

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &  
TRUYỀN THÔNG VIỆT HÀN  
KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO CUỐI KÌ  
HỌC PHẦN: LẬP TRÌNH SONG SONG  
ĐỀ TÀI: ĐẾM TỪ VỚI HADOOP MAPREDUCE**

Nhóm : Phan Thị Quỳnh – 21AD050  
Lớp : 21AD  
Giảng viên : TS. Nguyễn Đình Lầu

*Đà Nẵng, tháng 5 năm 2025*

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &  
TRUYỀN THÔNG VIỆT HÀN  
KHOA KHOA HỌC MÁY TÍNH**



**BÁO CÁO CUỐI KÌ**  
**HỌC PHẦN: LẬP TRÌNH SONG SONG**  
**ĐỀ TÀI: ĐẾM TỪ VỚI HADOOP MAPREDUCE**

Nhóm : Phan Thị Quỳnh – 21AD050  
Lớp : 21AD  
Giảng viên : TS. Nguyễn Đình Lầu

*Đà Nẵng, tháng 5 năm 2025*

## NHẬN XÉT CỦA GIẢNG VIÊN

---

---

---

---

---

---

---

---

---

*Giảng viên bộ môn*

**TS. Nguyễn Đình Lâu**

## LỜI CẢM ƠN

*Em xin bày tỏ lòng biết ơn chân thành và sâu sắc tới thầy **TS. Nguyễn Đình Lâu**, cảm ơn sự hướng dẫn và giúp đỡ tận tình của thầy trong suốt thời gian thực hiện bài tập lớn cũng như bài báo cáo này.*

*Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế của bản thân, em không tránh được những thiếu sót. Em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để em có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn trong công việc thực tế sau này.*

***Em xin chân thành cảm ơn!***

*Sinh viên*

*Phan Thị Quỳnh*

# MỤC LỤC

<b>MỞ ĐẦU</b>	<b>1</b>
1. Lý do chọn đề tài	1
2. Mục tiêu của đề tài	1
3. Đối tượng và phạm vi nghiên cứu	2
4. Cách tiếp cận	2
5. Phương pháp nghiên cứu	2
6. Nội dung của đề tài	2
<b>CHƯƠNG 1 - TỔNG QUAN VỀ XỬ LÝ SONG SONG VÀ PHÂN TÁN.</b>	<b>3</b>
1.1. Giới thiệu về xử lý song song và phân tán	3
1.2. So sánh xử lý song song và xử lý phân tán	3
1.3. Mô hình lập trình song song và phân tán.	4
1.3.1. Mô hình lập trình song song	4
1.3.2. Mô hình lập trình phân tán	4
1.4. Ưu điểm và Thách thức của Xử lý Song Song và Phân Tán	5
1.5. Kiến trúc hệ thống phân tán.	5
1.6. Ứng dụng của xử lý song song và phân tán	6
<b>CHƯƠNG 2 - NGHIÊN CỨU MÔ HÌNH MAPREDUCE.</b>	<b>7</b>
2.1. Giới thiệu về Hadoop.	7
2.2. Kiến trúc của HDFS.	8
2.3. Xử lý dữ liệu lớn với MapReduce.	12
2.4. Nguyên tắc hoạt động của MapReduce	12
2.5. Cài đặt Hadoop	13
2.5.1. Cài đặt JDK	13

2.5.2. Cài đặt Hadoop	22
<b>CHƯƠNG 3: BÀI TOÁN ĐẾM TỪ TRÊN CẤU TRÚC MAPREDUCE</b>	<b>28</b>
3.1. Bài toán đếm từ truyền thống	28
3.1.1. Thuật toán	28
3.1.2. Ví dụ cụ thể	28
3.2. Bài toán đếm từ trên MapReduce	29
3.2.1. Thuật toán trên Map	30
3.2.2. Thuật toán trên Reduce	31
3.2.3. Ví dụ đồ thị cụ thể cho thuật toán Map và Reduce ở ví dụ được đề cập ở phần 3.1.2	33
<b>CHƯƠNG 4: XÂY DỰNG DỮ LIỆU VÀ ĐỀ MODE</b>	<b>38</b>
4.1. Xây dựng code chạy bài toán đếm từ (Word Count)	38
4.2. Demo kết quả	40
<b>KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN</b>	<b>43</b>
<b>TÀI LIỆU THAM KHẢO</b>	<b>44</b>

## DANH MỤC HÌNH ẢNH

Hình 1: Tổng quát về kiến trúc của HDFS. ....	8
Hình 2: Quá trình ghi file vào HDFS. ....	10
Hình 3: Quá trình đọc file từ HDFS. ....	11
Hình 4: Hướng dẫn tải JDK 8u201_(1).....	13
Hình 5: Hướng dẫn tải JDK 8u201_(2).....	14
Hình 6: Tiến hành cài đặt JDK-(1).....	14
Hình 7: Tiến hành cài đặt JDK-(2).....	15
Hình 8: Tiến hành cài đặt JDK-(3).....	15
Hình 9: Tiến hành cài đặt JDK-(4).....	16
Hình 10: Tiến hành cài đặt JDK-(5).....	16
Hình 11: Tiến hành cài đặt JDK-(6).....	17
Hình 12: Hoàn thành đăng kí JDK. ....	17
Hình 13: Start menu, search “Edit the system enviroment”. ....	18
Hình 14: Hiện thị cửa sổ System Properties.....	18
Hình 15: Thiết lập môi trường Java JDK – (1). ....	19
Hình 16: Thiết lập môi trường Java JDK – (2). ....	19
Hình 17: Thiết lập môi trường Java JDK – (3). ....	20
Hình 18: Thiết lập môi trường Java JDK – (4). ....	20
Hình 19: Hoàn thành việc thiết lập môi trường JDK. ....	21
Hình 20: Giải nén gói Hadoop vào ổ C:/hadoop. ....	22
Hình 21: Đường dẫn mới cho Hadoop. ....	22
Hình 22: Thiết lập các biến mới cho Hadoop. ....	23
Hình 23: Thiết lập Hadoop thành công. ....	23
Hình 24: Định dạng namenode và datanode. ....	26
Hình 25: Gõ lệnh chạy hadoop.....	26
Hình 26: Khởi động thành công hadoop. ....	27

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Trong thời đại công nghệ số hiện nay, lượng dữ liệu sinh ra từ các hệ thống thông tin, mạng xã hội, thương mại điện tử và nhiều lĩnh vực khác đang tăng lên theo cấp số nhân. Việc xử lý và khai thác dữ liệu lớn (Big Data) đòi hỏi những công nghệ tiên tiến nhằm đảm bảo tốc độ xử lý nhanh, hiệu suất cao và khả năng mở rộng linh hoạt.

Hadoop là một hệ thống lưu trữ và xử lý dữ liệu phân tán phổ biến, trong đó Hadoop MapReduce là một mô hình lập trình mạnh mẽ giúp xử lý dữ liệu song song trên nhiều máy chủ. Một trong những bài toán kinh điển trong xử lý dữ liệu văn bản là đếm từ, đây là bước quan trọng trong nhiều ứng dụng như tìm kiếm thông tin, phân tích dữ liệu, xử lý ngôn ngữ tự nhiên (NLP) và khai phá dữ liệu.

Nhận thấy tầm quan trọng của vấn đề này, nhóm chúng tôi lựa chọn đề tài "Đếm từ với Hadoop MapReduce" nhằm nghiên cứu cách thức xử lý dữ liệu văn bản lớn một cách hiệu quả bằng cách áp dụng lập trình song song trên hệ thống phân tán.

## 2. Mục tiêu của đề tài

- **Nghiên cứu chuyên sâu:** Tìm hiểu chi tiết về mô hình lập trình MapReduce – một thành phần quan trọng trong hệ sinh thái Hadoop – và cách thức hoạt động của nó trong việc xử lý dữ liệu văn bản lớn theo phương pháp phân tán.
- **Áp dụng thực tế:** Áp dụng kiến thức về MapReduce để giải quyết bài toán đếm số lần xuất hiện của từng từ trong một tập dữ liệu văn bản lớn, từ đó minh họa khả năng xử lý song song hiệu quả của mô hình này.
- **Đánh giá hiệu năng:** Triển khai bài toán đếm từ trên môi trường Hadoop, thực hiện thử nghiệm với các tập dữ liệu có kích thước khác nhau nhằm đánh giá hiệu suất, khả năng mở rộng và độ ổn định của hệ thống.
- **Đóng góp kiến thức:** Góp phần giúp người thực hiện nắm vững mô hình MapReduce và kỹ thuật xử lý dữ liệu lớn, đồng thời cung cấp một ví dụ điển hình cho việc ứng dụng mô hình này trong các tác vụ phân tích văn bản trong thực tế.



### 3. Đối tượng và phạm vi nghiên cứu

- **Đối tượng nghiên cứu:** Công nghệ Hadoop, mô hình lập trình MapReduce và ứng dụng của nó trong xử lý dữ liệu lớn.
- **Phạm vi nghiên cứu:** Đề tài tập trung vào việc triển khai thuật toán đếm từ trên Hadoop MapReduce, thử nghiệm trên dữ liệu văn bản lớn và phân tích kết quả.

### 4. Cách tiếp cận

- Nghiên cứu tài liệu về Hadoop và MapReduce.
- Cài đặt môi trường Hadoop trên hệ thống phân tán.
- Phát triển ứng dụng đếm từ sử dụng MapReduce.
- Chạy thử nghiệm và đánh giá hiệu suất.

### 5. Phương pháp nghiên cứu

- **Phương pháp lý thuyết:** Tìm hiểu tài liệu về Hadoop, MapReduce và các công nghệ liên quan.
- **Phương pháp thực nghiệm:** Cài đặt, triển khai chương trình đếm từ, đo lường thời gian xử lý và đánh giá hiệu suất.
- **Phương pháp so sánh:** Đối chiếu hiệu suất giữa MapReduce và phương pháp xử lý tuần tự truyền thống.

### 6. Nội dung của đề tài

- Chương 1: Tổng Quan về xử lý song song và phân tán.
- Chương 2: Nghiên cứu mô hình MapReduce.
- Chương 3: Bài toán đếm từ cấu trúc MapReduce.
- Chương 4: Xây dựng dữ liệu và Demo.

# CHƯƠNG 1 - TỔNG QUAN VỀ XỬ LÝ SONG SONG VÀ PHÂN TÁN.

Trong kỷ nguyên dữ liệu lớn (Big Data), việc xử lý hiệu quả lượng thông tin khổng lồ trở thành một thách thức lớn. Xử lý song song và phân tán nổi lên như những giải pháp quan trọng để giải quyết vấn đề này. Chương này sẽ cung cấp một cái nhìn tổng quan về các khái niệm, mô hình và công cụ liên quan đến xử lý song song và phân tán, đồng thời giới thiệu về Hadoop, một framework mạnh mẽ trong lĩnh vực này.

## 1.1. Giới thiệu về xử lý song song và phân tán

Trong lĩnh vực khoa học máy tính, khi khối lượng dữ liệu ngày càng lớn và các bài toán ngày càng phức tạp, việc xử lý trên một máy tính đơn lẻ không còn đáp ứng được yêu cầu về hiệu suất. Do đó, xử lý song song và xử lý phân tán ra đời như một giải pháp để tăng tốc độ tính toán, tối ưu hóa tài nguyên và cải thiện hiệu quả xử lý dữ liệu lớn.

- **Xử lý song song (Parallel Computing):** Là kỹ thuật trong đó một bài toán được chia thành nhiều phần nhỏ và được thực thi đồng thời trên nhiều bộ xử lý hoặc nhiều lõi CPU trong cùng một hệ thống.
- **Xử lý phân tán (Distributed Computing):** Là mô hình trong đó dữ liệu và tác vụ tính toán được chia nhỏ và thực thi trên nhiều máy tính khác nhau được kết nối trong một mạng, mỗi máy tính có thể có bộ nhớ và tài nguyên riêng biệt.

Hệ thống xử lý song song và phân tán ngày càng trở nên quan trọng trong nhiều lĩnh vực như phân tích dữ liệu lớn (Big Data), trí tuệ nhân tạo (AI), mô phỏng khoa học và thương mại điện tử.

## 1.2. So sánh xử lý song song và xử lý phân tán

Đặc điểm	Xử lý song song	Xử lý phân tán
Cấu trúc hệ thống	Thực thi trên một hệ thống duy nhất với nhiều bộ xử lý (đa lõi, siêu phân luồng)	Thực thi trên nhiều hệ thống máy tính khác nhau, kết nối qua mạng

<b>Cách thức thực hiện</b>	Chia bài toán thành nhiều phần và chạy đồng thời trên nhiều lõi CPU hoặc nhiều GPU	Chia bài toán thành nhiều tác vụ và phân phối cho các máy tính khác nhau
<b>Bộ nhớ</b>	Chia sẻ bộ nhớ chung	Bộ nhớ riêng biệt giữa các nút trong hệ thống
<b>Tốc độ truyền dữ liệu</b>	Nhanh do sử dụng bộ nhớ chung	Chậm hơn do phụ thuộc vào mạng
<b>Khả năng mở rộng</b>	Giới hạn do phụ thuộc vào số lõi CPU/GPU	Cao vì có thể thêm nhiều máy tính vào hệ thống

### 1.3. Mô hình lập trình song song và phân tán.

#### 1.3.1. Mô hình lập trình song song

- **Thread-based Parallelism (Đa luồng):** Chương trình sử dụng nhiều luồng (threads) để thực hiện các tác vụ đồng thời trong cùng một tiến trình.
- **Message Passing Interface (MPI):** Mô hình giao tiếp qua thông điệp, sử dụng phổ biến trong các hệ thống siêu máy tính.
- **CUDA (Compute Unified Device Architecture):** Công nghệ lập trình song song trên GPU, phổ biến trong AI và đồ họa máy tính.

#### 1.3.2. Mô hình lập trình phân tán

- **MapReduce:** Mô hình lập trình do Google phát triển, giúp xử lý dữ liệu lớn trên hệ thống phân tán bằng cách chia nhỏ công việc thành hai bước chính: Map và Reduce. Đây là nền tảng của Hadoop.
- **Apache Spark:** Một nền tảng xử lý dữ liệu phân tán mạnh mẽ, cho phép xử lý dữ liệu lớn nhanh hơn MapReduce.
- **Actor Model:** Mô hình lập trình trong đó các "actor" giao tiếp với nhau bằng cách gửi và nhận tin nhắn, phổ biến trong các hệ thống thời gian thực.

