PyData Bratislava

GAP DATA INSTITUTE

Building AI pipelines using PySpark

(PyData Bratislava Meetup #3, Nervosa)

15. 5. 2017

Matúš Cimerman, Exponea #PyDataBA

Building AI data pipelines using PySpark

Matus Cimerman, matus.cimerman@{gmail,exponea}.com

EXPONEA

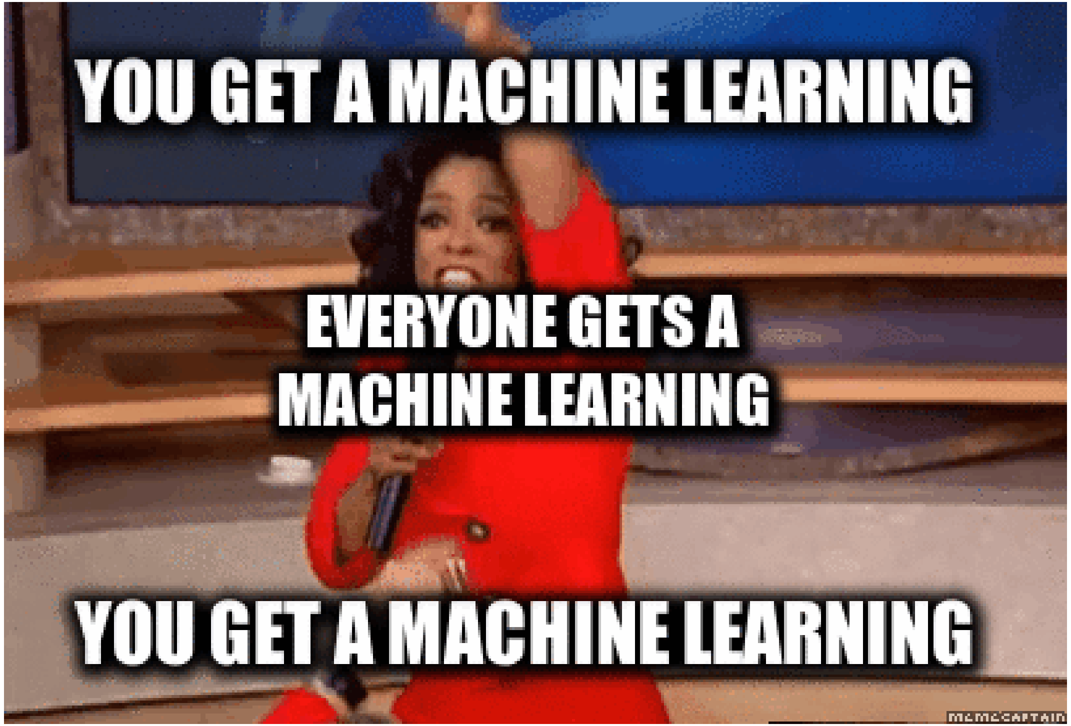
About me

- 1+y Data science @ Exponea, before BI intern and other stuff @ Orange.
- FIIT STU, Data Streams related studies
- Some links:
  - https://www.facebook.com/matus.cimerman
  - https://twitter.com/MatusCimerman
  - https://www.linkedin.com/in/mat%C3%BA%C5%A1-cimerman-4b08b352/
  - **Github link soon**

Few words regarding this talk

1. Spoken word will be in Slovak for this time.
2. This talk is not about machine learning algorithms, methods, hyperparameters tuning or anything similar.
3. I am still newbie and learner, don't hesitate to correct me.
4. Goal is to show you overall basics, experts hold your hate.
5. First time publicly speaking, prepare your tomatoes.
6. Comic Sans Font is used intentionally, I was told it's OK for slides.

Aren't you doing ML? WTF mate?

