TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GTVT

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----o0o-----**

****

**BÁO CÁO BÀI TẬP LỚN**

**BIG DATA**

**Đề tài số 14:**

**MapReduce, Apache Spark,**

**Sử dụng file vdv\_olympics, qg\_noc.csv để thực hiện yêu cầu**

|  |  |
| --- | --- |
| **GIÁO VIÊN HƯỚNG DẪN:** | **KHUẤT THỊ NGỌC ÁNH** |
| **SINH VIÊN THỰC HIỆN:** | **NGUYỄN HỮU TÚ** |
|  | **VŨ ĐỨC ANH**  **TRỊNH KIM CHI**  **TRỊNH XUÂN DƯƠNG** |
| **NHÓM:** | **1** |
| **LỚP:** | **72DCTT24** |

**HÀ NỘI, 2024**

**MỤC LỤC**

[Câu 1. Tại sao MapReduce xử lý dữ liệu chậm hơn các nền tảng xử lý dữ liệu khác? 3](#_Toc163662809)

[Câu 2. Trình bày hạn chế của Apache Spark 4](#_Toc163662810)

[Câu 3. Sử dụng file vdv\_olympics, qg\_noc.csv 4](#_Toc163662811)

[3.1. Schema của bộ dữ liệu, hiển thị số bản ghi của bộ dữ liệu. 5](#_Toc163662812)

[3.2. Đưa ra thông tin về các vận động viên Việt Nam tham gia Olympic trong thập kỷ 2000 6](#_Toc163662813)

[3.3. Đưa ra thông tin về 7 quốc gia có số lượng vận đông viên đông nhất tham gia thế vận hội 2016 6](#_Toc163662814)

[3.4. Số lượng vận động viên tham dự 5 kỳ thế vận hội gần nhất 7](#_Toc163662815)

[3.5. Số lượng vận động viên nam tại mỗi kỳ thế vận hội trong thế kỷ 21 8](#_Toc163662816)

[3.6. Tổng số vận động viên tại mỗi kỳ thế vận hội trong thập kỷ 1990 của Nga. 8](#_Toc163662817)

[3.7. Cho biết chiều cao tối thiểu, trung bình và tối đa của mỗi quốc gia tham gia thế vận hội mùa đông và sắp xếp theo thứ tự giảm dần. 9](#_Toc163662818)

[3.8. Các quốc gia không có thông tin về quốc gia trong qg\_noc 10](#_Toc163662819)

[3.9. Liệt kê danh sách 10 quốc gia giành nhiều huy chương vàng nhất thế vận hội mùa hè 2012. 11](#_Toc163662820)

[3.10. Liệt kê 10 vận động viên có giành nhiều huy chương vàng nhất thế kỷ 20 12](#_Toc163662821)

[Câu 4: Sử dụng dữ liệu các vận động viên tại thế vận hội 2016 (file đính kèm) để thực hành streaming các vận động viên giành huy chương vàng. 13](#_Toc163662822)

## Câu 1. Tại sao MapReduce xử lý dữ liệu chậm hơn các nền tảng xử lý dữ liệu khác?

MapReduce là một mô hình lập trình và triển khai để xử lý và tạo ra các tập dữ liệu lớn với thuật toán phân tán, song song trên một cụm máy tính.

Cách thức hoạt động:

1. Map:

* + Chia nhỏ dữ liệu đầu vào thành các cặp key-value.
  + Áp dụng một hàm map cho từng cặp key-value để tạo ra các tập dữ liệu trung gian.

2. Reduce:

* + Nhóm các cặp key-value có cùng key lại với nhau.
  + Áp dụng một hàm reduce cho từng nhóm key-value để tạo ra kết quả cuối cùng.

