

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

\*\*\*\*\*



**BÁO CÁO BÀI TẬP LỚN**

**Đề tài:** Lưu trữ và xử lý dữ liệu tiền ảo thời gian thực

Nhóm 20	
Thành viên	Mã số sinh viên
Lê Hà Phi	20215443
Phạm Tuấn Anh	20215310
Nguyễn Minh Đức	20215354
Lê Anh Thiên	20215483
Nguyễn Lương Phúc	20200475
Trương Việt Hưng	20204834

**Giảng viên giảng dạy :**

TS. Trần Việt Trung

**Học phần :**

Lưu trữ và xử lý dữ liệu lớn

HÀ NỘI, 12/2024

Chương 1 : Giới thiệu bài toán.....	3
1.1    Giới thiệu .....	3
1.2    Vấn đề .....	3
1.3    Phạm vi và giới hạn của project.....	4
Chương 2 : Thu thập dữ liệu .....	5
Chương 3 : Kiến trúc hệ thống.....	6
3.1 Kiến trúc hệ thống lựa chọn.....	6
3.2 Tầng xử lý dữ liệu .....	7
3.2.1 Tầng speed layer.....	7
3.2.2 Tầng batch layer .....	7
3.2.3 Serving layer.....	7
3.3 Data flow .....	8
Chương 4 : Các trải nghiệm với dự án .....	9
4.1 : Thu thập dữ liệu với API.....	9
4.2 : Sử dụng docker để thiết lập các container trên môi trường phân tán .....	10
4.3 : Thiết lập và sử dụng Single/Multinode Kafka Brokers cùng zookeeper .....	10
4.4 : Phân loại và tổ chức dữ liệu thông qua topics Kafka .....	11
4.5 : Kiểm tra sự chịu lỗi và phân bổ tài nguyên của Kafka khi tắt đột ngột các broker .....	11
4.6 : Lưu trữ, xử lý dữ liệu thời gian thực với Spark Streaming.....	12
4.7 : Thực hiện các thao tác truy vấn với Spark SQL .....	13
4.8 : Kiểm tra các running application.....	14
4.9 : Kiểm tra dữ liệu được lưu tại HDFS .....	14
4.10: Triển khai và lưu trữ trên cơ sở dữ liệu InfluxDB.....	15
4.11: Triển khai và lưu trữ dữ liệu trên MongoDB.....	15
4.12 : Trực quan hóa dữ liệu thông qua Grafana .....	16
4.13 : Một số lưu ý trong lúc làm dự án.....	16
Chương 5: Kết luận .....	17

# Chương 1 : Giới thiệu bài toán

## 1.1 Giới thiệu

Trong con mắt của những nhà đầu tư tiền ảo, việc có thể thấy được các thông tin về tiền ảo và cập nhật dữ liệu ngay tức thì là vô cùng quan trọng. Điều này không chỉ giúp các nhà đầu tư có được cái nhìn sâu sắc và đúng lúc về thị trường để đưa ra các quyết định, chiến lược giao dịch đúng đắn. 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 tiền điện tử trong thời gian thực.

## 1.2 Vấn đề

Giá tiền 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 tiền ảo 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 tiền ảo 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 tiền ảo, đặc biệt là trong môi trường giao dịch nhanh và biến động cao của ngày nay.

Do đó, đề tài khá là phù hợp với thời đại thông tin mới là thứ quan trọng nhất, quyết định mọi hướng đi của sự việc. Bên cạnh đó, lượng dữ liệu về tiền mã hóa thực sự là khổng lồ, nên phù hợp với các hướng đi áp dụng với bigdata.

### 1.3 Phạm vi và giới hạn của project

Ở đề tài này , em chỉ thực hiện việc lưu trữ, xử lý dữ liệu với độ trễ xấp xỉ thời gian thực. Với mong muốn hiểu được cách xây dựng một hệ thống lưu trữ và xử lý dữ liệu lớn, đúng với nội dung môn học này . Bên cạnh đó có áp dụng một số mô hình và xây dựng các dashboard để quản lý và trực quan hóa dữ liệu.