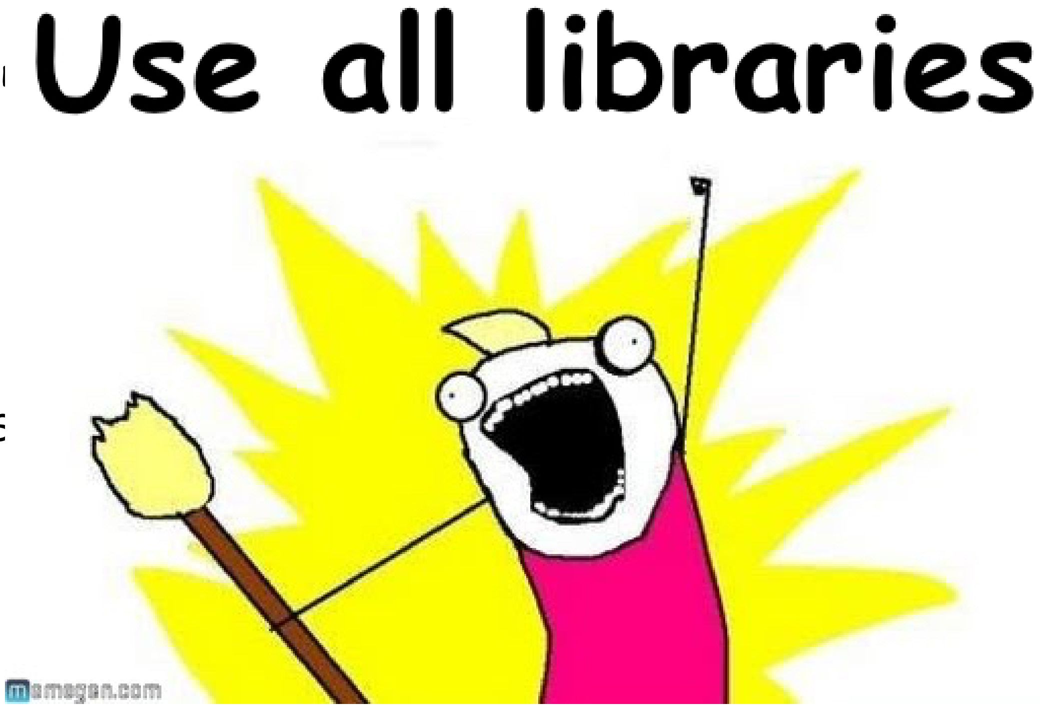


Data preprocessing nightmare*

Patter

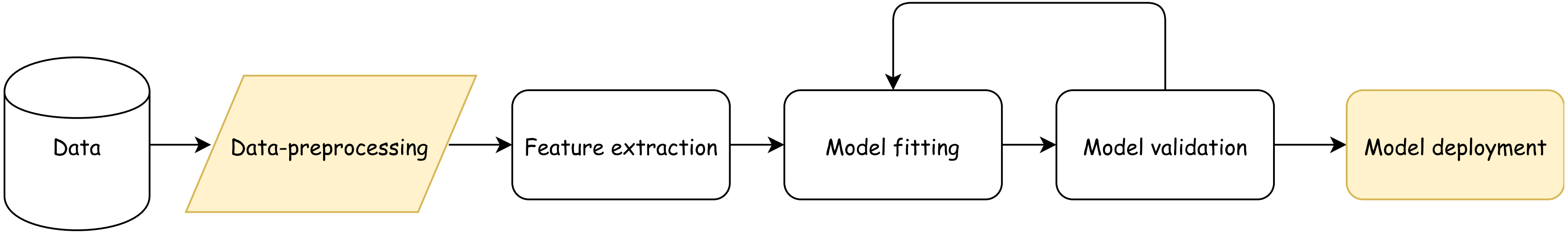
Dora

S

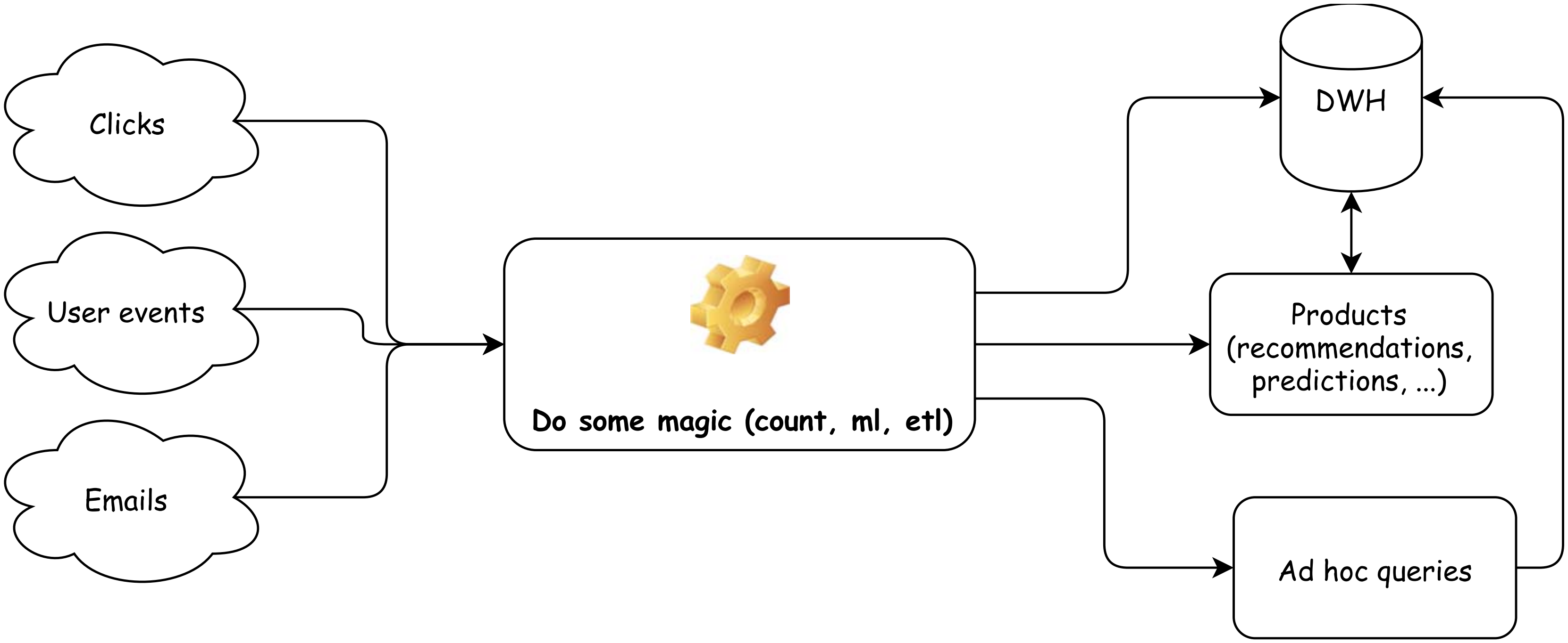


*I do like all of those libraries

