HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and Communication Technology

Big Data Storage and Processing

Tiki Products Storages

Group members:
Nguyen Van Thanh Tung - 20190090
Nguyen Vu Thien Trang - 20194459
Chu Hoang Duong - 20194429

Supervisor:
Dr. Dao Thanh Chung

Table of Contents

1 Introduction **3**

2 System Architecture **5**

3 Installation And Verification **7**
3.1 Apache Hadoop cluster 7
3.2 Run Apache Spark on top of YARN 7
3.3 ZooKeeper 7
3.4 HBase 9
3.5 Kafka 10

4 Data Flow to HBase **13**
4.1 Crawl Data 13
4.2 HBase Integration 13

5 Experiments **15**
5.1 Data Insights 15
5.1.1 Visualization 15
5.1.2 Machine Learning Model 17
5.2 Use case: Tiki Product Recommendation System 19

6 Conclusion **20**

List of Figures

2.1 System Architecture Overview 5

3.1 YARN 2 Slaves Resource 7
3.2 HDFS Data Nodes Resource 8
3.3 Spark Application sucessfully on YARN 8
3.4 ZooKeeper start up 9
3.5 HBase Startup 10
3.6 HBase Summary 10
3.7 HBase Regions Server 11
3.8 Start HBase Thrift Server 11
3.9 Python Script Create HBase Tables 11
3.10 categories table Created Successfully 11
3.11 Kafka Start Sucessfully with 3 brokers id 12

4.1 Crawler Python Source 14
4.2 HBase tables 14

5.1 Different category trees have different depth. 16
5.2 Category subspace. 16
5.3 Numerical Features Correlogram. 17
5.4 Numerical Features Correlogram. 18
5.5 Numerical Features Correlogram. 18

Chapter 1

Introduction

The genesis of Big Data has brought rich resources of knowledge that pave the way for science and analytics in various fields. Featuring an enormous amount of information, Big Data poses the challenge of how to effectively exploit the hidden insights to bring personal or business-level benefits. One of the most elementary examples of information overload is that the exploding number of options for a certain product or service makes it hard for users to make a decision. To that end, we focus on a recommendation system that offers a decision support tool for users to find their most relevant products.

On the other hand, the Big Data era also forces businesses face up to difficulties in storing and processing such a huge volume of resources. Big data is a term for data sets that are so large or complex that traditional data processing application software is inadequate to deal with them. We highlight the importance of developing a reliable architecture in the context of big data and put our application on top of such scenario.

In this project, we have designed a distributed system that have a salable architecture to collects, stores data, specifically here is data about products on the TIKI e-commerce platform. Besides, we also build Spark Application to do the machine learning task: clustering the products, and visualize the data.

- **Contribution 1**
  In this section, we introduce the our use case and highlight the challenge and potentiality of Big Data scenario.
- **Contribution 2**
  This section gives the detailed design of the model and explains each component's role in tackling the aforementioned challenges.
- **Contribution 3**
  We then give the detailed setting and benchmark datasets for our experiment to confirm the effectiveness of our model. The results are visualized and thoroughly discussed.
- **Contribution 4**
  We complete our report on the summary of our improvements to the Aspect Continual Sentiment Analysis and provide future work ideas.

Our source code and related documents are released at the project's repository [1].

Our report contains five chapters, including Introduction, System Architecture and Pipeline, Data Crawling and Processing, Data Application in Recommendation System, and Conclusion. The structure of this report is organized as follows:

- **Chapter 1: Introduction**
  In this section, we introduce the our use case and highlight the challenge and potentiality of Big Data scenario.
- **Chapter 2: System Architecture**
  The second section introduce Big Data technologies used in this projects as well as the blueprint of the project's architecture.
- **Chapter 3: Installation And Verification**
  This section gives the detailed installation and verify it step by step.
- **Chapter 4: Data Flow to Hbase**
  We declared the data flow to HBase that we can integrate the data from any source into this architecture where HBase is the center storage.
- **Chapter 5: Experiments**
  We then justify our architecture design to confirm its effectiveness in products analysis and production purpose.
- **Chapter 6: Conclusion**
  We finalize our report on the summary of our proposals to the Big Data scope and present future works ideas.

