The New Analytics Toolbox

Going beyond Hadoop

Robbie Strickland

DevNexus 2015

whoami

Robbie Strickland

Director, Software Engineering

linkedin.com/in/robbiestrickland

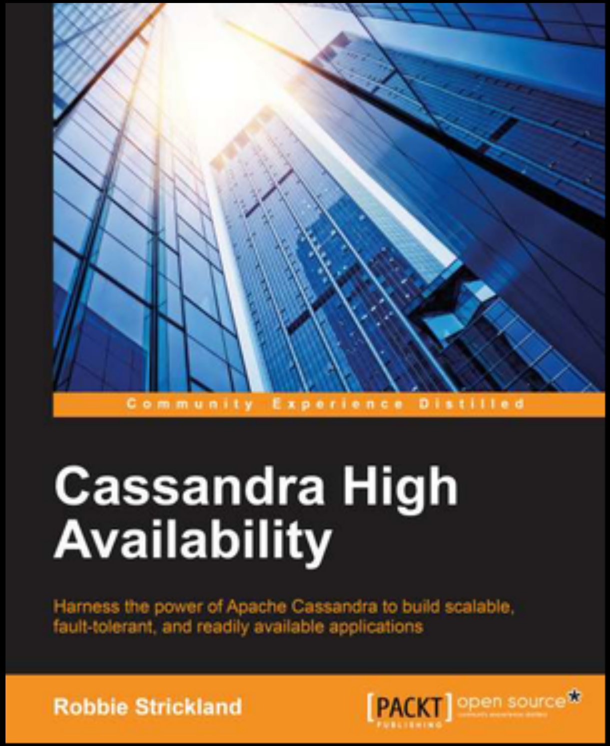
@rs_atl

rostrickland@gmail.com

The Weather Channel

whoami

- Contributor: core Cassandra, Java driver, Spark driver, Hive driver, other stuff
- DataStax MVP
- User since 2010 (0.5 release)
- Author, *Cassandra High Availability*
- Founder, ATL Cassandra Users



Thanks to ...

Helena Edelson

DataStax Engineering

@helenaedelson

Weather @ scale

- ~10 billion API requests per day
- 4 AWS regions
- Over 100 million app users
- Lots of data to analyze!

Agenda

- Tool landscape
- Spark vs. Hadoop
- Spark overview
- Spark + Cassandra
- A typical Spark application
- Spark SQL
- Getting set up
- Demo
- Spark Streaming
- Task distribution
- Language comparison
- Questions

Hadoop works fine, right?

Hadoop works fine, right?

- It’s 11 years old (!!)

Hadoop works fine, right?

- It’s 11 years old (!!)
- It’s relatively slow and inefficient

Hadoop works fine, right?

- It’s 11 years old (!!)
- It’s relatively slow and inefficient
- … because everything gets written to disk

Hadoop works fine, right?

- It’s 11 years old (!!)
- It’s relatively slow and inefficient
- … because everything gets written to disk
- Writing mapreduce code sucks

Hadoop works fine, right?

- It’s 11 years old (!!)
- It’s relatively slow and inefficient
- … because everything gets written to disk
- Writing mapreduce code sucks
- Lots of code for even simple tasks

Hadoop works fine, right?

- It’s 11 years old (!!)
- It’s relatively slow and inefficient
- … because everything gets written to disk
- Writing mapreduce code sucks
- Lots of code for even simple tasks
- Boilerplate

Hadoop works fine, right?

- It’s 11 years old (!!)
- It’s relatively slow and inefficient
- … because everything gets written to disk
- Writing mapreduce code sucks
- Lots of code for even simple tasks
- Boilerplate
- Configuration isn’t for the faint of heart

Landscape has changed ...

Lots of new tools available:

- Cloudera Impala
- Apache Drill
- Proprietary (Splunk, keen.io, etc.)
- Spark / Spark SQL
- Shark

Landscape has changed ...

Lots of new tools available:

- Cloudera Impala
- Apache Drill
- Proprietary (Splunk, keen.io, etc.)
- Spark / Spark SQL
- Shark

MPP queries for Hadoop data

Landscape has changed ...

Lots of new tools available:

- Cloudera Impala
- Apache Drill
- Proprietary (Splunk, keen.io, etc.) Varies by vendor
- Spark / Spark SQL
- Shark

Landscape has changed ...

Lots of new tools available:

- Cloudera Impala
- Apache Drill
- Proprietary (Splunk, keen.io, etc.)
- Spark / Spark SQL Generic in-memory analysis
- Shark

Landscape has changed ...

Lots of new tools available:

- Cloudera Impala
- Apache Drill
- Proprietary (Splunk, keen.io, etc.)
- Spark / Spark SQL
- Shark   Hive queries on Spark, replaced by Spark SQL

Spark wins?

Spark wins?

- Can be a Hadoop replacement

Spark wins?

- Can be a Hadoop replacement
- Works with any existing InputFormat

Spark wins?

- Can be a Hadoop replacement
- Works with any existing InputFormat
- Doesn’t require Hadoop

Spark wins?

- Can be a Hadoop replacement
- Works with any existing InputFormat
- Doesn’t require Hadoop
- Supports batch & streaming analysis

Spark wins?

- Can be a Hadoop replacement
- Works with any existing InputFormat
- Doesn’t require Hadoop
- Supports batch & streaming analysis
- Functional programming model

Spark wins?

