TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VIỆN TRÍ TUỆ NHÂN TẠO

-----*-----**

BÁO CÁO MÔN HỌC KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN

ĐỀ TÀI

Finding Average Temperature of Each Year using Hadoop-HDFS

Giảng viên hướng dẫn: TS. Trần Hồng Việt

Nhóm sinh viên thực hiện

1. Thái Nguyễn Hoàng Bách - 22022672
2. Quách Đắc Chính - 22022518
3. Vũ Minh Đức - 22022587
4. Đào Duy Hưng - 22022589

HÀ NỘI, 12/2024

MỞ ĐẦU

Big Data ngày nay đã trở thành một phần không thể thiếu trong việc xử lý và phân tích dữ liệu lớn khi xu hướng sử dụng công nghệ ngày càng lớn đối với tất cả mọi người. Với sự phát triển của các công nghệ hỗ trợ như Hadoop, việc xử lý các tập dữ liệu lớn không chỉ trở nên khả thi mà còn hiệu quả. Điều đó sẽ được ứng dụng trong rất nhiều lĩnh vực như đời sống, giáo dục, …

Chúng em chọn đề tài **"Finding Average Temperature using Hadoop-HDFS"** nhằm mục đích áp dụng thực tiễn công nghệ MapReduce để phân tích nhiệt độ từ dữ liệu khí tượng và cung cấp thông tin về biến động thời tiết qua các năm.

Báo cáo gồm 4 chương:

Chương 1: Tổng quan về dữ liệu lớn và Hadoop MapReduce

Chương 2: Cách triển khai phân tích dữ liệu khí tượng.

Chương 3: Đánh giá kết quả và hiệu quả xử lý.

Chương 4: Kết luận và hướng phát triển.

Chương 1: Tổng quan về dữ liệu lớn và Hadoop MapReduce

1.1 Định nghĩa của Big Data

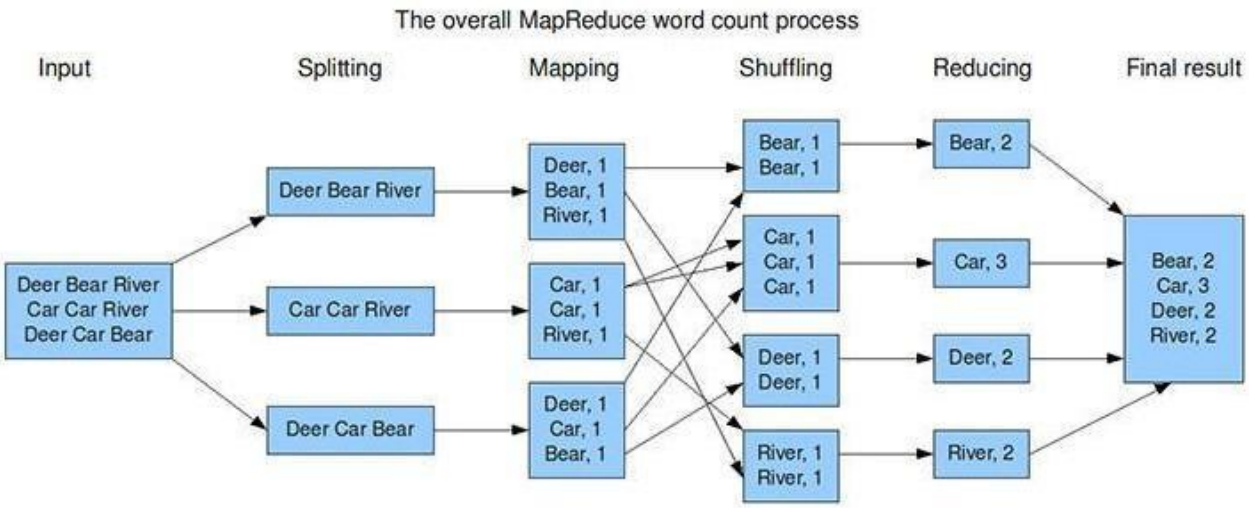
- Theo Wikipedia: Dữ liệu lớn thường bao gồm tập hợp dữ liệu với kích thước vượt xa khả năng của các công cụ phần mềm thông thường để thu thập, hiển thị, quản lý và xử lý dữ liệu trong một thời gian có thể chấp nhận được.
- Dữ liệu lớn bao gồm các thách thức như phân tích, thu thập, giám sát dữ liệu, tìm kiếm, chia sẻ, lưu trữ, truyền nhận, trực quan, truy vấn và tính riêng tư.

