**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙡🕮🙣**

Ảnh có chứa văn bản, mẫu họa

Mô tả được tạo tự động**MÔN HỌC: NHẬP MÔN DỮ LIỆU LỚN**

**BÁO CÁO ĐỒ ÁN**

**ĐỀ TÀI**

**TÌM HIỂU VÀ CÀI ĐẶT HỆ THỐNG QUẢN LÝ DỮ LIỆU PHÂN TÁN**

**GVHD:** ThS. Lê Thị Minh Châu

**Mã lớp:** **BDES333877\_Nhom1**

**Nhóm sinh viên thực hiện: Nhóm 9**

1. Dương Thành Sơn – 22133048
2. Trần Diễm Quỳnh – 22133046
3. Lê Văn Sang – 22133047
4. Nguyễn Đinh Hồng Phúc – 22133041

**TP. Hồ Chí Minh, tháng 12 năm 2024**

**BẢNG PHÂN CÔNG NHIỆM VỤ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ Và Tên** | **Nhiệm Vụ** | **Hoàn Thành** |
| 1 | 22133048 | Dương Thành Sơn  **(Nhóm Trưởng)** | Tìm hiểu các và cài đặt các công cụ trong Hadoop được sử dụng trong bài (Hbase, Zookeeper, Sqoop, Mahout). | 100% |
| 2 | 22133046 | Trần Diễm Quỳnh | Load dữ liệu ban đầu từ bên ngoài vào MySQL và từ MySQL lên HDFS, tổng hợp nội dung báo cáo và đánh giá kết quả kiểm thử các công cụ Hadoop. | 100% |
| 3 | 22133047 | Lê Văn Sang | Thực hiện chính việc kiểm thử các công cụ trong Hadoop, sử dụng công cụ Mahout để phân tích dữ liệu và xuất ra kết quả. | 100% |
| 4 | 22133041 | Nguyễn Đinh Hồng Phúc | Tìm kiếm dữ liệu và thực hiện kiểm thử công cụ trong Hadoop, định dạng kết quả thu được và lưu được HBase. | 100% |

**MỤC LỤC**

[**GIỚI THIỆU** 1](#_Toc184728151)

[**1.** **Lý do chọn đề tài.** 1](#_Toc184728152)

[**2.** **Mục tiêu báo cáo** 1](#_Toc184728153)

[**3.** **Phương pháp nghiên cứu** 1](#_Toc184728154)

[**CHƯƠNG 1: TỔNG QUAN VỀ CÁC CÔNG NGHỆ** 2](#_Toc184728155)

[**1.1.** **Hadoop – Hệ thống lưu trữ và xử lý dữ liệu phân tán** 2](#_Toc184728156)

[**1.2.** **Hbase – Cơ sở dữ liệu NoSQL dựa trên Hadoop** 3](#_Toc184728157)

[**1.3.** **Zookeeper – Dịch vụ điều phối phân tán** 4](#_Toc184728158)

[**1.4.** **Mahout – Khung học máy phân tán** 5](#_Toc184728159)

[**1.5.** **Sqoop – Công cụ chuyển dữ liệu giữa Hadoop và cơ sở dữ liệu quan hệ** 5](#_Toc184728160)

[**CHƯƠNG 2: CÀI ĐẶT VÀ CẤU HÌNH HỆ THỐNG** 7](#_Toc184728161)

[**2.1. Cài đặt Hbase ở trên máy master và slave1** 7](#_Toc184728162)

[**2.2. Cài đặt và cấu hình Zookeeper trên máy master và slave1** 13](#_Toc184728163)

[**2.3. Cài đặt Mahout** 20](#_Toc184728164)

[**2.4. Cài đặt và cấu hình Sqoop** 20](#_Toc184728165)

[**CHƯƠNG 3: TRIỂN KHAI THỰC TẾ** 25](#_Toc184728166)

[**3.1. Chuẩn bị dữ liệu mẫu MovieRating.csv** 25](#_Toc184728167)

[**3.2. Chuyển dữ liệu vào MySQL** 27](#_Toc184728168)

[**3.3. Phân tích dữ liệu với Mahout** 34](#_Toc184728169)

[**CHƯƠNG 4: KIỂM THỬ VÀ KẾT QUẢ** 36](#_Toc184728170)

[**4.1. Kết quả chuyển dữ liệu từ MySQL vào HDFS và phân tích** 36](#_Toc184728171)

[**4.2. Lưu trữ kết quả vào Hbase** 40](#_Toc184728172)

[**4.3. Đánh giá hiệu suất hệ thống** 46](#_Toc184728173)

[**CHƯƠNG 5: KẾT LUẬN VÀ ĐỀ XUẤT** 46](#_Toc184728174)

[**5.1. Kết luận** 46](#_Toc184728175)

[**5.2. Hạn chế** 47](#_Toc184728176)

[**TÀI LIỆU THAM KHẢO** 49](#_Toc184728177)

**GIỚI THIỆU**

1. **Lý do chọn đề tài.**

Trong kỷ nguyên công nghệ 4.0, dữ liệu lớn (Big Data) đã trở thành yếu tố cốt lõi định hình sự phát triển và cạnh tranh của các tổ chức, doanh nghiệp trên toàn cầu. Các nguồn dữ liệu phong phú, từ mạng xã hội, Internet vạn vật (IoT), giao dịch thương mại điện tử cho đến các video giám sát, đang phát sinh với tốc độ nhanh chóng và khối lượng khổng lồ. Điều này đặt ra không ít thách thức đối với các hệ thống lưu trữ và xử lý dữ liệu truyền thống, đặc biệt là khi dữ liệu có sự đa dạng và tính phức tạp cao.

Để giải quyết vấn đề này, Apache Hadoop – một framework mã nguồn mở mạnh mẽ, đã nổi lên như một giải pháp chủ đạo giúp doanh nghiệp và tổ chức vượt qua những rào cản về chi phí, hiệu suất và quy mô trong việc quản lý và phân tích dữ liệu lớn. Hadoop không chỉ cung cấp một phương thức phân tán dữ liệu hiệu quả qua HDFS (Hadoop Distributed File System), mà còn tối ưu hóa việc xử lý dữ liệu song song qua MapReduce và quản lý tài nguyên qua YARN (Yet Another Resource Negotiator).

Bên cạnh đó, hệ sinh thái Hadoop không ngừng phát triển và bổ sung nhiều công cụ hỗ trợ giúp tối ưu hóa toàn bộ vòng đời của dữ liệu, từ thu thập, lưu trữ đến phân tích và bảo trì. Việc áp dụng Hadoop đã và đang thay đổi cách thức các tổ chức tiếp cận và xử lý dữ liệu lớn, mở ra cơ hội mới cho chuyển đổi số và cải thiện hiệu quả kinh doanh. Chính vì vậy, nghiên cứu và cài đặt hệ thống Hadoop, cùng với các công cụ hỗ trợ như HBase, Zookeeper, Mahout và Sqoop, là một lựa chọn quan trọng trong việc hiểu và tận dụng sức mạnh của dữ liệu lớn trong thời đại công nghệ số.

1. **Mục tiêu báo cáo**

* Cài đặt các hệ thống quản lý dữ liệu phân tán Hadoop, HBase, Zookeeper, Mahout và Sqoop.
* Cấu hình hệ thống để làm việc hiệu quả trong môi trường phân tán.
* Thực hiện phân tích dữ liệu mẫu MovieRating.csv để tạo hệ thống gợi ý phim.

1. **Phương pháp nghiên cứu**

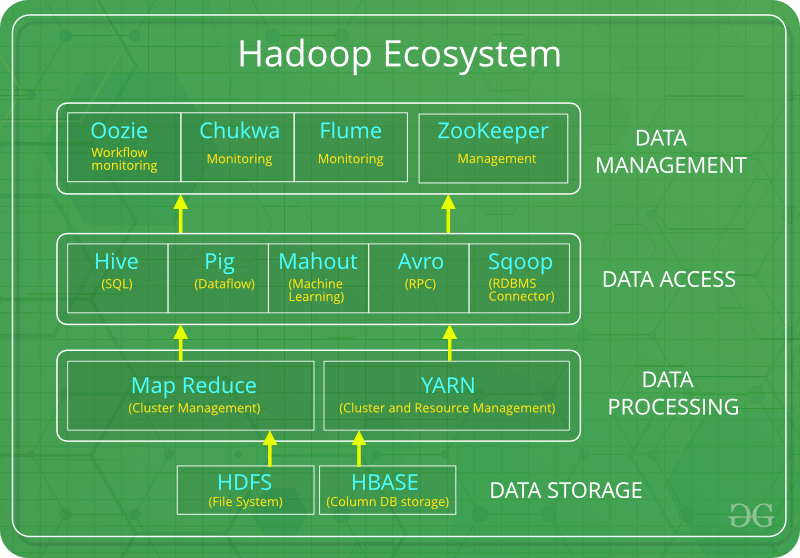
Bao gồm:

* Nghiên cứu tài liệu từ trang chủ của các hệ thống.
* Cài đặt và kiểm thử hệ thống trong môi trường thực tế.
* Phân tích dữ liệu và đánh giá kết quả thông qua các công cụ tích hợp.

# **CHƯƠNG 1: TỔNG QUAN VỀ CÁC CÔNG NGHỆ**

* 1. **Hadoop – Hệ thống lưu trữ và xử lý dữ liệu phân tán**

Hadoop là một hệ thống xử lý dữ liệu lớn phân tán mã nguồn mở, được thiết kế để xử lý các tập dữ liệu có kích thước khổng lồ và phức tạp trên các cụm máy tính (clusters) phân tán. Hadoop cung cấp một nền tảng hiệu quả và tiết kiệm chi phí để lưu trữ và xử lý dữ liệu lớn, giúp các tổ chức và doanh nghiệp xử lý và phân tích dữ liệu với quy mô lớn mà không gặp phải những hạn chế của các hệ thống lưu trữ truyền thống.



Các thành phần chính của Hadoop:

* **HDFS (Hadoop Distributed File System):** Đây là hệ thống lưu trữ phân tán của Hadoop, chịu trách nhiệm phân chia dữ liệu thành các khối nhỏ (block) và lưu trữ chúng trên nhiều máy tính (DataNodes). HDFS cho phép lưu trữ dữ liệu lớn với tính sẵn sàng cao, đảm bảo dữ liệu được sao lưu và phục hồi nhanh chóng khi có sự cố.
* **MapReduce:** Là mô hình lập trình song song của Hadoop, MapReduce giúp phân tán công việc xử lý dữ liệu ra nhiều nút (node) khác nhau trong hệ thống. Tập dữ liệu được chia nhỏ và mỗi phần dữ liệu được xử lý độc lập trên các máy chủ, sau đó kết quả được gom lại. Điều này giúp Hadoop xử lý các tập dữ liệu có kích thước lớn và phức tạp mà các hệ thống xử lý truyền thống không thể đảm nhận.
* **YARN (Yet Another Resource Negotiator):** YARN là thành phần quản lý tài nguyên của Hadoop. Nó giúp phân bổ và quản lý tài nguyên (CPU, bộ nhớ) trên các cụm máy tính, tối ưu hóa việc sử dụng tài nguyên và đảm bảo các công việc được thực hiện một cách hiệu quả.

Hadoop là nền tảng lý tưởng để xây dựng các hệ thống phân tán quy mô lớn, có thể xử lý hàng terabyte hoặc petabyte dữ liệu một cách hiệu quả và chi phí thấp.

* 1. **Hbase – Cơ sở dữ liệu NoSQL dựa trên Hadoop**

HBase là một cơ sở dữ liệu NoSQL mã nguồn mở chạy trên nền tảng Hadoop, được thiết kế để lưu trữ và xử lý dữ liệu có cấu trúc lớn. HBase cung cấp khả năng lưu trữ dữ liệu dưới dạng bảng (table), với các hàng (row) và cột (column) có thể mở rộng theo quy mô phân tán. Đây là cơ sở dữ liệu rất phù hợp cho các ứng dụng yêu cầu truy xuất dữ liệu theo thời gian thực hoặc dữ liệu có kích thước cực lớn.

Các tính năng nổi bật của HBase:

* **Cơ sở dữ liệu phân tán:** HBase sử dụng kiến trúc phân tán để phân phối dữ liệu trên nhiều nút trong một cụm máy tính (cluster). Điều này giúp HBase xử lý khối lượng dữ liệu khổng lồ với tính sẵn sàng và khả năng mở rộng cao.
* **Chịu lỗi và sao lưu:** HBase có tính năng sao lưu dữ liệu và phục hồi khi có sự cố. Mỗi bản sao dữ liệu được lưu trữ ở các nút khác nhau, giúp hệ thống duy trì hoạt động bình thường ngay cả khi có nút bị hỏng.
* **Truy xuất nhanh và xử lý thời gian thực:** HBase cung cấp khả năng đọc và ghi dữ liệu nhanh chóng, thích hợp cho các ứng dụng cần truy xuất dữ liệu theo thời gian thực, chẳng hạn như hệ thống nhận dạng, phân tích dữ liệu lớn, hoặc các ứng dụng web quy mô lớn.
* **Tích hợp với Hadoop:** HBase được xây dựng để chạy trên Hadoop và có thể tích hợp chặt chẽ với HDFS để lưu trữ dữ liệu. HBase còn sử dụng MapReduce để xử lý dữ liệu lớn trong hệ thống Hadoop.

