ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO LAB 2: Advanced MapReduce & Spark Structured APIs

Môn học: Nhập môn dữ liệu lớn

Mã lớp: CQ2022/21

GVHD1: Nguyễn Ngọc Thảo

GVHD2: Lê Ngọc Thành

Trợ Giảng: Huỳnh Lâm Hải Đăng

Thành viên: Nguyễn Đình Trí MSSV: 22120384

Vũ Hoàng Nhật Trường MSSV: 22120398

Nguyễn Anh Tường MSSV: 22120412

Nguyễn Hoàng Vũ MSSV: 22120439

Thành phố Hồ Chí Minh, ngày 07 tháng 04 năm 2025

Mục lục

SECTION 1: Phân công thành viên nhóm

MSSV	Họ và tên	Phân công công việc
22120384	Nguyễn Đình Trí	• Tìm hiểu và viết mã nguồn cho Task_2.3
22120398	Vũ Hoàng Nhật Trường	• Tìm hiểu và viết mã nguồn cho Task_2.1
22120412	Nguyễn Anh Tường	• Tìm hiểu và hướng dẫn nhóm cài đặt Spark trên cả 2 hệ điều hành Linux và Windows • Kiểm tra lại kết quả và thảo luận sửa đổi với các thành viên • Viết báo cáo
22120439	Nguyễn Hoàng Vũ	• Tìm hiểu và viết mã nguồn cho Task_2.2

SECTION 2: Báo cáo chi tiết về bài làm

1. Tìm hiểu và hướng dẫn cài đặt Spark

1.1 Cài đặt python 3.10:

```
python3 –version # Kiểm tra phiên bản đang sử dụng

sudo apt update # Cài đặt các gói phụ thuộc
sudo apt install software-properties-common –y

#Thêm kho lưu trữ chứa các phiên bản Python cũ
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update

# Cài đặt Python 3.10
sudo apt install python3.10 -y

# Kiểm tra cài đặt
python3.10 --version
ls /usr/bin/python3*  # Kiểm tra các phiên bản Python hiện có
```

1.2 Đảm bảo có java 8/11/17:

```
java –version # kiểm tra phiên bản
```

- Vì các bạn đã tải Hadoop trước đó nên vì thế sẽ không cần kiểm tra cũng được.

1.3 Tải và cài Spark (chọn phiên bản từ 3.5 trở lên):

- Tải và giải nén spark: https://spark.apache.org/downloads.html
- Cách số 2 (sử dụng cmd / terminal):

```
wget <link tải spark>
tar -xvzf <file tải về>
```

Sau khi tải thì kiểm tra lại:

```
java -version
python3.10 --version
cd Downloads
ls
```

Di chuyển Spark vào thư mục /opt/

```
sudo mv spark-3.5.5-bin-hadoop3 /opt/spark
ls /opt/
```

Cấu hình biến môi trường:

```
nano ~/.bashrc
```

Thêm vào cuối file:

```
export SPARK_HOME=/opt/spark

export PATH=$SPARK_HOME/bin:$PATH
```

Lưu file lại:

```
source ~/.bashrc
```

Chạy thử SPARK:

```
spark-shell
```

1.4 Cài PySpark:

Kiểm tra PySpark: pyspark

Cấu hình lại để PySpark dùng python 3.10: Trong file .bashrc ta tiến hành thêm dòng lệnh bên dưới rồi lưu lại.

```
export PYSPARK_PYTHON=python3.10
```

Kiểm tra lại PySpark: pyspark

Gõ thử đoạn sau để kiểm tra:

```
rdd = sc.parallelize([1, 2, 3, 4])
rdd.map(lambda x: x * 2).collect()
```

1.5 Tiếp tục với VSCODE:

- Cài Jupyter *(Vì code file .py debug khó, nên dùng Jupyter Notebook cho dễ.)*
- Chọn Python 3.10 trong VS Code
    - Nhấn Ctrl + Shift + P
    - Gõ Python: Select Interpreter
    - Chọn Python 3.10.16 (/usr/bin/python3.10)
- Cài đặt PySpark:

```
sudo apt install python3.10-venv -y
python3.10 -m ensurepip --upgrade
python3.10 -m pip install --upgrade pip
python3.10 -m pip install pyspark
```

