

Simplifying Data Visualization Pipelines with the NRDC-CHORDS Interface

Pattaphol Jirasessakul*, Zachary Waller*, Paul Marquis*, Connor Scully-Allison*,
Vinh Le*, Scotty Strachan*, Frederick C. Harris, Jr.*, and Sergiu M. Dascalu*
University of Nevada, Reno, Nevada, 89557, USA

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 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 multiple iterations of 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. We evaluate the success of our implementation by comparing metrics of use, determining that both iterations of our software were much faster and easier to use than CHORDS built-in configuration interfaces.

Key Words: 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 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 provide a visualization platform tailored for environmental research to make real-time data available to the research community in standard

formats [1].

However, CHORDS by itself comes with some limitations that makes the use of a middleware vital for operation in larger projects, such as the ones hosted at the NRDC. CHORDS provide an HTTP API that allows for the population of real-time data entry, but the API is only partially exposed and often forces configuration and setup 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.

In this paper we describe the iterative software engineering process we undertook to develop and refine this middleware between the NRDC and CHORDS. Our first iteration attempted to approach the problem of missing APIs by using web scraping and automation tools to emulate the manual setup of new CHORDS instances and visualizations. This approach allowed us to automate tedious metadata inputs by drawing necessary metadata data from NRDC databases and inserting it into form fields. Our second iteration improved on this by further mitigating the need for users to select desired data streams from the NRDC, and instead provides for an *en-masse* download and configuration. Preliminary results of this software indicate that this middleware improves on speed and usability compared to the CHORDS built-in configuration UI.

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, Section 8 details validation metrics and, finally Section 9 shows conclusions and future work.

2 Background & Related Work

This project was made in coordination with the NRDC and EarthCube, both of which fall under the umbrella of the Cyberinfrastructure research as defined by 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.

* Department of Computer Science and Engineering. E-mails: (pjirasessakul, cscully-allison)@nevada.unr.edu, (zacharywaller, paul.marquis1, vdacle)@gmail.com, scotty@dayhike.net, (fred.harris, dascalu)@cse.unr.edu.

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 [5].

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 [3].

2.3 CHORDS & Other Visualization Options

Currently, there are a handful of 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) [8]. VFire is an immersive visualization application that uses remote sensing data in conjunction with a simulation model to predict the behavior of wildfires [4]. 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 [13]. A workflow dedicated to visualizing big data on web

applications was created as an alternative to expensive third-party software [14]. 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 [12].

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 software. Libraries such as D3.js and Plotly.js, are well known libraries 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 an 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 [2, 7]. Unfortunately, both of these and other libraries suffer 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 teams 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 software. Examples of these software 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 [10, 11]. However, as compensation for having many features, they often come with a hefty price tag 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 multiple stakeholders using a formal interview process. The answers we received from these interviews went on to inform the project requirements. These requirements were split between functional requirements, which describe the overall technical functionality of the system, and nonfunctional requirements, which outline constraints on the system.

3.1 Elicitation Interviews

In order to best ensure that the functional requirements composed by developers met the expectations of project stakeholders, several interviews were undertaken to understand their needs and desires *vis a vis* this software. The interviewees were selected for their knowledge of the CHORDS software platform and for their technical understanding of the NRDC. Specifically, we interviewed Scotty Strachan (Environmental Scientist/ UNR Director of Cyber-infrastructure), Vinh Le (Software Developer for the

NRDC), and Zachary Waller (Developer for this Project).

From questions asked of Zachary Waller, the general theme of his needs and requests were oriented around technological specifications and limitations on what software should or should not be used. By contrast, Vinh Le's answers to questions were focused much more strongly about architectural suggestions. His input was crucial in understanding how data should be retrieved from the NRDC and manipulated by the program. Finally, Scotty Strachan provided the perspective of a highly technologically literate user. Additionally, he was the only stakeholder already familiar with the CHORDS platform. From him, we gleaned significant detail about the use cases of the software we were building in addition to understandings of how it should interact with CHORDS.