[1] `https://github.com/nguyenvanthanhtung2001/BigData-CapstoneProject`

Chapter 2

System Architecture

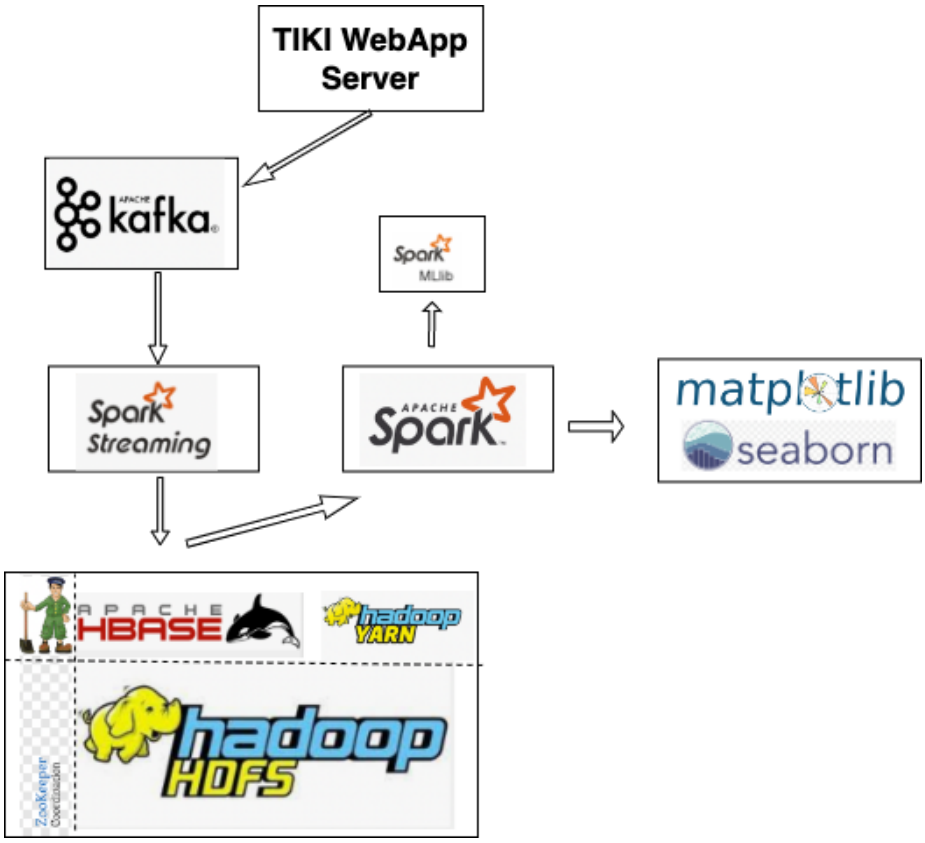


Figure 2.1: System Architecture Overview

We have designed a system like the one above on a cluster 3 physical Ubuntu 20.04 machines

In which frameworks are attached to the following roles:

- **Hadoop**: let us set up the cluster, allowing other frameworks to run on it and provide HDFS.

- **HBbase**: NoSQL database
- **Zookeeper**: provides Hadoop cluster management mechanism, making the cluster more reliable
- **Yarn**: resource negotiator
- **Spark Streaming**: data stream processing
- **Kafka**: data stream processing
- **Spark**: distributed data analytics
- **SparkMLlib**: Machine Learning engineering
- **Matplotlib, Seaborn**: visualization frameworks

And on top of the architecture is the TIKI WebApp Server, this is an app server we choose to get the data about the products and categories of products, which is the data source for our data pipeline.

In this system we will have 3 main steps for data flow:

- **Step 1**: Data will be taken from the source as Tiki WebApp Server, then go through Kafka, Spark Streaming are stream Pre-process steps to be stored on our main database, HBase.
- **Step 2**: The data stored at HBase will be retrieved for analysis and machine learning algorithms.
- **Step 3**: After analysis, the insight of the data will be visualized by Matplotlib, Seaborn in use of SparkContext (PySpark).

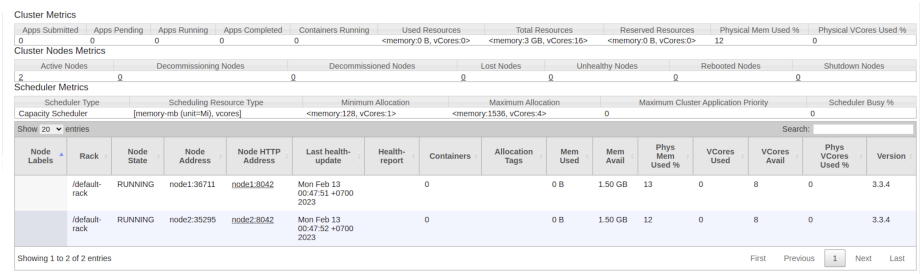
Chapter 3

Installation And Verification

3.1 Apache Hadoop cluster

We set up a Hadoop cluster of three physical machines with 1 name node and 2 data nodes. The name node can be set as the third data node but we did not do it, having separate name and data nodes helps to ensure scalability and reliability of the HDFS.

Below are the DataNodes and YARN report after we start up Hadoop cluster.



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical VCores Used %
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:3 GB, vCores:16>	<memory:0 B, vCores:0>	12	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
2	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority	Scheduler Busy %
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:128, vCores:1>	<memory:1536, vCores:4>	0	0

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	Phys Mem Used %	VCores Used	VCores Avail	Phys VCores Used %	Version
	/default-rack	RUNNING	node1:36711	node1:8042	Mon Feb 13 00:47:51 +0700 2023		0		0 B	1.50 GB	13	0	8	0	3.3.4
	/default-rack	RUNNING	node2:35295	node2:8042	Mon Feb 13 00:47:52 +0700 2023		0		0 B	1.50 GB	12	0	8	0	3.3.4

Showing 1 to 2 of 2 entries First Previous 1 Next Last

Figure 3.1: YARN 2 Slaves Resource

This mean the Hadoop cluster run successfully.

3.2 Run Apache Spark on top of YARN

For the main purpose of not have to setup another Spark Cluster. We run Spark Application on top of YARN to utilize the power of HDFS (for data storage) and YARN (for cpu usage) of Hadoop cluster.

3.3 ZooKeeper

Apache ZooKeeper is a distributed coordination service that provides a centralized infrastructure for applications to store and manage configuration information, naming, and provide synchronization services across a large number of nodes.

