Design and Evaluation of Experiment Methods for Improving Performance In GIS Web Services[1)]

Seung-Jun Cha[1], Yun-Young Hwang[1],
Yoon-Seop Chang[2], Kyung-Ok Kim[2], Kyu-Chul Lee[1]
[1]Department of Computer Engineering, Chungnam National University
[2]Electronics and Telecommunications Research Institute(ETRI), Korea
{junii, yyhwang, kclee}@cnu.ac.kr, {ychang, kokim}@etri.re.kr

Abstract

By introducing Web Services, distributed GIS(Geospatial Information System) services from different vendors can be dynamically integrated into a GIS application using the interoperable standard SOAP(Simple Object Access Protocol). However, it is debatable whether SOAP can really meet the performance requirement of GIS. Additionally, GIS Web Services' performance may be improved by using asynchronous technique. Ajax, one of the technologies of "Web 2.0". Integrating Ajax(Asynchronous JavaScript and XML) approach into GIS visualization Web Services have performance enhancement, because it provides more interactive user experience. This paper presents an experimental evaluation of the performance of different SOAP variants: standard SOAP, SwA/MIME, and SOAP/MTOM. SOAP/MTOM is proved to be the fastest and the most efficient messaging protocol. For sintegrating Ajax approach, compare performance of models between Web Services and Web Services using Ajax. This comparison results that Web Services using Ajax represent good performance in images fetching and user roundtrip time because it fetches required images beforehand.

1. Introduction

Over the past decade GIS(Geographic Information Systems) technology has evolved from the traditional model of stand-alone systems to distributed models. Distributed GIS services will be implemented more extensively by using Web Services. By introducing Web Services, distributed GIS services from different vendors can dynamically integrated into a GIS application using the interoperable standard SOAP(Simple Object Access Protocol) protocol[1]. However, it is debatable whether SOAP can really meet the performance requirements of GIS.

The bursting of the dot-com bubble in the fall of 2001 marked a turning point for the web. The companies that had survived from the collapse seemed to have some things in common. It is called "Web 2.0", because dot-com collapse marked some kind of turning point. Ajax(Asynchronous JavaScript and XML) is one of the Web 2.0's

[1)] This research was supported by the Ministry of Information and Communication, Korea, under the College Information Technology Research Center Support Program, grant number IITA-2006-C1090-0603-0031

This work was supported by the IT R&D program of MIC/IITA [2005-S044-02, Integrated Processing Technology for Multi-Sensor Spaital Imagery Information]."

technologies[14]. It provides asynchronous communication between the client and the server. So integrating Ajax approach into GIS visualization Web Services have performance enhancement for user interface.

This paper presents an experimental evaluation of the performance of different SOAP variants: standard SOAP, SwA(SOAP with Attachments) using MIME(Multipurpose Internet Mail Extension), and SOAP using MTOM(Message Transmission Optimization Mechanism). These standard protocols are evaluated by transporting multiple raster(JPEG) map data and vector(GML) data.

And then for additional performance enhancements, we integrate Ajax approach into GIS Web Services. For that, we first compare models of classical Web Services and of integrated Ajax into Web Services and evaluate performance between them. We prove that integrated Ajax into Web Service model indicate good performance because Ajax approach provides interactive user experience.

The rest of this paper is organized as follows: a brief discussion of OGC GIS Web Services, SOAP's variants, and Ajax is given in section 2. Section 3 we discuss some related achievements in recent years, and section 4 contains how we conduct the experimental evaluation for choosing the best protocol among SOAP and its variants, and evaluation results and analyses are followed. In section 5, there is comparison between models of classical Web Services and of integrated Ajax into Web Services, and results of performance evaluation between them are in section 6. Conclusions are followed in section 7.

2. Background

2.1 OGC GIS web services

GIS(Geographic Information Systems) has a lot to benefit by the adoption of the service computing model. As mentioned also in the introduction geographic information comes from different and diverse sources and in different formats. This is especially true for the environmental related information which has to combine not only data from different sources by also models and software.[15]

Technically, Web Services' technologies have provided the necessary standards for applications in different domains to integrate with GIS data and services, significant accomplishments in GIS Web Services have led to several exemplifying map and image services that adhere to Web Services standards and bring terabytes of geospatial data and digital maps to enterprise developers who house no GIS data.

The OGC(Open Geospatial Consortium) has successfully executed efforts for GIS Web Services(OWS) initiative has undergone multiple phases – including the Web Map Server(WMS), Web Feature Server(WFS), Web Coverage Server(WCS), and OGC Web Service Architecture, which support application developers in integrating a variety of online geoprocessing and location services.

2.2 SOAP and It's variants

SOAP is a platform-independent, extensible and XML-based protocol for distributed computing. A SOAP message consists of three different elements: the SOAP Envelop containing SOAP Body and the optional SOAP Header[9].

