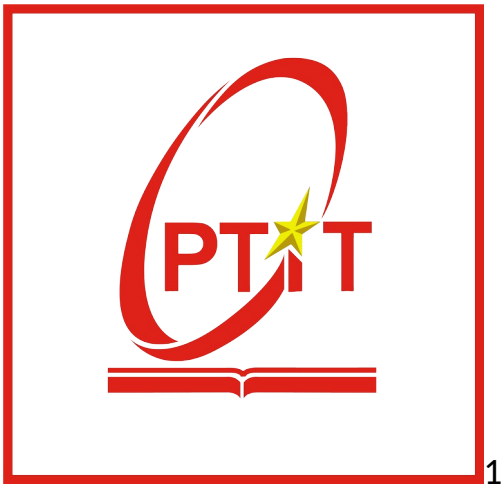
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



1

Báo cáo hàng tuần

Môn học: Thực tập cơ sở

Giảng viên: Kim Ngọc Bách

Họ và tên: Nguyễn Hữu Phúc

Mã SV: B22DCAT224

Lớp: E22CQCN04-B

Báo cáo tuần 2: Quy trình Pipeline mới trong Dự án Glamira UserFlow Insights

Ngày: 22/03/2025

Dự án: Glamira UserFlow Insights

I. Tổng quan và Lý do thay đổi quy trình Pipeline

Trong dự án **Glamira UserFlow Insights**, mục tiêu là xử lý và phân tích **32 GB** dữ liệu log người dùng từ trang web Glamira để xây dựng các mô hình machine learning (ML) dự đoán hành vi khách hàng. Ban đầu, quy trình pipeline được thiết kế để đưa dữ liệu trực tiếp từ tệp BSON vào Kafka, sau đó thực hiện ETL qua Spark và lưu trữ vào MongoDB. Tuy nhiên, em nhận thấy quy trình này không tốt, nên đã điều chỉnh thành: **Data -> MongoDB -> Kafka -> ETL -> SQL Server -> Machine Learning**. Dưới đây là lý do và lợi ích của quy trình mới:

1. MongoDB làm điểm khởi đầu

- **Linh hoạt**: MongoDB là cơ sở dữ liệu NoSQL không yêu cầu schema cố định, rất lý tưởng để nhập dữ liệu thô, không cấu trúc hoặc bán cấu trúc (như tệp BSON 32 GB). Điều này cho phép nhập dữ liệu mà không cần thiết kế schema trước.
- **Khả năng mở rộng**: MongoDB xử lý tốt các tập dữ liệu lớn (32 GB không phải là nhỏ) và hỗ trợ mở rộng ngang (horizontal scaling) nếu cần.
- **Tốc độ**: MongoDB có tốc độ ghi dữ liệu nhanh, phù hợp để nhập dữ liệu thô một cách nhanh chóng, đóng vai trò như một khu vực trung gian (data lake).
- **Lý do bắt đầu ở đây**: MongoDB hoạt động như một hệ thống kiểu "data lake", cho phép thu thập toàn bộ dữ liệu thô trước khi quyết định cách xử lý. Điều này đặc biệt hữu ích khi dữ liệu log người dùng có thể chứa các trường không đồng nhất.

2. Kafka để truyền dữ liệu

- **Tách biệt**: Kafka đóng vai trò trung gian, tách biệt giai đoạn nhập dữ liệu (MongoDB) khỏi các bước xử lý sau (ETL và SQL Server). Điều này tránh tắc nghẽn nếu hệ thống phía sau (SQL Server) chậm.
- **Xử lý thời gian thực**: Mặc dù 32 GB là một lần tải, Kafka rất phù hợp nếu dữ liệu log người dùng được cập nhật liên tục (ví dụ: hàng ngày hoặc hàng giờ). Nó hỗ trợ xử lý dữ liệu theo thời gian thực hoặc gần thời gian thực.
- **Độ tin cậy**: Kafka lưu trữ dữ liệu bền vững, đảm bảo không mất dữ liệu nếu ETL hoặc SQL Server gặp sự cố.
- **Lý do thêm bước này**: Kafka tăng tính chịu lỗi và hỗ trợ kiến trúc dữ liệu hiện đại, kết hợp giữa xử lý hàng loạt (batch) và luồng (streaming), phù hợp với các hệ thống phân tích dữ liệu lớn.

