Scala Meetup 22.06.2016

How Spark can improve my Hadoop Cluster?

Darius Murawski @ Wer liefert was? GmbH

Intro

- Wer liefert was? GmbH, B2B Search Engine: www.wlw.de
- Since Sep. 2014 Web Developer
- Since April 2016 doing Scala
- MAD Team, Increase data qualitity of company and product data
- Contact at Twitter: @dariusmur
- Github: https://github.com/dariusgm/scala-meetup-2016

Content

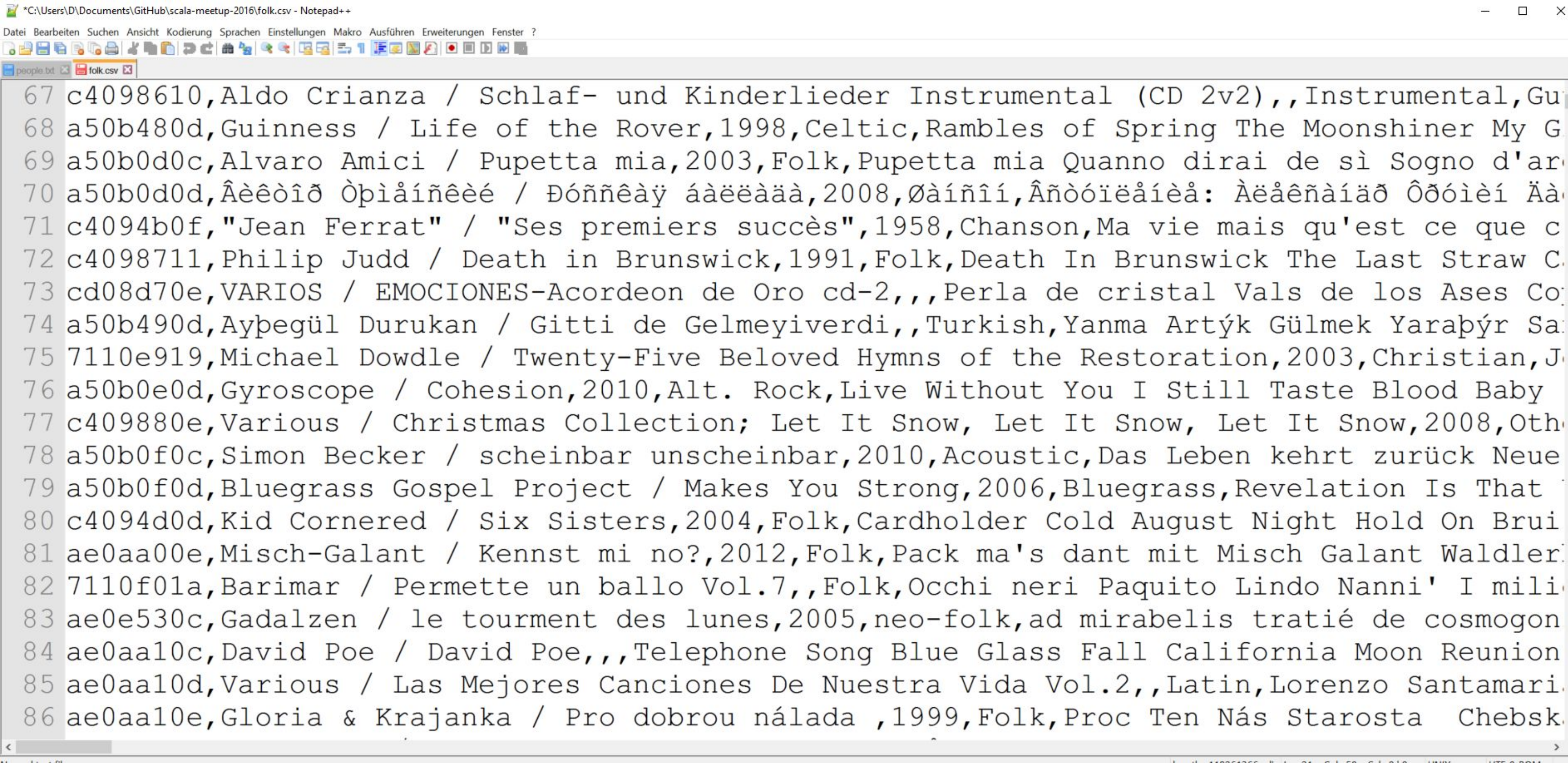
Brief Introduction to Apache Hadoop

Apache Hadoop

- Java Framework
- Hadoop Distributed File System (HDFS)
- Map Reduce, allows parallel processing of data in a HDFS

Our Testdata: freedb.org

- Metadata of CD Roms (You know them?!)
- Get it here: http://www.freedb.org
- Tar contains several hundert thoused of small files → Merge them together in one file
- Sourcecode on Github
- With merged gzip File
- For Demos, only “Folk”