- Test code với Spark:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
   .appName("MySimpleApp") \
   .getOrCreate()

sc = spark.sparkContext

rdd = sc.parallelize([1, 2, 3, 4])
print(rdd.map(lambda x: x * 2).collect())

spark.stop()
```

1.6 Cài đặt SPARK cho Windows: tham khảo tại: https://www.youtube.com/@MaiDE-uq7ws/playlists

1.7 Cài đặt Hadoop lại: tham khảo tại report trước của nhóm tại: https://docs.google.com/document/d/1ZXqjoJi-t8AmOcWNji6r6OVRT9xqIreo1PX4GKplPFY/edit?fbclid=IwZXh0bgNhZW0CMTEAAR5FFCBg1odGlgC_0Bc-Bk_ZTXHUkrWgLCckUBeXmBp0W-G9AarJPU9mYcnKIA_aem_7oDYmWbhbM7OoGPO0tdnKg&tab=t.0

2. Tìm hiểu và viết mã nguồn cho Task_2.1

- **Ngôn ngữ lập trình:** Python.
- **Công cụ sử dụng:** Visual Studio Code.
- **Môi trường chạy Linux:** Ubuntu 24.04.1 LTS.
- **Yêu cầu bài toán:** Đề bài yêu cầu sử dụng **thuật toán cửa sổ trượt 3 ngày** để tính tổng số lượng bán ra cho mỗi loại hàng trong từng ngày.

Phân tích:

+ Giả sử chúng ta có ngày bắt đầu và ngày kết thúc được ghi trong file .csv, lần lượt gọi là **start_date** và **end_date**.

+ Khi áp dụng thuật toán tính doanh thu trượt trong 3 ngày, khoảng thời gian được sử dụng để tính doanh thu sẽ kéo dài từ **start_date** đến **end_date + 2**.

+ **Lý do:** **Doanh thu của một mặt hàng tại ngày N** được tính bằng tổng doanh thu trong 3 ngày: **N, N-1 và N-2**. Với những ngày đầu như **start_date** và **start_date + 1**, do không có đủ dữ liệu của các ngày trước, *ta chỉ tính tổng dựa trên những ngày có sẵn* (ví dụ: **start_date** chỉ tính cho riêng ngày đó; **start_date + 1** sẽ tính từ **start_date** đến **start_date + 1**).

+ **Cài đặt thuật toán:** Với **mỗi dòng tương ứng với một ngày** trong file .csv, ta sẽ sinh ra **3 cặp keys** tương ứng với *"ngày hôm nay", "ngày mai"* và *"ngày mốt".* Cuối cùng thì ta sẽ lấy **tổng giá trị của các keys** giống nhau lại, sau đó sắp xếp keys theo **thứ tự tăng dần.**

+ Suy ra thuật toán trên sẽ tạo ra hai ngày dư **end_date + 1** và **end_date + 2** (là các ngày không có trong file .csv ban đầu).

- Do đề bài yêu cầu sử dụng cửa sổ trượt với **bước nhảy là 1 ngày**, tức là toàn bộ các ngày từ **start_date** đến **end_date** đều **phải có mặt** trong báo cáo, nên khi áp dụng thuật toán trên sẽ phát sinh một vấn đề:

  + Cụ thể, nếu file .csv ban đầu **bị thiếu dữ liệu ở một số ngày giữa chừng**, thuật toán có thể **bỏ qua những ngày đó khi trượt cửa sổ**, dẫn đến **vi phạm điều kiện tiên quyết** là cửa sổ phải trượt tịnh tiến mỗi ngày một lần.

  + Để khắc phục vấn đề này, em sẽ viết một file script có tên **getDatesAndCategories.py,** với chức năng **trích xuất toàn bộ tất cả các ngày** trong khoảng **từ start_date đến end_date**, cũng như **toàn bộ loại mặt hàng** có trong dữ liệu, sau đó lưu vào file **dates.txt** và **categories.txt**. Việc này nhằm đảm bảo không bỏ sót bất kỳ ngày nào hoặc loại mặt hàng nào, kể cả khi những thông tin đó không được liệt kê đầy đủ trong file .csv ban đầu.

getDatesAndCategories.py > ...