3.2 Functional Requirements

From those interviews, detailed in Section 3.1, we elicited seven base level functional requirements that define the operations of our solution. The first and second requirements are to create an interface that will not only be able to successfully communicate with a running instance of CHORDS and manage the data inside, but also, to talk 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 requirement is streamlining the user experience by creating a web client to simplify the originally tedious visualization process for users. The NRDC sensor networks currently exist 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 are 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 user's 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 a marker on the map. Additionally, when the user clicks on a marker, the latest photo streamed from that site will be displayed along with specific information about that site, such as name, latitude, longitude, and current measurements.

3.3 Non-Functional Requirements

This project operates under four 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 consists 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 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 component is loosely coupled. A high-level design of the project can be seen below in Figure 1.

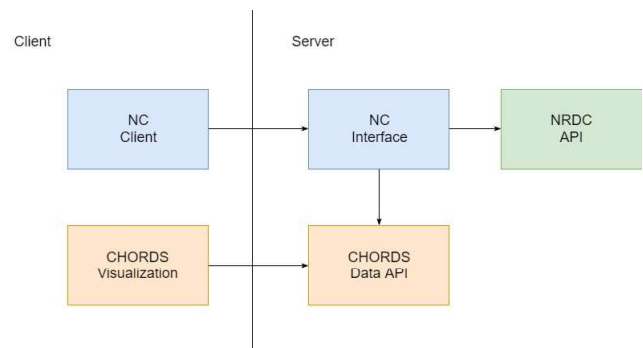


Figure 1: A diagram showing the high-level design of the project

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 each web page on the web page. It also implements 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 interface users that interact with when using the *Generalized Software Interface for CHORDS*: The interface web client and the CHORDSs interface. 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 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 measurements reported and include above Figure 3 the dates of those measurements. Additionally, a list of all visualized sessions 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.

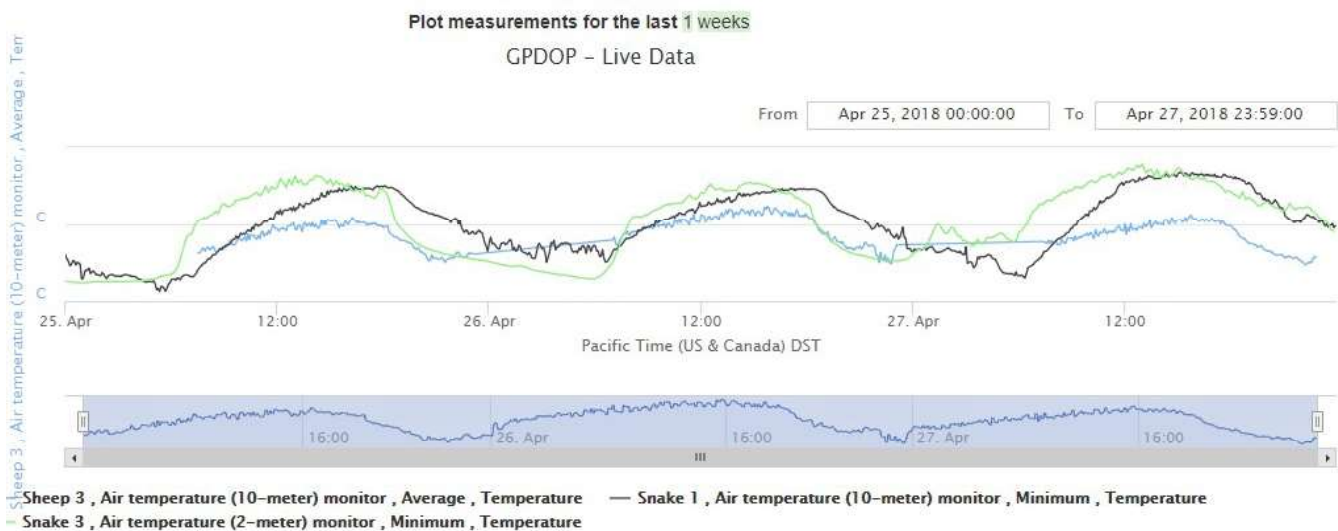


Figure 2: An example of a visualized session of 3 different data streams on a CHORDS instance

