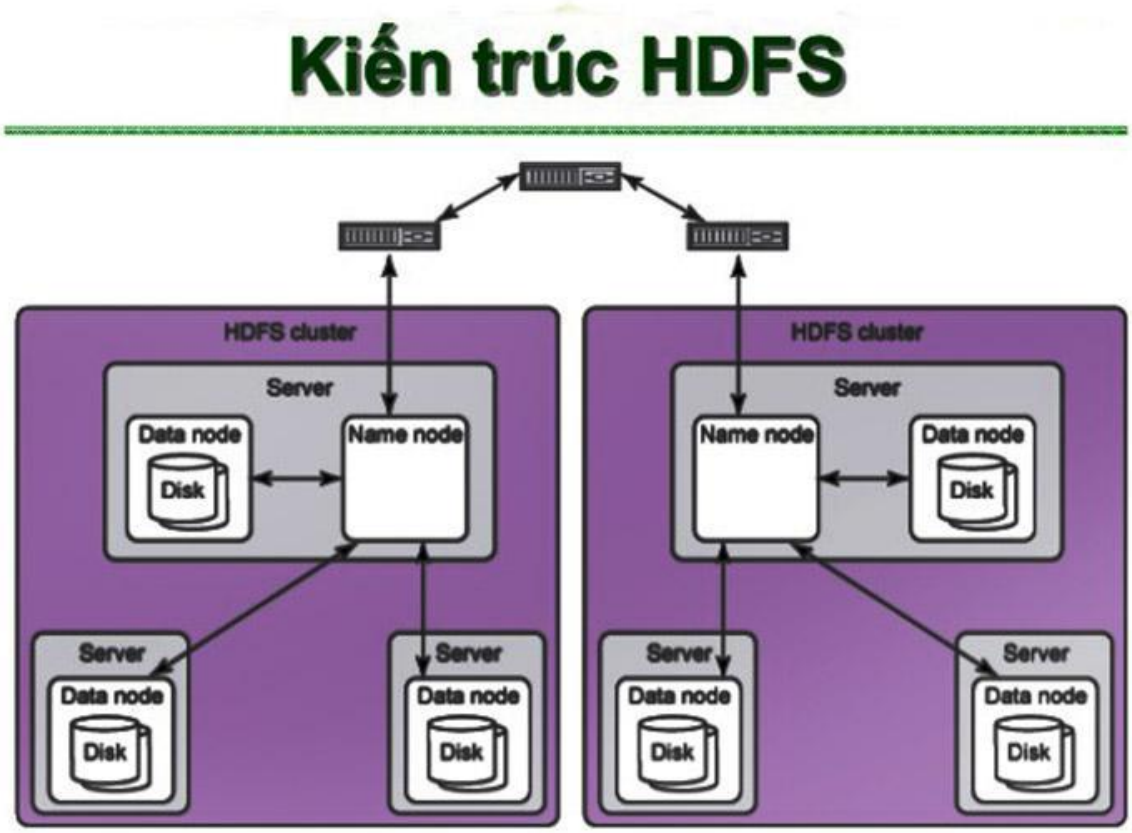
I. HDFS



Hệ thống tệp phân tán tuân theo **Master - Slave.** Mỗi cụm bao gồm 1 nút chính duy nhất và các nút phụ. Bên trong các tệp được chia thành một hoặc nhiều khối và mỗi khối được lưu trữ trên các máy phụ khác nhau .

- Nút **Master** lưu trữ và quản lý không gian tên hệ thống tệp, đó là thông tin về các khối tệp như vị trí khối, quyền, v.v. Các **Slave** lưu trữ các khối dữ liệu của tệp.
- Nút chính là NameNode và DataNodes là các nút phụ.
- NameNode là trung tâm của Hệ thống tệp phân tán Hadoop. Nó duy trì và quản lý không gian tên hệ thống tệp và cung cấp quyền truy cập phù hợp cho các máy khách. Nó lưu trữ dưới dạng 2 tệp : Fsimage và Edit log.

