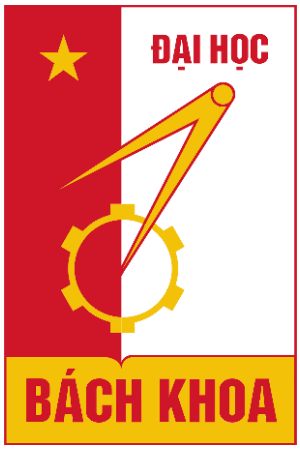
ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO PROJECT III

Đề tài: Xây dựng hệ thống hồ dữ liệu phân tích dữ liệu chuyến bay

Họ tên sinh viên	: Trần Ngọc Bảo
Mã số sinh viên	: 20215529
Giảng viên hướng dẫn	: TS. Ngô Thành Trung

Hà Nội, tháng 12 năm 2024

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Dữ liệu lớn (Big Data) phát triển nhanh chóng và đóng vai trò quan trọng trong chuyển đổi số và công nghệ, với dự báo khối lượng dữ liệu toàn cầu đạt 180 zettabyte vào năm 2025 nhờ IoT, mạng xã hội và thiết bị thông minh. Dữ liệu lớn được ứng dụng trong nhiều ngành để tối ưu hóa hiệu suất, cải thiện trải nghiệm người dùng và hỗ trợ ra quyết định. Tuy nhiên, nó đối mặt với thách thức về xử lý, lưu trữ, bảo mật, chất lượng dữ liệu và chi phí. Giải quyết các vấn đề này sẽ mang lại lợi ích lớn như tối ưu hoạt động, ra quyết định chính xác, cá nhân hóa dịch vụ và phát triển bền vững trong nhiều lĩnh vực, từ y tế đến môi trường.

1.2 Mục tiêu và phạm vi đề tài

Khi khối lượng dữ liệu ngày càng tăng, các hệ thống xử lý dữ liệu lớn như kho dữ liệu (Data Warehouse), hồ dữ liệu (Data Lake), và hồ kho dữ liệu (Data Lakehouse) đóng vai trò quan trọng trong quản lý và phân tích. Kho dữ liệu có cấu trúc cao nhưng chi phí và khó mở rộng; hồ dữ liệu linh hoạt và chi phí thấp nhưng không hỗ trợ tốt cho truy vấn; hồ kho dữ liệu kết hợp ưu điểm của cả hai nhưng gặp khó khăn trong việc duy trì chất lượng dữ liệu. Hồ dữ liệu nổi bật nhờ tính linh hoạt, khả năng lưu trữ nhiều loại dữ liệu và chi phí thấp. Các hệ sinh thái công nghệ như Hadoop, Google BigQuery và Amazon Redshift cung cấp các công cụ mạnh mẽ, với Hadoop nổi bật nhờ chi phí thấp và khả năng xử lý dữ liệu lớn. Hệ thống hồ dữ liệu phân tích chuyến bay sử dụng Hadoop giúp xử lý dữ liệu khổng lồ từ nhiều nguồn, tối ưu hóa hoạt động và nâng cao trải nghiệm hành khách.

1.3 Định hướng giải pháp

Hệ thống hồ dữ liệu phân tích chuyến bay sử dụng các thành phần của hệ sinh thái Hadoop để xử lý và phân tích dữ liệu lớn. Dữ liệu chuyến bay được lấy từ bộ dữ liệu trên Kaggle và được mô phỏng qua máy chủ API để cập nhật liên tục. Apache Kafka thu thập và truyền tải dữ liệu theo thời gian thực, trong khi Apache Airflow tự động hóa các quy trình ETL để làm mới và chuẩn hóa dữ liệu. Dữ liệu được lưu trữ trong HDFS, giúp đảm bảo tính ổn định và khả năng phục hồi. Apache Spark xử lý và phân tích dữ liệu nhanh chóng, hỗ trợ Spark SQL và Spark Streaming cho các truy vấn và xử lý theo thời gian thực.

