## Các câu hỏi liên quan đến báo cáo Apache Spark

1. Sử lý thời gian thực là như thế nào? Lấy ví dụ.

Trả lời:

Xử lý thời gian thực là quá trình xử lý dữ liệu đầu vào liên tục không đổi từ nhiều Nguồn dữ liệu khác nhau với độ trễ rất ngắn.

Ví dụ: khi doanh nghiệp thu thập thông tin về để phân tích thì cần qua các bước ở trung gian để sử lý, biên dịch thông tin thành các dạng dễ phân tích hơn, sử lý, còn đối với sử lý theo thời gian thực thì không cần qua các bước trung gian và sử lý trực tiếp, qua đó giúp các doang nghiệp thu được nhiều dữ liệu hơn, nhanh chóng hơn.

1. So sánh RDD, DataFrame và Spark SQL

Trả lời:

Ở một khía cạnh trừu tượng thì RDD và DataFrame ở 2 layer khác nhau

Layer 1 (core) là những thư viện, api cơ bản để spark có thể chạy trên java, python, scala, r,.. trong đó Spark RDD là cấu trúc dữ lưu trữ thao tác với dữ liệu.

Layer 2 là các thư viện, api cấp cao hơn trong đó có api như Dataframe, Spark Sql, Spark Mlib, …

Có các điểm giống như sau:s

* Đều sử dụng để thao tác dữ liệu

Các điểm khác nhau:

* RDD lưu trữ thao tác các dữ liệu cả có cấu trúc, không có cấu trúc, rdd không có lược đồ
* Spark SQL và DataFrame lưu trữ dữ liệu theo dạng bảng tương tự như các bảng trong cơ sở dữ liệu quan hệ, cả hai api đều có lược đồ (DataFrame còn có thể lưu trữ, thao tác các dữ liệu bán cấu trúc)
* Dữ liệu của RDD, DataFrame được lưu trữ trên ram, còn dữ liệu của SparkSQL được lưu trên ổ cứng

Vd: Khi tạo SparkSession , kết thúc Spark Session thì dữ liệu của Spark DataFrame sẽ biến mất, còn SparkSQL Sẽ không biến mất và có thể tái sử dụng.

Tùy thuộc vào dự án thì ta có thể sử dụng những api khác nhau.

1. Dataset Là gì? Và mối liên hệ với DataFrame?

Trả lời:

Là tập hợp các dữ liệu phân tán, là api được thêm vào Spark để cung cấp các lợi ích của rdd (kiểu dữ liệu mạng cùng với lưu trữ các hàm lamda mạnh mẽ) với các lợi ích công cụ thực thi được tối ưu hóa của Spark SQL

Mối liên hệ DataSet với DataFrame

DataFrame là dataset được tổ chức theo cột được đặt tên về mặt khái niệm, DataFrane tương đương với các bảng trong cơ sở dữ liệu quan hệ hoặc 1 khung trong R hoặc Python nhưng có tối ưu hơn

Trong Java hoặc Scala thì DataFrame được biểu diễn bằng Dataset<Row>

Note Dataset chỉ được hỗ trợ trong Java và Scala

1. Nguyên Lý phục hồi RDD (hay còn gọi là khả năng chịu lỗi của RDD)

Trả lời:

Nguyên lý phục hồi RDD dữ trên khả năng chịu lỗi (fault tolerance) và dòng dõi (linega). Tức khi một RDD bị lỗi khi đang thực hiện các tranform, … thì spark sẽ sử dụng lenega để tái tạo lại dữ liệu thay vì sao lưu lại, nguyên lý này giúp spark sử lý dữ liệu lớn một cách đáng tin cậy và hiệu quả trong môi trường phân tán

Nguyên lý phục hồi RDD

* Lineage (Dòng dõi)

Mỗi RDD lưu trữ thông tin về cách nó được tạo ra từ các RDD khác, thông qua các phép biến đổi (transformations) như map, filter, reduce, v.v. Đây là một dòng dõi (lineage) của RDD.

