BỘ GIÁO DỤC & ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN CHUYÊN NGÀNH

ĐỀ TÀI: TÌM HIỂU KIẾN TRÚC DELTA ARCHITECTURE VÀ XÂY DỰNG ỨNG DỤNG DEMO

GVHD: ThS. Quách Đình Hoàng

SVTH :

- **Trần Phát Đạt 19133018**
- **Trần Công Tuấn Mạnh 19133035**

TP.Hồ Chí Minh, ngày 27 tháng 11 năm 2022

ĐH SƯ PHẠM KỸ THUẬT TP.HCM **CỘNG HÒA XHCN VIỆT NAM**

KHOA CÔNG NGHỆ THÔNG TIN **Độc lập – Tự do – Hạnh phúc**

****** ******

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Họ và tên Sinh viên 1: **Trần Phát Đạt** MSSV 1: **19133018**

Họ và tên Sinh viên 2: **Trần Công Tuấn Mạnh** MSSV 2: **19133035**

Chuyên ngành: **Kỹ thuật dữ liệu**

Tên đề tài: **Tìm hiểu kiến trúc Delta và xây dựng ứng dụng demo.**

Họ và tên giảng viên hướng dẫn: **ThS. Quách Đình Hoàng**

NHẬN XÉT

1. Về nội dung đề tài và khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đề nghị cho bảo vệ hay không?
5. Đánh giá loại:
6. Điểm:

TP. Hồ Chí Minh, ngày tháng năm 2022

Giảng viên hướng dẫn

(Ký & ghi rõ họ tên)

ĐH SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
******

CỘNG HÒA XHCN VIỆT NAM
Độc lập – Tự do – Hạnh phúc
******

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

Họ và tên Sinh viên 1: **Trần Phát Đạt** MSSV 1: **19133018**

Họ và tên Sinh viên 2: **Trần Công Tuấn Mạnh** MSSV 2: **19133035**

Chuyên ngành: **Kỹ thuật dữ liệu**

Tên đề tài: **Tìm hiểu kiến trúc Delta và xây dựng ứng dụng demo.**

Họ và tên giảng viên hướng dẫn: **ThS. Quách Đình Hoàng**

NHẬN XÉT

1. Về nội dung đề tài và khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đề nghị cho bảo vệ hay không?
5. Đánh giá loại:
6. Điểm:

TP. Hồ Chí Minh, ngày tháng năm 2022

Giảng viên phản biện

(Ký & ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến thầy Quách Đình Hoàng, người đã giúp chúng em biết đến đề tài này và chỉ ra cho chúng em những đường đi tốt nhất để hoàn thành đồ án. Nhờ thầy đưa ra những lời khuyên từ kinh nghiệm thực tiễn của mình để định hướng cho chúng em đi đúng hướng với đề tài đã chọn, thầy luôn tận tình giải đáp các thắc mắc một cách chi tiết trong suốt quá trình học để chúng em có thể có thêm kiến thức để thực hiện đề tài.

Tiếp đó, nhóm chúng em xin được gửi lời cảm ơn chân thành đến khoa Công nghệ thông tin của trường Đại học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh đã cung cấp cho chúng em những kiến thức nền tảng, môi trường học tập và phát triển để có thể thực hiện tiểu luận chuyên ngành Kỹ thuật Dữ liệu trong kỳ học này. Vì Tiểu luận chuyên ngành của chúng em thực hiện trong thời gian không quá dài nên sẽ không thể tránh khỏi những sai sót và những mặt chưa hoàn thiện về mặt kỹ thuật cũng như là cách trình bày, chúng em mong quý thầy, cô sẽ thông cảm bỏ qua những sai sót cho nhóm chúng em.

Cuối cùng, chúng em xin chúc quý thầy, cô có thật nhiều sức khỏe, thành công hơn trên con đường sự nghiệp của mình. Chúng em xin chân thành cảm ơn.

DANH MỤC CÁC TỪ VIẾT TẮT

CNTT	Công nghệ thông tin
ML	Machine Learning
DE	Data Engineering
ALS	Alternating Least Square
TLCN	Tiểu luận chuyên ngành
RDD	Resilient Distributed Dataset
CRUD	insert (thêm), read (đọc), update (sửa), delete (xóa)
DML	Các thao tác dữ liệu

DANH MỤC BẢNG BIỂU VÀ HÌNH ẢNH

Chương 1 - Giới thiệu

1.1 Bối cảnh và động lực

Ngày nay, các công ty thu thập được rất nhiều dữ liệu người dùng. Dữ liệu đã dần trở thành một tài sản quan trọng của các công ty này. Làm sao để tổ chức, quản lý và tận dụng nguồn dữ liệu này để hiểu người dùng, tăng lợi nhuận, và tạo được lợi thế cạnh tranh là mối quan tâm hàng đầu của các công ty. Điều này làm cho nhu cầu và công việc liên quan đến dữ liệu ở Việt Nam nói riêng và thế giới nói chung đang rất phát triển. Delta architecture là một trong những kiến trúc rất mới để tổ chức và xử lý dữ liệu lớn. Kiến trúc này chỉ phổ biến trong khoảng 2 năm trở lại đây. Databricks là một trong những công ty đi đầu trong việc đề xuất và hoàn thiện kiến trúc này. Công ty này là một startup được thành lập từ nhóm nghiên cứu hàng đầu thế giới về big data, và distributed systems, và database của GS. Ion Stoica tại đại học Berkeley. Matei Zaharia, người đặt nền móng cho Apache Spark từ luận văn tiến sỹ của mình dưới sự hướng dẫn của GS Ion Stoica, hiện đang là giám đốc công nghệ (CTO) của công ty này. Hai bài báo quan trọng đề cập đến kiến trúc này là Delta Lake [1] (năm 2020) và Lakehouse [2] (năm 2021). Nhóm cho rằng trong tương lai kiến trúc này sẽ rất phát triển nên đã chọn đề tài này để bắt kịp xu hướng của thời đại.

1.2 Mục tiêu và phạm vi

Mục tiêu của nhóm là tìm hiểu và kiến trúc Delta, về những thành phần trong kiến trúc này, từ đó có thể hiểu được chi tiết về hệ thống đường ống dữ liệu này. Qua đó, nhóm sẽ ứng dụng những kiến thức đó để xây dựng hệ thống dữ liệu trực tuyến.

Phạm vi của hệ thống này sẽ ứng dụng kiến thức, kỹ năng trong lúc tìm hiểu, nghiên cứu về kiến trúc Delta, để xây dựng hệ thống phát trực tuyến dữ liệu từ Kafka, lữu trữ chúng trong Delta Lake, và sử dụng cho các mục đích báo cáo thời gian thực, xây dựng mô hình gợi ý nhà hàng. Về quy mô, do giới hạn về thời gian và tài nguyên, nhóm dự định chỉ phát triển ứng dụng ở mức căn bản. Mặt khác, việc phát triển một project về data engineering là rất khó trong môi trường trường học.

1.3 Bố cục của báo cáo

Bố cục của bài tiểu luận gồm các phần chính sau:

- Các kiến trúc nền tảng
- Lý thuyết kiến trúc delta
- Xây dựng ứng dụng kiến trúc Delta:
    + Xây dựng kiến trúc delta
    + Ứng dụng kiến trúc delta

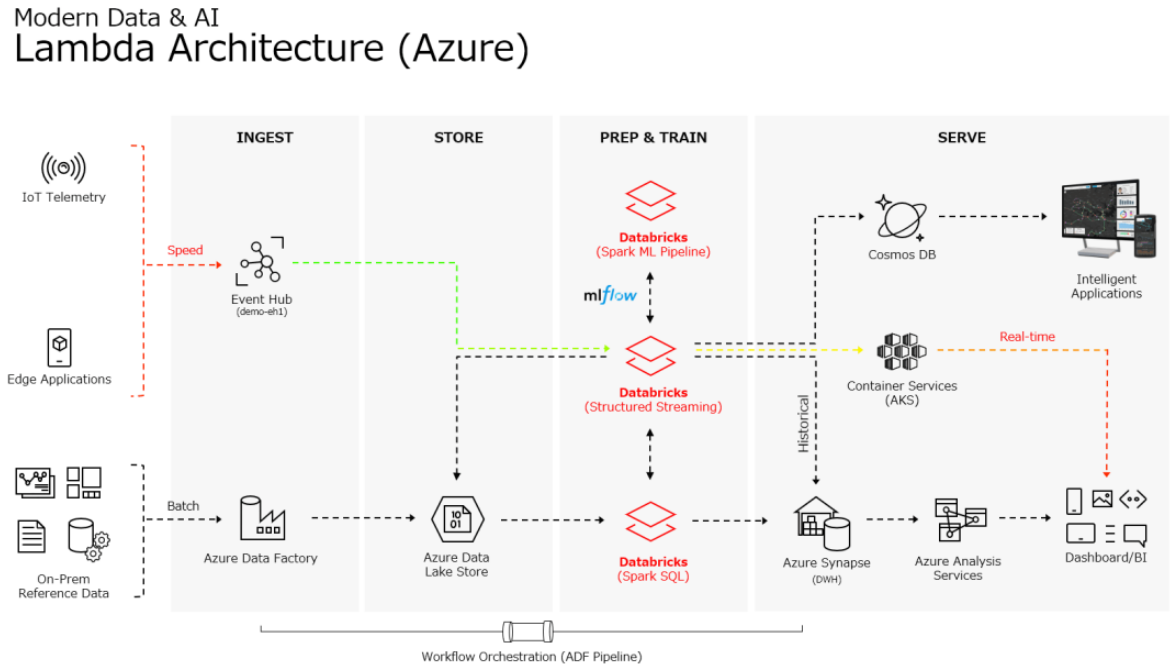
Chương 2 - Kiến thức nền tảng

2.1 - Lịch sử và bối cảnh bài toán

2.1.1 - Các kiến trúc phổ biến

1. Lambda Architecture

Lambda Architecture là một kiến trúc dữ liệu lớn mạnh mẽ và nổi tiếng nhất, nó có khả năng xử lý batch và real-time [3]. Dữ liệu on-premise được đưa vào pipeline theo batch, dữ liệu real-time từ các thiết bị IoT hay ứng dụng đi qua pipeline bằng cách streaming. Hai luồng xử lý này được phát triển trong hai pipelines riêng biệt nhau, dữ liệu đi qua pipeline được đưa vào quy trình tiền xử lý dữ liệu và training sử dụng *Databricks* và *Mlflow* sau đó được sử dụng vào các mục đích đầu cuối khác.



Hình 2.1.1: Kiến trúc Lambda [3]

Hình 2.1.1 biểu diễn một kiến trúc Lambda đầy đủ các thành phần quan trọng.

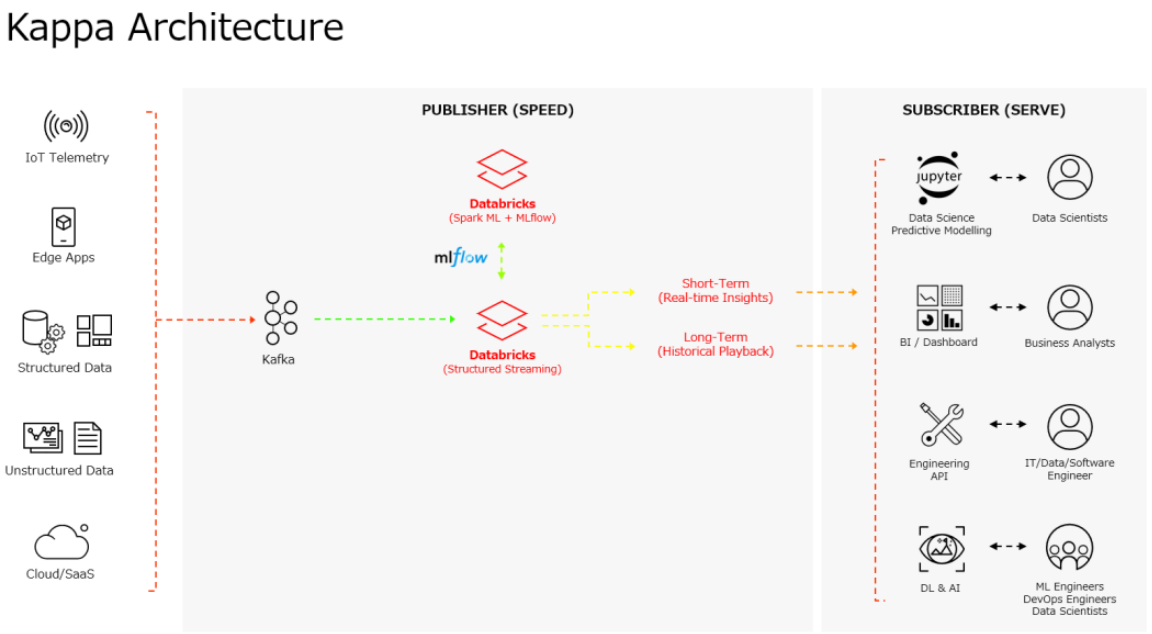
Một trong những thử thách của Lambda Architecture nằm ở chính sự phức tạp của nó.

