A Hadoop-driven Data Analysis System

Ashish Jindal
Rutgers University
Piscataway, NJ, USA
Email: ashish.jindal@rutgers.edu

Yikun Xian
Rutgers University
Piscataway, NJ, USA
Email: siriusxyk@gmail.com

Sanjivi Muttena
Rutgers University
Piscataway, NJ, USA
Email: sanjivi.muttena@rutgers.edu

***Abstract*—**The complexity of modern analytics needs is outstripping the available computing power of legacy systems. Distributed systems like Hadoop compliments this requirement for storing and analyzing huge sets of information by providing a platform for parallel processing of large data sets stored over multiple machines. This project aims to setup a Hadoop infrastructure and demonstrate its power for big data analysis. Some machine learning models will also be deployed in this system so that developers can directly call corresponding APIs to execute training and testing models.

***Index Terms*—**Hadoop-driven, Data Analysis, Infrastructure

I. Project Description

Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. Hadoop stores data as it comes in - structured or unstructured - saving time on configuring data for relational databases. In our project we will store a large dataset in Hadoop file system and NoSQL database like MongoDB, which will be used as source for data retrieval and visualization. Our project falls in the category *Scalable Algorithms Infrastructure*. The main stumbling points for this project are identifying a large dataset for Hadoop, configuring Hadoop for multiple clusters and building a data analytics system over Hadoop.

The project has four stages: requirement gathering, design, infrastructure implementation, and user interface.

A. Stage 1 - The Requirement Gathering Stage

Following are the deliverables for this stage:

a) General Description: This project is about setting up an infrastructure for big data analytics using Hadoop and demonstrate a data analytics application using Wikipedia data of different types. The first part is to construct a distributed computing platform based on Hadoop and some database for analytical result storage. The system provides some well-encapsulated APIs for running map-reduce jobs. The second part include a real application on data analysis driven by Hadoop. It mainly demonstrate the feasibility of architecture to apply Hadoop into existing web-based application.

b) User Type 1: People interested in analyzing data that can't be stored on a single machine.

- User Interaction Modes: Hive query language.
- Real World Scenarios:
  - Scenario 1 Description: Processing of large server logs.
  - System Data Input for Scenario 1: Log files.
  - Input Data Types for Scenario 1: Structured log files.
  - System Data Output for Scenario 1: Information based on query.
  - Scenario 2 Description: Searching PageRank Index.
  - System Data Input for Scenario 2: Big sources of page links.
  - Input Data Types for Scenario 2: Text.
  - System Data Output for Scenario 2: PageRank value based on query link.

c) Project Timeline and Division of Labor: We will start with studying about Hadoop and map reduce systems and then set up a Hadoop system on local host. When we are successful in doing that we will find a small dataset, large enough to fit on a single system and start test analysis over it using Hadoop. Then we will repeat the same process by setting up Hadoop in pseudo distributed mode. Parallelly two team members will start building the demo analytics application over Hadoop. In the end we will setup Hadoop on multiple clusters and deploy our application over it.

- Week 1 and Week 2: Work on requirement gathering and design of system using Hadoop. Setup Hadoop on local host as a single cluster and perform dummy analysis to test its functionality.
- Week 3 and week 4: Setup Hadoop infrastructure over multiple clusters and create an analytics application demonstrating big data analysis using Hadoop.
- Week 5: Test the application and prepare the necessary documentation & project report.

B. Stage 2 - The Design Stage

The project is divided into three parts. Infrastructure module involves Hadoop deployment, data storage management and integration with data analysis models and web application. Analytic module is responsible for processing large data set based on Hadoop clusters and provides interfaces for access to models. UI module is constructed along with a web application to read and visualize data from backend. The general architecture consisting of these three modules is shown in figure 1.

1) Infrastructure Module: The base Apache Hadoop framework is composed of the following modules:

1) Hadoop Common
2) Hadoop Distributed File System (HDFS)

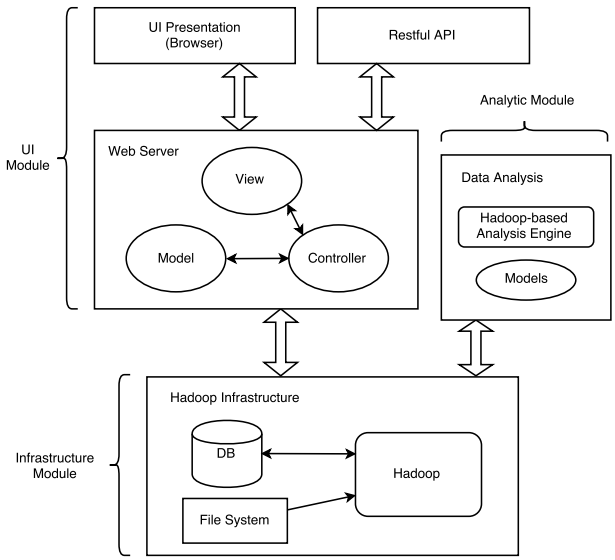


Fig. 1: System Architecture and Modules

3) Hadoop YARN
4) Hadoop MapReduce

Hadoop ecosystem refers to the above mentioned basic modules with the collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark, Impala, Apache Flume, Apache Sqoop, Apache Oozie etc. In addition to the base Hadoop modules, we will be using following additional modules from Hadoop ecosystem -

1) Flume - A distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into HDFS.
2) Hive - Built on the MapReduce framework, it is a data warehouse that enables easy data summarization and ad-hoc queries via an SQL-like interface for large datasets stored in HDFS.
3) hBase - A column-oriented NoSQL data storage system that provides random real-time read/write access to big data for user applications.
4) Spark - To implement fast, iterative algorithms for advanced analytics such as clustering and classification of datasets.

