Integrating AJAX Approach into GIS Visualization Web Services

Ahmet Sayar[1, 2, *], Galip Aydin[1, 2], Marlon Pierce[1] and Geoffrey Fox [1, 2, 3, 4]
[1] *Community Grids Lab, Indiana University, Bloomington, Indiana, 47404, USA*
[2]*Department of Computer Science, Indiana University*
[3]*Department of Physics, Indiana University*
[4]*School of Informatics, Indiana University*
{asayar, mpierce, gcf}@cs.indiana.edu

Abstract

As the Web platform continues to mature, we see an increasing number of amazing technologies that take Geographic Information Systems (GIS) visualization applications to new levels of power and usability. By integrating new powerful technologies into GIS systems, we get higher performance results with additional functionalities. The most recent development capturing the attention of the browser based application developers is AJAX (Asynchronous JavaScript and XML). In this paper we present a generic and performance efficient framework for integrating AJAX models into the browser based GIS Visualization Web Services systems.

1. Introduction

AJAX [3] is an important web development model for the browser based web applications. It uses several technologies which come together and incorporate to create a powerful new model. Technologies forming AJAX model such as XML JavaScript, HTTP and XHTML are widely-used and well-known. High performance Google mapping uses this new powerful browser based application model.

Web Services [2] are self-contained, self-describing, and modular. Unlike earlier, more tightly coupled distributed object approaches such as Common Objects Request Brokers (CORBA), Web Service systems support an XML message-centric approach, allowing us to build loosely coupled, highly distributed systems that span organizations. Web Services also generalize many of the desirable characteristics of GIS systems, such as standards for providing general purpose specifications for publishing, locating, and invoking services across the Web. Web Services also use widely-used and well-known technologies such as XML and HTTP as AJAX does. Since AJAX and Web Services are XML based structures they are able to leverage each others strength.

In this paper, we first give some background information about the web technologies we have been using in our proposed architecture. These are basically AJAX, Web Services, and GIS Web Services. In Section 3 we mention some related works about the AJAX and Web Services. In Section 4 we first give a generic architecture for integration of any Web Services and AJAX. Then, we give sample usage scenarios to prove our integration concepts; one of them is for Google and GIS Data Server (WFS) integration and the other one is for Google and GIS Mapping Server (WMS) integration. Section 5 is about the future work and Section 6 is the conclusion.

2. Background

Integration architecture was formed by using AJAX and Web Services in the GIS domain, here we explain their technical and theoretical structures, and advantages.

2.1. Asynchronous JavaScript and XML

AJAX is a style of web application development that uses a mix of modern web technologies to provide a more interactive user experience. AJAX [3] is not a technology. It is an approach to web applications that includes a couple of technologies. These are JavaScript, HTML, Cascading Style Sheets (CSS), Document Object Model (DOM), XML and XSLT, and XMLHttpRequest as messaging protocol.

These core technologies forming AJAX are mature, well-known and used in web applications widely. AJAX became so popular because it has a couple of

advantages for the browser based web applications developers. It eliminates the stop-start nature of interactions, user interactions with the server happen asynchronously, data can be manipulated without having to render the entire page again and again in the web browser, and requests and responses over the XMLHttpRequest protocol are structured XML documents. This enables developers easily integrate AJAX applications into Web Services.

After Google started to develop some new applications with the AJAX, AJAX drew some attention in the public. Some of the major products Google has introduced over the last year by using AJAX model are Google Groups, Google Suggests, and Google Maps. Besides the Google products Amazon also have used AJAX approach in its search engine application.

Client can use AJAX in her web applications just by writing her own custom JavaScript codes that directly use the XMLHttpRequest protocol's API. At that time, client should be careful of the coding and implementation differences between different web browsers. Instead of using pure AJAX and dealing with the browser differences, client can use some newly developed libraries providing higher level AJAX services and hide the differences between browsers. Among these are DWR, Prototype, Sajax, and AJAX.NET.

2.2. OGC GIS Web Services

The Open Geospatial Consortium (OGC) [1] defines a number of standards, both for data models and for online services, that has been widely adopted in the GIS community. OGC is a non-profit, international standards organization that is leading the development of standards for geographic data related operations and services. OGC has variety of contributors from different areas such as private industry and academia to create open and extensible software application programming interfaces for GIS [16].

