BÀI THU HOẠCH

Nội dung : Kiến trúc Hadoop

I. Apache Hadoop

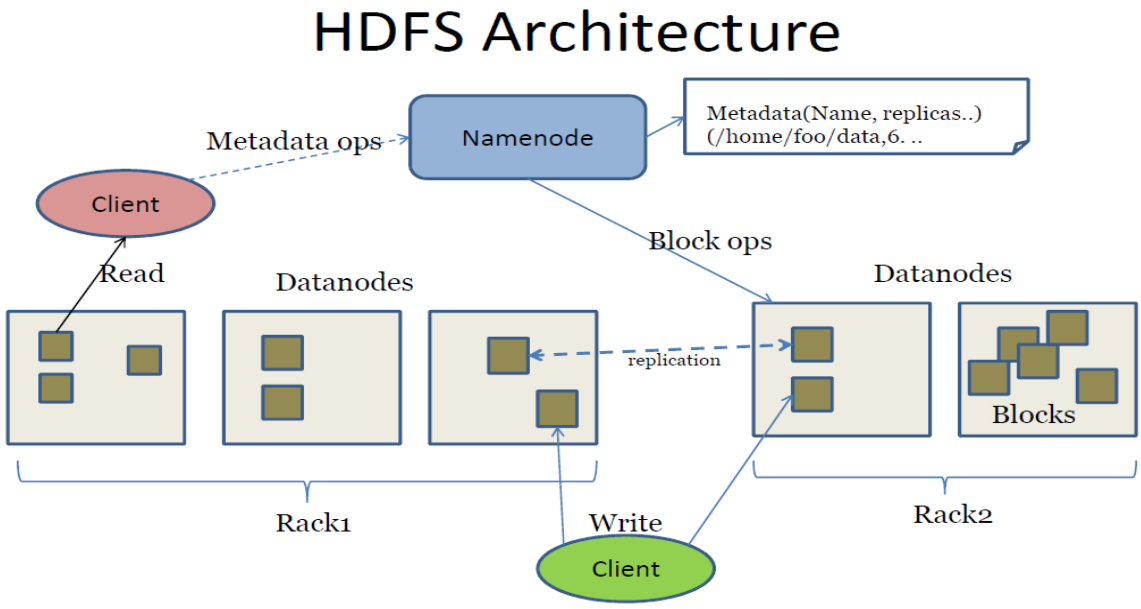
a. Định nghĩa:

- ***Apache Hadoop*** là một Framework hỗ trợ cho việc tối ưu và xử lí dữ liệu phân tán trên nhiều máy.
- ***Apache Hadoop*** được thiết kế để có thể scale được trên nhiều node.

b. Thành phần:

- Các thành phần trong Hadoop:
  - ***Hadoop Common***: Là các thư viện được viết bằng Java, cung cấp các phương thức tiện ích cho các module Hadoop khác.
  - ***HDFS*** (Hadoop Distributed File System): Là thành phần xử lí lưu trữ file phân tán của Hadoop.
  - ***Hadoop Yarn***: Framework dùng để lập lịch job và quản lí tài nguyên hệ thống.
  - ***MapReduce***: Hệ thống xử lí dữ liệu của Hadoop.

II. HDFS (Hadoop Distributed File System)



a. Định nghĩa:

- HDFS là hệ thống file lưu trữ dữ liệu phân tán, có khả năng chịu lỗi cao, deploy trên các phần cứng thông thường, khả năng truy cập throughput, thích hợp với dữ liệu lớn.
- Hệ thống tệp phân tán tuân theo mô hình Master-Slave.

b. Thành phần:

- Gồm 2 thành phần chính:
  - ***NameNode*** (Master Node)
  - ***DataNode*** (Worker Node)
- Chức năng của ***NameNode***:
  - Lưu trữ thông tin metadata (fsimage, edit logs)
  - Quản lí 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.
- Chức năng của ***DataNode***:
  - Cung cấp các block để lưu trữ file.
  - Giao tiếp với Client cho thao tác đọc ghi.
  - Khởi tạo và xóa block data.
  - Replicate data trong cụm.
  - Connect với ***NameNode*** định kì qua các heartbeat.
- ***Block*** trong Dadoop:
  - HDFS chia tệp thành các phần có kích thước khối được gọi là block. Kích thước của block là 128 Mb và có thể lên tới 256Mb.
- ***Rack Awareness***:
  - Làm tăng khả năng trao đổi dữ liệu, dễ dàng cho việc quản lý.
  - Không có quá 1 block trên 1 node, không có quá 2 block trên cùng một rack.
  - Trong một cụm Hadoop lớn, có nhiều racks. Mỗi rack bao gồm các ***DataNodes***. Giao tiếp giữa các ***DataNodes*** trên cùng rack hiệu quả hơn so với giao tiếp giữa các ***DataNodes*** nằm trên racks khác nhau.

III. YARN (Yet Another Resource Negotiator)

a. Thành phần:

- ***Resource Manager***:
  - Có nhiệm vụ quản lý toàn bộ tài nguyên trong cụm, nó cung cấp tài nguyên là core, RAM cho ứng dụng cần dùng, đồng thời cũng là đầu não quản lý các NodeManager.
  - ResourceManager có hai thành phần chính là ***Scheduler*** và ***ApplicationManager***:
    - ***Scheduler***: có nhiệm vụ cung cấp tài nguyên cho các ứng dụng sử dụng, nó là thuần scheduler và không có

nhiệm vụ quản lý hay theo dõi tình trạng của ứng dụng, hoặc như việc restart các task bị fail. Việc cung cấp tài nguyên là dựa trên yêu cầu cảu ứng dụng, các tài nguyên như RAM, core,..

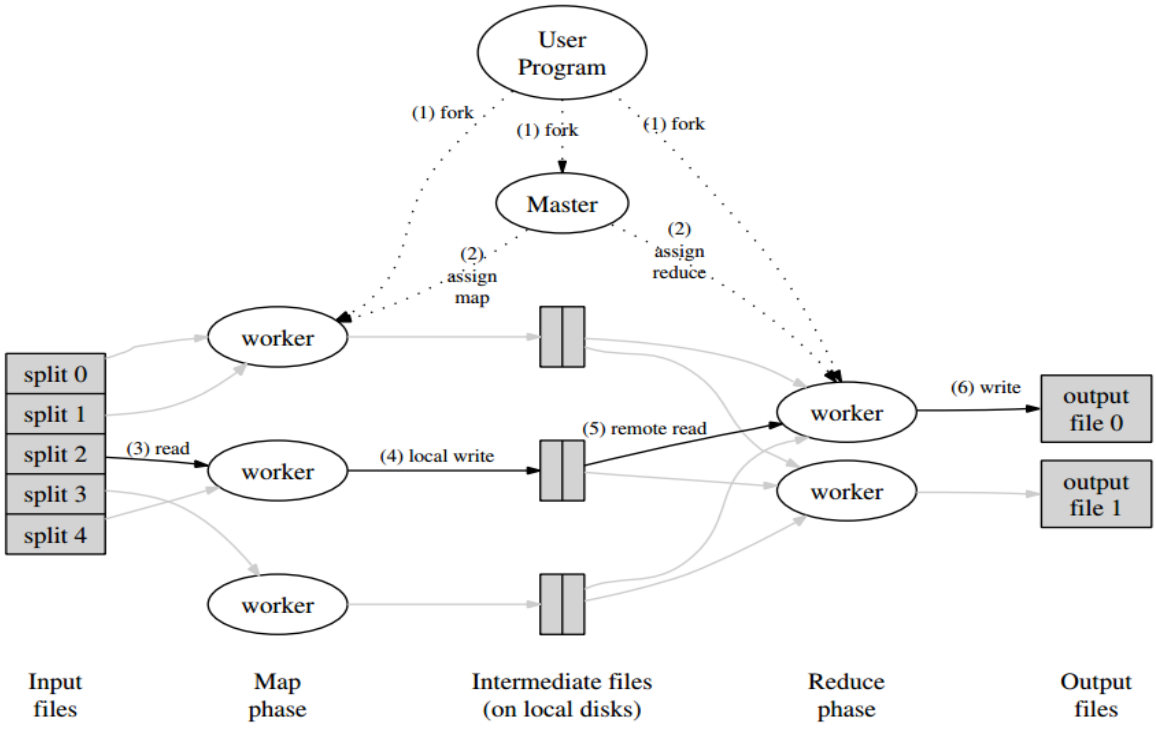
- ***ApplicationManager***: Có nhiệm vụ tiếp nhận các job từ client, cấp phát tài nguyên để start ApplicationMaster cho từng ứng dụng, đồng thời cung cấp dịch vụ restart ApplicationMaster trong trường hợp lỗi.

