HACKERS<>WEEK

Spark

Apache Spark: Paralelismo y Big Data

Antonio J. Nebro

- Dr. Ingeniero en Informática

- Grupo de Investigación Khaos
- Depto. Lenguajes y Ciencias de la Computación
- Universidad de Málaga

Khaos
INVESTIGACIÓN

uma
UNIVERSIDAD DE MÁLAGA

Contenidos del taller

- Introducción
- ¿Qué es Apache Spark?
- Spark y paralelismo
- Programación en Spark
- Conclusiones del taller

Introducción

- **Ponente**
  - Antonio Jesús Nebro Urbaneja
    - Doctor Ingeniero en Informática
    - Inicio de estudios en Informática: 1984
  - Afiliación
    - Departamento de Lenguajes y Ciencias de la Computación
      - E.T.S. De Ingeniería Informática
    - Grupo de investigación Khaos
      - Edificio de investigación Ada Byron
  - Contacto
    - antonio@lcc.uma.es
    - @AJNebro

Introducción

- Investigación (desde 1990)
  - Paralelismo
  - Optimización
- Software libre
  - Framework de optimización jMetal
  - http://jmetal.sourceforge.net
  - https://github.com/jMetal/jMetal

- Máster en Ingeniería Web (RIAtec)
  - http://riatec.lcc.uma.es

- Máster en Big Data Analytics (BDA)
  - http://bigdata.lcc.uma.es
- Curso de extensión universitaria "Introducción a Big Data y Hadoop"
  - http://bigdata.lcc.uma.es/curso/curso.html

Sobre el paralelismo

- El paralelismo intenta sacar partido a computadores paralelos para
  - Ejecutar programas más rápidamente
  - Ejecutar programas con altos requisitos de CPU, memoria, almacenamiento

- ¿Qué se estudia en las titulaciones de Ingeniería Informática?
  - Fundamentos: secciones críticas, interbloqueos, …
  - Soluciones de bajo nivel
    - Memoria compartida: semáforos, monitores
    - Memoria distribuida: paso de mensajes
  - ¿Qué tienen en común?
    - Complejidad, dificultad de reutilizar código

