__label__label __label__label __label__label

ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN

Ứng dụng nền tảng phân tích dữ liệu lớn vào bài toán phân tích tốc độ di chuyển phương tiện giao thông của thành phố New York

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

Chuyên ngành: TOÁN TIN

Giảng viên hướng dẫn: ThS. Nguyễn Tuấn Dũng

Sinh viên thực hiện: Lưu Liên Thảo

Mã sinh viên: 20206171

HÀ NỘI – 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đồ án

- Mục tiêu: Tìm hiểu về lý thuyết dữ liệu lớn và các nền tảng phân tích dữ liệu lớn. Hiểu và sử dụng được các công cụ phân tích dữ liệu lớn như Hadoop, Spark. Xây dựng được một hệ thống phân tích dữ liệu lớn cơ bản.
- Nội dung: Ứng dụng nền tảng phân tích dữ liệu lớn vào bài toán phân tích tốc độ di chuyển phương tiện giao thông của thành phố New York.

2. Kết quả đạt được

- Đồ án đã bước đầu xây dựng được một hệ thống phân tích dữ liệu lớn: từ bước thu thập, phân tích đến trực quan hóa.
- Trình bày các kiến thức cơ bản về dữ liệu lớn. Xây dựng được kiến trúc một hệ thống đơn giản.
- Ứng dụng được các công cụ phân tích dữ liệu lớn vào giải quyết bài toán.

3. Ý thức làm việc của sinh viên

.....

.....

Hà Nội, ngày 20 tháng 06 năm 2024

Giảng viên hướng dẫn

ThS. Nguyễn Tuấn Dũng

Lời cảm ơn

Lời đầu tiên, em xin gửi lời cảm ơn tới ThS. Nguyễn Tuấn Dũng đã tận tình hướng dẫn, đóng góp ý kiến trong suốt quá trình em thực hiện Đồ án tốt nghiệp. Trong quá trình thực hiện đề tài này em đã tìm hiểu và vận dụng được những kiến thức các Thầy Cô truyền đạt tuy nhiên do bản thân còn những hạn chế nên bản báo cáo của em không tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp của Thầy Cô để báo cáo của em được hoàn thiện hơn. Em xin chân thành cảm ơn!

Hà Nội, tháng 06 năm 2024

Sinh viên

Lưu Liên Thảo

Tóm tắt nội dung Đồ án

1. Dữ liệu lớn và kiến trúc hệ thống phân tích dữ liệu lớn.

2. Giới thiệu bài toán phân tích tốc độ di chuyển của phương tiện giao thông.

3. Ứng dụng tiếp cận phân tích dữ liệu lớn vào giải quyết bài toán.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Bảng ký hiệu và chữ viết tắt

DW:	Data Warehouse
OLAP:	Online analytical processing
ETL:	Extract, Transform, Load
ELT:	Extract, Load, Transform
HDFS:	Hadoop Distributed File System, hệ thống file phân tán Hadoop

Mở đầu

Lý do chọn đề tài

Con người luôn muốn sử dụng thời gian hiệu quả. Tuy nhiên việc này không hề đơn giản. Để tận dụng được thời gian, ta phải cải thiện tốc độ và muốn cải thiện được tốc độ thì cần một hệ thống giao thông mà ta có thể đi đến điểm đích mong muốn của mình trong thời gian ngắn nhất có thể. Khi một hệ thống giao thông cung cấp tốc độ tốt thì mọi người sẽ cố gắng sử dụng hệ thống đó nhiều nhất có thể và sau đó sự ùn tắc xuất hiện. Đây là kẻ thù cuối cùng của tốc độ. Nếu một hệ thống đường bộ bị chiếm bởi một số lượng lớn phương tiện thì không thể cho người điều khiển duy trì tốc độ mong muốn của họ, điều này là hậu quả của sự ùn tắc được tạo ra bởi số lượng phương tiện không bình thường chiếm giữ đường. Chính vì vậy, vai trò của một hệ thống phân tích dữ liệu giao thông trở nên quan trọng. Hệ thống phân tích dữ liệu giao thông thu thập dữ liệu của hệ thống các con đường (dữ liệu về tốc độ chiếm phần quan trọng nhất trong số đó. Nó chủ yếu cho thấy hiệu suất tổng thể của hệ thống giao thông), phân tích chúng và sau đó cung cấp giải pháp phù hợp nhất cho vấn đề.

Dữ liệu bài toán về tốc độ giao thông có thể rất lớn và đa dạng về mặt cấu trúc. Vì vậy, trong đồ án này, em đề xuất một hệ thống phân tích dữ liệu ứng dụng công nghệ dữ liệu lớn trong bài toán phân tích tốc độ giao thông.

Đối tượng và phạm vi

Trong phạm vi đồ án này, em sử dụng 02 bộ dữ liệu:

- **DOT Traffic Speeds NBE**: bộ dữ liệu về tốc độ giao thông của thành phố New York được thu thập từ trang web data.cityofnewyork.us
- Bộ dữ liệu lịch sử thời tiết được thu thập từ trang web open-meteo.com

khu vực lấy dữ liệu là thành phố New York, Hoa Kỳ trong khoảng thời gian từ năm 2021 cho đến nay.

Chương 1

Dữ liệu lớn và kiến trúc hệ thống phân tích dữ liệu lớn

1.1 Tổng quan về dữ liệu lớn

1.1.1 Khái niệm

Big Data (Dữ liệu lớn) là một lượng lớn thông tin đa dạng xuất hiện với khối lượng ngày càng tăng và với tốc độ ngày càng cao. Những dữ liệu này thường được lưu trữ trong cơ sở dữ liệu máy tính và được phân tích bằng phần mềm được thiết kế đặc biệt để xử lý các tập dữ liệu lớn và phức tạp. Độ lớn của Big Data lớn đến mức các phần mềm xử lý dữ liệu truyền thống không có khả năng thu thập, quản lý và xử lý dữ liệu trong một khoảng thời gian hợp lý. Bao nhiêu dữ liệu để đủ gọi là "big" vẫn còn được tranh luận, nhưng nó có thể là các bội số của petabyte – và các dự án lớn nhất với phạm vi exabytes.

Dữ liệu lớn bao gồm nhiều loại dữ liệu khác nhau bao gồm dữ liệu có cấu trúc, dữ liệu bán cấu trúc và dữ liệu phi cấu trúc:

- **Dữ liệu có cấu trúc**: là dữ liệu có định dạng chuẩn hóa để con người cũng như phần mềm có thể truy cập một cách hiệu quả. Loại dữ liệu này thường ở dạng bảng, bao gồm các hàng và cột xác định rõ ràng các thuộc tính dữ liệu.

- **Dữ liệu bán cấu trúc**: loại dữ liệu không tuân theo một mô hình cố định

như dữ liệu có cấu trúc (các bảng trong cơ sở dữ liệu quan hệ) nhưng vẫn có các thẻ hoặc đánh dấu để nhận diện các phần tử dữ liệu. Ví dụ: emails. Email thường có các trường như người gửi, người nhận, chủ đề, và nội dung, được định dạng theo một cách nhất định nhưng không nghiêm ngặt như cơ sở dữ liệu quan hệ.

- **Dữ liệu phi cấu trúc**: loại dữ liệu không có một mô hình hoặc cấu trúc cố định, khiến cho việc lưu trữ và xử lý trở nên phức tạp hơn. Dữ liệu không cấu trúc bao gồm: văn bản, hình ảnh, video, âm thanh,... Dữ liệu không cấu trúc thường cần các phương pháp xử lý và phân tích dữ liệu đặc biệt, như xử lý ngôn ngữ tự nhiên (NLP) và phân tích hình ảnh.

Tùy theo cách chấp nhận khác nhau, một bộ dữ liệu được coi là dữ liệu lớn thường phải đáp ứng đủ 3 hoặc 5 đặc tính tương ứng với 3 hoặc 5 chữ V (V's concept). Theo khái niệm truyền thống, dữ liệu lớn sẽ thoả mãn 3 chữ V bao gồm: Volume, Variety, Velocity. Tuy nhiên, để hoàn thiện hơn khái niệm về dữ liệu lớn 5V's concept đã được định nghĩa với 3 đặc tính cũ và bổ sung thêm 2 đặc tính nữa bao gồm: Veracity, Value.

1. **Volume (Tổng dung lượng lưu trữ)**

   Kích thước của dữ liệu sẽ được đánh giá là có giá trị và có tiềm năng hay không, và để xem xét liệu nó có thể được coi là dữ liệu lớn hay không.

   Với số lượng lớn thông tin hàng ngày liên tục được update trên Internet, ví dụ như: Facebook nhận được gần 350 triệu hình ảnh, hơn 4.5 tỷ lượt like, và gần 10 tỷ tin nhắn, comment mỗi ngày. Vì lý do đó, những kiểu lưu trữ và phân tích dữ liệu truyền thống không cách nào có thể làm được. Nhưng với công nghệ chúng ta đang nói tới đây, nó có thể dễ dàng xử lý và lưu trữ tất cả những thông tin trên các hệ thống chi nhánh nhỏ tách biệt.

2. **Variety (Đa dạng kiểu dữ liệu)**

   Khái niệm này nói về type of data (kiểu dữ liệu) và nature of data (tính chất của dữ liệu). Điều này giúp những người phân tích nó sử dụng hiệu

quả thông tin chi tiết về kết quả. Chúng được tập hợp từ những text (văn bản), image (hình ảnh), sound (âm thanh), video; cộng với nó hoàn thành phần còn thiếu thông qua những thuật toán tổng hợp dữ liệu.

3. **Velocity (Khả năng xử lý tốc độ cao)**

   Trong thời đại ngày nay, tốc độ dữ liệu được tạo ra và xử lý để đáp ứng nhu cầu và thách thức nằm trong con đường tăng trưởng và phát triển. Dữ liệu lớn thường có sẵn trong thời gian thực.

   Tốc độ của một data (dữ liệu) được tạo ra và rồi được chuyển từ nơi này sang nơi khác hiện tại đã đạt đến mức kinh ngạc. Như việc mọi người có thể chat với nhau trên Facebook với tốc độ nhanh chóng trong môi trường mạng hiện nay. Big Data cho phép chúng ta có thể phân tích các thông số của một dữ liệu được tạo ra mà không cần phải lưu chúng xuống database.

4. **Variability (Độ chính xác)**

   Vì đa dạng về các kiểu dữ liệu, nên sự không thống nhất của tập dữ liệu có thể cản trở các quy trình để xử lý và quản lý nó. Do đó, độ chính xác của công nghệ này có thể đảm bảo giúp cho việc giảm bớt sự sai lệch đáng tiếc có thể xảy ra.

5. **Value (Mức độ giá trị của thông tin)**

   Chất lượng dữ liệu của những dữ liệu lấy được có thể thay đổi rất nhiều, điều này sẽ ảnh hưởng rất mạnh đến việc phân tích chính xác những dữ liệu đấy. Ta có thể xem đây là tính chất cũng là khái niệm mà những doanh nghiệp hay nhà nghiên cứu muốn sử dụng và khai thác Big Data phải nắm giữ và am hiểu nó đầu tiên.

1.1.2 Ứng dụng thực tế

Dữ liệu lớn (Big Data) đang có ảnh hưởng lớn đến nhiều lĩnh vực trong cuộc sống và các ngành công nghiệp khác nhau. Dưới đây là một số ví dụ về ứng dụng phân tích dữ liệu lớn trong thực tế:

- Y tế: Cải thiện hệ thống chăm sóc sức khỏe: Công nghệ dữ liệu lớn được sử dụng rộng rãi trong lĩnh vực y học. Nó giúp dự đoán xu hướng bệnh tật, quản lý hồ sơ bệnh nhân và phát triển các phương pháp điều trị mới.
- Nghiên cứu vật lý hạt: Dữ liệu lớn hỗ trợ nghiên cứu về vật lý hạt, từ việc phân tích dữ liệu thử nghiệm đến dự đoán hiệu suất các thiết bị phức tạp.
- Quảng cáo và Tiếp thị: Tối ưu hóa chiến dịch tiếp thị: Big Data giúp dự đoán hành vi của khách hàng, xu hướng thị trường và tạo ra chiến dịch quảng cáo hiệu quả.
- Tùy chỉnh tiếp thị: Dựa vào dữ liệu khách hàng, doanh nghiệp có thể tạo ra chiến dịch tiếp thị cá nhân hóa.
- Truyền thông và Giải trí: Phân tích dữ liệu người dùng: Big Data giúp các công ty truyền thông và giải trí hiểu rõ hơn về sở thích của khán giả, từ đó tạo ra nội dung hấp dẫn và tối ưu hóa trải nghiệm người dùng.
- Chính phủ và Quản lý công việc công cộng: Quản lý dữ liệu dân số và tài chính: Big Data hỗ trợ chính phủ trong việc quản lý dữ liệu dân số, thuế và tài chính. Điều này giúp cải thiện quản lý và dự đoán xu hướng.
- Sản xuất và Tài chính – Ngân hàng: Tối ưu hóa quy trình sản xuất: Big Data giúp tối ưu hóa quy trình sản xuất, từ quản lý nguồn lực đến dự đoán nhu cầu thị trường.
- Phân tích giao dịch tài chính: Dữ liệu lớn hỗ trợ ngân hàng trong việc phát hiện gian lận, dự đoán rủi ro và quản lý tài chính.

Ngoài ra, việc phân tích dữ liệu lớn đã thể hiện vai trò quan trọng của mình trong lĩnh vực giao thông vận tải. Ví dụ việc thu thập và phân tích dữ liệu giao thông như tai nạn, tình trạng đường, tốc độ, và hành vi của người tham gia giao thông. Điều này giúp đề xuất các giải pháp và cải thiện an toàn giao thông. Đồ án này sẽ phân tích về tốc độ của phương tiện giao thông trong lĩnh vực này.

1.2 Kiến trúc hệ thống phân tích dữ liệu lớn

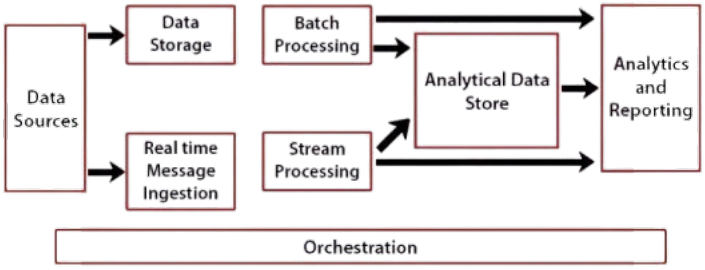
Các hệ thống phân tích dữ liệu lớn hiện nay gồm có bốn lớp chính:

1. **Nhập dữ liệu (Data Ingestion)**: có chức năng thu thập và lưu trữ dữ liệu từ nhiều nguồn khác nhau. Trong kiến trúc Big Data, nhập dữ liệu là quá trình trích xuất dữ liệu từ các nguồn đa dạng và đẩy vào thư mục để lưu trữ. Lớp nhập dữ liệu đóng vai trò quan trọng trong kiến trúc dữ liệu lớn bởi vì nó quyết định dữ liệu sẽ được thu thập, biến đổi và lưu trữ thế nào.

2. **Xử lý dữ liệu (Data Processing)**: là lớp thứ hai trong kiến trúc dữ liệu lớn, có chức năng làm sạch và chuẩn bị dữ liệu sẵn sàng cho việc phân tích. Lớp này đảm bao chất lượng và tính sẵn sàng trong tương lai của dữ liệu.

3. **Lưu trữ dữ liệu (Data Storage)**: đóng vai trò lưu trữ dữ liệu ở định dạng cho phép dữ liệu dễ dàng được truy cập và phân tích, đảm bảo tính khả dụng và sẵn sàng cho các lớp khác truy cập.

4. **Trực quan dữ liệu (Data Visualization)**: trực quan hóa dữ liệu giúp người dùng có thể dễ dàng hiểu và hỗ trợ việc ra quyết định.



Hình 1.1: Kiến trúc hệ thống phân tích dữ liệu lớn

Từ bốn lớp chính, kiến trúc một hệ thống Big Data có thể được cụ thể hóa thành các phần như hình 1.1. Phần dưới đây sẽ đi vào phân tích cụ thể từng phần trong kiến trúc dữ liệu và các công cụ của từng phần đó.

1.2.1 Nguồn dữ liệu (Data Sources)

Nơi dữ liệu được sinh ra, bao gồm dữ liệu có cấu trúc, dữ liệu phi cấu trúc cũng như dữ liệu bán cấu trúc. Dữ liệu có thể đến từ rất nhiều nguồn khác nhau như dữ liệu từ các ứng dụng, cơ sở dữ liệu quan hệ, hoặc dữ liệu file được tạo ra bởi các log của ứng dụng (dữ liệu log ghi lại các lỗi của hệ thống,...), hay dữ liệu thời gian thực từ các thiết bị IoT (hình ảnh theo dõi từ Camera, cảm biến nhiệt độ, độ ẩm,...).

Một số công cụ:

1. **Apache Kafka**

   Apache Kafka là một công cụ mạnh mẽ được sử dụng để thu thập dữ liệu, đặc biệt đối với dữ liệu thực.

