Smartphone Price Prediction in Big Data Environment

Duong Minh Duc
Dang Van Khai
Nguyen Viet Bac

Nội dung:

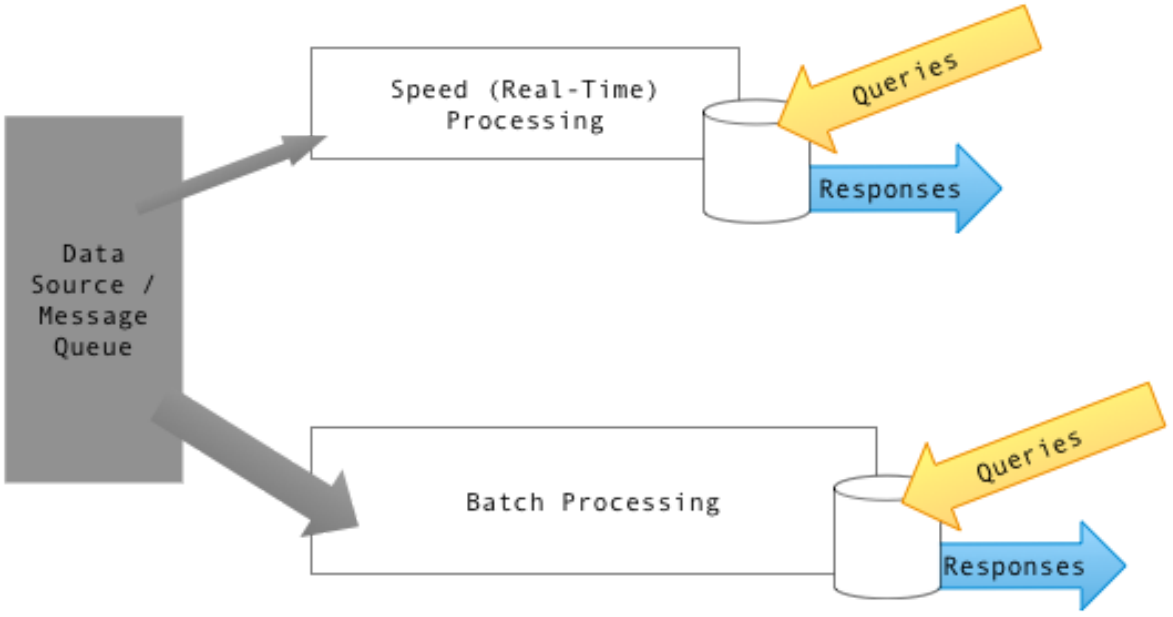
1. Cấu trúc Lambda và các công cụ liên quan
2. Ứng dụng trong Smartphone Price Prediction

Unit 1.

Kiến trúc Lambda và các công cụ liên quan

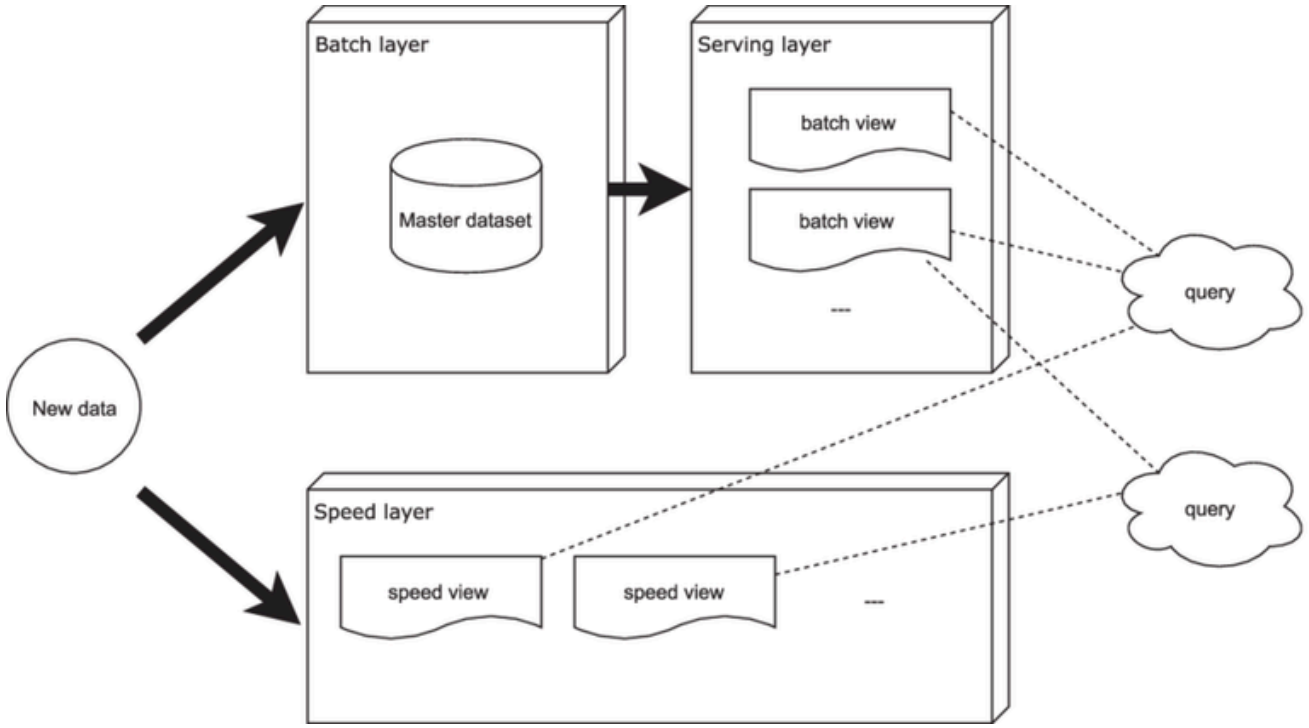
Kiến trúc Lambda là gì?