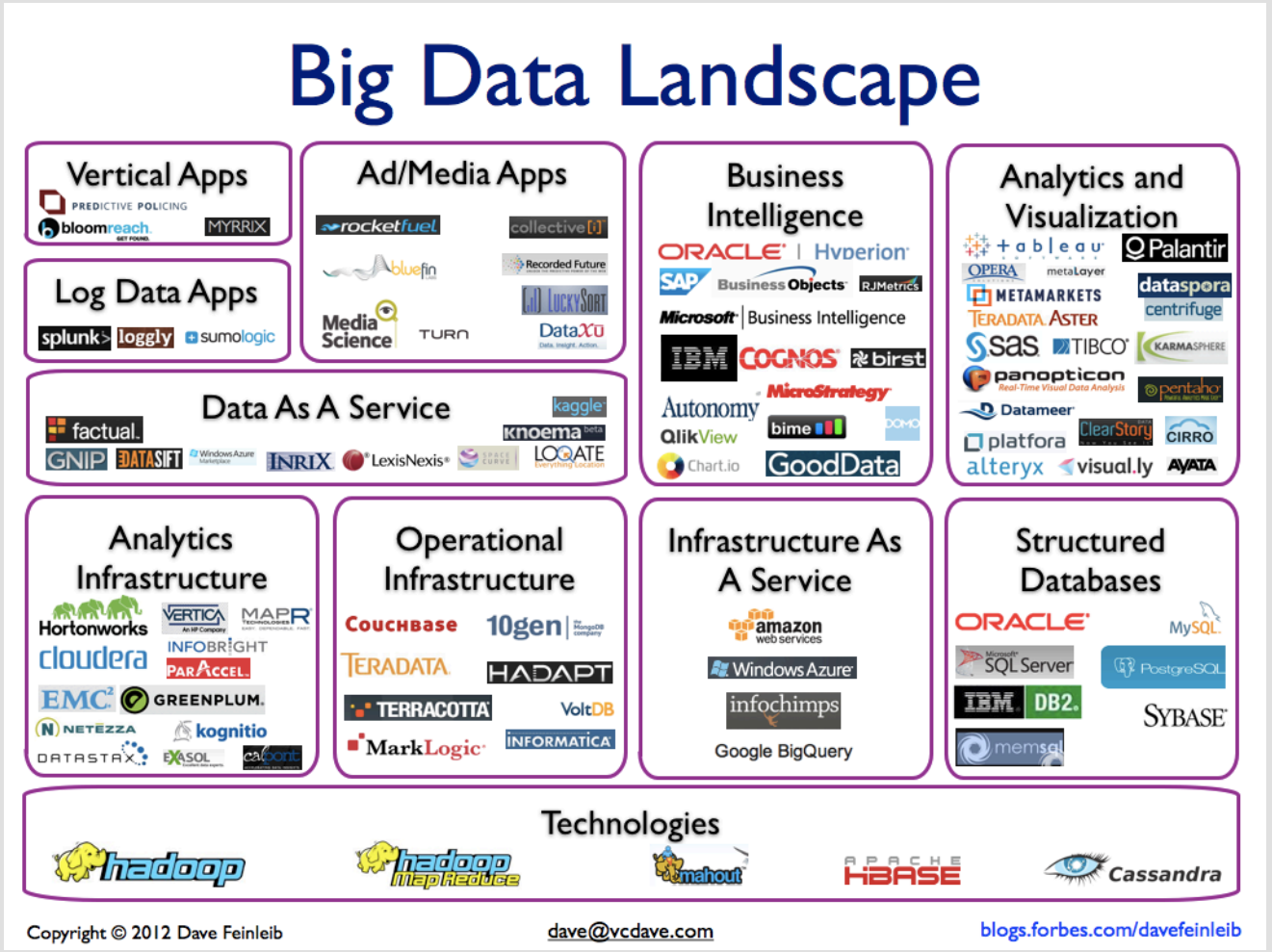
Sobre el Big Data

- Big Data
  - Procesar grandes cantidades de información cuando las tecnologías tradicionales de BBDD no son suficientes