```python
from datetime import datetime, timedelta

DATE_FORMAT = "%m-%d-%y"

input_file = "asr.csv"
dates_set = set()
categories_set = set()

with open(input_file, 'r', encoding='utf-8') as file:
    next(file)  # Bỏ dòng tiêu đề
    for line in file:
        parts = line.strip().split(',')

        date_str = parts[2].strip()
        category = parts[9].strip()

        date_obj = datetime.strptime(date_str, DATE_FORMAT)

        dates_set.add(date_obj)
        categories_set.add(category)

# Tìm min - max
min_date = min(dates_set)
max_date = max(dates_set) + timedelta(days=2)

# Sinh toàn bộ các ngày từ min đến max
with open("dates.txt", "w", encoding='utf-8') as f_dates:
    current = min_date
    while current <= max_date:
        f_dates.write(current.strftime(DATE_FORMAT) + "\n")
        current += timedelta(days=1)

# Ghi danh sách category
with open("categories.txt", "w", encoding='utf-8') as f_cate:
    for cate in sorted(categories_set):
        f_cate.write(cate + "\n")
```

- Tiến hành chạy file **getDatesAndCategories.py**:

```
hadoop@DESKTOP-ABGKMNK:~$ python3 getDatesAndCategories.py
```

dates.txt

```
31/03/2022
01/04/2022
02/04/2022
03/04/2022
04/04/2022
05/04/2022
06/04/2022
07/04/2022
08/04/2022
09/04/2022
10/04/2022
11/04/2022
12/04/2022
13/04/2022
14/04/2022
15/04/2022
16/04/2022
17/04/2022
18/04/2022
19/04/2022
20/04/2022
```

categories.txt

```
Blouse
Bottom
Dupatta
Ethnic Dress
Saree
Set
Top
Western Dress
kurta
```

(Nội dung trong 2 file dates.txt và categories.txt)

- Kích hoạt môi trường ảo để xài được thư viện MRJob:

```
hadoop@DESKTOP-ABGKMNK:~$ source ~/.local/share/pipx/venvs/mrjob/bin/activate
(mrjob) hadoop@DESKTOP-ABGKMNK:~$
```

- Tiến hành cấu hình file MapReduce. Khai báo các biến ban đầu:

```
class calculateRevenue(MRJob):
        OUTPUT_PROTOCOL = RawValueProtocol
        DATE_FORMAT_INPUT = "%m-%d-%y"
        DATE_FORMAT_OUTPUT = "%d/%m/%Y"

        all_dates = set()
        all_categories = set()
```

- Ta tạo hàm **mapper_init**: Hàm này dùng để lấy thông tin từ 2 file dates.txt và categories.txt, sau đó lưu vào 2 set đã tạo từ trước.

```
def mapper_init(self):
        with open("/home/hadoop/dates.txt", 'r') as f:
                for line in f:
                        date_obj = datetime.strptime(line.strip(), self.DATE_FORMAT_INPUT)
                        self.all_dates.add(date_obj.strftime(self.DATE_FORMAT_OUTPUT))

        with open("/home/hadoop/categories.txt", 'r') as f:
                for line in f:
                        self.all_categories.add(line.strip())
```

- Tiến hành viết hàm **mapper** chính. Hàm này sẽ đọc tuần tự từng dòng trong file **asr.csv**. Sau khi lấy ra những thông tin cần thiết: **Date, Status, Category, Amount**, với **mỗi dòng** ta sẽ có **3 cặp {keys – value}** – **{(Date, Category) – Amount}** được tạo ra.
- Lưu ý: Kiểm tra trước xem **Status** có nằm trong diện **hoàn trả hàng** hay không *(đã được liệt kê các trường hợp trong mảng **refund**)*, nếu có thì đặt lại Amount = 0.
- Hai trường hợp đặc biệt như: **Shipped – Damaged** và **Shipped – Lost in Transit**, các lỗi này do bên vận chuyển gây ra nên họ mới là người cần phải bồi thường, do đó vẫn tính vào doanh thu của người bán như bình thường.