1.2 Đặc trưng của Big Data

- Khối lượng lớn (Volume): Khối lượng dữ liệu rất lớn và đang ngày càng tăng lên, tính đến 2014 thì có thể trong khoảng vài trăm terabyte.
- Tốc độ (Velocity): Khối lượng dữ liệu gia tăng rất nhanh.
- Đa dạng (Variety): Ngày nay hơn 80% dữ liệu được sinh ra là phi cấu trúc(tài liệu, blog, hình ảnh,...)
- Độ tin cậy/chính xác(Veracity): Bài toán phân tích và loại bỏ dữ liệu thiếu chính xác và nhiễu đang là tính chất quan trọng của bigdata.
- Giá trị(Value): Giá trị thông tin mang lại.

1.3 Giới thiệu về Hadoop MapReduce

- Hadoop là một framework mã nguồn mở dùng để lưu trữ và xử lý dữ liệu lớn trên các cụm máy tính thông thường.
- Hadoop bao gồm hai thành phần chính:
    + HDFS (Hadoop Distributed File System): Hệ thống lưu trữ phân tán, giúp lưu trữ dữ liệu lớn bằng cách phân mảnh dữ liệu trên nhiều nút.
    + MapReduce: Mô hình lập trình song song, xử lý dữ liệu qua các bước Map và Reduce.
- MapReduce là mô hình lập trình được thiết kế để xử lý dữ liệu song song trên các cụm máy tính phân tán. Mô hình hoạt động qua hai pha chính:
    + Map: Chuyển đổi dữ liệu đầu vào thành các cặp key-value trung gian.
    + Reduce: Tổng hợp và xử lý các cặp key-value để tạo ra kết quả cuối cùng.


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Chương 2: Cách triển khai phân tích dữ liệu khí tượng

2.1 Cào và tạo file dữ liệu đầu vào

- Sử dụng web weather api để lấy api key và tiến hành cào dữ liệu

```python
import requests
from datetime import datetime, timedelta

# WeatherAPI key (thay thế API_KEY bằng key thực tế của bạn)
api_key = 'API_KEY'

# Hàm lấy dữ liệu thời tiết từ API cho một ngày cụ thể
def fetch_weather_data(location, date):
    date_str = date.strftime("%Y-%m-%d")
    url = f'http://api.weatherapi.com/v1/history.json?key={api_key}&q={location}&dt={date_str}'
    response = r
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error fetching data for {location} on {date_str}: {response.status_code}")
        return None
```

(variable) status_code: int

- Sau khi cào thành công tiến hành ghi dữ liệu vào file

```python
# Hàm ghi dữ liệu thời tiết vào file
def write_to_file(data, filename):
    with open(filename, 'w') as file:
        # Ghi tiêu đề cột
        file.write("city\tday/month/year\ttemperature\thumidity\trainfall\twind_speed\n")
        # Ghi từng dòng dữ liệu
        for entry in data:
            file.write(f"{entry['city']}\t{entry['date']}\t{entry['temp']:.2f} °C\t"
                       f"{entry['humidity']:.1f}%\t{entry['rainfall']:.1f} mm\t"
                       f"{entry['wind_speed']:.1f} kph\n")
    print(f"Weather data has been written to {filename}")
```

- Khi chạy chương trình tiến hành, lấy dữ liệu về thời tiết mong muốn từ người dùng

```python
# Nhập dữ liệu từ người dùng
if __name__ == "__main__":
    # Lấy danh sách thành phố và ngày từ người dùng
    cities = input("Enter the locations separated by commas (e.g., Hanoi, Ho Chi Minh City): ").split(", "
    start_date_str = input("Enter the start date (dd/mm/yyyy): ")
    end_date_str = input("Enter the end date (dd/mm/yyyy): ")

    # Chuyển đổi ngày sang định dạng datetime
    start_date = datetime.strptime(start_date_str, "%d/%m/%Y")
    end_date = datetime.strptime(end_date_str, "%d/%m/%Y")

    # Thu thập dữ liệu thời tiết
    weather_data = collect_weather_data(cities, start_date, end_date)

    # Ghi dữ liệu vào file
    write_to_file(weather_data, 'south_test.txt')
```