Big data Landscape 1.0

Spark

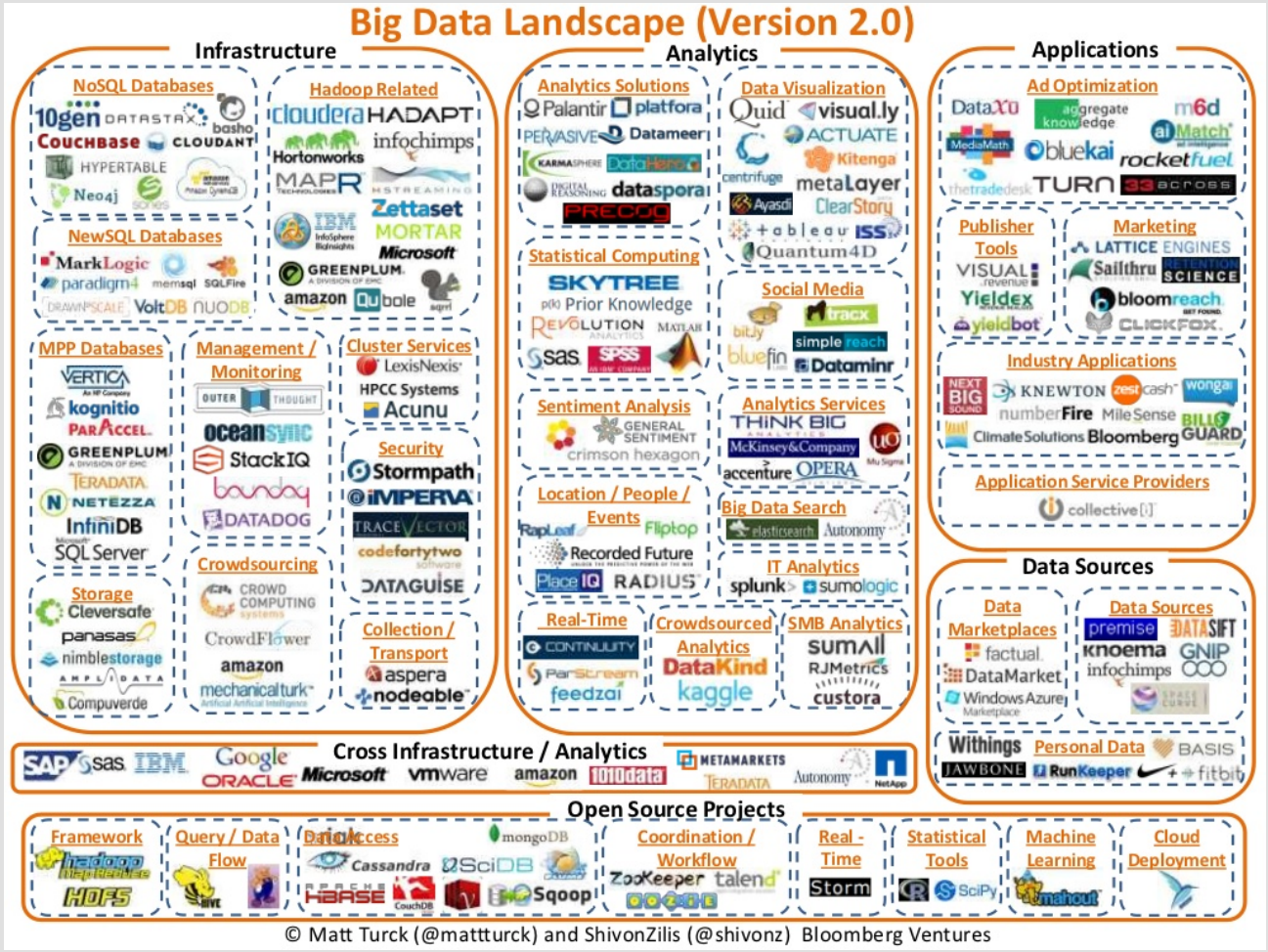
Paralelismo y Big Data



Big data Landscape 2.0

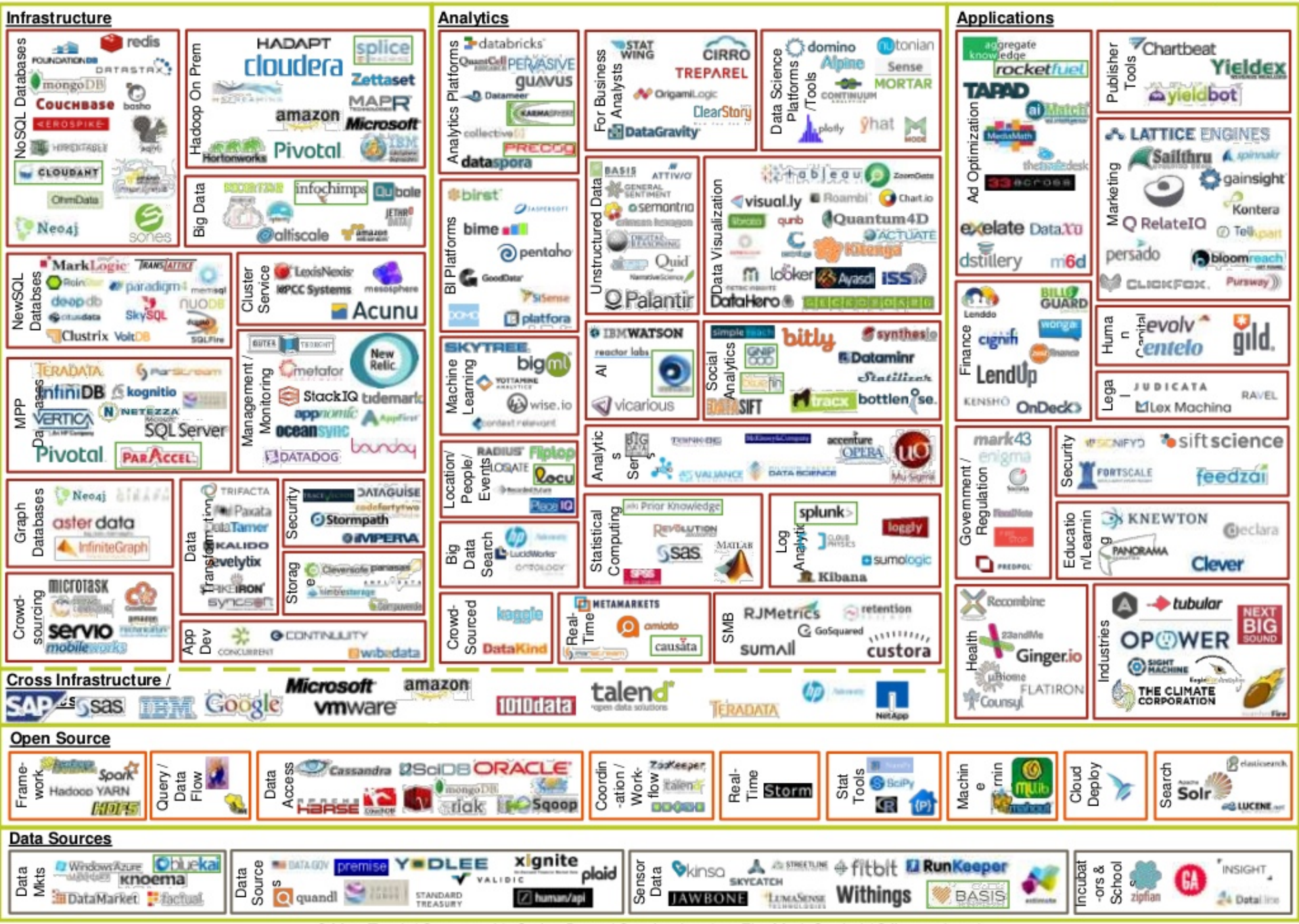
Spark

Paralelismo y Big Data



BIG DATA LANDSCAPE, VERSION 3.0

Exited: Acquisition or IPO



El ecosistema Hadoop

Spark

Paralelismo y Big Data

¿Qué es Apache Spark?

- Apache Spark

https://spark.apache.org/docs/latest/index.html

  - Is a fast and general-purpose cluster computing system
    - Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.
  - It provides high-level APIs in Java, Scala and Python
  - It also supports a rich set of higher-level tools including
    - Spark SQL for SQL and structured data processing
    - MLlib for machine learning, GraphX for graph processing
    - Spark Streaming
  - It can access diverse data sources
    - HDFS, Cassandra, Hbase, S3

Versiones de Spark

Latest News

Spark 1.3.0 released (Mar 13, 2015)

Spark 1.2.1 released (Feb 09, 2015)

Spark Summit East agenda posted, CFP open for West (Jan 21, 2015)

Spark 1.2.0 released (Dec 18, 2014)

Archive

Spark y paralelismo

- El paralelismo en Spark es a muy alto nivel
  - No hay que manejar semáforos, secciones críticas, etc.

