ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO MÔN HỌC
LƯU TRỮ VÀ XỬ LÝ DỮ LIỆU LỚN

Đề tài

LƯU TRỮ, XỬ LÝ VÀ PHÂN TÍCH DỮ LIỆU BẤT ĐỘNG SẢN

Sinh viên: Nguyễn Trung Hải – 20204545
Trần Quang Đạo – 20200128
Đinh Ngọc Quân – 20204849
Đỗ Hoàng Dường – 20204732
Tống Quang Huy – 20204756
GVHD: TS. Trần Việt Trung

Hà Nội, Ngày 5 tháng 1 năm 2024

MỞ ĐẦU

Trong những năm nay gần đây, lĩnh vực công nghệ thông nói chung đã và đang trải qua một sự tiến hóa mạnh mẽ, nhanh chóng hơn bao giờ hết, đặc biệt là sự bùng nổ về các trang thông tin, các trang web, các nền tảng số gia tăng không ngừng. Kéo theo đó là làn sóng dữ liệu lớn chưa từng có về quy mô và số lượng. Việc lưu trữ tốt các dữ liệu lớn đó có thể giúp ích chúng ta rất nhiều trong việc xử lý, phân tích chúng để tìm ra những tri thức, những hiểu biết quan trọng, tiềm năng về một lĩnh vực cụ thể, từ đó có thể đưa ra những quyết định, đánh giá dựa trên việc xử lý, phân tích dữ liệu đó một cách chính xác, phù hợp với thực tế hơn.

Dữ liệu là mỏ vàng của thế giới, là xu thế không thể nghịch chuyển khi bước vào kỷ nguyên số hóa toàn cầu. Tuy nhiên dữ liệu lớn là các tập dữ liệu rất lớn và phức tạp, rất khó để quản lý, lưu trữ và phân tích bằng các công cụ xử lý dữ liệu truyền thống. Để giải quyết vấn đề đó đã có rất nhiều giải pháp để giải quyết vấn đề này. Nhận thấy được tầm quan trọng và tiềm năng to lớn của việc lưu trữ, xử lý dữ liệu lớn, nhóm chúng em quyết định thực hiện đề tài: “Lưu trữ, xử lý và phân tích dữ liệu thị trường bất động sản”.

Đối với đề tài này, mục tiêu của nhóm là xây dựng 1 hệ thống tự động giúp con người dễ dàng tìm kiếm thông tin mong muốn về thị trường bất động sản, từ đó có những hiểu biết nhất định về thị trường bất động sản và đưa ra các quyết định đúng đắn. Trong báo cáo này, chúng em trình bày phương pháp tiếp cận và quá trình thực hiện theo các phần chính sau:

1. **Giới thiệu bài toán**.
2. **Công nghệ sử dụng**.
3. **Triển khai hệ thống**.
4. **Trải nghiệm khi cài đặt hệ thống**.

Nhóm chúng em hy vọng qua đề tài này có thể giúp ích trong việc lưu trữ xử lý dữ liệu. Trong quá trình triển khai thực hiện đề tài này, nhóm chúng em xin gửi lời cảm ơn chân thành tới giảng viên hướng dẫn TS. Trần Việt Trung, nhờ những giờ giảng dạy trên lớp, và những góp ý sát thực của thầy đã giúp nhóm chúng em hoàn thiện đề tài này một cách đầy đủ và thực tế hơn.

MỤC LỤC

DANH MỤC HÌNH VẼ

1. Giới thiệu bài toán

Thị trường bất động sản là một thị trường quan trọng trong nền kinh tế của bất kỳ quốc gia nào. Tại Việt Nam, thị trường bất động sản luôn là một trong những thị trường nóng nhất, thu hút sự quan tâm của nhiều người, bao gồm các nhà đầu tư, người mua nhà, các nhà quản lý nhà nước và các nhà nghiên cứu.

Đặc biệt, thị trường bất động sản ở Hà Nội là một thị trường lớn và phức tạp, với nhiều loại hình bất động sản khác nhau, bao gồm nhà ở, đất nền, văn phòng, nhà xưởng, ... Dữ liệu thị trường bất động sản ở Hà Nội có thể được thu thập từ nhiều nguồn khác nhau, bao gồm các cơ quan nhà nước, các công ty bất động sản, các trang web bất động sản và các nguồn dữ liệu trực tuyến khác. Đây luôn là một lĩnh vực nóng, nhận được sự quan tâm của mọi người với nhiều mục đích khác nhau.

Bên cạnh đó, thông tin về các các dự án bất động sản hiện được phân tán rộng khắp trên internet. Mỗi trang web có thể nắm giữ một lượng lớn bài đăng, nhưng hiện các nguồn tổng hợp và cập nhật tất cả dữ liệu này còn hạn chế. Người quan tâm cần phải sử dụng nhiều công cụ tìm kiếm, phân tích thủ công để hiểu rõ về tình hình thị trường bất động sản.

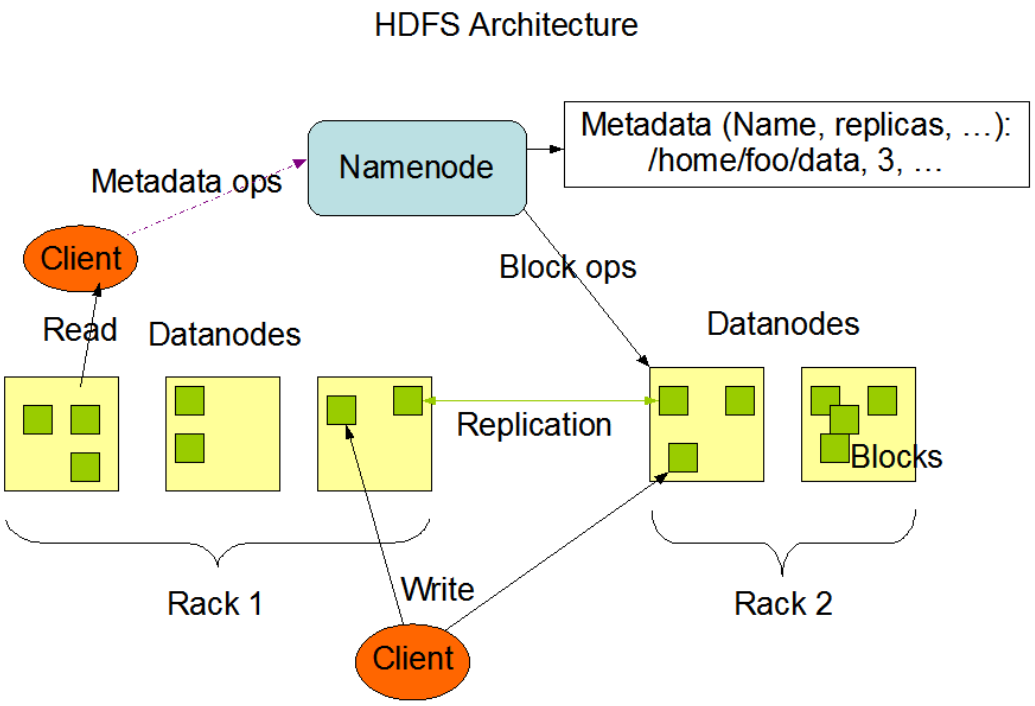
Do vậy, để giúp các nhà đầu tư, người mua bán nhà, các nhà quản lý trong thị trường bất động sản ở Hà Nội một cách hiệu quả, nhóm quyết định xây dựng một hệ thống lưu trữ và phân tích dữ liệu về thị trường bất động sản tại Hà Nội, từ đó có thể hỗ trợ:

Giúp các nhà đầu tư, mua bán nhà đất có được thông tin đầy đủ và chính xác về thị trường bất động sản ở Hà Nội, từ đó đưa ra các quyết định đúng đắn, hiệu quả.

Hỗ trợ các nhà quản lý nhà nước có bức tranh tổng quan về thị trường bất động sản, từ đó có thể đưa ra các quy hoạch tiếp theo phù hợp hơn.

Từ đó, nhóng nhóm đề xuất thực hiện ứng dụng công nghệ dữ liệu lớn nhằm tự động hóa:

- Thu thập dữ liệu các bài đăng mua bán bất động sản từ internet (cập nhật liên tục).
- Tổ chức lưu trữ và xử lý lượng lớn dữ liệu.
- Thực hiện phân tích dữ liệu và hỗ trợ tìm kiếm bất động sản theo yêu cầu.



Hình 2.1 Kiến trúc HDFS

2. Công nghệ sử dụng

2.1 Scrapy và Playwright

Scrappy là một open-source web crawling framework của Python được sử dụng nhằm thu thập dữ liệu từ các website. Có kiến trúc linh hoạt, dễ dàng thực hiện các công việc: điều hướng qua các trang web, thu thập dữ liệu, xử lý và lưu dữ liệu. **Playwright** là một open-source framework giúp kiểm thử và tự động hóa web. Được sử dụng cùng Scrapy để cung cấp khả năng tương tác với các trang web có chứa mã JavaScript. Sự kết hợp của Scrapy và Playwright cung cấp giải pháp đa dạng và hiệu quả cho việc thu thập và xử lý dữ liệu từ các nguồn website khác nhau.

2.2 HDFS

Hadoop Distributed File System (HDFS) là một hệ thống file phân tán được thiết kế để chạy trên phần cứng thông thường. HDFS cũng tương tự những hệ thống file phân tán hiện có. Tuy nhiên, sự khác biệt ở đây là HDFS có khả năng chịu lỗi cao (fault-tolerant) và được thiết kế để deploy trên các phần cứng rẻ tiền. HDFS cung cấp khả năng truy cập high throughput từ ứng dụng và thích hợp với các ứng dụng có tập dữ liệu lớn.