*C:\Users\D\Documents\GitHub\scala-meetup-2016\folk.csv - Notepad++
Datei Bearbeiten Suchen Ansicht Kodierung Sprachen Einstellungen Makro Ausführen Erweiterungen Fenster ?
people.txt
folk.csv

```
c4098610,Aldo Crianza / Schlaf- und Kinderlieder Instrumental (CD 2v2),,Instrumental,Gu
a50b480d,Guinness / Life of the Rover,1998,Celtic,Rambles of Spring The Moonshiner My G
a50b0d0c,Alvaro Amici / Pupetta mia,2003,Folk,Pupetta mia Quanno dirai de sì Sogno d'ar
a50b0d0d,Âèêòîð Òþìåíñêèé / Ðóññêàÿ áàëëàäà,2008,Øàíñîí,Âñòóïëåíèå: Àëåêñàíäð Ôðóìèí Äà
c4094b0f,"Jean Ferrat" / "Ses premiers succès",1958,Chanson,Ma vie mais qu'est ce que c
c4098711,Philip Judd / Death in Brunswick,1991,Folk,Death In Brunswick The Last Straw C
cd08d70e,VARIOS / EMOCIONES-Acordeon de Oro cd-2,,,Perla de cristal Vals de los Ases Co
a50b490d,Ayþegül Durukan / Gitti de Gelmeyiverdi,,Turkish,Yanma Artýk Gülmek Yaraþýr Sa
7110e919,Michael Dowdle / Twenty-Five Beloved Hymns of the Restoration,2003,Christian,J
a50b0e0d,Gyroscope / Cohesion,2010,Alt. Rock,Live Without You I Still Taste Blood Baby
c409880e,Various / Christmas Collection; Let It Snow, Let It Snow, Let It Snow,2008,Oth
a50b0f0c,Simon Becker / scheinbar unscheinbar,2010,Acoustic,Das Leben kehrt zurück Neue
a50b0f0d,Bluegrass Gospel Project / Makes You Strong,2006,Bluegrass,Revelation Is That
c4094d0d,Kid Cornered / Six Sisters,2004,Folk,Cardholder Cold August Night Hold On Brui
ae0aa00e,Misch-Galant / Kennst mi no?,2012,Folk,Pack ma's dant mit Misch Galant Waldler
7110f01a,Barimar / Permette un ballo Vol.7,,Folk,Occhi neri Paquito Lindo Nanni' I mili
ae0e530c,Gadalzen / le tourment des lunes,2005,neo-folk,ad mirabelis tratié de cosmogon
ae0aa10c,David Poe / David Poe,,,Telephone Song Blue Glass Fall California Moon Reunion
ae0aa10d,Various / Las Mejores Canciones De Nuestra Vida Vol.2,,Latin,Lorenzo Santamari
ae0aa10e,Gloria & Krajanka / Pro dobrou nálada ,1999,Folk,Proc Ten Nás Starosta  Chebsk
```

Apache Spark Basics

Apache Spark

- Framework on top of Apache Hadoop
- API in Java, Scala, Python and R
- get it here: spark.apache.org
- REPL (Read–eval–print loop)
- Runs locally if needed

Apache Spark Definitions

RDD	Resillient Distributed Dataset
sc	Spark context

Optimization

- Build up an directed execution Graph
- Optimize it
- and then run it instead of running several MapReduce Tasks

Lazy Evaluation

- Building the Execution Graph also for Lazy Evaluation
- Transformation
  - Build up Graph
  - http://spark.apache.org/docs/1.6.1/programming-guide.html#transformations
  - map, filter, flatmap, ...
- Action
  - Execute as MapReduce Task
  - reduce, collect, count, first, ...

Caching

- Save computed result from Graph in the Spark Worker

Initialize Spark

```
import org.apache.spark.{SparkConf, SparkContext}

val config = new SparkConf()

  .setAppName("LineCount")

  .setMaster("local[8]")

  .set("spark.executor.memory","2g")

val sc = new SparkContext(config)
```

Basics

```
val lines = sc.textFile("folk.csv").cache()

// First Entry in Line

val firstLine = lines.first() // Action

println(firstLine) // cd11750f,Broery Marantika / The best
of,,,hati yang ...
```

Most Popular Word in Album Name

```
val splittedAlbum = lines.map(line =>          line.split
(",")(1)).cache()
// Broery Marantika / The best of

val flatmap = splittedAlbum.flatMap(_.split(" "))

val words = flatmap.flatMap(line => line.split(" "))

  .map(word => (word, 1)) //("word",1), ("word",1)

  .reduceByKey(_ + _)   //("word",2)
```