## Chương 2 : Thu thập dữ liệu

Dữ liệu mà lần này bọn em chọn để thu thập và xử lý là nguồn dữ liệu các đồng tiền điện tử được crawl trực tiếp từ trang web **coindesk.com**.

Dữ liệu này bao gồm các thông tin về tiền điện tử như

- Name : tên đồng tiền điện tử
- Date : Thời gian của bản ghi theo thời gian thực
- Open : Giá mở cửa
- High : Giá cao nhất trong ngày
- Low : Giá thấp nhất trong ngày
- Close : Giá đóng cửa

Thông tin về các đồng tiền này sẽ được lưu lại và cập nhật cho hệ thống lưu trữ cũng như là phân tích dữ liệu.

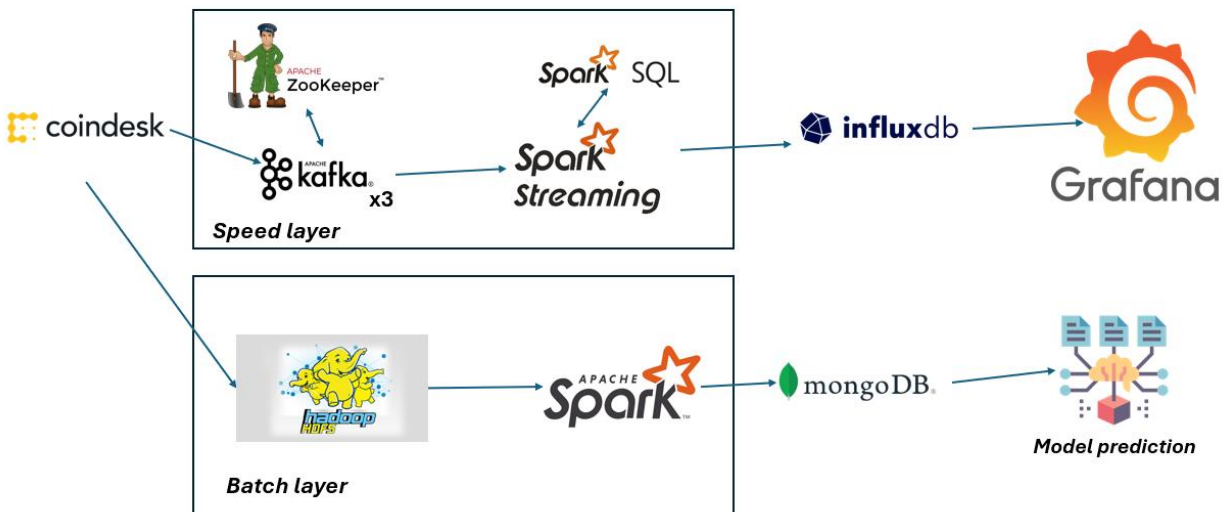
## Chương 3 : Kiến trúc hệ thống

Lưu trữ và phân tích tiền ảo thời gian thực đòi hỏi một hệ thống có khả năng xử lý dữ liệu lớn một cách nhanh chóng và chính xác.

Phần này em sẽ mô tả chi tiết về kiến trúc hệ thống và các công nghệ được sử dụng.

### 3.1 Kiến trúc hệ thống lựa chọn

Ở đề tài này bọn em sẽ sử dụng kiến trúc hệ thống Lambda với 2 phần riêng biệt là xử lý theo lô và xử lý theo luồng, mỗi tầng sẽ làm việc với mục đích riêng để đưa ra những thông tin hữu ích về dữ liệu thu thập được.



## 3.2 Tầng xử lý dữ liệu

### 3.2.1 Tầng speed layer

Tầng speed xử lý dữ liệu thời gian thực sử dụng SparkStreaming và SparkSQL 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 Tầng batch layer

Tầng batch xử lý dữ liệu lớn thông qua Hadoop HDFS và Apache Spark để thực hiện các tính toán phức tạp với lượng dữ liệu lớn được thu thập theo chu kì. Kết quả sau khi xử lý sẽ được lưu trữ hoặc áp dụng các mô hình để cho ra các thông tin hữu ích.

