IITB Summer Internship 2014

Project Report

Data Analytics Workbench For Educational Data

Principal Investigator

Prof. D.B. Phatak

Project In-Charge

Mr. Nagesh Karmali

Project Mentors	Project Team Members
Ms. Firuza Aibara	Mr. Ankush Arora
Mr. Vamsi Krishna	Ms. Palak Agrawal
Mrs. Sukla Nag	Mr. Prashant Gupta
	Ms. Purva Bansal
	Mr. Saatvik Shah

July 2, 2014

Summer Internship 2014
Project Approval Certificate

Department of Computer Science and Engineering

Indian Institute of Technology Bombay

The project entitled **Data Analytics Workbench For Educational Data** submitted by Mr.Ankush Arora, Ms.Palak Agrawal, Mr.Prashant Gupta, Ms.Purva Bansal, Mr.Saatvik Shah, is approved for Summer Internship 2014 programme from 10th May 2014 to 6th July 2014, at Department of Computer Science and Engineering, IIT Bombay.

Prof. Deepak B. Phatak	Mr. Nagesh Karmali
Dept of CSE, IITB	Dept of CSE, IITB
Principal Investigator	Project In-charge

External Examiner

Place: IIT Bombay, Mumbai
Date: July 2, 2014

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mr. Ankush Arora
NIT, GOA

Ms. Palak Agrawal
LNMIIT, JAIPUR

Mr. Prashant Gupta
ABV-IIITM, GWALIOR

Ms. Purva Bansal
LNMIIT, JAIPUR

Mr. Saatvik Shah
MNIT, JAIPUR

Date: _____

Abstract

EdX is a massive open online course (MOOC) destination site and online learning platform that is open sourced. EdX hosts online university-level courses in a wide range of disciplines to a worldwide audience at no charge, and conducts research into learning.EdX-platform generates tremendous amounts of data related to student, instructor, staff and course.Analysis of edX data enables us to answer the fundamenal questions about learning nature of students. For Data analysis of the edX data, the generated data must be extracted from different sources, cleaned and properly structured and finally stored in the Hadoop ecosystem because of its Big Data nature. This report elaborates and summarizes the working of a Big Data Workbench created at IIT Bombay for managing such voluminous data to obtain structured results from unstructured and unarranged data sources. Additionally it further enhances user experiance by providing a set of tools for visualising this data so that both researchers and professors can infer vital results with respect to EdX data. The workbench has been optimized using the latest Big Data technologies to create both an efficient and fault tolerant arhitecture.

Acknowledgement

We, the summer interns of the team Data Analytics Workbench for educational data, are overwhelmed in all humbleness and gratefulness to acknowledge our deep gratitude to all those who have helped us put our ideas to perfection and have assigned tasks, well above the level of simplicity and into something concrete and unique We, whole heartedly thank Prof. D.B. Phatak for having faith in us, selecting us to be a part of his valuable project and for constantly motivating us to do better. We are very thankful to our project incharge, Mr. Nagesh Karmali, Mrs. Sukla Nag and our mentor Mr. Vamsi Krishna for their valuable suggestions. They were and are always there to show us the right track when needed help. With help of their brilliant guidance and encouragement, we all were able to complete our tasks properly and were up to the mark in all the tasks assigned. During the process, we got a chance to see the stronger side of our technical and non-technical aspects and also strengthen our concepts. Here by, we gladly consider ourselves to be the most fortunate batch of interns. Last but not the least, we whole heartedly thank all our other colleagues working in different projects under Prof. D.B Phatak for helping us evolve better with their critical advice.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Educational Data Mining In Big Data

Educational Data Mining [17][18] refers to techniques, tools, and research designed for auto- matically extracting meaning from large repositories of data generated by or related to peoples learning activities in educational settings. Quite often, this data is extensive, fine-grained, and precise.

1.2 What is EDX?

EdX is a non-profit organisation which is formed by the collaboration of Harvard and MIT.Open EdX is an open source platform for building MOOCs with various advanced features to make the online education more effective.EdX organization itself has a MOOC named as edX which host courses from various institutes across the world.

1.3 Improving Trend in EDM

As numerous articles in both the academic and popular press have pointed out, the ability of eDX to generate a tremendous amount of data opens up considerable opportunities for educational research. edX and Coursera, which together claim almost four and a half million enrollees, have developed platforms that track students every click as they use instructional resources ,complete assessments ,and engage in social interactions. These data have the potential to help researchers identify ,at a finer resolution than ever before ,what contributes to students learning and what hampers their success. eDX generates large amounts of data which needs to be processed. Why collect and store large amounts of data if you cant analyze it in full context? Or if

you have to wait hours or days to get results? The idea is that if we know everything about a student, well be better able to help that student and future students who fit a similar profile.

1.4 Crunching the numbers

There was a mock test held at IITBombayX on 14th May

- 80 students participated
- 1-2 hours long per student
- Interacted with IITBombayX but only gave quizzes
- 100-120 questions answered
- 1.5GB of Data Generated
- 1.5/80 GB generated per student interacting with the system for 2 hours

An average EdX Course

- 40,000 participants [19]
- 3-4 hours per week
- Interact with forums,quizzes,videos,etc
  Considering the above data (1.5/80)*40000 => 750GB per week!
  Still doesnt look like a Big Data Application?
- 2-3 courses per institute
- 3*750=2250GB per Week!

Surely an application of Big Data!

1.5 Useful Results from Terabytes of Data

A data analytics workbench is required to setup a workflow to take different types of data generated as a source pipeline the data and process accordingly according to its type to ensure that it is ready to be transformed and loaded.The logs generated are unstructured so they need to be structured and cleaned so that they can be queried for useful information. It is also needed for analysing the large amounts of data

that is generated and thus derive some useful results from it. Data visualisation is a modern branch of descriptive statistics. It involves the creation and study of the visual representation of data, meaning information that has been abstracted in some schematic form, including attributes or variables for the units of information.To provide with a better understanding of the results obtained from the analysis of the eDX data,data visualization can be used. The second aim of this project is to make the data analytics workbench suitable for eDX data such that it is able to provide visualizations for different results.

Chapter 2

Installation

This Chapter covers all the description and installation of all the tools required in this project. All the softwares and packages used are Open Source software and are freely available.

2.1 System Requirements

For Single Node : 2.1 describes the hardware requirements for the Data Analytics System. We have developed the whole system on the following hardware.

Type of hardware	**Hardware requirements**
Hardware	• Dual core Intel Pentium compatible processor or multiprocessor-based computer with a 2Ghz or greater processor • 64-bit system • Network interface card
Disk space	5 GB free disk space (minimum). Requirements increase as data is gathered and stored in HDFS.
Memory	4 GB or more (recommended)

Table 2.1: Hardware Requirements

2.2 describes the basic software requirements we have used.

Type of software	**software requirements**
Operating System	Linux (Ubuntu 14.04 LTS)
Web browser	Mozilla Firefox 28.0
Text Editor	Vim, gedit
Optional software	Rstudio, edX-platform, Robomongo

Table 2.2: Software Requirements

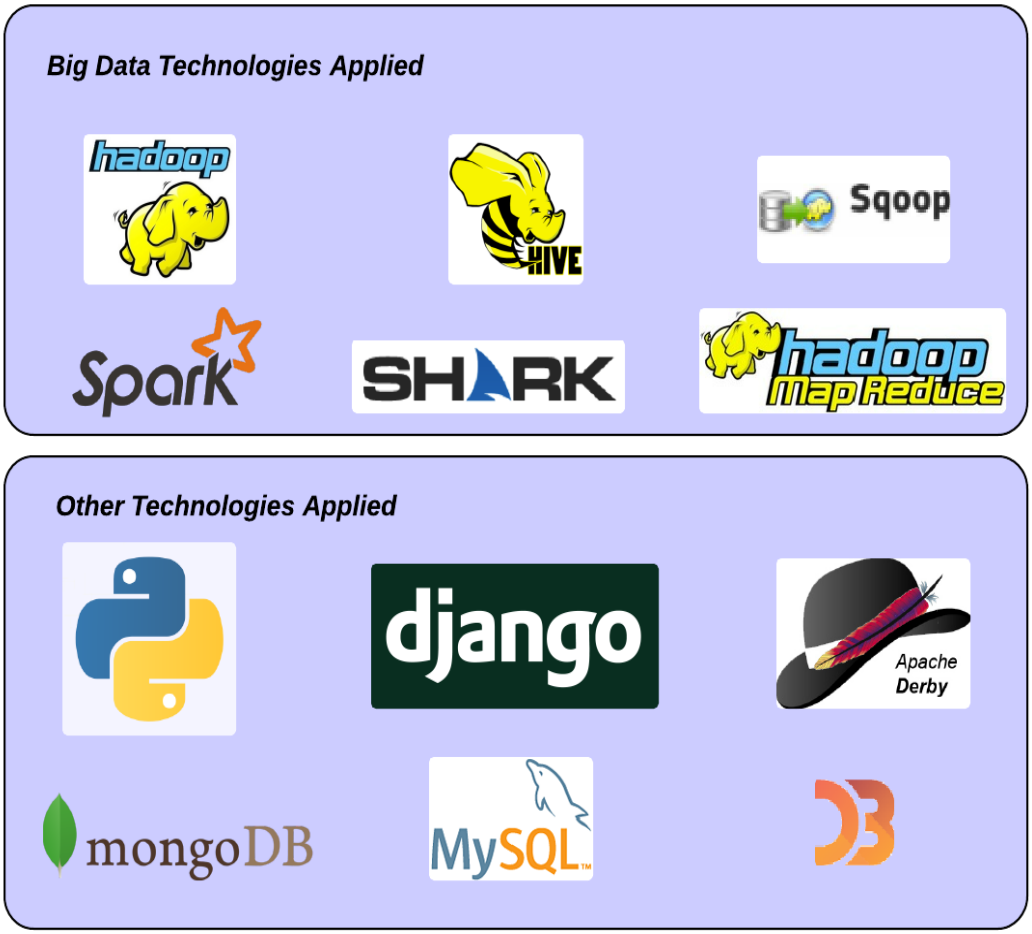


Figure 2.1: Tools for Big Data

2.2 Hadoop 2.2.0

2.2.1 Hadoop [1][2][3]

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware.The Apache Hadoop framework is composed of the following modules:

- Hadoop Common : Contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS) : A distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- Hadoop YARN : A resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.
- Hadoop MapReduce : A programming model for large scale data processing.

2.2.2 Hadoop Stack

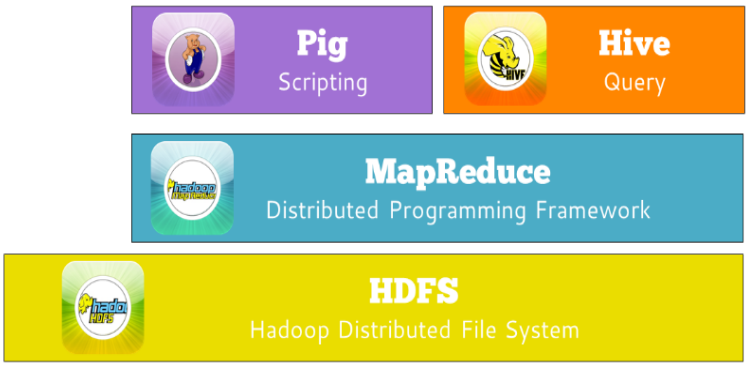


Figure 2.2: Hadoop Technology Stack

The Hadoop Technology Stack is made of the following components

1. *HDFS*
   The Hadoop Distributed File System is the customized file system made for the Hadoop Ecosystem which supports large block sizes and coordinates storage between multiple DataNodes

2. *MapReduce*
   Programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster

3. *Pig*
   Scripting Language that allows people using Hadoop to focus more on analyzing large data sets and spend less time having to write mapper and reducer programs

4. *Hive*
   Hive allows SQL developers to write Hive Query Language (HQL) statements that are similar to standard SQL statements; now you should be aware that HQL is limited in the commands it understands

2.2.3 Steps for installation

- Version : Hadoop 2.2.0
- OS Used : Ubuntu 14.04

2.2.4 Prerequisites

1. Setup OpenSSH Server
   `sudo apt-get install openssh-server`

2. Install and Setup JDK 7
   ```
   sudo apt-get install openjdk-7-jdk
   ln -s /usr/lib/jvm/java-7-openjdk-amd64 jdk
   ```

3. Group and Users for Hadoop Ecosystem
   ```
   sudo addgroup hadoop
   sudo adduser --ingroup hadoop hduser
   sudo adduser hduser sudo
   ```

4. Setup SSH Certificate for `hduser`
   ```
   ssh-keygen -t dsa -P '' -f  /.ssh/id_dsa
   cat  /.ssh/id_dsa.pub
   >>  /.ssh/authorized_keys ssh localhost exit
   ```

5. Disable IPv6
   Edit the `/etc/sysctl.conf` and append the following
   ```
   net.ipv6.conf.all.
   disable_ipv6 = 1
   net.ipv6.conf.default.disable_ipv6 = 1
   net.ipv6.conf.lo.disable_ipv6 = 1
   ```
   Now reboot your machine
   Note : IPv6 enabled/disabled can be checked by
   ```
   cat /proc/sys/net/ipv6/conf/all/
   disable_ipv6
   ```
   This gives a value of `1` if IPv6 is disabled else 0

2.2.5 Main Installation

1. Switch to `hduser`
   `su - hduser`

2. Download and Extract Hadoop 2.2.0
   `wget`
   `http://apache.mirrors.lucidnetworks.net/hadoop`
   `/common/stable/hadoop-2.2.0.tar.gz sudo tar vxzf hadoop-2.2.0.tar.gz`
   `-C /usr/local`
   `cd /usr/local`
   `sudo mv hadoop-2.2.0 hadoop`
   `sudo chown -R hduser:hadoop hadoop`

3. Setup Environment Variables for Hadoop
   Add the following entries to .bashrc
   ```
   export JAVA_HOME=/usr/lib/jvm/jdk/
   export HADOOP_INSTALL=/usr/local/hadoop
   export PATH=$PATH:$HADOOP_INSTALL/bin
   export PATH=$PATH:$HADOOP_INSTALL/sbin
   export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
   export HADOOP_COMMON_HOME=$HADOOP_INSTALL
   export HADOOP_HDFS_HOME=$HADOOP_INSTALL
   export YARN_HOME=$HADOOP_INSTALL
   export HADOOP_COMMON_LIB_NATIVE_DIR=
   $HADOOP_INSTALL/lib/native
   export HADOOP_OPTS="-Djava.library.path=
   $HADOOP_INSTALL/lib"
   ```

4. Replace JAVA_HOME value in
   `/usr/local/hadoop/etc/hadoop/hadoop-env.sh` with
   `/usr/lib/jvm/jdk/`

5. Restart the Shell session by `exit` and `su - hduser`

6. Test Hadoop Version
   `hadoop version`

2.2.6 User Based Configurations

Below Configuration is for Single Node Cluster
Go to `/usr/local/hadoop/etc/hadoop`
Rename `mapred-site.xml.template`
`to mapred-site.xml`
`mv mapred-site.xml.template`
`mapred-site.xml`

Configure the following files

- **core-site.xml**
  Paste the following between <configuration>
  ```
  <property>
  <name>fs.defaultFS</name>
  <value>hdfs://localhost:9000</value>
  </property>
  ```

- **yarn-site.xml**
  Paste the following between <configuration>
  ```
  <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
  </property>
  <property>
  <name>yarn.nodemanager.aux-services.
  mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.
  ShuffleHandler</value>
  </property>
  ```

- **mapred-site.xml**
  Paste the following between <configuration>
  ```
  <property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
  </property>
  ```

- **hdfs-site.xml**
  Paste the following between <configuration>
  ```
  <property>
  <name>dfs.replication</name>
  <value>1</value>
  </property>
  <property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/home/hduser/mydata
  /hdfs/namenode</value>
  </property>
  <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/home/hduser/mydata/
  hdfs/datanode</value>
  </property>
  ```

2.2.7 Prepare Namenode/Datanode

Create and Format Namenode and Datanode Directory
```
mkdir -p mydata/hdfs/namenode
mkdir -p mydata/hdfs/datanode
hdfs namenode -format
```

2.2.8 Start HDFS

The scripts to start HDFS are :
start-dfs.sh
start-yarn.sh
Test Hadoop Daemons by writing `jps` on the command line
Output should be
```
Namenode
Datanode
Secondary Namenode
ResourceManager
NodeManager
```

2.2.9 How Hadoop works?

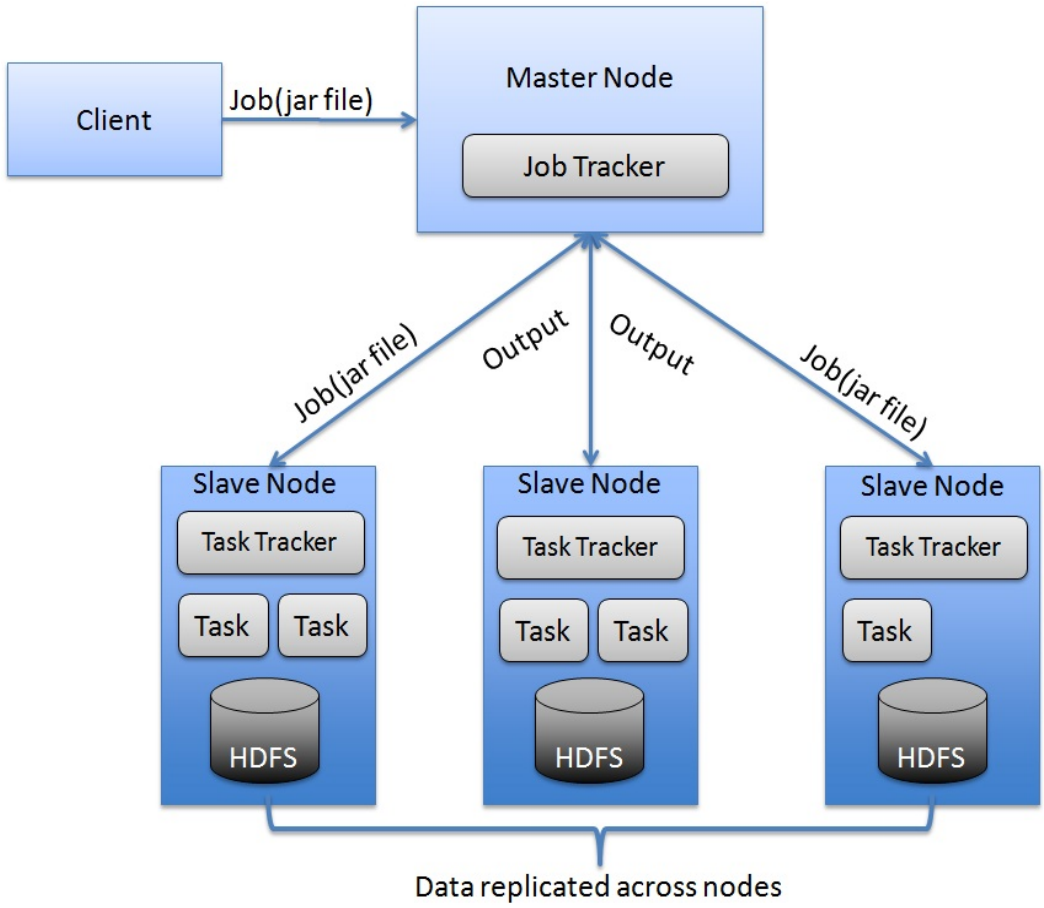


Figure 2.3: Multinode Hadoop Architecture

Hadoop and its various components fit together to ensure a fault-tolerant,durable and highly efficient model for storage and management of Big Data

- *Namenode*
  Namenode is the node which stores the filesystem metadata i.e. which file maps to what block locations and which blocks are stored on which datanode.
  The namenode maintains two in-memory tables, one which maps the blocks to datanodes (one block maps to 3 datanodes for a replication value of 3) and a datanode to block number mapping.
  Whenever a datanode reports a disk corruption of a particular block, the first table gets updated and whenever a datanode is detected to be dead (because of a node/network failure) both the tables get updated.

- *Secondary Namenode* The secondary namenode regularly connects to the primary namenode and keeps snapshotting the filesystem metadata into local/remote storage. It does so at a poor frequency and should not be heavily relied on.

- *Datanode* The data node is where the actual data resides.
  Points to Note:

1. All datanodes send a heartbeat message to the namenode every 3 seconds to say that they are alive.
2. If the namenode does not receive a heartbeat from a particular data node for 10 minutes, then it considers that data node to be dead/out of service and initiates replication of blocks which were hosted on that data node to be hosted on another data node.
3. The data nodes can talk to each other to rebalance data, move and copy data around and keep the replication high.
4. When the datanode stores a block of information, it maintains a checksum for it as well.
5. The data nodes updates the namenode with the block information periodically and before updating verify the checksums.
6. If the checksum is incorrect for a particular block i.e. there is a disk level corruption for that block, it skips that block while reporting the block information to the namenode.
7. In this way, namenode is aware of the disk level corruption on that datanode and takes steps accordingly.

- *Node Manager* This is a yarn daemon which runs on individual nodes and receive updated information on recource containers from their individual datanodes via background daemons. Different resources such as memory, cpu time, network bandwidth etc. are put into one unit called the Resource Container The Node manager in turn ensures fault tolerancy on the data nodes for any map reduce jobs

- *Resource Manager* This is a yarn daemon which manages the allocation of resources to the different jobs apart from comprising a scheduler which just takes care of the scheduling jobs without worrying about any monitoring or status updates

2.2.10 Usage

- copyFromLocal
  Usage:`hadoop fs -copyFromLocal <local-file> <hadoop-fs-dir>`
  Details:Copies file from local file system to HDFS

- copyToLocal
  Usage:`hadoop fs -copyToLocal <hadoop-fs-file> <local-dir>`
  Details:Copies file from local file system to HDFS

- mkdir
  Usage:`hadoop fs -mkdir <hadoop-fs-dir>`
  Details:Create directory in the Hadoop Filesystem

- ls
  Usage:`hadoop fs -ls <hadoop-fs-dir>`
  Details:List files inside a Hadoop Directory

- Note : Similarly other linux commands can also be used

2.3 Hive 0.13.0

2.3.1 Hive [4][5][6][7]

Hive is used as a substitute to SQL for the Hadoop File System which makes it easy for people acquainted with SQL to query data from it instead of having to learn Map-Reduce.

2.3.2 Prerequisites

1. Hadoop 2.2.0

Installation

1. Download
   Download the latest hive release from apache website
   `wget http://apache.mirrors.hoobly.com/hive/stable/apache-hive-0.13.0-bin.tar.gz`
2. Extract the files and Set them up
   `sudo tar -zxvf <hive install>`
   `sudo mv <hive-install> /usr/local/hive`
3. Setup Environment Variables
   Add the following entries to .bashrc
   `export HIVE_PREFIX=/usr/local/hive`
   `export PATH=$PATH:$HIVE_PREFIX/bin`

2.3.3 Usage

1. Launch Hive
   `$hive`
2. CREATE
   ```
   CREATE TABLE books(id INT,name STRING,author STRING)
   ROW FORMAT DELIMITED FIELDS
   TERMINATED BY ',' STORED AS TEXTFILE;
   ```
3. LOAD
   `LOAD DATA (LOCAL) INPATH "books.txt" INTO TABLE books;`

4. SELECT
   `SELECT * FROM books LIMIT 10;`

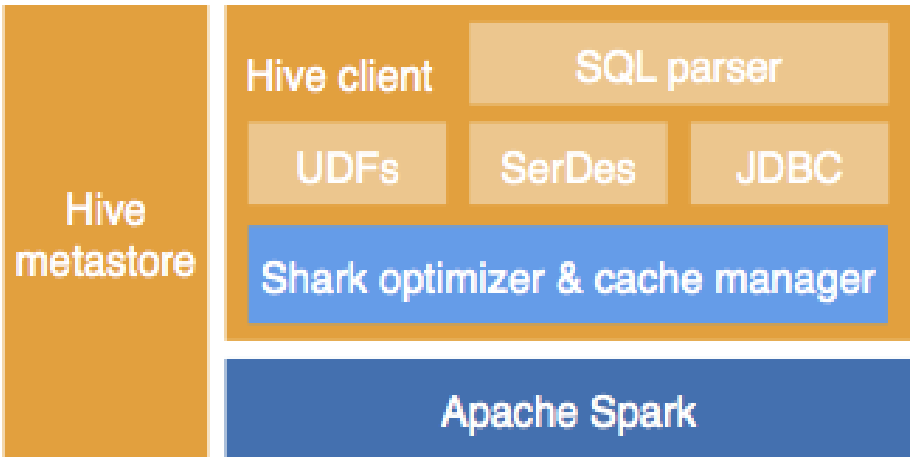


Figure 2.4: Spark and Shark Stack

2.4 Spark 1.0.0

2.4.1 Spark[8]

- Apache Spark is an open-source data analytics cluster computing framework.
- Spark fits into the Hadoop open-source community, building on top of the Hadoop Distributed File System (HDFS).
- Spark is not tied to the two-stage MapReduce paradigm, and promises performance up to 100 times faster than Hadoop MapReduce, for certain applications.
- Follows the concept of a Resilient Distributed Dataset (RDD), which allows to transparently store data on memory and persist it to disc if it's needed
- Provides Machine Learning Library `MLLib` for Data Analytics on the fly

2.4.2 Steps for installation

- Download the setup as:
  `wget http://apache.mirror.quintex.com/spark/spark-1.0.0/spark-1.0.0.tgz`
- Untar the file as:
  `sudo tar -xvzf spark-1.0.0.tgz`
- Move the setup to `/usr/local` as
  `sudo mv /path_to/spark-1.0.0.tgz /usr/local/spark`

- Set the path variables in .bashrc as:
  `export SPARK_HOME=/usr/local/spark`
  `export PATH=$PATH:$SPARK_HOME/bin`
  `export PATH=$PATH:$SPARK_HOME/sbin`

- Add Configuration settings in `/usr/local/spark/conf/spark-env.sh`
  `export SCALA_HOME=$SCALA_HOME`
  `export SPARK_WORKER_MEMORY=16g`

- Logout and then relogin

- Start spark master by:
  `start-master.sh`

- Run spark python shell:
  `pyspark`

2.5 Shark 0.9.1

2.5.1 Shark[9][10]

- Shark is a large-scale data warehouse system for Spark designed to be compatible with Apache Hive.

- It can execute Hive QL queries up to 100 times faster than Hive **without any modification to the existing data or queries**.

- Shark supports Hive's query language, metastore, serialization formats, and user-defined functions, providing seamless integration with existing Hive deployments.

- Shark supports Hive UDFs,SerDe and every major functionality in Hive some of which are not supported in similar query engines such as Impala.

- Spark is written in Scala, and has its own version of the Scala interpreter.

- So its preferable to use scala.

2.5.2 Installing Scala 2.11.1

- Dowload the setup for scala 2.11.1 as from `http://www.scala-lang.org/download/`

- Move it to the /usr/local as:
  `sudo mv /path_to/scala-2.11.1.tgz /usr/local`

- Rename it as:
  `sudo mv /usr/local/scala-2.11.1 /usr/local/scala`

- Set the path variables in the .bashrc file as:
  `export SCALA_HOME=/usr/local/scala`
  `export PATH=$PATH:$SCALA_HOME/bin`
  `export PATH=$PATH:$SCALA_HOME/sbin`
- Logout and relogin.
- Run scala as:
  scala

2.6 Install Shark 0.9.1

- Download Shark precompiled binaries
  `wget https://s3.amazonaws.com/spark-related-packages/shark-0.9.1-bin-hadoop2.tgz`
- Extract the package
  `sudo tar -xzvf shark-0.9.1-bin-hadoop2.tgz`
- Move the extracted package to `/usr/local`
  `sudo mv /path/to/sharkpackage /usr/local/shark`
- Add following Environment Variables
  - Shark Home
    `export SHARK_HOME=/usr/local/shark`
  - Hive Configuration Directory
    `export HIVE_CONF_DIR=$HIVE_HOME/conf`
  - Add variables to path
    `export PATH=$PATH:$HIVE_CONF_DIR export PATH=$PATH:$SHARK_HOME/bin`
    `export PATH=$PATH:$SHARK_HOME/sbin`
- Add Derby Server Jar(`derbyclient.jar,derbytools.jar`) in the $SHARK_HOME/lib directory
- Shark Configuration Settings
  - Rename `shark-env.sh.tempate` to `shark-env.sh`
  - `export SPARK_MEM=16g`
  - `export SHARK_MASTER_MEM=1g`
  - `SPARK_JAVA_OPTS` value will remain as is by default
  - `export HIVE_HOME,HIVE_CONF_DIR,SPARK_HOME`
- Support for Hive 0.12 and above
  Presently Shark officially only supports compatibility upto Hive 0.11
  To extend its compatibility to Hive 0.12 and above the following steps have to be followed

- Extract the $hive-exec-*.jar$ file in the lib_managed folder of shark.
- It includes a version of protocol buffer having a directory structure "$com\backslash google$" would validate that.
- Delete the folder and repackage the jar.

