DESIGN AND ARCHITECTURE OF VISUALIZATION SERVICES FOR THE CYBERINFRASTRUCTURE WEB APPLICATION FRAMEWORK

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Smita Digambar More
Spring 2013
The Undersigned Faculty Committee Approves the
Thesis of Smita Digambar More:

Design and Architecture of Visualization Services for the Cyberinfrastructure
Web Application Framework

Mary Thomas, Chair
Department of Computer Science

Roger Whitney
Department of Computer Science

Jose Castillo
Department of Mathematics and Statistics

December 19, 2012
Approval Date
DEDICATION

I want to thank my family for supporting me throughout my life.
ABSTRACT OF THE THESIS

Design and Architecture of Visualization Services for the Cyberinfrastructure Web Application Framework
by
Smita Digambar More
Master of Science in Computer Science
San Diego State University, 2013

Scientific visualization is used to graphically represent scientific data in order to provide a better understanding of crucial scientific results. It is widely used in different disciplines, including engineering, medicine, chemistry, education, economics, geographical systems. Visualization is often employed in scientific portals, which are used to analyze and monitor data produced by jobs run through the portal. This visualization can be used for the real-time dynamic monitoring of computations in progress, and can also provide better insight into individual scientific jobs or post processing procedures.

In this thesis, we present our approach to visualization in scientific portals built using the Cyberinfrastructure Web Application Framework (CyberWeb) and its services. The objective of this work is to develop dynamic visualization functionality in the CyberWeb which will allow clients to monitor jobs both during the job and after its completion. The Cyberweb Visualization service (CWViz) enables users to analyze intermittent results of jobs running on various remote hosts. CWViz provides a clear understanding of each job’s progress for the user by creating real-time and post-processing visualization plots. CWViz includes features such as the ability to create a variety of plot types, like 2D linear, gradient, and 3D surface or mesh plots; animation and movies; a dynamic web-based portal client; and a command line interface. The system uses the gnuplot library and gnuplot.py package to create the various plots. It also enables users to customize these plots by changing the different plot parameters via a web interface or client API. These plots are created dynamically, and all requests sent by the user are handled in run-time, with proper error handling. This visualization service tool in CyberWeb gives the user a very clear cut understanding of how jobs run, as well as their progression; thereby helping to control them more efficiently.
# TABLE OF CONTENTS

| PAGE |
|-------------------------|------------------|
| ABSTRACT | ix |
| LIST OF FIGURES | xi |
| ACKNOWLEDGEMENTS | 1 |
| CHAPTER |
| 1 INTRODUCTION | 1 |
| 1.1 Overview | 1 |
| 1.2 Motivation | 2 |
| 1.2.1 Need for Visualization and Interaction with Scientific Data | 2 |
| 1.2.2 The Cyberinfrastructure Web Application Toolkit (CyberWeb) | 3 |
| 1.2.3 Motivating Application: The General Curvilinear Ocean Model (GCOM) | 3 |
| 1.3 Thesis Project Goals | 4 |
| 2 RESEARCH AND BACKGROUND | 5 |
| 2.1 Science Portals and Gateways | 5 |
| 2.2 Visualization Services for Science Portals | 6 |
| 2.2.1 NERSC Gateways | 6 |
| 2.2.2 XSEDE Gateways | 7 |
| 2.3 Scientific Graphics Libraries | 8 |
| 2.3.1 Matplotlib | 8 |
| 2.3.2 Enthought | 9 |
| 2.3.3 Pychart | 11 |
| 2.3.4 NCAR Graphics | 11 |
| 2.3.5 Gnuplot | 12 |
| 3 CYBERWEB PROJECT OVERVIEW | 13 |
| 3.1 Overview | 13 |
| 3.2 Technologies | 14 |
| 3.2.1 Python | 14 |
3.2.2 Pylons .................................................................................................................. 15
  3.2.2.1 Model-View-Controller (MVC) Structure .................................................... 16
  3.2.2.2 Helpers ............................................................................................................. 17
  3.2.2.3 Unit Testing and Troubleshooting ............................................................... 17
  3.2.3 Python Paste ....................................................................................................... 18
  3.2.4 Secure Shell (SSH) and Secure Copy (SCP) .................................................... 18
3.3 Cyberweb Database Web Admin ............................................................................. 19
3.4 JODIS – Job Distribution Service ........................................................................ 19

4 DESIGN AND ARCHITECTURE OF THE CYBERWEB VISUALIZATION SERVICE (CWVIZ) .................................................................................................... 22
  4.1 Design Requirements and Key Features .............................................................. 22
  4.2 Technologies Utilized .......................................................................................... 23
    4.2.1 JavaScript and JSON ....................................................................................... 23
    4.2.2 AJAX (Asynchronous JavaScript and XML) ................................................. 24
    4.2.3 Mako Templates .............................................................................................. 25
    4.2.4 Gnuplot and Gnuplot.py Package ................................................................. 25
    4.2.5 NumPy ............................................................................................................. 26
    4.2.6 FFMPEG .......................................................................................................... 27
  4.3 CWViz Design ...................................................................................................... 27
    4.3.1 Plot Creation Workflow ................................................................................. 28
    4.3.2 Plot Selection Workflow ............................................................................... 28
  4.4 CWViz GUI .......................................................................................................... 29
  4.5 CWViz Features ................................................................................................... 31
    4.5.1 Web Page Design ............................................................................................ 31
    4.5.2 Customizing Parameters ............................................................................... 31
    4.5.3 Zooming .......................................................................................................... 32
    4.5.4 Default Plots ................................................................................................... 33
  4.6 Test Cases and Validation .................................................................................... 33
    4.6.1 Job Selection Validation ............................................................................... 34
    4.6.2 Checking Previous Plots ............................................................................... 35
    4.6.3 Checking Input File Format .......................................................................... 35

5 CWVIZ RESULTS: THE GCEM PORTAL VISUALIZATION TOOL ......................... 36
5.1 Browsing Job Files and Data .................................................................36
5.2 Generating 2D Linear Plots .................................................................39
5.3 Generating 2D Contour Maps ..............................................................41
5.4 Generating 3D Plots ...........................................................................41
5.5 Generating Animations .....................................................................45

6 CONCLUSIONS AND FUTURE WORK ...................................................49

REFERENCES ..............................................................................................50

APPENDIX

CYBERWEB DEPENDENCIES AND INSTALLATION ...............................52
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>GCOM-CE architecture</td>
</tr>
<tr>
<td>2.1</td>
<td>Matplotlib sample images showing different kinds of plots</td>
</tr>
<tr>
<td>2.2</td>
<td>Enthought product – Mayavi examples showing vector plots and a canyon plot</td>
</tr>
<tr>
<td>2.3</td>
<td>PyChart examples displaying a bar graph and pie chart</td>
</tr>
<tr>
<td>2.4</td>
<td>NCAR graphics examples of a vector map and a triangular mesh</td>
</tr>
<tr>
<td>3.1</td>
<td>CyberWeb architecture focusing on client, middleware, and back end cloud resources</td>
</tr>
<tr>
<td>3.2</td>
<td>Model-View-Controller overview</td>
</tr>
<tr>
<td>3.3</td>
<td>Pylons web framework implementation of the MVC architecture</td>
</tr>
<tr>
<td>3.4</td>
<td>CyberWeb database</td>
</tr>
<tr>
<td>3.5</td>
<td>JODIS architecture</td>
</tr>
<tr>
<td>4.1</td>
<td>Basic AJAX workflow</td>
</tr>
<tr>
<td>4.2</td>
<td>Gnuplot examples showing a 3D surface with a contour map and a parameteric graph of a data set including a cumulative mean</td>
</tr>
<tr>
<td>4.3</td>
<td>CWViz workflow</td>
</tr>
<tr>
<td>4.4</td>
<td>Cycle of visualization</td>
</tr>
<tr>
<td>4.5</td>
<td>MVC structure of plot selection workflow</td>
</tr>
<tr>
<td>4.6</td>
<td>Example of a plot display using the CWViz tool</td>
</tr>
<tr>
<td>4.7</td>
<td>General user interface layout of a visualization tool</td>
</tr>
<tr>
<td>4.8</td>
<td>Changing plot parameters</td>
</tr>
<tr>
<td>4.9</td>
<td>Example of a CWViz zooming plot</td>
</tr>
<tr>
<td>5.1</td>
<td>GCOM portal system</td>
</tr>
<tr>
<td>5.2</td>
<td>Job summary of job 140112</td>
</tr>
<tr>
<td>5.3</td>
<td>Elapsed time for job 140112</td>
</tr>
<tr>
<td>5.4</td>
<td>Time per iteration plot of job 140112</td>
</tr>
<tr>
<td>5.5</td>
<td>Run time plot of job 140112</td>
</tr>
<tr>
<td>5.6</td>
<td>Contour plot for job 140112</td>
</tr>
</tbody>
</table>
Figure 5.7. Grid 97x33x33.dat file contents. .................................................................43
Figure 5.8. Grid 97x33x33.prosize.dat file contents. .......................................................43
Figure 5.9. Bathymetry plot of the Monterey Bay grid. .....................................................44
Figure 5.10. Bathymetry plot of the Alarcon Seamount. ....................................................44
Figure 5.11. Bathymetry plot of the Alarcon Seamount, 2002. .........................................45
Figure 5.12. Contour movie plotting workflow. ...............................................................47
Figure 5.13. Sequence diagram on contour plots. ............................................................48
Figure A.1. Doxygen documentation for CyberWeb. .......................................................55
Figure A.2. Hierarchical diagram in Doxygen documentation. .........................................56
Figure A.3. CyberWeb documentation using Sphinx. .......................................................57
ACKNOWLEDGEMENTS