Ưu điểm	**Nhược điểm**
• Độ trễ xử lý dữ liệu tối thiểu • Đi kèm với các công cụ giám sát và quản lý • Giữ lại dữ liệu trong một khoảng thời gian để có thể xử lý lại dữ liệu lịch sử • Hệ sinh thái phong phú cung cấp sự tích hợp liền mạch với một loạt các công cụ và nền tảng	• Phức tạp khi cài đặt và bảo trì

2. **Apache Nifi**

   Apache NiFi là một công cụ nhập dữ liệu tiên tiến được thiết kế để thu thập, chuyển đổi và vận chuyển dữ liệu một cách hiệu quả từ nhiều nguồn khác nhau đến đích. Nó có thể xử lý một loạt các loại dữ liệu, bao gồm dữ liệu log và dữ liệu phi cấu trúc. Điều này làm cho nó trở thành một lựa chọn có giá trị cho các kỹ sư và tổ chức dữ liệu đang tìm kiếm một giải pháp linh

hoạt và đáng tin cậy.

Ưu điểm	**Nhược điểm**
• Đa dạng bộ xử lý • Cộng đồng sử dụng lớn • Xử lý dữ liệu thời gian thực • Chứa các tính năng khôi phục dữ liệu trong trường hợp lỗi hệ thống	• Các luồng dữ liệu phức tạp có thể trở nên khó quản lý

3. **Apache Flume**

Ưu điểm	**Nhược điểm**
• Chi phí hợp lý • Tích hợp tốt với hệ sinh thái Hadoop • Tích hợp các cơ chế chống chịu lỗi	• Cấu hình phức tạp và tốn thời gian

1.2.2 Lưu trữ dữ liệu (Data Storage)

Đây chính là hồ dữ liệu (Data Lake), thành phần này được thiết kế để lưu trữ lại khối lượng rất lớn các loại dữ liệu với các định dạng khác nhau được sinh ra bởi nguồn dữ liệu. Đại diện là hệ thống lưu trữ file phân tán, nhằm phục vụ công việc xử lý lượng rất lớn các file dữ liệu cũng như tính an toàn của dữ liệu. Hiện nay có khá nhiều công cụ sử dụng để lưu trữ dữ liệu lớn như: Hadoop Distributed File System (HDFS), Amazon S3.

HDFS phù hợp với các môi trường mà tính cục bộ và sao chép dữ liệu là chìa khóa. Trong khi S3 nổi bật về khả năng mở rộng, độ bền và khả năng tích hợp với các dịch vụ AWS khác.

Định dạng của dữ liệu được lưu trữ trong dữ liệu lớn rất đa dạng. Đối với dữ

liệu có cấu trúc với khối lượng lớn, việc quyết định sử dụng định dạng nào để tối ưu lưu trữ là rất quan trọng.

Apache Parquet là một định dạng lưu trữ cột mã nguồn mở dành cho các hệ sinh thái dữ liệu lớn như Apache Hadoop, Apache Spark và Apache Drill. Parquet được thiết kế để hiệu quả về cả không gian lưu trữ và tốc độ xử lý, đặc biệt là cho các tập dữ liệu lớn.

Ưu điểm của định dạng Parquet

- **Hiệu quả về không gian lưu trữ**: Parquet lưu trữ dữ liệu dưới dạng cột, điều này cho phép nén dữ liệu tốt hơn so với lưu trữ dạng hàng. Điều này giúp tiết kiệm không gian lưu trữ đáng kể.
- **Tốc độ đọc nhanh hơn**: Khi chỉ cần truy cập một số cột trong bảng, Parquet có thể đọc dữ liệu nhanh hơn vì chỉ cần đọc những cột cần thiết mà không phải quét toàn bộ bảng dữ liệu.
- **Tối ưu cho các hệ thống phân tán**: Parquet được thiết kế để làm việc tốt với các hệ thống phân tán như Hadoop và Spark, giúp tối ưu hoá quá trình xử lý và phân tích dữ liệu lớn.
- **Hỗ trợ loại dữ liệu phức tạp**: Parquet hỗ trợ nhiều loại dữ liệu phức tạp như nested structures, arrays và maps, điều này giúp lưu trữ và quản lý các dữ liệu có cấu trúc phức tạp dễ dàng hơn.
- **Tích hợp tốt với nhiều công cụ**: Parquet tương thích với nhiều công cụ và hệ sinh thái dữ liệu lớn như Apache Hadoop, Apache Spark, Apache Drill, Impala, và nhiều hơn nữa.

Nhược điểm của định dạng Parquet

- **Không tối ưu cho ghi dữ liệu nhỏ**: Parquet không phù hợp cho việc ghi dữ liệu nhỏ thường xuyên vì định dạng này tối ưu hóa cho việc ghi dữ liệu lớn thành các khối lớn.

- **Độ phức tạp trong xử lý dữ liệu thời gian thực**: Parquet không phù hợp cho các ứng dụng yêu cầu xử lý dữ liệu thời gian thực hoặc ghi liên tục vì thời gian truy cập và ghi dữ liệu có thể chậm hơn so với các định dạng khác.

Định dạng hàng (row-based formats) như CSV hoặc JSON lưu trữ dữ liệu theo hàng, tức là tất cả các cột của một hàng sẽ được lưu trữ liên tiếp nhau. Các định dạng như CSV hay JSON dễ hiểu và dễ thao tác cho người dùng, đặc biệt là cho những tác vụ đơn giản hoặc dữ liệu nhỏ. Định dạng hàng phù hợp cho các ứng dụng yêu cầu ghi dữ liệu nhỏ và liên tục, vì nó không yêu cầu phải nén dữ liệu hoặc lưu trữ theo khối lớn.

Tuy nhiên, đối với các tập dữ liệu lớn, định dạng hàng không hiệu quả về không gian lưu trữ và tốc độ xử lý, đặc biệt khi chỉ cần truy cập một số cột cụ thể. Định dạng hàng thường gặp khó khăn khi lưu trữ và xử lý các loại dữ liệu phức tạp như nested structures, arrays và maps.

Vì vậy khi làm việc với các ứng dụng phân tích dữ liệu lớn, nơi mà việc nén dữ liệu và tốc độ truy cập cột cụ thể là quan trọng sử dụng định dạng parquet để lưu trữ dữ liệu sẽ tối ưu hơn.

1.2.3 Thu thập dữ liệu thời gian thực (Real time Message Ingestion)

Đây là quá trình liên quan đến việc thu thập dữ liệu ngay khi nó được tạo ra và làm cho nó có tính sẵn sàng sử dụng. Đây là bước nền tảng trong việc phân tích dữ liệu theo thời gian thực

1.2.4 Xử lý luồng (Stream Processing)

Xử lý luồng là một kỹ thuật xử lý các luồng dữ liệu liên tục trong thời gian thực. Xử lý luồng liên quan đến việc xử lý dữ liệu liên tục khi nó được tạo ra, thay vì thu thập và xử lý dữ liệu theo đợt. Phương pháp này cho phép các tổ chức phân tích và phản hồi dữ liệu nhanh chóng, khiến nó đặc biệt có giá trị đối với các ứng dụng như phân tích thời gian thực, giám sát, phát hiện gian lận và

hệ thống đề xuất.

1.2.5 Xử lý theo lô (Batch Processing)

Xử lý hàng loạt liên quan đến việc phân tích và thao tác khối lượng dữ liệu đáng kể trong các khối hoặc lô lớn đã được thu thập trong một khoảng thời gian cụ thể. Thay vì xử lý dữ liệu trong thời gian thực hoặc khi nó đến, xử lý hàng loạt cho phép xử lý một lượng lớn dữ liệu cùng một lúc.

Các trường hợp sử dụng:

- MapReduce: Thích hợp chủ yếu cho xử lý theo lô, nơi hiệu suất là quan trọng hơn và có thể chấp nhận được độ trễ.

- Spark: Linh hoạt hơn, hỗ trợ cả xử lý theo lô và xử lý luồng thời gian thực. Nó cũng được sử dụng cho học máy và xử lý đồ thị. Tốc độ xử lý của Spark nhanh hơn đáng kể so với Hadoop MapReduce. Hiện nay, Spark đã được áp dụng rộng rãi, và các công ty như Netflix, Apple, Facebook và Uber sử dụng một cách triệt để.

1.2.6 Kho dữ liệu phân tích (Analytics-Based Datastore)

Chịu trách nhiệm lưu trữ dữ liệu đã được xử lý theo định dạng có cấu trúc để phục vụ cho các công cụ phân tích dữ liệu (BI Tools). Tuy nhiên, thông thường với các hệ thống dữ liệu lớn sẽ không lưu trữ trực tiếp dữ liệu dưới dạng có cấu trúc mà sẽ tạo ra các bảng (tương tự với các bảng để lưu trữ dữ liệu có cấu trúc) và sau đó ánh xạ tới dữ liệu được lưu trữ trong vùng Data Storage và có thể sử dụng truy vấn SQL trên các bảng này. Một số công cụ phổ biến giúp thực hiện công việc này đó là Apache Hive, Apache Impala,...

Hive cung cấp một cách tiếp cận dễ dùng và quen thuộc cho việc truy vấn và phân tích dữ liệu lớn sử dụng ngôn ngữ truy vấn HiveQL, tương tự như SQL. Các truy vấn trên HiveQL được chuyển đổi thành MapReduce tương ứng thực thi trên cụm máy và trả về kết quả cuối cùng.

Impala, giống như Apache Hive, sử dụng một ngôn ngữ truy vấn khai báo

tương tự, Hive Query Language (HiveQL). Tuy nhiên, Impala truy cập trực tiếp HDFS bằng cách sử dụng các phiên bản dịch vụ hệ thống cục bộ để cung cấp dữ liệu. Không giống như Apache Hive, tương tác trực tiếp như vậy giúp tiết kiệm đáng kể thời gian thực hiện truy vấn, vì kết quả trung gian không phải lưu.

1.2.7 Báo cáo và phân tích (Reporting and Analysis)

Thành phần này cho phép người dùng cuối trực quan hóa dữ liệu (Data Visualization), phân tích dữ liệu, cũng như kết xuất các báo cáo khác nhau.

Không những sử dụng cho trực quan hoá, dữ liệu được lưu trữ trong hồ dữ liệu còn rất thuận lợi cho việc tiến hành các bài toán học máy.

1.2.8 Điều phối (Orchestration)

Điều phối (Orchestration): Thành phần này có nhiệm vụ điều phối các công việc trong một hệ thống dữ liệu lớn để đảm bảo luồng xử lý dữ liệu được thông suốt, từ việc thu thập dữ liệu, lưu trữ dữ liệu đến lọc, tính toán trên dữ liệu. Một số công nghệ đang được sử dụng: Airflow, Pentaho, Apache Oozie, Apache Dolphin...

Chương 2

Giới thiệu bài toán phân tích tốc độ di chuyển của phương tiện giao thông

Tốc độ là một yếu tố quan trọng trong giao thông vận tải vì nó liên quan đến độ an toàn, thời gian, sự thoải mái, tiện lợi và khía cạnh kinh tế. Các nghiên cứu tốc độ tại điểm được sử dụng để xác định phân phối tốc độ của dòng giao thông tại một vị trí cụ thể. Dữ liệu được thu thập trong các nghiên cứu tốc độ tại điểm được sử dụng để xác định phân vị tốc độ của phương tiện, điều này hữu ích trong việc đưa ra nhiều quyết định liên quan đến tốc độ. Dữ liệu tốc độ tại điểm có một số ứng dụng về an toàn, bao gồm các ứng dụng sau đây:

- Quyết định vị trí đặt các biển cảnh báo phù hợp.
- Đánh giá và xác minh các vấn đề về việc lái xe quá tốc độ.
- Đánh giá mối liên quan của tốc độ với các vụ tai nạn.
- Giám sát xu hướng tốc độ giao thông.

Ngoài ra, bằng việc thu thập dữ liệu tốc độ từ các cảm biến, ta có thể phát hiện ra các vị trí mà khi đặt cảm biến ở đó không đo được tốc độ hoặc tốc độ đo được không chính xác. Từ đó, có các biện pháp xử lý kịp thời.

2.1 Nguyên lý hoạt động của máy đo tốc độ

Máy đo tốc độ hiện nay thường được sử dụng nhằm đo tốc độ tức thời của một phương tiện khi nó đi qua một điểm xác định dọc theo con đường, có thể được xác định bằng cách đo thời gian cần thiết để một phương tiện vượt qua một khoảng cách ngắn được chỉ định.

Có hai công nghệ phổ biến được ứng dụng trong máy đo tốc độ hiện nay:

- **Máy đo tốc độ bằng sóng radio**

  Trên máy đo tốc độ này có một radar hay còn gọi là hệ thống dò tìm và định vị bằng sóng vô tuyến. Radar phát ra một chùm sóng vô tuyến với tần số hoạt động xác định.

  Khi một ôtô tiến vào vùng sóng của radar, ngay lập tức một tín hiệu phản xạ điện từ dội lại. Đồng thời tần số sóng radio cũng thay đổi do khoảng cách tương đối giữa radar và xe thay đổi.

  Mức độ tăng hay giảm của tần số sóng radio phụ thuộc vào tốc độ của xe chuyển động trong vùng sóng phát ra từ radar. Nếu tần số tăng, xe đang di chuyển về hướng người cầm máy bắn tốc độ và ngược lại, tần số giảm khi xe di chuyển ra xa người cầm máy. Việc xác định tốc độ của xe dựa vào hiệu ứng vật lý Doppler khi có thể xác định tốc độ chuyển động của một vật nhờ vào sự thay đổi tần số của nguồn sóng phát ra.

- **Máy đo tốc độ bằng tia laser**

  Máy đo tốc độ bằng tia laser tính toán thời gian phản hồi của ánh sáng từ lúc phát ra, tiếp xúc với vật thể di chuyển và dội ngược về máy. Loại máy này có thể phát ra chùm sáng trong 0,3-0,7 giây và "khóa" mục tiêu trong phạm vi 800 m.

  Bằng cách thực hiện thao tác phát ra ánh sáng, thu thập dữ liệu liên tục trong thời gian ngắn, tốc độ của phương tiện đang di chuyển có thể được xác định.

2.2 Định dạng và quy mô của tập dữ liệu

Định dạng

Định dạng của dữ liệu phụ thuộc vào máy đo tốc độ nhưng thường ở dạng bán cấu trúc như json. Bên cạnh dữ liệu được lấy từ máy đo, một số tập dữ liệu cũng được đưa vào phân tích ví dụ: dữ liệu thời tiết (thường có cấu trúc), ảnh chụp từ camera trên đường (thường là dữ liệu bán cấu trúc do đi kèm với dữ liệu metadata chứa thời gian chụp, nhãn về địa lý, và có khi còn có ID của thiết bị).

Vì vậy, dữ liệu dùng để phân tích có thể mang tính đa dạng.

Quy mô

Quy mô của bộ dữ liệu phụ thuộc vào nhiều yếu tố: cơ sở hạ tầng, phạm vi thu thập và nhu cầu phân tích. Việc phân tích tốc độ giao thông thường đi kèm với việc sử dụng nhiều bộ dữ liệu khác như thời tiết, ảnh chụp giao thông, mật độ phương tiện, báo cáo tai nạn,...Dữ liệu được sử dụng có thể được phân tích ngay trong thời gian thực.

Vì vậy dữ liệu sử dụng có thể sẽ rất lớn khiến các hệ thống phân tích dữ liệu truyền thống không đủ tài nguyên để đáp ứng.

Dữ liệu truyền thống là dữ liệu có cấu trúc đang được duy trì chủ yếu bởi tất cả các loại hình doanh nghiệp, bắt đầu từ các tổ chức rất nhỏ đến lớn. Trong hệ thống cơ sở dữ liệu truyền thống, kiến trúc cơ sở dữ liệu tập trung được sử dụng để lưu trữ và duy trì dữ liệu ở định dạng hoặc trường cố định trong một tệp. Để quản lý và truy cập dữ liệu, Ngôn ngữ truy vấn có cấu trúc (SQL) được sử dụng.

Dữ liệu truyền thống được đặc trưng bởi mức độ tổ chức và cấu trúc cao, giúp dễ dàng lưu trữ, quản lý và phân tích. Kỹ thuật phân tích dữ liệu truyền thống liên quan đến việc sử dụng các phương pháp thống kê và trực quan hóa để xác định các mẫu và xu hướng trong dữ liệu.

Dữ liệu truyền thống thường được thu thập và quản lý bởi hệ thống trong doanh nghiệp. Dữ liệu này rất quan trọng để doanh nghiệp đưa ra quyết định sáng suốt và thúc đẩy cải thiện hiệu suất.

2.3 Quy trình phân tích dữ liệu truyền thống

Với kiến trúc truyền thống, dữ liệu từ nhiều nguồn khác nhau (chủ yếu là dữ liệu có cấu trúc) sẽ được đổ vào vùng Staging (vùng tạm) để thực hiện ETL dữ liệu, sau đó đưa vào kho dữ liệu (Data Warehouse). Tuy nhiên, dữ liệu trong vùng Staging sẽ không được lưu trữ lâu dài, sau một thời gian ngắn cũng sẽ bị xoá đi. Kho dữ liệu là nơi lưu trữ dữ liệu và thường sử dụng các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) như MySql hay Oracle làm nơi lưu trữ. Vì vậy, cần phải xác định rõ ràng các lược đồ dữ liệu OLAP, phục vụ việc phân tích đa chiều, bao gồm các bảng Fact và bảng Dimension. Bảng Fact chứa các số liệu đo lường quan trọng. Trong khi đó các bảng Dimension chứa thông tin chi tiết về các khía cạnh phân tích như khách hàng, sản phẩm, thời gian, địa điểm và các thuộc tính khác. Kho dữ liệu sau khi được xây dựng xong có thể kết nối với các công cụ BI nhằm tạo ra các báo cáo phục vụ cho phân tích dữ liệu.