2.7 MongoDB

MongoDB is a open source, document-oriented, NoSQL database system.In the edx platform, the discussion form and some course related material is stored as collections of JSON-like documents in a MongoDB database. We have used local MongoDb database system to store the mongo dump files retrieved from the edX data package.

2.7.1 Installation Steps

1. Import the public key used by the package management system.
   `sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10`

2. Create a list file for MongoDB.
   `echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list`

3. Reload local package database.
   `sudo apt-get update`

4. Install the MongoDB packages.
   `sudo apt-get install mongodb-org`

2.7.2 Basic Usage

After installing the MongoDB packages run the following commands to check whether the MongoDB is properly installed.

- To start MongoDB : sudo start mongodb
- To stop MongoDB : sudo stop mongodb
- To run mongo shell: mongo
  By default, mongo looks for a database server listening on port 27017 on the localhost interface.

2.8 MySQL 5.6

2.8.1 Introduction

MySQL is a relational database management system (RDBMS), and ships with no GUI tools to administer MySQL databases or manage data contained within the databases.The data from the IITBX (eDX of IITB) i.e. the data from the IITB's local server is the mysql dump files.We can process this data by imoprting it into HDFS using sqoop.

2.8.2 Step for installation

sudo apt-get install mysql-server-5.6

2.9 Sqoop

2.9.1 Introduction

Sqoop [20] is a tool designed to transfer data between Hadoop and relational databases. We used Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS).
Sqoop automates most of the process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance.
This section describes how to get started using Sqoop to move data between databases and Hadoop and provides reference information for the operation of the Sqoop command-line tool suite.

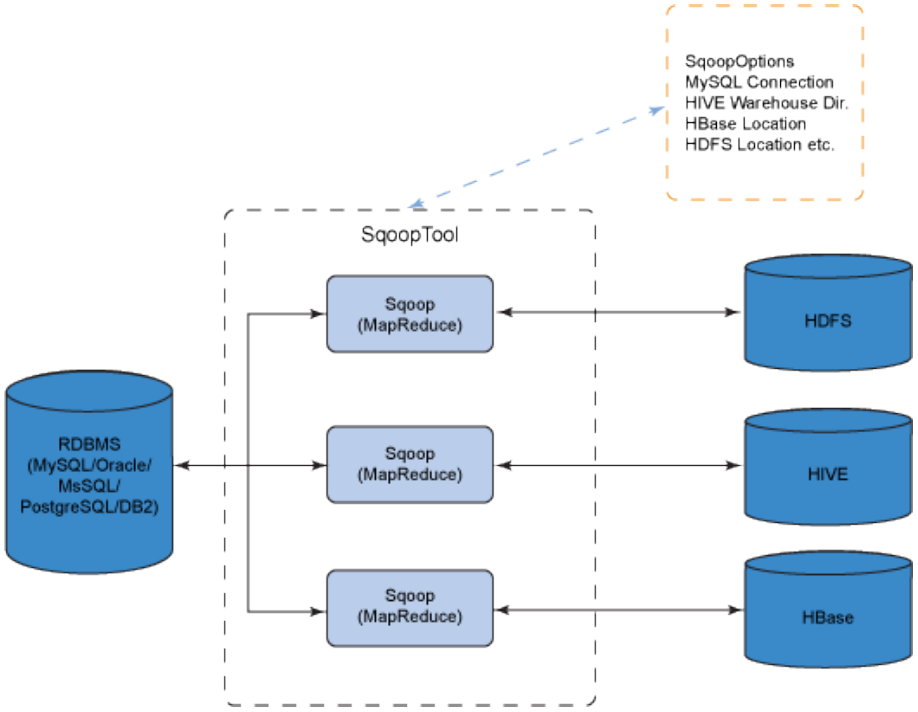


Figure 2.5: Sqoop data import

2.9.2 Stable release and Download

Sqoop is an open source software product of the Apache Software Foundation.Sqoop source code is held in the Apache GIT repository. You might clone the repository using following command: git clone https://git-wip-us.apache.org/repos/asf/sqoop.git
Latest stable release is 1.4.4. One can download it using following command:
wget http://mirror.symnds.com/software/Apache/sqoop/1.4.4/sqoop-1.4.4.bin_hadoop-1.0.0.tar.gz

This command will download the latest stable release of sqoop 1.4.4 and store the tar file in the working directory.

2.9.3 Prerequisites

Before you can use Sqoop, a release of Hadoop must be installed and configured. Sqoop is currently supporting 4 major Hadoop releases - 0.20, 0.23, 1.0 and 2.0. We have installed Hadoop 2.2.0 and it is compatible with sqoop 1.4.4.We are using a Linux environment Ubuntu 12.04 to install and run sqoop. The basic familiarity with the purpose and operation of Hadoop is required to use this product.

2.9.4 Installation

To install the sqoop 1.4.4 we followed the given sequence of steps.

1. Unzip the tar file:
   sudo tar -zxvf sqoop-1.4.4.bin__hadoop1.0.0.tar.gz
2. Move sqoop-1.4.4.bin__hadoop1.0.0 to sqoop:
   sudo mv sqoop-1.4.4.bin__hadoop1.0.0 /usr/local/sqoop
3. Set the SQOOP_HOME path in bashrc file:
   export SQOOP_HOME=/usr/lib/sqoop
   export PATH=$PATH$:SQOOP_HOME/bin
4. Test your installation by typing:
   sqoop help

2.10 Derby Server

Hive in embedded mode has a limitation of one active user at a time. We want to run Derby as a Network Server [21], this way multiple users can access it simultaneously from different systems.

2.10.1 Download and Install [11]

The Derby server must be compatible with Hive. We have already run Hive in embedded mode, the first line of derby.log contains the version. We installed the Hadoop,Hive and Sqoop in /usr/local directory. Locate the derby.log file in hive directory and read its first line. The first line of our derby.log file contains:

Booting Derby version The Apache Software Foundation - Apache Derby - 10.10.1.1

So we downloaded the derby server 10.10.1.1 and extract the contents of tar file in /usr/local/derby directory.

2.10.2 Configuration

- Create a directory data in /usr/local/derby/data.
  mkdir /usr/local/derby/data

- Set Environment variables in .bashrc file of hduser.
  sudo nano .bashrc
  add the following lines at the end of file:
  export DERBY_HOME=/usr/local/derby
  export DERBY_INSTALL=/usr/local/derby
  export PATH=$PATH:$DERBY_HOME/bin
  export HADOOP=/usr/local/hadoop/bin/hadoop

- Configure Hive to Use Network Derby
  In hive/conf/hive-site.xml
  change the value of following property
  javax.jdo.option.ConnectionURL
  jdbc:derby://localhost:1527/metastore db;create=true

- Copy the derbyclient.jar and derbytools.jar from the derby/lib/ to hive/lib/ and hadoop/lib/.

- Run startNetworkServer to start derby server.

- Always start derby server before start instance of hive.

- Now we can run multiple Hive instances working on the same data simultaneously and remotely.

2.10.3 Error in starting Derby Server

If startNetworkServer command doesnot start the derby server and you get an error like:
access denied (”java.net.SocketPermission” ”localhost:1527” ”listen,resolve”)
To resolve this error, take the following steps:

- Locate the java.policy file. In our case it is
  /usr/lib/jvm/java-7-openjdk-amd64/jre/lib/security/java.policy

- Give permission
  grant { permission java.net.SocketPermission "localhost:1527", "listen"; };

2.11 Django 1.6.5

2.11.1 Requirements

- django package
- redis [22]
- django-rq [23]
- django-pandas

2.11.2 Steps for Installation

Django setup

- sudo apt-get install python mysql-client mysql-server python-mysqldb
- `pip install django==1.6.5`
- check whether django is installed properly
  ```
  python
  import django
  print(Django.get_version())
  ```
- create a directory where you want to store your projects.Let the name be django
  `mkdir django`
- `django-admin.py startproject projectname`
- cd projectname
  You will find a folder named projectname which will contain files named views.py, urls.py, settings.py, wsgi.py
- to run django type `python manage.py runserver` inside your upper projectname folder
- In your browser type, localhost:8000, you will be able to see the default django index page

2.11.3 Redis

This is required to queue processes

- wget http://download.redis.io/redis-stable.tar.gz
- tar xvzf redis-stable.tar.gz
- cd redis-stable
- make. It will take some time to execute
- To check whether it is installed properly execute
  `make test`
- sudo cp redis-server /usr/local/bin/
- sudo cp redis-cli /usr/local/bin/
- Start redis-server for execution by executing `redis-server`

2.11.4 django-rq

- pip install rq
- pip install django-rq
- Add django_rq to INSTALLED_APPS in settings.py file
  ```
  INSTALLED_APPS =
  # other apps
  django_rq,
  )
  ```
- Configure your queues in django's settings.py
  ```
  RQ_QUEUES = { "default":  {
  "HOST":  "localhost",
  "PORT":  6379,
  "DB":  0,
  },
  "high":  {
  "URL":  os.getenv("REDISTOGO_URL", "redis://localhost:6379"), # If you‘re
  on Heroku
  "DB":  0,
  },
  "low":  {
  "HOST":  "localhost",
  "PORT":  6379,
  "DB":  0,
  }
  }
  ```

- Include django_rq.urls in your urls.py `urlpatterns += patterns("", (r"^django-rq/", include("django_rq.urls")), )`
- Inside that projectname directory, type `python manage.py rqworker`

2.11.5 Django-pandas

- `pip install numpy`
- `pip install pandas`
- `pip install https://github.com/chrisdev/django-pandas/tarball/master`
- `To use django-pandas in your Django project, modify the INSTALLED_APPS in your settings.py file to include django_pandas`

2.12 Python packages

This section describes the installation of different python packages required for various tasks such as query hive from client side, run bash commands, run Nosql(mongo) queries etc.

1. sasl : To install the pyhs2 package [24], sasl package has to be installed. It covers the authentication and security layers. To install it run :

   sudo pip install sasl

   If you get an error like this while installing sasl package :

   sasl/saslwrapper.cpp:21:23: fatal error: sasl/sasl.h: No such file or directory

   compilation terminated.

   error: command 'gcc' failed with exit status 1

   Then to rectify this error we installed the libsasl2-dev library using command :

   sudo apt-get install libsasl2-dev

   After installing this library, again try to install the sasl.

2. pyhs2 : This package enables us to run hive queries from python. Using pyhs2, we made connection with hiveserver2 and submit hive queries from client machine. To install pyhs2 run:

sudo pip install pyhs2

3. pymongo : Using pymongo, we wrote mongo queries in python, made connection to mongoDB database system, run query and stored the result in a cursor. To install it run :

   sudo pip install pymongo

4. pygeoip : Using pygeoip, we found the location of the registered users from their IP addresses.Along with pygeoip we have used two databases GeoLiteCity.dat and GeoLiteCountry.dat. The locations found are used to visualize the number of users registered from a particular country on the world map. To install it run:

   sudo pip install pygeoip

2.13 Multinode cluster Configuration

This section describes how to install Hadoop ecosystem in a multinode cluster and configure it properly. We used two nodes one as master and the other as slave in our cluster.

2.13.1 Hadoop 2.2.0 in multinode [12]

To install Hadoop 2.2.0 on multinodes, first setup Hadoop single node cluster on the master and slaves as described earlier in this report. Then follow the given steps to configure it.

1. Network Settings : To run a multinode cluster ensure that the master and all the slaves are on a singlenetwork. Identify the ip address of each system. Now make entries in the /etc/hosts file. In our system it is as follows

   ```
   10.129.46.116 master
   10.129.46.253 slave01
   ```

2. SSH login for all slaves : Add the public key of master to all slaves using the command:

   ```
   ssh-copy-id -i $HOME/.ssh/id_dsa.pub hduser@slave01
   ```

   Now ssh to the master and slaves for ensuring that passwordless ssh has been setup properly.

```
ssh master
ssh slave01
```

3. Configuration Files :Add the following lines bewteen <configuration> and <=configuration> tags to the files in $HADOOP_HOME/etc/hadoop folder for both master and slave. Configure the following files

   - **core-site.xml**
     Paste the following between <configuration>
     ```
     <property>
     <name>fs.defaultFS</name>
     <value>hdfs://localhost:9000</value>
     </property>
     ```

   - **yarn-site.xml**
     Paste the following between <configuration>
     ```
     <property>
     <name>yarn.nodemanager.aux-services</name>
     <value>mapreduce_shuffle</value>
     </property>
     <property>
     <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
     <value>org.apache.hadoop.mapred.ShuffleHandler</value>
     </property>
     ```

   - **mapred-site.xml**
     Paste the following between <configuration>
     ```
     <property>
     <name>mapreduce.framework.name</name>
     <value>yarn</value>
     </property>
     ```

   - **hdfs-site.xml**
     Paste the following between <configuration>
     ```
     <property>
     <name>dfs.replication</name>
     <value>3</value>
     </property>
     <property>
     <name>dfs.namenode.name.dir</name>
     <value>file:/home/hduser/mydata/hdfs/namenode</value>
     </property>
     <property>
     <name>dfs.datanode.data.dir</name>
     <value>file:/home/hduser/mydata/hdfs/datanode</value>
     </property>
     ```

Now add the following names of all slaves to the $HADOOP_HOME/etc/slaves file.

```
nano $HADOOP
HOME/etc/slaves
slave01
```

4. Starting Daemons : Format the namenode if you want to erase data on the Hadoop File System using the command

```
hdfs namenode -format
```

Run the following two scripts on the master node to start the hadoop and yarn daemons:

```
start-dfs.sh
start-yarn.sh
```

To test if the Daemons have started properly or not run the jps on the master and slave:
Master:

```
hduser@master:/usr/local/hadoop$: jps
9412 SecondaryNameNode
9784 NameNode
19056 Jps
10173 ResourceManager
```

Slave:

```
hduser@slave01:/usr/local/hadoop$: jps
18762 Datanode
18865 Nodemanager
20223 Jps
```

If the above processes do not start on either the master or slave, check the log file in $HADOOP_HOME/logs directory corresponding to the daemon that caused an error.

2.13.2 Spark 1.0.0 in multinode

This section describes the installation of spark on a multinode cluster.

- Install spark on all the masters and slaves as mentioned above in this report.

- Create a file called conf/slaves in your Spark directory, which should contain the hostnames of all the machines where you would like to start Spark workers, one per line.
- The master machine must be able to access each of the slave machines via password-less ssh (using a private key).
- Once youve set up this file, you can launch or stop your cluster with the following shell scripts, based on Hadoops deploy scripts, and available in SPARK_HOME/bin:
  - sbin/start-master.sh - Starts a master instance on the machine the script is executed on.
  - sbin/start-slaves.sh - Starts a slave instance on each machine specified in the conf/slaves file.
  - sbin/start-all.sh - Starts both a master and a number of slaves as described above.
  - sbin/stop-master.sh - Stops the master that was started via the bin/start-master.sh script.
  - sbin/stop-slaves.sh - Stops all slave instances on the machines specified in the conf/slaves file.
  - sbin/stop-all.sh - Stops both the master and the slaves as described above.

  Note that these scripts must be executed on the machine you want to run the Spark master on, not your local machine.

Chapter 3

Design details and Implementation

3.1 Data Inputs

This section describes the two types of input data to the data analytics system. One is the standard EdX data package which EdX transfers to university partners in regularly generated data packages. These Data packages are delivered to a single contact at each university, referred to as the "data czar". The data packages contain encrypted data in compressed archive files.

EdX stores the data packages in a secure bucket on the Amazon Web Services (AWS) Simple Storage Service (Amazon S3). Only the data czar is given access credentials (a user name and password) to the AWS S3 account.
To retrieve the each data package, the data czar must complete these steps:

1. Create the encryption keys.
2. Receive an email message from edX.
3. Access the AWS S3 account.
4. Download the data package.
5. Decrypt the data package.

Details for each of these tasks follow for the data czar at partner institution. For more information about the responsibilities of a data czar, see the official edX Data Documentation.

EdX provides two types of research data to partners who are running classes on edx.org and edge.edx.org:

- Log (event tracking) data
- Database data, including student information

To access log and database data packages [25], you download files from edX, then extract the information from those files for analysis.

3.1.1 Event tracking data

The edX platform gathers tracking information on almost every interaction of every student.Event tracking data for your institution is collected in a file: **Format** : Date-Institution-tracking.tar **Example**: 2014-03-23-IITBombayX-tracking.tar When we extract the contents of this TAR file, sub-directories are created for each edX server that the course is running on.
For example, we get the following sub-directories: **sub-directory Format**: prod-edxapp-number
prod-edxapp-001
prod-edxapp-002
prod-edxapp-003
Each of these sub-directories contains a file of tracking data for each day. The TAR file is cumulative; that is, it contains files for all previous days your course was running on that server. The filename format for event tracking data files is: Date_Institution.log.gpg.

3.1.2 Database data

Database data files are collected [26] in a ZIP file named: Institution-Date.zip
Example:iitbombayx-2014-03-24.zip
These data contains the Student information, progress data, courseware data, discussion forum data and wiki related data. The following table summarizes the different useful tables in EdX platform.

S.No.	Table Name	Description
1.	auth_user	Information about users authorized to access the course.
2.	auth_userprofile	Information about student demographics
3.	certificates_generatedcertificate	Certificate status for graded students after course completion.
4.	courseware_studentmodule	Information about courseware state for each student. There is a separate row for each (UNIT) the course. For courses that do not have any records in this table no file is produced.
5.	student_courseenrollment	Information about students enrolled in the course, enrollment status, and type of enrollment.
6.	user_id_map	A mapping of user IDs and obfuscated IDs used in surveys.
7.	wiki_article	Course wiki data.
8.	wiki_articlerevision	Changes in wiki article.
9.	user_api_usercoursetag	store metadata about a specific student's involvement in a specific course.

Table 3.1: Types of Database tables in edX

The database data can be divided into three parts :

1. Student Info and Progress Data
2. Discussion Forums Data
3. Wiki Data

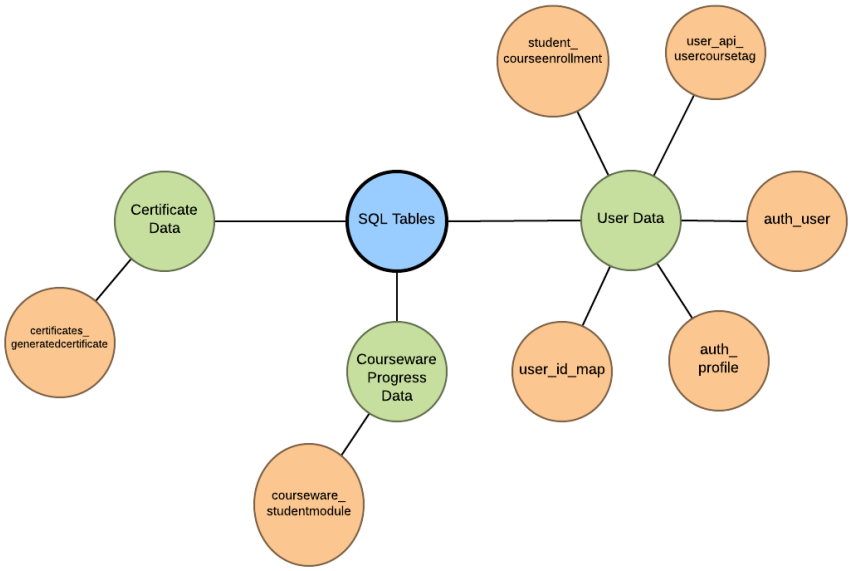


Figure 3.1: Mind Map of SQL Tables

EdX conventions:

- edX uses MySQL 5.1 relational database system with InnoDb storage engine.
- All strings are stored as UTF-8
- All datetimes are stored as UTC (Coordinated Universal Time).

MySQL Terminology The summary information provided about the SQL table columns uses the following MySQL schema terminology.

Value	Description
int	4 byte integer.
smallint	2 byte integer, sometimes used for enumerated values.
tinyint	1 byte integer, usually used to indicate a Boolean with 0 = False and 1 = True.
varchar	String, typically short and indexable. The length is the number of chars, not bytes, to support multi-byte character sets.
longtext	A long block of text, usually not indexed.
date	Date
datetime	Datetime in UTC, precision in seconds.
Null value : YES	NULL values are allowed.
Null value : NO	NULL values are not allowed.
Key value : PRI	Primary key for the table
Key value : UNI	Unique key for the table
Key value : MUL	Indexed for fast lookup, but the same value can appear multiple times. A unique index that allows NULL can also show up as MUL.

Table 3.2: MySql Terminology used

Student Info and Progress Data: The following sections detail how edX stores stateful data for students internally, and is useful for developers and researchers who are examining database exports. This data includes:

- **User Data:** The following tables comes under the User data

1. auth_user Table : This table contains information about the authorized users.
   Filename Format :Institution-Course-Date-auth_user-Server-analytics.sql
   Ex:IITBombayX-CS001-Fall_2013-auth_user-prod-edge-analytics.sql
   The auth_user table has the following columns: 3.2

Column	Type	Null	Key	Comment
id	int(11)	NO	PRI	
username	varchar(30)	NO	UNI	
first_name	varchar(30)	NO		# Never used
last_name	varchar(30)	NO		# Never used
email	varchar(75)	NO	UNI	
password	varchar(128)	NO		
is_staff	tinyint(1)	NO		
is_active	tinyint(1)	NO		
is_superuser	tinyint(1)	NO		
last_login	datetime	NO		
date_joined	datetime	NO		

Figure 3.2: auth_user table structure without obsolete columns

   Except these columns, there are other columns which are obsolete and not used such as status, email_key,avatar_typ, country,show_country, date_of_birth, interesting_tags, ignored_tags, email_tag_filter_strategy, display_tag_filter_strategy, consecutive_days_visit_count.

2. auth_userprofile Table :This table stores the full username and other information.
   Filename Format :Institution-Course-Date-auth_userprofile-Server-analytics.sql
   Ex: IITBombayX-CS001-Fall_2013-auth_userprofile-prod-edge-analytics.sql
   The auth_userprofile table has the following columns: 3.3

Column	Type	Null	Key	Comment
id	int(11)	NO	PRI	
user_id	int(11)	NO	UNI	
name	varchar(255)	NO	MUL	
language	varchar(255)	NO	MUL	# Obsolete
location	varchar(255)	NO	MUL	# Obsolete
meta	longtext	NO		
courseware	varchar(255)	NO		# Obsolete
gender	varchar(6)	YES	MUL	# Only users signed up after prototype
mailing_address	longtext	YES		# Only users signed up after prototype
year_of_birth	int(11)	YES	MUL	# Only users signed up after prototype
level_of_education	varchar(6)	YES	MUL	# Only users signed up after prototype
goals	longtext	YES		# Only users signed up after prototype
allow_certificate	tinyint(1)	NO		

Figure 3.3: auth_userprofile Table

The gender column contains f for female, m for male, o for other, blank if user not specify and NULL if user signed up before this information was collected. The level_of_education contains p for Doctorate, m for Master's or professional degree,b for Bachelor's degree,a forAssociate's degree, hs for Secondary/high school, jhs for Junior secondary/junior high/middle school,el for Elementary/primary school, none,other and blank if user did not specify level of education.

3. student_courseenrollment Table :A row in this table represents a student's enrollment for a particular course run.A row is created for every student who starts the enrollment process, even if they never complete registration. Filename Format : Institution-Course-Date-student_courseenrollment-Server-analytics.sql Ex: IITBombayX-CS001-Fall_2013-student_courseenrollment-prod-edge-analytics.sql

The student_courseenrollment table has the following columns: 3.4

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
user_id	int(11)	NO	MUL	NULL	
course_id	varchar(255)	NO	MUL	NULL	
created	datetime	YES	MUL	NULL	
is_active	tinyint(1)	NO		NULL	
mode	varchar(100)	NO		NULL	

Figure 3.4: student_courseenrollment Table

4. user_api_usercoursetag Table : This table uses key-value pairs to store metadata about a specific student's involvement in a specific course. For example, for a course that assigns students to groups randomly for A/B testing, a row in this table identifies the student's assignment to a partition and group.The user_api_usercoursetag table has the following columns: 3.5

Column	Type	Null	Key
user_id	int(11)	NO	PRI
course_id	varchar(255)	NO	
key	varchar(255)	NO	
value	textfield	NO	

Figure 3.5: user_api_usercoursetag Table

5. user_id_map Table :A row in this table maps a student's real user ID to an anonymous ID generated to obfuscate the student's identity.
Filename Format : Institution-Course-Date-user_id_map-Server-analytics.sql
Ex: IITBombayX-CS001-Fall_2013-user_id_map-prod-edge-analytics.sql
The user_id_map table has the following columns: 3.6

Column	Type	Null	Key
hashid	int(11)	NO	PRI
id	int(11)	NO	
username	varchar(30)	NO	

Figure 3.6: user_id_map Table

- **Courseware Progress Data:** Any piece of content in the courseware can store state and score in the courseware_studentmodule table. Grades and the user Progress page are generated by doing a walk of the course contents, searching for graded items, looking up a student's entries for those items in courseware_studentmodule via (course_id, student_id, module_id), and then applying the grade weighting found in the course policy and grading policy files.The courseware_studentmodule table holds all courseware state for a given user.

  Filename Format :Institution-Course-Date-courseware_studentmodule-Server-analytics.sql
  Ex :IITBombayX-CS001-Fall_2013-courseware_studentmodule-prod-edge-analytics.sql

  The courseware_studentmodule table has the following columns: 3.7

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
module_type	varchar(32)	NO	MUL	problem	
module_id	varchar(255)	NO	MUL	NULL	
student_id	int(11)	NO	MUL	NULL	
state	longtext	YES		NULL	
grade	double	YES	MUL	NULL	
created	datetime	NO	MUL	NULL	
modified	datetime	NO	MUL	NULL	
max_grade	double	YES		NULL	
done	varchar(8)	NO	MUL	NULL	
course_id	varchar(255)	NO	MUL	NULL	

Figure 3.7: courseware_studentmodule Table

- **Certificate Data:** The certificates_generatedcertificate table tracks the state of certificates and final grades for a course. The table is populated when a script is run to grade all of the students who are enrolled in the course at that time and issue certificates. The certificate process can be rerun and this table is updated appropriately.

  Filename Format :Institution-Course-Date-certificates_generatedcertificate-Server-analytics.sql **Ex** :IITBombayX-CS001-Fall_2013-certificates_generatedcertificate-prod-edge-analytics.sql The certificates_generatedcertificate table has the following columns: 3.8

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
module_type	varchar(32)	NO	MUL	problem	
module_id	varchar(255)	NO	MUL	NULL	
student_id	int(11)	NO	MUL	NULL	
state	longtext	YES		NULL	
grade	double	YES	MUL	NULL	
created	datetime	NO	MUL	NULL	
modified	datetime	NO	MUL	NULL	
max_grade	double	YES		NULL	
done	varchar(8)	NO	MUL	NULL	
course_id	varchar(255)	NO	MUL	NULL	

Figure 3.8: certificates_generatedcertificate Table

- **Discussion Forums Data :**EdX discussion data is stored as collections of JSON documents in a MongoDB database. MongoDB is a document-oriented, NoSQL database system. Documentation can be found at the mongodb web site.
  In the data package, discussion data is delivered in a .mongo file, identified by organization and course, in this format: edX-organization-course-source.mongo. The primary collection that holds all of the discussion posts written by users is "contents".
  - CommentThread : It represents a post that opens a new thread, often a student question of some sort.
  - Comment : It represents response made directly to the conversation started by a CommentThread is a Comment. Any further contributions made to a specific response are also in Comment objects.

