Spark Summit 2016:
CONNECTING PYTHON TO THE SPARK ECOSYSTEM

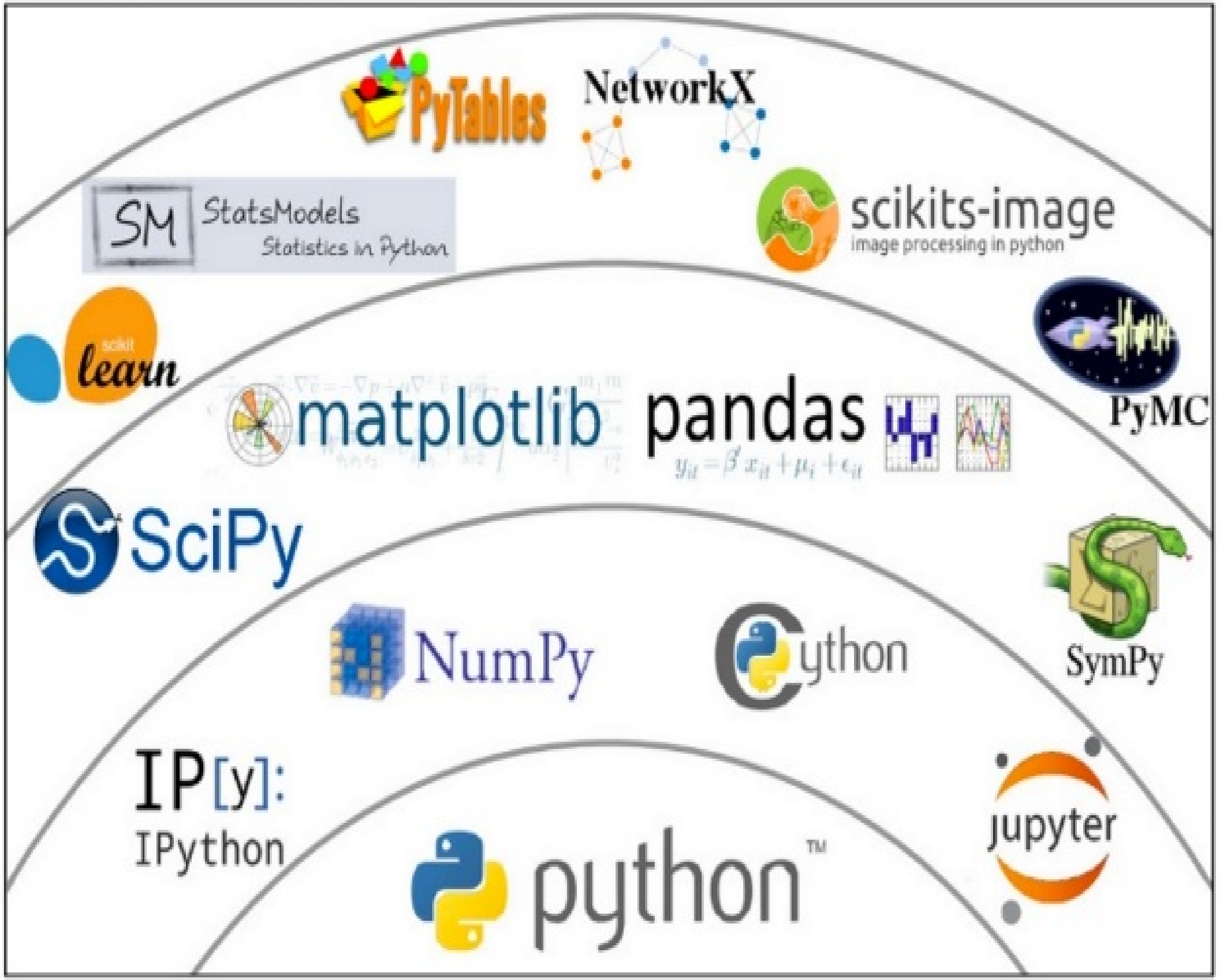
Daniel Rodriguez
Software developer/data scientist
Continuum Analytics
Twitter: @danielfrg
Github: github.com/danielfrg