Our main data sources are Wikipedia dumps. Wikipedia dumps can be directly loaded into HDFS but for collecting streaming data we need a aggregation framework to help us put data into HDFS, so we use Apache flume for this task. Hive is used to query from the large set of data in HDFS. Hive uses map reduce paradigm to accomplish this task, but we can also use Impala for fast data query if we want to skip map reduce and directly deal with HDFS. Depending on the scalability requirements of the application, we can either use Hive or Impala. We will be using Spark to implement our analytic algorithms and classifying data as per application requirement. The over all system infrastructure and data flow can be seen in the figure 2.

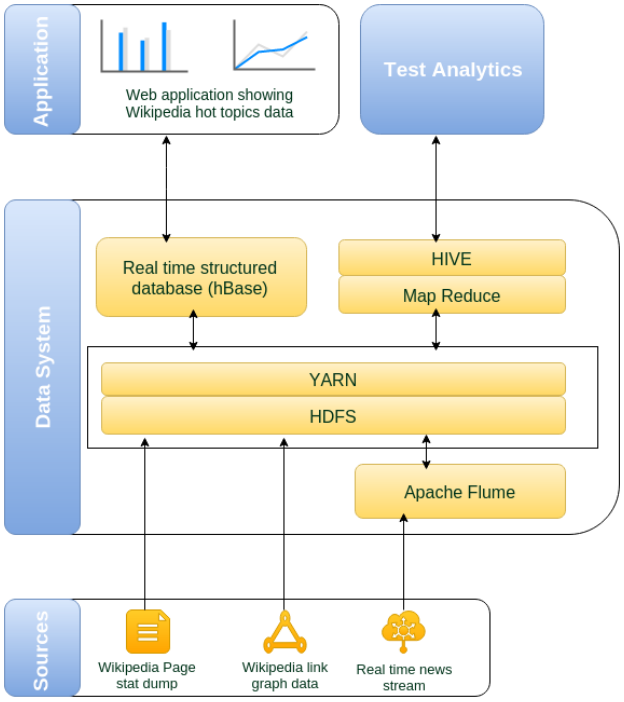


Fig. 2: Infrastructure

a) Integrity Constraint: In the project, raw data from Wikipedia is unstructured like graph and document and cannot be stored in traditional relational database system, but in NoSQL database. For entity integrity, each link or document is unique identified by additional ID. Since there is no relation between tables, foreign keys are not necessary. Meanwhile, Wikipedia dumps format is changing over the time and data set is diversified from one to the other, so we cannot define a fixed format to save the data, but to update data parsing algorithm according to changes to maintain reusability and stability of the system.

2) Analytic Module:

a) Description:

This part focuses on data management and analysis based on Hadoop infrastructure. Data sources includes Wikipedia dumps [1] are in the form of text information. Hadoop platform is used to store these large data sets and support data analysis accessed by interfaces described in infrastructure module. Specifically, analysis part in this project mainly provides encapsulated interfaces for training, testing and calling models including counting algorithm, trending algorithm and pagerank algorithm. Therefore, pagerank is designed for evaluating importance and resourcefulness of Wikipedia links and flow.

b) Flow Diagram:

As figure 3 shows, the module provides two functions, data crawler and data analysis engine. Data crawler manages data input and output by interacting with data sources and Hadoop infrastructure. Data analysis engine based on Hadoop

[1] https://dumps.wikimedia.org/

computing resources provides strong modeling function to analyze textual data.

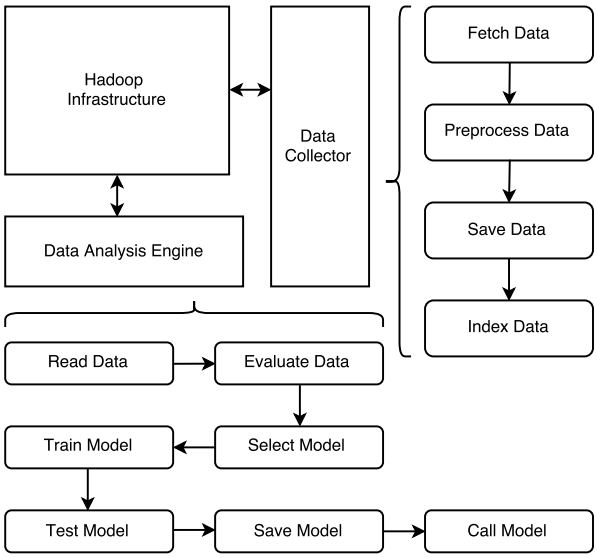


Fig. 3: Data Analysis Flow Diagram

c) High Level Pseudo Code System Description: The corresponding pseudo code for work flow of data management is shown as follows. The first one describes a basic procedure to retrieve data, clean data and store data.

1) Extract textual data from links
2) Batch clean and filter data
3) Save data to Hadoop platform
4) Index data for future search

The second one gives a general illustration of modeling data for any algorithms used in this project including page ranking, similarity measure and topic models.

1) Read data from Hadoop
2) Perform basic statistics on dataset
3) Select suitable model on requirement
4) Train model on Hadoop
5) Test model by evaluation index
6) Save model including parameters to database
7) Return model interface

d) Algorithms and Data Structures:

Three major algorithms in the project are page count, trending and page rank.

