Implementation of a complete Web based GIS Solution using Open Source Technologies

Siddharth Srivastava, Prashant Kumar, Ashu Krishna

Abstract: ***The benefits of using open source solutions are now obvious to the industrial applications. The Open Source GIS solutions are gaining popularity within the geo informatics community. Examples of popular FLOSS GIS solutions are GeoServer, MapServer, GRASS GIS, Quantum GIS. These when combined with open source tools such as MoNav, Mapnik etc make a robust mapping solution. This paper illustrates such an Open Source solution to web mapping service, by which the data preprocess and the construction of prototype system both used merely open source GIS software.***

1. INTRODUCTION

The open source license guarantees the freedom to read, redistribute, modify and use Software freely (OSI). This license plays an important role in breaking down barriers where the cost limits the use of public spatial data and the access to GIS tools. With an effort, the open source GIS has grown its popularity in geo-informatics community. This paper illustrates an open source GIS solution stack to web mapping. In this paper we are giving steps for implementing complete GIS Stack. It consists of different GIS tools for map rendering from spatial database, caching of rendered images, capability to edit map data, geo-coding and reverse geo-coding etc.
The development of such an open source system clearly illustrates the logistics of open system architecture, the orchestration of the interaction between open source softwares, the performance of integrated system, and a matter of concern to function limitations or bugs. As the geographic datasets used in this work contain a huge volume of collections, this, to some extent, is also an experiment that examines the capability of open source GIS solutions in building a large-scale GIS system.

Market Scenario
There are numerous commercial mapping services available including Google Maps and Bing Maps. But since none of them is open source they lack in providing the developers/organizations with the freedom and capability to extend such systems. Moreover many such systems impose restrictions on the organizations/agencies/indirect service providers of such applications. Such restrictions are not only limited to usage limitations, number of server requests but also on the usage of the proprietary data and tools so as to limit the competition from the service provider for similar applications
.
Challenge
In the project E-Paryatan, the main aim is to be able to build a self sustaining model for such services. The proprietary APIs restrict the usage of data and any manipulation or processing on them. Since the project aims at providing high quality information services to the users, it needs exclusive access to the geographical data. For example, Mapping APIs like Google, Bing, MapQuest etc. restrict the applications using their map services from displaying the data from third parties with their maps. They also do not allow performing any kind of processing on the information received or shown to the users via the APIs. It wouldn't have been possible to develop a robust application with the help of these services since the project also aims at providing personalized services to the users. Hence there is a need for setting up a complete solution at our own end.

The paper is organized as follows. In the next section the paper describes the tools used in generation of the model architecture. Accepted the architecture of the system is described, this paper also discusses the limitations and future scope of this approach.

2. Open Source GIS Solution

2.1 Prototype System

Prototype system consists of complete working stack of GIS tools. It provides functionality for worldwide routing, map download for offline usage, tile caching, automatic diff based synchronization of data, searching, It uses:

1. OpenStreetMap (**OSM**) is a collaborative project to create a free editable map of the world. Two major driving forces behind the establishment and growth of OSM have been restrictions on use or availability of map information across much of the world and the advent of inexpensive portable GPS devices. The maps are created using data from portable GPS devices, aerial photography, other free sources or simply from local knowledge. Both rendered images and the vector dataset are available for download under a Creative Commons *Attribution-ShareAlike 2.0* licence.

2. Mapnik for Map Rendering: Mapnik is a FLOSS Toolkit for developing mapping applications. It is written in modern C++ and has Python bindings that support fast-paced agile development. It can comfortably be used for both desktop map design and web development. It uses the AGG graphics library, which offers world-class anti-aliasing rendering with sub pixel accuracy for geographic data..

3. Tile Mill for Map Configuration: TileMill is built on a suite of modern open source libraries including Mapnik, node.js, backbone.js, express and CodeMirror. At TileMill's core is the Mapnik rendering engine. Mapnik's powerful, full-featured library including support for RGBA color, True Type fonts, raster's, patterns, and even SVG transforms All these features can be levereged using the simple, elegant syntax of the Carto styling language.

Carto is a CSS-like map styling language based on less.js. An example of Carto styling language is:

```
#road [zoom >= 15] {
::outline {
line-color:#000;
line-width:4.5;
}

line-color:#FFF;
line-width:1.5;
}
```

This results in an XML stylesheet that first draws the ::outline attachment and on top of that the inner white line.

The maps produced by TileMill are suitable for mobile applications and by using the MBTiles Format, they are self sufficient for offline usage.