ANACONDA®

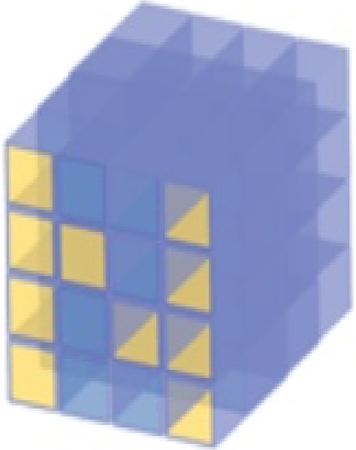
Content

- PyData and Spark
- Python (PyData libs) package management
- Python package management in a cluster
    - Multiple options
- Some usage of python (single node) libraries in a cluster
- Future

PyData ecosystem

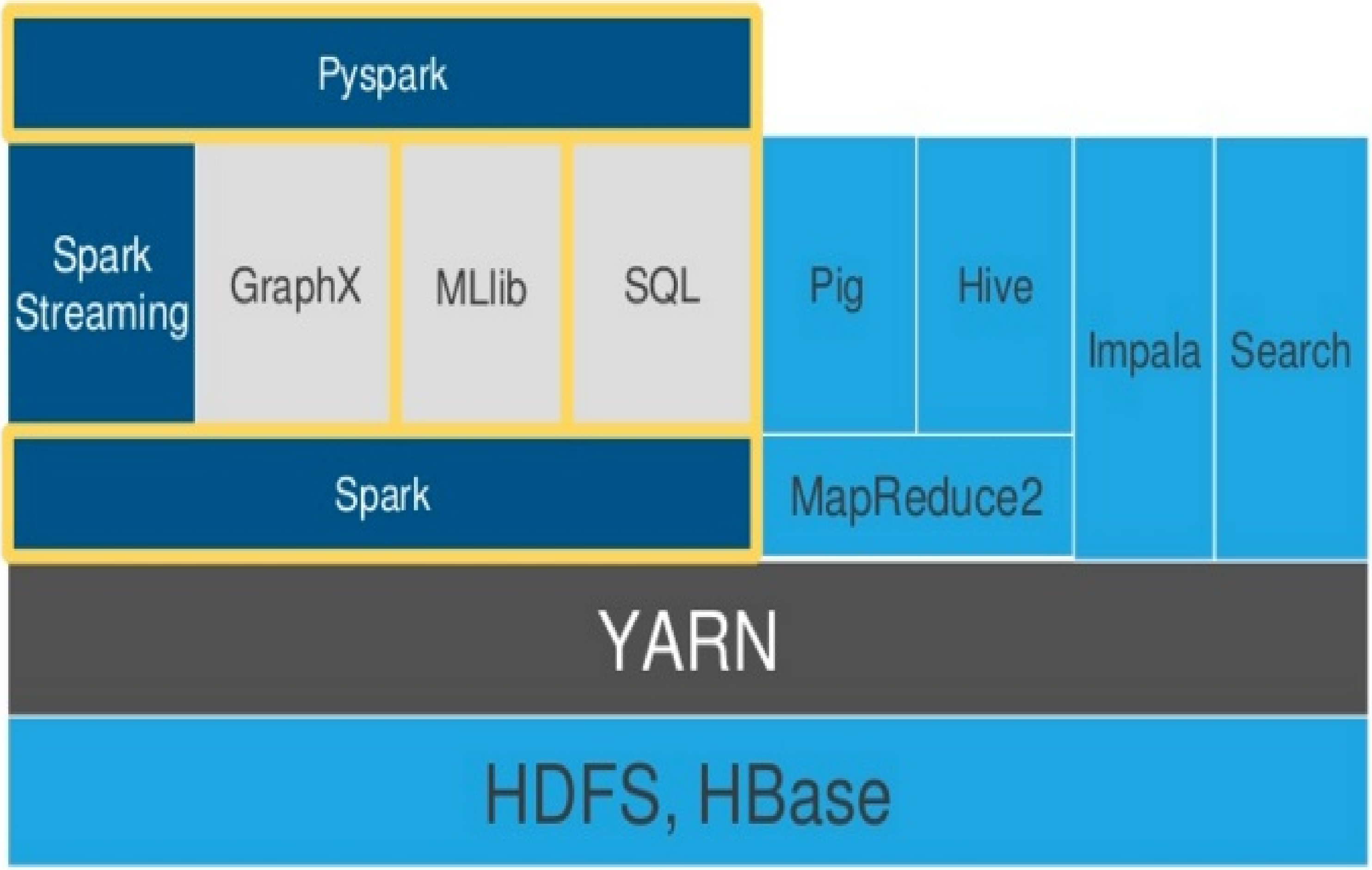


PyData ecosystem: Numpy



- Numpy as the core for the ecosystem:
  - Powerful N-dimensional array object
  - Broadcasting functions
  - Tools for integrating **C/C++ and Fortran code**
  - Linear algebra, Fourier transform, and random number capabilities
  - Single node
  - Python only

