Cloud Computing & Big Data

PARALLEL & SCALABLE MACHINE LEARNING & DEEP LEARNING

Prof. Dr. – Ing. Morris Riedel
Associated Professor
School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland
Research Group Leader, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany

@Morris Riedel @MorrisRiedel @MorrisRiedel

LECTURE 3

Apache Spark for Cloud Applications

September 22, 2020
Online Lecture

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

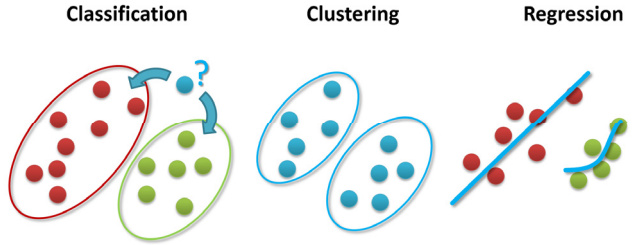
JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

DEEP Projects

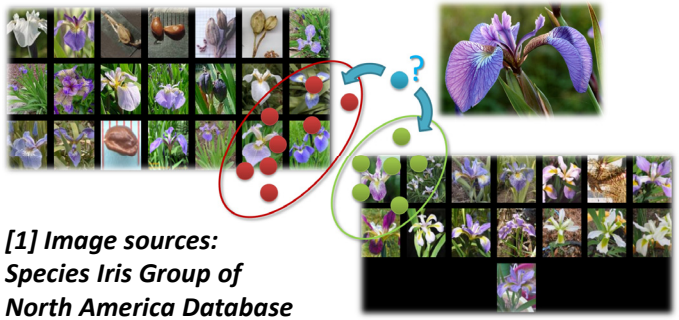
HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Review of Lecture 2 – Machine Learning Models in Clouds

Machine Learning Fundamentals

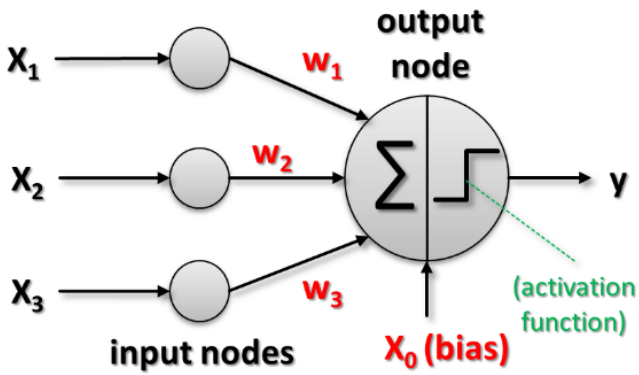


1. Some pattern exists
2. No exact mathematical formula
3. Data exists



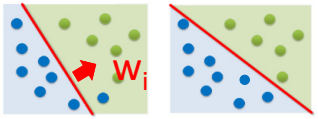
[1] Image sources: Species Iris Group of North America Database

'Simple' Perceptron Learning Model



$$h(\mathbf{x}) = sign\left(\left(\sum_{i=0}^{d} w_i x_i\right)\right); x_0 = 1$$

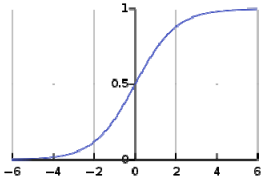
$$h(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x})$$



Logistic Regression Model

$$h(\mathbf{x}) = s\,(\mathbf{w}\cdot\mathbf{x})$$

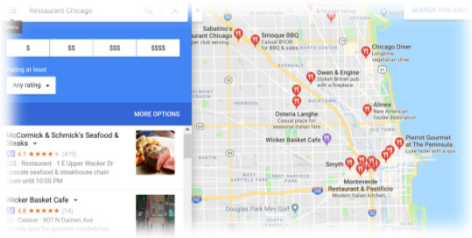
- Binary Classification Model




```
In [15]: tokenizer = Tokenizer(inputCol="violations", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.01)
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

model = pipeline.fit(labeledData)
model

PipelineModel_4ee589dd5c6759dd2297
```

jupyter

APACHE Spark



[2] Microsoft Azure HDInsight Service

Outline of the Course

1. Cloud Computing & Big Data Introduction
2. Machine Learning Models in Clouds
3. Apache Spark for Cloud Applications
4. Virtualization & Data Center Design
5. Map-Reduce Computing Paradigm
6. Deep Learning driven by Big Data
7. Deep Learning Applications in Clouds
8. Infrastructure-As-A-Service (IAAS)
9. Platform-As-A-Service (PAAS)
10. Software-As-A-Service (SAAS)
11. Big Data Analytics & Cloud Data Mining
12. Docker & Container Management
13. OpenStack Cloud Operating System
14. Online Social Networking & Graph Databases
15. Big Data Streaming Tools & Applications
16. Epilogue

+ additional practical lectures & Webinars for our hands-on assignments in context

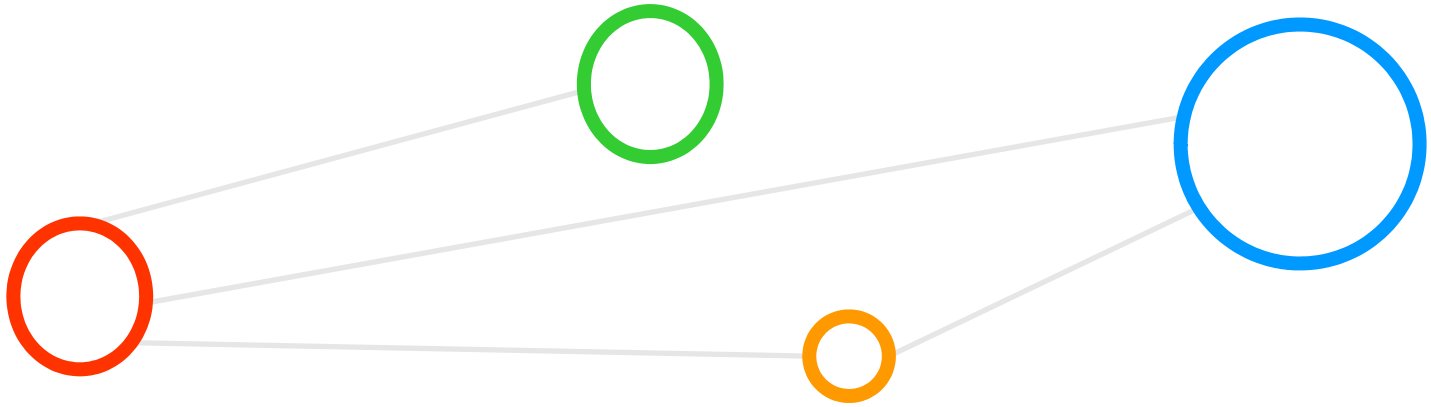
- Practical Topics
- Theoretical / Conceptual Topics

Outline

- Basic Apache Spark Capabilities
  - Design Goals & Scalable Programming Model for Big Data
  - Resilient Distributed Datasets (RDDs) Concept & Example
  - Understanding Apache Spark RDD Transformations & Actions
  - Analysing Apache Spark vs. Apache Hadoop Parallel Performance
  - In-Memory Execution with Applications & AWS Costs Reviews
- Apache Spark Libraries for Advanced Applications
  - Structured Query Language (SQL) via PySpark in Jupyter Notebooks
  - Machine Learning Library (MLlib) & Selected Algorithm Examples
  - Streaming Library & GraphX Library with PageRank Application
  - Machine Learning Applications in Clouds & PySpark in Google Colab
  - Examples of Apache Spark in Business Applications

- **Promises from previous lecture(s):**
- ***Lecture 1:*** **Lecture 3 provides more details on how the Apache Spark system works and how it is used in cloud computing applications today**
- ***Lecture 2:*** **Lecture 3 & 6 provides more insights into Logistic Regression & optimization techniques used in machine learning model training phase**
- ***Lecture 2:*** **Lecture 3 provides more details about the underlying Apache Spark and Apache Hadoop distributed Big Data Analytics technologies**
- ***Lecture 2:*** **Lecture 3 provides more details about the relevance of storage cloud resources to perform Big Data Analytics with Apache Spark tools**
- ***Lecture 2:*** **Lecture 3 provides details about underlying Apache Spark technologies and the meaning of Head and Worker nodes in a cloud cluster**
- ***Lecture 2:*** **Lecture 3 provides an introduction to Apache Spark with SparkSession, SparkContext and the relevance of the YARN resource manager**
- ***Lecture 2:*** **Lecture 3 provides a detailed introduction to Apache Spark data management & handling as well as its Spark SQL dataframe concepts**
- ***Lecture 2:*** **Lecture 3 provides details about Apache Spark and other available Spark MLlib models & pipelines used for machine learning in clouds**
- ***Lecture 2:*** **Lecture 3 provides more interesting examples on training sets & test datasets and other machine learning model evaluation techniques**

Basic Apache Spark Capabilities



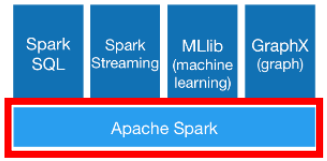
Networking & Big Data Impacts on Cloud Computing – Revisited (cf. Lecture 1)

- Requirements for scalable programming models and tools
  - CPU speed has surpassed IO capabilities of existing cloud resources
  - Data-intensive clouds with advanced analytics and analysis capabilities
  - Considering moving compute task to data vs. moving data to compute
  - E.g., MS Azure HDInsight is only one concrete example of many cloud solutions

Microsoft Azure HDInsight

[2] Microsoft Azure HDInsight Service

  - E.g., clouds often use open source implementations like Apache Spark & Apache Hadoop & related technologies



[4] Apache Spark

- **Apache Spark is a fast & general engine for large-scale data processing optimized for memory usage**
- **Open source & compatible with cloud data storage systems like Amazon Web Services (AWS) S3**
- **Apache Spark is often used as cloud service offerings in commercial clouds like MS Azure, AWS, or Google Cloud**

- Requirements for Reliable Filesystems
  - Traditional parallel filesystems need to prove their 'big data' feasibility
  - Emerging new forms of filesystems that assume hardware error constantly
  - E.g. Hadoop distributed file system (HDFS)

[3] Apache Hadoop

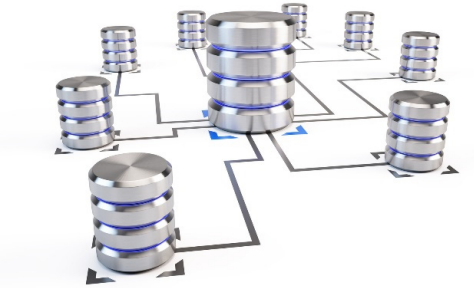
➢ **Lecture 5 provides more details on the Hadoop Distributed File System (HDFS) that is often used in conjunction with Apache Spark**

Big Data Analytics Frameworks – Revisited (cf. Lecture 1)

- Distributed Processing
  - ‘Map-reduce via files‘: Tackle large problems with many small tasks
  - Advantage of ‘data replication‘ via specialized distributed file system
  - E.g. Apache Hadoop

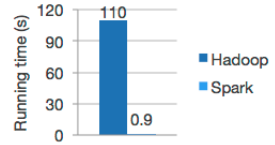
[27] Apache Hadoop

(does it make sense to transfer TBs/PBs again and again for applications?)



