

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



MÔN HỌC

CƠ SỞ DỮ LIỆU PHÂN TÁN

ĐỀ TÀI TIỂU LUẬN:

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU HDOOP/ HBASE

GVHD: Thầy Nguyễn Hồ Duy Tri

NHÓM: Chiến đồ án

THÀNH VIÊN:

Nguyễn Trương Đình Giang - 21520215

Đỗ Hiền Thảo - 21520460

Nguyễn Dương Chí Tâm - 21520439

Huỳnh Mạnh Huy - 21520259

TP HCM, Ngày 18 Tháng 12 Năm 2023
MỤC LỤC

I. Giới thiệu	4
1. Giới thiệu về HBase	4
2. Lịch sử về Hbase	5
3. Đặc điểm của HBase	6
4. Kiến trúc của HBase	6
5. Data flow trong HBase	8
6. HBase so với RDBMS và NoSQL khác	9
7. HBase Shell	10
8. Tổng quan về Hadoop	11
9. HDFS là gì?	12
10. Cách thức hoạt động của HDFS	13
II. Hướng dẫn cài đặt	14
1. Cài đặt trên một máy	14
1.1. Chế độ Standalone mode	14
1.2. Chế độ Pseudo-Distributed mode	15
2. Cài đặt trên một cụm máy phân tán	25
2.1. Chế độ Fully Distributed mode	25
III. Tài liệu tham khảo	26

LỜI CẢM ƠN

Trong chặng đường của cuộc sống, có lẽ ai cũng đã từng trải qua các cung bậc cảm xúc khác nhau, có thành công cũng có thất bại, dẫu thế đó cũng là thành quả của cá nhân cũng như tập thể. Và đằng sau đó chính là sự giúp đỡ từ mọi người xung quanh. Hôm nay, để có thể hoàn thành được đồ án môn học này, nhóm chúng em rất biết ơn quý thầy cô đã hỗ trợ tận tình, cung cấp cho chúng em kiến thức cũng như lời khuyên sâu sắc.

Lời đầu tiên, chúng em xin chân thành cảm ơn đến toàn thể giảng viên trường Đại học Công nghệ Thông tin – Đại học Quốc gia TP.HCM cũng như nhà trường đã cùng tri thức, tâm huyết đã truyền đạt cho chúng em trong suốt thời gian thực hiện đồ án môn học.

Đặc biệt hơn hết, nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Hồ Duy Tri – Giảng viên lý thuyết và thực hành môn Cơ sở dữ liệu phân tán đã trực tiếp giảng dạy, truyền đạt kiến thức, kinh nghiệm, hướng dẫn chúng em một cách cụ thể, tận tâm giúp chúng em hoàn thành tốt đồ án môn học của mình. Chúc thầy dồi dào sức khỏe, luôn luôn nhiệt huyết với nghề để có thể tiếp tục giảng dạy, dìu dắt cho thế hệ sinh viên tiếp theo.

Ngoài ra, chúng em cũng gửi lời cảm ơn tập thể lớp IS211.012.HTCL nói chung cũng như các bạn thành viên trong nhóm nói riêng đã cùng nhau học tập, cùng nhau nghiên cứu, học hỏi để thực hiện đồ án một cách tốt nhất.

Cuối cùng, nhóm chúng em đã hoàn thành tiểu luận “Hệ quản trị cơ sở dữ liệu Hadoop/HBase”. Mặc dù đã tìm hiểu, vận dụng tối đa những gì đã học nhưng vẫn khó tránh khỏi sai sót, vì vậy nhóm chúng em rất mong nhận được sự góp ý từ phía quý thầy để tiếp tục hoàn thiện dự án một cách tốt nhất. Qua đó, rút ra bài học và tích lũy thêm kinh nghiệm làm hành trang cho tương lai.

Chúng em xin hết và xin chân thành cảm ơn thầy và các bạn!

Trân trọng cảm ơn.

TP. Hồ Chí Minh, ngày 18 tháng 12 năm 2023

I. Giới thiệu

1. Giới thiệu về HBase

HBase (hay còn gọi là Hadoop Database) là một hệ thống lưu trữ dữ liệu phân tán, mở rộng và có khả năng chịu lỗi. Nó được thiết kế để xử lý lưu trữ lớn của dữ liệu có cấu trúc và được xây dựng trên nền tảng Apache Hadoop.

HBase là một cơ sở dữ liệu NoSQL, tức là nó không tuân theo mô hình quan hệ truyền thống của các hệ quản trị cơ sở dữ liệu như MySQL hay PostgreSQL.

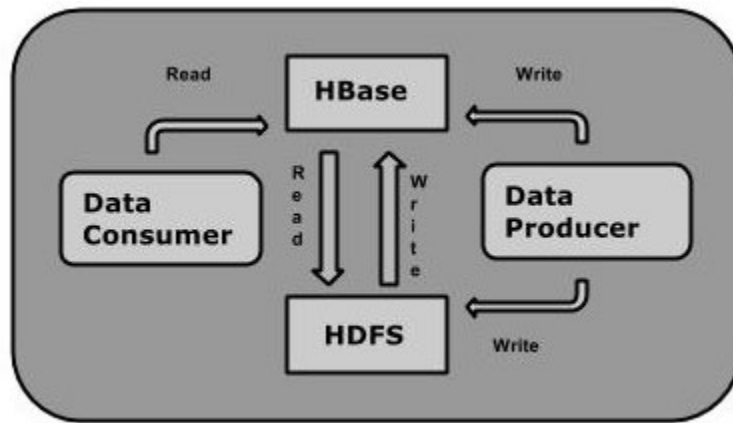
- Có nhiều cách phân loại các cơ sở dữ liệu NoSQL khác nhau, mỗi loại với các loại và loại con khác nhau, một số trong số đó có thể chồng chéo lên nhau. Một phân loại cơ bản dựa trên mô hình dữ liệu, với các ví dụ:

- **Column:** Accumulo, Cassandra, Druid, HBase, Vertica
- **Document:** Apache CouchDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, Lotus Notes, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB
- **Key-value:** Aerospike, CouchDB, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, Berkeley DB
- **Graph:** AllegroGraph, InfiniteGraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Stardog
- **Multi-model:** Alchemy Database, ArangoDB, CortexDB, FoundationDB, MarkLogic, OrientDB
- Tuy cùng mang những đặc điểm chung của NoSQL nhưng mỗi CSDL NoSQL cũng có những đặc điểm riêng, và vì thế thường được dùng cho những dự án khác nhau.

Hbase:

- o Sử dụng **Apache Hadoop Distributed File System (HDFS)** để lưu trữ dữ liệu. Dữ liệu được phân tán và sao chép trên các node trong mạng để đảm bảo tính sẵn sàng và độ tin cậy cao. HBase cũng có khả năng xử lý dữ liệu theo thời gian thực và hỗ trợ truy vấn dựa trên hàng ngàn hàng cột.
- o HBase sử dụng **kiến trúc cột (column-family-oriented)** và mô hình dữ liệu **key-value** với khả năng lưu trữ hàng tỷ hàng triệu hàng và hàng tỷ cột, lưu trữ dữ liệu dưới dạng các cặp key/value trong một mô hình cột. Trong mô hình này, tất cả các cột được nhóm lại với nhau thành các nhóm cột. HBase cung cấp mô hình dữ liệu linh hoạt và độ trễ thấp để truy cập vào một lượng nhỏ dữ liệu được lưu trong bộ dữ liệu lớn. HBase ở bên trên của Hadoop sẽ làm tăng thông lượng và hiệu suất của phân cụm được thiết lập. Đổi lại, nó giúp cho việc đọc và ghi nhanh hơn.

- o Là một phần của hệ sinh thái Hadoop cung cấp quyền truy cập đọc / ghi ngẫu nhiên theo thời gian thực vào dữ liệu trong Hệ thống tệp Hadoop.
- o Người ta có thể lưu trữ dữ liệu trong HDFS trực tiếp hoặc thông qua HBase. Người tiêu dùng dữ liệu đọc / truy cập dữ liệu trong HDFS một cách ngẫu nhiên bằng HBase. HBase nằm trên Hệ thống Tệp Hadoop và cung cấp quyền truy cập đọc và ghi.
- o Cung cấp các tính năng như dữ liệu được thêm, xóa và cập nhật theo hàng, khả năng sao chép và đồng bộ hóa dữ liệu tự động, hoạt động bất đồng bộ, và khả năng mở rộng linh hoạt. Nó cũng có giao diện dễ sử dụng và hỗ trợ truy vấn theo hàng, cột và phạm vi dữ liệu.



Hình 1. Sơ đồ cấp quyền của HBase trong Hệ thống tệp Hadoop.

2. Lịch sử về Hbase

HBase bắt nguồn từ dự án Google Bigtable, một hệ thống lưu trữ dữ liệu phân tán và có khả năng mở rộng, được mô tả trong một bài báo nổi tiếng của Google năm 2006.

Hadoop, dự án mã nguồn mở do Apache Software Foundation phát triển, đã cung cấp một nền tảng để xử lý và lưu trữ dữ liệu lớn. HBase xuất phát từ việc kết hợp cả hai ý tưởng này để tạo ra một cơ sở dữ liệu phân tán dựa trên Hadoop.

Và dự án HBase chính thức bắt đầu vào năm 2007 khi các nhà phát triển trong cộng đồng Apache Hadoop quyết định triển khai một phiên bản của Bigtable dựa trên Hadoop.

Ban đầu dự án được tạo ra để cung cấp một cơ sở dữ liệu có khả năng mở rộng cho Hadoop, cho phép lưu trữ và truy xuất dữ liệu lớn một cách hiệu quả.

Vì thế, HBase nhanh chóng thu hút sự chú ý và tham gia từ cộng đồng mã nguồn mở, và nó trở thành một dự án con của Apache Hadoop và các phiên bản mới liên tục được phát hành với cải tiến, sửa lỗi và tính năng mới.