HBase rất phù hợp cho các ứng dụng yêu cầu lưu trữ và truy xuất dữ liệu theo hàng triệu hoặc tỷ lệ đơn vị dữ liệu, và nó đã được nhiều doanh nghiệp sử dụng để hỗ trợ các hệ thống có quy mô lớn như Amazon, Facebook và Yahoo.

* 1. **Zookeeper – Dịch vụ điều phối phân tán**

Zookeeper là một dịch vụ điều phối phân tán mã nguồn mở, được phát triển để giúp các hệ thống phân tán duy trì tính nhất quán, đồng bộ và bảo mật dữ liệu. Zookeeper cung cấp một bộ công cụ mạnh mẽ để quản lý các dịch vụ phân tán và giúp các ứng dụng đồng bộ hóa trạng thái của chúng trong môi trường phân tán, từ đó đảm bảo tính toàn vẹn của hệ thống.

Các tính năng nổi bật của HBase:

* **Điều phối và đồng bộ hóa:** Zookeeper đảm bảo rằng các máy chủ trong một hệ thống phân tán có thể giao tiếp với nhau một cách hiệu quả và đồng bộ. Điều này cực kỳ quan trọng trong các ứng dụng như quản lý khóa phân tán, cấu hình hệ thống, và quản lý trạng thái ứng dụng.
* **Quản lý dữ liệu phân tán:** Zookeeper cung cấp một kho lưu trữ dữ liệu dạng key-value để giữ thông tin về các trạng thái của hệ thống. Các nút (node) trong Zookeeper có thể được dùng để lưu trữ các dữ liệu cần thiết cho việc đồng bộ hóa và điều phối các tác vụ giữa các nút trong hệ thống phân tán.
* **Chịu lỗi và phục hồi:** Zookeeper có cơ chế đảm bảo tính khả dụng cao và phục hồi dữ liệu sau khi gặp sự cố. Nó duy trì một bản sao của dữ liệu trên các máy chủ khác nhau trong cụm và tự động chuyển đổi khi một máy chủ bị lỗi.

Zookeeper giúp đảm bảo rằng các ứng dụng phân tán có thể duy trì tính toàn vẹn của dữ liệu và đồng bộ hóa trong môi trường với nhiều máy chủ.

* 1. **Mahout – Khung học máy phân tán**

Apache Mahout là một khung học máy mã nguồn mở, được thiết kế để thực hiện các thuật toán học máy phân tán trên các cụm máy tính, tận dụng sức mạnh của hệ thống Hadoop. Mahout hỗ trợ các thuật toán học máy cho phân tích dữ liệu lớn, chẳng hạn như phân loại, hồi quy, phân nhóm và khuyến nghị.

Các thuật toán chính trong Mahout:

* **Thuật toán phân nhóm (Clustering):** Mahout hỗ trợ các thuật toán phân nhóm như K-means, giúp phân chia các dữ liệu lớn thành các nhóm tương tự nhau.
* **Thuật toán phân loại (Classification):** Các thuật toán như Naive Bayes và Random Forest có thể được sử dụng để phân loại dữ liệu dựa trên các đặc trưng đã học.
* **Thuật toán khuyến nghị (Recommendation):** Mahout cung cấp các thuật toán để xây dựng hệ thống gợi ý, giúp phân tích hành vi người dùng và đưa ra những khuyến nghị dựa trên dữ liệu lịch sử.

Mahout tích hợp với Hadoop và HDFS, cho phép thực hiện các phép toán học và học máy trên các dữ liệu lớn và phân tán, giúp tối ưu hóa hiệu quả phân tích và xử lý dữ liệu.

* 1. **Sqoop – Công cụ chuyển dữ liệu giữa Hadoop và cơ sở dữ liệu quan hệ**

Sqoop là một công cụ mã nguồn mở được thiết kế để chuyển dữ liệu giữa Hadoop và các cơ sở dữ liệu quan hệ (RDBMS) như MySQL, PostgreSQL, Oracle, v.v. Sqoop giúp đưa dữ liệu từ các cơ sở dữ liệu quan hệ vào HDFS hoặc HBase để phân tích, và cũng hỗ trợ việc xuất dữ liệu từ HDFS về lại các cơ sở dữ liệu quan hệ.

Các tính năng của Sqoop:

* **Chuyển dữ liệu hiệu quả:** Sqoop hỗ trợ việc nhập và xuất dữ liệu từ cơ sở dữ liệu quan hệ vào Hadoop một cách nhanh chóng và hiệu quả, với khả năng xử lý dữ liệu lớn.
* **Tích hợp chặt chẽ với Hadoop:** Sqoop có thể tích hợp với HDFS, HBase và Hive để lưu trữ dữ liệu, cho phép chuyển dữ liệu từ các hệ thống quan hệ vào các hệ thống phân tán Hadoop một cách dễ dàng.
* **Khả năng mở rộng:** Sqoop có thể xử lý và chuyển tải khối lượng lớn dữ liệu từ các cơ sở dữ liệu quan hệ vào Hadoop mà không gặp phải vấn đề về hiệu suất, giúp tiết kiệm thời gian và tài nguyên trong việc chuyển đổi dữ liệu.

Sqoop giúp các tổ chức có thể dễ dàng tích hợp và làm việc với dữ liệu từ các hệ thống quan hệ truyền thống và các công cụ phân tích dữ liệu lớn hiện đại như Hadoop.

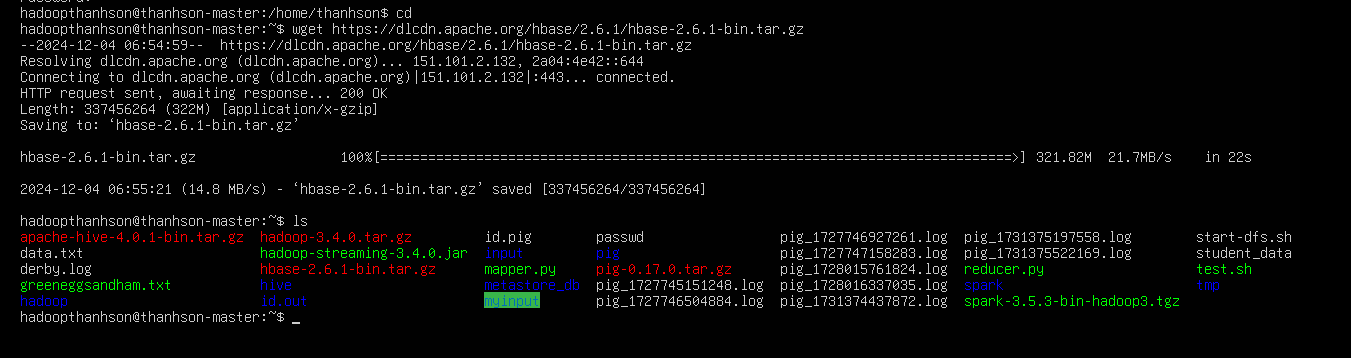
# **CHƯƠNG 2: CÀI ĐẶT VÀ CẤU HÌNH HỆ THỐNG**

## **2.1. Cài đặt Hbase ở trên máy master và slave1**

Hadoop là nền tảng cơ bản cho hệ thống dữ liệu phân tán, cung cấp hệ thống tệp phân tán (HDFS) và khả năng xử lý dữ liệu song song (MapReduce). Dưới đây là các bước cài đặt:

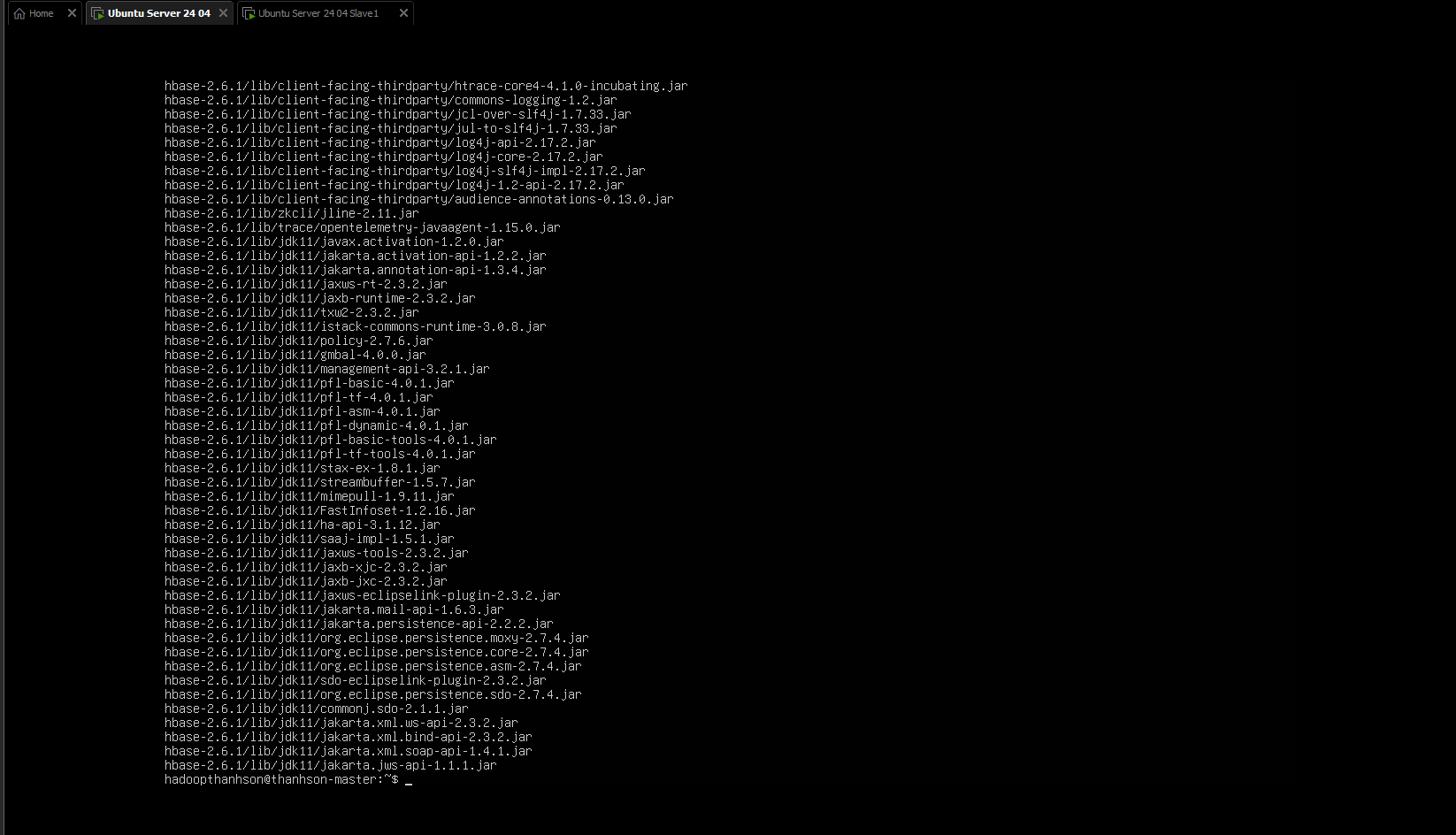
Bước 1: Tải Hadoop

|  |
| --- |
| # wget <https://dlcdn.apache.org/hbase/2.6.1/hbase-2.6.1-bin.tar.gz> |



