By Tsai Li Ming

PyData + Spark Meetup (SG) - 17 Nov 2015

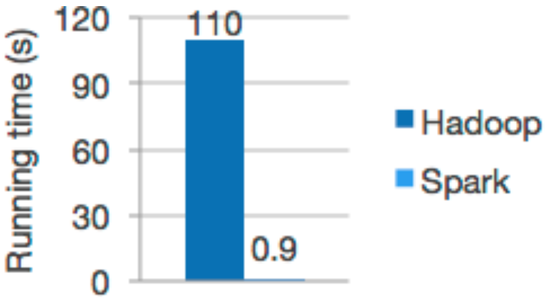
http://about.me/tsailiming

Presentation and source codes are available here:
http://github.com/tsailiming/pydatasg-17Nov2015

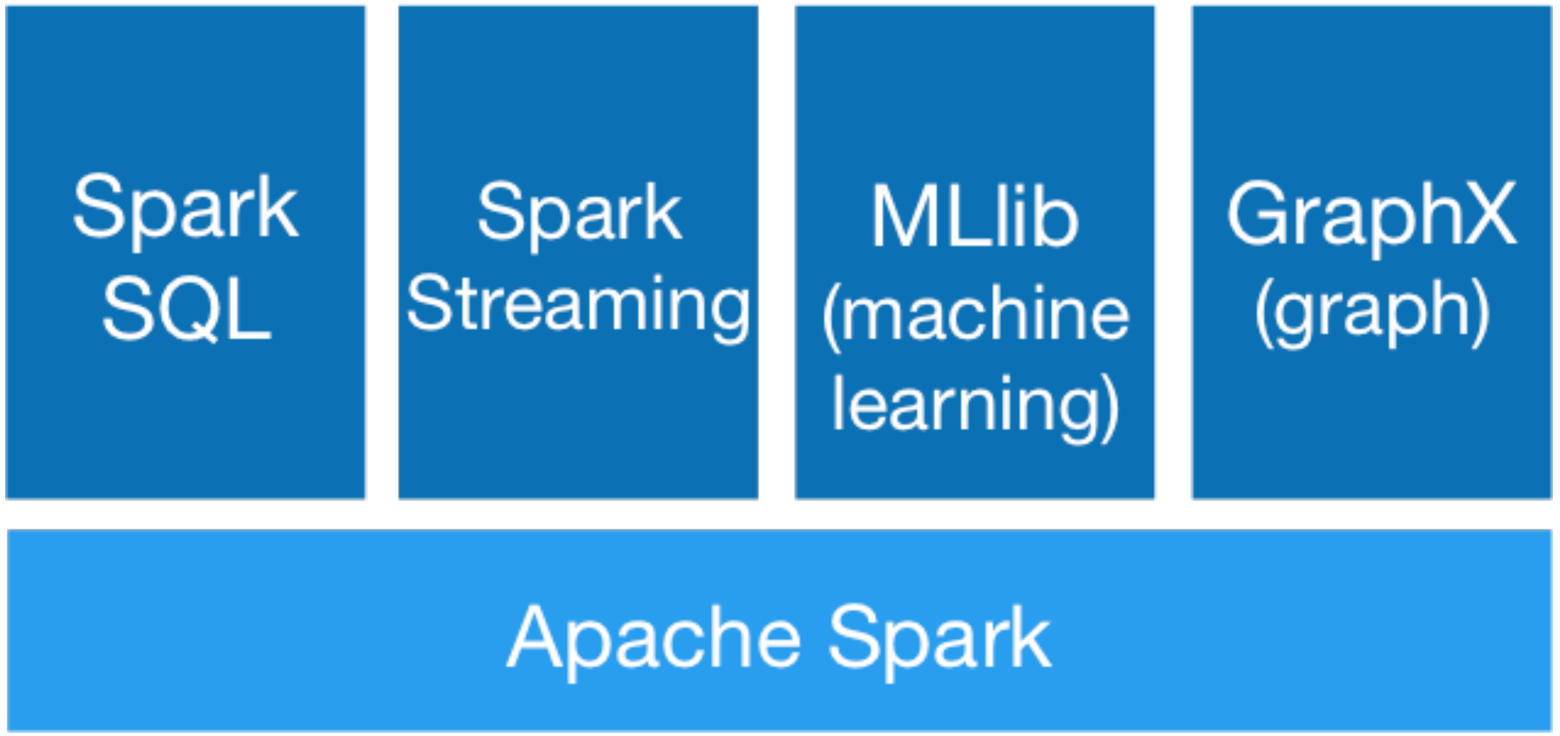
What is Spark?

