Hanoi University of Science and Technology

School of Information and Communication Technology

COURSE REPORT

Search Trend Analysis for Advertising Optimization

SUBMITTED BY

Nguyen Van Giang, Vu Tri An, Le Van Cuong

{giang.nv212517M, an.vt212432M, cuong.lv212179M}@sis.hust.edu.vn

SUPERVISED BY

Dr. Dao Thanh Chung

Dr. Do Ba Lam

Asoc. Prof. Dr. Nguyen Binh Minh

Hanoi, spring 2022

Table of contents

Chapter 1: Introduction

1.1 Search trend analysis for Ads

Online community uses search engines for several purposes such as locating appropriate online marketing strategies and routine tasks like shopping and gaming. Combining search trends is an effort towards collecting the most popular trends over the internet. Search trends data can be very useful for marketers. For example, if we run a seasonal business (such as a home and gardening supply store), we'll want to ramp up our marketing efforts when search terms relevant to your business are trending. During spikes in search volume, the cost per click (CPC) will likely be higher, so be sure to allocate more budget in the campaigns when our products or services are trending. Motivated by this, this project aims to ***build a tool for search trend analysis***.

1.2 Summary of accomplishments

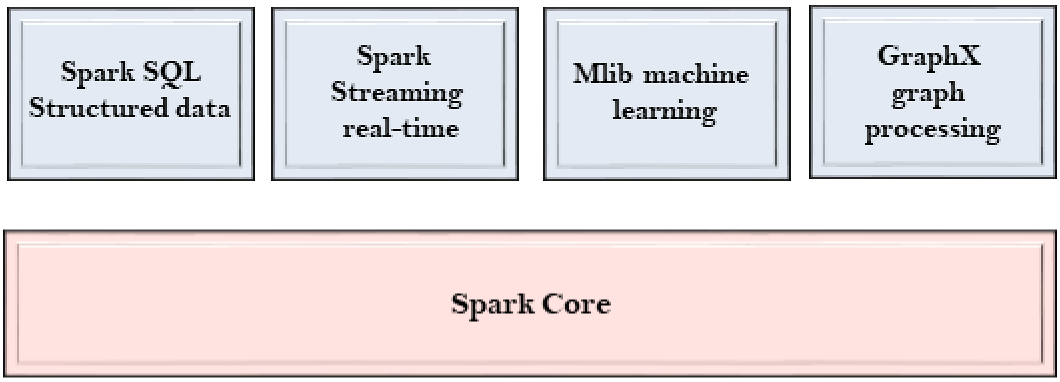
In this project, we have built a tool for search trend analysis:

- Our system first crawls the search trend data (over 800GB) from multiple sources (such as, https://trends.google.com, https://www.bing.com, etc) using **Kafka** streaming.
- The data is then pre-processed by **PyMongo** and stored in the **MongoDB**.
- Finally, the useful data is queried from the database for visualization and suggestion, with the leveraging of **Pyspark**.
- We have deployed our system on **Docker**, with three Spark nodes (1 master and 2 workers), three MongoDB nodes, and three Kafka streaming nodes.
- With our system, users can visualize the top trending searches, estimate the average cost for an advertising campaign, and find the best keywords for an interested topic in terms of the search volume, competition of keyword, average CPC, and recent search trend.

Chapter 2: Apache Spark

Apache Spark is a lightning-fast cluster computing technology, designed for fast com- putation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application. At a fundamental level, an Apache Spark application consists of two main components: a driver, which converts the user's code into multiple tasks that can be distributed across worker nodes, and executors, which run on those nodes and execute the tasks assigned to them. Some form of cluster manager is necessary to mediate between the two.

The Spark project consists of different types of tightly integrated components. At its core, Spark is a computational engine that can schedule, distribute and monitor multiple applications.



2.1 Spark Core

- The Spark Core is the heart of Spark and performs the core functionality.
- It holds the components for task scheduling, fault recovery, interacting with storage systems and memory management.