I want to thank my advisor, Professor Mary Thomas, as well as my thesis committee members, Dr. Roger Whitney and Dr. Jose Castillo, for guiding me through this process. I would also like to thank Carny Cheng and Hetang Shah for sharing their knowledge with me. Lastly, I would like to thank my family for supporting me throughout my life.
CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In the last few decades, the Web and the Internet have been widely used for scientific computing, because of their advanced data storage and management capabilities, available computational systems, and their advanced user interface and visualization environments and services. Cyberinfrastructure consists of a combination of hardware, advanced Internet, the Web, networking, and software components, whose goal is to provide both parallel and distributed computation capabilities for experimental research with high-level performance [1]. Currently, cyberinfrastructure is commonly used in technology-heavy disciplines such as engineering, medicine, the sciences, education, fluid mechanics, and geographical systems. Examples of existing cyberinfrastructure programs include the XSEDE project, the DOE’s Open Science Grid, and the nanoHUB and iPlant Collaborative Gateways [2].

The Cyberinfrastructure Web Application Framework (CyberWeb), developed at San Diego State University, provides users with scientific computational environments [3]. CyberWeb services include data and resource management, job execution, and management of resources. A key objective of CyberWeb is to provide users with easy and reliable web-based facilities with which to manage information regarding tasks/jobs, resources, and services. With CyberWeb, users can perform grid-enabled advanced distributed computations such as job creation and execution, dynamically configure machines as resources, manage groups and accounts, as well as oversee results.

The role of visualization technology in such highly complex computing environments is crucial. Visualization can be widely defined as either 2D or 3D graphical representations of any kind of data. The main goal of visualization is to enable the user to have a better understanding of important information, with the assistance of computer graphics such as charts, histograms, topological maps or contour maps, etc. Visualization services are needed in order to analyze results generated by the service. Often, computations and simulations can run for hours, and can create tera-scale or exascale output files. Analysis of these extremely
large data sets and creation of simple informative graphs, plots, and movies can help to better understand and interpret the results.

1.2 **Motivation**

Visualization is a very effective way of representing or viewing data in a graphical format that can be used for any kind of numerical data. In the 21st century, data visualization has become an important part of research, and the development of computation in scientific fields such as engineering, multimedia, medicine, education, science, etc. has become increasingly common. Scientific visualization is a branch of science that concentrates on rendering real-world 3D scientific data as a means of helping scientists understand their experiments. It is widely used for complex, real-time data analysis, and is also involved with the interactive display of analyzed images. It has enabled scientists to understand various theories and concepts, and has helped them to validate these theories within their practical implementation. Visualization can be performed through either a command line interface or advanced GUI programs. Science portals provide highly-advanced, user-friendly interfaces and user-interactive visualization services.

1.2.1 **Need for Visualization and Interaction with Scientific Data**

Scientific visualization is often used for analyzing distributed computational jobs in which large computations create outputs of terabyte or even exabyte file size. Scientific visualization provides a great advantage to the user by helping to simplify the process of analyzing scientific data through the processing of data in graphical form, and providing user interaction with the data.

Visualizing these large files is a daunting task, and often involves significant processing power. Visualizing large amounts of data can create mathematical errors; it is therefore suggested that the monitoring of simulations currently in process can provide information about the progress of a simulation, as well as give an indication of the projected results.

Interaction with the results at different stages of a job’s progress helps in understanding the performance of simulation. This interaction can include changing views or data types, or comparing data values with time evolution.
1.2.2 The Cyberinfrastructure Web Application Toolkit (CyberWeb)

CyberWeb provides key cyberinfrastructure services, including adding and managing resources, and running and monitoring serial and parallel jobs on remote hosts [3, 4]. Major computing efforts are needed to run large scientific applications. Cyberinfrastructure is a term used to collectively characterize the resources, services, and software that are integrated into a common environment and operate at tera, peta, or exascale levels; CyberWeb is used to connect science model users to these resources. A Cyberweb science portal does an excellent job of allowing users to run jobs remotely, as well as to monitor the status of these jobs; however, the CyberWeb framework had no data visualization services, and that is the motivation for this thesis.

1.2.3 Motivating Application: The General Curvilinear Ocean Model (GCOM)

Dr. Jose Castillo et al. are developing the General Curvilinear Ocean Model, which is an ocean model that uses a 3-dimensional curvilinear coordinate system to detect changes in different thermodynamic parameters of coastal ocean regions [5]. The Unified Curvilinear Ocean Model (UCOAM) is a full three-dimensional curvilinear model, which has been shown to have greater accuracy than similar models and to achieve results more efficiently. The UCOAM model has been proven to produce improved resolutions and smaller errors over other approaches. UCOAM will have the capability to simulate and predict ocean processes on a range of spatial and temporal scales, including local coastal processes, of less than 1 km. It has been validated against several types of water bodies, different coastline and bottom shapes. These include the Alarcon Seamount, Southern California Coastal Region, the Valencia Lake in Venezuela, and the Monterey Bay in California. CyberWeb is being used to develop a computational environment for the UCOAM project with the following features: community access portal for expert and non-expert users (see running and managing jobs, and interacting with long running jobs; managing input and output files; quick visualization of results; publishing of computational Web services to be used by other systems such as larger climate models. Figure 1.1 [3] shows the architecture of the GCOM – computational environment. It consists of components like middleware services and cyber-infrastructure, and layers providing distributed services. More details can be found in [6, 7].
1.3 THESIS PROJECT GOALS

The core functionality of CyberWeb was previously developed by the CyberWeb team, including Professor Mary Thomas, Carny Cheng, and Hetang Shah (another Masters in Science student). The goal of this thesis is to develop a user-friendly graphical interface visualization service in CyberWeb, which will enable the user to analyze their GCOM simulation and its output on a remote host. The service will be simple, and will provide the basic ability to analyze simulation data using a few different kinds of plots. The interface will also need to provide the user with an easy means of interacting with the data. The visualization services should be available from the Web browser, and should also be capable of being completed on a real-time basis (e.g., while the job is running), and should create plots on the same server that holds the job data. This functionality will provide a job summary and the creation of plots, which will provide more information about job simulation.
CHAPTER 2

RESEARCH AND BACKGROUND

Recall that, in the Internet world, the term “portal” describes a website made up of various web pages and information, and the navigation between them. Every website has its own purpose, and so provides different kinds of textual and graphical information to support that purpose. Portals can be public (the Internet through the World Wide Web), and so open to everyone, or private (Intranet websites), in which only a few users on the same LAN connection can view the information. Portals can also be used to display scientific research information for study and development (this is the role of scientific portals).

