Brave New World: Hadoop vs. Spark

Dr. Kurt Stockinger
Associate Professor of Computer Science
Director of Studies in Data Science
Zurich University of Applied Sciences

Datalab Seminar, Zurich, Oct. 7, 2015

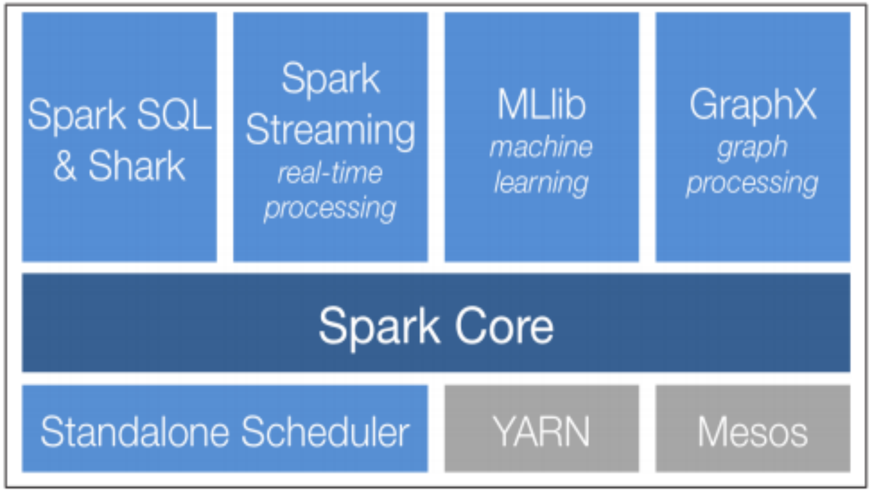
Outline

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Motivation: Main Memory Processing
- Spark:
  - Map Reduce
  - SQL processing
  - Streaming processing
  - Machine learning
  - Graph processing
- ZHAW Big Data project & industry trends

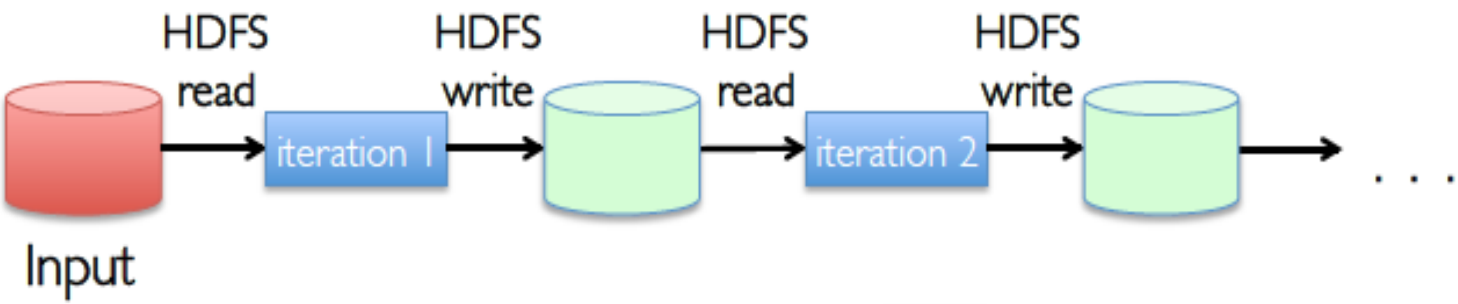
What is Apache Spark ?

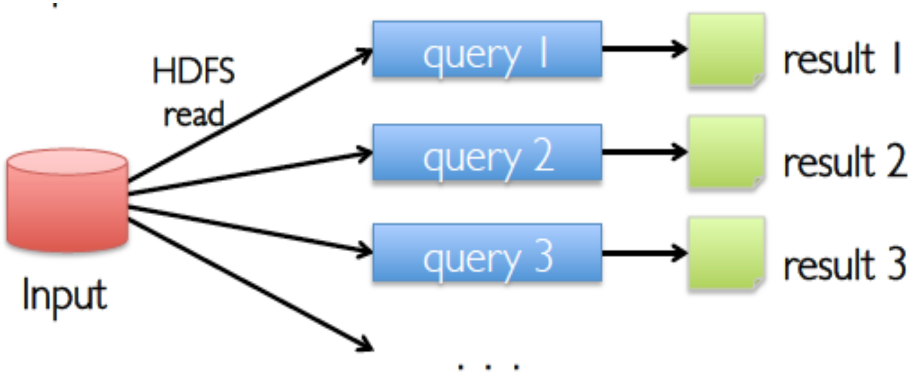
- General purpose cluster computing system
- Originally developed at UC Berkeley, now one of the largest Apache projects
- Typically faster than Hadoop due to main-memory processing
- High-level APIs in Java, Scala, Python and R
- Functionality for:
  - Map/Reduce
  - SQL processing
  - Real-time stream processing
  - Machine learning
  - Graph processing



Iterative Processing with Hadoop

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie





What is wrong with this approach?

