KDD2016
22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining
August 13 - 17, 2016 | San Francisco, California

Streaming Analytics Tutorial

Ashish Gupta (LinkedIn)
Neera Agarwal

Streaming Analytics

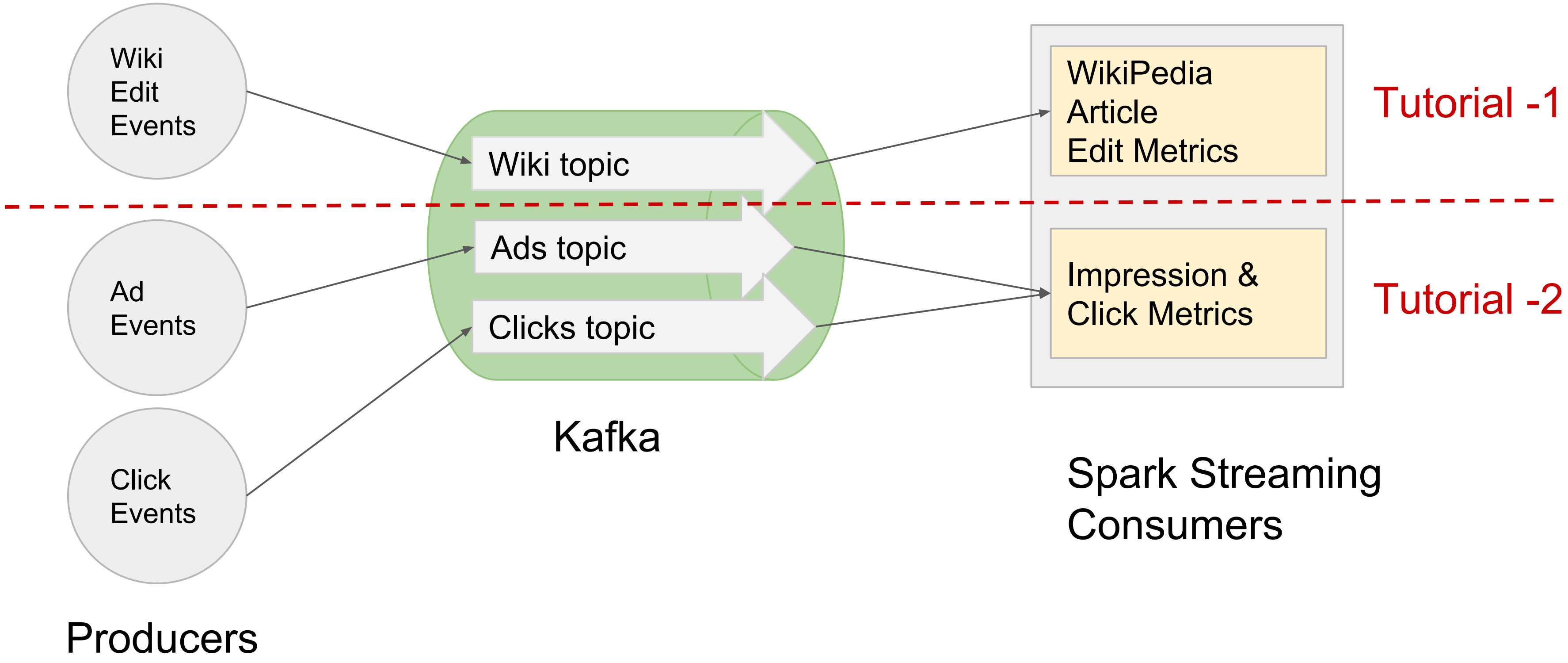
Before doing this tutorial please read the main presentation.

http://www.slideshare.net/NeeraAgarwal2/streaming-analytics

Technical Requirements

- OS: MAC OS X
- Programming Language: Scala 2.10.x
- Open source software used in tutorial: Kafka and Spark 1.6.2

Tutorials


Wiki Edit Events
Ad Events
Click Events
Producers
Wiki topic
Ads topic
Clicks topic
Kafka
WikiPedia Article Edit Metrics
Impression & Click Metrics
Spark Streaming Consumers
Tutorial -1
Tutorial -2

Step 1: Installation

In the conference we provided USB stick with the environment ready for the tutorials. Here are the instructions to create your own environment:

1. Check Java: java -version

   java version "1.8.0_92"   (Note: Java 1.7+ should work.)

2. Check Maven: mvn -v

   If not installed, check instructions in the **Additional slides** at the end.