What is Spark?

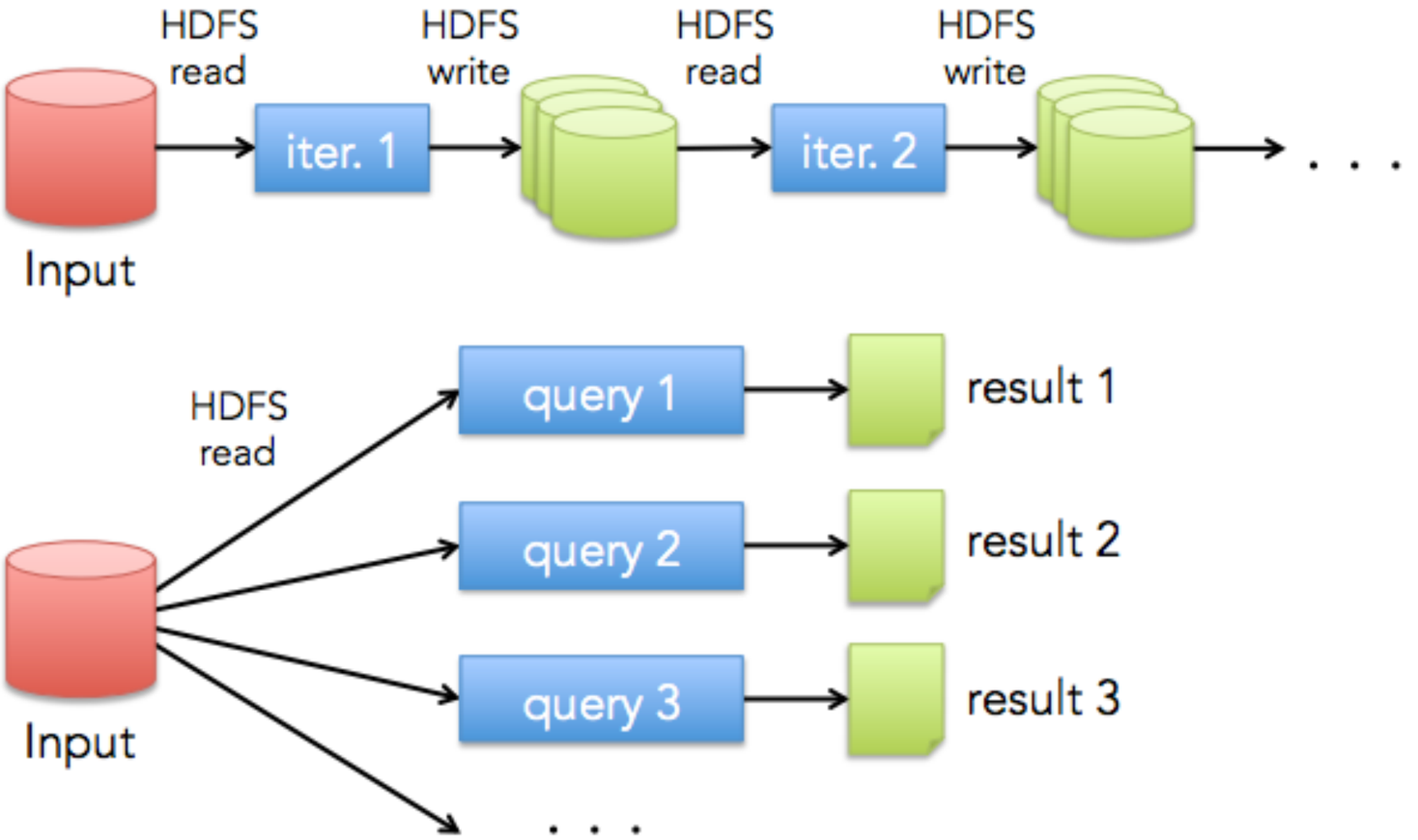
- Developed at UC Berkley in 2009. Open sourced in 2010.
- Fast and general engine for large-scale data processing
- Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk
- Multi-step Directed Acrylic Graphs (DAGs). Many stages compared to just Hadoop Map and Reduce only.
- Rich Scala, Java and Python APIs. R too!
- Interactive Shell
- Active development