4. Nominatim for Geo-Coding: Nominatim is a tool to search OSM data by name and address and to generate synthetic addresses of osm points (reverse geocoding). Nominatim is also used as one of the sources for the Search box on the OpenStreetMap home page and powers the search on the MapQuest Open Initiative websites. Nominatim can be used as an API as well as an installed module to get the geocoded results. In our implementation, we have deployed Nominatim on our server and queried the spatial database created using OSM data to get the desired results.

5. Monav as Routing Engine: **MoNav** is a Desktop / Mobile application that offers state-of-the-art fast and exact routing with OpenStreetMap Data. In contrast to many other routing applications, MoNav offers exact routing without heuristic assumptions and with very little computational work. Its routing core is based on Contraction Hierarchies.

6. Mod Tile for Caching: **Mod tile** is a system to serve raster tiles for example to use within a slippy map. It provides a dynamic combination of efficient caching and on the fly rendering. Due to its dynamic rendering, only a small fraction of overall tiles need to be kept on disk, reducing the resources required. At the same time, its caching strategy allows for a high performance serving and can support several thousand requests per second.

Mod_tile was originally written for serving the tiles of the main OSM map (Mapnik layer), but since is being used on a variety of different servers providing maps on top of OpenStreetMap data.

7. Open Layers: **OpenLayers** is an open-source implementation of a "Slippy Map" interface. It is a JavaScript library released under the BSD license, and is used on the OpenStreetMap homepage. The library includes components from the Rico JavaScript library and the Prototype JavaScript Framework. It provides an interface similar to Google Maps or Bing Maps and can be used with any data source. This provides us the flexibility in the E-Paryatan project to change the data source for maps etc, and yet the client side application is unaffected.

8. Osm2pgsql: osm2pgsql is a utility program that converts OpenStreetMap (.OSM) data into a format that can be loaded into PostgreSQL. It is often used to render OSM data visually using Mapnik, as Mapnik can query PostgreSQL for map data (Mapnik can read directly from other data source types including raw osm files directly.

9. Osmosis: Osmosis is use for automated synchronization of updates made in OSM data. It synchronizes the data with OpenStreetMap servers periodically. Hence the spatial database is always reflects the most recent information.