- Kiến trúc Lambda là kiến trúc xử lý dữ liệu được thiết kế để xử lý lượng dữ liệu khổng lồ bằng cách tận dụng cả phương pháp xử lý hàng loạt (batch) và xử lý luồng (stream). Kiến trúc Lambda cân bằng độ trễ, thông lượng và khả năng chịu lỗi bằng cách áp dụng xử lý hàng loạt (batch) để cung cấp chế độ xem toàn diện và chính xác cho dữ liệu, đồng thời áp dụng xử lý luồng thời gian thực (stream) để cung cấp dữ liệu thời gian thực.



Các lớp trong kiến trúc Lambda

- Lớp Xử Lý Theo Lô (Batch Layer)
- Lớp Xử Lý Luồng (Speed Layer)
- Lớp Phục Vụ (Serving Layer)



Batch Layer

- Lớp xử lý batch tính toán trước kết quả bằng cách sử dụng hệ thống xử lý phân tán có thể xử lý lượng dữ liệu rất lớn, hướng đến độ chính xác bằng cách xử lý tất cả dữ liệu. Điều này có nghĩa là ta có thể sửa bất kỳ lỗi nào bằng cách tính toán lại dựa trên tập dữ liệu hoàn chỉnh. Kết quả thường được lưu trữ trong cơ sở dữ liệu chỉ đọc, và thay thế hoàn toàn các dữ liệu tính toán trước đó.

- Một số hệ thống xử lý dữ liệu batch có thể kể đến như Apache Hadoop, nổi tiếng với việc phổ biến phương pháp map-reduce. Hoặc một số dịch vụ cung cấp giải pháp tổng hợp như Snowflake, Redshift, Synapse và Big Query.

Speed Layer

- Lớp Speed xử lý các luồng dữ liệu trong thời gian thực và không có yêu cầu sửa chữa hoặc hoàn thiện. Lớp này hy sinh thông lượng vì nó nhằm mục đích giảm thiểu độ trễ bằng cách xử lý dữ liệu trong thời gian thực. Về cơ bản, lớp tốc độ chịu trách nhiệm lấp đầy 'khoảng trống' do độ trễ của lớp batch. Dữ liệu của lớp speed có thể không chính xác hoặc đầy đủ như chế độ xem cuối cùng do lớp batch tạo ra, nhưng chúng có sẵn gần như ngay lập tức sau khi nhận được dữ liệu và có thể được thay thế khi bởi dữ liệu được xử lý do lớp batch.

- Thường hay bị nhầm lẫn với micro-batch, lớp speed hoàn toàn xử lý dữ liệu theo từng dòng dữ liệu.

- Các công nghệ xử lý luồng thường được sử dụng là Apache Kafka, Amazon Kinesis, Apache Storm, SQLstream, Apache Samza, Apache Spark, Azure Stream Analytics. Dữ liệu sau xử lý thường được lưu trữ trên cơ sở dữ liệu NoSQL phục vụ nhu cầu phân tích nhanh.

