# **SPARK VÀ MAPREDUCE**

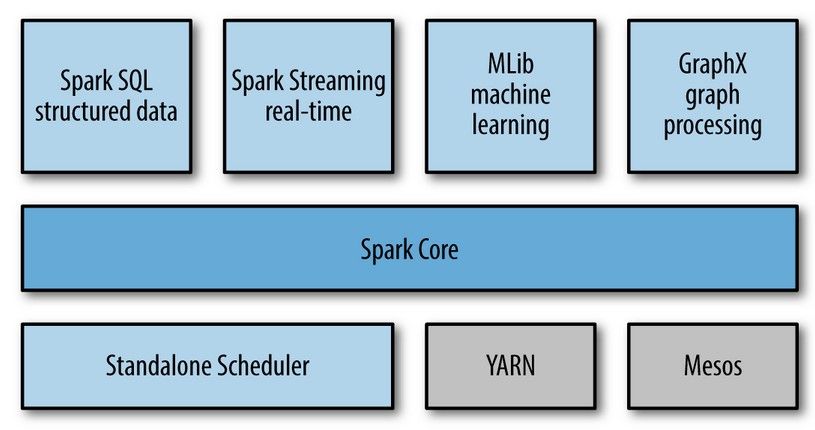
**I/ TỔNG QUAN VỀ SPARK**

**1. Giới thiệu về Apache Spark**

Apache Spark in-memory clusters đang là sự chú ý của nhiều doanh nghiệp trong việc ứng dụng công nghệ vào phân tích và xử lý dữ liệu nhanh chóng.

Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay. Nó cho phép xây dựng các mô hình dự đoán nhanh chóng với việc tính toán được thực hiện trên một nhóm các máy tính, có có thể tính toán cùng lúc trên toàn bộ tập dữ liệu mà không cần phải trích xuất mẫu tính toán thử nghiệm. Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

**2. Thành phần của Apache Spark**



Thành phần trung của Spark là Spark Core: cung cấp những chức năng cơ bản nhất của Spark như lập lịch cho các tác vụ, quản lý bộ nhớ, fault recovery, tương tác với các hệ thống lưu trữ…Đặc biệt, Spark Core cung cấp API để định nghĩa RDD (Resilient Distributed DataSet) là tập hợp của các item được phân tán trên các node của cluster và có thể được xử lý song song.

Spark có thể chạy trên nhiều loại Cluster Managers như Hadoop YARN, Apache Mesos hoặc trên chính cluster manager được cung cấp bởi Spark được gọi là Standalone Scheduler.

* Spark SQL cho phép truy vấn dữ liệu cấu trúc qua các câu lệnh SQL. Spark SQL có thể thao tác với nhiều nguồn dữ liệu như Hive tables, Parquet, và JSON.
* Spark Streaming cung cấp API để dễ dàng xử lý dữ liệu stream,
* MLlib Cung cấp rất nhiều thuật toán của học máy như: classification, regression, clustering, collaborative filtering…
* GraphX là thư viện để xử lý đồ thị.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được ( Spark Streaming).

Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3,…. Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json…) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào.

**3. Những tính năng nổi bật của Spark**

* “Spark as a Service”: Giao diện REST để quản lí (submit, start, stop, xem trạng thái) spark job, spark context
* Tăng tốc, giảm độ trễ thực thi job xuống mức chỉ tính bằng giây bằng cách tạo sẵn spark context cho các job dùng chung.
* Stop job đang chạy bằng cách stop spark context
* Bỏ bước upload gói jar lúc start job làm cho job được start nhanh hơn.
* Cung cấp hai cơ chế chạy job đồng bộ và bất đồng bộ
* Cho phép cache RDD theo tên , tăng tính chia sẻ và sử dụng lại RDD giữa các job
* Hỗ trợ viết spark job bằng cú pháp SQL
* Dễ dàng tích hợp với các công cụ báo cáo như: Business Intelligence, Analytics, Data Integration Tools

**4. Quản lý bộ nhớ của Apache Spark**

Về bộ nhớ, Spark giải quyết các vấn đề vấn đề xung quanh định nghĩa Resilient Distributed Datasets (RDDs). RDDs hỗ trợ hai kiểu thao tác thao tác: transformations và action. Thao tác chuyển đổi(tranformation) tạo ra dataset từ dữ liệu có sẵn. Thao tác actions trả về giá trị cho chương trình điều khiển (driver program) sau khi thực hiện tính toán trên dataset.

