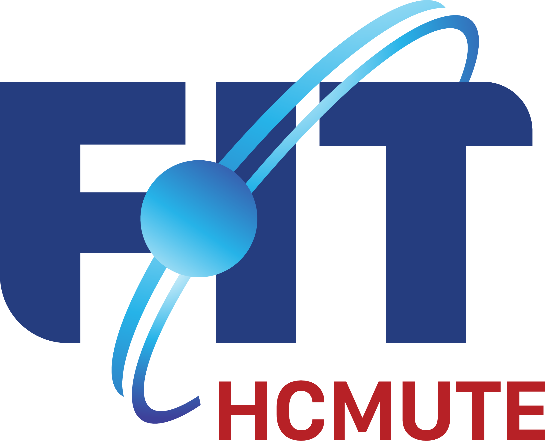
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----🙞🙞🙞🙞🙞-----**



**BÁO CÁO CUỐI KỲ**

**BIG DATA ESSENTIALS (NHẬP MÔN DỮ LIỆU LỚN)**

**ĐỀ TÀI:**

**BUILD A SIMPLE MOVIE RECOMMENDATION SYSTEM**

**USING USER RATINGS DATA**

**INITIALLY STORED IN MYSQL DATABASE**

**GVHD:** ThS. Lê Thị Minh Châu

|  |  |
| --- | --- |
| Nguyễn Thị Ngọc Hân | 22133017 |
| Nguyễn Hoàng | 22133020 |
| Võ Triệu Phúc | 22133043 |
| Nguyễn Thị Hồng Thơ | 22151305 |

**Sinh viên thực hiện:**

*Thành phố Hồ Chí Minh, tháng 12, năm 2024*

|  |  |
| --- | --- |
| BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc** |

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**Họ và tên sinh viên thực hiện:**

Nguyễn Thị Ngọc Hân - 22133020

Nguyễn Hoàng – 22133020

Võ Triệu Phúc – 22133043

Nguyễn Thị Hồng Thơ - 22151305

**Chuyên ngành:** Kỹ thuật dữ liệu (Data Engineering)

**Đề tài: Build a simple movie recommendation system using user ratings data initially in MySQL database.**

**Môn học:** Big Data Essentials – Nhập môn dữ liệu lớn

**Nhận xét:**

........................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

......................................................................................................................................................................................................................................................................................

...........................................................................................................................................

......................................................................................................................................................................................................................................................................................

TP. HCM, ngày tháng 12 năm 2024

Giảng viên hướng dẫn

(Họ tên và chữ ký)

MỤC LỤC

[DANH MỤC HÌNH ẢNH 5](#_Toc184673815)

[MỞ ĐẦU 1](#_Toc184673816)

[1. Lý do chọn đề tài 1](#_Toc184673817)

[2. Giới thiệu đề tài 1](#_Toc184673818)

[3. Mục tiêu nghiên cứu 2](#_Toc184673819)

[CHƯƠNG 1: HỆ SINH THÁI HADOOP 3](#_Toc184673820)

[1. Khái niệm 3](#_Toc184673821)

[2. Hệ sinh thái Hadoop trong hệ thống đề xuất phim (Simple Movie Recommendation System) 4](#_Toc184673822)

[CHƯƠNG 2: TỔNG QUAN VỀ MYSQL, SQOOP, HBASE, MAHOUT, ZOOKEEPER VÀ CÁCH CÀI ĐẶT 6](#_Toc184673823)

[1. Mysql 6](#_Toc184673824)

[1.1. Cài đặt mysql 6](#_Toc184673825)

[1.2. Thiết lập password cho user root 6](#_Toc184673826)

[2. Sqoop 9](#_Toc184673827)

[2.1. Tổng quan 9](#_Toc184673828)

[2.2. Quy trình hoạt động cơ bản của Sqoop 9](#_Toc184673829)

[2.3. Các thành phần chính của Sqoop 10](#_Toc184673830)

[2.4. Cài đặt Sqoop 10](#_Toc184673831)

[2.5. Cấu hình sqoop 11](#_Toc184673832)

[2. HBase 12](#_Toc184673833)

[2.1. Tổng quan 12](#_Toc184673834)

[2.2. Cấu trúc dữ liệu trong HBase 13](#_Toc184673835)

[2.3. Kiến trúc 14](#_Toc184673836)

[2.4. Cài đặt HBase ở chế độ Pseudo-Distributed 15](#_Toc184673837)

[2.5. Cài đặt HBase ở chế độ Fully Distributed 23](#_Toc184673838)

[3. Mahout 28](#_Toc184673839)

[3.1. Tổng quan 28](#_Toc184673840)

[3.2. Kiến trúc 30](#_Toc184673841)

[3.3. Các thuật toán trong Mahout 30](#_Toc184673842)

[3.4. Cài đặt Mahout 31](#_Toc184673843)

[4. Zookeeper 33](#_Toc184673844)

[4.1. Tổng quan 33](#_Toc184673845)

[4.2. Kiến trúc 39](#_Toc184673846)

[4.3. Cài đặt Zookeeper 40](#_Toc184673847)

[4.4. Cấu hình ZooKeeper chế độ standalone 42](#_Toc184673848)

[4.5. Thiết lập một cụm (cluster) gồm 3 máy chủ ZooKeeper 45](#_Toc184673849)

[CHƯƠNG 3: ỨNG DỤNG XÂY DỰNG HỆ THỐNG ĐỀ XUẤT PHIM 50](#_Toc184673850)

[1. Recommendations với Mahout và ứng dụng trong hệ thống đề xuất phim 50](#_Toc184673851)

[1.1. Giới thiệu về hệ thống đề xuất 50](#_Toc184673852)

[1.2. Recommendations trong Apache Mahout 53](#_Toc184673853)

[2. Tổng quan quy trình của hệ thống đề xuất phim 57](#_Toc184673854)

[3. Xây dựng hệ thống đề xuất phim với tập dữ liệu ratings.csv 58](#_Toc184673855)

[3.1. Đưa tập dữ liệu ratings.csv vào home của hadoop 58](#_Toc184673856)

[3.2. Chuyển dữ liệu ratings.csv vào MySQL 58](#_Toc184673857)

[3.3. Imort dữ liệu từ MySQL sang HDFS – Tạo input dữ liệu cho mahout 60](#_Toc184673858)

[3.4. Sử dụng Mahout và MapReduce để xây dựng mô hình gợi ý 61](#_Toc184673859)

[3.5. Quá trình MapReduce 64](#_Toc184673860)

[3.6. Chuyển kết quả vào tập dữ liệu trong HBase 65](#_Toc184673861)

[3.7. Zookeeper trong Recommendation Systems 70](#_Toc184673862)

[KẾT LUẬN 71](#_Toc184673863)

[TÀI LIỆU THAM KHẢO 72](#_Toc184673864)

# DANH MỤC HÌNH ẢNH

[Hình 1. Hadoop Ecosystem 4](#_Toc184673473)

[Hình 2. Minh họa dữ liệu trong một bảng HBase 13](#_Toc184673474)

[Hình 3. Kiến trúc của Apache HBase 14](#_Toc184673475)

[Hình 4. Lịch sử hình thành và phát triển Mahout 29](#_Toc184673476)

[Hình 5. Các thiết kế cao cấp của một Mahout 30](#_Toc184673477)

[Hình 6. Kiến trúc của Mahout trong Hadoop 30](#_Toc184673478)

[Hình 7. Cấu trúc phân cấp của Zookeeper 35](#_Toc184673479)

[Hình 8. Kiến trúc của Zookeeper 39](#_Toc184673480)

[Hình 9. Mô tả Content-based filtering 50](#_Toc184673481)

[Hình 10. Mô tả User-Based Collaborative Filtering 51](#_Toc184673482)

[Hình 11. Mô tả Item-Based Collaborative Filtering 52](#_Toc184673483)

[Hình 12. Recommendation Architecture 53](#_Toc184673484)

[Hình 13. Sự khác biệt giữa độ tương đồng khoảng cách Euclidean và độ tương đồng Cosine 54](#_Toc184673485)

[Hình 14. Hai phương pháp đo lường sự tương đồng: hệ số tương quan Pearson và khoảng cách Euclidean 55](#_Toc184673486)

[Hình 15. Thuật toán K-Nearest Neighbors 55](#_Toc184673487)

[Hình 16. Cơ chế Recommendation của Mahout 56](#_Toc184673488)

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Hiện nay, hệ thống đề xuất phim đã trở thành một phần không thể thiếu trong các nền tảng trực tuyến như Netflix, YouTube,... giúp người dùng tìm kiếm và trải nghiệm những bộ phim phù hợp với sở thích cá nhân. Việc phát triển một hệ thống đề xuất hiệu quả không chỉ mang lại trải nghiệm người dùng tốt hơn mà còn tăng cường khả năng giữ chân khách hàng cho các dịch vụ này. Các nền tảng phát trực tuyến hiện nay xử lý một lượng lớn dữ liệu từ người dùng, đặc biệt là các đánh giá và hành vi người dùng. Sử dụng các công cụ xử lý dữ liệu lớn trong hệ sinh thái Hadoop, chẳng hạn như HDFS (Hadoop Distributed File System), MapReduce, Mahout,... để lưu trữ và xử lý dữ liệu, đảm bảo khả năng mở rộng và tính linh hoạt trong việc xử lý dữ liệu phân tán.

Với sự phát triển mạnh mẽ của các dịch vụ phát trực tuyến, việc xây dựng hệ thống đề xuất phim không chỉ mang lại giá trị học thuật mà còn có tính ứng dụng cao, giúp cải thiện trải nghiệm người dùng và tăng trưởng doanh thu từ các dịch vụ cá nhân hóa. Do đó, đề tài ***"Build a simple movie recommendation system using user ratings data initially stored in MySQL database - Xây dựng hệ thống đề xuất phim đơn giản sử dụng dữ liệu đánh giá của người dùng được lưu trữ trong MySQL"*** không chỉ có ý nghĩa quan trọng trong bối cảnh hiện nay, mà còn đóng góp vào việc phát triển các hệ thống thông minh và phân tích dữ liệu quy mô lớn, giải quyết các thách thức trong việc xử lý và phân tích dữ liệu người dùng.

## 2. Giới thiệu đề tài

Hệ thống đề xuất sử dụng thuật toán học máy để phân tích mối quan hệ giữa các bộ phim và đánh giá của người dùng, từ đó đưa ra các gợi ý phù hợp.

Để xử lý lượng dữ liệu lớn và tối ưu hóa quá trình phân tích, hệ thống sử dụng các công cụ chính bao gồm MySQL giúp tổ chức dữ liệu một cách có hệ thống, dễ dàng truy vấn và phân tích, Sqoop được sử dụng để nhập dữ liệu từ MySQL vào HDFS, giúp quá trình di chuyển dữ liệu giữa cơ sở dữ liệu quan hệ và hệ thống phân tán Hadoop trở nên dễ dàng và nhanh chóng. Sqoop hỗ trợ tự động hóa quá trình nhập dữ liệu từ MySQL và giúp các công cụ Hadoop có thể truy cập và xử lý dữ liệu người dùng một cách hiệu quả. HDFS (Hadoop Distributed File System) để lưu trữ và xử lý dữ liệu phân tán, MapReduce cho các tác vụ tính toán phức tạp, và Mahout để thực hiện các thuật toán gợi ý. YARN (Yet Another Resource Negotiator) giúp quản lý tài nguyên và tối ưu hóa hiệu suất trong môi trường phân tán, trong khi Zookeeper đảm bảo tính đồng bộ và ổn định giữa các nút trong hệ thống.

## 3. Mục tiêu nghiên cứu

**Xây dựng mô hình đề xuất phim**: Phát triển hệ thống đề xuất phim dựa trên các đánh giá của người dùng, sử dụng thuật toán học máy và dữ liệu phân tán.

**Đảm bảo tính nhất quán và hiệu quả trong toàn bộ hệ thống phân tán:** Quản lý dữ liệu người dùng trong cơ sở dữ liệu MySQL, sau đó sử dụng Sqoop để nhập dữ liệu từ MySQL vào HDFS để phục vụ cho quá trình xử lý và phân tích.

**Tối ưu hóa chất lượng gợi ý:** Áp dụng Apache Mahout để tính toán độ tương đồng giữa các bộ phim, cung cấp gợi ý chính xác và phù hợp với sở thích người dùng.

**Lưu trữ và xử lý dữ liệu lớn:** Sử dụng HDFS để lưu trữ dữ liệu đánh giá phim và HBase để lưu trữ kết quả gợi ý, đảm bảo khả năng mở rộng và truy cập hiệu quả.

**Quản lý tài nguyên:** Sử dụng YARN để phân phối và quản lý tài nguyên trong xử lý dữ liệu lớn, đảm bảo tác vụ thực hiện nhanh chóng và tối ưu.

**Đảm bảo tính ổn định và đồng bộ**: Triển khai Zookeeper để quản lý sự đồng bộ giữa các nút, đảm bảo tính toán và lưu trữ diễn ra ổn định.

# CHƯƠNG 1: HỆ SINH THÁI HADOOP

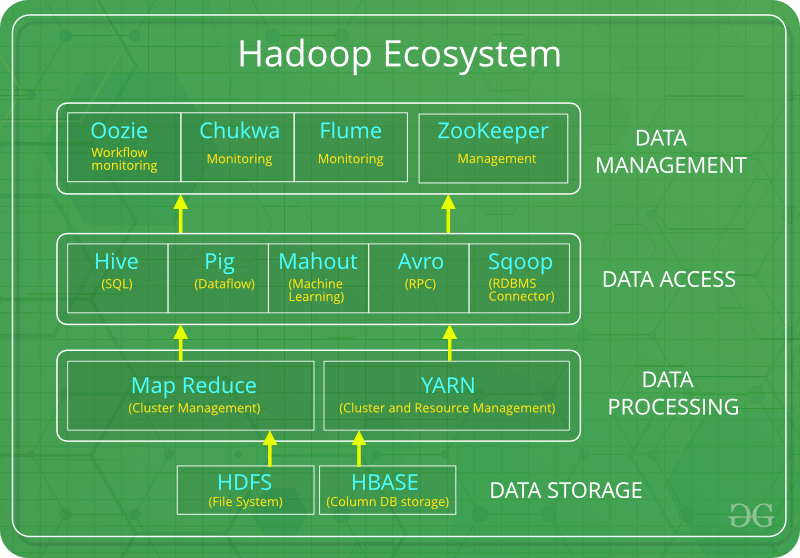
## 1. Khái niệm

**Hadoop Ecosystem** là một nền tảng hoặc một bộ công cụ cung cấp nhiều dịch vụ để giải quyết các vấn đề về dữ liệu lớn. Có bốn thành phần chính của Hadoop, đó là HDFS, MapReduce, YARN và Hadoop Common Utilities.

**Các thành phần tạo nên hệ sinh thái Hadoop:**

* **HDFS**: Hệ thống tệp phân tán Hadoop
* **YARN**: Bộ điều phối tài nguyên (Yet Another Resource Negotiator)
* **MapReduce**: Xử lý dữ liệu dựa trên lập trình
* **Spark:** Xử lý dữ liệu trong bộ nhớ
* **PIG, HIVE:** Dịch vụ xử lý dữ liệu dựa trên truy vấn
* **HBase**: Cơ sở dữ liệu NoSQL
* **Mahout, Spark MLLib**: Thư viện thuật toán học máy
* **Solr, Lucene**: Tìm kiếm và lập chỉ mục
* **Zookeeper:** Quản lý cụm máy chủ
* **Oozie:** Lập lịch công việc

Hệ sinh thái Hadoop được sắp xếp thành bốn lớp khác nhau, bao gồm: Lưu trữ dữ liệu (Data Storage), Xử lý dữ liệu (Data Processing), Truy cập dữ liệu (Data Access), và Quản lý dữ liệu (Data Management). Mỗi lớp bao gồm các thành phần khác nhau trong hệ sinh thái Hadoop.



Hình 1. Hadoop Ecosystem

## 2. Hệ sinh thái Hadoop trong hệ thống đề xuất phim (Simple Movie Recommendation System)

Trong hệ thống đề xuất phim **(simple movie recommendation system using user ratings data initially)**, nhóm sử dụng các công cụ ở các tầng:

**1. Data Storage Layer**

**HDFS**: HDFS lưu trữ tập dữ liệu lớn từ file **ratings.csv,** nơi chứa thông tin về các đánh giá của người dùng đối với các bộ phim. Dữ liệu sẽ được chia thành các khối lớn (64MB) và lưu trữ trong môi trường phân tán, giúp tối ưu cho xử lý dữ liệu hàng loạt với MapReduce.

**HBase**: Sau khi Mahout tạo ra các gợi ý, kết quả sẽ được lưu trữ trong HBase. Ví dụ, mỗi người dùng sẽ có một bản ghi với các gợi ý phim lưu trữ trong các cột của bảng HBase (recommendations). Điều này cho phép việc truy vấn gợi ý phim của từng người dùng một cách nhanh chóng và hiệu quả.

**2. Data Processing Layer**

Các thành phần xử lý dữ liệu trong hệ sinh thái Hadoop bao gồm **MapReduce** và **YARN**.

**MapReduce:** MapReduce được sử dụng để xử lý các đánh giá phim (ratings). Cụ thể, các đánh giá người dùng sẽ được xử lý trong các bước MapReduce để tạo ra các mô hình gợi ý, như tính toán sự tương đồng giữa các mục (items) và người dùng. Mahout, một thư viện học máy, sử dụng MapReduce để tính toán các thuật toán gợi ý (ví dụ: Collaborative Filtering) với độ chính xác cao trên tập dữ liệu lớn.

**YARN:** YARN giúp tối ưu hóa việc chạy các công việc MapReduce và các công việc không phải MapReduce, từ đó cải thiện hiệu suất xử lý và cho phép hệ thống chạy đồng thời các tác vụ gợi ý phim, xử lý dữ liệu đánh giá và các công việc khác trong môi trường phân tán.

**3. Data Access Layer**

**Sqoop:** Sqoop sẽ được sử dụng để nhập dữ liệu từ MySQL vào HDFS. Các đánh giá phim và thông tin người dùng được chuyển từ MySQL sang HDFS, sau đó sử dụng Mahout để tạo các gợi ý phim dựa trên dữ liệu đánh giá.

**Mahout**: Mahout cung cấp các thuật toán phân cụm, phân loại và gợi ý, giúp nhóm phim tương tự nhau hoặc gợi ý các phim mà người dùng có thể thích. Cụ thể, Mahout sẽ được sử dụng để thực hiện các thuật toán gợi ý (ví dụ: **Item-based Collaborative Filtering**, **User-based Collaborative Filtering**). Mahout sẽ sử dụng dữ liệu đánh giá phim từ HDFS và tạo ra các gợi ý phim cho từng người dùng dựa trên các mô hình học máy. Sau khi tạo ra các gợi ý, Mahout lưu trữ kết quả trong HBase để dễ dàng truy cập và sử dụng trong các ứng dụng khách.

**4. Data Management Layer**

**Zookeeper:** Zookeeper không trực tiếp tham gia vào quá trình tính toán gợi ý, nhưng nó có thể được sử dụng để đồng bộ hóa và quản lý các công việc giữa các nút trong hệ thống Hadoop, ví dụ, trong trường hợp các tác vụ gợi ý được chạy song song hoặc trong môi trường nhiều nút, Zookeeper sẽ giúp duy trì sự nhất quán và điều phối các tài nguyên và công việc của hệ thống.