HDFS có kiến trúc master-worker (Hình 2.1). Một cụm HDFS (HDFS cluster) bao gồm các Namenode và Datanode. Dữ liệu được lưu trên các block. Một cụm HDFS bao

gồm hai loại nút (node) hoạt động theo mô hình nút chủ - nút thợ (master-worker):

- Một cụm HDFS có một namenode (master – nút chủ).
- Một cụm HDFS có một hoặc nhiều các datanode (worker - nút thợ).

Namenode quản lý các namespace filesystem. Nó quản lý một filesystem tree và các metadata cho tất cả các file và thư mục trên tree. Thông tin này được lưu trữ trên đĩa vật lý dưới dạng không gian tên ảnh và nhật ký (edit log). Namenode còn quản lý thông tin các khối (block) của một tập tin được lưu trên những datanodes nào.

2.3 Apache Spark

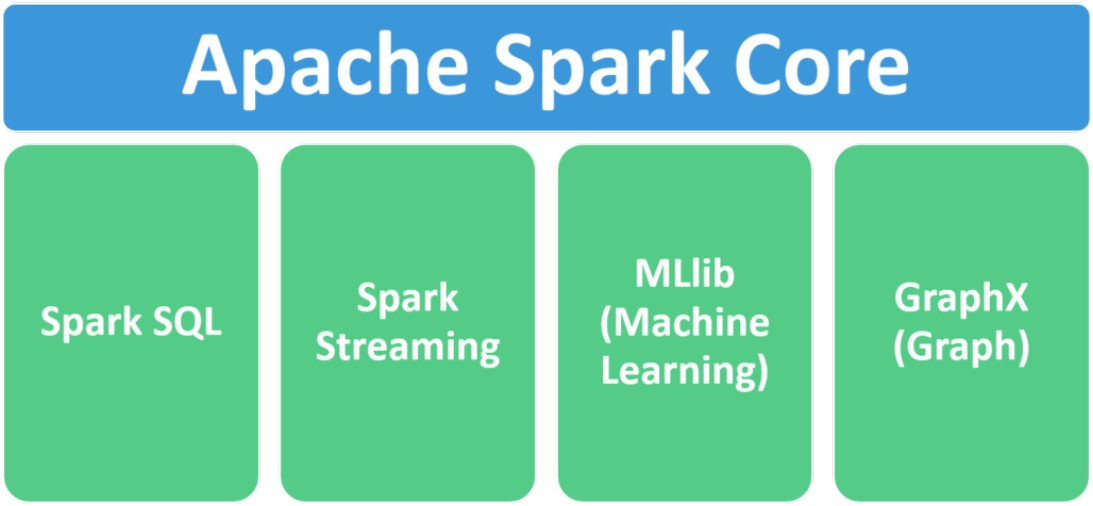
Apache Spark là một ứng dụng mã nguồn mở được xây dựng được xây dựng để xử lý dữ liệu phân tán, nhằm tăng tốc độ xử lý, dễ sử dụng và linh hoạt. Sử dụng để xử lý dữ liệu lớn một cách nhanh chóng, bằng cách cho phép thực hiện tính toán trên cụm tạo ra khả năng phân tích dữ liệu tốc độ cao khi đọc và ghi dữ liệu. Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM. Spark hỗ trợ nhiều ngôn ngữ lập trình được sử dụng rộng rãi (Python, Java, Scala và R), bao gồm các thư viện cho các tác vụ đa dạng khác nhau, từ SQL đến phát trực tuyến và học máy, và chạy ở mọi nơi từ máy tính xách tay đến một cụm hàng nghìn máy chủ. Điều này hỗ trợ cho Spark trở thành một hệ thống dễ dàng bắt đầu và mở rộng quy mô để xử lý dữ liệu lớn hoặc quy mô cực kỳ lớn.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được (Spark Streaming).

Spark tuân theo kiến trúc Master-Slave nên một ứng dụng Spark đều có một chương trình trình điều khiển (driver program) và nhiều trình thực thi (executors). Trình điều khiển giúp chuyển đổi chương trình người dùng thành các tác vụ sau đó lên lịch các tác vụ trên các trình thực thi. Trình thực thị chịu trách nghiệm chạy tác tác vụ riêng lẻ trong một tác vụ Spark nhất định và gửi kết quả cho trình điều khiển khi chạy xong.

Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3,.... Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json...) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào.

Để chạy nhanh hơn, Spark cung cấp: Mô hình tối ưu các tính toán đồ thị một cách tùy ý (optimize arbitrary operator graphs). Hỗ trợ tính toán tại bộ nhớ trong. Spark cung cấp bộ API hỗ trợ các ngôn ngữ Scalar, Java, Python. Spark 22 hỗ trợ các thư viện ứng dụng cơ bản của học máy như Rừng ngẫu nhiên (Random Forest), cây quyết định (Decision Tree)... hay các thư viện phân cụm (KMeans).



2.3.1 Các thành phần của Spark

1. **Spark Core**: là lõi ứng dụng (engine) thực thi chung làm nền tảng cho Spark. Tất cả các chức năng khác được xây dựng dựa trên nền tảng là Spark Core. Cung cấp khả năng tính toán trên bộ nhớ RAM và cả bộ dữ liệu tham chiếu trong các hệ thống cho phép mở rộng bộ nhớ vật lý (external storage).

2. **Spark SQL**: là một thành phần nằm trên Spark Core, giới thiệu một khái niệm trừu tượng hóa dữ liệu mới gọi là SchemaRDD, cung cấp hỗ trợ cho dữ liệu có cấu trúc và bán cấu trúc.

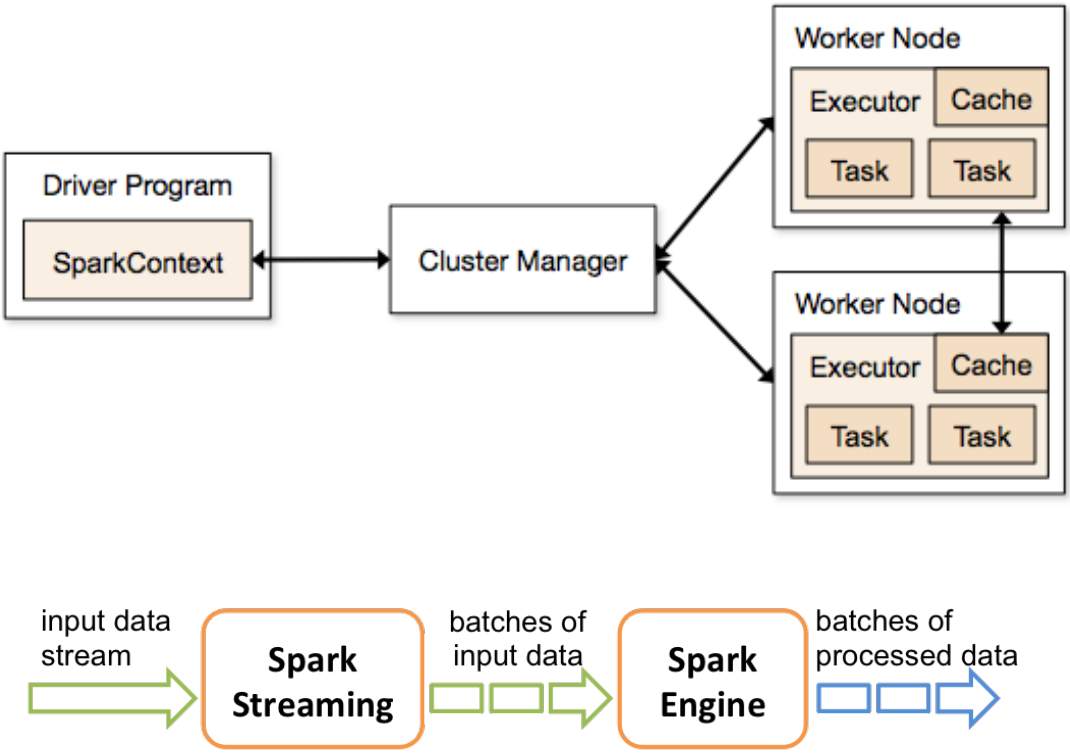
3. **Spark Streaming**: tận dụng khả năng lập lịch memory-base của Spark Core để thực hiện streaming analytics. Nó lấy dữ liệu theo mini-batches và thực hiện các phép biến đổi RDD (Bộ dữ liệu phân tán có khả năng phục hồi) trên các mini-batches dữ liệu đó.

4. **MLlib (Machine Learning Library)**: là một framework machine learning phân tán trên Spark tận dụng khả năng tính toán tốc độ cao nhờ distributed memory-based của kiến trúc Spark.

5. **GraphX**: là một framework xử lý đồ thị phân tán. Nó cung cấp một API để thực hiện tính toán biểu đồ có thể mô hình hóa các biểu đồ do người dùng xác định bằng cách sử dụng API đã được tối ưu sẵn.

