Master Project Report

Teakwood: An Web Framework for Handling Many-task Computing

Submitted in partial fulfillment of
the requirements for the degree of

Master in System Science
in
Louisiana State University and Agricultural and Mechanical College
The School of Electrical Engineering and Computer Science
Computer Science and Engineering Division

by

Rui Guo

Under the guidance of
Jian Zhang

Fall Semester 2014

Abstract

Using Linux commands to handle computing jobs can be a hurdle to those scientific researchers who don't have HPC related background. Teakwood provides a solution and beyond. Teakwood is a framework that migrates all the terminal typing work to a web console GUI, and provides user a total control of their jobs, data, computing resources and so on simply by clicking functional buttons. Teakwood is also an open platform that enables user to work cooperatively. Through Teakwood, user can share their models, results, and computing resources within their group and have discussions in Teakwood forum. Teakwood is powered by Django framework.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Four years ago, I know nothing about Linux. I then got a job to deal with HPC(High Performance Computing) systems. Suddenly I jumped into an Linux environment. I found that all those GUIs(Graphic User Interfaces), which I used to when using Windows OS, are gone, and I have to use Linux commands to manipulate those machines and manage my stuff, which makes me uncomfortable and cumbersome. With time goes by, I eventually can work with Linux comfortably. However, I still have the tendency to use GUI when proving me the option.
I am not the only one that had such experience. In my working building, a lot of scientists and researchers are suffering this pain. Not only this, since they usually have to work on cross-platforms, the OS conflict add them many wasteful jobs. For example, scientist A want to share a computing result to researcher B, as B need it to do the visualization. A uses Linux and B uses Windows. What should they do? Normally without tool's help, then need to type command, command, command... copy, paste, and then command, command ...
Why not create a GUI platform to reduce those repeated typing work and allow them to work cooperatively in one place? With all those desires, Here comes the Teakwood framework.

1.2 Teakwood

Teakwood is a GUI framework that allows user to summit HPC jobs from Teakwood web console, and to have a full control of their job status. Teakwood also provides a file management system for user to organize their project

data easily. What's more, Teakwood enables user to work cooperatively by allowing them to share result, models and computing resources within their group.
Teakwood migrates all the terminal typing work to Teakwood GUI, and wrap them with interateive web pages. This enables user to submit HPC jobs just by simply clicking functional buttons. No more terminal commands.
As Teakwood is a web framework, user can access from any where, any type of machine, as long as the machine has a browser and the Internet. no more hassle by the OS conflict.
What's more, Teakwood file server hosts all users' computing data in one place, which greatly facilitated the collaboration among users.

1.3 Features

Functionally, Teakwood have the following features:

• Perfect documentation
Teakwood homepage provides diverse software documentations including installation guide, user manual, developer manual, video tutorial, etc. user can grasp Teakwood soon.

• Neat web portal
A neat LSU style web console makes your job submission simple and easy. Drag, push and click, that's it. Let's say goodbye to terminal typing.

• Job monitor system
Job monitor system provides five labels: "uploading", "queued", "running", "finish", and "Data Ready" for user to read the job status. Job monitor system also periodically pulls the job running messages from computing server and displays them on working console, user may know more details.

• Project management system
All user's project data is well organized and web kept in a file server. user can compare, share, and download them as needed.

• Powerful admin
The powerful admin system is provided by Django itself. With tiny system configuration, user can activate their admin system and have a top-down control of their account.

1.4 System requirements

Teakwood is a Django powered web framework which integrated a lot of third party packages and external tools; Some of the packages or tools require extra libs and development packages to work, so before running Teakwood, we need to resolve all the dependencies as well as finish setting up all the packages and tools.

• Project Dependent libs
As mentioned above, install all dependent libs are the first thing. Please refer installation guide in homepage for more information.

• MySQL Database
Teakwood uses MySQL database, so make sure mysql-server and mysql-devel is installed.If you want a GUI control of your database, I recommend phpMyAdmin.

