51800538 Lê Hưng Duy

Github: https://github.com/hungduy2609/bigdata.git

SPARK

**Khái niệm:**

Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab.

**Thành phần:**

Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX, trong đó:

Spark Core là nền tảng cho các thành phần còn lại và các thành phần này muốn khởi chạy được thì đều phải thông qua Spark Core do Spark Core đảm nhận vai trò thực hiện công việc tính toán và xử lý trong bộ nhớ (In-memory computing) đồng thời nó cũng tham chiếu các dữ liệu được lưu trữ tại các hệ thống lưu trữ bên ngoài.

Spark SQL cung cấp một kiểu data abstraction mới (SchemaRDD) nhằm hỗ trợ cho cả kiểu dữ liệu có cấu trúc (structured data) và dữ liệu nửa cấu trúc (semi-structured data – thường là dữ liệu dữ liệu có cấu trúc nhưng không đồng nhất và cấu trúc của dữ liệu phụ thuộc vào chính nội dung của dữ liệu ấy). Spark SQL hỗ trợ DSL (Domain-specific language) để thực hiện các thao tác trên DataFrames bằng ngôn ngữ Scala, Java hoặc Python và nó cũng hỗ trợ cả ngôn ngữ SQL với giao diện command-line và ODBC/JDBC server.

Spark Streaming được sử dụng để thực hiện việc phân tích stream bằng việc coi stream là các mini-batches và thực hiệc kỹ thuật RDD transformation đối với các dữ liệu mini-batches này. Qua đó cho phép các đoạn code được viết cho xử lý batch có thể được tận dụng lại vào trong việc xử lý stream, làm cho việc phát triển lambda architecture được dễ dàng hơn. Tuy nhiên điều này lại tạo ra độ trễ trong xử lý dữ liệu (độ trễ chính bằng mini-batch duration) và do đó nhiều chuyên gia cho rằng Spark Streaming không thực sự là công cụ xử lý streaming giống như Storm hoặc Flink.

MLlib (Machine Learning Library): MLlib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ. Theo các so sánh benchmark Spark MLlib nhanh hơn 9 lần so với phiên bản chạy trên Hadoop (Apache Mahout).

GrapX: Grapx là nền tảng xử lý đồ thị dựa trên Spark. Nó cung cấp các Api để diễn tảcác tính toán trong đồ thị bằng cách sử dụng Pregel Api.

**Tính năng nổi bật:**

“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

**Spark Languages**

Scala

Java

Python

R

**MAPREDUCE**

**Khái niệm:**

MapReduce là mô hình được thiết kế độc quyền bởi Google, nó có khả năng lập trình xử lý các tập dữ liệu lớn song song và phân tán thuật toán trên 1 cụm máy tính. MapReduce trở thành một trong những thành ngữ tổng quát hóa trong thời gian gần đây.

MapReduce sẽ bao gồm những thủ tục sau: thủ tục 1 Map() và 1 Reduce(). Thủ tục Map() bao gồm lọc (filter) và phân loại (sort) trên dữ liệu khi thủ tục khi thủ tục Reduce() thực hiện quá trình tổng hợp dữ liệu. Đây là mô hình dựa vào các khái niệm biển đối của bản đồ và reduce những chức năng lập trình theo hướng chức năng. Thư viện của thủ tục Map() và Reduce() sẽ được viết bằng nhiều loại ngôn ngữ khác nhau. Thủ tục được cài đặt miễn phí và được sử dụng phổ biến nhất là là Apache Hadoop.

**Các hàm chính:**

MapReduce có 2 hàm chính là Map() và Reduce(), đây là 2 hàm đã được định nghĩa bởi người dùng và nó cũng chính là 2 giai đoạn liên tiếp trong quá trình xử lý dữ liệu của MapReduce. Nhiệm vụ cụ thể của từng hàm như sau:

Hàm Map(): có nhiệm vụ nhận Input cho các cặp giá trị/ khóa và output chính là tập những cặp giá trị/khóa trung gian. Sau đó, chỉ cần ghi xuống đĩa cứng và tiến hành thông báo cho các hàm Reduce() để trực tiếp nhận dữ liệu.

Hàm Reduce(): có nhiệm vụ tiếp nhận từ khóa trung gian và những giá trị tương ứng với lượng từ khóa đó. Sau đó, tiến hành ghép chúng lại để có thể tạo thành một tập khóa khác nhau. Các cặp khóa/giá trị này thường sẽ thông qua một con trỏ vị trí để đưa vào các hàm reduce. Quá trình này sẽ giúp cho lập trình viên quản lý dễ dàng hơn một lượng danh sách cũng như phân bổ giá trị sao cho phù hợp nhất với bộ nhớ hệ thống.

Ở giữa Map và Reduce thì còn 1 bước trung gian đó chính là Shuffle. Sau khi Map hoàn thành xong công việc của mình thì Shuffle sẽ làm nhiệm vụ chính là thu thập cũng như tổng hợp từ khóa/giá trị trung gian đã được map sinh ra trước đó rồi chuyển qua cho Reduce tiếp tục xử lý.

**Ưu điểm:**

MapReduce có khả năng xử lý dễ dàng mọi bài toán có lượng dữ liệu lớn nhờ khả năng tác vụ phân tích và tính toán phức tạp. Nó có thể xử lý nhanh chóng cho ra kết quả dễ dàng chỉ trong khoảng thời gian ngắn.

Mapreduce có khả năng chạy song song trên các máy có sự phân tán khác nhau. Với khả năng hoạt động độc lập kết hợp phân tán, xử lý các lỗi kỹ thuật để mang lại nhiều hiệu quả cho toàn hệ thống.

MapRedue có khả năng thực hiện trên nhiều nguồn ngôn ngữ lập trình khác nhau như: Java, C/ C++, Python, Perl, Ruby,… tương ứng với nó là những thư viện hỗ trợ.

Như bạn đã biết, mã độc trên internet ngày càng nhiều hơn nên việc xử lý những đoạn mã độc này cũng trở nên rất phức tạp và tốn kém nhiều thời gian. Chính vì vậy, các ứng dụng MapReduce dần hướng đến quan tâm nhiều hơn cho việc phát hiện các mã độc để có thể xử lý chúng. Nhờ vậy, hệ thống mới có thể vận hành trơn tru và được bảo mật nhất.

**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.