ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN

Lưu trữ và xử lý dữ liệu chứng khoán thời gian thực

NHÓM 37

Phạm Vũ Tuấn Đạt - 20210158
Trần Cao Sơn - 20215472
Trần Quang Khải - 20215401
Hoàng Nghĩa Hiệp - 20210329
Nguyễn Phúc Thắng - 20210784

Giảng viên giảng dạy: Trần Việt Trung

Học phần: Lưu trữ dữ liệu lớn - IT4931

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 12/2023

MỤC LỤC

DANH MỤC HÌNH VẼ

CHƯƠNG 1. GIỚI THIỆU BÀI TOÁN

1.1 Giới thiệu

Trong thế giới tài chính hiện đại, việc lưu trữ và xử lý dữ liệu chứng khoán thời gian thực là vô cùng quan trọng. Điều này không chỉ giúp các nhà đầu tư có được cái nhìn sâu sắc và kịp thời về thị trường, mà còn là cơ sở để phát triển các chiến lược giao dịch và quản lý rủi ro hiệu quả. Báo cáo này sẽ trình bày phương pháp và các công cụ lưu trữ dữ liệu lớn trong quá trình xây dựng hệ thống xử lý và phân tích dữ liệu chứng khoán trong thời gian thực.

1.2 Vấn đề

Giá cổ phiếu thay đổi trong thời gian thực và thường khó dự đoán bằng các phương pháp truyền thống, tạo ra một thách thức lớn cho các nhà đầu tư và nhà phân tích. Sự biến động không thể dự đoán của giá cổ phiếu là do nhiều yếu tố - từ tin tức kinh tế, báo cáo tài chính, đến hành động của các nhà đầu tư khác. Hơn nữa, các yếu tố dẫn đến thay đổi giá cổ phiếu phức tạp và đa dạng, bao gồm cả các yếu tố cơ bản của công ty và yếu tố kỹ thuật trên thị trường. Điều này đòi hỏi cần có phương pháp tiên tiến và đa dạng hơn trong việc lưu trữ và phân tích dữ liệu để có thể hiểu và dự báo chính xác các xu hướng giá cổ phiếu, đặc biệt là trong môi trường giao dịch nhanh và biến động cao của ngày nay.

CHƯƠNG 2. CÁC NGUỒN DỮ LIỆU & CẤU TRÚC

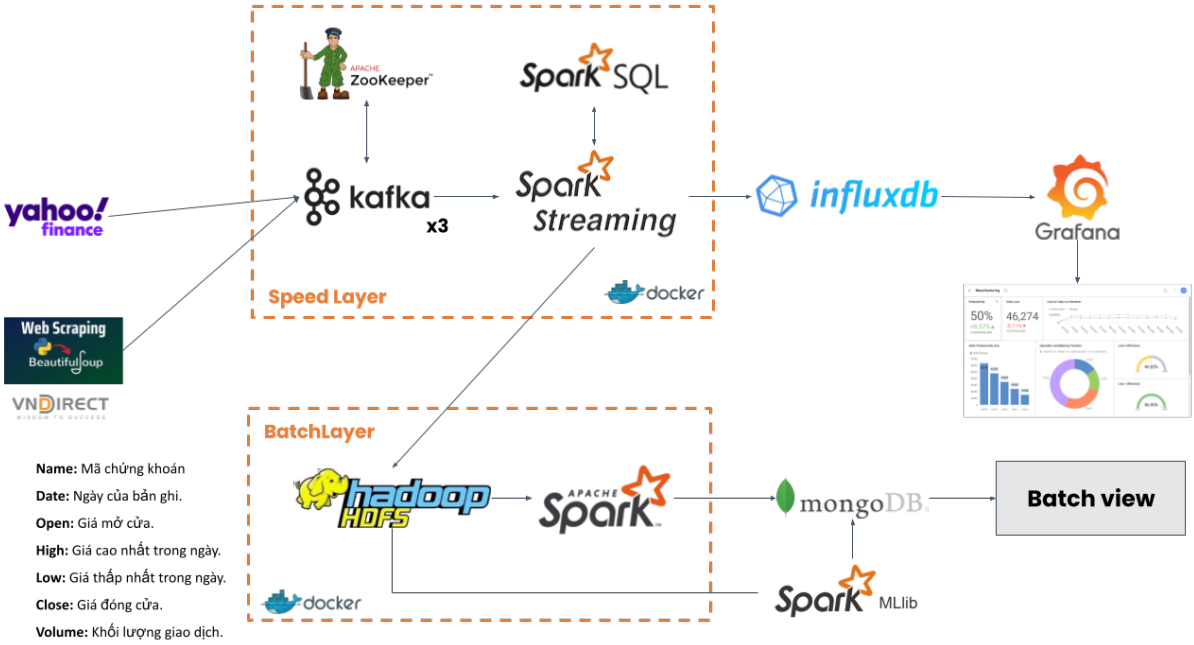
Nguồn dữ liệu cho việc phân tích và xử lý chứng khoán thời gian thực trong dự án này chủ yếu đến từ hai nguồn là YFinance và dữ liệu được crawl trực tiếp từ trang web `https://banggia.vndirect.com.vn/chung-khoan/hnx30`. Dữ liệu này bao gồm thông tin chi tiết về các mã chứng khoán, được cấu trúc như sau:

- **Name**: Mã chứng khoán
- **Date**: Ngày của bản ghi.
- **Open**: Giá mở cửa.
- **High**: Giá cao nhất trong ngày.
- **Low**: Giá thấp nhất trong ngày.
- **Close**: Giá đóng cửa.
- **Volume**: Khối lượng giao dịch.