### 3.2.3 Serving layer

Tầng này sẽ tổng hợp dữ liệu từ 2 tầng trên là xử lý theo lô và theo luồng, để xây dựng các dashboard hoặc cung cấp các truy vấn với dữ liệu cuối cùng.

Dữ liệu sẽ được crawl và gửi đến kafka thông qua producer, đây là thành phần quan trọng của hệ thống, mỗi khi gửi thành công sẽ báo rằng đã gửi message đến topic mà khởi tạo.

Các container giao tiếp với nhau qua cùng một mạng broker-kafka.

<input type="checkbox"/>		coinstock	-	-	-	122.51%	6.23GB / 137.05C	81.89%	0B / 0B	<input type="checkbox"/>	:	
<input type="checkbox"/>		zoo1	910c1b9208a7	confluentin	2181:2181	0.16%	79.68MB / 7.61GI	1.02%	0B / 0B	<input type="checkbox"/>	:	
<input type="checkbox"/>		spark-worker:	38f767d51be9	docker.io/b	8082:8082	26.52%	426.2MB / 7.61GI	5.47%	0B / 0B	<input type="checkbox"/>	:	
<input type="checkbox"/>		spark-worker	cb30e70f2d38	docker.io/b	8081:8081	20.48%	480.2MB / 7.61GI	6.16%	0B / 0B	<input type="checkbox"/>	:	
<input type="checkbox"/>		spark_speed	ea87abdc1789	my-spark-ir	-	49.13%	722.3MB / 7.61GI	9.26%	0B / 0B	<input type="checkbox"/>	:	
<input type="checkbox"/>		spark_master	f5d40e23ced7	docker.io/b	7077:7077  80...	0.1%	180.5MB / 7.61GI	2.32%	0B / 0B	<input type="checkbox"/>	:	
<input type="checkbox"/>		spark_batch	82c2f8d98e25	my-spark-ir	-	0.59%	479.5MB / 7.61GI	6.15%	0B / 0B	<input type="checkbox"/>	:	
<input type="checkbox"/>		resourceman	a59ee7b9d509	apache/har	8088:8088	0.82%	330.7MB / 7.61GI	4.24%	0B / 0B	<input type="checkbox"/>	:	
<input type="checkbox"/>		nodemanager	0ab4d0c6d909	apache/har	-	0.6%	297.5MB / 7.61GI	3.82%	0B / 0B	<input type="checkbox"/>	:	

Showing 93 items



### 3.4 Chi tiết triển khai

Hướng dẫn cài đặt và các bước triển khai cũng như sourcode đều ở đây:

[https://github.com/pinocchioo1412/Bigdata\\_test/tree/main](https://github.com/pinocchioo1412/Bigdata_test/tree/main)

## Chương 4 : Các trải nghiệm với dự án

### 4.1 : Thu thập dữ liệu với API

```
from requests import get

URL = 'https://production.api.coindesk.com/v2/tb/price/ticker?assets='

def default(args):
    if len(args) == 0:
        raise Exception("Missing argument...")

    final_url = URL + args
    result = get(final_url)
    json_data = result.json()

    individual_symbols = args.split(',')

    results = []


    for s in individual_symbols:
        coin_data = json_data['data'][s]
        results.append({
            "iso": coin_data['iso'],
            "name": coin_data['name'],
            "date_time": datetime.now().strftime("%Y-%m-%d %H:%M:%S%z"),
            "current_price": coin_data['ohlc']['c'],
            "open": coin_data['ohlc']['o'],
            "high": coin_data['ohlc']['h'],
            "low": coin_data['ohlc']['l'],
            "close": coin_data['ohlc']['c']
        })
```

## 4.2 : Sử dụng docker để thiết lập các container trên môi trường phân tán

Các container được cấu hình thông qua docker-compose.yml, và giao tiếp cùng một mạng:

