ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN CS232.M21: TÍNH TOÁN ĐA PHƯƠNG TIỆN

CHỦ ĐỀ: XỬ LÝ DỮ LIỆU LỚN BẰNG HADOOP

Giảng viên hướng dẫn: ThS. Đỗ Văn Tiến

Sinh viên thực hiện:

Họ và tên	**MSSV**
1. Trương Thế Tấn	19522180
2. Nguyễn Văn Thành	19522243
3. Lương Tường Quy	19522108

TP. Hồ Chí Minh, tháng 7, năm 2022

Mục lục

I. Giới thiệu về Dữ liệu lớn

1. Lịch sử của Dữ liệu lớn

Nguồn gốc của tập dữ liệu lớn bắt nguồn từ những năm 1960 và 70 khi thế giới dữ liệu chỉ mới bắt đầu với những trung tâm dữ liệu đầu tiên và sự phát triển của cơ sở dữ liệu quan hệ.

Khoảng năm 2005, mọi người bắt đầu nhận ra lượng dữ liệu mà người dùng tạo ra thông qua Facebook, YouTube và các dịch vụ trực tuyến khác. Hadoop (một khuôn khổ mã nguồn mở được tạo đặc biệt để lưu trữ và phân tích các tập dữ liệu lớn) đã được phát triển cùng năm đó. NoSQL cũng bắt đầu trở nên phổ biến trong thời gian này.

Trong những năm kể từ đó, khối lượng dữ liệu lớn đã tăng vọt. Người dùng vẫn đang tạo ra một lượng lớn dữ liệu - nhưng không chỉ con người đang làm việc đó. Với sự ra đời của Internet of Things (IoT), nhiều đối tượng và thiết bị được kết nối với internet hơn, thu thập dữ liệu về cách sử dụng của khách hàng và hiệu suất sản phẩm. Sự xuất hiện của học máy đã tạo ra nhiều dữ liệu hơn.

2. Khái niệm về Dữ liệu lớn?

2.1. Dữ liệu là gì?

Dữ liệu là các đại lượng, ký tự hoặc ký hiệu mà máy tính thực hiện các hoạt động, có thể được lưu trữ và truyền dưới dạng tín hiệu điện và được ghi lại trên phương tiện ghi từ tính, quang học hoặc cơ học.

2.2. Dữ liệu lớn là gì?

Dữ liệu lớn là dữ liệu có chứa nhiều loại dữ liệu hơn, đến với khối lượng ngày càng tăng và với tốc độ nhanh hơn. Nói một cách đơn giản, dữ liệu lớn là những tập dữ liệu lớn hơn, phức tạp hơn, đặc biệt là từ các nguồn dữ liệu mới. Các tập dữ liệu này quá lớn đến nỗi phần mềm xử lý dữ liệu truyền thống không thể quản lý chúng.

2.3. Ví dụ về dữ liệu lớn

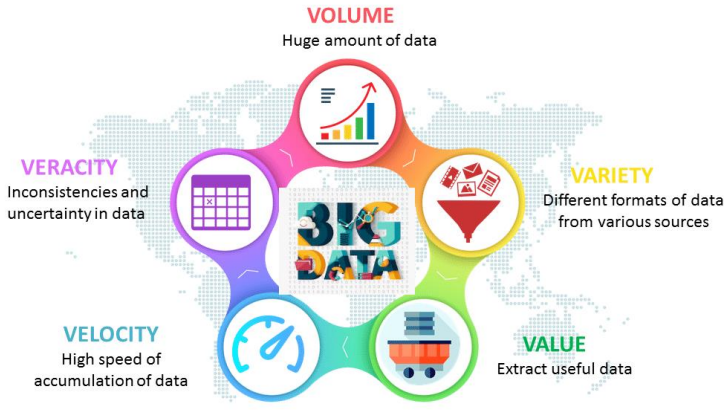
- Walmart xử lý hơn 1 triệu giao dịch của khách hàng mỗi giờ.

- Facebook lưu trữ, truy cập và phân tích hơn 30 Petabyte dữ liệu do người dùng tạo.
- Hơn 230 triệu tweet được tạo mỗi ngày.
- Hơn 5 tỷ người đang gọi điện, nhắn tin, viết tweet và duyệt web trên điện thoại di động trên toàn thế giới.
- Người dùng YouTube tải lên 48 giờ video mới mỗi phút trong ngày.
- Amazon xử lý 15 triệu lượt nhấp của khách hàng vào luồng dữ liệu người dùng mỗi ngày để giới thiệu sản phẩm.
- 294 tỷ email được gửi mỗi ngày. Các dịch vụ phân tích dữ liệu này để tìm ra các thư rác.
- Những chiếc xe hiện đại có gần 100 cảm biến theo dõi mức nhiên liệu, áp suất lốp, v.v., mỗi chiếc xe tạo ra rất nhiều dữ liệu cảm biến.

3. Các định dạng của Dữ liệu lớn

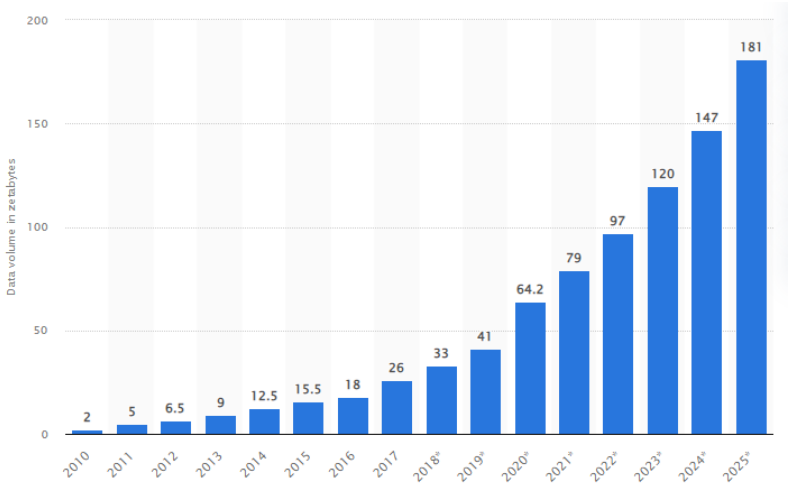
- *Có cấu trúc*: ví dụ tên, ngày tháng, địa chỉ, số thẻ tín dụng, thông tin chứng khoán, vị trí địa lý, v.v.
- *Không có cấu trúc*: ví dụ: tệp âm thanh, video, v.v.
- *Bán có cấu trúc*: ví dụ: XML, JSON

4. Đặc điểm của Dữ liệu lớn



Hình 1. 1 5 chữ V đặc điểm của Dữ liệu lớn

- ***Khối lượng (Volume)***: Kích thước của dữ liệu đóng một vai trò rất quan trọng trong việc xác định giá trị của dữ liệu. Một dữ liệu cụ thể có thực sự được coi là Dữ liệu lớn hay không, phụ thuộc vào khối lượng dữ liệu.



Hình 1. 2 Khối lượng dữ liệu/ thông tin được tạo, thu nhập, sao chép và sử dụng trên toàn thế giới từ năm 2010 đến 2025 ((https://www.statista.com/statistics/871513/worldwide-data-created/)

- ***Tốc độ (Velocity)***: Thuật ngữ 'tốc độ' đề cập đến tốc độ tạo ra dữ liệu. Tốc độ dữ liệu được tạo và xử lý để đáp ứng nhu cầu, xác định tiềm năng thực sự trong dữ liệu.
- ***Đa dạng (Variety)***: Đề cập đến các nguồn không đồng nhất và bản chất của dữ liệu, cả có cấu trúc và không có cấu trúc.
- ***Giá tri (Value):*** Đề cập đến giá trị mà dữ liệu lớn có thể cung cấp và nó liên quan trực tiếp đến những gì tổ chức có thể làm với dữ liệu được thu thập đó
- ***Tính xác thực (Veracity)***: đề cập đến chất lượng và độ chính xác của dữ liệu. Tính xác thực, về tổng thể, đề cập đến mức độ tin cậy có trong dữ liệu được thu thập.

5. Các ứng dụng của Dữ liệu lớn

Hầu như tất cả các ngành công nghiệp ngày nay đang tận dụng các ứng dụng Dữ liệu lớn theo cách này hay cách khác.

- *Chăm sóc sức khỏe thông minh hơn*: Sử dụng các petabyte dữ liệu của bệnh nhân, tổ chức có thể trích xuất thông tin có ý nghĩa và sau đó xây dựng các ứng dụng có thể dự đoán trước tình trạng xấu đi của bệnh nhân.

- *Bán lẻ*: Bán lẻ có một số tỷ suất lợi nhuận thấp nhất và là một trong những người hưởng lợi lớn nhất từ dữ liệu lớn. Nổi bật của việc sử dụng dữ liệu lớn trong bán lẻ là hiểu được hành vi của người tiêu dùng. Công cụ đề xuất của Amazon cung cấp gợi ý dựa trên lịch sử duyệt web của người tiêu dùng.
- *Sản xuất*: Phân tích dữ liệu lớn trong ngành sản xuất có thể giảm thiểu các lỗi thành phần, cải thiện chất lượng sản phẩm, tăng hiệu quả và tiết kiệm thời gian và tiền bạc.

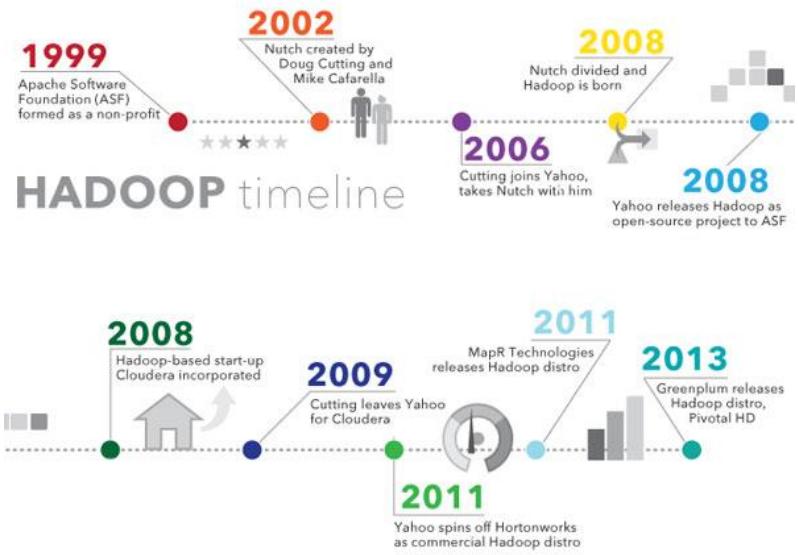
Ngoài ra còn nhiều ứng dụng sử dụng dữ liệu lớn để cách mạng hóa như: giải trí, giáo dục, ô tô không người lái, viễn thông, kiểm soát giao thông, v.v.

