HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO

MÔN HỌC: AN TOÀN HỆ ĐIỀU HÀNH

ĐỀ TÀI: TÌM HIỂU NỀN TẢNG TÍNH TOÁN PHÂN TÁN APACHE SPARK

Giáo viên hướng dẫn: Hoàng Xuân Dậu

Lớp môn học: D19AT03

Nhóm thực hiện: G6

Thành viên thực hiện:

1. Đặng Thế Long – B19DCAT113
2. Phạm Xuân Long – B19DCAT117
3. Nguyễn Thị Quỳnh Mai – B19DCAT121
4. Nguyễn Công Mạnh – B19DCAT123
5. Nguyễn Văn Mạnh – B19DCAT124
6. Nguyễn Văn Nam – B19DCAT129
7. Trần Thanh Nhàn – B19DCAT130

Hà Nội, Tháng 03/2022

PHÂN CÔNG CÔNG VIỆC

Nhóm G6

STT	Mã SV	Họ và tên	Công việc
1	B19DCAT113	Đặng Thế Long	Tìm hiểu và viết báo cáo phần Các thành phần
2	B19DCAT117	Phạm Xuân Long	Nhóm trưởng, phân công công việc. Tổng hợp báo cáo. Thuyết trình.
3	B19DCAT121	Nguyễn Thị Quỳnh Mai	Thiết kế slide.
4	B19DCAT123	Nguyễn Công Mạnh	Tìm hiểu và viết báo cáo phần Cài đặt
5	B19DCAT124	Nguyễn Văn Mạnh	Tìm hiểu và viết báo cáo phần Giới thiệu, Kiến trúc
6	B19DCAT129	Nguyễn Văn Nam	Tìm hiểu và viết báo cáo phần Cơ chế hoạt động
7	B19DCAT130	Trần Thanh Nhàn	Tìm hiểu và viết báo cáo phần Ưu và nhược điểm

Mục lục

Giới thiệu

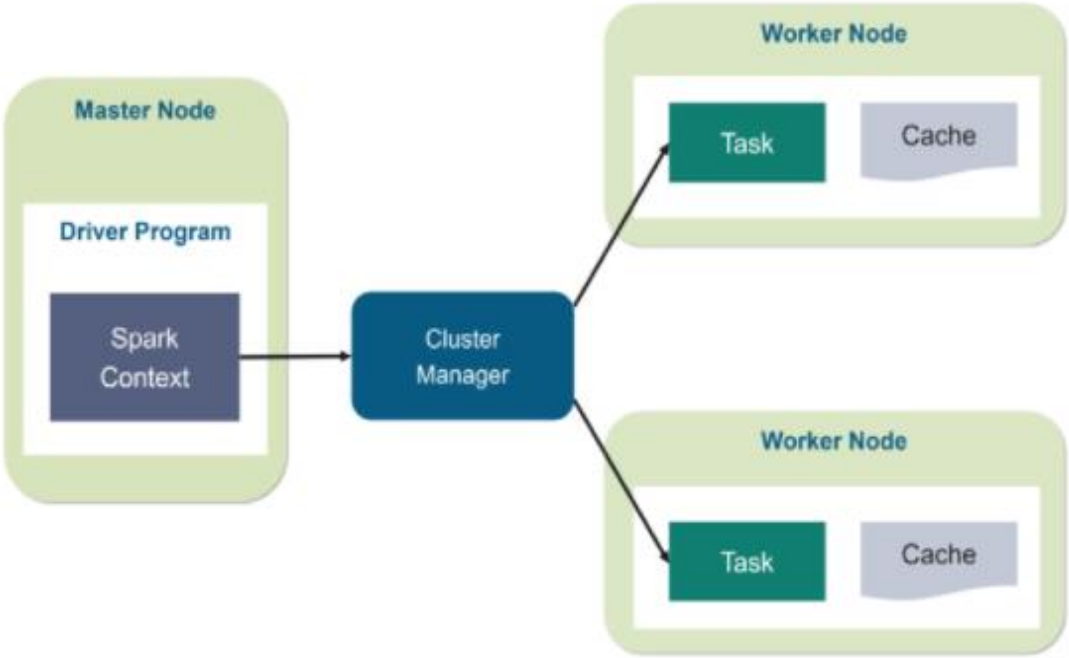
Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab tại đại học California. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay. Nó cho phép xây dựng các mô hình dự đoán nhanh chóng với việc tính toán 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.

