LƯU TRỮ VÀ XỬ LÝ DỮ LIỆU LỚN

Tên dự án: Hệ thống lưu trữ , phân tích và dự đoán các chỉ số của tiền mã hóa

Giáo viên hướng dẫn: TS. Trần Văn Đặng

Nhóm sinh viên thực hiện: Nhóm 13

Bùi Vũ Đức Nghĩa
Lê Tiến Khôi
Đỗ Đức Long
Trần Quang Minh

Nội dung

I. Tổng quan

II. Kiến trúc Lambda

III. Chi tiết trong hệ thống

IV. Kết luận

Tổng quan dự án

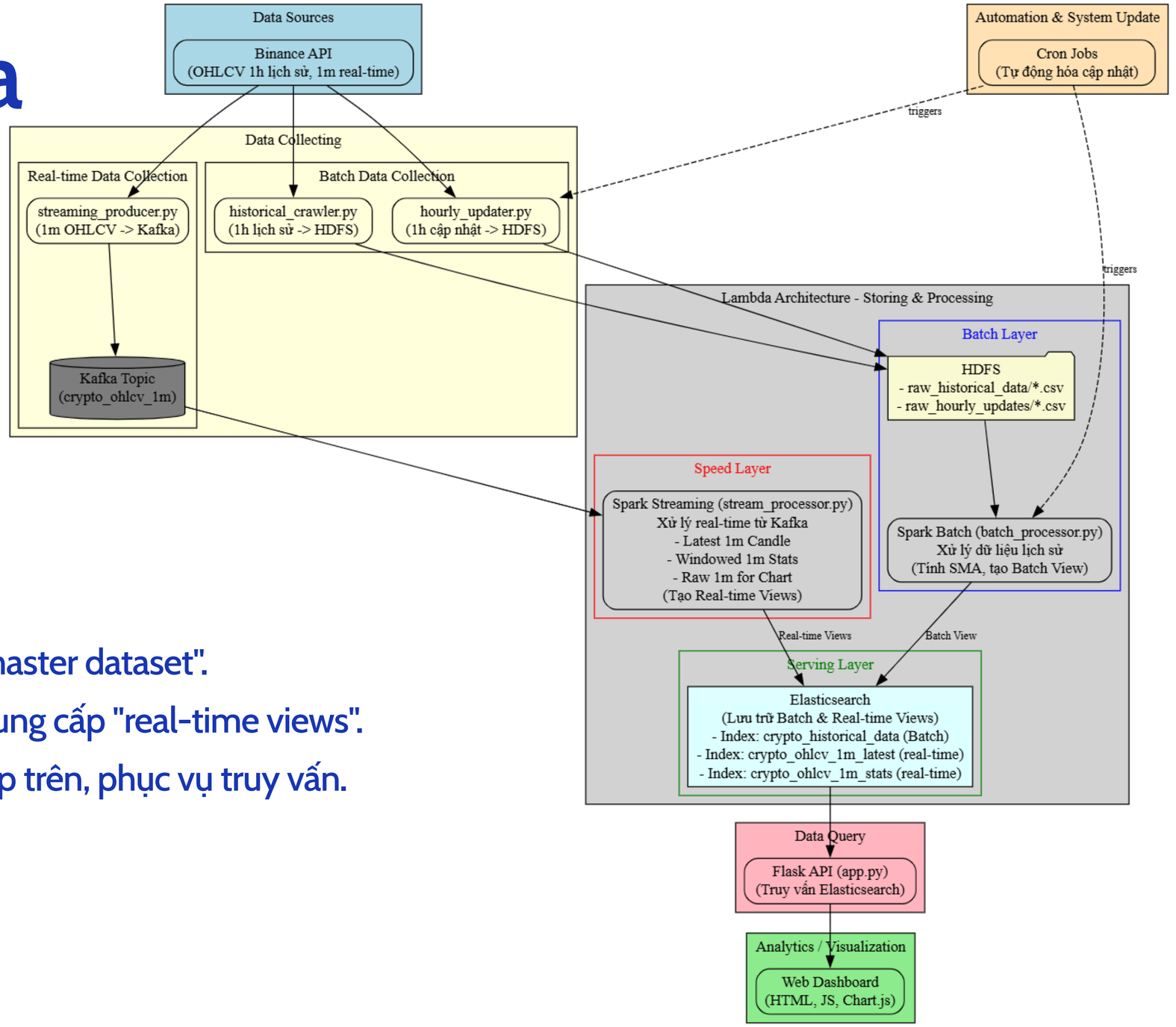
Giới thiệu chung

- Bối cảnh: Thị trường tiền điện tử biến động nhanh, dữ liệu lớn và liên tục.
- Nhu cầu: Cần một hệ thống mạnh mẽ để thu thập, xử lý, phân tích và trực quan hóa dữ liệu này.
- Giải pháp đề xuất: Áp dụng kiến trúc Lambda để xử lý hiệu quả cả dữ liệu lịch sử và thời gian thực.