```
def mapper(self, _, line):
        if "index" in line:  # Không tính dòng đầu tiên
                return

        columns = line.strip().split(",")
        date = columns[2].strip() # Thuộc tính Date
        status = columns[3].strip() # Thuộc tính Status
        category = columns[9].strip() # Thuộc tính Category
        amount = columns[15].strip() # Thuộc tính Amount - Số tiền thanh toán

        if amount == "":
                amount = 0.0

        refund = ["Cancelled", "Shipped - Rejected by Buyer", "Shipped - Returned to Seller", "Shipped - Returning to Seller"]
        if status in refund:
                amount = 0.0

        date_obj = datetime.strptime(date, self.DATE_FORMAT_INPUT)

        # Với mỗi dòng, phát ra 3 cặp key – value
        # Mỗi key là tuple (window_date, category) ứng với cửa sổ trượt 3 ngày
        for offset in range(3):
                window_date = date_obj + timedelta(days=offset)
                window_date_str = window_date.strftime(self.DATE_FORMAT_OUTPUT)
                yield (window_date_str, category), float(amount)
```

- Tạo hàm **mapper_final**: Hàm này dùng để tạo **toàn bộ các keys {date, category}** từ **start_date** đến **end_date**. Ta sẽ đặt số lượng bán ra của chúng bằng 0.

```
def mapper_final(self):
        for date in self.all_dates:
                for category in self.all_categories:
                        yield (date, category), 0.0
```

- Lấy tổng giá trị các keys giống nhau trong hàm **reducer**:

```
def reducer(self, keys, values):
        total_revenue = sum(values)
        date, category = keys
        yield None, (date, category, total_revenue)
```

- Tuy nhiên, ta phải làm thêm 1 bước nữa là **sắp xếp các keys theo thứ tự tăng dần** của **sorted_date**, sau đó tới **category**, do đó ta viết thêm hàm **reducer_final_sorting**:

```
def reducer_final_sorting(self, _, records):
        sorted_records = sorted(records, key=lambda x: datetime.strptime(x[0], self.DATE_FORMAT_OUTPUT))
        for date, category, revenue in sorted_records:
                yield None, f"{date},{category},{revenue:.2f}"
```

- Viết lại hàm **steps** để MRJob để thực thi các hàm **mapper**, **reducer** theo đúng thứ tự:

```
def steps(self):
        return [
                MRStep(
                mapper_init=self.mapper_init,
                mapper=self.mapper,
                mapper_final=self.mapper_final,
                reducer=self.reducer),

                MRStep(reducer=self.reducer_final_sorting)]
```

- Chạy file MapReduce, sau đó lưu kết quả vào file **output.csv**. Ta cũng cần viết thêm **tên các cột** vào trong file .csv đó.

```
(mrjob) hadoop@DESKTOP-ABGKMNK:~$ python3 mapreduce_lab2.py asr.csv > output.csv
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/mapreduce_lab2.hadoop.20250408.090126.192467
Running step 1 of 2...
Running step 2 of 2...
job output is in /tmp/mapreduce_lab2.hadoop.20250408.090126.192467/output
Streaming final output from /tmp/mapreduce_lab2.hadoop.20250408.090126.192467/output...
Removing temp directory /tmp/mapreduce_lab2.hadoop.20250408.090126.192467...
(mrjob) hadoop@DESKTOP-ABGKMNK:~$ sed -i '1ireport_date,category,revenue' output.csv
```

- Kiểm tra file **output.csv**:

```
(mrjob) hadoop@DESKTOP-ABGKMNK:~$ head -n 10 output.csv
report_date,category,revenue
31/03/2022,Blouse,280.00
31/03/2022,Bottom,0.00
31/03/2022,Dupatta,0.00
31/03/2022,Ethnic Dress,0.00
31/03/2022,Saree,0.00
31/03/2022,Set,50905.00
31/03/2022,Top,4511.00
31/03/2022,Western Dress,5479.00
31/03/2022,kurta,33008.00
```

- **Chương trình thực thi thành công!**

3. Tìm hiểu và viết mã nguồn cho Task_2.2

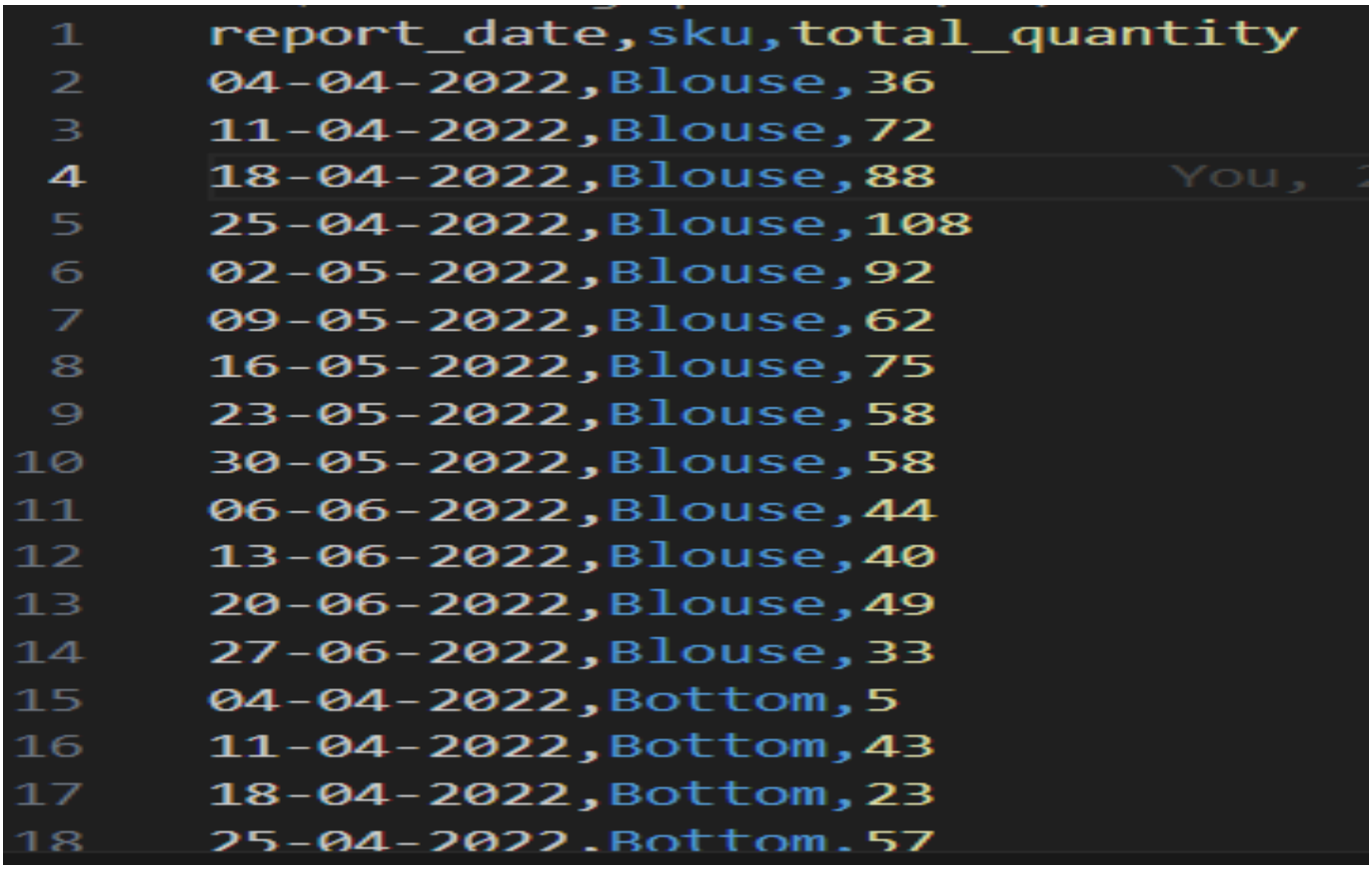
- Bài toán yêu cầu đếm số lượng sản phẩm được bán đi trong trong mỗi tuần của mỗi loại sản phẩm bắt đầu tình từ ngày thứ 2. Ví dụ: Ta cần tính số lượng được bán đi của loại sản phẩm **"Set"** trong khoảng thời gian **"25-04-2022" – "01-05-2022"** để báo cáo cho ngày **"02-05-2022"**.
- Đầu tiên ta đọc dữ liệu từ file **"asr.csv"** (Khi chạy code trên máy của mình cần thay đổi đường dẫn cho phù hợp) ra và bỏ đi header.
- Chọn ra các cột cần thiết.
    - Đầu tiên ta lấy cột **"Category"** để phù hợp với output mẫu mà thầy mong muốn thay vì cột **"SKU"** gốc.
    - Tiếp theo, ta cần lấy các sản phẩm được **"Shipped"** nên sẽ có 2 cột **"Status"** hoặc **"Courier Status".** Ta có thể thấy như ảnh 3.1 một vài giá trị trong Status bắt đầu với **"Shipped"** nhưng **"Courier Status"** là **"Trống"** => Ta lấy ra cột **"Status"** cho phần trạng thái thay vì cột **"Courier Status".**
    - Cuối cùng, ta lấy 2 cột còn lại là Date và Quantity

Status	Fulfilment	Sales Channel	ship-service-level	Style	SKU	Category	Size	ASIN	Courier Status
Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	SET278	SET278-KR-NP-S	Set	S	B0983F2ZJW	
Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	SET288	SET288-KR-NP-XXL	Set	XXL	B09M6TQTGV	
Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	JNE3730	JNE3730-KR-L	kurta	L	B09HLZKJ6G	
Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	MEN5028	MEN5028-KR-XL	kurta	XL	B08YYTL6BX	
Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	MEN5022	MEN5022-KR-M	kurta	M	B08YYV8NZL	
Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	J0151	J0151-KR-A-XL	kurta	XL	B08N19YB92	
Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	PJNE3252	PJNE3252-KR-N-6XL	kurta	6XL	B09LD24932	