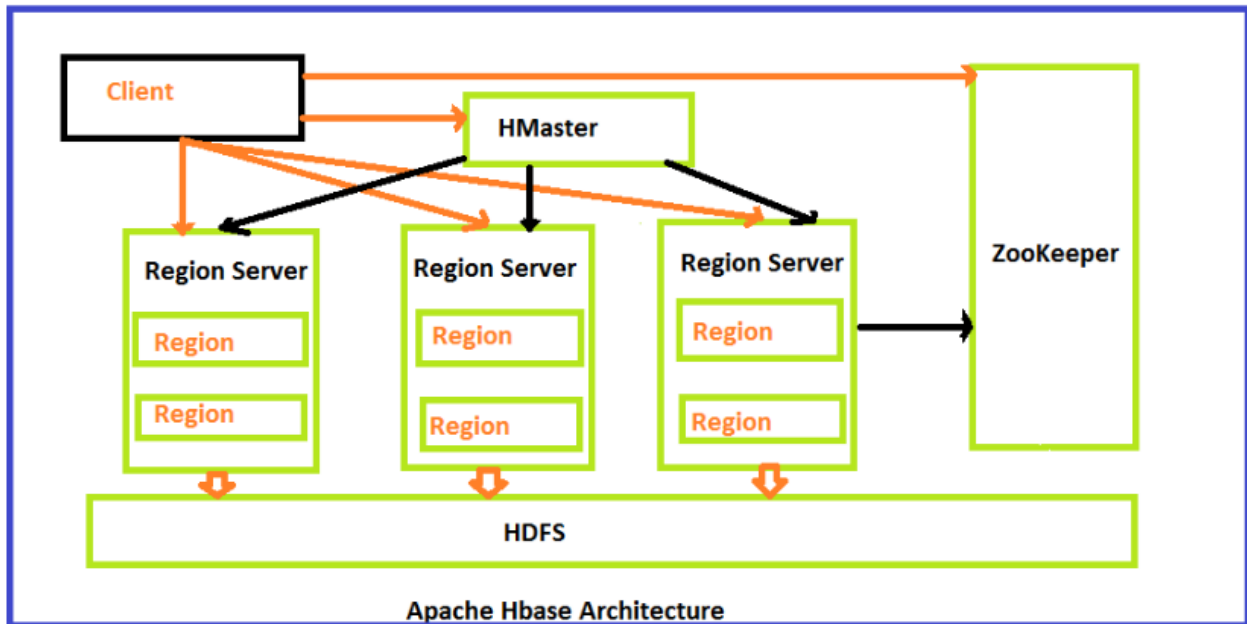
Cuối cùng, **HBase** đã trở thành một phần quan trọng của hệ sinh thái Apache Hadoop, cung cấp một giải pháp lưu trữ dữ liệu có khả năng mở rộng cho các ứng dụng lớn. Và đến hiện tại ngày càng nhiều tổ chức và doanh nghiệp lựa chọn HBase cho việc lưu trữ và xử lý dữ liệu lớn, đặc biệt là trong các lĩnh vực như phân tích dữ liệu, theo dõi thời gian thực, và các ứng dụng yêu cầu khả năng mở rộng cao.

3. Đặc điểm của HBase

- **Phân phối ngang (Horizontal Scalability):** HBase được thiết kế để mở rộng theo chiều ngang một cách dễ dàng, cho phép thêm máy chủ để tăng dung lượng lưu trữ và xử lý dữ liệu.
- **Cấu trúc dữ liệu linh hoạt:** HBase sử dụng mô hình cơ sở dữ liệu dạng cột (column-family-based), nơi dữ liệu được tổ chức thành các hàng và cột. Điều này tạo ra một không gian lưu trữ linh hoạt, với khả năng thêm và xóa cột mà không cần phải thay đổi cấu trúc cơ sở dữ liệu.
- **Hỗ trợ cho Dữ liệu Lớn:** HBase được tối ưu hóa để xử lý và lưu trữ lượng lớn dữ liệu. Điều này làm cho nó trở thành một lựa chọn phù hợp cho các ứng dụng yêu cầu khả năng mở rộng và xử lý dữ liệu lớn.
- **Tính Nhất Quán (Consistency):** HBase hỗ trợ tính nhất quán trong việc đọc và ghi dữ liệu. Điều này đặc biệt quan trọng trong các hệ thống đòi hỏi tính nhất quán như ngân hàng dữ liệu và hệ thống giao dịch.
- **Hỗ trợ cho Truy vấn Phân tán:** HBase hỗ trợ các truy vấn phân tán (distributed queries) thông qua sự tích hợp với Apache Hadoop. Điều này cho phép xử lý phức tạp trên dữ liệu phân tán.
- **Hỗ trợ cho Ghi đồng thời (Concurrent Writes):** HBase hỗ trợ ghi đồng thời từ nhiều nguồn, giúp đảm bảo tính nhất quán và khả năng mở rộng của hệ thống.
- **Hỗ trợ cho Tìm kiếm và Quét (Scan):** HBase cung cấp khả năng tìm kiếm và quét dữ liệu một cách hiệu quả, giúp người dùng truy xuất thông tin từ lượng lớn dữ liệu.

- **Tích hợp với Hadoop:** HBase tích hợp chặt chẽ với Hadoop, sử dụng Hadoop Distributed File System (HDFS) để lưu trữ dữ liệu và kết hợp với Hadoop MapReduce để xử lý dữ liệu.
- **Hệ sinh thái Mở rộng:** HBase là một phần của hệ sinh thái Apache Hadoop, có sẵn cho cộng đồng mã nguồn mở và hỗ trợ từ cộng đồng lớn.

4. Kiến trúc của HBase



Hình 4. Kiến trúc của HBase.

- **HBase Master Server:**
 - **Quản lý Metadata:** Thành phần quản lý hệ thống HMaster là thành phần trung tâm trong kiến trúc của HBase. HMaster giám sát tất cả các Region Server thuộc cluster. Tất cả các thay đổi liên quan đến metadata đều thực hiện thông qua HMaster. Những vai trò quan trọng nhất được thực hiện bởi HMaster: – Table : createTable, removeTable, enable, disable – ColumnFamily : add Column, modify Column – Region : move, assign.
- **Region Servers:**
 - **Chứa Dữ Liệu:** Quản lý và lưu trữ các region HRegionServer chịu trách nhiệm quản lý các region. Khi Region server nhận writes and read requests từ client, nó sẽ assign request cho region riêng biệt, nơi mà đang chứa column family. Tuy nhiên client có thể trực tiếp

liên lạc với Hregion servers, mà không cần sự cho phép của HMaster. Hregions: chứa 2 component chính là Memstore và Hfile. HRegions là thành phần kiến trúc cơ sở của Hbase cluster. Bao gồm 2 thành phần chính là Memstore và Hfile. Đây chính là nơi lưu trữ dữ liệu của các table.

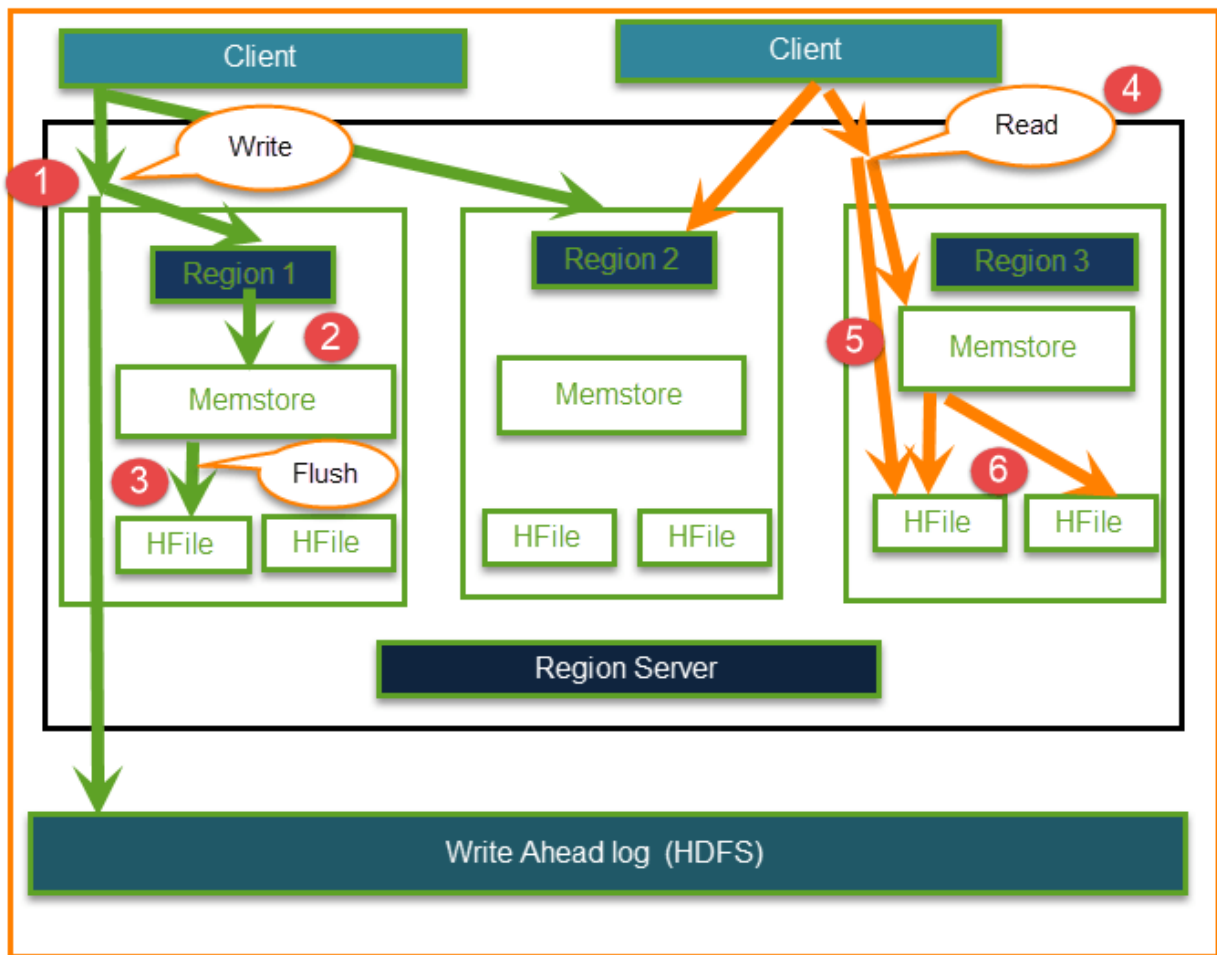
- **ZooKeeper:**
 - **Quản Lý Tình Trạng Hệ Thống:** lưu trữ các metadata, region info.
- **Hadoop Distributed File System (HDFS):**
 - **Lưu Trữ Dữ Liệu:** Dữ liệu của HBase được lưu trữ trong HDFS, một hệ thống tệp phân tán của dự án Apache Hadoop.
- **HBase Client:**
 - **Tương Tác với HBase:** HBase Client là thành phần chịu trách nhiệm cho việc tương tác với HBase, bao gồm việc thực hiện các truy vấn và ghi dữ liệu.

