TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC XỬ LÝ DỮ LIỆU LỚN

SPARK STRUCTURED STREAMING

Người hướng dẫn: **Thầy Lê Anh Cường**

Người thực hiện:

Mạch Trung Tín - 51801039

Phan Lê Hoài Nam - 51800905

Lưu Huy Thông - 51800631

Khoá: **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC XỬ LÝ DỮ LIỆU LỚN

SPARK STRUCTURED STREAMING

Người hướng dẫn: **Thầy Lê Anh Cường**

Người thực hiện:

Mạch Trung Tín - 51801039

Phan Lê Hoài Nam - 51800905

Lưu Huy Thông - 51800631

Khoá: **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn đến nhà trường, thầy cô giảng viên khoa công nghệ thông tin trường Đại Học Tôn Đức Thắng. Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên - thầy Lê Anh Cường. Thầy đã hướng dẫn cho chúng em tận tình về nhiều mặt, góp ý bổ ích để chúng em có thể hoàn thành bài báo cáo cuối kỳ môn Xử lý dữ liệu lớn.

Chúng em luôn hiểu rõ sự khó khăn trong việc dạy và học trong thời điểm dịch bệnh đang rất phức tạp. Tuy nhiên với sự nỗ lực không ngừng nghỉ của thầy và trò, thì chúng em cũng tự tin rằng đây chính là một sản phẩm mà chứa đựng những cố gắng, tìm tòi cũng như là nỗ lực của cả thầy và trò. Một lần nữa, em xin chân thành cảm ơn thầy Cường, cảm ơn các bạn cùng nhóm vì đã cố gắng hết sức của bản thân, để có thể viết nên những lời này.

BÁO CÁO ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm báo cáo của riêng chúng tôi và được sự hướng dẫn của Thầy Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 07 tháng 01 năm 2022

Tác giả

(ký tên và ghi rõ họ tên)

Mạch Trung Tín

Phan Lê Hoài Nam

Lưu Huy Thông

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(ký và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(ký và ghi họ tên)

MỤC LỤC

CHƯƠNG 1 - CƠ SỞ LÝ THUYẾT

Trước khi đi vào tìm hiểu Spark Streaming, ta cần biết Streaming là gì? Streaming là công nghệ truyền dữ liệu liên tục. Nếu như trước đây, khí xem 1 video, ta cần download toàn bộ video đó về thí streaming chia video thành nhiều phần nên ta chỉ cần loading trước 1 lượng dữ liệu nhỏ. Hiện nay, streaming đang dần trở thành 1 phần quan trọng trong sự phát triển của Internet.

1.1 Spark Streaming là gì?

Spark Streaming là một phần mở rộng của Spark core API cho phép xử lý luồng dữ liệu trực tiếp có khả năng mở rộng, thông lượng cao, chịu lỗi tốt. Dữ liệu input có thể từ nhiều nguồn như Kafka, Kinesis hay TCP sockets và có thể được xử lý bằng cách sử dụng các thuật toán phúc tạp như map(), reduce(), join(), window(),....

Dữ liệu đã được xử lý được đẩy vào database hoặc file system hay live dashboard. Trên thực tế, ta cũng có thể áp dụng các thuật toán học máy và xử lý đồ thị của Spark khi xử lý các data stream.

Spark Streaming cung cấp một high-level abstraction gọi là discretized stream hay DStream đại diện cho một luồng dữ liệu liên tục. DStream có thể được tạo ra từ các luồng dữ liệu đầu vào từ các nguồn như Kafka và Kinesis, hoặc bằng cách áp dụng các operations lên các DStream. Một DStream được biểu diễn dưới dạng một chuỗi các RDD.

1.1.1 Tính năng của Spark Streaming

Spark Streaming cho phép:

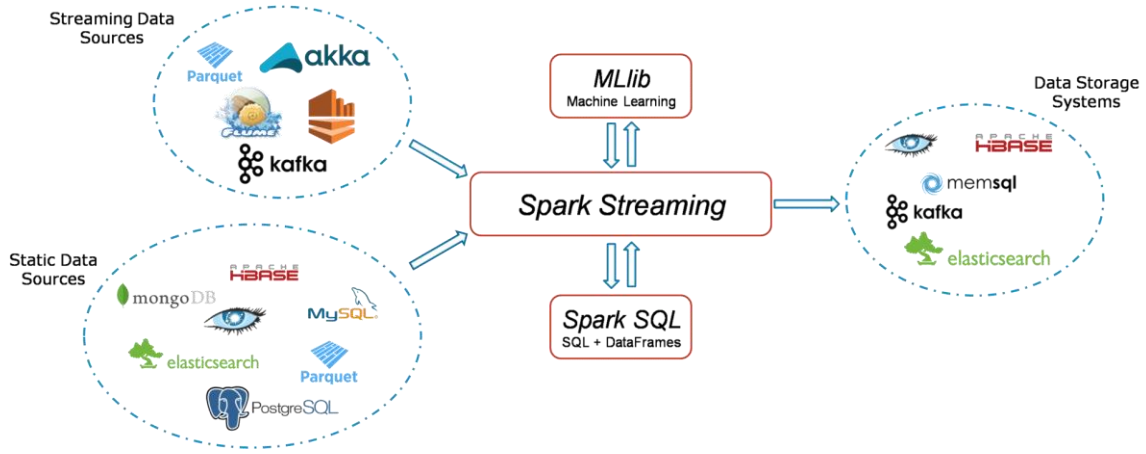
1. **Scaling**: dễ dàng scale lên hàng nghìn node.
2. **Speed**: spark streaming có khả năng giảm độ trễ xuống mức vài trăm milliseconds.
3. **Fault Tolerance**: với các hệ thống bình thường, khi 1 node bị lỗi, failed operator sẽ được tính toán lại ở node khác. Vì vậy, đôi lúc hệ thống không thể hoạt động tiếp cho đến khi node đó tính toán xong. Với Spark, 1 tính toán được chia nhỏ thành các task con cho nhiều node. Khi 1 task bị lỗi, các node khác sẽ thay thế thực hiện việc tính toán đó và do task đã rất nhỏ nên việc thực hiện lại này sẽ nhanh chóng hơn cách tiếp cận cũ.
4. **Integration**: DStream đại diện cho series các RDDs trong Spark. Vì vậy, bất cứ function nào của Apache Spark đều có thể dùng để xử lý dữ liệu.
5. **Business Analysis**: có thể áp dụng các thư viện MLlib, SQL, GraphX để phân tích dữ liệu.

1.1.2 Spark Streaming Workflow

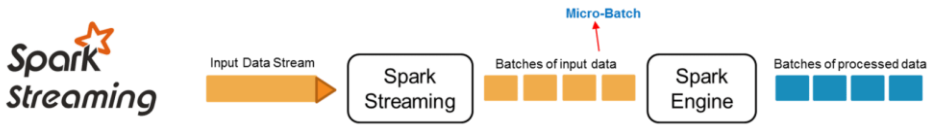
Tổng quan mà nói, một hệ thống Spark Streaming sẽ bao gồm 4 giai đoạn:

1. Giai đoạn 1: dữ liệu đẩy vào Spark Streaming rất đa dạng nguồn từ realtime streaming như Akka, Kafka, Flume, AWS, … hoặc static như HBase, MySQL, PostgreSQL, Elastic Search, MongoDB, Cassandra…
2. Giai đoạn 2: Từ Spark Streaming, dữ liệu có thể được đưa vào MLlib để áp dụng các mô hình học máy.
3. Giai đoạn 3: dữ liệu có thể được đưa vào Spark SQL phục vụ cho việc truy vấn dữ liệu.

4. Giai đoạn 4: cuối cùng, sau các thao tác với dữ liệu, nó sẽ được lưu vào database hoặc file system.



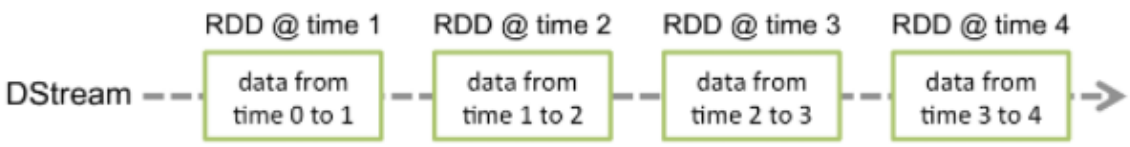
Mặt khác, dữ liệu được đẩy vào Spark Streaming sẽ được chia thành các batch nhỏ, sau đó được Spark Engine xử lý để output ra series các batch dữ liệu mới.