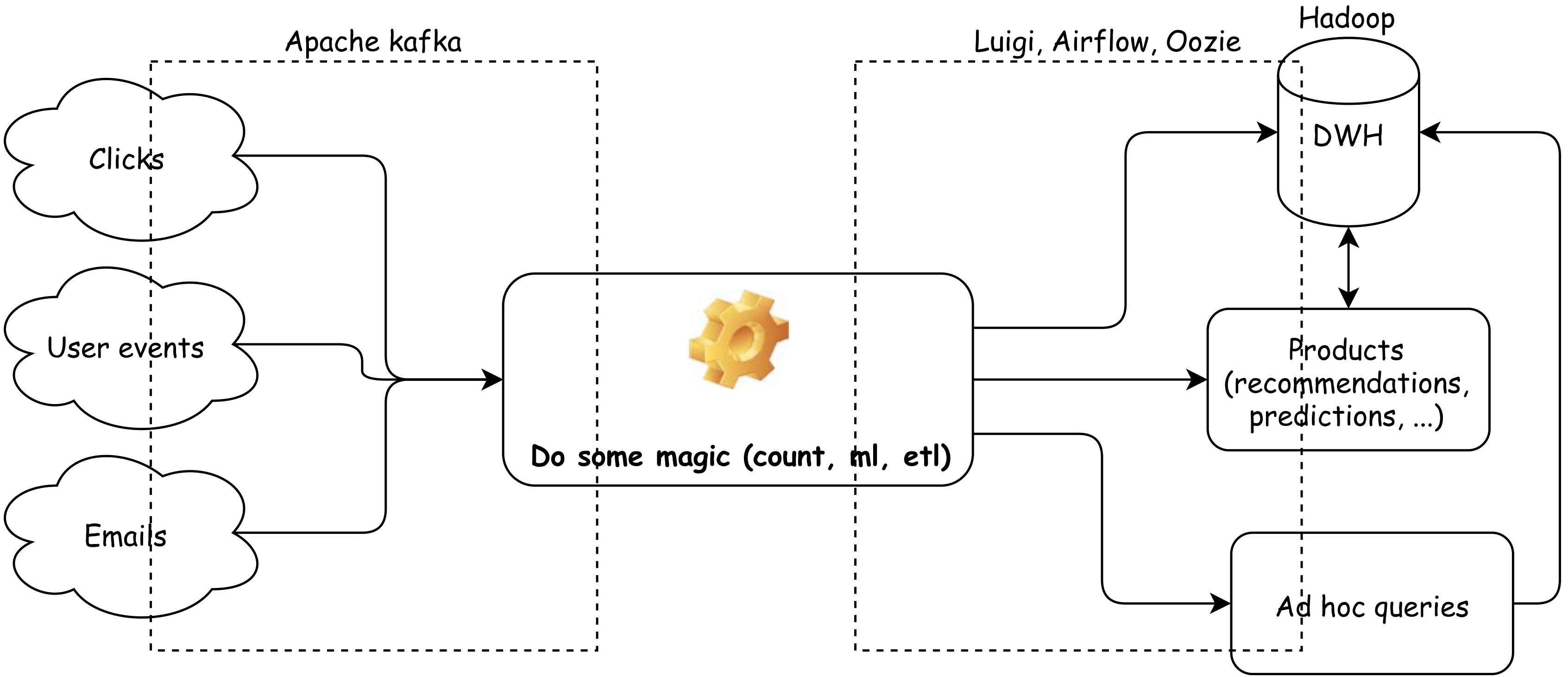
A practical ML pipeline



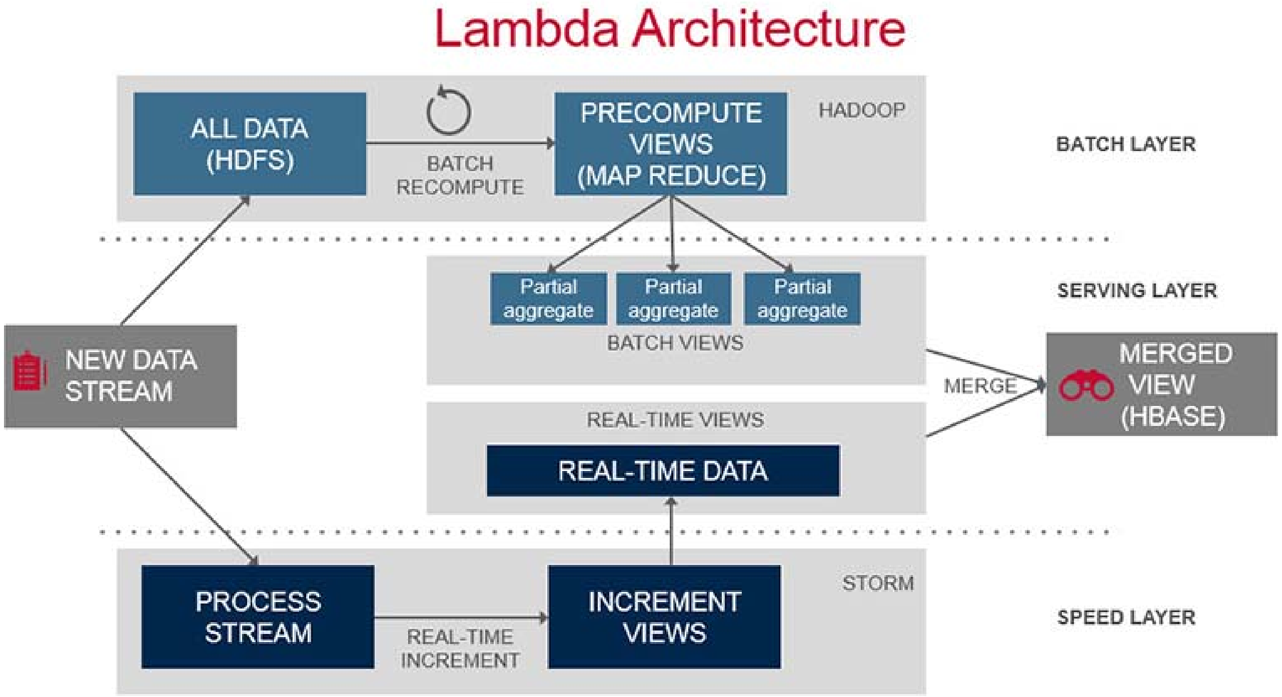
Data pipeline



Data pipeline

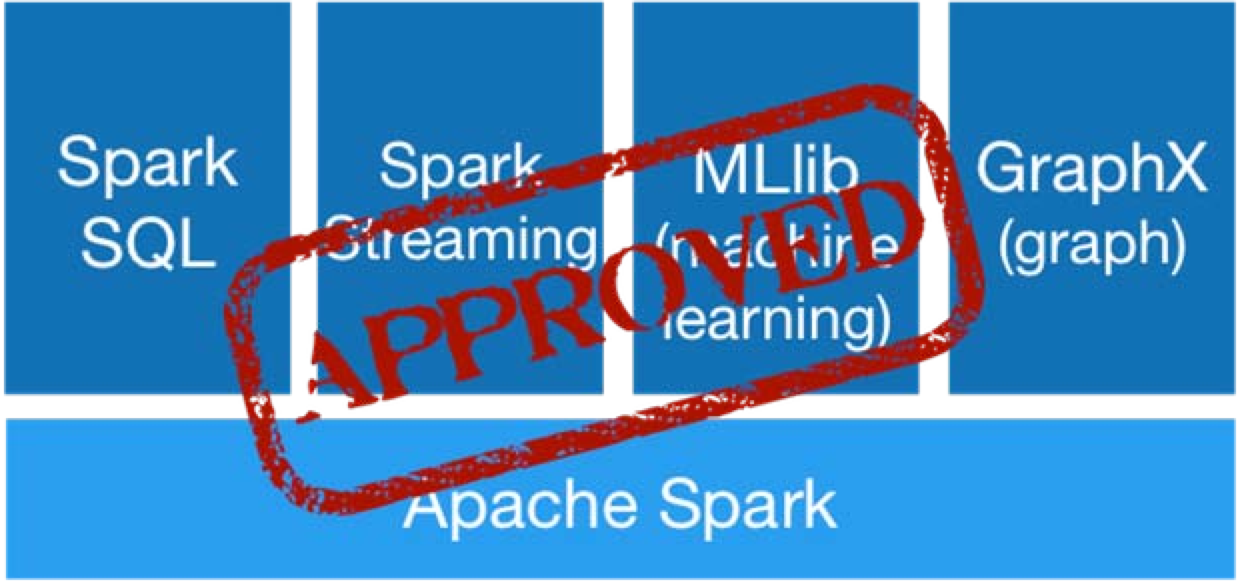