Firstly, page count is simply used for calculating sum of page views in a certain period. Trending is to calculate difference of two page views and return a sequence of values indicating the variation of a certain topic.

Second, algorithm 1 describes basic procedure to rank web links. The input is graph data file where each line represents a connection between two web links, so the whole data set can be regarded as a directed graph.

3) UI Module:

Algorithm 1 PageRank

function RANK($page - articles$)
  Extract pages from $page - articles$
  for each page p_i in $pages$ **do**
    Extract links l_i from p_i
  for $k = 1$ to K **do**
    for each page p_i in $pages$ **do**
      Calculate pagerank weight $PR(p_i) = (1-d) + d \times \sum \frac{PR(l_i)}{N(l_i)}$
  Normalize $PR(i)$ for every page p_i
  return list of $PR(i)$

a) Description:

The third phase of the project involves designing the UI for the user to access the trending data. The overall UI can be broken down into three segments. The first segment is the home page. The user will have two options. The first option would be to go for the current trending topics irrespective of the category. There will also be another search bar that allows the user to search for a specific topic. The search bar will also provide suggestions as the user types. These suggestions will be the topics that have been indexed in the database.

The second segment of the UI will be the listings of the topics that the user has requested. These can be either the current trending topics or the topic the user had searched. Once these listings have been generated, the user will have options to choose the source from which the data had been generated and go to that news source. The user also has options to change the source if multiple news sources are available. The final operation that the user can perform is to visualize the data stats on the topic.

The third segment of the UI is the data visualization. These stats will be the data obtained from the Wikipedia page of that topic. It would in essence visualize the number of page views over a period of time and would also provide a short description of the event whenever there is a sudden increase in the amount of traffic to the page. If location data is available, we also plan to show the regions in which the topic in question had generated interest.

b) Flow Diagram:

The figure 4 provides a general overview of the overall working of the UI. Each colored nodes represent webpages and the colorless nodes represent the actions the user can perform on them. The green nodes represent external webpages and the yellow nodes represent the webpages on the UI. The transition between the pages occur when the user clicks or interacts with the option in the colorless boxes.

c) High Level Pseudo Code System Description: The pseudocodes referred below provide a general overview of the working of the UI. The first pseudocode provides an overview of how the user can either search for the current trending topic or for any topic of his choice. The second pseudocode provides an overview of the choices that have been provided to the user. Here, the user can change the data source, obtain a

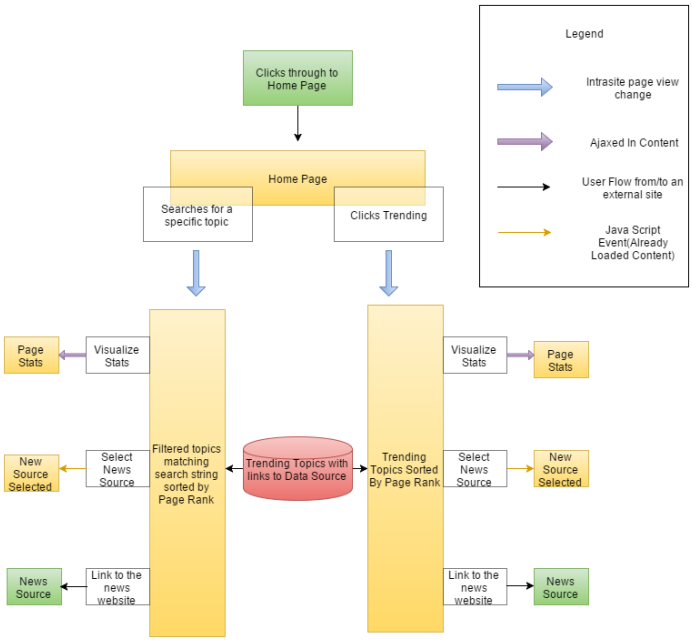


Fig. 4: UI Flow Diagram

Algorithm 2 Home Page

```
1: procedure HOMEPAGE
2:     initialpage ← homepage
3:     if getinputfrombutton = true then
4:         navigate ← Display("trending");
5:     else if getinputfromtextbox = true then
6:         navigate ← Display(textbox.text);
```

full description of the event by navigating to the data source or visualize the number of page views over time which essentially denotes the traffic.

Algorithm 3 News Source

```
1: procedure DISPLAY(contenttodisplay)
2:     if content = trending then
3:         Display topics sorted by page rank;
4:     else
5:         Search topics that match search string;
6:         Sort them by Page Rank;
7:     if user.choice = visualize then
8:         Display visualization for Page Traffic
9:     else if user.choice = gotosource then
10:        navigate ← source;
11:    else if user.choice = changedatasource then
12:        source = newsource
```

C. Stage 3 - The Implementation Stage

1) Development Environment Overview: Implementation of the project includes three parts: Map-Reduce Engine, MongoDB-based interactive backend and a web application based front end for data visualization. The development environment parameters shown shown in table I. Map-Reduce Engine is responsible for collecting data from given sources, cleaning the data, aggregating the information, applying the analytics operations on data and then dump the data into a MongoDB instance. The analytical results in mongoDB can be then accessed through REST based API which is implemented using Java Spring Framework[2] including Spring Web MVC Framework and Spring Data MongoDB Component. The Rest API is then invoked by web application to query data from MongoDB and visualize results using front-end CSS and Javascript libraries like Bootstrap[3] and HighCharts[4].

