Choosing the right GIS framework for an informed Enterprise Web GIS Solution

Written By Sneha Rao (srao@ciesin.columbia.edu) and

Sri Vinay (sri@ciesin.columbia.edu)

CIESIN, Columbia University & NASA

New York, USA

December 17, 2009

Abstract:

An Enterprise Web GIS Solution combines the knowledge of complex GIS systems with the standards and best practices of Information Technology to design and implement an end-to-end system that deliver geospatial data services, tools and applications on the web. In order to design an optimal solution that fits well with an enterprise workflow and provide robust, reliable, responsive and scalable map services and applications, it is important to understand the various components of the web GIS framework and consider the key factors that affect them when deciding on the type of technology stack that works best.

This paper reviews the important factors involved in choosing the right framework for an Enterprise Web GIS Solution and the various architectural components of a web GIS system. It begins with an overview of the multi-tier dynamic Client-server system architecture and implementation of specific configurations that web GIS systems apply in performing geospatial data analysis. Next, it introduces the web GIS System Architecture setup at Columbia University's Center for International Earth Science Information Network (CIESIN) as a case study. Using the knowledge gained from our implementation experiences, it then evaluates and compares the system components based on factors such as workflow, performance, backward compatibility, scalability, interoperability and total cost of ownership (TCO). Finally, this paper concludes on the note that these are the important factors to consider while designing an Enterprise Web GIS Solution. However, it is also important to note that every enterprise has its unique situation, business needs and strategic interests and the importance given to these factors may vary accordingly.

Introduction:

GIS Desktop tools and applications have enabled users to view and analyze spatial data in its proper format. With the evolution of GIS, the sophistication of these analytical tools has increased tremendously, thereby increasing the cost and time required to understand and use these tools efficiently. Additionally, most end-users today are interested in the interpretation of the final results in its final format or through visual representation.

Web GIS provides GIS users easy access to geographic information data, spatial information and GIS modeling and processing tools. It provides an open and distributed architecture for disseminating geospatial data and web processing tools on the internet. This makes it easier for larger organizations to distribute maps and tools without time and cost restrictions to the end user. To provide a successful Enterprise Web GIS Solution, it is required to understand the complexity of the implementation as a process rather than a step [Alesheikh, Helali & Behroz, 2002].

Understanding the complexity of GIS:
Geographic Information Service is a very complicated information service that requires a different solution than other types of information services. The contents of geographic information vary in different scales, resolutions, domains and times. Another complex feature of geographic information is the ability to overlay different layers of spatial data to generate new layers of information.

Development and Operational Issues:
From the development perspective, most current web GIS systems adopt a quick, ad-hoc technology centered solution for open and distributed GIS that are neither sustainable nor scalable. Once the technology changes, all ad-hoc solutions of the old system are abandoned and are redone for the new system [Peng & Tsou, 2003]. Another problem is that current implementations focus more on data and less on processing tools.

From the operational perspective, it is very difficult to separate the various web GIS components into simple, modular and independent parts. Ideally, the interactions between these components should be extensively defined with the clear definition of their relationships [Peng & Tsou, 2003].

Web GIS architecture:

Basic components:
*Client***:** Typically, the client or the user side of the web refers to the web browser on the user's machine. In the web GIS world it refers to a place where users interact with spatial data and analysis tools. It is also a place where GIS programs display different forms of output to the user based on commands, tools and tasks that are triggered by some client-side and server-side actions and may have some business logic associated with it.

Server: The web GIS server architecture generally has four components: web server, application server, map server and data server.

Web Server: The web server responds to requests sent from the web browser via HTTP. Any web site that is published on the Internet must have a web server program running in the background: e.g., Apache, a web server that is supported on multiple platforms such as UNIX, LINUX, and Windows. The web server can act as a proxy strengthening the system security and balancing the load between application servers in a clustered environment.

Application Server: Application server is a software that helps develop, deploy and manage large number of applications in a distributed environment. An application server acts as a middleware that establishes, maintains and terminates the connection between the web server and the map server. It also manages the concurrent requests and balances the load amongst map servers. From the developer's perspective, the main objective of an application server is to separate business logic from presentation logic and database logic.

Map Server: A map server is the brain of any web GIS application. It provides specific traditional GIS functions that include: spatial analysis, spatial and attribute queries, geo-coding, geoprocessing and generates and delivers dynamic maps to the client based on user requests. The map server can generate output in two broad forms: 1) Feature info as a result of a spatial query, such as feature selection or geoprocessing, that are sent to the client application for further user manipulation or 2) A graphical image as a response to feature filtering/search capabilities or a simple GetMap request etc.

Data Server: A data server manages data, spatial or non-spatial, in a relational or non-relational database management system. A client application gains access to the database through SQL statements.

Multi-Tier Architecture:

A generic multi-tier architecture illustrating the Client-server components of web GIS is illustrated below:

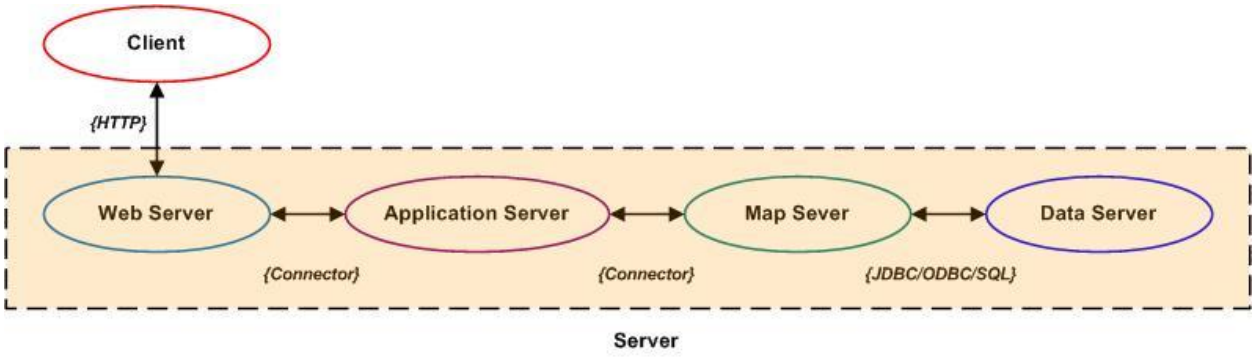


Figure 1. *Multi-tier Client-server web GIS Architecture*

A multi-tier architecture follows a Client-server architecture in which the presentation, application processing and data access are logically separate processes. In web GIS, the client or the application sends an HTTP request to the web server. The web server forwards the request to the application server. The application server then responds to the request by forwarding it to the appropriate map server and manages the load amongst the map servers. The Map server then further synthesizes the request and performs the appropriate GIS function while retrieving the data from the data server.

Next we illustrate the architecture of some of the web GIS products, both commercial and open source, and relate them to the generic multi-tier architecture presented above.

ESRI ArcGIS Server Architecture:

The ESRI ArcGIS Server implements a three-tier architecture as shown below. The web server component of the ArcGIS Server architecture provides the functionality of the application server described in the generic multi-tier web GIS architecture. The web server hosts the web services and applications that are developed. It receives requests from clients and relays appropriate tasks to the GIS server.

The GIS server in this architecture is equivalent to the map server in the generic architecture. The GIS server hosts GIS resources such as maps, globes, and address locators, and exposes them as services to client applications. The GIS server itself is composed of two distinct parts: the server object manager (SOM) and server object containers (SOCs). As the name implies, the SOM manages the services running on the server. When a client application requests the use of a particular service, it's the SOM that actually provides one for the client to use i.e. the SOM connects to one or more SOCs. The SOC machine hosts services that the SOM manages. Depending on your configuration, you can run the SOM and SOC on different machines and also have multiple SOC machines [ESRI 1, 2009].

The data server contains GIS resources that are published as services on the GIS server. These resources can be map documents, address locators, globe documents, geo-databases, and toolboxes. In an Enterprise web GIS system, the data server will usually include a combination of ArcSDE/Oracle database servers.

