Unlocking the True Value of Hadoop with Open Data Science

Kristopher Overholt
Solution Architect

Big Data Tech 2016
MinneAnalytics
June 7, 2016
Overview

• Overview of Open Data Science
• Python and the Big Data Ecosystem
• Visualizing Large Datasets with Python
• Parallel Computations with Python
• Example Parallel Workflows with Python
• Summary
Overview of Open Data Science
Modern Data Science Needs

**Interactivity**
- Interact with data
- Build high performance models
- Visualize results in context

**Collaboration**
- Iterate on analysis
- Share discoveries with team
- Interact with teams across the globe

**Integration**
- Work with distributed data systems
- Leverage existing libraries in new distributed frameworks: PySpark and SparkR
Open Data Science Team

Biz Analyst
- Spreadsheets
- Visualization
- Notebooks
- Analytic Development Environment

Data Scientist
- Hadoop / Spark
- Programming Languages
- Analytic Libraries
- IDE
- Notebooks
- Visualization

Developer
- Programming Languages
- Analytic Libraries
- IDE
- Notebooks
- Visualization

Data Engineer
- Database / Data Warehouse
- ETL

DevOps
- Database / Data Warehouse
- Middleware
- Programming Languages

Right technology and tools for the problem
Open Data Science is …

an inclusive movement

that makes open source tools of data science

— data, analytics, and computation —

easily work together as a connected ecosystem
Open Data Science Means....

- Availability
- Innovation
- Interoperability
- Transparency

for everyone on the data science team

Open Data Science is the foundation to modernization
Open Source Communities Create Powerful Technology for Data Science

### Machine Learning / Statistics
- pandas
- scikit-learn
- TensorFlow
- NumPy
- SciPy
- theano
- Lasagne

### Business Intelligence
- xlwings
- Blaze

### Distributed Systems
- Apache Hadoop
- Apache Spark
- Apache Hive
- DASK

### Scientific Computing / HPC
- Numba
- Cython
- PyTables

### Databases / Storage
- MySQL
- SQLite
- PostgreSQL
- HDFS
- MongoDB
- SQLAlchemy

---

[ANAconda] [LIGT] [AIRflow]
Python is the Common Language

Machine Learning / Statistics
- Scikit-learn
- TensorFlow
- Theano
- NumPy
- xarray
- SciPy

Business Intelligence
- pandas
- xlwings
- Blaze

Scientific Computing / HPC
- Numba
- Cython
- PyTables

Distributed Systems
- Hadoop
- Spark
- HIVE
- DASK

Databases / Storage
- HDFS
- MongoDB
- SQLAlchemy
- MySQL
- SQLite
- PostgreSQL
Not the Only One...

Machine Learning / Statistics

Business Intelligence

SQL

C

C++

Scientific Computing / HPC

Distributed Systems

Fortran

Databases / Storage

JavaScript
Python is also a great glue language

- Machine Learning / Statistics
- Business Intelligence
- Distributed Systems
- Databases / Storage
- Scientific Computing / HPC

Languages:
- SQL
- C
- C++
- Fortran
- JavaScript
- Python

Tools:
- Anaconda

Logo: ANACONDA®
Anaconda is the Open Data Science Platform Bringing Technology Together…

Machine Learning / Statistics
- Numba
- xlwings
- pandas
- Theano
- NumPy
- xarray
- SciPy
- Lasagne
- scikit-learn
- TensorFlow
- StatsModels

Business Intelligence
- Blaze
- xlwings

Scientific Computing / HPC
- Numba
- Cython
- PyTables
- HDFS

Distributed Systems
- Dask
- Hadoop
- Spark
- Hive

Databases / Storage
- MongoDB
- SQLAlchemy
- MySQL
- SQLite
- PostgreSQL
- Airflow
- Luigi
Anaconda

- 720+ Popular Packages
- Optimized & Compiled
- Free for Everyone
- Flexible *conda* package manager
- Sandboxed packages and libraries
- Cross-platform - Windows, Linux, Mac
- Not just Python - over 230 R packages
Python and the Big Data Ecosystem
Python for Scaling Up and Out

- Easy to use
  - Simple, easy to read/write syntax
- Batteries included
  - Ships with lots of basic functions
- Innovations from open source
  - Open access to a huge variety of existing libraries and algorithms
- Easy to get high performance when you need it
Scale Up and Scale Out with Python

Large memory and multi-core (CPU and GPU) machines

Best of both (e.g., GPU Cluster)

Multiple nodes in a cluster

Scale Up (Bigger Nodes)

Scale Out (More Nodes)
Using Python with Hadoop

- PySpark and SparkR
- Hadoop Streaming
- Spark Streaming
- Kafka clients (kafka-python, pykafka)
- Streamamparse for Storm
- Python integration with ELK
Continuum-Supported Foundational Open-Source Components

- **Conda**
  - Anaconda

- **NumPy**
  - SciPy
  - Pandas

- **Numba**
  - Bokeh
  - Datashader
  - matplotlib
  - HoloViews

- **Dask**

- **Package Management**

- **Data Analysis**

- **Optimization**

- **Visualization**

- **Parallelization**
Visualizing Large Datasets with Python
Bokeh

- Interactive visualization framework for web browsers
- No need to write JavaScript
- Python, R, Scala, and Lua bindings
- Easy to embed in web applications
- Easy to develop interactive applications and dashboards
- bokeh.pydata.org
Visualizing Large Datasets

Map showing 69795 tweets on 2012-02-16

Image: World map with blue dots representing tweets.
Visualizing Large Datasets

Overplotting

Undersampling
Datashader

- Graphics pipeline system for creating meaningful representations of large amounts of data
- Handles very large datasets in and out of core (e.g., billions of data points)
- [datashader.readthedocs.io](http://datashader.readthedocs.io)

NYC Census data by race
Datashader

NYC Taxi Pickups and Drop Offs

US Census Data
Parallel Computations with Python
Parallelizing the PyData ecosystem…

… without rewriting everything

The State of the Stack, Jake VanderPlas, SciPy 2015

Dask **complements** the Python ecosystem.

