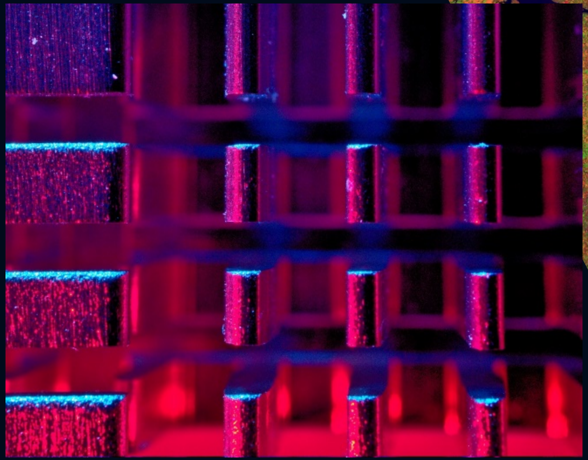
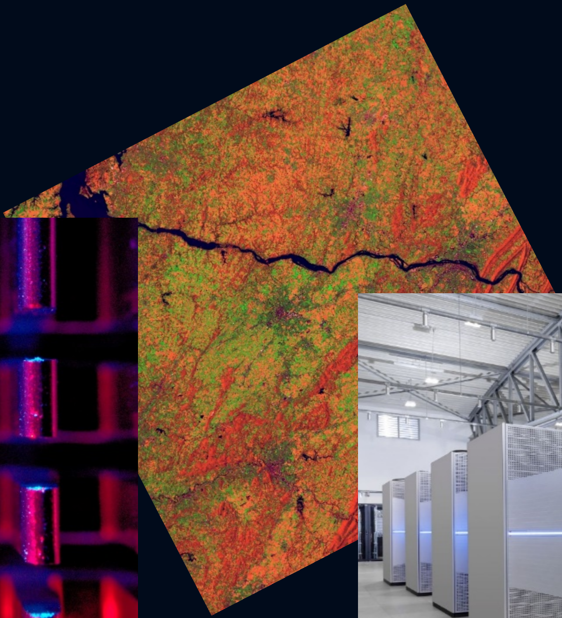




BIG DATA ANALYTICS USING APACHE SPARK

IEEE IGARSS 2021 Tutorial on Scalable Machine Learning with High Performance and Cloud Computing

DR. SHAHBAZ MEMON
HIGH PRODUCTIVITY DATA PROCESSING RESEARCH GROUP
JÜLICH SUPERCOMPUTING CENTRE

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

UNIVERSITY OF ICELAND

HDCRS

ESI

GRSS

igarss BRUSSELS 2021

STRUCTURE

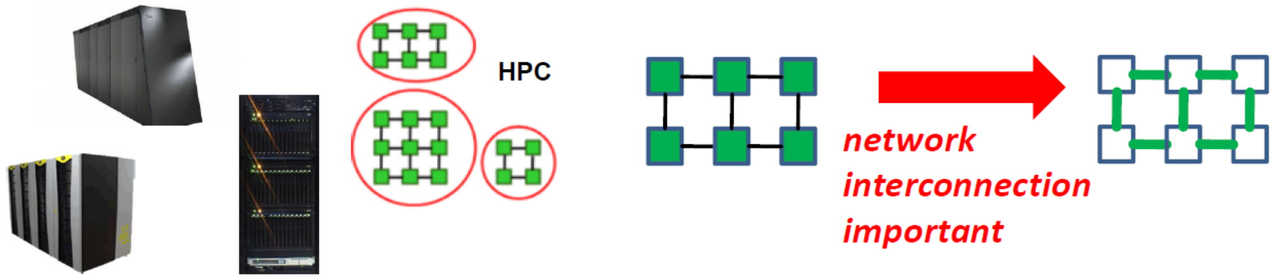
- Introduction
- Apache Spark Basics
- Developing on Spark
- Machine Learning on Spark
- Conclusions

INTRODUCTION

HPC AND HTC

Introduction

- High Performance Computing (HPC) is based on computing resources that enable the efficient use of parallel computing techniques through specific support with dedicated hardware such as high performance cpu/core interconnections. These are compute-oriented systems.

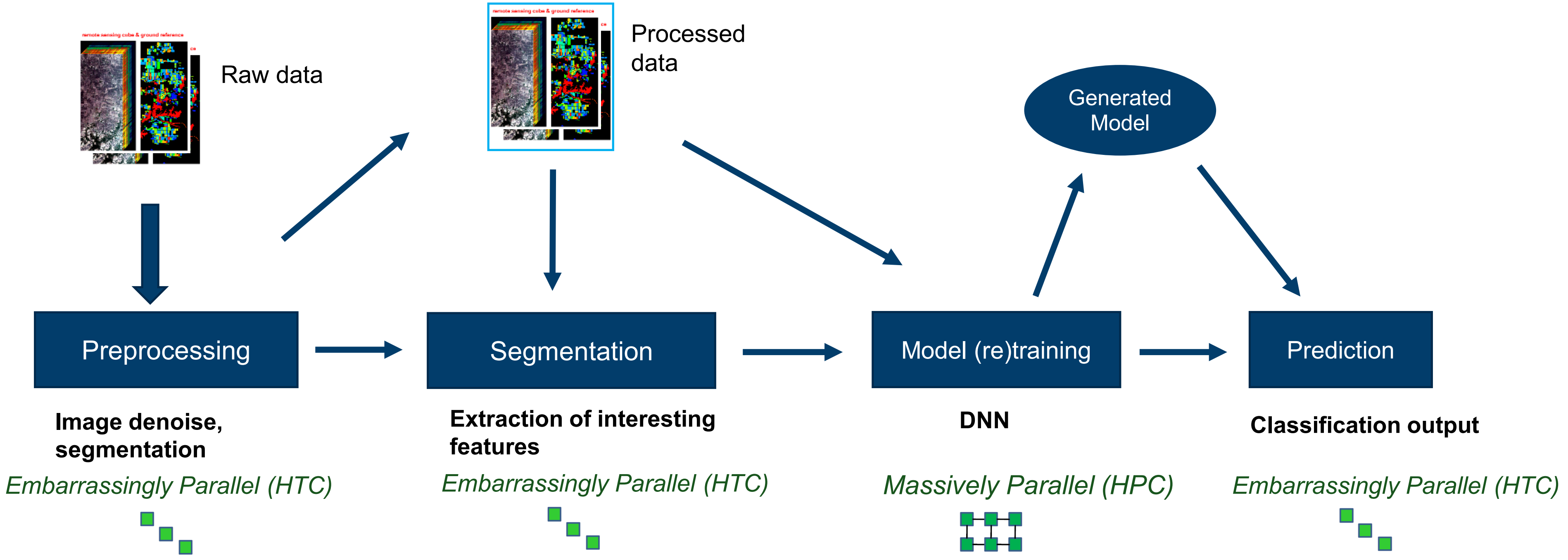


- High Throughput Computing (HTC) is based on commonly available computing resources such as commodity PCs and small clusters that enable the execution of ‘farming jobs’ without providing a high performance interconnection between the cpu/cores. These are data-oriented systems

Riedel [14]

END-TO-END IMAGE ANALYTICS

Motivational Scenario



QUESTIONS TO PONDER

- Pre- and post-processing in the scope of application, rather than on one step
- Manage all workflow tasks within one framework e.g. end-to-end Deep Learning
- Data export and import from multiple kind of storage systems
- Data-intensive rather than compute-intensive processing

BIG DATA ANALYTICS

- Support of multiple algorithms and frameworks
  - Machine Learning and Deep Learning
  - Integrated processing with HTC, HPC and ML/DL frameworks
- Abstract parallelization complexity from user
  - Parallel processing, batch systems, environmental intricacies are abstracted
- Encapsulate distributed computing and storage infrastructure details
  - Operating systems, security, networks and security interfaces