2.2 Spark SQL

- The Spark SQL is built on the top of Spark Core. It provides support for structured data.
- It allows to query the data via SQL (Structured Query Language) as well as the Apache Hive variant of SQL?called the HQL (Hive Query Language).
- It supports JDBC and ODBC connections that establish a relation between Java objects and existing databases, data warehouses and business intelligence tools.
- It also supports various sources of data like Hive tables, Parquet, and JSON.

2.3 Spark Streaming

- Spark Streaming is a Spark component that supports scalable and fault-tolerant processing of streaming data.
- It uses Spark Core's fast scheduling capability to perform streaming analytics.
- It accepts data in mini-batches and performs RDD transformations on that data.
- Its design ensures that the applications written for streaming data can be reused to analyze batches of historical data with little modification.
- The log files generated by web servers can be considered as a real-time example of a data stream.

2.4 MLlib

- The MLlib is a Machine Learning library that contains various machine learning algorithms.
- These include correlations and hypothesis testing, classification and regression, clustering, and principal component analysis.
- It is nine times faster than the disk-based implementation used by Apache Mahout.

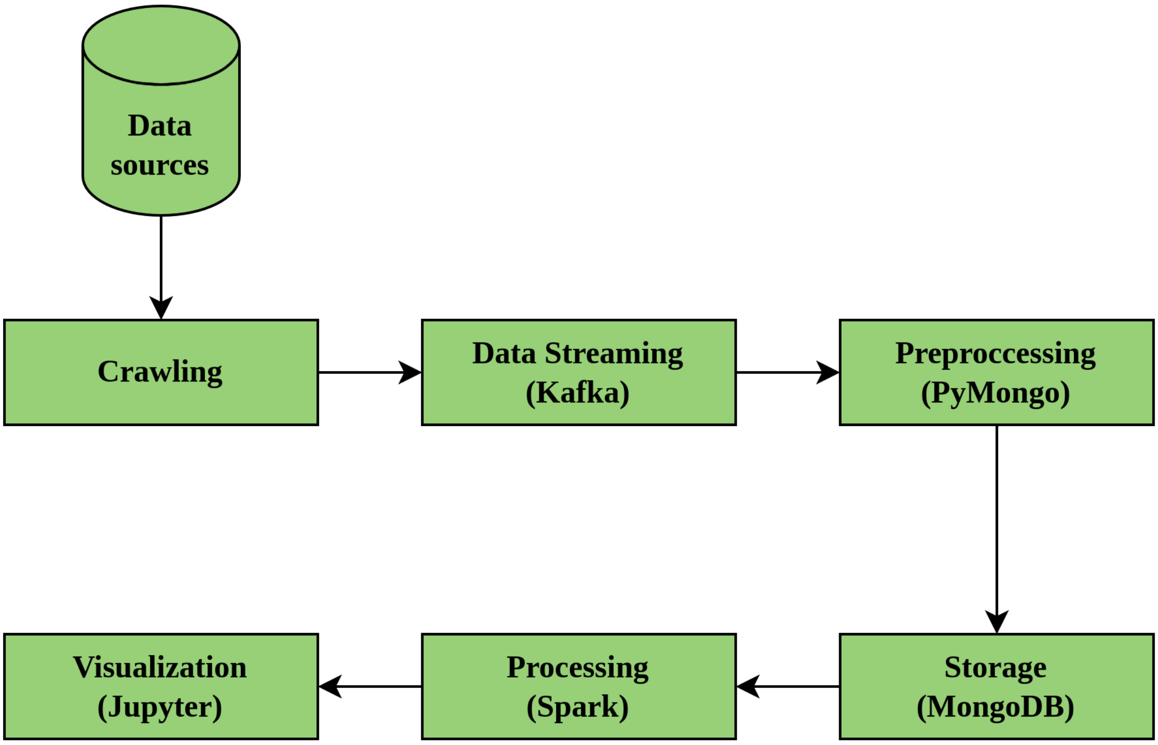
2.5 GraphX

- The GraphX is a library that is used to manipulate graphs and perform graph-parallel computations.
- It facilitates to create a directed graph with arbitrary properties attached to each vertex and edge.
- To manipulate graph, it supports various fundamental operators like subgraph, join Vertices, and aggregate Messages.

