

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VIỆN TRÍ TUỆ NHÂN TẠO

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



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

ĐỀ TÀI

Phân tích dữ liệu thời tiết sử dụng Hadoop HDFS, MapReduce, Spark

Học phần: Kỹ thuật công nghệ và dữ liệu lớn (24251_INT3229_37)

Giảng Viên Hướng Dẫn: TS. Trần Hồng Việt
ThS. Đỗ Thu Uyên

Ngành: Trí tuệ nhân tạo - QH-2022-I/CQ-A-AI

Nhóm: 23

| Tên thành viên | Mã sinh viên |
|---------------------|--------------|
| Lý Quốc An | 22022660 |
| Bàn Hoàng Sơn | 22022651 |
| Nguyễn Bảo Sơn | 22022613 |
| Cao Đặng Quốc Vương | 22022601 |

MỞ ĐẦU

Phân tích dữ liệu thời tiết là một trong những ứng dụng quan trọng của khoa học dữ liệu, đặc biệt trong bối cảnh biến đổi khí hậu ngày càng phức tạp. Việc tính toán nhiệt độ trung bình hàng năm không chỉ cung cấp thông tin hữu ích cho các nhà khí tượng học mà còn hỗ trợ chính phủ, doanh nghiệp và cộng đồng đưa ra các quyết định chiến lược liên quan đến nông nghiệp, năng lượng, và quy hoạch đô thị.

Tuy nhiên, với khối lượng dữ liệu khổng lồ được thu thập từ các trạm thời tiết trên toàn cầu mỗi ngày, việc xử lý dữ liệu này trở thành một thách thức lớn. Công nghệ Big Data, cụ thể là Hadoop và HDFS, đã mở ra giải pháp hiệu quả nhờ khả năng lưu trữ và tính toán phân tán, giúp giảm đáng kể thời gian và chi phí xử lý.

Đề tài này tập trung vào việc tính toán nhiệt độ trung bình hàng năm từ một tập dữ liệu lớn bằng cách sử dụng hệ thống Hadoop HDFS.

Mục tiêu chính: Tìm giá trị nhiệt độ trung bình hàng năm từ một tập dữ liệu thời tiết lớn được lưu trữ trên HDFS bằng mô hình tính toán song song MapReduce.

Mục tiêu cụ thể:

- Xây dựng và triển khai pipeline xử lý dữ liệu thời tiết trên HDFS.
- Thực hiện chương trình MapReduce để phân tích và tính toán nhiệt độ trung bình theo năm.
- Đánh giá hiệu quả của hệ thống Hadoop HDFS trong việc xử lý dữ liệu lớn

MỤC LỤC

| | |
|--|-----------|
| MỞ ĐẦU..... | 2 |
| MỤC LỤC..... | 3 |
| CHƯƠNG I: TỔNG QUAN VỀ DỮ LIỆU LỚN..... | 4 |
| 1.1 Định nghĩa..... | 4 |
| 1.2 Đặc trưng cơ bản của dữ liệu lớn..... | 4 |
| 1.3 Tổng quan về Hadoop..... | 5 |
| 1.4 Tổng quan về MapReduce..... | 6 |
| 1.5 Tổng quan về HDFS..... | 7 |
| 1.6 Tổng quan về Apache Spark..... | 10 |
| 1.7 Tổng quan bài toán:..... | 12 |
| CHƯƠNG II: Ý TƯỞNG VÀ CÀI ĐẶT..... | 15 |
| 2.1 Ý tưởng MapReduce hóa..... | 15 |
| 2.2 Cài đặt MapReduce..... | 17 |
| 2.3 Ý tưởng Spark..... | 30 |
| 2.4 Cài đặt Spark..... | 31 |
| CHƯƠNG III: DEMO CHƯƠNG TRÌNH..... | 35 |
| 3.1 Demo MapReduce:..... | 35 |
| 3.1.1 Dữ liệu đầu vào:..... | 35 |
| 3.1.2 Demo chương trình:..... | 36 |
| 3.2 Demo Spark:..... | 42 |
| CHƯƠNG IV: KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN..... | 47 |
| 4.1 Kết luận..... | 47 |
| 4.2 Hướng phát triển..... | 48 |
| TÀI LIỆU THAM KHẢO..... | 50 |
| Phân chia công việc..... | 51 |

CHƯƠNG I: TỔNG QUAN VỀ DỮ LIỆU LỚN

1.1 Định nghĩa

- **Theo wikipedia:** Dữ liệu lớn là một thuật ngữ chỉ bộ dữ liệu lớn hoặc phức tạp mà các phương pháp truyền thống không đủ các ứng dụng để xử lý dữ liệu này.
- **Theo Gartner:** Dữ liệu lớn là những nguồn thông tin có đặc điểm chung khói lượng lớn, tốc độ nhanh và dữ liệu định dạng dưới nhiều hình thức khác nhau, do đó muốn khai thác được phải đòi hỏi phải có hình thức mới để đưa ra quyết định khám phá và tối ưu hóa quy trình.
- Dữ liệu đến từ rất nhiều nguồn khác nhau:



Hình 1. Minh họa nguồn gốc của dữ liệu.

- Một số lợi ích có thể mang lại như: Cắt giảm chi phí, tiết kiệm thời gian và giúp tối ưu hóa sản phẩm, hỗ trợ con người đưa ra những quyết định đúng và hợp lý hơn.

1.2 Đặc trưng cơ bản của dữ liệu lớn

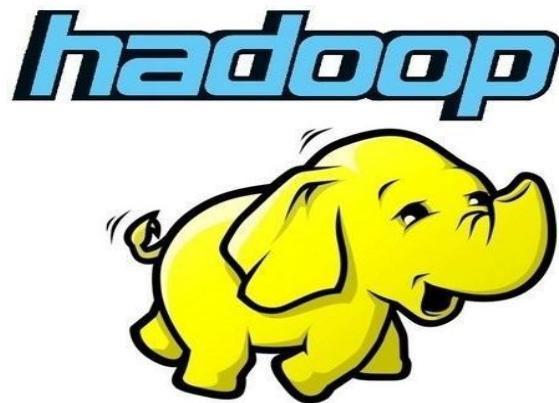
- (1) **Khối lượng lớn (Volume):** Khối lượng dữ liệu rất lớn và đang ngày càng tăng lên, tính đến 2014 thì có thể trong khoảng vài trăm terabyte.
- (2) **Tốc độ (Velocity):** Khối lượng dữ liệu gia tăng rất nhanh.
- (3) **Đa dạng (Variety):** Ngày nay hơn 80% dữ liệu được sinh ra là phi cấu trúc (tài liệu, blog, hình ảnh,...)
- (4) **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.

(5) **Giá trị(Value)**: Giá trị thông tin mang lại.

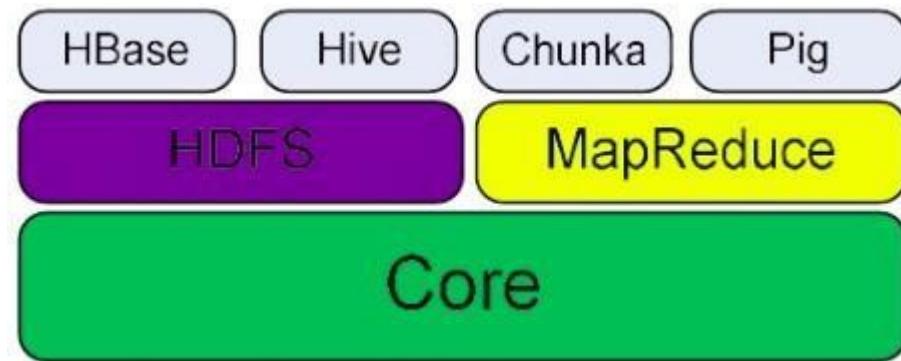
1.3 Tổng quan về Hadoop

- **Theo apache hadoop:** Apache Hadoop là một framework dùng để chạy những ứng dụng trên 1 cluster lớn được xây dựng trên những phần cứng thông thường



Hình 2. Biểu tượng của Hadoop

- + Các thành phần của hadoop: Core, MapReduce engine, HDFS, HBase, Hive, Pig, Chukwa,.. Tuy nhiên *tập chung* vào 2 thành phần quan trọng nhất: HDFS và MapReduce.



Hình 3. Thành phần của Hadoop

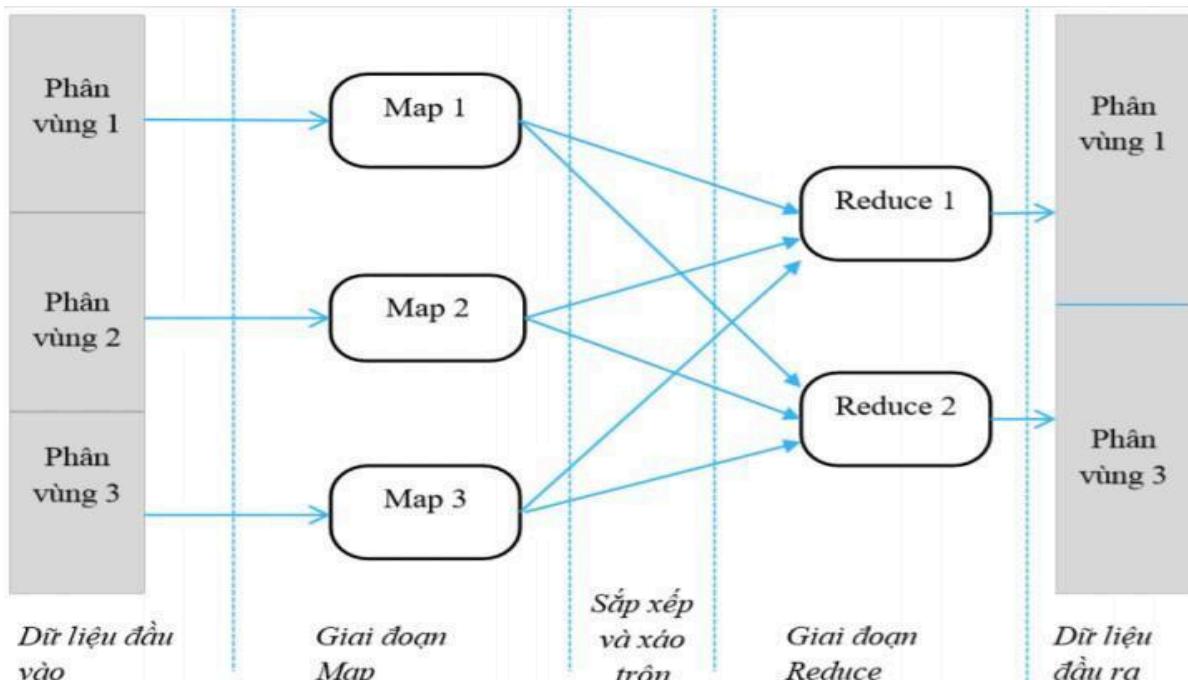
- + Hadoop hiện thực mô hình Map/Reduce, đây là mô hình mà ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn khác nhau, và các phân này sẽ được chạy song song trên nhiều node khác nhau
- + Hadoop hiện thực mô hình Map/Reduce, đây là mô hình mà ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn khác nhau, và các phân này sẽ được chạy song song trên nhiều node khác nhau.
- +Thêm vào đó, Hadoop cung cấp 1 hệ thống file phân tán (HDFS) cho phép lưu trữ dữ liệu lên trên

nhiều node. Cả Map/Reduce và HDFS đều được thiết kế sao cho framework sẽ tự động quản lý được các lỗi, các hư hỏng về phần cứng của các node.

=> **Kết luận:** Là một framework cho phép phát triển các ứng dụng phân tán. Viết bằng java.

1.4 Tổng quan về MapReduce

- **Định nghĩa:** Theo Google, MapReduce là mô hình dùng cho xử lý tính toán song song và phân tán trên hệ thống phân tán
 - B1: Phân rã từ nghiệp vụ chính (do người dùng muốn thể hiện) thành các công việc còn để chia từng công việc con này về các máy tính trong hệ thống thực hiện xử lý một cách song song
 - B2: Thu thập lại các kết quả.
- **Ứng dụng của MapReduce:**
 - + Dữ liệu cần xử lý kích thước lớn.
 - + Các ứng dụng thực hiện xử lý, phân tích dữ liệu, thời gian xử lý đáng kể, có thể tính bằng phút, giờ, ...
- **Thực thi mô hình MapReduce:**



Hình 4. Thực thi mô hình Mapreduce

- + Hàm Map : Hàm Map tiếp nhận mảng dữ liệu input, rút trích thông tin cần thiết các từng phần tử (ví dụ: lọc dữ liệu, hoặc trích dữ liệu) tạo kết quả trung gian
- + Hàm Reduce: tổng hợp kết quả trung gian, tính toán để cho kết quả cuối cùng.

1.5 Tổng quan về HDFS



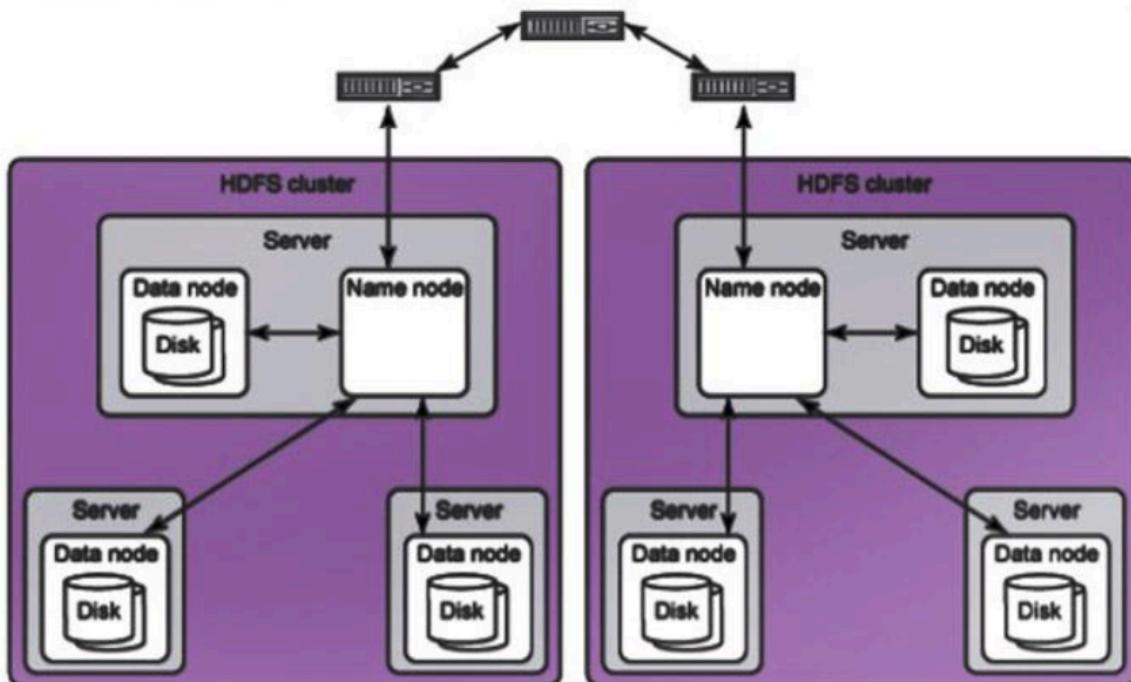
- HDFS là gì ?

