

RAPIDS

The Platform Inside and Out

Data Processing Evolution

Faster data access, less data movement

Hadoop Processing, Reading from disk

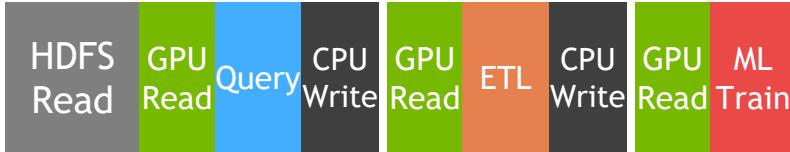


Spark In-Memory Processing



25-100x Improvement
Less code
Language flexible
Primarily In-Memory

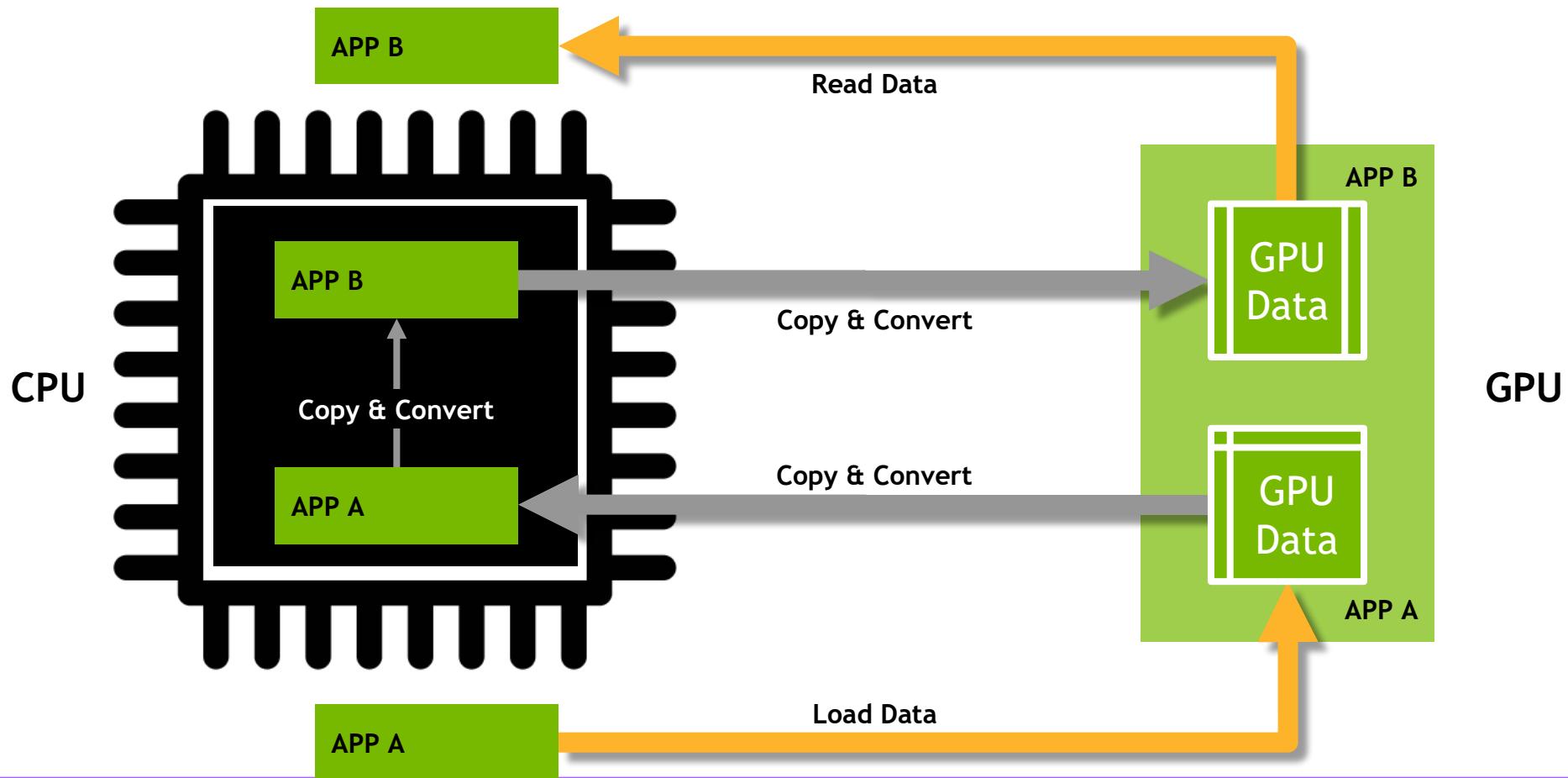