Spark cho phép bạn có thể xử lý các dữ liệu theo thời gian thực. Nghĩa là nó có thể vừa nhận dữ liệu từ các nguồn dữ liệu khác nhau, vừa có thể thực hiện xử lý ngay những dữ liệu mà nó vừa nhận được một cách đồng thời

I. Kiến trúc của Apache Spark

Apache Spark có kiến trúc phân lớp được xác định rõ ràng trong đó tất cả các thành phần và lớp tia lửa được ghép nối lỏng lẻo. Kiến trúc này được tích hợp thêm với nhiều phần mở rộng và thư viện khác nhau. Kiến trúc của một ứng dụng Spark gồm: Trình điều khiển(driver), Trình thực thi(Executors), Trình quản lý cụm(The Cluster manager).



Hình: Kiến trúc Spark

1. Trình điều khiển(driver).

Nó là người điều khiển việc thực thi ứng dụng Spark và duy trì tất cả các trạng thái của cụm Spark (trạng thái và nhiệm vụ của những người thực thi). Nó phải giao tiếp với trình quản lý cụm để thực sự có được tài nguyên vật lý và khởi chạy các trình thực thi.

Vào cuối ngày, đây chỉ là một quá trình trên một máy vật lý có nhiệm vụ duy trì trạng thái của ứng dụng đang chạy trên cụm.

2. Trình thực thi(Executors).

Trình thực thi Spark là các quy trình thực hiện các nhiệm vụ được giao bởi trình điều khiển Spark. Trình thực thi có một trách nhiệm cốt lõi: nhận các nhiệm vụ được giao bởi Trình điều khiển, chạy chúng và báo cáo lại trạng thái (thành công hay thất bại) và kết quả của họ. Mỗi Ứng dụng Spark có các quy trình thực thi riêng biệt.

3. Trình quản lý cụm(The Cluster manager).

Trình điều khiển Spark và Trình thực thi không tồn tại trong khoảng trống và đây là lúc Trình quản lý cụm tham gia. Trình quản lý cụm chịu trách nhiệm duy trì một nhóm máy sẽ chạy ứng dụng Spark của bạn.

Khi đến lúc thực sự chạy ứng dụng Spark, yêu cầu tài nguyên từ người quản lý cụm để chạy nó. Tùy thuộc vào cách ứng dụng xác được định cấu hình, điều này có thể bao gồm một nơi để chạy trình điều khiển Spark hoặc có thể chỉ là tài nguyên cho trình thực thi cho Ứng dụng Spark. Trong quá trình thực thi ứng dụng Spark, Trình quản lý cụm sẽ chịu trách nhiệm quản lý các máy cơ bản mà ứng dụng đang chạy.

II. Các thành phần của Apache Spark.

Apache Spark có 5 thành phần là: Spark Core, Spark SQL + DataFrames, Spark Streaming, Spark MLlib, Spark GrapX.

1. Apache Spark Core.

Spark core là thành phần cốt lõi thực thi cho các tác vụ cơ bản làm nền tảng cho các chức năng khác. Nó cung cấp khả năng tính toán trên bộ nhớ và database trong bộ nhớ hệ thống lưu trữ ngoài, để hỗ trợ nhiều ứng dụng và Java, Scala và Python API để phát triển. Đặc biệt, Spark core cung cấp API để định nghĩa RDD (Resilient Distributed DataSet) là tập hợp của các mục được phân tán trên các node của cluster và có thể được xử lý song song.

2. Spark SQL.

Spark SQL là một thành phần nằm trên Spark Core nó cung cấp một sự ảo hóa mới cho dữ liệu là SchemaRDD, hỗ trợ các dữ liệu có cấu trúc và bán cấu trúc. Nó cũng cung cấp khả năng tích hợp mạnh mẽ với các thành phần còn lại của Spark(ví dụ tích hợp xử lý truy vấn SQL với học máy).

3. Spark Streaming.

Spark Streaming cho phép thực hiện phân tích xử lý trực tuyến theo lô. Chạy trên Spark, Spark Streaming cho phép ứng dụng tương tác và phân tích mạnh mẽ

trên cả dữ liệu trực tuyến và dữ liệu lịch sử, đồng thời thừa hưởng tính dễ sử dụng của spark và đặc tính chịu lỗi. Nó dễ dàng tích hợp với nhiều nguồn dữ liệu phổ biến, bao gồm HDFS, Flume, Kafka và Twitter.

