Cloud Computing
Major Project Report
ON DEMAND HADOOP PART-2 (FRONT END)

Project Mentor
Dharmesh Kakadia

TEAM-MEMBERS
Saumya Choudhary(201001004)
Surya Baghrecha(201001115)
Saumya Dwivedi (201001126)
Sharika Pongubala(201001144)

ON DEMAND HADOOP:

This project exposed Hadoop on cloud via Openstack and EC2 API's. The project was split in to two parts and our team Bazinga focused on the Front end Aspect.

AIM:

Building a user interface consuming REST services from On Demand Hadoop Part-1 Project and openstack. It also displays job progress, logs, statistics and admin interface.

TECHNOLOGY USED:

We have used Django 1.4.2 for building up the interface and developed it with Hadoop 2.0.1-alpha aka Yarn.
Openstack Keystone
Pycurl, python 2.x
Web HDFS

BACKGROUND AND ASSUMPTIONS:

1) User(Hadoop) job details are taken and XML file is created accordingly. Since we could not test the XML file with Oozie (installation problems), we have just created a script that would generate a XML file based on some assumed values – which can be modified accordingly.
2) The project is deployed on cloud. Currently the authentication is based on Openstack account(Keystone) where the cloud provider is SERL-lab account 10.2.4.129. For testing purposes, one can login as (user) hadoop1 and (password) hadoop123.
3) For a user, additional features like user quota(current and remaining) and adding VM's to a job has been provided. Please note that these features could be explicitly tested once integrated with the back-end team.
4) User can add as many VM's he/she wants(from his quota) but all of them would be of same type or flavor.
5) Logs show all the files in the log folder of hadoop yarn.
6) Since we did not have a specific admin account, you can directly enter as admin by entering URL 127.0.0.1:8000/adminhome. An admin can view all the jobs of all the user running and succeeded in the homepage and can also navigate through the details.

PROJECT IN DETAIL:

1) ADD NEW JOB: Name- User has to enter the job name for example 'wordcount'. The jar file

has to be uploaded by the user ---for testing purposes we can upload the 'hadoop-mapreduce-examples-2.0.2-alpha.jar' file from yarn/hadoop/share/hadoop/mapreduce. However the user can enter his own valid jar file too.

2) BROWSE FILESYSTEM: This feature allows the user to select the input and output directory for the hadoop job in HDFS. The input data is assumed to be present in the input directory. Since output directory can not exist already, a temporary directory is created everytime where the results are stored. Once the job finishes/halts finally contents are copied to a newly created directory with the name of output_application_id and the temporary directory is deleted again.

   Submit job and you can see in your running server terminal that the job has started. Alternatively you can view the currently running jobs via JOB INFO.

3) JOB INFO: It will give you the status of all running finished jobs which it fetches via web HDFS port 8088 and 19888 of HadoopYarn. It has jobId, jobName, status, startTime and finishTime. For the admin , there are extra fields of username and edit and delete buttons are also provided.

4) DETAILED JOB INFO: Once you click on any one job id, you will get detailed information about the job like the map and reduce tasks, queues and their progress via a progress bar.

5) ADMIN ADVANCED FEATURES: We have provided a functionality for the admin to add resources to currently running job by clicking on the edit option aside every job. This open up a page where he/she can specify the number of VM's and provide a custom script to configure the VM's as namenode, resourcemanager etc

   The admin also can specify the flavors of the VM's by uploading similar type of file.

CONFIGURATION DETAILS:

More attention has been given to providing all the features that the front end will demand in near future once it is integrated with the back-end. For demonstration purposes, we have built our own script for accepting and running hadoop jobs, however once XML file is created it is sent to back-end team where they process it for scheduling. Some front-end features might not be functional as they require the back end elements to become operational.

To run the django server:
Download the tar version of project, untar and navigate to new/major. Run command

```
python manage.py runserver
```

Please note that you must have Hadoop Yarn running.
All the minor configuration changes in between the files are mentioned in a README provided in the same folder.

All the HTML pages are stored in new/major/major/template and corresponding views in new/major/major/app/views.py

The script to run hadoop job is start_job.py and to configure the settings config_hadoop.py.
The job details are extracted via web hdfs and the scripts are historyServer.py, jobBrowser.py, resourceManager_app.py and resourceManager_cluster.py. The corresponding configuration settings

are in configure.py

All the above files are in new/major folder.
The uploaded files go in media/documents and authentication done by curl.py

ERROR AND PROBLEMS:
The testing has only been done on localhost and single node cluster.
Filebrowser feature might not work depending on you proxy conditions. If clicking on File Browser gives you an error page then check if you are able to browse namenode port 50070 and from there if you are able to browse the filesystem option.
Admin page has to opened directly with the address, no authentication added till now.
Setting up Yarn could be a problem. Sample start and stop yarn script provided.