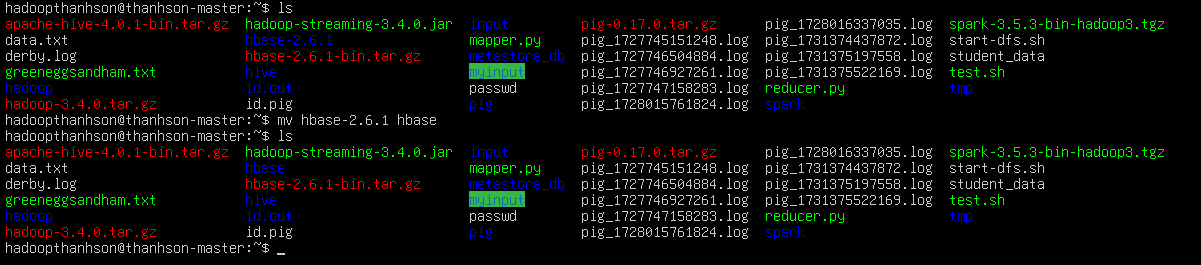
Bước 2: Giải nén và cấu hình

|  |
| --- |
| # tar -xzvf hbase-2.6.1-bin.tar.gz |



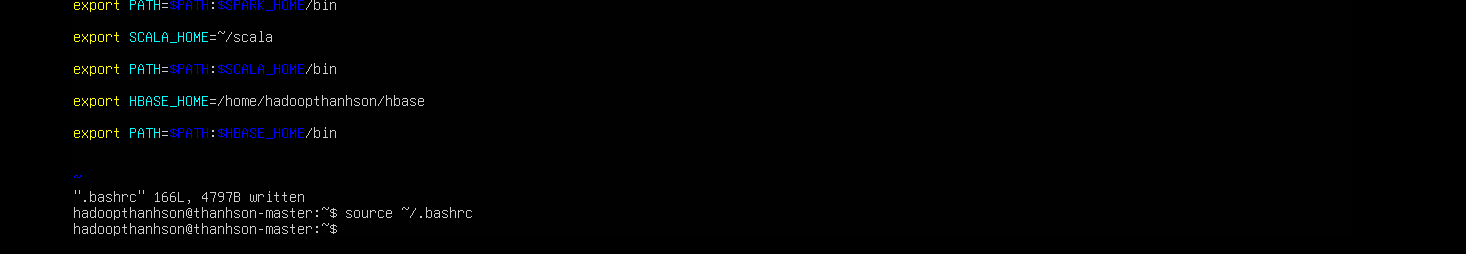
Đổi tên biến thư mục

|  |
| --- |
| # mv hbase-2.6.1 hbase |



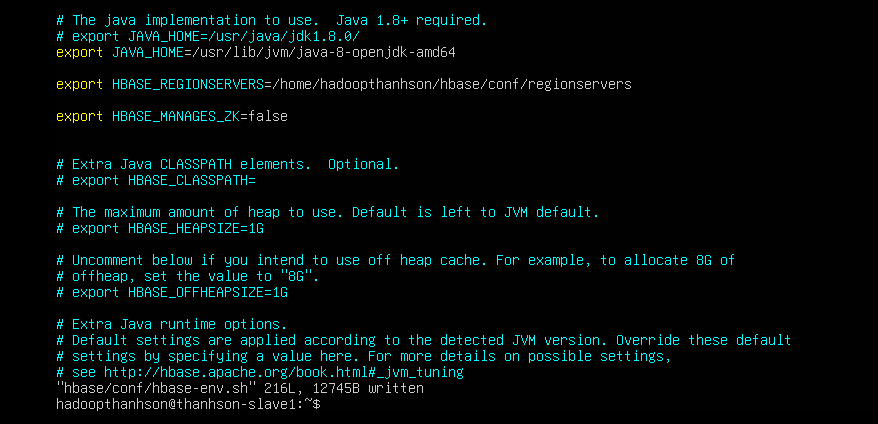
Khai báo biến môi trường HBASE\_HOME

|  |
| --- |
| vim ~/.bashrc  $ export HBASE\_HOME=/home/hadoopthanhson/hbase  $ export PATH=$PATH:$HBASE\_HOME/bin  source ~/.bashrc |



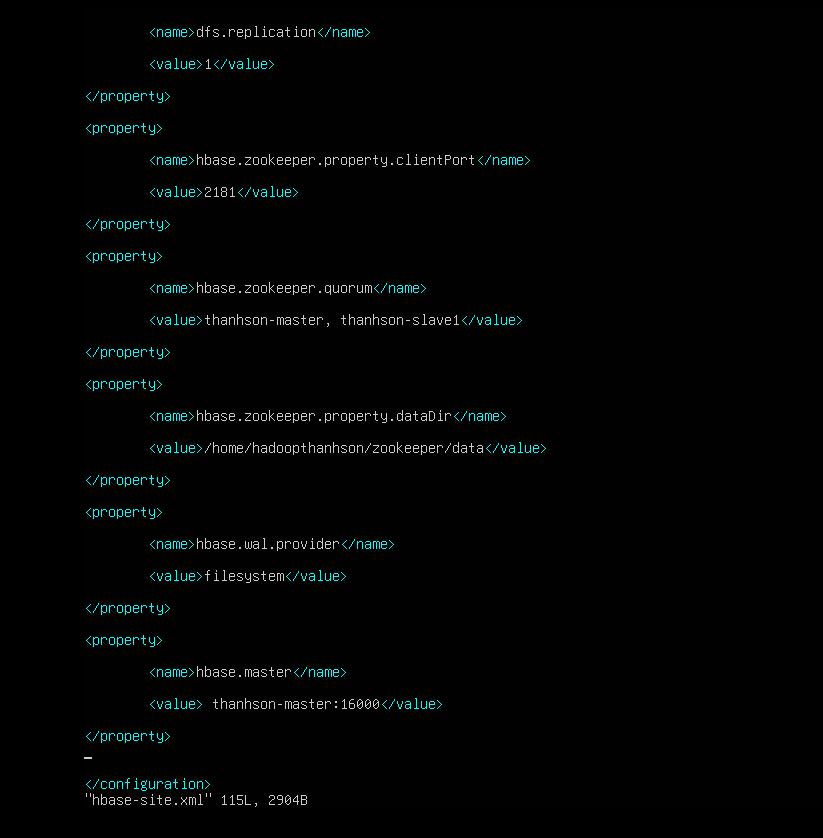
Cấu hình HBase

|  |
| --- |
| #  vim  ~/hbase/conf/hbase-env.sh  export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64  export HBASE\_REGIONSERVERS=/home/hadoopthanhson/hbase/conf/regionservers  export HBASE\_MANAGES\_ZK=false |

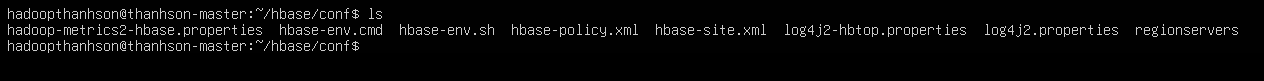
****

|  |
| --- |
| #  vim  ~/hbase/conf/hbase-site.xml |

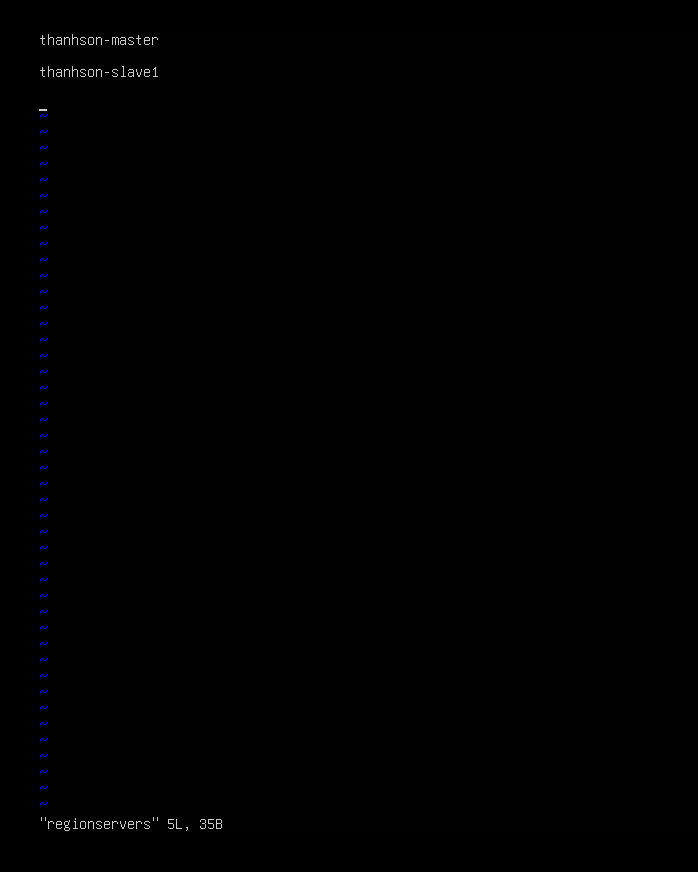
|  |
| --- |
| <property>  <name>hbase.rootdir</name>  <value>hdfs://thanhson-master:9000/hbase</value>  </property>  <property>  <name>hbase.cluster.distributed</name>  <value>true</value>  </property>  <property>  <name>hbase.unsafe.stream.capability.enforce</name>  <value>false</value>  </property>  <property>  <name>dfs.replication</name>  <value>1</value>  </property>  <property>  <name>hbase.zookeeper.property.clientPort</name>  <value>2181</value>  </property>  <property>  <name>hbase.zookeeper.quorum</name>  <value>thanhson-master,thanhson-slave1</value>  </property>  <property>  <name>hbase.zookeeper.property.dataDir</name>  <value>/home/hadoopthanhson/zookeeper/data</value>  </property>  <property>  <name>hbase.wal.provider</name>  <value>filesystem</value>  </property>  <property>  <name>hbase.master</name>  <value> thanhson-master:16000</value>  </property> |



Thiết lập Regionsevers

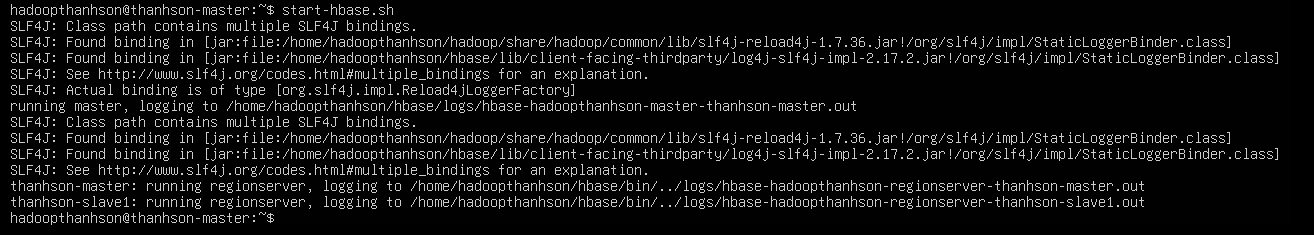


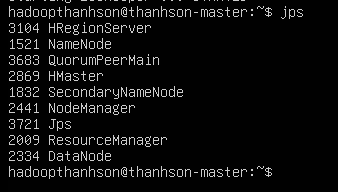
|  |
| --- |
| thanhson-master  thanhson-slave1 |



Khởi động HBase:

|  |
| --- |
| start-hbase.sh |

****

****

## **2.2. Cài đặt và cấu hình Zookeeper trên máy master và slave1**

Bước 1: Tải và giải nén Zookeeper

|  |
| --- |
| # wget <https://dlcdn.apache.org/zookeeper/zookeeper-3.8.4/apache-zookeeper-3.8.4-bin.tar.gz> |



|  |
| --- |
| tar -xzf apache-zookeeper-3.8.4-bin.tar.gz |



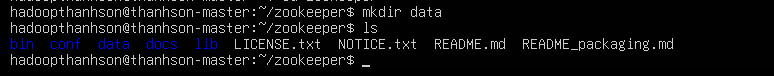
|  |
| --- |
| mv apache-zookeeper-3.8.4-bin zookeeper |



