HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Môn: AN TOÀN ỨNG DỤNG WEB VÀ CƠ SỞ DỮ LIỆU

Nền tảng xử lý song song Apache spark

Giảng viên : Ninh Thị Thu Trang
Nhóm lớp : 03
Nhóm BTL : 14
Tên sinh viên: Nguyễn Văn Khang - B20DCAT102
Hoàng Trung Kiên - B20DCAT098
Nguyễn Trần Minh - B20DCAT126

HÀ NỘI, 2023

Mục lục

I. Khái quát về Apache Spark.

1.Giới thiệu.

Apache Spark là một cluster computing framework mã nguồn mở được phát triển sơ khởi vào năm 2009 bởi AMPLab tại đại học California, Berkeley. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay. Apache Spark được phát triển nhằm tăng tốc khả năng tính toán xử lý của Hadoop.

Spark cho phép xây dựng và phân tích nhanh các mô hình dự đoán. Hơn nữa, nó còn cung cấp khả năng truy xuất toàn bộ dữ liệu cùng lúc, nhờ vậy ta không cần phải lấy mẫu dữ liệu đòi hỏi bởi các ngôn ngữ lập trình như R. Thêm vào đó, Spark còn cung cấp tính năng streaming, được dùng để xây dựng các mô hình real-time bằng cách nạp toàn bộ dữ liệu vào bộ nhớ.

Khi ta có một tác vụ nào đó quá lớn mà không thể xử lý trên một laptop hay một server, Spark cho phép ta phân chia tác vụ này thành những phần dễ quản lý hơn. Sau đó, Spark sẽ chạy các tác vụ này trong bộ nhớ, trên các cluster của nhiều server khác nhau để khai thác tốc độ truy xuất nhanh từ RAM.

Spark sử dụng API Resilient Distributed Dataset (RDD) để xử lý dữ liệu. Spark nhận được nhiều sự hưởng ứng từ cộng đồng Big data trên thế giới do cung cấp khả năng tính toán nhanh và nhiều thư viện hữu ích đi kèm như Spark SQL (với kiểu dữ liệu DataFrames), Spark Streaming, MLlib (machine learning: classification, regression, clustering, collaborative filtering, và dimensionality reduction) và GraphX (tính toán song song trên dữ liệu đồ thị).

2. Tính năng.

Các tính năng nổi bật của Apache Spark:

- Tốc độ: Spark có thể chạy trên cụm Hadoop và có thể chạy nhanh hơn 100 lần khi chạy trên bộ nhớ RAM, và nhanh hơn 10 lần khi chạy trên ổ cứng. Bằng việc giảm số thao tác đọc, ghi trên đĩa cứng. Nó lưu trữ trực tiếp dữ liệu và xử lý trên bộ nhớ.
- Hỗ trợ đa ngôn ngữ: Spark cung cấp các API có sẵn cho các ngôn ngữ: JAVA, Scala hoặc Python. Do đó, Có thể viết các ứng dụng bằng nhiều các ngôn ngữ khác nhau. Thêm nữa, Spark đi kèm 80 truy vấn tương tác mức cao.
- Phân tích nâng cao: Spark không chỉ hỗ trợ Map và Reduce. Nó còn hỗ trợ truy vấn SQL, xử lý theo Stream, học máy và các thuật toán đồ thị - Graph.

II. Kiến trúc của và cơ chế hoạt động Apache spark- Spark Architecture.

1. Spark Eco-System.

Hình 1. Kiến trúc của Spark Eco-System

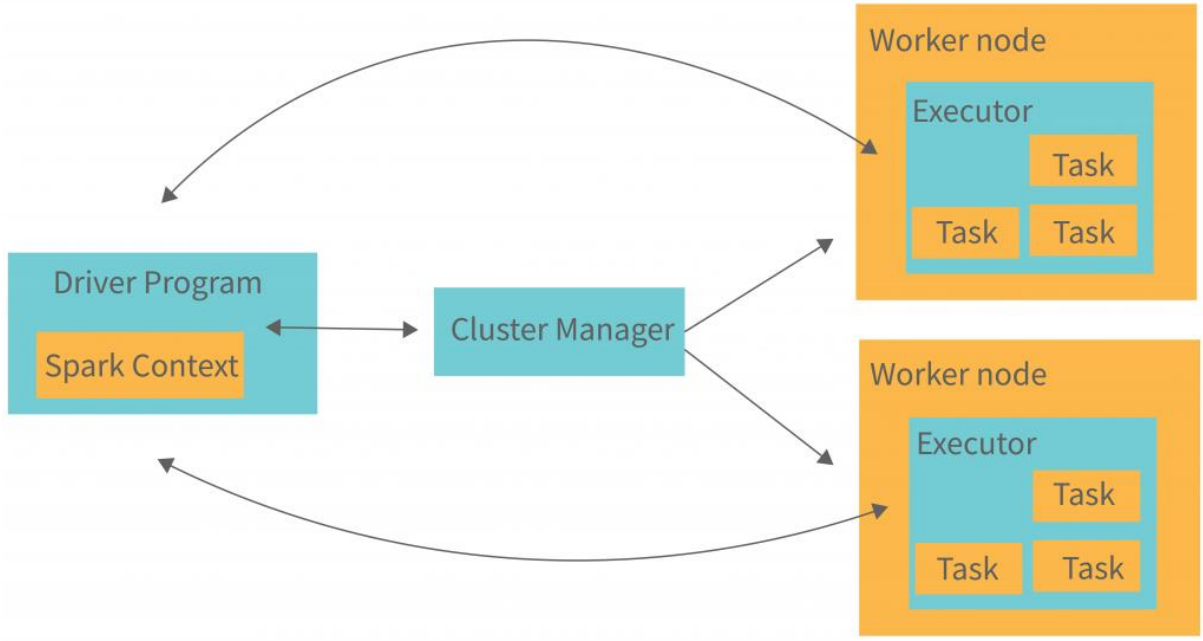
- Spark Core: Là phần cơ sở để xử lý dữ liệu song song và phân tán quy mô lớn. Hơn nữa, các thư viện bổ sung được xây dựng trên trên đây cho phép xử lý công việc đa dạng cho Streaming, SQL, và học máy. Nó chịu trách nhiệm quản lý bộ nhớ và phục hồi lỗi, lập lịch, phân phối và giám sát các công việc trên 1 cluster và tương tác với các hệ thống lưu trữ.
- Spark Streaming: Spark Streaming là một thành phần của Spark được sử dụng để xử lý dữ liệu phát trực tuyến trong thời gian thực. Vì vậy, nó là một API bổ sung hữu ích cho phần lõi của Spark
- Spark SQL: Spark SQL là một mô-đun mới trong Spark tích hợp xử lý quan hệ với API lập trình chức năng của Spark. Nó hỗ trợ dữ liệu truy vấn thông qua SQL hoặc thông qua ngôn ngữ truy vấn Hive. Đối với những người quen thuộc với RDBMS, Spark SQL sẽ là một sự chuyển đổi dễ dàng từ các công cụ trước đây của bạn, nơi bạn có thể mở rộng ranh giới của việc xử lý dữ liệu quan hệ truyền thống.
- GraphX: GraphX là API Spark cho đồ thị và tính toán song song đồ thị.
- MLlib (Machine Learning) Mllib là viết tắt của Thư viện học máy. Spark Mllib được sử dụng để thực hiện học máy trong Apache Spark.

