ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
VNU University of Engineering and Technology

XÂY DỰNG HỆ THỐNG LOG ANALYSIS TRONG MÔI TRƯỜNG BIG DATA

Nguyễn Văn Việt - 23020444
Ngô Đinh Minh Nhật - 23020408
Kiều Đức Nam - 23020404

Tổng quan dự án

Hệ thống Log Analysis được xây dựng nhằm giải quyết nhu cầu thu thập, quản lý và phân tích dữ liệu log từ nhiều nguồn khác nhau theo thời gian thực.

Bối cảnh

- Trong bối cảnh cuộc cách mạng công nghiệp 4.0, dữ liệu lớn (Big Data) không chỉ bao gồm những tập dữ liệu có khối lượng khổng lồ, tốc độ tăng nhanh mà còn mang tính đa dạng và phức tạp, đòi hỏi các phương pháp xử lý hiện đại.
- Lượng log phát sinh mỗi ngày là vô cùng lớn, tạo ra thách thức trong việc lưu trữ, xử lý và khai thác giá trị thực tiễn từ nguồn dữ liệu này.

Mục tiêu dự án

- Xây dựng một pipeline xử lý dữ liệu log hoàn chỉnh, kết hợp xử lý theo lô (batch) và xử lý theo thời gian thực (streaming).
- Ứng dụng các mô hình học máy để phân loại log, phát hiện bất thường và hỗ trợ cảnh báo tự động..
- Tích hợp các công nghệ như Kafka, Spark, Hadoop HDFS, HBase.

Big Data

Big Data là gì

- Một thuật ngữ chỉ bộ dữ liệu lớn hoặc phức tạp mà các phương pháp truyền thống không đủ các ứng dụng để xử lý dữ liệu này. (Wikipedia)
- Những nguồn thông tin có đặc điểm chung khối lượng lớn, tốc độ nhanh và dữ liệu định dạng dưới nhiều hình thức khác nhau.(Gartner)

Nguồn gốc Big Data

- Dữ liệu hành chính (phát sinh từ chương trình của một tổ chức).
- Dữ liệu từ hoạt động thương mại (phát sinh từ các giao dịch giữa hai thực thể).
- Dữ liệu từ các thiết bị cảm biến.
- Dữ liệu từ các thiết bị theo dõi.
- Dữ liệu từ các hành vi.
- Dữ liệu từ các thông tin về ý kiến, quan điểm.

Mô hình 5V của Big Data

01

Volume (Khối lượng)

Là đặc điểm tiêu biểu nhất, kích cỡ của big data ngày càng tăng, sử dụng công nghệ đám mây mới có khả năng lưu trữ.

02

Velocity (Tốc độ)

Khối lượng dữ liệu gia tăng rất nhanh / Xử lý dữ liệu nhanh ở mức thời gian thực (real-time).

03

Variety (Đa dạng):

Đối với dữ liệu truyền thống chúng ta hay nói đến dữ liệu có cấu trúc. Ngày nay hơn 80% dữ liệu được sinh ra là phi cấu trúc.

04

Veracity (Độ tin cậy)

Bài toán phân tích và loại bỏ dữ liệu thiếu chính xác và nhiễu đang là tính chất quan trọng của bigdata.

05

Value (Giá trị)

Khi bắt đầu triển khai xây dựng dữ liệu lớn thì việc đầu tiên chúng ta cần phải làm đó là xác định được giá trị của thông tin mang lại.

LOG HDFS

Log HDFS là gì?

- Là nguồn thông tin quan trọng trong việc giám sát hệ thống, phân tích hiệu năng và phát hiện bất thường.
- Được sinh ra từ NameNode, DataNode, SecondaryNameNode, và ClientProtocol.

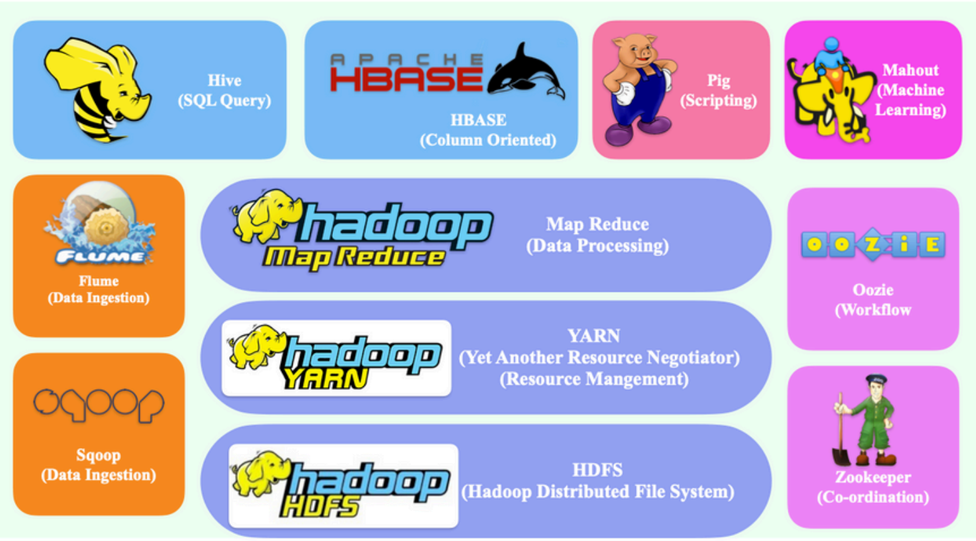
Đặc điểm log HDFS

- Sinh ra liên tục theo thời gian và theo từng sự kiện.
- Không có cấu trúc cố định mà tồn tại dưới dạng văn bản tự do.
- Đa dạng về loại sự kiện.
- Khối lượng rất lớn.