Hive tổ chức dữ liệu để truy vấn dễ dàng, và Trino (Presto) cung cấp khả năng truy vấn phân tán từ nhiều nguồn dữ liệu. Apache Superset tạo bảng điều khiển và trực quan hóa dữ liệu giúp người dùng ra quyết định nhanh chóng. Tất cả các thành phần này hoạt động trên Kubernetes, cung cấp khả năng mở rộng linh hoạt và tối ưu hóa hiệu suất hệ thống.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

2.1 Khảo sát hiện trạng

Các doanh nghiệp hiện nay phải đối mặt với khối lượng dữ liệu ngày càng lớn và phức tạp, đòi hỏi hệ thống dữ liệu mạnh mẽ để thu thập, lưu trữ và xử lý hiệu quả. Các hệ thống phổ biến như kho dữ liệu (Data Warehouse), hồ dữ liệu (Data Lake), và hồ kho dữ liệu (Data Lakehouse) được lựa chọn tùy theo nhu cầu cụ thể. Kho dữ liệu chuyên về dữ liệu có cấu trúc và phân tích định kỳ, trong khi hồ dữ liệu hỗ trợ dữ liệu phi cấu trúc và mở rộng linh hoạt. Hồ kho dữ liệu kết hợp ưu điểm của cả hai, cho phép lưu trữ và phân tích cả dữ liệu có cấu trúc và phi cấu trúc.

Doanh nghiệp phải quyết định giữa triển khai hệ thống tại chỗ (on-premise) hoặc đám mây. Mô hình tại chỗ mang lại kiểm soát và bảo mật cao nhưng chi phí triển khai và duy trì cao. Hệ thống đám mây giúp tiết kiệm chi phí và dễ dàng mở rộng, nhưng có thể gặp vấn đề về bảo mật và quyền kiểm soát dữ liệu.

Mặc dù các công nghệ mới như dịch vụ đám mây đang phát triển mạnh mẽ, hệ sinh thái Hadoop vẫn giữ vai trò quan trọng trong việc xử lý dữ liệu phân tán quy mô lớn, đặc biệt là dữ liệu phi cấu trúc. Hadoop kết hợp với các công nghệ như Apache Spark, Hive, HBase và Trino giúp tăng cường khả năng phân tích và truy vấn dữ liệu. Tuy nhiên, để khắc phục những hạn chế trong triển khai, Kubernetes (K8s) đang trở thành xu hướng mới, hỗ trợ tự động hóa và quản lý tài nguyên hiệu quả khi kết hợp với Hadoop, giúp tối ưu hóa việc lưu trữ và xử lý dữ liệu lớn, đồng thời giảm chi phí và nâng cao tính linh hoạt.

