ISTD 50.043 Database and Big Data Systems
Group 14 Documentation

Table of Contents

Group Members:

Liu Kaiyu 1003576
Joseph Chng 1003811
Soo Ming Wei 1003832

Web Application Features:

1. Home Page
    a. Users can login to an account or sign up for a new account. They can also search for books on the home page by asin or book title. On the home page, users can browse books and click on the book image to view the book details. There is also an option to view more books by clicking on the page number at the bottom of the home page. Users can also click on ‘About’ on the navigation bar to view more information about this website.

2. Types of users
    a. There are two types of users, one is a normal user and another is admin. An admin user and normal user will be able to add reviews to books but only an admin user can view the book management page and edit a book by clicking on it. Only an admin user can add books via the ‘Add Book’ page and indicate these fields: asin, title, price, description and image URL.

3. Book Management Page
    a. An admin user can click on any book’s image to edit the book details, such as ASIN, title, price, description and image URL

4. Edit Book Page
    a. An admin user can click on any book's image to edit the book details, such as ASIN, title, price, description and image URL

5. Book Details Page
    a. The book details page contains information such as book title, ASIN, categories, description and price of the book. In this page, the user is able to see what other books were bought with this book and what other books were viewed with this book by clicking on the ASIN of the respective book.

6. Review Page
    a. Users can leave reviews on the books in the 'Review' page, by leaving an overall rating of one to five, and then giving a summary of their review, followed by their review

7. Add Book Page
    a. An admin user can click on 'Add Book' on the navigation bar to add a book to the website, by indicating the book's details, such as ASIN, book title, price, description and image URL.

8. Login/Sign Up Page
    a. Users with existing accounts may login by entering User ID and password on the Login Page. Users without existing accounts may sign up for a new account by filling up User ID, Username, Password and Confirm Password fields.

Extra Features:

9. 'I'm Feeling Lucky' feature
    a. When a user clicks on the 'I'm Feeling Lucky' button on the homepage, it will generate a random book from the database and display its book details. From here, user can choose to add review or view the book's details.

10. Pagination feature in the Home and Book Management Page
    a. Each page contains 12 books with 3 books in each row for a total of 36229 pages to hold all the books in the database with added pages to accommodate books added by admin.

ISTD 50.043 Database and Big Data Systems
Group 14 Documentation

Extra Effort:

11. Retrieved users from MySQL database to obtain existing users' credentials so that there is an existing pool of users. We did this to mimic an actual book website, where there will be existing users already with an account and can immediately use the system.

Design:

Production System (Web Application)

- Flask full stack
    - UI: Flask templates and Bootstrap
    - Backend: Flask, SqlAlchemy (ORM framework)
- Database:
    - MySQL and MongoDB

Analytics System:

- Hadoop and Spark
- Data ingestion: MySQL and Mongo Client

Scripts:

- Python (Fabric - SSH client)
- Shell scripts

Analytics Task:

1. Pearson Correlation
    - Taking the review length of each book from review.csv and price of each book from book_meta.json from hdfs we use the following formula to calculate the correlation:

ISTD 50.043 Database and Big Data Systems
Group 14 Documentation

- Using formula for Pearson correlation

$$r = r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}.$$

we created the following map-reduce tasks:

- (purple) map average_review_length x price
- (red) map average_review_length [extract from RDD]
- (orange/yellow) map square of average_review_length
- (blue) map square of book price
- (green) map book price [extract from RDD]
- Each corresponding reduce task calculates the sum of each map separately i.e. (purple) sum of all *average_review_length x price*

- We then filter out all null values in our dataset and do a **flatmap** on the data to separate it out to different values that are required.
- That is followed by a **reduce** function to get the sum of each value in the various values that we need in the formula.
- We then simply calculate the correlation value using the value that we have obtained through the mapreduce function in pyspark

2. TF-IDF
   Implementation based on Apache Spark sample codes
   https://github.com/apache/spark/blob/master/examples/src/main/python/mllib/tf_idf_example.py

Automation Scripts:

Dependencies: pip3 install boto3 fabric
** If you are running on Windows, please make sure the line separator of the .sh files are in linux LF(\n)
** If errors occur when running set_up.py, delete all instances created manually and re-run again
** Running python scripts in system terminal may pause sometimes or fail to respond, which will lead to set up error, we recommend using VsCode/ Pycharm in built terminal to run the scripts.
** After running set_up.py, a URL will appear in the terminal which will bring you to the website. It is important that you sign up for an admin account immediately by clicking on the ‘Sign Up’ button at the top right-hand corner of the website and enter ‘admin’ for User ID, Username, Password and Confirm Password Fields.
** After running get_correlation.py, a URL will appear which will bring you to a page that you can

ISTD 50.043 Database and Big Data Systems
Group 14 Documentation

download the correlation results from.
** After running get_tfidf.py, a URL will appear which will bring you to a page that you can get the tfidf results from.