Cấu trúc điển hình

- Timestamp, Thread / Process, Level, Module, Message.

Hadoop



Hadoop là một nền tảng mã nguồn mở được thiết kế để lưu trữ và xử lý dữ liệu lớn theo mô hình phân tán.

Thành phần chính:

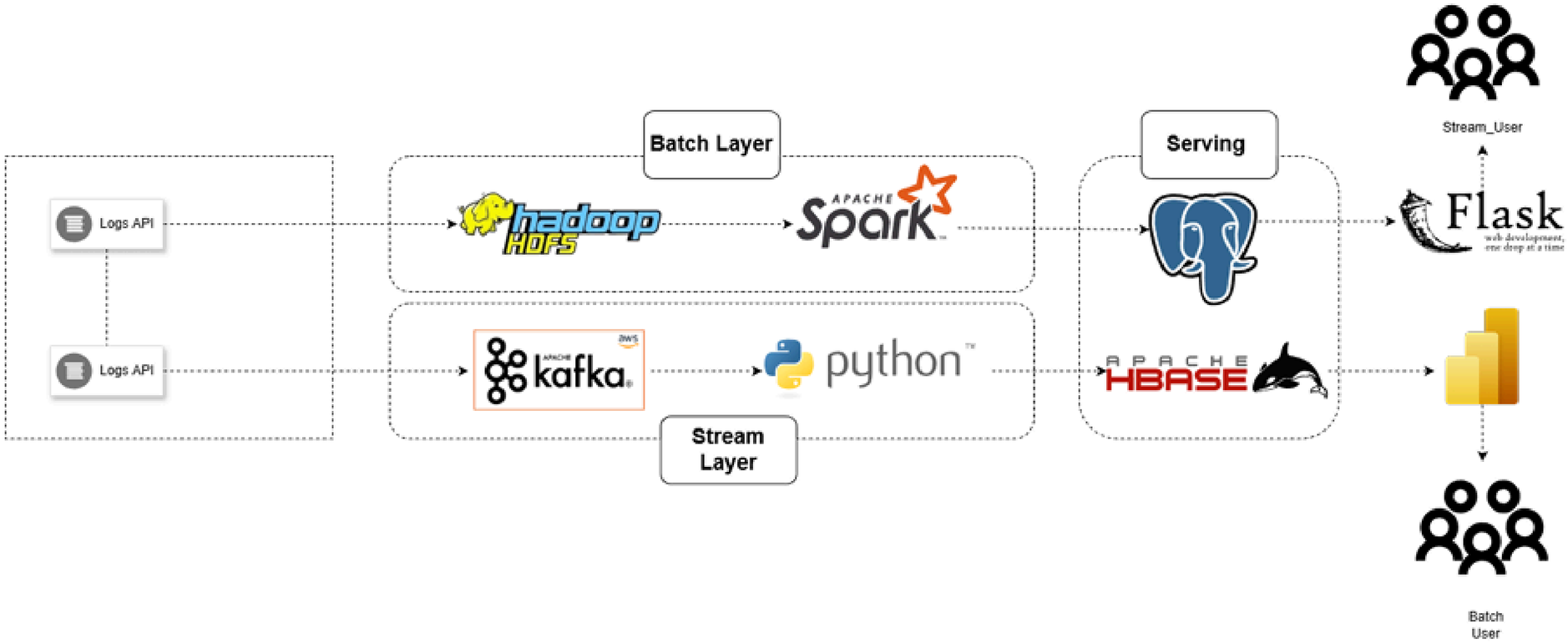
- HDFS (lưu trữ phân tán)
- YARN (quản lý tài nguyên)
- MapReduce (xử lý batch).

Hệ sinh thái mở rộng: Hỗ trợ xử lý batch/streaming, truy vấn, giám sát.

Công nghệ chính trong hệ thống

Phương pháp

Kiến trúc Lambda



Kiến trúc Lamda

Batch_layer

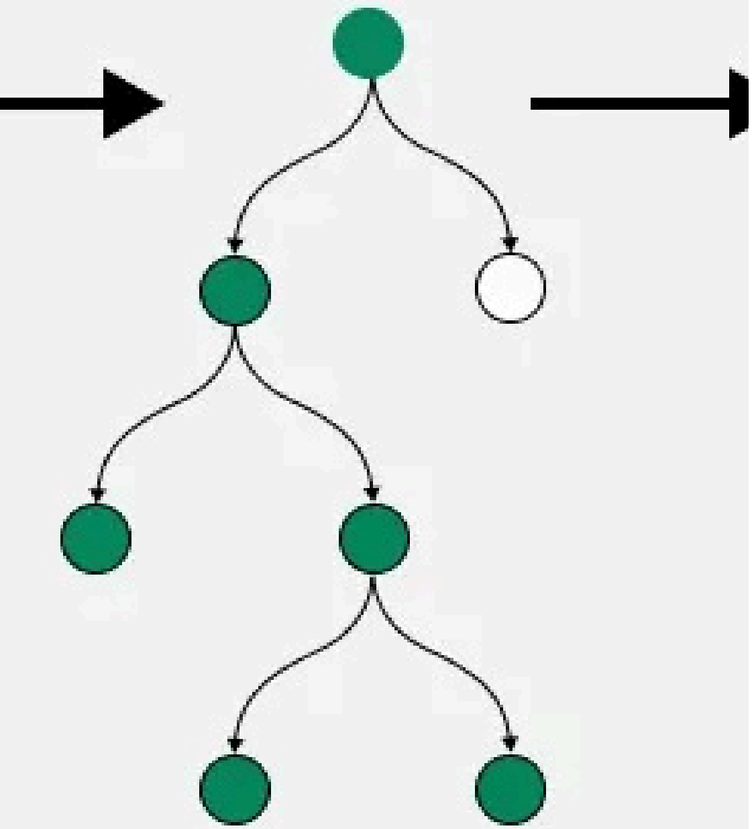
- Xử lý toàn bộ log lịch sử với quy mô lớn.
- Lưu trữ raw logs vào HDFS.
- Tiền xử lý bằng Spark: Làm sạch, chuẩn hóa, gom nhóm theo BlockId.
- Feature engineering: Số dòng log, thời gian xử lý block.
- Tạo batch views cho huấn luyện mô hình.
- Ưu điểm: Độ chính xác cao, nhưng không realtime.