Thông tin từ cả hai nguồn dữ liệu này đóng vai trò quan trọng trong việc cung cấp dữ liệu chính xác và cập nhật cho hệ thống phân tích.

CHƯƠNG 3. KIẾN TRÚC HỆ THỐNG

Phân tích cổ phiếu thời gian thực đòi hỏi một hệ thống có khả năng xử lý dữ liệu lớn một cách nhanh chóng và chính xác. Chương này sẽ mô tả chi tiết về kiến trúc hệ thống và các công nghệ được sử dụng.



Hình 3.1: Kiến trúc tổng quan của hệ thống phân tích cổ phiếu thời gian thực.

3.1 Thu Thập Dữ Liệu

Dữ liệu được thu thập real-time từ YFinance và crawl từ bảng giá VNDirect (như đã trình bày trong chương trước). Sau đó được chuyển đến Kafka, một nền tảng xử lý stream-data, để quản lý và phân phối dữ liệu đầu vào.

3.2 Tầng Xử Lý Dữ Liệu

Tầng xử lý dữ liệu được chia thành hai lớp chính: Batch Layer và Speed Layer.

3.2.1 Speed Layer

Lớp Speed xử lý dữ liệu thời gian thực sử dụng Spark Streaming và Spark SQL để cung cấp thông tin cập nhật liên tục và nhanh chóng. Dữ liệu sau khi transform được chuyển đến InfluxDB và Hadoop HDFS để lưu trữ.

3.2.2 Batch Layer

Lớp Batch xử lý dữ liệu lớn thông qua Hadoop HDFS và Apache Spark để thực hiện các tính toán theo lô (batch processing) với tần suất là 1 lần / ngày. Kết quả sau khi xử lý xong sẽ được lưu trữ vào MongoDB.

3.3 Phân Tích và Trực Quan Hóa

Để phân tích dữ liệu, MLlib được sử dụng để cài đặt mô hình học máy với dữ liệu được lưu trữ tại Hadoop HDFS.

Kết quả cuối cùng, đối với Speed Layer, dữ liệu tại InfluxDB sẽ được trực quan hóa dưới dạng stream thông qua Grafana. Đối với Batch Layer dữ liệu từ MongoDB sẽ được hiện thị dưới dạng Batch View vào cuối mỗi ngày.

Như vậy, hệ thống sử dụng kết hợp giữa xử lý theo lô (batch processing) và xử lý theo dòng dữ liệu (stream processing). Với mục tiêu, cung cấp cái nhìn sâu sắc và cập nhật liên tục về thị trường cổ phiếu.

CHƯƠNG 4. CÁC TRẢI NGHIỆM TỪ DỰ ÁN

Qua thời gian hoàn thiện dự án, nhóm đã có nhiều trải nghiệm khác nhau. Trong đó, có thể kể đến những trải nghiệm giá trị bao gồm:

4.1 Thu thập dữ liệu: Web Scraping với BeautifulSoup

```
27      def retrieve_real_time_data(producer, stock_symbol, kafka_topic):
60              driver.get("https://banggia.vndirect.com.vn/chung-khoan/hnx30")
61              sourceCode = driver.page_source
62              soup = BeautifulSoup(sourceCode, "html.parser")
63              items = soup.select('#banggia-khop-lenh-body tr')
64              for row in items:
65                  row_data = {}
66                  symbols = row.select('td span')
67                  stock = row.select_one('.has-symbol').text[1:-1]
68                  if not stock in stock_symbols:
69                      continue
70                  print(stock)
71                  for symbol in symbols:
72                      row_data[symbol.get('id')] = symbol.text
73                  real_time_data_point = {
74                      'stock': stock,
75                      'date': datetime.now().strftime("%Y-%m-%dT%H:%M:%S%z"),
76                      'open': None,
77                      'high': row_data[f'{stock}ceil'],
78                      'low': row_data[f'{stock}floor'],
79                      'close': None,
80                      'volume': None
81                  }
```