- SparkR: Đây là gói R cung cấp triển khai khung dữ liệu phân tán. Nó cũng hỗ trợ các hoạt động như lựa chọn, lọc, tổng hợp nhưng trên các bộ dữ liệu lớn.

Có thể thấy, Spark được đóng gói với các thư viện cấp cao, bao gồm hỗ trợ cho R, SQL, Python, Scala, Java, v.v ... Các thư viện tiêu chuẩn này làm tăng các tích hợp liền mạch trong một quy trình công việc phức tạp. Về điều này, nó cũng cho phép các bộ dịch vụ khác nhau tích hợp với nó như MLLIB, GraphX, SQL + Frames, Dịch vụ phát trực tuyến, v.v. để tăng khả năng của nó.

2. Kiến trúc của Apache Spark.

Sơ đồ kiến trúc cơ sở của Apache Spark được cung cấp trong hình sau:



Hình 2. Kiến trúc của Apache Spark

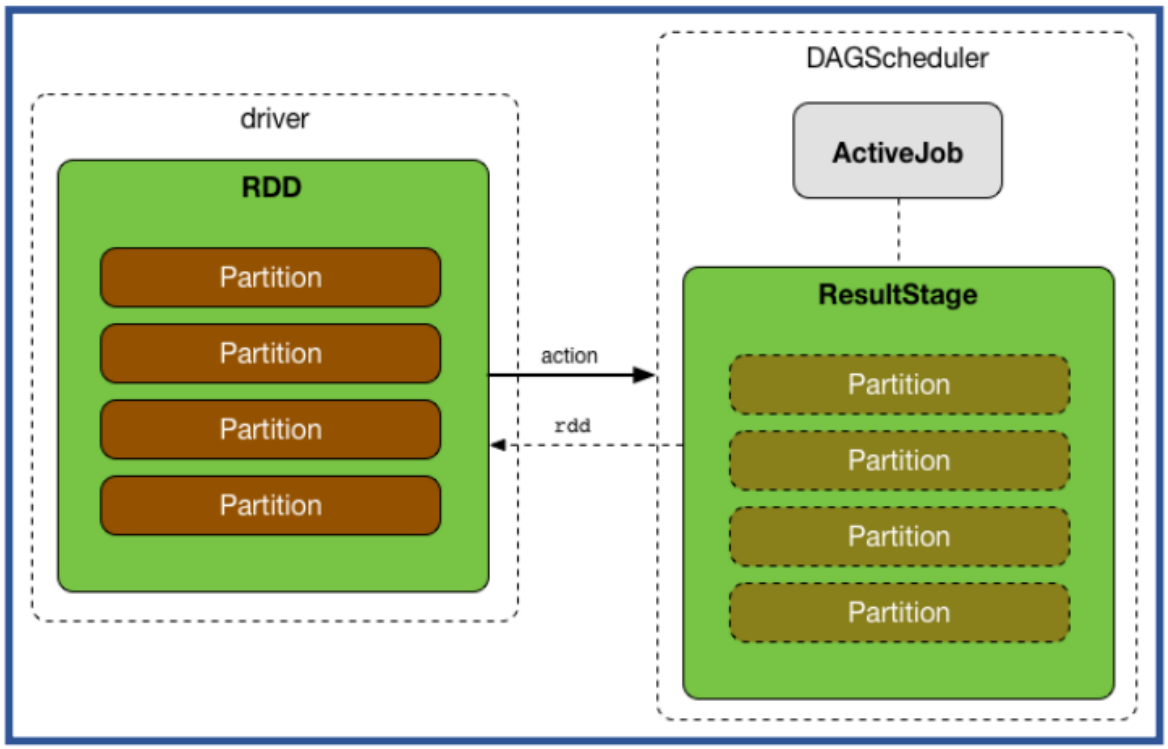
Apache Spark có kiến trúc gồm một node master và nhiều node worker:

- Spark Driver sẽ liên hệ với master node để điều phối các worker node nơi có chứa các executor đang thực thi công việc và giám sát các tác vụ. Trong biểu đồ, Driver gọi ứng dụng chính và tạo một Spark context. Mọi thứ đều được thực thi bởi Spark Context.
- Master node: Chứa trình quản lý cụm (Spark Standalone /YARN /MESOS) các worker node. Chương trình lập lịch sẽ chịu trách nhiệm lập lịch cho các tác vụ và yêu cầu các worker node thực hiện.
- Worker node: Mỗi worker bao gồm một hoặc một số Executor thực hiện việc lưu trữ, đọc ghi khi xử lý dữ liệu.
- Executor: Chịu trách nhiệm xử lý các task nhỏ riêng biệt bằng các luồng độc lập.