Stream_layer

- Xử lý log mới sinh realtime.
- Kafka tiếp nhận log qua topic, Python Consumer xử lý tức thời.
- Trích xuất feature, gán nhãn sơ bộ.
- Lưu vào HBase cho truy vấn thấp trễ.
- Tích hợp anomaly detection cho cảnh báo.
- Phản hồi nhanh cho log mới.

Serving_layer

- Cung cấp dữ liệu cho ứng dụng và UI.
- HBase: Lưu realtime, phục vụ truy vấn cao tốc.
- PostgreSQL: Lưu tổng hợp batch, cho báo cáo.
- Flask: Xây dựng UI và API.
- Hợp nhất Batch và Speed cho dữ liệu nhất quán.



LightGBM

Giới thiệu

LightGBM
(Light Gradient Boosting Machine)

- Thuộc nhóm Gradient Boosting Decision Tree (GBDT)
- Dùng cho phân loại và hồi quy trên dữ liệu lớn
- Tối ưu tốc độ huấn luyện, bộ nhớ sử dụng, mở rộng tố

Cốt lõi thuật toán:

- Histogram-based Decision Tree Learning → giảm độ phức tạp O(n) → O(B)
- Leaf-wise Growth Strategy → giảm nhanh hàm mất mát, đạt độ chính xác cao với ít cây
- GOSS → giữ mẫu gradient lớn, giảm số lượng mẫu nhỏ
- EFB → gom đặc trưng loại trừ để giảm chiều dữ liệu

Thời gian

Nguyên lý & Ưu điểm

Nguyên lý GBDT

- Dự đoán tại vòng t:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

- $f_t(x_i)$ là cây mới huấn luyện từ gradient và Hessian của hàm mất mát
- Binary classification → Logistic Loss

Hàm mục tiêu & tối ưu

- $$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t)$$
- Leaf-wise: chọn lá cải thiện cao nhất
- Gain dùng để quyết định chia lá
- Tối ưu bằng Taylor bậc hai

Ưu điểm

- Giảm hàm mất mát nhanh, đạt độ chính xác cao với ít cây
- Xử lý dữ liệu lớn, thưa, nhiều đặc trưng hiệu quả
- Hỗ trợ class imbalance với scale_pos_weight
- Cung cấp feature importance, dễ giải thích mô hình

Thực nghiệm

01

Chuẩn bị dữ liệu

02

Tiền xử lý bằng Apache Spark

03

Huấn luyện model LightGBM

04

Xử lý dữ liệu realtime bằng Kafka, Python và HBase

05

Đánh giá mô hình

Chuẩn bị dữ liệu

```
+----------------------------------------------------------------------------------------------------------------------------------------------------------------+
|value                                                                                                                                                           |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------+
|081109 203518 143 INFO dfs.DataNode$DataXceiver: Receiving block blk_-1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010                 |
|081109 203518 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /mnt/hadoop/mapred/system/job_200811092030_0001/job.jar. blk_-1608999687919862906|
|081109 203519 143 INFO dfs.DataNode$DataXceiver: Receiving block blk_-1608999687919862906 src: /10.250.10.6:40524 dest: /10.250.10.6:50010                   |
|081109 203519 145 INFO dfs.DataNode$DataXceiver: Receiving block blk_-1608999687919862906 src: /10.250.14.224:42420 dest: /10.250.14.224:50010               |
|081109 203519 145 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-1608999687919862906 terminating                                           |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------+
only showing top 5 rows

+-------+-----------------------------------+
|EventId|EventTemplate                      |
+-------+-----------------------------------+
|E1     |[*]Adding an already existing block[*]|
|E2     |[*]Verification succeeded for[*]   |
|E3     |[*]Served block[*]to[*]            |
|E4     |[*]Got exception while serving[*]to[*]|
|E5     |[*]Receiving block[*]src:[*]dest:[*]|
+-------+-----------------------------------+
only showing top 5 rows

+----------------------+-------+
|BlockId               |Label  |
+----------------------+-------+
...
|blk_7854771516489510256 |Normal |
+----------------------+-------+
```

- Nguồn: Tập log HDFS_v1 từ Loghub, >11 triệu dòng từ NameNode, DataNode, DataXceiver, ClientProtocol.
- Lưu trữ: Đưa log thô vào HDFS cho batch.
- Tiền xử lý: Chuyển unstructured text thành structured cho phân tích và ML.