2.2 Cấu trúc dữ liệu đầu vào

- Dữ liệu đầu vào là 1 bảng bao gồm các cột city, day/month/year, temperature, humidity, rainfall, wind_speed
- Ý nghĩa của từng trường:
    + City: Thành phố mà người dùng mong muốn thu thập dữ liệu
    + Day/month/year: là các ngày thu thập dữ liệu từ ngày bắt đầu đến ngày kết thúc
    + Temperature: Nhiệt độ của các ngày
    + Humidity: Độ ẩm
    + Rainfall: Lượng mưa
    + Wind_speed: Tốc độ gió

2.3 Quy trình xử lý dữ liệu với MapReduce

2.3.1 Mapper

- Lớp Temperature_Mapper được thiết kế để xử lý các nhiệm vụ sau:

1. **Đọc từng dòng dữ liệu đầu vào:** Mỗi dòng là một quan sát thời tiết từ trạm.
2. **Trích xuất các trường dữ liệu cần thiết:** Gồm city, date, temperature, humdity, rainfall, windspeed
3. **Phát các cặp key-value:** Với key là city và month sau khi đã kết hợp lại thành 1 chuỗi và value là các giá trị temperature, humidity, rainfall và windspeed kết hợp lại thành 1 chuỗi

```java
package org.example;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class TemperatureMapper extends Mapper<Object, Text, Text, Text> {

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        String[] fields = line.split("\\t");

        if (fields.length == 6) { // Đảm bảo dòng có đúng 6 cột: D
            String city = fields[0]; // Lấy tên thành phố
            String date = fields[1]; // Lấy ngày
            String temperature = fields[2].replace("°C", "").trim(); // Lấy nhiệt độ và loại bỏ ký tự '°C'
            String humidity = fields[3].replace("%", "").trim(); // Lấy độ ẩm và loại bỏ ký tự '%'
            String rainfall = fields[4].replace(" mm", "").trim(); // Lấy lượng mưa và loại bỏ ký tự 'mm'
            String windspeed = fields[5].replace(" kph", "").trim(); // Lấy tốc độ gió và loại bỏ ký tự 'k
```

```
            // Chuyển đổi date thành định dạng tháng-năm
            SimpleDateFormat inputFormat = new SimpleDateFormat("dd/MM/yyyy");
            SimpleDateFormat outputFormat = new SimpleDateFormat("MM/yyyy");
            try {
                Date parsedDate = inputFormat.parse(date);
                String month = outputFormat.format(parsedDate);

                // Gộp city và month thành key
                String cityMonthKey = city + "-" + month;

                // Gộp các giá trị lại thành chuỗi để truyền cho Reducer
                String combinedValues = temperature + "\t" + humidity + "\t" + rainfall + "\t" + windspeed

                context.write(new Text(cityMonthKey), new Text(combinedValues));
            } catch (ParseException e) {
                System.err.println("Error parsing date: " + date);
                e.printStackTrace();
            }
        }
    }
}
```

2.3.2 Reducer

- Lớp Temperature_Reducer thực hiện các nhiệm vụ quan trọng nhằm tổng hợp và tính toán các thông tin từ dữ liệu nhiệt độ trung gian do Mapper phát ra:

1. **Nhận các cặp key-value từ Mapper:**
    - Các giá trị nhiệt độ được nhóm theo từng city-month (key) do Mapper phát ra.
2. **Tính toán các thông số nhiệt độ:**
    - **Tính trung bình các thuộc tính:** Tính tổng các thuộc tính sau đó chia cho số lượng các bản ghi

```
    for (Text val : values) {
        String[] fields = val.toString().split("\\t");
        double temperature = Double.parseDouble(fields[0]);
        double humidity = Double.parseDouble(fields[1]);
        double rainfall = Double.parseDouble(fields[2]);
        double windspeed = Double.parseDouble(fields[3]);

        sumTemperature += temperature;
        sumHumidity += humidity;
        sumRainfall += rainfall;
        sumWindspeed += windspeed;
        count++;
    }

    double avgTemperature = sumTemperature / count;
    double avgHumidity = sumHumidity / count;
    double avgRainfall = sumRainfall / count;
    double avgWindspeed = sumWindspeed / count;
```