Step 1: Installation

3. Install Scala

```
curl -O http://downloads.lightbend.com/scala/2.10.6/scala-2.10.6.tgz
```

```
tar -xzf  scala-2.10.6.tgz
```

Set scala home to the path pointing to scala-2.10.6 folder. For example on Mac:

```
export SCALA_HOME=/Users/<username>/scala-2.10.6
```

```
export PATH=$PATH:$SCALA_HOME/bin
```

4. Install Spark

```
curl -O http://d3kbcqa49mib13.cloudfront.net/spark-1.6.2-bin-hadoop2.6.tgz
```

Step 1: Installation

5. Install Kafka

```
curl -O http://apache.claz.org/kafka/0.10.0.0/kafka_2.10-0.10.0.0.tgz

tar -xzf kafka_2.10-0.10.0.0.tgz
```

6. Download tutorial
   https://github.com/NeeraAgarwal/kdd2016-streaming-tutorial

Step 2: Start Kafka

Start a terminal window. Start Zookeeper

```
> cd kafka_2.10-0.10.0.0
> bin/zookeeper-server-start.sh config/zookeeper.properties
```

Wait for zookeeper to start. Start another terminal window and start Kafka

```
> cd kafka_2.10-0.10.0.0
> bin/kafka-server-start.sh config/server.properties
```

Tutorial 1:

Bot and Human edit counts on Wikipedia Edit stream

Step 1: Listening to WikiPedia Edit Stream

Start a new terminal window. Run WikiPedia Connector

```
> cd kdd2016-streaming-tutorial

> java -cp target/streamingtutorial-1.0.0-jar-with-dependencies.jar example.WikiPediaConnector
```

After some messages, stop using CTRL-C. We will run it again after writing streaming code.

Does not run, build package and try again...

```
> mvn package
```

Step 1: WikiPedia Stream message structure

[[-acylglycerol O-acyltransferase]] MB https://en.wikipedia.org/w/index.php?diff=733783045&oldid=721976415 * BU RoBOT * (-1) /* References */Sort into more specific stub template based on presence in [[Category:EC 2.3]] or subcategories (Task 25)

[[City Building]] https://en.wikipedia.org/w/index.php?diff=733783047&oldid=732314994 * Hmains * (+9) refine category structures

---

[[Wikipedia:Articles for deletion/Log/2016 August 10]] B https://en.wikipedia.org/w/index.php?diff=733783051&oldid=733783026 * Cyberbot

---

Fields: title, flags, diffUrl, user, byteDiff, summary

Flags (2nd field): ‘M’=Minor, ‘N’ = New, ‘!’ = Unpatrolled, ‘B’ = Bot Edit

WikiPedia Stream pattern = `"\\[\\[(.*)\\]\\]\\s(.*)\\s(.*)\\s\\*\\s(.*)\\s\\*\\s\\(\\+?(.\\d*)\\)\\s(.*)".r`

Spark: RDD

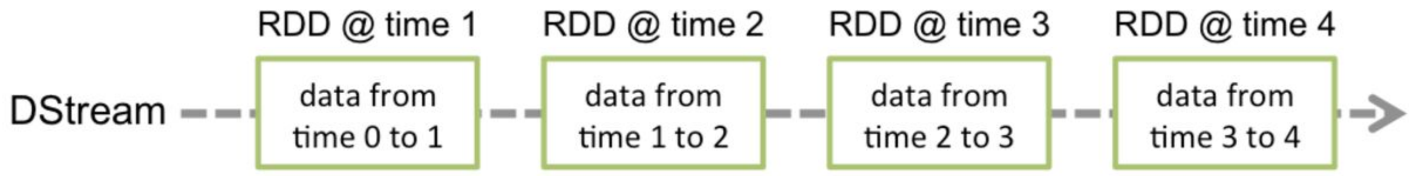
An RDD is an immutable distributed collection of objects. Each RDD is split into multiple partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of objects, including Python, Java, Scala or user-defined classes.

RDDs offer two types of operations:

- Transformations construct a new RDD from a previous one.
- Actions compute a result based on an RDD, and either return it to the driver program or save it to an external storage system.

Spark: DStream

DStream is a sequence of data arriving over time.



Internally, each DStream is represented as a sequence of RDDs arriving at each time step. DStreams offer two types of operations:

- Transformations yield a new DStream.
- Output operations write data to an external system.

