumer_2012_full”. The data is Person Individual U.S. Consumers and White Pages, which is 68GB, 173million records, and for each records there are 136 fields. The loading process map the workload into 40 parts, and eventually takes 28 hours in to construct the final index, the loading process can be accelerated by adding computing resources.

For the query performance, we comparing KNN query and KNN query with Boolean restriction. For KNN query, query the top 50 records near Miami Beach, the query results as long as 38339 characters returned in 1.211971 seconds includes the disk access time for record retrieval. For KNN query Boolean restriction CITY=Miami and FIRST_NAME=jose, query the top 50 records near Miami Beach, the query results as long as 33308 characters returned in 1.707193 seconds includes the disk access time for record retrieval. Two Boolean restrictions just take 30% more query time, and comparing with most of Boolean queries which have to retrieve data respectively and then join with each other, this is a good performance.

3.5.3 User experience study

In this section, we present a case study on using TerraFly sksOpen for spatial data indexing, query, and visualization.

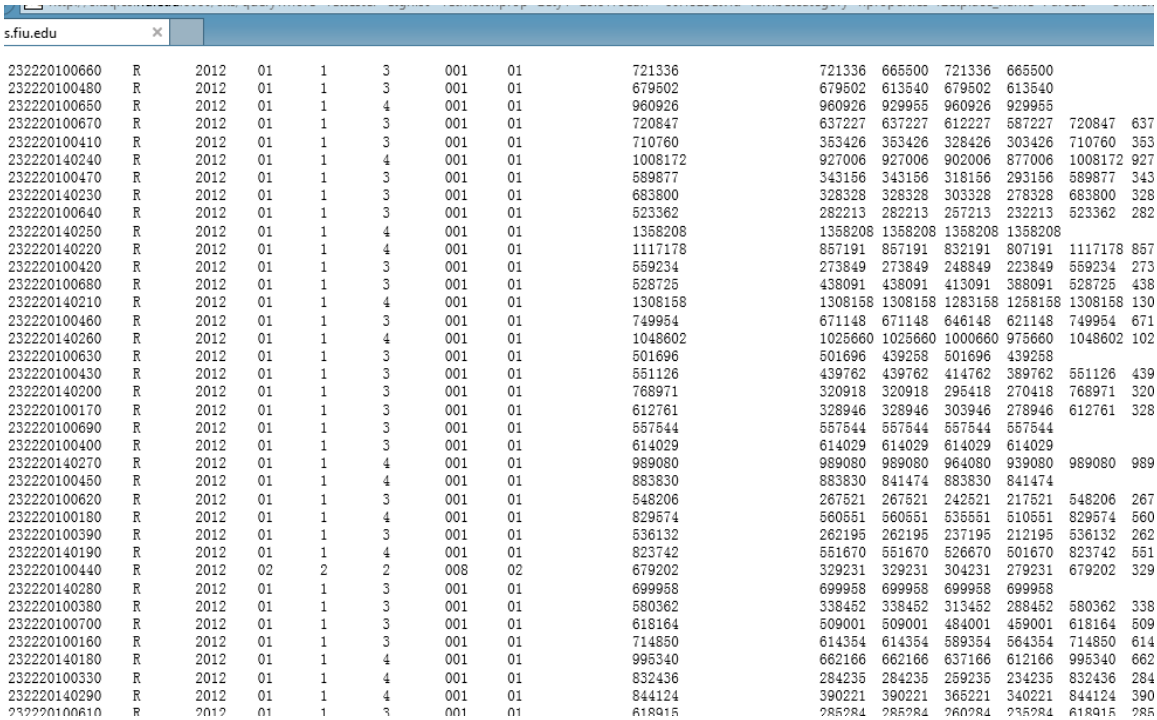


links to locations & details	Centered at	Land (sq mi)	Water (sq mi)	People	Homes (homes)	Age (years)	Adults (persons)	Family size (persons)	Home rooms (rooms)	Sum of income	Median income	Home value	Year built (year)	Rent	Owner % (%)	Vacant % (%)
1: 317'	4400 ADAMS Ave, MIAMI	0.1063		774	283	35.7	522	2.79	6.8	$32.52M	$66K	$320K	1954	$656	87.375	1.954
2: 400'	4301 N JEFFERSON Ave, MIAMI	0.0104		58	16	34	37	3.87	9.1	$1.61M	$101K	$421K	1942		100	
3: 1320'	4300 ROYAL PALM Ave,	0.1271	0.0175	713	275	37	516	2.77	7	$25.18M	$74K	$351K	1941	$715	70.12	5.993

Figure 3.7 Visualization of Block-group Median Income query

John wants to analyze the relationship between median income in locales and property values. John enters a TerraFly page presenting visualization of median income data of U.S. Census Block Groups, as shown in Figure 3.7. John notices a place near Miami (zip code 33140) that has a lower median average income than areas nearby. Then John wants to examine the property values of this location.

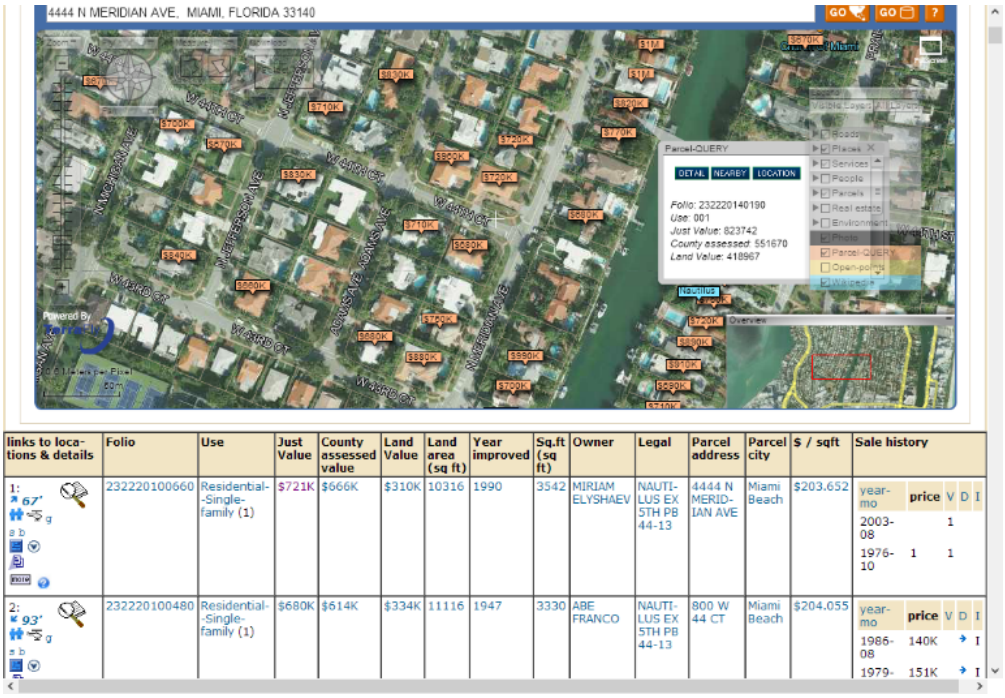
Although he has access to a data set of all Florida properties, it is too large to use directly. There are 10 million records in the data set, and each record has 173 fields. He decides to index the file by sksOpen, in order to search the property information near the place.



232220100660	R	2012	01	1	3	001	01	721336	721336	665500	721336	665500		
232220100480	R	2012	01	1	3	001	01	679502	679502	613540	679502	613540		
232220100650	R	2012	01	1	4	001	01	960926	960926	929955	960926	929955		
232220100670	R	2012	01	1	3	001	01	720847	637227	637227	612227	587227	720847	637
232220100410	R	2012	01	1	3	001	01	710760	353426	353426	328426	303426	710760	353
232220140240	R	2012	01	1	4	001	01	1008172	927006	927006	902006	877006	1008172	927
232220100470	R	2012	01	1	3	001	01	589877	343156	343156	318156	293156	589877	343
232220140230	R	2012	01	1	3	001	01	683800	328328	328328	303328	278328	683800	328
232220100640	R	2012	01	1	3	001	01	523362	282213	282213	257213	232213	523362	282
232220140250	R	2012	01	1	4	001	01	1358208	1358208	1358208	1358208	1358208		
232220140220	R	2012	01	1	4	001	01	1117178	857191	857191	832191	807191	1117178	857
232220100420	R	2012	01	1	3	001	01	559234	273849	273849	248849	223849	559234	273
232220100680	R	2012	01	1	3	001	01	528725	438091	438091	413091	388091	528725	438
232220140210	R	2012	01	1	4	001	01	1308158	1308158	1308158	1283158	1258158	1308158	130
232220100460	R	2012	01	1	3	001	01	749954	671148	671148	646148	621148	749954	671
232220140260	R	2012	01	1	4	001	01	1048602	1025660	1025660	1000660	975660	1048602	102
232220100630	R	2012	01	1	3	001	01	501696	501696	439258	501696	439258		
232220100430	R	2012	01	1	3	001	01	551126	439762	439762	414762	389762	551126	439
232220140200	R	2012	01	1	3	001	01	768971	320918	320918	295418	270418	768971	320
232220100170	R	2012	01	1	3	001	01	612761	328946	328946	303946	278946	612761	328
232220100690	R	2012	01	1	3	001	01	557544	557544	557544	557544	557544		
232220100400	R	2012	01	1	3	001	01	614029	614029	614029	614029	614029		
232220140270	R	2012	01	1	4	001	01	989080	989080	989080	964080	939080	989080	989
232220100450	R	2012	01	1	4	001	01	883830	883830	841474	883830	841474		
232220100620	R	2012	01	1	3	001	01	548206	267521	267521	242521	217521	548206	267
232220100180	R	2012	01	1	4	001	01	829574	560551	560551	535551	510551	829574	560
232220100390	R	2012	01	1	3	001	01	536132	262195	262195	237195	212195	536132	262
232220140190	R	2012	01	1	4	001	01	823742	551670	551670	526670	501670	823742	551
232220100440	R	2012	02	2	2	008	02	679202	329231	329231	304231	279231	679202	329
232220140280	R	2012	01	1	3	001	01	699958	699958	699958	699958	699958		
232220100380	R	2012	01	1	3	001	01	580362	338452	338452	313452	288452	580362	338
232220100700	R	2012	01	1	3	001	01	618164	509001	509001	484001	459001	618164	509
232220100160	R	2012	01	1	3	001	01	714850	614354	614354	589354	564354	714850	614
232220140180	R	2012	01	1	4	001	01	996340	662166	662166	637166	612166	996340	662
232220100330	R	2012	01	1	4	001	01	832436	284235	284235	259235	234235	832436	284
232220140290	R	2012	01	1	4	001	01	844124	390221	390221	365221	340221	844124	390
232220100610	R	2012	01	1	3	001	01	618915	285284	285284	260284	235284	618915	285

Figure 3.8 Query result of properties data in an unfriendly database

By triggering a URL to put the dataset into the sksOpen server, the loading process begins. With the help of MapReduce, sksOpen finishes loading in a couple of minutes. Then, John enters search conditions to finalize the query: properties near 33140 with pricing lower than 1M. Instead of the result shown in Figure 3.8, as most open index and query tools offer, John got a map visualization shown in Figure 3.9. John can change the query conditions to explore the data set as he desires. All the 173 fields of the data set can be queried.



links to locations & details	Folio	Use	Just Value	County assessed value	Land Value	Land area (sq ft)	Year improved	Sq.ft (sq ft)	Owner	Legal	Parcel address	Parcel city	$ / sqft	Sale history
1: 67'	232220100660	Residential--Single-family (1)	$721K	$666K	$310K	10316	1990	3542	MIRIAM ELYSHAEV	NAUTILUS EX 5TH PB 44-13	4444 N MERIDIAN AVE	Miami Beach	$203.652	year-mo: 2003-08, 1976-10; price: 1, 1; V: 1; D: 1; I: 1
2: 93'	232220100480	Residential--Single-family (1)	$680K	$614K	$334K	11116	1947	3330	ABE FRANCO	NAUTILUS EX 5TH PB 44-13	800 W 44 CT	Miami Beach	$204.055	year-mo: 1986-08, 1979-; price: 140K, 151K; I

Figure 3.9 TerraFly Visualization of Query results

3.6 Related work

Spatial object index, query, and visualization services, can significantly improve the data analysis efficiently. TerraFly sksOpen is one of the first systems open online that allows users to index their own data, and provide both interactive map and list visualization.

For the algorithm of processing Top-k Spatial Boolean queries, the R-tree traversal method in our work is inspired by Hjaltason and Samet's incremental top-k nearest neighbor algorithm using R-trees [17]. Performance improvements on the original R-tree work have been proposed, e.g. R*-tree, R+-tree, and Hilbert R-tree. Any of these variants can replace the R-tree index used in the proposed hybrid spatial keyword index without modifying our search algorithms. In information retrieval, inverted files are arguably the most efficient index structures for free-text search [21].Our approach

combines modified versions of R-trees and inverted files to achieve effective pruning of the search space [32], with an extra quad-tree index to implement MapReduce.

CHAPTER 4

GeoCloud: Online Spatial Data Analysis and Visualization

With the exponential growth of the usage of web map services, the geo data analysis has become more and more popular. This paper develops an online Spatial Data Analysis System, TerraFly GeoCloud, which facilitates the end user to visualize and analyze spatial data, and to share the analysis results. Built on the TerraFly Geo spatial database, TerraFly GeoCloud is an extra layer running upon TerraFly map supporting many different visualization functions and spatial data analysis models. TerraFly GeoCloud also enables the MapQL technology to create maps using SQL-like statements.

4.1 Introduction

With the exponential growth of the World Wide Web, there are many domains, such as water management, crime mapping, disease analysis, and real estate, open to Geographic Information System (GIS) applications. The Web can provide a giant amount of information to a multitude of users, making GIS available to a wider range of public users than ever before. Web-based map services are the most important application of modern GIS systems. For example, Google Maps currently has more than 350 million users. There are also a rapidly growing number of geo-enabled applications which utilize web map services on traditional computing platforms as well as the emerging mobile devices.

However, due to the highly complex and dynamic nature of GIS systems, it is quite challenging for the end users to quickly understand and analyze the spatial data, and to efficiently share their own data and analysis results to others. First, typical

geographic visualization tools are complicated and fussy with a lot of low-level details, thus they are difficult to use for spatial data analysis. Second, the analysis of large amount spatial data is very resource-consuming. Third, current spatial data visualization tools are not well integrated for map developers and it is difficult for end users to create the map applications on their own spatial datasets.

To address the above challenges, this paper presents TerraFly GeoCloud, an online spatial data analysis and visualization system, which allows end users to easily visualize and share various types of spatial data. First, TerraFly GeoCloud can accurately visualize and manipulate point and polygon spatial data with just a few clicks. Second, TerraFly GeoCloud employs an analysis engine to support the online analysis of spatial data, and the visualization of the analysis results. Many different spatial analysis functionalities are provided by the analysis engine. Third, based on the TerraFly map API, TerraFly GeoCloud offers a MapQL language with SQL-like statements to execute spatial queries, and render maps to visualize the customized query results.

Our TerraFly GeoCloud online spatial data analysis and visualization system is built upon the TerraFly system using TerraFly Maps API and JavaScript TerraFly API add-ons in a high performance cloud Environment. The function modules in the analysis engine are implemented using C and R language and python scripts. Comparing with current GIS applications, our system is more user-friendly and offers better usability in the analysis and visualization of spatial data. The system is available at http://terrafly.fiu.edu/GeoCloud/.

4.2 Background

4.2.1 TerraFly geospatial database

TerraFly is a system for querying and visualizing of geospatial data developed by High Performance Database Research Center (HPDRC) lab in Florida International University (FIU). This TerraFly system serves worldwide web map requests over 125 countries and regions, providing users with customized aerial photography, satellite imagery and various overlays, such as street names, roads, restaurants, services and demographic data [1].

TerraFly Application Programming Interface (API) allows rapid deployment of interactive Web applications and has been used to produce systems for disaster mitigation, ecology, real estate, tourism, and municipalities. TerraFly's Web-based client interface is accessible from anywhere via any standard Web browser, with no client software to install.

TerraFly allows users to virtually 'fly' over enormous geographic information simply via a web browser with a bunch of advanced functionalities and features such as user-friendly geospatial querying interface, map display with user-specific granularity, real-time data suppliers, demographic analysis, annotation, route dissemination via autopilots and application programming interface (API) for web sites, etc. [1-7][10].

TerraFly's server farm ingests geo-locates, cleanses, mosaics, and cross-references 40TB of base map data and user-specific data streams. The 40TB TerraFly data collection includes, among others, 1-meter aerial photography of almost the entire United States and 3-inch to 1-foot full-color recent imagery of major urban areas. TerraFly vector collection

includes 400 million geo-located objects, 50 billion data fields, 40 million polylines, 120 million polygons, including: all US and Canada roads, the US Census demographic and socioeconomic datasets, 110 million parcels with property lines and ownership data, 15 million records of businesses with company stats and management roles and contacts, 2 million physicians with expertise detail, various public place databases (including the USGS GNIS and NGA GNS), Wikipedia, extensive global environmental data (including daily feeds from NASA and NOAA satellites and the USGS water gauges), and hundreds of other datasets [66][82].

4.2.2 Visualizing spatial data

Information visualization (or data visualization) techniques are able to present the data and patterns in a visual form that is intuitive and easily comprehendible, allow users to derive insights from the data, and support user interactions [83].

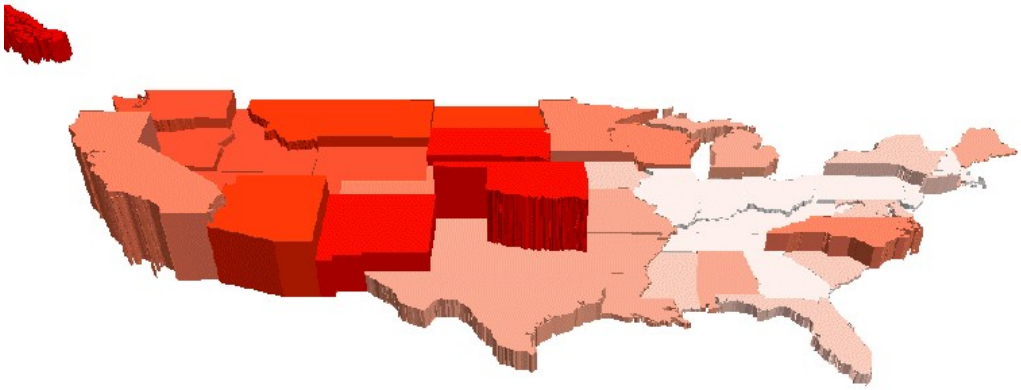


Figure 4.1 Population Total (height) vs. Density (color) of US

For example, Figure 4.1 shows the map of Native American population statistics which has the geographic spatial dimensions and several data dimensions. The figure displays both the total population and the population density on a map, and users can

easily gain some insights on the data by a glance [5]. In addition, visualizing spatial data can also help end users interpret and understand spatial data mining results. They can get a better understanding on the discovered patterns.

Visualizing the objects in geo-spatial data is as important as the data itself. The visualization task becomes more challenging as both the data dimensionality and richness in the object representation increase. In TerraFly GeoCloud, we have devoted lots of effort to address the visualization challenge including the visualization of multi-dimensional data and the flexible user interaction.

TerraFly GeoCloud integrates spatial data mining and data visualization. The integration of spatial data mining and information visualization has been widely to discover hidden patterns. For spatial data mining to be effective, it is important to include the visualization techniques in the mining process and to generate the discovered patterns for a more comprehensive visual view [68].

4.2.3 Map Rendering

The process of rendering a map generally means taking raw geospatial data and making a visual map from it. Often it applies more specifically to the production of a raster image, or a set of raster tiles, but it can refer to the production of map outputs in vector-based formats. "3D rendering" is also possible when taking the map data as an input. The ability of rendering maps in new and interesting styles, or highlighting features of special interest, is one of the most exciting aspects in spatial data analysis and visualization.

TerraFly map render engine is a toolkit for rendering maps and is used to render the main map layers. It supports a variety of geospatial data formats and provides flexible styling options for designing many different kinds of maps, and the render speed is fast [85][86].

TerraFly map render engine is written in C++ and can be used as a web service. It uses the AGG library and offers anti-aliasing rendering with pixel accuracy. It can read different kind of file like PostGIS, TIFF rasters, .osm files, and other shape files. Packages are available for both Window and Linux [86].

