MapBeing: An Efficient Vector Data Manipulation Framework for WebGIS

Moshiur Rahman*, Shafiuzzaman Hira†, Rayhanur Rahman‡ and Kazi Sakib§
Institute of Information Technology, University of Dhaka, Bangladesh
*moshi.iit@gmail.com †hira0322@yahoo.com ‡rayhan@du.ac.bd §sakib@iit.du.ac.bd

***Abstract*—Web Geographic Information Systems (WebGIS) have been introduced for fulfilling the growing need of Geographic Information Systems (GIS) in real world technologies evolving around the Internet. WebGIS can also be used to manipulate the published vector data according to the user need. However, challenges arise in case of ensuring seamless sharing, integration and interoperability of large volumes of geospatial data in such a complex web-based system. Especially, publishing of large volume interactive vector data might degrade the performance of WebGIS, due to extensive data transmission and rendering time. In this research, a WebGIS architecture named MapBeing is proposed which facilitates manipulating and publishing huge volumes of vector data on maps without degrading the data transmission performance. The proposed architecture consists of four core layers named User End, Request Handler, Service and Data Provider. The User End layer is constructed using an open source JavaScript library called OpenLayers which shows vector data on the interactive map presented on the web. The Request Handler layer processes data provisioning based on user interaction. GeoServer is used in the Service layer for map creation and modification and PostGIS, a database for storing spatial data, is used as the Data Provider layer. An open source vector data manipulation and visualization project is also introduced on top of these aforementioned frameworks as well. The experiment shows that along with the ability to manipulate and publish vector data, the architecture also achieves 23% lower usage in user bandwidth between server and client side.**

***Keywords*—Geographic Information System**

I. Introduction

Geographic Information System (GIS) is a collection of computer hardware, software and data for managing, analyzing and displaying all forms of geographic information. The vector data format is one of the spatial data types to store geographic data such as points, lines, polylines and polygons. For the proper usage of this data, a ubiquitous technology is needed for accessing these data from anytime and anywhere such as smartphones or desktops. In order to address those issues, WebGIS has been introduced for publishing and interacting vector data using web technologies. However, publishing of large volume geographic data on the map is really challenging in the web platform as excessive data transfer will be required which will increase data rendering time on the client side. WebGIS also provides several value added services to its users such as modifying and sharing of vector data [18], [13]. It has been widely recognized that future developments in GIS should be based on WebGIS to facilitate access and analysis of geospatial data on the web. Hence, the core manipulation of spatial data such as create, edit and delete should be considered with utmost attention.

For a highly interactive WebGIS based map, users often need to modify vector data to create and update geographical models such as rivers, cities, roads, terrains, hotels, hospital, banks, stations and other infrastructures. For this purpose, the server sends whole vector data to client sides such as desktop, smartphones or smartwears. Those client sides are then responsible for the rendering and modifying of vector data obtained from server side data providers. The main problem in such mechanism is the limited processing power and storage of aforementioned client devices. There is a network bandwidth cost as well for downloading the whole vector data each ocassion which is often infeasible and costly in real life scenario.

Recently researchers have gathered momentum in GIS domain regarding this context. TerraFly [15] and GeoHosting [4] are instances of those which focused on publishing and analyzing of vector data. S. Puri et. al. proposed a distributed algorithm for large scale vector data overlay processing [14]. A WebGIS framework for Vector Data sharing has also been proposed by F. Yin et. al. [18]. However, none of those contributions considered manipulation of vector data. Moreover, existing technologies for communication between server and client side involves only tile images which are the representation of those huge vector data residing on the server side. Those data are rendered here on the fly as per the requests from clients with required parameters. Hence, it is really challenging for the users to edit or delete geospatial shapes on the client devices efficiently since they are provided only partial images of the total vector data.

On top of these challenges, this research thrives to provide a WebGIS architecture named MapBeing for performing manipulation of vector data on the web without transmitting a large volume of data between server and client side. It helps users modify vector data on the map as well as publish with open source server side named GeoServer, client side named OpenLayers and data storage named PostGIS [11]. MapBeing is also fully capable of publishing and handling interactivity of shape files [9] provided by Environmental Systems Research Institute (ESRI) on the map without considering the data volume. Spatial data services are developed to control the interactions between the browser and the spatial data. Finally, based on the proposed architecture, a spatial data management and visualization framework has been designed and implemented. PostgreSQL and PostGIS are used to manage and

store GIS spatial data and metadata. GeoServer performs as the map server, an ASP.NET web server is used as the MapBeing server, OpenLayers is used for client side map rendering and interaction with vector data.