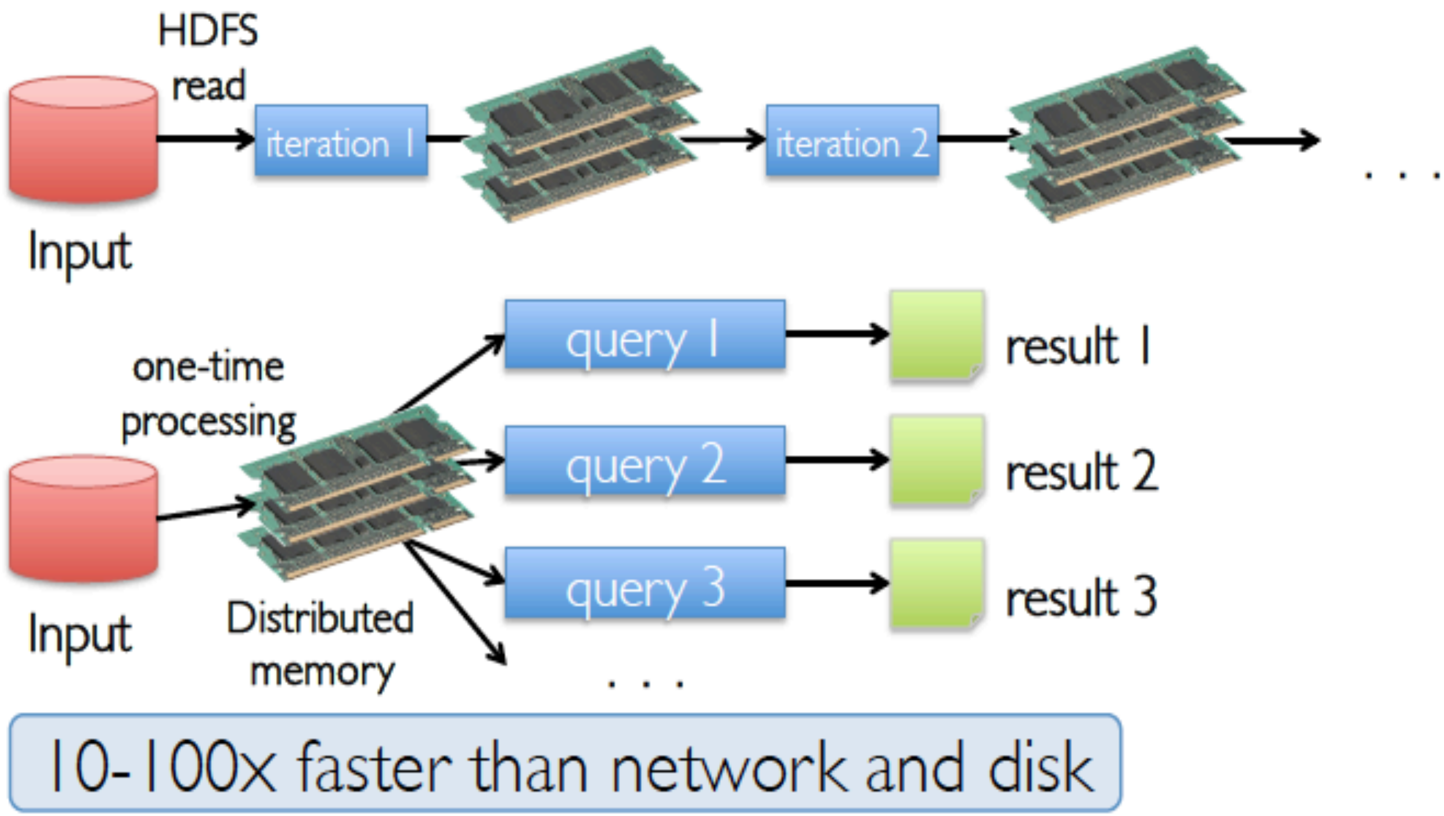
Throughput of Main Memory vs. Disk

- Typical throughput of disk: ~ 100 MB/sec

- Typical throughput of main memory: 50 GB/sec

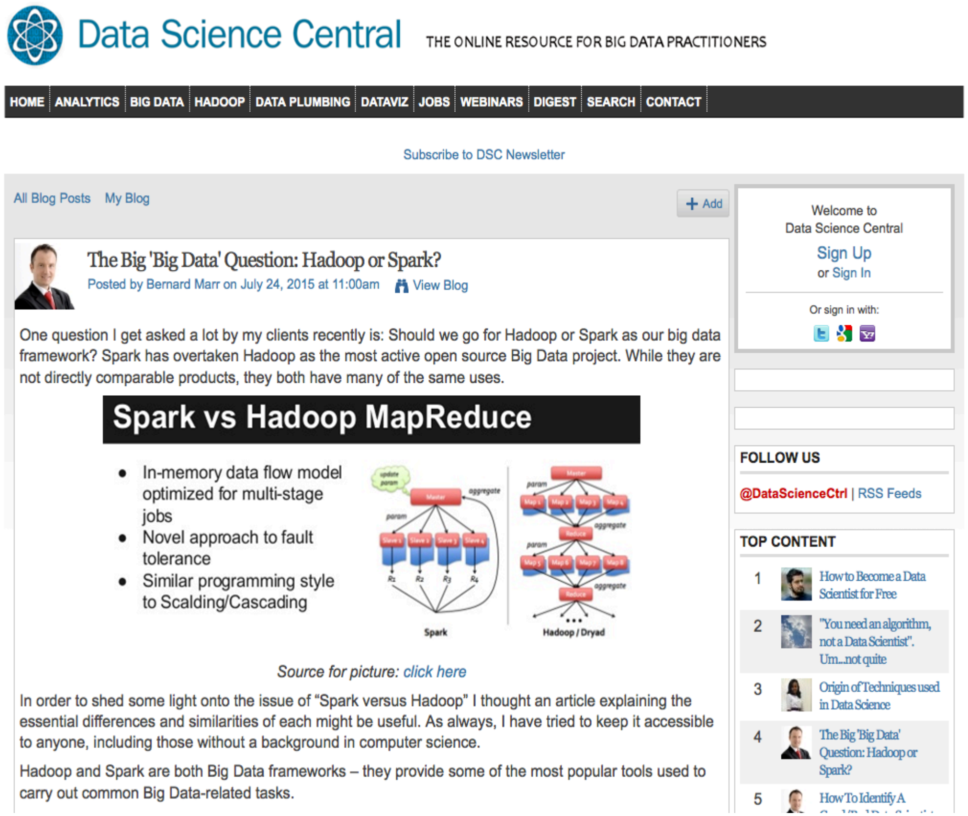
=> Main memory is ~ 500 times faster than disk

Apache Spark Approach: In-Memory Data Sharing



(from Matei Zaharia 2012, UC Berkeley)

Spark vs. Hadoop #1



Data Science Central THE ONLINE RESOURCE FOR BIG DATA PRACTITIONERS

HOME | ANALYTICS | BIG DATA | HADOOP | DATA PLUMBING | DATAVIZ | JOBS | WEBINARS | DIGEST | SEARCH | CONTACT

Subscribe to DSC Newsletter

All Blog Posts My Blog

Add

The Big 'Big Data' Question: Hadoop or Spark?

Posted by Bernard Marr on July 24, 2015 at 11:00am View Blog

One question I get asked a lot by my clients recently is: Should we go for Hadoop or Spark as our big data framework? Spark has overtaken Hadoop as the most active open source Big Data project. While they are not directly comparable products, they both have many of the same uses.