Mục tiêu của project

- Xây dựng hệ thống hoàn chỉnh từ thu thập đến trực quan hóa dữ liệu giá tiền điện tử.
- Triển khai kiến trúc Lambda với Batch Layer và Speed Layer.
- Cung cấp dashboard tương tác cho người dùng theo dõi:
- Dữ liệu lịch sử (khung thời gian 1 giờ, SMA).
- Dữ liệu real-time (thống kê cửa sổ, biểu đồ giá 1 phút).

Kiến trúc lambda



Batch Layer: Xử lý dữ liệu lịch sử, tạo "master dataset".

Speed Layer: Xử lý dữ liệu streaming, cung cấp "real-time views".

Serving Layer: Kết hợp kết quả từ hai lớp trên, phục vụ truy vấn.

Chi tiết trong hệ thống

Apache Kafka:

Thu thập message stream OHLCV 1 phút.

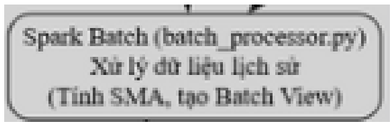
3.7.0

HDFS:

Lưu trữ dữ liệu lịch sử thô (CSV) và dữ liệu cập nhật hàng giờ.

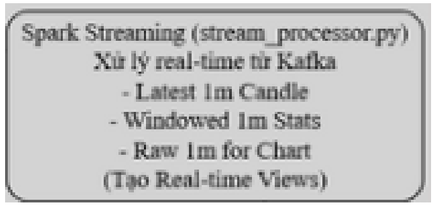
3.3.6

Apache Spark:



Batch Processing: Xử lý dữ liệu lịch sử, tính SMA.

3.4.3



Spark Streaming: Xử lý stream OHLCV 1 phút từ Kafka.

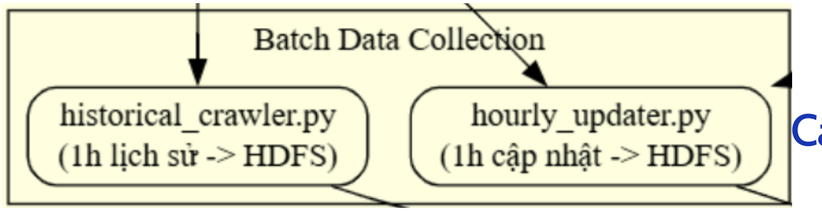
Elasticsearch:

Serving Layer - lưu trữ dữ liệu đã xử lý cho truy vấn nhanh.

8.14.0

Batch Layer

Nguồn: Binance API (qua ccxt).



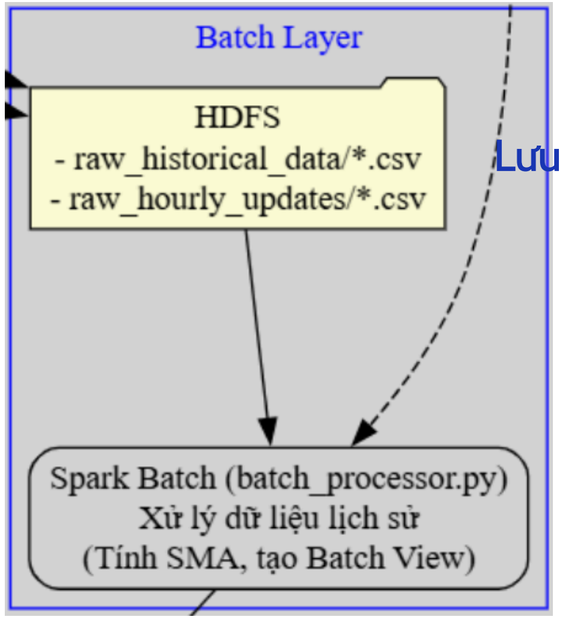
Cập nhật dữ liệu OHLCV 1 giờ hàng giờ.

