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 the scientific researchers who dont 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 just by clicking buttons. Teakwood is also an open platform that enables user to work co-operatively. Through Teakwood, user can share their models, results, and computing resources within their group and have discussion in Teakwood forum. Teakwood is powered by Django.

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 Cerformance Computing) systems. Suddenly I jumped in to an UNIX environment. I found all those GUIs(Graphic User Interfaces), which I used to when I use Window OS, are gone, and I have to use Linux commands to control those machines and manage my stuff, which makes me uncomfortable and cumbersome. With time goes by, I can work with Linux comfortably eventually. However, I still have the tendency to use GUI when proving me options.
I am not the only one had such experience, In my working building, a lot of Scientists and researchers are suffering this pain. Further, cross-platform work adding them extra wasteful jobs. For example, scientist A want to share a computing result for researcher B to do visualization. A uses Linux and B uses Window. What should they do? command, command, command, copy, paste, command, command ...
Why not create a GUI platform to reduce those repeated typing work and allow them to work cooperatively?
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 its web console, and to have a full control of their job status. Teakwood also provides a file management systems for user to organize their job data easily. What's more, Teakwood enables user to work cooperatively by sharing their result, models and computing resources within their group.

Teakwood migrates all the terminal typing work to Teakwood GUI, enables user to submit the HPC jobs just by simply clicking functional buttons. Teakwood is a web framwork, which means user can access from any type of machine as lons as the machine has a browser.

1.3 Feature

Functionally, Teakwood have the following features:

- **Perfect documentation**

Teakwood homepage provides diverse software information including installation guide, user manual, developer manual,video tutorial, etc. user can easily fetch info and grasp Teakwood soon.

- **Extensible deployment**

Teakwood is loose coupling designed. User can easily add or remove features, models or even python functions without mess the whole system.

- **Neat GUI**

A neat LSU style web console makes your work simple as Windows. Drag, push, click, that's it!

- **Job monitor System**

Job monitor system provides five labels: uploading, queued, running, finish, and Data ready to let the user monitor the job status. Job monitor system also periodically pulls the running message and displays it in the console, user can know more details about their job.

- **Project management system**

All user's project 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, use can activate their models.

1.4 System requirements

Teakwood is a Django powered web framework which integrated a lot of third party python packages. Some of the packages require extra libs or development packages to work, so before run Teakwood, we need to resolve these dependencies. The following installing process is just based on my system experience (my system is Fedora 20 x86$_{6}4withPython2.7.5preinstalled$).
$SystemRequirement$
Since I am doing software development on Virtual Server, "Development tools" is an necessary. Development tools are a yum group, which is a pre-defined bundle of software that can be installed at once, instead of having to install each application separately.during the installation. Please notify that during the installation, they may recall extra libs or '*-devels', just install them accordingly. For example, mine recalled libxsl and libffi.
Database
Teakwood use mysql database. so make sure mysql-server and mysql-devel is installed.
If you want an better control of your database, I recommend phpMyAdmin. The installation in my computer is like this:
Version Control
Teakwood uses git as version control. Git is a very popular version control software. Also, Teakwood code can be found at Github.
Virtual Environment
It is highly recommend that we set up an independent working environment for Teakwood project. I use "virtualenv", install and set up is fairly easy.
Latex+sphinx
Teakwood uses sphinx packge to generate diverse documentations such as html, latex, pdf, etc. so these tools need to be installed.

Chapter 2

Teakwood System

2.1 Overview

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

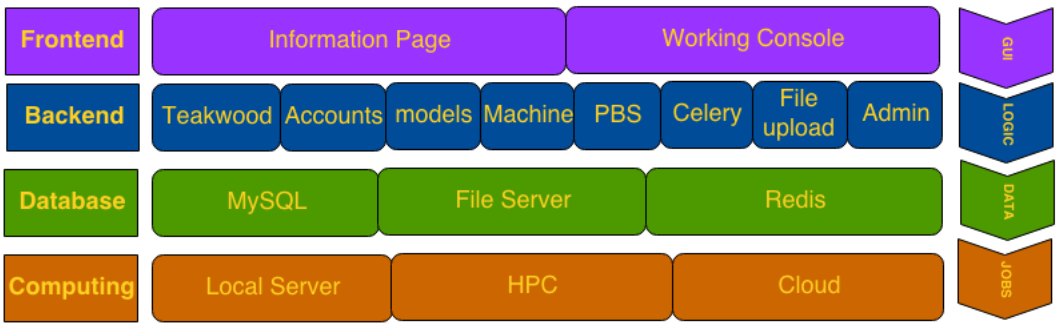

Figure 2.1: Teakwood System Overview

On the above figure, we can see four straight layers bottom up. In each layer, the left part is the layer name, the middle part is the layer content, and the right part is what its actally is. We will go through details one by one.