Lambda architecture

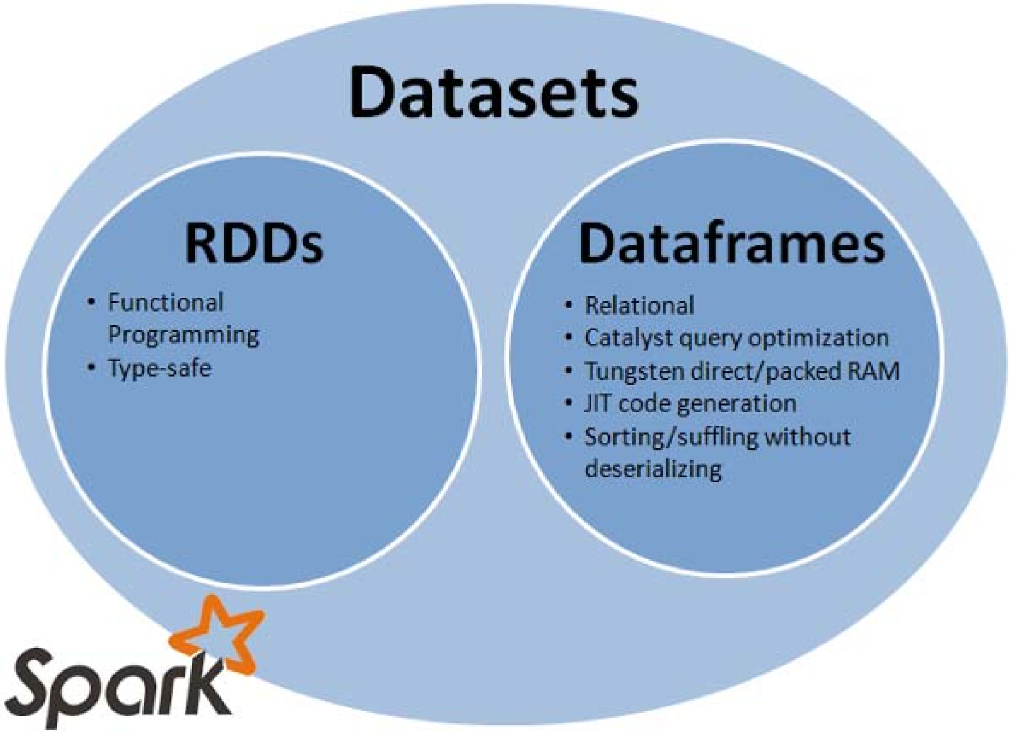


http://lambda-architecture.net/, https://mapr.com/developercentral/lambda-architecture/

Connecting dots is not easy for large-scale datasets