GIS introduce methods and environments to visualize, manipulate, and analyze geospatial data. The nature of the geographical applications requires seamless integration and sharing of spatial data from a variety of providers. To solve the interoperability problems, the OGC has introduced standards by publishing specifications for the GIS services.

The emergence of Web Service technique overcomes the shortcomings of traditional Distributed Object technique and provides the interoperable capability of cross-platform and cross-language in distributed net environments. GIS services will be implemented more extensively by using the Web Service approach. A spatial data infrastructure lets many GIS vendors share data stores and applications in a distributed environment. GIS basically involves the integration of data and services from multiple sources from different vendors. The Web services architecture establishes a standard interconnection rules between services and information clients that nicely support the dynamic integration of data, which is the key to creating a spatial data infrastructure. By introducing Web Services, distributed GIS services from different vendors can be dynamically integrated into the GIS applications using the interoperable standard communication protocols of the Web Services [17].

Porting OGC services to Web Services will offer several key benefits, including:

Distribution: It will be easier to distribute geospatial data and applications across platforms, operating systems, computer languages, etc. They are platform and language neutral.

Integration: It will be easier for application developers to integrate geospatial functionality and data into their custom applications. It is easy to create client stubs from WSDL files and invoke the services.

Infrastructure: We can take advantage of the huge amount of infrastructure that is being built to enable the Web Services architecture – including development tools, application servers, messaging protocols, security infrastructure, workflow definitions, etc [5].

The most commonly used and well-known visualization related OGC GIS services are GIS Mapping Services and GIS data services. OGC call mapping services as Web Map Services (WMS) and data services as Web Feature Services (WFS) and Web Coverage Services (WCS) [13]. WFS provides feature data in vector format encoded in Geographic Markup Language (GML) and WCS provides coverage data in raster format.

3. Related Work

There are some well-known projects and efforts on the technologies participating in the proposed integration framework, such as Web Services and AJAX. OGC and ESRI use Web Services in GIS. OGC is actually a consortium and a standards body defining and publishing standards for the GIS services interfaces. Cubewerx, Demis and Intergraph are commercial GIS

companies involved the Web Services technologies into their systems. Google Maps and Ka-Map integrated the AJAX model into GIS visualization systems. Ka-Map is AJAX-based web mapping sites using an open source web mapping toolkit. ka-Map uses the MapServer [9] mapping server behind the scenes with AJAX and PHP to serve up the map content. In all these efforts mentioned above, Web Services and AJAX are used separately.

ECMAScript [17] for XML E4X is the only related work involving AJAX and Web Services together. E4X is a simple extension to JavaScript that makes XML scripting very simple. It is actually the official name for JavaScript. The European Computer Manufacturers Association (ECMA) is the standards body where JavaScript is standardized E4X uses all other incorporated AJAX technologies without extension.

Via the E4X, you don't have to use XML APIs such as DOM or SAX; XML documents become one of the native types that JavaScript understands. You can update XML documents from the JavaScript very easily. These properties of E4X enable creating calls to Web Services from the browser, but the only browser that supports E4X so far is the developer release of Mozilla 1.8.

E4X helps to interact with Web Services but again it is just an extended version of JavaScript. Some issues regarding how to put request in SOAP messages and how to manipulate returned SOAP messages are still complicated. If you use E4X for a web applications based on AJAX model, you can not use the application on every browser. This is another drawback of the system.

In our approach, you don't have to extend any technology involved in the AJAX model. We use all the technologies in AJAX with their original forms. This gives the developers and users the ability to integrate and customize their applications easily.

4. Architecture: Invoking Web Services in the AJAX Model

In this Section, we describe and illustrate the generic integration framework for integrating AJAX into browser based Web Service applications. There are two main actors in the integration architecture – the visualization client and GIS Web Services. Web Services are invoked by using Simple Object Access Protocol (SOAP) - XML based messaging protocol for the message exchange.

4.1. Generic Integration

How to invoke Web Services in the AJAX model?

