# Hadoop

## Hadoop là gì?

Là một framework cho phép phát triển ứng dụng phân tán có cường độ dữ liệu lớn một cách miễn phí.Nó giúp lưu trữ và xử lý Big Data áp dụng MapReduce.

Hadoop viết bằng Java, tuy nhiên nhờ cơ chế streaming mà nó cho phép phát triển các ứng dụng phân tán trên java và cá C++, Python, ...

## Cấu trúc Hadoop

Hadoop gồm 4 module :

* *Hadoop Common*: Các thư viện và tiện ích cần thiết để các module khác sử dụng. Chứa hệ thống file, lớp OS trừu tượng, mã lệnh Java để khởi động.
* *Hadoop Yarn*: Framework quản lý tiến trình và tài nguyên của các cluster.
* Hadoop Distributed File System (HDFS): Hệ thống file phân tán cung cấp truy cập thông lượng cao cho ứng dụng khai thác dữ liệu.
* Hadoop MapReduce: Hệ thống dựa trên Yarn dùng để xử lý song song các tập dữ liệu lớn.

## Hadoop hoạt động như thế nào?

*Giai đoạn 1:*

Submit job lên Hadoop với yêu cầu xử lý cùng với các thông tin:

* Truyền dữ liệu lên server để phân tán và cho ra output.
* Các dữ liệu chạy qua hai hàm chính là Map và Reduce:
* Map: quét toàn bộ dữ liệu và phân tán thành dữ liệu con
* Reduce: Thu thập các dữ liệu con và sắp xếp lại chúng
* Các thiết lập cụ thể thông qua thông số truyền vào

*Giai đoạn 2:*

Hadoop job client submit job (fire jar, file thực thi) và bắt đầu lịch làm việc (Job Tracker) đưa job vào hàng đợi. Sau khi tiếp nhận yêu cầu từ Job Tracker, server cha(master) sẽ phân chia công việc cho các server con(slave). Các server con sẽ thực hiện các job được giao và trả kết quả cho server cha.

*Giai đoạn 3:*

TaskTracker dùng để kiểm tra đảm bảo các MapReduce hoạt động bình thường và kiểm tra kết quả nhận được.

## Ưu điểm của Hadoop

* Cho phép nhanh chóng viết và kiểm tra các hệ thống phân tán
* Không dựa vào cơ chế chịu lỗi của phần cứng fault-tolerance and high availability (FTHA) , thay vào đó Hadoop có các thư viện để phát hiện xử lý lỗi ở lớp ứng dụng.
* Có thể phát triển trên nhiều server với cấu trúc master-slave để đảm bảo công việc linh hoạt và không bị ngắt quãng do chia nhỏ công việc cho các server slave.
* Tương thích mọi nền tảng như Windows, Linux, MacOS.

## Yarn

Viết tắt của “Yet-Another-Resource-Negotiator” là một framework hỗ trợ phát triển ứngdụng phân tán. Yarn cung cấp deanons và APIs cần thiết để phát triển ứng dụng phân tán, xử lí và lập lịch sử dụng tài nguyên tính toán và giám sát quá trình thực thi ứng dụng. Yarn tổng quát hơn mapreduce đời đầu.

\*\*\* MapReduce đời đầu:

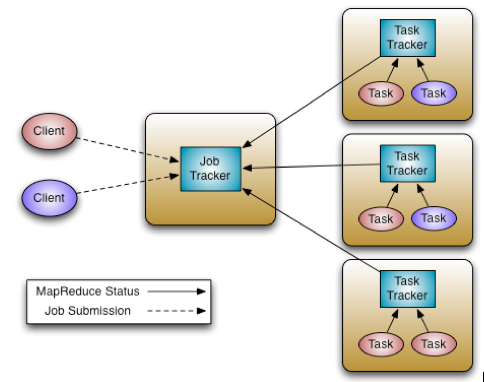
Chia ra 2 modules:

-HDFS

-JobTracker/ TaskTracker



HDFS chia dữ liệu thành các block (thường có kích thước 128M) và sao chép các block này ra các máy tính khác nhau. Thông tin metadata về files quản lý bởi HDFS như số block, vị trí block, quyền hạn v.v được quản lý tại một máy tính gọi là NameNode. Các máy tính chứa dữ liệu gọi là DataNode. Khi Datanode bị sự cố, NameNode sẽ dựa vào thông tin metadata đang quản lý để sao chép các block dữ liệu trên Datanode bị sự cố sang các máy khác để đảm bảo tính toàn vẹn của dữ liệu.