Serving Layer

- Dữ liệu được xử lý từ các lớp batch và speed được lưu trữ trong lớp serving, đáp ứng các truy vấn bằng cách trả về dữ liệu được tính toán trước hoặc dữ liệu từ kho dữ liệu (data warehouse).
- Để triển khai lớp serving, ta có thể dùng các giải pháp như Apache Cassandra, Apache HBase, Azure Cosmos DB, MongoDB, VoltDB hoặc Elasticsearch cho dữ liệu của lớp Speed và Elephant DB, Apache Impala, SAP HANA hoặc Apache Hive cho dữ liệu của lớp Batch.
- Để cung cấp một điểm duy nhất (end-point) để có thể để tổng hợp dữ liệu đầu ra từ cả hai lớp, ta có thể áp dụng một số giải pháp như là ảo hoá dữ liệu Dremio hoặc hệ thống truy vấn phân tán như Trino, Presto.
- Một số giải pháp cơ sở dữ liệu thời gian thực cũng có thể dùng như một cơ sở dữ liệu tập trung cho cả dữ liệu batch và speed như Druid, SingleStore.

Các công cụ sử dụng trong kiến trúc Lambda

- Apache Hadoop: Dùng để lưu trữ dữ liệu và tạo các cụm phân tán.
- Hadoop Distributed File System (HDFS): Quản lý dữ liệu không thay đổi trong lớp xử lý lô.
- Apache Spark: Xử lý dữ liệu luồng, xử lý đồ thị và xử lý dữ liệu theo lô.
- Apache Cassandra: Lưu trữ các lượt xem thời gian thực.
- Apache Kafka: Luồng dữ liệu trong lớp xử lý nhanh.
- Apache Storm: Thực hiện các tác vụ trong lớp xử lý nhanh.
- Apache HBASE: Thực hiện các tác vụ trong lớp phục vụ.

Ưu điểm của kiến trúc Lambda

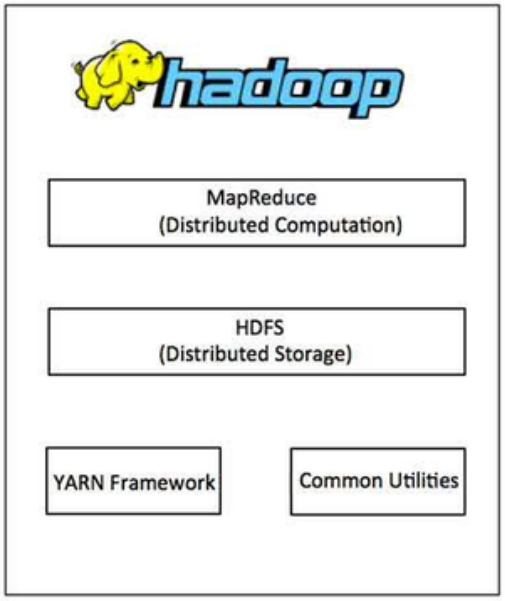
- Cân bằng tốt giữa tốc độ, độ tin cậy và khả năng mở rộng.
- Lớp xử lý lô quản lý dữ liệu lịch sử với lưu trữ phân tán chịu lỗi, đảm bảo ít khả năng xảy ra lỗi ngay cả khi hệ thống gặp sự cố.
- Lớp xử lý luồng quản lý dữ liệu thời gian thực với phản hồi ngay lập tức nhưng với độ chính xác thấp hơn.
- Truy cập vào cả dữ liệu thời gian thực và dữ liệu ngoại tuyến giúp bao quát nhiều kịch bản phân tích dữ liệu.

Nhược Điểm Của Kiến Trúc Lambda

- Kiến trúc phức tạp với nhiều lớp liên quan.
- Cần duy trì mã nguồn khác nhau cho lớp xử lý lô và lớp xử lý luồng, dẫn đến sự trùng lặp và yêu cầu đồng bộ hóa.
- Hiệu suất hệ thống giảm do tính toán nhiều lần trong mỗi chu kỳ lô, yêu cầu nhiều tài nguyên hơn.
- Mô hình dữ liệu khó di chuyển hoặc tổ chức lại.