## 1.4. Ưu điểm và Thách thức của Xử lý Song Song và Phân Tán

- **Ưu điểm:**

- **Tăng tốc độ xử lý:** Tận dụng sức mạnh của nhiều bộ xử lý để thực hiện các tác vụ đồng thời, giúp giảm thời gian xử lý.
- **Khả năng mở rộng:** Dễ dàng thêm tài nguyên tính toán (bộ xử lý, bộ nhớ) để đáp ứng nhu cầu xử lý ngày càng tăng.
- **Tính chịu lỗi:** Nếu một thành phần trong hệ thống bị lỗi, các thành phần khác vẫn có thể tiếp tục hoạt động, đảm bảo tính liên tục của hệ thống.
- **Giải quyết bài toán lớn:** Cho phép xử lý các bài toán có dữ liệu quá lớn không thể xử lý trên một máy tính đơn lẻ.

- **Thách thức:**

- **Độ phức tạp:** Việc thiết kế và lập trình các ứng dụng song song và phân tán thường phức tạp hơn so với các ứng dụng tuần tự, đòi hỏi phải quản lý đồng bộ, truyền thông và xử lý lỗi giữa các thành phần.
- **Cân bằng tải:** Đảm bảo rằng công việc được phân phối đều giữa các bộ xử lý để tránh tình trạng một số bộ xử lý bị quá tải trong khi các bộ xử lý khác lại nhàn rỗi.
- **Hiệu suất:** Đôi khi việc chia nhỏ công việc và xử lý song song có thể dẫn đến chi phí liên lạc và đồng bộ hóa giữa các thành phần, làm giảm hiệu suất tổng thể của hệ thống.

## 1.5. Kiến trúc hệ thống phân tán.

Một hệ thống phân tán bao gồm nhiều máy tính (nút - nodes) liên kết với nhau qua mạng, hoạt động theo các kiến trúc chính như sau:

- Kiến trúc Client-Server: Máy khách gửi yêu cầu và máy chủ xử lý.
- Kiến trúc Peer-to-Peer (P2P): Các nút trong hệ thống đều có vai trò ngang hàng, không có máy chủ trung tâm.
- Kiến trúc Master-Slave: Một nút trung tâm (Master) điều phối công việc cho các nút phụ (Slaves), phổ biến trong Hadoop.

## 1.6. Ứng dụng của xử lý song song và phân tán

Xử lý song song và phân tán có nhiều ứng dụng quan trọng trong thực tế, bao gồm:

- **Phân tích dữ liệu lớn (Big Data Analytics):** Xử lý và phân tích khối lượng dữ liệu lớn như trong hệ thống Hadoop.
- **Trí tuệ nhân tạo (AI) và Machine Learning:** Đào tạo mô hình học máy trên GPU và hệ thống phân tán.
- **Tính toán khoa học:** Mô phỏng vật lý, dự báo thời tiết, nghiên cứu y học.
- **Giao dịch tài chính:** Phân tích rủi ro, giao dịch chứng khoán tốc độ cao.

## CHƯƠNG 2 - NGHIÊN CỨU MÔ HÌNH MAPREDUCE.

### 2.1. Giới thiệu về Hadoop.

Hadoop là một Apache framework nguồn mở viết bằng Java cho phép phát triển các ứng dụng phân tán có cường độ dữ liệu lớn một cách miễn phí. Nó được thiết kế để mở rộng quy mô từ một máy chủ đơn sang hàng ngàn máy tính khác có tính toán và lưu trữ cục bộ (local computation and storage). Hadoop được phát triển dựa trên ý tưởng từ các công bố của Google về mô hình Map-Reduce và hệ thống file phân tán Google File System (GFS). Và có cung cấp cho chúng ta một môi trường song song để thực thi các tác vụ Map-Reduce.

Nhờ có cơ chế streaming mà Hadoop có thể phát triển trên các ứng dụng phân tán bằng cả java lẫn một số ngôn ngữ lập trình khác như C++, Python, Pearl,...

Hadoop có một cấu trúc liên kết master-slave. Trong cấu trúc này, chúng ta có một node master và nhiều node slave. Chức năng của node master là gán một tác vụ cho các node slave khác nhau và quản lý tài nguyên. Các node slave là máy tính thực tế có thể không mạnh lắm. Các node slave lưu trữ dữ liệu thực trong khi trên master chúng ta có metadata.

*Kiến trúc Hadoop gồm có ba lớp chính đó là:*

- **HDFS (Hadoop Distributed File System):** Một hệ thống tập tin phân tán được tối ưu hóa để lưu trữ dữ liệu lớn trên các cụm máy tính. HDFS chia nhỏ dữ liệu thành các khối và phân phối chúng trên nhiều máy tính, đảm bảo tính khả dụng và chịu lỗi cao.
- **YARN (Yet Another Resource Negotiator):** Một thành phần quản lý tài nguyên và lập lịch các tác vụ trên cụm Hadoop. YARN cho phép chạy nhiều ứng dụng khác nhau trên cùng một cụm Hadoop và đảm bảo việc sử dụng tài nguyên một cách hiệu quả.
- **MapReduce:** Một mô hình lập trình song song và phân tán để xử lý dữ liệu lớn. MapReduce chia nhỏ công việc thành các tác vụ nhỏ hơn (map) và sau đó tổng hợp kết quả (reduce) để tạo ra kết quả cuối cùng.

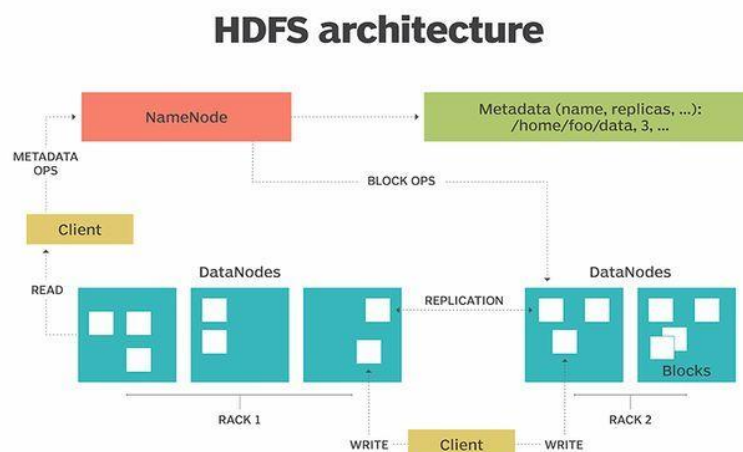
## 2.2. Kiến trúc của HDFS.

HDFS là một hệ thống tập tin phân tán được thiết kế để lưu trữ dữ liệu lớn trên các cụm máy tính. Dữ liệu được chia thành các khối (block) và lưu trữ trên các DataNode. Một NameNode đóng vai trò quản lý metadata của hệ thống, bao gồm thông tin về vị trí của các khối dữ liệu, tên tập tin, thư mục, quyền truy cập,...

### Mục tiêu của HDFS

- Tiết kiệm chi phí cho việc lưu trữ dữ liệu lớn: có thể lưu trữ dữ liệu megabytes đến petabytes, ở dạng có cấu trúc hay không có cấu trúc
- Dữ liệu có độ tin cậy cao và có khả năng khắc phục lỗi: Dữ liệu lưu trữ trong HDFS được nhân bản thành nhiều phiên bản và được lưu tại các DataNode khác nhau, khi có 1 máy bị lỗi thì vẫn còn dữ liệu được lưu tại DataNode khác
- Tính chính xác cao: Dữ liệu lưu trữ trong HDFS thường xuyên được kiểm tra bằng mã checksum được tính trong quá trình ghi file, nếu có lỗi xảy ra sẽ được khôi phục bằng các bản sao
- Khả năng mở rộng: có thể tăng hàng trăm node trong một cluster
- Có throughput cao: tốc độ xử lý truy nhập dữ liệu cao
- Data Locality: xử lý dữ liệu tại chỗ

### HDFS Architecture



Hình 1: Tổng quát về kiến trúc của HDFS.

Với HDFS, dữ liệu được ghi trên 1 máy chủ và có thể đọc lại nhiều lần sau đó tại bất cứ máy chủ khác trong cụm HDFS. HDFS bao gồm 1 Namenode chính và nhiều Datanode kết nối lại thành một cụm (cluster).

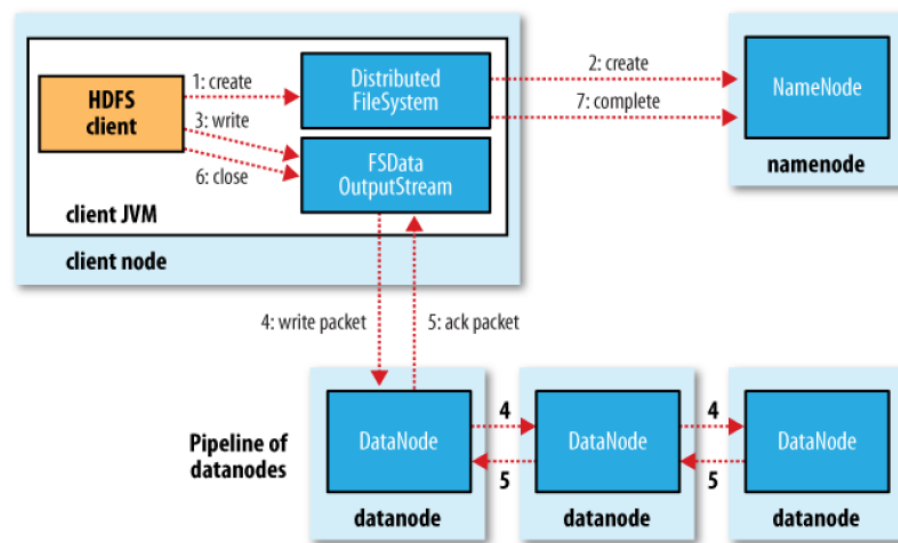
- ***Namenode:*** HDFS chỉ bao gồm duy nhất 1 namenode được gọi là master node thực hiện các nhiệm vụ:
  - Lưu trữ metadata của dữ liệu thực tế (tên, đường dẫn, blocks id, cấu hình datanode vị trí blocks,...).
  - Quản lý không gian tên của hệ thống file ( ánh xạ các file name với các blocks, ánh xạ các block vào các datanode).
  - Quản lý cấu hình của cụm.
  - Chỉ định công việc cho datanode.
- ***Datanode:*** Chức năng của Datanode:
  - Lưu trữ dữ liệu thực tế.
  - Trực tiếp thực hiện và xử lý công việc ( đọc/ghi dữ liệu).
- ***Secondary Namenode:*** Secondary Namenode là một node phụ chạy cùng với Namenode, nhìn tên gọi nhiều người nhầm tưởng rằng nó để backup cho Namenode tuy nhiên không phải vậy, Secondary Namenode như là một trợ lý đắc lực của Namenode, có vai trò và nhiệm vụ rõ ràng:
  - Nó thường xuyên đọc các file, các metadata được lưu trên RAM của datanode và ghi vào ổ cứng.
  - Nó liên tục đọc nội dung trong Editlogs và cập nhật vào FsImage, để chuẩn bị cho lần khởi động tiếp theo của namenode.
  - Nó liên tục kiểm tra tính chính xác của các tệp tin lưu trên các datanode.
- ***Cơ chế heartbeat:*** Heartbeat là cách liên lạc hay là cách để datanode cho namenode biết là nó còn sống. Định kì datanode sẽ gửi một heartbeat về cho namenode để namenode biết là datanode đó còn hoạt động. Nếu datanode không gửi heartbeat về cho namenode thì namenode coi rằng node đó đã hỏng và không thể thực hiện nhiệm vụ được giao. Namenode sẽ phân công task đó cho một datanode khác.



- **Rack:** Theo thứ tự giảm dần từ cao xuống thấp thì ta có Rack > Node > Block. Rack là một cụm datanode cùng một đầu mạng, bao gồm các máy vật lý (tương đương một server hay 1 node ) cùng kết nối chung 1 switch.
- **Blocks:** Blocks là một đơn vị lưu trữ của HDFS, các data được đưa vào HDFS sẽ được chia thành các block có các kích thước cố định (nếu không cấu hình thì mặc định nó là 128MB).

### Nguyên lí chung của đọc ghi dữ liệu trên HDFS:

- **Quy trình ghi dữ liệu vào HDFS:**



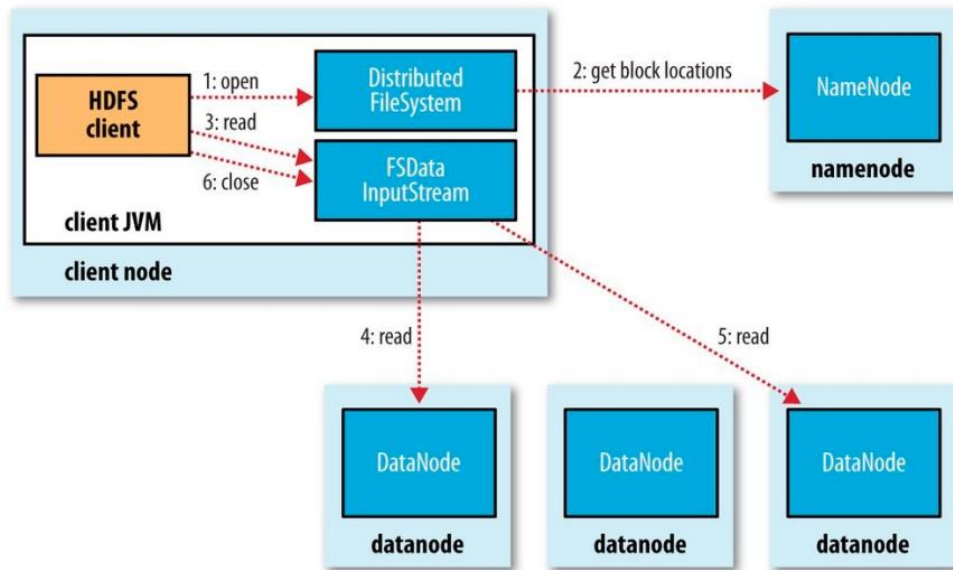
Hình 2: Quá trình ghi file vào HDFS.