Source for picture: click here

In order to shed some light onto the issue of "Spark versus Hadoop" I thought an article explaining the essential differences and similarities of each might be useful. As always, I have tried to keep it accessible to anyone, including those without a background in computer science.

Hadoop and Spark are both Big Data frameworks – they provide some of the most popular tools used to carry out common Big Data-related tasks.

Welcome to Data Science Central

Sign Up

or Sign In

Or sign in with:

FOLLOW US

@DataScienceCtrl | RSS Feeds

TOP CONTENT

1 How to Become a Data Scientist for Free

2 "You need an algorithm, not a Data Scientist". Um...not quite

3 Origin of Techniques used in Data Science

4 The Big 'Big Data' Question: Hadoop or Spark?

5 How To Identify A

http://www.datasciencecentral.com/profiles/blogs/the-big-big-data-question-hadoop-or-spark

Spark vs. Hadoop #2

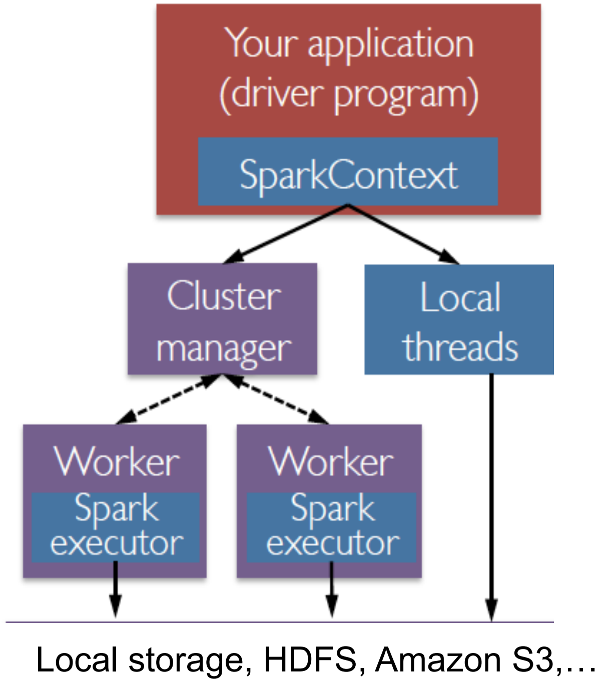
Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

