# **THÀNH VIÊN NHÓM 2**

***Lâm Thị Phương Thảo*** (Nhóm trưởng): Tìm hiểu về **YARN**

***Võ Minh Hân***: Tìm hiểu về **APACHE** **HBASE**

***Trà Ngọc Thông***: Tìm hiểu về **HDFS**

Cả nhóm: tìm hiểu sơ lược về các thành phần còn lại cùng nhau tạo nên một hệ sinh thái Hadoop

* Các mục tìm hiểu rõ có đánh dấu \* ở phần Find Heading

# **FILE PPTX BÁO CÁO CỦA NHÓM**

<https://docs.google.com/presentation/d/1N5mq7Q2F5itT8_tFQogO_xOVSh4eRuKa/edit?usp=sharing&ouid=113808381688500628996&rtpof=true&sd=true>

# **Tài liệu tham khảo**

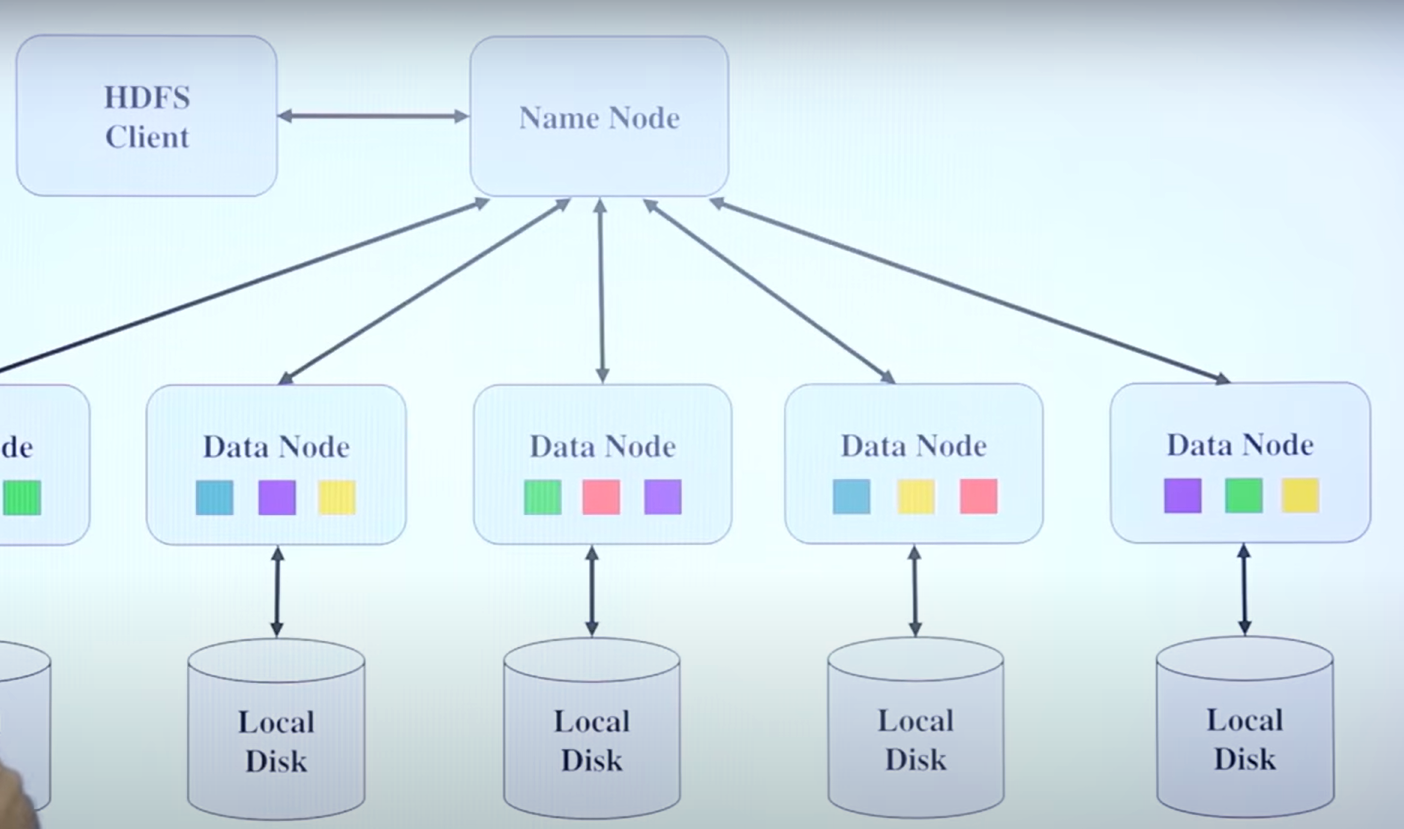
Sinha, S. (2023, 3 13). EDUREKA! Retrieved from https://www.edureka.co/blog/hadoop-ecosystem

# **HADOOP ECOSYSTEM**

Dưới đây là các thành phần của Hadoop: cùng nhau tạo thành một hệ sinh thái Hadoop: tôi sẽ trình bày từng thành phần trong blog này:

* [HDFS](https://www.edureka.co/blog/hadoop-ecosystem#hdfs): Hệ thống tệp phân tán Hadoop
* [YARN](https://www.edureka.co/blog/hadoop-ecosystem#yarn):  Một nhà đàm phán tài nguyên khác
* [MapReduce](https://www.edureka.co/blog/hadoop-ecosystem#mapreduce): Xử lý dữ liệu bằng lập trình
* [Spark](https://www.edureka.co/blog/hadoop-ecosystem#apache_spark):Xử lý dữ liệu trong bộ nhớ
* [PIG](https://www.edureka.co/blog/hadoop-ecosystem#apache_pig) : [HIVE](https://www.edureka.co/blog/hadoop-ecosystem#apache_hive) : Dịch vụ xử lý dữ liệu bằng truy vấn (giống SQL)
* [HBase](https://www.edureka.co/blog/hadoop-ecosystem#apache_hbase): Cơ sở dữ liệu NoSQL
* [Mahout](https://www.edureka.co/blog/hadoop-ecosystem#apache_mahout) : Spark MLlib : Học máy
* [Máy khoan Apache](https://www.edureka.co/blog/hadoop-ecosystem#apache_drill) : SQL trên Hadoop
* [Người quản lý vườn thú](https://www.edureka.co/blog/hadoop-ecosystem#apache_zookeeper): Quản lý cụm
* [Oozie](https://www.edureka.co/blog/hadoop-ecosystem#apache_oozie): Lập kế hoạch công việc
* [Flume](https://www.edureka.co/blog/hadoop-ecosystem#apache_flume) : [Sqoop](https://www.edureka.co/blog/hadoop-ecosystem#apache_sqoop) : Dịch vụ nhập dữ liệu
* [Solr & Lucene](https://www.edureka.co/blog/hadoop-ecosystem#apache_solr_and_lucene) : Tìm kiếm & lập chỉ mục
* [Ambari](https://www.edureka.co/blog/hadoop-ecosystem#apache_ambari): Cung cấp: giám sát và duy trì cụm

## **HDFS \***



### **Cách thức hoạt động của HDFS**

Khi dữ liệu ngày càng nhiều, hệ thống bị quá tải đòi hỏi cần có công cụ giúp chia nhỏ và lưu dữ chúng trên nhiều máy tính. Chính vì vậy, HDFS ra đời và đảm nhận nhiệm vụ trên. 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 tập 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.

* [Hệ thống tệp phân tán Hadoop](https://www.edureka.co/blog/hdfs-tutorial) là thành phần cốt lõi hay có thể nói là xương sống của Hệ sinh thái Hadoop.
* HDFS là một loại cho phép lưu trữ các loại tập dữ liệu lớn khác nhau (tức là dữ liệu có cấu trúc, không cấu trúc và bán cấu trúc).
* HDFS tạo ra một mức độ trừu tượng đối với các tài nguyên, từ đó chúng ta có thể xem toàn bộ HDFS dưới dạng một đơn vị duy nhất.
* Nó giúp chúng tôi lưu trữ dữ liệu của mình trên nhiều nút khác nhau và duy trì tệp nhật ký về dữ liệu được lưu trữ (siêu dữ liệu).
* HDFS có các thành phần: NameNode, DataNode và Secondary NameNode.

Hai thành phần cốt lõi là **NameNode và DataNode** .