Traditional GPU Processing



5-10x Improvement
More code
Language rigid
Substantially on GPU

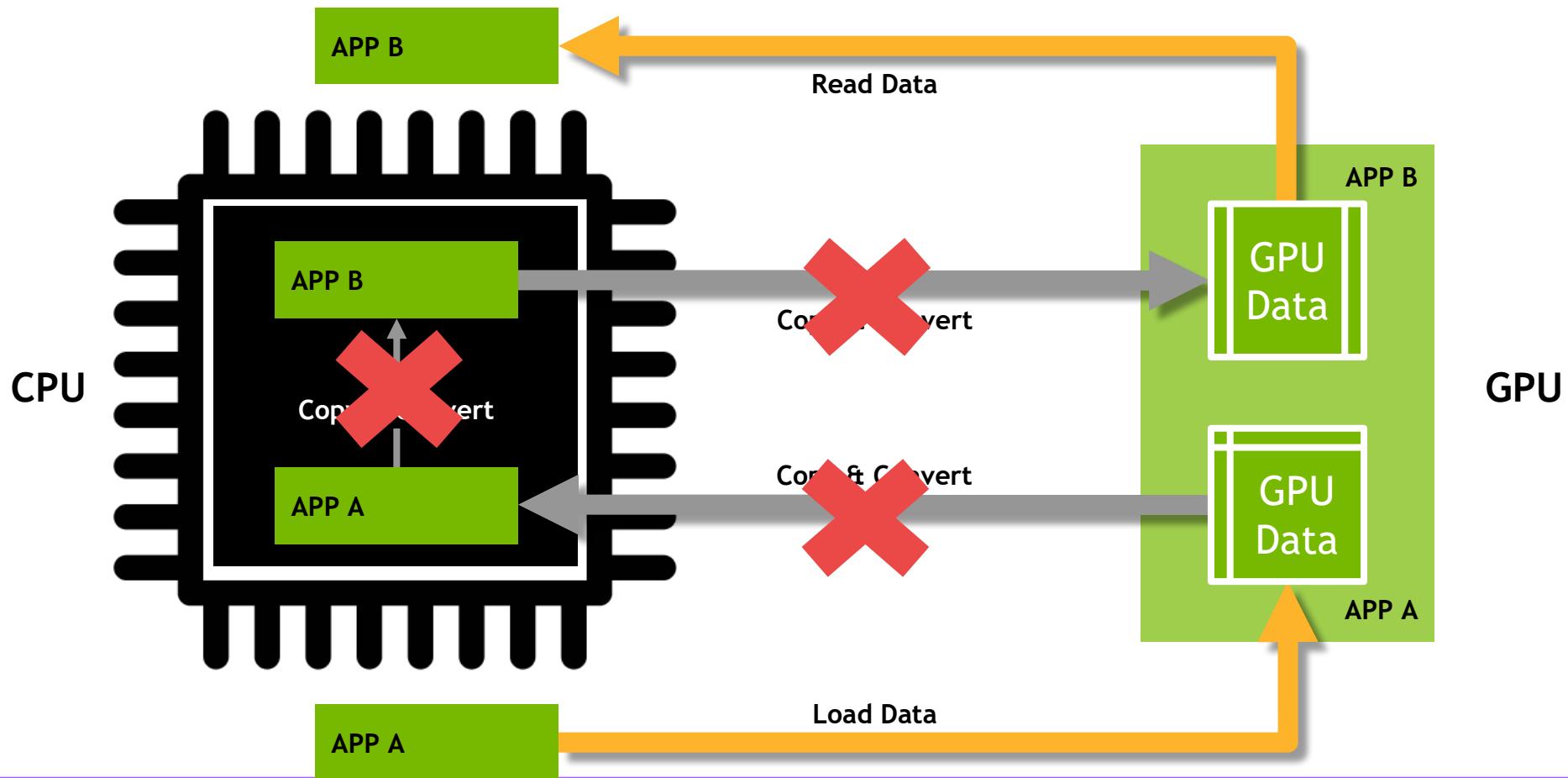
Data Movement and Transformation

The bane of productivity and performance

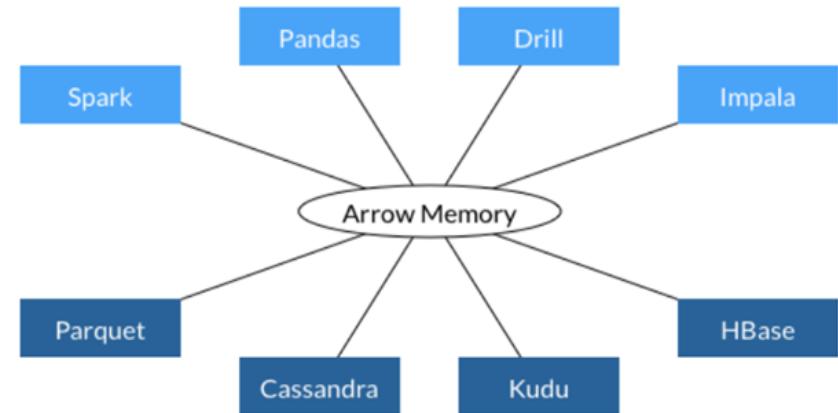
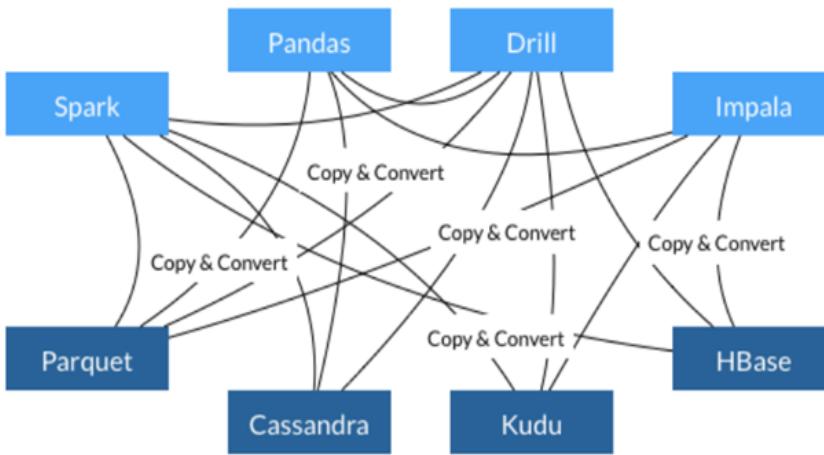