A typical SOAP message is structured as follows:

```
<?xml version='1,0' ?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
<soap:Body>
   <m:data xmlns:m=;http://example.org/people">
    <photo>/aWKKapGGyQ=</photo>
          <wav>Faa7vROi2VQ=</wav>
   </m:data>
  </soap:Body>
</soap:Envelope>
```

SwA/MIME applies MIME attachments to SOAP, using multipart/mime content type and putting the SOAP Envelope in the root MIME part and other related attachments in ensuing MIME parts inside the MIME package. It relies on HREF attribute and Content-ID MIME header to relate attachments to SOAP message parts[4].

A typical SwA/MIME is structured as follows:

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: mymessage.xml@example.org
<?xml version='1,0' ?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
<soap:Body>
<m:data xmlns:m='http://example.org/stuff' >
<photo href="http://example.org/me.jpg" />
<wav href="http://example.org/my.wav" />
</m:data>
</soap:Body>
</soap:Envelope>

--MIME_boundary
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-Location: 'http:// example.org/me.jpg
fd a5 8a 29 aa 46 1b 24
….
```

WSA/DIME(WS-Attachment using DIME) uses DIME to send binary data and is more efficient than SwA. DIME is a simpler protocol than MIME[11]. But there are no plans for standardizing WSA/DIME, which has restricted its development. So we also exclude it from the test.

A typical WSA/DIME is structured as follows:

```
00001 1 0 0 0010 0000000000000000000
0000000000000000 0000000000101000
00000000000000000000000000110110101
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: mymessage.xml@example.org
Content-Description: A SOAP Envelope with my picture in it

<?xml version='1,0' ?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
<soap:Body>
<m:data xmlns:m='http://example.org/stuff' >
```

```
<photo href="http://example.org/me.jpg" />
<wav href="http://example.org/my.wav" />
</m:data>
</soap:Body>
</soap:Envelope>
--------------------------------------------------------------------
00001 0 0 0 0001 0000000000000000000
000000000011000 0000000000001001
00000000000000000000000000110000
 http://example.org/me.jpg
 image/jpg
 fd a5 8a 29 aa 46 1b 24
…
```

SOAP/MTOM, a part of SOAP 1.2 specification, applies XOP to SwA. Staying in the XML Infoset, SOAP/MTOM-based attachments are equivalent to embedded SOAP elements semantically to the endpoint SOAP Nodes. With many Web Services standards defined on XML Infoset model, its significance lies in the fact that the presence of attachments in SOAP messages is no longer an exceptional or special case[5].

A SOAP/MTOM message, which is equivalent to the previous SOAP message, can be represented as follows:

```
Content-Type: application/xop+xml;  charset=UTF-8;
   type="application/soap+xml; action=\"ProcessData\""
Content-Transfer-Encoding: 8bit
Content-ID: mymessage.xml@example.org
Content-Description: A SOAP Envelope with my picture in it
<?xml version='1,0' ?>
<soap:Envelope xmlns:soap='http://www.w3.org/2002/12/soap-envelope' >
<soap:Body>
<m:data xmlns:m='http://example.org/stuff'>
<m:photo  xmlmime:contentType='image/jpg'>
<xop:Include href='cid:http://example.org/me.jpg'/></m:photo>
<m:wav mlmime:contentType='sound/wav'>
<xop:Include     xmlns:xop='http://www.w3.org/2004/08/xop/include'
   href='cid:http://example.org/my.wav'/></m:wav>
</m:data>
</soap:Body>
</soap:Envelope>
--MIME_boundary
Content-Type: image/jpg
Content-Transfer-Encoding: binary
Content-ID: http://example.org/me.jpg
fd a5 8a 29 aa 46 1b 24
…
```

2.3 Ajax (Asynchronous JavaScript and XML)

Ajax is a style of web application development that used a mix of modern web technologies to provide a more interactive user experience. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change.

Ajax is not a technology. It is an approach to web applications that includes a couple of technologies. These are JavaScript, HTML, Cascading Style Sheet(CSS), Document

Object Model(DOM), XML and XSLT, and XMLHttpRequest as messaging protocol. [6]

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 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.

3. Related works

Previous researchers had examined the performance of using SOAP in a number of different scenarios. The paper[16] reported that performance could be simply and effectively improved by reducing the payload of a SOAP message using SwA using MIME, WSA using DIME, and XSOAP. The paper[10] argued on the overall performance of SOAP and the effectiveness of compression and binary encoding as ways of improving performance or reducing bandwidth requirements. They used and tested SOAP, gZip, remoting binary, and MTOM for compression and binary encoding.

Another study had described about applying Web Services to GIS. The paper[7] indicated about GIS Web Services about satellite imaging. It says there are problems when using GIS Web Services, because of sending large size of data. The paper[13] describes about developing GIS visualization Web Services for geophysical applications. Since images and capabilities documents can be too large and transferring these data over the internet is cumbersome, its first priority is researching technique for improving WMS performance.

Related to the Ajax, The paper[12] is introduce integrating Ajax approach into GIS Web Services. It describes of invoking Web Services in the Ajax Model but it is only theoretical so it has nonexistence of implementation.

Our study differs from previous studies in that we compare performance among standard SOAP, SwA/MIME, and SOAP/MTOM, and apply Ajax into the Web Service at the first time. There are no studies of comparing SOAP protocols with GIS Web Services. And there are no studies of implementing and evaluating applications for GIS Web Services using Ajax.

4. A performance evaluation of SOAP variants for GIS web services

In our test scenario (Figure 1), the client reads external GIS data and attaches it to the SOAP message, and then the message is sent to the server. The server reads that message and composes simple response message, then sends it back to the client again.

In the real situation of GIS web services, the server should send GIS data, and the client displays the data on the screen. However, the web server is not implemented to send large data(e.g. the Axis web server cannot send over 16Mbytes data when we have tested), the test is performed in the opposite direction. As you know the result of our evaluation later, there is no case that the web server should send over 16Mbytes GIS data in a normal Internet environment.

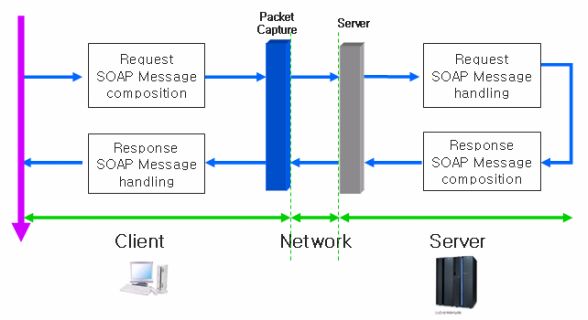


Figure 1. Test scenario

4.1 Metrics and methods

The following performance evaluation metrics and methods are used during our test. A packet capture program in the client-side is used in evaluating message size and serialization/deserialization time.