- Un mismo programa Spark se puede ejecutar
  - En un PC con procesador multi-núcleo
  - En un cluster propio (standalone deploy mode)
  - Apache Mesos (https://mesos.apache.org)
  - Hadoop YARN

Programación en Spark

Paralelismo y Big Data

- Objetivo del taller
  - Aplicación (simplificada) para obtener *trending topics* de Twitter
  - Fuente datos
    - Fichero de tweets de un partido de baloncesto de la liga universitaria de los EE.UU

443955894288535552	@SebastianH007	RT BaylorMBB: That's a wrap from Sprint Center. Baylor moves on to the #Big12MBB Quarterfinals to face Oklahoma. #SicOU http://t.co/uhjn3C?	Thu Mar 13 04:45:08 CET 2014
443955835442429953	@BaylorMBB	Cory J presenting coach Drew with game ball from his 200th win at Baylor. #SicEm #Big12MBB http://t.co/JnbwHZIxZc Wac	Thu Mar 13 04:44:54 CET 2014
443955799417552896	@DerBaylorBaer	RT HeathNielsen: It's unbelievable how far this man has taken Baylor basketball since we hit rock bottom in 2003. Congrats BUDREW 200 htt?	Thu Mar 13 04:44:45 CET 2014
443955790194282496	@wingoz	RT VoiceofBears: RT BaylorAthletics: RT BaylorMBB: Congratulations to Coach BUDREW on career win No. 200. #SicEm http://t.co/Z74PQydzU5	Thu Mar 13 04:44:43 CET 2014
443955733457944576	@HeathNielsen	It's unbelievable how far this man has taken Baylor basketball since we hit rock bottom in 2003. Congrats BUDREW 200 http://t.co/I2dsT5Psxg Heart of Texas	Thu Mar 13 04:44:29 CET 2014
443955697273671681	@austin_anderson	RT BaylorMBB: Congratulations to Coach BUDREW on career win No. 200. #SicEm http://t.co/l7lBT0llyR Dallas	Thu Mar 13 04:44:21 CET 2014
443955693758857217	@EmbracinChaos	RT BaylorMBB: That's a wrap from Sprint Center. Baylor moves on to the #Big12MBB Quarterfinals to face Oklahoma. #SicOU http://t.co/uhjn3C?	Thu Mar 13 04:44:20 CET 2014

Programación en Spark

- Herramientas de desarrollo:
  - Spark: https://spark.apache.org/downloads.html
  - IDE de desarrollo: Eclipse Luna, IntelliJ Idea 14
  - Maven (gestión de dependencias): http://maven.apache.org/download.cgi?Preferred=ftp://mirror.reverse.net/pub/apache/

Ejemplo 1: sumar números

Paralelismo y Big Data

```java
package es.uma.hackersweek.sparkdemo;

import org.apache.spark.SparkConf;

 * Example to show how to start a Spark application and create and work with a JavaRDD (Res

public class SparkDemoSumAListOfNumbers {
  public static void main(String[] args) {
    // STEP 1: create a SparkConf object
    SparkConf conf = new SparkConf().setAppName("SparkDemo") ;

    // STEP 2: create a Java Spark context
    JavaSparkContext sparkContext = new JavaSparkContext(conf) ;

    // An array of Integer objects
    Integer[] values = new Integer[]{1, 2, 3, 4, 5, 6, 8, 9} ;

    // A list of Integers
    List<Integer> data = Arrays.asList(values);

    // STEP 3: create a JavaRDD
    JavaRDD<Integer> distributedData = sparkContext.parallelize(data);

    // STEP 4: compute the sum
    int sum = distributedData.reduce(new Function2<Integer, Integer, Integer>() {
      @Override public Integer call(Integer integer, Integer integer2) throws Exception {
        return integer+integer2;
      }
    }) ;

    // STEP 5: print the result
    System.out.println("The sum is: " + sum) ;

    // STEP 6: stop de spark context
    sparkContext.stop();
  }
}
```

Fundamentos de Spark: RDD

- JavaRDD
  - *Resilient distributed dataset*
  - Colección de elementos que pueden ser procesados en paralelo
  - Tolerantes a fallos

- Creación de RDDs
  - Paralelizando una colección de datos que ya existe
  - A partir de datos almacenados en un sistema de ficheros o base de datos externa

Fundamentos de Spark: tipos de operaciones

- **Transformaciones**
  - **Crean datos a partir de otros**
- Acciones
  - Devuelven un valor tras procesar un conjunto de datos
- Ejemplos

Transformación	Significado
map(func)	Return a new distributed dataset formed by passing each element of the source through a function func.
filter(func)	Return a new dataset formed by selecting those elements of the source on which func returns true.
flatMap(func)	Similar to map, but each input item can be mapped to 0 or more output items (so func should return a Seq rather than a single item).
union(otherDataset)	Return a new dataset that contains the union of the elements in the source dataset and the argument.

Fundamentos de Spark: tipos de operaciones

- Transformaciones
  - Crean datos a partir de otros
- **Acciones**
  - **Devuelven un valor tras procesar un conjunto de datos**
- Ejemplos

Acción	Significado
reduce(*func*)	Aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	Return the number of elements in the dataset.
first()	Return the first element of the dataset.
saveAsTextFile(*path*)	Write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system.