Data Movement and Transformation

What if we could keep data on the GPU?



Learning from Apache Arrow ➤➤➤



- Each system has its own internal memory format
- 70-80% computation wasted on serialization and deserialization
- Similar functionality implemented in multiple projects

- All systems utilize the same memory format
- No overhead for cross-system communication
- Projects can share functionality (eg, Parquet-to-Arrow reader)

From Apache Arrow Home Page - <https://arrow.apache.org/>

Data Processing Evolution

Faster data access, less data movement

Hadoop Processing, Reading from disk

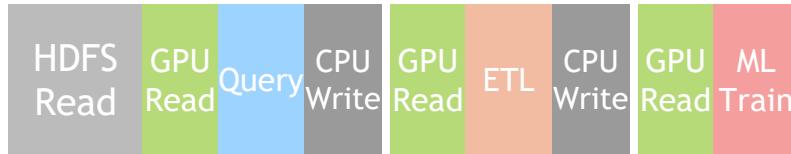


Spark In-Memory Processing



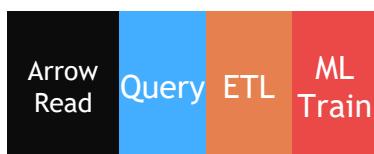
25-100x Improvement
Less code
Language flexible
Primarily In-Memory

Traditional GPU Processing



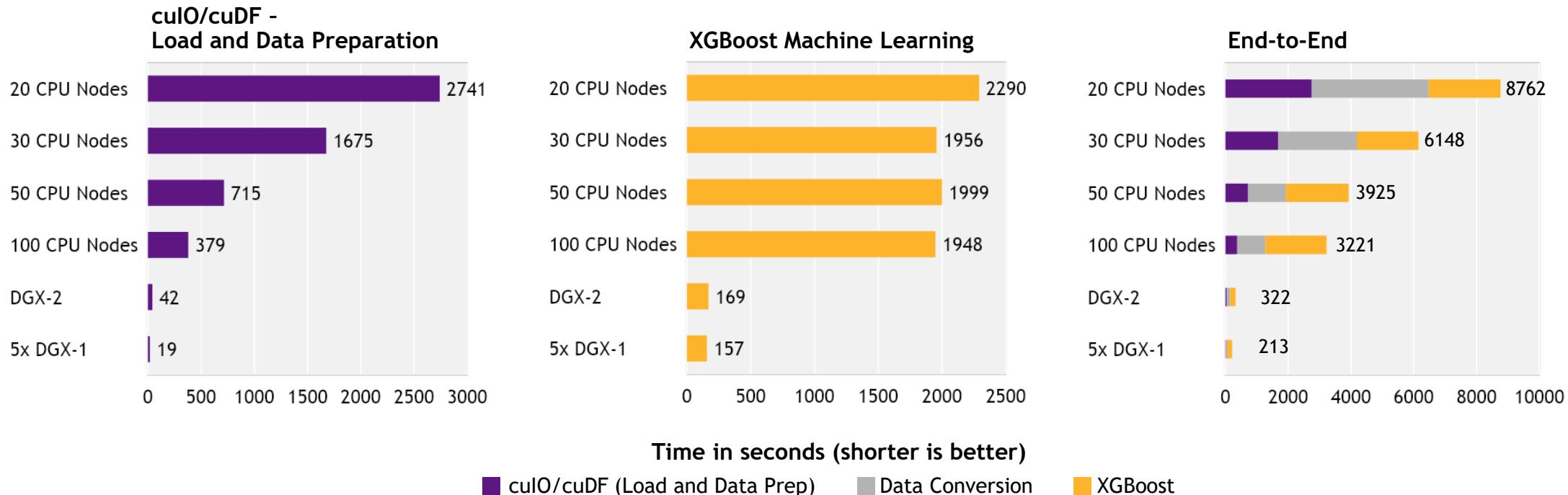
5-10x Improvement
More code
Language rigid
Substantially on GPU