3. **Xuất kết quả:**
    - Kết quả được định dạng thành chuỗi bao gồm thông tin city, month, avgTemperature, avgHumidity, avgRainfall, avgWindspeed

```
        // Định dạng kết quả đầu ra đúng định dạng mong muốn
        String result = String.format("%s\t%s\t%.2f°C\t%.2f%%\t%.2fmm\t%.2fkph",
                                      city, month, avgTemperature, avgHumidity, avgRainfall, avgWindspeed);
        context.write(new Text(city), new Text(result));
    }
}
```

2.3.3 Driver

- Dùng để tạo file jars và chạy các lớp

```java
public class TemperatureDriver {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: TemperatureDriver <input path> <output path>");
            System.exit(-1);
        }

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Temperature Average");

        job.setJarByClass(TemperatureDriver.class);
        job.setMapperClass(TemperatureMapper.class);
        job.setReducerClass(TemperatureReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

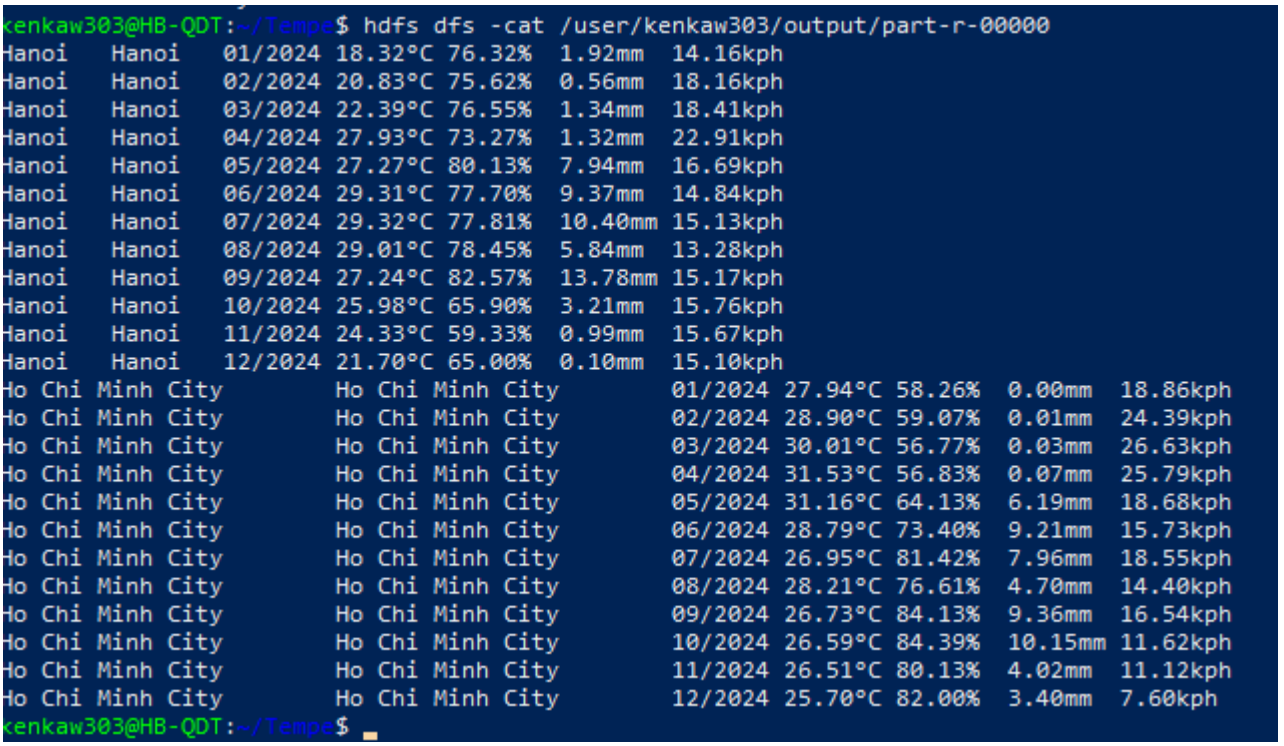
AvgTemp_Mapper.java