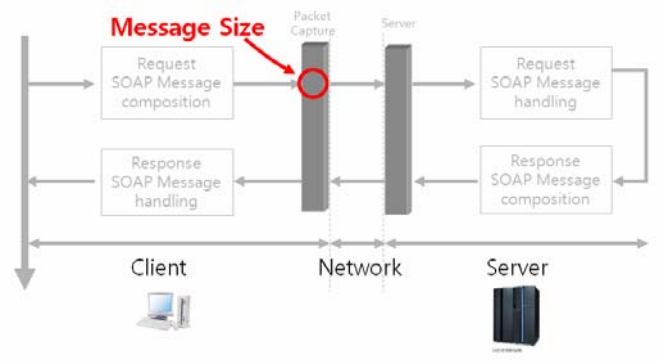


Figure 2. Message size

4.1.1 Message Size: It means the size of the SOAP message which includes attachment data at the client. Packet capture program is used for evaluating it (Figure 2).

4.1.2 Roundtrip time: It means the time taken from the point after the SOAP message composition ends at the client to the point the client receives the response message from the server. It is evaluated by using timestamp in the source code (Figure 3).

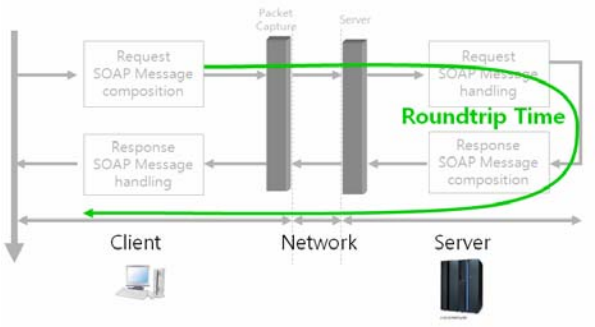


Figure 3. Roundtrip time

4.1.3 Serialization and deserialization time: Serialization is the conversion of an object instance to a data stream of byte values in order to prepare it for transmission. Deserialization is opposed to serialization, and it is for computer's handling. The test evaluates the client-side's serialization/deserialization and the server-side's serialization/deserialization. But the server-side's serialization time and deserialization time cannot be clearly evaluated separately, because its web server cannot be modified to insert test code and the packet capture program cannot be used at the server (

Figure **4**).

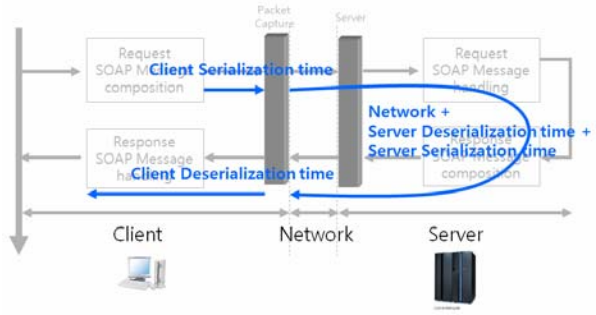


Figure 4. Serialization and Deserialization time

4.2 Environments

The configuration of the server systems is:
- Pentium 4 - 2.6GHz processor
- 1024Mbytes of memory (512Mbytes * 2)
- RedHat Linux 9

The configuration of the client systems is:
- Mobile Pentium 1.7GHz
- 512Mbytes of memory
- Microsoft Windows XP Professional

The SOAP implementations used in our test are Apache Axis because Axis 1.4 supports SwA/MIME, Axis 2.0 supports standard SOAP and SOAP/MTOM, and both along with the Xerces(XML parser). Java 1.5.06 is used for the programming language. Apache Httpd is used for web application, and Tomcat 5.0 is used as the container for Apache Httpd. The Ethereal[3], a shareware TCP traffic monitor, is used to evaluate metrics. The Server and the client are connected with a hub(NetGear FS2005, 100MBytes LAN), so that it is not affected by other network traffics. Two different operating systems are chosen for testing the platform independence and interoperability of SOAP messages.

4.3 Test data design

When we design the test data, we refer to the Google Map (
Figure 5). The Google Map provides 12 tiles of image in a window. And the average size of each tile is 20Kbytes. Assume that current Internet speed which is commonly used is

500Kbps. But the test environment is consists of the server and the client, and just one hub. The Ethernet speed is measured almost 90Mbps. The Ethernet speed is 180 times faster than the Internet speed, so the data size should be set 180 times bigger than the Google Map; 3Mbytes for one tiles and 36Mbytes of one window(12 tiles). Tests are performed by sending from 1Mbytes to around 64Mbytes of data.

Vector test data is composed of GML-typed data, because GIS services serve it for the response. Assume that it needs 1Kbyte polygon data for expressing a building, test are performed by communicating different numbers of buildings. The size varied from 1, 10, 100, 1,000, 5,000 to 10,000. 10,000 numbers of buildings are enough for GIS services to provide in one window.

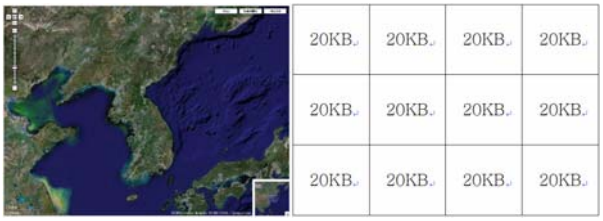


Figure 5. Image representation in Google map

4.4 Results

The message size result of sending raster data of different size is presented in Figure 6. SwA/MIME and SOAP/MTOM show similar in size. When using standard SOAP, attachment data must be encoded by either Base 64 Encoding or Hexadecimal text Encoding. Base64 is a method of encoding arbitrary binary data as ASCII text. Since Base64 encoding divides three bytes of the original data into four bytes of ASCII text, the encoded data typically is about 33% bigger.

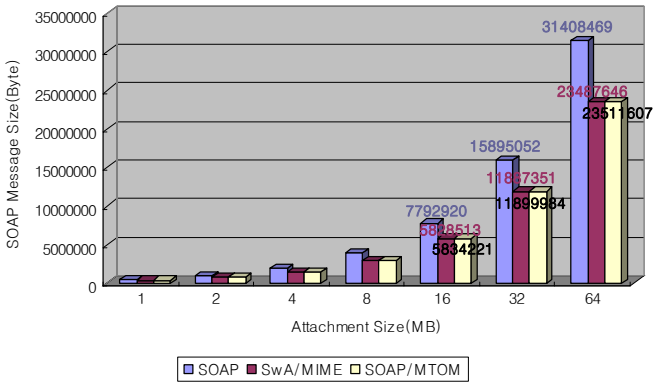


Figure 6. Message size of raster data

When we test the roundtrip time and serialization / deserialization time, SOAP/MTOM achieves the best performance (Figure 7 and Figure 8)

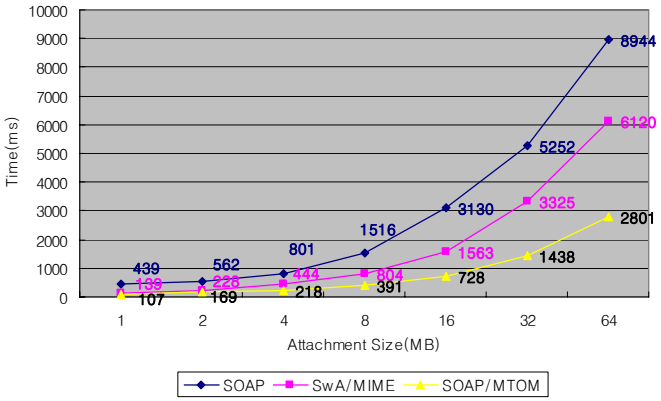


Figure 7. Roundtrip time of raster data