2.4 Nhược điểm của quy trình truyền thống sử dụng hệ quản trị cơ sở dữ liệu quan hệ

Dữ liệu ngày càng tăng lên đáng kể, đạt hàng petabyte (1 petabyte = 1.024 terabyte, 1 terabyte = 1.024 GB). Hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) gặp khó khăn trong việc xử lý khối lượng dữ liệu lớn như vậy. Để giải quyết vấn đề này, RDBMS đã thêm nhiều đơn vị xử lý trung tâm (CPU) hoặc bộ nhớ hơn vào hệ thống quản lý cơ sở dữ liệu để mở rộng theo chiều dọc.

Thứ hai, phần lớn dữ liệu hiện nay đến dạng bán cấu trúc hoặc không cấu trúc từ các nguồn như mạng xã hội, âm thanh, video, văn bản và email. Tuy nhiên, vấn đề thứ hai liên quan đến dữ liệu không cấu trúc nằm ngoài phạm vi của RDBMS vì cơ sở dữ liệu quan hệ không thể phân loại dữ liệu không cấu trúc. Chúng được thiết kế và cấu trúc để chứa dữ liệu có cấu trúc như dữ liệu weblog, cảm biến và tài chính.

Ngoài ra, "big data" được tạo ra với tốc độ rất cao. RDBMS thiếu khả năng

xử lý với tốc độ cao vì nó được thiết kế để duy trì dữ liệu ổn định thay vì tăng trưởng nhanh chóng. Ngay cả khi RDBMS được sử dụng để xử lý và lưu trữ "big data", nó sẽ trở nên rất đắt đỏ.

Do đó, khả năng của cơ sở dữ liệu quan hệ trong việc xử lý "big data" đã dẫn đến sự xuất hiện của các công nghệ mới cho quy trình phân tích dữ liệu với kiến trúc dữ liệu lớn.

2.5 Đề xuất quy trình phân tích dữ liệu lớn cho bài toán

Thấy rằng bài toán phân tích dữ liệu tốc độ giao thông nếu xử lý theo kiến trúc của quy trình phân tích dữ liệu truyền thống sẽ mất nhiều thời gian và không hiệu quả. Vì vậy, em đề xuất một quy trình phân tích dữ liệu lớn để giải quyết bài toán trên.

2.5.1 Công cụ sử dụng

1. **Thu thập dữ liệu**

   Apache Nifi

   Apache Nifi là công cụ mã nguồn mở được xây dựng để tự động hóa luồng dữ liệu từ các hệ thống, nó được viết trên nền Java. Nifi có thể thu thập và xử lý dữ liệu thời gian thực, hỗ trợ lấy dữ liệu từ nhiều nguồn khác nhau. Một số đặc điểm của Nifi:

   - Cung cấp giao diện người dùng dựa trên trình duyệt Web, giúp thiết kế các luồng dữ liệu một cách dễ dàng.
   - Nền tảng này cung cấp khả năng xử lý dữ liệu với tốc độ cao, giúp tăng hiệu suất và khả năng xử lý dữ liệu lớn.
   - Apache Nifi cho phép người dùng thiết lập độ ưu tiên với các luồng dữ liệu. Điều này giúp đảm bảo rằng các tác vụ quan trọng được xử lý trước, đồng thời đảm bảo mức độ ưu tiên có thể thay đổi theo nhu cầu.
   - Apache Nifi hỗ trợ các giao thức bảo mật như SSL, HTTPS, SSH,...

Việc hỗ trợ các giao thức bảo mật và mã hóa này trong Nifi đảm bảo rằng việc truyền dữ liệu thông qua nền tảng này được thực hiện một cách an toàn và bảo mật, giúp bảo vệ dữ liệu quan trọng khỏi sự xâm nhập và lộ thông tin.

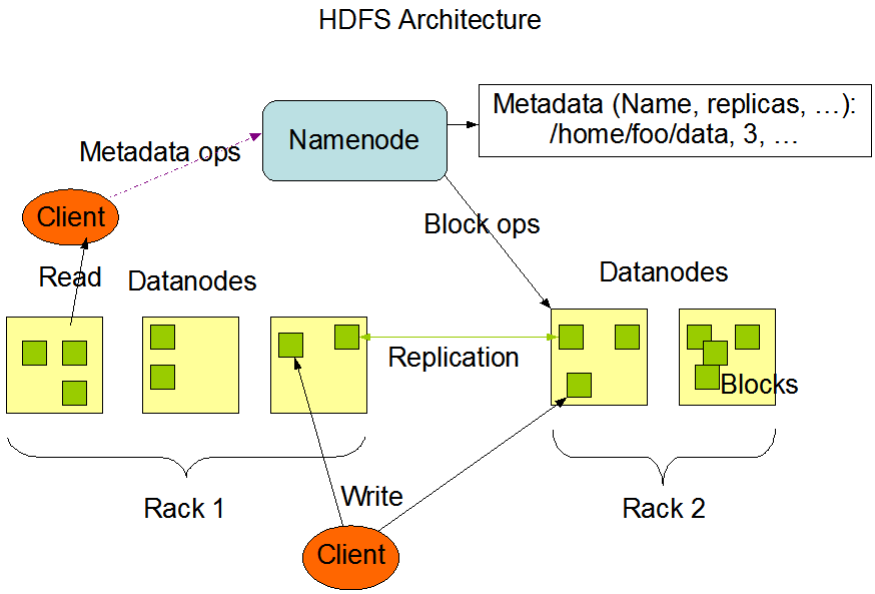
2. **Lưu trữ dữ liệu**

   Hadoop Distributef File System (HDFS)

   Hệ thống tệp phân tán Hadoop (HDFS) là thành phần chính của hệ sinh thái Apache Hadoop, được thiết kế để lưu trữ và quản lý khối lượng lớn dữ liệu trên nhiều máy theo cách phân tán. Nó cung cấp khả năng truy cập dữ liệu thông lượng cao, giúp nó phù hợp với các ứng dụng xử lý các tập dữ liệu lớn, chẳng hạn như phân tích dữ liệu lớn , học máy và lưu trữ dữ liệu. Hệ thống tệp phân tán Hadoop (HDFS) là giải pháp lưu trữ có khả năng mở rộng và có khả năng chịu lỗi được thiết kế cho các bộ dữ liệu lớn. Nó bao gồm NameNode (quản lý siêu dữ liệu), DataNodes (lưu trữ khối dữ liệu) và giao diện máy khách. Các ưu điểm chính bao gồm khả năng mở rộng, khả năng chịu lỗi, thông lượng cao, hiệu quả về chi phí và vị trí dữ liệu, khiến nó trở nên lý tưởng cho các ứng dụng dữ liệu lớn.

   Kiến trúc HDFS

   HDFS được thiết kế để có khả năng mở rộng cao, đáng tin cậy và hiệu quả, cho phép lưu trữ và xử lý các bộ dữ liệu lớn. Kiến trúc của nó bao gồm một số thành phần chính: NameNode, DataNode, Secondary NameNode, HDFS Client, Block Structure.



Hình 2.1: Kiến trúc Hệ thống file phân tán Hadoop HDFS

- **NameNode** là máy chủ chính quản lý không gian tên hệ thống tệp và kiểm soát quyền truy cập vào tệp của khách hàng. Nó thực hiện các hoạt động như mở, đóng và đổi tên các tập tin và thư mục. Ngoài ra, NameNode ánh xạ các khối tệp vào DataNodes, duy trì siêu dữ liệu và cấu trúc tổng thể của hệ thống tệp. Siêu dữ liệu này được lưu trữ trong bộ nhớ để truy cập nhanh và được lưu trên đĩa để đảm bảo độ tin cậy.

  Trách nhiệm chính:

  – Duy trì cây hệ thống tập tin và siêu dữ liệu (metadata).

  – Quản lý ánh xạ các khối tệp tới DataNodes.

  – Đảm bảo tính toàn vẹn dữ liệu và phối hợp sao chép các khối dữ liệu.

- **DataNode** là các nút công nhân (worker nodes) trong HDFS, chịu trách nhiệm lưu trữ và truy xuất các khối dữ liệu thực tế theo hướng dẫn của NameNode. Mỗi DataNode quản lý bộ lưu trữ gắn liền với nó và báo cáo định kỳ danh sách các khối mà nó lưu trữ cho NameNode.

  Trách nhiệm chính:

  – Lưu trữ các khối dữ liệu và phục vụ các yêu cầu đọc/ghi từ máy khách.

– Thực hiện tạo, xóa và sao chép khối theo hướng dẫn từ NameNode.

– Định kỳ gửi báo cáo tới NameNode để xác nhận trạng thái của nó.

- **Secondary NameNode** đóng vai trò là người trợ giúp cho NameNode chính, chịu trách nhiệm chính trong việc hợp nhất EditLogs (lưu lại lịch sử hoạt động khi các thay đổi metadata xảy ra trong HDFS) với hình ảnh hệ thống tệp hiện tại (FsImage) để giảm tải cho NameNode, đảm bảo rằng metadata của hệ thống files được cập nhật và có thể được khôi phục trong trường hợp NameNode bị lỗi.
- **HDFS Client** là giao diện mà qua đó người dùng và ứng dụng tương tác với HDFS. Nó cho phép thực hiện các thao tác tạo, xóa, đọc và ghi tập tin. Máy khách giao tiếp với NameNode để xác định DataNode nào giữ các khối của tệp và tương tác trực tiếp với DataNode cho các hoạt động đọc/ghi dữ liệu thực tế.
- **Block Structure** HDFS lưu trữ các tập tin bằng cách chia chúng thành các khối lớn, thường có kích thước 128MB hoặc 256MB. Mỗi khối được lưu trữ độc lập trên nhiều DataNodes, cho phép xử lý song song và có khả năng chịu lỗi. NameNode theo dõi các vị trí khối và bản sao của chúng.

  Các tính năng chính:

  – Kích thước khối lớn giúp giảm chi phí quản lý số lượng lớn khối.

  – Các khối được sao chép trên nhiều DataNode để đảm bảo tính khả dụng của dữ liệu và khả năng chịu lỗi.

3. **Xử lý dữ liệu**

   Apache Spark

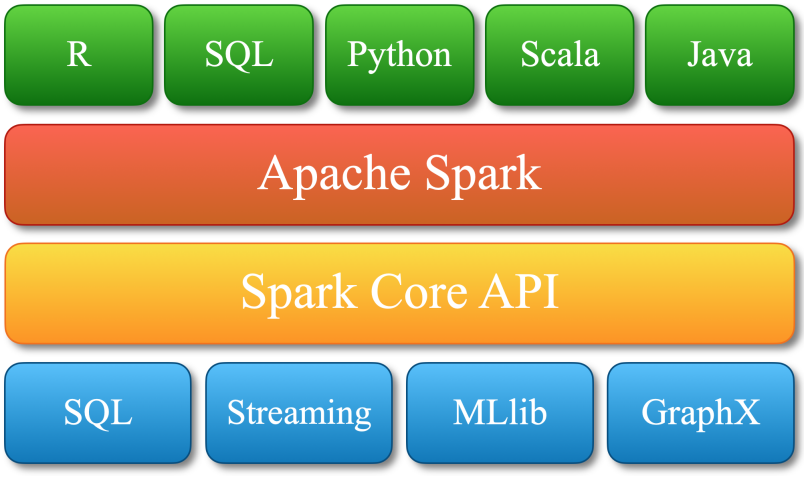
   Apache Spark là một phần mềm mã nguồn mở được phát triển để hỗ trợ xử lý song song dữ liệu lớn trên các cụm máy chủ phân tán.

   Spark xử lý dữ liệu phân tán với tốc độ nhanh và khả năng mở rộng linh hoạt (mở rộng bằng cách thêm node mới). Theo như công bố của chính công

ty làm ra Apache Spark thì Spark có thể xử lý dữ nhiệu nhanh gấp 100 lần so với Hadoop MapReduce (là công cụ xử lý dữ liệu lớn của Hadoop) khi thực hiện trên RAM và 10 lần nếu thực hiện trên ổ đĩa. Có được điều này bởi vì Spark có thể tính toán trực tiếp trên RAM thay vì ổ đĩa như ở Hadoop MapReduce. Ngoài vượt trội về tốc độ xử lý, Spark còn có những ưu điểm sau:

- Spark sử dụng khả năng ghi vào bộ nhớ đệm nằm trong bộ nhớ và thực thi truy vấn tối ưu hóa, giúp truy vấn phân tích nhanh dữ liệu có kích thước bất kỳ.
- Cung cấp các API phát triển bằng ngôn ngữ Java, Scala, Python và R, hỗ trợ tái sử dụng mã trên nhiều khối lượng công việc, chẳng hạn như xử lý lô dữ liệu, truy vấn tương tác, phân tích theo thời gian thực, máy học và xử lý đồ thị.
- Cung cấp nhiều thư viện hỗ trợ cho truy vấn SQL và học máy.
- Hỗ trợ xử lý dữ liệu trong thời gian thực.

Tuy nhiên, Spark đơn thuần chỉ là công cụ xử lý và tính toán dữ liệu, có khả năng xử lý dữ liệu phân tán từ các hệ thống lưu trữ phân tán như HDFS.



Hình 2.2: Hệ sinh thái Apache Spark

Các thành phần của Spark:

- **Spark Core** là thành phần cốt lõi của Apache Spark, các thành phần khác muốn hoạt động đều cần thông qua Spark Core.

  Spark Core cung cấp các chức năng cơ bản để xử lý dữ liệu phân tán (distributed data processing - DDP) bao gồm quản lý bộ nhớ, thiết lập lịch tác vụ và khôi phục lỗi.

- **Spark SQL** cho phép người dùng truy vấn dữ liệu bằng ngôn ngữ SQL trên các tập dữ liệu lớn. Spark SQL cung cấp một cơ chế để tích hợp với các cơ sở dữ liệu quan hệ như MySQL, Oracle hoặc PostgreSQL và xử lý dữ liệu bằng cách sử dụng các tính năng phân tán của Spark.

  Ngoài ra, Spark SQL còn tích hợp tốt với các công cụ ETL và Spark Streaming, điều đó giúp nó trở thành một phần quan trọng trong hệ thống phân tích dữ liệu phân tán và các ứng dụng trực tuyến trong thời gian thực.

- **Spark Streaming** là một module cho phép xử lý dữ liệu trực tiếp trong thời gian thực (real-time). Spark Streaming cung cấp một cơ chế xử lý dữ liệu liên tục (stream processing) bằng cách chia nhỏ dữ liệu thành một chuỗi các microbatch nhỏ hơn và xử lý chúng thông qua API Apache Spark.

  Bên cạnh đó, nó còn tích hợp với các module khác của Apache Spark như Spark SQL và MLlib. Spark Streaming được sử dụng rộng rãi cho các ứng dụng như phân tích dữ liệu trực tuyến, phát hiện sự cố hệ thống và phân tích dữ liệu giám sát trong thời gian thực.

- **Spark MLlib** là thư viện Machine Learning được tích hợp sẵn trong Apache Spark, cung cấp các thuật toán Machine Learning phổ biến giúp ích trong việc xử lý big data.

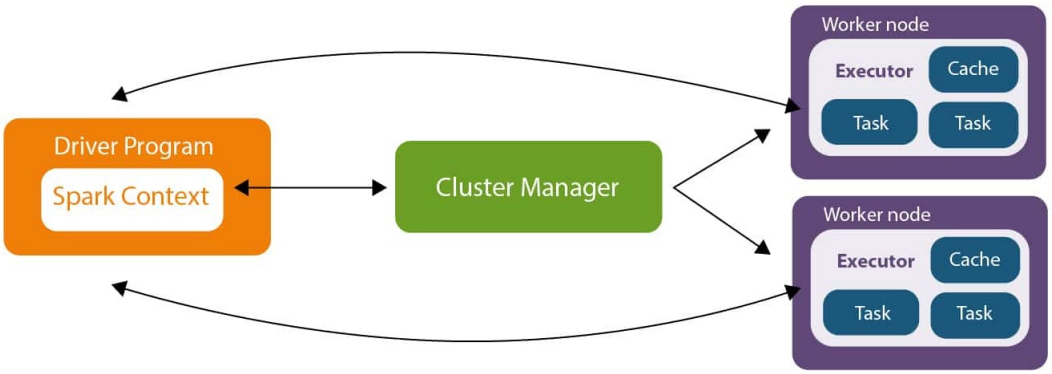
  MLlib cung cấp các thuật toán Machine Learning phổ biến như: Regression, Classification, Clustering, Collaborative Filtering, Dimensionality Reduction, Feature Extraction and Transformation,...

  Spark MLlib cũng tích hợp với các module khác của Apache Spark nhằm

đáp ứng nhu cầu phân tích và xử lý dữ liệu lớn.

- **GraphX** đi kèm với lựa chọn các thuật toán phân tán để xử lý cấu trúc đồ thị.