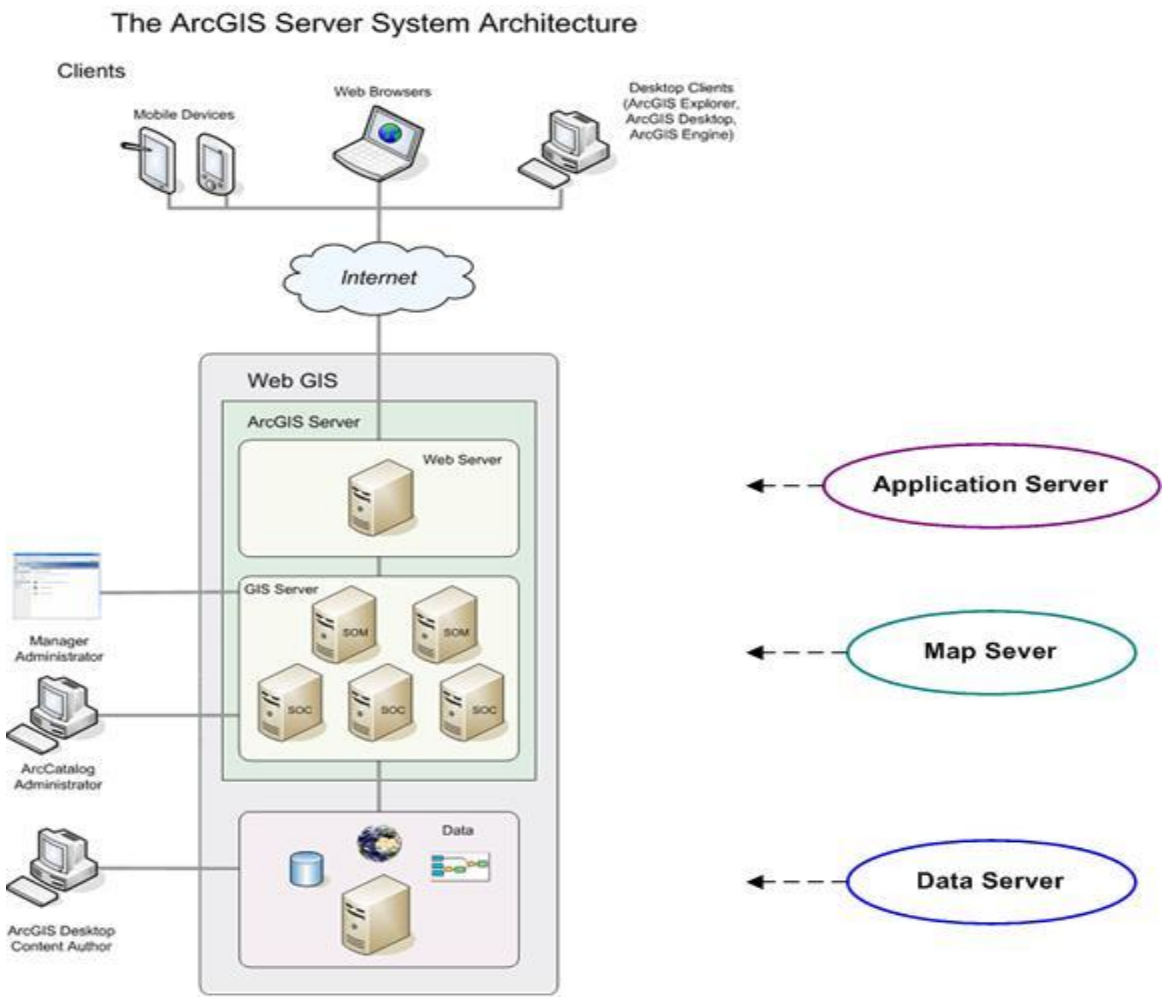


Figure 2. *ESRI ArcGIS Server Architecture*

The ArcGIS Server architecture implementation is very scalable and supports various functionalities needed in an end-to-end web GIS system such as geospatial data modeling, geoprocessing, analysis, web services development, and applications and tools development.

GeoServer Architecture:

GeoServer, a well known open source product, uses two-tier architecture as illustrated below. The application server and map server components are combined into one server application that consists of many different modules that are actively interacting with each other. All these modules are connected using Spring IOC so that at runtime a module can make use of services provided by other classes.

GeoServer supports various data servers such as PostGIS, Oracle Spatial, and ArcSDE and file formats including shapefiles and GeoTiff and produces output as KML, GML, shapefiles, GeoRSS, GeoJSOn etc. GeoServer follows the OGC specs i.e. WMS, WFS and WCS specifications to the letter, and forms a platform to develop GIS applications based on these specifications [GeoServer 1, 2008], [Ajay].

In an Enterprise web GIS system, the GeoServer can fit well as a module for serving various geospatial datasets via WMS, WFS, and WCS interfaces. However, to support more complex modeling, analysis, and web processing capabilities, other products supporting such functions are needed.

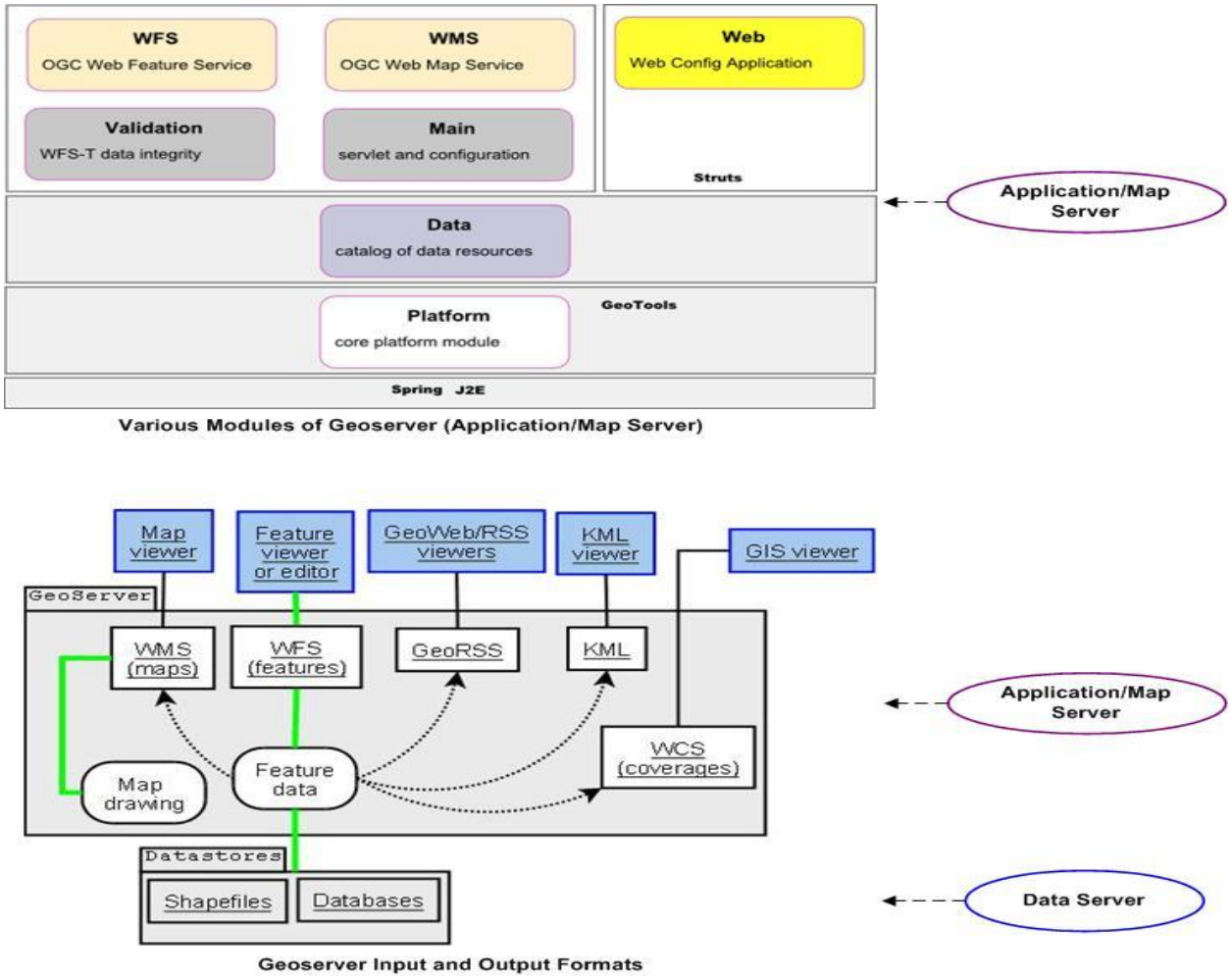


Figure 3. *GeoServer Architecture*

IONIC RedSpiderWeb Architecture:

IONIC RedSpiderWeb is a commercial web GIS software product and has been superseded by a suite of ERDAS Apollo products. The RedSpiderWeb uses a two-tier architecture illustrated below. Various application and map server modules and components are combined into the RedSpiderWeb Application/Map server. It also supports a variety of data servers including Oracle Spatial and ArcSDE as well as various file formats such as shapefiles, GeoTiff etc.

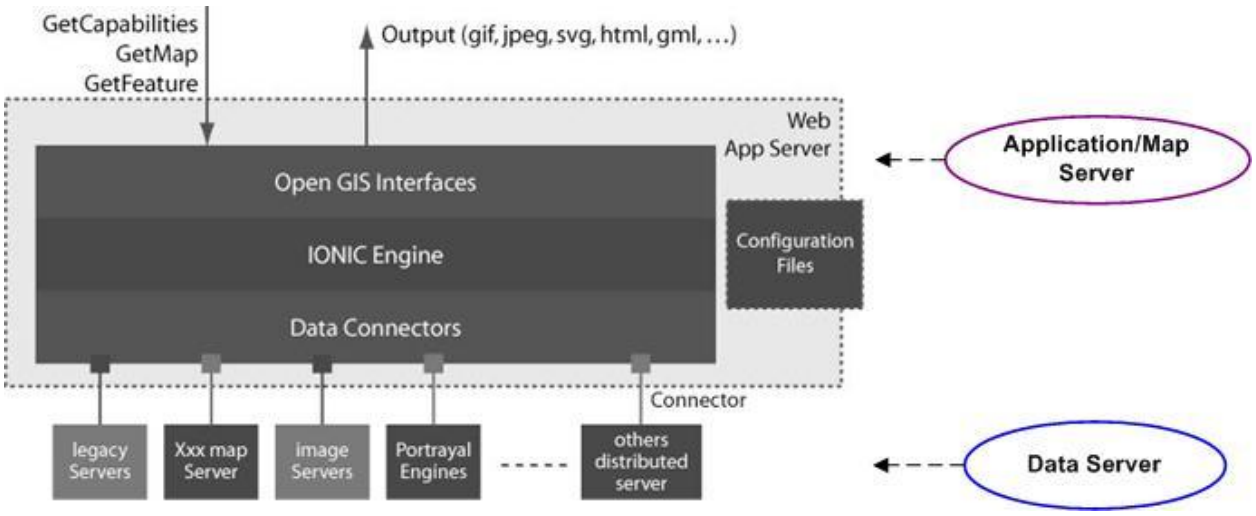


Figure 4. *IONIC RedSpiderWeb Architecture*

RedSpiderWeb supports the OGC specifications WMS, WFS, and WCS and can be used as a module for serving geospatial datasets through these interfaces. Other products are needed to support advanced geoprocessing functions such as access to models and web processing.