• Version Control
Teakwood uses git as version control. Git is a very popular version control software. Teakwood source code is hosted by Github.

• Virtual Environment
It is highly recommended that we set up an independent working environment for Teakwood development. "Virtualenv" is a reliable solution.

• Latex and Sphinx
Teakwood uses sphinx package to generate diverse documentations such as html, latex, pdf, so these tools need to be installed.

• Celery and RabbitMQ
This is a solution for resolving synchronous process.

• Django 1.4.x
No need to explain.

• Third Party Packages
Teakwood uses a lot of "wheels" to construct the Teakwood structure, so those wheels need to be installed in the right path before Teakwood starts runing. Refer "requirements.txt" for more information.

Chapter 2

Teakwood System

2.1 Overview

Structurally, like most websites, Teakwood system has a three layers layout: frontend, backend, and database. Becuase Teakwood uses computing servers to run jobs, so the eco Teakwood system actually has four layers; the up mentioned three plus a computing layer. the below figure shows how is the four-layer structure looks like:

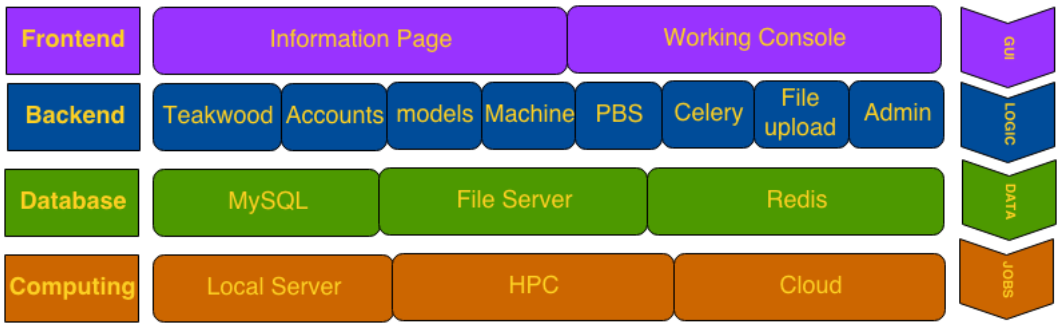

Figure 2.1: Teakwood System Overview

On the above figure, we can see the four straight layers lays bottom-up. Each layer from left to right: the left part is the layer name; the middle part is the layer content; and the right part is the layer reality. Let's go through these layers one by one.

2.2 Frontend

The frontend is a visible GUI that the user interact with. Basically all what we can see in the Teakwood website can be called the "frontend". For neat purpose, Teakwood separated the frontend into two parts: the **information pages** and the **working console**. See below:

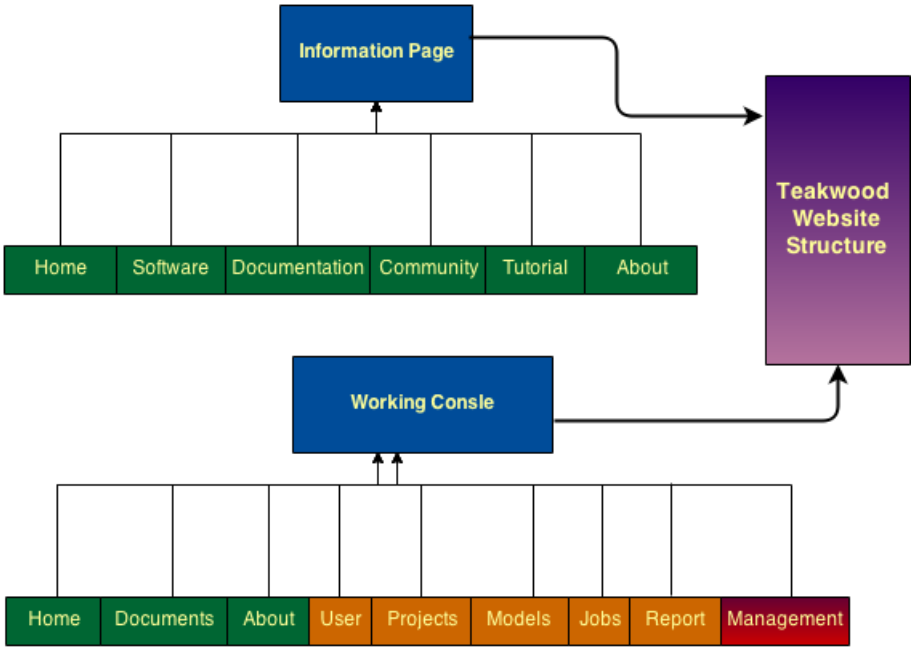

Figure 2.2: Website Strucutre

The **information pages** introduces you all general things about Teakwood. For examples:

- What is Teakwood?
- What can Teakwood do?
- How to install Teakwood?
- How to use Teakwood?
- The user manual.
- Video tutorial.
- Teakwood forum.

The **working console** is the working place you actually tango with your jobs and data. The functional buttons will guide you to different functional web pages for different purpose.

Let me explain these functional buttons one by one.

- **Home**: Directs you to the homepage.
- **Documents**: Directs you to the user manual.
- **About**: Teakood self introduction.
- **User**: Displays user information.
- **Projects**: Creates projects and provies project overview.
- **Models**: Create models and provides models overview.
- **Jobs**:Creates jobs, overview jobs and job monitoring.
- **Report**:Download job output.
- **Management**:Access to the admin system.

Note the color differences in the bottom layer.

- The **Green**: all visitors can see and manipulate.
- The **orange**: only logging user can see and manipulate.
- The **red**: only superuser can see and manipulate.

2.3 Backend

The backend contains all the logical design of data manipulating and HTML producing. In detail, it includes verifying user's request, pulling requested data, generating HTML web page and displaying the web page. Teakwood follows the MVC(Model-View-Controller)design pattern and separates all the system functions in to loose coupled parts, each part is a well wrapped MVC. in Django, it call "app". These parts will work independently or cooperatively in order to achieve a user's request. We will have a backend chapter to reveal the logic mystery, so this section I will only give a general explanation. In Teakwood system, there are mainly eight parts(Apps).

- **Teakwood**: controls the frontend presentation.
- **Accounts**: provides the registration logic.
- **Models**: invoke and control the computing models.
- **Machine**: invoke and control the computing resources.
- **PBS**: guides the PBS script generation.
- **File upload**: gathers the input files to buffer for downloading.
- **Admin**: provies an overall control of user and data.
- **Celery**: third party app, handle synchronous processing.

2.4 Data handling

Teakwood system handles three types of data: the website data, the computing data and the message queue data. For each type of data we provide a different storage. see this table:

Website data	Computing data	Message queue data
MySQL	File server	redis server
Teakwood data	inputs and outputs	Asynchronous handling

Teakwood uses MySQL database for storing the website data, e.g. the user account and the project labels.
For the computing data, Teakwood periodically rsync them to a separate file server for backup and downloading purpose, e.g. input files and the output result.
Message queue data is generated when we use Celery to asynchronous processing time consuming process. Those data are ephemeral, so we simply use a redis server to keep it.

2.5 Remote Configuration

Before the first time we can run a job in HPC or cloud, we have set up a connection between Teakwood and the computing server. Also we have to prepare everything that running a job will need. The main tasks are:

- Establish an password-less ssh login. to the server.
- Compile the tools and packages we will use in the remote machine.
- Ready all the import path for machine configuration Teakwood.
- Configuration Teakwood by the form guidance.

After those steps are done, we've successfully "plugged" Teakwood onto the remote machine. Now everything we interact with the terminal will migrates to the Teakwood web portal.

Chapter 3

Backend Mystery

Teakwood is powered by Django framework, so its backend inherits all the Django backend feathers. For example, Teakwood follows the MVC(Model-View-Controller) design pattern and DRY(Don't Repeat Yourself) principle. When you go through Teakwood's code, you will found that teakwood have a neat and loose coupling coding structure. This chapter will go through Teakwood's four important feathers in the backend: The MTV design pattern, the model to database mapping, the template language and the powerfull admin system.

