2012

Web Mapping and Application Towards a Cloud: Enabling a WebGIS Prototype in an Open Source Environment

Ravi Dhungel
ravi.dhungel@siu.edu

This research paper addresses the cutting-edge technology of cloud computing in the domain of Geographic Information (GI) science. In recent years, the exponential proliferation of spatial data has prompted researchers to address the difficulties in managing Geographic Information System (GIS) applications in the cloud. The purpose of this paper is to propose a software web-mapping prototype in order to further the understanding of cloud computing framework to manage large spatial data. A review of the literature along with the development of the prototype exemplifies the technical architecture to best manage spatial data in the cloud. The web-mapping prototype has been developed in open source environment to simulate the n-tiered architecture of a cloud. This research finds web mapping a perfect fit for the cloud computing model and also underpins the importance of spatial cloud computing for solving the domains of physical science. This paper is a comprehensive analysis of cloud computing model in the geospatial landscape.

WEB MAPPING AND APPLICATION TOWARDS A CLOUD: ENABLING A WEBGIS PROTOTYPE IN AN OPEN SOURCE ENVIRONMENT

by

Ravi Dhungel

B.Eng., Kathmandu University, 2001

A Research Paper
Submitted in Partial Fulfillment of the Requirements for the
Master of Science Degree

Department of Geography and Environmental Resources
In the Graduate School
Southern Illinois University Carbondale
December, 2012

RESEARCH PAPER APPROVAL

WEB MAPPING AND APPLICATION TOWARDS A CLOUD: ENABLING A WEBGIS PROTOTYPE IN AN OPEN SOURCE ENVIRONMENT

by

Ravi Dhungel

A Research Paper Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Master of Science

in the field of Geography and Environmental Resources

Approved by:

Dr. Justin Schoof, Chair

Dr. Guangxing Wang

Dr. Mengxia Zhu

Mr. Samuel Adu-Prah

Graduate School
Southern Illinois University Carbondale
November 03, 2012

AN ABSTRACT OF THE RESEARCH PAPER OF

RAVI DHUNGEL, for the Master of Science Degree in GEOGRAPHY AND ENVIRONMENTAL RESOURCES, presented on October, 2012, at Southern Illinois University Carbondale.

TITLE: WEB MAPPING AND APPLICATION TOWARDS A CLOUD: ENABLING A WEBGIS PROTOTYPE IN AN OPEN SOURCE ENVIRONMENT

MAJOR PROFESSOR: Dr. Justin Schoof

This research paper addresses the cutting-edge technology of cloud computing in the domain of Geographic Information (GI) science. In recent years, the exponential proliferation of spatial data has prompted researchers to address the difficulties in managing Geographic Information System (GIS) applications in the cloud. The purpose of this paper is to propose a software web-mapping prototype in order to further the understanding of cloud computing framework to manage large spatial data. A review of the literature along with the development of the prototype exemplifies the technical architecture to best manage spatial data in the cloud. The web-mapping prototype has been developed in open source environment to simulate the n-tiered architecture of a cloud. This research finds web mapping a perfect fit for the cloud computing model and also underpins the importance of spatial cloud computing for solving the domains of physical science. This paper is a comprehensive analysis of cloud computing model in the geospatial landscape.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

LIST OF ACRONYMS

API	Application Programming Interface
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CSW	Catalogue Service for the Web
CTA	Chicago Transport Authority
DaaS	Data as a Service
ESRI	Environmental System Research Institute
FGDC	Federal Geographic Data Committee
GCI	Geographic Cyber Infrastructure
GDAL	Geospatial Data Abstraction Library
GI	Geographic Information
GIS	Geographic Information System
GOS	Geo-Spatial One-Stop
GPS	Global Positioning System
HP	Hewlett Packard
HPC	High Performance Computing
IaaS	Infrastructure as a Service
IBM	International Business Machine
INSPIRE	Infrastructure for Spatial Information in Europe
KAGIS	Kingston Automated Geoinformation Service
LBS	Location Based Service

MBR	Minimum Bounding Rectangle
NSDI	National Spatial Data Infrastructure
OGC	Open Geospatial Consortium
OPenDAT	Open Source Project for Network Data Access Control
PaaS	Platform as a Service
PDA	Personal Digital Assistant
QoS	Quality of Services
SaaS	Software as a Service
SAM	Spatial Access Methods
SOA	Service Oriented Architecture
SOC	Service Oriented Computing
SQL	Structured Query Language
SDI	Spatial Data Infrastructure
SWE	Sensor Web Enablement
TCP/IP	Transmission Control Protocol/ Internet Protocol
USDOL	United States Department of Labor
VM	Virtual Machines
WCS	Web Coverage Services
WFS	Web Feature Services
WMS	Web Mapping Services

1. INTRODUCTION

Background

Activities in space are naturally governed by space time dimensions. Geographic Information (GI) Science applications exist in an increasingly data-rich environment. Lee *et. al.* (2008) suggested that more than 80 to 90 percent of all information is geospatially related. The inherent complexities of spatial data bring GI Science to the unique frontier of communication, computation, and visualization. The unprecedented growths in sensor technologies, imaging systems, and computational power have resulted on dramatic increase and usage of spatial data in structured and unstructured formats. Location Based Services (LBS) seamlessly use Geographic Information Systems (GIS), Global Positioning Systems (GPS), and remote-sensing for real-time as well as offline applications. LBS like cell phones, navigational systems, remote sensors, and ubiquitous computing led spatial data inexorably towards the repositories of complex data structures, semantics, and ontologies.

The rapid development of GIS in the preceding decades has been an exciting phenomenon experienced by GI Science communities. The recent developments of GIS applications have ranged from the desktop to enterprise GIS architectures and further towards cloud architecture. While these applications have scaled GIS to broader communities, they have also created a new challenge for managing and processing large data sets. Ubiquitous computing has spawned the need for seamless access of LBS applications embedded with GIS applications and has contributed to the development of distributed GIS in a Cloud Computing environment of Service Oriented Architecture (SOA).

Geographic data sets are moving beyond the well-structured vector and raster formats to include semi-structured and unstructured data, especially geo-referenced stream data and

multimedia data (Han *et al*. 2002). Traditional spatial analysis methods were developed when data collection was expensive and computational power was weak. The high dimension of geographic data was easily overwhelmed by techniques that were designed to tease information from small, scientifically sampled, and homogeneous datasets (Harvey and Han, 2001). Spatial analysis of data became increasingly more complicated with the advent of high resolution satellite imagery, sensors, and GPS. Further, De Smith *et al*. (2007) explained these pertinent issues in modern GIS applications.

The impetus for managing spatial data infrastructure across various domains through collection, management, and utilization of geospatial data, traces back to 1884, when the national program for topographic mapping was created. Following the advent of the United States Federal Geographic Data Committee (FGDC) and the subsequent National Spatial Data Infrastructure (NSDI), along with the effort of the Open Geospatial Consortium (OGC) and Industry Organization for Standardization (ISO), geospatial data management has been in a state of continuous progress. The initiative of FGDC to develop the GeoCloud was a milestone in the deployment of geospatial cloud applications. Further, Yang *et al.* (2010) proposed a collaborative geospatial infrastructure that utilizes geospatial principles for research, development, and education. The authors also highlighted the importance of SOA to provide functional and intermediate services. Further, they emphasized on the semantic web and geospatial cloud computing to implement transparent and opaque platforms through the notion of a Geographic Cyber Infrastructure (GCI) cube shown in Figure 1.1.

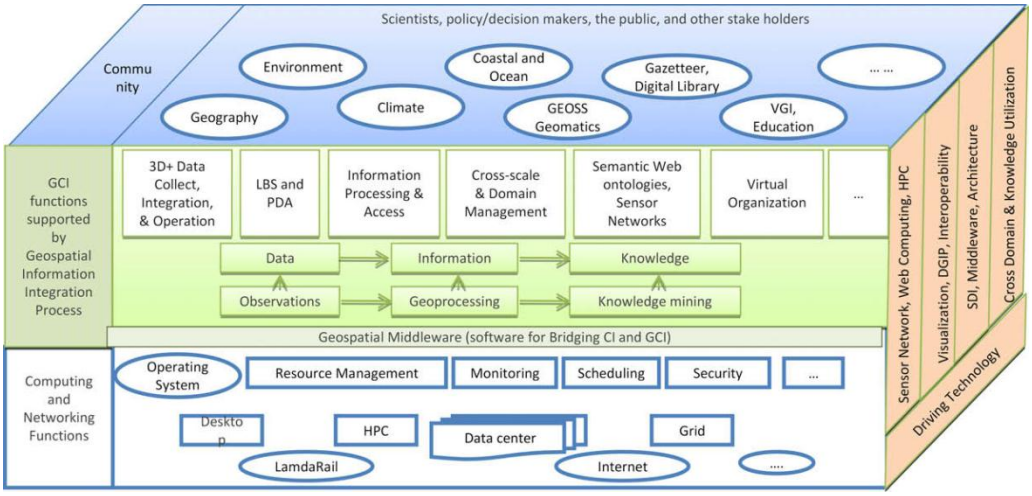


Figure 1.1 GCI cube (Yang *et. al* 2010). The GCI cube focuses on the geospatial middleware for bridging the interface between Cyber Infrastructure and Geospatial Cyber Infrastructures.

The GCI cube shows the complex multi-dimensional facet of geo-spatial data and its association among different scientific domains. Further, it also illustrate the complexities in terms of work flow management, underlying computing and networking technologies, and emphasizes on the geospatial middleware for bridging the Cyber Infrastructure and Geographic Cyber Infrastructure.

Mobile GIS applications have been widely used by service providers in SOA in a cloud computing environment. Service Oriented Computing (SOC) represents a new generation of distributed computing platform architecture often referred to as SOA. SOAs hide the technical details of the datasets in question by exposing them through standard implementation-neutral web interfaces, potentially making them available to wider audiences. Cloud provides heterogeneous services in various domains. Some of the domains include social networking Four Square, storage services like Drop Box, mapping services Google Earth, and mobile advertising services that are enabled in the vicinity of certain geographic location in LBS. The emergence of

LBS in SOA is critical for maintaining Quality of Services (QOS) that depends largely upon spatial queries, existing system architecture, communication infrastructure, and social behavior of users.

The use of SOA in GIS is becoming increasingly widespread. The architecture of LBS (Figure 1.2) shows the fusion of the heterogeneous technologies of communication, computation, navigation, and visualization, to support GIS applications in the cloud computing environment.

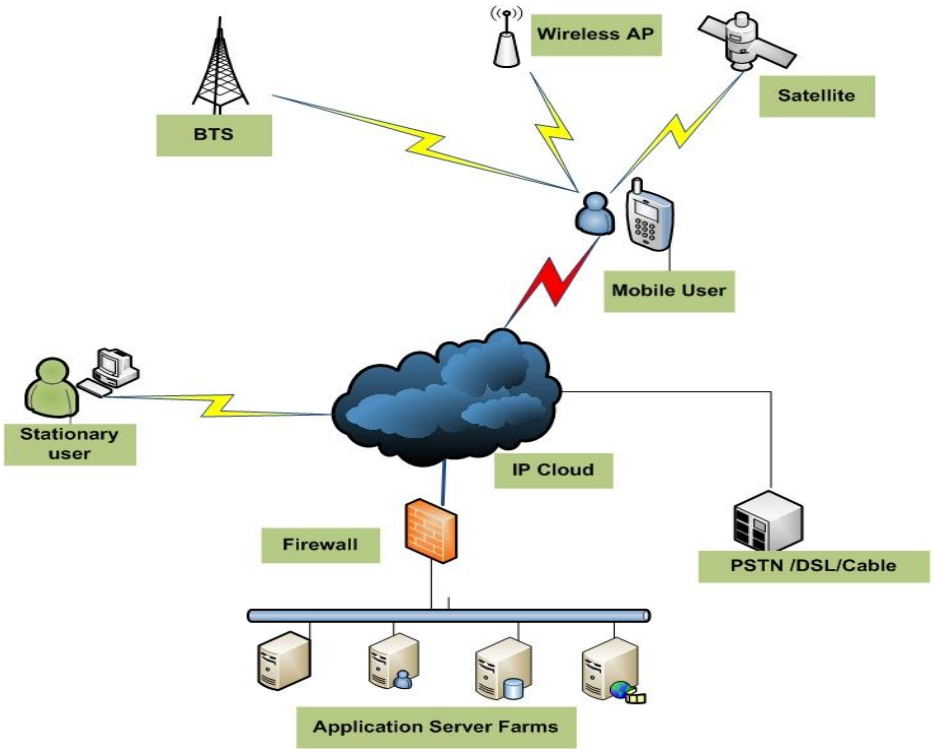


Figure 1.2. Architecture of LBS. The architecture shows the complex interaction between users and systems through heterogeneous computational and communication frameworks in LBS.

The cloud computing paradigm which enables the use of computers as a service rather than a product could be an important framework to manage and query spatial data. Goodchild *et al.*

(2011) described the importance and the motivation for geospatial data sharing in the wake of the initiation of Geo-Spatial One-Stop (GOS) (http://www.geo.data.gov), a recent US e-government cloud initiative.

This paper scrutinizes the various technical aspects of cloud computing for GI Science applications and how GI Science principles can be used to optimize the cloud computing environment. The paper also proposes a prototype to provide Web Mapping Services (WMS) in a cloud and illustrates how Web Mapping Applications fit in the cloud architecture.