- **Wiki Data :**In the data package, wiki data is delivered in two SQL files:
  - wiki_article Table : The wiki_article table is a container for each article that is added to the wiki.
    Filename Format : edX-organization-course-wiki_article-source-analytics.sql
    Ex : IITBombayX-ME209x-2T2014-wiki_article-prod-analytics.sql
    The wiki_article Table has the following columns: 3.9

Field	Type	Null	Key
id	int(11)	NO	PRI
current_revision_id	int(11)	NO	UNI
created	datetime	NO	
modified	datetime	NO	
owner_id	int(11)	YES	MUL
group_id	int(11)	YES	MUL
group_read	tinyint(1)	NO	
group_write	tinyint(1)	NO	
other_read	tinyint(1)	NO	
other_write	tinyint(1)	NO	

Figure 3.9: wiki_article Table

- wiki_articlerevision Table :The wiki_articlerevision table stores data about the articles, including data about changes and deletions.
  Filename Format :edX-organization-course-wiki_articlerevision-source-analytics.sql
  Ex :IITBombayX-ME209x-2T2014-wiki_articlerevision-prod-analytics.sql
  The wiki_articlerevision Table has the following columns: 3.10

Field	Type	Null	Key
id	int(11)	NO	PRI
revision_number	int(11)	NO	
user_message	longtext	NO	
automatic_log	longtext	NO	
ip_address	char(15)	YES	
user_id	int(11)	YES	MUL
modified	datetime	NO	
created	datetime	NO	
previous_revision_id	int(11)	YES	MUL
deleted	tinyint(1)	NO	
locked	tinyint(1)	NO	
article_id	int(11)	NO	MUL
content	longtext	NO	
title	varchar(512)	NO	

Figure 3.10: wiki_articlerevision Table

3.1.3 Input Data from Local EdX server

The other type of input data to the Analysis system is from the local EdX platform deployed in the server of IIT Bombay. EdX platform uses the Mysql 5.1 to store the database data and MongoDB to store the discussion forum data and course related data. Our Data analysis system take input data from the backup files. Three types of dump files are there:

MySql Dump Files : These are the mysql dump files in the .sql file format. In these there are sql files for edxapp, ora and xqueue. We used the edxapp sql file for the analysis purpose. It contains all the important database tables which we have described in previous section.

Log Files : These are the tracking log files which contains the information about

the event tracking data of different event types.

Mongo Dump Files : These files contain information about forum data and course data.Two database found in mongo dump are :

- cs_comments_service_development : This database consists of BSON files of contents collection which stores the forum data.
- edxapp : This database consists of BSON files of modulestore collection which stores the course data

3.2 Querying

After the data has been stored onto the HDFS, next step is to extract useful information from that data by querying from the HDFS. Queries are in Hive Query Language(HQL) which is very much similar to SQL. All these hive queries are initially executed through hive shell and later on through the python script.

3.2.1 Queries

Below mentioned are the various queries that a user may want to retrieve information of:-

1. List the staff members who are active.

```
select distinct id ,username
from auth_user
where (id is not NULL and is_staff=1 and is_active=1)
order by username;
```

2. How many users are female?

```
select id,name,gender
from auth_userprofile
where (id is not null and gender=='f')
limit 10;
```

3. How many students pursuing their bachelors have registered?

```
select id,name
from auth_userprofile
where (id is not NULL and level_of_education='b')
order by id,name limit 10;
```

4. How many have registered for a particular course are born after 1980?

```
select id,name,gender
from auth_userprofile
where year_of_birth > 1980 limit 3;
```

5. How many courses a particular user is registered for?

```
select user_id,count(course_id)
from student_courseenrollment
where (id is not null)
GROUP BY user_id;
```

6. How many students have a mode as honored?

```
select mode,count(mode)
from student_courseenrollment
where (mode='honor') group by mode;
```

7. How many students have enrolled in a particular course?

```
select course_id,count(user_id)
from student_courseenrollment
where (id is not null) group by course_id;
```

8. What was the last time when a student has registered for a course?

```
select distinct user_id,course_id,created
from student_courseenrollment
where id is not NULL
order by user_id DESC,created DESC;
```

9. In which subject a student has registered more than once?

```
select user_id
from student_courseenrollment
group by user_id having count(course_id) > 1;
```

10. How many students have their enrolled un-verified?

```
select count(distinct user_id)
from student_courseenrollment
where (mode not like 'verify' and id is not null);
```

11. To find the actual id of a user?

```
select id,username
from user_id_map
where id is not null;
```

12. What is the highest grade obtained by any student in a particular module ?

```
select student_id,max(grade)
from courseware_studentmodule
where id is not null group by student_id;
```

13. How many chapters have been solved by the students in a particular course ?

```
select student_id,course_id,count(module_type)
from courseware_studentmodule
where id is not null and module_type='chapter'
group by student_id,course_id;
```

14. Total amount of time student spends on a particular course

```
select distinct(student_id),modified
from courseware_studentmodule
where id is not null order by modified desc;
```

15. Finding response times for students by course

    - Get usernames by course

      ```
      select distinct username
      from pi_log_table
      where course_id="Summer_Intern_IIT_Mumbai/SI003/2014_SI_May";
      ```

    - To get the cleaned logs where the event type is "problem_get" and "problem_check".

      ```
      select username,event_type,pid,course_id,time
      from (select username,event_type,time,
       split(split(split(path,"\\;")[5],"_")[1],"/")[0] as pid,course_id
       from pi_log_table
       where course_id="Summer_Intern_IIT_Mumbai/SI003/2014_SI_May" and
       (event_type="problem_check" and
       submission is not null and attempts=1) or event_type="problem_get")
      order by username,time));
      ```

16. Finding event catgeorization by course.

```
select username,count(*) as event_num
from pi_log_table
where event_type='problem_check'
and course_id="Summer_Intern_IIT_Mumbai/SI003/2014_SI_May"
group by username";
```

17. Finding the login times for users each session.

```
select username,session,min(time) as time
from log_table where session is not null and username!=""
and username is not null
group by username,session
order by time;
```

18. Finding the logout times for users each session.

```
select username,session,max(time) as time
from log_table
where session is not null  and username!=""
and username is not null
group by username,session
order by time;
```

19. Number of male and female students registered for a course.

```
select a.c_id,auth_userprofile.gender,count(distinct a.username)
from (select username,user_id,course_id
 as c_id,split(split(split(path,'\\;')[5],'_')[1],'/')[0]
 as q_id,split(split(submission,'input_type\":\"')[1],'\",\"')[0]
   as q_type,split(split(submission,'question\":\"')[1],'\",\"')[0]
 as q,split(split(submission,'response_type\":\"')[1],'\",\"')[0]
 as a_type,
 case when split(split(submission,'response_type\":\"')
 [1],'\",\"')[0]=\"multiplechoiceresponse\"
 then split(split(split(split(submission,'\\\\\\{\"')[2],'\\\\\\}')[0],
 'answer\":\"')[1],'\",\"')[0]
 when split(split(submission,'response_type\":\"')
 [1],'\",\"')[0]=\"choiceresponse\"
 then split(split(split(split(submission,'\\\\\\{\"')[2],'\\\\\\}')[0],
 'answer\":\\\\\\[')[1],'\\\\\\],')[0] end
 as a,
 split(split(submission,'correct\":')[1],'}')[0]
 as check from pi_log_table
 where submission is not null
 and course_id="Summer_Intern_IIT_Mumbai/SI003/2014_SI_May"
 and event_type=\"problem_check\")"
join auth_userprofile on a.user_id=auth_userprofile.user_id
where auth_userprofile.gender!="NULL" and auth_userprofile.gender!=""
group by auth_userprofile.gender,a.c_id';
```

20. Finding the years when different courses are offered.

```
select distinct year(time) as year
from log_table
where year(time) is not null;
```

21. Finding the courses offered in a particular year.

```
select course_id,min(time) from log_table
where course_id is not null
and course_id!='' and year(time)="2013"
group by course_id";
```

22. Finding the users registered in a course.

```
select distinct username
from pi_log_table
where course_id="Summer_Intern_IIT_Mumbai/SI003/2014_SI_May"
where username is not null and username!="";
```

23. Finding the age distribution of the users registered for a course.

```
select users.user_id,
(year(from_unixtime(unix_timestamp())) - auth_userprofile.year_of_birth)
from (select distinct user_id
     from pi_log_table
     where course_id="Summer_Intern_IIT_Mumbai/SI003/2014_SI_May"
     and username is not null
     and username!="")
join auth_userprofile
on auth_userprofile.user_id=users.user_id';
```

24. Finding the correct incorrect response for questions in a course.

```
select username,q_id,check
from (select username,user_id,course_id as c_id,
  split(split(split(path,'\\;')[5],'_')[1],'/')[0]
  as q_id,
  split(split(submission,'input_type\":\"')[1],'\",\"')[0]
  as q_type,
  split(split(submission,'question\":\"')[1],'\",\"')[0]
  as q,
  split(split(submission,'response_type\":\"')[1],'\",\"')[0]
  as a_type,
  case when split(split(submission,'response_type\":\"')[1],'\",\"')
  [0]=\"multiplechoiceresponse\"
  then split(split(split(split(submission,'\\\\\\{\"')[2],'\\\\\\}')
  [0],'answer\":\"')
  [1],'\",\"')[0]
  when split(split(submission,'response_type\":\"')
  [1],'\",\"')[0]=\"choiceresponse\"
  then split(split(split(split(submission,'\\\\\\{\"')[2],'\\\\\\}')
  [0],'answer
  \":\\\\\\[')[1],'\\\\\\],')[0] end
  as a,
  split(split(submission,'correct\":')[1],'}')[0]
```

```
  as check from pi_log_table
  where submission is not null and event_type=\"problem_check\")
where q is not null and c_id="Summer_Intern_IIT_Mumbai/SI003/2014_SI_May"
order by q_id'
```

25. Finding the marks obtained by a student in particular course.

```
select username,q_id,check
from(select username,user_id,course_id as c_id,
 split(split(split(path,'\\;')[5],'_')[1],'/')[0]
 as q_id,
 split(split(submission,'input_type\":\"')[1],'\",\"')[0]
 as q_type,
 split(split(submission,'question\":\"')[1],'\",\"')[0]
 as q,
 split(split(submission,'response_type\":\"')[1],'\",\"')[0]
 as a_type,
 case when split(split(submission,'response_type\":\"')[1],'\",\"')
 [0]=\"multiplechoiceresponse\"
 then split(split(split(split(submission,'\\\\\\{\"')[2],'\\\\\\}')
 [0],'answer\":\"')
 [1],'\",\"')[0]
 when split(split(submission,'response_type\":\"')[1],'\",\"')
 [0]=\"choiceresponse\"
 then split(split(split(split(submission,'\\\\\\{\"')[2],'\\\\\\}')[0],
 'answer\":\\\\\\[')[1],'\\\\\\],')[0] end
 as a,
 split(split(submission,'correct\":')[1],'}')[0]
 as check from pi_log_table
 where submission is not null and event_type=\"problem_check\")
where q is not null and c_id="Summer_Intern_IIT_Mumbai/SI003/2014_SI_May"
order by q_id;
```

26. Finding the age distribution of all the registered users.

```
select auth_userprofile.user_id,auth_user.username,
(year(from_unixtime(unix_timestamp())) - auth_userprofile.year_of_birth)
as age from auth_userprofile
join auth_user on auth_user.id=auth_userprofile.user_id";
```

27. Finding the degree distribution of all the registered users.

```
select level_of_education,count(id)
as edcount from auth_userprofile
where level_of_education is not null
and level_of_education not in ('','null','none')
group by level_of_education";
```

28. Finding the number of videos watched by a user in a course.

```
select username,count(*) as videos_watched
from vi_log_table
where event_type='play_video'
and course_id='Summer_Intern_IIT_Mumbai/SI003/2014_SI_May'
group by username;
```

29. Finding the number of users who just jumped through the videos in a course.

```
select id,count(distinct(username)) as jumps
from vi_log_table
where event_type='seek_video'
and course_id='Summer_Intern_IIT_Mumbai/SI003/2014_SI_May'
group by id;
```

30. Finding the number of users who changed(increased) the speed of the videos in a course.

```
select id,count(distinct(username)) as count
from vi_log_table
where event_type='speed_change_video'
and (old_speed < new_speed)
and course_id='Summer_Intern_IIT_Mumbai/SI003/2014_SI_May'
group by id;
```

31. Finding the number of users who used transcript to understand the videos in a course.

```
select id,count(distinct(username)) as count
from vi_log_table
where event_type='speed_change_video'
and (old_speed < new_speed)
and course_id='Summer_Intern_IIT_Mumbai/SI003/2014_SI_May'
group by id;
```

32. Finding the number of users who used transcript to understand the videos in a course.

```
select id,count(distinct(username)) as jumps
from vi_log_table
where event_type='show_transcript'
and course_id='Summer_Intern_IIT_Mumbai/SI003/2014_SI_May'
group by id;
```

33. Finding the number of views of a video in a course.

```
select id,count(distinct(username)) as views
from vi_log_table
where event_type='play_video'
and course_id='Summer_Intern_IIT_Mumbai/SI003/2014_SI_May'
group by id;
```

34. Finding the time for which a video was watched in a course.

```
select id,distinct(user_id),max(currentTime)
from vi_log_table
where event_type in ('play_video','pause_video')
and course_id='Summer_Intern_IIT_Mumbai/SI003/2014_SI_May'
group by id;
```

3.2.2 Hive with Python

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.We are using hive query language to query the data stored in HDFS.But while automating the whole process of transfering the data from the MySQl database to Hive and quering the data in Hive, we have faced several problems.

- How to run hive queries from other hosts?
- How to run hive queries using python?
- Tables are not found in hive?
- How to automate data transfer from MySQL to Hive using Sqoop?

After we are done with executing queries directly from hive shell, all of these hive queries are executed by calling them from python script so that user have an interface where he can see the output without the need of accessing and typing them on shell. For this we used hiveserver2.

HiveServer2 [27]

HiveServer2 (HS2) is a server interface that enables remote clients to execute queries against Hive and retrieve the results. The current implementation, based on Thrift RPC, is an improved version of HiveServer and supports multi-client concurrency and authentication. It is designed to provide better support for open API clients like JDBC and ODBC. HiveServer2 can run in either TCP mode or the HTTP mode.It supports Anonymous (no authentication), Kerberos, pass through LDAP etc.

1. **Configuring hiveServer2 :** We have installed the hive in /usr/ local/hive directory. In the ../hive/conf directory rename the hive-default.xml le to hive-site.xml. Conguration Properties in the hive-site.xml File

- hive.server2.thrift.port TCP port number to listen on, default 10000.
- hive.server2.thrift.bind.host TCP interface to bind to.
- hive.server2.thrift.max.worker.threads Maximum number of worker threads, de-fault 500.
- hive.server2.thrift.min.worker.threads Minimum number of worker threads, default 5.
- To enable HTTP mode:hive.server2.transport.mode Set this to http.
- hive.server2.thrift.http.port HTTP port number to listen on; default is 10001.
- Default TCP port number to listen on: 10000.

2. **Start HiveServer2 :**First start the hadoop using start-dfs.sh and start-yarn.sh commands and then use the following command to start the HiveServer2.

   hiveserver2

   To check whether its running check default TCP port is listening at 10000 or not using command:

   netstat -nl
   and Runjar in jps command.

3. **Hive in Python :**Python enables exception handling in queries. To run hive queries in python we need to install the python package pyhs2. To install it refer installation chapter.

4. **Python Client :** Run the hiveserver2 and test this python client code for HiveServer2:

```
import pyhs2
conn = pyhs2.connect(host='localhost',
port=10000,
authMechanism="PLAIN",
user='hduser',
password='test',
database='default')
```

```
cur = conn.cursor()
cur.execute("show tables")
cur.execute("select * from test")
for i in cur.fetch():
print i
cur.close()
conn.close()
```

3.2.3 Merits of using Python than R

Python is a high level language.The language provides constructs intended to enable clear programs on both a small and large scale.

R is a free software programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing statistical software.
Python and R Comparision

- The main advantage of Python over R is that it's a real programming language in the C family. It scales easily, so it's conceivable that anything you have in your sandbox can be used in production.
- The JDBC connector of python is much faster than the R hive connector.Thus the timing taken by python to return the data from hive is lesser than that in R.
- Also python has a library "Cython" used for improving computational speed in python.Cython allows one to statically type variables e.g. cdef int i declares i to be an integer. This gives massive speedups, as typed variables are now treated using low-level types rather than Python variables.

3.3 User Interface

The last step was to integrate everything such that both data loading and data visualization can be done at a single place from web framework which requires no other software installation in clients machine.

To accomplish this, we have used Django - a python based web framework. Django can easily execute python queries, make a Django web template(interface) and has support of running Hive, R queries from it. It is also capable of queuing the processes i.e., if a process takes large time to execute, other processes are queued up and completed when the previous one has finished its task.

3.3.1 Django: Basic Design and Layout [13]

1. Django project Layout

mysite/
- manage.py
- mysite/
- __init__.py
- settings.py
- urls.py
- wsgi.py

These files are as follows:

- **mysite/ :** The outer mysite/ directory is just a container for your project. Its name doesnt matter to Django; you can rename it to anything you like.
- **manage.py:** A command-line utility that lets you interact with this Django project in various ways. Type python manage.py help to get a feel for what it can do. You should never have to edit this file; its created in this directory purely for convenience.
- **mysite/mysite/ :** The inner mysite/ directory is the actual Python package for your project. Its name is the Python package name youll need to use to import anything inside it (e.g. import mysite.settings).
- **__init__.py :** A file required for Python to treat the mysite directory as a package (i.e., a group of Python modules). Its an empty file, and generally you wont add anything to it.

- **settings.py :** Settings/configuration for this Django project. Take a look at it to get an idea of the types of settings available, along with their default values.
- **urls.py :** The URLs for this Django project. Think of this as the table of contents of your Django-powered site.
- **wsgi.py :** An entry-point for WSGI-compatible webservers to serve your project. See How to deploy with WSGI (https://docs.djangoproject.com/en/1.4/howto/deployment/wsgi/) for more details.

2. Libraries imported

- To return a Http object
  - `from django.http import HttpResponse`
  - `from django.http import HttpResponseRedirect`
- For a template
  - `from django.template import Template, Context`
  - `from django.template import *`
  - `from django.shortcuts import render`
  - `from django.template.loader import get_template`
  - `from django.shortcuts import render_to_response`
  - `from django.templatetags import *`
  - `from django.template.base import Library`
- For using date-time type object
  - `import datetime`
- For using csv
  - `import csv`
- For using POST as the method to send data via form
  - `from django.core.context_processors import csrf`
- For including other processes in a process

  - `import subprocess`
- To include other views of the project

  - `from mysite import *`
- For including settings

  - `from django.conf import settings`
- For zipping up different lists together so that they can be accessed together

  - `from itertools import izip`

3. Folders to be manually added
   - **static**: Inside mysite/mysite/ to store external static files like javascript libraries, .csv files, images etc
   - **templates**: Inside mysite/mysite/ to store html templates
4. New settings.py file
   - Addition of path to the static folder
     - `STATIC_URL = '/static/'`
     - `STATICFILES_DIRS = ('path_to_directory/mysite/mysite/static/',)`
   - Adding **csrf** library so as to enable data transport using "post"
     - `MIDDLEWARE_CLASSES = ( 'django.middleware.csrf.CsrfViewMiddleware', other classes`
     - Above class should be above any other class that uses csrf
   - Addition of path to templates folder
     - Add `TEMPLATE_DIRS = ('path_to_folder/mysite/mysite/templates',)`
5. Points to remember while using Django
   - Django folder should have read and write permission by all users which can access it on the same system
   - To make django run on server, start django server by typing

  - `python manage.py runserver 0.0.0.0:8000`

- For accessing any external static file

  - Add {% load staticfiles %} once in the top of the template which uses static files
  - "A.csv" will be written as "{% static 'A.csv' %}"

- To access python variables inside template, use
  `{% variable name %}`

- To render any template, use
  `return render(request, 'templatename.html', {'variable_name': "variable_value",`
  `'variable2': variables_value})`

  - variable_name implies the variable which will be used inside the templates and its value to be substituted is passes while rendering the template

6. Django Views Syntax

```
import library1
import library2

def function1_name:
-  description
def function2_name:
-  description
..
```

7. urls.py

- Contains the url of various views and their respective templates
- `urlpatterns = patterns('',`
  `- other urls`
  `- url(r'^hello/$', "mysite.views.function1"),`
- "^" denotes start of the url and "$" denotes the end. Therefore, above example means that if url contains only **"hello"**, execute **function1** of **views** view.

Symbol	Matches
. (dot)	Any single character
\d	Any single digit
[A-Z]	Any character between A and Z (uppercase)
[a-z]	Any character between a and z (lowercase)
[A-Za-z]	Any character between a and z (case-insensitive)
+	One or more of the previous expression (e.g., + matches one or more digits)
[^/]+	One or more characters until (and not including) a forward slash
?	Zero or one of the previous expression (e.g., ? matches zero or one digits)
*	Zero or more of the previous expression (e.g., * matches zero, one or more than one digit)
{1,3}	Between one and three (inclusive) of the previous expression

Table 3.3: Special syntaxes to be used for specifying the url format

3.3.2 Visualization : Java Script Libraries

The output of the hive queries is exported to tsv or csv file. These files are used as input for plotting the graphs. We have tried the D3.js and Dimple.js to plot the different kinds of charts.
D3.js [28] is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS.
Dimple [29] is a library to aid in the creation of standard business visualisations based on d3.js.We must include d3.js in any pages in which we wish to use dimple.

- **D3.js and Dimple.js** To use D3.js and Dimple.js one must include the following scripts in their html pages. A local source path of these scripts files can also be used.

```
<script src="http://d3js.org/d3.v3.min.js"></script>
<script src="http://dimplejs.org/dist/dimple.v1.min.js">
</script>
```

These can also be stored locally and referenced.

```
<script src="js/d3.min.js"></script>
<script src="js/dimple.min.js"></script>
```

- **Load data and create bar chart** First we need to create a new svg container, specifying its dimensions and div in which it will be placed. Then we load our data. The dataset used is stored in a csv file. We need to set bounds of the chart, and specify what data is to be added to x- and y-axis. D3 allows to take input from different types of files (tsv,xls) etc.
  Load data from csv file

```
<script type="text/javascript">
        var svg = dimple.newSvg("#chartContainer", 450, 450);
        d3.csv("data.csv", function (data) {
            var myChart = new dimple.chart(svg, data);
            myChart.setBounds(60, 10, 400, 330)
            var x = myChart.addCategoryAxis("x","Age");
            myChart.addMeasureAxis("y", "user");
            var s = myChart.addSeries(null, dimple.plot.bar);
            s.barGap = 0.05;
            myChart.draw();
        });
    </script>
```

- **Customise axis label and position**

```
// Override x-axis title
x.titleShape.text("Students Age");
// Title is placed below the tick labels by default.
// This overrides this setting and places it immediately
// below the axis.
x.titleShape.attr("y", myChart.height + 55);
```

- **Some charts drawn on edX data**

  1. This grouped step line chart shows the number of males and females registered in each course. 3.11

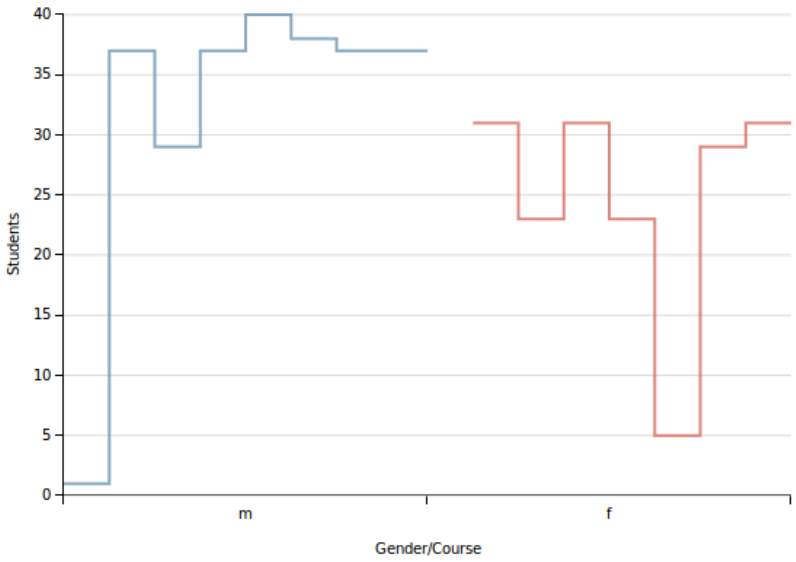


Figure 3.11: Registered students

  2. This pie chart shows the age distribution of the students. 3.12

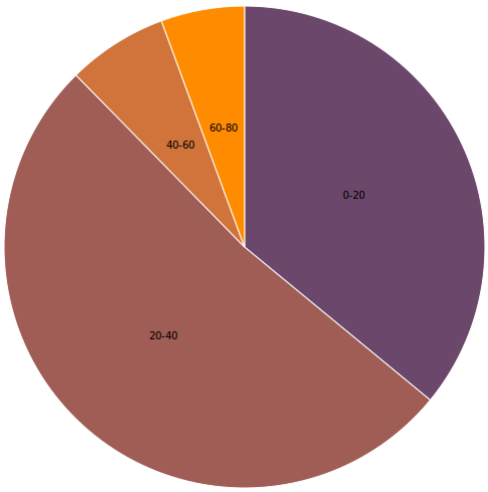


Figure 3.12: Age distribution pie chart

3. This unstacked grouped vertical bar chart shows the correct and incorrect responses of students for each problem id. 3.13

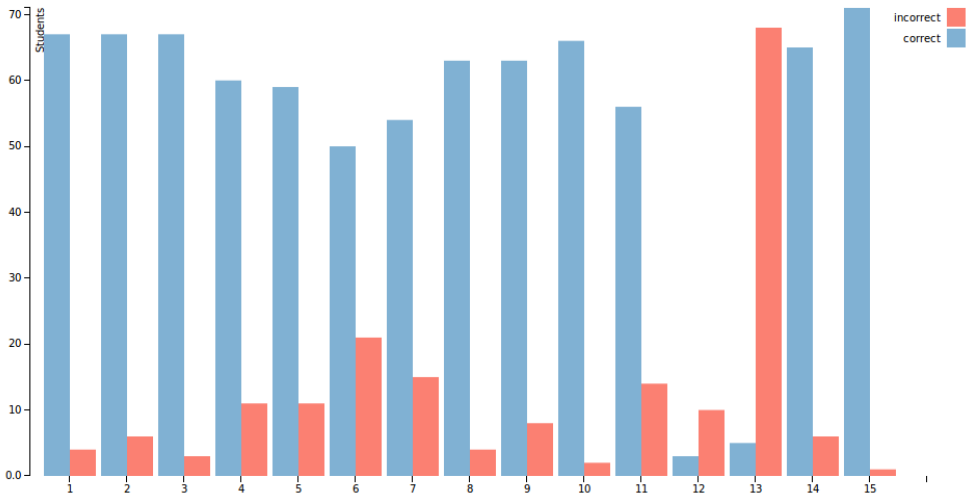


Figure 3.13: Correct/Incorrect responses for problems

4. This Donut chart represents the degree distribution of the students. 3.14

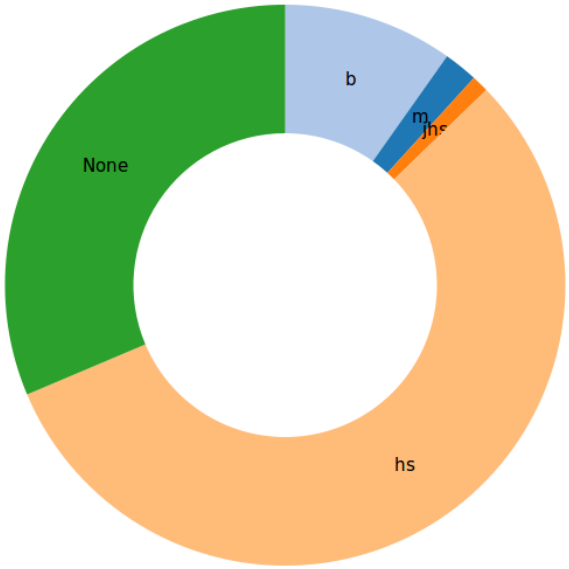


Figure 3.14: Degree distribution

3.3.3 Google Charts

1. **Using Google Charts**

   Google Charts provides a perfect way to visualize data on your website. From simple line charts to complex hierarchical tree maps, the chart gallery provides a large number of ready-to-use chart types.