- In-Memory Processing
  - Perform many operations fast via ‘in-memory‘
  - Enable tasks such as ‘map-reduce in-memory‘
  - Needs hardware systems that offer large memory
  - E.g. Apache Spark, Apache Flink

[28] Apache Spark



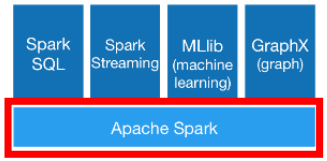
Logistic regression in Hadoop and Spark

- **Big Data analytics frameworks shift the approach from ‘bring data to compute resources‘ into ‘bring compute tasks close to data‘ including infrastructure technologies (e.g., Apache Hadoop and/or Apache Spark)**

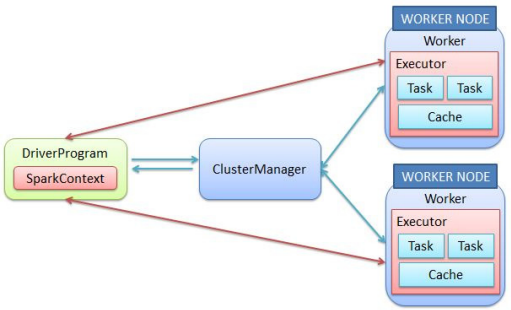
➢ **Lecture 5 provides more details on the Hadoop Distributed File System (HDFS) that is often used in conjunction with Apache Spark**

Apache Spark Basic Properties

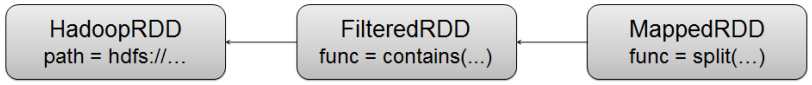
- Production deployments
  - Adobe, Intel and Yahoo have production deployments
- Open Source (initially invented by UC Berkeley)
  - One of the 'most active big data' open source projects today
- Idea: Support two properties beyond traditional 'map-reduce'
  - Extension towards Directed Acyclic Graphcs (DAGs)
  - Improvements on 'data sharing capabilities' & 'in-memory computing'
- Key Concepts: 'Resilient Distributed Datasets (RDDs)' & Usability
  - RDDs are a fault-tolerant collection of elements to operate on in parallel
  - Good usability w.r.t. write less code via rich APIs
  - Scalability by applying more worker nodes to problems
  - Design goal is to enable easy programming
  - Creating 'pipelines' that work on datasets



[4] Apache Spark




```
messages = textFile(...).filter(lambda s: s.contains("ERROR"))
                        .map(lambda s: s.split('\t')[2])
```



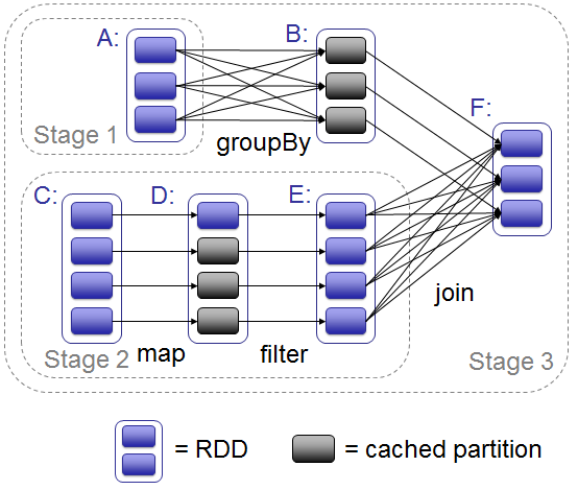
[5] Parallel Programming with Spark

[24] Understanding Parallelization of Machine Learning Algorithms in Apache Spark

Resilient Distributed Datasets (RDDs)

- Key Approach
  - Immutable collections of objects across a cluster (i.e., partitions)
  - Created through parallel transformations (e.g., map, filter, group, etc.)
  - Controllable persistence & Cached partition
    (e.g., caching in RAM, if it fits in memory)

```
messages = textFile(...).filter(lambda s: s.contains("ERROR"))
                        .map(lambda s: s.split('\t')[2])
```



- **The key idea of Resilient Distributed Datasets (RDDs) in Apache Spark is that they enable the work with distributed data collections as if one would work with local data collections**
- **RDDs automatically rebuilt on failure: RDD tracks the transformations that created them (i.e., lineage) to recompute lost data**
- **RDDs can be created through parallel transformations such as map, filter, group, etc. and use caching in memory if data fits into memory**

[5] Parallel Programming with Spark

RDD Transformations – Example

(domain decomposition)

(create a parallelized collection holding the numbers 1,2, and 3 in 3 different slices, the size can be configured for bigger data)

```
nums = sc.parallelize([1, 2, 3])

# Pass each element through a function
squares = nums.map(lambda x: x*x)   # => {1, 4, 9}

# Keep elements passing a predicate
even = squares.filter(lambda x: x % 2 == 0) # => {4}

# Map each element to zero or more others
nums.flatMap(lambda x: range(0, x))  # => {0, 0, 1, 0, 1, 2}
```

(note: this is computed on nums, not squares or even!)

Range object (sequence of numbers 0, 1, …, x-1)

(starting with 0, not x itself!)

[5] Parallel Programming with Spark

- In Apache Spark on can use 'lambda' that enables a small anonymous functions to be defined, e.g. f(x) = x*x, and returns directly the result or outcome of the function in the corresponding expression

RDD Actions – Example

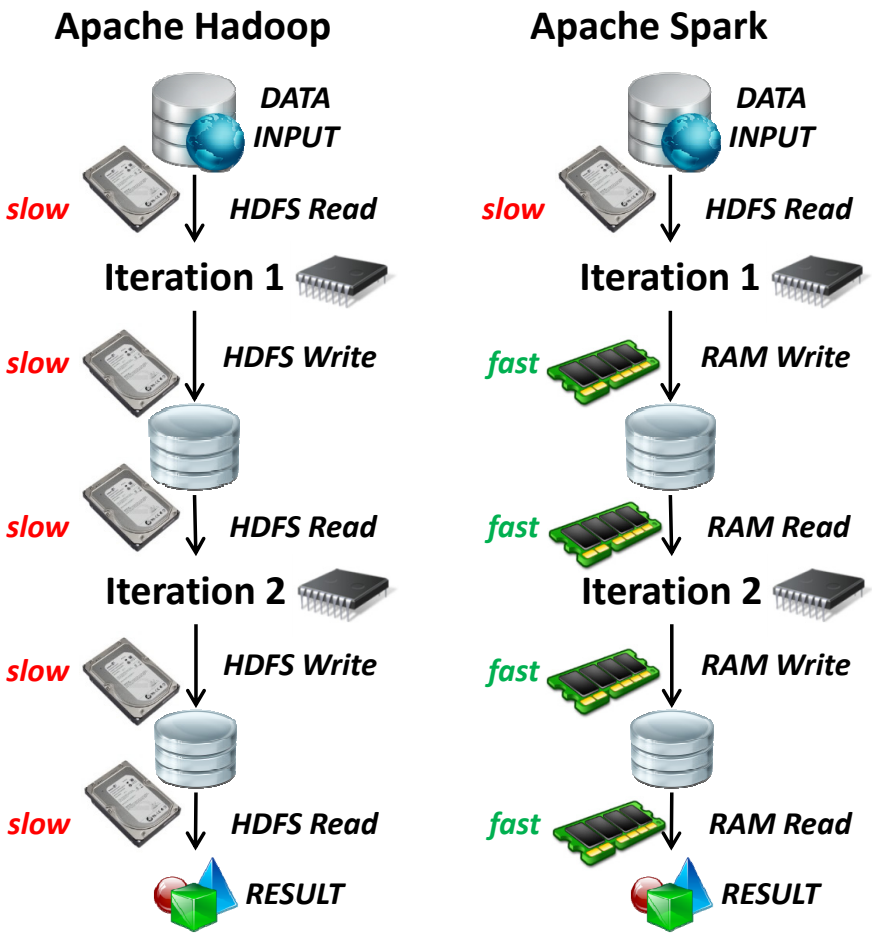
```
nums = sc.parallelize([1, 2, 3])

# Retrieve RDD contents as a local collection
nums.collect() # => [1, 2, 3]

# Return first K elements
nums.take(2)   # => [1, 2]

# Count number of elements
nums.count()   # => 3

# Merge elements with an associative function
nums.reduce(lambda x, y: x + y)  # => 6

# Write elements to a text file
nums.saveAsTextFile("hdfs://file.txt")
```

(collect returns elements of the dataset as an array back to the driver program, often used in debugging)

- In Apache Spark the Actions on RDDs enable some specific action to be taken on the dataset by the Spark runtime such as collect, take, count, reduce, or saveAsTextFile

[5] Parallel Programming with Spark

Apache Spark vs. Apache Hadoop

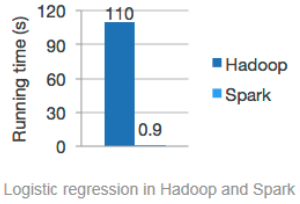


- Apache Spark Results
  - Up to 100% faster then traditional Hadoop (for some applications)
  - Requires 'big memory' systems to perform well

- **One key difference between Apache Spark vs. Apache Hadoop is that slow HDFS reads and writes are exchanged with fast RAM reads and writes in several iterations that might be part of a larger workflow**
- **A workflow is a sequence of processing tasks on data that leads to results via 'iterations'**

[6] big-data.tips, Apache Spark vs. Hadoop

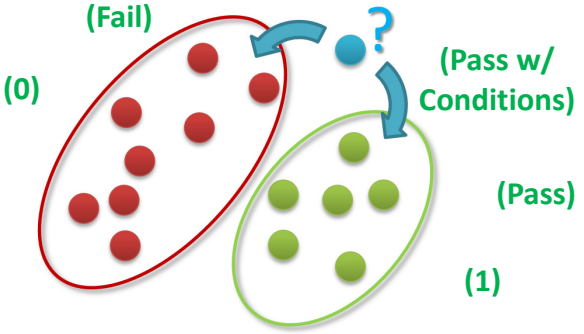
(e.g., application of food inspection analysis, cf. Lecture 2)



Logistic regression in Hadoop and Spark

Microsoft Azure

HDInsight



Amazon Web Services – High Memory Computing Resources & Cost Review

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
m6gd.medium	1	N/A	4 GiB	1 x 59 NVMe SSD	$0.0452 per Hour
m6gd.large	2	N/A	8 GiB	1 x 118 NVMe SSD	$0.0904 per Hour
m6gd.xlarge	4	N/A	16 GiB	1 x 237 NVMe SSD	$0.1808 per Hour
m6gd.2xlarge	8	N/A	32 GiB	1 x 475 NVMe SSD	$0.3616 per Hour
m6gd.4xlarge	16	N/A	64 GiB	1 x 950 NVMe SSD	$0.7232 per Hour
m6gd.8xlarge	32	N/A	128 GiB	1 x 1900 NVMe SSD	$1.4464 per Hour
m6gd.12xlarge	48	N/A	192 GiB	2 x 1425 NVMe SSD	$2.1696 per Hour
m6gd.16xlarge	64	N/A	256 GiB	2 x 1900 NVMe SSD	$2.8928 per Hour

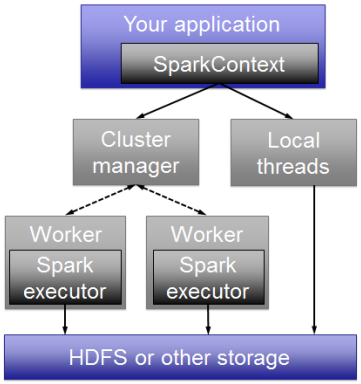
Memory Optimized - Current Generation					
x1.16xlarge	64	174.5	976 GiB	1 x 1920 SSD	$6.669 per Hour
x1.32xlarge	128	349	1,952 GiB	2 x 1920 SSD	$13.338 per Hour
x1e.xlarge	4	12	122 GiB	1 x 120 SSD	$0.834 per Hour
x1e.2xlarge	8	23	244 GiB	1 x 240 SSD	$1.668 per Hour
x1e.4xlarge	16	47	488 GiB	1 x 480 SSD	$3.336 per Hour
x1e.8xlarge	32	91	976 GiB	1 x 960 SSD	$6.672 per Hour
x1e.16xlarge	64	179	1,952 GiB	1 x 1920 SSD	$13.344 per Hour
x1e.32xlarge	128	340	3,904 GiB	2 x 1920 SSD	$26.688 per Hour

General Purpose - Current Generation					
a1.medium	1	N/A	2 GiB	EBS Only	$0.0255 per Hour
a1.large	2	N/A	4 GiB	EBS Only	$0.051 per Hour
a1.xlarge	4	N/A	8 GiB	EBS Only	$0.102 per Hour
a1.2xlarge	8	N/A	16 GiB	EBS Only	$0.204 per Hour
a1.4xlarge	16	N/A	32 GiB	EBS Only	$0.408 per Hour
a1.metal	16	N/A	32 GiB	EBS Only	$0.408 per Hour

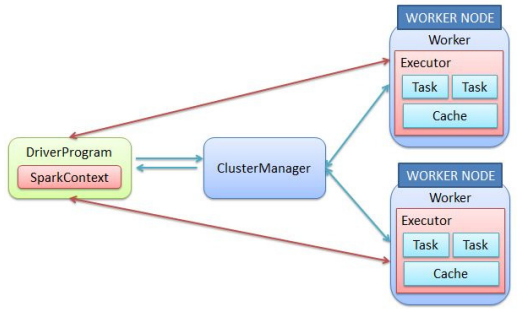
[7] AWS EC2 Pricing

Distributed Operations & Parallel Job Execution

- Different parallel options for tasks
  - SparkContext is head/master and used for starting applications
  - E.g. use local threads
  - E.g. use cluster via Apache (e.g. Hadoop YARN)
- Spark Task Scheduler
  - Supports general task graphs
  - Cache-aware data re-use & locality (i.e. multiple iterations over data benefit most, e.g. machine learning)
  - Partitioning-aware to avoid shuffles

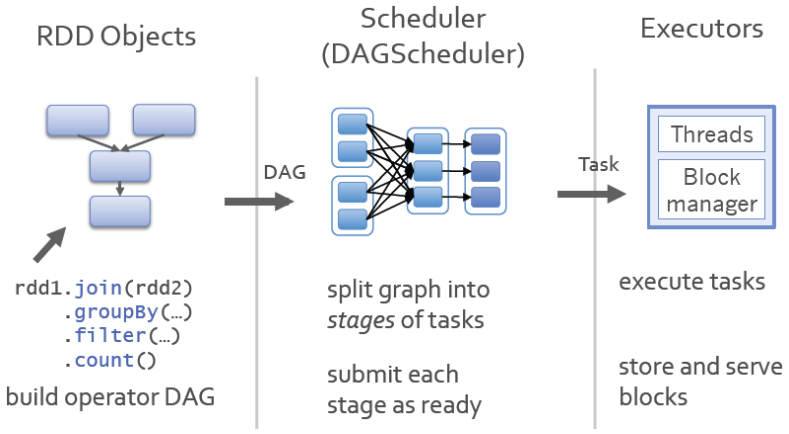


(also named as Spark Driver → master)