File cấu hình cho craw của batch layer

```
CRYPTO_SYMBOLS="BTC/USDT,ETH/USDT"
# Timeframe for historical data and hourly updates. Example: "1h", "15m", "1d"
CRYPTO_TIMEFRAME="1h"
CRYPTO_START_DATE="YYYY-MM-DDTHH:MM:SSZ"
CRYPTO_END_DATE="YYYY-MM-DDTHH:MM:SSZ"
CRYPTO_EXCHANGE="binance"
```

Cấu hình crontab cho việc tự động lấy dữ liệu mỗi giờ

```
# 5 * * * * /path/to/venv/bin/python /path/to/project/hourly_updater.py
   >> /path/to/project/logs/hourly_updater.log 2>&1
# 10 * * * * /bin/bash /path/to/project/run_batch_processor.sh >> /path/
   to/project/logs/batch_processor_cron.log 2>&1
```

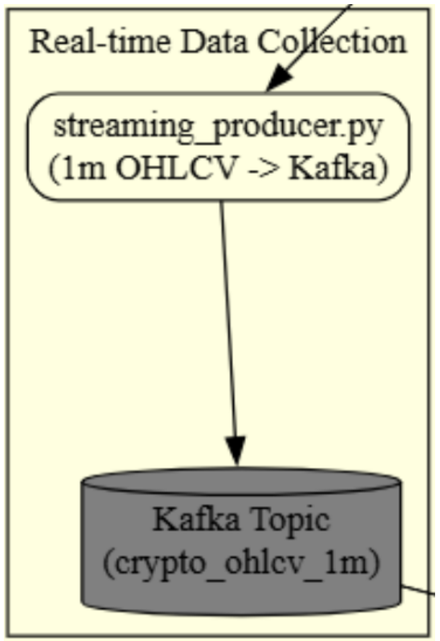


Lưu trữ: File CSV trên HDFS

Đọc dữ liệu CSV từ cả hai thư mục HDFS.
Xử lý: Làm sạch, trích xuất symbol/timeframe, chuyển đổi timestamp.
Tính toán: Đường trung bình động SMA7, SMA30.

Kết quả từ Spark Batch được ghi vào index Elasticsearch: crypto_historical_data.

Speed Layer

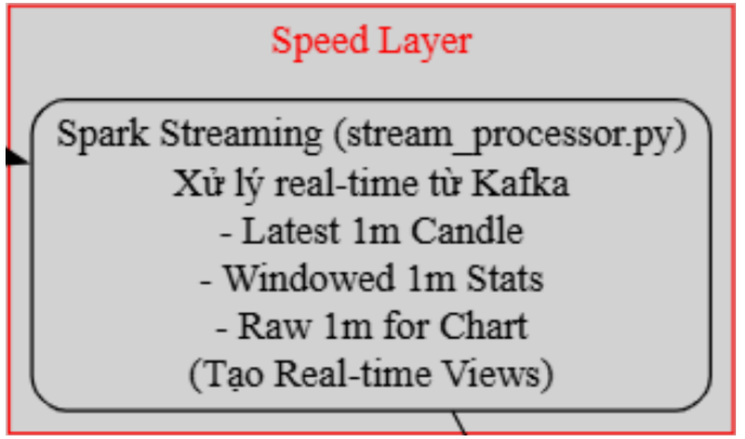


Fetch dữ liệu OHLCV 1 phút từ Binance API.
Gửi vào Kafka topic crypto_ohlcv_1m dưới dạng JSON.
Chạy liên tục, cập nhật mỗi phút.

stream_processor.py (Spark Streaming Job):

- Đọc dữ liệu từ Kafka topic crypto_ohlcv_1m.
- Ba luồng xử lý chính:

1. Nến 1 phút mới nhất.
2. Thống kê theo cửa sổ trượt (avg, min, max từ giá close 1 phút).
3. Dữ liệu OHLCV 1 phút thô cho biểu đồ.