   The most common way to use Google Charts is with simple JavaScript that you embed in your web page. You load some Google Chart libraries, list the data to be charted, select options to customize your chart, and finally create a chart object with an id that you choose. Then, later in the web page, you create a ¡div¿ with that id to display the Google Chart.That's all you need to get started.

   Charts are exposed as JavaScript classes, and Google Charts provides many chart types for you to use. The default appearance will usually be all you need, and you can always customize a chart to fit the look and feel of your website. Charts are highly interactive and expose events that let you connect them to create complex dashboards or other experiences integrated with your webpage. Charts are rendered using HTML5/SVG technology to provide cross-browser compatibility (including VML for older IE versions) and cross platform portability to iPhones, iPads and Android. Your users will never have to mess with plugins or any software. If they have a web browser, they can see your charts.

   All chart types are populated with data using the DataTable class, making it easy to switch between chart types as you experiment to find the ideal appearance. The DataTable provides methods for sorting, modifying, and filtering data, and can be populated directly from your web page, a database, or any data provider supporting the Chart Tools Datasource protocol. (That protocol includes a SQL-like query language and is implemented by Google Spreadsheets, Google Fusion Tables, and third party data providers such as SalesForce. You can even implement the protocol on your own website and become a data provider for other services.)

Figure 3.15: Using Google Charts

2. **Quick Start With an Example**

   Here's a simple example of a page that displays a pie chart.

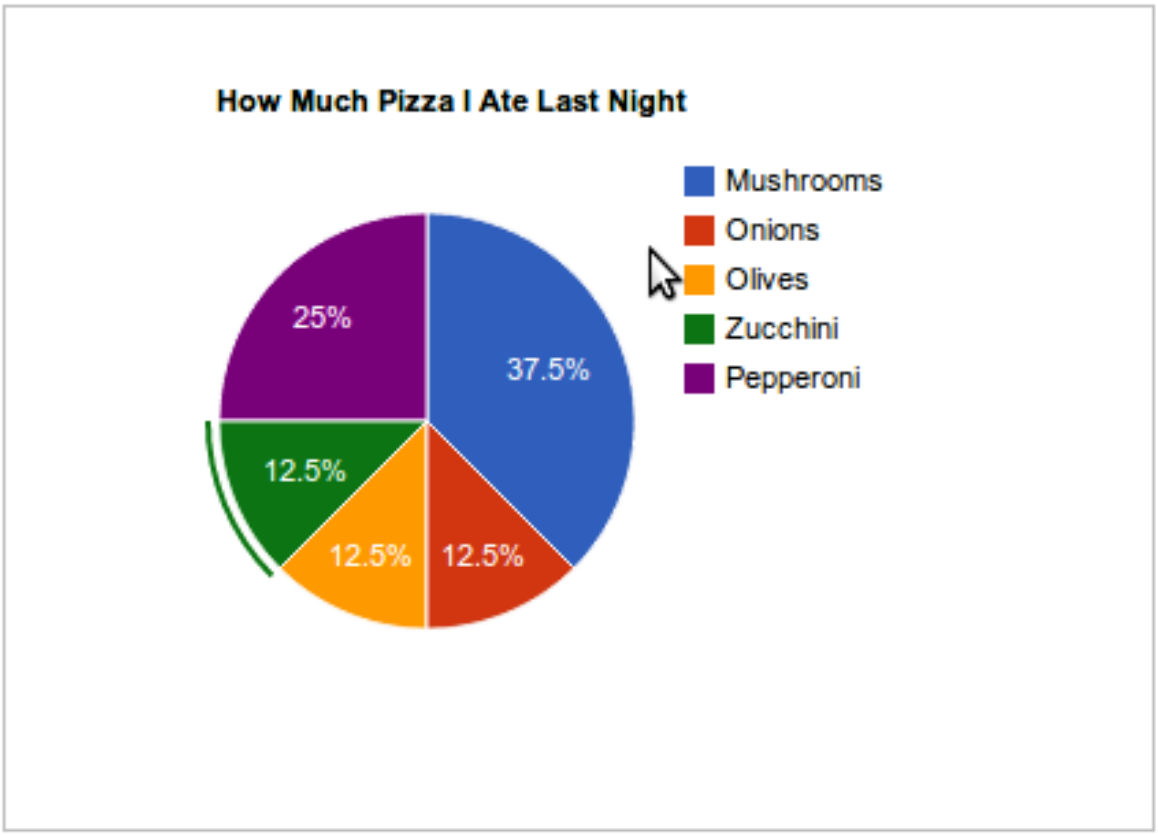


Figure 3.16: Pie Chart Example

```
<html>
<head>
<!--Load the AJAX API-->
<script type="text/javascript" src=
    "https://www.google.com/jsapi">
    </script>
    <script type="text/javascript">

    // Load the Visualization API
    and the piechart package.
    google.load('visualization','1.0',
    {'packages':['corechart']});

    // Set a callback to run when the
    Google Visualization API is loaded.
    google.setOnLoadCallback(drawChart);

    // Callback that creates and populates a data table,
    // instantiates the pie chart, passes in the data and
    // draws it.
    function drawChart() {

    // Create the data table.
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Topping');
    data.addColumn('number', 'Slices');
    data.addRows([
       ['Mushrooms', 3],
       ['Onions', 1],
       ['Olives', 1],
       ['Zucchini', 1],
       ['Pepperoni', 2]
    ]);

    // Set chart options
    var options = {'title':'How Much Pizza I Ate Last Night',
                   'width':400,
                   'height':300};
```

```
    // Instantiate and draw our chart, passing in some options.
    var chart = new google.visualization.PieChart
              (document.getElementById('chart_div'));
    chart.draw(data, options);
    }
   </script>
 </head>

<body>
   <!--Div that will hold the pie chart-->
   <div id="chart_div"></div>
</body>
</html>
```

For other charts like Barchart one need to just replace **google.visualization.PieChart** with **google.visualization.BarChart** in the code and reloading your browser.

3. **Loading the Libraries**

   A chart requires three libraries:

   - The Google JSAPI API
   - The Google Visualization library
   - The library for the chart itself

   These libraries are loaded using two $<$ *script* $>$ links in your page code, as shown here:

```
<!--Load the AJAX API-->
<script type="text/javascript" src="https://www.google.com/jsapi">
</script>
<script type="text/javascript">

// Load the Visualization API library and the piechart library.
google.load('visualization', '1.0',
{'packages':['corechart']});
google.setOnLoadCallback(drawChart);
// ... draw the chart...
</script>
```

The first script tag loads the JSAPI library.

The second script loads the Google Visualization and chart libraries. It also typically holds your chart code.

The first line of the second script tag should call **google.load()**. This function takes the following syntax:

```
google.load('visualization','1.0',
{'packages':[<list_of_package_names>]});
```

- **visualization**
  Loads the **google.visualization** library. This library defines all the core utility classes and functions.

- **1.0**
  Which version of the visualization to load. 1.0 is always the current production version.

- **list_of_package_names**
  A list of Google chart libraries to load. The **'corechart'** library in the example defines most basic charts, including the **pie, bar, and column charts**. Any other Google charts not defined in this library that you want to include on your page must be added as separate array entries. Each chart's documentation lists which library it is defined in. For example, here is how to load the core charts plus a table chart: google.load('visualization', '1.0', 'packages':['corechart','table']);

  Immediately after calling google.load(), your code should call google.setOnLoadCallback(my_handler), a JSAPI function that calls your handler as soon as all the libraries are loaded. Your handler function should create and define the chart, as described next

4. **Preparing Data**

   All charts require data. Google Chart Tools charts require data to be wrapped in a JavaScript class called google.visualization.DataTable.

This class is defined in the Google Visualization library that you loaded previously.

A DataTable is a two-dimensional table with rows and columns, where each column has a datatype, an optional ID, and an optional label. The example above creates the following table:

type: string label: Topping	type: number label: Slices
Mushrooms	3
Onions	1
Olives	1
Zucchini	1
Pepperoni	2

Figure 3.17: DataTable

There are several ways to create a DataTable; you can see a list and comparison of each technique in **DataTables and DataViews**. You can modify your data after you add it, and add, edit, or remove columns and rows.

You must organize your chart's DataTable in a format that the chart expects: for instance, both the Bar and Pie charts require a two-column table where each row represents a slice or bar. The first column is the slice or bar label, and the second column is the slice or bar value. Other charts require different and possibly more complex table formats.

Rather than populate a table yourself, you could instead query a website that supports the Chart Tools Datasource protocol–for example, a Google Spreadsheets page. Using the **google.visualization.Query** object, you can send a query to a website and receive a populated DataTable object that you can pass into your chart. This thing is properly explained in **Querying a Datasource** to learn how to send a query

5. **Customizing the Chart**

```
// Set chart options
 var options = {'title':'How Much Pizza I Ate Last Night',
                'width':400,
                'height':300};
```

Every chart has many customizable options, including title, colors, line thickness, background fill, and so on.

The following object defines the legend position, chart title, chart size, and a 3D option for a Pie Chart:

```
var options = {
  'legend':'left',
  'title':'My Big Pie Chart',
  'is3D':true,
  'width':400,
  'height':300
}
```

6. **Drawing the Chart**

```
// Instantiate and draw our chart, passing in some options.
   var chart = new google.visualization.PieChart
   (document.getElementById('chart_div'));
   chart.draw(data, options);
   ....
   ....
   <!--Div that will hold the pie chart-->
   <div id="chart_div" style="width:400; height:300"></div>
```

The last step is to draw the chart. First you must instantiate an instance of the chart class that you want to use, and then you must call draw() on the it

- **Instantiating Your Chart**
  Each chart type is based on a different class, listed in the chart's documentation.

For instance, the pie chart is based on the **google.visualization.PieChart** class, the bar chart is based on the **google.visualization.BarChart** class, and so on. Both pie and bar charts are included in the corechart package that you loaded at the beginning of this tutorial. However, if you want a **treemap** or **geo chart** on your page, you must **additionally load the 'treemap' or 'geomap' packages.**

Google chart constructors take a single parameter: a reference to the DOM element in which to draw the visualization.

```
var chart =
new google.visualization.
PieChart(document.getElementById('chart_div'));
```

- **Drawing your chart**

  After you've prepared your data and options, you are ready to draw your chart using chart.draw() and function drawChart()

7. **Chart Gallery**

   Charts gallery provides a variety of charts designed to address your data visualization needs. These charts are based on pure HTML5/SVG technology (adopting VML for old IE versions), so no plugins are required. All of them are interactive, and many are pannable and zoomable. Adding these charts to your page can be done in a few simple steps.

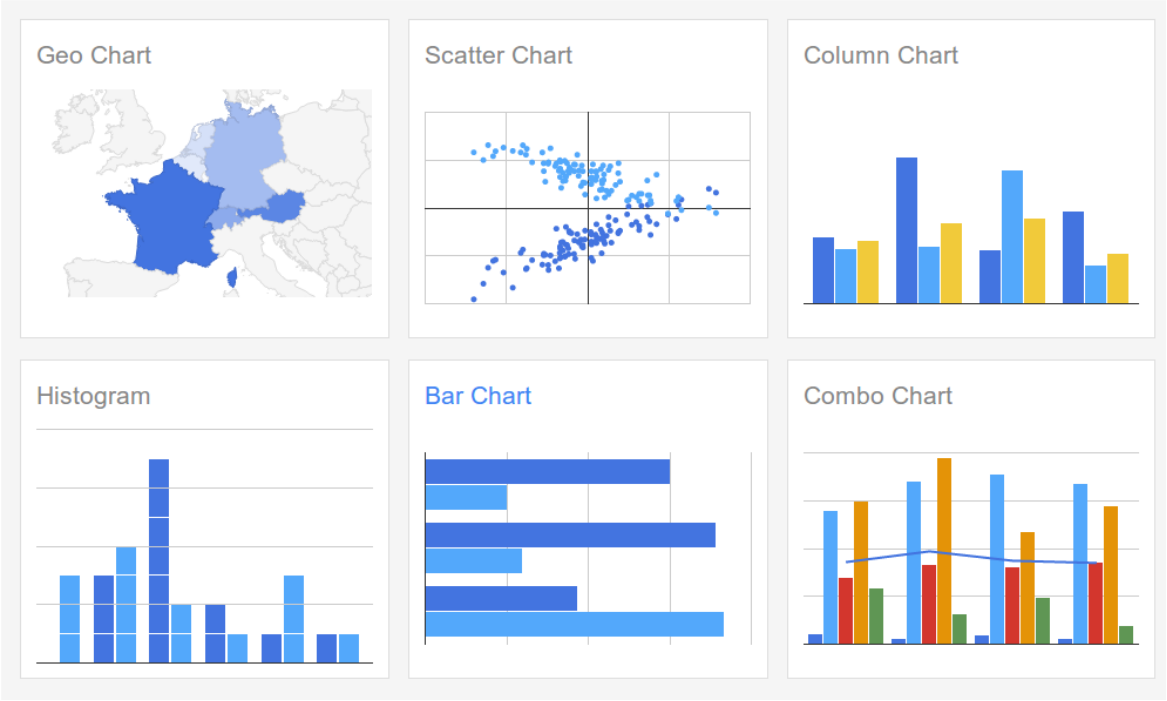


Figure 3.18: Chart Gallery

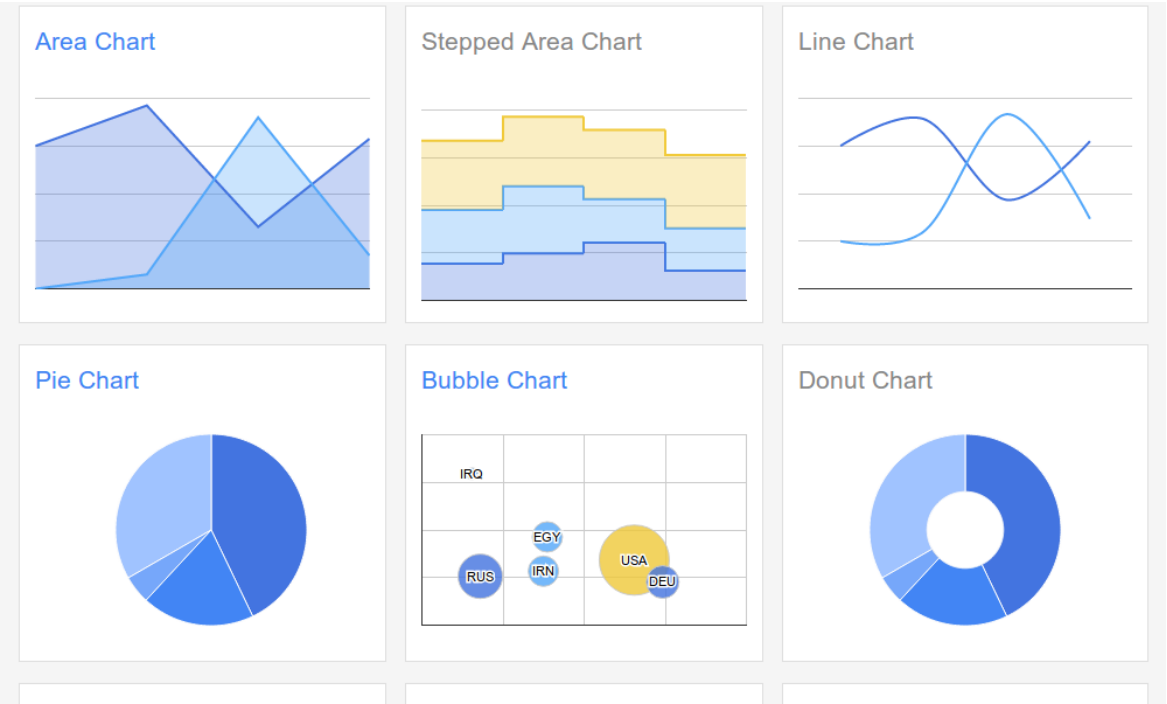


Figure 3.19: Chart Gallery

8. **Data Loading Into Charts**

The data can be loaded into google charts in three ways mainly:

- **Populating it manually**

  The data can be manually feeded to the charts as follows:

```
// ------- Version 1------------
// Add rows + data at the same time
// ----------------------------
var data = new google.visualization.DataTable();

// Declare columns
data.addColumn('string', 'Employee Name');
data.addColumn('DateTime', 'Hire Date');

// Add data.
data.addRows([
['Mike', {v:new Date(2008,1,28), f:'February 28, 2008'}],
// Example of specifying actual and formatted values.
```

```
['Bob', new Date(2007,5,1)],
// More typically this would be done using a
['Alice', new Date(2006,7,16)],
// formatter.
['Frank', new Date(2007,11,28)],
['Floyd', new Date(2005,3,13)],
['Fritz', new Date(2011,6,1)]
]);

// ------- Version 2------------
// Add empty rows, then populate
// -----------------------------
var data = new google.visualization.DataTable();
// Add columns
data.addColumn('string', 'Employee Name');
data.addColumn('date', 'Start Date');
// Add empty rows
data.addRows(6);
data.setCell(0, 0, 'Mike');
data.setCell(0, 1, {v:new Date(2008,1,28),
f:'February 28, 2008'});
data.setCell(1, 0, 'Bob');
data.setCell(1, 1, new Date(2007, 5, 1));
data.setCell(2, 0, 'Alice');
data.setCell(2, 1, new Date(2006, 7, 16));
data.setCell(3, 0, 'Frank');
data.setCell(3, 1, new Date(2007, 11, 28));
data.setCell(4, 0, 'Floyd');
data.setCell(4, 1, new Date(2005, 3, 13));
data.setCell(5, 0, 'Fritz');
data.setCell(5, 1, new Date(2007, 9, 2));

var data = google.visualization.arrayToDataTable([
       ['Employee Name', 'Salary'],
       ['Mike', {v:22500, f:'18,500'}],
       ['Bob', 35000],
       ['Alice', 44000],
       ['Frank', 27000],
       ['Floyd', 92000],
```

```
        ['Fritz', 18500]
      ],
      false);
```

- **Loading it from local csv file**

```
$.get("http://localhost/AirPassengers.csv",
function(csvString) {
//transform the CSV string into a 2-dimensional array
 var arrayData = $.csv.toArrays(csvString,
 {onParseValue: $.csv.hooks.castToScalar});
 var data =
 new google.visualization.arrayToDataTable(arrayData);
var view = new google.visualization.DataView(data);
view.setColumns([{calc:stringID, type: "string"},0,1,2]);
      ....
      ....
```

- **Loading it from Google Spreadsheets and querying it**

```
function drawVisualization() {
var query = new google.visualization.Query(
'http://spreadsheets.google.com/
tq?key=pCQbetd-CptGXxxQIG7VFIQ&pub=1');

// Apply query language statement.
query.setQuery('SELECT A,D WHERE D > 100 ORDER BY D');

// Send the query with a callback function.
query.send(handleQueryResponse);
}

function handleQueryResponse(response) {
if (response.isError()) {
alert('Error in query: ' + response.getMessage() +
' ' + response.getDetailedMessage());
return;
}

var data = response.getDataTable();
visualization = new google.visualization.LineChart
```

```
(document.getElementById('visualization'));
visualization.draw(data, {legend: 'bottom'});
}
```

9. **User Interface Developed for Data Visualization Using Google Charts**

   The entire frontend (UI) is developed using **html,css,javascript,jquery,php**. The complete flow of web pages is described in the flow chart as follows:

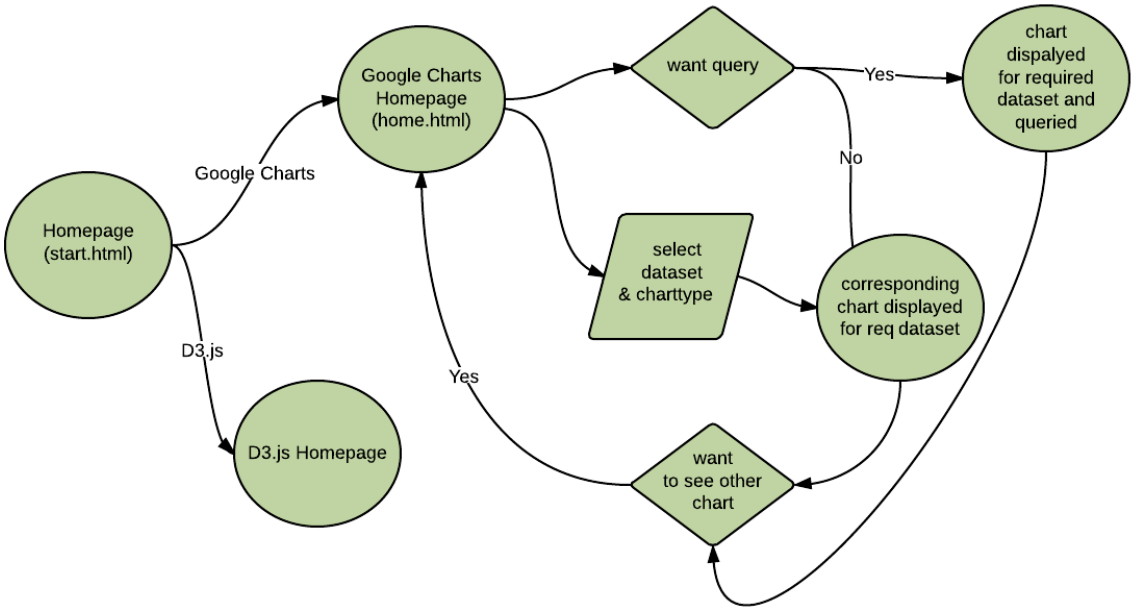


Figure 3.20: Flow of web pages

10. **Datasets Used**

   (a) **AirPassengers Dataset(test dataset**

   AirPassengers Dataset basically contains the id ,month of travel(time) and total no. of passengers travelled in that particular month(AirPassengers

   (b) **ResponseTime Dataset**

   The ResponseTime Dataset basically contains the quiz data that was conducted for the summer interns in IIT Bombay . It contains id, username,course,pid(problem id) and timeinsecs

taken by the student to answer that particular problem.

- **Homepage(start.html)**

  The home page to the data visualisation where you are provided the with introduction of Data Visualization, homepage of Google Charts and D3.js .Alongside the links to the Google Charts and D3 is provided

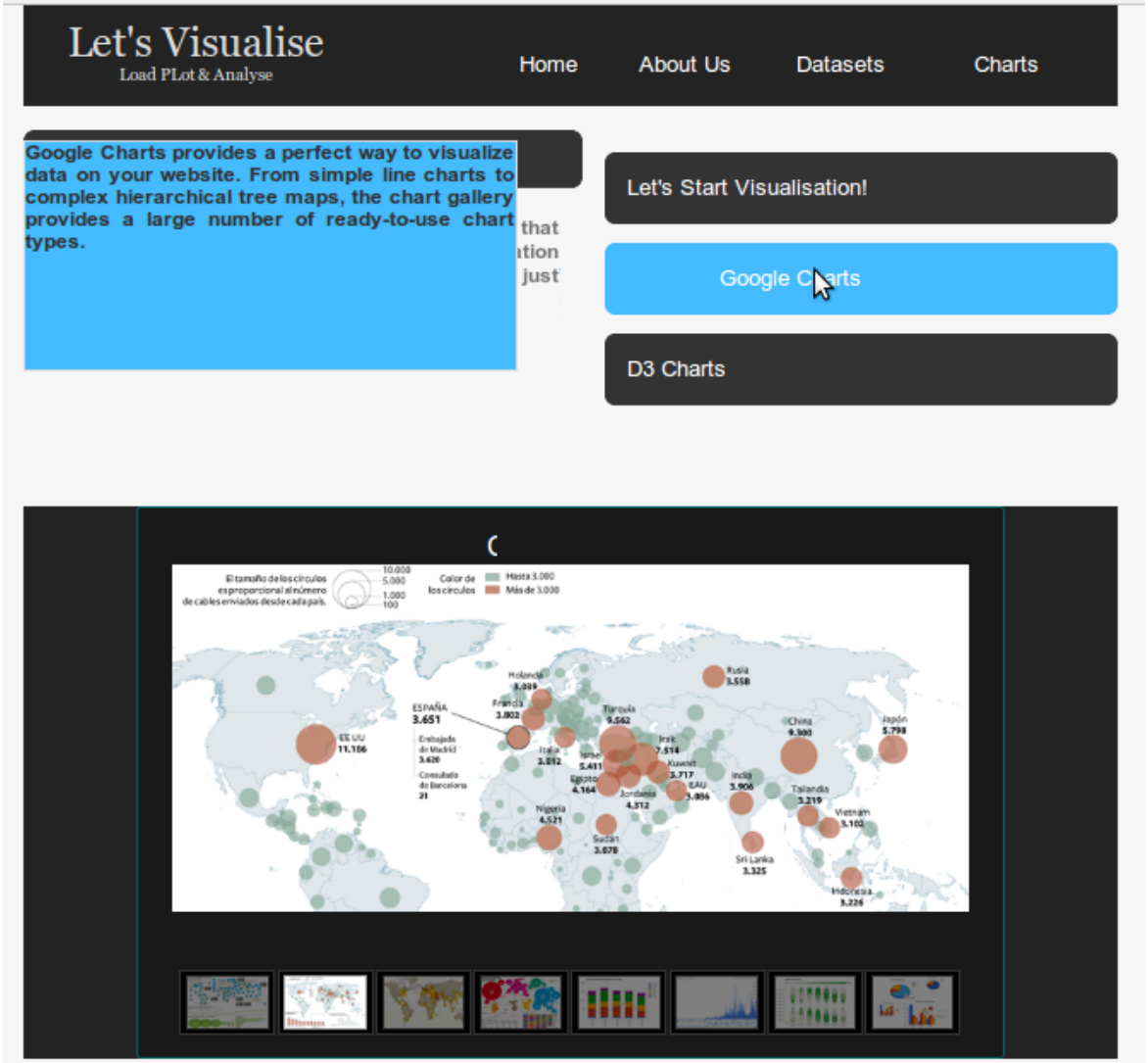


Figure 3.21: Selecting Google Charts

- **Google Charts Homepage(home.html)**

  The home page of the Google Charts where you are provided the with introduction to Google Charts .Alongside the web form for Chart Settings consisting of Dataset , Charttype selection and flexibility to query or not is provided