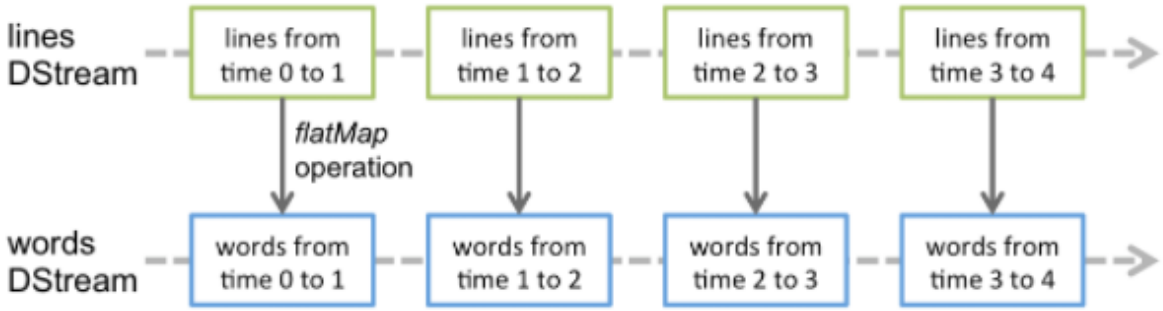
1.1.3 Cơ chế xử lý của DStream

DStream là đối tượng trừu tượng cơ bản được cung cấp bởi Spark Streaming. Nó đại diện cho một luồng dữ liệu liên tục hoặc một luồng dữ liệu đầu vào.

Một DStream được biểu diễn bằng một chuỗi RDD liên tục, là đối tượng trừu tượng của Spark trong tập dữ liệu phân tán, bất biến). Mỗi RDD trong một DStream chứa dữ liệu từ một khoảng thời gian nhất định, như hình bên dưới:



Bất kỳ tính toán nào được áp dụng trên DStream đều được chuyển thành các tính toán trên RDDs. Ví dụ:



1.1.4 Input DStream và Receivers

Input của DStream là DStream đại diện cho các luồng dữ liệu đầu vào nhận được từ các streaming sources. Như ví dụ trên, lines là DStream đầu vào vì nó đại diện cho dòng dữ liệu nhận được từ netcat. Mọi DStream đầu vào được liên kết với đối tượng Receiver nhận dữ liệu từ nguồn và lưu trữ trong bộ nhớ của Spark để xử lý.

Spark Streaming cung cấp 2 phương thức built-in streaming sources:

- Basic sources: sources có sẵn, trực tiếp trong StreamingContext API. Ví dụ: file systems, socket connections.
- Advanced sources: sources như Kafka, Kinesis,.... có sẵn thông qua các class tiện ích bổ sung.

1.2 Structured Stream là gì?

Chúng ta cần phân biệt giữa Spark Streaming với Structured Streaming. Nói một cách đơn giản, Spark Streaming được coi là phiên bản cũ, xử lý stream dựa trên RDD. Trong khi Structured Streaming là phiên bản mới, xử lý stream dựa trên Dataset/DataFrame. Các chuyên gia khuyến cáo người dùng nên sử dụng phiên bản mới với nhiều tính năng cải thiện hơn.

Structured Streaming là một công cụ xử lý luồng có thể mở rộng và chịu lỗi được xây dựng dựa trên Spark SQL engine. Ta có thể thực hiện tính toán trực tuyến giống như cách tính toán hàng loạt trên static data.

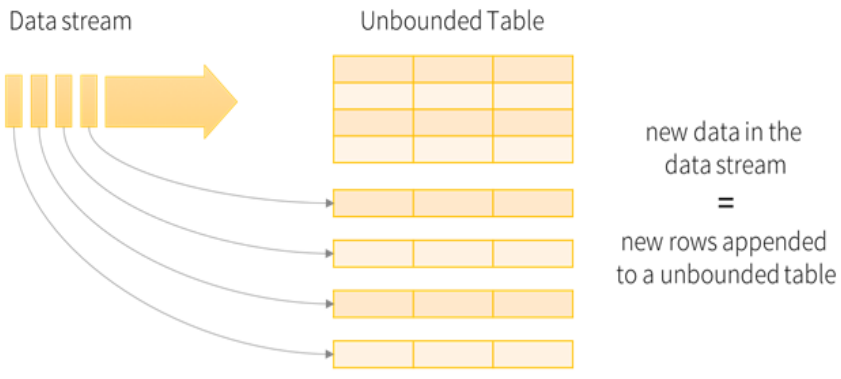
Spark SQL engine sẽ đảm nhận việc chạy chương trình tuần tự, liên tục và cập nhật kết quả cuối cùng khi dữ liệu đến.