4.3 TerraFly GeoCloud

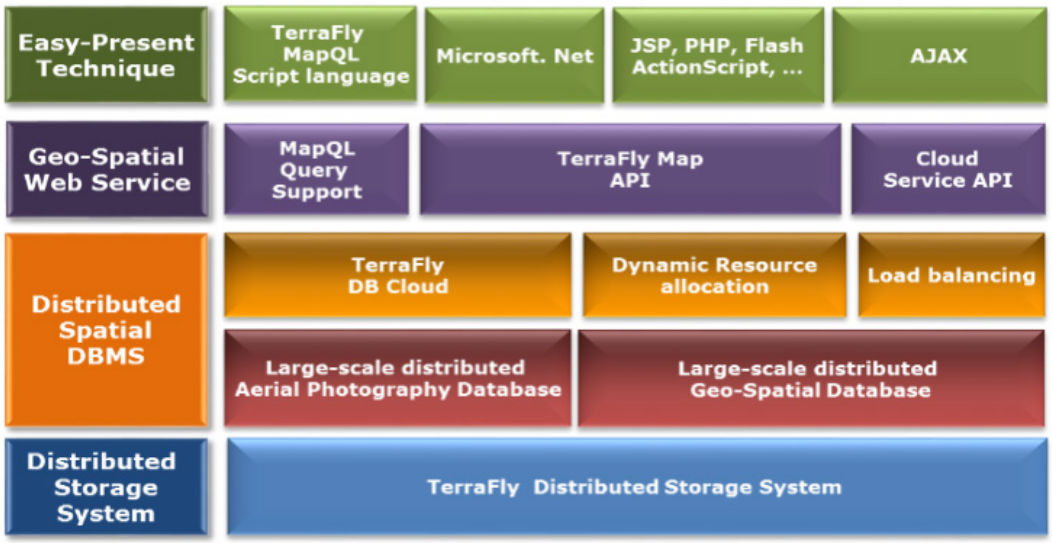


Figure 4.2 The Architecture of TerraFly GeoCloud

Figure 4.2 shows the system architecture of TerraFly GeoCloud. Based on the current TerraFly system including the Map API and all sorts of TerraFly data, we developed the TerraFly GeoCloud system to perform online spatial data analysis and visualization. In TerraFly GeoCloud, users can import and visualize various types of spatial data (data with geo-location information) on the TerraFly map, edit the data,

perform spatial data analysis, and visualize and share the analysis results to others. Available spatial data sources in TerraFly GeoCloud include but not limited to demographic census, real estate, disaster, hydrology, retail, crime, and disease. In addition, the system supports MapQL, which is a technology to customize map visualization using SQL-like statements.

The spatial data analysis functions provided by TerraFly GeoCloud include spatial data visualization (visualizing the spatial data), spatial dependency and autocorrelation (checking for spatial dependencies), spatial clustering (grouping similar spatial objects), and Kriging (geo-statistical estimator for unobserved locations).

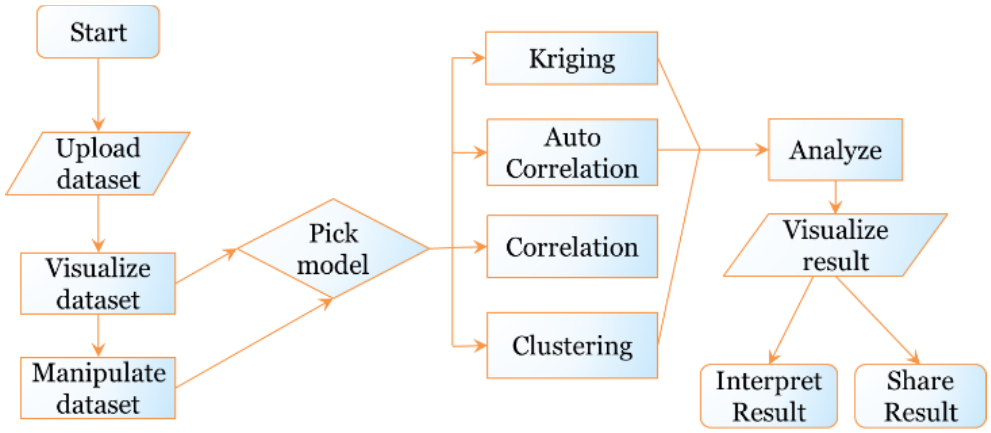


Figure 4.3 The Workflow of TerraFly GeoCloud Analysis

Figure 4.3 shows the data analysis workflow of the TerraFly GeoCloud system. Users first *upload datasets* to the system, or view the available datasets in the system. They can then *visualize the data sets* with customized appearances. By *Manipulate dataset*, users can edit the dataset and perform pre-processing (e.g., adding more columns). Followed by pre-processing, users can choose proper spatial analysis functions and perform the analysis. After the analysis, they can visualize the results and are also able to share them with others.

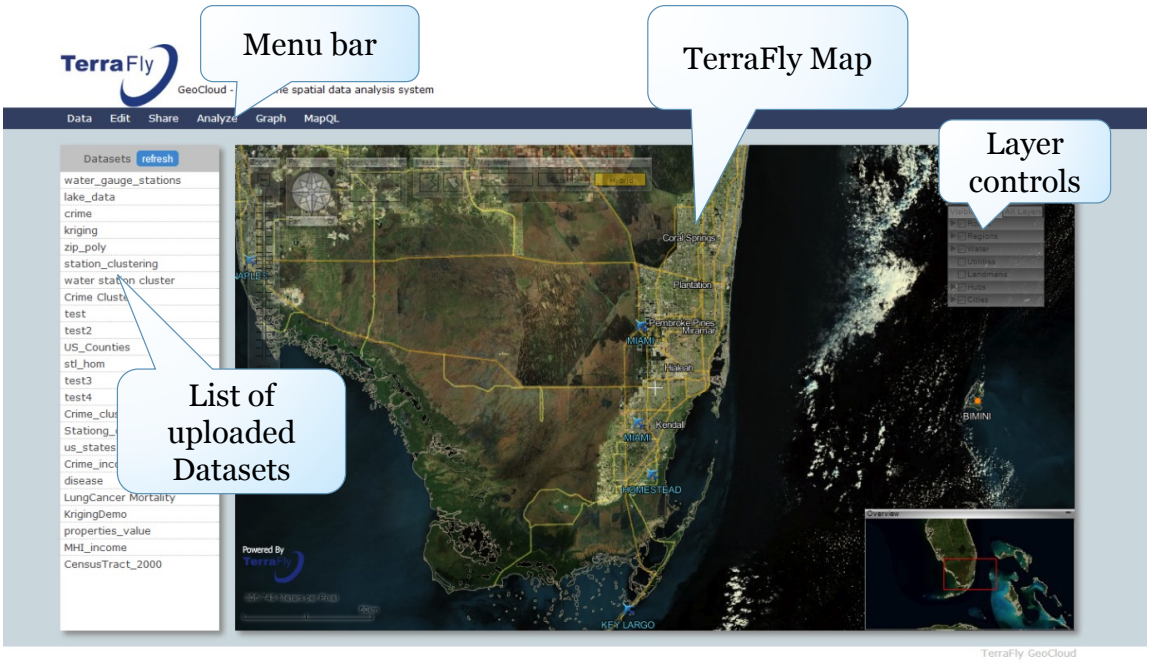


Figure 4.4 Interface of TerraFly GeoCloud

Figure 4.4 showed the interface of the TerraFly GeoCloud system. The top bar is the menu of all functions, including *Data, analysis, Graph, Share,* and *MapQL*. The left side shows the available datasets, including both the uploaded datasets from the user and the existing datasets in the system. The right map is the main map from TerraFly. This map is composed by TerraFly API, and it includes a detailed base map and diverse overlays which can present different kinds of geographical data [87].

TerraFly GeoCloud also provides MapQL spatial query and render tools. MapQL supports SQL-like statements to realize the spatial query, and after that, render the map according to users' inputs. MapQL tools can help users visualize their own data using a simple statement [88]. This provides users with a better mechanism to easily visualize geographical data and analysis results.

4.4 Visualization in TerraFly GeoCloud

4.4.1 Spatial Data Visualization

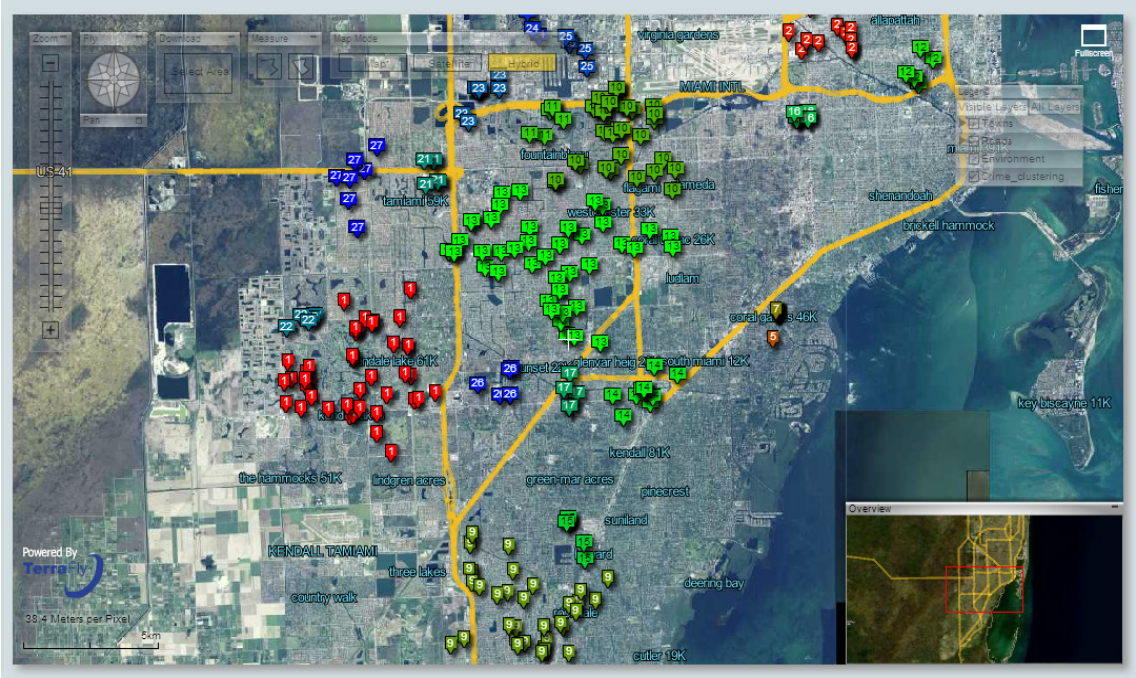


Figure 4.5 Spatial Data Visualization: Point Data

For spatial data visualization, the system supports both point data and polygon data and users can choose color or color range of data for displaying. As shown in Figures, the point data is displayed in Figure 4.5, and the polygen data is displayed in Figure 4.6.

The data labels are shown on the base map as extra layers for point data, and the data polygons are shown on the base map for polygon data. Many different visualization choices are supported for both point data and polygon data. For point data, user can customize the icon style, icon color or color range, label value and so on. For polygon data, user can customize the fill color or color range, fill alpha, line color, line width, line alpha, label value and so on.

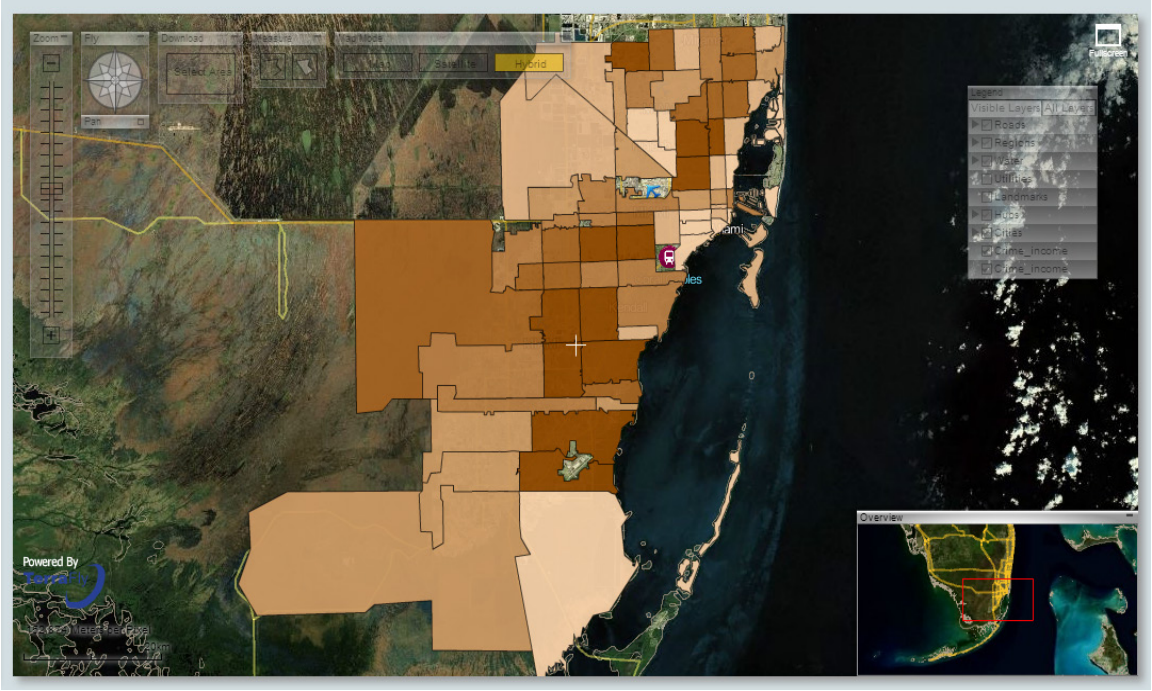


Figure 4.6 Spatial Data Visualization: Polygon Data

4.4.2 Spatial Data Mining Results Visualization

TerraFly GeoCloud integrates spatial data mining and data visualization. The spatial data mining results can be easily visualized. In addition, visualization can often be incorporated into the spatial mining process.

a) Spatial dependency and Auto-Correlation

Spatial dependency is the co-variation of properties within geographic space: characteristics at proximal locations that appear to be correlated, either positively or negatively. Spatial dependency leads to the spatial autocorrelation problem in statistics [89].

Spatial autocorrelation is more complex than one-dimensional autocorrelation because spatial correlation is multi-dimensional (i.e. 2 or 3 dimensions of space) and multi-directional. The TerraFly GeoCloud system provides auto-correlation analysis tools

to discover spatial dependencies in a geographic space, including global and local clusters analysis where Moran's I measure is used [90].

Formally, Moran's I, the slope of the line, estimates the overall global degree of spatial autocorrelation as follows:

$$I = \frac{n}{\sum_i^n \sum_j^n w_{ij}} \times \frac{\sum_i^n \sum_j^n w_{ij}(y_i - \bar{y})(y_j - \bar{y})}{\sum_i^n (y_j - \bar{y})^2}$$

where w_{ij} is the weight, $w_{ij}=1$ if locations i and j are adjacent and zero otherwise $w_{ii}=0$ (a region is not adjacent to itself).y_i and $\bar{y}$ are the variable in the ith location and the mean of the variable, respectively. n is the total number of observations. Moran's I is used to test hypotheses concerning the correlation, ranging between –1.0 and +1.0.

Moran's I measures can be displayed as a checkerboard where a positive Moran's I measure indicates the clustering of similar values and a negative Moran's I measure indicate dissimilar values. TerraFly GeoCloud system provides auto-correlation analysis tools to check for spatial dependencies in a geographic space, including global and local clusters analysis [91].

Local Moran's I is a local spatial autocorrelation statistic based on the Moran's I statistic. It was developed by Anselin as a local indicator of spatial association or LISA statistic [92]. The fact that Moran's I is a summation of individual cross products is exploited by the "Local indicators of spatial association" (LISA) to evaluate the clustering in those individual units by calculating Local Moran's I for each spatial unit and evaluating the statistical significance for each Ii. From the previous equation we then obtain:

$$I_i = z_i \sum_j^n w_{ij} z_j$$

where z_i are the deviations from the mean of $y_{i,}$ and the weights are row standardized.

Figure 4.7 Average properties price by zip code in Miami

Figure 4.7 shows an example of spatial auto-correlation analysis on the average properties price by zip code data in Miami (polygon data). Each dot here in the scatterplot corresponds to one zip code. The first and third quadrants of the plot represent positive associations (high-high and low-low), while the second and fourth quadrants represent associations (low-high, high-low). For example, the *green circle area* is in the low-high quadrants. The density of the quadrants represents the dominating local spatial process. The properties in Miami Beach are more expensive, and are in the high-high area.

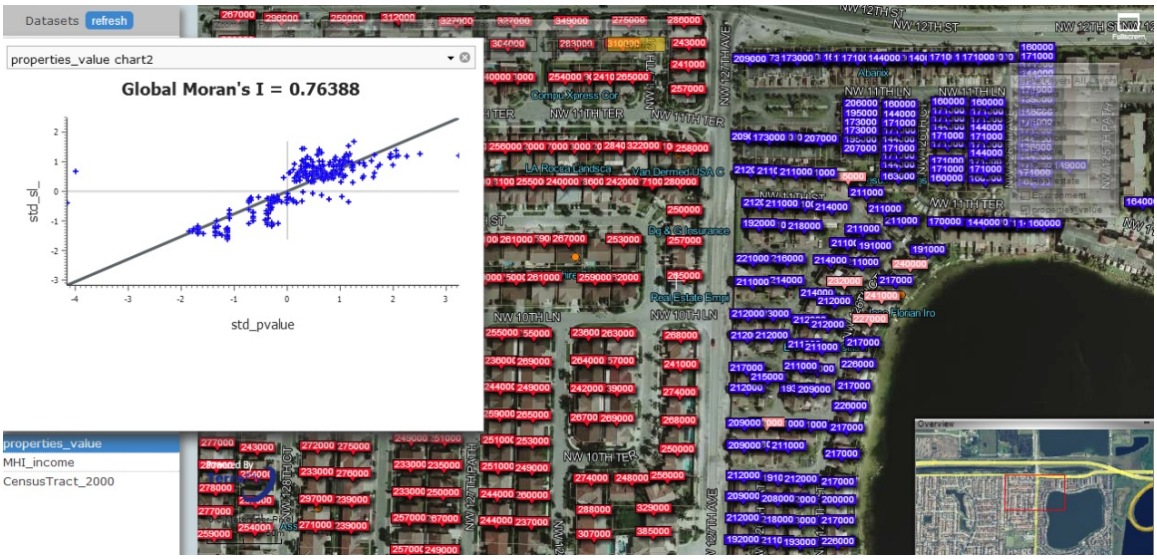


Figure 4.8 Properties value in Miami

Figure 4.8 presents the auto-correlation analysis results on the individual properties price in Miami (point data). Each dot here in the scatterplot corresponds to one property. As the figure shows, the properties near the big lake are cheaper, while the properties along the west are more expensive.

b) Spatial Data Clustering

The TerraFly GeoCloud system supports the DBSCAN (for density-based spatial clustering of applications with noise) data clustering algorithm [93]. It is a density-based clustering algorithm because it finds a number of clusters starting from the estimated density distribution of corresponding nodes.

DBSCAN requires two parameters as the input: *eps* and the minimum number of points required to form a cluster *minPts*. It starts with an arbitrary starting point that has not been visited so far. This point's neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as a noise point [93]. If a point is found to be a dense part of a cluster, its neighborhood is also part

of that cluster. Hence, all points that are found within the neighborhood are added. This process continues until the density-connected cluster is completely identified. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise points [94].

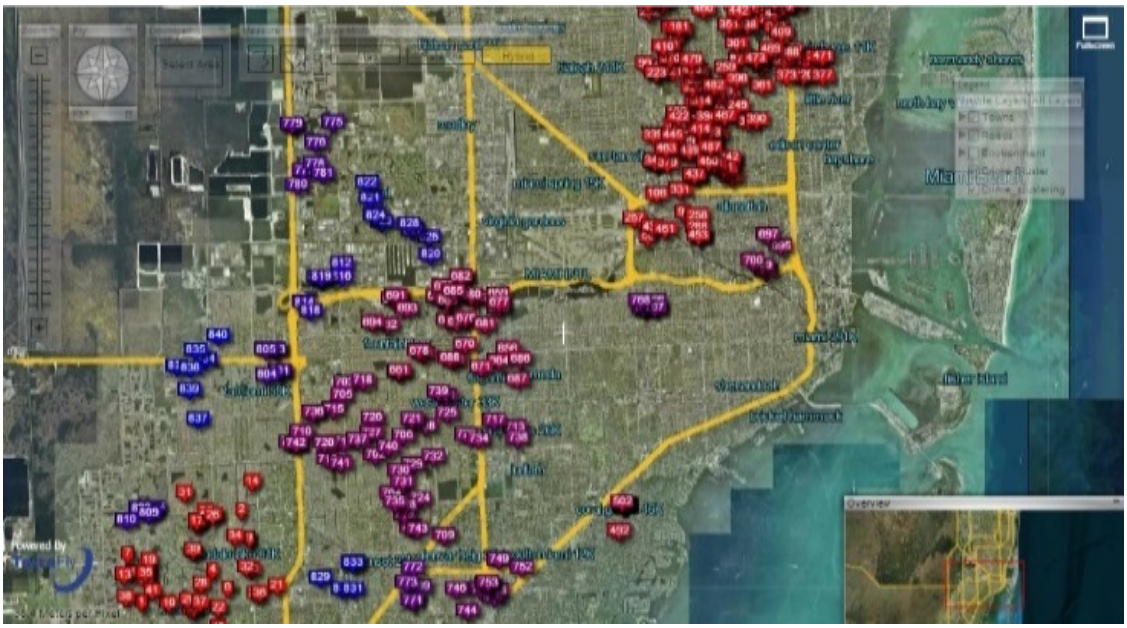


Figure 4.9 DBSCAN clustering on the crime data in Miami

Figure 4.9 shows an example of DBSCAN clustering on the crime data in Miami. As shown in Figure 6, each point is an individual crime record marked on the place where the crime happened, and the number displayed in the label is the crime ID. By using the clustering algorithm, the crime records are grouped, and different clusters are represented by different colors on the map.

c) Kriging

Kriging is a geo-statistical estimator that infers the value of a random field at an unobserved location (e.g. elevation as a function of geographic coordinates) from samples (see spatial analysis) [95].

