

High performance Jupyter: Faster workloads with Dask and RAPIDS

JupyterCon 2020



<https://github.com/rikturr/high-performance-jupyter>

Hi!

Aaron Richter, PhD



Senior Data Scientist @ Saturn Cloud
Organizer @ PyData Miami

> *I work to make data scientists faster and happier*

aaron@saturncloud.io
rikturr.com
[@rikturr](#)

Saturn Cloud

Bringing together the fastest hardware + OSS



- Pythonic parallelism
- Rapidly scale PyData

RAPIDS

- Multi-GPU computing
- The future of HPC



- Workflow orchestration
- Flow insight and mgmt



- Fast setup
- Enterprise secure

Data science with Jupyter



Data science with Jupyter

A screenshot of the JupyterLab web interface in a browser. The browser address bar shows "localhost:8888/lab". The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar with icons for file operations and execution. The main area displays a code editor for a file named "we_heart_pydata.ipynb". The code is written in Python 3 and consists of three cells. The first cell imports pandas and numpy, and reads a CSV file. The second cell uses numpy.where to create a new column 'ycol' based on the value of 'mycol', and then selects specific features from the dataframe. The third cell imports RandomForestClassifier from sklearn.ensemble and fits the model to the data. The status bar at the bottom shows "Python 3 | I...", "Saving comple...", "Mode: Comm...", "Ln 1, C...", and the filename "we_heart_pydata.i...".

```
[1]: import pandas as pd
import numpy as np

df = pd.read_csv('...')

[2]: df['ycol'] = np.where((df['mycol'] >= 42), 1, 0)

X = df[['feat1', 'feat2', 'feat3']]
Y = df['ycol']

[3]: from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, n_jobs=-1)
rf.fit(X, y)
```

High performance data science





DASK

RAPIDS

Dask

- *Parallel computing for Python people*
- Anaconda, ~2015
- Built in Python; Python API
- Mature, scientific computing communities
- Low-level task library
- High-level libraries for DataFrames, arrays, ML
- Integrates with PyData ecosystem
- Runs on laptop, scales to clusters

<https://dask.org/>



RAPIDS

- *GPU accelerated data science*
- NVIDIA, ~2018
- Built in C++(CUDA), Python; Python API
- Large dev team, support from NVIDIA
- Native DataFrames, arrays, ML, graph, streaming, spatial
- Integrates with PyData ecosystem
- *Scales to clusters with Dask integration*

<https://rapids.ai/>

RAPIDS

Single GPU
Data <16GB

100x faster than
CPU

RAPIDS

Numpy -> cuPy
Pandas -> cuDF
Scikit-learn -> cuML

RAPIDS



Dask Array [cuPy]
Dask DataFrame [cuDF]
Dask + cuML

1000s GPUs
Data TB++

100x faster +
"Big data"

Better
hardware

Why not
both??

Single machine
Data <200GB

Slow
computation



More
hardware



Dask Array [numpy]
Dask DataFrame [pandas]
Dask ML [scikit-learn]

1000s machines
Data TB++

Horizontal
scaling

Code time!



<https://github.com/rikturr/high-performance-jupyter>