2.2 Frontend

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

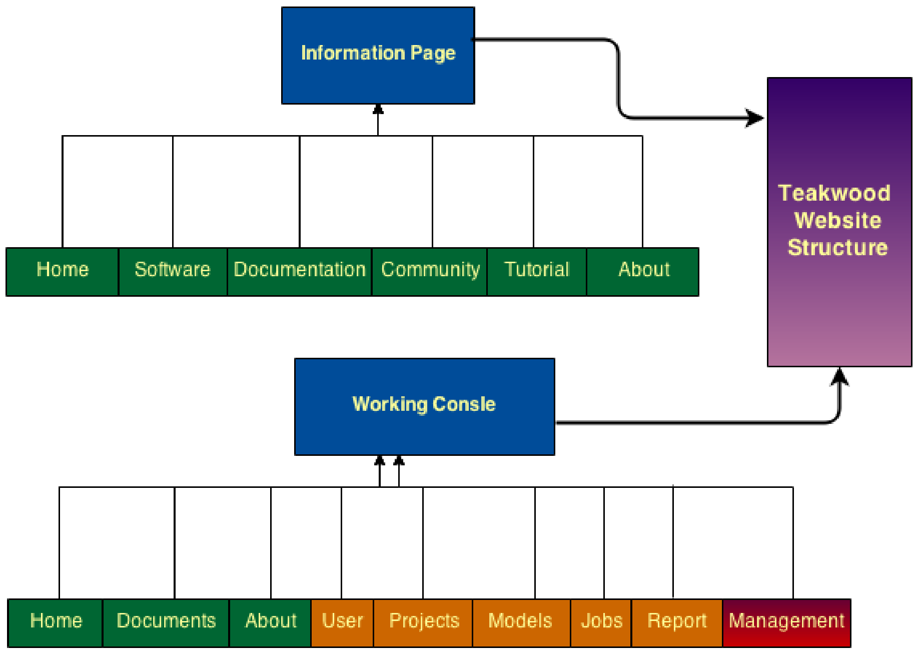

Figure 2.2: Website Strucutre

The infomation page gives you all things about Teakwood.

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

Note the color differences in the bottom layer.

•The Green: all visitor can see and manipulate.
•The orange: only logged user can see and manipulate.

•The red: only superuser can see and manipulate.

This logic is achieved by desiang a if-else template structure, and the authentication function is provided by Django itself.

2.3 Backend

The backend is the logical design of handling data. Teakwood system follows the loose-coupling design manner, so she separated the logic in to different parts, any changes in one part will not mess other parts. (We will have a backend chapter to reveal the logic mystery.) In Teakwood system, there are mainly eight big logic designs.

- **Teakwood**: control the frontend presentation.
- **Accounts**: control the user identification.
- **Models**: invoke and control the computing models.
- **machine**:invoke and control the computing resources
- **PBS**:Guide the PBS script generation.
- **File upload**:achieve the input files gathering function
- **Admin**:overall control of user and data.
- **Celery**:asynchronous handling.

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**
Database	**File server**	**redis server**
data in Teakwood	**nputs and outputs**	**Asynchronous handling**

Teakwood system uses MySQL database for store it website data. Teakwood periodically rsync server data to Teakwood file server for backup purpose. Messge queue key-values are ephemeral data, so we simply use a redis server to keep it.

2.5 Remote Configuration

Before the first time we can use remote computing resources, we have set up a connection and ready everthing. the main things we should done are:

- Establish an passwordless ssh login.
- Compile the tools/packages we will use in remote machine.
- Ready all the import path for Teakwood to use.

Chapter 3

Backend Mystery

3.1 MTV Framework

The apps are modularized. You can remove your application by deleting one directory instead of hunting through several directories deleting each piece.

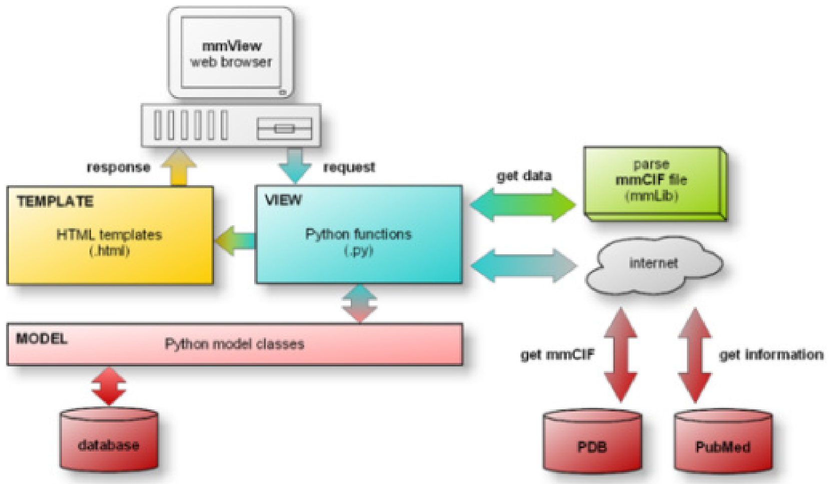

Figure 3.1: Django MTV framwork(cite from internet)

M represents the model (Model), i.e. the data access layer. The layer processing and data in all matters related to: how to access, how to confirm the validity of including which the relationship between behavior and data.
T Representative of T template (Template), i.e., the presentation layer. The layer processing performance related decision: how to display the page or other types of documents.
V represents the view (View), business logic layer. This layer contains the

access models and the transfer of appropriate template logic. You can see it as a bridge between the model and the template.

