Introduction to Apache Spark: running Spark on CERN resources

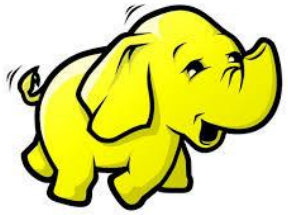
Luca Canali
CERN IT, Data Analytics and Spark Service

Start Small

- You can develop and run Spark on your laptop or desktop or VM
- Many way to install and run Spark
  - pip install pyspark
  - download from https://spark.apache.org/downloads.html
  - docker run -it apache/spark-py /opt/spark/bin/pyspark
  - docker run -it apache/spark /opt/spark/bin/spark-shell # scala

Apache Spark Clusters at CERN

- Spark running on clusters:
  - YARN/Hadoop -> established
  - Spark on Kubernetes -> cloud-like use cases



NXCALS: accelerator logging	Hadoop - YARN - 42 nodes (Cores – 1.8k, Mem - 18 TB, Storage – 14 PB)
ANALYTIX: General Purpose	Hadoop - YARN, 58 nodes (Cores – 2.6k, Mem – 30 TB, Storage – 20 PB)
Cloud containers	Kubernetes on CERN OpenStack Private Cloud Cores - 270, Mem – 2 TB Storage: remote HDFS or custom storage (CERN EOS, for physics data, S3 on Ceph also available). Note: GPU resources available.

CERN

Getting Started with the Spark Services at CERN

Documentation, Spark on Hadoop:

- https://hadoop-user-guide.web.cern.ch/spark/Using_Spark_on_Hadoop/#getting-started

Request Access to Hadoop clusters:

- https://cern.service-now.com/service-portal?id=sc_cat_item&name=access-cluster-hadoop&se=Hadoop-Service

Available Spark Clients at CERN

- SWAN
  - Hosted Jupyter notebook service: http://swan.cern.ch
- Ad-hoc edge nodes
  - **`ssh it-hadoop-client.cern.ch`**
  - Managed Spark Client environment
- CERN Hadoop container image
  - https://hadoop-user-guide.web.cern.ch/getstart/client_docker
- Run from lxplus.cern.ch
  - https://hadoop-user-guide.web.cern.ch/getstart/client_cvmfs

Using it-hadoop-client

```
ssh it-hadoop-client

# configuration for Hadoop and Spark
source hadoop-setconf.sh hadoop-analytix

# run PySpark
pyspark
```

Creating Spark Applications

- There are several ways to create a Spark Application:
  - REPL: pyspark (python), spark-shell (Scala)
  - Self-contained applications (aka using Spark as a library)
  - spark-submit (batch mode)

Spark REPL / Shell

- REPL stands for “read-eval-print loop”
  - Best for interactivity, development and debugging
  - Spark comes with a Python (pyspark) and a Scala (spark-shell) shell where the SparkSession is already set up and available as the spark object

Using Spark as a Library

```
from pyspark.sql import SparkSession
spark = (SparkSession
                .builder
                .appName("training")
                .master("local[*]")
                .getOrCreate())

df=spark.sql("select 'hello, world!' as msg")
df.show()
```

```
$ pip install pyspark
$ python my_code.py
+-------------+
|          msg|
+-------------+
|hello, world!|
+-------------+
```

Using spark-submit

- Uniform interface to all the cluster manager
- Best for production (batch - ETL) use cases.

```
bin/spark-submit --master yarn \
--num-executors 2 --executor-memory 8g \
examples/src/main/python/pi.py 1000
```

Spark Configuration

Three main configuration files in `$SPARK_HOME/conf` or `$SPARK_CONF_DIR`

1) Spark Properties: control application parameters
   - spark-defaults.conf
2) Environment variables: per-machine settings
   - spark-env.sh
3) Logging: through log4j2
   - log4j2.properties
   - was log4j.properties for older Spark versions

Spark Configuration

Spark properties

1) set properties when creating a SparkConf object in your code

```
>>> from pyspark.sql import SparkSession
>>> spark = (SparkSession.builder
...     .appName("training")
...     .config("spark.executor.cores", "2")
...     .getOrCreate())
```

Spark Configuration

Spark properties

2) Set as runtime command line options

```
pyspark --master yarn \
--app-name ‘training’ \
--conf spark.executor.cores=2
```