3 component quan trọng nhất là:

- **HBase Master Server**
- **Region Servers**
- **ZooKeeper**

5. Data flow trong HBase

Các thao tác Đọc và Ghi từ Client vào Hfile được hiển thị trong sơ đồ dưới:



Hình 5.1. Sơ đồ thao tác Đọc và Ghi từ Client vào Hfile.

- Write operations

- **BƯỚC 1:** Client muốn write data, tạo kết nối lần đầu tiên với Regions server và sau đó là regions.
- **BƯỚC 2:** Regions liên lạc với memstore, lưu lại liên kết với column family.
- **BƯỚC 3:** Dữ liệu trước tiên sẽ được lưu trữ tại Memstore. Tại đây dữ liệu sẽ được sắp xếp (sorted) trước khi chuyển tới HFile. Có 2 lý do chính cho việc sử dụng Memstore là :
 - + Hệ thống lưu trữ dữ liệu phân tán dựa trên row Key nên cần sắp xếp trước khi lưu trữ.

+ Tối ưu hóa luồng ghi dữ liệu khi sử dụng kiến trúc The Log-Structured Merge Tree

- Read operations

- **BƯỚC 4:** Client muốn đọc data từ Regions
- **BƯỚC 5:** Client có thể trực tiếp truy cập tới Mem store và yêu cầu dữ liệu.
- **BƯỚC 6:** Client get data từ HFile.

Memstore giữ các sửa đổi trong bộ nhớ cho cửa hàng. Hệ thống phân cấp các đối tượng trong Regions HBase được hiển thị từ trên xuống dưới trong hình dưới.

Table	HBase table present in the HBase cluster
Region	HRegions for the presented tables
Store	It stores per ColumnFamily for each region for the table
Memstore	<ul style="list-style-type: none"> • Memstore for each store for each region for the table • It sorts data before flushing into HFiles • Write and read performance will increase because of sorting
StoreFile	StoreFiles for each store for each region for the table
Block	Blocks present inside StoreFiles

Hình 5.2. Sơ đồ hệ thống phân cấp trong Regions HBase.

6. HBase so với RDBMS và NoSQL khác

	HBase	RDBMS	Hệ thống NoSQL khác
Mô hình dữ liệu	Mô hình dữ liệu cột (columnar), dữ liệu được tổ chức thành các cột và hàng	Mô hình dữ liệu quan hệ, dữ liệu được tổ chức thành các bảng có quan hệ với	Mô hình dữ liệu đa dạng như key-value, graph, document... Tổ

		nhau thông qua khóa chính và khóa ngoại	chức dữ liệu riêng biệt
Schema	Linh hoạt: Không yêu cầu định nghĩa cấu trúc dữ liệu trước khi lưu trữ	Cố định: Yêu cầu định nghĩa cấu trúc dữ liệu trước khi lưu trữ	Linh hoạt: Không yêu cầu định nghĩa cấu trúc dữ liệu trước khi lưu trữ
Tính chất	Tính mở rộng: khả năng mở rộng ngang, cho phép xử lý dữ liệu lớn. Có thể chia nhỏ dữ liệu và lưu trữ trên nhiều nút trong một cụm máy phân tán	Tính nhất quán và toàn vẹn: tuân thủ ràng buộc ACID	Tính nhất quán và toàn vẹn. Một số hệ thống NoSQL không tuân thủ ràng buộc ACID mà tập trung vào tính linh hoạt và hiệu suất
	Hiệu suất đọc/ghi: có khả năng xử lý lượng dữ liệu lớn	Đọc và ghi dữ liệu: bị giới hạn bởi dữ liệu lớn	

7. HBase Shell

HBase shell là một giao diện dòng lệnh được cung cấp bởi HBase để truy cập và thao tác với cơ sở dữ liệu HBase. Nó cho phép người dùng tương tác với HBase thông qua các lệnh dòng lệnh để thực hiện các thao tác như tạo bảng, thêm dữ liệu vào bảng, xóa dữ liệu khỏi bảng và truy vấn dữ liệu từ bảng.

Để sử dụng Hbase shell, ta sử dụng lệnh: ***hbase shell***

- Một số lệnh phổ biến trong Hbase Shell:

- create: tạo bảng mới trong HBase.
- put: thêm dữ liệu vào bảng HBase.
- get: lấy dữ liệu từ bảng HBase.
- scan: quét toàn bộ bảng HBase.
- delete: xóa dữ liệu từ bảng HBase.
- disable: tạm dừng một bảng HBase.
- enable: kích hoạt lại một bảng HBase.

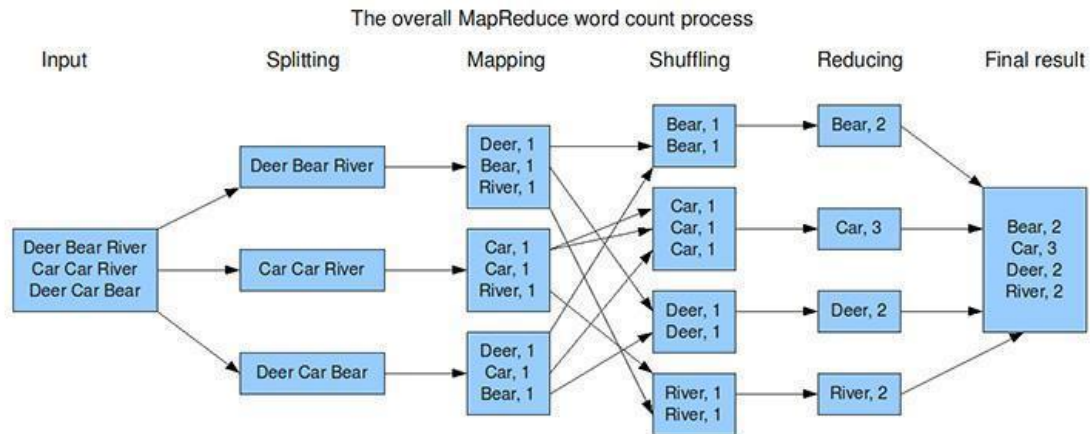
HBase shell là một công cụ mạnh mẽ để quản lý và tương tác với dữ liệu trong HBase. Cung cấp các chức năng cơ bản để tạo, đọc, ghi và xóa dữ liệu trong HBase từ giao diện dòng lệnh để sử dụng.

8. Tổng quan về Hadoop

Hadoop là một framework mã nguồn mở được thiết kế để xử lý và lưu trữ dữ liệu lớn (big data) trên một cụm máy tính phân tán. Nó cung cấp một mô hình phân tán để xử lý dữ liệu song song trên nhiều máy tính thông qua việc chia nhỏ dữ liệu và phân phối chúng trên các nút trong cụm.

- Kiến trúc của Hadoop: gồm 4 module khác nhau như sau

- **Hadoop Distributed File System (HDFS):** Đây là hệ thống tệp phân tán được thiết kế để lưu trữ và quản lý dữ liệu trên các nút trong cụm Hadoop. HDFS cho phép phân chia dữ liệu thành các khối nhỏ và nhân bản chúng trên nhiều máy tính để đảm bảo tính an toàn và sẵn sàng cao.
- **MapReduce:** Đây là mô hình lập trình và hệ thống tính toán phân tán trong Hadoop. MapReduce phân chia quá trình xử lý dữ liệu thành hai pha chính: pha Map và pha Reduce. Pha Map thực hiện xử lý song song trên các phần của dữ liệu đầu vào, sau đó pha Reduce tổng hợp và xử lý kết quả từ pha Map để cho ra kết quả cuối cùng.



Hình 8.1. Mô hình MapReduce.

- **YARN (Yet Another Resource Negotiator):** Đây là một bộ quản lý tài nguyên trong Hadoop, chịu trách nhiệm phân chia và quản lý thư viện tài nguyên máy tính (bộ nhớ, CPU, lưu trữ, băng thông mạng) trên cụm

Hadoop. YARN cho phép chạy nhiều ứng dụng khác nhau trên cùng một cụm Hadoop và quản lý việc phân phối tài nguyên cho từng ứng dụng.

- **Hadoop Common:** Đây là tập hợp các thành phần và thư viện chung cho toàn bộ hệ thống Hadoop. Nó bao gồm các công cụ hỗ trợ như giao diện dòng lệnh, thư viện xử lý tệp, giao thức mạng và các tiện ích khác.