3. Quản lý bộ nhớ trong Spark.

Việc quản lý bộ nhớ của Spark trong bộ nhớ dựa trên 2 khái niệm:

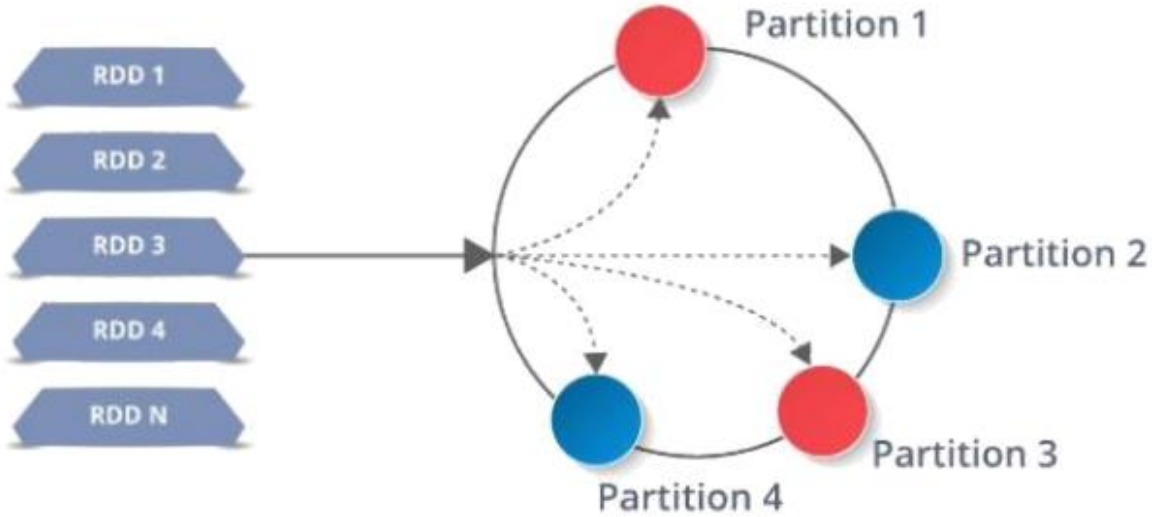
- Resilient Distributed Datasets (RDD) – các tập dữ liệu phân tán.
- Directed Acyclic Graph (DAG) – đồ thị tuần hoàn có hướng.



Hình 3. Quản lý bộ nhớ trong Spark

a, RDD (Resilient Distributed Datasets):

- Resilient: Khả năng phục hồi dữ liệu khi bị lỗi
- Distributed: Dữ liệu được phân phối giữa nhiều nút trong một cụm
- Datasets: Tập hợp dữ liệu được chia vùng theo giá trị



Hình 4. Mô hình của RDD

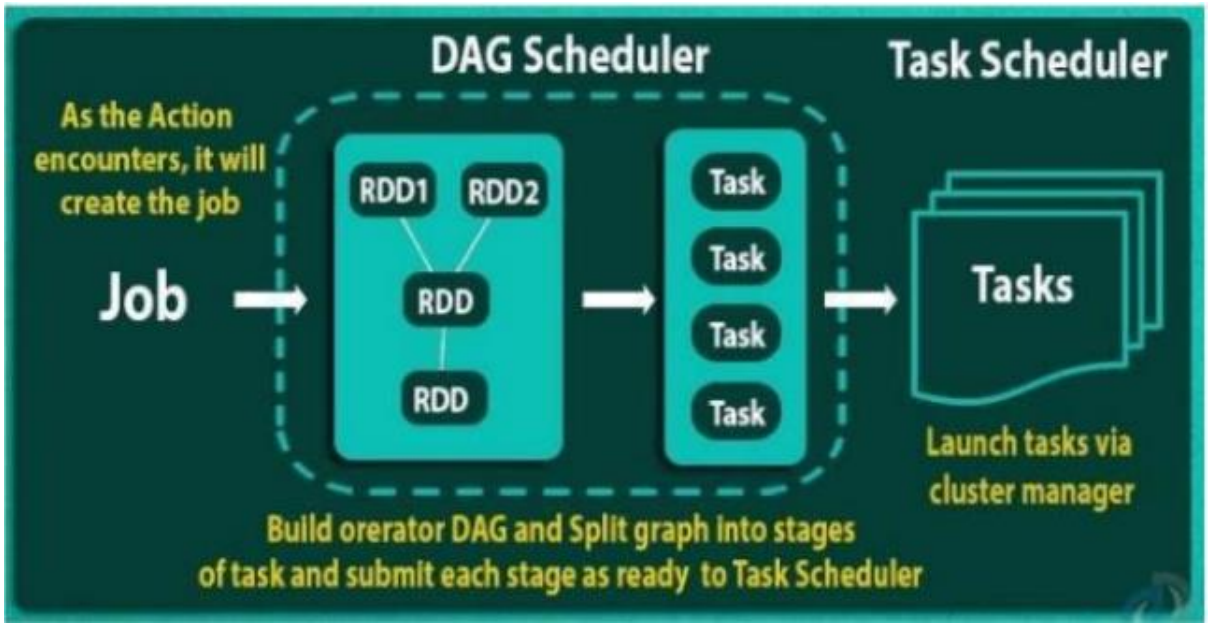
- RDD: Là tập dữ liệu phân tán mà các dữ liệu này được phân tán vào các node của cluster để thực hiện tính toán song song. Lưu trữ trên bộ nhớ phục vụ việc sử dụng lại một cách nhanh chóng. Cung cấp cơ chế cache trên bộ nhớ. Tự động phục hồi dữ liệu khi xảy ra lỗi.
- RDDs hỗ trợ hai kiểu thao tác thao tác: transformation và action
- Transformation (map, filter, groupBy, join…): tạo ra dataset từ dữ liệu có sẵn, nghĩa là biến đổi một hoặc nhiều RDD thành một RDD mới. Tất cả các transformation đều là lazy, có nghĩa là các transformation này sẽ không thực hiện tính toán trong phương thức ngay mà chúng sẽ được lưu lại thành dữ liệu cơ bản (ví dụ như file) và chúng chỉ thực hiện tính toán khi 1 action được gọi để yêu cầu trả về kết quả cho driver program. Nhờ thiết kế này mà Spark chạy hiệu quả hơn
- Action (count, collect, save…): trả về giá trị cho chương trình điều khiển (driver program) sau khi thực hiện tính toán trên dataset.
- Lazy Evaluation:

  Spark sử dụng đánh giá lười biếng (lazy evaluation) để tối ưu hóa việc tính toán. Khi một phép biến đổi RDD được gọi, Sparksẽ không thực hiện ngay lập tức, mà chỉ ghi nhớ phép biến đổi đó trong DAG. Các phép biến đổi chỉ được thực thi khi một hành động (action) được gọi. Đánh giá lười biếng cho phép Spark tối ưu hóa việc tính toán bằng cách tránh tính toán không cần thiết và tối ưu thứ tự thực thi các phép biến đổi.
- Memory Management:

  Apache Spark sử dụng mô hình bộ nhớ chia sẻ (shared memory model) để quản lý dữ liệu. Dữ liệu RDD được lưu trữ trong bộ nhớ và có thể được tái sử dụng trong nhiều phép biến đổi và hành động. Spark sử dụng quản lý bộ nhớ tự động (automatic memory management) để xác định các dữ liệu nào nên được giữ trong bộ nhớ và các dữ liệu nào nên được lưu trữ trên đĩa.