- + HDFS (Hadoop Distributed File System) là một hệ thống lưu trữ dữ liệu được Hadoop sử dụng. HDFS hoạt động như một hệ thống tệp phân tán được thiết kế để chạy trên các thiết bị phần cứng thông thường.
- + Khi dữ liệu người dùng ngày càng lớn và kích thước của các file dữ liệu ngày càng vượt quá giới hạn lưu trữ. Lúc này, nhu cầu phân chia dữ liệu ra các ổ cứng trên nhiều máy tính sẽ rất cần thiết. Và HDFS đã ra đời để hỗ trợ giải quyết triệt để được vấn đề này.
- + HDFS có khả năng chịu lỗi cao, nhằm giảm rủi ro cho doanh nghiệp. Chúng được thiết kế cho việc triển khai trên phần cứng hàng hóa với chi phí rẻ. Doanh nghiệp có thể truy cập dữ

liệu thông lượng cao vào các dữ liệu ứng dụng hoặc truy cập trực tuyến vào các dữ liệu hệ thống tệp của Apache Hadoop.

- **Kiến trúc HDFS**

Kiến trúc HDFS



* HDFS hoạt động theo 3 giai đoạn:

1. **Giai đoạn 1:**

- + Một người dùng hoặc một ứng dụng submit một job lên Hadoop với các yêu cầu cơ bản, cụ thể
- + Truyền dữ liệu lên máy chủ và bắt đầu phân tán dữ liệu, trả về kết quả.
- + Các dữ liệu sẽ được chạy dựa trên hàm Map và Reduce. Hàm Map sẽ quét tất cả dữ liệu và phân tán chúng thành các dữ liệu nhỏ hơn, hàm Reduce sẽ thu thập các dữ liệu nhỏ này và

sắp xếp lại.

- + Các thiết lập sẽ phụ thuộc vào job thông qua những thông số truyền vào.

2. *Giai đoạn 2:*

- + Hadoop submit job và lên lịch làm việc, đưa job vào danh sách hàng đợi. Sau khi nhận yêu cầu của JobTracker, các Server Master (Server cha) sẽ chia công việc cho các Server Slave (Server con). Lúc này, Server Slave sẽ thực hiện những job được giao và trả về kết quả cho Server Master.

3. *Giai đoạn 3:*

- + TaskTrackers thực hiện nhiệm vụ kiểm tra, đảm bảo tất cả các MapReduce hoạt động bình thường và kết quả trả về là đúng.

- **Daemon lưu trữ HDFS**

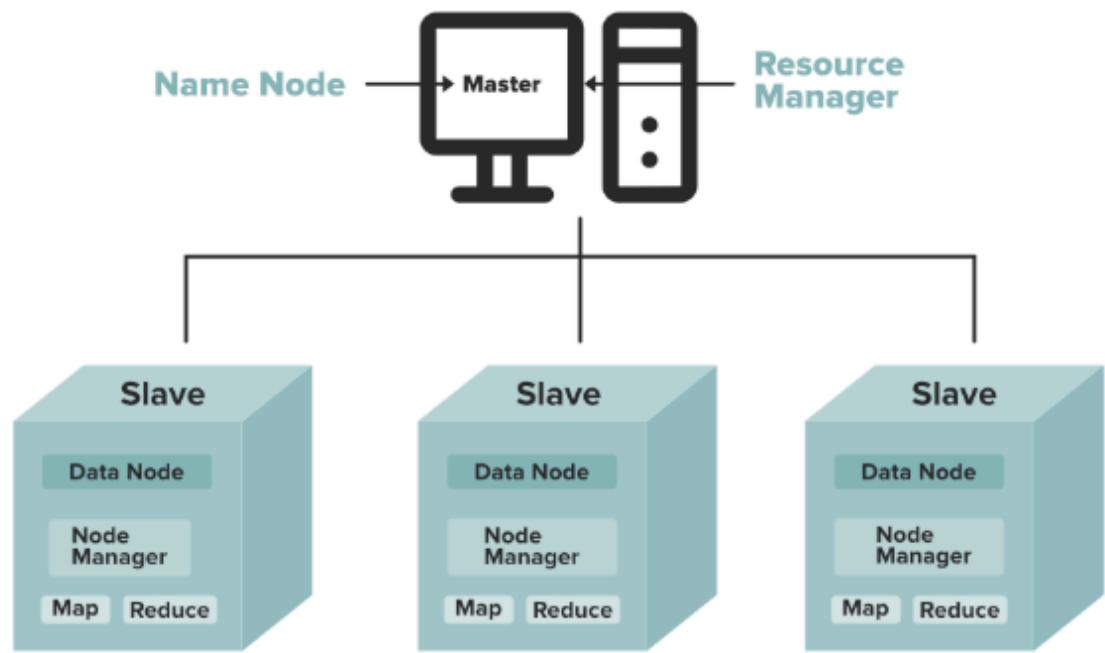
Như chúng ta đã biết, Hadoop hoạt động trên thuật toán MapReduce, là kiến trúc chủ-tớ, HDFS có NameNode và DataNode hoạt động theo mô hình tương tự.

1. NameNode(Chủ)

2. DataNode(Nô lệ)

- + *NameNode*: NameNode hoạt động như Master trong cụm Hadoop hướng dẫn Datanode (Slaves). Namenode chủ yếu được sử dụng để lưu trữ Metadata tức là không có gì ngoài dữ liệu về dữ liệu. Meta Data có thể là nhật ký giao dịch theo dõi hoạt động của người dùng trong cụm Hadoop.
- + Meta Data cũng có thể là tên của tệp, kích thước và thông tin về vị trí (Số khôi, ID khôi) của Datanode mà Namenode lưu trữ để tìm DataNode gần nhất để Giao tiếp nhanh hơn. Namenode hướng dẫn DataNode thực hiện các thao tác như xóa, tạo, Sao chép, v.v.
- + Vì NameNode của chúng ta hoạt động như một Master nên nó phải có RAM hoặc công suất xử lý cao để duy trì hoặc hướng dẫn tất cả các slave trong một cụm Hadoop. Namenode nhận tín hiệu nhịp tim và báo cáo khôi từ tất cả các slave tức là DataNode.
- + *DataNode*: DataNode hoạt động như một Slave DataNode chủ yếu được sử dụng để lưu trữ dữ liệu trong cụm Hadoop, số lượng DataNode có thể từ 1 đến 500 hoặc thậm chí nhiều hơn thế, cụm Hadoop của bạn có càng nhiều DataNode thì càng có thể lưu trữ được nhiều Dữ

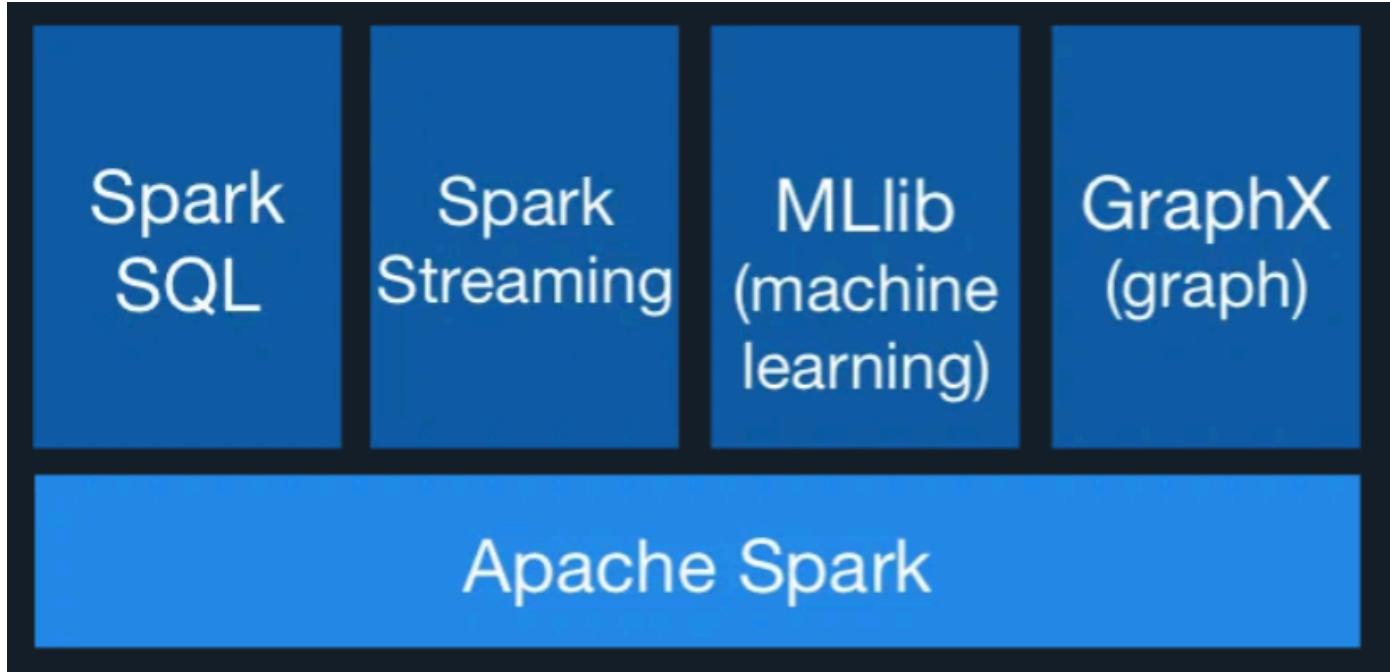
liệu hơn. Vì vậy, nên khuyên DataNode có dung lượng lưu trữ Cao để lưu trữ một số lượng lớn các khối tệp. Datanode thực hiện các hoạt động như tạo, xóa, v.v. theo hướng dẫn do NameNode cung cấp.



=> Tóm lại, Hadoop HDFS là một giải pháp mạnh mẽ và hiệu quả để xử lý bài toán tính toán nhiệt độ trung bình cho năm nhờ khả năng quản lý, xử lý dữ liệu lớn, phân tán tính toán, và tích hợp dễ dàng với các công cụ mạnh mẽ như MapReduce và Spark.

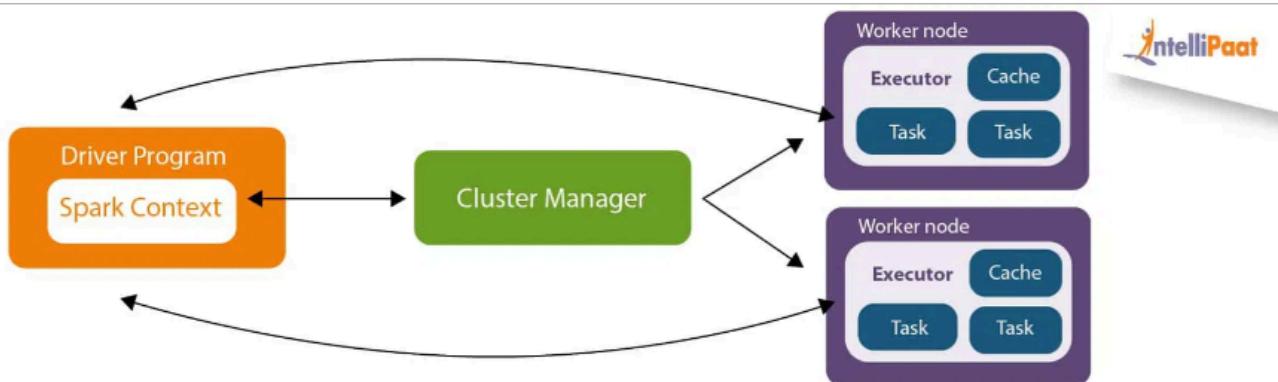
1.6 Tổng quan về Apache Spark:

- **Định nghĩa:** Apache Spark là framework mã nguồn mở tính toán cụm được thiết kế để cung cấp tốc độ tính toán, khả năng mở rộng và khả năng lập trình cho Bigdata
- **Các thành phần của Apache Spark:**



- + **Spark Core:** Cung cấp các chức năng cơ bản cho xử lý dữ liệu phân tán, bao gồm quản lý bộ nhớ, lịch tác vụ và khôi phục lỗi.
- + **Spark SQL:** Cho phép truy vấn dữ liệu bằng SQL và tích hợp với các cơ sở dữ liệu quan hệ, công cụ ETL và Spark Streaming.
- + **Spark Streaming:** Xử lý dữ liệu theo thời gian thực (real-time) bằng cách chia dữ liệu thành microbatch và tích hợp với các module khác như Spark SQL và MLlib.
- + **Spark MLlib (Machine Learning Library):** MLlib cung cấp các thuật toán phổ biến như phân tích hồi quy, phân loại, phân cụm và giảm chiều dữ liệu.
- + **GraphX:** Cung cấp các thuật toán và công cụ xử lý dữ liệu đồ thị phân tán với RDD, VertexRDD và EdgeRDD.

- Kiến trúc của Apache Spark



- + Trong kiến trúc Apache Spark, **Driver Program** là thành phần chính quản lý và điều khiển quá trình xử lý trên cluster. Nó tạo và giám sát **Spark Context**, cung cấp các chức năng cơ bản cho xử lý dữ liệu phân tán.

- + **Spark Driver** và **Spark Context** giám sát quá trình thực thi và phối hợp với **Cluster Manager** để quản lý tài nguyên và phân phối công việc trên các node trong cluster. Khi **Resilient Distributed Datasets (RDD)** được tạo, chúng sẽ được phân phối và lưu trữ trên các **worker nodes**
- + Các **Executors** thực thi nhiệm vụ trên các node, mỗi **Executor** có thể chứa nhiều task và được quản lý bởi **Driver Program**.

1.7 Tổng quan bài toán:

Bài toán là một bài toán phân tích dữ liệu lớn(big data) điển hình, tập trung vào xử lý, phân tích và tính toán dữ liệu thời tiết.

1. Mục tiêu bài toán:

- Dữ liệu thời tiết được thu thập từ nhiều nguồn với khối lượng lớn và khó xử lý bằng các công cụ truyền thống.
- Xử lý dữ liệu lớn hiệu quả bằng cách sử dụng HDFS để lưu trữ phân tán và MapReduce để thực hiện các phép tính phân tán.
- Tính toán dữ liệu theo từng tháng và từng năm từ các tệp dữ liệu thô.