6. Thách thức của Dữ liệu lớn

- *Chất lượng dữ liệu*: Dữ liệu ở đây rất lộn xộn, không thống nhất và đầy đủ.
- *Khám phá*: Việc phân tích hàng petabyte dữ liệu bằng các thuật toán cực kỳ mạnh mẽ để tìm ra các mẫu và thông tin chi tiết là rất khó.
- *Lưu trữ*: Tổ chức càng có nhiều dữ liệu thì các vấn đề về quản lý tổ chức càng trở nên phức tạp hơn. Câu hỏi đặt ra ở đây là "Lưu trữ nó ở đâu?". Chúng ta cần một hệ thống lưu trữ có thể dễ dàng tăng hoặc giảm quy mô theo yêu cầu.
- *Phân tích*: Trong trường hợp Dữ liệu lớn, hầu hết thời gian chúng ta không biết về loại dữ liệu mà chúng ta đang xử lý, vì vậy việc phân tích dữ liệu đó thậm chí còn khó khăn hơn.
- *Bảo mật*: Vì dữ liệu có kích thước lớn nên việc giữ an toàn cho dữ liệu là một thách thức khác. Nó bao gồm xác thực người dùng, hạn chế quyền truy cập dựa trên người dùng, ghi lại lịch sử truy cập dữ liệu, sử dụng mã hóa dữ liệu thích hợp, v.v.
- *Công nghệ thay đổi nhanh chóng và thiếu nhân tài*: Việc bắt kịp với công nghệ dữ liệu lớn là một thách thức không ngừng và cần nhân lực để hiểu, vận dụng vào các dự án về dữ liệu lớn không nhiều.

II. Giới thiệu về Hadoop

1. Sự phát triển của Hadoop



Hình 2. 1 Sự phát triển của Hadoop qua các năm

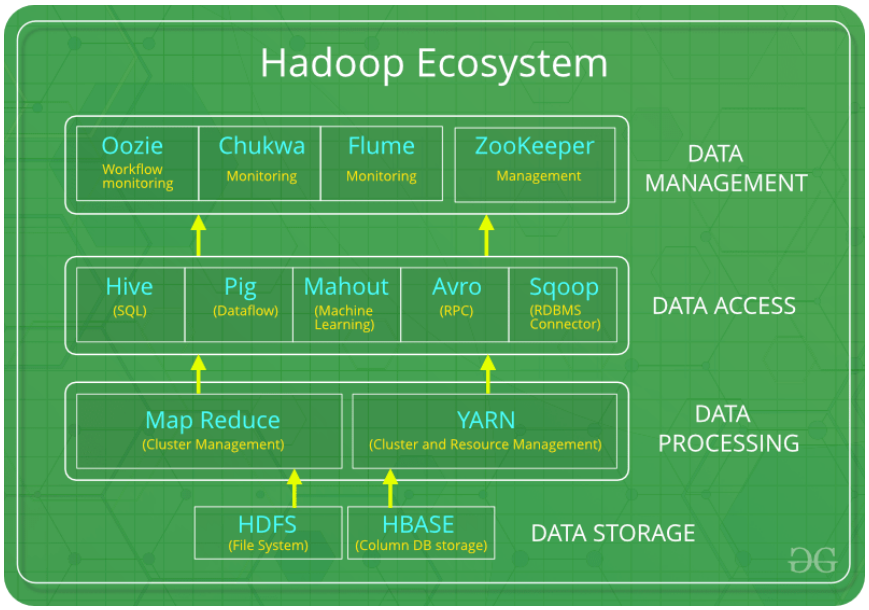
Năm 2003, Doug Cutting khởi động dự án Nutch để xử lý hàng tỷ lượt tìm kiếm và lập chỉ mục hàng triệu trang web. Cuối tháng 10 năm 2003 - Google phát hành bài báo với GFS (Google File System - Hệ thống tệp của Google). Vào tháng 12 năm 2004, Google phát hành các bài báo với MapReduce. Năm 2005, Nutch sử dụng GFS và MapReduce để thực hiện các hoạt động. Năm 2006, Yahoo đã tạo ra Hadoop dựa trên GFS và MapReduce với Doug Cutting và nhóm. Bạn sẽ ngạc nhiên nếu tôi nói với bạn rằng, vào năm 2007, Yahoo bắt đầu sử dụng Hadoop trên một cụm 1000 nút.

Sau đó vào tháng 1 năm 2008, Yahoo đã phát hành Hadoop như một dự án mã nguồn mở cho Apache Software Foundation. Vào tháng 7 năm 2008, Apache đã thử nghiệm thành công một cụm 4000 nút với Hadoop. Năm 2009, Hadoop đã sắp xếp thành công một petabyte dữ liệu trong vòng chưa đầy 17 giờ để xử lý hàng tỷ lượt tìm kiếm và lập chỉ mục hàng triệu trang web. Tiếp tục vào tháng 12 năm 2011, Apache Hadoop đã phát hành phiên bản 1.0. Sau đó vào tháng 8 năm 2013, Phiên bản 2.0.6 đã có sẵn.

2. Hadoop là gì?

Apache Hadoop là một framework mã nguồn mở, cho phép quản lý và giải quyết nhiều thách thức do dữ liệu lớn đặt ra. Giải pháp hiệu quả này phân phối sức mạnh lưu trữ và xử lý trên hàng nghìn nút trong một cụm.

3. Hệ sinh thái của Hadoop



Hình 2. 2 Hệ sinh thái của Hadoop (Nguồn: https://www.geeksforgeeks.org)

Hệ sinh thái Hadoop không phải là một ngôn ngữ lập trình cũng không phải là một dịch vụ, nó là một nền tảng hoặc framework giải quyết các vấn đề về dữ liệu lớn. Chúng ta có thể coi nó như một bộ bao gồm một số dịch vụ (nhập, lưu trữ, phân tích và bảo trì) bên trong nó.

Sau đây là các thành phần hình thành chung một hệ sinh thái Hadoop:

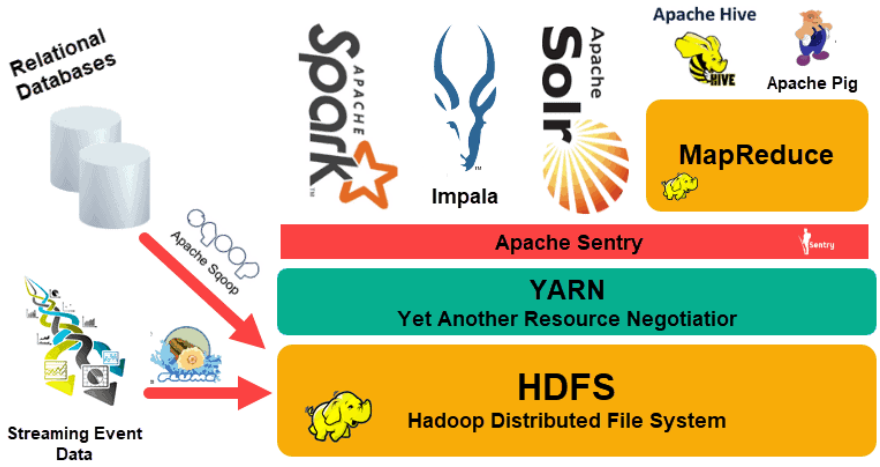
- *HDFS*: Hệ thống tệp phân tán Hadoop
- *YARN*: Một trình đàm phán tài nguyên khác
- *MapReduce*: Xử lý dữ liệu bằng lập trình
- *Spark*:Xử lý dữ liệu trong bộ nhớ
- *PIG , HIVE*: Dịch vụ xử lý dữ liệu sử dụng Truy vấn (giống SQL)
- *HBase*: Cơ sở dữ liệu NoSQL
- *Mahout , Spark MLlib*: Học máy

- *Apache Drill*: SQL trên Hadoop
- *Zookeeper*: Quản lý cụm
- *Oozie*: Lập lịch công việc
- *Flume , Sqoop*: Dịch vụ thay đổi dữ liệu
- *Solr & Lucene*: Tìm kiếm & Lập chỉ mục
- *Nhóm Ambari*: Cung cấp, Giám sát và Duy trì

Dựa trên các trường hợp sử dụng, chúng tôi có thể chọn một tập hợp các dịch vụ từ Hệ sinh thái Hadoop và tạo ra một giải pháp phù hợp cho một tổ chức

4. Kiến trúc của Hadoop

4.1. Tổng quan về kiến trúc Hadoop



Hình 2. 3 Kiến trúc của Hadoop

Hệ thống tệp phân tán Hadoop (HDFS - The Hadoop Distributed File System) , YARN và MapReduce là trung tâm của hệ sinh thái Hadoop. HDFS là một tập hợp các giao thức được sử dụng để lưu trữ các tập dữ liệu lớn, trong khi MapReduce xử lý dữ liệu đến một cách hiệu quả.