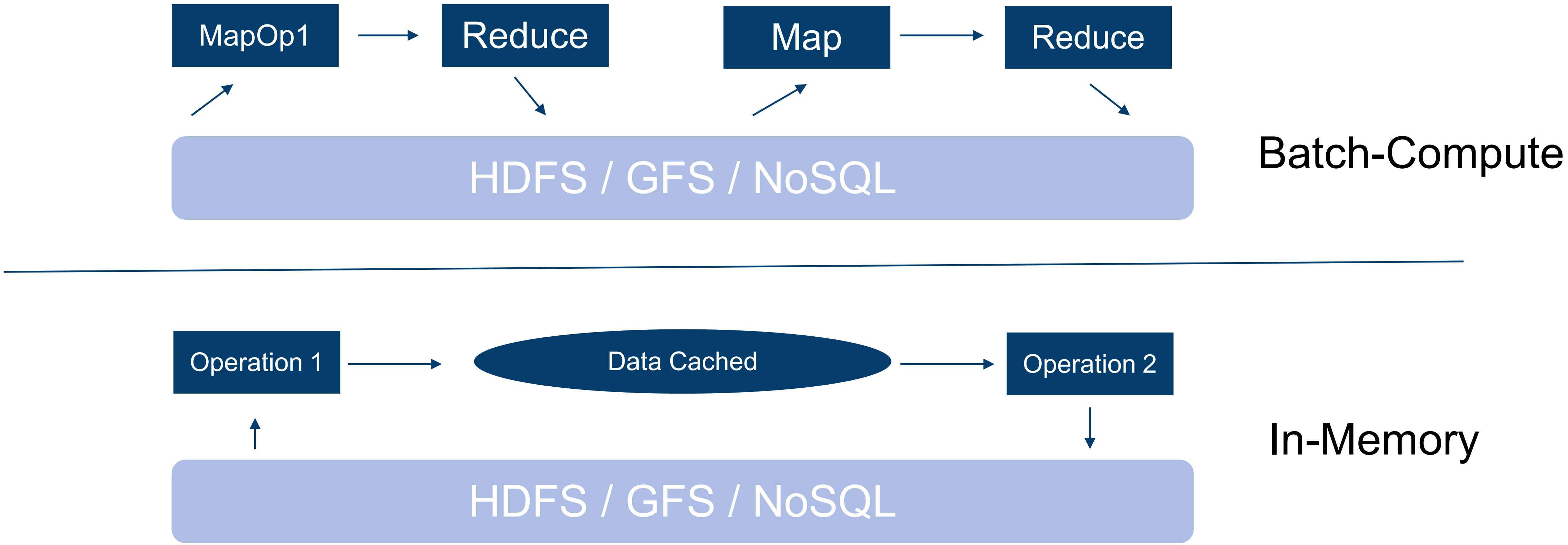
BIG DATA USE CASES

- Web mining and search (e.g Page Rank and Ad Analytics)
- Stream analytics (Twitter, Facebook and Trading)
- Graph processing
- IoT (Remote sensing, Automotive and Smart devices)
- Large scale image and video processing
- Time-series analysis

BIG DATA ANALYTIC FRAMEWORKS

- **Motto: Bring compute to data**
- Batch Processing
  - Manage job requests as batches
  - Map-reduce framework: Large problem space to many small tasks
  - E.g. Apache Hadoop
- In-Memory processing
  - Data processing in memory
  - Efficient map-reduce, filter and transform, Extract Transform and Load (ETL)
  - E.g. Apache Spark (Focus of this talk) and Apache Flink

IN-MEMORY: MORE I/O EFFICIENCY



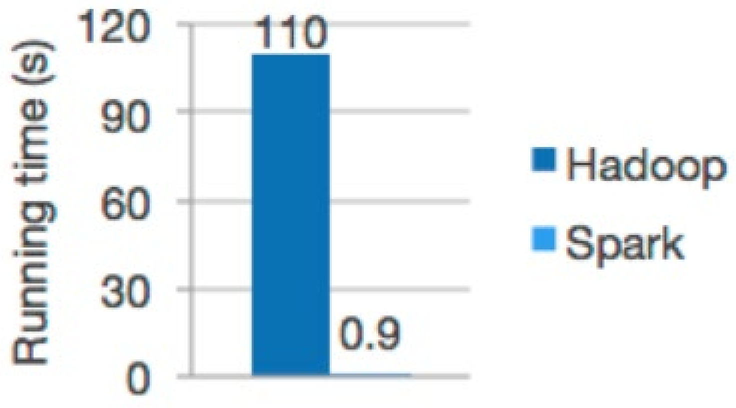
APACHE SPARK BASICS

APACHE SPARK

In-Memory Computing Framework

- Open source, unified analytics [2] engine for distributed and parallel data processing
  - Data transformations + AI and ML

- Provides a set of extensible APIs for
  - SQL for interactive queries
  - Machine learning
  - Stream processing
  - Graph processing

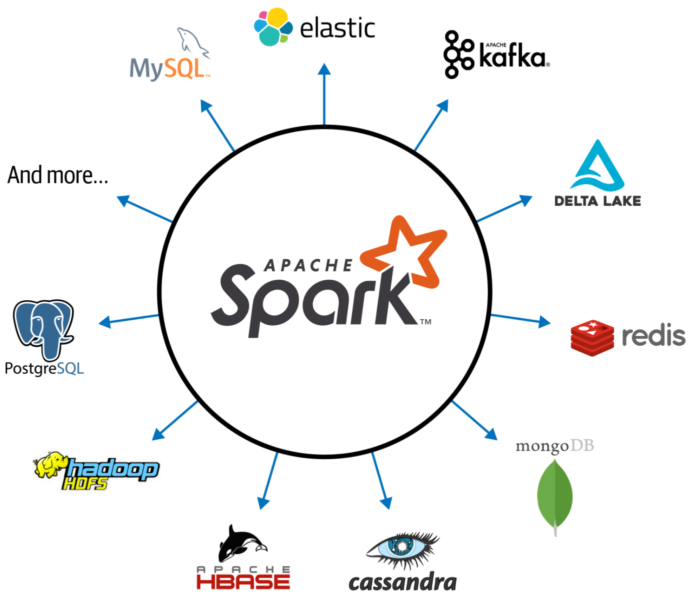


Logistic regression in Hadoop and Spark

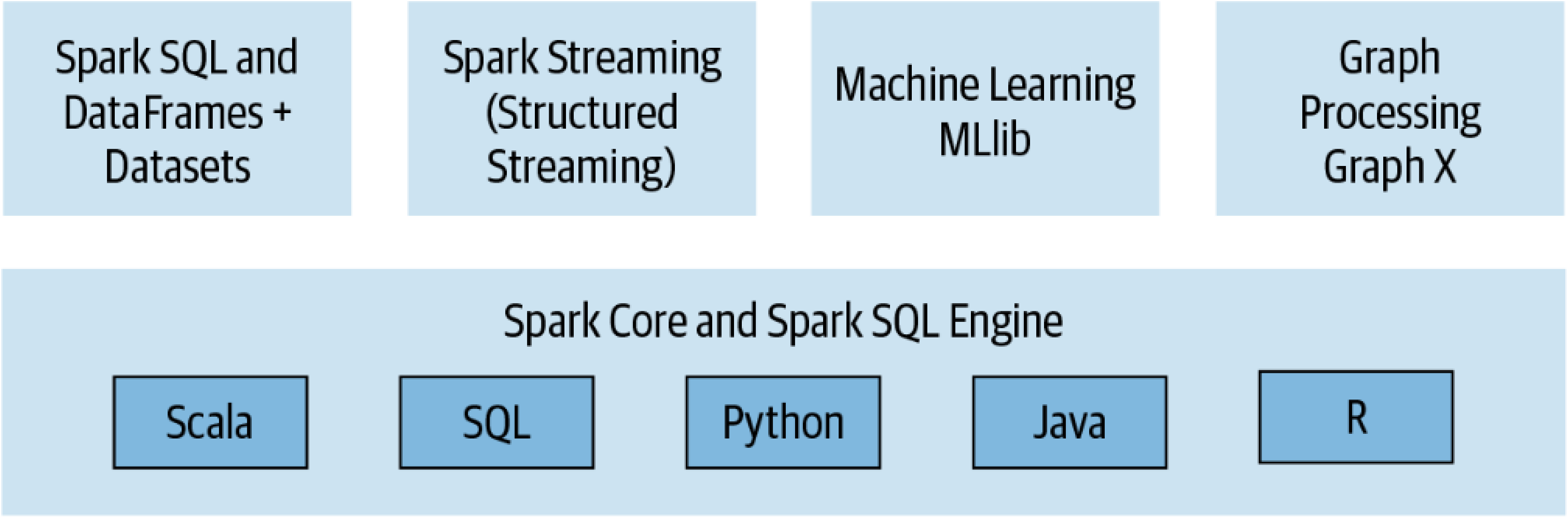
spark.apache.org [2]

ECOSYSTEM

- Apache Project and open source
- Databricks: main development driver
- Supported by major cloud computing providers
  - E.g. Amazon and Google
- Integrated with multiple schedulers, file systems, DBMS and data stores, a.k.a Connectors
- Hadoop (Batch processing) supported
  - Seamlessly complement / replace Map-Reduce layer
- HPC Supported
  - SLURM as a scheduler



APACHE SPARK COMPONENTS



Learning Spark [2]

APACHE SPARK COMPONENTS

- Spark SQL (Dataframes):
  - Standard SQL and Hive QL, but parallel execution
  - Data sources: JSON, Images, Parquet, Binary and Hive tables
- Spark Streaming (Structured Streaming)
  - High throughput and low latency scenarios
  - Log inflow, sensor data (IOT), Twitter streams
- Machine Learning (ML Lib)
  - Clustering, classification and recommender systems
  - Support Deep learning frameworks – Keras, Tensorflow and Pytorch
- Graph Processing (GraphX)
  - Iterative and parallel Graph computations
  - Social computing, semantic networks and link data

SPARK APPLICATION AREAS

Commercial and Scientific Applications

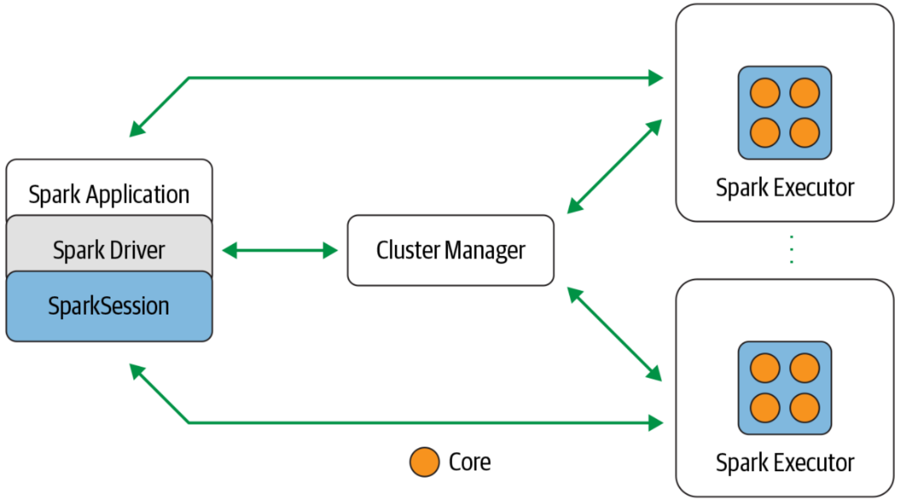
- Data processing pipeline (load, preparation and transform) include machine learning (Alibaba [7])