Việc triển khai hai pipelines tách rời khiến cho việc maintain chúng một cách đồng bộ với nhau trở nên khó khăn hơn bao giờ hết. Tồi tệ hơn nữa là luồng dữ liệu và các logic xử lý trong bối cảnh dữ liệu lớn hiện nay đang có tốc độ gia tăng khi mà dữ liệu được sinh ra vô cùng lớn.

Vậy nên, để giải quyết vấn đề đó, chúng ta cần một giải pháp tốt hơn tối ưu pipelines trong kiến trúc của nó. Điều này dẫn đến sự ra đời của Kappa Architecture.

2. *Kappa Architecture*

Trong kiến trúc của Kappa, việc xây dựng một đường ống dữ liệu riêng cho xử lý luồng (batch) là không cần thiết bởi vì tất cả dữ liệu được xử lý bởi một đường ống dữ liệu phát trực tuyến (streaming pipeline) duy nhất [3].



Hình 2.1.2: Kiến trúc Kappa [3]

Hình 2.1.2 biểu diễn một kiến trúc Kappa đầy đủ các thành phần cần thiết.

Việc đưa tất cả dữ liệu vào chung một đường ống (pipeline) như vậy cần một khả năng xử lý và mở rộng mạnh mẽ để có thể đảm bảo được đường ống đó (pipeline) không bị hỏng và dữ liệu phải đồng bộ trong quá trình vận hành. Thay vì xử lý dữ liệu hai lần như Lambda Architecture, Kappa xử lý dữ liệu luồng một lần duy nhất, điều này khiến nó tương thích cao với Spark khi kết hợp chúng lại để xử lý dữ liệu trong thời gian thực.

Bất kể là loại dữ liệu hay nguồn nào, tất cả đều được giữ trong một luồng và subscribers (người dùng downstream) sẽ phát lại các luồng dữ liệu đã được tính toán trước trong khoảng thời gian mong muốn dựa trên các trường hợp sử dụng.

Do thiếu hồ dữ liệu (data lake) để tổng hợp lại dữ liệu vào một nơi nhất định, hệ thống phát trực tuyến (streaming) có thể cần phải duy trình hàng đợi (queue) hoạt động trong khoảng thời gian tồn tại của pipelines để phục vụ cho trường hợp xem lại một lượng lớn dữ liệu trong quá khứ.

2.2 - Các giải pháp hiện tại và vấn đề

2.2.1 Delta Architecture

Quay trở lại với Hadoop, HDFS-based (Hadoop File System) data lake được thiết kế với tính bất biến, tức là không thể sửa đổi cấu trúc dữ liệu của hồ dữ liệu.

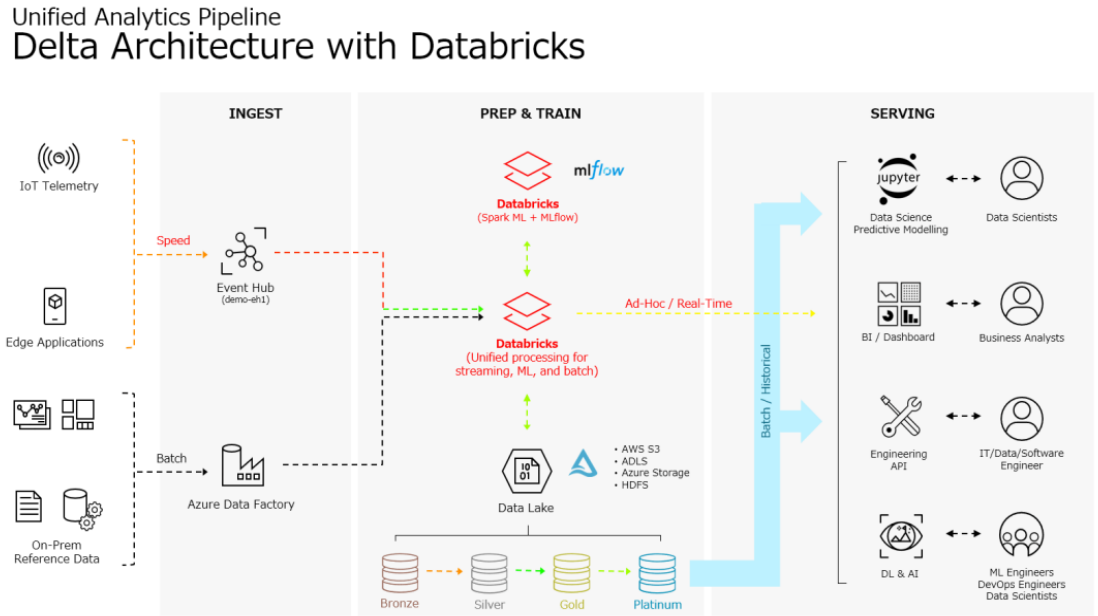
Điều này gây ra việc không tối ưu về chi phí vì cấu trúc dữ liệu hiện có thể thay đổi bởi các phép biến đổi mới xuất hiện, do đó dẫn tới việc tạo lại toàn bộ cấu trúc dữ liệu.

Trong kiến trúc Delta [3], data lake không còn được coi là bất biến. Ngược lại, dữ liệu được xử lý dưới dạng bản ghi "*delta*" chứ không phải là bản ghi nối tiếp nhau.

Các phép biến đổi hàng loạt (batch) có thể thực hiện các thao tác với dữ liệu chẳng hạn như thêm, đọc, xóa, sửa trên các cấu trúc dữ liệu hiện có trong data lake bằng cách sử dụng công nghệ được gọi là *Delta Lake* [4].

Trên thực tế, Delta Lake mang đến các khả năng truy vấn, dữ liệu minh bạch như Data Warehouse, lưu trữ các định dạng dữ liệu như Data Lake, giúp nó hoạt động hiệu quả hơn và đáng tin cậy hơn trong các đường ống xử lý dữ liệu lớn.

Do đó, nó hợp nhất hai lớp có hiệu quả cho một quy trình xử lý liền mạch nhau (cùng một API, có nghĩa là một bảng dạng "delta" có thể vừa dùng để ghi dữ liệu trực tuyến và cũng thể dùng để ghi dữ liệu tĩnh, cho batch và streaming) với chi phí thấp hơn (tức là có tối ưu hóa hiệu suất). Lợi ích là rất lớn vì khả năng này có nghĩa là các tổ chức không còn phải xử lý dữ liệu theo cách khác nhau dựa trên tốc độ đọc và phương pháp xử lý dữ liệu, thay vì phải sử dụng Data Lake để lưu trữ dữ liệu thô, sau đó sử dụng Data Warehouse để hệ thống lại cấu trúc dữ liệu. Giờ đây, tất cả dữ liệu ở ngay trong một nền tảng, cung cấp khả năng quản lý nhất quán, tối ưu hóa chi phí khi triển khai hệ thống dữ liệu.



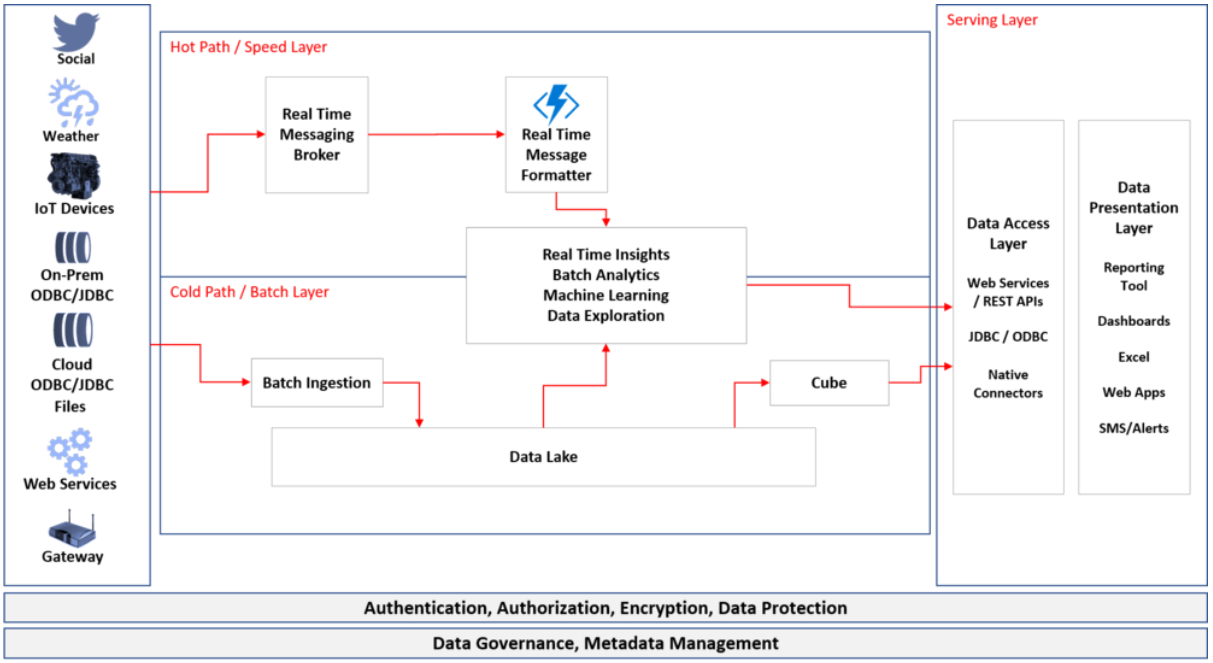
Hình 2.1.3: Kiến trúc Delta [3]

Hình 2.1.3 biểu diễn một kiến trúc Delta đơn giản. Delta theo nghĩa đen cũng được sử dụng để biểu thị sự thay đổi gia tăng (như trong Delta trong Toán học).

Sự thay đổi gia tăng trong quá trình xử lý dữ liệu đến từ bất kỳ dữ liệu mới nào được tạo, cập nhật hay phát trực tuyến kể từ lần xử lý cuối cùng. Cách chèn và cập nhật dữ liệu có thể được hợp nhất với dữ liệu hiện có trong lớp dữ liệu và các tệp hệ thống tệp có thể được cập nhật. Hệ thống tệp hiện hỗ trợ hoạt động CRUD (Tạo / Đọc / Cập nhật / Xóa) với công nghệ có sẵn và hệ thống tệp có thể được thực hiện tuân thủ ACID.