# CHƯƠNG 2: TỔNG QUAN VỀ MYSQL, SQOOP, HBASE, MAHOUT, ZOOKEEPER VÀ CÁCH CÀI ĐẶT

## 1. Mysql

### 1.1. Cài đặt mysql

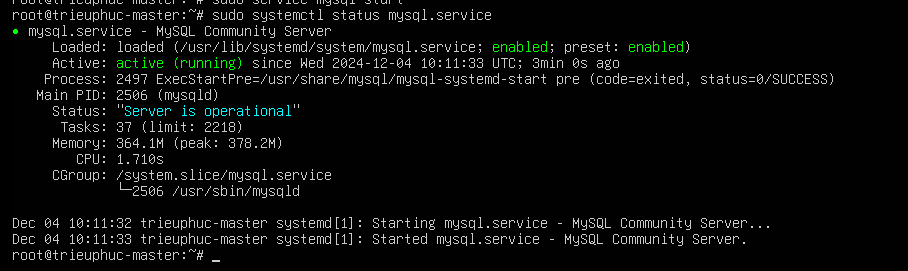
**Chạy các lệnh sau**

$ sudo apt update

$ sudo apt-get install mysql-server

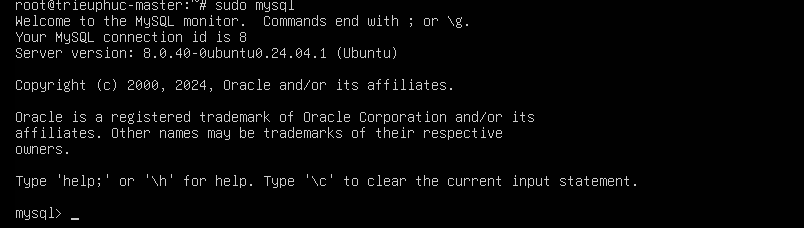
$ sudo service mysql start

$ sudo systemctl status mysql.service



Nếu Active: crash, stop => sudo systemctl start mysql.service

$ sudo mysql

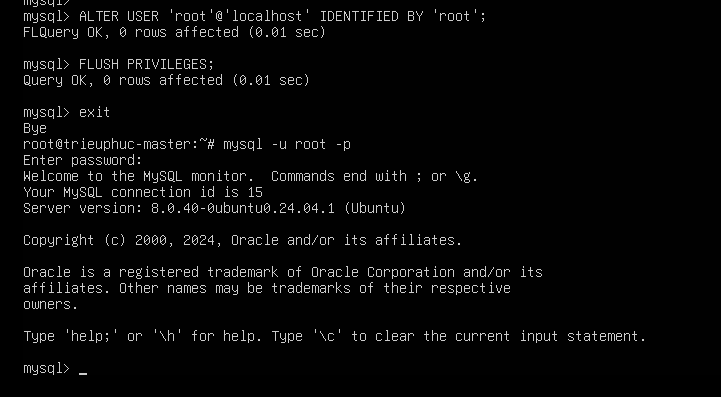


### 1.2. Thiết lập password cho user root

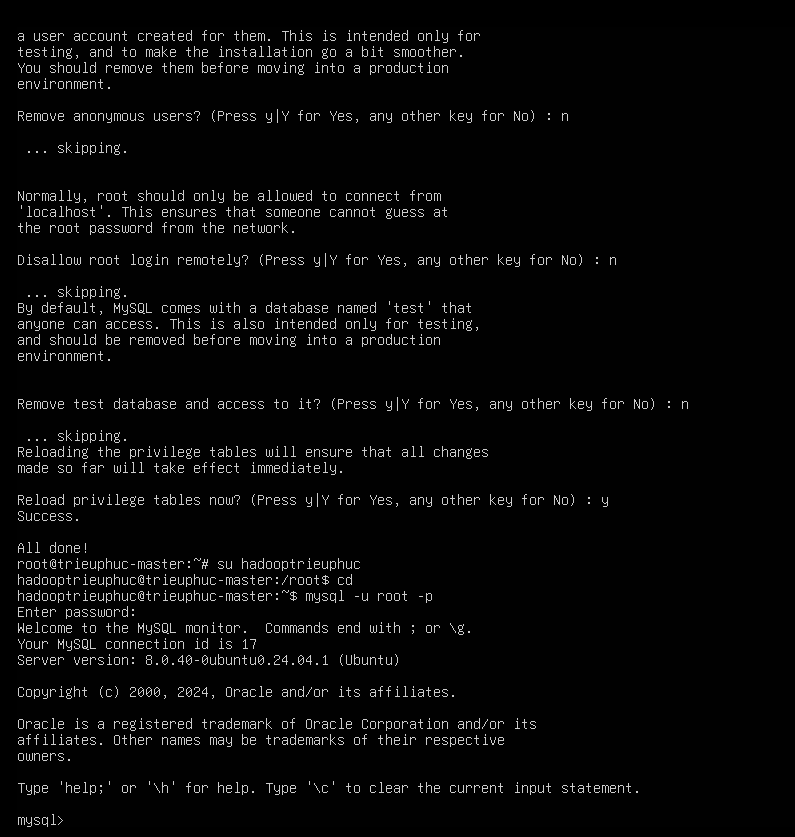
$ su root

$ sudo mysql

> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password by ' root’;



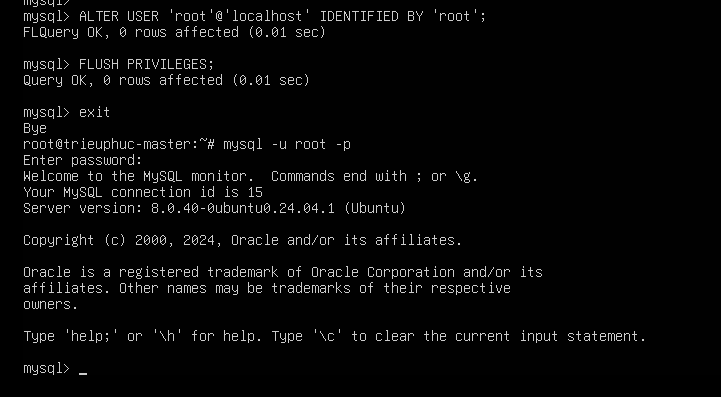
$ sudo mysql\_secure\_installation



Đăng nhập lại:

$ mysql -u root -p

Sau đó nhập password: root



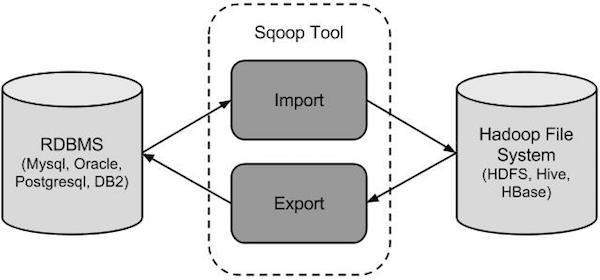
Thoát => mysql> exit

## 2. Sqoop

### 2.1. Tổng quan

Apache **Sqoop** (SQL-to-Hadoop) là công cụ hỗ trợ di chuyển dữ liệu giữa các hệ thống quan hệ (RDBMS) (như MySQL, PostgreSQL) và Hadoop (HDFS, Hive, HBase, v.v.).

### 2.2. Quy trình hoạt động cơ bản của Sqoop



* **Kết nối tới hệ thống RDBMS:**
* Sqoop sử dụng JDBC (Java Database Connectivity) để kết nối đến cơ sở dữ liệu quan hệ như MySQL, PostgreSQL, Oracle, SQL Server.
* Người dùng cung cấp thông tin kết nối như: username, password, và URL JDBC.
* **Trích xuất dữ liệu từ RDBMS:**
* Sqoop thực thi lệnh SQL SELECT trên bảng hoặc truy vấn được chỉ định.
* Dữ liệu được chia thành các phần (splits) để hỗ trợ nhập/xuất song song, tận dụng khả năng phân tán của Hadoop.
* **Nhập dữ liệu vào Hadoop:**
* **HDFS (Hadoop Distributed File System):** Sqoop chuyển dữ liệu từ RDBMS lên HDFS dưới dạng file CSV, text, hoặc file Avro/Parquet.
* **Hive:** Sqoop chuyển dữ liệu từ RDBMS vào bảng trong Hive để phân tích bằng SQL-like queries.
* **HBase:** Sqoop chuyển dữ liệu từ RDBMS trực tiếp vào bảng HBase để lưu trữ theo cột.
* **Xử lý dữ liệu (tùy chọn):** Ta có thể áp dụng các tham số để cấu hình xử lý dữ liệu, ví dụ:
* Ánh xạ kiểu dữ liệu.
* Lọc các cột/record không cần thiết.
* Định nghĩa cấu trúc file đầu ra.
* **Xuất dữ liệu ngược lại (Export):** Sqoop cũng hỗ trợ xuất dữ liệu từ HDFS hoặc Hive quay trở lại RDBMS.

### 2.3. Các thành phần chính của Sqoop

* **Connectors:** Sqoop sử dụng JDBC connectors để giao tiếp với các cơ sở dữ liệu như MySQL, PostgreSQL, SQL Server, Oracle, v.v.
* **MapReduce Framework**: Sqoop sử dụng MapReduce để phân tán và xử lý dữ liệu khi chuyển giữa RDBMS và Hadoop.
* **Data Import:** Dữ liệu từ RDBMS được chia nhỏ và nạp lên Hadoop thông qua các Map tasks.
* **Data Export:** Dữ liệu từ Hadoop được chia nhỏ và tải về RDBMS thông qua các Map tasks.

### 2.4. Cài đặt Sqoop

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

Hoặc tải rồi đưa vào sharedFolder



* **Giải nén file tải về**:

$ tar -xvzf sqoop-1.4.7.bin\_\_hadoop-2.6.0.tar.gz

$ sudo mv sqoop-1.4.7.bin\_\_hadoop-2.6.0 sqoop

* **Cấu hình biến môi trường**:

Mở file cấu hình ~/.bashrc:

$ vim ~/.bashrc

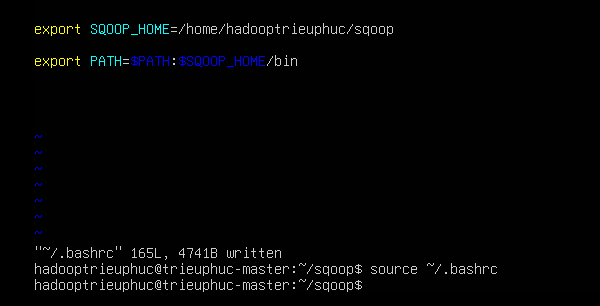
Thêm các dòng sau vào cuối file:

export SQOOP\_HOME=/home/hadooptrieuphuc/sqoop

export PATH=$PATH:$SQOOP\_HOME/bin

Sau đó, tải lại biến môi trường:

$ source ~/.bashrc



* **Kiểm tra lại Sqoop**

$ sqoop version

### 2.5. Cấu hình sqoop

$ cd home/hadoopotrieuphuc/conf

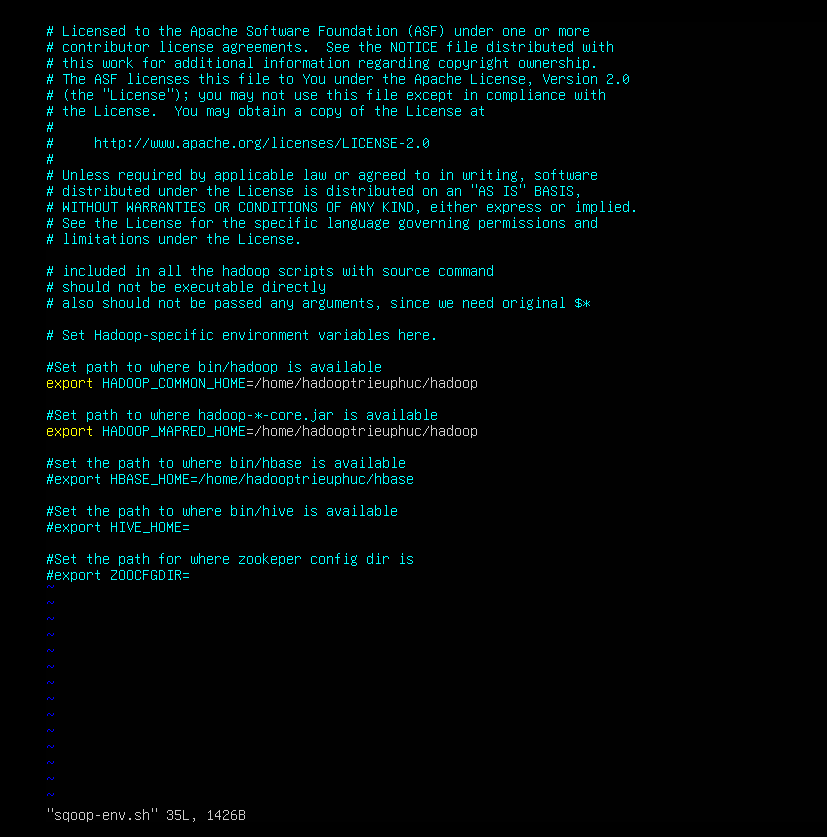
$ mv sqoop-env-template.sh sqoop-env.sh

$ vi sqoop-env.sh

**Edit các dòng sau**

export HADOOP\_COMMON\_HOME=/home/hadooptrieuphuc/hadoop

export HADOOP\_MAPRED\_HOME=/home/hadooptrieuphuc/hadoop

****

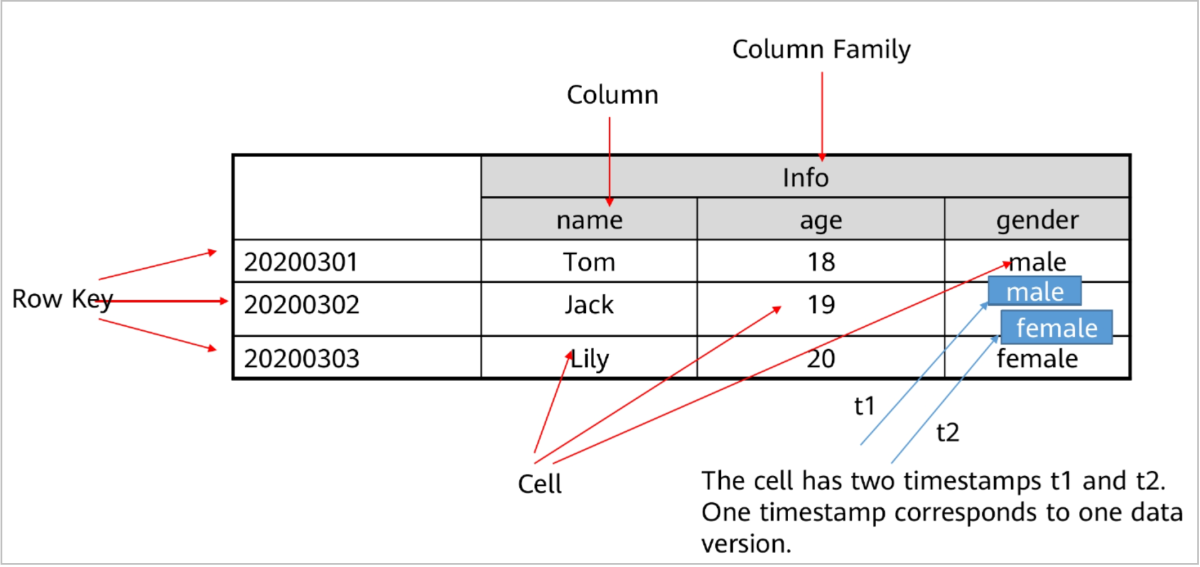
## 2. HBase

### 2.1. Tổng quan

HBase (Hadoop Database) là một **hệ quản trị cơ sở dữ liệu phân tán** (Distributed Database Management System) mã nguồn mở, thuộc **Hadoop Ecosystem của Apache.** HBase được thiết kế để lưu trữ và quản lý **dữ liệu NoSQL dạng cột** (Column-Oriented NoSQL Database), hoạt động trên hệ thống tệp phân tán **HDFS.**

HBase cung cấp khả năng **đọc, ghi nhanh**, **khả năng mở rộng theo chiều ngang** và **khả năng xử lý dữ liệu lớn**, phù hợp với các ứng dụng yêu cầu độ trễ thấp và lưu trữ lượng dữ liệu khổng lồ như xử lý dữ liệu lớn trong thời gian thực.

### 2.2. Cấu trúc dữ liệu trong HBase



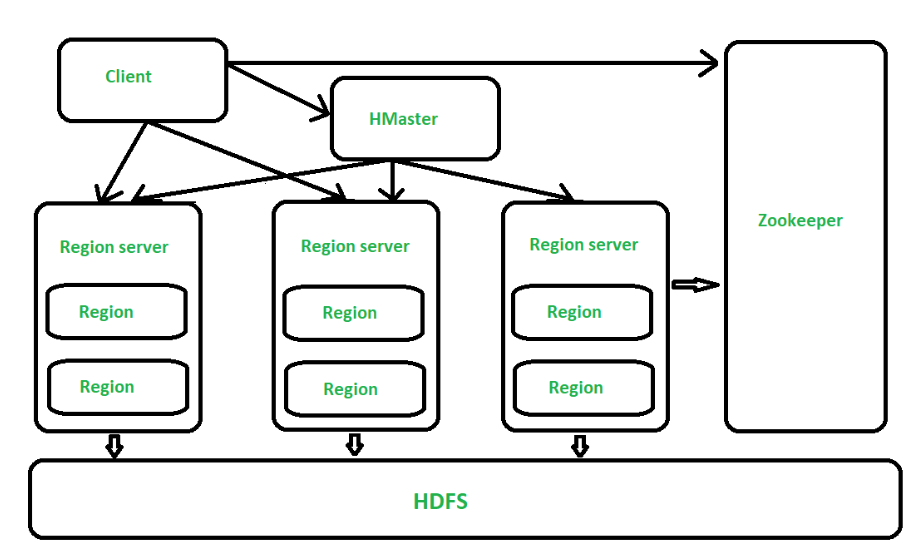
Hình 2. Minh họa dữ liệu trong một bảng HBase

Dữ liệu trong HBase không được lưu trữ dưới dạng bảng với hàng và cột cố định như trong các cơ sở dữ liệu quan hệ truyền thống, mà được tổ chức theo mô hình cột động với khả năng mở rộng linh hoạt.

Mỗi hàng trong bảng HBase được xác định bằng một Row Key duy nhất, đóng vai trò như khóa chính để phân biệt các bảng ghi (row) khác nhau. Row Key được sắp xếp theo thứ tự từ điển (lexicographical order), cho phép quét dữ liệu theo khoảng giá trị. Một bảng HBase không có giới hạn về số lượng cột, giúp tăng cường tính linh hoạt trong lưu trữ dữ liệu.

Dữ liệu trong một bảng được chia thành một hoặc nhiều Column Family, các cột thuộc cùng một family được lưu trữ cùng nhau trên HDFS, nói cách khác là dữ liệu của các column family khác nhau có thể được phân tán trên nhiều máy. Column family phải được tạo trước khi dữ liệu có thể được lưu trữ trong nó. Ngoài ra, số lượng column family thường được giả định là nhỏ (tối đa vài trăm) và ít thay đổi trong quá trình vận hành. Tuy nhiên, một bảng có thể có số lượng cột không giới hạn, và cột có thể được thêm vào một gia đình cột một cách linh hoạt. Do đó, các hàng của cùng một bảng phải có cùng các column family, nhưng có thể chứa các cột hoàn toàn khác nhau.

### 2.3. Kiến trúc



Hình 3. Kiến trúc của Apache HBase

HBase bao gồm 3 thành phần chính: HMaster, Zookeeper, và RegionServers

* HMaster: Xử lý các thao tác DDL (tạo và xóa).
* Zookeeper: Là dịch vụ điều phối phân tán cho cụm HBase, giúp quản lý và duy trì trạng thái giữa các thành phần trong cụm.
* RegionServer: Quản lý các phân vùng dữ liệu (Regions), xử lý các yêu cầu đọc và ghi dữ liệu từ phía client.

Dữ liệu của một bảng HBase được lưu trong một hoặc nhiều region. Khi kích thước bảng tăng lên, số lượng region cũng tăng theo. HMaster quản lý việc phân bổ các region này. Các region được lưu trữ trong RegionServers, nơi xử lý các yêu cầu PUT (ghi dữ liệu) và GET (đọc dữ liệu) từ client. Mỗi RegionServer thường được đặt trên một DataNode của HDFS và dữ liệu được lưu trữ trên hệ thống tệp phân tán HDFS.

Thông tin về việc ánh xạ giữa Region và RegionServer được lưu trữ trong một metatable trên ZooKeeper. Khi client gửi yêu cầu, nó sẽ được chuyển đến ZooKeeper để ZooKeeper cung cấp cho RegionServer vị trí chứa dữ liệu cần thiết. Sau đó, client sẽ thực hiện trực tiếp các thao tác đọc hay ghi trên RegionServer đó.