It was developed with NumPy, Pandas, and scikit-learn developers.
Overview of Dask

**Dask** is a Python parallel computing library that is:

- **Familiar**: Implements parallel NumPy and Pandas objects
- **Fast**: Optimized for demanding for numerical applications
- **Flexible**: for sophisticated and messy algorithms
- **Scales up**: Runs resiliently on clusters of 100s of machines
- **Scales down**: Pragmatic in a single process on a laptop
- **Interactive**: Responsive and fast for interactive data science
Spectrum of Parallelization

Explicit control: Fast but hard

Implicit control: Restrictive but easy

Threads
Processes
MPI
ZeroMQ

Dask

Hadoop
Spark

SQL:
Hive
Pig
Impala
Dask: From User Interaction to Execution

Collections → Graphs → Schedulers

- array
- bag
- dataframe
- imperative

- synchronous
- threaded
- multiprocessing
- distributed
Dask Collections: Familiar Expressions and API

<table>
<thead>
<tr>
<th>Dask array (mimics NumPy)</th>
<th>Dask bag (collection of data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x.T - x.mean(axis=0)</td>
<td>b.map(json.loads).foldby(...)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dask dataframe (mimics Pandas)</th>
<th>Dask delayed (wraps custom code)</th>
</tr>
</thead>
</table>
| df.groupby(df.index).value.mean() | def load(filename):
def clean(data):
def analyze(result):
|
Dask dataframe (mimics Pandas)

**Pandas dataframe**

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

**Dask dataframe**

```python
import dask.dataframe as dd
df = dd.read_csv('2015-***.csv')
df.groupby(df.user_id).value.mean().compute()
```
import numpy as np
f = h5py.File('myfile.hdf5')
x = np.array(f['/small-data'])
x - x.mean(axis=1)

import dask.array as da
f = h5py.File('myfile.hdf5')
x = da.from_array(f['/big-data'], chunks=(1000, 1000))
x - x.mean(axis=1).compute()
import dask.bag as db

b = db.read_text('2015-***.json.gz').map(json.loads)
b.pluck('name').frequencies().topk(10, lambda pair: pair[1]).compute()

from dask import delayed
L = []
for fn in filenames:
    data = delayed(load)(fn)  # Delay execution of function
    L.append(delayed(process)(data))  # Build connections between variables

result = delayed(summarize)(L)  # Use for loops to build up computation
result.compute()
Dask Graphs: Example Machine Learning Pipeline
Dask Graphs: Example Machine Learning Pipeline + Grid Search
Example Parallel Workflows with Python
Jupyter

• Open source, interactive data science and scientific computing across over 40 programming languages.

• Allows you to create and share documents that contain live code, equations, visualizations and explanatory text.
Examples

1. Analyzing NYC Taxi data using distributed Dask DataFrames
   - Demonstrate Pandas at scale
   - Observe responsive user interface

2. Distributed language processing using Dask Bags
   - Explore data using a distributed memory cluster
   - Integrate libraries into custom workflows

3. Visualizing large geospatial datasets using Datashader
   - Easily visualize large datasets
   - Reduce overplotting, saturation, and other pitfalls
Summary
Using Python with Hadoop

Bottom Line
10-100X faster performance
• Interact with data in HDFS and Amazon S3 natively from Python
• Distributed computations without the JVM & Python/Java serialization
• Framework for easy, flexible parallelism using directed acyclic graphs (DAGs)
• Interactive, distributed computing with in-memory persistence/caching

Python & R ecosystem
MPI

PySpark & SparkR

YARN
HDFS

HDFS
Impala
Kudu
Hive

Ibis

Batch Processing
Interactive Processing

High Performance, Interactive, Batch Processing
NumPy, Pandas, … 720+ packages

JVM

Python & R ecosystem

• Leverage Python & R with Spark
Open Data Science Stack

<table>
<thead>
<tr>
<th>Category</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Jupyter/IPython Notebook</td>
</tr>
<tr>
<td>Analytics</td>
<td>pandas, NumPy, SciPy, Numba, NLTK, scikit-learn, scikit-image, and more from Anaconda…</td>
</tr>
<tr>
<td>Parallel Computation</td>
<td>Dask, Spark, Hive, Impala</td>
</tr>
<tr>
<td>Data and Resource Management</td>
<td>HDFS, YARN, SGE, Slurm or other distributed systems</td>
</tr>
<tr>
<td>Server</td>
<td>Bare-metal or Cloud-based Cluster</td>
</tr>
</tbody>
</table>
Resources

- Packaging - Conda - conda.pydata.org
- Optimization - Numba - numba.pydata.org
- Visualization - Bokeh - bokeh.pydata.org
- Visualization - Datashader - datashader.readthedocs.io
- Parallelization - Dask - dask.pydata.org
- Notebooks - notebooks.anaconda.org/anaconda-cluster/
- Notebooks - anaconda.org/dask/notebooks
- Anaconda - continuum.io/downloads
Thank you

Kristopher Overholt
Twitter: @koverholt

Web: www.continuum.io
Twitter: @ContinuumIO