RAPIDS



50-100x Improvement
Same code
Language flexible
Primarily on GPU

Faster Speeds, Real-World Benefits



Benchmark

200GB CSV dataset; Data prep includes joins, variable transformations

CPU Cluster Configuration

CPU nodes (61 GiB memory, 8 vCPUs, 64-bit platform), Apache Spark

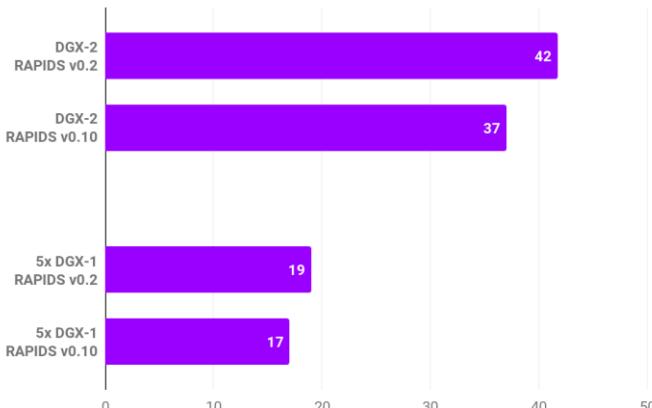
DGX Cluster Configuration

5x DGX-1 on InfiniBand network

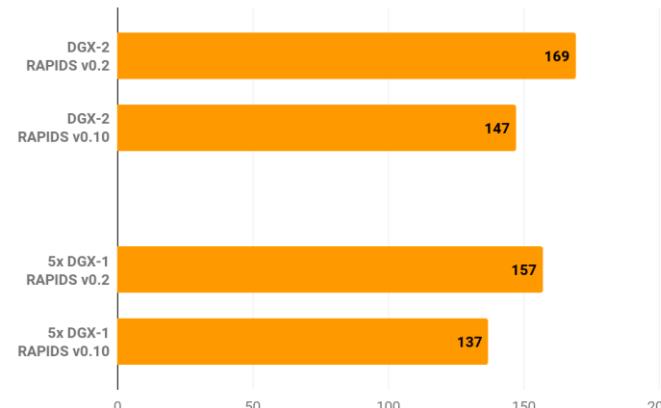
Faster Speeds, Real-World Benefits

Improving Over Time

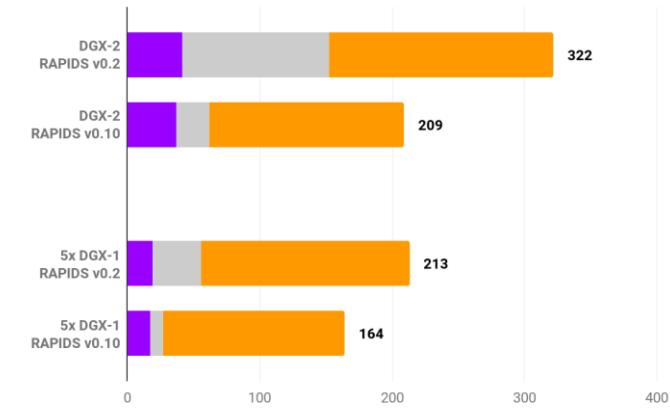
cuIO/cuDF -
Load and Data Preparation



XGBoost Machine Learning



End-to-End



Time in seconds (shorter is better)

■ cuIO/cuDF (Load and Data Prep)

■ Data Conversion

■ XGBoost

Benchmark

200GB CSV dataset; Data prep includes joins, variable transformations

CPU Cluster Configuration

CPU nodes (61 GiB memory, 8 vCPUs, 64-bit platform), Apache Spark

DGX Cluster Configuration

5x DGX-1 on InfiniBand network

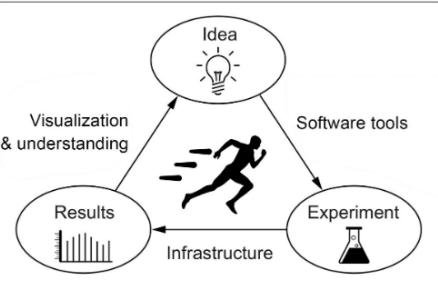
Speed, UX, and Iteration

The Way to Win at Data Science

François Chollet  @fchollet Following

Winners are those who went through *more iterations* of the "loop of progress" -- going from an idea, to its implementation, to actionable results. So the winning teams are simply those able to run through this loop *faster*.

And this is where Keras gives you an edge.



12:31 PM - 3 April 2019

50 Retweets 158 Likes

5 50 158

François Chollet  @fchollet · Apr 3

We often talk about how following UX best practices for API design makes Keras more accessible and easier to use, and how this helps beginners.

But those who stand to benefit most from good UX *aren't* the beginners. It's actually the very best practitioners in the world.

1 7 50

François Chollet  @fchollet · Apr 3

Because good UX reduces the overhead (development overhead & cognitive overhead) to setting up new experiments. It means you will be able to iterate faster. You will be able to try more ideas.

2 11 78

François Chollet  @fchollet · Apr 3

So I don't think it's mere personal preference if Kaggle champions are overwhelmingly using Keras.

Using Keras means you're more likely to win, and inversely, those who practice the sort of fast experimentation strategy that sets them up to win are more likely to prefer Keras.