Kiến trúc Apache Spark



Hình 2.3: Kiến trúc Apache Spark

Kiến trúc của Apache Spark theo kiểu Master-Slave, bao gồm các thành phần chính như Driver Program, Spark Context, Cluster Manager, Executors và Resilient Distributed Datasets (RDDs). Các thành phần này hoạt động cùng nhau để quản lý và thực thi quy trình xử lý dữ liệu trên một môi trường phân tán. Điều này giúp Apache Spark đạt được hiệu suất cao, khả năng mở rộng và độ tin cậy trong việc xử lý dữ liệu lớn và tính toán phân tán.

- **Driver Program:** Đây là chương trình chính của ứng dụng Spark. Nó chạy trên một node trong cluster và quản lý quá trình xử lý trên toàn bộ cluster. Driver Program tạo và quản lý Spark Context.
- **Spark Context:** Spark Context là một đối tượng quan trọng trong Spark. Nó đại diện cho kết nối với một cụm Spark và bao gồm các chức năng cơ bản như quản lý bộ nhớ, lập lịch tác vụ và khôi phục lỗi.
- **Cluster Manager:** Cluster Manager giúp quản lý và phân phối tài nguyên trên các node trong cluster. Nó quản lý việc phân phối và giám

sát quá trình xử lý trên các node để đảm bảo hiệu suất và độ tin cậy.

- **Executors:** Executors là các tiến trình chạy trên các worker nodes trong cluster. Chúng được quản lý bởi driver program để thực hiện các tác vụ xử lý dữ liệu. Mỗi executor có thể chứa nhiều nhiệm vụ được giao để thực hiện.
- **Resilient Distributed Datasets (RDDs):** RDDs là cấu trúc dữ liệu cốt lõi trong Spark. Chúng là tập hợp dữ liệu phân tán và bất biến có thể được xử lý song song trên nhiều node. RDDs được tạo trong Spark Context, được phân phối trên các worker nodes và được lưu trữ trong bộ nhớ cache để tăng hiệu suất.

Driver Program gọi một chương trình và phân chia thành các công việc nhỏ hơn, đồng thời tạo ra SparkContext. SparkContext yêu cầu tài nguyên từ Cluster Manager dựa vào những gì Driver Program đã chia. Cluster Mangager nhận yêu cầu từ SparkContext và trực tiếp đảm nhiệm công việc phân phối tài nguyên (CPU, memory) cho worker nodes. Sau khi tài nguyên được phân bổ, SparkDriver sẽ kiểm soát việc thực thi.

4. **Lưu trữ dữ liệu phân tích Apache Impala**

   Apache Impala là một công cụ truy vấn SQL mã nguồn mở được thiết kế để xử lý hiệu quả dữ liệu quy mô lớn được lưu trữ trong một cụm máy tính chạy Apache Hadoop. Nó cho phép người dùng chạy các truy vấn thời gian thực, tương tác trên các tập dữ liệu khổng lồ với tốc độ đáng kinh ngạc. Là một công cụ xử lý song song hàng loạt, Apache Impala phân phối khối lượng công việc qua nhiều node, cho phép thực hiện các truy vấn song song và tăng tốc quá trình truy xuất và phân tích dữ liệu.

   Apache Impala tích hợp liền mạch với hệ sinh thái Apache Hadoop, cho phép người dùng tận dụng sức mạnh của Hadoop để xử lý và quản lý dữ liệu lớn. Bằng cách cung cấp một giao diện SQL quen thuộc, Impala đơn giản hóa quá trình truy vấn và phân tích dữ liệu cho các nhà phân tích dữ

liệu, nhà khoa học dữ liệu và các chuyên gia kinh doanh, những người có thể không thành thạo về lập trình.

Nhờ tính chất xử lý song song, Apache Impala cho phép có được những cái nhìn phân tích nhanh chóng và khám phá tương tác các tập dữ liệu lớn mà không làm giảm hiệu suất. Điều này làm cho nó trở thành giải pháp lý tưởng cho các tổ chức cần thực hiện truy vấn nhanh trên lượng dữ liệu khổng lồ, hỗ trợ việc ra quyết định dựa trên dữ liệu.

Với tính chất mã nguồn mở, Apache Impala hưởng lợi từ một cộng đồng phát triển và đóng góp sôi động, dẫn đến các bản cập nhật thường xuyên, sửa lỗi và bổ sung tính năng. Điều này đảm bảo rằng người dùng có thể liên tục nâng cao khả năng phân tích dữ liệu của mình và luôn cập nhật với những tiến bộ mới nhất trong công nghệ xử lý song song.

Do Apache Impala thực thi truy vấn trong bộ nhớ nên thời gian truy vấn nhanh hơn Apache Hive.

5. **Điều phối Apache DolphinScheduler**

   Apache DolphinScheduler được ra mắt bởi Analysys Inc. vào năm 2017 và trở thành mã nguồn mở vào tháng 3 năm 2019, sau đó gia nhập Apache incubator vào tháng 8 năm 2019. Hiện tại, hơn 400 công ty đã sử dụng DolphinScheduler trong sản xuất.

   Apache DolphinScheduler là một nền tảng lập lịch công việc nguồn mở hiện đại, được thiết kế để điều phối các quy trình dữ liệu phức tạp và lập lịch tác vụ. Nó nhằm đơn giản hóa việc quản lý các luồng công việc dữ liệu lớn, cung cấp giao diện người dùng trực quan và các khả năng lập lịch mạnh mẽ.

6. **Trực quan hóa dữ liệu**

   Tableau

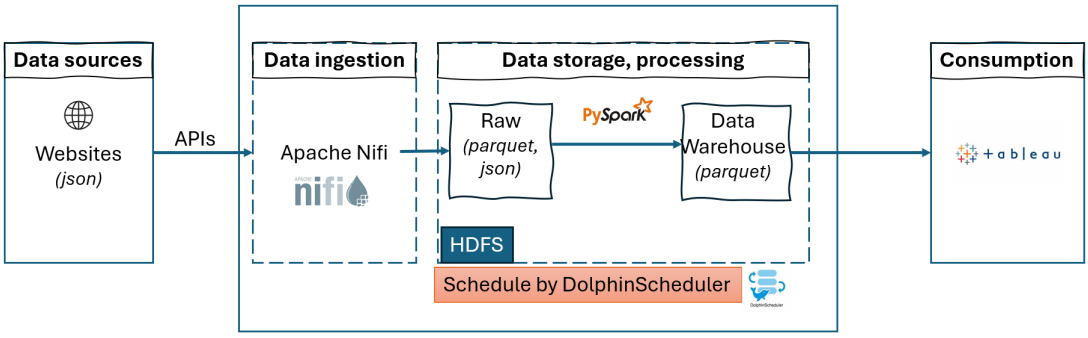
   Tableau là một công cụ phân tích và trực quan hóa dữ liệu mạnh mẽ được phát triển bởi Tableau Software, cho phép người dùng kết nối dữ liệu từ nhiều nguồn khác nhau, thực hiện phân tích dữ liệu, tạo ra các biểu đồ

và báo cáo trực quan. Tableau Desktop có thể chạy trên hai hệ điều hành Wimdows và Mac OS.

Một số ưu điểm của Tableau:

- Tableau cho phép kết nối và tích hợp dữ liệu từ nhiều nguồn khác nhau, có thể kết nối dữ liệu từ cơ sở dữ liệu SQL, tệp tin Excel, bảng tính Google, dịch vụ Web và nhiều nguồn dữ liệu khác. Điều này giúp tổ chức và phân tích dữ liệu từ nhiều nguồn khác nhau trong một bảng điều khiển duy nhất.
- Tableau tích hợp tốt với các công cụ và ngôn ngữ phân tích dữ liệu khác như R và Python. Điều này cho phép người dùng mở rộng khả năng phân tích bằng cách sử dụng các thuật toán phức tạp và tùy chỉnh thông qua các kịch bản R và Python.
- Tableau hoàn toàn vận hành theo công nghệ In-Memory giúp cho tốc độ xử lý các phân tích với tốc độ cao.
- Tableau có thể sử dụng trên máy tính và cả thiết bị di động.
- Tableau có khả năng bảo mật mạnh mẽ, hệ thống phân quyền và xác thực có sẵn giúp Tableau giảm thiểu nguy cơ mất mát dữ liệu.

2.5.2 Kiến trúc hệ thống



Hình 2.4: Kiến trúc hệ thống

1. Thu thập dữ liệu: Dữ liệu được lấy về thông qua bộ xử lý trên Nifi. Dữ liệu lấy về có thể có định dạng json, csv... Sau đó có thể chuyển đổi thành file

parquet để giảm dung lượng trước khi lưu trữ.

2. Lưu trữ dữ liệu: Dữ liệu thô và dữ liệu sau khi được chuyển đổi đều được lữu trữ trên HDFS ở hai thư mục khác nhau (Raw và DataWarehouse).

3. Xử lý dữ liệu: Dùng Spark để xử dữ liệu thô trong thư mục Raw trước khi đẩy vào Data Warehouse.

4. Lập lịch: Toàn bộ quá trình này được cái đặt tự động bằng DolphinScheduler.

5. Lưu trữ dữ liệu phân tích: sử dụng Impala tạo các bảng external có cấu trúc ánh xạ đến dữ liệu được lưu trữ trong thư mục Data Warehouse trên HDFS để dễ dàng thực hiện truy vấn và kết nối với BI-tools.

6. Trực quan hóa dữ liệu: Kết nối Impala với Tableau, dùng Tableau để thể hiện các biểu đồ.

Chương 3

Ứng dụng cách giải quyết bài toán vào tập dữ liệu của thành phố New York

3.1 Đặc điểm bộ dữ liệu

3.1.1 Bộ dữ liệu NYC DOT Speed Traffic

NYC DOT Speed Traffic là bộ dữ liệu mở, được lấy từ dịch vụ cảm biến tốc độ giao thông của Sở Giao thông New York (NYCDOT), cho phép các nhóm người dùng khác nhau (tức là công chúng, tư nhân, các nhà cung cấp thương mại, cơ quan giao thông, nhà nghiên cứu, truyền thông và những người khác) tải xuống thông tin về tốc độ giao thông một cách định kỳ để sử dụng trong ứng dụng và nghiên cứu. Dữ liệu này chứa thông tin về tình trạng giao thông 'thời gian thực' từ các địa điểm mà NYCDOT đã lắp đặt cảm biến, chủ yếu trên các đường lớn và xa lộ chính trong giới hạn của Thành phố. NYCDOT sử dụng thông tin này cho phản ứng khẩn cấp và quản lý.

Tuy nhiên, hiện nay bộ dữ liệu NYC DOT Speed Traffic chỉ được tải về thông qua website data.cityofnewyork.us, và dữ liệu được cập nhật vài lần một ngày chứ không còn có thể lấy theo 'thời gian thực'.

Bộ dữ liệu gồm 13 trường, được mô tả trong file metadata được công bố như bảng 3.1,

Bảng 3.1: Giải thích các trường trong bộ dữ liệu tốc độ theo metadata

Tên trường	**Định nghĩa**
Id	TRANSCOM Link ID
Speed	Tốc độ trung bình của một phương tiện di chuyển giữa các điểm đo trong khoảng thời gian gần nhất
TravelTime	Thời gian trung bình để phương tiện di chuyển trên đoạn đường
Status	Không hữu dụng
DataAsOf	Thời gian cuối cùng dữ liệu được thu thập
linkId	TRANSCOM Link ID (giống ID field)
linkPoints	Chuỗi kinh độ vĩ độ của các cảm biến đo tốc độ
EncodedPolyLine	Google compatible polyline từ http://code.google.com/apis /maps/documentation/polylineutility.html
EncodedPolyLineLvls	Google compatible poly levels từ http://code.google.com/apis /maps/-documentation/polylineutility.html
Owner	Chủ sở hữu máy đo
Transcom_id	Không hữu dụng
Borough	Quận của Thành phố New York (Brooklyn, Bronx, Manhattan, Queens, Staten Island)
LinkName	Mô tả vị trí con đường

Đặc điểm của bộ dữ liệu lấy về:

- **Khoảng thời gian**: 01/01/2021 - nay
- **Dung lượng**: 15.1GB (tính đến 09/05/2024)
- **Định dạng**: csv

Bộ dữ liệu sau đó sẽ được chuyển sang định dạng parquet để lưu trữ lâu dài. Dữ liệu lấy về hàng ngày ở định dạng json, sau đó cũng được biến đổi trực tiếp trên Apache Nifi thành định dạng parquet trước khi được lưu trữ trong thư mục HDFS.

3.1.2 Bộ dữ liệu về thời tiết

Bộ dữ liệu lịch sử thời tiết được lấy từ các trạm thời tiết, máy bay, radar và vệ tinh thông qua website open-meteo.com. Dữ liệu gồm các trường:

Tên trường	Định nghĩa
time	Thời gian theo định dạng yyyy-MM-dd HH:00:00
temperature	Nhiệt độ ($°C$)
rain	Lượng mưa (mm)
snowfall	Lương tuyết rơi (cm)
cloud_cover	Độ che phủ của mây (%)

Đặc điểm của bộ dữ liệu lấy về:

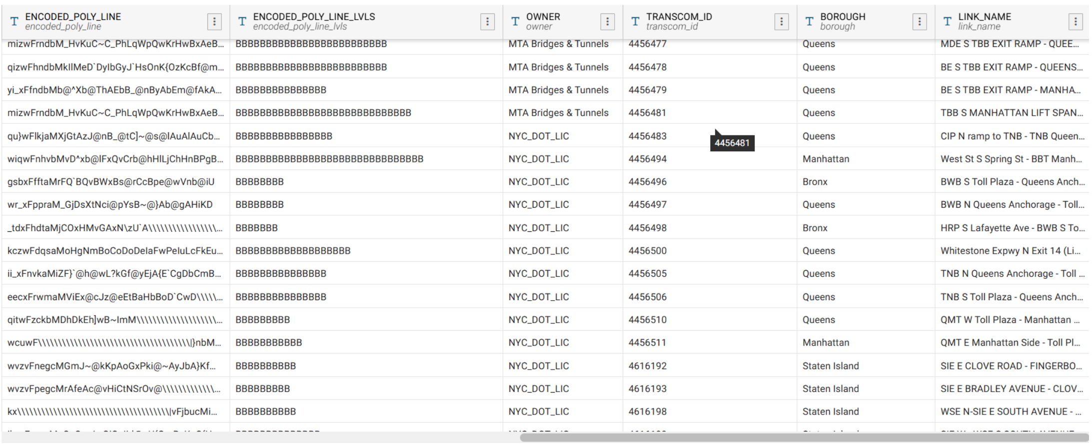
- **Khoảng thời gian**: 01/01/2021 - nay
- **Định dạng**: json

Bộ dự liệu sau đó sẽ được chuyển sang định dạng parquet để lưu trữ lâu dài. Dữ liệu lấy về hàng ngày ở định dạng json.

3.2 Thu thập và chuẩn hóa dữ liệu

3.2.1 Phân tích yêu cầu

Bộ dữ liệu NYC DOT Speed Traffic Dữ liệu ban đầu đã được chuẩn hóa thành dạng bảng như hình 3.1.