– ***Node Manager***:
  + Được cài trên 1 node có nhiệm vụ quản lý tài nguyên (RAM, core...) và job trên node đó.
  + NodeManager thường xuyên gửi heartbeat đến RM thông báo tình trạng, mục đích chính của NM là quản lý các container mà RM giao cho nó, kill các container này nếu RM yêu cầu.
– ***Container***:
  + Là tiến trình của YARN, tập hợp các tài nguyên vật lý như core, RAM.
  + YARN container được quản lý bởi đối tượng Container Launch Context, đối tượng này lưu trữ các thông tin như biến môi trường, dependencies, security tokens và các chỉ thị để tạo tiến trình.
  + Container cấp quyền cho các ứng dụng chạy trên nó sử dụng một lượng tài nguyên nhất định.

IV. MapReduce

a. Định nghĩa

– Mapreduce là một mô hình lập trình, thực hiện quá tình xử lý tập dữ liệu lớn. Mapreduce gồm 2 pha : map và reduce.
  + ***Hàm Map*** : Các xử lý một cặp (key, value) để sinh ra một cặp (key_i, $value_i$) - key và value trung gian. Dữ liệu này input vào hàm Reduce.
  + ***Hàm Reduce*** : Tiếp nhận các (key_i, $value_i$) và trộn các cặp (key_i, $value_i$) trung gian , lấy ra các $value_i$ có cùng key_i.



- Thông qua thư viện MapReduce ứng dụng với từng ngôn ngữ, chương trình có nhiệm vụ phân mảnh tệp dữ liệu đầu vào. Dữ liệu vào được chia thành các phần nhỏ 16 megabytes đến 64 megabytes (MB). Sau đó khởi động việc sao chép chương trình trên các clusters.
- Các máy gồm có: master và worker. Trong đó máy master làm nhiệm vụ điều phối sự hoạt động của quá trình thực hiện MapReduce trên các máy worker, các máy worker làm nhiệm vụ thực hiện Map và Reduce với dữ liệu mà nó nhận được. Bằng cách đặt trạng thái idle máy workers và sau đó gắn cho từng máy task map hoặc reduce.
- Máy master sẽ thực hiện phân phối các tác vụ Map và Reduce vào các worker đang rãnh rỗi. Các tác vụ này được master phân phối cho các máy tính dựa trên vị trí của dữ liệu liên quan trong hệ thống. Máy worker khi nhận được tác vụ Map sẽ đọc dữ liệu mà nó nhận từ phân vùng dữ liệu đã gán cho nó và thực hiện hàm Map. Kết quả đầu ra là các cặp (keyI,valueI) trung gian. Các cặp này được lưu tạm trên bộ nhớ đệm của các máy.
- Sau khi thực hiện xong công việc Map, các máy worker làm nhiệm vụ chia các giá trị trung gian thành R vùng (tương ứng với R tác vụ Reduce) lưu xuống đĩa và thông báo kết quả, vị trí lưu cho máy master.

- Master sẽ gán giá trị trung gian và vị trí của các dữ liệu đó cho các máy thực hiện công việc Reduce. Các máy reducer làm nhiệm vụ xử lý sắp xếp các key, thực hiện hàm Reduce và đưa ra kết quả cuối cùng.
- Master sẽ kích hoạt thông báo cho chương trình người dùng quá trình MapReduce đã hoàn tất, kết quả đầu ra được lưu trữ trên R tập tin.