Hình 4.1: Code web scraping với BeautifulSoup

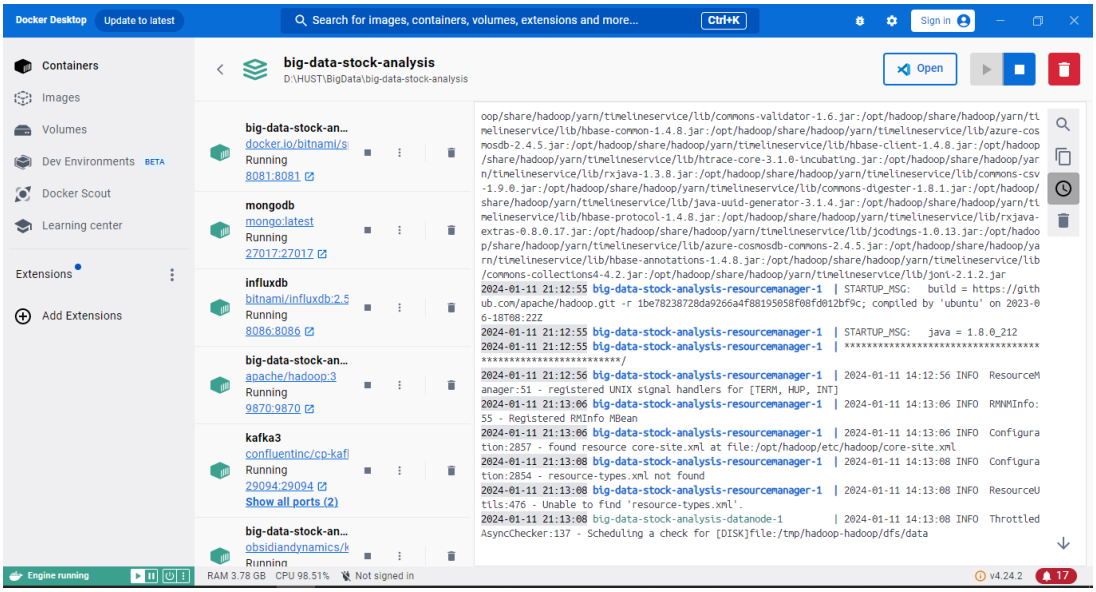
```
PS E:\src code 2\big-data-stock-analysis> python -u "e:\src code 2\big-data-stock-analysis\producer\producer_utils.py"
[**********100%**********]  1 of 1 completed
{'stock': 'BTC-USD', 'date': '2024-01-11T16:07:00+00:00', 'open': 46605.546875, 'high': 46605.546875, 'low': 46605.546875, 'close': 46605.546875, 'volume': 0.0}
{'stock': 'BCC', 'date': '2024-01-11T23:10:01', 'high': '10.70', 'low': '8.90', 'volume': '197,10'}
{'stock': 'BVS', 'date': '2024-01-11T23:10:01', 'high': '28.00', 'low': '23.00', 'volume': '194,70'}
{'stock': 'CEO', 'date': '2024-01-11T23:10:01', 'high': '24.60', 'low': '20.20', 'volume': '7,474,20'}
{'stock': 'DDG', 'date': '2024-01-11T23:10:01', 'high': '6.60', 'low': '5.40', 'volume': '1,159,90'}
{'stock': 'DTD', 'date': '2024-01-11T23:10:01', 'high': '27.60', 'low': '22.60', 'volume': '649,30'}
{'stock': 'DVM', 'date': '2024-01-11T23:10:01', 'high': '12.70', 'low': '10.50', 'volume': '759,60'}
{'stock': 'DXP', 'date': '2024-01-11T23:10:01', 'high': '13.60', 'low': '11.20', 'volume': '412,40'}
{'stock': 'HLD', 'date': '2024-01-11T23:10:01', 'high': '28.10', 'low': '23.10', 'volume': '29,60'}
{'stock': 'HUT', 'date': '2024-01-11T23:10:01', 'high': '22.80', 'low': '18.80', 'volume': '6,508,10'}
{'stock': 'IDC', 'date': '2024-01-11T23:10:01', 'high': '58.30', 'low': '47.70', 'volume': '2,548,60'}
{'stock': 'L14', 'date': '2024-01-11T23:10:01', 'high': '48.00', 'low': '39.40', 'volume': '192,00'}
```

Hình 4.2: Kết quả producer khi sử dụng dữ liệu crawl

Dữ liệu được crawl sử dụng BeautifulSoup sẽ được sử dụng làm một nguồn producer cung cấp dữ liệu cho hệ thống.

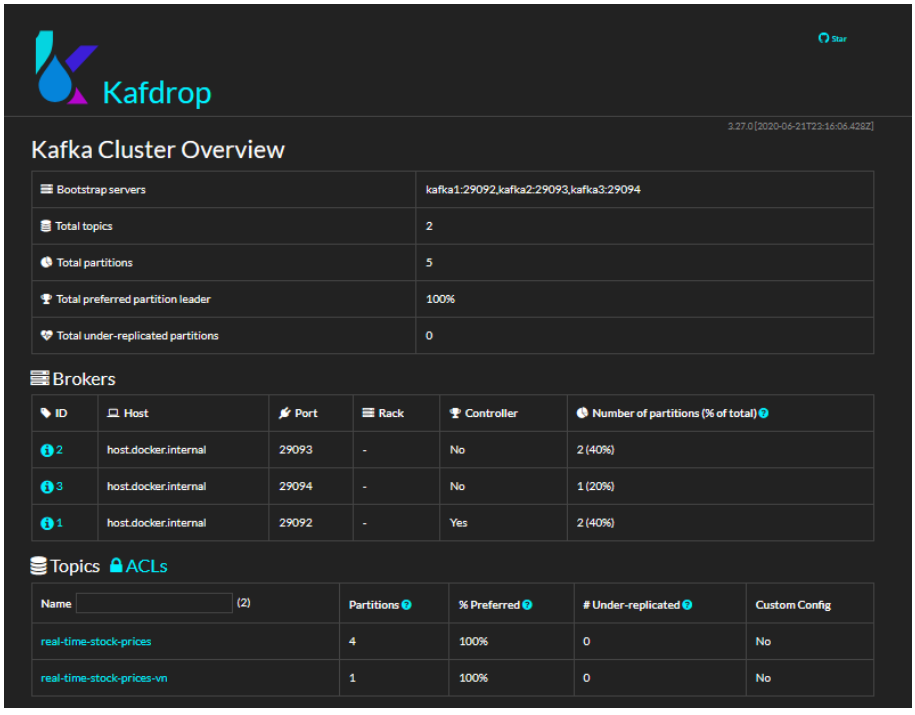
4.2 Sử dụng Docker để thiết lập các thành phần hệ thống trên môi trường phân tán