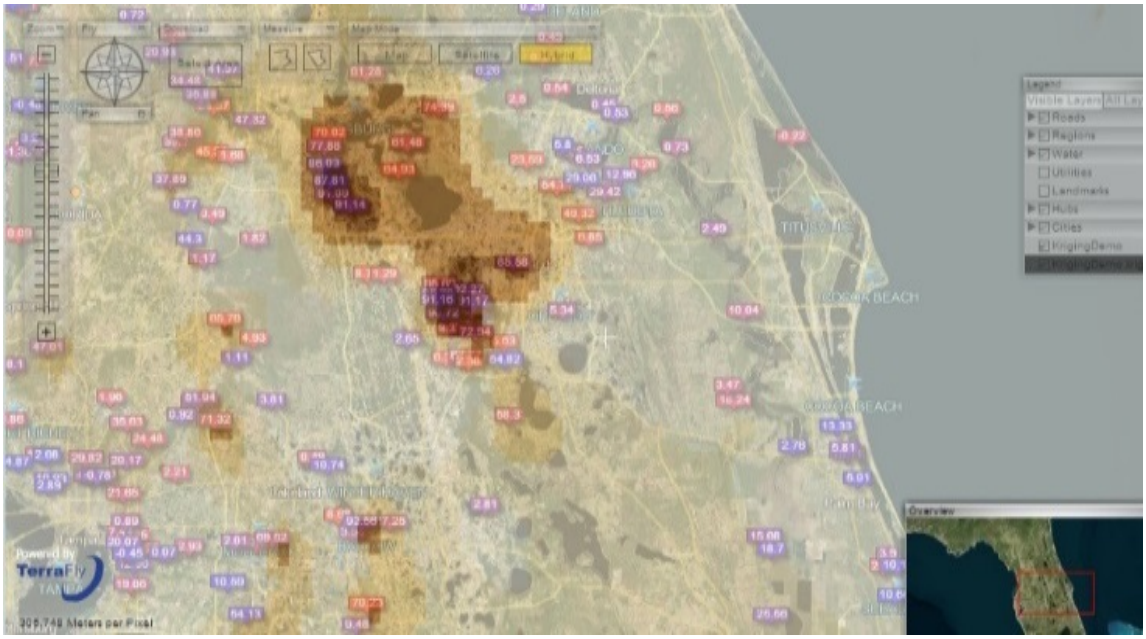


Figure 4.10 Kriging data of the water level in Florida

Figure 4.10 shows an example of Kriging. The data set is the water level from water stations in central Florida. Note that not all the water surfaces are measured by water stations. The Kriging results are estimates of the water levels and are shown by the yellow layer.

4.4.3 Customized Map Visualization (Supported by MapQL)

TerraFly GeoCloud also provides MapQL spatial query and render tools, which supports SQL-like statements to facilitate the spatial query and more importantly, render the map according users' requests. This is a better interface than API to facilitate developer and end user to use the TerraFly map as their wish. By using MapQL tools, users can easily create their own maps.

a) Implementation

The implementation of MapQL is shown in Figure 4.11. The input of the whole procedure is MapQL statements, and the output is map visualization rendered by the MapQL engine.

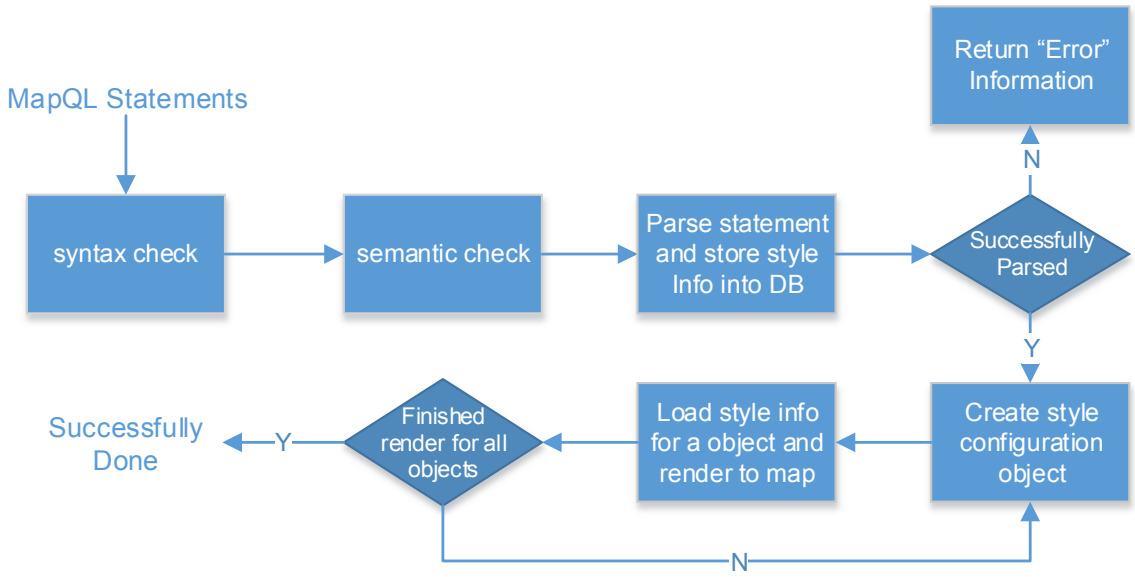


Figure 4.11 MapQL implementation

Shown in Figure 4.11, the first step is syntax check of the statements. Syntax check guarantees that the syntax conforms to the standard, such as the spelling-check of the reserved words. Semantic check ensures that the data source name and metadata which MapQL statements want to visit are correct. After the above two checks, system will parse the statements and store the parse results including the style information into a spatial database. The style information includes where to render and what to render. After all the style information is stored, system will create style configuration objects for render. The last step is for each object, load the style information form spatial database and render to the map according to the style information.

We implemented the MapQL tools using C++. For the last step which is rendering the objects to the map visualization, we employed the TerraFly map render engine [8].

For example, if we want to query the house prices near Florida International University, we use MapQL statements like this:

```
SELECT
   '/var/www/cgi-bin/house.png' AS T_ICON_PATH,
r.price AS T_LABEL,
   '15' AS T_LABEL_SIZE,
r.geo AS GEO
FROM
realtor_20121116 r
WHERE
ST_Distance(r.geo,  GeomFromText('POINT(-80.376283 25.757228)')) < 0.03;
```

There are four reseverd words in the statements, *T_ICON_PATH* , *T_LABEL*, *T_LABEL_SIZE ,* and *GEO.* We use *T_ICON_PATH* to store the customized icon. Here we choose a local png file as icon. *T_LABEL* denotes that icon label that will be shown on the map, . *T_LABEL_SIZE* is the pixel size of the label; and *GEO* is the spatial search geometry.

The statement goes through the syntax check first. If there is incorrect usage of reserved words or wrong spelling of the syntax, it will be corrected or Error information will be sent to users. For example, if the spelling of "*select*" is not correct, Error information will be sent to user. Semantic check makes sure that the data source name *realtor_20121116 and* metadata *r. price* and *r.geo* are exist and available.

After the checks, the system parsed the statements. The SQL part will return corresponding results including the locations and names of nearby objects, the MapQL part will collect the style information like icon path and icon label style. Both of them are

stored into a spatial database. The system then created style configuration objects for query results. The last step is rendering all the objects on the map visualizations. The style information needed includes icon picture and label size, and the data information includes label value and location (Lat, Long).

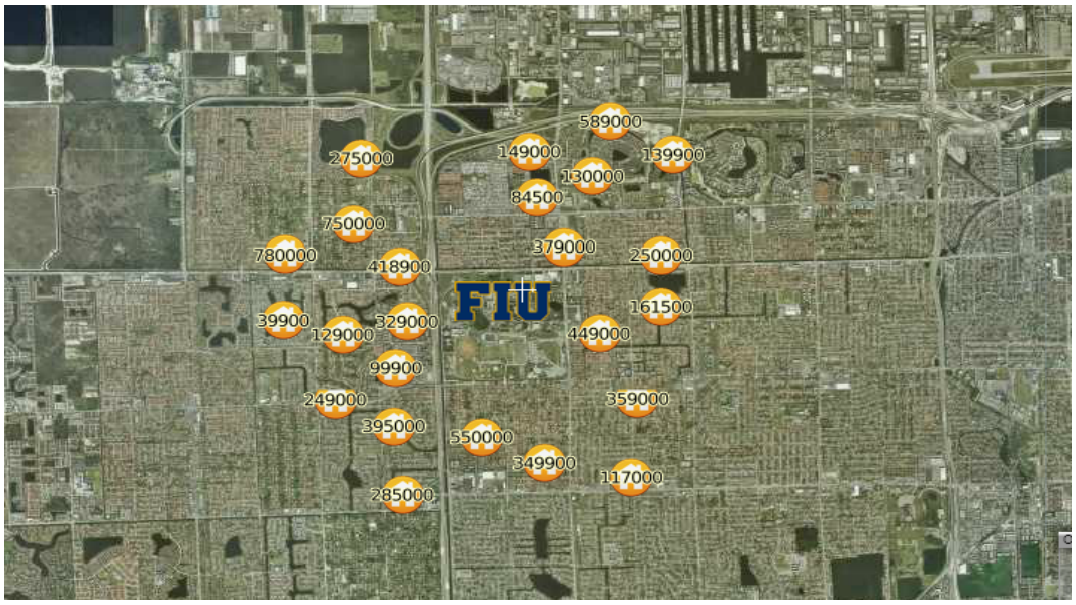


Figure 4.12 Query data near the point

Figure 4.12 shows the result of this query. Please be noticed that the unit of the distance function in all the demos is Lat-Long.

b) More Samples

Figure 4.13 shows all the hotels along a certain street within a certain distance and also displays the different stars of the hotels. The MapQL statement for this query is listed below:

```
SELECT
  CASE
    WHEN star >= 1 and star < 2 THEN '/var/www/cgi-bin/hotel_1star.png'
    WHEN star >= 2 and star < 3 THEN '/var/www/cgi-bin/hotel_2stars.png'
    WHEN star >= 3 and star < 4 THEN '/var/www/cgi-bin/hotel_3stars.png'
```

```
    WHEN star >= 4 and star < 5 THEN '/var/www/cgi-bin/hotel_4stars.png'
    WHEN star >= 5 THEN '/var/www/cgi-bin/hotel_2stars.png'
    ELSE '/var/www/cgi-bin/hotel_0star.png'
  END AS T_ICON_PATH,
h.geo AS GEO
FROM
osm_fl o
LEFT JOIN
hotel_all h
ON
ST_Distance(o.geo, h.geo) < 0.05
WHERE
  o.name = 'Florida Turnpike';
```

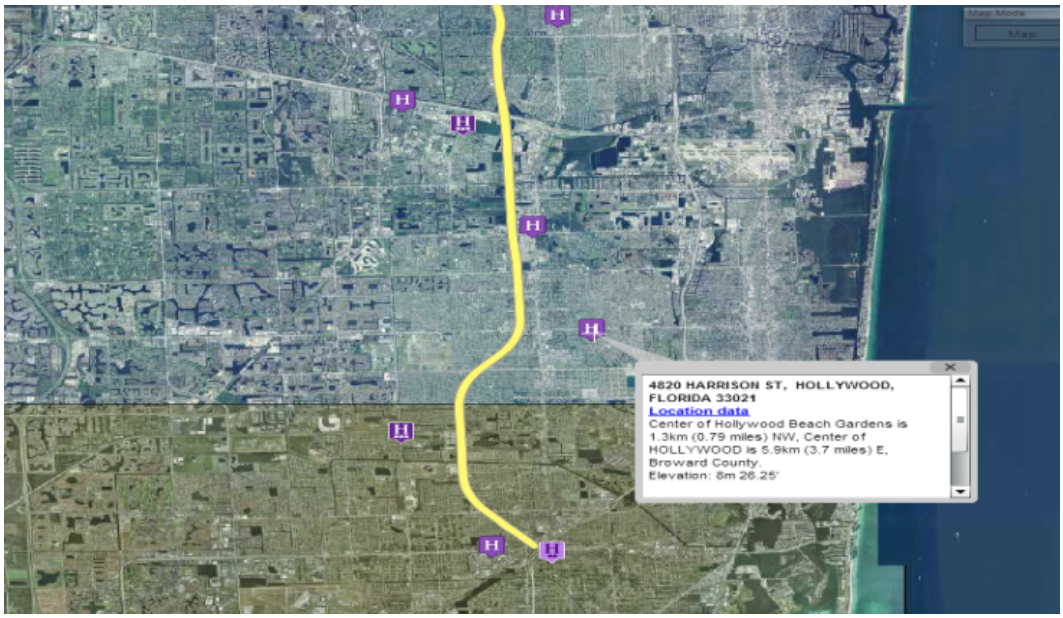


Figure 4.13 Query data along the line

Figure 4.14 shows the traffic of Santiago where the colder the color is, the faster the traffic is, the warmer the color is, and the worse the traffic is. The MapQL statement is listed below:

```
SELECT
  CASE
    WHEN speed >= 50 THEN 'color(155, 188, 255)'
    WHEN speed >= 40 and speed < 50 THEN 'color(233, 236, 255)'
    WHEN speed >= 30 and speed < 40 THEN 'color(255, 225, 198)'
```

```
    WHEN speed >= 20 and speed < 30 THEN 'color(255, 189, 111)'
    WHEN speed >= 10 and speed < 20 THEN 'color(255, 146, 29)'
    WHEN speed >= 5 and speed < 10 THEN 'color(255, 69, 0)'
    WHEN speed >= 0 and speed < 5 THEN 'color("red")'
else 'color("grey")'
  END AS T_FILLED_COLOR,
  '3' AS T_THICKNESS,
GEO
FROM santiago_traffic;
```

Figure 4.14 Traffic of Santiago

Figure 4.15 shows the different average incomes with in different zip codes. In this demo, users can customize the color and style of the map layers, different color stand for different average incomes. And the MapQL statement is listed below:

```
SELECT
u.geo AS GEO,
u.zip AS T_LABEL,
  '0.7' AS T_OPACITY,
  '15' AS T_LABEL_SIZE,
'color("blue")' AS T_BORDER_COLOR,
  CASE
    WHEN avg(i.income) < 30000 THEN 'color(155, 188, 255)'
    WHEN avg(i.income) >= 30000 and avg(i.income) < 50000 THEN 'color(233, 236, 255)'
    WHEN avg(i.income) >= 50000 and avg(i.income) < 70000 THEN 'color(255, 225, 198)'
```

```
    WHEN avg(i.income) >= 70000 and avg(i.income) < 90000 THEN 'color(255, 189, 111)'
    WHEN avg(i.income) >= 90000 and avg(i.income) < 110000 THEN 'color(255, 146, 29)'
    WHEN avg(i.income) >= 110000 and avg(i.income) < 130000 THEN 'color(255, 69, 0)'
    WHEN avg(i.income) >= 130000 THEN 'color("red")'
else 'color("grey")'
  END AS T_FILLED_COLOR
FROM
us_zip u left join income i
ON
ST_Within(i.geo, u.geo)='t'
GROUP BY
u.geo, u.zip;
```

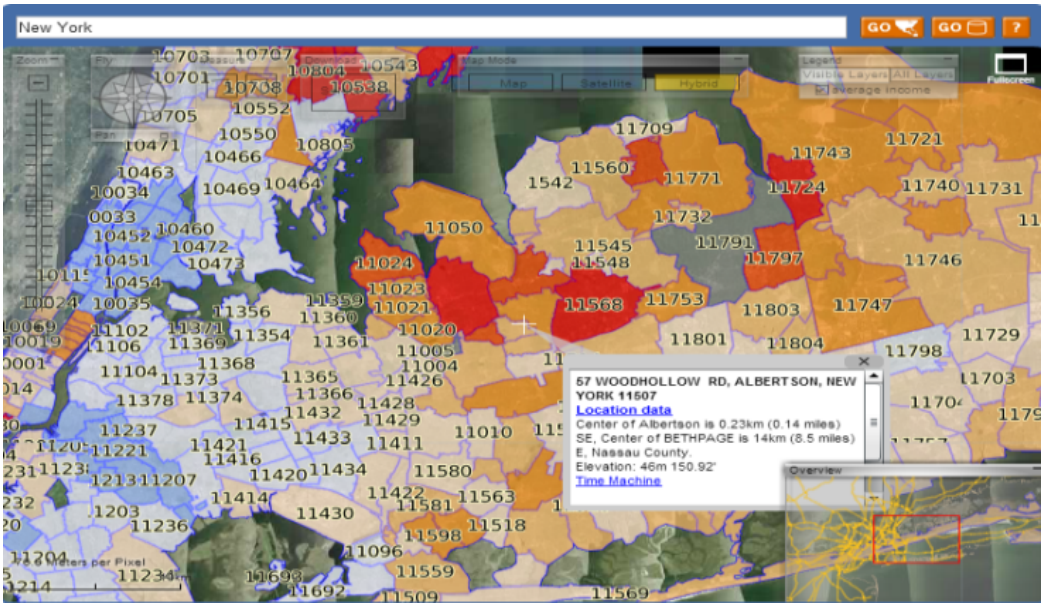


Figure 4.15 Income at New York

All these examples demonstrate that in TerraFly GeoCloud, users can easily create different map applications using simple SQL-like statements.

4.5 Case Study

This section is the cases studies of the proposed GeoCloud service.

4.5.1 Setup

As a typical web application, GeoCloud provides a variety of web services via Internet Information Services (IIS) to serve online web requests. The test bed is set up on a Dell PowerEdge servers, each with XEON Intel E5520 2.27 GHz, 16GB (4x4GB) ECC -- DDR3 1066MHz, and one 1TB 7.2 RPM SAS disk. Windows server 2008 and SQL Server 2008 are installed to provide the environment for GeoCloud.

4.5.2 Case Study for realtor data analysis

In this section, we present a case study on using TerraFly GeoCloud for spatial data analysis and visualization. As discussed in 3.4.2, we know the results of auto correlation can be shown in a scatter diagram, where the first and third quadrants of the plot represent positive associations, while the second and fourth quadrants represent negative associations. The second quadrant stands for low-high which means the value of the object is low and the values of surrounding objects are high.

A lay user whose name is Erik who has some knowledge about the database and data analysis wanted to invest a house property in Miami with a good appreciation potential. By using TerraFly GeoCloud, he may obtain some ideas about where to buy. He believes that if a property itself has low price and the surrounding properties have higher values, then the property may have good appreciation potential, and is a good choice for investment. He wants to first identify such properties and then do a field trip with his friends and the realtor agent.

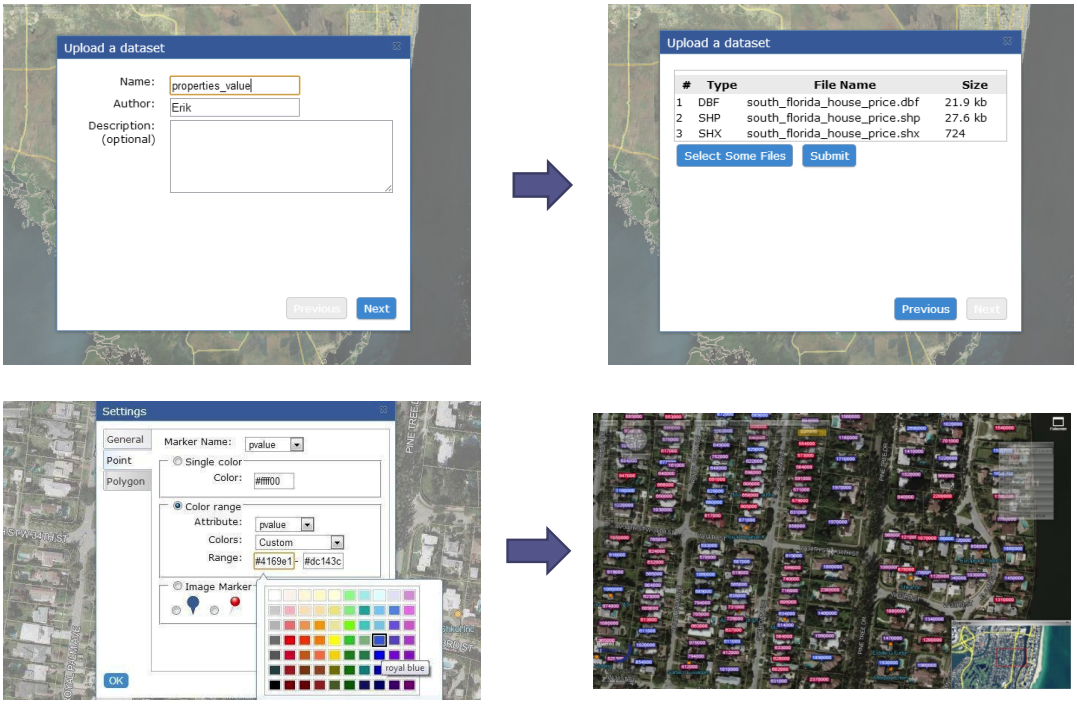


Figure 4.16 Data Set Upload and Visualization

To perform the task, first, Erik checked the average property prices by zip code in Miami which is shown in Figure 4.7. He found the *green circled area* in the low-high quadrants, which means that the average price of properties of this area is lower than the surrounding areas. Then, Erik wanted to obtain more insights on the property price in this area. He uploaded a detailed spatial data set named as *south_florida_house_price* into the TerraFly GeoCloud system as shown in Figure 4.16. He customized the label color range as the properties price changes. And then, he chose different areas in the *green circled area* in Figure 4.7 to perform the auto-correlation analysis.