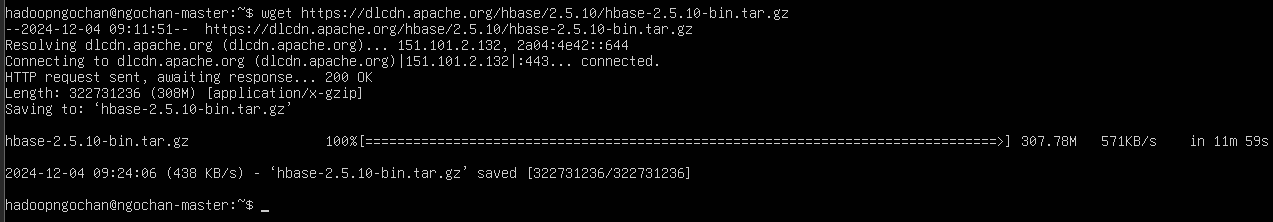
### 2.4. Cài đặt HBase ở chế độ Pseudo-Distributed

Yêu cầu hệ thống: Apache Hadoop 3, JDK 8

#### 2.4.1. Tải và cấu hình HBase

* **Tải Hbase:**

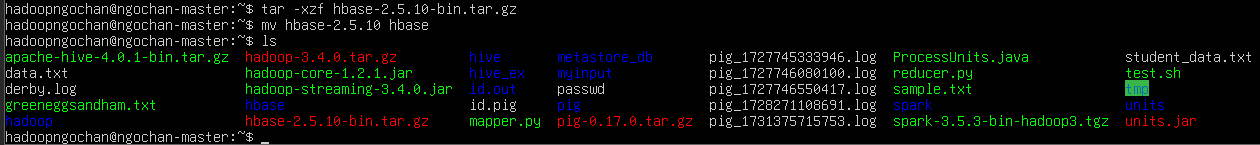
$ wget <https://dlcdn.apache.org/hbase/2.5.10/hbase-2.5.10-bin.tar.gz>



* **Giải nén và đổi tên thư mục thành hbase**

$ tar -xzf hbase-2.5.10-bin.tar.gz

$ mv hbase-2.5.10 hbase

****

* **Khai báo biến môi trường**

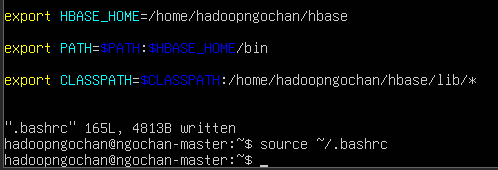
$ vim ~/.bashrc

export HBASE\_HOME=/home/hadoopngochan/hbase

export PATH=$PATH:$HBASE\_HOME/bin

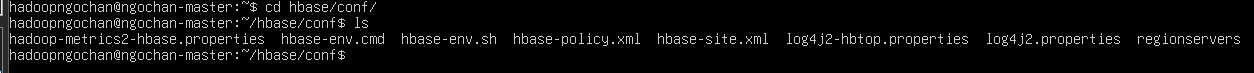
export CLASSPATH=$CLASSPATH:/home/hadoopngochan/hbase/lib/\*

$ source ~/.bashrc



* **Thiết lập biến môi trường cho Hbase**

$ cd hbase/conf

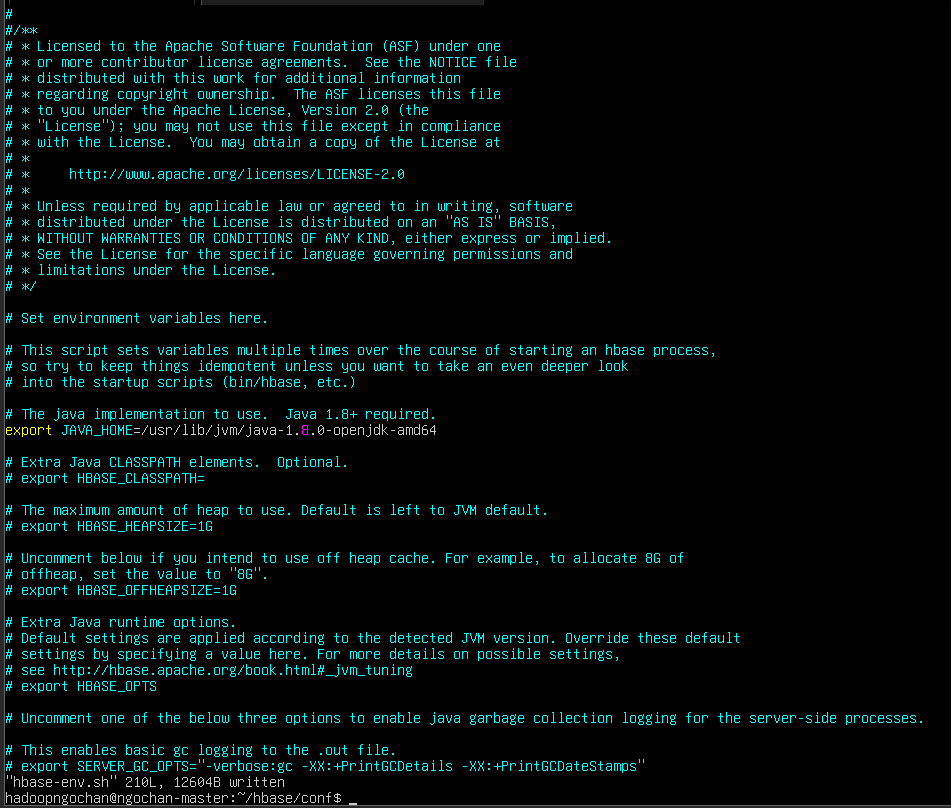


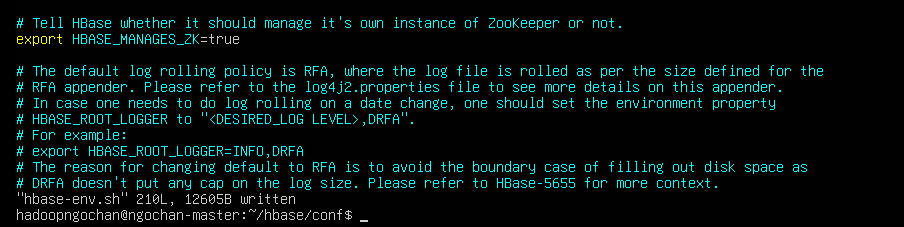
$ vim hbase-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64

export HBASE\_MANAGES\_ZK=true

HBase yêu cầu Java để hoạt động, vì vậy cần xác định đúng đường dẫn đến thư mục chứa Java. Biến HBASE\_MANAGES\_ZK cho biết HBase có tự quản lý ZooKeeper hay không. Ở đây em đang thực hiện cài đặt kiểm thử, nên sẽ để HBase tự động quản lý ZooKeeper.





$ vim hbase-site.xml



  <property>

<name>hbase.cluster.distributed</name>

<value>true</value>

</property>

Cấu hình này xác định HBase hoạt động trong một môi trường phân tán thay vì là standalone ở local filesystem.

<property>

<name>hbase.rootdir</name>

<value>hdfs://ngochan-master:9000/hbase</value>

</property>

Tham số hbase.rootdir xác định vị trí thư mục gốc mà HBase sử dụng để lưu trữ dữ liệu của mình. Trong cấu hình này, hbase.rootdir được thiết lập thành hdfs://ngochan-master:9000/hbase, chỉ ra rằng HBase sẽ lưu trữ dữ liệu của mình trên hệ thống phân tán HDFS tại máy chủ ngochan-master với cổng 9000. Lưu ý không gần tạo trước thư mục hbase trong hdfs.

<property>

<name>hbase.wal.provider</name>

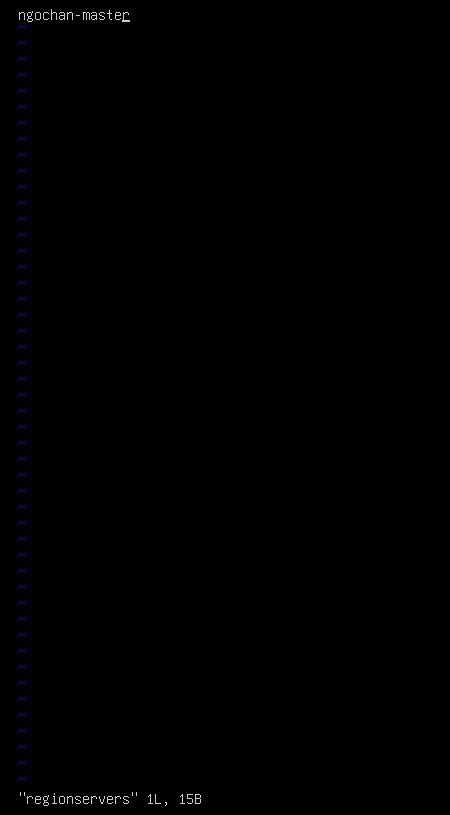
<value>filesystem</value>

</property>

Cấu hình này xác định cơ chế ghi WAL (Write-Ahead Log) trong HBase sẽ được lưu trực tiếp trên HDFS. WAL là cơ chế ghi log trước khi ghi dữ liệu vào hệ thống chính nhằm bảo vệ tính toàn vẹn của dữ liệu và hỗ trợ phục hồi sau sự cố.

$ vim regionservers

File này là nơi mà HBase xác định các nodes chạy RegionServer để nó phân bổ công việc

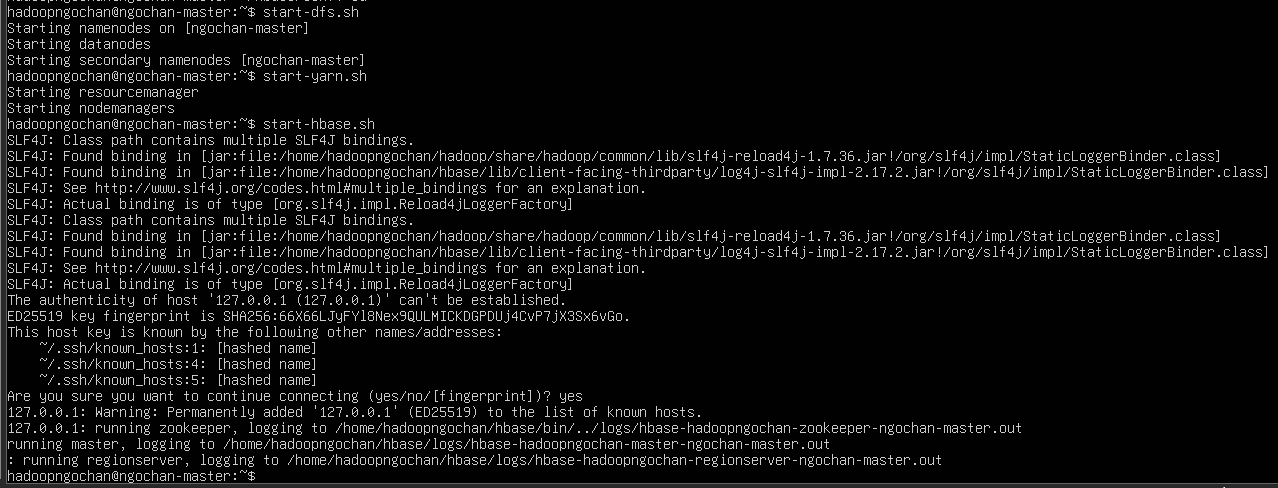


#### 2.4.2. Chạy HBase và kiểm tra

* **Khởi động các tiến trình Hadoop**

$ start-dfs.sh

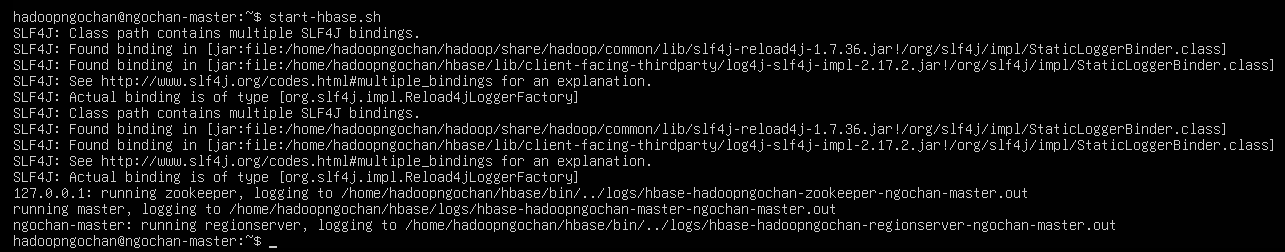
$ start-yarn.sh



$ zkServer.sh start

* **Khởi động HBase**

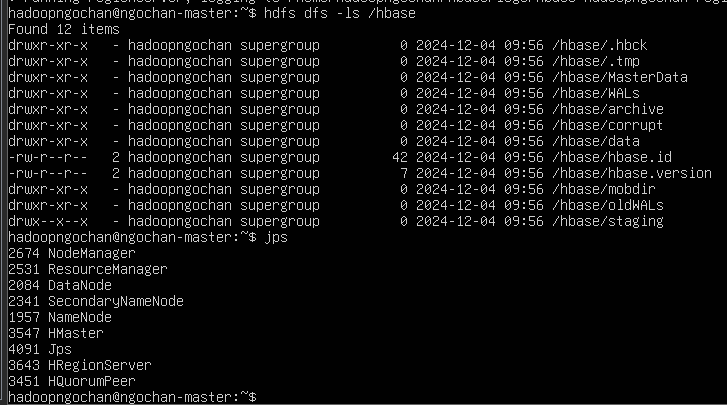
$ start-hbase.sh



* **Kiểm tra**

$ hdfs dfs -ls /hbase

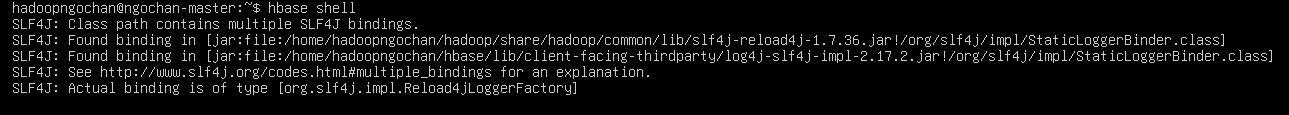
$ jps



Thư mục lưu dữ liệu của HBase trên hdfs đã được tạo và các tiến trình HMaster, HRegionServer, HquorumPeer đã được khởi động thành công.

* **Kết nối đến HBase**

$ hbase shell



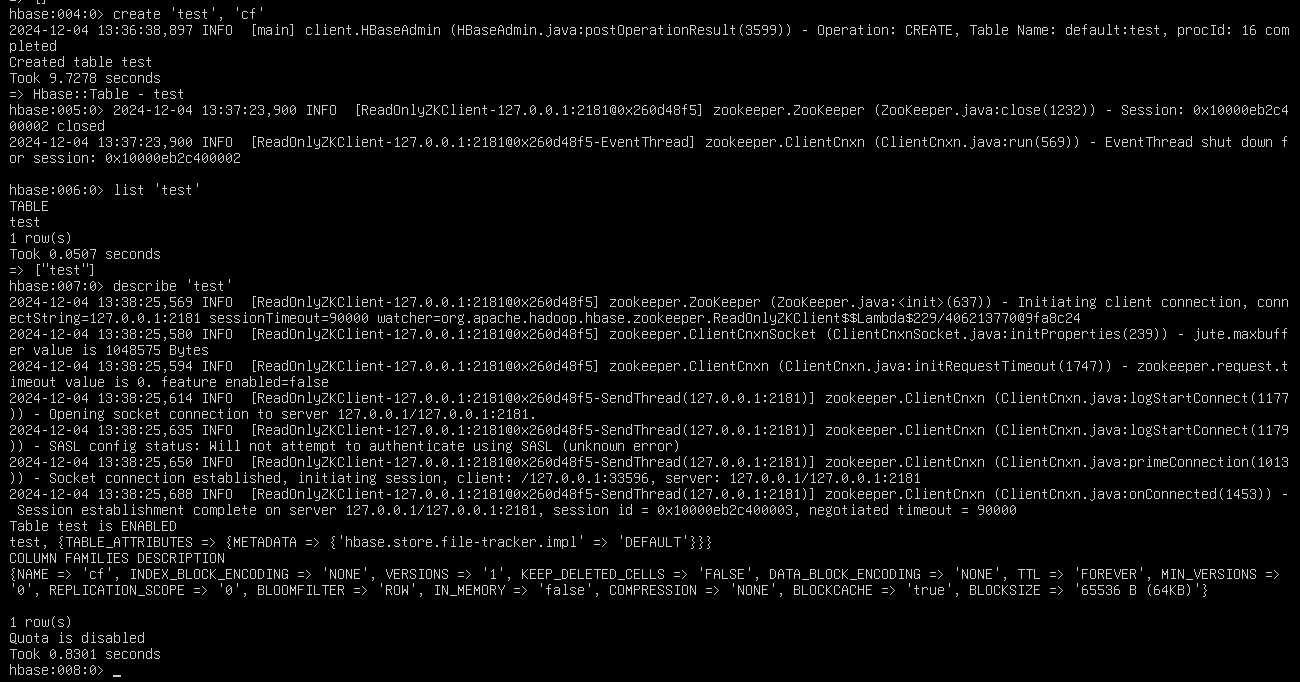


* **Tạo bảng test với một column family cf và xem mô tả bảng**

>create 'test', 'cf'

>list ‘test’

>describe ‘test’

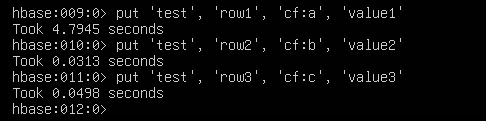


* **Thêm dữ liệu vào bảng test**

>put 'test', 'row1', 'cf:a', 'value1'

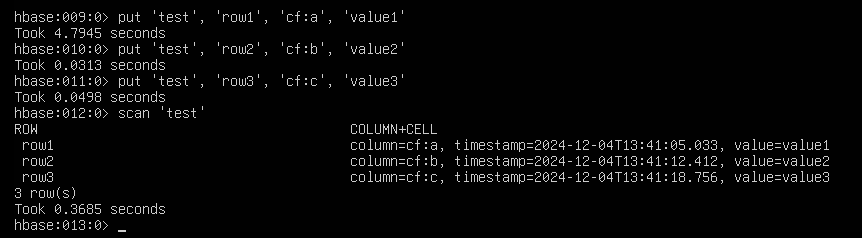
>put 'test', 'row2', 'cf:b', 'value2'

>put 'test', 'row3', 'cf:c', 'value3'



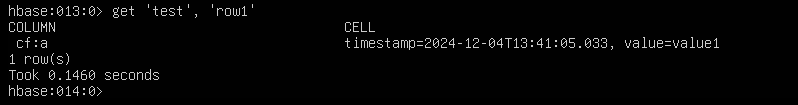
* **Xem toàn bộ dữ liệu của bảng test**

>scan 'test'



* **Xem dữ liệu của hàng có row key là “row1”**

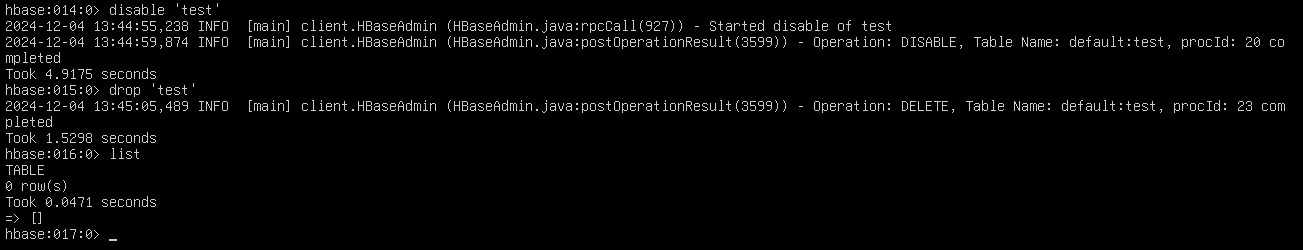
>get 'test', 'row1'



* **Trong một số trường hợp, ta cần disable một bảng trước khi thực hiện thao tác lên bảng, ví dụ như khi xóa bảng**

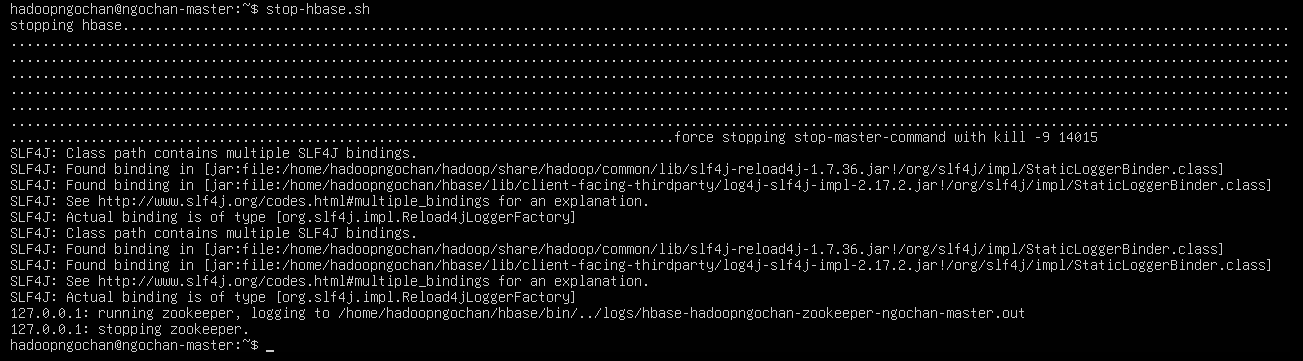
>disable 'test'

>drop 'test'



* **Dừng HBase**

stop-hbase.sh



### 2.5. Cài đặt HBase ở chế độ Fully Distributed

Ở mục trên, chúng ta đã cài đặt HBase hoạt động ở chế độ phân tán nhưng chỉ trên một máy chủ và sử dụng ZooKeeper tích hợp sẵn. Lưu ý rằng khi dùng built-in ZoopKeeper trong HBase thì chỉ có một instance của ZooKeeper chạy trên mỗi node được xác định bởi biến *hbase.zookeeper.quorum* trong tệp hbase-site.xml, và khi tắt/bật HBase thì ZooKeeper cũng tắt/bật theo, nên chỉ phù hợp cho testing hoặc môi trường nhỏ. Đối với môi trường sản xuất, sử dụng ZooKeeper độc lập là một lựa chọn tốt hơn, giúp đảm bảo hiệu suất, khả năng chịu lỗi và mở rộng của cả HBase và ZooKeeper.