3.2 Models and Database

Django provides an abstraction layer (the models) for structuring and manipulating the data of your Web application.
In Django, a lot of this hassle is taken care of for you by Djangos object relational mapping (ORM) functions, and how Django encapsulates databases tables through models. Essentially, a model is a Python object that describes your data model/table. Instead of directly working on the database table via SQL, all you have to do is manipulate the corresponding Python object.

3.3 Django 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:

```
{% block navigation %}
<nav id="navigation">
{% with bookpage.book.pages.all as bookpages %}
    <ul class="book-page-list">
    {% for page in bookpages %}
    {% if page != bookpage %}
    <li><a href="{{ page.get_absolute_url }}">{{ page.name }}
    {% else %}
        <li class="active">{{ page.name }}</li>
    {% endif %}
    {% endfor %}
    </ul>
{% endwith %}
</nav>
{% endblock %}
```

Variable A a Tag b c filter dc—d
we can add variables tags and filters into html code.

3.4 Request-Response Flow

How Teakwood process a request from user?

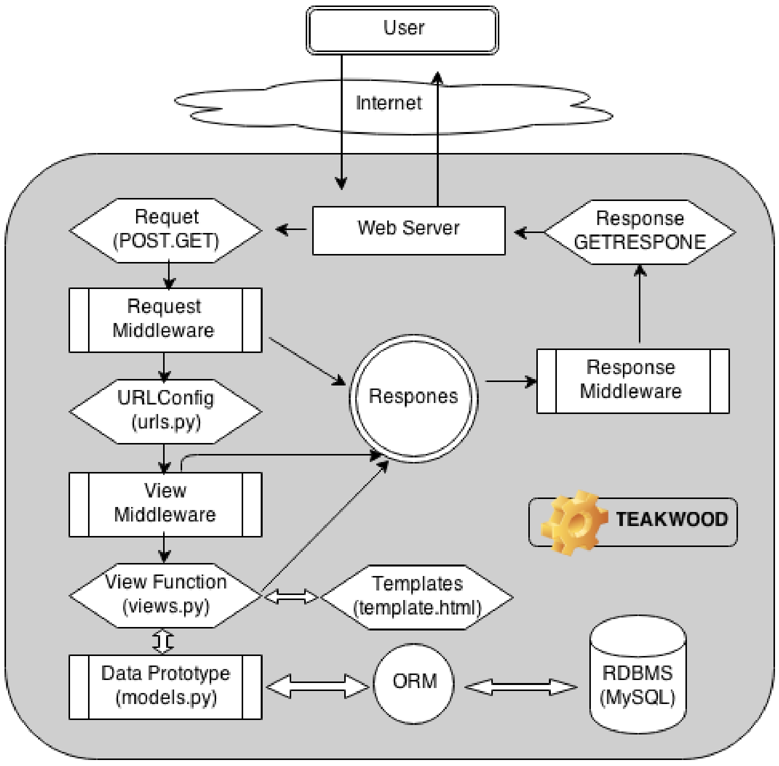

Figure 3.2: ¡Caption here¿

3.5 Powerful Admin

Django comes with a user authentication system. It handles user accounts, groups, permissions and cookie-based user sessions. This section of the documentation explains how the default implementation works out of the box, as well as how to extend and customize it to suit your projects needs.

Chapter 4

Asynchronous Handling

4.1 Celery

Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well. The execution units, called tasks, are executed concurrently on a single or more worker servers using multiprocessing, Eventlet, or gevent. Tasks can execute asynchronously (in the background) or synchronously (wait until ready).

4.2 RabbitMQ

Messaging enables software applications to connect and scale. Applications can connect to each other, as components of a larger application, or to user devices and data. Messaging is asynchronous, decoupling applications by separating sending and receiving data.

Chapter 5

Use Case

5.1 Job Submission Flow

5.2 Job Monitoring

5.3 Job Report

Chapter 6

Conclusion

Teakwood provides a solution and beyond. Teakwood is a framework that migrate 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 just by click buttons. Teakwood is also a platform that let users work cooperatively. Through Teakwood, users can share their models, outputs, and computing resources to people within the group. Teakwood is powered by Django.

Chapter 7

Future Work

7.1 Docker Hub

7.2 Visualization

7.3 Computing on the GO

Acknowledgments

¡Acknowledgements here¿

¡Name here¿

¡Month and Year here¿
National Institute of Technology Calicut

References

[1] ¡Name of the reference here¿, `<urlhere>`

[2] ¡Name of the reference here¿, `<urlhere>`