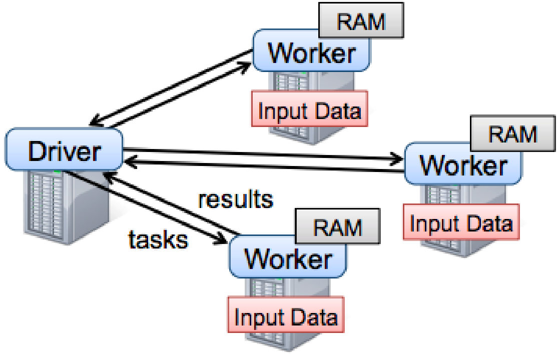
Apache Spark basics

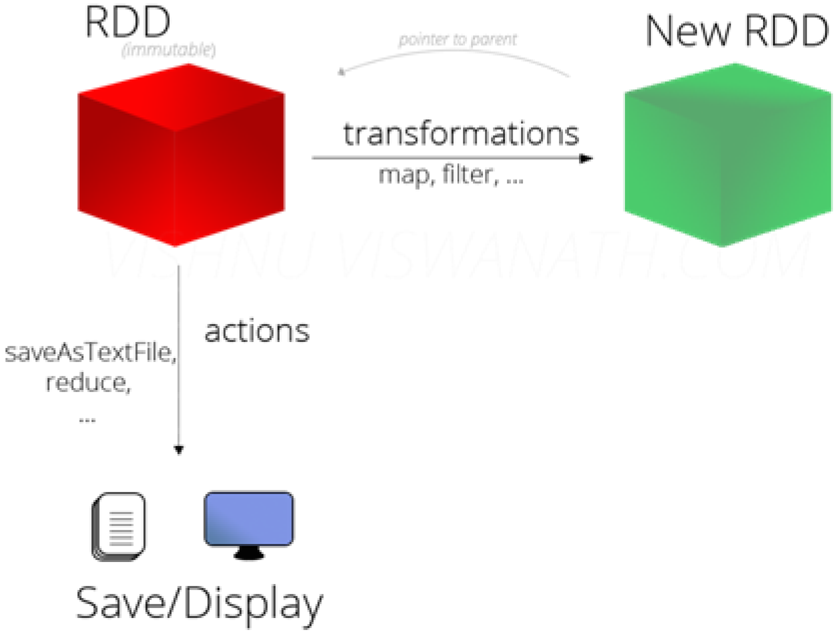


http://why-not-learn-something.blogspot.sk/2016/07/apache-spark-rdd-vs-dataframe-vs-dataset.html