2. Quy trình thực hiện:

Dữ liệu đầu vào được phân tích bằng 3 thành phần chính: **WeatherMapper.java**, **WeatherReducer.java**, **WeatherAnalysysis.java(Driver)**

- Chuẩn bị dữ liệu:
 - File **VNweather_data.txt** chứa dữ liệu thời tiết dưới dạng thô của các tỉnh thành Việt Nam từ 2009 đến 2021 với các trường: Tỉnh , Thời gian và các thông số thời tiết
 - Tiền xử lý dữ liệu: Làm sạch và chuyển đổi về định dạng phù hợp để nhập vào Hadoop HDFS
 - Lưu trữ dữ liệu vào HDFS: Tải dữ liệu lên hệ thống tệp phân tán Hadoop HDFS.
- Xây dựng chương trình MapReduce:
 - **Mapper:** Dữ liệu được đưa vào Map. Hàm Map sẽ phân tích từng dòng dữ liệu đầu vào và trích xuất các thông tin cần thiết và tạo ra các cặp chứa thông tin tương ứng (key-value).
 - **Reducer:** Đối với mỗi khoá, hàm Reduce sẽ nhận được một danh sách các giá trị tương ứng. Hàm này sẽ tính toán các giá trị tổng hợp như trung bình, lớn nhất, nhỏ nhất; tính toán xu

hướng (slope) và hệ số tương quan Pearson giữa các yếu tố thời tiết,..

- **Driver:** Điều khiển luồng xử lý MapReduce và định nghĩa các hướng dẫn dữ liệu đầu vào và đầu ra
- Dữ liệu đầu ra
 - Các thống kê được nhóm theo tỉnh và thời gian
 - Các thông tin về xu hướng và tương quan được cung cấp cho từng tỉnh

3. Ý nghĩa:

Bài toán cung cấp số liệu và nền tảng quan trọng để:

- Dự báo thời tiết và xu hướng khí hậu dài hạn
- Hỗ trợ ra quyết định trong quản lý tài nguyên và ứng phó thiên tai
- Đánh giá tác động của biến đổi khí hậu tại cấp địa phương.

4. Dữ liệu sử dụng:

- Dữ liệu mặc định là 1 file .csv ghi dữ liệu thời tiết của 40 tỉnh ở Việt nam mỗi ngày từ năm 2009 đến năm 2021 với các trường dữ liệu như sau:
 - Tên tỉnh (Viết hoa không dấu)
 - Nhiệt độ cao nhất trong ngày (độ C)
 - Nhiệt độ thấp nhất trong ngày (độ C)
 - Sức gió (km/h)
 - Hướng gió (không sử dụng trong phân tích)
 - Lượng mưa trong ngày (mm)
 - Độ ẩm (%)
 - Lượng mây che phủ (%)
 - Áp suất không khí (không sử dụng trong phân tích)
 - Ngày, tháng, năm (yyyy/mm/dd)

| province | max | min | wind | wind_d | rain | humidi | cloud | pressure | date |
|----------|-----|-----|------|--------|------|--------|-------|----------|------------|
| Bac Lieu | 27 | 22 | 17 | NNE | 6.9 | 90 | 71 | 1010 | 2009-01-01 |
| Bac Lieu | 31 | 25 | 20 | ENE | 0.0 | 64 | 24 | 1010 | 2010-01-01 |
| Bac Lieu | 29 | 24 | 14 | E | 0.0 | 75 | 45 | 1008 | 2011-01-01 |
| Bac Lieu | 30 | 24 | 30 | E | 0.0 | 79 | 52 | 1012 | 2012-01-01 |
| Bac Lieu | 31 | 25 | 20 | ENE | 0.0 | 70 | 24 | 1010 | 2013-01-01 |
| Bac Lieu | 28 | 23 | 14 | ENE | 0.0 | 75 | 55 | 1012 | 2014-01-01 |
| Bac Lieu | 29 | 23 | 10 | ENE | 0.4 | 75 | 42 | 1012 | 2015-01-01 |
| Bac Lieu | 32 | 24 | 22 | ENE | 0.0 | 63 | 9 | 1015 | 2016-01-01 |
| Bac Lieu | 30 | 24 | 20 | ENE | 0.5 | 76 | 35 | 1011 | 2017-01-01 |
| Bac Lieu | 29 | 23 | 16 | E | 0.0 | 70 | 33 | 1010 | 2018-01-01 |
| Bac Lieu | 27 | 22 | 19 | WSW | 4.5 | 73 | 55 | 1012 | 2019-01-01 |
| Bac Lieu | 32 | 23 | 22 | ENE | 0.0 | 61 | 9 | 1014 | 2020-01-01 |
| Bac Lieu | 28 | 21 | 9 | WSW | 0.0 | 74 | 48 | 1011 | 2021-01-01 |
| Bac Lieu | 28 | 21 | 12 | ENE | 0.0 | 74 | 26 | 1014 | 2009-01-09 |
| Bac Lieu | 31 | 25 | 20 | E | 0.6 | 68 | 29 | 1011 | 2010-01-09 |
| Bac Lieu | 31 | 23 | 16 | E | 0.0 | 69 | 27 | 1009 | 2011-01-09 |
| Bac Lieu | 30 | 24 | 17 | E | 0.5 | 73 | 58 | 1011 | 2012-01-09 |
| Bac Lieu | 30 | 25 | 6 | SSE | 11.5 | 81 | 48 | 1009 | 2013-01-09 |
| Bac Lieu | 25 | 23 | 21 | E | 2.5 | 79 | 71 | 1011 | 2014-01-09 |
| Bac Lieu | 31 | 24 | 23 | ENE | 0.0 | 71 | 17 | 1012 | 2015-01-09 |
| Bac Lieu | 31 | 25 | 23 | E | 0.0 | 72 | 25 | 1013 | 2016-01-09 |
| Bac Lieu | 31 | 25 | 23 | E | 0.0 | 75 | 25 | 1008 | 2017-01-09 |
| Bac Lieu | 32 | 25 | 13 | E | 0.0 | 79 | 28 | 1011 | 2018-01-09 |
| Bac Lieu | 31 | 24 | 20 | E | 0.0 | 77 | 13 | 1013 | 2019-01-09 |
| Bac Lieu | 32 | 24 | 16 | E | 0.0 | 70 | 15 | 1010 | 2020-01-09 |
| Bac Lieu | 29 | 23 | 17 | ENE | 0.0 | 70 | 33 | 1010 | 2021-01-09 |
| Bac Lieu | 29 | 22 | 10 | NE | 1.2 | 81 | 40 | 1013 | 2009-01-08 |

- Sau khi tiền xử lý dữ liệu, dữ liệu sử dụng cho chương trình là 1 file .txt với đặc điểm:

- Dòng đầu chứa tên các trường dữ liệu
- Mỗi dòng tiếp theo chứa lần lượt các dữ liệu của các trường sắp xếp theo thứ tự (dòng đầu), các dòng được sắp xếp theo thứ tự tên tĩnh theo và mỗi tĩnh sẽ có các dòng dữ liệu được sắp xếp tăng dần theo thời gian (từ 01/01/2009 - 31/12/2021).

```
province,max,min,wind,wind_d,rain,humidi,cloud,pressure,date
Bac Lieu,27,22,17,NNE,6.9,90,71,1010,2009-01-01
Bac Lieu,28,22,15,ENE,0.5,85,61,1010,2009-01-02
Bac Lieu,23,21,9,ESE,16.7,91,77,1011,2009-01-03
Bac Lieu,27,21,9,E,2.2,86,32,1011,2009-01-04
Bac Lieu,29,22,10,SE,0.0,81,25,1010,2009-01-05
Bac Lieu,30,23,10,E,0.3,81,25,1011,2009-01-06
Bac Lieu,29,24,12,ESE,1.9,83,30,1012,2009-01-07
Bac Lieu,29,22,10,NE,1.2,81,40,1013,2009-01-08
Bac Lieu,28,21,12,ENE,0.0,74,26,1014,2009-01-09
Bac Lieu,26,18,14,E,0.0,69,16,1014,2009-01-10
Bac Lieu,26,18,14,S,0.0,71,45,1014,2009-01-11
Bac Lieu,28,20,11,E,0.0,70,30,1015,2009-01-12
Bac Lieu,28,21,13,E,0.2,70,54,1016,2009-01-13
Bac Lieu,28,19,14,NE,0.0,66,9,1016,2009-01-14
Bac Lieu,28,20,14,SE,0.0,69,19,1016,2009-01-15
Bac Lieu,29,21,12,ENE,0.0,72,26,1015,2009-01-16
Bac Lieu,29,22,13,ENE,0.0,73,19,1014,2009-01-17
Bac Lieu,30,22,16,E,0.0,71,15,1013,2009-01-18
Bac Lieu,32,22,13,E,0.0,71,5,1012,2009-01-19
Bac Lieu,31,22,12,E,0.0,71,1,1011,2009-01-20
Bac Lieu,31,23,14,E,0.0,69,4,1011,2009-01-21
Bac Lieu,32,23,12,E,0.0,69,22,1010,2009-01-22
Bac Lieu,26,24,11,E,0.4,79,61,1010,2009-01-23
Bac Lieu,26,24,20,ENE,1.7,78,66,1011,2009-01-24
Bac Lieu,29,24,23,E,0.0,70,38,1011,2009-01-25
Bac Lieu,30,23,16,E,0.0,70,36,1012,2009-01-26
Bac Lieu,31,23,17,E,0.0,68,14,1011,2009-01-27
Bac Lieu,31,23,16,ENE,0.0,68,11,1011,2009-01-28
Bac Lieu,32,23,14,ENE,0.0,66,7,1011,2009-01-29
Bac Lieu,32,23,14,E,0.0,66,8,1010,2009-01-30
Bac Lieu,32,23,17,E,0.0,66,3,1011,2009-01-31
```

CHƯƠNG II: Ý TƯỞNG VÀ CÀI ĐẶT

2.1 Ý tưởng MapReduce hóa

1. Hàm Mapper

- Hàm Map có nhiệm vụ trích xuất và xử lý dữ liệu từ các dòng đầu vào để phát ra các cặp key-value dùng trong giai đoạn giảm (Reducer). Ý tưởng chính là lấy thông tin thời tiết (nhiệt độ, độ ẩm, mưa, gió, ...) từ tệp dữ liệu thô và chuẩn bị chúng cho việc tính toán trong các bước sau.
- Các giá trị thời tiết được phát ra với key đại diện cho :
 - Tỉnh và năm-tháng (province, yearMonth): Tính toán trung bình /tháng như nhiệt độ trung bình tháng, độ ẩm, tốc độ gió, v.v.
 - Tỉnh và năm (province, year): Tính toán trung bình/năm tổng lượng mưa, hoặc các thông số thời tiết khác theo năm.
- Các giá trị value sẽ được biểu diễn dưới dạng cặp giá trị bao gồm loại giá trị và giá trị của nó

Ví dụ :

Đầu vào : Hanoi,30,20,15,0.2,50,60,2023-06-15

Key value sẽ đưa ra:

Key: Hanoi,2023-06 | Value: TMAX,30

Key: Hanoi,2023-06 | Value: TMIN,20

Key: Hanoi,2023-06 | Value: WIND,15

Key: Hanoi,2023-06 | Value: RAIN,0.2

Key: Hanoi,2023-06 | Value: HUMIDITY,50

Key: Hanoi,2023-06 | Value: TAVG_MONTH,25.0

...

Key: Hanoi,2023 | Value:TMAX_YEAR,36

Key: Hanoi,2023 | Value:TAVG_YEAR,30

2. Hàm Reducer

- Hàm Reduce thực hiện các phép tính tổng hợp, phân tích cũng như xác định các xu hướng và mô

tương quan liên quan đến thời tiết từ nhiều khu vực và thời điểm. Nó tính toán các giá trị tổng hợp, phân tích xu hướng, và xác định mối tương quan

- Tách thông tin theo loại dữ liệu (nhiệt độ, gió, mưa, độ ẩm, mây) và tính tổng, số lượng, hoặc giá trị lớn nhất/nhỏ nhất
- Tính trung bình nhiệt độ cao nhất, thấp nhất, nhiệt độ trung bình, tốc độ gió, độ ẩm, và lượng mưa trong tháng, năm
- Tính toán đặc biệt cho từng loại thời gian
- Tính toán theo tháng
 - Tính nhiệt độ cao nhất / thấp nhất
 - Tính nhiệt độ trung bình theo tháng
 - Tính trung bình : gió, mưa, độ ẩm
- Tính toán theo năm :
 - Tính trung bình độ ẩm, gió
 - Tính nhiệt độ trung bình ngày theo năm
 - Tính tổng lượng mưa
- Sử dụng hệ số tương quan Pearson để xác định mối tương quan

$$r_{X,Y} = \frac{\text{Cov}(X, Y)}{\sigma_X \cdot \sigma_Y}$$

Trong đó:

$\text{Cov}(X, Y)$: Hiệp phương sai của X và Y

σ_X và σ_Y : Độ lệch chuẩn của X và Y

- Sử dụng độ dốc tuyến tính (slope) để xác định xu hướng thời tiết
 - Phân tích xu hướng nhiệt độ và lượng mưa theo năm (tăng/giảm/ ổn định)
 - Tính độ dốc (slope) của tuyến tính

$$\text{slope} = \frac{n \cdot \text{sumXY} - \text{sumX} \cdot \text{sumY}}{n \cdot \text{sumX}^2 - (\text{sumX})^2}$$

Nếu :

Slope > 0: Xu hướng tăng.

Slope < 0: Xu hướng giảm.

Slope = 0: Xu hướng ổn định.

- So sánh giữa các tỉnh

Xác định tinh có:

- Nhiệt độ trung bình cao nhất/thấp nhất.
- Lượng mưa cao nhất/thấp nhất (theo tổng/năm)

- Tính toán các thông tin theo mùa
- Tính toán chỉ số nhiệt
- Thực hiện kiểm định Mann-Kendall
- Xác định mùa dựa trên chỉ số nhiệt độ

2.2 Cài đặt MapReduce

1. Class WeatherMapper

- Loại bỏ dòng đầu tiên chứa thông tin của dữ liệu

```
public class WeatherMapper extends Mapper<LongWritable, Text, Text, Text> {
    private boolean isFirstLine = true;

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        if (isFirstLine) {
            isFirstLine = false;
            return; // Bỏ qua dòng đầu tiên
        }
    }
}
```

- Trích xuất các thông tin cần thiết:
Mỗi dòng dữ liệu được phân tách bằng dấu phẩy (split(",")) và được chia thành các trường.

```
String[] fields = value.toString().split(",");
if (fields.length == 10) {
    String province = fields[0];
    String date = fields[9];
    String yearMonth = date.substring(0, 7); // Lấy năm-tháng
    String year = date.substring(0, 4); // Lấy năm
    double maxTemp = Integer.parseInt(fields[1]);
    double minTemp = Integer.parseInt(fields[2]);
    double wind = Integer.parseInt(fields[3]);
    double rain = Double.parseDouble(fields[5]);
    double humidity = Integer.parseInt(fields[6]);
    double cloud = Integer.parseInt(fields[7]);
```

+ *Thông tin cơ bản:*

province: Tên địa phương (vd: Hanoi).
date: Ngày tháng (vd: 2023-06-15).
yearMonth: Năm và tháng (vd: 2023-06).
year: Chỉ năm (vd: 2023).