2.1 SCIENCE PORTALS AND GATEWAYS

Portals used for scientific high-performance computation can be described as “gateways.” Science portals are frameworks, which consist of back-end components that perform the processing of data; and advanced user interface front-end components, which enable users to interact with it [8]. Science portals provide certain capabilities to groups of users, which allow them to access important data on shared resources, as well as perform different operations with scientific data. Scientists from various technology domains do research, which generally consists of real-world data and the theoretical concepts/theories applied to them. This data can be very complicated and large-scale, and is often maintained on online systems in order to provide access to users at all times. If the data to be processed is very large, it can be simulated simultaneously by distributing equal amounts of data on multiple processors, and then combining the final results into one data set. This distributed processing is helpful in load balancing, and also saves time over the entire simulation. If the input data is excessively large, the distributed simulation can also create a large amount of output files. Many science portals have been developed for high-end scientific manipulation, and the CyberWeb toolkit is designed to build these types of science portals. Science portals are discussed in further detail in the following section.
2.2 Visualization Services for Science Portals

There are many science portals currently in use; each designed for specific research domains, including geography, chemistry, physics, etc. This section highlights two well-known scientific portals – the NERSC Gateways and the XSEDE Gateways, which provide similar types of visualization services for their users.

2.2.1 NERSC Gateways

According to the NERSC website, the National Energy Research Scientific Computing Center is the main scientific computing facility of the Department of Energy’s Office of Science. It is famous for its large computing and storage systems, as well as its highly efficient scientific research environment. The NERSC provides reliable and secure computing systems for different research areas, like physics, chemistry, materials science, analysis of data, and astronomy, etc. [9]. From the NERSC website:

It has various computational systems for every research area; some of them are Hopper Cray XE6, Edison, Carver, Euclid, Dirac GPU Cluster, PDSF and Genepool. Hopper is the fastest NERSC’s computer with peta-flop system, 153,216 computer cores, 212 terabyte memory and 2 petabytes disk memory. Carver is an IBM iDataPlex system providing good CPU performance for mid-range parallel applications. PDSF is a distributed network-computing cluster and facilitates data analysis for physics, nuclear science and astrophysics. The Genepool cluster designed for Joint Genome Institute for their computing research.

NERSC provides HPC computers and storage to users or scientists to perform data computations and interaction with the help of web-based interface. Users can also create their own scientific gateways with the help of NERSC building blocks and modules. The scientific gateways with fully featured backend environment can be developed in either of the programming languages – Java, Python, Ruby or PHP. They provide databases like MySQL, PostgreSQL, etc. [10]

The NERSC provides visualization tools for data analysis and inspection of simulation output at different time intervals and for specific areas of interest without spending a significant amount of time. Some of its visualization applications and libraries include AVS/Express, VisIt, Matlab, IDL (Interactive Data Language), ParaView, Mathematica, Grace, R, NCView, HDFView, movie making with FFmpeg, and ImageMagick [11].

VisIt is a 2D and 3D scientific visualization application that supports both structured and unstructured grids. It offers both a CLI (Command Line Interface) and a GUI (Graphic
User Interface); thereby providing an extremely interactive visualization service. VisIt also enables users to analyze and visualize large data sets in either a distributed or parallel computing environment. It can be divided into 2 components: viewer and engine. Viewer is capable of rendering on a localhost, while engine performs data processing on remote machines as well as data transfers between different heterogeneous machines. It is capable of creating various plots, including contour, mesh, streamline, vector, volume plots, etc. By loading plug-ins dynamically, it also improves the performance and efficiency of an application [12].

### 2.2.2 XSEDE Gateways

XSEDE (Extreme Science and Engineering Discovery Environment) is a single virtual system used by scientists to compute data using shared resources such as supercomputers. It provides different resources, including high-performance computing, high-throughput computing, visualization, storage, and networking. It supports 16 supercomputers for high-level visualization and data analysis on multiple resources. With the help of these supercomputers, it facilitates high-performance and throughput computing systems, extensive data storage, networking and visualization services. It also provides a web portal for users to monitor jobs, report issues, and analyze and visualize data. It is protected from software hacks as a result of its advanced cyber-security [13].

The XSEDE project (formerly the TeraGrid) hosts many scientific gateways, some of which are listed below:

- b. Chemical Informatics and Cyberinfrastructure Collaboratory for biochemistry and molecular structure and function.
- c. CyberGIS Gateway for geography and regional science.
- d. EPSCoR Desktop to TeraGrid Ecosystem for systematic and population biology.

More information about these gateways can be found in [14].

XSEDE provides visualization services for all its users via a variety of software. Below is a list of visualization resources.

- a. Longhorn –
  It is a large visualization cluster designed for data analysis and interactive serial and
parallel visualization. It includes multiple computing cores (2048 cores) and graphic processors with 14.5 TB of memory [15].

b. Nautilus –
Is a shared memory remote visualization system configured with both CPU and GPU computing nodes [16]. It consists of one very large shared memory for access to all processors, and is therefore helpful in large data manipulations.

c. Spur –
Spur is a Sun Visualization Cluster with 128 computing cores, 32 GPUs, and 1 TB of memory [17]. It is a very powerful visualization system and eliminates the migration of files from different file systems on hosts.

Both of these science portals are famous in their domains for their powerful computing and visualization services. CyberWeb is designed to develop gateways and services similar to the projects listed above. CyberWeb currently provides features such as job building, job execution, and data resource and account configuration and management. However, CyberWeb has no visualization services, aside from the ability to view job status and raw data files. The next step for this project would be to add visualization services to those already provided, and offer a web interactive toolkit to interact with the data created by these jobs. This idea is what encouraged us to develop a visualization tool in CyberWeb API, which will help us to achieve this goal.

2.3 Scientific Graphics Libraries

There are many scientific graphics libraries that can be used for visualization purposes. Each library utilizes different techniques and syntax, and has both advantages and disadvantages of use. Some important graphics libraries that are relevant to this thesis are discussed below, and can be used with the python programming language.

2.3.1 Matplotlib

Matplotlib is one of the most popular 2D plotting libraries for use with python. It is an open-source, object-oriented API and supports different types of plots, like histogram, contour maps, pie charts, bar charts, error charts, etc. It also features support for NumPy mathematical extensions, in order to create plots more efficiently. It also enables users to embed these plots into different web applications. Additionally, it includes many graphical user interface toolkits. As its development
origin was inspired by MATLAB, its syntax is very similar to that of MATLAB. And, as it uses NumPy extension, the plots can be created with very minimal code.

Matplotlib consists of 3 basic components:

a. Frontend / Matplotlib API
b. Pylab interface
c. Backend

The following diagram (Figure 2.1 [18]) shows examples of 2D graph and 3D mesh plots created using the matplotlib library.


The matplotlib API consists of groups of classes in which the entire plot creation is executed. It consists of different algorithms and techniques to take user preference inputs, and optimize all of the plots accordingly. The Pylab interface consists of functions that enable users to create plots, which are similar to MATLAB. And finally, the backend includes renderers which create hardcopy image files, and drawing or display devices. It supports different output formats, like scalable vector graphics (svg), ps (post-script), pdf, eps (encapsulated postscript), and png.

With all the advantages of matplotlib, it is unfortunate that it is difficult to port it with different Python web application frameworks.

2.3.2 Enthought

Enthought is not just a plotting library: it is also a collection of different components for Python, including 2D and 3D graphics libraries, mathematical and scientific libraries, application frameworks, and developer tools. These packages include Traits, TraitsUI,
Chaco, Mayavi, BlockCanvas, SciMath, CodeTools, EnvisageCode, EnvisagePlugins, etc. [19]. Traits packages act as the heart of Enthought, having a type of definition that can be used by Python objects. Chaco is a 2D plotting library toolkit which provides different levels of script complexity for use in plotting interactive data visualization. It also provides tools to integrate complex data and create relationships between them.

Figure 2.2 [20] shows examples of both a vector graph and a canyon topology image created using the Mayavi-plotting library in Enthought.

![Figure 2.2. Enthought product – Mayavi examples showing vector plots and a canyon plot. Source: Enthought. Mayavi Example Gallery, 2010. http://docs.enthought.com/mayavi/mayavi/auto/examples.html, accessed Nov. 2012.](image)

Features of Enthought include modular, extensible architecture and configurable rendering options, which can be looped together. Mayavi is a 3D plotting library consisting of two packages – Mayavi provides interactive data visualization, and TVTK (Traits-based Visualization ToolKit) is a wrapper of a toolkit that supports characteristic attributes and NumPy arrays. BlockCanvas also supports plotting, and provides different functions for optimization.

Enthought is easy to install and can be used for high-end projects in which all its features can be used. However, for comparatively small projects, it is difficult to maintain with all of its required dependency libraries.
2.3.3 Pychart

PyChart is another Python plotting library, which supports pdf, png, and svg charts, as well as encapsulated postscript output formats for line plots, bar plots, pie charts, and range-fill plots. It is installed through a terminal and does not provide a GUI toolkit. It can be used for small projects as it is lightweight, and is also syntactically easy to write code with [21]. Figure 2.3 [22] shows an example charts generated using PyChart library.