Steps to run the scripts:

1. git clone https://github.com/Pomatoo/db_project.git
2. enter this command on the terminal: cd db_project/automation_scripts
3. enter this command on the terminal:pip3 install boto3 fabric
4. enter this command on the terminal: python3 set_up.py (
5. enter this command on the terminal: python3 get_correlation.py
6. enter this command on the terminal: python3 get_tfidf.py
7. enter this command on the terminal: python3 shutdown.py

AWS credentials and instance configuration will be specified in file “aws_config.conf”
When instantiating an AwsManager() object it will read config from “aws_config.conf”, and create a security group and an access key.

set_up.py

This script will create all instances needed for both production and analytics systems.
MySQL instance will initiate with mysql_setup.sh (install MySQL server, and import data)
Mongo instance will initiate with mongo_setup.sh (install mongo server, and import data)
Web instance will initiate with web_setup.sh (install web application)
So on and so forth

After all nodes are running, it will configure the name node and data nodes.
Configuration:

1. Get the public key from name node to local, then send the key to each data node.
2. Configure name node by running set_up_namenode.sh in name node instance
    a. Set MASTER, WORKERS IPs, configure Hadoop and Spark
    b. Send (correlation.py , run_correlation.sh) and (tfidf.py and run_tfidf,sh) to name node
    c. Send http_server.py to name node.
3. Configure data node by executing set_up_datanode.sh in each data node
4. Start Hadoop and spark, start http server.
5. Start Web application, URL will be displayed in console.

All created key, security group and instances will be logged in “created_aws_instances.json”

get_tfidf.py and get_correlation.py

These two scripts have similar functions, ssh to namenode and execute respective jobs by running bash ./run_tfidf.sh or ./run_correlation.sh

shutdown.py

Read information from “created_aws_instances.json” and delete them all.

tfidf.py

TF-IDF implementation based on apache Spark sample code.
https://github.com/apache/spark/blob/master/examples/src/main/python/mllib/tf_idf_example.py

ISTD 50.043 Database and Big Data Systems
Group 14 Documentation

correlation.py
Pearson Correlation implementation code

Bash Shell Scripts:

data_node.sh
Update, install java 8 and set swappiness

name_node.sh
Update, install java 8, pyspark, mongo/MySQL clients
Download and unzip Hadoop and Spark Zip files
Set up SSH keys

run_correlation.sh
Select data from MySQL via MySQL client, pipe output to reviews.csv
Import data from mongo via mongoexport, output to book_meta.json
Save reviews.csv and book_meta.json to HDFS and execute spark job (correlation.py)
Merge output in HDFS and save to ~/result/correlation_result.txt with timestamp

run_tfidf.sh
Select data from MySQL via MySQL client, pipe output to reviews.txt
Save reviews.txt to HDFS and execute spark job (tfidf.py)
Merge output in HDFS and save to ~/result/tfidf_result.txt with timestamp

set_up_namenode.sh
Set MASTER and WORKER variables, configure Hadoop and Spark.
Zip configured Hadoop and Spark, send to each data node via scp
Move and set permissions for Hadoop and Spark folder

set_up_datanode.sh
Unzip Hadoop and Spark that sent by name node
Move and set permissions for Hadoop and Spark folder

ISTD 50.043 Database and Big Data Systems
Group 14 Documentation

Classes:

We created a class AwsManager which adds a layer of abstraction to boto3. We also created a WorkerThread class too.

AwsManager()

- instance_is_reachable(instance_id):
    - Return True if input instance is initialized
- get_instance_state(instance_id):
    - State of instance ‘running’/’terminated’
- terminate_instances(instance_id)
- remove_key_pair(key_name)
- remove_security_group(group_name)
- create_an_instance()
- __create_access_key()
- __create_security_group()

WorkerThread(AwsManagerInstance, thread_name, sh_file)

- get_instance_details():
    - Return details of the instance running on current thread
- instance_set_up():
    - Call AwsManagerInstance.create_an_instance() and run the input sh_file on the instance.