+ *Thông tin thời tiết:*

maxTemp: Nhiệt độ cao nhất.

minTemp: Nhiệt độ thấp nhất.

wind: Tốc độ gió.

rain: Lượng mưa.

humidity: Độ ẩm.

cloud: Độ che phủ mây.

- Tạo các cặp key value :

Cặp key value bao gồm tính toán theo tháng và tính toán theo năm , nhiệt độ trung bình của tháng/năm được tính toán bằng trung bình của tổng giữa nhiệt độ cao nhất và thấp nhất trong tháng/năm.

```
context.write(new Text(province + "," + yearMonth), new Text("TMAX," + maxTemp));
context.write(new Text(province + "," + yearMonth), new Text("TMIN," + minTemp));
context.write(new Text(province + "," + yearMonth), new Text("WIND," + wind));
context.write(new Text(province + "," + yearMonth), new Text("RAIN," + rain));
context.write(new Text(province + "," + yearMonth), new Text("HUMIDITY," + humidity));
context.write(new Text(province + "," + yearMonth), new Text("TAVG_MONTH," + ((maxTemp + minTemp)/2)));

context.write(new Text(province + "," + year), new Text("TMAX_YEAR," + maxTemp));
context.write(new Text(province + "," + year), new Text("TMIN_YEAR," + minTemp));
context.write(new Text(province + "," + year), new Text("TAVG_YEAR," + ((maxTemp + minTemp)/2)));
context.write(new Text(province + "," + year), new Text("WIND_YEAR," + wind));
context.write(new Text(province + "," + year), new Text("TOTAL_RAIN_YEAR," + rain));
context.write(new Text(province + "," + year), new Text("HUMIDITY_YEAR," + humidity));
context.write(new Text(province + "," + year), new Text("CLOUD_YEAR," + cloud));
```

2. Class WeatherReducer

- Tạo Hashmap để lưu giá trị:

Các giá trị được khởi tạo gồm

provinceYearlyAvgTemp: nhiệt độ trung bình hàng năm của từng tỉnh

provinceYearlyAvgRain: lượng mưa trung bình hàng năm của từng tỉnh

provinceYearlyTotalRain: tổng lượng mưa của từng tỉnh

tempTrend: xu hướng nhiệt độ theo năm của từng tỉnh

rainTrend: xu hướng lượng mưa theo năm của từng tỉnh

```
public class WeatherReducer extends Reducer<Text, Text, Text> {  
    private Map<String, Double> provinceYearlyAvgTemp = new HashMap<>();  
    private Map<String, Double> provinceYearlyAvgRain = new HashMap<>();  
    private Map<String, Double> provinceYearlyTotalRain = new HashMap<>();  
    private Map<String, TreeMap<Integer, Double>> tempTrend = new HashMap<>();  
    private Map<String, TreeMap<Integer, Double>> rainTrend = new HashMap<>();
```

- Khởi tạo các giá trị để bắt đầu tính toán:

```
double maxTempYear = Integer.MIN_VALUE;  
double minTempYear = Integer.MAX_VALUE;  
double maxHumidityYear = Integer.MIN_VALUE;  
double minHumidityYear = Integer.MAX_VALUE;  
  
double sumTemp = 0, sumHumidity = 0, sumRain = 0, sumCloud = 0;  
double sumTempSquared = 0, sumHumiditySquared = 0, sumRainSquared = 0, sumCloudSquared = 0;  
double sumTempHumidity = 0, sumTempRain = 0, sumTempCloud = 0, sumRainCloud = 0, sumRainHumidity = 0;  
int count = 0;  
double currentTemp = 0;  
double currentRain = 0;
```

- Cộng dồn và đếm các giá trị:

```

for (Text val : values) {
    String[] valParts = val.toString().split(",");
    String type = valParts[0];
    double value = Double.parseDouble(valParts[1]);

    switch (type) {
        case "TMAX":
            tempSum.put(type, tempSum.getOrDefault(type, (double) 0) + value);
            tempCount.put(type, tempCount.getOrDefault(type, 0) + 1);
            break;
        case "TMIN":
            tempSum.put(type, tempSum.getOrDefault(type, (double) 0) + value);
            tempCount.put(type, tempCount.getOrDefault(type, 0) + 1);
            break;
        case "TAVG_MONTH":
            tempSum.put(type, tempSum.getOrDefault(type, (double) 0) + value);
            tempCount.put(type, tempCount.getOrDefault(type, 0) + 1);
            break;
        case "WIND":
            windSum.put(type, windSum.getOrDefault(type, (double) 0) + value);
            windCount.put(type, windCount.getOrDefault(type, 0) + 1);
            break;
        case "RAIN":
            rainSum.put(type, rainSum.getOrDefault(type, 0.0) + value);
            rainCount.put(type, rainCount.getOrDefault(type, 0) + 1);
            break;
        case "HUMIDITY":
            humiditySum.put(type, humiditySum.getOrDefault(type, (double) 0) + value);
            ...
    }
}

```

- + Dựa trên các cặp key,value được tách ra từ hàm **WeatherMapper** ở trên , các giá trị value được tách ra tùy vào kiểu dữ liệu từ phần tử thứ 2
- + Tùy vào kiểu dữ liệu, giá trị sẽ được cộng dồn vào các tổng và đếm khác nhau, có thể tính toán trung bình, giá trị tối đa, tối thiểu, và thực hiện các phép toán bổ sung (như mối quan hệ giữa nhiệt độ và các yếu tố thời tiết khác)

- Tính toán các giá trị trung bình theo từng tháng trên năm

```

if (yearMonthOrYear.length() == 7) { // Year-Month
    double avgMaxTemp = tempSum.getOrDefault("TMAX", (double) 0) / tempCount.getOrDefault("TMAX", 1);
    double avgMinTemp = tempSum.getOrDefault("TMIN", (double) 0) / tempCount.getOrDefault("TMIN", 1);
    double avgTempMonth = tempSum.getOrDefault("TAVG_MONTH", (double) 0) / tempCount.getOrDefault("TAVG_MONTH", 1);
    double avgWindSpeed = windSum.getOrDefault("WIND", (double) 0) / windCount.getOrDefault("WIND", 1);
    double avgHumidity = humiditySum.getOrDefault("HUMIDITY", (double) 0) / humidityCount.getOrDefault("HUMIDITY", 1);
    double avgRainMonth = rainSum.getOrDefault("RAIN", (double) 0) / rainCount.getOrDefault("RAIN", 1);

    context.write(new Text("Average MaxTemperature of \"" + province + "\" in \"" + yearMonthOrYear + "\":"),
                 new Text(String.format("%.2f", avgMaxTemp)));
    context.write(new Text("Average MinTemperature of \"" + province + "\" in \"" + yearMonthOrYear + "\":"),
                 new Text(String.format("%.2f", avgMinTemp)));
    context.write(new Text("Average Temperature of \"" + province + "\" in \"" + yearMonthOrYear + "\":"),
                 new Text(String.format("%.2f", avgTempMonth)));
    context.write(new Text("Average Wind speed of \"" + province + "\" in \"" + yearMonthOrYear + "\":"),
                 new Text(String.format("%.2f", avgWindSpeed)));
    context.write(new Text("Average Humidity of \"" + province + "\" in \"" + yearMonthOrYear + "\":"),
                 new Text(String.format("%.2f", avgHumidity)));
    context.write(new Text("Average Rain of \"" + province + "\" in \"" + yearMonthOrYear + "\":"),
                 new Text(String.format("%.2f", avgRainMonth)));
}

```

- + Kiểm tra độ dài của chuỗi có bằng 7 (tương ứng với định dạng yyyy-MM) để tiến hành bước tính toán tiếp theo
- + Các giá trị được tính toán bao gồm
 - Nhiệt độ tối đa trung bình
 - Nhiệt độ tối thiểu trung bình
 - Nhiệt độ trung bình giữa các tháng
 - Tốc độ gió trung bình
 - Độ ẩm trung bình
 - Lượng mưa trung bình trong tháng
- + Tính giá trị tối đa trung bình bằng cách chia tổng tối đa
- + Mỗi giá trị trung bình được ghi vào context dưới dạng một chuỗi. Ví dụ, giá trị nhiệt độ tối đa trung bình sẽ được ghi vào dưới dạng một dòng có cấu trúc:

"Average MaxTemperature of "<province>" in "<yearMonthOrYear>\": <avgMaxTemp>"

- Tính toán các giá trị trung bình theo năm

```

        new DecimalFormat("#.##", avgRainFormatter));
} else { // Year
    double avgTempYear = tempSum.getOrDefault("TAVG_YEAR", (double) 0) / tempCount.getOrDefault("TAVG_YEAR", 1);
    double avgWindSpeedYear = windSum.getOrDefault("WIND_YEAR", (double) 0) / windCount.getOrDefault("WIND_YEAR", 1);
    double totalRain = rainSum.getOrDefault("TOTAL_RAIN_YEAR", (double) 0);
    double avgHumidityYear = humiditySum.getOrDefault("HUMIDITY_YEAR", (double) 0) / humidityCount.getOrDefault("HUMIDITY_YEAR", 1);
    int rainDays = rainCount.getOrDefault("TOTAL_RAIN_YEAR", 1);
    double avgRainYear = totalRain / rainDays;
}

```

- So sánh giữa các tỉnh

```

for (Map.Entry<String, Double> entry : provinceYearlyAvgTemp.entrySet()) {
    if (entry.getValue() > maxTemp) {
        maxTemp = entry.getValue();
        maxTempProvince = entry.getKey();
    }
}

for (Map.Entry<String, Double> entry : provinceYearlyAvgTemp.entrySet()) {
    if (entry.getValue() < minTemp) {
        minTemp = entry.getValue();
        minTempProvince = entry.getKey();
    }
}

for (Map.Entry<String, Double> entry : provinceYearlyTotalRain.entrySet()) {
    if (entry.getValue() > maxRainTotal) {
        maxRainTotal = entry.getValue();
        maxRainTotalProvince = entry.getKey();
    }
}

for (Map.Entry<String, Double> entry : provinceYearlyTotalRain.entrySet()) {
    if (entry.getValue() > minRainTotal) {
        minRainTotal = entry.getValue();
        minRainTotalProvince = entry.getKey();
    }
}

for (Map.Entry<String, Double> entry : provinceYearlyAvgRain.entrySet()) {

```

- + Tìm tỉnh có nhiệt độ trung bình hàng năm cao nhất và thấp nhất.
 - + Tìm tỉnh có tổng lượng mưa cao nhất và thấp nhất.
 - + Tìm tỉnh có lượng mưa trung bình hàng năm cao nhất.
- Phân tích xu hướng nhiệt độ và lượng mưa cho từng tỉnh:
- ```

for (String province : tempTrend.keySet()) {
 TreeMap<Integer, Double> tempData = tempTrend.get(province);
 TreeMap<Integer, Double> rainData = rainTrend.get(province);

 double tempSlope = calculateTrend(tempData);
 String tempTrendResult = tempSlope > 0 ? "Increasing" : (tempSlope < 0 ? "Decreasing" : "Stable");

 double rainSlope = calculateTrend(rainData);
 String rainTrendResult = rainSlope > 0 ? "Increasing" : (rainSlope < 0 ? "Decreasing" : "Stable");

 context.write(new Text("Temperature Trend for province \"" + province + "\":"), new Text(tempTrendResult));
 context.write(new Text("Rain Trend for province \"" + province + "\":"), new Text(rainTrendResult));
}

```
- + Sử dụng 2 biến **tempTrend** và **rainTrend** chứa dữ liệu nhiệt độ và lượng mưa theo năm cho từng tỉnh , được lưu dưới dạng `TreeMap<Integer, Double>`
  - + Từ đó áp dụng hàm `calculateTrend` để tính độ dốc (slope) của dữ liệu tuyến tính
- tempSlope:** Xu hướng nhiệt độ (nếu slope > 0, nhiệt độ tăng)
- rainSlope:** Xu hướng lượng mưa (nếu slope > 0, lượng mưa tăng)
- Tính toán heat index (chỉ số nóng bức):

```

// Phương thức tính toán Heat Index
private double calculateHeatIndex(double temperature, double humidity) {
 // Chuyển đổi nhiệt độ sang độ F
 double tempF = (temperature * 9 / 5) + 32;

 // Tính toán Heat Index
 double heatIndex = -42.379 + 2.04901523 * tempF + 10.14333127 * humidity
 - 0.22475541 * tempF * humidity - 0.00683783 * tempF * tempF
 - 0.05481717 * humidity * humidity + 0.00122874 * tempF * tempF * humidity
 + 0.00085282 * tempF * humidity * humidity - 0.00000199 * tempF * tempF * humidity * humidity;

 if (humidity < 13 && tempF >= 80 && tempF <= 112) {
 heatIndex -= ((13 - humidity) / 4) * Math.sqrt((17 - Math.abs(tempF - 95)) / 17);
 }
 else if (humidity > 85 && tempF >= 80 && tempF <= 87) {
 heatIndex += ((humidity - 85) / 10) * ((87 - tempF) / 5);
 }

 if (heatIndex < 80) {
 heatIndex = 0.5 * (tempF + 61 + (tempF - 68) * 1.2 + humidity * 0.094);
 }

 // Chuyển đổi Heat Index sang độ C
 return (heatIndex - 32) * 5 / 9;
}

```

- + Chuyển đổi nhiệt độ từ độ C sang độ F
- + Tính toán chỉ số nhiệt theo 3 trường hợp

Th1: Độ ẩm thấp (< 13%) và nhiệt độ cao ( $80^{\circ}\text{F} \leq \text{tempF} \leq 112^{\circ}\text{F}$ )

Th2: Độ ẩm cao (> 85%) và nhiệt độ vừa phải ( $80^{\circ}\text{F} \leq \text{tempF} \leq 87^{\circ}\text{F}$ )

Th3:Nhiệt độ thấp hơn  $80^{\circ}\text{F}$

- + Chuyển đổi nhiệt độ quay trở lại độ C sau khi tính chỉ số nhiệt ở độ

- Phân tích các yếu tố thời tiết theo chu kỳ 5 năm và 10 năm|

Phân tích chu kỳ 5 và 10 năm được thực hiện thông qua phương thức **performCyclicalAnalysis**

- + Tính trung bình theo năm:

Chuẩn hóa dữ liệu để sử dụng trong các tính toán chu kỳ

1. Tách dữ liệu theo từng năm
2. Tính tổng giá trị(**yearlySum**) và số lượng giá trị(**yearlyCount**) của mỗi năm
3. Tính giá trị trung bình(**yearlyAvg**) cho từng năm bằng cách chia tổng giá trị cho số

lượng giá trị

```
// Tính trung bình theo năm
Map<Integer, Double> yearlySum = new HashMap<>();
Map<Integer, Integer> yearlyCount = new HashMap<>();
for (Map.Entry<Integer, Double> entry : data.entrySet()) {
 int year = entry.getKey();
 double value = entry.getValue();
 yearlySum.put(year, yearlySum.getOrDefault(year, 0.0) + value);
 yearlyCount.put(year, yearlyCount.getOrDefault(year, 0) + 1);
}

Map<Integer, Double> yearlyAvg = new HashMap<>();
for (int year : yearlySum.keySet()) {
 yearlyAvg.put(year, yearlySum.get(year) / yearlyCount.get(year));
}
```