Resilient Distributed Datasets (RDDs)

- Distributed, immutable
- Lazy-execution,
- Fault-tolerant
- Functional style programming (actions, transformations)
- Can be persisted/cached in memory/disk for fast iterative





http://vishnuviswanath.com/spark_rdd.html

Zaharia, Matei, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012.

Resilient Distributed Datasets (RDDs)

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

Zaharia, Matei, et al. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012.

RDD example (1)

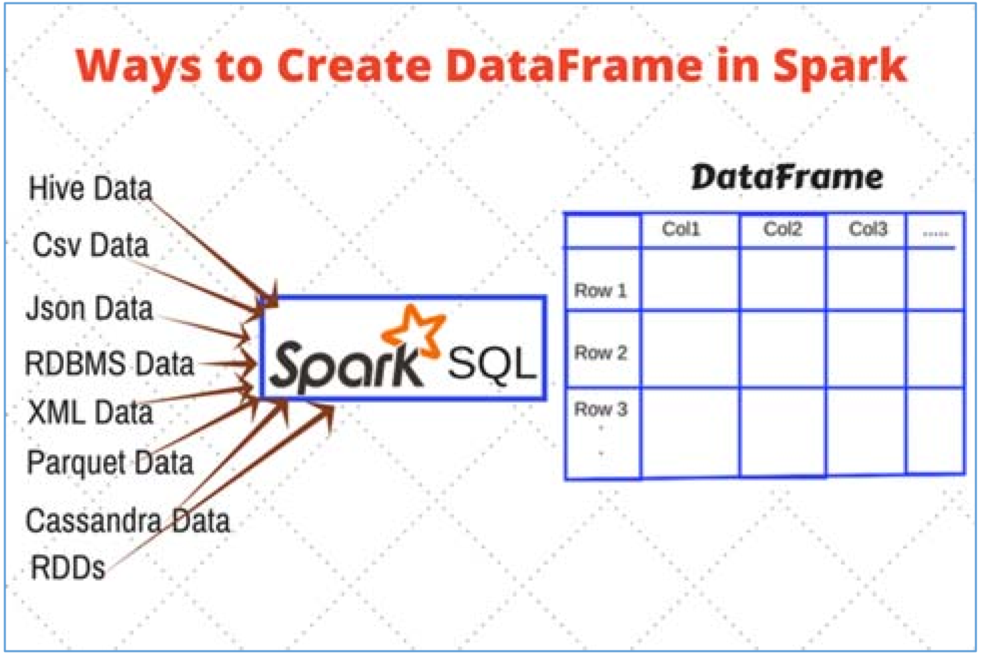
```
from random import random


def f(number):
    return (0, number) if number < 0.5 else (1, number)

rdd = sc.parallelize([random() for i in range(1000)])
rdd.take(2)  # [0.8528183968066678, 0.3513345834291187]
rdd.filter(lambda x: x > 0.95).count()  # 53

new_rdd = rdd.persist().map(f)  # Nothing happened - lazy eval

new_rdd.countByKey()  # {0: 481, 1: 519}

new_rdd.reduceByKey(lambda a,b: a + b).take(2)  # [(0, 110.02773787885363), (1, 408.68609250249494)]
```

RDD example (2)

```python
from pyspark.mllib.feature import Word2Vec

inp = sc.textFile("/apps/tmp/text8").map(lambda row: row.split(" "))

word2vec = Word2Vec()
model = word2vec.fit(inp)

synonyms = model.findSynonyms('research', 5)

for word, cosine_distance in synonyms:
    print("{}: {}".format(word, cosine_distance))
    """
    studies: 0.774848618142
    institute: 0.71544256553
    interdisciplinary: 0.684204944488
    medical: 0.659590635889
    informatics: 0.648754791132
    """
```