### **NameNode**

* + - là nút chính và nó không lưu trữ dữ liệu thực tế **.**Nó chứa siêu dữ liệu, giống như một tệp nhật ký hoặc bạn có thể nói như một bảng nội dung. Do đó, nó đòi hỏi ít bộ nhớ hơn và tài nguyên tính toán cao.
    - Lưu trữ thông tin tên metaData (fsimage, edit logs)
    - Quản ly user truy cập vào hệ thống
    - Giao tiếp với client cho thao tác đọc ghi dữ liệu
    - Cung cấp dịch vụ đăng ký DataNode mới trong cụm, nhận heartbeat của DataNode
    - Xác định location của file, replicated file

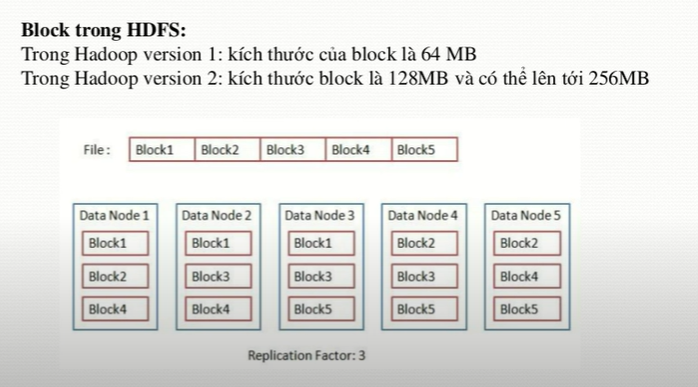
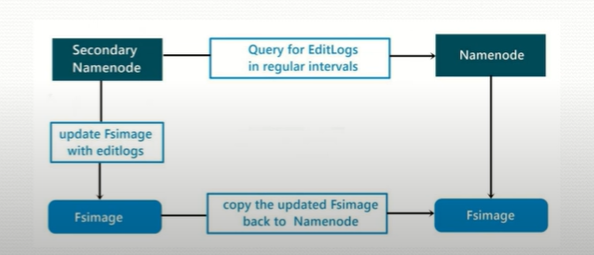
### **DataNode**

* + - tất cả dữ liệu của bạn được lưu trữ trên DataNodes **và** do đó nó đòi hỏi nhiều tài nguyên lưu trữ hơn. Các DataNode này là phần cứng thông dụng (như máy tính xách tay và máy tính để bàn của bạn) trong môi trường phân tán. Đó là lý do tại sao các giải pháp Hadoop rất hiệu quả về mặt chi phí.
    - Cung cấp các block để lưu trữ file
    - Giao tiếp với client cho thao tác đọc ghi
    - Khởi tạo và xóa block data trong cụm
    - Giữ liên lạc với NamenNode thông qua việc gửi các heartbeat theo định kỳ

🡪 Bạn luôn liên lạc với NameNode trong khi ghi dữ liệu. Sau đó, nó sẽ gửi yêu cầu nội bộ đến máy khách để lưu trữ và sao chép dữ liệu trên các DataNode khác nhau.

### **Secondary NameNode:**

* + - Lưu các thông tin trên fsimage của NameNode theo định kỳ
    - Phục vụ cho việc khôi phục lại thông tin metadata trên namenode khi gặp sự cố



## YARN \*

YARN cho phép các phương pháp xử lý dữ liệu khác nhau như xử lý đồ thị, xử lý tương tác, xử lý luồng cũng như xử lý hàng loạt để chạy và xử lý dữ liệu được lưu trữ trong HDFS. Do đó YARN mở ra Hadoop cho các loại ứng dụng phân tán khác ngoài MapReduce.

YARN cho phép người dùng thực hiện các thao tác theo yêu cầu bằng cách sử dụng nhiều công cụ khác nhau như [Spark](https://www.edureka.co/blog/spark-tutorial/) để xử lý thời gian thực, [**Hive**](https://www.edureka.co/blog/hive-tutorial/) cho SQL, [**HBase**](https://www.edureka.co/blog/hbase-tutorial) cho NoSQL và các công cụ khác.

Hãy coi YARN là bộ não của Hệ sinh thái Hadoop của bạn. Nó thực hiện tất cả các hoạt động xử lý của bạn bằng cách phân bổ tài nguyên và lập lịch các tác vụ.

* Nó có hai thành phần chính là ResourceManager và NodeManager .

### **ResourceManager (**Trình quản lý tài nguyên**)**

* + - Lại là nút chính trong bộ phận xử lý. Nó nhận được các yêu cầu xử lý và sau đó chuyển các phần của yêu cầu đến các Trình quản lý nút tương ứng, nơi quá trình xử lý thực tế diễn ra.
    - Đây là cơ quan có thẩm quyền cuối cùng trong việc phân bổ nguồn lực .
    - Khi nhận được yêu cầu xử lý, nó sẽ chuyển các phần yêu cầu đến người quản lý nút tương ứng, nơi quá trình xử lý thực tế diễn ra.
    - Nó là người phân xử các tài nguyên của cụm và quyết định việc phân bổ các tài nguyên sẵn có cho các ứng dụng cạnh tranh.
    - Tối ưu hóa việc sử dụng cụm như giữ tất cả các tài nguyên luôn được sử dụng trước các hạn chế khác nhau như đảm bảo năng lực, tính công bằng và SLA.
    - Nó có hai thành phần chính:
      * a) Bộ lập lịch: Dựa trên yêu cầu tài nguyên ứng dụng của bạn, Bộ lập lịch thực hiện các thuật toán lập lịch và phân bổ tài nguyên.

Bộ lập lịch chịu trách nhiệm phân bổ tài nguyên cho các ứng dụng đang chạy khác nhau tùy thuộc vào các hạn chế về năng lực, hàng đợi, v.v.

Nó được gọi là bộ lập lịch thuần túy trong ResourceManager, có nghĩa là nó không thực hiện bất kỳ giám sát hoặc theo dõi trạng thái nào cho các ứng dụng.

Nếu có lỗi ứng dụng hoặc lỗi phần cứng, Bộ lập lịch không đảm bảo sẽ khởi động lại các tác vụ bị lỗi.

Thực hiện lập lịch dựa trên yêu cầu tài nguyên của ứng dụng.

Nó có một plug-in chính sách có thể cắm được, chịu trách nhiệm phân vùng tài nguyên cụm giữa các ứng dụng khác nhau. Có hai phần bổ trợ như vậy:  **Bộ lập lịch công suất**  và  **Bộ lập lịch công bằng** , hiện được sử dụng làm Bộ lập lịch trong ResourceManager.

* b )  Trình quản lý ứng dụng: Trong khi Trình quản lý ứng dụng chấp nhận gửi công việc, hãy đàm phán với các vùng chứa (tức là môi trường nút Dữ liệu nơi quy trình thực thi) để thực thi ApplicationMaster cụ thể của ứng dụng và theo dõi tiến trình. ApplicationMasters là các deamons cư trú trên DataNode và giao tiếp với các thùng chứa để thực thi các tác vụ trên mỗi DataNode. ResourceManager có hai thành phần, tức là **Schedulers**và **ApplicationsManager**.
  + - * + Nó có trách nhiệm tiếp nhận hồ sơ công việc.
        + Thương lượng vùng chứa đầu tiên từ Trình quản lý tài nguyên để thực thi Ứng dụng chủ ứng dụng cụ thể của ứng dụng.
        + Quản lý việc chạy Ứng dụng chính trong một cụm và cung cấp dịch vụ khởi động lại vùng chứa Ứng dụng chính khi bị lỗi.

### **NodeManagers(Trình quản lý nút)**

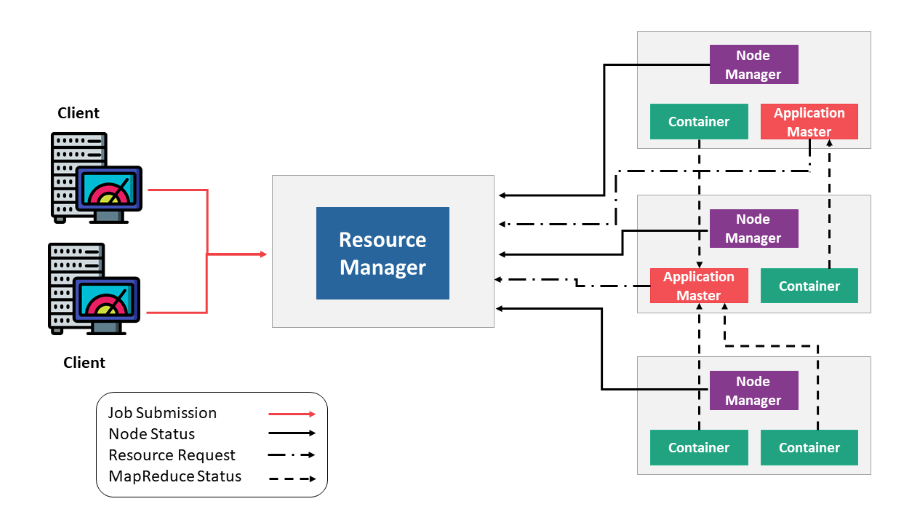
* + - được cài đặt trên mọi DataNode. Nó chịu trách nhiệm thực thi nhiệm vụ trên mỗi DataNode.
    - Nó quản lý các nút riêng lẻ trong cụm Hadoop và  quản lý công việc cũng như quy trình làm việc của người dùng trên nút nhất định.
    - Nó đăng ký với Trình quản lý tài nguyên và gửi nhịp tim với tình trạng sức khỏe của nút.
    - Mục tiêu chính của nó là quản lý các thùng chứa ứng dụng được người quản lý tài nguyên gán cho nó.
    - Nó luôn cập nhật với Trình quản lý tài nguyên.
    - Ứng dụng Master yêu cầu vùng chứa được chỉ định từ Trình quản lý nút bằng cách gửi cho nó Bối cảnh khởi chạy vùng chứa (CLC) bao gồm mọi thứ mà ứng dụng cần để chạy. Trình quản lý nút tạo quy trình vùng chứa được yêu cầu và khởi động nó.
    - Giám sát việc sử dụng tài nguyên (bộ nhớ, CPU) của từng vùng chứa.
    - Thực hiện quản lý nhật ký.
    - Nó cũng giết container theo chỉ dẫn của Trình quản lý tài nguyên.