- Netflix recommendation ML pipeline (Netflix [8])

- Remote sensing and image analytics [5] and [6]

ARCHITECTURE

Application Concepts



Learning Spark [9]

- Jobs: A parallel computation
- Stages: A job is divided into stages
- Tasks: A single unit of execution

RESILIENT DISTRIBUTED DATASETS (RDD)

Data Structure Representation

- Basic programming abstraction (not used by every user level)
  - Dependencies: DAG structure of tasks
  - Partitions: Data locality and parallel computation on partitions
  - Iterator [T]: Handle to multiple type of collections
- Less expressive and complex
  - Computations are opaque
  - Difficult to introspect and debug
- New releases (> 3.0) prefer ***Spark DataFrames***

SPARK DATAFRAMES

Structured computing API – High level wrappers to RDDs

- Structured and the format is inspired by pandas DataFrames
- Distributed in-memory tables
  - Rows and columns, data types and schemas
  - E.g. integer, string, array and map
  - Simple and complex data types
- Scala (main implementation), Python, Java and R bindings
- Supports many formats as external data sources
  - Parquet, JSON, CSV, Images and Binary, etc

TRANSFORMATIONS AND ACTIONS

Operation Types

Transformations

- Transform Spark DataFrame into a new DataFrame without modifying the original data
- E.g. select, filter, groupBy, orderBy, join
- Lazy evaluation: not computed until action is called or any read / write occurs

Actions

- Compute operations
- E.g. show, take, count, collect, save

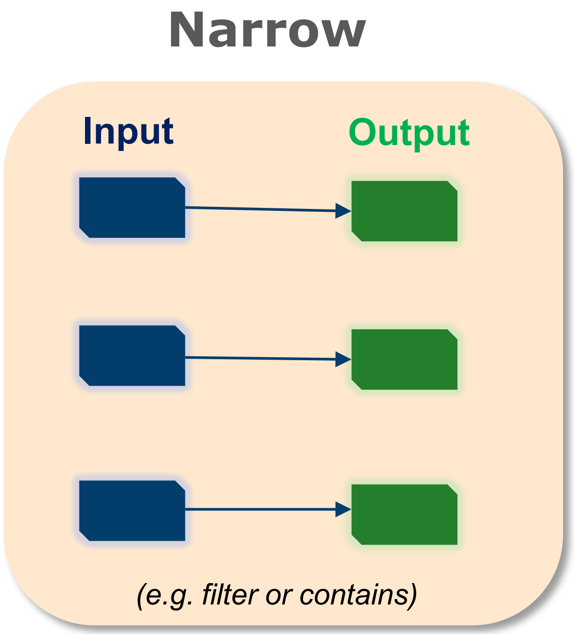
```
# In Python
>>> strings = spark.read.text("../README.md")
>>> filtered = strings.filter(strings.value.contains("Spark"))
>>> filtered.count()
20
```

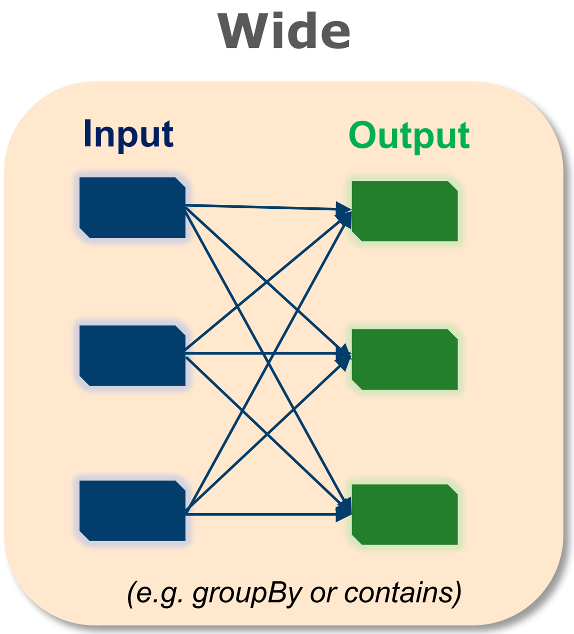
Narrow and Wide Transformations

- Wide transformations use multiple partitions

NARROW AND WIDE TRANSFORMATIONS

Transformation Types





SPARK EXECUTION WORKFLOW

1. Create or load data
   - e.g. load data set in DataFrames or RDDs

2. Apply 1-n tranformations data (narrow or wode)
   - e.g. select, filter, groupBy, orderBy

3. Perform actions that return or store data
   - E.g. reduce; count; collect

DEVELOPING ON SPARK

JÜLICH
Forschungszentrum

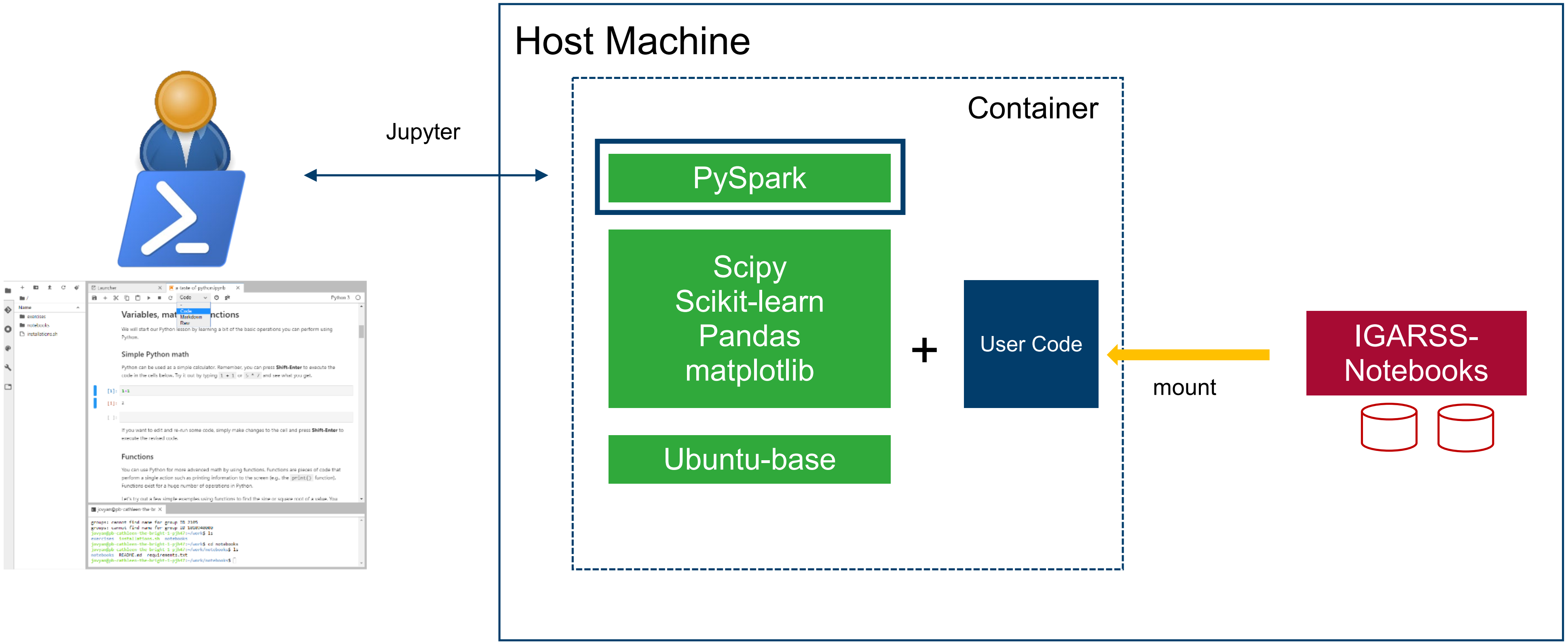
APACHE SPARK DEPLOYMENTS ON CLOUDS

- **Private Clouds:** in-premise deployments
  - E.g. OpenStack or Apache CloudStack or Containers
- **Public Clouds:** external provider, Pay-per use and Elastic scaling
  - Amazon EMR (Elastic Map-Reduce)
  - Microsoft Azure (Databricks or HDInsight)
  - Google (DataProc)