#### 3.5.1. Cài đặt

Sau đây em sẽ thực hiện cài đặt để HBase hoạt động trên một Hadoop Cluster gồm hai node với kiến trúc sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Node Name** | **Master** | **ZooKeeper** | **RegionServer** |
| ngochan-master | yes | yes | yes |
| ngochan-slave1 | no | no | yes |

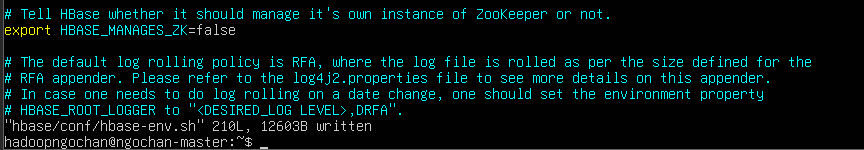
* **Chỉ định các RegionServer**

$ vim hbase/conf/regionservers

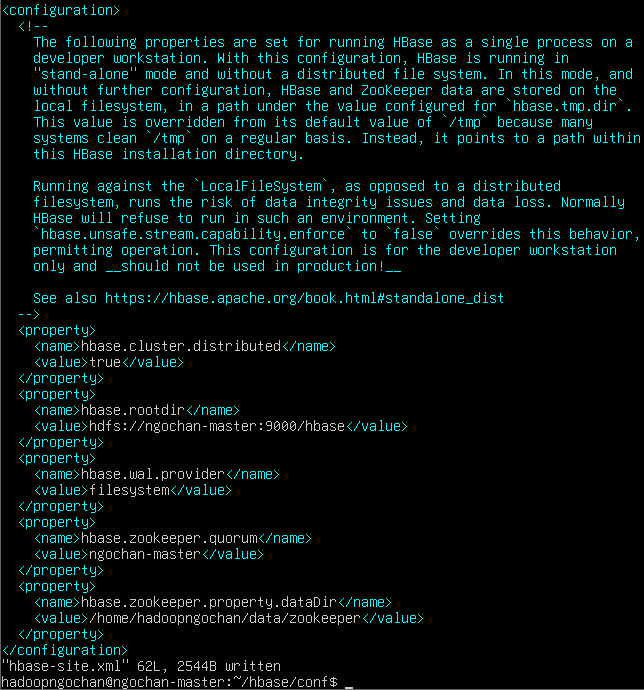


* **Cập nhật cấu hình ZooKeeper trong HBase**

$ vim hbase/conf/hbase-env.sh



$ vim hbase/conf/hbase-site.xml



<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/home/hadoopngochan/data/zookeeper </value>

</property>

Giá trị của hbase.zookeeper.property.dataDir chỉ định nơi lưu trữ dữ liệu của ZooKeeper, mặc định là /tmp nhưng em đổi sang một địa chỉ cố định khác (cùng giá trị với dataDir trong tệp cấu hình ZooKeeper zoo.cfg) để dữ liệu của nó không bị mất khi hệ thống khởi động lại.

<property>

<name>hbase.zookeeper.quorum</name>

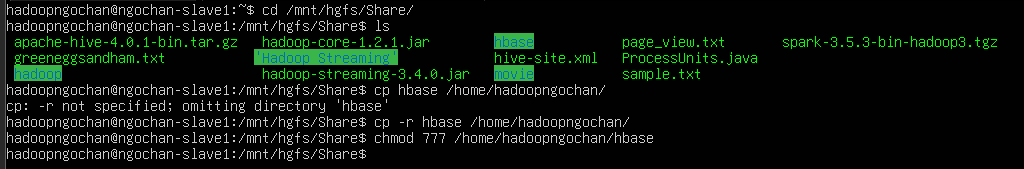
<value>ngochan-master </value>

</property>

*hbase.zookeeper.quorum* xác định danh sách các máy chủ Zookeeper mà HBase sử dụng để quản lý trạng thái và phối hợp trong cụm. Càng nhiều máy chủ Zookeeper thì khả năng chịu lỗi của hệ thống càng cao. Ngoài ra, số lượng máy chủ Zookeeper nên là số lẻ để đảm bảo tính ổn định. Ở đây chỉ có một máy chạy Zookeeper là ngochan-master. Trong trường hợp muốn chạy nhiều máy chủ Zookeeper thì thêm hostname khác vào và ngăn cách nhau bằng dấy phẩy.

* **Các node trong cluster cần phải cài đặt HBase và có thông tin cấu hình HBase giống nhau, nên thực hiện copy hbase sang máy Slave**





* **Khai báo biến môi trường**

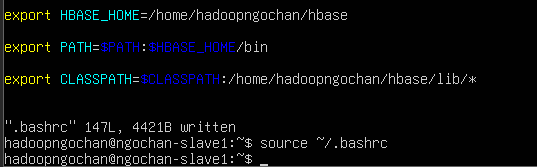
$ vim ~/.bashrc

export HBASE\_HOME=/home/hadoopngochan/hbase

export PATH=$PATH:$HBASE\_HOME/bin

export CLASSPATH=$CLASSPATH:/home/hadoopngochan/hbase/lib/\*

$ source ~/.bashrc



#### 1.5.2. Chạy HBase

* **Khởi động HBase Shell**

Lưu ý cần khởi động các hdfs (start-dfs.sh và start-yarn.sh)

$ start-hbase.sh

$ hbase-daemons.sh start zookeeper

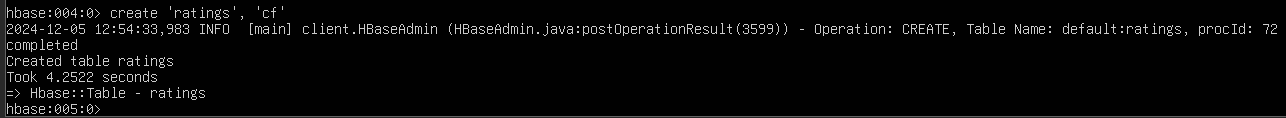


Thông tin trong hình trên cho thấy đã cấu hình thành công: ngochan-master chạy Hmaster, RegionServer và ZooKeeper. ngochan-slave1 chạy RegionServer.

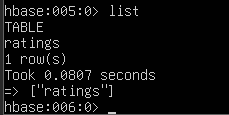
$ hbase shell

* **Tạo bảng “ratings” với column family “cf”**

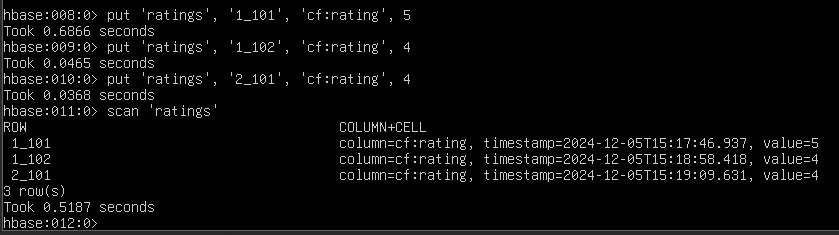
>create 'ratings', 'cf'



> list



* **Thêm dữ liệu vào bảng và scan bảng**

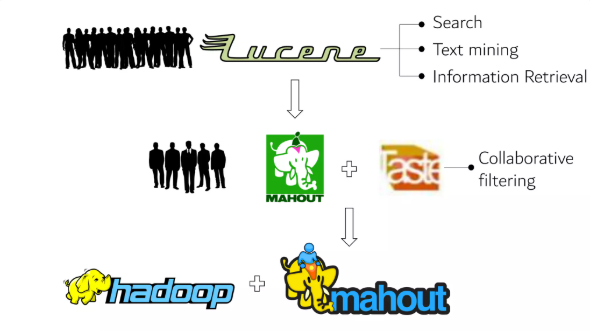


## 3. Mahout

### 3.1. Tổng quan

Apache Mahout là một dự án mã nguồn mở phát triển bởi Apache Software Foundation (ASF) được sử dụng để xây dựng các mô hình học máy (machine learning models) và xử lý dữ liệu lớn. Mahout ban đầu được phát triển trên nền tảng Hadoop và được thiết kế để thực hiện các thuật toán học máy phân tán cho phân cụm, phân loại và hệ thống gợi ý, cho phép xử lý dữ liệu với quy mô lớn mà không gặp phải vấn đề về hiệu suất.

Lịch sử hình thành và phát triển của Apache Mahout:



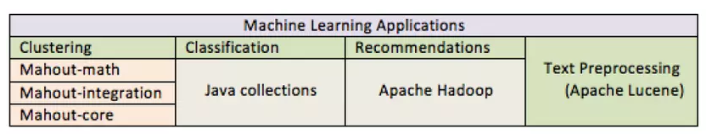
Hình 4. Lịch sử hình thành và phát triển Mahout

Apache Mahout được ra đời vào năm 2008 như một dự án con của Apache Lucene, tập trung phát triển các thuật toán học máy như phân cụm, phân loại và gợi ý. Năm 2010, Mahout trở thành dự án cấp cao của Apache và được tối ưu hóa cho các nền tảng phân tán như Hadoop. Hiện nay, Mahout hỗ trợ Apache Spark, giúp triển khai học máy hiệu quả trên quy mô lớn.

Các đặc điểm chính:

* Đây là một dự án của nền tảng phần mềm Apache.
* Là một thư viện học máy khả năng mở rộng:
* Thi hành MapReduce theo quy mô đường thẳng với dữ liệu
* Thuật toán tuần tự nhanh ( quá trình thi hành không phụ thuộc vào kích thước của tập dữ liệu )
* Chủ yếu là các thuật toán phân nhóm, phân loại, và đề xuất (lọc cộng tác).
* Thuật toán học máy có thể được thi hành tuần tự: (in-memory mode) or distributed mode (MapReduce được kích hoạt)
* Hầu hết các thuật toán được thực hiện bằng cách mô hình hóa MapReduce.
* Chạy trên nền tảng Hadoop cho việc mở rộng.
* Dữ liệu được lưu trữ trong HDFS (data storage) hoặc trong bộ nhớ.
* Nó là một thư viện Java (không có giao diện người dùng).

### 3.2. Kiến trúc

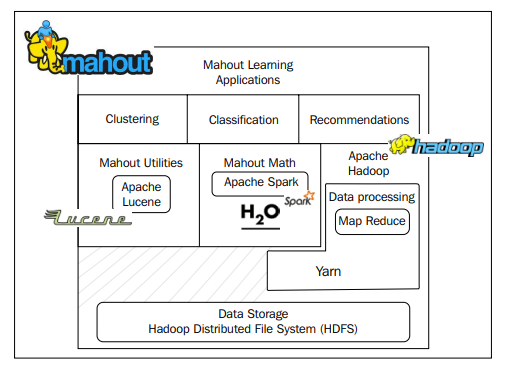


Hình 5. Các thiết kế cao cấp của một Mahout

Ứng dụng học máy sẽ truy cập API của mahout cung cấp hỗ trợ triển khai kỹ thuật như phân cụm (Clustering), phân loại(Classification), và đề xuất (Recommendation)

Nếu đầu vào là văn bản, các bước như **loại bỏ stop words** hoặc **stemming** có thể được thực hiện nhờ **Apache Lucene.** Sử dụng **Java Collections** và thư viện toán học **Mahout-math** để xử lý các phép toán phức tạp (vector, ma trận, phân rã SVD). Tích hợp thư viện **Mahout-integration** để cung cấp các tiện ích hiển thị dữ liệu và kết quả.

Ngoài ra, tích hợp với **Apache Hadoop** để cung cấp khả năng xử lý và lưu trữ dữ liệu phân tán, đảm bảo khả năng mở rộng khi làm việc với tập dữ liệu lớn.



Hình 6. Kiến trúc của Mahout trong Hadoop

### 3.3. Các thuật toán trong Mahout

**Classification**

* Logistic Regression
* Bayes
* Support Vector Machines
* Perceptrons
* Neural Networks
* Restricted Boltzmann Machines
* Hidden Markov Models

**Clustering**

* Canopy
* K-Means
* Fuzzy K-Means
* Hierarchical Clustering
* Minhash Clustering

**Recommendation**

* User-based Collaborative Filtering
* Item-based Collaborative Filtering
* SlopeOne Recommenders
* Singular Value Decomposition

### 3.4. Cài đặt Mahout

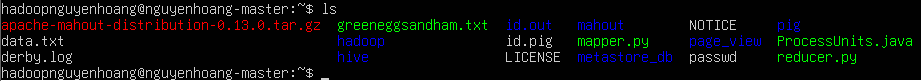
* **Tải Apache Mahout**

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



* **Giải nén và đổi tên thư mục mahout**

Kiểm tra file giải nén đã tải về : $ ls



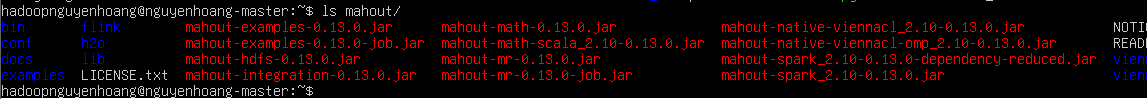
* **Giải nén và đổi tên:**

$ tar -xzf [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)

$ mv apache-mahout-distribution-0.13.0 mahout



* **Kiểm tra thư mục đã giải nén:**

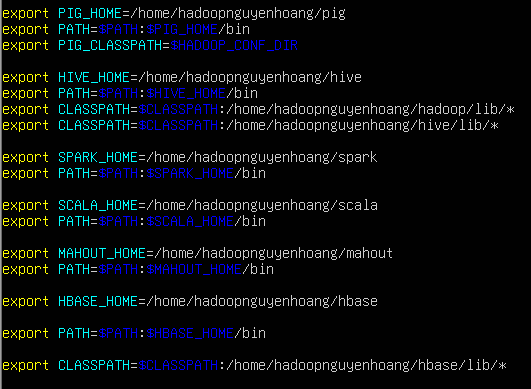


* **Khai báo biến môi trường ở máy master:**

$ vim ~/.bashrc

export MAHOUT\_HOME=/home/hadoopnguyenhoang/mahout

export PATH=$PATH:$MAHOUT\_HOME/bin



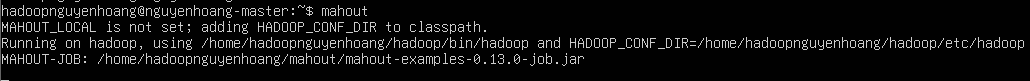
Nhấn Esc và gõ wq để lưu lại

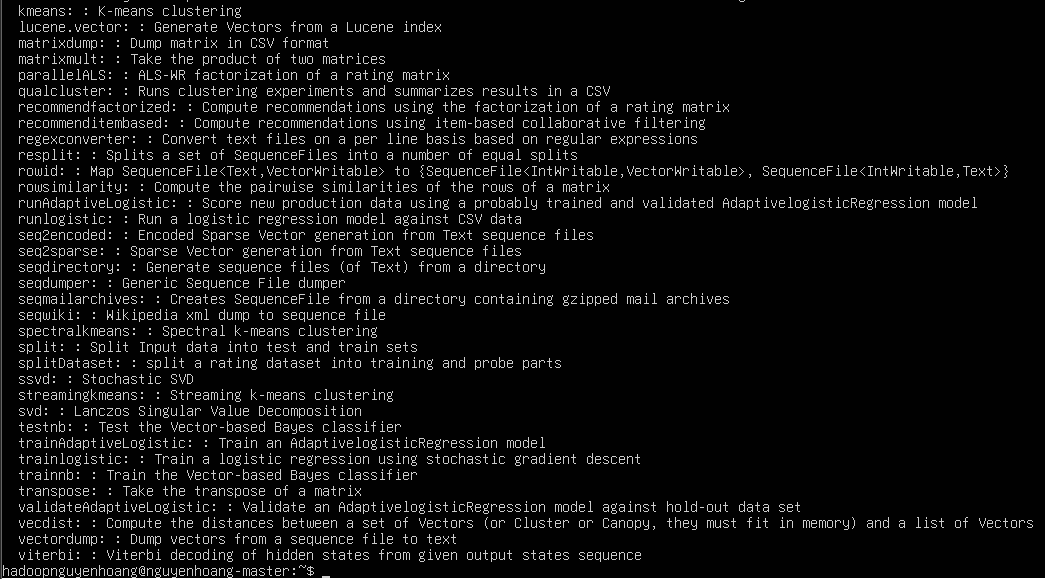
Soucre file .bashrc

$ source ~/.bashrc

* **Check Mahout**

Chạy lệnh mahout:





## 4. Zookeeper

### 4.1. Tổng quan

#### 4.1.1. Khái niệm Apache Zookeeper

Apache ZooKeeper là một dịch vụ phân tán dùng để điều phối các ứng dụng trong môi trường phân tán. Nó cung cấp một hệ thống đồng bộ hóa phân tán, quản lý cấu hình, và cung cấp dịch vụ đặt tên (naming service).

ZooKeeper rất phổ biến trong các hệ thống lớn như Hadoop, Kafka, và HBase.

**Lợi ích của việc sử dụng ZooKeeper:**

**-** Quy trình phối hợp phân tán đơn giản: ZooKeeper giúp quản lý và điều phối các thành phần trong hệ thống phân tán một cách dễ dàng, giảm độ phức tạp khi xây dựng và vận hành hệ thống.

**-** Đồng bộ hóa (Synchronization):

* Loại trừ lẫn nhau (Mutual exclusion): Đảm bảo rằng các tiến trình trên các máy chủ không truy cập đồng thời vào tài nguyên chung.
* Hợp tác giữa các tiến trình (Co-operation): Hỗ trợ các tiến trình phối hợp với nhau một cách hiệu quả. Ví dụ: Trong Apache HBase, ZooKeeper được sử dụng để quản lý cấu hình.

**-** Tuần tự hóa (Serialization):

* Dữ liệu được mã hóa theo các quy tắc cụ thể, giúp đảm bảo ứng dụng hoạt động nhất quán.
* Ứng dụng thực tế: Trong MapReduce, ZooKeeper được sử dụng để điều phối hàng đợi (queue) nhằm quản lý và thực thi các luồng chạy song song.

**-** Độ tin cậy (Reliability):ZooKeeper đảm bảo độ tin cậy cao cho hệ thống phân tán thông qua các cơ chế sử dụng tính nguyên tử (Atomicity): Mọi thao tác chuyển dữ liệu chỉ có thể thành công hoàn toàn hoặc thất bại hoàn toàn, không bao giờ có trạng thái trung gian. Điều này giúp đảm bảo tính nhất quán và toàn vẹn dữ liệu.

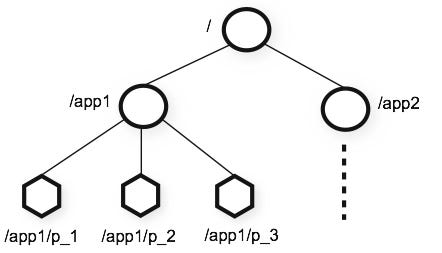
#### 4.1.2. ZooKeeper Data Model

**ZooKeeper Data Model:** Là một trong những khái niệm cốt lõi của ZooKeeper, định nghĩa cách dữ liệu được tổ chức, truy cập và thao tác trong hệ thống. Nó hoạt động như một không gian phân cấp (hierarchical namespace), tương tự như cấu trúc thư mục trong một hệ thống tệp (file system).

**Đặc điểm chính của ZooKeeper Data Model:**

**1. Cấu trúc phân cấp (Hierarchical Namespace):**

* **Node:** Mỗi điểm trong cấu trúc được gọi là một **znode**.
* Một **znode** có thể lưu trữ dữ liệu và cũng có thể có các znode con (giống như một thư mục vừa có thể chứa file, vừa có thể chứa thư mục khác).



Hình 7. Cấu trúc phân cấp của Zookeeper

Trong ví dụ trên:

* /app1 và /app2 là các znode gốc.
* /app1/p\_1 là một znode con của /app1.