Ref: https://spark.apache.org/docs/latest/streaming-programming-guide.html

Step 2: Write Code

In kdd2016-streaming-tutorial

Change code in src/main/scala/example/WikiPediaStreaming.scala file. Use your favorite editor.

```
val lines = messages.foreachRDD { rdd =>

  // ADD CODE HERE

}
```

Note: In the conference participants were asked to write the code while in github repository full code is provided.

Step 2: Continued

```
rdd =>

 val linesDF = rdd.map(row => row._2 match {

   case pattern(title, flags, diffUrl, user, byteDiff, summary) => WikiEdit(title, flags, diffUrl, user, byteDiff.toInt, summary)

   case _ => WikiEdit("title", "flags", "diffUrl", "user", 0, "summary")

 }).filter(row => row.title != "title").toDF()
```

Step 2 Continued

```
// Number of records in 10 second window.
val totalCnt = linesDF.count()

// Number of bot edited records in 10 second window.
val botEditCnt = linesDF.filter("flags like '%B%'").count()

// Number of human edited records in 10 second window.
val humanEditCnt = linesDF.filter("flags not like '%B%'").count()

val botEditPct = if (totalCnt > 0) 100 * botEditCnt / totalCnt else 0

val humanEditPct = if (totalCnt > 0) 100 * humanEditCnt / totalCnt else 0
```

Step 3: Build Program

Start a new terminal window.

```
> cd kdd2016-streaming-tutorial
> mvn package
```

Step 4: Run Programs

Run WikiPediaConnector in terminal window. It starts receiving data from WikiPedia IRC channel and writes to Kafka.

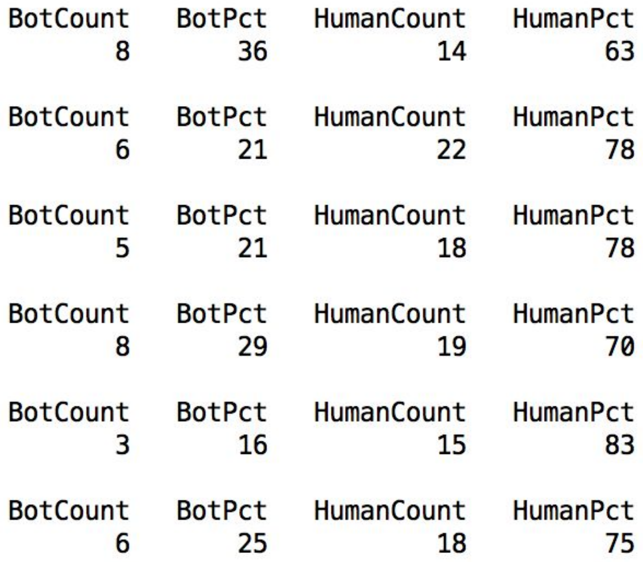
```
>java -cp target/streamingtutorial-1.0.0-jar-with-dependencies.jar
example.WikiPediaConnector
```

Run WikiPediaStream in a new terminal window.

```
> cd kdd2016-streaming-tutorial
```

```
> ../spark-1.6.2-bin-hadoop2.6/bin/spark-submit --class
example.WikiPediaStreaming
target/streamingtutorial-1.0.0-jar-with-dependencies.jar
```

