SMACK Energy Forecasting

Low Level Design DocuMent

2015

Table of Contents

[Table of Contents 1](#_Toc433918238)

[Project Purpose: 2](#_Toc433918239)

[Design Overview: 2](#_Toc433918240)

[Diagram: 2](#_Toc433918241)

[Principle Components: 2](#_Toc433918242)

[Design Details: 3](#_Toc433918243)

[A - Cloud Based Deployment Solution 3](#_Toc433918244)

[Hosting Provider 3](#_Toc433918245)

[Hosting Infrastructure 3](#_Toc433918246)

[Deployment Scripts 4](#_Toc433918247)

[Transferability 4](#_Toc433918248)

[B – Scalable Cluster Computation Framework 4](#_Toc433918249)

[Master Name Server + Web Host 5](#_Toc433918250)

[*N* Worker Nodes 5](#_Toc433918251)

[C – Scalable Storage System 5](#_Toc433918252)

[D – Web Driven User Interface 5](#_Toc433918253)

[E – Data Access API 6](#_Toc433918254)

[Additional Components: 6](#_Toc433918255)

[Licensing: 6](#_Toc433918256)

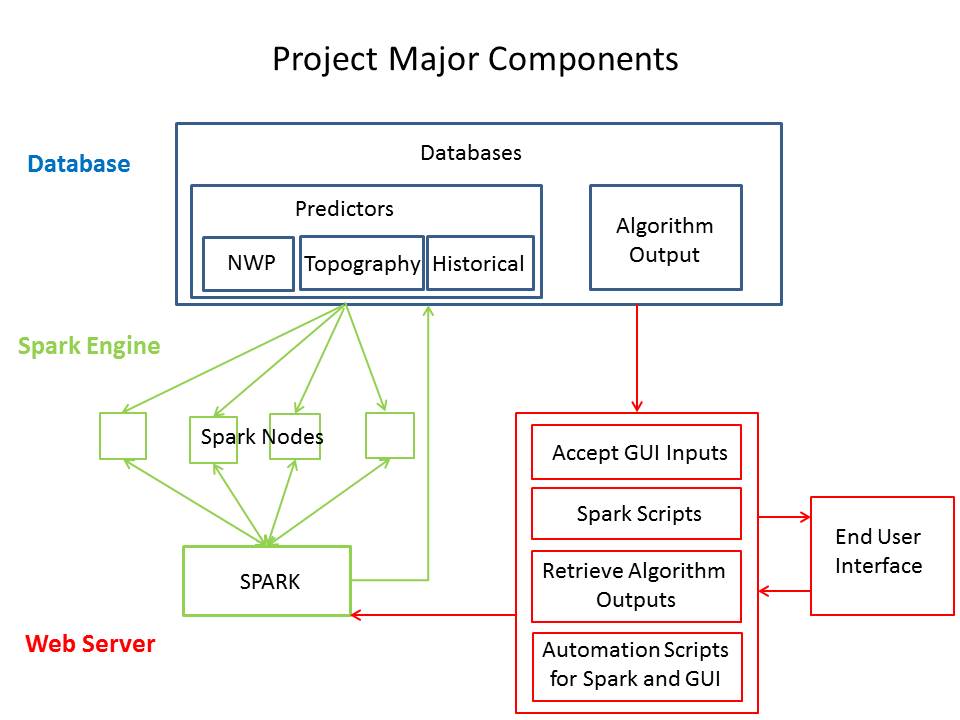
[References: 6](#_Toc433918257)

# Project Purpose:

<- Outline the Objectives of the Project and the Final Realizable Goal ->

# Design Overview:

## Diagram:



## Principle Components:

The project consists of the following principle components:

1. Cloud Based Deployment Solution
   1. Deploys and Initializes all components onto the cloud host.
   2. Allows for a “turn-key” solution to get a spark cluster running
2. Scalable Cluster Computation Framework
   1. Adaptable Cluster Framework for running simulations and models across many nodes for increased efficiency.
3. Scalable Storage System
   1. Scalable storage designed to accommodate heavy data usage and cluster computing
4. Web Driven User Interface
   1. A presentation platform that can be used to stream data both real-time and historical for both marketing and forecasting purposes.
5. Data Access API
   1. An API to access predictive data directly for download and readout.

# Design Details:

Each of the following sections outlines the various details regarding their respective principle components:

## A - Cloud Based Deployment Solution

This component has the following breakdown:

1. Hosting Provider
2. Hosting Infrastructure
3. Deployment Scripts
4. Transferability

### Hosting Provider

The hosting provider chosen is the local company (Calgary, AB) Cybera. Rapid Access Cloud (RAC) is a service offered by Cybera for the purpose of pre-profit and pre-production use in order to help promote SME’s within the province. We chose to go with Cybera due to the locality as well as the cost. Commercial alternatives were beyond the scope of our project’s budget.

The following is offered for free as part of their Rapid Access Cloud (RAC) service.

* 8 Free Virtual Machine (VM) Instances that can be deployed at will.\*
* One Floating-IP that can be used as a static IP for external connection.\*
* Automatically configured CentOS-6 for the cloud.
* OpenStack Infrastructure-as-a-Service (IaaS) API Access
* Unlimited Usage Bandwidth
* Adjustable RAM and Virtual CPUs

\* With permission from Cybera, it may be possible to increase usage allotments (at least for short periods of time)

### Hosting Infrastructure

As part of Cybera’s hosting services, OpenStack is used as their cloud platform. This requires that parts of the deployment solution be tailored towards OpenStack. OpenStack is a widely accepted and recognized cloud platform – it is open source software and it is widely compatible with enterprise versions of Linux.

