Python Dask And Apache Spark.

Christian Voß

NUC Meeting

January 21, 2021

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

Motivation

Why Investigate Python Dask and Apache Spark

- IT point of view
  - Investigate new methods in data analytics
  - Need for a data analysis plattform within IT
- Observations on the NAF
  - Jupyer Notebooks require a lot of memory
  - Python tools lack good memory management
  - Request from Belle II to offer Support
  - Offered insight to Apache Spark
  - Request to investigate Python Dask

Fundamental Characteristics

Both tools are not classical CPU schedulers

- Both based on scheduler and workers in place before begin of analysis
- Allow for interactive analyses by keeping the data in memory
- Allow easy access through Jupyter

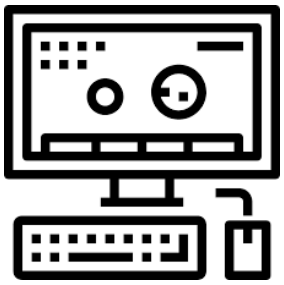
Apache Spark – In production in IT since 2018

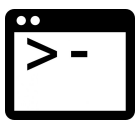
- Written in JAVA with bindings for e.g. Python
- Can be used with classic submit scripts
- Native driver for ROOT files (basis for CERN Swan)

Python Dask – In use by the Europan XFEL

- Written in Python based on numpy/pandas libraries
- Access directly through Python
- Use Scikit-HEP's uproot package to interface ROOT files

How Does it Work?





Select Data: `/.../*.root`
Select Job: `MySelection`

pass on

Spark Master

- Configures Tasks
- Selects workers

Spark Workers

Worker 1

`Run-1.1.root`

Event 1 - 100

Worker 2

`Run-1.1.root`

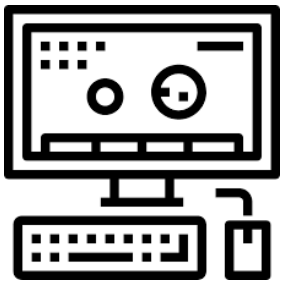
Event 101 - 200

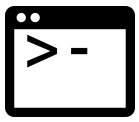
Worker 3

`Run-1.2.root`

Event 1 - 100

How Does it Work?





Select Data: `/.../*.root`
Select Job: `MySelection`

pass on

Spark Master

- Configures Tasks
- Selects workers

Spark Workers

Worker 1

`Run-1.1.root`

Event 1 - 100

Worker 2

`Run-1.1.root`

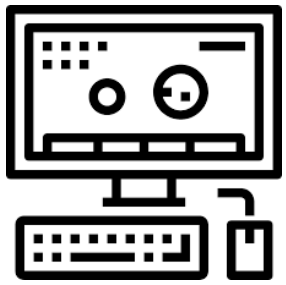
Event 101 - 200

Worker 3

`Run-1.2.root`

Event 1 - 100

How Does it Work?



How to connect?

Select Data: `/.../*.root`
Select Job: `MySelection`

pass on

Spark Master

- Configures Tasks
- Selects workers

Spark Workers

Worker 1

`Run-1.1.root`

Event 1 - 100

Worker 2

`Run-1.1.root`

Event 101 - 200

Worker 3

`Run-1.2.root`

Event 1 - 100

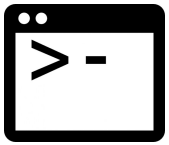
How to Connect to these Tools

Native bindings among others for Python

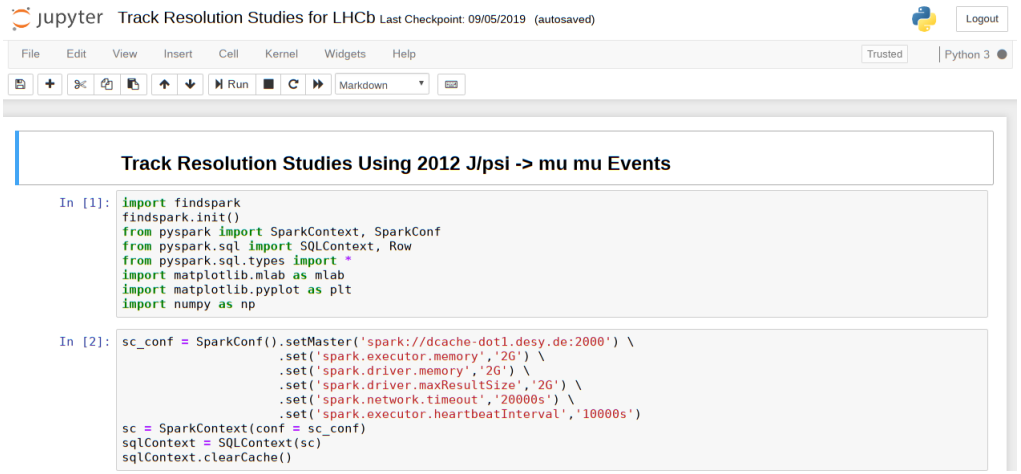
1. Command line/scripts (for Spark direct submission possible)

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.0
      /_/

Using Python version 3.6.7 (default, Dec  5 2018 15:02:05)
SparkSession available as 'spark'.
>>> help(spark)
```



2. Jupyter Notebooks



Implications for the NAF

Embracing Spark/Dask is a Major Paradigm Change

- Reserve resources befor submitting payload – no classical queueing
- Schedule memory by caching data in memory – decreased efficiency
- Blockage of resources by carefree users
- Improvements to data access patterns

Advantages for the Users

- Very fast and interactive analysis plattform
- Descreased input file juggling
- Load data once (slow), then query for information (fast)
- Depending on setup a more tedious deployment

Status of Dask/Spark the NAF

Deployment

- Different Setup on NAF compared to CERN Swan with Spark
- Currently static clusters on decommissioned hardware in IT use
- Deployment fully container-based (Singularity) $\rightarrow$ compatible to the NAF
- Deployment through htcondor jobs similar to Jupyter-Notebooks
- Additional Python dependencies through user build containers

Current Status

- Presented Spark to Belle II and ATLAS
- Got feedback and test users from Belle II incl. request for Dask
- Deployed a Dask cluster in parallel to Spark
- Granted access to Belle II users for evaluation
- As IT user tested deployment and analysis on NAF