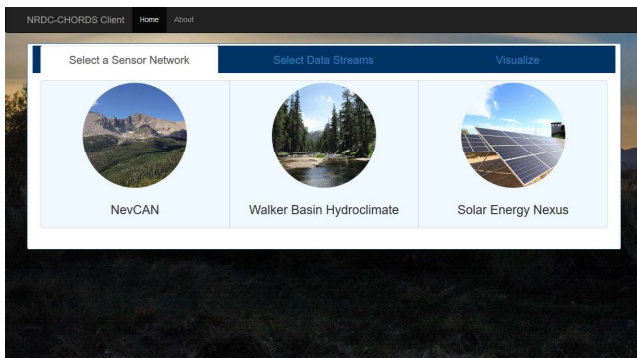


Figure 3: The main page for the NRDC-CHORDS interface web client. A user can begin finding a datastream to visualize by clicking on one of the three available site networks associated with the NRDC.

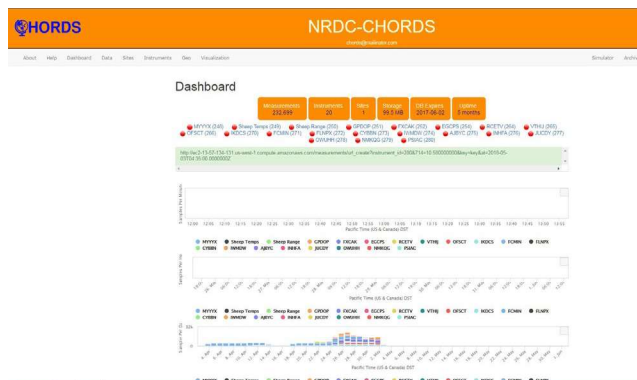


Figure 4: A picture of a research team’s CHORDS’s portal. Currently displayed is a list of the team’s currently visualized session

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 gathered 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 four 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 selects their stream(s), that information is sent to “Session Manager” 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.

To explain our workaround in detail, we utilized the web automation software packages of Selenium and PhantomJS to simulate user interaction with the CHORDS UI [6, 9]. More specifically, after a user successfully selected a data stream, they wanted to visualize from the NRDC-CHORDS interface, our ChordsBot would parse and interact with the various CHORDS web pages required to set up a data visualization on the fly with all necessary metadata about the location, datatypes, variables, etc. required.

The use of this automation software helped us surpass what seemed like an insurmountable requirement given the lack of API support on the CHORDS side. However, it also changed our initial designs somewhat substantially by turning a generic interface into a UI driven web crawler. Additionally, this approach introduced some unique problems which limited the casual use of this software. Most notably, when a data stream was selected from our interface it spawned an entirely new CHORDS session each time. This, in itself, was not an insurmountable problem as we added a module for saving and returning to old sessions, however it seemed to go further against the original design intention of this interface as our own UI was becoming increasingly complicated to act merely as a wrapper for the CHORDS UI. Accordingly, we decided to reexamine our design and make a substantial iteration on our existing work.

7 Prototype Iteration

In the spirit of upholding agile design principles and practices, the initial prototype, detailed in Section 6 was thoroughly examined to evaluate how well the software met the needs and goals of project stakeholders. Accordingly, as part of our evaluation we consulted with multiple project stakeholders and asked them how well it fit their vision. This evaluation produced significant meaningful feedback which provided sufficient cause for us to reconfigure the NRDC-

CHORDS interface in many ways to better fit the express needs and desires of stakeholders.

7.1 Stakeholder Feedback

In order to best determine what adjustments should be made for an iteration on the NRDC-CHORDS interface software, we informally interviewed stakeholders and asked them to play with software. As they interacted with the software package we asked them to speak their minds and express what they like or did not like about the first iteration. We performed this exercise with three individuals who are key to NRDC and the development of this project: Connor Scully-Allison (NRDC System Administrator), Vinh Le and Scotty Strachan.

In these informal sessions we noted several similar points of feedback from the above stakeholders. First, though all stakeholders acknowledged the API limitations we were working with, they unanimously indicated that our UI/web crawling-based solution did not fully meet their earlier expectations of what the software would be. More ideally, they wanted a continuously running CHORDS instance, preconfigured with NRDC data streams that passively accepted new data as it was dropped into the NRDC database.