Kết quả từ Spark Streaming được ghi vào 3 index Elasticsearch:
crypto_ohlcv_1m_latest.
crypto_ohlcv_1m_stats,
crypto_ohlcv_1m_chartdata-YYYY-MM-DD

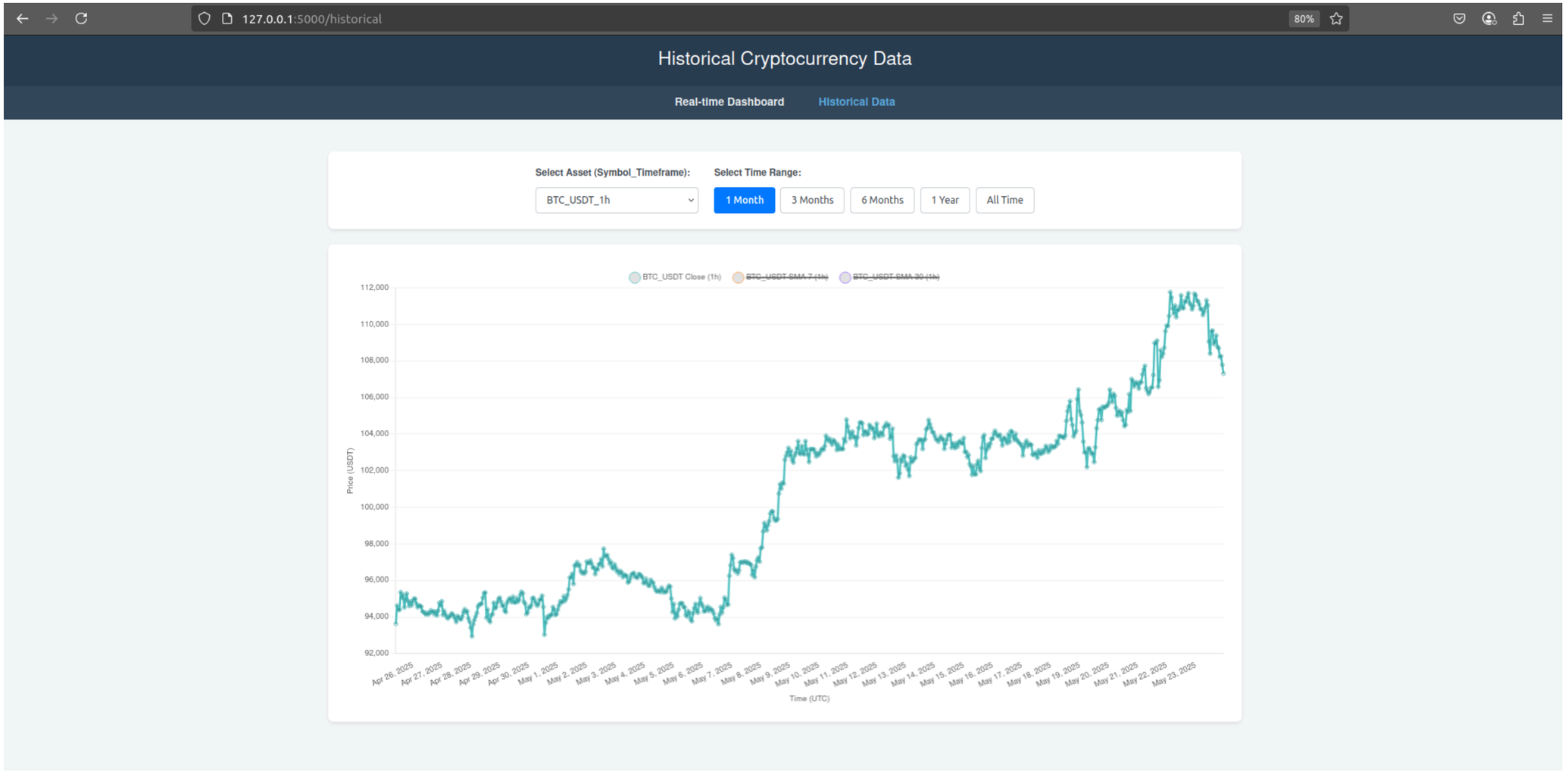
Backend, Data Query

+ app.py:

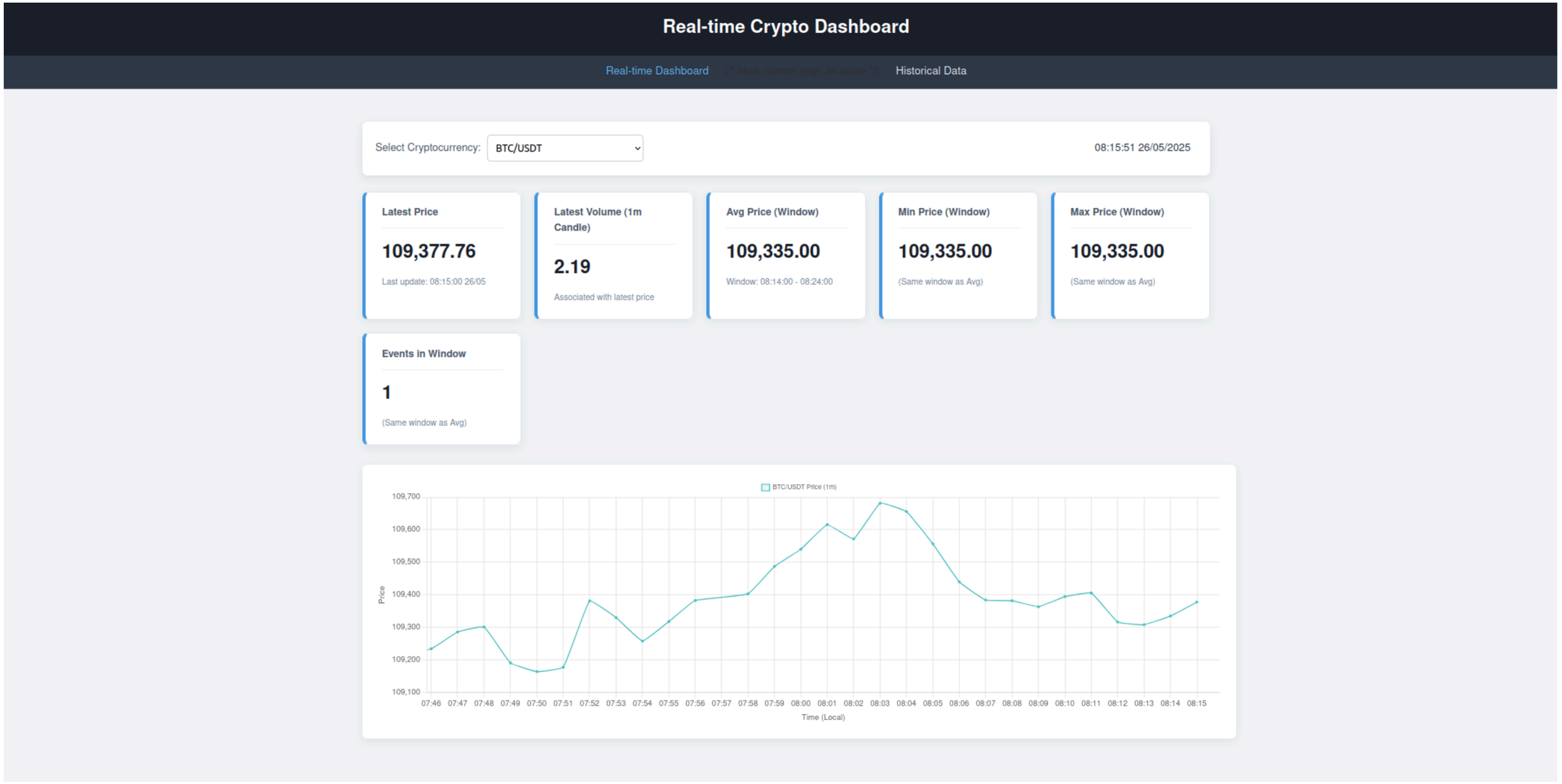
- Cung cấp các RESTful API endpoints.
- /historical: Phục vụ trang dữ liệu lịch sử.
- /api/historical_data/...: Trả về dữ liệu lịch sử đã xử lý từ Elasticsearch.
- /: Phục vụ trang dashboard real-time.
- /api/realtime_stats/...: Trả về thống kê real-time và giá mới nhất.
- /api/chart_data_1m/...: Trả về dữ liệu cho biểu đồ 1 phút.

+ Tương tác với Elasticsearch để lấy dữ liệu: sử dụng thư viện elasticsearch-py để xây dựng và gửi các câu lệnh truy vấn (queries) đến Elasticsearch.