Lineage là một biểu đồ các phép biến đổi, cho phép Spark tái tạo lại một RDD từ các RDD gốc nếu một phần của nó bị mất do lỗi.

Khi một node hoặc một phân vùng của RDD bị mất, Spark không cần phải đọc lại toàn bộ dữ liệu từ nguồn (như HDFS), thay vào đó nó sẽ sử dụng thông tin lineage để tái tạo lại phân vùng bị mất từ các RDD trước đó.

* Phục hồi bằng cách tái tính toán

Khi một phân vùng của RDD bị mất, Spark không sao lưu dữ liệu đó mà chỉ lưu trữ thông tin về cách dữ liệu được tạo ra. Do đó, Spark có thể tái tạo lại dữ liệu từ đầu bằng cách áp dụng lại các phép biến đổi từ lineage.

Ví dụ: Nếu bạn có một RDD rdd được tạo ra từ một file và áp dụng một phép filter để tạo ra filteredRdd, nếu phần của filteredRdd bị mất, Spark sẽ sử dụng thông tin lineage để lấy lại dữ liệu từ rdd gốc và áp dụng lại phép filter.

* Fault Tolerance (Khả năng chịu lỗi)

Spark đảm bảo tính nhất quán và tính chịu lỗi của RDD bằng cách tái tạo lại dữ liệu bị mất mà không làm gián đoạn toàn bộ chương trình. Điều này giúp Spark xử lý được các lỗi trong môi trường phân tán mà không cần sao lưu đầy đủ dữ liệu.

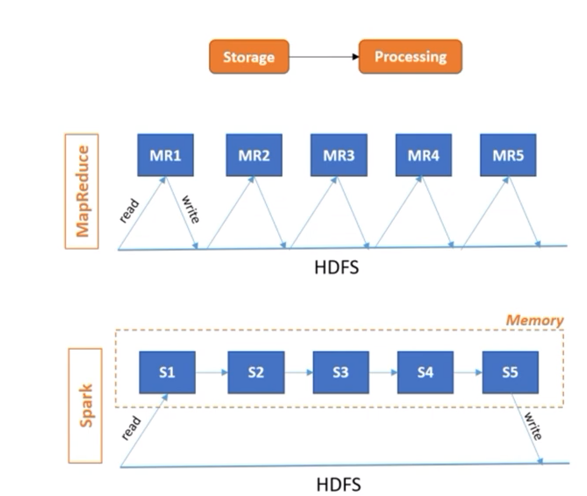
Spark dựa vào Lineage Graph để theo dõi các phép biến đổi được thực hiện trên RDD và tái tạo lại dữ liệu từ các bước biến đổi trước đó.

Caching và Persisting

Để giảm bớt việc phải tái tính toán lại dữ liệu từ đầu trong các lần sử dụng sau, Spark hỗ trợ caching và persisting các RDD.

Khi một RDD được cache hoặc persist, Spark sẽ lưu trữ dữ liệu đã tính toán vào bộ nhớ hoặc đĩa (tùy chọn). Điều này giúp giảm thời gian tính toán và phục hồi khi một phần của RDD bị mất.

1. Tại sao Spark lại nhanh hơn Hadoop



Lấy ví dụ về sử lý dữ liệu trên một tệp HDFS

Với cùng số Job thì mỗi lần thực thi một task thì MapReduce lại đọc và ghi xuống một lần qua đó với ví dụ là 5 job thì sẽ phải mất tổng là 10 lần thực hiện đọc và ghi, có thể thấy là rất tốn tài nguyên.

Với Spark có cùng số Job, mỗi lần thực thi thì Dữ liệu của task sẽ chuyển sang luôn node khác để thực thi, tính toán một lệnh tiếp theo, thực thi hết thì mới đưa tệp xuống phần bộ nhớ

Và một điều làm cho Spark nhanh hơn MapReduce của Hadoop là những phép tính trong spark được thực hiện trên bộ nhớ tạm thời, có tốc độ thực thi rất nhanh.