1. NameNode

- Trong một Hadoop cluster, chỉ có một Namenode hoạt động tại 1 thời điểm.
- Namenode có nhiệm vụ duy trì và quản lý các DataNode.
- Namenode là nơi lưu trữ và cập nhật MetaData như: logs, tên file, size, vị trí các DataNode. Namenode cũng chỉ dẫn các Datanode thực hiện các thao tác như thêm, xóa, replicate, …
- Namenode luôn lắng nghe và theo dõi để đảm bảo các datanode còn hoạt động, quản lý lượng truy cập tới các datanode và cân bằng dung lượng lưu trữ trong các datanode.
  - ⇨ ***Vì những đặc điểm và chức năng trên, nếu NameNode gặp sự cố và dừng hoạt động, cả cụm Hadoop Cluster sẽ dừng hoặt động, tuy nhiên dữ liệu sẽ không bị mất.(What all the reasons for NameNode Failure in Hado... - Cloudera Community - 306669)***

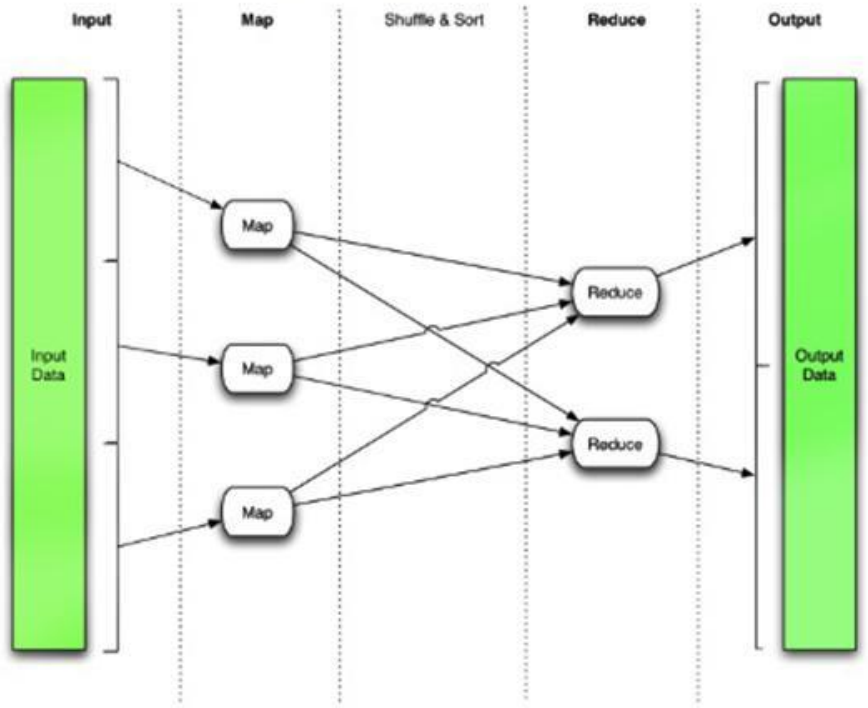
2. **Datanode**
   - Datanode được sử dụng để lưu trữ data trong hadoop cluster khi dữ liệu được đưa vào hdfs. Một hadoop cluster thường có từ 1 đến 500 hoặc nhiều hơn các Datanodes.
   - Datanode là nơi chạy các tiến trình xử lý dữ liệu.
   - Các datanode sẽ định kỳ gửi dữ liệu về status của nó tới namenode, mặc định là 3s.
3. **Secondary namenode**
   - Secondary namenode đóng vai trò như một node phụ cùng chạy với namenode. Secondary namenode không phải một node dự phòng cho namenode mà có các vai trò và nhiệm vụ rõ ràng.
   - Nó thường xuyên đọc các file, các metadata lưu trữ trên RAM của datanode và ghi vào ổ cứng.

- Nó liên tục kiểm tra tính chính xác của các tệp tin lưu trữ trên các datanode.

4. **Block**
    - Dữ liệu được đưa vào HDFS sẽ được chia nhỏ thành nhiều block và nằm rải rác ở khắp các nơi trong các datanode. Việc dữ liệu được chia làm bao nhiêu block và mỗi block nằm ở đâu sẽ được quyết định bở metadata trong namenode. Mặc định, mỗi block sẽ có kích thước là 123M(có thể chỉnh sửa).
5. **Replication**
    - HDFS có khả năng phục hồi sau lỗi tốt. Data được chia thành nhiều blocks nằm rải rác ở các datanode và các datanode sẽ được replicate thành các bản sao nằm ở các datanode khác nhau. Theo mặc định, mỗi blocks sẽ có 3 bản sao(có thể cấu hình lại).
    - Trong trường hợp một datanode gặp sự cố, thì các blocks mà nó đang lưu trữ vẫn có các bản sao nằm ở các datanode khác.
    - Việc replicate càng nhiều sẽ càng giảm rủi ro mất mát dữu liệu những cũng chiếm nhiều bộ nhớ hơn.