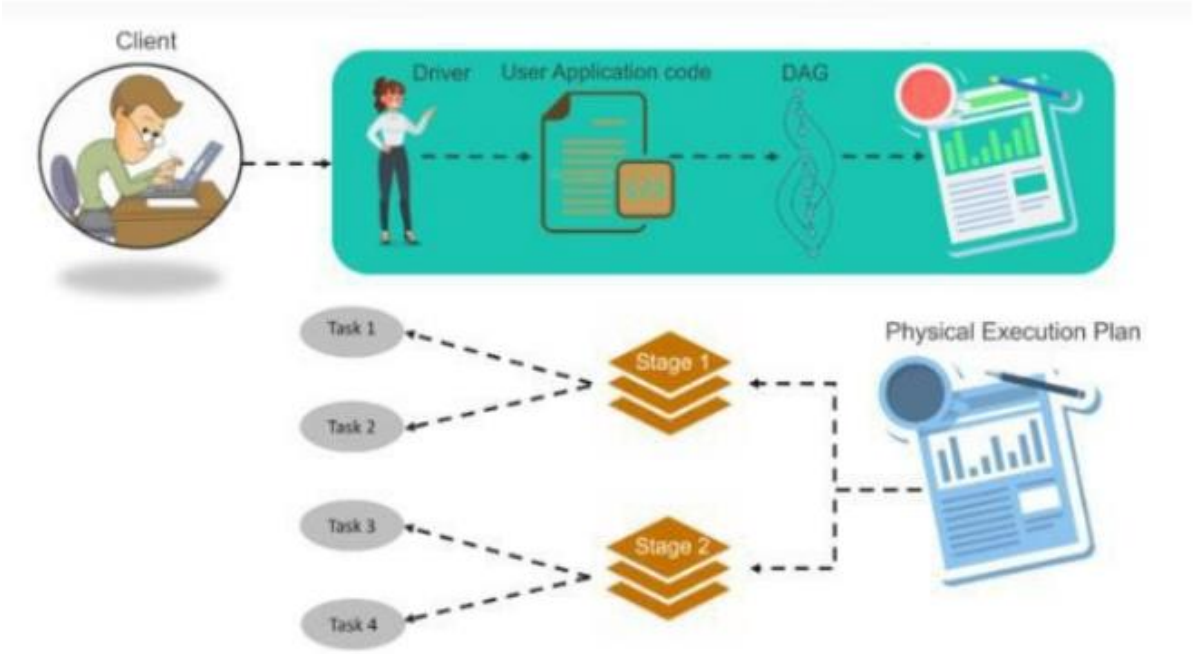
b, DAG (Directed Acyclic Graph):

- RDD được hình thành sau mỗi lần chuyển đổi. Ở cấp độ cao khi áp dụng action trên các RDD này, Spark tạo ra một DAG. DAG là một chuỗi các tính toán được thực hiện trên dữ liệu trong đó mỗi nút là một phân vùng RDD.



Hình 5. Mô hình của DAG

- Chia các stage thành các tasks và thực hiện các phép biến đổi bên trong mỗi stage Phân chia phân vùng hạn chế tối đa việc xáo trộn dữ liệu giữa các node.
- 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 nhóm (stage) bao gồm các task dựa trên phân vùng của dữ liệu đầu vào có thể pipeline 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
- 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).
- 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.



Hình 6. Cách hoạt động của DAG

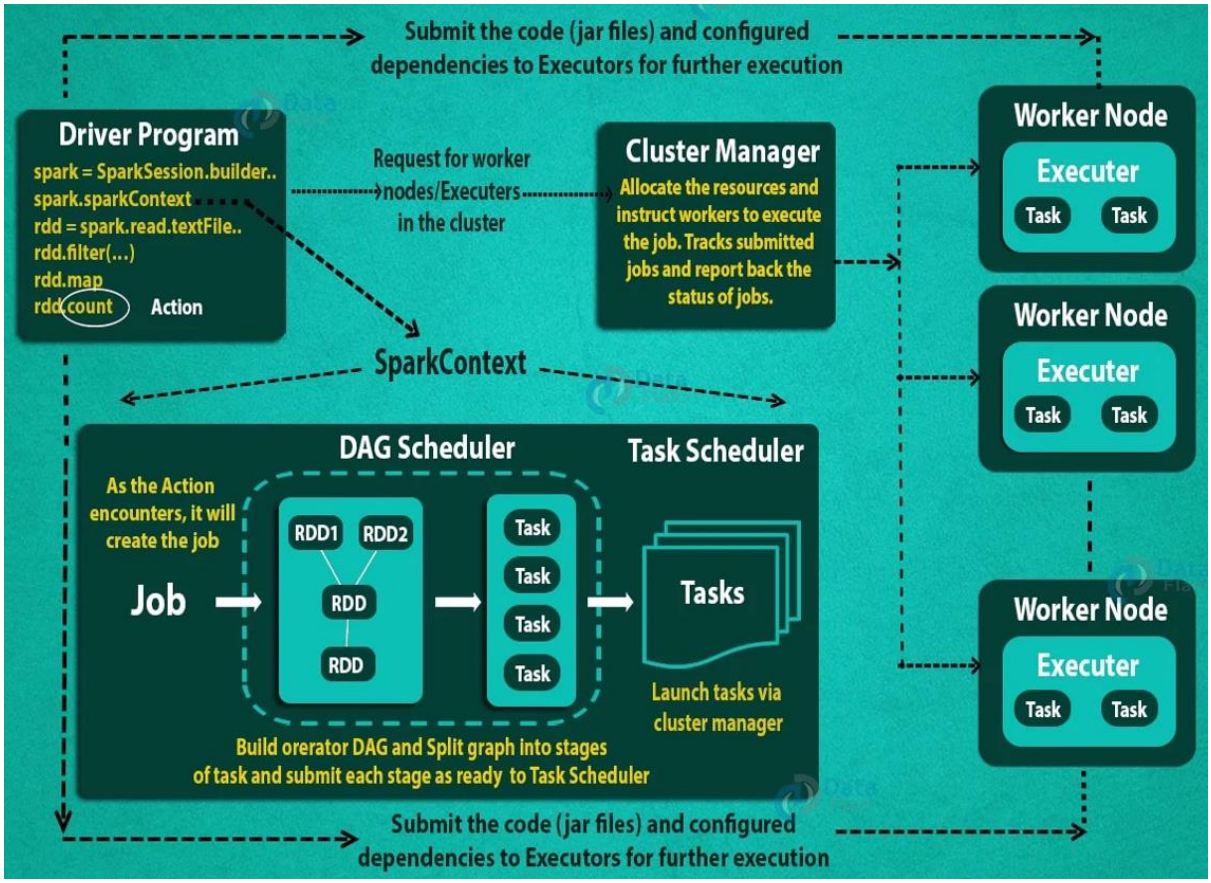
BƯỚC 1: Khách hàng gửi mã ứng dụng người dùng spark. Khi mã ứng dụng được gửi, trình điều khiển sẽ ngầm chuyển đổi mã người dùng chứa các phép biến đổi và hành động thành *biểu đồ tuần hoàn được định hướng* hợp lý được gọi là ***DAG.*** Ở giai đoạn này, nó cũng thực hiện các tối ưu hóa như chuyển đổi đường ống.