- Can be a Hadoop replacement
- Works with any existing InputFormat
- Doesn’t require Hadoop
- Supports batch & streaming analysis
- Functional programming model
- Direct Cassandra integration

Spark vs. Hadoop

MapReduce:

```java
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.*;

import org.apache.cassandra.hadoop.cql3.*;
import org.apache.cassandra.hadoop.ConfigHelper;
import org.apache.cassandra.utils.ByteBufferUtil;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.util.*;
import org.apache.log4j.Logger;

public class CheckinsByHour extends Configured implements Tool {

    private static final Logger _logger = Logger.getLogger(CheckinsByHour.class);
    private static final String _minTimestamp = "minTimestamp";

    public static void main(String[] args) throws Exception {
        ToolRunner.run(new Configuration(), new CheckinsByHour(), args);
        System.exit(0);
    }
```

MapReduce:

```java
public int run(String[] args) throws Exception {

    _logger.info("Starting TestMR");

    final String cassHost = args[0];
    final int numReducers = Integer.parseInt(args[1]);
    final String keyspace = args[2];
    final String inputCF = args[3];
    final String outputCF = args[4];
    final long minTimestamp = Long.parseLong(args[5]);

    //set up job
    _logger.info("Setting up job");
    final Job job = new Job(getConf(), "test");
    final Configuration conf = job.getConfiguration();

    conf.set(_minTimestamp, Long.toString(minTimestamp));
    job.setJarByClass(CheckinsByHour.class);
    job.setNumReduceTasks(numReducers);
```

MapReduce:

```java
    //set up cassandra
    _logger.info("Setting up Cassandra");
    ConfigHelper.setInputRpcPort(conf, "9160");
    ConfigHelper.setInputInitialAddress(conf, cassHost);
    ConfigHelper.setInputColumnFamily(conf, keyspace, inputCF);
    ConfigHelper.setInputPartitioner(conf, "Murmur3Partitioner");
    CqlConfigHelper.setInputCQLPageRowSize(conf, "1000000");
    CqlConfigHelper.setInputWhereClauses(conf, "WHERE time > " + minTimestamp);
    _logger.info("Read consistency = " + ConfigHelper.getReadConsistencyLevel(conf));
    ConfigHelper.setOutputColumnFamily(conf, keyspace, outputCF);
    CqlConfigHelper.setOutputCql(conf, "UPDATE " + keyspace + "." + outputCF + " SET count=?");
    ConfigHelper.setOutputInitialAddress(conf, cassHost);
    ConfigHelper.setOutputPartitioner(conf, "Murmur3Partitioner");
    _logger.info("Write consistency = " + ConfigHelper.getWriteConsistencyLevel(conf));

    //set up input
    _logger.info("Configuring input");
    job.setMapperClass(TestMapper.class);
    job.setInputFormatClass(CqlPagingInputFormat.class);

    //cass output
    _logger.info("Configuring output");
    job.setReducerClass(Reduce.class);
    job.setCombinerClass(Combiner.class);
    job.setOutputFormatClass(CqlOutputFormat.class);
    job.setMapOutputKeyClass(LongWritable.class);
    job.setMapOutputValueClass(IntWritable.class);
    job.setOutputKeyClass(ByteBuffer.class);
    job.setOutputValueClass(List.class);

    job.waitForCompletion(true);
    return 0;

}
```

MapReduce:

```
public static class TestMapper extends Mapper<Map<String, ByteBuffer>, Map<String, ByteBuffer>, LongWritable, IntWritable> {
    private final IntWritable one = new IntWritable(1);
    private LongWritable outKey = new LongWritable();
    private long minTimestamp = -1;

    public void map(Map<String, ByteBuffer> keys, Map<String, ByteBuffer> columns, Context context) throws IOException, Interrupte
        if (minTimestamp == -1) minTimestamp = Long.parseLong(context.getConfiguration().get(_minTimestamp));
        long timestamp = ByteBufferUtil.toLong(keys.get("time"));
        if (timestamp >= minTimestamp) {
            long hour = Math.round(timestamp/(60*60*1000));
            outKey.set(hour);
            context.write(outKey, one);
        }
    }
}
```

MapReduce:

```
public static class Combiner extends Reducer<LongWritable, IntWritable, LongWritable, IntWritable> {
    private IntWritable outCount = new IntWritable();

    public void reduce(LongWritable key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
        int count = 0;
        for (IntWritable val : values) count += val.get();
        outCount.set(count);
        context.write(key, outCount);
    }
}
```

MapReduce:

```
public static class Reduce extends Reducer<LongWritable, IntWritable, Map<String, ByteBuffer>, List<ByteBuffer>> {
    private Map<String, ByteBuffer> keys;

    protected void setup(org.apache.hadoop.mapreduce.Reducer.Context context) throws IOException, InterruptedException {
        keys = new LinkedHashMap<String, ByteBuffer>();
    }

    public void reduce(LongWritable key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
        int count = 0;
        for (IntWritable val : values) count += val.get();
        long timestamp = key.get() * 60 * 60000;
        long day = (timestamp / 86400000) * 86400000;
        int hour = (int) ((timestamp % 86400000) / 3600000);
        keys.put("day", ByteBufferUtil.bytes(day));
        keys.put("hour", ByteBufferUtil.bytes(hour));
        context.write(keys, countToList(count));
    }

    private List<ByteBuffer> countToList(long count) {
        List<ByteBuffer> variables = new ArrayList<~>();
        variables.add(ByteBufferUtil.bytes(count));
        return variables;
    }
}
```

Spark (angels sing!):

```
import com.datastax.driver.spark._
import org.apache.spark.{SparkConf, SparkContext}

object CheckinsByHourSpark extends App {
  val master = args(0)
  val cHost = args(1)
  val minTimestamp = args(2).toLong

  val conf = new SparkConf().set("cassandra.connection.host", cHost)
  val sc = new SparkContext(master, "wxcheckin", conf)

  val chickens = sc.cassandraTable[(String, Long)]("wxcheckin", "geocheckin_perm").select("user", "time").where("time >= ?", minTimestamp)

  val grouped = chickens.map { case (_, time) =>
    val day = (time / 86400000) * 86400000
    val hour = ((time % 86400000) / 3600000).toInt
    (day, hour)
  }.groupBy(identity)

  val output = grouped.map { case ((day, hour), vals) => (day, hour, vals.length.toLong) }
  output.saveToCassandra("wxcheckin", "count", Seq("day", "hour", "count"))
}
```

What is Spark?

What is Spark?

- In-memory cluster computing

What is Spark?

- In-memory cluster computing
- 10-100x faster than MapReduce

What is Spark?

- In-memory cluster computing
- 10-100x faster than MapReduce
- Collection API over large datasets

What is Spark?

- In-memory cluster computing
- 10-100x faster than MapReduce
- Collection API over large datasets
- Scala, Python, Java

What is Spark?