Experiments have been performed for the use case demonstration of the proposed architecture using Open Source vector data as well as comparison with one of the architectures which sends all the vector data to the client side. The architecture has been implemented as a web application in the managed .NET runtime environment. There are several modes of operation in the application so that user can select the required one from the toolbar. In the MapService Mode which is the default one, OpenLayers shows tile images obtained from the server no matter how large the vector data volume is. In the Edit mode, the image tile to be edited, can be selected and manipulated. The selected image tiles vector data are obtained from the server and after editing, it can successfully save the edited vector data. User can also draw and save data as vector format in the Drawing Mode. As the application does not retrieve the whole vector data at a time, the response time is better than conventional applications and bandwidth cost is also reduced.

In comparison with a production level legacy GIS solution which consumes all the vector data from the server side, a significant amount, which is about 23%, of the bandwidth cost is reduced as a lower amount of vector data have been communicated between the client and the server. Such outcome demonstrates how MapBeing can effciently facilitate users to publish and modify vector data without occurring much bandwidth transmission.

II. Background and Related Work

The GIS development has drawn attention of researchers for its growing usage in business, research and education etc. Hence, GIS researches focuses on efficiently vector data publishing, analyzing, sharing and also modifying. In this section, the most notable work on GIS has been highlighted.

Many definitions for the term GIS have been proposed in the literature, each considering the functionality of the system from diversified perspectives. All of those focus on three different aspects: [5]

- GIS is a collection of computer tools to perform geographic analysis and simulations
- GIS is supported by a set of data structures and algorithms to represent, retrieve and manipulate geographic information
- GIS is a utility that helps people make decisions in tasks related to geography.

In principle, GIS is a set of frameworks with underlying computational techniques which supports complex data analysis and simulation of real world appliances in geographical domains.

GeoHosting [4] proposed a WebGIS architecture called GeoHosting as a Spatial Data Infrastructure (SDI). The main objective of this architecture is to offer services supporting the creation of a spatial data sharing system with possibility to publish data for any user having access to the Web. As it is developed focusing on the spatial data sharing compatibility, it lacks the ability to modify the vector data.

TeraFly [15] is a high performance WebGIS. It uses Internally Distributed Multi-threading method to achieve high performance and semantic R-tree to search an object on both spatial and semantic information. It gives a high speed heterogeneous data publishing architecture but it does not provide much facility to modify large volume vector data efficiently.

R. Puri et. al. [14] developed a Parallel and Distributed algorithm for large scale Polygon Overlay processing with the help of MapReduce framework [6]. It can find the result of a query consisting of two vector overlay layers with large amount of vector data. It is also implemented on General Purpose Graphics Processing Unit to accelerate flotaing point operations during rendering. Although this algorithm can find result by processing two vector overlay but cannot edit data of those overlays.

F. Yin et. al. [18] proposed a WebGIS framework for Vector Data sharing based on open source projects. It focused on practical implementation of large vector data sharing and interoperability. The authors have divided this application into the application layer, service layer, function layer and storage layer using following open Source libraries: PostgreSQL, GeoServer, OpenLayers and TileCache. As it is developed to facilitate user for sharing large scale data, it does not consider the modification of vector data on the map which is very important for an interactive mapping system.

A Spatial Data Infrastructures (SDIs) has been implemented in their paper by Steiniger et. al. [17]. The proposed architecture is based on Free and Open Source Software (FOSS) [16] so that users can buy a full SDI with a limited cost. The SDI is developed with Web Map Servers, Web-GIS Servers for data processing, spatial DBMS for storage, registry/catalogue and metadata software. In this architecture, a complete GIS is proposed but interaction between vector data and users on the map is not considered.

An Web based GIS System has been developed targeted for the water resource management [7] on top of open source GIS libraries. It is capable of publishing, storing vector data as well as raster data. XMLParser is used here while sending and receiving data format for communication between server and client. This architecture was developed and applied to spatial data management and utilized that data for problem analysis and solving such as monitoring natural disasters, solving water insufficiency in agricultural areas etc. The main part of this work was used at GIS in water resource management field. Hence this architecture does not help users to modify or to create vector data from map.