1. Client gửi yêu cầu tạo file qua API DistributedFileSystem.
2. NameNode kiểm tra quyền truy cập và sự tồn tại của file.
3. Nếu hợp lệ, FSDataOutputStream được trả về cho client để ghi dữ liệu.
4. DFSOutputStream chia dữ liệu thành các packet, đẩy vào hàng đợi DataQueue.
5. DataStreamer yêu cầu NameNode phân phối block đến các DataNode.
6. Các DataNode tạo thành pipeline theo số bản sao. Dữ liệu được gửi tuần tự qua các node này.
7. AckQueue theo dõi các packet chưa được xác nhận. Mỗi packet chỉ được loại bỏ khi nhận đầy đủ xác nhận từ các DataNode.

8. Khi client.close() được gọi, các packet còn lại được gửi đi, và quá trình ghi kết thúc.

**Lệnh ghi file lên HDFS: `hdfs dfs -put <local_path> <hdfs_path>`.**

- **Quy trình đọc dữ liệu từ HDFS:**



*Hình 3: Quá trình đọc file từ HDFS.*

1. Client gọi open() để mở file.
2. DistributedFileSystem gửi yêu cầu tới NameNode để lấy vị trí các DataNode lưu các block của file.
3. NameNode trả về danh sách các DataNode chứa bản sao từng block.
4. FSDataInputStream được trả về cho client, bên trong chứa DFSInputStream để quản lý quá trình I/O.
5. Client gọi read(), DFSInputStream kết nối với DataNode gần nhất để đọc block đầu tiên.
6. Khi đọc xong một block, DFSInputStream tự động chuyển sang DataNode kế tiếp chứa block tiếp theo.
7. Nếu có lỗi, DFSInputStream chuyển sang DataNode khác có bản sao của block đó.
8. Khi đọc hoàn tất, client gọi close() để đóng file.

### 2.3. Xử lý dữ liệu lớn với MapReduce.

MapReduce là một mô hình lập trình song song và phân tán được sử dụng để xử lý dữ liệu lớn trên Hadoop. Mô hình này chia công việc thành hai giai đoạn chính là Map và Reduce.

#### Giai đoạn Map:

1. Dữ liệu đầu vào được chia thành các phần nhỏ và phân phối đến các Mapper.
2. Mỗi Mapper đọc một phần dữ liệu, xử lý và tạo ra các cặp key-value trung gian.
3. Các cặp key-value trung gian được sắp xếp theo key.

#### Giai đoạn Reduce:

1. Các cặp key-value trung gian được nhóm theo key và gửi đến các Reducer.
2. Mỗi Reducer nhận một tập hợp các giá trị có cùng key, xử lý và tạo ra kết quả cuối cùng.

### 2.4. Nguyên tắc hoạt động của MapReduce

MapReduce hoạt động theo nguyên tắc chia để trị (divide and conquer), trong đó một bài toán lớn được chia thành các bài toán con nhỏ hơn và giải quyết độc lập.

- **Chia nhỏ dữ liệu:** Dữ liệu đầu vào được chia thành các phần nhỏ và phân phối đến các Mapper.
- **Xử lý song song:** Các Mapper xử lý các phần dữ liệu một cách độc lập và đồng thời.
- **Tổng hợp kết quả:** Các kết quả trung gian từ các Mapper được tổng hợp và gửi đến các Reducer.
- **Kết hợp kết quả:** Các Reducer kết hợp các kết quả trung gian để tạo ra kết quả cuối cùng.

## 2.5. Cài đặt Hadoop

### 2.5.1. Cài đặt JDK














Hadoop yêu cầu Java Development Kit (JDK) để hoạt động. Các bước cài đặt JDK bao gồm:

**a. Tải xuống JDK từ trang chủ của Oracle:** Hadoop sử dụng JDK 1.8

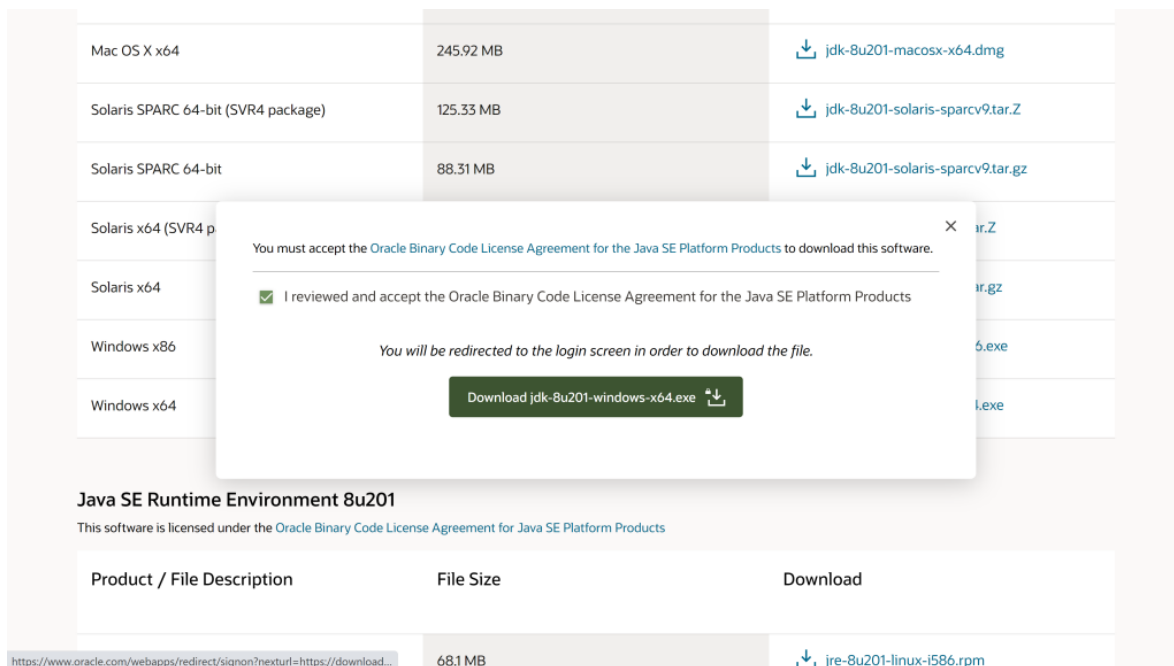
Ta vào liên kết sau:

<https://www.oracle.com/java/technologists/javase/javase8-archive-downloads.html>

Ví dụ cài đặt Java SE Development Kit 8u201

Java SE Development Kit 8u201		
This software is licensed under the <a href="#">Oracle Binary Code License Agreement for Java SE Platform Products</a>		
Product / File Description	File Size	Download
Linux ARM v6/v7 Soft Float ABI	72.98 MB	 <a href="#">jdk-8u201-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM v6/v7 Soft Float ABI	69.92 MB	 <a href="#">jdk-8u201-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	170.98 MB	 <a href="#">jdk-8u201-linux-i586.rpm</a>
Linux x86	185.77 MB	 <a href="#">jdk-8u201-linux-i586.tar.gz</a>
Linux x64	168.05 MB	 <a href="#">jdk-8u201-linux-x64.rpm</a>
Linux x64	182.93 MB	 <a href="#">jdk-8u201-linux-x64.tar.gz</a>
Mac OS X x64	245.92 MB	 <a href="#">jdk-8u201-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	125.33 MB	 <a href="#">jdk-8u201-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	88.31 MB	 <a href="#">jdk-8u201-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	133.99 MB	 <a href="#">jdk-8u201-solaris-x64.tar.Z</a>
Solaris x64	92.16 MB	 <a href="#">jdk-8u201-solaris-x64.tar.gz</a>
Windows x86	197.66 MB	 <a href="#">jdk-8u201-windows-i586.exe</a>
Windows x64	207.46 MB	 <a href="#">jdk-8u201-windows-x64.exe</a>

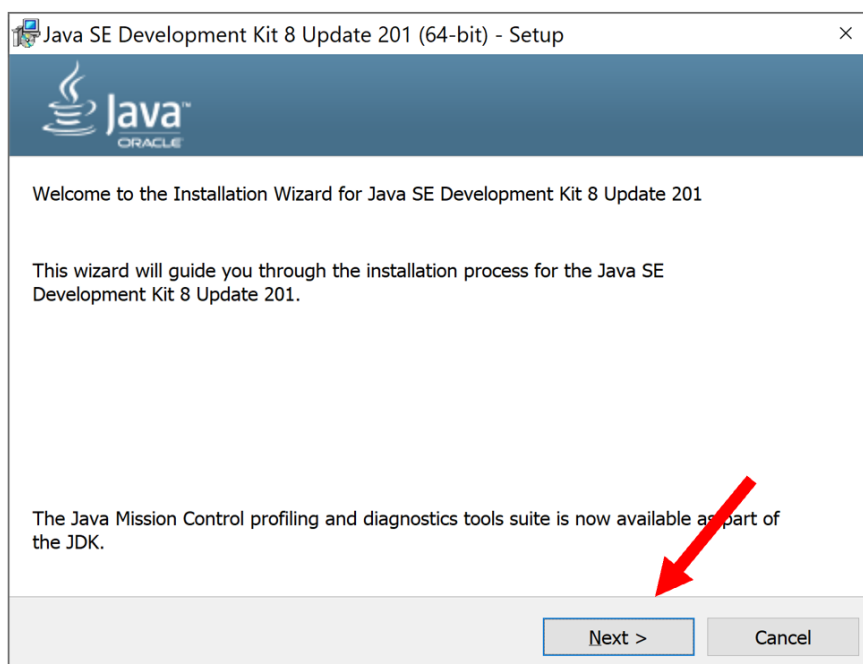
Hình 4: Hướng dẫn tải JDK 8u201\_(1)



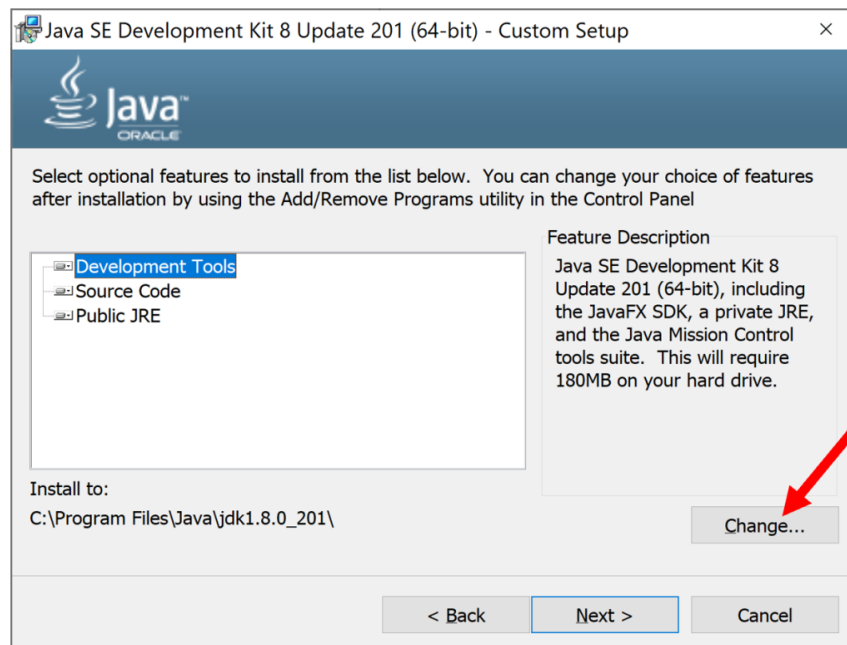
Hình 5: Hướng dẫn tải JDK 8u201\_(2)

## b. Cài đặt JDK

- Tiến hành cài đặt:

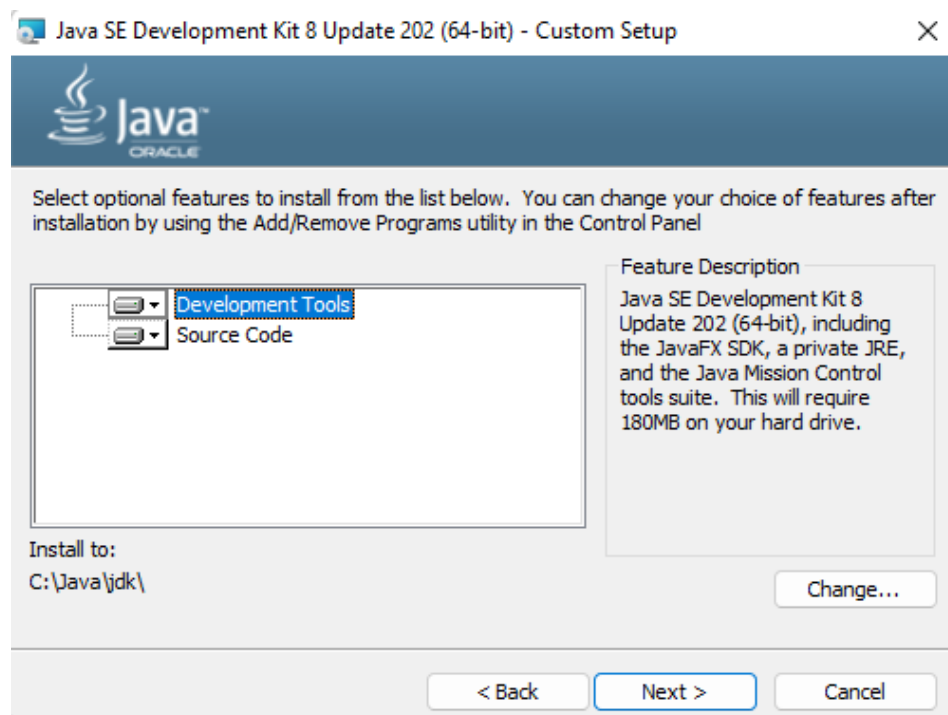


Hình 6: Tiến hành cài đặt JDK-(1).