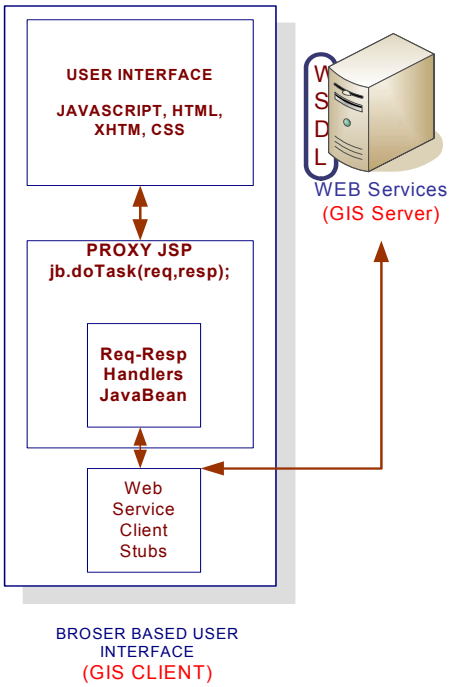

Figure 1: Invoking Web Services from the AJAX applications.

The client browser makes a request to the server broker (via a JSP page), which in turn makes a request to the Web Service by using previously prepared Web Service client stubs. The response from the Web Service is then transformed by the service broker, and presented to the client browser. Below we will go in more detail to explain all these steps.

First create an XMLHttpRequest object to make a remote scripting call.

- var http = new XMLHttpRequest();

Then, define the end point as an URL to make a call. The URL address should be local. This an intermediary proxy service to make appropriate requests for the GIS Web Service.

- var url = "proxy.jsp";

Then, make a call to the local proxy service end point defined above by the user given parameters.

- http.open ("GET", url + "?bbox = " + bbox +…[other parameter-value pairs]……)

proxy.jsp is an intermediary server page to capture request (HttpServletRequest) and response (HttpServletResponse) objects. Proxy JSP includes just one line of codes to forward the HttpServletRequest and HttpServletResponse parameters coming from the first page via XMLHttpRequest protocol.

- jb.doTask(request,response)

"request" and "response" parameters come from the user interface page. This first page includes some JavaScript, XHTML, CSS and JSP to capture the user given parameters and to display the returned result on the screen.

"jb" is a java class object which handles creating appropriate requests by using its request-response handlers and Web Service client stubs. Request-response handler also handles receiving and parsing response object coming from GIS Web Services interacted with.

After having received response from the GIS Web Service, "jb" object sends the returned result to XMLHttpRequest object in the first page.

- PrintWriter pw = response.getWriter();
- pw.write(response);

XMLHttpRequest object at the user interface page captures this value by making a call as below

- http.onreadystatechange = handleHttpResponse

This generic integration architecture can be applied to any kind of Web services. Since return types of each Web services are different and they provide different service API, you need to handle application specific implementations and requirements in browser based client side.

In Section 4.2, we proof the applicability and efficiency of the proposed integration framework by giving two important usage scenarios in GIS domain.

4.2. Usage Scenarios –Integrating Google Maps with GIS Visualization Systems

Integration is basically coupling AJAX actions with the Web Services invocations, and synchronizing the actions and returned objects from the point of end users. The usage scenarios in Section 4.2.1 and 4.2.2 use the generic integration architecture illustrated in Figure 1. In the usage scenarios there will be minor difference in the form of extensions. Differences come from the service specific requests created according to the service provider's service API (published as WSDL), or handling returned data to display on the screen. But these are all implementation differences.

4.2.1. Google's AJAX Integration with WMS: There are two different path working in parallel by the given user parameters created by the client actions. Actions are interpreted by the browser through the Google Mapping tools. JavaScript captures these actions by ActionListeners and Google Binding APIs and gives to Layer-2 object. Please see the Figure 2.

On the browser user interface class is a JSP page. It includes two JavaScript class-references. One is for the Google Map object and the other is for the WMS map image and bindings to the Google Map object.

Interconnection for creating Layer-2 is done in accordance with the proposed architecture defined above in Figure 1. For Layer-1, a classic Google mapping application is used through the AJAX web application module and XMLHttpRequest protocol. Google handles creating the map by using XMLHttpRequest and given remote JavaScript file in the browser [4].

When we use this type of interaction interface to WMS, we can utilize all the OGC compatible functionalities of the WMS such as "getMap", "getCapabilities" and "getFeatureInfo". The client is going to be a thin client; it just takes the map and overlays it over the Google map. Overlay is done by using some advanced JavaScript techniques. The client does not need to make rendering or mapping jobs to create the map image. The map is already returned by the WMS and in a ready to use format such as JPEG or PNG or TIFF. Return type is defined as a parameter in the "getMap" request given to WMS. These images in different formats are converted to a JavaScript object before overlaying.