ERDAS Apollo Products:

The successor to the RedSpiderWeb, the ERDAS Apollo suite of products provides a unified enterprise platform solution for GIS, photogrammetry and remote sensing, extending geospatial data to business applications. ERDAS APOLLO implements an out-of-the box Service Oriented Architecture (SOA).

ERDAS APOLLO Server catalogs and delivers enterprise geospatial data over the web though a user-friendly interface. ERDAS APOLLO Server implements a Spatial Data Infrastructure with Open Geospatial Consortium (OGC) and the International Standardization Organization (ISO) compliant web services. ERDAS APOLLO Server is the core module of any APOLLO solution: self-sufficient for meeting common use cases, with the ability to be extended to fulfill the most sophisticated business workflows. The architecture of the ERDAS APOLLO, even though not illustrated here, seems to be well suited for an Enterprise web GIS system with complex workflows.

CIESIN implementation of a Multi-Tier web GIS Architecture:

Columbia University's Center for International Earth Science Information Network (CIESIN – http://www.ciesin.columbia.edu) has more than fifteen years of experience in the field of GIS, in particular web GIS. CIESIN has worked with various commercial (COTS) and open source GIS software products including suite of ESRI products (ArcInfo, ArcIMS, and ArcGIS Server), IONIC RedSpiderWeb, GeoServer, Google Maps, Google Earth, and related remote sensing software products such as ERDAS Imagine, and ENVI. During the past five years, there has been tremendous growth in

GIS technologies and several national agencies, international organizations, and private corporations and software companies are supporting various collaborative efforts, such as OGC, GSDI, and GEOSS, etc., to establish standards, architecture, pilot projects, and best practices in support of web GIS. CIESIN is leveraging its wealth of past GIS experience and its ongoing participation in these collaborative efforts to design, develop and operate a robust, responsive, and scalable Enterprise web GIS architecture. It is also making available, on the web, a rich collection of interdisciplinary data, services, applications, and tools related to human interactions in the environment.

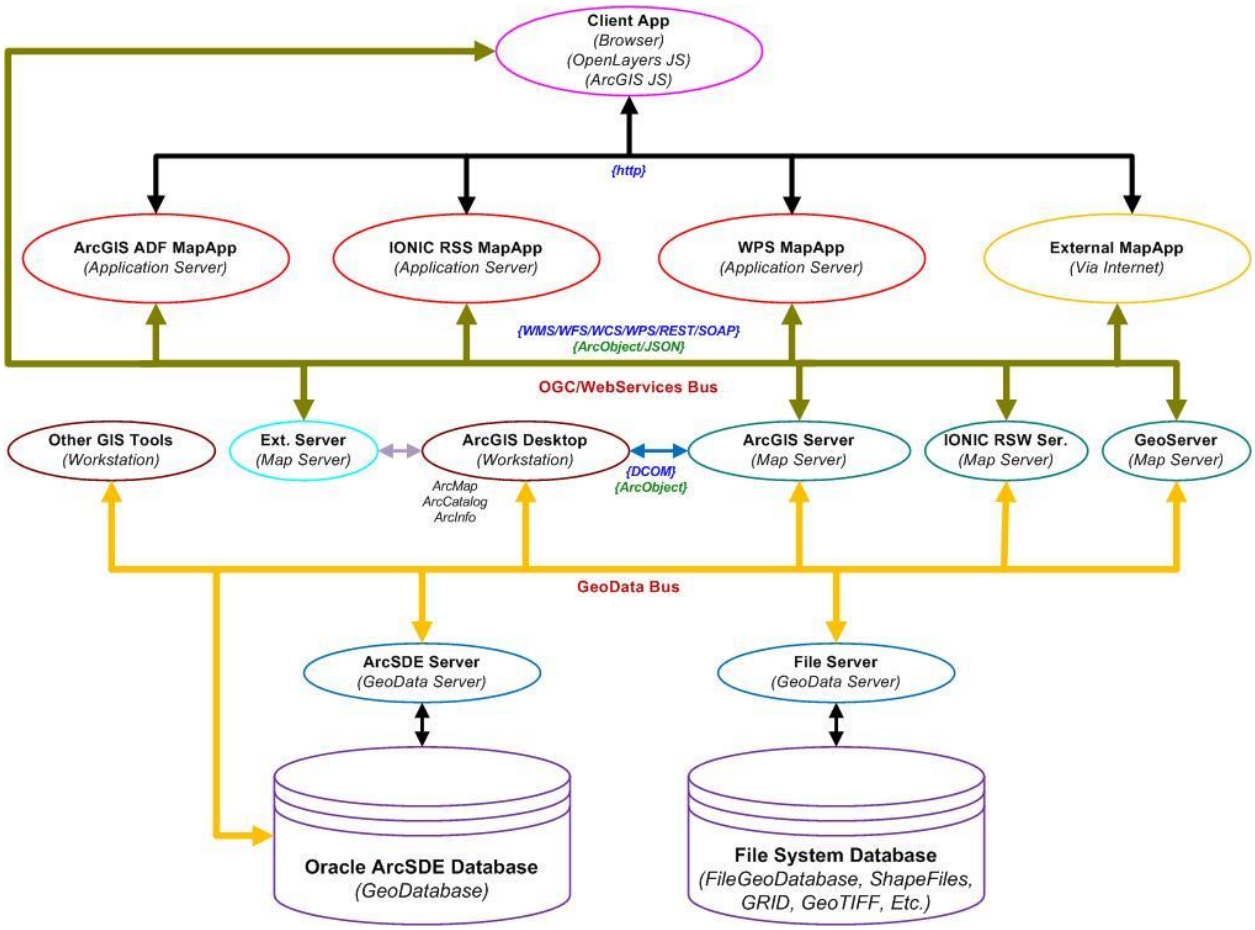


Figure 5. *Enterprise web GIS Architecture Implementation at CIESIN*

The design of this architecture takes into account our past experiences and lessons learnt. The main focus of the design is flexibility and the ability to adapt to the rapidly changing needs and demands of the enterprise, user requirements, and technology, while continuing to support existing and legacy GIS capabilities and services. At the same time, careful attention is paid to various factors, such as workflow, performance, reliability, total cost of ownership (TCO), backward compatibility to support existing and legacy capabilities, services etc., while selecting the system components. These factors are discussed in detail in the next section.

At the bottom tier of this architecture is the GeoData server (data server). Two main types of data stores are supported:

- An Oracle ArcSDE Geodatabase to which map servers and tools can either connect directly or through an ArcSDE server. From past experiences, we learnt that a file system was an inefficient platform when used in a multi-user editing environment for geo data development. For example, the development of several global scale geospatial datasets, such as Gridded Population of the World (GPW) [CIESIN, Columbia University & CIAT-1, 2005], which involved a large team of researchers, GIS analysts, and students proved to be very cumbersome and unproductive. With the introduction of Oracle ArcSDE Geodatabase, we hope to overcome these issues and streamline the geo data development activities for such complex projects. Another significant benefit is the ability to provide secure and controlled access to data editors from external data providers and collaborating organizations.

- File System Database supports a wide variety of data formats such as ESRI FileGeoDatabase, ESRI GRID, shapefiles, GeoTiff, etc. For smaller projects involving a fairly small team of data editors, it is still efficient to use such data stores. Migrating to a more complex environment such as Oracle ArcSDE is not likely to yield any significant dividend.

- We are also considering the addition of PostGIS/PostgreSQL data server to have an end-to-end open source solution, which are a preference and/or requirement of some of our projects.

Various map servers and data editing tools, such as ArcGIS Desktop, are in the middle tier on top of the data server and access the data stores to process, manipulate, publish, and serve data:

- We continue to use and support the ArcGIS Desktop tools for data processing and editing due to their rich functionality and lack of comparable alternate solutions and the tools fit well within an enterprise workflow and suit the skill sets of the workforce. Besides, the ArcGIS Desktop tools can access data directly or indirectly through the Oracle/ArcSDE data server and through the ArcGIS map server and ArcCatalog. This provides data editors with a flexible and open environment to access, analyze, and integrate data from various sources including those within the enterprise and on the web.

Currently three different map server products are supported in our architecture:

- The IONIC RedSpiderWeb (RSW) map server supports several of our existing WMS, WFS, and WCS services. Although these services are robust, scalable, and responsive, we experienced several issues and limitations that hindered our workflow and productivity. For instance, preparing and publishing a dataset proved to be tedious and time consuming due to the lack of an out of the box tool to style layers for proper rendering on the client application. Also, the lack of an easy to use GUI tool to ingest data and configure the services for publishing

limited the number of GIS analysts to one or two people with XML editing skills. Due to these reasons, publishing services through this interface has become a difficult task. In addition, the enterprise required advanced geoprocessing capabilities and exposure of models and Web Processing Services (WPS) to client applications on the web. To overcome these limitations and support advanced web processing functionalities and new requirements, we decided to introduce two additional map servers.