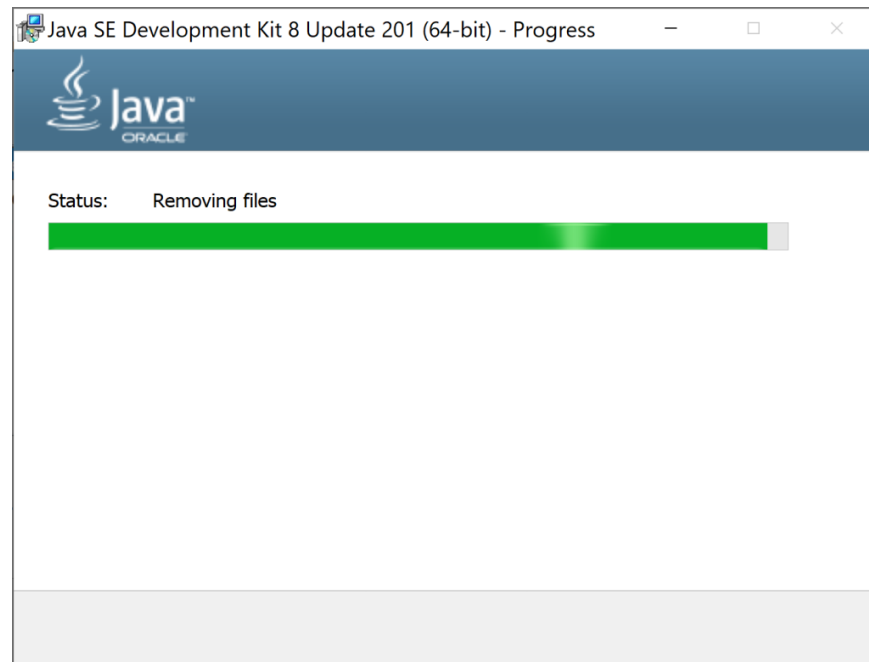
Hình 7: Tiến hành cài đặt JDK-(2).

- Tới chỗ này nhớ chỉnh vào Ổ C, không có dấu và không khoảng trắng

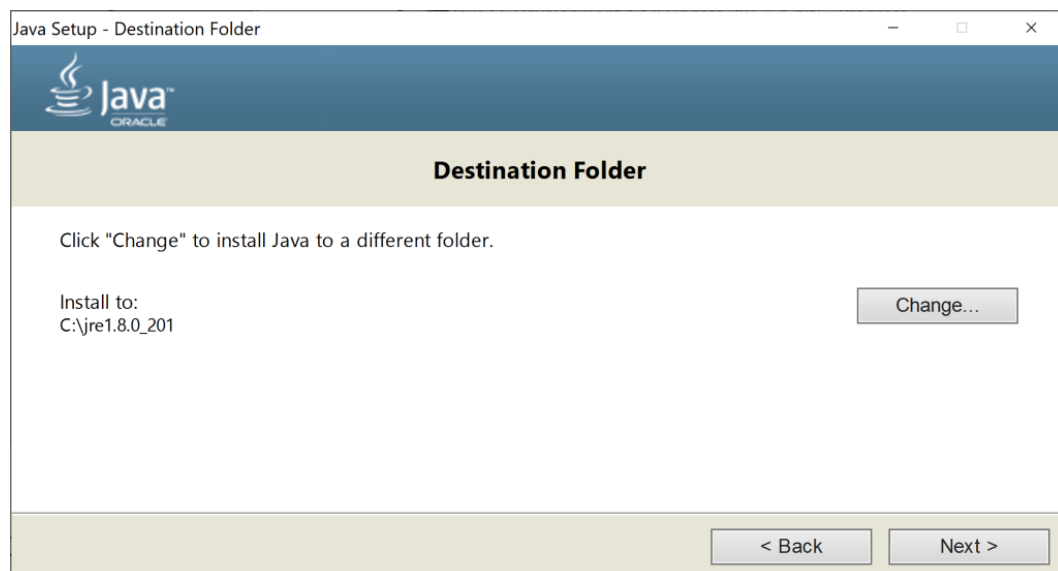


Hình 8: Tiến hành cài đặt JDK-(3).

- Chọn được nơi cài JDK không có khoảng trắng, nhấn Next để tiếp tục



*Hình 9: Tiến hành cài đặt JDK-(4).*



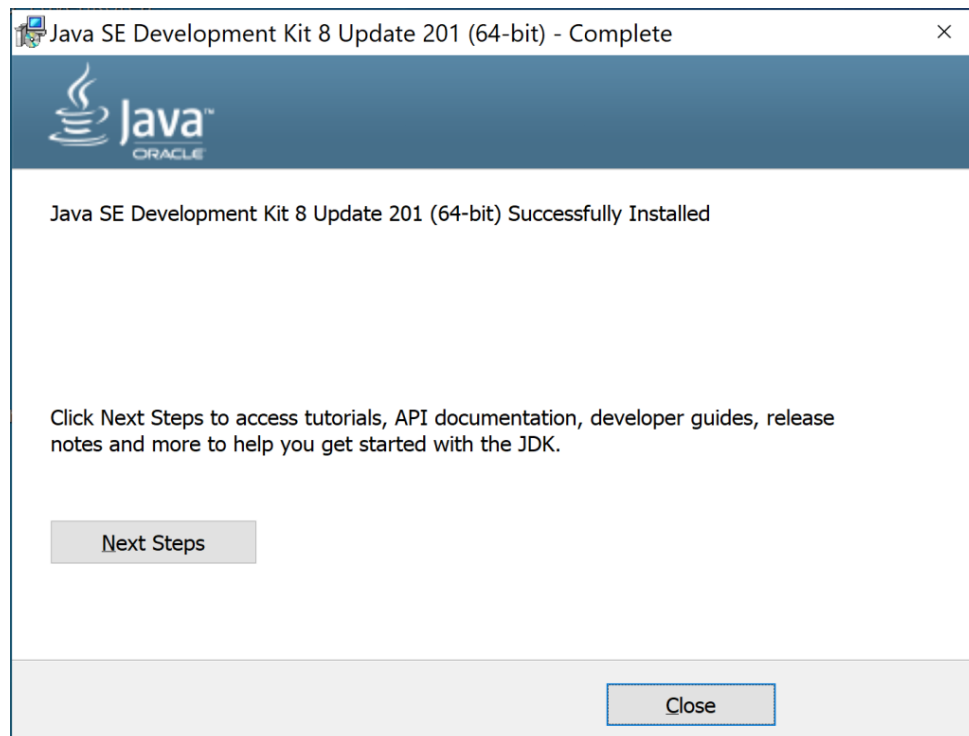
*Hình 10: Tiến hành cài đặt JDK-(5).*

- Nếu jre yêu cầu cài thì cũng chính vào ô C như trên và bấm Next rồi tiếp tục chờ



Hình 11: Tiến hành cài đặt JDK-(6).

- Khi xuất hiện màn hình dưới đây tức là đã hoàn tất quá trình cài đặt JDK



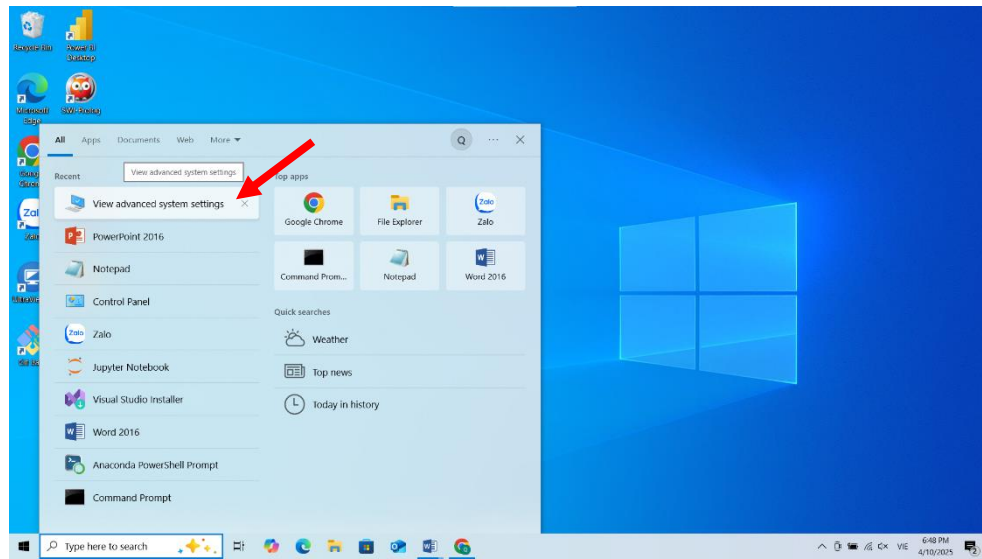
Hình 12: Hoàn thành đăng kí JDK.

- Bấm Close để hoàn tất và như vậy đã cài đặt xong JDK 1.8



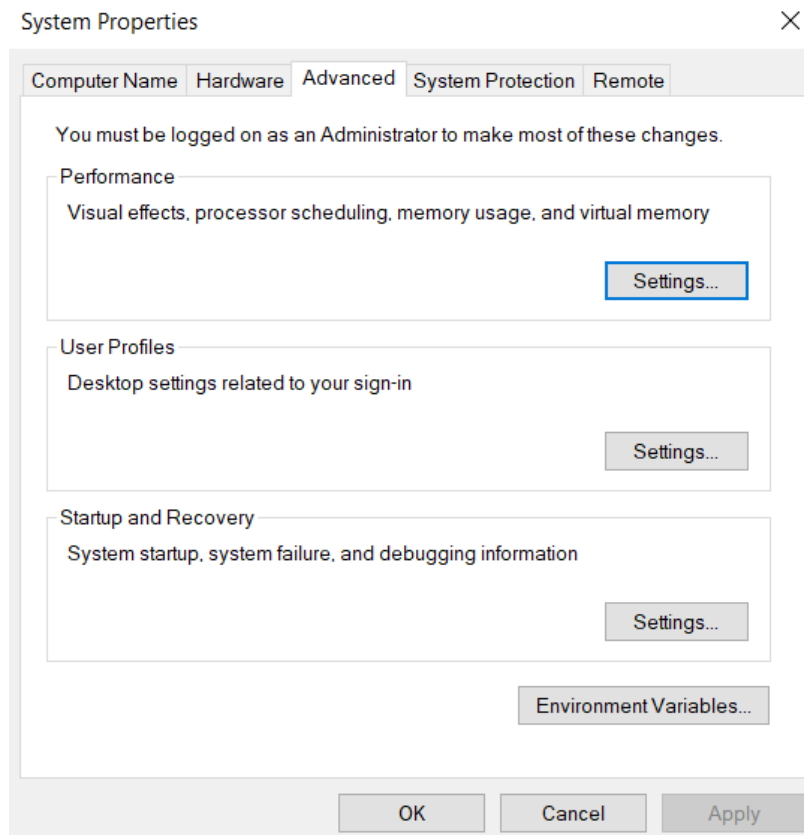
### c. Thiết lập môi trường biến cho Java JDK

- Vào Start menu, search “Edit the system environment”.



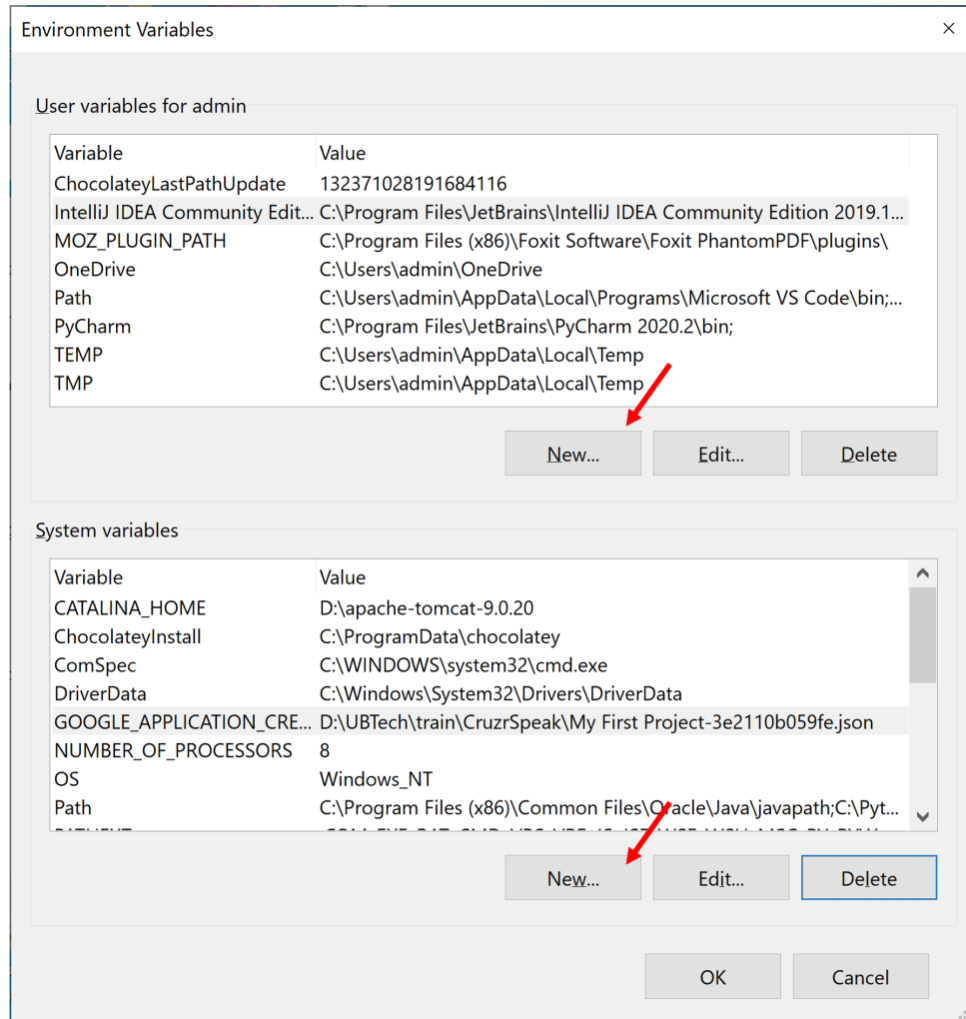
Hình 13: Start menu, search “Edit the system environment”.