**2. Đường dẫn tuyệt đối:**

* Mọi znode đều có một **đường dẫn tuyệt đối** (absolute path), bắt đầu bằng dấu /.
* Không hỗ trợ các đường dẫn tương đối như . hay ...
* Ví dụ:
  + Hợp lệ: /app1/p\_1
  + Không hợp lệ: /app1/./p\_1, /app1/../p\_1

**3. Dữ liệu trong znode:**

* Mỗi znode có thể lưu trữ một khối dữ liệu nhỏ (dung lượng tối đa khoảng 1 MB).
* Znode không được thiết kế để lưu trữ dữ liệu lớn như một hệ thống file thông thường.

**4. Các ký tự hợp lệ trong đường dẫn:**

* ZooKeeper hỗ trợ hầu hết các ký tự Unicode, nhưng có một số ký tự bị cấm hoặc bị hạn chế như: ký tự null: \u0000, tên đặc biệt(ví dụ: /zookeeper) là tên dành riêng của hệ thống và không được sử dụng bởi người dùng.

#### 4.1.3. Khái niệm Znodes, Session và Watches

* **Znode**: Znode là khái niệm trung tâm trong mô hình dữ liệu của ZooKeeper, dùng để biểu diễn các nút trong cây phân cấp (namespace). Mỗi ZNode chứa dữ liệu, metadata (stat structure), và có các đặc tính quan trọng được thiết kế để hỗ trợ các hệ thống phân tán.
* **Các loại Znode:**

**Persistence znode (Znode bền vững)**:

Znode bền vững sẽ tồn tại ngay cả khi client, mà đã tạo ra znode đó, bị ngắt kết nối. Mặc định, tất cả các znode đều là bền vững trừ khi có yêu cầu khác.

**Ephemeral znode (Znode tạm thời)**:

Znode tạm thời sẽ vẫn hoạt động cho đến khi client kết nối bị ngắt. Khi client bị ngắt kết nối với ZooKeeper ensemble, các znode tạm thời sẽ tự động bị xóa. Vì lý do này, các znode tạm thời không được phép có con (children) nữa. Nếu một znode tạm thời bị xóa, thì node tiếp theo phù hợp sẽ thay thế vị trí của nó. Znode tạm thời đóng vai trò quan trọng trong việc bầu chọn Leader.

**Sequential znode (Znode tuần tự)**:

Znode tuần tự có thể là bền vững hoặc tạm thời. Khi một znode mới được tạo ra như là một znode tuần tự, ZooKeeper sẽ gán một số thứ tự gồm 10 chữ số vào tên ban đầu của znode đó. Ví dụ, nếu một znode với đường dẫn là /myapp được tạo ra như một znode tuần tự, ZooKeeper sẽ thay đổi đường dẫn thành /myapp0000000001 và gán số thứ tự tiếp theo là 0000000002. Nếu hai znode tuần tự được tạo ra đồng thời, ZooKeeper sẽ không bao giờ sử dụng cùng một số cho mỗi znode. Znode tuần tự đóng vai trò quan trọng trong việc khóa và đồng bộ hóa.

* **Sessions (Phiên làm việc)**

Phiên làm việc (Session) rất quan trọng đối với hoạt động của ZooKeeper. Các yêu cầu trong một phiên làm việc sẽ được thực hiện theo thứ tự FIFO (First In, First Out). Khi một client kết nối với server, một phiên làm việc sẽ được thiết lập và một ID phiên (session ID) sẽ được cấp cho client.

Client gửi tín hiệu "heartbeat" (tín hiệu sống) ở một khoảng thời gian cụ thể để duy trì phiên làm việc hợp lệ. Nếu ZooKeeper ensemble không nhận được tín hiệu heartbeat từ một client trong khoảng thời gian vượt quá thời gian (session timeout) đã được chỉ định khi bắt đầu dịch vụ, hệ thống sẽ xác định rằng client đã "chết".

Thời gian hết phiên (session timeout) thường được biểu thị bằng mili giây. Khi một phiên làm việc kết thúc vì bất kỳ lý do nào, các znode tạm thời (ephemeral znodes) được tạo ra trong phiên đó cũng sẽ bị xóa.

* **Watches**

Watches là một cơ chế đơn giản giúp client nhận thông báo về các thay đổi trong ZooKeeper ensemble. Client có thể thiết lập watch khi đọc một znode cụ thể. Các watch gửi thông báo tới client đã đăng ký khi có sự thay đổi nào xảy ra đối với znode (mà client đã đăng ký) đó.

Sự thay đổi của znode có thể là sự sửa đổi dữ liệu liên kết với znode hoặc sự thay đổi trong các con (children) của znode. Watches chỉ được kích hoạt một lần. Nếu client muốn nhận thông báo lại, họ phải thực hiện một thao tác đọc khác. Khi một phiên làm việc hết hạn, client sẽ bị ngắt kết nối với server và các watch liên kết cũng sẽ bị xóa.

#### 4.1.4. Zookeeper và HBase

**Apache HBase** là một hệ thống lưu trữ dữ liệu phân tán, thường được sử dụng cùng với Hadoop để xử lý và lưu trữ dữ liệu lớn. HBase là cơ sở dữ liệu NoSQL, có khả năng lưu trữ và quản lý khối lượng dữ liệu rất lớn và cung cấp khả năng truy cập dữ liệu theo thời gian thực. ZooKeeper đóng vai trò quan trọng trong việc duy trì sự đồng bộ hóa và quản lý các thành phần trong hệ thống HBase.

Cụ thể, **ZooKeeper** trong HBase được sử dụng cho các chức năng sau:

1. **Bầu chọn Master Cluster:**

ZooKeeper giúp HBase xác định và bầu chọn một **Master Node** trong cụm (cluster). Master này chịu trách nhiệm quản lý các Region Server và các hoạt động điều phối giữa các máy chủ trong HBase. Khi có sự cố hoặc thay đổi trạng thái của Master, ZooKeeper đảm bảo rằng một Master mới được bầu chọn kịp thời và hoạt động không bị gián đoạn.

1. **Theo dõi các máy chủ có sẵn:**

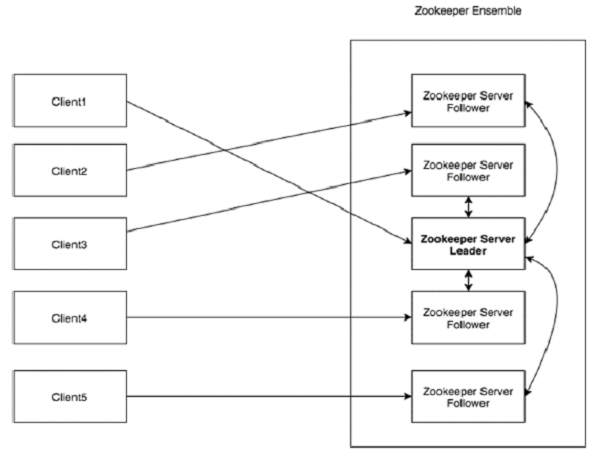
ZooKeeper theo dõi trạng thái của các **Region Servers** trong cụm. Các Region Server là các nút (node) lưu trữ dữ liệu của HBase. ZooKeeper đảm bảo rằng khi một Region Server gặp sự cố, nó sẽ thông báo và có thể được thay thế hoặc khởi động lại mà không gây ảnh hưởng đến toàn bộ hệ thống.

1. **Quản lý metadata của cluster:**

ZooKeeper lưu trữ dữ thông tin về **metadata** của HBase cluster, bao gồm thông tin về các bảng (tables), các region, và các phân vùng (partitions) của dữ liệu trong hệ thống. Các thông tin này rất quan trọng để duy trì tính nhất quán của liệu và giúp HBase có thể phân phối dữ liệu trên các Region Servers một cách hiệu quả.

Tóm lại, **ZooKeeper** giúp HBase duy trì tính ổn định, đồng bộ và phân tán trong môi trường cụm, đảm bảo hệ thống có thể phục hồi nhanh chóng khi gặp sự cố và duy trì hoạt động liên tục mà không gặp gián đoạn.

### 4.2. Kiến trúc



Hình 8. Kiến trúc của Zookeeper

Mỗi thành phần trong kiến trúc của ZooKeeper:

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả** |
| **Client** | Máy khách, một trong các node trong cụm ứng dụng phân tán, truy cập thông tin từ server. Trong một khoảng thời gian nhất định, mỗi client gửi một thông báo tới máy server để thông báo rằng client vẫn đang hoạt động. Tương tự, server sẽ gửi lại một xác nhận khi một client kết nối. Nếu không có phản hồi từ server đã kết nối, client sẽ tự động chuyển tiếp thông báo tới một server khác. |
| **Server** | Máy chủ, một trong các node trong cụm ZooKeeper, cung cấp tất cả các dịch vụ cho các client. Server gửi xác nhận cho client để thông báo rằng server vẫn đang hoạt động. |
| **Ensemble** | Nhóm các server ZooKeeper. Số lượng server tối thiểu cần thiết để tạo thành một nhóm là 3. |
| **Leader** | Máy chủ leader thực hiện khôi phục tự động nếu một trong các máy chủ kết nối bị lỗi. Máy chủ leader được bầu chọn khi dịch vụ khởi động. |
| **Follower** | Máy chủ follower tuân theo các chỉ thị từ máy chủ leader. |

### 4.3. Cài đặt Zookeeper

Cài đặt trên cả Master và Slave

* **Kiểm tra phiên bản Java:**



* **Tải xuống Apache ZooKeeper**

Truy cập trang web của Apache ZooKeeper và tải tệp:

<https://dlcdn.apache.org/zookeeper/zookeeper-3.9.3/apache-zookeeper-3.9.3-bin.tar.gz>



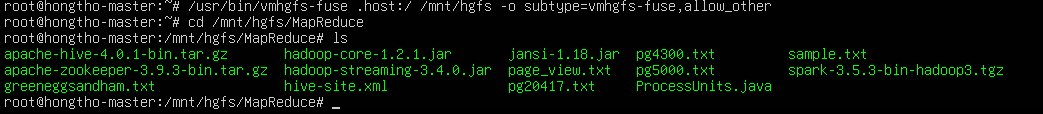
* **Share file vào hadoop**

Share file vào thư mục /mnt/hgfs:

$ /usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow\_other

$ cd /mnt/hgfs/MapReduce

$ ls

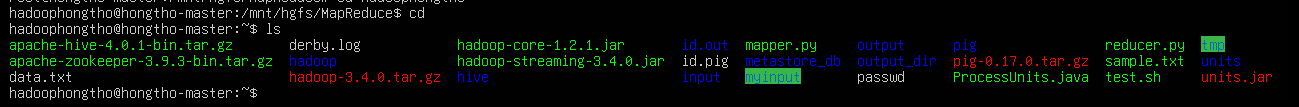


* **Copy sang hadoophongtho và cấp quyền:**

$ cp apache-zookeeper-3.9.3-bin.tar.gz /home/hadoophongtho/

$ chmod 777 /home/hadoophongtho/apache-zookeeper-3.9.3-bin.tar.gz



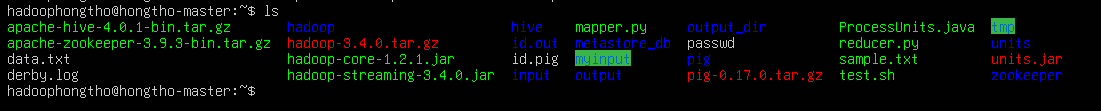


* **Giải nén và đổi tên thư mục**

$ tar -xzf apache-zookeeper-3.9.3-bin.tar.gz

$ mv apache-zookeeper-3.9.3-bin zookeeper





* **Cấu hình biến môi trường**

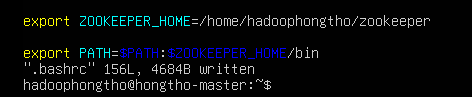
Mở tệp ~/.bashrc và thêm đường dẫn thực thi ZooKeeper:

$ vim ~/.bashrc

Thêm vào file 2 dòng sau:

export ZOOKEEPER\_HOME=/home/hadoophongtho/zookeeper

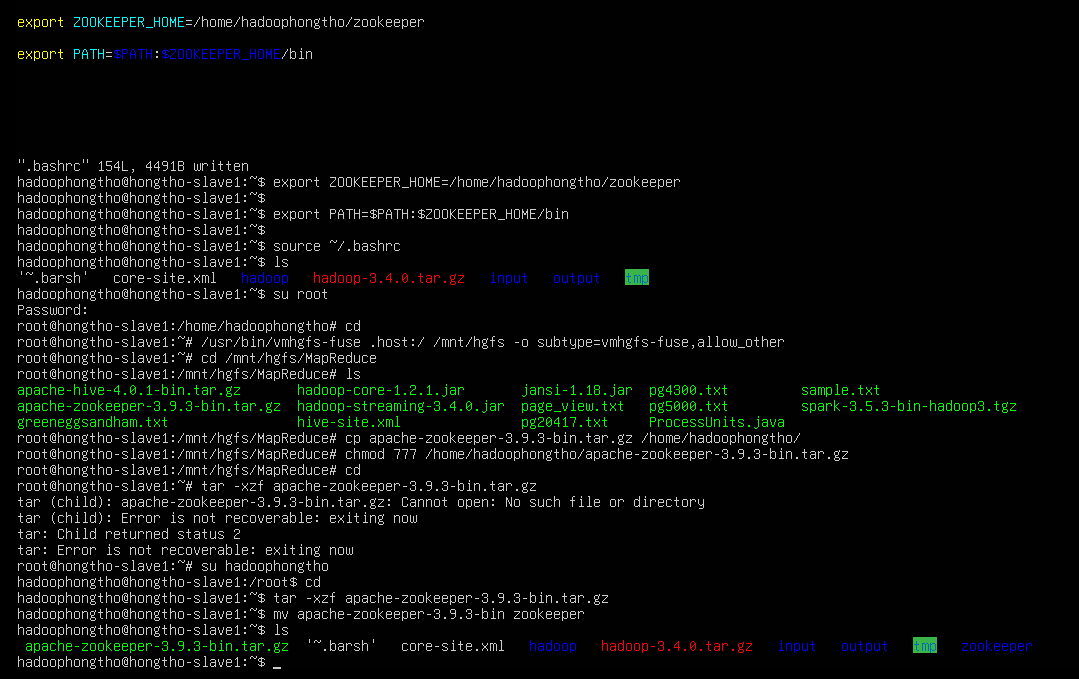
export PATH=$PATH:$ZOOKEEPER\_HOME/bin

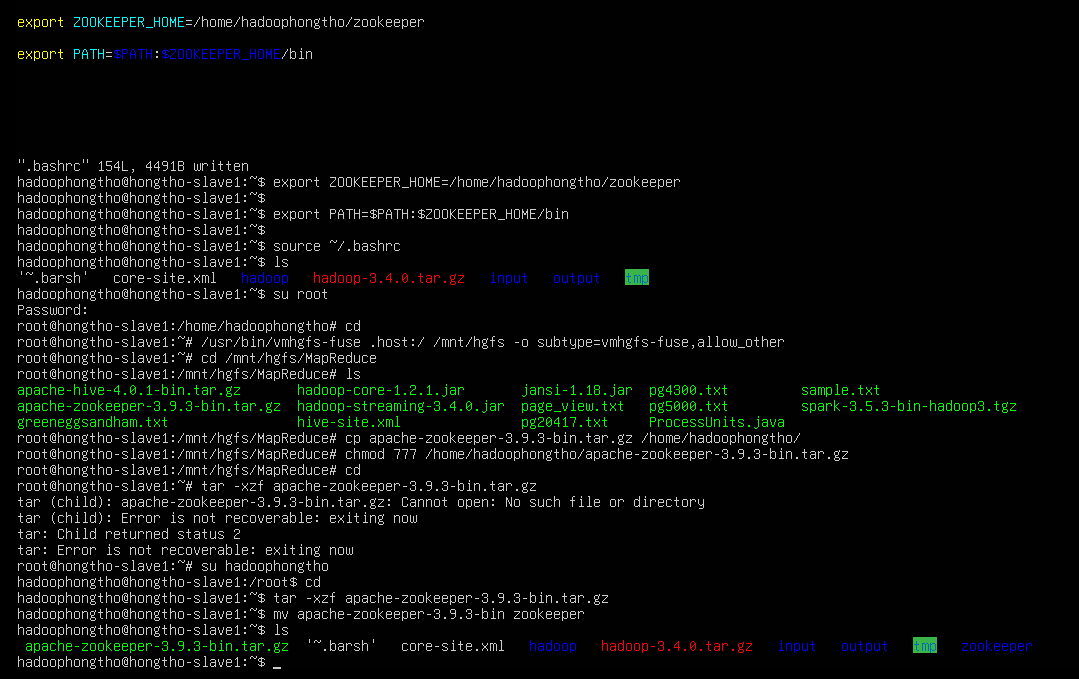


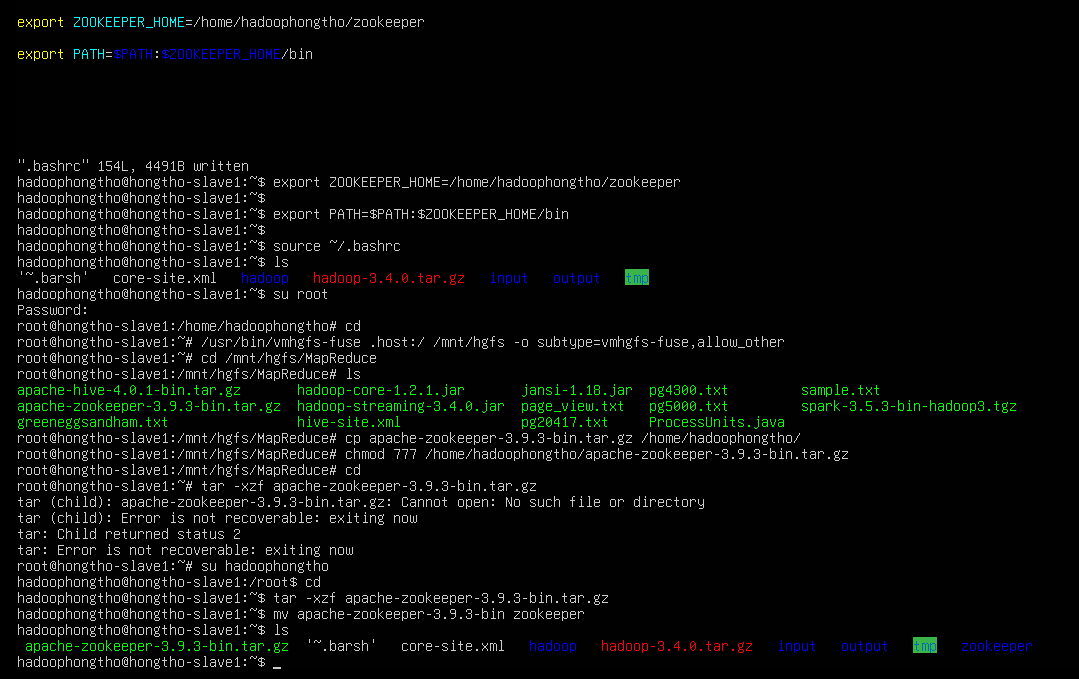
Chạy lệnh source ~/.bashrc để cập nhật biến môi trường.