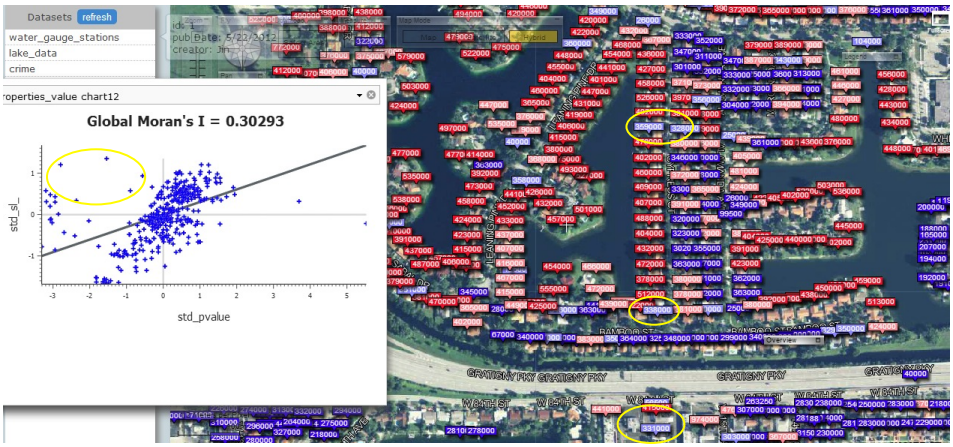


Figure 4.17 Properties in Miami

Finally, he found an area shown in Figure 4.17, where there are some good properties in the low-high quadrants (in yellow circles) with good locations. And one interesting observation is, lots of properties along the road *Gratigny Pkwy* has lower prices. He was then very excited and wanted to do a query to find all the cheap properties with good appreciation potential along the *Gratigny Pkwy*. Erik composed the MapQL statements like:

```
SELECT
  CASE
    WHEN h.pvalue >= 400000 THEN '/var/www/cgi-bin/redhouse.png'
    WHEN h.pvalue >= 200000 and h.pvalue < 400000 THEN '/var/www/cgi-bin/bluehouse.png'
    WHEN h.pvalue >= 100000 and h.pvalue < 200000 THEN '/var/www/cgi-bin/greenhouse.png'
    ELSE '/var/www/cgi-bin/darkhouse.png'
  END AS T_ICON_PATH,
h.geo AS GEO
FROM
osm_fl o
LEFT JOIN
south_florida_house_price h
ON
ST_Distance(o.geo, h.geo) < 0.05
WHERE
o.name = 'Gratigny Pkwy' AND
```

```
h.std_pvalue<0 AND
h.std_sl_pvalue>0;
```

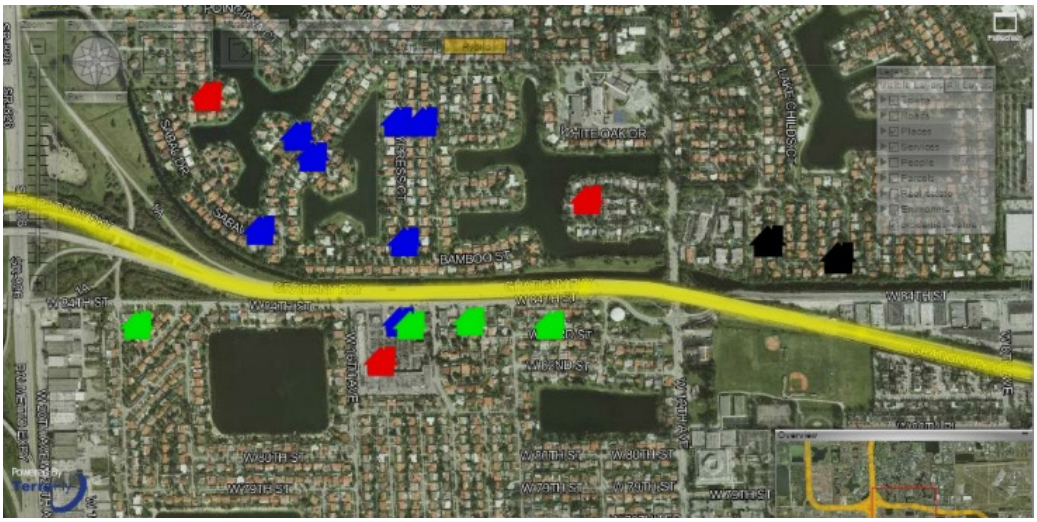


Figure 4.18 MapQL results

The Figure 4.18 presents the final results of the MapQL statements. Finally, Erik sent the URL of the map visualization out by email, waiting for the response of his friends and the realtor agent.

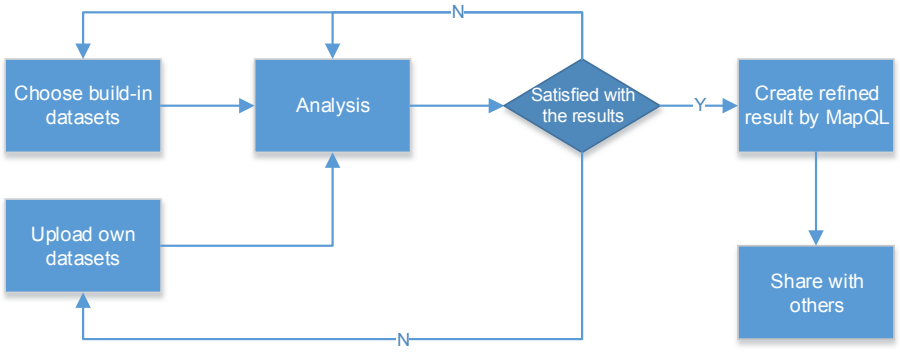


Figure 4.19 The flow path of Erik case

Figure 4.19 illustrates the whole workflow of the case study. In summary, Erik first viewed the system build-in datasets, conducted the data analysis, and then he identified properties of interest. He then composed MapQL statements to create his own map visualization to share with his friends. The case study demonstrates that TerraFly

GeoCloud supports the integration of spatial data analysis and visualization and also offers user-friendly mechanisms for customized map visualization.

4.5.3 A case study for Epidemiological Data Analysis

In this section we provide an example of how our geospatial epidemiology system can be employed in epidemiologic research [96][97]. Assume a researcher studies lung cancer in Florida. She can upload and choose the mor_price_income dataset to TerraFly GeoCloud - shown in Figure 20.

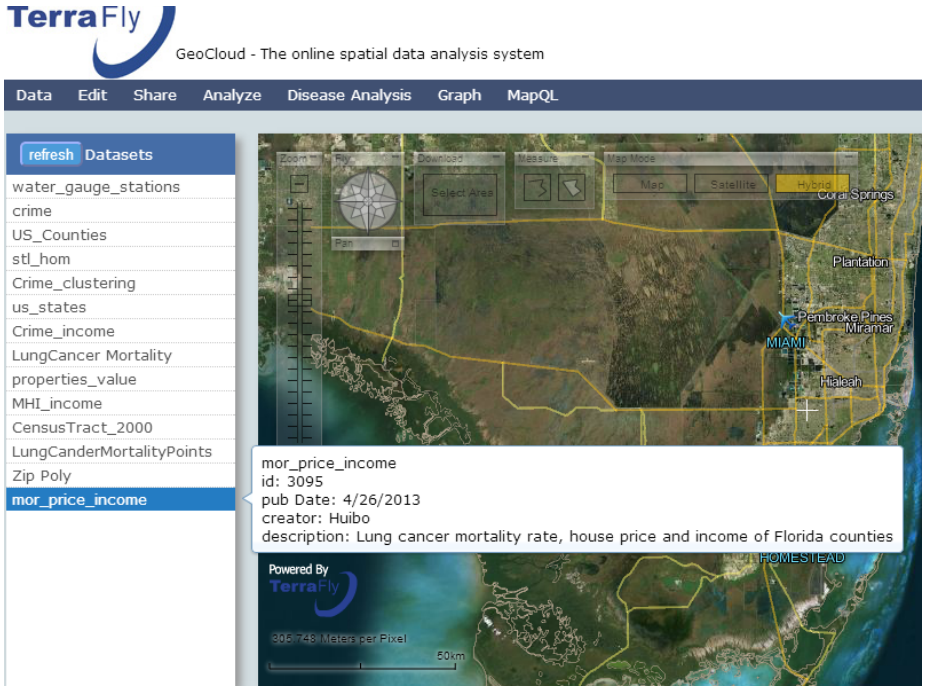


Figure 4.20 Datasets in TerraFly GeoCloud

She can then choose the disease analysis button to draw a disease map. In this function, she can choose a legend group number; a disease map is displayed then, as shown in Figure 21.

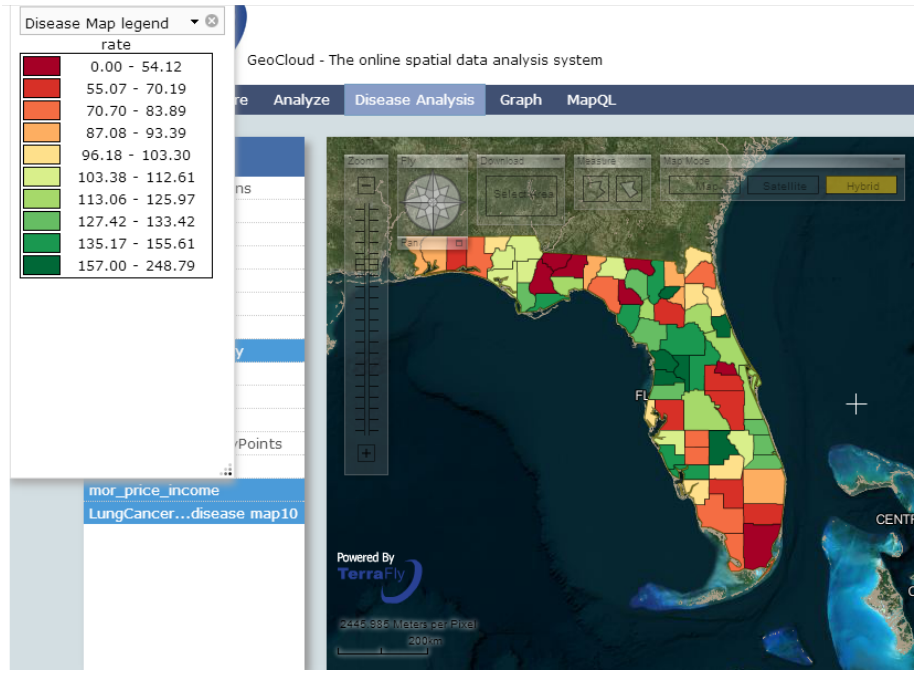


Figure 4.21 Lung cancer disease map

From Figure 21 we see how this map, with legend at the top left corner, gives a direct summary of the disease data [98-110]. For lung cancer in Florida, the mortality in the central region is higher and in the south is lower. However, the researcher cannot have an accurate analysis just from this one map. She can further choose the cluster and outliner function, which uses Local Moran's I to perform further analysis. This function provides three maps: local Moran's I map, z-value map, and p-value map. Figure 9 shows the p-value map, from which the researcher can know which counties form a statistically significant cluster and which counties are statistically significant outliners.

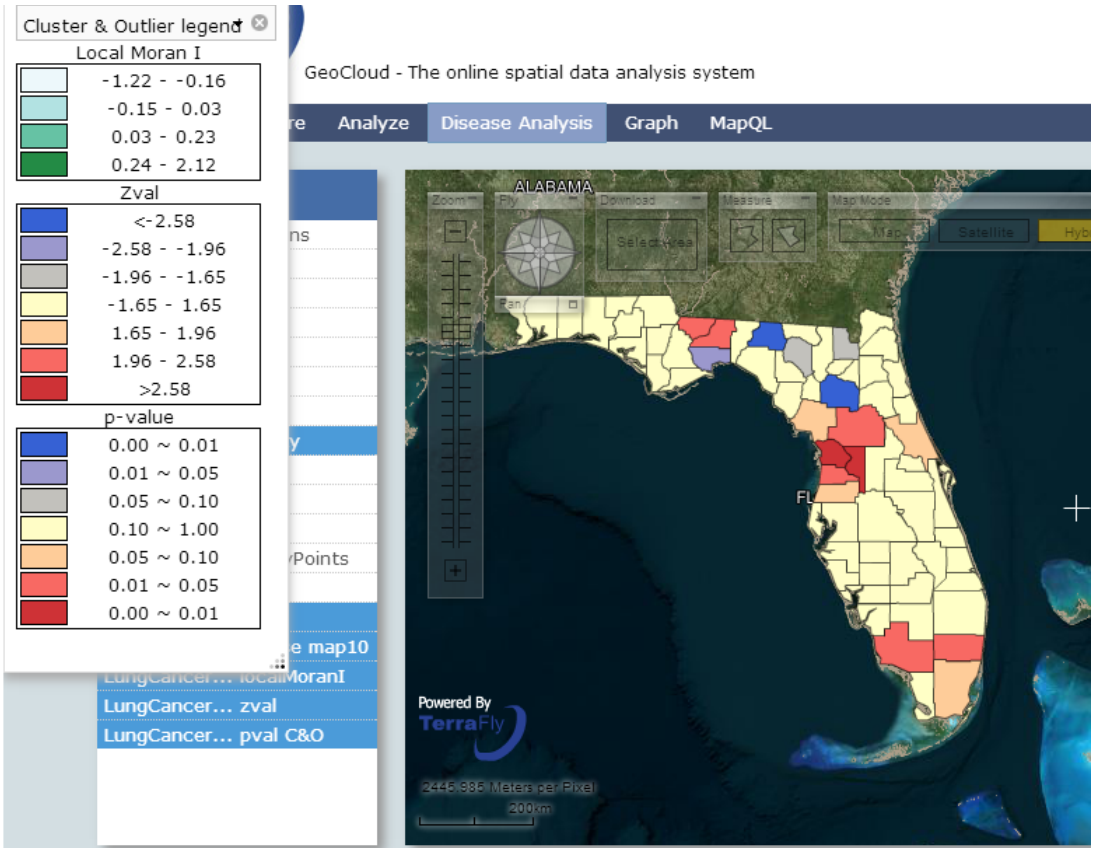


Figure 4.22 P-value map of Local Moran I

Now the researcher may want to know what kind of relationship there is between lung cancer mortality and the median income of each county. For this purpose, she can use the median income dataset provided by the GeoCloud system, and apply to it the spatial auto-regression tool [111-117]. Figure 10 shows the result of this model. From the result, we learn that when the mortality of surrounding areas increase by 1, the mortality of this county will increase of 0.233, and when the median income in the surrounding area increases by $10000, the mortality of this county will decrease of 0.09.

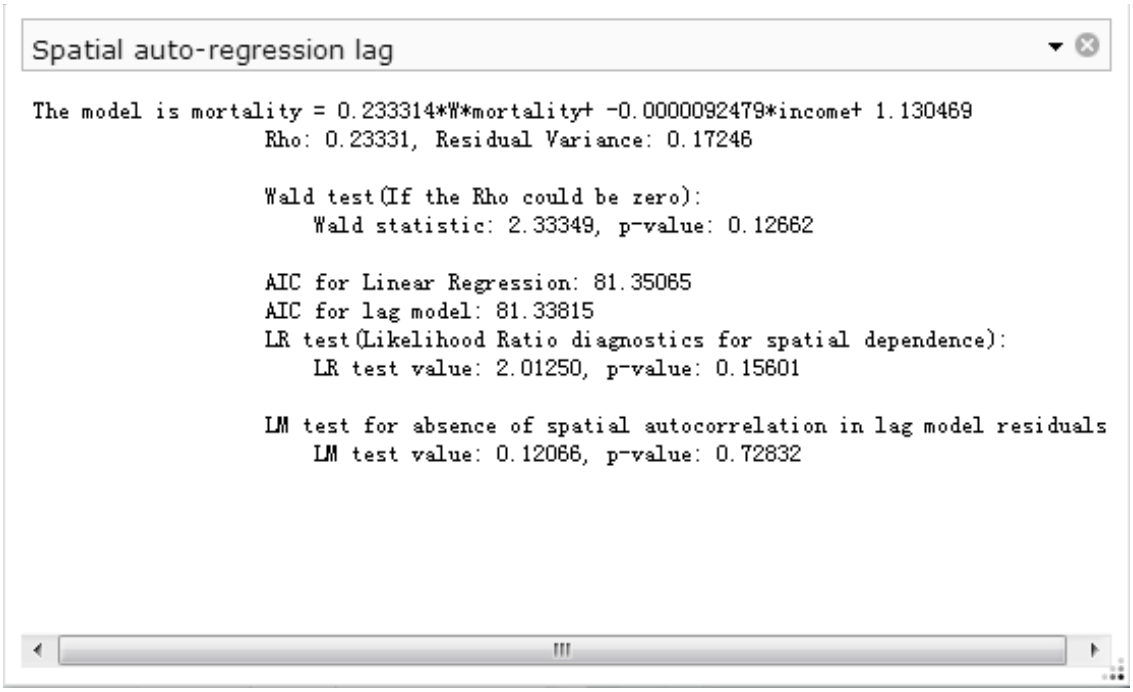


Figure 4.23 Spatial auto-regression of lung cancer mortality and median income

4.6 Related Work and Products

In the geospatial discipline, web-based GIS services can significantly reduce the data volume and required computing resources at the end-user side [33][34]. To the best of our knowledge, TerraFly GeoCloud is one of the first systems to study the integration of online visualization of spatial data, data analysis modules and visualization customization language.

Various GIS analysis tools are developed and visualization customization languages have been studied in the literature. ArcGIS is a complete, cloud-based, collaborative content management system for working with geographic information. But systems like ArcGIS and Geoda focus on the content management and share, not online analysis[36][37]. Azavea has many functions such as optimal Location find, Crime analysis, data aggregation and visualization. It is good at visualization, but has very limited analysis functions [38].

Various types of solutions have been studied in the literature to address the problem of visualization of spatial analysis [37]. However, on one hand, good analysis visualization tools like Geoda and ArcGIS do not have online functions. To use them, users have to download and install the software tools, and download the datasets. On the other hand, good online GIS systems like Azavea, SKE, and GISCloud have limited analysis functions. Furthermore, none of above products provides a simple and convenient way like MapQL to let user create their own map visualization [39][40]. The related products are summarized in Table 1. Our work is complementary to the existing works and our system also integrates the data mining and visualization.

Table 1: GIS Visualization Products

Name	Website	Product features description	Comments
ArcGIS Online	http://www.arcgis.com	http://www.arcgis.com ArcGIS Online is a complete, cloud-based, collaborative content management system for working with geographic information.	No online Analysis, focus on the content management and share.
Azavea	http://www.azavea.com/products/	optimal Location find, Crime analsis, data aggregated and visualized	Good visualization. Very limited Analysis functions
SKE	http://www.skeinc.com/GeoPortal.html	Spatial data Viewer	Focus on the spatial data viewer.
GISCloud	http://www.giscloud.com	with few analysis (Buffer , Range , Area , Comparison , Hotspot , Coverage , Spatial Selection)	Very limited simple analysis.
GeoIQ	http://www.geoiq.com/ http://geocommons.com/	filtering, buffers, spatial aggregation and predictive	Focus on GIS, very good Visualization and interactive operation. Very limited and simple analysis: currently provide predictive(Pearsons Correlation).

CHAPTER 5

v-TerraFly: Autonomic Resource Management for Virtualized Web Map

v-TerraFly: Autonomic Resource Management for Virtualized Web Map Service System is another very important part of modern map system. With the fast growing use of web-based map services, the resource management of such services are becoming increasing important to deliver user desired Quality of Service. Map systems often serve dynamic web workloads and involve multiple CPU and I/O intensive tiers, which make it challenging to meet the response time targets of map requests while using the resources efficiently. This paper proposes a virtualized web map service system, v-TerraFly, and its autonomic resource management in order to address this challenge. Virtualization facilitates the deployment of web map services and improves their resource utilization through encapsulation and consolidation. Autonomic resource management allows resources to be automatically provisioned to a map service and its internal tiers on demand. Specifically, this paper proposes new techniques to predict the demand of map workloads online and optimize resource allocations considering both response time and data freshness as the QoS target. The proposed v-TerraFly system is prototyped on TerraFly, a production web map service, and evaluated using real TerraFly workloads. The results show that v-TerraFly can accurately predict the workload demands: 18.91% more accurate; and efficiently allocate resources to meet the QoS target: improves the QoS by 26.19% and saves re-source usages by 20.83% compared to traditional peak-load-based resource allocation.

5.1 Introduction

With the exponential growth of the World Wide Web, there are more domains open to Geographic Information System (GIS) applications. Internet can provide information to a multitude of users, making GIS available to a wider range of public users than ever before. Web-based map services are the most important application of modern GIS systems. For example, Google Maps has more than 350 million users. There are also a rapidly growing number of geo-enabled applications which consume web map services on traditional computing platforms as well as the emerging mobile devices.

