ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

Tính toán song song

Triển khai và đánh giá hiệu năng xử lý dữ liệu lớn trong mô hình Big data mới dựa vào Lakehouse sử dụng công cụ Spark.

GVHD: Thoại Nam
SV thực hiện: Đặng Nguyễn Điền Trung – 1814501
Trần Phương Tĩnh – 1927038
Phạm Ngọc Sang – 1813810

TP. HỒ CHÍ MINH, THÁNG 10/2021

Mục lục

1 Giới thiệu đề tài **2**

2 Data Streaming **2**
2.1 Data Streaming là gì? . . . 2
2.2 Vì sao chúng ta cần Data Streaming . . . 2

3 Mô hình Lambda, Kappa & Delta **3**
3.1 Lambda . . . 3
3.2 Kappa . . . 3

4 Công cụ Delta Lake **4**
4.1 Data lake . . . 4
4.2 Lakehouse . . . 4
4.3 Delta Lake . . . 4
4.3.1 Giới thiệu . . . 4
4.3.2 Kiến trúc tổng quan . . . 5
4.3.3 Cách cài đặt và một số thao tác cơ bản . . . 5
4.3.3.a Cài đặt Apache Spark với Delta Lake . . . 5
4.3.3.b Tạo bảng mới . . . 6
4.3.3.c Đọc dữ liệu từ bảng . . . 6
4.3.3.d Cập nhật dữ liệu trong bảng . . . 6
4.3.3.e Ghi dòng dữ liệu vào bảng . . . 7
4.3.3.f Đọc dòng thay đổi từ bảng . . . 7

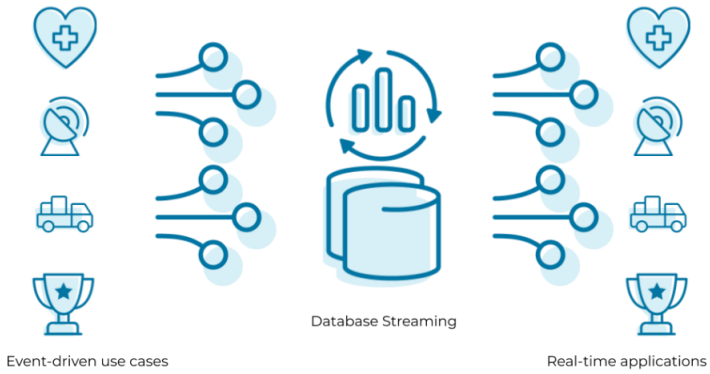
1 Giới thiệu đề tài

Xu hướng hướng dữ liệu (data-driven) là giải pháp tất yếu, tiên tiến để giải quyết nhiều bài toán. Kiến trúc tổng thể xử lý dữ liệu lớn khác với mô hình truyền thống sử dụng Data Warehouse với kỹ thuật trích xuất, biến đổi và tải ETL. Một số mô hình mới được đề xuất như Lambda, Kappa & Delta. Bài toán chính là đánh giá mô hình Delta (mới từ 2019) dựa trên công cụ mã mở Delta Lake phát triển trên Spark.

2 Data Streaming

2.1 Data Streaming là gì?

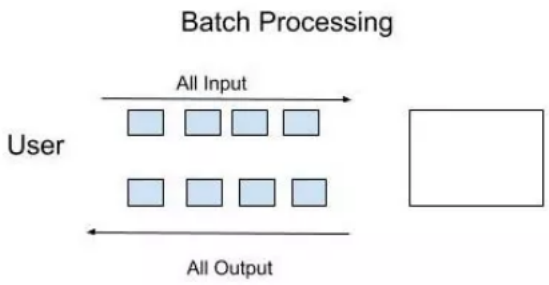
Data Streaming hay còn được hiểu là Stream Processing, là luồng dữ liệu liên tục được tạo ra bởi nhiều nguồn khác nhau. Bằng cách sử dụng Data Streaming, các luồng dữ liệu có thể được xử lý, lưu trữ, phân tích và hoạt động như nó được tạo trong thời gian thực.