4. Spark 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. Được xây dựng trên Spark, MLlib là một thư viện học có thể mở rộng được, cung cấp cả các thuật toán chất lượng cao (ví dụ, lặp lại nhiều lần để tăng độ chính xác) và tốc độ phát sáng (lên đến 100x nhanh hơn MapReduce). Thư viện này có thể sử dụng được trong Java, Scala và Python như là một phần của các ứng dụng Spark, để bạn có thể đưa nó vào quy trình làm việc hoàn chỉnh.

5. Spark GrapX.

Grapx là nền tảng xử lý đồ thị dựa trên Spark, cho phép người dùng tương tác xây dựng, biến đổi và lý luận về dữ liệu có cấu trúc dạng đồ thị theo quy mô. Nó cung cấp các API để diễn tả các tính toán trong đồ thị bằng cách xử dụng Pregel Api.

Trong các thư viện mà Spark cung cấp thì có 69% người dùng Spark SQL, 62% sử dụng DataFrames, Spark Streaming và MLlib + GrapX là 58%. Ngoài ra các thư viện trên, còn có những thư viện khác có thể tích hợp với Spark như:

- Alluxio (trước đây gọi là Tachyon) là một tệp tin phân tán tập trung vào bộ nhớ cho phép chia sẻ tập tin tin cậy ở tốc độ bộ nhớ giữa các cluster các thư viện như Spark và MapReduce. Nó lưu trữ tập tin thiết lập làm việc trong bộ nhớ, do đó tránh đi đến đĩa để tải thường xuyên đọc datasets. Điều này cho phép các công việc / truy vấn và các thư viện khác nhau truy cập các tệp tin lưu trữ ở tốc độ bộ nhớ.
- BlinkDB là một công cụ truy vấn gần đúng để sử dụng để chạy tương tác truy vấn SQL trên khối lượng lớn dữ liệu. Nó cho phép người sử dụng tính toán truy vấn chính xác cho thời gian phản hồi.

Nó hoạt động trên các tập dữ liệu lớn bằng cách chạy truy vấn trên các mẫu dữ liệu và trình bày các kết quả được ghi chú có lỗi có ý nghĩa. Các phiên bản sau này của BlinkDB đang được duy trì dưới 1 dự án mới gọi là iOLAP.

III. Cơ chế hoạt động của Apache Spark.

Mọi ứng dụng spark bao gồm 1 Driver program để chạy hàm main của người dùng và thực hiện tính toán song song trên 1 cụm.

Có 2 khái niệm trừu tượng quan trọng trong Spark là RDD và Shared Variables.

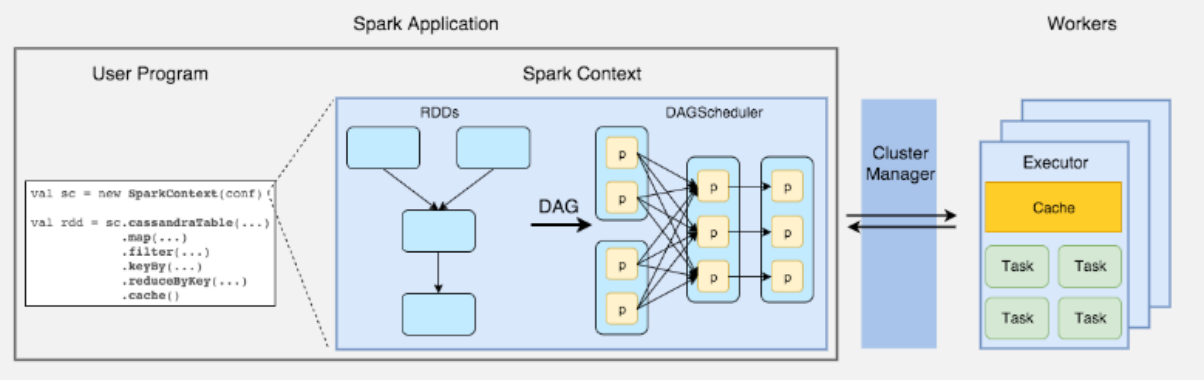
- RDD là tập dữ liệu phân tán mà các dữ liệu này được phân tán vào các node của cluster để thực hiện tính toán song song.
- Shared variables thực hiện chia sẻ biến giữa các task hoặc giữa các task với driver program.