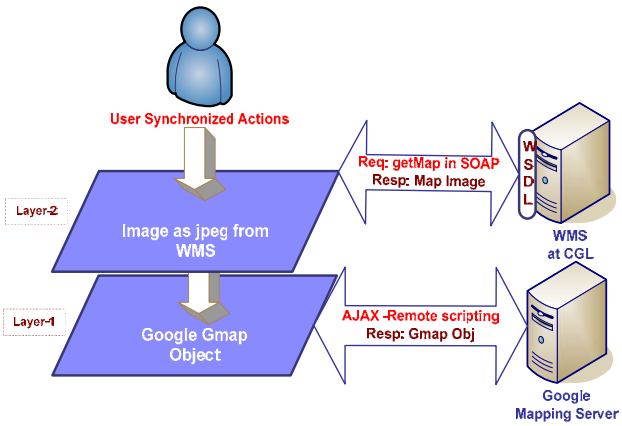

Figure 2: Integration of Google Maps with OGC WMS by using architecture defined in Figure 1.

4.2.2. Google's AJAX Integration with WFS: WFS provides feature data in vector format and vector data are encoded in GML according to OGC WFS specifications and depending on the parameters given in the "getFeature" request. GML is an XML encoding for the transport and storage of geographic information, including both the geometry and properties of geographic features.

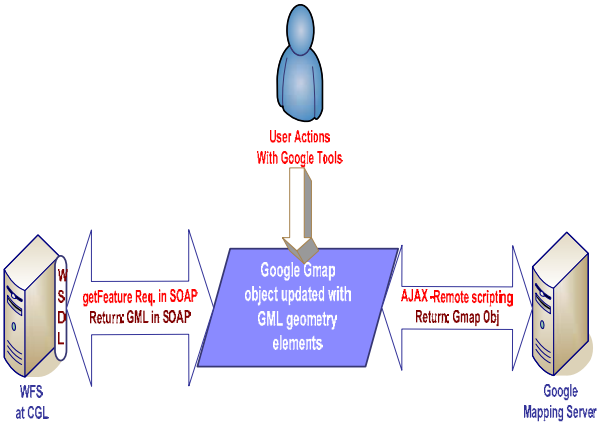

Figure 3: Integration of Google Maps with OGC WFS by using architecture defined in Figure 1.

In response to the "getFeature" request, the GML file encoded in XML is returned in a SOAP envelope as a response to this request. After getting a response, the client extracts geometry elements. The most important and commonly used geometry elements are Points, LineStrings, LinearRings, and Polygons. GMLis an OGC standard for feature data representation.

Even though Google Mapping API supports just two of them, Points and LineStrings, the other geometry elements can also be converted to these two types with minor updates. Having extracted and obtained geometry elements, these elements are plotted over the Google Map by using "GPoints" and "GPolylines" objects and the "mapOverlay" function of the Google Map API.

By setting returned GML's non-geometry elements and using 'GMarker" object of the Google API, this architecture also provides the "getFeatureInfo" functionalities of the OGC WMS services. All these tasks are achieved by using XMLHttpRequest API and JavaScript functionalities.

XMLHttpRequest uses DOM for parsing returned structured responses in XML. If returned data is oversized for the server then the DOM parser throws "Out of Memory" exception. In order to overcome this drawback of the DOM and Google Map, we have used Pull Parsing via. After parsing and handling GML documents returned from WFS, the result is written into the web browsers response object. Through the responseXML call of the XMLHttpRequest in JavaScript, the browser gets the result and makes appropriate modifications to the data and display on the screen.