Hình 1: *Data Screaming*

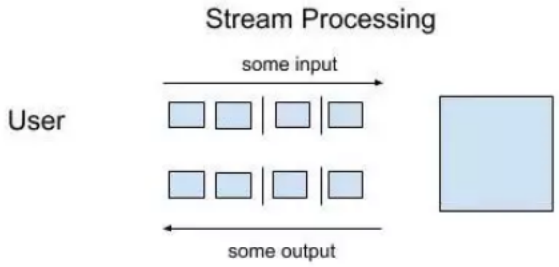
2.2 Vì sao chúng ta cần Data Streaming

Bối cảnh lập trình hiện nay bùng nổ với big data và các công nghệ đính kèm nó. Cùng với đó là sự gia tăng mạnh mẽ về thị trường thiết bị di động cũng như sự phát triển nhanh chóng của các ứng dụng di động. Tuy nhiên dù thị trường thiết bị di động lớn đến đâu, ứng dụng di động có biến động ra sao thì có một thứ bất biến đó là mỗi ngày chúng ta phải xử lý ngày càng nhiều dữ liệu. Dữ liệu tăng lên thì nhu cầu phân tích tận dụng dữ liệu đó cũng tăng lên tương ứng . Việc xử lý lượng lớn dữ liệu thường được thực hiện bằng Batch Processing (dữ liệu chứa hàng triệu bản ghi mỗi ngày được lưu trữ dưới dạng các tệp tin hoặc bản ghi. Các tệp tin này sẽ trải qua quá trình xử lý vào cuối ngày để phân tích theo cách khác nhau mà công ty muốn thực hiện.)



Hình 2: *Batch Processing*

Càng ngày các tổ chức càng thấy rằng họ cần xử lý dữ liệu ngay khi có sẵn (stream processing), có nghĩa là bạn phải xử lý từng bản ghi ngay khi nó có sẵn. Ví dụ: Có bao nhiêu đăng nhập không hợp lệ trong vòng 10 phút qua? Có bao nhiêu tính năng được phát hành gần đây được người dùng sử dụng? Xu hướng hiện tại đang là gì ? Rõ ràng cần phải có giải pháp khác thay thế cho việc sử dụng batch như hiện tại, và đó chính là stream processing

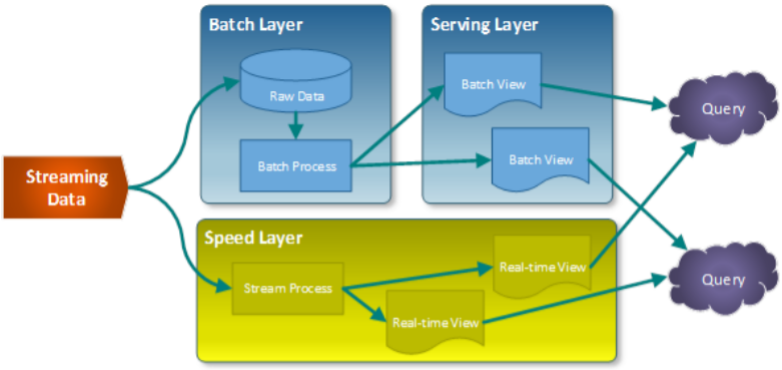


Hình 3: *Stream Processing*

3 Mô hình Lambda, Kappa & Delta

3.1 Lambda

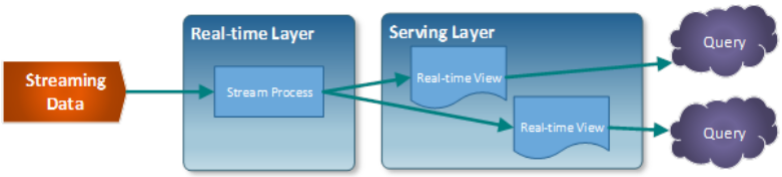
Mô hình Lambda là một kỹ thuật xử lý dữ liệu có khả năng xử lý lượng dữ liệu khổng lồ một cách hiệu quả. Hiệu quả của kiến trúc này trở nên rõ ràng dưới dạng tăng thông lượng, giảm độ trễ và lỗi không đáng kể.
Để xử lý nhiều sự kiện xảy ra trong hệ thống hoặc delta processing , kiến trúc Lambda cho phép xử lý dữ liệu bằng cách đưa vào ba lớp riêng biệt. Kiến trúc Lambda bao gồm Batch Layer, Speed Layer (còn được gọi là Stream layer) và Serving Layer.