Justification

GI Science provides an analytical framework based on spatio-temporal principles. The algorithms and models developed based on our understanding of datasets and earth phenomena are complex and are driven by spatio-temporal principles (Yang *et al*. 2011). Among the various challenges associated with GI services, the traditional GIS applications including web mapping have some shortcomings for managing large-scale spatial data, on-demand resources, customization, and delivery to larger communities. The management of real-time spatial data is even more challenging but appropriate concepts to integrate such geo-information are not yet sufficient (Craglia, 2009). To meet the need of mass market interdisciplinary applications, scalable solutions are necessary. Also, Spatial Data Infrastructure (SDI) initiatives explicitly require a guaranteed response time for some critical spatial queries (INSPIRE, 2007). This makes the quality of GI services in LBS critical for its operational requirement.

Although the technical advantages of LBS -using a cloud computing based infrastructure- are well-articulated and a variety of domain applications are available to confirm this, there is still little information on how these applications are designed, organized or implemented. This

research illustrates a technical framework for the design of large-scale GIS applications in a cloud computing environment in the open source framework by developing the prototype for Web Mapping Application

Purpose of the Study

The purpose of this study is to analyze the broader organizational and technical framework of cloud computing models for GIS applications emphasizing Web-Mapping Applications. Technical aspects are further synthesized and scrutinized to examine the architecture and implementation of large scale GIS applications in a cloud. Specific objectives of this study are: (i) to critically analyze the conceptual and technical framework for managing the large-scale GIS applications in a cloud computing framework; and (ii) To introduce a prototype to examine the architecture of WMS services in the cloud computing environment using open source software. Finally, this research summarizes the research challenges and prospect of GIS applications in the cloud computing environment. Two research questions were used to guide this research.

Research Questions

1) What is the most appropriate technical framework for handling large-scale GIS applications in a computing environment?
2) How can Web Mapping Services (WMS) be enabled using open source framework?

2. LITERATURE REVIEW

Architecture and Implementation of Location Based Services

Recently, Spatial Data Infrastructures (SDIs) have undergone a transition towards web based geospatial information systems (Kiehle *et al*. 2007, Schaffer *et al*. 2009). SDI consists of technical, organizational, and legal frameworks for enabling GI services (McLaughlin and Groot 2000). This section provides relevant conceptual framework for understanding the large scale GIS applications in a cloud computing environment.

Location based services (LBS) are computer applications that deliver information depending upon the location of the device and the user. With the advent of the Service Oriented Architecture (SOA) paradigm (Erl, 2005), GIS experienced a tectonic change from stand-alone applications to a distributed environment illustrated in SDI (Masser 2005). Raper *et al* (2007) studied the architecture of LBS and emphasized the diverse applications of GI services in the SOA. The Venn diagram in Figure 2.1 illustrates the major components of LBS as internet, mobile devices, and GIS databases. The associations and intersections among these technologies become the foundational framework for the LBS.

Figure 2.1. LBS as an intersection of technologies (Brimicombe, 2002). The LBS is an intersection between Internet, GIS/Spatial databases and mobile devices.

Distributed GI Services

Although distributed computing environments are not specifically designed for GI services, computers in distributed networks can communicate with each other seamlessly despite of heterogeneous transmission medium and applications. Many requirements of GI services that include high bandwidth and higher computational power for applications like geo-visualization, web mapping, analytics, and real-time geo-processing were not realized in the early development of distributed computing infrastructures. The development of GIS from the desktop to the enterprise and from enterprise to the cloud has largely been dependent on the existing communication infrastructure particularly the internet and Transmission Control Protocol /Internet Protocol (TCP/IP). Although the distributed computing system has been in existence for a long period of time, it is successful in assimilating the noble technologies that includes Web 2.0, web-mapping, GPS, and other GI services. The evolution of GIS from two layer architecture (client and server) to the multi-layered architecture is driven by the pertinent need of GIS applications. Peng and Tsou (2003) studied the various frameworks of GI services and criticized the continuum of heterogeneous communication protocols available for GI services even in the future given the absence of perfect distributed systems. Friis-Christensen *et al*. (2007) prototyped service architecture for distributed geo-processing and concluded that current Open Geospatial Specifications (OGC) provide a blueprint for implementing SOA in GI Science applications. The author however cautioned that further scientific collaboration is required to manage a large amount of spatial data.

The internet has become increasingly more instrumental in providing GI services in the distributed environment. Geoportals are leveraging the values of GIS applications through the World Wide Web (WWW). Distributed Web Service standards are available to support the SOA

through the Open Geospatial Consortium (OGC). OGC (2008) facilitates the interoperability and SOA through following OpenGIS standards.

- *Web Map Services (WMS):* WMS provides implementation specification for the creation and display of map-like views of information.
- *Web Feature Services (WFS):* WFS provides implementation specification and allows a client to retrieve and update geospatial data encoded in Geography Markup Language from multiple web feature services. The specification defines interfaces for data access and manipulation operations on geographic features. Via these interfaces, a web user or service can combine, use and manage geodata from different sources. A Transactional WFS includes the optional transaction operation to insert, update, or delete a feature.
- *Web Coverage Services (WCS):* WCS Implementation Specification allows clients to access part of a grid coverage offered by a server. The data served by a WCS is grid data usually encoded in a binary image format. The output includes coverage metadata.
- *The Catalogue Service for the Web (CSW):* The Catalog standard defines common interfaces to discover, browse, and query metadata about data, services, and other potential resources.

Additionally, OGC supports Sensors Web Enablement (SWE) for the standardization of sensor data. OGC has been the backbone in addressing the standardization and interoperability pertaining to the spatial Data and GI services. The granularity of the data structure has been well defined by OGC and has aided loose coupling among various GI services through its unified neutral interfaces. Figure 2.2 shows the recent model of the OGC (2011) that emphasizes on

multi-tiered architecture for the deployment of GI services. The figure emphasizes access tier, business process tier, and a client tier as main tiers of applications in GI services.

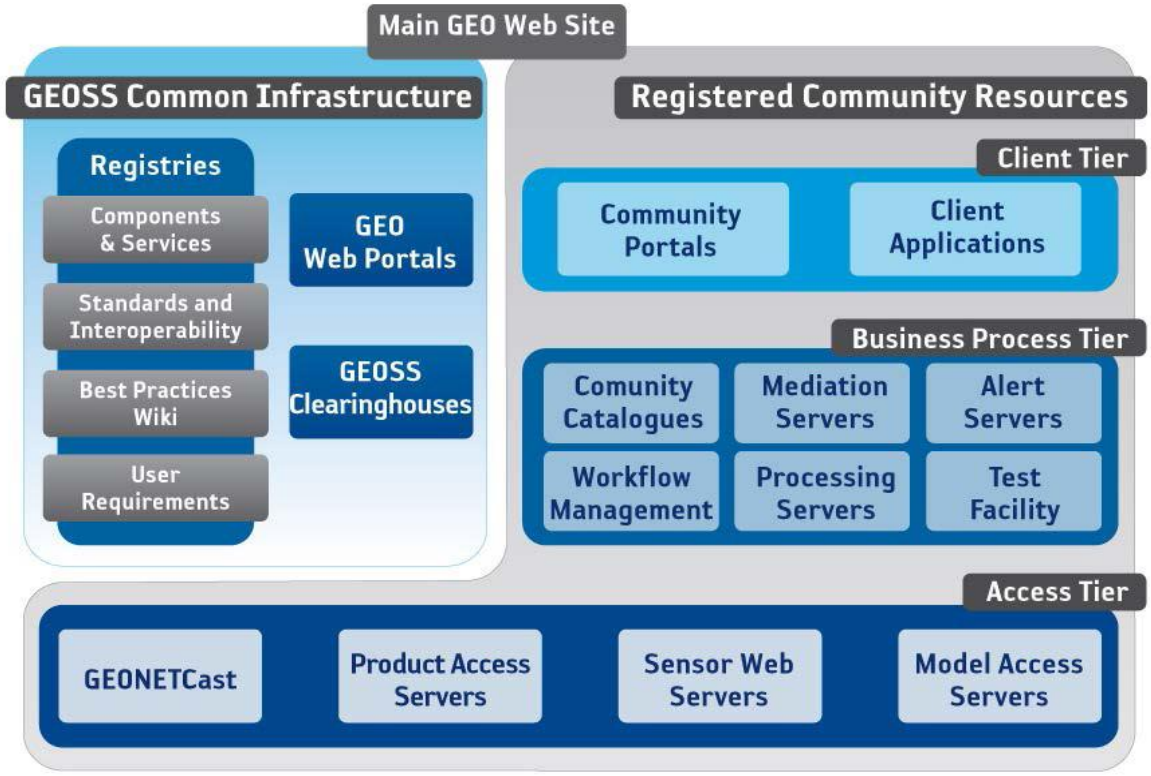


Figure 2.2. Tiered Architecture (OGC, 2011). OGC tiered architecture shows the interaction among modular tiers in the geospatial landscape.

Cloud Computing

Cloud computing refers to the recent advancement of distributed computing by providing 'computing as a service' for end users in a 'pay-as-you-go' mode; Cloud Computing defined by National Institute of Standards and Technology (2011) provides at least three types of services: Infrastructure as a Service (Iaas), Platform as a Service (PaaS), Software as a Service (SaaS). In addition, Olson (2009) proposed the use of Data as a Service (DaaS). Furthermore, Yang *et al.*

(2011) illustrated the importance of DaaS for GI Science applications. DaaS is the least defined but supports data discovery, access and utilization. DaaS also delivers data and data processing on demand to end users regardless of geographic or organizational location of provider and consumer. The cloud conceptual reference model of NIST (2012) in Figure 2.3 shows the high level architecture of cloud computing. Although this conceptual reference model provides the foundational basis for any cloud computing application; the framework lacks the issues pertinent to the GI science applications.

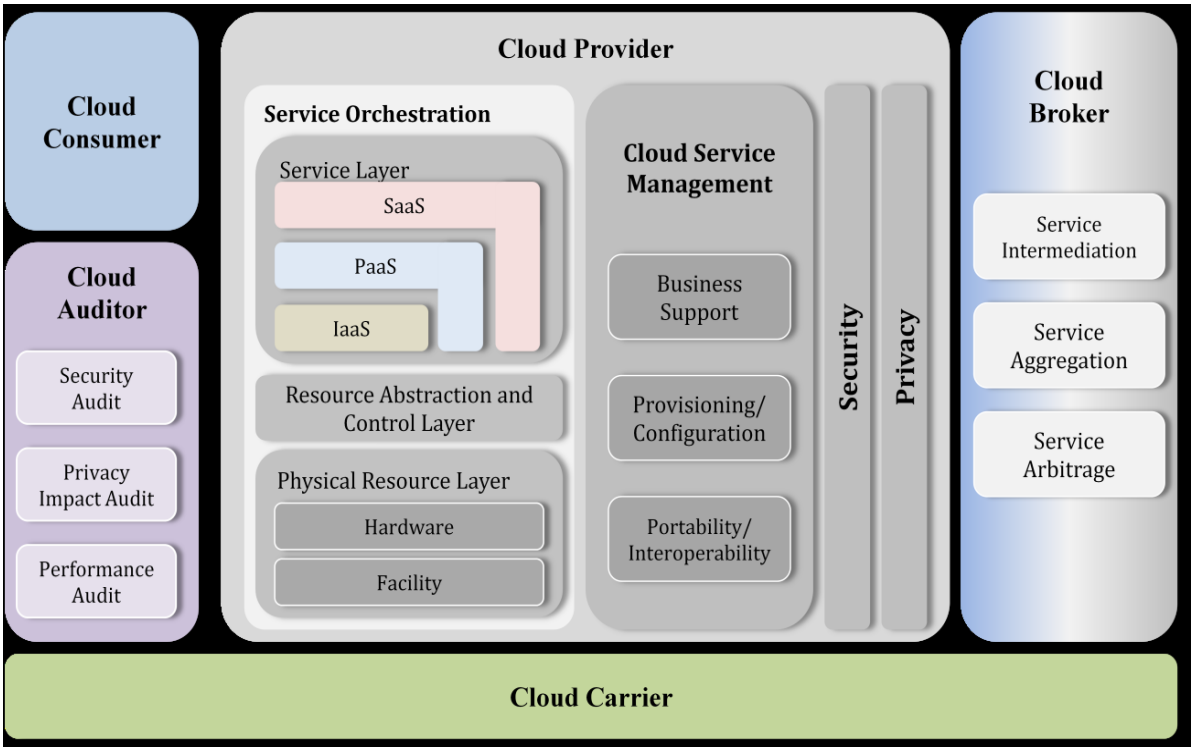


Figure 2.3. Cloud Architecture (NIST, 2012). The cloud architecture shows modular services layers, auditors, brokers and cloud service management in the service oriented architecture.

NIST (2011) defines characteristics for cloud computing as follows:

- *On Demand self-service:* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
- *Broad network access:* Capabilities are available over the network and accessed through standard mechanism that promotes use by heterogonous thick or thin client platforms (eg. Mobile phones, laptops, PDA etc).
- *Resource Pooling:* The provider's computing resources are pooled to serve multiple consumers using a multitenant model, with different physical and virtual resources dynamically assigned and re-assigned according to the consumer demand. There is a sense of location independence in that the consumer generally has no control or over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (ex. Country, state or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- *Rapid elasticity:* Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- *Measured Services:* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g. storage, processing, bandwidth, active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized services.

