Spatial Cloud Computing (SC2) on the AppEngine Platform

Prabuddha Ghosh RMSI (Prabuddha.ghosh@acm.org)
Sandip Roy RMSI (Sandip.roy@rmsi.com)

Section 1: Abstract

In the computer field platforms evolve. We started with mainframes, moved to PCs, then moved to a client server model with web based software and now we are moving on to cloud based computing. The purpose of the work described in this paper is to evaluate the cloud as a platform for GIS better known as Spatial Cloud Computing. Specifically we evaluate Google AppEngine as a platform for spatial query systems. After the transition of GIS from Desktop based systems to mostly server based systems we are now at the cusp of a new transition, from server based GIS to Cloud Based GIS. While Cloud computing has been around for a while and is considered mainstream in the fields of CRM, ERP with upto 15% of all new deployments being Cloud deployments; the field of Spatial Cloud Computing (SC2) has emerged as a real option in the year 2010. In this paper we shall look at what is the status of the field of SC2 as well as what are the compelling reasons driving its adoption. We shall start by defining what constitutes cloud computing including what is meant by Spatial Cloud Computing (SC2). We shall present work on SC2 being done at RMSI on the Google AppEngine platform using the open source platforms of JTS Topology Suite and JavaGeoModel. We shall explain the challenges of developing spatial software on a platform (Datastore) which has no database level support for spatial attributes. This support has been built by us in user space using the JavaGeoModel and JTS Topology Suite.

Section 2: Background

Let us first define a few terms

GIS: While we are all know what is GIS; for the purpose of this paper let us use a concise definition. Geographical Information Systems are any information system where the data stored has a spatial/geographical component and the system actively utilizes this component. So for example this could range all the way from traffic density analysis with demographic data to something simple like querying for all the shops within 1 km of a location.

Cloud Computing: According to the NIST definition [1] cloud computing has the following 5 characteristics - On-demand self-service (users are able to provision resources without talking to a human), Broad network access (Same resources are available over the network to variety of platforms from mobile to thin clients to Desktops), Resource pooling (Hardware and system Resources are shared across a number of different applications) ,Rapid elasticity ( Hardware should scale up and down according to demand) and Measured Service (You pay for only the resource you consume). Further Cloud Computing can be available in a number of modes – Infrastructure as a Service (e.g. Amazon Web Services which provides machine images for hire and we can deploy our own application software and customization on top of it). From a GIS perspective an example would be renting Arcgis server images on the Amazon platform), Platform as a Service (e.g. Google AppEngine which provides its own webserver, datastore and other platform apis and we develop software using these APIs and deploy on Google’s platform) or Software as a Service (where we rent usage on a software and the same software is used by many users a prime example being Gmail). Further Clouds can be Public like Amazon, Microsoft Azure;
private i.e. Deployed inside a company’s own architecture but providing internal self services, rapid scaling, resource sharing, measured billing; or hybrid where some data stays on a company’s own servers e.g. Using Google’s Secure Data Connector and the rest stays in the public cloud say AppEngine.

Our focus in this paper shall be on how we can use a PAAS namely Google AppEngine and build a SAAS based GIS –Cloud GIS on top of it. Google AppEngine supports apps written in several programming languages. With App Engine's Java runtime environment, you can build your app using standard Java technologies, including the JVM, Java servlets, and the Java programming language—or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby. App Engine also features a dedicated Python runtime environment, which includes a fast Python interpreter and the Python standard library. The Java and Python runtime environments are built to ensure that your application runs quickly, securely, and without interference from other apps on the system. For the purpose of the current work we have concentrated on the Java environment keeping in mind the greater availability of programmers with exposure to Java in the GIS industry. One interesting factor about AppEngine is that it uses a NoSQL based datastore for storing data instead of an RDBMS. The App Engine datastore is a schemaless object datastore, with a query engine and atomic transactions. The Java SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces, as well as a low-level datastore API.

A critical part of our adaptation of the Google AppEngine to provide support for GIS is to use Geohashing. Geohash is a latitude/longitude geocode system invented by Gustavo Niemeyer when writing the web service at geohash.org, and put into the public domain. It is a hierarchical spatial data structure which subdivides space into buckets of grid shape. Geohashes offer properties like arbitrary precision and the possibility of gradually removing characters from the end of the code to reduce its size (and gradually lose precision). As a consequence of the gradual precision degradation, nearby places will often (but not always) present similar prefixes. On the other side, the longer a shared prefix is, the closer the two places are. [2]

Section 3: Approach

While a number of different companies have launched Cloud Based versions of their existing GIS offerings - such as ERDAS Apollo, Mapinfo Stratus, ArcGIS mostly companies have stuck to either supporting Infrastructure as a Service – e.g. ArcGIS images on an Amazon EC2 server or created their own SAAS Apollo and Stratus. However we believe our approach is unique in that we are taking one of the mainstream PAAS AppEngine and building a GIS SAAS on top of it. This means unlike ArcGIS.com we don’t have to maintain a vast server farm and can instead offload this task to Google AppEngine’s server farm. This lets us concentrate on our core competency of GIS.