From the above discussion, it has been found that the manipulation of large vector data in WebGIS is still time and data intensive. Currently, the browser itself does all the collaboration between GIS and vector data in spite of being unable to handle large amounts of data because of limited resources and low processing power. Hence, such GIS architecture for vector data publishing is required which can keep a tight bound on the data transmission and rendering time of the service.

III. MapBeing Architecture

In order to address the issues stated above, in this section, a WebGIS architecture named MapBeing is proposed which has the capability to edit and publish vector data without triggering huge bandwidth consumption. This section provides thorough details on how MapBeing manages to do so. First, core modules of MapBeing is discussed followed by the policy according to which, modules interact with one another.

A. Overview

In this section, a vector data manipulating and publishing architecture for GIS solutions named MapBeing is proposed which facilitates users with interactive capabilities of modification, deletion, presentation, analysis and capture of spatial representation of real-world features such as roads, buildings and terrains. In order to perform such operations on vector data, MapBeing transmits just the data of specific shapes which is requested from the client side and thus shakes off a huge amount of time during data transmission between the client and server side. This section highlights the detailed architecture of MapBeing and how it manages to reduce client-server computational and communication overheads. The top level view of the framework architecture is shown in Fig. 1. The architecture is divided in three servers and one storage.

A new vector data manipulating and publishing architecture for GIS solutions is proposed in this paper. GIS data set is the spatial representation of real-world features such as roads, buildings and terrains. The interaction, modification, deletion, presentation, analysis and capture of vector data are all done by GIS. For manipulating vector data, MapBeing sends just the data of specific shape which is requested from the client side.

The proposed architecture facilitates the modification of large amount of vector data on the map and raster data which are not possible to be handled by the browser. It also focuses on the improving the rendering time issue in client side data rendering. Finally, the facility to manipulate vector data is provided as well in this new framework architecture.

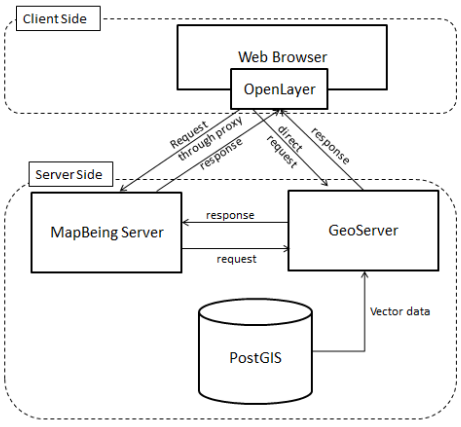

Fig. 1. Top Level View of MapBeing

B. Web Client

The web client, as shown in Fig. 1, listens to the user interactions on the map and communicates those with the MapBeing Server. This task is accomplished by OpenLayer which receives the user requests and sends it to the MapBeing server. Having completed the corresponding tasks, the response from the MapBeing server is sent back to the user by the Web Client. The MapBeing server is used here for the manipulation purpose of the vector data. However, the OpenLayers directly make service calls to GeoServer to display the data. It acts as a transparent layer on the map which makes a canvas on it and draws the shapes responded by the GeoServer.

C. MapBeing Server

The MapBeing server is responsible for connecting the OpenLayers with the GeoServer. As a HTTP service, it has the conventional task to provide web content to the browser through the Internet and processing server scripts and sending an appropriate response to client requests. However, the main responsibility of the MapBeing server here is to process user manipulation requests and sending those back to the GeoServer. Different services of GeoServer are used for different purposes which will be mentioned later in this section. Some of the services can be used directly from browser while others require the MapBeing server to be involved in some of its processing tasks. It takes co-ordinate points obtained from user interaction in the map images through the OpenLayers, gets the vector data information of co-ordinates after manipulation and finally sends request back to the GeoServer.