- ArcGIS Server is ideally suited for our enterprise workflow. With ArcGIS Server, styling spatial data layers, publishing and administering advanced geoprocessing models, Web Processing Services, REST and SOAP web services, and OGC web services using the existing data stores is fairly straightforward. Web based GUI tools are provided for such purposes making it easy for a GIS analyst to fully participate in these tasks. Consequently, the productivity of the workforce increases and in turn the enterprise is able to meet its project deliverables and satisfy sponsor and user needs in a timely-manner. Furthermore, the enterprise gains the capacity to pursue new business opportunities and undertake new projects.

- The reasons for adding GeoServer into the architecture is mainly three-fold: a) it follows the WMS, WFS and WCS specifications to the letter, and forms a robust platform to develop GIS applications based on these specifications, b) it provides web based GUI tools for ingesting, styling, configuring, and publishing services using existing data stores managed by the suite of ArcGIS Desktop tools, and c) as an open source product it gives the enterprise the ability to provide a fully open source based web GIS solution, which sometimes is a preference or requirement of some sponsored projects.

The top tier of the server side architecture consists of Application servers hosting various mapping applications and tools that provide the client with access to geospatial data and services and enable users to visualize, query, analyze, process, and download data. We discuss below, the technologies being used in the enterprise, the reasons for their selection, and our experiences with them.

- The IONIC RedSpiderStudio (RSS) product has been used for developing thick map client applications (http://sedac.ciesin.columbia.edu/mapviewer, http://beta.sedac.ciesin.columbia.edu/daddi/mapviewer, http://beta.www.ciesin.columbia.edu/icap/mapviewer). These applications provide access to any WMS and WFS service on the web and provide good visualization with layer overlays, spatial feature querying, and attribute filtering capabilities. They also support the OGC Web Mapping Context (WMC) specification that enables users to save and share maps they dynamically created by integrating layers from various services. WMC also enables thematic book marking i.e. pre-grouping of layers into thematic categories, e.g. climate, sustainability etc. that are accessible via a permalink so that the application can be initialized by context dependent views. In addition, spatial book marking is also supported. Substantial amount of custom code was developed to support these functions and

appropriate user interface components as per project requirements. However, integration of these custom codes with the code base was difficult. Subsequently, compatibility with the newer versions of the product was another issue and required the custom code to be re-written in most cases thereby causing delays in upgrading and long-term maintenance of legacy products.

- The Java Server Faces (JSF) based ArcGIS application development framework (ADF) has been introduced recently into the application server tier for developing thick map client applications (http://nbii-nin.ciesin.columbia.edu/beacon). The application provides access to a variety of base layers and operational thematic layers for visualization with overlays and supports spatial and attribute querying and access to feature attribute data. It also supports scale dependent display of layers and features. The ADF is tightly coupled with the ArcGIS map server and enables very easy creation of mapping applications. However, from our experience so far, customization, performance improvements, and adding advanced functional features requires lot of custom code, which invariably introduces compatibility issues with future versions causing delays in upgrades. It is also difficult to understand the inner workings of pre-packaged frameworks like the ADF and hence developing and maintaining custom applications using these frameworks can sometimes be as resource consuming as building an application from scratch.

- We have also added open source application software 52 North to support OGC Web Processing Service (WPS). This application receives WPS requests from clients and translates them to SOAP web service requests and forwards them to the ArcGIS map server for analytical processing. A model deployed on the ArcGIS Server performs the requested geoprocessing and returns the results to the WPS application, which then returns the response to the client according to WPS specifications. The current WPS implementation accepts a set of polygon features, computes the relevant population count and other relevant statistics of those features using the Gridded Population of the World (GPW) [CIESIN, Columbia University & CIAT-1, 2005] dataset, and finally returns the computed feature information along with the features to the client.

In addition to these stand-alone thick clients with rich functionality, there is an increasing need in the enterprise to develop thin clients with simple and or medium level functionality to support simple visualization and base layer overlays, querying of feature attribute data and download. Such clients will often be embedded into project web sites to satisfy user needs and enrich usability. The server side frameworks, described above, are not suitable for developing such thin clients. Given these factors and some of the custom code and maintenance issues mentioned above, we have added the following client side application development solutions to our architecture.

- Open Layers API: Open Layers is an open source Java Script Library that provides easy access to OGC web services (WMS, WFS etc.). One of its attractive features is that it accepts code contribution from developers making it easy to incorporate custom features and code into the code base. However,

handling WFS services with large number of features can be a problem and, as with any maturing open source product, finding documentation and resources for development and troubleshooting may not be easy. CIESIN has recently released a simple map client application (http://sedac.ciesin.columbia.edu/wildareas) and is planning to release several others soon by building our own JS library on top of Open Layers.

- The ArcGIS API for JavaScript™ (JavaScript API) is a browser based API for developing high performance, easy to use mapping applications. The API allows you to easily embed maps in your web page. It also provides easy access to map services on an ArcGIS map server via REST calls. In addition to basic visualization and overlay functions several other features such as spatial feature attribute querying etc. are also supported. CIESIN has developed and released couple of map applications based on this technology; http://www.nbii.gov/portal/community/Communities/Geographic_Perspectives/Northeast/Maps_and_Geospatial_Data/ (Featured Map Tool), and http://www.nbii.gov/portal/community/Communities/Geographic_Perspectives/Northeast/Maps_and_Geospatial_Data/Hudson_Watershed_Stream_Gage_Mapper

- Google Map API: The Google Maps API is a free service, available for use on any website that is free to consumers. It enables developers to embed Google Maps in web pages and includes a number of services for customizing and adding content to the map. OGC services can also be accessed via Google Maps API with some custom coding. CIESIN has developed web pages embedded with maps using Google Maps API; one of the most recent applications is a map application demonstrating the population estimation Web Processing Service (WPS) available at http://beta.sedac.ciesin.columbia.edu/gpw/wps.jsp.

Finally, access to all application and map servers are provided via proxy web servers that provide system security and load balancing. They are not shown in the architecture diagram in Figure 5 for the sake of simplicity and readability.

Factors affecting System Components:

In multi-tier web GIS architecture, there are several factors that contribute to the performance and responsiveness of the overall system. These factors are discussed below. We have considered these factors carefully in the selection of the hardware and software for our web GIS architecture.

Workflow:

It is very important that the software products chosen integrate well within the enterprise workflow and enable staff to work with them seamlessly. Otherwise, it will have an adverse impact on the efficiency of enterprise operations, productivity, and project deliverables. This is one of the reasons to add ArcGIS Server and GeoServer into our map server mix.

One other aspect of workflow we considered is the ability to support data management and editing in a multi user environment where large team of data analysts and editors work on the same data development project. ArcGIS Server and desktop tools with Oracle/ArcSDE data server supports this feature.

The ESRI framework is very tightly integrated and provides end to end solutions in GIS. Hence it provides ease in development starting from data preparation and processing using Arc Desktop tools, followed by publishing services through ArcGIS Server Manager Interface or ArcCatalog. Finally it provides several client development frameworks such as Web ADF, ArcGIS API for JavaScript, etc. Updating or editing existing data can be tricky; depending on the scale of the change the server may or may not pick up the changes. In these cases the server is forced to refresh or restart.

The GeoServer is a JAVA based tool specifically created for web service publication. It expects data in a particular format and then provides a platform to serve the data on the Internet. Updating existing services, applications, shapefiles and databases is simple and effective.

Server Performance:

The overall system response is most influenced by the performance and speed of the various servers involved. Hence, monitoring and performance tuning of servers is very critical. There are a variety of tools available, in various platforms, for load testing and monitoring servers. We illustrate below, some of the work being carried out at CIESIN.

We used the Apache JMeter tool for load testing i.e. to measure the response time on both map applications and map services. For each load test, JMeter was configured to simulate 10 concurrent user sessions and 10 sequential requests within each user session. In cases where the response was slow, server resources such as memory use, CPU use, and average system load were monitored. Since the application and map servers are Java based the JVM heap memory usage were monitored using the jvisualvm tool and the CPU usage and average system load were monitored using system tools available on the servers.

Mapping Applications: Thick Client vs. Thin Client

The thick application (http://nbii-nin.ciesin.columbia.edu/beacon) uses the ArcGIS ADF framework and SOAP interface whereas the thin application (http://www.nbii.gov/portal/community/Communities/Geographic_Perspectives/Northeast/Maps_and_Geospatial_Data/Hudson_Watershed_Stream_Gage_Mapper/) uses the

ArcGIS JS library and REST interface. They both utilize the same mapping service. The JMeter results are presented below.