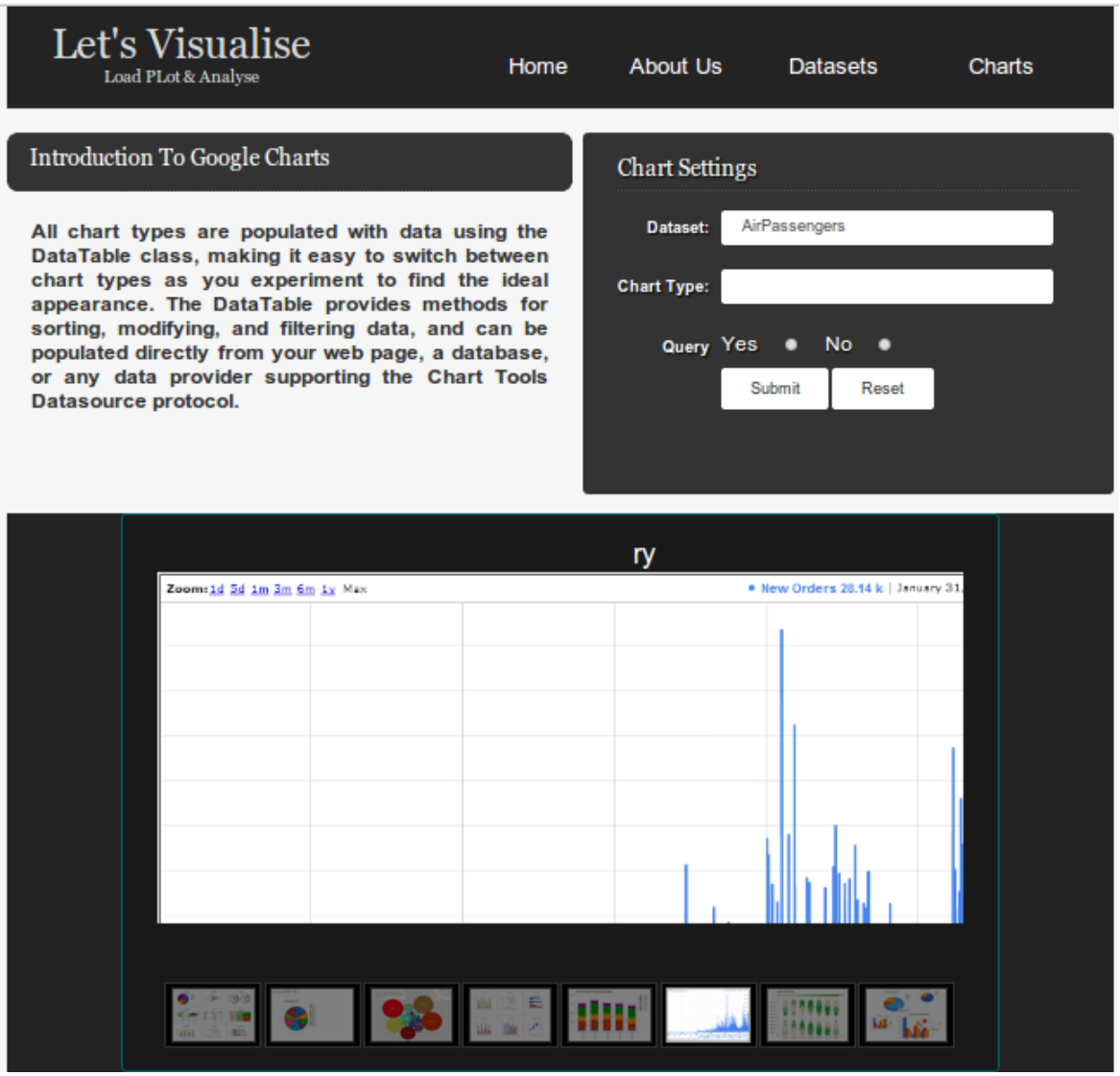


Figure 3.22: Google Charts Homepage

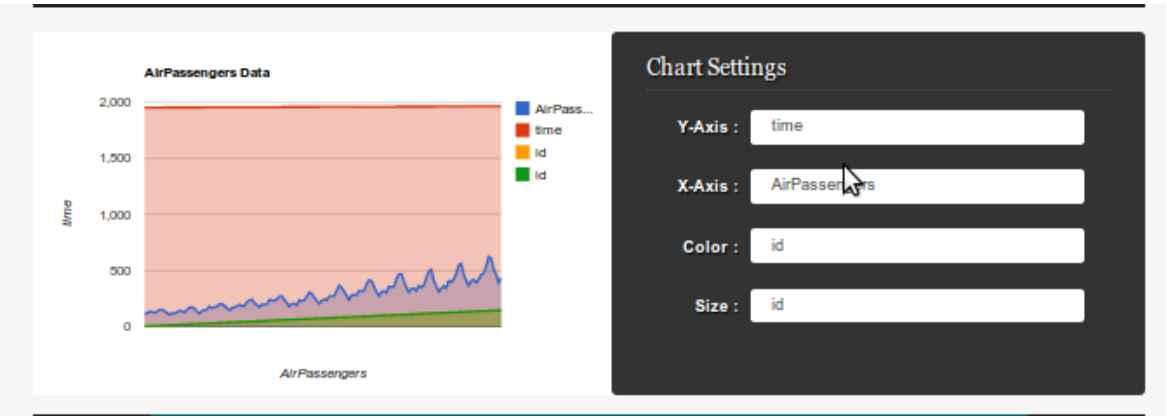


Figure 3.23: Areachart for AirPassengers Dataset(test data)

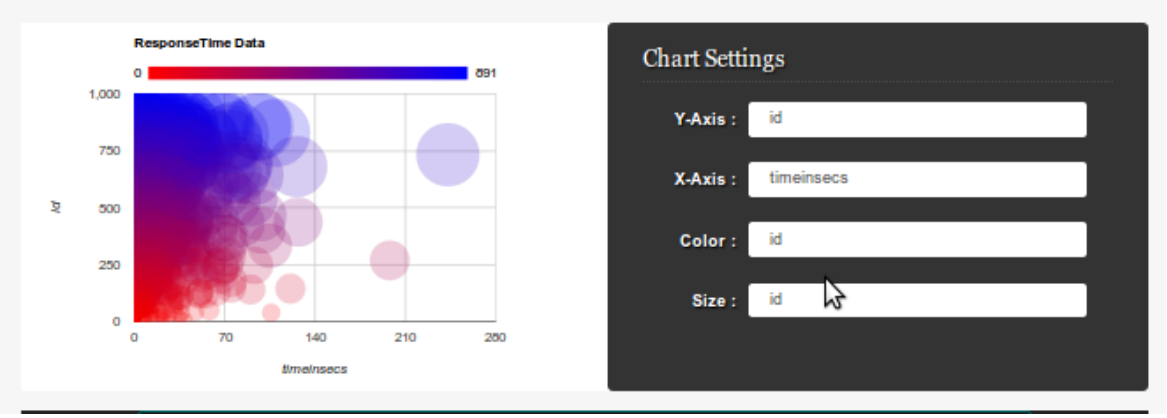


Figure 3.24: Bubblechart for Responsetime dataset

– Running a Query

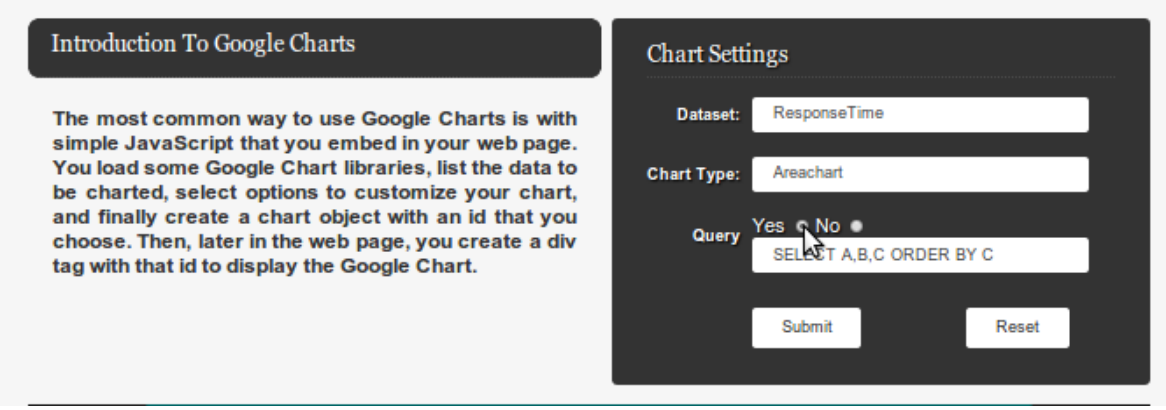


Figure 3.25: Running a query

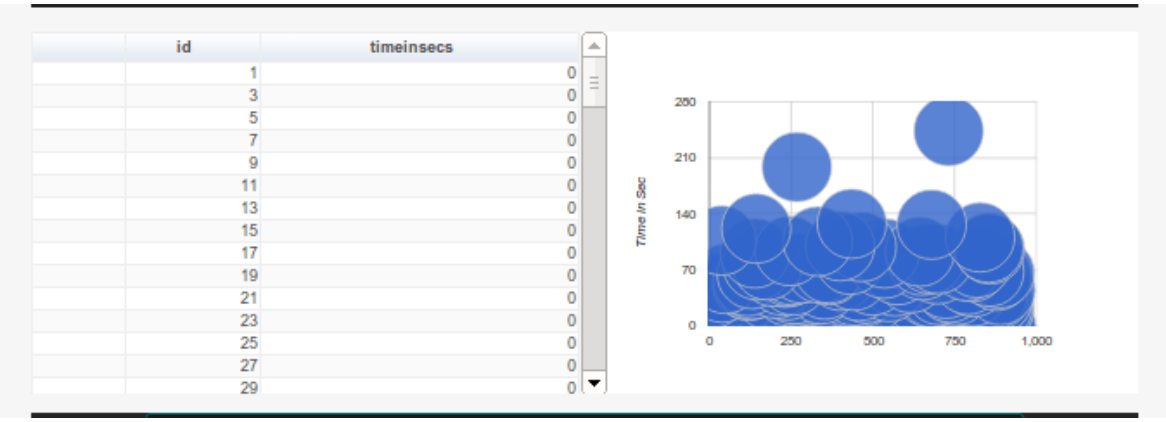


Figure 3.26: BubbleChart of ResponseTime Dataset

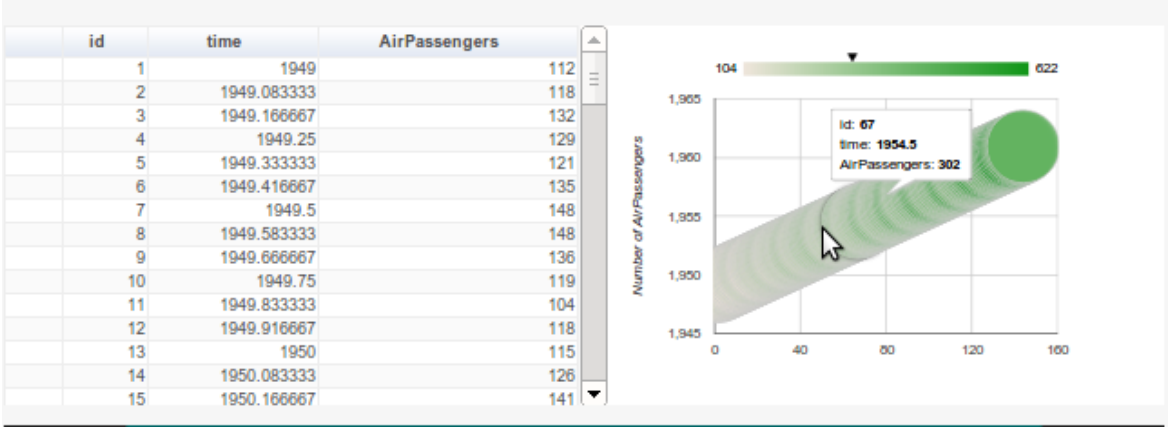


Figure 3.27: BubbleChart of AirPassengers Dataset

- **Selecting Dataset**

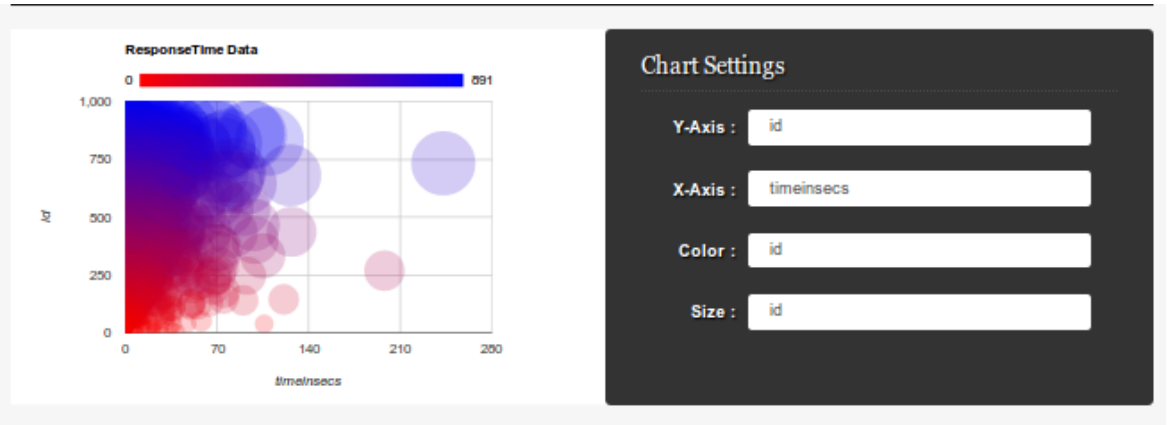


Figure 3.28: Selecting AirPassengers Dataset from Datasets Tab

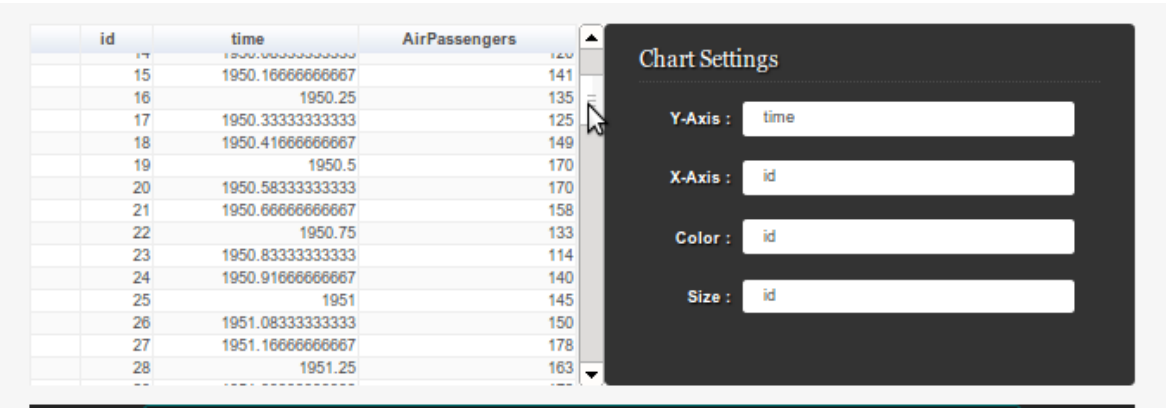


Figure 3.29: AirPassengers Dataset Selected

3.3.4 Integration

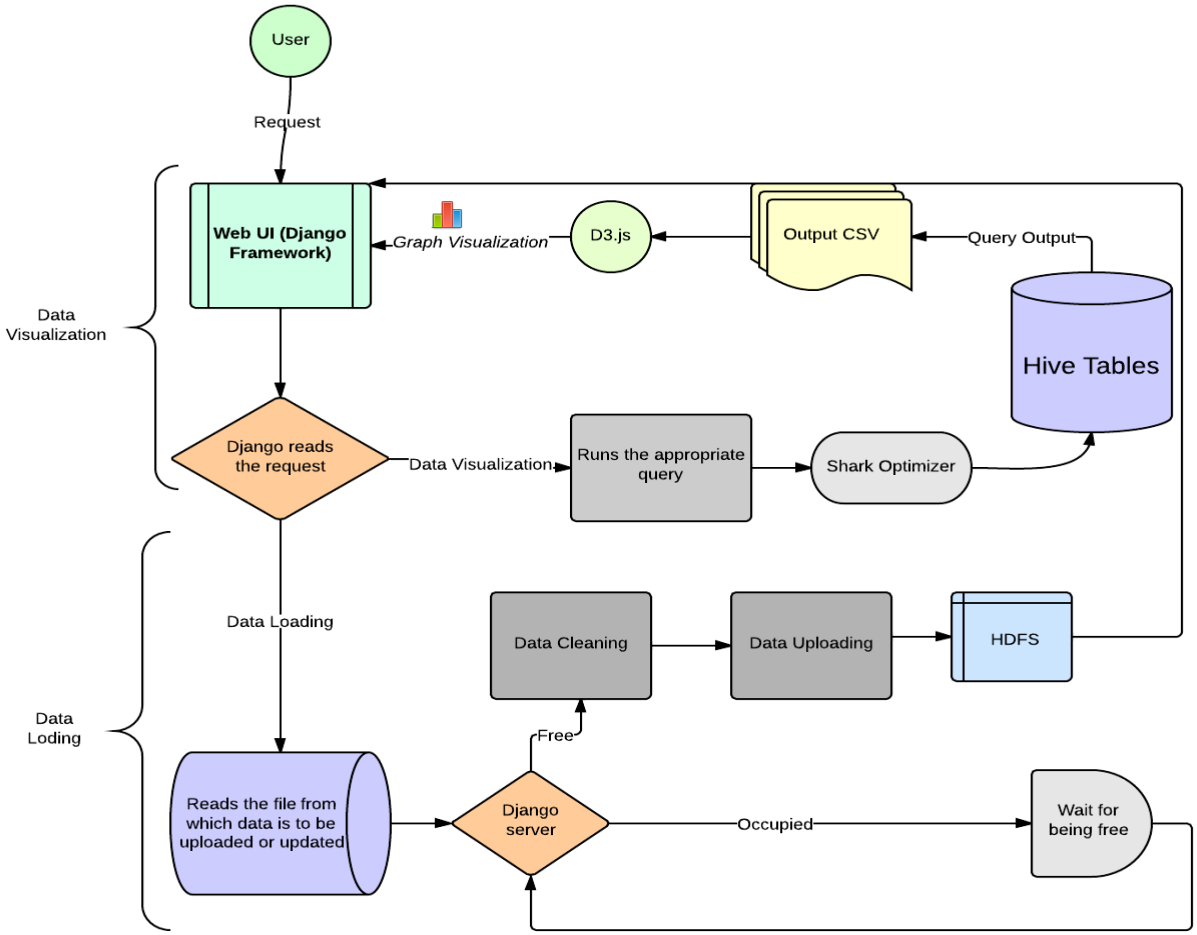


Figure 3.30: Combined Working

Whenever user visits the interface, he receives two options

- Data Loading: To load the data onto HDFS

- Data Visualization: To visualize things like records, progress, enrollment status of students

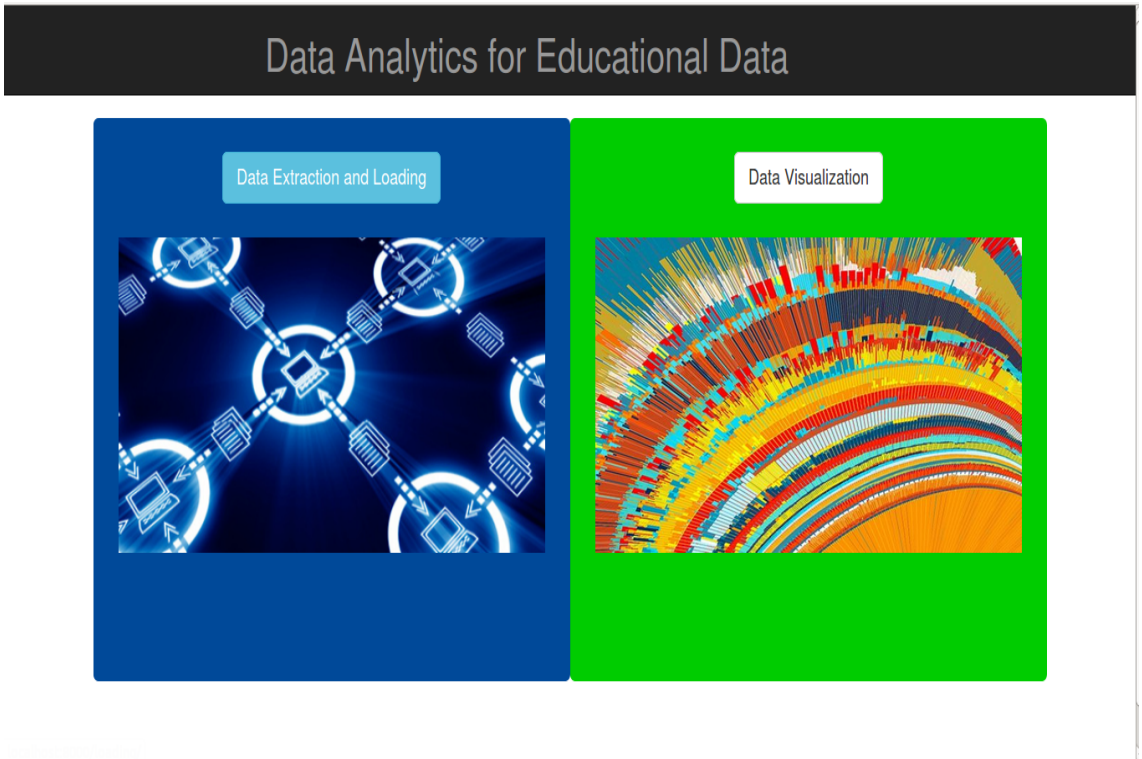


Figure 3.31: Home Page

Data Loading

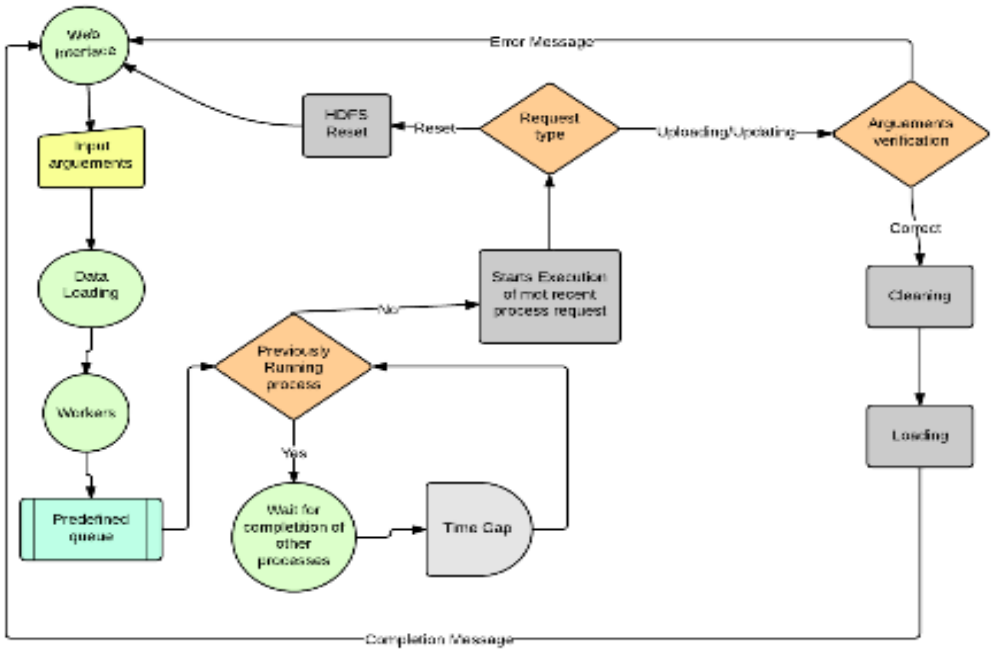


Figure 3.32: Data Loading

On choosing this option, he gets a form on which client has to fill up all the required details to upload, update or reset HDFS.
On clicking the submit button, data is sent through a Django server to a function which starts the process. First of all, it is checked whether some other process is already running or not.
If not, then current process starts by passing the required information to a python script which depending on inputs starts with the cleaning of the data stored in the path given by user followed by loading onto HDFS. Before cleaning starts,it is also verified whether the path given contains appropriate data for processing. If data found is not proper, then a message is sent from the script to the django views showing the details which is then shown to the user through a web interface and asking him for correct details.
If a process was already running, the latest instruction cannot be performed as this can cause HDFS to be in inconsistent state, Also, user instruction cannot be made to lost if not executed. To solve this issue, a queuing mechanism is used using **django-rq**
Django-rq [23] is a a queuing mechanism based on redis and rq.It allows

us to create multiple queues(are defined in settings.py file) out of which as per the programmers choice, any queue can be used without affecting other queues and easily put jobs into them. Jobs kept into queue are executed as and when they previous job gets completed. Till then they are stored somewhere waiting for their turn. There are workers which reads the request of enqueuing processes and runs them. These rq workers need to be active when request is made otherwise processes will not be able to execute

- `python manage.py rqworker`
  Used to start workers

- `use_connection()`
  `queue=Queue(connection=Redis())`
  is used to establish connection to the default queue

- `y=queue.enqueue_call(func=main,args=`
  `(path,mode,server,username,password,), timeout=500)`
  It is used to enqueue the process in function "main" with arguements and timeout period to be 500 sec. Timeout signifies the maximum time for which output of the preocess will be stored after its completion.

- `y.id`
  It is used to fetch the id of the process just enqueued

- `job=Job().fetch(x)`
  It is used to fetch the details of the job using its id

- `job.is_finished` To check whether the job is finished

- `job.is_failed` To check whether the job has failed

Data Visualization

After we have data in our HDFS, various queries can be made to see the enrollment status in a course, performance of students and various other stuffs.