TABLE I: Development Environment List

General Environment	
Operating System	Ubuntu 14.04
Map-Reduce Engine	
Programming Language	Java
Database	MongoDB
IDE	Eclipse Mars with HDT plugin
Other	Hadoop 2.2
Data-set overview	
Size of one log file	300MB - 400MB (Uncompressed)
Number of log files used	14
Number of Wikipedia Pages	Around 6 Million
Data Visualizer (Back-end)	
Programming Language	Java
Database	MongoDB
Framework	Spring, SpringMVC, Gradle
Data Visualizer (Front-end)	
Languages	HTML, CSS, Javascript
CSS library	Bootstrap
JS library (JS)	jQuery, HighCharts, select2

2) Development Environment Setup: As an infrastructure related project, development environment setup is as an essential part as model and application implementation. There are two parts required to be configured from scratch. The first one is to setup Hadoop distributed cluster and Eclipse development environtment, and the second one is to setup web application development including package management control.

a) Hadoop Environment Setup: Following configuration is based on Ubuntu 14.04. First, let's setup Hadoop environment on our system.

1) We need to create a group and a user to manage Hadoop with following commands:

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
sudo adduser hduser sudo
```

2) Hadoop 2.2 can be downloaded from its archive and extracted to specified installed directory, for instance, `/usr/local/hadoop-2.2.0` in this case. Change its owner and group by following command:

```
$ sudo tar vxzf hadoop-2.2.0.tar.gz -C /usr/
    local
$ cd /usr/local
$ sudo chown -R hduser:hadoop hadoop-2.2.0
```

[2]http://spring.io/
[3]http://getbootstrap.com/
[4]http://www.highcharts.com/

3) Environment variables including Hadoop installation directory, Hadoop executable programs, HDFS and YARN locations, are required to be added to system configuration:

```
$ vi ~/.bashrc
export JAVA_HOME=/usr/lib/jvm/jdk_version/
export HADOOP_INSTALL=/usr/local/hadoop-2.2.0
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
```

4) Now, we change to Hadoop configuration directory to setup environment.

```
$ cd $HADOOP_INSTALL/etc/hadoop
```

Hadoop is run in Java environment, so Java home directory must be explicitly specified in Hadoop environment file.

```
$ vi hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/jdk_version/
```

Now, Hadoop can be tested if it has been correctly installed by checking version:

```
$ hadoop version
```

5) Limited to hardware resource, we configure Hadoop as single node running on pseudo-distributed mode by adding following content:

```
$ vi core-site.xml
<property>
  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
```

6) Setup YARN including resource manager and node manager in above pseudo-distributed mode by adding following content:

```
$ vi yarn-site.xml
<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>
```

And configure its corresponding parameters:

```
$ mv mapred-site.xml.tempalte mapred-site.xml
$ vi mapred-site.xml
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

7) Finally, setup HDFS by specifying data node and name node directories:

```
$ mkdir -p /tmp/hddata/hdfs/namenode
$ mkdir -p /tmp/hddata/hdfs/datanode
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/tmp/hddata/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/tmp/hddata/hdfs/datanode</value>
</property>
```

8) Now, we can run Hadoop in pseudo-distributed mode by formatting name node on HDFS, starting HDFS and YARN manager.

```
$ sudo rm -rf /tmp/hadoop-hduser
$ hdfs namenode -format
$ start-dfs.sh
$ start-yarn.sh
```

By running `jps` command , we can check if all following 6 Hadoop processes have already started successfully:

```
2583 DataNode
2970 ResourceManager
3461 Jps
3177 NodeManager
2361 NameNode
2840 SecondaryNameNode
```

where the first column is pid in Linux and differs from various systems. Opening browser and type:

```
http://localhost:8088/cluster
```

we can see Hadoop dashboard as shown in figure 5.

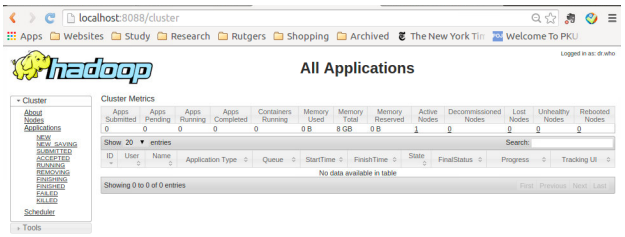


Fig. 5: Hadoop Dashboard Information

Then, we setup Eclipse development environment for Hadoop by installing a key plugin called HDT (Hadoop Development Tool)[5]. After installation, we can create new Map/Reduce project (figure 6) and set up HDFS directory in Eclipse (figure 7).

b) Web Application Environment Setup: The web application is implemented in Java EE technology based on Spring Framework and MongoDB. Since there are a lot of Java packages involved in web application development, we use Gradle[6] to manage these packages and to automate project compilation and building. Its installation is simple: extract it to local path, and add following configuration to $.bashrc$:

[5]http://hdt.incubator.apache.org/
[6]http://gradle.org/gradle-download/

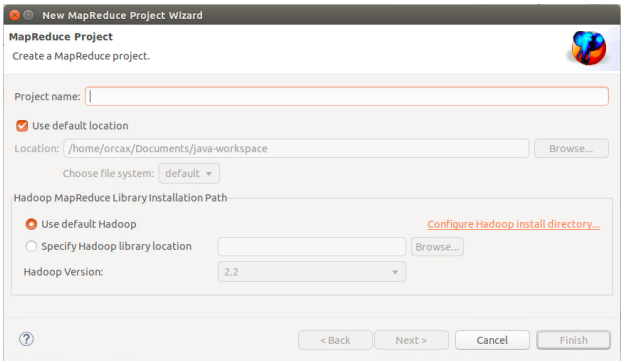


Fig. 6: Create Hadoop Map/Reduce Project in Eclipse

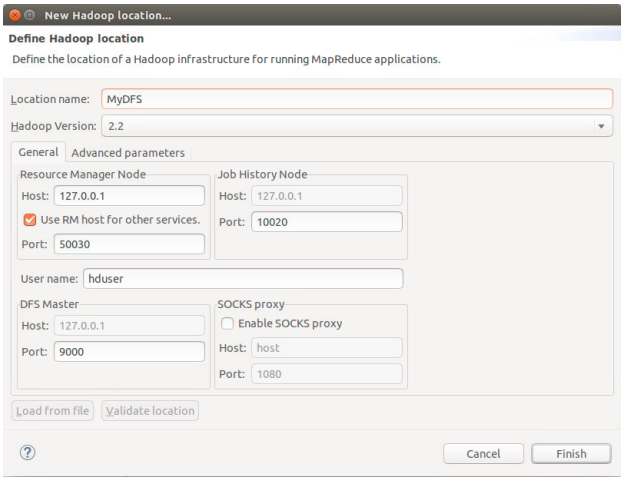


Fig. 7: Setup HDFS in Eclipse

```
export GRADLE_HOME=/usr/local/gradle-2.9
export PATH=$PATH:$GRADLE_HOME/bin
```

and type `gradle -v` to check if it is successfully installed. Meanwhile, we also need to install "Eclipse Jetty" and "Gradle Integration for Eclipse" in Eclipse Java EE version so that Gradle project for Java web application can be created in Eclipse.

Then, we need to edit `build.gradle` to specify plugins and library dependencies for the project. The plugin configuration is shown as following:

```
apply plugin: 'java'
apply plugin: 'eclipse-wtp'
apply plugin: 'war'
apply plugin: 'jetty'
```

The library dependencies are configured as follows, where we mainly use Spring framework, Jackson for JSON conversion, MongoDB library and some other common libraries:

```
dependencies {
  compile group: 'commons-collections', name: '
      commons-collections', version: '3.2'
  testCompile(
    'junit:junit:4.10',
    'org.springframework:spring-test:4.1.6.RELEASE'
    )
  compile 'javax.servlet:javax.servlet-api:3.1.0',
    'javax.servlet:jstl:1.2',
    'org.springframework:spring-context:4.1.6.
        RELEASE',
    'org.springframework:spring-context-support
        :4.1.6.RELEASE',
    'org.springframework:spring-webmvc:4.1.6.RELEASE
        ',
    'org.springframework:spring-orm:4.1.6.RELEASE',
    'org.springframework:spring-tx:4.1.6.RELEASE',
    'org.springframework:spring-aop:4.1.6.RELEASE',
    'org.springframework.security:spring-security-
        web:4.0.1.RELEASE',
    'org.springframework.security:spring-security-
        config:4.0.1.RELEASE',
    'org.springframework.data:spring-data-mongodb
        :1.8.1.RELEASE',
    'org.codehaus.jackson:jackson-mapper-asl
        :1.9.13',
    'org.codehaus.jackson:jackson-core-asl:1.9.13',
    'com.fasterxml.jackson.core:jackson-core:2.4.3',
    'com.fasterxml.jackson.core:jackson-databind
        :2.4.3',
    'com.fasterxml.jackson.core:jackson-annotations
        :2.4.3',
    'org.mongojack:mongojackname:2.1.0',
    'log4j:log4j:1.2.16',
    'dom4j:dom4j:1.6.1',
    'org.mongodb:mongo-java-driver:3.0.4',
    'commons-io:commons-io:2.4',
    'commons-logging:commons-logging:1.2'
  compile fileTree(dir: 'src/main/webapp/WEB-INF/lib
      ', include: '*.jar')
}
```

To use MongoDB libraries for integration with Spring framework, we need to add a maven source:

```
repositories {
    maven { url 'https://oss.sonatype.org/content/
        repositories/snapshots' }
    mavenCentral()
}
```

Now, after Gradle project is refreshed in Eclipse, the project structure of Java web application will be automatically generated and libraries will be downloaded from Maven repositories and imported to the project as well.

Then, to integrate Spring framework into Java web application, we need to set dispatch servlet in `web.xml`:

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.
      DispatcherServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

and specify package place and Mongo sources in Spring configuration file `dispatch-servlet.xml` as following:

```
<context:component-scan base-package="
    java_package_name" />
<mvc:annotation-driven />
<bean class="org.springframework.web.servlet.view.
    InternalResourceViewResolver">
  <property name="prefix" value="/WEB-INF/views/"/>
  <property name="suffix" value=".jsp"/>
</bean>
```

```
<mvc:resources mapping="/css/**" location="/WEB-INF/
    css/" />
<mvc:resources mapping="/js/**" location="/WEB-INF/
    js/" />
<mvc:resources mapping="/images/**" location="/WEB-
    INF/images/" />
<mvc:resources mapping="/fonts/**" location="/WEB-
    INF/fonts/" />
```

After above configuration, the web application project can be run as Jetty and visited through browser.

