Generalized Software Interface for CHORDS

Pattaphol Jirasessakul, Zachary Waller, Paul Marquis,
Vinh Le, Connor Scully-Allison, Scotty Strachan, Frederick C. Harris, Jr. & Sergiu M. Dascalu
Department of Computer Science and Engineering, University of Nevada, Reno
Reno, Nevada, 89557, United States of America
(pjirasessakul, cscully-allison)@nevada.unr.edu,
(zacharydwaller, paul.marquis1, vdacle)@gmail.com, scotty@dayhike.net,
(fred.harris, dascalus)@cse.unr.edu

Abstract

In the physical sciences, the observation and analysis of environmental readings, such as wind speed, sap flow, atmospheric pressure, temperature, and precipitation, benefit greatly from real-time visualization as they allow environmental scientists to create faster actionable intelligence. However, the scarcity of easily accessible and customizable real-time visualization software often creates logistical problems for researchers focused in environmental sciences. The goal of this paper is to present an alternative approach, based on usability and open source software, for the Nevada Research Data Center (NRDC) to visualize environmental data in near-real time and confirm its viability for usage with other research projects of similar size. This approach involves creating an open-source near-real time interface to act as middle-ware between the NRDC's data repository and CHORDS, a cloud-hosted data visualization package. While this interface is primarily being built for the NRDC, there is an emphasis on tooling it to be as generalized and generic as possible.

keywords: Data Visualization, Environmental Science, Middleware, Web Scraping, Web Service

1 Introduction

Data visualization is a critical tool for scientists working with large and constantly updating data streams. However, these scientists, usually of a physical science background, are often presented with two options for robust visualization: expensive proprietary solutions, or programming language libraries that require developer-level knowledge to use.

This is where Cloud Hosted Real-Time Data Services for the Geosciences, or CHORDS, comes in. CHORDS is a project that was developed by EarthCube, an NSF-funded project that supports the development of cyberinfrastructure for the Geosciences. CHORDS provides a visualization platform tailored for environmental research to make real-time data available to the research community in standard formats[7].

However, CHORDS by itself comes with a very critical limitation that makes the use of a middle-ware vital for operation in larger projects, such as the ones hosted at the NRDC. CHORDS provides an HTTP API that allows for the population of real-time data entry, but the API is only partially exposed and often forces the task of manual inputting new hardware and stream deployments onto the user. This is especially tedious for larger environmental research groups because the addition of new sensor equipment is not an uncommon occurrence during a multi-year operating period.

The middleware described in this paper first interfaces with the NRDCs own HTTP library to access the data stores. This then automates the creation of a CHORDS portal is created, if not already existing, and allows users to specify the data to be stream based on the way the data is structured. The middleware compares the amount of the datastreams being monitored and automates the creation of new ones based on the metadata provided by the NRDC. The data is then streamed in near real-time to CHORDS, and users may visualize their data in standard formats while provided complex scientific tools such as prediction models. Preliminary results of a discussion indicate that this middleware represents a satisfactory solution to the NRDC's problem of data visualization.

The remainder of this paper is structured as follows: Section 2 introduces a basic background of the project and some related works, Section 3 goes into the specifications of the software, Section 4 discusses the overall design of the software, Section 5 contains details on the UI design of the web client, Section 6 and 7 includes details of the prototype development and discussion about viability, finally Section 8 details conclusions and future work.

2 Background & Related Work

This project was made in coordination with the NRDC and EarthCube, both of which falls under the umbrella of the Cyberinfrastructure branch of the National Science Foundation (NSF). The following section will go over the goals of both organizations as well as a more detailed explanation on other visualization options inside and outside of the Nexus Project.