Output

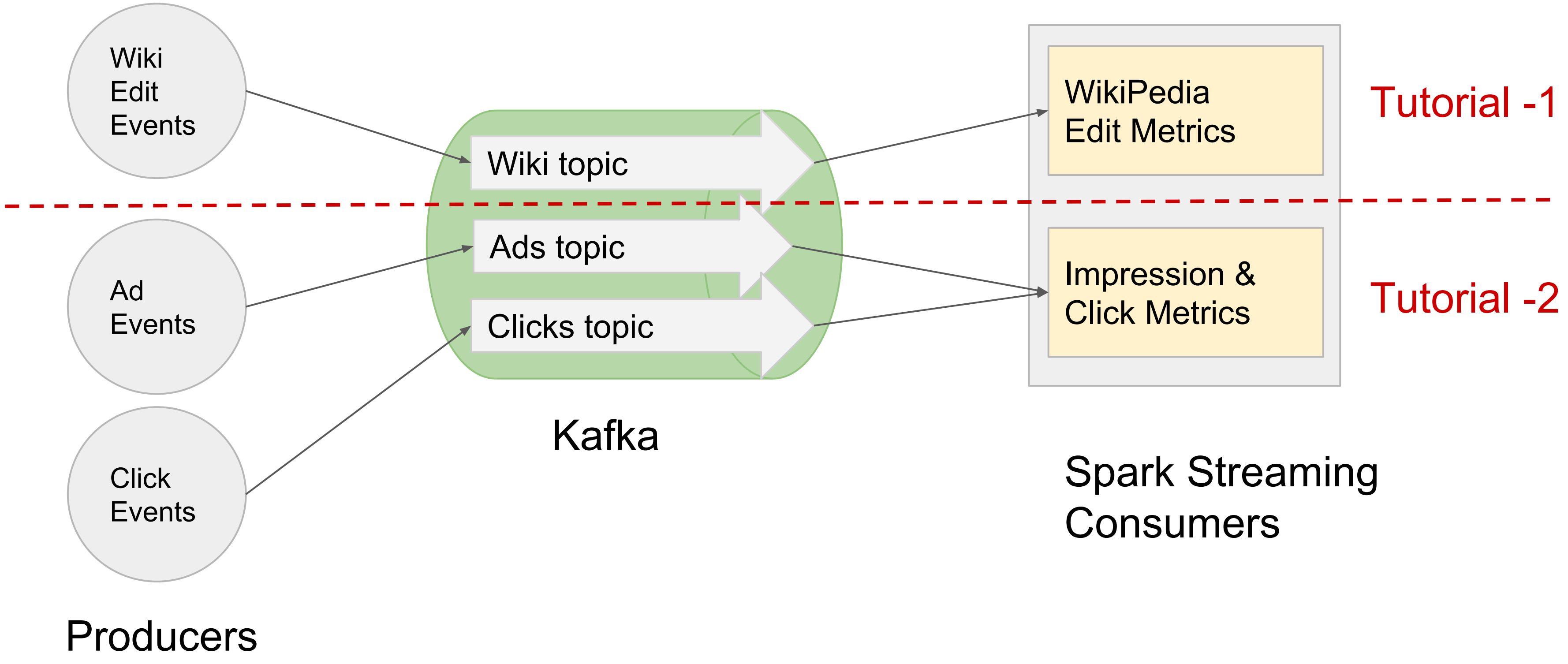


```
BotCount   BotPct   HumanCount   HumanPct
       8       36           14         63

BotCount   BotPct   HumanCount   HumanPct
       6       21           22         78

BotCount   BotPct   HumanCount   HumanPct
       5       21           18         78

BotCount   BotPct   HumanCount   HumanPct
       8       29           19         70

BotCount   BotPct   HumanCount   HumanPct
       3       16           15         83

BotCount   BotPct   HumanCount   HumanPct
       6       25           18         75
```

Tutorial 2:

Impression Click metrics on Ad and Click streams

Tutorials


Wiki Edit Events
Ad Events
Click Events
Producers
Wiki topic
Ads topic
Clicks topic
Kafka
WikiPedia Edit Metrics
Impression & Click Metrics
Spark Streaming Consumers
Tutorial -1
Tutorial -2

Step 1: Listening to Ads and Clicks stream

Run program to replay Ad and Click Events from file

```
> cd kdd2016-streaming-tutorial
```

```
> java -cp target/streamingtutorial-1.0.0-jar-with-dependencies.jar example.AdClickEventReplay
```

After some messages, stop using CTRL-C. We will run it again after writing streaming code.

Does not run, build package and try again...

```
> mvn package
```

Step 1: Ad and Click Event message structure

Ad Event:

QueryID, AdId, TimeStamp: 6815, 48195, 1470632477761

Click Event:

QueryID, ClickId, TimeStamp: 6815, 93630, 1470632827088

Join on QueryId, show metrics by Ad Id.

Step 2: Write Code

In kdd2016-streaming-tutorial

Change code in src/main/scala/example/AdEventJoiner.scala file.

```
val adEventDStream =  adStream.transform( rdd => {
   rdd.map(line => line._2.split(",")).
     map(row => (row(0).trim.toInt, AdEvent(row(0).trim.toInt, row(1).trim.toInt, row(2).trim.toLong)))
})
// ADD CODE HERE..
```

Note: In the conference participants were asked to write the code while in github repository full code is provided.