Máy Slave cài đặt tương tự:







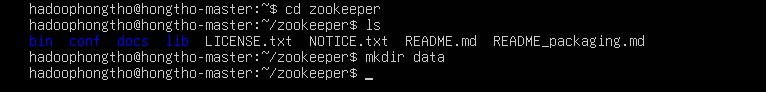
### 4.4. Cấu hình ZooKeeper chế độ standalone

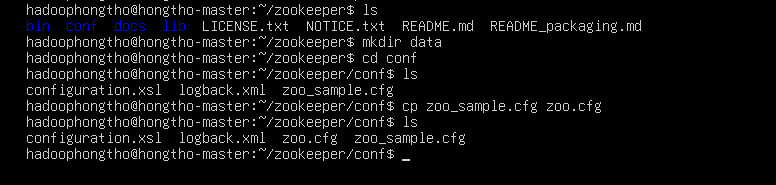
* **Cấu hình ZooKeeper (zoo.cfg): trên máy Master (Máy Server)**
* **Tạo folder tên data trong thư mục của zookeeper:**

$ mkdir data

* **Điều hướng đến thư mục /zookeeper/conf và sao chép tệp cấu hình mẫu:**

$ cp zoo\_sample.cfg zoo.cfg





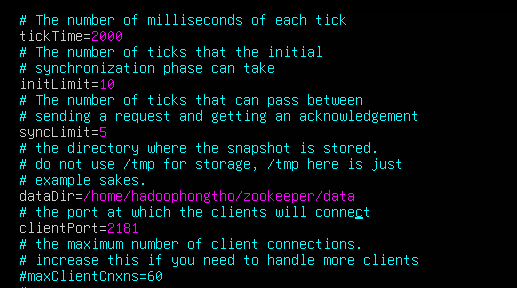
* **Chỉnh sửa tệp zoo.cfg:**

$ vim zookeeper/conf/zoo.cfg



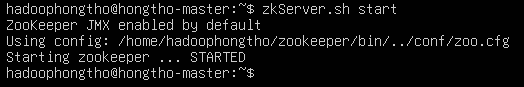
* **Cấu hình các tham số của Zookeeper:**

dataDir=/home/hadoophongtho/zookeeper/data



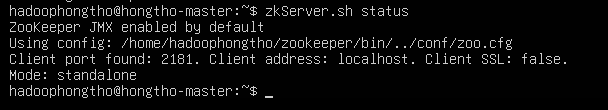
* **tickTime** là đơn vị thời gian cơ bản của ZooKeeper.
* **dataDir** là nơi lưu trữ dữ liệu trạng thái của ZooKeeper.
* **clientPort** là cổng giao tiếp với client.
* **initLimit** và **syncLimit** xác định các giới hạn thời gian cho follower đồng bộ hóa và giao tiếp với leader.
* **Khởi động ZooKeeper ở chế độ standalone:**

$ zkServer.sh start



* **Kiểm tra trạng thái**:

$ zkServer.sh status

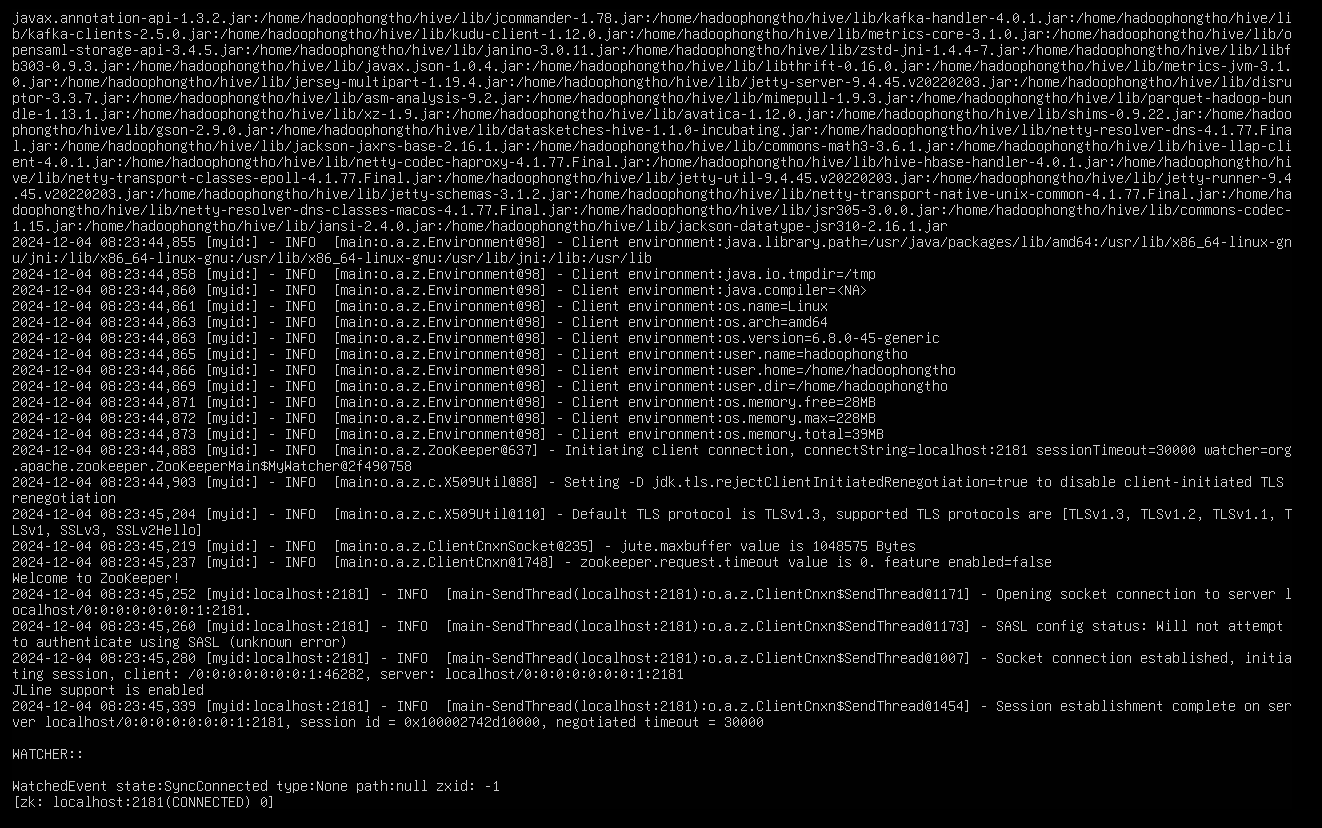


Thấy thông báo giống như sau:

Mode: standalone

* **Khởi động CLI:**

$ zkCli.sh



**Connecting to localhost:2181**: CLI đang kết nối đến ZooKeeper server chạy trên máy cục bộ (localhost) ở cổng 2181.

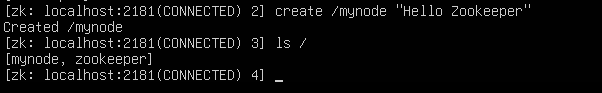
**CONNECTED**: Xác nhận rằng client đã kết nối thành công với server.

Sau khi kết nối thành công, có thể sử dụng các lệnh sau:

* **ls /**: Hiển thị các znodes ở cấp cao nhất (root).

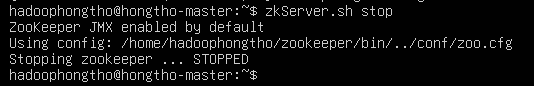


* **create /myNode "Hello ZooKeeper"**: Tạo một znode tên myNode với giá trị "Hello ZooKeeper".



* **Dừng máy chủ Zookeeper**

$ zkServer.sh stop



### 4.5. Thiết lập một cụm (cluster) gồm 3 máy chủ ZooKeeper

Apache ZooKeeper không phân biệt khái niệm master và slave như một số hệ thống khác. Thay vào đó, nó hoạt động theo cơ chế quorum trong một cụm các máy chủ đồng đẳng (peers). Một trong các máy chủ trong cụm được bầu chọn làm leader, và các máy còn lại sẽ là followers. Vai trò của các máy có thể thay đổi động khi leader gặp sự cố.

**Cấu hình ZooKeeper Trên Master và Slave (Khác vai trò)**

* **Master:** Dùng làm ZooKeeper cluster chính (quorum cluster).
* **Slave:** Chỉ làm client để kết nối đến cluster master.
* **Thiết lập IP tĩnh cho 3 máy Server:**

# vim /etc/netplan/50-cloud-init.yaml

Máy Server 1: 192.168.232.1/24

Máy Server 2: 192.168.232.4/24

Máy Server 3: 192.168.232.5/24

* **Đổi hostname ở Server 2 và Server 3 để phân biệt các server:**

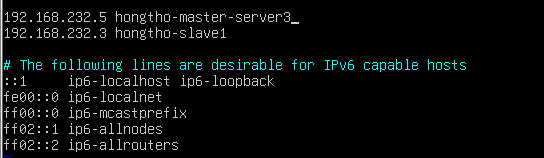
$ vim /etc/hostname





$ vim /etc/hosts





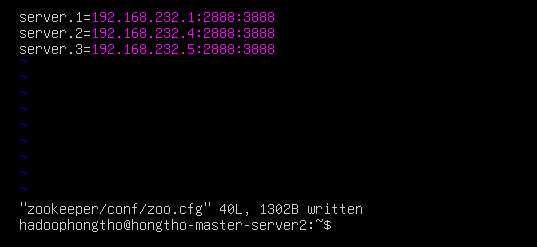
* **Cấu hình file zoo.cfg:**

Thêm vào file **zoo.cfg** các thiết lập một cụm ZooKeeper quorum trên các master node:

server.1=192.168.232.1:2888:3888 # Server 1

server.2=192.168.232.4:2888:3888 # Server 2

server.3=192.168.232.5:2888:3888 # Server 3



**Hai cổng sau tên máy chủ**: "2888" và "3888". Các máy chủ sử dụng cổng đầu tiên để kết nối với các máy chủ khác. Kết nối này cần thiết để các máy chủ có thể giao tiếp với nhau, ví dụ như để thỏa thuận về thứ tự cập nhật. Một máy chủ ZooKeeper sẽ sử dụng cổng này để kết nối các follower đến leader. Khi một leader mới xuất hiện, một follower sẽ mở kết nối TCP tới leader thông qua cổng này. Cổng thứ hai là để bầu cử leader.

* **Tạo tệp myid:**

Trong thư mục dữ liệu (/home/hadoophongtho/zookeeper/data), tạo tệp myid cho từng máy chủ:

$ echo "1" > /home/hadoophongtho/zookeeper/data/myid # Máy chủ 1





$ echo "2" > /home/hadoophongtho/zookeeper/data/myid # Máy chủ 2





$ echo "3" > /opt/zookeeper/data/myid # Máy chủ 3





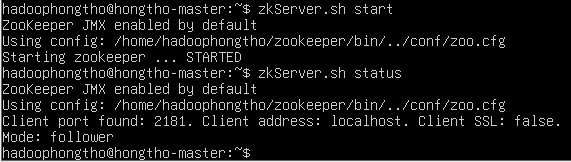
**Lưu ý:** ID trong tệp myid phải tương ứng với server.n trong tệp zoo.cfg.

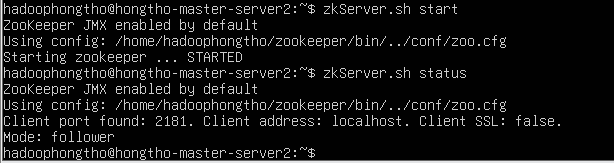
* **Kiểm tra trạng thái cụm:**

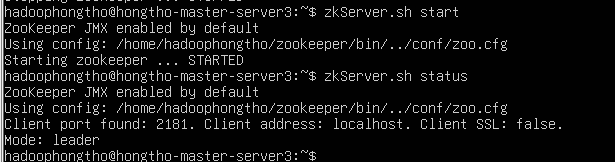
Chạy 2 lệnh sau trên mỗi máy chủ để khởi động ZooKeeper và kiểm tra trạng thái:

$ zkServer.sh start

$ zkServer.sh status







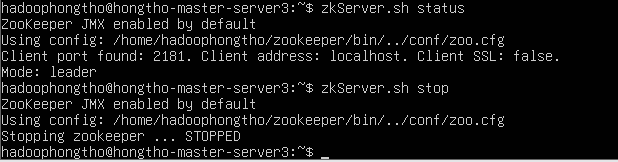
Trường hợp này Server 3 được bầu chọn làm leader.

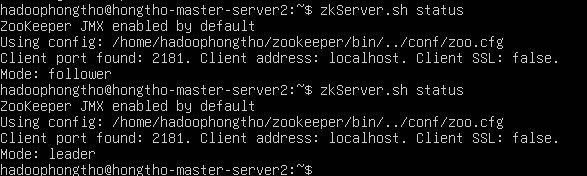
* **Dừng máy chủ ZooKeeper (nếu cần):**

Khi cần bảo trì, có thể dừng một máy chủ bằng lệnh:

$ zkServer stop

Khi dừng máy leader:



Các máy chủ còn lại sẽ tự động bầu chọn một máy chủ khác làm **leader**.

* + Máy chủ Server2 được bầu làm leader.
* **Kiểm tra kết nối**

**Ở máy client (ở đây dùng máy Slave) chạy lệnh kết nối sau để kết nối đến cụm**

$ zkCli.sh -server 192.168.232.1:2181,192.168.232.4:2182,192.168.232.5:2183

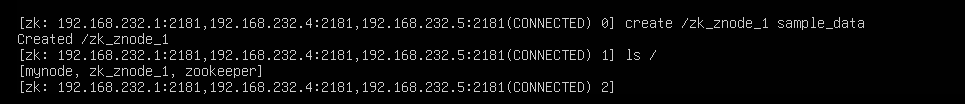


* **Kiểm tra quản lý Znode**

Tạo, liệt kê và sau đó xóa một znode:

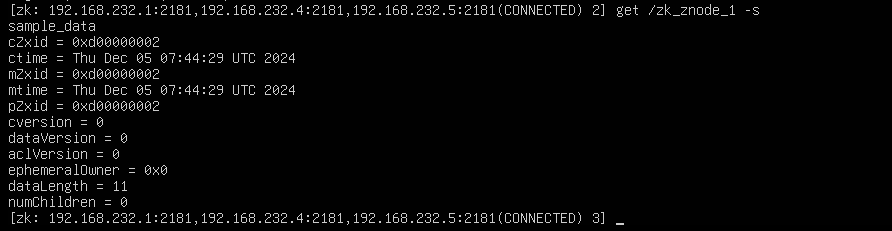
Tạo một znode tên là zk\_znode\_1 và gán chuỗi sample\_data cho nó:

$ create /zk\_znode\_1 sample\_data



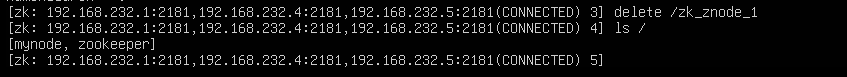
Lấy dữ liệu liên kết với nó:

$ get /zk\_znode\_1 -s



Xóa znode:

$ delete /zk\_znode\_1



Cài ZooKeeper cluster trên 2 máy là không khuyến nghị, mặc dù có thể thực hiện, vì nó không đảm bảo được tính fault tolerance (khả năng chịu lỗi). Lý do là ZooKeeper hoạt động dựa trên cơ chế quorum (đa số phiếu), và với 2 máy thì không thể có đa số nếu một máy gặp sự cố.

# CHƯƠNG 3: ỨNG DỤNG XÂY DỰNG HỆ THỐNG ĐỀ XUẤT PHIM

## 1. Recommendations với Mahout và ứng dụng trong hệ thống đề xuất phim

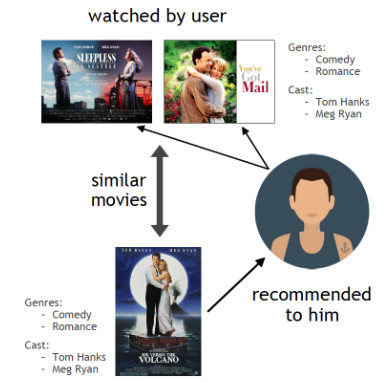
### 1.1. Giới thiệu về hệ thống đề xuất

Hệ thống đề xuất là kỹ thuật cung cấp những gợi ý về một sản phẩm, dịch vụ nào đó đang có nhu cầu được sử dụng trên internet. Những gợi ý này được cung cấp nhằm hỗ trợ người sử dụng trong quá trình ra quyết định lựa chọn sản phẩm, dịch vụ đó. Một vài ứng dụng nổi tiếng về hệ thống đề xuất này như: đề xuất sản phẩm của Amazon.com, đề xuất video của YouTube hay các bộ phim của Netflix… không chỉ giải quyết vấn đề cá nhân hóa trải nghiệm người dùng mà còn tăng doanh thu thông qua việc thúc đẩy người dùng mua sắm hoặc tiêu thụ nội dung nhiều hơn.

Có 3 phương pháp tiếp cận khi thực hiện hệ thống đề xuất:

#### 1.1.1. Content-based filtering

Content-Based Filtering (lọc dựa trên nội dung) là một phương pháp đưa ra các đề xuất dựa trên đặc điểm của sản phẩm hoặc nội dung mà người dùng đã tương tác trước đó. Phương pháp này không yêu cầu dữ liệu từ người dùng khác mà tập trung vào sự phù hợp giữa đặc điểm của sản phẩm và sở thích cá nhân của người dùng để đưa ra gợi ý.



Hình 9. Mô tả Content-based filtering

#### 1.1.2. Collaborative filtering

Collaborative Filtering (lọc cộng tác) dựa trên khái niệm về sự tương đồng giữa các sản phẩm và người dùng. Không giống như lọc nội dung, phương pháp này không xem xét các đặc điểm của sản phẩm hay thuộc tính của người dùng. Phương pháp này sử dụng dữ liệu lịch sử (như lượt xem, nhấp chuột, đánh giá, mua hàng) để hình thành các gợi ý.

**Nguyên lý hoạt động:**

* **Dựa trên sự tương đồng:**

Xác định mức độ tương đồng giữa các sản phẩm hoặc người dùng dựa trên dữ liệu hành vi, như lượt xem, nhấp chuột, đánh giá, lịch sử mua hàng,…)

* **Tạo nhóm lân cận (Neighborhood):**

Tạo các nhóm sản phẩm hoặc người dùng có hành vi hoặc đánh giá tương tự nhau.

Ví dụ: "Người dùng A và B đều thích các sản phẩm X, Y => Có khả năng A sẽ thích sản phẩm Z mà B đã mua."

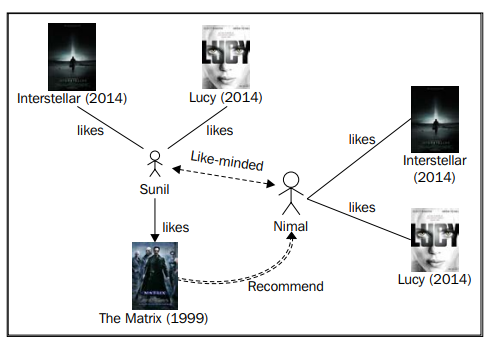
* **Gợi ý dựa trên nhóm lân cận:**

Khi người dùng tương tác với một sản phẩm, hệ thống sẽ đưa ra các sản phẩm khác từ nhóm lân cận (những sản phẩm mà người dùng tương tự đã quan tâm).

**Các phương pháp lọc cộng tác:**

* User-Based Collaborative Filtering**:** Tìm các người dùng có hành vi hoặc đánh giá tương tự.

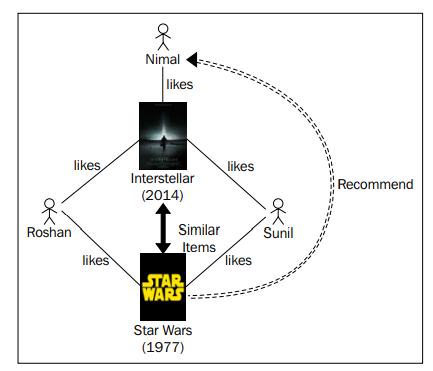
Ví dụ: "Người dùng A thích xem các bộ phim giống người dùng B, vậy các bộ phim mà B thích nhưng A chưa xem sẽ được gợi ý cho người dùng A."



Hình 10. Mô tả User-Based Collaborative Filtering

* Item-Based Collaborative Filtering: Tìm các sản phẩm tương tự dựa trên hành vi của người dùng.

Ví dụ: "Người dùng A xem bộ phim X, vậy hệ thống gợi ý bộ phim Y vì những người xem X cũng thường xem Y."



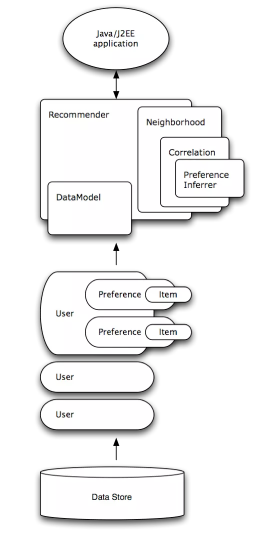
Hình 11. Mô tả Item-Based Collaborative Filtering

#### 1.1.3. Hybrid filtering

Hybrid filtering là giải pháp hiệu quả và toàn diện, tận dụng ưu điểm của cả lọc nội dung và lọc cộng tác, mang lại gợi ý chất lượng hơn cho người dùng trong các hệ thống

### 1.2. Recommendations trong Apache Mahout

#### 1.2.1. Recommendation Architecture



Hình 12. Recommendation Architecture

DataModel: là giao diện để truy xuất thông tin về sở thích của người dùng, đại diện cho cách chúng ta đọc dữ liệu từ các nguồn dữ liệu khác nhau.

Apache Mahout hỗ trợ các phương pháp nhập dữ liệu sau:

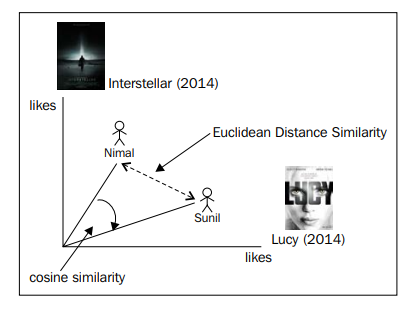
* **FileDataModel:** nhận đầu vào là file CSV
* JDBCDataModel**:** đọc dữ liệu từ JDBC Driver
* GenericDataModel**:** điền dữ liệu thông qua các lệnh gọi Java.
* GenericBooleanPrefDataModel**:** sử dụng dữ liệu người dùng đã cho, phù hợp cho các thử nghiệm nhỏ.