3.1 MTV Framework

Teakwood has an updated MVC, that is MTV.

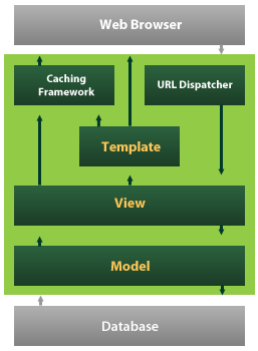

Figure 3.1: Teakwood MTV framework

M represents the model (Model), i.e., **the data access layer**. This layer processing data in all matters related to: how to access, how to confirm the

validity , which behaviors it has, and the relationships between the data.
T Representative of template (Template), i.e., **the presentation layer**. This layer response for how to display the page or other types of documents.
V represents the view (View), **business logic layer**. You can see it as a bridge between the model and the template.This layer calls the model to provide data and sent it template for rendering web page.

As we can see from the figure, Teakwood has a loose coupled modeling design. "M","T", and "V" are separated from each other; Each MTV wrapper(i.e. a Django app) is also separated from each other. This design makes the Teakwood sytem flexible for make changes and easy to extend feathers.

3.2 Models and Database

Teakwood provides an abstraction layer (the models) for structuring and manipulating the data of Teakwood apps. Essentially, a model is a Python object that describes your data model/table.Thanks for Django provides an integrated object relational mapping (ORM) functions, we don't have work with SQL database itself in order to manipulate data. All we should done is manipulate the corresponding Python object. ORM will do the mapping work.
Django provides many APIs to deal with Databases.