DataFrames

- Schema view of data
- Also lazy like RDD
- Significant performance improvement in compare to RDDs (Tungsten & Catalyst optimizer)
- No serialization between stages
- Great for semi-structured data
- RDD on the background
- Can be created from RDD



https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/

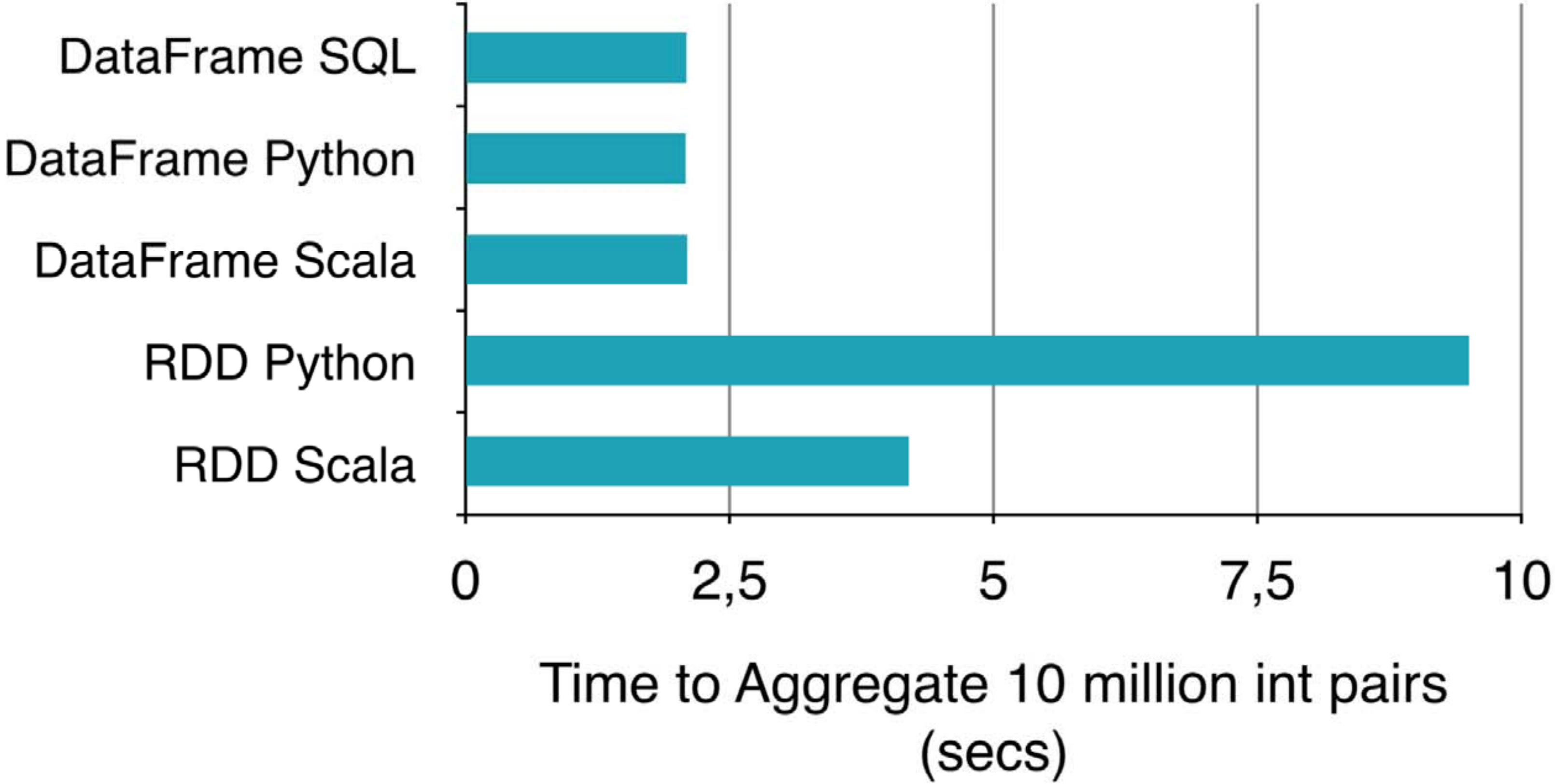
DataFrame schema

```
root
 |-- create_time: long (nullable = true)
 |-- id: string (nullable = true)
 |-- properties: struct (nullable = true)
 |    |-- age: string (nullable = true)
 |    |-- birthday: long (nullable = true)
 |    |-- city: string (nullable = true)
 |    |-- cookie_id: string (nullable = true)
 |    |-- created_ts: double (nullable = true)
 |    |-- email: string (nullable = true)
 |-- raw: string (nullable = true)
 |-- ids: struct (nullable = true)
 |    |-- cookie: array (nullable = true)
 |    |    |-- element: string (containsNull = true)
 |    |-- registered: array (nullable = true)
 |    |    |-- element: string (containsNull = true)
```

DataFrame example

```
users = spark.read.parquet('project_id=9e898732-a289-11e6-bc55-14187733e19e')

users.count()  # 49130

# SQL style operations
users.filter("properties.gender == 'female'").count()  # 590

# Expression builder operations
users.filter(users.properties.gender.like("female")).count()  # 590

# Show results
users.filter(users.properties.gender.like("female")).select('properties.age').describe('age').show()

+-------+------------------+
|summary|               age|
+-------+------------------+
|  count   |               590|
|   mean   |        50.3762|
| stddev   |        20.5902|
|    min   |                15|
|    max   |                85|
+-------+------------------+
```