Spark Ecosystem



Pyspark

Spark Streaming

GraphX

MLlib

SQL

Pig

Hive

Impala

Search

Spark

MapReduce2

YARN

HDFS, HBase

Third party: spark-packages.org

Spark ecosystem

- Fast and general engine for large-scale data processing
- High level libraries:
  - **Dataframes**, MLlib, SQL, Graphs
  - Similar as PyData
- Distributed by design: RDD as core
- JVM with multi language interfaces: Scala, Java, Python, R

PyData vs Spark

- PyData for scientists - maybe they also use MPI
- Spark for a Big Data (after MapReduce)
- Both relatively fine and happy
- Scientist have to scale
- Big data people have to do data science, statistics, ML
- Data scientists now do both

PySpark issues

- PySpark source is not very pythonic (imo)
- Packages
- Performance:
  - Serialization and pipes
  - RDD of pickle objects
  - Py4J to communicate
  - Driver and each worker: java <-> python

PyData package management

- Numpy and other have C/C++ and Fortran code
- Compiling. **Windows**

Conda
github.com/conda/conda

+

ANACONDA®

Conda-forge: Community powered packages

Package management in a cluster

- Lot (100s to 1000s) nodes
- Not really a new problem in DevOps or CM

SALTSTACK

CHEF

Search on github for available conda modules

Cloudera Manager and Anaconda Parcel

- Most popular Hadoop distribution
- Anaconda parcel for CDH *:
  - Static python distribution
  - Super easy installation if you have CDH

*: Joint work between Cloudera and Continuum
More info:
- https://docs.continuum.io/anaconda/cloudera
- http://blog.cloudera.com/blog/2016/02/making-python-on-apache-hadoop-easier-with-anaconda-and-cdh

I don't use Cloudera

HORTONWORKS®

Want to make this happen?
Call me maybe?

You manage your own Hadoop cluster?

For real?

Anaconda for cluster management

- Agnostic: Cloud, on premise, Air-gap
- Dynamic package and environment management
- Based on salt
- Extra plugins: Jupyter Notebook and more

SALTSTACK

More info: https://docs.continuum.io/anaconda-cluster/index

Leverage Spark and YARN

- New ways to deploy environments:
- Sparkonda: https://github.com/moutai/sparkonda

Leverage Spark: Sparkonda

```
conda create -n sparkonda-test-env python=2.7 pip pandas scikit-learn numpy numba
source activate sparkonda-test-env
pip install sparkonda


sc.addPyFile('path/to/sparkonda_utils.py')


skon.CONDA_ENV_NAME = 'sparkonda-test-env'
skon.CONDA_ENV_LOCATION = ''.join([home_dir,'/miniconda/envs/',skon.CONDA_ENV_NAME])
skon.SC_NUM_EXECUTORS = 2
skon.SC_NUM_CORE_PER_EXECUTOR = 2

skon.pack_conda_env()
skon.distribute_conda_env(sc)
skon.list_cwd_files(sc)
skon.install_conda_env(sc)
skon.set_workers_python_interpreter(sc)


def check_pandas(x): import pandas as pd; return [pd.__version__]
skon.prun(sc, check_pandas, include_broadcast_vars=False)
```

Leverage Spark and YARN

- Beta
- Conda + Spark: http://quasiben.github.io/blog/2016/4/15/conda-spark/
- Security: Kerberos

PySpark

```
lines = sc.textFile("data.txt")
lineLengths = lines.map(lambda s: len(s))
totalLength = lineLengths.reduce(lambda a, b: a + b)
```

PySpark: NLTK

```
def word_tokenize(x):
    import nltk
    return nltk.word_tokenize(x)

def pos_tag(x):
    import nltk
    return nltk.pos_tag([x])

words = data.flatMap(word_tokenize)
pos_word = words.map(pos_tag)
```

```
pos_word.take(5)

[[('Address', 'NN')],
[('on', 'IN')],
[('the', 'DT')],
[('State', 'NNP')],
[('of', 'IN')]]
```