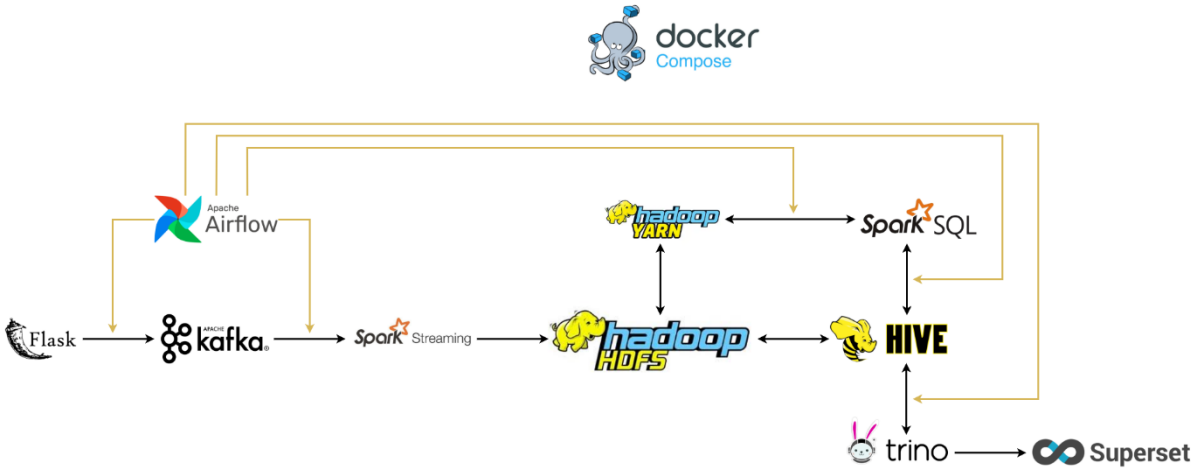
2.2 Tổng quan chức năng

Hệ thống hồ dữ liệu hiện đại bao gồm các mô đun cơ bản để hỗ trợ lưu trữ, xử lý và phân tích dữ liệu quy mô lớn. Mô đun lập lịch tác vụ tự động hóa quá trình thu thập, xử lý và chuyển giao dữ liệu, đảm bảo các tác vụ diễn ra đúng thời gian. Mô đun thu thập dữ liệu lấy dữ liệu từ nhiều nguồn khác nhau, bao gồm dữ liệu có cấu trúc và phi cấu trúc từ các hệ thống bên ngoài như cơ sở dữ liệu, ứng dụng, cảm biến và nguồn dữ liệu thời gian thực. Dữ liệu sau đó được đưa vào mô đun lưu trữ, nơi nó được tổ chức và lưu trữ trong môi trường phân tán trên đám mây hoặc tại chỗ. Mô đun xử lý dữ liệu chuẩn hóa và làm sạch dữ liệu, chuyển đổi nó thành dạng có thể phân tích được.

Sau khi xử lý, mô đun truy vấn dữ liệu cho phép người dùng thực hiện các truy vấn SQL và các công cụ phân tích để khai thác thông tin. Cuối cùng, mô đun trực quan hóa dữ liệu giúp hiển thị kết quả phân tích dưới dạng đồ thị và báo cáo, hỗ trợ người dùng đưa ra quyết định nhanh chóng và chính xác. Sự kết hợp của các mô đun này tạo ra một hệ thống hồ dữ liệu mạnh mẽ, linh hoạt, có khả năng xử lý và phân tích dữ liệu từ nhiều nguồn khác nhau, đồng thời hỗ trợ trực quan hóa giúp người dùng dễ dàng nắm bắt thông tin.

CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

3.1 Kiến trúc tổng quan



Hệ thống do em đề xuất sử dụng các công nghệ nguồn mở để xử lý và phân tích dữ liệu lớn, bắt đầu từ việc thu thập dữ liệu qua ứng dụng Flask, sau đó gửi dữ liệu đến Apache Kafka. Kafka hoạt động như nền tảng truyền tải tin nhắn, xử lý dữ liệu luồng theo thời gian thực và chuyển tiếp đến Apache Spark Streaming để xử lý. Dữ liệu được lưu trữ trong HDFS, một hệ thống lưu trữ phân tán, với tài nguyên tính toán do Hadoop YARN quản lý để tối ưu hóa hiệu suất. Spark SQL được sử dụng để xử lý dữ liệu theo lô và thời gian thực.

Để tự động hóa quy trình, Apache Airflow quản lý các luồng công việc, đảm bảo tính liên tục và chính xác. Trino kết hợp với Apache Hive để thực hiện truy vấn phân tán trên HDFS, cho phép thực hiện các truy vấn phức tạp và kết hợp dữ liệu từ nhiều nguồn. Dữ liệu đã xử lý sau đó được chuyển đến Apache Superset, công cụ trực quan hóa dữ liệu, để tạo báo cáo và biểu đồ hỗ trợ ra quyết định. Toàn bộ hệ thống được triển khai trên nền Kubernetes, đảm bảo khả năng mở rộng linh hoạt, quản lý tài nguyên hiệu quả và duy trì tính sẵn sàng cao.

3.2 Mô-đun thu thập dữ liệu

Đầu tiên, em sẽ xây dựng mô-đun thu thập dữ liệu bằng cách giả lập quá trình này thông qua việc tạo một máy chủ sử dụng ngôn ngữ lập trình Python và framework Flask. Máy chủ này sẽ đóng vai trò là nơi lưu trữ dữ liệu thô ban đầu, mô phỏng nguồn dữ liệu thực tế. Bằng cách này, máy chủ có khả năng cung cấp dữ liệu theo thời gian thực khi có yêu cầu thông qua API. Đây sẽ là nền tảng để thử nghiệm và đánh giá các chức năng xử lý dữ liệu trong các bước tiếp theo.