Tiền xử lý bằng Apache Spark

Đọc dữ liệu log và các bảng hỗ trợ từ HDFS

- Hệ thống khởi tạo SparkSession kết nối trực tiếp với cụm Spark và HDFS thông qua tham số spark.hadoop.fs.defaultFS. Ba nguồn dữ liệu được nạp:
  - File log thô HDFS (HDFS.log).
  - Bảng template sự kiện HDFS.log_templates.csv.
  - Bảng nhãn bất thường anomaly_label.csv.
- File log được đọc bằng spark.read.text dưới dạng một DataFrame đơn cột (value) chứa toàn bộ dòng log tương ứng với cấu trúc log của HDFS.

```
+--------------------------------------------------------------------------------------------------------------------------------------------------------------+
|value                                                                                                                                                         |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------+
|081109 203518 143 INFO dfs.DataNode$DataXceiver: Receiving block blk_-1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010            |
|081109 203518 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /mnt/hadoop/mapred/system/job_200811092030_0001/job.jar. blk_-1608999687919862906|
|081109 203519 143 INFO dfs.DataNode$DataXceiver: Receiving block blk_-1608999687919862906 src: /10.250.10.6:40524 dest: /10.250.10.6:50010               |
|081109 203519 145 INFO dfs.DataNode$DataXceiver: Receiving block blk_-1608999687919862906 src: /10.250.14.224:42420 dest: /10.250.14.224:50010            |
|081109 203519 145 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-1608999687919862906 terminating                                        |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------+
only showing top 5 rows

+-------+----------------------------------------+
|EventId|EventTemplate                           |
+-------+----------------------------------------+
|E1     |[*]Adding an already existing block[*]  |
|E2     |[*]Verification succeeded for[*]        |
|E3     |[*]Served block[*]to[*]                 |
|E4     |[*]Got exception while serving[*]to[*]  |
|E5     |[*]Receiving block[*]src:[*]dest:[*]    |
+-------+----------------------------------------+
only showing top 5 rows

+-----------------------+-------+
|BlockId                |Label  |
+-----------------------+-------+
...
|blk_7854771516489510256 |Normal |
+-----------------------+-------+
```

Trích xuất thông tin log bằng Regular Expressions

- Do log HDFS có định dạng không cố định theo từng dòng, Spark sử dụng tập hợp các mẫu regex để trích xuất các trường quan trọng bao gồm:
  - timestamp: thời gian dạng yyMMdd HHmmss.
  - pid: mã mã tiến trình sinh log.
  - level: mức độ log (INFO, WARN, ERROR, DEBUG).
  - component: module sinh log, ví dụ DataNode$DataXceiver.
  - message: nội dung thông điệp log.
  - BlockId: khóa nhận dạng block (blk_...).
- Các trường được tạo ra bằng hàm regexp_extract trực tiếp trên DataFrame ban đầu.
- Timestamp sau khi trích xuất được chuyển đổi về kiểu timestamp bằng hàm to_timestamp.
- Sau bước này, DataFrame chỉ giữ lại các cột quan trọng gồm: datetime, pid, level, component, message, BlockId.

```
+-----------------+---+-----+-----------------------+-------------------------------------------------------------------------------------------------------------+-----------------------+
|datetime         |pid|level|component              |message                                                                                                      |BlockId                |
+-----------------+---+-----+-----------------------+-------------------------------------------------------------------------------------------------------------+-----------------------+
|2008-11-09 20:35:18|143|INFO |dfs.DataNode$DataXceiver    |Receiving block blk_-1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010                |blk_-1608999687919862906|
|2008-11-09 20:35:18|35 |INFO |dfs.FSNamesystem            |BLOCK* NameSystem.allocateBlock: /mnt/hadoop/mapred/system/job_200811092030_0001/job.jar. blk_-1608999687919862906|blk_-1608999687919862906|
|2008-11-09 20:35:19|143|INFO |dfs.DataNode$DataXceiver    |Receiving block blk_-1608999687919862906 src: /10.250.10.6:40524 dest: /10.250.10.6:50010                   |blk_-1608999687919862906|
|2008-11-09 20:35:19|145|INFO |dfs.DataNode$DataXceiver    |Receiving block blk_-1608999687919862906 src: /10.250.14.224:42420 dest: /10.250.14.224:50010                |blk_-1608999687919862906|
|2008-11-09 20:35:19|145|INFO |dfs.DataNode$PacketResponder|PacketResponder 1 for block blk_-1608999687919862906 terminating                                             |blk_-1608999687919862906|
+-----------------+---+-----+-----------------------+-------------------------------------------------------------------------------------------------------------+-----------------------+
only showing top 5 rows
```

Tiền xử lý bằng Apache Spark

Gán EventId theo bảng template

- Chuyển mỗi EventTemplate thành mẫu regex để kiểm tra khớp.
- Duyệt qua từng template và kiểm tra dòng log (sau khi trích xuất) có khớp với mẫu regex hay không.
- Nếu khớp, gán EventId tương ứng cho dòng log bằng hàm when().otherwise() trong Spark, tạo ra cột EventId thể hiện kiểu sự kiện cho từng dòng log.
- Các dòng log không khớp với bất kỳ template nào sẽ được loại bỏ ở bước sau để đảm bảo dữ liệu sạch.

Tạo chuỗi Event Sequence cho từng BlockId

- Lọc các dòng log đã có EventId (loại bỏ những dòng không khớp template).
- Gom nhóm (group by) theo BlockId và sử dụng hàm collect_list(EventId) để tạo danh sách các EventId theo thứ tự thời gian (dựa trên timestamp).
- Kết quả là mỗi BlockId sẽ có một chuỗi sự kiện đại diện cho trình tự hoạt động từ bắt đầu đến kết thúc.