8 8 74

Joshua Patterson  @datametrician · Apr 3

Replying to @fchollet

This is the fundamental belief that drives @RAPIDSai. @nvidia #GPU infrastructure is fast, people need to iterate quickly, people want a known #python interface. Combine them and you're off to the races!

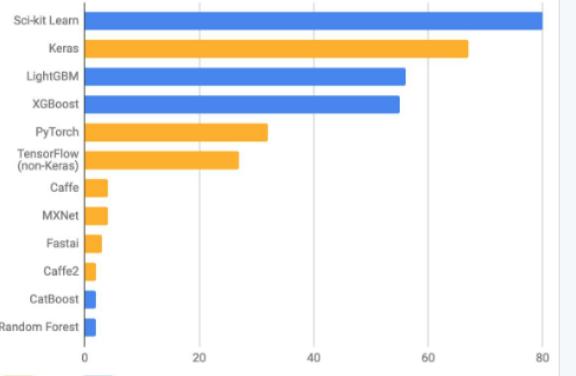
2 11

François Chollet  @fchollet · Apr 3

The second question asked about secondary frameworks -- usually teams win with an ensemble that involves many different ML frameworks. Here are *all* frameworks used.

Sklearn tops that ranking: everyone uses sklearn (although often as an auxiliary, for preprocessing or scoring).

All (primary + auxiliary) ML software tools used by top-5 Kaggle teams in each competition (n=120)



Tool	Count
Sci-kit Learn	~78
Keras	~65
LightGBM	~55
XGBoost	~55
PyTorch	~30
TensorFlow (non-Keras)	~25
Caffe	~10
MXNet	~10
Fastai	~10
Caffe2	~10
CatBoost	~10
R Random Forest	~10