Hình 4: *Mô hình Lambda*

3.2 Kappa

Kiến trúc Kappa được xem là giải pháp thay thế cho kiến trúc Lambda khi hiệu suất của Batch Layer là không cần thiết. Kiến trúc này cho phép sử dụng ít tài nguyên hơn và có khả năng hoạt động tốt hơn trong một số tình huống doanh nghiệp nhất định khi sử dụng mô hình Lambda có vẻ phức tạp. Như chúng ta thấy ở hình dưới, mô hình Kappa chỉ còn hai lớp: Real-time Layer và Serving Layer.

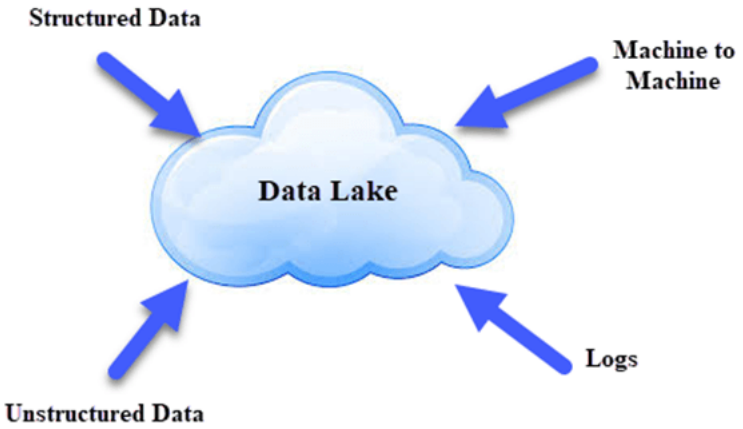


Hình 5: *Mô hình Kappa*

4 Công cụ Delta Lake

4.1 Data lake

Data lake là một kho lưu trữ có thể lưu trữ một lượng lớn dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc. Đây là nơi lưu trữ mọi loại dữ liệu ở định dạng gốc mà không có giới hạn cố định về kích thước tài khoản hoặc tệp. Nó cung cấp số lượng dữ liệu lớn để tăng hiệu suất phân tích và tích hợp gốc.



Hình 6: *Data lake*

4.2 Lakehouse

Lakehouse là một kiến trúc mới, mở, kết hợp các yếu tố tốt nhất của data lake và data warehouse. Lakehouse được kích hoạt bởi một thiết kế hệ thống mở và tiêu chuẩn hóa mới: triển khai các cấu trúc dữ liệu và tính năng quản lý dữ liệu tương tự với các cấu trúc dữ liệu trong data warehouse, trực tiếp trên loại lưu trữ chi phí thấp được sử dụng cho các data lake. Do đó ta có được một kho lưu trữ giá rẻ và có độ tin cậy cao.