- Hiện thị cửa sổ như bên dưới, chọn “Environment Variables...”



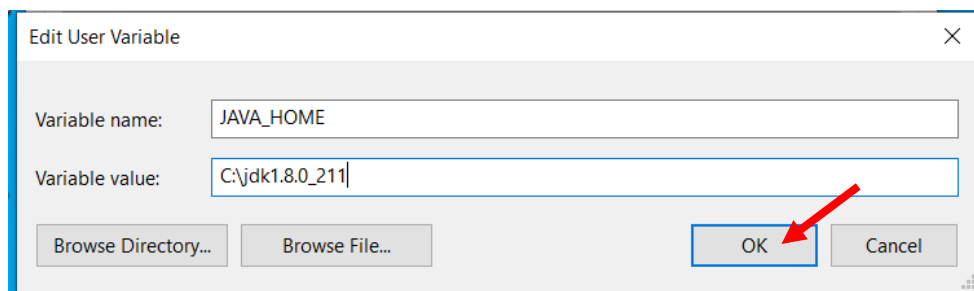
Hình 14: Hiện thị cửa sổ System Properties.

- Ở mục “User variables for..” chọn vào mục “New”

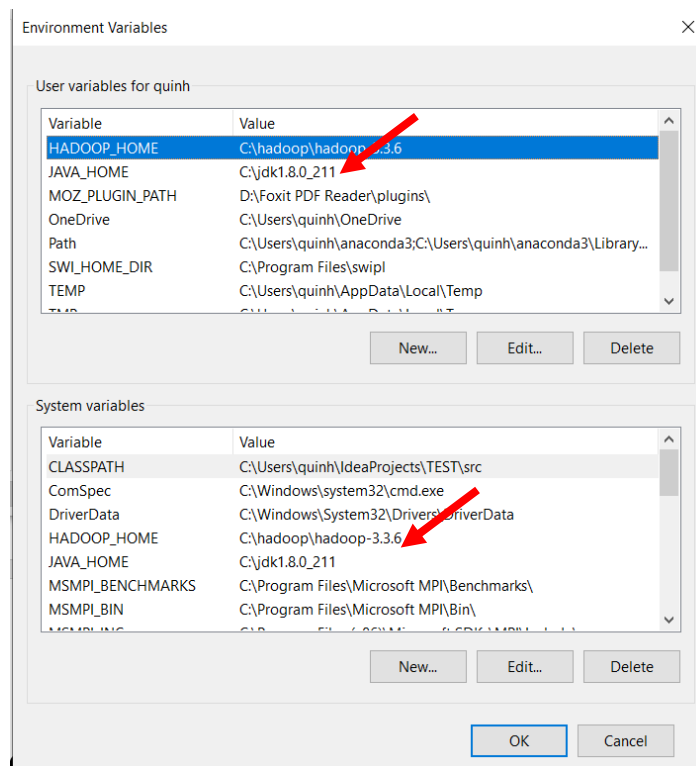


Hình 15: Thiết lập môi trường Java JDK – (1).

- Hiện cửa sổ, ta đặt tên như bên dưới hình, Nhấn OK.

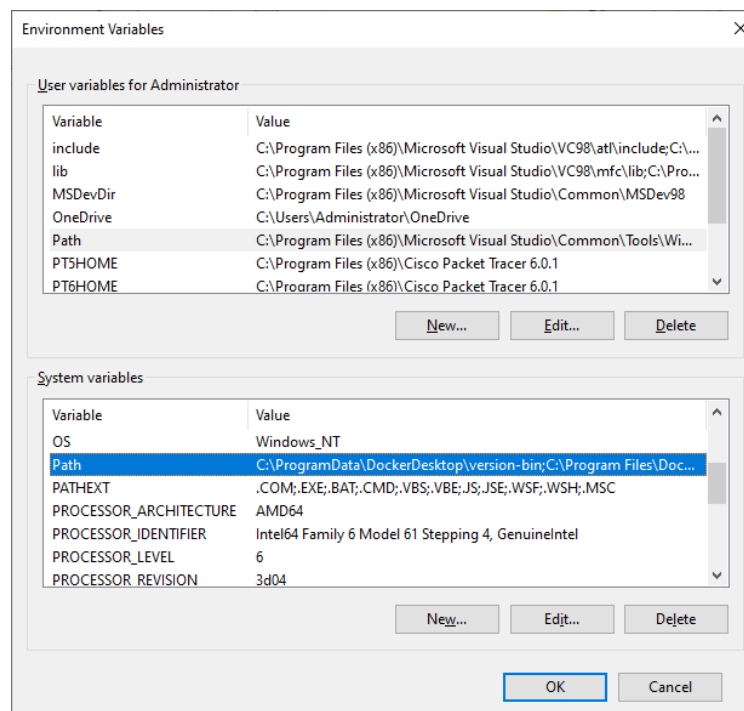


Hình 16: Thiết lập môi trường Java JDK – (2).



Hình 17: Thiết lập môi trường Java JDK – (3).

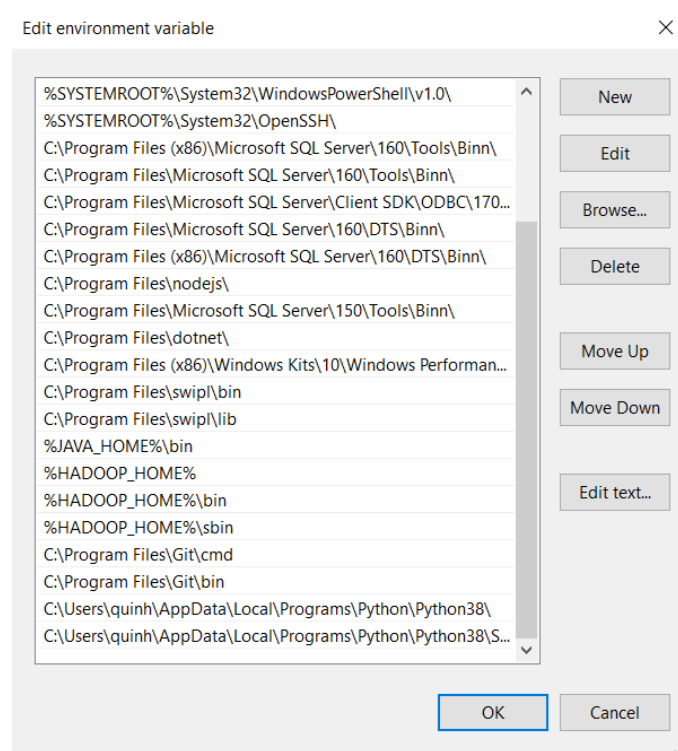
- Thêm các biến môi trường vào PATH. Trong cửa sổ *System Variables*, chọn mục *Path*  
=> Chọn nút *Edit*



Hình 18: Thiết lập môi trường Java JDK – (4).

- Chọn nút New, thêm các biến như sau:

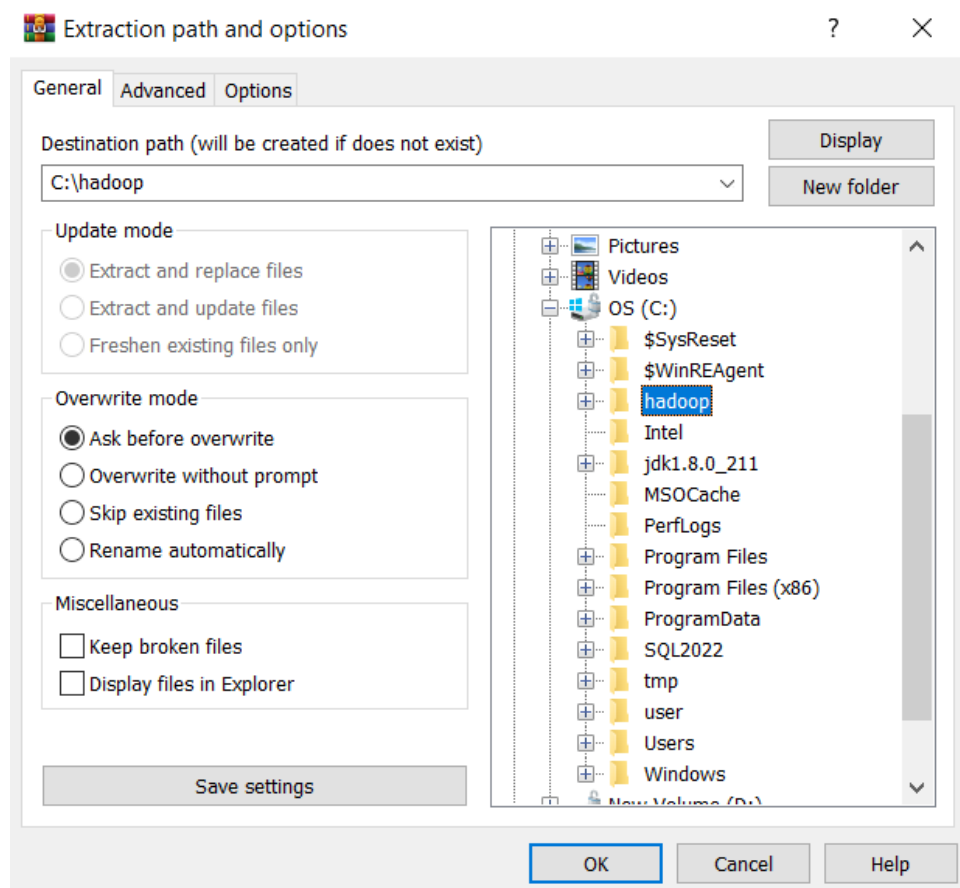
- %JAVA\_HOME%\bin



*Hình 19: Hoàn thành việc thiết lập môi trường JDK.*

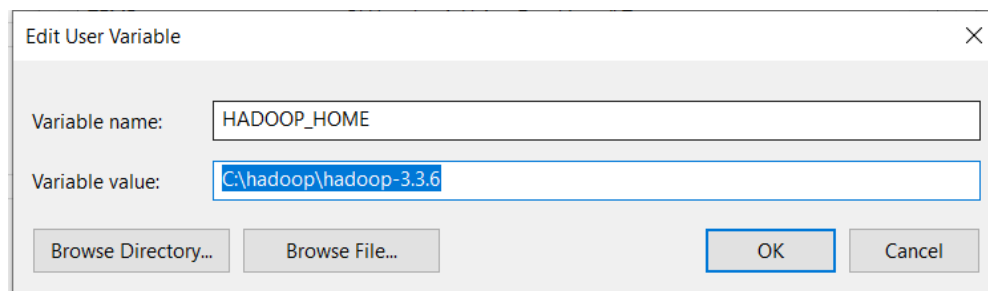
### 2.5.2. Cài đặt Hadoop

- Tải xuống phiên bản Hadoop từ trang chủ của Apache Hadoop ở link sau:  
<https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz>
- Giải nén gói Hadoop vừa tải xuống vào ổ C.



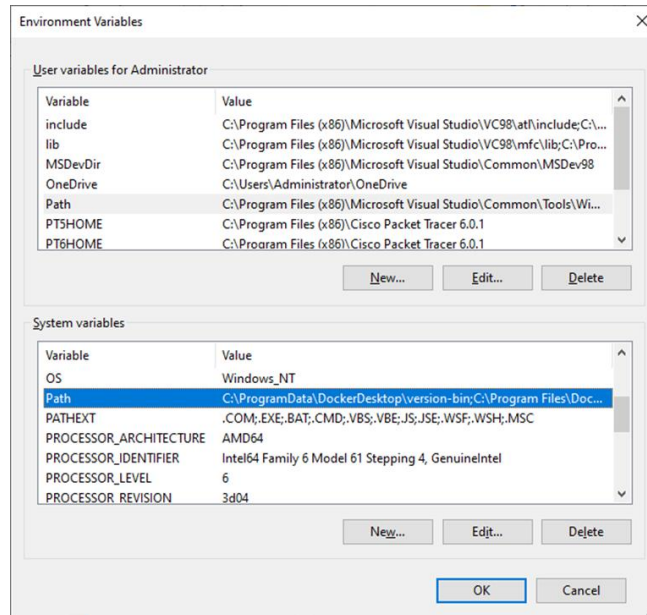
Hình 20: Giải nén gói Hadoop vào ổ C:/hadoop.

- Cấu hình các tập tin cần thiết để Hadoop hoạt động.  
Tương tự với các bước setup JDK, thiết lập các biến và đường dẫn mới cho Hadoop.



Hình 21: Đường dẫn mới cho Hadoop.

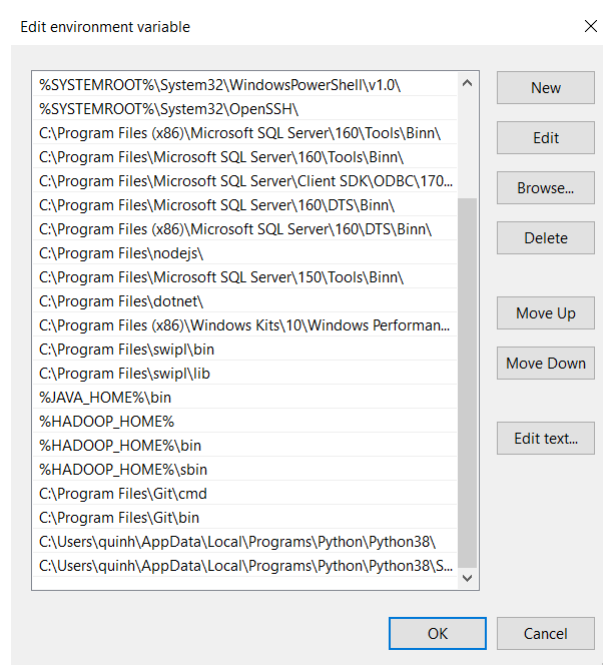
- Ở mục “System variables” chọn mục “Path” và nhấn “Edit”.



Hình 22: Thiết lập các biến mới cho Hadoop.

- Chọn nút New, thêm các biến như sau:

- %HADOOP\_HOME%
- %HADOOP\_HOME%\bin
- %HADOOP\_HOME%\sbin



Hình 23: Thiết lập Hadoop thành công.