2.3.2 Thực thi

Chương trình Spark chạy như một bộ tiến trình độc lập trên mỗi cluster. Các tiến trình này được điều khiển bởi SparkContext trong chương trình điều khiển (Driver program), SparkContext sẽ kết nối với một số loại Cluster Manager (các trình quản lý cụm chạy standalone của Spark hoặc YARN hoặc MESOS) trình quản lý việc phân bố tài



Hình 2.2 Cơ chế Spark streaming

nguyên cho các ứng dụng để xác định các nút sẽ làm việc. Sau đó, Spark sẽ kết nối tới một số Executor trên các nút này (thực chất là các tiến trình chạy các tác vụ tính toán, lưu trữ dữ liệu cho ứng dụng), sau đó sẽ gửi mã của ứng dụng (được gửi từ SparkContext) tới các Executor này. Cuối cùng SparkContext sẽ gửi các tác vụ tới các Executor để chạy (Hình 3.3).

2.3.3 Spark Streaming

Spark Streaming là một phần mở rộng của API Spark cho có khả năng mở rộng, thông lượng cao, xử lý luồng có khả năng chịu lỗi của luồng dữ liệu trực tiếp. Dữ liệu có thể được nhập từ nhiều nguồn như các socket Kafka, Kinesis hoặc TCP và có thể được xử lý bằng cách sử dụng phức tạp các thuật toán được thể hiện bằng các hàm cấp cao. Cuối cùng, dữ liệu được xử lý có thể được đẩy ra hệ thống tệp, cơ sở dữ liệu, và bảng điều khiển trực tiếp. Trên thực tế, có thể áp dụng các thuật toán học máy và xử lý đồ thị của Spark trên các luồng dữ liệu này.

Thực tế, Spark Streaming hoạt động như sau: nó nhận các luồng dữ liệu đầu vào trực tiếp và phân chia dữ liệu thành các lô, sau đó được xử lý bởi Spark Engine để tạo ra luồng kết quả theo đợt (Hình 2.2).

Spark Streaming cung cấp một khái niệm trừu tượng cấp cao được gọi là discretized stream hoặc DStream, đại diện cho một luồng dữ liệu liên tục, hoặc luồng dữ liệu đầu vào

nhận được từ nguồn, hoặc luồng dữ liệu đã xử lý được tạo bằng cách chuyển đổi luồng đầu vào. Một DStream được đại diện bởi các RDD liên tục, mỗi RDD trong DStream chữa dữ liệu trong khoảng thời gian nhất định. Bất kì những hoạt động nào áp dụng trên DStream đều chuyển thành các hoạt động trong RDD cơ bản.

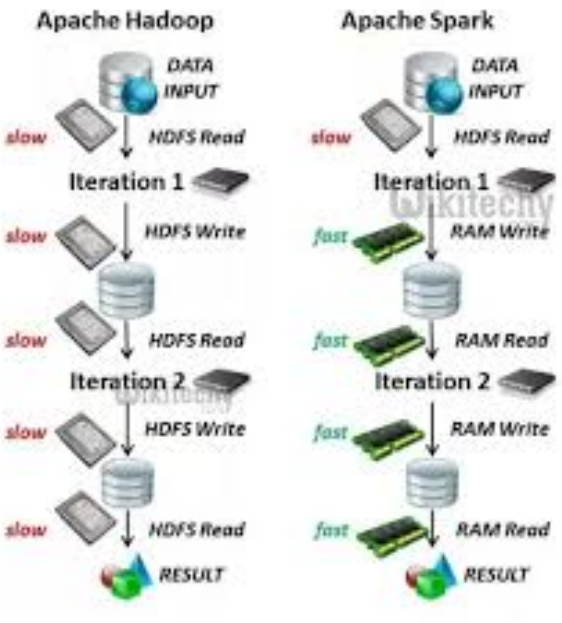
2.3.4 Spark vs Hadoop MapReduce

Về cơ chế hoạt động của Map-Reduce: dữ liệu đầu vào được đọc từ HDFS (ứng dụng phụ trách việc lưu trữ trong Hadoop), xử lý bằng các thao tác chỉ định, dữ liệu đầu ra được ghi vào HDFS, dữ liệu tiếp tục được đưa lên, thao tác tiếp theo được thực hiện, dữ liệu đầu ra tiếp tục ghi vào HDFS … chuỗi các bước [đọc - xử lý - ghi] đó được lặp cho đến khi hoàn thành tác vụ.

Vì dữ liệu đầu vào được chia thành các khối (block) độc lập với nhau, các tiến trình map-reduce được thực hiện song song, nên về cơ bản nó hữu ích để xử lí những bộ dữ liệu lớn. Tuy nhiên, map-reduce vẫn còn những tồn tại là quá trình xử lý không thực sự hiệu quả trong trường hợp phải lặp lại nhiều bước, vì mỗi bước cần thiết phải ghi đầu ra dữ liệu vào HDFS trước khi bước tiếp theo được thực hiện, việc này tạo ra các vấn đề trong việc lưu trữ và cơ chế tạo lặp các vùng lưu trữ, tăng độ trễ xử lý do phần lớn thực hiện trên bộ nhớ ngoài vốn có hiệu suất I/O không cao. Bên cạnh đó là việc thực hiện viết code với Map-Reduce có phần khó khăn vì viết lệnh giao tiếp khá dài dòng.

So với Hadoop, Apache Spark có mô hình Tập dữ liệu phân tán linh hoạt (RDD) và mô hình Đồ thị vòng có hướng (DAG) được xây dựng trên khung tính toán bộ nhớ được hỗ trợ cho Spark. Cho phép lưu trữ một bộ nhớ 21 cache dữ liệu trong bộ nhớ và thực hiện tính toán và lặp lại cho cùng một dữ liệu trực tiếp từ bộ nhớ. Nền tảng Spark tiết kiệm một lượng lớn thời gian hoạt động I / O của đĩa. Do đó, Spark phù hợp hơn cho việc khai thác dữ liệu với tính toán lặp đi lặp lại (Hình 2.3).

Cơ chế hoạt động của Spark khắc phục những tồn tại của Hadoop MapReduce, Spark đưa ra một khái niệm mới RDD - Resilient Distributed Dataset đóng vai trò như một cấu trúc dữ liệu cơ bản trong Spark, RDD được định nghĩa là trừu tượng cho một tập hợp các phần tử bất biến (bản chất là được lưu trên các ô nhớ chỉ đọc readOnly), được phân vùng có thể được chia sẻ, tác động song song. Qua đó, dữ liệu vào từ hệ thống lưu trữ chỉ cần đẩy lên lần duy nhất, các bước thực hiện biến đổi, xử lý dữ liệu đầu vào được lên kế hoạch, tối ưu hóa và thực hiện một cách liên tục cho đến khi dữ liệu đầu ra được trả khi kết thúc công việc. Toàn bộ quá trình đó được diễn ra trên bộ nhớ RAM (khi hết RAM sẽ được chuyển sang xử lý trên Disk) tận dụng được hiệu suất I/O cao từ đó có thể giảm thời gian thực thi nhỏ hơn 10-100 lần Hadoop MapReduce.



Hình 2.3 Spark vs Mapreduce

2.4 Apache Kafka

Apache Kafka là một nền tảng phân phối sự kiện phân tán mã nguồn mở được phát triển bởi Apache Software Foundation và được viết bằng Java và Scala. Kafka được tạo ra để giải quyết những thách thức trong việc xử lý lượng dữ liệu khổng lồ trong thời gian thực (real-time), cho phép các ứng dụng xuất bản (publish), đăng ký (subscribe), lưu trữ (store) và xử lý (process) các luồng bản ghi (streaming event) một cách hiệu quả.

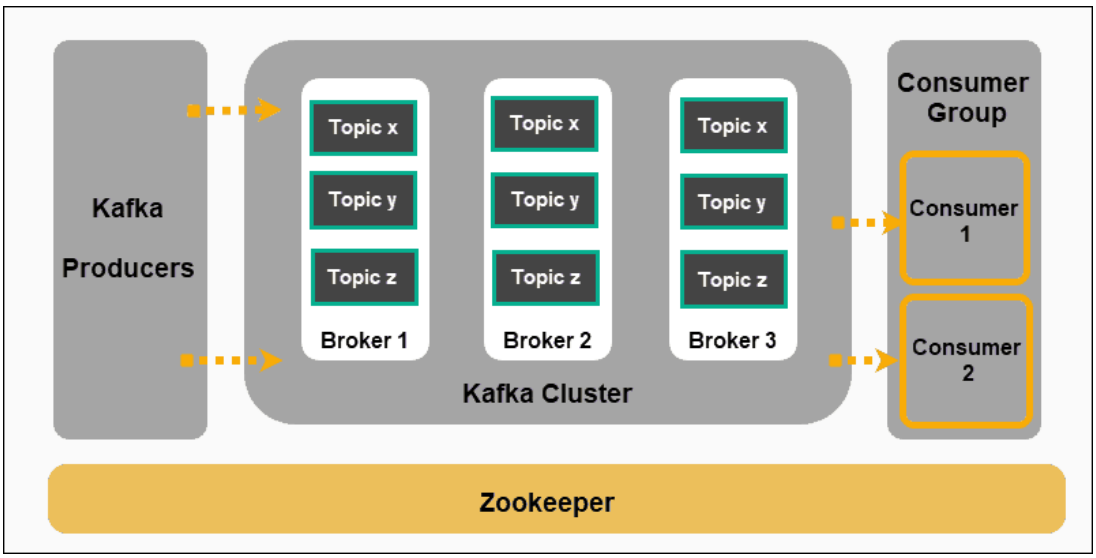
2.4.1 Kafka Events

Một Kafka event (sự kiện) ghi lại thực tế rằng "điều gì đó đã xảy ra" trên thế giới hoặc trong doanh nghiệp của bạn. Nó còn được gọi là record (bản ghi) hoặc message (thông điệp). Trong nhiều tài liệu về Kafka, việc sử dụng 3 thuật ngữ event, record và message mang nghĩa tương đương nhau.