Hệ thống đảm bảo khả năng chịu lỗi chính xác từ đầu đến cuối thông qua các điểm kiểm tra và nhật ký trước đó. Nói tóm lại, Structured Streaming cung cấp khả năng xử lý luồng nhanh, có thể mở rộng, chịu lỗi, độ chính xác cao.

1.2.1 Mô hình cơ bản của Structured Streaming

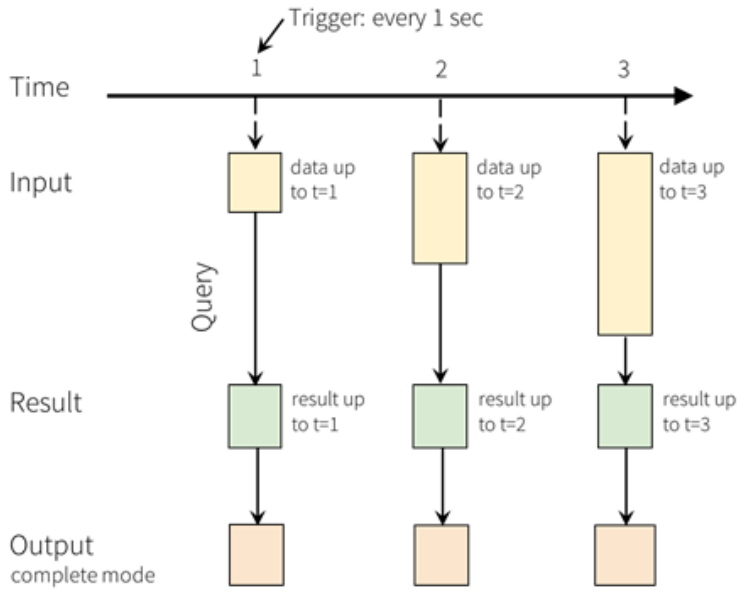
Ý tưởng chính trong phát trực tiếp có cấu trúc là coi luồng dữ liệu trực tiếp như một bảng liên tục được tổng hợp. Điều này dẫn đến một mô hình xử lý luồng mới rất giống với mô hình xử lý hàng loạt. Bạn có thể chạy tính toán luồng của nó như một lô tiêu chuẩn - như một truy vấn trên một bảng tĩnh và Spark chạy nó như một truy vấn ngược dòng trên một bảng đầu vào không bị giới hạn.

- Mô hình cơ bản: Chúng ta sẽ xem qua hình ảnh ví dụ ở dưới . Mỗi mục dữ liệu đến trong luồng giống như một hàng mới được thêm vào bảng đầu vào.



Data stream as an unbounded table

- Một truy vấn của đầu vào tạo ra một "bảng kết quả". Tại mỗi khoảng thời gian kích hoạt (ví dụ: cứ sau 1 giây), các hàng mới được thêm vào bảng đầu vào, cuối cùng sẽ cập nhật bảng kết quả. Mỗi khi bảng kết quả được cập nhật, bạn muốn ghi các hàng kết quả đã thay đổi vào một phần chìm bên ngoài.



Programming Model for Structured Streaming

- "Đầu ra" được định nghĩa là những gì được ghi ra bộ nhớ ngoài. Đầu ra có thể được xác định ở một chế độ khác như sau:

+ Chế độ Hoàn thành - Toàn bộ Bảng Kết quả được cập nhật sẽ được ghi vào bộ nhớ ngoài. Trình kết nối lưu trữ quyết định cách xử lý việc ghi toàn bộ bảng.

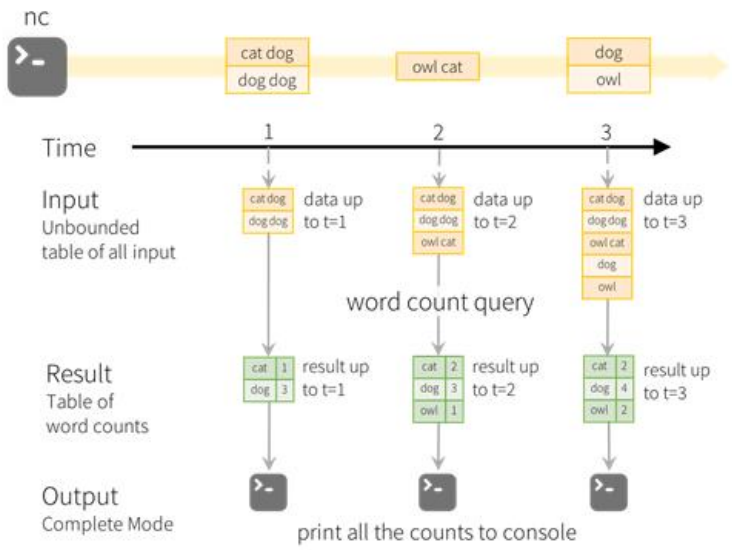
+ Chế độ Nối - Chỉ các hàng mới được thêm vào Bảng Kết quả vì lần kích hoạt cuối cùng sẽ được ghi vào bộ nhớ ngoài. Điều này chỉ áp dụng cho các truy vấn mà các hàng hiện có trong Bảng Kết quả sẽ không thay đổi.

