Spark Streaming

Mateusz Kopeć, Michał Okulewicz

Institute of Computer Science
Polish Academy of Sciences

Big Data
27 November 2014

Presentation Plan

What is Apache Spark™?

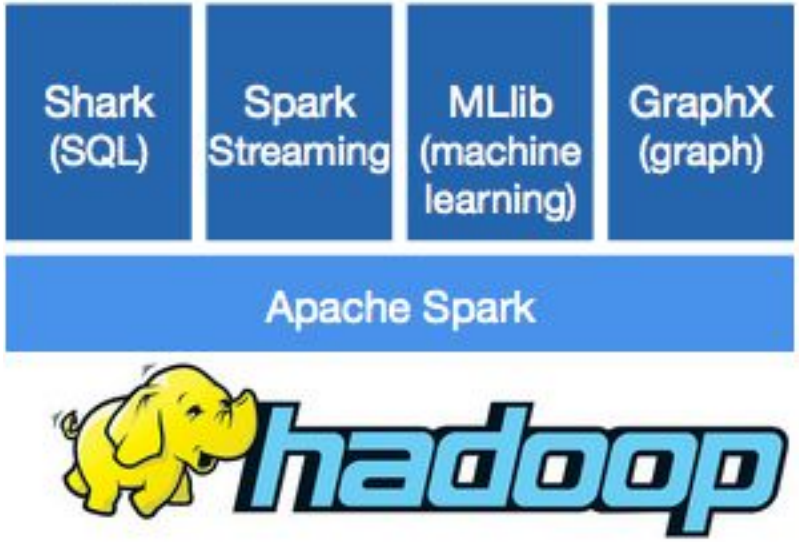
Apache Spark™

- distributed computations system
- not only MapReduce applications
- supports in-memory operations (Resilient Distributed Dataset)
- may use HDFS

APIs

- Scala
- Java
- python

Spark architecture



Why do we want Spark Streaming?

Many applications require fast processing of lots of live data, for example:

- Fraud detection
- Financial market analysis
- On-line surveillance
- Early earthquakes detection

We need a system, which:

- Scales to hundreds of nodes
- Achieves second-scale latencies
- Recovers from node failures
- Integrates batch and online processing

State of the art

In 2012, existing frameworks couldn't do both:

- low-latency stream processing (ex. Storm)
- high latency batch processing (ex. Hadoop)

Why do we want Spark Streaming?

Many applications require fast processing of lots of live data, for example:

- Fraud detection
- Financial market analysis
- On-line surveillance
- Early earthquakes detection

We need a system, which:

- Scales to hundreds of nodes
- Achieves second-scale latencies
- Recovers from node failures
- Integrates batch and online processing

State of the art

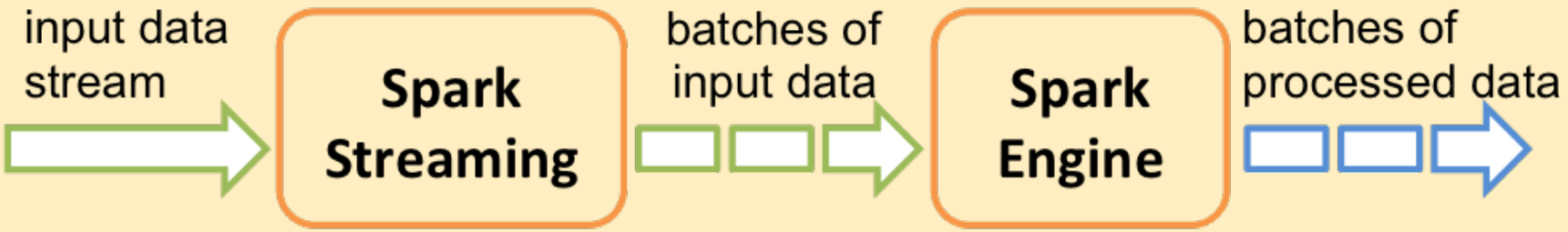
In 2012, existing frameworks couldn't do both:

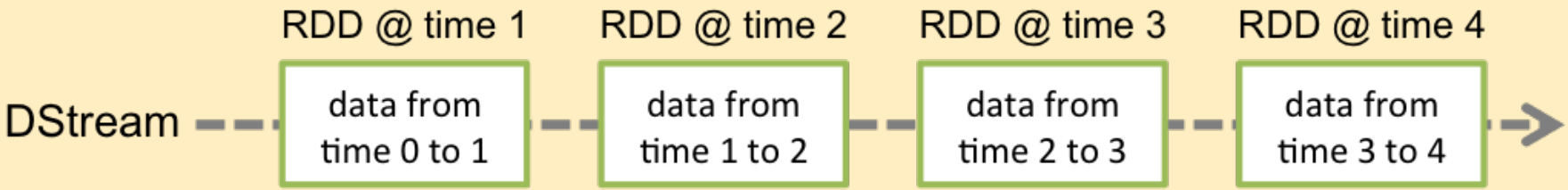
- low-latency stream processing (ex. Storm)
- high latency batch processing (ex. Hadoop)

What is Spark Streaming?