```java
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class AvgTemp_Mapper extends Mapper<Object, Text, Text, Text> {
    private Text yearKey = new Text();
    private Text tempValue = new Text();

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, Int
        String line = value.toString();
        // Tách các trường từ dữ liệu
        String year = line.substring(15, 19); // Năm
        String temp = line.substring(87, 92); // Nhiệt độ
        String quality = line.substring(92, 93); // Chất lượng dữ liệu
        if (!temp.trim().equals("+9999") && quality.matches("[01459]")) {
            yearKey.set(year);
            tempValue.set(temp);
            context.write(yearKey, tempValue);
        }
    }
}
```

Chương 3: Đánh giá kết quả

3.1 Kết quả chạy chương trình

- Sau khi triển khai chương trình trên tập dữ liệu mẫu, kết quả đầu ra như sau:



3.2 Đánh giá chương trình

Ưu điểm:

- Có thể lấy dữ liệu từ thực tế với số lượng ngày tháng mong muốn
- Chương trình xử lý chính xác các dữ liệu nhiệt độ lớn và phức tạp.
- Khả năng mở rộng dễ dàng nhờ sử dụng Hadoop.
- Xử lý hiệu quả dữ liệu không hợp lệ thông qua bộ lọc Mapper.

Hạn chế:

- Kết quả đầu ra chưa được tích hợp visualization để dễ hiểu hơn.

Chương 4: Kết luận và hướng phát triển

4.1 Kết luận

- Đề tài "Phân tích dữ liệu khí tượng bằng Hadoop MapReduce" đã hoàn thành các mục tiêu đề ra, bao gồm:
- Triển khai thành công hệ thống phân tích nhiệt độ từ dữ liệu lớn.
- Xử lý hiệu quả dữ liệu không hợp lệ và cung cấp các thông số quan trọng như nhiệt độ trung bình, cao nhất và thấp nhất theo từng năm.
- Chương trình cho thấy tiềm năng ứng dụng lớn của Hadoop trong các bài toán xử lý dữ liệu khí tượng và khoa học môi trường.

4.2 Hướng phát triển

- **Tích hợp thêm yếu tố phân tích:** Bao gồm độ ẩm, lượng mưa, tốc độ gió để mở rộng phân tích khí hậu.
- **Visualization:** Sử dụng các công cụ như Tableau hoặc Power BI để trực quan hóa kết quả.
- **Tối ưu hóa hiệu suất:** Sử dụng các kỹ thuật caching và nén dữ liệu để giảm thời gian xử lý.
- **Ứng dụng thực tế:** Mở rộng hệ thống để xử lý dữ liệu khí tượng theo thời gian thực, hỗ trợ các cơ quan nghiên cứu khí hậu và dự báo thời tiết.

BẢNG PHÂN CHIA CÔNG VIỆC CỦA TỪNG THÀNH VIÊN

Họ và tên	Công việc
Vũ Minh Đức	• Tìm hiểu tổng quan về Hadoop • Tìm hiểu về Map Reduce • Xây dựng lớp Mapper, Reducer, Driver • Làm slide thuyết trình
Quách Đắc Chính	• Tìm hiểu tổng quan về Hadoop • Lên ý tưởng về việc lấy dữ liệu từ weather api • Xây dựng lớp get_api để cào và xây dựng dữ liệu từ weather api • Chạy demo chương trình
Thái Nguyễn Hoàng Bách	• Tìm hiểu tổng quan về Hadoop • Phân tích dữ liệu thu thập được • Viết báo cáo • Xây dựng phần code python
Đào Duy Hưng	• Tìm hiểu tổng quan về Hadoop • Hỗ trợ các phần code bị lỗi • Tìm hiểu về các hướng phát triển của đề tài • Xây dựng phần code python