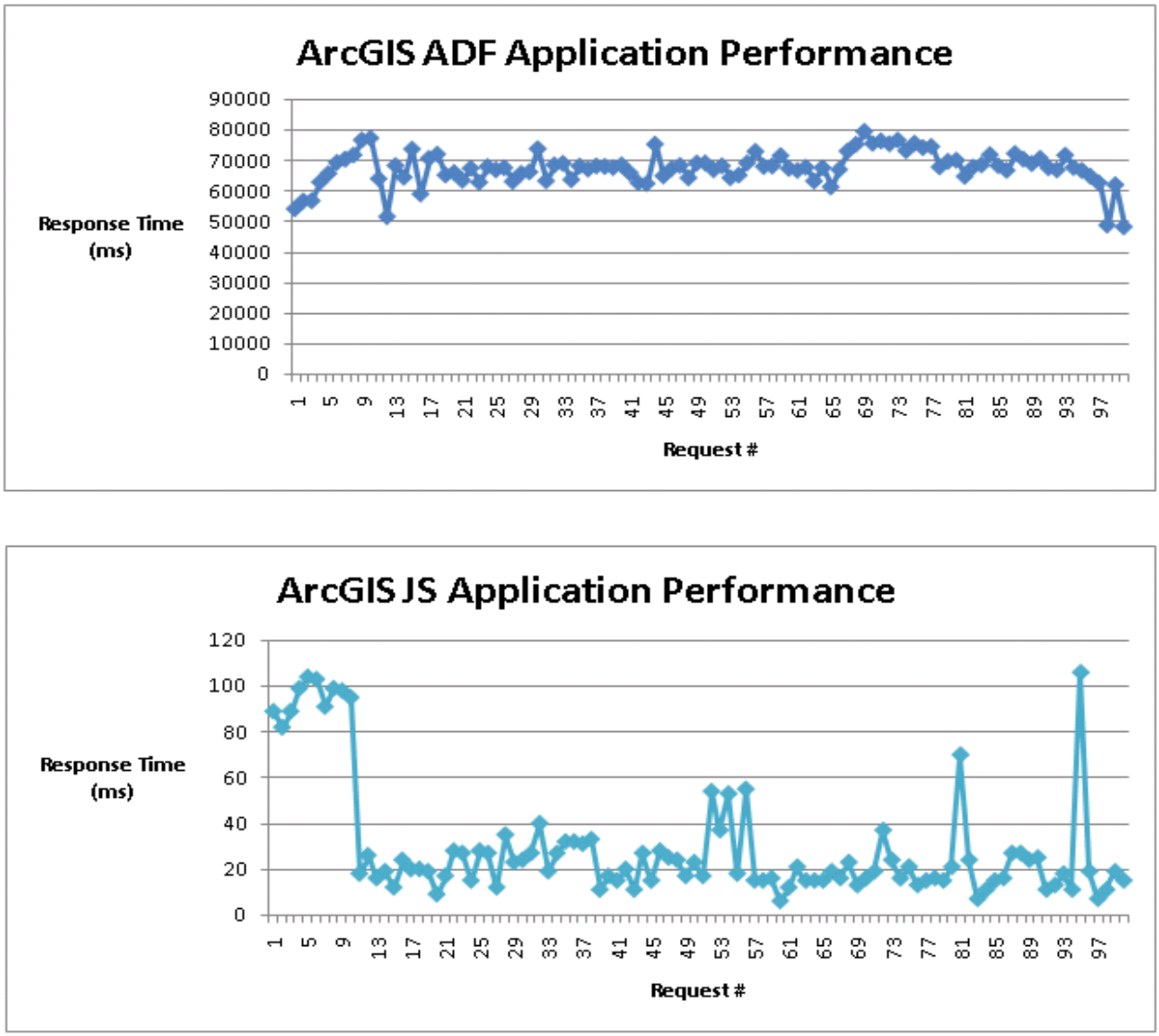


Figure 6. *Response Time: Thick vs. Thin Applications*

The average response time of the thick application is 67 seconds as opposed to 28 milliseconds for the thin application. It is important to note on initial load of the thick client, some maps get cached and subsequent operations such as zoom and pan work relatively faster. However, it is a significant delay for initial load and hence further tests were needed to identify the problem.

ArcGIS ADF application server Heap Memory Usage:

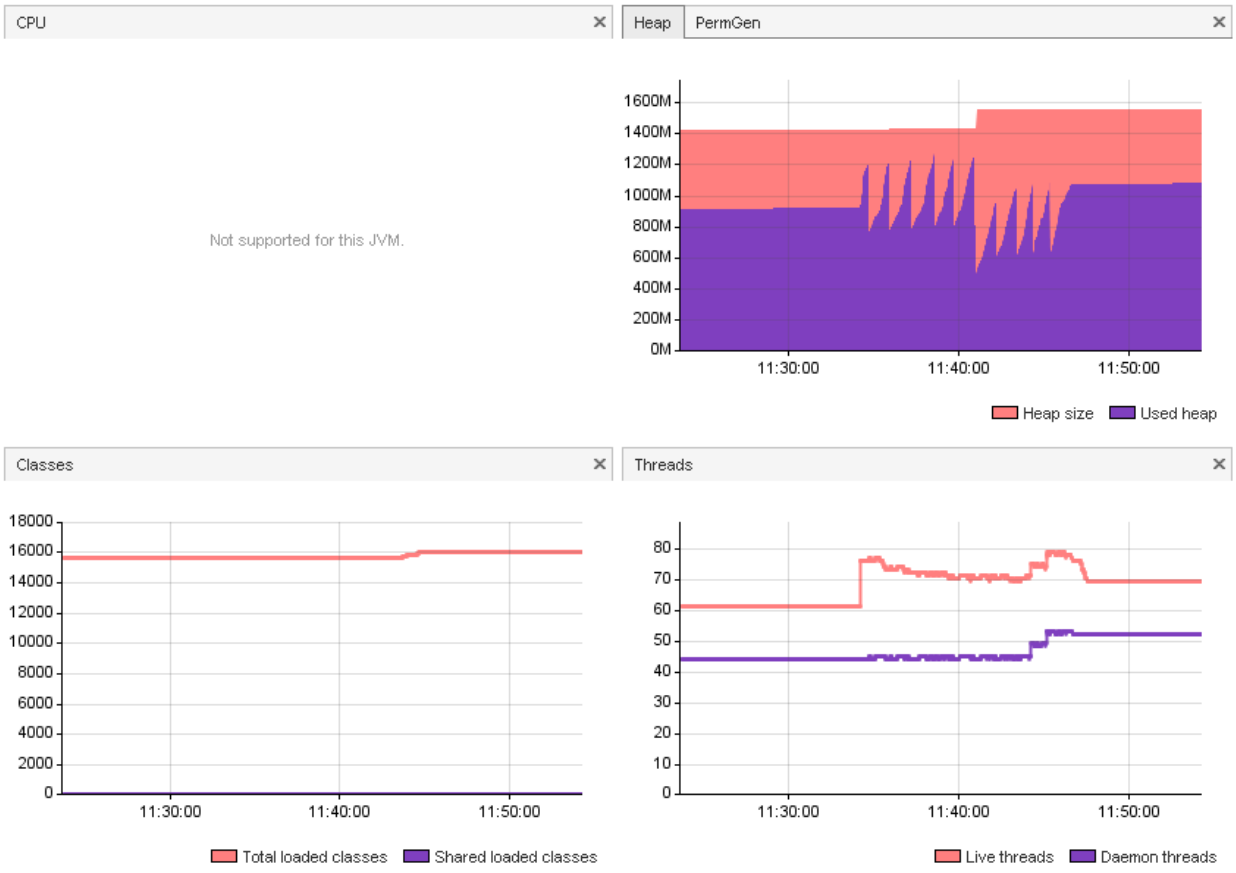


Figure 7. *Heap Memory Use of ArcGIS ADF Application Server*

The heap memory used by the ArcGIS ADF application server went up to 1.2 GB and the total allocated heap size actually went up to almost 1.6 GB as JMeter started sending requests. To improve performance, we dedicated a separate application server instance for hosting the thick application instead of deploying it in the ArcGIS application server. Furthermore, we fine tuned the JVM memory parameters to allocate a minimum of 0.5 GB memory and a maximum of 2.0 GB memory in increments of 0.5 GB. Although these changes did make some performance improvements, they were not significant enough. Then we tested the heap memory use of the application server handling the web services request between client applications and the ArcGIS map server. The test results are depicted below.

ArcGIS web services Heap Memory Usage:

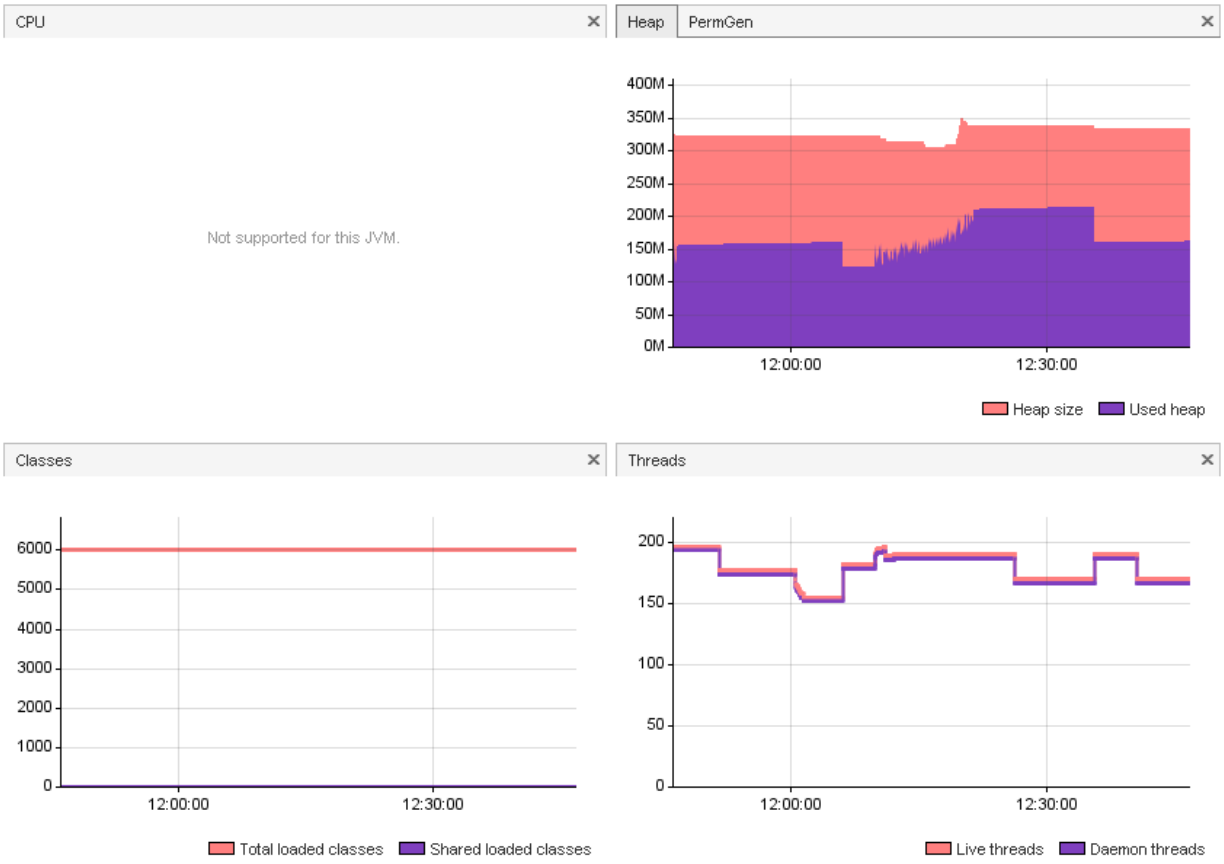


Figure 8. *Heap memory use of ArcGIS web services application server*

As you can see, there was no significant increase in the used (200 MB maximum) and allocated heap memory (350 MB maximum). The server actually was configured to use a maximum of 1.0 GB memory. It does make sense as this application server is just a broker for forwarding requests and responses between the client application (ADF) server and the ArcGIS map server (SOM & SOC). Subsequently, we monitored the SOC processes handling the service requests i.e. the CPU use of the SOC process and the average system load are shown below.

SOC process CPU usage and average system load:

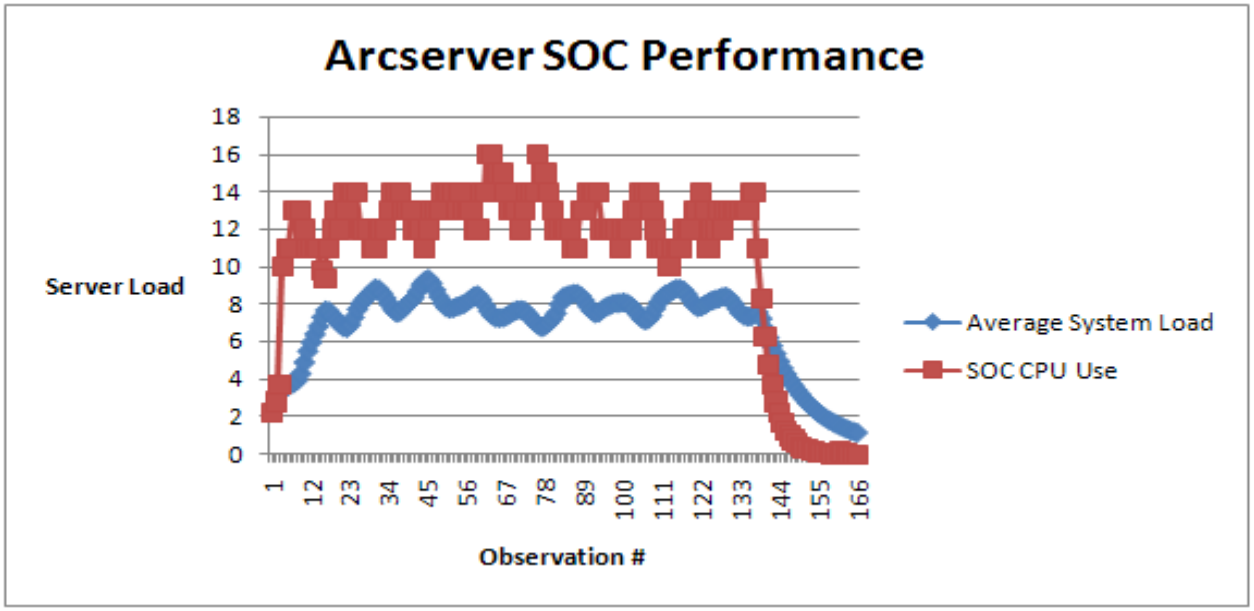


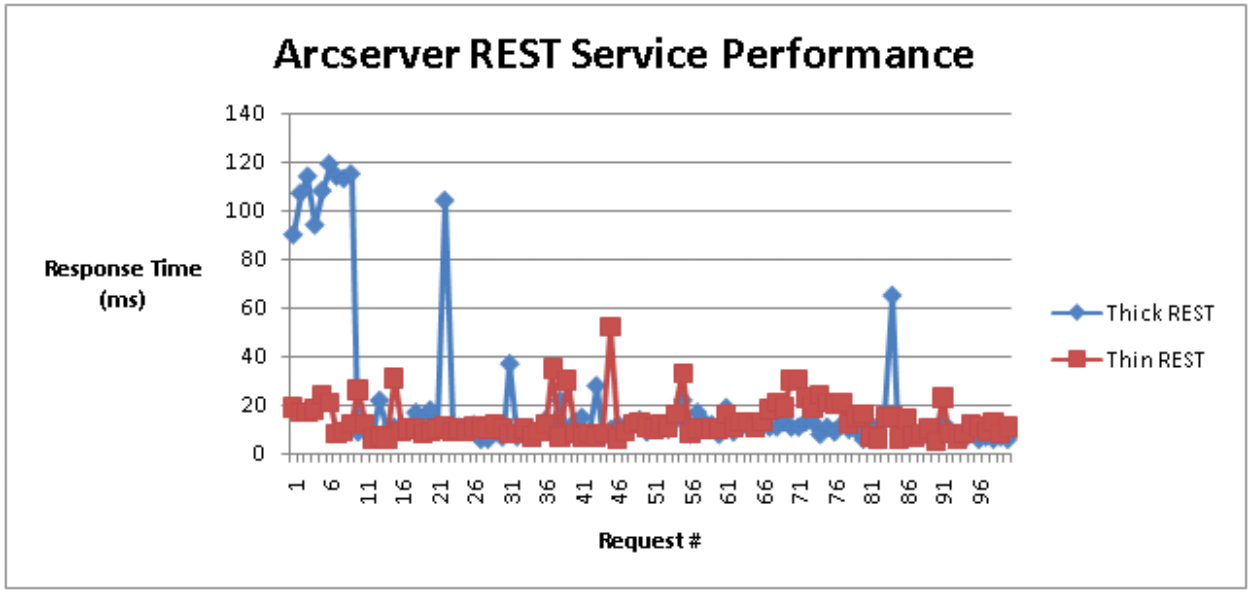
Figure 9. *ArcGIS SOC Performance*

As service requests from the thick client increased, there was significant increase in the CPU usage of the SOC processes and average system load.

We concluded from these tests that potentially three steps can be taken to further improve the performance. 1) The map service, hosted on the ArcGIS map server, has too many layers and data that slow down the performance of the SOM and SOC processes. A quick solution would be to break the set into thinner services. However, there is overhead and performance issues associated with running too many services – the right balance has to be struck, 2) The SOC processes of the ArcGIS map server could be clustered for load balancing, and 3) The ArcGIS ADF application server could be clustered for load balancing. Other tricks such as caching etc. could be used for speeding up performance. See sections below for further details.

ArcGIS Mapping Services: REST and WMS on Thick and Thin Services

We also tested the mapping services by sending requests to them directly from JMeter instead of through a client application. We tested two services; one is a thick service with several layers and data, the same one used by the client applications above and the other one is a thin service with a limited subset of layers and data. Both services were hosted on the ArcGIS Server. We tested them through the ArcGIS REST interface as well as though the OGC WMS interface.



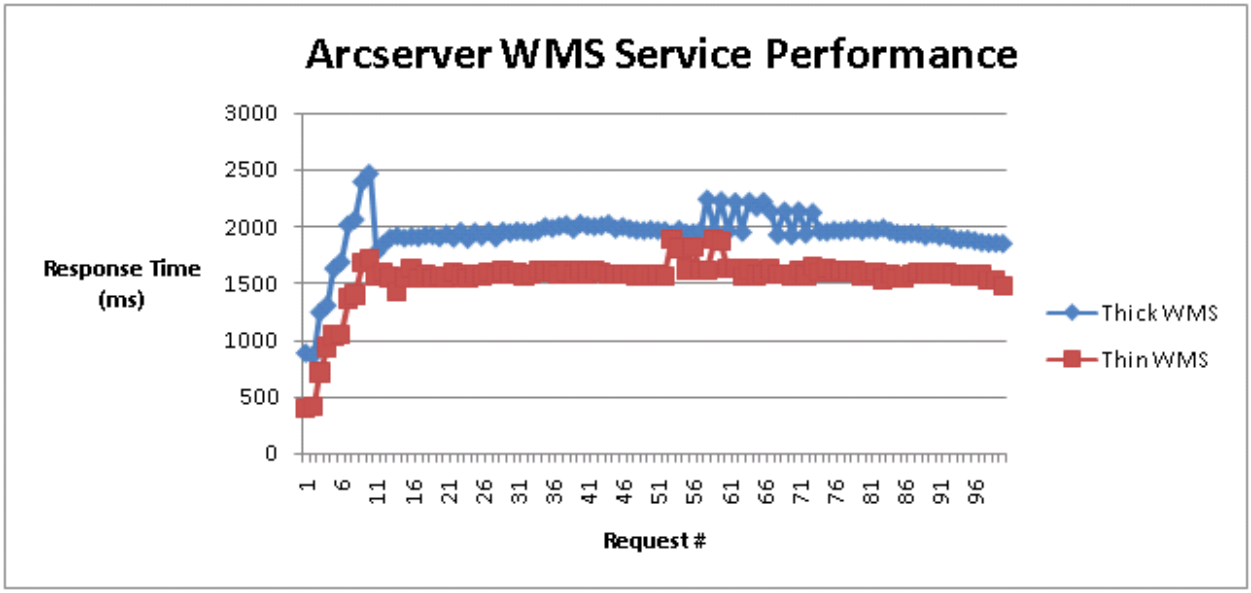


Figure 10. *ArcGIS Server REST & WMS Service Performance*

The average response time of the thick REST service is 22 milliseconds compared to 13 milliseconds for the thin REST service. The average response time of the WMS service is actually a little higher compared to REST service. The WMS requests to the thick service took an average of 1800 milliseconds compared to 1500 milliseconds for the WMS requests to the thin service. As expected, a thin service does improve performance and it is better to have few thin services instead of a thick service loaded with several map layers and data.