Using standard SOAP the external data is embedded into the SOAP envelope as an element. It takes a long time for serialization and deserialization (Figure 8). SwA/MIME and SOAP/MTOM use a URI as an element value to reference external data. That means the data isn't included in the SOAP envelop. Furthermore, SOAP/MTOM uses XOP optimization mechanism with XML infoset. It gives the recipient the option of using either the original file that may be identified by a URI, or to use a cached copy that accompanies the actual SOAP message. It enhances greatly the speed and of processing as the external data is already present when the recipient is starting processing the message.

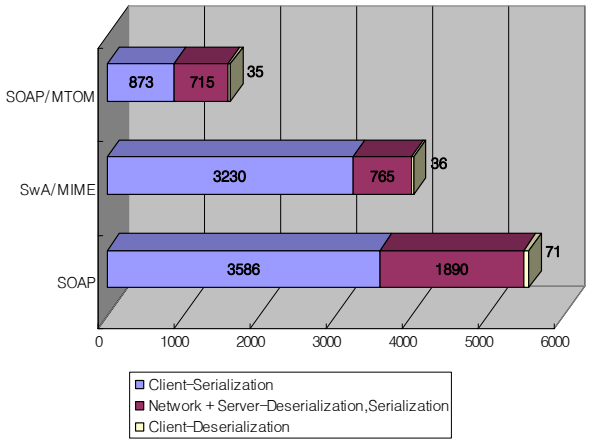


Figure 8. Serialization/Deserialization time of raster data

32Mbytes of raster data is used when we perform serialization / deserialization tests in Figure 10.

As a result, SOAP/MTOM takes almost 1500ms when evaluated 32Mbytes of raster data. But standard SOAP takes over three times than that.

Because vector data is ASCII-type, not binary, no encoding is necessary for attachments. SwA/MIME and SOAP/MTOM's message size is bigger than standard SOAP rather. This is because attachment data are in the boundary, and extra tags are produced additionally. In using SOAP/MTOM, it transfer SOAP message by using XML Infoset, its additional tags are more than SwA/MIME. (Figure 9).

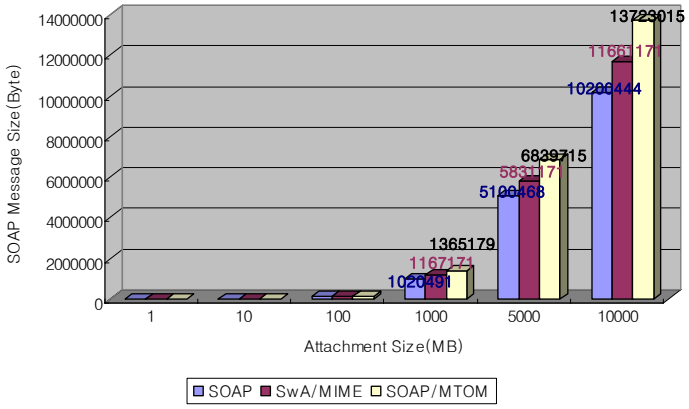


Figure 9. Message size of vector data

The result of the vector data's roundtrip time (Figure 10) is similar to that of the raster data's.

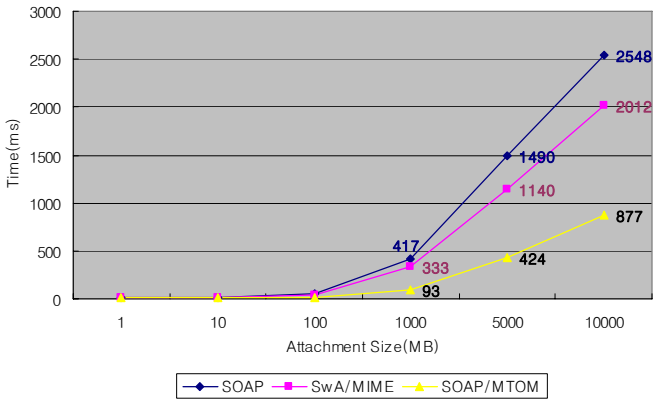


Figure 10. Roundtrip time of vector data

All attachments are tagged in standard SOAP. SwA/MIME takes less time because deserialization is faster than standard SOAP. As SOAP/MTOM uses XOP optimization and XML Infoset, it has the smallest roundtrip time.

As a result, when sending 10,000 numbers of GML Vector data SOAP/MTOM takes three times less than standard SOAP. It means that SOAP/MTOM also archives good performance even when sending a large amount of vector data.

5. Integrating Ajax into GIS Web Services

In Ajax, the client communicated with the server by using an XMLHttpRequest method. It provides asynchronous access to the server simply like the Googlemap. Its performance seems better because web pages using Ajax can appear to load relatively quickly since the payload coming down is much smaller in size. And the roundtrip time is less than the classic web application, so it provides more interactive user interface.

Figure 11 shows how applied Ajax to GIS Web Services. When an application is launched, images, which may be displayed on the screen, are fetched from the server. User may wait for looking displayed images for the moment. During that time, Ajax is

executed. It fetches extra images roundabout from Web Services from the server. When user drag the screen for watching another images, already fetched images is displayed faster.

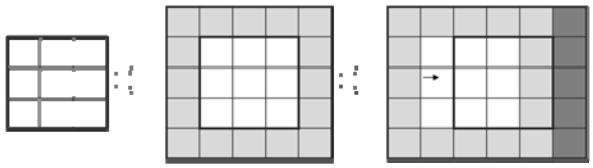


Figure 11. Test scenario

5.1 Classical web services

Web Services provide integrated environments within distributed systems. When the client makes a request for images files from the server, the server runs the inner JSP module which can access specific Web Service of providing image files. And then the server saves them to its repository. The client's working process must be waited until response message is arrived from the server, because the client's request is synchronously (

Figure 12).

Maybe there is some overhead when using the JSP module. It means performance is better than that without using the JSP module. But Web Services provide images with binary encoded, and there is no way to handle binary data in JavaScript. It is essential to use the JSP module for saving received binary data to JPEG image files.

The most significant benefit of classical Web Services is the characteristic of its platform independent. In the GIS Web Services, SOAP protocol is used for transferring messages like Web Services. So it can transfer messages without regarding other's platform.