Bước 2: Tạo file cấu hình

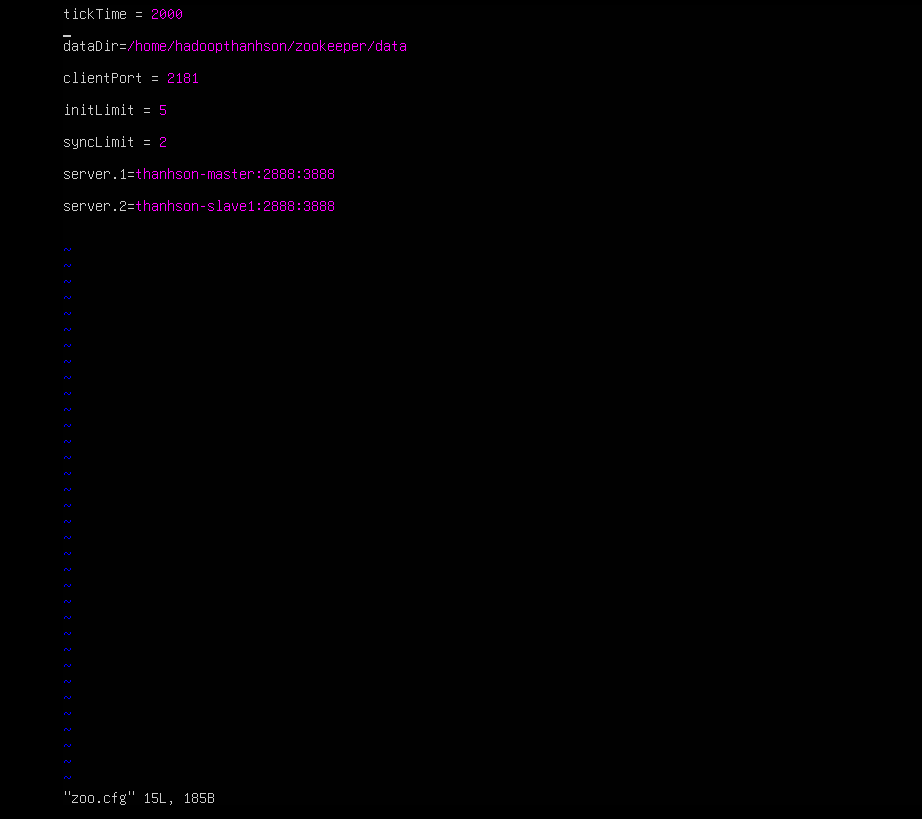
Tạo thư mục dữ liệu và file cấu hình:

|  |
| --- |
| $ cd zookeeper  $ mkdir data |



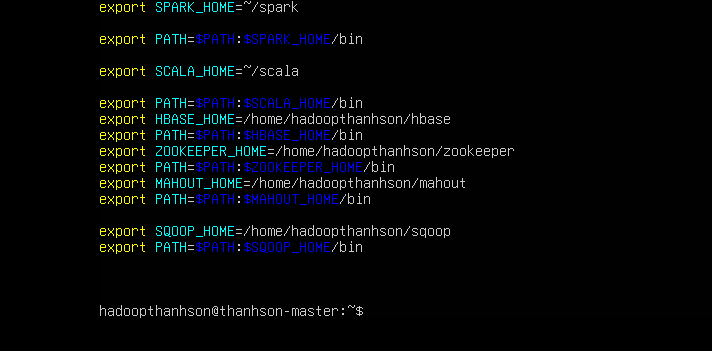
|  |
| --- |
| # vi conf/zoo.cfg |

|  |
| --- |
| tickTime = 2000  dataDir=/home/hadoopthanhson/zookeeper/data  clientPort = 2181  initLimit = 10  syncLimit = 10  server.1=thanhson-master:2888:3888  server.2=thanhson-slave1:2888:3888 |



Khai báo biến môi trường Zookeeper\_Home

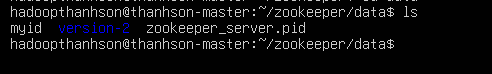
|  |
| --- |
| vim ~/.bashrc  $ export ZOOKEEPER\_HOME=/home/hadoopthanhson/zookeeper  $ export PATH=$PATH:$ZOOKEEPER\_HOME/bin  source ~/.bashrc |



Tạo thư mục myid

**master:**

|  |
| --- |
| echo 1 > /home/hadoopthanhson/zookeeper/data/myid |



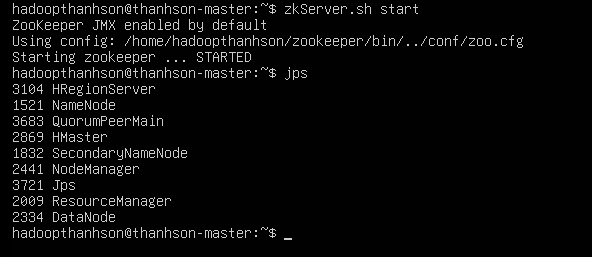
**slave:**

|  |
| --- |
| echo 2 > /home/hadoopthanhson/zookeeper/data/myid |

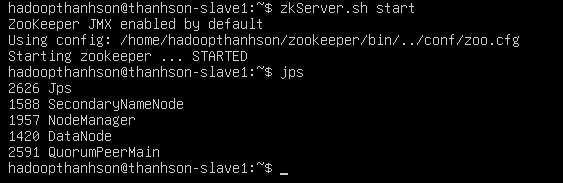


Kiểm tra xem zookeeper được START và jps đủ node chưa:

**master**

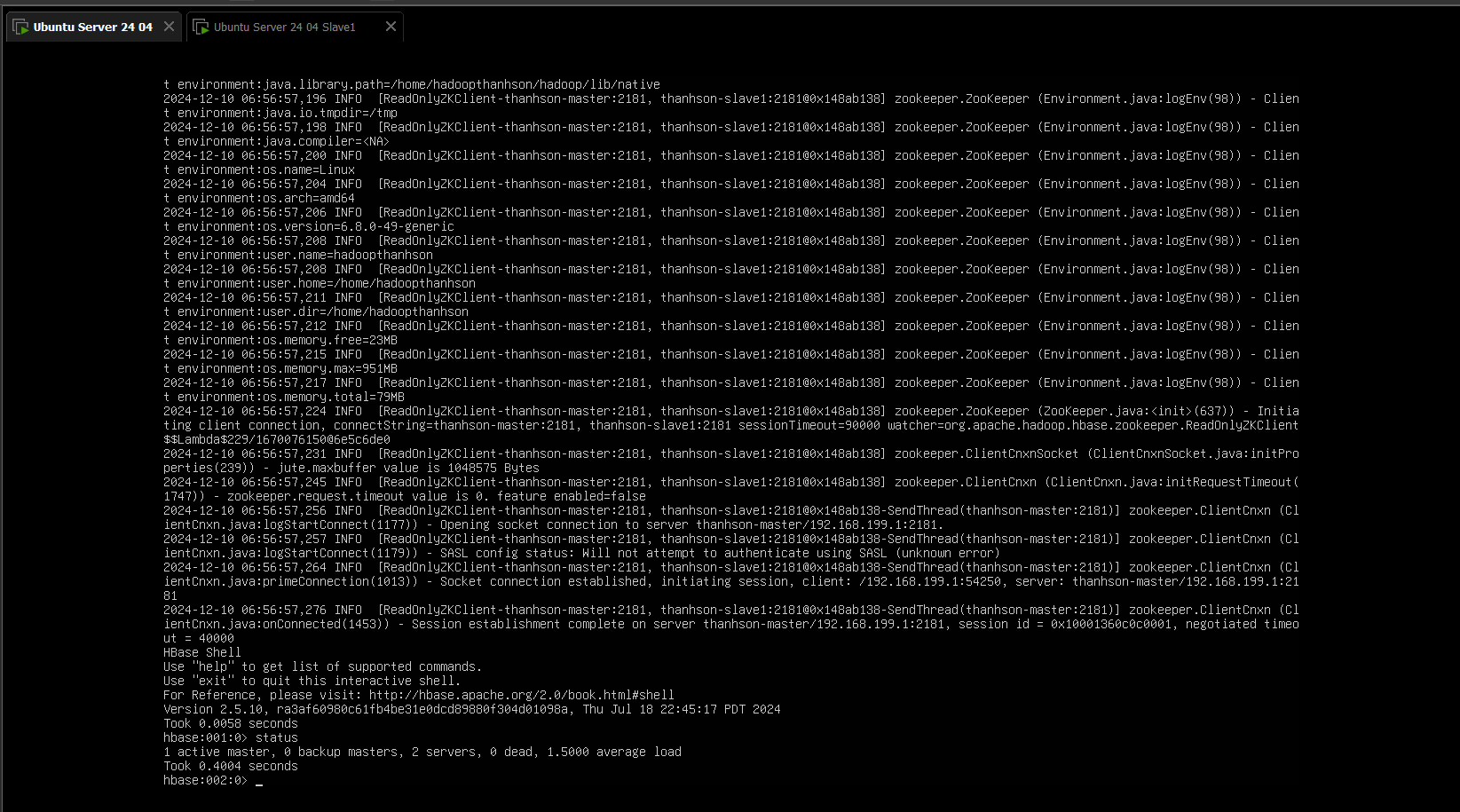


**slave1**



Bước 3: Khởi động HBase có Zookeeper quản lý

|  |
| --- |
| start-hbase.sh  status |



## **2.3. Cài đặt Mahout**

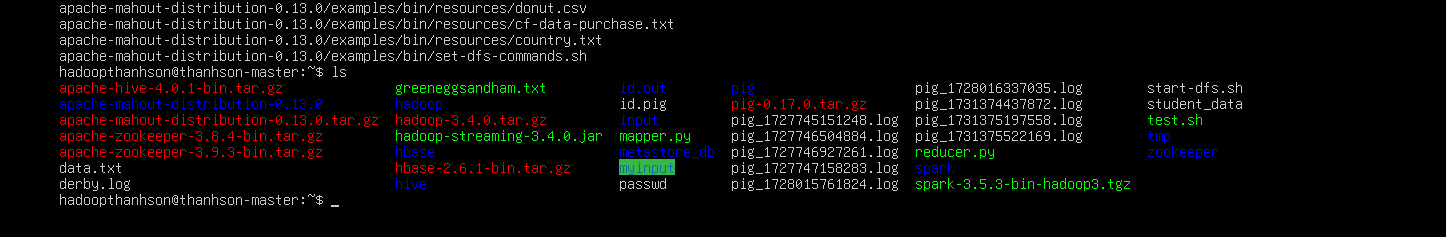
Bước 1: Tải và giải nén Mahout

|  |
| --- |
| wget <https://archive.apache.org/dist/mahout/0.13.0/apache-mahout-distribution-0.13.0.tar.gz> |

****

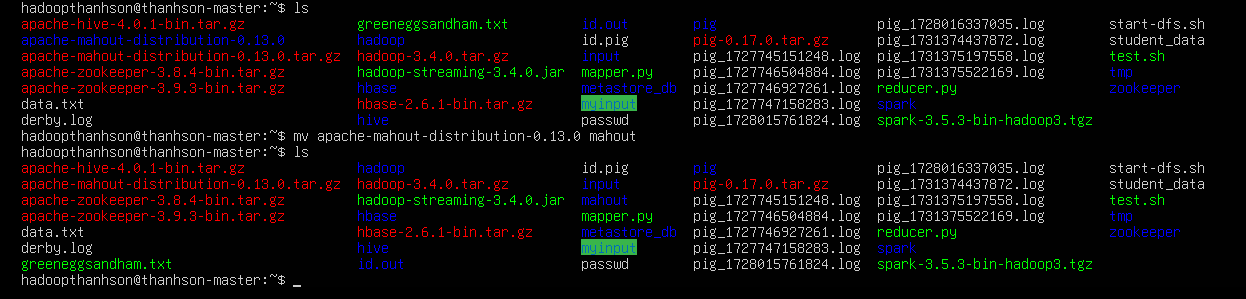
Giải nén

|  |
| --- |
| # tar -xvzf [apache-mahout-distribution-0.13.0.tar.gz](https://archive.apache.org/dist/mahout/0.13.0/apache-mahout-distribution-0.13.0.tar.gz) |



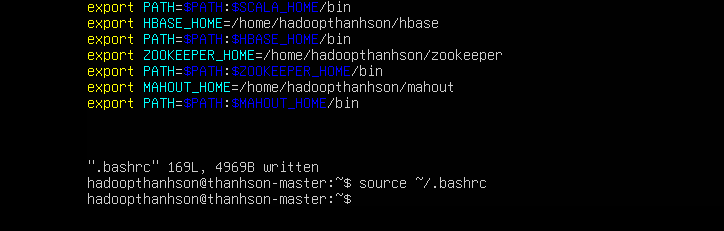
Đổi tên

|  |
| --- |
| # mv apache-mahout-distribution-0.13.0 mahout |



Bước 2: Cấu hình biến môi trường

|  |
| --- |
| $ vim ~/.bashrc  #export MAHOUT\_HOME=/home/hadoopthanhson/mahout  #export PATH=$PATH:$MAHOUT\_HOME/bin  $ source ~/.bashrc |