1. RRDs

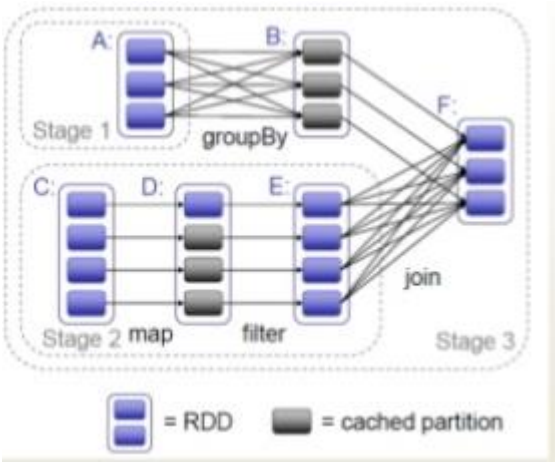
RDDs (Resilient Distributed Datasets) là một cấu trúc dữ liệu cơ bản của Spark. Nó là một tập hợp bất biến phân tán của một đối tượng. Mỗi dataset trong RDD được chia ra thành nhiều phân vùng logical. Có thể được tính toán trên các node khác nhau của một cụm máy chủ (cluster). RDDs có thể chứa bất kỳ kiểu dữ liệu nào của Python, Java, hoặc đối tượng Scala, bao gồm các kiểu dữ liệu do người dùng định nghĩa.

RDDs hỗ trợ hai kiểu thao tác: Transformations (Chuyển đổi) và Action (Hành động). Thao tác chuyển đổi (transformation) tạo ra dataset từ dữ liệu có sẵn. Qua 1 phương thức transformations thì sẽ cho phép tạo mới 1 RDD từ 1 RDD đã tồn tại. Thao tác actions trả về giá trị cho chương trình điều khiển (driver program) sau khi thực hiện tính toán trên dataset.

Transformations là một thao tác “lazy”, có nghĩa là thao tác này sẽ không thực hiện ngay lập tức, mà chỉ ghi nhớ các bước thực hiện lại. Thao tác này chỉ được thực hiện khi trong quá trình thực hiện có Actions được gọi thì công việc tính toán của Transformations mới được diễn ra. Nhờ thiết kế này mà Spark chạy hiệu quả hơn.



Cơ chế lập lịch:



Một số Action là: giảm, thu thập, đếm, lần đầu tiên, lấy, countByKey, và foreach.

+ 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 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.

+ Mỗi Worker bao gồm một hoặc nhiều Excutor. Các excutor 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).

2. Shared variables

Spark cung cấp 2 kiểu chia sẻ biến: broadcast variable và accumulators.

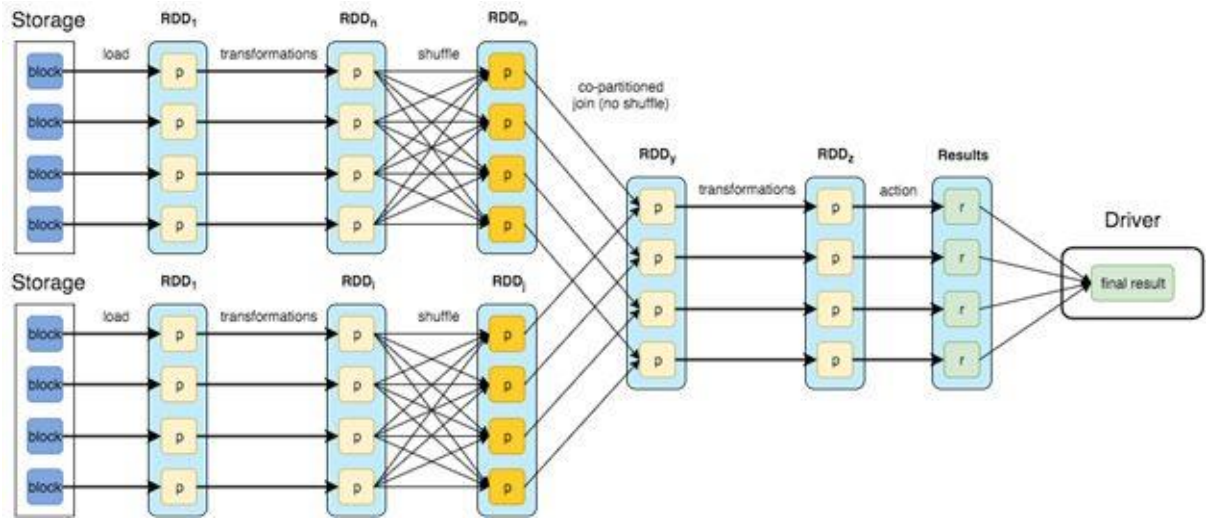
- Broadcast variable là biến mà chỉ được phép đọc giá trị của biến trên mỗi máy, không cho phép sửa đổi giá trị nhằm mục đích đảm bảo cùng giá trị của biến broadcast trên tất cả các node. Có thể sử dụng giá trị của biến qua phương thức value.

