BÀI THU HOẠCH

1. HDFS

* Hadoop Distributed File System (HDFS) là một hệ thống tập tin phân tán cho phép lưu trữ lượng dữ liệu lớn trên nhiều node trong cụm Hadoop
* Nguyên tắc thiết kế của HDFS :

+ Write once, read many times. Chỉ cho phép append giúp hạn chế việc phải đồng bộ nhiều lần

+ Dữ liệu phân tán : File được chia thành các block (thường là 128MB) => giúp giảm kích thước metadata

+ Nhân bản dữ liệu : các block được replicated trên các node khác nhau, các rack khác nhau (rack là cụm server có cùng nguồn, mạng, …)

+ Khả năng chịu lỗi : đối với DataNode có cơ chế chịu lỗi là replication, còn đối với NameNode có các cơ chế chịu lỗi là Secondary NameNode hoặc High Availability NameNode (active, standby)

* Các ưu điểm của HDFS :

+ Lưu trữ phân tán : giúp lưu trữ được lượng dữ liệu lớn mà chỉ cần dùng các phần cứng phổ thông

+ Tính toán phân tán và song song : vì được lưu trữ phân tán nên có thể xử lý, tính toán trên các node một cách song song

+ Chi phí : HDFS có khả năng scale out từ các phần cứng giá rẻ, scale out giúp lưu trữ được dữ liệu lớn và không bị downtime khi thêm node vào cụm

+ Có khả năng lưu trữ dữ liệu lớn và đa dạng

+ Mang code (các job) vào data thay vì mang data vào code, việc này là cần thiết vì dữ liệu trong HDFS là rất lớn, ta không thể load tất cả dữ liệu về để xử lý

+ Có khả năng chịu lỗi, độ tin cậy cao

+ Tính toàn vẹn dữ liệu

* Kiến trúc của HDFS : là một kiểu kiến trúc Master/Slave, trong đó bao gồm :

+ Duy nhất 1 NameNode (active) là master node trong kiến trúc Hadoop HDFS, NameNode có các vai trò chính là :

* Giúp duy trì và quản lý các DataNodes (slave nodes). NameNode sẽ đều đặn nhận được một Heartbeat và một block report từ tất cả DataNode trong cụm để đảm bảo các DataNodes vẫn còn hoạt động
* Ghi lại metadata của tất các các files được lưu trữ trong cụm (bao gồm các thông tin về list file, list block, permissions, …). Có 2 file của NameNode liên kết với metadata :
  + FsImage : chứa state của namespace hệ thống kể từ khi NameNode chạy
  + EditLogs : chứa tất cả sửa đổi được thực hiện đối với hệ thống tệp

+ HDFS có thể gồm nhiều DataNodes nằm trên các nodes có phần cứng phổ thông, có nhiệm vụ lưu trữ dữ liệu lớn được đẩy vào HDFS. DataNodes có các vai trò chính :

* User nhìn thấy các file dữ liệu được lưu trong cây thư mục của HDFS (HDFS tổ chức hệ thống tập tin giống với UNIX) nhưng thực chất dữ liệu được lưu trữ tại các DataNodes
* Xử lý đọc/ghi với dữ liệu mà nó lưu trữ
* Gửi heartbeat để giúp NameNode đảm bảo DataNode còn sống

+ Secondary NameNode : được coi là một checkpoint node, nó có nhiệm vụ đọc tất cả các file hệ thống và metadata của NameNode và ghi vào ổ cứng. Khi NameNode restart thì nó gửi FsImage mới nhất mà nó đã sao chép đến NameNode, giúp NameNode nhanh chóng khôi phục lại trạng thái trước khi restart

1. MAP-REDUCE

* MapReduce là một engine mặc định để xử lý dữ liệu của Hadoop, nó là style xử lý dữ liệu lớn chứ không phải ngôn ngữ lập trình
* MapReduce chia job thành các task độc lập được xử lý trực tiếp trên các node lưu trữ dữ liệu trong cụm, việc “move code to data” như vậy giúp việc xử lý dữ liệu lớn được thực hiện một cách nhanh chóng (không cần phải load tất cả dữ liệu lên để xử lý) và song song
* MapReduce bao gồm các phase :

+ Mapper : Gồm 2 bước chính là splitting và mapping. Dữ liệu được chia thành các phần nhỏ bằng nhau (chunks) tại bước splitting, Hadoop có RecordReader sử dụng TextInputFormat để chuyển các split input thành các cặp key-value, các cặp key-value này được sử dụng làm đầu vào cho mapping, mapping ánh xạ các cặp key-value thành các cặp key-value mới.

+ Shuffle : Gồm 2 bước chính là sorting và merging. Sorting giúp sắp xếp các cặp key-value theo key, sau đó merging nhóm các cặp key-value có key giống nhau lại.

+ Reducer: Gộp value có key giống nhau lại, output là cặp key-value mới được lưu tại HDFS

1. YARN

* Do ở Hadoop v1.0, MapReduce thực hiện cả chức năng xử lý và quản lý tài nguyên, bao gồm 1 Job Tracker có nhiệm vụ cấp phát tài nguyên, điều phối và quản lý quá trình xử lý job và nó chia job thành các task và gán cho các Task Tarcker xử lý. Do chỉ có 1 Job Tracker điều phối dễ dẫn đến hiện tượng bottle-neck khi số lượng task tăng lên nên ở Hadoop v2.0 YARN (Yet Another Resource Negotiator) được giới thiệu để thay thế MapReduce trong việc quản lý, cấp phát tài nguyên.
* YARN có các thành phần chính là :

+ Resource Manager : Chịu trách nhiệm phân công và quản lý tài nguyên giữa các ứng dụng, khi nhận được yêu cầu xử lý nó sẽ chuyển tiếp đến Node manager tương ứng và phân bổ tài nguyên để hoàn thành yêu cầu. Bao gồm 2 thành phần chính:

* Scheduler: Có nhiệm vụ phân bổ tài nguyên cho các ứng dụng khác nhau. Đây là Scheduler thuần túy vì nó không thực hiện theo dõi trạng thái cho ứng dụng. Nó cũng không sắp xếp lại các tác vụ bị lỗi do lỗi phần cứng hoặc phần mềm. Bộ lập lịch phân bổ các tài nguyên dựa trên các yêu cầu của ứng dụng
* Application Manager: Chấp nhận nộp công việc. Đàm phán container đầu tiên để thực thi ApplicationMaster. Một nơi chứa kết hợp các yếu tố như CPU, bộ nhớ, đĩa và mạng Khởi động lại container ApplicationMaster khi không thành công

+ Node Manager : Quản lý mỗi node trên cụm Hadoop và các ứng dụng trên mỗi node. Nó đăng ký với Resource Manager và gửi health status của node. Giám sát tài nguyên sử dụng, quản lý log và kill container dựa trên chỉ dẫn từ Resource Manager

+ Application Master : Application master chịu trách nhiệm thương lượng tài nguyên với người quản lý tài nguyên, theo dõi trạng thái và giám sát tiến độ của một ứng dụng duy nhất. Application master yêu cầu vùng chứa từ trình quản lý nút bằng cách gửi Bối cảnh khởi chạy vùng chứa (CLC) bao gồm mọi thứ mà ứng dụng cần để chạy. Khi ứng dụng được khởi động, nó sẽ gửi báo cáo tình trạng đến người quản lý tài nguyên theo thời gian.

+ Container : Quản lý các tài nguyên vật lý như CPU, RAM, … trên các node