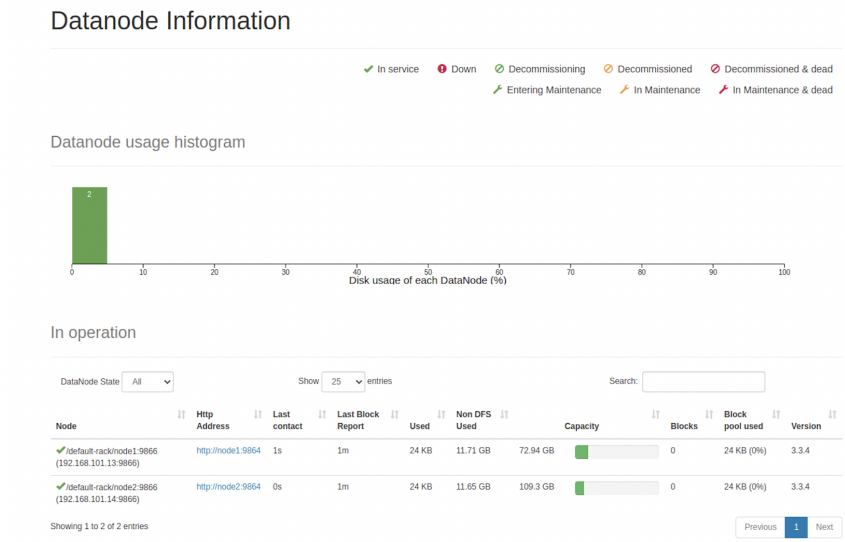


Figure 3.2: HDFS Data Nodes Resource

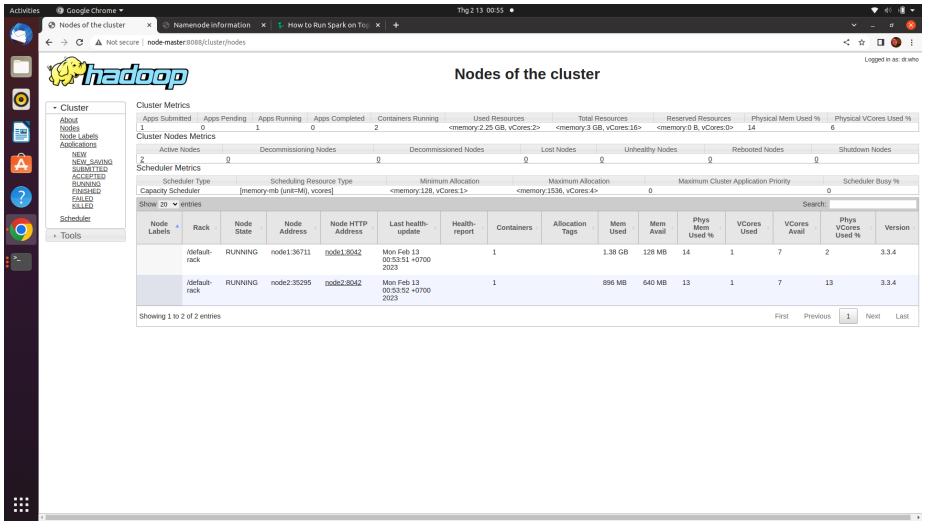


Figure 3.3: Spark Application sucessfully on YARN

ZooKeeper helps to manage distributed systems and applications, making it easier to coordinate and maintain the consistency of data across multiple nodes.

Because of that reason, we must start zookeeper for the purpose of server's consistency.

We can see that the cluster have 3 node and 1 will be the leader and 2 other are followers. when 1 leader shut down, the other follower will be the leader, then the cluster will consistent.

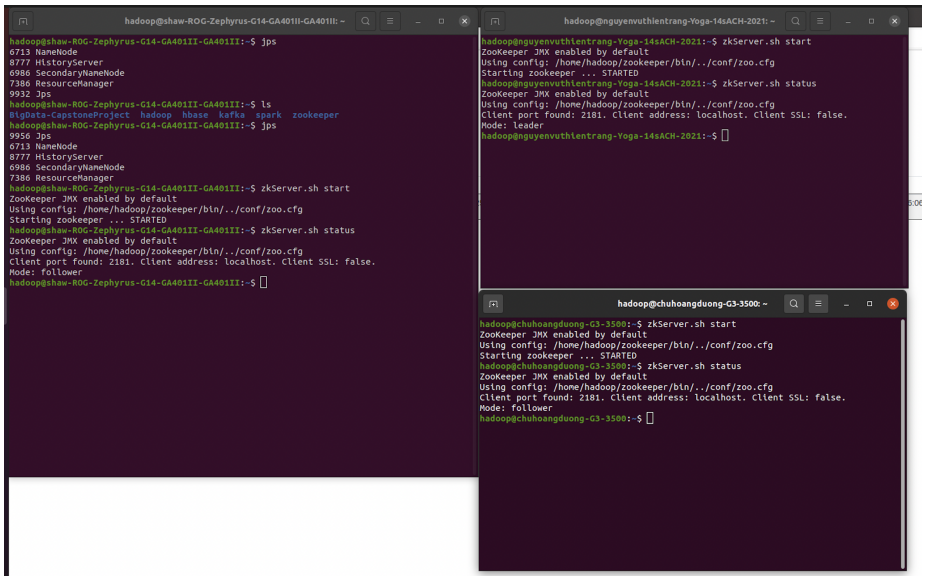


Figure 3.4: ZooKeeper start up

3.4 HBase

HBase is the main place where store the TIKI data of our architecture. Because the response from TIKI: products, categories will be JSON type. So store the data using HBase will take much more benefit compared to MYSQL:

- **Scalability**: HBase is designed to store and process large amounts of data, and it can scale horizontally by adding more nodes to the cluster. This makes it well suited for handling big data workloads, whereas MySQL is generally not as scalable for such use cases.
- **High availability**: HBase is a distributed database, which means that it has built-in mechanisms for ensuring data availability, such as automatic failover and data replication. In comparison, MySQL provides high availability solutions, but they require additional configuration and setup.
- **Performance**HBase is optimized for read-intensive workloads and can provide low-latency access to data. This makes it ideal for use cases such as real-time analytics and internet of things (IoT) applications, where quick access to data is critical.
- **Schema-less data storage** HBase is a NoSQL database and supports a schema-less data model, allowing it to store data in a flexible manner without defining a strict schema upfront. In contrast, MySQL requires a predefined schema, making it less flexible for handling semi-structured or unstructured data.
- **Flexibility** HBase allows for easy data retrieval and manipulation, as it supports random, real-time read/write access to large amounts of data. In comparison, MySQL is more limited in this regard, as it is optimized for transactional workloads.

For the large data set like TIKI Products we crawled up to 1.700.000 samples, nearly 4.0 GB, the present of a NoSQL database in our architecture is sufficient.

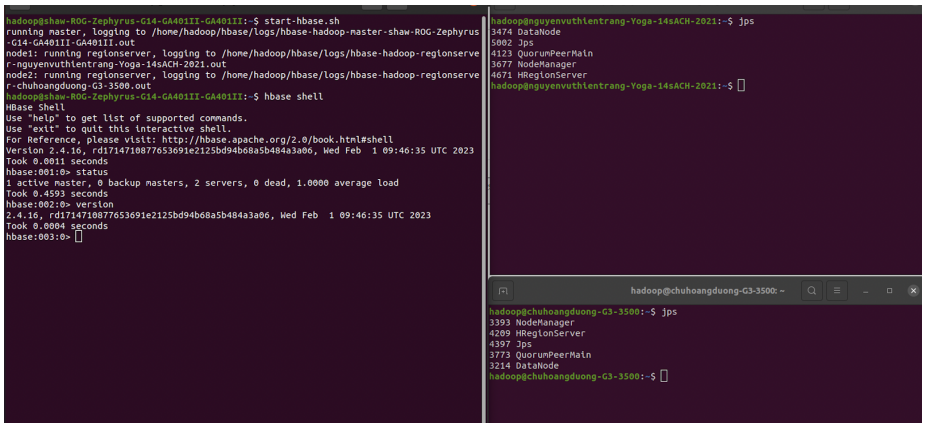


Figure 3.5: HBase Startup

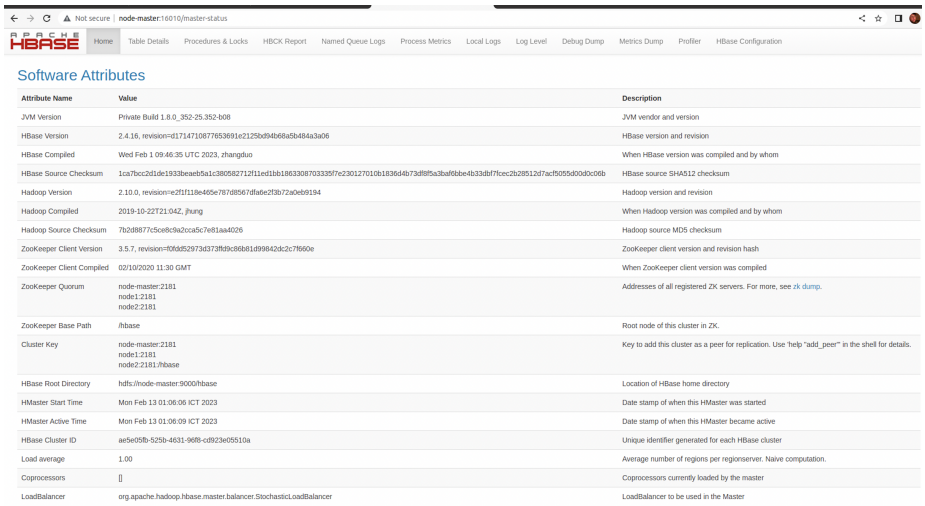


Figure 3.6: HBase Summary

Beside the HBase server run successfully on top of Hadoop HDFS, we start an HBase Thrift Server to provide HBase API, which help us to run python script to connect with the HBase database.

3.5 Kafka

Apache Kafka is a high-performance, distributed, and scalable messaging system designed for handling large amounts of data in real-time. It provides a publish-subscribe model for exchanging data between producers and consumers in a decentralized and fault-tolerant manner.