Front-end, Analytics



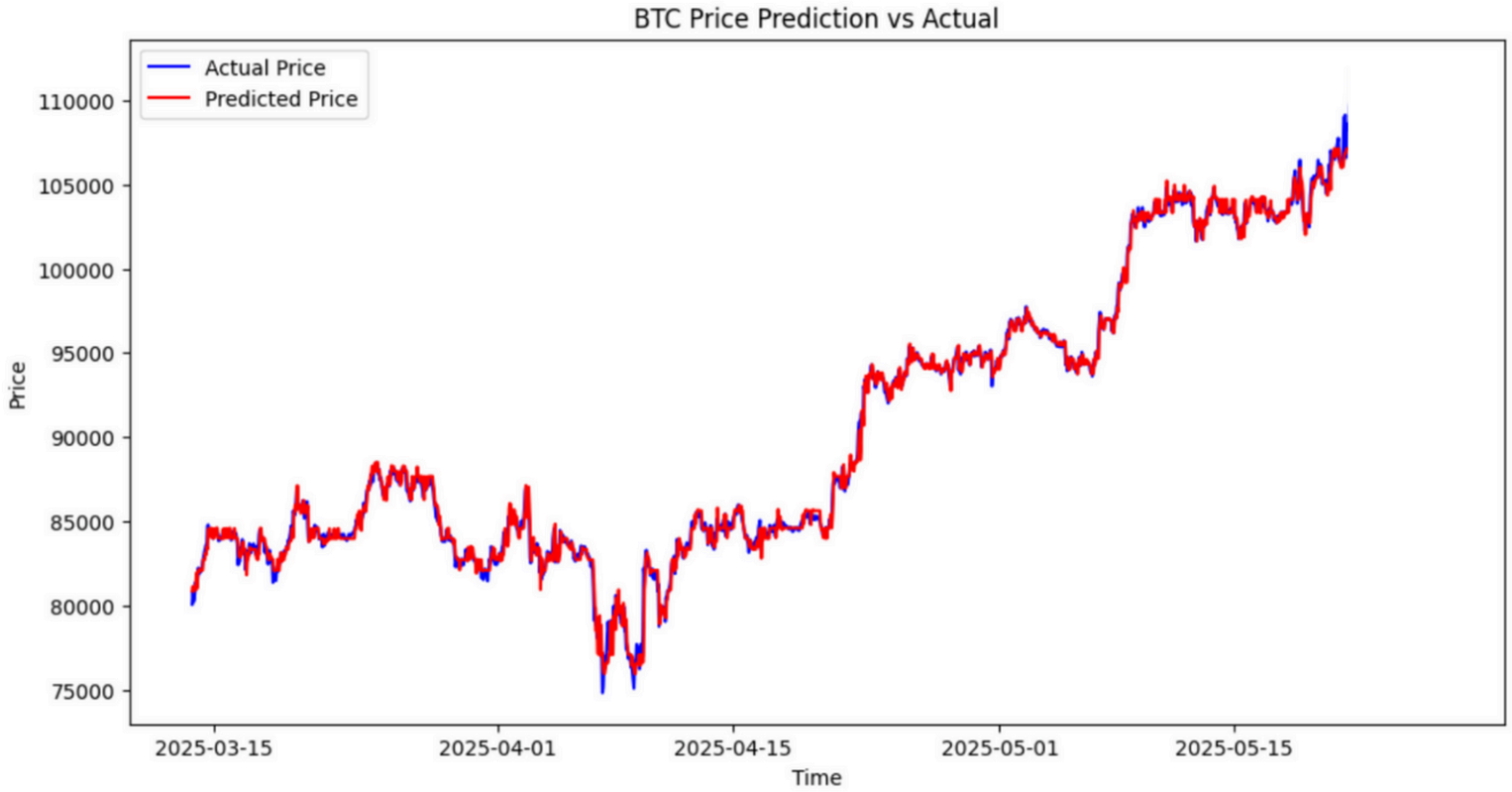
Front-end, Analytics



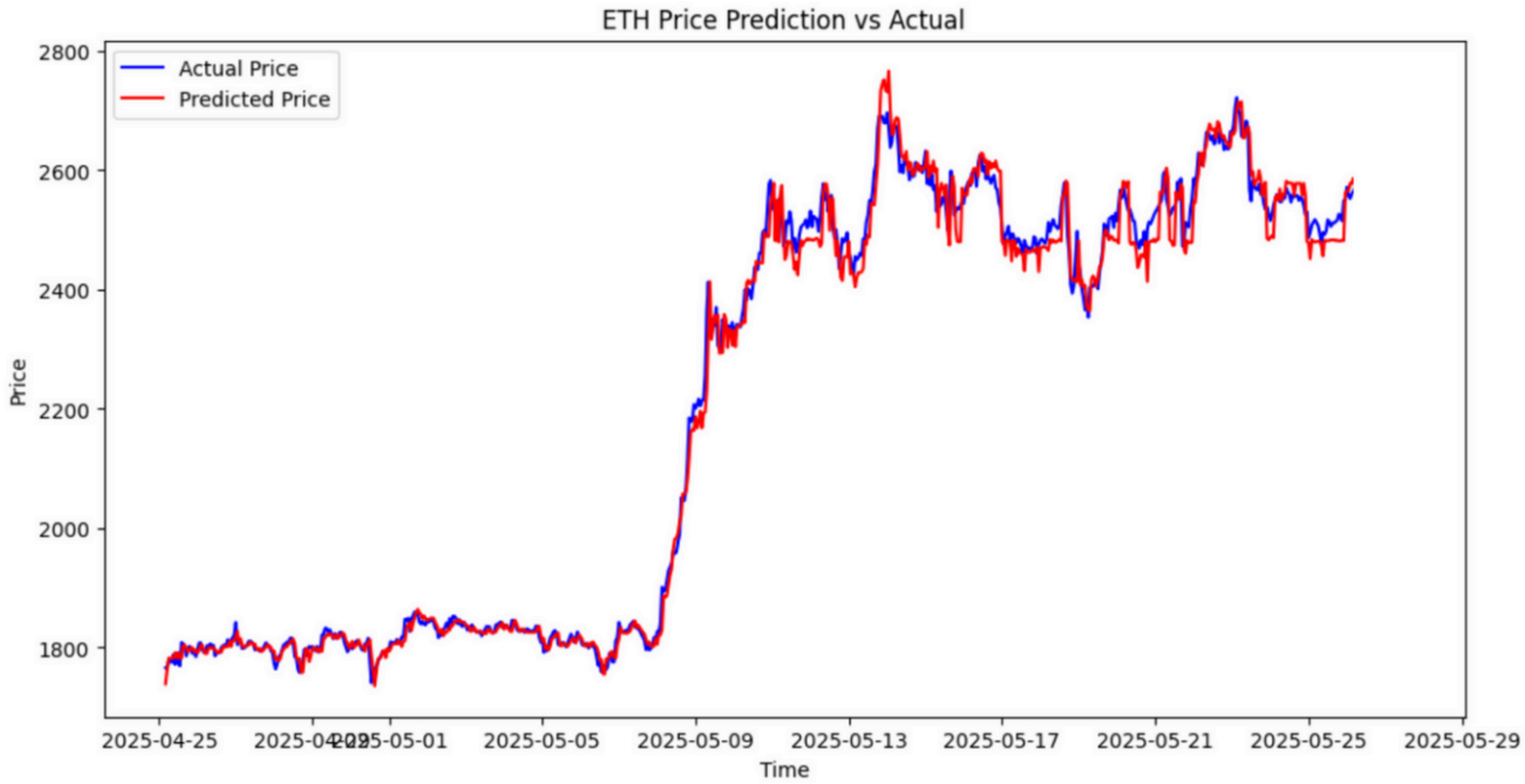
Dự đoán giá theo giờ

Time (UTC)	Predicted Close Price
2025-05-27 00:00	2577.96
2025-05-27 01:00	2577.96
2025-05-27 02:00	2577.96
2025-05-27 03:00	2577.96
2025-05-27 04:00	2589.25
2025-05-27 05:00	2602.51
2025-05-27 06:00	2602.51
2025-05-27 07:00	2602.92
2025-05-27 08:00	2602.92
2025-05-27 09:00	2602.51
2025-05-27 10:00	2602.51
2025-05-27 11:00	2614.56
2025-05-27 12:00	2619.77
2025-05-27 13:00	2628.60
2025-05-27 14:00	2627.54
2025-05-27 15:00	2627.54
2025-05-27 16:00	2628.57
2025-05-27 17:00	2628.33
2025-05-27 18:00	2627.95
2025-05-27 19:00	2630.84
2025-05-27 20:00	2633.57
2025-05-27 21:00	2643.44
2025-05-27 22:00	2649.83
2025-05-27 23:00	2652.59

Biểu đồ so sánh giá thực tế và giá dự đoán của BTC



Biểu đồ so sánh giá thực tế và giá dự đoán của ETH



Kết luận

- Ưu Điểm Của Hệ Thống

Triển khai thành công kiến trúc Lambda, xử lý cả batch và streaming.
Cập nhật dữ liệu tự động và định kỳ.
Giao diện trực quan, cung cấp thông tin hữu ích.
Nền tảng công nghệ Big Data mạnh mẽ, có khả năng mở rộng.