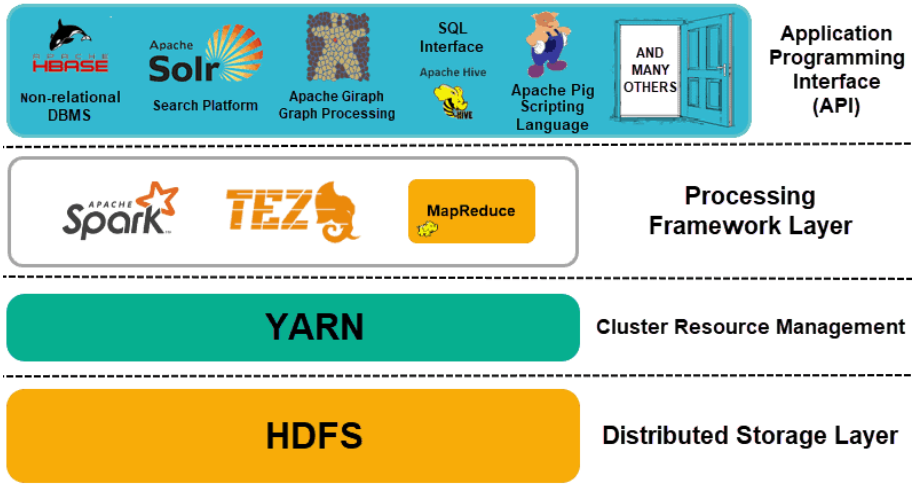
Một cụm Hadoop bao gồm một hoặc một số Nút chính (Master Nodes) và nhiều nút khác được gọi là Nút phụ (Slave Nodes). HDFS và MapReduce tạo thành một nền tảng linh hoạt có thể mở rộng tuyến tính bằng cách thêm các nút bổ sung.

YARN là viết tắt của cụm từ Yet Another Resource Negotiator, YARN đã được tạo ra để cải thiện quá trình lập lịch và quản lý tài nguyên trong một cụm Hadoop. Sự ra đời của YARN, với giao diện chung, đã mở ra cánh cửa cho các công cụ xử lý dữ liệu khác được tích hợp vào hệ sinh thái Hadoop.

Với HDFS, YARN và MapReduce làm cốt lõi của Hadoop, làm cho Hadoop trở thành giải pháp tối ưu để xử lý dữ liệu lớn.

4.2. Hiểu về các lớp của kiến trúc Hadoop

Việc tách các phần của hệ thống phân tán thành các lớp chức năng giúp hợp lý hóa việc quản lý và phát triển dữ liệu. Các nhà phát triển có thể làm việc trên các framework mà không ảnh hưởng tiêu cực đến các quy trình khác trên hệ sinh thái rộng lớn hơn.



Hình 2. 4 Các lớp của Hadoop

Hadoop có thể được chia thành 4 lớp đặc biệt:

- ***Lớp lưu trữ phân tán:***

Mỗi nút trong một cụm Hadoop có không gian đĩa, bộ nhớ, băng thông và quá trình xử lý riêng. Dữ liệu đến được chia thành các khối dữ liệu riêng lẻ, sau đó được lưu trữ trong lớp lưu trữ phân tán HDFS.

- ***Lớp quản lý tài nguyên cụm:***

Hadoop cần phối hợp các nút một cách hoàn hảo để vô số ứng dụng và người dùng chia sẻ tài nguyên của họ một cách hiệu quả. YARN là công cụ quản lý tài nguyên trên thực tế cho Hadoop.

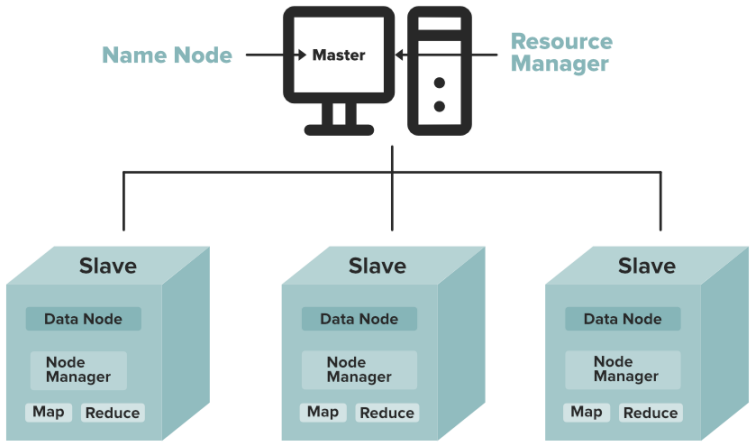
- ***Lớp framework xử lý:***

Lớp xử lý bao gồm các framework phân tích và xử lý các tập dữ liệu đi vào cụm. Các tập dữ liệu có cấu trúc và không có cấu trúc được ánh xạ, xáo trộn, sắp xếp, hợp nhất và thu gọn thành các khối dữ liệu nhỏ hơn có thể quản lý được.

- ***Giao diện lập trình ứng dụng:***

Sự ra đời của YARN trong Hadoop 2 đã dẫn đến việc tạo ra các framework và API mới. Với sự phát triển của dữ liệu, các công cụ với giao diện thân thiện với người dùng trở thành một phần của hệ sinh thái Hadoop.

5. Giải thích về HDFS



Hình 2. 5 Hệ thống tệp phân tán Hadoop – HDFS

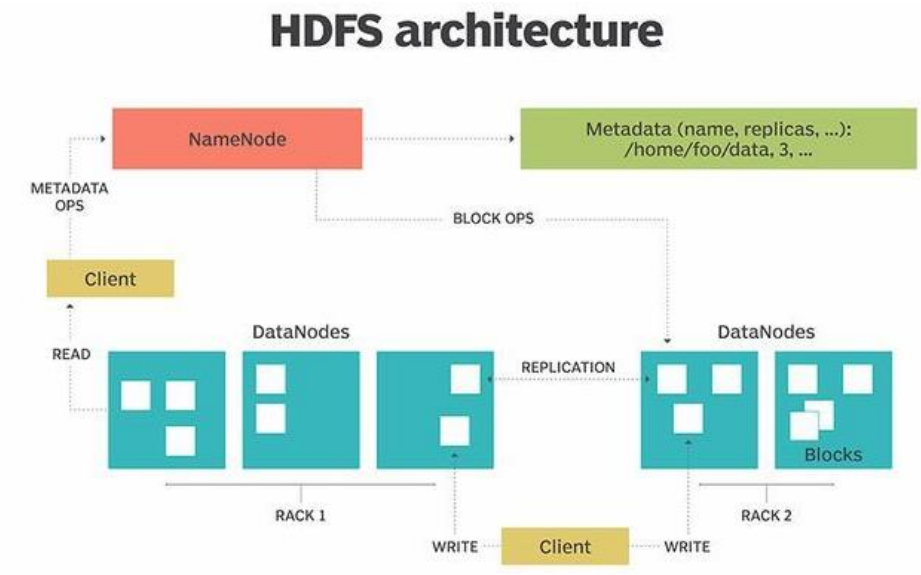
Apache HDFS hoặc Hệ thống tệp phân tán Hadoop là một hệ thống tệp có cấu trúc khối trong đó mỗi tệp được chia thành các khối có kích thước được xác định trước. Các khối này được lưu trữ trên một cụm gồm một hoặc một số máy. Apache Hadoop HDFS Architecture tuân theo Kiến trúc Master / Slave, trong đó một cụm bao gồm một NameNode duy nhất (nút Master) và tất cả các nút khác là DataNodes (nút Slave).HDFS có thể được triển khai trên nhiều máy hỗ trợ Java. Mặc dù người

ta có thể chạy một số DataNodes trên một máy, nhưng trong thế giới thực tế, các DataNodes này được trải rộng trên nhiều máy khác nhau.

5.1. Mục tiêu của HDFS

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

5.2. Kiến trúc HDFS



Hình 2. 6 Kiến trúc của HDFS

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

5.2.1. NameNode

HDFS chỉ bao gồm duy nhất một NamenNode được gọi là master node thực hiện các nhiệm vụ:

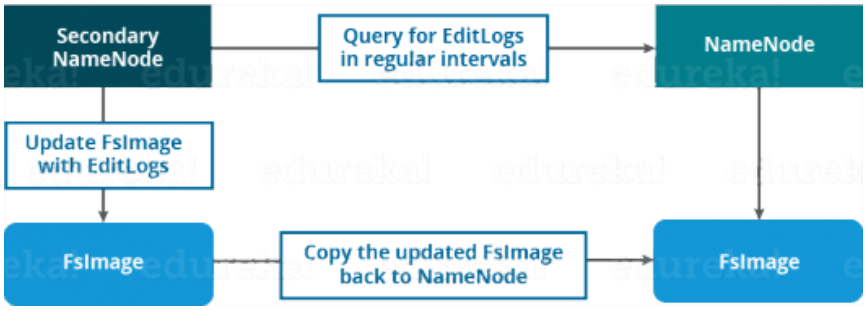
- Lưu trữ siêu dữ liệu (metadata) của dữ liệu thực tế (tên, đường dẫn, blocks id, cấu hình DataNode vị trí blocks,…). Có hai tệp được liên kết với siêu dữ liệu:
  - *FsImage*: Nó chứa trạng thái hoàn chỉnh của không gian tên hệ thống tệp kể từ khi bắt đầu NameNode.
  - *EditLogs*: Nó chứa tất cả các sửa đổi gần đây được thực hiện đối với hệ thống tệp liên quan đến FsImage mới nhất.
- Quản lý không gian tên của hệ thống file (ánh xạ các tên tệp với các blocks, ánh xạ các block vào các DataNode)
- Quản lý cấu hình của cụm
- Chỉ định công việc cho DataNode

5.2.2. DataNode

Chức năng của Datanode:

- Lưu trữ dữ liệu thực tế
- Trực tiếp thực hiện và xử lý công việc (đọc/ghi dữ liệu)

5.2.3. Secondary NameNode



Hình 2. 7 Chu trình hoạt động của Secondary NameNode

Secondary Namenode là một node phụ chạy cùng với Namenode, nhìn tên gọi nhiều người nhầm tưởng rằng nó để dự phòng cho Namenode tuy nhiên không phải vậy, Secondary Namenode như là một trợ lý đắc lực của Namenode, có vai trò và nhiệm vụ rõ ràng:

- Nó thường xuyên đọc các tệp, các siêu dữ liệu được lưu trên RAM của NameNode và ghi vào ổ cứng.
- Nó tải xuống các EditLogs từ NameNode theo khoảng thời gian đều đặn và cập nhật cho FsImage. FsImage mới được sao chép trở lại NameNode, được sử dụng bất cứ khi nào NameNode được khởi động vào lần tiếp theo.