3) Implementation: Implementation detail will be illustrated in 3 parts: Map-reduce engine, back-end data retriever and front-end visualizer.

a) Map-reduce Engine Implementation: We deployed a Hadoop version 2.2 instance on our local machine in pseudo distributed mode to run the map-reduce tasks. The map-reduce tasks, jobs and DB connectors are implemented using Java and we use Hadoop's local/standalone instance for easy debugging of map-reduce tasks which helps in fast development and testing. Apache HDT provides an eclipse plugin for easy integration with Hadoop 2.2 which we used to access HDFS and create map-reduce Java projects. The infrastructure of map-reduce engine is shown in figure 8.

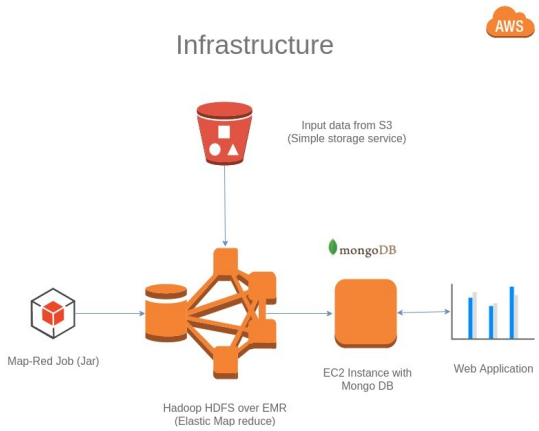


Fig. 8: Map-reduce Infrastructure

Currently we have implemented following tasks through map-reduce jobs:

1) Read input a path of a directory, containing the Wikipedia page view log files in compressed format (gzip).
2) Hadoop helps in decompression of input files and then provides the input to the mapper task.
3) Mapper reads the document containing hourly pageview count, cleans the data and saves the hourly page view count data.
4) Reducer takes in the hourly page view counts and aggregates them to generate daily page view stats and then dumps the calculated data into a mongoDB instance.
5) Find most popular Wikipedia pages over a week using a simple baseline algorithm.
6) Calculate Page ranks for Wikipedia pages, using another data-set from Wikipedia containing the page link information (inbound and outbound links for each page).

b) Back-end Data Retriever Implementation: This part is implemented by following Java EE convention integrated with Spring framework. In detail, it incorporates two layers: controller layer and service layer.

For controller layer, it processes web requests and responses and interacts with service layer. There are two types of response types: JSP page and JSON format. The previous one includes visual presentation elements, style sheet and element control scripts. The second one can be retrieved through HTTP request on any platform application. We also expose a REST API to query data from MongoDB which can be used for other analytical purposes. The controller class is defined and created using Spring annotation, i.e. the page count controller can be written as following:

```
@Controller
@RequestMapping("/pagecount")
public class PageCountController {..}
```

where `@Controller` annotation tells Spring which class will be initialize as a controller, and `@RequestMapping` annotation limits controller action URI.

For service layer, it manages connection with MongoDB to retrieve data and transforms it into Java entity. MongoDB is configured independently so that the system can retrieve data from any deployed sources. Meanwhile, it encapsulates friendly API for querying data with regular expression and provides auto conversion from JSON data in MongoDB into Java bean. For example, The page count service class is defined as follows:

```
@Service
public class PageCountServiceImpl extends
    BaseServiceImpl implements PageCountService {..}
```

where `@Service` tells Spring which class will be instantialized as service, `BaseServiceImpl` is predefined class that provides common methods to read from database, and `PageCountService` is the interface that defines required methods.

c) Front-end Data Visualizer Implementation: The front-end web pages are enhanced by jQuery[7], Bootstrap, Highcharts and select2[8]. Web page are created in JSP file, which will be transformed into HTML format in user's browser. Since every page has header and footer, these common elements in the pages are defined as layout and loaded by every web pages through `include` tag:

```
<jsp:include page="header.jsp" flush="true" />
<jsp:include page="footer.jsp" flush="true" />
```

where `header.jsp`, `footer.jsp` are layout files.

User browser will first request web pages with HTML, CSS and Javascript. When user starts to search keywords to see the visualized charts, the browser will send request

[7]https://jquery.com/
[8]https://select2.github.io/

to back-end data retriever to query for data through AJAX. Therefore, the page will not refresh to load data and visualize it except when user requests for new web pages.

4) Sample Data and Results: We are using two data-sets in this project.

- Wikipedia page view count logs (Generated hourly)[9]: We downloaded a subset of data: 1 log file pertaining to 1 hour of page view counts for each day from Nov 1st, 2015 to Nov 14th, 2015. Size of each file (Uncompressed) is around 300MB. Number of clean records in DB = 6426361. This dataset is used for page view count and topic popularity trend.
- Wikipedia page-to-page link database[10]: We use the set of Wikipedia Page link dataset to calculate page ranks using map-reduce. The dataset will have around 5 million pages and around 100 million links. There is also another cleaned version of this dataset available on web, which we may use without bothering to write the clean/aggregation map-reduce jobs to calculate the page rank.

TABLE II: Data Set Overview

Wikipedia page-count dataset	
Size of one log file	300MB - 400MB (Uncompressed)
Number of log files used	14
Number of Wikipedia titles	Around 5 Million
Wikipedia page-article dataset	
Size of one file	45MB - 3.54GB (compressed)
Number of log files used	27
Number of Wikipedia Pages	Around 5 Million
Number of Links	Around 100 Million

We use a small part of page view count data for visualization. As shown in figure 9, the sample plot from our application shows page popularity comparison of wiki pages of 3 countries (India, China and United States).

