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(Operating System), are gone, and I have to use Linux commands to manipulate those machines and manage my stuff. This 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 found that 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 needs 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 commands. 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 this desire, 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 migrates all the terminal typing work to Teakwood GUI, and wrap

them with interactive web pages. This enables user to submit HPC jobs just by clicking functional buttons. No more terminal commands.
In Teakwood, all user's computing data are hosted in a file server, user can share their result, models and computing resources within their group. This greatly facilitated the collaboration among users.
Teakwood also provides a project management system for user to organize their project data easily.
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.

1.3 Features

Functionally, Teakwood has 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 with document support.

• **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, machines, models and so on.

1.4 System requirements

Teakwood is a Django 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 those dependencies as well as finish setting up all the packages and tools. Below is a list of the required installation during the Teakwood development.

• Project Dependent libs
As mentioned above, install all dependent libs are the first step. Please refer to the 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 php-MyAdmin.

• Version Control
Teakwood uses "git" as version control. Git is a very popular version control software. Teakwood source code is hosted in 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 sphinx and latex need to be installed.

• Celery and RabbitMQ
This is a solution for resolving synchronous process. install and start them before you run Teakwood.

• Django 1.4.x
No need to explain.

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

Chapter 2

Teakwood System

2.1 Overview

Structurally, like most websites, Teakwood system has a three-layer 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 like:

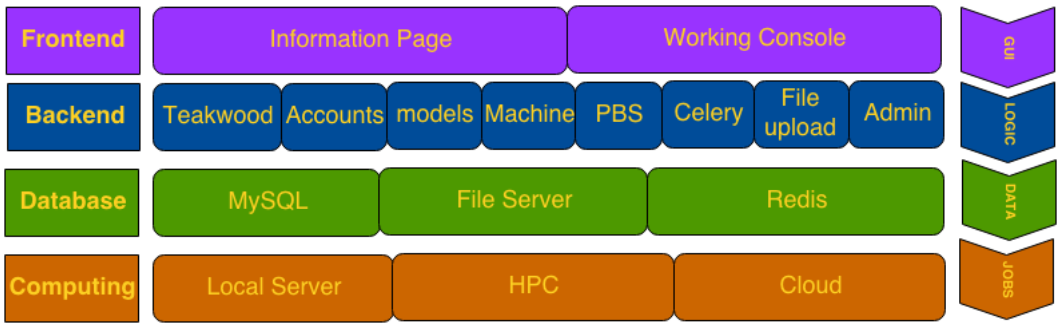

Figure 2.1: Teakwood System Overview

On the above figure, we can see the four straight layers 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 user interact with. Basically all what we see in the Teakwood website can be called the "frontend", that includes the HTML files, Functional butthons, the pictures, etc. For neat purpose, Teakwood separated the frontend into two parts: the **information pages** and the **working console**. See below:

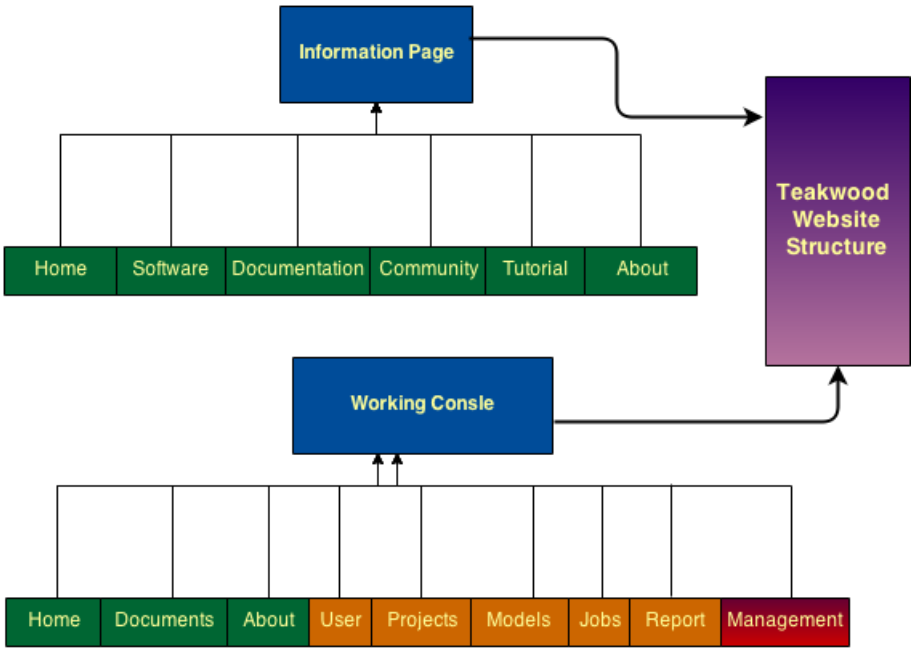

Figure 2.2: Website Strucutre

The **information pages** introduces you all the general information about Teakwood, including:

- 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 project. The functional buttons will guide you to different 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 provides project overview.
- **Models**: Create models and provides models overview.
- **Jobs**:Creates jobs, overview jobs and job monitoring.
- **Report**:Directs you to the output downloading page.
- **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 separated all the system functions in to loose coupled parts, each part is a well wrapped MVC. in Django, it calls "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. This section will only provides 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 uploading.
- **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 Teakwood provides 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 synchronize them to the 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 process time consuming tasks. 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 on HPC or cloud, we have set up a connection between Teakwood and the computing servers. In detail, we have four tasks to do:

- 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 into these remote machines. 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 find that teakwood have a neat and loose coupling coding structure. What makes Teakwood's coding structure so unique? 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.

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 Represents the 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.

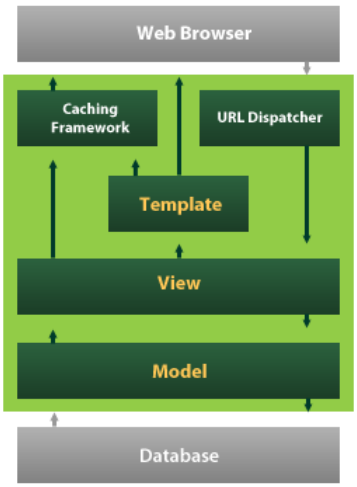

Figure 3.1: Teakwood MTV framework

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 making 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's integrated object relational mapping (ORM) functions, we don't have work with SQL database in order to manipulate data, all we should do is manipulating the corresponding Python object. ORM will do the mapping work.
Django provides many APIs to deal with Databases by invoking corespondent python object.

3.3 Teakwood Template Language

Teakwoods template language is designed to strike a balance between power and ease. Its designed to please both the HTML designer and the Python coder.
Teakwood template language is not just HTML file or Python code. 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 the 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. All these cases can be handled in Admin console.

3.5 A Practical Case

So far we've already know that Teakwood is loose coupling designed and follows a "MTV"design pattern; we also know that Teakwood uses python object to work with database by object oriented mapping; plus, Teakwood template language is powerful for generating HTML files, now let's connect 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.

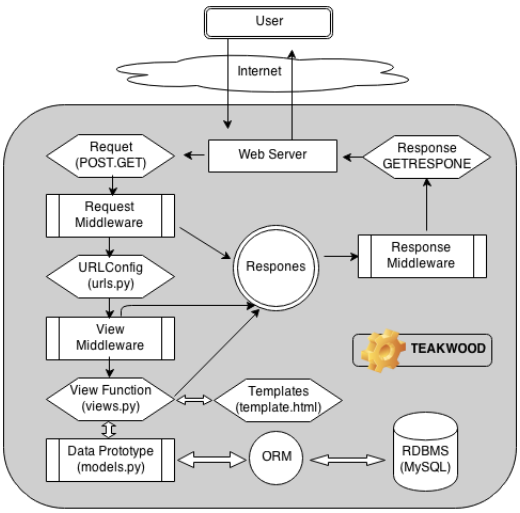

Figure 3.2: Teakwood Request-Response Working Flow

Now let take a real example to see how this structure works. For example, an user want to view all the job status under his/her account. • Click the job button.
• Button triggered an request.
• Request goes to request middleware for identity check, if legal? auth?
• if yes, goes to response and raise error page;
• if no, goes to urlconfig to locate the right URL.
• URL goes to view middleware for legibility check, if wrong URL, auth?
• if yes, goes to response and raise an error page;
• if no, goes to view function.
• View function check if need data? if need template?
• if need data, trigger models to fetch data from database;
• if need template, find the right template, also fetch css, js and imgs.
• Send all files to response and rendering web page.
• Display the final page to user.

