IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011
ISSN (Online): 1694-0814
www.IJCSI.org

Performance Analysis of Web Service Model for Video on Demand

Lavanya Rajendran[1], Ramachandran Veilumuthu[2]
[1] Department of Media Sciences, CEG, Anna University,
Chennai, Tamilnadu, 600 035, India.

[2] Department of Information Science and Technology,
CEG, Anna University,
Chennai, Tamilnadu, India

Abstract

In today's world, the demand for video in the Internet has gained high popularity. One of the best and optimal ways to provide the video contents on demand to users is through web services. This research aims to develop a generalized web service model for providing video on demand. To fetch the video file requested by the user, JAX-RPC is been used, which is more flexible and loosely coupled. The implementation of the developed web service depicts the methodology for providing video contents on demand in real time. The video distributors can adopt this model, as it ensures smooth and jitter-free videos, thereby both the users and the distributors will be benefitted. The performance measure, Round Trip Time (RTT) is estimated for the developed JAX-RPC video on demand model by requesting the video files of various sizes in both local invocation and remote invocation. The result shows that the developed model solves the connectivity issues and interoperability which was a major constraint with the rmi model. The performance was better for smaller video files when compared with the larger files and there was very little variation obtained between the local and remote invocation.

Keywords: *Video Service, Web Service model, Video on Demand, JAX-RPC.*

1. Introduction

Today, Internet users expect the on-line video to possess best resolution, less round trip time, without jitter, delay and essentially high throughput. The attempt to display media files was started from the mid-20th century. Little progress was made for several decades, primarily due to high cost and limited capabilities of computer hardware and standards. The standard Web1.0 used dial-up connection and 50K average bandwidth during 1991 to 2003 which does not support for Video Service as this standard supports only static web pages. The advent of Web 2.0 from 2004 has drastically changed the way in which people see and use Internet as it supports two way communications. The Web 2.0 uses an average of 1Mb of bandwidth [7]. The distributed application development using Java and related tools and implementations evolved with many approaches, specifications and relevant APIs from SUN, Open standards like OMG, XML-RPC, Oasis, Apache and many open groups and forums [1]. As there are multiple standards available, the developers will always have a dilemma in selecting the architecture to work with. The confusion is even higher when compared to provide multimedia using different JAVA platform. This research paper aims to develop a video on demand service using JAX-RPC and its performance is analyzed to throw light to the developers working with video files.

This research paper is organized in the following ways: The first part of the paper putforth the JAX-RPC model adopted for the video on demand service. In the second part, the pipeline for developing the JAX-RPC model for video on demand service depicts the methodology for providing video service in real time. Following which, the research paper elaborates on the performance analysis of the developed JAX-RPC model. In the final part, the results of the performance analysis of various video file with different file sizes are reported in two different scenarios namely local and remote. The significance of this paper and the scope for the future research is revealed in the conclusion.

2. JAX-RPC Model for Video on Demand

The architecture of the proposed JAX-RPC model for video on demand service model is given in Figure 1. The client using the VideoClient Application creates a request that contains all the details required by the video implementation. This request is appended as a SOAP message context for further processing.

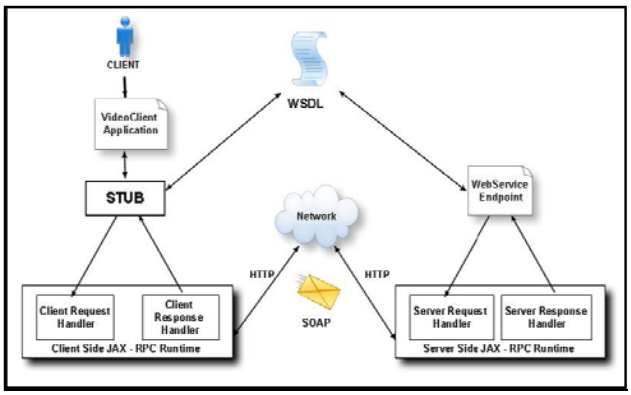

Figure 1: JAX – RPC Architectural Model for Video Service

The JAX-RPC Runtime Environment invokes the Client Request Handler to process the request, by checking whether all the required attachments are available and if all the required information is available then it sends to the video implementation through the SOAP Request Message. The WSDL file contains all the methods that the generated stubs can invoke. The generated stub in the client side has all the required details about the Web Service end point. Thus the client through the VideoClient invokes the method in the Web Service endpoint. This model solves the connectivity issues and interoperability.