Shipped - Returned to Seller	Merchant	Amazon.in	Standard	JNE3798	JNE3798-KR-A-S	Western Dress	S	B09TH4P8P7	
Shipped - Returned to Seller	Merchant	Amazon.in	Standard	SET401	SET401-KR-NP-L	Set	L	B09VC5ZGD8	
Shipped - Delivered to Buyer	Merchant	Amazon.in	Standard	NW012	NW012-TP-PJ-XL	Set	XL	B0922TK8CF	
Shipped - Returned to Seller	Merchant	Amazon.in	Standard	JNE3679	JNE3679-TU-L	Top	L	B0943J4SL3	

Ảnh 3.1

- Định nghĩa một hàm để chuyển đổi chuỗi ngày tháng trong file **"asr.csv"** từ **"m-d-Y"** thành đối tượng datetime để dễ xử lý.
- Vì để tránh giá trị None gây lỗi trong quá trình ép kiểu nên ta quyết định:
    - Loại bỏ các hàng có **Date = None**
    - Thay thế giá trị của các hàng có **Quantity = None** thành giá trị trung bình của cột
- Lấy ra 3 cột **Category, Date, Quantity** mà có giá trị của Status bắt đầu bằng **"Shipped"**
- Để tính toán số lượng mỗi loại sản phẩm bán được trong từng tuần, ta cần lọc ra các ngày thứ 2.

- Sau đó ta xác định khoảng thời gian cần tính là **[Monday – 7; Monday)** cho mỗi tuần
- Với mỗi ngày thứ 2 duyệt qua ta lấy số lượng bán được của các ngày trong tuần trước đó rồi cộng lại theo từng loại sản phẩm.
- Thêm kết quả vừa tính cho tuần vừa rồi vào list kết quả.
- Ghi kết quả vào file **"output.csv"**



Ảnh 3.2. Output

4. Tìm hiểu và viết mã nguồn cho Task_2.3

- Yêu cầu bài toán: Cho một tập các hình chữ nhật dưới dạng danh sách đỉnh (vertices), hãy tìm ra tất cả các **cặp hình có giao nhau** thực sự (tức là diện tích vùng giao > 0).
- Dữ liệu lưu trong file .parquet có dạng: <shape_id;vertices>. Trong đó:
    - **vertices** là danh sách 4 đỉnh biểu diễn một hình chữ nhật (theo thứ tự ngược chiều kim đồng hồ).
    - **shape_id**, nếu bị thiếu (NULL), cần được tạo lại để định danh các hình.
- Công cụ và thư viện sử dụng:
    - **Apache Spark**: Xử lý dữ liệu lớn (đọc file, tính toán phân tán).
    - **PySpark**: API Python của Spark.
    - **Shapely**: Thư viện Python xử lý hình học 2D (Polygon, Intersection...).
- Ý tưởng và thuật toán:
    - Bước 1: Đọc và gán ID cho các hình
        + Đọc dữ liệu từ file .parquet.
        + Gán cột shape_id mới bằng monotonically_increasing_id() để phân biệt từng hình.