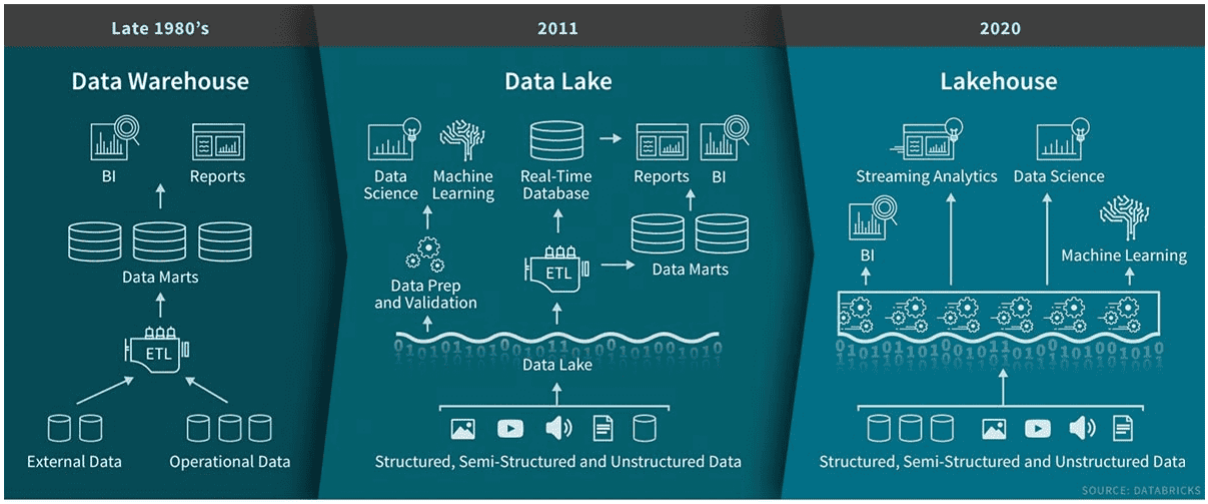
4.3 Delta Lake

4.3.1 Giới thiệu

Delta Lake là một dự án mã nguồn mở cho phép xây dựng một kiến trúc Lakehouse trên data lake. Delta Lake cung cấp các giao dịch ACID, xử lý siêu dữ liệu có thể mở rộng và thống nhất việc xử lý dữ liệu streaming và batch trên các data lake hiện có, chẳng hạn như S3, ADLS, GCS và HDFS.

Cụ thể, Delta Lake cung cấp:

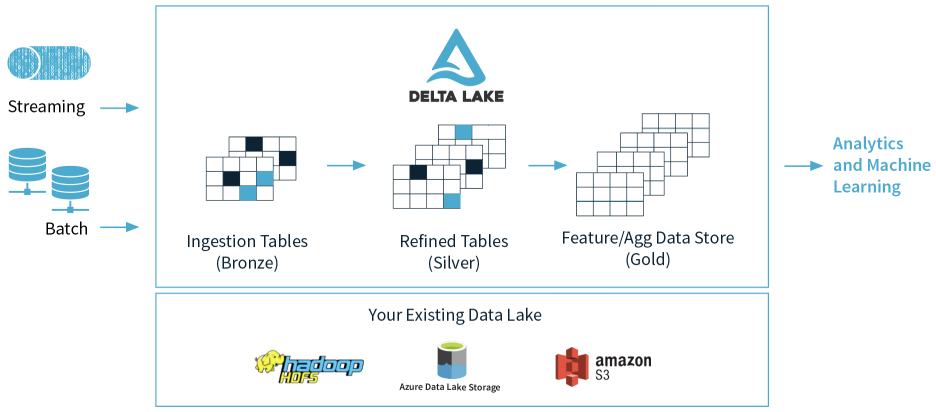
- Các giao dịch ACID trên Spark: Các mức độ cô lập tuần tự (Serializable) đảm bảo rằng người đọc không bao giờ thấy dữ liệu không nhất quán.
- Xử lý siêu dữ liệu có thể mở rộng: Tận dụng sức mạnh xử lý phân tán của Spark để xử lý tất cả siêu dữ liệu cho các bảng có quy mô petabyte với hàng tỷ tệp một cách dễ dàng.
- Thống nhất việc xử lý dữ liệu streaming và batch: Một bảng trong Delta Lake là một bảng batch cũng như một streaming source và sink. Nhập dữ liệu streaming, lấp đầy lịch sử batch, truy vấn tương tác, tất cả đều hoạt động hiệu quả.



Hình 7: *Lakehouse*

- Thực thi lược đồ: Tự động xử lý các biến thể lược đồ để ngăn chặn việc chèn các bản ghi không hợp lệ trong quá trình nhập dữ liệu.
- Du hành thời gian: Lập các phiên bản dữ liệu cho phép khôi phục, kiểm tra toàn bộ lịch sử và các thử nghiệm máy học có thể tái tạo.
- Thêm, cập nhật và xóa: Hỗ trợ các thao tác hợp nhất, cập nhật và xóa để cho phép các trường hợp sử dụng phức tạp như các thao tác thu thập dữ liệu thay đổi (change-data-capture), các thao tác kích thước thay đổi chậm (slowly-changing-dimension hay SCD), thêm và cập nhật streaming, v.v.