Mostly Virtualised Computing Infrastructures

JUPYTER-DOCKER STACK

User Development Environment



PYSPARK

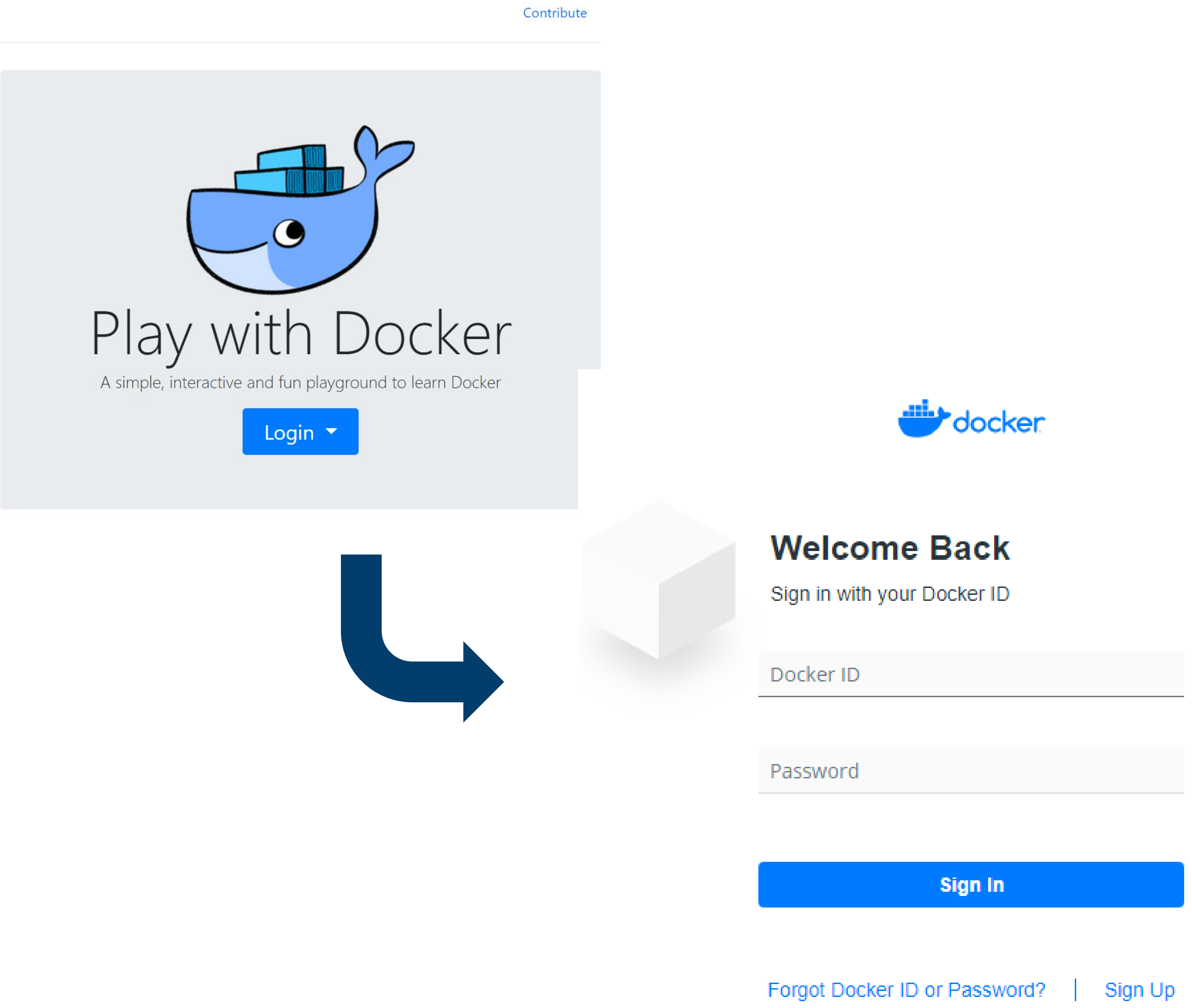
Embedded Development Environment

- Python bindings for Apache Spark (implemented in Scala)
- Mostly every functionality is available in Python
- Easily developed on Jupyter-lab instances
- PySpark Image Specifics
  - PySpark v 3.1.2
    - Includes core Spark libraries
  - PyArrow (for interoperability between Pandas and Spark Dataframes)
  - JAVA (OpenJDK) 11 and Scala 2.12.10

LAUNCHING THE PYSPARK CONTAINER

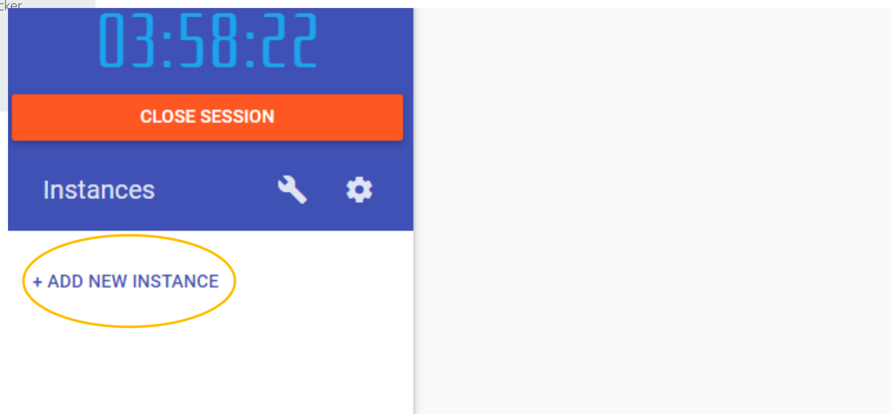
Based on Jupyter-Docker Stack

- Enter the following URL:
  - https://labs.play-with-docker.com/
- Press “Start”
- Login using your existing credentials or Sign up for a new Docker account (It is free)
- Account created

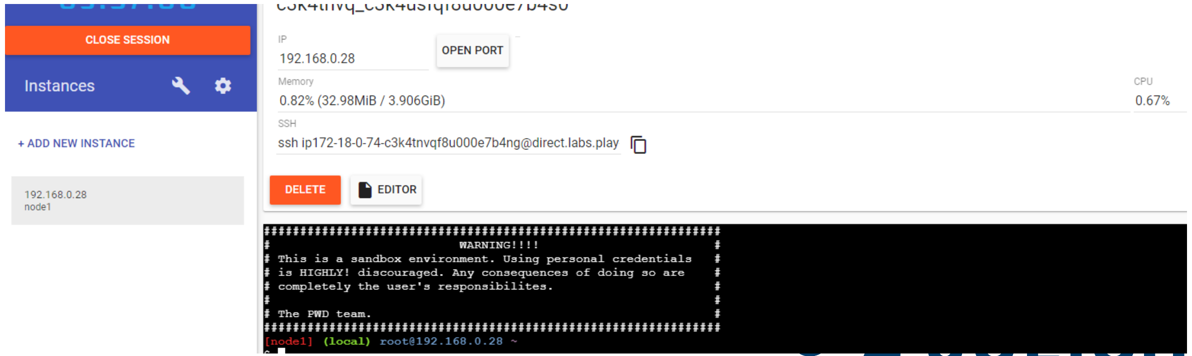


STARTING AN IMAGE

Hit “Start”



Click “Add New Instance”



Terminal is started

PREPARE THE WORKING DIRECTORY

- On the Terminal, write the following commands (step-wise)

$> mkdir igarss-nb (Press Enter)

$> chmod 777 igarss-nb (Press Enter)

$> cd igarss-nb (Press Enter)

LAUNCH THE PYSPARK IMAGE

$> docker run -p 8888:8888 -p 4040:4040 –p 4041:4041 –p 4042:4042 \

-v ~/igarss-nb:/home/jovyan/work jupyter/pyspark-notebook \

start-notebook.sh --NotebookApp.token='ig2021'

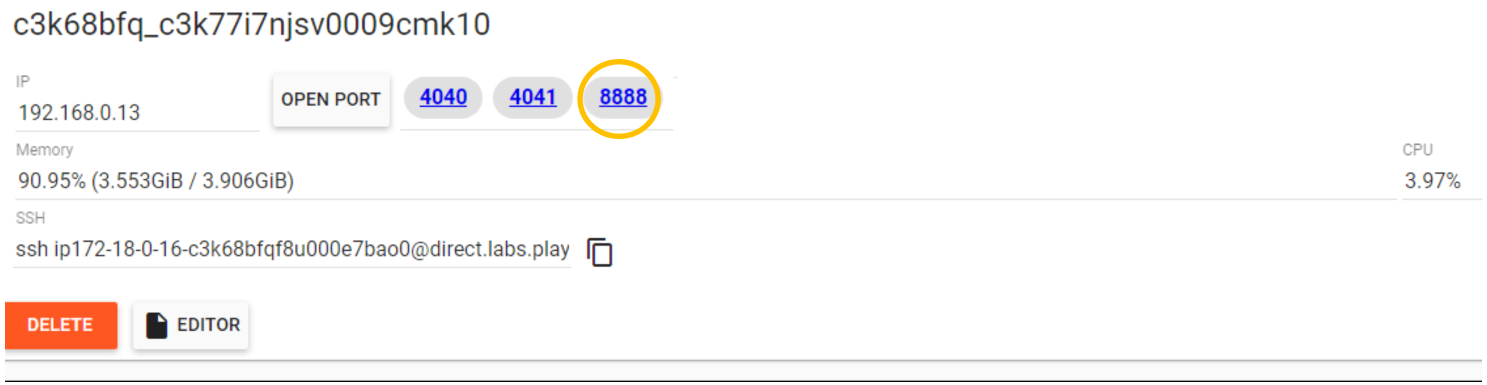
```
$ docker run -p 8888:8888 -p 4040:4040 -p 4041:4041 \
> -v ~/igarss-nb:/home/jovyan/work jupyter/pyspark-notebook \
> start-notebook.sh --NotebookApp.token='ig2021'
Unable to find image 'jupyter/pyspark-notebook:latest' locally
latest: Pulling from jupyter/pyspark-notebook
c549ccf8d472: Pull complete
6c65c50510f3: Pull complete
a89eb75169a1: Pull complete
4f4fb700ef54: Pull complete
c18636dc46c6: Pull complete
```

Server started.