### 2.5.3. Cấu hình các tập tin cho Hadoop

- Các tập tin cấu hình chính của Hadoop bao gồm:
  - core-site.xml: Cấu hình các thiết lập chung cho Hadoop.
  - hdfs-site.xml: Cấu hình các thiết lập cho HDFS.
  - mapred-site.xml: Cấu hình các thiết lập cho MapReduce.
  - yarn-site.xml: Cấu hình các thiết lập cho YARN (Yet Another Resource Negotiator).
  - hadoop-env.cmd

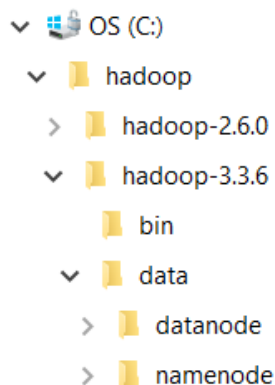
#### a. Cấu hình core-site.xml như dưới đây:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

#### b. Cấu hình mapred-site.xml như dưới đây:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

#### c. Cấu hình hdfs-site.xml như dưới đây:



- Tạo thư mục “data” trong “C:/hadoop”
- Tạo thư mục con “datanode” trong “C:/hadoop/data”
- Tạo thư mục con “namenode” trong “C:/hadoop/data”

**d. Sau đó cấu hình hdfs-site.xml như sau:**

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/hadoop/ hadoop-3.3.6/data/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/hadoop/ hadoop-3.3.6/data/datanode</value>
  </property>
</configuration>
```

**e. Cấu hình yarn-site.xml như dưới đây:**

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```



#### f. Cấu hình `hadoop-env.cmd`:

Mở file này lên và tìm tới lệnh:

- `set JAVA_HOME=%JAVA_HOME%`

sửa `%JAVA_HOME%` thành đường dẫn cài JDK trong ổ C:

- `set JAVA_HOME= C:\Java\jdk`

Sau đó format lại **namenode** và **datanode**: mở command line lên, gõ lệnh sau:

- **`hdfs namenode -format`**
- **`hdfs datanode -format`**
- Bước format này chỉ cần làm 1 lần.

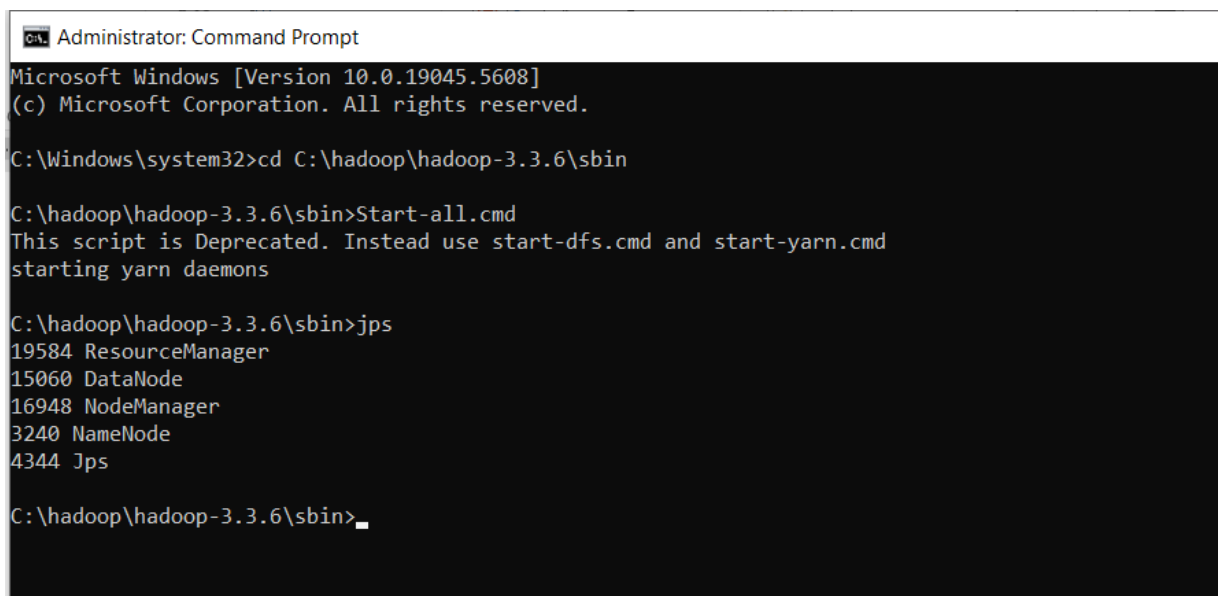


```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\admin>hdfs namenode -format
2021-01-01 18:44:31,000 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = thanhtran/172.20.224.1
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 3.3.0
STARTUP_MSG: classpath = C:\hadoop-3.3.0\etc\hadoop;C:\hadoop-3.3.0\share\hadoop\common;C:\hadoop-3.3.0\share\hadoop\
common\lib\accessors-smart-1.2.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\animal-sniffer-annotations-1.17.jar;C:\h
adoop-3.3.0\share\hadoop\common\lib\asm-5.0.4.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\audience-annotations-0.5.0.
jar;C:\hadoop-3.3.0\share\hadoop\common\lib\avro-1.7.7.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\checker-qual-2.5.2
.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-beanutils-1.9.4.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\comm
ons-cli-1.2.jar;C:\hadoop-3.3.0\share\hadoop\common\lib\commons-codec-1.11.jar;C:\hadoop-3.3.0\share\hadoop\common\li
```

Hình 24: Định dạng namenode và datanode.

#### 2.5.4. Hoàn thành và chạy thử nghiệm.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\hadoop\hadoop-3.3.6\sbin

C:\hadoop\hadoop-3.3.6\sbin>Start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop\hadoop-3.3.6\sbin>jps
19584 ResourceManager
15060 DataNode
16948 NodeManager
3240 NameNode
4344 Jps

C:\hadoop\hadoop-3.3.6\sbin>_
```

Hình 25: Gõ lệnh chạy hadoop.



# CHƯƠNG 3: BÀI TOÁN ĐẾM TỪ TRÊN CẤU TRÚC MAPREDUCE

## 3.1. Bài toán đếm từ truyền thống

### 3.1.1. Thuật toán

Bài toán đếm từ là một trong những ví dụ kinh điển trong lĩnh vực xử lý văn bản. Mục tiêu của bài toán là xác định số lần xuất hiện của mỗi từ trong một tập văn bản đầu vào. Thuật toán truyền thống để giải quyết bài toán này bao gồm các bước sau:

- Đọc dữ liệu đầu vào: Đọc từng dòng văn bản từ tệp hoặc nguồn dữ liệu.
- Tách từ: Mỗi dòng văn bản được tách thành các từ bằng cách sử dụng khoảng trắng hoặc các ký tự đặc biệt làm dấu phân cách.
- Chuẩn hóa từ (chuyển về chữ thường, loại bỏ dấu câu).
- Cập nhật số lượng đếm trong một bảng băm hoặc từ điển (HashMap<String, Integer>).
- In ra kết quả thống kê số lần xuất hiện của từng từ.

### 3.1.2. Ví dụ cụ thể

- **Cho đoạn văn bản sau:**

This is the example text file for word count example also knows as hello world example of the Hadoop ecosystem.

This example is written for the examples article of java code geek

The quick brown fox jumps over the lazy dog.

The above line is one of the most famous lines which contains all the english language alphabets.

- **Bước 1: Chuẩn hóa văn bản** - Loại bỏ dấu câu, chuyển chữ hoa thành chữ thường.

this is the example text file for word count example also knows as hello world example of the hadoop ecosystem

this example is written for the examples article of java code geek

the quick brown fox jumps over the lazy dog

the above line is one of the most famous lines which contains all the english language alphabets

- **Bước 2: Đếm từ** - Ta thu được kết quả thống kê như sau:

Từ	Số lần xuất hiện
the	7
example	4
is	3
of	3
for	2
word	1
count	1
hadoop	1
java	1
language	1
...	...

- **Mỗi từ xuất hiện một hoặc nhiều lần.**

### 3.2. Bài toán đếm từ trên MapReduce

Trong Hadoop, MapReduce là một phép tính phân tích các tác vụ thao tác lớn thành các tác vụ riêng lẻ có thể được thực hiện song song trên một cụm máy chủ. Kết quả của các tác vụ có thể được kết hợp với nhau để tính toán kết quả cuối cùng.

Đối với dữ liệu lớn không thể xử lý hiệu quả bằng thuật toán truyền thống, MapReduce là một giải pháp tối ưu. Đây là mô hình lập trình do Google phát triển, giúp xử lý dữ liệu lớn phân tán trên nhiều máy tính.

Bài toán đếm từ được triển khai hiệu quả trên MapReduce qua hai giai đoạn chính: Map và Reduce.

### 3.2.1. Thuật toán trên Map

- **Hàm Map** – Hàm này lấy một tập dữ liệu và chuyển đổi nó thành một tập dữ liệu khác, trong đó các phần tử riêng lẻ được chia thành các cặp (cặp khóa-giá trị). Cụ thể, hàm này thực hiện các bước sau:
  - **Đầu vào:** Hàm Map nhận vào một tập dữ liệu (thường là một tệp văn bản hoặc một chuỗi), trong đó mỗi phần tử có thể là một dòng hoặc một đoạn văn bản.
  - **Chuyển đổi:** Hàm sẽ phân tách (tokenize) dữ liệu đầu vào thành các từ riêng lẻ. Mỗi từ sau đó sẽ được chuẩn hóa (ví dụ: chuyển thành chữ thường) để đảm bảo việc đếm là chính xác và không phân biệt chữ hoa hay chữ thường.
  - **Phát ra cặp (key-value)**
  - **Khóa (key):** Là từ vừa được phát hiện trong dữ liệu.
  - **Giá trị (value):** Là số lượng đếm ban đầu, thường là 1, biểu thị rằng từ đó đã xuất hiện một lần.
- **Thuật toán chi tiết:**

```
public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {
    String line = value.toString(); // Chuyển đổi nội dung dòng thành chuỗi
    StringTokenizer tokenizer = new StringTokenizer(line); // Tạo StringTokenizer để tách
dòng
    while (tokenizer.hasMoreTokens()) { // Lặp qua từng từ
        word.set(tokenizer.nextToken()); // Lấy từ tiếp theo
        output.collect(word, one); // Phát ra cặp (từ, 1)
    }
}
```

- **Ví dụ – (Hàm Map trong Word Count)**

<b>Đầu vào</b>	Bộ dữ liệu gốc (có chữ hoa và dấu câu)	This is the example text file for word count example also knows as hello world example of the Hadoop ecosystem.
----------------	--	---

		<p>This example is written for the examples article of java code geek The quick brown fox jumps over the lazy dog.</p> <p>The above line is one of the most famous lines which contains all the english language alphabets.</p>
<b>Tiền xử lý</b>	Chuyển toàn bộ sang chữ thường, loại bỏ dấu câu	<p>this is the example text file for word count example also knows as hello world example of the hadoop ecosystem this example is written for the examples article of java code geek the quick brown fox jumps over the lazy dog the above line is one of the most famous lines which contains all the english language alphabets</p>
<b>Đầu ra</b>	Chuyển thành các cặp (từ, 1)	<p>(this, 1), (is, 1), (the, 1), (example, 1), (text, 1), (file, 1), (for, 1), (word, 1), (count, 1), (example, 1), (also, 1), (knows, 1), (as, 1), (hello, 1), (world, 1), (example, 1), (of, 1), (the, 1), (hadoop, 1), (ecosystem, 1), (this, 1), (example, 1), (is, 1), (written, 1), (for, 1), (the, 1), (examples, 1), (article, 1), (of, 1), (java, 1), (code, 1), (geek, 1), (the, 1), (quick, 1), (brown, 1), (fox, 1), (jumps, 1), (over, 1), (the, 1), (lazy, 1), (dog, 1), (the, 1), (above, 1), (line, 1), (is, 1), (one, 1), (of, 1), (the, 1), (most, 1), (famous, 1), (lines, 1), (which, 1), (contains, 1), (all, 1), (the, 1), (english, 1), (language, 1), (alphabets, 1)</p>

### 3.2.2. Thuật toán trên Reduce

- **Hàm Reduce** là một phần quan trọng trong mô hình MapReduce, có nhiệm vụ tổng hợp và kết hợp đầu ra từ hàm Map thành các tập hợp dữ liệu nhỏ hơn. Cụ thể, hàm này thực hiện các bước sau:
  - **Nhận đầu vào:** Hàm Reduce nhận vào một cặp (key, list of values) từ đầu ra của hàm Map.

- **Key:** Là từ đã được phát ra từ hàm Map.
  - **List of values:** Là danh sách các giá trị (thường là 1) tương ứng với từ đó.
  - **Tính toán tổng:** Hàm sẽ tính tổng số lần xuất hiện của từ bằng cách cộng tất cả các giá trị trong danh sách.
  - **Phát ra kết quả:** Kết quả cuối cùng sẽ là một cặp (từ, tổng số lần xuất hiện), được phát ra để lưu trữ hoặc xử lý tiếp.
- **Thuật toán chi tiết**  

```
public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
    int sum = 0; // Khởi tạo biến để tính tổng
    while (values.hasNext()) {
        sum += values.next().get(); // Cộng tổng số lần xuất hiện
    }
    output.collect(key, new IntWritable(sum)); // Phát ra cặp (từ, tổng số lần xuất hiện)
}
```
  - **Ví dụ – (Chức năng Reduce trong Word Count)**

<b>Đầu vào</b>	Bộ Tuple	(this, [1, 1]) (is, [1, 1, 1]) (the, [1, 1, 1, 1, 1, 1, 1, 1]) (example, [1, 1, 1, 1]) (text, [1]) (file, [1]) (for, [1, 1]) (word, [1]) (count, [1]) (also, [1]) (knows, [1]) (as, [1]) (hello, [1]) (world, [1]) (of, [1, 1, 1]) (hadoop, [1]) (ecosystem, [1]) (written, [1]) (examples, [1]) (article, [1]) (java, [1]) (code, [1]) (geek, [1]) (quick, [1]) (brown, [1]) (fox, [1]) (jumps, [1]) (over, [1]) (lazy, [1]) (dog, [1]) (above, [1]) (line, [1]) (one, [1]) (most, [1]) (famous, [1]) (lines, [1]) (which, [1]) (contains, [1]) (all, [1]) (english, [1]) (language, [1]) (alphabets, [1])
<b>Đầu ra Reduce</b>	Chuyển đổi thành tập hợp	(this, 2), (is, 3), (the, 7), (example, 4), (text, 1), (file, 1), (for, 2), (word, 1), (count, 1), (also, 1), (knows,

	các tuple nhỏ hơn	1), (as, 1), (hello, 1), (world, 1), (of, 3), (hadoop, 1), (ecosystem, 1), (written, 1), (examples, 1), (article, 1), (java, 1), (code, 1), (geek, 1), (quick, 1), (brown, 1), (fox, 1), (jumps, 1), (over, 1), (lazy, 1), (dog, 1), (above, 1), (line, 1), (one, 1), (most, 1), (famous, 1), (lines, 1), (which, 1), (contains, 1), (all, 1), (english, 1), (language, 1), (alphabets, 1)
--	-------------------	--

### 3.2.3. Ví dụ đồ thị cụ thể cho thuật toán Map và Reduce ở ví dụ được đề cập ở phần 3.1.2

#### a. Dữ liệu đầu vào: Chúng ta sẽ sử dụng văn bản sau làm dữ liệu đầu vào cho bài toán đếm từ.

This is the example text file for word count example also knows as hello world example of the Hadoop ecosystem.