![PyChart Example Charts](image)


Some of the disadvantages of PyChart are that it does not currently support contour plots. It also does not support a very wide range of plots, and does not contain functions to make plots optimized and attractive.

2.3.4 NCAR Graphics

The NCAR (National Center for Atmospheric Research) graphics library package supports the FORTRAN and C programming languages, and can be used to create different types of plot-like maps, vectors, streamlines, histograms, etc. [23]. It also contains a math library to manipulate 2- and 3-dimensional data. One of the important features of NCAR is its ability to store a maps database. The NCAR command language (NCL) [24] is an interpreted language within NCAR used for scientific visualization. As the programmer needs to write the scripts in the NCL language, it is not very portable with different
programming languages. Also, compared to other plotting libraries, it does not provide an easy means for customizing plots.

Figure 2.4 [25] shows a vector plot and a topology mesh diagram created using the NCAR graphics library.


2.3.5 Gnuplot

Gnuplot is one of the most important command-line graphics libraries used for scientific visualization. The most significant feature of gnuplot is that it is very flexible, and is also easily portable on various operating systems, including Windows, Mac, Linux, OS/2, etc. It is open source and can be coupled with different web application frameworks. It also provides a wide variety of plots, such as histograms, vector plots, contour maps, bar graphs, etc. The syntax of gnuplot is easy, as multiple commands can be combined into one for optimizing scripts. Gnuplot.py is fully object-oriented, and there is an open source package that can be used as a bridge between the Python programming language and gnuplot. Gnuplot supports many output formats, like png, postscript, gif, svg, etc., and also provides the ability to create animated gifs. After comparing gnuplot with the competitive plotting libraries mentioned above, we determined that the CWViz should be implemented using gnuplot. We chose gnuplot due to the fact that it is portable and easy to maintain on heterogeneous systems. Gnuplot is discussed in more detail in Chapter 4.
CHAPTER 3

CYBERWEB PROJECT OVERVIEW

The CyberWeb project is designed to provide a user-friendly web interface for submitting and managing jobs [3]. CyberWeb offers an advanced computational environment for applications such as the General Curvilinear Coastal Ocean Model (GCCOM). CyberWeb is based on Python and the Pylons Web Application Framework, and other emerging web technologies including JavaScript, Sqlite 3 database, XML, etc [7]. CyberWeb portals allow users to monitor the current information within queued or running jobs as well as their status, carry out different demonstration tests, and view data files and transfer them between different resource machines. CyberWeb has a rich interface, and provides tools for the deletion and modification of different database records.

3.1 OVERVIEW

CyberWeb is a web-based application framework, which combines hardware and software for computational purposes such as creating a job, managing a job, accounts, data, and performance analysis. Figure 3.1 [7] describes the architecture of CyberWeb. CyberWeb includes both frontend and back-end structures which work together to provide these computational facilities. The front end (client) consists of a desktop browser or web interface, and the back end consists of back end web services, remote host services, etc. The communication between the front end and back end layers is bridged using a Pylons framework and scripts. CyberWeb uses a MySQL database to store all of the important administration accounts and job data for post-processing purposes.

A key objective of CyberWeb is to provide a quick and easy way to configure and manage portals; i.e., to manage the associated resources and accounts and host different applications on the portal. Please refer to the Appendix for cyberweb dependencies, installation and code documentation.
3.2 TECHNOLOGIES

The main building blocks of CyberWeb are Python, the Pylons web framework, and python Paste. This section will concentrate on these important technologies, with an emphasis on those that have an impact on the CWViz tool.

3.2.1 Python

Python is one of the most popular object-oriented languages, along with Perl, Ruby, and Java. It is widely used as a scripting language, as it has a very clear, elegant syntax and makes it easier for programmers to write the code in a structured way (thereby making long-term code maintenance simpler). It is simple to use; is ideal for prototype development; and supports various tasks, such as string operation with regular expressions, connectivity to web servers, file operations, etc. It also provides an interactive mode to check code modularity, which is particularly helpful in managing large projects and tracking test code. Python is also easy to extend by adding new modules implemented in either the C or C++ language. Due to its portability, Python programs can run on Windows, Mac OS, and different versions of
UNIX. It has a large set of standard libraries, providing multiple new features. Python is also fully dynamic, and provides automatic memory management. It is easily extendable, and can be embedded within different programs written in C or C++.

The latest version of Python is 3.2; its new features are listed below.

- Flexible string representation – The Unicode string datatype supports internal representation, depending on the character with the largest Unicode in the represented string.

- Functionality – The difference between narrow and wide builds does not exist, and is just considered wide build. As a result, some of the built-in functions, like \texttt{len()} (Note: this function is used to determine the length of the string) returns 1 for non-BMP characters.

- Qualified name for classes and functions – Functions and class objects have a new attribute, called \texttt{"__qualname__"}, which represents the path from the model top-view and its accessibility with respect to the scope.

- New and improved modules – The new version of Python has improved advancement in modules like array, bz2, codec’s, crypt, curses, etc.

3.2.2 Pylons

Pylon is an open-source web application framework in Python, based on Model-View-Controller (MVC). MVC describes a design pattern that is used in software engineering. The model contains data that is used by your web application; but does not contain any meta-data. Most of the time, the model contains data manipulation of the database tables. The view handles the front end of the application; e.g., such as the way the data is displayed on web pages to the user. The controller consists of the logic of the application; i.e., it has control of everything that can be done in the application. Whenever a user requests an operation, the controller is responsible for directing that request to its proper destination and displaying the results to the user. The main purpose of having a MVC structure is to make a clear separation between back-end and front-end functionalities, which helps in modifying, debugging, and maintaining large applications easily. Some basic features of Pylons are explained in detail in the following section.
3.2.2.1 Model-View-Controller (MVC) Structure

Model-View-Controller architecture acts as the heart of most web applications and frameworks. The Pylons MVC, shown in Figures 3.2 [26] and 3.3 [4], consists of three components. Figure 3.2 shows an overview of these three components:

- **Model** – Stores and retrieves data from a database.
- **View** – Represents data/content to the user with the help of different front end languages, such as HTML, CSS, and JavaScript.
- **Controller** – Contains the logic needed to manipulate the data and control its flow.


Whenever the user needs to request data or send a message to the server, he must interact with the controller component. The controller analyzes the request and gives the proper directions needed to complete the request. In order to get the information, the controller will interact with the model, which will in turn get information from the database and send it back to the controller. The controller will then send the result to view component, which will display the information in a human readable format. In Pylons, the controller and model components are built using Python classes and the view component can be built using HTML, Mako, or another templates library. Figure 3.3 shows Pylons web framework of MVC architecture.

3.2.2.2 HELPERS

Frontend web technologies like HTML manipulate data and then display it in a human readable format. To assist these templates with data manipulation, Pylons provide helper functions to implement advanced functionalities like hierarchical tags building, and the pagination of large amounts of complex, data [27]. Pylons provide various helper packages so that programmers do not have to create any feature from scratch. To use these helpers, they must first be imported into view components. Programmers can also add customized functions in the built-in helper modules, to be used in different class templates.

3.2.2.3 UNIT TESTING AND TROUBLESHOOTING

One of the most powerful features of Pylons is its unit testing capabilities. Through these testing features, different areas of a web application can be tested. To take advantage of this feature, a test runner/discovery package should first be installed [28]. The programmer must create a separate directory for test cases so that no single package will run all test cases.
The testing of different parts can be configured using test.ini file, and the entire set of configurations will be declared.

To avoid unexpected errors in web application, programmers can create their own error handling code and then debug it. Debugging helps in the building of quality applications, and Pylon provides different means of error handling, including an interactive debugger interface, the display of error messages with details, and e-mailing traceback to owners or administrators. The error-handling configuration can be found in config/middleware.py under Pylons installation [28].

### 3.2.3 Python Paste

Python Paste is an open source project consisting of a set of utilities to help with the implementation of WSGI middleware [29]. WSGI (Web Server Gateway Interface) is an interface which defines between the web application/web frameworks and the web servers [30]. This interface also provides support for the Python language; thus, WSGI enables Python to receive HTTP requests. The paste consists of a WebOb wrapper for the WSGI environment, a deploy system to find and configure WSGI applications, and modules such as paste scripts, WSGIProxy, and WSGIFiter. As CyberWeb is built on Pylons, it can also be started with paste script.