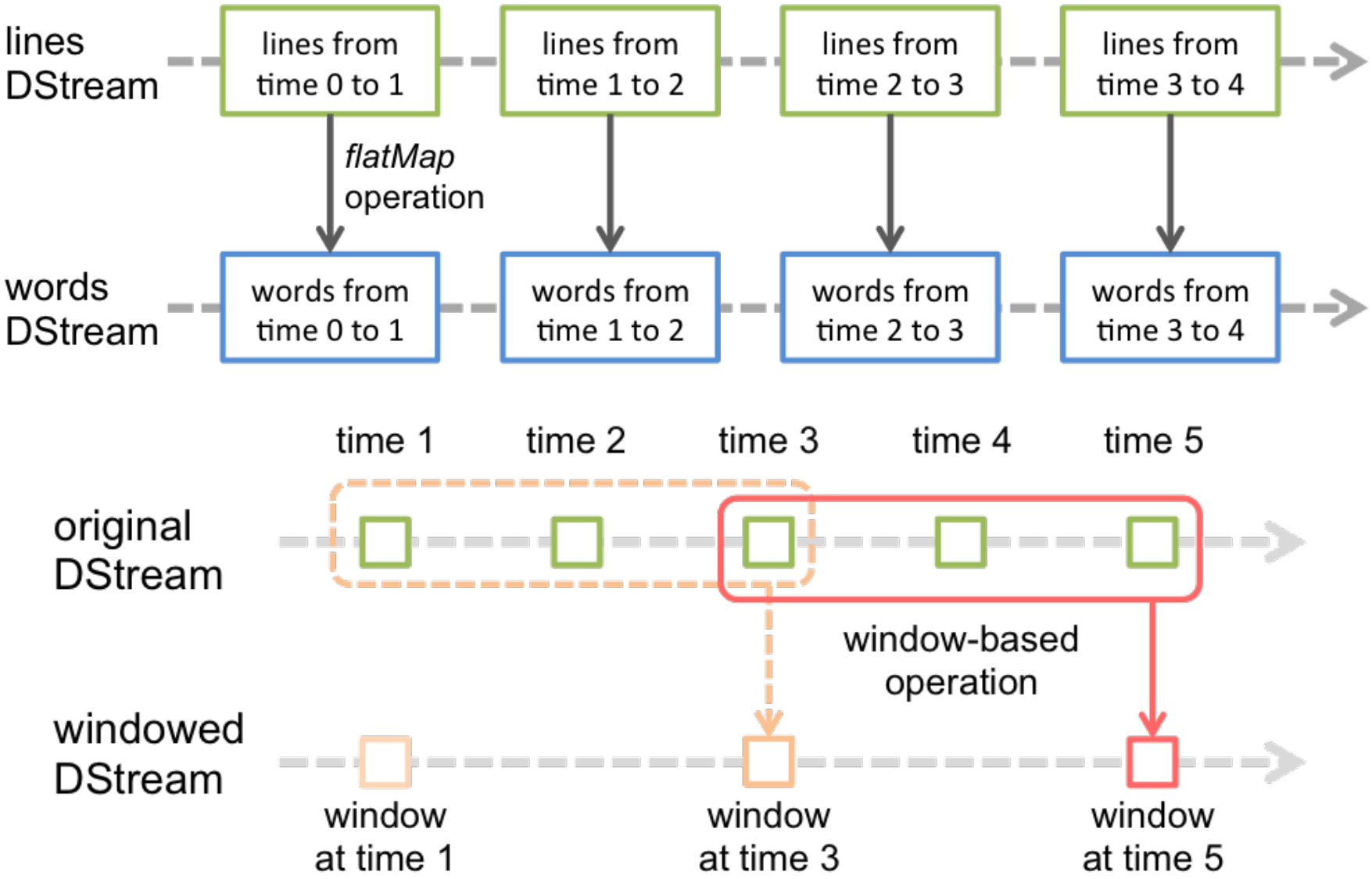
Spark Streaming

- subproject of Apache Spark™
- allows for real-time distributed stream processing
- utilizes an idea called Discretized Stream (DStream)
- no python API, only Java and Scala





What is Spark Streaming?