Most Popular Word in Album Name

```
val sortByValues = words.map((item) => item.swap).sortByKey
(false)

sortByValues.take(200).map(println)
```

Most Popular Word in Album Name

```
(290247,/) (34946,The) (34445,Various) (26245,-) (15685,&)
(12836,of) (11071,Artists) (10772,de) (10748,the) (8950,2)
(7293,) (7027,1) (6856,Of) (6385,A) ...
```

Results

```
(290247,/) (34946,The) (34445,Various) (26245,-) (15685,&) (12836,of) (11071,Artists) (10772,de)
(10748,the) (8950,2) (7293,) (7027,1) (6856,Of) (6385,A)
```

Hint: “The Various of Artists” not a good Name for your next (Folk) Band

Results

```
(290247,/) (34946,The) (34445,Various) (26245,-) (15685,&) (12836,of) (11071,Artists) (10772,de)
(10748,the) (8950,2) (7293,) (7027,1) (6856,Of) (6385,A)
```

At least hard for your SEO...

Spark Streaming

Spark Streaming

- Read data from various System
- Process the data in a set of RDDs in a timeframe - a DStream
- Save the data where you want to

- Support for Flume, Kafka, HDFS (surprise!) , RDMS, Elasticsearch and more

Spark Streaming

```
val ssc = new StreamingContext(config, Seconds(10))

val inputDirectory = "input"

val lines = ssc.textFileStream(inputDirectory)

val words = lines.flatMap(_.split(" "))
```

Spark Streaming

```
val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)

print(wordCounts.print())

ssc.start()

ssc.awaitTermination()
```

Create Files

```
(1 to 5).foreach(counter => {

 val string = "a" * counter

 scala.tools.nsc.io.File("input" + File.separator +
counter + ".txt").writeAll(string)

Thread.sleep(counter * 1000)

})
```

Example Output

```
-------------------------------------------

Time: 1466350670000 ms

-------------------------------------------

(aaaa,1)

(aaaaa,1)
```

Machine Learning Library (MLlib)

Machine Learning - Overview

Statistics	
Classification	Yes / No Descision
Regression	Numeric Descision
Trees / Forest	Multiple Desiscions with a target
Clustering	Grouping of Elements
Recommendations	

Statistics

```
val sc = new SparkContext(config)

val vectors = normalVectorRDD(sc,1000000L,2,8,123456)

val summary =  Statistics.colStats(vectors)

println("mean:  " +summary.mean)

println("variance" + summary.variance)

println("nonzeros" + summary.numNonzeros)
```

Clustering

```
val numClusters = 2

val numIterations = 20

val clusters = KMeans.train(vectors, numClusters,
numIterations)

clusters.clusterCenters.foreach(vector => {

 vector.toArray.map(_.toString).foreach(println)

 })
```

Clustering - Result

```
-0.7109633732036389

0.35891044312505577

0.7125945505236859

-0.36111194209959724
```

Clustering - Result

```
-0.7109633732036389

0.3589104431250557 7

0.7125945505236859

-0.3611194209959724
```

No good clustering on random data

MLlib Pipelines

MLlib Pipeline

- A Standard API for different algorithms that let you combine them
- Save and Load Pipeline Model
- http://spark.apache.org/docs/latest/ml-guide.html#overview-estimators-transformers-and-pipelines-sparkml

GraphX

GraphX

- Store Verticies and Edges
- Many RDD Operations on Vertecies and Edges
- PageRank
- Connected Components
- Triangle Counting

GraphX Users

1	Martin
2	Jennifer
3	Matti
5	Florian
6	Johannes
7	Jens
8	Darius

GraphX Followed By

1	Martin	2 6 7 8
2	Jennifer	3 5 6 8
3	Matti	5 8
5	Florian	2 3 8
6	Johannes	2 5 8
7	Jens	1 3 5 6 8
8	Darius	1 2 3 5 6 7

GraphX PageRank

```
val graph = GraphLoader.edgeListFile(sc, "followers.txt")

val ranks = graph.pageRank(0.0001).vertices

val users = sc.textFile("users.txt").map { line =>

 val fields = line.split(",")

 (fields(0).toLong, fields(1))

}
```

GraphX PageRank

```
val ranksByUsername = users.join(ranks).map {

 case (id, (username, rank)) => (username, rank)

}

ranksByUsername.map(_.swap).top(7).foreach(println)
```

GraphX PageRank Results

```
(1.7768988260358134,darius)

(1.3087233766354789,jens)

(1.2661395633749972,martin)

(0.7706521076429856,jennifer)

(0.6801921696513588,johannes)

(0.6751418105342486,florian)

(0.5164601769253268,matti)
```

GraphX PageRank Results

```
(1.7768988260358134,darius)

(1.3087233766354789,jens)

(1.2661395633749972,martin)

(0.7706521076429856,jennifer)

(0.6801921696513588,johannes)

(0.6751418105342486,florian)

(0.5164601769253268,matti)
```

Be number 1 in test data is just...