Chapter 4

Use Case

This chapter teaches user how to use Teakwood. Teakwood is designed for scientific researcher's daily use, so a simple logic is important. Teakwood created a maximized general flow to fit most computing tools.

4.1 Job Submission Flow

Let's take a look at the classical job submission flow in Teakwood.

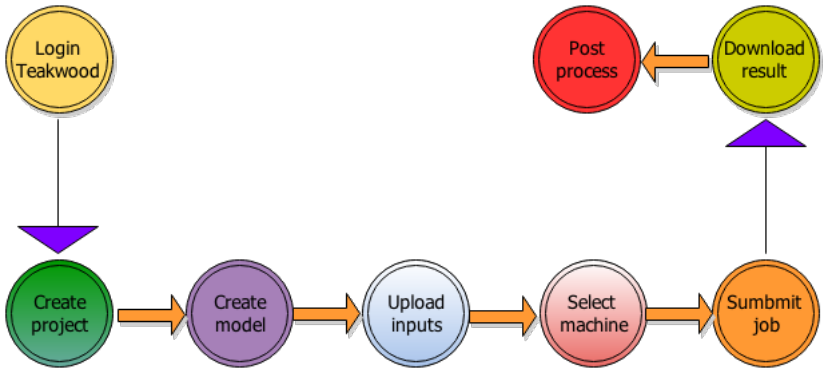

Figure 4.1: Job Submission Flow

In the above figure:

• **Login**: Only logged user can use the working console, visitors only can see the information page.
• **Project**: A project relates to a big problem resolving. A project may use a lot of computing tools(packages), a lot of models, a lot of machine in order

to solve this big problem.
• **Model**:A configuration package guides the computing tool how to use inputs and machines.
• **Inputs**: Data that will be used during the computing.
• **Machine**: Where your job runs.
• **Job**: a single running on a single machine.
• **Result**: Final output of the job running.
• **Post-process**: refine the output data, visualize the data, validate the data, etc.

This job submission figure give you the general working flow on how Teakwood works, if you want a step by step hands on guide, please refer to the video tutorial in homepage.

4.2 Job Monitoring

In order to let user have a better view of their job status, Teakwood provides a five- stage monitoring.

• **Uploading**: means your job is sending to the remove machine.
• **Queued**: means your job is accepted and placed in the to-run list.
• **running**: means your job is running now.
• **Finished**: means you finished your job or your hour is used over.
• **Data Ready**: means your data is synchronized to file server and ready for download.

In project console, teakwood also provides a live job monitoring by displaying the running message on the console. that can gives user more information about the job status.

4.3 Download output

Navigate to the report page and download your output result. you've done all.

Chapter 5

Conclusion

The primary goal of this report is to give readerS a general overview of Teakwood. The backend logic details is not expand greatly, user can refer to the Teakwood developer manual for details.
The general things like:
What is teakwood?
How is Teakwood like?
What can Teakwood do?
What are the features?
How can I tell that Teakwood is different from other web frameworks?
are the main guidance for writing this report.

This report has a moderate system structure explaination on how are the frontend, the backend and the database communicate with each other. Also this report provides tips on how Teakwood connect to the remote server for submitting jobs.
Teakwood is written in Python, the glue language. Python is very popular in mathematical and scientific area; while Teakwood is focus on user experience in scientific computing. This is a good match.
It is good that Teakwood follows the MVC(model-view-controller)design pattern, because adding or deleting feathers in Teakwood are simply and easy.
Thanks to the Django app concept, Teakwood has a flexible and neat coding structure. There is no more repeat coding and deploying and detaching app is easy.

All in all, I hope this report reveals Teakwood to both scientific researchers and software professionals.

Chapter 6

Future Work

• Docker Hub

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. 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 that machine.
Teakwood aims to become a 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.

• Visualization

Visualization is a challenge for Teakwood because output data are various, while visualization tools only accept structured data. In the future Teakwood is working on create general data parser for most popular output types.

• Computing on the go

If your job only requires small scale computing and you don't have public computing resources, 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 all your models can be used instantly without re-compiling. Terminate your cloud service one you done with computing.

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 job's supervisor Dr.Tao at Center for Computation and Technology at LSU. Thanks for the project suggestions and providing me models and account for testing my project.
Thanks for all the open sources 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/