[24] Understanding Parallelization of Machine Learning Algorithms in Apache Spark

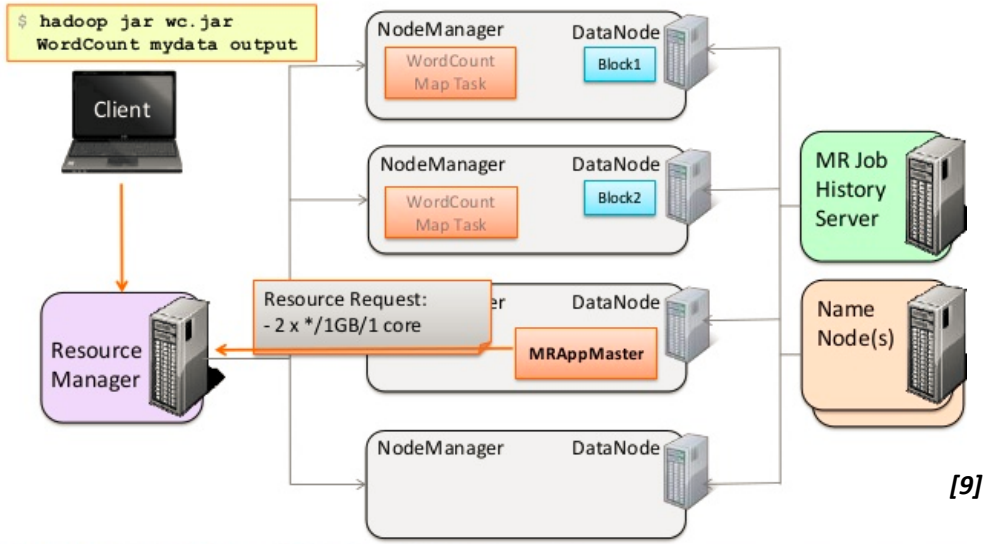
- **Transformations build RDDs from other RDDs (i.e. map, filter, groubBy, join, union, etc.)**
- **Actions return a real result or write it to storage (e.g. count, collect, save, etc.)**



[8] Advanced Spark ***[5] Parallel Programming with Spark***

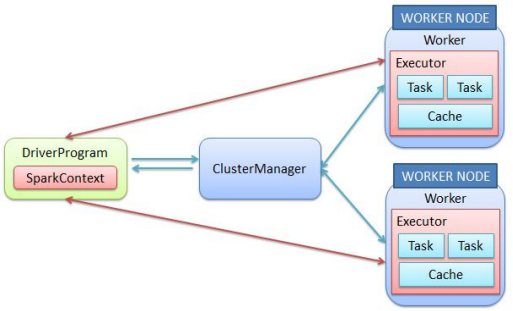
Hadoop Resource Manager Yarn & Wordcount Example

- Apache Hadoop offers the Yet Another Resource Negotiator (YARN) scheduler for map-reduce jobs
- Idea of Yarn is to split up the functionalities of resource management & job scheduling/monitoring
- The ResourceManager is a scheduler that controls resources among all applications in the system
- The NodeManager is the per-machine system that is reponsible for containers, monitoring of their resource uage (cpu, memory, disk, network) and reporting to the ResourceManager



[9] Introduction to Yarn

[3] Apache Hadoop Web page



[24] Understanding Parallelization of Machine Learning Algorithms in Apache Spark

➢ Lecture 5 provides more details on the Hadoop Distributed File System (HDFS) that is often used in conjunction with Apache Spark

Understand AWS Cloud Service Portfolio – Analytics Services

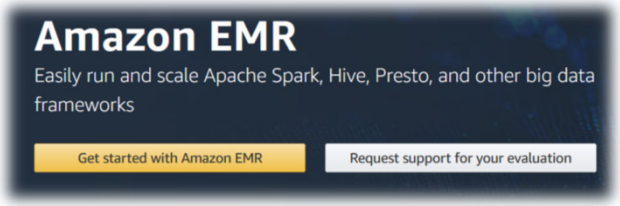
- Multiple analytics products
  - Extracting insights and actionable information from data requires technologies like analytics & machine learning
- Analytics Services
  - Amazon Athena: Serverless Query Service
  - Amazon ElasticMapReduce (EMR): Hadoop
  - Amazon ElasticSearch Service: Elasticsearch on AWS
  - Amazon Kinesis: Streaming Data
  - Amazon QuickSight: Business Analytics
  - Amazon Redshift: Data Warehouse

- **AWS offers a wide variety of Big Data Analytics Cloud services such as Amazon Athena (interactive analytics), Amazon EMR (big data processing with Apache Hadoop & Spark), Amazon Redshift (data warehousing), Amazon Kinesis (real-time analytics), Amazon Elasticsearch Service (operational analytics), and Amazon Quicksight (dashboards and visualizations)**

[10] AWS Web page

AWS Analytics services

Category	Use cases	AWS service
Analytics	Interactive analytics	Amazon Athena
	Big data processing	Amazon EMR
	Data warehousing	Amazon Redshift
	Real-time analytics	Amazon Kinesis
	Operational analytics	Amazon Elasticsearch Service
	Dashboards and visualizations	Amazon Quicksight



➢ **Lecture 5 provides more details on the Hadoop Distributed File System (HDFS) that is often used in conjunction with Apache Spark**

Understanding Chicago Restaurant Food Inspection Example (cf. Lecture 2)

```
In [1]: from pyspark.ml import Pipeline
        from pyspark.ml.classification import LogisticRegression
        from pyspark.ml.feature import HashingTF, Tokenizer
        from pyspark.sql import Row
        from pyspark.sql.functions import UserDefinedFunction, col
        from pyspark.sql.types import *
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1536508990334_0003	pyspark	idle	Link	Link	✔

```
SparkSession available as 'spark'.
```

(remember that even idle SparkSessions and Azure HDInsight Clusters cost per minute)

```
In [2]: inspections = spark.read.csv('wasb:///HdiSamples/HdiSamples/FoodInspectionData/Food_Inspections1.csv', inferSchema=True)
```

- **The execution of the import cell using PySpark kernel notebooks automatically generates in the background a SparkSession with a SparkContext in Azure HDInsight Clusters**
- **The startup and printout of the Spark application status might take a while when executing the cell (shift+enter)**

(after having started Jupyter in the cloud we have started our example application by reading data into a dataframe)

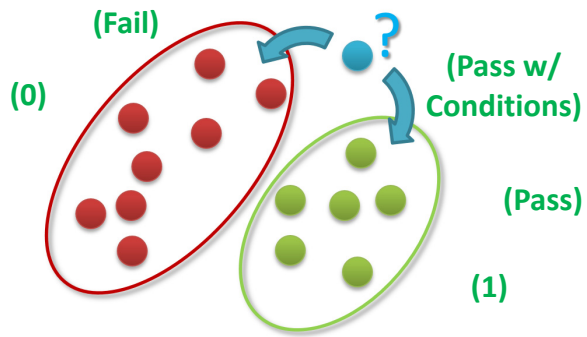
- **A Spark Session (since Version 2.0) is an entry point that subsumed the SQLContext and HiveContext - it is an entry point for programming Spark with the Dataset & DataFrame API**

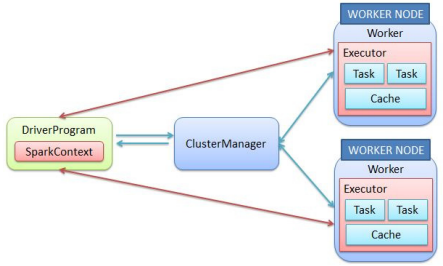
Microsoft Azure

HDInsight

hadoop

[3] Apache Hadoop Web page



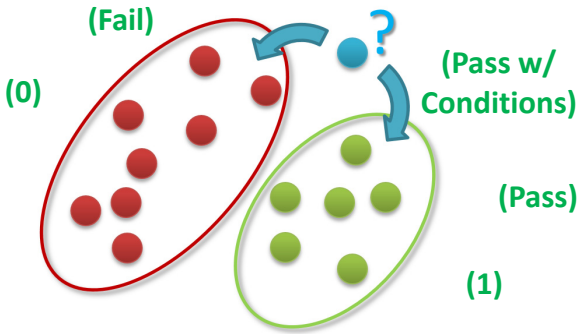


[24] Understanding Parallelization of Machine Learning Algorithms in Apache Spark

Apache Spark Dataset & DataFrame API

- Example: Load Dataset
  - Read/load dataset from csv file
  - Generated a Spark dataframe for further use

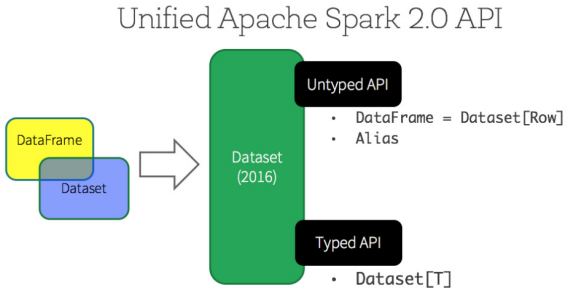
HDInsight



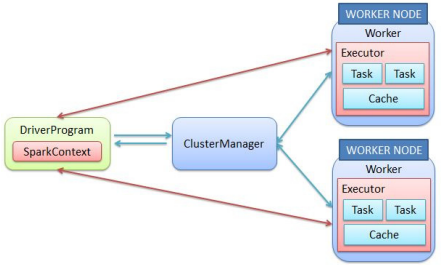

```
In [2]: inspections = spark.read.csv('wasb:///HdiSamples/HdiSamples/FoodInspectionData/Food_Inspections1.csv', inferSchema=True)

In [3]: inspections.printSchema()

root
 |-- _c0: integer (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: integer (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: integer (nullable = true)
 |-- _c10: string (nullable = true)
 |-- _c11: string (nullable = true)
 |-- _c12: string (nullable = true)
 |-- _c13: string (nullable = true)
 |-- _c14: double (nullable = true)
 |-- _c15: double (nullable = true)
 |-- _c16: string (nullable = true)
```

(DataFrame is an alias for a collection of generic objects Dataset[Row], where a Row is a generic untyped Java Virtual Machine object)

(Dataset is a collection of strongly-typed Java Virtual Machine objects, dictated by a case class you define in as a class in Java)



[11] Databricks

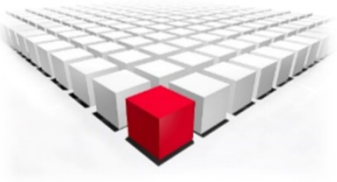


[24] Understanding Parallelization of Machine Learning Algorithms in Apache Spark

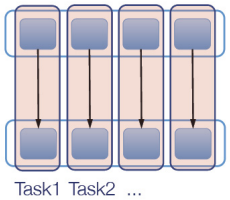
- **A Spark DataFrame is (like an RDD) also an immutable distributed collection of data**
- **The key difference between DataFrame & RDDs is that data is organized into named columns that is closer to the traditional handling of data with relational databases & clear structure/schema of it**

Understanding Apache Spark Benefits – WordCount Application Example

- Example: Error message log analysis (e.g. wordcount)
  - From a log into memory, then interactively search for patterns
  - Benefit: scaled to 1 TB data in 5-7 sec (vs 170 sec for on-disk data)
  - Reasoning: Take advantage of configured Cache at workers ('in-memory')



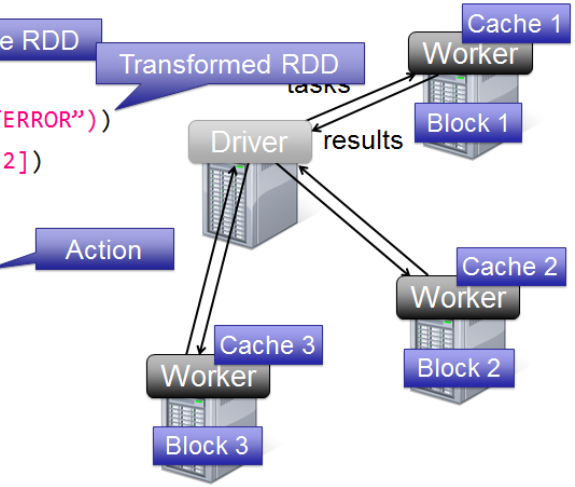
(inherent parallel)



[8] Advanced Spark

(spark is a created SparkContext object)

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split('\t')[2])
messages.cache()

messages.filter(lambda s: "foo" in s).count()
messages.filter(lambda s: "bar" in s).count()
. . .
```




