Design Strategies to Improve Performance of GIS Web Services

Shengru Tu, Maik Flanagin, Ying Wu, Mahdi Abdelguerfi, Eric Normand, Venkata Mahadevan
Computer Science Department
University of New Orleans, New Orleans, LA 70148
[shengru, ywu, mahdi, enormand, venkata]@cs.uno.edu; Maik.C.Flanagin@mvn02.usace.army.mil

Jay Ratcliff
U.S. Army Corps of Engineers
New Orleans, LA 70118
Jay.Ratcliff@mvn02.usace.army.mil

Kevin Shaw
Naval Research Laboratory
Stennis Space Center, Mississippi
shaw@nrlssc.navy.mil

Abstract

GIS systems are ubiquitous distributed systems, since geo-spatial information adheres to almost everything. Considering the characteristics of GIS, the following four design-decision issues are particularly crucial: transactional mode (synchronous versus asynchronous), service granularity (fine-grained versus coarse-grained), delivery manner (chunk versus stream), and transmission formats (GML versus binary). In this paper, we have shared our experience in making choices in these four dimensions.

Key words: web service, GIS, performance, Internet map service, SOAP.

1. Introduction

Traditional IT infrastructures in which systems and applications were managed and owned by one enterprise are giving way to networks of applications owned and managed by many business partners. Web services have been gaining strong momentum as a platform upon which to develop applications that take advantage of the Internet infrastructure. By web services, we mean the self-contained, web-enabled applications capable not only of performing business activities on their own but also possessing the ability to engage other web services in order to complete higher-order business transactions. Technically, web services refer to the web applications based on three specifications, namely SOAP (Simple Object Access Protocol), WSDL (the Web Service Definition Language), and UDDI (the Universal Description, Discovery and Integration standard). These efforts are critical because service-oriented computing inherently requires wide acceptance. However, standards themselves do not deliver working systems. Constructing successful working systems requires smart design.

In this paper, we consider design strategies for heavy-duty web services that have a large number of concurrent requests, are involved with complex computation, and require large-quantity data transmission. An eminent example of such is the web service that connects to Geographic Information System (GIS) back-end servers. There are at least three characteristics of GIS services that make it difficult to design GIS web services with satisfactory performance. First, services provided by a GIS typically requires heavy CPU usage due to the complex computation involved in the underlying computational geometry. Second, GIS services often transmit large resulting data sets such as images. Third, the “clients” of GIS web services are often some complex software tools such as the CAD desktop applications. For scalable GIS, simply establishing communications between components is not sufficient. Performance should always be a central consideration in the design of GIS web service systems.

In this paper, we are to highlight the crucial design-decision issues for GIS web service systems by establishing a 4-dimensional decision-making framework. The four dimensions are transactional mode, service granularity, delivery manner, and transmission formats. We hope that our paper can garner sufficient attention from the web services designers to take note of the design principles and build effective solutions.

2. Related Works

The World Wide Web Consortium (W3C) has been successfully steering SOAP’s evolution from an HTTP-based RPC mechanism in XML to a leading interoperable technology with replaceable bindings. The web service technology is a practical engineering outcome in the software industry. Design of web services systems have received attention from both industry [1, 2, 5] and academia [3, 10, 11].

Parallel to the evolution of the web service technology, the Open GIS Consortium (OGC) has been pursuing web map services with interoperability of map servers and clients. The first specification on simple web map services was released in 2000. The current web mapping services standards include the Web Map Service (WMS) and the Web Feature Service Implementation Specifications (WFS) [6]. Since OGC's WMS was formalized before SOAP emerged, WMS and WFS do not refer to SOAP.

3. Background

Most of the work reported in this paper is based on the needs of the U.S. Army Corps of Engineers New Orleans District (USACE – New Orleans). The district plans, designs, constructs, operates and maintains federally sponsored navigation, flood control, hurricane protection and water resources development projects in south central and coastal Louisiana. At USACE – New Orleans, engineers and analysts have been using a myriad disparate commercial software packages to manage their GIS and CAD projects, including products from companies such as Intergraph, ESRI, and Bentley. A team of IT workers have been working on integration of all the software programs through a centralized means of data access. The efforts of data consolidation in the early stages was reported in [9].

4. Design Strategies for GIS Web Services

For nontrivial web services systems, behind each web service is a backend system that fulfills the task. In front of each web service client is typically an application that consumes the service. This application can be as simple as a web browser or as complex as a GIS or CAD application suite. Fig. 4.1 depicts a conceptual structure of such a web services system. The "service consumer" and the "web services client agent" components are in the same local area network (LAN) or even in the same computer (In our experiments, we used Java RMI). Similarly, the backend system and the web services interaction component are typically located in the same LAN. However, between the web services provider and the web services client, we can only assume the SOAP/HTTP protocol because different departments and external clients may be involved in the system.