Để giả lập môi trường phân tán, các thành phần trong hệ thống đều được khởi chạy trên Docker thông qua docker-compose.yml



Hình 4.3: Các thành phần hệ thống được khởi chạy trên Docker

4.3 Thiết lập và sử dụng Single node - Multi Kafka brokers cùng ZooKeeper



Hình 4.4: Kafka Web UI hiển thị Single node - Multi Kafka brokers

Giao diện Kafdrop được sử dụng để hiển thị Kafka topics, trực quan hoá các thông tin về: các brokers, topics, partitions, consumers, v.v.

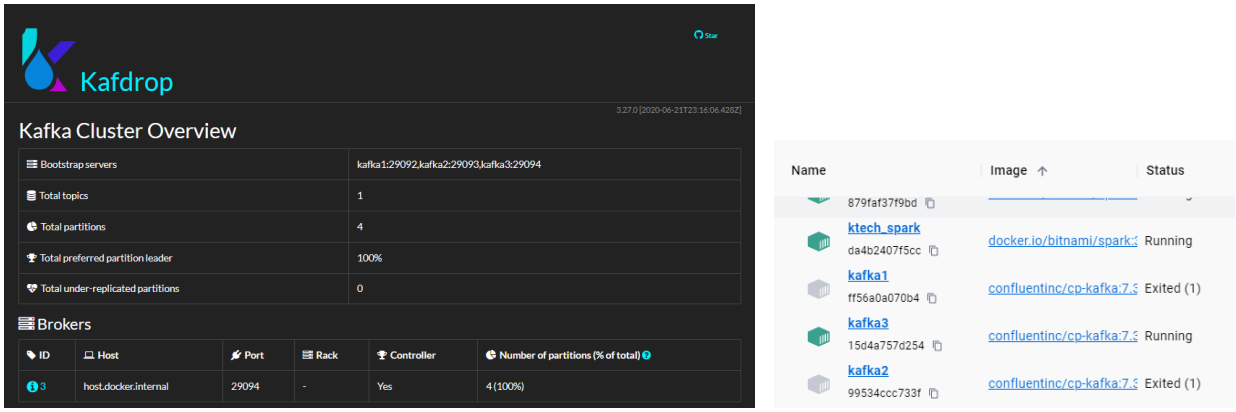
Thông qua Kafdrop [1] có thể thấy, hệ thống gồm một node đơn và 3 Kafka brokers bao gồm: kafka1:29092, kafka2:29093, kafka3:29094.

[1] Kafdrop có thể truy cập được tại địa chỉ: http://localhost:19000/

4.4 Phân loại và tổ chức dữ liệu thông qua topics Kafka

Từ hình 4.4, có thể thấy 2 topics real-time-stock-prices và real-time-stock-prices-vn được sử dụng để phân loại và tổ chức dữ liệu.

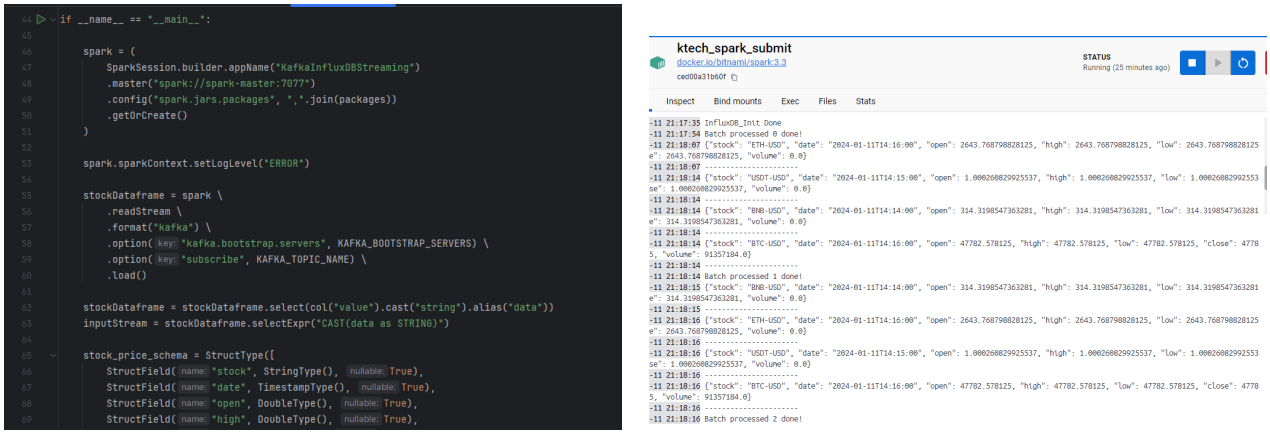
4.5 Kiểm tra khả năng chống chịu lỗi của Kafka



Hình 4.5: Hệ thống vẫn hoạt động ngay cả khi broker 1 và 2 ngừng hoạt động.