Apache Hadoop

- Hadoop là một framework nguồn mở viết bằng Java cho phép phát triển các ứng dụng phân tán có cường độ dữ liệu lớn một cách miễn phí. Nó cho phép các ứng dụng có thể làm việc với hàng ngàn node khác nhau và hàng petabyte dữ liệu. Hadoop lấy được phát triển dựa trên ý tưởng từ các công bố của Google về mô hình MapReduce và hệ thống file phân tán Google File System (GFS).



Apache Hadoop

Hadoop YARN

- Phần này được dùng như một framework. Nó hỗ trợ hoạt động quản lý thư viện tài nguyên của các cluster và thực hiện chạy phân tích tiến trình.

Hadoop Distributed File System (HDFS)

- Một trong những vấn đề lớn nhất của các hệ thống phân tích Big Data là quá tải. Không phải hệ thống nào cũng đủ khỏe để có thể tiếp nhận một lượng thông tin khổng lồ như vậy. Chính vì thế, nhiệm vụ của Hadoop Distributed File System là phân tán cung cấp truy cập thông lượng cao giúp cho ứng dụng chủ. Cụ thể, khi HDFS nhận được một tệp tin, nó sẽ tự động chia file đó ra thành nhiều phần nhỏ. Các mảnh nhỏ này được nhân lên nhiều lần và chia ra lưu trữ tại các máy chủ khác nhau để phân tán sức nặng mà dữ liệu tạo nên
- HDFS sử dụng cấu trúc master node và worker/slave node. Trong khi master node quản lý các file metadata thì worker/slave node chịu trách nhiệm lưu trữ dữ liệu. Chính vì thế nên worker/slave node cũng được gọi là data node. Một Data node sẽ chứa nhiều khối được phân nhỏ của tệp tin lớn ban đầu. Dựa theo chỉ thị từ Master node, các Data node này sẽ trực tiếp điều hành hoạt động thêm, bớt những khối nhỏ của tệp tin.