Many design principles for distributed software are not only applicable to the design of web services, but also are more crucial to apply to the GIS web services. In this section, we will discuss the design decisions regarding the transactional mode, the service granularity, the communication manners, and the transmission formats.

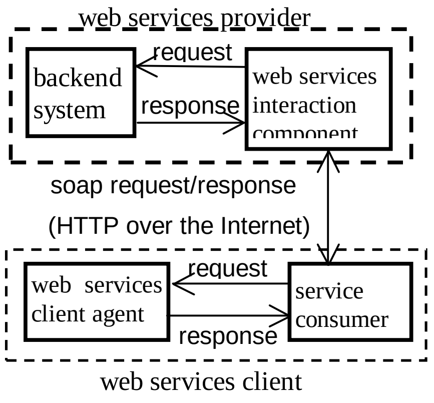

Fig. 4.1 The web service model with backend

4.1 Synchronous services versus asynchronous services

When we push GIS systems to the Internet, the success of a GIS web service will usually be measured by the number of hits to it. Ironically, increasing the number of concurrent hits inevitably will overwhelm the server's capacity because GIS services typically require intensive computation and enormous data, straining CPU and bandwidth respectively. A general strategy to deal with the performance issue is to minimize synchronous transactions. A classic method to turn a client's synchronous service call into an asynchronous request is by using the callback design pattern. When the results are ready, the caller is informed.

A GIS web services client is often another application or a cascading map server. As described by the OGC WMS specification, a cascading map server is a WMS that behaves like a client of other WMSes and also behaves like a WMS to yet other clients. In an example given by the same specification, a cascading map server can aggregate the contents of several distinct map servers into one service. Furthermore, a cascading map server can perform additional functions such as output format conversion or coordinate transformation on behalf of other servers. In the latter scenario, the cascading map server obviously would prefer to make fetching data and format conversion in parallel. This requires asynchronous services from the map services providers. Applying the callback design pattern is a straightforward way to achieve asynchronous services for the cascading map server or the applications that also provide other web services. Because they already have their web services server instances, all it needs is to include a URL of the client (the cascading map server or the application) that will inform the client to fetch the result. The cost of the implementation of callback is the creation of a web services server.

By providing asynchronous services, the server can not only conceal its computation time from the users' perspective by overlapping server's computation with the clients' other operations, but can also optimize job scheduling by prioritizing the requests.

4.2 Fine-grained versus coarse-grained services

For GIS web services, the communication through the Internet is the weakest among distributed systems, and the amount of data to transmit is often large. To avoid excessive communication overhead, granularity of services deserves particular attention.

For web services systems, coarse-grained services should be preferred to fine-grained services, even though the conventional object-oriented design tends to be in favor of fine-grained services for flexibility. For instance, a GIS database provides users with accesses to every geometric element. However, if a GIS web feature service provider allows remote users with the same accessing power as that to a local database, assembling a map with thousands of features at the user side would cause thousands of web services requests, the communication overhead of which would severely hinder the performance. Rather, we prefer to provide services that deliver a collection of features (with certain chosen attributes in a bounding box).

Using the GIS services as resources for analysis, the users' typical usage pattern is query-select-fetch. For example, a GIS expert will first query all the available feature classes in an area. Then the useful feature classes are selected manually from the query result. Finally the chosen feature classes are fetched from the resources and added to the current workspace. To efficiently implement such kind of service, we highly recommend a design pattern called "secondary object identities", which has been used by the CORBA software community [7] for years. The **GetCapabilities** operation defined in the WMS specification is another good example, which fetches a rich set of service-level metadata of each web map service. Realizing the reason of the "secondary object identities" design pattern can help us actively apply it in our own design. We will explain this design pattern using an example of a query-select-fetch process.

Assuming that the bounding box is a parameter, a simple object-oriented design for the query-select-fetch process would consist of four services: (1) getMaps -- a service that returns all the maps that overlaps with a bounding box; (2) getFeatureClasses -- a service that returns all the names of the available feature classes in a map; (3) getDesc -- a service that returns the description of a feature class; (4) getFeatureContent -- a service that returns the content of a feature class. Using these fine-grained services, a query-select-fetch process can be carried out as described in the following pseudo code (The variable declarations and other details are omitted).

```
map_list = getMaps(bounding_box);
for each map in map_list {
    feature_class_list = getFeatureClasses(map);
    for each feature_class in feature_class_list {
        display(getDesc(feature_class));
    }
}
... // the user selects feature classes
for each selected feature_class in its residing map {
    getFeatureContent(map, feature_class);
    ...
}
```