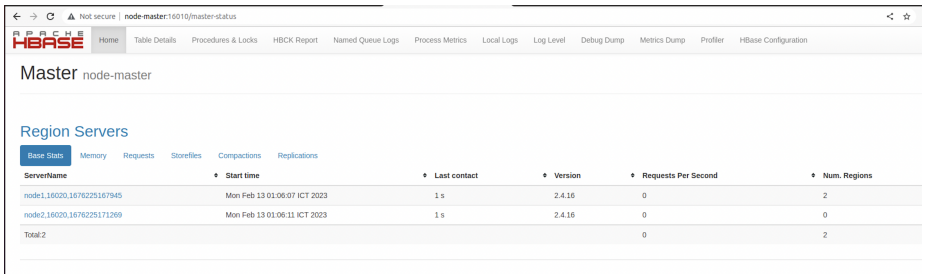


Figure 3.7: HBase Regions Server

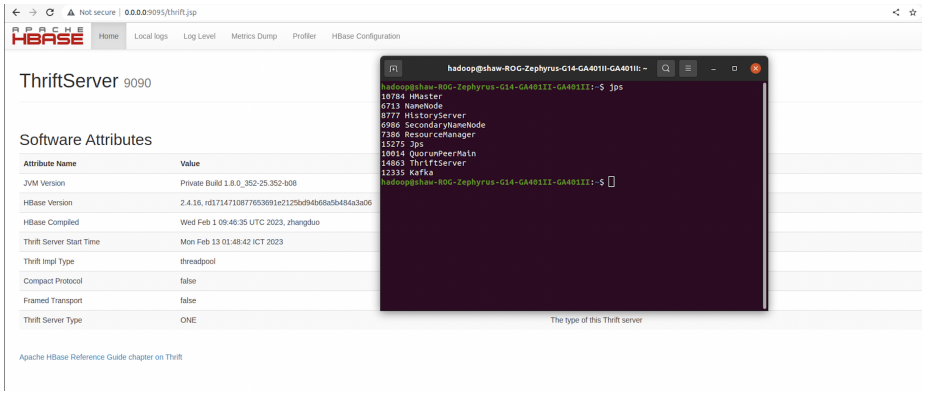


Figure 3.8: Start HBase Thrift Server

```python
import happybase

# establish a connection to HBase
connection = happybase.Connection("node-master", 9090)

# create a table
table_name = "categories_table"
column_families = {"category_info": dict()}
connection.create_table(table_name, column_families)
```

Figure 3.9: Python Script Create HBase Tables

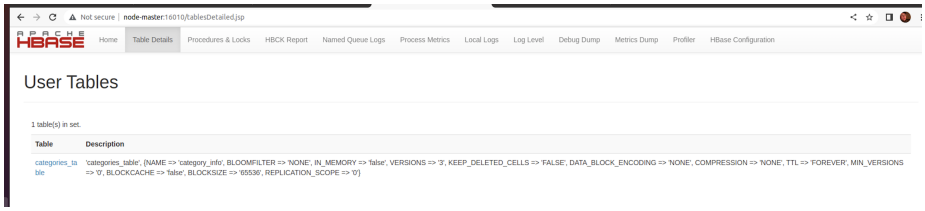


Figure 3.10: categories table Created Successfully

Because of this reason, we use Kafka to provide an message system between the crawler and the HBase database.

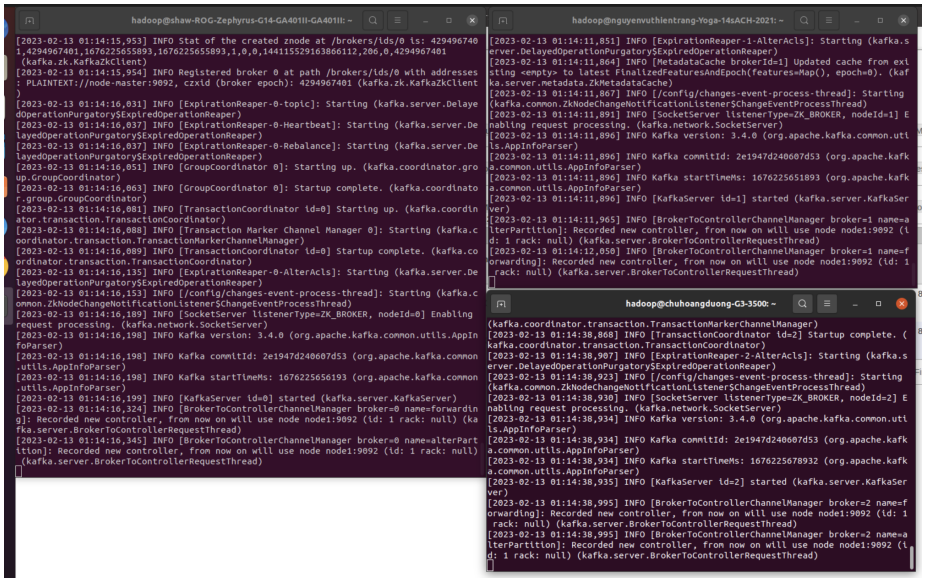


Figure 3.11: Kafka Start Sucessfully with 3 brokers id

Chapter 4

Data Flow to HBase

4.1 Crawl Data

The TIKI e-commerce platform have API to access to their products data. This is lucky because we don not have to use any browser to load page by page for each products of TIKI.

But the challenge is they have no documents for the API, so we use network catcher to analyze the network and extract the API end-points, parameters.

The API endpoint tasks we found are:

- 1. Get the sub-categories of one big category of products
- 2. Get the product list given an category id

From that clue, we use multiple thread from one client machine, send http requests through multiple proxy data server to collect the data.

The proxy data server will help we not be banished from the Tiki server when connect to them many times (nearly 1m7 products).

The way to do with categories will be the same as the above source code with products.

4.2 HBase Integration

To integrate data from TIKI crawler, we connect it through Kafka producer and consumer.

The producer will messages the item to consumer with the same topic: "product", "category" and then the consumer connects to the HBase through happybase to HBase thrift server to connect to HBase API.