Chính vì cách thức hoạt động “Chia để trị” trên làm cho MapReduce có nhược điểm là chậm hơn các nền tảng xử lý dữ liệu khác:

* + Khởi động và kết thúc tác vụ: Mỗi tác vụ map và reduce trong MapReduce đều phải khởi động và kết thúc riêng biệt. Quá trình này có thể tốn thời gian, đặc biệt là cho các tác vụ ngắn.
  + Lỗi và phục hồi: Khi một tác vụ map hoặc reduce gặp lỗi, MapReduce phải khởi động lại toàn bộ tác vụ. Quá trình này có thể tốn thời gian và ảnh hưởng đến hiệu suất tổng thể.
  + MapReduce không phù hợp với tất cả các loại dữ liệu.Nó có thể xử lý dữ liệu dạng văn bản hiệu quả, nhưng có thể gặp khó khăn với dữ liệu dạng ảnh hoặc video.
  + MapReduce có thể mở rộng tốt theo chiều ngang (thêm nhiều nút vào cụm máy tính). Tuy nhiên, khả năng mở rộng theo chiều dọc (tăng tài nguyên cho mỗi nút) có thể gặp hạn chế.
  + Giao tiếp: MapReduce sử dụng mạng để truyền dữ liệu giữa các máy tính, có thể chậm hơn so với các phương thức khác như bộ nhớ dùng chung. Giảm hiệu suất xử lý dữ liệu thời gian thực

## Câu 2. Trình bày hạn chế của Apache Spark

Phụ thuộc vào hệ thống lưu trữ: Spark không có hệ thống lưu trữ dữ liệu riêng.

Tốn nhiều tài nguyên:

* + Xử lý dữ liệu trong bộ nhớ, yêu cầu nhiều RAM.
  + Chi phí cao hơn so với các hệ thống xử lý dữ liệu khác.
  + Không phù hợp cho các tập dữ liệu nhỏ hoặc các cụm máy tính có cấu hình thấp.

Khó khăn trong việc gỡ lỗi: Khó khăn trong việc xác định và sửa lỗi do tính chất phân tán của Spark. Thiếu các công cụ hỗ trợ debug chuyên dụng.

Spark không được thiết kế cho các tác vụ xử lý dữ liệu thời gian thực.

Không phù hợp cho tất cả các loại dữ liệu: Khả năng xử lý dữ liệu nhỏ, không có cấu trúc hoặc thời gian thực hạn chế

## Câu 3. Sử dụng file vdv\_olympics, qg\_noc.csv

Phần khai báo sẽ dùng cho câu 3 (khi kết thúc code sẽ đóng spark spark.stop()):

from pyspark.sql import SparkSession

from pyspark.sql.functions import \*

spark = SparkSession.builder \

    .appName("Big Data") \

    .getOrCreate()

vdv\_olympics\_df = spark.read.csv(r'D:\Y3\BigData\BigData\DeTaiSo14\Data\vdv\_olympics.csv', header=True, inferSchema=True)

qg\_noc\_df = spark.read.csv(r'D:\Y3\BigData\BigData\DeTaiSo14\Data\qg\_noc.csv', header=True, inferSchema=True)

### 3.1. Schema của bộ dữ liệu, hiển thị số bản ghi của bộ dữ liệu.

print("Schema của bộ dữ liệu vdv\_olympics.csv:")

vdv\_olympics\_df.printSchema()

print("Số bản ghi trong bộ dữ liệu:", vdv\_olympics\_df.count())

print("==========================================\n")

print("Schema của bộ dữ liệu qg\_noc.csv:")

qg\_noc\_df.printSchema()

print("Số bản ghi trong bộ dữ liệu:", qg\_noc\_df.count())

Kết quả:

A screen shot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

### 3.2. Đưa ra thông tin về các vận động viên Việt Nam tham gia Olympic trong thập kỷ 2000

filtered\_df = vdv\_olympics\_df.filter((vdv\_olympics\_df["NOC"] == "VIE") & (vdv\_olympics\_df["Year"].cast("int").between(2000, 2009)))

filtered\_df.show(filtered\_df.count(), False)

vdv\_olympics\_df.select()