3. ETL vào SQL Server

- **Chuyển đổi**: Dữ liệu thô từ MongoDB (dạng JSON) thường cần được làm sạch, chuẩn hóa, hoặc tổng hợp trước khi sử dụng cho ML hoặc phân tích. Quá trình ETL (Extract, Transform, Load) đảm nhiệm việc này, sử dụng Pyspark để thực hiện
- **Lưu trữ có cấu trúc**: SQL Server vượt trội trong việc xử lý dữ liệu quan hệ với schema rõ ràng, rất phù hợp để thực hiện các truy vấn có cấu trúc hoặc nối dữ liệu—những yêu cầu thường thấy trong giai đoạn chuẩn bị dữ liệu cho ML.
- **Lý do dùng SQL Server**: Nhiều quy trình ML yêu cầu dữ liệu dạng bảng (như CSV), và SQL Server tích hợp tốt với các công cụ như Python (qua ODBC) hoặc Power BI cho các bước phân tích sau.

4. Machine Learning là mục tiêu cuối

- **Dữ liệu sẵn sàng**: Các mô hình ML yêu cầu dữ liệu sạch, có cấu trúc, và được kỹ thuật hóa đặc trưng. Dữ liệu từ SQL Server đáp ứng nhu cầu này.
- **Tích hợp**: SQL Server có thể cung cấp dữ liệu trực tiếp cho các framework ML (như pandas trong Python hoặc Azure ML) hoặc lưu trữ tập dữ liệu huấn luyện.
- **Lý do có bước này**: Giá trị cuối cùng của 32 GB dữ liệu log nằm ở việc tạo ra các hiểu biết hoặc dự đoán (ví dụ: dự đoán khả năng mua hàng của khách hàng trên Glamira). ML là bước tự nhiên để đạt được mục tiêu này.

II. Phân biệt JSON và BSON

Dữ liệu log người dùng trong dự án được lưu trữ dưới dạng BSON (trong tệp summary.bson) và cần chuyển đổi sang JSON để đẩy lên Kafka. Dưới đây là sự khác biệt giữa JSON và BSON:

1. Cấu trúc dữ liệu

- **JSON**:
    - Là định dạng văn bản (text-based), cấu trúc dạng cặp khóa-giá trị (key-value pairs).
    - Được thiết kế để dễ đọc và hiểu đối với con người.
    - Hỗ trợ các kiểu dữ liệu cơ bản: đối tượng, mảng, chuỗi, số, giá trị boolean.

- **BSON**:
    - Là định dạng nhị phân (binary-based), có cấu trúc tương tự JSON nhưng hỗ trợ nhiều kiểu dữ liệu phức tạp hơn.
    - Hỗ trợ các kiểu dữ liệu đặc biệt như ObjectId, Date, BinData (dữ liệu nhị phân), ngoài các kiểu cơ bản.
    - Không dễ đọc bằng mắt người vì là dữ liệu nhị phân.

2. Kích thước và hiệu suất

- **JSON**:
    - Kích thước nhẹ vì là văn bản thuần, nhưng không hỗ trợ các kiểu dữ liệu đặc biệt như BSON.
    - Đọc/ghi dữ liệu dạng văn bản tốn nhiều tài nguyên tính toán hơn so với dữ liệu nhị phân.
- **BSON**:
    - Kích thước lớn hơn JSON một chút do lưu trữ thêm metadata, nhưng cho phép truy vấn và thao tác dữ liệu nhanh hơn trong MongoDB.
    - Đọc/ghi nhanh hơn vì không cần chuyển đổi qua lại giữa văn bản và nhị phân.

3. Dễ đọc và sử dụng

- **JSON**:
    - Dễ đọc, dễ hiểu đối với con người.
    - Dễ sử dụng, được hỗ trợ bởi hầu hết các ngôn ngữ lập trình.
- **BSON**:
    - Khó đọc vì là dữ liệu nhị phân.
    - Chủ yếu dùng trong các ứng dụng cần tốc độ cao hoặc lưu trữ dữ liệu phức tạp (như MongoDB).

4. Ứng dụng phổ biến

- **JSON**:
    - Dùng để trao đổi dữ liệu giữa các ứng dụng web và API (HTTP, RESTful APIs).
- **BSON**:
    - Dùng trong cơ sở dữ liệu NoSQL như MongoDB, phù hợp với dữ liệu phức tạp và truy vấn hiệu suất cao.