As the developed JAX-RPC model for video on demand service uses the dynamic proxy, therefore the stub referring to the specific endpoint is not required. The dynamic invocation interface dynamically access the service endpoints. At the server side, the JAX-RPC run time environment, invokes the server request handler to handle the received SOAP message. This checks for the attachment inside the SOAP message and if available, it repackages to the SOAP message context and forwards it to the service endpoint. The response from the web service end point is sent to the client following the same procedure.

The issues that have carefully considered during design and implementation of JAX-RPC architecture includes the reliability, performance, and cost of the system. Web Services are essentially real time processes in which performance and availability problems have a high cost. The developed JAX-RPC model for video on demand service provides a guaranteed quality of service.

3. Implementation of JAX – RPC Model For Video Service

The video on demand service is adopted to illustrate the pipeline of JAX-RPC Web Service and the developed model is tested under local invocation and remote invocation. Java based XML-RPC model has been developed to deploy Video on Demand services in a distributed environment. It is examined how to use JAX-RPC system to publish Video on Demand Web Services and how to use this system on the client to consume VoD services. The getVideo method accepts the video file name as its parameter from the user and returns the video.

3.1 Web Service Endpoint Interface for Video Service

The interface for video on demand service consists of the declaration for the getVideo method to be invoked from the remote reference by users. This method invokes the RemoteException and the VideoInt class by itself extends the Remote interface of the rmi package. The following web service endpoint interface "VideoInt" is shown below:

```
public interface VideoInt extends Remote{
  public byte[] getVideo(String fname) throws
                                    RemoteException;
}
```

This VideoInt defines a method "getVideo" that accepts the video file name to be fetched and returns the byte array of the video.

3.2 Implementation for Video Service

The video on demand service is implemented in a remote entity. As it is an instance of the VideoInt Interface class, it implements the video on demand interface. The definition for the getVideo method declared in the interface program is defined here.

```
public class VideoImpl implements VideoInt{
   byte [] buffer;
   public byte[] getVideo(String fname) throws
                                    RemoteException {
      // Accept the Video File and Returns the
         Video Byte Array
     return buffer;
    }   }
```

This getVideo method is being triggered by the clients through the VideoInt Interface to fetch the required Video.

3.3 Configuration for Video on Demand Interface

The mapping file consists of all the data required to associate the VideoInt Interface to the definition of wsdl file. This mapping file and "VideoService" wsdl file which is used to generate the client side stubs are generated by defining and compiling the configuration file for the VideoInt interface. This wsdl file will have all the necessary details like the location of the web service, purpose of the web service and by what approach will the method be invoked. In detail, it contains of the datatype of the messages, the message itself which is sent from one endpoint to the other, the portnumber at which the service is operating, the input, the output and the fault details, the necessary binding information like which protocol will be used to carry the message and at the end, the service details are generated.

The definition of the configuration file includes the name of the service, various NameSpace, the Package Name and the details of the video service interface. This generated "VideoService" wsdl file acts as the tie between the web service end point and the JAX-RPC Runtime. The following is the configuration file for the VideoInt interface.

```
<configuration
   xmlns="http://java.sun.com/xml/ns/jax-
rpc/ri/config">
   <service name="VideoService"
     targetNamespace="urn:video"
     typeNamespace="urn:video"
     packageName="video">
   <interface name="video.VideoInt"/>
   </service>
```

As the video service implementation (VideoImpl) returns the byte array, the generated wsdl file will return the datatype as xsd:base64Binary. The xsd represents the XML schema definition datatype. The JAX-RPC provides the direct mapping between these datatypes. The following code shows a portion of the generated wsdl file where it clearly denotes that the datatype of the input is a string, which is the video filename and the output is the bytearray.

```
….
<message name="VideoInt_getVideo">
   <part      name="String_1"      type="xsd:string"/>
</message>

 <message name="VideoInt_getVideoResponse">
   <part   name="result"   type="xsd:base64Binary"/>
</message>
….
```

All the necessary files are packaged into war file and the same is deployed. After deployment of the service, the wsdl file in the server is ready for the remote object to access it. The tie class required to communicate with the clients are generated by the server during deployment process. The video service definition language can be viewed by requesting the following URL: http://192.168.1.2:1078/myvideo/VideoStream?WSDL