Hiện tại, tại thời điểm báo cáo này được viết, Delta Lake đã cung cấp mã nguồn mở, điều này có nghĩa là chúng ta có thể cài đặt, thiết lập kiến trúc Delta này ngay dưới máy của mình mà không cần phải sử dụng các nền tảng của các nhà cung cấp khác như Databricks. Như đã mô tả ban đầu, Databricks là một nhà cung cấp hệ thống, môi trường đám mây cho phép xử lý, quản lý dữ liệu lớn một cách mạnh mẽ dựa trên công nghệ xử lý phân tán và Spark, Delta Lake. Về mặt kiến trúc kỹ thuật, đó là một bảng dữ liệu có định dạng Delta, bản chất không có định dạng Delta ở đây, định dạng thật sự của nó là tập Parquet (định dạng tệp thường được sử dụng trong Hadoop) với độ tin cậy và tối ưu hóa được tích hợp sẵn, mang lại hiệu suất nhanh hơn 10 ~ 100 lần so với Spark trên Parquet. Cùng với các dữ liệu log ghi lại thông tin lịch sử, lược đồ của bảng Delta.

Dữ liệu được lưu trữ trong Databricks Delta có thể được truy cập (đọc / ghi) bằng cách sử dụng cùng một API (giao diện lập trình ứng dụng) Apache Spark SQL thống nhất cả quy trình hàng loạt và luồng.



Hình 2.2.2: Một Pipeline của Delta Architecture [3]

Hình 2.2.2 là một ví dụ về một pipeline sử dụng Delta Architecture, ngoài việc những việc như đã nói ở trên. Việc đảm bảo xác thực, ủy quyền, mã hóa, bảo vệ, quản

trị dữ liệu và quản lý siêu dữ liệu là những việc vô cùng quan trọng trong việc phát triển Delta Architecture.

2.3 - Triển khai kiến trúc Delta

Spark Engine xử lý mỗi batch như một stream (luồng) và mỗi stream như một đầu ra liên tục của các lô (batch).

Batch as a stream, xem một bảng tĩnh như một batch. Với tính năng phát trực tuyến của Spark, chúng ta có thể xem một bảng như một nguồn phát trực tuyến (streaming source), nơi mà dữ liệu liên tục được tạo và xử lý một cách tuần tự.

Ví dụ cụ thể, Giả sử ta muốn đọc dữ liệu về taxi, nó sẽ được lưu trữ như kiến trúc delta. Lúc này, kiến trúc này sẽ cho phép truyền dữ liệu dạng bảng, thực hiện các thao tác tính toán tổng hợp (đếm số xe, tổng số xe …) trong các micro batches. Lúc này, trên bảng biểu diễn lượng dữ liệu đầu vào, ta sẽ thấy số lượng đếm được tiếp tục tăng lên theo stream từ nguồn.

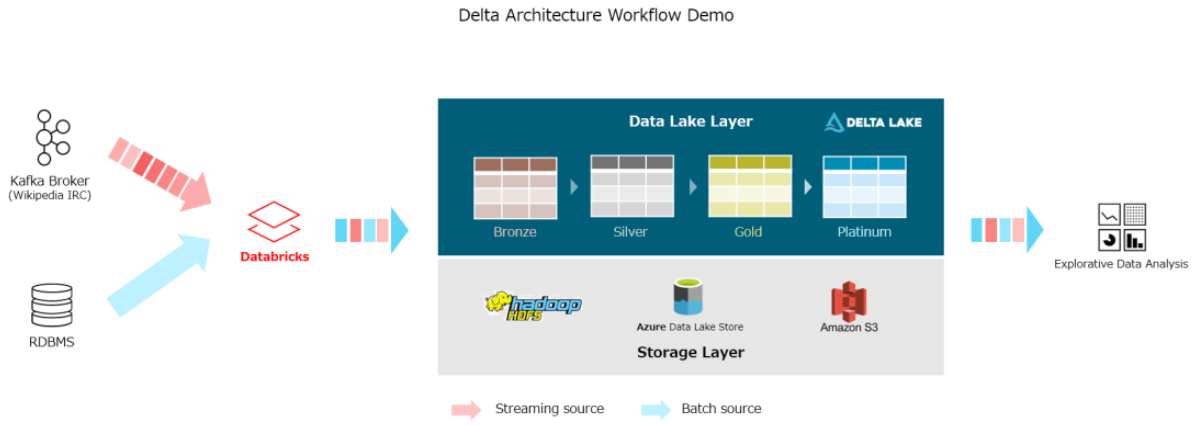
Stream as a batch, sử dụng cùng một phương thức (chính xác hơn, cùng một APIs) để làm việc với dữ liệu luồng và xem nó như một loạt các batch jobs nhỏ.

Bởi vì streaming vốn dĩ luôn liên tục và không bao giờ kết thúc, nên nó cũng có những cân nhắc và kỹ thuật cụ thể khác (windowing, temporal aggregations) để xử lý loại dữ liệu này. Tuy nhiên, một API hợp nhất cho cả batch và streaming, xử lý chúng về cơ bản trở nên giống nhau.

2.3.1 Mô hình Delta Architecture

Với sự ra đời của Delta Lake, nó mang đến các giao dịch tuân thủ ACID (Atomicity - nguyên tử, Consistency - nhất quán, Isolation - cô lập, và Durability - lâu bền), quản lý chất lượng dữ liệu, hỗ trợ đầy đủ tương tác dữ liệu hay các thao tác phục hồi, cũng như tối ưu hóa hiệu suất quan trọng cho data lakes. Đó là một bước tiến lớn về mặt bổ sung nhiều thiếu sót cho các hồ dữ liệu hiện có.

Do đó, có thể kết hợp batch với streaming vào trong một bộ xử lý, và nhiều công việc thực hiện đọc / ghi đồng thời dựa trên cùng một tập dataset trong data lake. Điều này trở thành cốt lõi chính của kiến trúc Delta. Ví dụ minh họa như hình 2.3.1.1:



Hình 2.3.1.1: Bốn tầng của 1 Delta Lake [3]

Qua hình 2.3.1.1, có thể nhận thấy rằng, luồng làm việc giống như một quy trình ETL cổ điển. Mặc dù không có định nghĩa cứng nhắc vào điều này, nhưng kiến trúc Delta thường vận hành chuyển đổi dữ liệu theo các cách sau:

a) Nhập dữ liệu thô (Ingest raw data - Bronze)
   Đổ các loại dữ liệu thô khác nhau vào cùng một khu vực được chuẩn bị trước.
b) Xử lý dữ liệu (Process data - Silver)
   Làm sạch và cấu trúc lại dữ liệu thô.
c) Tính toán tổng hợp dữ liệu (Aggregate data - Gold)
   Thực hiện các hàm tổng hợp và summation để sản xuất các tập dữ liệu hướng business có sẵn cho các trường hợp sử dụng báo cáo và phân tích.
d) Lưu dữ liệu hoàn chỉnh (Present data - Platinum)
   Lưu trữ dữ liệu hoàn chỉnh và trực quan hóa và phân tích dữ liệu để có được những hiểu biết sâu sắc về business.

Trong suốt quá trình này, dữ liệu được lặp đi lặp lại trên cùng một tập dữ liệu, khi đó dữ liệu sẽ có giá trị hơn. Lý do có thể làm điều này chính là lý do tại sao Delta Lake cung cấp các giao dịch ACID, xử lý streaming, batch trên cũng một bảng và hỗ trợ đầy đủ các thao tác trên dữ liệu cho các hồ dữ liệu truyền thống. Để hồ dữ liệu có thể cung cấp các khả năng giống như kho dữ liệu trên các tập dữ liệu lớn. Delta Lake là một bước quan trọng để đạt được khái niệm được gọi là Lakehouse [2].

Hình 2.3.1.2:: Khởi nguồn của một Delta Architecture

Hình 2.3.1.2 mô tả một kiến trúc delta, nó thực chất là sự kết hợp giữa delta lake và spark streaming.

Nó dựa vào data lake và hoạt động dựa trên cấu trúc streaming (tính mở rộng và tính chịu lỗi). Nó mở rộng việc dùng big data cho xử lý spark với các transaction ACID ở data lake.

Nguyên nhân ra đời: Với sự phát triển của công nghệ ngày một nhanh như ngày nay, Data Warehouse đã gặp những thiếu sót trong việc hỗ trợ khả năng cung cấp dữ liệu trong thời gian thực. Cùng với đó, Data Lake cũng gặp vấn đề về quá trình thao tác dữ liệu bởi nó phải làm việc trên các tệp, dẫn tới hiệu suất truy vấn dữ liệu không tốt. Vậy nên, Delta Lake là sự kết hợp giữa khả năng cung cấp ACID, thao tác dữ liệu và khả năng lưu trữ các loại dữ liệu khác nhau, để khắc phục những khuyết điểm của Data Lake và Data Warehouse.

Ưu nhược điểm của Delta Architecture:

+ **Ưu điểm:**
    - Chi phí thấp: Bằng cách cung cấp một tầng duy nhất cho tất cả các loại dữ liệu, Delta Lake cung cấp giải pháp lưu trữ tiết kiệm chi phí hơn vì các nhóm chỉ phải quản lý một nguồn dữ liệu duy nhất.
    - Giảm trùng lặp dữ liệu: Delta Lake cung cấp một nền tảng lưu trữ dữ liệu đa mục đích duy nhất có thể đáp ứng mọi nhu cầu kinh doanh, giảm trùng lặp dữ liệu.
    - Sự cởi mở: Delta Lake cho phép truy cập dữ liệu bằng bất kỳ công cụ nào, thay vì bị giới hạn ở các ứng dụng chỉ có thể xử lý dữ liệu có cấu trúc như SQL.
    - Hỗ trợ các công cụ BI và ML: Delta Lake cho phép phân tích nâng cao bằng cách tích hợp với các công cụ BI phổ biến nhất như Tableau và PowerBI. Delta Lake sử dụng các định dạng dữ liệu mở (ví dụ: Parquet) với API và thư viện máy học, cho phép các nhà khoa học dữ liệu tận dụng dữ liệu
    - Quản trị dữ liệu dễ dàng: Bằng cách thực thi tính toàn vẹn của dữ liệu, kiến trúc kho dữ liệu cho phép triển khai các lược đồ bảo mật dữ liệu tốt hơn so với các kho dữ liệu.

+ **Nhược điểm:**
    - Có thể giảm chức năng: Nó có thể là một thách thức để thiết kế và duy trì thiết kế nguyên khối của ngôi nhà bên hồ. Bên cạnh đó, các thiết kế phổ quát có thể có chức năng thấp hơn so với các thiết kế được phát triển cho các trường hợp sử dụng cụ thể.
    - Khái niệm kém phát triển: Delta Lake hay Data lakehouses là một công nghệ tương đối mới và cần phát triển hơn nữa. Tình trạng công nghệ hiện tại không cho phép phát huy hết khả năng của công nghệ này.

2.4 - Tổng quan các thành phần trong đường ống dữ liệu

2.4.1 - Streaming source

Trong kiến trúc đường ống dữ liệu, điểm xuất phát đầu tiên phải nói đến đó chính là nguồn dữ liệu được sử dụng, các nguồn phát trực tiếp của đường ống dữ liệu có thể bắt đầu từ các cảm biến, API, hoặc các nguồn truyền phát sự kiện (event streaming) như Kafka [4], Spark Streaming.

Về mặt kỹ thuật, truyền sự kiện là phương pháp thu thập dữ liệu trong thời gian thực từ các nguồn sự kiện như cơ sở dữ liệu, cảm biến, thiết bị di động, dịch vụ đám mây và ứng dụng phần mềm dưới dạng luồng sự kiện; lưu trữ lâu dài các luồng sự kiện này để truy xuất sau này; thao tác, xử lý và phản hồi với các luồng sự kiện trong thời gian thực; và định tuyến các luồng sự kiện đến các công nghệ đích khác nhau khi cần. Hình 2.4.1 chính là biểu tượng thường thấy của Apache Kafka.

Hình 2.4.1: Apache Kafka [4]

Kafka kết hợp ba khả năng chính để chúng ta có thể triển khai các trường hợp sử dụng của mình để phát trực tiếp sự kiện từ đầu đến cuối bằng một giải pháp đã được thử nghiệm duy nhất:

1. Xuất bản (viết) và đăng ký (đọc) các luồng sự kiện, bao gồm nhập/xuất liên tục dữ liệu của bạn từ các hệ thống khác.
2. Lưu trữ các luồng sự kiện một cách lâu dài và đáng tin cậy.
3. Xử lý các luồng sự kiện khi chúng xảy ra, cung cấp khả năng truy hồi lại dữ liệu..