5.2.4. Cơ chế heartbeat

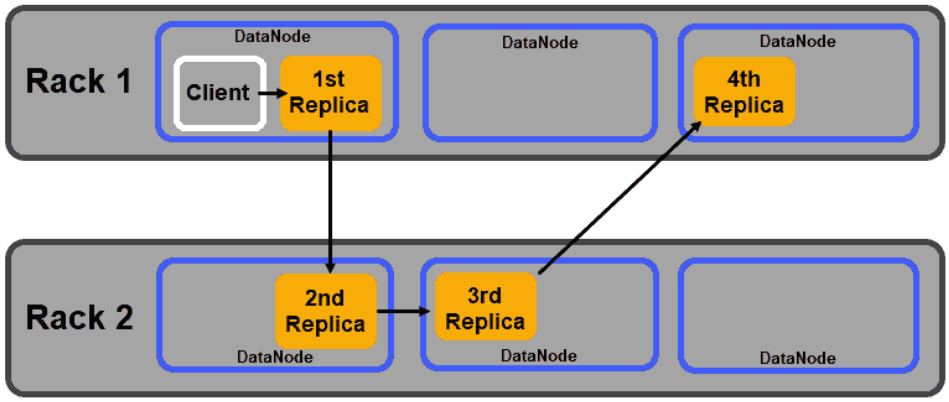
Heartbeat là cách liên lạc hay là cách để DataNode cho NameNode biết là nó còn sống. Định kì DataNode sẽ gửi một heartbeat về cho NameNode để NameNode biết là DataNode đó còn hoạt động. Nếu DataNode không gửi heartbeat về cho NameNode thì NameNode coi rằng nút đó đã hỏng và không thể thực hiện nhiệm vụ được giao. Namenode sẽ phân công nhiệm vụ đó cho một DataNode khác.

5.2.5. Các khối – Blocks

Blocks là một đơn vị lưu trữ của HDFS, các data được đưa vào HDFS sẽ được chia thành các block có các kích thước cố định. Kích thước mặc định của mỗi khối là 128 MB trong Apache Hadoop 2 . x (64 MB trong Apache Hadoop 1 .x) mà bạn có thể định cấu hình theo yêu cầu của mình..

HDFS sẽ không tốt khi xử lý một lượng lớn các file nhỏ. Mỗi dữ liệu lưu trữ trên HDFS được đại diện bằng một blocks với kích thước là 128MB, vậy nếu lưu trữ lượng lớn file nhỏ thì sẽ cần một lượng lớn các block để lưu chữ chúng và mỗi block chúng ta chỉ cần dùng tới một ít và còn thừa rất nhiều dung lượng gây ra sự lãng phí. Chúng ta cũng có thể thấy là block size của hệ thống file ở các hệ điều hành tiêu biểu như linux là 4KB là rất bé so với 128MB.

5.2.6. Chính sách về vị trí của Rack Aware



Hình 2. 8 Chính sách về vị trí tạo bản sao Rack Aware

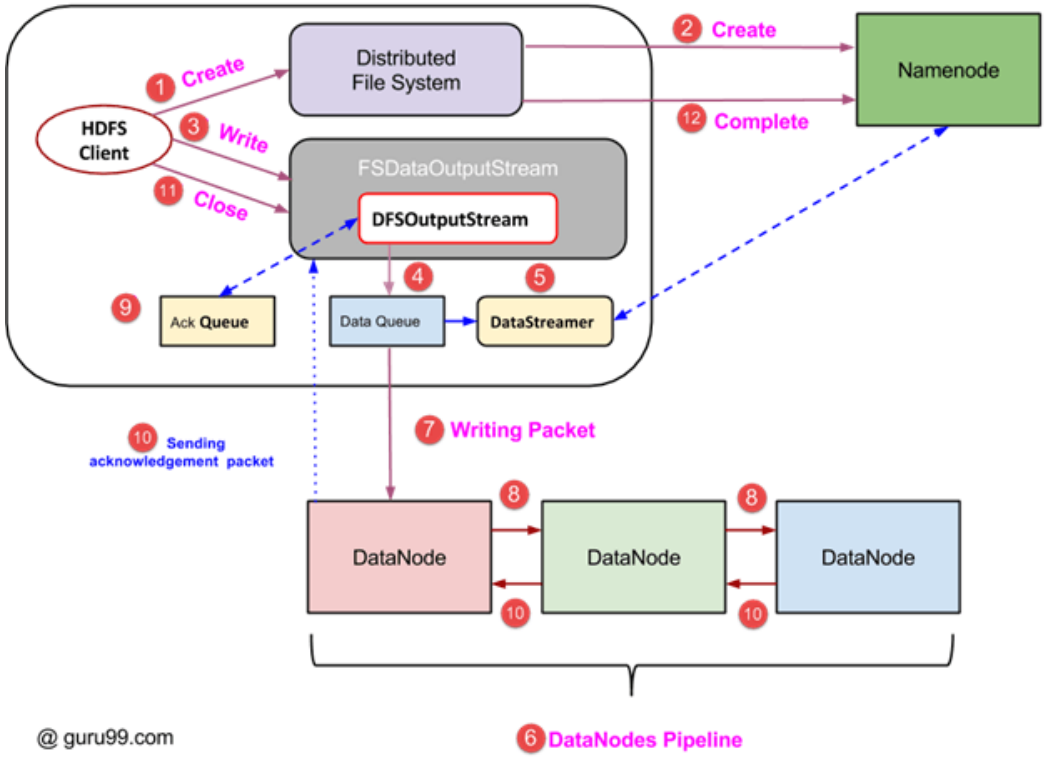
Một trong những mục tiêu chính của hệ thống lưu trữ phân tán như HDFS là duy trì tính khả dụng và nhân rộng cao. Do đó, các khối dữ liệu cần được phân phối không chỉ trên các DataNodes khác nhau mà trên các nút nằm trên các server rack khác nhau.

Điều này đảm bảo rằng sự cố của toàn bộ giá đỡ không chấm dứt tất cả các bản sao dữ liệu. HDFS NameNode duy trì chính sách đặt bản sao rack-aware mặc định:

- Bản sao khối dữ liệu đầu tiên được đặt trên cùng một nút với máy khách.
- Bản sao thứ hai được đặt tự động trên một DataNode ngẫu nhiên trên một giá đỡ khác.
- Bản sao thứ ba được đặt trong một DataNode riêng biệt trên cùng một giá đỡ với bản sao thứ hai.
- Bất kỳ bản sao bổ sung nào được lưu trữ trên các Mã dữ liệu ngẫu nhiên trong toàn bộ cụm.

5.3. Hoạt động Ghi/ Đọc dữ liệu của HDFS

5.3.1. *Ghi dữ liệu*



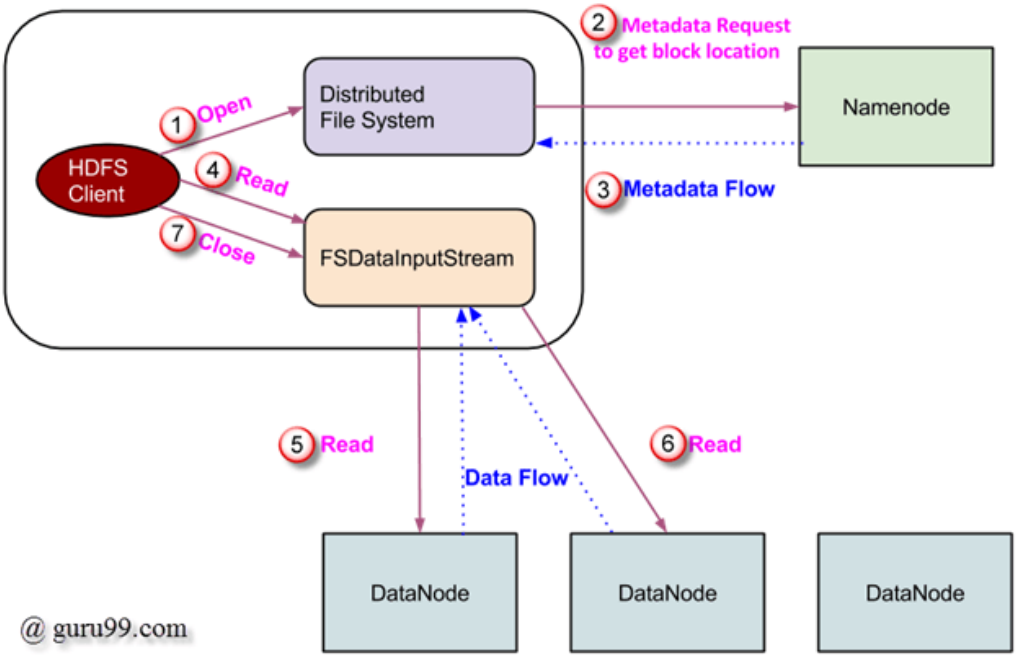
Hình 2. 9 Ghi (Write) dữ liệu trong HDFS

Ghi dữ liệu được thực hiện theo các bước dưới đây (tương ứng với sơ đồ Hình….)

1. Khách (Client) gọi phương thức 'create ()' ở DistributedFileSystem để tạo một tệp mới.
2. Đối tượng DistributedFileSystem kết nối với NameNode để bắt đầu tạo tệp mới. NameNode kiểm tra quyền của máy khách và kiểm tra tệp có tồn tại hay không. Nếu tệp đã tồn tại hoặc máy khách không có đủ quyền để tạo tệp mới, thì IOException sẽ được chuyển đến máy khách. Nếu không, thao tác sẽ thành công và một bản ghi mới cho tệp được tạo bởi NameNode.
3. Khi một bản ghi mới trong NameNode được tạo, một đối tượng kiểu FSDataOutputStream được trả lại cho máy khách để ghi dữ liệu vào HDFS. Phương thức ghi dữ liệu được gọi (bước 3 trong sơ đồ).