Bất kể từ nguồn nào, dữ liệu luôn có chung một cách triển khai trong Mahout đó là ánh xạ vào đối tượng Preference, là một triple(user, item, score) được lưu trong UserPreferenceArray.

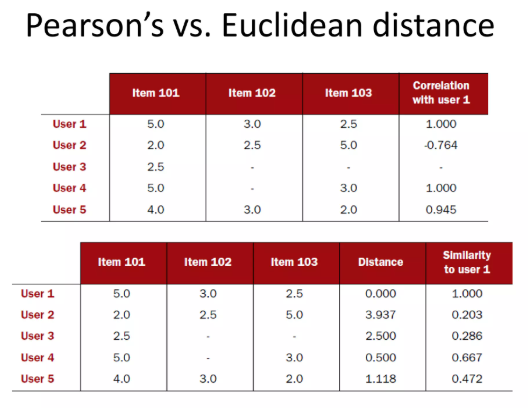
* **The similarity mesure:**

Độ tương đồng thể hiện mức độ giống nhau, UserSimilarity tương tác giữa hai người dùng trong hệ thống gợi ý user-based hoặc ItemSimilarity giữa hai sản phẩm trong hệ thống gợi ý item-based.

Hai người dùng có thể được coi là giống nhau nếu khoảng cách hoặc góc giữa họ trong không gian người dùng – sản phẩm là nhỏ.



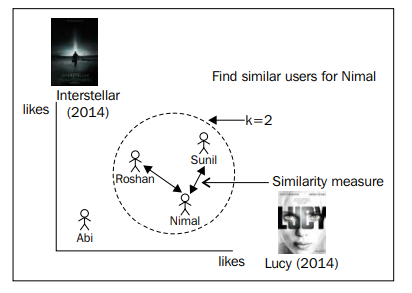
Hình 13. Sự khác biệt giữa độ tương đồng khoảng cách Euclidean và độ tương đồng Cosine



Hình 14. Hai phương pháp đo lường sự tương đồng: hệ số tương quan Pearson và khoảng cách Euclidean

* **UserNeighborhood:**

Khi sử dụng thuật toán "neighborhood", chúng ta có thể tính toán **"vùng lân cận"** của các người dùng đối với một người dùng cụ thể. Vùng lân cận này sẽ được sử dụng để đưa ra các gợi ý.



Hình 15. Thuật toán K-Nearest Neighbors

* Nearest Neighbour Algorithm:

Thuật toán này tính toán vùng lân cận bao gồm **N người dùng gần nhất** so với một người dùng được cho.

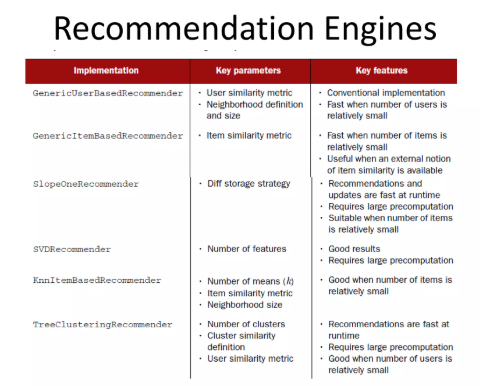
Người dùng gần nhất được xác định dựa trên độ tương đồng (similarity) giữa họ.

* ThresholdUserNeighborhood:

Thuật toán này tính toán vùng lân cận bao gồm tất cả những người dùng có độ tương đồng với người dùng được cho **bằng hoặc vượt qua một ngưỡng (threshold)** được xác định trước.

* Recommender:

Dựa trên DataModel, vùng lân cận (UserNeighborhood) và độ tương đồng (Similarity), hệ thống gợi ý Apache Mahout được chọn sẽ ước tính các mục tương tự cho những mục chưa được nhìn thấy hoặc mới đối với người dùng.



Hình 16. Cơ chế Recommendation của Mahout

#### 1.2.2. Evaluation techniques

Có hai phương pháp khác nhau để đánh giá chất lượng của một hệ thống gợi ý trên một tập dữ liệu cụ thể, bao gồm:

* Dựa trên dự đoán (Prediction-based): Đánh giá mức độ chính xác của các dự đoán mà hệ thống đưa ra.
* Mean Average Error
* RMSE (Root Mean Square Error)
* Dựa trên Truy xuất Thông tin (Information Retrieval, IR-based): Đánh giá dựa trên các chỉ số của lĩnh vực truy xuất thông tin, chẳng hạn như độ chính xác (precision), độ thu hồi (recall), hoặc F1-score.

Trong quá trình đánh giá, tập dữ liệu được chia thành tập huấn luyện (training dataset) và tập kiểm tra (test dataset). Tập huấn luyện được sử dụng để xây dựng mô hình, và việc đánh giá được thực hiện dựa trên tập kiểm tra.

## 2. Tổng quan quy trình của hệ thống đề xuất phim

Dưới đây là quy trình tổng thể xây dựng hệ thống đề xuất phim theo các tầng:

**1. Data Storage Layer**

**Input:** Tập tin **ratings.csv** chứa các đánh giá của người dùng đối với các bộ phim. Dữ liệu này sẽ được lưu trữ trong HDFS để phục vụ việc xử lý dữ liệu phân tán.

**Lưu trữ kết quả gợi ý:** Sau khi mô hình gợi ý được xây dựng, các kết quả gợi ý từ Mahout sẽ được lưu trữ trong HBase để truy xuất nhanh chóng, hỗ trợ khả năng mở rộng và truy vấn hiệu quả.

**2. Data Processing Layer**

Sử dụng MapReduce để xử lý dữ liệu phân tán trên HDFS, tính toán độ tương đồng giữa các bộ phim và xây dựng mô hình gợi ý.

YARN sẽ đảm nhận việc phân phối và quản lý tài nguyên giữa các công việc MapReduce, đảm bảo rằng các tác vụ tính toán và xử lý dữ liệu được thực hiện hiệu quả và tối ưu.

1. **Data Access Layer**

Sqoop sẽ đảm nhận việc nhập và xuất dữ liệu giữa hệ thống cơ sở dữ liệu quan hệ (MySQL) và hệ sinh thái Hadoop. Dữ liệu người dùng và thông tin phim sẽ được lưu trữ trong MySQL và sử dụng Sqoop để nhập dữ liệu từ MySQL vào HDFS.

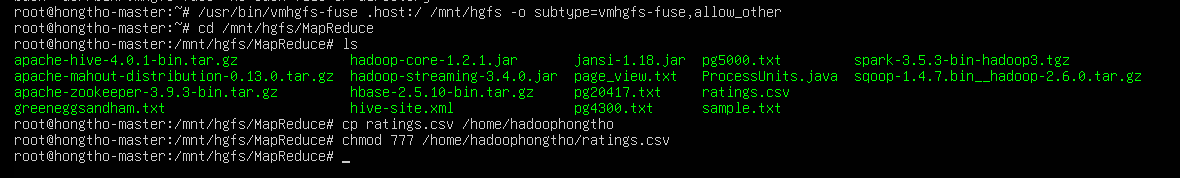
Mahout sẽ được sử dụng trong tầng này để thực hiện thuật toán gợi ý.

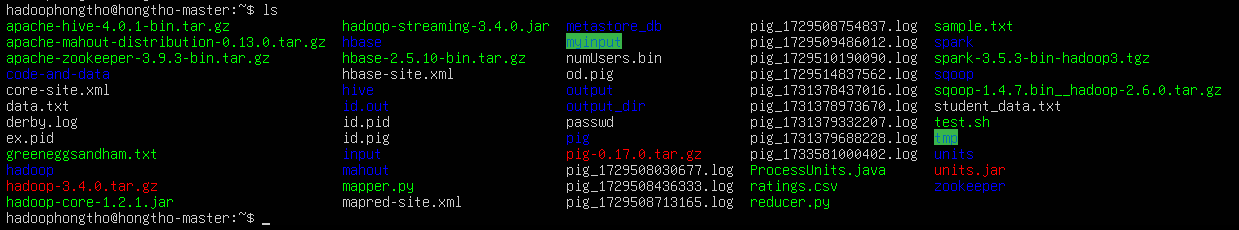
**4. Data Management Layer**

Zookeeper sẽ đảm nhận nhiệm vụ đồng bộ hóa giữa các nút trong cụm Hadoop, đảm bảo các tác vụ được thực hiện chính xác và có tổ chức, đồng thời quản lý các tác vụ trong môi trường phân tán, tránh xung đột và giúp các quá trình tính toán và lưu trữ diễn ra trơn tru.

## 3. Xây dựng hệ thống đề xuất phim với tập dữ liệu ratings.csv

### 3.1. Đưa tập dữ liệu ratings.csv vào home của hadoop



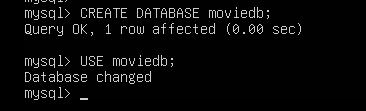


### 3.2. Chuyển dữ liệu ratings.csv vào MySQL

* **Tạo databaes moviedb;**

CREATE DATABASE moviedb;

USE moviedb;



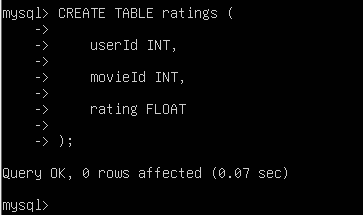
CREATE TABLE ratings (

userId INT,

movieId INT,

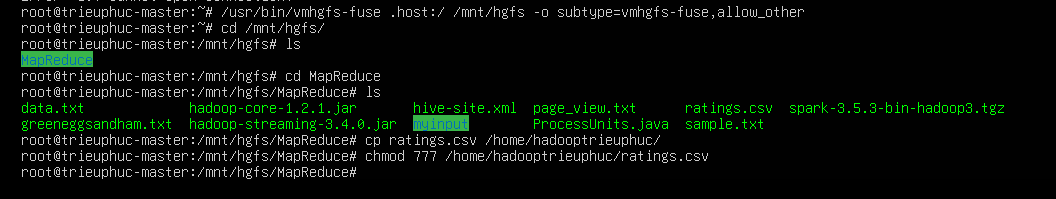
rating FLOAT

);



* **Đưa file csv vào /home/hadooptrieuphuc**

$ /usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow\_other



* **Kiểm tra thư mục được chỉ định bởi --secure-file-priv**
* **Mở MySQL và chạy lệnh sau:**

SHOW VARIABLES LIKE 'secure\_file\_priv';

Kết quả sẽ là một thư mục:

+------------------+-----------------------+

| Variable\_name | Value |

+------------------+-----------------------+

| secure\_file\_priv | /var/lib/mysql-files/ |

+------------------+-----------------------+

* **Di chuyển file ratings.csv vào thư mục này:**

$ sudo mv /home/hadooptrieuphuc/ratings.csv /var/lib/mysql-files/

* **Sau đó, chạy lệnh LOAD DATA INFILE:**

LOAD DATA INFILE '/var/lib/mysql-files/ratings.csv'

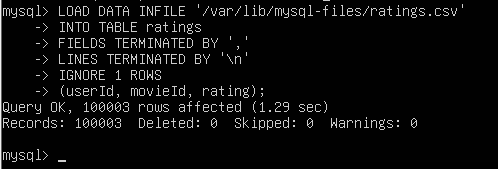
INTO TABLE ratings

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(userId, movieId, rating);



### 3.3. Imort dữ liệu từ MySQL sang HDFS – Tạo input dữ liệu cho mahout

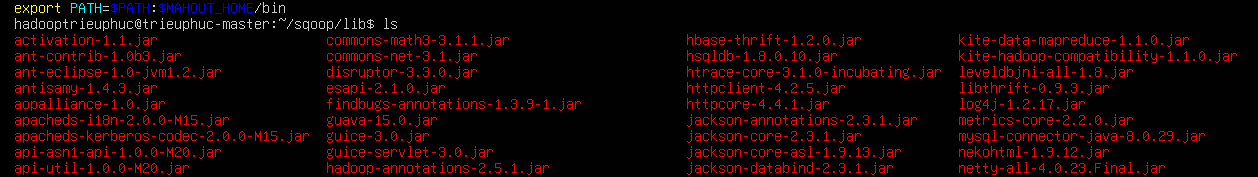
* **Cài đặt mysqlconnector và bỏ vào folder lib của sqoop**

$ wget <https://ftp.ntu.edu.tw/MySQL/Downloads/Connector-J/mysql-connector-java-8.0.29.tar.gz>

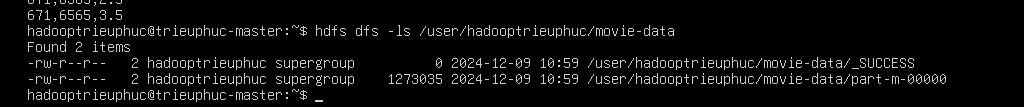
$ tar -zxf mysql-connector-java-8.0.29.tar.gz

$ cd mysql-connector-java-8.0.29/

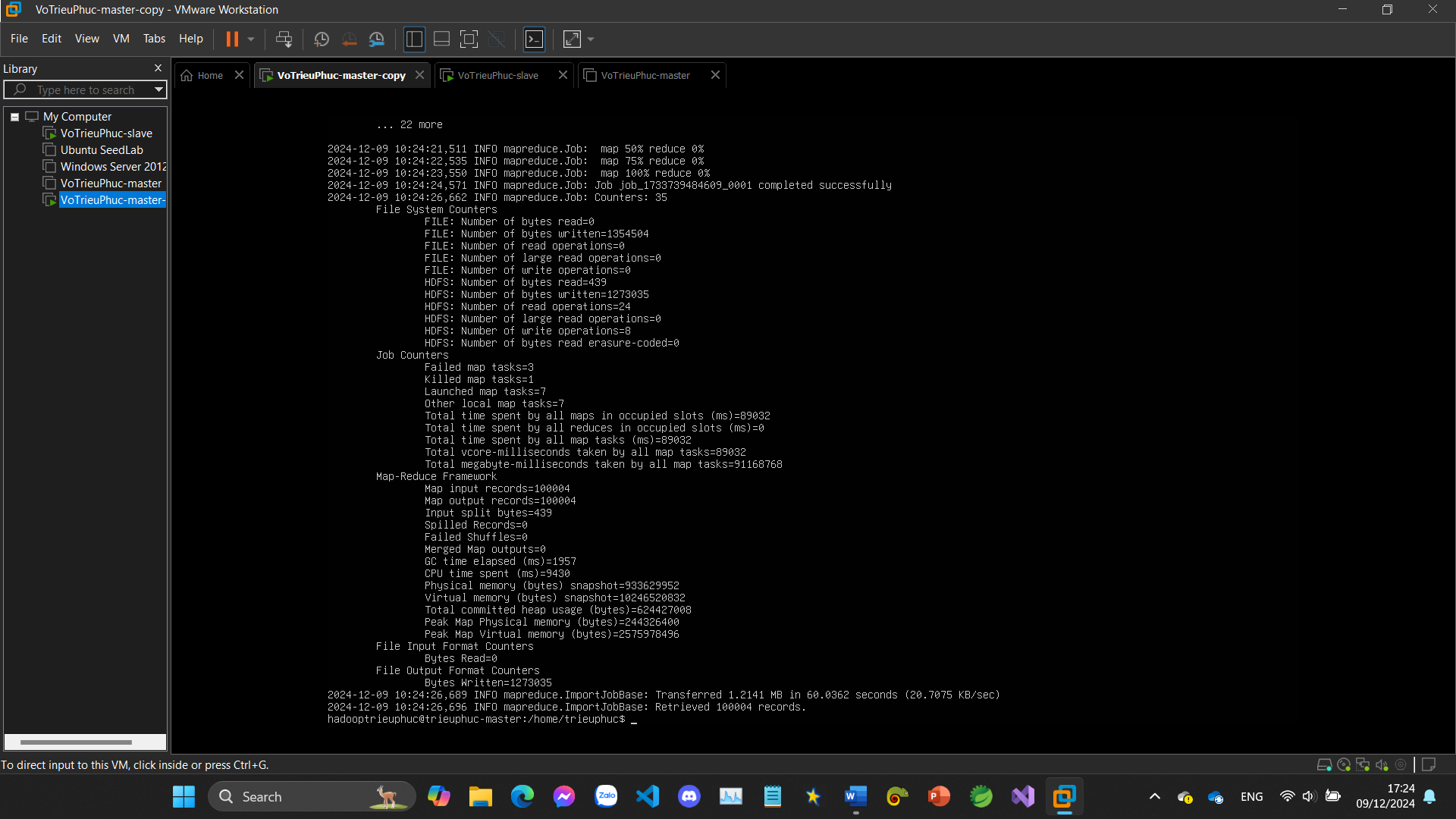
$ mv mysql-connector-java-8.0.29.jar /home/hadooptrieuphuc/sqoop/lib



* **Tiến hành imort từ MySQL sang HDFS ( Kết quả sẽ là thư mục movie-data sẽ chứa file part-m-00000 dùng làm input cho mahout)**



$ sqoop import --connect jdbc:mysql://localhost:3306/moviedb --username root --password root --table ratings --target-dir /user/hadooptrieuphuc/movie-data –split-by userId --columns "userId,movieId,rating" --num-mappers 1



### 3.4. Sử dụng Mahout và MapReduce để xây dựng mô hình gợi ý

* **Sau khi có tập dữ liệu, khởi động hệ thống hadoop:**

$ start-dfs.sh

$ start-yarn.sh

* **Kiểm tra datanode:**

$ hdfs dfsadmin -report

* **Tạo thư mục trong hdfs và copy file ratings.csv vào:**

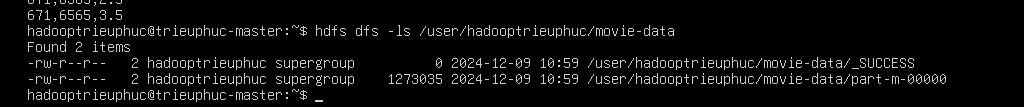
$ hdfs dfs -mkdir /user

$ hdfs dfs -mkdir /user/hadoopnguyenhoang

$ hdfs dfs -mkdir /user/hadoopnguyenhoang/movie-data

$ hadoop fs -put ratings.csv /user/hadoopnguyenhoang/movie-data

* **Hoặc có thể dùng file part-m-00000 để làm input dữ liệu cho việc xây dựng mô hình:**

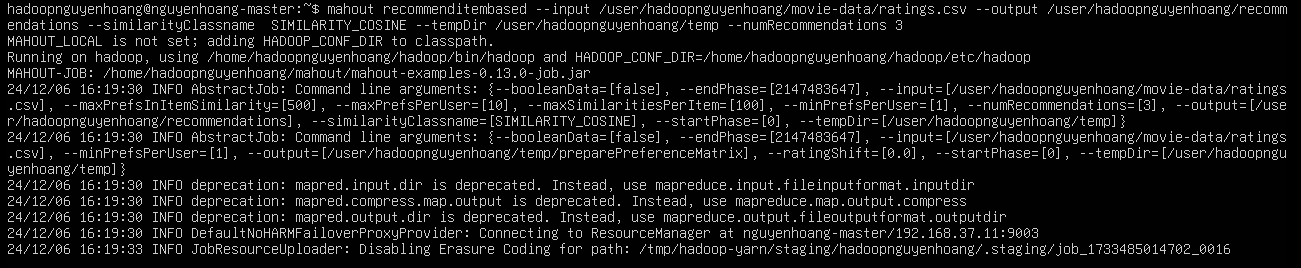


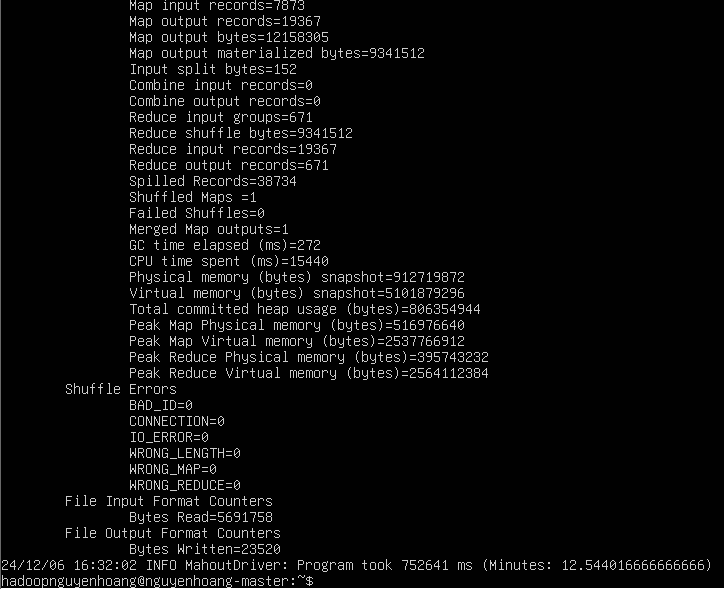
* **Sử dụng lệnh sau để tạo mô hình và gợi ý:**

$ mahout recommenditembased --input /user/hadoopnguyenhoang/movie-data/ratings.csv --output /user/hadoopnguyenhoang/recommendations --similarityClassname SIMILARITY\_COSINE --tempDir /user/hadoopnguyenhoang/temp --numRecommendations 3

Giải thích các tham số:

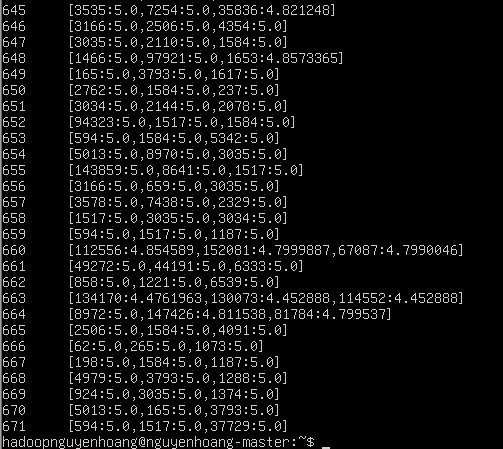
* --input /user/hadoopnguyenhoang/movie-data/ratings.csv: Chỉ định đường dẫn HDFS của file dữ liệu đầu vào chứa các đánh giá (ratings) với định dạng userID,itemID,rating.
* --output /user/hadoopnguyenhoang/recommendations: Chỉ định thư mục HDFS để lưu kết quả gợi ý (recommendations).
* --similarityClassname SIMILARITY\_COSINE: Sử dụng **Cosine Similarity** để tính độ tương đồng giữa các mục (items).
* --tempDir /user/hadoopnguyenhoang/temp: Thư mục tạm trong HDFS để lưu các file tạm khi xử lý.
* --numRecommendations 3: Số lượng gợi ý được tạo ra cho mỗi người dùng là **3 mục.**





* Sau khi chạy lệnh trên, Mahout sẽ tạo ra các gợi ý và lưu kết quả trong thư mục recommendations. Kiểm tra các tệp kết quả recommendations/part-r-00000 để xem các gợi ý sản phẩm cho từng người dùng.