- In-memory cluster computing
- 10-100x faster than MapReduce
- Collection API over large datasets
- Scala, Python, Java
- Stream processing

What is Spark?

- In-memory cluster computing
- 10-100x faster than MapReduce
- Collection API over large datasets
- Scala, Python, Java
- Stream processing
- Supports any existing Hadoop input / output format

What is Spark?

- Native graph processing via GraphX

What is Spark?

- Native graph processing via GraphX
- Native machine learning via MLlib

What is Spark?

- Native graph processing via GraphX
- Native machine learning via MLlib
- SQL queries via SparkSQL

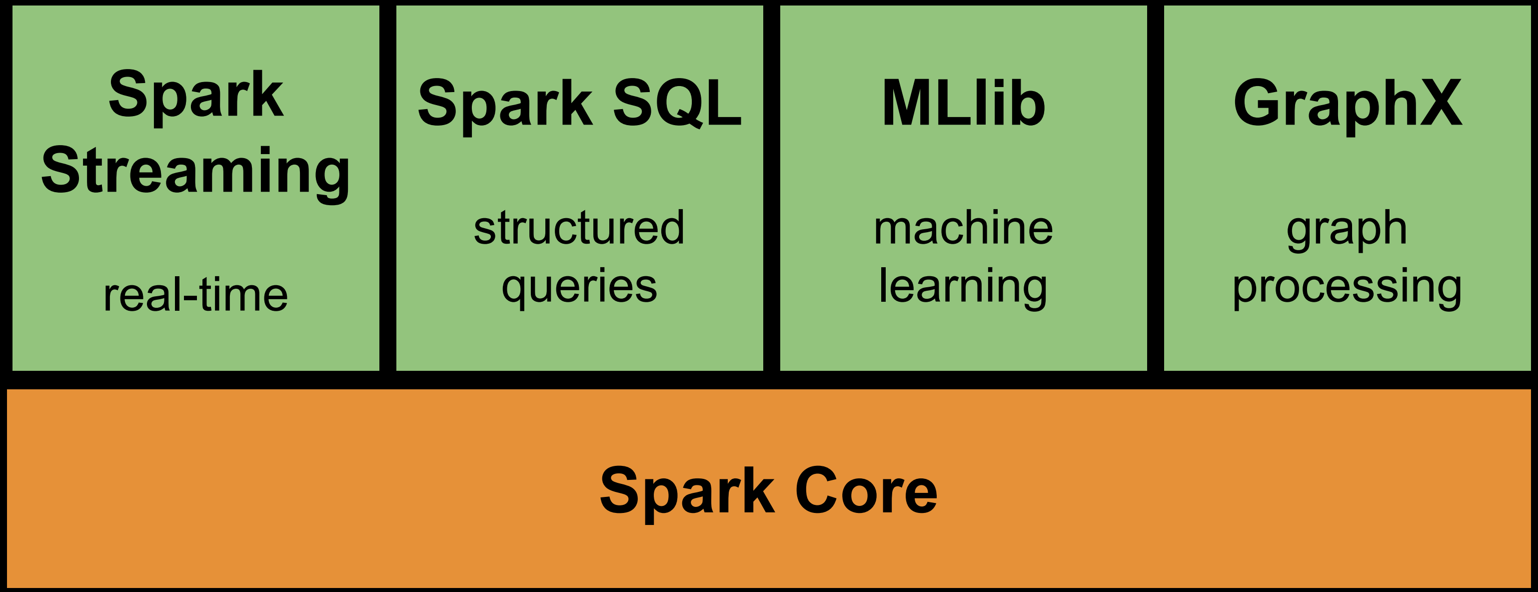
What is Spark?

- Native graph processing via GraphX
- Native machine learning via MLlib
- SQL queries via SparkSQL
- Works out of the box on EMR

What is Spark?

- Native graph processing via GraphX
- Native machine learning via MLlib
- SQL queries via SparkSQL
- Works out of the box on EMR
- Easily join datasets from disparate sources

Spark Components



Deployment Options

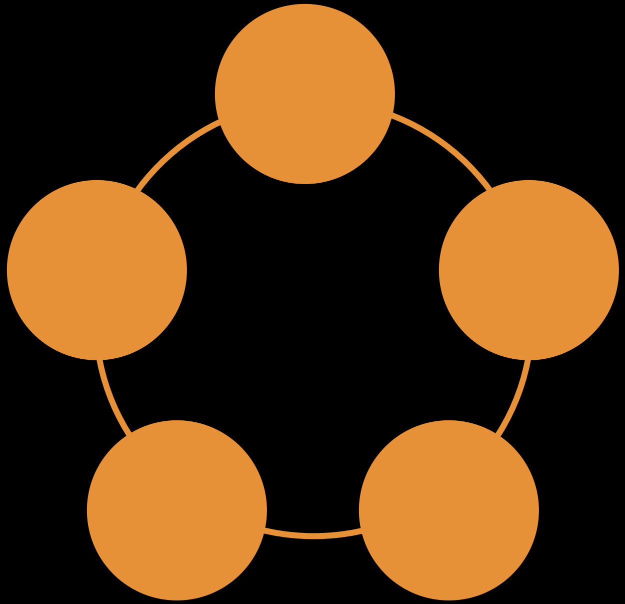
3 cluster manager choices:

- **Standalone** - included with Spark & easy to set up
- **Mesos** - generic cluster manager that can also handle MapReduce
- **YARN** - Hadoop 2 resource manager

Spark Word Count

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

Cassandra



Why Cassandra?

It’s fast:

- No locks
- Tunable consistency
- Sequential R/W

Why Cassandra?

It scales (linearly):

- Peer-to-peer (decentralized)
- DHT
- Read/write to any node
- Largest cluster = 75,000 nodes!

Why Cassandra?

It's fault tolerant:

- Automatic replication
- Masterless (i.e. no SPOF)
- Failed nodes replaced with ease
- Multi data center

Why Cassandra?

It’s perfect for analysis:

- Unstructured & semi-structured data
- Partition aware
- Multi-DC replication
- Sharding is automatic
- Natural time-series support

Why Cassandra?