D. GeoServer

The GeoServer [8] plays the most critical part in this architecture as it establishes the communication between MapBeing Server and Web Client. It also involves PostGIS in the proposed architecture for storage purposes of manipulated vector data. GeoServer provides these three following services:

1) Web Feature Service (WFS): WFS provides an interface for making requests to get geographical features. Two types of WFS have been used in this architecture. The first one is generic WFS which allows only querying and retrieving of features. Meanwhile, a transactional WFS (WFS-T) [8] is also used which allows creation, deletion and modification of those. It also processes HTTP request of clients and executes appropriate operations to serve those.

2) Web Map Service (WMS): A standard protocol called WMS has been utilized here to serve geo-referenced map images over the Internet. These map images can be generated by a map server using data from the GIS database. Although it usually serves the map in bitmap format, vector graphics can also be served in SVG or WebCGM format.

3) Web Coverage Service (WCS): Digital geospatial information representing space time-varying phenomena is retrieved by the WCS. It also provides access to coverage data in web forms. It allows clients to choose portions of a server's information rather than the providing whole. The Web Browser directly communicates with the GeoServer through WMS

service. The map images got from this service is displayed on the browser. As these are the only images, these cannot be manipulated directly like vector data. This is where the MapBeing server is needed to process the clicked image position to get the specific vector data. This is done by WFS-T service of the GeoServer.

E. PostGIS Storage

The PostGIS database is used for vector data storage. It provides data to GeoServer which exposes its services to the user through the browsers. This solution lessens the tasks of the browsers to manipulate vector data that automatically enhances client-side performance.

F. Core Layers of MapBeing

The component stack of the proposed framework is represented in Fig. 2. It is a layered architecture where each of those serves different responsibilities.

- User End layer
- Request Handler layer
- Service layer
- Data Provider layer

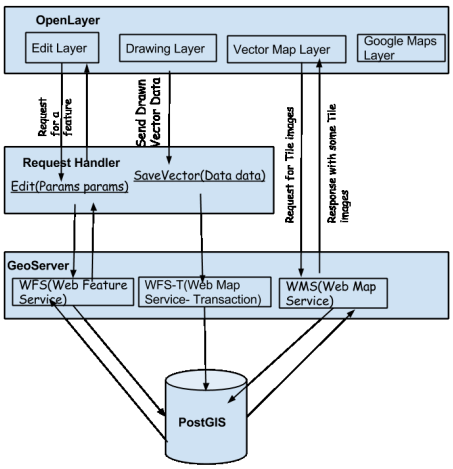

Fig. 2. Architecture of MapBeing

The User End layer abstracts the OpenLayers and interacts with the users viewing locations in map. It has the responsibility to take user commands and show corresponding changes in the map. The Request Handler layer processes user demands and requests Service layer for required services appropriately. The Service layer provides the three previously mentioned Geo services to the User End directly or through Request Handler and thus can be called as GeoServer layer. The Data Provider layer stores vector data and supplies it to the GeoServer on demand.

The first layer, named OpenLayers, consists of four components: Edit layer, Drawing layer, Vector Map and Google Map. OpenLayers is a transparent layer on the map where the locations are drawn. It is a canvas drawn with points, lines, rectangles and various other shapes representing the vector data locations. Google Map is a service provided by Google which is used here for the background of the vector data locations in the browser. The Vector Map component requests directly to the GeoServer with map b-box, zoom-level and layer information as parameters. As a response, it gets some tile images and places those on the map. The Edit and Drawing layers are used for the map data manipulation. Both of these components contact with Request Handler. The Edit layer is responsible for editing map features on OpenLayers and the Drawing layer is responsible for saving the manipulated information.

Request Handler layer is the web server layer which is in a sense, the middle layer between OpenLayers and GeoServer Layer. It has the task to do the manipulation processes on vector data. It consists of two components- Edit and Save component. The Edit component takes user interaction information on the OpenLayers tile images as input and provides the vector data information about that area. It also makes all the manipulation of the data. The Save Vector component takes the drawn data information about users and saves it using GeoServer Service.

The third layer is the GeoServer layer. It provides the WFS, WMS and WCS services to the User End and Request Handler layer. And it also offers WFS-T which is an advance version of WFS service. For manipulation task WFS and WFS-T services are used. These services are accessed through Request Handler layer. The viewing is done by using WMS Service. It is directly approached by the OpenLayers.

The fourth layer of MapBeing is Data Provider layer. It is the storage layer and the warehouse of the framework. The vector data which can also be termed as geographical data is stored here. The database used for this storage is PostGIS database.