<div>spark_speed</div> <div>my-spark-image</div> <div></div> <div></div> <div></div>	ting
<div>spark_batch</div> <div>my-spark-image</div> <div></div> <div></div> <div></div>	2024-12-10 17:33:07 datanode1   2024-12-10 10:33:07 INFO webhdfs:147 - 172.18.0.19 PUT /webhdfs/v1/data/24eb6021-b6e2-11ef-9160-67334d3d7c8d.json?op=CREATE&user.name=hdfs&namenoderpcaddress=namenode:8020&createflag=&createparent=true&overwrite=false 201
<div>spark_master</div> <div>docker.io/bitnami/spark:3.5.0</div> <div>7077:7077</div> <div>8080...</div> <div></div> <div></div> <div></div>	2024-12-10 17:33:07 datanode1   2024-12-10 10:33:07 INFO DataNode:750 - Receiving BP-1071224186-172.18.0.10-1733824349950:blk_1073741991_1167 src: /172.18.0.14:56554 dest: /172.18.0.14:9866
<div>kafdrop</div> <div>obsidiandynamics/kafdrop:0.11.0</div> <div>9100:9000</div> <div></div> <div></div> <div></div>	2024-12-10 17:33:07 datanode1   2024-12-10 10:33:07 INFO clienttrace:1550 - src: /172.18.0.14:56554, dest: /172.18.0.14:9866, bytes: 108, op: HDFS_WRITE, cliID: DFSCliet_NONMAPREDUCE_469913641_108, offset: 0, srvID: 8335748c-726d-4902-bfcd-d7b61a96822d, blockid: BP-1071224186-172.18.0.10-1733824349950:blk_1073741991_1167, duration(ns): 2380738
<div>datanode2</div> <div>apache/hadoop:3.7.0</div> <div></div> <div></div> <div></div>	2024-12-10 17:33:07 datanode1   2024-12-10 10:33:07 INFO DataNode:1523 - PacketResponder: BP-1071224186-172.18.0.10-1733824349950:blk_1073741992_1168 src: /172.18.0.14:56562 dest: /172.18.0.14:9866
<div>datanode1</div> <div>apache/hadoop:3.7.0</div> <div></div> <div></div> <div></div>	2024-12-10 17:33:07 datanode1   2024-12-10 10:33:07 INFO DataNode:1523 - PacketResponder: BP-1071224186-172.18.0.10-1733824349950:blk_1073741992_1168, duration(ns): 2068873
<div>kafka3</div> <div>apache/kafka:3.6.0</div> <div></div> <div></div> <div></div>	2024-12-10 17:33:07 datanode1   2024-12-10 10:33:07 INFO DataNode:750 - Receiving BP-1071224186-172.18.0.10-1733824349950:blk_1073741993_1169 src: /172.18.0.14:56566 dest: /172.18.0.14:9866

## 4.3 : Thiết lập và sử dụng Single/Multinode Kafka Brokers cùng zookeeper



Kafdrop

4.0.2 [2024-07-09T13:11:53.509Z]

## Kafka Cluster Overview

Bootstrap servers	kafka1:19092,kafka2:19093,kafka3:19094
Total topics	2
Total partitions	53
Total preferred partition leader	100%
Total under-replicated partitions	0

### Brokers

ID	Host	Port	Rack	Controller	Number of partitions (% of total)
1	kafka1	19092	-	Yes	17 (32%)
2	kafka2	19093	-	No	18 (34%)
3	kafka3	19094	-	No	18 (34%)

### Topics ACLs

Name	Partitions	% Preferred	# Under-replicated	Custom Config
_consumer_offsets	50	100%	0	Yes
coin_price	3	100%	0	No