Biến đổi dãy sự kiện thành đặc trưng số (Feature Encoding)

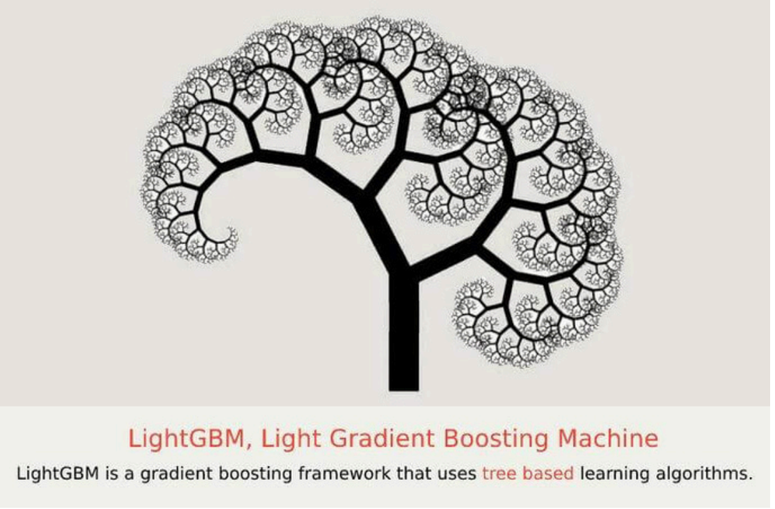
- Đếm tần suất xuất hiện của từng loại EventId (từ E1 đến E29, dựa trên bảng template có 29 sự kiện) trong chuỗi của mỗi BlockId.
- Biến đổi kết quả đếm thành vector đặc trưng, nơi mỗi chiều đại diện cho tần suất của một EventId cụ thể.
- Vector này sẽ được sử dụng làm đầu vào cho mô hình LightGBM để phân loại bất thường. Kết quả sau encoding được lưu vào HDFS và PostgreSQL để huấn luyện và phân tích.

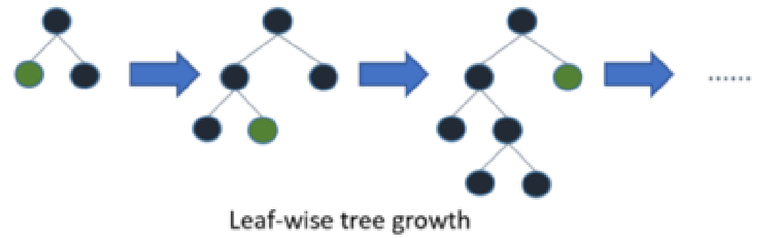
Tiền xử lý bằng Apache Spark

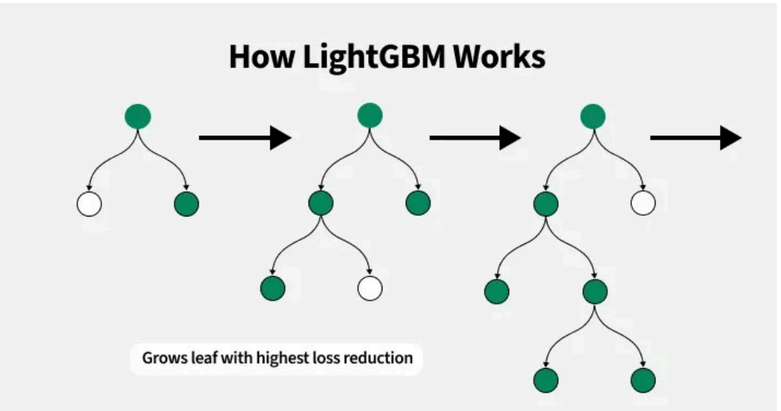
Kết quả

BlockId	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	E21	E22	E23	E24	E25	E26	E27	E28	E29
blk_-1001138135617662562	0	0	8	4	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1005590426013980840	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-102130907746974051	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1024067452279512003	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0	0
blk_-1027487181718073118	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0	0
blk_-1028418231551336995	0	1	1	2	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1029280765014954488	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0	0
blk_-1036632738441638662	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1040412692236014953	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1041728781629481437	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1043804223311329266	0	1	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1046696409333586608	0	0	4	2	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-10504368397740793	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-106108071417019486	0	0	1	2	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1061729600448953014	0	0	1	2	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1066159766740796836	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-108380212683047118	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-1089336313406296579	0	0	1	2	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0
blk_-109386570129652225	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0	0
blk_-1096621677333075077	0	0	0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	3	1	3	0	0	3	0	0	0

- Đặc trưng gồm: chuỗi sự kiện, tần suất EventId, thời gian hoạt động và thông tin log liên quan.
- Dữ liệu lưu trên HDFS + PostgreSQL để phục vụ huấn luyện mô hình và phân tích.
- Bộ đặc trưng được tích hợp vào pipeline thời gian thực, đảm bảo phát hiện bất thường liên tục và nhất quán.