2.1 NRDC

The NRDC was born out of a data portal that was developed during a previous Track 1 NSF EPSCoR project on climate change called the Nevada Climate Change Portal. The current project that the NRDC is affiliated with, the Solar Energy-Water-Environment Nexus, was created in order to increase research awareness, and productivity of alternative energy sources, and the conservation of natural resources in the state of Nevada. The NRDC serves in a critical role of cyberinfrastructure within the Nexus Project, which includes the provision of technical skills and resources to members of the research project. The tasks that the NRDC covers include the acquisition, transport, storage, querying, and dissemination of observational data gathered by automated digital sensor systems. The NRDC participates in cutting-edge software and systems development to enhance next-generation science that leverages the Internet of Things (IoT). Their goal is to transform the scale, quality, impact, and bottom-line cost of research projects in Nevada that seek to deploy automated sensor systems as part of their scientific workflow[3].

2.2 EarthCube

EarthCube is a quickly growing community of scientists across all geoscience domains, including geoinformatics researchers and data scientists. They are a joint effort between the NSF Directorate for Geosciences and the Division of Advanced Cyberinfrastructure. EarthCube was initiated by the NSF in 2011 to transform geoscience research by developing cyberinfrastructure to improve access, sharing, visualization, and analysis of all forms of geoscience data and related resources. As a community-governed effort, EarthCube's goal is to enable geoscientists to tackle the challenges of understanding and predicting a complex and evolving solid Earth, hydrosphere, atmosphere, and space environment systems. The NSF's Directorate for Geosciences (GEO) and the Division of Advanced Cyberinfrastructure (ACI) partnered to sponsor EarthCube, which NSF anticipates supporting through 2022[2].

2.3 CHORDS & Other Visualization Options

Currently, there are a handful projects within the NRDC that utilizes data visualization: VISTED and VFire. VISTED (Visualization Tool of Environmental Data) is a web application, which enables data selection, extraction, download, conversion, and visualization of environmental data sets that extends for over 30 years (1980 - 2009)[9]. VFire is an immersive visualization application that uses remote sensing data in conjunction with simulation model to predict the behavior of wildfires[8]. RWWSS (Real-time Web-based Wildfire Simulation System) is a web application that provides users with wildfire simulations using data from the Lehman Creek Watershed in Great Basin National Park [11]. A workflow dedicated to visualizing big data on web applications was created as an alternative to expensive third-party software[12]. Finally, a system revolving around MongoDB and some accompanying tools were developed to visualize big data as a way to address the mass influx of data in the recent years[10].

Aside from CHORDS, there are also alternatives out for near real-time data visualization. There are multiple programming languages out there with data visualization library along with existing proprietary data visualization softwares. Libraries such as D3.js and Plotly.js, are well known library within the field of data visualization. D3.js is a library made for visualizing data using web standards. It combines powerful visualization and interaction techniques with data driven approach to give users the freedom to design the visual interface anyway they want . Plotly.js is open-source library that supports many chart types including scientific ones such as heatmaps and contour plots to use for plotting sensor data in real-time[4][1]. Unfortunately, both of these and other libraries suffers from the same problem as they require some sort of programming knowledge on their respective programming languages. This results in lower accessibility of these libraries for smaller research team as they may not have someone with the programming knowledge in the team or have the time and patience to learn the language and library by themselves.

Alternatives to using programming libraries would be to use real-time data visualization softwares. Examples of these softwares are Tableau and Visualr, both offer many features such as the ability to connect with multiple data sources such as MySQL, Oracle and MS Excel, being able to fetch data from API Data Providers and a plug and play feature where all the users have to do get the software running is to install it[5][6] . However, as compensation for having many features,

they often come with a hefty price tags along with some sort of training session in order to use them effectively.

3 Software Specification

The main requirements of this project were elicited from the main stakeholder, Scotty Strachan, a domain scientist specializing in in-situ environmental sensing and data management. These requirements were split between functional requirements, which describe the overall technical functionality of the system, and non-functional requirements, which outline constraints on the system.

