Introducing Apache Spark

Lesson 1

Hortonworks

Learning Objectives

- After you complete this lesson you should be able to:
  - Describe the origin of Apache Spark
  - Understand the rapid rate of growth of the Spark
  - Recognize some of the use cases for Spark
  - Describe some major differences between Spark and MR

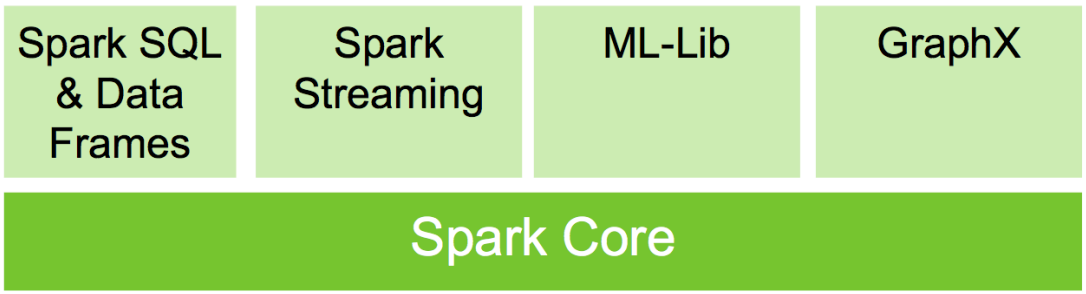
Hortonworks

What is Apache Spark?

- Apache open source project, originally developed at AmpLab at UC-Berkeley
  - Started as a grad research project in 2009
- A general data processing engine, focused on in-memory distributed computing use-cases
- APIs in Scala, Python and Java
  - Recently API for R was introduced

Hortonworks

The Spark ecosystem



Hortonworks

Why Spark?

- Elegant Developer APIs: Data Frames/SQL, Machine Learning, Graph algorithms and streaming
  - Scala, Python, Java and R
  - Single environment for importing, transforming, and exporting data
- In-memory computation model
  - Effective for iterative computations
- High level API
  - Allows users to focus on the business logic and not internals

Hortonworks

Why Spark cont.

- Supports wide variety of workloads
  - Mllib for Data Scientists
  - Spark SQL for Data Analysts
  - Spark Streaming for micro batch use cases
  - Spark Core, SQL and Streaming for Data Processing Applications
- Integrated fully with Hadoop and creating a highly available tool, with minimal outside intervention

Hortonworks

Who uses Spark!?

- NASA JPL
  - Deep Space Network
- eBay
  - Analysts are clustering sellers together
- Conviva
  - Video stream health statistics
- Yahoo
  - News story personalization

Hortonworks

Spark vs MapReduce

- Higher level API
- In-memory data storage
  - Up to 100x performance improvement

pyspark

```
text_file = spark.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

Java MapReduce

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {

 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
 }

 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
 }

 public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

        Job job = new Job(conf, "wordcount");

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);
 }
}
```


Hortonworks

Spark Growth is Massive

- One of the largest open source projects
  - Last release had over 1000 commits and 230 developers contributing
- On average release a .X version every 3 months
- Currently at spark 1.5.1
  - March 2015 – Spark SQL Dataframes Release (v1.3)
  - Dec 2014 – Spark Streaming on Python Released (v1.2)

Hortonworks

Spark and HDP

- HDP 2.3.2 – Spark 1.4.1
- HDP 2.2.8 – Spark 1.3.1
- HDP 2.2.4 – Spark 1.2.1

- For this class we'll focus on 1.3.1, with some references to 1.4.1

Hortonworks

Conclusion

- Spark started at AmpLab at UC – Berkeley
- It takes advantage of in-memory processing to offer huge processing speed up
- High Level API allows users to worry less about the plumbing and more on business logic
- Spark is moving FAST

Hortonworks