Ejemplo 1: sumar números. Compilación

Paralelismo y Big Data

- Maven POM.XML

http://search.maven.org/

The Central Repository

SEARCH

spark-core

New: About Central | Advanced Search | API Guide | Help

Artifact Details For org.apache.spark : spark-core_2.10 : 1.3.0

Click on a link above to browse the repository.

Project Information

GroupId:	org.apache.spark
ArtifactId:	spark-core_2.10
Version:	1.3.0

Dependency Information

Apache Maven

```
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.10</artifactId>
    <version>1.3.0</version>
</dependency>
```

Apache Buildr

Apache Ivy

Groovy Grape

Gradle/Grails

Scala SBT

Leiningen

Project Object Model (POM)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/
  <parent>
    <artifactId>spark-parent_2.10</artifactI
    <groupId>org.apache.spark</groupId>
    <version>1.3.0</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.10</artifactId>
  <name>Spark Project Core</name>
  <url>http://spark.apache.org/</url>
  <build>
    <outputDirectory>target/scala-${scala.bi
    <testOutputDirectory>target/scala-${scal
    <resources>
      <resource>
        <directory>src/main/resources</direc
      </resource>
      <resource>
        <directory>../python</directory>
        <includes>
          <include>pyspark/*.py</include>
        </includes>
      </resource>
      <resource>
```

Ejemplo 1: sumar números. Compilación

- Compilación

```
hackersWeek — antonio@master:~/Softw/hackersWeek/hackersweek2015 — bash — 100×39
MacBook-Pro-de-Antonio:hackersWeek ajnebro$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Spark 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Spark ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platfor
m dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Spark ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ Spark ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platfor
m dependent!
[INFO] skip non existing resourceDirectory /Users/ajnebro/Softw/hackersWeek/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ Spark ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ Spark ---
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ Spark ---
[INFO] Building jar: /Users/ajnebro/Softw/hackersWeek/target/Spark-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-assembly-plugin:2.4:single (make-assembly) @ Spark ---
[INFO] Building jar: /Users/ajnebro/Softw/hackersWeek/target/Spark-1.0-SNAPSHOT-jar-with-dependencie
s.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 24.151 s
[INFO] Finished at: 2015-03-19T18:46:57+01:00
[INFO] Final Memory: 30M/756M
[INFO] ------------------------------------------------------------------------
MacBook-Pro-de-Antonio:hackersWeek ajnebro$
```

Ejemplo 1: sumar números. Ejecución

hackersWeek — antonio@master:~/Softw/hackersWeek/hackersweek2015 — bash — 97×25

```
MacBook-Pro-de-Antonio:hackersWeek ajnebro$ cat runSparkDemoSumAListOfNumbers.sh
spark-submit --class "es.uma.hackersweek.sparkdemo.SparkDemoSumAListOfNumbers" --master local[8]
target/Spark-1.0-SNAPSHOT-jar-with-dependencies.jar
MacBook-Pro-de-Antonio:hackersWeek ajnebro$
```

hackersWeek — antonio@master:~/Softw/hackersWeek/hackersweek2015 — bash — 97×25

```
MacBook-Pro-de-Antonio:hackersWeek ajnebro$ ./runSparkDemoSumAListOfNumbers.sh
The sum is: 38
MacBook-Pro-de-Antonio:hackersWeek ajnebro$
```

Ejemplo 2: sumar números de ficheros


```java
package es.uma.hackersweek.sparkdemo;

import org.apache.spark.SparkConf;

/ * Example showing how to read data from files. The applications sum a list of integer values

public class SparkDemoSumAListOfNumbersStoredInAFile {
  public static void main(String[] args) {

    // STEP 1: argument checking
    if (args.length == 0) {
      throw new RuntimeException("The number of args is 0. Usage: "
          + "SparkDemoSumAListOfNumberStoredInAFile fileOrDirectoryName") ;
    }

    // STEP 2: create a SparkConf object
    SparkConf conf = new SparkConf().setAppName("SparkDemo") ;

    // STEP 3: create a Java Spark context
    JavaSparkContext sparkContext = new JavaSparkContext(conf) ;

    // STEP 4: read the lines from the file(s)
    JavaRDD<String> lines = sparkContext.textFile(args[0]) ;

    // STEP 5: map the string coded numbers into integers
    JavaRDD<Integer> listOfNumbers = lines.map(new Function<String, Integer>() {
      @Override public Integer call(String s) throws Exception {
        return Integer.valueOf(s);
      }
    }) ;

    // STEP 6: compute the sum
    int sum = listOfNumbers.reduce(new Function2<Integer, Integer, Integer>() {
      @Override public Integer call(Integer integer, Integer integer2) throws Exception {
        return integer+integer2;
      }
    }) ;

    // STEP 7: print the result
    System.out.println("The sum is: " + sum) ;

    // STEP 8: stop de spark context
    sparkContext.stop();
  }
}
```

Ejemplo 2.1: sumar números de ficheros

Paralelismo y Big Data

```java
package es.uma.hackersweek.sparkdemo;

import org.apache.spark.SparkConf;

/**
 * Example showing how to read data from files. The applications sum a list of integer values.
 * Each data file must contain an integer per line. Compact version using lambda expression
 * and fluent code.
 *
 * @author Antonio J. Nebro <antonio@lcc.uma.es>
 */

public class SparkDemoSumAListOfNumbersStoredInAFileCompactVersion {
  public static void main(String[] args) {