3.4 Generating the Dynamic Proxy

The remote reference will trigger the getVideo method through the stub. The client side stub is acquainted with the service offered by the remote interface. As the stub is available with the client, the wsdl file which contains all the required details about the service is not required for the client. The stub for the video on demand service is generated by defining and compiling the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
<wsdl location="http://192.168.1.2:1078/myvideo/
   VideoStream?WSDL"  packageName="dynamicstub"/>
</configuration>
```

The configuration file contains the location of the wsdl file deployed in the above step. On compiling the configuration program, the dynamic stub is created using the VideoImpl and other required runtime entities like serializers and type value. Now, the client will be able to invoke the getVideo method using this dynamic stub. During runtime, the client creates the dynamic proxy stubs using the rpc.Service interface. This stub has the knowledge of the video service and the WSDL file.

3.5 Fetching the Video

A service end-point interface is created to encapsulate fetching video service and Real Time Streaming service. The Service End-point Interface for the proposed JAX-RPC based video service model is given below:

```
Stub stub = (Stub)(new
 VideoService_Impl().getVideoIntPort());

stub._setProperty(javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
"http://192.168.1.2:1078/myvideo/VideoStream");

VideoInt client = (VideoInt)stub;
 resp = client.getVideo(request.getParameter("VideoFileName"));