This example is written for the examples article of java code geek.

The quick brown fox jumps over the lazy dog.

The above line is one of the most famous lines which contains all the english language alphabets.

#### b. Bước 1: Map

Trong bước này, mỗi Mapper sẽ đọc một đoạn văn bản (mỗi dòng), phân tách dòng thành các từ và phát ra các cặp (từ, 1). Dưới đây là quy trình chi tiết:

##### - Dòng 1:

This is the example text file for word count example also knows as hello world example of the Hadoop ecosystem.

##### ▪ Đầu ra từ Mapper cho Dòng 1:

(this, 1)

(is, 1)

(the, 1)

(example, 1)

(text, 1)

(file, 1)



(for, 1)  
(word, 1)  
(count, 1)  
(example, 1)  
(also, 1)  
(knows, 1)  
(as, 1)  
(hello, 1)  
(world, 1)  
(example, 1)  
(of, 1)  
(the, 1)  
(hadoop, 1)  
(ecosystem, 1)

- ***Dòng 2:***

This example is written for the examples article of java code geek.

▪ *Đầu ra từ Mapper cho Dòng 2:*

(this, 1)  
(example, 1)  
(is, 1)  
(written, 1)  
(for, 1)  
(the, 1)  
(examples, 1)  
(article, 1)  
(of, 1)  
(java, 1)  
(code, 1)  
(geek, 1)

- ***Dòng 3:***

The quick brown fox jumps over the lazy dog.

▪ *Đầu ra từ Mapper cho Dòng 3:*

(the, 1)

(quick, 1)

(brown, 1)

(fox, 1)

(jumps, 1)

(over, 1)

(the, 1)

(lazy, 1)

(dog, 1)

- ***Dòng 4:***

The above line is one of the most famous lines which contains all the english language alphabets.

▪ *Đầu ra từ Mapper cho Dòng 4:*

(the, 1)

(above, 1)

(line, 1)

(is, 1)

(one, 1)

(of, 1)

(the, 1)

(most, 1)

(famous, 1)

(lines, 1)

(which, 1)

(contains, 1)

(all, 1)

(the, 1)

(english, 1)

(language, 1)

(alphabets, 1)

### c. Bước 2: Shuffle & Sort

Trong bước này, các cặp (từ, 1) từ các Mapper sẽ được gom nhóm theo key (từ). Điều này sẽ giúp cho mỗi Reducer nhận được tất cả các giá trị 1 tương ứng với mỗi từ.

*Ví dụ nhóm cho một số từ:*

- (example, [1, 1, 1, 1]) // từ "example" xuất hiện 4 lần
- (the, [1, 1, 1, 1, 1, 1]) // từ "the" xuất hiện 6 lần
- (is, [1, 1, 1]) // từ "is" xuất hiện 3 lần
- .....

### d. Bước 3: Reduce

Trong bước này, mỗi Reducer sẽ nhận một từ duy nhất và danh sách các giá trị 1 tương ứng. Mục tiêu là cộng tổng số lần xuất hiện của mỗi từ.

- **Đầu vào** của Reducer từ "example": (example, [1, 1, 1, 1]) || **Đầu ra** sẽ là: (example, 4)
- **Đầu vào** của Reducer từ "the": (the, [1, 1, 1, 1, 1, 1]) || **Đầu ra** sẽ là: (the, 6)
- **Đầu vào** của Reducer cho từ "is": (is, [1, 1, 1]) || **Đầu ra** sẽ là: (is, 3)
- Tương tự cho các từ khác ...

### e. Kết quả cuối cùng

Sau khi tất cả các Reducer hoàn tất công việc của mình, chúng ta sẽ có kết quả cuối cùng cho bài toán đếm từ:

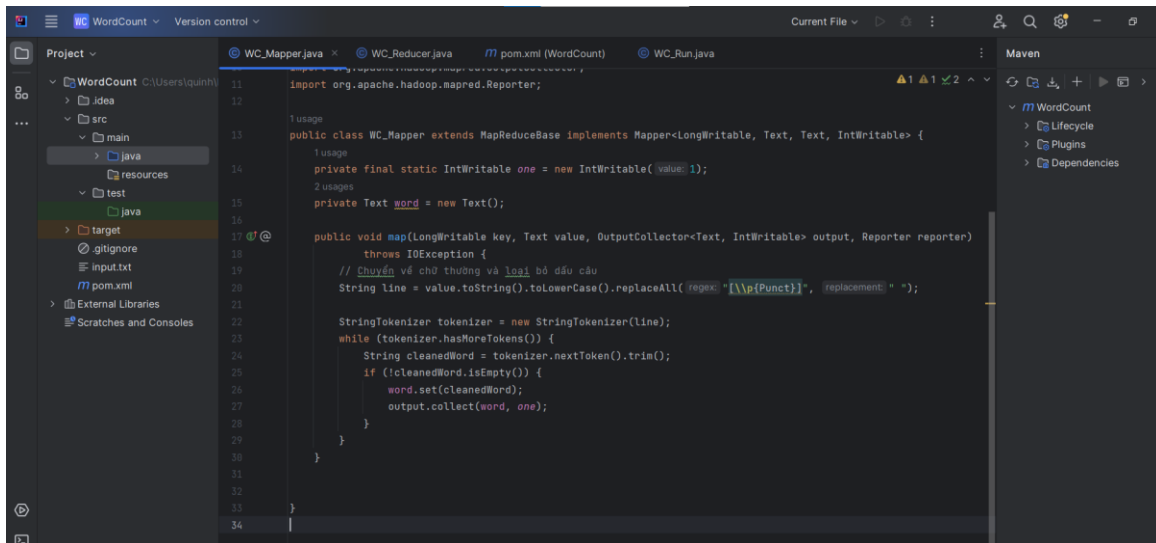
this	2
is	3
the	6
example	4
text	1
file	1
for	2
word	1

count 1  
also 1  
knows 1  
as 1  
hello 1  
world 1  
of 3  
hadoop 1  
ecosystem 1  
quick 1  
brown 1  
fox 1  
jumps 1  
over 1  
lazy 1  
dog 1  
above 1  
line 1  
one 1  
most 1  
famous 1  
lines 1  
which 1  
contains 1  
all 1  
english 1  
language 1  
alphabets 1

# CHƯƠNG 4: XÂY DỰNG DỮ LIỆU VÀ ĐỀ MODE

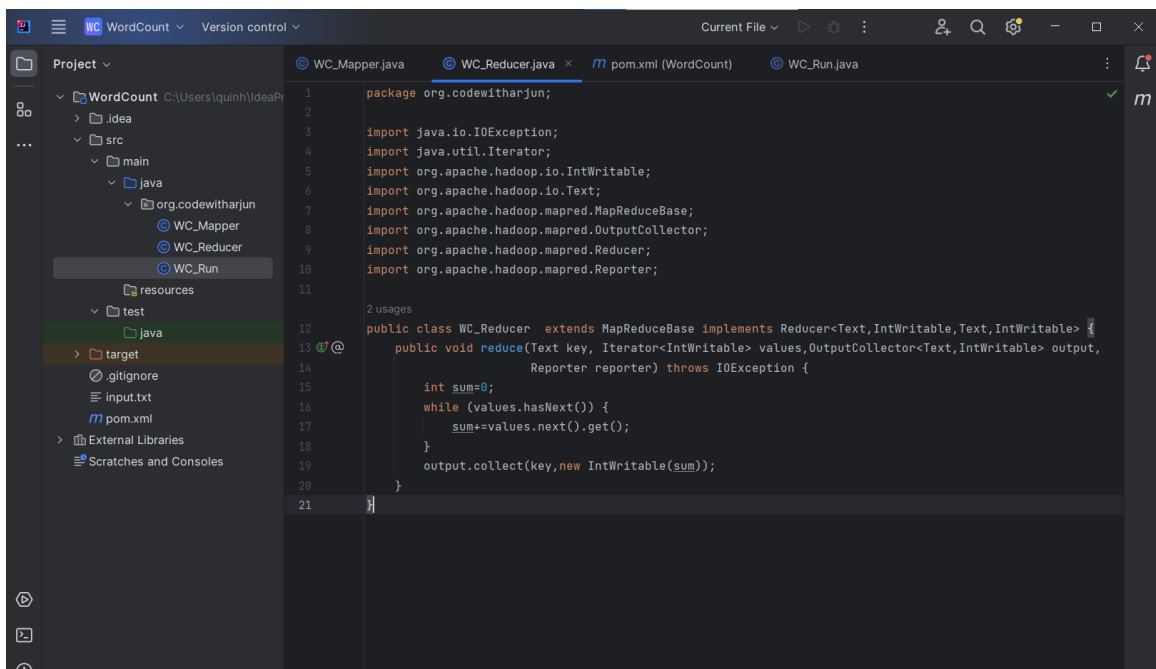
## 4.1. Xây dựng code chạy bài toán đếm từ (Word Count)

- **Tạo file java WC\_Mapper:** Phân tích từng dòng văn bản đầu vào, tách từ và phát ra (emit) từng từ với giá trị 1.



```
11 import org.apache.hadoop.mapred.Reporter;
12
13 public class WC_Mapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
14     1 usage
15     private final static IntWritable one = new IntWritable(1);
16     2 usages
17     private Text word = new Text();
18
19     public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)
20         throws IOException {
21         // chuyển về chữ thường và loại bỏ dấu câu
22         String line = value.toString().toLowerCase().replaceAll(regex: "[\\p{Punct}]", replacement: " ");
23
24         StringTokenizer tokenizer = new StringTokenizer(line);
25         while (tokenizer.hasMoreTokens()) {
26             String cleanedWord = tokenizer.nextToken().trim();
27             if (!cleanedWord.isEmpty()) {
28                 word.set(cleanedWord);
29                 output.collect(word, one);
30             }
31         }
32     }
33 }
34
```

- **Tạo file java WC\_Reduce:** Cộng tổng số lần xuất hiện của mỗi từ (các giá trị 1 từ Mapper).



```
1 package org.codewithharryjun;
2
3 import java.io.IOException;
4 import java.util.Iterator;
5 import org.apache.hadoop.io.IntWritable;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapred.MapReduceBase;
8 import org.apache.hadoop.mapred.OutputCollector;
9 import org.apache.hadoop.mapred.Reducer;
10 import org.apache.hadoop.mapred.Reporter;
11
12 2 usages
13 public class WC_Reduce extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {
14     public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,
15         Reporter reporter) throws IOException {
16
17         int sum=0;
18         while (values.hasNext()) {
19             sum+=values.next().get();
20         }
21         output.collect(key,new IntWritable(sum));
22     }
23 }
```

- **Tạo file java WC\_Run:** Cấu hình và chạy chương trình MapReduce, liên kết các lớp Mapper, Reducer và xác định đường dẫn dữ liệu đầu vào/đầu ra.

```

2
3
4 import java.io.IOException;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.IntWritable;
7 import org.apache.hadoop.io.Text;
8 import org.apache.hadoop.mapred.FileInputFormat;
9 import org.apache.hadoop.mapred.FileOutputFormat;
10 import org.apache.hadoop.mapred.JobClient;
11 import org.apache.hadoop.mapred.JobConf;
12 import org.apache.hadoop.mapred.TextInputFormat;
13 import org.apache.hadoop.mapred.TextOutputFormat;
14
15 public class WC_Run {
16
17     public static void main(String[] args) throws IOException{
18         JobConf conf = new JobConf(WC_Run.class);
19         conf.setJobName("WordCount");
20         conf.setOutputKeyClass(Text.class);
21         conf.setOutputValueClass(IntWritable.class);
22         conf.setMapperClass(WC_Mapper.class);
23         conf.setCombinerClass(WC_Reducer.class);
24         conf.setReducerClass(WC_Reducer.class);
25         conf.setInputFormat(TextInputFormat.class);
26         conf.setOutputFormat(TextOutputFormat.class);
27         FileInputFormat.setInputPaths(conf, new Path(args[0]));
28         FileOutputFormat.setOutputPath(conf, new Path(args[1]));
29         JobClient.runJob(conf);
30     }
31 }