Virtual machines (VM) are powerful platforms for hosting web map service systems. VMs support flexible resource allocation to both meet web map services system demands and share resources with other applications. Virtualization is also enabling technology for the emerging cloud computing paradigm, which further allows highly scalable and cost-effective web map services hosting leveraging its elastic resource availability and pay-as-you-go economic model [54].

However, due to the highly complex and dynamic nature of web map service systems, it is challenging to efficiently host them using virtualized resources. First, typical web map services have to serve dynamically changing workloads, which makes it difficult to host map services on shared resources without compromising performance or wasting resources. Second, a web map service often consists of several tiers which have different intensive resource needs and result in dynamic internal resource contention. Third, for a typical web map service, both response time for requests and the freshness of the returned data are critical factors of the Quality of Service (QoS) required by users.

To address the above challenges, this paper presents v-TerraFly, an autonomic resource management approach for virtualized map service systems, which can automatically optimize the QoS (considering both response time and data freshness) while minimizing the resource cost [118][119]. First, v-TerraFly can accurately predict the workload demands of a web map service online based on a novel two-way forecasting algorithm that considers both historical hourly patterns and daily patterns. Second, based on the predicted workload, v-TerraFly can automatically estimate the resource demands of its various tiers based on performance profiles created using machine learning techniques. Third, v-TerraFly employs a new QoS model that captures the balance between response time and data freshness and uses this model to automatically optimize the resource allocation of a web map service system [120].

This proposed v-TerraFly system is realized on Hyper-V virtual machine environments and evaluated by experiments using real workloads collected from the production TerraFly system. The results show that the proposed two-level workload prediction method is outperforms traditional exponential smoothing prediction by 18.91%, and the system improves the QoS by 26.19% compared to traditional statically node allocation. In the meantime, it saves resource usages by 20.83% compared to traditional peak-load-based resource allocation.

In summary, this paper's main contributions are: 1) created a VMbased map service system, v-TerraFly, which virtualizes all tiers of a typical web map service and supports dynamic resource allocations to the different tiers; 2) proposed a novel autonomic resource management approach for virtualized map services, which automatically allocate resources to different tiers of the service and optimize the

allocations based on the performance and data freshness tradeoff; 3) evaluated v-TerraFly using real workloads collected from production web map service system which shows substantial improvement on QoS and resource efficiency.

5.2 Background

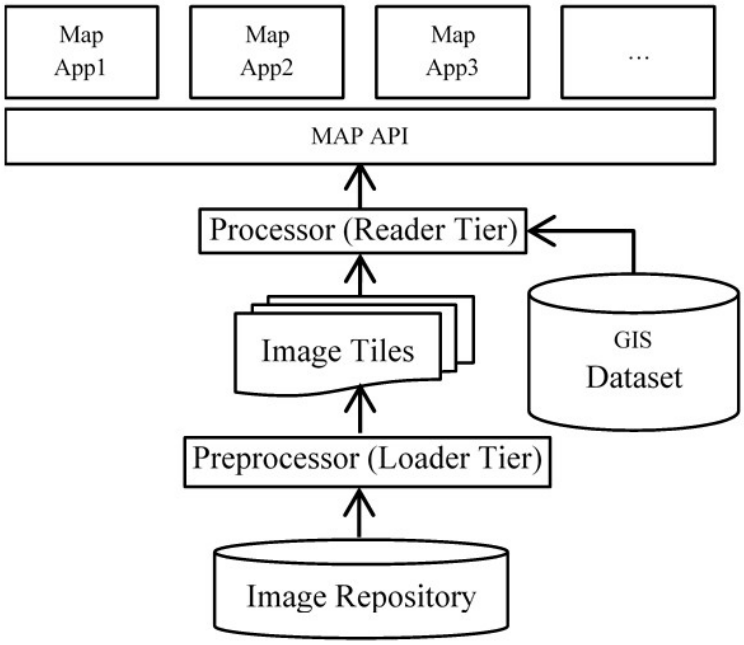


Figure 5.1 Web enabled Map Service

As a promising new trend in GIS, web map service exhibits its excellence in serving online map requests responsively and delivering geographical information precisely over the Internet [121]. A typical online satellite-based web map service, such as Google maps, Bing maps, and Yahoo maps [122], are usually built upon several major tiers (Figure 5.1). A *Preprocessor* preloads images and geographic features from raw data repository and splits them into grid format, known as image tiles, to facilitates the *Processor* quickly locate and fetch data. Then a tile *Processor* retrieves and integrates all

tiles needed in a customer query. As its upper tier, a generic map interface access this imagery by geo-location, and a client app (or browser) to show the map to end users.

In this paper, we use TerraFly as a case study of the web map system [1]. TerraFly serves worldwide web map requests over 125 countries and regions, providing users with customized aerial photography, satellite imagery and various overlays, such as street names, roads, restaurants, services and demographic data. Following the typical architecture described above, TerraFly contains two major tiers (Figure 5.1): an Image *Loader Tier* preprocesses the raw imagery data from repository; an Image *Reader Tier* processes image tiles and retrieves queried images [123]. More details about these tiers are described in Next Section.

Traditionally, web map services are hosted on dedicated physical servers with sufficient hardware resources to satisfy their expected peak workloads in order to provide responsive web services to the users. However, this becomes inefficient for real-world situations where the workloads are intrinsically dynamic in terms of their busty arrival patterns and ever changing unit processing costs [124]. Consequently, peak-load based resource provision often leads to underutilization of resources for normal state workloads and causes substantial overhead.

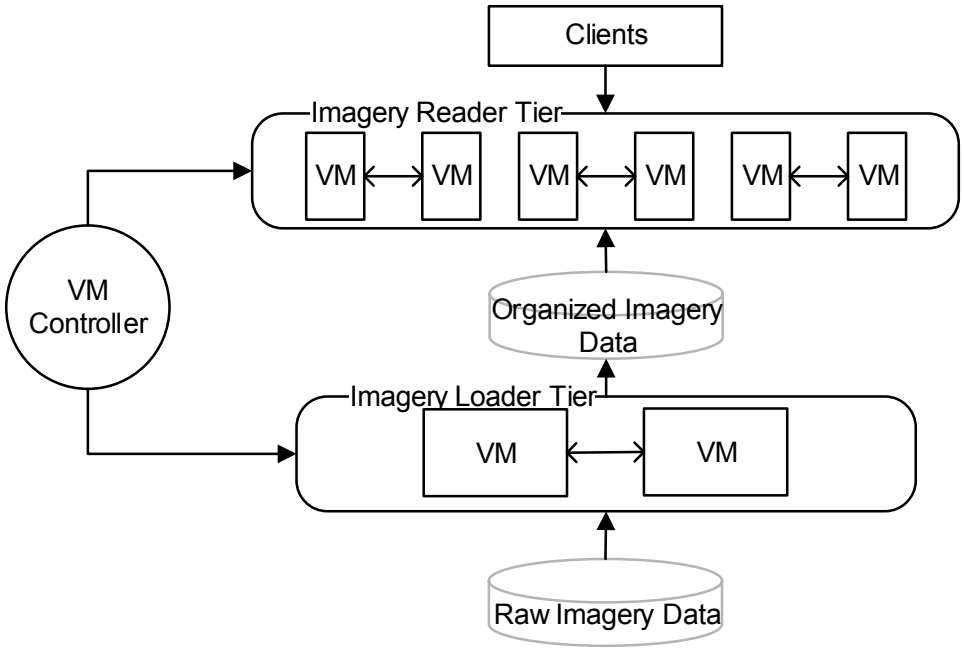


Figure 5.2 v-TerraFly system

Using VMs to host multi-tier web map services can effectively address this limitation because virtualized resources, including CPU, memory, and I/O, are decoupled from their physical infrastructures and can be flexibly allocated to different tiers of the web map system [125]. This approach allows the resource capacity of each tier to elastically grow and shrink to serve its dynamic. In this way, different tiers transparently share the consolidated resources with each other and/or other applications with strong isolation. Such benefits are important to the efficiency of web map service hosting in both typical data centers and emerging cloud systems. On one hand, users need to pay for only the resources their services actually consume. On the other hand, resource providers only need to allocate resources as required by the services while saving valuable resources for hosting other applications [126].

Virtualization also offers a new paradigm for web map service deployments. Modern web map services are sophisticated systems, where their installation and

configuration require substantial domain knowledge and experience as well as considerable efforts from the administrators. VM-based web map service hosting allows carefully installed software to be distributed as simply as copying the data that represents the VMs. In addition, this approach allows web map services to be quickly replicated and distributed for performance and reliability improvements.

5.3 v-TerraFly

5.3.1 Architecture

To enable the autonomic resource management in TerraFly, we leverage VM techniques to virtualize this multi-tier system, denoted as v-TerraFly. The two critical resource intensive tiers of TerraFly, the image *Loader* and *Reader Tier*s, are deployed on the VMs instead of physical servers.

Figure 5.2 shows the architecture of this v-TerraFly. The users interact with the application tier which handles most of the business logic and provides advanced application services, such as universal mapping, realtor mapping and water management, by sending the mapping queries including position and resolution requirements to the tiers below. Then the image *Reader Tier* is invoked to compute and locate associated map tiles from indexed imagery database according to the requests from the application. To maintain the data freshness, the organized imagery database is updated by the *Loader Tier* periodically at the same time. *Loader* contiguously extracts the incoming raw map data from the raw imagery repository, preprocesses and organizes the raw data to destination projection, and then convert them destination file type, finally update them into organized imagery database.

Due to the internal nature of mapping service system, these two tiers of v-TerraFly exhibit distinct resource usage behaviors in the production environment. On one hand, the *Reader Tier* may experience different number of concurrent users during different periods of a day, which results in highly dynamic workloads with varying intensity against the *Loader*. On the other hand, the *Leader* does not have stringent performance requirement as the *Roader* does but still needs reserved resource to guarantee the data freshness. Therefore, it is beneficial to host these two tiers together on virtualized cluster nodes to multiplex the common computing resources so that the total resource capacity can be better utilized among different tiers. For example, more VMs are allocated to the *Roader Tier* during daytime when peak-load of user requests is expected to happen but the loading process is less active; but shifting more VMs to the *Leader* over night to allow data updates accumulated in daytime.

Virtualization in TerraFly also improves the flexibility in terms of the system reliability and scalability. VM is the computing resource in both *Loader* and *Reader Tier*s. With the load balance in both tier, the work load of each VM is the same, therefore, the VMs in the same tier are considered identical. Since the computing resources can be partitioned through VM nodes, the network bandwidth which is always a bottleneck in the original system can be now well balanced among VMs. Furthermore, by pairing every two VMs as complementary *Reader* nodes, it is able to provide more reliable service under unexpected system failure by simply replacing the failure VM with its corresponding backupped VM [127].

5.3.2 Virtual Load balance cluster

Network Load Balancing can provide high availability and reliability, as well as high scalability. Web applications are stateless applications, and every client request to a stateless application is a separate transaction, so it is possible to distribute the requests among multiple servers to balance the load. One attractive feature of Network Load Balancing is that all servers in a cluster monitor each other with a heartbeat signal, so there is no single point of failure.

Use of a virtual machine will facilitate the build of load balancing clusters. Two host servers build pairs of VMs, and then the pairs construct different load balance clusters (Figure 3). For application layers, each server will have a paired server, to respond to requests together. This is an implementation of dual-server auto fail-over, offering better reliability.

5.3.3 Motivating Examples

In this section, we demonstrate both the necessity and benefits of resource consolidation in a map service system.

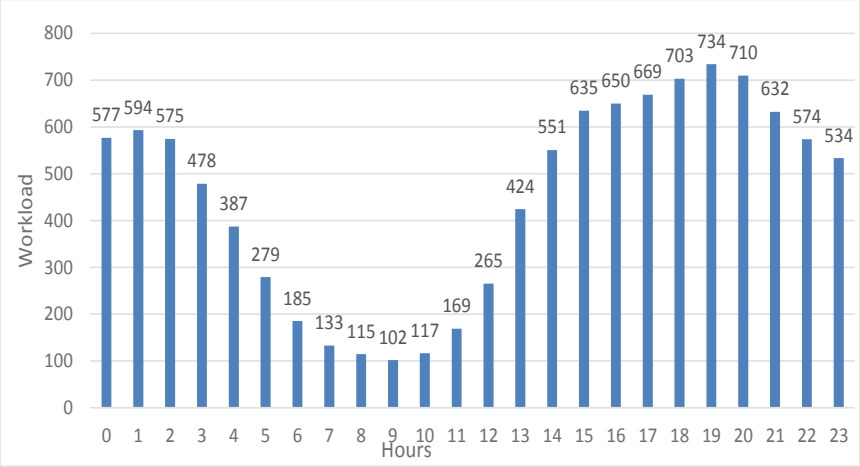


Figure 5.3 TerraFly System Workload Hourly Distribution

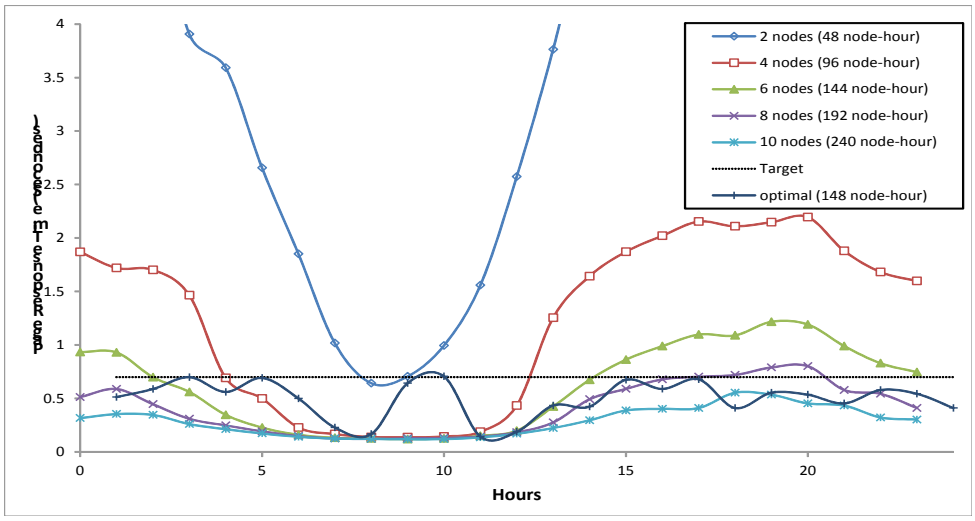


Figure 5.4 Performance and Resource cost comparison using different deployment schemes

For web map services, the performance of *Reader Tier* and *Loader Tier* are both important. Better *Reader Tier* performance provides shorter page response time; better *Loader Tier* performance provides faster loading of new map data. But both tiers are resource intensive and they will compete with each other on resource allocation. The goal is balance the both tier to achieve best quality of service when we have limited resource.

The workloads of web map service can be highly dynamic over time. Based on the analysis on the web service logs of TerraFly, it is observed that there were millions of web requests received on the *Reader* server over the year of 2012, i.e., more than 450 visits per second on average. However, this workload varies significantly on hourly basis. Figure 5.3 shows a typical one-day TerraFly workload trace. It shows that the request rate drops to 150 (visits per second) in the morning (around 9:00am) while rising quickly up to 900 (visits per second) in the afternoon. It would be more efficiently for us to turn off some *Reader* VM.

Assuming the variation in workload which follows such a time-related pattern is predictable, by virtualizing the *Reader Tier* of TerraFly we can easily save resources

when the workload intensity is low by simply turning off some *Reader* VMs, and as the workload intensity increases, we can bring back them online to process the additional requests.

To further quantify the resource savings, we replay this one-day trace using two deployment schemes for the *Reader Tier*: the static scheme deploy the Read tier on the fixed number of computing nodes throughout the entire experiment (2, 4, 6, 8 and 10 nodes respectively); the dynamic schema only assigns sufficient nodes needed by the workload in every hour. The response time is used as the performance metric and the desired QoS target is set to (0.7s). Fig. 4 compares the average response time in every hour using different schemes. We use *node-hour* as the cost unit in terms of computing resource. By measuring the number N_i of active nodes used in the ith hour, the total amount of the resource during certain time period T (hours) can be computed as $\sum_i^T N_i$.

Fig. 4 compared the measured response times in every hour using different deployments as well as the total resource costs needed in one day. As we can see among the static deployment configurations, only the 10-node configuration can always meet the desired target, however at the cost of the highest total resource amount (240 *node-hour*); others suffer different levels of QoS violation as the workload changes dynamically.

In contrast, the dynamic deployment scheme is able to track the QoS target all the time with only 148 *node-hour,* saving about 23% of the resources compared to the static 8-node configuration which cannot satisfy the QoS target, and saving 38 % of the resources compared to the static 10-node configuration which can satisfy the QoS target. Its resource utilization is as efficient as the 6-node configuration but delivers much better performance.

The above example shows strong evidence of the importance of map service virtualization and its online resource management. Currently, our traditional TerraFly system is deployed on the 8 physical *Reader Tier* nodes and 2 *Loader Tier* nodes. It works well for supporting up to 800 concurrent users, and about 6GB fresh data can be load each hour by the 2 *Loader* nodes (Refer to 4.3 Resource Model); but the system scalability is limited due to its fixed physical capacity. It cannot shift resources between *Reader* and *Loader Tier*s even when one tier has idle resource and another has insufficient resources. The inability of shifting resources between tiers results the waste of resources.

However, there are several challenges to dynamic resource management of a virtualized web map service. First, the dynamics in the realistic workload causes the demand of CPU consumption change over time; second, resources also need to be dynamically allocated between the *Reader Tier* to optimize response time and the *Loader Tier* to keep the data fresh.

These challenges can be well addressed by an autonomic VM-based resource management solution which can flexibly partition shared resources and allocate resource on-demand for dynamic workload in order to guarantee performance and improve resource utilization.

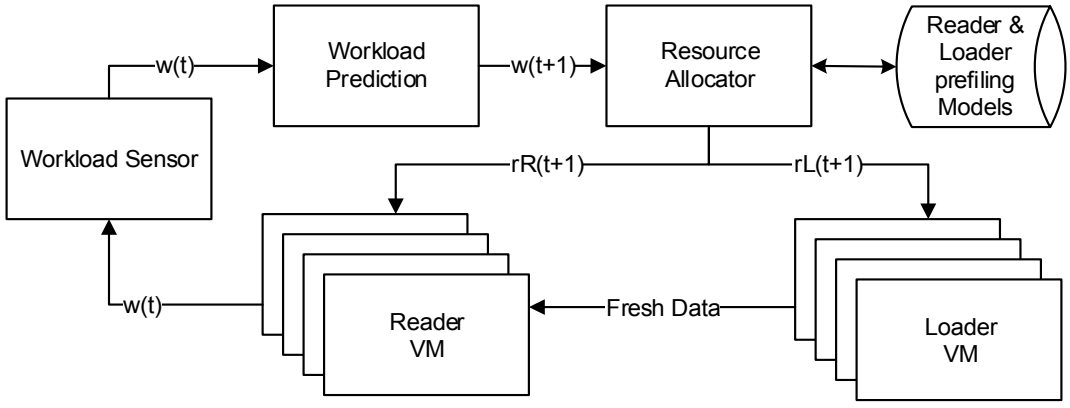


Figure 5.5 Autonomic resource management system for v-TerraFly

5.4 Autonomic Resource Management in v-TerraFly

5.4.1 General Approach

Figure 5.5 illustrates the framework of our proposed autonomic resource management system for v-TerraFly, which consists of three key modules. In this paper, we focus on the resource management for both *Reader* and *Loader Tier*s， since they are the most resource intensive tiers in v-TerraFly.

Table. 1. Parameter Description

Parameter	Description
$\boldsymbol{w(t)}$	Workload at time t
$\boldsymbol{rR(t)}$	Reader VM CPU resource need at time *t*
$\boldsymbol{rL(t)}$	Loader VM CPU resource need at time *t*
$\boldsymbol{w_h^{Des}}$	Horizontal double exponential smoothing prediction
$\boldsymbol{w_d^{Des}}$	Vertical double exponential smoothing prediction
$\boldsymbol{w'}$	Two-level double exponential smoothing

As a workload executes on the VMs, the *Workload Sensor* monitors the actual workload at current time step, noted as $w(t)$. The *Workload Predictor* then forecasts the future workload $w(t+1)$ for the next time step based on a prediction model. Based on the predicted workload, the *Reader* profile and *Loader* profile which are trained offline are used to estimate their resource demands for time $t+1$, denoted by $rR(t+1)$ and $rL(t+1)$ respectively. The estimated resource demands are then used by the *Resource Allocator* to make the actual allocations by assigning appropriate number of VMs to the *Reader Tier* and *Loader Tier*. Together, these modules form a closed-loop which runs continuously (e.g., every hour,) for v-TerraFly's resource control and optimization. In the rest of this section, we describe the key components of this autonomic system in details.

5.4.2 Workload Prediction

In order to accurately and timely predict the workload on v-TerraFly, we propose new forecasting techniques to discover and exploit patterns in user visiting behaviors such as those observed in Figure 5.3. Specifically, we propose a new two-level time series prediction approach to build a prediction model based on the historical workload measurements, i.e., the request rate observed from the *Reader Tier* of v-TerraFly. Based on such a model, the workload predictor in v-TerraFly is able to estimate the workload intensity for the next time period.