ENCODED_POLY_LINE encoded_poly_line	ENCODED_POLY_LINE_LVLS encoded_poly_line_lvls	OWNER owner	TRANSCOM_ID transcom_id	BOROUGH borough	LINK_NAME link_name
mizwFrndbM_HvKuC~C_PhLqWpQwKrHwBxAeB…	BBBBBBBBBBBBBBBBBBBBBBBBB	MTA Bridges & Tunnels	4456477	Queens	MDE S TBB EXIT RAMP - QUEE…
qizwFhndbMkIlMeD`DylbGyJ`HsOnK{OzKcBf@m…	BBBBBBBBBBBBBBBBBBBBBBBBB	MTA Bridges & Tunnels	4456478	Queens	BE S TBB EXIT RAMP - QUEENS…
yi_xFfndbMb@^Xb@ThAEbB_@nByAbEm@fAkA…	BBBBBBBBBBBBBBB	MTA Bridges & Tunnels	4456479	Queens	BE S TBB EXIT RAMP - MANHA…
mizwFrndbM_HvKuC~C_PhLqWpQwKrHwBxAeB…	BBBBBBBBBBBBBBBBBBBBBBBBBBBBB	MTA Bridges & Tunnels	4456481	Queens	TBB S MANHATTAN LIFT SPAN…
qu}wFlkjaMXjGtAzJ@nB_@tC]~@s@lAuAlAuCb…	BBBBBBBBBBBBBBBB	NYC_DOT_LIC	4456483	Queens	CIP N ramp to TNB - TNB Queen…
wiqwFnhvbMvD^xb@lFxQvCrb@hHlLjChHnBPgB…	BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB	NYC_DOT_LIC	4456494	Manhattan	West St S Spring St - BBT Manh…
gsbxFfftaMrFQ`BQvBWxBs@rCcBpe@wVnb@iU	BBBBBBBB	NYC_DOT_LIC	4456496	Bronx	BWB S Toll Plaza - Queens Anch…
wr_xFppraM_GjDsXtNci@pYsB~@}Ab@gAHiKD	BBBBBBBB	NYC_DOT_LIC	4456497	Queens	BWB N Queens Anchorage - Toll…
_tdxFhdtaMjCOxHMvGAxN\zU`A\\\\\\\\\\\\\\\\…	BBBBBBB	NYC_DOT_LIC	4456498	Bronx	HRP S Lafayette Ave - BWB S To…
kczwFdqsaMoHgNmBoCoDoDelaFwPeluLcFkEu…	BBBBBBBBBBBBBBBBBBB	NYC_DOT_LIC	4456500	Queens	Whitestone Expwy N Exit 14 (Li…
ii_xFnvkaMiZF}`@h@wL?kGf@yEjA{E`CgDbCmB…	BBBBBBBBBBBBBBB	NYC_DOT_LIC	4456505	Queens	TNB N Queens Anchorage - Toll …
eecxFrwmaMViEx@cJz@eEtBaHbBoD`CwD\\\\…	BBBBBBBBBBBBBBB	NYC_DOT_LIC	4456506	Queens	TNB S Toll Plaza - Queens Anch…
qitwFzckbMDhDkEh]wB~ImM\\\\\\\\\\\\\\\\\\…	BBBBBBBBB	NYC_DOT_LIC	4456510	Queens	QMT W Toll Plaza - Manhattan …
wcuwF\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|}nbM…	BBBBBBBBBBB	NYC_DOT_LIC	4456511	Manhattan	QMT E Manhattan Side - Toll Pl…
wvzvFnegcMGmJ~@kKpAoGxPki@~AyJbA}Kf…	BBBBBBBBBB	NYC_DOT_LIC	4616192	Staten Island	SIE E CLOVE ROAD - FINGERBO…
wvzvFpegcMrAfeAc@vHiCtNSrOv@\\\\\\\\\\\\…	BBBBBBBBB	NYC_DOT_LIC	4616193	Staten Island	SIE E BRADLEY AVENUE - CLOV…
kx\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|vFjbucMi…	BBBBBBBBBB	NYC_DOT_LIC	4616198	Staten Island	WSE N-SIE E SOUTH AVENUE - …

Hình 3.1: Bảng dữ liệu NYC DOT Speed Traffic

ID id	SPEED speed	TRAVEL_TIME travel_time	STATUS status	DATA_AS_OF data_as_of	LINK_ID link_id	LINK_POINTS link_points
378	14.29	184	0	2024 May 31 06:39:10 AM	4616197	40.6210105,-74.168861 40.620760...
381	14.29	110	0	2024 May 31 06:39:10 AM	4616194	40.608031,-74.13212 40.60759,-74....
377	14.29	271	0	2024 May 31 06:39:10 AM	4616195	40.61486,-74.15738 40.60931,-74.1...
350	0.00	0	-101	2024 May 31 06:39:10 AM	4616196	40.63092,-74.14592 40.62975,-74.1...
439	60.27	47	0	2024 May 31 06:39:10 AM	4616200	40.5900505,-74.19356 40.59141,-74...
441	60.27	131	0	2024 May 31 06:39:10 AM	4616201	40.56058,-74.199581 40.568141,-74...
436	60.27	86	0	2024 May 31 06:39:10 AM	4616202	40.56058,-74.199581 40.55926,-74....
385	60.89	23	0	2024 May 31 06:39:10 AM	4616208	40.6077805,-74.14091 40.60826,-74...
390	60.89	64	0	2024 May 31 06:39:10 AM	4616209	40.6152105,-74.157401 40.61231,-7...
351	53.43	100	0	2024 May 31 06:39:10 AM	4616210	40.63092,-74.14592 40.62975,-74.1...
388	60.89	46	0	2024 May 31 06:39:10 AM	4616211	40.6151706,-74.15738 40.61739,-74...
434	58.40	55	0	2024 May 31 06:39:10 AM	4616212	40.6020904,-74.1877 40.600331,-74...
430	58.40	135	0	2024 May 31 06:39:10 AM	4616213	40.5902,-74.19332 40.57748,-74.19...
431	58.40	86	0	2024 May 31 06:39:10 AM	4616214	40.56042,-74.199391 40.55924,-74....
382	0.00	0	-101	2024 May 31 06:39:10 AM	4616216	40.63089,-74.14569 40.6298,-74.14...
383	0.00	0	-101	2024 May 31 06:39:10 AM	4616217	40.6167105,-74.15242 40.61572,-74...
159	41.63	133	0	2024 May 31 06:39:10 AM	4616252	40.8563506,-73.87233 40.85219,-73...

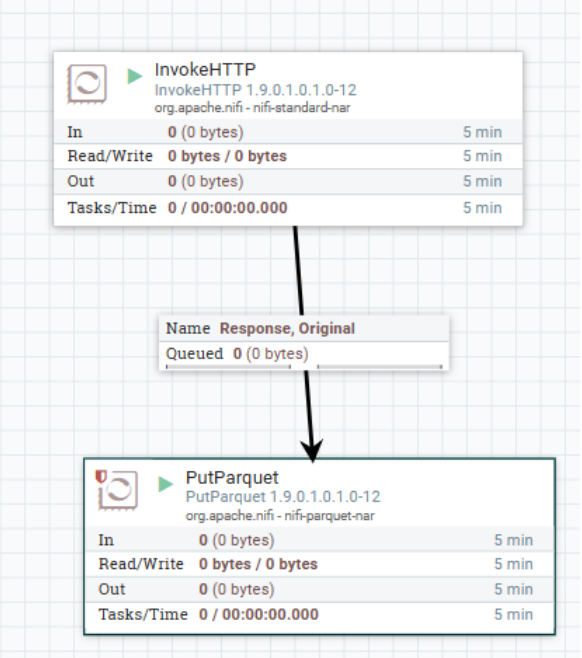
Hình 3.2: Bảng dữ liệu NYC DOT Speed Traffic

Dữ liệu không cần chuẩn hóa. Tuy nhiên, dữ liệu lịch sử khi lấy về ở dạng csv đã được chuyển đổi sang parquet để lưu trữ. Vì vậy, dữ liệu hàng ngày khi được lấy về dưới định dạng json gồm khoảng 50,000 objects/ngày cũng cần chuyển thành định dạng parquet để lưu trữ cho đồng nhất.

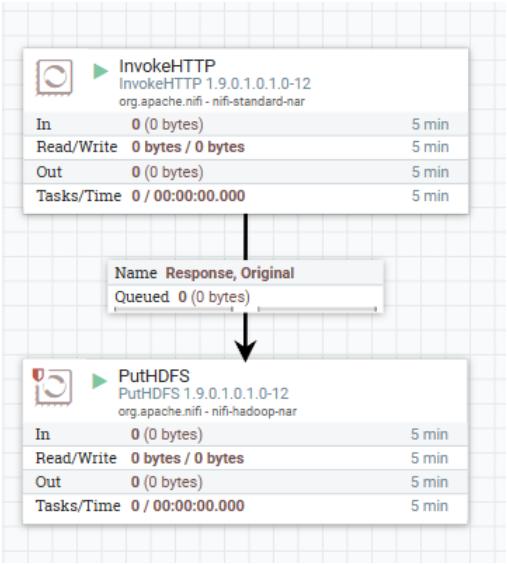
Bộ dữ liệu về thời tiết Dữ liệu được lấy về luôn có định dạng json cố định nên không cần chuẩn hóa.

3.2.2 Thu thập và chuẩn hóa dữ liệu

Cả hai bộ dữ liệu đều được lấy về theo ngày thông qua API, sử dụng Processor InvokeHTTP của Nifi và được cài đặt chạy vào 12:00 trưa hàng ngày. Do dữ liệu lấy về phụ thuộc hoàn toàn vào thời gian cập nhật bộ dữ liệu data.cityofnewyork.us nên sẽ có độ trễ trung bình là 02 ngày so với thời gian thực, tức là dữ liệu ngày hôm nay lấy về sẽ là dữ liệu của hai ngày trước đó.



Hình 3.3: Luồng lấy dữ liệu NYC DOT Speed Traffic trên Nifi



Hình 3.4: Luồng lấy dữ liệu Thời tiết trên Nifi

Dữ liệu tốc độ sau khi lấy về sẽ tự động chuyển đổi từ json thành file Parquet và đẩy vào HDFS bằng processor PutParquet. Còn đối với bộ dữ liệu thời tiết với định dạng json sẽ được đẩy trực tiếp lên HDFS bằng Processor PutHDFS.

Dữ liệu lấy về sau khi đọc bằng Spark có dạng như hình 3.5.

```
>>> df = spark.read.parquet('/MyData/raw/DOT_Traffic_Speeds_NBE')
24/06/05 14:40:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
>>> df.show()
+---+-----+-----------+------+-------------------+-------+--------------------+--------------------+--------------------+-----------+----------+-----------+--------------------+---+
| ID|SPEED|TRAVEL_TIME|STATUS|          DATA_AS_OF|LINK_ID|         LINK_POINTS|   ENCODED_POLY_LINE|ENCODED_POLY_LINE_LVLS|      OWNER|TRANSCOM_ID|   BOROUGH|           LINK_NAME|day|
+---+-----+-----------+------+-------------------+-------+--------------------+--------------------+--------------------+-----------+----------+-----------+--------------------+---+
|212|49.70|        218|     0|10/31/2022 11:59:...|4362244|40.78802,-73.7900...|cl}wFtbkaMfFiIdBy...|  BBBBBBBBBBBBBBBB...|NYC-DOT-Region 10|   4362244|     Queens|CVE NB LIE - WILL...|  1|
|211|10.56|       1541|  -101|10/31/2022 11:59:...|4362250|40.78795,-73.7901...|uk}wFtckaMfH}Kr@c...|  BBBBBBBBBBBBBBBB...|NYC-DOT-Region 10|   4362250|     Queens|CVE NB GCP - WILL...|  1|
|168|46.60|         93|     0|10/31/2022 11:59:...|4456496|40.814761,-73.836...|gsbxFfftaMrFQ`BQv...|            BBBBBBBB|    NYC_DOT_LIC|   4456496|      Bronx|BWB S Toll Plaza ...|  1|
|164|48.46|         90|     0|10/31/2022 11:59:...|4456497|40.79932,-73.8280...|wr_xFppraM_GjDsXt...|            BBBBBBBB|    NYC_DOT_LIC|   4456497|     Queens|BWB N Queens Anch...|  1|
|411| 0.00|          0|  -101|10/31/2022 11:59:...|4763652|40.6040405,-74.05...|gnyvF~i~bMcCsKaPy...|               BBBBB|Verrazano-Narrows...|   4763652|  Brooklyn|VNB E SI GANTRY U...|  1|
|364|18.01|        298|     0|10/31/2022 11:59:...|4456511|40.745726,-73.973...|wcuwF\\\\\\\\\\\\...|          BBBBBBBBBB|    NYC_DOT_LIC|   4456511| Manhattan|QMT E Manhattan S...|  1|
|205|55.30|        292|  -101|10/31/2022 11:59:...|4362247|40.78819,-73.7905...|em}wFvekaMwByHaAw...|  BBBBBBBBBBBBBBBB...|NYC-DOT-Region 10|   4362247|     Queens|    CIP NB GCP - TNB|  1|
|318|45.98|         86|     0|10/31/2022 11:59:...|4362249|40.7442206,-73.77...|kztwFzogaMqFeOu@w...|  BBBBBBBBBBBBBBBB...|NYC-DOT-Region 10|   4362249|     Queens|LIE WB LITTLE NEC...|  1|
|319|51.57|        185|     0|10/31/2022 11:59:...|4362251|40.7537504,-73.74...|}uvwFlebaMkMcUcIk...|        BBBBBBBBBBBB|NYC-DOT-Region 10|   4362251|     Queens|LIE WB LITTLE NEC...|  1|
|206|54.05|        217|     0|10/31/2022 11:59:...|4362252|40.7889706,-73.79...|ar}wFzekaMDaEy@kJ...|  BBBBBBBBBBBBBBBB...|NYC-DOT-Region 10|   4362252|     Queens|CIP N LIE ramp - TNB|  1|
|213|36.03|         66|     0|10/31/2022 11:59:...|4456450|40.80069,-73.9287...|i{_xFzefbMyBvGUlA...|  BBBBBBBBBBBBBBBB...|MTA Bridges & Tun...|   4456450| Manhattan|FDR N - TBB E 116...|  1|
|399|33.55|         72|     0|10/31/2022 11:59:...|4456452|40.8011005,-73.92...|{}_xFzcfbMqAvLTxA...|  BBBBBBBBBBBBBBBBBB|MTA Bridges & Tun...|   4456452| Manhattan|TBB W - FDR S MAN...|  1|
|395|44.11|        184|     0|10/31/2022 11:59:...|4456476|40.7723,-73.91983...|{izwF\\\\\\\\\\\\...|  BBBBBBBBBBBBBBBB...|MTA Bridges & Tun...|   4456476|     Queens|TBB N QUEENS ANCH...|  1|
|398|43.49|        173|     0|10/31/2022 11:59:...|4456481|40.77223,-73.9199...|mizwFrndbM_HvKuC~...|  BBBBBBBBBBBBBBBB...|MTA Bridges & Tun...|   4456481|     Queens|TBB S MANHATTAN L...|  1|
|124| 0.62|       6236|  -101|10/31/2022 11:59:...|4456501|40.68036,-74.0044...|gkhwFp~tbMoJdGkHn...|          BBBBBBBBBB|MTA Bridges & Tun...|   4456501| Manhattan|BBT W Toll Plaza ...|  1|
|119|35.41|        199|     0|10/31/2022 11:59:...|4456502|40.70631,-74.0150...|mmmwFx`wbMxDt@rAb...|        BBBBBBBBBBBB|MTA Bridges & Tun...|   4456502| Manhattan|BBT E Manhattan P...|  1|
|402|46.60|        140|     0|10/31/2022 11:59:...|4456505|40.797815,-73.793...|ii_xFnvkaMlZF}^@h...|      BBBBBBBBBBBBBB|    NYC_DOT_LIC|   4456505|     Queens|TNB N Queens Anch...|  1|
|406|47.84|        138|     0|10/31/2022 11:59:...|4456506|40.81763,-73.8036...|eecxFrwmaMViEx@cJ...|      BBBBBBBBBBBBBB|    NYC_DOT_LIC|   4456506|     Queens|TNB S Toll Plaza ...|  1|
|365|30.44|        179|     0|10/31/2022 11:59:...|4456510|40.741534,-73.954...|qitwFzckbNDhDkEh]...|            BBBBBBBB|    NYC_DOT_LIC|   4456510|     Queens|QMT W Toll Plaza ...|  1|
|417| 0.00|          0|  -101|10/31/2022 11:59:...|4763649|40.60414,-74.0524...|{nyvFpj~bMaC{KcP{...|               BBBBB|Verrazano-Narrows...|   4763649|  Brooklyn|VNB W BROOKLYN GA...|  1|
+---+-----+-----------+------+-------------------+-------+--------------------+--------------------+--------------------+-----------+----------+-----------+--------------------+---+
only showing top 20 rows
```

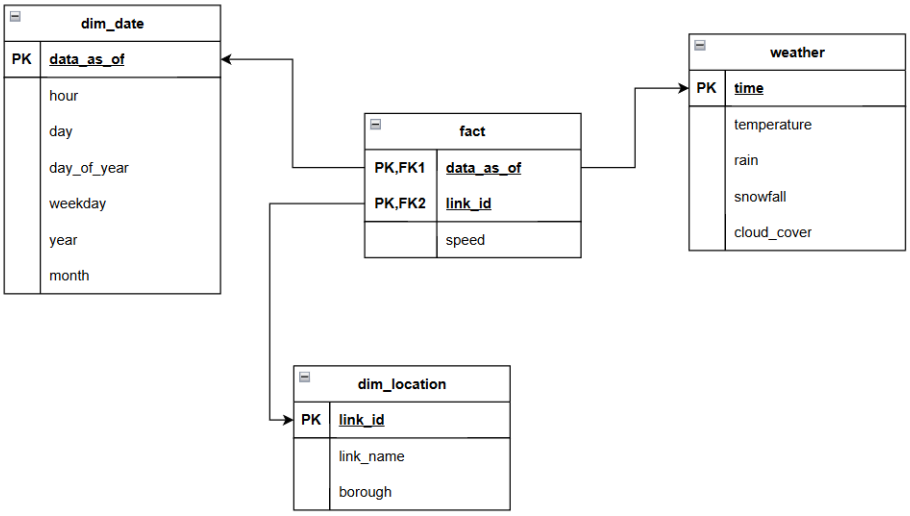
Hình 3.5: Dữ liệu tốc độ đọc bằng PySpark

```
>>> df = spark.read.json('/MyData/raw/weather')
24/06/03 11:15:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
>>> df.show()
+---------+------------------+--------------------+--------------------+--------+---------+----------------+----------------------+--------------------+--------+
|elevation|  generationtime_ms|              hourly|         hourly_units|latitude|longitude|        timezone|timezone_abbreviation|utc_offset_seconds|     day|
+---------+------------------+--------------------+--------------------+--------+---------+----------------+----------------------+------------------+--------+
|     14.0|  4.757046699523926|{[0, 0, 1, 9, 16,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240511|
|     14.0|0.41306018829345703|{[100, 100, 100, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240527|
|     14.0| 0.3739559413208008|{[100, 100, 100, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240516|
|     14.0|0.28908252716064453|{[100, 100, 92, 8...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240526|
|     14.0| 0.0699758529663086|{[93, 90, 94, 100...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240515|
|     14.0|0.37097930908203125|{[10, 0, 0, 2, 22...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240517|
|     14.0| 0.3769397735595703|{[50, 46, 47, 68,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240518|
|     14.0|0.33605098724365234|{[16, 63, 42, 91,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240519|
|     14.0| 0.2859830856323242|{[29, 10, 30, 27,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240523|
|     14.0|0.42808055877685547|{[0, 25, 38, 30, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240514|
|     14.0|0.34296512603759766|{[92, 47, 72, 80,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240530|
|     14.0|0.41794776916503906|{[2, 2, 1, 82, 0,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240520|
|     14.0|0.41306018829345703|{[28, 29, 23, 26,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240521|
|     14.0|0.29289722442626953|{[0, 0, 0, 21, 21...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240525|
|     14.0| 1.3380050659179688|{[38, 5, 27, 41, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240531|
|     14.0| 0.7890462875366211|{[29, 29, 25, 15,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240522|
|     14.0| 0.3439188003540039|{[100, 100, 90, 4...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240528|
|     14.0| 0.2779960632324219|{[0, 11, 8, 8, 26...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240524|
|     14.0|      0.06103515625|{[0, 0, 0, 0, 0, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240529|
|     14.0| 0.2709627151489258|{[93, 98, 59, 72,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                   EDT|            -14400|20240512|
+---------+------------------+--------------------+--------------------+--------+---------+----------------+----------------------+------------------+--------+
only showing top 20 rows
```

Hình 3.6: Dữ liệu thời tiết đọc bằng PySpark

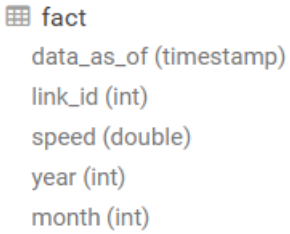
3.3 Xây dựng mô hình dữ liệu OLAP

Với nhu cầu phân tích xu hướng tốc độ theo thời gian, địa điểm và xét mối tương quan với dữ liệu thời tiết, nên từ những dữ liệu được thu thập có thể xây dựng mô hình OLAP như hình 3.7.