Cloud computing often depends on virtualization technology where many clients run on separate operating virtual machines (VMs), especially for providers of Infrastructure as a Service (IaaS). Blower (2010) implemented web mapping in SaaS model on Google Application Engine and shows that web-mapping applications can be scaled well in the cloud computing environment. Mola *et al*. (2011) studied about managing the cloud operation through the hierarchy of policy-based autonomic managers. Mohapata et al. (2012) implemented virtualization of ArcGIS 10 in Windows 2008 environment and underlined its importance for instructional purpose at Minnesota State University

Spatial Cloud Computing

Cloud computing and distributed GIS inherently exhibit spatio-temporal phenomena. There is only a handful of literature reviews that exists in the domain of spatial cloud computing. Most of the literature identifies spatial cloud computing as the new research paradigm for GI science and emphasizes broader interdisciplinary research and collaboration.

Yang *et al*. (2011) highlighted two key aspects of spatio-temporal principles in the cloud computing domain: (i) improving the cloud computing capability by leveraging the spatio-temporal phenomena and (ii) using cloud computing technology to address the issues of geospatial science.

The geographic perspective of cloud computing and spatial analysis of a cloud brings GI Science towards the unique frontier of computational science. The elastic nature of resources changes because of its invocation by different geographically dispersed users located in geographically dispersed computing environment brings the problem domain towards GI Science principles. Spatio-temporal principles can be applied to increase the performance of the cloud

computing environment. The understanding of spatio-temporal characteristics will help to shape the cloud computing architecture and the services it provides. The quality of services of LBS is inherently dependent on spatio-temporal locations of data centers, query locations, queried objects and end users locations. Yang *et al.* (2011) also highlighted on environmental impacts of cloud computing on enabling green IT by decreasing the carbon foot print and optimizing computing resources. Schaffer *et al.* (2010) studied the various technical and economic factors for integration of SDI and cloud computing. The authors also illustrated the importance of standardization and the role of OGC for interoperability of GI services in the cloud environment. Cary et al. (2010) studied that leveraging the architecture of spatial cloud computing environment, spatial index partitioning and data partitioning strategies in the parallel computing framework within Hadoop MapReduce clusters that significantly improve performance.

There has been a growing interest in the geospatial industries to enable the spatial cloud computing. Recent technical white papers published by industry leaders such as Amazon and Intergraph identify the importance of cloud computing model for geospatial industry. Further Amazon's white paper titled "Mapping and Geospatial Analysis in Amazon Web Services using ArcGIS "illustrate how large spatial imageries data could be accessed and stored in cloud. The paper outlines caching an image service in different scale drastically to improve the system's performance.

Implementation Models

The deployment models of a cloud can be broadly divided into public, private and hybrid clouds. Public clouds are publicly available, mutli-tenant pools of computing resources that build on economies of scale paradigm. Private clouds are computing resources operated, owned, and

controlled by the organization that uses them internally inside the enterprise. These clouds are more secure and private. GI services can be implemented in any of these models according to the organizational goal and availability of computing resources. Exploring the GI services into various cloud models (private, public, and hybrid) also helps to unravel the potentiality of cloud computing for geospatial landscape.

The GI services in the cloud in reference of licenses can be broadly divided into two main categories namely proprietary and open source models.

Open source software is usually developed as a public collaboration and often freely available. Open source refers to software system whose source code is available for use or modification by third-party developers; the term was adopted by a group of people on January 1998 at free software movement in California.

Open Source uses a UNIX-like operating system environment and its various clones to implement distributed GI services. The open source applications are available as cloud software, system software, desktop software, applications software, etc. Among them that are widely used in industries and academic settings are Hadoop, GRASS, MapServer, Geoserver, Terralib, Quantum GIS, PostGIS, etc. Cloud computing frameworks like Hadoop provide a platform for managing very large scale data often called big data. Other open source cloud framework suites include Eucalyptus, Nimbus, and Open Nebula. Django-python has been used widely for PaaS, specifically for web application development in open source for GIS applications. Rodriguez-Martinez *et al.* (2010) evaluated the various open source frameworks in the terms of capability matrix criteria for managing the weather data and also implemented Open 911 for emergency management in Ubuntu Enterprise Cloud.Various Open Source projects are continuing in progress at the national and regional level to address the need of large data sets in science and

engineering applications. One of them is through the Open Source Project for Network Data Access Protocol (OPenDAT) that provides access to the oceanographic data stored in remote locations.

Commercial GI services also exist in the cloud, enterprise and desktop environment that include ESRI's Arc products, ERDAS Imagine, Apollo Server, ArcGIS Server, AutoCAD etc. The cloud environment has also been offered by IBM, HP, Amazon, Microsoft, VMware, etc. More recently, the GI services have also been offered by various industries in cloud like Amazon, Google, and GIS cloud Inc. There are various industries white papers that describe the architecture of these systems, and it's beyond the scope of this paper to illustrate the architectural aspect of these commercial products. Victoria (2010) emphasized the importance of the cloud computing environment in the future for ESRI's product suite.

Spatial Queries

Conceptually, geographic data have been divided into vector and raster types. Both represent different ways to encode and generalize geographic objects and phenomena. The spatial queries use a lot of vector and raster data that are periodically generated through sensors, satellites, and GPS devices. The large size of spatial repositories and complexities of geospatial models require upper time complexity in computation.

The properties of spatial data make the spatial queries explicitly complicated. Pre-processing the data, analyzing the spatial data evolution, pattern identification and modeling are a critical component that makes spatial queries very complex. Domain specific knowledge is inevitable to identify patterns of interest (Han *et al.*2002). To address these complex issues, there has been a progress in parallel data mining to handle huge volumes of spatial data in distributed

farms of clusters. Also, there has been substantial work in progress for spatial data mining query languages and geographic visualization. Furthermore, different techniques have been employed to improve the efficiency and representation of raster and vector queries. Spatial queries are both Central Processing Unit (CPU) and I/O (Input Output) intensive. Spatial databases have been explicitly studied and different Spatial Access Methods (SAM) has been proposed in the past. Hierarchical data structures are predominantly used for representing spatial data structure. R-tree proposed by A. Aguttman (1984) is spatial data structure that uses the concept of Minimum Bounding Rectangle (MBR) and has been widely used in different commercial and open source spatial database management systems.

Brinkhoff *et al.* (1993) emphasized the complexity associated with spatial operations for query evaluation of large volumes of complex objects that can't be naturally sorted in one dimensional arrays. Parallel Processing can be used to improve the performance of spatial queries that are non-linear. Papadopoulos *et al.* (1996) used parallel programming to implement Nearest Neighbor Finding (NNF), and P-NNF algorithm improves the efficiency by 60%. Further, Papadopoulos *et al*. (2003) also justified that index construction time can be reduced considerably by exploiting parallelism. Koperski *et al.* (1996) studied spatial indexing techniques to improve the performance of spatial query and access methods. Carey *et al.* (2010) used MapReduce parallel programming model and suggested that cloud computing paradigm is a suitable framework for spatial queries.

Zhang *et al*. (2003) studied spatial queries for mobile computing environments and argued that query results within a validity region can improve the query performance by decreasing the transmission overhead in the same region. Wang *et al.* (2010) used the MapReduce model and Hadoop platform to analyze spatial data storage, spatial index, and

spatial operations into the domain of GI Science. Wang *et al*. (2011) proposed the CloudMW to design hybrid cluster storage architecture in the Amazon Cloud Computing environment. However, Walker (2008) argued that High Performance Clusters (HPC) is better than cloud computing for scientific research. Chiu *et al.* (2010) experimented with the impact of cloud caching in service oriented computation and implemented the cooperative caching framework for storing the service‘s output data in-memory for facilitating fast accesses. Karemi *et al.* (2011) used GeoModel an open source tool to provide basic indexing and querying of geospatial data and emphasized inherent characteristics of spatial cloud computing.

However, there has been little progress in designing and implementing middleware for performing spatial analysis. The client server architecture during the initial development phase focused on implementation in the server-side. However, the client side could also yield overall performance enhancements of these queries (Vatsavai *et al*., 2006).

Security and Interoperability

LBS have a deep root beyond the technical scope of its implementation as a service. The implementations bring some of the social externalities as LBS by virtue depend upon the location data either for an operation or a legislative requirement. The potential of privacy invasion by state and a service provider can’t be overlooked. Hence, the applications of LBS that are used to monitor behavior of humans or resources and enhance participation could lead us to an era of surveillance. The juxtaposition between quality of services in LBS and individual’s privacy is simply a tradeoff of security and performance. Onsrud *et al*. (1994) and Raper *et al*. (2007) summarized the broader aspects and challenges of legal, ethical, and social issues and become an exciting social dimension for research due to technological development. The accuracy and

precision of spatial information for law enforcement agencies, transportation, and emergency management are crucial for their successful operations. The technical dimension of security and interoperability is again a key issue in distributed GI services like the cloud computing environment. This could be an exciting research opportunity that the GI science community should address in the future.

3. METHODOLOGY

Geographic Information Science (GI Sc.) exists in the plethora of geographic data that are seamlessly generated through the sensors, satellite-imageries, and GPS. The tools and framework needed to analyze these large spatial data sets are inexorably complex. Besides analyzing the architectural design issues, the second objective of this study was to design a prototype to enable Web Mapping Services (WMS) in a cloud using open source infrastructure. There are also many commercial COTS (Commercial Off-The-Shelf) products available to provide Web Mapping Services in the web. However, the open source paradigm, which is a pragmatic methodology for free distribution of software codes and design, has been chosen to enable WMS in a cloud.

As the open source software provides source code, installation, configuration, and management of a prototype can be optimized for the underlying hardware's resources. It is trivial that each of the configuration files of all modules can be fully optimized and customized to our need for better performance of the system.

Tools

The prototype was implemented using various open source modules. Table 3.1 illustrates the details of each of the tools that were used to develop the prototype. As the prototype is completely developed in the open source environment, most of these tools were downloaded from the internet and installed from a source codes. The documentation that comes with most of the open source software was used to install and customize the modules.

Table 3.1. Assemblage of tools used for the development of a prototype

Name	Functions
Ubuntu 11	System Software
Apache 2.0	Web Server
PostgreSQL 9.1	Database Management System
PostGIS 2.0	Spatial database Tier
MapServer 6.02	Web Mapping Engine
Python/PHP mapscript	Application Programming Interface
pgAdminIII	Graphical User Interface for postgreSQL
Kompozer	HTML editor
ArcDesktop10	Geographic Information System
GDAL/OGRINFO	Geographic Data Abstraction Library
Shp2sql	Tool to convert shape file to sql schema
Shp2img	Tool to convert shape file to image file

Data management and processing

The data sets were downloaded as shapefiles from the city of Chicago website, (http://www.cityofchicago.org/city/en/depts/doit/provdrs/gis.html). Table 3.2 lists the dataset, their feature types, number of records, number of attributes and their size. All feature-types were used to enable the web mapping application in the prototype. The boundary data was provided in polygon feature type that has six different attributes with only one record set. The roads data were provided in a lines feature type with 50 different attributes and 55747 different record sets. In addition, Chicago Transport Authority (CTA) bus routes data was used to enable the line feature type and CTA bus stops data was used to enable the point feature type. Different sizes of the databases were used that varies with the number of attributes, record-sets and feature-types. All geometric feature-types were used in the prototype.

Table 3.2 List of Data sets used in the Prototype

Layer Name	Type	Number of Rows	Number of Attributes	Size (Kb)
Boundary	Polygon	1	6	8192
Roads	Lines	55747	50	2600
CTA_Routes	Lines	147	6	520
CTA_Bus Stops	Points	11782	17	1952

Figure 3.1 illustrates the data management and pre-processing processes for developing the prototype environment. The shapefiles were converted to the physical SQL schema using the shp2sql tool. The cloud database was created in PostgreSQL server and all the schemas were imported in a cloud database. All the database schemas that were created for the prototype are listed in Appendix A. The PostGIS, which is a middle tier of database helps to store the geographic data type and was enabled on the top of PostgreSQL. The geometry columns created by postGIS allows to store the geographic data in a PostgreSQL. The SQL schema was imported into the cloud database of the PostgreSQL using psql command. PgAdmin III, the Graphical User Interface was used to manage the PostgreSQL.

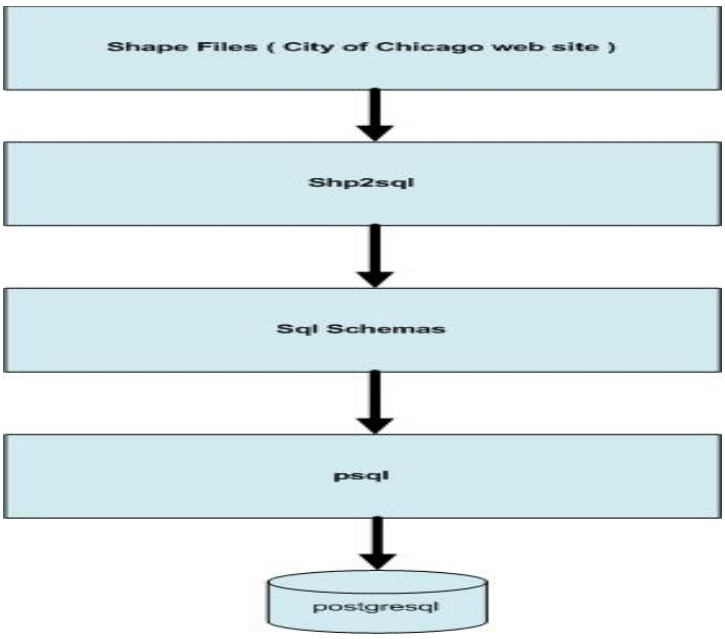


Figure 3.1. Process used to manage the spatial data in the prototype. The shape file was downloaded from the city of Chicago web site. Shp2sql tool was used to create an intermediary SQL schemas, the schemas were finally imported to the postgreSQL database management system.