Chapter 3: Program Architecture

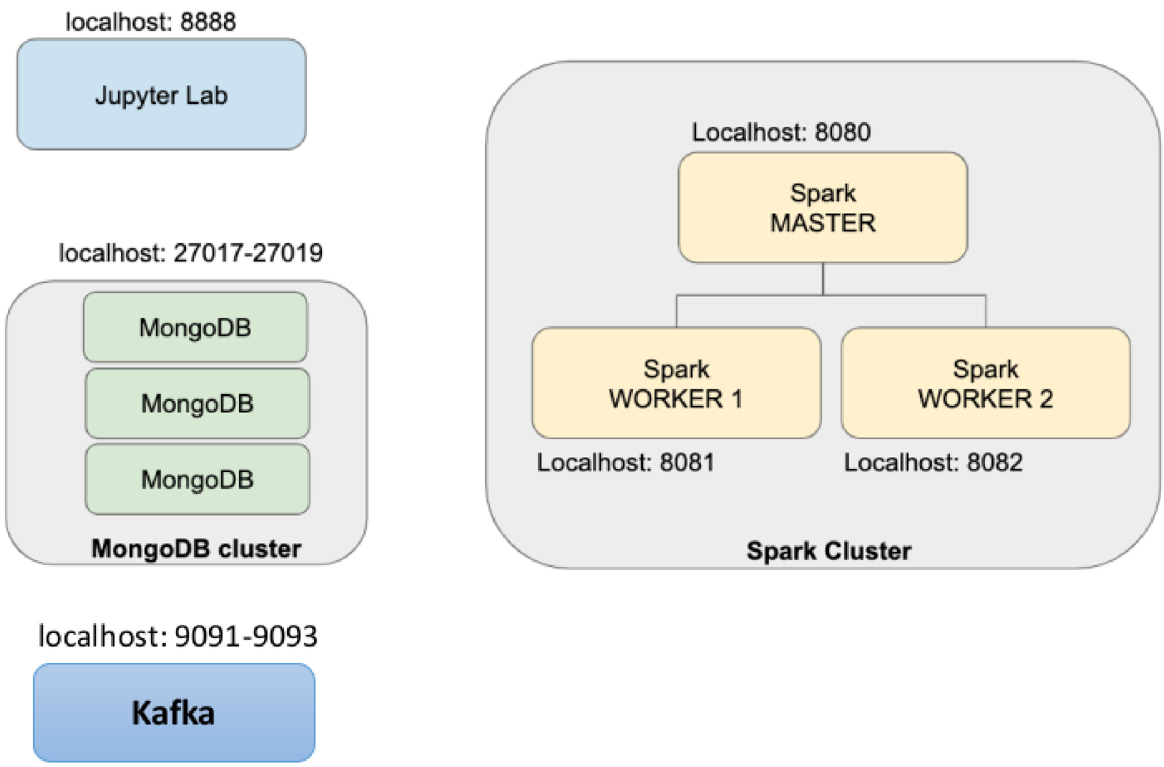
3.1 Data model

In our data model, the search trend data (about 800BG) is first crawled from multiple sources (such as, https://trends.google.com, https://www.bing.com, etc) using Kafka streaming. The data is then pe-processed by PyMongo and stored in the MongoDB. Finally, the useful data is queried from the database for visualization and suggestion, with the leveraging of Pyspark:



3.2 Overview of program architecture

Our system is deployed on Docker, with three Spark nodes (1 master and 2 workers), three MongoDB nodes, and three Kafka streaming nodes. The analysis, visualization and suggestion are executed on jupyter notebook, with the leveraging of Pyspark:



3.4 Data processing and storage

The table below is the data obtained after the preprocessing. The data is stored in MongoDB for later uses.