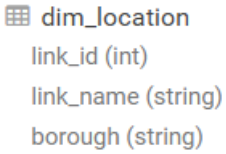


Hình 3.7: Mô hình OLAP

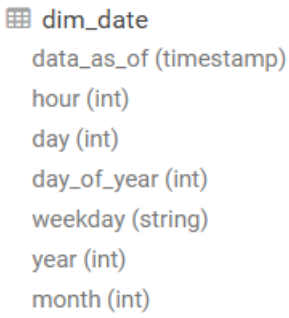
Mỗi bảng có các trường cụ thể như hình 3.8 đến 3.11.



Hình 3.8: Các trường trong bảng fact



Hình 3.9: Các trường trong bảng dim_location



Hình 3.10: Các trường trong bảng dim_date

Hình 3.11: Các trường trong bảng weather

3.4 Tổng hợp và làm giàu dữ liệu

3.4.1 Đối với bộ dữ liệu tốc độ

Bảng fact

Từ dữ liệu ban đầu lấy ra các trường SPEED, DATA_AS_OF, LINK_ID để tạo thành bảng fact. Sau đó lọc bỏ các hàng null, trùng lặp và lọc ra các hàng có giá trị SPEED > 0.

Các trường sử dụng có kiểu dữ liệu ban đầu:

```
>>> df.dtypes
[('ID', 'string'), ('SPEED', 'string'), ('TRAVEL_TIME', 'string'), ('STATUS', 'string'), ('DATA_AS_OF', 'string'), ('LINK_ID', '
string'), ('LINK_POINTS', 'string'), ('ENCODED_POLY_LINE', 'string'), ('ENCODED_POLY_LINE_LVLS', 'string'), ('OWNER', 'string'),
 ('TRANSCOM_ID', 'string'), ('BOROUGH', 'string'), ('LINK_NAME', 'string'), ('day', 'int')]
```

Hình 3.12: Kiểu dữ liệu ban đầu của các trường trong bộ dữ liệu tốc độ

Cột DATA_AS_OF có định dạng là MM/dd/yyyy HH:mm:ss a (ví dụ: 10/31/2022 11:59:04 PM). Cần chuyển đổi trường SPEED từ kiểu dữ liệu string về kiểu dữ liệu double và trường DATA_AS_OF về kiểu dữ liệu timestamp có định dạng là MM/dd/yyyy HH:mm:ss để dễ dàng cho việc tổng hợp theo giờ.

Sau đó cột SPEED sẽ được tổng hợp lại theo giờ và tuyến đường và lấy giá trị trung bình.

	data_as_of	link_id	speed	year	month
1	2024-04-30 02:00:00	4362314	49.49500115712484	2024	4
2	2024-04-29 05:00:00	4616267	7.400000095367432	2024	4
3	2024-04-28 11:00:00	4616272	22.365000247955322	2024	4
4	2024-04-28 05:00:00	4616250	64.71999963124593	2024	4
5	2024-04-30 09:00:00	4456450	30.545832951863606	2024	4
6	2024-04-29 16:00:00	4362251	52.81250031789144	2024	4
7	2024-04-29 10:00:00	4616353	39.55583302179972	2024	4
8	2024-04-29 10:00:00	4362249	24.385833263397217	2024	4
9	2024-04-30 07:00:00	4456506	41.21250041325887	2024	4
10	2024-04-28 23:00:00	4456450	35.9316668510437	2024	4
11	2024-04-28 20:00:00	4616210	48.51583353678385	2024	4
12	2024-04-29 07:00:00	4456498	45.30333375930786	2024	4
13	2024-04-28 19:00:00	4616271	39.96916675567627	2024	4
14	2024-04-29 01:00:00	4616282	45.5641663869222	2024	4
15	2024-04-23 02:00:00	4616259	40.643333752950035	2024	4

Hình 3.13: Bảng fact

Bảng dim_location

Từ dữ liệu ban đầu lấy ra các trường LINK_ID, LINK_NAME, BOROUGH lọc bỏ null và trùng lặp.

	link_id	link_name	borough
1	4329507	LINCOLN TUNNEL W NORTH TUBE NY - NJ	MANHATTAN
2	4616319	GOW S 9TH STREET - 7TH AVENUE	BROOKLYN
3	4616196	MLK S - SIE E WALKER STREET - WOOLEY AVENUE	STATEN ISLAND
4	4616323	12th Ave S 57th St - 45th St	MANHATTAN
5	4456476	TBB N QUEENS ANCHORAGE - MANHATTAN LIFT SPAN	QUEENS
6	4616225	TBB N QUEENS ANCHORAGE - BE N	QUEENS
7	4763657	GOW S VNB W 92ND STREET - BKLYN GANTRY LOWER LEVEL	BROOKLYN
8	4763652	VNB E SI GANTRY UPPER LEVEL - BROOKLYN GANTRY UPPER LEVEL	BROOKLYN
9	4616261	BE S Griswold - Castle Hill Avenue	BRONX
10	4456496	BWB S Toll Plaza - Queens Anchorage	BRONX
11	4456502	BBT E Manhattan Portal - Toll Plaza	MANHATTAN
12	4362314	TNB S Qns Anchorage - CIP S @ TNB	QUEENS
13	4456510	QMT W Toll Plaza - Manhattan Side	QUEENS
14	4763651	VNB W BROOKLYN GANTRY LOWER LEVEL - SI GANTRY LOWER LEVEL	BROOKLYN
15	4616312	BWB S Queens Anchorage - WSE S Exit 14 (Linden Pl)	QUEENS

Hình 3.14: Bảng dim_location

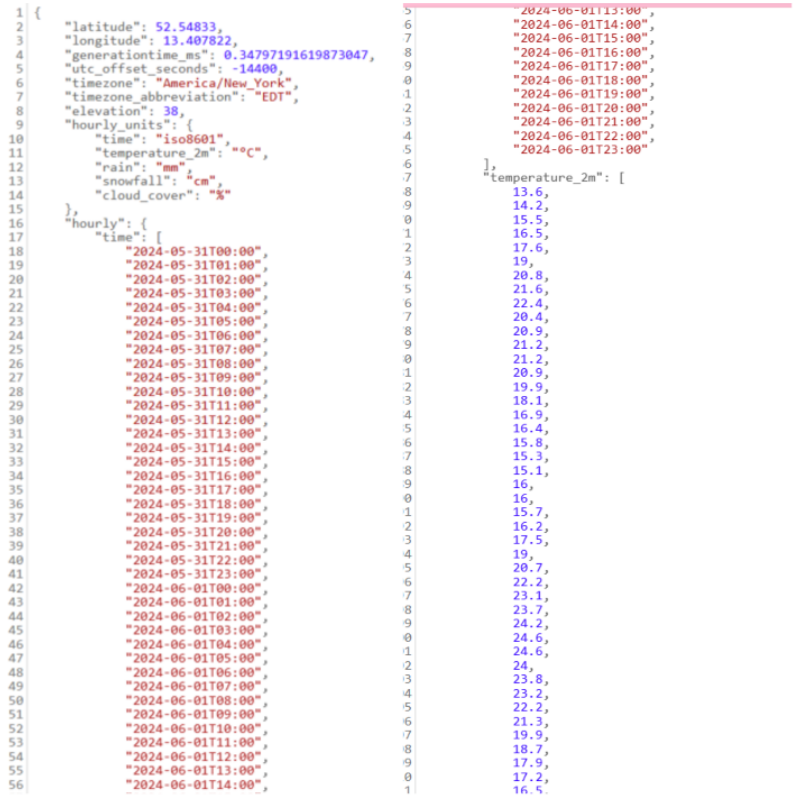
Bảng dim_date

Dữ liệu trong bảng dim_date được sinh độc lập với bộ dữ liệu lấy mốc thời gian từ ngày 01/01/2021 đến nay. Có trường thời gian tuân theo định dạng của trường DATA_AS_OF trong bảng fact.

	data_as_of	hour	day	day_of_year	weekday	year	month
1	2020-12-31 17:00:00	0	1	1	Fri	2021	1
2	2020-12-31 18:00:00	1	1	1	Fri	2021	1
3	2020-12-31 19:00:00	2	1	1	Fri	2021	1
4	2020-12-31 20:00:00	3	1	1	Fri	2021	1
5	2020-12-31 21:00:00	4	1	1	Fri	2021	1
6	2020-12-31 22:00:00	5	1	1	Fri	2021	1
7	2020-12-31 23:00:00	6	1	1	Fri	2021	1
8	2021-01-01 00:00:00	7	1	1	Fri	2021	1
9	2021-01-01 01:00:00	8	1	1	Fri	2021	1
10	2021-01-01 02:00:00	9	1	1	Fri	2021	1
11	2021-01-01 03:00:00	10	1	1	Fri	2021	1
12	2021-01-01 04:00:00	11	1	1	Fri	2021	1
13	2021-01-01 05:00:00	12	1	1	Fri	2021	1
14	2021-01-01 06:00:00	13	1	1	Fri	2021	1
15	2021-01-01 07:00:00	14	1	1	Fri	2021	1

Hình 3.15: Bảng dim_date

3.4.2 Đối với bộ dữ liệu thời tiết



Hình 3.16: Dữ liệu thời tiết dạng json

```
>>> df = spark.read.json('/MyData/raw/weather')
24/06/03 11:15:08 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
>>> df.show()
+---------+-------------------+--------------------+--------------------+---------+---------+---------------+--------------------+--------------------+--------+
|elevation|  generationtime_ms|              hourly|        hourly_units| latitude|longitude|       timezone|timezone_abbreviation|utc_offset_seconds|     day|
+---------+-------------------+--------------------+--------------------+---------+---------+---------------+--------------------+--------------------+--------+
|     14.0|  4.757046699523926|{[0, 0, 1, 9, 16,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240511|
|     14.0|0.41306018829345703|{[100, 100, 100, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240527|
|     14.0| 0.3739595413208008|{[100, 100, 100, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240516|
|     14.0|0.28908252716064453|{[100, 100, 92, 8...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240526|
|     14.0| 0.0699758529663086|{[93, 90, 94, 100...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240515|
|     14.0|0.37097930908203125|{[10, 0, 0, 2, 22...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240517|
|     14.0| 0.3769397735595703|{[50, 46, 47, 68,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240518|
|     14.0|0.33605098724365234|{[16, 63, 42, 91,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240519|
|     14.0| 0.2859830856323242|{[29, 10, 30, 27,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240523|
|     14.0|0.42808055877685547|{[0, 25, 38, 30, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240514|
|     14.0|0.34296512603759766|{[92, 47, 72, 80,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240530|
|     14.0|0.41794776916503906|{[2, 2, 1, 82, 0,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240520|
|     14.0|0.41306018829345703|{[28, 29, 23, 26,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240521|
|     14.0|0.29289722442626953|{[0, 0, 0, 21, 21...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240525|
|     14.0| 1.3380050659179688|{[38, 5, 27, 41, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240531|
|     14.0| 0.7890462875366211|{[29, 29, 25, 15,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240522|
|     14.0| 0.3439188003540039|{[100, 100, 90, 4...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240528|
|     14.0| 0.2779960632324219|{[0, 11, 8, 8, 26...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240524|
|     14.0|      0.06103515625|{[0, 0, 0, 0, 0, ...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240529|
|     14.0| 0.2709627151489258|{[93, 98, 59, 72,...|{%, mm, cm, °C, i...|40.738136|-73.91489|America/New_York|                 EDT|              -14400|20240512|
+---------+-------------------+--------------------+--------------------+---------+---------+---------------+--------------------+--------------------+--------+
only showing top 20 rows
```

Hình 3.17: Dữ liệu thời tiết raw

```
>>> df.dtypes
[('elevation', 'double'), ('generationtime_ms', 'double'), ('hourly', 'struct<cloud_cover:array<bi
gint>,rain:array<double>,snowfall:array<double>,temperature_2m:array<double>,time:array<string>>')
, ('hourly_units', 'struct<cloud_cover:string,rain:string,snowfall:string,temperature_2m:string,ti
me:string>'), ('latitude', 'double'), ('longitude', 'double'), ('timezone', 'string'), ('timezone_
abbreviation', 'string'), ('utc_offset_seconds', 'bigint'), ('day', 'int')]
>>>
```

Hình 3.18: Kiểu dữ liệu của bảng thời tiết

Cột hourly chứ tất cả các trường cần thiết cho bảng weather. Có thể thấy dữ liệu trong cột hourly bao gồm time, temperature, rain, snowfall và cloudcover ở định dạng array. Dữ liệu ở đây có sự phân cấp nên đòi hỏi phải tách một cột thành nhiều cột. Sau đó cần chuẩn hóa định dạng thời gian của cột time về cùng định dạng với trường DATA_AS_OF trong bảng fact.

	time	temperature	rain	snowfall	cloud_cover
1	2024-05-14 17:00:00	14.5	0.1	0	93
2	2024-05-14 18:00:00	14.2	0.4	0	90
3	2024-05-14 19:00:00	13.9	0.2	0	94
4	2024-05-14 20:00:00	13.7	0.1	0	100
5	2024-05-14 21:00:00	13.7	0.3	0	100
6	2024-05-14 22:00:00	13.5	0.2	0	100
7	2024-05-14 23:00:00	13.5	0.1	0	91
8	2024-05-15 00:00:00	13.9	0.1	0	95
9	2024-05-15 01:00:00	14.7	0	0	90
10	2024-05-15 02:00:00	15	1	0	94
11	2024-05-15 03:00:00	15.7	1.1	0	100
12	2024-05-15 04:00:00	16.4	1	0	100
13	2024-05-15 05:00:00	17.7	0.3	0	56
14	2024-05-15 06:00:00	18.2	0.2	0	89
15	2024-05-15 07:00:00	18.9	0.1	0	94

Hình 3.19: Bảng weather

3.5 Tích hợp mô hình dự đoán tắc nghẽn

3.5.1 Phân tích yêu cầu

Dự báo tắc nghẽn là một công việc cần thiết để người điều khiển phương tiện giao thông có thể đoán trước tình hình giao thông trên đoạn đường mình sẽ đi qua. Từ đó, quyết định trước lộ trình phù hợp để đảm bảo thời gian và công việc.

Thời gian tắc nghẽn phụ thuộc vào giờ trong ngày (thường giờ đi làm và tan tầm đường sẽ đông hơn), vào ngày trong tuần (ngày làm việc đường sẽ đông hơn), vào ngày cụ thể trong năm (ví dụ ngày nghỉ lễ,...), hoặc có thể phụ thuộc vào thời tiết (mưa, tuyết).

3.5.2 Tích hợp mô hình

Xây dựng mô hình dự báo đơn giản sử dụng cây ra quyết định Decision Tree. Dữ liệu đầu vào gồm các trường "LINK_ID", "HOUR", "NWEEKDAY", "DAY_OF_YEAR" và dữ liệu đầu ra là kết quả dự đoán có tắc đường hay không ở trường "prediction". Nếu có tắc đường thì giá trị ở trường "prediction" là 1, nếu không tắc đường thì có giá trị 0.