Huấn luyện mô hình LightGBM

Chia dữ liệu và Xử lý mất cân bằng

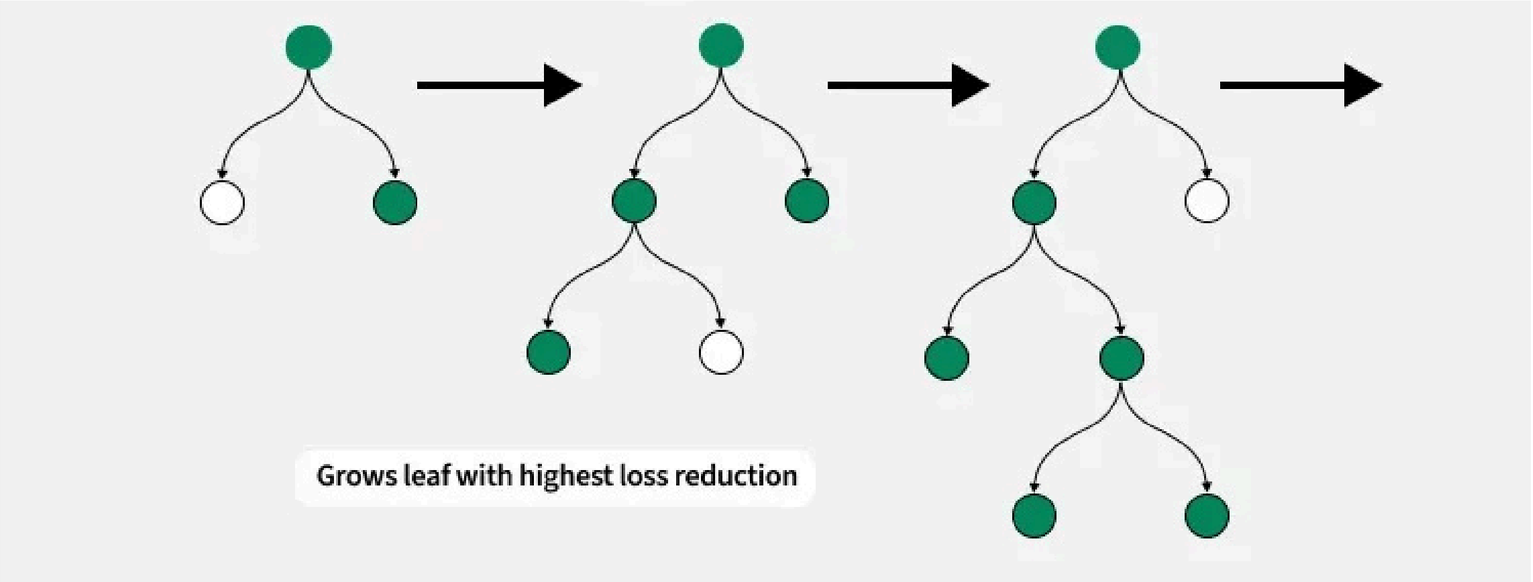
Chia dữ liệu có stratify

- Dữ liệu được chia theo đúng tỉ lệ nhãn:
  - Train: 60%
  - Validation: 20%
  - Test: 20%
- Validation dùng để:
  - Tối ưu tham số
  - Kích hoạt early stopping
- Test giữ nguyên để đánh giá khách quan sau huấn luyện.

Mục tiêu dài hạn

- Tỉ lệ bất thường thấp → mô hình dễ bỏ sót.
- scale_pos_weight = 33 giúp:
  - Mô hình chú trọng hơn vào mẫu bất thường.
  - Tăng khả năng phát hiện true anomalies.

Huấn luyện mô hình LightGBM



- Sử dụng bộ đặc trưng sau tiền xử lý để dự đoán block bất thường.
- Chọn LightGBM vì:
  - Huấn luyện nhanh, phù hợp dữ liệu lớn.
  - Xử lý tốt mất cân bằng nhãn trong log HDFS.
- Tỉ lệ nhãn mất cân bằng nghiêm trọng (~1:33), lớp bất thường rất hiếm.
- Dùng tham số scale_pos_weight = 33 để tăng trọng số mẫu hiếm → cải thiện phát hiện bất thường.

Huấn luyện mô hình LightGBM

Cấu hình mô hình & Đánh giá

Cấu hình LightGBM

- n_estimators = 3000 + early_stopping (100 rounds)
- Các siêu tham số chính:
- num_leaves = 63, max_depth = 6 → tránh overfitting
- subsample = 0.8, colsample_bytree = 0.8 → tăng khả năng tổng quát
- reg_lambda = 1.0 → regularization L2
- scale_pos_weight = 33 → xử lý class imbalance

Chỉ số đánh giá

- AUC – đo khả năng phân tách hai lớp.
- Average Precision (AP) – phù hợp cho dữ liệu mất cân bằng.
- Early stopping đảm bảo:
- Dừng tại thời điểm mô hình tốt nhất
- Tránh overfitting validation

Đánh giá mô hình

- **Mô hình phân tách gần như tuyệt đối hai lớp.**
- **Chỉ 16 lỗi trên hơn 115.000 mẫu → hiệu năng cực cao và rất ổn định.**
- **LightGBM xử lý tốt dữ liệu mất cân bằng mạnh.**

Kết quả trên tập validation

Chỉ số	Giá trị
ROC-AUC	0.9999543
PR-AUC	0.9980548
Ngưỡng dự đoán tối ưu	0.83
F1-score tại ngưỡng tối ưu	0.99423
Precision	0.99028
Recall	0.99822