Mapping Services: ArcGIS WMS and GeoServer WMS

Finally a comparison of response time between an ArcGIS WMS service and GeoServer WMS service is presented below. Both services actually serve identical data, a map of Hudson River area in NY, using the same data source (shape file).

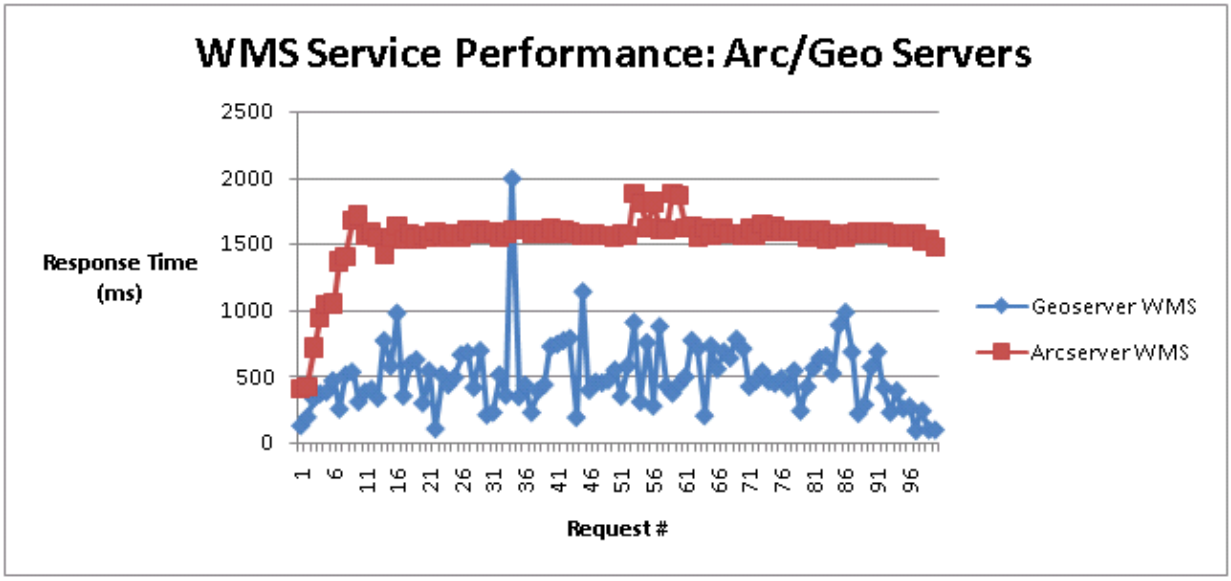


Figure 11. *WMS Service Performance: ArcGIS Server and GeoServer*

The WMS requests to the GeoServer took an average of less than half the response time compared to the requests sent to ArcServer.

In this section, we have provided some general ideas on performance monitoring and tuning and presented some of our own experiences. Please note that this is not an exhaustive testing and not meant to provide any concrete conclusions on the performance of various products. Besides, there are several other factors that may affect performance and thus a much more controlled environment will be needed for thorough testing.

Maintenance and Backward Compatibility:

Another important factor we considered is the issue related to maintenance, upgrade, and compatibility. The ability to maintain a constituent software product operational and up to date by applying patches and upgrades in a timely manner reduces the system down time and improves system security. This is one of the advantages of COTS software and to some extent supported open source software compared to open source software. One of the main issues we faced with upgrades is backward compatibility. Even though most of the software products in our architecture are backward compatible and provided a migration path, the mapping applications we developed required significant custom code and a large part of it was not compatible with newer versions. Also, it was not easy to contribute the custom code back to the original code base. Consequently we had to maintain old versions of the software as well as re-write the custom code tasks which required significant additional resources. Many open source

software address this problem by enabling developers to contribute code back to the code base repository. This is one of the reasons we introduced Open Layers into our mix of software products.

Scalability:

Another important factor in our evaluation is scalability. As our enterprise grows and engages in newer projects, the volume of data to be processed internally and served to public increases significantly. We address this problem by choosing Oracle/ArcSDE as our main data server and ArcGIS Server as the primary map server. These products are highly scalable and can be clustered for load balancing and fail over. In addition, applications and tools supporting simple to advance and complex geoprocessing functions, visualization, and analysis need to be supported. Often it is hard to accomplish this with one software product. Hence we chose thin client side technologies such as Open Layers, ArcGIS JS, and Google Map API for developing visualization and analysis tools with simple to intermediate level functionality. These tools can be embedded in web pages quite easily and enrich the user experience considerably. On the other hand, thicker server side frameworks such as ArcGIS ADF and RedSpiderStudio are needed for developing applications that can handle large number of map layers and data and provide advanced visualization and analysis functions. However, such frameworks are often overkill for development and maintenance and take up more resources such as system and staff than necessary.

Interoperability:

Publishing geospatial data services through interoperable interfaces enable enterprise data to be accessed by third party client applications, thereby increasing the use of data and the ability to integrate, visualize and analyze data with other datasets. Subsequently, this can lead to new and innovative use of data and shed new light on interdisciplinary research problems. Interoperability increases the intrinsic value of the data and could potentially provide the enterprise with new project opportunities.

One of the missions of CIESIN, as the host of the NASA's Socioeconomic Data and Applications Center (SEDAC), is to develop and operate applications that support the integration of socioeconomic and earth science data and to serve as an "information gateway" between the earth sciences and social sciences. We serve our geospatial datasets via OGC compliant WMS, WFS, WCS, and WPS interfaces. In addition, we also support KML files to support our data on Google Earth and Google Maps API.

The important thing to consider here is if and to what extent a map server software supports these standards. Some software products may only be partially conforming to standards. Also, we wanted to pick one software package to support multiple standards instead of different products for different standards. For these reasons, we selected the IONIC RedSpiderWeb software earlier and the GeoServer software recently. A comparison between ArcGIS Server and GeoServer is provided in the table below.

Service Capabilities	ArcGIS Server 9.3.1	GeoServer 2.0
WCS compliant	Fully compliant to WCS 1.0, 1.1 and 1.1.1	Fully compliant with WCS 1.0 and 1.1
WFS compliant	Fully compliant to WFS 1.0 and 1.1 (transactions)	Fully compliant to WFS 1.0 and 1.1 (transactions and locking)
WMS compliant	Fully compliant to WMS 1.1.1 and 1.3	Fully compliant to WMS 1.1.1
Web map Output	As JPEG, GIF, PNG, PDF, SVG, KML and GeoRSS	As JPEG, GIF, PNG, PDF, SVG, KML and GeoRSS
Projection on demand	Geometry service enables sophisticated projection operations on applications that do not have the ability to perform such operations independently	On the fly reprojection for WMS and WFS from a pool of hundreds of supported EPSG projections stored in a database
SLD compliant	Fully compliant to SLD 1.0	Full SLD support, to support map styles
Filter encoding	Fully compliant to Filter Encoding 1.0 and 1.1	Full Filter support on all data formats in WFS

Data Formats for Publishing:

Another factor to consider is the formats supported for publishing. The data formats supported for publishing by ArcGIS Server and GeoServer are listed below.

Service Data publishing format	ArcGIS Server 9.3.1	GeoServer 2.0
Vector Data	Map document (*.mxd) and Map service definition (*.msd)	Shapefile - ESRI(tm) Shapefiles (*.shp) and directory of spatial files stored as a datastore
Raster Data	Raster dataset (from a geo-databases or file on disk), GeoTiff, BIL or layer file referencing a raster dataset or compiled image service definition (containing one or more raster datasets and defined processes)	ArcGrid (coverage format), GeoTiff (Tagged Image File Format with Geographic Information), Gtopo30 (coverage format), ImageMosaic (Image mosaicking plugin), WorldImage (raster file with a spatial data file) and Image pyramids
Spatial database	Database connection file (*.sde), personal and file geo-databases, map document referencing data from a versioned geo-database	PostGIS compliant with OpenGIS Simple Features Interface Standard (SFS), ArcSDE (*.sde), DB2 and Oracle
Processing Tool	Support for geoprocessing map document with a tool layer or toolbox (*.tbx)	Not supported

Advantages of using MXD's and MSD's:

Mxd file is a map file format native to the ESRI suite of products such as ArcMap. Mxd file is saved as a compound format that includes map layout, styles, map descriptions, and embedded objects saved in the map from various sources.

Msd file is a map service definition format which was recently introduced with the ArcGIS Server 9.3.1 version. With the Mxd you need to manually develop strategies for optimization for better performance such as map caching etc. The more dynamic the map, the less gains you get from caching. With Msd's ArcGIS Server addresses these challenges with a new drawing engine that speeds up the dynamic map rendering time and shortens the tile creating time for cached services. [ESRI 2, 2009]

Advantages of using Layers and Group Layers:

A layer in GeoServer refers to raster or vector data that contains geographic features. Vector layers are also known as feature types and raster layers as coverages. Layers basically correspond to every feature that needs to be represented on a map.