5. Future Work

In the future, we will be working on the performance issues of the architecture. Google Map provide the map data in an efficient time, but WMS and WFS Web Services returns the data in a much longer time. This is because of the characteristics and sizes of the geographical data [12] and some CPU and time consuming rendering algorithms to produce the map images. Since we do not have high performance servers and private networks we need to consider improving the performance in different ways. For that purpose, we will be using streaming version of the WMS and WFS to get geographic data in the form of image and GML correspondingly by using message middleware systems such as NaradaBrokering [6,7], developed in CGL (Community Grids Lab.) at Indiana University. NaradaBrokering provides some features that are important in the GIS area. These are Quality of Service (QoS) and security profiles for sent and received messages, interface with reliable storage for persistent events, reliable delivery via WS-Reliable messaging, fault tolerant data transport, support for different underlying transport implementations such as TCP, UDP, Multicast, SSL, RTP, HTTP, discovery

service to find nearest brokers / resources (efficient routing). For our ongoing work, please have a look at our official project page, crisisgrid [8].

6. Conclusion

If the GIS visualization client uses Web Services from the desktop browser application and Web Services are capable of responding fast enough, then using the AJAX model for calling Web Services gives high performance increases. Since both AJAX and Web Services use XML based protocols for the request and responses, they leverage their advantages. This enables application developers to easily integrate AJAX based browser applications into Web Services.

Using just Google Maps has some disadvantages in extracting and displaying the information about the specific feature selected by clicking on the map. By using GIS Web Services in the same application and assigning this part to WMS, we eliminated the Google Maps' drawbacks and made it much faster. Since Google Map API uses DOM parsing, if the data size is oversized for the server it is impossible to parse and get feature information from the large geographic data set represented in structured XML data such as GML. By integrating GIS Web Services into the visualization application and using Pull Parsing techniques we can easily eliminate this drawback.

In our proposed architecture design implementation, we have not modified or extended any technologies in the AJAX model or Web Services. By using the same theoretical standards, you can integrate any GIS Web Service into your visualization applications just by doing some application specific extensions such as creating requests according to the service API of the Web Service and handling the returned objects.

7. Acknowledgements

This work is supported by the Advanced Information Systems Technology Program of NASA's Earth-Sun System Technology Office and the National Science Foundation's National Middleware initiative.

8. References

[1] OGC (Open Geospatial Consortium) official web site http://www.opengeospatial.org/

[2] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. "Web Service Architecture." W3C Working Group Note, 11 February 2004. Available from http://www.w3c.org/TR/ws-arch

[3] Jesse James Garret, Ajax: A New Approach to Web Applications. http://www.adaptivepath.com/publications/essays/archives/000385.php

[4] Murray G., "Asynchronous JavaScript Technology and XML (AJAX) With Java 2 Platform, Enterprise Edition" http://java.sun.com/developer/technicalArticles/J2EE/AJAX/

[5] Jerome Sonnet, Charles Savage. OGC Web Service Soap Experiment Report 0.8 Document#03-014, Jan 2003.

[6] Message based middleware project at Community Grids Lab, Project Web Site: http://www.naradabrokering.org/

[7] Pallickara S. and Fox G., "NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids" ACM/IFIP/USENIX International Middleware Conference Middleware-2003, Rio Janeiro, Brazil June 2003

[8] GIS Research at Community Grids Lab, Project Web Site: http://www.crisisgrid.org

[9] MapServer official web site http://ms.gis.umn.edu/.

[10] Ka-Map official web site http://ka-map.maptools.org/.

[11] Tyler Mitchell, "Build AJAX-Based Web Maps Using ka-Map" http://www.xml.com/pub/a/2005/08/10/ka-map.html.

[12] de La Beaujardière, J. editor, 2002. Web Map Service Implementation Specification, Version 1.1.1, OGC 01-068r3. http://www.opengis.org/techno/specs/01-068r3.pdf

[13] Evans, J. eds, 2003. Web Coverage Service Implementation Specification, OpenGIS® Project Document OGC 03-065r6, http://www.opengis.org/docs/03-065r6.pdf

[14] ISO, 2001. ISO 19119: Geographic Information – Services. http://www.isotc211.org.

[15] Vretanos, P. A. editor, 2002. Web Feature Service Implementation Specification, Version 1.0.0 OGC 02-058. http://www.opengis.org/techno/specs/02-058.pdf.

[16] Sayar A., Pierce M., Fox G. "OGC Compatible Geographical Information Services", Technical Report (Mar 2005), Indiana Computer Science Report TR610.

[17] Sayar A., Pierce M., Fox G., "Developing GIS Visualization Web Services for Geophysical Applications" ISPRS 2005 Spatial Data Mining Workshop, Ankara, Turkey.

[18] EcmaScript web site http://www.ecma-international.org/