Đánh giá mô hình

Kết quả trên tập test

Chỉ số	Giá trị
ROC-AUC	0.9999976
PR-AUC	0.9999125

Confusion Matrix trên tập test

	Dự đoán 0	Dự đoán 1
Thực tế 0	111637	8
Thực tế 1	8	3360

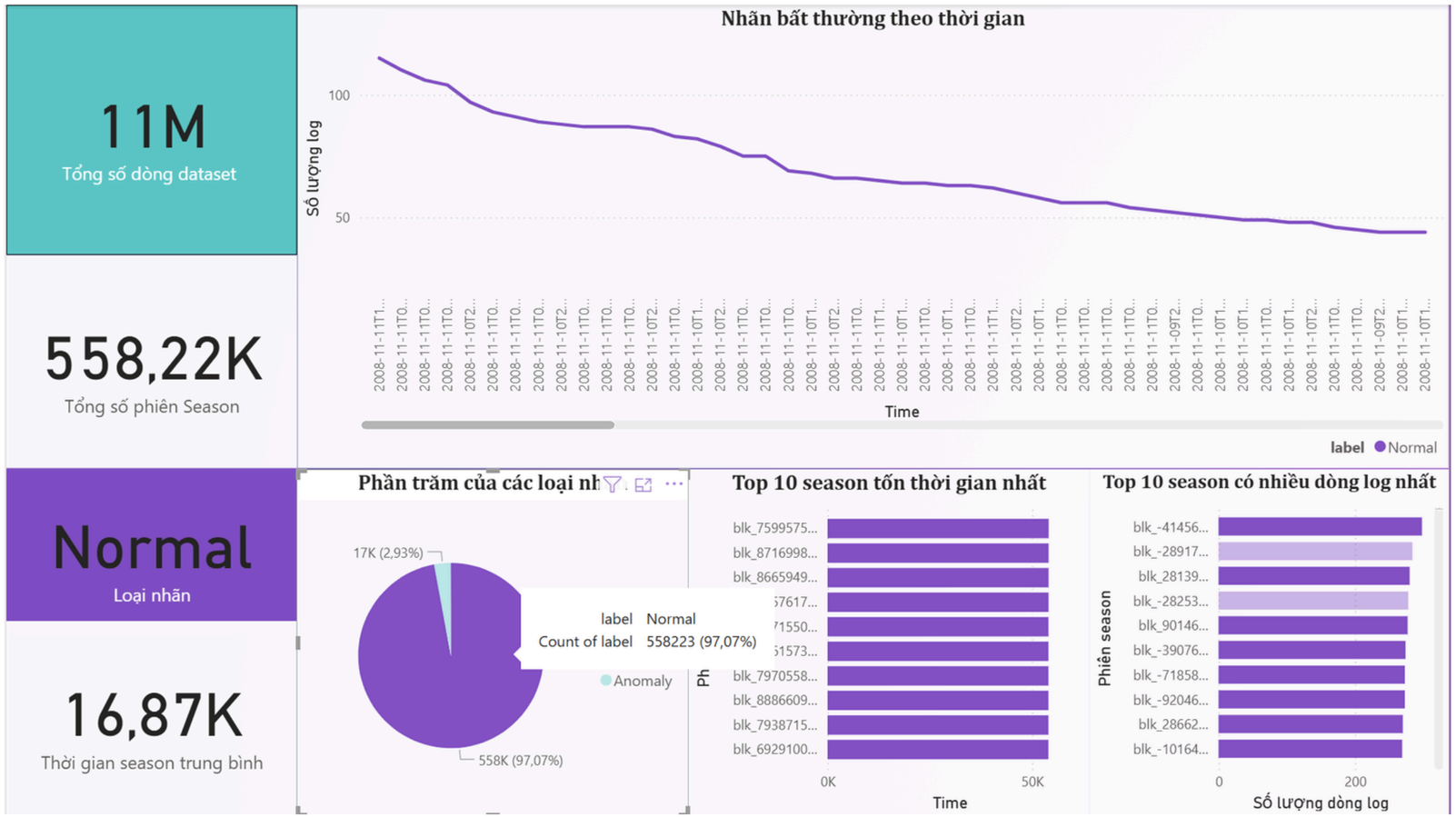
Báo cáo phân loại trên tập test

Lớp	Precision	Recall	F1-score	Support
0	0.9999	0.9999	0.9999	111645
1	0.9976	0.9976	0.9976	3368
Accuracy		0.9999		
Macro avg	0.9988	0,9988	0.9988	115013
Weighted avg	0.9999	0.9999	0.9999	115013

- **Accuracy toàn mô hình: 0.9999**
- **Nhận xét: Mô hình phân loại cực chính xác, gần như không bỏ sót block bất thường, tổng quát hóa tốt trên dữ liệu chưa từng thấy.**

Đánh giá kết quả thực nghiệm

- Mô hình hiệu quả, pipeline ổn định.
- Dashboard PowerBI: 11M log, 575K session, 2.93% anomaly.
- Xu hướng bất thường giảm dần.



Xử lý dữ liệu realtime bằng Kafka – Python – HBase

Xây dựng pipeline realtime mô phỏng log sinh liên tục từ HDFS, chạy song song với pipeline batch Spark.

Thành phần chính

- **Kafka:** truyền log realtime qua topic log_stream_topic
- **Python Producer/Consumer:** sinh log, xử lý và dự đoán bất thường
- **HBase:** lưu dữ liệu realtime để giao diện Flask truy vấn ngay lập tức

Mục tiêu:

- Tái tạo luồng log thật của HDFS
- Trích xuất đặc trưng và dự đoán bất thường theo thời gian thực
- Đảm bảo hệ thống hoạt động liên tục và ổn định

Xử lý dữ liệu realtime bằng Kafka – Python – HBase

Sinh dữ liệu realtime

- Dùng test_stream_data.csv làm danh sách BlockId cần mô phỏng.
- Hàm generate_real_time_data() lọc từ logs_full_labeled.csv để tạo DataFrame chứa:
- BlockId, start_ts, end_ts, duration_sec
- log_full (nội dung log), num_lines
- Hàm choose_random_log_full() chọn ngẫu nhiên 1 block để tạo log realtime.

Khởi chạy Kafka

- wait_for_kafka() kiểm tra Kafka sẵn sàng.
- Tạo 2 luồng:
- Producer thread: gửi 5 bản ghi ngẫu nhiên lên topic.
- Consumer thread: chạy nền để đọc liên tục.
- Dừng an toàn bằng STOP Event và SIGINT/SIGTERM.

Gửi dữ liệu lên Kafka

- Producer cấu hình:
  - acks='all' → không mất message
  - retries=5 → retry khi Kafka bận
  - value_serializer → JSON
- Mỗi message gồm: log_full, thời gian hoạt động, số dòng log, BlockId, thống kê.
- Gửi mỗi 10 giây để mô phỏng log thật.