```
# Đọc dữ liệu từ file parquet
input_path = "shapes.parquet"
raw_df = spark.read.parquet(input_path)

# Gán shape_id nếu bị NULL
df = raw_df.withColumn("shape_id", monotonically_increasing_id())
```

    - Bước 2: Chuyển vertices thành Polygon
        + Mỗi vertices là 4 điểm → chuyển thành Polygon bằng thư viện Shapely.
        + Giúp dễ tính toán giao nhau và diện tích hình.

```python
# Thu thập dữ liệu về driver
shape_data = df.select("shape_id", "vertices").collect()

# Chuyển đổi vertices thành Polygon
polygon_list = []
for row in shape_data:
    shape_id = row["shape_id"]
    vertices = [(x, y) for x, y in row["vertices"]]
    polygon = Polygon(vertices)
    polygon_list.append((shape_id, polygon))
```

- Bước 3: So sánh từng cặp hình
  + Duyệt tất cả cặp hình không trùng lặp.
  + Nếu vùng giao giữa hai hình có diện tích > 0, ghi lại cặp đó.

```python
# Tìm các cặp polygon giao nhau
overlap_pairs = []
n_polygons = len(polygon_list)

for i in range(n_polygons):
    id1, poly1 = polygon_list[i]

    for j in range(i + 1, n_polygons):
        id2, poly2 = polygon_list[j]
        intersection = poly1.intersection(poly2)
        if  intersection.area > 0:
            overlap_pairs.append((int(id1), int(id2)))
```

- Bước 4: Tạo DataFrame kết quả
  + Chuyển danh sách cặp (shape_1, shape_2) thành Spark DataFrame.
  + Sắp xếp kết quả theo thứ tự tăng dần theo shape_1, tiếp theo là shape_2.

```
# Tạo Spark DataFrame từ kết quả
columns = ["shape_1", "shape_2"]
result_df = spark.createDataFrame(overlap_pairs, columns)

# Sắp xếp kết quả
result_df = result_df.orderBy("shape_1", "shape_2")
```

Bước 5: Kết quả được lưu về file output.csv:

```
# Sao chép dữ liệu từ Spark DataFrame
rows = result_df.collect()

columns = ["shape_1", "shape_2"]
output_path = "./output.csv"

with open(output_path, "w") as file:
    # Ghi dòng tiêu đề
    file.write(",".join(columns) + "\n")

    # Ghi từng dòng dữ liệu
    for row in rows:
        values = [str(row[col]) for col in columns]
        file.write(",".join(values) + "\n")
```

Tổng cộng có 6562 dòng. Hiển thị 10 dòng kết quả đầu tiên:

```
shape_1,shape_2
0,41
0,143
0,198
0,430
0,473
0,490
0,496
0,605
0,637
0,729
```