Khi thử tắt 2 brokers số 1 và 2, tại Kafdrop ta thấy số lượng broker còn hoạt động hiện tại chỉ còn lại 1 là broker số 3

4.6 Xử lý dữ liệu thời gian thực sử dụng Apache Spark Streaming



Hình 4.6: Xử lý dữ liệu thời gian thực sử dụng Apache Spark Streaming

Từ hình 4.6 phía bên phải, log của dòng dữ liệu (stream processing) đang được hiển thị theo các mini-batch.

4.7 Thực hiện các thao tác chuyển đổi và truy vấn dữ liệu bằng Spark SQL

```
df = df.dropDuplicates()

# Convert date from StringType to TimestampType and sort
df = df.withColumn( colName: "date", F.to_timestamp("date"))
df = df.orderBy( *cols: "stock", "date")

# Basic Statistics for each stock
basic_stats = df.groupBy("stock").agg(
    F.mean("open").alias("avg_open"),
    F.mean("high").alias("avg_high"),
    F.mean("low").alias("avg_low"),
    F.mean("close").alias("avg_close"),
    F.mean("volume").alias("avg_volume"),
    F.stddev("close").alias("std_dev_close"),
    F.max("high").alias("historical_high"),
    F.min("low").alias("historical_low")
)

daily_window_spec = Window.partitionBy("stock").orderBy("date")

# Calculate daily returns
df = df.withColumn( colName: "prev_day_close", F.lag("close").over(daily_window_spec))
df = df.withColumn( colName: "daily_return", (F.col("close") - F.col("prev_day_close")) / F.col("prev_day_close"))

# Calculate 1-day moving average
df = df.withColumn( colName: "moving_avg_1d", F.avg("close").over(daily_window_spec.rowsBetween( start: 0,  end: 0)))
```

Hình 4.7: Các thao tác trên dữ liệu sử dụng Spark SQL

Hình 4.7, là một số thao tác trong quá trình xử lý dữ liệu với Spark SQL như: dropDuplicate(), orderBy(), tính max, min, mean, std-dev, v.v. hay phân nhỏ dữ liệu với Window.partitionBy().

4.8 Xử lý dữ liệu theo lô sử dụng Apache Spark

```
+--------+----------------+----------------+----------------+----------------+----------+-------------------+-----------------+----------------
--+
|   stock|       avg_open|        avg_high|         avg_low|       avg_close| avg_volume|      std_dev_close|   historical_high|    historical_l
ow|
+--------+----------------+----------------+----------------+----------------+----------+-------------------+-----------------+----------------
--+
|USDT-USD|0.999695748090744|0.999695748090744|0.999695748090744|0.999695748090744|1.2331008E8|4.488639366410215E-5|0.9997274875640869|0.99966400861740
11|
| BNB-USD|307.5506896972656|307.5506896972656|307.5506896972656|307.5506896972656|   811904.0| 0.01959390128532059| 307.5645446777344| 307.53683471679
69|
| ETH-USD| 2612.1494140625| 2612.1494140625| 2612.1494140625| 2612.1494140625|2.8660736E7|  0.6867360291894742| 2612.635009765625| 2611.6638183593
75|
| BTC-USD|       46892.875|       46892.875|       46892.875|       46892.875|7.5083776E7|  13.650475439937178|    46902.52734375|    46883.222656
25|
+--------+----------------+----------------+----------------+----------------+----------+-------------------+-----------------+----------------
--+

+--------+-------------------+--------------------+------------------+
|   stock|               date|        daily_return|     moving_avg_1d|
+--------+-------------------+--------------------+------------------+
| BNB-USD|2024-01-11 15:47:00|                null| 307.5645446777344|
| BNB-USD|2024-01-11 15:48:00|-9.00947830853990...| 307.5368347167969|
| BTC-USD|2024-01-11 15:48:00|                null|    46883.22265625|
| BTC-USD|2024-01-11 15:49:00|4.117611035730815...|    46902.52734375|
| ETH-USD|2024-01-11 15:48:00|                null| 2611.663818359375|
| ETH-USD|2024-01-11 15:49:00|3.718669299711378E-4| 2612.635009765625|
|USDT-USD|2024-01-11 15:47:00|                null|0.9996640086174011|
|USDT-USD|2024-01-11 15:48:00|6.350028223341403E-5|0.9997274875640869|
+--------+-------------------+--------------------+------------------+
```

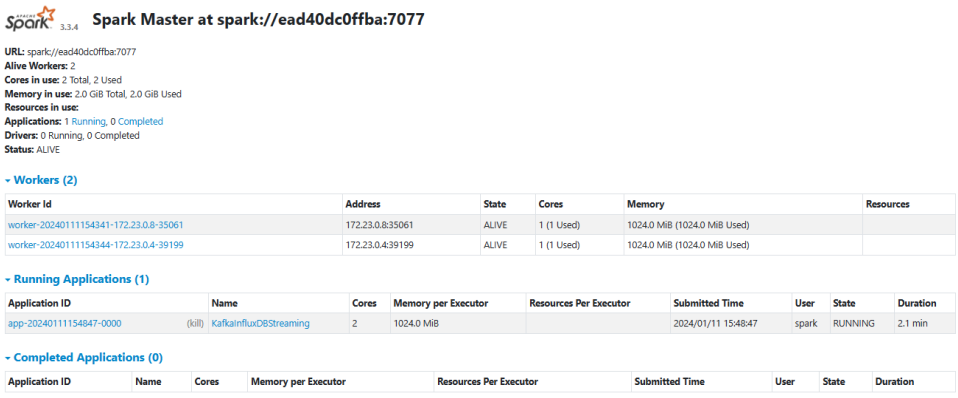
Hình 4.8: Kết quả sau khi xử lý theo lô với 4 mã chứng khoán.

```
spark = SparkSession.builder \
    .appName("Stock Analysis") \
    .config("spark.mongodb.output.uri", "mongodb://root:admin@mongodb:27017/bigdata.stock2024") \
    .getOrCreate()

# Setup the HDFS client
hdfs = pyhdfs.HdfsClient(hosts="namenode:9870", user_name="hdfs")
directory = '/data'
files = hdfs.listdir(directory)
print("Files in '{}':".format(directory), files)
```

Hình 4.9: Một đoạn code trong batch processing.