Kết quả:

A screen shot of a computer screen

Description automatically generated

### 3.3. Đưa ra thông tin về 7 quốc gia có số lượng vận đông viên đông nhất tham gia thế vận hội 2016

year\_filtered\_df = vdv\_olympics\_df.filter((vdv\_olympics\_df["Year"] == "2016"))

country\_count\_df = year\_filtered\_df.groupBy("NOC").agg(countDistinct("ID").alias("AthleteCount"))

top\_7\_countries = country\_count\_df.orderBy(country\_count\_df["AthleteCount"].desc()).limit(7)

top\_7\_countries.show(top\_7\_countries.count(), False)

Kết quả:

A screen shot of a computer

Description automatically generated

### 3.4. Số lượng vận động viên tham dự 5 kỳ thế vận hội gần nhất

athlete\_counts\_by\_year = vdv\_olympics\_df.groupBy("Year").agg(countDistinct("ID").alias("AthleteCount"))

recent\_olympics = athlete\_counts\_by\_year.orderBy("Year", ascending=False).limit(5)

recent\_olympics.show()

Kết quả:

A black screen with white text

Description automatically generated

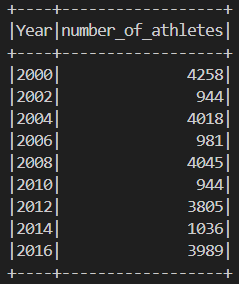
### 3.5. Số lượng vận động viên nam tại mỗi kỳ thế vận hội trong thế kỷ 21

filtered\_df = vdv\_olympics\_df.filter((vdv\_olympics\_df["Year"] >= 2000) & (vdv\_olympics\_df["Sex"] == "M"))

male\_athlete\_count\_by\_year = filtered\_df.groupBy("Year").agg(countDistinct("ID").alias("number\_of\_athletes")).orderBy("Year")

male\_athlete\_count\_by\_year.show()

Kết quả:



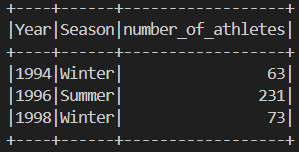
### 3.6. Tổng số vận động viên tại mỗi kỳ thế vận hội trong thập kỷ 1990 của Nga.

filtered\_df = vdv\_olympics\_df.filter((vdv\_olympics\_df["Year"].cast("int").between(1990, 1999)) & (vdv\_olympics\_df["NOC"] == "RUS"))

athlete\_count\_by\_year = filtered\_df.groupby("Year", "Season").agg(countDistinct("ID").alias("number\_of\_athletes")).orderBy("Year")

athlete\_count\_by\_year.show()

Kết quả:



### 3.7. Cho biết chiều cao tối thiểu, trung bình và tối đa của mỗi quốc gia tham gia thế vận hội mùa đông và sắp xếp theo thứ tự giảm dần.

joined\_df = vdv\_olympics\_df.join(qg\_noc\_df, "NOC", "left").select(vdv\_olympics\_df["\*"], qg\_noc\_df["Country"].alias("Country")).drop("NOC")

winter\_df = joined\_df.filter((joined\_df["Season"] == "Winter") & (joined\_df["Height"] != "NA"))

height\_stats\_by\_noc = winter\_df.groupBy("Country").agg(

    min("Height").alias("min\_height"),

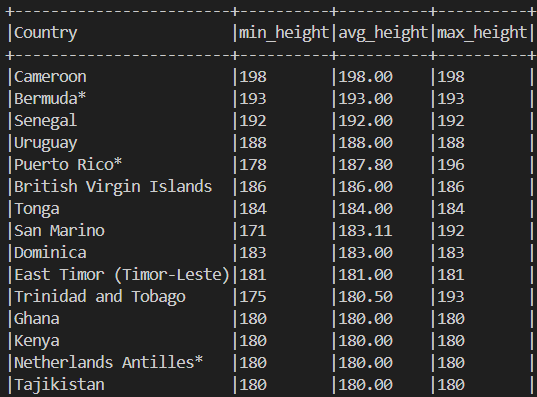
    format\_number(avg("Height"), 2).alias("avg\_height"),

    max("Height").alias("max\_height")