Tất cả các chức năng này được cung cấp theo cách phân tán, có khả năng mở rộng cao, linh hoạt, chịu lỗi và an toàn. Kafka có thể được triển khai trên phần cứng, máy ảo và bộ chứa cũng như trên đám mây. Có thể chọn giữa việc tự quản lý môi

trường Kafka và sử dụng các dịch vụ được quản lý hoàn toàn do nhiều nhà cung cấp phát triển [4].

2.4.2 - Kiến trúc Delta Lake

Delta Lake là một dự án mã nguồn mở cho phép xây dựng kiến trúc Lakehouse trên các hồ dữ liệu [5]. Delta Lake cung cấp các giao dịch ACID, xử lý siêu dữ liệu có thể mở rộng và hợp nhất xử lý dữ liệu hàng loạt và truyền trực tuyến trên các hồ dữ liệu hiện có, chẳng hạn như S3 (AWS Simple Storage Service) , GCS (Google Cloud Storage) và HDFS. Hình 2.4.2 chính là biểu tượng thường thấy của một Delta Lake.

Hình 2.4.2: Delta Lake [6]

Cụ thể, Delta Lake cung cấp:

- Giao dịch ACID trên Spark: Các mức cách ly có thể tuần tự hóa đảm bảo rằng người đọc không bao giờ nhìn thấy dữ liệu không nhất quán [6].
- Xử lý siêu dữ liệu có thể mở rộng: Tận dụng sức mạnh xử lý phân tán của Spark để xử lý tất cả siêu dữ liệu cho các bảng quy mô petabyte với hàng tỷ tệp một cách dễ dàng [6].
- Hợp nhất batch và phát trực tuyến: Một bảng trong Delta Lake là một bảng batch cũng như nguồn phát trực tuyến và tiêu thụ. Nhập dữ liệu trực tuyến, chèn lấp lịch sử hàng loạt, truy vấn tương tác, tất cả đều hoạt động ngay lập tức [6].
- Thực thi giản đồ: Tự động xử lý các biến thể giản đồ để ngăn việc chèn các bản ghi xấu trong quá trình nhập [6].
- Time travel: Lập phiên bản dữ liệu cho phép lùi thời gian, theo dõi quá trình kiểm tra lịch sử đầy đủ và thử nghiệm máy học có thể lặp lại [6].
- Upserts và deletes: Hỗ trợ các hoạt động hợp nhất, cập nhật và xóa để kích hoạt các trường hợp sử dụng phức tạp như thu thập dữ liệu thay đổi, hoạt động trên các bảng dimension thay đổi chậm (SCD), upserts trực tuyến, v.v [6].

2.4.3 - Spark Streaming

Spark Streaming là một phần mở rộng của API Spark cốt lõi cho phép xử lý luồng dữ liệu trực tiếp có khả năng mở rộng, thông lượng cao, chịu lỗi. Dữ liệu có thể được nhập từ nhiều nguồn như Kafka, Kinesis (AWS Kinesis - Dịch vụ phát trực tuyến của AWS) và có thể được xử lý bằng các thuật toán phức tạp được biểu thị bằng các hàm cấp cao như map, sort, append và windows. Cuối cùng, dữ liệu đã xử lý có thể được đẩy ra hệ thống tệp, cơ sở dữ liệu và bảng điều khiển trực tiếp. Trên thực tế,

chúng ta có thể áp dụng các thuật toán xử lý đồ thị và máy học của Spark trên các luồng dữ liệu [7].

Hình 2.4.3.1: Luồng xử lý của Spark Streaming [7]

Spark Streaming nhận các luồng dữ liệu đầu vào trực tiếp và chia dữ liệu thành các lô, sau đó được công cụ Spark xử lý để tạo ra luồng kết quả cuối cùng theo lô. Để rõ hơn thì ta có thể xem hình 2.4.3.1.

Hình 2.4.3.2: Việc tách luồng của Spark Streaming [7]

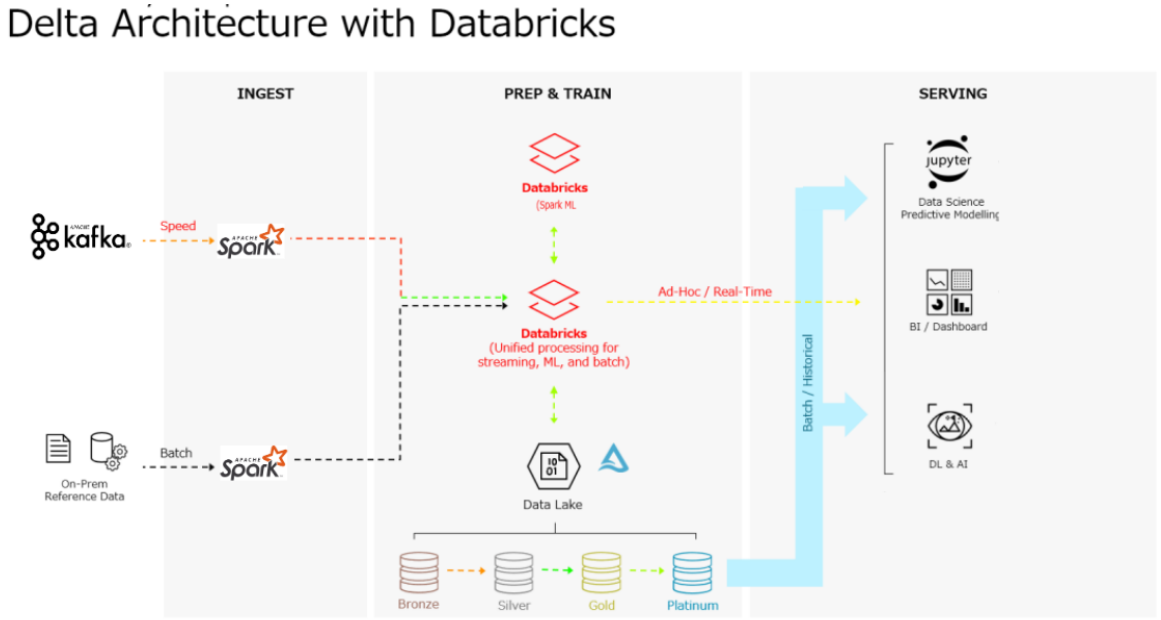
Spark Streaming cung cấp một mức độ trừu tượng cao được gọi là luồng rời rạc hoặc DStream, mô tả ở hình 2.4.3.2, đại diện cho một luồng dữ liệu liên tục. DStream có thể được tạo từ các luồng dữ liệu đầu vào từ các nguồn như Kafka và Kinesis hoặc bằng cách áp dụng các thao tác cấp cao trên các DStream khác. Bên trong, một DStream được biểu diễn dưới dạng một chuỗi các RDD [7].

Chương 3 - Xây dựng ứng dụng

3.1 Xây dựng đường ống dữ liệu

3.1.1 Mô hình kiến trúc Delta

Trong mô hình kiến trúc này, nó bao gồm hai đường ống dữ liệu là speed và batch. Batch là luồng dữ liệu được lấy từ các file dữ liệu dạng JSON được lưu trữ trực tiếp trên hệ điều hành, sau đó lưu trữ vào các bảng Delta. Speed là luồng dữ liệu được phát trực tuyến từ Kafka được lưu trữ vào bảng Bronze, sau đó luồng trực tuyến đi qua các bảng còn lại để thực hiện các phương pháp dữ liệu để dữ liệu trở nên hữu ích hơn cho việc lập báo cáo,..., cho đến bảng cuối cùng lưu trữ dữ liệu hoàn chỉnh nhất trong bảng Platinum.



Hình 3.1.1: Kiến trúc của một Delta Architecture trong Databricks

Sau khi có dữ liệu hoàn chỉnh, bước tiếp đó là sử dụng dữ liệu đó cho các mục đích sử dụng thông tin. Cụ thể, trong kiến trúc này, dữ liệu được sử dụng cho việc biểu diễn Dashboard theo thời gian thực, và huấn luyện mô hình gợi ý nhà hàng.

3.1.2 Dữ liệu đầu vào

Tập dữ liệu mô tả danh sách các nhà hàng được đánh giá bởi các người dùng, được cung cấp bởi Yelp.com. Đây là liên kết dẫn tới bộ dữ liệu: Yelp Dataset

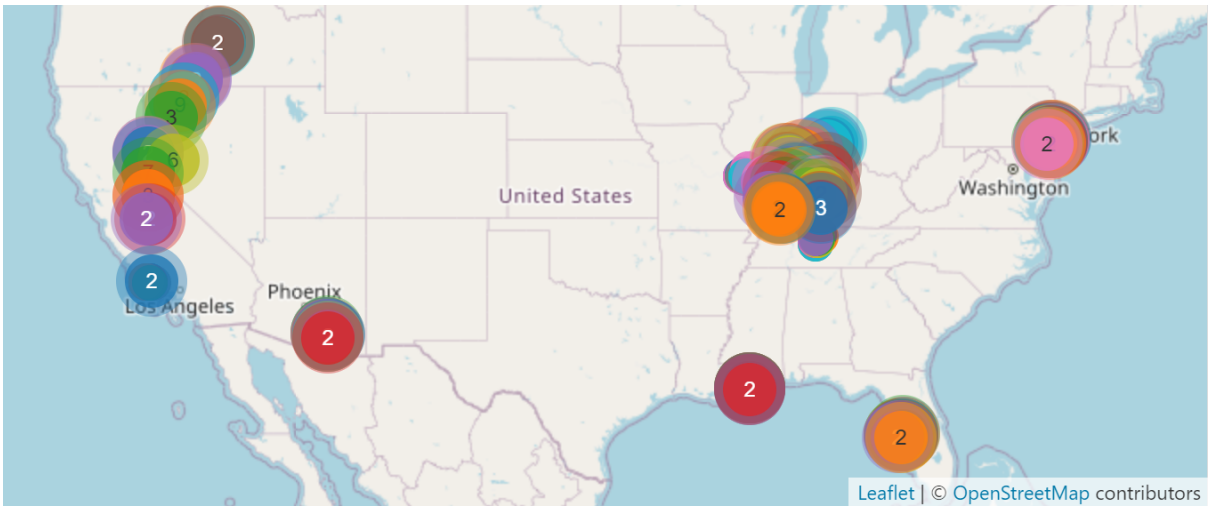
Bộ dữ liệu bao gồm các file dưới đây, được lưu dưới dạng file Json:

- `yelp_academic_dataset_business.json`
- `yelp_academic_dataset_checkin.json`
- `yelp_academic_dataset_review.json`
- `yelp_academic_dataset_tip.json`
- `yelp_academic_dataset_user.json`

3.1.3 Xử lý dữ liệu bằng batch job

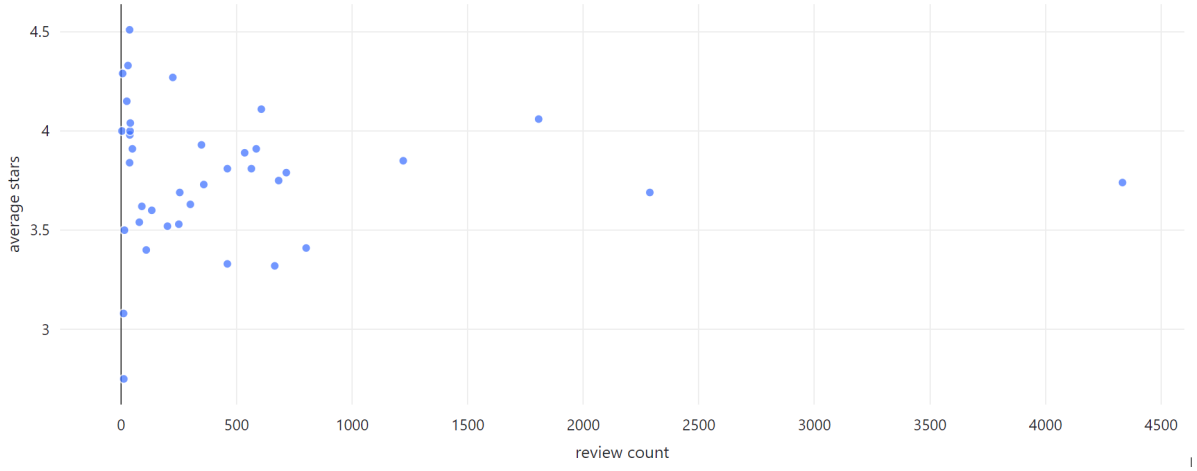
Bởi vì bảng dữ liệu chứa thông tin về nhà hàng, người dùng, check-in, tip rất ít khi thay đổi, vậy nên không cần phải xử lý cập nhật thời gian thực mà chỉ cần một bảng cập nhật mỗi ngày 1 lần cho các bảng này.