In principle, User End layer does the user interactivity with vector data, Request Handler receives requests and sends responses to User End layer, Service layer is responsible for rendering, manipulating of vector data and finally Data Provider stores user data. This four layered architecture achieved the minimization of data bandwidth cost of manipulation of vector data in web based GIS.

IV. Experiment and Result

In this section, experimental setup for the proposed WebGIS system architecture is discussed followed by the results.

The proposed architecture is based on free and open source software where MapBeing used OpenLayers as client side, GeoServer as web server and PostGIS as storage for spatial data. MapBeing also has a server to process XMLHTTP requests from OpenLayers. OpenLayer generates requests to GeoServer with map's view-port and zoom-level as request parameters for tile images. GeoServer also generates request to PostGIS for vector data according to request parameters and creates tile images of those vector data on the fly. Then the tile images are sent to OpenLayers to render those images on the map-overlay. When OpenLayers need a specific shape data

based on user interaction such as mouse click, it requests via MapBeing server to GeoServer.

The detailed technical description is given below. OpenLayers is an open source object oriented mapping library based on HTML5 canvas element. It also has capability to draw with SVG (Scalable Vector Graphics) techniques. It is divided into three parts as Map layers, Edit layer and Drawing layer. Map layers is responsible for displaying tile images on above base layers such as Google Maps, Open Street Maps or Microsoft Virtual Earth. More than one layer can be added at a time such as Road Map, Administrative Map, River Map, and Rails Way Map of a country. For displaying base map, OpenLayers uses predetermined API of corresponding map providers and displays maps of vector data based on OpenLayers request routed to GeoServer for tile images.When user clicks on the tile images to edit a shape, edit layer request GeoServer with required parameters via MapBeing server for a specific shape's vector data. By analyzing parameters, GeoServer can uniquely identify a shape and sends vector data of that shape. This will facilitate users to modify properties of spatial shape data with the help of edit layer. After editing, the edit layer sends modified data to GeoServer via MapBeing server then GeoServer store this data in PostGIS. Drawing layer is responsible for creating new map with just sketching of Lines, polylines, polygons and points. After drawing, it sends drawn data to GeoServer via MapBeing server and GeoServer stores this to PostGIS by creating new shape file.

MapBeing uses ASP.NET web application as MapBeing server where OpenLayers sends user modified or drawn data. The MapBeing server receives those data from client side and sends it to GeoServer after pre-processing. Then GeoServer modifies those data and sends to PostGIS. With the help of WFS , GeoServer sends data to MapBeing server as user request from client side with OpenLayers. After editing those received data, GeoServer updates spatial table in PostGIS. MapBeing server also has a management module to communicate with GeoServer and OpenLayers. It has a processing module to process in coming request from client-side and added extra information to fit the request parameters for requesting GeoServer. It also has a data management module by which it maintains data uploading and data authorization. GeoServer has been customized so that MapBeing server can use it to upload user data. This customization has been done by modifying the catalog.xml which is made from GeoServers data source point to our data storage where MapBeing server uploads users data.

PostGIS is a spatial database extender for PostgreSQL object-relational database. It adds support for geographic objects allowing location queries to be run in SQL. GeoServer just execute some query as MapBeing severs request. After receiving PostGISs response, GeoServer sends feature data to OpenLayers through MapBeing server.

OpenSource vector data is used to test how MapBeing performs in real life operations. All the experimental data was downloaded from http://www.gadm.org/country or from GeoServer's default provided set of data. USA population data, comes with GeoServer installation, named 'topp:states' has been used to check modification capability of MapBeing. In order to compare MapBeing's performance, a legacy GIS system [18] has also been implemented alongside. The Building [1] data is used to show that MapBeing does not consider data size to process them. Windows 8.1 x64 operating system, Intel Core 2 Duo E8400 @3.00GHz and 2.00 GB RAM were used as experimental platform of MapBeing.