### 3.2.4 Secure Shell (SSH) and Secure Copy (SCP)

Secure Shell (SSH) is a network protocol for secure data transfer between different machines, and can also be used to execute commands on remote machines. It connects multiple machines via a secure channel over an insecure network. There are two versions of SSH – SSH-1 and SSH-2. It authenticates between machines using cryptography keys. Each machine can produce a pair of public and private keys where the public keys are required to be exchanged between machines in order to communicate with each other through SSH. Note: it is important not to exchange private keys. Along with authentication, files can also be securely transferred via SSH using Secure Copy (SCP).

CyberWeb uses SSH and SCP protocols to securely connect between different hosts or resources.
3.3 CyberWeb Database Web Admin

Organizing the data associated with hosting science portals and services can be very helpful for managing simulations, users, and multiple applications. The organization of this data can be done using different types of database, such as SQLite, MySQL, and ORACLE. These systems provide different ways of maintaining data and relating different objects to each other.

CyberWeb maintains its internal database in SQL database form. MySQL is a popular database which stores data in tables, but does not provide mapping between them. SQLAlchemy consists of a relational algebra engine, which helps in mapping between objects in a relational database [31]. This feature provides a great performance while maintaining large-scale data. However, the most powerful feature of SQLAlchemy is its object-relational mapper, where classes can be mapped into the database in different ways. This gives flexibility to programmers and enables them to connect with databases and handle various database operations.

For these reasons, CyberWeb uses SQLAlchemy for its database. There are 19 tables in the CyberWeb database, which are used to maintain information on the job, the users, groups, queuing system, and services, host accounts, etc. Because we are using SQLAlchemy, which is integrated into the Pylons Framework, it is easy to add new tables or modify existing tables. Figure 3.4 shows all of the tables and their attributes in CyberWeb.

3.4 JODIS – Job Distribution Service

CyberWeb provides a service to the user which enables him to submit distributed jobs on remote hosts. Figure 3.5 shows JODIS architecture. To distribute the work evenly, CyberWeb has another service named JODIS (job distribution service), which was developed based on a master-worker design pattern [3]. The master-worker pattern provides simultaneous processing on multiple machines: the master host handles the execution and the work is done on remote. The resources are heterogeneous and include computation, data storage and networking on different operating systems and computational environments. JODIS has a rich API, which provides different features, such as authentication, SSH, job management, job queuing, and file transfer.
Figure 3.4. CyberWeb database.
Figure 3.5. JODIS architecture.
CHAPTER 4

DESIGN AND ARCHITECTURE OF THE CYBERWEB VISUALIZATION SERVICE (CWVIZ)

As described above, data visualization tools facilitate users’ ability to analyze simulation results. CWViz is designed to have a user-friendly interface, and a simple set of graphical tools and features. This chapter will describe the design and architecture of CWViz, as well as the workflow of all the plotting functionalities and the client API. Jobs running through a CyberWeb portal can take days to complete their simulations; to monitor the progress of the simulation, we have developed a web-based visualization feature that allows users to analyze intermittent results. This new capability will allow users to monitor jobs, and delete or stop computations if needed; thereby saving time, and hopefully improving results.

Every job creates different output meta-data and data files after each step interval; this data then acts as the input for the visualization scripts.

4.1 DESIGN REQUIREMENTS AND KEY FEATURES

CyberWeb provides a comprehensive set of services for data and resource management, which are ideal for running and managing serial and parallel jobs on remote hosts. Every job is assigned a job ID via the JODIS interface, which can be used to create information and job files. As these files also contain meta-data relevant to the job, CWViz can also use them. A CyberWeb visualization tool should use CyberWeb API, and can then interact with it. Some key design requirements include the following aspects:

a. Job browsing / listing.
b. The ability to select different plots for the current job.
c. The interface should be user-friendly for ease of navigation and image viewing.
d. Interactive visualization plots or graphs according to their preferences.

A key feature of CyberWeb visualization tool is its dynamic nature. The user should be able to analyze the data dynamically, and should also be able to interact with it.
4.2 TECHNOLOGIES UTILIZED

A key design choice was that CWViz be made to provide simple and dynamic visualization tools that are supported by web browser technologies. Considering the abovementioned design requirements and features for the visualization tool, CWViz was built using several common web technologies, as described below.

4.2.1 JavaScript and JSON

JavaScript (JS) is a client-side scripting language used for building web applications. It is a structured language and dynamically typed, which implies that variables can be assigned values or data without prior initialization. It also supports object-oriented programming, where associated arrays act like objects. As it is client-side based, it also runs on a client’s browser, through which it connects the end user to the server program.

The JavaScript language not only provides different data types, objects, and arrays, it also provides powerful regular expression for text manipulation with no overhead. JavaScript can be used to traverse through an HTML document and perform dynamic creation of HTML elements. This provides good way to interact with the user in run-time. Along with data manipulation, JavaScript can be used to change the style of webpage. Some of the more important features of JavaScript are that it gives more dynamic control to the user, validates online forms, performs simple client-side manipulation, and detects user browser details, such as browser and screen size.