Suppose there are *m* maps that overlap with the given bounding box; each map has *f* features; and the user chooses *c* feature classes. Then the process will make ($1 + m * f + c$) service calls. The essential idea of the "secondary object identifier" design pattern is to use a data structure that contains a combination of an object identifier along with adequate information to support the user's selection. Correspondingly, the above task can be served with the following two services: (1) getFeatureClassInfo -- a service that returns all the necessary information of the available feature classes in the maps that overlaps with a bounding box; (2) getFeatureClassContent -- a service that returns the content of a feature's identifier. Using these coarse-grained services, a query-select-fetch process can be simplified into the following pseudo code.

```
feature_class_Info_list=getFeatureClassInfo(bounding_box);
for each feature_class_Info in feature_class_Info_list {
    display(feature_class_Info);
}
... // the user selects features
for each selected feature_class in its residing map {
    getFeatureClassContent(feature_id);
    // feature_id is from in feature_class_Info
    ...
}
```

The feature_id is contained in feature_class_Info. Thus, the number of service calls are reduced to (1 +

c). A little hidden complexity of this approach is that the structure featureClassInfo has to contain feature name, description, and identifying information such as the residing map name and the feature name that composes feature_id. The client side must be able to extract the value of feature_id. In SOAP, such kind of data structures can be easily represented and parsed.

In Fig. 4.1, the "response" from the "backend system" seems to be simply forwarded by the "web services interaction component" to the web service client as a SOAP response. However, their difference can be more than formatting but significantly different granularity because the communications in these two sections are greatly different. For flexibility, we have defined a set of fine-grained services for the interaction between the "backend system" and the "web services interaction component". These two components are in the same LAN. On the other hand, the services provided by the web services provider are mostly coarse-grained ones because the "web services interaction component" and the "web services client agent" communicate through the Internet.

4.3 Transmitting in streaming versus single chunk

The streaming technique is commonly used in transmitting multimedia contents. For web browsing images, streaming is not very crucial; the required number of pixels will be limited by the web browser's screen size. However, for browsing large vector data sets or for powerful web service clients such as the MicroStation user, large vector data sets or high-resolution TIFF images are commonly needed.

The specification of SOAP 1.2 considers streaming in the bindings of the *Request-Response* message exchange pattern (MEP). It is said: "In the web services systems, responding SOAP nodes may begin transmission of a SOAP response while a SOAP request is still being received and processed web services.". However, that is not what we need. The SOAP Request-Response MEP does not mandate any correlation between multiple requests nor specific timing for multiple requests. To support iterating over a large dataset in fix-size chunks, the *iterator* design pattern can be applied. Considering that transmitting large datasets is a common need for GIS analysts, the *iterator* pattern is included in our web services design.

Having the separation between the "web services client agent" and "service consumer", the final streaming delivery to the service consumer component requires another streaming between the "web services client agent" and a destination object in the "service consumer" components. This separation has at least two benefits for the consumer components. First, the consumer component needs to know nothing about the web services. Second, a streaming delivery can be emulated even if the web services transmit in a whole chunk. In our experimental implementation, we used the Java ObjectStream objects. While the ObjectInputStream object constantly attempts to read, the ObjectOutputStream object sends geometric elements for vector datasets or byte arrays for images whenever a chunk arrives to the "web services client agent".

4.4 Transmitting in binary versus GML

The specification of the GML has been growing rapidly with the efforts of the OGC, to pursue the ultimate interoperability among geometric data. WFS mandates GML to express features within the interface. (The datastore used to store geographic features is opaque to client applications.) Leading GIS software vendor such as ESRI and Intergraph have implemented WFS. Intergraph also provides their GeoMedia users with a GML exporter [3]. In our design, we have chosen to transmit data in binary between our web services providers and MicroStation clients. This choice was made mainly due to performance concerns. MicroStation uses the DGN file as their optimized operational data representation. For the same geometric information, GML documents' sizes are often tens of times larger than DGN files. Many MicroStation installations run on relatively old computers. Converting GML to DGN takes tangible delays. Many of USACE's partners such as the local governments' users communicate with our web services without high-speed connections.

5. Challenges: Web Services for Simulation

Our services anticipate more hydrologic analysts and civil engineers to request access to the Enterprise GIS from their software tools. Therefore, we have been extending our efforts to general web services for highly computational clients. Our strategy is to properly pipeline the existing WMS and WFS capacities of the GIS systems – ESRI's ArcGIS and ArcSDE software family and Intergraph's GeoMedia software family. Fortunately, both vendors have made the WMS and WFS available without extra cost.