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

[5] Parallel Programming with Spark

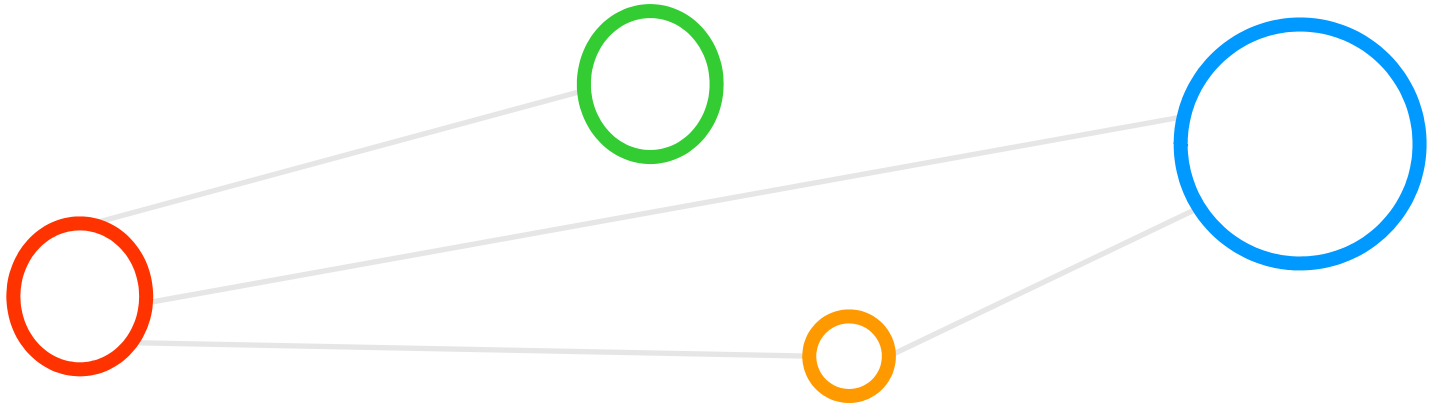
(.flatmap: applies given function to each element of this iterable – THEN concatenates the results, e.g. [1,2], [], [4] = [1,2,4])

(.map: returns iterable resulting from applying the given function to each element of this iterable: e.g. word,1)

[Video] Apache Spark Summary

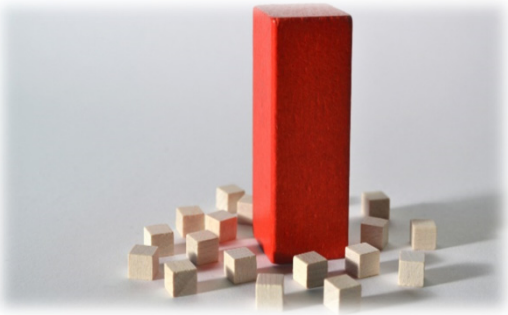
[12] YouTube video, Solving Big Data with Apache Spark

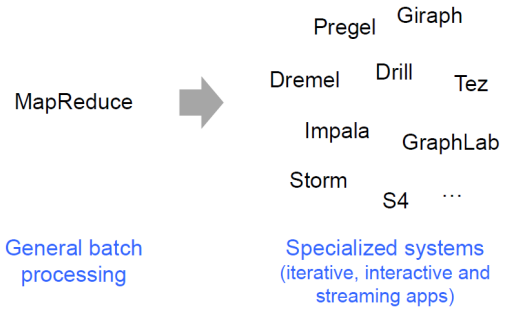
Apache Spark Libraries for Advanced Applications

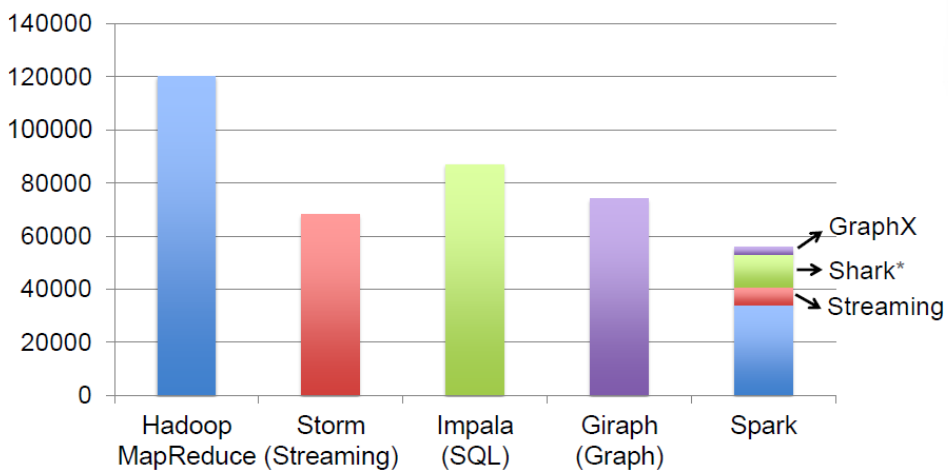


Apache Spark as Base Platform for Big Data Challenges

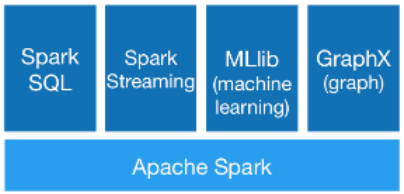
- Generalized platform for a wide variety of applications
  - Extension towards Directed Acyclic Graphcs (DAGs)
  - Improvements on ‘data sharing capabilities‘ & ‘in-memory computing‘







[13] Berkeley Data Analytics Stack



[4] Apache Spark

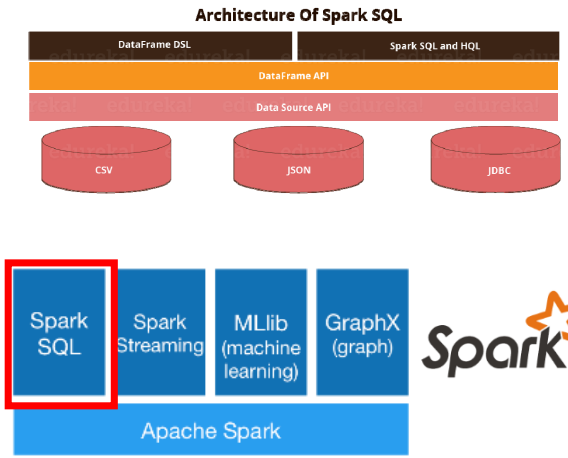
- **Apache Spark can be considered as a base platform for advanced big data analytics such as provided by four distinct higher level libraries: (a) Spark SQL; (b) Spark Streaming; (c) MLlib; and (d) GraphX library**

Apache Spark SQL Library

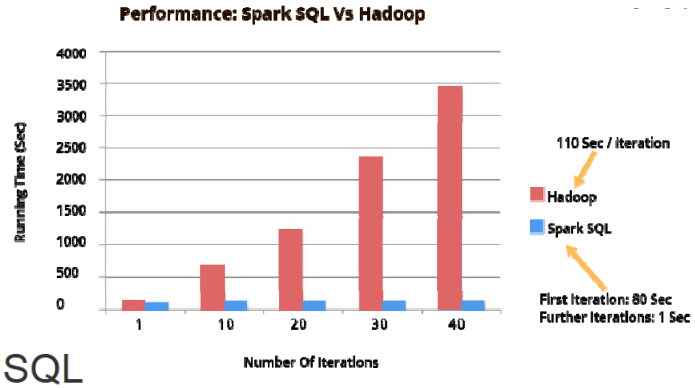
- Usage example
  - Connect ‘Apache Spark context‘ to existing Apache Hive Database
  - Apply functions to results of SQL queries
  - Relational Databases

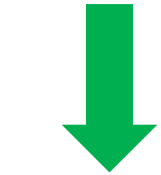
```
context = HiveContext(sc)
results = context.sql(
  "SELECT * FROM people")
names = results.map(lambda p: p.name)
```

[14] Apache Hive



[4] Apache Spark





(i.e. HiveQL, SQL-like language)

- **Apache Spark Structured Query Language (SQL) library works with structured data and enables queries for data inside Spark programs using SQL (e.g. based on Apache Hive/HiveQL)**

Chicago Restaurant Food Inspection Example (cf. Lecture 2)

In [1]:
```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer
from pyspark.sql import Row
from pyspark.sql.functions import UserDefinedFunction, col
from pyspark.sql.types import *
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1536508990334_0003	pyspark	idle	Link	Link	✔

```
SparkSession available as 'spark'.
```

In [8]:
```
df.registerTempTable('CountResults')
```

In [9]:
```
df.show(5)
```

```
+------+--------------------+-------+--------------------+
|    id|                name|results|          violations|
+------+--------------------+-------+--------------------+
|413707|       LUNA PARK INC|   Fail|24. DISH WASHING ...|
|391234|       CAFE SELMARIE|   Fail|2. FACILITIES TO ...|
|413751|          MANCHU WOK|   Pass|33. FOOD AND NON-...|
|413708|BENCHMARK HOSPITA...|   Pass|                    |
|413722|           JJ BURGER|   Pass|                    |
+------+--------------------+-------+--------------------+
only showing top 5 rows
```

- **pyspark is a Python Application Programming Interface (API) for Spark available in Microsoft Azure HDInsight Clusters**
- **Import required libraries for our logistic regression application via selected libraries from pyspark and the Spark Machine Learning Library (MLlib)**
- **Import selected libraries for better analyzing data & creating features using the Apache Spark Structured Query Language (SQL) library**

[14] Apache Hive

- **Apache Hive offers an optimized column-style table for 'big data queries' by storing columns separately from each other to reduce read, decompression & processing of data not even queried**

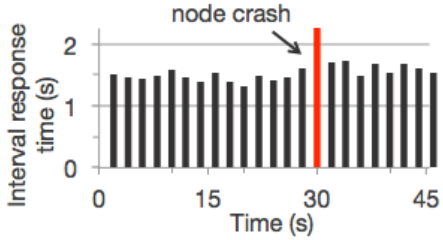
Apache Spark Streaming Library

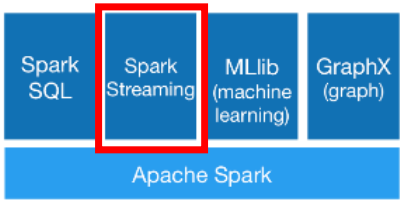
- Usage example
  - Combine streaming with batch and interactive queries or using 'windows'
  - E.g. find words with higher frequency than historic data
  - Recovers lost work without any extra code for users to write (e.g. cf. RDD fault tolerance features)

```
TwitterUtils.createStream(...)
    .filter(_.getText.contains("Spark"))
    .countByWindow(Seconds(5))
```

Counting tweets on a sliding window

```
stream.join(historicCounts).filter {
  case (word, (curCount, oldCount)) =>
    curCount > oldCount
}
```





Spark Streaming

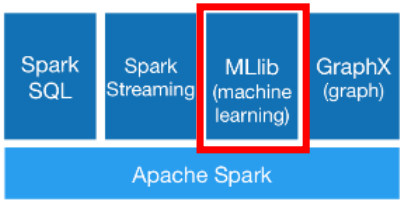
[4] Apache Spark

- **Apache Spark Streaming library enables to write streaming jobs the same way users would write batch jobs or to combine streaming with batch and interactive queries or filters**

➢ **Lecture 15 provides more details on big data streaming tools like Apache Spark Streaming and other more advanced analytics toolkits**

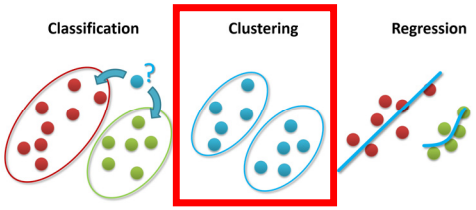
Apache Spark Machine Learning Library (MLlib) & K-Means Clustering Example

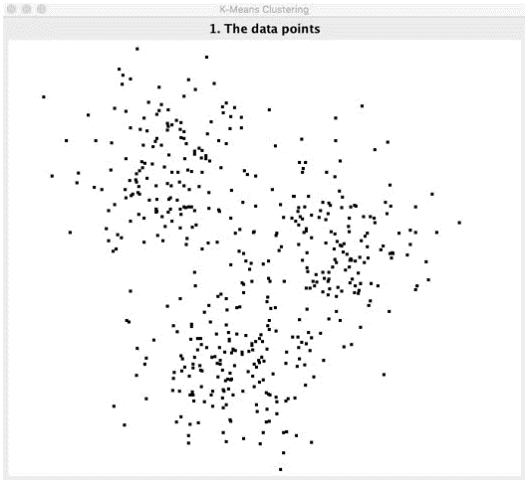
- Usage example
  - MLlib contains high-quality algorithms that leverage iteration during the training process (cf. Lecture 2) as key benefit of Apache Spark
  - E.g. Clustering of data via K-Means Algorithm
  - E.g. Classification of data via LogisticRegression
  - Parallel algorithms for clustering, classification, and regression
  - Offers an implementation for the fast realization of scalable recommendation engines



[4] Apache Spark

Spark MLlib





(example of k-means algorithm)