3.3 Mô-đun lập lịch tác vụ

Tiếp theo, em xây dựng mô-đun lập lịch tác vụ bằng cách sử dụng Airflow để thiết kế luồng xử lý dữ liệu cho hệ thống. Trong Airflow, em tạo một biểu đồ phụ thuộc công việc (DAG - Directed Acyclic Graph), trong đó mỗi tác vụ (task) được đại diện bởi một operator. Mỗi operator đảm nhiệm một giai đoạn cụ thể

trong quy trình xử lý dữ liệu: thu thập, xử lý, lưu trữ và truy vấn. DAG giúp kết nối các tác vụ theo trình tự logic, đảm bảo luồng dữ liệu diễn ra liên tục, không bị gián đoạn. Ngoài ra, Airflow cung cấp các công cụ giám sát và kiểm tra lỗi, giúp theo dõi trạng thái thực thi của từng tác vụ, phát hiện và xử lý lỗi kịp thời, từ đó đảm bảo tính ổn định và hiệu quả của toàn bộ hệ thống.

3.4 Mô-đun lưu trữ dữ liệu

Trong hệ thống có 3 nơi lưu trữ dữ liệu đó là: (i) dữ liệu nguồn tại máy chủ Flask cung cấp API, lưu trữ 13.4 Gb dữ liệu dưới định dạng CSV; (ii) dữ liệu thời gian thực được lưu trữ tại cụm Kafka được chia theo các topic, partition, offset; (iii) dữ liệu lưu trên HDFS. Trong đó HDFS là nơi lưu trữ dữ liệu chính của hệ thống. HDFS lưu trữ dữ liệu sau quá trình tiền xử lý Spark Streaming, dữ liệu sau quá trình xử lý theo lô của Spark SQL và các bảng dữ liệu được tạo ra sau quá trình truy vấn dữ liệu bằng Trino

Trên HDFS, thư mục "staging/" được sử dụng để lưu trữ dữ liệu thô, được tổ chức theo cấu trúc phân cấp với các thư mục con theo năm và tháng. Mỗi thư mục tháng chứa các tệp dữ liệu dạng Parquet, đây là kết quả của quá trình thu thập dữ liệu. Thư mục "processed_data/" chứa dữ liệu đã qua xử lý, được tổ chức theo năm và tháng với các thư mục con lưu trữ dữ liệu sau quá trình tiền xử lý và xử lý dữ liệu. Thư mục này còn bao gồm cơ sở dữ liệu "process_data.db", nơi các tệp dữ liệu đã xử lý được lưu trữ và tổ chức theo dạng cơ sở dữ liệu bằng Spark SQL kết hợp với Hive. Thư mục "datasets/" chứa các bộ dữ liệu đầu ra được phân loại theo các nhóm thời gian như tháng, quý, năm và các loại báo cáo. Các thư mục con trong "datasets/" được đặt tên theo thời gian và loại báo cáo, ví dụ như "datasets_month/", "datasets_quarter/", "datasets_year/", và "datasets/", nơi lưu trữ các tệp dữ liệu đã được truy vấn và xử lý sẵn sàng để sử dụng cho các bước phân tích và trực quan hóa dữ liệu. Mỗi tệp dữ liệu trong các thư mục này đều có thể được sử dụng để tạo các báo cáo, bảng điều khiển, hoặc để phân tích thêm trong hệ thống.

3.5 Mô-đun xử lý dữ liệu

Dữ liệu sẽ được xử lý, biến đổi sử dụng Spark Streamming và Spark SQL. Quá trình này được thực hiện thông qua việc Airflow nộp (submit) mã nguồn Spark lên cụm YARN trong chế độ client mode. Cách tiếp cận này cho phép tận dụng tài nguyên và cơ chế quản lý của cụm YARN để đảm bảo hiệu quả và độ ổn định trong quá trình xử lý dữ liệu. Quá trình xử lý dữ liệu gồm hai phần chính là xử lý dữ liệu theo thời gian thực (BashOperator "load_data") và xử lý dữ liệu theo lô (BashOperator "transform_data") đã nêu qua trong mô-đun lập lịch tác vụ.

