CS x55: Distributed Systems [Spark Streaming]

Shrideep Pallickara
Computer Science
Colorado State University

Topics covered in this lecture

- Spark Streaming
  - Performance considerations
  - Example

As we have seen in the previous class

- Streams arrive continually (logs, transactions, tweets …)
- Batch systems (Hadoop MapReduce, Spark basic) wait for data to be at rest on disk
  - Streaming systems need to react
- Spark streaming extends the basic Spark. model

Spark streaming summary

- The incoming stream is broken up into discretized batches (DStreams)
- Each batch is amenable to processing like an RDD
  - E.g., `map`, `filter`, `reduce` still apply
- A `StreamingContext` defines batch duration and orchestrates execution
- Outputs update continuously as new data streams (or slides) in

PROCESSING TWITTER STREAMS USING SPARK

Let's look at a Twitter Hashtag pipeline in our example

- Capture live tweets from Twitter's API stream
- Extract text and identify hashtags (#...)
- Count hashtags over a 5-minute window; this is updated every second
- Display the top-10 trending hashtags in real time

Spark-streaming example [1/5]

- Step-by-step approach to finding the top 10 hashtags from a stream of tweets using counts [Every second there is an output over data from the last 300 seconds]
- Step-1: Create a SparkStream context and Twitter credential setup

```
SparkConf sparkConf = new SparkConf().setAppName("Spark-
streaming-twitter-trends");

/*
Twitter authentication details … [Not included here]
*/
//JavaStreamigContext
JavaStreamingContext jssc =
      new JavaStreamingContext(sparkConf, new Duration(1000));

//Discretized stream of tweets
JavaDStream<Status> twitterStream = (JavaDStream<Status>)
TwitterUtils.createStream(jssc);
```

Spark-streaming example [2/5]

- Step-2: Map Input DStream of `Status` to `String`

```
//Discretized stream of Strings
JavaDStream<String> statuses = twitterStream.map(
    new Function<Status, String>() {
    public String call(Status status) {
        return status.getText();
    }
  }
);

statuses.print();

//trigger the execution of code
jssc.start();
jssc.awaitTermination();
```

Spark-streaming example [3/5]

- Step-3: Stream of hashtags from stream of tweets

```
//Tokenize words from status
JavaDStream<String> wordsFromStatuses = statuses.flatMap(
    new FlatMapFunction<String, String>() {
       public Iterable<String> call(String input) {
          return Arrays.asList(input.split(" "));
       }
    }
);

//Extract hashtags
JavaDStream<String> hashTags = wordsFromStatuses.filter(
    new Function<String, Boolean>() {
       public Boolean call(String word) {
          return word.startsWith("#");
       }
    }
);
```

Spark-streaming example [4/5]

- Step-4: Count the hashtag over 5 min window

```
//Mapping to tuple of (hashtag,1) in order to count
JavaPairDStream<String, Integer> hashtagtuples = hashTags.mapToPair(
new PairFunction<String, String, Integer>() {
   public Tuple2<String, Integer> call(String input) {
      return new Tuple2<String, Integer>(input, 1);
   }
});
//Aggregating over window of 5 min and silde of 1s
JavaPairDStream<String, Integer> counts =
hashtagtuples.reduceByKeyAndWindow(
   new Function2<Integer, Integer, Integer>() {
      public Integer call(Integer int1, Integer int2) {
         return int1 + int2;
      }
   },  new Function2<Integer, Integer, Integer>() {
         public Integer call(Integer int1, Integer int2) {
            return int1 - int2;
         }
   },  new Duration(60 * 5 * 1000), new Duration(1 * 1000));
```

Spark-streaming example [5/5]

- Step-5: Find top 10 hashtags according to counts

```
JavaPairDStream<Integer, String> swapCounts = counts.mapToPair(
new PairFunction<Tuple2<String, Integer>, Integer, String>() {
  public Tuple2<Integer, String> call(Tuple2<String, Integer> input)
    return input.swap();
  }});
JavaPairDStream<Integer, String> sortedCount = swapCounts.transformToPair(
new Function<JavaPairRDD<Integer, String>,JavaPairRDD<Integer, String>>(){
public JavaPairRDD<Integer, String> call(JavaPairRDD<Integer, String> input)
throws Exception {
  return input.sortByKey(false);
}});
sortedCount.foreach(new Function<JavaPairRDD<Integer, String>, Void> () {
  public Void call(JavaPairRDD<Integer, String> rdd) {
    String out = "\nTrending hashtags:\n";
    for (Tuple2<Integer, String> t: rdd.take(10)) {
      out = out + t.toString() + "\n";
}
System.out.println(out);
return null;}});
```

The contents of this slide-set are based on the following references

- Processing Twitter Streams using Spark: https://databricks-training.s3.amazonaws.com/realtime-processing-with-spark-streaming.html