Open Source Web Mapping: The Oregon Experience

By David Percy

Portland State University
Department of Geology
1721 SW Broadway
Portland, OR 97201
Telephone: (503) 725-3373
Fax: (503) 725-3325
e-mail: percyd@pdx.edu

SUMMARY

Open source tools have enabled a superior level of productivity for our online interactive map data delivery efforts. At Portland State University (PSU), we have delivered a web-accessible interactive map of the geology of Oregon since 1999. We also began delivering glacier data and coastal data in the ensuing years as it became obvious that all scientific data needed a web presence. Initial versions of our interactive web maps were based on MapObjects, followed by ArcIMS, both of which are products of ESRI, the dominant provider of proprietary GIS software. As more scientific data needed to be delivered via the Internet, the limitations of ESRI's software became evident, and the search began for better solutions.

Eventually, a new direction was established using Open Source software, which allows for a greater degree of customization and the transfer of existing skills in web design. Since PSU has an expert group of developers that use open source tools, such as PHP and MySQL, the training was minimized. Many of the skills that are already used for other web applications are directly transferable to web mapping when we use open source tools. Additionally, we have a culture of Open Source software use and development at PSU and, in general, within the state of Oregon, so this direction of development makes sense on many levels.

Overall, this effort has resulted in a web-mapping framework that provides considerably faster web page updates than its proprietary counterparts. Additionally, we have the ability to re-use components from applications, developed initially for certain organizations, to solve new problems for other organizations. Each time we initiate a project, we consider how it can benefit the larger goals of the web-mapping framework, which is essentially to provide feature for feature replacement of our proprietary competition, and thus everyone benefits.

BACKGROUND

Since Linux and Apache are supported on our campus as the defacto web platform, it is obvious that, to minimize the support required, we should use this platform. Mapserver was developed by researchers at the University of Minnesota under an NASA grant and is a mature server side application for delivering map data. PostGreSQL is an open source project that traces its roots to UC Berkeley, but is currently maintained in Germany. It is a hybrid relational-object oriented database, similar in functionality to Oracle. PostGIS, which was developed by Refractions Research (an open source consulting company in Victoria, BC), is a set of extensions that enhance PostGreSQL to give it a full set of GIS capabilities. PostGIS implements the full set of "OpenGIS Simple Features for SQL" capabilities as specified by the Open Geospatial Consortium. We refer to this mapping platform as LAMP for Linux/Apache/Mapserver/PostGIS.

METHODS

Once an organization has decided to use this set of open source mapping tools, they must decide how to deliver the data in an interactive web application. The previously enumerated tools provide the back end for web data delivery, but a front-end is needed to allow the end-user to interact with the data in a web browser.

Several mature web-mapping frameworks exist, but on close examination, it was clear that some had the patina of an older web application. That is, applications on the internet mature and age quickly, and new developments also happen quickly—the term "internet time" has currency because it is true that things happen rapidly on the internet. Thus, even though we could use one of the existing map-frameworks, that would not mean that it would be as functional as something developed with an eye to the future.

Initially, we considered using one of the existing web-mapping frameworks. There are several robust applications in existence such as Chameleon, Mapbuilder, and Ka-Map. After examining these mapping frameworks, which were already deployed, and in light of the previous notes and the ascendency of Web 2.0 and AJAX, we decided to develop our own. The functional requirements were to zoom/pan, query, turn off and on thematic layers, and dynamically resize the window to maximize map area. In a single weekend of development, several PSU graduate students wrote a new framework. One year later, the resulting product was named Map-Fu and became its own open-source project on Sourceforge in December, 2006 (http://sourceforge.net/projects/map-fu/).

In the meantime, several other mapping front-end products have become available that provide the same types of features we developed in Map-Fu. Thus, there are many options for the open source enthusiast to pursue. The main cautionary note is for the potential user to follow the listserve of any particular project for a few weeks to determine how active the community is. A healthy open source project will have several posts to the listserve every day, usually even 10 to 20. A project that has not had any posts to the listserve for more than a month is probably dead or perhaps mature, yet used only by one group.

Regardless of what front-end an organization chooses, the first step is to develop a mapfile Mapserver can read and generate images from. The mapfile will consist of names of data sources such as shapefiles for vector data and geotiffs for basemaps. It will also specify how to symbolize individual classes of data, for example a "Qal" unit would likely be displayed with an RGB value of 255 255 0.

In terms of optimizing data for web delivery, a few tasks are required. Large raster data sets need to be tiled and have internal overviews built. This is done via a series of command line operations that utilize an open source library known as GDAL. To build internal tiles we issue a command like this:

```
gdal_translate -of GTiff -co "TILED=YES" shaded_relief.tif shaded_relief_tiled.tif
```

After this, it is useful to build internal overviews (similar to "pyramids" in ArcMap) using a command like:

```
gdaladdo shaded_relief_tiled.tif 2 8 32 128
```

Note that the first command, gdal_translate, creates a new file, while the second command, gdaladdo, works "in situ" (without creating a new file). Also, the execution order of commands matters. Overviews are not copied during a gdal_translate operation, so the user should build tiles first, followed by overviews, as illustrated above.

To optimize the vector data, the shapefiles are imported to the open source database PostGIS, which is an extension of PostGreSQL. From here a command line function is executed that produces a lower resolution data set for initial delivery at low resolution ("zoomed-out") levels. It uses the conversion from postgresql to shapefile with the addition of an SQL operation. In this example, we have a table named lithology in the database named geology. We request that the output be a shapefile name simplelith and the sql command simplify the vertices down to one every 1000 feet:

```
pgsql2shp -f simplelith -h localhost -u mapserve geology -s "select simplify(the_geom, 1000) as the_geom, gnlith_u from lithology"
```

Techniques like this can considerably speed up the delivery of web-accessible data. While it may seem strange to do such things, it is simply the reality of providing data on the internet, where delivery times mean the difference between users accessing your site or simply abandoning it for lack of responsiveness.

With regard to returning query results, we have implemented an approach that uses the geospatial database PostGIS. When the user clicks on the map with the "info-query" tool, the coordinate pair they clicked on is sent to the database, which then returns all objects from all tables that intersect the point that was clicked. We then have a query handler that outputs data related to the objects, depending on which layers in the view are on or off.

It is relatively trivial at this point to set up an Open Standards based output system. Note that Open Standards are different from Open Source; in the first case we are talking about a committee of vendors and organizations that decide upon a protocol for data interoperability, while in the latter we refer to a formal system by which users are allowed to view and legally modify and redistribute the source code of programs.

The Open Standards protocols that are of interest are Web Map Services (WMS) and Web Feature Services (WFS), though the suite of open standards-based web services are collectively called Open Web Services (OWS). The entire set of OWS is still under development, though it is maturing rapidly and there are some viable uses now. By inserting certain metadata statements into the same mapfiles we use for our interactive maps, we can simultaneously serve as OWS providers. This allows our data to be aggregated by others into other useful web-interfaces.

CONCLUSION

We have had great success and satisfaction using open source tools for our web delivery of scientific data. Using open source tools has given us the ability to leverage existing strengths, as opposed to having to learn techniques that only apply to one monolithic proprietary software program. We also have the ability to use data from other OWS providers, such as NASA or the USGS, as base layers on which to overlay our data. In the end, we assume that these “stove pipe” solutions we are building will be converted into pure open standards based formats like WMS and WFS, such that any standards-based interface can integrate our data with whatever other data they deem useful to addressing the situation at hand.