PySpark: Image processing GPUs

Jupyter Notebook:
https://gist.github.com/danielfrg/0afee63072793e9e9d6ebaee27865a4a

PySpark: Scikit-learn: spark-sklearn

```
from sklearn import grid_search, datasets
from sklearn.ensemble import RandomForestClassifier
from sklearn.grid_search import GridSearchCV
digits = datasets.load_digits()
X, y = digits.data, digits.target
param_grid = {"max_depth": [3, None],
              "max_features": [1, 3, 10],
              "min_samples_split": [1, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"],
              "n_estimators": [10, 20, 40, 80]}
gs = grid_search.GridSearchCV(RandomForestClassifier(),
param_grid=param_grid)
gs.fit(X, y)
```

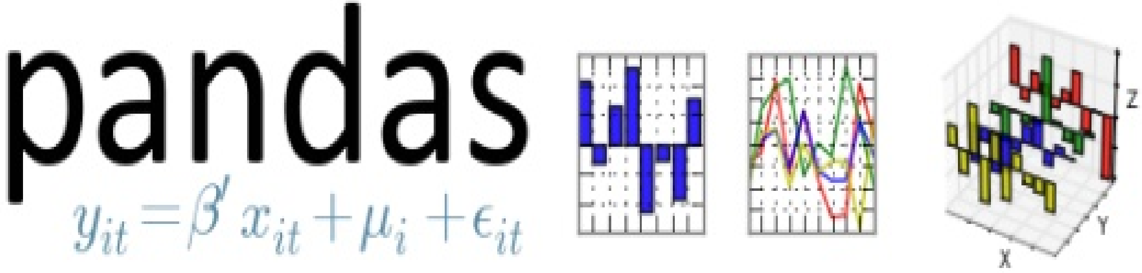
More info: https://github.com/databricks/spark-sklearn

PySpark: Scikit-learn: spark-sklearn

```
from sklearn import grid_search, datasets
from sklearn.ensemble import RandomForestClassifier
from spark_sklearn import GridSearchCV
digits = datasets.load_digits()
X, y = digits.data, digits.target
param_grid = {"max_depth": [3, None],
              "max_features": [1, 3, 10],
              "min_samples_split": [1, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"],
              "n_estimators": [10, 20, 40, 80]}
gs = grid_search.GridSearchCV(RandomForestClassifier(),
param_grid=param_grid)
gs.fit(X, y)
```