Kafka event/record/message là một đơn vị dữ liệu được sử dụng trong hệ thống Kafka.

2.4.2 Kafka Topics

Các event được tổ chức và lưu trữ lâu dài trong các topics (chủ đề). Có thể coi một topic ví như một thư mục (folder) trong hệ thống tập tin (filesystem), còn mỗi event là một tập tin (file) nằm bên trong thư mục đó.



Hình 2.4 Kiến trúc của Kafka

2.4.3 Kafka Brokers và Kafka Clusters

Kafka được chạy dưới dạng một Kafka cluster gồm một hoặc nhiều Kafka server có thể mở rộng trên nhiều data center hoặc cloud. Một Kafka server này tạo thành lớp lưu trữ, được gọi là Kafka broker.

Brokers chịu trách nhiệm quản lý bộ lưu trữ, xử lý các yêu cầu đọc và ghi cũng như sao chép dữ liệu trên toàn cluster (cụm). Trong mỗi cluster sẽ có một broker sẽ hoạt động như một cluster controller (bộ điều khiển cụm), chịu trách nhiệm chỉ định phân vùng cho brokers và theo dõi lỗi của brokers.

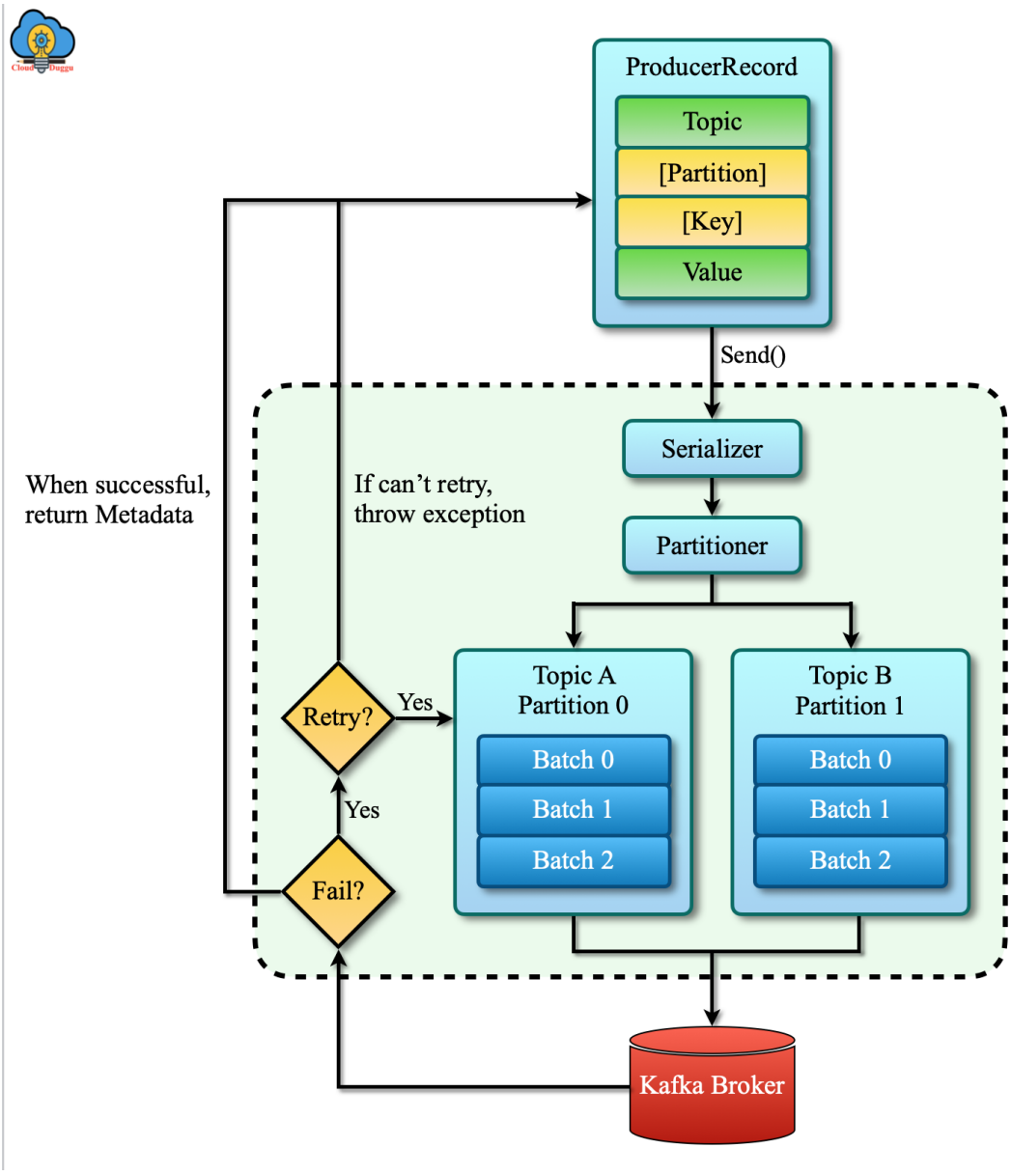
2.4.4 Kafka Producers

Kafka Producer (Hình 2.5) là một client appication (ứng dụng khách), publish (xuất bản) event vào một topic cụ thể trong Kafka và luôn ghi vào leader broker. Theo mặc định, producers không quan tâm tới event được ghi ở partition nào mà sẽ publish đều event trên tất cả partition của một topic. Trong vài trường hợp, một producer sẽ gửi trực tiếp event tới các partition cụ thể.

Producers kết nối tới Kafka Brokers thông qua giao thức mạng TCP. Đây là kết nối hai chiều (bidirectional connection).

2.4.5 Kafka Consumer

Kafka consumer là một client application (ứng dụng khách), subscribe (đăng ký) một hoặc nhiều Kafka topics và đọc các bản ghi theo thứ tự chúng được tạo ra. Consumers đọc dữ liệu theo thời gian thực hoặc theo tốc độ của riêng chúng, cho phép các ứng dụng phản ứng với các sự kiện khi chúng xảy ra.



Hình 2.5 Kiến trúc của Kafka

Consumers kết nối tới Kafka Brokers thông qua giao thức mạng TCP. Đây là kết nối hai chiều (bi-directional connection).

Consumers hoạt động trong một consumer group, làm việc cùng nhau để xử lý dữ liệu từ các partitions, cung cấp khả năng mở rộng theo chiều ngang và cho phép nhiều phiên bản của cùng một ứng dụng xử lý dữ liệu đồng thời.

2.5 Elasticsearch

Elasticsearch là một công cụ tìm kiếm dựa trên phần mềm Lucene. Nó cung cấp một bộ máy tìm kiếm dạng phân tán, có đầy đủ công cụ với một giao diện web HTTP có hỗ trợ dữ liệu JSON. Elasticsearch được phát triển bằng Java và được phát hành dạng nguồn mở theo giấy phép Apache.

Ngoài ra, ES cũng được xem là một document oriented database. Nhiệm vụ chính là **store** và **retrieve** document. Trong ES, tất cả các document được hiển thị trong JSON format. ES được xây dựng trên Lucene – phần mềm tìm kiếm và trả về thông tin với hơn 15 năm kinh nghiệm về full text indexing and searching.

Các đặc điểm cơ bản của Elasticsearch:

- **JSON based data**: hệ thống sử dụng cơ sở dữ liệu noSQL để lưu trữ và truy vấn dữ liệu, tập trung vào việc tối ưu hóa hiệu suất tìm kiếm. Điều này đảm bảo rằng Elasticsearch có thể cung cấp kết quả tìm kiếm chính xác trong thời gian gần như thời gian thực, ngay cả trên tập dữ liệu lớn.
- **RESTful APIs**: Elasticsearch cũng cung cấp một giao diện RESTful API, cho phép các ứng dụng và dịch vụ khác có thể tương tác với nó một cách dễ dàng. Điều này làm cho việc tích hợp Elasticsearch vào các ứng dụng hiện có trở nên rất linh hoạt và thuận tiện.
- **Multi data resources**: Dữ liệu có thể lấy từ nhiều nguồn khác nhau. Đó có thể là Logs từ ứng dụng, Metrics hệ thống hoặc bất nguồn kỳ dữ liệu với bất kỳ loại dữ liệu nào đến từ bất kỳ ứng dụng khác nhau.

2.5.1 Tổ chức lưu trữ dữ liệu

Tổ chức logic

Cấu trúc lưu trữ dữ liệu được tổ chức logic hoá có nét tương đồng với hệ cơ sở dữ liệu quan hệ RDMS với các thành phần như sau:

- **Indexes**: Index trong Elasticsearch là một tập hợp các tài liệu có tính chất tương tự (có cùng các trường hoặc field). Mỗi index có thể được coi như một cơ sở dữ liệu

RDMS	Elasticsearch
DB	Indexes/Indicies
Tables	Patterns/Types
Row	Documents
Columns	Fields

Hình 2.6 RDMS vs Elasticsearch

nhỏ trong hệ thống Elasticsearch. Index được sử dụng để tìm kiếm, lọc và lưu trữ dữ liệu. Một cơ sở dữ liệu có thể bao gồm nhiều index.

- **Documents**: Document là dữ liệu cơ bản mà Elasticsearch lưu trữ và tìm kiếm. Document thường được biểu diễn dưới dạng JSON (JavaScript Object Notation). Ví dụ: Trong một index chứa thông tin người dùng, mỗi người dùng sẽ được biểu diễn bằng một document.