3.1 Functional Requirements

From formal interviews with Dr. Strachan, we elicited seven base level functional requirements that form the core functionality of our solution. The first and second requirements are to create an interface that will not only be able to successfully interface with a running instance of CHORDS and manage the data inside, but to also interface to the research team's data source in the NRDC in order to query data. The third requirement involves implementing the functionality to visualize data displayed on CHORDS in near-real time. The fourth requirements is streamlining the user experience by creating a web client to simplify the originally tedious visualization process for users. The NRDC sensor networks currently exists in a structured hierarchy, so the fifth requirement is to fetch that hierarchy and format it into integrates intuitively with the user interface. Finally, the sixth and seventh requirements is the functionality that allows users to specify whether they wish to stream data in a near-real time mode, or stream from a specific date range.

There are four higher level functionalities that this solution provides outside of the scope of the main functionalities. The first requirement let users compute and display summary statistics, such as the minimum, maximum, mean and standard deviation, of the data streams that they chose for their visualized session. The second requirement enables users to share their visualized session by adding in the functionality to export a snapshot, which is an interactable instance of the users visualization at the time it was created. The third requirement enables users the option of having the visualized instance alert them through email when the data leaves an expected range. The fourth requirement builds upon the web client by embedding a customized Google Maps API onto it. The map will list all available sites in NRDCs hierarchy network that represented as marker on the map. Additionally, when the user clicks on a marker, the latest photo streamed from that site

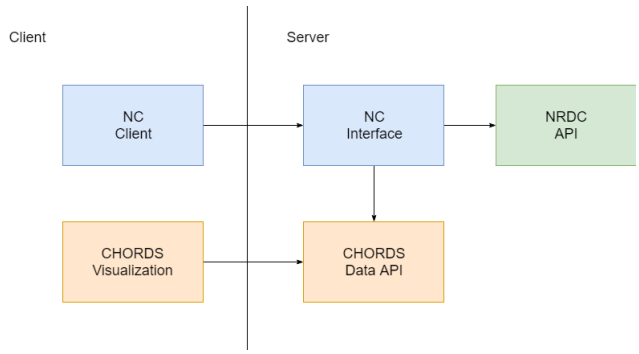

Figure 1: A diagram showing the high level design of the project.

will be displayed along with specific information about that site, such as name, latitude, longitude, and current measurements.

3.2 Non-Functional Requirements

This project operates under 4 non-functional requirements acting as constraints on the design and development of this system. The first requires that the interface portion of the software be written in C# with the .NET Web API library as its framework. The second requirement is that the development team uses a modified instance of the CHORDS visualization package for the visualization aspect. The third requirement is that the software should be compatible with all major web browsers. The fourth requirement is that the software should be able to consistently maintain near-real time execution when streaming data from sensor instruments.

4 Software Design

This project consist of three major parts: NC-Client, NC-Interface and the Chord's visualization. The goal of this section is to show, in detail, the design of this software and everything that is required to produce a similar design. Beginning with a high level explanation of each component and delving into how of each of these components interact with each other. The architecture of this project is component based to ensure fast and robust development as well as strong interoperability as each components are loosely coupled. A high level design of the project can be seen below in Figure 1.

4.1 Components

NC-Client - The NC-Client consists of two web pages and a scripts file. The script contains code to drive the spawning of views and navigation logic for