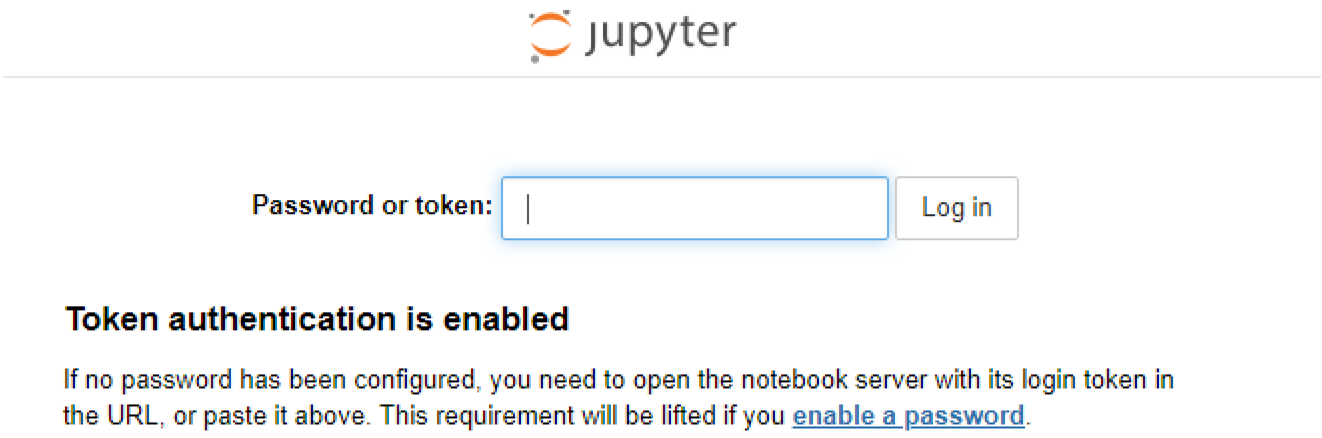
```
[I 2021-07-09 16:15:53.198 LabApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 16:15:53.207 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 16:15:53.207 NotebookApp] Jupyter Notebook 6.4.0 is running at:
[I 16:15:53.207 NotebookApp] http://b6679ed25797:8888/?token=...
[I 16:15:53.207 NotebookApp]  or http://127.0.0.1:8888/?token=...
[I 16:15:53.207 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

VIEW THE NOTEBOOK SERVER

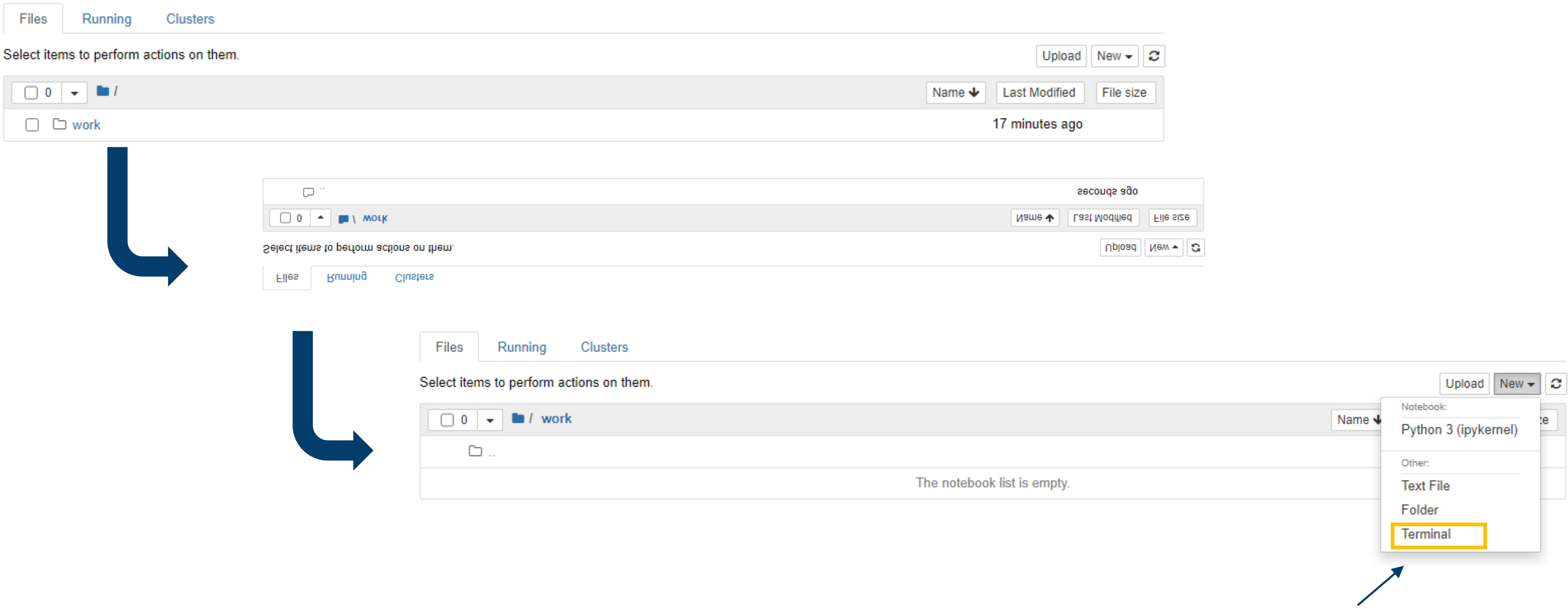
Click 8888



New tab opens and there enter **ig2021** and press Log in



OPEN THE JUPYTER TERMINAL



DOWNLOAD THE NOTEBOOKS ARCHIVE

On the shell type:
$ cd work
$ wget https://fz-juelich.sciebo.de/s/xiNXrpfOfmqLmMX/download

```
(base) jovyan@b6679ed25797:~$ cd work/
(base) jovyan@b6679ed25797:~/work$ wget https://fz-juelich.sciebo.de/s/xiNXrpfOfmqLmMX/downlo
--2021-07-09 16:37:10--  https://fz-juelich.sciebo.de/s/xiNXrpfOfmqLmMX/download
Resolving fz-juelich.sciebo.de (fz-juelich.sciebo.de)... 132.252.183.1
Connecting to fz-juelich.sciebo.de (fz-juelich.sciebo.de)|132.252.183.1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12029484 (11M) [application/gzip]
Saving to: 'download'
```

Untar the archive: $tar –xvf download

```
(base) jovyan@b6679ed25797:~$ tar -xvf download
IGARSS2021/
IGARSS2021/SparkUDFExample.ipynb
IGARSS2021/LogisticRegressionExample.ipynb
IGARSS2021/datasets/
```

Switch to the previous browser tab and click IGARSS2021

JÜLICH
Forschungszentrum

BROWSER VIEW

Demonstration: Open “BasicDataFrame.ipynb”

USER DEFINED FUNCTIONS (UDF)

Explicit Customization

- Types: Simple and Pandas UDF
- Define new domain specific modules that extend the vocabulary of Spark's built-in functions
- Useful for data normalization and cleaning (e.g. handling nulls, feature scaling)
- **Simple UDF:** Row-wise operation on a data frame – sequential processing, see example notebook *(next slide)*
- **Pandas UDF:** Vectorized operations (process entire array at once)

```
import pandas as pd
from scipy import stats

@pandas_udf('double')
def cdf(v):
    return pd.Series(stats.norm.cdf(v))