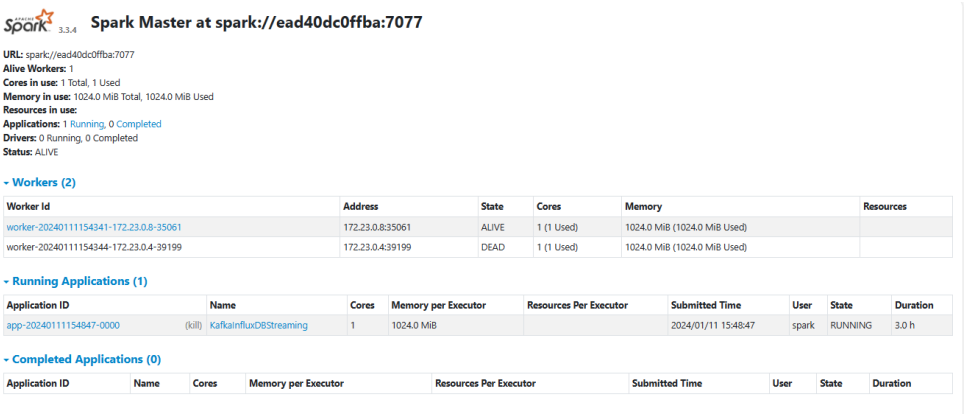
4.9 Thiết lập và sử dụng Multi-Node Spark Cluster



Hình 4.10: Giao diện hiển thị thông tin Spark Master

Từ Spark Master (hình 4.10) có thể thấy Apache Spark hoạt động với 2 worker node.

4.10 Kiểm tra tính sẵn sàng cao của cụm Multi-Node Spark



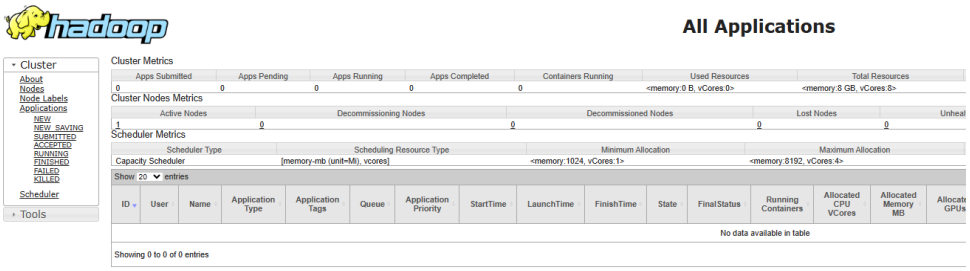
Hình 4.11: Thông tin 1 worker node trong cụm spark ngừng hoạt động

Khi tắt một worker trong cụm multi-node, từ trang thông tin của Spark master và log của consumer ta thấy hệ thống vẫn hoạt động.

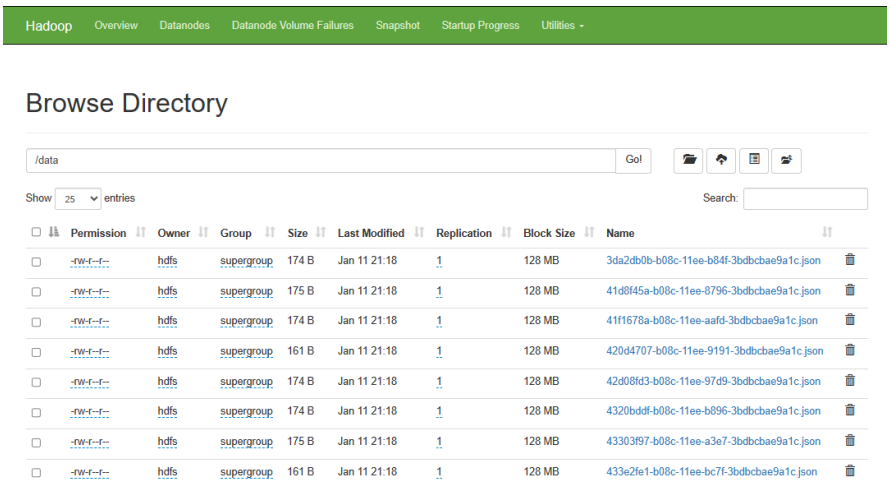
```
01:35:47 ----------------------
01:35:47 Batch processed 3038 done!
01:35:48 {"stock": "ETH-USD", "date": "2024-01-11T18:34:00", "open": 2597.612548828125, "high": 2597.612548828125, "low": 2597.612548828125
2597.612548828125, "volume": 0.0}
01:35:48 ----------------------
01:35:48 {"stock": "BNB-USD", "date": "2024-01-11T18:33:00", "open": 308.2210388183594, "high": 308.2210388183594, "low": 308.2210388183594
308.2210388183594, "volume": 0.0}
01:35:48 ----------------------
01:35:48 {"stock": "USDT-USD", "date": "2024-01-11T18:32:00", "open": 0.9997022151947021, "high": 0.9997022151947021, "low": 0.999702215194
e": 0.9997022151947021, "volume": 0.0}
01:35:48 ----------------------
01:35:48 Batch processed 3039 done!
01:35:53 {"stock": "BTC-USD", "date": "2024-01-11T18:34:00", "open": 46477.45703125, "high": 46477.45703125, "low": 46477.45703125, "close"
03125, "volume": 0.0}
01:35:53 ----------------------
01:35:53 Batch processed 3040 done!
01:35:54 {"stock": "BNB-USD", "date": "2024-01-11T18:33:00", "open": 308.2210388183594, "high": 308.2210388183594, "low": 308.2210388183594
308.2210388183594, "volume": 0.0}
01:35:54 ----------------------
01:35:54 {"stock": "ETH-USD", "date": "2024-01-11T18:34:00", "open": 2597.612548828125, "high": 2597.612548828125, "low": 2597.612548828125
2597.612548828125, "volume": 0.0}
01:35:54 ----------------------
01:35:55 {"stock": "USDT-USD", "date": "2024-01-11T18:32:00", "open": 0.9997022151947021, "high": 0.9997022151947021, "low": 0.999702215194
e": 0.9997022151947021, "volume": 0.0}
01:35:55 ----------------------
01:35:55 Batch processed 3041 done!
```