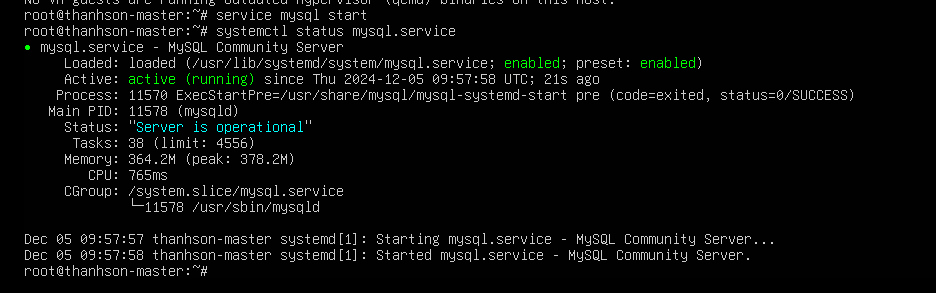


## **2.4. Cài đặt và cấu hình Sqoop**

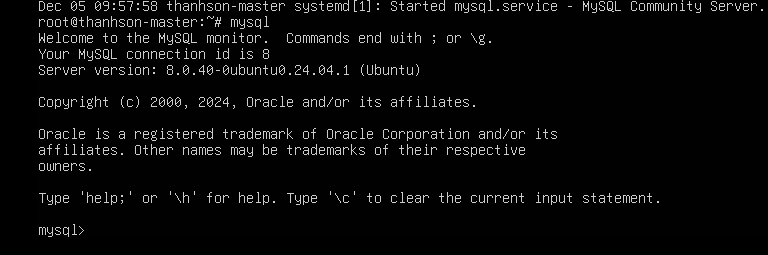
Bước 1: Cài MySQL

(Các bước cài MySQL sẽ chạy trong root)

|  |
| --- |
| apt update  apt-get install mysql-server  service mysql start  systemctl status mysql.service |



|  |
| --- |
| mysql |

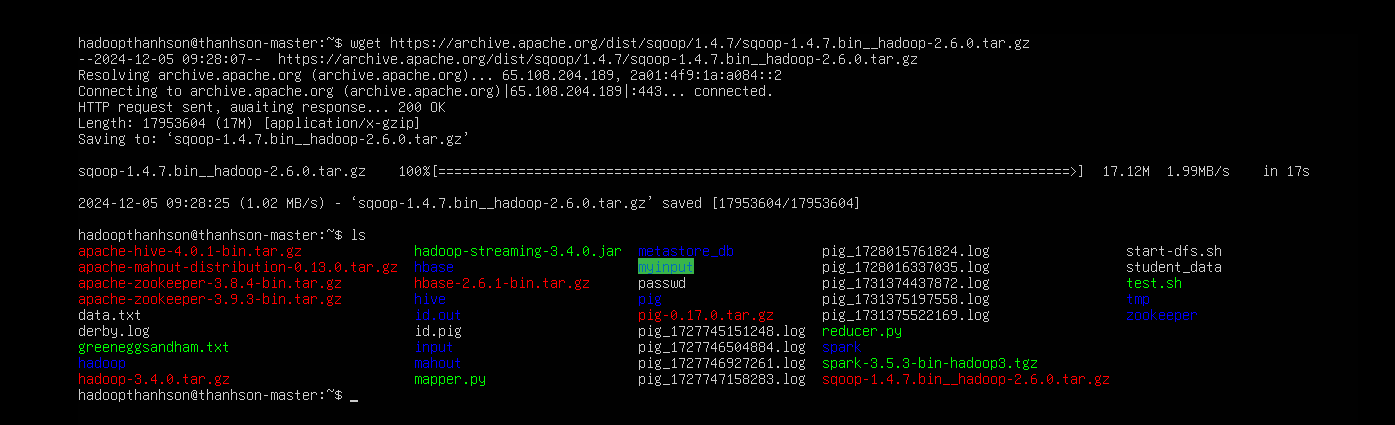


Để thoát thì: mysql>exit

Bước 2: Cài Sqoop

* + 1. Tải và giải nén

|  |
| --- |
| # wget <https://archive.apache.org/dist/sqoop/1.4.7/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz> |

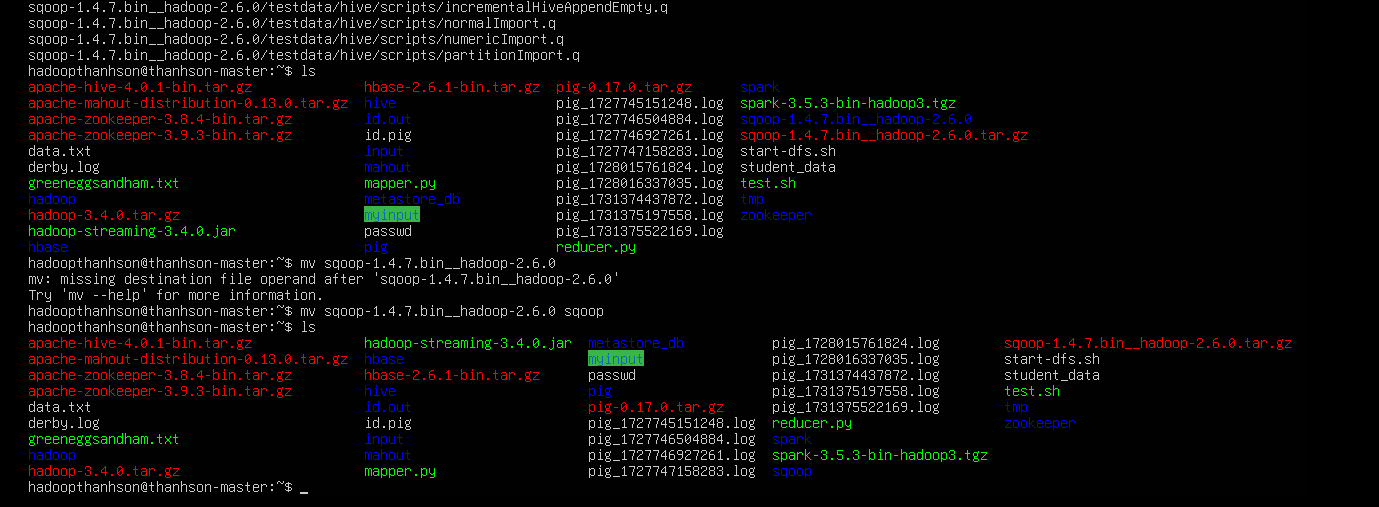


Giải nén:

|  |
| --- |
| # tar -xvf sqoop-1.4.7.bin\_\_hadoop-2.6.0.tar.gz |

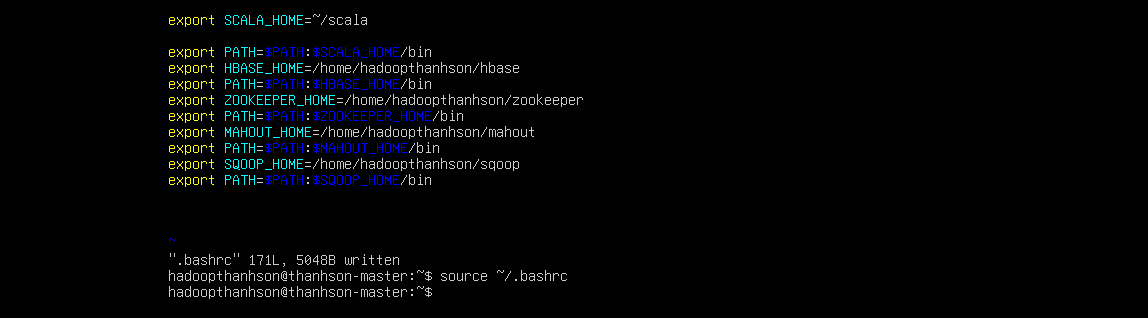
Đổi tên:

|  |
| --- |
| # mv sqoop-1.4.7.bin\_\_hadoop-2.6.0 sqoop |



* + 1. Cấu hình biến môi trường

|  |
| --- |
| $ vim ~/.bashrc  # export SQOOP\_HOME=/home/hadoopthanhson/sqoop  # export PATH=$PATH:$SQOOP\_HOME/bin  $ source ~/.bashrc |



Tại /sqoop/conf đổi tên tệp cấu hình:

|  |
| --- |
| # mv sqoop-env-template.sh sqoop-env.sh |

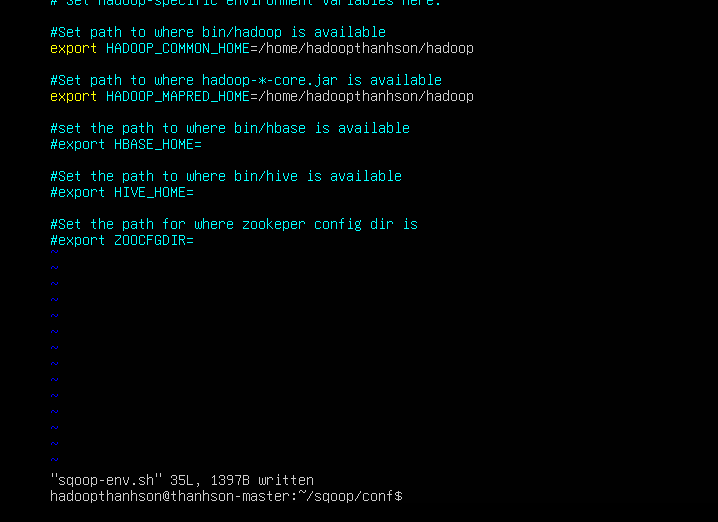


Chỉnh sửa sqoop-env.sh

|  |
| --- |
| # vim sqoop-env.sh |

Chỉnh sửa thêm phần

|  |
| --- |
| # export HADOOP\_COMMON\_HOME=/home/hadoopthanhson/hadoop  # export HADOOP\_MAPRED\_HOME=/home/hadoopthanhson/hadoop |

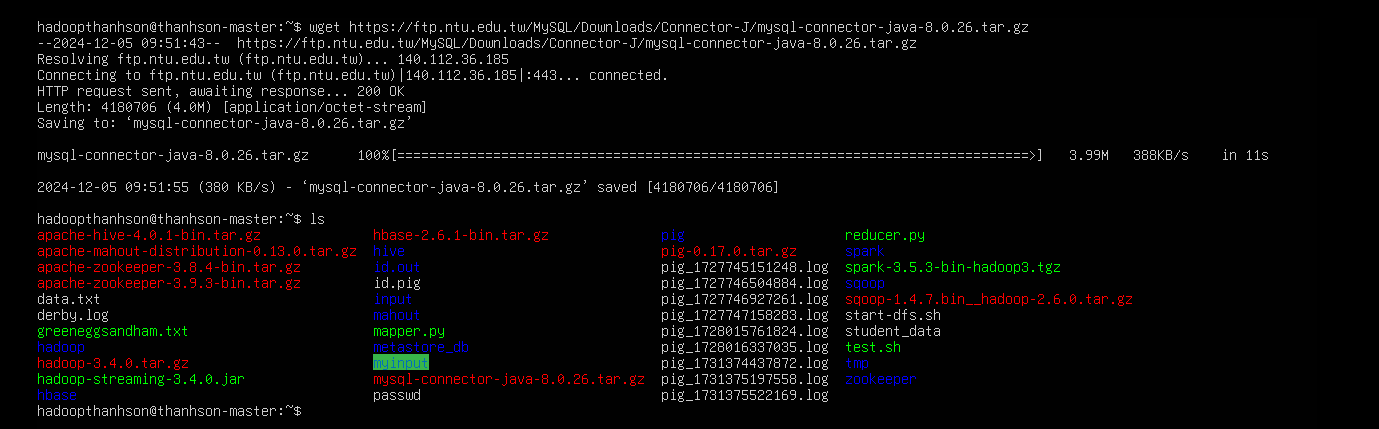


|  |
| --- |
| source ~/.bashrc |

Bước 3: Cài SQL Connector

* + 1. Tải và giải nén

|  |
| --- |
| # wget <https://ftp.ntu.edu.tw/MySQL/Downloads/Connector-J/mysql-connector-java-8.0.26.tar.gz> |