### **Application Master(**Bậc thầy ứng dụng)

* + - Quản lý vòng đời công việc của người dùng và nhu cầu tài nguyên của từng ứng dụng. Nó hoạt động cùng với Trình quản lý nút và giám sát việc thực hiện các tác vụ.
    - Một ứng dụng là một công việc duy nhất được gửi tới framework. Mỗi ứng dụng như vậy có một Master ứng dụng duy nhất được liên kết với nó, đây là một thực thể khung cụ thể.
    - Đây là quá trình điều phối việc thực thi ứng dụng trong cụm và cũng quản lý các lỗi.
    - Nhiệm vụ của nó là đàm phán các tài nguyên từ Trình quản lý tài nguyên và làm việc với Trình quản lý nút để thực thi và giám sát các tác vụ thành phần.
    - Nó chịu trách nhiệm đàm phán các vùng chứa tài nguyên thích hợp từ ResourceManager, theo dõi trạng thái của chúng và giám sát tiến trình.
    - Sau khi bắt đầu, nó sẽ gửi nhịp tim định kỳ tới Trình quản lý tài nguyên để xác nhận tình trạng của nó và cập nhật hồ sơ về nhu cầu tài nguyên.

### Container

* + - Gói tài nguyên bao gồm RAM, CPU, Mạng, HDD, v.v. trên một nút duy nhất.
    - Nó là tập hợp các tài nguyên vật lý như RAM, lõi CPU và đĩa trên một nút.
    - Các thùng chứa YARN được quản lý bởi bối cảnh khởi chạy vùng chứa là vòng đời của vùng chứa (CLC). Bản ghi này chứa bản đồ các biến môi trường, các phần phụ thuộc được lưu trữ trong bộ lưu trữ có thể truy cập từ xa, mã thông báo bảo mật, tải trọng cho các dịch vụ Trình quản lý nút và lệnh cần thiết để tạo quy trình.
    - Nó cấp quyền cho ứng dụng sử dụng một lượng tài nguyên cụ thể (bộ nhớ, CPU, v.v.) trên một máy chủ cụ thể.Tìm hiểu thêm về Dữ liệu lớn và các ứng dụng của nó từ [chứng chỉ Kỹ thuật dữ liệu](https://www.edureka.co/microsoft-azure-data-engineering-certification-course) .



### Ví dụ : **Gửi đơn đăng ký bằng YARN,** Tham khảo hình ảnh và xem các bước liên quan đến việc gửi đơn đăng ký Hadoop YARN:

1. Gửi công việc

2. Nhận ID ứng dụng

3. Bối cảnh nộp đơn

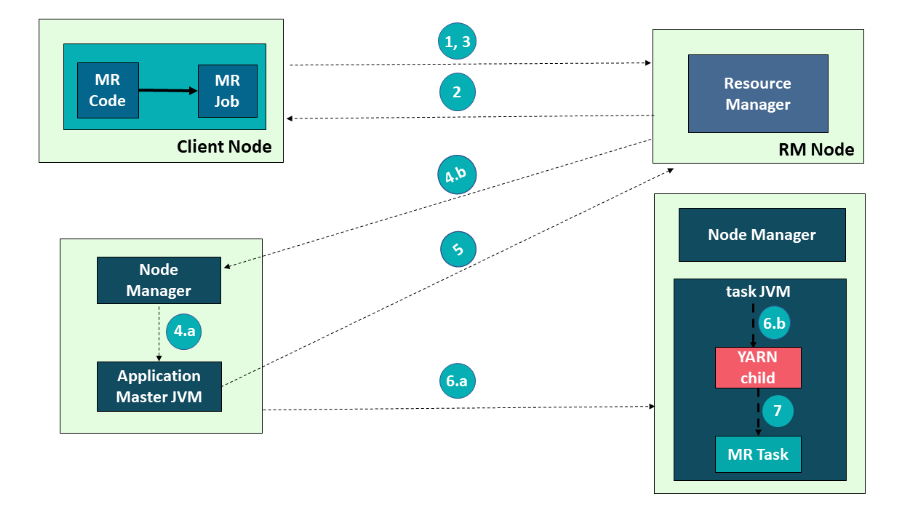
4.a. Bắt đầu  khởi chạy vùng chứa

4.b. Khởi chạy ứng dụng Master

5. Phân bổ nguồn lực

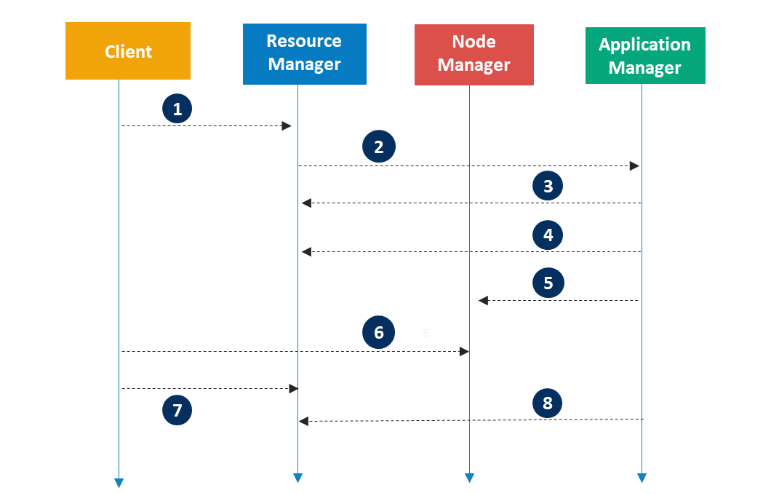
6.a. Thùng chứa

6.b. Khởi động

7. Thực hiện

**Quy trình làm việc của ứng dụng trong Hadoop YARN**

1. Khách hàng nộp đơn
2. Trình quản lý tài nguyên phân bổ một vùng chứa để khởi động Trình quản lý ứng dụng
3. Trình quản lý ứng dụng đăng ký với Trình quản lý tài nguyên
4. Trình quản lý ứng dụng yêu cầu các vùng chứa từ Trình quản lý tài nguyên
5. Trình quản lý ứng dụng thông báo Trình quản lý nút khởi chạy vùng chứa
6. Mã ứng dụng được thực thi trong vùng chứa
7. Khách hàng liên hệ với Trình quản lý tài nguyên/Trình quản lý ứng dụng để theo dõi trạng thái của ứng dụng
8. Trình quản lý ứng dụng hủy đăng ký với Trình quản lý tài nguyên