Apache Hadoop

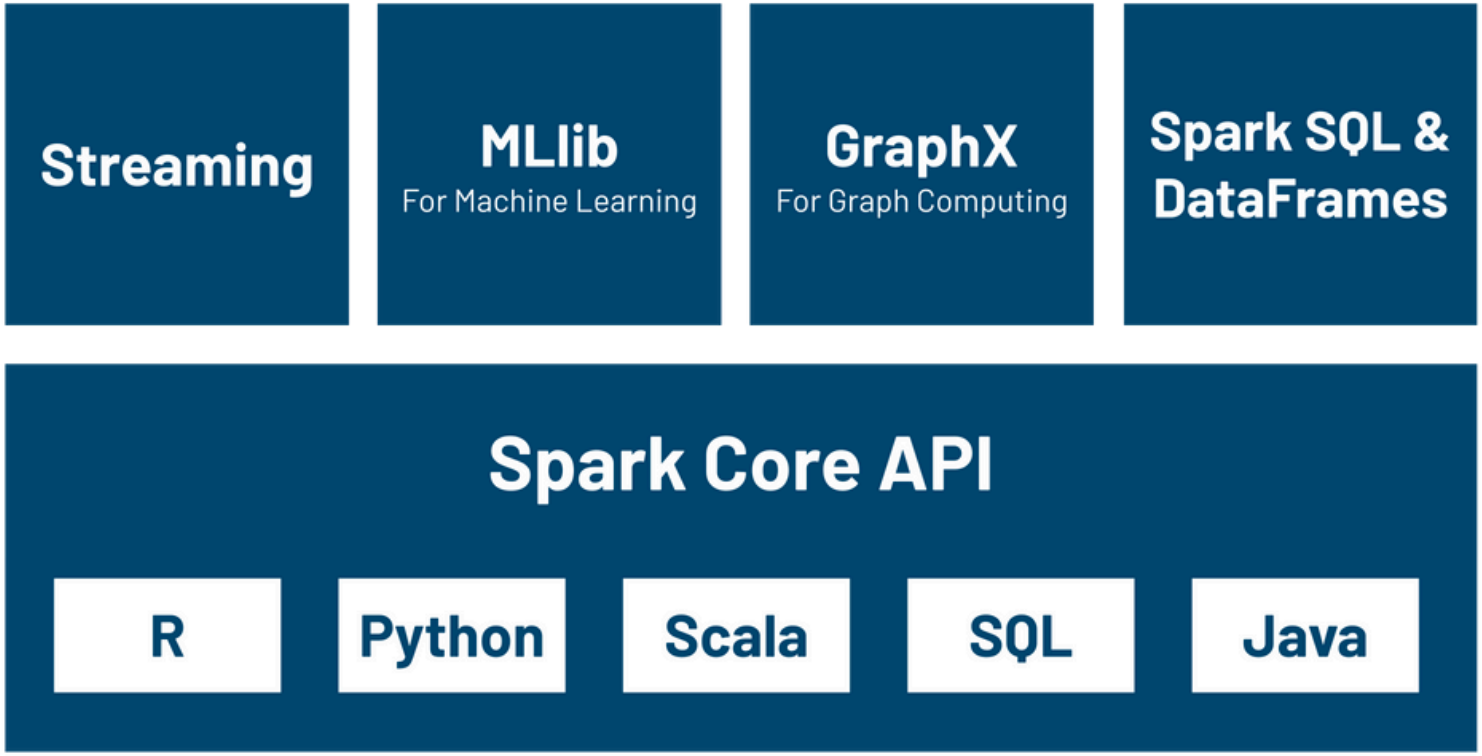
MapReduce

- Đây là hệ thống dựa trên YARN dùng để xử lý song song các tập dữ liệu lớn. Là cách chia một vấn đề dữ liệu lớn hơn thành các đoạn nhỏ hơn và phân tán nó trên nhiều máy chủ. Mỗi máy chủ có 1 tập tài nguyên riêng và máy chủ xử lý dữ liệu trên cục bộ. Khi máy chủ xử lý xong dữ liệu, chúng sẽ gửi trở về máy chủ chính.

Apache Spark

Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab. Apache Spark có thể chạy nhanh hơn 10 lần so với Hadoop ở trên đĩa cứng và 100 lần khi chạy trên bộ nhớ RAM.
Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

Apache Spark



Apache Spark

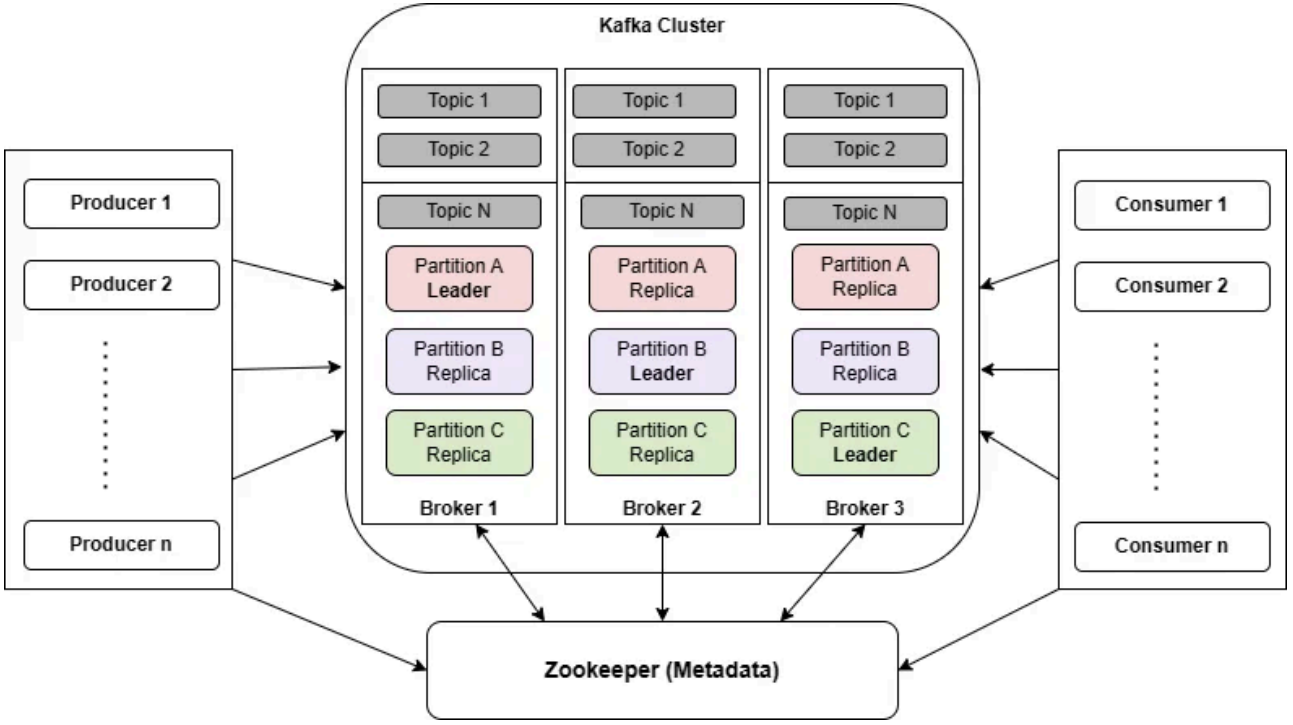
Apache Spark gồm có 5 thành phần chính: Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX, trong đó:

- Spark Core là nền móng cho nền tảng. Công cụ này chịu trách nhiệm quản lý bộ nhớ, phục hồi sau lỗi, lên lịch, phân phối và giám sát các công việc và tương tác với các hệ thống lưu trữ.
- Spark SQL cung cấp một kiểu data abstraction mới (SchemaRDD) nhằm hỗ trợ cho cả kiểu dữ liệu có cấu trúc (structured data) và dữ liệu nửa cấu trúc. Spark SQL hỗ trợ DSL (Domain-specific language) để thực hiện các thao tác trên DataFrames bằng ngôn ngữ Scala, Java hoặc Python và nó cũng hỗ trợ cả ngôn ngữ SQL với giao diện command-line và ODBC/JDBC server.
- Spark Streaming được sử dụng để thực hiện việc phân tích stream bằng việc coi stream là các mini-batches và thực hiệc kỹ thuật transformation đối với các dữ liệu mini-batches này.
- MLlib (Machine Learning Library): MLlib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ.
- GrapX: Grapx là nền tảng xử lý đồ thị dựa trên Spark. Nó cung cấp các Api để diễn tả các tính toán trong đồ thị bằng cách sử dụng Pregel Api.

Apache Kafka

- Apache Kafka là một nền tảng phát trực tuyến sự kiện phân tán mã nguồn mở. Apache Kafka được xây dựng nhằm mục đích xử lý dữ liệu streaming real-time (theo thời gian thực). Nói một cách đơn giản - Apache Kafka được phát triển để lưu trữ các streams of records (luồng ghi dữ liệu).
- Kafka được xây dựng dựa trên mô hình publish/subcribe, tương tự như bất kỳ hệ thống message nào khác. Các ứng dụng (đóng vai trò là producer) gửi các messages (records) tới một node kafka (broker) và nói rằng những messages này sẽ được xử lý bởi các ứng dụng khác gọi là consumers. Các messages được gửi tới kafka node sẽ được lưu trữ trong một nơi gọi là topic và sau đó consumer có thể subcribe tới topic đó và lắng nghe những messages này. Messages có thể là bất cứ thông tin gì như giá trị cảm biến, hành động người dùng, …