Để đơn giản về mặt tính toán, do các con đường trong bộ dữ liệu đều là các đại lộ lớn nên nếu tốc độ trung bình trong giờ đó nhỏ hơn hoặc bằng 20 thì coi như có sự tắc nghẽn và ngược lại.

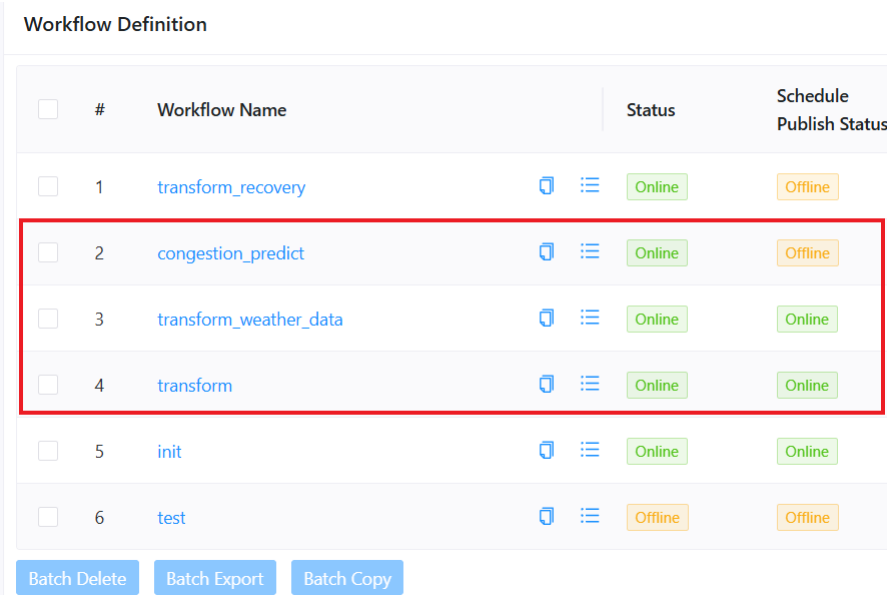
Độ chính xác của mô hình là 82.56%.

	link_id	data_as_of	prediction
1	4329472	2024-06-04 17:00:00	0
2	4329472	2024-06-04 18:00:00	0
3	4329472	2024-06-04 19:00:00	0
4	4329472	2024-06-04 20:00:00	0
5	4329472	2024-06-04 21:00:00	0
6	4329472	2024-06-04 22:00:00	0
7	4329472	2024-06-04 23:00:00	0

Hình 3.20: Đầu ra của mô hình dự báo tắc nghẽn

3.6 Xây dựng luồng và lập lịch

Sau khi tổng hợp tính toán, hệ thống được lập lịch chạy tự động hàng ngày trên DolphinScheduler.



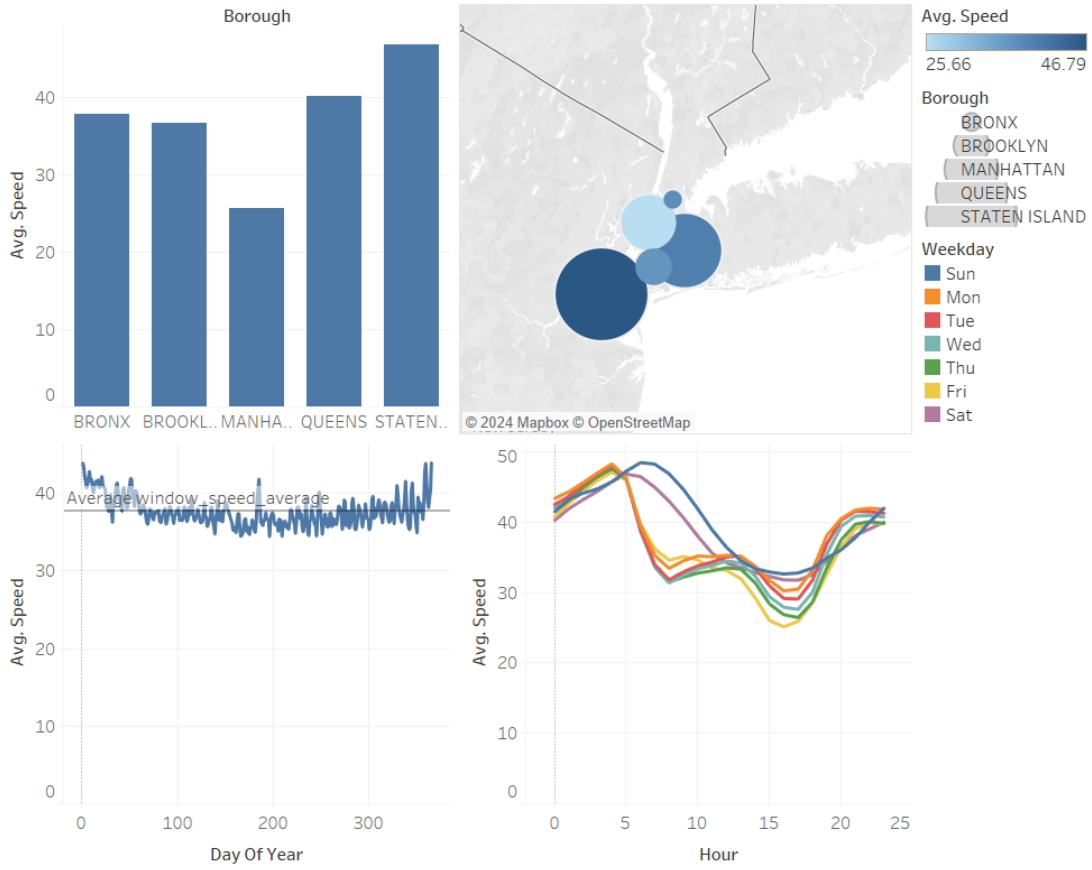
Hình 3.21: Workflows trên DolphinScheduler

Trong đó, chức năng của các luồng là:

- **transform**: chuyển đổi dữ liệu trong bộ dữ liệu tốc độ DOT Traffic Speed NBE.
- **transform_weather_data**: chuyển đổi dữ liệu trong bộ dữ liệu thời tiết.
- **congestion_predict**: dự báo tắc đường theo giờ.

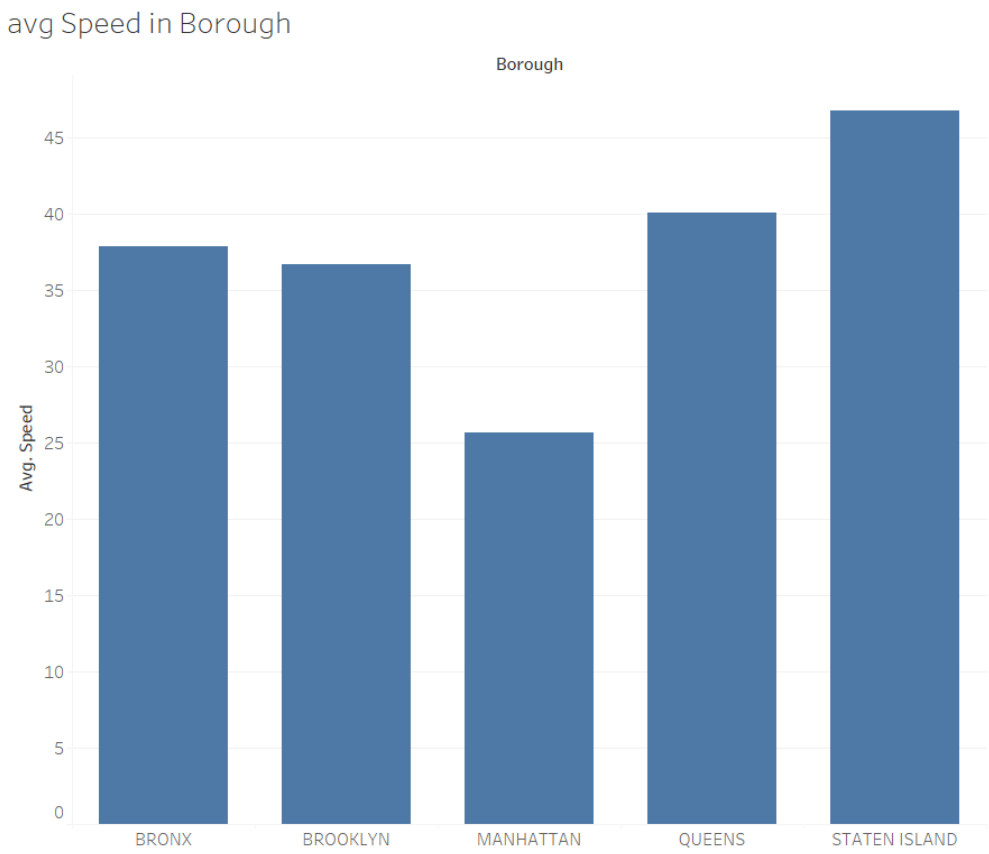
Mỗi luồng trên chỉ chứa một file .py làm chức năng tương ứng.

3.7 Dashboard

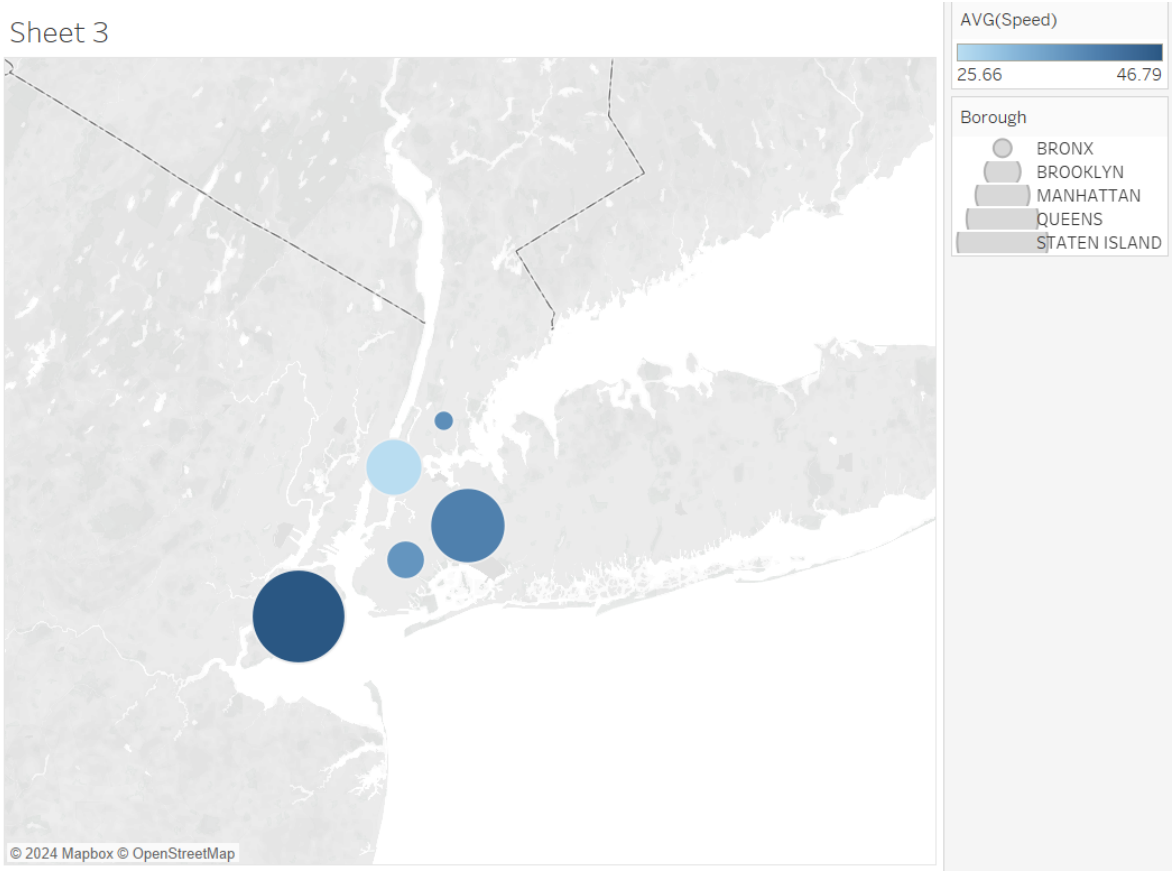


Hình 3.22: Dashboard

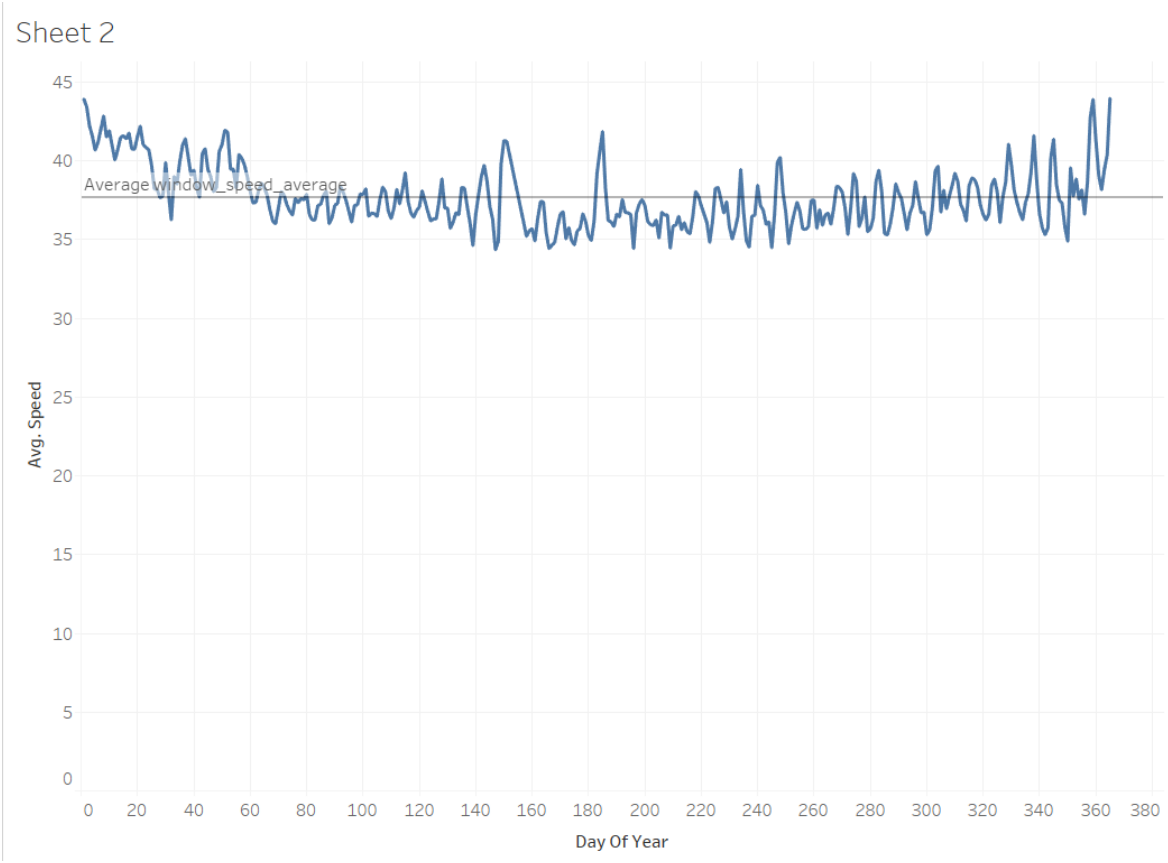
Dễ dàng nhận thấy tốc độ trung bình đo được tại quận Manhattan thấp nhất (khoảng 25km/h), so với quận có tốc độ trung bình cao nhất là Staten Island (khoảng 46km/h).