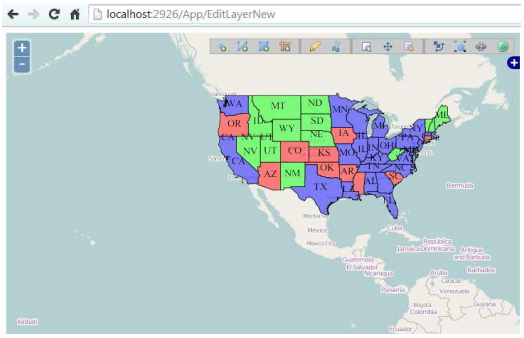

Fig. 3. Web Map Service Layer

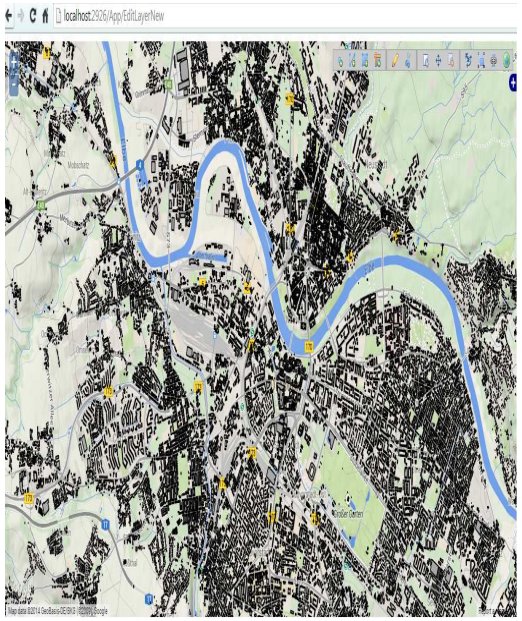

Fig. 4. Web Map Service Layer (Large Data Set)

MapBeing can be run in several mode of operations. User can select a mode to work with from toolbar. There is a layer chooser tool to select a layer to work with. Different types of map such as Google Map, Open Street Map are also available as base layer.

By Default, MapBeing remains in Map Service mode displayed at Fig. 3. When a user wants to see a layer, OpenLayers Map Service [10] requests to GeoServer for that layer's tile images. Then GeoServer gives responses to OpenLayers and OpenLayers show tile images where they should be displayed.

The example of large data rendering in the map is shown in Fig. 4. Without considering volume of vector data MapBeing is capable of display tile images on the map. Vector data

ESRIs data name-size	Source of Vector Data	.shp file size (W)	Number of Shapes (N)	Bandwidth in MapBeing (W/N)
USA population (202 KB)	Default Geo Server installation	183 KB	49	.003
Virginia Roads (349 MB)	http://download.geofabrik.de/northamerica/us/virginialatest.shp.zip	272 MB	755483	.0003
gis.osm roads (14.6 MB)	http://www.divagis.org/gdata	7.68 MB	50000	.0001
bangladesh latest.osm (12.5 MB)	http://download.geofabrik.de/asia/bangladesh.html	10.2 MB	4000	.003
India Roads(328 MB)	http://www.geofabrik.de/data/shapefiles.html	240 MB	85516	.003
Indonesia Roads (126MB)	http://download.geofabrik.de/asia/indonesia-140908.shp.zip	88 MB	344713	.0003