4. FSDataOutputStream chứa đối tượng DFSOutputStream sau khi giao tiếp với DataNodes và NameNode. Trong khi máy khách tiếp tục ghi dữ liệu, DFSOutputStream tiếp tục tạo các gói với dữ liệu này. Các gói này được xếp vào hàng đợi được gọi là DataQueue .
5. Có một thành phần nữa được gọi là DataStreamer sử dụng DataQueue này . DataStreamer cũng yêu cầu NameNode phân bổ các khối mới, từ đó chọn các DataNode mong muốn được sử dụng để nhân rộng.
6. Bây giờ, quá trình sao chép bắt đầu bằng cách tạo một đường dẫn sử dụng DataNodes. Trong ví dụ trên, chúng ta đã chọn mức sao chép là 3 và do đó có 3 DataNodes trong đường dẫn.
7. DataStreamer đổ các gói vào DataNode đầu tiên trong đường dẫn.
8. Mọi DataNode trong một đường ống đều lưu trữ gói tin mà nó nhận được và chuyển tiếp cùng một gói dữ liệu đến DataNode thứ hai trong một đường ống.
9. Một hàng đợi khác, 'Ack Queue' được DFSOutputStream duy trì để lưu trữ các gói đang chờ xác nhận từ DataNodes.
10. Sau khi xác nhận cho một gói trong hàng đợi được nhận từ tất cả các Mã dữ liệu trong đường ống, nó sẽ bị xóa khỏi 'Hàng đợi Ack'. Trong trường hợp có bất kỳ lỗi DataNode nào, các gói từ hàng đợi này sẽ được sử dụng để khởi động lại hoạt động.
11. Sau khi một máy khách hoàn tất việc ghi dữ liệu, nó sẽ gọi một phương thức close() (Bước 9 trong sơ đồ) Gọi tới close (), dẫn đến việc đẩy các gói dữ liệu còn lại vào đường ống, sau đó chờ xác nhận.
12. Sau khi nhận được xác nhận cuối cùng, NameNode sẽ được liên hệ để thông báo rằng thao tác ghi tệp đã hoàn tất.

5.3.2. *Đọc dữ liệu*



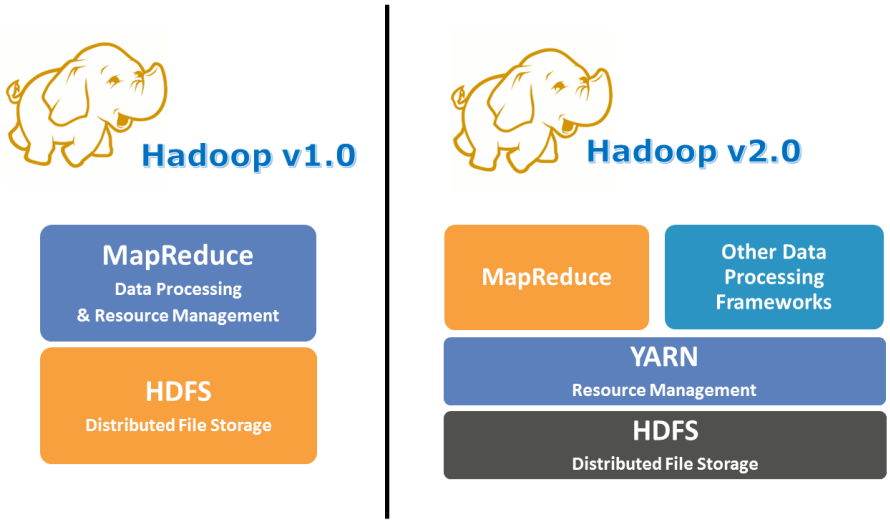
Hình 2. 10 Đọc (Read) dữ liệu trong HDFS

1. Máy khách khởi tạo yêu cầu đọc bằng cách gọi phương thức 'open()' của đối tượng FileSystem; nó là một đối tượng kiểu DistributedFileSystem .
2. Đối tượng này kết nối với NameNode và nhận thông tin siêu dữ liệu như vị trí của các khối của tệp.
3. Đáp lại yêu cầu siêu dữ liệu này, địa chỉ của các DataNode có bản sao của khối đó sẽ được trả lại.
4. Khi địa chỉ của các DataNode được nhận, một đối tượng kiểu FSDataInputStream được trả lại cho máy khách. FSDataInputStream chứa DFSInputStream xử lý các tương tác với DataNode và NameNode. Một máy khách gọi phương thức 'read()' khiến DFSInputStream thiết lập kết nối với DataNode đầu tiên với khối đầu tiên của tệp.
5. Dữ liệu được đọc ở dạng luồng trong đó ứng dụng khách gọi phương thức 'read()' lặp đi lặp lại. Quá trình hoạt động read() này tiếp tục cho đến khi nó đi đến cuối khối.

6. Khi đến cuối khối, DFSInputStream đóng kết nối và chuyển sang định vị Mã dữ liệu tiếp theo cho khối tiếp theo
7. Khi khách hàng đã thực hiện xong việc đọc, nó sẽ gọi một phương thức close() .

6. Giải thích về YARN

Trong Hadoop phiên bản 1.0 còn được gọi là MRV1 (MapReduce Phiên bản 1), MapReduce thực hiện cả chức năng xử lý và quản lý tài nguyên. YARN được giới thiệu trong Hadoop phiên bản 2.0 vào năm 2012 bởi Yahoo và Hortonworks. Ý tưởng cơ bản đằng sau YARN là giảm bớt MapReduce bằng cách đảm nhận trách nhiệm Quản lý tài nguyên và Lập lịch công việc. Với sự ra đời của YARN, hệ sinh thái Hadoop đã hoàn toàn được cách mạng hóa. Nó trở nên linh hoạt, hiệu quả và có thể mở rộng hơn nhiều



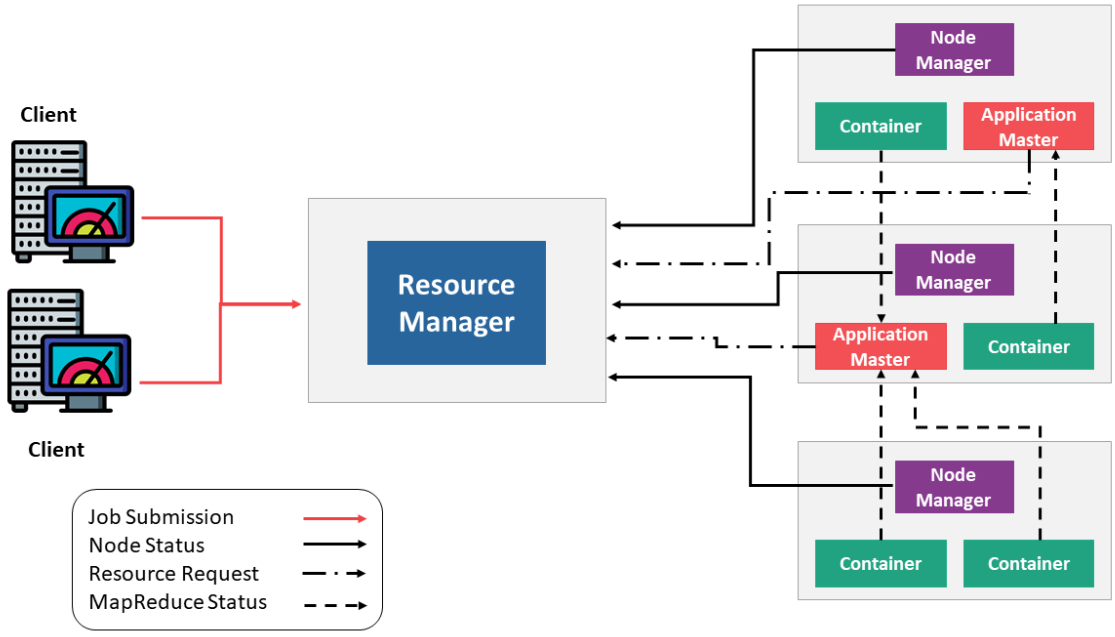
Hình 2. 11 Sự khác nhau của Hadoop v1.0 và Hadoop v2.0

Kiến trúc Apache Hadoop YARN bao gồm các thành phần chính sau:

- *Trình quản lý tài nguyên (Resource Manager)*: Chạy trên trình nền chính và quản lý việc phân bổ tài nguyên trong cụm.
- *Trình quản lý nút (Node Manager)*: Chúng chạy trên các daemon nô lệ và chịu trách nhiệm thực hiện một tác vụ trên mỗi Nút dữ liệu.

- *Ứng dụng chính (Application Master):* Quản lý vòng đời công việc của người dùng và nhu cầu tài nguyên của các ứng dụng riêng lẻ. Nó hoạt động cùng với Node Manager và giám sát việc thực thi các tác vụ.
- *Vùng chứa (Container):* Gói tài nguyên bao gồm RAM, CPU, Mạng, HDD, v.v. trên một nút duy nhất.

6.1. Thành phần của YARN



Hình 2. 12 Kiến trúc của YARN

6.1.1. Resource Manager

Resource Manager kiểm soát tất cả các tài nguyên xử lý trong một cụm Hadoop. Mục đích chính của nó là chỉ định tài nguyên cho các ứng dụng riêng lẻ nằm trên các nút phụ. Nó duy trì một cái nhìn tổng thể về các quy trình đang diễn ra và được lên kế hoạch, xử lý các yêu cầu tài nguyên, lập lịch trình và phân công tài nguyên cho phù hợp

Resource Manager có hai thành phần chính là: Scheduler (bộ lập lịch) và Application Manager (trình quản lý ứng dụng).

- *Scheduler*: Chịu trách nhiệm phân bổ tài nguyên cho các ứng dụng đang chạy khác nhau tùy thuộc vào các ràng buộc về dung lượng, hàng đợi, v.v.
- *Application Manager*: Có trách nhiệm chấp nhận các báo cáo công việc. Quản lý việc chạy Application Master trong một cụm và cung cấp dịch vụ khởi động lại container khi Application Master bị lỗi.

6.1.2. Node Manager

Nằm dưới sự kiểm soát của Resource Manager. Chức năng chính của Node Manager là theo dõi dữ liệu tài nguyên xử lý trên nút phụ của nó và gửi báo cáo thường xuyên đến Resource Manager.

6.1.3. Application Master

Mỗi vùng chứa trên một nút phụ đều có Application Master chuyên dụng của nó. Ứng dụng Master cũng được triển khai trong một vùng chứa.