Đối với Spark Streaming, dữ liệu thời gian thực từ Kafka được xử lý và lưu trữ vào HDFS. Quá trình bắt đầu bằng việc khởi tạo một SparkSession với các cấu hình đặc biệt hỗ trợ tích hợp Kafka và xử lý dữ liệu dạng luồng thông qua "readStream". Dữ liệu được đọc từ Kafka topic "flight_data_{year}" và chuyển đổi thành định dạng chuỗi JSON. Sau đó, JSON được ánh xạ vào một schema đã

xác định trước "json_schema" để phân tách thành các cột dữ liệu riêng biệt, giúp dễ dàng truy cập và thao tác. Dữ liệu sau khi được xử lý sẽ được ghi vào HDFS dưới dạng tệp Parquet, lưu tại đường dẫn "/staging/{year}/{month}". Việc ghi sử dụng chế độ "append" để thêm dữ liệu mới liên tục, đồng thời kết hợp "checkpointing" tại một đường dẫn tạm thời để đảm bảo quá trình ghi có khả năng phục hồi trong trường hợp xảy ra sự cố, tăng độ tin cậy cho hệ thống. Cuối cùng, Spark Streaming chờ hoàn tất quá trình xử lý thông qua "awaitTermination()", đảm bảo toàn bộ luồng dữ liệu được xử lý liền mạch và không bị gián đoạn.

Đối với Spark SQL thực hiện xử lý và chuyển đổi dữ liệu từ HDFS, sau đó lưu trữ vào một bảng trong Hive. Đầu tiên, "SparkSession" được khởi tạo với hỗ trợ Hive, và thư mục kho dữ liệu được cấu hình tại "hdfs://namenode:9000/processed_data". Giá trị "year" và "month" được truyền từ dòng lệnh. dữ liệu Parquet từ HDFS trong đường dẫn "/staging/{year}/{month}" được đọc vào. Dữ liệu được chọn lọc các cột cần thiết thông qua "select", loại bỏ các bản ghi trùng lặp bằng "dropDuplicates", và xử lý các cột thời gian "DepTimeBlk" và "ArrTimeBlk" bằng cách tách thành các giá trị bắt đầu, kết thúc và tính khoảng cách giữa chúng. Các cột này được thêm mới vào DataFrame dưới dạng các cột tính toán. Sau đó, mã kiểm tra và tạo một schema Hive "processed_data" nếu chưa tồn tại, chuyển sang sử dụng schema này và tạo bảng Hive tương ứng cho từng năm "flight_{year}". Cuối cùng, dữ liệu được chuyển đổi kiểu dữ liệu cho các cột cần thiết và chèn vào bảng Hive từ một "temp_view". Điều này đảm bảo dữ liệu được lưu trữ có cấu trúc và sẵn sàng cho các truy vấn tiếp theo.

3.6 Mô-đun truy vấn dữ liệu

Dữ liệu sau khi được xử lý bởi Spark sẽ được truy vấn và phân tích bằng Trino. Quá trình này tập trung vào việc khai thác thông tin từ dữ liệu chuyến bay để tạo ra các báo cáo phù hợp, được phân tích theo các mốc thời gian như tháng, quý, năm, hoặc so sánh giữa các năm. Công việc này được thực hiện bởi PythonOperator "query_data" như đã trình bày trong mục \ref{subsection:4.1.4}. Các báo cáo đầu ra bao gồm: (i) phân tích mạng lưới tiếp thị hàng không; (ii) phân bố chuyến bay theo các ngày trong tuần; (iii) phân bố điểm xuất phát của các chuyến bay; (iv) hủy chuyến bay theo các ngày trong tuần; (v) tổng số chuyến bay bị hủy và phân bố theo từng hãng hàng không; (vi) phân tích đặc điểm của các ngày trong tháng; (vii) phân tích độ trễ trung bình theo từng hãng hàng không; (viii) số lượng chuyến bay theo từng tháng qua các năm; (ix) phân bố điểm xuất phát của các chuyến bay qua từng năm.