Hình 4.12: Consumer vẫn hoạt động sau khi 1 worker node ngừng hoạt động.

4.11 Lưu trữ dữ liệu trên Hadoop HDFS



Hình 4.13: Thông tin data node trong Hadoop HDFS



Hình 4.14: Dữ liệu được lưu trữ tại 1 data node trong HDFS

Hình 4.13 và hình 4.14 cho thấy dữ liệu từ Spark streaming chuyển sang Batch Layer đã được lưu trữ tại Hadoop HDFS.

4.12 Cài đặt mô hình máy học thông qua Spark MLlib

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans

# K-Means Model
vec_assembler = VectorAssembler(inputCols=["avg_open", "avg_high", "avg_low", "avg_close", "avg_volume"], outputC
df_kmeans = vec_assembler.transform(basic_stats)

kmeans = KMeans().setK(3).setSeed(1).setFeaturesCol("features")
model = kmeans.fit(df_kmeans)

predictions = model.transform(df_kmeans)
predictions.select("stock", "prediction").show()
```

(a) Thuật toán K-means

```
+--------+----------+
|   stock|prediction|
+--------+----------+
|USDT-USD|         2|
| BNB-USD|         0|
| ETH-USD|         0|
| BTC-USD|         1|
+--------+----------+
```

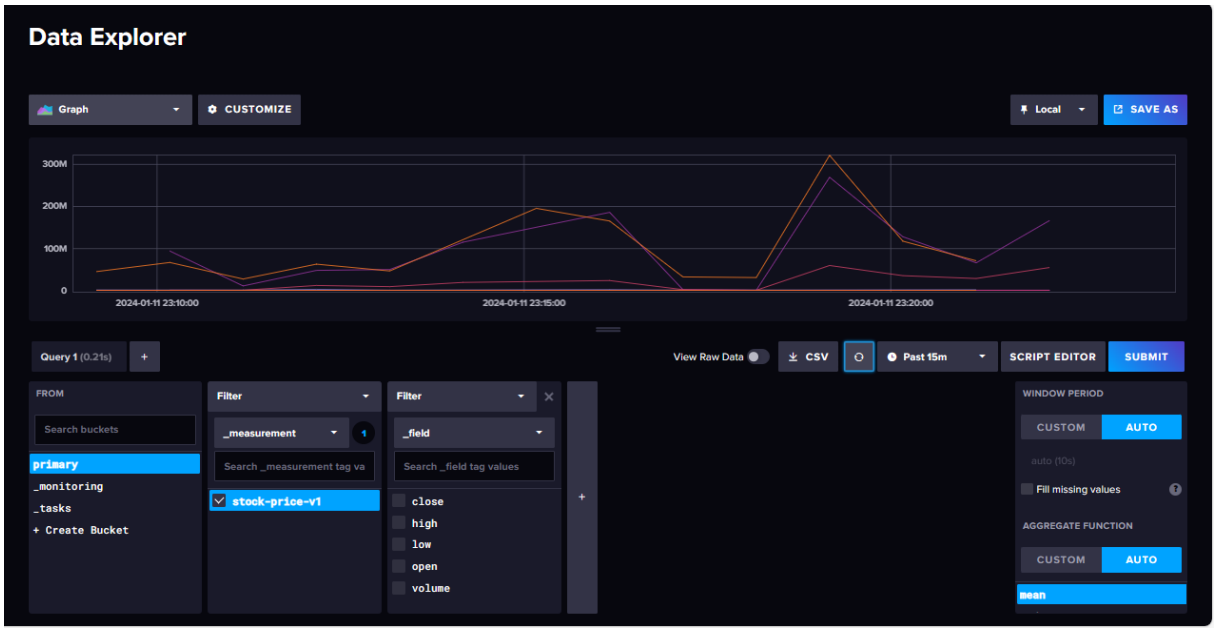
(b) Kết quả phân cụm

Hình 4.15: Cài đặt K-means sử dụng Spark MLlib trên dữ liệu demo 4 mã chứng khoán.

Tại Batch Layer, mô hình K-means được cài đặt thông qua Spark MLlib để phân cụm theo ngày sự thay đổi của các cổ phiếu thành các nhóm.

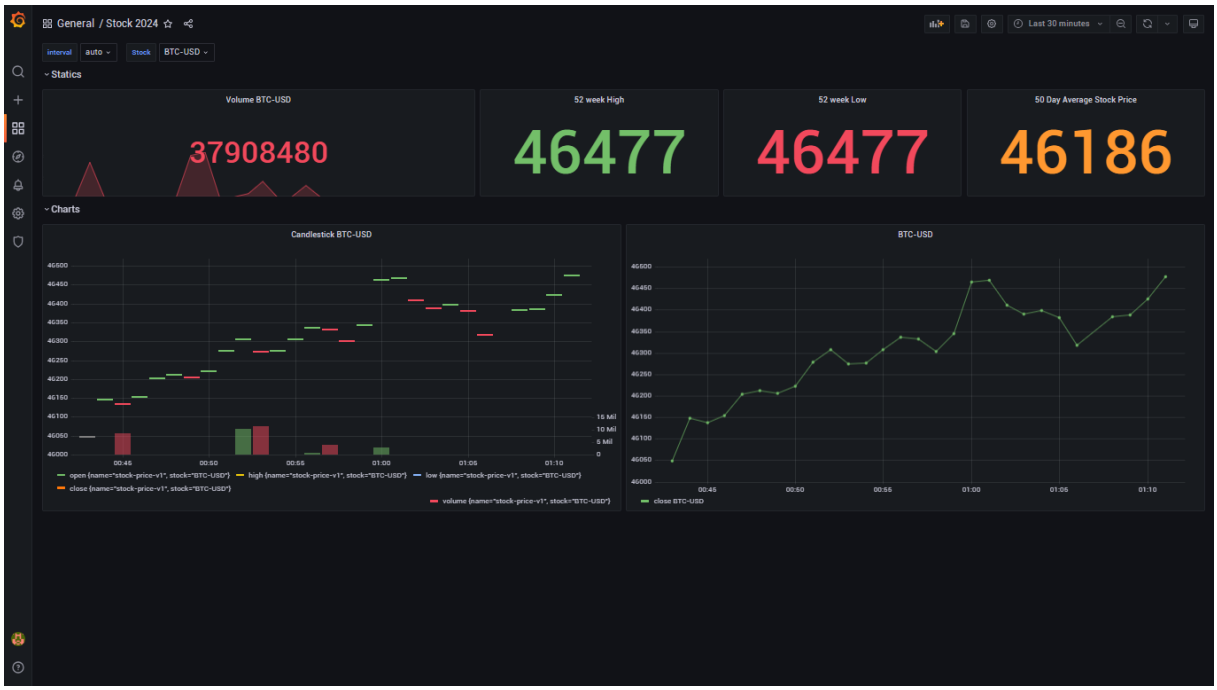
4.13 Triển khai và lưu trữ cơ sở dữ liệu NoSQL MongoDB