+ Tính tổng và đếm số lượng:

- Sử dụng **cycle5Sum** và **cycle10Sum** để lưu tổng giá trị cho mỗi chu kỳ
- Sử dụng **cycle5Count** và **cycle10Count** để lưu số lượng giá trị
- Lấy các giá trị trong khoảng thời gian: chu kỳ 5 năm từ 2009-2013 và chu kỳ 10 năm từ 2009-2018

+ Tính giá trị trung bình cho mỗi chu kỳ có ít nhất một giá trị hợp lệ

- Chu kỳ 5 năm: **cycle5Avg = cycle5Sum / cycle5Count**
- Chu kỳ 10 năm: **cycle10Avg = cycle10Sum / cycle10Count**

- Tính toán nhiệt độ từng mùa

```

private int getSeasonFromMonth(int month) {
 if (month >= 3 && month <= 5) {
 return 1; // Xuân
 } else if (month >= 6 && month <= 8) {
 return 2; // Hạ
 } else if (month >= 9 && month <= 11) {
 return 3; // Thu
 } else {
 return 4; // Đông
 }
}

```

- + Nếu tháng hiện tại là tháng 3-5 sẽ trả về giá trị 1(tương ứng với mùa xuân)
- + Nếu tháng hiện tại là tháng 6-8 sẽ trả về giá trị 2(tương ứng với mùa hạ)
- + Nếu tháng hiện tại là tháng 9-11 sẽ trả về giá trị 3(tương ứng với mùa thu)
- + Nếu tháng hiện tại là tháng 12-2 sẽ trả về giá trị 4(tương ứng với mùa đông)
- + getSeasonName(int season) dùng để xác định mùa của tháng

- Tính toán hệ số tương quan:

Hàm **calculateCorrelation** được thiết kế để tính toán hệ số tương quan Pearson giữa hai tập dữ liệu. Hệ số này đo lường mối quan hệ tuyến tính giữa hai biến, với giá trị nằm trong khoảng từ -1 đến 1.

- Sử dụng 2 biến **currentTemp** và **currentRain** để lưu giá trị hiện tại (value) đang được xử lý trong khi Reduce và sử dụng cho tính tổng các tích giữa 2 biến (sumXY).
- Các biến chỉ chứa các giá trị của một thành phần (chỉ X hoặc Y) được tính trong các case tính giá trị theo năm tương ứng với các thành phần đang được phân tích tương quan.

```

79 case "TOTAL_RAIN_YEAR":
80 rainSum.put(type, rainSum.getOrDefault(type, 0.0) + value);
81 rainCount.put(type, rainCount.getOrDefault(type, 0) + 1);
82
83 currentRain = value;
84 sumRain += currentRain;
85 sumRainSquared += value * value;
86 sumTempRain += currentTemp * value;
87 break;

```

- Kết quả được lưu vào các biến và được gọi khi xử lý kết quả theo năm, thể hiện xu hướng của lượng mưa và nhiệt độ của các tỉnh trong năm.

```

double corrTempHumidity = calculateCorrelation(count, sumTemp, sumHumidity, sumTempSquared, sumHumiditySquared, sumTempHumidity);
double corrTempRain = calculateCorrelation(count, sumTemp, sumRain, sumTempSquared, sumRainSquared, sumTempRain);
double corrTempCloud = calculateCorrelation(count, sumTemp, sumCloud, sumTempSquared, sumCloudSquared, sumTempCloud);
double corrRainCloud = calculateCorrelation(count, sumRain, sumCloud, sumRainSquared, sumCloudSquared, sumRainCloud);
double corrRainHumidity = calculateCorrelation(count, sumRain, sumHumidity, sumRainSquared, sumHumiditySquared, sumRainHumidity);

```

```

context.write(new Text("Correlation between Temperature and Humidity for \'" + province + "," + yearMonthOrYear + "\":"),
 new Text(String.format("%.4f", corrTempHumidity)));
context.write(new Text("Correlation between Temperature and Rain for \'" + province + "," + yearMonthOrYear + "\":"),
 new Text(String.format("%.4f", corrTempRain)));
context.write(new Text("Correlation between Temperature and Cloud for \'" + province + "," + yearMonthOrYear + "\":"),
 new Text(String.format("%.4f", corrTempCloud)));
context.write(new Text("Correlation between Rain and Cloud for \'" + province + "," + yearMonthOrYear + "\":"),
 new Text(String.format("%.4f", corrRainCloud)));
context.write(new Text("Correlation between Rain and Humidity for \'" + province + "," + yearMonthOrYear + "\":"),
 new Text(String.format("%.4f", corrRainHumidity)));

```

- Hàm tính hệ số tương quan Pearson:

```

private double calculateCorrelation(int n, double sumX, double sumY, double sumX2, double sumY2, double sum
// Tính độ chỉ số tương quan Pearson
double numerator = n * sumXY - sumX * sumY;
double denominator = Math.sqrt((n * sumX2 - sumX * sumX) * (n * sumY2 - sumY * sumY));
double result = numerator / denominator;
return Math.max(-1.0, Math.min(1.0, result));
}

```

- Hàm thực hiện kiểm định Mann-Kendall

```

// Phương thức thực hiện kiểm định Mann-Kendall
private double calculateMannKendall(TreeMap<Integer, Double> data) {
 int n = data.size();

 double[] values = data.values().stream().mapToDouble(Double::doubleValue).toArray();
 double s = 0;

 for (int i = 0; i < n - 1; i++) {
 for (int j = i + 1; j < n; j++) {
 s += Math.signum(values[j] - values[i]);
 }
 }

 double varS = (n * (n - 1) * (2 * n + 5)) / 18.0;
 double z;

 if (s > 0) {
 z = (s - 1) / Math.sqrt(varS);
 } else if (s < 0) {
 z = (s + 1) / Math.sqrt(varS);
 } else {
 z = 0;
 }

 return z;
}

```

- + Hàm **calculateMannKendall** được sử dụng để kiểm tra xu hướng tăng hoặc giảm trong dữ liệu chuỗi thời gian, mà không yêu cầu dữ liệu phải phân phối chuẩn.
  - + Mann-Kendall giúp xác định liệu các yếu tố thời tiết (như nhiệt độ, độ ẩm) đang có xu hướng thay đổi qua thời gian.
  - + Kết quả kiểm định cung cấp cơ sở để đánh giá sự thay đổi khí hậu trên từng khu vực.
- Cách hoạt động:
1. Tính **S** bằng cách so sánh tất cả các cặp giá trị trong chuỗi thời gian.
  2. Tính phương sai **varS** để chuẩn hóa giá trị **S**.
  3. Tính giá trị **Z**, dùng để xác định ý nghĩa thống kê của xu hướng.
    - a. **Z > 0**: xu hướng tăng.
    - b. **Z < 0**: Xu hướng giảm.
    - c. **Z = 0**: Không có.
- Hàm tính toán Sen's Slope:

```

// Phương thức tính toán Sen's Slope
private double calculateSensSlope(TreeMap<Integer, Double> data) {
 if (data.size() < 2) {
 return 0;
 }

 SimpleRegression regression = new SimpleRegression();
 for (Map.Entry<Integer, Double> entry : data.entrySet()) {
 regression.addData(entry.getKey(), entry.getValue());
 }

 return regression.getSlope();
}

```

- + Hàm **calculateSensSlope**: được sử dụng để định lượng tốc độ thay đổi trung bình trong dữ liệu thời gian
  - + Cung cấp thông tin về mức độ thay đổi của nhiệt độ, lượng mưa hay độ ẩm theo thời gian.
  - + Cung cấp thông tin về mức độ thay đổi của nhiệt độ, lượng mưa hay độ ẩm theo thời gian.
- Cách hoạt động:
1. Tính tất cả các độ dốc giữa mọi cặp giá trị trong chuỗi thời gian.
  2. Tìm giá trị trung vị (median slope) làm đại diện cho tốc độ thay đổi trung bình.
  3. Độ dốc dương chỉ xu hướng tăng, âm chỉ xu hướng giảm.

## 2.3 Ý tưởng Spark

- **Mục tiêu**
- Phân tích dữ liệu khí hậu của các tỉnh thành Việt Nam để đánh giá các yếu tố khí hậu như nhiệt độ, lượng mưa, độ ẩm, áp suất, tốc độ gió, mây.
- Đưa ra các thống kê và xu hướng liên quan đến khí hậu, bao gồm:
  - Trung bình cực trị theo tỉnh, mùa, năm.
  - Tỉnh có yếu tố khí hậu cao/thấp nhất.
  - Xu hướng thay đổi lượng mưa và nhiệt độ trung bình.
  - Tương quan giữa các yếu tố khí hậu tại một tỉnh cụ thể trong một năm.
  
  
- **Phương pháp**  
Sử dụng Apache Spark trong để xử lý dữ liệu khí hậu quy mô lớn với hiệu suất cao. Các thư viện và kỹ thuật chính bao gồm:
- **PySpark** để xử lý và tính toán phân tán.

- Các hàm tích hợp như `max`, `min`, `avg`, `corr` để phân tích dữ liệu.
- Kỹ thuật nhóm dữ liệu theo cột (GroupBy) và xử lý dữ liệu theo mùa, năm.

## 2.4 Cài đặt Spark

- **Khởi tạo SparkSession:** Sử dụng Spark để xử lý dữ liệu phân tán.

```
Khởi tạo SparkSession
spark = SparkSession.builder.appName("ClimateAnalysis").getOrCreate()
```

- **Đọc dữ liệu:** Đọc tệp và chuyển đổi kiểu dữ liệu của các cột liên quan đến khí hậu sang dạng số.

```
Đọc dữ liệu từ file input.txt
df = spark.read.csv("/content/drive/MyDrive/dataset/VNweather_data.txt", header=True, inferSchema=True)
```

```
Chuyển đổi kiểu dữ liệu của các cột sang số
numeric_cols = ["max", "min", "wind", "rain", "humidi", "cloud", "pressure"]
for col_name in numeric_cols:
 df = df.withColumn(col_name, col(col_name).cast("double"))
```

- **Tạo thêm cột:** Tính nhiệt độ trung bình (`temp`), đổi tên các cột.

```
Thêm cột temp (nhiệt độ trung bình), maxtemp, mintemp
df = df.withColumn("temp", (col("max") + col("min")) / 2)
df = df.withColumnRenamed("max", "maxtemp")
df = df.withColumnRenamed("min", "mintemp")
df = df.withColumnRenamed("humidi", "humidity")
```

- Các bước thực hiện chính:

1. **Tính trung bình cực trị theo tỉnh:** Sử dụng GroupBy để tính giá trị lớn nhất, nhỏ nhất và trung bình.

```

1. Tính trung bình cực trị của các yếu tố khí hậu theo từng tỉnh
analysis_cols = ["maxtemp", "mintemp", "wind", "rain", "humidity", "cloud", "pressure", "temp"]

for col_name in analysis_cols:
 # Calculate max and min first
 extreme_df = df.groupBy("province").agg(
 max(col_name).alias(f"max_{col_name}"),
 min(col_name).alias(f"min_{col_name}")
)
 # Then calculate the average of max and min in a separate step
 extreme_avg_by_province = extreme_df.groupBy("province").agg(
 avg(col(f"max_{col_name}")).alias(f"avg_max_{col_name}"),
 avg(col(f"min_{col_name}")).alias(f"avg_min_{col_name}")
)
 print(f"Trung bình cực trị của {col_name} theo tỉnh:")
 extreme_avg_by_province.show()

```

## 2. Phân tích dữ liệu theo mùa:

Thêm cột mùa dựa trên tháng.

```

2. Tính trung bình cực trị của các yếu tố khí hậu của một tỉnh theo mùa
df = df.withColumn("month", month(col("date")))

def get_season(month):
 if month in [3, 4, 5]:
 return "Spring"
 elif month in [6, 7, 8]:
 return "Summer"
 elif month in [9, 10, 11]:
 return "Autumn"
 else:
 return "Winter"

get_season_udf = F.udf(get_season)
df = df.withColumn("season", get_season_udf(col("month")))

for col_name in analysis_cols:
 # Calculate max and min first
 extreme_df = df.groupBy("province", "season").agg(
 max(col_name).alias(f"max_{col_name}"),
 min(col_name).alias(f"min_{col_name}")
)
 # Then calculate the average of max and min in a separate step
 extreme_avg_by_season = extreme_df.groupBy("province", "season").agg(# Group by both province and season
 avg(col(f"max_{col_name}")).alias(f"avg_max_{col_name}"),
 avg(col(f"min_{col_name}")).alias(f"avg_min_{col_name}")
)
 print(f"Trung bình cực trị của {col_name} theo tỉnh và mùa:")
 extreme_avg_by_season.show()

```

## 3. Xu hướng khí hậu:

Sử dụng hàm `lag` để tính sự thay đổi qua các năm.

```

5. Xu hướng thay đổi lượng mưa và nhiệt độ trung bình của từng tỉnh
for province_name in df.select("province").distinct().rdd.flatMap(lambda x: x).collect():
 province_df = df.filter(col("province") == province_name)

 # Lượng mưa
 rain_trend = province_df.orderBy("year").select(
 "year", "rain",
 (col("rain") - F.lag("rain", 1).over(Window.orderBy("year"))).alias("rain_diff")
).agg(avg("rain_diff").alias("avg_rain_diff")).first().avg_rain_diff

 rain_trend_label = "Increasing" if rain_trend > 0 else "Decreasing" if rain_trend < 0 else "Stable"

 # Nhiệt độ trung bình
 avg_temp_trend = province_df.orderBy("year").select(
 "year", "temp",
 (col("temp") - F.lag("temp", 1).over(Window.orderBy("year"))).alias("temp_diff")
).agg(avg("temp_diff").alias("avg_temp_diff")).first().avg_temp_diff

 temp_trend_label = "Increasing" if avg_temp_trend > 0 else "Decreasing" if avg_temp_trend < 0 else "Stable"

 print(f'Tỉnh {province_name}:')
 print(f' Xu hướng thay đổi lượng mưa: {rain_trend_label}')
 print(f' Xu hướng thay đổi nhiệt độ trung bình: {temp_trend_label}')