Step 2 Continued

```
//Connects Spark Streaming to Kafka Topic and gets DStream of RDDs (click event message)

val clickStream = KafkaUtils.createDirectStream[String, String, StringDecoder, StringDecoder](ssc,
kafkaParams, clickStreamTopicSet)

//Create a new DStream by extracting kafka message and converting it to DStream[queryId, ClickEvent]

val clickEventDStream = clickStream.transform{ rdd =>

 rdd.map(line => line._2.split(",")).

   map(row => (row(0).trim.toInt, ClickEvent(row(0).trim.toInt, row(1).trim.toInt, row(2).trim.toLong)))

}
```

Step 2 Continued

```
// Join adEvent and clickEvent DStreams and output DStream[queryId, (adEvent, clickEvent)]

val joinByQueryId = adEventDStream.join(clickEventDStream)

joinByQueryId.print()



// Transform DStream to DStream[adId, count(adId)] for each RDD

val countByAdId = joinByQueryId.map(rdd => (rdd._2._1.adId,1)).reduceByKey(_+_)
```

Step 2 Continued

```
// Update the state [adId, countCummulative(adId)] by values from the next RDDs

val updateFunc = (values: Seq[Int], state: Option[Int]) => {

          val currentCount = values.sum

          val previousCount = state.getOrElse(0)

          Some(currentCount + previousCount)

}

val countByAdIdCumm = countByAdId.updateStateByKey(updateFunc)

// Transform (key, value) pair to (adId, count(adId), countCummulative(adId))

val ad = countByAdId.join(countByAdIdCumm).map {case (adId, (count, cumCount)) => (adId, count, cumCount)}
```

Step 2 Continued

```
//Print report

ad.foreachRDD( ad => {

   println("%5s %10s %12s".format("AdId", "AdCount", "AdCountCumm"))

   ad.foreach( row => println("%5s %10s %12s".format(row._1,  row._2,  row._3)))

})
```

Step 3: Build Program

```
> cd kdd2016-streaming-tutorial

> mvn package
```

Step 4: Run Programs

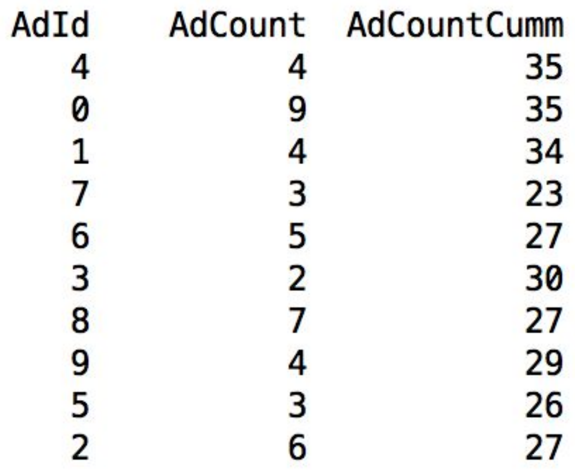
Run AdClickEventReplay in terminal window. It reads data from Ad and Click event files writes to Kafka.

```
> cd kdd2016-streaming-tutorial

> java -cp target/streamingtutorial-1.0.0-jar-with-dependencies.jar
example.AdClickEventReplay
```

Run AdEventJoiner in a new terminal window.

```
> cd kdd2016-streaming-tutorial

> ../spark-1.6.2-bin-hadoop2.6/bin/spark-submit --class example.AdEventJoiner
target/streamingtutorial-1.0.0-jar-with-dependencies.jar
```

Output



```
AdId    AdCount  AdCountCumm
   4          4           35
   0          9           35
   1          4           34
   7          3           23
   6          5           27
   3          2           30
   8          7           27
   9          4           29
   5          3           26
   2          6           27
```

Contact Us

Ashish Gupta - ahgupta@linkedin.com

https://www.linkedin.com/in/guptash

Neera Agarwal - neera8work@gmail.com

https://www.linkedin.com/in/neera-agarwal-21b9473

Additional Notes - Install Java (Mac 10.11)

- java -version

  java version "1.8.0_92"

If java does not exist, try

In .bash_profie add export JAVA_HOME=$(/usr/libexec/java_home)

(Install Java - https://java.com/en/download/help/mac_install.xml)

Additional Notes - Install Maven

- Check Maven on a terminal window
  - mvn -v
  - Apache Maven 3.2.5+
- Install Maven
  - brew install maven

OR if you do not have brew then do:

1. curl -O http://mirror.nexcess.net/apache/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz
2. tar -xzvf apache-maven-3.3.9-bin.tar.gz
3. In .bash_profile: export PATH=/<path>/apache-maven-3.3.9/bin:$PATH