3. System Architecture

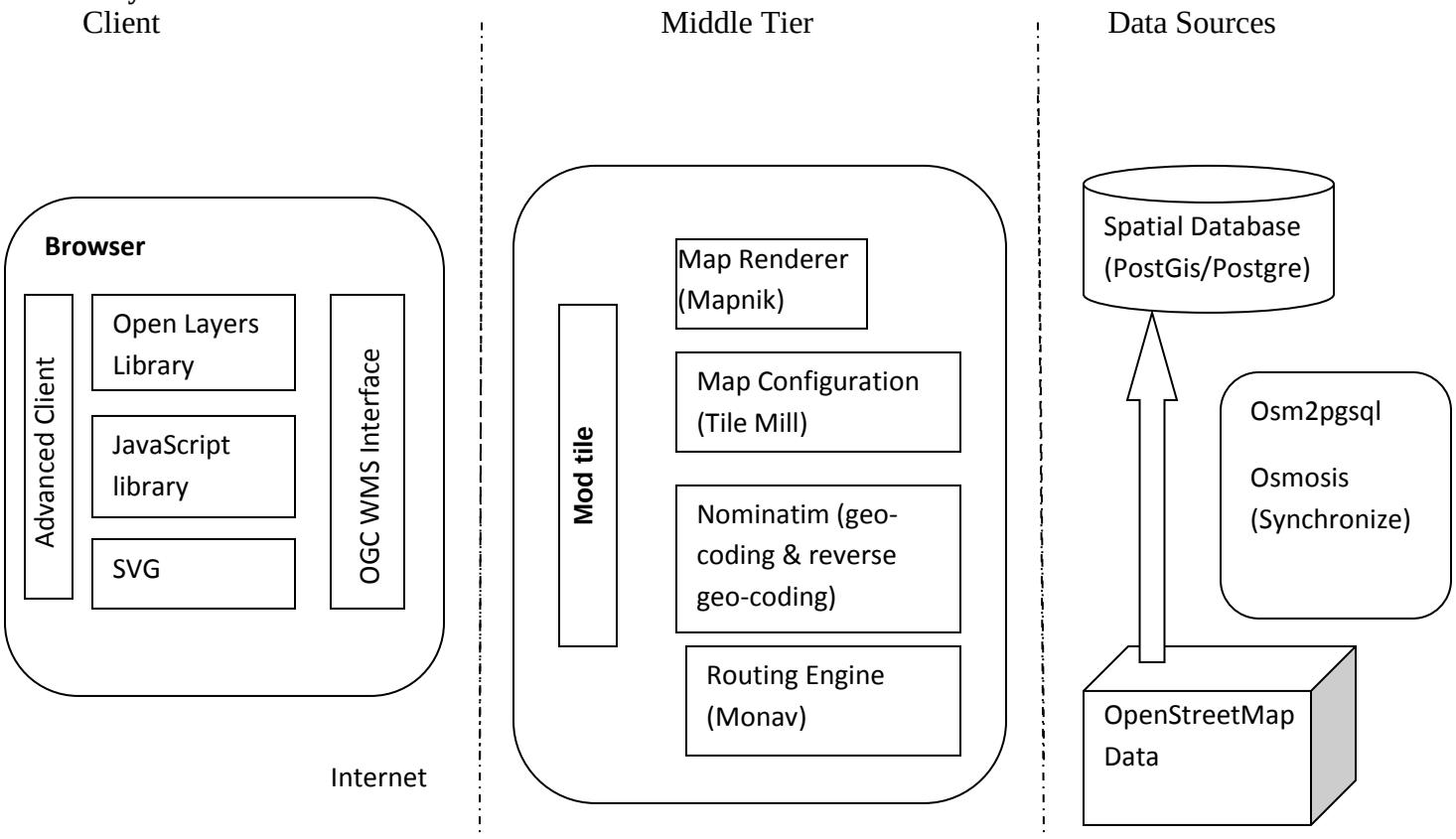

Fig 1: System Architecture

The architecture of the system proposed is structured in 3 tiers: client tier, middle tier and data sources. This approach emphasizes the independence of the various components of the system that can be deployed and combined

in different contexts. Figure1 illustrates the interaction among tiers, outlining the main components of the system and their logical position. The following sections describe these tiers in detail.

3.1.1 Client Tier

The user interaction with the system takes place in this tier. The user visualises web pages containing spatial information on their device through a common web browser. These web pages display interactive maps and monitor certain actions performed by the user, such as mouse clicks, zoom, etc.

3.1.2 Middle Tier

This tier contains the core services and functionalities of the system. A web application hosts the Web Server Pages which constitute the access point to the system for the users and the personalization and visualization service, in which several web services are deployed and exposed on the Internet. It consists of a Map Renderer, Map Configuration, a geo coding service and a routing engine.

Nomenclature:

- Tiles: The OSM data is organized in the form of *Tiles*. Tiles are small square map images which, typically get pieced together by the web browser running a javascript library for display "slippy map"
- Reverse GeoCoding: Reverse geocoding is the process of back (reverse) coding of a point location (latitude, longitude) to a readable address or place name. This permits the identification of nearby street addresses, places, and/or areal subdivisions such as neighbourhoods, county, state, or country. Combined with geocoding and routing services, reverse geocoding is a critical component of mobile location-based services to convert a coordinate obtained by GPS to a readable street address which is easier to understand by the end user.

Approach

1) Map Streaming

- The proposed architecture prefers pre-rendering of tiles over dynamic rendering of tiles. The tiles are rendered with the help of Mapnik and mode_tile. The tiles are managed with the help of TileMill which also helps in rendering the map for specific purposes if required.

  The advantages of pre-rendering the tiles than generating the tiles dynamically are:
  1) It is faster
  2) If the tiles are generated dynamically, the application can struggle at higher load levels if too many users try to look at unrendered tiles simultaneously.
- The scheduling capability of mod_tile is used for rendering tiles on the fly. This is needed in case the data for tiles have been updated on the OSM server.

2) Global Search and Routing

- Nominatim is used to search osm data by name and address and to generate synthetic addresses of osm points. When the client requests for details of a geographical coordinate, the query string is reverse geocoded and the details are fetched from the postgresql/PostGIS database. The response contains the information about the desired location.
- When the client searches for a place, the database is queried for the place and the result is geocoded to a valid geographical location on the map.
- Routing is accomplished with the help of Monav routing daemon. The monav preprocessor is used to convert the OSM routing instructions to a format to be used by the monav client. To be able to use monav on a large number of platforms, the monav is run as a daemon on the server.

3.1.3 Data Source

Nomenclature:

- OSM-XML: The OSM-XML format enlists the instances of the OSM data primitives i.e nodes, ways and relations.
- Node: A *Node* defines a single geospatial point using a latitude and longitude.
- Way*: A Way* is an ordered list of between 2 and 2000 nodes.
- Relation: A *Relation* consists of an ordered list of nodes, ways and sometimes also other relations as member of the new relation.
- Planet.osm: Planet.osm is the OpenStreetMap data in one file: Planet.osm and planet are used interchangeably in the following discussion.