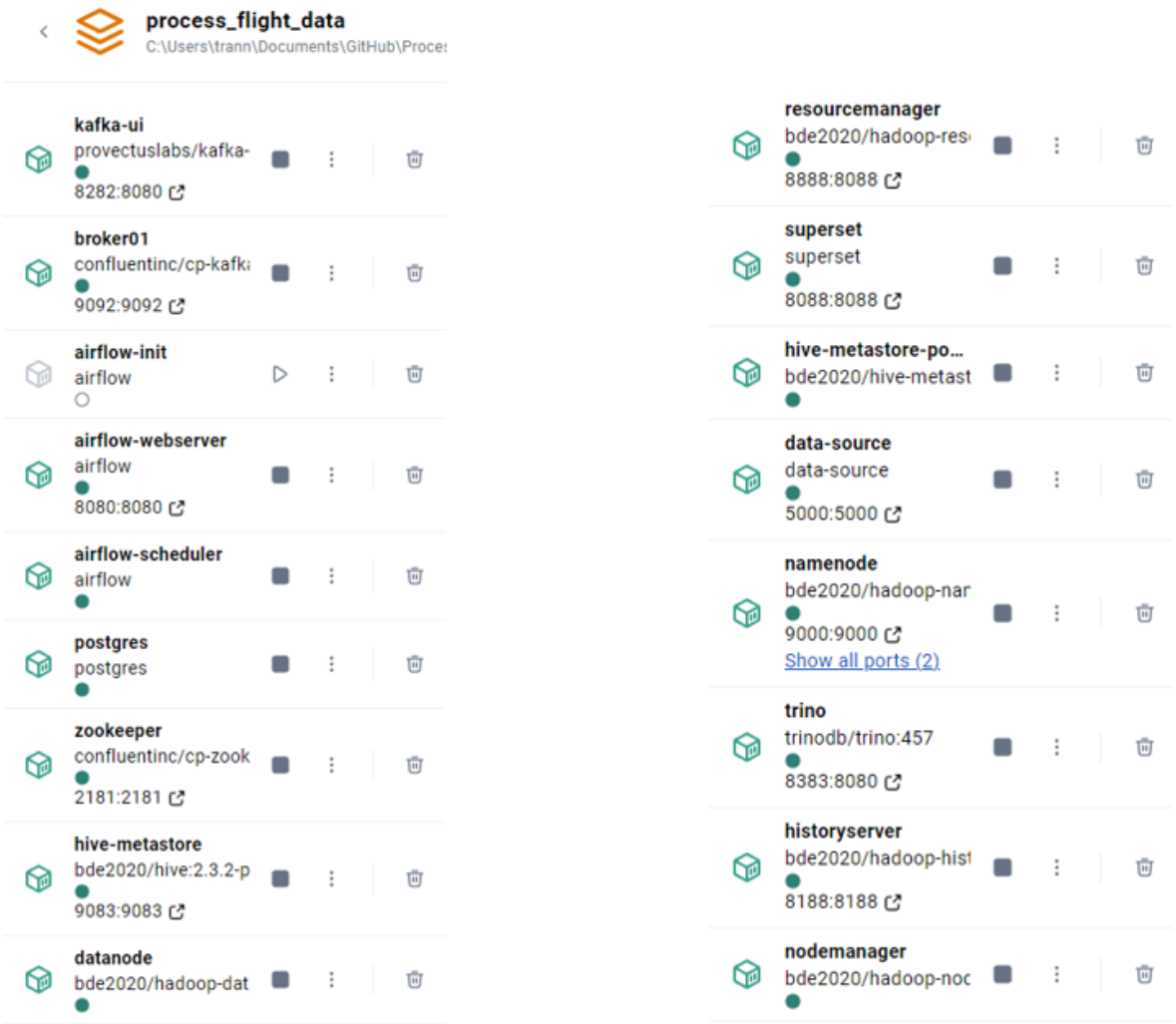
3.7 Mô-đun trực quan hóa dữ liệu

Sau khi truy vấn dữ liệu chuyến bay bằng Trino và tạo các bảng kết quả phân chia theo tháng, quý, năm và giữa các năm, dữ liệu từ các bảng này sẽ được nhập vào Superset dưới dạng datasets để thực hiện trực quan hóa. Quá trình trực quan hóa tổ chức thành các bảng điều khiển (dashboard), bao gồm bảng điều khiển cho từng năm và bảng điều khiển so sánh dữ liệu giữa các năm. Mỗi bảng điều khiển

trình bày dữ liệu qua các biểu đồ (chart) theo từng loại kết quả phân tích, giúp người dùng dễ dàng theo dõi và đánh giá thông tin.