Apache Kafka



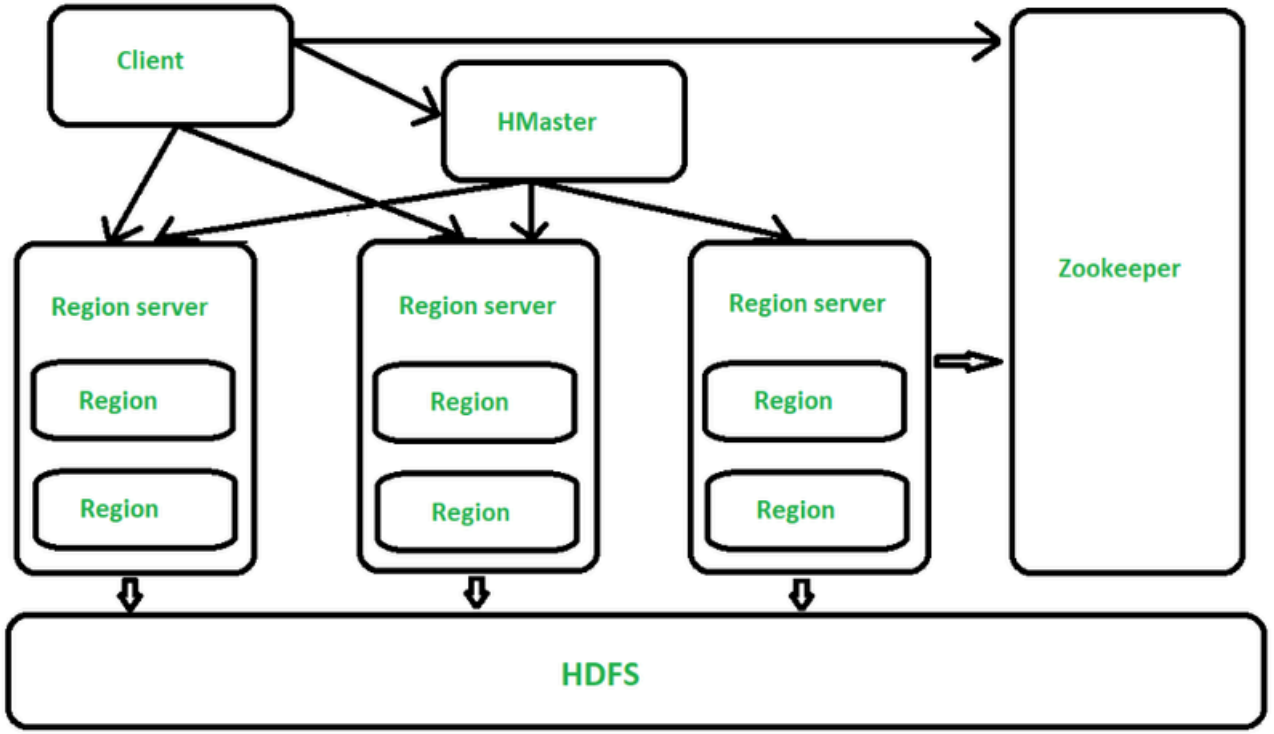
Apache HBase

- Hbase là giải pháp lưu trữ dữ liệu lớn (Big Data), linh hoạt và hoàn toàn miễn phí được rất nhiều tập đoàn công nghệ ưa chuộng.
- Hbase là hệ quản trị cơ sở dữ liệu dựa trên Hadoop, đây là mã nguồn mở nằm trong dự án của Apache, phát triển và mở rộng từ dự án lưu trữ Big Data của google. (được xây dựng dựa trên Google Bigtable). Hbase được viết bằng ngôn ngữ Java, có thể lưu trữ dữ liệu cực lớn từ terabytes đến petabytes.
- Hbase thực chất là một NoSQL điển hình nên vì thế các table của Hbase không có một schemas cố định nào và cũng không có mối quan hệ giữa các bảng. Hiện nay, có rất nhiều công ty và tập đoàn công nghệ lớn trên thế giới sử dụng Hbase, có thể kể đến: Facebook, Twitter, Yahoo, Adobe….

Apache HBase

- Mỗi table bao gồm rất nhiều row, có thể lên tới hàng tỷ rows trong 1 table của HBase, các row được xác định với nhau bởi 1 khóa duy nhất “rowkey”, rowkey trong HBase có chức năng tương tự với Primary key trong các hệ cơ sở dữ liệu thông thường. Các row trong cùng 1 talbe luôn được sắp xếp theo thứ tự tự điển theo rowkey.
- Mỗi row lại bao gồm nhiều cột (column) khác nhau, các cột này gộp lại thành column families. Tất cả các columns ở trong cùng 1 column families đều được lưu trữ cùng nhau ở trong storage file được gọi là HFile. Giá trị của mỗi column được gọi là cell, và mỗi cell chứa rất nhiều cặp “version (timestamp) value”.

Apache HBase



Unit 2.