Awesome!

Spark SQL

Apache Spark Release - Overview

1.0.0	Add Spark SQL
1.2.0	MLlib Pipelines, GraphX Stable, External Data Sources
1.3.0	DataFrames API, Spark SQL Stabile
1.4.0	Add Support for R
1.6.0	DataSets API, SQL on Files
2.0.0 (preview)	● SparkSession replace SQLContext ● ...

Spark SQL

- Write Hive SQL Queries
- Use the benefits of Spark while processing your Statements
- Returns an RDD

DataFrames

- New since 1.3.0, returned by Spark SQL
- data organized into named columns
- conceptually equivalent to a RDMS table
- Converts from and to RDD

Spark SQL

```
val df = sqlContext.sql("SELECT * FROM table")
```

RDD to DataFrames

```
case class FreeDB(

  discId: String,

  discTitle: String,

  year: String,

  genre: String,

  titles: String

)
```

RDD to DataFrames

```
val lines = sc.textFile("folk.csv").cache()

val mappedResult = lines.map(line => {

 val splitted = line.split(",")

 val discId = splitted(0)

 val discTitle = ...

 FreeDB(discId,discTitle,year,genre,titles)

})
```

RDD to DataFrames

```
val sqlContext = new SQLContext(sc)

import sqlContext.implicits._

val df = mappedResult.toDF()

println(df.show())

println(df.printSchema())
```

RDD to DataFrames (show)

```
+--------+--------------------+----+--------------+---------------------+
|  discId|           discTitle|year|         genre|               titles|
+--------+--------------------+----+--------------+---------------------+
|cd11750f|Broery Marantika ...|    |              |hati yang terluka...|
|ae0e371d|Anders Bjernulf /...|2001|          Folk|"Nyåkers Mor" Ann...|
|2a02c704|Mary Karlzen / I'...|1994|          Folk|I'd Be Lying Run ...|
|a50b350c|Various / ACOUSTI...|2009|          FOLK|What A Wonderful ...|
|c409730d|The Glacial Errat...|    |              |Highwind Alive an...|
```

RDD to DataFrames (schema)

```
root

 |-- discId: string (nullable = true)

 |-- discTitle: string (nullable = true)

 |-- year: string (nullable = true)

 |-- genre: string (nullable = true)

 |-- titles: string (nullable = true)
```

SQL on Files

```
val df = sqlContext.sql("SELECT * FROM parquet.
`examples/src/main/resources/users.parquet`")
```

Query DataFrames

```
df.filter(df("year") === "1987").show()

df.filter((df("genre") === "folk").and(df("year") ===
"1987")).show()
```

RDD to DataFrames (schema)

```
+--------+--------------------+----+-----+--------------------+
|  discId|           discTitle|year|genre|              titles|
+--------+--------------------+----+-----+--------------------+
|8408460a|   Melanie / Melanie|1987| folk|Rock and roll hea...|
|5c091009|Adriano Celentano...|1987| folk|Ready Teddy L'alb...|
|ec0d0011|Dominig Bouchaud ...|1987| folk|Gwerz Penmarc'h T...|
|8f088a0c|Eric Weber Quarte...|1987| folk|Jingle Bells Stil...|
|74099c09|å ‹éƒ¨æ–£ä°° / å…...|1987| folk|å…–æœˆã ®é›¨ã ®å¤...|
```

DataSets

- New since 1.6.0, experimental
- Not use Java Serialization or Kryo
- Use Encoder
- code generated dynamically
- perform many operations without deserializing

DataSets

```
val sqlContext = new SQLContext(sc)

import sqlContext.implicits._



val list = (1 to 1000000).toList

val dataSet = list.toDS()

dataSet.map(_ * 2).take(100).foreach(println)
```

Example Stack

Development

- Configuration using docker-compose

Sandbox

- Every Service run in a docker container
- Running containers using rancher: http://rancher.com/
- Rancher running on two hosts for load balancing and scaling
- Hadoop 2.7.2 - Hive 2.0.0 - Spark 1.6.1

Resources

- http://statrgy.com/2015/06/04/spark-streaming-simple-example-streaming-data-from-hdfs/

  (Spark Streaming Code)
- https://spark.apache.org/docs/1.6.1/ (Most Examples)
- freedb.org - Demo Data
- Spark 2.0.0 https://issues.apache.org/jira/browse/SPARK/fixforversion/12329449/
- Sourcecode and Slides: https://github.com/dariusgm/scala-meetup-2016

Get a beer or a coke

Ask Questions

Get in Contact

And thank you for your Time