TABLE I. COMPARISON OF DOWNLOADED DATA SIZE BETWEEN MAPBEING AND CLIENT SIDE RENDERING ARCHITECTURE

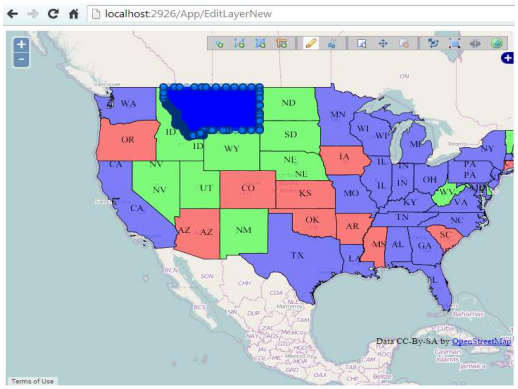

Fig. 5. Edit Layer

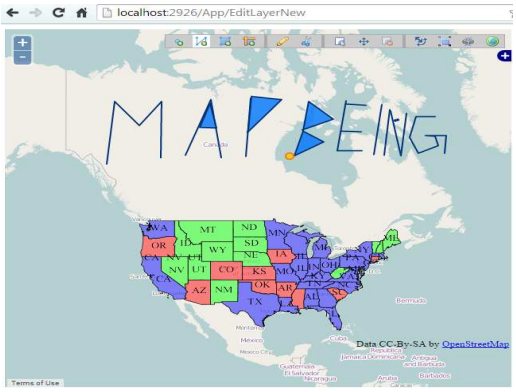

Fig. 6. Drawing Layer

editing mode is shown at Fig. 5. When User clicks to go to edit a shape then OpenLayers Edit Service [10] receives click position, Maps' bbox, layer's information and request to MapBeing server. After analyzing the request, MapBeing server returns requested vector data as Json format to OpenLayers. With these data, user can modify and after modification clicks *save* button. After that OpenLayers save strategy sends edited data to MapBeing server.

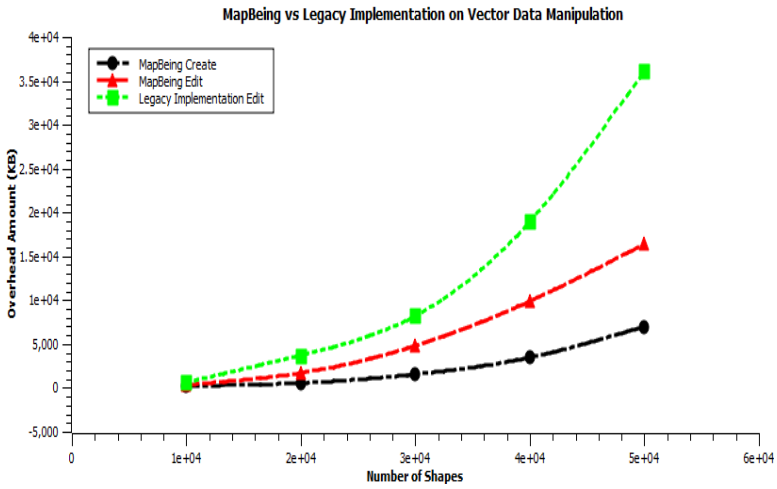

Fig. 7. MapBeing Performance in Creation and Modification

Another value added service of MapBeing is the drawing capability shown at Fig. 6. User can draw shapes and by using OpenLayers Save Strategy [10] user can save drawn data as vector format. User also can select drawing tools from toolbar situated on above of map window. Lines, Polygons, Point tools are available to draw.

There is a comparison at Table 1 on the basis of downloaded amount of data between server side rendering GIS and client side rendering GIS architecture. User can publish, analysis and manipulate vector data on the map by client side rendering GIS. But at the first time of map display, client side rendering GIS downloads whole .shp file from server side to client. On contrary MapBeing downloads just few tile images rendered in the server side. To compare downloaded data size, Comparison Matrix (CM) was defined. From data storage, numbers of shape are known from ESRI's shapefile [12] and its size:

$$N = number\ of\ shapes\ in\ .shp\ file, CM = W/N$$

$$W = size\ of\ .shp\ file\ in\ Mega\ Byte(MB)$$

The larger value of CM means that more data should be downloaded from client side to server side. Moreover, as it can be seen in Table 1, client side rendering architecture download whole .shp file from server side to client side but MapBeing does not do that. MapBeing download only *CM* sized vector data as per user request. As MapBeing does not consider data size, it performs well even when data size increases. It is also prominent from the Fig. 7 that MapBeing shows linear growth of data transmission in the case of random shape file creation and modification. On the other hand, legacy implementation shows exponential growth in transmitted data volume on the same ocassion. Overall, MapBeing manages to improve the data rendering time upto 23% in comparison with legacy implementation. Such behavior proves that, MapBeing is capable to scale from the aspect of large volume of geospatial data publishing and manipulating triggered by the users.

On the basis of simulation and result analysis discussed above, MapBeing is efficient in downloading vector data file for manipulation purposes such as create, edit and delete. It downloads only the corresponding shape's vector data at time when user wants to edit instead of downloading the whole vector dataset from the server. Thus the proposed architecture facilitates both the manipulating and publishing of geospatial data on the web based GIS as well as keeping the data transmission level significantly lower.