Thông tin về nhà hàng bao gồm như mã số, tên, địa chỉ,... có thể biểu diễn dưới nhiều dạng biểu đồ khác nhau, nhưng ở đây, có thể xem sự phân bổ của nhà hàng, vị trí địa lý dựa vào biểu đồ 3.1.3.1.



3.1.3.1: Vị trí địa lý của các doanh nghiệp dịch vụ ăn uống

Một phần quan trọng không thể thiếu đó là thông tin người dùng, trong bộ dữ liệu, nó đang được mô tả với các thuộc tính như user id, tên, số sao đánh giá trung bình, … Dưới đây là biểu đồ thể hiện sự tương quan giữa số sao trung bình và tổng số lượt đánh giá của người dùng.



Hình 3.1.3.2: Sự tập trung về ratings của các user

Tùy thuộc vào lượng dữ liệu hiện có trong bảng, biểu đồ sẽ có sự thay đổi khác nhau. Trong hình 3.1.3.2, bởi vì lượng dữ liệu trong bảng thông tin người dùng không

nhiều nên các điểm đang tập trung ở gần mốc 0 nhất. Hình 3.1.3.3 cho ta thấy những nhà hàng cũ nhất ở database.

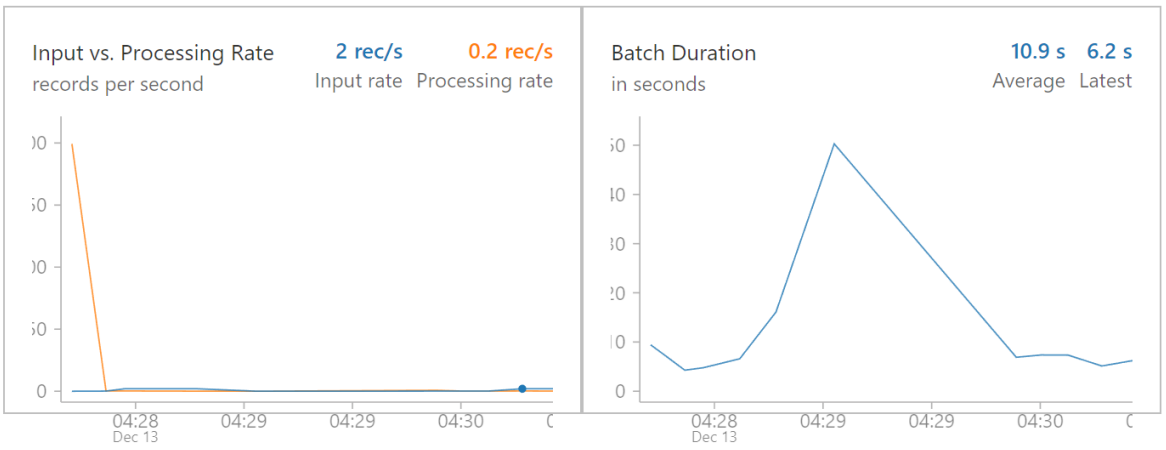
#	business_id	date
1	-5Rah4ZvWsDu4oilUZxhtw	2010-01-23 00:22:09, 2010-02-14 23:54:30, 2010-02-27 23:42:38, 2010-03-18 00:51:33, 2010-04-16 00:01:17, 2010-...
2	-3IOd5YntpkbK6RwT3HYtA	2010-01-29 00:11:34, 2010-08-23 03:10:43, 2010-10-19 22:21:09, 2010-12-03 17:01:45, 2011-08-29 22:41:50, 2011-...
3	-2aGyAUenQEZWAtNUnMHzg	2010-02-10 20:01:19, 2010-03-31 15:09:55, 2010-06-30 15:18:16, 2010-08-19 15:09:12, 2010-11-27 19:38:46, 2011-...
4	-4HMjEGQgduIMMTe0WPQBA	2010-02-13 01:37:42, 2010-03-27 01:10:00, 2010-06-21 23:09:45, 2010-06-22 01:04:26, 2010-07-15 20:15:56, 2010-...
5	-6klZWnXPuDC6JiQJ-A1fg	2010-02-13 16:06:09, 2010-07-11 17:30:59, 2010-07-17 20:33:54, 2010-08-03 22:53:56, 2010-08-31 21:19:21, 2010-...
6	-ATiAtTikuGuqvaW2O6tNA	2010-02-14 18:13:35, 2010-02-20 18:02:19, 2010-05-07 21:53:44, 2010-07-31 13:22:10, 2010-08-06 14:19:55, 2010-...
7	-1MhPXk1FglglUAmuPLIGg	2010-02-18 06:23:47, 2010-04-09 05:41:02, 2010-07-23 04:31:32, 2010-07-25 22:14:17, 2010-07-31 01:51:09, 2010-...
8	-BLKZfw-FX7602K59OpBgg	2010-02-26 18:09:47, 2012-02-22 01:11:01, 2012-02-22 23:07:16, 2012-05-31 17:18:21, 2014-05-27 22:55:26, 2014-...
9	-9NmUeTphyS9Lq1o9MACGw	2010-02-26 23:44:51, 2010-03-15 01:46:05, 2010-09-28 23:18:08, 2010-09-28 23:24:51, 2010-10-02 16:47:03, 2010-...
10	-A10ZTqT4X49cneKlxXTcw	2010-03-17 22:13:38, 2010-03-29 16:21:49, 2010-05-14 16:17:46, 2010-05-17 16:17:55, 2010-05-26 16:09:33, 2010-...
11	-2gmbMDzKgYZ_8DOnJtPyw	2010-03-21 01:40:57, 2011-02-07 19:04:21, 2011-02-08 18:40:44, 2011-02-11 22:27:02, 2011-06-30 22:21:22, 2012-...
12	-0i2KNr7WrCsDF5m0IViJq	2010-03-25 17:00:35, 2010-08-11 17:50:29, 2010-08-21 17:42:01, 2010-09-10 23:20:11, 2010-09-26 23:58:29, 2010-...

Hình 3.1.3.3: Top các nhà hàng thành lập cũ nhất

3.1.4 Thu thập dữ liệu real-time bằng streaming job

Bởi vì trong kiến trúc này, dữ liệu phát trực tuyến sẽ được lấy từ nguồn và di qua các bảng, vậy nên đầu tiên cần lấy dữ liệu từ Kafka lưu trữ vào bảng 'Bronze', sau đó lấy từ 'Bronze', xử lý vào đẩy vào bảng 'Silver', tiếp tục lấy data từ bảng 'Silver' để thêm cột và lưu vào bảng 'Gold', cuối cùng lấy dữ liệu từ bảng 'Gold', join với các bảng từ batch job để lấy thêm thông tin của nhà hàng và người dùng, sau đó lưu vào bảng 'Platinum'. Luồng dữ liệu đi từ nguồn trực tuyến, sau qua các bảng cho tới bảng cuối cùng, dữ liệu luôn luôn cập nhật, luôn luôn được lưu trữ vào các bảng.

Trước tiên data dưới dạng thô sẽ đi vào bảng bronze, hình 3.1.4.1 với 3.1.4.2 mô tả về bảng bronze.

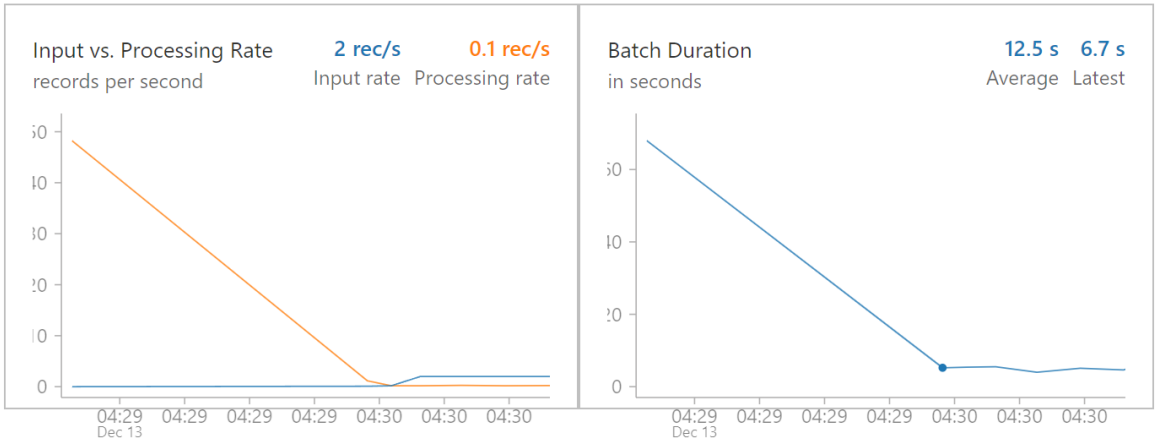


Hình 3.1.4.1: Biểu đồ RealTime Bảng Bronze

	user_id	business_id	stars	date	cool	funny	useful	timestamp
1	42ZID-0WVKEX1QFMEIp09Q	l6L0Zxi5Ww0zEWSAVgngeQ	5	2017-11-06	0	0	0	2022-12-15T18:42:41.967+0000
2	c35f8FumQy5045zFblir6w	_uyLoz0BbrQlVmUEm_td1Q	3	2018-06-09	0	0	0	2022-12-15T18:43:30.450+0000
3	0XmgOinrZWNO15DlimRQeg	PpZqlVUAP2i8_hNfpQyKhg	5	2012-03-01	0	0	0	2022-12-15T18:42:29.841+0000
4	FNf6kM1wfeBeB2UW6pif7g	qdpdUjdkEUpzYDdce2yZng	4	2016-11-09	0	0	0	2022-12-15T18:43:42.624+0000
5	yyxEa8y4HGij_m7g1wDKWg	Jc6MxMyCLp4DMfN4C1t1aQ	3	2015-04-15	0	0	0	2022-12-15T18:43:18.328+0000
6	jIT1qbzpfPQ9xRI_CBw6PA	jX8i0O2x-koganf8WwHXdA	5	2017-01-27	0	0	0	2022-12-15T18:43:54.743+0000

Hình 3.1.4.2: Data nhận vào Bronze

Sau đó, từ bảng bronze, nó tiếp tục được lấy real time từ bronze, ở đây chủ yếu nó lọc bỏ NULL và NA với data nhận được. Rồi tiếp đến data sẽ được đưa vào bảng Silver. Hình 3.1.4.3 và 3.1.4.4 mô tả dữ liệu nhận được khi bảo bảng Silver.