_id	id of the record in MongoDB
keywords	The keyword
search_vol	The average number of searches of the keyword
competition	The competition level of the keyword. competition = 1, 2, 3 indicate that the competition level is high, medium, and low, respectively
low_CPC	The lowest cost (1E-11 USD) per click that a customer pay for the keyword in the past
high_CPC	The highest cost (1E-11 USD) per click that a customer is willing to pay for the keyword in the past

trend	The search volume of the keyword in the previous 12 months, which indicate the search trend

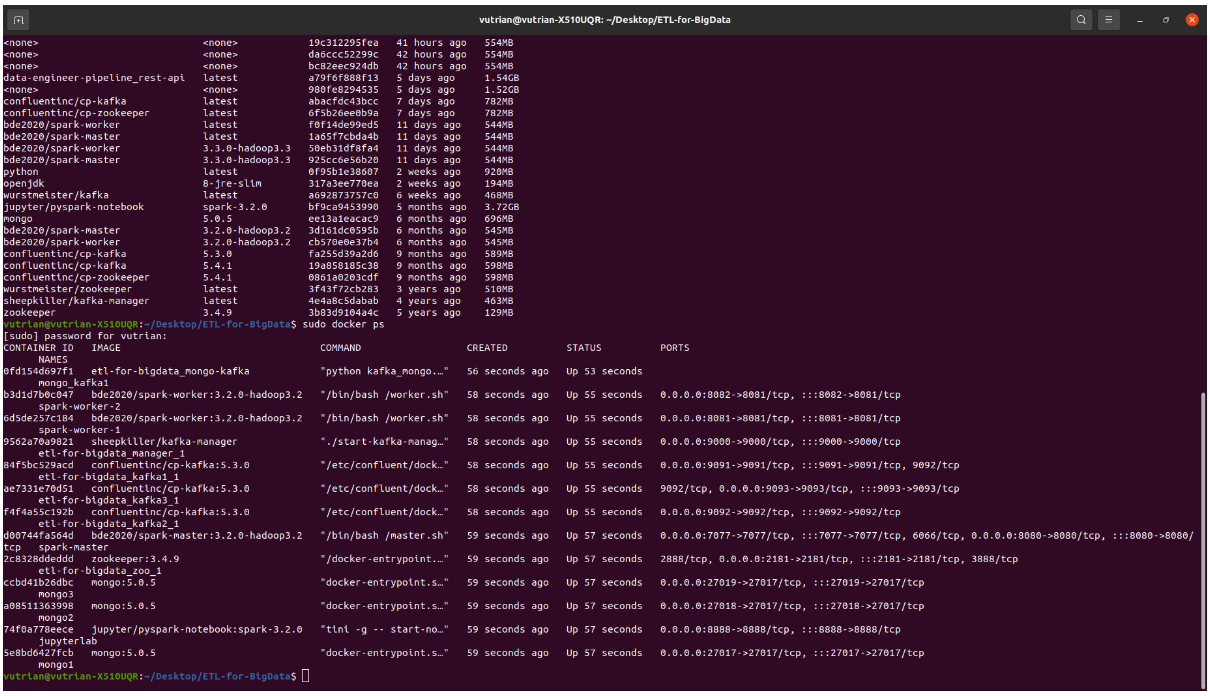
Chapter 4: Configurations and Demonstrations

First, we execute the **run.sh** script file.

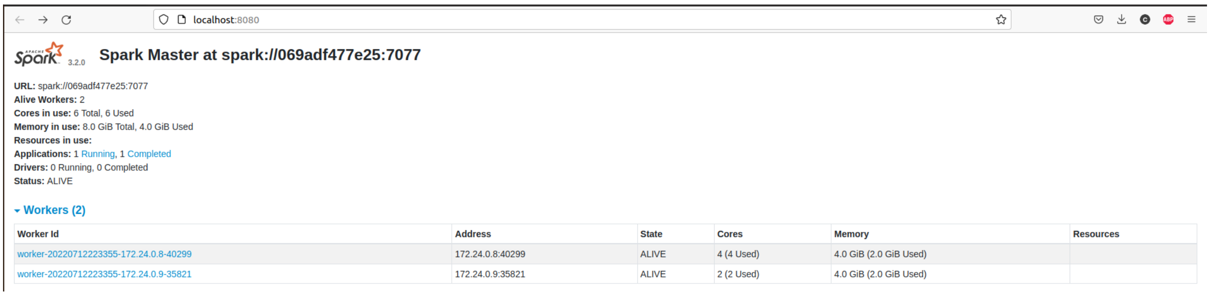
- This runs the docker compose file which creates three nodes of MongoDB, configures it as a replica set on port 27017.
- Spark is also deployed in this environment with a master node located at port 8080 and two worker nodes listening on ports 8081 and 8082 respectively.