II. MapReduce

Map Reduce là gì ?



MapReduce là một mô hình được thiết kế độc quyền bởi Google để xử lý các tập dữ liệu lớn song song và phân tán. MapReduce bao gồm 2 thủ tục chính là Map và Reduce.

1. **Map**

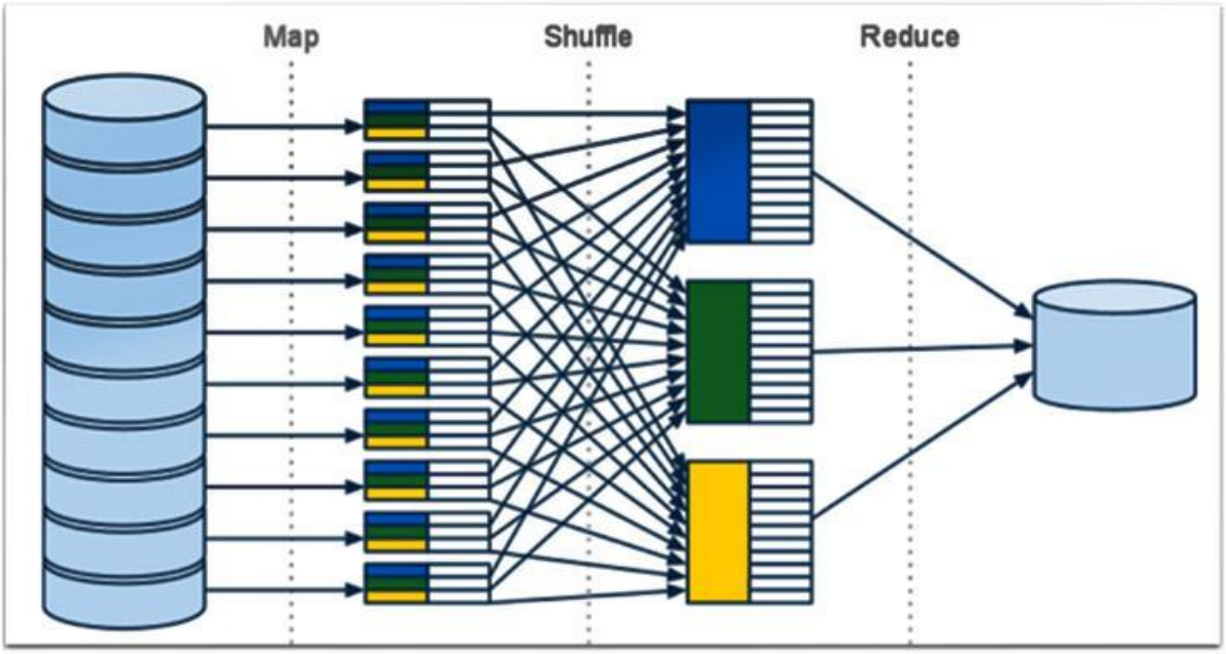
   Hàm Map có nhiệm vụ nhận các Input là các Data block và chia chúng vào các partition và tạo ra các cặp key/value trung gian. Sau đó, nó ghi dữ liệu xuống đĩa cứng và thông báo cho hàm reduce để nhận dữ liệu.

2. **Suffle**

   Giữa pha map và reuce là suffle. Sau khi map hoàn thành công việc thì suffle sẽ thu thập và tổng hợp các cặp key/values rồi chuyển qua cho reduce tiếp tục xử lý.

3. **Reduce**

   Hàm reduce sẽ tiếp nhận các cặp key/value thực hiện các tác vụ để đưa ra output.



Nguyên tắc hoạt động

Mapreduce hoạt động dựa vào nguyên tắc chính là "Chia để trị", như sau:

- Phân chia các dữ liệu cần xử lý thành nhiều phần nhỏ trước khi thực hiện.
- Xử lý các vấn đề nhỏ theo phương thức song song trên các máy tính rồi phân tán hoạt động theo hướng độc lập.
- Tiến hành tổng hợp những kết quả thu được để đề ra được kết quả sau cùng.