Spark Stack


Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

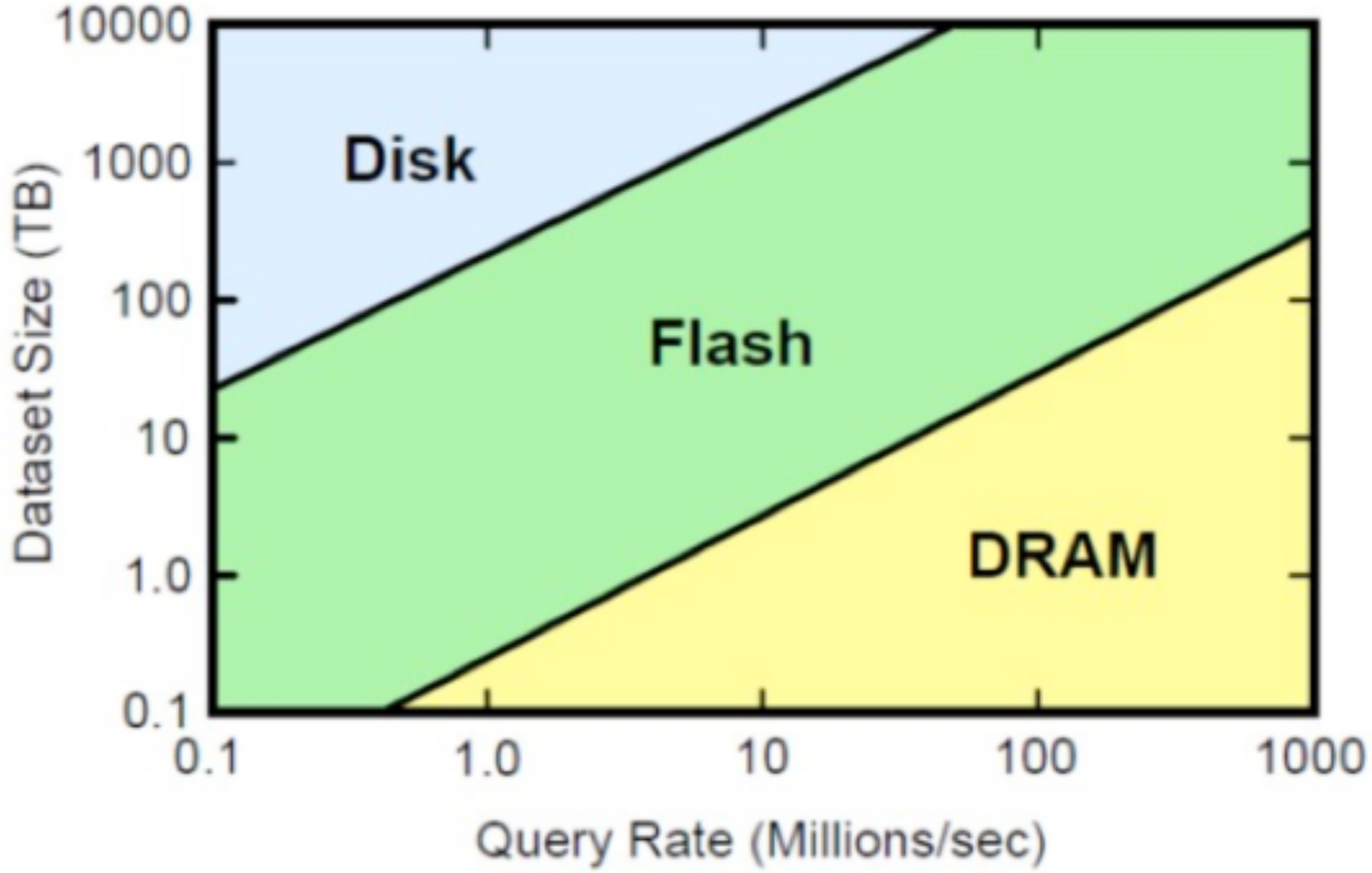
Data Sharing in MapReduce