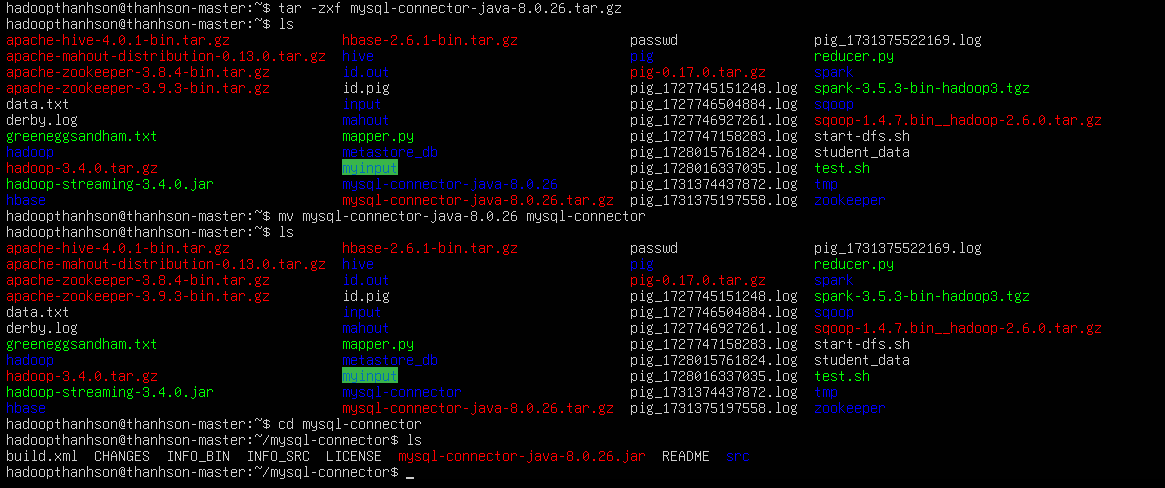


Giải nén

|  |
| --- |
| # tar -zxf mysql-connector-java-8.0.26.tar.gz |

Đổi tên:

|  |
| --- |
| # mv mysql-connector-java-8.0.26 mysql-connector |

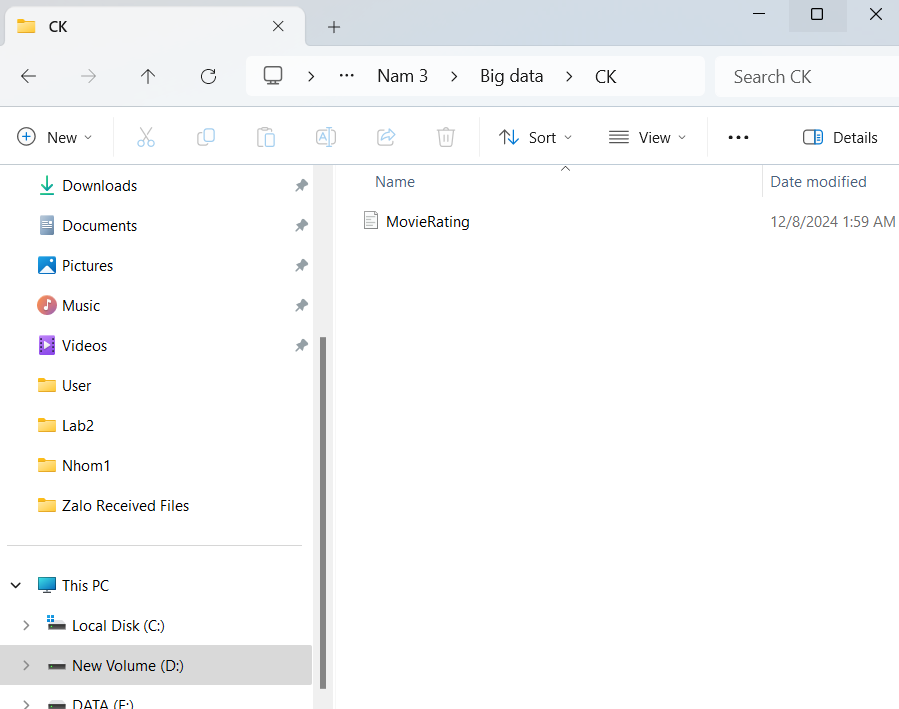


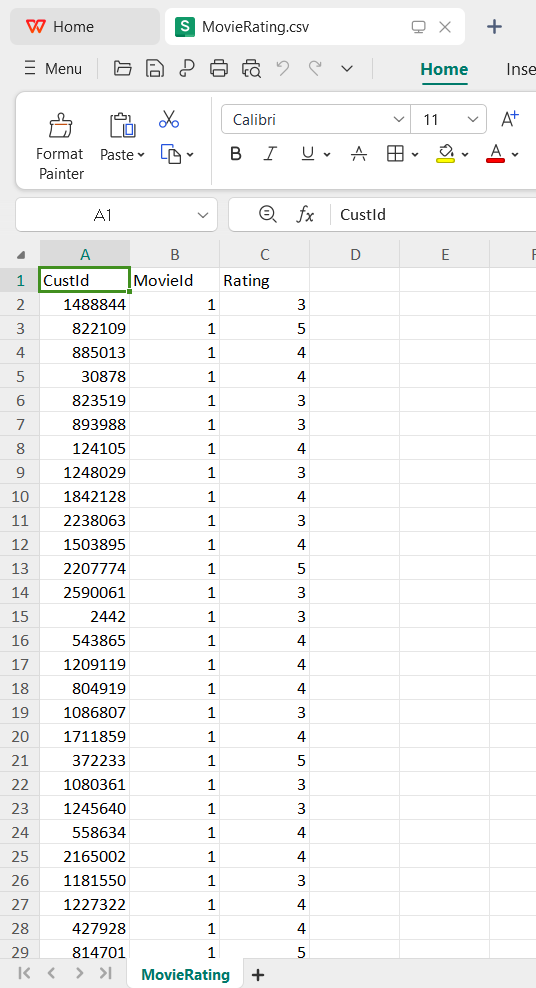
# **CHƯƠNG 3: TRIỂN KHAI THỰC TẾ**

## **3.1. Chuẩn bị dữ liệu mẫu MovieRating.csv**

Tập dữ liệu **MovieRating.csv** chứa các đánh giá của người dùng về các bộ phim, với ba trường dữ liệu chính:

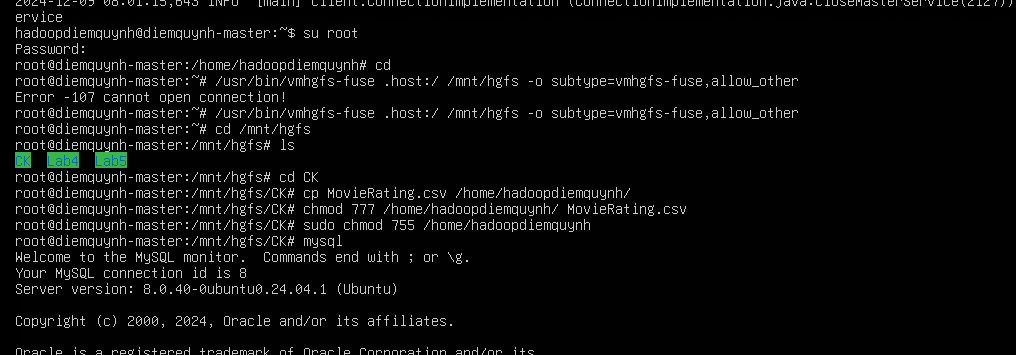
* **CustId**: ID của khách hàng.
* **MovieId**: ID của bộ phim.
* **Rating**: Điểm đánh giá của khách hàng đối với bộ phim (thang điểm từ 1 đến 5).

****



Chuyển tập dữ liệu từ máy vô máy ảo

|  |
| --- |
| /usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow\_other  cd /mnt/hgfs  ls  cd CK |
| cp MovieRating.csv /home/hadoopdiemquynh/  chmod 777 /home/hadoopdiemquynh/ MovieRating.csv |



Dữ liệu này sẽ được sử dụng trong quá trình:

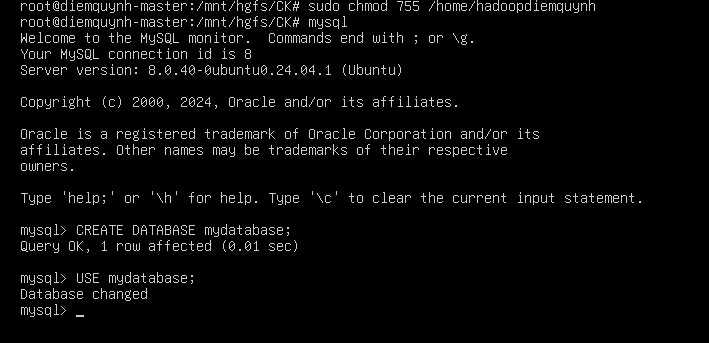
1. **Chuyển dữ liệu vào MySQL**.
2. **Dùng Sqoop để chuyển dữ liệu từ MySQL vào HDFS**.
3. **Sử dụng Mahout để phân tích dữ liệu từ HDFS**.
4. **Chuyển kết quả phân tích vào HBase**.

## **3.2. Chuyển dữ liệu vào MySQL**

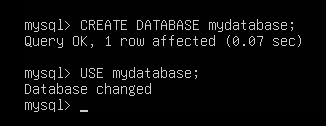
Các bước thực hiện:

**Cài đặt và cấu hình MySQL:** Trước tiên, chúng tôi cần tạo cơ sở dữ liệu trong MySQL và chèn dữ liệu từ tập tin MovieRating.csv vào bảng MySQL.

Vào MySQL, trong root:

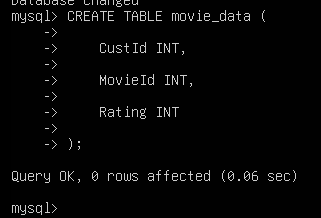
Tạo cơ sở dữ liệu mydatabase

|  |
| --- |
| CREATE DATABASE mydatabase;  USE mydatabase; |



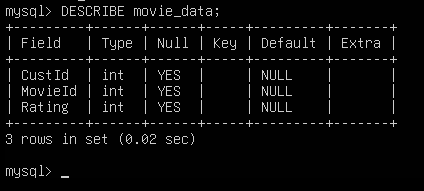
**Tạo bảng movie\_data**

|  |
| --- |
| CREATE TABLE movie\_data (      CustId INT,      MovieId INT,      Rating INT  ); |



Kiểm tra bảng vừa được tạo:

|  |
| --- |
| DESCRIBE movie\_data; |



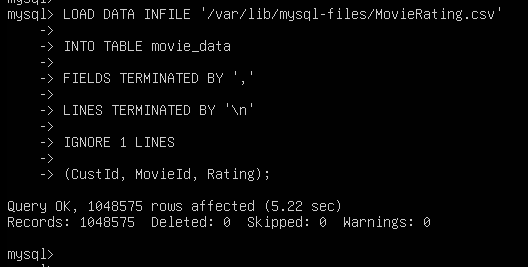
**Chèn dữ liệu vào bảng MySQL:** Để nhập dữ liệu từ tập tin CSV vào MySQL, chúng tôi sử dụng lệnh SQL sau:

Chuyển dữ liệu MovieRating.csv vào thư viện lưu trữ:

|  |
| --- |
| sudo mv /home/hadoopdiemquynh/MovieRating.csv /var/lib/mysql-files/ |

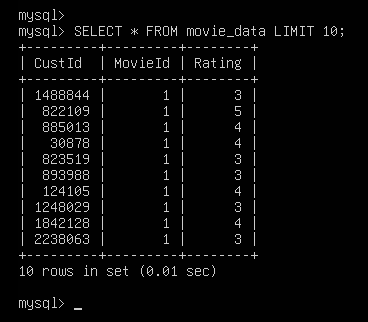
Sau đó load dữ liệu vào bảng movie\_data:

|  |
| --- |
| LOAD DATA INFILE '/var/lib/mysql-files/MovieRating.csv'  INTO TABLE movie\_data  FIELDS TERMINATED BY ','  LINES TERMINATED BY '\n'  IGNORE 1 LINES  (CustId, MovieId, Rating); |



Kiểm tra cơ bản dữ liệu đã load hay chưa:

|  |
| --- |
| SELECT \* FROM movie\_data LIMIT 10; |



**Sử dụng Sqoop để chuyển dữ liệu vào HDFS:** Sau khi dữ liệu đã được chèn vào MySQL, chúng tôi sử dụng Sqoop để chuyển dữ liệu từ MySQL vào HDFS, giúp lưu trữ và xử lý dữ liệu phân tán.