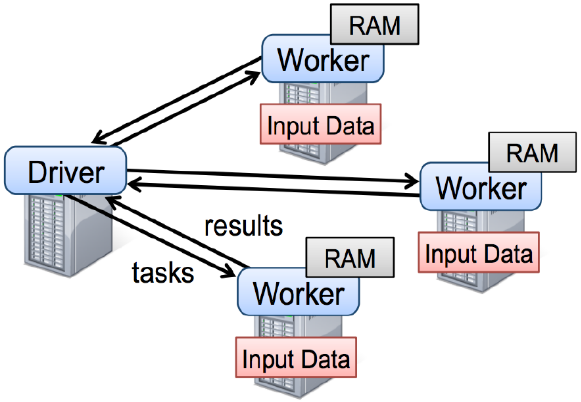
	Hadoop Map Reduce	Spark
Storage	Disk only	In-memory or on disk
Operations	Map and Reduce	Map, Reduce, Join, Sample, etc…
Execution model	Batch	Batch, interactive, streaming
Programming environments	Java	Scala, Java, R, and Python

(from Ameet Talwalkar, UCLA, 2015)

Spark Runtime

- **Driver:**
  - Application developer writes driver program
  - Driver connects to cluster of workers
- **Workers:**
  - Read data blocks from distributed file system
  - Process and store data partitions in RAM across operations





Resilient Distributed Dataset (RDD)

- **Resilience**:

> Resilience is the ability of the network to provide and maintain an **acceptable level of service** in the face **of various faults and challenges** to normal operation. (Wikipedia)

  Fault tolerance

- **Resilient Distributed Dataset (RDD):**
  - Immutable, partitioned collections of objects spread across a cluster
  - Controllable persistence: stored in memory or on disk
  - Built and manipulated through:
    - Parallel transformations (map, filter, join)
    - Parallel actions (count, collect, save)
  - Automatically rebuilt on machine failure

Spark Transformations and Actions

Zürcher Hochschule
für Angewandte Wissenschaften
zh
aw
School of Engineering
InIT Institut für angewandte Informationstechnologie

Transformations	$map(f : \mathrm{T} \Rightarrow \mathrm{U})$	:	$\mathrm{RDD[T]} \Rightarrow \mathrm{RDD[U]}$
	$filter(f : \mathrm{T} \Rightarrow \mathrm{Bool})$	:	$\mathrm{RDD[T]} \Rightarrow \mathrm{RDD[T]}$
	$flatMap(f : \mathrm{T} \Rightarrow \mathrm{Seq[U]})$	:	$\mathrm{RDD[T]} \Rightarrow \mathrm{RDD[U]}$
	$sample(fraction : \mathrm{Float})$	:	$\mathrm{RDD[T]} \Rightarrow \mathrm{RDD[T]}$ (Deterministic sampling)
	$groupByKey()$	:	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, Seq[V])]}$
	$reduceByKey(f : (\mathrm{V}, \mathrm{V}) \Rightarrow \mathrm{V})$	:	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, V)]}$
	$union()$	:	$(\mathrm{RDD[T]}, \mathrm{RDD[T]}) \Rightarrow \mathrm{RDD[T]}$
	$join()$	:	$(\mathrm{RDD[(K, V)]}, \mathrm{RDD[(K, W)]}) \Rightarrow \mathrm{RDD[(K, (V, W))]}$
	$cogroup()$	:	$(\mathrm{RDD[(K, V)]}, \mathrm{RDD[(K, W)]}) \Rightarrow \mathrm{RDD[(K, (Seq[V], Seq[W]))]}$
	$crossProduct()$	:	$(\mathrm{RDD[T]}, \mathrm{RDD[U]}) \Rightarrow \mathrm{RDD[(T, U)]}$
	$mapValues(f : \mathrm{V} \Rightarrow \mathrm{W})$	:	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, W)]}$ (Preserves partitioning)
	$sort(c : \mathrm{Comparator[K]})$	:	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, V)]}$
	$partitionBy(p : \mathrm{Partitioner[K]})$	:	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{RDD[(K, V)]}$
Actions	$count()$	:	$\mathrm{RDD[T]} \Rightarrow \mathrm{Long}$
	$collect()$	:	$\mathrm{RDD[T]} \Rightarrow \mathrm{Seq[T]}$
	$reduce(f : (\mathrm{T}, \mathrm{T}) \Rightarrow \mathrm{T})$	:	$\mathrm{RDD[T]} \Rightarrow \mathrm{T}$
	$lookup(k : \mathrm{K})$	:	$\mathrm{RDD[(K, V)]} \Rightarrow \mathrm{Seq[V]}$ (On hash/range partitioned RDDs)
	$save(path : \mathrm{String})$	:	Outputs RDD to a storage system, *e.g.*, HDFS