Các bước thực hiện:

1. Tiến hành chuẩn bị các dữ liệu đầu vào để cho Map() có thể xử lý.
2. Lập trình viên thực thi các mã Map() để xử lý.

3. Tiến hành trộn lẫn các dữ liệu được xuất ra bởi Map() vào trong Reduce Processor.
4. Tiến hành thực thi tiếp mã Reduce() để có thể xử lý tiếp các dữ liệu cần thiết.
5. Thực hiện tạo các dữ liệu xuất ra cuối cùng.

III. Yarn

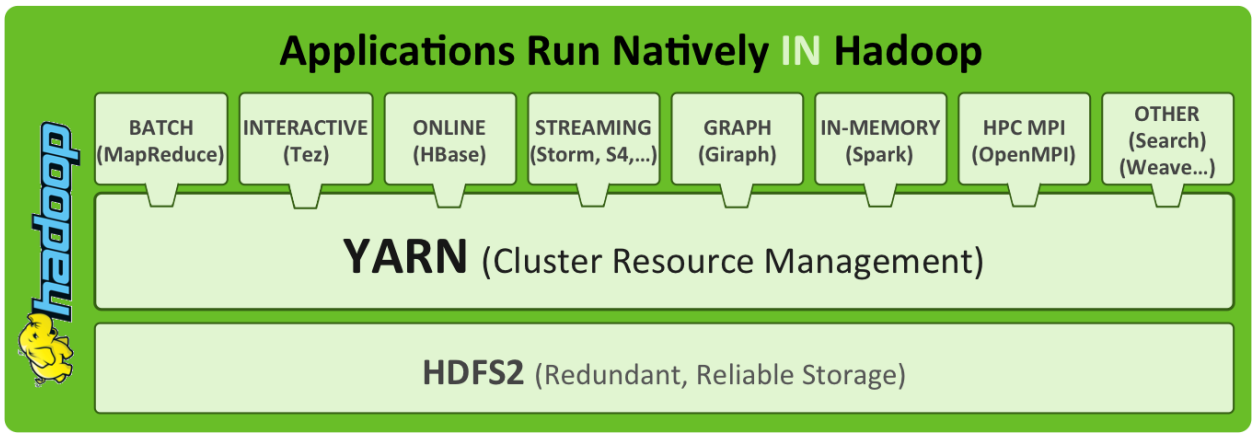
Kiến trúc mới chia 2 chức năng chính của JobTracker - quản lý tài nguyên và quản lý job thành 2 components riêng biệt:

- Resource Manager (RM): quản lý toàn bộ tài nguyên tính toán của cluster.
- Application Master (AM): đơn vị là trên 1 ứng dụng và quản lý vòng đời của Job.

Do vậy đối với YARN, MapReduce sẽ là 1 ứng dụng chạy trên YARN, sử dụng tài nguyên do RM cấp phát. Các node tính toán trong cluster bây giờ sẽ chạy NodeManager quản lý các tiến trình chạy trên máy đó. Resource Manager và Node Manager trở thành xương sống của tính toán phân tán trong YARN. Việc mỗi ứng dụng được tách ra riêng cho phép các process chạy lâu (long running process) cũng có thể được khởi động trên YARN.

ApplicationMaster trên 1 ứng dụng là một thư viện cho phép yêu càu tài nguyên từ Resource Manager và giao tiếp với Node Manager để chạy và thực thi các tasks. Trong YARN, MapReduce2 là thay vì là linh hồn của hadoop như ở hadoop 1 thì chỉ là một ứng dụng. Application Master cho phép xây dựng các ứng dụng khác MR chạy trên YARN.

Hiện tại có rất nhiều ứng dụng BigData được port chạy trên YARN, trong đó có một số ứng dụng nổi tiếng như Spark. Cụ thể bạn có thể tham khảo hình dưới đây:


Applications Run Natively IN Hadoop
BATCH (MapReduce)
INTERACTIVE (Tez)
ONLINE (HBase)
STREAMING (Storm, S4,…)
GRAPH (Giraph)
IN-MEMORY (Spark)
HPC MPI (OpenMPI)
OTHER (Search) (Weave…)
YARN (Cluster Resource Management)
HDFS2 (Redundant, Reliable Storage)
hadoop