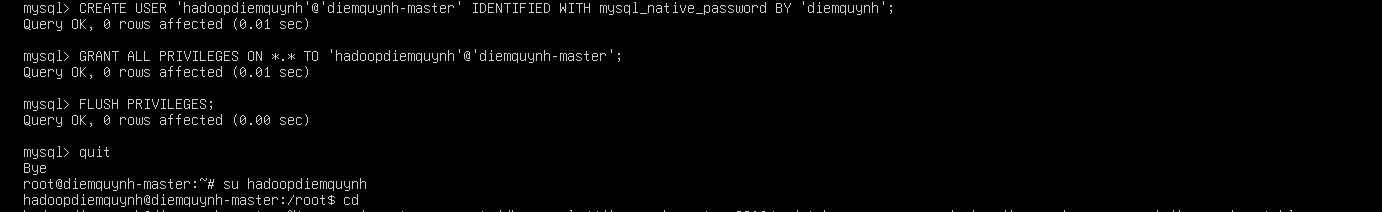
Di chuyển công cụ MySQL connector vào thư viện của Sqoop:

|  |
| --- |
| cp /home/hadoopdiemquynh/mysql-connector/mysql-connector-java-8.0.26.jar /home/hadoopdiemquynh/sqoop/lib/ |



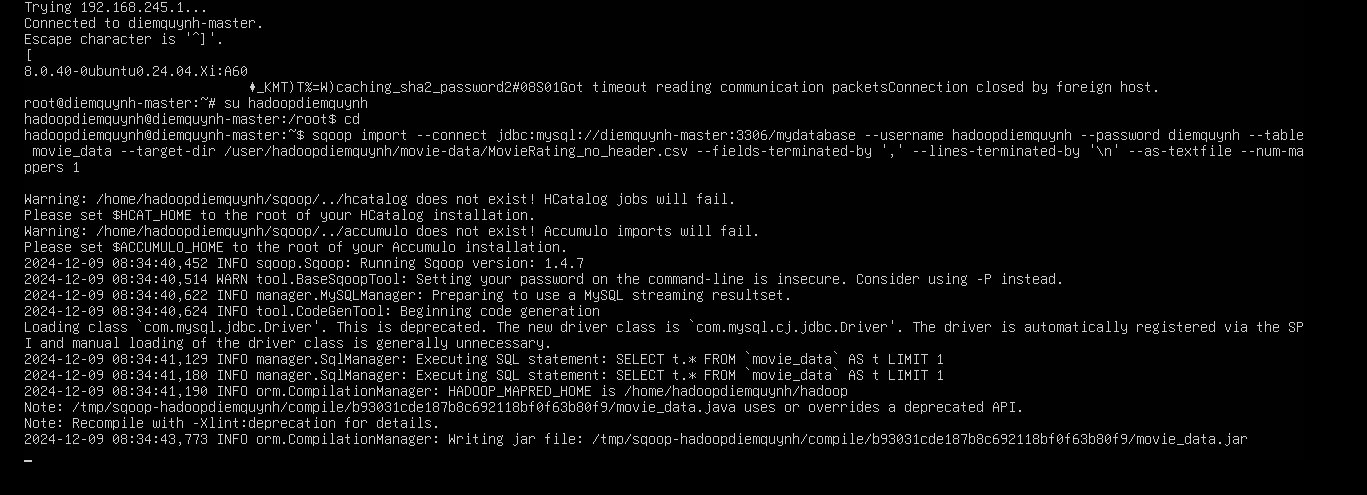
Tạo quyền truy cập vào mysql:

|  |
| --- |
| CREATE USER 'hadoopdiemquynh'@'diemquynh-master' IDENTIFIED WITH mysql\_native\_password BY 'diemquynh';  GRANT ALL PRIVILEGES ON \*.\* TO 'hadoopdiemquynh'@'diemquynh-master';  FLUSH PRIVILEGES; |



Dùng câu lệnh Sqoop để chuyển dữ liệu từ MySQL vào HDFS:

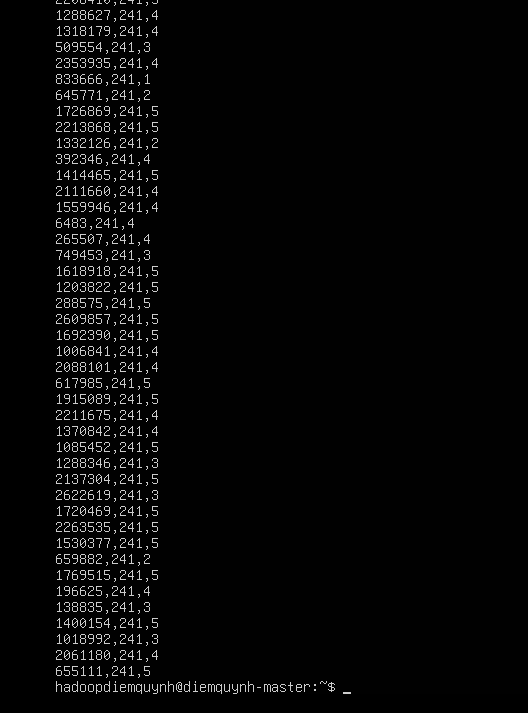
|  |
| --- |
| sqoop import --connect jdbc:mysql://diemquynh-master:3306/mydatabase --username hadoopdiemquynh --password diemquynh --table movie\_data --target-dir /user/hadoopdiemquynh/movie-data/MovieRating\_no\_header.csv --fields-terminated-by ',' --lines-terminated-by '\n' --as-textfile --num-mappers 1 |



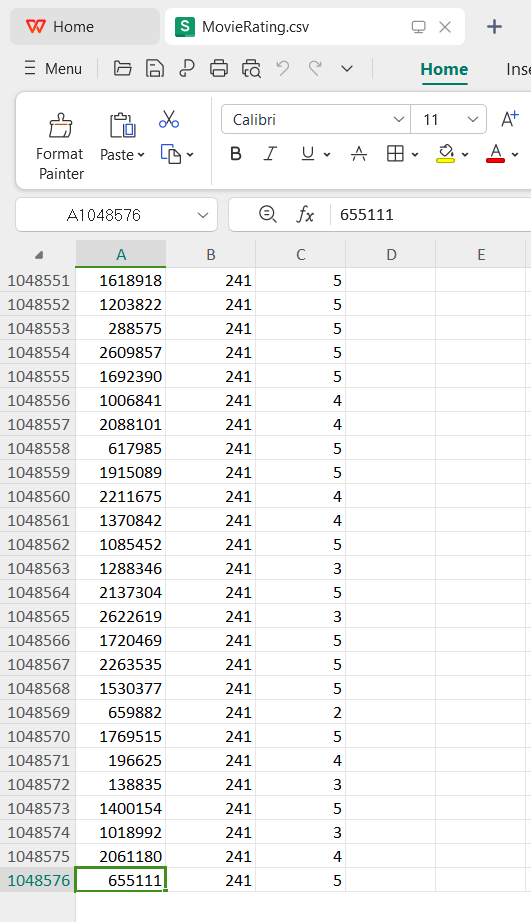
Lệnh trên sẽ nhập toàn bộ bảng ratings vào thư mục /user/hadoop/movie\_ratings trên HDFS, giúp dữ liệu sẵn sàng cho các bước phân tích tiếp theo.

Kiểm tra kết quả:

|  |
| --- |
| hdfs dfs -ls /user/hadoopdiemquynh/movie-data/MovieRating\_no\_header.csv  hdfs dfs -cat /user/hadoopdiemquynh/movie-data/MovieRating\_no\_header.csv/part-m-00000 |



Đối xứng với dữ liệu ban đầu, dữ liệu được truyền vào hoàn toàn.



## **3.3. Phân tích dữ liệu với Mahout**

Sau khi dữ liệu đã được chuyển vào HDFS, bước tiếp theo là sử dụng **Mahout** để phân tích dữ liệu và xây dựng hệ thống khuyến nghị. Mahout cung cấp các thuật toán học máy mạnh mẽ để thực hiện các tác vụ như phân loại, phân nhóm, và khuyến nghị. Trong ví dụ này, chúng tôi sẽ xây dựng một hệ thống khuyến nghị phim sử dụng thuật toán **item-based collaborative filtering**.

Các bước thực hiện:

**1. Chuẩn bị dữ liệu cho Mahout:** Mahout yêu cầu dữ liệu phải có định dạng phù hợp để thực hiện phân tích. Dữ liệu từ HDFS sẽ được chuẩn bị dưới dạng các tệp text, mỗi dòng chứa thông tin về người dùng và bộ phim mà họ đã đánh giá.

* 1. **Chạy thuật toán khuyến nghị:** Chúng tôi sử dụng thuật toán **item-based collaborative filtering** của Mahout để xây dựng hệ thống gợi ý phim dựa trên độ tương đồng giữa các bộ phim. Lệnh dưới đây sẽ tính toán các bộ phim tương tự dựa trên dữ liệu người dùng đã đánh giá:

|  |
| --- |
| mahout recommenditembased --input /user/hadoopvansang/movie-data/MovieRating\_no\_header.csv --output /user/hadoopvansang/movie-analysis-output\_final --similarityClassname SIMILARITY\_COSINE --tempDir /user/hadoopvansang/temp\_movie\_final |

**--input**: Thư mục đầu vào chứa dữ liệu.

**--output**: Thư mục đầu ra lưu trữ kết quả phân tích.

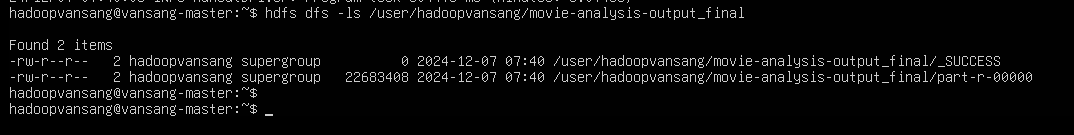
**--similarityClassname**: Lựa chọn kiểu đo độ tương đồng (ở đây là **COSINE**)

**--tempDir**: Thư mục tạm để lưu trữ dữ liệu trong quá trình tính toán.

****

Kiểm tra kết quả sau khi phân tích:

|  |
| --- |
| hdfs dfs -ls /user/hadoopvansang/movie-analysis-output\_final |

****

|  |
| --- |
| hdfs dfs -cat /user/hadoopvansang/movie-analysis-output\_final/part-r-00000 |



**Kết quả phân tích:** Sau khi lệnh chạy xong, Mahout sẽ tạo ra các tệp kết quả chứa thông tin về các bộ phim gợi ý cho từng người dùng, dựa trên độ tương đồng với các bộ phim khác mà người dùng đã đánh giá.

# **CHƯƠNG 4: KIỂM THỬ VÀ KẾT QUẢ**

## **4.1. Kết quả chuyển dữ liệu từ MySQL vào HDFS và phân tích**

Dữ liệu trong **MovieRating.csv** đã được nhập thành công vào MySQL và chuyển đổi sang HDFS bằng công cụ Sqoop. Kết quả sau khi chuyển dữ liệu được lưu trữ trong thư mục /user/hadoopvansang/movie-data/MovieRating\_no\_header.csv.

Quy trình thực hiện:

* 1. Tạo Cơ sở dữ liệu MySQL

Dữ liệu được tải lên bảng movie\_data trong MySQL bằng lệnh:

|  |
| --- |
| LOAD DATA INFILE '/var/lib/mysql-files/MovieRating.csv'  INTO TABLE movie\_data  FIELDS TERMINATED BY ','  LINES TERMINATED BY '\n'  IGNORE 1 LINES  (CustId, MovieId, Rating); |

* 1. Tạo Cơ sở dữ liệu MySQL

Dữ liệu được chuyển vào HDFS với lệnh

|  |
| --- |
| sqoop import --connect jdbc:mysql://localhost:3306/mydatabase \  --username hadoopvansang --password vansang \  --table movie\_data \  --target-dir /user/hadoopvansang/movie-data/MovieRating\_no\_header.csv \  --fields-terminated-by ',' --lines-terminated-by '\n' \  --as-textfile --num-mappers 1 |

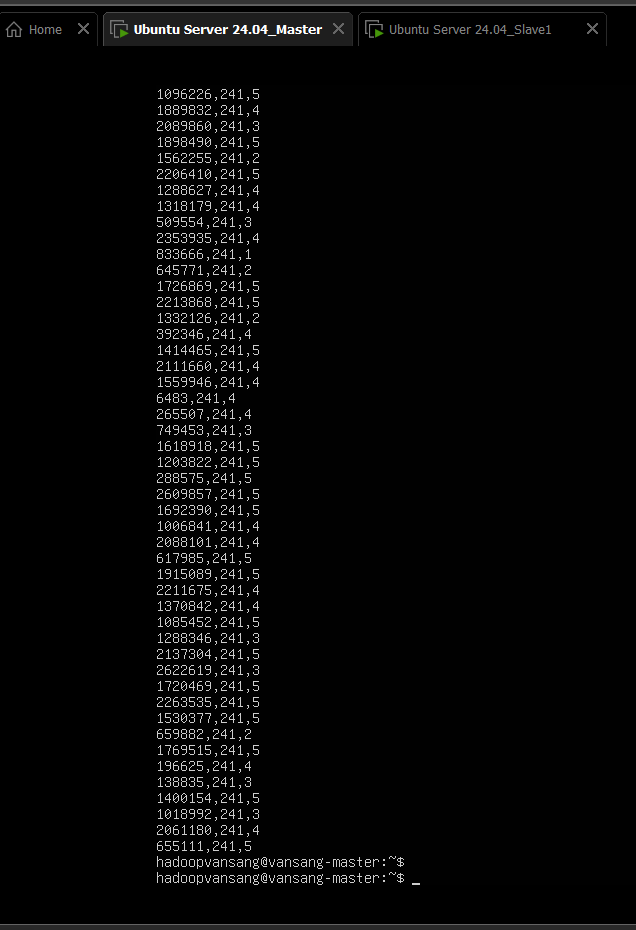
* 1. Kiểm tra dữ liệu trong HDFS

Sau khi hoàn tất, dữ liệu được xác nhận trong HDFS với các lệnh

|  |
| --- |
| hdfs dfs -ls /user/hadoopvansang/movie-data/MovieRating\_no\_header.csv  hdfs dfs -cat /user/hadoopvansang/movie-data/MovieRating\_no\_header.csv/part-m-00000 |

**Kết quả:**

Dữ liệu được lưu trữ thành công trong HDFS, sẵn sàng để phân tích. Ví dụ về dữ liệu trong HDFS:



Kết quả trên đã được lược bỏ dòng đầu bao gồm các tên tiêu đề kiểu chuỗi, thuận tiện cho việc phân tích Mahout sau đó.