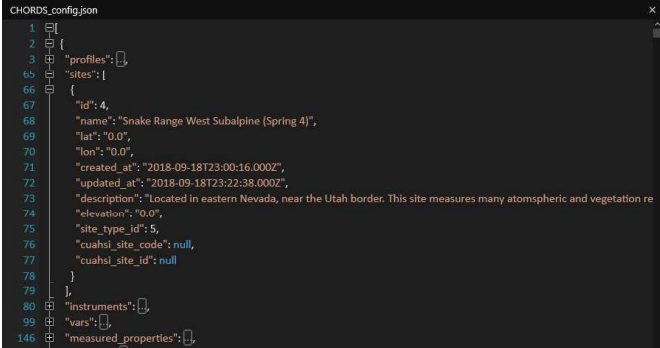
From there, these stakeholders generally indicated that they did not see much of a need for the UI we have built if the CHORDS service could just be preconfigured. Especially since the UI seemed to complicate the adding of many data streams, on different sensors, and at different sites. Although it reduced the tedious typing required of the CHORDS UI, it still seemed to them that there should be a better workaround.

7.2 NRDC-CHORDS Redesign

In exploring options to better meet the expectations of our potential users and stakeholders we redoubled our efforts to find a means to better automate CHORDS configuration. After some searching, we found a previously overlooked function on the CHORDS primary configuration page: a “download configuration” link and an “upload configuration” link. These links enabled us to download a standard-format JSON configuration file to our local machine, modify it and send it back up to the CHORDS instance, whereupon the instance would reload and reflect all the configurations put into the configuration file. To illustrate, an excerpt of this file can be seen in Figure 5.

From this revelation we were able to alter the current architecture and design of the NRDC-CHORDS interface towards a more streamlined and automated package with minimal human interaction required. However, to implement this redesign we needed to first break down the existing code from the prior prototype and evaluate what we could reconfigure for use in this iteration.

We ended up with a short list of re-usable modules after examining our code. We found that the two “managers” were not especially important if we could maintain a static and constant pre-configured instance of instance of CHORDS



```

1  {
2  }
3  "profiles": [],
65 "sites": [
66   {
67     "id": 4,
68     "name": "Snake Range West Subalpine (Spring 4)",
69     "lat": "0.0",
70     "lon": "0.0",
71     "created_at": "2018-09-18T23:00:16.000Z",
72     "updated_at": "2018-09-18T23:22:38.000Z",
73     "description": "Located in eastern Nevada, near the Utah border. This site measures many atmospheric and vegetation re
74     "elevation": "0.0",
75     "site_type_id": 5,
76     "cuahsi_site_code": null,
77     "cuahsi_site_id": null
78   }
79 ],
80 "instruments": [],
99 "vars": [],
146 "measured_properties": []

```

Figure 5: An example of the CHORDS configuration file. The sites, instruments, vars and other fields in this JSON file can be manually edited (or edited by a program) and re-uploaded to CHORDS, just so long as they contain the same data fields which CHORDS expects

running, so we scrapped those. From there we determined that we no longer required the web crawling component of ChordsBot, so we excluded that codebase as well. In the end we ended up re-using large parts of ChordsBot’s backend functionality. Specifically, we adapted those parts that managed the near-real time data upload to CHORDS instance in addition to those that managed the mapping between NRDC’s schema and CHORDS. We were also able to reconfigure part of the DataCenter module which handled communication with the NRDC’s data and metadata APIs.

From this redesign the new high-level architecture shown in Figure 6 was produced. In this figure you will note that the high-level architecture has undergone significant change from that detailed in Figure 1. Now it is no longer broken up as a basic client server pattern but corresponds more roughly to a pipeline that runs on a server chosen to host the “Automated CHORDS Client”. Now this new architecture is not without its own need for manual interaction from users, however in choosing to utilize this configuration file as the main means of configuration, the need for direct user interaction is significantly lessened.