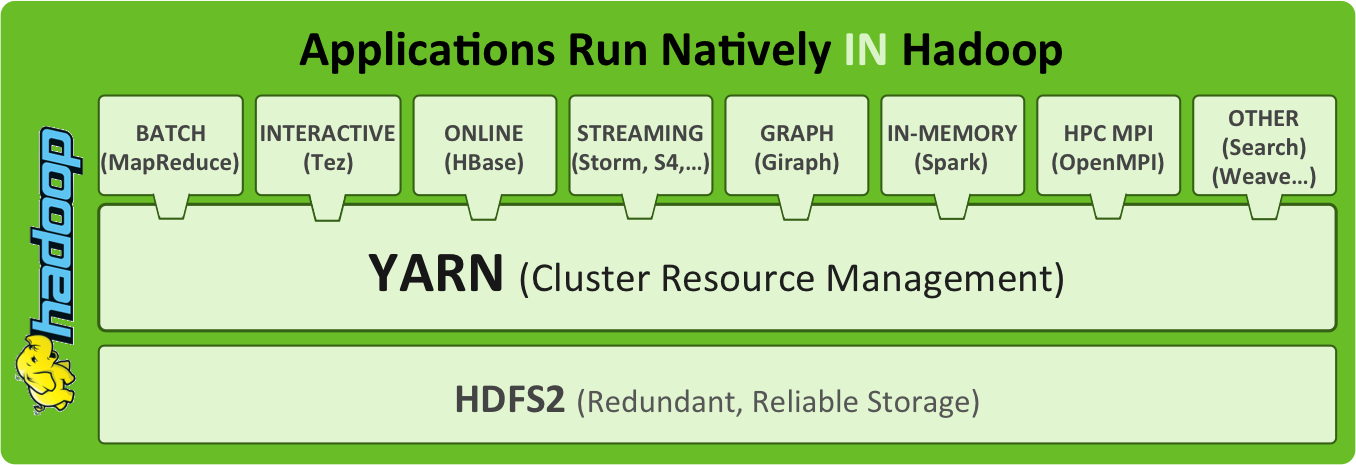
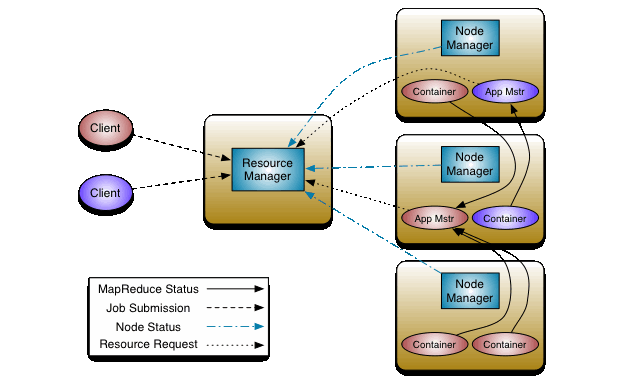
Trên HDFS, để chạy các thủ tục MapReduce, Hadoop 1 đặt ra 2 khái niệm JobTracker và TaskTracker. JobTracker làm nhiệm vụ kiểm soát tài nguyên tính toán của toàn cluster và chia việc cho các node tính toán. TaskTracker làm nhiệm vụ thông báo tình trạng tính toán của node trong cluster đồng thời chạy các Task được giao bởi JobTracker. Mỗi Task này có thể là một Mapper hay 1 Reducer. Người dùng giao việc cho cluster bằng cách nói chuyện trực tiếp với JobTracker. Dữ liệu sử dụng bởi các Task được lấy thông qua HDFS API.

\*\*\*MapReduce 2 (YARN):

Chia JobTracker thành:

- Resource Manager( RM) : quanr lý toàn bộ tài nguyên của cluster

- Application Master(AM): quản lý vòng đời của job.