It’s easy to use:

- Familiar CQL syntax
- Light administrative burden
- Simple configuration

Spark on Cassandra

Spark on Cassandra

- Direct integration via DataStax driver - cassandra-driver-spark on github

Spark on Cassandra

- Direct integration via DataStax driver - cassandra-driver-spark on github
- No job config cruft

Spark on Cassandra

- Direct integration via DataStax driver - cassandra-driver-spark on github
- No job config cruft
- Supports server-side filters (where clauses)

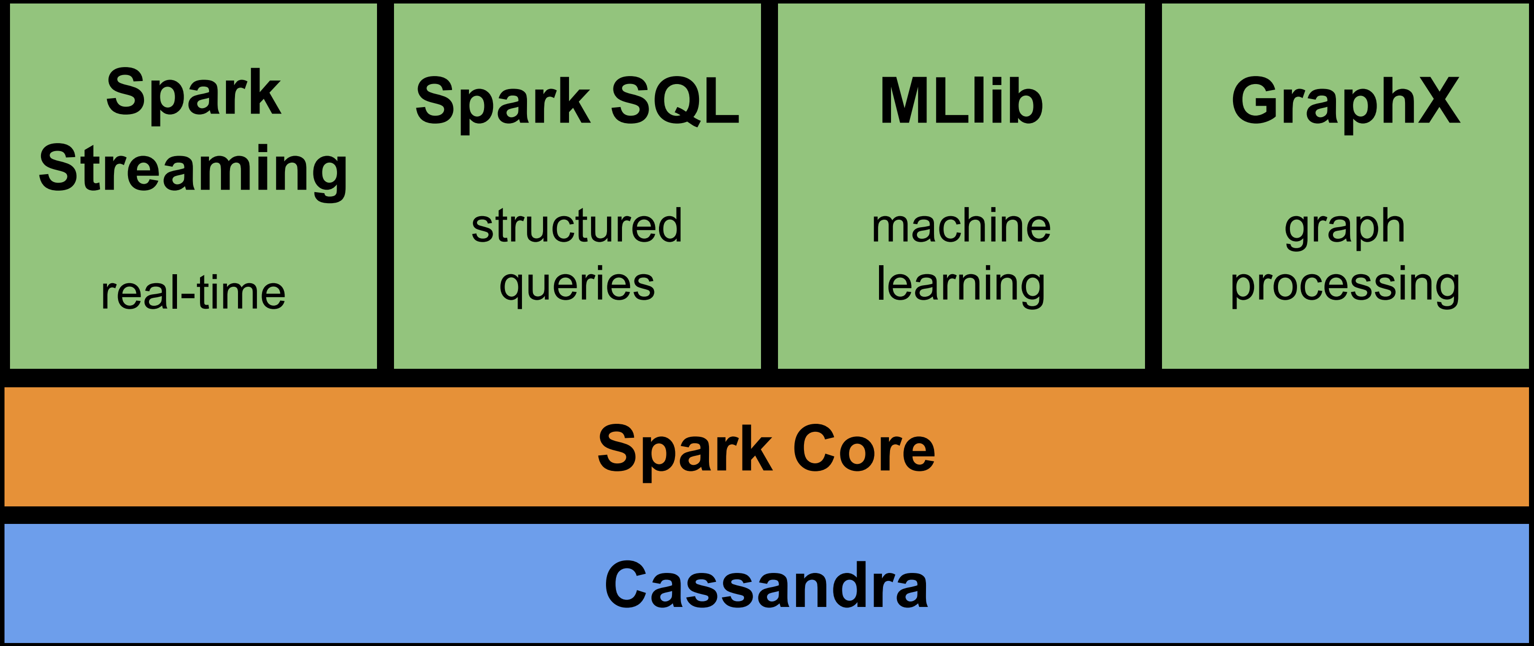
Spark on Cassandra

- Direct integration via DataStax driver - cassandra-driver-spark on github
- No job config cruft
- Supports server-side filters (where clauses)
- Data locality aware

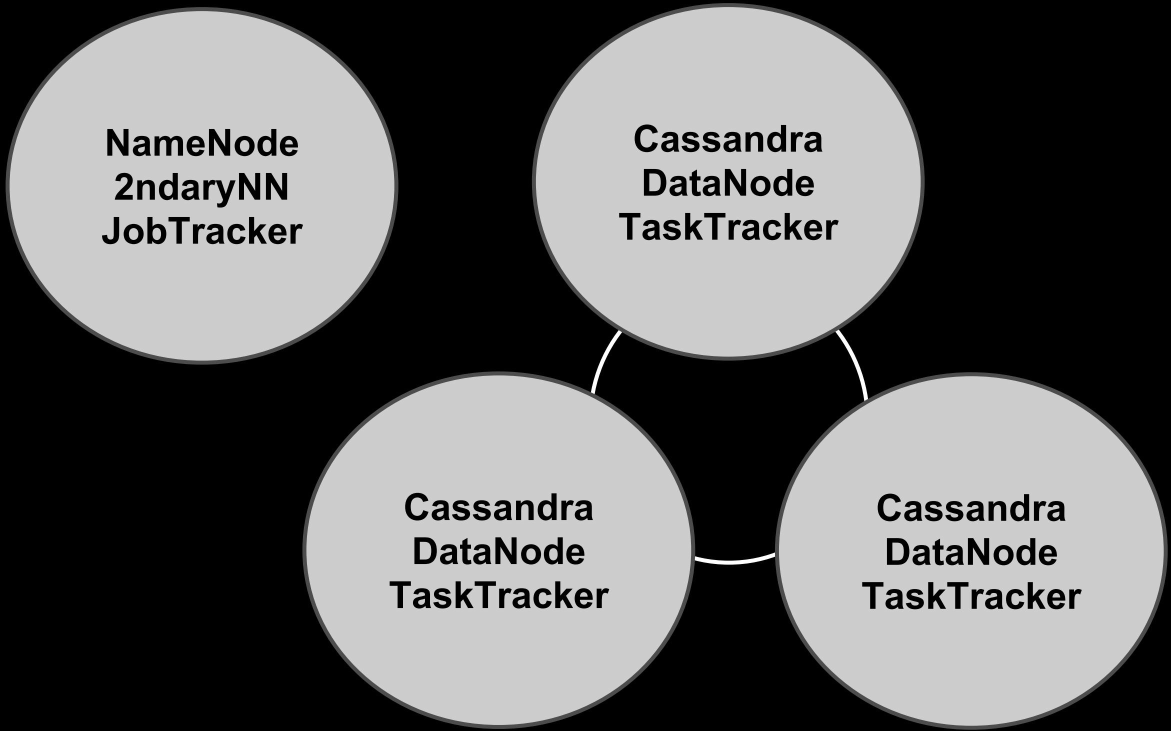
Spark on Cassandra

- Direct integration via DataStax driver - cassandra-driver-spark on github
- No job config cruft
- Supports server-side filters (where clauses)
- Data locality aware
- Uses HDFS, CassandraFS, or other distributed FS for checkpointing