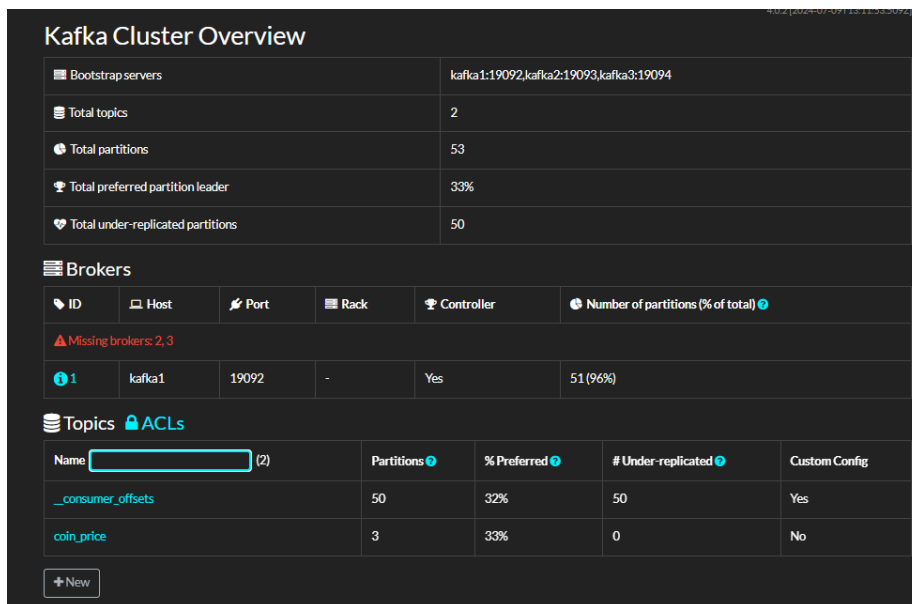
+ New

Cấu hình được thiết lập với 3 kafka broker cùng 2 topic.

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

Với 2 topic là coin\_price phụ trách gửi dữ liệu đến kafka và consumer\_offsets để lưu dữ liệu vào HDFS và các cơ sở dữ liệu khác.

#### 4.5 : Kiểm tra sự chịu lỗi và phân bổ tài nguyên của Kafka khi tắt đột ngột các broker



**Kafka Cluster Overview**

Bootstrap servers	kafka1:19092,kafka2:19093,kafka3:19094
Total topics	2
Total partitions	53
Total preferred partition leader	33%
Total under-replicated partitions	50

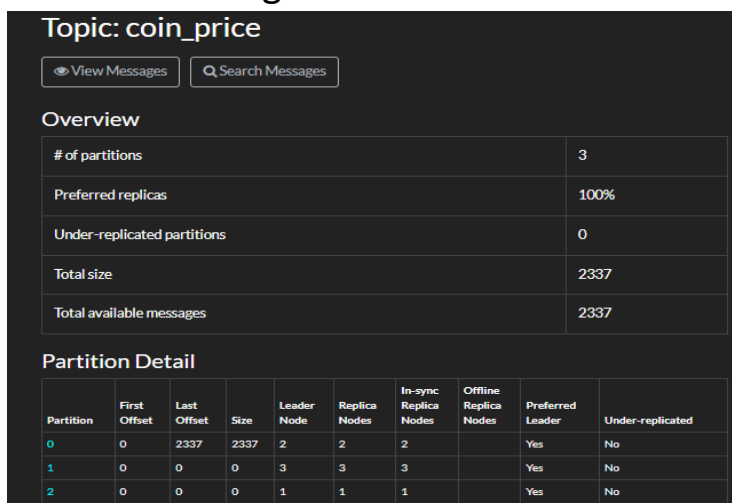
**Brokers**

ID	Host	Port	Rack	Controller	Number of partitions (% of total)
Missing brokers: 2, 3					
1	kafka1	19092	-	Yes	51 (96%)

**Topics**

Name	Partitions	% Preferred	# Under-replicated	Custom Config
__consumer_offsets	50	32%	50	Yes
coin_price	3	33%	0	No

Khi tắt thử broker số 2 và 3 thì hệ thống hoạt động với kafka1, tuy nhiên không chiếm toàn bộ phân vùng, và kafka1 hoàn toàn đảm nhận việc trao đổi message.



**Topic: coin\_price**

View Messages | Search Messages

**Overview**

# of partitions	3
Preferred replicas	100%
Under-replicated partitions	0
Total size	2337
Total available messages	2337

**Partition Detail**

Partition	First Offset	Last Offset	Size	Leader Node	Replica Nodes	In-sync Replica Nodes	Offline Replica Nodes	Preferred Leader	Under-replicated
0	0	2337	2337	2	2	2		Yes	No
1	0	0	0	3	3	3		Yes	No
2	0	0	0	1	1	1		Yes	No

Tuy nhiên 3 broker không hoàn toàn được chia đều việc quản lý dữ liệu ngay lập tức như trước nữa.