Miễn là Application Master đang hoạt động, Application Master sẽ gửi thông báo đến Resource Manager về trạng thái hiện tại của nó và trạng thái của ứng dụng mà nó giám sát. Dựa trên thông tin được cung cấp, Trình quản lý tài nguyên lên lịch cho các tài nguyên bổ sung hoặc chỉ định chúng ở nơi khác trong cụm nếu chúng không còn cần thiết.

Application Master giám sát toàn bộ vòng đời của một ứng dụng, từ việc yêu cầu các vùng chứa cần thiết từ Resource Manager đến việc gửi các yêu cầu thuê vùng chứa cho Node Manager.

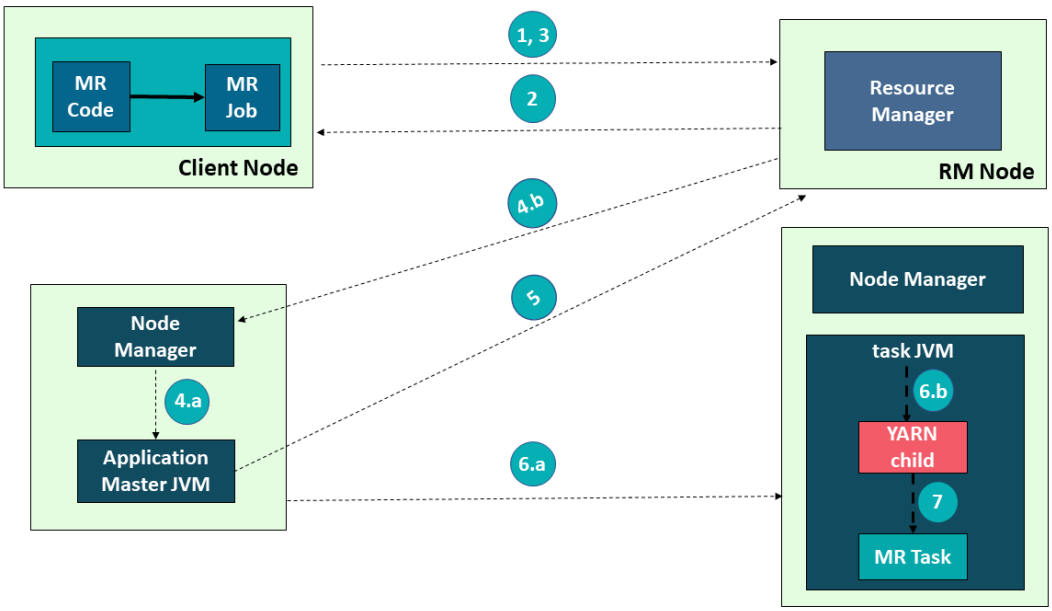
6.1.4. Container

Các tài nguyên xử lý trong một cụm Hadoop luôn được triển khai trong các Container (vùng chứa). Một vùng chứa có bộ nhớ, tệp hệ thống và không gian xử lý.

Triển khai vùng chứa là chung và có thể chạy bất kỳ tài nguyên tùy chỉnh nào được yêu cầu trên bất kỳ hệ thống nào. Nếu lượng tài nguyên cụm được yêu cầu nằm trong giới hạn có thể chấp nhận được, Resource Manager sẽ phê duyệt và lên lịch triển khai vùng chứa đó.

6.2. Nộp đăng ký trong YARN

Các bước liên quan đến việc nộp đơn đăng ký (application submission) trong Hadoop YARN:



Hình 2. 13 Nộp đăng ký trong YARN

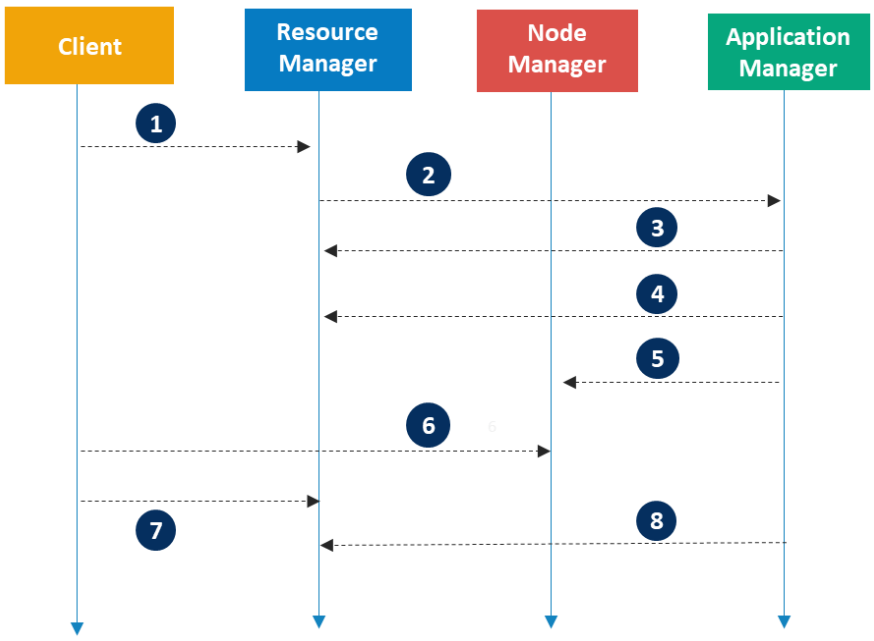
1. Gửi công việc
2. Nhận ID ứng dụng
3. Bối cảnh nộp đơn đăng ký
4. a) Bắt đầu khởi chạy vùng chứa

   b) Khởi chạy Ứng dụng Master
5. Phân bổ nguồn lực
6. a) Thùng chứa

   b) Khởi chạy
7. Thực thi

6.3. Quy trình làm việc của Hadoop YARN

Các bước sau liên quan đến quy trình làm việc của Apache Hadoop YARN



Hình 2. 14 Quy trình làm việc trong YARN

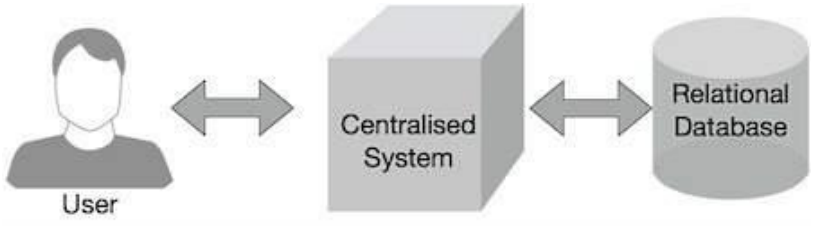
1. Client (máy khách) nộp đơn đăng ký.
2. Resource Manager (trình quản lý tài nguyên) phân bổ một container (vùng chứa) để khởi động Application Manager (Trình quản lý ứng dụng).
3. Application Manager đăng ký với Resource Manager.
4. Application Manager yêu cầu vùng chứa từ Resource Manager.
5. Application Manager thông báo Node Manager khởi chạy vùng chứa.
6. Mã ứng dụng được thực thi trong vùng chứa.
7. Client liên hệ Resource Manager/ Application Manager để theo dõi trạng thái của ứng dụng.
8. Application Manager hủy đăng ký với Resource Manager.

7. Giải thích MapReduce

7.1. Tại sao lại sử dụng MapReduce

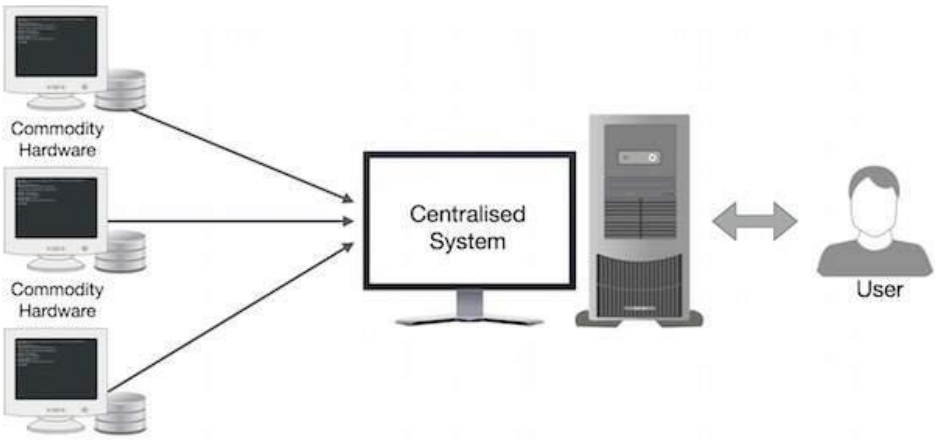
Hệ thống Doanh nghiệp truyền thống thường có một máy chủ tập trung để lưu trữ và xử lý dữ liệu (Hình 2. 15). Mô hình truyền thống chắc chắn không phù hợp để xử lý khối lượng lớn dữ liệu có thể mở rộng và không thể đáp ứng được bởi các máy chủ cơ sở dữ liệu tiêu chuẩn. Hơn nữa, hệ thống tập trung tạo ra quá

nhiều nút cổ chai (*điểm tắc nghẽn trong hệ thống*) trong khi xử lý nhiều tệp đồng thời.



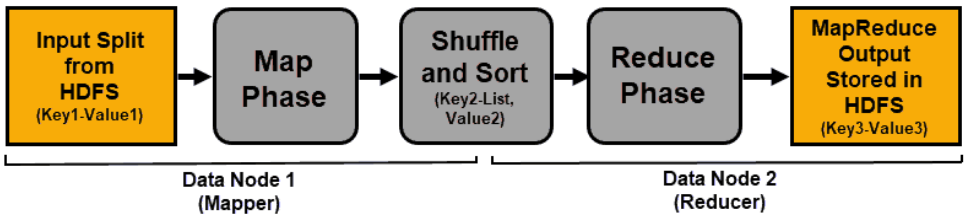
Hình 2. 15 Xử lý dữ liệu kiểu truyền thống

Google đã giải quyết vấn đề tắc nghẽn này bằng cách sử dụng một thuật toán có tên là MapReduce. MapReduce chia một nhiệm vụ thành các phần nhỏ và giao chúng cho nhiều máy tính. Sau đó, kết quả được thu thập tại một nơi và được tích hợp để tạo thành tập dữ liệu kết quả.