For some of these analysis-oriented clients, the GML mandatory requirement of WFS can be overkill for the non-GIS clients' software. For example, a number of critical hydrologic simulations require huge amounts of data. The analysis programs are in Fortran 90 and running on supercomputers for days. In such cases, performance is a serious concern. We plan to provide web services to meet the needs for data collection and transmit data efficiently. Often such web services are long-duration jobs. Therefore, our system model will include persistent queues between the WS Client Agent and the clients' applications as shown in Fig. 5.1.

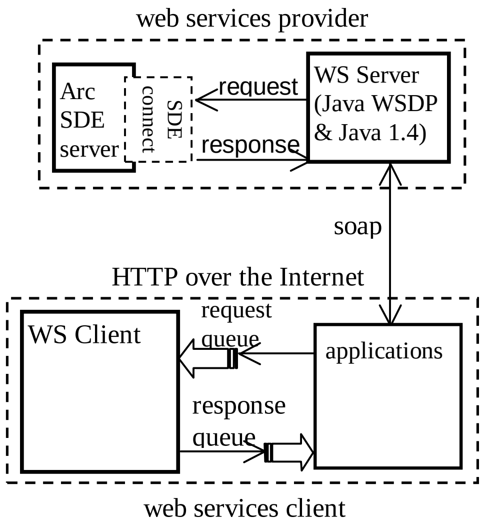

Fig. 5.1 Web Services for other applications

Recently, a more interesting challenge comes from requirements of simulation that supports walk-through for training and emergency response exercises. In contrast to typical fly-through simulations that are based on datasets from GISs, the walk-through simulations require datasets across GISs (for voyaging streets and landscapes) and ACE systems (for walking into buildings and constructions) as well as human resource systems (for reviewing affected people). For large-scale simulation, a significant part of the costs will be data collection. For real-world-scenarios simulations, static data preparation will not be adequate. Having the simulation access each system through the corresponding web services would make the whole system very flexible and reusable. A specific desirable requirement from simulation is streaming. Especially for large map imagery data sets, streaming can allow the simulation software to consume (display) the map long before it completely finishes loading. Furthermore, streaming will be a necessity for running simulations in small devices such as PDAs that have limited storage space.

6. Conclusion

SOAP-based web services have made communications between varied software components flexible and easy. It is ready to extend our daily use of the Internet from merely browsing web pages to carrying out distributed computing and transactions. However, we have to realize that simply enabling communications between computers and software components does not mean that we can construct efficient systems. Many principles of design for distributed software have never been so important as they are today because there have never before been so many large-scale distributed software systems that need to be built. One type of the most ubiquitous distributed systems is the GIS system, since geo-spatial information adheres to almost everything.

By no means have we covered all the important principles, but we hope that our paper can garner sufficient attention from the GIS web services designers and providers to take note of the design principles, so that the GIS society can meet the future challenges effectively.

7. References

[1] A. Arsanjani, et al, "Web Services: Promises and Compromises", *Queue*, pp 49-58, March 2003.
[2] P. Fremantle, S. Weerawarana, and R. Khalaf, "Enterprise Services", CACM, 45:10, 77-82, Oct. 2002.
[3] N. Gibbins, S. Harris and N. shadbolt, "Agent-based Semantic Web Services", Proceedings of the 12th WWW, Budapest, Hungary, pp 710-717, May 2003.
[4] Intergraph: GeoMedia GML Exporter, http://imgs.intergraph.com/interop/extensions.asp
[5] S. Kleijnen and S. Raju, "An Open Web Service Architecture", *Queue*, pp 39-46, March 2003.
[6] Open GIS Implementation Specifications, http://www.opengis.org/techno/implementation.htm
[7] D. Slama, J. Garbis, and P. Russell, *Enterprise CORBA*, Prentice Hall, Upper Saddle River, NJ, 1999.
[8] S. Tu, et al, "A Systematic Approach to Reduction of User-Perceived Response Time for GIS Web Services", Proceedings ACM GIS 2001, Atlanta, GA, pp 47-52.
[9] S. Tu, et al, "Achieving Interoperability for Integration of Heterogeneous COTS Geographic Information Systems", Proceedings of ACM_GIS 2002, McLean, VA, pp 162-167.
[10] J. Yin, et al, "Engineering Server-Driven Consistency for Large Scale Dynamic Web Services", Proceedings of the 10th international conference on World Wide Web, Hong Kong, pp 45-57, May 2001.
[11] L. Zeng, B. Benatallah and M. Dumas, "Quality Driven Web Services Composition", Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary, pp 411-421, May 2003.