```

#### 4. Hệ số tương quan: Tính tương quan giữa các yếu tố khí hậu.

```

7. Hệ số tương quan giữa các yếu tố khí hậu của Hà Nội trong năm 2019
province_name = "Ha Noi" # Specify the province
year_val = 2019 # Specify the year

Filter data for the specific province and year
year_df = df.filter((col("province") == province_name) & (year(col("date")) == year_val))

print(f"Hệ số tương quan giữa các yếu tố khí hậu của tỉnh {province_name} trong năm {year_val}:")

correlation_matrix = {}
for i in range(len(analysis_cols)):
 for j in range(i + 1, len(analysis_cols)):
 col1 = analysis_cols[i]
 col2 = analysis_cols[j]
 correlation = year_df.select(corr(col1, col2)).first()[0]
 # Check if correlation is None before adding to the matrix
 if correlation is not None:
 correlation_matrix[(col1, col2)] = correlation
 else:
 # Handle the case where correlation is None (e.g., print a message or assign a default value)
 print(f"Correlation between {col1} and {col2} is None. Check your data or calculation.")
 correlation_matrix[(col1, col2)] = 0.0 # For example, assign 0.0

In bảng hệ số tương quan
header = f":<15" + "".join([f":<15" for col in analysis_cols])
print(header)
for col1 in analysis_cols:
 row = f"{col1:<15}"
 for col2 in analysis_cols:
 if (col1, col2) in correlation_matrix:
 row += f"{correlation_matrix[(col1, col2)]:<15.2f}"
 elif (col2, col1) in correlation_matrix:
 row += f"{correlation_matrix[(col2, col1)]:<15.2f}"
 else:
 row += f"1.0:<15.2f" # Đường chéo chính
 print(row)
print("\n")

```

# CHƯƠNG III: DEMO CHƯƠNG TRÌNH

## 3.1 Demo MapReduce:

### 3.1.1 Dữ liệu đầu vào:

- Đẩy dữ liệu đầu vào lên HDFS

```
hson017828@Hson:~$ cp /mnt/d/VNweather/VNweather_data.txt ~/
hson017828@Hson:~$ ls
KMeans.jar VNweather.jar WordCount.jar data.txt points_1.txt result.txt
NaiveBayes.jar VNweather_data.txt data-kmeans.txt iris.csv points_2.txt
```

- Thêm **VNweather\_data.txt** vào /input

```
hson017828@Hson:~$ hadoop fs -put /home/hson017828/VNweather_data.txt /input/
hson017828@Hson:~$ hadoop fs -ls /input
Found 8 items
-rw-r--r-- 1 hson017828 supergroup 265338 2024-12-02 23:56 /input/VNweather.txt
-rw-r--r-- 1 hson017828 supergroup 8710222 2024-12-08 21:33 /input/VNweather_data.txt
-rw-r--r-- 1 hson017828 supergroup 205 2024-10-14 17:01 /input/data-kmeans.txt
-rw-r--r-- 1 hson017828 supergroup 84 2024-10-24 12:43 /input/data.txt
-rw-r--r-- 1 hson017828 supergroup 326 2024-10-04 23:18 /input/input-1.txt
-rw-r--r-- 1 hson017828 supergroup 326 2024-10-04 22:57 /input/input.txt
-rw-r--r-- 1 hson017828 supergroup 4698 2024-10-14 17:43 /input/iris.csv
-rw-r--r-- 1 hson017828 supergroup 172 2024-11-14 11:43 /input/points_1.txt
```

- Cài đặt đầu vào thành công

The screenshot shows the Hadoop File Explorer interface at [localhost:9870/explorer.html#/input](http://localhost:9870/explorer.html#/input). The top navigation bar includes links for Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The main area is titled "Browse Directory" and displays the contents of the /input directory. The table lists 8 entries:

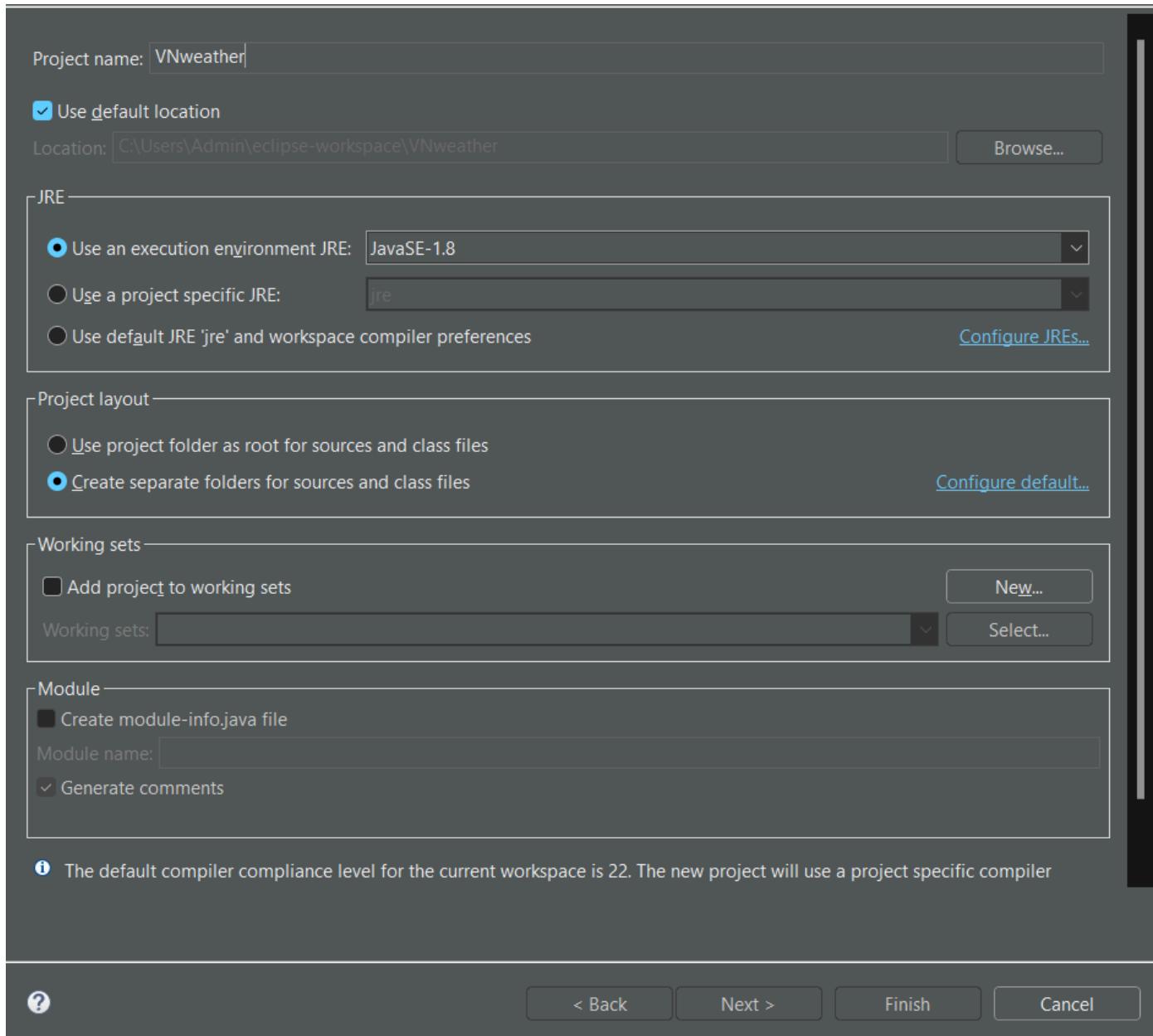
|   | Permission | Owner      | Group      | Size      | Last Modified | Replication | Block Size | Name               |
|---|------------|------------|------------|-----------|---------------|-------------|------------|--------------------|
| □ | -r--r--r-- | hson017828 | supergroup | 259.12 KB | Dec 02 23:56  | 1           | 128 MB     | VNweather.txt      |
| □ | -r--r--r-- | hson017828 | supergroup | 8.31 MB   | Dec 08 21:33  | 1           | 128 MB     | VNweather_data.txt |
| □ | -r--r--r-- | hson017828 | supergroup | 205 B     | Oct 14 17:01  | 1           | 128 MB     | data-kmeans.txt    |
| □ | -r--r--r-- | hson017828 | supergroup | 84 B      | Oct 24 12:43  | 1           | 128 MB     | data.txt           |
| □ | -r--r--r-- | hson017828 | supergroup | 326 B     | Oct 04 23:18  | 1           | 128 MB     | input-1.txt        |
| □ | -r--r--r-- | hson017828 | supergroup | 326 B     | Oct 04 22:57  | 1           | 128 MB     | input.txt          |
| □ | -r--r--r-- | hson017828 | supergroup | 4.59 KB   | Oct 14 17:43  | 1           | 128 MB     | iris.csv           |
| □ | -r--r--r-- | hson017828 | supergroup | 172 B     | Nov 14 11:43  | 1           | 128 MB     | points_1.txt       |

At the bottom, it says "Showing 1 to 8 of 8 entries" and has "Previous" and "Next" buttons. The footer notes "Hadoop, 2023."

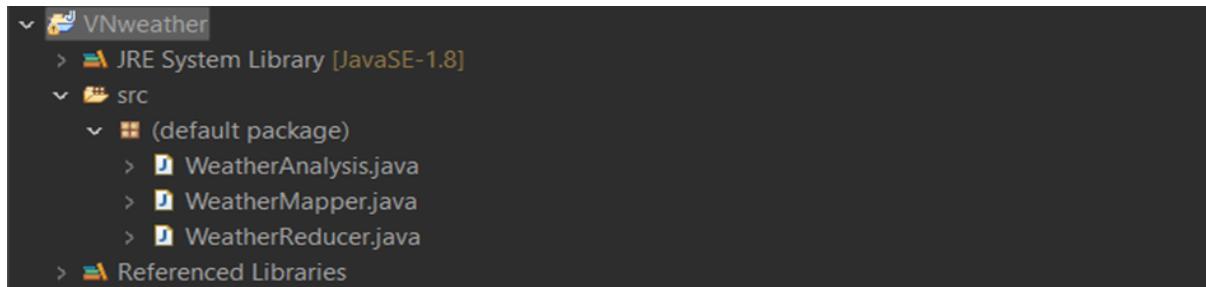
### 3.1.2 Demo chương trình:

#### 1. Cấu trúc thư mục của project

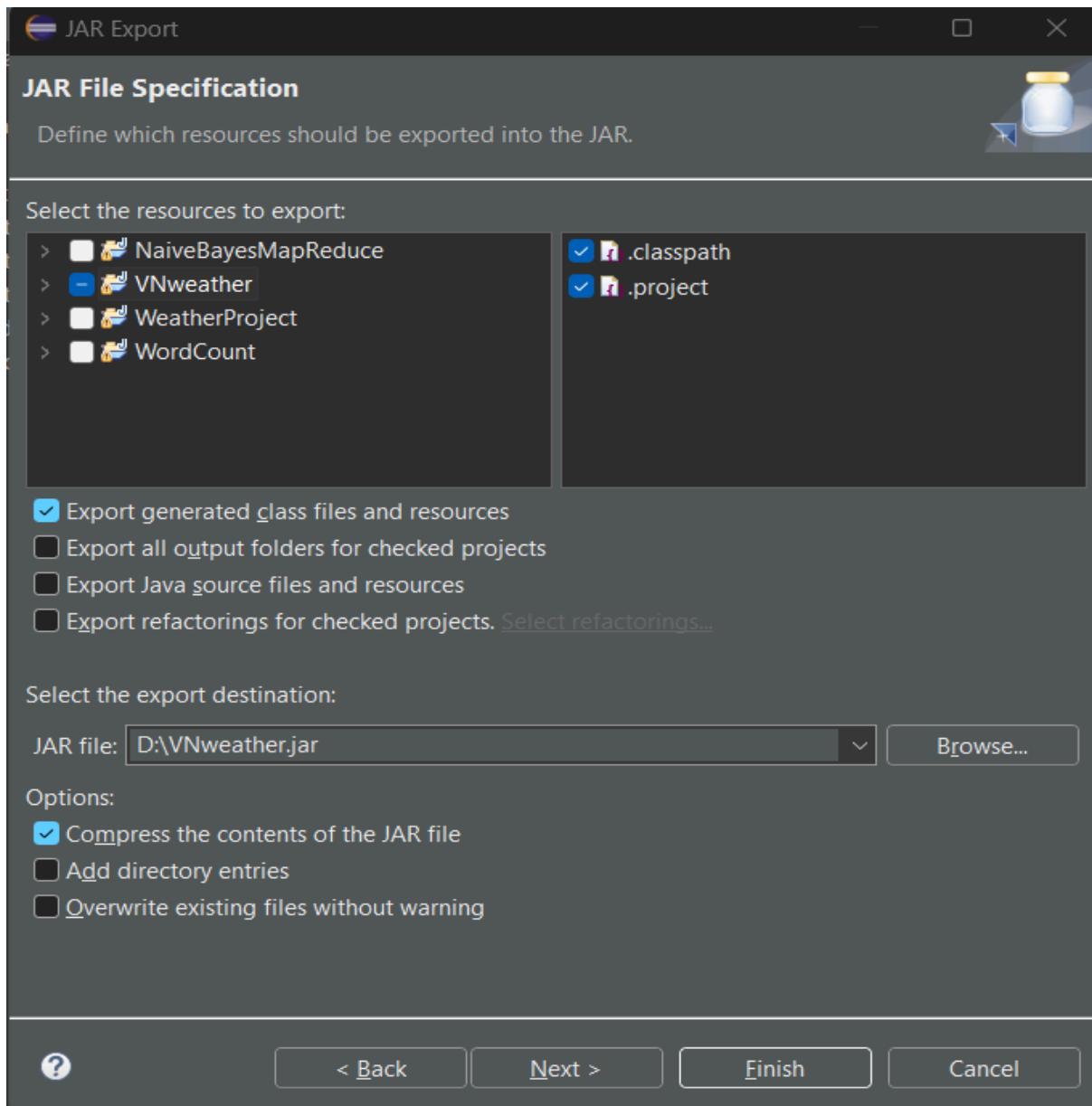
- Khởi tạo Java project trong eclipse, chọn **File -> New -> Java Project**, Đặt tên project là **VNweather** và chọn môi trường là **JavaSE-1.8**



- Tạo class để xử lý nhiệm vụ Map, đặt tên là WeatherMapper
- Tạo class để xử lý nhiệm vụ Reduce, đặt tên là WeatherReducer
- Tạo class để xử lý nhiệm vụ Analysis, đặt tên là WeatherAnalysis



- Xuất file VNweather.jar, chọn **Export -> JAR file -> Finish**



## 2. Tải file jar lên HDFS

```
hson017828@Hson:~$ cp /mnt/d/VNweather/VNweather.jar ~/
hson017828@Hson:~$ ls
KMeans.jar Testweather.jar WordCount.jar data.txt points_1.txt result.txt
NaiveBayes.jar VNweather.jar data-kmeans.txt iris.csv points_2.txt
```