7.3 New NRDC-CHORDS Workflow

The new workflow of this application begins with the user manually downloading the default configuration provided with a new CHORDS instance. This configuration file provides a template informing our configuration service how certain metadata items should be formatted. For example, as seen in Figure 5, a “site” will always have a *name* field, an *id*, a *lat*, a *lon*, etc. The configuration document output by our configuration program has to conform to these fields to be properly accepted by CHORDS.

After our “template” configuration file has been successfully downloaded, and placed in a specified folder, the user can run the configuration program, signified by the three green-colored submodules. The configuration program first

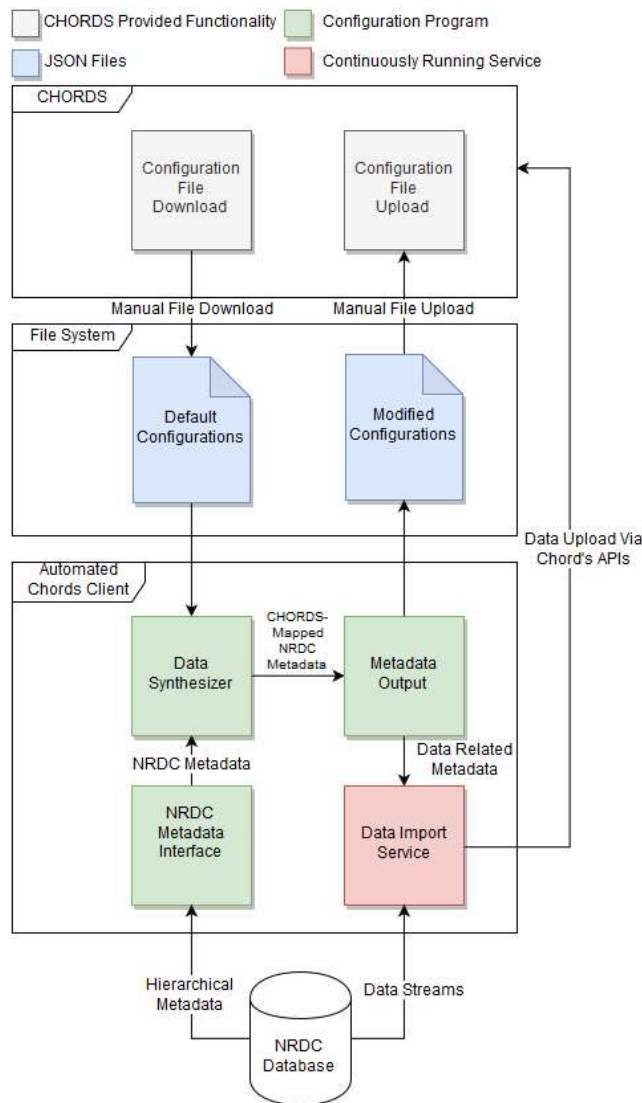


Figure 6: The refined and re-configured architecture of the NRDC-CHORDS interface application. This architecture improves on prior designs by eliminating configuration UIs and utilizing existing configuration files to lessen user interaction in setting up visualization streams.

takes in the JSON configuration file and holds it in memory as a dictionary. From there it calls the NRDC Metadata Interface submodule and retrieves relevant metadata from a particular database in the NRDC. For the purposes of this prototype, the specified database was a database for one project of limited size: NevCAN.

From the database it recovers the full sensor network hierarchy which maps data streams to specific sites and sensors, explained in Section 5.1 of this paper. After recovering the full sensor network hierarchy we use ChordsBot's reconfigured schema mapping functionality to map this hierarchy to the CHORDS expected schema structure. In our case, the NRDC "sites" maps well to

CHORDS's "sites", however our "deployments" are combined with our "components" to map to CHORDS "instruments". Essentially the NRDC has most of the metadata required to set up these sites and instruments, however the data must be extracted from various different tables in the NRDC database and meaningfully mapped to specific fields in the CHORDS configuration file. This mapping is the primary responsibility of the Data Synthesizer module.