- **Fields**: Field là các thành phần cơ bản của một document. Chúng là các thuộc tính hoặc trường thông tin về một document. Mỗi field có thể chứa một loạt các giá trị như chuỗi ký tự, số, ngày tháng, v.v. Ví dụ: Trong một document người dùng, các field có thể bao gồm "name", "age", "email", v.v

Tổ chức vật lý

Về mặt vật lý, dữ liệu thực sự được Elasticsearch tổ chức theo mô hình phân tán cho một Index với các thành phần như Cluster, Node, Shard.

Cụm Elaticsearch là một nhóm gồm một hoặc nhiều nút Elaticsearch được kết nối với nhau. Mỗi nút có mục đích và trách nhiệm riêng nhưng mỗi nút có thể chuyển tiếp các yêu cầu (điều phối) của máy khách đến các nút thích hợp. Sau đây là các nút được sử dụng trong cụm Elaticsearch:

- **Master node**: Nhiệm vụ của nút Master chủ yếu được sử dụng cho các hoạt động nhẹ trên toàn cụm, bao gồm tạo hoặc xóa chỉ mục, theo dõi các nút cụm và xác định vị trí của các phân đoạn được phân bổ.

- **Data node**: Nút dữ liệu chứa dữ liệu chứa các tài liệu được lập chỉ mục. Nó xử lý các hoạt động liên quan như CRUD, tìm kiếm và tổng hợp. Theo mặc định, vai trò nút dữ liệu được bật và người dùng có thể vô hiệu hóa vai trò đó cho một nút bằng cách đặt node.data thành false trong tệp elasticsearch.yml.

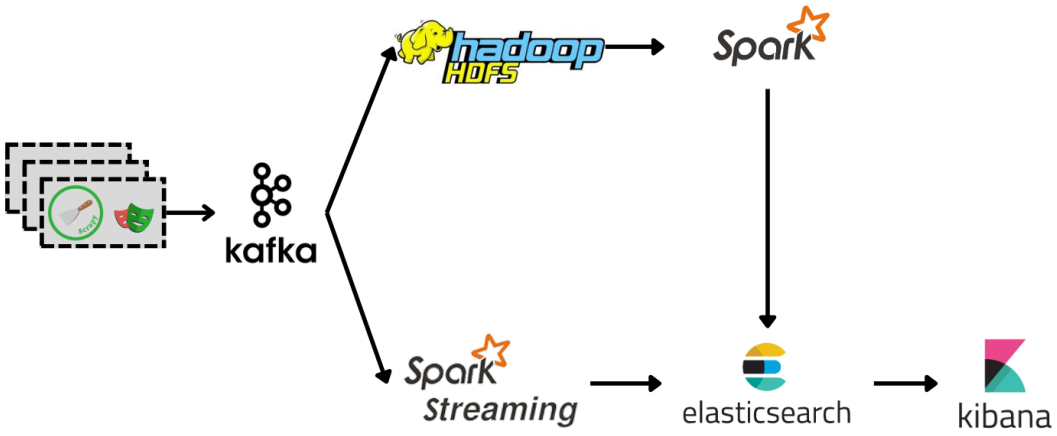
- **Ingest node**: Sử dụng Ingest node là một cách để xử lý tài liệu ở chế độ đường dẫn trước khi lập chỉ mục tài liệu. Theo mặc định, vai trò Ingest node được bật, ngoài ra có thể tắt vai trò đó cho một nút bằng cách đặt node.ingest thành false trong tệp elasticsearch.yml.

- **Coordinating-only node**: Nếu cả ba vai trò (master eligible, data, và ingest) đều bị tắt, nút sẽ chỉ hoạt động như một nút phối hợp thực hiện các yêu cầu định tuyến, xử lý giai đoạn giảm tìm kiếm và phân phối công việc thông qua lập chỉ mục hàng loạt.

2.6 Kibana

Kibana là một ứng dụng giao diện người dùng mở và miễn phí nằm trên Elastic Stack, cung cấp khả năng tìm kiếm và trực quan hóa dữ liệu cho dữ liệu được lập chỉ mục trong Elasticsearch. Kibana được thiết kế để tích hợp chặt chẽ với Elasticsearch, giúp trực quan hóa và hiểu dữ liệu lưu trữ trong Elasticsearch một cách dễ dàng.

Kibana cung cấp các tính năng cho người dùng quản lý như biểu đồ cột, biểu đồ đường, biểu đồ tròn, biểu đồ nhiệt và nhiều loại chart khác nữa. Kibana cho phép tổ chức và sắp xếp các biểu đồ và đồ thị vào các dashboard. Mỗi dashboard là một bảng điều khiển tổng hợp giúp theo dõi và hiểu dữ liệu một cách dễ dàng.



Hình 3.1 Hệ thống triển khai

3. Triển khai hệ thống

3.1 Tổng quan hệ thống

Hệ thống triển khai gồm tương đối đầy đủ các thành phần cần thiết để làm việc với dữ liệu lớn (Hình 3.1).

- **Thu thập dữ liệu** các trang web bất động sản với Scrapy và Playwright.
- Dữ liệu được gửi tới message queue (Kafka) và tiếp tục đi tới các thành phần **lưu trữ và xử lý** (HDFS và Spark).
- Sau khi được xử lý, dữ liệu sẽ được lưu trữ trên cụm Elasticsearch và Kibana phục vụ mục đích **tìm kiếm và trực quan hóa**.

Hầu hết các thành phần của hệ thống được triển khai trên nền tảng Docker Container giúp cài đặt nhanh chóng, dễ dàng và có tính khả mở cao.

3.2 Thu thập dữ liệu

Nhóm thực hiện thu thập dữ liệu các bài đăng rao bán bất động sản trên website cùng chủ đề. Các website này thường chứa nhiều trang web với nội dung là các bài đăng, thông tin của bài đăng, ở đây là thông tin của các bất động sản được rao bán, sẽ được nhúng trong các thẻ HTML của trang. Để thu thập được dữ liệu đã đề xuất, nhiệm vụ của nhóm là phải xác định được các website có dữ liệu bài đăng bất động sản, từ các website này đi tới các trang web chứa bài đăng và thực hiện bóc tách dữ liệu từ mã nguồn HTML

của những trang này. Quá trình này nhóm sử dụng Scrapy và Playwright để tự động hóa và tối ưu tốc độ.

Dữ liệu thu thập được sử dụng Scrapy và Playwright gồm khoảng 300,000 điểm dữ liệu thô, chủ yếu từ ba website *alonhadat.com.vn*, *bds.com.vn* và *123nhadatviet.com*. Dữ liệu này được lưu dưới định dạng JSON, tất cả các trường dữ liệu chưa qua xử lý đều ở dạng xâu, mỗi điểm dữ liệu gồm các trường sau đây:

- *'title'*: Tiêu đề bài đăng.
- *'description'*: Thông tin mô tả bất động sản được đính kèm.
- *'price'*: Giá bán bất động sản, đã đưa về đơn vị VNĐ nếu có thể.
- *'square'*: Diện tích bất động sản, đã đưa về đơn vị mét vuông nếu có thể.
- *'estate_type'*: Loại bất động sản.
- *'address'*:
  - *'full_address'*: Địa chỉ bất động sản theo bài đăng.
  - *'province'*: Tỉnh/ Thành phố (trích xuất từ 'full_address')
  - *'district'*: Quận/ Huyện/ Thị xã/ Thành phố thuộc tỉnh.
  - *'ward'*: Phường/ Xã
- *'post_date'*: Ngày đăng bài.
- *'post_id'*: ID bài đăng (riêng cho mỗi website)
- *'contact_info'*: Thông tin người đăng bài
  - *'name'*: Tên
  - *'phone'*: Số điện thoại
- *'extra_info'*: Các thông tin thêm không bắt buộc, nội dung trường này có thể khác nhau giữa các website do bài đăng trên mỗi trang web sẽ có những thông tin khác nhau không nằm trong phần bắt buộc (tình trạng pháp lý, số tầng, số phòng ngủ, . . .)
- *'link'*: Đường dẫn tới bài đăng.

3.3 Phân phối dữ liệu với Kafka

Trong thực tế một hệ thống làm việc với dữ liệu lớn sẽ có luồng dữ liệu rất phức tạp, dữ liệu đi theo nhiều hướng và khó để năm bắt. Kafka cung cấp giải pháp phân tách luồng dữ liệu này. Nhóm sử dụng Kafka để làm trung gian giữa thành phần thu thập dữ liệu và các thành phần lưu trữ, xử lý. Cụm Kafka được triển khai trên server Azure cloude với 3 broker và các topic tương ứng với tên website nhóm thực hiện thu thập dữ liệu là: **bds**, **i-batdongsan** và**123nhadatviet**. Tất cả các topic này đều được cài đặt với 3 phân vùng (partition) là hệ số nhân bản (replication) là 3.

3.4 Lưu trữ và xử lý dữ liệu

Dữ liệu thô dưới dạng xâu chưa thể sử dụng để đưa ra thông tin có ý nghĩa. Do đó cần có thêm các bước xử lý dữ liệu để tận dụng tài nguyên này. Đối với dữ liệu thô thu thập trong đề tài này sẽ được xử lý lần đầu bằng các item pipeline của Scrapy và sau đó tiếp tục được xử lý thêm sử dụng các biến đổi của Spark. Quá trình xử lý dữ liệu có thể thực hiện theo lô với Spark hoặc theo luồng với Spark Streaming, tuy nhiên logic và các bước xử lý là tương tự nhau. Bao gồm các công việc chính: loại bỏ dữ liệu trùng lặp, xử lý văn bản, xử lý các trường số, xử lý các trường thông tin thêm, chuẩn hóa địa chỉ.