## 3. Chạy chương trình trên ubuntu

```
hson017828@Hson:~$ hadoop jar /home/hson017828/VNweather.jar WeatherAnalysis /input/VNweather_data.txt /weather-
output
2024-12-08 21:35:32,986 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-12-08 21:35:33,640 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface a
nd execute your application with ToolRunner to remedy this.
2024-12-08 21:35:33,680 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hson017828/.staging/
job_1733667560806_0001
2024-12-08 21:35:34,026 INFO input.FileInputFormat: Total input files to process : 1
2024-12-08 21:35:34,171 INFO mapreduce.JobSubmitter: number of splits:1
2024-12-08 21:35:34,462 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1733667560806_0001
2024-12-08 21:35:34,462 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-12-08 21:35:34,718 INFO conf.Configuration: resource-types.xml not found
2024-12-08 21:35:34,718 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-12-08 21:35:35,164 INFO impl.YarnClientImpl: Submitted application application_1733667560806_0001
2024-12-08 21:35:35,226 INFO mapreduce.Job: The url to track the job: http://Hson.:8088/proxy/application_1733667560806_0001/
2024-12-08 21:35:35,227 INFO mapreduce.Job: Running job: job_1733667560806_0001
2024-12-08 21:35:43,350 INFO mapreduce.Job: Job job_1733667560806_0001 running in uber mode : false
2024-12-08 21:35:43,352 INFO mapreduce.Job: map 0% reduce 0%
2024-12-08 21:35:52,480 INFO mapreduce.Job: map 100% reduce 0%
2024-12-08 21:36:00,523 INFO mapreduce.Job: map 100% reduce 100%
2024-12-08 21:36:00,539 INFO mapreduce.Job: Job job_1733667560806_0001 completed successfully
2024-12-08 21:36:00,643 INFO mapreduce.Job: Counters: 54
 File System Counters
 FILE: Number of bytes read=148523862
 FILE: Number of bytes written=223338071
 FILE: Number of read operations=0
 FILE: Number of large read operations=0
 FILE: Number of write operations=0
 HDFS: Number of bytes read=8710333
 HDFS: Number of bytes written=2303744
 HDFS: Number of read operations=8
 HDFS: Number of large read operations=0
 HDFS: Number of write operations=2
 HDFS: Number of bytes read erasure-coded=0
 Job Counters
 Launched map tasks=1
 Launched reduce tasks=1
 Data-local map tasks=1
 Total time spent by all maps in occupied slots (ms)=6560
 Total time spent by all reduces in occupied slots (ms)=5861
 Total time spent by all map tasks (ms)=6560
 Total time spent by all reduce tasks (ms)=5861
 Total vcore-milliseconds taken by all map tasks=6560
```

```

Total time spent by all reduces in occupied slots (ms)=5861
Total time spent by all map tasks (ms)=6560
Total time spent by all reduce tasks (ms)=5861
Total vcore-milliseconds taken by all map tasks=6560
Total vcore-milliseconds taken by all reduce tasks=5861
Total megabyte-milliseconds taken by all map tasks=6717440
Total megabyte-milliseconds taken by all reduce tasks=6001664
Map-Reduce Framework
 Map input records=181961
 Map output records=2365480
 Map output bytes=69530962
 Map output materialized bytes=74261928
 Input split bytes=111
 Combine input records=0
 Combine output records=0
 Reduce input groups=6520
 Reduce shuffle bytes=74261928
 Reduce input records=2365480
 Reduce output records=43365
 Spilled Records=7096440
 Shuffled Maps =1
 Failed Shuffles=0
 Merged Map outputs=1
 GC time elapsed (ms)=161
 CPU time spent (ms)=15420
 Physical memory (bytes) snapshot=785842176
 Virtual memory (bytes) snapshot=5655515136
 Total committed heap usage (bytes)=430964736
 Peak Map Physical memory (bytes)=377438208
 Peak Map Virtual memory (bytes)=2808557568
 Peak Reduce Physical memory (bytes)=408403968
 Peak Reduce Virtual memory (bytes)=2846957568
Shuffle Errors
 BAD_ID=0
 CONNECTION=0
 IO_ERROR=0
 WRONG_LENGTH=0
 WRONG_MAP=0
 WRONG_REDUCE=0
File Input Format Counters
 Bytes Read=8710222
File Output Format Counters
 Bytes Written=2303744

```

## 4. Cài đặt chương trình thành công

The screenshot shows the Hadoop File Explorer interface at the URL [localhost:9870/explorer.html#/weather-output](http://localhost:9870/explorer.html#/weather-output). The top navigation bar includes links for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The main area is titled "Browse Directory" and displays the contents of the "/weather-output" directory. A search bar and a file operations toolbar are visible above the file list. The file list table shows the following entries:

|                          | Permission | Owner      | Group      | Size   | Last Modified | Replication | Block Size | Name         |
|--------------------------|------------|------------|------------|--------|---------------|-------------|------------|--------------|
| <input type="checkbox"/> | -rw-r--r-- | hson017828 | supergroup | 0 B    | Dec 08 21:35  | 1           | 128 MB     | SUCCESS      |
| <input type="checkbox"/> | -rw-r--r-- | hson017828 | supergroup | 2.2 MB | Dec 08 21:35  | 1           | 128 MB     | part-r-00000 |

At the bottom left, it says "Showing 1 to 2 of 2 entries". At the bottom right, there are "Previous", "1", and "Next" buttons, along with a refresh icon.

## 5. Kết quả đầu ra

- Thống kê giá trị trung bình, cực trị của các yếu tố khí hậu của một tỉnh theo từng tháng

|                                                   |       |
|---------------------------------------------------|-------|
| Average MaxTemperature of "Yen Bai" in "2019-01": | 20.45 |
| Average MinTemperature of "Yen Bai" in "2019-01": | 14.55 |
| Average Temperature of "Yen Bai" in "2019-01":    | 17.50 |
| Average Wind speed of "Yen Bai" in "2019-01":     | 7.06  |
| Average Humidity of "Yen Bai" in "2019-01":       | 77.68 |
| Average Rain of "Yen Bai" in "2019-01":           | 1.62  |
| Average MaxTemperature of "Yen Bai" in "2019-02": | 25.57 |
| Average MinTemperature of "Yen Bai" in "2019-02": | 19.29 |
| Average Temperature of "Yen Bai" in "2019-02":    | 22.43 |
| Average Wind speed of "Yen Bai" in "2019-02":     | 11.50 |
| Average Humidity of "Yen Bai" in "2019-02":       | 76.46 |
| Average Rain of "Yen Bai" in "2019-02":           | 3.07  |
| Average MaxTemperature of "Yen Bai" in "2019-03": | 26.58 |
| Average MinTemperature of "Yen Bai" in "2019-03": | 19.94 |
| Average Temperature of "Yen Bai" in "2019-03":    | 23.26 |
| Average Wind speed of "Yen Bai" in "2019-03":     | 11.74 |
| Average Humidity of "Yen Bai" in "2019-03":       | 73.81 |
| Average Rain of "Yen Bai" in "2019-03":           | 1.92  |

- Thống kê giá trị trung bình, cực trị của các yếu tố khí hậu của một tỉnh theo mùa

|                                                                |       |
|----------------------------------------------------------------|-------|
| Seasonal Average Temperature for province "Nam Dinh" (Spring): | 26.23 |
| Seasonal Average Temperature for province "Nam Dinh" (Summer): | 30.79 |
| Seasonal Average Temperature for province "Nam Dinh" (Autumn): | 26.27 |
| Seasonal Average Temperature for province "Nam Dinh" (Winter): | 19.52 |
| Seasonal Average Rain for province "Nam Dinh" (Spring):        | 1.92  |
| Seasonal Average Rain for province "Nam Dinh" (Summer):        | 5.52  |
| Seasonal Average Rain for province "Nam Dinh" (Autumn):        | 3.51  |
| Seasonal Average Rain for province "Nam Dinh" (Winter):        | 0.80  |
| Seasonal Average Humidity for province "Nam Dinh" (Spring):    | 78.86 |
| Seasonal Average Humidity for province "Nam Dinh" (Summer):    | 75.01 |
| Seasonal Average Humidity for province "Nam Dinh" (Autumn):    | 76.59 |
| Seasonal Average Humidity for province "Nam Dinh" (Winter):    | 74.21 |
| Seasonal Average Heat Index for province "Nam Dinh" (Spring):  | 29.06 |
| Seasonal Average Heat Index for province "Nam Dinh" (Summer):  | 38.42 |
| Seasonal Average Heat Index for province "Nam Dinh" (Autumn):  | 28.77 |
| Seasonal Average Heat Index for province "Nam Dinh" (Winter):  | 19.77 |

- Thống kê giá trị trung bình, cực trị của các yếu tố khí hậu của một tỉnh theo năm

|                                             |         |
|---------------------------------------------|---------|
| Total Rain of "Yen Bai" in "2020":          | 2236.50 |
| Max Temperature of "Yen Bai" in "2020":     | 40.00   |
| Min Temperature of "Yen Bai" in "2020":     | 7.00    |
| Average Temperature of "Yen Bai" in "2020": | 25.72   |
| Max Humidity of "Yen Bai" in "2020":        | 96.00   |
| Min Humidity of "Yen Bai" in "2020":        | 38.00   |
| Average Humidity of "Yen Bai" in "2020":    | 74.15   |
| Average Temperature of "Yen Bai" in "2020": | 25.72   |
| Average Wind speed of "Yen Bai" in "2020":  | 8.72    |
| Heat Index of "Yen Bai" in "2020":          | 26.99   |

- Thông tin về tỉnh có yếu tố khí hậu cao nhất và thấp nhất

|                                                   |                                        |
|---------------------------------------------------|----------------------------------------|
| Province with highest yearly average temperature: | Chau Doc, 2015 with 29.804109589041097 |
| Province with lowest yearly average temperature:  | Da Lat, 2015 with 20.376712328767123   |
| Province with highest yearly average humidity:    | Da Lat, 2009 with 91.75616438356164    |
| Province with lowest yearly average humidity:     | Bien Hoa, 2021 with 64.24852071005917  |
| Province with highest yearly total rain:          | Da Lat, 2017 with 7376.199999999999    |
| Province with lowest yearly total rain:           | Soc Trang, 2021 with 279.2999999999999 |
| Province with highest daily average rain:         | Da Lat, 2017 with 20.208767123287664   |

- Xu hướng thay đổi lượng mưa và nhiệt độ trung bình của từng tỉnh (giảm: Decreasing; tăng: Increasing) , thông tin về tỉnh có lượng mưa hoặc nhiệt độ cực trị trong 5 năm, 10 năm

|                                                                        |                                                                                   |
|------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| Temperature Trend for province "Nam Dinh":                             | Increasing (Mann-Kendall: 0.5490799884779067, Sen's Slope: 0.007169890982811523)  |
| Rain Trend for province "Nam Dinh":                                    | Decreasing (Mann-Kendall: -2.135311066302971, Sen's Slope: -0.099963176970553)    |
| Humidity Trend for province "Nam Dinh":                                | Decreasing (Mann-Kendall: -1.1591688645644698, Sen's Slope: -0.23517944323650744) |
| Heat Index Trend for province "Nam Dinh":                              | Increasing (Mann-Kendall: 0.5490799884779067, Sen's Slope: 0.012921280267004245)  |
| 5-Year Cycle Average Temperature for province "Nam Dinh" (2009–2013):  | 25.39                                                                             |
| 10-Year Cycle Average Temperature for province "Nam Dinh" (2009–2018): | 25.65                                                                             |
| 5-Year Cycle Average Rain for province "Nam Dinh" (2009–2013):         | 3.28                                                                              |
| 10-Year Cycle Average Rain for province "Nam Dinh" (2009–2018):        | 3.07                                                                              |
| 5-Year Cycle Average Humidity for province "Nam Dinh" (2009–2013):     | 77.56                                                                             |
| 10-Year Cycle Average Humidity for province "Nam Dinh" (2009–2018):    | 76.16                                                                             |
| 5-Year Cycle Average Heat Index for province "Nam Dinh" (2009–2013):   | 26.55                                                                             |
| 10-Year Cycle Average Heat Index for province "Nam Dinh" (2009–2018):  | 26.94                                                                             |

- Hệ số tương quan giữa các yếu tố khí hậu của 1 tỉnh trong năm

```

Correlation between Temperature and Humidity for "Yen Bai,2019": 0.9706
Correlation between Temperature and Rain for "Yen Bai,2019": 0.4598
Correlation between Temperature and Cloud for "Yen Bai,2019": 0.7899
Correlation between Rain and Cloud for "Yen Bai,2019": 0.4296
Correlation between Rain and Humidity for "Yen Bai,2019": 0.4748

```

- Kết quả đầu ra được lưu vào file weather-output.txt

```

hson017828@Hson:~$ ls
KMeans.jar VNweather.jar WordCount.jar data.txt points_1.txt result.txt
NaiveBayes.jar VNweather_data.txt data-kmeans.txt iris.csv points_2.txt
hson017828@Hson:~$ hadoop fs -get /weather-output/part-r-00000 ~/
hson017828@Hson:~$ ls
KMeans.jar VNweather.jar WordCount.jar data.txt part-r-00000 points_2.txt
NaiveBayes.jar VNweather_data.txt data-kmeans.txt iris.csv points_1.txt result.txt
hson017828@Hson:~$ mv ~/part-r-00000 ~/weather-output.txt
hson017828@Hson:~$ ls
KMeans.jar VNweather.jar WordCount.jar data.txt points_1.txt result.txt
NaiveBayes.jar VNweather_data.txt data-kmeans.txt iris.csv points_2.txt weather-output.txt

```

## 3.2 Demo Spark:

### 1. Dữ liệu đầu vào

Sử dụng dữ liệu giống với MapReduce:

```
df = spark.read.csv("/content/drive/MyDrive/dataset/VNweather_data.txt", header=True,
inferSchema=True)
```