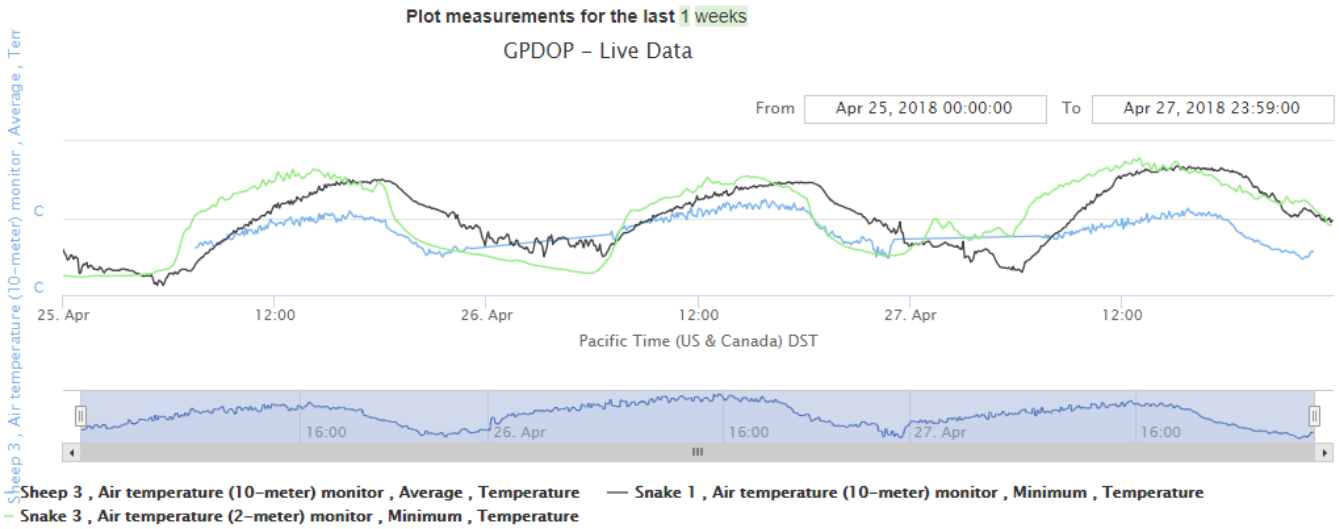

Figure 2: An example of a visualized session of 3 different data streams on a CHORDS instance

each web page on the web page. It also impements an auto-refreshing function call to allow for near-real time streaming.

NC-Interface - The NC Interface is the main component of our project. It contains four modules: ChordsBot, DataCenter, GrafanaManager, and SessionManager. Three of these four modules also have a Web API controller associated with them. Each of these modules are further explained in section 6.

Chords Visualization - The Chords Visualization is the main component of visually and actively interfacing with the data from the NRDC repositories. After a user has made a selection of a desired datastream using the NC-Client, the NC-Interface will fetch that data, reformat it, create the users CHORDS session (via SessionManager) and then push the data to the newly created session with ChordsBot. Figure 2 shows an example of a visualized CHORDS session.

5 UI Design

There are two primary interfaces users interact with when using the *Generalized Software Interface for CHORDS*: The interface web client and the CHORDSs interface. And, although functionality was implemented to integrate with Graphana, we cannot include it here due to space limitations.

5.1 Interface Web Client

Our custom built web client, visible in Figure 3, is a single-page-web-application allowing users to view the sensor network hierarchy, select a deployment, and begin streaming data. This feature enables users to specify the type of data they want to stream.

The NRDC sensor networks exist in a hierarchy. Each sensor network (NevCAN, Solar Nexus, Walker Basin Hydro) contains a list of sites, which refer to geographic locations. Each site contains a list of systems, which are logical groupings of deployments or sensors. In order to make it easy for a user to access specific deployments to view their data on the user interface, we implemented a way to retrieve and display this entire hierarchy on said interface. This requires our web interface to make calls to the NRDCs Infrastructure API and format the data returned in a user-readable format. By implementing this feature, it makes accessing specific data streams much easier for the user.

Upon visiting the client page, the user can select between the three sensor networks. Then, the user can select which site they want to see data for from a list of all sites in that sensor network. Next, the users select which data streams they would like visualized. The user can select one stream or multiple. Finally, the user can save their session with a name and specify the time period for which they would like the data streamed. Leaving the end date of the stream blank will result in a continuous live data stream.