Finally, the products table and categories table saved in the HBase for the future experiments.

```python
import os
import json
import time
import datetime
import requests
import pandas as pd
from threading import Thread


def get_products(category, page):
    headers = {
        "accept": "application/json, text/plain, */*",
        "accept-language": "en-US,en;q=0.9,vi-VN;q=0.8,vi;q=0.7",
        "origin": "https://tiki.vn",
        "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36",
    }

    params = {"category": category, "page": page, "limit": settings["page_limit"]}

    s_time = time.time()
    while time.time() - s_time < settings["waiting_time"]:
        try:
            res = requests.get(
                "https://tiki.vn/api/personalish/v1/blocks/listings",
                params=params,
                headers=headers,
                proxies=settings["proxies"],
            )
            res.json()
            break
        except:
            pass
    try:
        return res.json()["data"]
    except:
        raise Exception(f"{res.text}")
    return res.json()


def thread_flow():
    while True:
        try:
            category = categories.pop(0)
        except:
            return

        page = 0
        while True:
            page += 1
            try:
                page_products = get_products(category["id"], page)
                print("\n" + "\t".join([f"{x}" for x in [f"category={category['id']}", f"page={page}", f"{len(page_products)} products"]]))
                for product in page_products:
                    if product["id"] not in products:
                        products[product["id"]] = True
                        print(f"\nCollected {len(products)} products: {product}")
                        with open(f"{os.getcwd()}/output/{len(products)//1000+1}.txt", "a", encoding="utf-8-sig") as f:
                            f.write(json.dumps(product) + "\n")
                if len(page_products) != settings["page_limit"]:
                    break
            except Exception as e:
                print(e)

```

Figure 4.1: Crawler Python Source

Tables

User Tables | System Tables | Snapshots

2 table(s) in set. [Details]. Click count below to see list of regions currently in 'state' designated by the column title. For 'Other' Region state, browse to hbase:meta and adjust filter on 'Meta Entries' to query on states other than those listed here. Queries may take a while if the *hbase:meta* table is large.

Namespace	Name	State	Regions							Description
			OPEN	OPENING	CLOSED	CLOSING	OFFLINE	SPLIT	Other	
default	categories_table	ENABLED	1	0	0	0	0	0	0	'categories_table', {NAME => 'category_info', BLOOMFILTER => 'NONE', VERSIONS => '3', BLOCKCACHE => 'false'}
default	products_table	ENABLED	1	0	0	0	0	0	0	'products_table', {NAME => 'product_info', BLOOMFILTER => 'NONE', VERSIONS => '3', BLOCKCACHE => 'false'}

Figure 4.2: HBase tables

Chapter 5

Experiments

5.1 Data Insights

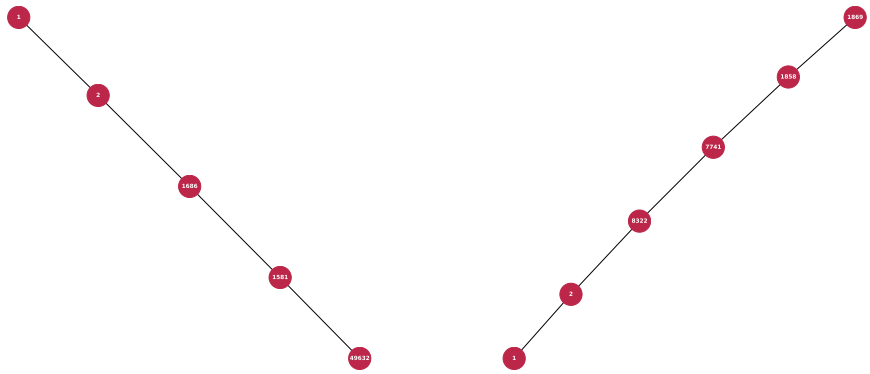
5.1.1 Visualization

The potential of the data the team collects is not small. The database that the team obtained after processing it included about 1 million and 700,000 products on the Tiki shopping platform. For each product collected, we have a total of 51 fields related to the product's characteristics. However, within the scope of this course, we only analyze and visualize the most prominent fields.

The initial data will be read from the HBase json file. For the easiest and most intuitive illustration, we use familiar Python visualization libraries such as Seaborn, NetworkX, and Matplotlib. Thus, the data will be converted into the Pandas DataFrame. However, since the goal of the course is to store and process big data, we always put every task in that context. Indeed, before we convert the data to pandas, we always query to get the required fields and samples. In this way we can minimize the curse of dimensionality phenomenon. At the same time, even though we have a large number of samples, we only choose 1–3 specific fields for each query and visualization. This way, the research environment fatality when calling toPandas() is avoided.

The most important feature of the products on the Tiki shopping floor is the category. Each product has a category tree corresponding to it, and each tree has a certain depth. Since there are more than 50,000 categories at different depths, we only illustrate a subspace of the category network. From Figure 5.2 It can be seen that the network is diverse and has a large spread, which is an important field for us to evaluate the correlation of products. In fact, we recommend products that are in the same category as the product the user has previously purchased/viewed.

We also consider numerical variables in Figure 5.3. However, the real variables don't give much insight. The distribution of numerical variables is quite skewed, however, it can be seen that there are a few variables that split into clusters. This poses a data clustering task based on real numeric fields, we will defer the details of the clustering problem to the subsection 5.1.2 Machine Learning Model.



(a) Product A (b) Product B

Figure 5.1: Different category trees have different depth.