ID	Host	Port	Rack	Controller	Number of partitions (% of total)
1	kafka1	19092	-	Yes	34 (64%)
2	kafka2	19093	-	No	18 (34%)
3	kafka3	19094	-	No	1 (2%)

Mà sau một thời gian thì sẽ lại hoạt động bình thường.

ID	Host	Port	Rack	Controller	Number of partitions (% of total)
1	kafka1	19092	-	Yes	17 (32%)
2	kafka2	19093	-	No	18 (34%)
3	kafka3	19094	-	No	18 (34%)

## 4.6 : Lưu trữ, xử lý dữ liệu thời gian thực với Spark Streaming

Dữ liệu được truyền theo các mini\_batch

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

hdfs = pyhdfs.HdfsClient(hosts="namenode:9870", user_name="hdfs")
directory = '/data'
if not hdfs.exists(directory):
    hdfs.mkdirs(directory)
files = hdfs.listdir(directory)
print("Files in '{}':".format(directory), files)

You, 17 hours ago • first commit

schema = StructType([
    StructField("iso", StringType(), True),
    StructField("name", StringType(), True),
    StructField("current_price", DoubleType(), True),
    StructField("open", DoubleType(), True),
    StructField("high", DoubleType(), True),
    StructField("low", DoubleType(), True),
    StructField("close", DoubleType(), True),
    StructField("date_time", StringType(), True)
])
```

```

2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["ZETA", "ZetaChain", 0.6987285674, 0.8131568553, 0.842029986, 0.6410541658, 0.6987285674, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["BNB", "BNB", 698.294903942, 716.6305182075, 731.8833650137, 648.2519400995, 698.294903942, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["SHIB", "Shiba Inu", 2.7107e-05, 2.98887e-05, 3.0781e-05, 2.44911e-05, 2.7107e-05, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["SOL", "Solana", 215.63485097831253, 226.48330636584672, 231.6110008261816, 205.29242573747086, 215.63485097831253, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["TON", "Toncoin", 5.8947086826, 6.4156724105, 6.5689951122, 5.254084421, 5.8947086826, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["WIF", "dogwifhat", 2.9082462952, 3.3555189429, 3.542758514, 2.6726797108, 2.9082462952, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["OKB", "OKB", 54.1671937682, 56.3609508122, 57.9622172741, 16.1094719048, 54.1671937682, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["PEPE", "Pepe", 2.51101e-05, 2.46916e-05, 2.83863e-05, 2.22989e-05, 2.51101e-05, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["RUNE", "THORChain", 6.3519202363, 6.938402111, 7.1669484471, 5.7433900304, 6.3519202363, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:03 ["AVA", "TravaLa.com", 0.7038959891, 0.7478178882, 0.775015809, 0.6589918794, 0.7038959891, "2024-12-10 18:46:48"]
2024-12-10 18:47:03 -----
2024-12-10 18:47:04 ["INJ", "Injective", 28.0231841096, 30.4928687215, 31.3647509532, 24.7032598689, 28.0231841096, "2024-12-10 18:46:48"]
2024-12-10 18:47:04 -----
2024-12-10 18:47:04 ["SUN", "Sun", 0.0245480594, 0.0283943698, 0.0291309853, 0.0206715484, 0.0245480594, "2024-12-10 18:46:48"]
2024-12-10 18:47:04 -----
2024-12-10 18:47:04 Batch processed 1 done!
2024-12-10 18:47:04 24/12/10 11:47:04 INFO CheckpointFileManager: Writing atomically to file:/tmp/temprary-e7dd7c8f-d61a-4767-9de8-07d1a17cfe3e/commits/1

```

## 4.7 : Thực hiện các thao tác truy vấn với Spark SQL