- **Loại bỏ dữ liệu trùng lặp**: Thực hiện loại bỏ các bài đăng trùng lặp thông qua các trước thông tin: tiêu đề, mô tả và địa chỉ. Ở pipeline của Scrapy ta lọc các bài đăng trùng lặp trên một website bằng việc so sánh nội dung của các bài qua véc tơ TF-IDF. Trong bước tiếp theo khi xử lý với Spark, nhóm tận dụng thư viện ML của Spark, áp dụng thuật toán MinHash để tiếp tục lọc bỏ các bài đăng trùng lặp được đăng tải trên các website khác nhau. Bước này đã loại bỏ lượng lớn các bài đăng, cho thấy việc đăng tin rao bán một bất động sản trên nhiều website là phổ biến.
- **Xử lý văn bản** Một lượng lớn thông tin của các bài đăng được thể hiện dưới dạng văn bản, nhóm tiến hành làm sạch các trường dữ liệu kiểu này sử dụng các biến đổi python và áp dụng lên các DataFrame của Spark qua các UDF (User Defined Function), các biến đổi áp dụng như sau:
  - Chuẩn hóa unicode về dạng NFC.
  - Chuẩn hóa bảng mã Unicode tiếng việt về Unicode dựng sẵn.
  - Chuẩn hóa kiểu gõ dấu (oà $\rightarrow$ òa).
  - Chuẩn hóa chính tả (đột quị $\rightarrow$ đột quỵ).
  - Loại bỏ các kí tự đặc biệt không mang ý nghĩa (các kí tự tượng hình, emoji, ...).

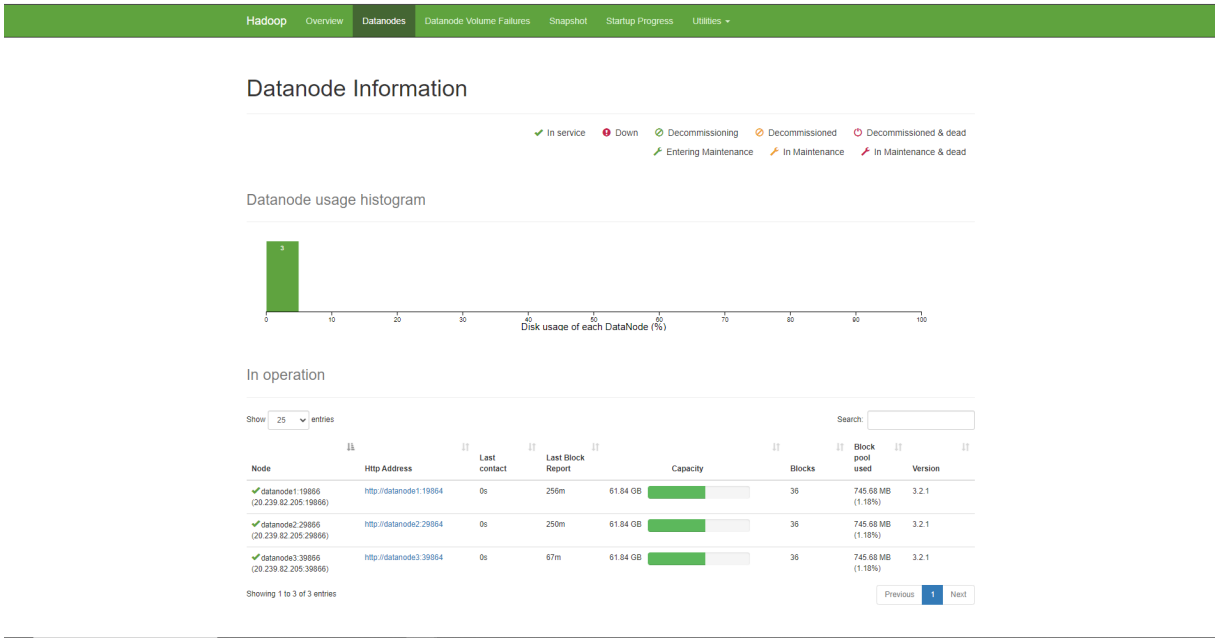
- Loại bỏ các dấu câu không mang ý nghĩa (các dấu câu được lặp lại thành một chuỗi dài để gây chú ý).

- **Xử lý các trường số** Các trường dữ liệu kiểu số quan trọng nhất là giá và diện tích ('price' và 'square'). Hai trường này thu thập được ở dạng xâu, nhóm chuyển chúng về kiểu số theo các luật được lập trình sao cho trường giá sẽ có đơn vị VNĐ và trường diện tích có đơn vị mét vuông (thay thế dựa vào hậu tố, ví dụ: 6 tỷ $\rightarrow$ 6,000,000,000). Một số bước xử lý tiếp theo của nhóm nhằm tiếp tục chuẩn hóa các giá trị xâu chưa thể chuyển về kiểu số theo cách trên và xử lý các giá trị quá lớn:
  - Với các bài đăng không có con số chính xác, giá thỏa thuận, coi như không có dữ liệu (None).
  - Các bài đăng có thông tin giá được tính theo diện tích sẽ kết hợp với thông tin trường diện tích để đưa ra con số chính xác.
  - Các giá trị giá bất động sản lớn vô lý hoặc quá nhỏ, phát hiện ngoại lai dựa trên các quantile range.

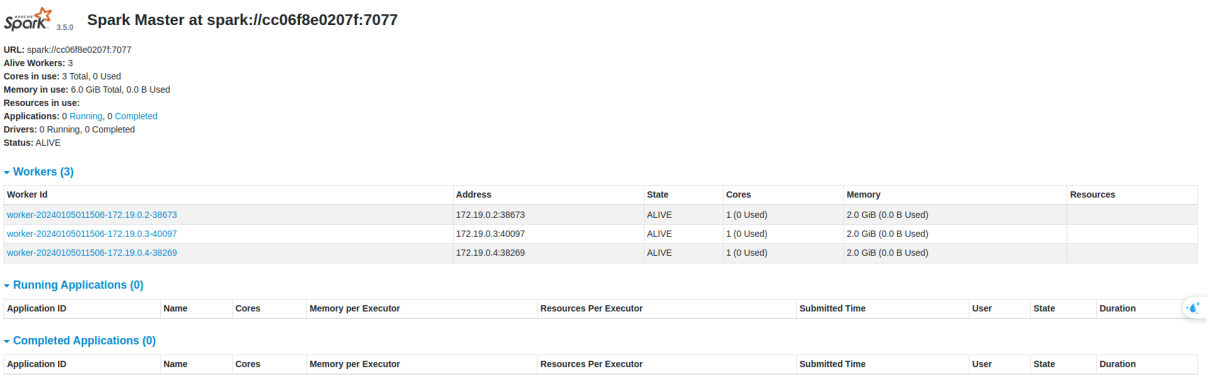
- **Xử lý các trường thông tin khác**:
  - Chuẩn hóa kiểu bất động sản, trường này có nhiều giá trị tương đồng, nhóm tiến hành gộp các nhãn này (ví dụ: Biệt thự, liền kề; Nhà biệt thự liền kề; Biệt thự liền kề).
  - Ngày đăng bài được chuyển về dạng 'yyyy/mm/dd' để thuận tiện so sánh.
  - Chuẩn hóa trường thông tin thêm ('extra_info'), các nhãn tương đồng được gộp lại, loại bỏ các nhãn không có giá trị, đồng thời chuyển các giá trị về kiểu dữ liệu hợp lý. Sau khi xử lý, trường thông tin thêm sẽ bao gồm các trường con sau: 'Chiều dài', 'Chiều ngang', 'Chính chủ', 'Chỗ để xe hơi', 'Hướng', 'Lộ giới', 'Nhà bếp', 'Pháp lý', 'Phòng ăn', 'Sân thượng', 'Số lầu', 'Số phòng ngủ', 'Số toilet', 'Tầng'.

- **Chuẩn hóa trường địa chỉ**: Giá trị của một bất động sản phụ thuộc rất lớn vào vị trí địa lý, vì vậy nhóm tiến hành xử lý trường địa chỉ, chuẩn hóa thông tin về vị trí tới cập phường, xã . Việc này được thực hiện bằng các luật, so sánh thông tin trường địa chỉ bất động sản với dữ liệu các đơn vị quản lý hành chính của Việt Nam, với các bài đăng không thực hiện được theo luật có thể sử dụng các mô hình đã được huấn luyện để chuẩn hóa địa chỉ tiếng Việt, tuy nhiên theo quan sát thực tế, chuẩn hóa theo luật là chính xác và nhanh hơn.