```
points = spark.textFile("hdfs://...")
                .map(parsePoint)

model = KMeans.train(points, k=10)
```

Calling MLlib in Python

[22] YouTube video, Visualization of K-Means Clustering

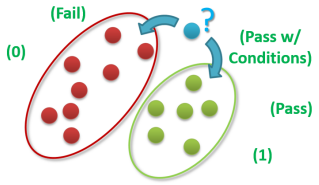
- **Apache Spark Machine Learning library (MLlib) is a scalable & parallel machine learning library with a number of algorithms for classification, clustering, regression, and building recommender systems fast**

➢ **Practical Lecture 3.1 offers a more practical introduction in a variety of machine learning algorithms using Apache Spark Mllib in Clouds**

Apache Spark Machine Learning Library (MLlib) & Logistic Regression Example

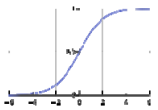
- Classification as optimization problem
  - E.g., food inspection in restaurants (cf. Lecture 1 & Assignment #1)
  - Using Logistic Regression in Spark
  - Solves inherent optimization problem via optimizer/solver
  - E.g., Stochastic Gradient Descent (SGD) instead of Perceptron Learning Algorithm

[24] Understanding Parallelization of Machine Learning Algorithms in Apache Spark



$$h(\mathbf{x}) = sign\left(\left(\sum_{i=0}^{d} w_i x_i\right)\right); x_0 = 1$$

$$h(\mathbf{x}) = \sigma\ (\mathbf{w} \cdot \mathbf{x})$$



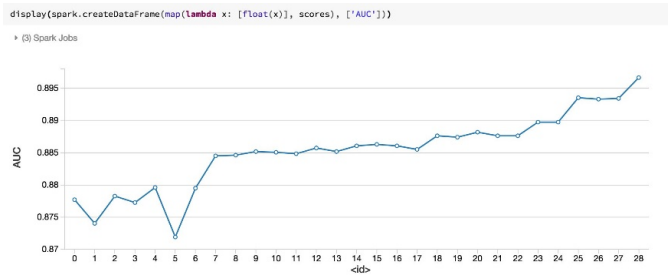
$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

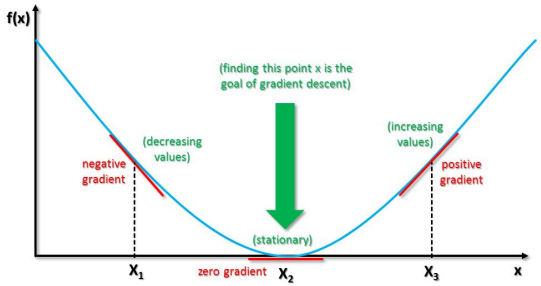
(logistic function)

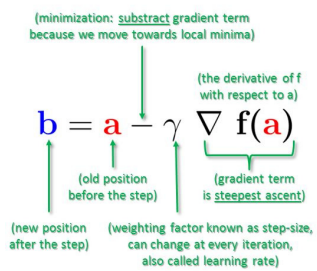
```
1 / (1 + exp(-w^T x))
```

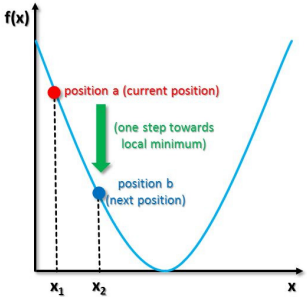
```
log(1 + exp(-yw^T x))
```

(loss function, logistic logic)







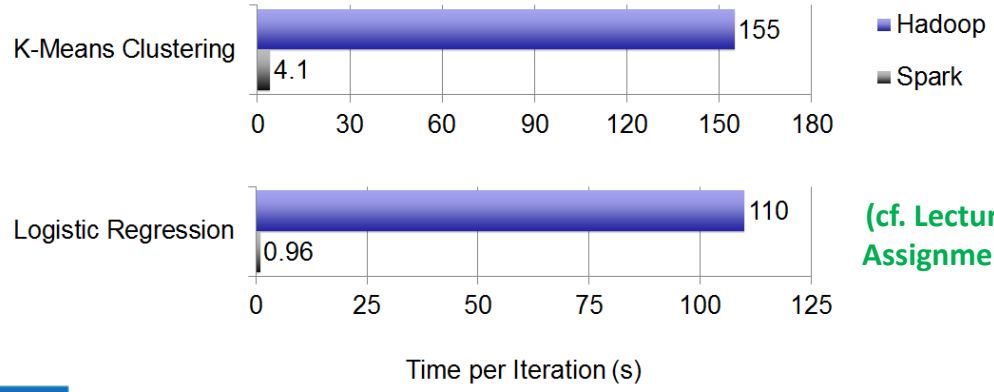


- **Training machine learning is an optimization problem used with many optimizers like stochastic gradient descent (SGD)**
- **The optimization has the goal to learn the coefficients (i.e. weights) based on the examples and fit a line based on the points - Getting towards the optimal model is called convergence**
- **The goal of the optimization is to either maximize a positive metric like the area under the curve (AUC) or minimize a negative metric (e.g., error on testset using a specific error function)**

➢ **Practical Lecture 3.1 offers a practical introduction into optimization and its relationship to techniques like stochastic gradient descent**

Apache Spark Machine Learning Library (MLlib) – Performance Comparisons

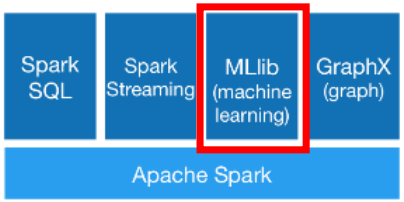
- Selected Algorithms use ‘iterations to learn‘ from datasets
  - E.g. Logistic Regression: Explore unknown trends in datasets
  - E.g. K-Means Clustering: Explore unknown groups in datasets



(cf. Lecture 2 & Assignment #1)

```
points = spark.textFile("hdfs://...")
                .map(parsePoint)