+ Chế độ Cập nhật - Chỉ những hàng đã được cập nhật trong Bảng kết quả kể từ lần active cuối cùng sẽ được ghi vào bộ nhớ ngoài (khả dụng kể từ Spark 2.1.1). Lưu ý rằng điều này khác với Chế độ hoàn chỉnh ở chỗ chế độ này chỉ xuất ra

các hàng đã thay đổi kể từ lần kích hoạt cuối cùng. Nếu truy vấn không chứa tổng hợp, nó sẽ tương đương với chế độ Nối.

1.2.2 Lợi ích khi sử dụng Structured Streaming

Để minh họa việc sử dụng mô hình này, hãy hiểu mô hình trong ngữ cảnh của Ví dụ nhanh ở trên. Các dòng đầu tiên DataFrame là bảng đầu vào và từ cuối cùng DataCounts và Data Frame là bảng kết quả. Lưu ý rằng truy vấn trên các dòng DataFrame để tạo word Counts hoàn toàn giống như truy vấn DataFrame tĩnh. Tuy nhiên, khi truy vấn này được bắt đầu, Spark sẽ liên tục kiểm tra dữ liệu mới từ kết nối socket. Nếu có dữ liệu mới, Spark sẽ chạy truy vấn "gia tăng" kết hợp số lần chạy trước đó với dữ liệu mới để tính số lần cập nhật, như được hiển thị bên dưới.



Model of the Quick Example

Mô hình này khác biệt đáng kể so với nhiều công cụ xử lý luồng khác. Nhiều hệ thống phát trực tuyến yêu cầu người dùng tự duy trì các tập hợp đang chạy, do đó phải lý giải về khả năng chịu lỗi và tính nhất quán của dữ liệu (ít nhất một lần, hoặc nhiều nhất một lần hoặc chính xác một lần). Trong mô hình này, Spark chịu trách nhiệm cập nhật Bảng kết quả khi có dữ liệu mới, do đó giúp người dùng không phải suy luận về nó.

CHƯƠNG 2 - DỮ LIỆU ĐƯA VÀO STREAMING

2.1 Đưa dữ liệu vào Spark Streaming

Dữ liệu được đưa vào Spark Streaming qua các phương pháp như sau: Để đọc dữ liệu từ các tệp trên bất kỳ hệ thống tệp nào tương thích với HDFS API (that is, HDFS, S3, NFS, ...). Một Dstream nó được tạo thành từ cấu trúc sau: StreamingContext.fileStream[KeyClass, ValueClass, InputFormatClass].

Spark Streaming có thể nhận dữ liệu từ nhiều nguồn khác nhau, bao gồm Flume, Kinesis, Twitter, và TCP sockets. Một số ngữ cảnh chính khi triển khai hệ thống Spark Streaming mà bạn cần biết là Discretized Streams (DStreams); có thể được hiểu là một luồng dữ liệu liên tục được tạo ra từ các nguồn đầu vào. Bên trong một DStream bao gồm các Resilient Distributed Datasets (RDDs) đóng vai trò cốt lõi trong kiến trúc Spark.

2.2 Đưa dữ liệu vào Structured Streaming

- Dữ liệu được đưa vào Structured Streaming qua các phương pháp như sau:

2.1.1 File source

Đọc các sources được viết trong một thư mục dưới dạng một luồng dữ liệu. Các tệp sẽ được xử lý theo thứ tự thời gian sửa đổi tệp. Các định dạng tệp được hỗ trợ là Docs, CSV, JSON, ORC, Parquet.

2.1.2 Kafka source

Đọc dữ liệu từ Kafka. Nó tương thích với phiên bản Kafka 0.10.0 trở lên. Apache Kafka là một hệ thống xử lý hàng đợi theo cơ chế publish-subscribe;

Kafka còn hỗ trợ triển khai hệ thống thu thập log theo mô hình phân tán (distribute), phân chia (partition), và đồng bộ (replicate). Mã nguồn Kafka được thiết kế cho việc xử lý dữ liệu lớn khi đọc/ghi dữ liệu, giảm độ trễ trong quá trình truyền tải dữ liệu.