5.2 CHORDS's Interface

CHORDSs UI primarily functions on the back end of our software by creating new CHORDS instruments for user created data streams. At the top of the CHORDS page, the name the user chose for the session in our web client is displayed as the name of the CHORDS instrument, along with the total number of measure-

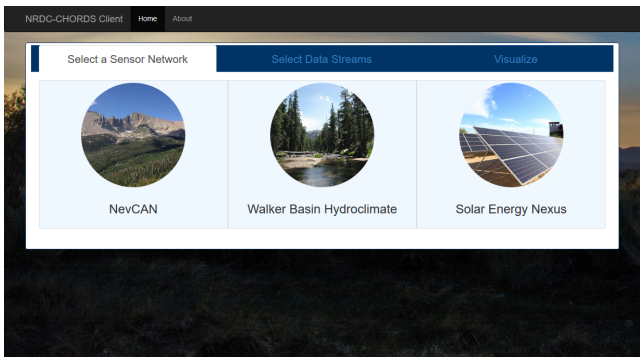

Figure 3: The main page for the NRDC-CHORDS-Interface Web Client. A user can begin finding a datastream to visualize by clicking on one of the three available site networks associated with the NRDC.

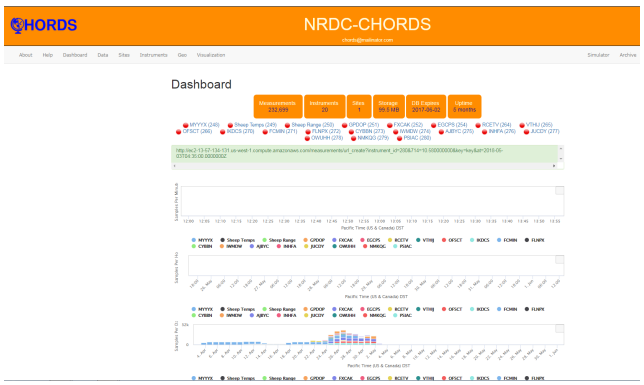

Figure 4: A picture of a research team's CHORDS's Portal. Currently displayed is a list of the team's currently visualized session

ments reported and the dates of those measurements. Additionally, a list of all visualized session created by a research team is available to them as well as seen in Figure 4.

The main section of the visualization page displays the actual graphed data from the data stream displayed alongside of the names of the data streams and above the times that the data was received by CHORDS like the one seen in Figure 2. Below, each variable corresponding to a selected data stream is displayed. For each variable, the user is shown the units of the variable, the property measured, and the name of the variable, which is a combination of data about the stream including the location of the sensors, what the sensors are measuring, and other information.

6 Prototype Development

A prototype of this project was developed as a proof of concept for the NRDC. It implemented the majority of the functionality detailed in Section 3. On the server side, we created a service called NC Interface. It acts as an interface between the NRDC's data center and CHORD's Data API so that data can be gather from the former, formatted and sent to the latter. On the client side, we created a single page web application called NC Client that allow users to select the specific data stream they want to visualize out of the NRDC's sensor network hierarchy.

The NC Interface was developed using the .NET Web API framework. It is composed of 4 different modules: ChordsBot, DataCenter, GrafanaManager and SessionManager. When the user first opened up the web client, the DataCenter module is called to fetch the NRDC's sensor hierarchy network for the user to select their data streams. Once the user select their stream(s), that information is sent to SessionManager that will create the user's session on the research team's CHORDS Portal. Once the session is created on CHORDS, ChordsBot is called to automate the data streaming to the session along with performing other functionalities like filling forms in the session with information regarding the selected data streams (their name, location, units of measurements, etc.) and generating the session's Grafana dashboard. Automated tests are performed in order to confirm that DataCenter was getting the correct data and that ChordsBot was performing the functionalities that it was automating for the user on CHORDS properly.

CHORDS's Data API was very inflexible in terms of what our development team wanted from it. While certain things like data put and fetch activities are well documented on the API, functionalities like the automation of Sites and Instrument creation on a particular CHORDS instance are not. From our communication with the API's developer, we have learned that since the API was developed using Ruby on Rails, a lot of its functionality are written for them which doesn't give their development team a lot of room to formalize the API.