Classical Web Services' running process is followed. ① The client request an image from the server. ② The server checks its existence. If it is not, call the specific Web Service for the request. ③ Web Services send a requested image by encoding SOAP/MTOM. ④ Save a received image from the Web Services. ⑤ If the server has already had a requested image, it returns the images name immediately. ⑥ Return the image's name that is requested, received, and saved.

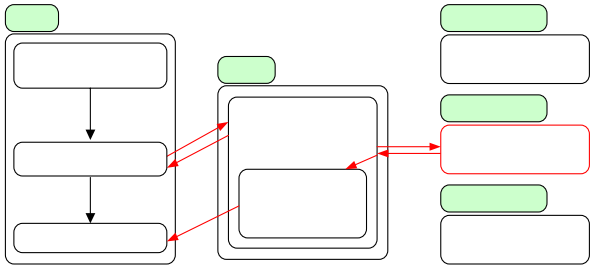


Figure 12. Classical Web Services

5.2 Integrating web services with Ajax

Integrating Web Services with Ajax is in
Figure 13. This model is added an XMLHttpRequest method to the classical Web Services model, so needed images can be requested and saved previously.

Because both the Ajax and Web Services are used together, two different sides' advantages are applied. It uses SOAP protocol for message transfer as Web Services do, so it is platform independent too. And by using an XMLHttpRequest method as Ajax, it enhances user interface. In the part of reading images from Web Services, It can read needed images and save them previously within user's waiting time not being reloaded whole web pages. And it reduces additional time of calling service unnecessarily by checking needed images' existence previously.

So it can integrate Web Services from different platform's application and access to the server asynchronous. It enhances user interface and reduces roundtrip time.

The running process is almost same as explained beforehand. It is different that needed images are taken previous by using Ajax technology.

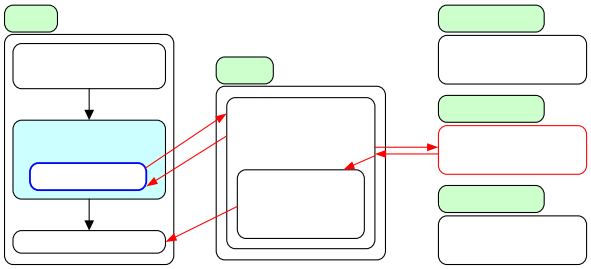


Figure 13. Integrated Web Services with Ajax

6. Performance Evaluation

When an application is launched, images which may be displayed on the screen are fetched from the server. If the server does not have those images, the server requests for fetching images from specific Web Service. User may wait for looking displayed images for the moment. During that time, Ajax is executed, so it fetches extra images roundabout from Web Services to the server. When user drag the screen for watching another images, already fetched image is displayed faster.

6.1 Metrics and Methods

6.1.1 Consideration characteristic of fetching time in Ajax: The time when user look at displayed images, the server creates a XMLHttpRequest Object and executes it. Extra images are fetched from Web Services to the server in that operation. So it takes less than user's waiting time. (Figure 14- ①)

6.1.2 Image fetching time: Displayed images must be fetched from Web Service regardless of using Ajax. Because the request using Ajax is asynchronous, images fetching time is less than using non-Ajax's fetching time. (Figure 14- ②, ③)

6.1.3 User response time: When users drag the screen, addition images must be fetched. Non-Ajax application is not include images in themselves, so it accesses Web Services for fetching images every time. But Ajax application has already fetched necessary images, so they are displayed faster. (Figure 14-④, ⑤)

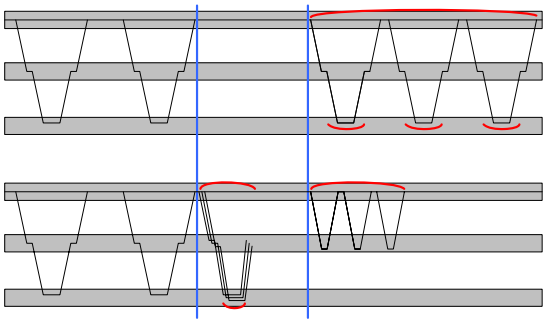


Figure 14. Evaluation metrics

6.2 Environments and test data design

For the test, we need three computers (Figure 15). The configuration of the client systems is:

- Pentium 4-1.66GHz processor
- 1Gbytes of memory
- Windows XP Pro.

The configuration of the server systems is:

- Pentium 3-1GHz *2 (Dual CPU) processors
- 2Gbytes of memory
- RedHat Linux 9

The configuration of the Web Server systems is:

- Pentium 4-1.66GHz processor
- 1Gbytes of memory
- RedHat Linux 9

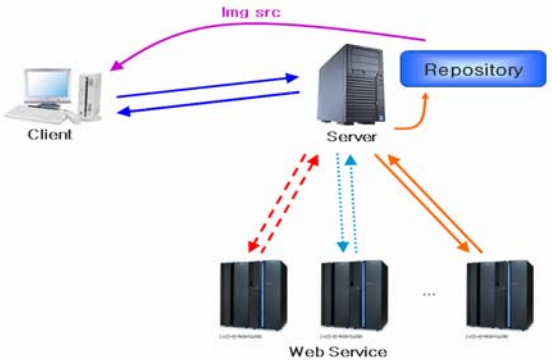


Figure 15. Evaluation Environments

All computers are connected in commonly used network. So test data is selected by considering generally used data. Google map use 20Kbytes of image tile so the tests are performed by receiving 20Kbytes of image.

6.3 Results

The result of consideration of fetching time in Ajax is presented in Table 1. It takes almost 2 seconds while all images(12 tiles) are fetched. It shows that 2 seconds is reasonable time to fetch images while user waited.

Table 1. Consideration characteristic of fetching time