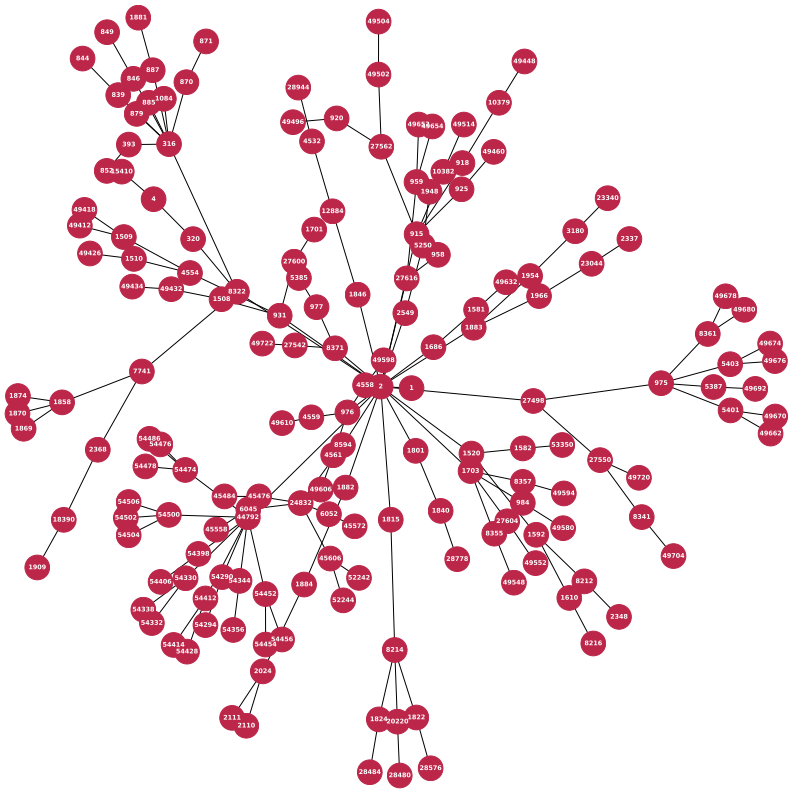


Figure 5.2: Category subspace.

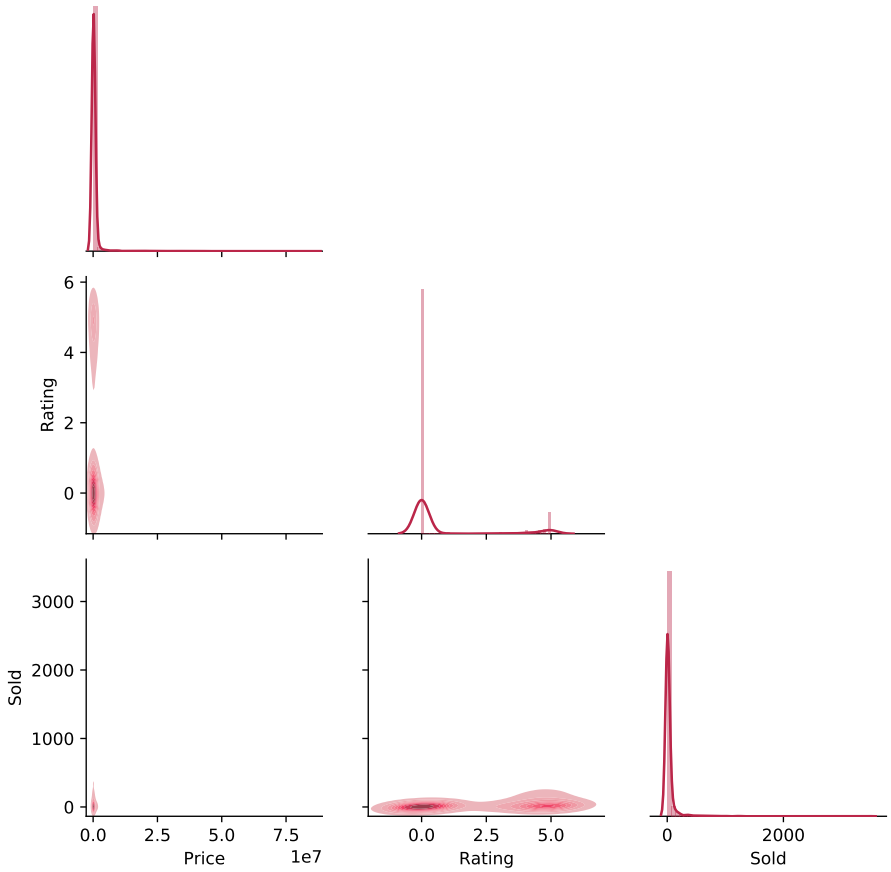


Figure 5.3: Numerical Features Correlogram.

5.1.2 Machine Learning Model

After reading JSON data from HBase, a machine learning application will proceed to receive data from HBase to feed into a machine learning model. The machine learning model will receive data from HBase and then build an ML library in PySpark. Learning about data with such complicated and imbalanced fields is not easy. Within the scope of this course, the application of a machine learning model is only experimental. Our team conducted an experiment on a data set of 1 million and 700,000 products taken from the Tiki shopping platform. Our team tested an unsupervised math problem by clustering the collected and processed products. In this clustering problem, we will obtain data with the same market segmentation and the same quality.

Based on Elbow's theorem in Figure 5.5, we can choose the number of clusters as 7 or 4. However, we prefer to choose the number of clusters as 7 due to having a better clustering score.