RDD, DF performance comparision



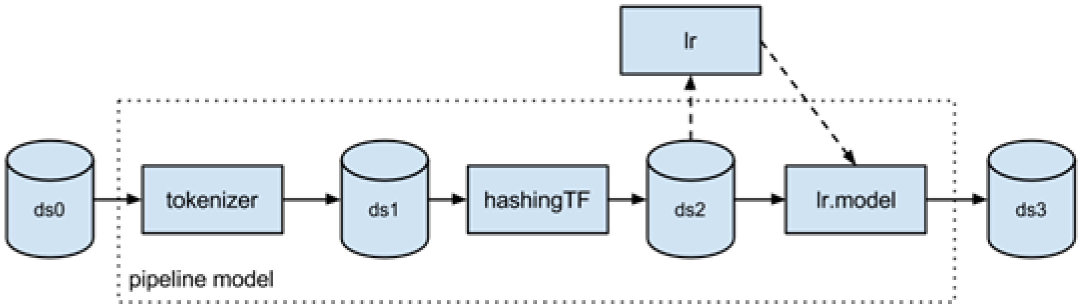
https://www.slideshare.net/databricks/2015-0616-spark-summit

Datasets

- Strongly typed
- …so, not available in Python

Spark ML pipelines - high level api

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import Tokenizer, HashingTF

tokenizer = Tokenizer() \
    .setInputCol("text") \
    .setOutputCol("words")

hashingTF = HashingTF()
.setNumFeatures(1000) \
    .setInputCol(tokenizer.getOutputCol) \
    .setOutputCol("features")

lr = LogisticRegression() \
    .setMaxIter(10) \
    .setRegParam(0.01)

pipeline = new
Pipeline() \
    .setStages([tokenizer, hashingTF, lr])

model = pipeline.fit(trainingDataset)
```



https://databricks.com/blog/2015/01/07/ml-pipelines-a-new-high-level-api-for-mllib.html

Spark ML pipelines - cross validation

```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator


paramGrid = ParamGridBuilder()\
  .addGrid(hashingTF.numFeatures, [10, 20, 40])\
  .addGrid(lr.regParam, [0.01, 0.1, 1.0])\
  .build()

cv = CrossValidator()\
  .setNumFolds(3)\
  .setEstimator(pipeline)\
  .setEstimatorParamMaps(paramGrid)\
  .setEvaluator(BinaryClassificationEvaluator)

cv.save('cv-pipeline.parquet')

cvModel = cv.fit(trainingDataset)
cvModel.save('cv-model.parquet')
```

ML persistence

- You can create model in Python and deploy in Java/Scala app
- Support for almost all Mllib algorithms
- Support for fitted and unfitted Pipelines, so that also Pipelines are exchangeable
- Suited for large distributed models – binary format Parquet is used to store model data
- Metadata and model parameters are stored in JSON format

```
paramGrid = ParamGridBuilder()...

cv = CrossValidator().setEstimator(pipeline)...
cv.save('cv-pipeline.parquet')

cvModel = cv.fit(trainingDataset)
cvModel.save('cv-model.parquet')
```

Submitting apps

1. Locally, suitable for dev (avoid this in production)
2. Cluster in client mode
3. Cluster in cluster mode

Running Spark locally

fastest way

Submitting apps to YARN

config.yml

```yaml
spark.app.name: "PyData Bratislava 2017"
spark.master: "yarn"
spark.submit.deployMode: "client"
spark.yarn.dist.files: "file:/pyspark.zip,file:py4j-0.10.3-src.zip"
spark.executorEnv.PYTHONPATH: "pyspark.zip:py4j-0.10.3-src.zip"
spark.executorEnv.PYTHONHASHSEED: "0"
spark.executor.instances: "12"
spark.executor.cores: "3"
spark.executor.memory: "6g"
```

```python
import yaml

from pyspark import SparkConf, SparkContext


config = yaml.load('config.yml')
sparkConf = SparkConf().setAll([(k, v) for k, v in config.items()])
spark_context = SparkContext(conf=sparkConf).getOrCreate()
```

Submitting apps to YARN

```bash
#!/usr/bin/env bash

# 1. Create virtualenv
virtualenv venv --python=/usr/bin/python3.5

# 2. Create zip file with all your python code
zip -ru pyfiles.zip * -x "*.pyc" -x "*.log" -x "venv/*"

# 3. Submit your app
PYSPARK_PYTHON=/venv/ /spark/spark-2.0.1/bin/spark-submit --py-files pyfiles.zip pipeline.py
```

Note:
- *venv* should be created on each physical node or HDFS so that every worker can use it
- Also any external config files should be either manually distributed or kept on HDFS

Jobs scheduling

- Jenkins hell
- Luigi
- Airflow

Invitation for hacking Thursday

- **Topic:** Offline evaluation of recommender systems
- This Thursday 5pm

Interested in demo? Let us know:

matus.cimerman@exponea.com