df.withColumn('cumulative_probability', cdf(df.v))
```

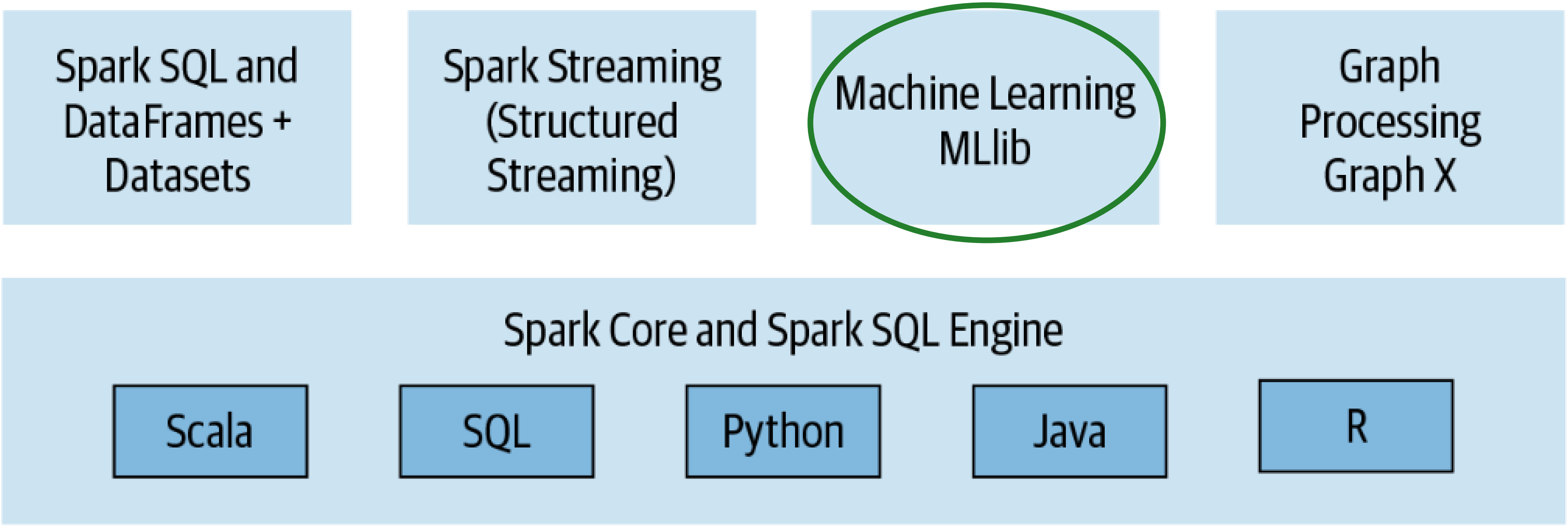
[12] Pandas UDF

JÜLICH Forschungszentrum

Demonstration: Open “SparkUDFExample.ipynb”

MACHINE LEARNING ON SPARK

MACHINE LEARNING WITH SPARK



Learning Spark [2]

SPARKML (MLLIB)

- Promises Machine Learning at Scale
- Parallel processing made easy
  - Develop locally (e.g. Jupyter Notebook) -> deploy on cluster
- MLLib features
  - Distributed with ML algorithms (clustering, classification..)
  - Parallel implementations
  - Processing data is cached in-memory (optimal for iterative algorithms)
  - Support of Python, Scala, Java, R

SPARK ML CONCEPTS

- **Transformer:** Data preparation and rule-based transformations. Input DataFrame and output a new DataFrame instance.

```
newDF = myDF.transform()
```

- **Estimators:** Learning or fitting parameters. Returns a Model (a transformer)

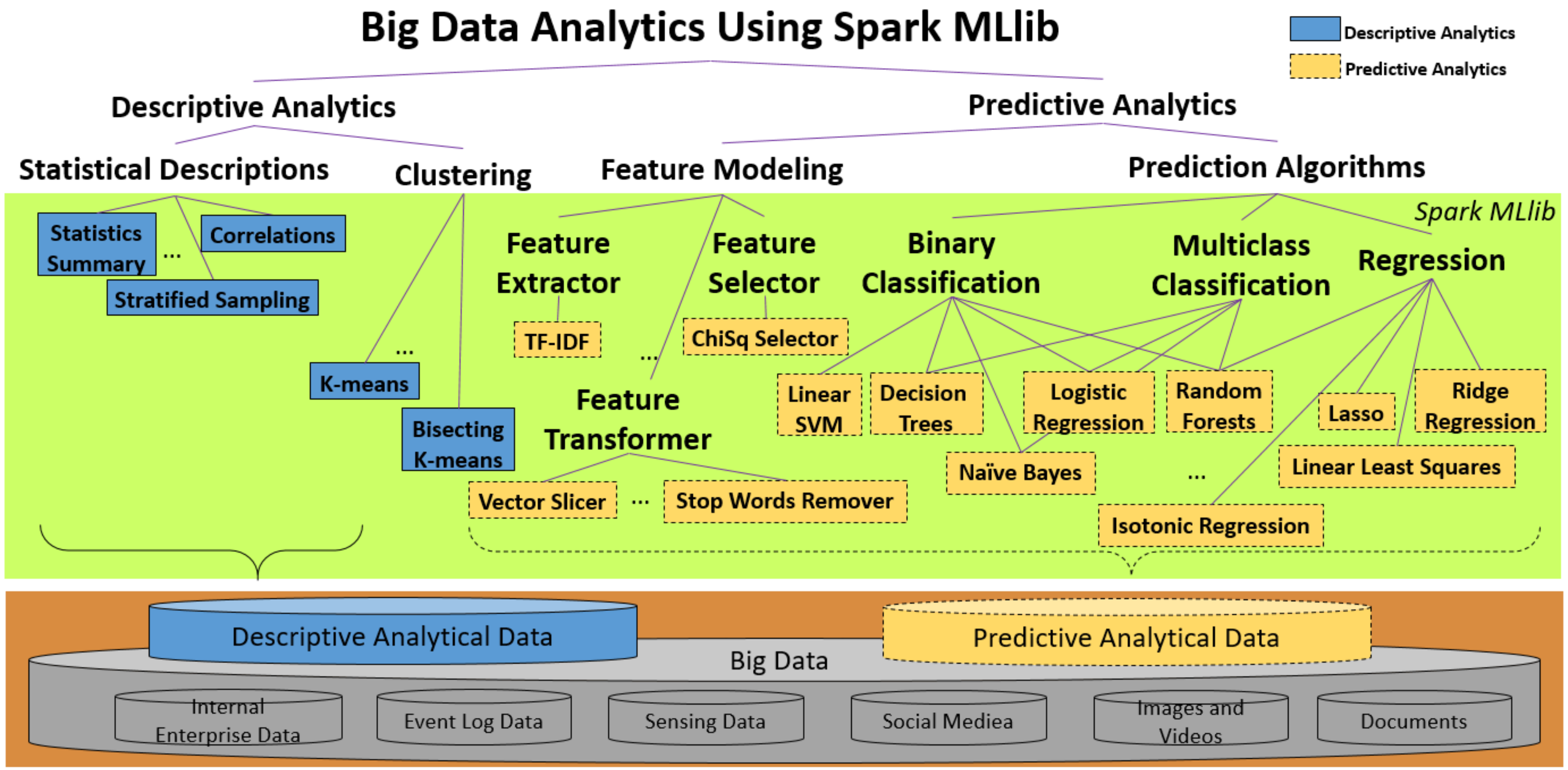
```
svmModel = svm.fit(newDF)
```

- **Pipeline:** A kind of estimator that orchestrates a series of transformers and estimators into a single model