The MongoDB cluster will be used for both writing data from PyMongo into MongoDB and reading data into Spark for visualization.

The Jupyter notebook URL which includes its access token will be listed at the end of the script. This token will be generated when you run the docker image so it will be different for you. Here is what it looks like:

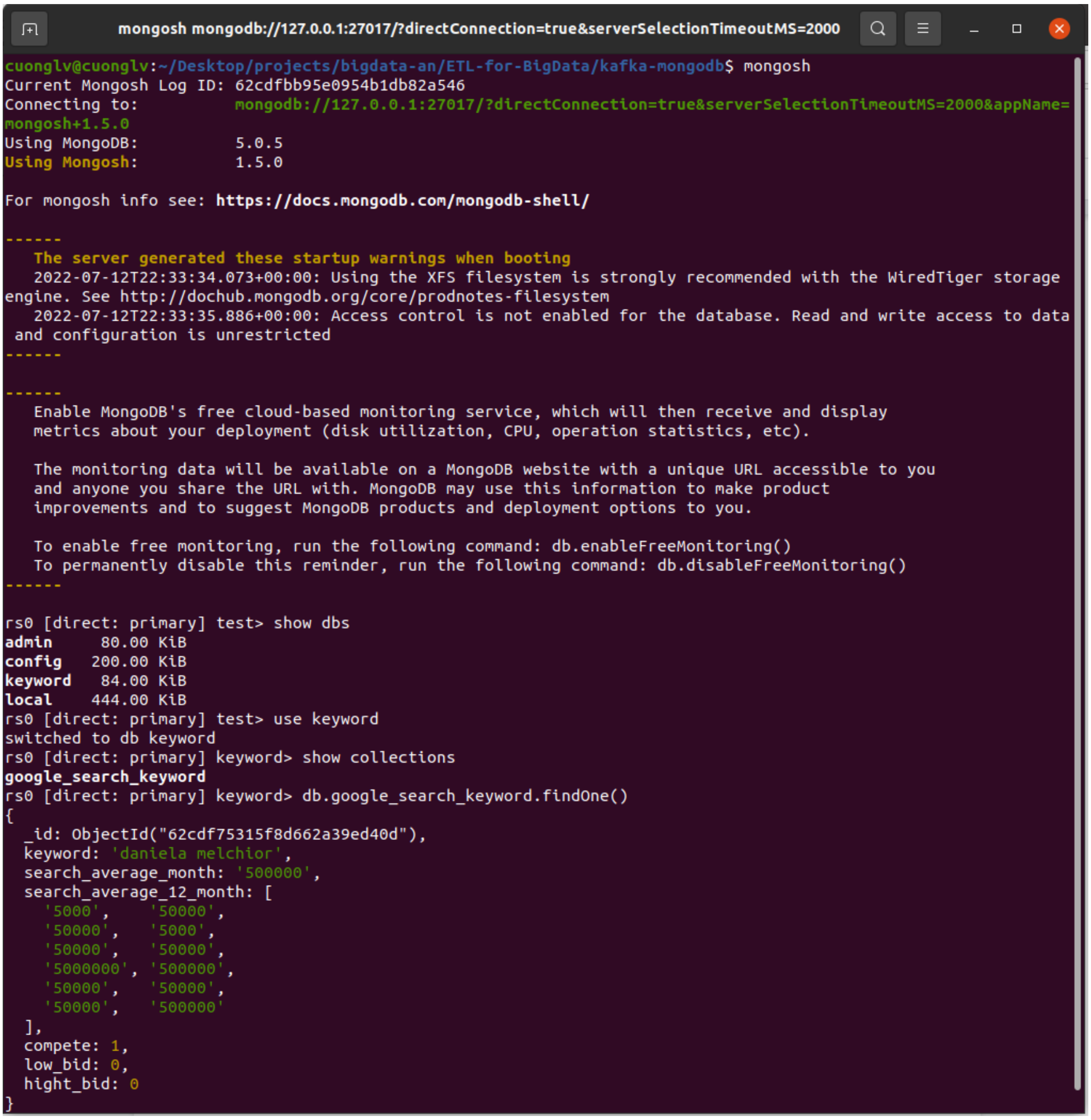


To verify our Spark master and works are online navigate to http://localhost:8080. Here is what it looks like:



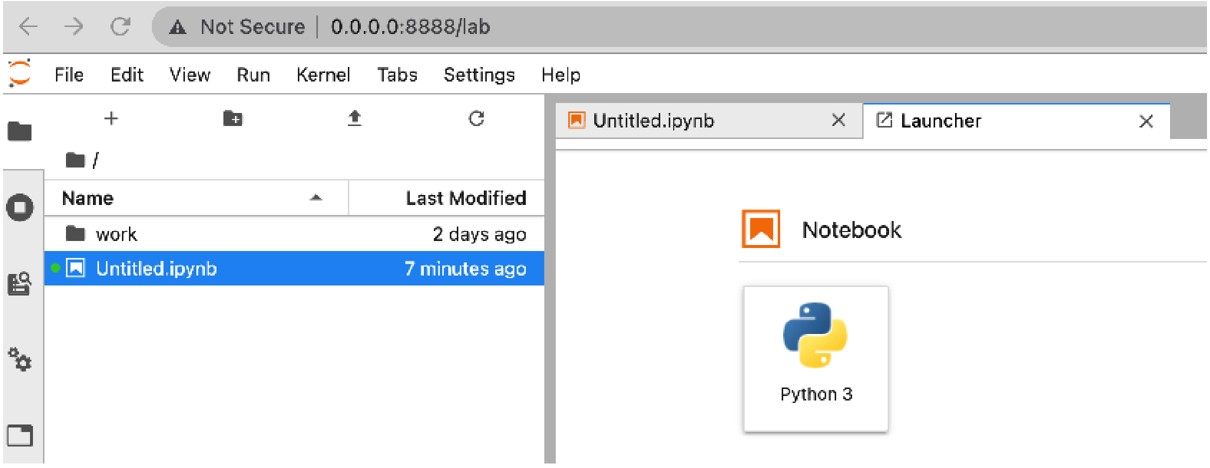
After creating the MongoDB cluster, we run the ***publisher.py*** script in the folder **kafka-mongodb** to stream data from Apache Kafka. As data arrives it's run through PyMongo with the message contents being parsed, transformed, and written into MongoDB.

Now you can use mongosh to check the data was write to MongoDB successfully:



```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
cuonglv@cuonglv:~/Desktop/projects/bigdata-an/ETL-for-BigData/kafka-mongodb$ mongosh
Current Mongosh Log ID: 62cdfbb95e0954b1db82a546
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=
mongosh+1.5.0
Using MongoDB:          5.0.5
Using Mongosh:          1.5.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2022-07-12T22:33:34.073+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage
engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2022-07-12T22:33:35.886+00:00: Access control is not enabled for the database. Read and write access to data
 and configuration is unrestricted
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

rs0 [direct: primary] test> show dbs
admin     80.00 KiB
config   200.00 KiB
keyword   84.00 KiB
local    444.00 KiB
rs0 [direct: primary] test> use keyword
switched to db keyword
rs0 [direct: primary] keyword> show collections
google_search_keyword
rs0 [direct: primary] keyword> db.google_search_keyword.findOne()
{
  _id: ObjectId("62cdf75315f8d662a39ed40d"),
  keyword: 'daniela melchior',
  search_average_month: '500000',
  search_average_12_month: [
    '5000',    '50000',
    '50000',   '5000',
    '50000',   '50000',
    '5000000', '500000',
    '50000',   '50000',
    '50000',   '500000'
  ],
  compete: 1,
  low_bid: 0,
  hight_bid: 0
}
```