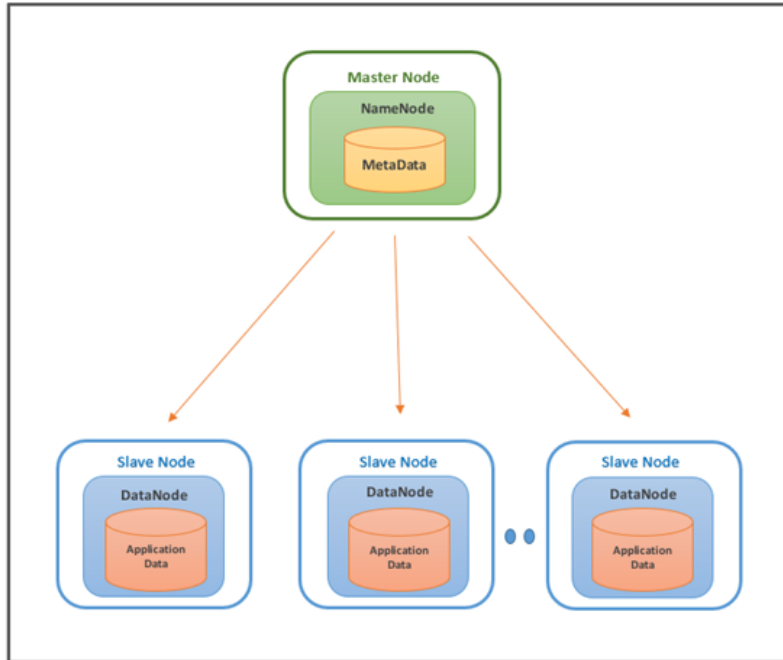
- Ưu điểm của Hadoop: cho phép người dùng nhanh chóng kiểm tra được tiến trình hoạt động của các phân tán; được sử dụng rộng rãi trong các ứng dụng xử lý dữ liệu lớn; có khả năng mở rộng tuyến tính, cho phép xử lý hiệu suất cao và độ tin cậy cao trên cụm máy tính phân tán.

9. HDFS là gì?

Hadoop Distributed File System (HDFS) là hệ thống lưu trữ dữ liệu được Hadoop sử dụng. Với việc ứng dụng kiến trúc NameNode và DataNode để triển khai hệ thống tệp phân tán. Nó đã cung cấp khả năng truy cập với hiệu suất cao đến các dữ liệu trên các cụm của Hadoop.

- HDFS sử dụng mô hình **master/slave** trong việc quản lý tệp và tài nguyên. Có hai thành phần chính trong HDFS:

- **NameNode:** Là thành phần master của HDFS, quản lý siêu dữ liệu (metadata) của hệ thống tệp. NameNode duy trì thông tin về cấu trúc thư mục, vị trí và kích thước của các khối dữ liệu, quản lý quyền truy cập và theo dõi trạng thái của các nút dữ liệu (DataNode) trong cụm.
- **DataNode:** Là thành phần slave trong HDFS, chịu trách nhiệm lưu trữ các khối dữ liệu thực tế trên đĩa cứng.



Hình 9.1. Kiến trúc của HDFS.

- HDFS cung cấp tính năng đọc và ghi dữ liệu với hiệu suất cao và khả năng mở rộng tuyến tính. Nó được sử dụng rộng rãi trong việc lưu trữ và xử lý dữ liệu lớn trong các ứng dụng Hadoop, cho phép xử lý dữ liệu trên nhiều máy tính song song một cách hiệu quả.

10. Cách thức hoạt động của HDFS

- Một tệp tin được di chuyển trên HDFS sẽ được chia nhỏ thành các phần riêng biệt. Các mảnh này sẽ được lưu trữ và phân tán trên các node khác nhau. Sau đó dữ liệu được ghi trên máy chủ, rồi sao chép và sử dụng lại nhiều lần.

- Một tệp tin có định dạng HDFS sẽ chia nhiều khối riêng biệt và những cụm này được lưu trong DataNodes. Các khối cũng được nhân rộng trên các nút cho phép xử lý và khắc phục được sự cố một cách nhanh chóng. NameNode định nghĩa ánh xạ, vị trí các khối đến DataNodes. Bên cạnh đó, nó cũng quản lý quyền truy cập vào các tệp cho phép đọc, ghi, xóa, sao chép và tạo mới dữ liệu.

- Các NameNode theo dõi và luôn biết được các trạng thái của DataNodes. Khi nhận thấy các chúng hoạt động không bình thường, nó sẽ chuyển nhiệm vụ của DataNodes này cho các node khác trong cùng một khối dữ liệu.

Hadoop Distributed File System (HDFS) được thiết kế với khả năng xử lý lỗi, dữ liệu lớn và tính sẵn sàng cao.

II. Hướng dẫn cài đặt

1. Cài đặt trên một máy

1.1. Chế độ Standalone mode

Bước 1. Tải java

```
sudo apt install openjdk-8-jdk -y
```

Bước 2. Kiểm tra cài đặt java và đường dẫn

```
java -version; javac -version
```

```
which javac
```

```
readlink -f /usr/bin/javac
```

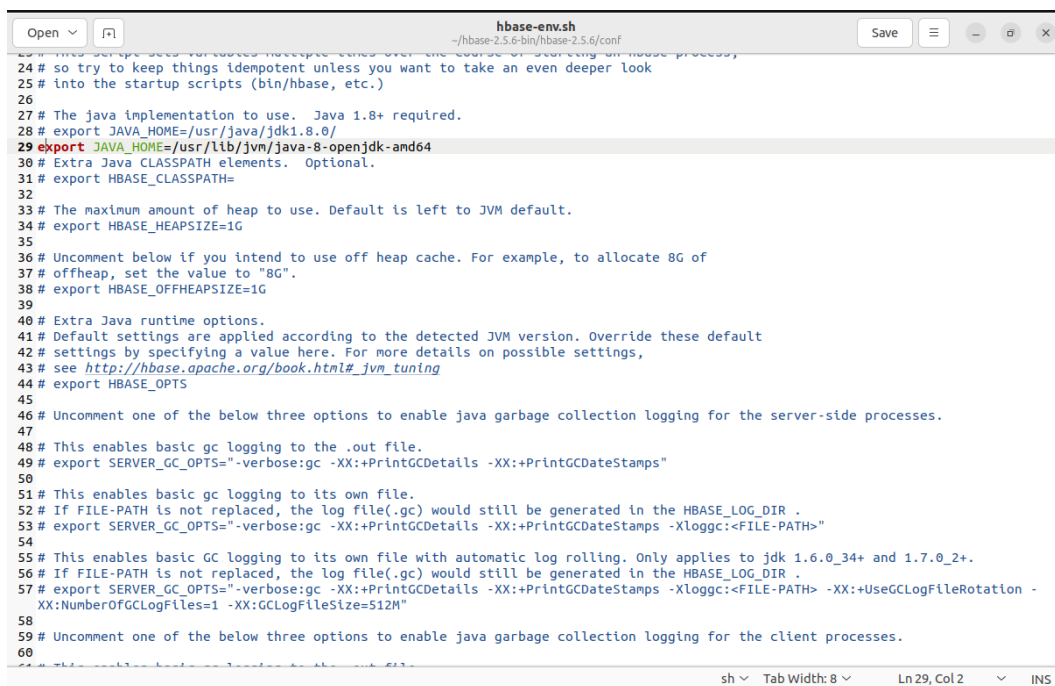
Bước 3. Tải hbase

<https://www.apache.org/dyn/closer.lua/hbase/>

Bước 4. Giải nén file hbase và đặt tại thư mục home

Bước 5. Vào conf mở hbase-env.sh

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```



```

24 # so try to keep things idempotent unless you want to take an even deeper look
25 # into the startup scripts (bin/hbase, etc.)
26
27 # The java implementation to use.  Java 1.8+ required.
28 # export JAVA_HOME=/usr/java/jdk1.8.0/
29 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
30 # Extra Java CLASSPATH elements.  Optional.
31 # export HBASE_CLASSPATH=
32
33 # The maximum amount of heap to use. Default is left to JVM default.
34 # export HBASE_HEAPSIZE=1G
35
36 # Uncomment below if you intend to use off heap cache. For example, to allocate 8G of
37 # offheap, set the value to "8G".
38 # export HBASE_OFFHEAPSIZE=1G
39
40 # Extra Java runtime options.
41 # Default settings are applied according to the detected JVM version. Override these default
42 # settings by specifying a value here. For more details on possible settings,
43 # see http://hbase.apache.org/book.html#_jvm_tuning
44 # export HBASE_OPTS
45
46 # Uncomment one of the below three options to enable java garbage collection logging for the server-side processes.
47
48 # This enables basic gc logging to the .out file.
49 # export SERVER_GC_OPTS="-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps"
50
51 # This enables basic gc logging to its own file.
52 # If FILE-PATH is not replaced, the log file(.gc) would still be generated in the HBASE_LOG_DIR .
53 # export SERVER_GC_OPTS="-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:<FILE-PATH>"
54
55 # This enables basic GC logging to its own file with automatic log rolling. Only applies to jdk 1.6.0_34+ and 1.7.0_2+.
56 # If FILE-PATH is not replaced, the log file(.gc) would still be generated in the HBASE_LOG_DIR .
57 # export SERVER_GC_OPTS="-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:<FILE-PATH> -XX:+UseGCLogFileRotation -
  XX:NumberOfGCLogFiles=1 -XX:GCLogFileSize=512M"
58
59 # Uncomment one of the below three options to enable java garbage collection logging for the client processes.
60

```

Hình 1.1. Cấu hình file hbase-env.sh

Bước 6. Khởi động hbase

```
cd -
```

```
cd -/hbase-2.5.6-bin/hbase-2.5.6/bin
```

```
./start-hbase.sh
```

```
kiem tra jps
```

Bước 7. Mở hbase shell

1.2. Chế độ Pseudo-Distributed mode

Phần I. Cài đặt Hadoop

Bước 1. Cài đặt java

```
sudo apt install openjdk-8-jdk -y
```

Bước 2. Kiểm tra cài đặt java và đường dẫn

```
java -version; javac -version
```

```
which javac
```

```
readlink -f /usr/bin/javac
```