JSON or JavaScript Object Notation is a text-based lightweight notation for data exchange. This notation is user readable and easy for machines to create and manipulate. It is also language independent, but is mostly used in JavaScript, Python, Perl, and Java. JSON objects can have basic types, like the integer, strings, Boolean, array, or even an object. An example of JSON follows:

```json
Json_object = {
    "Name": "Smita",
    "Dept": "Computer Science",
    "Address": {
        "Street": "9825 Kika Court",
        "City": "San Diego",
        "State": "California"
    }
}```
The purpose of designing JSON was to make the process of serializing and un-serializing the data for different JavaScript data transfers easier. As a result, JSON can be used for AJAX applications, and can also be easily parsed using “JSON.parse” functionality.

4.2.2 AJAX (Asynchronous JavaScript and XML)

AJAX is used on the client side to asynchronously request data or perform server-side manipulation. Using AJAX, we can request data directly from the server, or send data or messages to the server without either refreshing or interfering with the web page on the client-side. To transfer data, AJAX can be used with any data type, including integer, string or even XML object. But JSON is most commonly used because of its easy parsing feature. AJAX, like JavaScript, is platform independent and is supported by all browsers. Due to its asynchronous feature, it is used for dynamic display of data. For example, the pagination of thousands of products can be done so that 10 products are displayed at a time, and the next 10 items can be displayed without refreshing page. Figure 4.1 shows sequence diagram of AJAX workflow.

![Figure 4.1. Basic AJAX workflow.](image)

Whenever the user sends a synchronous call to the server, it takes the form of an HTTP request. In AJAX, each time a request needs to be sent, it is sent to the AJAX engine. The AJAX engine then sends the request to the server, without interrupting the web page,
and gets the data back. Once it receives the data, it then displays it with the help of JavaScript.

The CWViz uses AJAX for file browsing capability, and the selection and creation of plots.

**4.2.3 Mako Templates**

A template is software which is designed to process web templates and information in order to produce web documents as output [26]. Mako is a template engine written in Python [27]. It has a simple API and provides non-XML syntax, which gets compiled in Python; this makes it extremely fast and enables maximum performance. Some common features of the Mako engine are listed below [28]:

- Control structures are built using Python.
- Supports dynamic inheritance, where any function can be specified to calculate inheritance on run-time for any request.
- Supports URL encoding, decoding, and caching.
- Supports Python 2.4 or newer version and the Google App Engine.
- The Mako engine is used in CyberWeb framework and CWViz.

**4.2.4 Gnuplot and Gnuplot.py Package**

Gnuplot acts as the heart of the CWViz tool. All plots are created using the gnuplot library. Gnuplot is a command-line, easy-to-use graphing utility, and is supported by many operating systems, including Windows, Mac OS, Linux, and CentOS. It is open source and used widely for creating different plots, such as bar graphs, histogram plots, arithmetic functional plots, impulse plots, and 2D and 3D plots like contour plots. It was originally created to interactively visualize mathematical functions and data. As a result, it has built-in knowledge of mathematical functions (e.g., logarithmic and exponential), and provides a flexible way to customize almost any type of plot. Its powerful architecture empowers us to render 3D plots with more accuracy. It also facilitates the creation of colored graphs or changing the view of 3D plots with its simple single-line commands.
Gnuplot is case-sensitive and supports many types of output, such as png, eps, post terminal, x11 (for Windows systems), LaTeX, aquaterm, etc. It not only includes different plotting diagrams, but also provides plotting for the same plot in different styles. Figure 4.2 shows examples of plots created using Gnuplot library.

![Gnuplot Examples](image)

**Figure 4.2.** Gnuplot examples showing a 3D surface with a contour map and a parameteric graph of a data set including a cumulative mean.

Gnuplot can also be used by web applications, but not directly. The gnuplot.py package forms a bridge between gnuplot commands and Python. It is also open source and allows user to plot data from arrays, data files, or mathematical functions. The advantage of using gnuplot.py is that it enables the user to plot diagrams dynamically on remote host. Commands are sent to gnuplot through the Python pipeline. Linux, Windows, Mac OS, CentOS and other platforms support gnuplot.py. This package has an object-oriented design which allows for code reuse, and is able to run many gnuplot sessions simultaneously. A simple example of a script is as follows:

1. `g = Gnuplot.Gnuplot(debug=0)`
2. `g.splot(Gnuplot.File(filename, with_='lines'))`
3. `g.hardcopy(plot_filename, terminal='png')`

The first line creates an object of gnuplot class with the debug turned off (debug=0). In line 2, the plot command, g.splot, is used to plot data contained in *filename*, using lines. Statement 3 calls g.hardcopy to plot in png format with the name *plot_filename*.

### 4.2.5 NumPy

NumPy is a mathematical package for scientific computing with Python [32]. It consists of a comprehensive set of numerical functions and objects for the high-level
computation of data. It also provides tools for integrating C/C++ and FORTRAN code with Python. Due to its support for efficient data manipulation capabilities and its extensive library of useful mathematical functions, it is widely used for scientific purposes in conjunction with the Python programming language [33]. It is also open source, and is easily extendible.

In CWViz, the plotting scripts use the NumPy module to read and manipulate large 3D matrices from job results. These 3D matrices can be reshaped into 2D matrices for changing views in plots. In addition, NumPy has been used to traverse grid or mesh data sets.

### 4.2.6 FFMPEG

FFmpeg is powerful open-source software for recording, converting, and streaming audio and video. It can convert audio and video files into various formats. The CWViz tool provides the creation of contour and vector plot movies. This feature has been built using FFMPEG to create movies with a series of sequential plots. With its command-line interface, each input file is read by a FFMPEG decoder, and encodes them into a specific type of output format.

### 4.3 CWVIZ DESIGN

CWViz provides a simple user-interface to analyze and visualize jobs. Figure 4.3 shows the general workflow of CWViz.

![Figure 4.3. CWViz workflow.](image)
Almost all of the plots created in CWViz need to select a job (except the bathymetry plot). Once the user selects the job and clicks the “analyze job” button, he is ready to select a supported plot from the list. The workflow of the plot creation is explained below:

a. Using the file browser, the user selects “analyze job.” This will display the meta-data about that particular job. In this process the Pylons controller stores the associated information as session variables for future plotting purposes.

b. The user then selects the plot option from the plot selection area and clicks the submit button.

c. The user request is sent to the Pylons controller. The controller will determine whether a plot has already been created for that job on the remote host. If not, the controller will send a request for the creation of a new plot. If the plot is present, then it will check to see if the plot was created recently (e.g., within the past 7 days). If yes, then the controller will mark that plot as outdated and will send a request to create a new plot and overwrite the previous one.

d. Once the plot is created, the controller gets the plot image bytes from the remote host and sends it to the Mako file to display it to the user.

4.3.1 Plot Creation Workflow

All plotting functionalities have common steps, such as “checking previously created plots” and “returning plot images.” However, each of them has separate plotting functionality, which is contained in the related script files. These script files are configured and copied as needed to the remote host using JODIS SCP API. All scripts are configured to take arguments from the controller in any order. Some arguments are required, whereas some can be passed off as optional parameters. The controller uses these extra arguments for optimizing the plots according to the user’s preferences. The gnuplot library is used on the remote host to create plots. The plot names can be set in the controller. Figure 4.4 explains the generalized tasks for plotting functionality.

4.3.2 Plot Selection Workflow

The user can browse through the plot selection components to see which plot options are available. For the selection of an item in one of the dropdown lists, a related set of sub-plots choices are displayed for the next step selection. These option calls are made from the Mako file to the controller through AJAX calls. The controller gets these options from the helper library file, viz.py, which maintains the list of all categories and sub-categories of plots. This library file stores all data in a Python dictionary object for controller access. This
4.4 CWViz GUI

As mentioned in the motivation section of this thesis, the results visualization tool uses a model for the GCEM project portal. The design and components that were originally designed for this portal have evolved to become the CWViz tool. In this section we describe the general user interface of the CWViz toolkit.

The visualization tool consists of the following 3 main components:

a. File browsing tool –
   This tool displays the directories owned by the user or the project on remote resources. The list of resources is pulled from the CyberWeb database. Once the user selects a host via the drop-down box, the parent directory, as well as the list of files
Figure 4.5. MVC structure of plot selection workflow.

and directories is listed in the bottom frame. Before displaying this list the system checks to see that the user has permission to browse through the files. If not, no data is displayed. The user can browse in the forward or backward direction. Once the user clicks on the parent job directory, the visualization tools are updated for that job. This will trigger execution of a script on the remote host, where the meta-data about the output files from the “.info” file are retrieved and deployed to the user.

b. Plot options –
This component allows the user to dynamically select the type of plot to be generated. It consists of three basic types of plots –

- Performance plot
- Bathymetry plot
- Contour plot
- Velocity vector plot

In the performance plot section, there are 2 types of plots available:

- Cumulative time
- Time per iteration

All of these plot options are displayed dynamically.

c. Job Summary and plot area –
The job summary section displays all the details or meta-data of the job output files. Whenever a user clicks on “analyze job,” it finds these details from that job folder and displays them. The plot area displays the different plots requested by the user from the plot option section.
Figure 4.6 shows an example of job visualization with a job summary and a sample plot.

4.5 CWViz Features

The visualization tool provides features that make plotting more user-interactive.

4.5.1 Web Page Design

The CWViz user interface is designed according to the requirements of the GCOM ocean model. Figure 4.7 shows the basic components of the CWViz web page. These components include:

- File browser
- Analysis type, plot type, and data value selectors
- Jobs summary and image plotting area

4.5.2 Customizing Parameters

When the user creates a plot for the first time, the controller is configured to use a set of default values for the plots. For a web page, these default values get displayed through a
Mako file. Once displayed, the plot parameters can be changed using widget controls in the plotting area. After changing the default values in the input text areas, the user can click the “submit” button, which triggers the creation of a plot with these new values. While rebuilding the plot, the controller does not check for any previously existing plots, and will directly create a new one.

Figure 4.8 shows the web interface for plotting. It displays the default x-min, x-max, y-min, y-max, z-min, and z-max values of the grid. The display ensures that the new values are in the range of the default values. For example, if the default values for x-min = 0 and x-max = 100, the user will need to enter new x-min or x-max values in the 0 to 100 range, and the x-min should be less than the x-max. Similar rules apply to all user-controlled values for any plotting component. Below is an example of a plot utilizing user input for the Monterey Bay mesh.

4.5.3 Zooming

Due to small browser screen limitations, plots can look small and unclear. To overcome this limitation, the zooming feature was built. With it, the user can just click on the plot to zoom into it, which will show a plot image with its original width.
and height dimensions. To zoom out from the image, the user can click once again on the image or can press the ‘‘Esc’’ button on the keyboard. Figure 4.9 shows an example of zooming feature of CWViz.

4.5.4 Default Plots

Using the CWViz widget, the user can customize the plot according to the given choices. If the user is not satisfied with the customized plot, he can request the default plot by clicking “default plot” in the widget.

4.6 Test Cases and Validation

Visualization functionality code is used to check and validate the different parameters of the job at the time of the plot creation. These validations are done in the controller and in the plot scripts. Every plotting script maintains a Python dictionary to store the status of the running script. If any test fails, then the script will add an error flag and corresponding message into the tracking dictionary, which goes back to the controller. Because of this script feedback, the controller can take action
Figure 4.9. Example of a CWViz zooming plot.

accordingly. This process helps improve the user interface, and optimizes plotting functionality. Specific test cases are explained in detail in the following section.

4.6.1 Job Selection Validation

Job selection validation is done to make sure that the user has selected a job with proper format for visualization. The examples and test cases cited here are specifically related to the GCEM portal, but the process can be generalized for any application. If the job format is not correct, the test case will fail and notify the user. Whenever a user wishes to visualize a job, he must first select the primary folder of job in the file browsing area, which should have a naming convention similar to “job.xxxx,” where xxxx is the id of the job generated by the JODIS API. The job should contain specific folders, such as OUTPUT and INFO, as these folders might have output files of the simulation, which will be used as inputs for visualization purposes. If the job does not satisfy these conditions, then the controller will
send a corresponding error message to theMako file. If the test case passes, then it will read the job summary and display it to the user.

4.6.2 Checking Previous Plots

Before creating any plot, the controller first checks to see if that plot was already created with the help of a `check_plot_present.py` script. This script is copied to the remote host through secure copy. The script takes a plot filename as a parameter from the controller.

If the plot already exists on the remote host, then the controller will check the age of the plot (this helps to improve the visualization performance). If the plot is outdated, it is possible that the job data might have changed. In this case, the controller will send a new plot command. For now, it is set to 7 days. If the plot is older than 7 days, the controller will trigger the creation of a new plot. This setting can be changed in `check_plot_present.py`.

4.6.3 Checking Input File Format

Different plotting scripts must parse different job data files. The format of these files should remain constant, in order to ensure that the job data is valid and the script is not reading any garbage values. If the format changes, the parsing will fail.

For example, the writetimes.dat file used for performance plotting needs to be in the following format for each row:

```
-rw-r--r-- 1 mthomas mthomas 2483200 2012-01-07 09:14:30.000000000 -0800
OUTPUT/u0000.dat
```

For bathymetry plot creation, the gridfile should be in the following format:

```
-2.000000,-1.000000,-1.000000
```

This file format is checked in all of the plot scripts. If these test cases fail, the controller will notify the user.
CHAPTER 5

CWVIZ RESULTS: THE GCEM PORTAL VISUALIZATION TOOL

CyberWeb is being used to develop a web portal system for the GCEM project (see Figure 5.1 [7]). The portal includes the following features: a community access portal for expert and non-expert users; the running and managing of jobs, and interaction with long-running jobs; the ability to manage input and output files; quick visualization of results; and the publishing of computational web services, to be used by other systems (such as larger climate models).

In this chapter we present examples of how the CWViz is used to implement key features and capabilities needed by the GCEM portal system. The web pages and plots shown below were developed for the General Curvilinear Environmental Modeling (GCEM) portal, and were built using the CyberWeb toolkit and CWViz. They demonstrate how CWViz processes and manages the data in both real-time and post-job time frames. The examples were chosen to demonstrate how the CWViz can be used, and the software can be reused to generate new types of plots as needed.

5.1 BROWSING JOB FILES AND DATA

Every job simulation creates an output file, which consists of the meta-data related to the remaining output files. When the user clicks the “analyze job” button, these meta-data values are read from the job.ID/INFO/analytics.dat file and displayed for the user. This meta-data plays an important role in the visualization of these jobs. Figure 5.2 is a sample job summary from job 140112.

The following section explains each of these parameters in detail.

a. **jobtype** – Shows whether it is a serial or parallel job.
b. **iMAX, jMax, kMax** – Every job runs simulations on 3-dimensional grid data. These parameters give maximum values of the grid data. iMax is the maximum limit of the i-coordinate, jMax gives the maximum limit of the j-coordinate, and kMax similarly corresponds to the maximum limit of the k-coordinate. In the example above, the job considers a rectangular cube with an iMax equal to 97, a jMax = 33, and a kMax = 33.

c. **MaxFileNo** – The GCEM job creates a set of output files after a few iterations. These file names are numbered sequentially, with the corresponding step interval. For example – output files u0010.dat, v0010.dat, w0010.dat are created at each 10th step interval. MaxFileNo gives the maximum number of values for the output files to write.

d. **wrthz** – This defines the write frequency of the job, or the number of iterations to be computed in-between write times.

e. **JOBID** – This tells the job id. These are created randomly and are unique numbers for the type of job.
f. **procs** – This provides information about the number of processors used to run the job. In our example, which is a serial job, the procs is equal to 1.

g. **gridname** – As stated earlier, every job simulation works on 3-dimensional grid data. This parameter gives that grid a file name.

h. **Write frequency filename** – This is the input file, which should be present on the remote host in the INFO directory of the job. The controller sends the filename with its relative path to the script file.

i. **Plot filename** – This is the name of the plot image that will be created by the script.

j. **Write frequency of job** – This is the wrthz parameter of the job. (Refer to section 4.2, page 22, for more details.)

MaxFileNo and wrthz are used to control when the data is written to the disk. They can also be used to get information about the jobs run time performance. The write frequency file stores the time at which the output files were created.

The file contains entries for each time step. If the `MaxFileNo` (refer to Chapter 4.2) is 100, there will be 100 lines in the file. The `cumulative_time_plot.py` script parses this file and stores the time elapsed between writetimes using the dateutil.relativedelta module in Python dictionary objects. This delta time is stored as the time per iteration. The time step data is contained on each line. The step interval at which the data was written is computed as \((\text{write}_f\text{requency} \times \text{step})\) and stored as the iteration number. These x-axis and y-axis values are then stored in the .dat file on the remote host, and the script creates a plot in png format using gnuplot.py functionality.

### 5.2 Generating 2D Linear Plots

The CWViz is capable of providing various types of 1D linear line plots to analyze job simulation. For the GCEM model, there is a need to analyze the run-time performance of the model. The plots below show results for the total runtime and the time per iteration.

The elapsed time (or the wall-clock time) plot falls under the performance plotting category, and is created using a script `cumulative_time_plot.py` file.

Figure 5.3 shows the time per iteration plot, and is included in the performance category. The scripts use parameters similar to those utilized by the cumulative time plot script (write frequency filename, plot filename, and write frequency of job).
Figure 5.3. Elapsed time for job 140112.

Figure 5.4 shows the elapsed time versus the iteration number for the seamount test case and Figure 5.5 shows the run time plot of job 140112.

Figure 5.4. Time per iteration plot of job 140112.
5.3 GENERATING 2D CONTOUR MAPS

Contour plots provide useful information by transforming 3-dimensional data on the 2-dimensional plane with definite z-axis slices. They are used to study different coastal waters and their changes over time. They are also used to analyze variables such as temperature, pressure, and water velocity, sediment depositions, etc. Simulations run through the GCEM application create different output files with each of these velocity, pressure, and salinity parameters for every step interval.

With the help of the visualization tool, the user can create contour plots. CWViz uses `contour_script.py` script to create contour plots. Figure 5.6 shows contour plot for job 140112.

5.4 GENERATING 3D PLOTS

The CWViz system can easily produce 3D visualizations. An example of this is the bathymetry plot shown in Figures 5.7, 5.8, 5.9, 5.10, and 5.11.

The bathymetry plot requires mesh or grid data with x-, y-, and z-coordinates. The user must select the grid file and the bathymetry plotting option. This will send a request to the controller, which in turn reads the selected grid file with its problem size file to create a
3D bathymetry plot. The problem size file contains maximum values for all the coordinates (i.e., x-max, y-max, z-max).

In the CWViz, the 3D plots are created by a `bathymetry_script.py` file. The required inputs for this script are as follows:

a. **Grid file** – This file consists of 3-dimensional grid data, which is its x-axis, y-axis, and z-axis values. Each row represents one point in the grid, and the values are stored with comma separations. The first column gives the x-axis value, the second gives the y-axis value, and the last column gives the z-axis value. Example data for the grid file is shown below.

The controller sends this grid file name with its absolute path to the plotting script.

b. **Grid problem size** – This file gives the dimensions of the grid. For the grid example above, the problem size is shown below.

It gives maximum values for all 3 axes. First is the maximum for the x-axis, second is the maximum for the y-axis, and third is the maximum for the z-axis. The last 3 are not used for our plotting. In the example given above, X-MAX = 97, Y-MAX = 33, and Z-MAX = 33. The grid file is parsed according to these dimensions.

c. **Plot file name** – This is the new plot image name.
Figure 5.7. Grid 97x33x33.dat file contents.

Figure 5.8. Grid 97x33x33.probsize.dat file contents.
Figure 5.9. Bathymetry plot of the Monterey Bay grid.

Figure 5.10. Bathymetry plot of the Alarcon Seamount.
The extra parameters are x-min, x-max, y-min, y-max, z-min, and z-max. These extra parameters are used for optimizing the plots.

After parsing the grid file and grid problem size file, the grid data is manipulated and stored in the Python dictionaries and the gridname_gnuplot_grid.dat file, where gridname is the actual name of the grid. Again, the reason for storing the grid data in the file is because gnuplot.py functionalities cannot take plot data from Python objects. So, this data is stored in gnuplot-compatible format in a new “.dat” file.

### 5.5 Generating Animations

When the user gives a request for a contour movie, the postjob.py controller analyzes the request and copies the script file on the remote host. The script analyzes the user inputs provided by the controller, reads the grid data file, and stores it into Python dictionary objects. It also calculates the center of each grid cell for maximum accuracy, using the CalCenter function. Now, for every step interval, the script separates the contour plots and saves it on the remote host. As the job can have many steps (100 to 1000), it is not feasible to