Để thực hiện các bước này, nhóm triển khai cụm Spark với một master và ba worker trên Docker container trên máy tính cá nhân. Đối với Spark Streaming để xử lý dữ liệu



Hình 3.2 Cụm HDFS

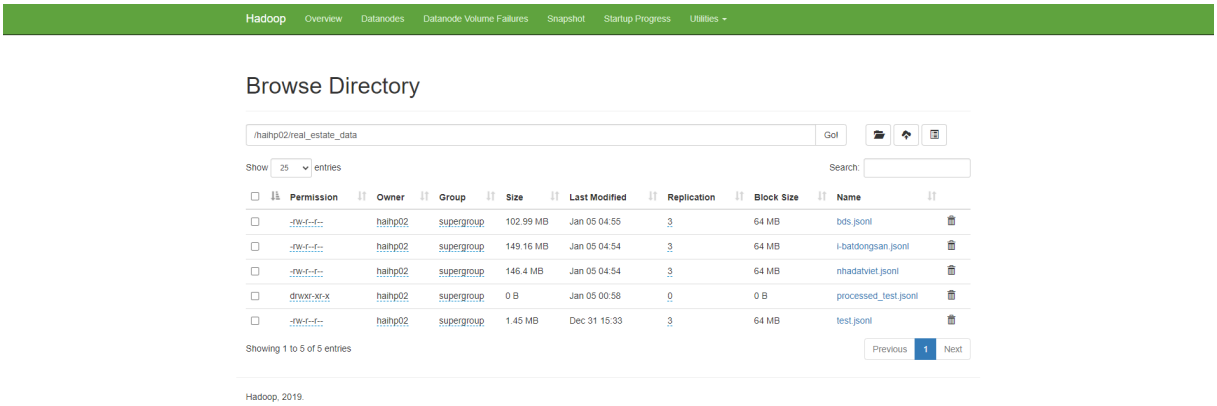


Hình 3.3 Cụm Spark

theo luồng sẽ chạy trực tiếp trên máy tính cá nhân do vấn đề tương thích phiên bản. Đồng thời, nhóm triển khai cụm HDFS với namenode và ba datanode bằng Docker container trên server của Azure Cloud để thực hiện lưu trữ dữ liệu thô và cả dữ liệu đã được xử lý bởi Spark. Cụm Spark và HDFS triển khai được thể hiện ở hai Hình 3.2 và 3.3

3.5 Tìm kiếm và trực quan hóa

Phần tìm kiếm và trực quan hóa trong hệ thống của nhóm đóng một vai trò quan trọng trong việc cung cấp trải nghiệm người dùng tốt và giúp người quản trị hiểu rõ hơn về dữ liệu và thông tin được thu thập từ các nguồn khác nhau. Nhóm triển khai cụm elasticsearch gồm 3 node cùng với một node kibana trên server Azure cloud (các node



Hình 3.4 Dữ liệu lưu trữ trên HDFS

đều là Docker container).

3.5.1 Tìm kiếm bằng elasticsearch

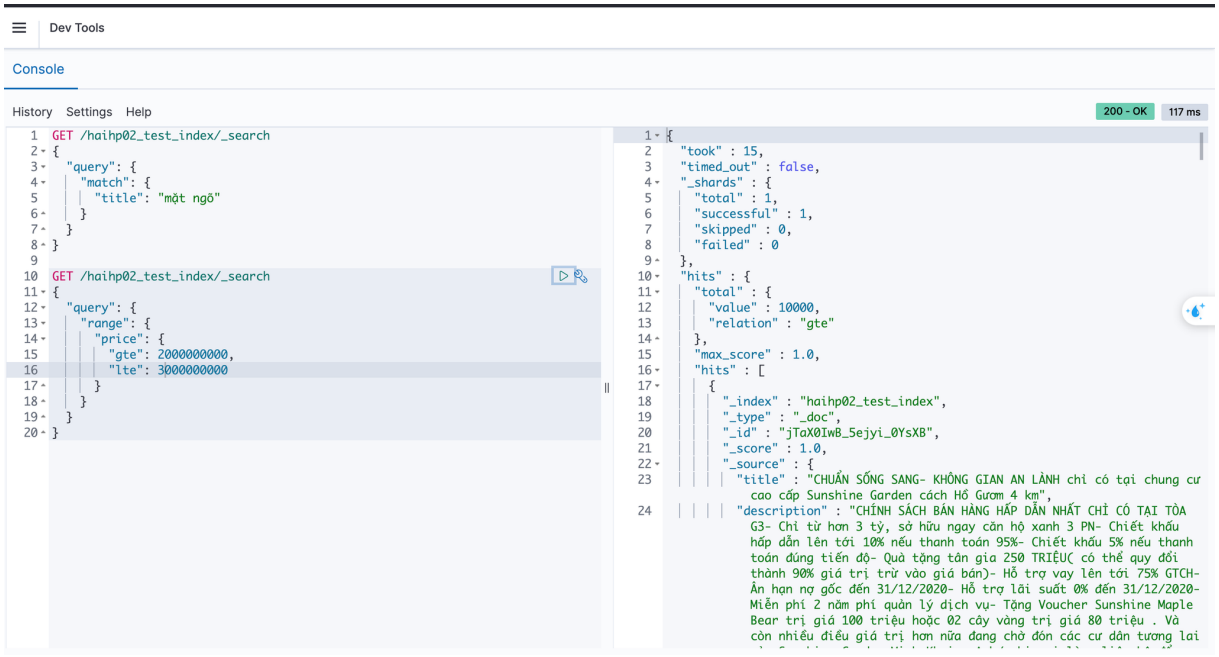
Elasticsearch sử dụng các thuật toán tìm kiếm hiệu quả và cơ sở dữ liệu không gian lưu trữ đảm bảo tìm kiếm diễn ra một cách nhanh chóng và mượt mà. Indexing và tokenization được thực hiện để tối ưu hóa quá trình tìm kiếm ở hình 3.5.

Elasticsearch tích hợp chặt chẽ với các thành phần khác của hệ thống, đặc biệt là với phần tìm kiếm và trực quan hóa thông qua Kibana. Kết hợp với các công cụ theo dõi để đảm bảo hiệu suất và tin cậy.

3.5.2 Trực quan hóa dữ liệu bằng Kibana

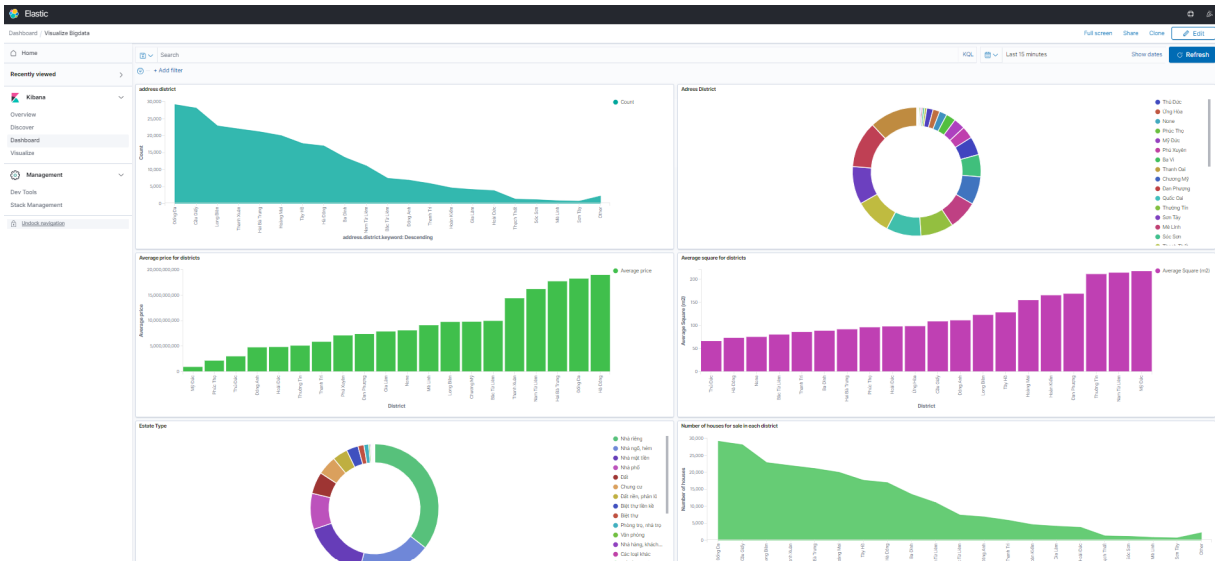
Trực quan hóa bằng Kibana trong hệ thống của nhóm chủ yếu tập trung vào việc cung cấp công cụ mạnh mẽ để biểu diễn dữ liệu một cách trực quan và hiểu rõ hơn thông tin được thu thập từ các nguồn khác nhau.

Sử dụng Kibana để tạo các bảng điều khiển (dashboards) linh hoạt và dễ sử dụng. Kết hợp nhiều loại biểu đồ hình 3.6 và bảng để thể hiện thông tin một cách toàn diện. Đồng thời có thể sử dụng các tính năng lọc và phân loại để tương tác và xem xét dữ liệu từ nhiều góc độ khác nhau. Tích hợp các yếu tố tìm kiếm để chọn lọc dữ liệu theo yêu cầu cụ thể. Từ các biểu đồ trên có thể nhanh chóng nhận thấy Đống Đa và Cầu Giấy là hai quận có nhiều bất động sản được rao bán nhất, theo ngay sau cũng là các quận thuộc khu vực nội thành, điều này cho thấy khu vực nội thành Hà Nội có cường độ giao dịch bất động sản và mức độ quan tâm lớn hơn hẳn so với ngoại thành. Quan sát thêm ta cũng



Hình 3.5 Tìm kiếm bằng Devtools trên Elasticsearch

nhận thấy diện tích trung bình của các bất động sản ở khu vực ngoại thành sẽ lớn hơn, điều này phù hợp với thực tế khi mật độ dân cư trong nội thành lớn hơn hẳn, đồng thời các loại bất động sản được rao bán ở vùng ngoại ô cũng thường là đất, trái ngược với nhà ở và chung cư ở nội thành. Việc phân tích và trực quan hóa còn có thể được thực hiện thêm tuy nhiên do giới hạn thời gian nên nhóm dừng lại ở một số biểu đồ đã trình bày.