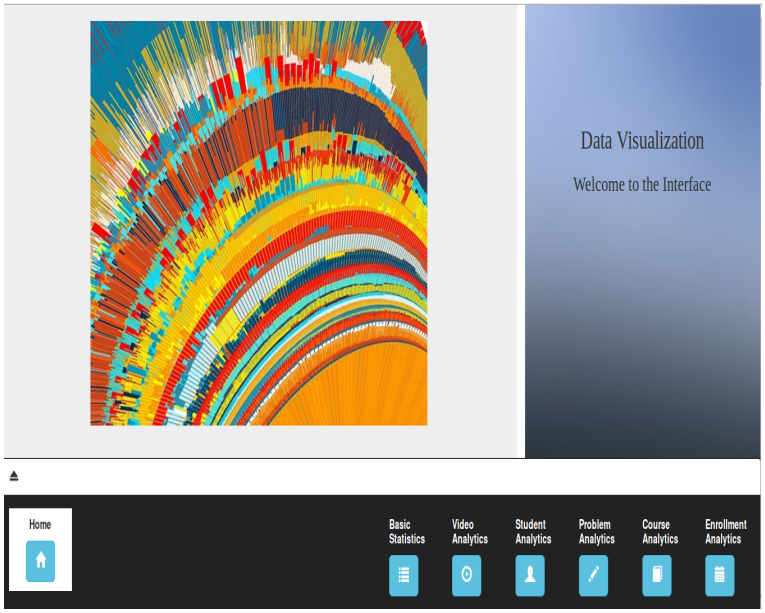


Figure 3.33: Data Visualization 1

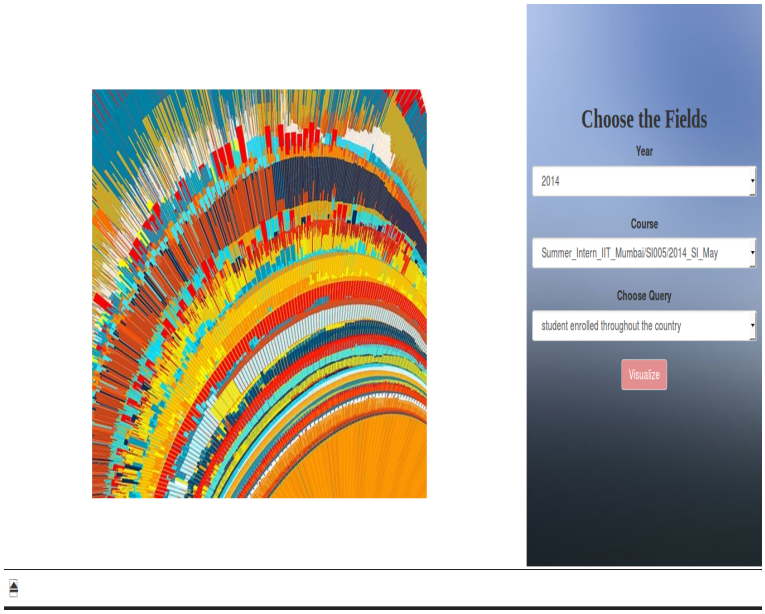


Figure 3.34: Data Visualization 2

Depending on what user chooses from the navigation bar at the bottom of the page, options in the right-side of the page changes. User is provided with six buttons in the navigation bar:

- **Home**: Redirects to the Home page
- **Basic Statistics**: Depending on the course, gives the last of all currently enrolled students
- **Video Analyics**: Analytics based on the videos watched by students
- **Student Analytics**: Analytics based on student's general information like gender, age
- **Problem Analytics**: Analytics based on their marks and response times in sloving question
- **Course Analytics**: Analytics based on the steps student follow to go through the course
- **Enrollment Analytics**: Analytics showing enrollment from all over the country

Whenever user chooses any of these, he gets a form from which he has to choose course, year and query which is to be run. On clicking `Visualize` button, data is sent to the django view function which on checking what data is sent from where and which function and query are to be executed, calls the appropriate Django template. Simultaneously, a script is called which starts querying the data from HDFS and after completition of querying stores the output in csv ot tsv file. That file is given as input to html files which contain javascript libraries(d3 and dimple) to plot the graph. After this process is completed, Django templated is rendered and output graph is shown. Again user will get other options to view other query outputs **An example showing how it is decided which template is to be rendered along with data that is to be sent**

```
if (request.method == "POST"):

if 'studentb' in request.POST:
  stq = request.POST.get("studentq", "1")
  sty = request.POST.get("studentyear", "2014")
  stc = request.POST.get('studentcourse',
  'Summer_Intern_IIT_Mumbai/SI003/2014_SI_May')
  if stq == "0":
   flag=ageDistributionByCourse(stc)
   file1 = "agedistro.html"
   if(flag==1):
     return rrender(request,"oops.html",{})
     time.sleep(3)
     elif stq == "1":
      flag=studentMarkStats(stc)
      ile1 = "marks.html"
     if(flag==1):
      return render(request,"oops.html",{})
     else:
      flag=degreeDistributionByCourse(stc)
      file1 = "degreedist.html"
     if(flag==1):
      return render(request,"oops.html",{})
     time.sleep(4)
      return render(request, 'visualstudent.html',
     {'v1': v1, 'v2': v2, 'v3': v3, 'year':
     years, 'course': x, 'vid1q': vid1q,
     studq': studq, 'vidq': vidq,
     'enrollq': enrollq, 'probq': probq, 'temp': file1},
     context_instance=RequestContext(request))
```

Example code of the functions running that queries and generate csv file

```
def ageDistributionByCourse(course):
    c = getHiveConn("localhost", 10000,
    "PLAIN", "hduser", "", "localdb")
    try:
      usersSubQ = '(select distinct user_id
      from log_table where course_id="' + course + '"
      and username is not null and username!="")users'
      agesPC = DataFrame(columns=["range", "number"])
      ageDis = execute(c,
          'select users.user_id,
          (year(from_unixtime(unix_timestamp()))
          - auth_userprofile.year_of_birth)
           as age from ' + usersSubQ + '
           join auth_userprofile on
           auth_userprofile.user_id=users.user_id')
     for i in range(4):
           agesPC.loc[agesPC.shape[0]] =
           [str(i * 20) + "-" + str((i + 1) * 20), len(
          ageDis[(ageDis.age > i * 20) &
          (ageDis.age <= ((i + 1) * 20))]["age"].tolist())]
      ageDis.to_csv("/home/hduser/django/testex
      /testex/static/ageDis.tsv", sep="\t")
      agesPC.to_csv("/home/hduser/django/testex/testex/static
      /ageDisPC.tsv", sep="\t")
    except ValueError:
        return 1
    c.close()
    return 0
```

Example showing the html file which uses csv and tsv file

```
<body>
{% load staticfiles %}
<script src="{% static 'd3.min.js' %}"></script>
----
var pie = d3.layout.pie()
    .sort(null)
    .value(function(d) { return d.population; });

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height)
  .append("g")
    .attr("transform",
    "translate(" + width / 2 + "," + height / 2 + ")");
d3.csv("{% static 'ageDisPC.csv' %}", function(error, data) {
  data.forEach(function(d) {
  d.population = +d.population;
});
```

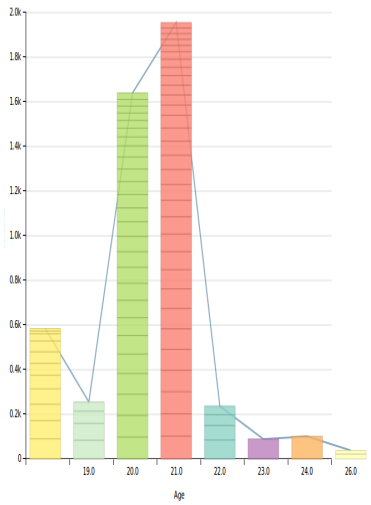
Example of a template which includes other html file having query output: visulastudent.html

```
<div id="head1" style="width:68%;padding-top:10px;float:left;">
   <center>
   <div class="jumbotron" style="margin-top:5px;
   display:none;background-color:white;
    background-size: cover" id="resetd1">
    <br>
    <br>
    <img src="{% static 'images/visualization.jpg' %}"
    class="img-responsive">
    </div>
    <div style="margin-top:10px;margin-left:30px;
    width:700px" name="htm" id="htm">
    </center>
    {% include temp %}
```

```
</center>
</div>
<div style="margin-top:10px;display:none"
name="studente" id="studente">
</center>
imgstudent
</center>
```

Student Analytics
Contains three options

- Age Distribution chart



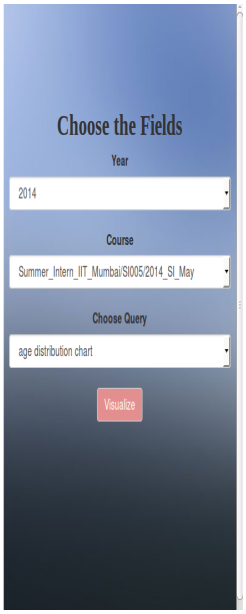
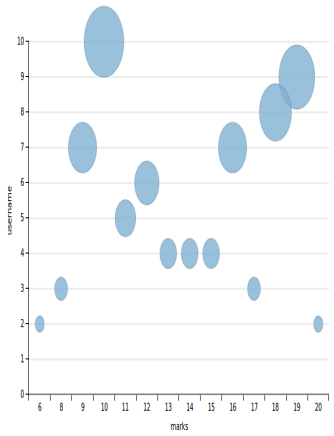


Figure 3.35: Age Distribution

- Student Marks Statistics



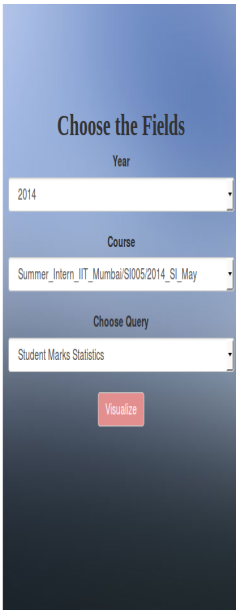


Figure 3.36: Student Marks

- Degree Distribution chart

Course Analytics

Contains two options

- User activity

Figure 3.37: User Activity

- Event sequence

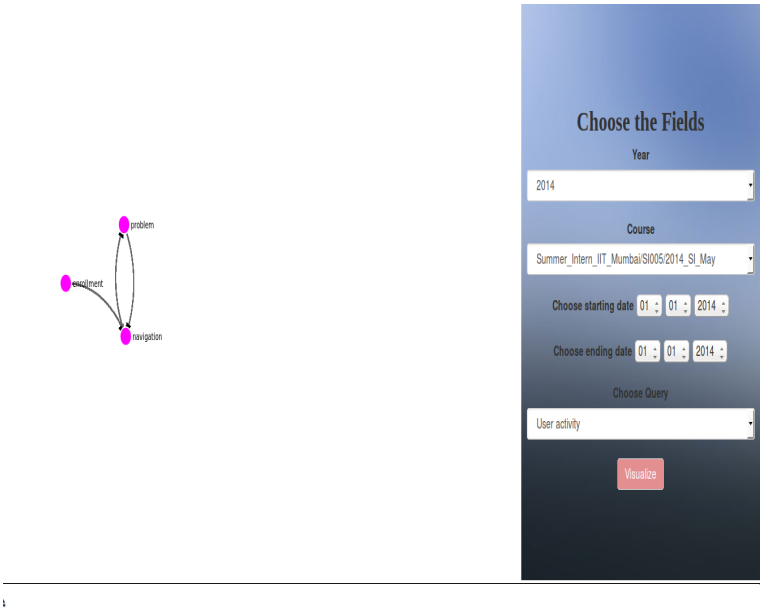


Figure 3.38: Event Sequence

Basic Statistics

Contains a dropdown for course selection

Figure 3.39: Baic Statistics 1

Student name	User id	Emailid
Sukla	12	sukla80@yahoo.co.in
Birundha	11	birundha@cse.iitb.ac.in
birundha22	22	birundha.cs@gmail.com
BhargaviParanjape	30	bhargavi22294@gmail.com
Prakhar	31	prakhar.sethi2@gmail.com
SukalyanBhakat	28	sukalyan.bhakat@gmail.com
gaurav1994	29	gauravmi@iitrpr.ac.in

Figure 3.40: Basic Statistics 2

Problem Analytics

- Response Time By Course

Figure 3.41: Response Time for the students enrolled in a course

Enrollment Analytics

- Students enrolled throughout the country based on their course shown in world map

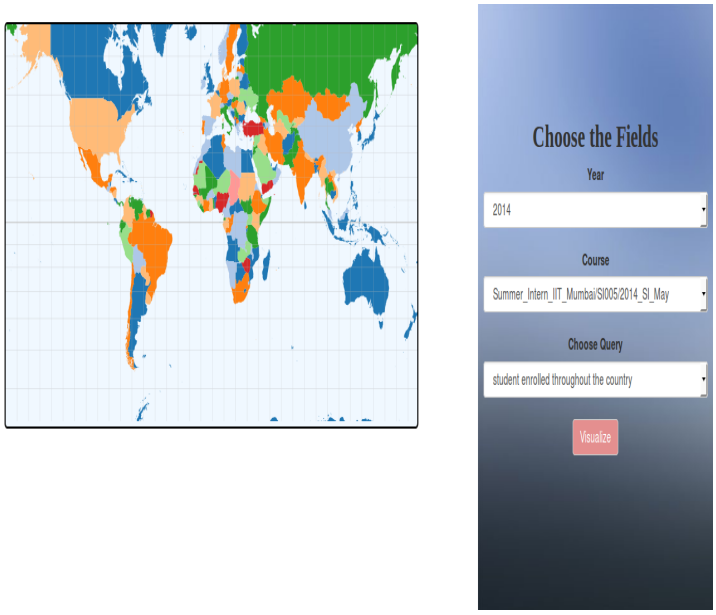


Figure 3.42: Students enrolled from different countries in a particular course

Video Analytics

- Videos watched

Figure 3.43: Videos watched by students in a course

- Showing the users who have used transcript

Figure 3.44: Showing the statistics of transcript usage during video in a course

- Statistics showing the number of students who have jumped the video
- Changes in video speed

Exception Handling [13] If data for a particular query is not found, to avoid display of django default error message exception handling is used.

A script function which uses try except

```
try:
    ans = execute(c,
         "select id,count(distinct(username))
          as count from vi_log_table
          where event_type='speed_change_video'
          and (old_speed < new_speed)
          and course_id='" + course + "' group by id")
      ans.to_csv("/home/hduser/django/testex/
      testex/static/speedincrease.tsv", sep="\t")
   except ValueError:
           return 1
    c.close()
    return 0
```

If return value is 1, then accordingly error message is shown using

```
file1 = "videoviews.html"
        if(flag==1):
         return render(request,"oops.html",{})
        time.sleep(3)
```

Figure 3.45: Page rendered when no data is found for a particular query

3.3.5 The ETL Process

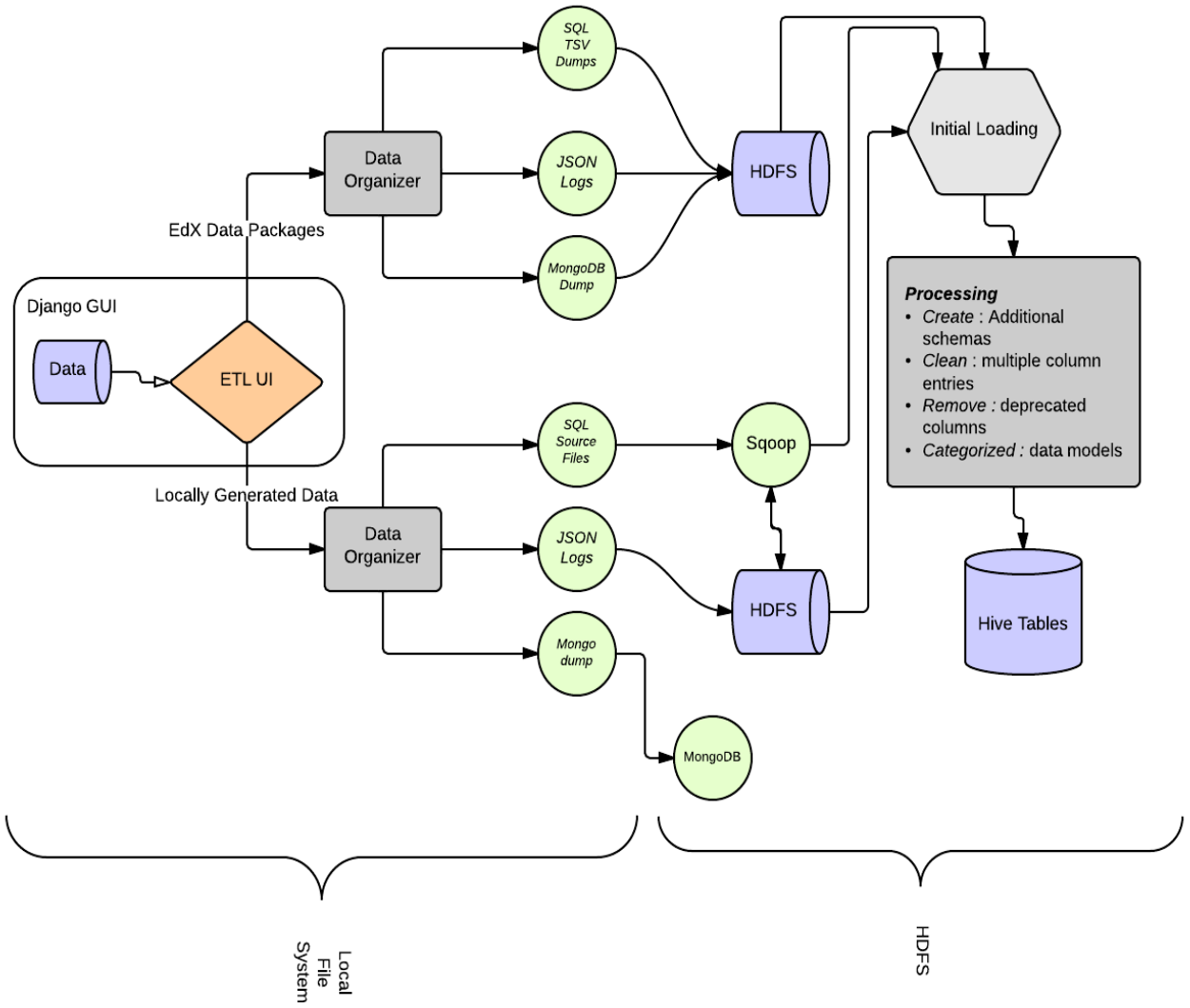


Figure 3.46: ETL Flowchart

3.3.5.1 User Input

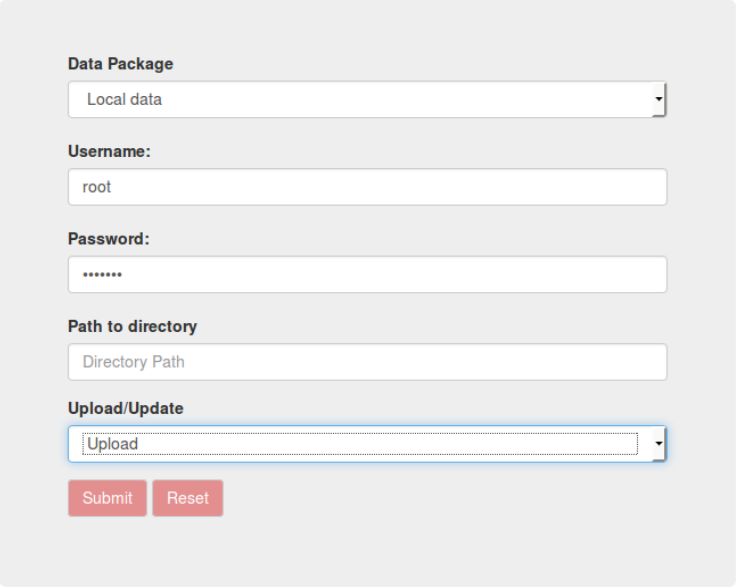


Figure 3.47: Loading Local Data

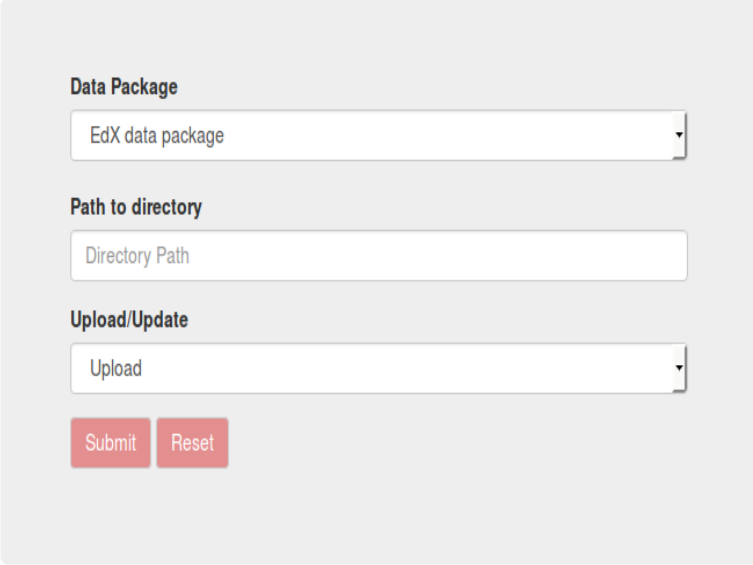


Figure 3.48: Loading EdX Data Packages

The user provides the input of the data source in the ETL GUI as shown above which provides the necessary options and is customized to load data either from EdX Data Packages or Locally generated data.The fields and their detailed description are summarized below which is a direct input to the backen *automater.py* which is controlling the ETL flow.

1. Locally Generated Data
   (a) Data Source : Type of Data whether EdX Data Package or Local Data
   (b) Username : MySQL username(required by Sqoop)
   (c) Passord : MySQL password(required by Sqoop)
   (d) Path to Directory : Path to folder containing files to upload(being a folder is a must)
   (e) Mode :
      i. Upload will upload the data after clearing the HDFS and emptying previously present data
      ii. Update will upload the data after clearing HDFS but without emptying previously present data in columns
      iii. Reset will reset botht the HDFS and clear all present tables without recreating or filling them with any data
2. EdX Data Packages
   (a) Data Source : Type of Data whether EdX Data Package or Local Data
   (b) Path to Directory : Path to folder containing files to upload(being a folder is a must)
   (c) Mode :
      i. Upload will upload the data after clearing the HDFS and emptying previously present data
      ii. Update will upload the data after clearing HDFS but without emptying previously present data in columns
      iii. Reset will reset botht the HDFS and clear all present tables without recreating or filling them with any data

Note :

1. Data should be provided inside a folder whose absolute path has to be mentioned

2. When updating the data no old data should be provided else there can be replication of rows as hive does not impose relatinal structure strictly

3.3.6 Data Organizer

The next block in the ETL Diagram is the Data Organizer which organizes all the files present inside the folder from an unordered and unarranged manner to ordered and arranged subfolders with each subfolder corresponding to specific type of data/input. The different forms of data input have ben summarized under the Data Inputs section above. The Data Organizer works separately for Locally generated and EdX Data Packages.

1. Locally Generated Data

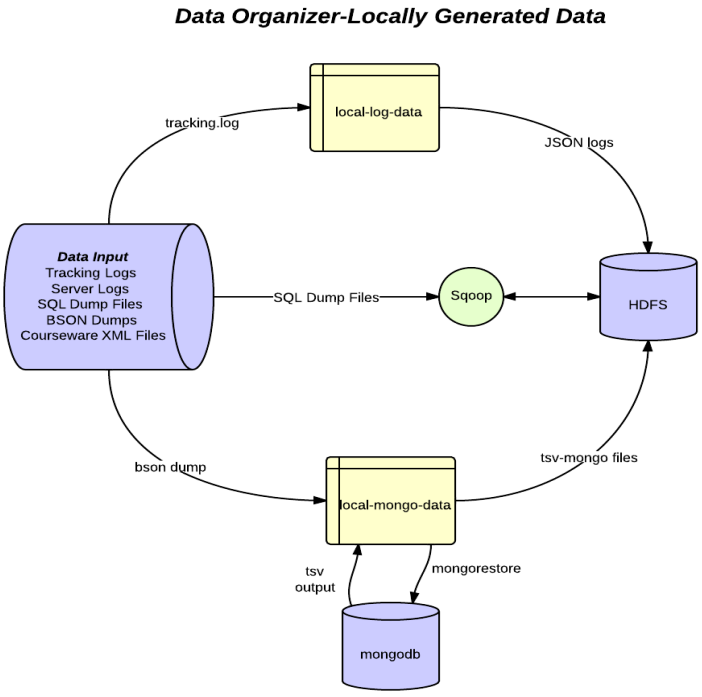


Figure 3.49: Loading Local Data

The block diagram explains how the Data Organizer works for locally generated data.The *automater* script automatically performs a recursive search through the provided data input and separates the following files

(a) Tracking Logs
The tracking logs are first detected by searching for a pattern in the filename. Once correctly segregated in specific subfolders it is directly loaded onto the HDFS

(b) SQL Dump files
The SQL Dump files are first ported to the MySQL Server running on the local system. After this a search and load technique is applied where specific tables are first searched and then loaded using Sqoop which automatically transfers the files into HDFS and loads it into Hive Tables

(c) BSON Dumps
The BSON dumps(only modulestore.bson) are first ported to a specific subfolder which is then loaded onto MongoDB using *mongorestore* queried appropriately and finally provide tsv output corresponding to specific tables which are then ported onto HDFS.
The complete procedure for ETL of BSON Files has been summarized by the flowchart below

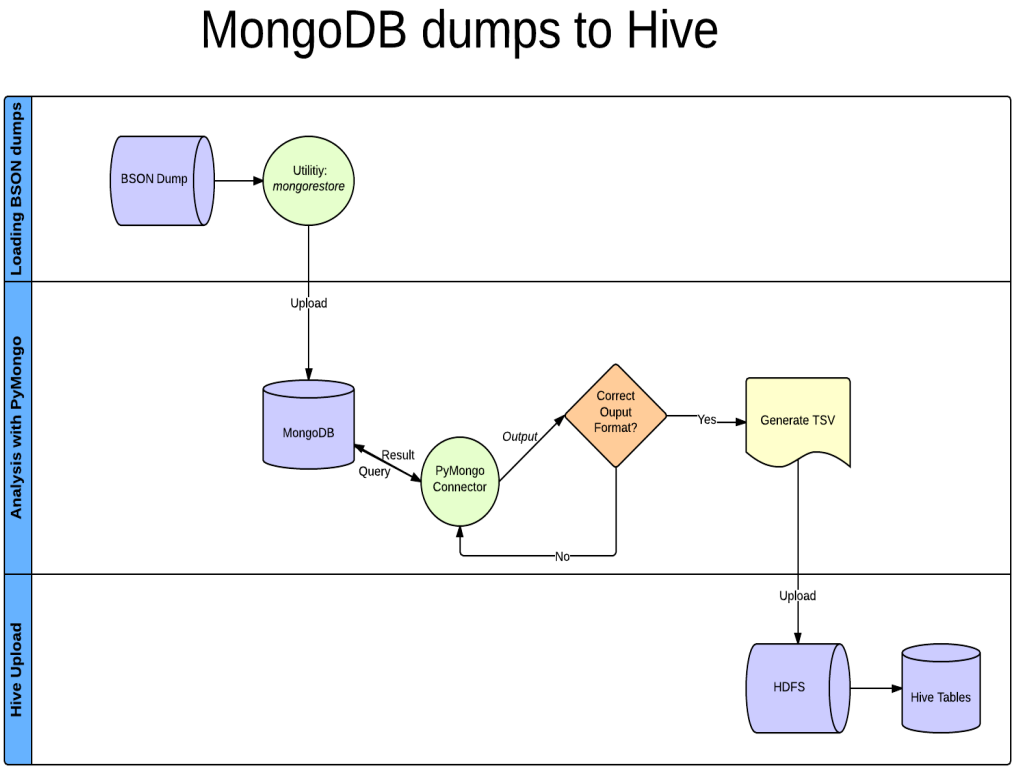


Figure 3.50: MongoDB Data Loading

2. EdX Data Packages

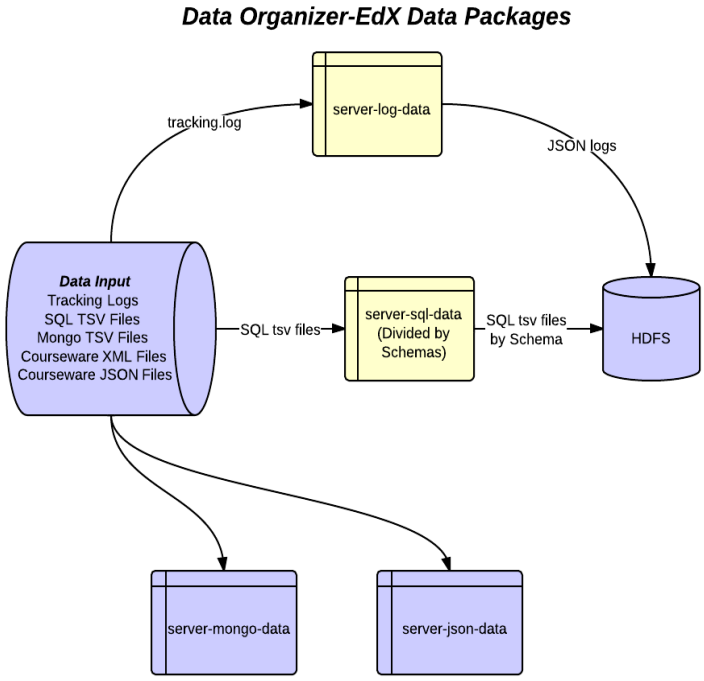


Figure 3.51: Loading EdX Data Packages

The block diagram explains how the Data Organizer works for data from downloaded EdX Data Packages.The *automater* script automatically performs a recursive search through the provided data input and separates the following files

(a) Tracking Logs
The tracking logs are first detected by searching for a pattern in the filename. Once correctly segregated in specific subfolders it is directly loaded onto the HDFS

(b) SQL TSV files
The SQL TSV files are first detected by searching for a pattern in the filename.Specific filenaming conventions are provided in the EdX documentation corresponding to individual schemas for every table and for different types of data.Once correctly segregated in specific subfolders it is directly loaded onto the HDFS

(c) Mongo Files
The Mongo files are similarly detected by searching recursively

and ported to specific subfolders. However since the data present in it is empty it is not loaded onto HDFS/Hive and further functionalities can be added later

3.3.7 Data Loading and Transformation

Finally the data has to be processed and loaded into Hive tables so that it is ready for Data Analysis.The data loading can again be split for the log files and others.