Hình 2. 16 Mô hình xử lý dữ liệu theo MapReduce

7.2. Hoạt động của MapReduce

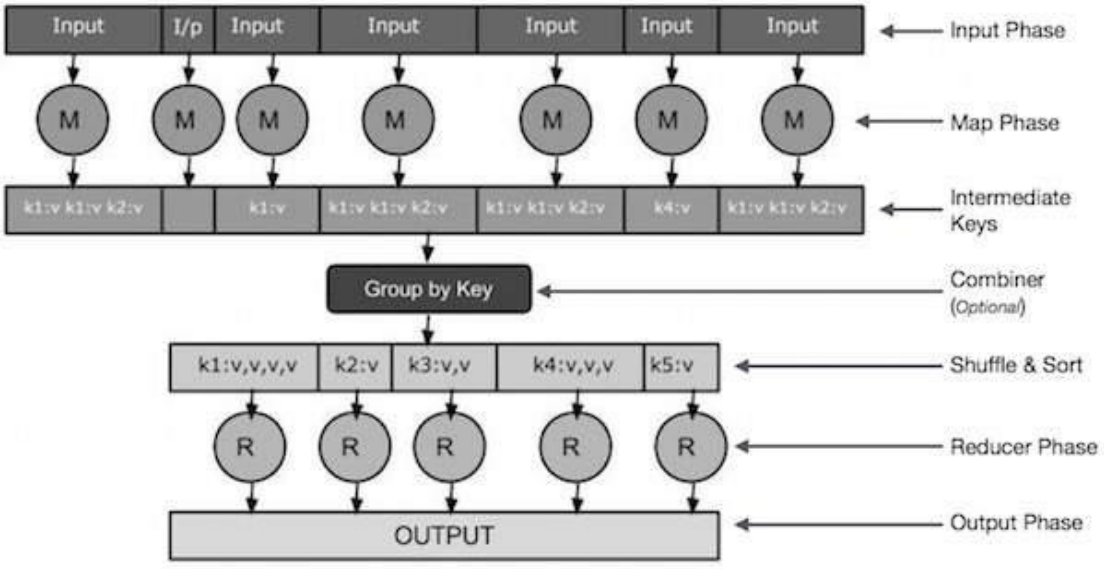


Hình 2. 17 Hoạt động của MapReduce

Thuật toán MapReduce chứa hai nhiệm vụ quan trọng, đó là Map (ánh xạ) và Reduce (rút gọn).

- Tác vụ Map nhận một tập hợp dữ liệu và chuyển đổi nó thành một tập dữ liệu khác, trong đó các phần tử riêng lẻ được chia nhỏ thành các bộ giá trị (key-value hay cặp khóa-giá trị).
- Tác vụ Reduce lấy đầu ra từ Bản đồ làm đầu vào và kết hợp các bộ dữ liệu đó (cặp khóa-giá trị) thành một bộ bộ giá nhỏ hơn. Tác vụ Reduce luôn được thực hiện sau tác vụ Map.

Tìm hiểu kỹ hơn về các giai đoạn trong MapReduce

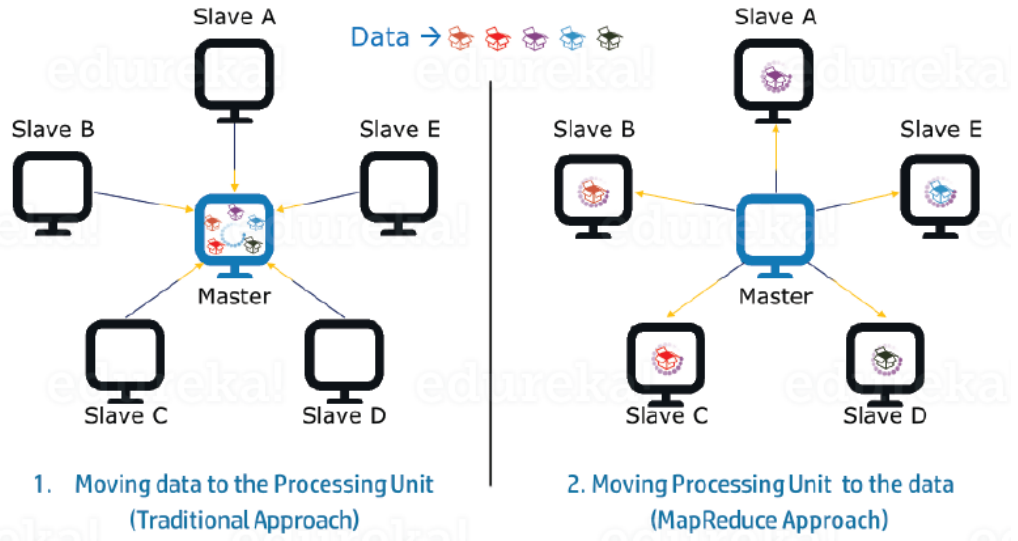


Hình 2. 18 Chi tiết các thành phần và giai đoạn trong MapReduce

- ***Giai đoạn đầu vào (Input Phase)***: Ở đây chúng ta có Trình đọc bản ghi (Record Reader) dịch từng bản ghi trong tệp đầu vào và gửi dữ liệu đã phân tích cú pháp tới trình ánh xạ (mapper) dưới dạng các cặp khóa-giá trị (key-value).
- ***Giai đoạn ánh xạ (Map Phase)***: Map là một chức năng do người dùng định nghĩa, lấy một loạt các cặp khóa-giá trị và xử lý từng cặp một trong số chúng để tạo ra không hoặc nhiều cặp khóa-giá trị.

- ***Khóa trung gian (Intermediate Keys)***: Các cặp khóa-giá trị do trình ánh xạ tạo ra được gọi là khóa trung gian.
- ***Bộ kết hợp (Combiner)***: Combiner là một loại Reducer cục bộ kết hợp dữ liệu tương tự từ giai đoạn Map thành các bộ có thể nhận dạng. Nó lấy các khóa trung gian từ trình ánh xạ làm đầu vào và áp dụng mã do người dùng xác định để tổng hợp các giá trị trong một phạm vi nhỏ của một trình ánh xạ. Nó không phải là một phần của thuật toán MapReduce chính; nó là tùy chọn.
- ***Xáo trộn và Sắp xếp (Shuffle & Sort)***: Tác vụ Reducer bắt đầu với bước Xáo trộn và Sắp xếp. Nó tải các cặp khóa-giá trị đã kết hợp vào máy cục bộ, nơi Reducer đang chạy. Các cặp khóa-giá trị riêng lẻ được sắp xếp theo khóa thành một danh sách dữ liệu lớn hơn. Danh sách dữ liệu nhóm các khóa tương đương với nhau để các giá trị của chúng có thể được lặp lại một cách dễ dàng trong tác vụ Reducer.
- ***Bộ rút gọn (Reducer)***: Reducer lấy dữ liệu được ghép nối khóa-giá trị đã kết hợp làm đầu vào và chạy chức năng Reducer trên mỗi một trong số chúng. Tại đây, dữ liệu có thể được tổng hợp, lọc và kết hợp theo một số cách và nó yêu cầu nhiều quá trình xử lý. Khi quá trình thực thi kết thúc, nó không cung cấp hoặc nhiều cặp khóa-giá trị cho bước cuối cùng.
- ***Giai đoạn đầu ra (Output Phase)***: Trong giai đoạn đầu ra, chúng ta có một trình định dạng đầu ra để dịch các cặp khóa-giá trị cuối cùng từ chức năng Reducer và ghi chúng vào một tệp bằng cách sử dụng trình ghi bản ghi.

7.3. Ưu điểm của MapReduce



Hình 2. 19 Ưu điểm xử lý song song của MapReduce

- ***Xử lý dữ liệu song song:***

Trong MapReduce, chúng ta phân chia công việc cho nhiều nút và mỗi nút hoạt động đồng thời với một phần công việc. Vì vậy, MapReduce dựa trên mô hình Chia và Chinh phục (*Divide and Conquer*), giúp chúng ta xử lý dữ liệu bằng các máy khác nhau. Vì dữ liệu được xử lý bởi nhiều máy thay vì một máy song song, thời gian thực hiện để xử lý dữ liệu sẽ giảm đi rất nhiều.

- ***Vị trí dữ liệu:***

Thay vì di chuyển dữ liệu đến đơn vị xử lý, chúng ta đang chuyển đơn vị xử lý sang dữ liệu trong MapReduce. Vì vậy, như bạn có thể thấy trong Hình 2.19 rằng dữ liệu được phân phối giữa nhiều nút, nơi mỗi nút xử lý một phần dữ liệu nằm trên đó. Điều này cho phép chúng tôi có những lợi thế sau:

- Tiết kiệm chi phí khi di chuyển đơn vị xử lý vào dữ liệu.
- Thời gian xử lý được giảm xuống vì tất cả các nút đang làm việc song song với một phần dữ liệu của chúng.
- Mỗi nút đều có một phần dữ liệu để xử lý và do đó, không có khả năng nút bị quá tải.

8. Ưu điểm và Hạn chế của Hadoop

8.1. Ưu điểm

- ***Chi phí:***

Hadoop là mã nguồn mở và sử dụng phần cứng thương mại hiệu qả, cung cấp một mô hình hiệu quả về chi phí, không giống như cơ sở dữ liệu Quan hệ truyền thống yêu cầu phần cứng đắt tiền và bộ xử lý cao cấp để xử lý Dữ liệu lớn

- ***Khả năng mở rộng:***

Hadoop là một mô hình có khả năng mở rộng cao. Một lượng lớn dữ liệu được chia thành nhiều máy rẻ tiền trong một cụm được xử lý song song. Số lượng các máy hoặc nút này có thể được tăng hoặc giảm theo yêu cầu của doanh nghiệp.

- ***Tính linh hoạt:***

Hadoop được thiết kế theo cách mà nó có thể xử lý bất kỳ loại tập dữ liệu nào như có cấu trúc (Dữ liệu MySql), Bán cấu trúc (XML, JSON), Không có cấu trúc (Hình ảnh và Video) rất hiệu quả.

- ***Tốc độ:***

Khi bạn đang xử lý một lượng lớn dữ liệu không có cấu trúc, tốc độ là một yếu tố quan trọng, với Hadoop, bạn có thể dễ dàng truy cập dữ liệu của TB chỉ trong vài phút.