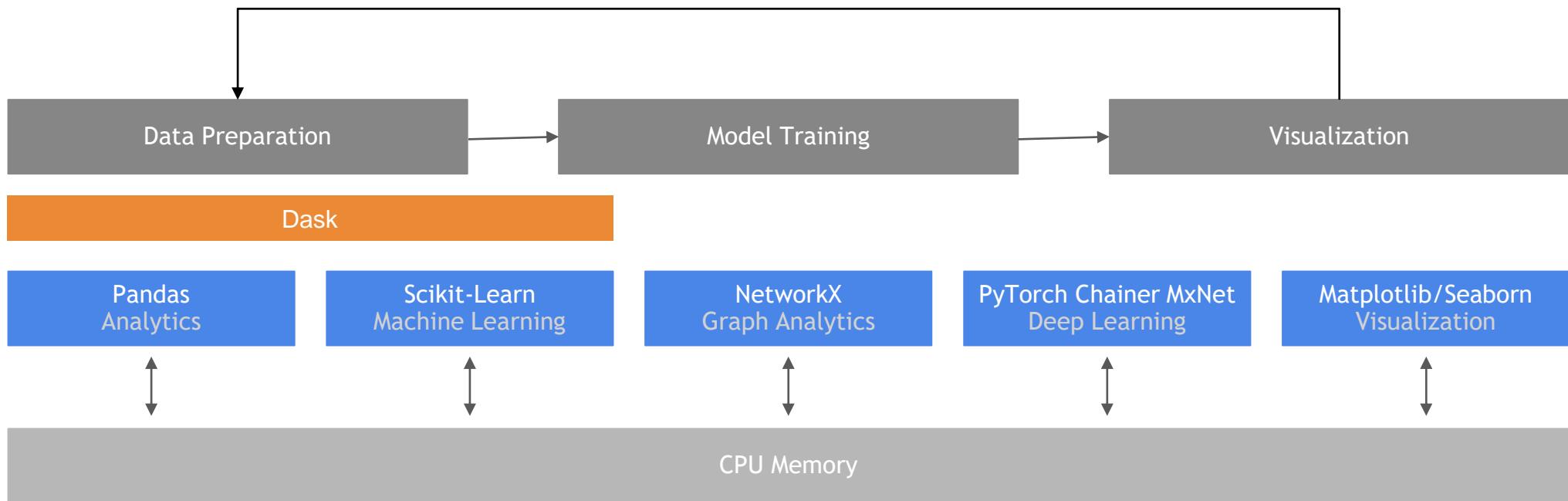
4 44 129

kaggle

RAPIDS Core

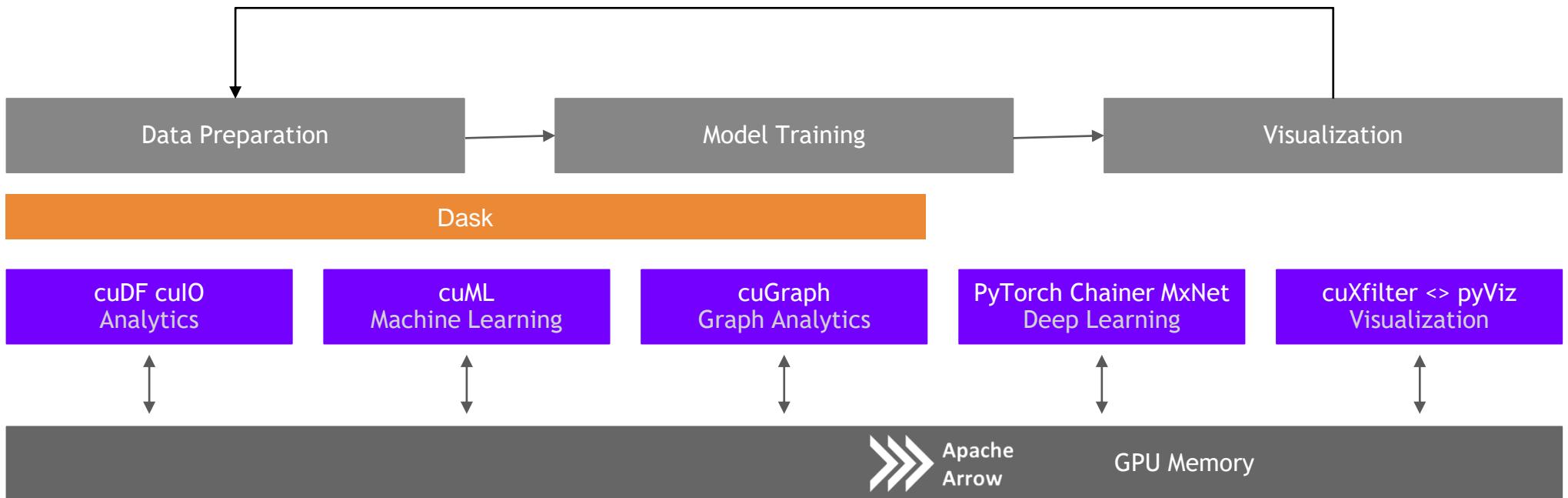
Open Source Data Science Ecosystem

Familiar Python APIs



RAPIDS

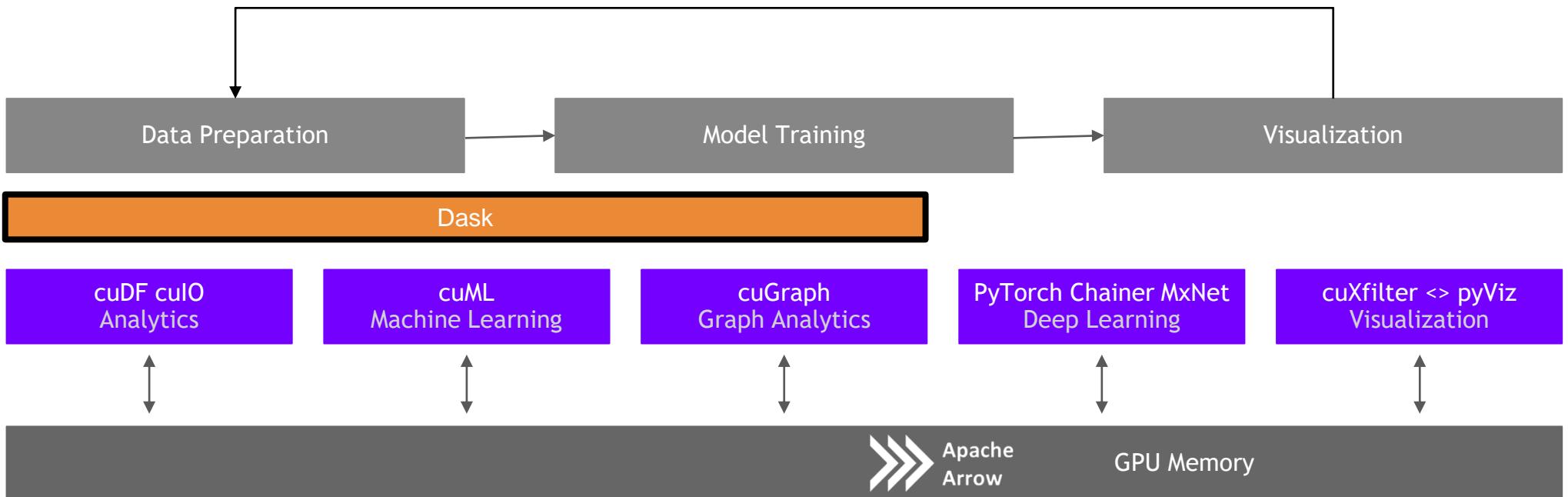
End-to-End Accelerated GPU Data Science



Dask

RAPIDS

Scaling RAPIDS with Dask



Why Dask?