	1	3	6	9	12
Ajax	186ms	475ms	938ms	1410ms	1988ms

In images fetching time, we compare Web Services using Ajax with Web Services using non-Ajax. Figure 16 shows that if only one image is fetched, the one takes longer than the other, because it has some overheads at XMLHttpRequest method. But, if multiple images are fetched the other takes longer than the one, because Ajax can communicate asynchronously.

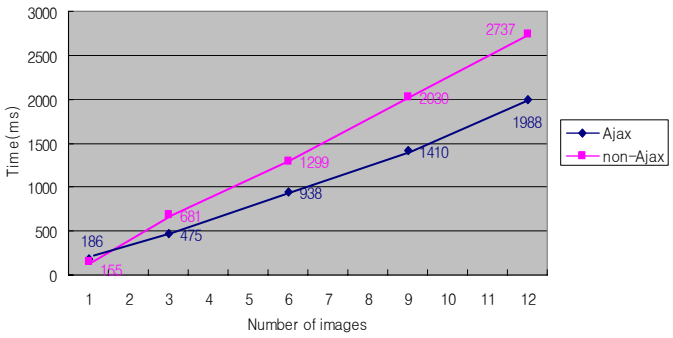


Figure 16. Image fetching time

Figure 17 indicated the user roundtrip time. In Web Services using Ajax the server has already read extra images and save them to its repository. So when the client requests mages, it can read them directly and the server does not need to connect Web Services. However in non-Ajax application, when the client requests images the server must connect Web Services for fetching images every time. So it takes longer than before. The roundtrip time is most important factor for user interface. If roundtrip time measured less, users realize that the application has good performance. The more evaluation data comes to many, the more gaps we get.

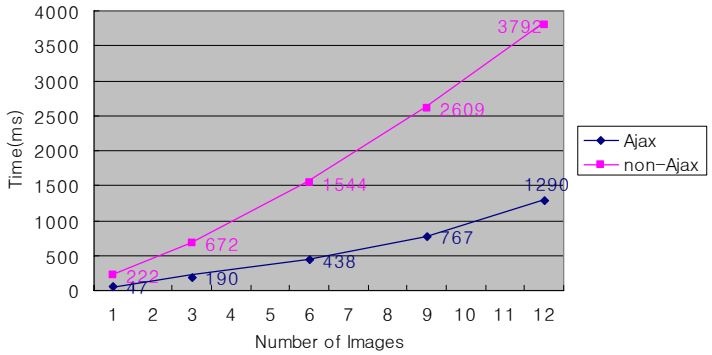


Figure 17. User response time

7. Conclusion

Our research is for performance enhancement in GIS Web Services. So we first have performed evaluation experiments of standard SOAP, SwA/MIME, and SOAP/MTOM using workload that reflect variety range of typical data types of GIS Web services.

Our tests indicate that SOAP performance dealing with raster data is improved in SOAP/MTOM. It is because SOAP/MTOM uses XOP optimizing mechanism so that serialization time and deserialization time is reduced remarkably. The larger data we test, the more gaps we get.

It also works when dealing with vector data. A standard SOAP message is made by embedding vector data as elements or attributes. Serialization/deserialization time of its SOAP message increases according to its number of attachments. Although total message size using SwA/MIME and SOAP/MTOM is bigger than standard SOAP because of tagging, roundtrip time is less than that. It is because SwA/MIME and SOAP/MTOM consist of small SOAP message and attachments exist in boundaries. Therefore, SOAP/MTOM's performance is almost equals in small Vector data, but in large amount of vector data it is better than using standard SOAP and SwA/MIME.

In addition to precede test, GIS Web Services' performance may have enhancement by using asynchronous technology. So we have chosen Ajax technology, one of the Web 2.0 technologies. Ajax provides asynchronous communication when the client request required images to the server. Web Services and Web Services using Ajax comparison results that Web Services using Ajax represent good performance in images fetching and user roundtrip time because it fetches them beforehand. In results of comparing and evaluating performance between them, Ajax's feature of user interactive interface enhances performance for user interface. Just one image fetching time tasks takes longer because asynchronous request has some overhead, but multiple images fetching time takes less to the contrary because it sends data with overlapping.

These results will contribute to integration construction's main technology of GIS systems. And also they will also contribute that in Web Services of dealing with large size of data used in GIS system, its performance should be enhanced.

8. References

[1] Alameh, N., "Chaining Geographic Information Web Services", IEEE Internet Computing, pp 22-29, 2003

[2] Barton John J., Thatte, S., Frystyk Nielsen, H., "SOAP Messages with Attachments", (on-line) http://www.w3c.org/TR/2000/NOTE-SOAp-attachments-20001211 , December 2000

[3] Combs, G., Ethereal, http://www.ethereal.com/

[4] Freed, N., Borenstein, N., "Multipurpose Internet Main Extentions (MIME) Part One: Format of Internet Message Bodies", (on-line) http://www.ietf.org/rfc/rfc2045.txt, November 1996

[5] Gudgin, M., Mendelsohn, N., Nottingham, M., et al. "SOAP Message Transmission Optimzation Mechanism", (on-line) http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/, January 2005