5. Tính tương thích

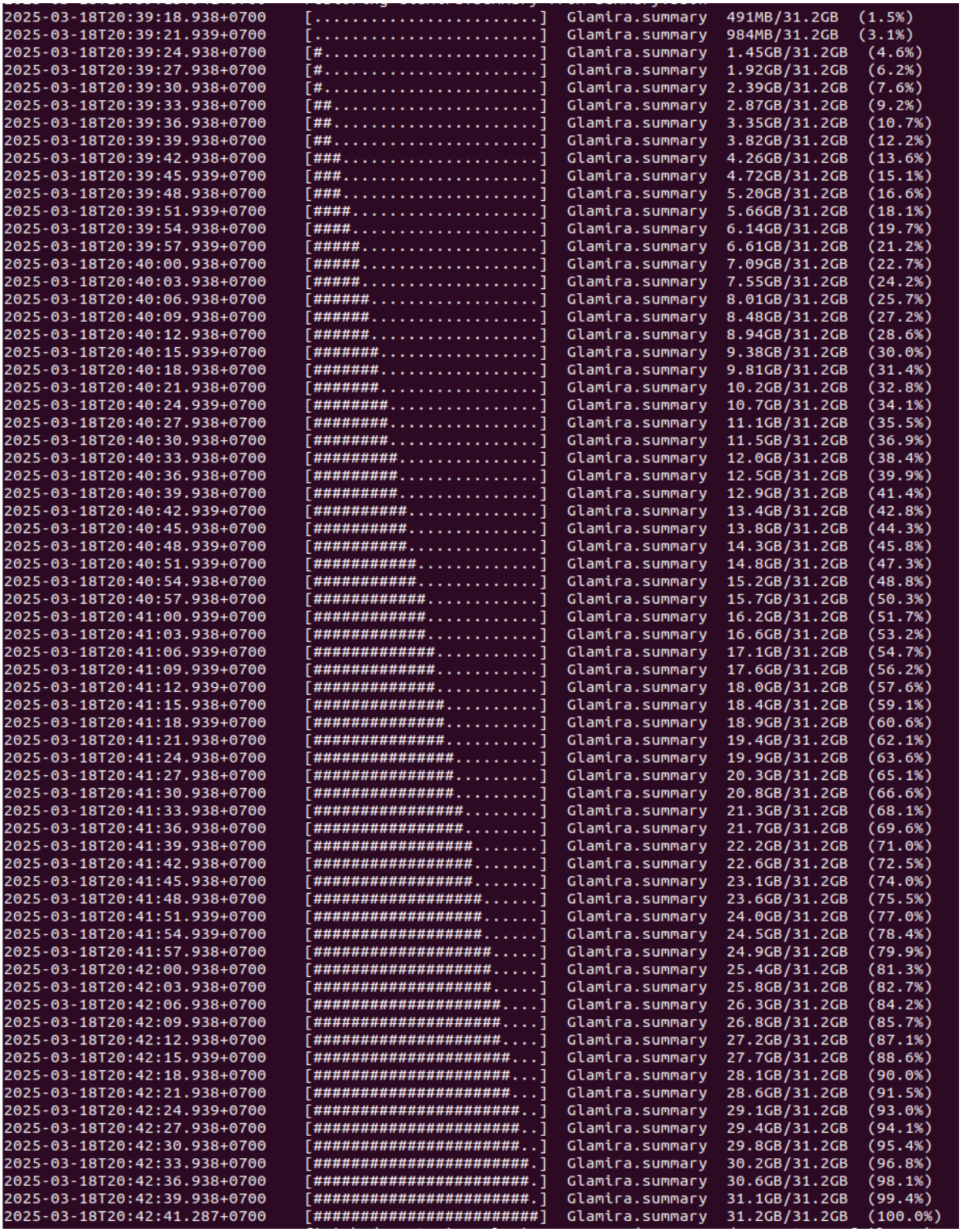
- **JSON**:
    - Tương thích với hầu hết các hệ thống và ngôn ngữ lập trình, dễ trao đổi giữa các hệ thống (ví dụ: REST APIs).
- **BSON**:
    - Hạn chế hơn, chủ yếu dùng trong MongoDB hoặc các ứng dụng tương thích với BSON.

Ứng dụng trong Glamira:

- **Tệp summary.bson (32 GB) được lưu trữ dưới dạng BSON vì nó chứa các kiểu dữ liệu đặc biệt của MongoDB (như ObjectId, Date).**
- **Để đẩy lên Kafka, cần chuyển sang JSON vì Kafka thường xử lý dữ liệu dạng văn bản (JSON) để đảm bảo tính tương thích với các hệ thống sau (Spark, SQL Server).**

III. Đưa dữ liệu BSON vào MongoDB

Mở terninal, sử dụng lệnh: mongorestore --db Glamira --collection summary /home/nguyenphuc/Documents/GlamiraUserFlowInsightsProject/glamira_ubl_oct2019_nov2019/dump/countly/summary.bson



```
2025-03-18T20:42:41.288+0700    finished restoring Glamira.summary (41432473 documents, 0 failures)
2025-03-18T20:42:41.289+0700    restoring indexes for collection Glamira.summary from metadata
2025-03-18T20:42:41.289+0700    index: &idx.IndexDocument{Options:primitive.M{"name":"time_stamp_-1", "ns":"c
2025-03-18T20:42:41.289+0700    index: &idx.IndexDocument{Options:primitive.M{"name":"device_id_1", "ns":"cou
2025-03-18T20:44:54.320+0700    41432473 document(s) restored successfully. 0 document(s) failed to restore.
```

.