### 2. Kết quả

#### a. Trung bình cực trị theo tỉnh

Trung bình cực trị của temp theo tỉnh:

| province         | avg_max_temp | avg_min_temp |
|------------------|--------------|--------------|
| Long Xuyen       | 34.5         | 21.0         |
| My Tho           | 35.0         | 20.5         |
| Ho Chi Minh City | 34.5         | 21.0         |
| Ben Tre          | 35.0         | 20.5         |
| Nha Trang        | 31.5         | 18.0         |
| Hong Gail        | 32.0         | 6.5          |
| Hai Phong        | 34.0         | 6.0          |
| Cam Pha          | 31.5         | 6.5          |
| Hai Duong        | 35.0         | 7.0          |
| Ca Mau           | 33.5         | 20.0         |
| Cam Ranh         | 31.5         | 18.0         |
| Play Cu          | 31.0         | 15.0         |
| Nam Dinh         | 37.0         | 5.5          |
| Chau Doc         | 34.5         | 19.5         |
| Can Tho          | 34.0         | 20.5         |
| Da Lat           | 27.5         | 12.0         |
| Qui Nhon         | 32.5         | 15.5         |
| Phan Thiet       | 32.0         | 21.0         |
| Tan An           | 35.0         | 21.5         |
| Hue              | 33.5         | 13.5         |

only showing top 20 rows

**b. Trung bình cực trị theo mùa**

Trung bình cực trị của humidity theo tỉnh và mùa:

| province      | season | avg_max_humidity | avg_min_humidity |
|---------------|--------|------------------|------------------|
| Hue           | Summer | 95.0             | 58.0             |
| Tan An        | Spring | 94.0             | 47.0             |
| Buon Me Thuot | Autumn | 99.0             | 64.0             |
| Nam Dinh      | Spring | 97.0             | 38.0             |
| Hong Gai      | Spring | 97.0             | 48.0             |
| Rach Gia      | Summer | 95.0             | 66.0             |
| Long Xuyen    | Autumn | 96.0             | 57.0             |
| Play Cu       | Summer | 100.0            | 70.0             |
| Hue           | Winter | 97.0             | 60.0             |
| Bac Lieu      | Spring | 91.0             | 54.0             |
| Cam Pha       | Spring | 97.0             | 48.0             |
| Bac Lieu      | Winter | 92.0             | 51.0             |
| Chau Doc      | Summer | 96.0             | 61.0             |
| Hanoi         | Winter | 97.0             | 28.0             |
| Bien Hoa      | Winter | 96.0             | 45.0             |
| Ha Noi        | Winter | 97.0             | 28.0             |
| Phan Thiet    | Autumn | 94.0             | 63.0             |
| Ca Mau        | Spring | 92.0             | 56.0             |
| Hai Phong     | Summer | 93.0             | 57.0             |
| Cam Pha       | Winter | 96.0             | 32.0             |

only showing top 20 rows

### c. Trung bình cực trị của tỉnh theo năm

Trung bình cực trị của wind theo tỉnh và năm:

| province   | year | avg_max_wind | avg_min_wind |
|------------|------|--------------|--------------|
| Ha Noi     | 2013 | 16.0         | 2.0          |
| Bien Hoa   | 2021 | 18.0         | 4.0          |
| Ha Noi     | 2011 | 23.0         | 3.0          |
| Hoa Binh   | 2012 | 12.0         | 2.0          |
| Hoa Binh   | 2020 | 10.0         | 2.0          |
| Cam Pha    | 2020 | 29.0         | 4.0          |
| Chau Doc   | 2011 | 24.0         | 3.0          |
| Play Cu    | 2014 | 20.0         | 3.0          |
| Hoa Binh   | 2011 | 13.0         | 2.0          |
| Ben Tre    | 2013 | 23.0         | 4.0          |
| Cam Pha    | 2011 | 39.0         | 5.0          |
| Hanoi      | 2011 | 23.0         | 3.0          |
| Nha Trang  | 2016 | 31.0         | 4.0          |
| Nha Trang  | 2020 | 28.0         | 4.0          |
| Cam Ranh   | 2020 | 44.0         | 5.0          |
| Hai Duong  | 2015 | 27.0         | 5.0          |
| Long Xuyen | 2019 | 26.0         | 5.0          |
| Tam Ky     | 2018 | 25.0         | 5.0          |
| Can Tho    | 2012 | 22.0         | 4.0          |
| Da Lat     | 2020 | 15.0         | 1.0          |

only showing top 20 rows

#### d. Yếu tố khí hậu cao, thấp nhất của tỉnh

Yếu tố maxtemp:

Cao nhất: 46.0 ở tỉnh Hoa Binh  
Thấp nhất: 4.0 ở tỉnh Hoa Binh

Yếu tố mintemp:

Cao nhất: 32.0 ở tỉnh Ha Noi  
Thấp nhất: 2.0 ở tỉnh Ha Noi

Yếu tố wind:

Cao nhất: 54.0 ở tỉnh Tam Ky  
Thấp nhất: 1.0 ở tỉnh Da Lat

Yếu tố rain:

Cao nhất: 596.4 ở tỉnh Cam Ranh  
Thấp nhất: 0.0 ở tỉnh Bac Lieu

Yếu tố humidity:

Cao nhất: 100.0 ở tỉnh Da Lat  
Thấp nhất: 23.0 ở tỉnh Hoa Binh

Yếu tố cloud:

Cao nhất: 100.0 ở tỉnh Buon Me Thuot  
Thấp nhất: 0.0 ở tỉnh Bac Lieu

Yếu tố pressure:

Cao nhất: 1038.0 ở tỉnh Thai Nguyen  
Thấp nhất: 988.0 ở tỉnh Tam Ky

Yếu tố temp:

Cao nhất: 37.0 ở tỉnh Ha Noi  
Thấp nhất: 3.5 ở tỉnh Hoa Binh

### e. Tính có lượng mưa hoặc nhiệt độ cực trị trong 5 năm, 10 năm

Trong 5 năm gần đây (2017-2021):

Lượng mưa cao nhất: 596.4 ở tỉnh Cam Ranh

Lượng mưa thấp nhất: 0.0 ở tỉnh Bac Lieu

Nhiệt độ trung bình cao nhất: 36.5 ở tỉnh Ha Noi

Nhiệt độ trung bình thấp nhất: 4.5 ở tỉnh Uong Bi

Trong 10 năm gần đây (2012-2021):

Lượng mưa cao nhất: 596.4 ở tỉnh Cam Ranh

Lượng mưa thấp nhất: 0.0 ở tỉnh Bac Lieu

Nhiệt độ trung bình cao nhất: 37.0 ở tỉnh Ha Noi

Nhiệt độ trung bình thấp nhất: 3.5 ở tỉnh Hoa Binh

### f. Hệ số tương quan

Hệ số tương quan giữa các yếu tố khí hậu của tỉnh Ha Noi trong năm 2019:

|          | maxtemp | mintemp | wind  | rain  | humidity | cloud | pressure | temp  |
|----------|---------|---------|-------|-------|----------|-------|----------|-------|
| maxtemp  | 1.00    | 0.90    | -0.05 | 0.05  | -0.10    | -0.66 | -0.82    | 0.98  |
| mintemp  | 0.90    | 1.00    | 0.02  | 0.24  | 0.20     | -0.40 | -0.89    | 0.97  |
| wind     | -0.05   | 0.02    | 1.00  | 0.08  | 0.17     | 0.21  | 0.04     | -0.02 |
| rain     | 0.05    | 0.24    | 0.08  | 1.00  | 0.41     | 0.16  | -0.33    | 0.14  |
| humidity | -0.10   | 0.20    | 0.17  | 0.41  | 1.00     | 0.60  | -0.19    | 0.03  |
| cloud    | -0.66   | -0.40   | 0.21  | 0.16  | 0.60     | 1.00  | 0.35     | -0.56 |
| pressure | -0.82   | -0.89   | 0.04  | -0.33 | -0.19    | 0.35  | 1.00     | -0.87 |
| temp     | 0.98    | 0.97    | -0.02 | 0.14  | 0.03     | -0.56 | -0.87    | 1.00  |

## CHƯƠNG IV: KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN

### 4.1 Kết luận

Báo cáo này đã trình bày cách sử dụng **Hadoop MapReduce** và **Apache Spark** để xử lý và phân tích lượng lớn dữ liệu nhiệt độ, nhằm tính toán và phân tích dữ liệu thời tiết của các tỉnh ở Việt Nam theo từng năm. Hai công nghệ này đều mạnh mẽ trong việc xử lý dữ liệu lớn, nhưng mỗi công cụ có những ưu và nhược điểm riêng, phụ thuộc vào yêu cầu cụ thể của bài toán.

#### 1. Hadoop MapReduce:

- Cung cấp một mô hình xử lý phân tán dựa trên việc chia công việc thành các cặp **Map** và **Reduce**.
- Tối ưu cho xử lý dữ liệu lô (batch processing), phù hợp với các bài toán phân tích dữ liệu thời tiết truyền thống, nơi dữ liệu có thể được xử lý sau khi thu thập đầy đủ.
- Nhược điểm là độ trễ cao, do dữ liệu trung gian được lưu trữ và truy xuất từ đĩa, làm giảm hiệu suất xử lý khi lượng dữ liệu tăng.

#### 2. Apache Spark:

- Sử dụng bộ nhớ (RAM) để xử lý dữ liệu trung gian, giảm độ trễ và tăng hiệu suất.
- Hỗ trợ nhiều tính năng mở rộng như xử lý dữ liệu thời gian thực (real-time processing) qua Spark Streaming, và tích hợp thư viện MLlib cho các ứng dụng học máy.

#### 3. So sánh hiệu quả giữa MapReduce và Spark trong phân tích dữ liệu thời tiết

| Tiêu chí                    | Hadoop MapReduce                                         | Apache Spark                                           |
|-----------------------------|----------------------------------------------------------|--------------------------------------------------------|
| <b>Hiệu suất xử lý</b>      | Chậm hơn, do phụ thuộc vào ghi/đọc dữ liệu từ đĩa.       | Nhanh hơn nhờ xử lý trong bộ nhớ.                      |
| <b>Xử lý thời gian thực</b> | Không hỗ trợ.                                            | Hỗ trợ tốt với Spark Streaming.                        |
| <b>Độ phức tạp</b>          | Dễ triển khai cho xử lý dữ liệu lô.                      | Phức tạp hơn khi tích hợp thời gian thực hoặc học máy. |
| <b>Ứng dụng học máy</b>     | Không tích hợp sẵn, cần triển khai thủ công.             | Tích hợp thư viện MLlib hỗ trợ trực tiếp.              |
| <b>Khả năng mở rộng</b>     | Tốt cho các cụm máy lớn, nhưng có độ trễ cao.            | Tốt hơn nhờ kiến trúc tối ưu hóa DAG.                  |
| <b>Chi phí tài nguyên</b>   | Yêu cầu thấp hơn, tối ưu cho các hệ thống sử dụng ổ đĩa. | Yêu cầu cao hơn về bộ nhớ và CPU.                      |

#### 4. Ưu và nhược điểm của MapReduce và Spark trong bài toán phân tích dữ liệu thời tiết

##### - MapReduce:

###### • Ưu điểm:

- **Tính ổn định và tin cậy:** MapReduce đã được sử dụng rộng rãi và kiểm chứng qua thời gian, phù hợp với xử lý dữ liệu lô.
- **Khả năng chịu lỗi:** Dữ liệu được lưu trữ trên HDFS với tính năng tự động sao lưu, đảm bảo an toàn trong trường hợp lỗi phần cứng.
- **Đề dang triển khai:** MapReduce phù hợp cho các bài toán phân tích cơ bản, nơi không yêu cầu xử lý thời gian thực hoặc học máy.

###### • Nhược điểm:

- **Độ trễ cao:** Phụ thuộc vào việc ghi/đọc dữ liệu từ đĩa, dẫn đến thời gian xử lý lâu hơn.
- **Hạn chế linh hoạt:** Không phù hợp cho các ứng dụng học máy hoặc phân tích phức tạp cần nhiều vòng lặp tính toán.

##### - Spark:

###### • Ưu điểm:

- **Hiệu năng cao:** Tăng tốc độ xử lý nhờ sử dụng bộ nhớ và tối ưu hóa DAG.
- **Linh hoạt:** Hỗ trợ nhiều tính năng mở rộng như học máy, xử lý đồ thị (GraphX), và phân tích thời gian thực.
- **Đa dạng nguồn dữ liệu:** Hỗ trợ kết nối với nhiều loại hệ thống lưu trữ như HDFS, S3, Kafka.

###### • Nhược điểm:

- **Chi phí tài nguyên:** Đòi hỏi nhiều RAM và CPU để đạt hiệu suất cao.
- **Phức tạp hơn khi triển khai:** Yêu cầu kỹ năng cao hơn để tối ưu hóa và quản lý.

## 4.2 Hướng phát triển

### 1. Tối ưu hóa hiệu suất hệ thống với MapReduce

#### • Áp dụng cơ chế nén dữ liệu trên HDFS:

- Sử dụng các thuật toán như Snappy hoặc Parquet để giảm kích thước tệp, tăng tốc độ truyền tải dữ liệu.

- **Tối ưu hóa cấu hình MapReduce:**
  - Điều chỉnh kích thước block của HDFS, số lượng mapper/reducer, và tài nguyên dành cho các node xử lý.
- 2. **Mở rộng phạm vi bài toán**
  - **Phân tích dữ liệu khí hậu phức tạp hơn:**
    - Xây dựng thêm các chỉ số như nhiệt độ cao nhất/thấp nhất theo mùa hoặc xu hướng nhiệt độ theo khu vực địa lý.
  - **Phân tích dữ liệu không gian:**
    - Tích hợp MapReduce với các công cụ GIS để phân tích nhiệt độ theo vị trí địa lý cụ thể.
- 3. **Tăng cường khả năng trực quan hóa dữ liệu**
  - **Sử dụng công cụ trực quan hóa:**
    - Tích hợp MapReduce với các công cụ như Tableau hoặc Power BI để tạo báo cáo nhiệt độ tương tác.
- 4. **Ứng dụng học máy (kết hợp MapReduce và Spark)**
  - **Dự đoán nhiệt độ:**
    - Sử dụng Spark MLlib để xây dựng mô hình dự đoán trên dữ liệu được xử lý trước bởi MapReduce.
  - **Phát hiện bất thường:**
    - Áp dụng các thuật toán học không giám sát để tìm các biến động nhiệt độ bất thường trong dữ liệu lịch sử.

## TÀI LIỆU THAM KHẢO

- [1] <https://github.com/needmukesh/Hadoop-Books/>
- [2] <https://www.slideshare.net/tantrieuf31/tng-quan-v-d-liu-ln-bigdata>
- [3] <https://stackoverflow.com/questions/48298798/how-does-cleanup-method-work?>
- [4] [https://vi.wikipedia.org/wiki/Ch%E1%BB%89\\_s%E1%BB%91\\_n%C3%B3ng\\_b%E1%BB%A9c](https://vi.wikipedia.org/wiki/Ch%E1%BB%89_s%E1%BB%91_n%C3%B3ng_b%E1%BB%A9c)
- [5] [https://hadoop.apache.org/docs/r1.2.1/mapred\\_tutorial.html](https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html)
- [6] <https://www.geeksforgeeks.org/hashmap-class-methods-java-examples-set-1-put-get-isempty-size/>

## Phân chia công việc

| STT | Họ và tên           | MSV      | Công việc                                                                                                  |
|-----|---------------------|----------|------------------------------------------------------------------------------------------------------------|
| 1   | Bàn Hoàng Sơn       | 22022651 | Phân chia công việc<br>Cài đặt và thử nghiệm MapReduce<br>Quay Demo MapReduce<br>Viết báo cáo và làm slide |
| 2   | Nguyễn Bảo Sơn      | 22022613 | Tìm hiểu tổng quan MapReduce, HDFS<br>Cài đặt và thử nghiệm MapReduce<br>Viết báo cáo và làm slide         |
| 3   | Lý Quốc An          | 22022660 | Tìm hiểu tổng quan Spark<br>Cài đặt và thử nghiệm Spark<br>Viết báo cáo và làm slide                       |
| 4   | Cao Đặng Quốc Vương | 22022601 | Phân chia công việc<br>Cài đặt và thử nghiệm Spark<br>Quay Demo Spark<br>Viết báo cáo và làm slide         |