Time series analysis techniques are widely applied in economic data analysis to provide statistical prediction and therefore guide business decisions. A variety of time series prediction methods are available such as the *Moving Averages*, *Linear Regression* and *Exponential Smoothing* [129][130][131]. In this paper, the TerraFly workload

prediction based on the double exponential smoothing (DES) method [132] which is suitable for discrete data sequence with repeated changing patterns.

DES is a smoothing-based forecasting method that can be applied to time series data, a sequence of observations with equally spaced intervals, expressed as {$Y(0)$, $Y(1)$,.., $Y(t)$}. Then in DES, the estimate for the t+1 time intervals can be computed as:

$$Y^{Des}(t+1) = 2S'(t) - S''(t) + \left(\frac{\alpha}{1-\alpha}\right)\left(S'(t) - S''(t)\right)$$

$$S'(t) = \alpha Y(t) + (1-\alpha)S'(t-1)$$

$$S''(t) = \alpha S'(t) + (1-\alpha)S''(t-1)$$

The equation shows a linear combination of smoothing based statistics associated with a smoothing weight α. The first two components reflect the variation of mean of the overall data while the third tracks the trend of the data. *S'* is denoted as the singly-smoothed series which smoothed the next measure by assigning a exponentially decreased smoothing weight to the data of the series and computing the weighted average of the observed series. More intuitively, the most recent data is of more importance to the current estimates, i.e., the weight assigned to the data k periods old is$(1-\alpha)^k$, therefore the closer to 1 of the value of α, less smoothing effect but greater weight to the recent changes. *S''* is denoted as the doubly-smoothed series computed by recursively applying the same exponential smoothing operation to the singly-smoothed series *S'* using the same smoothing weight.

In order to perform the time-series-based forecast in v-TerraFly, the workload can be represented as a sequence of intensity measurements that come from a continuing time series at time intervals T, denoted as { … $w(t\text{-}2T)$, $w(t\text{-}T)$, $w(t)$}. More specifically, the

workload measurement can be either the average request rate or the number of concurrent client sessions observed in every hour; the time interval *T* can be either one hour or one day (24 hour).

We propose a new two-level double exponential smoothing forecasting model to capture both the daily pattern and hourly pattern of v-TerraFly workload as follows.

$$\text{Eq. 1: } w'(t+1) = \mu_h w_h^{Des}(t+1) + \mu_d w_d^{Des}(t+1)$$

where w_h^{Des} is the *horizontal* double exponential smoothing prediction based on the hourly pattern in the workload, and w_d^{Des} is the *vertical* double exponential smoothing prediction based on the daily pattern of the workload.

$$\text{Eq. 2: } w_h^{Des}(t) = 2S'(t-1) - S''(t-1) + \left(\frac{\alpha_h}{1-\alpha_h}\right)\left(S'(t-1) - S''(t-1)\right)$$

$$\text{Eq. 3: } w_d^{Des}(t) = 2S'(t-24) - S''(t-24) + \left(\frac{\alpha_d}{1-\alpha_d}\right)\left(S'(t-24) - S''(t-24)\right)$$

More specifically, $w_h^{Des}(t)$, called horizontal prediction, is predicted based on $\{w(t-3), w(t-2), w(t-1)\}$ from a hourly series; while $w_d^{Des}(t)$, vertical prediction, is based on the observation series $\{w(t-48), w(t-24), w(t)\}$ that are extracted vertically at the same hours but from continuing days, i.e., a 24-hour vertical time span. The associated μ factors which are set between 0 to 1 are used to balance the importance between three components.

Since each level of DES operation is associated with a smoothing weights, we denote the weights in horizontal and vertical predictions as α_h and α_d respectively. Then the proposed two-level DES model can be considered as a function of α_h and α_d, given observed workload series. Therefore the workload model is trained continuously online as soon as the new measurement is observed by optimizing both α_h and α_d to minimize

the weighted sum of squared errors between the prediction and the actual observation. Once the model is updated, it applies to the system immediately for the next prediction.

5.4.3 Performance Profiling

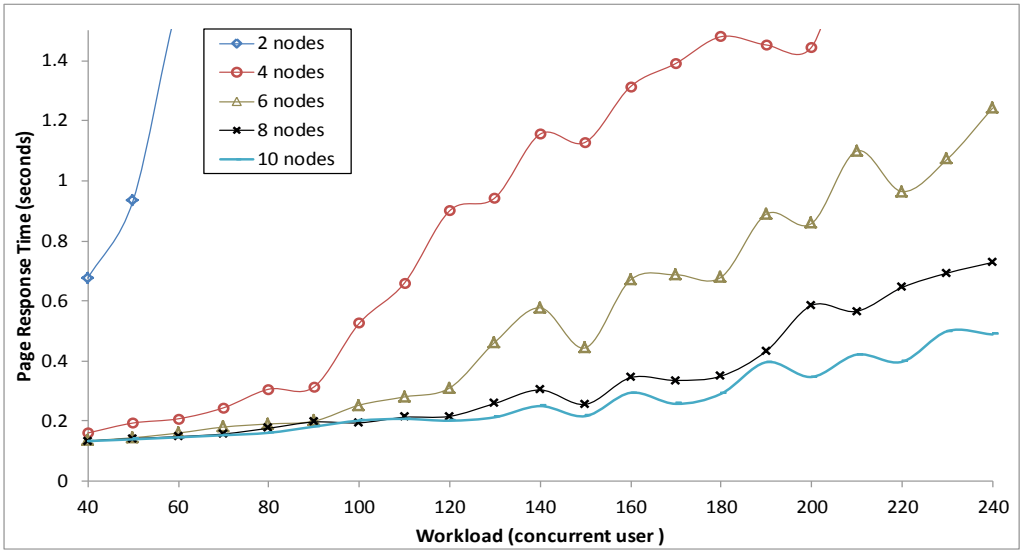


Figure 5.6 Reader tier profiling

Performance profiles relate the workload of the *Reader* and *Loader Tier*s to their resource demands according to the desired performance. Taking the predicted workload $w(t+1)$ as the input, these profiles are used by the *Resource Allocator* to allocate resources to the *Reader* and *Loader Tier*s dynamically in order to achieve the desired QoS.

For the *Reader Tier*, since the workload consists of online web requests, the intensity of the workload is specified by request rate $w(t)$ as discussed in Section 4.2. The relevant performance metric is the average response time $RT(t)$ of the requests completed during each control period (e.g., one hour). It can be considered as a function of the workload and the number of VM nodes $rR(t)$ allocated to *Reader Tier*. Thus,

$$\text{Eq. 4: } RT(t) = \emptyset(w(t), rR(t))$$

The strategy to build this mapping is offline profiling. Given a specific workload *w*, we map the number of nodes *n* allocated to the *Reader Tier* to the performance *RT* by iterating over the allocation space and collecting corresponding performance measurements under each allocation candidate. Then we repeat the above step under different workloads by varying the number of the concurrent users in v-TerraFly. Figure 5.6 illustrates the mapping results by using two to ten *Reader* nodes to serve a workload with 40 to 240 concurrent users. Such a mapping provides the least number of VM nodes needed for a given workload to meet a specific QoS target. For example, if desired response time is set to 0.7 second, then the minimal number of VMs needed is two for a workload with about 40 users and 10 when there are more than 230 concurrent users.

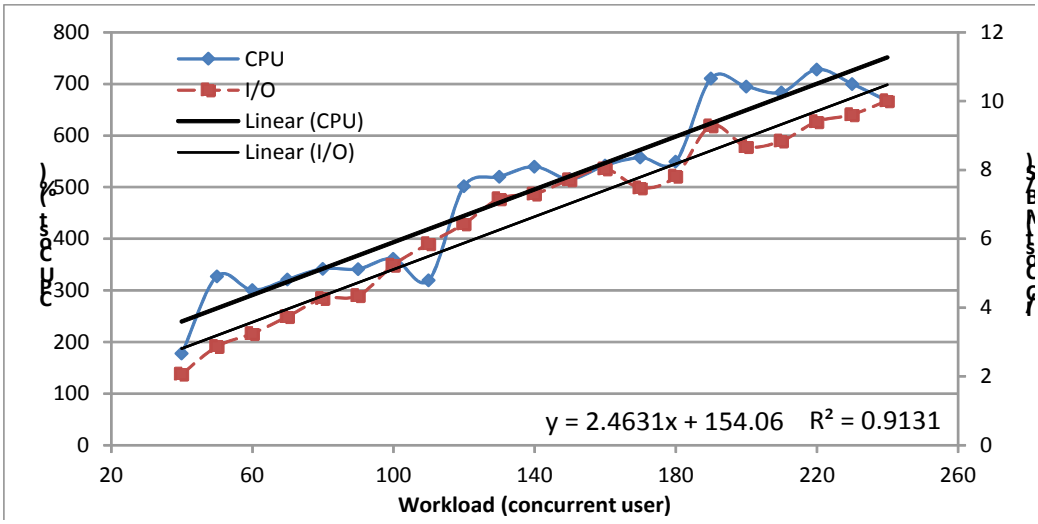


Figure 5.7 Reader tier profiling

In order to reduce the time required for performance profiling, we collected only a subset of the *Reader* configurations under a subset of the workload intensities, and use linear regression to build the *Reader Tier* entire performance profile. As shown in Figure 5.7, we got the profile mode and the R square is 0.9131.

We profiled both the CPU and I/O resource usages of the *Reader Tier*, as shown in Figure 5.7. The results show that both the CPU and I/O demands follow the exact same pattern as the workload varies, which validates the use of identical VM nodes as the resource allocation unit of the *Reader Tier*.

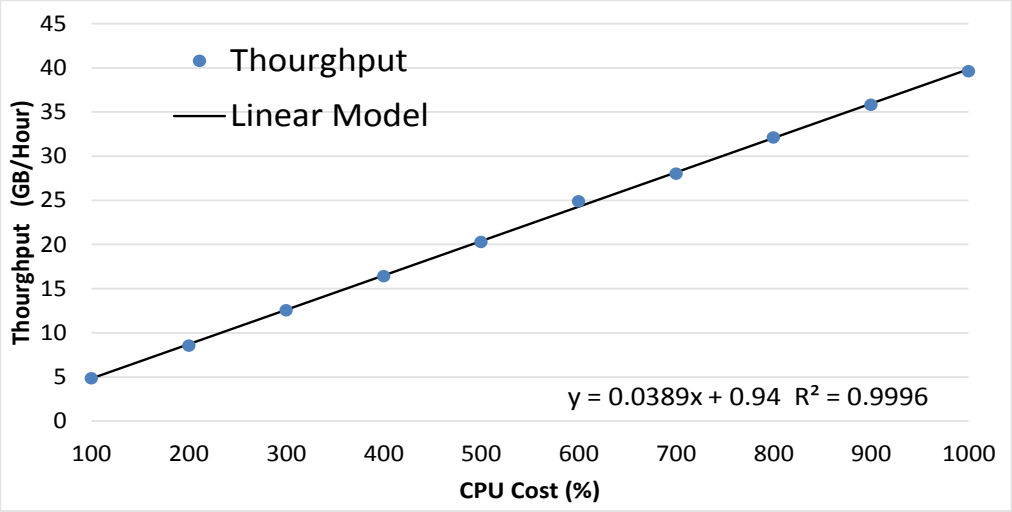


Figure 5.8 Loader tier CPU usage profiling

For the *Loader Tier*, since the workload mainly consists of batch jobs which loads raw data into organized repository, the workload intensity is given by the concurrency level, i.e., the average throughput achieved every control period. Allocating more VMs to the *Loader Tier* allows it to obtain higher throughout and finish the loading process sooner. We use offline profiling to create a model for *Loader* that represents the relationship between the throughput (I/O) and the number of VM nodes for the *Loader Tier*. We use different amount of map *Loader* nodes to load a given imagery dataset and monitor the throughput. We then use linear regression to learn the entire model based on the training data. As shown in Figure 5.8, the throughput of *Loader Tier* is almost linear with respect to the number of *Loader* VMs. Using linear regression, the R square is only 0.9996.

5.4.4 QoS Model

In this section, we propose a novel QoS model to consider both the responsiveness in serving user mapping requests and the quality of returning geographic information. In a virtualized web mapping system, both *Reader* and *Loader* VMs are usually co-hosted in a cluster/data center and compete for the common physical resources, while the former guarantees acceptable response time and the latter keeps the imagery data up to date. Since the performances from both tiers are critical, we need to well balance the importance between them especially when the total resource capacity is constrained. Therefore, a new QoS model is defined to represent the overall system performance at measuring time period t.

$$\text{Eq. 5: } QoS(t) = r(t) \times f(t)$$

where r(t) and f(t) are the performance metrics for the *Reader* and *Loader Tier* respectively.

The former QoS component r(t) is called the normalized response time, which measures the quality of web mapping services at time t and can be calculated as following:

$$\text{Eq. 6: } r(t) = {RT_{ref}}\Big/{RT(t)}$$

where RT_{ref} is the desired average response time at *Reader Tier* and $RT(t)$ is the actual performance measurement at time t. The higher of the value, the quicker the user requests served at *Reader Tier*.

The latter QoS component $f(t)$ is called the cumulative data freshness which measures the quality of data-loading process at *Loader Tier* at time t. It is calculated

recursively based on the previous data freshness value and the current data incremental rate, expressed as following:

$$\text{Eq. 7: } f(t) = (1-\rho)\times f(t-1) + \Delta D(t)/D_{ref}$$

where ρ is the decaying factor (predefined in the range of 0 to 1) indicating the data quality loss in terms of freshness during the past time period; D_{ref} is the desired amount of fresh data per time period, $\Delta D(t)$ is the actual amount of data loaded during time period t. Initially phase $t = 0, f(0) = \Delta D(0)/D_{ref}$.

For instance, assuming we need to maintain $D_{ref} = 300GB$ amount of fresh data in repository every control period and the data freshness value at previous time period was $f(t-1) = 0.9$ will be reduced to 0.864 at current time t given a decaying factor of 4.0%. If there is 15GB data loaded during current time period, then the incremental rate is 0.05 and therefore the freshness value $f(t) = 0.864 + 0.05 = 0.914$. Intuitively, we can say the current data quality is 91.4% fresh. Note that it is evident that the data freshness can be adjusted by controlling data incremental rate via resource management at *Loader Tier*.

By maximizing the QoS as computed above, the v-TerraFly resource management system automatically optimize the response time and data freshness simultaneously, which are both important to the map service received by users.

5.5 Evaluation

5.5.1 Setup

This section evaluates the proposed virtual web map service system and its autonomic resource management using the v-TerraFly prototype and real traces collected

from the TerraFly production system. As a typical web application, TerraFly usually provides a variety of web services via IIS (Internet Information Services) to serve online web requests. The test bed is set up on two Dell PowerEdge 2970 servers, each with two six-core 2.4GHz AMD Opteron CPUs, 32GB of RAM, and one 1TB 7.2 RPM SAS disk. Windows Server 2008 and Hyper-V are installed to provide the virtualization environment for v-TerraFly. The resource management system for v-TerraFly is hosted on the hypervisor's management VM. All guest VMs including both *Reader* and *Loader Tier* of TerraFly are installed Windows Server 2008 Data Center as the OS. Each Reader and Loader VM is configured with one core CPU, 2G memory, and 64 GB disk. The resource allocation is done by starting or stopping VMs via Hyper-V PowerShell Script.

5.5.2 Workload Prediction

We first evaluate the accuracy of our proposed two-level DES workload prediction algorithm by comparing it to two one-level DES approaches based on hourly pattern only (*Horizontal*) and daily pattern only (*Vertical*) respectively, as well as history average statistics (*History Average*). The evaluation is performed using a real one-month workload trace of the November 2012 extracted from the production TerraFly system's logs. To conduct the experiment more efficiently, the real trace is replayed with a 60-fold speedup, i.e., using one minute in the experiment to simulate one hour in real world. The prediction and updates of the workload models (*smoothing weights* α_h and α_d, refer to Eq. 2 and Eq. 3) are performed every minute to adapt to the dynamics.

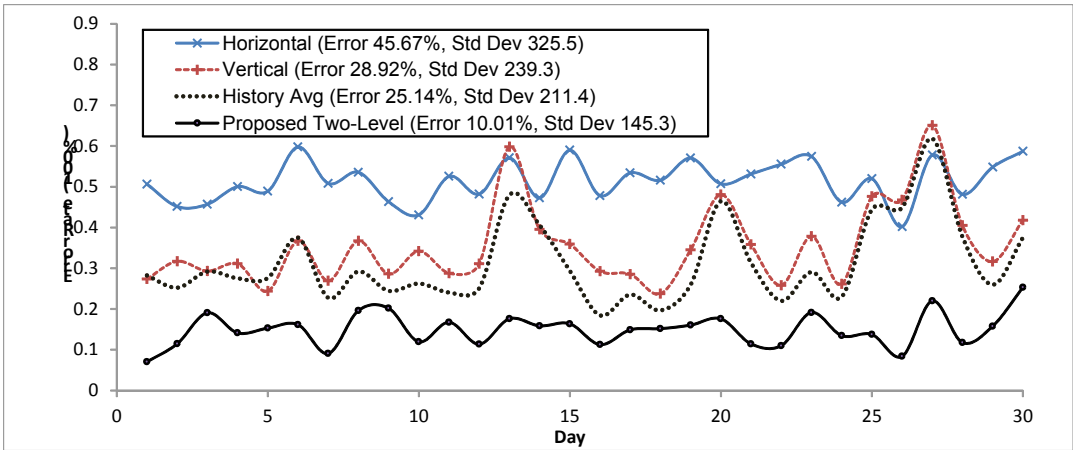


Figure 5.9 Error and Standard deviation of different workload prediction approaches

Figure 5.9 compares the online prediction errors of different approaches. Our proposed two-level prediction method delivers significantly better accuracy in predicting the request rate of one month workload. Overall, the 90 percentile average error rate of our two-level method is 10.01% with the lowest standard deviation of 145.3, both much lower than the other three prediction approaches which are 45.67% (*Horizontal*), 28.92% (*Vertical*), and 25.14% (*History Average*) respectively. These results demonstrate that our proposed method can effectively exploit both the hourly pattern and daily pattern in the workload and achieve accurate workload prediction.

5.5.3 Resource Management of *Reader Tier*

As discussed in Section 4.3, based on the predicted workload, the v-TerraFly resource management system automatically allocates resources to the *Reader* and *Loader Tier*s in order to optimize the QoS. This section evaluates the resource management for the *Reader Tier* alone and demonstrates whether it can achieve the *Reader Tier*'s response time target with the least amount of resources.

In the experiment, a real daily workload trace of October 4th 2012 is replayed against v-TerraFly with a 60-fold speedup. The resource allocator adjusts resources allocation every 1 minute. The QoS target is set to 0.7s in response time. Based on the profiling, the performance model of *Reader Tier* is $R(t) = 2.4631\ W(t) + 154.06$, (as shown in Fig.7).

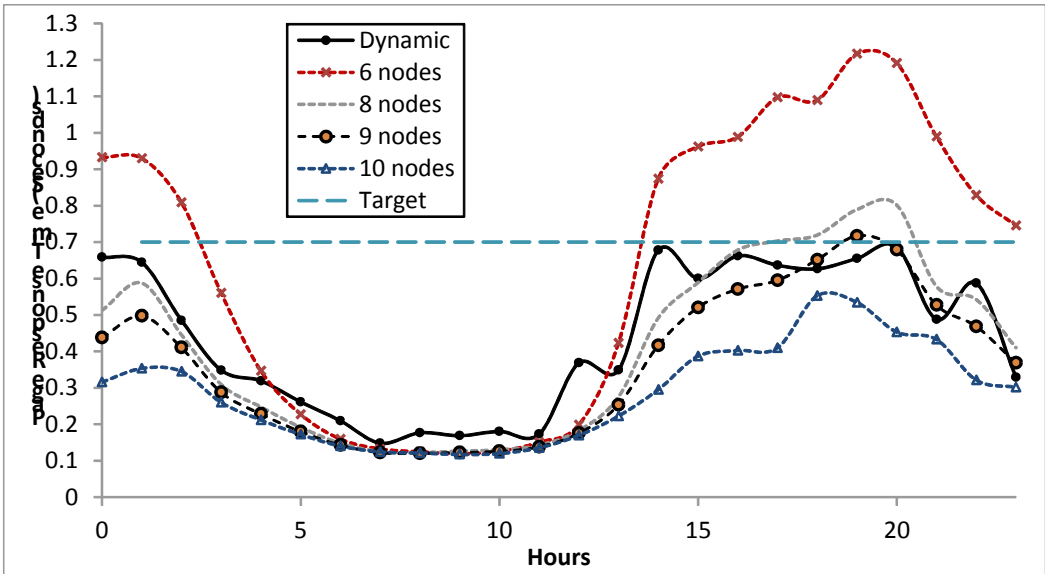


Figure 5.10 Result: Response time

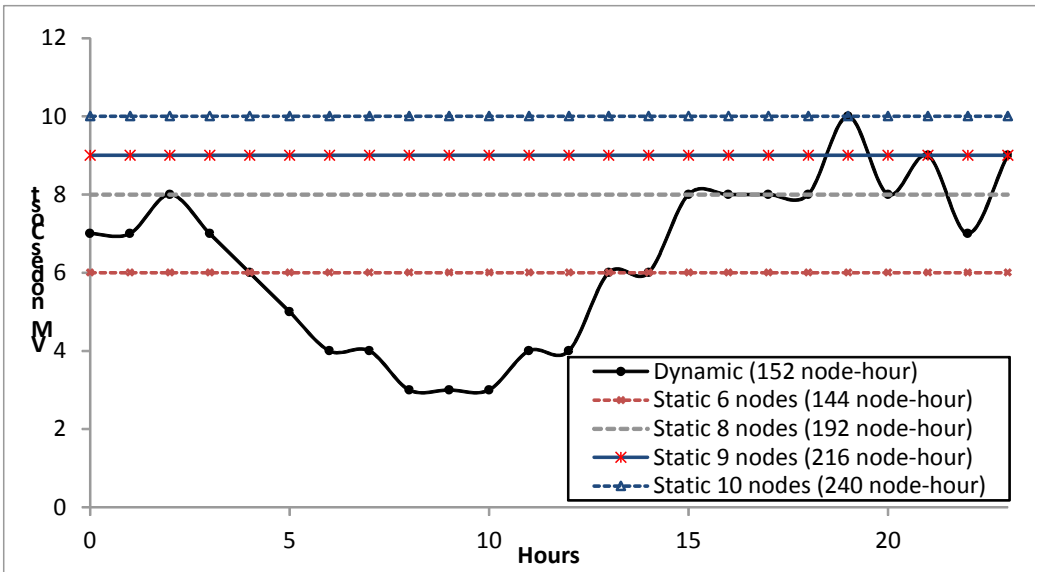


Figure 5.11 Result: VM nodes cost by hours and total VM nodes Cost

Figure 5.10 and Figure 5.11 compare the response time and allocations of our dynamic approach to static 6, 8, 9 and 10-node deployment plans respectively. From the results, we can see that the online dynamic approach is able to achieve the response time target all the time throughout the entire experiment. Compared to the 6-node static plan which achieves 0.634 second in average page response time, the online dynamic plan improves the performance by 32.18%, i.e., 0.430 second in average response time, but at the cost of only 5.6% more resource allocation, i.e., 8 additional node-hour. There are 13 data points where its response time exceeds 0.7 second in the 6-node static plan, which causes as much as 54.17% QoS violation; in contrast, no QoS violation occurs in the dynamic plan.