- ***Khả năng chịu lỗi:***

Hadoop sử dụng phần cứng thương mại (hệ thống rẻ tiền) có thể bị hỏng bất cứ lúc nào. Trong Hadoop, dữ liệu được sao chép trên các DataNode khác nhau trong một cụm Hadoop, đảm bảo tính khả dụng của dữ liệu nếu bằng cách nào đó bất kỳ hệ thống nào của bạn bị lỗi.

- ***Thông lượng cao:***

Hadoop hoạt động trên Hệ thống tệp phân tán nơi các công việc khác nhau được giao cho các nút Dữ liệu khác nhau trong một cụm, thanh dữ liệu này được xử lý song song trong cụm Hadoop tạo ra thông lượng cao.

- ***Lưu lượng mạng tối thiểu:***

Trong Hadoop, mỗi nhiệm vụ được chia thành nhiều nhiệm vụ con nhỏ khác nhau, sau đó được gán cho từng nút dữ liệu có sẵn trong cụm Hadoop. Mỗi nút dữ liệu xử lý một lượng nhỏ dữ liệu dẫn đến lưu lượng truy cập thấp trong một cụm Hadoop.

8.2. Hạn chế

- ***Sự cố với tệp nhỏ:***

Hadoop có thể thực hiện hiệu quả trên một số lượng nhỏ các tệp có kích thước lớn. Hadoop bị lỗi khi cần truy cập tệp kích thước nhỏ với số lượng lớn. Quá nhiều tệp nhỏ này làm phụ phí Namenode và gây khó khăn cho hoạt động.

- ***Lổ hổng:***

Hadoop là một framework được viết bằng java và java là một trong những ngôn ngữ lập trình được sử dụng phổ biến nhất, điều này làm cho nó trở nên không an toàn hơn vì nó có thể dễ dàng bị khai thác bởi bất kỳ tội phạm mạng nào.

- ***Hiệu suất thấp trong môi trường dữ liệu nhỏ:***

Hiệu quả của Hadoop giảm khi hoạt động trong môi trường dữ liệu nhỏ vì Hadoop chủ yếu được thiết kế để xử lý các bộ dữ liệu lớn.

- ***Chỉ hỗ trợ xử lý hàng loạt:***

Về cốt lõi, Hadoop có một công cụ xử lý hàng loạt không hiệu quả trong việc xử lý luồng. Nó không thể tạo ra đầu ra trong thời gian thực với độ trễ thấp. Nó chỉ hoạt động trên dữ liệu mà chúng tôi thu thập và lưu trữ trước trong một tệp trước khi xử lý.

III. Thực nghiệm

1. Vấn đề

Để mô tả hoạt động của Hadoop – MapReduce, nhóm chọn ví dụ Word Count – Đếm tù để thực nghiệm. Đếm từ chỉ đơn giản là đếm số từ trong một tập dữ liệu và đây là một ví dụ điển hình để mô tả và giải thích cách hoạt động của MapReduce dễ dàng.

2. Dữ liệu

Dữ liệu được nhóm Crawl trên trang báo The Sun (https://www.thesun.co.uk) của nước Anh. Dữ liệu được crawl gồm tiêu đề và nội dung bài viết trong mục World News (https://www.thesun.co.uk/news/worldnews/) trong năm 2022.

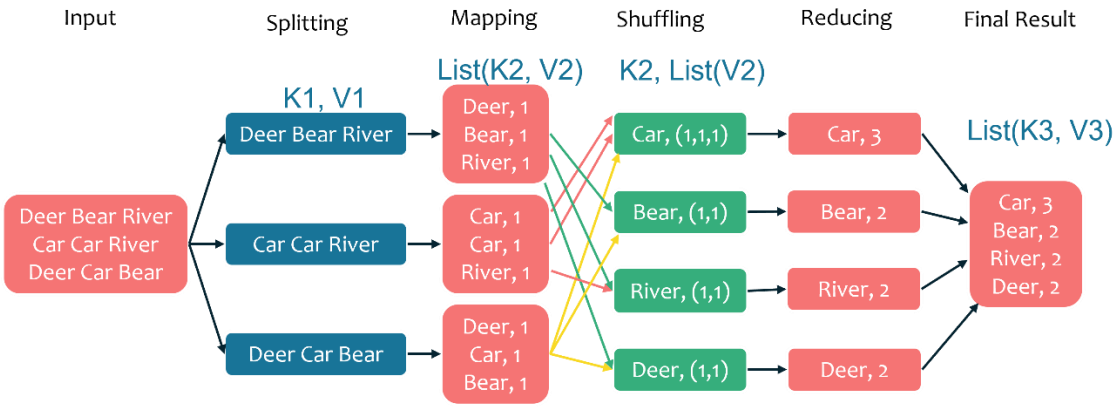
Dữ liệu được lưu dưới dạng tệp .txt với dung lượng 11,648 KB và 1942220 từ (mở trên Microsoft Word).

3. Cài đặt môi trường

Cài đặt thực nghiệm nhóm em chạy trên window 10 nên sẽ có cách cài đặt và thiết lập khác nếu thực nghiệm trên Linux. Ngoài ra, ta có thể cài đặt máy ảo Linux để cài đặc thực nghiệm chạy bằng Java. Hai phần trọng tâm dưới đây được thực hiện cho window 10.

- Cài đặt và thiết lập biến môi trường cho Java SE Development Kit 8 Update 333
- Tải Hadoop 3.3.0 (cho window 10) và thiết lập biến môi trường, cấu hình tập tin cho Hadoop 3.3.0.

4. Mô tả cách hoạt động của MapReduce



- Đầu tiên, chúng ta chia đầu vào thành ba phần như trong hình. Điều này sẽ phân phối công việc giữa tất cả các nút bản đồ.
- Sau đó, chúng tôi mã hóa các từ trong mỗi trình ánh xạ và cung cấp giá trị mã hóa cứng cho mỗi mã thông báo hoặc từ. Cơ sở lý luận đằng sau việc đưa ra giá trị mã cứng bằng 1 là mỗi từ, bản thân nó, sẽ xuất hiện một lần.
- Bây giờ, một danh sách các cặp khóa-giá trị sẽ được tạo trong đó khóa không là gì ngoài các từ và giá trị riêng lẻ là một. Vì vậy, đối với dòng đầu tiên (Dear Bear River), chúng ta có 3 cặp khóa-giá trị - Dear, 1; Bear, 1; River, 1. Quá trình ánh xạ vẫn giống nhau trên tất cả các nút.
- Sau giai đoạn ánh xạ, một quá trình phân vùng diễn ra trong đó việc sắp xếp và xáo trộn xảy ra để tất cả các bộ dữ liệu có cùng một khóa được gửi đến trình rút gọn tương ứng.
- Vì vậy, sau giai đoạn sắp xếp và xáo trộn, mỗi bộ rút gọn sẽ có một khóa duy nhất và danh sách các giá trị tương ứng với chính khóa đó. Ví dụ, Bear, [1,1]; Xe hơi, [1,1,1] .., v.v.
- Bây giờ, mỗi Bộ rút gọn đếm các giá trị có trong danh sách các giá trị đó. Như được hiển thị trong hình, bộ rút gọn nhận được một danh sách các giá trị là [1,1] cho khóa Bear. Sau đó, nó đếm số lượng đơn vị trong danh sách và đưa ra kết quả cuối cùng là - Bear, 2.

- Cuối cùng, tất cả các cặp khóa / giá trị đầu ra sau đó được thu thập và ghi vào tệp đầu ra.

5. Thực hiện demo

Video demo mà nhóm thực hiện: demo-wordcount.mp4

TÀI LIỆU THAM KHẢO

[1] S. K. says, "Apache Hadoop YARN | Introduction to YARN Architecture", *Edureka*, 19 Tháng Sáu 2018. https://www.edureka.co/blog/hadoop-yarn-tutorial/ (truy cập 25 Tháng Sáu 2022).

[2] "Apache Hadoop Architecture Explained (In-Depth Overview)", *Knowledge Base by phoenixNAP*, 25 Tháng Năm 2020. https://phoenixnap.com/kb/apache-hadoop-architecture-explained (truy cập 25 Tháng Sáu 2022).

[3] T. D. Thanh, "Cài đặt Hadoop trên Windows", *Advanced programming*, 1 Tháng Giêng 2021. https://duythanhcse.wordpress.com/2021/01/01/cai-dat-hadoop-tren-windows/ (truy cập 25 Tháng Sáu 2022).

[4] "Hadoop Ecosystem", *GeeksforGeeks*, 18 Tháng Chạp 2018. https://www.geeksforgeeks.org/hadoop-ecosystem/ (truy cập 25 Tháng Sáu 2022).

[5] trannguyenhan longpt, "HDFS", *De Manejar*, 4 Tháng Bảy 2021. https://demanejar.github.io/posts/hdfs-introduction/ (truy cập 25 Tháng Sáu 2022).

[6] D. Taylor, "HDFS Tutorial: Architecture, Read & Write Operation using Java API", 21 Tháng Giêng 2020. https://www.guru99.com/learn-hdfs-a-beginners-guide.html (truy cập 25 Tháng Sáu 2022).

[7] "HDFS Tutorial | Introduction to HDFS & its Features", *Edureka*, 21 Tháng Mười 2016. https://www.edureka.co/blog/hdfs-tutorial (truy cập 25 Tháng Sáu 2022).

[8] "MapReduce Tutorial | Mapreduce Example in Apache Hadoop", *Edureka*, 15 Tháng Mười-Một 2016. https://www.edureka.co/blog/mapreduce-tutorial/ (truy cập 25 Tháng Sáu 2022).

[9] R. A. says, "What is Big Data | Big Data Definition", *Edureka*, 8 Tháng Sáu 2018. https://www.edureka.co/blog/what-is-big-data/ (truy cập 25 Tháng Sáu 2022).

[10] "What is Hadoop? Hadoop Big Data Processing", *phoenixNAP Blog*, 14 Tháng Tám 2020. https://phoenixnap.com/blog/what-is-hadoop (truy cập 25 Tháng Sáu 2022).