Hình 3.23: Tốc độ trung bình theo từng quận

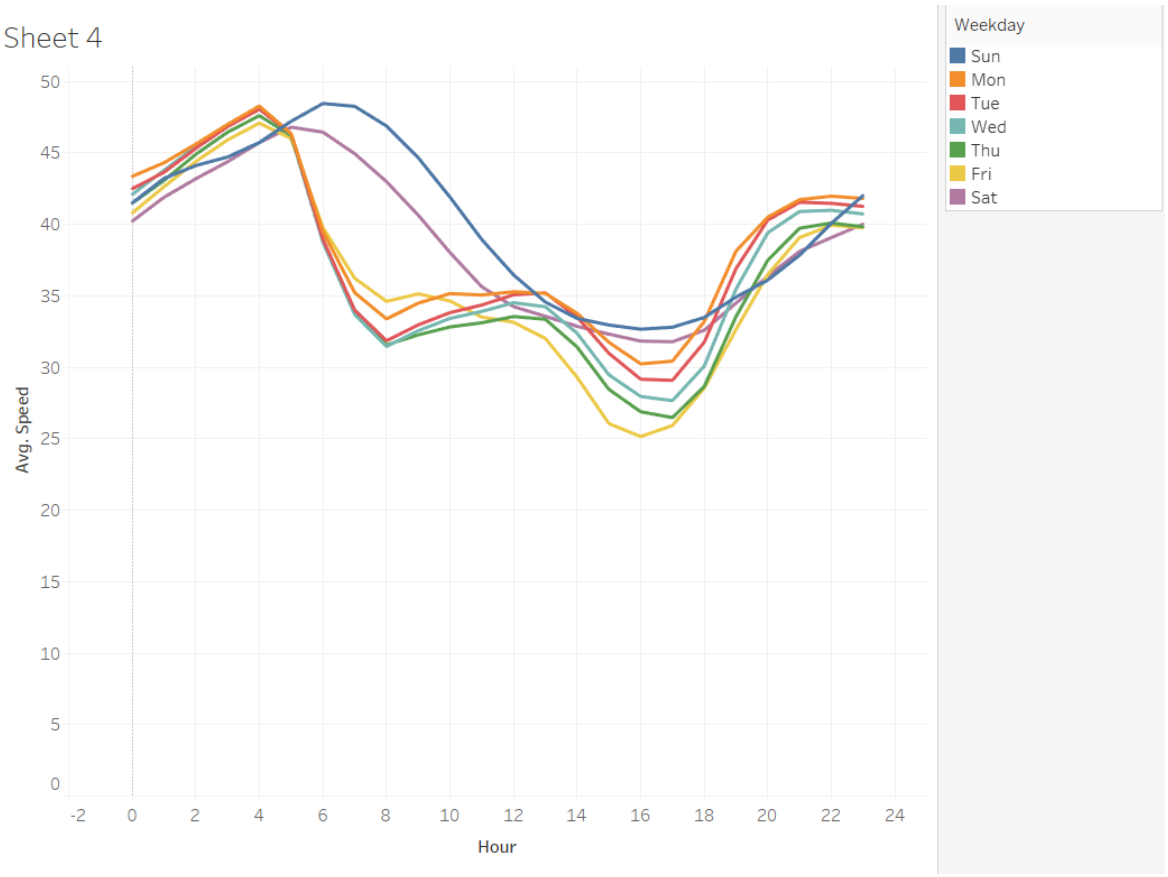


Hình 3.24: Tốc độ trung bình theo từng quận



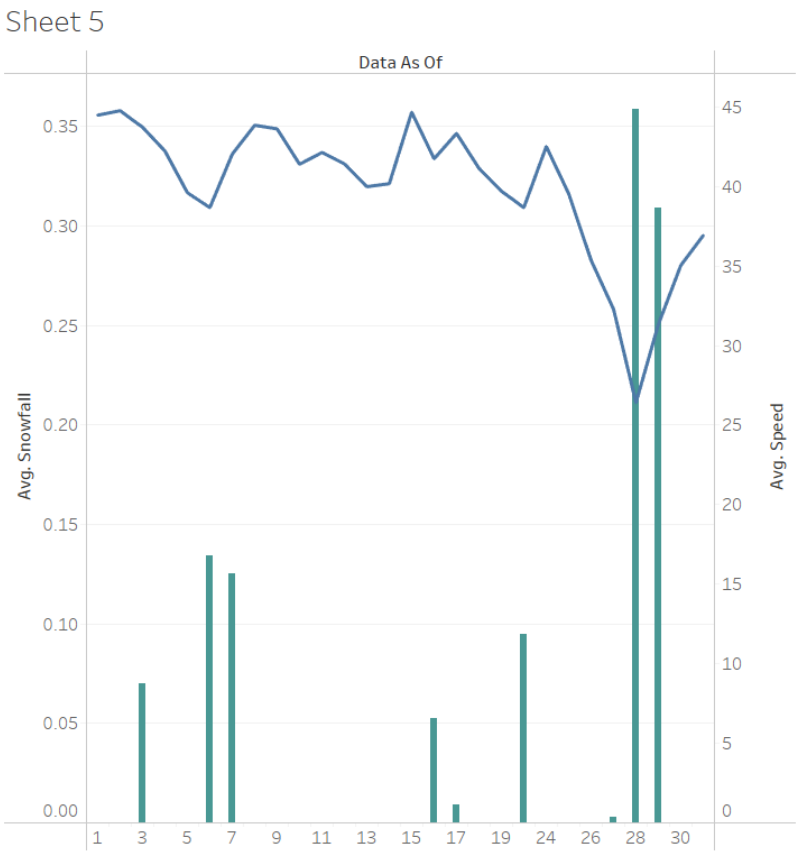
Hình 3.25: Tốc độ trung bình theo ngày trong năm

Tốc độ trung bình quanh năm không biến đổi quá nhiều dao động trong khoảng 34 - 44 km/h.



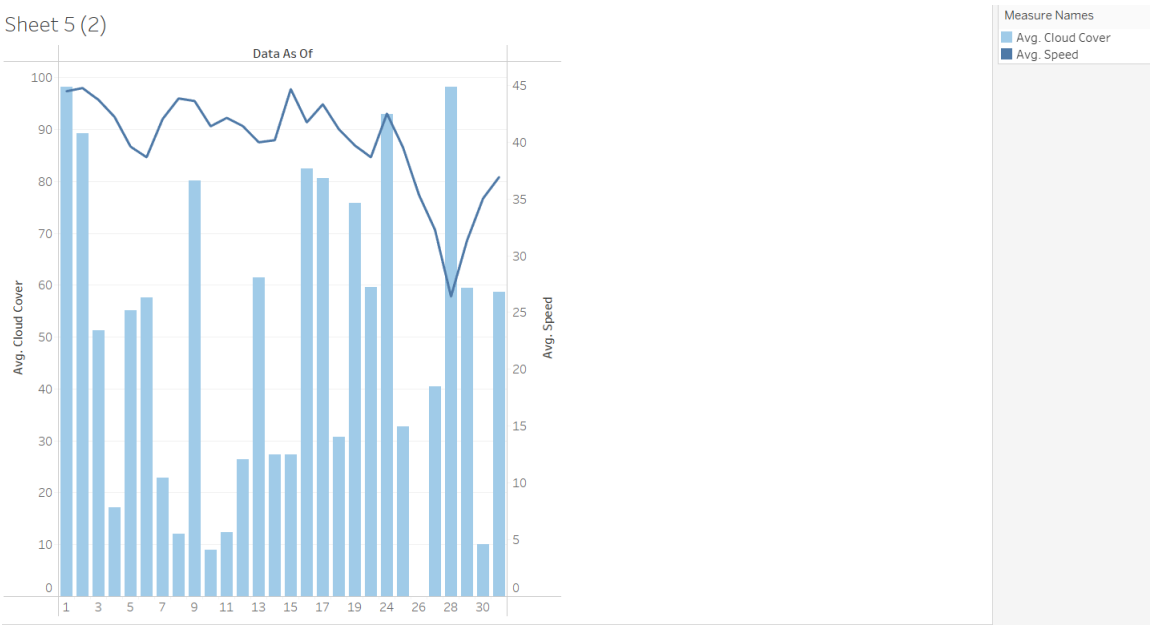
Hình 3.26: Tốc độ trung bình theo ngày trong tuần và theo giờ

Tất cả các ngày trong tuần ghi nhận tốc độ trung bình giảm mạnh vào khoảng 16 đến 18 giờ do thời gian này là thời gian tan làm nên đường đông hơn. Đối với các ngày làm việc (từ thứ 2 đến thứ 6) có sự giảm tốc độ trung bình mạnh hơn vào khoảng 6 đến 8 giờ sáng so với những ngày thứ 7 chủ nhật và cuối tuần.

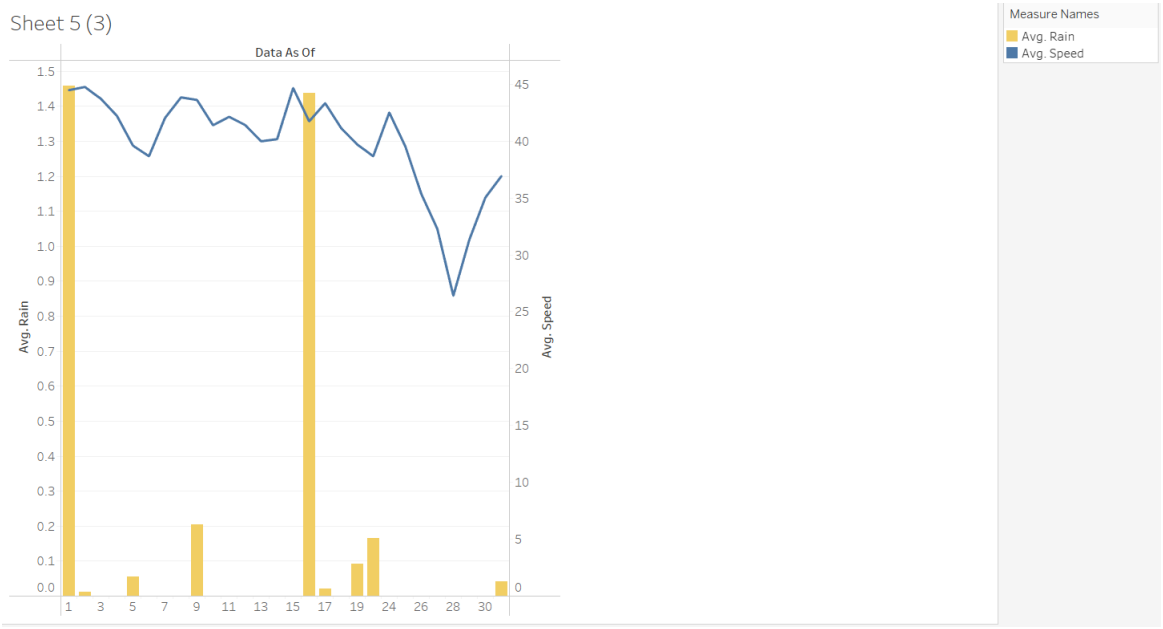


Hình 3.27: Tương quan giữa tốc độ trung bình và độ dày tuyết phủ vào tháng 1 năm 2022

Có thể thấy đối với lượng tuyết không đáng kể (0.15 cm) thì lượng tuyết rơi không ảnh hưởng đáng kể đến xu thế của tốc độ. Tuy nhiên, khi lượng tuyết rơi lớn bất thường vào ngày 28 (0.35 cm), có ghi nhận sự giảm tốc độ trung binh đáng kể. Bên cạnh đó tuyết chỉ rơi vào vài tháng mùa đông và có xu hướng khá đều, ít ngày nào tuyết rơi quá dày. Vì vậy sử dụng giá trị độ dày tuyết đo được để đánh giá tốc độ trung bình là không cần thiết.



Hình 3.28: Tương quan giữa tốc độ trung bình và độ che phủ sương mù



Hình 3.29: Tương quan giữa tốc độ trung bình và lượng mưa

Dễ thấy không có sự tương quan giữa tốc độ trung bình với lượng mưa và lượng sương mù do hầu hết mọi người sử dụng phương tiện di chuyển là ô tô. Bên cạnh đó lượng mưa và sương mù không đạt đến mức ảnh hưởng cho các phương tiện di chuyển.

Vì vậy, các yếu tố thời tiết gần như không ảnh hưởng nhiều đến việc đi lại

của người dân thành phố New York.

3.8 Đánh giá và thử nghiệm

3.8.1 Thiết lập môi trường

Đồ án được thử nghiệm trên môi trường ảo hoá của Cloudera phiên bản CDH 6, bao gồm 5 node (máy chủ).

Cấu hình các node

Các WorkerNode của Spark được cài trên chính các DataNode của HDFS và MasterNode của Spark trên chính NameNode của HDFS. Dưới đây là cấu hình của máy NameNode:

STT	Thông số	Giá trị	Chú thích
1	Name	node1.nms.net	Tên node
2	IP	10.11.10.101	Địa chỉ IP node
3	Cores	8	Số core CPU tối đa mà node được sử dụng
4	Distribution	centos.7.9.2009	Phiên bản hệ điều hành của node
5	Disk	1 TB	Dung lượng ổ đĩa
6	Memory	32 GB	Dung lượng bộ nhớ

Bảng 3.2: Cấu hình NameNode

4 máy được sử dụng làm DataNode (hay WorkerNode), cấu hình của 4 máy này giống nhau, chỉ khác tên và địa chỉ IP.

STT	Thông số	Giá trị	Chú thích
1	Name	node3.nms.net	Tên node
2	IP	10.11.10.103	Địa chỉ IP node
3	Cores	12	Số core CPU tối đa mà node được sử dụng
4	Distribution	centos.7.9.2009	Phiên bản hệ điều hành của node
5	Disk	1 TB	Dung lượng ổ đĩa
6	Memory	32 GB	Dung lượng bộ nhớ

Bảng 3.3: Cấu hình một DataNode

3.8.2 Đánh giá kết quả

Với dữ liệu đầu vào lấy từ DataWarehouse của bộ dữ liệu DOT_Traffic_Speeds_NBE, gồm 2.5 triệu bản ghi (từ đầu năm 2021 đến hết năm 2023), định dạng file parquet, bảng 3.4 đánh giá số lượng executors được sử dụng với thời gian xử lý qua các lần đo mô hình dự báo tắc nghẽn.

Số lượng executors	**Lần đo 1**	**Lần đo 2**	**Lần đo 3**	**Trung bình**
2	1 phút 59 giây	1 phút 54 giây	1 phút 42 giây	1 phút 52 giây
3	1 phút 41 giây	1 phút 17 giây	1 phút 24 giây	1 phút 27 giây
4	1 phút 20 giây	1 phút 29 giây	1 phút 11 giây	1 phút 20 giây
5	55 giây	1 phút 4 giây	1 phút 8 giây	1 phút 2 giây
6	1 phút 8 giây	1 phút 4 giây	1 phút 5 giây	1 phút 6 giây
7	54 giây	49 giây	1 phút 6 giây	56 giây
8	1 phút 4 giây	1 phút 2 giây	1 phút 10 giây	1 phút 5 giây

Bảng 3.4: Thời gian xử lý khi chạy mô hình dự báo tắc nghẽn

Thực hiện tổng hợp tính toán tốc độ trung bình năm 2022 của thành phố NewYork như bảng 3.5.

Thông số	**Mô tả**
Dữ liệu đầu vào	Dữ liệu đo tốc độ di chuyển của phương tiện giao thông năm 2022 được thu thập dưới định dạng csv, dung lượng 3.6GB
Dữ liệu đầu ra	Tốc độ trung bình của năm 2022
Cách tính toán	df.select(mean(col('SPEED')).alias('mean_speed')).collect()

Thu được được kết quả như trong bảng 3.5.

Số lượng executors	**Lần đo 1**	**Lần đo 2**	**Lần đo 3**	**Trung bình**
1	1 phút 50 giây	1 phút 37 giây	1 phút 48 giây	1 phút 45 giây
2	1 phút 7 giây	1 phút 20 giây	1 phút 16 giây	1 phút 14 giây
3	1 phút 10 giây	1 phút 7 giây	57 giây	1 phút 15 giây
4	55 giây	52 giây	49 giây	52 giây
5	1 phút 4 giây	54 giây	55 giây	58 giây
6	1 phút 9 giây	59 giây	1 phút 1 giây	1 phút 3 giây

Bảng 3.5: Thời gian xử lý khi chạy tổng hợp tính toán theo năm

Nhận xét:

Bảng 3.4 và 3.5 đều cho thấy có sự giảm rõ rệt về thời gian trung bình thực thi khi tăng số lượng executor (bảng 3.4 từ 2 lên 7 executors và bảng 3.5 từ 1 lên 4 executors). Điều đó chứng tỏ, việc xử lý dữ liệu trên hệ thống phân tán giảm thời gian xử lý. Tuy nhiên khi số lượng executor tăng lên đên một ngưỡng nhất định (lên 8 executors đối với bảng 3.4 và lên 5 executors đối với bảng 3.5) thì thời gian bắt đầu tăng trở lại có thể vì dữ liệu chưa đủ lớn khiến việc chia nhỏ dữ liệu cho nhiều executor làm tăng thời gian giao tiếp giữa chúng.

Kết luận

Kết quả của Đồ án

Đồ án đã bước đầu xây dựng được một hệ thống phân tích dữ liệu lớn: từ bước thu thập, phân tích đến trực quan hóa.

- Trình bày các kiến thức cơ bản về dữ liệu lớn.
- Xây dựng được kiến trúc một hệ thống đơn giản.
- Ứng dụng được các công cụ phân tích dữ liệu lớn vào giải quyết bài toán.

Hướng phát triển

Trong tương lai, nếu có thể sử dụng nguồn dữ liệu trực tiếp từ logs của các máy đo, đồ án có thể phát triển tiếp để:

- Xử lý dữ liệu thời gian thực. Từ đó có thể xây dựng hệ thống cảnh bảo nếu phát hiện quá tốc độ hoặc thể hiện các đoạn đường đang bị ùn tắc.
- Xây dựng các luồng chống chịu lỗi, trong trường hợp dữ liệu không lấy được về.
- Kết hợp với bộ dữ liệu về các vụ tai nạn, số lượng phương tiện di chuyển để đánh giá được hoàn thành hơn.

Tài liệu tham khảo

[1] Apache Software Foundation,"Spark Documentation",
https://spark.apache.org/docs/3.4.1/index.html.

[2] Apache Software Foundation, "Apache NiFi Overview",
https://nifi.apache.org/docs.html

[3] Tom White, *Hadoop: The Definitive Guide*, O'Reilly Media Inc., 2009.

[4] Bill Chambers, Matei Zaharia, *Spark: The Definitive Guide*, O'Reilly Media Inc., 2018.

[5] Lior Rokach, Oded Maimon, *The Data Mining and Knowledge Discovery Handbook*, 2005, 165-192.