Kafka thường được triển khai theo mô hình cluster; mỗi node trong cluster được gọi là broker. Mỗi broker có thể quản lý hàng trăm megabyte đọc/ghi dữ liệu một giây từ hàng nghìn người dùng một lúc. Một cluster có thể mở rộng một cách linh động mà không gây ra tình trạng dừng hệ thống.

Kafka cung cấp một khái niệm gọi là topic, mỗi dữ liệu đến hệ thống sẽ thuộc một topic cụ thể. Hệ thống gởi dữ liệu đến Kafka được gọi là producer. Hệ thống tiếp nhận đầu ra từ Kafka được gọi là consumer.

Việc truyền tải dữ liệu giữa các hệ thống khác với Kafka broker được thực hiện thông qua giao thức TCP; việc phát triển, phân tích dữ liệu có thể thực hiện trên nhiều nền tảng ngôn ngữ lập trình khác nhau.

2.1.3 Socket source (dành cho thử nghiệm):

Đọc dữ liệu văn bản UTF8 từ kết nối socket. Server Socket nằm ở trình điều khiển. Phương pháp này chỉ phù hợp cho chúng ta khi sử dụng để thử nghiệm vì hệ thống sẽ chưa đảm bảo khả năng chịu lỗi.

2.1.4 Rate source (dành cho thử nghiệm):

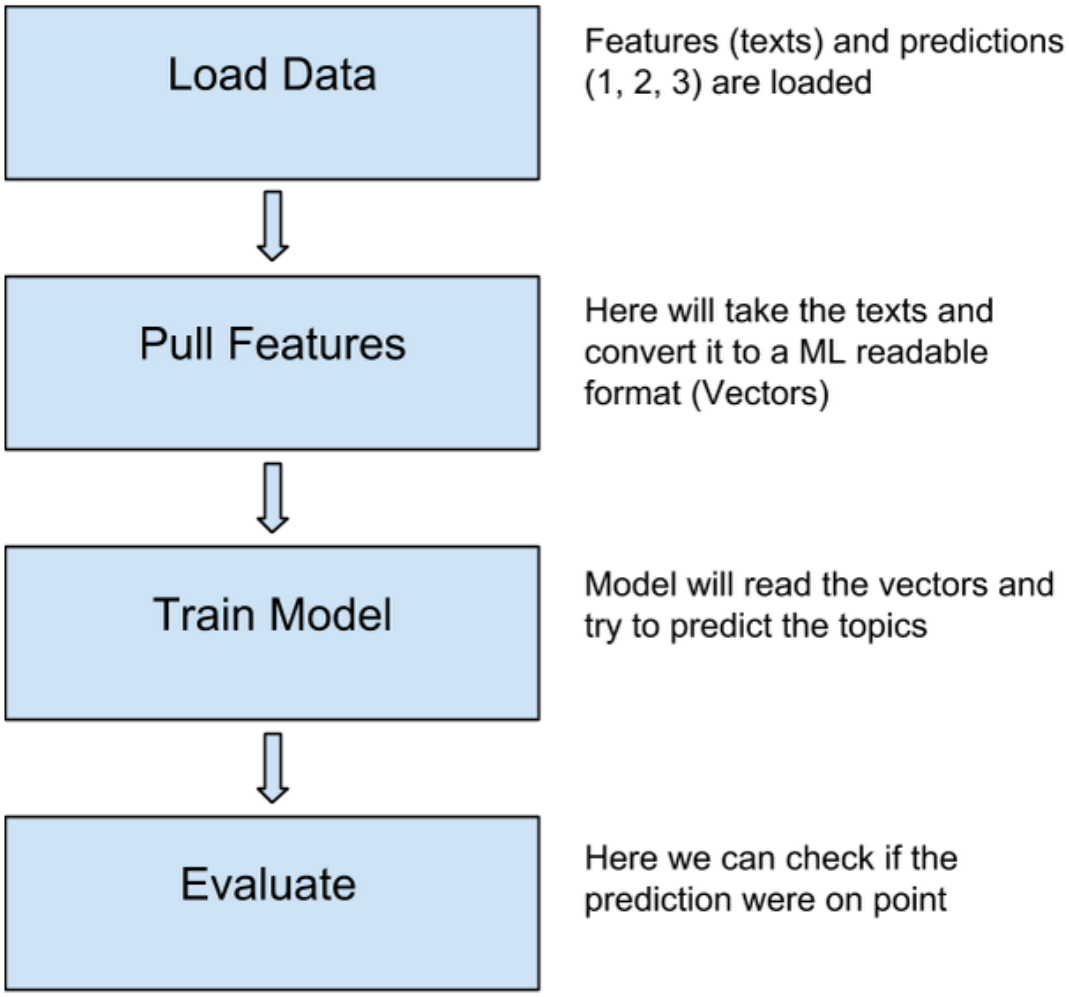
Tạo dữ liệu với số lượng hàng được setting mỗi giây, mỗi hàng đầu ra chứa một timestamp và giá trị. Trong đó timestamp là loại Timestamp chứa thời gian gửi