To use MongoDB data with Spark, create a new Python Jupyter notebook by navigating to the Jupyter URL and under notebook select Python 3 :



To start, we create the SparkSession and set the environment to use our local MongoDB cluster. When the SparkSession is created successfully, we load the DataFrame from MongoDB and verify the data was loaded by looking at the schema:

```
spark = SparkSession.\
    builder.\
    appName("bigdata-pyspark").\
    master("spark://069adf477e25:7077").\
    config("spark.executor.memory", "2g").\
    config("spark.mongodb.input.uri","mongodb://mongo1:27017,mongo2:27018,mongo3:27019/keyword.google_search_keyword?replicaSet=rs0").\
    config("spark.mongodb.output.uri","mongodb://mongo1:27017,mongo2:27018,mongo3:27019/keyword.google_search_keyword?replicaSet=rs0").\
    config("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector_2.12:3.0.0").\
    getOrCreate()
```

```
df = spark.read.format("mongo").load()
df.printSchema()
# df.show()
```

```
root
 |-- competition: integer (nullable = true)
 |-- high_CPC: double (nullable = true)
 |-- keywords: string (nullable = true)
 |-- low_CPC: double (nullable = true)
 |-- trend: array (nullable = true)
 |    |-- element: integer (containsNull = true)
 |-- search_vol: integer (nullable = true)
 |-- search_vol_level: integer (nullable = true)
```

To verify our application is running, navigate back to http://localhost:8080 and find the application name ‘**bigdata-pyspark**’. Here is what it looks like:

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220712223355-172.24.0.8-40299	172.24.0.8:40299	ALIVE	4 (4 Used)	4.0 GiB (2.0 GiB Used)	
worker-20220712223355-172.24.0.9-35821	172.24.0.9:35821	ALIVE	2 (2 Used)	4.0 GiB (2.0 GiB Used)	

Running Applications (1)

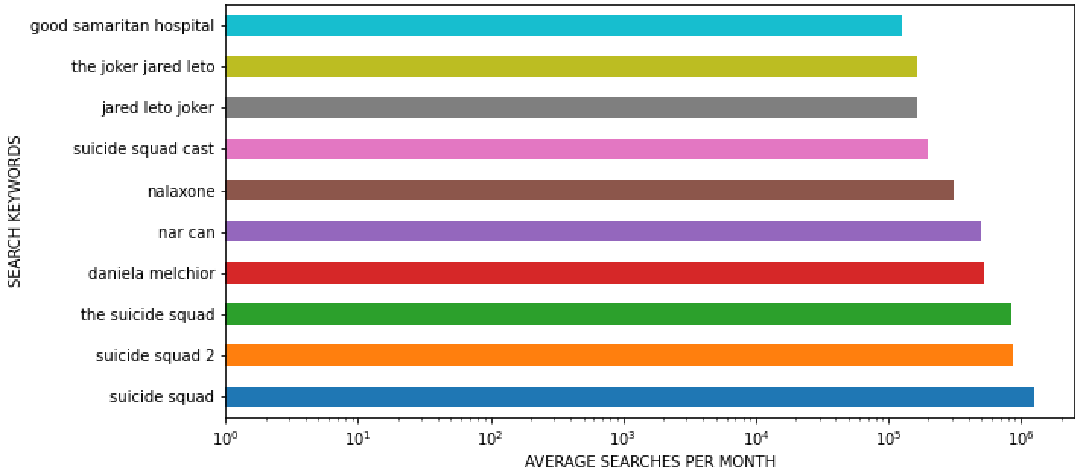
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20220712223858-0001 (kill)	bigdata-pyspark	6	2.0 GiB		2022/07/12 22:38:58	jovyan	RUNNING	59 s