7 Discussion

Our software was successful in visualizing data from the NRDC database in a way that was configurable by users. While CHORDS was intended to work with static sets of data sent to it, our software was configured to be able to send CHORDS live updates as the data enters the database. To our knowledge, this has not been done using CHORDS specifically, yet our software allowed for both live streams and static viewing of data. Our software was able to take the streams of data from NRDC Databases, place that data in a CHORDS-

compatible data structure and send chunks of data to CHORDS via HTTP.

While many data visualization solutions on the market support live streaming of data, most come at a high price that limits their availability to those outside of industrial applications. Our service is open source and could be adapted to work with systems other than the NRDC database, which could allow for greater availability of live-streaming data visualization.

8 Conclusion & Future Work

This software was intended to act as a middleware between data coming in from the NRDC database and the CHORDS data visualization service. It performed that function successfully, sending data to CHORDS in a format that could be interpreted and displayed to the user.

Using C#, the program stored the data from the NRDC sites, deployments, and data streams into data structures and processed those data structures to be compatible with CHORDS. Using HTTP and a variety of services like Selenium and PhantomJS, the program interfaced with CHORDS and sent the data successfully even though CHORDS lacked a functional API to input data.

With some modification this software could be used as an open-source data visualization solution for labs that cannot afford more expensive software. This software can also help those who are not knowledgeable enough at programming to interface their database to CHORDS. Additionally, there are plans for the software to be used by the Desert Research Institute to monitor data incoming from lysimeters. This software has great potential to help many people visualize their data.

9 Acknowledgements

This material is based upon work supported by the National Science Foundation under grant number IIA1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] D3.js. `https://d3js.org/\#introduction`. [Online; accessed November 18, 2018].

[2] Earthcube. `https://earthcube.org/info/about`. [Online; accessed November 18, 2018].

[3] Nevada Research Data Center. `www.sensor.nevada.edu`. [Online; accessed November 18, 2018].

[4] Plot.ly. `https://plot.ly/javascript/`. [Online; accessed November 18, 2018].

[5] Tableau. `https://www.tableau.com/2018-1-features`. [Online; accessed November 18, 2018].

[6] Visualr. `https://visualrsoftware.com/features.html`. [Online; accessed November 18, 2018].

[7] MD Daniels, SJ Graves, B Kerkez, V Chandrasekar, F Vernon, CL Martin, M Maskey, K Keiser, and MJ Dye. Connecting real-time data to algorithms and databases: Earthcube's cloud-hosted real-time data services for the geosciences (chords). In *AGU Fall Meeting Abstracts*, 2015.

[8] Roger V Hoang, Matthew R Sgambati, Timothy J Brown, Daniel S Coming, and Frederick C Harris Jr. Vfire: Immersive wildfire simulation and visualization. *Computers & Graphics*, 34(6):655–664, 2010.

[9] Likhitha Ravi, S Dascalu, Frederick C Harris, John Mejia, and Noureddine Belkhatir. Visted: a visualization toolset for environmental data. In *Proceedings of the 2015 International Conference on Computers and their Application (CATA-2015)*, pages 335–342, 2015.

[10] Rui Wu. *Environment for Large Data Processing and Visualization Using MongoDB*. University of Nevada, Reno, 2015.

[11] Rui Wu, Chao Chen, Sajjad Ahmad, John M Volk, Cristina Luca, Frederick C Harris, and Sergiu M Dascalu. A real-time web-based wildfire simulation system. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pages 4964–4969. IEEE, 2016.

[12] Rui Wu, Jose T Painumkal, Nimrat Randhawa, Lisa Palathingal, Sage R Hiibel, Sergiu M Dascalu, and Frederick C Harris. A new workflow to interact with and visualize big data for web applications. In *Collaboration Technologies and Systems (CTS), 2016 International Conference on*, pages 302–309. IEEE, 2016.