$ hdfs dfs -cat recommendations/part-r-00000



### 3.5. Quá trình MapReduce

Cụ thể, khi sử dụng lệnh Mahout recommenditembased, Mahout sẽ tự động triển khai các tác vụ MapReduce để:

1. **Map Phase**: Trong giai đoạn "Map", mỗi bộ phim và đánh giá của người dùng sẽ được chia nhỏ thành các cặp (key, value) và được phân phối cho các nút trong hệ thống Hadoop. Việc phân phối này giúp xử lý đồng thời các bộ phim và đánh giá từ nhiều người dùng.
2. **Reduce Phase**: Giai đoạn "Reduce" sẽ kết hợp và tính toán độ tương đồng giữa các bộ phim dựa trên các đánh giá người dùng. Mahout sử dụng các thuật toán như **Cosine Similarity** để tính toán độ tương đồng giữa các bộ phim. Sau đó, kết quả của quá trình tính toán được gộp lại để tạo ra các gợi ý cho người dùng.

Kết quả cuối cùng của bước này là các gợi ý phim cho từng người dùng, được lưu trữ trong thư mục HDFS **./recommendations/.**

Sau khi chạy Mahout và nhận được kết quả gợi ý, cần viết một chương trình Java để nhập các đề xuất vào HBase.

### 3.6. Chuyển kết quả vào tập dữ liệu trong HBase

* **Kiểm tra file bashrc để ý tên classpath và đường dẫn:**



* **Tạo thư mục movie-recommender và chuẩn bị file ImportRecommendations.java:**

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.HBaseConfiguration;

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.TableName;

import org.apache.hadoop.hbase.util.Bytes;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class ImportRecommendations {

    public static void main(String[] args) throws IOException {

        if (args.length < 1) {

            System.err.println("Usage: ImportRecommendations <input-file>");

            System.exit(-1);

        }

        String inputFile = args[0];      //Yêu cầu một đường dẫn tới file đầu vào qua args[0].

        Configuration config = HBaseConfiguration.create();

        try (Connection connection = ConnectionFactory.createConnection(config);// Mở kết nối với HBase.

                Table table = connection.getTable(TableName.valueOf("user\_recommendations")); //Truy cập bảng user\_recommendations.

                BufferedReader reader = new BufferedReader(new FileReader(inputFile))) { //Đọc file đầu vào từng dòng một.

            String line;

            while ((line = reader.readLine()) != null) {

                String[] parts = line.split("\\s+");

                if (parts.length < 2)

                    continue;

                  // Đọc từng dòng từ file và chia thành hai phần: userId (khóa hàng) và recommendations (chuỗi đề xuất phim).

                String userId = parts[0];

                String recommendations = parts[1].replaceAll("[\\[\\]]", ""); //Bỏ dấu ngoặc vuông khỏi chuỗi đề xuất bằng .replaceAll("[\\[\\]]", "").

                Put put = new Put(Bytes.toBytes(userId)); // Tạo đối tượng chứa dữ liệu để chèn vào HBase.

                String[] movies = recommendations.split(",");

                for (int i = 0; i < Math.min(3, movies.length); i++) { // Limit to 3 movies

                    String movieId = movies[i].split(":")[0]; // Extract movie\_id

                    put.addColumn(

                            Bytes.toBytes("recommendations"),

                            Bytes.toBytes("recommend" + (i + 1)),

                            Bytes.toBytes(movieId));

              }//   addColumn: Chèn giá trị vào:

                // Column family recommendations.

                // Column qualifier recommend1, recommend2, ...

                // Giá trị tương ứng là movieId.

                table.put(put);

            }

            System.out.println("Data imported successfully.");

        }

    }

}

* **Chuyển ImportRecommendations.java và ratings.csv vào thư mục movie-recommender:**



* **Biên dịch mã nguồn Java từ file ImportRecommendations.java và tạo JAR**

$ javac -d . ImportRecommendations.java

**Ý nghĩa**: Dùng để biên dịch mã nguồn Java từ file **ImportRecommendations.java** thành file **.class** có thể chạy được.

**Tham số:**

* -d : Đặt đường dẫn của các file .class đầu ra ( đường dẫn ở đây là . )
* ImportRecommendations.java: File nguồn chứa mã Java cần biên dịch.

Sau đó, chạy tiếp lệnh:

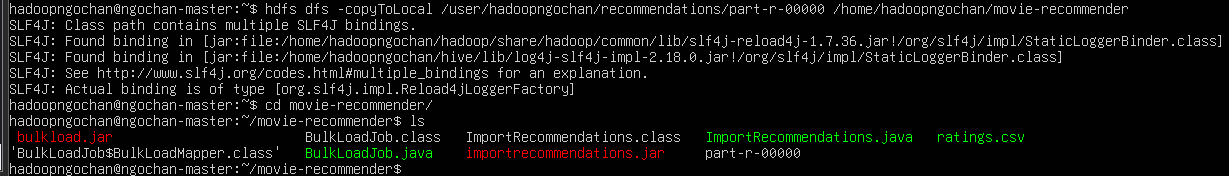
$ jar cf importrecommendations.jar ImportRecommendations.class

**Ý nghĩa**: Tạo file JAR (Java ARchive) từ file lớp (.class) vừa biên dịch.

**Tham số:**

* cf:
  + c: Tạo file JAR mới.
  + f: Chỉ định tên file JAR.
* importrecommendations.jar: Tên file JAR sẽ được tạo.
* ImportRecommendations.class: Tên file class sẽ được thêm vào file JAR.





* **Tạo bảng user\_recommendations:**



* **Sao chép file kết quả chạy từ Mahout part-r-00000 từ HDFS về hệ thống file cục bộ (vào thư mục movie-recommender)**

Sau khi Mahout tạo ra các gợi ý và lưu vào HDFS, cần sao chép kết quả về hệ thống cục bộ để xử lý với chương trình Java.

$ hdfs dfs -copyToLocal /user/hadoopngochan/recommendations/part-r-00000 /home/hadoopngochan/movie-recommender

* **Chạy chương trình Java với file JAR vừa tạo**

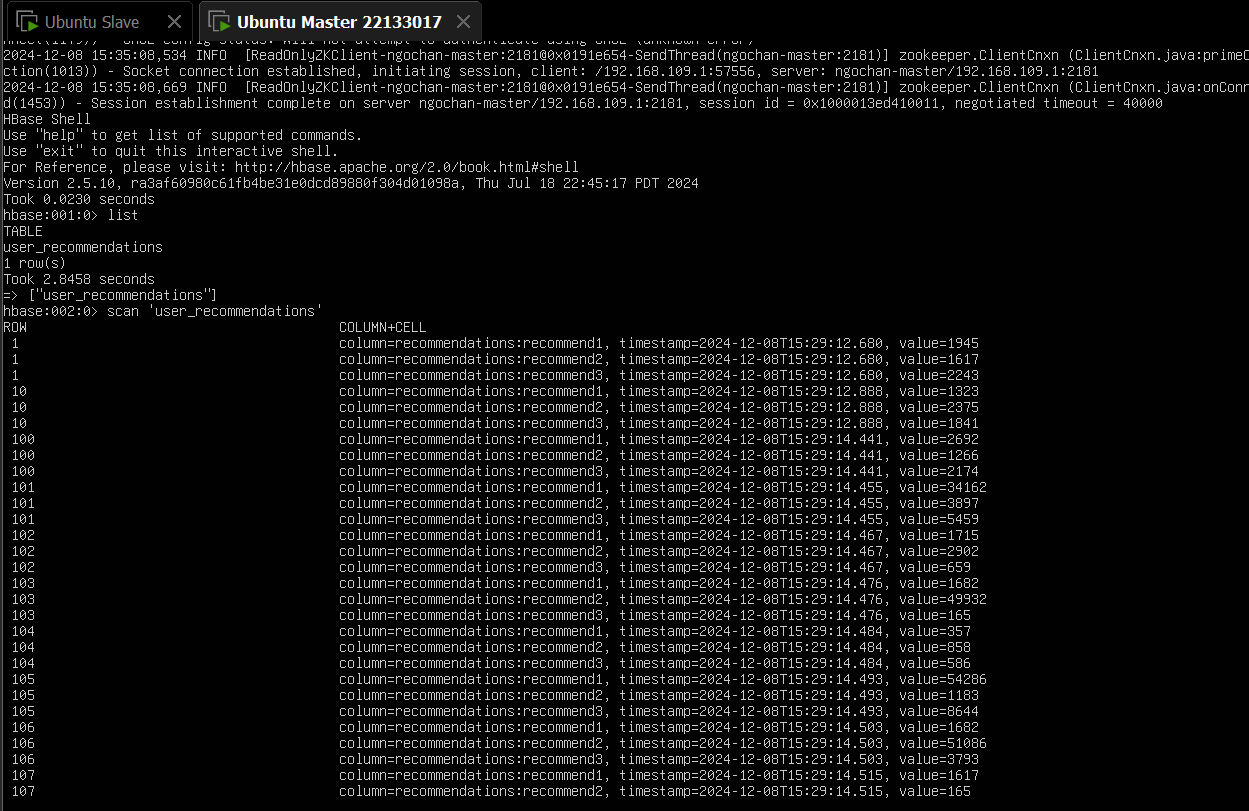
$ java -verbose:class -classpath $CLASSPATH:importrecommendations.jar ImportRecommendations ./part-r-00000

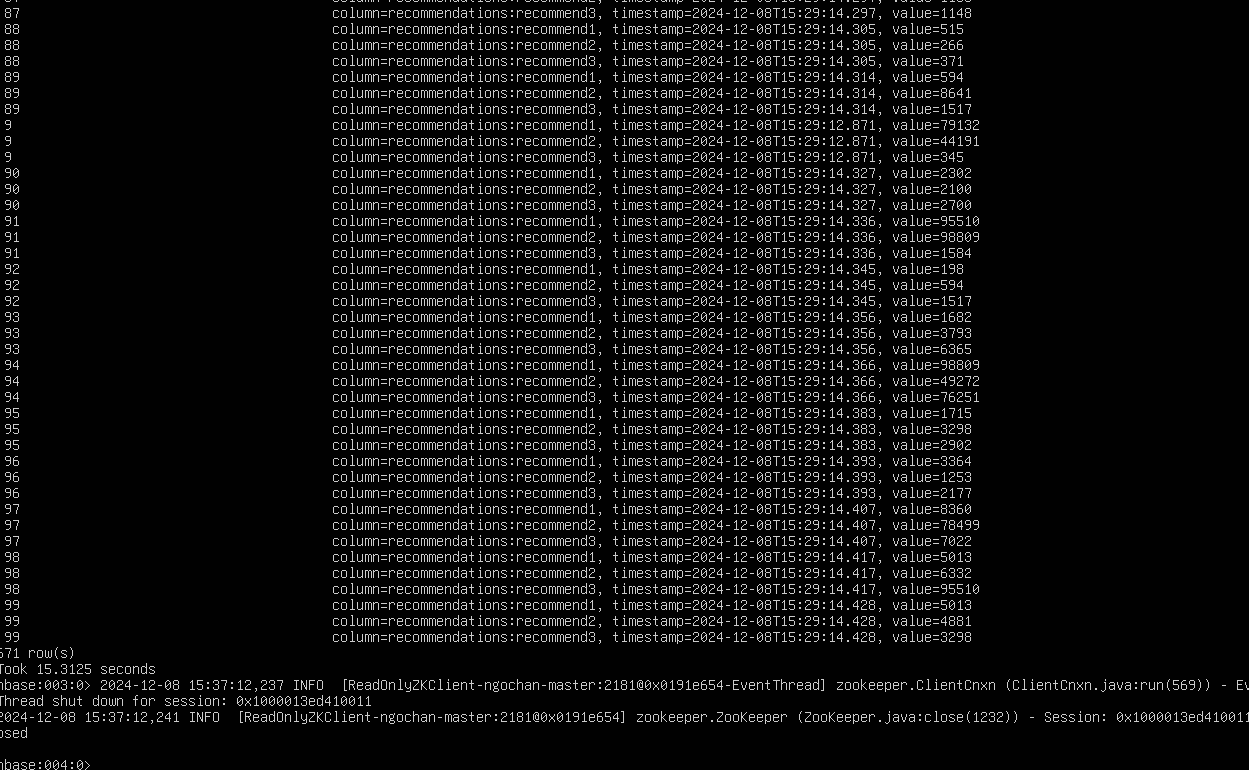
**Tham số:**

* -classpath $CLASSPATH:importrecommendations.jar: Thêm file JAR vào đường dẫn classpath.
  + $CLASSPATH: Biến môi trường chứa các thư mục hoặc file JAR đã được định nghĩa từ trước.
  + importrecommendations.jar: File JAR chứa lớp cần chạy.
* ImportRecommendations: Tên lớp chứa phương thức main() để chạy chương trình.
* ./part-r-00000: file chứa dữ liệu đầu vào được truyền vào chương trình.



**Kết quả:**

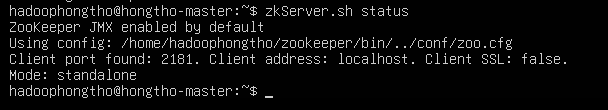




### 3.7. Zookeeper trong Recommendation Systems

Sau khi dữ liệu từ Mahout được tính toán và chuẩn bị, Zookeeper sẽ giúp điều phối quá trình ghi dữ liệu vào HBase. Với chế độ standalone, Zookeeper sẽ đảm nhiệm việc đồng bộ hóa các thao tác giữa các node trong hệ thống, đảm bảo tính nhất quán khi dữ liệu được đưa vào các bảng HBase.





Trong chế độ standalone, Zookeeper sẽ hoạt động với một máy chủ duy nhất, giúp giảm thiểu độ phức tạp trong việc triển khai và quản lý, đồng thời vẫn đảm bảo hiệu quả trong việc điều phối các thao tác đồng bộ hóa với HBase.

# KẾT LUẬN

Việc xây dựng hệ thống đề xuất phim sử dụng dữ liệu đánh giá người dùng đã đạt được mục tiêu tạo ra các gợi ý chính xác và cá nhân hóa cho người dùng thông qua việc kết hợp các công nghệ phân tán mạnh mẽ của Hadoop. Quy trình hệ thống được thiết kế một cách có hệ thống từ việc lưu trữ và xử lý dữ liệu trong HDFS, thực hiện các tác vụ tính toán phân tán bằng MapReduce, cho đến việc sử dụng Mahout để tạo các mô hình gợi ý.

Quá trình triển khai sử dụng YARN để tối ưu hóa phân phối tài nguyên và Zookeeper để đảm bảo tính đồng bộ giữa các nút trong hệ thống phân tán, giúp hệ thống hoạt động ổn định và hiệu quả. Sau khi mô hình gợi ý được xây dựng, kết quả được chuyển vào HBase để truy xuất nhanh chóng, hỗ trợ khả năng mở rộng của hệ thống.

Để đảm bảo khả năng nhập và xuất dữ liệu dễ dàng, hệ thống cũng tích hợp Sqoop để chuyển dữ liệu giữa các hệ thống cơ sở dữ liệu quan hệ MySQL và Hadoop. Dữ liệu gốc từ MySQL được chuyển sang HDFS để xử lý, trong khi các kết quả gợi ý được lưu trữ và truy xuất từ HBase.

Cuối cùng, thông qua việc triển khai chương trình Java, các kết quả từ Mahout được nhập vào HBase để cung cấp các gợi ý phim chính xác, đáp ứng nhu cầu người dùng và đảm bảo tính linh hoạt, khả năng mở rộng trong môi trường dữ liệu lớn.

# TÀI LIỆU THAM KHẢO

1. Apache HBase Team, *Apache HBase Reference Guide*, [https://hbase.apache.org/ apache\_hbase\_reference\_guide.pdf](https://hbase.apache.org/apache_hbase_reference_guide.pdf).
2. Deepak Vohra (2016), *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools*, Apress.
3. GeeksforGeeks (2023), *Architecture of HBase*, [https://www.geeksforgeeks.org/ architecture-of-hbase/](https://www.geeksforgeeks.org/architecture-of-hbase/).
4. Olive Zhao (2022), *Introduction to HBase*, [https://forum.huawei.com/enterprise /en/HCIA-Big-Data-Introduction-to-HBase/thread/667260233535340544-667213860102352896](https://forum.huawei.com/enterprise/en/HCIA-Big-Data-Introduction-to-HBase/thread/667260233535340544-667213860102352896).
5. Jayani Withanawasam (2015), *Apache Mahout Essentials: Implement top-notch machine learning algorithms for classification, clustering, and recommendations with Apache Mahout.*
6. Meey Land, *“Phần mềm Apache Mahout là gì?” Thông tin cơ bản và ứng dụng trong xử lý dữ liệu mã nguồn mở*, <https://meeyland.com/tin-tuc/phan-mem-apache-mahout-la-gi-thong-tin-co-ban-va-ung-dung-trong-xu-ly-du-lieu-ma-nguon-mo-110378159476>, truy cập ngày 04/12/2024.
7. Cataldo Musto (2013), *Tutorial Mahout – Recommendation,* [*https://fr.slideshare.net/slideshow/tutoria-mahout recommendation/15946627#20*](https://fr.slideshare.net/slideshow/tutoria-mahout%20recommendation/15946627#20)*,* truy cập ngày 04/12/2024.
8. Marco Ghisellini (2014), *Mahout and Hadoop: a simple integration,* [*https://techannotation.wordpress.com/2014/04/03/mahout-and-hadoop-a-simple-integration/*](https://techannotation.wordpress.com/2014/04/03/mahout-and-hadoop-a-simple-integration/)*,* truy cập ngày 04/12/2024.
9. The Apache Software Foundation (2022), *ZooKeeper Programmer's Guide*, [zookeeper.apache.org](https://zookeeper.apache.org), <https://zookeeper.apache.org/doc/r3.1.2/zookeeperProgrammers.html#_introduction>, truy cập ngày 01/12/2024.
10. Tutorialspoint, *Zookeeper Tutorial*, <https://www.tutorialspoint.com/zookeeper/index.htm>, *Tutorialspoint.com,* truy cập ngày 05/12/2024.
11. Flavio Junqueira & Benjamin Reed (11/2013), *ZooKeeper*, First Edition, O’Reilly Media.
12. GeeksforGeeks (2024), *Hadoop Ecosystem,* <https://www.geeksforgeeks.org/hadoop-ecosystem/>, truy cập ngày 02/12/2024.
13. Anushree Raj1, Rio D’Souza (2016), *A Review on Hadoop Eco System for Big Data*, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, [Technoscience Academy](https://technoscienceacademy.com/).