Now, we can extract useful information from the data frame for the visualization. In this project, we have conducted three visualizations, including:

- Display **top trending** searches
- Find the **relationship between the search volume and the lowest cost per click (CPC)**.
- We build a **keyword suggestion tool** that help user to find the best search keyword from the related topic

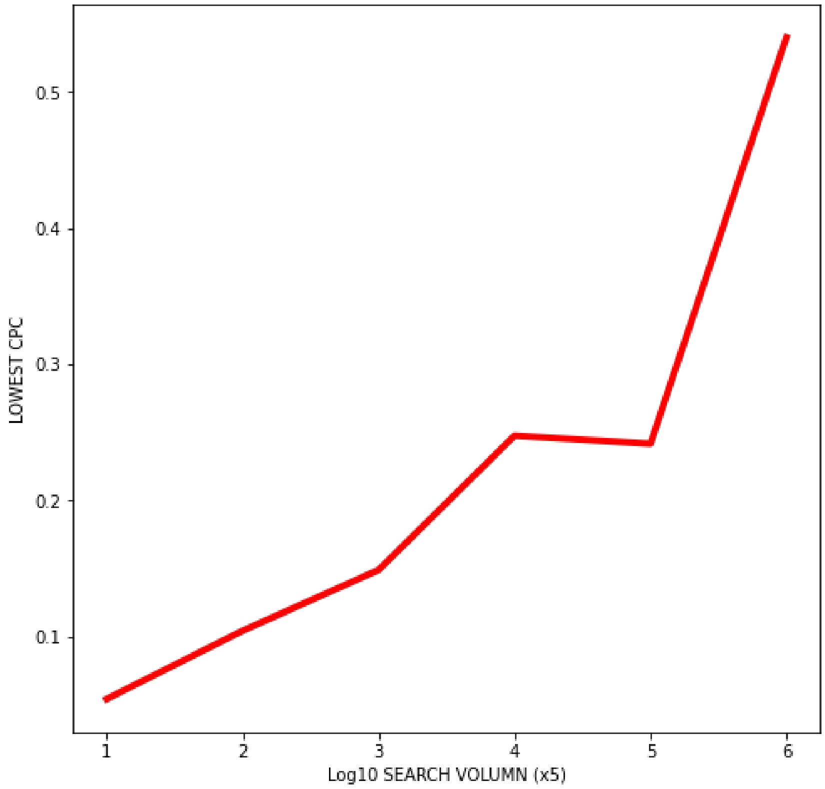
Top 10 Trending Searches

```
get_top_trending(limit=10)
```



The Cost For One Click

```
show_CPC()
```



Keyword Suggestion Tool

```
keyword = "Spark"

get_suggestion(keyword, orderBy='search_vol')
```

Found	Total search volume	Average CPC (USD)
9	1665	0.425556

Keywords	Search Volume	Competition	Average CPC (USD)	Average trend
glossier cloud paint spark	500	Low	0.190000	0.0%
cloud paint spark	462	Low	1.080000	+8.23%
glossier spark cloud paint	275	Low	2.140000	-81.82%
organys spark rejuvenating eye formula	237	Low	0.270000	-78.9%
organys spark eye cream	50	Low	0.000000	0.0%
revlon colorstay skinny liquid eyeliner green spark	50	Low	0.150000	0.0%
becca spark the light kit	33	None	0.000000	-100.0%
becca spark the light	29	None	0.000000	-100.0%
blush spark ar	29	None	0.000000	-100.0%

Conclusion

Search trend analysis is important and helpful for marketers to collect the most popular trends over the internet and make decisions for a marketing campaign. In this project, we successfully built a system for Search trend analysis. The system leverages over 800GB data of search keywords and the advanced technologies, such as Kafka, PyMongo, MongoDB, Pyspark, etc. With our system, users can visualize the top trending search, estimate the average Ads cost needed to pay for on click of customer, and find best-related keywords (in terms of search volume, competition of keywords, average CPC, recent search trend) for an interested topic.