Bước 3. Tải Hadoop

```
https://hadoop.apache.org/releases.html
```

Bước 4. Giải nén file đặt tại thư mục home

Bước 5. Cài đặt openSSH

```
sudo apt install openssh-server openssh-client -y
```

Bước 6. Tạo và cài đặt SSH Certificates

```
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
```

```
-- lưu lại thông tin
```

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
-- cấp quyền cho user
```

```
chmod 0600 ~/.ssh/authorized_keys
```


-- kiểm tra ssh

ssh localhost

Bước 7. Cấu hình lại các file Files trong hadoop-3.3.6/etc/Hadoop

7.1. cấu hình: bashrc

gedit ~/.bashrc

---copy vào cuối file để setup các đường dẫn env

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

```
export HADOOP_HOME=/home/huy/hadoop-3.3.6
```

```
export HADOOP_INSTALL=$HADOOP_HOME
```

```
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

```
export HADOOP_COMMON_HOME=$HADOOP_HOME
```

```
export HADOOP_HDFS_HOME=$HADOOP_HOME
```

```
export YARN_HOME=$HADOOP_HOME
```

```
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
```

```
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

```
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

```

94
95 # Add an "alert" alias for long running commands. Use like so:
96 #   sleep 10; alert
97 alias alert='notify-send --urgency=low -i "${?} = 0" && echo terminal || echo error)' "${history|tail -n1|sed -e '\s/'
98   ^\s*[0-9]+\s*//;\s/[;&]\s*alert$/'\''}'
99
100 # Alias definitions.
101 # You may want to put all your additions into a separate file like
102 # ~/.bash_aliases, instead of adding them here directly.
103 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
104
105 if [ -f ~/.bash_aliases ]; then
106     . ~/.bash_aliases
107 fi
108
109 # enable programmable completion features (you don't need to enable
110 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
111 # sources /etc/bash.bashrc).
112 if ! shopt -oq posix; then
113     if [ -f /usr/share/bash-completion/bash_completion ]; then
114         . /usr/share/bash-completion/bash_completion
115     elif [ -f /etc/bash_completion ]; then
116         . /etc/bash_completion
117     fi
118 fi
119
120 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
121 export HADOOP_HOME=/home/huy/hadoop-3.3.6
122 export HADOOP_INSTALL=$HADOOP_HOME
123 export HADOOP_MAPRED_HOME=$HADOOP_HOME
124 export HADOOP_COMMON_HOME=$HADOOP_HOME
125 export HADOOP_HDFS_HOME=$HADOOP_HOME
126 export YARN_HOME=$HADOOP_HOME
127 export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
128 export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
129 export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

```

Hình 1.2 Cấu hình file bashrc

7.2 Cấu hình file hadoop-env.sh

--thêm địa chỉ JAVA_HOME cho hadoop

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

```

16 # Limitations under the license.
17
18 # Set Hadoop-specific environment variables here.
19
20 ##
21 ## THIS FILE ACTS AS THE MASTER FILE FOR ALL HADOOP PROJECTS.
22 ## SETTINGS HERE WILL BE READ BY ALL HADOOP COMMANDS. THEREFORE,
23 ## ONE CAN USE THIS FILE TO SET YARN, HDFS, AND MAPREDUCE
24 ## CONFIGURATION OPTIONS INSTEAD OF xxx-env.sh.
25 ##
26 ## Precedence rules:
27 ## {yarn-env.sh|hdfs-env.sh} > hadoop-env.sh > hard-coded defaults
28 ##
29 ## {YARN_xyz|HDFS_xyz} > HADOOP_xyz > hard-coded defaults
30 ##
31 ##
32
33 # Many of the options here are built from the perspective that users
34 # may want to provide OVERRIDING values on the command line.
35 # For example:
36 #
37 # JAVA_HOME=/usr/java/testing hdfs dfs -ls
38 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
39 #
40 # Therefore, the vast majority (BUT NOT ALL!) of these defaults
41 # are configured for substitution and not append. If append
42 # is preferable, modify this file accordingly.
43
44 ###
45 # Generic settings for HADOOP
46 ###
47
48 # Technically, the only required environment variable is JAVA_HOME.
49 # All others are optional. However, the defaults are probably not
50 # preferred. Many sites configure these options outside of Hadoop,
51 # such as in /etc/profile.d
52
53 # The java implementation to use. By default, this environment
54 # variable is REQUIRED on ALL platforms except OS X!

```

Hình 1.3 Cấu hình file *hadoop-env.sh*

7.3 Cấu hình file *core-site.xml*

```
<property>
<name>hadoop.tmp.dir</name>
<value>/home/huy/hadoop-3.3.6/tmp</value>
</property>
<property>
<name>fs.default.name</name >
<value>hdfs://localhost:9000</value>
</property>
```



Hình 1.4 Cấu hình file *core-site.xml*

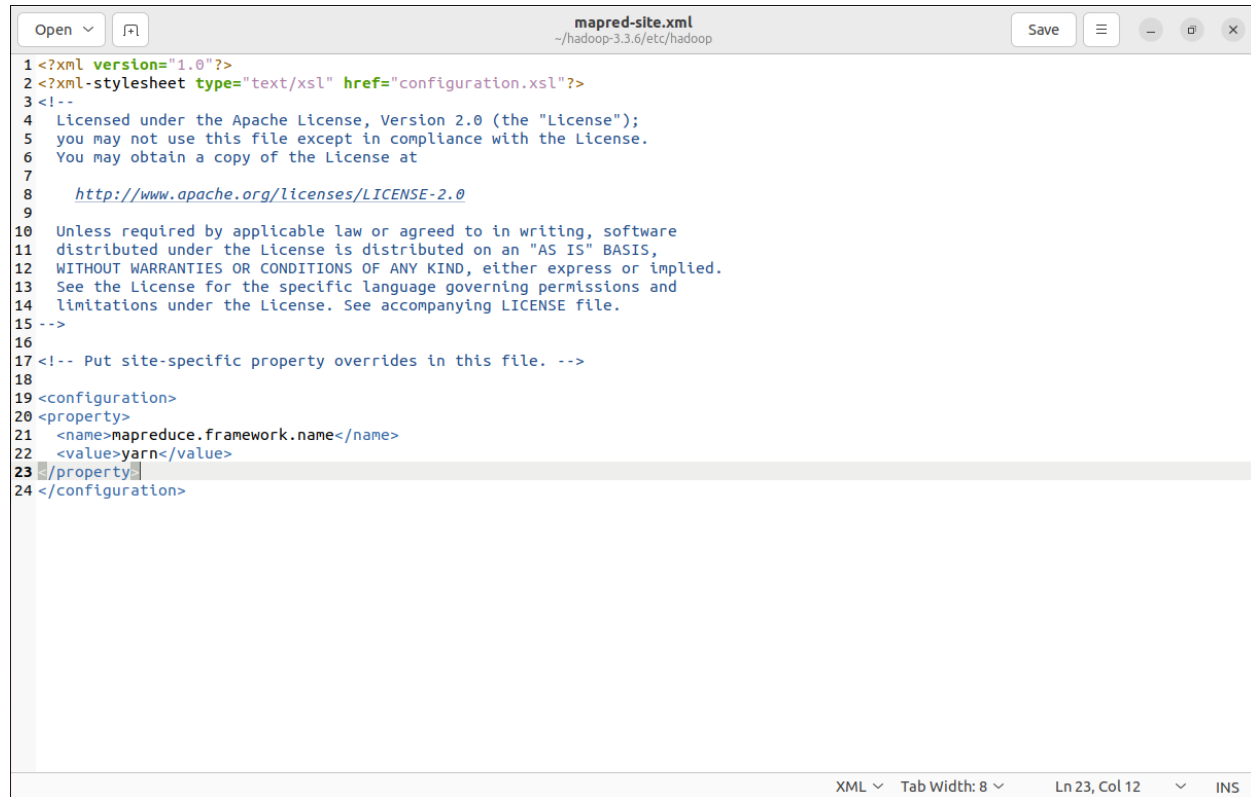
7.4 Cấu hình file *mapred-site.xml*

```
<property>
```

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>



Hình 1.5 Cấu hình file mapred-site.xml

7.5 Cấu hình file hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>/home/huy/hadoop-3.3.6/data/namenode</value>

</property>

```
<property>
<name>dfs.datanode.data.dir</name>
<value>/home/huy/hadoop-3.3.6/data/datanode</value>
</property>
```



Hình 1.6 Cấu hình file hdfs-site.xml

7.6 Cấu hình file yarn-site.xml

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

```

</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>127.0.0.1</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HA
DOOP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOM
E,HADOOP_MAPRED_HOME</value>
</property>