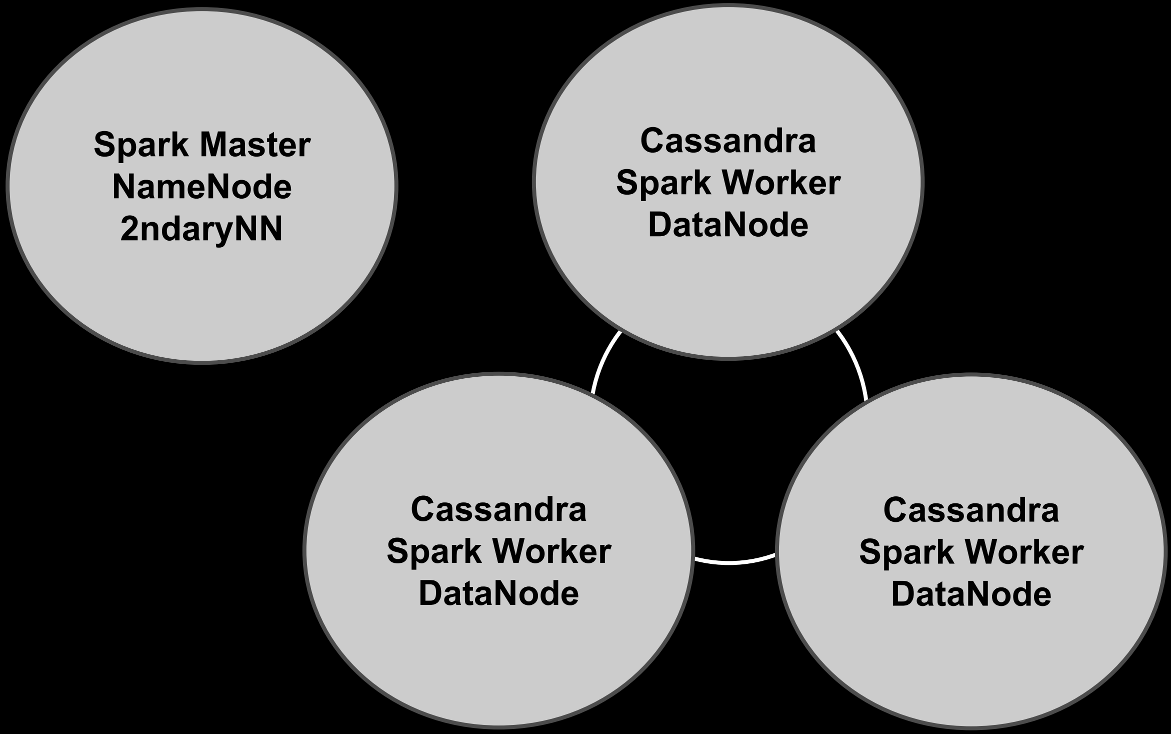
Spark on Cassandra



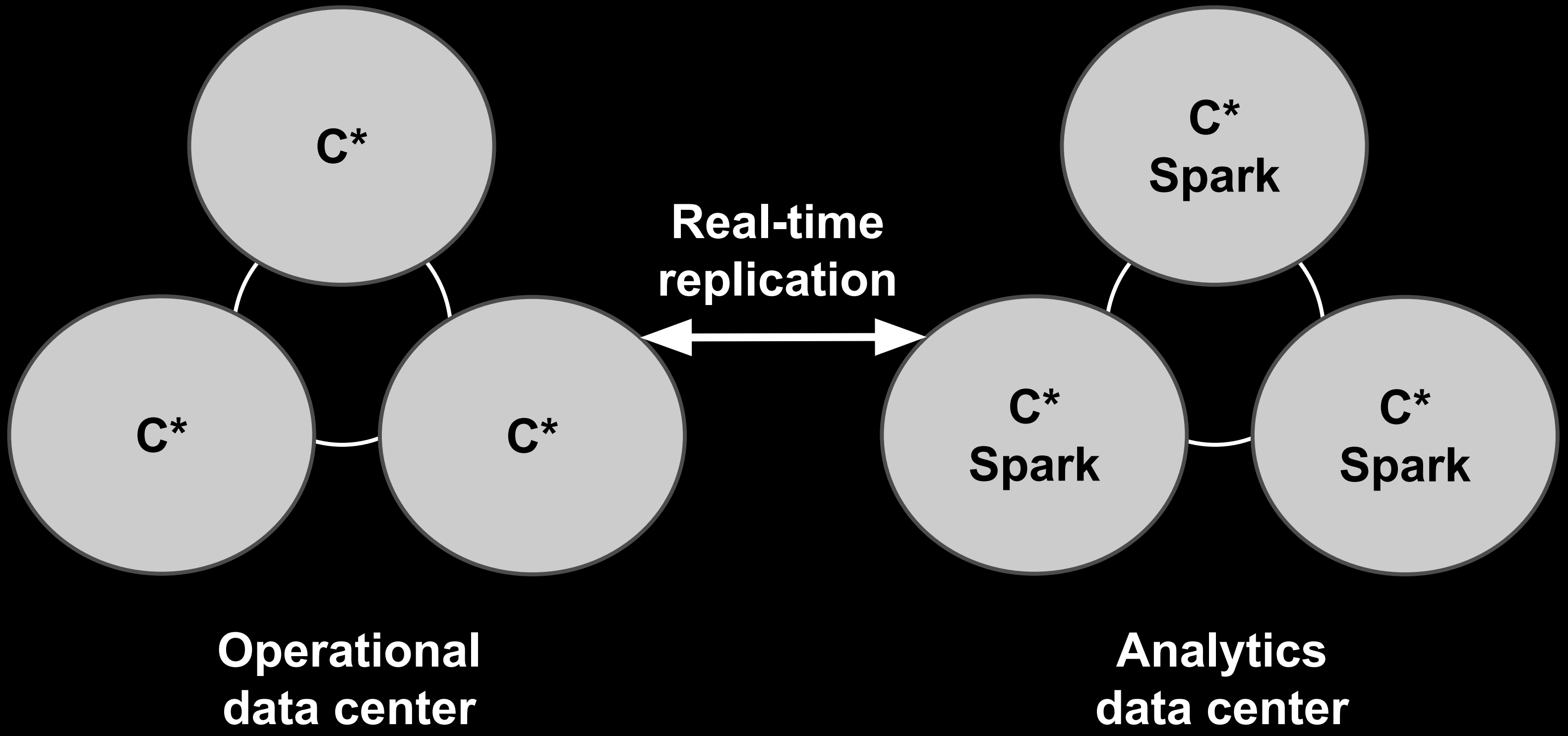
Cassandra with Hadoop



Cassandra with Spark (using HDFS)



Online analytics



A Typical Spark Application

A Typical Spark Application

- SparkContext + SparkConf

A Typical Spark Application

- SparkContext + SparkConf
- Data Source to RDD[T]

A Typical Spark Application

- SparkContext + SparkConf
- Data Source to RDD[T]
- Transformations/Actions

A Typical Spark Application

- SparkContext + SparkConf
- Data Source to RDD[T]
- Transformations/Actions
- Saving/Displaying

Resilient Distributed Dataset (RDD)

Resilient Distributed Dataset (RDD)

- A **distributed** collection of items

Resilient Distributed Dataset (RDD)

- A **distributed** collection of items
- Transformations
  - Similar to those found in scala collections
  - Lazily processed

Resilient Distributed Dataset (RDD)

- A **distributed** collection of items
- Transformations
  - Similar to those found in Scala collections
  - Lazily processed
- Can recalculate from any point of failure

RDD Transformations vs Actions

Transformations:

Produce new RDDs

Actions:

Require the materialization of the records to produce a value

RDD Transformations/Actions

Transformations:

filter, map, flatMap, collect(λ):RDD[T], distinct, groupBy, subtract, union, zip, reduceByKey ...

Actions:

collect:Array[T], count, fold, reduce ...

Resilient Distributed Dataset (RDD)

```
val numOfAdults = persons.filter(_.age > 17).count()
```

Transformation

Action

Example

```scala
case class Person(id: String, fname: String, lname: String, age: Int)