OpenStack is known as an Infrastructure-as-a-Service (IaaS) and it provides many utilities and processes to manage the deployment of the VMs that are being used by subscribers of Cybera’s services. Through the use of the Command-Line-Interface (CLI) tools provided as part of the OpenStack client suite, it is possible to create VM instances, install OS images onto them, destroy them, as well as manage the storage services provided. Through these CLI tools it becomes possible to develop scheduled jobs and automated deployment of much of the configuration.

### Deployment Scripts

As part of the project, it will be the intent to develop numerous automation scripts for the purpose of deploying and destroying clusters through a simple and intuitive method. The scripts will handle the various tasks related to instantiating a cluster and performing all the necessary installation steps to prepare an environment for simulation.

The following is a breakdown of the scripts that will be deployed:

**Initialization**:

1. Deploy Cluster
   1. Deploy Nodes
   2. Deploy Database
   3. Deploy Spark
   4. Deploy Web UI
   5. Populate Database (Grib2)

**Shutdown**:

1. Shutdown Cluster
   1. Depopulate Database (Grib2)
   2. Shutdown Web UI
   3. Shutdown Spark
   4. Shutdown Database
   5. Shutdown Nodes

**Maintenance**:

1. Schedule Jobs
   1. Gather Grib2 Data
2. Miscellaneous
   1. Add Node
   2. Add Database
   3. Add Model

### Transferability

When pre-production usage has finished and the need for a commercial host arrives, there will be little trouble in transferring the deployment over to another OpenStack provider. Should attempts be made to transfer to a different infrastructure, some additional work would need to be done to make the deployment scripts compatible, so it is in the best interest of future operability to look into other OpenStack providers should the project wish to go commercial.

## B – Scalable Cluster Computation Framework

This component is broken up into the following:

1. A Master Name Server + Web Host
2. *N* Worker Nodes

### Master Name Server + Web Host

This will be the central node which will control and command the spark cluster. It will act as the spark master node. This node will also be the location of the web server and it will be able to access whatever data is desired. The functionality will consist of:

1. Spark Master Name Server
2. Open R Revolution (R Programming Language Environment)
3. Shiny Web Server

### *N* Worker Nodes

These will be the numerous *slave* nodes which are controlled by the central name server. The number will vary, but they will each be cloned from the same image and will be launched with the cluster in number as decided at the time of launch. There may also, time permitting, be functionality to allow for the addition of worker nodes to existing clusters during runtime, but that is not expected initially.

Their functionality is:

1. Scalable in number
2. The ability to use the MLLib (Machine Learning Library) to perform modeling calculations using Spark
3. Distributed job distribution allowing for parallel computation

## C – Scalable Storage System

The project will incorporate a scalable storage system which will scale with our requirements and will allow for redundancy should it be required for additional security and dependability. The chosen platform will be the Swift Object Storage system as part of the OpenStack framework.

Main Functionality:

1. Can be automatically replicated across numerous locations in the cloud for dependability and resilience to failure.
2. Can be accessed both internally from Spark and externally as both a scheduled job and as an initialization script.
3. Thanks to Cybera’s agreement with us, we have access to more than enough space using their hosting service, Rapid Access Cloud.

## D – Web Driven User Interface

The web driven user interface (UI) will be used to present the modeling capabilities of the cloud cluster without necessarily giving them access to the raw data or our predictions. This will allow for marketing and presentation in a user-friendly and eye-catching manner which can be used to gain acknowledgement and expand our audience.

Main Functionality:

1. Will be deployable as part of the initial deployment solution and will automatically configure for the use and presentation.
2. Will feature an authentication method whereby access to certain data can be limited as seen fit.
3. Will run on a combination of PHP and R as the general server-side back-end to facilitate authentication, data generation, and communication to additional services such as the Spark engine and the databases.
4. As part of the R implementation the main UI platform will be constructed using the R package Shiny from RStudio along with charting libraries acquired from GitHub and the CRAN servers.
5. The SparkR library will also be used in order to interface with the Spark cluster in order to utilize SparkSQL capabilities.
6. CSS and additional HTML and JavaScript will be incorporated as needed for aesthetics and interactivity, time permitting.

## E – Data Access API

As our final functionality to the project should time permit and the remainder of the objectives have been met, the addition of a data access API will be incorporated. The purpose of this API will be to offer direct access to our predictive data should they not wish to use the UI, or should they wish to download it. This could be done via subscription and would feature some sort of authentication in order to offer security. This could be implemented using a public-private key encryption method, but that is beyond the scope of this design as of this stage.

# Additional Components:

Over the duration of this course, the SMACK project will be incorporating the use of various free-to-use tools and pre-production licenses. The current stage of the project has seen the use of the following tools:

1. Slack – A communication platform
2. GitHub – A software repository
3. Google Drive – A shared file server

# Licensing:

Many of the tools we have chosen to use are open source and allow for both commercial and non-commercial use. Where possible, the open source solution has been chosen in order to facilitate the use of it for future use.

# References:

GitHub: <http://www.github.com>

Slack: <http://www.slack.com>

Google Drive: <http://drive.google.com>

Cybera: <http://www.cybera.ca>

OpenStack: <http://www.openstack.org>

Spark: <http://spark.apache.org>

Shiny: <http://shiny.rstudio.com>