Slow due to data replication and disk I/O

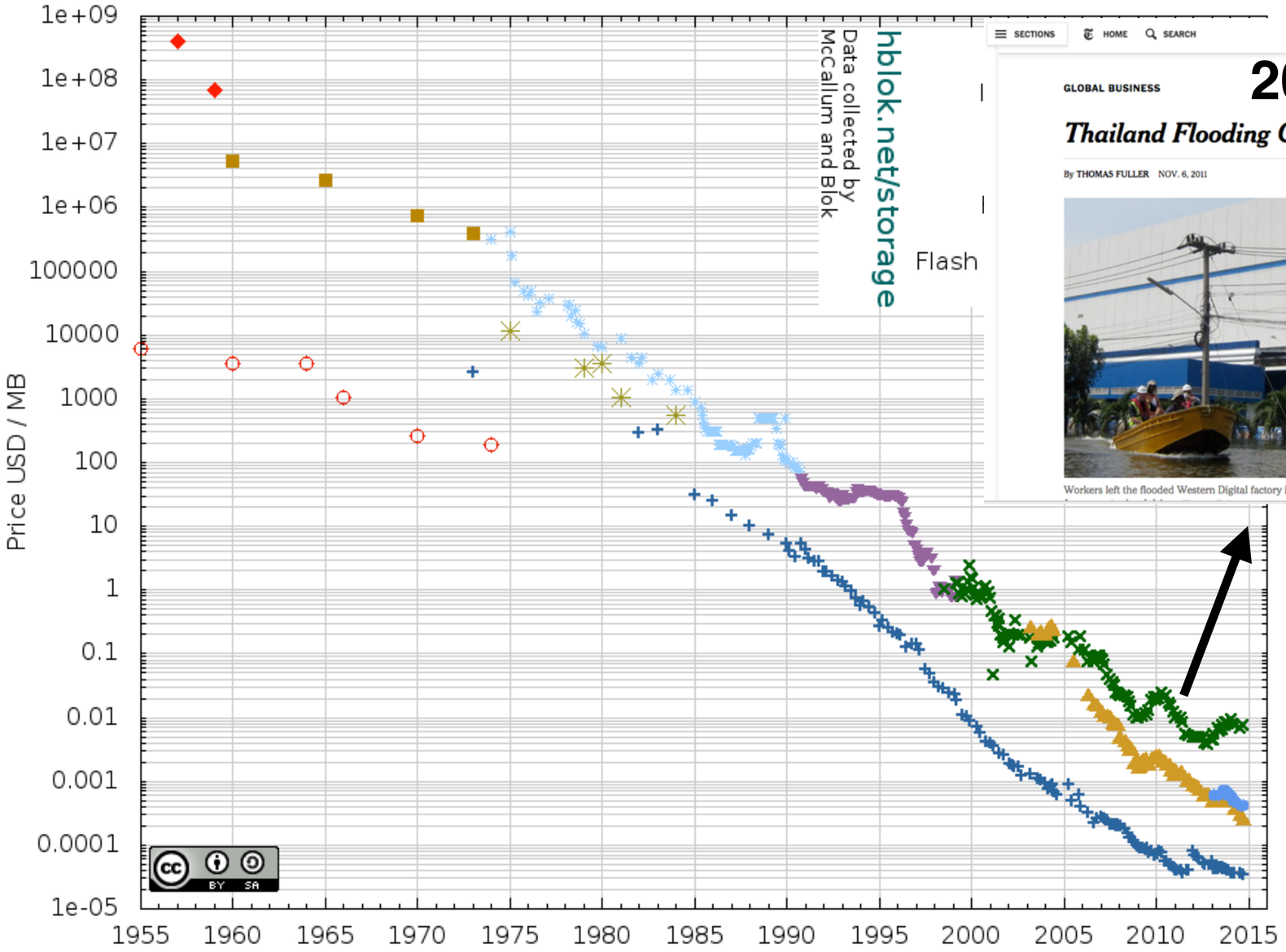
http://www.slideshare.net/jamesskillsmatter/zaharia-sparkscaladays2012