(from Matei Zaharia et al. 2012, UC Berkeley)

Outline

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Motivation: Main Memory Processing
- Spark:
  - Map Reduce
  - SQL processing
  - Streaming processing
  - Machine learning
  - Graph processing
- ZHAW Big Data project & industry trends

Word Count Example

Zürcher Hochschule
für Angewandte Wissenschaften
zh aw
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Hadoop 2.0:
  - Java: ~130 lines
  - Python: 2 programs, ~30 lines

- Spark:
  - Python

```
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://...")
```

Spark + HDFS

- HDFS: Hadoop Distributed Files System
  - Fault-tolerant distributed file system
  - One of the main strengths: data replication

- >15 years back in time:
  - Data replication was a major research topic at CERN

Data Management in an International Data Grid Project

Wolfgang Hoschek[1,3], Javier Jaen-Martinez[1], Asad Samar[1,4], Heinz Stockinger[1,2], and Kurt Stockinger[1,2]

[1] CERN, European Organization for Nuclear Research, Geneva, Switzerland
[2] Inst. for Computer Science and Business Informatics, University of Vienna, Austria
[3] Inst. of Applied Computer Science, University of Linz, Austria
[4] California Institute of Technology, Pasadena, CA, USA

W. Hoschek, J. J. -Martinez,Asad Samar, H. Stockinger, K. Stockinger. Data Management in an International Data Grid Project. ACM International Workshop on Grid Computing (Grid 2000), Bangalore, India, December 2000, IEEE Computer Society Press (distinguished paper award)

- Spark can leverage this Hadoop technology

Spark SQL Processing Example

Load data from text file into RDD, create table, and query it:

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row

sc = SparkContext(appName="SparkSQLPeople")
sqlContext = SQLContext(sc)

# Load a text file and convert each line to a Row.
lines = sc.textFile("data/people.txt", 4)
words = lines.map(lambda l: l.split(","))
people = words.map(lambda p: Row(name=p[0], age=int(p[1])))

schemaPeople = sqlContext.createDataFrame(people) schemaPeople.registerTempTable("peopleT")

# SQL can be run over SchemaRDDs that have been registered as a table.
teenagers = sqlContext.sql("SELECT name
                            FROM peopleT
                            WHERE age >= 13 AND age <= 19")

teenagers.collect()
```

The csv file people.txt looks as follows:
Michael, 29
Andy, 30
Justin, 19

Reflections on SQL Processing

Zürcher Hochschule
für Angewandte Wissenschaften
zh
aw
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Pros:
  - Good integration with core Spark concepts (RDD)
  - Easy to use
  - Integration with Apache Hive ("Hadoop Data Warehouse")

- Cons:
  - No full support of SQL yet (no views, indexing, etc.)
  - No easy overview of schema
  - "Cloudera is not happy about it"

Spark Stream Processing Example #1


```
// Create the context with a 2 second batch size
val sparkConf = new SparkConf().setAppName("SqlNetworkWordCount")
val ssc = new StreamingContext(sparkConf, Seconds(2))

// Create a socket stream on target ip:port and count the
// words in input stream of \n delimited text (eg. generated by 'nc')
// Note that no duplication in storage level only for running locally.
// Replication necessary in distributed scenario for fault tolerance.
val lines = ssc.socketTextStream(args(0), args(1).toInt, StorageLevel.MEMORY_AND_DISK_SER)
val words = lines.flatMap(_.split(" "))
```

Spark Stream Processing Example #2

Zürcher Hochschule
für Angewandte Wissenschaften
zh aw
School of Engineering
InIT Institut für angewandte Informationstechnologie

```scala
// Convert RDDs of the words DStream to DataFrame and run SQL query
words.foreachRDD((rdd: RDD[String], time: Time) => {
  // Get the singleton instance of SQLContext
  val sqlContext = SQLContextSingleton.getInstance(rdd.sparkContext)
  import sqlContext.implicits._

  // Convert RDD[String] to RDD[case class] to DataFrame
  val wordsDataFrame = rdd.map(w => Record(w)).toDF()

  // Register as table
  wordsDataFrame.registerTempTable("words")