- Accumulators là biến mà chỉ thực hiện chỉ "added" qua phương thức kết hợp (như sum hoặc count).

Khác với biến broadcast, biến accumulator cho phép thay đổi giá trị, cho phép đọc ghi dữ liệu lên biến này. Điều này rất tiện lợi cho việc tính toán song song trên các cụm và thực hiện cho cùng mục đích là cùng nguồn tài nguyên. Sau khi tính toán song nó sẽ trả về driver program.

3. Tổng kết

Cơ chế hoạt động của Spark: Spark đưa ra một khái niệm mới RDD - Resilient Distributed Dataset đóng vai trò như 1 cấu trúc dữ liệu cơ bản trong Spark, RDD được định nghĩa là trừu tượng cho một tập hợp các phần tử bất biến (bản chất là được lưu trên các ô nhớ read-only), được phân vùng có thể được chia sẻ, tác động song song. Qua đó, input data từ storage system chỉ cần load 1 lần duy nhất, các step thực hiện biến đổi, xử lý input data được lên kế hoạch, optimize và thực hiện một cách liên tục cho đến khi output được trả khi kết thúc công việc. Toàn bộ quá trình đó được diễn ra trên bộ nhớ RAM (khi hết RAM sẽ được chuyển sang xử lý trên Disk) tận dụng được hiệu suất I/O cao.



IV. Cài đặt Apache Spark trên Ubuntu

1. Download và cài đặt Java

Dùng câu lệnh sau để cài đặt jdk.

```
nguyencongmanh123@ubuntu:~$ sudo apt-get install openjdk-8-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Kiểm tra phiên bản java đã cài đặt bằng câu lệnh: java –version

```
nguyencongmanh123@ubuntu:~$ java -version
openjdk version "1.8.0_292"
OpenJDK Runtime Environment (build 1.8.0_292-8u292-b10-0ubuntu1~16.04.1-b10)
OpenJDK 64-Bit Server VM (build 25.292-b10, mixed mode)
```

Thiết lập biến môi trường JAVA_HOME và PATH.

Ta mở file ~/.bashrc với dòng lệnh: gedit ~/.bashrc

```
nguyencongmanh123@ubuntu:~$ gedit ~/.bashrc
```

Thêm hai dòng lệnh dưới vào cuối file

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
```

Nhấn Save để lưu file và mở một cửa sổ terminal rồi dùng dòng lệnh “source ~/.bashrc” để thực thi thay đổi của file ~/.bashrc trên toàn bộ môi trường hiện tại.

```
nguyencongmanh123@ubuntu:~$ source ~/.bashrc
```

2. Download và cài đặt Scala

Dùng lệnh sau để cài đặt Scala:

```
nguyencongmanh123@ubuntu:~$ sudo apt-get install scala
[sudo] password for nguyencongmanh123:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Kiểm tra phiên bản scala đã cài đặt bằng câu lệnh: scala –version

```
nguyencongmanh123@ubuntu:~$ scala -version
Scala code runner version 2.11.6 -- Copyright 2002-2013, LAMP/EPFL
nguyencongmanh123@ubuntu:~$
```

3. Download và cài đặt Spark

Truy cập trình duyệt web để tải Spark về máy

https://www.apache.org/dyn/closer.lua/spark/spark-2.4.0/spark-2.4.0-bin-hadoop2.7.tgz

Tiến hành giải nén file vừa tải về