Xử lý dữ liệu realtime bằng Kafka – Python – HBase

Consumer xử lý log realtime

- Nhận message JSON → giải mã thành dict Python.
- Hàm transformation() thực hiện:

1. Tách log bằng regex: timestamp, level, component, message, BlockId
2. Gán EventId bằng HDFS.log_templates.csv
3. Gom nhóm theo BlockId → tạo EventSequence
4. Đếm 29 EventId (E1–E29) → tạo vector đặc trưng
5. Gọi predict_block() để dự đoán bất thường

Kết quả transformation

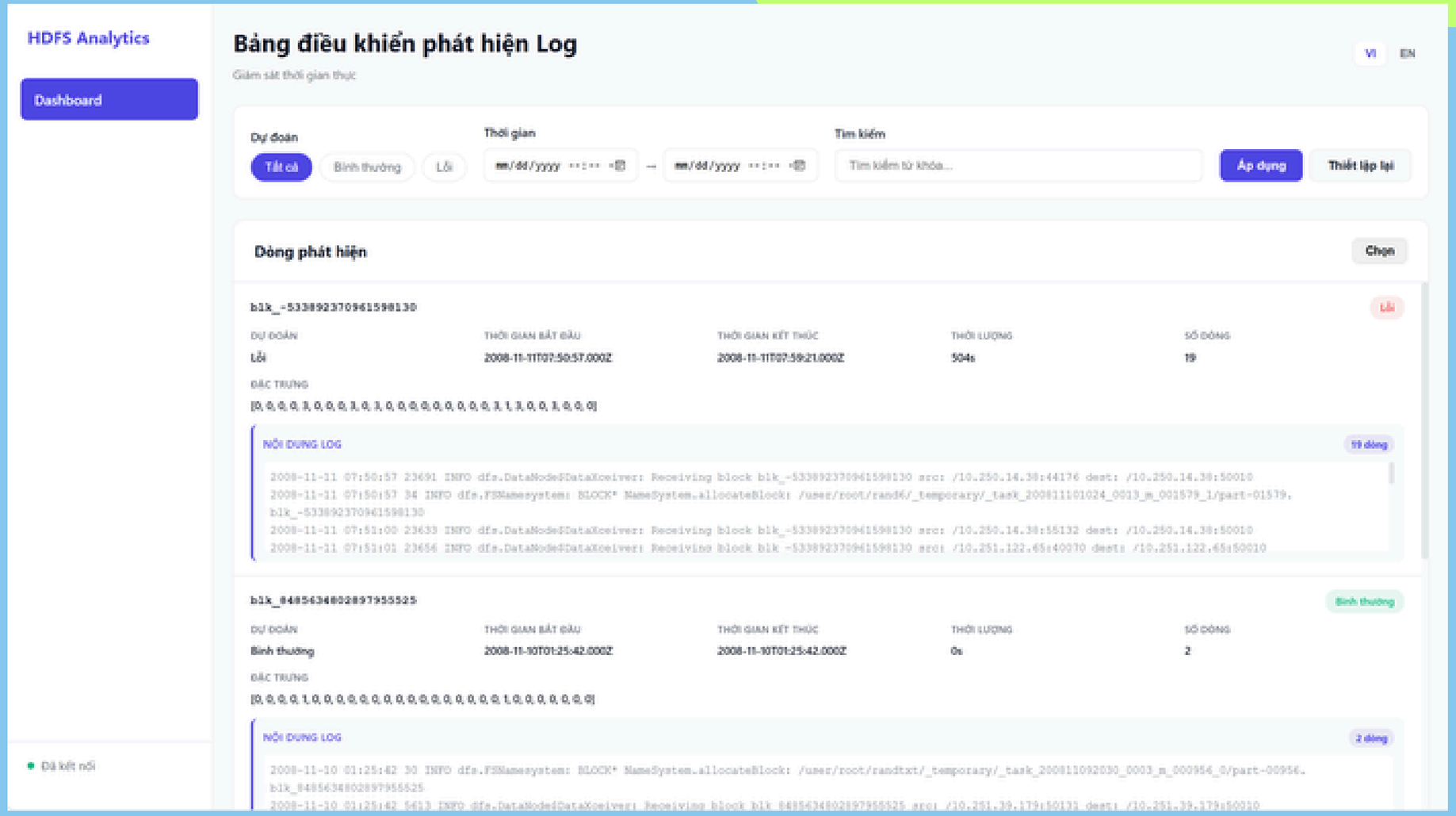
Trả về dict: thông tin block, đặc trưng, log đầy đủ, nhãn dự đoán.

Ghi vào HBase

- insert_dataHbase() lưu vào bảng log_stream_data:
- Key: BlockId
- Các cột: start_ts, end_ts, duration_sec, log_full, num_lines
- features (E1..E29), prediction, ts_ms
- Flask truy vấn HBase để hiển thị log và kết quả dự đoán ngay khi Consumer xử lý xong.

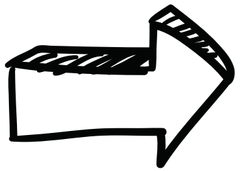
Xử lý dữ liệu realtime bằng Kafka – Python – HBase

Kết quả



Tổng kết quá trình thực nghiệm

- Batch Processing (Spark): trích xuất >11 triệu dòng log, chuẩn hóa dữ liệu, ánh xạ EventId, xây dựng bảng đặc trưng ML, khả năng mở rộng tốt.
- Realtime Processing (Kafka + HBase): tiếp nhận và lưu log realtime, phản hồi nhanh, giám sát block mới tức thời.
- Mô hình LightGBM: ROC-AUC & PR-AUC gần tuyệt đối, số lỗi dự đoán rất thấp, tổng quát hóa tốt ngay cả với dữ liệu mất cân bằng mạnh.
- Trực quan hóa Power BI: phân bố nhãn, xu hướng bất thường, block hoạt động dài/log nhiều, hỗ trợ phân tích nguyên nhân gốc.



Kiến trúc Lambda hoạt động hiệu quả
Pipeline batch + realtime + ML ổn định
Sẵn sàng mở rộng và triển khai thực tế
Nền tảng cho tối ưu mô hình và phân tích chuyên sâu.

Xin cảm ơn!