```



Hình 1.7 Cấu hình file yarn-site.xml

Bước 8. Format file hệ thống Hadoop mới

hdfs namenode -format

Bước 9. Chạy namenode, datanode, secondary namenode

cd -/hadoop-3.3.6/sbin

start-dfs.sh

Bước 10. chạy yarn: resourcemanager, nodemanagers

start-yarn.sh

Bước 11. Kiểm tra liên kết

Jps

Phần II. Cài đặt Hbase

Bước 1. Tải hbase

<https://www.apache.org/dyn/closer.lua/hbase/>

Bước 2. Giải nén file hbase và đặt tại thư mục home

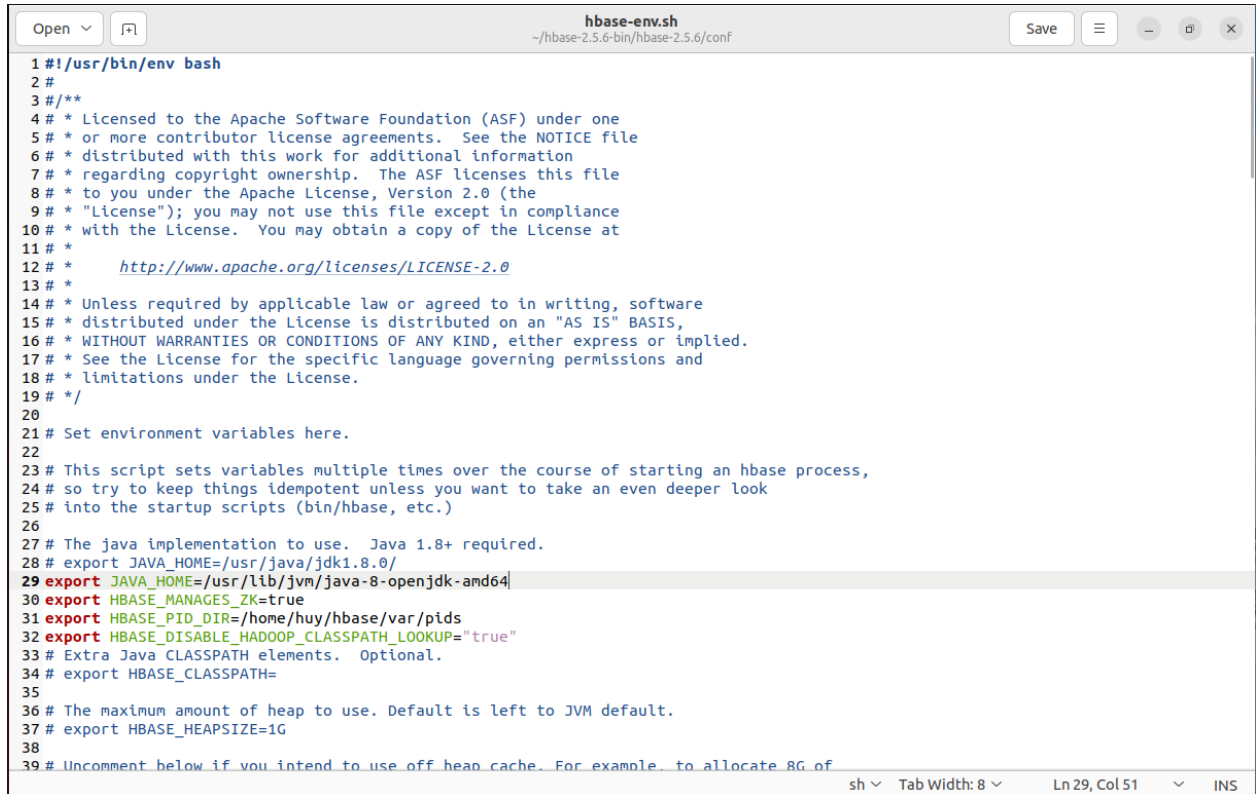
Bước 3. Vào conf mở hbase-env.sh

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export HBASE_MANAGES_ZK=true

export HBASE_PID_DIR=/home/huy/hbase/var/pids

export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"



```

1 #!/usr/bin/env bash
2 #
3 /**
4  * Licensed to the Apache Software Foundation (ASF) under one
5  * or more contributor license agreements. See the NOTICE file
6  * distributed with this work for additional information
7  * regarding copyright ownership. The ASF licenses this file
8  * to you under the Apache License, Version 2.0 (the
9  * "License"); you may not use this file except in compliance
10 * with the License. You may obtain a copy of the License at
11 *
12 * http://www.apache.org/licenses/LICENSE-2.0
13 *
14 * Unless required by applicable law or agreed to in writing, software
15 * distributed under the License is distributed on an "AS IS" BASIS,
16 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
17 * See the License for the specific language governing permissions and
18 * limitations under the License.
19 */
20
21 # Set environment variables here.
22
23 # This script sets variables multiple times over the course of starting an hbase process,
24 # so try to keep things idempotent unless you want to take an even deeper look
25 # into the startup scripts (bin/hbase, etc.)
26
27 # The java implementation to use. Java 1.8+ required.
28 # export JAVA_HOME=/usr/java/jdk1.8.0/
29 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
30 export HBASE_MANAGES_ZK=true
31 export HBASE_PID_DIR=/home/huy/hbase/var/pids
32 export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"
33 # Extra Java CLASSPATH elements. Optional.
34 # export HBASE_CLASSPATH=
35
36 # The maximum amount of heap to use. Default is left to JVM default.
37 # export HBASE_HEAPSIZE=1G
38
39 # Uncomment below if you intend to use off heap cache. For example, to allocate 8G of

```

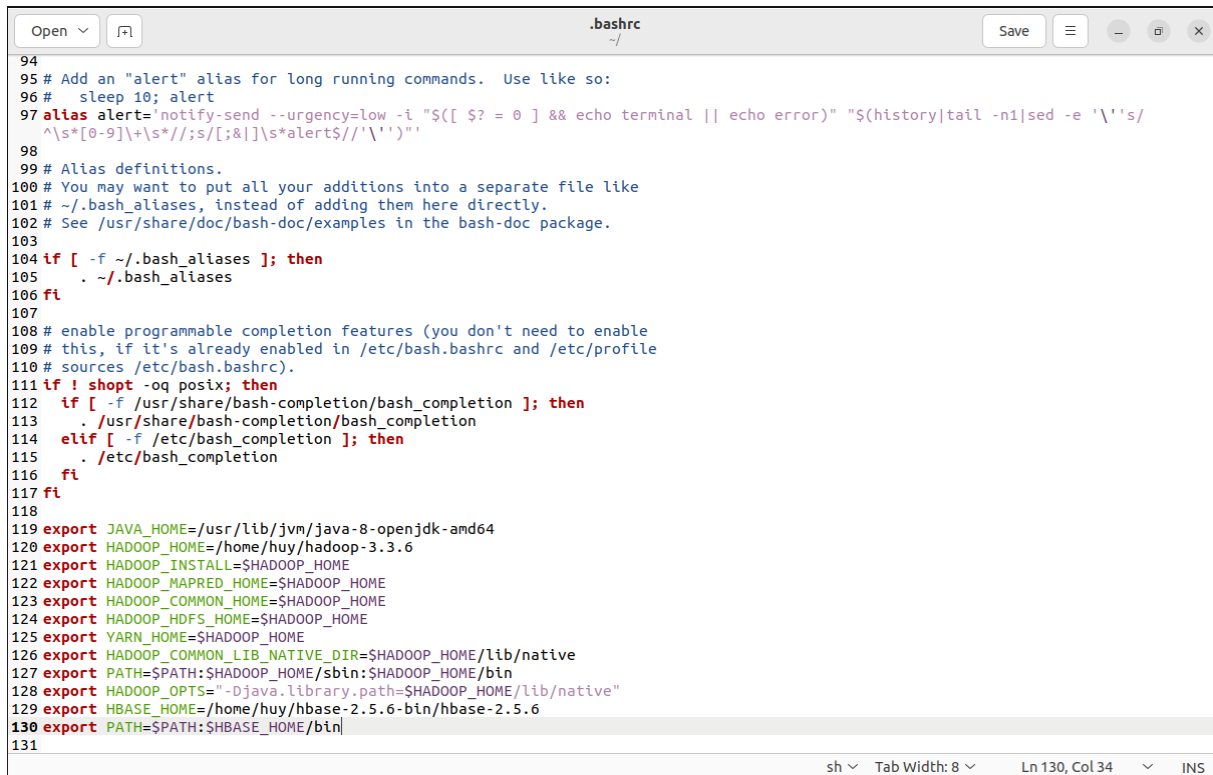
Hình 1.8 Cấu hình file hbase-env.sh

Bước 4. Mo bashrc va them duong dan HBASE_HOME

gedit ~/.bashrc

export HBASE_HOME=/home/huy/hbase-2.5.6-bin/hbase-2.5.6

export PATH=\$PATH:\$HBASE_HOME/bin



```

94
95 # Add an "alert" alias for long running commands. Use like so:
96 # sleep 10; alert
97 alias alert='notify-send --urgency=low -i "${[ $? = 0 ]} && echo terminal || echo error" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/;;s/;&[]\s*alert$/'\`)'"'
98
99 # Alias definitions.
100 # You may want to put all your additions into a separate file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc package.
103
104 if [ -f ~/.bash_aliases ]; then
105     . ~/.bash_aliases
106 fi
107
108 # enable programmable completion features (you don't need to enable
109 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
110 # sources /etc/bash.bashrc).
111 if ! shopt -oq posix; then
112     if [ -f /usr/share/bash-completion/bash_completion ]; then
113         . /usr/share/bash-completion/bash_completion
114     elif [ -f /etc/bash_completion ]; then
115         . /etc/bash_completion
116     fi
117 fi
118
119 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
120 export HADOOP_HOME=/home/huy/hadoop-3.3.6
121 export HADOOP_INSTALL=$HADOOP_HOME
122 export HADOOP_MAPRED_HOME=$HADOOP_HOME
123 export HADOOP_COMMON_HOME=$HADOOP_HOME
124 export HADOOP_HDFS_HOME=$HADOOP_HOME
125 export YARN_HOME=$HADOOP_HOME
126 export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
127 export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
128 export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
129 export HBASE_HOME=/home/huy/hbase-2.5.6-bin/hbase-2.5.6
130 export PATH=$PATH:$HBASE_HOME/bin
131

```

Hình 1.9 Cấu hình file bashrc

Bước 5. Mở hbase-site.xml và thêm vào properties

```

<property>
<name>hbase.rootdir</name>
<value>file:///home/huy/hbase</value>
</property>
<property>
<name>hbase.zookeeper.property.dataDir</name>
<value>/home/huy/zookeeper</value>
</property>