**Chạy ứng dụng trên YARN** [[](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/yarn_architecture.gif_ajcycucx8o)](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/yarn_architecture.gif_ajcycucx8o)

Quá trình 1 ứng dụng chạy trên YARN được mô tả bằng sơ đồ trên. Cụ thể các bước khởi động 1 ứng dụng như sau:

* Client giao 1 task cho Resource Manager
* Resource Manager tính toán tài nguyên cần thiết theo yêu cầu của ứng dụng và tạo 1 App Master (App Mstr). Application Master được chuyển đến chạy 1 một node tính toán. Application Master sẽ liên lạc với các NodeManager ở các node khác để ra yêu cầu công việc cho node này.
* Node Manager nhận yêu cầu và chạy các task trên container
* Các thông tin trạng thái thay vì được gửi đến JobTracker sẽ được gửi đến App Master.

Các ứng dụng khác nhau sẽ có các AppMstr khác nhau và sẽ được khởi động ở các máy trạm khác nhau. Điều này làm việc khi một máy trạm bị lỗi phần cứng và sập, các ứng dụng khác không bị ảnh hưởng, chỉ ứng dụng có AppMstr chạy trên máy đó bị ảnh hưởng.

#### Lập lịch (schedulers)

Resource Manager dùng một module lập lịch mà có thể **tháo lắp tự do**. Hiện tại YARN có 2 schedulers là:

* CapacityScheduler
* FairScheduler

Scheduler mặc định của YARN là FairScheduler ở đó các Job sẽ được cho vào hàng đợi.

#### Tài nguyên

Ở YARN, tài nguyên không đơn thuần là CPU cores nữa mà bao gồm cả bộ nhớ, DiskIO, và GPUs... Một ứng dụng khi khởi động sẽ yêu cầu

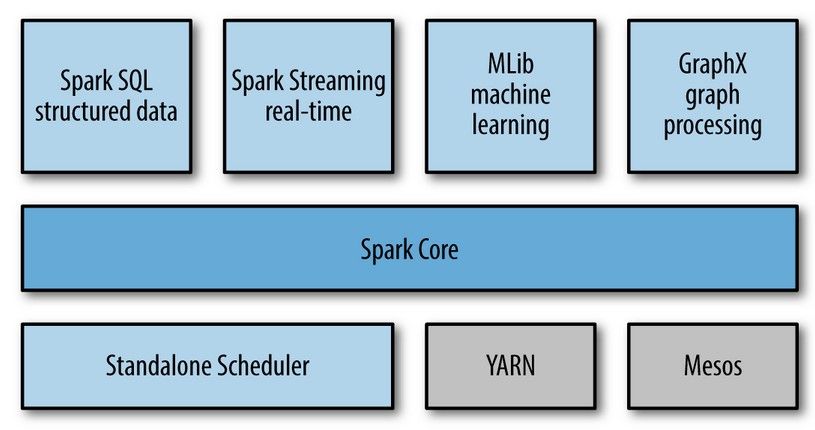
* Priority: đặc quyền?
* Bộ nhớ MB: chương trình ứng dụng dự định sẽ sử dụng bao nhiêu MB bộ nhớ
* CPU: ứng dụng dự định sử dụng bao nhiêu cores.
* Số lượng containers.

Containers là số lượng tasks có thể được triển khai song song trên 1 node tính toán. Giới hạn tài nguyên sử dụng bởi 1 containers được thực hiện thông qua tính năng [cgroups](http://en.wikipedia.org/wiki/Cgroups) của Linux kernels.

**II. Spark**

Là một framework cho phép xây dựng các mô hình dự đoán nahnh chóng với việc thực hiện trên một nhóm máy tính có thể tính toán cùng lúc trên toàn bộ tập dữ liệu mà không phải trích xuất mẫu tính toán thử nghiệm. Việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện toàn bộ trên ram.

\*\*\*Thành phần:



+ Spark Core: lập lịch cho các tác vụ, quản lý bộ nhớ, fault recovery, tương tác với hệt hống lưu trữ,... Hơn nữa, spark corecung cấp API để định nghĩa RDD (Resilient Ditributed Dataset) là tập hợp các item được phân tán trên các node của clusster

+Spark SQL: cho phép truy ván dữ liệu

+Spark Streaming cung cấp API để dễ dàng xử lý dữ liệu stream

+MLlib cung cấp nhiều thuật toán học máy

+GraphX là thư viện xử lý đồ thị.

**1. RDD**

-Là tập hợp các item được phân tán trên các node của cluster

- RDD hỗ trợ 2 kiểu thao tác:

+ Transformations: tạo dataset từ dữ liệu có sẵn.

+ Action: trả về giá trị cho trình điều khiển sau khi tính toán trên dataset.