create plot images for each interval. Under such conditions, the user can plot the nth frame after each interval. E.g., If the user wants an interval of 5, then the script will create plots for 0th, 5th, 10th, 15th interval, and so on. Each step interval follows the same routine on the corresponding set of input files. The inputs for this plotting feature are the jobs and their respective grid files. The script will read different output files for the job from the OUTPUT/folder, such as u0010.dat, v0010.dat, etc. The naming convention provides information about its step interval, so the u0010.dat informs it that it was created on the 10th step interval.

These routine functions are described, as follows:

1. **ReadField**
   This function reads corresponding steps in the input parameter file. If the user wants to create a contour movie of the change of velocity, the script will read u, v, and w data from the output files of the job. If the current step is 5th, the script file will read u0005.dat, v0005.dat, and w0005.dat files. If it is a pressure contour plot, then the script will read p0005.dat file. Similar steps apply for the salinity and density.

2. **GetFileNum**
   This function returns the exact string of input parameter files for the current step and is used by the ReadField function.

3. **VecToCenter**
   This function will determine the final velocity at the center of each grid cell, as the velocity is a vector measure of the combination of u, v, and w measurements. For pressure, the salinity script does not have to carry out this step, as it is a scalar factor.

4. **CreateContour**
   This function creates respective contour plot from the data manipulated by the abovementioned functions and saves it on the remote host with its step interval naming convention.

The abovementioned steps are repeated for every interval, creating a sequence of diagrams. These sequence diagrams are then combined to create a movie file in the .mov format, using the FFMPEG program. FFMPEG does an excellent job of creating movie files in minimal time, and features increased clarity from the input set of images. The script uses different options of FFMPEG to improve the quality of the movie. Some of these options are listed below:

   - **-qscale**
     Qscale is used to provide constant quality. It can be given a value anywhere between 1 and 31, where 1 is best quality and 31 is lowest quality [34].
- **-r**
  This determines the frame rate of the input file. For better quality, this should be set to 1.
- **-b**
  It determines the bitrate (bits/s), and should be set to a maximum value. The script in the CWViz tool sets the bitrate to 3000k.

Figure 5.12 gives an overview of the script file.

![Contour movie plotting workflow](image)

Figure 5.12. Contour movie plotting workflow.
The CWViz tool also enables the user to view sequence diagrams, which create the movie. Figure 5.13 shows a sample sequence diagram created by the CWViz tool.

Figure 5.13. Sequence diagram on contour plots.
CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Cyberweb Viz is designed to provide an easy GUI interface and simple to use dynamic visualizations of both running jobs or completed job datasets created through Cyberweb system. With the AJAX implementation, the user GUI capabilities include: file navigation and browsing of all files and directories on all of the systems remote hosts; displaying lists of plots dynamically. The visualization services provide different types of visualization plots designed to provide users with insight about the expected results of job. The visualization scripts can create plots through the CWViz GUI or a command-line interface. Different features of CWViz such as customizing parameters and zooming, gives a better understanding on plots.

CWViz proves two user interfaces: a portal GUI and command line scripts. A future improvement would be to host visualization services that can be accessed by remote clients as a Web service, which is easily done via the CyberWeb framework. CWViz displays plots using Mako functionality, but the output of the visualization tool can be displayed in any format, including in XML, CLI, and Google Gadget. In addition, the CWViz tool provides the user with the ability to create performance, contour, and bathymetry plots. Future work would be to expand this set of plots to create vector plots: for example showing direction of velocity in 2D plane. The basic CWViz controller and scripts have been designed flexible to expand to provide additional plotting capabilities. Currently CWViz supports GCEM model application jobs. In future the MVC model of CWViz can be modified to support different application models.
REFERENCES


APPENDIX

CYBERWEB DEPENDENCIES AND INSTALLATION
A.1 CYBERWEB DEPENDENCIES AND INSTALLATION

Being a scientific web application portal, CyberWeb needs initial configuration and installation on the server to provide different services. This section describes these dependencies.

Dependencies of CyberWeb are as follows:

- Python and Pylons framework which acts as the backbone of the application.
- FormAlchemy, SQLAlchemy, SQLite for database support on an Apache Web Server.
- Paster for the initial application set up.
- NumPy for different numerical manipulations.
- Authkit, pexpect for setting up authorization keys between hosts.
- All of these dependencies are recommended to be installed prior to installing CyberWeb.

Below are the steps to install CyberWeb on any server:

1. Download the CyberWeb development code from the svn server to the local host.
2. As all the initial database settings are stored in initial_data.json, remove any previously existing development.db files.
3. Now create a fresh copy of the database using the development.ini using the command below
   
   `paster setup-app development.ini`

4. Configure the host name in the development.ini file using the command
   
   `paster make-config cyberweb config.ini`

5. Now, set up the application:
   
   `paster setup-app config.ini`

Once CyberWeb is installed, the admin can fire up the server using either of the following commands:

`paster serve development.ini –reload setup.py`

or

`./startup.sh port_number`

The latter commands can only be used if the admin has pseudo permissions, and a

`port_number` should be configured in the development.ini file.
A.2 CYBERWEB TECHNICAL CODE DOCUMENTATION

Code documentation plays an important role in software engineering, and especially in high-end applications. It explains the relationship between the different parts of the programs, thereby giving a clear view of object-oriented structure and algorithms. It also makes the development process easy to maintain when more than one person is working on the same project. This documentation can be created using different types of software, which reads/parses raw code locally and creates a human readable format.

There are many types of paid and open-source documentation software available, which can be downloaded locally and also run locally without any dependencies. Every software type supports a different variety of programming languages. As CyberWeb was developed in the Python programming language, we found the following software types particularly helpful: Doxygen, doc-O-matic, Sphinx, phpDocumentor, Epydoc, etc. PhpDocumentor is mainly created for PHP whereas Doxygen and Sphinx are very popular for use with Python. The following section describes these two types of software in detail.

A.2.1 Doxygen

Doxygen supports a wide range of programming languages, including C, C++, Python, Java, Objective-C, PHP, C#, Fortran, etc. Most documenter software requires proper commenting in the source code to create documentation. Doxygen has features with which undocumented source files can also be parsed to create a quick guide to understand the relationship between object-oriented classes. This can be very helpful in large source distributions. Figure A.1 shows CyberWeb documentation created by Doxygen. Doxygen also creates references between class to super class or inherited and overridden members automatically. It also provides search functionality, where different classes and members can be searched using ranks. It features different ways to create hierarchy, collaboration, and dependency diagrams. It has its own in-built system and also uses the dol tool from the Graphviz toolkit.
Doxygen can create documentation in different formats, including styled HTML pages, Postscript, PDF, MS Word, and Unix pages. It is a lightweight software and can be installed easily on different operating systems. One of the many advantages of using Doxygen is that it can create visual relationship between classes, their inheritance, class and object variables and functions, etc. Figure A.2 shows an example of the same kind of relationship for a visualization controller.
A.2.2 Sphinx

Sphinx was basically developed for creating documentation for Python, but can also be used for C and C++ projects. It creates documentation in different formats, such as HTML and LaTeX for PDF printing purposes, as well as plain text. It consists of extensions to test Python code blocks, use docstrings for documentation, etc. Like Doxygen, Sphinx cannot create structural class hierarchy relationship diagrams, and does not provide a graphic user interface. Sphinx can be installed through a command prompt/terminal. Once installed, the user can set the preferences of the documentation and give a source directory of the project.

Figure A.3 shows a CyberWeb documentation created by Sphinx.
Figure A.3. CyberWeb documentation using Sphinx.