4.14 Triển khai và lưu trữ cơ sở dữ liệu time-series InfluxDB



Hình 4.16: Dữ liệu tại bucket primary trong InfluxDB

4.15 Trực quan hoá dữ dạng stream-data thông qua Grafana



Hình 4.17: Trực quan hoá dữ liệu với Grafana

Dữ liệu từ dạng time-series từ InfluxDB, sữ được trực quan hoá theo dạng stream thông qua nền tảng Grafana.

4.16 Thiết kế kiến trúc hệ thống theo mô hình lambda

Việc thiết kế và khởi tạo kiến trúc lambda đã giúp cho hệ thống xử lý dữ liệu phân tán linh hoạt và hiệu quả hơn, có thể đáp ứng được các yêu cầu khác nhau. Điển hình của kiến trúc là 2 nhánh Batch Layer và Speed Layer như hình 3.1. Chi tiết đã được mô tả trong chương 3.

CHƯƠNG 5. KẾT LUẬN

Trong dự án này, nhóm đã thành công trong việc xây dựng một hệ thống xử lý và phân tích dữ liệu chứng khoán thời gian thực. Kết hợp các công cụ hỗ trợ xử lý dữ liệu lớn một cách nhanh chóng và chính xác gồm: Kafka, Apache Spark, Hadoop cùng với các cơ sở dữ liệu NoSQL như MongoDB hay time-series như InfluxDB.

Nhóm cũng đã gặp phải nhiều thách thức, và các vấn đề khác nhau. Tuy nhiên, từ đây nhiều bài học và trải nghiệm trong lưu trữ và xử lý dữ liệu lớn đã được rút ra. Dự án là bước đầu cho việc ứng dụng dữ liệu lớn trong các lĩnh vực tài chính, đặc biệt là trong việc phân tích và dự báo thị trường chứng khoán của nhóm.

Nhóm hy vọng rằng, dự án có thể hoàn thiện hơn trong tương lai. Cùng với các thông tin, phân tích đa dạng, thực tế và hữu ích hơn nữa cho người sử dụng.