For each "site" and "deployment" retrieved from the NRDC database, a new, synthesized "site" and "instrument" object is pushed into their respective arrays. These arrays are then loaded into the CHORDS configuration object we read earlier. This object is then passed along to the metadata output service which is in charge of writing out a properly formatted output file containing the modified configuration information. This file must be manually uploaded back to the CHORDS instance and will cause a full reset of all exiting configuration and data contained in the instance. Accordingly, this configuration and setup should only be done when first starting a new CHORDS instance or when major changes were made to the number of sensors or sites have occurred since the last configuration.

The final module in this new architecture can be seen in the lower left-hand corner colored red. This module is an active service which queries a specific set of data streams, on a timed loop, from the configured data base and pushes them up to the pre-configured CHORDS visualizations as they come in from various sites around Nevada. This service works in exactly the same way as in the prior iteration of our software, using the CHORDS-provided GET calls to pass up data into the CHORDS instance.

8 Validation

The initial iteration of our software was successful in significantly reducing the time and keystrokes required to set up a short-term CHORDS instance. Although not formally evaluated with a user study, we made rough estimations from our own experiences with this software on the time saved to set up a visualization with our UI based interface.

To set up a single data stream, instrument and site from scratch takes on the order of minutes (approx. 2-5 depending on one's familiarity with the necessary metadata fields). With our interface we significantly reduce that setup time by allowing the user to just click through the NRDC hierarchy and select their desired data streams. The same process of setting up a single data stream, instrument and site with the UI could be easily done in under 30 seconds if one is acquainted with the interface. Additionally, no manual typing is required when using this method.

With the second iteration of this software package, we improve on these metrics of usability and speed even further by reducing user interaction to the mere download of a file, the running of a command line program and the upload of an output file. In addition to this simplified workflow from a user's perspective this new architecture allows for establishing a long running, readily accessible CHORDS

instance which has data continuously streaming to it over a long period of time. The prototype developed for the second iteration of the NRDC-CHORDS interface was successfully running for the month of October 2018, ingested over 100,000+ data points, and readily visualized all uploaded data in real time when the site hosting the instance was visited.

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.

9 Conclusion and Future Work

In this paper we detailed the iterative software process undertaken to build and refine a generalizable software interface connecting a data repository like the NRDC to the CHORDS visualization service. We additionally validated our software interface by comparing it against the usability and speed of CHORDS built-in configuration interface.

Although significant future work is planned for this software solution, the most obvious addresses the need for enhanced configurability with the data and metadata sources. For the purposes of this prototype much of the NRDC schema and connection information was hard-coded into our prototype. In order to enhance the generalizability of this software package, a clearly defined configuration file or module should be developed to enable other users to define their own schemas and data sources.

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.

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] M. D. Daniels, S. J. Graves, B. Kerkez, V. Chandrasekar, F. Vernon, C. L. Martin, M. Maskey, K. Keiser, and M. J. Dye, "Connecting Real-Time Data to Algorithms and Databases: Earthcube's Cloudhosted

Real-Time Data Services for the Geosciences (Chords)," *AGU Fall Meeting Abstracts*, 2015.

- [2] D3.js, <https://d3js.org/#/introduction>, [Online; accessed February 22, 2019].
- [3] Earthcube, <https://earthcube.org/info/about/>, [Online; accessed February 22, 2019].
- [4] 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.
- [5] Nevada Research Data Center, www.sensor.nevada.edu, [Online; accessed February 22, 2019].
- [6] PhantomJS, "PhantomJS - Scriptable Headless Browser," *PhantomJS - Scriptable Headless Browser*, [Online], Available: <http://phantomjs.org/>, [Online; accessed February 26, 2019].
- [7] Plot.ly, <https://plot.ly/javascript/>, [Online; accessed February 22, 2019].
- [8] Likhitha Ravi, S. Dascalu, Frederick C. Harris, John Mejia, and Nouredine Belkhatir, "Visted: A Visualization Toolset for Environmental Data," *Proceedings of the 2015 International Conference on Computers and their Application (CATA2015)*, pp. 335–342, 2015.
- [9] Selenium, <https://www.seleniumhq.org/about/>, [Online; accessed February 26, 2019].
- [10] Tableau, <https://www.tableau.com/>, 2018-1-features, [Online; accessed February 22, 2019].
- [11] Visualr, <https://visualrsoftware.com/features.html>, [Online; accessed February 22, 2019].
- [12] Rui Wu, *Environment for Large Data Processing and Visualization Using MongoDB*, University of Nevada, Reno, 2015.
- [13] 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," *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, IEEE, pp. 4964–4969, 2016.
- [14] 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," *2016 International Conference on Collaboration Technologies and Systems (CTS)*, IEEE, pp. 302-309, 2016.