tin nhắn và giá trị thuộc Longtype chứa số lượng tin nhắn, bắt đầu từ 0 là hàng đầu tiên. Nguồn này được dùng để thử nghiệm và đánh giá.

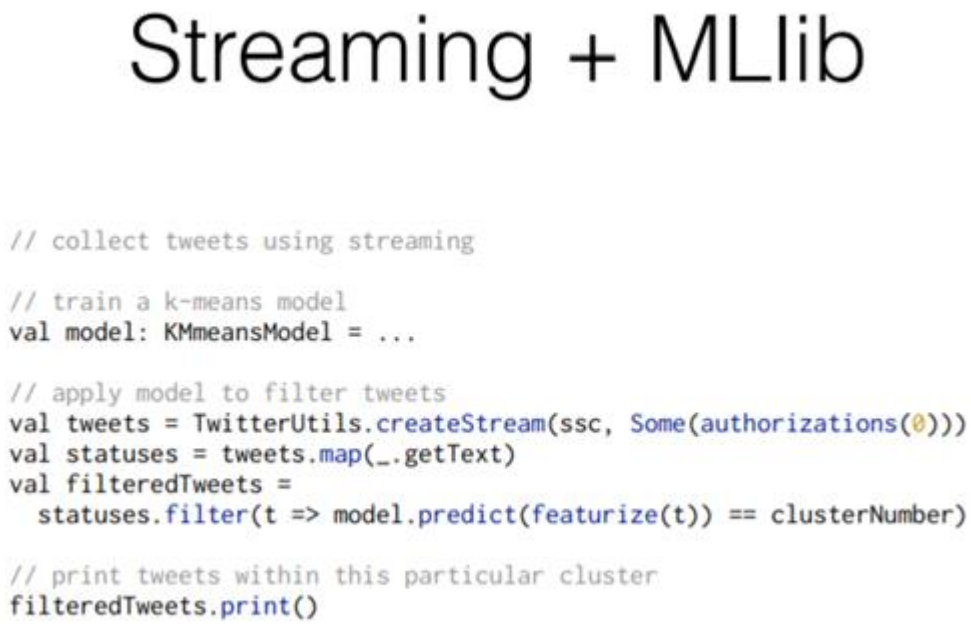
CHƯƠNG 3 - XỬ LÝ, THAO TÁC DỮ LIỆU TRONG STREAMING

3.1 Xử lý, thao tác trên dữ liệu trong Streaming bằng công cụ MLlib

Với Mllib, chúng ta có thể sử dụng thuật toán ML được cung cấp để sử dụng trong Streaming. Một số thuật toán ML hỗ trợ Streaming như là: SLR(Streaming Linear Regression), SKM(Streaming Kmeans), … Đây là những thuật toán có thể đồng thời học hỏi từ dữ liệu Streaming, áp dụng nó lên mô hình dữ liệu trực tuyến. Ngoài những điều này, đối với một lớp thuật toán máy học lớn hơn nhiều, chúng ta sẽ quan tâm mô hình học tập ngoại tuyến (tức là sử dụng dữ liệu lịch sử) và sau đó áp dụng mô hình trực tuyến trên dữ liệu truyền trực tuyến.



+ Spark MLlib được sử dụng để thực hiện học máy trong Apache Spark. MLlib bao gồm các thuật toán và tiện ích phổ biến. MLlib trong Spark là một thư viện Máy học có thể mở rộng thảo luận về cả thuật toán chất lượng cao và tốc độ cao. Các thuật toán học máy như hồi quy, phân loại, phân cụm, khai thác mẫu và lọc cộng tác. Các nguyên thủy của máy học cấp thấp hơn như thuật toán tối ưu hóa dốc gradient chung cũng có trong MLlib.