In principle, the proposed architecture handles the problem of large data transfer in publishing map on the web by user interaction based data provisioning techniques rather than providing the whole map data at once. The feature toolbar of manipulating the vector data on the map is also presented here. The WMS service of GeoServer is used in this architecture for generating the tile images to be shown on the map by OpenLayers. On the basis of user interaction on a position of the map, WFS service is used to get the corresponding vector data. Thus rendering time and data transmission bandwidth are minimized as well. Finally WFS-T is used to save the manipulated vector data on the server side. From the experiment, it has also been shown that minimal amount of data has been transfered between client and server side.

V. Conclusions

WebGIS has gained popularity because of the consistent availability and use of geographic information everywhere, especially in desktop and mobile clients. However, a barrier in the mainstream adoption of WebGIS is the transmission of large vector dataset between server and client sides. The unavailability of data manipulation platform in the web based map is another limitation of currently available WebGIS solutions as well. To address these problems, MapBeing is proposed in this work which facilitates both manipulating and publishing vector data on the web based GIS. From the experimental results, it is also observed that, MapBeing triggers very low amount of vector data transmission between client and server sides. Thus, MapBeing facilitates users with a vector data manipulation and publishing framework that is efficient as well. The future scope of this research includes providing of a multiple user based map manipulation interface for the same set of vector data. Moreover, this research only focuses on shape file format, in future, other types of vector data format such as GeoJSON [3], KML [2] can be incorporated.

VI. Acknowledgment

This research endeavor is funded by the UGC Research Grant, 2014-15 by University Grant Commission, Bangladesh with the reference no: Reg/Prosha-3/2015/48750

References

[1] Toulouse: streets, railways, waterways, July 2014.

[2] S Bacharach. Ogc approves kml as open standard, 2008.

[3] Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Tim Schaub, and Christopher Schmidt. The geojson format specification, 2008.

[4] Karel Charvat, Petr Horak, Martin Vlk, Jachym Cepicky, and Stepan Kafka. Geohosting–publish your spatial data yourself. *IST Africa, Kampala May*, 2009.

[5] GEOG4340 Q Cheng. Geographic information system. 1995.

[6] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[7] Prattana Deeprasertkul and Royol Chitradon. An internet gis system to support the water resource management. *information science*, 1(01), 2012.

[8] Justin Deoliveira. Geoserver: uniting the geoweb and spatial data infrastructures. In *Proceedings of the 10th International Conference for Spatial Data Infrastructure, St. Augustine, Trinidad*, 2008.

[9] ESRI ESRI. Shapefile technical description. *An ESRI White Paper*, 1998.

[10] Erik Hazzard. *Openlayers 2.10 beginner's guide*. Packt Publishing Ltd, 2011.

[11] Leo Hsu. Postgis in action, 2011.

[12] Feng LIU, Ji-xian ZHANG, and Hai-tao LI. Research and application of shp file format in land over/use map updating [j]. *Science of Surveying and Mapping*, 6:042, 2006.

[13] Moshiur Moshi, Nadia Nahar, Rayhanur Rahman, and Kazi Sakib. Mapbeing: An architecture for manipulating and publishing vector data in web based geographic information system. In *Software, Knowledge, Information Management and Applications (SKIMA), 2014 8th International Conference on*, pages 1–7. IEEE, 2014.

[14] Satish Puri and Sushil K Prasad. Efficient parallel and distributed algorithms for gis polygonal overlay processing. In *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, pages 2238–2241. IEEE Computer Society, 2013.

[15] Naphtali Rishe, Shu-Ching Chen, Nagarajan Prabakar, Mark Allen Weiss, Wei Sun, Andriy Selivonenko, and D Davis-Chu. Terrafly: A high-performance web-based digital library system for spatial data access. In *ICDE Demo Sessions*, pages 17–19, 2001.

[16] Stefan Steiniger and Erwan Bocher. An overview on current free and open source desktop gis developments. *International Journal of Geographical Information Science*, 23(10):1345–1370, 2009.

[17] Stefan Steiniger and Andrew JS Hunter. Free and open source gis software for building a spatial data infrastructure. In *Geospatial free and open source software in the 21st century*, pages 247–261. Springer, 2012.

[18] F. Yin and M. Feng. A webgis framework for vector geospatial data sharing based on open source projects. In *Proc. of 2009 International Symposium on Web Information Systems and Applications (WISA09)*, pages 124–127, May 2009.