Compared to the 8-node static plan, the dynamic plan saves as much as 20.83% of total resources, i.e., 40 *node-hour*. Although the static 8-node plan allocates substantially more with surplus resources, it still causes three QoS violations during the experiment. The 9-node static plan meets the QoS target all the time except the 19th hour, and it costs 29.63% more total resources than the dynamic plan. The 10-node static plan is the only static plan that meets the QoS target all the time, but it costs 36.67% more total resources than the dynamic plan.

Overall, it is evident that the online dynamic deployment plan can efficiently allocate resources to the *Reader Tier* while at the same time meet the response time target by flexibly adjusting its VM assignments in an online manner.

5.5.4 Resource Management of both *Reader* and *Loader Tier*s

This section evaluates the proposed autonomic resource management approach for both *Reader* and *Loader Tier*s. Based on the QoS model defined in Section 4.4, the importance of both tiers needs to be balanced in order to optimize an overall QoS value which not only guarantees the responsiveness of map service but also maintains the data freshness of returned maps.

In the experiment, we use the same workload trace described in Section 5.3, and compare our proposed approach to the traditional static deployment plan. The traditional method allocates a fixed number of nodes to *Reader Tier* to satisfy average response time by past experience and gives rest nodes to *Loader Tier*. Specifically, it assigns 7 nodes to the *Read Tier* and 3 nodes to the *Loader Tier* in order to achieve an average response time of 0.9 second and 303.3GB fresh data per day (average $f(t) = 0.809$, refer to Eq. 7).

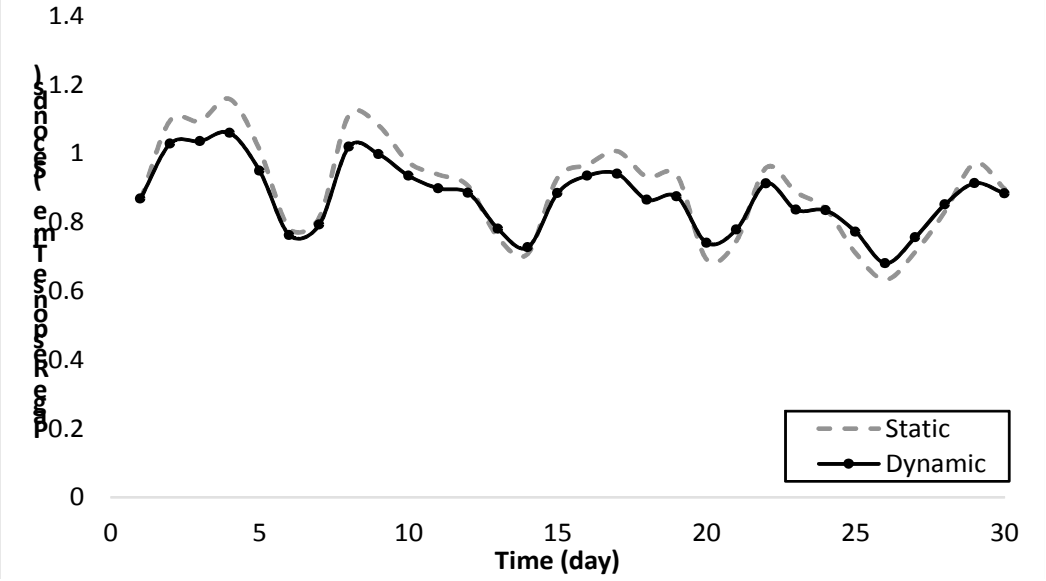


Figure 5.12 Result: response time improvement

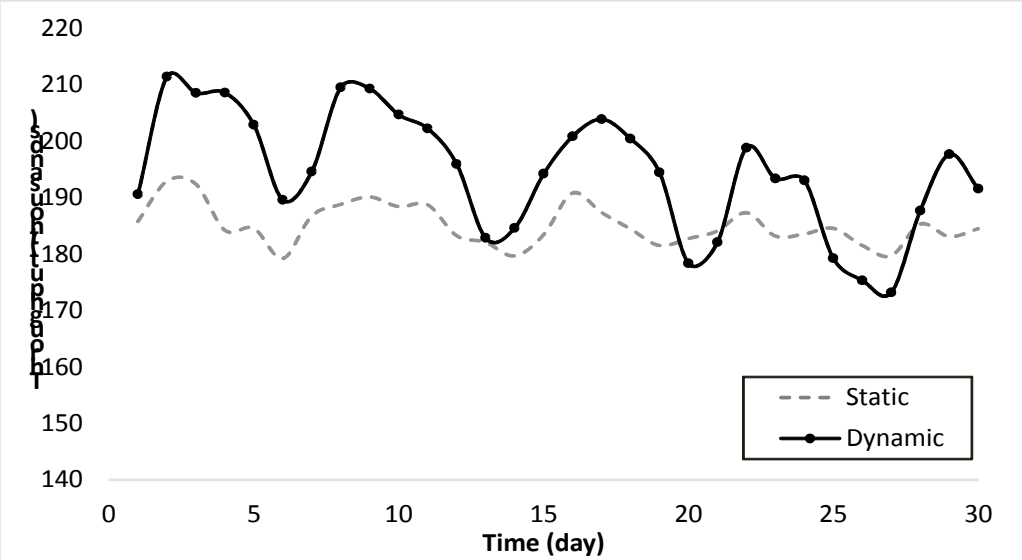


Figure 5.13 Result: Throughput improvement

Figure 5.12 and Figure 5.13compare the performance of the proposed dynamic plan to the traditional static plan in both response time of the *Reader Tier* and throughput of the *Loader Tier*. The static plan achieves an average response time of 0.897 seconds, while the dynamic plan shows slightly better 0.873 seconds. The latter also achieves higher average throughput (194.6 thousands requests per day) than the static one (185.1 thousands requests per day).

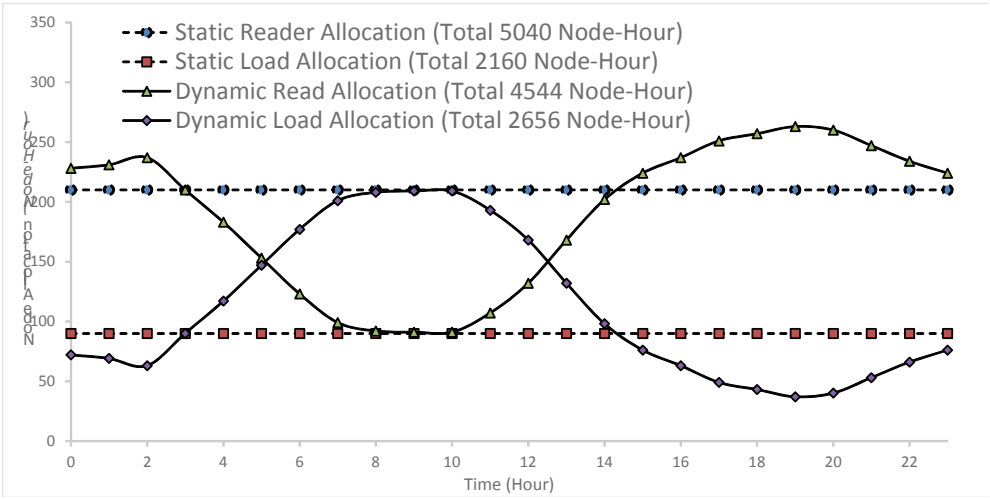


Figure 5.14 Node Allocation per Hour

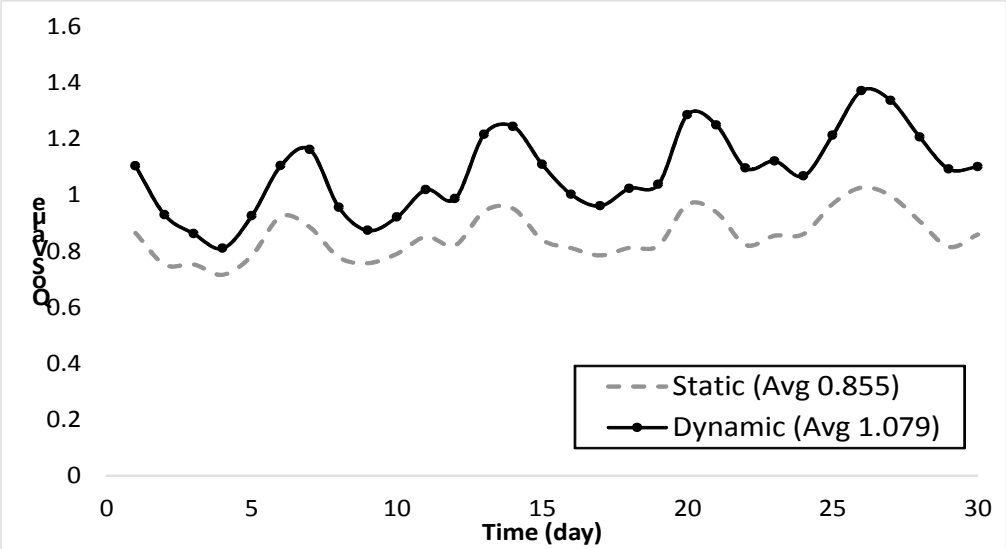


Figure 5.15 Result: QoS value improvement

Although the performance improvement on the *Reader Tier* is not significant, Figure 5.15 shows that the proposed dynamic plan achieves much better overall QoS (26.19% improvement). The reason behind this substantial improvement is because the dynamic plan saves resources from the *Reader Tier* and allocates them to the *Loader Tier*, thereby making data loading faster without sacrificing *Reader* performance. Resources are dynamically balanced between these two tiers as the workload changes, where the autonomic resource management allocates only the necessary number of VM nodes to the *Reader Tier* to satisfy current workload, and reserve the rest to *Loader Tier* to load new data.

For example, as showed in Figure 5.14, from Hour 7 to 10, since the workload on *Reader Tier* is less intense, the dynamic plan allocates more resource to the *Loader Tier* to allow the new data to be loaded as fast as possible. As a result, the dynamic plan loads much more new data (473.3GB Per day) at a varying loading rate than the traditional plan (303.3GB Per day), which loads data at a fixed rate. And the QoS value of the dynamic

plan (*Avg QoS=1.079,* refer to Eq. 5) is *26.20%* higher the traditional plan (*Avg QoS=0.855,* refer to Eq. 5).

In summary, our proposed autonomic resource management approach is able to automatically optimize the tradeoff between service responsiveness and data freshness by balancing the resource allocations between the *Reader* and *Loader Tiers.*

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this dissertation, we have improved a distributed disk-resident hybrid index for efficiently answering k-NN queries with Boolean constraints on textual content with MapReduce in sksOpen project. With this algorithm, we have implemented efficient online indexing, querying, and visualization system for Big Geospatial data, to allow online users to index their own spatial data, and offer query visualization. Our experimental study showed an improved performance and scalability on large spatial datasets over alternate methods, and a better interactive user interface.

Web map services become increasingly widely used for various commercial and personal purposes. GIS application needs to be able to easily analyze and visualize spatial data and satisfy the increasing demand of information sharing. TerraFly GeoCloud is an online spatial data analysis and visualization system, to address the challenges. TerraFly GeoCloud is built upon the TerraFly Geo spatial database, to offer a convenient way to analyze geo spatial data, visualize the results, and share the result by a unique URL. Our system also allows users to customize their own spatial data visualization using a SQL-like MapQL language rather than writing codes with Map API.

Virtualization can greatly facilitate the deployment of web map service systems and substantially improve their resource utilization. To fulfill this potential, the resource management of a virtualized web map system needs to be able to handle the dynamic workloads that the system typically serves and satisfy the often competing demands of

the various tiers of the system. v-TerraFly is presented to address these challenges. v-TerraFly is created by virtualizing the various tiers of a typical map service system and allowing resources to be dynamically allocated across the tiers. The resource management is done by predicting the workload intensity based on historical data and estimating the resource needs of the map service's Reader and Loader Tiers based on their performance models. A unique QoS metric is then defined to capture the tradeoff between the service responsiveness and data freshness, and it is used to optimize the resource allocation to the Reader and Loader VMs.

Experiments based on real TerraFly workload show that our system can accurately predict the workload's resource demands online and automatically allocate the resources accordingly to meet the performance target and save substantial resource cost compared to peak-load-based resource allocation. It can also automatically optimize the tradeoff between responsiveness and data freshness by dynamically balancing the shared resources between the Reader and Loader VMs.

6.2 Future Work

In our future work, we will research and develop an extra layer between end users who have limit knowledge in writing SQL statements and the MapQL, a query composing interfaces for the MapQL statements, to facilitate lay users to create their own map visualizations. Also, we will improve the scale of TerraFly GeoCloud, conduct large-scale experiments and employ distributed computing as additional mechanisms for optimizing the system. In addition, we will explore how to apply the principle of MapQL to other applications that share similar characteristics with web GIS services.

Also, we will improve the scale of v-TerraFly, conducting larger experiments and employing live VM migration as an additional mechanism for optimizing resource management. We will also explore how to apply the principle of v-TerraFly to other applications that have similar dynamic and multi-tier characteristics as a web map service.

LIST OF REFERENCES

[1] Rishe, N., Chen, S. C., Prabakar, N., Weiss, M. A., Sun, W., Selivonenko, A., & Davis-Chu, D. (2001, April). TerraFly: A high-performance web-based digital library system for spatial data access. In The 17th IEEE International Conference on Data Engineering (ICDE), Heidelberg, Germany (pp. 17-19).

[2] N. Rishe, M. Gutierrez, J. Janvier. "A View from Above Without Leaving the Ground." NASA Spinoff 2004, pp 72-73.

[3] Bo Xu, Ouri Wolfson, Sam Chamberlain, Naphtali Rishe "Cost Based Data Dissemination in Satellite Networks" Mobile Networks & Applications, Vol. 7 (2002), No. 1, pp 49-66.

[4] Jaime Ballesteros, Mahmudur Rahman, Bogdan Carbunar, Naphtali Rishe. "Safe Cities. A Participatory Sensing Approach". 37th IEEE Conference on Local Computer Networks (LCN), October 22-25, 2012. pp. 626-634.

[5] Piotr Szczurek, Bo Xu, Ouri Wolfson, Jie Lin, Naphtali Rishe. "Learning the relevance of parking information in VANETs". Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking. Chicago, Illinois. September 24, 2010. ISBN:978-1-4503-0145-9. pp 81-82, September 2010.

[6] Piotr Szczurek, Bo Xu, Ouri Wolfson, Jie Lin, Naphtali Rishe. "Prioritizing Travel Time Reports in Peer-to-Peer Traffic Dissemination". Proceedings of the IEEE International Symposium on Communication Systems, Networks and Digital Signal Processing (7th CSNDSP). Newcastle, U.K. July 21-23, 2010. pp 454-458.

[7] Daniel Ayala, Jie Lin, Ouri Wolfson, Naphtali Rishe, Masaaki Tanizaki. "Communication Reduction for Floating Car Data-based Traffic Information Systems". Second International Conference on Advanced Geographic Information Systems, Applications, and Services, pp 44-51, February 10-16, 2010. Best Paper Award.

[8] N. Rishe. Database Design: The Semantic Modeling Approach. McGraw-Hill, 1992, 528 pp.

[9] N. Rishe. Database Design Fundamentals: A Structured Introduction to Databases and a Structured Database Design Methodology. Prentice-Hall, Englewood Cliffs, NJ, 1988. 436 pp. (Also printed as International Edition. Prentice-Hall International, Inc., 1988. 432 pp. ISBN 0-13-197476-9.)

[10] Lu, Y., Zhang, M., Tao Li, Y. G., & Rishe, N. (2013). Online Spatial Data Analysis and Visualization System. ACM KDD IDEA 2013, pp.73-79

[11] Anick Jesdanun, AP (2008-01-18). "GPS adds dimension to online photos". Retrieved 2008-01-19.

[12] Wiki Project on Geotagging. http://en.wikipedia.org/wiki/Geotagging#cite_note-2

[13] P. Martin, S. Elnaffar and T. Wasserman, “Workload Models for Autonomic Database Management Systems”, ICAS, 2006.

[14] P. Padala, K. Hou, K. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal and A. Merchant, “Automated Control of Multiple Virtualized Resources”, SIGOPS/EuroSys, 2009

[15] Hariharan, R., Hore, B., Li, C., & Mehrotra, S. (2007, July). Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In Scientific and Statistical Database Management, 2007. SSBDM'07. 19th International Conference on (pp. 16-16). IEEE.

[16] Hjaltason, G., Samet, H.: Distance browsing in spatial databases. ACM Trans.Database Syst. 24 (2), 265–318, 1999.

[17] Guttman, A.: R-trees: A dynamic index structure for spatial searching. In SIGMOD, pp. 47–57. ACM, New York, 1984.

[18] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: An efficient and robust access method for points and rectangles. In SIGMOD, pp. 322–331, 1990.

[19] Sellis, T.K., Roussopoulos, N., Faloutsos, C.: The R+-tree: A dynamic index for multi-dimensional objects. In VLDB, pp. 507–518, 1987.

[20] Kamel, I., Faloutsos, C.: Hilbert R-tree: An improved R-tree using fractals. In VLDB, pp. 500–509, 1994.

[21] Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. 38 (2), 2006.

[22] Zobel, J., Moffat, A., Ramamohanarao, K.: Inverted files versus signature files for text indexing. ACM Trans. Database Syst. 23 (4), 453–490, 1998.

[23] R.R. Larson. Geographic Information Retrieval and Spatial Browsing. In GIS and Libraries: Patrons, Maps and Spatial Information, pages 81-125, 1996.

[24] C.B. Jones, A.I. Abdelmoty, D. Finch, G. Fu, and S. Vaid. The Spirit Spatial Search Engine.: Architecture, Ontologies and Spatial Indexing. In Proc. of GIScience, pages 125-139, October 2004.

[25] A. Markowetz, Y. Chen, T. Suel, X. Long, and B. Seeger. Design and Implementation of a Geographic Search Engine. In Proc. of WebDB, June 2005.

[26] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W. Ma. Hybrid Index Structures for Location-Based Web Search. In Proc. Of CIKM, pages 155-162, November 2005

[27] Hariharan, R., Hore, B., Li, C., Mehrotra, S.: Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In SSDBM, p. 16, 2007.

[28] Park, D.J., Kim, H.J.: An enhanced technique for k-nearest neighbor queries with non-spatial selection predicates. Multimedia Tools and Apps., 79–103, 2004.

[29] Chang, W.W., Schek, H.J.: A signature access method for the Starburst database system. In VLDB, pp. 145–153, 1989.

[30] De Felipe, I., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In ICDE, pp. 656–665, IEEE Computer Society, 2008.

[31] Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. Proc. VLDB Endow. 2 (1), 337–348, 2009.

[32] Cary, A., Wolfson, O., & Rishe, N. (2010, January). Efficient and scalable method for processing top-k spatial boolean queries. In Scientific and Statistical Database Management (pp. 87-95). Springer Berlin Heidelberg.

[33] Xiaoyan Li, Sharing geoscience algorithms in a Web service-oriented environment, Computers & Geosciences Volume 36, Issue 8, August 2010

[34] Fotheringham, S., & Rogerson, P. (Eds.). (2004). Spatial analysis and GIS. CRC Press.