Spark thực hiện đưa các thao tác RDD chuyển đổi vào DAG (Directed Acyclic Graph) và bắt đầu thực hiện. Khi một action được gọi trên RDD, Spark sẽ tạo DAG và chuyển cho DAG scheduler. DAG scheduler chia các thao tác thành các nhóm (stage) khác nhau của các task. Mỗi Stage bao gồm các task dựa trên phân vùng của dữ liệu đầu vào có thể pipline với nhau và có thể thực hiện một cách độc lập trên một máy worker. DAG scheduler sắp xếp các thao tác phù hợp với quá trình thực hiện theo thời gian sao cho tối ưu nhất. Ví dụ: các thao tác map sẽ được đưa vào cùng một stage do không xảy ra shuffle dữ liệu giữa các stage. Kết quả cuối cùng của DAG scheduler là một tập các stage. Các Stages được chuyển cho Task Scheduler. Task Scheduler sẽ chạy các task thông qua cluster manager (Spark Standalone/Yarn/Mesos). Task scheduler không biết về sự phụ thuộc của các stages. Nó chỉ chịu trách nhiệm thực hiện sắp xếp các task một cách tối ưu nhất.

Mỗi Worker bao gồm một hoặc nhiều Excuter. Các excuter chịu trách nhiệm thực hiện các task trên các luồng riêng biệt. Việc chia nhỏ các task giúp đem lại hiệu năng cao hơn, giảm thiểu ảnh hưởng của dữ liệu không đối xứng (kích thước các file không đồng đều).

Quá trình Spark xây dựng DAG: Có hai kiểu chuyển đổi có thể áp dụng trên các RDDs đó là chuyển đổi hẹp và chuyển đổi rộng:

* Chuyển đổi hẹp: không yêu cầu xáo trộn dữ liệu vượt qua các phân vùng (partition).Ví dụ như các thao tác map, filter,..
* Chuyển đổi rộng yêu cầu dữ liệu phải xáo trộn. Ví dụ: reduceByKey, sortByKey, groupByKey,…

**Các thành phần chính trong quản lý bộ nhớ:**

Spark truy cập dữ liệu được lưu trữ ở các nguồn khác nhau như: HDFS, Local Disk, RAM. Cache Manager sử dụng Block Manager để quản lý dữ liệu. Cache Manager quản lý dữ liệu nào được Cache trên RAM, thông thường là dữ liệu được sử dụng thường xuyên nhất, nó cũng có thể được xác định thủ công bằng tay sử dụng phương thức persit hoặc cache. Nếu kích thước RAM không đủ chứa dữ liệu thì dữ liệu sẽ được lưu trữ sang Tachyon và cuối cùng là lưu trữ lên đĩa. Khi dữ liệu(RDD) không được lưu trữ trên RAM, khi có nhu cầu sử dụng đến, chúng sẽ được recompute lại. Nó sử dụng một khái niệm là “storage level” để quản lý cấp độ của lưu trữ của dữ liệu.

**I/ TỔNG QUAN VỀ MAPREDUCE**

**Khái niệm :**

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 (keyI, valueI) - key và value trung gian. Dữ liệu này input vào hàm Reduce.

Hàm Reduce : Tiếp nhận các (keyI, valueI) và trộn các cặp (keyI, valueI) trung gian , lấy ra các valueI có cùng keyI.

Việc của lập trình viên là quan tâm tới 2 hàm Map và Reduce. Còn các vấn đề khác như : phân chia các dữ liệu đầu vào, lịch trình thực thi các machines, handling các machines failure, quản lý việc giao tiếp giữa các machines là việc của hệ thống run-time.

=> Lập trình viên có thể không có kinh nghiệm về hệ thống song song và phân tán vẫn dễ dàng vận hành một hệ thống phân tán lớn.

Áp dụng mô hình MapReduce chạy trên lượng lớn các machine cỡ hàng ngàn machine và data lên đến mức Terabytes.

**Sơ đồ :**