  // Do word count on table using SQL and print it
  val wordCountsDataFrame =
    sqlContext.sql("select word, count(*) as total from words group by word")
  println(s"========= $time =========")
  wordCountsDataFrame.show()
})
```

Reflections on Stream Processing

Zürcher Hochschule
für Angewandte Wissenschaften
zh
aw
School of
Engineering
InIT Institut für angewandte
Informationstechnologie

- Pros:
  - Streaming functionality well integrated with core Spark (RDD)
  - Similar functionality to Apache Storm but simpler API

- Cons:
  - Main support currently only for Scala and Java
  - Python?

Spark Machine Learning Example

```
from pyspark.mllib.classification import LogisticRegressionWithLBFGS
from pyspark.mllib.regression import LabeledPoint
from numpy import array

# Load and parse the data
def parsePoint(line):
    values = [float(x) for x in line.split(' ')]
    return LabeledPoint(values[0], values[1:])

data = sc.textFile("data/mllib/sample_svm_data.txt")
parsedData = data.map(parsePoint)

# Build the model
model = LogisticRegressionWithLBFGS.train(parsedData)

# Evaluating the model on training data
labelsAndPreds = parsedData.map(lambda p: (p.label, model.predict(p.features)))
trainErr = labelsAndPreds.filter(lambda (v, p): v != p).count() / float(parsedData.count())
print("Training Error = " + str(trainErr))
```

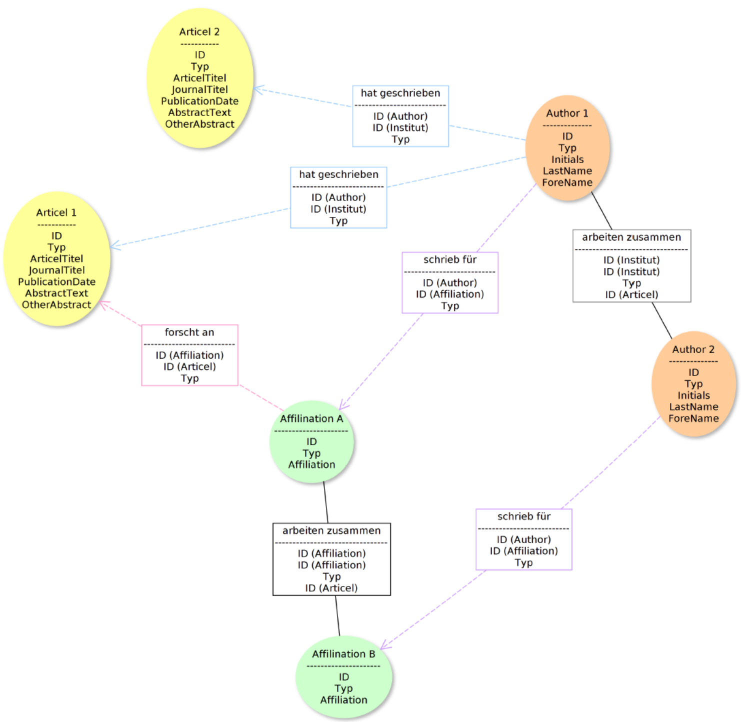
Reflections on Machine Learning

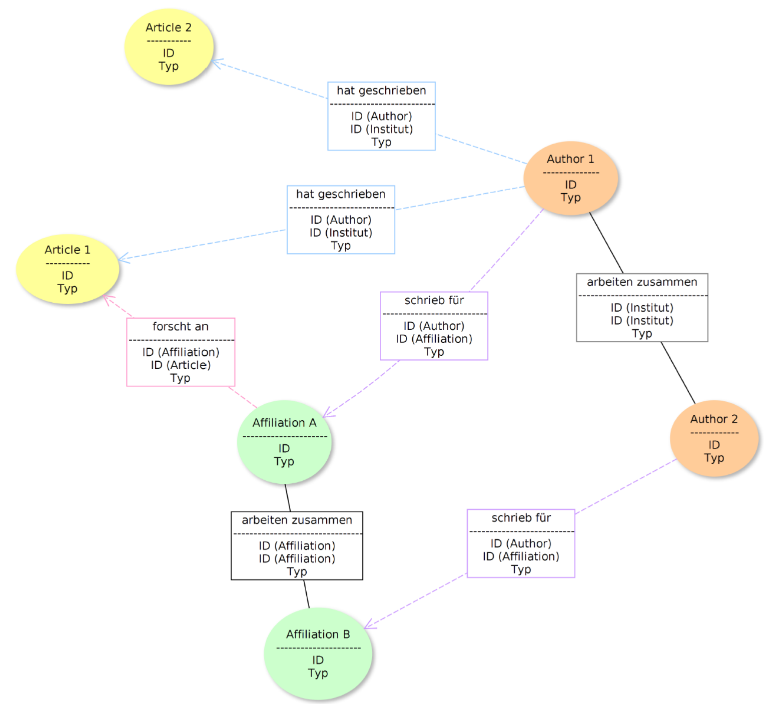
- Pros:
  - Many machine learning algorithms are iterative such as gradient descent or clustering
  - Can fully take advantage of main memory architecture
  - Similar functionality to Apache Mahout

- Cons:
  - ?

Spark Graph Processing Example

- Use Case: Analysis of author graph in Medline database





- Source: Axel Rogg, Analyse von strukturierten und unstrukturierten Daten mit Apache Spark, Bachelor Thesis, ZHAW, June 2015