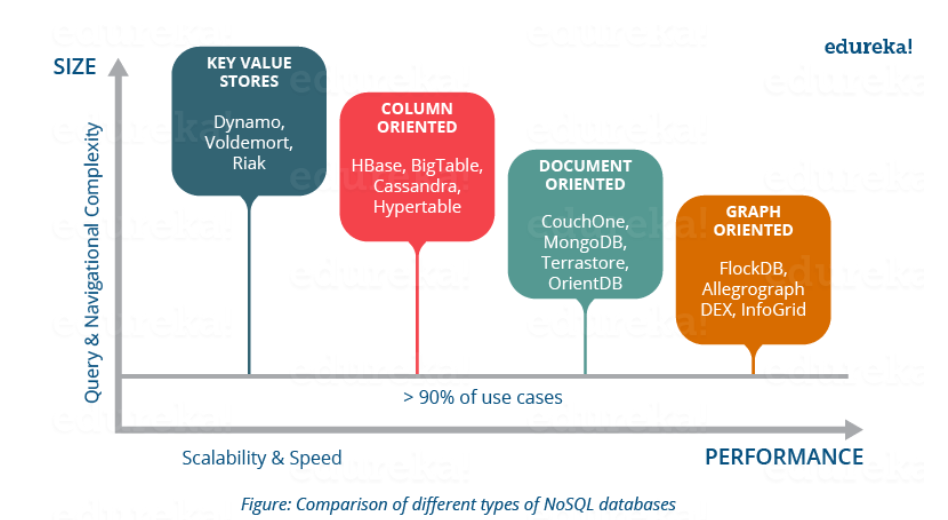
## **APACHE HBASE \***

* HBase là một cơ sở dữ liệu phân tán mã nguồn mở, không quan hệ. Nói cách khác, nó là cơ sở dữ liệu NoSQL.
* Nó hỗ trợ tất cả các loại dữ liệu và đó là lý do tại sao, nó có khả năng xử lý mọi thứ và mọi thứ trong hệ sinh thái Hadoop.
* Nó được mô phỏng theo BigTable của Google, một hệ thống lưu trữ phân tán được thiết kế để xử lý các tập dữ liệu lớn.
* HBase được thiết kế để chạy trên HDFS và cung cấp các khả năng giống như BigTable.
* Nó cung cấp cho chúng tôi một cách có khả năng chấp nhận lỗi để lưu trữ dữ liệu thưa thớt, điều này phổ biến trong hầu hết các trường hợp sử dụng Dữ liệu lớn.
* HBase được viết bằng Java, trong khi các ứng dụng HBase có thể được viết bằng API REST, Avro và Thrift.

Để hiểu rõ hơn, chúng ta hãy lấy một ví dụ. Bạn có hàng tỷ email của khách hàng và bạn cần tìm hiểu số lượng khách hàng đã sử dụng từ phàn nàn trong email của họ. Yêu cầu cần được xử lý nhanh chóng (tức là theo thời gian thực). Vì vậy, ở đây chúng tôi đang xử lý một tập dữ liệu lớn trong khi truy xuất một lượng nhỏ dữ liệu. Để giải quyết những loại vấn đề này, HBase đã được thiết kế.

### **Hướng dẫn về Apache HBase: Cơ sở dữ liệu NoSQL**

NoSQL có nghĩa là Không chỉ SQL . Cơ sở dữ liệu NoSQL được mô hình hóa theo cách nó có thể biểu diễn dữ liệu ngoài các định dạng dạng bảng, cơ sở dữ liệu quan hệ độc đáo. Nó sử dụng các định dạng khác nhau để biểu diễn dữ liệu trong cơ sở dữ liệu và do đó, có nhiều loại cơ sở dữ liệu NoSQL khác nhau dựa trên định dạng biểu diễn của chúng. Hầu hết các cơ sở dữ liệu NoSQL đều tận dụng tính khả dụng và tốc độ hơn là tính nhất quán. Bây giờ, chúng ta hãy tiếp tục và tìm hiểu về các loại cơ sở dữ liệu NoSQL khác nhau cũng như định dạng biểu diễn của chúng.



#### **Key-Value stores (Cửa hàng khóa-giá trị):**

Nó là một cơ sở dữ liệu không có lược đồ, chứa các khóa và giá trị. Mỗi khóa, trỏ đến một giá trị là một mảng byte, có thể là một chuỗi, BLOB, XML, v.v. ví dụ: Lamborghini là một khóa và có thể trỏ đến một giá trị Gallardo, Aventador, Murciélago, Reventón, Diablo, Huracán, Veneno, Centenario, v.v.

Cơ sở dữ liệu lưu trữ khóa-giá trị: Aerospike, Couchbase, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, MemcacheDB, MUMPS, Cơ sở dữ liệu Oracle NoSQL, OrientDB, Redis, Riak, Berkeley DB.

**Use-case**(Trường hợp sử dụng)

Các cửa hàng khóa-giá trị xử lý tốt kích thước và xử lý tốt luồng hoạt động đọc/ghi liên tục với độ trễ thấp. Điều này làm cho chúng trở nên hoàn hảo cho các cửa hàng hồ sơ và sở thích của Người dùng,  Đề xuất sản phẩm; các mặt hàng mới nhất được xem trên trang web của nhà bán lẻ để thúc đẩy đề xuất sản phẩm cho khách hàng trong tương lai,  Dịch vụ quảng cáo; thói quen mua sắm của khách hàng dẫn đến quảng cáo tùy chỉnh, phiếu giảm giá, v.v. cho từng khách hàng theo thời gian thực.

#### **Document Oriented** (**Định hướng tài liệu**)

Nó tuân theo cùng một cặp giá trị khóa nhưng có cấu trúc bán giống như XML, JSON, BSON. Những cấu trúc này được coi là tài liệu.

Cơ sở dữ liệu dựa trên tài liệu: Apache CouchDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, IBM Domino, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB.

**Use-case** (Trường hợp sử dụng)

Vì tài liệu hỗ trợ lược đồ linh hoạt, khả năng đọc và phân vùng đọc nhanh giúp nó phù hợp để tạo cơ sở dữ liệu người dùng trong các dịch vụ khác nhau như Twitter, trang web thương mại điện tử, v.v.

#### **Column Oriented** (**Định hướng theo cột):**

Trong cơ sở dữ liệu này, dữ liệu được lưu trữ trong ô được nhóm theo cột thay vì hàng. Các cột được nhóm một cách hợp lý thành các họ cột có thể được tạo trong quá trình định nghĩa lược đồ hoặc trong thời gian chạy.

Các loại cơ sở dữ liệu này lưu trữ tất cả các ô tương ứng với một cột dưới dạng mục nhập đĩa liên tục, do đó giúp việc truy cập và tìm kiếm nhanh hơn nhiều.

Cơ sở dữ liệu dựa trên cột: HBase, Accumulo, Cassandra, Druid, Vertica.

**Use-case** (Trường hợp sử dụng)

Nó hỗ trợ bộ nhớ khổng lồ và cho phép truy cập đọc ghi nhanh hơn. Điều này làm cho cơ sở dữ liệu định hướng theo cột phù hợp để lưu trữ hành vi của khách hàng trong trang web thương mại điện tử, hệ thống tài chính như Google Finance và dữ liệu thị trường chứng khoán, Google maps, v.v.

#### **Graph Oriented (Định hướng đồ thị):**

Nó là một biểu diễn đồ họa linh hoạt hoàn hảo, được sử dụng không giống như SQL. Những loại cơ sở dữ liệu này dễ dàng giải quyết các vấn đề về khả năng mở rộng địa chỉ vì nó chứa các cạnh và nút có thể được mở rộng theo yêu cầu.

Cơ sở dữ liệu dựa trên đồ thị: AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Stardog.

**Use-case** (Trường hợp sử dụng)

Về cơ bản, điều này được sử dụng trong Phát hiện gian lận, Công cụ đề xuất thời gian thực (trong hầu hết các trường hợp là thương mại điện tử), Quản lý dữ liệu chính (MDM), Hoạt động mạng và CNTT, Quản lý nhận dạng và truy cập (IAM), v.v.

HBase và Cassandra là hai cơ sở dữ liệu định hướng theo cột nổi tiếng. Vì vậy, bây giờ nói ở cấp độ cao hơn, chúng ta hãy so sánh và hiểu sự khác biệt về kiến ​​trúc và hoạt động giữa HBase và Cassandra.