```
nguyencongmanh123@ubuntu:~$ sudo tar -xvf /home/nguyencongmanh123/Downloads/spark-2.4.0-bin-hadoop2.7.tgz
spark-2.4.0-bin-hadoop2.7/
spark-2.4.0-bin-hadoop2.7/python/
spark-2.4.0-bin-hadoop2.7/python/setup.cfg
spark-2.4.0-bin-hadoop2.7/python/pyspark/
spark-2.4.0-bin-hadoop2.7/python/pyspark/resultiterable.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/python/
spark-2.4.0-bin-hadoop2.7/python/pyspark/python/pyspark/
spark-2.4.0-bin-hadoop2.7/python/pyspark/python/pyspark/shell.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/heapq3.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/join.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/version.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/rdd.py
```

Khai báo biến môi trường để Spark có thể hoạt động. Mở file ~/.bashrc

```
nguyencongmanh123@ubuntu:~$ gedit ~/.bashrc
```

Thêm hai dòng lệnh dưới vào cuối file:

```
export SPARK_HOME=/home/nguyencongmanh123/Documents/spark/spark-2.4.0-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin
```

sh ▾ Tab Width: 8 ▾ Ln 122, Col 34 ▾ INS

Nhấn Save để lưu file và mở một cửa sổ terminal rồi dùng dòng lệnh “source ~/.bashrc” để thực thi thay đổi của file ~/.bashrc trên toàn bộ môi trường hiện tại.

```
nguyencongmanh123@ubuntu:~$ source ~/.bashrc
nguyencongmanh123@ubuntu:~$
```

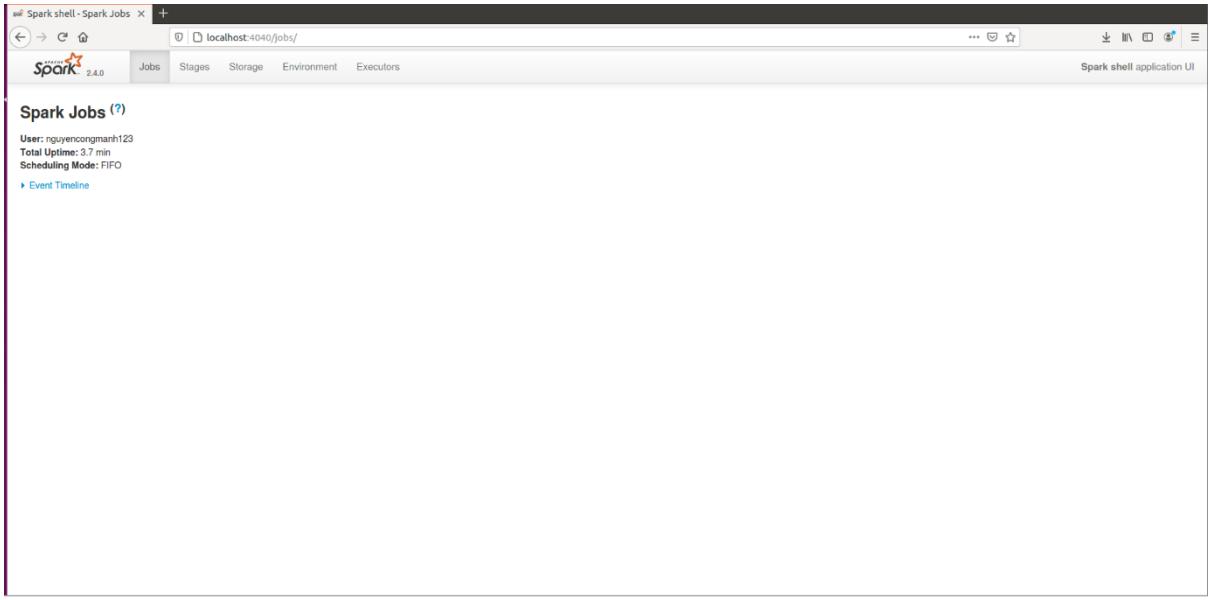
Như vậy chúng ta đã cài đặt xong Apache Spark trên Ubuntu. Tiến hành khởi động dịch vụ bằng câu lệnh: spark-shell