- Nhược Điểm và Hạn Chế

Hiệu năng trên VM đơn lẻ khi chạy tất cả các thành phần.
Logic trích xuất thông tin từ tên file cần cải thiện độ robust.
Quản lý ILM (Index Lifecycle Management) cho Elasticsearch
cần cấu hình cẩn thận.
Chưa có các phân tích sâu bằng Machine Learning.

Kết luận

- Khó Khăn Gặp Phải

Tương thích phiên bản giữa các công nghệ (Spark, Hadoop, Kafka, Elasticsearch).
Cấu hình kết nối mạng, đặc biệt Spark - Elasticsearch.
Quản lý tài nguyên trên VM.
Debug lỗi trong Spark Streaming.


thank you

Hadoop config

fs.defaultFS: hdfs://localhost:9000. HDFS và NameNode đang chạy tại port 9000.

hadoop.tmp.dir: /home/root/hadoop_tmp_data. Thư mục tạm thời cho Hadoop.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/root/hadoop_tmp_data</value>
        <description>A base for other temporary directories.</description>
    </property>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value> <!-- HDFS run in port 9000 -->
        <description>The name of the default file system.</description>
    </property>
</configuration>
```

HDFS config

dfs.replication: 1. chạy trên một máy đơn (pseudo-distributed mode), số lượng bản sao là 1.

dfs.namenode.name.dir: file:/home/YOUR_USERNAME/hadoop_data/hdfs/namenode. Đường dẫn lưu trữ metadata của NameNode.

dfs.datanode.data.dir: file:/home/YOUR_USERNAME/hadoop_data/hdfs/datanode. Đường dẫn lưu trữ các block dữ liệu của DataNode.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
        <description>Default block replication.</description>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/home/YOUR_USERNAME/hadoop_data/hdfs/namenode</value>
        <description>Directory where NameNode metadata is stored.</description>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/home/YOUR_USERNAME/hadoop_data/hdfs/datanode</value>
        <description>Directory where DataNode blocks are stored.</description>
    </property>
</configuration>
```

Elasticsearch config

network.host: 192.168.30.128:
http.port: 9200
http.host: 0.0.0.0
cluster.initial_master_nodes: ["ubuntu"]
xpack.security.enabled: false:

```
network.host: 192.168.30.128
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
http.port: 9200
# ---------------------------------- Discovery -----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
#discovery.seed_hosts: ["host1", "host2"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
#cluster.initial_master_nodes: ["node-1", "node-2"]
# ---------------------------------- Various -----------------------------------
#
# Allow wildcard deletion of indices:
#
#action.destructive_requires_name: false

# Enable security features
xpack.security.enabled: false

xpack.security.enrollment.enabled: true

# Enable encryption for HTTP API client connections, such as Kibana, Logstash, and Agents
xpack.security.http.ssl:
  enabled: false
 # keystore.path: certs/http.p12

# Enable encryption and mutual authentication between cluster nodes
xpack.security.transport.ssl:
  enabled: false
#  verification_mode: certificate
#  keystore.path: certs/transport.p12
#  truststore.path: certs/transport.p12
# Create a new cluster with the current node only
# Additional nodes can still join the cluster later
cluster.initial_master_nodes: ["ubuntu"]

# Allow HTTP API connections from anywhere
# Connections are encrypted and require user authentication
http.host: 0.0.0.0
```

Zookeeper config

```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# the directory where the snapshot is stored.
dataDir=/tmp/zookeeper
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections since this is a non-production config
maxClientCnxns=0
# Disable the adminserver by default to avoid port conflicts.
# Set the port to something non-conflicting if choosing to enable this
admin.enableServer=false
# admin.serverPort=8080
```

kafka config

```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

############################# Socket Server Settings #############################

# The address the socket server listens on. If not configured, the host name will be equa
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 90
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092

# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
#advertised.listeners=PLAINTEXT://your.host.name:9092

# Maps listener names to security protocols, the default is for them to be the same. See
#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEX

# The number of threads that the server uses for receiving requests from the network and
num.network.threads=3

# The number of threads that the server uses for processing requests, which may include
num.io.threads=8

# The send buffer (SO_SNDBUF) used by the socket server
socket.send.buffer.bytes=102400

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept (protection against O
socket.request.max.bytes=104857600


############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=/tmp/kafka-logs
```