(a) SQL/Mongo Data
The SQL(optionally Mongo) Data is loaded by Sqoop directly in case of local sql dump files and loaded from TSV into Hive for EdX Data Packages using the `load data inpath` statement. Optionally Improved Data models can be provided with removal of all deprecated fields so that the Data Warehouse is ready to use for Data Analysis

(b) Log Data
Log Data follows a similar format for EdX Data Packages and locally generated data.However being one of the most important forms of data it has to be loaded and further segregated internally in Hive while retrieving its most important columns for data analysis. However the biggest problem faced in this approach was the sheer number of different types of logs which can be generated. This fact is neatly summarized in the tree diagram below

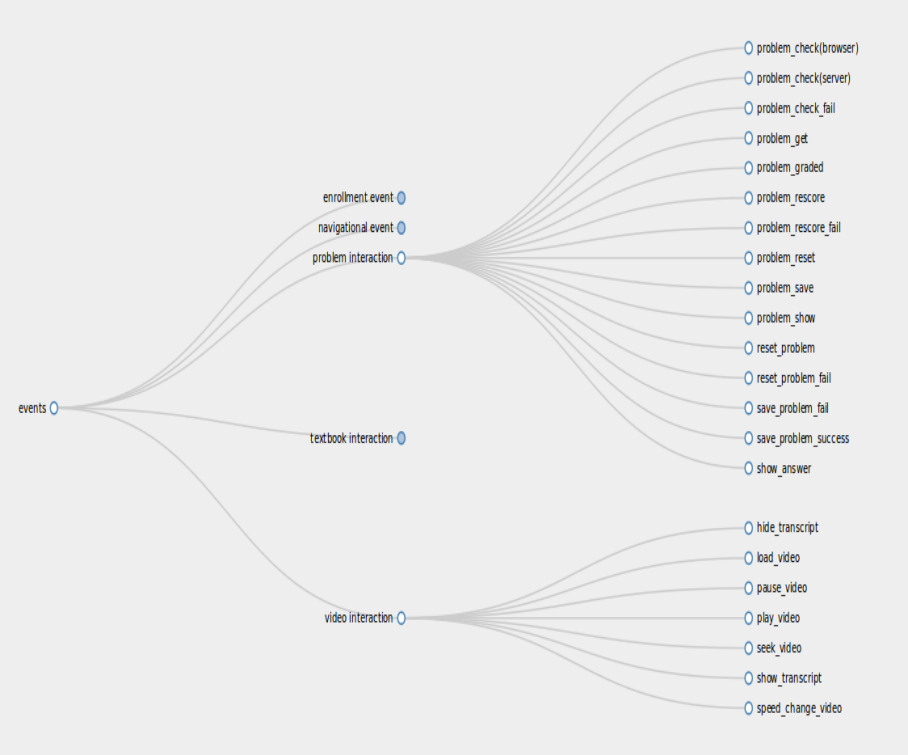


Figure 3.52: Log Distribution Tree

As can be seen in the tree distribution above there are a large number of different types of logs that can be generated. Additionally different types of logs have different entries which vary widely between different event types. In fact appending all possible columns results in greater than 50 columns where depending on the type of log would result in a number of null columns.Thus the best solution to this problem was first dividing tables into respective event types.
The method of dividing the logs by category has been demonstrated below in the form of a simple mind-map diagram

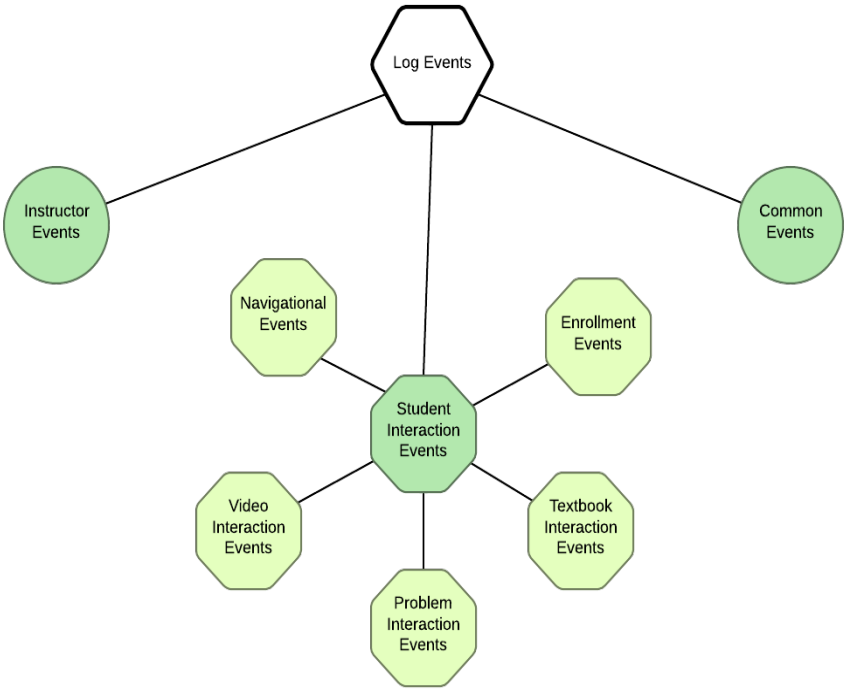


Figure 3.53: Event Type Distribution

Thus after analysing the logs and going through entries of different event types the tables holding log entries have been divided and used as follows

i. *log_table*
Contains all the information with internal metadata remaining unparsed. It is important for sequence related or any such information utilising complete logs

ii. *rem_log_table*
Table for logs related to common fields which are not categorized under any event

iii. *vi_log_table*
Table for video related logs to obtain information about videos watched by users

iv. *pi_log_table*
Table for problem related logs which is useful for determining marks,response times and essential for most EDM inferences

v. *ee_log_table*
Table containing Enrollment related logs to find when users have enrolle,dropped out,etc

vi. *ne_log_table*
Navigation Events which provide information related to when users have navigated to and away from pages

The procedure used to divide and further process log entries to categorized tables is intuitive from the *wireframe* images depicting different log entries, how they were extracted by json_tuple and finally cleaned in hive internally.

The wireframes above first show an example of a log related to a specific event type. It then shows how after the logs have been parsed by *json_tuple* it is further cleaned applying multiple hive string functions such as *concat_ws,if,case-when,split,regexp_extract*

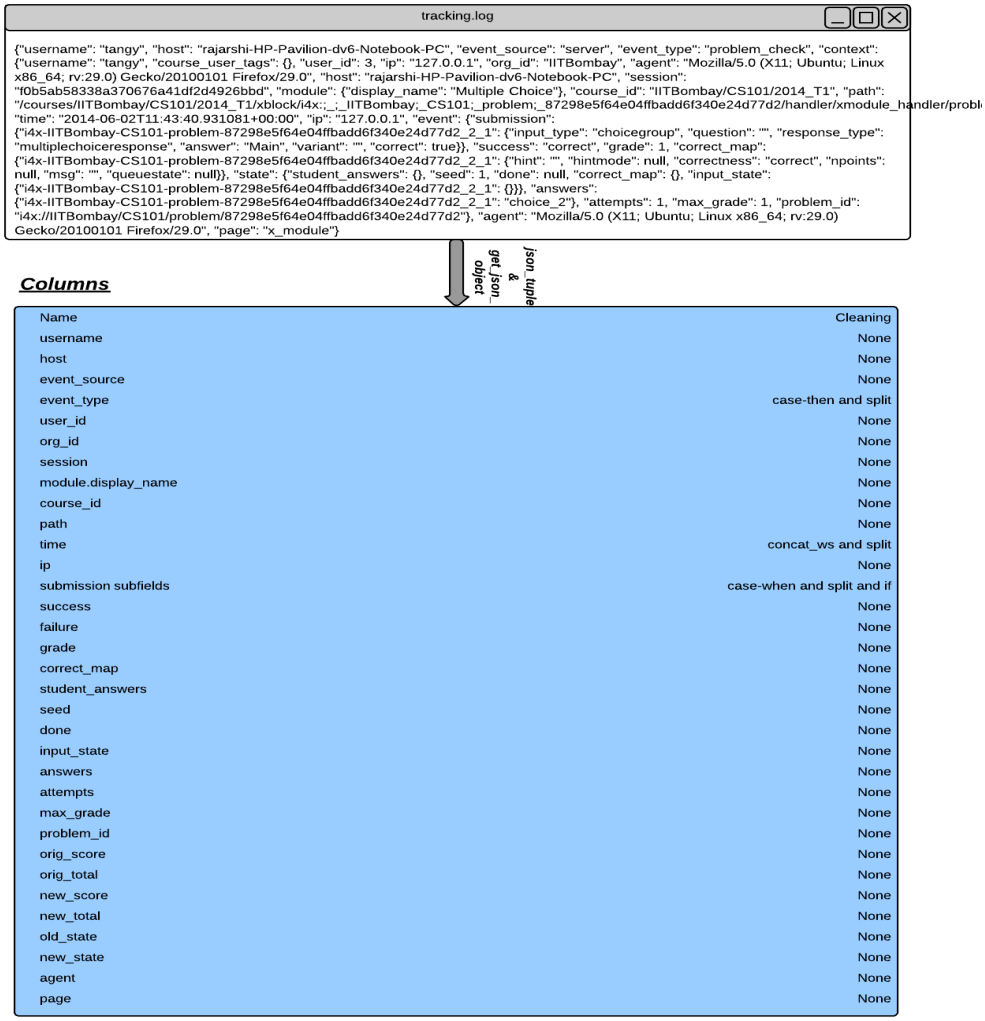


Figure 3.54: Problem Interaction Log Table

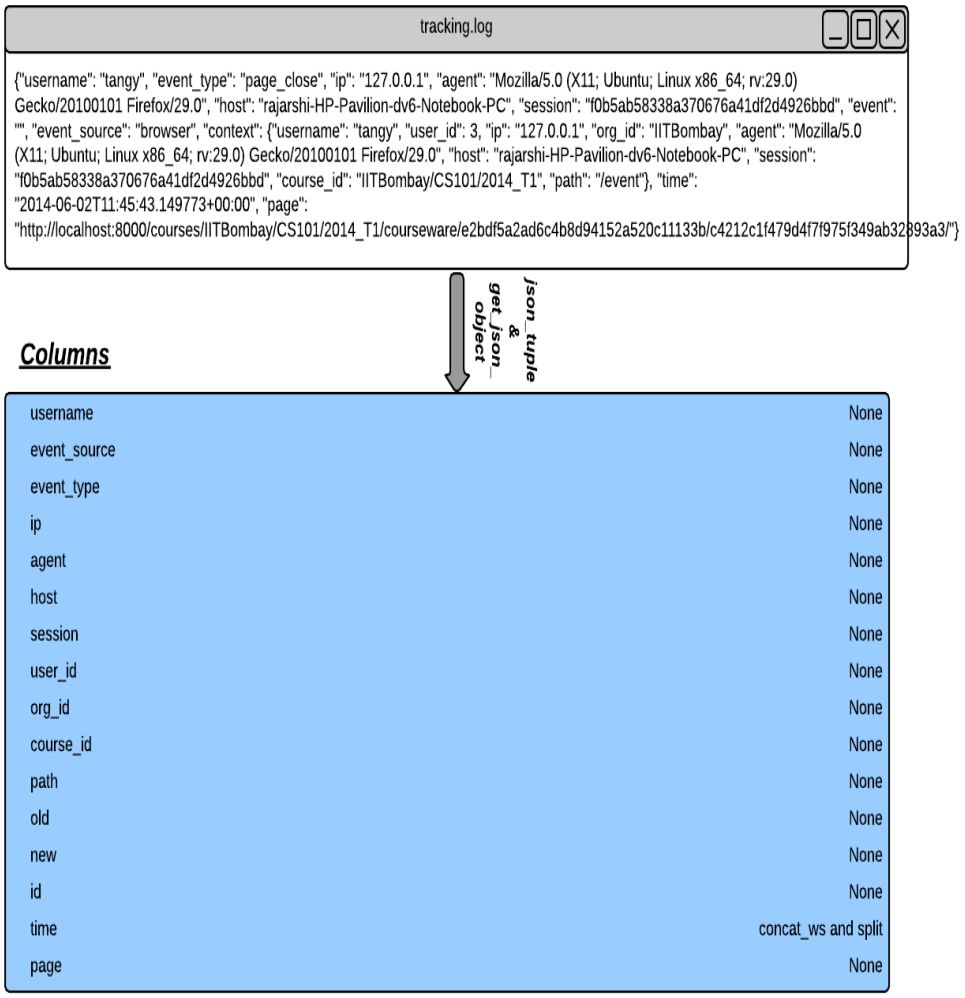


Figure 3.57: Navigational Events Log Table

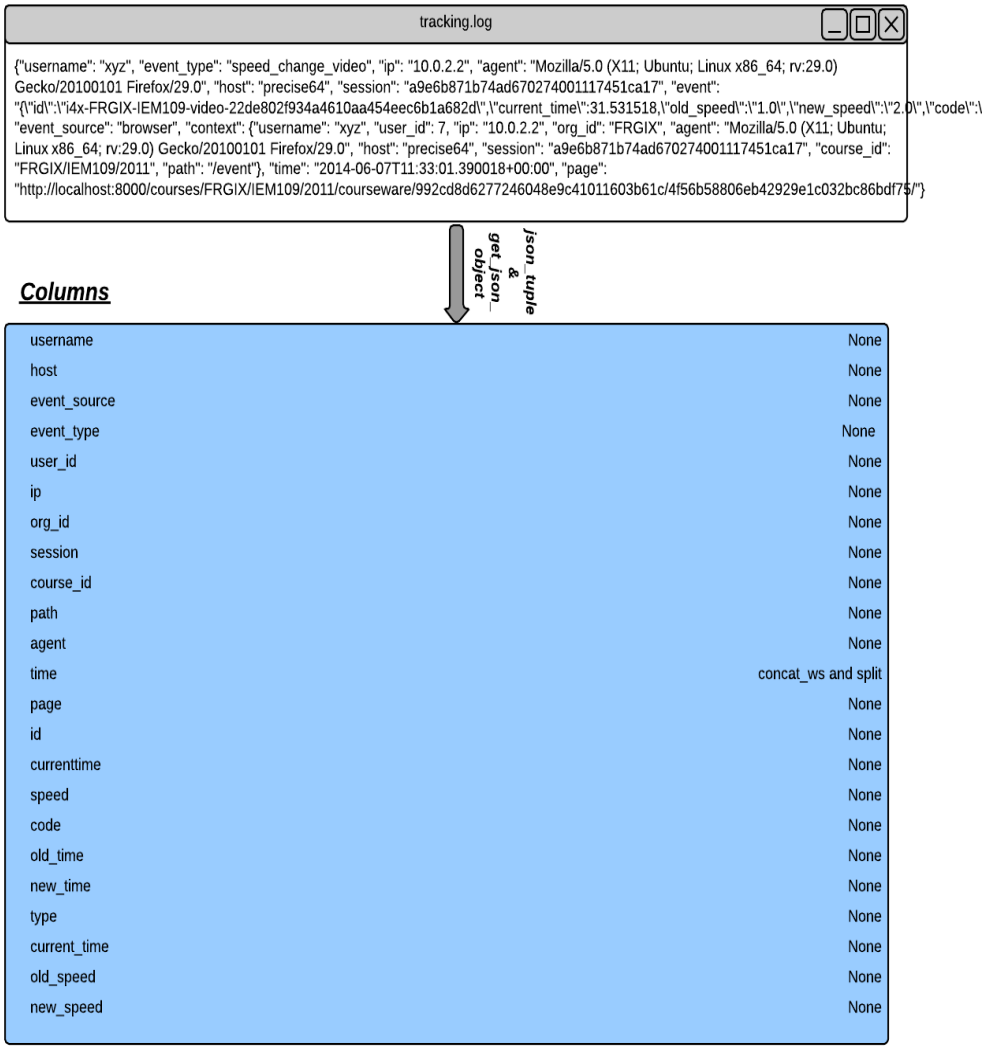


Figure 3.55: Video Interaction Log Table

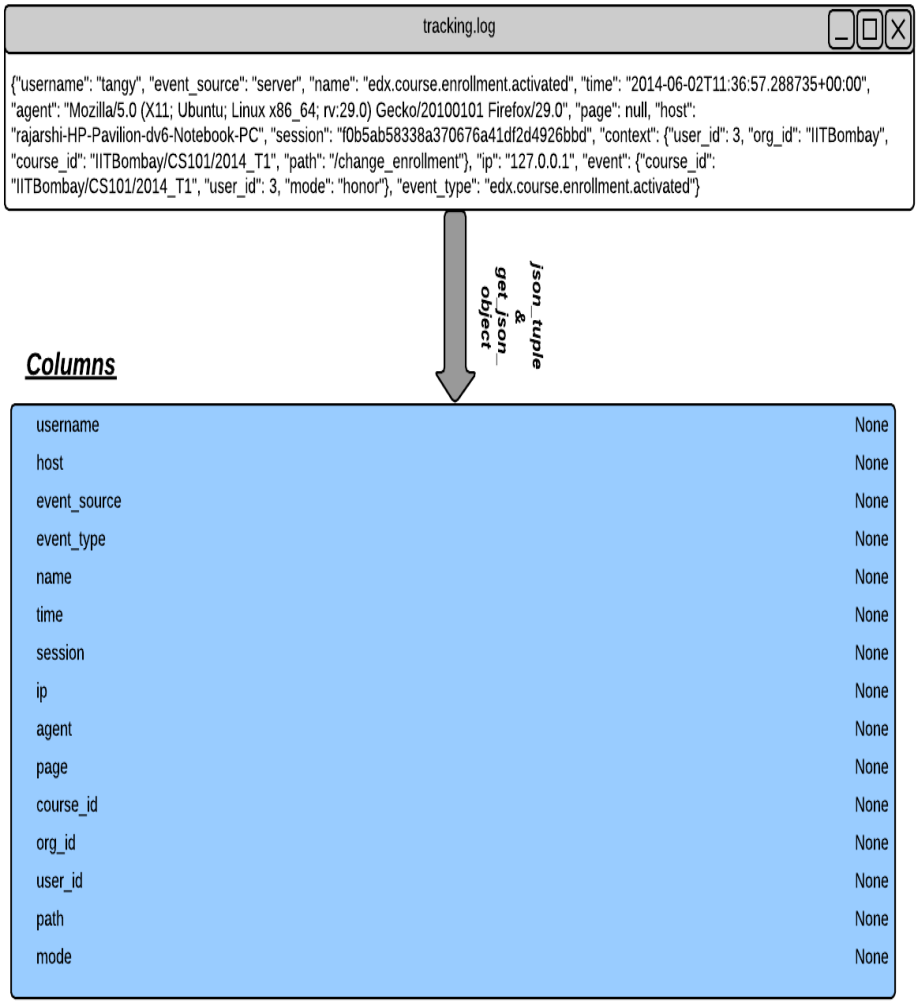


Figure 3.56: Enrollment Events Log Table

Chapter 4

Optimization

4.1 Measuring Performance using Ganglia

4.1.1 What is Ganglia?

On the High Performance Computing systems (HPCs), a tool that can monitor [30] and show real-time usage of resources is very useful. Such a tool can indicate to the potential users, the current state of resource availability so that necessary planning for job execution can be made. It also helps to analyse the running jobs in terms of how they are using the resources allocated. It is necessary that the data be presented in intuitive and visually understandable way so that quick inferences can be made. Ganglia is one such open source software that provides such monitoring with a visual interface and can be easily integrated into a variety of HPCs. Ganglia is an open-source, scalable and distributed monitoring system for large clusters. It collects, aggregates and provides time-series views of tens of machine-related metrics such as CPU, memory, storage, network usage. Ganglia is also a popular solution for monitoring Hadoop and HBase clusters, since Hadoop (and HBase) has built-in support for publishing its metrics to Ganglia. With Ganglia you may easily see the number of bytes written by a particular HDFS datanode over time, the block cache hit ratio for a given HBase region server, the total number of requests to the HBase cluster, time spent in garbage collection and many, many others

4.1.2 Major Components to Ganglia

Ganglia consists of three components:

- **Ganglia monitoring daemon (gmond)-**a daemon which needs to run on every single node that is monitored. It collects local monitoring metrics and announce them, and (if configured) receives and aggregates metrics sent to it from othergmond s (and even from itself)

  sudo apt-get install ganglia-monitor

  Configuration in /etc/gmond.conf, and associated service is `ganglia-monitor`.

- **Ganglia meta daemon (gmetad)** a daemon that polls from one or more data sources (a data source can be agmond or othergmetad) periodically to receive and aggregate the current metrics. The aggregated results are stored in database and can be exported as XML to other clients for example, the web frontend.

  sudo apt-get install ganglia-webfrontend

  Configuration in /etc/gmetad.conf, and associated service is `gmetad`.

- **Ganglia PHP web frontend** It retrieves the combined metrics from the meta daemon and displays them in form of nice, dynamic HTML pages containing various real-time graphs.

4.1.3 Configuration Of Ganglia 3.1.7 [14]

1. **Ganglia for a small Hadoop cluster** While Ganglia is scalable, distributed and can monitor hundreds and even thousands of nodes, small clusters can also benefit from it . We have configured Ganglia on a five-node cluster (1 masters and 4 slaves) that runs Hadoop and HBase.Our monitoring requirements can be specified as follows:

   - easily get metrics from every single node

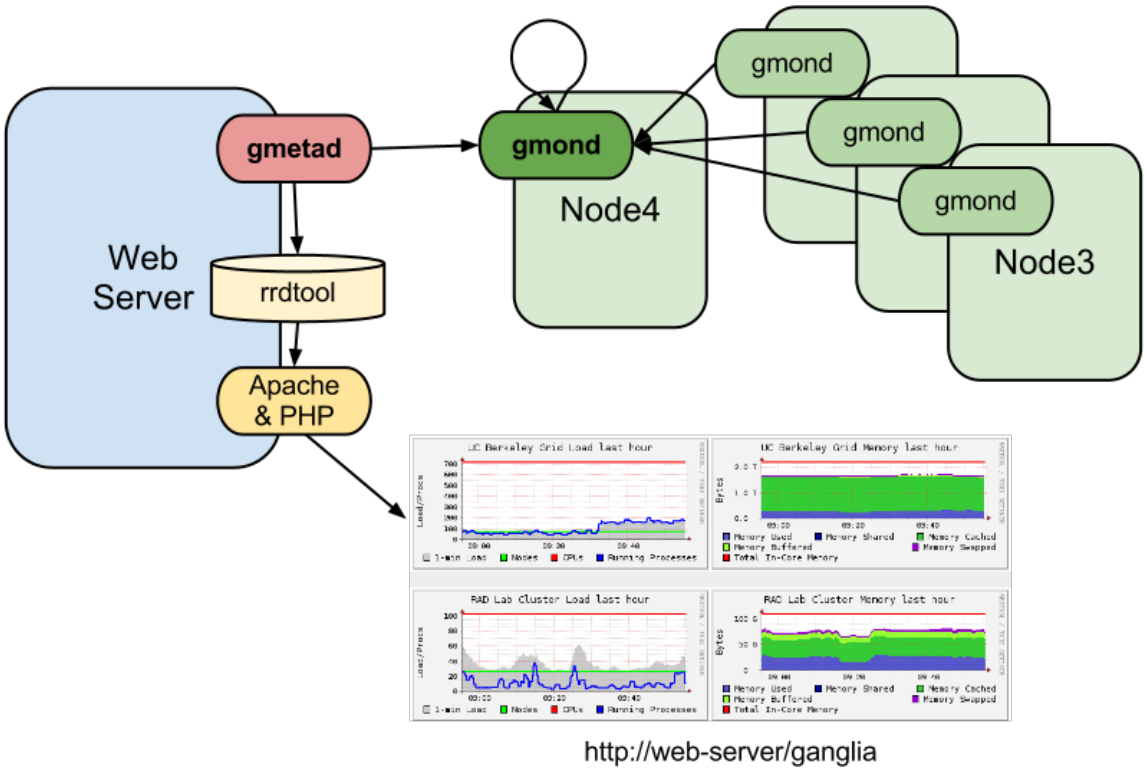


Figure 4.1: Major Components of Ganglia

- easily get agregated metrics for all slave nodes (so that we will know how much resources is used by MapReduce jobs and HBase operations)

- easily get agregated metrics for all master nodes (so far we have only one master, but when the cluster grows, we will move some master deamons (e.g JobTracker, Secondary Namenode) to separate machines)

- easily get agregated metrics for all nodes (to get overall state of the cluster)

We wanted Ganglia to see the cluster as two logical subclusters e.g. masters and slaves which looks like-

Figure 4.2: Master and Slave subclusters

2. **Possible Ganglias configuration** Here is an illustrative picture which shows simple Ganglias configuration for 5-node Hadoop cluster that meets our all requirements may look like. So lets configure it in this way!

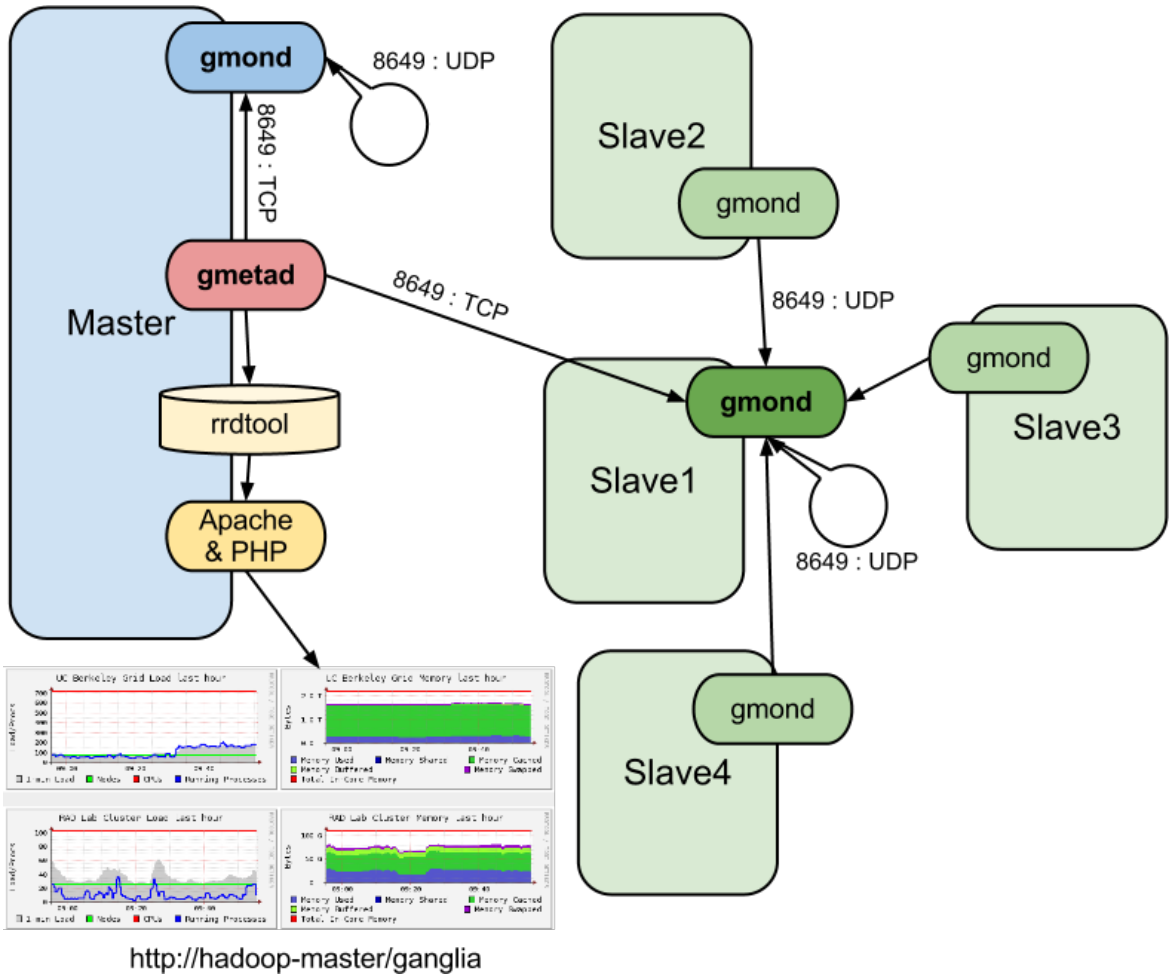


Figure 4.3: Possible Ganglia's configuration

Please note, that we would like to keep as many default settings as possible. By default:

- gmond communicates on UDP port 8649
- gmetad downloads metrics on TCP port 8649
- However, one setting will be changed. We set the communication method between gmonds to be unicast UDP messages (instead of multicast UDP messages). Unicast has following advantages over multicast: it is better for a larger cluster (say a cluster with more than a hundred of nodes) and it is supported in the Amazon EC2 environment (unlike multicast).