```
nguyencongmanh123@ubuntu:~$ spark-shell
2022-03-07 09:51:15 WARN  Utils:66 - Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.17.132 instead (on interface ens33)
2022-03-07 09:51:15 WARN  Utils:66 - Set SPARK_LOCAL_IP if you need to bind to another address
2022-03-07 09:51:15 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://192.168.17.132:4040
Spark context available as 'sc' (master = local[*], app id = local-1646675482220).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0
      /_/

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_292)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Chúng ta có thể theo dõi tình trạng hoạt động trên web qua địa chỉ localhost:4040



4. Tùy chọn thêm với PySpark

Mặc định Spark đi kèm với ngôn ngữ lập trình Scala. Nếu muốn dùng ngôn ngữ Python thì cần phải cài thêm gói "PySpark" để sử dụng Spark vs Python

Dùng câu lệnh sau để cài đặt Python

```
nguyencongmanh123@ubuntu:~$ sudo apt-get install python3-pip
[sudo] password for nguyencongmanh123:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Khai báo biến môi trường để Spark có thể hoạt động. Mở file ~/.bashrc

```
nguyencongmanh123@ubuntu:~$ gedit ~/.bashrc
```

Thêm dòng lệnh dưới vào cuối file

```
export PYSPARK_PYTHON=/usr/bin/python3
```

Khởi động dịch vụ Spark:

```
nguyencongmanh123@ubuntu:~$ pyspark
Python 3.5.2 (default, Jan 26 2021, 13:30:48)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
2022-03-07 10:47:46 WARN  Utils:66 - Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.17.132 instead (on interface ens33)
2022-03-07 10:47:46 WARN  Utils:66 - Set SPARK_LOCAL_IP if you need to bind to another address
2022-03-07 10:47:46 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.0
      /_/