**2. Spark streaming**

Spark Streaming đóng vai trò thượng tầng trong toàn bộ nền tảng Spark computing framework. Spark là một hệ thống xử lý dữ liệu hàng loạt (batch processing) có thể hoạt động đơn lẻ hoặc đóng vai trò là quản lý dữ liệu tương tự như YARN hoặc MEsos. Spark Streaming hỗ trợ các kỹ thuật xử lý dữ liệu như windowing, joining stream,…

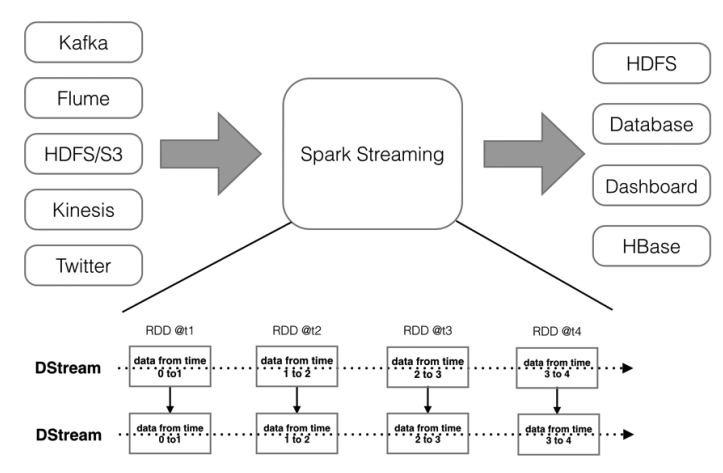
Spark Streaming có thể nhận dữ liệu từ nhiều nguồn khác nhau, bao gồm Kafka, Flume, Kinesis, Twitter, và TCP sockets. Một số ngữ cảnh chính khi triển khai hệ thống Spark Streaming là **Discretized Streams (DStreams)**; có thể được hiểu là một luồng dữ liệu liên tục được tạo ra từ các nguồn đầu vào

Bên trong một DStream bao gồm các **Resilient Distributed Datasets (RDDs)** đóng vai trò cốt lõi trong kiến trúc Spark.

Các RDD được tạo ra dựa trên các khoảng thời gian (time interval) được cấu hình trong ứng dụng Spark Streaming; nó đóng vai trò định mức tần số dữ liệu sẽ được xử lý bởi ứng dụng. Hình 2 mô tả kiến trúc hoạt động của mô hình DStreams.

Spark streaming xử lý dữ liệu với các tính năng cấp cao như map, reduce, join, và window. Sau khi xử lý , các dữ liệu sẽ được lưu trữ vào các hệ thống như HDFS, HBase, Solr, và kết nối đến các hệ thống hiển thị trên giao diện quả lý hoặc được điều khiển đến một hệ thống Kafka khác cho việc xử lý tiếp theo.

Khi nhận được các luồng dữ liệu đầu vào, Spark Streaming phân chia dữ liệu thành các khối nhỏ hơn (mini batch). Mỗi khối này được lưu trữ trong một RDD và các RDD này sau đó sẽ được xử lý bởi Spark để tạo ra các RDD mới.



III. RESTful

REST là viết tắt của cụm từ Representational State Transfer (đôi khi còn được viết là ReST) là một kiểu kiến trúc được sử dụng trong việc giao tiếp giữa các máy tính (máy tính cá nhân và máy chủ của trang web) trong việc quản lý các tài nguyên trên internet. REST được sử dụng rất nhiều trong việc phát triển các ứng dụng Web Services sử dụng giao thức HTTP trong giao tiếp thông qua mạng internet. Các ứng dụng sử dụng kiến trúc REST này thì sẽ được gọi là ứng dụng phát triển theo kiểu RESTful.

REST đặt ra một quy tắc đòi hỏi lập trình viên xác định rõ ý định của mình thông qua các phương thức của HTTP. Thông thường ý định đó bao gồm lấy dữ liệu, trèn dữ liệu, cập nhập dữ liệu hoặc xóa dữ liệu. Vậy khi bạn muốn thực hiện một trong các ý định trên hãy lưu ý các quy tắc sau:

* Để tạo một tài nguyên trên máy chủ, bạn cần sử dụng phương thức POST.
* Để truy xuất một tài nguyên, sử dụng GET.
* Để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó, sử dụng PUT.
* Để huỷ bỏ hoặc xoá một tài nguyên, sử dụng DELETE.