```

- **Tạo file input.txt:** Chuẩn bị và đưa file văn bản (input.txt) vào hệ thống file HDFS để làm dữ liệu đầu vào cho chương trình MapReduce.
  - notepad input.txt: Mở file văn bản input.txt để soạn nội dung dữ liệu đầu vào.
  - type input.txt: Hiển thị nội dung của file input.txt trong terminal để kiểm tra lại nội dung.
  - `hadoop fs -mkdir /input`: Tạo thư mục /input trên HDFS, nơi sẽ chứa file đầu vào cho job MapReduce.
  - `hadoop fs -put input.txt /input`: Đưa file input.txt từ máy tính cục bộ lên thư mục /input trên HDFS, để chương trình MapReduce có thể truy cập.

```

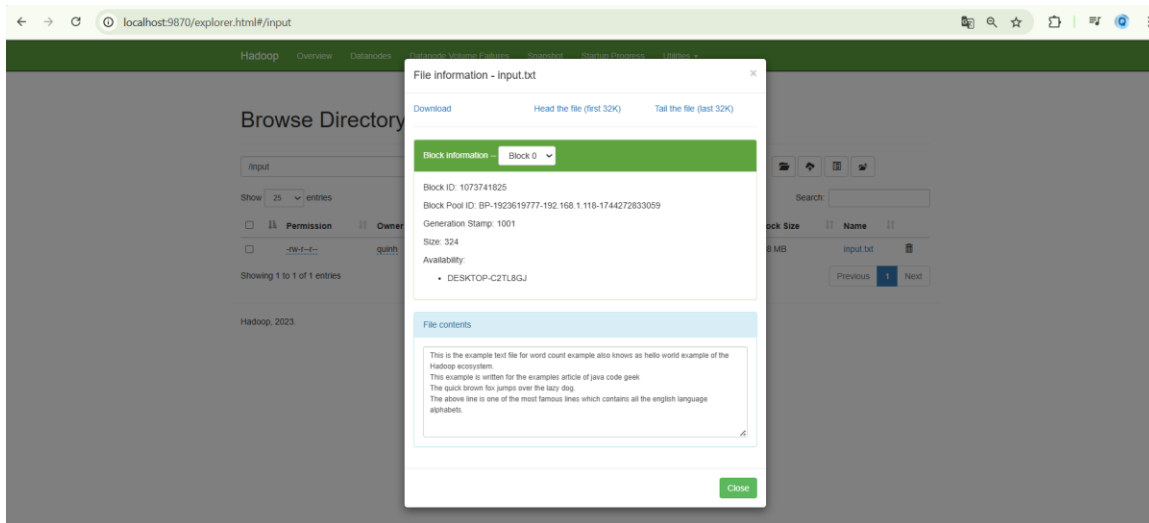
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\quinh\IdeaProjects\WordCount
C:\Users\quinh\IdeaProjects\WordCount>notepad input.txt
C:\Users\quinh\IdeaProjects\WordCount>type input.txt
This is the example text file for word count example also knows as hello world example of the Hadoop ecosystem.
This example is written for the examples article of java code geek
The quick brown fox jumps over the lazy dog.
The above line is one of the most famous lines which contains all the english language alphabets.
C:\Users\quinh\IdeaProjects\WordCount>hadoop fs -mkdir /input
C:\Users\quinh\IdeaProjects\WordCount>hadoop fs -put input.txt /input
C:\Users\quinh\IdeaProjects\WordCount>

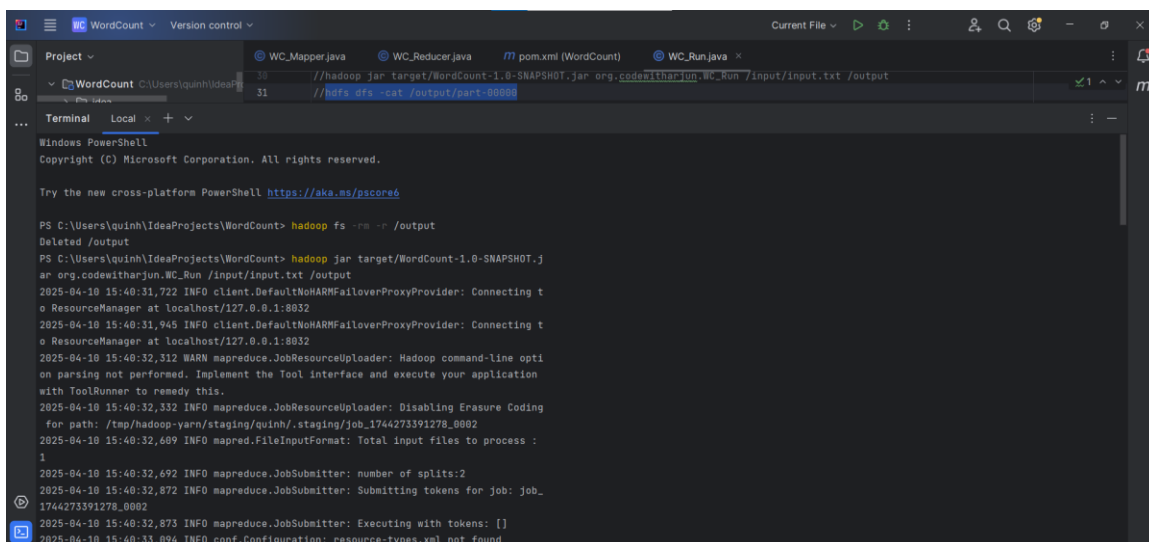
```

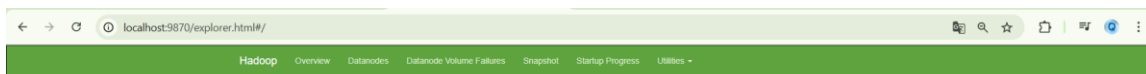
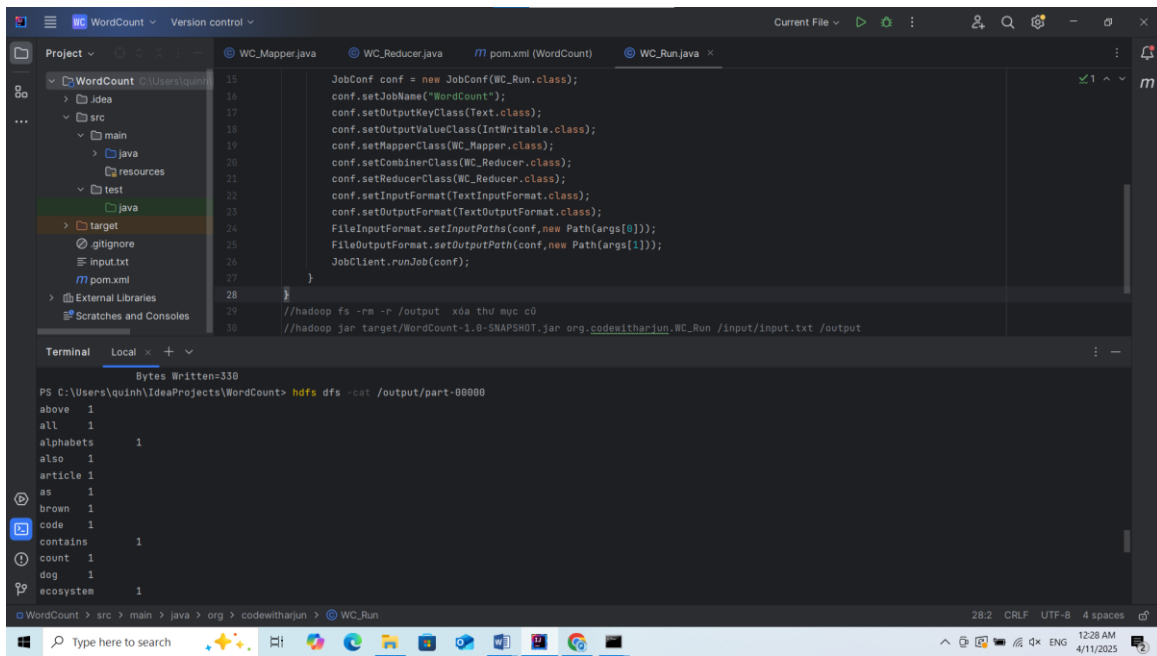
## 4.2. Demo kết quả

- **Đưa file input.txt từ máy tính lên HDFS**
  - `hadoop fs -mkdir /input`: Tạo thư mục /input trên HDFS, nơi sẽ chứa file đầu vào cho job MapReduce.
  - `hadoop fs -put input.txt /input`: Đưa file input.txt từ máy tính cục bộ lên thư mục /input trên HDFS, để chương trình MapReduce có thể truy cập.



- **Chạy chương trình MapReduce để đếm số từ trong file input.txt**
  - Chạy job: `hadoop jar target/WordCount-1.0-SNAPSHOT.jar org.codewitharjun.WC_Run /input/input.txt /output`
  - Chạy xem kết quả: `hdfs dfs -cat /output/part-00000`



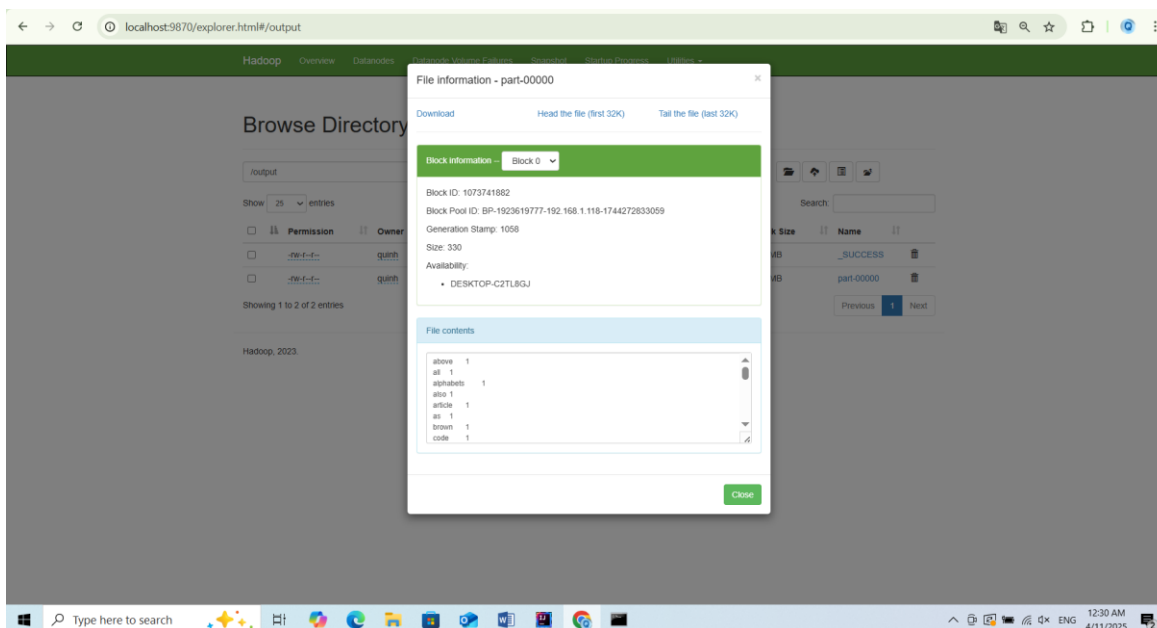


## Browse Directory

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	quinh	supergroup	0 B	Apr 10 15:32	0	0 B	input
drwxr-xr-x	quinh	supergroup	0 B	Apr 10 15:40	0	0 B	output
drwxr-xr-x	quinh	supergroup	0 B	Apr 10 15:38	0	0 B	tmp

Showing 1 to 3 of 3 entries

Hadoop, 2023.





```
Administrator: Command Prompt
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\quinh\IdeaProjects\WordCount

C:\Users\quinh\IdeaProjects\WordCount>hdfs dfs -cat /output/part-00000
above 1
all 1
alphabets 1
also 1
article 1
as 1
brown 1
code 1
contains 1
count 1
dog 1
ecosystem 1
english 1
example 4
examples 1
famous 1
file 1
for 2
fox 1
geek 1
hadoop 1
hello 1
is 3
java 1
jumps 1
knows 1
language 1
lazy 1
line 1
lines 1
most 1
of 3
one 1
over 1
quick 1
text 1
the 8
this 2
which 1
word 1
world 1
written 1

C:\Users\quinh\IdeaProjects\WordCount>_
```

## KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong báo cáo này, em đã trình bày về lý thuyết, cài đặt và ứng dụng của mô hình lập trình MapReduce trên nền tảng Hadoop để giải quyết bài toán đếm số lượng từ trong văn bản. Báo cáo cũng giới thiệu tổng quan về xử lý song song và phân tán, cùng với các khái niệm cơ bản về dữ liệu lớn và hệ sinh thái Hadoop.

Tiếp theo, chúng em đi sâu vào mô hình MapReduce, giải thích chi tiết về cách hoạt động của hai thành phần chính là Mapper và Reducer, cũng như quá trình triển khai thuật toán đếm từ trên Hadoop. Phần cài đặt bao gồm việc hướng dẫn cấu hình Hadoop, biên dịch chương trình Java và thực thi ứng dụng MapReduce trên hệ thống phân tán.

Ngoài ra, báo cáo còn trình bày quy trình thực hiện demo chương trình đếm từ với một tập dữ liệu đầu vào cụ thể, kết quả cho thấy mô hình xử lý hiệu quả và có khả năng mở rộng tốt khi khối lượng dữ liệu tăng lên.

Tuy nhiên, vẫn còn một số hạn chế cần cải thiện trong tương lai như:

- Tối ưu hóa chương trình để giảm thời gian thực thi và sử dụng tài nguyên hiệu quả hơn.
- Mở rộng chương trình để xử lý các bài toán thống kê văn bản phức tạp hơn như đếm tần suất cụm từ, lọc từ dừng, phân tích cảm xúc, v.v.
- Ứng dụng mô hình vào các bài toán thực tế khác trong xử lý ngôn ngữ tự nhiên, khai thác văn bản, hoặc hệ thống tìm kiếm thông tin.

Em hy vọng rằng báo cáo này đã mang đến cái nhìn tổng quan về mô hình lập trình MapReduce và cách ứng dụng nó trong bài toán đếm từ trên nền tảng Hadoop. Đây là một công cụ mạnh mẽ, hữu ích cho xử lý dữ liệu lớn và có thể mở rộng ứng dụng trong nhiều lĩnh vực khác nhau.

# TÀI LIỆU THAM KHẢO

[1] <https://hadoop.apache.org/>

[2] <https://www.oreilly.com/library/view/hadoop-the-definitive/9781449361900/>

[3] <https://www.oreilly.com/library/view/mapreduce-design-patterns/9780596521678/>

[4] [Find Shortest Paths from Source to all Vertices using Dijkstra's Algorithm \(geeksforgeeks.org\)](#)

[5] [The MapReduce based approach to improve the shortest path computation / Lau / J. Math. Comput. Sci. \(scik.org\)](#)