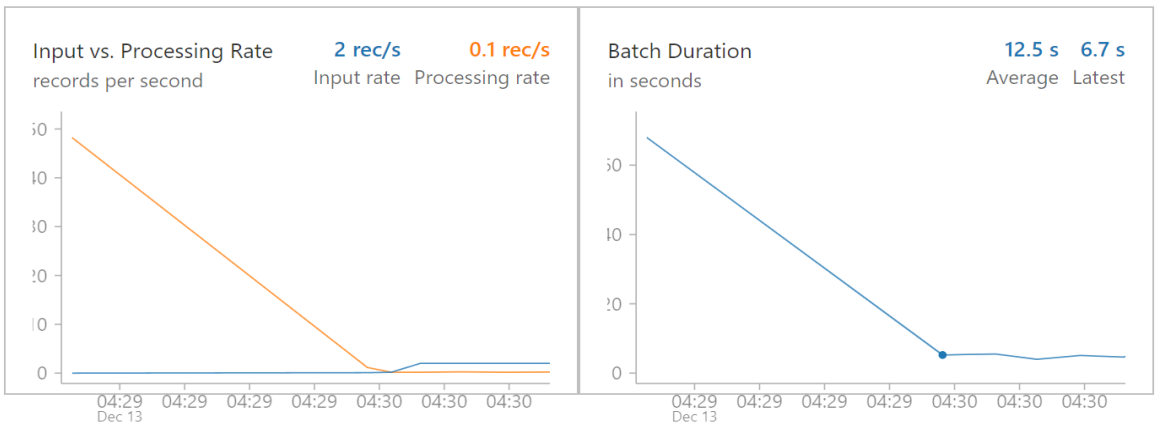


Hình 3.1.4.3: Biểu đồ RealTime Bảng Silver

	user_id	business_id	stars	date	cool	funny	useful	timestamp	year
1	4mbLmbA-thaDIZTlgxsaCQ	q0Fi4n7shUTmlxl-mMPVXA	2	2016-07-04	0	0	1	2022-12-15T18:41:04.948+0000	2016
2	qWSAH4MzFbHV6UsseJVlZg	dUctvEfHQccW_uxtRup2QQ	1	2015-12-20	0	0	0	2022-12-15T18:41:53.441+0000	2015
3	0VMuCPgwZliInxGWfJnxKQ	25Uww0C0wvF9CZ_3B6vWtA	5	2016-07-04	0	0	0	2022-12-15T18:41:29.193+0000	2016
4	f10WH1fXhy-68r4AEEhAWA	9OG5YkX1g2GReZM0AskizA	4	2016-01-30	0	0	0	2022-12-15T18:41:41.320+0000	2016
5	CNyXcn0c0V5CFmigqqw-Xg	oY5LFo6Yxxf32ePna6mEUQ	5	2014-12-30	0	0	1	2022-12-15T18:42:17.720+0000	2014

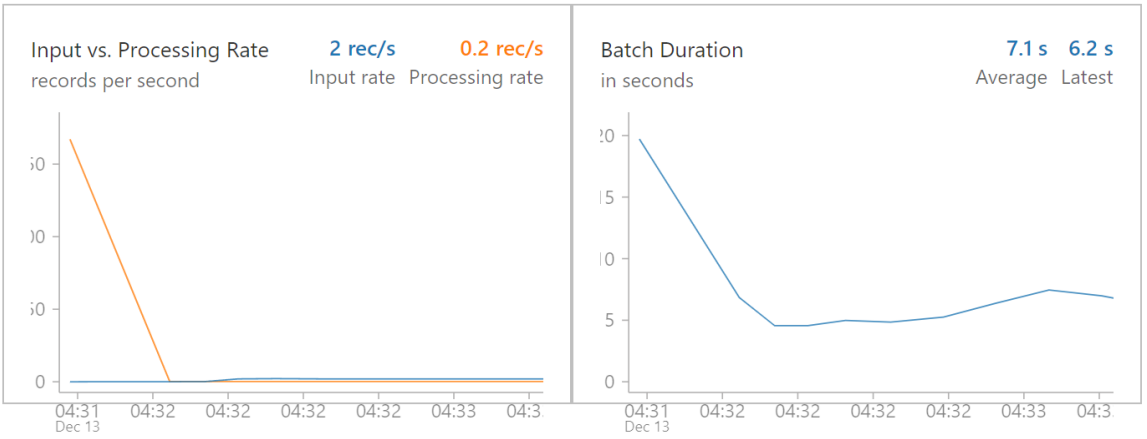
Hình 3.1.4.4: Data nhận vào Silver

Sau khi lưu trữ dữ liệu vào bảng ‘Silver’, luồng streaming sẽ tiếp tục lấy dữ liệu từ bảng đó, xử lý và lưu trữ vào bảng ‘Gold’.



Hình 3.1.4.5: Biểu đồ RealTime Bảng Gold

Dữ liệu tiếp tục từ bảng ‘Gold’ join với hai bảng business và user để ra được bảng cuối cùng là ‘Platinum’.



Hình 3.1.4.6: Biểu đồ RealTime Bảng Platinum

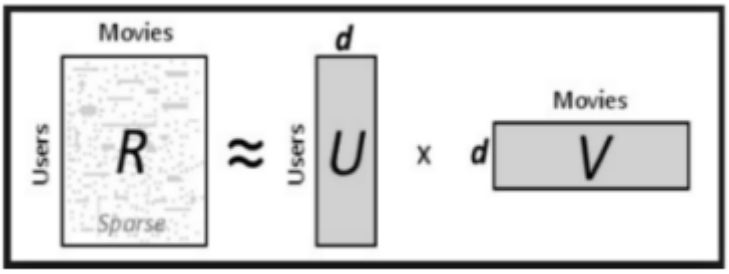
Cuối cùng, dữ liệu sau khi join với các thuộc cần thiết. Tại đây cũng là nơi cuối cùng trước khi dữ liệu được dùng cho visualize hay cụ thể ở đây là ứng dụng cho mô hình dự đoán.

3.2 Huấn luyện mô hình

- Ở đây, nhóm đã đưa ra một ứng dụng đơn giản, đó là dùng tập data thu thập được tạo một mô hình recommendation system đơn giản, cụ thể ở đây nhóm sử dụng thuật toán ALS. Trong mô hình này, nhóm chỉ tập trung predict cho users những nhà hàng ưa thích.

Thuật toán ALS(Alternating Least Square): Giải thích đơn giản thì thuật toán thuộc matrix fractorization trong recommendation system. Ở hình 3.2.1, hình bên trái biểu thị một ma trận đánh giá với user và movie (trong bài toán của nhóm là business). Vì tính chất rỗng của ma trận này (user chỉ đánh giá ở 1 vài nhà hàng). Do đó, thuật toán matrix factorization ra đời, mục đích của nó là phân rã ma trận ban đầu làm 2 ma trận con, với d ở đây có thể coi như nhân tố ẩn, điều này giúp ta giải quyết được các giá trị rỗng không cần thiết. Khi cần thiết, ta chỉ cần lấy vector trong 2 ma trận này với giá trị user và business tương ứng, điều này giúp cải thiện tốc độ dự đoán rất nhiều.[8]

Tóm lại, ở mức tổng quan, thuật toán này dựa vào các thuộc tính user, item và rating của user cho item đó, mục đích để tính ra mức tương quan của nó với các giá trị có sẵn khác mà gợi ý các sản phẩm phù hợp. Có thể nói thuật toán này là hiệu nhất vì nó kết hợp cả 2 phương pháp giảm chiều matrix và tìm kiếm nhóm tương đồng (lọc cộng tác).

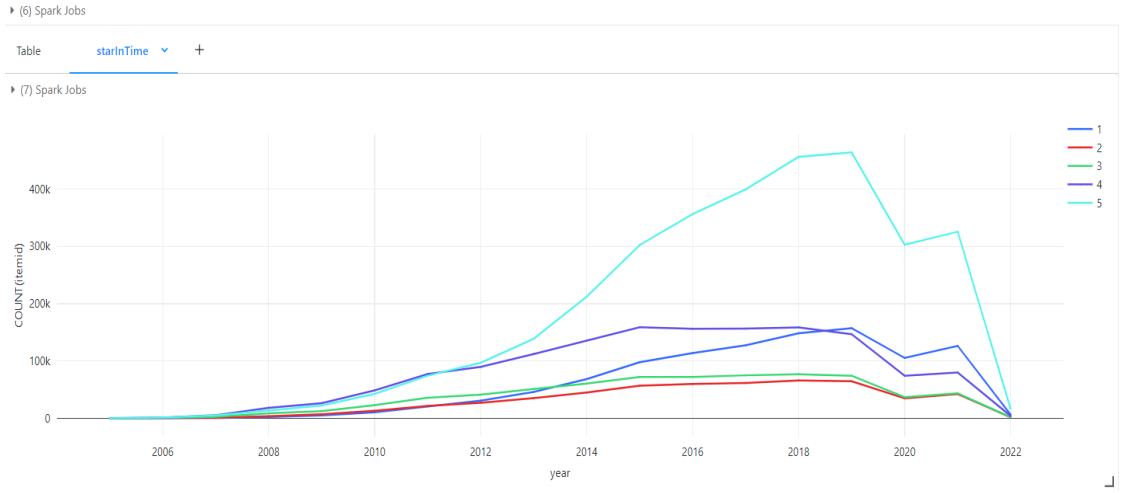


Hình 3.2.1: Mô tả thuật toán ALS [8]

Hình 3.2.1 mô tả về cách nó giảm dimension, từ đó nó tận dụng lại lọc cộng tác.

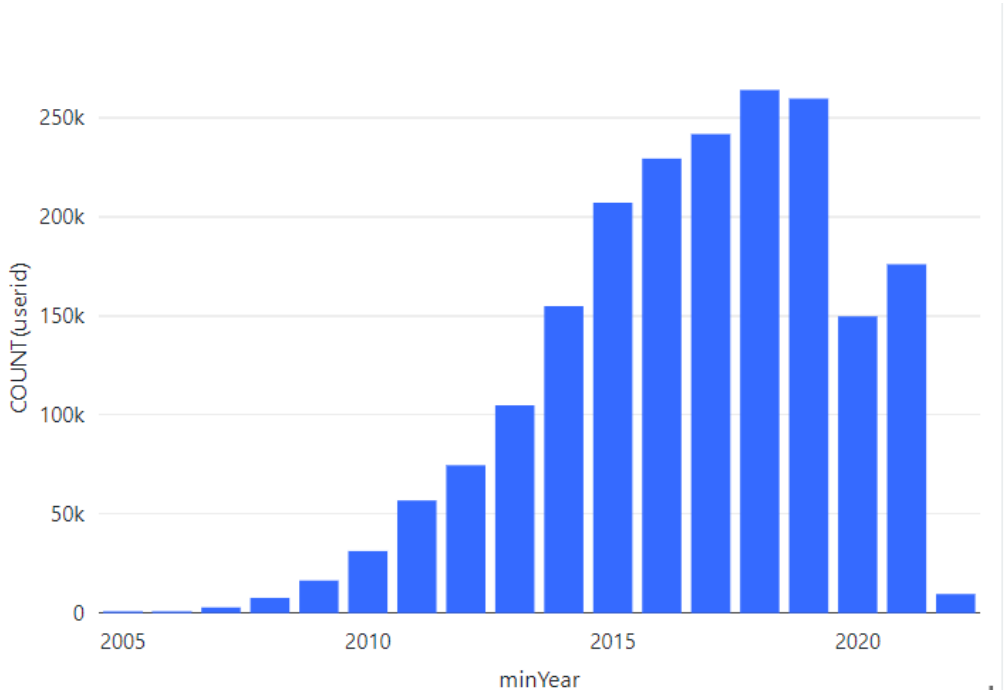
Lọc cộng tác: giải thích đơn giản thì đây là phương pháp chọn ra các user có sở thích giống nhau, thông thường là thông qua việc rating sản phẩm.

Ở đây mục tiêu của sản phẩm là dự đoán tương lai nên ta sẽ lấy những dữ liệu gần nhất dựa vào date trong bảng review.



Hình 3.2.2: Phân bố vote theo thời gian

Trước tiên ta nhìn vào hình 3.2.2 sẽ thấy lượng người vote các nhà hàng sẽ tập trung ở thời gian gần đây nhiều hơn, và mục tiêu ta sẽ không dự đoán ở quá khứ. Cũng có thể thấy rằng lượng người có xu hướng vote 5 sao cũng tập trung đa số ở hiện tại.