    // STEP 1: argument checking
    if (args.length == 0) {
      throw new RuntimeException("The number of args is 0. Usage: "
          + "SparkDemoSumAListOfNumbersStoredInAFileCompactVersion fileOrDirectoryName") ;
    }

    // STEP 2: create a SparkConf object
    SparkConf conf = new SparkConf().setAppName("SparkDemo") ;

    // STEP 3: create a Java Spark context
    JavaSparkContext sparkContext = new JavaSparkContext(conf) ;

    // STEP 4: read the lines from the file(s)
    JavaRDD<String> lines = sparkContext.textFile(args[0]) ;

    // STEP 5: compute the sum
    int sum = lines.map(s -> Integer.valueOf(s)).reduce((integer, integer2) -> integer+integer2) ;

    // STEP 6: print the result
    System.out.println("The sum is: " + sum) ;

    // STEP 8: stop de spark context
    sparkContext.stop();
  }
}
```

Fundamentos de Spark: tipos de operaciones

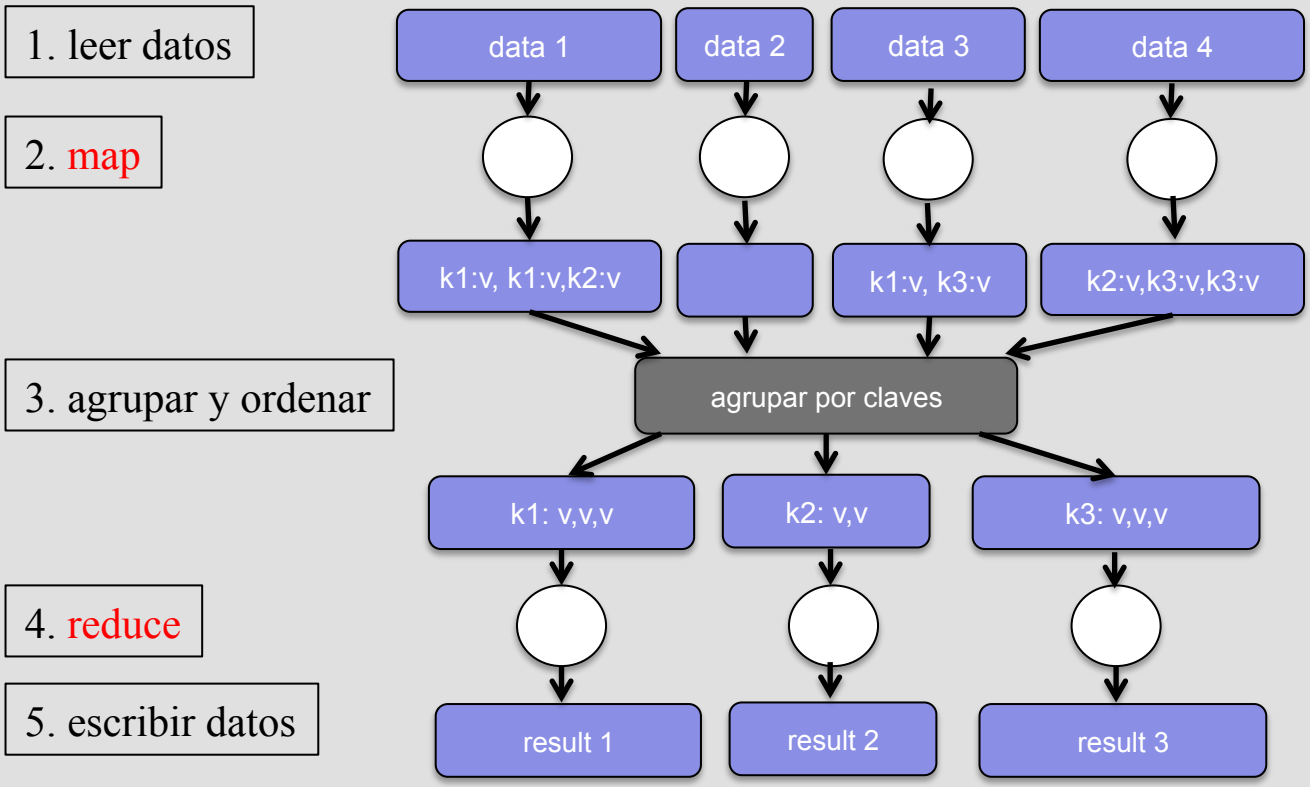
- Transformaciones
  - Crean datos a partir de otros
- Acciones
  - Devuelven un valor tras procesar un conjunto de datos
- **Carácterística CLAVE**:
  - Las transformaciones son LAZY (se ejecutan de forma perezosa)
  - Únicamente se computan cuando una acción requiere devolver un valor

Ejemplo 3: contar caracteres de ficheros


```java
package es.uma.hackersweek.sparkdemo;

import org.apache.log4j.Logger;