```
pipeline = Pipeline(stages=[vec, svm]) # combine vectorization and classifier
pipelineModel = pipeline.fit(trainData) # model training
preds = pipelineModel.transform(testData) # model evaluation
```

SPARK ML IMPLEMENTATION

Taxonomy

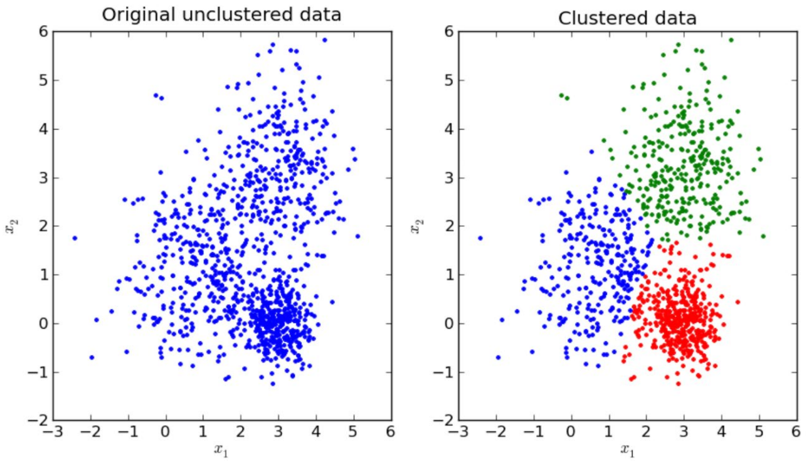


[16]

K-MEANS CLUSTERING

Clustering Example

- Partition-based clustering
- Clusters are associated with respective centroids
- Number of clusters must be known



[3] Tan et al.

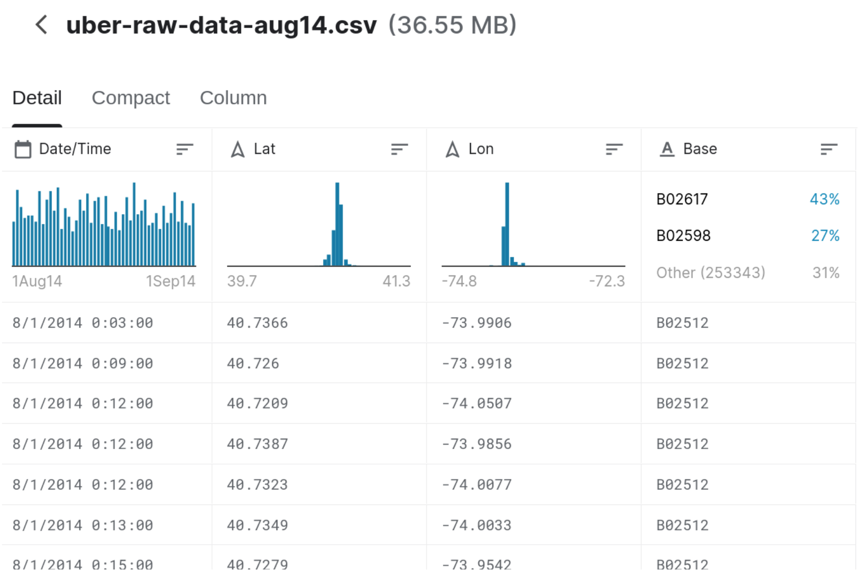
Select K points as the initial centroids.
repeat
    Form K clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change

JÜLICH Forschungszentrum

K-MEANS: UBER PICKUP DATA

Clustering Example

- Dataset: Uber Pickups in the CSV format
- Problem: Cluster the dense pickup points
- Uber trip data of August 2014
- Attributes: Lat, Lon, Date and Time, Base (TLC)



Date/Time	Lat	Lon	Base
8/1/2014 0:03:00	40.7366	-73.9906	B02512
8/1/2014 0:09:00	40.726	-73.9918	B02512
8/1/2014 0:12:00	40.7209	-74.0507	B02512
8/1/2014 0:12:00	40.7387	-73.9856	B02512
8/1/2014 0:12:00	40.7323	-74.0077	B02512
8/1/2014 0:13:00	40.7349	-74.0033	B02512
8/1/2014 0:15:00	40.7279	-73.9542	B02512

[2] Kaggle-Uber

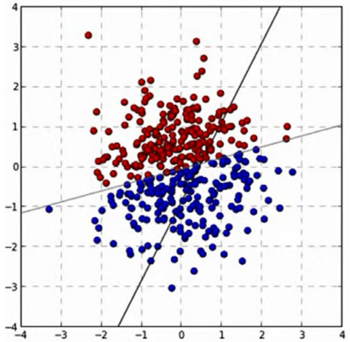
Example

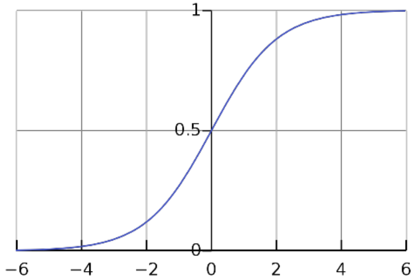
Demonstration: Switch to Notebook “KMeansExample.ipynb”

LOGISTIC REGRESSION

Classification Example

- Supervised machine learning algorithm
- Classification algorithm to deal with categorical response
- Predict binomial outcomes between 0 and 1
- Predictions are generated in the form of probabilities
- Uses Sigmoid function (a.k.a Logistic Function)





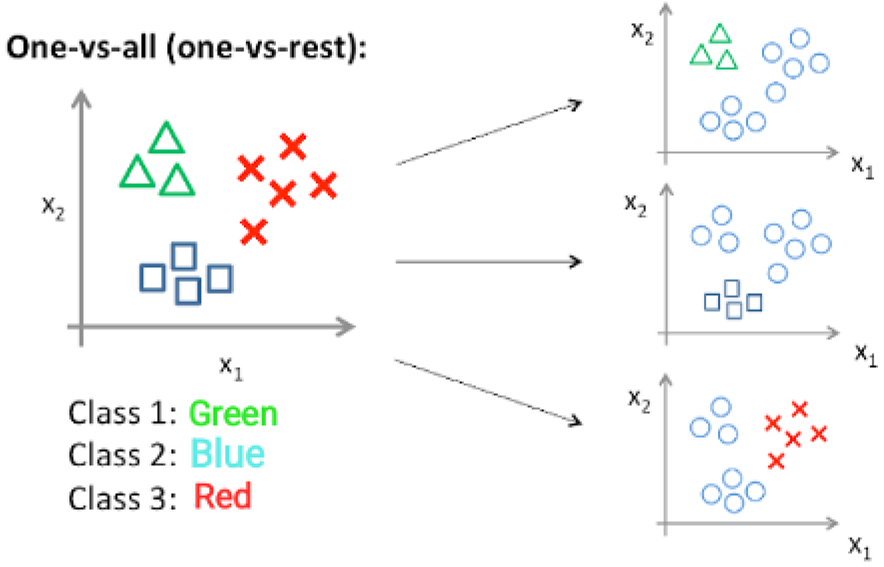
Example

Demonstration: Open “LogisticRegressionExample.ipynb”

ONEVSREST CLASSIFICATION

Multi-class classification

- Multiple class labels dataset
- Resolve multi-class as binary-class problem
- Apply N-binary classifiers for N-classes
- Example: The shape is triangle, square or cross



[1] Jatin Nanda

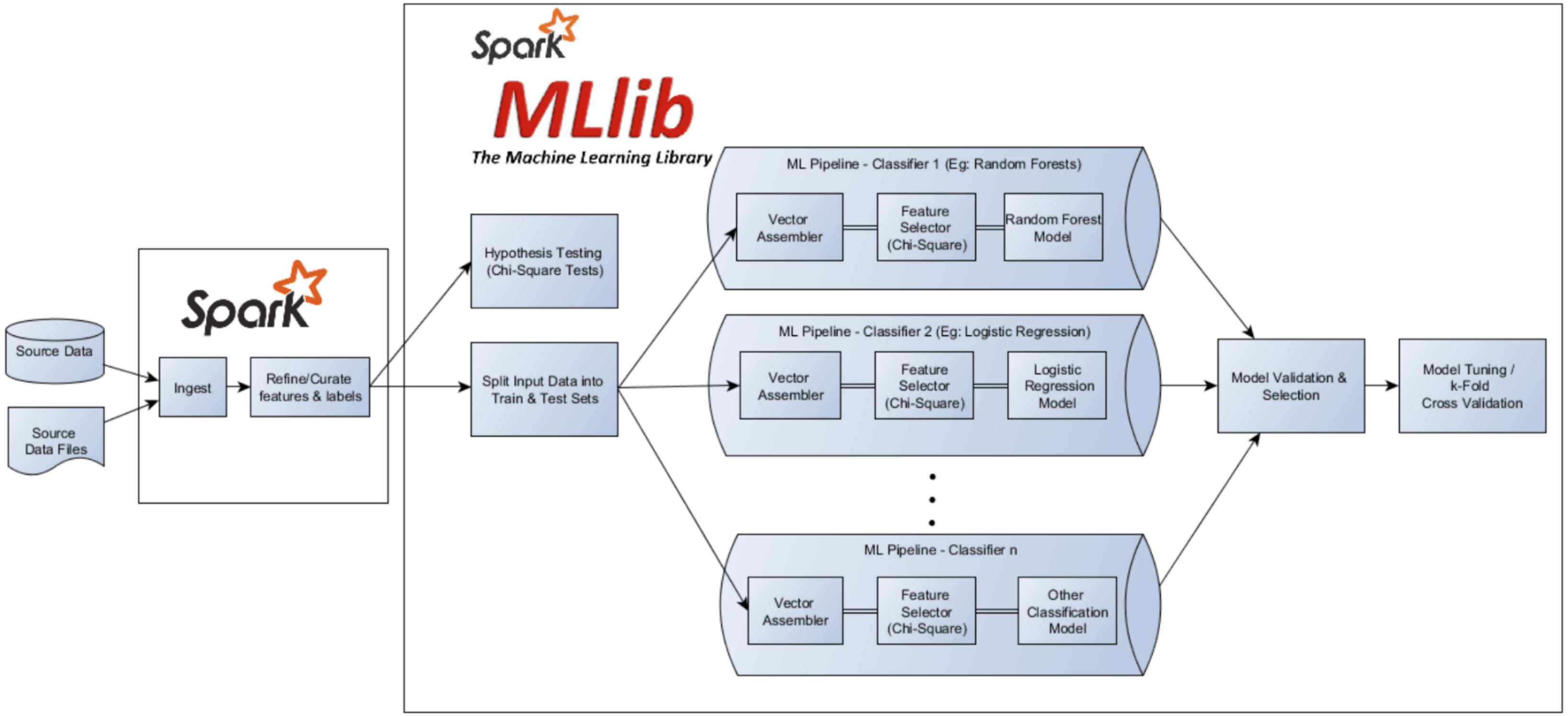
Example

Demonstration: Open "OneVsRestExample.ipynb"

ML PIPELINE – LINEAR REGRESSION EXAMPLE

Refresher

<u>Pipeline</u> is a kind of estimator that orchestrates a series of transformers and estimators into a single model



[17]

Demonstration: Open Notebook “PipelineExample.ipynb”

DOWNLOAD THE NOTEBOOKS TO THE LOCAL FILESYSTEM

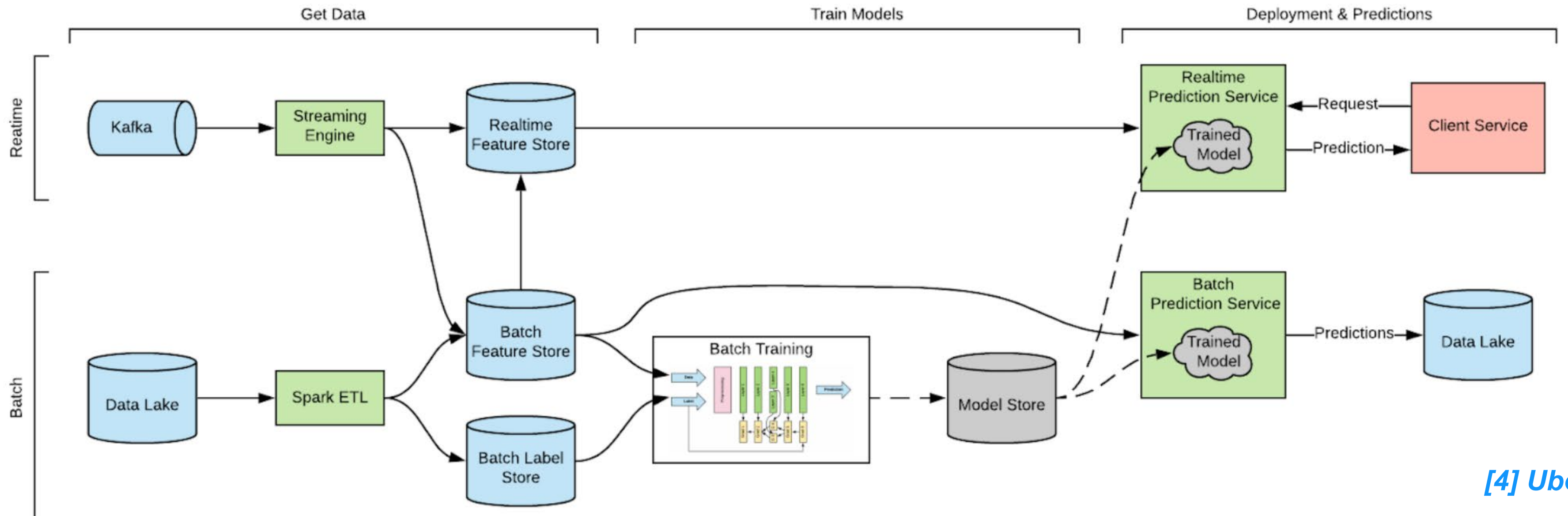
- Open the Jupyter Terminal
- CD to the /home/jovyuan/work directory ($> cd ~/work)
- Create a Tar archive ($> tar –czvf igarss21.tar.gz IGARSS2021/)
- Switch to the file browser view and download the created (igarss21.tar.gz) archive

JÜLICH Forschungszentrum

SPARK AND DEEP LEARNING

Introduction

- Combine ETL/ELT, model training and hyper-parameter tuning in one workflow
- Train TensorFlow / PyTorch models integrated with the Spark ecosystem



[4] Uber@Spark AI Summit 2020

Pre-Processing (Spark)

Distributed Training Keras/TF/PyTorch with Horovod)

Prediction DL and Spark

DISTRIBUTED DEEP LEARNING

Options

Distributed Inference

- Pandas UDF (User Defined Functions) and Apache Arrow

Distributed Training

1) Spark-TensorFlow-Distributor