Hình 3.6 Visualize dữ liệu bằng Kibana

4. Trải nghiệm khi cài đặt hệ thống

1. Crawl dữ liệu từ các trang bất động sản: Đã thực hiện thu thập hết tất cả các bài đăng thuộc thành phố Hà Nội trong 3 website phía trên
2. Viết các hàm xử lý làm sạch dữ liệu: Dữ liệu thô dưới dạng xâu chưa xử dụng được nên thêm các bước xử lý lần đầu bằng các item pipeline của Scrapy và xử lý thêm ở Spark
3. Sử dụng Kafka để phân phối dữ liệu: Nhóm sử dụng Kafka để làm trung gian giữa thành phần thu thập dữ liệu và các thành phần lưu trữ, xử lý. Cụm Kafka được triển khai trên server Azure cloude với 3 broker và các topic tương ứng với tên website nhóm thực hiện thu thập dữ liệu là: bds, i-batdongsan và123nhadatviet.
4. Sử dụng HDFS để lưu dữ liệu về: Nhóm triển khai cụm HDFS với namenode và ba datanode trên server của Azure Cloud để thực hiện lưu trữ dữ liệu thô và cả dữ liệu đã được xử lý bởi Spark
5. Sử dụng Spark: Nhóm sử dụng Kafka xử lý dữ liệu theo các công việc chính: loại bỏ dữ liệu trùng lặp, xử lý văn bản, xử lý các trường số, xử lý các trường thông tin thêm, chuẩn hóa địa chỉ
6. Sử dụng Spark Streaming : Spark Streaming để xử lý dữ liệu theo luồng sẽ chạy trực tiếp trên máy tính cá nhân do vấn đề tương thích phiên bản
7. Đẩy dữ liệu lên Elasticsearch: Nhóm đẩy dữ liệu lên Elasticsearch bằng Spark sau khi xử lý từ kho lưu trữ hoặc từ luồng dữ liệu trực
8. Sử dụng Elasticsearch để search tìm kiếm dữ liệu: Nhóm sử dụng các câu lệnh query để tìm kiếm dữ liệu trên Devtools
9. Sử dụng Kibana để visualize dữ liệu: Sử dụng Kibana để biểu diễn và trực quan hóa dữ liệu đã được đẩy lên bằng công cụ trên Kibana
10. Phát triển triển khai mô hình trên server: Nhóm chạy các thành phần hệ thống trên server Azure Microsoft
11. Kết nối các hệ thống qua server: Triển khai các docker container kafka, hadoop, elasticsearch trên server với các cổng khác nhau và kết nối theo luồng pipline
12. Kết nối hệ thống đến các ip của server: Vấn đề gặp phải khi triển khai server là các thành phần trỏ sai vào địa chỉ ip và đã được chỉnh sửa để chạy hoàn thành

5. Nhận xét và hướng phát triển

5.1 Nhận xét

Trong bài tập lớn này, nhóm đã thực hiện một hành trình quan trọng trong việc lưu trữ và xử lý dữ liệu bất động sản ở Hà Nội, sử dụng một quy trình tích hợp từ việc crawl dữ liệu, lưu trữ vào Hadoop Distributed File System (HDFS), xử lý thông qua Apache Spark, và cuối cùng, trực quan hóa thông qua Elasticsearch và Kibana. Nhóm đã bắt đầu với quá trình thu thập dữ liệu, thu thập thông tin quan trọng từ các nguồn đáng tin cậy. Điều này đặt nền tảng cho việc xây dựng một kho dữ liệu đa dạng và phong phú về thị trường bất động sản ở Hà Nội. Các bước thu thập đã hoàn thiện tuy nhiên do nhóm nhận thấy dữ liệu không có tính thời sự nên chưa cài đặt lập lịch thu thập hàng ngày.

Sau đó, nhóm đã sử dụng HDFS để lưu trữ dữ liệu một cách hiệu quả và phân tán. Điều này không chỉ giúp giảm áp lực lưu trữ mà còn tăng cường khả năng mở rộng của hệ thống để đáp ứng với sự gia tăng liên tục của dữ liệu.

Tiếp theo, việc sử dụng Apache Spark đã mang lại khả năng xử lý dữ liệu mạnh mẽ và hiệu quả. Spark không chỉ giúp nhóm thực hiện các phép toán phức tạp mà còn tăng tốc quá trình xử lý dữ liệu, giảm thời gian tính toán và tăng hiệu suất của hệ thống.

Cuối cùng, thông qua Elasticsearch và Kibana, nhóm đã có khả năng trực quan hóa dữ liệu một cách dễ dàng và hiệu quả. Điều này giúp người dùng dễ dàng theo dõi và phân tích thông tin, từ đó đưa ra những quyết định thông minh và chiến lược.

Tổng cộng, quy trình này không chỉ là một ví dụ minh họa cho cách tích hợp các công nghệ khác nhau mà còn là một cơ hội để hiểu sâu hơn về quy trình lưu trữ và xử lý dữ liệu trong môi trường bất động sản đầy thách thức. Sự kết hợp giữa crawl, lưu trữ, xử lý và trực quan hóa dữ liệu mang lại một cơ sở hạ tầng mạnh mẽ cho việc nghiên cứu và phát triển trong lĩnh vực này.

5.2 Hướng phát triển

Hướng phát triển tiếp theo cho bài tập lớn về lưu trữ và xử lý dữ liệu bất động sản ở Hà Nội có thể tập trung vào một số khía cạnh quan trọng để nâng cao hiệu suất và tính linh hoạt của hệ thống.

- Mở rộng nguồn dữ liệu: Tăng cường khả năng thu thập dữ liệu từ nhiều nguồn hơn để có một bức tranh toàn diện và chính xác về thị trường bất động sản. Kết hợp dữ liệu từ các nguồn khác nhau như các cơ quan chính phủ, trang web bất động sản, và các hệ thống thông tin khác.

- Tối ưu hóa xử lý dữ liệu: Nghiên cứu và triển khai các kỹ thuật tối ưu hóa xử lý dữ liệu trong môi trường Apache Spark để giảm thời gian xử lý và tăng cường hiệu suất. Sử dụng các cơ sở dữ liệu phân tán khác như Apache Cassandra để tối ưu hóa việc truy vấn dữ liệu lớn.

- Bảo mật và quản lý phiên: Tăng cường các biện pháp bảo mật để đảm bảo an toàn cho dữ liệu bất động sản, đặc biệt là khi liên quan đến thông tin cá nhân và giao dịch tài chính. Phát triển mô hình quản lý phiên để kiểm soát quyền truy cập và theo dõi hoạt động người dùng.

- Tích hợp trí tuệ nhân tạo (AI) và Học máy: Sử dụng trí tuệ nhân tạo và học máy để dự đoán xu hướng thị trường, giá cả và nhu cầu của khách hàng. Tích hợp các mô hình dự đoán để cung cấp thông tin chi tiết và dự báo chính xác.

- Phát triển ứng dụng di động: Xây dựng ứng dụng di động để người dùng có thể truy cập thông tin bất động sản mọi lúc, mọi nơi. Tích hợp các tính năng tương tác như tìm kiếm nâng cao, thông báo về thị trường, và đặt lịch hẹn xem nhà.

- Hợp tác và kết nối: Xây dựng cộng đồng hoặc hợp tác với các tổ chức bất động sản, chính quyền địa phương để chia sẻ thông tin và tạo ra một hệ sinh thái mạnh mẽ. Tích hợp các tiêu chuẩn mở để tạo ra khả năng kết nối và chia sẻ dữ liệu dễ dàng hơn với các hệ thống khác.

KẾT LUẬN

Qua báo cáo này nhóm đã trình bày toán bộ công việc trong quá trình thực hiện đề tài triển khai hệ thống thu thập, lưu trữ, xử lý và phân tích bất động sản với dữ liệu là các bài đăng rao bán bất động sản tại Hà Nội. Đây là một bài toán hay có ý nghĩa thực tế to lớn.

Trong quá trình thực hiện đề tài này, nhóm đã được làm quen với các công nghệ, bước đầu học được cách làm việc với dữ liệu lớn và triển khai một hệ thống tự động hóa quá trình này. Do còn chưa có nhiều kinh nghiệm, nhóm nhận thấy việc thực hiện còn gặp những khó khăn trong việc tiếp cận nguồn dữ liệu tốt và có nhiều vấn đề phát sinh trong khâu triển khai nhưng kết quả đã có một hệ thống tương đối hoàn chỉnh. Trong thời gian tới, nhóm mong muốn có thể tiếp tục công việc và hoàn thiện hơn nữa quy trình, hệ thống đề xuất để tiến tới khả năng sử dụng có ích trong thực tế.

Trong quá trình làm việc, nhóm đã cố gắng thu thập thông tin và học hỏi, thử nghiệm các giải pháp cho vấn đề, tuy nhiên do chưa có nhiều kiến thức chuyên sâu về miền bài toán đặt ra cũng như tiếp cận với các công nghệ dữ liệu lớn mà nhóm còn chưa có nhiều kinh nghiệm nên kết quả không thể tránh khỏi những thiếu sót, hạn chế. Nhóm rất mong nhận được đóng góp để có thể hoàn thiện tốt hơn nữa.

Một lần nữa nhóm xin gửi lời cảm ơn chân thành tới TS. Trần Việt Trung, các giờ giảng dạy của thầy là nền tảng không thể thiếu để nhóm có thể thực hiện được đề tài này.