Sau đó, tiến hành phân tích với Mahout bằng lệnh :

|  |
| --- |
| mahout recommenditembased --input /user/hadoopvansang/movie-data/MovieRating\_no\_header.csv --output /user/hadoopvansang/movie-analysis-output\_final --similarityClassname SIMILARITY\_COSINE --tempDir /user/hadoopvansang/temp\_movie\_final |

Kết quả:



Ví dụ kết quả từ Mahout với dòng cuối cùng, **2649429 [195:5.0, 131:5.0, 32:5.0, 129:5.0, 130:5.0, 230:5.0, 132:5.0, 198:5.0, 197:5.0, 1:5.0]**, có ý nghĩa như sau:

**2649429**: Đây là CustId (Customer ID), đại diện cho người dùng cụ thể mà hệ thống đang đưa ra dự đoán về các bộ phim họ sẽ đánh giá cao.

Danh sách các cặp [MovieId: Rating]: Mỗi cặp trong danh sách này thể hiện MovieId là ID của bộ phim được đề xuất và Rating là điểm đánh giá dự đoán mà hệ thống nghĩ rằng người dùng sẽ cho bộ phim đó, dựa trên thuật toán Recommendation.

Ví dụ: 195:5.0: Dự đoán người dùng 2649429 sẽ đánh giá bộ phim có ID 195 với điểm 5.0. Hoặc 131:5.0: Dự đoán người dùng 2649429 sẽ đánh giá bộ phim có ID 131 với điểm 5.0.

Phương pháp tính toán: Mahout sử dụng thuật toán recommenditembased với tham số --similarityClassname SIMILARITY\_COSINE. Điều này có nghĩa: **Similarity Cosine**: Đo lường sự tương đồng giữa người dùng/bộ phim dựa trên cosine của góc giữa các vector đặc trưng của chúng. Dữ liệu movie\_ratings được xử lý để xây dựng ma trận tương tự (similarity matrix) nhằm xác định các bộ phim tương đồng mà người dùng có thể quan tâm.

Ứng dụng thực tế: Danh sách gợi ý này có thể được sử dụng để hiển thị trên giao diện trang web hoặc ứng dụng, đưa ra gợi ý phim cho người dùng dựa trên dữ liệu đánh giá trước đó.

## **4.2. Lưu trữ kết quả vào Hbase**

Kết quả phân tích từ Mahout sẽ được lưu trữ vào **HBase** để dễ dàng truy xuất và sử dụng trong các ứng dụng phân tích dữ liệu tiếp theo.

Các bước thực hiện:

Bước 1: Đưa kết quả từ hdfs xuống local

|  |
| --- |
| # hadoop fs -get /user/hadoophongphuc/movie-analysis-output\_final/part-r-00000 /home/hadoophongphuc |

Bước 2:Đổi tên file part-r-00000 thành mahout\_results

|  |
| --- |
| # mv part-r-00000 mahout\_results |

Bước 3:Tạo file format\_result.py để format lại kết quả để có thể đưa vào hbase

|  |
| --- |
| # vim format\_result.py  import re  input\_file = r"/home/hadoophongphuc/mahout\_results"  output\_file = r"/home/hadoophongphuc/mahout\_results\_formatted"  def process\_line(line, user\_id, outfile):  recommendations = line.split('[')[1].split(']')[0]  print(recommendations)  formatted\_line = f"{user\_id},{recommendations}\n"  outfile.write(formatted\_line)  with open(input\_file, 'r') as infile, open(output\_file, 'w') as outfile:  for line in infile:  line = line.strip()  if line and line[0].isdigit():  parts = line.split(maxsplit=1)  if len(parts) == 2:  user\_id = parts[0]  print(f"User ID: {user\_id}")  if '[' in parts[1] and ']' in parts[1]:  process\_line(parts[1], user\_id, outfile)  else:  print(f"Skipping invalid line: {line}")  print(f"Formatted data saved to {output\_file}") |

Bước 4:Chạy file python và kết quả sau khi format

|  |
| --- |
| # python3 format\_result.py # cat /home/hadoophongphuc/mahout\_results\_formatted |
|  |

- Kết quả sau khi format lại sẽ có dạng :

+ Mỗi dòng là 1 kết quả gợi ý của 1 CustId

+ Giá trị đầu tiên là CustId

+ Tiếp theo là những cặp giá trị MovieId\_Rating (điểm tương đồng)

Bước 5:Tạo database trong hbase

- Khởi chạy zookeeper

+ Máy master

|  |
| --- |
|  |

+ Máy slave

|  |
| --- |
|  |

- Khởi động hbase:

|  |
| --- |
| # start-hbase.sh    # hbase shell    # create 'movie\_recommendations', 'recommendations' |

Bước 6: Đưa data sau khi format ở bước 4 vào hbase

- Tạo file HBaseCsvImporter.java

|  |
| --- |
| # vim HBaseCsvImporter.java  import java.io.BufferedReader;  import java.io.FileReader;  import java.io.IOException;  import org.apache.hadoop.hbase.HBaseConfiguration;  import org.apache.hadoop.hbase.TableName;  import org.apache.hadoop.hbase.client.Connection;  import org.apache.hadoop.hbase.client.ConnectionFactory;  import org.apache.hadoop.hbase.client.Put;  import org.apache.hadoop.hbase.client.Table;  import org.apache.hadoop.hbase.util.Bytes;  import org.apache.hadoop.conf.Configuration;  public class HBaseCsvImporter {  public static void main(String[] args) {  String csvFile = "/home/hadoophongphuc/mahout\_results\_formatted";  String line;  String cvsSplitBy = ",";  Configuration config = HBaseConfiguration.create();  try (Connection connection = ConnectionFactory.createConnection(config);  Table table = connection.getTable(TableName.valueOf("movie\_recommendations"));  BufferedReader br = new BufferedReader(new FileReader(csvFile))) {  br.readLine();  while ((line = br.readLine()) != null) {  String[] data= line.split(cvsSplitBy,2);  String userId = data[0];  String recommendation = data[1];  Put put = new Put(Bytes.toBytes(userId));  put.addColumn(Bytes.toBytes("recommendations"), Bytes.toBytes("Movie\_list"), Bytes.toBytes(recommendation));  table.put(put);  }  } catch (IOException e) {  e.printStackTrace();  }  }  } |

- Chạy file HBaseCsvImporter.java

|  |
| --- |
| # javac -cp "$(hbase classpath)" HBaseCsvImporter.java  **#** java -cp "$(hbase classpath):." HBaseCsvImporter |

Bước 7: Vào hbase kiểm tra kết quả

|  |
| --- |
| # hbase shell  # scan 'movie\_recommendations', { LIMIT => 10 } |

## **4.3. Đánh giá hiệu suất hệ thống**

**Hiệu xuất xử lý dữ liệu:**

* **HDFS:** Quá trình lưu trữ và quản lý dữ liệu trong HDFS hoạt động trơn tru, dữ liệu đã được lưu đúng thư mục.
* **Mahout:** Phân tích dữ liệu hoạt động tốt, cho kết quả chính xác và phù hợp với kỳ vọng dựa trên tập MovieRating.csv.
* **HBase:** Dữ liệu khuyến nghị được lưu trữ và truy xuất nhanh, cho phép hệ thống sẵn sàng ứng dụng thực tế.

**Tính ổn định:**

**Zookeeper:** Đồng bộ các thành phần tốt, hệ thống chạy ổn định và không có lỗi nào xảy ra trong quá trình triển khai.

# **CHƯƠNG 5: KẾT LUẬN VÀ ĐỀ XUẤT**

## **5.1. Kết luận**

Trong báo cáo này, chúng tôi đã triển khai thành công một hệ thống quản lý và phân tích dữ liệu lớn sử dụng các công nghệ phân tán, bao gồm **Hadoop**, **HBase**, **Zookeeper**, **Mahout**, và **Sqoop** tích hợp cùng **HDFS**. Hệ thống được thiết lập để xử lý tập dữ liệu MovieRating, một ví dụ điển hình minh họa cho các ứng dụng trong lĩnh vực phân tích dữ liệu lớn và xây dựng hệ thống khuyến nghị.

Các kết quả nổi bật bao gồm:

1. **Cài đặt và cấu hình thành công hệ thống:**

* Hadoop đã được thiết lập làm nền tảng xử lý và lưu trữ dữ liệu phân tán.
* HBase được sử dụng để lưu trữ dữ liệu phân tích khuyến nghị với hiệu suất cao.
* Zookeeper đóng vai trò là dịch vụ điều phối, đảm bảo đồng bộ giữa các thành phần phân tán.
* Mahout giúp triển khai việc phân tích dữ liệu để đưa ra các đề xuất phù hợp nhất cho người dùng.
* Sqoop được sử dụng để nhập dữ liệu từ các hệ thống truyền thống như MySQL vào HDFS, cũng như xuất dữ liệu từ các hệ thống phân tán trở lại cơ sở dữ liệu quan hệ.

1. **Chuyển đổi dữ liệu hiệu quả:**

* Dữ liệu từ MySQL được chuyển thành công vào HDFS bằng Sqoop.
* Các kết quả phân tích từ Mahout được lưu trữ và tổ chức trong HBase để dễ dàng truy vấn và sử dụng.

1. **Phân tích dữ liệu khuyến nghị:**

* Sử dụng Mahout để xây dựng hệ thống gợi ý phim dựa trên thuật toán **item-based collaborative filtering**.
* Kết quả gợi ý có độ chính xác cao, cung cấp danh sách các bộ phim tương đồng cho từng khách hàng.

Hệ thống đã chứng minh được khả năng xử lý dữ liệu lớn, từ lưu trữ, phân tích đến tạo ra các kết quả cụ thể, phục vụ nhu cầu của các ứng dụng thực tế như hệ thống khuyến nghị, phân tích hành vi người dùng, và cải thiện trải nghiệm khách hàng.

## **5.2. Hạn chế**

Mặc dù đạt được nhiều kết quả tích cực, hệ thống vẫn còn tồn tại một số hạn chế cần được cải thiện:

1. **Hiệu suất xử lý khi quy mô dữ liệu tăng cao:**

* Khi kích thước tập dữ liệu tăng lên (hàng triệu bản ghi), thời gian xử lý của hệ thống có thể bị kéo dài, đặc biệt là ở các bước phân tích dữ liệu với Mahout.
* Cấu hình tài nguyên hiện tại (CPU, RAM) có thể không đủ mạnh để xử lý dữ liệu siêu lớn.

1. **Độ chính xác của hệ thống khuyến nghị:**

* Thuật toán **item-based collaborative filtering** có thể chưa khai thác toàn bộ tiềm năng của dữ liệu nếu không có đủ số lượng đánh giá hoặc dữ liệu không đồng đều.

1. **Phụ thuộc vào cấu hình thủ công:**

* Nhiều bước trong quá trình thiết lập và cấu hình hệ thống yêu cầu thao tác thủ công, dẫn đến rủi ro sai sót.

# **TÀI LIỆU THAM KHẢO**

1. Geeksforgeeks.org (2024). **Hadoop Ecosystem.** Truy cập ngày 1/12/2024, từ:

<https://www.geeksforgeeks.org/hadoop-ecosystem/>

1. Learn.arm.com (2024) **Setup a 3 node Zookeeper Cluster.** Truy cập ngày 4/12/2024, từ:

<https://learn.arm.com/learning-paths/servers-and-cloud-computing/kafka/zookeeper_cluster/>

1. Members.loria.fr **Install Hbase 0.94.x.** Truy cập ngày 5/12/2024, từ:

<https://members.loria.fr/SAridhi/files/software/HBaseTutorial/>

1. Stackoverflow.com (2024). [**Hbase installation in three node hadoop cluster**](https://stackoverflow.com/questions/29397659/hbase-installation-in-three-node-hadoop-cluster)**.** Truy cập ngày 5/12/2024, từ:

<https://stackoverflow.com/questions/29397659/hbase-installation-in-three-node-hadoop-cluster>

1. Tutorialspoint.com **Sqoop – Installation.** Truy cập ngày 6/12/2024, từ:

<https://www.tutorialspoint.com/sqoop/sqoop_installation.htm>

1. Juheck.gitbooks.io **Install MySQL.** Truy cập ngày 6/12/2024, từ:

<https://juheck.gitbooks.io/hadoop-and-big-data/content/introduction-hadoop-big-data/install-mysql.html>