[35] Anselin, L. (1999). Interactive techniques and exploratory spatial data analysis. Geographical Information Systems: principles, techniques, management and applications, 1, 251-264.

[36] Johnston, K., Ver Hoef, J. M., Krivoruchko, K., & Lucas, N. (2001). Using ArcGIS geostatistical analyst (Vol. 380). Redlands: Esri.

[37] Anselin, L., Syabri, I., & Kho, Y. (2006). GeoDa: An introduction to spatial data analysis. Geographical analysis, 38(1), 5-22.

[38] Boyer, D., Cheetham, R., & Johnson, M. L. (2011). Using GIS to Manage Philadelphia's Archival Photographs. American Archivist, 74(2), 652-663.

[39] Hearnshaw, H. M., & Unwin, D. J. (1994). Visualization in geographical information systems. John Wiley & Sons Ltd.

[40] Boyer, D. (2010). From internet to iPhone: providing mobile geographic access to Philadelphia's historic photographs and other special collections. The Reference Librarian, 52(1-2), 47-56.

[41] Forecasting with Exponential Smoothing: The State Space Approach, Hyndman, R., Koehler, A.B., Ord, J.K., Snyder, R.D. 2008, XIII, 362 p

[42] A state space framework for automatic forecasting using exponential smoothing methods International Journal of Forecasting, 18 (2002), pp. 439–454

[43] The fundamental theorem of exponential smoothing, RG Brown, RF Meyer - Operations Research, 1961 or.journal.informs.org

[44] Exponential smoothing: The state of the art, ES Gardner Jr - International Journal of Forecasting, 1985 - Elsevier

[45] Xiaoyan Li, Sharing geoscience algorithms in a Web service-oriented environment, Computers & Geosciences Volume 36, Issue 8, August 2010

[46] Peng Yue, Semantics-based automatic composition of geospatial Web service chains, Computers & Geosciences Volume 33, Issue 5, May 2007

[47] Huebscher, M. C., & McCann, J. A. (2008). A survey of autonomic computing—degrees, models, and applications. ACM Computing Surveys (CSUR), 40(3), 7.

[48] Horn, P. (2001). Autonomic computing: IBM\'s Perspective on the State of Information Technology.

[49] Muscettola, N., Morris, P., & Tsamardinos, I. (1998). Reformulating temporal plans for efficient execution. In In Principles of Knowledge Representation and Reasoning.

[50] Morato, Daniel, et al. "On linear prediction of Internet traffic for packet and burst switching networks." Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on. IEEE, 2001.

[51] Dinda, Peter, et al. "The case for prediction-based best-effort real-time systems." Parallel and Distributed Processing 1999

[52] J. Wildstrom, P. Stone and E. Witchel, “CARVE: A Cognitive Agent for Resource Value Estimation”, ICAC, 2008.

[53] J. Rao, X. Bu, C. Xu, L. Wang and G. Yin, “VCONF: A Reinforcement Learning Approach to Virtual Machines Auto-configuration”, ICAC, 2009.

[54] L. Wang, J. Xu, M. Zhao, Y. Tu and J. A.B. Fortes, “Fuzzy Modeling Based Resource Management for Virtualized Database Systems”, MASCOTS. 2011

[55] J. Xu, M. Zhao and J. Fortes, “Autonomic Resource Management in Virtualized Data Centers Using Fuzzy-logic-based Control”, Cluster Computing, 2008.

[56] P. Padala, K. Hou, K. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal and A. Merchant, "Automated Control of Multiple Virtualized Resources", SIGOPS/EuroSys, 2009.

[57] X. Liu, X. Zhu, P. Padala, Z. Wang and S. Singhal,"Optimal Multivariate Control for Differentiated Services on a Shared Hosting Platform", CDC, 2007.

[58] P. Lama and X. Zhou, "PERFUME: Power and Performance Guarantee with Fuzzy MIMO Control in Virtualized Servers", IWQoS, 2011.

[59] L. Wang, et al., "Adaptive Virtual Resource Management with Fuzzy Model Predictive Control" FeBID, 2011.

[60] Lester Melendez, Ouri Wolfson, Malek Adjouadi, Naphtali Rishe. "Qualitative Analysis of Commercial Social Network Profiles". Handbook of Social Network Technologies and Applications. Borko Furht, editor. Springer Verlag, 2010. pp 95-114.

[61] N. Rishe, M. Gutierrez, A. Selivonenko, S. Graham. "TerraFly: A Tool for Visualizing and Dispensing Geospatial Data." Imaging Notes, Summer 2005, Vol. 20, No. 2. pp 22-23.

[62] Rishe, N., Sun, Y., Chekmasov, M., Selivonenko, A., & Graham, S. (2004, December). System architecture for 3D terrafly online GIS. In Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on (pp. 273-276). IEEE.

[63] Naphtali Rishe. U.S. Patent 6,795,825 "A Database Querying System and Method" issued 2004.09.21.

[64] Naphtali Rishe. U.S. Patent 6,339,773 "Data extractor" issued 2002.01.15.

[65] Naphtali Rishe, Borko Furht, Malek Adjouadi, Armando Barreto, Evgenia Cheremisina, Debra Davis, Ouri Wolfson, Nabil Adam, Yelena Yesha, Yaacov Yesha. "Geospatial Data Management With TerraFly." Handbook of Data Intensive Computing. Furht and Escalante, eds. Springer Verlag, 2011. pp.637-665.

[66] Rishe, N., Gutierrez, M., Selivonenko, A., & Graham, S. (2005). TerraFly: A tool for visualizing and dispensing geospatial data. Imaging Notes, 20(2), 22-23.

[67] Naphtali Rishe. TerraAtlas: Central Washington, DC. The McDonald & Woodward Publishing Company, Blacksburg, Virginia, 2006. ISBN 0-939923-99-8.

[68] Yi Zhang and Tao Li. DClusterE: A Framework for Evaluating and Understanding Document Clustering Using Visualization. ACM Transactions on Intelligent Systems and Technology, 3(2):24, 2012.

[69] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[70] Wikipedia, Apache Hadoop http://en.wikipedia.org/wiki/Apache_Hadoop

[71] IBM What is MapReduce. http://www-01.ibm.com/software/data/infosphere/hadoop/mapreduce/

[72] Wikipedia, k-nearest neighbor algorithm. http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm

[73] Bremner, D., Demaine, E., Erickson, J., Iacono, J., Langerman, S., Morin, P., & Toussaint, G. (2005). Output-sensitive algorithms for computing nearest-neighbour decision boundaries. Discrete & Computational Geometry, 33(4), 593-604.

[74] Shu-Ching Chen, Xinran Wang, Naphtali Rishe, and Mark Allen Weiss. "A Web-Based Spatial Data Access System Using Semantic R-trees." Information Sciences: An International Journal. vol. 167, no. 1-4, pp 44-61, December 2004.

[75] Cary, A., Wolfson, O., & Rishe, N. (2010, January). Efficient and scalable method for processing top-k spatial boolean queries. In Scientific and Statistical Database Management (pp. 87-95). Springer Berlin Heidelberg.

[76] N. Rishe, A. Shaposhnikov. U.S. Patent 5,920,857 "Efficient Optimistic Concurrency Control and Lazy Queries for Databases and B-Trees" issued 1999.07.06.

[77] C. Yu, K.L. Liu, W. Meng, Z. Wu, N. Rishe. "A Methodology to Retrieve Text Documents from Multiple Databases". IEEE Transactions on Knowledge and Data Engineering. Vol. 14 (2002) nbr 6 pp 1347-1361.

[78] W. Meng, K. Liu, C. Yu, W. Wu, and N. Rishe. "A Statistical Method for Estimating the Usefulness of Text Databases." IEEE Transactions on Knowledge and Data Engineering. Vol. 14 (2002) nbr 6 pp 1422-1437.

[79] Morton, G. M. (1966), A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing, Technical Report, Ottawa, Canada: IBM Ltd.

[80] Naphtali Rishe, Borko Furht, Malek Adjouadi, Armando Barreto, Debra Davis, Ouri Wolfson, Yelena Yesha, Yaacov Yesha. "Semantic Wrapper: Concise Semantic Querying of Legacy Relational Databases." Handbook of Data Intensive Computing. Furht and Escalante, eds. Springer Verlag, 2011. pp.415-444.

[81] Bern, M.; Eppstein, D.; Teng, S.-H. (1999), "Parallel construction of quadtrees and quality triangulations", Int. J. Comp. Geom. & Appl. 9 (6): 517–532

[82] Jaime Ballesteros, Mahmudur Rahman, Bogdan Carbunar, Naphtali Rishe. “Safe Cities. A Participatory Sensing Approach”. 37th IEEE Conference on Local Computer Networks (LCN), October 22-25, 2012. pp. 626-634.

[83] Spence, R., & Press, A. (2000). Information visualization.

[84] Wang, H. (2011). A Large-scale Dynamic Vector and Raster Data Visualization Geographic Information System Based on Parallel Map Tiling.

[85] Teng, W., Rishe, N., & Rui, H. (2006, May). Enhancing access and use of NASA satellite data via TerraFly. In Proceedings of the ASPRS 2006 Annual Conference.

[86] Wang, H. (2011). A Large-scale Dynamic Vector and Raster Data Visualization Geographic Information System Based on Parallel Map Tiling.

[87] N. Rishe and B. Wongsaroj. “Infrastructure for Research and Training in Database Management for Web-based Geospatial Data Visualization with Applications to Aviation.” Infrastructure 2003: NSF CISE/EIA RI and MII PI's Workshop. Arlington, VA, August 17-19, 2003. pp. 48-52.

[88] Bogdan Carbunar, Mahmudur Rahman, Jaime Ballesteros, Naphtali Rishe. “Private Location Centric Profiles for Online Social Networks.” Proceedings of the 20th International Conference on Advances in Geographic Information Systems (GIS 2012) November 6-9 2012. Redondo Beach, California, pp. 458-461.

[89] De Knegt, H. J., Van Langevelde, F., Coughenour, M. B., Skidmore, A. K., De Boer, W. F., Heitkönig, I. M. A., ... &Prins, H. H. T. (2010). Spatial autocorrelation and the scaling of species-environment relationships. Ecology, 91(8), 2455-2465.

[90] Li,Hongfei; Calder, Catherine A, "Beyond Moran's I: Testing for Spatial Dependence Based on the Spatial Autoregressive Model". Geographical AnalysisCressie, Noel (2007).

[91] Naphtali Rishe, Maxim Chekmasov, Marina Chekmasova, Scott Graham, Ian De Felipe. “On-demand Geo-referenced TerraFly Data Miner.” International Conference on Intelligent User Interfaces, Miami, FL January 12-15, 2003. pp. 277-279.

[92] Anselin, L. (1995). Local indicators of spatial association—LISA. Geographical analysis, 27(2), 93-115.

[93] Ester, M., Kriegel, H. P., Sander, J., &Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. ACM SIGKDD.

[94] Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (1998). Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data Mining and Knowledge Discovery, 2(2), 169-194.

[95] Stein, M. L. (1999). Interpolation of spatial data: some theory for kriging. Springer Verlag.

[96] Naphtali Rishe, Carlos Espinal, Tajana Lucic, Yelena Yesha, Yaacov Yesha Kalai Mathee, Aileen Marty. “Geospatial Data for Intelligent Solutions in Public Health”. e-Proceedings of Vaccinology 2012, Rio de Janeiro, September 3-7, 2012.

[97] Reza Amini, Christine Lisetti, Ugan Yasavur, Naphtali Rishe.”On-Demand Virtual Health Counselor for Delivering Behavior-Change Health Interventions”. 2013 IEEE International Conference on Healthcare Informatics (ICHI'13), 2013. Philadelphia, PA, USA, September 9-11, 2013.

[98] Naphtali, et al. "System architecture for 3D terrafly online GIS." Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on. IEEE, 2004.

[99] Lai, Alvin CK, Tracy L. Thatcher, and William W. Nazaroff. "Inhalation transfer factors for air pollution health risk assessment." Journal of the Air & Waste Management Association 50.9 (2000): 1688-1699.

[100] Wakefield, Jon, and Paul Elliott. "Issues in the statistical analysis of small area health data." Statistics in Medicine 18.17-18 (1999): 2377-2399.

[101] Kulldorff M. A spatial scan statistic. Communications in Statistics-Theory and methods. 1997;26:1481–96.

[102] Besag, Julian, and James Newell. "The detection of clusters in rare diseases."Journal of the Royal Statistical Society. Series A (Statistics in Society) (1991): 143-155.

[103] Kulldorff M, Nagarwalla N: Spatial disease clusters: detection and inference. Statistics in Medicine 1995, 14:799-810.

[104] Mantel, Nathan. "The detection of disease clustering and a generalized regression approach." Cancer research 27.2 Part 1 (1967): 209-220.

[105] Openshaw, Stan, et al. "A mark 1 geographical analysis machine for the automated analysis of point data sets." International Journal of Geographical Information System 1.4 (1987): 335-358.

[106] Getis, Arthur, and J. Keith Ord. "The analysis of spatial association by use of distance statistics." Geographical analysis 24.3 (1992): 189-206.

[107] Dubin, Robin, R. Kelley Pace, and Thomas G. Thibodeau. "Spatial autoregression techniques for real estate data." Journal of Real Estate Literature 7.1 (1999): 79-96.

[108] Kelejian, Harry H., and Ingmar R. Prucha. "A generalized spatial two-stage least squares procedure for estimating a spatial autoregressive model with autoregressive

disturbances." The Journal of Real Estate Finance and Economics 17.1 (1998): 99-121.

[109] Moran, Patrick AP. "Notes on continuous stochastic phenomena." Biometrika 37.1/2 (1950): 17-23.

[110] Sagit Zolotov, Dafna Ben Yosef, Naphtali Rishe, Yelena Yesha, Eddy Karnieli. "Metabolic profiling in personalized medicine: bridging the gap between knowledge and clinical practice in Type 2 diabetes" Personalized Medicine, Vol. 8, No. 4, July 2011, pp. 445-456.

[111] Naphtali Rishe, Carlos Espinal, Tajana Lucic, Yelena Yesha, Yaacov Yesha Kalai Mathee, Aileen Marty. "Geospatial Data for Intelligent Solutions in Public Health." e-Proceedings of Vaccinology 2012, Rio de Janeiro, September 3-7, 2012.

[112] Naphtali Rishe, Yelena Yesha, Yaacov Yesha, Tajana Lucic. "Intelligent solutions in public health: models and opportunities." Proceedings of the Second Annual International Conference on Tropical Medicine: Intelligent Solutions for Emerging Diseases. February 23-24, 2012, Miami, Florida.

[113] Naphtali Rishe, Yelena Yesha, Tajana Lucic. "Data Mining and Querying in Electronic Health Records." Proceedings of Up Close and Personalized, International Congress on Personalized Medicine (UPCP 2012), Florence, Italy, February 2-5, 2012.

[114] Yelena Yesha, Naphtali Rishe, Tajana Lucic. "Clinical-Genomic Analysis using Machine Learning Techniques to Predict Risk of Disease." Proceedings of Up Close and Personalized, International Congress on Personalized Medicine (UPCP 2012), Florence, Italy, February 2-5, 2012.

[115] Aniket Bochare, Aryya Gangopadhyay, Yelena Yesha, Anupam Joshi, Yaacov Yesha, Michael A. Grasso, Mary Brady, Naphtali Rishe. "Integrating Domain Knowledge in Supervised Machine Learning to Assess the Risk of Breast Cancer". International J of Medical Engineering and Informatics, 2013

[116] Rohit Kugaonkar, Aryya Gangopadhyay, Yelena Yesha, Anupam Joshi, Yaacov Yesha, Michael Grasso, Mary Brady and Naphtali Rishe. "Finding associations among SNPs for prostate cancer using collaborative filtering". DTMBIO-12: Proceedings of the ACM sixth international workshop on Data and text mining in biomedical informatics. Hawaii, USA. October 29, 2012. ACM New York, NY.

[117] Ron Ribitzky, Yelena Yesha, Eddy Karnieli, Naphtali Rishe. "Knowledge Mining & Bio-informatics Techniques to Advance Personalized Diagnostics & Therapeutics". Report to the U.S. National Science Foundation (NSF) on the Outcomes and Consensus Recommendations of the NSF-sponsored International Workshop, February 2012, in Florence, Italy, http://CAKE.fiu.edu/HIT-

papers/Book_post_NSF_Workshop_Knowledge_Mining_and_Bioinformatics_Techniques_to_Advance_Personalized_Diagnostics_and_Therapeutics.pdf

[118] P. Martin, S. Elnaffar and T. Wasserman, “Workload Models for Autonomic Database Management Systems”, ICAS, 2006.

[119] P. Padala, K. Hou, K. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal and A. Merchant, “Automated Control of Multiple Virtualized Resources”, SIGOPS/EuroSys, 2009

[120] Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., & Jiang, G. (2009). Power and performance management of virtualized computing environments via lookahead control. Cluster Computing, 12(1), 1–15. Special Issue on Autonomic Computing.

[121] "Web Mapping." Wikipedia. Wikimedia Foundation, 28 Feb. 2013. Web. 01 Mar. 2013.

[122] Brian Craig. "Online Satellite and Aerial Images: Issues and Analysis" North Dakota Law Review 85 (2007): 547

[123] William Teng, Enhancing Access and Use of Nasa Satellite Data via Terrafly reno2006

[124] Robust Database Structures with Dynamic Query Patterns”, EJOR, 2006.

[125] T. E. Anderson, L. L. Peterson, S. Shenker, and J. S. Turner, "Overcoming the internet impasse through visualization." IEEE Computer, vol. 38, no. 4, pp. 34-41, 2005.

[126] Bennani, M. N., & Menasce, D. A. (2005, June). Resource allocation for autonomic data centers using analytic performance models. In Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on (pp. 229-240). IEEE.

[127] Network reconfiguration in distribution systems for loss reduction and load balancing, ME Baran, FF Wu - Power Delivery, IEEE Transactions on, 1989 - ieeexplore.ieee.org

[128] T. Wood, L. Cherkasova, K. Ozonat and P. Shenoy, “Profiling and Modeling Resource Usage of Virtualized Applications”, Middleware, 2008.

[129] Forecasting with Exponential Smoothing: The State Space Approach, Hyndman, R., Koehler, A.B., Ord, J.K., Snyder, R.D. 2008, XIII, 362 p

[130] A state space framework for automatic forecasting using exponential smoothing methods International Journal of Forecasting, 18 (2002), pp. 439–454

[131] The fundamental theorem of exponential smoothing, RG Brown, RF Meyer - Operations Research, 1961 or.journal.informs.org

[132] Exponential smoothing: The state of the art, ES Gardner Jr - International Journal of Forecasting, 1985 – Elsevier

VITA

YUN LU

2006	B. E., Software Engineering Beihang University Beijing, China
2006-2009	Dean assistant, Software Engineering School of Software Beihang University Beijing, China
2009	M. S., Software Engineering Beihang University Beijing, China
2010-2013	Graduate Research Assistant Florida International University Miami, FL, USA
2013	Ph. D., Computer Science Florida International University Miami, FL, USA

PUBLICATIONS AND PRESENTATIONS

CONFERENCES

Yun Lu, Mingjin Zhang, Tao Li, Yudong Guang and Naphtali Rishe. (2013 Aug). *Online Spatial Data Analysis and Visualization System.* In ACM SIGKDD Conference on Knowledge Discovery and Data Mining Workshop on Interactive Data Exploration and Analytics on (pp.73-79), Chicago, IL. ACM.

Yun Lu, Mingjin Zhang, Tao Li, Chang Liu, Erik Edrosa, Naphtali Rishe. (2013 Oct). *TerraFly GeoCloud: Online Spatial Data Analysis System.* In ACM International Conference on Information and Knowledge Management demo paper on (pp. 2457-2461), San Francisco, CA. ACM.

Yun Lu, Ming Zhao, Guangqiang Zhao, Lixi Wang, Naphtali Rishe. (2013 Dec). *Massive GIS Database System with Autonomic Resource Management.* In International Conference on Machine Learning and Applications BigData Workshop, ACCEPTED, Miami, FL.

Yun Lu, Mingjin Zhang, Shonda Witherspoon, Yelena Yesha, Yaacov Yesha, Naphtali Rishe. (2013 Dec). sksOpen: Efficient Indexing, Querying, and Visualization of Geo-spatial Big Data. In International Conference on Machine Learning and Applications BigData Workshop short paper, ACCEPTED, Miami, FL.

Huibo Wang, Yun Lu, Yudong Guang, Erik Edrosa, Mingjin Zhang, Raul Camarca, Yelena Yesha, Tajana Lucic, Naphtali Rishe. *Epidemiological Data Analysis in TerraFly Geo-Spatial Cloud*. In International Conference on Machine Learning and Applications BigData Workshop, ACCEPTED, Miami, FL.

JOURNALS

Yun Lu, Ming Zhao, Lixi Wang, Naphtali Rishe. *v-TerraFly: Large Scale Distributed Spatial Data Visualization with Autonomic Resource Management.* In Journal Of Big Data, SUBMITTED.

Lixi Wang, Yun Lu, Jing Xu, Ming Zhao. *Cross-layer Optimization for Virtual Machine Resource Management.* In IEEE Transactions on Parallel and Distributed Systems. SUBMITTED.