```

```

22 <configuration>
23 <!--
24 The following properties are set for running HBase as a single process on a
25 developer workstation. With this configuration, HBase is running in
26 "stand-alone" mode and without a distributed file system. In this mode, and
27 without further configuration, HBase and ZooKeeper data are stored on the
28 local filesystem, in a path under the value configured for 'hbase.tmp.dir'.
29 This value is overridden from its default value of '/tmp' because many
30 systems clean '/tmp' on a regular basis. Instead, it points to a path within
31 this HBase installation directory.
32
33 Running against the 'LocalFileSystem', as opposed to a distributed
34 filesystem, runs the risk of data integrity issues and data loss. Normally
35 HBase will refuse to run in such an environment. Setting
36 'hbase.unsafe.stream.capability.enforce' to 'false' overrides this behavior,
37 permitting operation. This configuration is for the developer workstation
38 only and __should not be used in production!__
39
40 See also https://hbase.apache.org/book.html#standalone\_dist
41 -->
42 <property>
43 <name>hbase.rootdir</name>
44 <value>file:///home/huy/hbase</value>
45 </property>
46
47 <property>
48 <name>hbase.zookeeper.property.dataDir</name>
49 <value>/home/huy/zookeeper</value>
50 </property>
51 <property>
52 <name>hbase.cluster.distributed</name>
53 <value>false</value>
54 </property>
55 <property>
56 <name>hbase.tmp.dir</name>
57 <value>./tmp</value>
58 </property>
59 <property>
60 <name>hbase.unsafe.stream.capability.enforce</name>

```

Hình 1.10 Cấu hình file hbase-site.xml

Bước 6. Tạo 2 folder hbase và zookeeper trong thư mục home và cấp quyền

```
sudo chmod 777 zookeeper/
```

```
sudo chmod 777 hbase
```

Bước 7. Khởi động HBase

```
cd -
```

```
cd -/hbase-2.5.6-bin/hbase-2.5.6/bin
```

```
./start-hbase.sh
```

Kiểm tra jps

Bước 8. Mở hbase shell

2. Chế độ Fully Distributed mode

2.1. Cài đặt trên 1 cụm máy

Trước tiên ta phải thiết lập multi Node Cluster trong Hadoop:

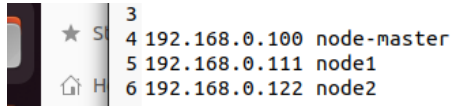
Ta có 3 máy có ip lần lượt là :

node-master :192.168.0.100

node1: 192.168.0.111

node2: 192.168.0.122

-Bước 1: Ta phải thêm ip của master node và datanode vào file /etc/hosts:



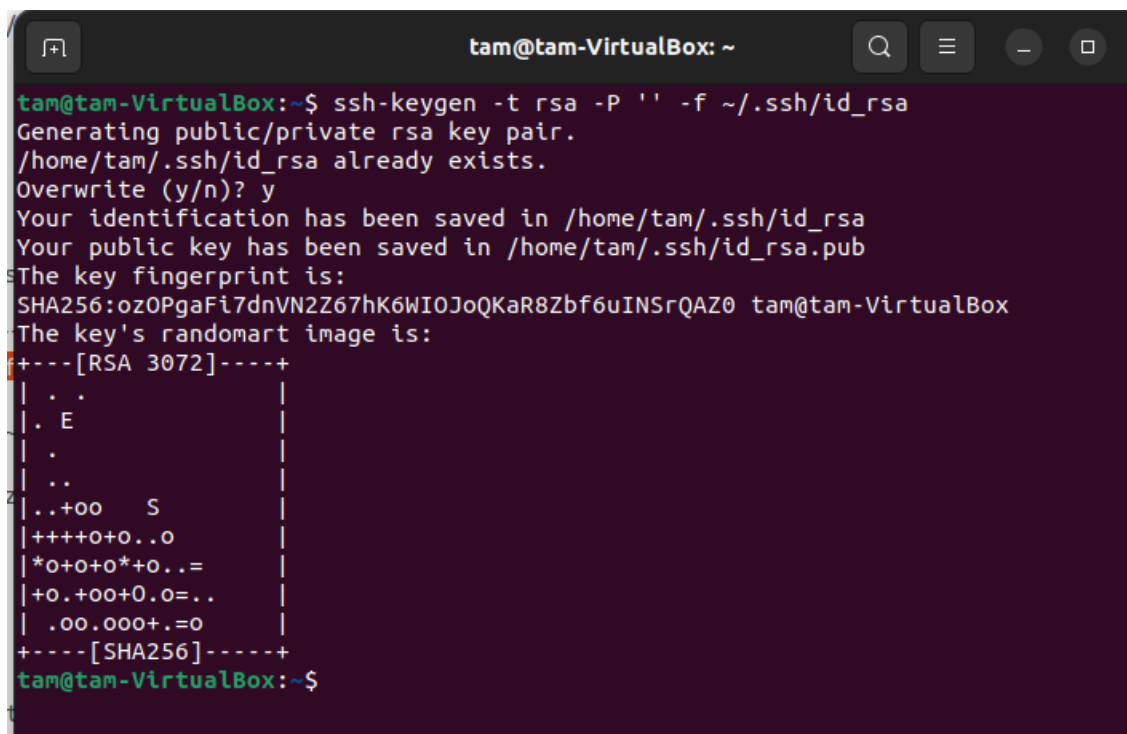
```

3
4 192.168.0.100 node-master
5 192.168.0.111 node1
6 192.168.0.122 node2

```

Hình 2.1: thêm ip vào file /etc/host

-Bước 2: Tiếp theo ta tạo SSH key trên node master và gửi cho các node1,node2 :



```

tam@tam-VirtualBox: ~
tam@tam-VirtualBox:~$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
/home/tam/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Your identification has been saved in /home/tam/.ssh/id_rsa
Your public key has been saved in /home/tam/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:oz0PgaFi7dnVN2Z67hK6WIOJoQKaR8Zbf6uINSrQAZ0 tam@tam-VirtualBox
The key's randomart image is:
+---[RSA 3072]-----+
|  .  |
|. E  |
|  .  |
|..   |
|..+oo S|
|+++o+o..o|
|*o+o+o*+o..=|
|+o.+oo+o.o=..|
| .oo.ooo+. =o|
+---[SHA256]-----+
tam@tam-VirtualBox:~$

```

Hình 2.2: Tạo SSH key

Bước 3: Sao chép SSH key vừa tạo để làm authorized key của Master Node và sao chép key ssh master node sang authorized key của slave node

```

..
...oo  S
+++++o+o..o
*o+o+o*+o..=
+o..+oo+o..o=..
..oo..ooo+..=o
+----[SHA256]-----+
tam@tam-VirtualBox:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
tam@tam-VirtualBox:~$ chmod 0600 ~/.ssh/authorized_keys
tam@tam-VirtualBox:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub tam@node1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/tam/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'tam@node1'"
and check to make sure that only the key(s) you wanted were added.

tam@tam-VirtualBox:~$

Now try logging into the machine, with: "ssh 'tam@node1'"
and check to make sure that only the key(s) you wanted were added.

tam@tam-VirtualBox:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub tam@node2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/tam/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

Number of key(s) added: 1

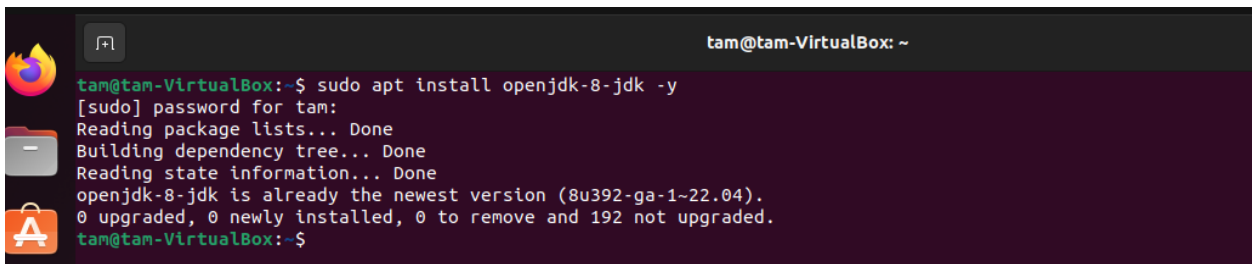
Now try logging into the machine, with: "ssh 'tam@node2'"
and check to make sure that only the key(s) you wanted were added.

tam@tam-VirtualBox:~$

```

Hình 2.3: Copy key sang cho các datanode

Bước 4 : cài Java trên tất cả các node bằng lệnh : `sudo apt install openjdk-8-jdk-y`



```

tam@tam-VirtualBox:~$ sudo apt install openjdk-8-jdk -y
[sudo] password for tam:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openjdk-8-jdk is already the newest version (8u392-ga-1~22.04).
0 upgraded, 0 newly installed, 0 to remove and 192 not upgraded.
tam@tam-VirtualBox:~$

```

Hình 2.4: Tải JDK

Bước 5: Cài đặt Hadoop trên tất cả các node (nếu chưa cài đặt) ở báo cáo này xài Hadoop-3.3.6.