val persons = sc.cassandraTable[Person]("test", "persons")
val adults = persons.filter(_.age > 17)
adults.saveToCassandra("test", "adults")
```

Spark SQL

- Provides SQL access to **structured data**
  - Existing **RDDs**
  - **Hive** warehouses (uses existing metastore, SerDes and UDFs)
  - **JDBC/ODBC** - use existing BI tools to query large datasets

Spark SQL RDD Example

```
val persons = sc.cassandraTable[Person]("test", "persons").registerAsTable("persons")
val adults = sql("SELECT * FROM persons WHERE age > 17")
adults.foreach(t => println(s"Adult: ${t(1)} ${t(2)}"))
```

Getting set up

- Download **Spark 1.2.x**
- Download **Cassandra 2.1.x**
- Add the **spark-cassandra-connector** to your project

```
"com.datastax.spark" % "spark-cassandra-connector_2.10" % "1.2.0-alpha3"
```

Running applications

```
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master local[8] \
  /path/to/examples.jar
```

Running applications

```
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master spark://192.168.1.1:7077 \
  --executor-memory 20G \
  --total-executor-cores 100 \
  /path/to/examples.jar
```

Demo

Spark Streaming

Spark Streaming

Spark Streaming

- Creates RDDs from stream source on a defined interval

Spark Streaming

- Creates RDDs from stream source on a defined interval
- Same ops as “normal” RDDs

Spark Streaming

- Creates RDDs from stream source on a defined interval
- Same ops as “normal” RDDs
- Supports a variety of sources

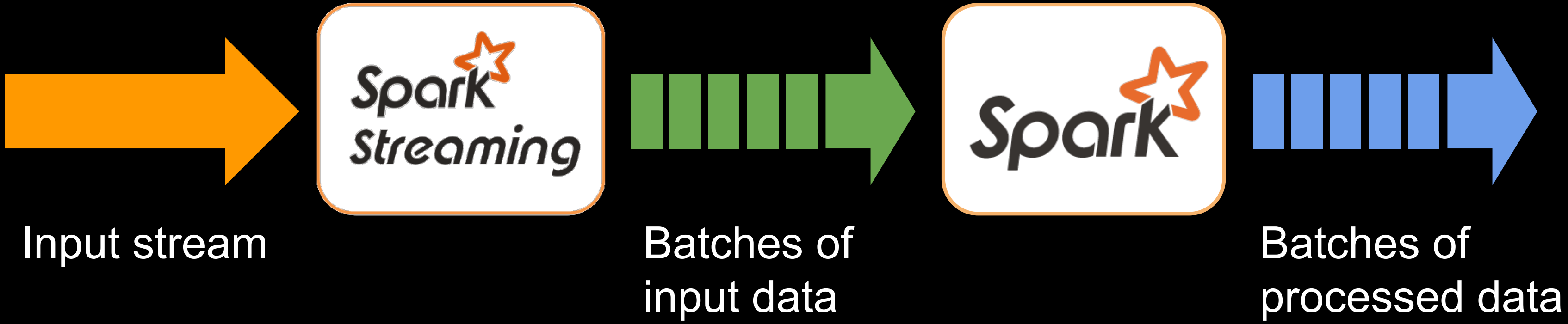
Spark Streaming

- Creates RDDs from stream source on a defined interval
- Same ops as “normal” RDDs
- Supports a variety of sources
- Exactly once message guarantee