### **Hướng dẫn về HBase: HBase VS Cassandra**

* HBase được mô phỏng theo BigTable (Google) trong khi Cassandra dựa trên DynamoDB (Amazon) do Facebook phát triển ban đầu.
* HBase tận dụng cơ sở hạ tầng Hadoop (HDFS, ZooKeeper) trong khi Cassandra phát triển riêng nhưng bạn có thể kết hợp Hadoop và Cassandra theo nhu cầu của mình.
* HBase có một số thành phần giao tiếp với nhau như HBase HMaster, ZooKeeper, NameNode, Region Severs. Trong khi Cassandra là loại nút đơn, trong đó tất cả các nút đều bằng nhau và thực hiện tất cả các chức năng. Bất kỳ nút nào cũng có thể là nút điều phối; điều này loại bỏ Điểm thất bại duy nhất.
* HBase được tối ưu hóa để đọc và hỗ trợ ghi đơn, dẫn đến tính nhất quán nghiêm ngặt. HBase hỗ trợ quét theo Phạm vi, giúp quá trình quét nhanh hơn. Trong khi đó Cassandra hỗ trợ đọc một hàng để duy trì tính nhất quán cuối cùng.
* Cassandra không hỗ trợ quét hàng theo phạm vi, điều này làm chậm quá trình quét so với HBase.
* HBase hỗ trợ phân vùng theo thứ tự, trong đó các hàng của Nhóm Cột được lưu trữ theo thứ tự RowKey, trong khi ở phân vùng theo thứ tự Casandra là một thách thức. Do phân vùng RowKey nên quá trình quét trong HBase nhanh hơn so với Cassandra.
* HBase không hỗ trợ cân bằng tải đọc, một Máy chủ Khu vực phục vụ yêu cầu đọc và các bản sao chỉ được sử dụng trong trường hợp bị lỗi. Trong khi Cassandra hỗ trợ cân bằng tải đọc và có thể đọc cùng một dữ liệu từ nhiều nút khác nhau. Điều này có thể làm tổn hại đến tính nhất quán.
* Trong định lý CAP (Tính nhất quán, Tính sẵn có & Phân vùng - Dung sai) HBase duy trì Tính nhất quán và Tính sẵn sàng trong khi Cassandra tập trung vào Tính sẵn có và Phân vùng - Dung sai.

### **Hướng dẫn về Apache HBase: Các tính năng của HBase**

* **Atomic read and write (Đọc và ghi nguyên tử):** Ở cấp độ hàng, HBase cung cấp khả năng đọc và ghi nguyên tử. Nó có thể được giải thích là, trong một quá trình đọc hoặc ghi, tất cả các quá trình khác đều bị ngăn không cho thực hiện bất kỳ thao tác đọc hoặc ghi nào.
* **Consistent reads and writes (Đọc và ghi nhất quán):** HBase cung cấp khả năng đọc và ghi nhất quán nhờ tính năng trên.
* **Linear and modular scalability (Khả năng mở rộng tuyến tính và mô-đun):** Vì các tập dữ liệu được phân phối trên HDFS, do đó, nó có khả năng mở rộng tuyến tính trên nhiều nút khác nhau, cũng như có thể mở rộng theo mô-đun, vì nó được chia thành nhiều nút khác nhau.
* **Automatic and configurable sharding of tables (Phân chia bảng tự động và có thể định cấu hình):** Các bảng HBase được phân phối trên các cụm và các cụm này được phân bổ trên các khu vực. Các vùng và cụm này được phân chia và phân phối lại khi dữ liệu tăng lên.
* **Easy to use Java API for client access (API Java dễ sử dụng để truy cập máy khách):** Nó cung cấp API Java dễ sử dụng để truy cập theo chương trình.
* **Thrift gateway and a REST-ful Web services (Cổng tiết kiệm và dịch vụ Web REST-Ful):** Nó cũng hỗ trợ API Thrift và REST cho các giao diện người dùng không phải Java.
* **Block Cache and Bloom Filters (Chặn bộ lọc bộ đệm và bộ lọc Bloom):** HBase hỗ trợ Bộ lọc khối bộ đệm và bộ lọc Bloom để tối ưu hóa truy vấn khối lượng lớn.
* **Automatic failure support (Hỗ trợ lỗi tự động):** HBase với HDFS cung cấp WAL (Nhật ký ghi trước) trên các cụm cung cấp hỗ trợ lỗi tự động.
* **Sorted rowkeys (Các khóa hàng được sắp xếp):** Khi việc tìm kiếm được thực hiện trên nhiều hàng, HBase lưu trữ các khóa hàng theo thứ tự từ điển. Bằng cách sử dụng các mã hàng và dấu thời gian được sắp xếp này, chúng ta có thể xây dựng một yêu cầu được tối ưu hóa.

### **Hướng dẫn về HBase: Chúng ta có thể sử dụng HBase ở đâu?**

* Chúng ta nên sử dụng HBase khi chúng ta có các tập dữ liệu lớn (hàng triệu hoặc hàng tỷ hoặc hàng và cột) và chúng tôi yêu cầu quyền truy cập đọc và ghi dữ liệu nhanh, ngẫu nhiên và theo thời gian thực.
* Các bộ dữ liệu được phân phối trên nhiều cụm khác nhau và chúng tôi cần khả năng mở rộng cao để xử lý dữ liệu.
* Dữ liệu được thu thập từ nhiều nguồn dữ liệu khác nhau và đó là dữ liệu bán cấu trúc hoặc không cấu trúc hoặc kết hợp tất cả. Nó có thể được xử lý dễ dàng với HBase.
* Bạn muốn lưu trữ dữ liệu theo định hướng cột.
* Bạn có rất nhiều phiên bản của tập dữ liệu và bạn cần lưu trữ tất cả chúng.

### **Hướng dẫn về HBase: HBase VS HDFS**

Vì cả HDFS và HBase đều lưu trữ bất kỳ loại dữ liệu nào (tức là có cấu trúc, bán cấu trúc và không cấu trúc) trong môi trường phân tán, vì vậy hãy xem xét sự khác biệt giữa hệ thống tệp HDFS và HBase, cơ sở dữ liệu NoSQL.

* HBase cung cấp quyền truy cập có độ trễ thấp vào một lượng nhỏ dữ liệu trong các tập dữ liệu lớn trong khi HDFS cung cấp các hoạt động có độ trễ cao.
* HBase hỗ trợ đọc và ghi ngẫu nhiên trong khi HDFS hỗ trợ WORM (Viết một lần Đọc nhiều hoặc nhiều lần).
* HDFS về cơ bản hoặc chủ yếu được truy cập thông qua các công việc MapReduce trong khi HBase được truy cập thông qua các lệnh shell, API Java, REST, Avro hoặc Thrift API.

HDFS lưu trữ các tập dữ liệu lớn trong môi trường phân tán và thúc đẩy xử lý hàng loạt trên dữ liệu đó. Ví dụ: nó sẽ giúp một trang web thương mại điện tử lưu trữ hàng triệu dữ liệu của khách hàng trong môi trường phân tán phát triển trong một khoảng thời gian dài (có thể là 4-5 năm hoặc hơn). Sau đó, nó thúc đẩy việc xử lý hàng loạt dữ liệu đó và phân tích hành vi, mô hình, yêu cầu của khách hàng. Sau đó công ty có thể tìm ra loại sản phẩm nào, khách hàng mua hàng vào tháng nào. Nó giúp lưu trữ dữ liệu đã lưu trữ và thực hiện xử lý hàng loạt dữ liệu đó.

Trong khi HBase lưu trữ dữ liệu theo cách định hướng cột trong đó mỗi cột được lưu trữ cùng nhau để việc đọc trở nên nhanh hơn nhờ xử lý thời gian thực. Ví dụ: trong môi trường thương mại điện tử tương tự, nó lưu trữ hàng triệu dữ liệu sản phẩm. Vì vậy, nếu bạn tìm kiếm một sản phẩm trong số hàng triệu sản phẩm, nó sẽ tối ưu hóa quá trình yêu cầu và tìm kiếm, đưa ra kết quả ngay lập tức (hoặc bạn có thể nói là theo thời gian thực). Tôi sẽ trình bày phần giải thích chi tiết [**về kiến ​​trúc HBase trong blog tiếp theo của mình.**](https://www.edureka.co/blog/hbase-architecture/)

Như chúng ta biết HBase được phân phối trên HDFS, do đó, sự kết hợp của cả hai mang lại cho chúng ta cơ hội tuyệt vời để sử dụng các lợi ích của cả hai, trong một giải pháp phù hợp, như chúng ta sẽ thấy trong nghiên cứu điển hình về Facebook Messenger bên dưới.

### **Hướng dẫn HBase: Nghiên cứu điển hình về Facebook Messenger**

**Nền tảng nhắn tin Facebook** đã chuyển từ Apache Cassandra sang HBase vào tháng 11 năm 2010.

Facebook Messenger kết hợp Tin nhắn, email, trò chuyện và SMS thành một cuộc trò chuyện trong thời gian thực. Facebook đang cố gắng xây dựng một cơ sở hạ tầng mạnh mẽ và có thể mở rộng để xử lý tập hợp các dịch vụ này.

Vào thời điểm đó, cơ sở hạ tầng tin nhắn đã xử lý hơn 350 triệu người dùng gửi hơn 15 tỷ tin nhắn cá nhân mỗi tháng. Dịch vụ trò chuyện hỗ trợ hơn 300 triệu người dùng gửi hơn 120 tỷ tin nhắn mỗi tháng.