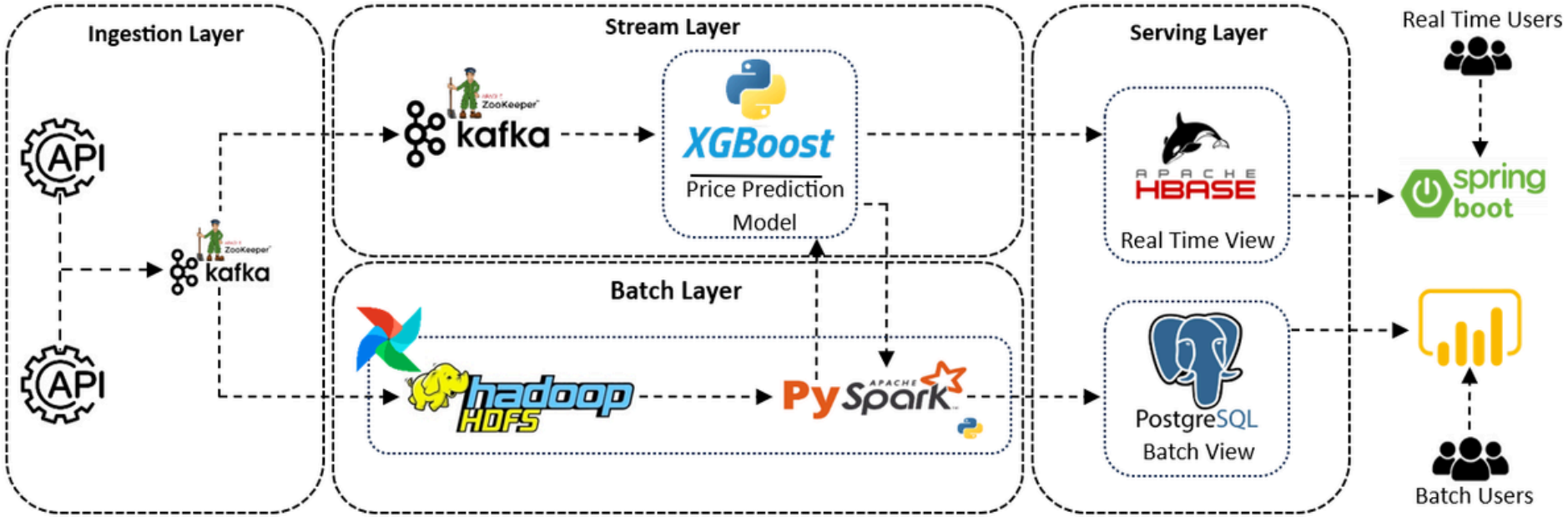
Ứng dụng trong Smartphone Price Prediction

1. Tổng quan

Dự án này nhằm mục đích dự đoán giá điện thoại thông minh bằng cách sử dụng kết hợp các kỹ thuật xử lý hàng loạt và xử lý luồng trong môi trường Dữ liệu lớn. Kiến trúc tuân theo mẫu Kiến trúc Lambda, cung cấp cả khả năng xử lý hàng loạt và thời gian thực cho người dùng.

2. Kiến trúc

Kiến trúc dự án bao gồm năm lớp chính: lớp nhập, lớp lô, lớp luồng, lớp phục vụ và lớp trực quan hóa.



By Aymane-MG

2. Kiến trúc

- **Lớp nhập**

Apache Kafka: Được sử dụng để nhập dữ liệu theo thời gian thực từ API cung cấp dữ liệu điện thoại thông minh.
Người tiêu dùng: Thu thập dữ liệu từ API và đưa dữ liệu đó vào lớp luồng và lớp bó.

- **Lớp luồng**

Nhà sản xuất: Một mô hình máy học được phát triển bằng XGBoost để ước tính giá điện thoại thông minh. Mô hình này chạy trong thời gian thực và lưu trữ các dự đoán ở chế độ xem thời gian thực.

- **Lớp hàng loạt**

HDFS: Dữ liệu từ API được lưu trữ trong HDFS như một phần của giải pháp hồ dữ liệu.
PySpark: Thực hiện chuyển đổi dữ liệu trên dữ liệu được lưu trữ bằng PySpark.
Luồng khí Apache: Điều phối quy trình xử lý hàng loạt.

- **Lớp phục vụ**

Chế độ xem thời gian thực: Được triển khai bằng HBase để cung cấp quyền truy cập theo thời gian thực vào giá điện thoại thông minh được dự đoán.
Chế độ xem hàng loạt: Dữ liệu đã chuyển đổi được lưu trữ trong PostgreSQL, dưới dạng giải pháp kho dữ liệu.

- **Lớp trực quan**

Ứng dụng web Spring Boot: Cung cấp giao diện người dùng để xem giá điện thoại thông minh theo thời gian thực

3. Công cụ

Big Data Stack:

- Apache Kafka (version 2.6.0)
- Apache HBase (version 1.2.6)
- Apache Hadoop (version 2.7.0)
- Apache Spark (version 3.3.4)
- PostgreSQL database

Programming Languages and Frameworks:

- Python (version 3.10.x or later)
- Java 17 (or compatible version)
- Spring Boot

Machine Learning Library:

- XGBoost