Spark Streaming - Use Cases

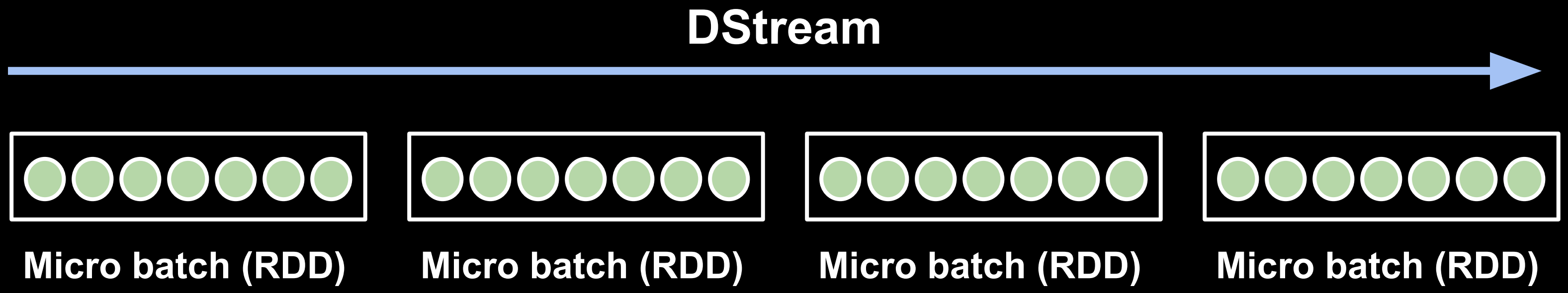
Applications	Sensors	Web	Mobile
Intrusion detection	Malfunction detection	Site analytics	Network analytics
Fraud detection	Dynamic process optimization	Recommendations	Location based advertising
Log processing	Supply chain planning	Sentiment analysis	...

Spark Streaming



Spark Streaming

- DStream = continuous sequence of micro batches
- Each batch is an RDD
- Interval is configurable



Spark Streaming

```scala
val happyWords = Set("happy", "love", "laugh", "excited")
val whitespace = """\s+""".r

TwitterHelper.configureTwitterCredentials()

val tweets: ReceiverInputDStream[Status] = TwitterUtils.createStream(ssc, None)
val statuses: DStream[String] = tweets.map(status => status.getText)

def filterTweetsWithWords(filterWords: Set[String], statuses: DStream[String]) = statuses.filter { status =>
  !whitespace.split(status).find(word => happyWords.contains(word.toLowerCase)).isEmpty
}

val happyTweets = filterTweetsWithWords(happyWords, statuses)

happyTweets.foreachRDD(rdd => println(s"${rdd.take(10).mkString("\n")}\n\n"))
```

The Output

```
@camerondallas made this account for you. Wish you could notice me💖 please #CallMeCam I love you x6

RT @k_tolls: id rather get my heart broken than never know what it is to be madly in love
I just be gettin my laugh on

RT @surfmedallas: #CallMeCam PLEASE CAM I LOVE YOU SO MUCH 😩💗 x16
@camerondallas can you please #CallMeCam tonight? My birthdays tommorow and it would be awesome, i love you 😎
💖
💖
💖
💖
💖
💖
#CallMeCam
💖
13
I've developed a newly found love for ice cream 🍦🍨🍧💘

RT @WhatsAFeeling_: Happy birthday slim thick @LittleMissPete
@camerondallas #CallMeCam #CallMeCam #CallMeCam call mee cam I love you😍 #CallMeCam #CallMeCam #CallMeCam @camerondallas #CallMeCam 3

#CallMeCam Please I love you and you never noticed me. Ily ❤😘💘💋 Please Call Me x12

Can we please order a ball gown made from the new Stained Glass pattern @cindabusa ? Love all the… http://t.co/M3oaSF3tLy
just really sad and disappointed. happy vibe def killed.
@TTLYTEALA U HAVE MADE ME SO HAPPY AKDJSKKS LOVE YOU SO MUCH

Break free is life full of love #BuyBreakFreeOniTunes http://t.co/1gRuAh0lc2
Ground Up's 'Let's Ride' Is Your New Summer Party Anthem -  We love Ground Up here at the Huffington Post, and our... http://t.co/5OVh0ohi42

Next time I come to Atlanta I'm going to find the whole cast from Love & Hip Hop!
```

Real World Task Distribution

Real World Task Distribution

Transformations and **Actions**:

Similar to Scala but …

> Your choice of transformations need be made with **task distribution** and **memory** in mind

Real World Task Distribution

How do we do that?

- **Partition the data** appropriately for number of cores (at least 2x number of cores)

Real World Task Distribution

How do we do that?

- **Partition the data** appropriately for number of cores (at least 2x number of cores)
- **Filter early** and often (Queries, Filters, Distinct...)

Real World Task Distribution

How do we do that?

- **Partition the data** appropriately for number of cores (at least 2x number of cores)
- **Filter early** and often (Queries, Filters, Distinct...)
- Use **pipelines**

Real World Task Distribution

How do we do that?

- Partial **Aggregation**

Real World Task Distribution

How do we do that?

- Partial **Aggregation**
- Create an algorithm to be as **simple/efficient** as it can be to appropriately answer the question

Real World Task Distribution

```
 id    | age | email          | fname | gender | interests | lname | phone
-------+-----+----------------+-------+--------+-----------+-------+---------
 x5rd3 |  40 | foo@gmail.com |  Matt |      m |     music |   Kew | 7654321


Person(age, email) // SELECT email FROM persons WHERE age > 17

val overTheHillDemo = persons.filter(_.age > 39) // Assumes there are other demographics as well...

subsequent filtering ...
```

Real World Task Distribution

Some Common Costly Transformations:

- sorting
- groupByKey
- reduceByKey
- ...

Partitioning Example

User event log:

- **Time-series** events
- Tracks **user interactions** with system over time
- Location check-ins, page/module views, profile changes, ad impressions/clicks, etc.