```

def create_dataframe_from_file(file_path):
    try:
        file_content = hdfs.open(file_path).read().decode('utf-8')
        data = json.loads(file_content)
        return spark.createDataFrame([data], schema)
    except Exception as e:
        print("Failed to read '{}': {}".format(file_path, e))
        return None

df = spark.createDataFrame([], schema)

for file in files:
    file_path = "{}/{}/{}".format(directory, file)
    file_df = create_dataframe_from_file(file_path)
    if file_df:
        df = df.unionByName(file_df)
df = df.dropDuplicates()

basic_stats_t = df.groupBy("iso").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.stddev("close").alias("std_dev_close"),
    F.max("high").alias("historical_high"),
    F.min("low").alias("historical_low")
)

```

4.8 : Kiểm tra các running application

URL: spark/rd9fc3a807a87077

Alive Workers: 2

Cores in use: 2 Total, 2 Used

Memory in use: 2.0 GiB Total, 2.0 GiB Used

Resources in use:

Applications: 2 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20241210122616-172.18.0.5-43959	172.18.0.5:43959	ALIVE	1 (1 Used)	1024.0 MiB (1024.0 MiB Used)	
worker-20241210122616-172.18.0.6-33253	172.18.0.6:33253	ALIVE	1 (1 Used)	1024.0 MiB (1024.0 MiB Used)	

Running Applications (2)

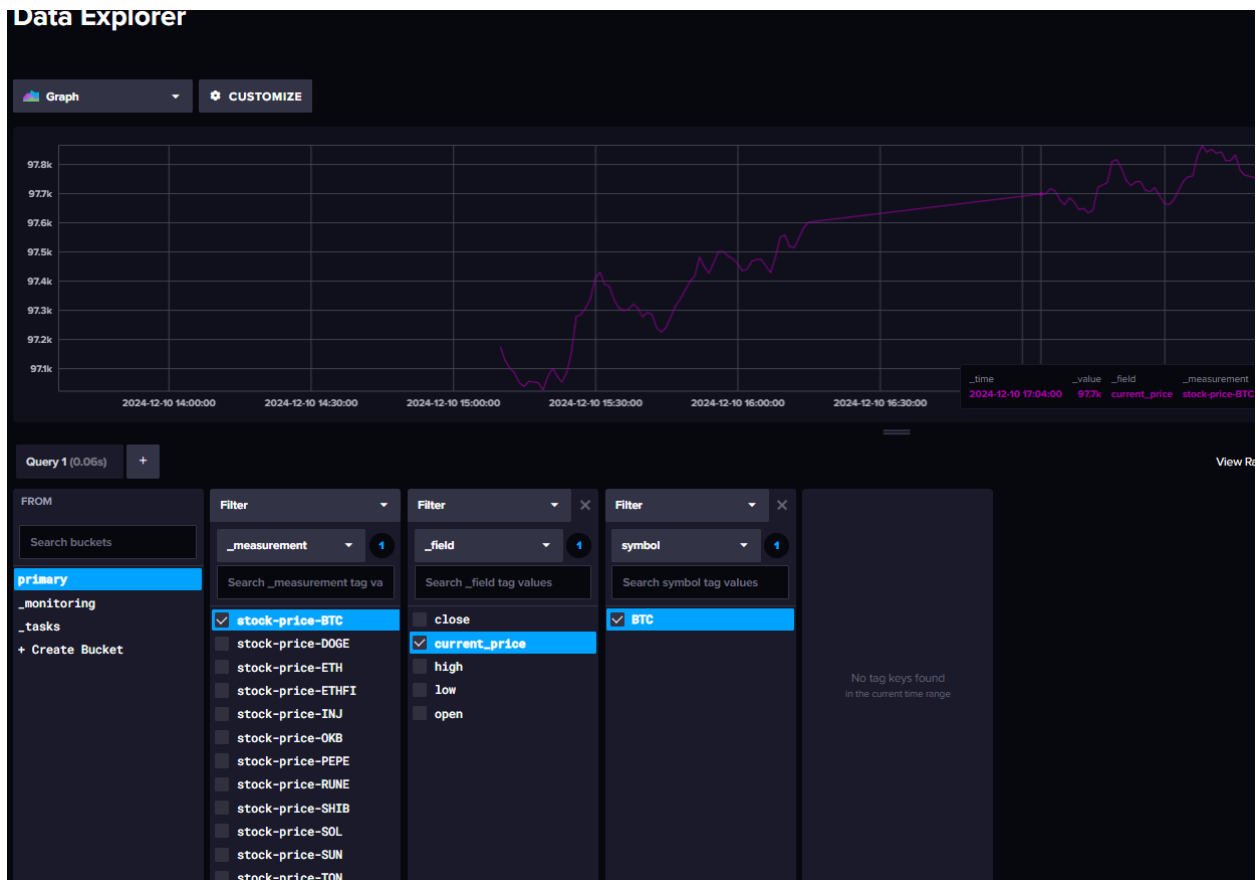
Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241210122807-0001	(kill) Crypto Dependency Analysis	0	1024.0 MiB		2024/12/10 12:28:07	spark	WAITING	20 s
app-20241210122758-0000	(kill) KafkaInfluxDBStreaming	2	1024.0 MiB		2024/12/10 12:27:58	spark	RUNNING	28 s

Có thể thấy 2 applications đang hoạt động với 2 spark\_worker.

4.9 : Kiểm tra dữ liệu được lưu tại HDFS

<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	137 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">083dc198-b6e2-11ef-a876-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	137 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">0f5cd2f4-b6e2-11ef-95d0-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	114 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">0f6931ec-b6e2-11ef-b726-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	113 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">0f738b25-b6e2-11ef-80a8-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	113 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">1229e02e-b6e2-11ef-843a-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	117 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">1238d1e2-b6e2-11ef-844b-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	108 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">1243a95b-b6e2-11ef-9077-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	137 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">124d1411-b6e2-11ef-b50d-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	110 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">12573d66-b6e2-11ef-843f-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	112 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">1262752e-b6e2-11ef-bd60-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	112 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">126e276d-b6e2-11ef-be92-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	104 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">127ced91-b6e2-11ef-aec2-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	114 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">1286335f-b6e2-11ef-82d2-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	113 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">12949478-b6e2-11ef-ab0f-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	118 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">129e10de-b6e2-11ef-9780-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	107 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">12a8a2af-b6e2-11ef-8c73-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	137 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">12b24464-b6e2-11ef-a9a7-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	137 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">12bb1c2d-b6e2-11ef-8bd2-67334d3d7c8d.json</a>	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">hdfs</a>	<a href="#">supergroup</a>	114 B	Dec 10 17:32	<a href="#">2</a>	128 MB	<a href="#">12c37ccc-b6e2-11ef-9e06-67334d3d7c8d.json</a>	

## 4.10: Triển khai và lưu trữ trên cơ sở dữ liệu InfluxDB



## 4.11: Triển khai và lưu trữ dữ liệu trên MongoDB

Dữ liệu được lưu trữ trên mongodb sẽ được phục vụ cho các tính toán khác sau này.

[+ ADD DATA](#)
[EXPORT DATA](#)
[UPDATE](#)
[DELETE](#)

