CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 15, 2017

COVERAGEJSON RESPONSE HANDLER FOR OPENDAP

PREPARED FOR

NASA JET PROPULSION LABORATORY

LEWIS JOHN MCGIBBNEY

PREPARED BY

GROUP55

RIVER HENDRIKSEN

Abstract

This document reviews the possible technologies choices that can be used for the CoverageJSON Response Handler for OPeNDAP project.

1 TECHNOLOGY: HOSTS

1.1 Overview

1.1.1 Hyrax Data Server

1.1.2 AWS

1.1.3 Local Host

1.2 Discussion

1.3 Conclusion

2 TECHNOLOGY: CODING LANGUAGES

2.1 Overview

2.1.1 C++

2.1.2 Python

2.1.3 Java

2.2 Discussion

2.3 Conclusion

3 TECHNOLOGY: USER INTERFACE

3.1 Overview

3.1.1 Hyrax

3.1.2 Web interface

3.1.3 Desktop

3.2 Discussion

3.3 Conclusion

1 Technology: Hosts

1.1 Overview

Hosts are one of the most important parts to consider when implementing a tool that is expected to be usable by users across the web. Hosts, defined in this context as webservers and their OS's will hold the opendap protocol code that will be called when a user makes a request for the coverage JSON. For this reason the server must be accessible, easy to access, and be able to handle many requests. The hosts we will talk about are Hyrax servers, AWS and local hosting.

1.1.1 Hyrax data Server

The Hyrax data server architecture is a design that is currently implemented with OPeNDAP. The purpose of Hyrax is specifically to allow for the OPenNDAP protocol and the handling of the requests from a web daemon to DAP software services. Hyrax offers both a client interface (see the NASA JPL website for an example of this), and web daemon backend for users to make requests. This is done via Java servlets, which are used to take in requests. The front end OPeNDAP Lightweight Front end Server(OLFS) will take in requests and push it to the second server, this can be the same server, called the Back End Server (BES).

1.1.2 AWS

AWS, or Amazon Web Services, is one of the most widely used web services in the world. AWS is versatile in its implementation allowing for any type of service to be stored on their platform, this can mean data server to red hat server. Think of AWS as a cloud computing platform rather than a specific server, since the applications allowed are so wide.

1.1.3 Local Hosting

Local hosting is less of a specific server and more an option for where to host. Having a home server, or one physically accessible, allows for customization of hardware and software.

1.2 discussion

There are differences and similarities between all these options, so as to give them all fair discussion we will discuss the pros and cons of each. Hyrax servers have the obvious pro of being already implemented with the OPeNDAP protocol, allowing less time being allocated to creation of a request processor. The con's of Hyrax are that the current implementation is complicated and a lot of time would be spent having to understand the current systems. AWS has the benefit of being one of the most popular we server service in the world this would help with scalability and accessibility to users across the globe. Cons of AWS are that it does not have the OPeNDAP integration and it would be up to use to set up handlers and processing of requests. Local hosting has the most obvious pros and cons, pros being that being closer to the server allows for customization and easy debugging of hardware and software. However there are many cons, ISP's generally do not take kindly to business applications being implemented on domestic subscriptions, the system would not be scalable, and we would have to implement our own handler requests.

1.3 Conclusion

The obvious benefits of Hyrax make it the clear winner. There will have to be resources dedicated to understanding the architecture that OPeNDAP created, but this is far less time then making a different handler. Not to mention that if the

CovJSON handler had a different handler than the request of other data formats that are already implemented it would fragment CovJSON and make it less likely to be adopted.

2 Technology: Coding Languages

2.1 Overview

It is well known among programmers that all coding languages have their own benefits and quirks. Languages are robust but some are specifically better for some things than others, for example javascript is almost universally used for web design but would not be used to manipulate data in a server. For the case of OPeNDAP there are options for what language to use, the three to be discussed are Java, Python, and C++.

2.1.1 Java

Java is an object oriented coding language that is robust in the sense it can be implemented in any platform. Java is popular in client-server connections generally on the server side. In the case of OPeNDAP it is used for the Hyrax server which is going to be used. The popularity of Java can be seen in its use in the Android SDK, the most widely used mobile operating system in the world.

2.1.2 Python

Python is an interpreted coding language that is useful in teaching and implementing non-time dependent code. That is not to say that python cannot handle time, more so the interpretation of the code can cause slow run times. Python is useful in that it is easy to share libraries and modules between users. This means that if someone has written a function that you need you can look for it and implement it quickly, rather than having to rewrite it yourself.

2.1.3 C++

C++ is similar to Java in that it is an object oriented coding language that works on most hardware. C++ is an extension of the C coding language which makes up most all operating systems today (besides windows). C++ is a standard in many systems and is ideal for data server manipulation, it makes it easy to interact with SQL data coding language.

2.2 Discussion

Again we will go into a discussion of the pros and cons of each language. Java has the pro of being what is currently implemented with the Hyrax server on the OLFS, and being object oriented it is easier to work with the request information. The Con of Java is that it doesn't work well when interacting with data sources. Python has the benefit of CovJSON interpreter already being implemented within it, however python is slow and doesn't work well as a server. C++ has only pro's as it fast, object oriented, and can manipulate data from a source.

2.3 conclusion

C++ is the best choice in this case, for one it the required language based on the prerequisites for the project. C++ has many advantages, namely its ability to manipulate and call SQL. This project is based around the manipulation of data, so this feature is imperative.

3 Technology: User interface

3.1 Overview

While user interfaces are not necessarily a technology, they are an integral part of how users interact with a tool. For this project there is both a UI and back end portion, for this section we will not talk about the back end since it is already predetermined.

3.1.1 Hyrax

Hyrax server has, by default, a simple user interface that a developer can manipulate to have specific options for the user to enter data. An example of this UI is on the NASA JPL website. It is not the prettiest UI but is utilitarian in its function, the data is generally being used by scientists who do not need to pretty interface.

3.1.2 Web interface

Another option is to avoid the Hyrax methodology and go for a more streamline web interface. This would include javascript to make the page more dynamic. This would help to get non-scientist interest, as they would not have to use a webpage that looks like it is from the early 90's.

3.1.3 Desktop

A final option is to create a desktop application that a user can download. This would be helpful if a user was constantly needing to access data that NASA JPL has. This would also help in creating a cohesive environment that is dedicated to OPeNDAP.

3.2 Discussion

The pros of the Hyrax server are obvious, it is already implemented, this has already been discussed ad nauseam in other sections so it will be skipped here. A unique web interface would be helpful in interesting a wider user base to use the data that NASA has, however it would take much more time. A desktop app in theory is a good idea, but would be too hard to implement in the time provided.

3.3 Conclusion

We will use the Hyrax server implementation for the reasons that were discussed in section one. While not necessarily the best implementation of a UI, it is the one that is required for this project. There is not much to say beyond that.