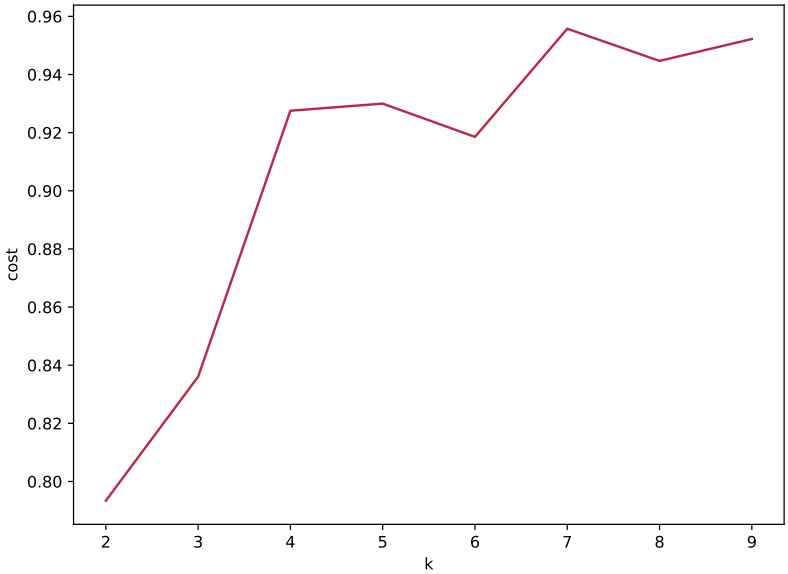


Figure 5.4: Numerical Features Correlogram.

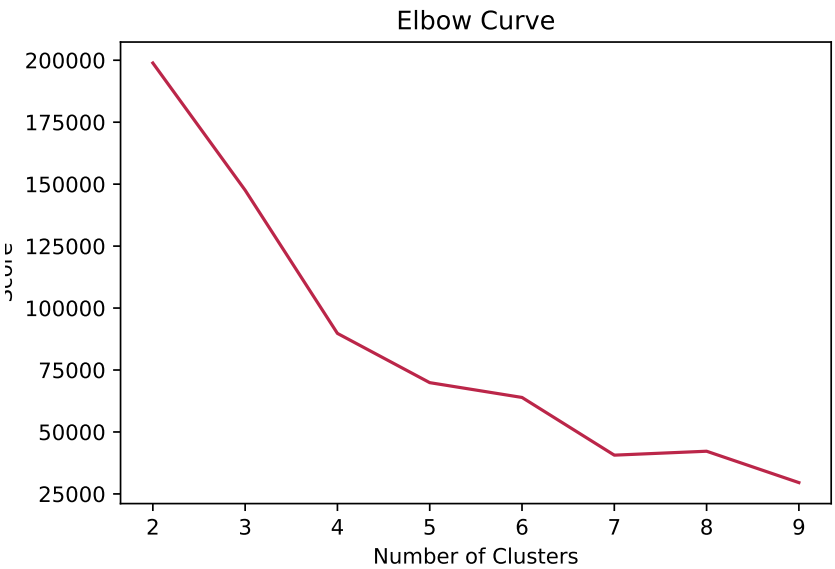


Figure 5.5: Numerical Features Correlogram.

5.2 Use case: Tiki Product Recommendation System

Our recommendations will be based entirely on product data of the Tiki platform. Because Tiki's user data and purchase history are highly confidential and we cannot apply for administrative rights, the problem of suggesting related products will be modeled by us as follows:

Input: A products that was bought buy an anonymous user. We denote this product by a.
Output: A list of products that have the potential to be purchased by that user.

```
PotentialProduct = []
for product p in the product space P do
    if p have the same name as product a then
        if p have the same category as product a then
            if p have the same cluster as product a then
                Add product p to PotentialProduct
            end if
        end if
    end if
end for
Filter PotentialProduct by Shippable, Inventory Availability.
Sort PotentialProduct by ratings, sold quantity, etc.
Return top-k products in PotentialProduct.
```

Chapter 6

Conclusion

In this project, We have created a distributed system with a scalable architecture for this project that collects and stores data, specifically data on products on the TIKI e-commerce platform. Additionally, we created the Spark Application to perform the machine learning task of clustering the items and data visualization.

Bibliography

[1] Apache Software Foundation. (n.d.). Spark SQL and DataFrames - Spark 3.2.0 Documentation., from https://spark.apache.org/docs/latest/sql-programming-guide.html

[2] Apache Software Foundation. (n.d.) | Apache Kafka., from https://kafka.apache.org/

[3] Apache Software Foundation. (2010). Hadoop. Retrieved from https://hadoop.apache.org

[4] Free and Open Search: The Creators of search, ELK Kibana. (n.d.). Elastic. https://www.elastic.co/

[5] Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, Martx27;in, & Vrgoc, Domagoj. (2016). Foundations of JSON schema. In Proceedings of the 25th International Conference on World Wide Web (pp. 263–273).

[6] Shafranovich, Y. (2005). Common format and MIME type for comma-separated values (CSV) files.

[7] Abadi, Martin, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... others. (2016). Tensorflow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265–283).

[8] Han, Jiawei; Kamber, Micheline; Pei, Jian (2011). "Data Transformation and Data Discretization". Data Mining: Concepts and Techniques. Elsevier. pp. 111–118.

[9] Kluyver, T., Ragan-Kelley, B., Fernando Px27;erez, Granger, B., Bussonnier, M., Frederic, J., ... Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides B. Schmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas (pp. 87–90).