/ * Spark applications which counts the number of lines of a file (or the files in a di
public class SparkDemoFindCharactersInFiles {
  static Logger log = Logger.getLogger(SparkDemoFindCharactersInFiles.class.getName())

  public static void main(String[] args) throws ClassNotFoundException {

    // STEP 1: argument checking
    if (args.length < 1) {
      log.fatal("Syntax Error: there must be one argument (a file or directory name)")
      throw new RuntimeException();
    }

    String fileName = args[0];

    // STEP 2: create a SparkConf object
    SparkConf sparkConf = new SparkConf().setAppName("FindCharacters");

    // STEP 3: create a Java Spark context
    JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);

    // STEP 4: read lines of files
    JavaRDD<String> lines = sparkContext.textFile(fileName).cache();

    // STEP 5: filter and count the number of lines containing the character 'a'
    long numAs = lines.filter(new Function<String, Boolean>() {
      public Boolean call(String s) {
        return s.contains("a");
      }
    }).count();

    // STEP 6: filter and count the number of lines containing the character 'b'
    long numBs = lines.filter(new Function<String, Boolean>() {
      public Boolean call(String s) {
        return s.contains("b");
      }
    }).count();

    // STEP 8: print the results
    System.out.println("Lines with a: " + numAs + ", lines with b: " + numBs);

    // STEP 9: stop the spark context
```

Ejemplo 4: contar palabras de ficheros

- Programa "Hello Word" en Big Data
  - Contar todas las ocurrencias de todas las palabras de un conjunto de ficheros
  - Paradigma Map/Reduce



Ejemplo 4: contar palabras de ficheros


```java
package es.uma.hackersweek.sparkdemo;

import org.apache.log4j.Logger;

 * Example implementing the Spark version of the "Hello World" Big Data program: counting the

public class SparkDemoCountWordsFromFiles {
  static Logger log = Logger.getLogger(SparkDemoCountWordsFromFiles.class.getName());

  public static void main(String[] args) {

    // STEP 1: create a SparkConf object
    if (args.length < 1) {
      log.fatal("Syntax Error: there must be one argument (a file name or a directory)")  ;
      throw new RuntimeException();
    }

    // STEP 2: create a SparkConf object
    SparkConf sparkConf = new SparkConf().setAppName("Spark Word count") ;

    // STEP 3: create a Java Spark context
    JavaSparkContext sparkContext = new JavaSparkContext(sparkConf) ;

    // STEP 4: read lines of files
    JavaRDD<String> lines = sparkContext.textFile(args[0]);
```

Ejemplo 4: contar palabras de ficheros

Paralelismo y Big Data

```
43      // STEP 5: split the lines into words
44      JavaRDD<String> words = lines.flatMap(
45          new FlatMapFunction<String, String>() {
46            public Iterable call(String s) throws Exception {
47              return Arrays.asList(s.split(" "));
48            }
49          }) ;
50
51      // STEP 6: map operation to create pairs <word, 1> por each word
52      JavaPairRDD<String, Integer> ones = words.mapToPair(
53          new PairFunction<String, String, Integer>() {
54            public Tuple2<String, Integer> call(String string) {
55              return new Tuple2<String, Integer>(string, 1);
56            }
57          }
58          ) ;
59
60      // STEP 6: reduce operation that sum the values of all the pairs having the same key (word),
61      //         generating a pair <key, sum>
62      JavaPairRDD<String, Integer> counts = ones.reduceByKey(
63          new Function2<Integer, Integer, Integer>() {
64            public Integer call(Integer integer, Integer integer2) throws Exception {
65              return integer + integer2 ;
66            }
67          }
68          ) ;
69
70      // STEP 7: sort the results by key
71      List<Tuple2<String, Integer>> output = counts.sortByKey().collect() ;
72
73      // STEP 8: print the results
74      for (Tuple2<?, ?> tuple : output) {
75        System.out.println(tuple._1() + ": " + tuple._2()) ;
76      }
77
78      // STEP 9: stop the spark context
79      sparkContext.stop();
80    }
81  }
```

Ejemplo 5: trending topics in tweets

- Aplicación (muy simplificada) para obtener *trending topics* de Twitter
- Fuente datos
  - Fichero de tweets de un partido de baloncesto de la liga universitaria de los EE.UU

443955894288535552	@SebastianH007	RT BaylorMBB: That's a wrap from Sprint Center. Baylor moves on to the #Big12MBB Quarterfinals to face Oklahoma. #SicOU http://t.co/uhjn3C?	Thu Mar 13 04:45:08 CET 2014
443955835442429953	@BaylorMBB	Cory J presenting coach Drew with game ball from his 200th win at Baylor. #SicEm #Big12MBB http://t.co/JnbwHZIxZc Wac	Thu Mar 13 04:44:54 CET 2014
443955799417552896	@DerBaylorBaer	RT HeathNielsen: It's unbelievable how far this man has taken Baylor basketball since we hit rock bottom in 2003. Congrats BUDREW 200 htt?	Thu Mar 13 04:44:45 CET 2014
443955790194282496	@wingoz	RT VoiceofBears: RT BaylorAthletics: RT BaylorMBB: Congratulations to Coach BUDREW on career win No. 200. #SicEm http://t.co/Z74PQydzU5	Thu Mar 13 04:44:43 CET 2014
443955733457944576	@HeathNielsen	It's unbelievable how far this man has taken Baylor basketball since we hit rock bottom in 2003. Congrats BUDREW 200 http://t.co/I2dsT5Psxg Heart of Texas	Thu Mar 13 04:44:29 CET 2014
443955697273671681	@austin_anderson	RT BaylorMBB: Congratulations to Coach BUDREW on career win No. 200. #SicEm http://t.co/l7lBT0llyR Dallas	Thu Mar 13 04:44:21 CET 2014
443955693758857217	@EmbracinChaos	RT BaylorMBB: That's a wrap from Sprint Center. Baylor moves on to the #Big12MBB Quarterfinals to face Oklahoma. #SicOU http://t.co/uhjn3C?	Thu Mar 13 04:44:20 CET 2014

Ejemplo 5: trending topics in tweets

Paralelismo y Big Data