Further, GiST (Generalized Search Trees) indexes were created for each of the generated schemas from Shp2sql for creating the indexes of spatial data. The GiST index creation processes of each of the schemas are listed in Appendix C.

Architecture, Design and Implementation of Prototype

The prototype was implemented in n-tiered client server architecture in a Local Area Network in the Advanced GIS lab, Department of Geography, Southern Illinois University Carbondale. The prototype was configured on multiprocessor machine in Dell Precision 690 that had Intel® Xeon™CPU 3.20 GHz and 4.00 GB RAM. The architecture of the system is illustrated in Figure 3.2

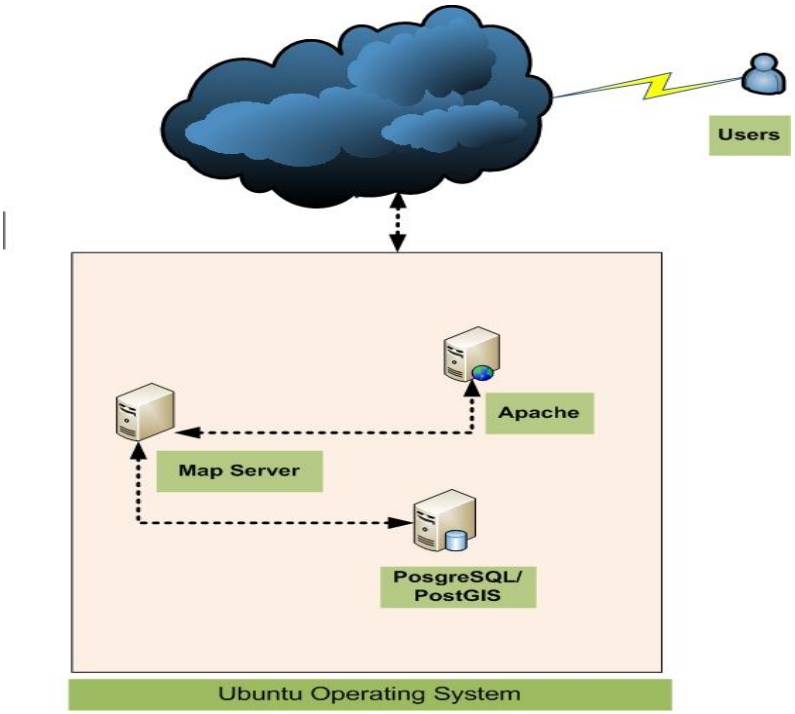


Figure 3.2. Design architecture of the prototype showing the key open source modules. The dotted line shows the interaction among the key components to publish a map in a cloud.

The web-mapping prototype was developed in the open source framework using Ubuntu as a system operating software. The python was used as a programming language for a development framework. PostgreSQL was used as the Object-Relational database management system to store the spatial data. PostGIS, the spatial component of the prototype was enabled on top of PostgreSQL to support the geographic data types and spatial functions. Map Server, the image rendering engine acts as a middle tier web-mapping component to publish the map in a cloud through an Apache web server. Apache web server also served as a middle-ware to host web mapping application in the cloud. All of these modules were downloaded from the internet with their source code and were compiled using gcc compiler to optimize with underlying hardware architecture and operating system. It ensured that the web mapping prototype was optimized for underlying hardware resources.

The prototype can be accessed by users through the Local Area Network (LAN) of the Southern Illinois University through browsers such as Internet Explorer, Mozilla, etc. However, the Internet Protocol (IP) address of the server should be provided to locate the prototype in LAN as the server is not mapped to the Domain Name System (DNS). For authorization purposes, the username and password should be verified to the PostgreSQL to query and publish the map. The user requests a map in the cloud through a browser provided to the user. The requested map would be generated on real-time from the data stored in PostgreSQL server through the Apache server and is rendered by a map server. The map server will read the map files that are specific to the requested individual map and it has all the information regarding the layout of maps and the

queries to the PostgreSQL.The different map files that are read by map server were created in advance. Those map files have granular information specified through a map class written in a structured semantic. Map files constitute the detail technical specification of maps such as color, pixels size, extent, cartographic elements etc. Five different map files were created for the prototype. The lists of map files created and used in this prototype are listed in Appendix B. The python codes served as a backend programming environment and uses Application Programming Interface (API) among these modules in the prototype. Appendix C lists all the python codes written to publish a map in the browser in a cloud.

Graphical User Interface (GUI)

GUI is an interface between the user and the database management system to query the maps. The GUI was developed in python, php and HTML. GUI can be accessed through network for processing the spatial queries to generate map files. The map requests are sent through the GUI on the LAN that uses TCP/IP (Transmission Control Protocol/Internet Protocol). Results are sent to the server through the LAN using TCP/IP protocol. The server processes the request and displays the map in GUI through HTML page. For authentication purpose, the HTML forms are used as controls to check the username and passwords stored in PostgreSQL database.

4. RESULTS AND DISCUSSION

The dream of utility computing has pushed all the stakeholders in a noble frontier of technological revolution. The omnipresent of spatial data and ubiquitous computing led to tremendous increase in a usage of services like social networking, Location Based Services, and navigation systems and vice versa. Implementing Web Mapping Services (WMS) in a cloud infrastructure in an open source framework provides the challenges along with opportunities to leverage the value of service oriented architecture for GIS applications.

The prototype was developed using various loosely coupled modules. Each of the modules and components that were used are developed by different open source communities with their different design goals and objectives. However, by using all the modules in the prototype environment has illustrated the abstraction of encapsulated service of complex web mapping services. The prototype can be analyzed as a Software as a Service and Software as a Platform paradigm in a cloud computing model. The design and architecture of the prototype was limited to specified map productions but the prototype is highly scalable to the larger production environment provided ample computing and communication bandwidth. Figure 4.1 compares the prototype with the cloud layered n-tiered architecture (NIST, 2012) and illustrates how web mapping can be enabled on the Service Oriented Architecture in the cloud using the open source environment. The prototype can be examined further on level of services it provides to the higher and lower layers of its SOA and also to the users of the prototype. The higher layer of the prototype is an access layer that provides GUI to query the map and publish it in the user's browser. The users can only published the specified map in this layer. The services it provided can be analogous to Software as a Service meaning that only specific maps can be published by the prototype. Apache is an interface between Map server and a GUI. Map servers are a middle

tier that renders the map in a real time by querying the spatial data stored in the PostgreSQL/PostGIS. The users can use the map server to query the database and publish a map using a specific map file. Map files can be created by the users themselves if the users can logon to the system remotely using remote clients with administrative privileges. Also, the user can directly create the spatial database in the postgreSQL/PostGIS where all the spatial datasets are stored in object-relational model. The postgreSQL/PostGIS provides SQL statements and spatial functions to query the database. This is the lower level of the services provided by the proto-type where we can even specify the feature-types and data-types of the schema.

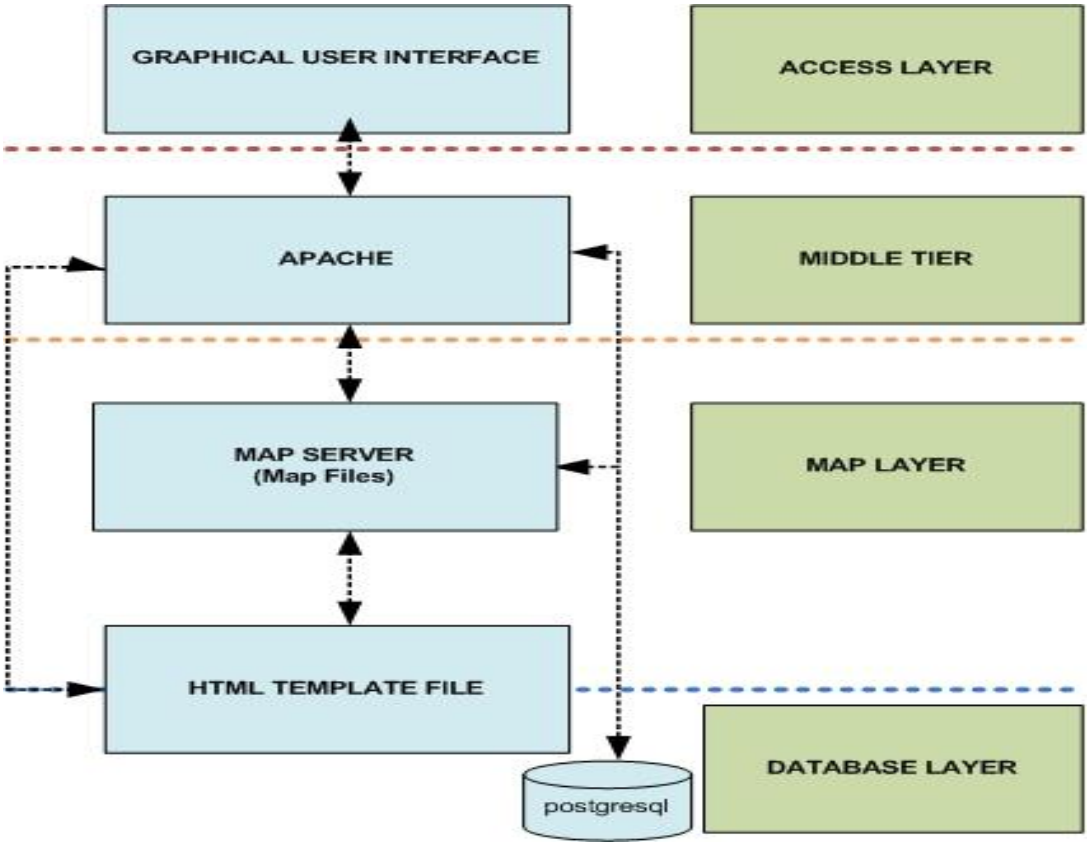


Figure 4.1. Prototype in layered architecture. The dotted line shows the different hierarchy of the different modules. The arrow line shows the interaction among key modules of the prototype.

Xiaogiang *et al.* (2010) explored the available distributed computing frameworks in a cloud for GI services and emphasized the cloud computing framework for the GI services. The designed prototype also illustrated how web mapping applications can be enabled in a cloud computing environment in Platform as a Service model and can take advantage of layered SOA architecture of the cloud. Schaffer *et. al* (2012) implements an open source framework for building geo-processing workflows using real time geo-data based on distributed web services and suggested the use of wrapper for geo-processing functionality to use the large functionalities of the popular stand-alone open source GRASS GIS. The prototype designed also used middle-tiers (Apache web server and Map server) illustrated on Figure 4.1 open to support the web mapping application in a cloud computing environment. Yang *et. al* (2010) suggested the importance of the open-source software and middle ware for the visualization and distribution of spatial data and urges for more research to integrate, synchronize, distribute and balance the processing within the distributed environment. Further, the prototype was fully developed using open source environment. Sieber *et. al* (2011) used the spatial cyber infrastructure in an open source framework for humanities that includes conflicting data, heterogeneous data models, disambiguation and a geographic scale. The literature clearly illustrated that the cloud computing tiered architecture represents the best computing model for the GI science applications including web-mapping applications. Further, the prototype is a Platform as Service models of a cloud reinforce the importance of modular components for scalability in a cloud environment. The prototype by virtue of being open source is interoperable than Blower (2010) web mapping in Google Application Engine (GAE) which is implemented as a SaaS rather than PaaS. SaaS limits the prototype to Application Programming Interface (API) with underlying proprietary file system of Google. However, the scalability, provisioning and granularity of GAE easily outpace

the developed prototype. The files generated in prototype could be easily ported from the prototype to another compatible system and that is interoperable among multiple systems which is not possible in Google Application Engine environment.

The prototype can be further scrutinized to the n-tier architecture of a cloud computing environment proposed by NIST (2012). The access layer can be associated as GUI, which is analogous to access layer of the layered architecture of the cloud. The Apache and Map Server serves as a middle-tier layer in the architecture with map server being the key engine for web-mapping, which acts as an interface between the Apache web server and PostgreSQL server. The map server and database server are a scalable component meaning that they can handle large sets of spatial data and render data into the web to generate the requested maps. The SOA of the prototype can simply be exposed as each of the components are loosely coupled and provides the specific services to the other components and to users. The map server generates the map file. The apache server publishes it in the web. The PostgreSQL/PostGIS is an Object-Relational database management system that stores the spatial data. Furthermore, the prototype gives a notion of an abstraction where the underlying complexity is hidden to provide the web-mapping services.

Figure 4.2 shows one of the samples of map generated in the browser through LAN by a prototype. The map generated as the road map of the city of Chicago road networks in a line features-type. Due to the large size of the records in roads database, there is a some delay to render the map in prototype system. However, the delay can be improved by increasing the physical hardware infrastructure of the prototype.

The detail technical specifications of the generated maps in the prototype depend upon the map files read by the map server. The map file consists of information of the database server,

metadata of map, and other mapping components. All the map files used are written similar like Xtra Markup Language Type (XML) and are listed in Appendix B. The map files helps to customize maps generated by the prototype.