4.3.2 Kiến trúc tổng quan



Hình 8: *Delta Lake*

4.3.3 Cách cài đặt và một số thao tác cơ bản

4.3.3.a Cài đặt Apache Spark với Delta Lake

Có 2 cách để cài đặt như sau:

1. Chạy tương tác: Khởi động Spark shell (ta chọn Python) với Delta Lake và chạy các đoạn mã tương tác trong shell.

   - Cài đặt phiên bản PySpark tương thích với phiên bản Delta Lake bằng cách chạy như sau:

```
pip install pyspark==<compatible-spark-version>
```

   - Chạy PySpark với gói Delta Lake và các cấu hình bổ sung:

```
pyspark --packages io.delta:delta-core_2.12:1.0.0
    --conf "spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension"
    --conf "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
```

2. Chạy dưới dạng dự án: Thiết lập một dự án Python với Delta Lake, sao chép các đoạn mã vào tệp nguồn và chạy dự án.

   Để thiết lập một dự án Python (ví dụ: để unit testing), bạn có thể cài đặt Delta Lake bằng cách sử dụng pip install delta-spark và sau đó cấu hình SparkSession với hàm tiện ích config_spark_with_delta_pip() trong Delta Lake.

```
from delta import *

builder = pyspark.sql.SparkSession.builder.appName("MyApp") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog")

spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

4.3.3.b Tạo bảng mới

Để tạo bảng Delta, ta ghi DataFrame ra định dạng delta. Bạn có thể sử dụng mã Spark SQL hiện có và thay đổi định dạng từ parquet, csv, json, v.v. sang delta.

```
data = spark.range(0, 5)
data.write.format("delta").save("/tmp/delta-table")
```

4.3.3.c Đọc dữ liệu từ bảng

Bạn đọc dữ liệu trong bảng Delta của mình bằng cách chỉ định đường dẫn đến tệp: '/tmp/delta-table':

```
df = spark.read.format("delta").load("/tmp/delta-table")
df.show()
```

4.3.3.d Cập nhật dữ liệu trong bảng

Delta Lake hỗ trợ một số thao tác để sửa đổi bảng bằng cách sử dụng API DataFrame tiêu chuẩn. Ví dụ này chạy một batch job để ghi đè dữ liệu trong bảng:

```
data = spark.range(5, 10)
data.write.format("delta").mode("overwrite").save("/tmp/delta-table")
```

Nếu bạn đọc lại bảng này, bạn sẽ chỉ thấy các giá trị 5-9 bạn đã thêm vì bạn đã ghi đè dữ liệu trước đó.

4.3.3.e Ghi dòng dữ liệu vào bảng

Bạn cũng có thể ghi vào bảng Delta bằng Structured Streaming. Nhật ký giao dịch Delta Lake đảm bảo xử lý chính xác một lần, ngay cả khi có các dòng hoặc truy vấn hàng loạt khác chạy đồng thời với bảng. Theo mặc định, các dòng chạy ở chế độ nối thêm, chế độ này sẽ thêm các bản ghi mới vào bảng:

```
streamingDf = spark.readStream.format("rate").load()
stream = streamingDf.selectExpr("value as id").writeStream.format("delta")
                    .option("checkpointLocation", "/tmp/checkpoint").start("/tmp/delta-table")
```

4.3.3.f Đọc dòng thay đổi từ bảng

Trong khi dòng đang ghi vào bảng Delta, bạn cũng có thể đọc từ bảng đó dưới dạng stream source. Ví dụ: bạn có thể bắt đầu một truy vấn streaming khác in tất cả các thay đổi được thực hiện đối với bảng Delta.

```
stream2 = spark.readStream.format("delta").load("/tmp/delta-table").writeStream.format("console").start()
```

Tài liệu tham khảo

[1] Confluent. *Streaming Data - Overview*. URL: https://www.confluent.io/learn/data-streaming/. (accessed: 05.10.2021).

[2] Delta Lake. *Build Lakehouses with Delta Lake*. URL: https://delta.io/. (accessed: 05.10.2021).

[3] Iman Samizadeh. *A brief introduction to two data processing architectures — Lambda and Kappa for Big Data*. URL: https://towardsdatascience.com/a-brief-introduction-to-two-data-processing-architectures-lambda-and-kappa-for-big-data-4f35c28005bb. (accessed: 05.10.2021).

[4] David Taylor. *What is Data Lake?* URL: https://www.guru99.com/data-lake-architecture.html. (accessed: 05.10.2021).

[5] Ben Lorica & Michael Armbrust & Ali Ghodsi & Reynold Xin & Matei Zaharia. *What is a Lakehouse?* URL: https://databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html?_ga=2.108955522.1826215209.1633447809-2071492079.1629644828. (accessed: 05.10.2021).