Bằng cách theo dõi việc sử dụng, họ phát hiện ra rằng có hai mẫu dữ liệu chung đã xuất hiện:

* Một tập hợp dữ liệu tạm thời ngắn có xu hướng không ổn định
* Một tập hợp dữ liệu ngày càng phát triển mà hiếm khi được truy cập

Facebook muốn tìm giải pháp lưu trữ cho hai kiểu sử dụng này và họ bắt đầu nghiên cứu để tìm giải pháp thay thế cho cơ sở hạ tầng Messages hiện có.

Trước đó vào năm 2008, họ đã sử dụng cơ sở dữ liệu nguồn mở, tức là Cassandra, một kho lưu trữ khóa-giá trị nhất quán cuối cùng đã được sản xuất để phục vụ lưu lượng truy cập cho Tìm kiếm trong Hộp thư đến. Nhóm của họ có kiến ​​thức sâu rộng về cách sử dụng và quản lý cơ sở dữ liệu MySQL, vì vậy việc chuyển đổi một trong hai công nghệ là mối quan tâm thực sự đối với họ.

Họ đã dành vài tuần để thử nghiệm các khung công tác khác nhau để đánh giá các cụm MySQL, Apache Cassandra, Apache HBase và các hệ thống khác. Cuối cùng họ đã chọn HBase.

Vì MySQL không thể xử lý các tập dữ liệu lớn một cách hiệu quả, khi các chỉ mục và tập dữ liệu ngày càng lớn nên hiệu suất bị ảnh hưởng. Họ nhận thấy Cassandra không thể xử lý được mô hình khó khăn để điều hòa cơ sở hạ tầng Tin nhắn mới của họ.

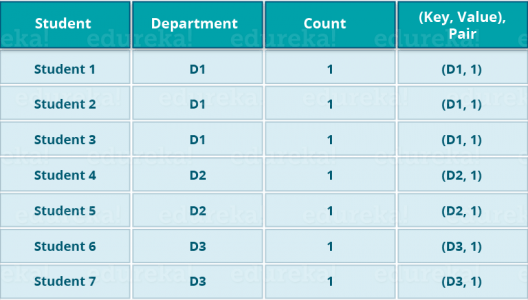
### **Các vấn đề chính là:**

* Lưu trữ lượng lớn dữ liệu liên tục tăng từ các dịch vụ khác nhau của Facebook.
* Yêu cầu Cơ sở dữ liệu có thể tận dụng khả năng xử lý cao trên đó.
* Cần có hiệu suất cao để phục vụ hàng triệu yêu cầu.
* Duy trì tính nhất quán trong lưu trữ và hiệu suất.
* Đối với tất cả những vấn đề này, Facebook đã đưa ra một giải pháp tức là HBase. Facebook đã sử dụng HBase để phục vụ Facebook Messenger, trò chuyện, email, v.v. do các tính năng khác nhau của nó.
* HBase có khả năng mở rộng và hiệu suất rất tốt cho khối lượng công việc này với mô hình nhất quán đơn giản hơn Cassandra. Mặc dù họ nhận thấy HBase là phù hợp nhất về các yêu cầu của họ như tự động cân bằng tải và chuyển đổi dự phòng, hỗ trợ nén, nhiều phân đoạn trên mỗi máy chủ, v.v.
* HDFS, hệ thống tệp cơ bản được HBase sử dụng cũng cung cấp cho họ một số tính năng cần thiết như tổng kiểm tra từ đầu đến cuối, sao chép và cân bằng lại tải tự động.
* Khi áp dụng HBase, họ cũng tập trung vào việc đưa kết quả trở lại chính HBase và bắt đầu hợp tác chặt chẽ với cộng đồng Apache.
* Vì tin nhắn chấp nhận dữ liệu từ nhiều nguồn khác nhau như SMS, cuộc trò chuyện và email nên họ đã viết một máy chủ ứng dụng để xử lý mọi quyết định về tin nhắn của người dùng. Nó giao tiếp với số lượng lớn các dịch vụ khác. Các tệp đính kèm được lưu trữ trong Haystack (hoạt động trên HBase). Họ cũng đã viết một dịch vụ khám phá người dùng trên Apache ZooKeeper, dịch vụ này giao tiếp với các dịch vụ cơ sở hạ tầng khác về mối quan hệ bạn bè, xác minh tài khoản email, quyết định gửi và quyết định về quyền riêng tư.

## **MAPREDUCE**

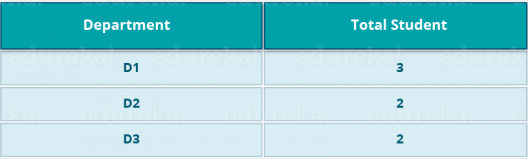
Đây là thành phần cốt lõi của quá trình xử lý trong Hệ sinh thái Hadoop vì nó cung cấp logic xử lý. Nói cách khác, MapReduce là một khung phần mềm giúp viết các ứng dụng xử lý các tập dữ liệu lớn bằng thuật toán phân tán và song song trong môi trường Hadoop.

* Trong chương trình MapReduce, **Map() and Reduce()**  là hai hàm.
  1. Hàm **Map function** thực hiện các hành động như lọc, nhóm và sắp xếp.
  2. Trong khi **Reduce function** tổng hợp và tóm tắt kết quả do hàm bản đồ tạo ra.
  3. Kết quả do hàm Map tạo ra là cặp giá trị khóa (K, V) đóng vai trò là đầu vào cho hàm Giảm.



Chúng ta hãy lấy ví dụ trên để hiểu rõ hơn về chương trình MapReduce.

Chúng tôi có một trường hợp mẫu về sinh viên và các khoa tương ứng của họ. Chúng tôi muốn tính số lượng sinh viên trong mỗi khoa. Ban đầu, chương trình Map sẽ thực hiện và tính toán số sinh viên xuất hiện ở mỗi khoa, tạo ra cặp giá trị khóa như đã đề cập ở trên. Cặp giá trị khóa này là đầu vào của hàm Giảm. Khi đó hàm Giảm sẽ tổng hợp từng khoa rồi tính tổng số sinh viên của từng khoa và cho ra kết quả cho trước.



## **APACHE PIG**

* PIG có hai phần: Pig Latin , ngôn ngữ và **the pig runtime (thời gian chạy của pig)**, dành cho môi trường thực thi. Bạn có thể hiểu rõ hơn về nó là Java và JVM.
* Nó hỗ trợ ngôn ngữ Latin lợn , có cấu trúc lệnh giống như SQL.

Vì không phải ai cũng có nền tảng lập trình. Vì vậy, Apache PIG giải quyết chúng. Bạn có thể tò mò muốn biết làm thế nào?

Chà, tôi sẽ kể cho bạn một sự thật thú vị:

10 dòng lợn Latin = khoảng. 200 dòng mã Java Map-Reduce

Nhưng đừng ngạc nhiên khi tôi nói rằng ở phần cuối của công việc Pig, một công việc thu nhỏ bản đồ sẽ được thực thi.

* Trình biên dịch chuyển đổi nội bộ tiếng Latin lợn thành MapReduce. Nó tạo ra một tập hợp tuần tự các công việc MapReduce và đó là một sự trừu tượng hóa (hoạt động giống như hộp đen).
* PIG ban đầu được phát triển bởi Yahoo.
* Nó cung cấp cho bạn nền tảng để xây dựng luồng dữ liệu cho ETL (Trích xuất, Chuyển đổi và Tải), xử lý và phân tích các tập dữ liệu khổng lồ.

Lợn hoạt động như thế nào?

Trong PIG, lệnh tải đầu tiên sẽ tải dữ liệu. Sau đó, chúng tôi thực hiện các chức năng khác nhau trên đó như nhóm, lọc, nối, sắp xếp, v.v.  Cuối cùng, bạn có thể kết xuất dữ liệu trên màn hình hoặc bạn có thể lưu trữ kết quả trở lại HDFS.

## **APACHE HIVE**

* Facebook đã tạo HIVE cho những người thông thạo SQL. Do đó, HIVE khiến họ cảm thấy như ở nhà khi làm việc trong Hệ sinh thái Hadoop.
* Về cơ bản, HIVE là thành phần lưu trữ dữ liệu thực hiện việc đọc, ghi và quản lý các tập dữ liệu lớn trong môi trường phân tán bằng giao diện giống SQL.