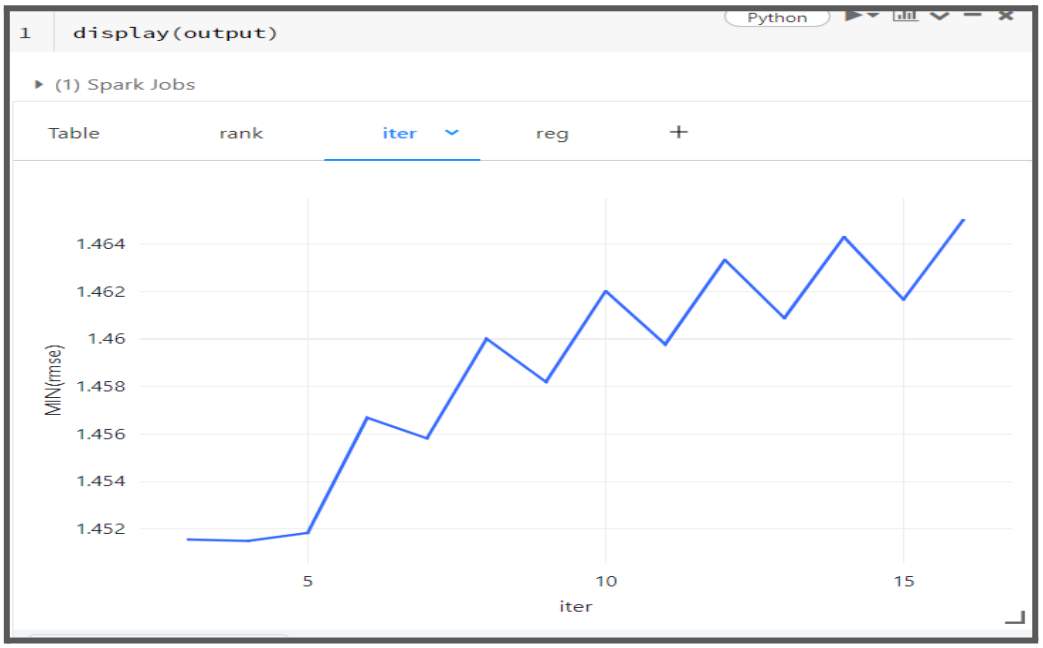


Hình 3.2.3: Phân bố user vote lần đầu theo thời gian

Hình 3.2.3 chính là mô tả đơn giản hơn của hình 3.2.2. Trước tiên dữ liệu rating ban đầu có gần 7 triệu rating, matrix MxN có Ta chia dữ liệu làm 7/2/1 (train/validate/test).

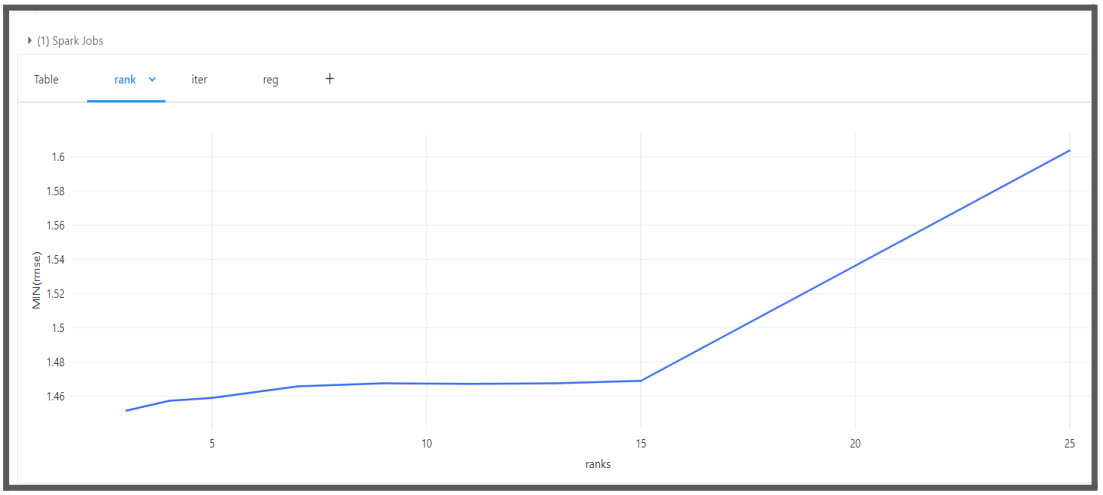
Tuning: Trong ALS, ta phải tuning các thông số để minimize RMSE. Có 3 thông số quan trọng trong mô hình:

+ Rank: số nhân tố ẩn (latent factor) của mô hình.
+ iterations: số lần lặp
+ regularization: tham số chuẩn hóa (để tránh over fitting)



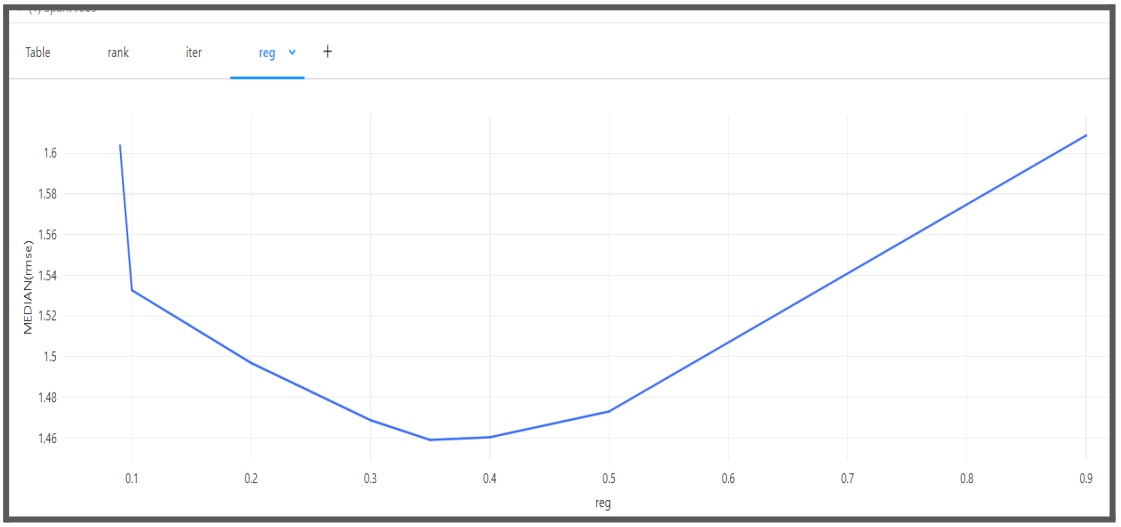
Hình 3.2.4: Tinh chỉnh siêu tham số iter (số lần lặp)

Hình 3.2.4 mô tả sự dao động của iter, về cơ bản thì việc xét trung bình thì hàm này hiếm khi thay đổi nhiều, tuy nhiên nếu xét là giá trị min thì iter=4 sẽ dễ đạt RMSE thấp nhất.



Hình 3.2.5: Tinh chỉnh siêu tham số rank(số nhân tố ẩn)

Hình 3.2.5 mô tả sự dao động của rank, ở đây chúng ta muốn giảm đi sự phức tạp ở factor, theo ta quan sát, điểm rank=4 sẽ an toàn vì ở đây giá trị RMSE khá thấp, điểm 3 thì lại sát cận nên ta sẽ bỏ qua. Do đó ta sẽ chọn rank=4



Hình 3.2.6: Tinh chỉnh siêu tham số reg (regularization)

Hình 3.2.6 mô tả sự dao động của regularization, đây là thông số quan trọng trong việc giảm overfitting. Nhìn vào hình sẽ dễ dàng nhận thấy nó hội tụ ở vị trí reg=0.35. Do đó, ta sẽ chọn reg=0.35 sẽ là giá trị tốt nhất.

Như vậy, sau quá trình kiểm tra ta sẽ được rank=4, maxIter=4, reg=0.35.

Kết quả cho lần chạy tốt nhất là: RMSE= 1.3710067457429629

Lý do mô hình có sai số vẫn còn cao vì do mô hình thật sự quá thưa thớt, trong khi matrix MxN là 1,9M x 190K nhưng chỉ có khoảng 6M record, tuy nhiên về lâu dài thì mô hình này sẽ có sự cải thiện do số rating về sau đang ngày càng nhiều.

Sau khi train xong nếu nhận thấy model đảm bảo độ tin cậy, ta có thể save mode, để dùng cho việc predicct lần sau mà không cần phải huấn luyện lại. Hình 3.2.7 ghi một folder khi chứa một model được save lại.

```
/databricks/driver/yelp_dataset/save
itemFactors  metadata  userFactors
```

Hình 3.2.7 Folder chứa save model

Sau đó, ta có thể lấy mode để recommend cho user và collect user mong muốn:

userid	businessid	rating
20	26777	5.2499027
20	59308	4.724409
20	13385	4.68757
20	41707	4.6628594
15	26777	5.0053034
15	26624	4.466009
15	41707	4.3750315
15	13385	4.3227687

Hình 3.2.8: Recommendation

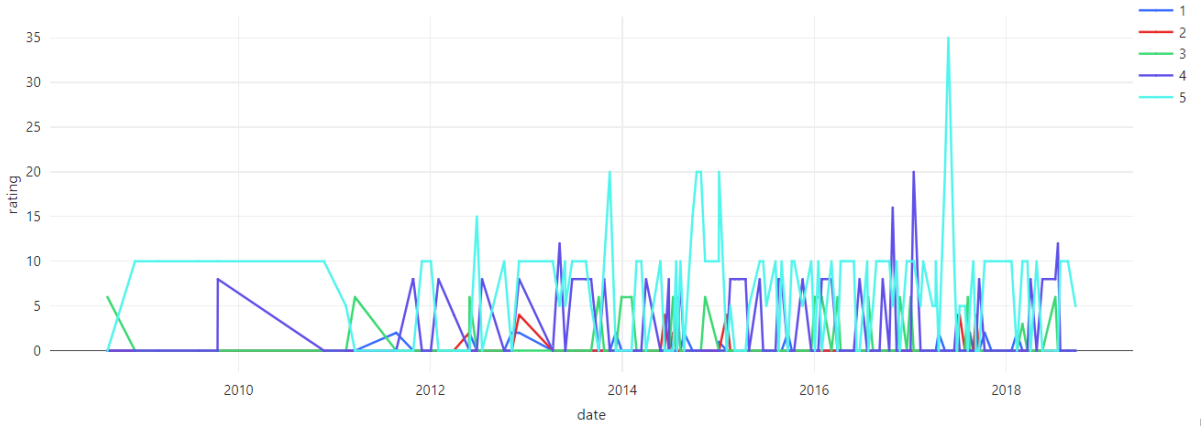
Hình 3.2.8 chính là mô tả output của recommendation cho user 15 và 20. Tất cả các nhà hàng được recommend đều có rating cao mới được đề xuất cho user.

3.3 Xây dựng dashboard thời gian thực

Sau khi có dữ liệu hoàn chỉnh nhất, bước tiếp theo là sử dụng lượng dữ liệu đó cho các mục đích sử dụng thông tin. Cụ thể ở đây, chúng ta cần xây dựng một trang báo cáo thời gian thực. Đầu tiên, hình 3.3.1 là thẻ biểu thị số lượt đang được ghi vào hệ thống.