Speed matters



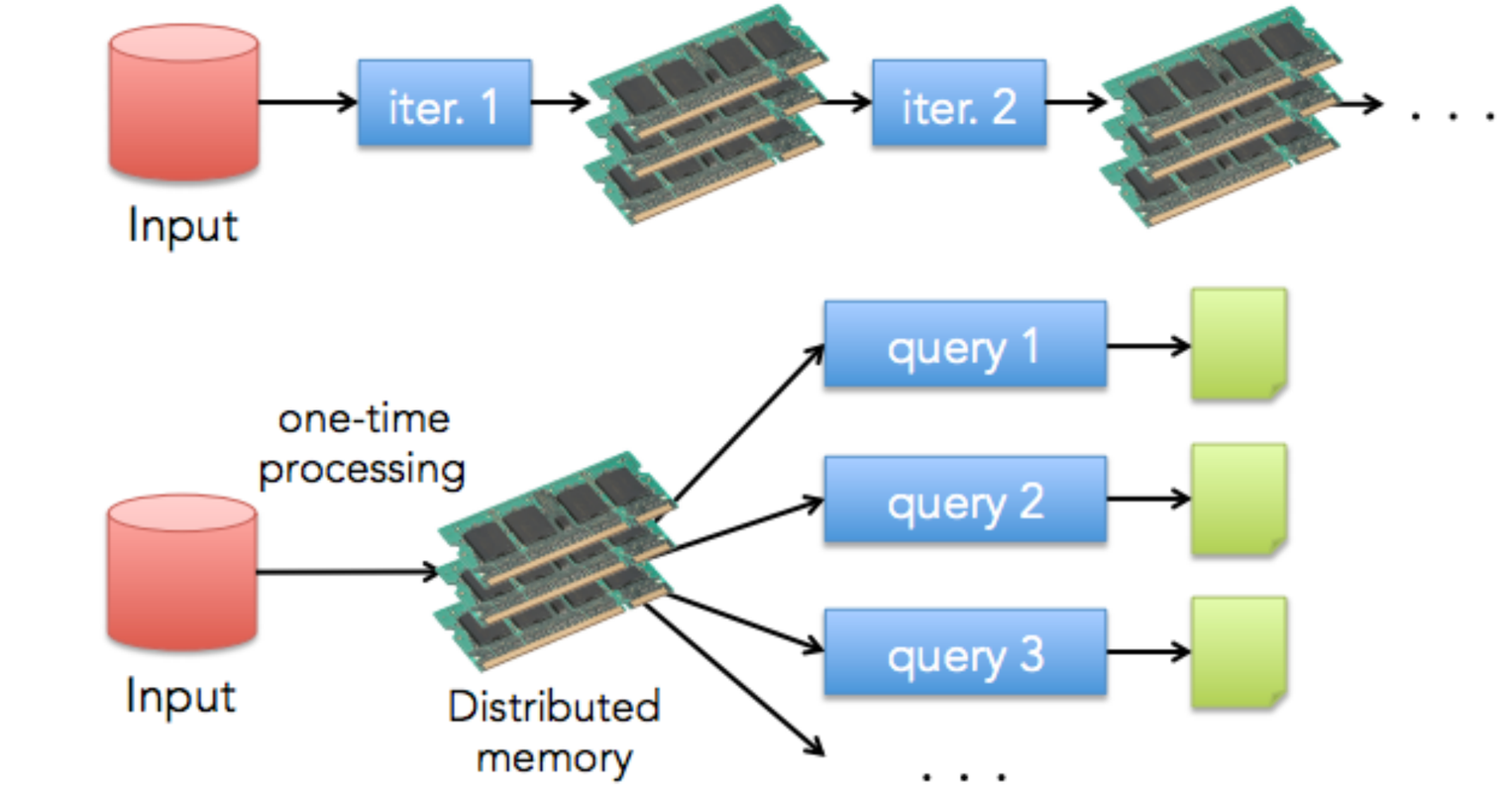
http://web.stanford.edu/~ouster/cgi-bin/papers/ramcloud.pdf