//convert the video file data into byte array - buffer
if (resp) {
   client.RTS_Service(buffer);
}
```

A Service class is created which is effectively a factory for calling the object, which in turn encapsulates the mechanisms to call a VoD Service Endpoint.

4. Test Cases And Results

The developed JAX-RPC model for video service is implemented in a distributed environment. It works for different file formats and for different sizes. The following Figure 2 shows the requested video which is streamed as a

response to the client.

Figure 2 : Client Response

5. Performance Analysis

The performance measure, Round Trip Time (RTT) is estimated for the developed JAX-RPC video service model by requesting the Video files of various size in local and remote environment. RTT is the time that elapses between the initiations of a getVideo method invoked by the video requestor till the specified video is played to them. To analyse the performance exactly, the getVideo method was invoked 10 times for each video file, and the average of these were considered for analyzing the final results.

5.1 Local Invocation Performance

To identify the RTT for fetching video files of different sizes locally, the server and the client was executed in the same system. The below Figure 3 shows the average Round Trip Time in ms obtained for different sizes of video files starting from 1MB to 30MB.

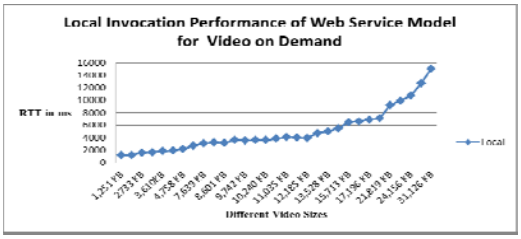

Figure 3 : Local Invocation Performance of Web Service Model for Video on Demand

It is very simple and easy to create and maintain the web service model for video on demand using JAX-RPC. The RTT increases as the size of the video file increases. The below Figure 4 shows the Local Invocation performance for web service model for video on demand for different video sizes starting from 30MB to 75MB .

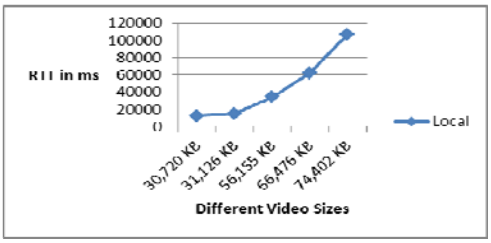

Figure 4 : Local Invocation Performance of Web Service Model for Video on Demand

The RTT is very high for large video files. From, the above, it is clear that the web service model for video on demand has better performance for smaller video files than large files.

5.2 Remote Invocation Performance

The performance measure, Round Trip Time (RTT) is estimated for the developed JAX-RPC video service model by requesting the Video files of various size from remote System. Two computers connected in a LAN with 100Mbps speed with similar hardware configuration of Dual Core Processor at 2.00 GHZ, and 2 GB RAM was used for testing the remote invocation. The below Figure 5 shows the Round Trip Time obtained for different sizes of video files starting from 1MB to 30MB invoked locally and remotely.

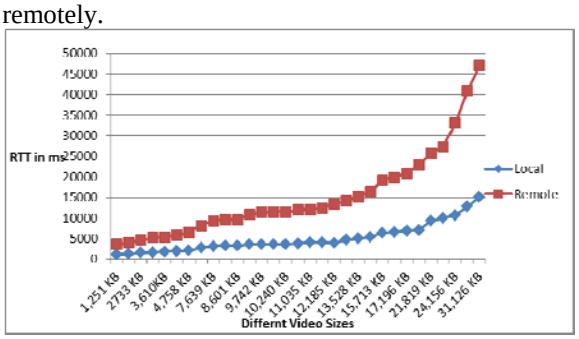

Figure 5 : Comparison of Performance of Web Service Model for Video on Demand in both local and remote scenarios.

When the Video files were invoked remotely, the connectivity and the interoperability was not an issue as

the JAX-RPC model uses the XML-based SOAP communication between remote services. It is clear from the graph that the RTT is very less for smaller video files but still higher than the local performance. The below Figure 6 shows the comparison of performance of web service model for video on demand in both local and remote scenarios for larger video files. The performance of JAX-RPC model for video on demand is better for video files with small size in both local and remote invocation.

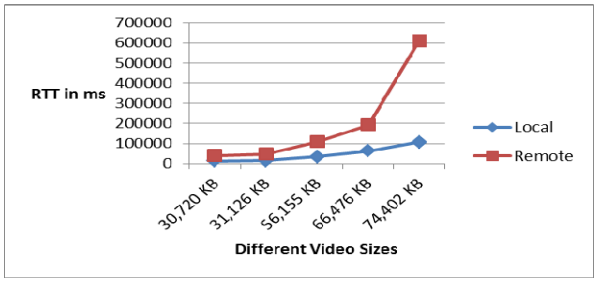

Figure 6 : Comparison of Performance of Web Service Model for Video on Demand in both local and remote scenarios.

When comparing the local performance results of individual approaches with remote invocation strategy, the factor of variation is also calculated and shown below in Figure 7.

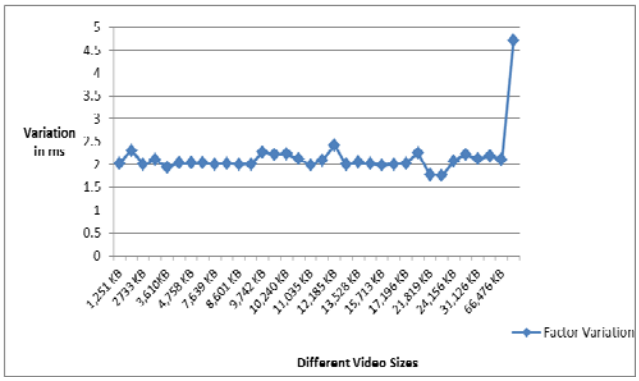

Figure 7 : Factor variation between Local and Remote invocation of web service model for video on demand.

The above result shows that very little variation between the performance of local invocation and the remote invocation.

6. Conclusion

Though webservices bring more flexibility in different sectors, there raises new challenges for the multimedia files especially video files. Thus, the webservice model is created using Java and deployed in J2EE deployment tool. The JAX-RPC was selected to develop the Video on Demand model, thereby solving the connectivity issues and interoperability. Thus when the clients request the video to the server, the server fetches the same to the client. The performance was better for smaller video files when compared with the larger files and there was very little variation obtained between the local and remote invocation. In the future study, to increase the security issues the SAAJ model for Video Service can be adopted.

References

[1] D. Jagannadham, V. Ramachandran and H.N. Harish Kumar, “Java2 Distributed Application development (Socket, RMI, Servlet, CORBA) approaches, XML-RPC and Web Services Functional analysis and Performance comparison”, “International Symposium on Communications and Information Technologies”, 2007

[2] Developing Web Services for Existing Java Applications, “Mind Fire Technologies” at “http://www.mindfiresolutions.com”, 2003.

[3] Hovedoppgave , “E-wallet decentralized credential keepers”, European Research Project CAFÉ,, 2003.

[4] Huang, Cheng, Jin Li, and Keith W. Ross, "Can Internet Video-on-Demand Be Profitable?", SIGCOMM, Vol. 2 No. 4, 2007.

[5] Johnson P Thomas, Mathews Thomas, George Ghinea, “Modeling of Web Services Flow”, “Proceedings of the IEEE International Conference on E-Commerce”, 2003

[6] Jong, Alex De, Karen Hsing, and David Su, "A VoD Application Implemented in Java", Multimedia Tools and Applications Vol. 5 No. 2 pp. 161-70, 1997.

[7] Reed Hastings, Retrieved May 26, 2010, “http://roseindia.net/Technology-revolution/web3.0/history-of-web-3.shtml”,2006.

Lavanya Rajendran received her Bachelor’s degree in Computer Applications. She completed her M.Sc. in Electronic Media and Informatics. She is currently pursuing her research in College of Engineering, Anna University, Chennai. Her areas of interest include Cloud Computing, Computer Networks, Web Designing and Video Production.

Ramachandran Veilumuthu received his Masters degree and Ph.D. in Electrical Engineering from College of Engineering, Anna University, Chennai, India. He is currently working as a Professor in the Department of Information Science and Technology, College of Engineering, Anna University, Chennai. His research interest includes Cloud Computing, Network Security, Soft Computing and Web Technology.