HIVE + SQL = HQL

* Ngôn ngữ truy vấn của Hive được gọi là Ngôn ngữ truy vấn Hive (HQL), rất giống với SQL.
* Nó có 2 thành phần cơ bản: Hive Command Line và JDBC/ODBC driver .
* Giao diện dòng lệnh Hive được sử dụng để thực thi các lệnh HQL.
* Trong khi đó, Kết nối cơ sở dữ liệu Java ( JDBC ) và Kết nối cơ sở dữ liệu đối tượng ( ODBC ) được sử dụng để thiết lập kết nối từ bộ lưu trữ dữ liệu.
* Thứ hai, Hive có khả năng mở rộng cao. Vì nó có thể phục vụ cả hai mục đích, tức là xử lý tập dữ liệu lớn (tức là xử lý truy vấn hàng loạt) và xử lý thời gian thực (tức là xử lý truy vấn tương tác).
* Nó hỗ trợ tất cả các kiểu dữ liệu nguyên thủy của SQL.
* Bạn cũng có thể sử dụng các hàm được xác định trước hoặc viết các hàm do người dùng xác định (UDF) phù hợp để đáp ứng các nhu cầu cụ thể của mình.  
  Bạn thậm chí có thể xem chi tiết về Dữ liệu lớn với [Khóa đào tạo Kỹ thuật dữ liệu Azure tại Singapore](https://www.edureka.co/microsoft-azure-data-engineering-certification-course-singapore) .

## APACHE MAHOUT

Bây giờ, chúng ta hãy nói về Mahout, công ty nổi tiếng về học máy. Mahout cung cấp một môi trường để tạo các ứng dụng học máy có khả năng mở rộng.

Vậy **học máy là gì?**

Các thuật toán học máy cho phép chúng ta xây dựng những cỗ máy tự học có khả năng tự phát triển mà không cần lập trình rõ ràng. Dựa trên hành vi của người dùng, mẫu dữ liệu và trải nghiệm trong quá khứ, nó đưa ra các quyết định quan trọng trong tương lai. Có thể gọi nó là hậu duệ của Trí tuệ nhân tạo (AI).

Mahout làm gì?

Nó thực hiện **collaborative filtering (**lọc cộng tác), **clustering**(phân cụm) và **classification** (phân loại). Một số người cũng coi bộ vật phẩm thường xuyên bị thiếu là chức năng của Mahout. Hãy để chúng tôi hiểu chúng một cách riêng lẻ:

1. **Collaborative filtering(**Lọc cộng tác): Mahout khai thác hành vi của người dùng, mô hình và đặc điểm của họ, từ đó dự đoán và đưa ra đề xuất cho người dùng. Trường hợp sử dụng điển hình là trang web thương mại điện tử.
2. **Clustering** (Phân cụm): Nó tổ chức một nhóm dữ liệu tương tự với nhau như các bài viết có thể chứa blog, tin tức, tài liệu nghiên cứu, v.v.
3. **Classification(**Phân loại) : Có nghĩa là phân loại và phân loại dữ liệu thành các phần nhỏ khác nhau như các bài báo có thể được phân loại thành blog, tin tức, bài luận, tài liệu nghiên cứu và các danh mục khác.
4. **Frequent item set missing(**Nhóm mục thường xuyên bị thiếu) : Tại đây Mahout kiểm tra những đối tượng nào có khả năng xuất hiện cùng nhau và đưa ra đề xuất nếu chúng bị thiếu. Ví dụ, điện thoại di động và vỏ bọc nói chung được kết hợp với nhau. Vì vậy, nếu bạn tìm kiếm một chiếc điện thoại di động, nó cũng sẽ đề xuất cho bạn vỏ và vỏ.

Mahout cung cấp một dòng lệnh để gọi các thuật toán khác nhau. Nó có một bộ thư viện được xác định trước chứa sẵn các thuật toán sẵn có khác nhau cho các trường hợp sử dụng khác nhau.

## **APACHE SPARK**

* Apache Spark là một khung phân tích dữ liệu thời gian thực trong môi trường điện toán phân tán.
* Spark được viết bằng Scala và ban đầu được phát triển tại Đại học California, Berkeley.
* Nó thực thi các tính toán trong bộ nhớ để tăng tốc độ xử lý dữ liệu trên Map-Reduce.
* Nó nhanh hơn 100 lần so với Hadoop để xử lý dữ liệu quy mô lớn bằng cách khai thác các tính toán trong bộ nhớ và các tối ưu hóa khác. Vì vậy, nó đòi hỏi khả năng xử lý cao hơn Map-Reduce.



Như bạn có thể thấy, Spark có rất nhiều thư viện cấp cao, bao gồm hỗ trợ cho R, SQL, Python, Scala, Java, v.v. Những thư viện tiêu chuẩn này tăng khả năng tích hợp liền mạch trong quy trình làm việc phức tạp. Về điều này, nó cũng cho phép nhiều bộ dịch vụ khác nhau tích hợp với nó như MLlib, GraphX, SQL + Data Frames, Streaming services, v.v. để tăng khả năng của nó.

Đây là một câu hỏi rất phổ biến trong đầu mọi người:

“Apache Spark: Kẻ giết người hay vị cứu tinh của Apache Hadoop?” – O'Reily

Câu trả lời cho vấn đề này – Đây không phải là sự so sánh giữa táo với táo. Apache Spark phù hợp nhất để xử lý thời gian thực, trong khi Hadoop được thiết kế để lưu trữ dữ liệu phi cấu trúc và thực hiện xử lý hàng loạt dữ liệu đó. Khi chúng tôi kết hợp, khả năng của Apache Spark, tức là tốc độ xử lý cao, phân tích nâng cao và hỗ trợ tích hợp nhiều lần với hoạt động chi phí thấp của Hadoop trên phần cứng thông thường, nó sẽ mang lại kết quả tốt nhất.

Đó là lý do tại sao Spark và Hadoop được nhiều công ty sử dụng cùng nhau để xử lý và phân tích Dữ liệu lớn được lưu trữ trong HDFS.

## **APACHE DRILL**

Đúng như tên gọi, Apache Drill được sử dụng để truy sâu vào bất kỳ loại dữ liệu nào. Đây là một ứng dụng nguồn mở hoạt động với môi trường phân tán để phân tích các tập dữ liệu lớn.

* Nó là một bản sao của Google Dremel.
* Nó hỗ trợ các loại cơ sở dữ liệu và hệ thống tệp NoSQL khác nhau, đây là một tính năng mạnh mẽ của Drill. Ví dụ: Azure Blob Storage, Google Cloud Storage, HBase, MongoDB, MapR-DB HDFS, MapR-FS, Amazon S3, Swift, NAS và các tệp cục bộ.

Vì vậy, về cơ bản, mục đích chính của Apache Drill là cung cấp khả năng mở rộng để chúng tôi có thể xử lý hàng petabyte và exabyte dữ liệu một cách hiệu quả (hoặc bạn có thể nói trong vài phút).

* Sức mạnh chính của Apache Drill nằm ở việc kết hợp nhiều kho dữ liệu khác nhau chỉ bằng cách sử dụng một truy vấn duy nhất.
* Apache Drill về cơ bản tuân theo ANSI SQL.
* Nó có hệ số khả năng mở rộng mạnh mẽ trong việc hỗ trợ hàng triệu người dùng và phục vụ các yêu cầu truy vấn của họ trên dữ liệu quy mô lớn.

## **APACHE ZOOKEEPER**

* Apache Zookeeper là điều phối viên của bất kỳ công việc Hadoop nào bao gồm sự kết hợp của nhiều dịch vụ khác nhau trong Hệ sinh thái Hadoop.
* Apache Zookeeper phối hợp với nhiều dịch vụ khác nhau trong môi trường phân tán.

Trước Zookeeper, việc phối hợp giữa các dịch vụ khác nhau trong Hệ sinh thái Hadoop là rất khó khăn và tốn thời gian. Các dịch vụ trước đó gặp nhiều vấn đề về tương tác như cấu hình chung trong khi đồng bộ dữ liệu. Ngay cả khi các dịch vụ được định cấu hình, những thay đổi trong cấu hình của dịch vụ sẽ khiến việc xử lý trở nên phức tạp và khó khăn. Việc phân nhóm và đặt tên cũng là một yếu tố tốn thời gian.

Do những vấn đề trên, Zookeeper đã được ra đời. Nó tiết kiệm rất nhiều thời gian bằng cách thực hiện  đồng bộ hóa, bảo trì cấu hình, nhóm và đặt tên.

Mặc dù đây là một dịch vụ đơn giản nhưng nó có thể được sử dụng để xây dựng các giải pháp mạnh mẽ.

Những tên tuổi lớn như Rackspace, Yahoo, eBay sử dụng dịch vụ này trong nhiều trường hợp sử dụng của họ và do đó, bạn có thể hình dung được tầm quan trọng của Zookeeper.

## **APACHE OOZIE**

Hãy coi Apache Oozie như một dịch vụ đồng hồ và báo thức bên trong Hệ sinh thái Hadoop. Đối với các công việc của Apache, Oozie giống như một người lập lịch trình. Nó lên lịch các công việc của Hadoop và liên kết chúng lại với nhau thành một công việc logic.