Spark Configuration

Spark properties

3) Set in the configuration directory, the spark-defaults.conf file

```
$ head spark-defaults.conf

spark.master                    yarn
spark.executor.cores            4
spark.executor.memory           8g
spark.executor.instances        2
```

CERN

Logging and Environment

- More advanced configurations
  - log4j2.properties file in the configuration folder
  - environment variables are sourced from spark-env.sh

CERN Hadoop Configuration

- CERN configuration file for Spark. Run using:

```
source hadoop-setconf.sh hadoop-analytix
```

- Already setup in IT managed clients, it's in $HADOOP_CONF_DIR
  - default HDFS namespace
    - <property>
    - <name>fs.defaultFS</name>
    - <value>hdfs://analytix/</value>
    - </property>
  - Hadoop specific settings
  - YARN configuration

Dependencies - Java

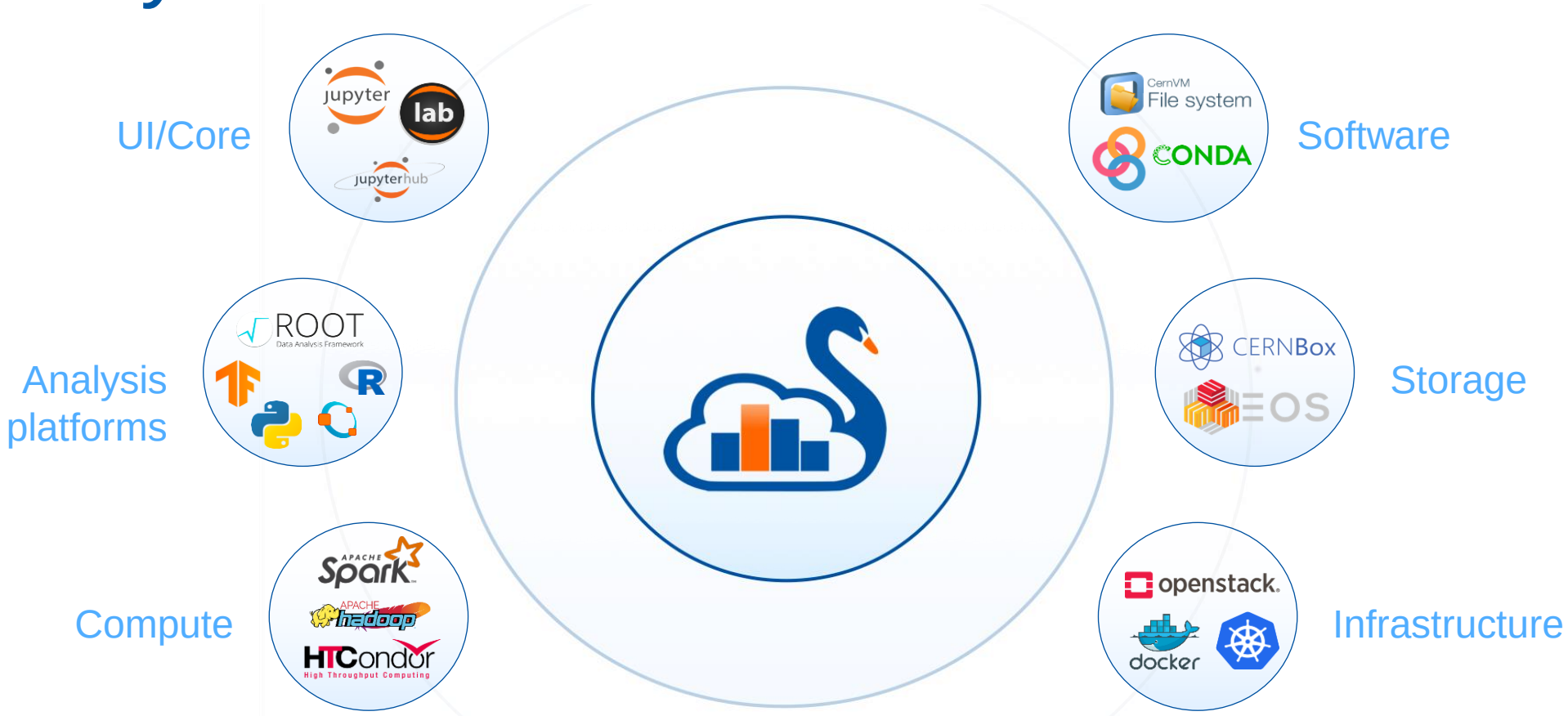
- Extend Spark with custom jar files
  - --jars <list of jar files>
  - The jars will be copied to the executors and added to their classpath
- Ask Spark to download jars from a repository
  - --packages <list of Maven Central coordinates>
  - Will download the jars and dependencies in the local cache, jars will be copied to executors and added to their classpath