First example I

Data producer

Generates next integer every 100ms

Data analyser

Counts all distinct numbers

Data analyser code I

Init spark stream

```
SparkConf sparkConf = new SparkConf().setAppName("Socket");

JavaStreamingContext ssc =
    new JavaStreamingContext(sparkConf, new Duration(1000));

JavaReceiverInputDStream<String> lines =
    ssc.socketTextStream(host, port));
```

Data analyser code II

Transform stream

```
JavaDStream<String> words = lines
    .flatMap(line -> Lists.newArrayList("␣".split(line)))
    .map(word -> Long.valueOf(word))
    .mapToPair(number -> {
        Set<Long> set = new HashSet<>();
        set.add(number);
        return new Tuple2<String,Set<Long>>("distinct", set);
    })
    .updateStateByKey((values, state) -> {
        Set<Long> result = state.or(new HashSet<Long>());
        for (Set<Long> set : values)
            result.addAll(set);
        return Optional.of(result);
    });
```

Data analyser code III

Print stream

```
stream.foreachRDD(rdd -> {
    if (rdd.count() != 1) {
        System.out.println("Empty␣RDD");
        return null;
    }
    List<Tuple2<String, Set<Long>>> collect = rdd.collect();
    Set<Long> set = collect.get(0)._2;
    System.out.println("Distinct:␣" + set.size());
    return null;
});
```

How to run Spark? I

Running master on Linux

- Run `sbin/start-master.sh`
- Check in browser if `http://localhost:8080` is available

Running worker on Linux

- Get precompiled Spark 1.1.0 for Hadoop 1.x from `/home/2012/m.okulewicz/spark` and unpack it
- If necessary edit: `conf/spark-env.sh` and add location of `JAVA_HOME`
- Run `sbin/start-slave.sh 1 spark://phd22.phd.ipipan.waw.pl`
- Check in browser if `http://localhost:8081` is available and master points to `phd22.phd.ipipan.waw.pl`

How to run Spark? II

Running task on Linux

- Run:

```
./bin/spark-submit
--class
pl.waw.ipipan.phd.mkopec.sparkReceiver.SocketReceiver
--master spark://phd22.phd.ipipan.waw.pl:7077
--executor-memory 20G
--total-executor-cores 100
/path/to/jar.jar localhost 9999 1000 1
```

Example stream processing task

Data producer

Retrieves stream of tweets containing specified keywords

Data analyser

Counts most frequent words in tweets

Data analyser code I

Init spark stream

```
SparkConf sparkConf = new SparkConf().setAppName("Twitter");

JavaStreamingContext ssc =
    new JavaStreamingContext(sparkConf, new Duration(1000));

JavaReceiverInputDStream<String> lines =
   TwitterUtils.createStream(ssc,
       new Array[] {"Polska", "Poland"});
```

Data analyser code II

Transform stream

```
JavaPairDStream<String, Long> counts = twitterStream
    .flatMap(tweet -> Arrays.asList(" ".split(tweet.getText())
    .filter(word -> {
        sleep(1000);
        return !StringUtils.isBlank(word);
    })
    .mapToPair(word -> new Tuple2<String, Long>(word, 1L))
    .reduceByKey((x, y) -> x + y)
    .updateStateByKey((values, state) -> {
        Long total = state.or(0L);
        for (Long v : values)
            total += v;
        return Optional.of(total);
    });
```

Data analyser code III

Print stream

```
stream.foreachRDD(rdd -> {
    if (rdd.count() == 0) {
        System.out.println("Empty␣RDD");
        return null;
    }
    List<Tuple2<String, Long>> list = rdd.collect();
    Collections.sort(list, ...);

    System.out.println("␣Top␣10␣words");
    for (Tuple2<String, Long> tuple :
        list.subList(0, Math.min(list.size(), 10)))
        System.out.println(tuple._1 + "\t" + tuple._2);
    return null;
});
```

How to run this task?

Running task on Linux

- Run:

```
./bin/spark-submit
--class
pl.waw.ipipan.phd.mkopec.sparkReceiver.TwitterReceiver
--master spark://phd22.phd.ipipan.waw.pl:7077
--executor-memory 20G
--total-executor-cores 100
/path/to/jar.jar localhost 1000 1 Polska,Poland
```

Bibliography I

https://spark.apache.org/docs/0.9.0/streaming-programming guide.html.
Streaming Programming Guide, 2014.

Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica.
Discretized Streams: Fault-tolerant Streaming Computation at Scale.
In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 423–438, New York, NY, USA, 2013. ACM.

Matei Zaharia, Tathagata Das, Haoyuan Li, Scott Shenker, and Ion Stoica.
Discretized Streams: An Efficient and Fault-tolerant Model for Stream Processing on Large Clusters.
In *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing*, HotCloud'12, pages 10–10, Berkeley, CA, USA, 2012. USENIX Association.