```java
package es.uma.hackersweek.twitter;

import org.apache.spark.SparkConf;

/**
 * Example showing how to count the most used words in files containing tweets
 *
 * @author Antonio J. Nebro <antonio@lcc.uma.es>
 */

public class TwitterTrendingTopics {
  public static void main(String[] args) {

    // STEP 1: argument checking
    if (args.length == 0) {
      throw new RuntimeException("The number of args is 0. Usage: "
          + "TwitterTrendingTopics fileOrDirectoryName") ;
    }

    // STEP 2: create a SparkConf object
    SparkConf conf = new SparkConf().setAppName("TwitterTrendingTopics") ;

    // STEP 3: create a Java Spark context
    JavaSparkContext sparkContext = new JavaSparkContext(conf) ;

    // STEP 4: read the lines from the file(s)
    JavaRDD<String> lines = sparkContext.textFile(args[0]) ;

    // STEP 5: map to select the strings in the column 2, which contains the tweet message
    JavaRDD<String> messages = lines.map(new Function<String, String>() {
      public String call(String s) {
        String [] split = s.toString().split("\t+") ;

        return split[2];
      }
    });

    // STEP 5: split the lines into words
    JavaRDD<String> words = messages.flatMap(
        new FlatMapFunction<String, String>() {
          public Iterable call(String s) throws Exception {
            return Arrays.asList(s.split(" "));
          }
        }) ;
```

HACKERS<>WEEK

Ejemplo 5: trending topics in tweets

Paralelismo y Big Data

```
56      // STEP 6: map operation to create pairs <word, 1> por each word
57      JavaPairRDD<String, Integer> ones = words.mapToPair(
58          new PairFunction<String, String, Integer>() {
59            public Tuple2<String, Integer> call(String string) {
60              return new Tuple2<String, Integer>(string, 1);
61            }
62          }
63      ) ;
64
65      // STEP 6: reduce operation that sum the values of all the pairs having the same key (word),
66      //         generating a pair <key, sum>
67      JavaPairRDD<String, Integer> counts = ones.reduceByKey(
68          new Function2<Integer, Integer, Integer>() {
69            public Integer call(Integer integer, Integer integer2) throws Exception {
70              return integer + integer2 ;
71            }
72          }
73      ) ;
74
75      // STEP 7: invert the keys and values to use sortByKey() in step 8 on the values instead of th
76      //         keys
77      JavaPairRDD<Integer, String> reverse = counts.mapToPair(
78          new PairFunction<Tuple2<String, Integer>, Integer, String>() {
79        @Override public Tuple2<Integer, String> call(Tuple2<String, Integer> stringIntegerTuple2)
80            throws Exception {
81          return new Tuple2<Integer, String>(stringIntegerTuple2._2, stringIntegerTuple2._1);
82        }
83      }) ;
84
85      /*// STEP 8: sort the results by key
86      List<Tuple2<Integer, String>> output = reverse.sortByKey().collect() ;
87
88      // STEP 9.1: print the results to screen
89      for (Tuple2<?, ?> tuple : output) {
90        System.out.println(tuple._1() + ": " + tuple._2()) ;
91      }*/
92
93      // STEP 9.2 print the results to file
94      reverse.sortByKey().saveAsTextFile("data/tweets/output");
95
96      // STEP 10: stop de spark context
97      sparkContext.stop();  }
98  }
```

Conclusiones del taller

- En los últimos 30 años las tecnologías informáticas han estado en continua evolución
  - Pero en los últimos 5 años los avances se suceden a ritmo vertiginoso
  - Ámbitos:
    - Web: HTML 5, CSS3, Javascript, AngularJS, NodeJS, Bootstrap, etc.
    - Dispositivos móviles: Android, iOS, Windows 10?
    - Big Data: Hadoop, MapReduce, Hive, Hbase, Spark, etc.
- Spark es un ejemplo de tecnología que ha aparecido recientemente y que tiene una gran proyección
  - Nuevas versiones casi cada mes
  - Adopción por muchas empresas
  - Herramienta de alto nivel y elevado rendimiento
  - Puede ser la nueva tecnología principal sobre la que se sustenten las aplicaciones de Big Data
- A saber qué ocurre el año que viene por estas fechas …

Gracias por asistir

Paralelismo y Big Data

Antonio J. Nebro