The attached source codes include two parts: wikiPageViewDataParser is the Java code for map-reduce job on Hadoop and WebApp is the Java EE code for back-end data management and visualization.

D. Stage 4 - User Interface

UI of this system is functionally divided into two web applications. Wikipedia Page View Statistics provides a search interface to look for page counts and page trend for any of the titles. Wikipedia PageRank Index provides customized search along with some interesting findings on PageRank values regarding certain categories. The default pages of two interface are shown in figure 10a and figure 10b.

1) Page View Statistics UI: This single-page app consists of a search bar and two charts display areas. User can type keywords in the search bar to look for specific page view counts. It supports two search modes: fuzzy search and

[9]https://dumps.wikimedia.org/other/pagecounts-raw/

[10]https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2

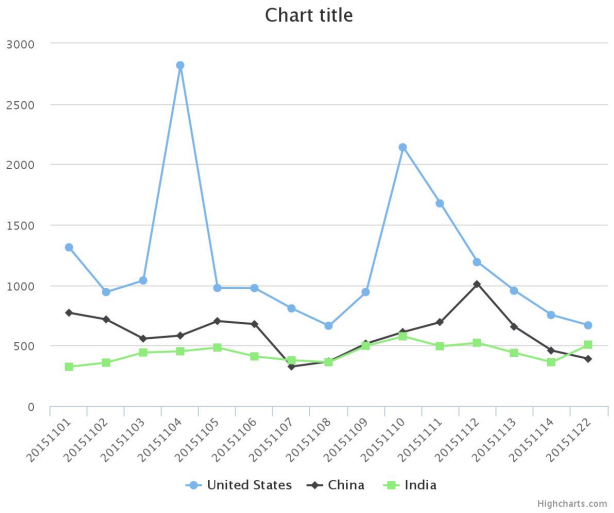
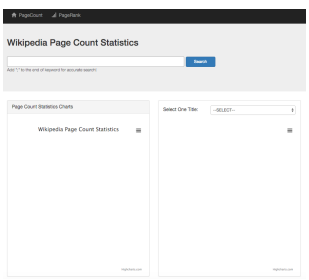
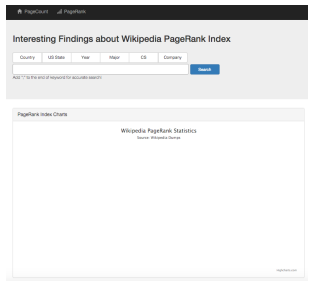


Fig. 9: Sample chart

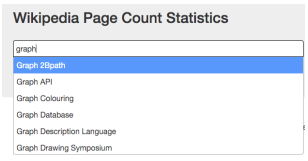


(a) Page View Default Interface

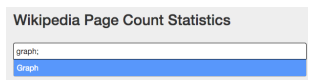


(b) PageRank Default Interface

accurate search. System will match prefix of potential titles in the database with input keyword and return first 20 titles to users to select from. taking into consideration that a large number of words will match a short keyword, the user can also use accurate search by typing a semicolon at the end of the word so that system will find the title which is exactly the same as that of the input keyword. These two modes are shown in figure 11a.



(a) Fuzzy Keyword Matching



(b) Accurate Keyword Matching

If there is no matching data found, the web app will prompt the message ”No result found” in the dropdown list, as shown in figure 12.

The search interface also supports multi-item selection as shown in figure 13;

After ”Search” button is clicked, the web app will load data according to the search input via AJAX technology and plot the page view count data as a line chart. Figure 14a shows three lines each represents a sequence of page counts of a

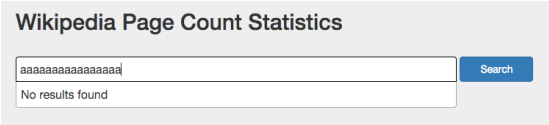
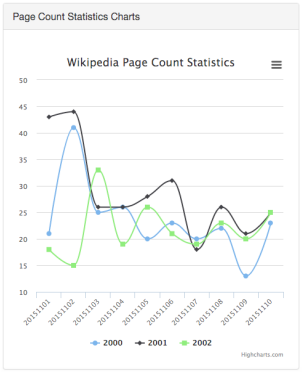


Fig. 12: No-result-found message

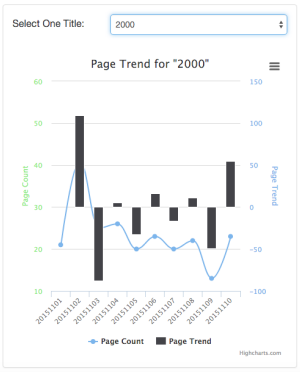
Fig. 13: Multi-selection

specific title, where ”2000”, ”2001” and ”2002” are three title names.

Then, on the right panel box, user can select one of the three topics, for example, ”2000”, after which the page trend together with page view count will be displayed on the same chart as shown in figure 14b. The blue line represents the same meaning as that in page view count chart line, and the black histogram means the page trends values. When user selects other titles in the drop down list, the corresponding chart will be refreshed.



(a) Page Count Chart



(b) Page Trend

An example of the complete user interface is shown in figure 15.

2) PageRank Index UI: In PageRank web app page, there is a query area and a display area as well, except that some extra buttons can be clicked to shown some predefined charts. After each button is clicked, the chart will be refreshed and a corresponding chart is displayed in the below area. Six predefined charts respectively represents the PageRank value for some selected countries, US states, years from 1950 to 2016, some selected academic majors, some research areas in computer science and part of companies whose leader is under 40. Their PageRank chart are shown in figure 16.