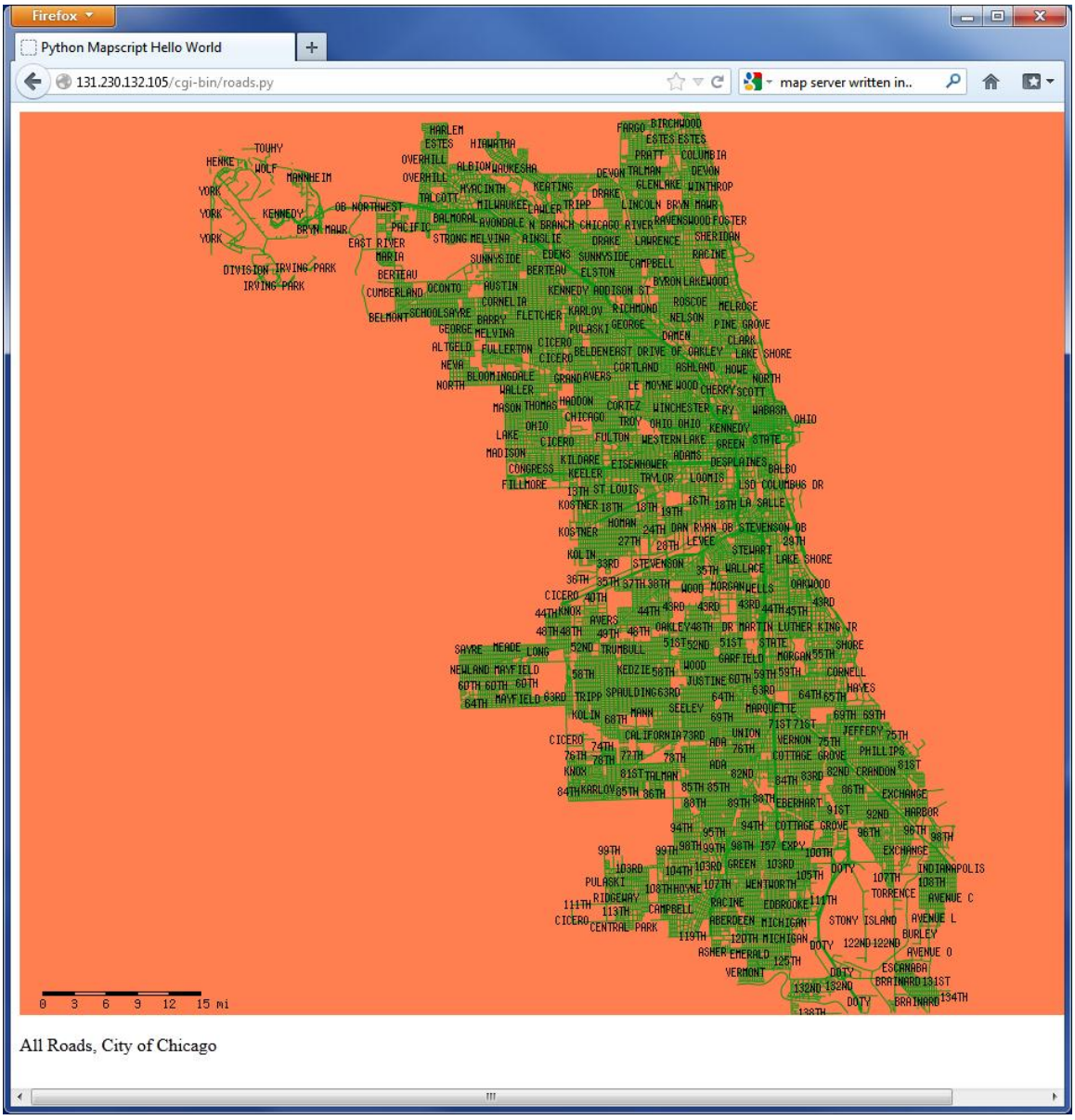


Figure 4.2 Roads, City of Chicago. The user's browser showing the snapshot of prototype that queries all roads from a spatial database.

Figure 4.3 shows the City of Chicago sample map that has polygon feature-type generated by the prototype environment in a browser accessed from LAN of the Southern Illinois University, Carbondale. Due to the small size of spatial database with polygon feature type, the map was published instantly.

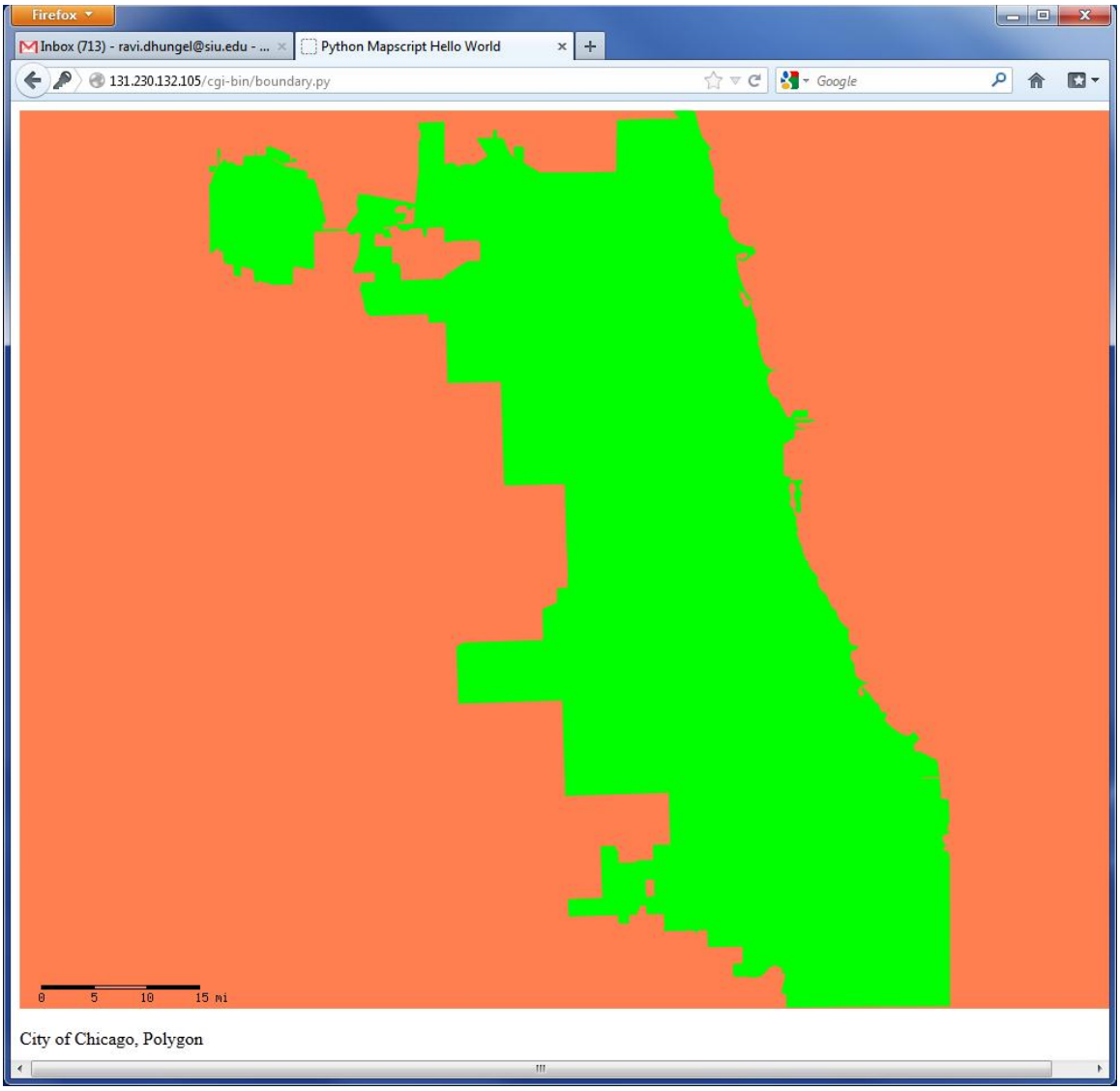
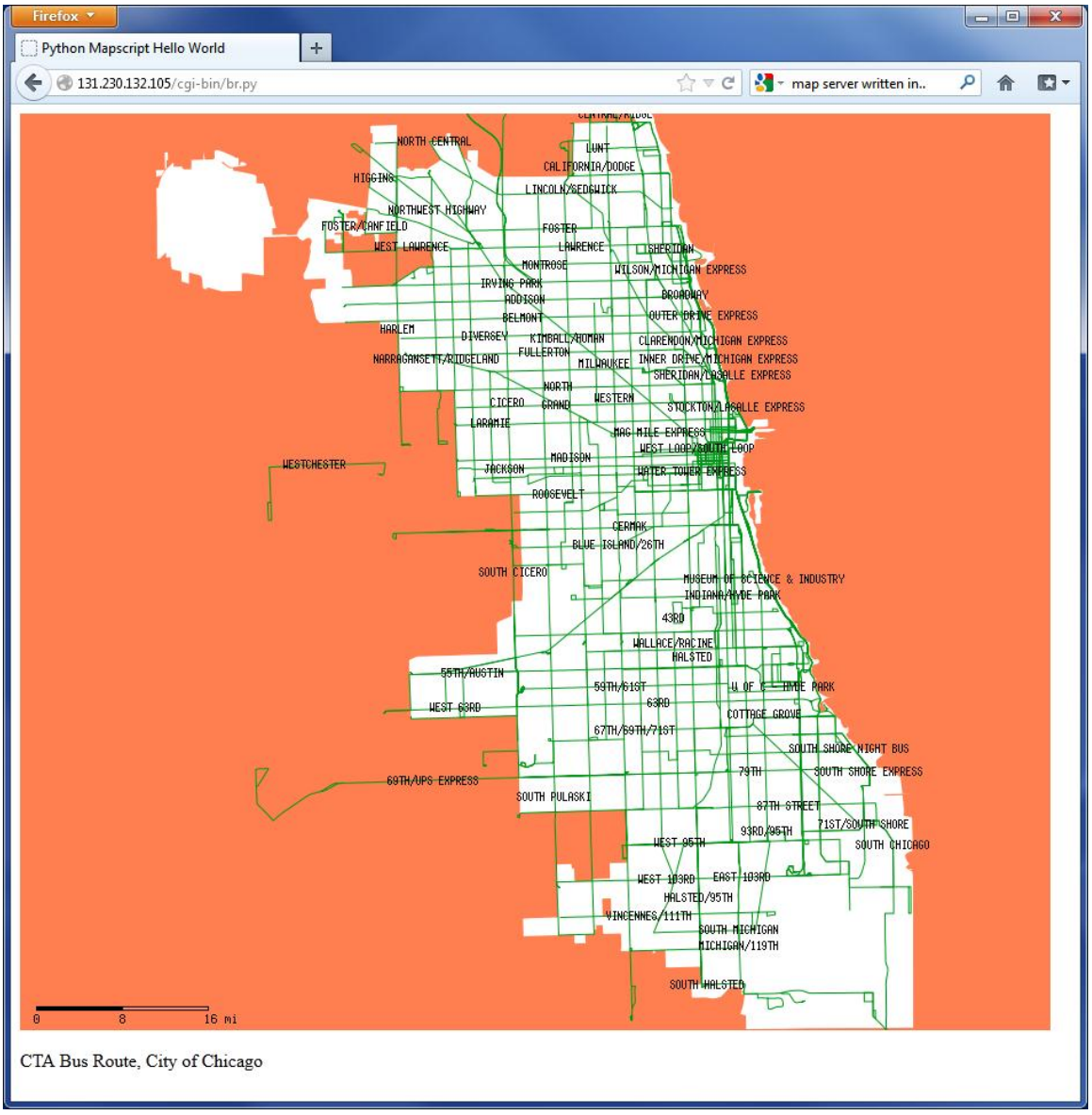


Figure 4.3. Area, City of Chicago. The user's browser showing the snapshot of prototype that queries polygon features from a spatial database.

Figure 4.4 shows the Chicago Transportation Authority (CTA) bus routes map with labels generated by the prototype. The map constitute of two different layers. The CTA bus routes and boundary of Chicago. The single map file can generate multiple layers of map.



CTA Bus Route, City of Chicago

Figure 4.4 CTA bus Routes. The user's browser showing the snapshot of prototype that queries CTA bus routes from a spatial database.

Likewise Figure 4.5 maps all the CTA bus stops in the city of Chicago. Since there are more than eleven thousands CTA bus-stops, the maps points look like a line feature type.