model = KMeans.train(points, k=10)
```

Calling MLlib in Python



[4] Apache Spark

Spark MLlib

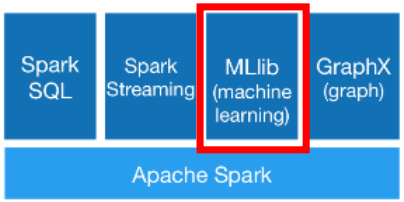
[5] Parallel Programming with Spark

- **Apache Spark Machine Learning library (MLlib) offers machine learning algorithms that are optimized for in-memory computing and are thus much faster than the traditional underlying Apache Hadoop approach**

➢ **Practical Lecture 3.1 offers a practical introduction into optimization and its relationship to techniques like stochastic gradient descent**

Apache Spark Machine Learning Library (MLlib) – Selected Algorithms

- Classification algorithms
  - include logistic regression and naïve Bayes
- Regression
  - Generalized linear regression and survival regression
- Tree-based approaches
  - Standard decision trees, random forests, gradient-boosted trees
- Alternating least squares (ALS)
  - Used to create recommendation engines



Spark MLlib

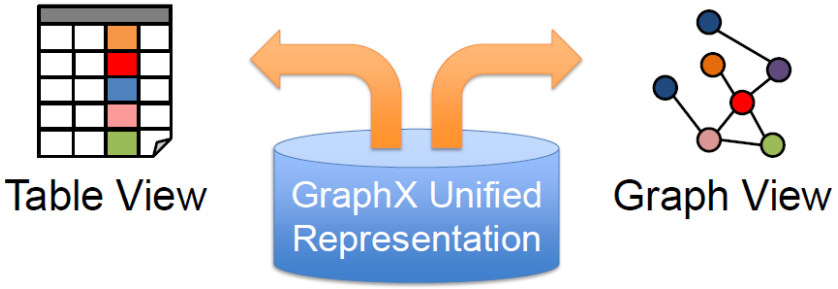
[4] Apache Spark

- Clustering
  - K-Means and Gaussian Mixture Models (GMM)
- Pattern mining
  - Frequent itemsets & Association rule mining

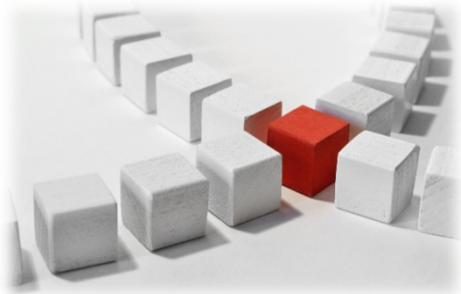
➢ **Practical Lecture 3.1 offers a more practical introduction in a variety of machine learning algorithms using Apache Spark Mllib in Clouds**

Apache Spark GraphX Library

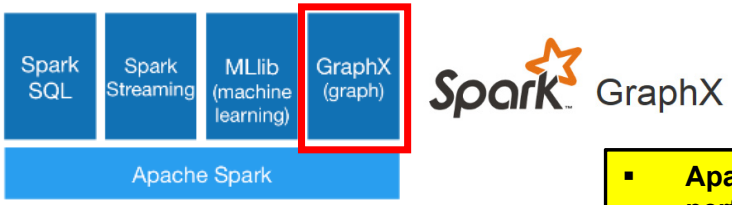
- Usage example
  - Tables and Graphs are composable ‘views‘ of the same physical data
  - Each ‘view‘ has its own Operators that exploit the semantics of the view to achieve efficient execution



[13] Berkeley Data Analytics Stack



(e.g. ranking of Web pages can be represented as graph whereby rank computations depends only on neighbours that link to the corresponding page, graph parallel pattern)



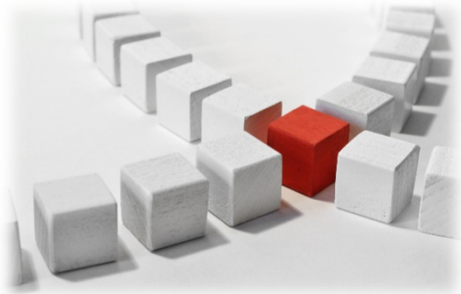
[4] Apache Spark

- **Apache Spark GraphX enables iterative graph computations within a single system with a comparable performance to the fastest specialized graph processing systems**

➢ **Lecture 14 provides more details on the major importance of using graph databases for certain big data application challenges & tools**

Web Application Example: PageRank Technique

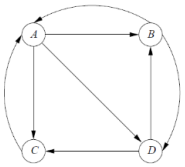
- Change of Live: Availability of efficient & accurate Web search
  - Many search engines, but search engine 'Google as pioneer'
  - Innovation by Google was a nontrivial technological advance: 'PageRank'
  - (term comes from Larry Page, inventor of it and Google founder)



- Goal: approach against so-called 'term spam'
  - Provide techniques against approaches for fooling search engines into believing your page is about something it is not (e.g. terms in background)

- **PageRank is a tool for evaluating the importance of Web pages in a way that it is not easy to fool**
- **PageRank technique can leverage graph databases**

- Basic Algorithm Idea:
  - Analyse links and assign weights to each vertex in a 'Web graph' by iteratively aggregating weights of inbounds



modified from [15] Mining of Massive Datasets

Understanding the Page Rank Technique

- Big data problem ‘whole Web of links‘
  - Memory challenge for whole Web: In order to compute the PageRank at a given iteration we also need the PageRank for the iteration before
  - E.g. two extremely large arrays required: $PR_{(t)}$, and $PR_{(t+1)}$
  - It is important in implementations to not mess these two time steps up
- Simplified formula
  - C is factor of normalization
- Better formula
  - Solves problem ‘rank sink‘, two pages just link themselves and one incoming link (‘accumulate‘)
  - Add ‘random picks‘ (aka rank source)

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

(N_v is number of links from v)

(B_u is set of pages that point to u)

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u)$$

(add decay factor, aka ‘random picks‘ called a rank source)

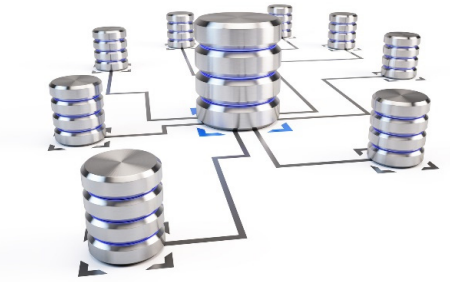
- **Page Rank is driven by probability of landing on one page or another (aka ‘being at a node‘)**
- **The probability can be interpreted as the higher the chance the more important the page is**

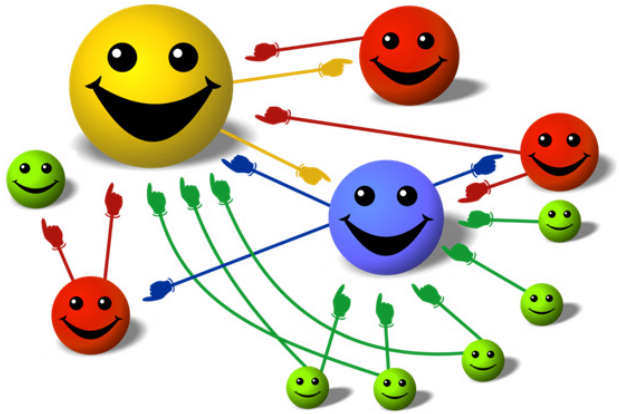
[16] The PageRank Citation Ranking

[17] Big-data.tips, ‘Page Rank Technique‘

Web Application Example: PageRank (1)

- Goal: Give Web pages a rank (score)
  - Reasoning: score based on two different 'types of links' to them
  - Links from many pages brings a high rank
  - Link from a high-rank page gives a high rank
- Easy problem – Complex algorithm
  - Multiple stages of map & reduce
  - Benefits from in-memory caching
  - Multiple iterations over the same data
  - Known also to be used initially for plain map-reduce

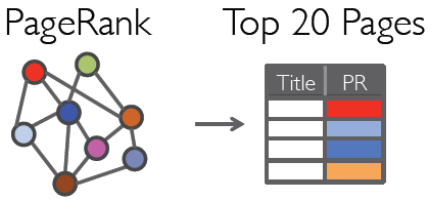


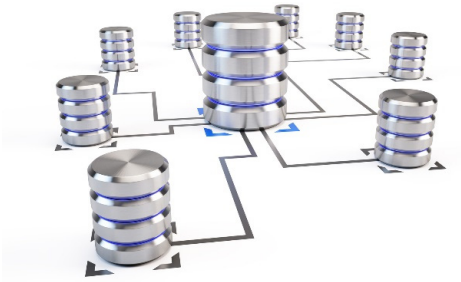


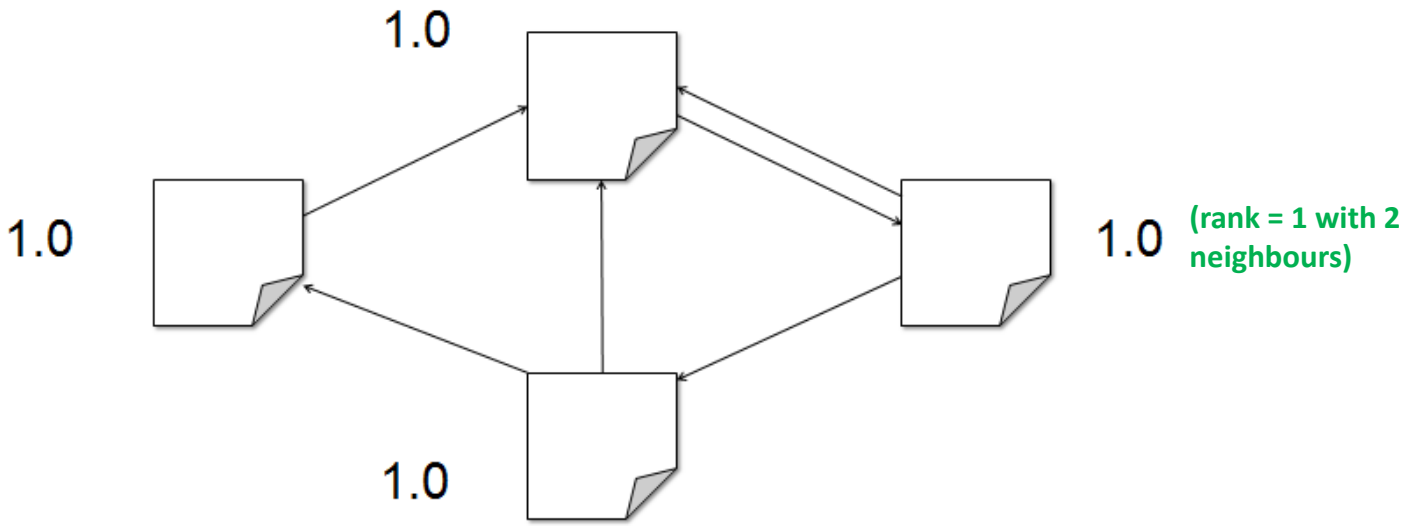