Approach

Osmosis can be used to synchronize postgis data with osm data at a certain interval of time.

- For setting up the complete mapping infrastructure, OSM data is used which is provided in the form of OSM-XML and PBF file formats. The implementation uses OSM-XML format.
- The raw osm data cannot be directly used for generating the tiles. It is required to be stored into a spatial database table. PostgreSql is used as a database, and to enable it to store spatial data, PostGIS is integrated with it. To map the OSM-XML format to a spatial database, osm2pgsql is used. Spatial index greatly speeds up GIS data access as the indexes on normal fields do.
- An issue with hosting map on a custom server is that the updates on the OSM server, are not directly available. OpenStreetMap periodically releases there updated data. Since it is not feasible to update the entire planet file on the server, a difference based approach is used for this purpose. Osmosis is used to produce the change sets using the planet dump files and then the change sets are applied to the postgresql/PostGIS database. This enables the system to have

4. DISCUSSION

While there are benefits of the approach described here, it must also be recognised that there are some limitations which must be addressed. This section discusses these issues and how they can be overcome through further development and refinement. Furthermore, details of the evaluation which is required to assess the benefits of the approach are also provided. The client-server architecture offers many advantages. In particular, it does not require any additional software on the user device. However, there are some limitations with this approach. The client-server architecture has well known bottle-neck issues and it is recognised that beyond a certain number of clients, performance of the server is compromised (Vatsavai et al., 2006). This is potentially troublesome in a geo-spatial application where complex analysis is carried on the server. However, as the power of portable devices improves, there are possibilities to resolve this issue by utilising load-balancing techniques described by Vatsavai et al. (2006), where some computation can be carried out on the user device. The system described in this paper assumes that the user has a permanent connection to the Internet. This permits a constant stream of information to be exchanged between client and server. However, there are occasions when connection is interrupted or unavailable and contingencies to resolve this limitation are required. One possible solution involves developing client-side caching, whereby events and interactions are stored on the client and transmitted to the server when connection is available. This permits the server to continue user profiling and provide adapted and personalised maps using a complete interaction history. In order to improve map rendering and performance at the interface level, tile caching needs to be implemented on the server side. Software such as TileCache (http://tilecache.org) works by prefetching map content which is most likely to requested next by the user. This improves performance of the interface and the response time of map interactions. Now that a stable architecture has been implemented it is possible to use this as a test-bed for future developments. It would also be helpful to develop a wrapper around client application(especially if it is a mobile application) to contribute to OpenStreetMap project via the same interface extending the functionalities of existing open source projects like JOSM etc.

5. Conclusion

This paper presents a complete GIS stack of open source tools for prototype system for web mapping. The map service is a WMS compliant map server that uses a technically competitive solution to serve a large-scale GIS database on the Web. The map clients are lightweight cross-browser, which enable enhanced graphics and sophisticated interactivity. This successful development shows us once more how the Open Source solution benefits users, particularly those working in education, research, and in developing countries.

6. Acknowledgement

The authors express their sincere thanks to Dr. George Varkey, ED, C-DAC, Noida for his encouragement and permission to present/publish this paper. We are also thankful to all the team members of our Group for extending all possible support.

7. References

[1] Ku¨pper, A., 2005. Location-based Services: Fundamentals and Operation. John Wiley & Sons Inc.

[2] White Paper : The OpenGeo Architecture(http://opengeo.org/publications/opengeo-architecture/)

[3] An Open-Source Web Architecture For Adaptive Location-Based Services,Gavin McArdle, Andrea Ballatore, Ali Tahir, Michela Bertolotto, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. 38, Part II

[4] The Development Of Web Mapping Application Open Source GIS Solution
Xianfeng Song, Yasuyuki Kono, and Mamoru Shibayama

[5] An Efficient Web-GIS Solution based on Open Source Technologies: A Case-Study of Urban Planning and Management of the City of Zagreb, Croatia
Mario MILER, Drazen ODOBASIC and Damir MEDAK

[6] Medak, D., Pribicevic, B., Djapo, A., Medved, I. 2003. Open Source based Spatial Data Infrastructure - Why and How? /Proceedings of the ISPRS WG VI/3 Workshop: Geoinformation for Practice, Vol. XXXIV, Part 6/W11, 193-196, Zagreb.

[7] OpenLayers webpage, http://openlayers.org/

[8] Mapnik web page, http://mapnik.org/

[9] OSI - Open Source Initiative, http://www.opensource.org/licenses/

[10] OpenStreetMap Wiki, http://wiki.openstreetmap.org/wiki/Main_Page