```

_id: ObjectId('6758378a2aeca281ac9c3301')
symbol: "BTC"
iso: "BTC"
name: "Bitcoin"
date_time: "2024-12-10 19:43:54"
current_price: 97246.55578486719
open: 98403.67534750569
high: 100438.29776509697
low: 94421.53922461752
close: 97246.55578486719

```

```

_id: ObjectId('6758378b2aeca281ac9c3302')
symbol: "BTC"
iso: "BTC"
name: "Bitcoin"
date_time: "2024-12-10 19:43:54"
current_price: 97246.55578486719
open: 98403.67534750569
high: 100438.29776509697
low: 94421.53922461752
close: 97246.55578486719

```

## 4.12 : Trực quan hóa dữ liệu thông qua Grafana



## 4.13 : Một số lưu ý trong lúc làm dự án

### Mô tả vấn đề

- Context: Hệ thống thu thập dữ liệu thời gian thực từ nhiều nguồn và lưu trữ vào cơ sở dữ liệu thông qua các container trong thời gian dài
- Thách thức: Lượng dữ liệu lớn nhanh chóng vượt quá khả năng lưu trữ và xử lý của hệ thống, dẫn đến tình trạng cạn tài nguyên (RAM, CPU) sau 3-4 giờ hoạt động.



- Impact: Hệ thống trở nên chậm chạp, không thể xử lý thêm dữ liệu và thậm chí bị "crash", làm gián đoạn pipeline.

Giải pháp đã sử dụng : thêm một broker để phân chia việc truyền dữ liệu hợp lý hơn

## 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 tiền ảo 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 tiền điện tử 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.