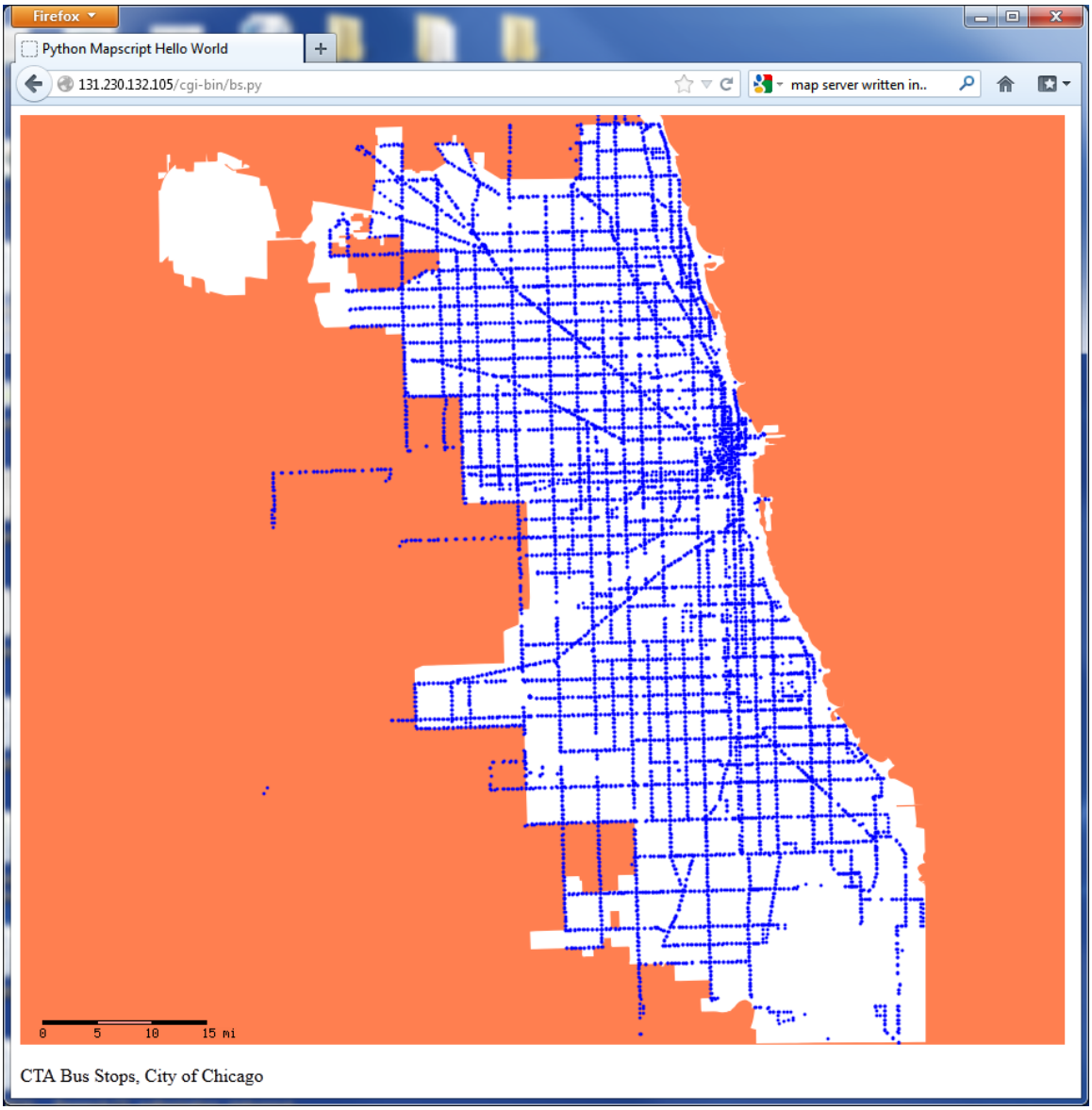


Figure 4.5 CTA bus stops. The user's browser showing the snapshot of prototype that queries CTA bus stops from a spatial database.

Figure 4.6 illustrated the sample map generated as a single map that includes Chicago Area, Roads, CTA bus routes, and CTA bus Stops by using a single map file. Although, the map rendering time increases proportionally with the number of rows and number of layers, the prototype published the multi-layer map in less than 2 seconds in a server. The prototype will slow down if there is the bandwidth limitation when we are querying a large spatial database like roads. It clearly illustrates that web mapping applications needs more bandwidth to transfer a map in GUI of the client browser. The bandwidth can also be scaled as the Infrastructure as a Service in a network access layer in a cloud computing model but this is the limitation of the prototype.

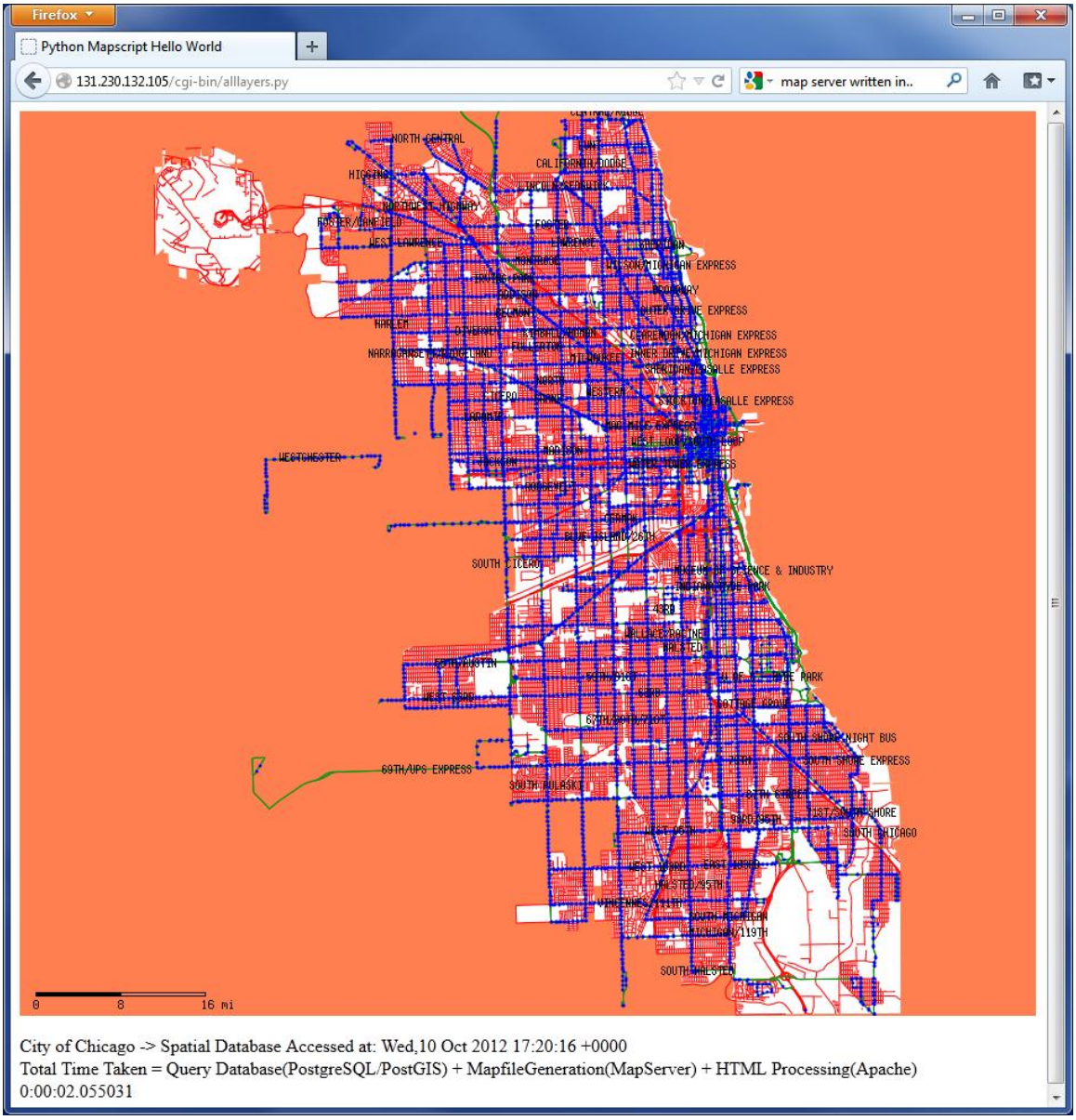


Figure 4.6. All layers, City of Chicago. The user's browser showing the snapshot of prototype that queries all feature types from a spatial database.

Nevertheless, the prototype system designed gives the notion of implementing web mapping applications in a cloud computing environment and supports the scalability of the system in terms of users, services and architecture. The prototype is limited to web mapping applications in a cloud computing environment. Also, due to the limitation of the hardware

infrastructure and bandwidth, the scalability and performance of the system was analyzed only on the basis of the available data-sets and only in the local area network of Southern Illinois University, Carbondale. In addition, the spatial analysis and spatial operations are CPU intensive beside file I/O operations required in non-spatial databases. The prototype development platforms are well tested in commercial operational environment. The prototype is limited to understand the architecture of web mapping applications in a cloud, thus, emphasized on the modularity of the open-source environment for scalability and optimization.

As open source modules are loosely coupled, the SOA of the cloud can be well supported not only for Web Mapping Applications (WMS) but also for LBS and even for geo-processing applications like in KAGIS. HPC in a cloud could be other options to analyze the large data sets in a cloud computing environment. Hadoop Map/Reduce framework could also be another distributed model implemented in a cloud that can be used to process large spatial dataset. On contrary, it's trivial that managing the open source framework requires high level skills and knowledge. The operational cost of the open source framework might be a financial setback instead of using metered proprietary services.

The recent development of Kingston Automated Geoinformation Service (KAGIS) in 2012 illustrates how geo-processing in a cloud was successfully enabled in a national level in United Kingdom as a fully integrated service system. Further, Shi and Walford (2012) outlined the importance of SOA and the effort of OGC for automated geo-processing mechanism for the success of KAGIS in cloud computing environment.

On enabling spatial cloud computing, most of the literature reviews only calls for the broader collaboration among GI science communities that showed spatial cloud computing is still in infancy. Yang *et. al* (2011) conducted empirical studies to determine how spatial

computing can facilitate the advancement of physical science and optimize distributed computing environment for simulation, high power computing, and data service search, access, and utilization. The GCI framework proposed by Yang *et. al* (2010) can be considered as a blueprint for spatial cloud computing. However, there has been little progress beside a lot of awareness with limited collaboration and a few success stories on the research spectrum of spatial cloud computing.

Moreover, the pertinent problem of interoperability and security in a cloud would always be pertinent in the GI applications too. Nevertheless, the OGC framework has further pushed stakeholders for uniformity of spatial data and towards the cloud computing framework. Further, the transportation of large volume of spatial data sets from users to the providers could also be a key critical issue that has been largely neglected on most of the literature. Beside technical limitation of cloud computing itself, the security and privacy in social dimension should be critically analyzed in the future as location data are highly sensitive.

5. CONCLUSION AND IMPLICATIONS

The omnipresence of spatial data generated by ubiquitous computing and recent advances in remote sensing and GIS applications demand reliable and elastic GI Services. The prototype designed clearly illustrated that WMS services can be enabled on the cloud using open source framework in the cloud computing environment. The prototype emphasized on neutral middle-tiers that are critical for interoperability in SOA. The cloud becomes a perfect computing framework to implement large scale web-mapping applications due to its SOA. Although the commercial Web-Mapping Applications (Bing, Google map) support the SaaS architecture of the cloud computing and provide Application Programming Interface (API) to access their services, It's the open source environment that gives more control and granularity to the users to optimize their Web-Mapping Applications. The user's specification in written in a map file illustrates that fact. Further, it also supports the elasticity of Web-Mapping Application in a brink of Platform as a Service model in a cloud.

"Thinking and computing spatio-temporally is needed to enable cloud computing to better manage, arrange, select, and schedule pooled computing resources to best satisfy the biggest number of end users, and better optimize the discoverability, accessibility, utilizability, and computability of science and application data, information, knowledge and phenomena (Yang *et al*. 2011)".

Implementing the large scale GI services besides web mapping requires high level of management and collaboration in multiple tiers of a cloud computing framework. Additionally, GI Science applications are inevitably interdisciplinary, so addressing the collaboration and work flow in GI Services and enabling spatial cloud computing is exciting research dimension. Hence,

further interdisciplinary collaboration among researchers is required in the future to implement the spatial cloud computing framework.

While the effectiveness of spatial data processing depends on many factors such as workload patterns, data volumes, portioning, and I/O performance between storage and processing units, the massive elastic scalability and parallel processing foundation of the public cloud offers real promise for large-scale geospatial processing. Due to the requirement of the large volume of spatial data and their inherent association, cloud computing by virtue of its characteristics provides a promising framework for handling large scale GIS applications In addition, the geographic perspective of cloud computing is emphasized.

As the USDOL (2010) recognized geospatial industry as one of the fastest growing industries, the growth of geo-spatial data is inevitable. With the inception of cloud in geospatial landscape, further regulatory constraints will ease as cloud computing grows in acceptance. Managing these large data often termed as “big data” needs further scientific enquiry, collaboration, and research. With only a handful of cloud GI service providers in the market in geospatial landscape, developing GI science applications beside web mapping services in a cloud is still in its infancy and needs further scientific attention for the advancement of GI science.

REFERENCES

Amazon Web Services 2012. Mapping and Geospatial Analysis in Amazon Web Services using ArcGIS. http://media.amazonwebservices.com/AWS_ESRI_Mapping_GeoSpatial_Analysis_Using_ArcGIS.pdf

Blower, J.D. 2010. GIS in the cloud: implementing a Web Map Service on Google App Engine. In Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research \&\#38; Application, ACM, New York, NY, USA, , Article 34 , 4 pages. DOI=10.1145/1823854.1823893 http://doi.acm.org/10.1145/1823854.182389

Brimicombe, A. J. 2002. GIS: Where are the frontiers now? Proceedings GIS: 33-45

Brinkho,T., H.P. Kriegel, and B. Seeger. 1993. Efficient processing of spatial joins using r-trees. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data,pages 237

Cary, A., Y. Yaacov, A. Malek, and R. Naphtali. Leveraging cloud computing in Geodatabase Management. In: Proceedings of the 2010 IEEE International Conference on Granular Computing, 1416 August 2010, San Jose, CA, USA, IEEE, 7378, 14-16.

Chiu,D., A. Shetty, and G. Agrawal. Elastic Cloud Caches for Accelerating Service-Oriented Computations." In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10). IEEE Computer Society, Washington, DC, USA, 1-11. DOI=10.1109/SC.2010.21 http://dx.doi.org/10.1109/SC.2010.21

Craglia M. 2009. From INSPIRE to Digital Earth: Research Challenges for the Next

Generation Spatial Data Infrastructures. In: Proceedings of the 12th AGILE Conference, Hannover, Germany.