[6] Jesse James Garret, "Ajax: A New Approach to Web Applications.", http://www.adaptivepath.com/ publications/essays/archives/000385.php, Feb 2005

[7] Kotzinos, D., et al., "GIS Web Services and Satellite Imaging as a Framework for Environmental Monitoring: The Design and Implementation of a Vegetation Indices Calculation Test Case, Computer Science Department", University of Crete, Greece

[8] Lutz, M., Riedemann, C., Probst, F., "A Classification Framework for Approaches to Achieving Semantic Interoperability Between GIS Web Services", Institute for Geoinfomatics, University of Munster

[9] Mitra, N., SOAP Version 1.2 Part 0:Primer, (on-line) http://www.w3.org/TR/2003/REC-soap12-part0-20030624/, June 2003

[10] Ng, A., Greenfield, P., Chen, S., "A Study of the Impact of Compression and Binary Encoding on SOAP performance", Department of Computing, Macquarie University, North Ryde, NSW2109, Australia

[11] Sanders, H., Butek, R., Nash, S., et al. "Direct Internet Message Encapsulation (DIME)", (on-line) http://msdn.microsoft.com/library/en-us/dnglobspec/html/draft-nielsen-dime-02.txt, June 2002

[12] Sayar, A., Pierce, M., Fox, G., "Integrating AJAX Approach into GIS Visualization Web Services", Community Grids Lab, Indiana University, Bloomington, Indeana, 47404, USA

[13] Sayar, A., et al., "Developing GIS Visualization Web Services for Geophysical Applications", Community Grids Lab, Indiana University, Bloomington, Indiana

[14] Tim O'Reilly, "What Is Web 2.0. Design Pattern and Business Models for the Next Generation of Software", O'REILLY, September 2005

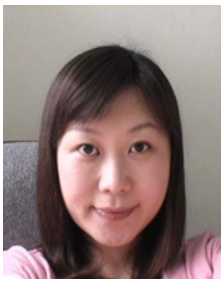
[15] Tu, S., et al., "Web Services for Geographic Information System, IEEE internet Computing", pp. 13-15, 2006

[16] Ying, Y., et al., "A performance Evaluation of Using SOAP with Attachments for e-Science", School of Computer Science, Cardiff University, 2004

Authors

Seung-Jun Cha

Received a B.E. degree in computer engineering from Chungnam National University, Korea, 2006. He is currently in amaster's course in department of computer engineering. His research interests include database, XML, Web Services, and GIS.



Yun-Young Hwang

Received a B.E. degree in computer engineering from Chungnam National University, South Korea, 2002, and M.E. degree in computer engineering from Chungnam National University, South Korea, 2006. She is currently a doctor's course in Department of Computer Engineering. His research interests include Web Services, Semantic Web, Ubiquitous Web Services.

Yoon-Seop Chang

Received a B.S. degree in mineral and petroleum engineering from Seoul National University, Korea, 1999, and M.S. degree in civil, urban and geosystem engineering from Seoul National University, Korea, 2001 and Ph D. degree in civil, urban and geosystem engineering from Seoul National University, Korea, 2005. In 2005 he joined ETRI (Electronics and Telecommunications Research Institute), Korea where he is currently a researcher in Telematics and USN Research Division.

Kyung-Ok Kim

Received a B.S. degree in clothing and textiles from Seoul National University, Korea, 1976, and M.S. degree in computer science from Ohio State University, USA, 1987 and Ph D. degree in computer science from Chungnam National University, Korea, 1998. In 1998 she joined ETRI, Korea where she is currently a team leader in Telematics and USN Research Division.

Kyu-Chul Lee

Received B.E., M.E., and Ph.D. degrees in computer cngineering from Seoul National University in 1984, 1986, and 1996, respectively. In 1994 he worked as a visiting researcher at the IBM Almaden Research Center, San Jose, California. From 1995 to 1996, he worked as a Visiting Professor at the CASE Center at Syracuse University, Syracuse, New York. He is currently a Professor in the Department of Computer Engineering at Chungnam National University, Daejeon, Korea.
His current areas include Multimedia Database System, Hypermedia Systems, Object-Oriented Systems, and Digital Libraries. He has published over 50 technical articles in various journals and conferences. He is a member of ACM, IEEE Computer Society, and Korea Information Science Society.