3.3 Teakwood Template Language

Djangos template language is designed to strike a balance between power and ease. Its designed to feel comfortable to those used to working with HTML. Django template language is not just HTML file. Here is an example:

```
{% extends "base_generic.html" %}
{% block title %}{{ section.title }}{% endblock %}
{% block content %}
<h1>{{ section.title }}</h1>
{% for story in story_list %}
<h2><a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a></h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

We can see there are some special symbols in the HTML code. In Django template language they are called variables, tags and filters.
Variable: When the template engine encounters a variable, it evaluates that variable and replaces it with the result.
Tag : Tags are more complex than variables: Some create text in the output, some control flow by performing loops or logic, and some load external information into the template to be used by later variables. Some tags require beginning and ending tags.
filter:Filter is basically a restricted variable.

Through variables and tags, static html files becomes dynamic html files and can interact with python code. Django system even provides build-in tags for achieving logical control.

3.4 Powerful Admin

Teakwood comes with a user authentication system which is inherited from Django framework also. The admin system handles user accounts, groups, permissions and cookie-based user sessions.
Teakwood Admin system limited to trusted site administrators, that enables the adding, editing and deletion of site content. Some common examples: the interface you use to post to your blog, the backend site managers use to moderate user-generated comments, the tool your clients use to update the press releases on the Web site you built for them.

3.5 A Practical Case

So far we've already know that Teakwood is loose coupling designed and follows a "MTV"design pattern. Teakwood uses python object to work with database by object oriented mapping. we also know that Teakwood template language is way surpass a static html file with the help of python variables and tags. Now let's try to go trhough all these details to see How Teakwood process a request from user. Fisrt, let's take a look at the Teakwood HTTP request response flow.

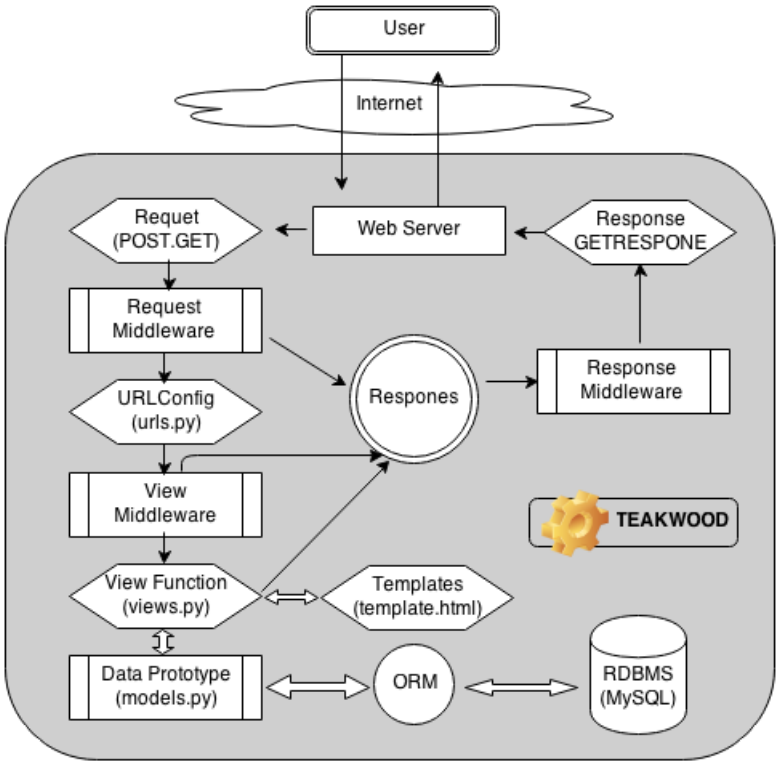

Figure 3.2: Teakwood Request-Response Working Flow

Chapter 4

Use Case

4.1 Job Submission Flow

4.2 Job Monitoring

4.3 Job Report

Chapter 5

Conclusion

The primary goal of this report is to give the reader a general overview on Teakwood. The Deep inside Logic and the Database is hard to extend. Teakwood is written in Python, so it will be good for scientific models to

Chapter 6

Future Work

6.1 Docker Hub

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool.
Any software can be wrapped as a docker for use. Once a Docker is built, it can plugged into any machine for direct use without any system configuration or compiling or so. The only requirement is Docker is installed on the machine.
Teakwood aims to become an Docker hub, so that all users can contribute their Dockers to Teakwood platform. In the future, models deployment for Teakwood can also be a standard procedure.

6.2 Visualization

Visualization is a challenge for Teakwood because output data various, but visualization tools only accept structured data. In the future Teakwood will Create some data parsers for most popular output types. With the help of Visualization Docker, Visualization can may have a big achievement.

6.3 Computing on the go

If your job only requires small scale computing time and you don't have public computing resouces, then "Computing on the go" is right for you.
The ideal is this:
Deploy your models to an pure AMI(Amazon image), and save it as an new

AMI. When you need your models for computation, just connect your AWS account from Teakwood, use the new AMI for initialization, then you model can be used as normal.

Acknowledgments

I would like to say thanks to my committee for their support and encouragement: Dr. Jian Zhang, my committee chair; Dr. Jianhua Chen; and Dr. Konstantin Busch. Especially, I want to express my appreciation to Dr.Jian zhang for giving me useful comments on my project design and report written. I offer my sincere appreciation for the learning opportunities provided by my committee members.
My completion of this project could not have been accomplished without the support of my formal project supervisor Dr.Tao at CCT. Thanks for the project suggestions and providing me models and account for testing my project.
Thanks for all the open source providers: Django framwork, Celery, RabbitMQ, Bootstrap CSS, Bootstrap JavaScript, Studio MX, FAMFAMFAM icons, Google groups, Youtube, JQuery UI.

References

[1] Django framwork. https://www.djangoproject.com/

[2] Python programming language https://www.python.org/

[3] Celery: Distributed Task Queue. http://www.celeryproject.org/

[4] Sphinx Documentation generator. http://sphinx-doc.org/

[5] Message queue broker. http://www.rabbitmq.com/

[6] Science gateway https://www.xsede.org/

[7] Simulating wave nearshore http://swanmodel.sourceforge.net/

[8] LSU HPC: Supermike II http://www.hpc.lsu.edu/