Các biểu đồ được lựa chọn phù hợp với từng báo cáo, bao gồm: biểu đồ cột cho các báo cáo về phân bố chuyến bay, điểm xuất phát, hủy chuyến bay, độ trễ trung bình và phân bố các chuyến bay qua các năm; biểu đồ tròn cho phân tích mạng lưới tiếp thị hàng không; và biểu đồ đường cho số lượng chuyến bay theo từng tháng qua các năm.

CHƯƠNG 4. TRIỂN KHAI HỆ THỐNG



Hình trên là hệ thống được em triển khai bằng cách sử dụng Docker Compose, với một cấu hình phức tạp bao gồm nhiều dịch vụ kết nối qua một mạng riêng (flight_net) nhằm xây dựng một hệ sinh thái xử lý và phân tích dữ liệu lớn. Các dịch vụ trong hệ thống bao gồm Apache Kafka, Apache Airflow, PostgreSQL, HDFS, Trino, Hive và Superset. Các dịch vụ này được cấu hình với các tham số môi trường khác nhau để thiết lập các kết nối và cài đặt cần thiết cho hệ thống.

Trong đó, Apache Airflow được sử dụng làm công cụ điều phối luồng công việc, hỗ trợ chạy và giám sát các tác vụ tự động, với các cài đặt như LocalExecutor, kết nối cơ sở dữ liệu PostgreSQL cho việc lưu trữ thông tin về các DAGs (Directed Acyclic Graphs). PostgreSQL hoạt động như một cơ sở dữ liệu cho Airflow, nơi các metadata của các DAGs và task instances được lưu trữ.

Kafka được sử dụng để xử lý và truyền tải dữ liệu thời gian thực, với các dịch vụ như Zookeeper đảm bảo việc quản lý các node Kafka và Broker01 chịu trách nhiệm lưu trữ và quản lý các message topic. Kafka UI cũng được cấu hình để cung cấp giao diện web cho phép người dùng theo dõi và quản lý các cluster Kafka.

Các dịch vụ HDFS bao gồm NameNode, DataNode, ResourceManager, NodeManager và HistoryServer, cùng với các volumes để lưu trữ dữ liệu phân tán trên HDFS, tạo ra một hệ thống lưu trữ dữ liệu có thể mở rộng và phân tán cho các ứng dụng xử lý dữ liệu lớn.

Trino, một công cụ truy vấn phân tán, được sử dụng để truy vấn dữ liệu từ các hệ thống khác nhau, trong khi Hive được triển khai để quản lý siêu dữ liệu, hỗ trợ truy vấn các dữ liệu đã được lưu trữ trên HDFS. Hệ thống Hive gồm Hive Metastore PostgreSQL và Hive Metastore, giúp quản lý các bảng và dữ liệu được lưu trữ trong HDFS.

Cuối cùng, Superset được sử dụng như một công cụ trực quan hóa dữ liệu, với các cài đặt để kết nối và trực quan hóa dữ liệu từ Trino, cung cấp các dashboard và báo cáo cho người dùng.

Mỗi dịch vụ được cấu hình với các tham số môi trường phù hợp, đảm bảo các kết nối giữa các dịch vụ hoạt động chính xác và hiệu quả, với các cài đặt như môi trường Zookeeper cho Kafka, môi trường Hadoop cho các dịch vụ HDFS và các tham số cấu hình đặc biệt cho Airflow, Hive và Superset. Hệ thống này hỗ trợ tính mở rộng và có thể mở rộng dễ dàng nhờ vào Docker Compose, đồng thời cung cấp các dịch vụ mạnh mẽ cho việc xử lý và phân tích dữ liệu thời gian thực và dữ liệu phân tán quy mô lớn. Các volumes được sử dụng để lưu trữ dữ liệu và nhật ký, giúp duy trì tính toàn vẹn dữ liệu giữa các lần khởi động lại container, và mạng flight_net giúp các container giao tiếp với nhau một cách an toàn và hiệu quả.

CHƯƠNG 5. KẾT QUẢ THỰC NGHIỆM

Kết quả của hệ thống được em trình bày trong slide và được lưu theo đường link sau:

https://github.com/Tran-Ngoc-Bao/Process_Flight_Data/blob/master/demo/demo_final.mp4