3.2 Xử lý, thao tác trên dữ liệu trong Streaming bằng công cụ Spark SQL

Spark SQL là một mô-đun Spark để xử lý dữ liệu có cấu trúc. Không giống như API Spark RDD cơ bản, các giao diện do Spark SQL cung cấp cung cấp cho Spark nhiều thông tin hơn về cấu trúc của cả dữ liệu và tính toán đang được thực hiện. Bên trong, Spark SQL sử dụng thông tin bổ sung này để thực hiện các tối ưu hóa bổ sung. Có một số cách để tương tác với Spark SQL bao gồm SQL và API tập dữ liệu. Khi tính toán một kết quả, cùng một công cụ thực thi được sử dụng, không phụ thuộc vào API / ngôn ngữ bạn đang sử dụng để diễn đạt tính toán. Sự thống nhất này có nghĩa là các nhà phát triển có thể dễ dàng chuyển đổi qua lại giữa các API khác nhau, dựa trên đó cung cấp cách tự nhiên nhất để thể hiện một chuyển đổi nhất định.

Spark SQL + MLlib

```
// Data can easily be extracted from existing sources,
// such as Apache Hive.
val trainingTable = sql("""
  SELECT e.action,
         u.age,
         u.latitude,
         u.longitude
  FROM Users u
  JOIN Events e
  ON u.userId = e.userId""")

// Since `sql` returns an RDD, the results of the above
// query can be easily used in MLlib.
val training = trainingTable.map { row =>
  val features = Vectors.dense(row(1), row(2), row(3))
  LabeledPoint(row(0), features)
}

val model = SVMWithSGD.train(training)
```

CHƯƠNG 4 - LƯU TRỮ KẾT QUẢ

Có vài cách để lưu trữ output sink:

- File Sink: Sẽ lưu trữ thông tin ra một thư mục local.

```
writeStream
    .format("parquet")        // can be "orc", "json", "csv", etc.
    .option("path", "path/to/destination/dir")
    .start()
```

- Kafka Sink: Lưu trữ đầu ra của một hoặc nhiều topics trong Kafka.

```
writeStream
    .format("kafka")
    .option("kafka.bootstrap.servers", "host1:port1,host2:port2")
    .option("topic", "updates")
    .start()
```

- Foreach Sink: Chạy tính toán trực tiếp trên các bản ghi trong output.

```
writeStream
    .foreach(...)
    .start()
```

- Console sink (Dùng để debug): In đầu ra vào console/stdout mỗi khi có kích hoạt. Cả hai chế độ output là Thêm và Thực hiện đều được hỗ trợ. Điều này sẽ được sử dụng cho mục đích gỡ lỗi trên lượng dữ liệu thấp vì toàn bộ đầu ra được thu thập và lưu trữ trong bộ nhớ của driver sau mỗi trigger.

```
writeStream
    .format("console")
    .start()
```

- Memory sink(Dùng để debug): Đầu ra được lưu trong bộ nhớ dưới dạng bảng trong bộ nhớ. Điều này sẽ được sử dụng cho mục đích gỡ lỗi trên khối lượng dữ liệu thấp vì toàn bộ đầu ra được thu thập và lưu trữ trong bộ nhớ của driver.

```
writeStream
    .format("memory")
    .queryName("tableName")
    .start()
```

CHƯƠNG 5 - TÀI LIỆU THAM KHẢO

Tài liệu tiếng Việt:

1. https://viblo.asia/p/ung-dung-spark-streaming-vao-phan-tich-cam-xuc-mang-xa-hoi-twitter-WAyK82r9lxX
2. https://ichi.pro/vi/bat-du-lieu-phat-truc-tuyen-voi-spark-structured-streaming-va-kafka-162515420296245
3. https://helpex.vn/article/cau-truc-truyen-phat-la-gi-5c6b1eb1ae03f628d053c172
4. https://filegi.com/tech-term/spark-streaming-9640/

Tài liệu tiếng Anh:

1. https://spark.apache.org/docs/latest/streaming-programming-guide.html
2. https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
3. https://www.slideshare.net/databricks/deploying-mllib-for-scoring-in-structured-streaming-with-joseph-bradley
4. https://www.analyticsvidhya.com/blog/2020/11/introduction-to-spark-mllib-for-big-data-and-machine-learning/
5. http://cloudurable.com/blog/spark-tutorial-part3-streaming-mlib/index.html