Reflections on Graph Processing

- Pros:
  - Powerful graph processing library
  - SQL integration into graph algorithms
  - Similar functionality to Apache Giraph

- Cons:
  - Main support mostly for Scala
  - Graph processing only graphs with “simple” nodes

Outline

- Motivation: Main Memory Processing

- Spark:
  - Map Reduce
  - SQL processing
  - Streaming processing
  - Machine learning
  - Graph processing

- ZHAW Big Data project & industry trends

Data-Driven Financial System Modeling: Motivation: Crisis of 2007-2009

Zürcher Hochschule für Angewandte Wissenschaften

School of Engineering

InIT Institut für angewandte Informationstechnologie

US Secretary of the Treasury Hank Paulson in September 2008 (when Lehman Brother filed for bankruptcy):

*"The problems at Lehman have been known for many months. The counterparties have had ample opportunity to adjust their exposure. Therefore, we can **safely** let Lehman go down."*

The Economist

Free exchange
Economics

European stress tests
Let's try again

Apr 29th 2014, 11:59 by P.W. | LONDON

"THIS time is different" is a slogan usually to be found on the lips of bullish financiers and investors. But in effect this is the message that European regulators are trying to send as they set out how this year's banking stress tests really will mark a break from the past. Today (http://www.eba.europa.eu/) the European Banking Authority (EBA), which is responsible for coordinating the tests across the 28-country EU, gave more details on how they will work.

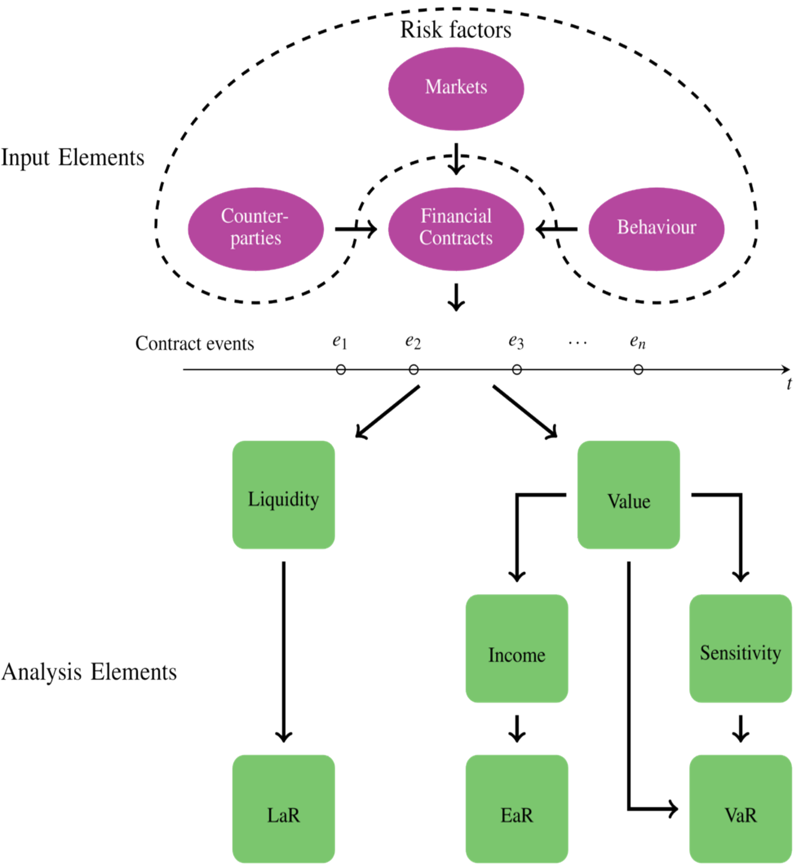
ZHAW project to make stress tests and risk expose calculations comparable across banks

Collaboration with regulators as European Central Bank, Bank of England

ZHAW Project Algorithmic Contract Type Unifying Standard

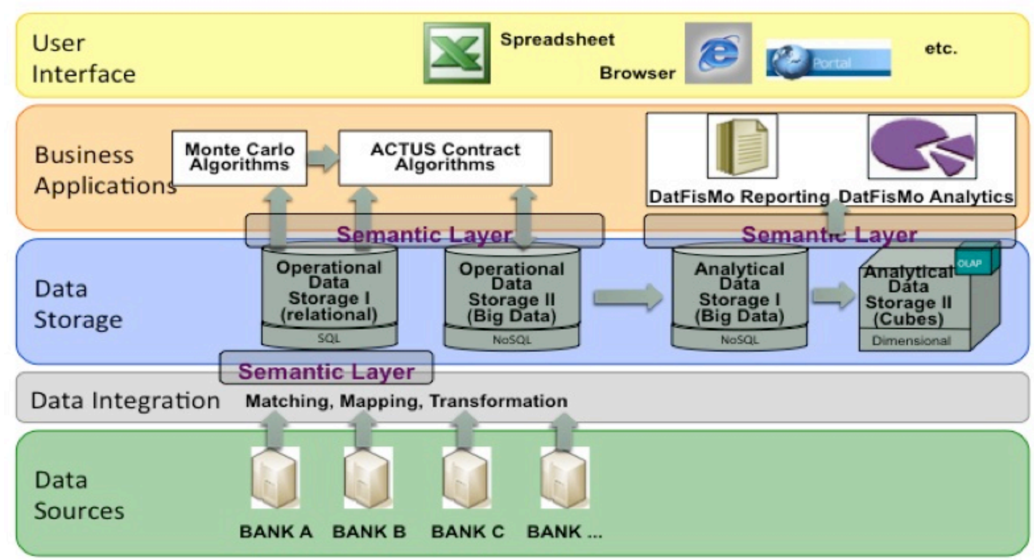
ACTUS
ALGORITHMIC CONTRACT TYPES UNIFIED STANDARDS