```
from spark_tensorflow_distributor import MirroredStrategyRunner

runner = MirroredStrategyRunner(num_slots=1, local_mode=True, use_gpu=USE_GPU)
runner.run(train)
```

2) HorovodRunner (*only available for Databricks Runtime ML users*)

SPARK-TF-DISTRIBUTOR

Code Snapshot

```
from spark_tensorflow_distributor import MirroredStrategyRunner

# Adapted from https://www.tensorflow.org/tutorials/distribute/multi_worker_with_keras
def train():
    import tensorflow as tf
    import uuid

    BUFFER_SIZE = 10000
    BATCH_SIZE = 64

    def make_datasets():
        (mnist_images, mnist_labels), _ = \
            tf.keras.datasets.mnist.load_data(path=str(uuid.uuid4())+'mnist.npz')

        dataset = tf.data.Dataset.from_tensor_slices((
            tf.cast(mnist_images[..., tf.newaxis] / 255.0, tf.float32),
            tf.cast(mnist_labels, tf.int64))
        )
        dataset = dataset.repeat().shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
        return dataset
```

```
    def build_and_compile_cnn_model():
        model = tf.keras.Sequential([
            tf.keras.layers.Conv2D(32, 3, activation='relu', input_shape=(28, 28, 1)),
            tf.keras.layers.MaxPooling2D(),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(64, activation='relu'),
            tf.keras.layers.Dense(10, activation='softmax'),
        ])
        model.compile(
            loss=tf.keras.losses.sparse_categorical_crossentropy,
            optimizer=tf.keras.optimizers.SGD(learning_rate=0.001),
            metrics=['accuracy'],
        )
        return model

    train_datasets = make_datasets()
    options = tf.data.Options()
    options.experimental_distribute.auto_shard_policy = tf.data.experimental.AutoShardPolicy.DATA
    train_datasets = train_datasets.with_options(options)
    multi_worker_model = build_and_compile_cnn_model()
    multi_worker_model.fit(x=train_datasets, epochs=3, steps_per_epoch=5)

MirroredStrategyRunner(num_slots=8).run(train)
```

[13] SparkTFDistributor Code Repository

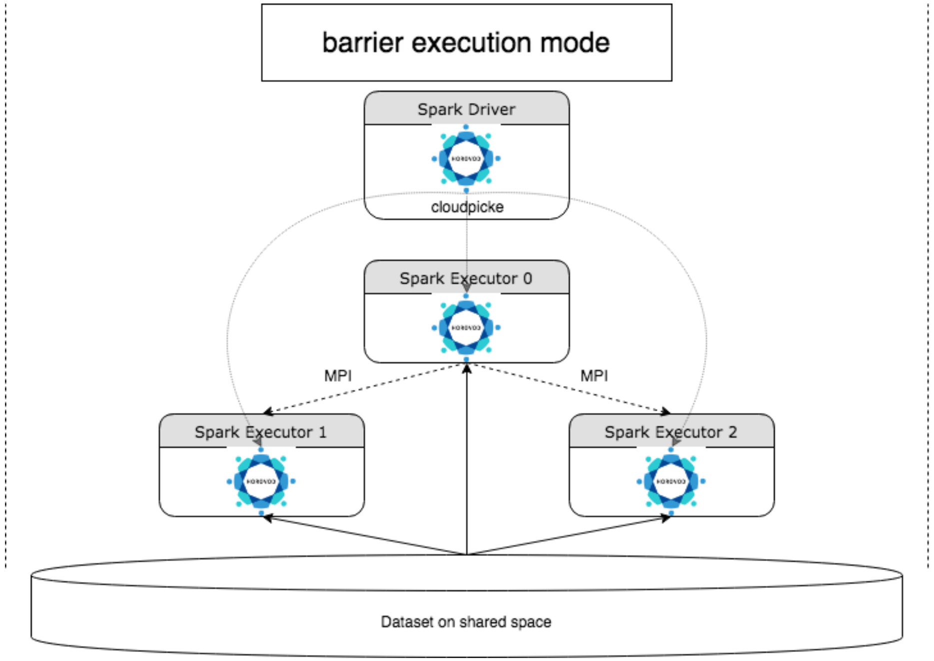
HOROVOD RUNNER

Distributed Training

- A generic API to manage distributed DL workloads
- Implemented through Spark's barrier execution mode scheduling (to support the MPI execution model)

Development workflow

1) Write single node DL code (e.g. TF/Keras)
2) Horovod-ify your code
3) Invoke HorovodRunner <hr.run(hvd_tr,..)>




```
hr = HorovodRunner(np=2)

def train():
  import tensorflow as tf
  hvd.init()

hr.run(train)
```

[10] Databricks

JÜLICH Forschungszentrum

CONCLUSIONS

- Big data analytics frameworks such as Apache Spark allows end-to-end ML/DL pipelines
- A viable direction for remote sensing and image analysis applications where whole processing workflow runs HPC and HTC simultaneously
- Harness public clouds (e.g. Amazon or Google) that provides stable deployments; integrated with state-of-the-art data analysis and DL frameworks (e.g. TF or PyTorch)

THANKS FOR LISTENING

REFERENCES

[1] Jatin Nanda: https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj4/html/jnanda3/index.html

[2] Kaggle-Uber: https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city?select=uber-raw-data-aug14.csv

[3] Tan, Pan-Ning and Steinbach, Introduction to Data Mining

[4] https://databricks.com/session_na20/end-to-end-deep-learning-with-horovod-on-apache-spark

[5] H. Lan et al. Spark Sensing: A Cloud Computing Framework to Unfold Processing Efficiencies for Large and Multiscale Remotely Sensed Data, with Examples on Landsat8 and MODIS Data: https://doi.org/10.1155/2018/2075057

[6] J.Haut et al. Cloud Deep Networks for Hyperspectral Image Analysis: https://doi.org/10.1109/TGRS.2019.2929731

[7] Using Apache Spark for Data Processing and Analysis: https://www.alibabacloud.com/blog/using-apache-spark-for-data-processing-and-analysis_596428

[8] DB Tsai, Netflix's Recommendation ML Pipeline Using Apache Spark. https://databricks.com/session/netflixs-recommendation-ml-pipeline-using-apache-spark

[9] Damji, Wenig, Das and Lee, Learning Spark 2nd Edition (O'Reilly Books)

[10] Databricks documentation: HorovodRunner: distributed deep learning with Horovod

REFERENCES

[10] Databricks documentation: HorovodRunner: distributed deep learning with Horovod

[11] M. Zahara et al. Apache Spark: A Unified Engine for Big Data Processing. DOI: 10.1145/2934664

[12] Introducing Pandas UDF for PySpark. https://databricks.com/blog/2017/10/30/introducing-vectorized-udfs-for-pyspark.html

[13] Spark-Tensorflow-Distributor: https://github.com/tensorflow/ecosystem/tree/master/spark/spark-tensorflow-distributor

[14] M. Riedel: High-Performance Computing, Fall Semester 2017

[15] B. Hagemeier, HDF Cloud – Helmholtz Data Federation Cloud Resources. Source: 10.17815/jslrf-5-173

[16] G. Park, IRIS: A Conceptual Modeling Framework for Big Data Analytics. https://sites.google.com/site/irisforbigdata/3-supporting-tool1/machine-learning-lib-spark

[17] Machine Learning Workflow on Qubole. https://www.qubole.com/developers/spark-getting-started-guide/workflow/