De Smith, M.J., P.A. Longley, and M.F. Goodchild. 2007. Geospatial Analysis: A comprehensive guide to principles, techniques and software tools. Winchelsea Press

Erl T. 2005. Service-oriented architecture: concepts, technology, and design.Prentice Hall PTR.

Friis-Christensen, A., N. Ostländer, M. Lutz, and L. Bernard. 2007. Designing service

architectures for distributed geoprocessing: Challenges and future directions. *Transactions in GIS* 11, (6): 799–818.

Goodchild,M. F., P. Fu, and P. Rich. 2007. Sharing Geographic Information: An assessment of the Geospatial One-Stop. Annals of the Association of American Geographers 97 (2): 250-266

Han, J., Russ Altman, V. Kumar, H. Mannila, and D. Pregibon. 2002. Emerging Scientific Applications in Data Mining. Communications of the ACM 45 (8).

INSPIRE.2007. INSPIRE Network Services Performance Guidelines, INSPIRE Consolidation Team, European Commission.

Intergraph 2011. Cloud Computing Raising Geospatial Technology to the Cloud: Intergraph Strategy for Leveraging Cloud-based Resources. http://www.intergraph.com/assets/plugins/sgicollaterals/downloads/CloudComputing_WhitePaper.pdf

Karimi, H. A., D. Roongpiboonsopit, and H Wang. Exploring Real-Time Geoprocessing in Cloud Computing: Navigation Services Case Study. *Transactions in GIS* 15: 613–633. doi: 10.1111/j.1467-9671.2011.01263.x

Kiehle, C., K. Greve, and C Heier. 2007. Requirements for Next Generation Spatial Data Infrastructures-Standardized Web Based Geoprocessing and Web Service Orchestration. In *Transactions in GIS* 11(6): 819–834, doi:10.1111/j.1467-9671.2007.01076.x

Koperski,K. and J. Han. 1996. Data Mining Methods for the Analysis of Large Geographic Databases. In: Proc. Of Conference on Geographic Information Systems, Canada.

Lee, C. and G Percivall. 2010. Standards-Based Computing Capabilities for Distributed Geospatial Applications. Computer .41(11): 50-57. doi: 10.1109/MC.2008.468 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4668683&isnumber=4668666

Liu, F., J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf. 2011. Recommendations of National Institute of Standards and Technology. NIST Cloud Computing Reference Architecture, special publication 500-292, 0-35.

Masser I. 2005. GIS worlds: creating spatial data infrastructures, 1st edn. ESRI Press, Redlands

McLaughlin J. and R. Groot. 2000. Geospatial data infrastructure: concepts, cases and good practice. In: Spatial Information Systems and Geostatistics Series University Press, Oxford)

Harvey, M. J. and J. Han. 2009. Geographic Data Mining and Knowledge

Discovery. Chapman &Hall.

Mohapatra, R., D. A. Friend, and B. H. Hoffman. 2012. ArcGIS Desktop Virtualization: Anywhere Anytime GIS at Minnesota State University, Mankato. Cloud Computing For Geographers, AAG Conference.

Mola, O. and M.A. Bauer. 2011. Towards Cloud Management by Autonomic Manager Collaborations. *Int. J. Communications, Network and System Sciences*, 4: 790-802.

NIST. 2012."A NIST Definition of Cloud Computing" http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

OGC.2011. OGC Reference Model. https://portal.opengeospatial.org/files/?artifact_id=47245

Olson A. John. 2009. Data as a Service: Are We in the Clouds? Journal of Map & Geography Libraries 6 (1): 76-78

Onsrud, H. J., J. P. Johnson, and X. Lopez. 1994. Protecting Personal Privacy in Using Geographic information System. *Photogrammetric Engineering and Remote Sensing* LX (9): 1083-1095

Papadopoulos, A., and Y. Manolopous. 1996. Parallel Processing of Nearest Neighbor Queries in Declustered Spatial Data. In: GIS '96 Proceeding 4th ACM international workshop on Advances in Geographic Information Systems.

Peng, Z and M.H. Tsou.2003. Internet GIS : Distributed Geographic Information Services for the internet and wireless networks. John Wiley and Sons: 720

Raper, J., G. Gartner, H. Karimi, and C. Rizos. 2007. A critical evaluation of location based services and their potential. *Journal of Location Based Services* 1 (1): 5-45

Rodriguez-Martinez, M., J. Seguel, and M. Greer. 2010. Open Source Cloud Computing Tools: A Case Study with a Weather Application. 2012 IEEE Fifth International Conference on Cloud Computing, 443-449.

Schaffer, B., B. Baranski, and T. Foerster. 2010. Towards Spatial data infrastructures in the Clouds. In: Painho,M.,Santos,M.Y.,Pundt,H.(Eds.),GeospatialThinking. Springer-Verlag, Berlin: 399-418.

Shi, S. and N. Walford 2012. Automated Geoprocessing Mechanism, Processes and Workflow for Seamless Online Integration of Geodata Services and Creating Geoprocessing Services. *IEEE Journal of Applied Earth Observations and Remote Sensing*, (99): 1-6. doi: 10.1109/JSTARS.2012.2187433 URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6170903&isnumber=46

09444

Vatsavai, R.R., S. Shekhar, T.E. Burk, and S. Lime. 2006. UMN-MapServer: A High-Performance, Interoperable, and Open Source Web Mapping and Geo-spatial Analysis System. *In GIScience* : 400-417.

Wang, J, Peter Varman, and C. Xie. 2011. Optimizing storage performance in public cloud platform. *Journal of Zhejiang University-Science* (Comput & Electron) 12 (12): 951-964

Wang,Y., S. Wang. 2010. Research and Implementation on Spatial Data Storage and Operation Based on Hadoop Platform. Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference 2: 275-278, 28-31 Aug. 2010, doi: 10.1109/IITA-GRS.2010.5603956,
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5603956&isnumber=5601975

Walker, E. 2008. Benchmarking Amazon EC2 for high-performance scientific computing. USENIX log Mag, 33 (5): 18-23

Xiaoqiang, Y. and Deng Yuejin. 2010. Exploration of Cloud Computing Technologies for Geographic Information Services. Geoinformatics, 18th International Conference: 1-5, 18-20 June 2010 doi: 10.1109/GEOINFORMATICS.2010.5567515
URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5567515&isnumber=5567473

Yang, C., M.F. Goodchild, Q. Huang, D. Nebert, R. Raskin, Y. Xu, M. Bambacus, and D. Fay. 2011. Spatial cloud Computing: How can the Geospatial Sciences Use and help Shape Cloud Computing? *International Journal of Digital Earth* 4 (11): 305-329

Yang, C., R. Raskin, M.F. Goodchild, and M. Gahegan. 2010.Geospatial Cyberinfrastructure: Past, Present and Future. Computers, Environment, and Urban Systems 34 (4): 264-277.

Yang, C., H. Wu, Q. Huang, Z. Li, and J. Li. 2011. Using spatial principles to optimize distributed computing for enabling the physical science discoveries. Proc. Nat'l Academy of Sciences (PNAS)108 (14): 5498-5503.

Zhang,J., M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. 2003. Location-based spatial queries. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD '03). ACM, New York, NY, USA, 443-454. DOI=10.1145/872757.872812 http://doi.acm.org/10.1145/872757.872812

USDOL- United States Department of Labor, Last updates 07/08/2010 http://www.dol.gov/opa/media/press/eta/eta20100950.htm

Victoria, K. GIS In the cloud. ESRI, 2010

APPENDICES

APPENDIX A
Database Schemas

```
CREATE  TABLE roads
        (
         gid serial NOT NULL,
         fnode_id numeric(10,0),
         tnode_id numeric(10,0),
         trans_id numeric(10,0),
         pre_dir character varying(1),
         street_nam character varying(50),
         street_typ character varying(5),
         suf_dir character varying(5),
         streetname numeric(10,0),
         l_f_add numeric(10,0),
         l_t_add numeric(10,0),
         r_f_add numeric(10,0),
         r_t_add numeric(10,0),
         logiclf numeric(10,0),
         logiclt numeric(10,0),
         logicrf numeric(10,0),
         logicrt numeric(10,0),
         class character varying(4),
         status character varying(4),
         status_dat date,
         tiered character varying(1),
         oneway_dir character varying(1),
         dir_travel character varying(1),
         ewns numeric(10,0),
         l_parity character varying(1),
         r_parity character varying(1),
         f_zlev numeric(10,0),
         t_zlev numeric(10,0),
         l_fips numeric(10,0),
         r_fips numeric(10,0),
         r_zip character varying(5),
         l_zip character varying(5),
         r_censusbl character varying(15),
         l_censusbl character varying(15),
         f_cross character varying(75),
         f_cross_st numeric(10,0),
         t_cross character varying(75),
         t_cross_st numeric(10,0),
         length numeric,
```

```
 edit_date numeric(10,0),
 edit_type character varying(20),
 flag_strin character varying(10),
 ewns_dir character varying(1),
 ewns_coord numeric(10,0),
 create_use character varying(10),
 create_tim date,
 update_use character varying(10),
 update_tim date,
 shape_len numeric,
 geom geometry(MultiLineString,4326),
 CONSTRAINT roads_pkey PRIMARY KEY (gid )
)
WITH (
 OIDS=FALSE
);
ALTER TABLE roads
 OWNER TO postgres;

-- Index: gistindexroads

-- DROP INDEX gistindexroads;

CREATE INDEX gistindexroads
 ON roads
 USING gist
 (geom );

-- Table: cta_routes

-- DROP TABLE cta_routes;

CREATE TABLE cta_routes
(
 gid serial NOT NULL,
 objectid numeric(10,0),
 route character varying(4),
 name character varying(32),
 shape_len numeric,
 geom geometry(MultiLineString,4326),
 CONSTRAINT cta_routes_pkey PRIMARY KEY (gid )
)
WITH (
 OIDS=FALSE
);
ALTER TABLE cta_routes
```

```
 OWNER TO postgres;


CREATE TABLE ctabustop
(
 gid serial NOT NULL,
 systemstop numeric,
 street character varying(75),
 cross_st character varying(75),
 dir character varying(3),
 pos character varying(4),
 routesstpg character varying(75),
 owlroutes character varying(20),
 city character varying(20),
 status numeric(10,0),
 point_x numeric,
 point_y numeric,
 public_nam character varying(75),
 buspad character varying(10),
 ada character varying(10),
 sign_int integer,
 geom geometry(Point,4326),
 CONSTRAINT ctabustop_pkey PRIMARY KEY (gid )
)
WITH (
 OIDS=FALSE
);
ALTER TABLE ctabustop
 OWNER TO postgres;
```

Boundary :

```
-- Table: boundary

-- DROP TABLE boundary;

CREATE TABLE boundary
(
 gid serial NOT NULL,
 objectid numeric(10,0),
 name character varying(25),
 shape_area numeric,
 shape_len numeric,
 geom geometry(MultiPolygon,4326),
 CONSTRAINT boundary_pkey PRIMARY KEY (gid )
)
```

```
WITH (
 OIDS=FALSE
);
ALTER TABLE boundary
 OWNER TO postgres;
-- Table: login

-- DROP TABLE login;

CREATE TABLE login
(
 username text,
 password text
)
WITH (
 OIDS=FALSE
);
ALTER TABLE login
 OWNER TO postgres;
(
gid serial NOT NULL,
objectid numeric(10,0),
name character varying(25),
shape_area numeric,
shape_len numeric,
geom geometry(MultiPolygon,4326),
CONSTRAINT boundary_pkey PRIMARY KEY (gid )
)
WITH (
OIDS=FALSE
);
ALTER TABLE boundary
OWNER TO postgres;
CREATE TABLE login
(
```

```
username text,
password text
)
WITH (
OIDS=FALSE
);
```