More info: https://github.com/databricks/spark-sklearn

PySpark: Dataframes



Spark Dataframes

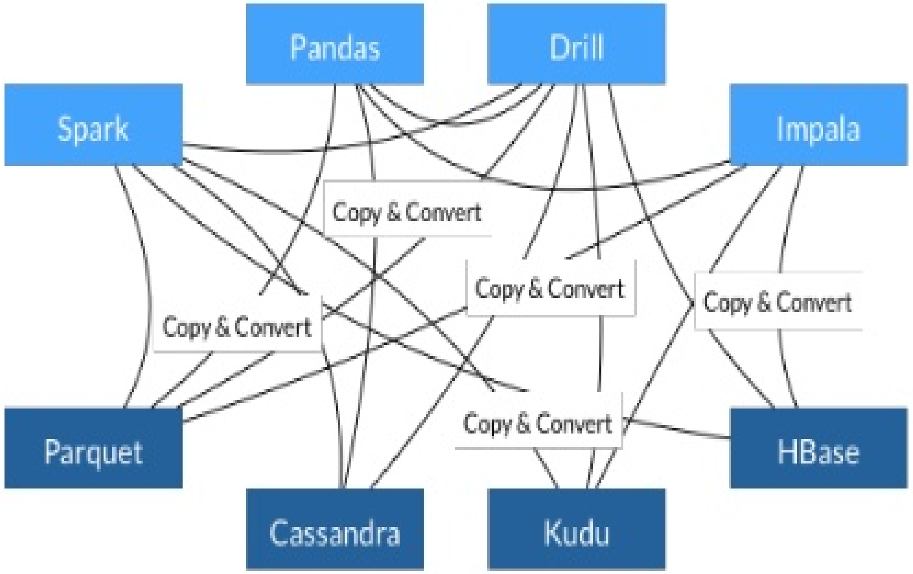
Performance happy

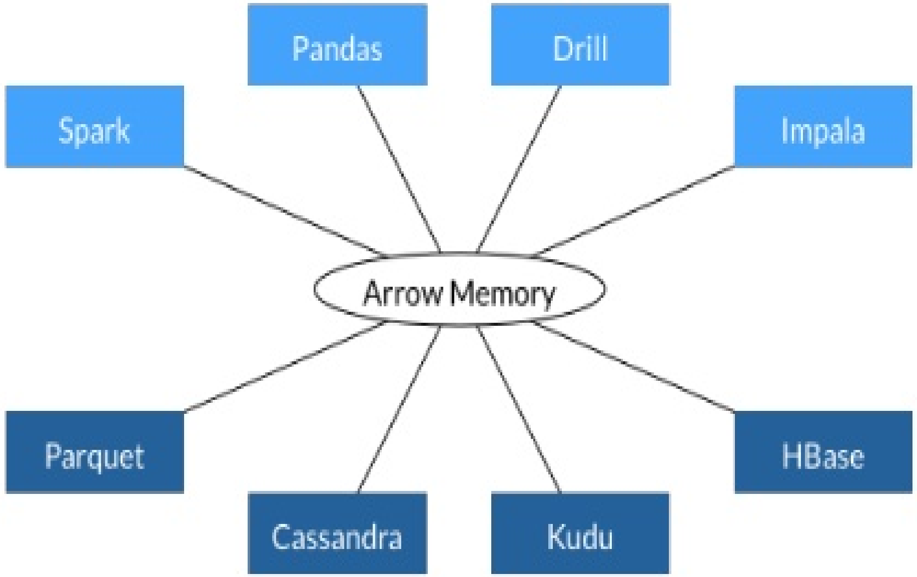
Future / Alternatives

- Apache Arrow
- Tensorflow
- Dask

Apache Arrow

CONTINUUM ANALYTICS





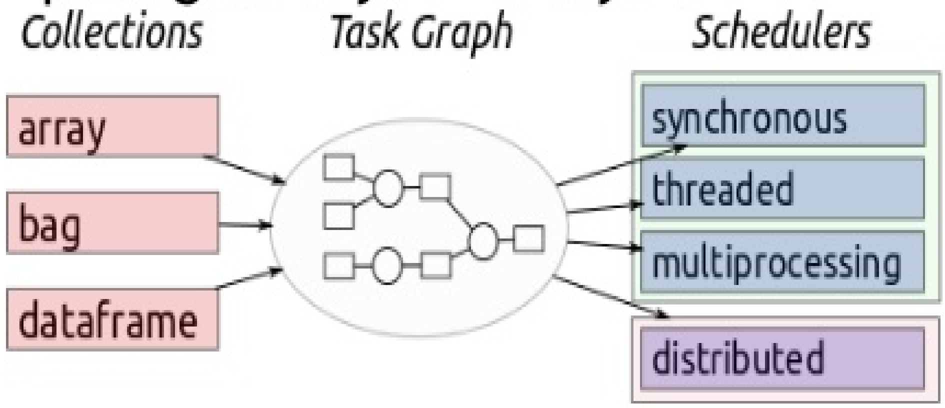
More info:

- https://github.com/databricks/spark-sklearn
- https://blog.cloudera.com/blog/2016/02/introducing-apache-arrow-a-fast-interoperable-in-memory-columnar-data-structure-standard/

TensorFlow

Dask

Python parallel computing library for analytics



```
import pandas as pd
df = pd.read_csv('2015-01-01.csv')
df.groupby(df.user_id).value.mean()
```

```
import dask.dataframe as dd
df = dd.read_csv('2015-*-*.csv')
df.groupby(df.user_id).value.mean().compute()
```

```
pip install dec2 OR conda install -c conda-forge dec2

dec2 up --keyname YOUR-AWS-KEY-NAME
        --keypair ~/.ssh/YOUR-AWS-KEY-FILE.pem
        --count 9   # Provision nine nodes
        --nprocs 8  # Use eight separate worker processes per node
```

Spark Summit 2016:
CONNECTING PYTHON TO THE SPARK ECOSYSTEM

Questions?

Daniel Rodriguez
Software developer/data scientist
Continuum Analytics
Twitter: @danielfrg
Github: github.com/danielfrg

ANACONDA