Dependencies - Python

- Command line option
  - --py-files <list of .py files or .zip bundles>
  - Allows to ship Python packages to executors
  - When using SWAN, there is a configuration option Include PropagateUserPythonModules options that adds packages installed with pip install --user
- More advanced uses cases
  - See Python package management in the documentation https://spark.apache.org/docs/latest/api/python/user_guide/python_packaging.html

CERN SWAN

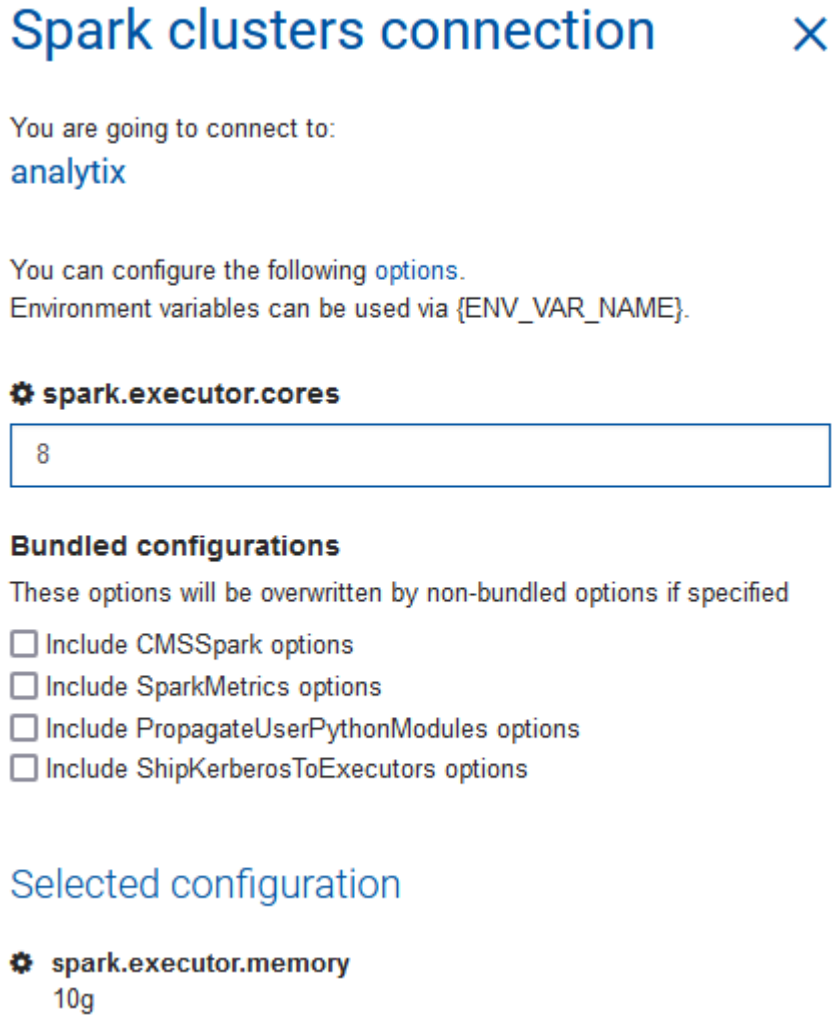
- Analytics and ML Platform



Spark Configuration with SWAN

Spark configuration via a GUI tool

- Customize specific parameters
- Pre-defined parameter bundles also available



Key Learning Points

- You can run Apache Spark at scale on CERN resources: Hadoop and Cloud
- Spark has a large set of configuration options
  - You can use Spark in multiple modes: from shell, as a library, batch, with Python, with Scala,…
- The CERN SWAN web notebooks service is integrated with Spark/Hadoop clusters
  - Reduces complexity of running Spark
  - Best option to start exploring Spark at CERN

Tutorials

- See notebooks and associated videos