Historical Cost of Computer Memory and Storage



http://hblok.net/blog/storage/

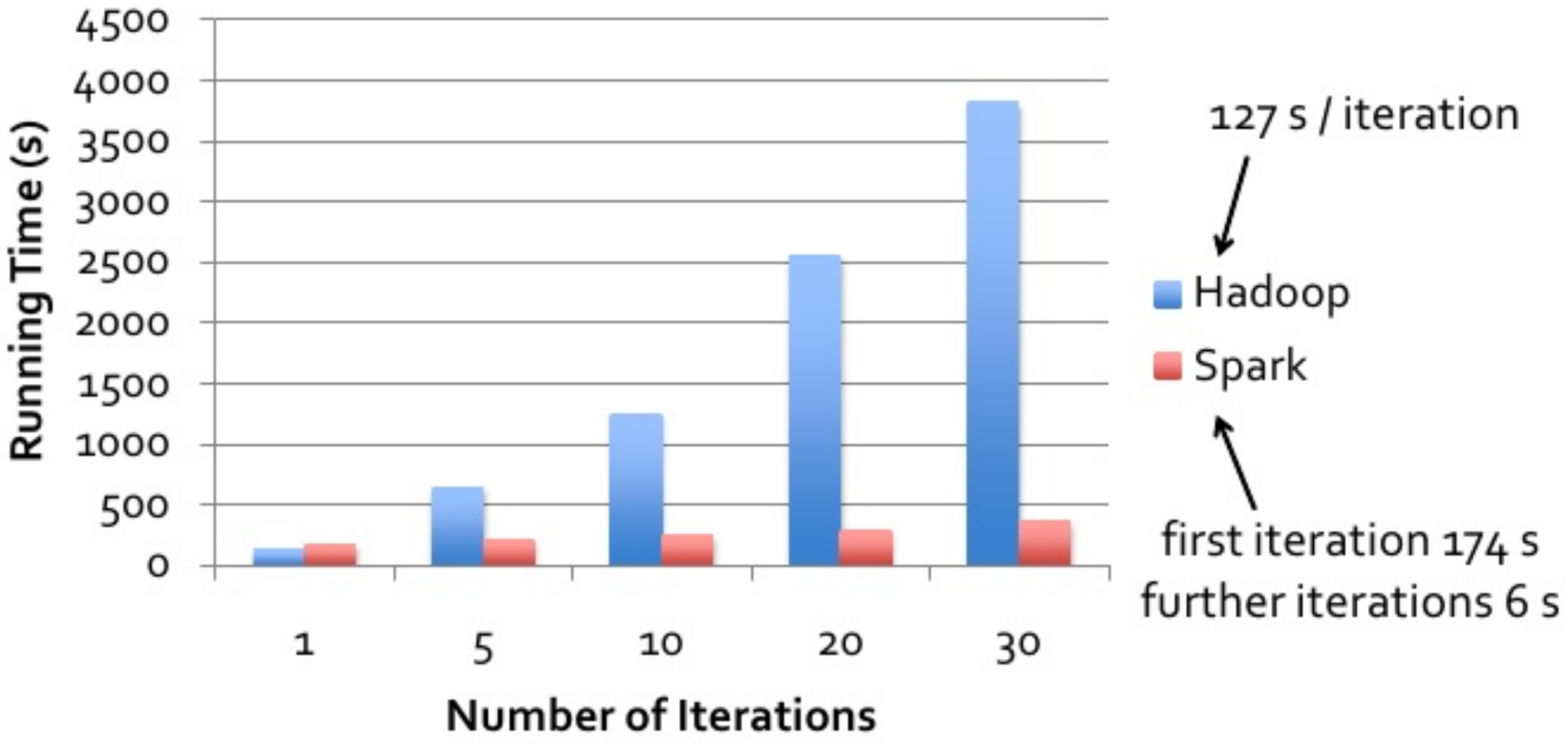
What We'd Like



10-100× faster than network and disk

http://www.slideshare.net/jamesskillsmatter/zaharia-sparkscaladays2012

Logistic Regression Performance



http://www.slideshare.net/jamesskillsmatter/zaharia-sparkscaladays2012

How Spark works

Resilient Distributed Datasets (RDDs)

- Basic abstraction in Spark. Fault-tolerant collection of elements that can be operated on in parallel
- RDDs can be created from local file system, HDFS, Cassandra, HBase, Amazon S3, SequenceFiles, and any other Hadoop InputFormat.
- Different levels of caching: MEMORY_ONLY, MEMORY_AND_DISK, DISK_ONLY, OFF_HEAP, etc
- Rich APIs for Transformations and Actions
- Data Locality: PROCESS_LOCAL -> NODE_LOCAL -> RACK_LOCAL

RDD Operations

map	flatMap	sortByKey
filter	union	reduce
sample	join	count
groupByKey	distinct	saveAsTextFile
reduceByKey	mapValues	first

Spark Example

Wordcount Example

Hadoop 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.mapred.*;
import org.apache.hadoop.util.*;

public class WordCount {

	public static class Map extends MapReduceBase implements 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, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {
			String line = value.toString();
			StringTokenizer tokenizer = new StringTokenizer(line);
			while (tokenizer.hasMoreTokens()) {
				word.set(tokenizer.nextToken());
				output.collect(word, one);
			}
		}
	}

	public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {
		public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
			int sum = 0;
			while (values.hasNext()) {
				sum += values.next().get();
			}