3. **Ganglia monitoring daemon (gmond) configuration** According to the figure above: Every node runs a gmond.

   - **Slaves subcluster configuration**
     - Each gmond on slave1, slave2, slave3 and slave4 nodes defines udp_send_channel to send metrics to slave1 (port 8649)
     - gmond on slave1 defines udp_recv_channel (port 8649) to listen to incoming metrics and tcp_accept_channel (port 8649) to announce them. This means this gmond is the lead-node for this subcluster and collects all metrics sent via UDP (port 8649) by all gmonds from slave nodes (even from itself), which can be polled later via TCP (port 8649) by gmetad.
   - **Masters subcluster configuration**
     - gmond on master node defines udp_send_channel to send data to master (port 8649),udp_recv_channel (port 8649) and tcp_accept_channel (port 8649). This means it becomes the lead node for this one-node cluster and collects all metrics from itself and exposes them to gmetad. The configuration should be specified in gmond.conf file (you may find it in /etc/ganglia/). gmond.conf for slave1 (only the most important settings included):

```
cluster {
  name = 'hadoop-slaves'
  ...
}
udp_send_channel {
  host = slave1.node.IP.address
  port = 8649
}
udp_recv_channel {
  port = 8649
}
tcp_accept_channel {
  port = 8649
}
```

-gmond.conf for slave2, slave3, slave4 (actually, the same gmond.conf file as for slave1 can be used as well):

```
cluster {
  name = 'hadoop-slaves'
  ...
}
udp_send_channel {
  host = slave1.node.IP.address
  port = 8649
}
udp_recv_channel {}
tcp_accept_channel {}
```

The gmond.conf file for the master node should be similar to slave1 gmond.conf file just replace slave1 IP address with masters IP and set cluster name to hadoop-masters.

4. **Ganglia meta daemon (gmetad)** gmetad configuration is even simpler:

   - Master runs gmetad.
   - gmetad defines two data sources:

     ```
     data_source 'hadoop-masters' master.node.IP.address
     data_source 'hadoop-slaves' slave1.node.IP.address
     ```

     The configuration should be specified in gmetad.conf file (you may find it in /etc/ganglia/).

5. **Hadoop and HBase integration with Ganglia** Hadoop and HBase use GangliaContext class to send the metrics collected by each daemon (such as datanode, tasktracker, jobtracker, HMaster etc) to gmonds. Once you have setup Ganglia successfully, you may want to edit /etc/hadoop/conf/hadoop-metrics.properties and /etc/hbase/conf/hadoop-metrics.properties to announce Hadoop and HBase-related metric to Ganglia.

   - **Metrics configuration for slaves**

     ```
     # /etc/hadoop/conf/hadoop-metrics.properties
     ...
     ```

```
dfs.class=org.apache.hadoop.metrics.ganglia.GangliaContext31
dfs.period=10
dfs.servers=hadoop-slave1.IP.address:8649
...
mapred.class=org.apache.hadoop.metrics.
ganglia.GangliaContext31
mapred.period=10
mapred.servers=hadoop-slave1.IP.address:8649
...
```

- **Metrics configuration for master** Should be the same as for slaves just use hadoop-master.IP.address:8649 (instead of hadoop slave1.IP.address:8649) for example:

  ```
  # /etc/hbase/conf/hadoop-metrics.properties
  ...
  hbase.class=org.apache.hadoop.metrics.
  ganglia.GangliaContext31
  hbase.period=10
  hbase.servers=hadoop-master.IP.address:8649
  ...
  ```

  Remember to edit both properties files (/etc/hadoop/conf/hadoop-metrics.properties for Hadoop and /etc/hbase/conf/hadoop-metrics.properties for HBase) on all nodes and then restart Hadoop and HBase clusters. No further configuration is necessary.

4.2 Improving Performance Using Apache Spark

4.2.1 What is Apache Spark? [15]

Apache Spark is an open source cluster computing system that aims to make data analytics fast both fast to run and fast to write.To run programs faster, Spark offers a general execution model that can optimize arbitrary operator graphs, and supports in-memory computing, which lets it query data faster than disk-based engines like Hadoop.To make programming faster, Spark provides clean, concise APIs in **Scala**,

Java and **Python**. You can also use Spark interactively from the Scala and Python shells to rapidly query big datasets.

4.2.2 What can it do?

Spark was initially developed for two applications where placing data in memory helps: iterative algorithms, which are common in machine learning, and interactive data mining. In both cases, Spark can run up to 100x faster than Hadoop MapReduce. However, you can use Spark for general data processing too. Spark is also the engine behind **Shark**, a fully Apache Hive-compatible data warehousing system that can run 100x faster than Hive.While Spark is a new engine, it can access any data source supported by Hadoop, making it easy to run over existing data

4.2.3 Who uses it?

Spark was initially created in the UC Berkeley AMPLab, but is now being used and developed at a wide array of companies.Spark is open source under an Apache license.

4.2.4 Efficiencies and how Spark Compares to Hadoop

Unlike Hadoop, Spark is not strictly a Map/Reduce framework; it is not limited to alternating Map and Reduce operations with an implicit group-by in between. For example, in Spark you can group on one key for one aggregation and then group by another key for a subsequent aggregation, without requiring a Map step in between. You can also do things like sampling or mapPartitions (i.e. apply a function to each split).

All sorts of algorithms can be written in a Map/Reduce style so this isnt an advantage of functionality. It is a performance advantage. The extra Map step, for example, would require serialization, deserialization, disk IO, etc. By keeping data in memory, you avoid another ser/deser, IO

and the overhead of spinning up the task instances. These distinctions allow a more efficient use of resources and are an important step forward for a certain class of Big Data problems.

Naturally, youre limited to working with datasets that mostly fit in memory . Even with that, its reasonable to have hundred of GBs (or event TBs) across your cluster, so this still allows for some interesting Big Data work. Also note that it doesnt replicate data in memory (so you dont need 3X storage as with HDFS), instead it save lineage for your splits which can be used to reconstruct lost data partitions.We would continue to use Hadoop for a lot of the heavy lifting (and more static work flows), and would initially consider Spark for ad-hoc queries and post-processing on results.

4.2.5 Spark Examples

These examples give a quick overview of the Spark API. Spark is built on the concept of distributed datasets, which contain arbitrary Java or Python objects. You create a dataset from external data, then apply parallel operations to it. There are two types of operations: **transformations**, which define a new dataset based on previous ones, and **actions**, which kick off a job to execute on a cluster.

1. **Text Search** In this example, we search through the error messages in a log file:

   - **Python**

     ```
     file = spark.textFile("hdfs://...")
     errors = file.filter(lambda line: "ERROR" in line)
     # Count all the errors
     errors.count()
     # Count errors mentioning MySQL
     errors.filter(lambda line: "MySQL" in line).count()
     # Fetch the MySQL errors as an array of strings
     errors.filter(lambda line: "MySQL" in line).collect()
     ```

   - **Scala**

```
val file = spark.textFile("hdfs://...")
val errors = file.filter(line => line.contains("ERROR"))
// Count all the
errors
errors.count()
// Count errors mentioning MySQL
errors.filter(line => line.contains("MySQL")).count()
// Fetch the MySQL errors as an array of strings
errors.filter(line => line.contains("MySQL")).collect()
```

- **Java**

```
JavaRDD<String> file = spark.textFile("hdfs://...");
JavaRDD<String> errors = file.filter
(new Function<String, Boolean>() {
  public Boolean call(String s)
  { return s.contains("ERROR"); }
});
// Count all the errors
errors.count();
// Count errors mentioning MySQL
errors.filter(new Function<String, Boolean>() {
  public Boolean call(String s)
  { return s.contains("MySQL"); }
}).count();
// Fetch the MySQL errors as an array of strings
errors.filter(new Function<String, Boolean>() {
  public Boolean call(String s)
  { return s.contains("MySQL"); }
}).collect();
```

2. **In-Memory Text Search** Spark can cache datasets in memory to speed up reuse. In the example above, we can load just the error messages in RAM using:

   - **Python,Scala,Java**

```
errors.cache();
```

3. **Word Count** In this example, we use a few more transformations

to build a dataset of (String, Int) pairs called counts and then save it to a file.

- **Python**

```
file = spark.textFile("hdfs://...")
counts = file.flatMap(lambda line: line.split(" ")) \
         .map(lambda word: (word, 1)) \
         .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

- **Scala**

```
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
             .map(word => (word, 1))
             .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

- **Java**

```
JavaRDD<String> file = spark.textFile("hdfs://...");
JavaRDD<String> words =
file.flatMap(new FlatMapFunction<String, String>() {
  public Iterable<String> call(String s)
  { return Arrays.asList(s.split(" ")); }
});
JavaPairRDD<String, Integer> pairs =
words.map(new PairFunction<String, String, Integer>() {
  public Tuple2<String, Integer> call(String s)
  { return new Tuple2<String, Integer>(s, 1); }
});
JavaPairRDD<String, Integer> counts =
pairs.reduceByKey(new Function2<Integer, Integer>() {
  public Integer call(Integer a, Integer b) { return a + b; }
});
counts.saveAsTextFile("hdfs://...");
```

4. **Estimating Pi** Spark can also be used for compute-intensive tasks. This code estimates by "throwing darts" at a circle. We pick

random points in the unit square ((0, 0) to (1,1)) and see how many fall in the unit circle. The fraction should be / 4, so we use this to get our estimate

- **Python**

```
def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

count = spark.parallelize
(xrange(0, NUM_SAMPLES)).map(sample) \
             .reduce(lambda a, b: a + b)
print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

- **Scala**

```
val count = spark.parallelize(1 to NUM_SAMPLES).map{i =>
  val x = Math.random()
  val y = Math.random()
  if (x*x + y*y < 1) 1 else 0
}.reduce(_ + _)
println("Pi is roughly " + 4.0 * count / NUM_SAMPLES)
```

- **Java**

```
int count = spark.parallelize(makeRange(1, NUM_SAMPLES)).
filter(new Function<Integer, Boolean>() {
  public Integer call(Integer i) {
    double x = Math.random();
    double y = Math.random();
    return x*x + y*y < 1;
  }
}).count();
System.out.println("Pi is roughly " + 4 * count / NUM_SAMPLES)
```

5. **Logistic Regression** This is an iterative machine learning algorithm that seeks to find the best hyperplane that separates two sets of points in a multi-dimensional feature space. It can be used to classify messages into spam vs non-spam, for example. Because the

algorithm applies the same MapReduce operation repeatedly to the same dataset, it benefits greatly from caching the input in RAM across iterations.

- **Python**

```
points = spark.textFile(...).map(parsePoint).cache()
w = numpy.random.ranf(size = D) # current separating plane
for i in range(ITERATIONS):
    gradient = points.map(
        lambda p: (1 / (1 + exp(-p.y*(w.dot(p.x)))) - 1) * p.y *
    ).reduce(lambda a, b: a + b)
    w -= gradient
print "Final separating plane: %s" % w
```

- **Scala**

```
val points = spark.textFile(...).map(parsePoint).cache()
var w = Vector.random(D) // current separating plane
for (i <- 1 to ITERATIONS) {
  val gradient = points.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= gradient
}
println("Final separating plane: " + w)
```

- **Java**

```
class ComputeGradient extends Function<DataPoint, Vector> {
  private Vector w;
  ComputeGradient(Vector w) { this.w = w; }
  public Vector call(DataPoint p) {
    return p.x.times(p.y * (1 / (1 + Math.exp(w.dot(p.x))) - 1)
  }
}

JavaRDD<DataPoint> points =
spark.textFile(...).map(new ParsePoint()).cache();
```

```
Vector w = Vector.random(D); // current separating plane
for (int i = 0; i < ITERATIONS; i++) {
  Vector gradient =
  points.map(new ComputeGradient(w)).reduce(new AddVectors());
  w = w.subtract(gradient);
}
System.out.println("Final separating plane: " + w);
```

The graph below compares the running time per iteration of this Spark program against a Hadoop implementation on 100 GB of data on a 100-node cluster, showing the benefit of in-memory caching:

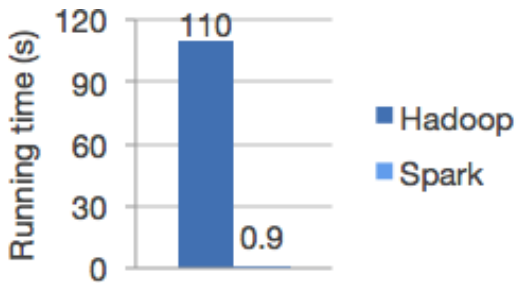


Figure 4.4: Spark Performance Comparision With Hadoop

4.3 Improving Performance Using Shark

4.3.1 What is Shark? [16]

Shark is a large-scale data warehouse system for Spark designed to be compatible with Apache Hive. It can execute Hive QL queries up to 100 times faster than Hive without any modification to the existing data or queries. Shark supports Hive's query language, metastore, serialization formats, and user-defined functions, providing seamless integration with existing Hive deployments and a familiar, more powerful option for new ones.

4.3.2 Shark Highlights

1. **Fast Execution Engine** Shark is built on top of Spark, a data-parallel execution engine that is fast and fault-tolerant. Even if data are on disk, Shark can be noticeably faster than Hive because of the fast execution engine. It avoids the high task launching overhead of Hadoop MapReduce and does not require materializing intermediate data between stages on disk. Thanks to this fast engine, Shark can answer queries in sub-second latency

2. **Columnar Memory Store** Analytical queries usually focus on a particular subset or time window, e.g., http logs from the previous month, touching only the (small) dimension tables and a small portion of the fact table. These queries exhibit strong temporal locality, and in many cases, it is plausible to t the working set into a clusters memory.

   Shark allows users to exploit this temporal locality by storing their working set of data across a cluster's memory, or in database terms, to create in-memory materialized views. Common data types can be cached in a columnar format (as Java primitives arrays), which is very efficient for storage and garbage collection, yet provides maximum performance (orders of magnitude faster than reading data from disk). Below is an example on how to cache data in Shark:

```
CREATE TABLE logs_last_month_cached
AS SELECT *
FROM logs WHERE time > date(...);

SELECT page, count(*) c
FROM logs_last_month_cached
GROUP BY page
ORDER BY c DESC LIMIT 10;
```

3. **Spark/Machine Learning Integration** Shark provides a simple API for programmers to convert results from SQL queries into a special type of RDDs (Resilient Distributed Datasets). This integrates SQL query processing with machine learning, and provides a unified system for data analysis using both SQL and sophisticated statistical learning functions.

```
val youngUsers = sql2rdd("SELECT * FROM
  users WHERE age < 20")
println(youngUsers.count)
val featureMatrix = youngUsers.map(extractFeatures(_))
kmeans(featureMatrix)
```

4. **The Shark Query Language** Shark supports a subset of SQL nearly identical to that implemented by Hive. This guide assumes you have some familiarity with Hive, and focuses on the extra functionality included in Shark

   Unlike Hive, Shark allows users to exploit this temporal locality by caching their working set of data, or in database terms, to create in-memory materialized views. Common data types can be cached in a columnar format (as Java primitives arrays), which is very efficient for storage and garbage collection, yet provides maximum performance (orders of magnitude faster than reading data from disk)

   To create a cached table from the rows (or subset of rows) of an existing table, set the shark.cache table property:

```
CREATE TABLE ... TBLPROPERTIES
("shark.cache" = "true") AS SELECT ...
```

We also extend HiveQL to provide a shortcut for this syntax. Simply append _cached to the table name when using CREATE TABLE AS SELECT, and that table will be cached in memory. To disable this shortcut, see the configuration options section. Below is an example:

```
CREATE TABLE logs_last_month_cached AS
SELECT * FROM logs WHERE time > date(...);
```

Once this table has been created, we can query it like any other Hive table.

```
SELECT count(*) from logs_last_month_cached;
```

Note that queries which shuffle data require you to set the number of reducers:

```
set mapred.reduce.tasks=[num_tasks];
SELECT page, count(*) c
FROM logs_last_month_cached
GROUP BY page
ORDER BY c DESC LIMIT 10;
```

In addition to caching, Shark employs a number of optimization techniques such as limit push downs and hash-based shuffle, which can provide significant speedups in query processing. You can directly inspect the execution plan for a given query using the explain statement:

```
explain SELECT *
FROM logs_last_month
ORDER BY timestamp LIMIT 10;
```

5. **Executing Queries**

   - **Shark CLI** The easiest way to run Shark is to start a Shark Command Line Client (CLI) and being executing queries. The Shark CLI connects directly to the Hive Metastore, so it is compatible with existing Hive deployments. Shark executables are available in the bin/ directory. To start the Shark CLI, simply run:

```
$ ./bin/shark  # Start CLI for interactive session
$ ./bin/shark -e "SELECT * FROM foo"
$ ./bin/shark -i queries.hql # Run queries from a file
$ ./bin/shark -H # Start CLI and print hel
```

You can enter queries into the CLI directly, or use a flag to pass it a file. The Shark CLI will only work correctly if the HIVE_HOME environment variable is set (see Configuration). Alternative versions of the CLI exist which print out more information: bin/shark-withinfo and bin/shark-withdebug.

- **Table Persistence** Cached tables are ephemeral they will fall out of cache once a user's session is terminated. To create long-lived cached tables, use the SharkServer described below.
- **SharkServer** It is also possible to run a persistent SharkServer which retains session state, such as cached tables, across executions of the Shark CLI. This can be very useful for those hoping to keep a few tables in cache for long periods of time. To start a SharkServer, run

  ```
  $./bin/shark --service sharkserver <port>
  ```

  A client can connect to the server using

  ```
  $./bin/shark -h <server-host> -p <server-port>
  ```

- **Spark Scala Shell With Spark Bindings (Advanced)** Shark provides a simple API for programmers to convert results from SQL queries into a special type of RDDs (Resilient Distributed Datasets). RDD's are a key abstraction in Spark. This lets you integrate SQL query processing with machine learning, and provides a unified system for data analysis using both SQL and sophisticated statistical learning functions. To launch a Spark shell with Spark functionality, run

  ```
  $./bin/shark-shell
  scala> val youngUsers =
  sc.sql2rdd("SELECT * FROM users WHERE age < 20")
  scala> println(youngUsers.count)
  ...
  scala> val featureMatrix =
  youngUsers.map(extractFeatures(_))
  ```

```
scala> kmeans(featureMatrix)
```

Chapter 5

Results

The report sections above summarized the project details, design and implementation along with the various problems we faced and the solutions to those problems. The results highlight a summary of the results of our project with respect to technologies used,architecture implemented for Data Loading, and the user interface setup for Data Visualization.The results have been been elaborated below

1. Technologies Used
   The project helped us in getting acquainted with both Big Data and other technologies most significantly *Hadoop,MapReduce,Hive,Sqoop ,Spark,Shark,MongoDB,MySQL,D3* amongst many others. It helped comprehend the urgent need of Big Data today, how the basic hadoop stack operates and the novelty of Real Time Spark and Shark to get results of queries returning millions of rows in a matter of seconds

2. Architecture
   This project was initially started with a standalone single node hadoop cluster on which the complete ETL system and Data Visualization system is setup. We were able to complete setting up a multinode cluster however the final connection of the Django UI with multinode hadoop cluster is still pending

3. ETL Automater
   For relational database systems automated technologes like SSIS previously exist to ease the ETL process. However lack of a similar technology which is highly customizable for our use resulted in

scripting of the python automater which is a one click solution to the complete ETL process required for both EdX data packages as well as locally generated data. It is the main backend engine being used in Django to ease loading of data into Hive tables efficiently

4. Data Visualisation
   Te very first step to any kind of mining is to visualise data. Then main aim of this part of our project was to create a single platform not only for any researcher wanting to visualise input data but also for professors wanting to observe interesting results via visualisation.It harnesses state-of-the-art javascript libraries such as D3,Dimple and Google Charts to ensure highest quality of visualisations to ease comprehension

5. All-in-One UI Finally to ensure complete ease of use by the end user we have ensured that the complete system is supported by a slick easy to use UI which supports both Data Loading and Visualisation. Snapshots of the User Interface and its resulting processes have been added The complete backend runs on Django which is python compatible which easily allowed our systems to fit in with Django which powers the UI

6. Right Technology Right Place The Data Loading and Visualisation have been separated so that the correct systems are being used for each process. Data Loading fully utilises Hive and MapReduce which is the ideal system of loading of data. On the other hand for Data Analysis which requires multiple fast computations Spark and Shark are used.

Chapter 6

Conclusion And Future Work

IITBombayX is a MOOC (Massive Open Online Course) with an objective to provide free online education. We have designed a UI based system capable of uploading both the amazon server data of the EDX (edX packages) and the IIT Bombay's local server data onto hadoop file system.For the purpose of loading the data to HDFS we have used hive.This data is then queried with the use of spark and shark to get the useful results from the data. At last javascript libraries D3.js and Dimple.js are used to visualize the results obtained. Our system would help the stackholders (students as well as instructors) to analyse the data easily and derive useful information from it so that weak students could be helped to imporve and good ones praised. The future work in this area colud be :

- Sequential Data Mining

  The Sequences which the different users follow during their learning is determined. These sequences can be used to find the most optimal sequence i.e. the sequence followed by the students scoring the highest marks. Also it could be found which sequence is followed by most of the users.

- Detecting Undesirable Student Behaviors[31]

  The objective of detecting undesirable student behavior is to discover/detect those students who have some type of problem or unusual behavior such as: erroneous actions, low motivation, playing

games, misuse, cheating, dropping out, academic failure, etc. Several DM techniques (mainly, classification and clustering) have been used to reveal these types of students in order to provide them with appropriate help in plenty of time.

- Latent Knowledge Estimation[32][33]

  Latent Knowledge Estimation means that predidcting the knowledge level of a student while he is giving a test. Also LKE can be used to find how much has the student learned from the course and the knowledge he already had about the subject.

- Detecting Possibility of Student's Drop Out[34]

  The probability of a student to drop out the course can be found by analysing the users activity in various parts of the course like forum, wiki, videos etc.

- Using the mongodb data of the EDX for getting more results

  The mongodb data of edx got from the amazon server can be used further to get more resutls and predict student behavior.

- Integrating the django frontend with multinode cluster

  The system we made using django is to be implemented on the multinode cluster.

References

[1] "Hadoop installation for single node." `http://hadoop.apache.org/docs/r1.2.1/single_node_setup.html`, 2014. [Online accessed 01-July-2014].

[2] "Apache Documentation. Available at "`http://hadoop.apache.org/docs/r0.18.2/`". Downloaded in May 2014."

[3] T. White, *Hadoop: The Definitive Guide: The Definitive Guide*. O'Reilly Media, 2009.

[4] "Hive getting started." `https://cwiki.apache.org/confluence/display/Hive/GettingStarted`, 2014. [Online accessed 01-July-2014].

[5] "Hive Documentation Wiki. Available at "`https://cwiki.apache.org/confluence/display/Hive`". Downloaded in May 2014."

[6] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: a warehousing solution over a map-reduce framework," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1626–1629, 2009.

[7] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive-a petabyte scale data warehouse using hadoop," in *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pp. 996–1005, IEEE, 2010.

[8] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pp. 10–10, 2010.

[9] C. Engle, A. Lupher, R. Xin, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica, "Shark: fast data analysis using coarse-grained distributed memory," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 689–692, ACM, 2012.

[10] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica, "Shark: Sql and rich analytics at scale," in *Proceedings of the 2013 international conference on Management of data*, pp. 13–24, ACM, 2013.

[11] "Derby server documentation." `https://cwiki.apache.org/confluence/display/Hive/HiveDerbyServerMode`, 2014. [Online accessed 01-July-2014].

[12] "Hadoop installation for multinode." `http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/`, 2014. [Online accessed 01-July-2014].

[13] "Django book." `http://www.djangobook.com/en/2.0/index.html`, 2014. [Online accessed 01-July-2014].

[14] "Installing ganglia." `http://docs.hortonworks.com/HDP2Alpha/index.htm#Deploying_Hortonworks_Data_Platform/rpminstall/rpm_install_ganglia/rpm_install_ganglia.html`, 2014. [Online accessed 01-July-2014].

[15] "Spark documentation." `http://spark.apache.org/docs/latest/configuration.html`, 2014. [Online accessed 01-July-2014].

[16] "Shark documentation." `http://shark.cs.berkeley.edu/`, 2014. [Online accessed 01-July-2014].

[17] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *JEDM-Journal of Educational Data Mining*, vol. 1, no. 1, pp. 3–17, 2009.

[18] C. Romero and S. Ventura, "Educational data mining: a review of the state of the art," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 6, pp. 601–618, 2010.

[19] "Average edx enrollments." http://techcrunch.com/2014/03/03/study-massive-online-courses-enroll-an-average-of-43000-studer 2014. [Online accessed 01-July-2014].

[20] "Sqoop documentation." http://sqoop.apache.org/docs/1.4.4SqoopUserGuide.html, 2014. [Online accessed 01-July-2014].

[21] "Django tutorial." http://db.apache.org/derby/papers/DerbyTut/install_software.html, 2014. [Online accessed 01-July-2014].

[22] "Django redis." http://redis.io/,https://github.com/ui/django-rq, 2014. [Online accessed 01-July-2014].

[23] "Django rq." http://python-rq.org/docs/, 2014. [Online accessed 01-July-2014].

[24] "Python package pyhs2." https://github.com/BradRuderman/pyhs2, 2014. [Online accessed 01-July-2014].

[25] "edx data package details." https://edx-wiki.atlassian.net/wiki/display/OA/Research+Data+Package+Details, 2014. [Online accessed 01-July-2014].

[26] "Test data." http://www.it.iitb.ac.in/frg/wiki/index.php/G7_-_Tools_for_Big_Data#edX_Logs, 2014. [Online accessed 01-July-2014].

[27] "Hiveserver2 documentation." https://cwiki.apache.org/confluence/display/Hive/SettingUpHiveServer2, 2014. [Online accessed 01-July-2014].

[28] "D3.js documentation." https://github.com/mbostock/d3/wiki, 2014. [Online accessed 01-July-2014].

[29] "Dimple.js examples." http://dimplejs.org/examples_index.html, 2014. [Online accessed 01-July-2014].

[30] "Monitoring hadoop cluster with ganglia." https://coderwall.com/p/qye7_q,http://blog.manula.org/2013/01/monitoring-hadoop-cluster-with-ganglia.html, 2014. [Online accessed 01-July-2014].

[31] M. Cocea and S. Weibelzahl, "Cross-system validation of engagement prediction from log files," in *Creating new learning experiences on a global scale*, pp. 14–25, Springer, 2007.

[32] R. S. d Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," in *Intelligent Tutoring Systems*, pp. 406–415, Springer, 2008.

[33] M. Khajah, R. M. Wing, R. V. Lindsey, and M. C. Mozer, "Integrating latent-factor and knowledge-tracing models to predict individual differences in learning," in *Proceedings of the Seventh International Conference on Educational Data Mining*, 2014.

[34] S. B. Kotsiantis, C. Pierrakeas, and P. E. Pintelas, "Preventing student dropout in distance learning using machine learning techniques," in *Knowledge-Based Intelligent Information and Engineering Systems*, pp. 267–274, Springer, 2003.