PyData Native

- **Easy Migration:** Built on top of NumPy, Pandas, Scikit-Learn, etc.
- **Easy Training:** With the same APIs
- **Trusted:** With the same developer community

Deployable

- HPC: SLURM, PBS, LSF, SGE
- Cloud: Kubernetes
- Hadoop/Spark: Yarn



Easy Scalability

- Easy to install and use on a laptop
- Scales out to thousand-node clusters

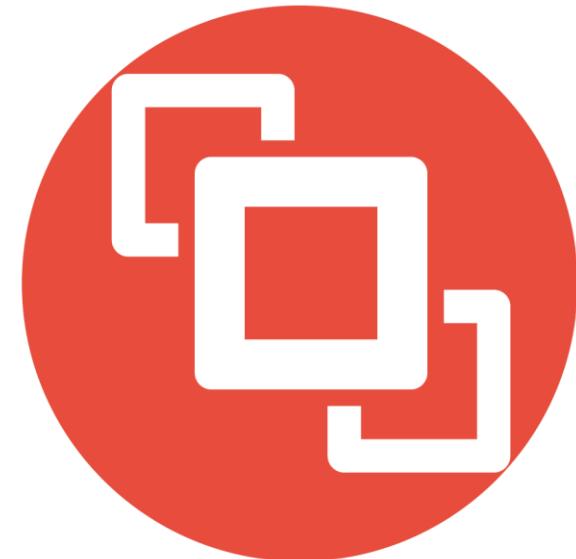
Popular

- Most common parallelism framework today in the PyData and SciPy community

Why OpenUCX?

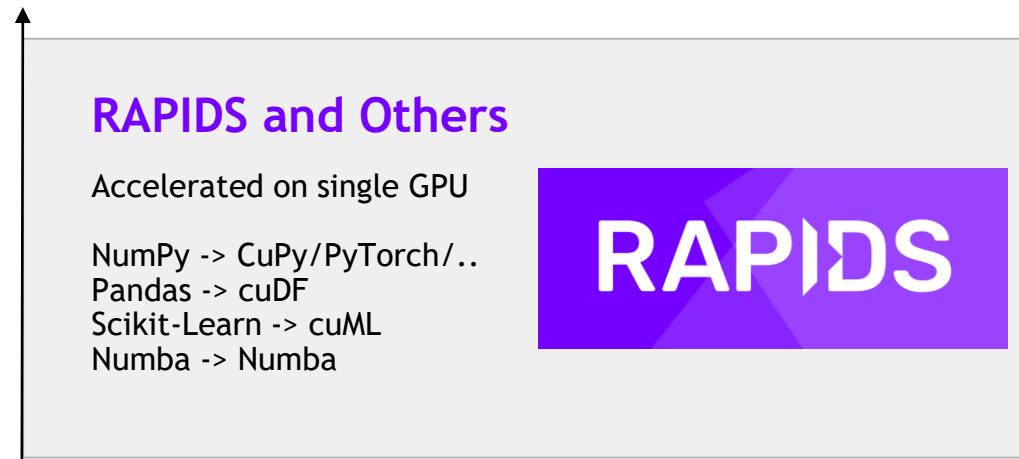
Bringing hardware accelerated communications to Dask

- TCP sockets are slow!
- UCX provides uniform access to transports (TCP, InfiniBand, shared memory, NVLink)
- Alpha Python bindings for UCX (ucx-py)
<https://github.com/rapidsai/ucx-py>
- Will provide best communication performance, to Dask based on available hardware on nodes/cluster