Pattaphol Jirasessakul graduated from the University of Nevada Reno in 2018. He is experienced with Python, C++, and R with interests in big data, statistics, data mining, and web design. He has one conference publication and is currently working as a Freelance Software Engineer.



Zachary Waller, born and raised in Reno, is experienced with C++ as well as C# and the .NET framework. He graduated from the University of Nevada, Reno in 2018. He has a strong interest in web services and Human-Computer Interaction. He has one conference publication and is

employed as a software developer at Conformance Technologies in Reno, Nevada.



Paul Marquis is a Las Vegas native with experience in C++, python and an interest in big data and data streaming technologies. He graduated the University of Nevada Reno in December of 2018 and currently has one conference publication. He is currently employed at Hamilton

Robotics as a software engineer.



Connor Scully-Allison received his B.A. in Philosophy in 2012 from the University of Nevada, Reno (UNR). Accepted into the master's program at UNR for Computer Science and Engineering in 2015, he is currently working as a research assistant on the Track 1 Nexus Project for the Cyber-Infrastructure lab located in the College of Engineering. His research interests include Human Computer

Interaction, High Performance Computing, and Software Engineering. He has published 7 Conference papers and 3 Journal Papers. As of February 2019, Connor holds a position as a student fellow for the Earth Science Information Partners (ESIP) Organization.



Vinh Le graduated from the University, Reno with a B.S in Computer Science and Engineering in 2015 and a MS in August of 2018. Vinh is a Graduate Research Assistant affiliated with the Cyber Infrastructure Lab at the University of Nevada, Reno. His research interests consist primarily of Software Engineering, Internet Architecture, and Human-Computer Interaction. He is currently employed

with the Reno-based software startup Inlumon.



Scotty Strachan is the Director of Cyberinfrastructure in the Office of Information Technology at the University of Nevada, Reno. Strachan graduated from the University of Nevada, Reno in 2001 with a bachelor's degree in geography and minor in economics. After spending some additional years as a geotechnical consultant and project

manager, he returned to the University and completed a M.S. and Ph.D., both in geography, along with a graduate minor in business administration. Strachan's primary research interests lie in mountain ecosystems and observational networks, and he relies heavily on the integration of information technologies with research to accomplish his goals of producing useful, long-term science.



Frederick C. Harris Jr. received his BS and MS degrees in Mathematics and Educational Administration from Bob Jones University, Greenville, SC, USA in 1986 and 1988 respectively. He then went on and received his MS and Ph.D. degrees in Computer Science from Clemson University, Clemson, SC, USA in 1991 and 1994

respectively.

He is currently a Professor in the Department of Computer Science and Engineering and the Director of the High-Performance Computation and Visualization Lab at the University of Nevada, Reno, USA. He has published more than 200 peer-reviewed journal and conference papers along with several book chapters. His research interests are in parallel computation, computational neuroscience, computer graphics, and virtual reality.

He is also a Senior Member of the ACM, and a Senior Member of the International Society for Computers and their Applications (ISCA).



Sergiu Dascalu is a Professor in the Department of Computer Science and Engineering at the University of Nevada, Reno, USA, which he joined in 2002. In 1982 he received a Master's degree in Automatic Control and Computers from the Polytechnic University of Bucharest, Romania and in 2001 a Ph.D. in Computer Science

from Dalhousie University, Halifax, NS, Canada. His main research interests are in the areas of software engineering and human-computer interaction. He has published over 180 peer-reviewed papers and has been involved in numerous projects funded by industrial companies as well as federal agencies such as NSF, NASA, and ONR.