[5] Parallel Programming with Spark

Web Application Example: PageRank (2)

- Simplified basic algorithm
  - Start each page at a rank of 1
  - On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors **(aka 'link out')**
  - Set each page's rank to 0.15 + 0.85 × contribs





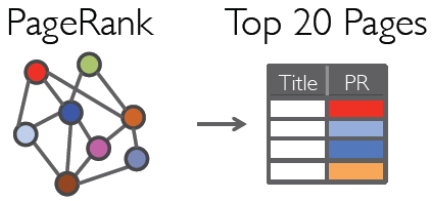


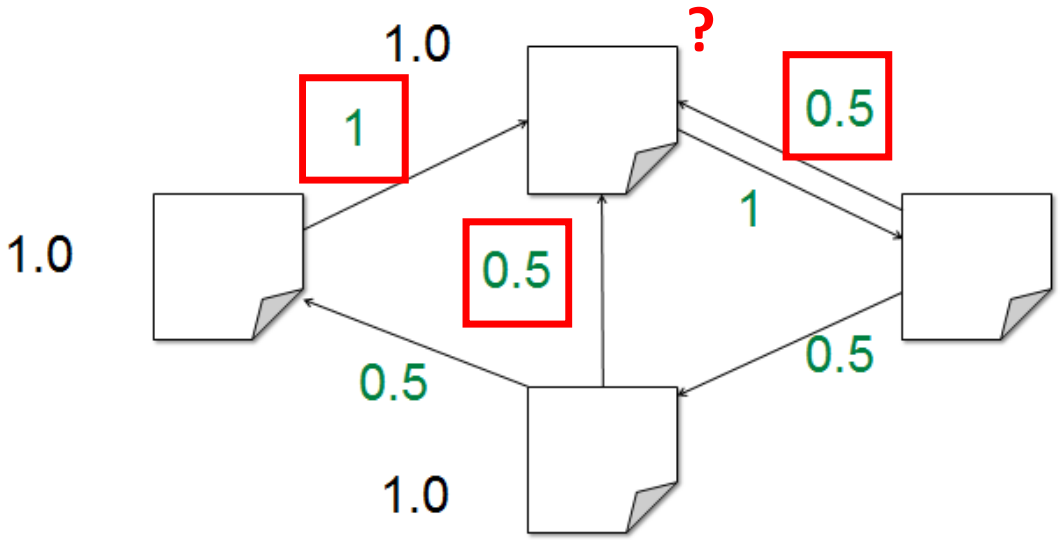
(rank = 1 with 2 neighbours)

[5] Parallel Programming with Spark

Web Application Example: PageRank (3)

- Simplified basic algorithm
  - Start each page at a rank of 1
  - On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors **(aka 'link out')**
  - Set each page's rank to 0.15 + 0.85 × contribs



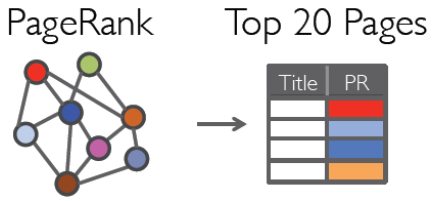


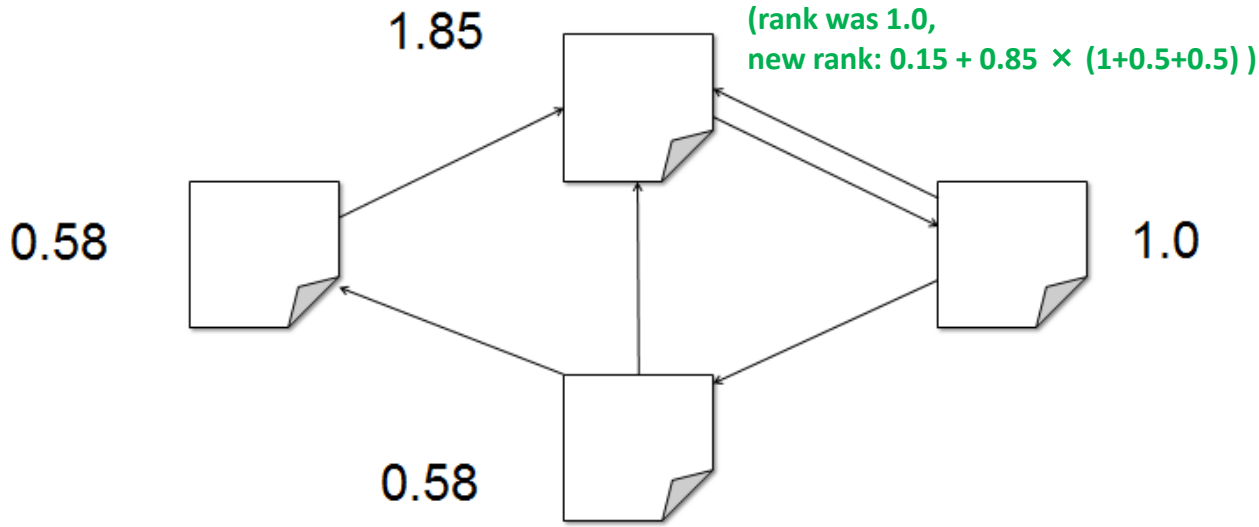
(rank = 1 with 2 neighbours → contribute ½ = 0.5 each via 'link out')

[5] Parallel Programming with Spark

Web Application Example: PageRank (4)

- Simplified basic algorithm
  - Start each page at a rank of 1
  - On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors **(aka 'link out')**
  - Set each page's rank to 0.15 + 0.85 × contribs

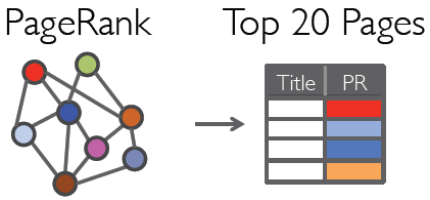


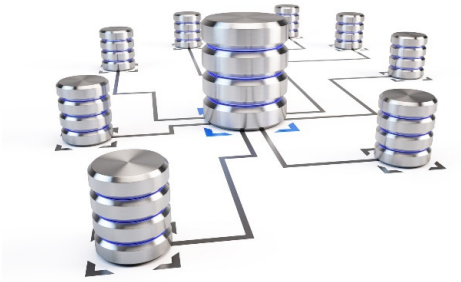


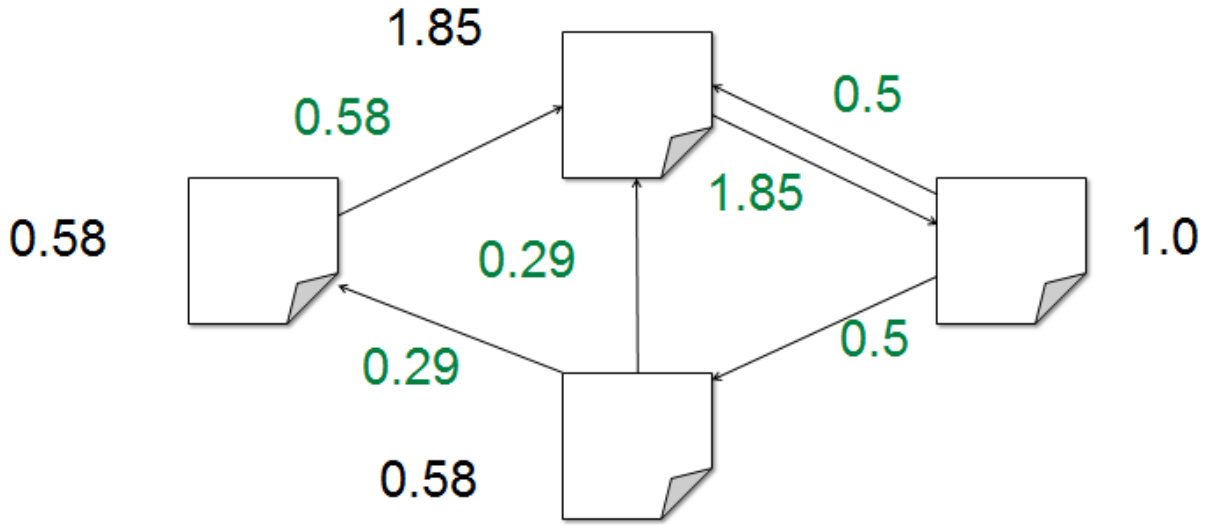
[5] Parallel Programming with Spark

Web Application Example: PageRank (5)

- Simplified basic algorithm
  - Start each page at a rank of 1
  - On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors **(aka 'link out')**
  - Set each page's rank to 0.15 + 0.85 × contribs



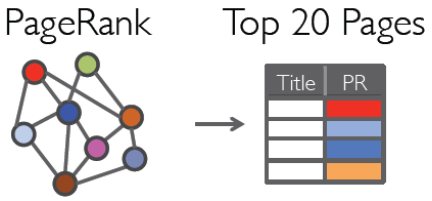


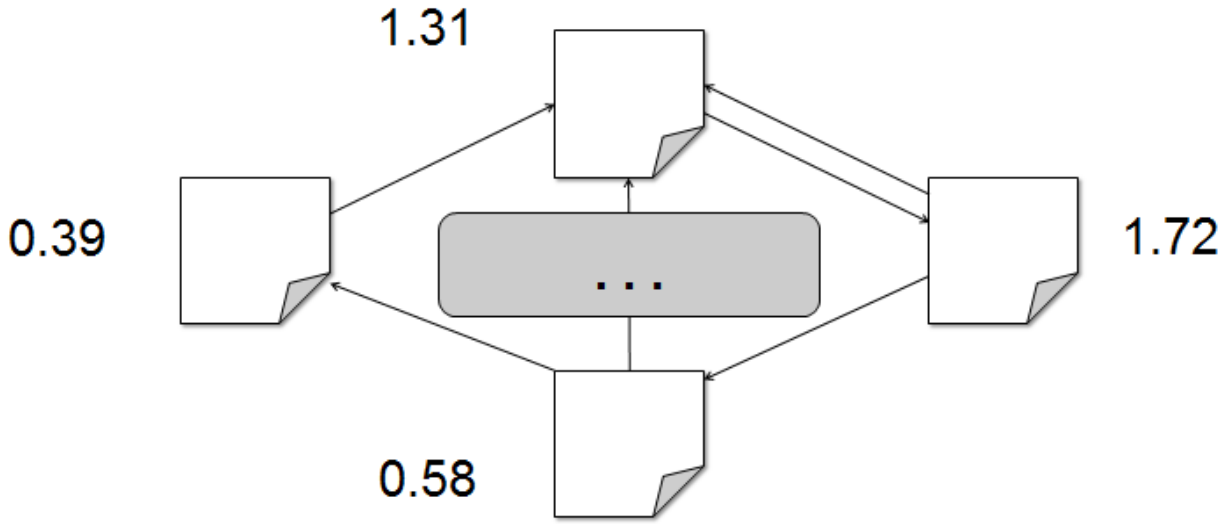


[5] Parallel Programming with Spark

Web Application Example: PageRank (6)

- Simplified basic algorithm
  - Start each page at a rank of 1
  - On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors **(aka 'link out')**
  - Set each page's rank to 0.15 + 0.85 × contribs

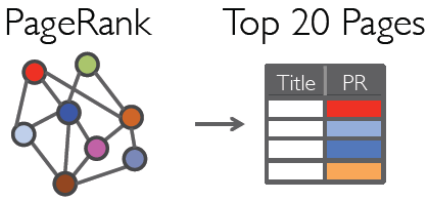




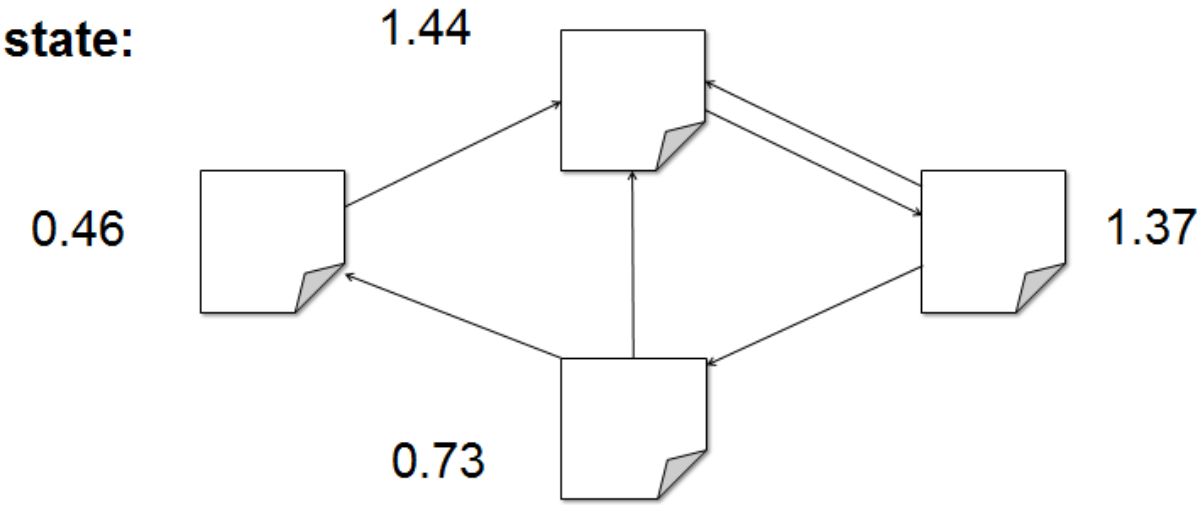
[5] Parallel Programming with Spark

Web Application Example: PageRank (7)

- Simplified basic algorithm
  - Start each page at a rank of 1
  - On each iteration, have page p contribute $rank_p$ / $|neighbors_p|$ to its neighbors **(aka 'link out')**
  - Set each page's rank to 0.15 + 0.85 × contribs

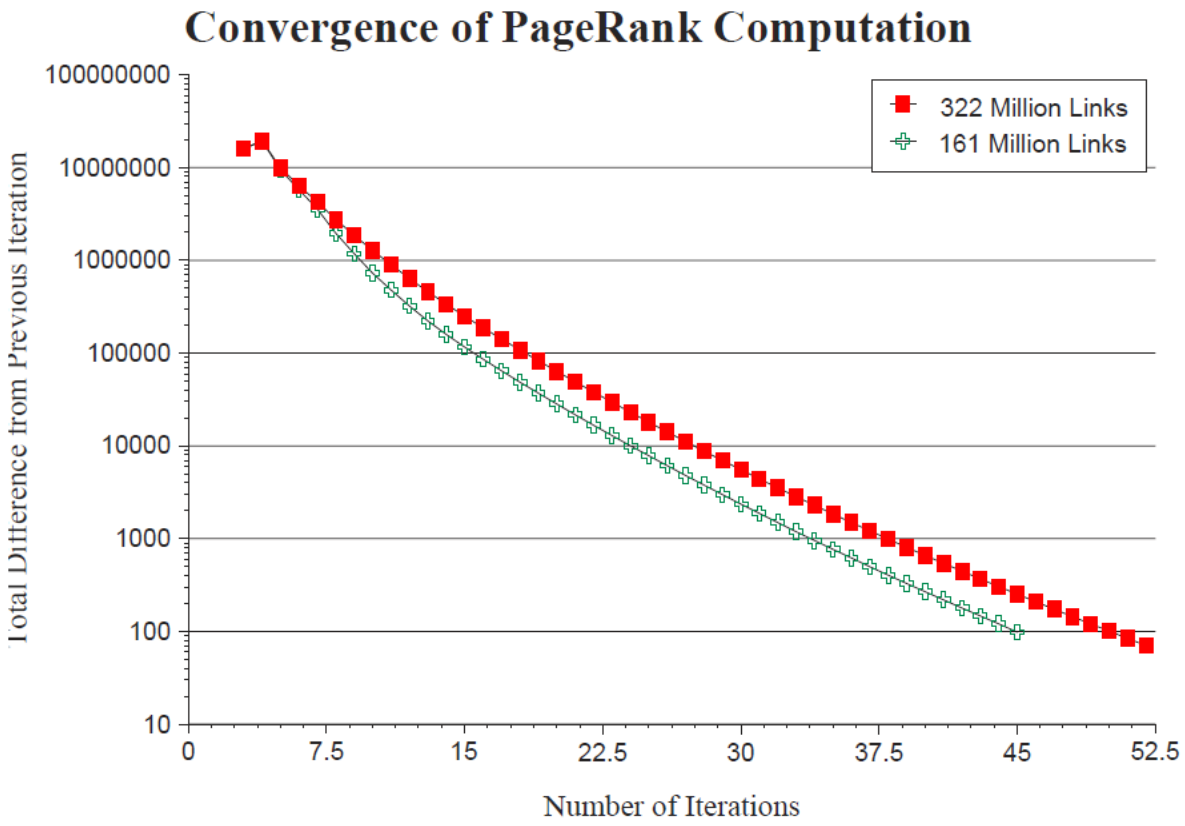


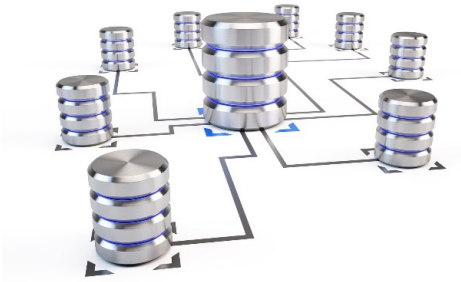
Final state:



[5] Parallel Programming with Spark

Web Application Example: PageRank (8)

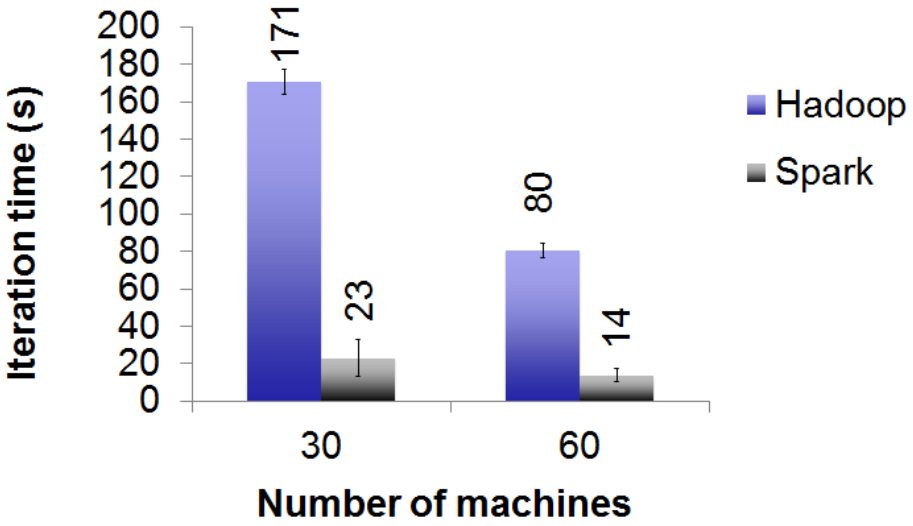




[16] The PageRank Citation Ranking

Web Application Example: PageRank (9)

- Performance Comparisons
  - Using Apache Spark (i.e. Cache, etc.) vs. plain Apache Hadoop



[5] Parallel Programming with Spark

Examples of Apache Spark in Business Applications

- Spark application use cases at Yahoo
  - Personalizing news pages for Web visitors, ML algorithms running on Spark to figure out what individual users are interested in
  - Categorize news stories as they arise to figure out what types of users would be interested in reading them
  - Running analytics for advertising, Hive (previous lectures) on Spark (Shark's) interactive capability, use existing BI tools to view and query their advertising analytic data collected in Hadoop
- Spark application use cases at Conviva
  - One of the largest streaming video companies on the Internet, uses Spark Streaming to learn network conditions in real time
- Spark application use cases at ClearStory
  - Relies on the Spark technology as one of the core underpinnings of its interactive & real-time product

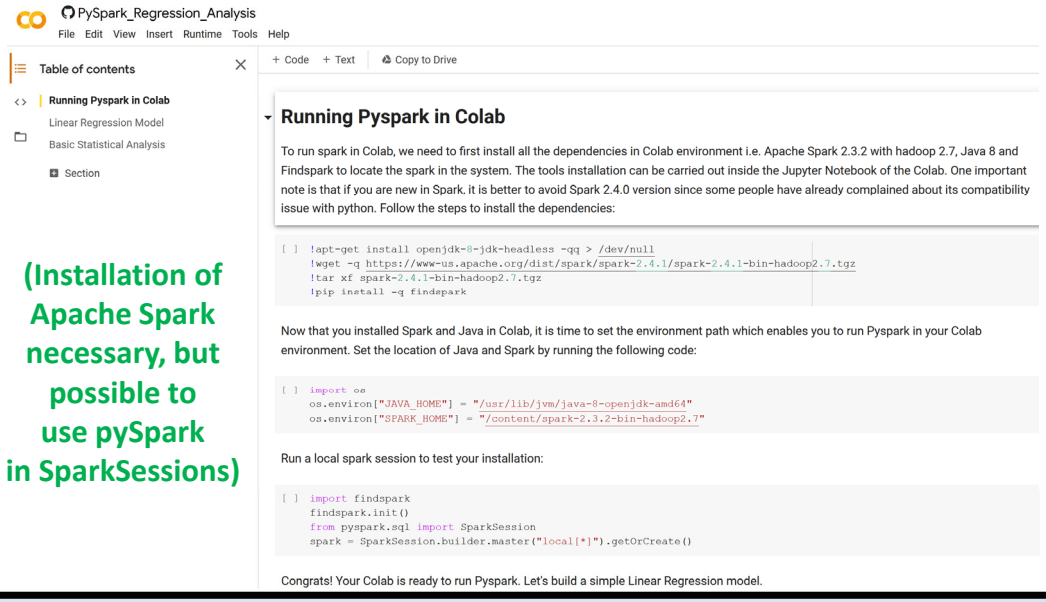
[18] Yahoo

[19] Conviva

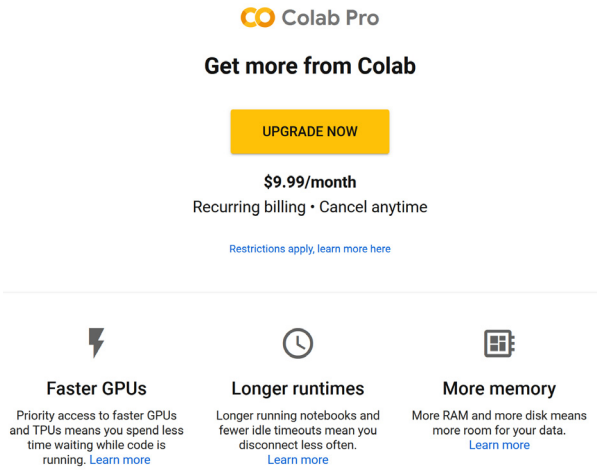
[20] ClearStory

Using Apache Spark on Google Colaboratory Cloud Infrastructure

- Google Colaboratory (free & pro version for 9.99 $ / month)
  - ‘Colab’ notebooks are Jupyter notebooks that run in the Google cloud
  - Possible to run Apache Spark via PySpark Jupyter notebooks in Colab
  - Highly integrated with other Google services (e.g., Google Drive for data)



(Installation of Apache Spark necessary, but possible to use pySpark in SparkSessions)



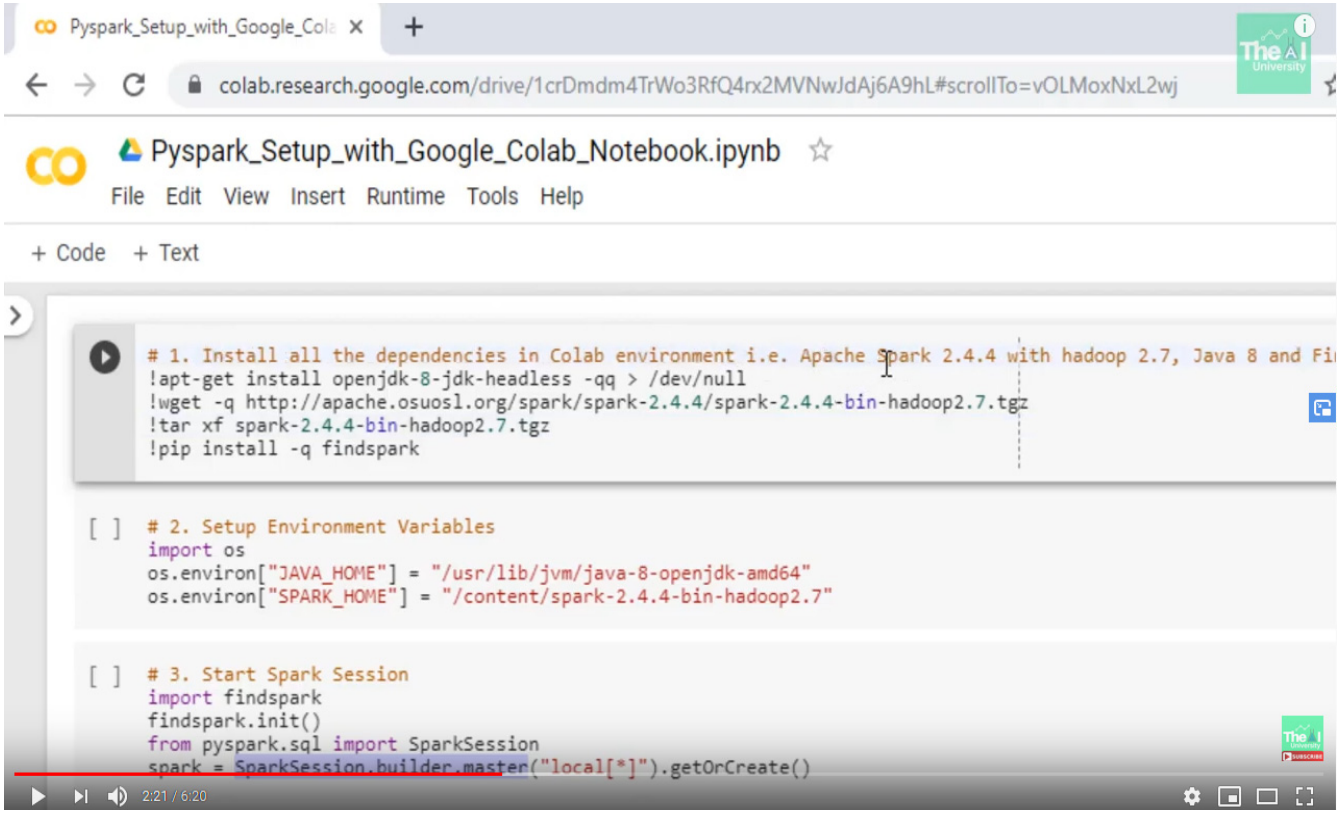
[23] Google Colaboratory

(for international students: watch out – it uses the browser language automatically)

- **Google Colaboratory offers ‘Colab‘ notebooks that are implemented with Jupyter notebooks that in turn run in the Google cloud and are highly integrated with other Google cloud services such as Google Drive thus making ‘Colab‘ notebooks easy to set up, access, and share with others**

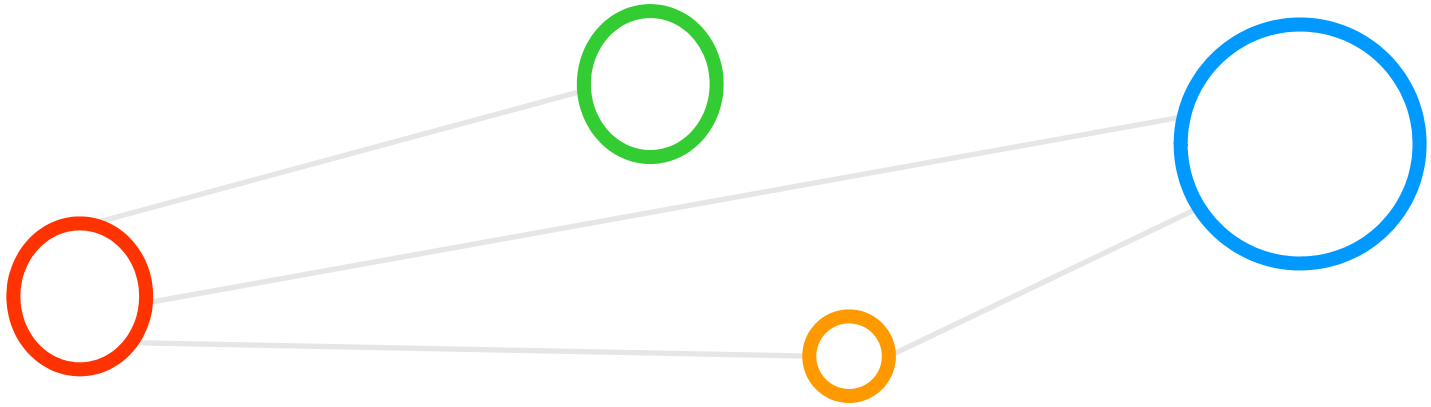
➢ **Lecture 6 & 7 offer insights of how to use deep learning with cutting-edge GPUs via Google ‘colab’ notebooks within the Google Cloud**

[Video] Building Machine Learning Pipelines with PySpark in Google ‘Colab’