Partitioning Example

```
CREATE TABLE Event (
  user_id text,
  timestamp int,
  event_type text,
  event_data map<text, text>,
  PRIMARY KEY (user_id, timestamp, event_type)
);
```

Partitioning Example

```
CREATE TABLE Event (
  user_id text,
  timestamp int,
  event_type text,
  event_data map<text, text>,
  PRIMARY KEY (user_id, timestamp, event_type)
);
```

partition key

Partitioning Example

```
CREATE TABLE Event (
  user_id text,
  timestamp int,
  event_type text,
  event_data map<text, text>,
  PRIMARY KEY (user_id, timestamp, event_type)
);
```

clustering columns

Partitioning Example

Potential analysis:

- Location graph
- Individual usage habits
- Page/module view counts

Partitioning Example

Potential analysis:

- Location graph
- Individual usage habits
- Page/module view counts

Grouped by:

user_id

Partitioning Example

Potential analysis:

- Location graph
- Individual usage habits
- Page/module view counts

Grouped by:

user_id

user_id

Partitioning Example

Potential analysis:	Grouped by:
● Location graph	**user_id**
● Individual usage habits	**user_id**
● Page/module view counts	**event_data**

Partitioning Example

Node 1

rstrickland
jsmith
tjones

Node 2

awilson
ptaylor
gwatson

Node 3

lmiller
mjohnson
scarter

Partitioning Example

Node 1	Node 2	Node 3
rstrickland	awilson	lmiller
jsmith	ptaylor	mjohnson
tjones	gwatson	scarter

users.reduceByKey { … }

Partitioning Example

Node 1	Node 2	Node 3
rstrickland	awilson	lmiller
jsmith	ptaylor	mjohnson
tjones	gwatson	scarter

users.reduceByKey { … }

No shuffling required!

Partitioning Example

Node 1

rstrickland: *home, local, video*
jsmith: *home, radar*
tjones: *radar, video*

Node 2

awilson: *home, local, radar*
ptaylor: *home, video*
gwatson: *local, radar*

Node 3

lmiller: *home, radar, video*
mjohnson: *radar*
scarter: *home, local, radar*

```
users.filter(_.eventType == "PageView")
```

Partitioning Example

Node 1

home: 2
local: 1
radar: 2
video: 2

Node 2

home: 2
local: 2
radar: 2
video: 1

Node 3

home: 2
local: 1
radar: 2
video: 1

```
users.filter(_.eventType == “PageView”)
      .map { e => (e.event_data[“page”], 1) }
```

Combining is automatic :)

Partitioning Example

Node 1	Node 2	Node 3
home: 2, 2, 2	local: 1, 2, 1 radar: 2, 2, 2	video: 2, 1, 1

```
users.filter(_.eventType == "PageView")
      .map { e => (e.event_data["pageview"], 1) }
      .reduceByKey(_ + _)
```

Requires a shuffle!

Java vs. Scala

Should I learn Scala?

… or leverage existing Java skills?

Java vs. Scala

- Spark uses a **functional paradigm**

Java vs. Scala

- Spark uses a **functional paradigm**
- Spark is written in Scala

Java vs. Scala

- Spark uses a **functional paradigm**
- Spark is written in Scala
- Scala > Java 8 > Java 7

Java vs. Scala

- Spark uses a **functional paradigm**
- Spark is written in Scala
- Scala > Java 8 > Java 7
- You will want **lambdas**!

Language Comparison - Scala

```
text.flatMap { line => line.split(“ “) }
      .map(word => (word, 1))
      .reduceByKey(_ + _)
```

Language Comparison - Java 8

```
text.flatMap(line -> Arrays.asList(line.split(“ “)))
    .mapToPair(word -> new Tuple2<String, Integer>(word, 1))
    .reduceByKey((x, y) -> x + y)
```

Language Comparison - Java 7

```
JavaRDD<String> words = text.flatMap(
  new FlatMapFunction<String, String>() {
    public Iterable<String> call(String line) {
      return Arrays.asList(line.split(" "));
    }
})
```

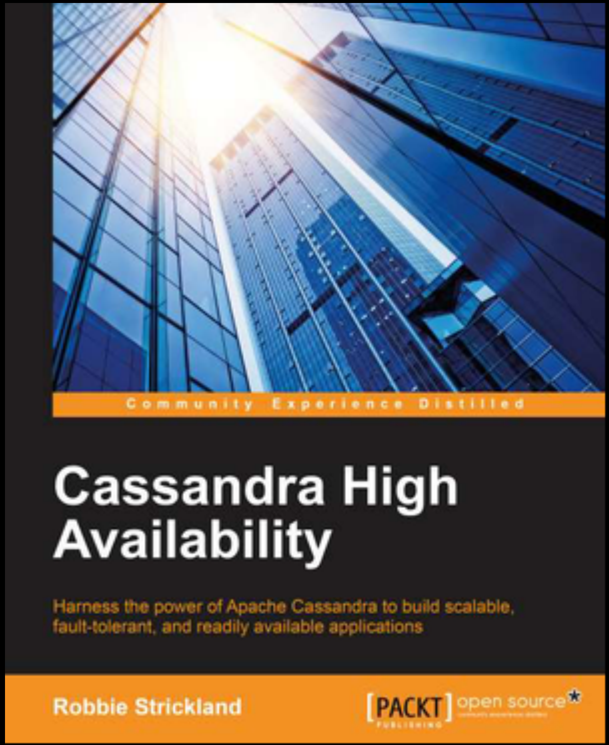
Language Comparison - Java 7

```
JavaPairRDD<String, Integer> ones = words.mapToPair(
  new PairFunction<String, String, Integer>() {
    public Tuple2<String, Integer> call(String w) {
      return new Tuple2<String, Integer>(w, 1);
    }
});
```

Language Comparison - Java 7

```
ones.reduceByKey(
  new Function2<Integer, Integer, Integer>() {
    public Integer call(Integer i1, Integer i2) {
      return i1 + i2;
    }
});
```

Shameless book plug



https://www.packtpub.com/big-data-and-business-intelligence/cassandra-high-availability

The Weather Channel

We’re hiring!

Thank you!

Robbie Strickland
linkedin.com/in/robbiestrickland
@rs_atl
rostrickland@gmail.com