Bước 6: Thêm Hadoop path và Java path vào bash file trên tất cả các node :
 -Mở file `.bashrc` để thêm các đường dẫn Hadoop và Java. Thêm các nội dung sau:

```

115 . /etc/bash_completion
116 fi
117 fi
118
119
120 export HADOOP_HOME=/home/tam/hadoop
121 export PATH=${PATH}:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin
122
123

```

Hình 2.5: Cấu hình trong `/.bashrc`

-Bước 7: Export thêm Java path trong file:
`$HADOOP_HOME/etc/hadoop/hadoop_env.sh`

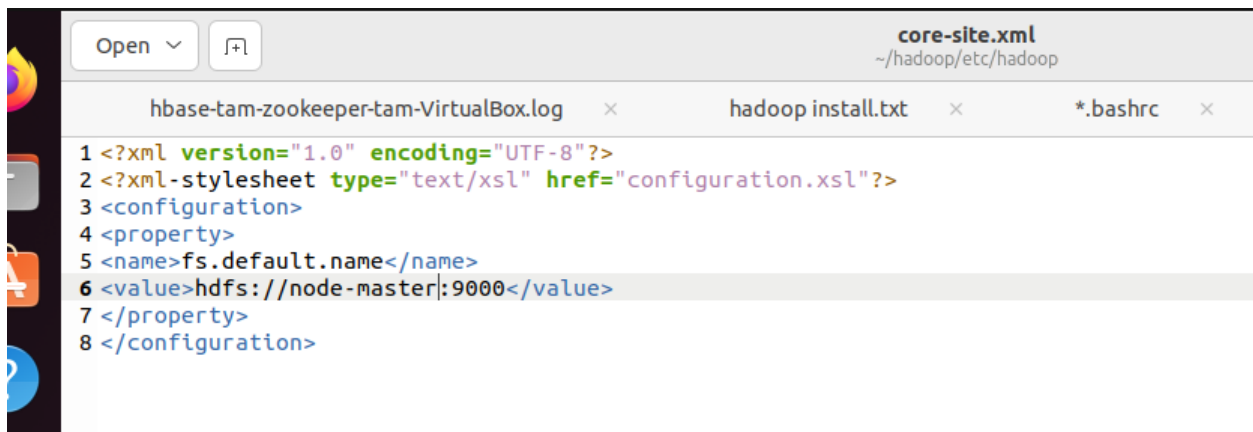
```

13 # distributed under the License is distributed on an "AS IS" BASIS,
14 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 # See the License for the specific language governing permissions and
16 # limitations under the License.
17
18 # Set Hadoop-specific environment variables here.
19 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
20 ##
21 ## THIS FILE ACTS AS THE MASTER FILE FOR ALL HADOOP PROJECTS.
22 ## SETTINGS HERE WILL BE READ BY ALL HADOOP COMMANDS. THEREFORE,
23 ## ONE CAN USE THIS FILE TO SET YARN, HDFS, AND MAPREDUCE

```

Hình 2.6: Cấu hình trong file `Hadoop-env.sh`

Bước 8 : thiết đặt file `core-site.xml` tại
`"~/hadoop/etc/hadoop/core-site.xml"`:



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3 <configuration>
4 <property>
5 <name>fs.default.name</name>
6 <value>hdfs://node-master:9000</value>
7 </property>
8 </configuration>

```

Hình 2.7: Cấu hình file `core-site.xml`

Bước 9: Đặt đường dẫn cho HDFS:

```

14 limitations under the License. See accompanying LICENSE file.
15 -->
16
17 <!-- Put site-specific property overrides in this file. -->
18
19 <configuration>
20 <property>
21 <name>dfs.namenode.name.dir</name>
22 <value>/home/tam/data/nameNode</value>
23 </property><property>
24 <name>dfs.datanode.data.dir</name>
25 <value>/home/tam/data/dataNode</value>
26 </property><property>
27 <name>dfs.replication</name>
28 <value>3</value>
29 </property>
30 </configuration>

```

Hình 2.7: Đặt đường dẫn cho HDFS

-Bước 10 : chỉnh sửa tệp mapred-site.xml, đặt YARN làm khung mặc định cho các hoạt động MapReduce:

```

18
19 <configuration>
20 <property>
21 <name>mapreduce.framework.name</name>
22 <value>yarn</value>
23 </property>
24 <property>
25 <name>yarn.app.mapreduce.am.env</name>
26 <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
27 </property>
28 <property>
29 <name>mapreduce.map.env</name>
30 <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
31 </property>
32 <property>
33 <name>mapreduce.reduce.env</name>
34 <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
35 </property>
36 <property>
37 <name>yarn.app.mapreduce.am.resource.mb</name>
38 <value>512</value>
39 </property><property>
40 <name>mapreduce.map.memory.mb</name>
41 <value>256</value>
42 </property><property>
43 <name>mapreduce.reduce.memory.mb</name>

```

Hình 2.8: Chỉnh sửa mared-site.xml

Bước 11 :Chỉnh sửa yarn-site.xml :

```

12  see the License for the specific language governing permissions and
13  limitations under the License. See accompanying LICENSE file.
14 -->
15 <configuration>
16 <property>
17 <name>yarn.acl.enable</name>
18 <value>0</value>
19 </property><property>
20 <name>yarn.resourcemanager.hostname</name>
21 <value>192.168.0.100</value>
22 </property><property>
23 <name>yarn.nodemanager.aux-services</name>
24 <value>mapreduce_shuffle</value>
25 </property>
26 <property>
27 <name>yarn.nodemanager.resource.memory-mb</name>
28 <value>1536</value>
29 </property><property>
30 <name>yarn.scheduler.maximum-allocation-mb</name>
31 <value>1536</value>
32 </property><property>
33 <name>yarn.scheduler.minimum-allocation-mb</name>
34 <value>128</value>
35 </property><property>
36 <name>yarn.nodemanager.vmem-check-enabled</name>
37 <value>false</value>
38 </property>
39 </configuration>

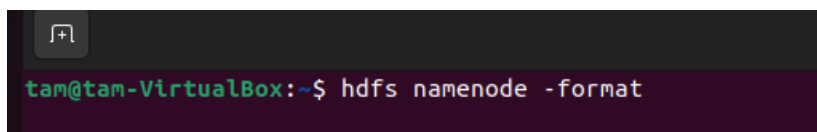
```

Bracket match found on line: 1

Hình 2.9: Chỉnh sửa yarn-site.xml

- bước 12 :Ta chỉnh sửa “~/hadoop/etc/hadoop/worker” để bao gồm cả hai nút: ta thêm node1,node2
- Bước 13 :Ta thực hiện nhân bản cấu hình namenode này cho các node còn lại.
- Bước 14: Cuối cùng trong cài đặt ta phải thực hiện format lại HDFS trước khi sử dụng , ta sử dụng lệnh:

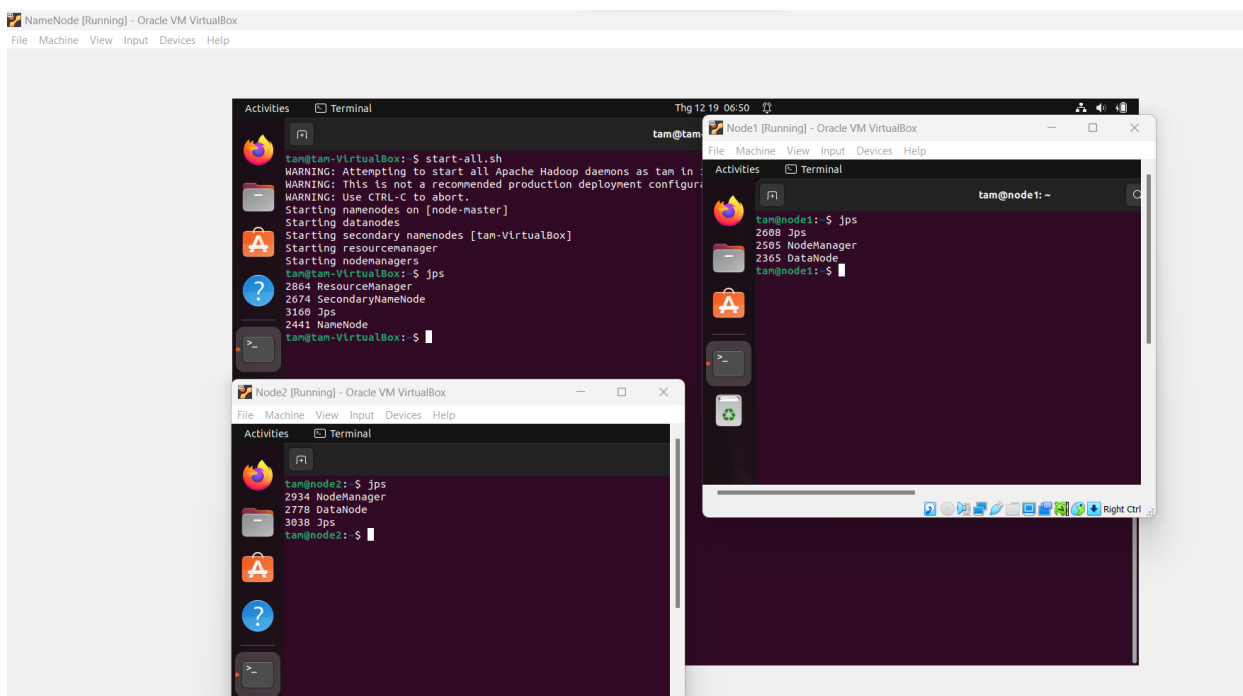
Hdfs namenode -format



Hình 2.10: Sử dụng lệnh để format HDFS

Bước 14 : Ta thử chạy kiểm tra thử HDFS và YARN đã hoạt động bình thường chưa bằng các lệnh : start-dfs.sh , start-yarn.sh. Sau khi chạy hoàn tất ta có cấu hình 3 máy gồm 1 node-master và 2 namenode như sau :

Vậy là đã cài thành công Hadoop ở chế độ phân tán trên nhiều nút .



Hình 2.11: Cài đặt thành công Hadoop cluster

Tiếp theo ta thực hiện cài Hmaster trên chung với namnode và 2 regionserver trên 2 datanode.

III. Tài liệu tham khảo

- [1]. HBase – Tổng quan (isolution.pro)
- [2]. Tìm hiểu hbase là gì và ứng dụng trong lưu trữ dữ liệu (memart.vn)
- [3]. Giới thiệu về HBase | HBase | Laptrinh.vn
- [4]. Apache HBase™ Reference Guide (2023)
- [5]. HBase Architecture: Use Cases, Components & Data Model (guru99.com)
- [6]. Trịnh Duy Thanh (2022) – HDFS là gì? Tính năng và lợi ích của HDFS
- [7]. Cài đặt và cấu hình Hadoop Cluster (123host.vn)