BƯỚC 2: Sau đó, nó chuyển đổi biểu đồ logic gọi là DAG thành kế hoạch thực hiện vật lý với nhiều giai đoạn. Sau khi chuyển đổi thành kế hoạch thực thi vật lý, nó sẽ tạo ra các đơn vị thực thi vật lý được gọi là nhiệm vụ trong từng giai đoạn. Sau đó, các nhiệm vụ được nhóm lại và gửi đến cụm.

BƯỚC 3: Bây giờ trình điều khiển nói chuyện với người quản lý cụm và thương lượng các tài nguyên. Trình quản lý cụm khởi chạy các trình thực thi trong các nút công nhân thay mặt cho trình điều khiển. Lúc này, trình điều khiển sẽ gửi nhiệm vụ cho người thực thi dựa trên vị trí dữ liệu. Khi người thực thi bắt đầu, họ tự đăng ký với trình điều khiển. Vì vậy, trình điều khiển sẽ có cái nhìn toàn cảnh về những người thực thi đang thực thi tác vụ.

BƯỚC 4: Trong quá trình thực thi tác vụ, chương trình trình điều khiển sẽ giám sát tập hợp các trình thực thi đang chạy. Nút trình điều khiển cũng lên lịch các tác vụ trong tương lai dựa trên vị trí dữ liệu.

4. Hoạt động của Apache Spark.



Hình 7. Quy trình hoạt động của Apache Spark

- Khi một client gửi ứng dụng của người dùng spark, driver ngầm chuyển đổi code chứa các phép transformations và actions thành một DAG. Ở giai đoạn này, DAG được chuyển cho DAG scheduler. DAG scheduler phân chia đồ thị thành các nhóm (stage) và gửi từng nhóm khi sẵn sàng lên Task Scheduler.
- Sau khi đã tạo ra một tập các stages, nó tạo ra các đơn vị thực thi nhỏ được gọi là các task theo từng stage. Sau đó, các task được nhóm lại để được gửi tới cluster manager.
- Driver program sau đó sẽ giao tiếp với cluster manager và đàm phán các nguồn lực. Sau đó, cluster manager sẽ khởi chạy các executor trên các worker nodes thay cho driver. Tại thời điểm này driver gửi tác vụ (tasks) đến cluster manager dựa trên vị trí dữ liệu.
- Trước khi các executor bắt đầu thực hiện, chúng tự xác nhận với driver program để driver có nắm được tổng thể về tất cả các executor. Bây giờ các executors bắt đầu thực hiện các task khác nhau do driver program gán.
- Tại bất kỳ thời điểm nào khi ứng dụng spark đang chạy, driver program sẽ theo dõi tập hợp các executor chạy. Driver Program trong kiến trúc Spark cũng lên lịch các tasks trong tương lai dựa trên vị trí dữ liệu bằng cách theo dõi vị trí của dữ liệu được

lưu trong bộ nhớ cache. Khi phương thức main () của driver program thoát hoặc khi nó gọi phương thức stop () của Spark Context.

5. Ưu điểm và nhược điểm của Apache Spark.

a, Ưu điểm

- Có thể xử lý cùng môt bộ dữ liệu nhanh hơn đáng kể.
- Phân tích nâng cao
- Dễ sử dụng: Apache Spark có API dễ sử dụng, hoạt động trên các tập dữ liệu lớn
- Đa ngôn ngữ: Spark cung cấp API cho nhiều ngôn ngữ lập trình như Scala, Java, Python và R
- Tích hợp dễ dàng: Spark có tích hợp tốt với các công cụ và hệ thống khác như Hadoop, Hive, HBase, Cassandra, và nhiều hệ thống lưu trữ và xử lý dữ liệu khác.

b, Nhược điểm.

- Spark Streaming có thể không tích hợp để xử lý khi bắt buộc phải có độ trế thấp.
- Khả năng xử lý dữ liệu nhỏ: Spark là một công cụ phân tán được tối ưu hóa cho xử lý dữ liệu lớn. Vì vậy, khi xử lý các tập dữ liệu nhỏ, Spark có thể gặp hiện tượng tốn thời gian khởi động và tốn tài nguyên hơn so với các công cụ xử lý dữ liệu nhỏ hơn.
- Spark yêu cầu sử dụng nhiều tài nguyên: Spark yêu cầu tài nguyên phần cứng khá lớn, bao gồm bộ nhớ RAM và dung lượng đĩa, để xử lý và lưu trữ dữ liệu trong bộ nhớ.
- Không có quy trình tối ưu hóa tự động
- Không có hệ thống quản lý tệp riêng của nó.

III. Ứng dụng của Apache Spark.

1. Các ứng dụng phổ biến của Apache Spark:

- Phân tích dữ liệu: Spark được sử dụng để phân tích dữ liệu lớn từ các nguồn khác nhau, chẳng hạn như dữ liệu bán hàng, dữ liệu khách hàng và dữ liệu mạng xã hội.
- Học máy: Spark được sử dụng để đào tạo và triển khai các mô hình học máy trên dữ liệu lớn.
- Xử lý dữ liệu thời gian thực: Spark có thể được sử dụng để xử lý dữ liệu thời gian thực từ các nguồn khác nhau, chẳng hạn như dữ liệu phát trực tuyến và dữ liệu cảm biến.

2. Một số ví dụ cụ thể về việc sử dụng Apache Spark:

- Netflix sử dụng Spark để phân tích dữ liệu phát trực tuyến của mình để cải thiện trải nghiệm của người dùng.
- Facebook sử dụng Spark để phân tích dữ liệu người dùng của mình để cải thiện quảng cáo và các sản phẩm khác.
- Amazon sử dụng Spark để xử lý dữ liệu bán hàng của mình để cải thiện hiệu quả hoạt động.