[21] YouTube video, Solving Big Data with Apache Spark

Lecture Bibliography



Lecture Bibliography (1)

- [1] Species Iris Group of North America Database, Online: http://www.signa.org
- [2] Microsoft Azure HDInsight Service, Online: https://azure.microsoft.com/en-us/services/hdinsight/
- [3] Apache Hadoop Web page, Online: http://hadoop.apache.org/
- [4] Apache Spark Web page, Online: http://spark.apache.org/
- [5] M. Zaharia, 'Parallel Programming with Spark', 2013, Online: http://ampcamp.berkeley.edu/wp-content/uploads/2013/02/Parallel-Programming-With-Spark-Matei-Zaharia-Strata-2013.pdf
- [6] Big Data Tips, 'Apache Spark vs Hadoop', Online: http://www.big-data.tips/apache-spark-vs-hadoop
- [7] Amazon Web Services EC2 On-Demand Pricing models, Online: https://aws.amazon.com/ec2/pricing/on-demand/
- [8] Reynold Xin, 'Advanced Spark', 2014, Spark Summit Training, Online: https://databricks-training.s3.amazonaws.com/slides/advanced-spark-training.pdf
- [9] SlideShare, ,'Introduction to Yarn and MapReduce', Online: https://www.slideshare.net/cloudera/introduction-to-yarn-and-mapreduce-2
- [10] Amazon Web Services Web Page, Online: https://aws.amazon.com
- [11] Databricks, A Tale of Three Apache Spark APIs: RDDs, DataFrames, and Datasets, Online: https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html

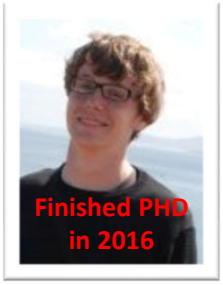
Lecture Bibliography (2)

- [12] YouTube Video, 'Solving Big Data with Apache Spark', Online: https://www.youtube.com/watch?v=WFoFLJOCOLA
- [13] The Berkeley Data Analytics Stack: Present and Future, Online: http://events-tce.technion.ac.il/files/2014/04/Michael-Franklin-UC-Berkeley.pdf
- [14] Apache Hive, Online: https://hive.apache.org/
- [15] Mining of Massive Datasets, Online: http://infolab.stanford.edu/~ullman/mmds/book.pdf
- [16] L. Page, S. Brin, R. Motwani, T. Winograd, 'The PageRank Citation Ranking: Bringing Order to the Web', Technical Report of the Stanford InfoLab, 1999
- [17] www.big-data.tips, 'Page Rank Technique', Online: http://www.big-data.tips/page-rank
- [18] Yahoo Web page, Online: http://www.yahoo.com
- [19] Conviva Web page, Online: http://*www.conviva.com*
- [20] Clear Story Data Web page, Online: https://www.clearstorydata.com/
- [21] YouTube Video, 'Run PySpark on Google Colab for FREE! | PySpark on Jupyter', Online: https://www.youtube.com/watch?v=kHtl2I1tXgY
- [22] YouTube Video, 'Visualization of k-means clustering', Online: https://www.youtube.com/watch?v=nXY6PxAaOk0

Lecture Bibliography (3)

- [23] Google Colaboratory, Online: https://colab.research.google.com
- [24] Understanding Parallelization of Machine Learning Algorithms in Apache Spark, Online: https://www.slideshare.net/databricks/understanding-parallelization-of-machine-learning-algorithms-in-apache-spark/11
- [25] www.big-data.tips, 'Gradient Descent', Online: http://www.big-data.tips/gradient-descent

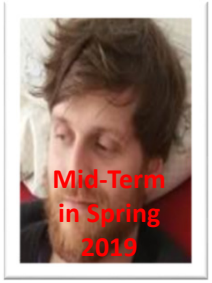
Acknowledgements – High Productivity Data Processing Research Group



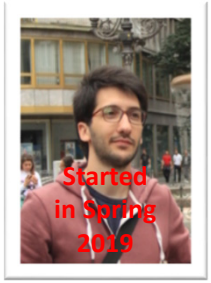
PD Dr.
G. Cavallaro

Senior PhD
Student A.S. Memon

Senior PhD
Student M.S. Memon



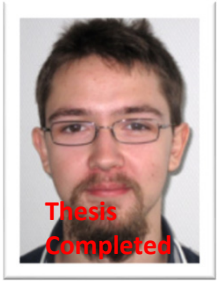
PhD Student
E. Erlingsson



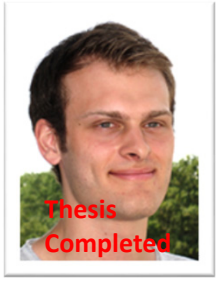
PhD Student
S. Bakarat

PhD Student
R. Sedona

Dr. M. Goetz
(now KIT)



MSc M.
Richerzhagen
(now other division)



MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now
Soccerwatch.tv)

MSc Student
G.S. Guðmundsson
(Landsverkjun)

This research group has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 763558 (DEEP-EST EU Project)


services
measurement
within
fusion
new
funding
Policy-based
concepts
device
Analysis
forms
cross-disciplinary
resources
Centre
climate
Computer
expertise
Approaches
construction
Fusion
year
like
Computational
Test
E.g
Computing
disciplines
results
Enable
different
Cross-Disciplinary
methods
High
big
nodes
PB
community
Environment
Modelling
EU
Science
centers
Key
scientific
important
files
model
international
etc
DKRZ
Use
storage
basis
Infrastructure
simulations
ENES
technologies
SMAQ
engineering
References
formats
brain
Data
increasing
via
EUDAT
physical
IPCC
Big
Project
processing
Common
TB
MapReduce
create
computing
Using
Methods
European
databases
GB
Supercomputing
Work
images
codes
analysis
Large
Summary
data-intensive
Service
Scientific
levels
often
stored
Resources
analyze
external
DLCL
Level
Research
Matters
Concrete
performance
JSC
use
computational
research
e.g
science
HPC
Juelich
system
datasets
acquisition
large-scale
compute
Landscape
management
System
Climate
modelling
AR5
Hardware
distributed
cores
access
hundreds
Nutshell
Services
DLCLs
Understanding
Technologies
Earth
Structure
current
network
structures
directory
project
General
Germany
Simulation
Processing
Provide
NASA
Energy
systems
day
Health