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie



Brammertz et al., *Unified Financial Analysis*. Chichester, 2009.

DatFisMo ICT Architecture



Project requires **interdisciplinary approach:**

- Finance policies, mathematics, economics
- Data warehousing, big data, cloud computing
- Parallel programming
- Security

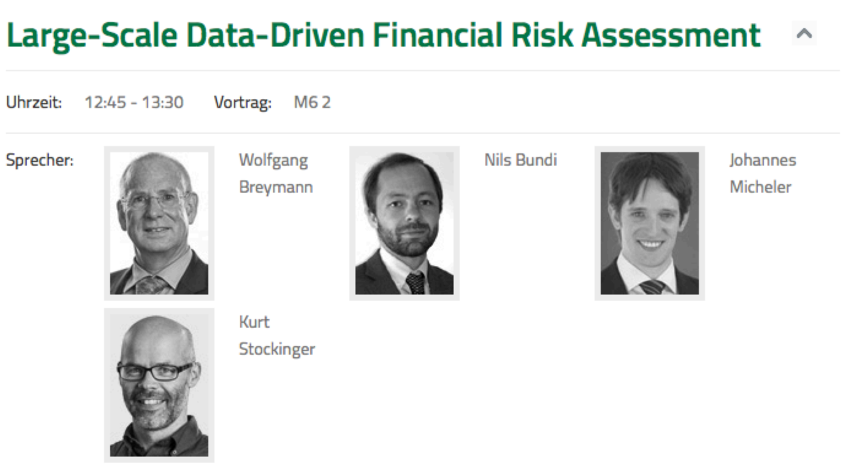
ZHAW Project:
Data-Driven Financial Risk Modeling

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Implemented in Hadoop and simulated cash flows of millions of financial contracts

- Re-implement in Apache Spark leveraging HDFS + Hive

- More info at DW 2015, November in Zurich
  - http://www.dw-konferenz.ch/dw2015/konferenz/konferenzprogramm/vortrag/m6/title/large-scale-data-driven-financial-risk-assessment.html



Trends from Industry

Zürcher Hochschule
für Angewandte Wissenschaften
zh aw School of Engineering
InIT Institut für angewandte Informationstechnologie

- Personal impressions & insights from ISC Cloud & Big Data (Frankfurt, Sept 29-30, 2015)
  - Hadoop is gaining traction in Europe
  - Spark is picking up
  - Intel seems to be focused on Hadoop
  - IBM Research Zurich is working on Spark for RDMA
  - Swisscom has heavily embarked on Spark (especially for streaming)
  - Many companies use the concept of Data Lake:
    - "Wild" collection of "data management/processing tools"
    - Data Warehousing technology
    - Big Data technology (Hadoop, Spark,…)

Conclusions

- Spark combines functionality that is spread across various Apache Big Data tools:
  - MapReduce
  - SQL processing
  - Stream processing
  - Graph processing
    Machine Learning

- Main concept: resilient distributed dataset (RDD)
- Main memory architecture is well-suited for iterative data processing
- For Big and Small Data
- Spark can leverage the strengths of HDFS + Hive

- Contact: Kurt.Stockinger@zhaw.ch