).orderBy(desc("avg\_height"))

height\_stats\_by\_noc.show(height\_stats\_by\_noc.count(), False)

Kết quả:



### 3.8. Các quốc gia không có thông tin về quốc gia trong qg\_noc

missing\_country\_info = qg\_noc\_df.filter(qg\_noc\_df['Population'].isNull() | qg\_noc\_df['GDP per Capita'].isNull())

print("Tên các quốc gia không có thông tin về quốc gia trong qg\_noc:")

missing\_country\_info.drop("Population").drop("GDP per Capita").show()

Kết quả:

A screen shot of a computer program

Description automatically generated

### 3.9. Liệt kê danh sách 10 quốc gia giành nhiều huy chương vàng nhất thế vận hội mùa hè 2012.

top\_10\_gold\_medal\_countries\_2012 = vdv\_olympics\_df.filter((col('Year') == 2012) &

(col('Season') == 'Summer') & (col('Medal') == 'Gold'))

.groupBy('NOC').agg(count('\*').alias('gold\_medals\_count'))

.orderBy('gold\_medals\_count', ascending=False).limit(10)

print("Danh sách 10 quốc gia giành nhiều huy chương vàng nhất")

top\_10\_gold\_medal\_countries\_2012.show()

Kết quả:

A black screen with white text

Description automatically generated

### 3.10. Liệt kê 10 vận động viên có giành nhiều huy chương vàng nhất thế kỷ 20

top\_10\_athletes\_20th\_century = vdv\_olympics\_df.filter((col('Year') >= 1900) & (col('Year') <= 2000) & (col('Medal') == 'Gold')).groupBy('Name').agg(count('\*')

.alias('gold\_medals\_count')).orderBy('gold\_medals\_count', ascending=False).limit(10)

print("Danh sách 10 vận động viên giành nhiều huy chương vàng nhất thế kỷ 20:")

top\_10\_athletes\_20th\_century.show()

Kết quả:

A screen shot of a computer

Description automatically generated

## Câu 4: Sử dụng dữ liệu các vận động viên tại thế vận hội 2016 (file đính kèm) để thực hành streaming các vận động viên giành huy chương vàng.

from pyspark.sql.types import StructType, StructField, StringType

from pyspark.streaming import StreamingContext

from pyspark.sql import SparkSession

# Define schema cho DataFrame

schema = StructType([

    StructField("ID", StringType(), True),

    StructField("Name", StringType(), True),

    StructField("Sex", StringType(), True),

    StructField("Age", StringType(), True),

    StructField("Height", StringType(), True),

    StructField("Weight", StringType(), True),

    StructField("Team", StringType(), True),

    StructField("NOC", StringType(), True),

    StructField("Games", StringType(), True),

    StructField("Year", StringType(), True),

    StructField("Season", StringType(), True),

    StructField("City", StringType(), True),

    StructField("Sport", StringType(), True),

    StructField("Event", StringType(), True),

    StructField("Medal", StringType(), True)

])

spark = SparkSession.builder.appName("Streaming").getOrCreate()

streaming\_df = spark.readStream.schema(schema).option("header", "true").format("csv")\

.option("path", "D:\Y3\BigData\BigData\DeTaiSo14\Streaming").load()

transformed\_df = streaming\_df.filter("Medal = 'Gold'")

selected\_df = transformed\_df.select("ID", "Name", "Medal")

query = selected\_df.writeStream.outputMode("append").format("console").start()

query.awaitTermination()

Kết quả:

A screen shot of a computer

Description automatically generated