IV. Demo.

1. Cài đặt môi trường.

- Cài đặt Scala:

```
└─$ scala -version
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
```

```
└─$ sudo apt-get install scala

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

- Cài đặt Apache Spark: Tải xuống file zip và giải nén thông qua url "https://www.apache.org/dyn/closer.lua/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz"

```
└─$ ls -all
total 391036
drwxr-xr-x   3 z3r0day z3r0day      4096 Oct 31 17:28 .
drwxr-xr-x 109 z3r0day z3r0day     12288 Oct 31 17:27 ..
drwxr-xr-x  13 z3r0day z3r0day      4096 Sep  9 09:08 spark
-rw-r--r--   1 z3r0day z3r0day 400395283 Oct 31 16:43 spark-3.5.0-bin-hadoop3.tgz
```

- Chuẩn bị nội dung cho file text cho chương trình "WordCount":

```
└─$ cat > input.txt
In as name to here them deny wise this. As rapid woody my he me which. Men but they fail shew just wish next put. Led a
ll visitor musical calling nor her. Within coming figure sex things are. Pretended concluded did repulsive education sm
allness yet yet described. Had countryman his pressed shewing. No gate dare rose he. Eyes year if miss he as upon.
^C
```

- Chuẩn bị mã nguồn cho chương trình "WordCount":

```
1 import org.apache.spark.SparkContext
2 import org.apache.spark.SparkContext._
3 import org.apache.spark._
4
5 object SparkWordCount {
6    def main(args: Array[String]) {
7
8     val sc = new SparkContext( "local", "Word Count", "/usr/local/spark", Nil, Map(), Map())
9       val input = sc.textFile("input.txt")
10      Val count = input.flatMap(line ⇒ line.split(" "))
11      .map(word ⇒ (word, 1))
12      .reduceByKey(_ + _)
13      count.saveAsTextFile("outfile")
14      System.out.println("OK");
15   }
16 }
```

2. Xác minh cài đặt Apache Spark.

a, Thiết lập các biến môi trường

- Thiết lập biến môi trường: + pwd

```
└─$ pwd
/home/z3r0day/Desktop/ApacheSpark
```

=>Lấy về đường dẫn thư mục hiện tại của Folder chứa file giải nén của Spark

+ export PATH = $PATH:/home/z3r0day/Desktop/ApacheSpark/spark/bin

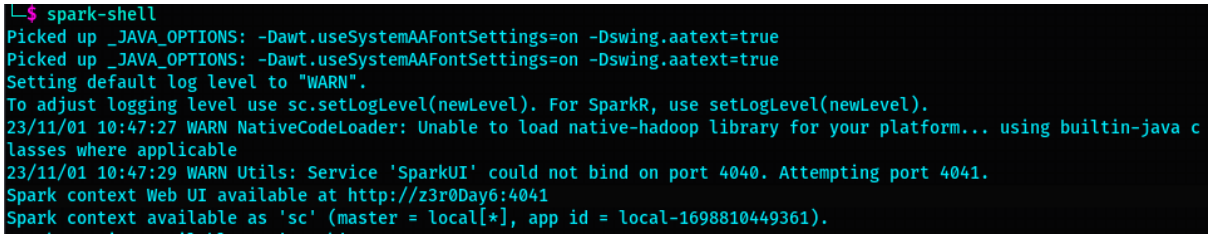
```
└─$ export PATH=$PATH:/home/z3r0day/Desktop/ApacheSpark/spark/bin
```

=>Đặt biến PATH sẽ định vị các tệp thực thi Spark ở vị trí spark/bin. Kiểm tra lại giá trị của biến môi trường:

=>Đường dẫn tới các file thực thi của Spark được gán thành công

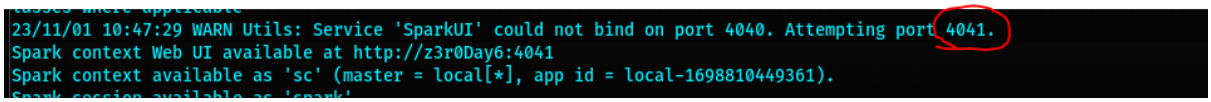
b, Khởi chạy Spark-shell

- Khởi chạy Spark-shell: + spark-shell

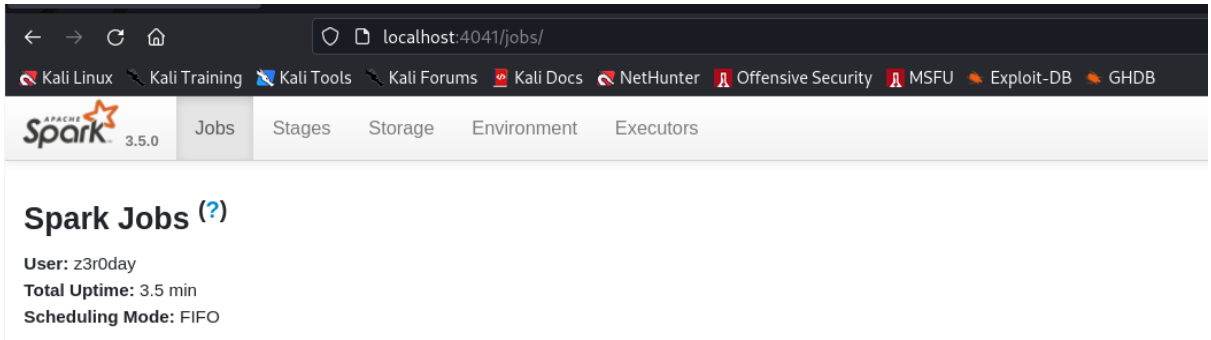


=>Kết quả: khởi chạy thành công Spark-shell

-Theo như kết quả trên, khi khởi chạy CLI của spark-shell, công cụ này sẽ đồng thời chạy SparkUI trên port 4041 local:



=>Kết quả: khi ta truy cập vào SparkUI theo đường dẫn "http://localhost:4041/jobs/" ta sẽ được chuyển hướng tới giao diện đồ họa của Apache Spark:



3. Giải thích mã nguồn.

- Dưới đây là mã nguồn của chương trình "WordCount" trong Apache Spark:

```scala
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark._

object SparkWordCount {
   def main(args: Array[String]) {

    val sc = new SparkContext( "local", "Word Count", "/usr/local/spark", Nil, Map(), Map())
      val input = sc.textFile("input.txt")
      Val count = input.flatMap(line ⇒ line.split(" "))
      .map(word ⇒ (word, 1))
      .reduceByKey(_ + _)
      count.saveAsTextFile("outfile")
      System.out.println("OK");
   }
}
```

=>Trong đó ý nghĩa của các thành phần là:

+ import org.apache.spark.SparkContext
  import org.apache.spark.SparkContext.
  import org.apache.spark.

=> nhập các lớp spark cần thiết vào chương trình Spark.

+ val sc = new SparkContext ("local", "Word Count", "/usr/local/spark", Nil, Map (), Map ()).

+ “Word Count”: Đây là tên của ứng dụng mà bạn muốn chạy “local”.

+ Tham số này biểu thị URL chính để kết nối ứng dụng spark với.

+ “/usr/local/spark”: tham số này biểu thị thư mục chính của Apache Spark.“Map ()” đầu tiên chỉ định môi trường trong khi “Map()” thứ hai chỉ định các biến cho các nút làm việc.

+ input = sc.textFile("input.txt")

=>Bước tiếp theo tạo Spark RDD đầu vào để đọc tệp văn bản input.txt bằng cách sử dụng SparkContext được tạo ở bước trước.

+Val count = input.flatMap (line ⇒ line. Split (" "))
  .map (word ⇒ (word, 1))
  .reduceByKey (_ + _)

=>Trong đoạn mã trên, “flatMap()” được sử dụng để mã hóa các dòng từ tệp văn bản đầu vào thành từ.Phương thức “Map()” đếm tần số của mỗi từ.Phương thức “reduceByKey()” đếm số lần lặp lại của từ trong tệp văn bản.

4.Vận hành triển khai cụm Spark theo cách thủ công.

- Quay trở lại Spark-shell, ta có thể chạy trực tiếp chương trình “WordCount” như đã giải thích trước đó trên cli của công cụ này:

+ val inputfile = sc.textFile ("input.txt")

```
scala> val inputfile = sc.textFile ("input.txt")
inputfile: org.apache.spark.rdd.RDD[String] = input.txt MapPartitionsRDD[1] at textFile at <console>:23
```

=>thông báo này chỉ ra rằng bạn đã thành công trong việc tạo một RDD từ tệp "input.txt" và lưu trữ nó trong biến inputfile. Bạn có thể sử dụng inputfile cho các phép biến đổi và hành động khác trên dữ liệu trong RDD này.

+ val counts = inputfile. flatMap (line => line.split (" ")).map (word => (word, 1)).reduceByKey (_+_)

```
scala> val counts = inputfile. flatMap (line => line.split (" ")).map (word => (word, 1)).reduceByKey (_+_)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:23
```

=>Thông báo này chỉ ra rằng bạn đã thành công trong việc tạo một RDD sau khi thực hiện phép biến đổi reduceByKey, và RDD này chứa các cặp từ (String) và số lần xuất hiện của chúng (Int) sau phép biến đổi này. Bạn có thể sử dụng biến counts cho các phép biến đổi và hành động khác trên dữ liệu trong RDD này.

+ counts.saveAsTextFile ("output")

```
scala> val counts = inputfile. flatMap (line => line.split (" ")).map (word => (word, 1)).reduceByKey (_+_)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:23
```

=>Dòng lệnh sau có nhiệm vụ lưu dữ liệu từ RDD counts vào một thư mục hoặc vị trí cụ thể trong hệ thống tệp. Khi bạn chạy dòng lệnh này, dữ liệu trong RDD counts sẽ được lưu thành các tệp văn bản trong thư mục "output" trên hệ thống tệp của bạn. Mỗi phần tử của RDD sẽ được ghi thành một dòng trong các tệp văn bản này. Điều này thường được sử dụng để lưu kết quả của xử lý dữ liệu phân tán thành tệp văn bản có thể sử dụng hoặc xem xét sau này.

5.Kết quả.

- Sử dụng lệnh cat, in nội dung của tệp đầu ra để tìm sự xuất hiện của từng từ trong tệp input.txt:

```
└─$ cat output/part-00000
(fail,1)
(his,1)
(miss,1)
(education,1)
(calling,1)
(pressed,1)
(rapid,1)
(musical,1)
(countryman,1)
(deny,1)
(this.,1)
(are.,1)
(here,1)
(As,1)
(sex,1)
(woody,1)
(as,2)
(dare,1)
(visitor,1)
(figure,1)
(wise,1)
(they,1)
(just,1)
(my,1)
(Eyes,1)
(them,1)
(upon.,1)
(yet,2)
(smallness,1)
(me,1)
(Men,1)
```

=>Như vậy ta đã thành công trong việc thực hiện chương trình “WordCount” thông qua Apache Spark và hiển thị được số lần lặp lại của các từ trong tệp văn bản cung cấp thành công

6.SparkUI.

a, Khái niệm

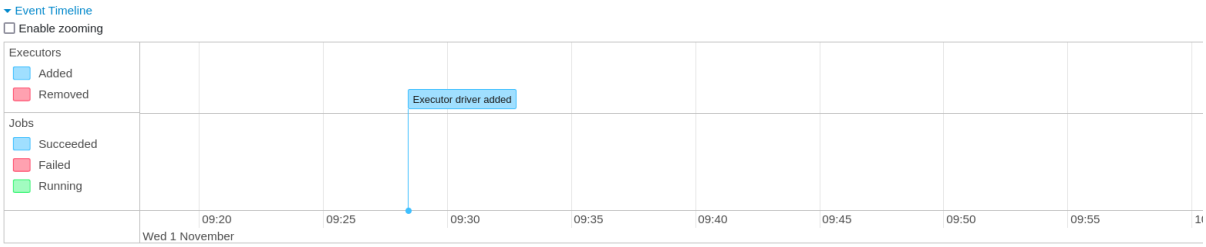
- SparkUI là một giao diện người dùng web dành cho ứng dụng và môi trường tính toán Spark. Nó cung cấp thông tin, giám sát và điều khiển cho các ứng dụng Spark chạy trong môi trường phân tán. Đây là một số điểm quan trọng về SparkUI:
  + Giao diện người dùng web: SparkUI là một trang web mà bạn có thể truy cập thông qua trình duyệt để theo dõi và quản lý việc thực thi ứng dụng Spark.
  + Thông tin giám sát: SparkUI cung cấp thông tin chi tiết về tiến trình chạy, thông số kỹ thuật và hiệu suất của ứng dụng Spark, bao gồm thời gian thực thi, số lượng phân vùng, tài nguyên được sử dụng, lỗi, và nhiều thông tin khác.
  + Trực quan hóa dữ liệu: Giao diện người dùng này thường đi kèm với biểu đồ và đồ thị cho phép bạn theo dõi sự tiến triển của công việc tính toán và tài nguyên được sử dụng.
  + Quản lý công việc: SparkUI cho phép bạn quản lý và theo dõi tiến trình công việc Spark, bao gồm việc tạo, quản lý và hủy bỏ các công việc.
  + Tham số cấu hình: Bạn có thể kiểm tra các tham số cấu hình của ứng dụng Spark thông qua SparkUI.
  + Liên kết: SparkUI thường có một liên kết đến nó trong quá trình chạy ứng dụng Spark. Ví dụ: "http://your-spark-cluster:4040" là URL thường được sử dụng để truy cập SparkUI.

b, Event Timeline

- Trường "Event Timeline" trên SparkUI là một biểu đồ hoặc đồ thị thời gian sự kiện (event timeline) được hiển thị trên giao diện người dùng web của ứng dụng Spark. Nó cho phép bạn theo dõi và hiểu rõ cách mà các sự kiện quan trọng diễn ra trong quá trình thực thi của ứng dụng Spark.Cụ thể, trong trường "Event Timeline," bạn sẽ thấy một biểu đồ thời gian thể hiện các sự kiện quan trọng trong quá trình thực thi ứng dụng Spark. Các sự kiện này có thể bao gồm:
  + Bắt đầu và kết thúc công việc: Biểu đồ sẽ hiển thị thời điểm bắt đầu và kết thúc của các công việc Spark.
  + Thời gian xử lý: Thời gian mà mỗi công việc hoặc phân vùng mất để thực hiện.
  + Sự kiện lỗi: Các sự kiện lỗi hoặc ngoại lệ xuất hiện trong quá trình thực thi.
  + Thông tin về tải tài nguyên: Các sự kiện liên quan đến việc tải dữ liệu từ lưu trữ hoặc phân phối tài nguyên.

+ Các sự kiện khác liên quan đến quá trình thực thi: Ví dụ, thông báo về tiến độ công việc hoặc sự kiện quan trọng khác.

=> Kết quả dựa trên việc chạy chương trình “WordCount” sẽ được hiển thị như sau:



=>Trong đó:

+ “Executor driver added” sẽ đại diện cho chương trình “WordCount” được chạy trên Apache Spark

+ Khoảng thời gian bắt đầu thực hiện chương trình nằm trong khoảng 9h25 đến 9h30

+ Không có sự kiện lỗi xảy ra

+ Cả 2 trường “Executor” và “Jobs” đều có nền xanh dương -> điều đó có nghĩa là chương trình trong quá trình chạy không gặp lỗi

-Trường "Completed Jobs" trên SparkUI là một phần của giao diện người dùng web của ứng dụng Spark. Nó hiển thị thông tin về các công việc (jobs) đã hoàn thành thành công trong quá trình thực thi của ứng dụng Spark. Các công việc là các đơn vị chính để thực hiện xử lý trong Spark và thường được tổ chức thành các tác vụ (tasks) cụ thể để thực thi trên các phân vùng dữ liệu.

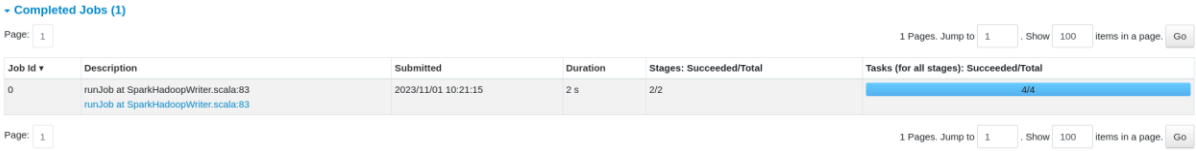
c, Completed Jobs

- Trường "Completed Jobs" trên SparkUI là một phần của giao diện người dùng web của ứng dụng Spark. Nó hiển thị thông tin về các công việc (jobs) đã hoàn thành thành công trong quá trình thực thi của ứng dụng Spark. Các công việc là các đơn vị chính để thực hiện xử lý trong Spark và thường được tổ chức thành các tác vụ (tasks) cụ thể để thực thi trên các phân vùng dữ liệu.
- Thông qua trường "Completed Jobs," bạn có thể xem thông tin về các công việc đã hoàn thành, bao gồm:
  + ID công việc: Mỗi công việc sẽ có một ID duy nhất để xác định nó.
  + Thời gian bắt đầu và kết thúc: Biết được khi nào công việc bắt đầu và kết thúc thực hiện.
  + Thời gian thực thi: Thời gian mà công việc mất để hoàn thành.
  + Số lượng tác vụ (tasks): Một công việc có thể được chia thành nhiều tác vụ để thực thi song song trên các phân vùng dữ liệu.

+ Tài nguyên sử dụng: Thông tin về tài nguyên (CPU, bộ nhớ) được sử dụng bởi công việc.

+ Thông tin về lỗi (nếu có): Nếu công việc gặp lỗi trong quá trình thực thi, bạn có thể tìm hiểu chi tiết về lỗi này.

=> Kết quả dựa trên việc chạy chương trình “WordCount” sẽ được hiển thị như sau:



=>Trong đó:

+ “Job id” của chương trình “WordCount” sẽ là: 1

+ Thời gian bắt đầu sẽ là vào lúc 10h21 phút

+ Thời gian thực hiện chương trình “WordCount” sẽ là 2.5 giây

+ Số lượng tác vụ là 2 và đều chạy thành công

TÀI LIỆU THAM KHẢO

Apache Spark Architecture – Spark Cluster Architecture Explained. (2023, 6). Retrieved from https://www.edureka.co/blog/spark-architecture/

Apache Spark Tutorial. (n.d.). Retrieved from tutorialspoint.com: https://www.tutorialspoint.com/apache_spark/index.htm

Apache Spark Tutorial. (n.d.). Retrieved from https://spark.apache.org/docs/latest/quick-start.html: https://spark.apache.org/

Tìm hiểu về Apache Spark. (2022, 3). Retrieved from Viblo: https://viblo.asia/p/tim-hieu-ve-apache-spark-ByEZkQQW5Q0

Tổng quan về Apache Spark. (2022, 6). Retrieved from bkhost: https://bkhost.vn/blog/apache-spark/