The search bar in the PageRank web app works the same as that in page view count web app, which also supports fuzzy search, accurate search, multi-selection, etc. After typing some keywords and clicking the ”Search” button, the corresponding chart regarding Pagerank values will be plotted and displayed.

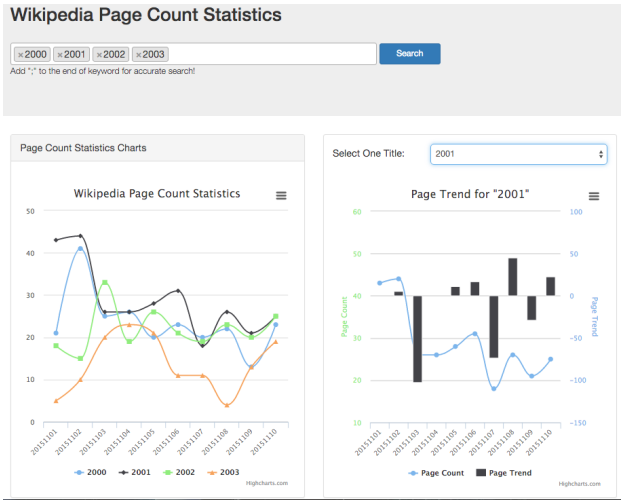
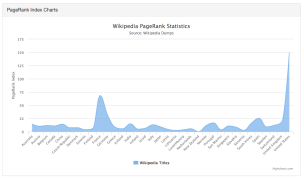
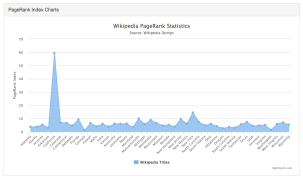


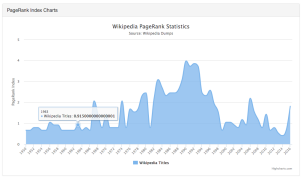
Fig. 15: Integrated UI for Page View



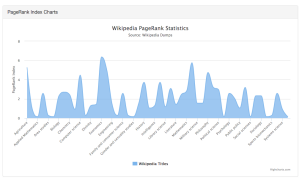
(a) Countries



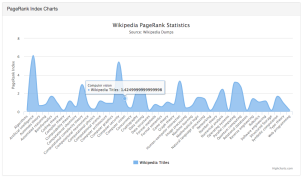
(b) US States



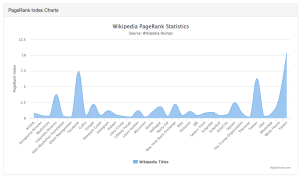
(c) Years



(d) Majors



(e) CS



(f) Companies

Fig. 16: Some findings about PageRank

Figure 17 shows that the user search for four keywords: ”Graph embedding”, ”Graph invariant”, ”Graph isomorphism” and ”Graph isomorphism problem” and the result is a histogram with respect to PageRank values of these four titles.

II. Conclusion

In this project, we mainly explored features of Hadoop infrastructure by setting up Hadoop on local computer, implementing and running several algorithms with respect to big data set and finally visualizing the analytical results. In detail, we setup Hadoop 2.2 in pseudo-distributed mode on Ubuntu 14.04 as hands-on practice and got familiar with Hadoop configuration including HDFS and YARN. Then, we

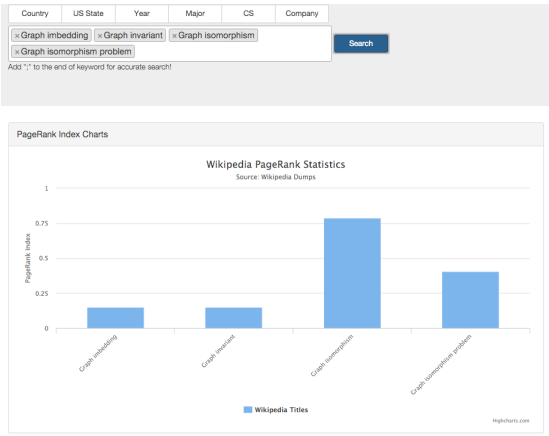


Fig. 17: PageRank Search Example

selected Wikipedia dumps as our data source including one set of Wikipedia page view count statistics and the other set of Wikipedia page-article documents. For the first dataset, we implemented map-reduced version of page counting and trending algorithm on Hadoop. For the second dataset, we implemented map-reduced version of PageRank algorithm which can analyze an evaluation value for each web page on Wikipedia. Finally, we build a database to store these analytical results by using NoSQL database MongoDB imported from Hadoop HDFS. A web application is also built to display these results in a vivid way by visualizing page count, page trend and page rank into charts and enabling user-friendly interaction.

ACKNOWLEDGMENT

We really appreciate that we could have this chance to make this project and felt satisfied with our achievements. We thank our supervisor, Professor James Abello, for assisting us with technology, algorithm and constructive suggestion, which definitely made our project promising and competitive.

REFERENCES

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web." 1999.

[2] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual web search engine," *Computer networks*, vol. 56, no. 18, pp. 3825–3833, 2012.

[3] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon s3 for science grids: a viable solution?" in *Proceedings of the 2008 international workshop on Data-aware distributed computing*. ACM, 2008, pp. 55–64.

[4] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[5] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[6] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[7] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1.

[8] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742*, 2007.

[9] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001, pp. 289–296.

[10] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. IEEE, 2010, pp. 684–689.