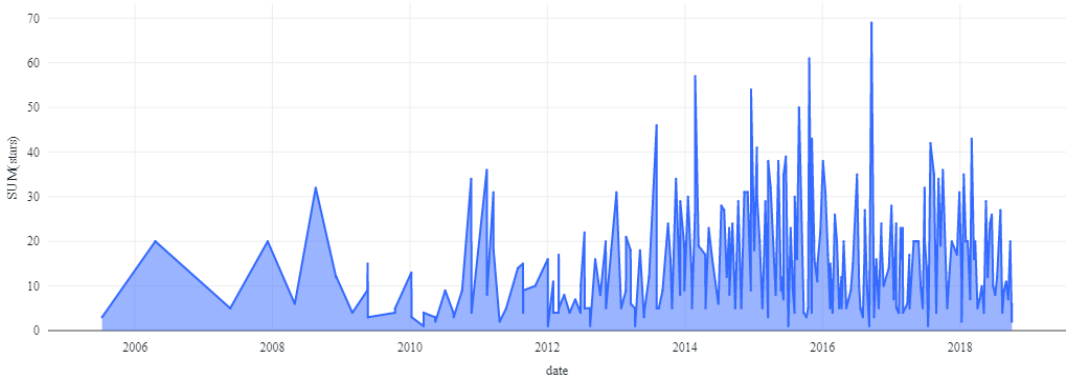
Hình 3.3.1: Thẻ biểu thị số lượt đánh giá của người

Điều mà chúng ta cần quan tâm nhất trong mỗi lượt đánh giá của người dùng chính là số lượt đánh giá mà họ cung cấp cho mình. Vậy nên, ở đây chúng ta cần vẽ một biểu đồ thể hiện số sao qua từng năm để xem rằng năm nào thì nhiều nhà hàng được đánh giá 5 sao nhiều nhất, tương tự cho các chỉ số xếp hạng còn lại. Cụ thể là ở hình 3.3.2 mô tả số sao qua từng năm:



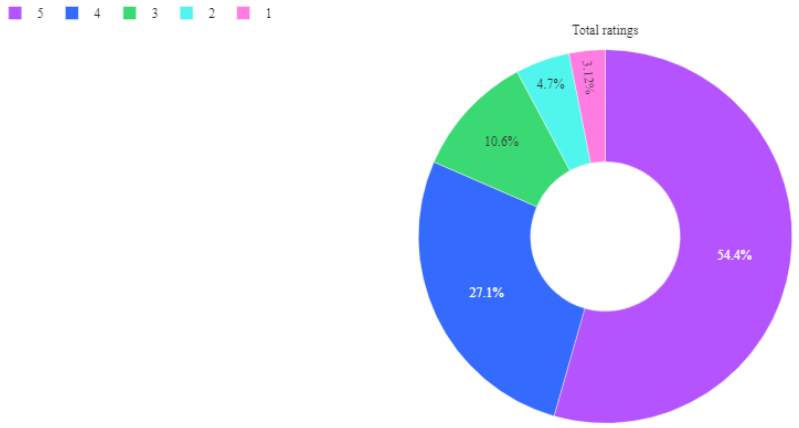
Hình 3.3.2: Biểu đồ đường biểu diễn số xếp hạng nhà hàng qua từng năm

Sau khi đã có số lượng xếp hạng cụ thể, tiếp theo cần phải biểu diễn số lượng xếp hạng đó qua từng năm, cụ thể ở hình 3.3.3, Ở đây chúng ta không chỉ xem chỉ số xếp hạng nào chiếm số lượng nhiều nhất, mà cần phải quan tâm tới tổng số lượng đánh giá qua từng năm để xem ở những năm nào số lượng đánh giá nhiều nhất.



Hình 3.3.3: Biểu đồ vùng biểu diễn số lượng đánh giá phân bổ qua từng năm

Theo hình 3.3.4, tỉ lệ phân bổ của loại xếp hạng cũng quan trọng không kém trong quá trình tìm hiểu về xu hướng của người dùng. Chúng ta thực hiện điều đó qua cách vẽ biểu đồ tròn, xem xét tỉ lệ giữa chúng.



Hình 3.3.4: Biểu đồ tròn biểu diễn tỉ lệ phân bổ xếp hạng

Biểu đồ trên đó thể hiện loại xếp hạng nào chiếm nhiều nhất, ở đây 5 sao đang chiếm số lượng nhiều nhất với 54.4%, ít nhất là 1 sao với 3.1%.

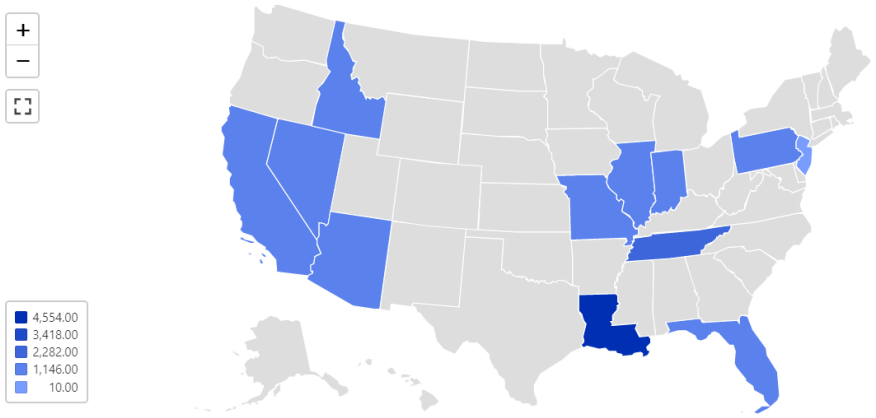
Khi đã có một số thông tin về xếp hạng cũng như lượt đánh giá, chúng ta cần quan tâm tới các doanh nghiệp nhà hàng để có được thông tin hữu ích hơn. Điều đầu tiên là xem xét nhà hàng nào có số lượt đánh giá cao nhất, bởi điều có nghĩa là càng nhiều lượt đánh giá thì nhà hàng đó càng nổi tiếng và nhiều người biết đến.

Hình bên dưới sử dụng word cloud biểu diễn tên của các nhà hàng. Tên nhà hàng nào càng lớn thì nhà hàng đó càng có nhiều lượt đánh giá.

Hình 3.3.5: Biểu đồ word cloud biểu diễn số lượng đánh giá của nhà hàng

Ở biểu đồ 3.3.5, nhà hàng The Green Project đang chiếm số lượng lớn nhất với cỡ chữ nổi bật hơn các nhà hàng còn lại.

Hình 3.3.6 là bản đồ biểu diễn số lượt đánh giá mà các nhà hàng nhận được qua các bang ở nước Mỹ. Số lượng hiện trên bản đồ sẽ nhiều lên theo thời gian bởi dữ liệu được cập nhật trực tuyến.



Hình 3.3.6: Biểu đồ map biểu diễn số lượt đánh giá của nhà hàng

Màu xanh càng đậm nghĩa là số lượt đánh giá nhà hàng ở bang đó càng nhiều, màu xanh càng nhạt nghĩa là số lượng đánh giá của người dùng dành cho bang đó càng ít.

3.4 Xây dựng hệ thống gợi ý dùng mô hình đã huấn luyện

Sau khi đã huấn luyện mô hình gợi ý nhà hàng với dữ liệu trước đó, chúng ta sẽ đi vào sử dụng mô hình để gợi ý nhà hàng cho người dùng. Dùng mô hình gợi ý 4 nhà hàng cho 2 người dùng có userid là 20 và 15, ta thu được kết quả như hình dưới đây:

```
+------+----------+---------+
|userid|businessid|   rating|
+------+----------+---------+
|    20|     26777|5.2499027|
|    20|     59308| 4.724409|
|    20|     13385|  4.68757|
|    20|     41707|4.6628594|
|    15|     26777|5.0053034|
|    15|     26624| 4.466009|
|    15|     41707|4.3750315|
|    15|     13385|4.3227687|
+------+----------+---------+
```

Hình 3.4.1: Kết quả gợi ý nhà hàng cho người dùng

Mặc dù sai số RMSE trong lúc huấn luận mô hình khá cao, những khi gợi ý mô hình sẽ đưa ra rating cao nhất vậy nên kết quả gợi ý vẫn sẽ khá ổn và chấp nhập được.

Chương IV - Kết luận

Về mặt tìm hiểu và ứng dụng, nhóm đã tìm hiểu được các kiến trúc xử lý luồng dữ liệu trực tuyến như Lambda, Kappa và các ưu, nhược điểm của các kiến trúc. Các giải pháp cho các nhược điểm đó, là sự khởi đầu của kiến trúc Delta, một kiến trúc hoàn toàn mới không những cải thiện được những khuyết điểm của các kiến trúc đời trước, mà còn khắc phục được các nhược điểm về hiệu suất truy vấn, khả năng cung cấp dữ liệu trong thời gian thực của Data Lake và Data Warehouse. Các thành phần trong kiến trúc Delta bao gồm công cụ phát trực tuyến hàng đợi Kafka, Spark Structured Streaming, Delta Lake. Từ đó, nhóm đã ứng dụng những kiến thức đã tìm hiểu, nghiên cứu được để áp dụng vào xây dựng kiến trúc luồng xử lý dữ liệu trong thời gian thực ứng dụng kiến trúc Delta. Sau đó ở tầng sử dụng dữ liệu sẽ dùng để kết hợp bảng báo cáo gồm các biểu đồ thay đổi theo thời gian thực và ứng dụng máy học (ở đây nhóm áp dụng thuật toán ALS) đưa ra các nhà hàng gợi ý cho người dùng.

Ở mặt hạn chế, việc xây dựng luồng xử lý dữ liệu lớn rõ ràng cần tài nguyên nhiều hơn là xây dựng ứng dụng web hay ứng dụng điện thoại, khi xây dựng ứng dụng này, nhóm cũng đã gặp phải trường hợp tràn bộ nhớ khi sử dụng Spark, cụ thể là trong việc truy vấn, chuyển đổi một lượng lớn dữ liệu và huấn luyện mô hình học máy. Bên cạnh đó, dữ liệu cũng là một điều cần lưu ý khi phát triển các đường ống, nhất là việc cần một lượng dữ liệu đủ lớn để có thể xây dựng luồng xử lý dữ liệu lớn đúng như ý nghĩa của nó.

Còn về hướng phát triển, nhóm sẽ lưu ý xem xét các nguồn dữ liệu trực tuyến có sẵn khác thay vì sử dụng tệp như bây giờ. Các công cụ, môi trường phát triển ứng dụng cũng sẽ là điều mà nhóm sẽ xem xét thay đổi để xây dựng luồng dữ liệu lớn có khả năng sử dụng trong thực tế hơn. Apache Airflow đang là công cụ mà nhóm hướng tới để quản lý, điều phối các luồng dữ liệu batch thay vì thiết lập thời gian chạy bằng mã. Ngoài ra, nhóm đang nghiên cứu sử dụng các công nghệ container như Docker để đóng gói mã nguồn, thư viện để có thể dễ dàng triển khai luồng dữ liệu này lên các nền tảng đám mây như Amazon Web Services, Google Cloud Platform. Đồng thời, thay vì viết bằng notebook, nhóm có thể phát triển nó thành một trang web có ứng dụng kiến trúc delta cũng như áp dụng thuật toán gợi ý để đề xuất sản phẩm trực tuyến.

TÀI LIỆU THAM KHẢO

[1] Michael Armbrust el al., Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores, Proceedings of the VLDB Endowment, Vol. 13, No. 12, 2020. https://www.databricks.com/wp-content/uploads/2020/08/p975-armbrust.pdf

[2] Michael Armbrust et al., Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics, Conference on Innovative Data Systems Research (CIDR), 2021.
https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

[3] Jixin Jia, 29-3-2020, Lambda kappa and now delta,

https://jixjia.com/delta-architecture

[4] Gwen Shapira et al., 10-10-2017, Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale

[5] Tathagata Das el al., Get Your Free Copy of Delta Lake: The Definitive Guide, Engineering Blog, Vol. 1, No. 7-14, 22-6-2021.

https://www.databricks.com/blog/2021/06/22/get-your-free-copy-of-delta-lake-the-definitive-guide-early-release.html

[6] Vini Jaiswal el al., 6-12-2020, Welcome to the Delta Lake documentation,
https://docs.delta.io/2.0.0/index.html

[7] Matei Zaharia el al., 28-2-2022, Spark Streaming Programming Guide,
https://spark.apache.org/docs/latest/streaming-programming-guide.html

[8] P. Srinivasa Rao et al., Matrix Factorization Based Recommendation System using Hybrid Optimization Technique, IAI.EU, Vol. 8, No 14, 9-2021.