Using Python version 3.5.2 (default, Jan 26 2021 13:30:48)
SparkSession available as 'spark'.
>>>
```

V. Ưu điểm và nhược điểm của Apache Spark

1. Ưu điểm

- Tốc độ

  Khi nói đến Dữ liệu lớn, tốc độ xử lý luôn quan trọng. Apache Spark cực kỳ phổ biến với các nhà khoa học dữ liệu vì tốc độ của nó. Spark nhanh hơn 100 lần so với Hadoop để xử lý dữ liệu quy mô lớn. Apache Spark sử dụng hệ thống tính toán trong bộ nhớ (RAM) trong khi Hadoop sử dụng không gian bộ nhớ cục bộ để lưu trữ dữ liệu. Spark có thể xử lý nhiều petabyte dữ liệu nhóm của hơn 8000 nút cùng một lúc.

- Dễ sử dụng

  Apache Spark mang các API dễ sử dụng để hoạt động trên các tập dữ liệu lớn. Nó cung cấp hơn 80 nhà khai thác cấp cao giúp bạn dễ dàng xây dựng các ứng dụng song song.

- Phân tích nâng cao

  Spark không chỉ hỗ trợ 'MAP' và 'Reduce'. Nó cũng hỗ trợ học máy (ML), thuật toán đồ thị, truyền dữ liệu trực tuyến, truy vấn SQL, xử lý theo stream…

- Hỗ trợ đa ngôn ngữ

  Apache Spark hỗ trợ nhiều ngôn ngữ để viết mã như Python, Java, Scala, v.v.

- Mạnh mẽ

  Apache Spark có thể xử lý nhiều thách thức phân tích vì khả năng xử lý dữ liệu trong bộ nhớ có độ trễ thấp. Nó có các thư viện được xây dựng tốt cho các thuật toán phân tích đồ thị và học máy.

- Tăng quyền truy cập vào dữ liệu lớn

  Apache Spark đang mở ra nhiều cơ hội cho dữ liệu lớn và theo khảo sát gần đây do IBM thực hiện đã công bố rằng họ sẽ đào tạo hơn 1 triệu kỹ sư dữ liệu và nhà khoa học dữ liệu về Apache Spark.

- Nhu cầu đối với nhà phát triển Spark

  Apache Spark không chỉ mang lại lợi ích cho tổ chức của bạn mà còn mang lại lợi ích cho bạn. Các nhà phát triển Spark có nhu cầu cao đến mức các công ty cung cấp các lợi ích hấp dẫn và cung cấp thời gian làm việc linh hoạt chỉ để thuê các chuyên gia có kỹ

năng về Apache Spark. Đối với những người muốn tạo dựng sự nghiệp trong lĩnh vực dữ liệu lớn, công nghệ có thể học Apache Spark.

- Vì là mã nguồn mở nên chúng ta có thể dễ dàng cập nhật và chia sẻ thông tin trên các diễn đàn.

2. Nhược điểm

- Không hỗ trợ xử lý thời gian thực

  Trong Spark Streaming , luồng dữ liệu trực tiếp đến được chia thành các lô của khoảng thời gian được xác định trước và mỗi lô dữ liệu được xử lý giống như Cơ sở dữ liệu phân tán có khả năng phục hồi của Spark (RDD). Sau đó, các RDD này được xử lý bằng cách sử dụng các hoạt động như ánh xạ, giảm, nối, v.v. Kết quả của các hoạt động này được trả về theo lô. Do đó, nó không phải là xử lý thời gian thực mà Spark gần như xử lý dữ liệu trực tiếp theo thời gian thực. Xử lý hàng loạt vi mô diễn ra trong Spark Streaming.

- Không có quy trình tối ưu hóa tự động

  Trong trường hợp của Apache Spark, bạn cần phải tối ưu hóa mã theo cách thủ công vì nó không có bất kỳ quy trình tối ưu hóa mã tự động nào. Điều này sẽ trở thành bất lợi khi tất cả các công nghệ và nền tảng khác đang hướng tới tự động hóa.

- Hệ thống quản lý tệp

  Apache Spark không đi kèm với hệ thống quản lý tệp của riêng nó. Nó phụ thuộc vào một số nền tảng khác như Hadoop hoặc các nền tảng dựa trên đám mây khác.

- Ít thuật toán hơn

  Có ít thuật toán hơn trong trường hợp Apache Spark Machine Learning Spark MLlib. Nó tụt hậu về một số thuật toán có sẵn.

- Gặp vấn đề với tệp nhỏ

  Một lý do nữa để đổ lỗi cho Apache Spark là vấn đề với các tệp nhỏ. Các nhà phát triển gặp phải các vấn đề về tệp nhỏ khi sử dụng Apache Spark cùng với Hadoop. Hệ thống tệp phân tán Hadoop (HDFS) cung cấp một số lượng hạn chế các tệp lớn thay vì một số lượng lớn các tệp nhỏ.

- Tiêu chí cửa sổ

  Dữ liệu trong Apache Spark chia thành các lô nhỏ theo khoảng thời gian được xác định trước. Vì vậy, Apache sẽ không hỗ trợ tiêu chí cửa sổ dựa trên bản ghi. Thay vào đó, nó cung cấp tiêu chí cửa sổ dựa trên thời gian.

- Không phù hợp với môi trường nhiều người dùng

  Apache Spark không phù hợp với môi trường nhiều người dùng. Nó không có khả năng xử lý đồng thời nhiều người dùng hơn.

Tài liệu tham khảo

1. https://www.analyticsvidhya.com/blog/2020/11/data-engineering-for-beginners-get-acquainted-with-the-spark-architecture/
2. https://bizfly.vn/techblog/apache-spark-la-gi.html
3. https://www.knowledgehut.com/blog/big-data/apache-spark-advantages-disadvantages
4. https://data-flair.training/blogs/limitations-of-apache-spark/
5. https://www.linkedin.com/pulse/four-important-advantages-apache-spark-elegant-microweb

Lời cảm ơn

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành tới thầy Hoàng Xuân Dậu đã đồng hành cùng chúng em và truyền đạt cho sinh viên những kiến thức giá trị và cho bọn em cơ hội được tìm hiểu và nghiên cứu đề tài Nền tảng tính toán phân tán Apache Spark. Đây là một đề tài giúp chúng em có cái nhìn rõ hơn về An toàn thông tin. Mặc dù đã cố gắng nhưng chắc chắn những hiểu biết và kỹ năng của chúng em còn nhiều thiếu sót và hạn chế. Vậy nên, bài Báo cáo khó có thể tránh khỏi những thiếu sót và những chỗ chưa chuẩn xác, kính mong thầy xem xét và góp ý giúp bài Báo cáo của chúng em được hoàn thiện hơn để em rút kinh nghiệm và chỉnh chu hơn trong những bài tập khác.