APPENDIX B
Map Files

```
MAP
   IMAGETYPE PNG
     EXTENT 1091138.264200  1813891.893000 1205138.634000 1951652.763600
   SIZE 900 800
   UNITS METERS
   IMAGECOLOR 255 127 80
  CONFIG "MS_ERRORFILE" "/var/log/ms/ms_error.txt"
   DEBUG 5
   WEB
      TEMPLATE "/var/www/poi1.html"
      IMAGEPATH "/var/www/map/"
      IMAGEURL "/map/"
   END
   SYMBOL
        NAME 'circle'
        TYPE ELLIPSE
        FILLED TRUE
        POINTS
              1 1
        END
   END
   LAYER
           NAME "boundary"
           STATUS ON
           TYPE POlYGON
           CONNECTIONTYPE POSTGIS
           CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
           DATA 'geom FROM (select * FROM "boundary") as myquery using unique gid using
srid=4326'
            CLASS
                 STYLE
                 SIZE 40
                 COLOR 255 255 255
                 END
           END
     END
   LAYER
      NAME "bustops"
      STATUS ON
      TYPE POINT
```

```
      CONNECTIONTYPE POSTGIS
      CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
       PROCESSING "CLOSE_CONNECTION=DEFER"
      DATA 'geom FROM (select * FROM "ctabustop") as myquery using unique gid using
srid=4326'
       CLASS
          SYMBOL 'circle'
          SIZE 3
          COLOR 0 0 255
          NAME "pi"
          LABEL
               COLOR 0 0 0
               FONT arial
               TYPE TRUETYPE
               POSITION CC
               SIZE 7
               BUFFER 1
               OUTLINECOLOR 255 255 255
          END
        END
  END
  SCALEBAR
     STATUS EMBED
     INTERVALS 3
     TRANSPARENT TRUE
     BACKGROUNDCOLOR 250 150 140
     OUTLINECOLOR 0 0 0
  END
END
MAP
  IMAGETYPE PNG
       EXTENT 1091138.264200  1813891.893000 1205138.634000 1951652.763600
  SIZE 900 800
  UNITS METERS
  IMAGECOLOR 255 127 80
  CONFIG "MS_ERRORFILE" "/var/log/ms/ms_error.txt"
  DEBUG 5
  WEB
     TEMPLATE "/var/www/poi1.html"
     IMAGEPATH "/var/www/map/"
     IMAGEURL "/map/"
  END
  SYMBOL
       NAME 'circle'
       TYPE ELLIPSE
```

```
        FILLED TRUE
        POINTS
            1 1
        END
    END
    LAYER
            NAME "boundary"
            STATUS ON
            TYPE POlYGON
            CONNECTIONTYPE POSTGIS
            CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
            DATA 'geom FROM (select * FROM "boundary") as myquery using unique gid using
srid=4326'
             CLASS
                  STYLE
                  SIZE 40
                  COLOR 255 255 255
                  END
            END
       END
LAYER
            NAME "cta_roads"
            STATUS ON
            TYPE LINE
            CONNECTIONTYPE POSTGIS
            CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
            PROCESSING "CLOSE_CONNECTION=DEFER"
            DATA 'geom FROM (select * FROM "cta_routes") as myquery using unique gid using
srid=4326'
            LABELITEM "name"
       CLASS
                  STYLE
                       SIZE 20
                       COLOR 0 153 22
                  END
                 LABEL
              COLOR 0 0 0
              SIZE SMALL
                END
        END
       END
    SCALEBAR
       STATUS EMBED
       INTERVALS 2
```

```
    TRANSPARENT TRUE
    OUTLINECOLOR 0 0 0
  END
END
MAP
  IMAGETYPE PNG
  EXTENT 1091138.264200  1813891.893000 1205138.634000 1951652.763600
  SIZE 1000 800
  UNITS METERS
  IMAGECOLOR 255 127 80
  CONFIG "MS_ERRORFILE" "/var/log/ms/ms_error.txt"
  DEBUG 5
  WEB
    TEMPLATE "/var/www/poi1.html"
    IMAGEPATH "/var/www/map/"
    IMAGEURL "/map/"
  END
  REFERENCE
    IMAGE "/var/www/map/2.png"
    EXTENT 1091138.264200  1813891.893000 1205138.634000 1951652.763600
    STATUS ON
    COLOR -1 -1 -1
    OUTLINECOLOR 0 0 0
    SIZE 96 96
  END
  LAYER
    NAME relief
    DATA "/var/www/map/2.tif"
    STATUS DEFAULT
    TYPE RASTER
  END
  LAYER
    NAME "roads"
    STATUS ON
    TYPE LINE
    CONNECTIONTYPE POSTGIS
    CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
     DATA 'geom FROM (select * FROM "roads") as myquery using unique gid using
srid=4326'
  LABELITEM "street_nam"
        CLASS
            STYLE
                SIZE 20
                COLOR 0 153 22
            END
```

```
            LABEL
                COLOR 0 0 0
                SIZE SMALL
            END
        END
END
  SCALEBAR
    STATUS EMBED
    INTERVALS 5
    TRANSPARENT TRUE
    OUTLINECOLOR 0 0 0
  END
END
MAP
IMAGETYPE PNG
EXTENT 1091130 1813891 1205198 1951669
SIZE 1000 800
IMAGECOLOR 128 128 128
CONFIG "MS_ERRORFILE" "/var/log/ms/ms_error.txt"
DEBUG 5
WEB
  TEMPLATE "/var/www/poi1.html"
  IMAGEPATH "/var/www/map/"
  IMAGEURL "/map/"
END
LAYER
NAME "boundary"
STATUS ON
TYPE POlYGON
CONNECTIONTYPE POSTGIS
CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres password=dhungel"
 DATA 'geom FROM (select * FROM "boundary") as myquery using unique gid using
srid=4326'
  CLASS
     STYLE
   SIZE 20
   COLOR  0 255 0
     END
  END
LABELITEM "shape_area"
        CLASS
            STYLE
                SIZE 20
                COLOR 0 153 22
            END
            LABEL
```

```
                COLOR 0 0 0
                SIZE SMALL
            END
        END
    END
     SCALEBAR
    STATUS EMBED
    UNITS MILEs
    INTERVALS 5
    TRANSPARENT TRUE
    OUTLINECOLOR 0 0 0
  END
END
MAP
  IMAGETYPE PNG
   EXTENT 1091138.264200  1813891.893000 1205138.634000 1951652.763600
  SIZE 900 800
  UNITS METERS
  IMAGECOLOR 255 127 80
  CONFIG "MS_ERRORFILE" "/var/log/ms/ms_error.txt"
  DEBUG 5
  WEB
    TEMPLATE "/var/www/poi1.html"
    IMAGEPATH "/var/www/map/"
    IMAGEURL "/map/"
  END
  SYMBOL
      NAME 'circle'
      TYPE ELLIPSE
      FILLED TRUE
      POINTS
          1 1
      END
  END
  LAYER
        NAME "boundary"
        STATUS ON
        TYPE POlYGON
        CONNECTIONTYPE POSTGIS
        CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
        DATA 'geom FROM (select * FROM "boundary") as myquery using unique gid using
srid=4326'
         CLASS
              STYLE
              SIZE 40
```

```
            COLOR 255 255 255
            END
        END
    END
  LAYER
    NAME "roads"
    STATUS ON
    TYPE LINE
    CONNECTIONTYPE POSTGIS
    CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
    PROCESSING "CLOSE_CONNECTION=DEFER"
    DATA 'geom FROM (select * FROM "roads") as myquery using unique gid using
srid=4326'
     CLASS
        STYLE
         SIZE 20
         COLOR 255 0 0
           END
      END
  END
LAYER
        NAME "cta_roads"
        STATUS ON
        TYPE LINE
        CONNECTIONTYPE POSTGIS
        CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
        PROCESSING "CLOSE_CONNECTION=DEFER"
        DATA 'geom FROM (select * FROM "cta_routes") as myquery using unique gid using
srid=4326'
         CLASS
            STYLE
                SIZE 20
               COLOR 0 153 22
            END
        END
    END
 LAYER
        NAME "cta_roads"
        STATUS ON
        TYPE LINE
        CONNECTIONTYPE POSTGIS
        CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
        PROCESSING "CLOSE_CONNECTION=DEFER"
```

```
        DATA 'geom FROM (select * FROM "cta_routes") as myquery using unique gid using
srid=4326'
        LABELITEM "name"
        CLASS
            STYLE
                SIZE 20
                COLOR 0 153 22
            END
            LABEL
                COLOR 0 0 0
                SIZE SMALL
            END
         END
      END
   LAYER
      NAME "bustops"
      STATUS ON
      TYPE POINT
      CONNECTIONTYPE POSTGIS
      CONNECTION "host=127.0.0.1 port=5432 dbname=cloud user=postgres
password=dhungel"
       PROCESSING "CLOSE_CONNECTION=DEFER"
      DATA 'geom FROM (select * FROM "ctabustop") as myquery using unique gid using
srid=4326'
       CLASS
          SYMBOL 'circle'
          SIZE 3
          COLOR 0 0 255
          NAME "pi"
          LABEL
              COLOR 0 0 0
              FONT arial
              TYPE TRUETYPE
              POSITION CC
              SIZE 7
              BUFFER 1
              OUTLINECOLOR 255 255 255
          END
        END
   END
   SCALEBAR
      STATUS EMBED
      INTERVALS 2
      TRANSPARENT TRUE
      OUTLINECOLOR 0 0 0
   END
```

END

APPENDIX C
PYTHON AND PHP CODES

bs.py >>>Code to show render a map file

```
#!/usr/bin/python
import mapscript
import random
import cgitb;
cgitb.enable()
# path defaults
map_path = "/var/www/map/"
map_file = "ctabustops.map"
# Create a unique image name every time through
image_name = "bustops" \
        + str(random.randrange(999999)).zfill(6) \
        + ".png"
leg_name="bslegend"+".png"
leg_url="/var/www/tmp/"+leg_name
# Create a new instance of a map object
map =  mapscript.mapObj(map_path+map_file)
# Create an image of the map and save it to disk
img=map.draw()
img.save("/var/www/tmp/" + image_name)
leg=map.drawLegend()
leg.save("/var/www/tmp/"+leg_name)
# Output the HTML form and map image
print "Content-type: text/html"
print
print "<html>"
print "<header><title>Python Mapscript Hello World</title></header>"
print "<body>"
print """
<form name="hello" action="bs.py" method="POST">
<input type="image" name="img" src="//131.230.132.105/tmp/%s">
</form>
""" % image_name
print " CTA Bus Stops, City of Chicago"
print "</body>"
print "</html>"
```

CTA bus route file :

```
#!/usr/bin/python
import mapscript
import random
import cgitb;
cgitb.enable()
```

```
# path defaults
map_path = "/var/www/map/"
map_file = "ctabusroute.map"
# Create a unique image name every time through
image_name = "ctabusroute" \
        + str(random.randrange(999999)).zfill(6) \
        + ".png"
# Create a new instance of a map object
map =  mapscript.mapObj(map_path+map_file)
# Create an image of the map and save it to disk
img=map.draw()
img.save("/var/www/tmp/" + image_name)
# Output the HTML form and map image
print "Content-type: text/html"
print
print "<html>"
print "<header><title>Python Mapscript Hello World</title></header>"
print "<body>"
print """
<form name="hello" action="br.py" method="POST">
<input type="image" name="img" src="//131.230.132.105/tmp/%s">
</form>
""" % image_name
print "CTA Bus Route, City of Chicago"
print "</body>"
print "</html>"
```

Roads:

```
#!/usr/bin/python
import mapscript
import random
import cgitb;
cgitb.enable()
# path defaults
map_path = "/var/www/map/"
map_file = "roads.map"
# Create a unique image name every time through
image_name = "roads" \
        + str(random.randrange(999999)).zfill(6) \
        + ".png"
# Create a new instance of a map object
map =  mapscript.mapObj(map_path+map_file)
# Create an image of the map and save it to disk
img=map.draw()
img.save("/var/www/tmp/" + image_name)
# Output the HTML form and map image
```

```
print "Content-type: text/html"
print
print "<html>"
print "<header><title>Python Mapscript Hello World</title></header>"
print "<body>"
print """
<form name="hello" action="roads.py" method="POST">
<input type="image" name="img" src="//131.230.132.105/tmp/%s">
</form>
""" % image_name
print " All Roads, City of Chicago"
print "</body>"
print "</html>"
```

POLYGON: Chicago

```
#!/usr/bin/python
import mapscript
import random
import cgitb;
cgitb.enable()
# path defaults
map_path = "/var/www/map/"
map_file = "boundary.map"
# Create a unique image name every time through
image_name = "boundary" \
          + str(random.randrange(999999)).zfill(6) \
          + ".png"
# Create a new instance of a map object
map =  mapscript.mapObj(map_path+map_file)
# Create an image of the map and save it to disk
img=map.draw()
img.save("/var/www/tmp/" + image_name)
# Output the HTML form and map image
print "Content-type: text/html"
print
print "<html>"
print "<header><title>Python Mapscript Hello World</title></header>"
print "<body>"
print """
<form name="hello" action="boundary.py" method="POST">
<input type="image" name="img" src="//131.230.132.105/tmp/%s">
</form>
""" % image_name
print " City of Chicago, Polygon"
print "</body>"
print "</html>"
```

all layers:

```
#!/usr/bin/python
import mapscript
import random
import cgitb;
from time import gmtime,strftime
import datetime
cgitb.enable()
# path defaults
map_path = "/var/www/map/"
map_file = "alllayers.map"
t1=datetime.datetime.now()
# Create a unique image name every time through

image_name = "roadsalllayers" \
        + str(random.randrange(999999)).zfill(6) \
        + ".png"
# Create a new instance of a map object
map =  mapscript.mapObj(map_path+map_file)
# Create an image of the map and save it to disk
img=map.draw()
img.save("/var/www/tmp/" + image_name)

# Output the HTML form and map image
print "Content-type: text/html"
print
print "<html>"
print "<header><title>Python Mapscript Hello World</title></header>"
print "<body>"
print """
<form name="hello" action="alllayers.py" method="POST">
<input type="image" name="img" src="//131.230.132.105/tmp/%s">
</form>
""" % image_name
print " City of Chicago -> Spatial Database Accessed at:  "
print strftime("%a,%d %b %Y %H:%M:%S +0000",gmtime())
t2=datetime.datetime.now()
print "<BR>"
print "Total Time Taken = Query Database(PostgreSQL/PostGIS) +
MapfileGeneration(MapServer) + HTML Processing(Apache)"
print "<BR> "
print(t2-t1)
print "</body>"
print "</html>"
Template file
<html>
```

```
<head> <title>poi1</title></head>
<body>
   <form method=POST action="/cgi-bin/mapserv">
     <input type="submit" value="Click Me">
     <input type="hidden" name="map" value="/var/www/poi1.map">
     <input type="hidden" name="map_web_imagepath"
          value="/var/www/tmp/">
   </form>
   <IMG SRC="[img]" width=400 height=300 border=0>
</body>
</html>
```

VITA

Graduate School
Southern Illinois University

Ravi Dhungel
Ravi.dhungel@gmail.com

Kathmandu University
Bachelor of Engineering, Computer
August 2001

Research Paper Title:
WEB MAPPING AND APPLICATION TOWARDS A CLOUD: ENABLING A WEBGIS PROTOTYPE IN OPEN SOURCE ENVIRONMENT
Major Professor: Dr. Justin Schoof