			output.collect(key, new IntWritable(sum));
		}
	}

	public static void main(String[] args) throws Exception {
		JobConf conf = new JobConf(WordCount.class);
		conf.setJobName("wordcount");

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

		conf.setMapperClass(Map.class);
		//conf.setCombinerClass(Reduce.class);
		conf.setReducerClass(Reduce.class);

		conf.setInputFormat(TextInputFormat.class);
		conf.setOutputFormat(TextOutputFormat.class);

		FileInputFormat.setInputPaths(conf, new Path(args[0]));
		FileOutputFormat.setOutputPath(conf, new Path(args[1]));

		JobClient.runJob(conf);
	}
}
```

Spark Scala

```
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

Spark Python

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

Spark SQL and Dataframe Example

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

# Create the DataFrame
df = sqlContext.read.json("people.json")

# Show the content of the DataFrame
df.show()
## age  name
## null Michael
## 30   Andy
## 19   Justin

# Print the schema in a tree format
df.printSchema()
## root
## |-- age: long (nullable = true)
## |-- name: string (nullable = true)

# Select only the "name" column
df.select("name").show()
## name
## Michael
## Andy
## Justin
```

```
# Select everybody, but increment the age by 1
df.select(df['name'], df['age'] + 1).show()
## name    (age + 1)
## Michael null
## Andy    31
## Justin  20

# Select people older than 21
df.filter(df['age'] > 21).show()
## age name
## 30  Andy

# Count people by age
df.groupBy("age").count().show()
## age  count
## null 1
## 19   1
## 30   1
```

Notebooks for Spark

Apache Zeppelin

andypetrella/spark-notebook
(forked from Scala notebook)

Actual Demo


Strata+
Hadoop
WORLD

Thank You!