Có hai loại công việc Oozie:

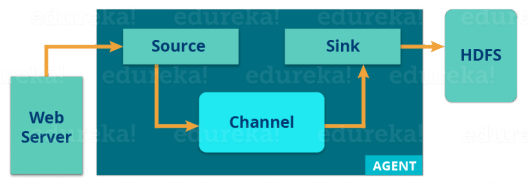
1. Quy trình làm việc của Oozie : Đây là tập hợp các hành động tuần tự được thực hiện . Bạn có thể coi đó là một cuộc chạy tiếp sức. Nơi mỗi vận động viên chờ đợi người cuối cùng hoàn thành phần thi của mình.
2. Điều phối viên Oozie : Đây là các công việc Oozie được kích hoạt khi dữ liệu được cung cấp cho nó. Hãy coi đây như hệ thống kích thích phản ứng trong cơ thể chúng ta. Theo cách tương tự như cách chúng ta phản ứng với một kích thích bên ngoài, điều phối viên Oozie phản ứng với sự sẵn có của dữ liệu và nó sẽ không hoạt động.

## **APACHE FLUME**

Nhập dữ liệu là một phần quan trọng trong Hệ sinh thái Hadoop của chúng tôi.

* Flume là một dịch vụ giúp nhập dữ liệu phi cấu trúc và bán cấu trúc vào HDFS.
* Nó cung cấp cho chúng tôi một giải pháp đáng tin cậy và có tính phân tán, đồng thời giúp chúng tôi **collecting(**thu thập), **aggregating (**tổng hợp) và **moving large amount of data sets**.(di chuyển lượng lớn tập dữ liệu)
* Nó giúp chúng tôi nhập dữ liệu phát trực tuyến từ nhiều nguồn khác nhau như lưu lượng truy cập mạng, phương tiện truyền thông xã hội, tin nhắn email, tệp nhật ký, v.v. trong HDFS.

     Bây giờ, chúng ta hãy hiểu kiến ​​trúc của Flume từ sơ đồ bên dưới:



Có một  **Flume agent(**tác nhân Flume) nhập dữ liệu truyền phát từ nhiều nguồn dữ liệu khác nhau vào HDFS. Từ sơ đồ, bạn có thể dễ dàng hiểu rằng máy chủ web cho biết nguồn dữ liệu. Twitter là một trong những nguồn nổi tiếng về truyền dữ liệu.

Tác nhân khói có 3 thành phần: **source(**nguồn), **sink(**chìm) và **channel(**kênh) .

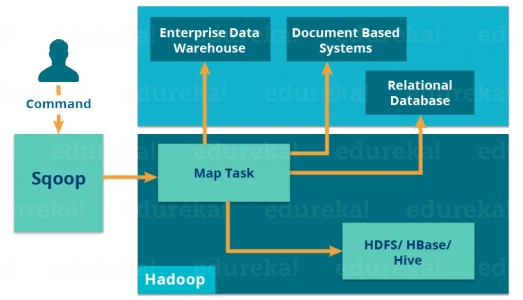
1. Nguồn : nó chấp nhận dữ liệu từ dòng đến và lưu trữ dữ liệu trong kênh.
2. Kênh:  nó hoạt động như bộ nhớ cục bộ hoặc bộ nhớ chính. Kênh là nơi lưu trữ tạm thời giữa nguồn dữ liệu và dữ liệu liên tục trong HDFS.
3. Chìm: Sau đó, thành phần cuối cùng của chúng tôi, tức là Chìm, thu thập dữ liệu từ kênh và cam kết hoặc ghi dữ liệu vào HDFS vĩnh viễn.

## **APACHE SQOOP**

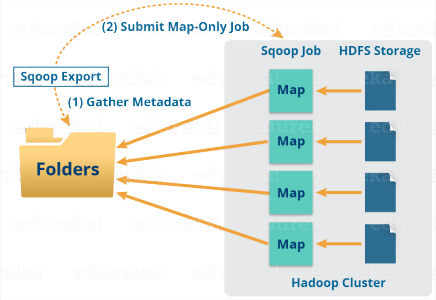
Bây giờ, chúng ta hãy nói về một dịch vụ nhập dữ liệu khác, đó là Sqoop. Sự khác biệt chính giữa Flume và Sqoop là:

* Flume chỉ nhập dữ liệu phi cấu trúc hoặc dữ liệu bán cấu trúc vào HDFS.
* Trong khi Sqoop có thể nhập cũng như xuất dữ liệu có cấu trúc từ kho dữ liệu RDBMS hoặc Enterprise sang HDFS hoặc ngược lại.

Hãy cho chúng tôi hiểu cách Sqoop hoạt động bằng sơ đồ bên dưới:



Khi chúng tôi gửi lệnh Sqoop, nhiệm vụ chính của chúng tôi sẽ được chia thành các nhiệm vụ phụ do từng Nhiệm vụ bản đồ xử lý nội bộ. Nhiệm vụ bản đồ là nhiệm vụ phụ, nhập một phần dữ liệu vào Hệ sinh thái Hadoop. Nói chung, tất cả các tác vụ Bản đồ sẽ nhập toàn bộ dữ liệu.



Xuất khẩu cũng hoạt động theo cách tương tự.

Khi chúng tôi gửi Công việc của mình, nó sẽ được ánh xạ vào Nhiệm vụ bản đồ để mang lại khối dữ liệu từ HDFS. Các khối này được xuất đến đích dữ liệu có cấu trúc. Kết hợp tất cả các khối dữ liệu đã xuất này, chúng tôi nhận được toàn bộ dữ liệu tại đích, trong hầu hết các trường hợp là RDBMS (MYSQL/Oracle/SQL Server).

## **SOLR APACHE & LUCENE**

Apache Solr và Apache Lucene là hai dịch vụ được sử dụng để tìm kiếm và lập chỉ mục trong Hệ sinh thái Hadoop.

* Apache Lucene dựa trên Java, cũng giúp kiểm tra chính tả.
* Nếu Apache Lucene là động cơ thì Apache Solr là chiếc xe được chế tạo xung quanh nó. Solr là một ứng dụng hoàn chỉnh được xây dựng xung quanh Lucene.
* Nó sử dụng thư viện tìm kiếm Lucene Java làm lõi để tìm kiếm và lập chỉ mục đầy đủ.

## **APACHE AMBARI**

Ambari là một Dự án Quỹ Phần mềm Apache nhằm mục đích làm cho hệ sinh thái Hadoop dễ quản lý hơn.

Nó bao gồm phần mềm để cung cấp, quản lý và giám sát các cụm Apache Hadoop.

Ambari cung cấp:

1. Cung cấp cụm Hadoop :
   * Nó cung cấp cho chúng tôi quy trình từng bước để cài đặt dịch vụ Hadoop trên một số máy chủ.
   * Nó cũng xử lý cấu hình các dịch vụ Hadoop trên một cụm.
2. Quản lý cụm Hadoop:
   * Nó cung cấp dịch vụ quản lý trung tâm để khởi động, dừng và định cấu hình lại các dịch vụ Hadoop trên toàn cụm.
3. Giám sát cụm Hadoop:
   * Để theo dõi sức khỏe và tình trạng, Ambari cung cấp cho chúng tôi một bảng điều khiển.

* Khung cảnh báo Amber là một dịch vụ cảnh báo thông báo cho người dùng bất cứ khi nào cần chú ý. Ví dụ: nếu một nút bị hỏng hoặc dung lượng ổ đĩa thấp trên một nút, v.v.  
    
  Tìm hiểu thêm về Dữ liệu lớn và các ứng dụng của nó từ [Chứng chỉ Kỹ sư Dữ liệu Azure](https://www.edureka.co/microsoft-azure-data-engineering-certification-course) .

Cuối cùng, tôi muốn thu hút sự chú ý của bạn vào ba điều quan trọng:

1. Hệ sinh thái Hadoop có được thành công nhờ vào toàn bộ cộng đồng nhà phát triển, nhiều công ty lớn như Facebook, Google, Yahoo, Đại học California (Berkeley), v.v. đã góp phần nâng cao khả năng của Hadoop.
2. Bên trong Hệ sinh thái Hadoop, kiến ​​thức về một hoặc hai công cụ (các thành phần Hadoop) sẽ không giúp ích gì trong việc xây dựng giải pháp. Bạn cần tìm hiểu một bộ thành phần Hadoop hoạt động cùng nhau để xây dựng giải pháp.
3. Dựa trên các trường hợp sử dụng, chúng tôi có thể chọn một bộ dịch vụ từ Hệ sinh thái Hadoop và tạo giải pháp phù hợp cho tổ chức.

# 