When undertaking this work we had a few simple aims- be able to read a standard industry format (hence we chose KML as the input), perform Spatial Queries and Display Spatially enabled data and reports all on a Cloud platform hence demonstrating the suitability of the Cloud for GIS applications.

The first and biggest challenge we faced is that the Google AppEngine (GAE) datastore does not have Spatial support in the database itself aka Oracle Spatial etc. So the only option is to build the support for Spatial Indexes and Spatial Queries in the User space. For building this support we have leveraged an existing open source project JavaGeoModel. JavaGeoModel in turn supports the creation of spatial indexes utilizing the Geohashing concept explained in Section 2. This means we can create and store our
own spatial indexes/geohashes when data is loaded into the AppEngine Datastore. In turn when doing spatial queries we can utilize these user space indexes to efficiently do spatial searches, bounding box queries and nearest neighbor searches.

For the actual software we have used Google Web Toolkit (GWT) and followed the Model View Presenter paradigm as recommended for GWT. As this software is based on AppEngine what this means is that we can deploy almost any other Java library (with certain limitations) on the backend. What this means is that we can utilize the highly versatile and open source JTS Topology Suite for executing complex GIS operations such as finding the intersections of parcels on a particular layer with a polygon on another layer. While there is some amount of interfacing and conversion code needed to allow GWT geometries to interact with JTS geometries; this approach still allows us to provide complex GIS functionalities without having to rewrite them. This is where the advantage of a PAAS based approach really shines through as compared to say the traditional Google Map based mashups where only a limited amount of processing can be done based on the JavaScript API exposed by Google. While we do use Google Map as our viewer and renderer, AppEngine does give us the freedom to replace it with something else written in Java.

Section 4: Results

Currently we have created a system and deployed on appengine a system www.rmsicloudgis.appspot.com which actually allows us to load KML files from anywhere in the world; the system then parses these files and stores in tables in the Datastore – 1 table per layer plus some administration tables. Once stored we can utilize our self developed Layer Manager to select active layers and utilize our Query Engine to query the various layers based on Address geocoding (utilizing the googlemap geocoding api), query by Geographical drill down State- District- Local- Body, geographical coordinates and/or name with spatial filters applied based on distance thresholds.

Further we can display additional data about the features as captured from the KML file and stored in the Datastore in popup windows and generate reports such as show tower coverage etc.
The unique feature of this type of development is we are able to get response times on par with Google Map for panning, zooming and other basic functionalities while being able to add custom functionalities which we would not be able to add to Google Map or ArcGIS based mashups.

From the perspective of an Independent GIS Software firm some of the advantages of this approach are-

1. Enables selling GIS to new type of customers- those not satisfied with simple mashups but also not willing to invest in entire systems and IT setups.
2. Enables us to concentrate on our core competency – Custom Software development and Data creation instead of having to spend lot of effort on managing infrastructure and licenses.
3. Enables us to sell our software as a service in a subscription based model.
4. Enables fast to market delivery by leveraging existing open source libraries.
5. Most importantly allows us to scale on Google’s data center and provide response times similar to Google Map something is very difficult to do on traditional Web Servers without massive overprovisioning.
Section 5: Future Work

Currently our work is supporting only KML files. In the Future we plan to support Shape and other standard industry formats for upload. We also plan to explore adding more administrative support. Currently we have an administration tab which allows us to load files. We plan to integrate administration with the administration panel of AppEngine (which is extendable) so as to enable this platform to be used for “pay as you go” GIS. This would involve some way of metering the amount of data uploaded, data downloaded, page views and requests. AppEngine already provides APIs for the same. We also plan to fully exploit the JTS capabilities to provide more detailed reports. For the generation of such reports we plan to use the TaskQueues feature of AppEngine which allows us to run background tasks in parallel. We also plan to evaluate how easily this system can be ported to the VMForce Cloud platform when the same is released. VMForce is promised to be backward compatible with AppEngine if the Spring framework is used so we plan to port the existing system to the Spring framework in preparation for VMForce.

About the Authors:

Prabuddha Ghosh has a Bachelors in Computer Engineering from the Delhi College of Engineering and a Master’s in Computer Science from the University of Texas at Austin. He has extensive research in the fields of geospatial applications, telecom billing, data mining and hardware bench marking. His current research interests are - use of GIS in cloud infrastructure, mobile and social networking platforms.

Contact at +91-9910409701 ; RMSI A-7 Sector 16 Noida Uttar Pradesh India

Sandip Kumar Roy has Masters in Technology from IIT, Delhi. With more than 18 years of experience in the industry, he is currently responsible for managing multiple software Projects and delivery of Land Information Management Business Unit in RMSI. His research interest area is GIS solution involving Land Management, Location Based System and Disaster Management. Successfully implemented GIS based Systems for some of the key clients of RMSI across diverse geographies using leading mapping platforms.

Contact at +91-9811201202 ; RMSI A-7 Sector 16 Noida Uttar Pradesh, India
References:


Footnote1: Please note that all the Trademarked names of Software referred to in the paper are the property of the respective companies.

Footnote2: The Authors would like to thank RMSI for its support in undertaking this work.