All layers have a source of data, called a datastore. When publishing layers on GeoServer you can set the projections, bounding box, styles etc. Hence GeoServer gives you full control on the representation of a layer. However, the workflow for publishing a set of 10 or more layers can get messy when making WMS requests to individual layer objects over and over again. With the group layers you can make a single WMS request to multiple individual layers by a single layer reference name. You can then set the projection for the group of layers together and set styles on the individual layers. For the Server, the group layer generates a single image as a response just like an individual layer. [GeoServer 2, 2009]

Factors leading to choosing one data format over another and the advantage thereof depend primarily on the usage and data process workflow in an Enterprise for a web GIS solution.

Total Cost of Ownership (TCO):

TCO tries to analyze and offer a statement on the financial impact of deploying an Information technology product over its life cycle. The TCO in the deployment of a web GIS system includes the cost of hardware, software, operating expenses, and long term expenses.

In choosing the hardware, we have mainly considered the ease of integration into our existing infrastructure and expertise of the existing staff to minimize cost of additional training. That means leveraging or purchasing servers and workstations similar to those in the existing infrastructure.

Software selection is one of the most complicated processes. One hard choice is the selection of open source versus commercial off the shelf (COTS) software. Our long experience in working with both type of software suggests that there is no general answer to this and the evaluation has to be done on a case by case basis. For COTS software, one has to consider responsiveness and accessibility of the support team to

resolve issues, bugs, and questions as well as training, documentation and resources provided. For the open source software, one has to consider maturity of the product, community support and adoption, active development, and on-line resources available for solving issues. Apache web server software meets these requirements. One has to think carefully if one is an early adopter of open source software. One of the approaches we have been taking recently in general is choosing open source software for which paid support is provided by a company. This minimizes the risk of introducing open source software into a production environment. For example, paid support options are available for the open source software GeoServer.

Customization:

ESRI has a support center with its online technical resources. It maintains a large knowledge base, user forums and download center for code snippets and examples. It also provides web based help and support blogs in addition to a support call center.

GeoServer is a free open source, open standard Java based software server which allows flexibility in map creation and data sharing. Since it follows an evolutionary model and is not proprietary they do not maintain as large a support group as ESRI. But do have a large user/developer community who keep improving on the server and respond to queries from fellow users through blogs and provide good documentation for the resources.

Network Bandwidth Capacities:

One important factor contributing to the overall system performance is the bandwidth of the enterprise network that connects the web servers, application servers, map servers, work stations, and data servers. For example, in the CIESIN implementation, when we upgraded the network connection between the workstations running ArcGIS Desktop tools, the map servers and the data servers to a switched 1Gbps speed from 100 Mbps, there was significant improvement in data processing and data development activities. There are various tools available to measure and monitor the speed and throughput of the network between servers and workstations. Such measurements can be used for designing and improving the network bandwidth.

In a web GIS environment, if the public facing services and applications have to transfer large volumes of data, the upstream bandwidth connecting the enterprise LAN to the WAN and the Internet is important. CIESIN utilizes an Internet 2 and 100 Mbps Commodity Internet connections.

Important Usability factors

Besides architectural considerations there are some important usability issues and scenarios that may help improve application performance and thereby improve the overall user experience. In the web GIS world, high performing maps basically implies applications should load fast initially and that response time to user requests within the application such as zoom, pan, identify should be as quick as possible.

Thinner Services with appropriate Symbology

In GIS, too much Information may not necessarily add value to the application and may in fact confuse the end user. Even if your enterprise has the resources to publish large amount of services with enormous number of layers, the end result of such an approach is slower system performance. A more focused approach with 3-5 operational layers overlayed over a baselayer that is cached will garner better performance gains.

Another important point is developing business needs and use cases where complex symbologies and annotations are required to enhance the end users experience for better visualization and analysis. It's simple use the plain vanilla approach with very basic symbology and save the complexity for base cached layers.

Cached Layers

Map caching is a very effective way to make your maps run faster in a more slick and seamless manner [ESRI 2, 2009]. A map cache is basically a store of layers at different scales fused together and rendered as map image tiles. So a call to a server for a map cache is much quicker than dynamic rendering of a map each time a user requests for it. It improves performance, quality of work and helps your enterprise meet the industry standards set by Google and Yahoo amongst others.

For implementation of an effective caching scheme that sits right for an enterprise solution, these are some of the factors to be considered:

- Logical separation of operational layers from base layers
    - Operational layers: layers that will be directly involved in additional GIS functions
    - Base Layers: layers that give orientation and added aesthetic value to the map. Though this will differ from enterprise to enterprise a few examples are: Boundaries, Ocean, Rivers, Roads etc.

- Choosing scales and scale dependencies: This is crucial, as this sets apart a well designed map from an ad-hoc map design method. It is important to always keep the base layers in mind when designing a web mapping application. A well designed base layer should be reusable to give a more standardized map development enterprise solution.

- Choosing a coordinate system: There may be instances where you will be overlaying your cache with another cache, it is absolutely essential in these situations to use the same coordinate systems for both.

Aggregating and Clustering Large Datasets

Representation of large number of features in the client degrades performance after a couple of hundred simple point features. With complex polygon features the number is even less than that [Noyle, 2009]. To improve performance, you may want to cluster the sample features and aggregate data at smaller map scales and gradually show more features as the user zooms in to a larger map scale. By following this approach, you have control on the level of aggregation that is needed at a particular map extent before it is rendered on the client-side.

For e.g. the Global Population of the World (GPW) dataset is a very large dataset that uses a proportional allocation gridding algorithm utilizing more than 300,000 national and sub-national administrative units to assign population values to grid cells[CIESIN, Columbia University & CIAT-1, 2005]. To create a web map service for this dataset alone would imply reduced performance and increased loading and response time. The method used to reduce degradation in performance is the conversion of complex grids to simple centroid point features [CIESIN, Columbia University & CIAT-2, 2005]. To further reduce the response time, the centroids are clustered together at different resolutions, thereby reducing the number of features rendered at a particular scale threshold.

Conclusion:

Finally, this paper concludes on the note that these are the various factors to consider while designing an Enterprise web GIS solution. However, it is also important to note that every enterprise has its unique situation, business needs, and strategic interests and the importance given to these factors may vary accordingly.

References:

Alesheikh AA., Helali H., Behroz HA. *web GIS: Technologies and its Applications.* ISPRS Technical Commission IV Symposium 2002, Ottawa: Canada, 2002. 1-9

Center for International Earth Science Information Network (CIESIN), Columbia University; and Centro Internacional de Agricultura Tropical (CIAT)-1. *Gridded Population of the World Version 3 (GPWv3)*. Socioeconomic Data and Applications Center (SEDAC), Columbia University, Palisades: NY, 2005 - [2009 Oct 22]. Available from: http://sedac.ciesin.columbia.edu/gpw

Center for International Earth Science Information Network (CIESIN), Columbia University; and Centro Internacional de Agricultura Tropical (CIAT)-2. *Gridded Population of the World Version 3 (GPWv3): Centroids*. Socioeconomic Data and Applications Center (SEDAC), Columbia University, Palisades: NY, 2005 - [2009 Oct 22]. Available from: http://sedac.ciesin.columbia.edu/gpw

ERDAS: ERDAS *Products ERDAS APOLLO*. Available from: http://www.erdas.com/Products/tabid/56/cdf/1850/default.aspx

ESRI 1: ArcGIS Server 9.3 Help: *Components of an ArcGIS Server system. Release 9.3 [Internet].* Redlands: California. c1995 - 2009 - [cited 2009 Nov 25]. Available from: http://webhelp.esri.com/arcgisserver/9.3/java/components_of_server.htm

ESRI 2: ArcGIS Server 9.3 Help: *Publishing optimized map services**?** Release 9.3 [Internet].* Redlands: California. c1995 - 2009 - [cited 2009 Nov 25]. Available from: http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Publishing_optimized_map_services

ESRI 3: ArcGIS Server 9.3 Help: *What is map caching? Release 9.3 [Internet].* Redlands: California. c1995 - 2009 - [cited 2009 Nov 25]. Available from: http://webhelp.esri.com/arcgisserver/9.3/java/index.htm#what_is_map_caching.htm

GeoServer 1: GeoServer :*GeoServer Architecture. [Internet].* c2009 GeoServer 2008 Nov 18- [cited 2009 Dec 1]. .. Available from: http://geoserver.org/display/GEOSDOC/1+GeoServer+Architecture

GeoServer 2: GeoServer :*Data. [Internet].* c2009 GeoServer - [cited 2009 Dec 1]. Available from: http://docs.geoserver.org/2.0.0/user/webadmin/data/

Ajay. *GeoServer - a migration story!! – Part 2 - Talking the architecture.* Available from: http://www.jeevanchaaya.com/2009/03/12/geoserver-a-migration-story-%E2%80%93-part-2-talking-the-architecture/

Noyle B. GIS AND .NET DEVELOPMENT: *Usability and the GeoWeb Part 4: Make it Fast [Internet]. Fort Collins: Colorado. 2009 Oct 7 – [cited 2009 Nov 24].* Available from: http://briannoyle.wordpress.com/2009/10/07/usability-and-the-geoweb-part-4-make-it-fast.

Peng Z., Tsou M. *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Network*s. John Wiley and Sons, New Jersey: USA, 2003. 15-19