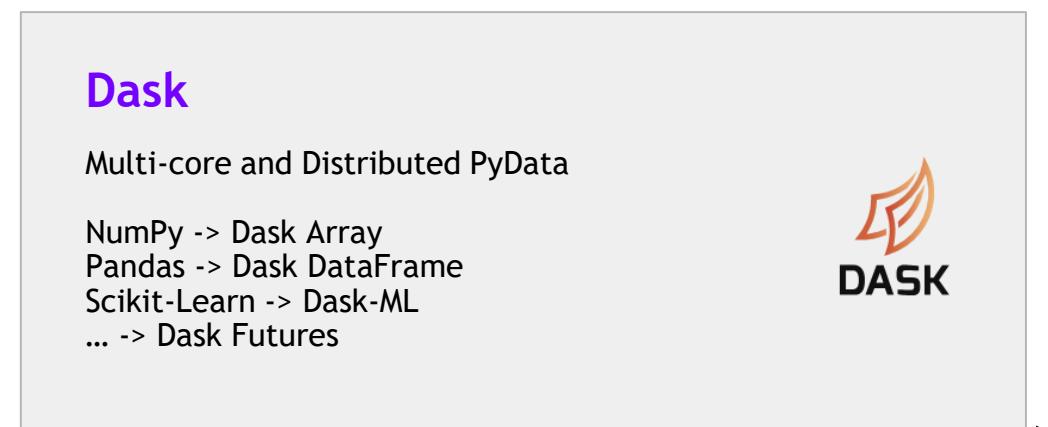
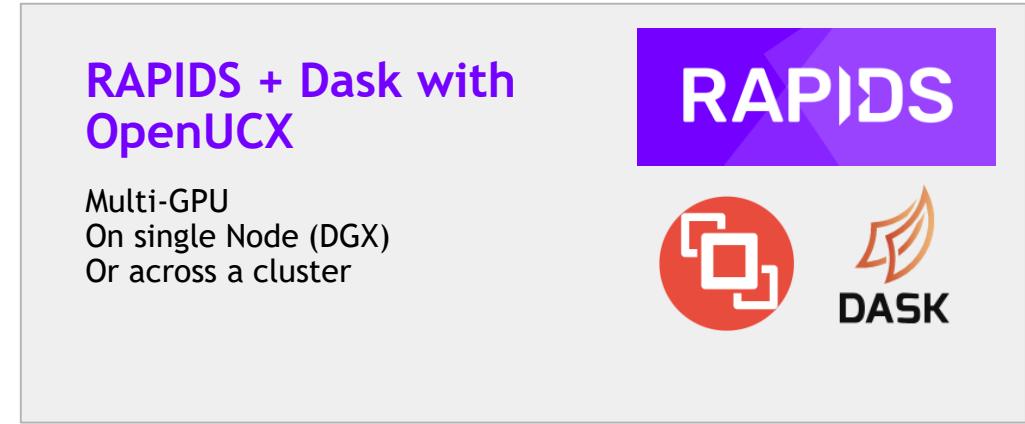
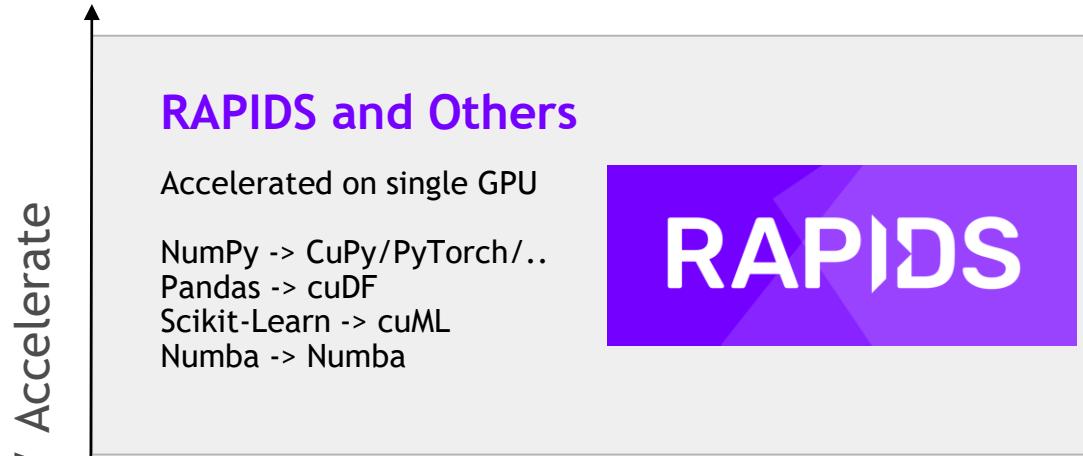


Scale up with RAPIDS

Scale Up / Accelerate



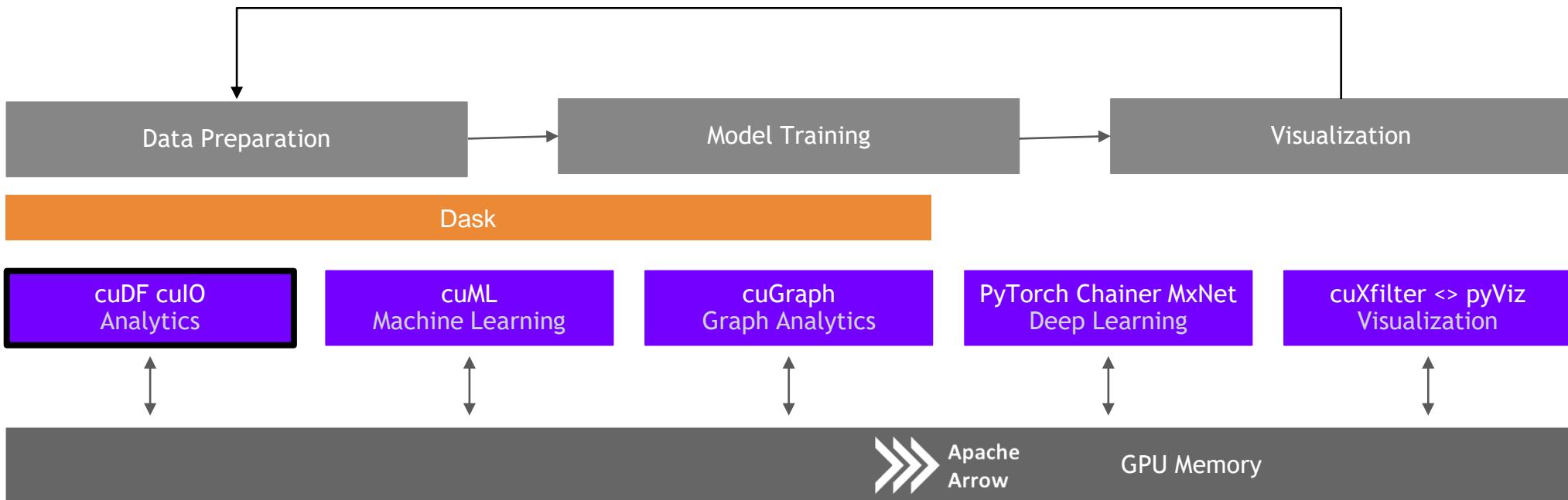
Scale out with RAPIDS + Dask with OpenUCX



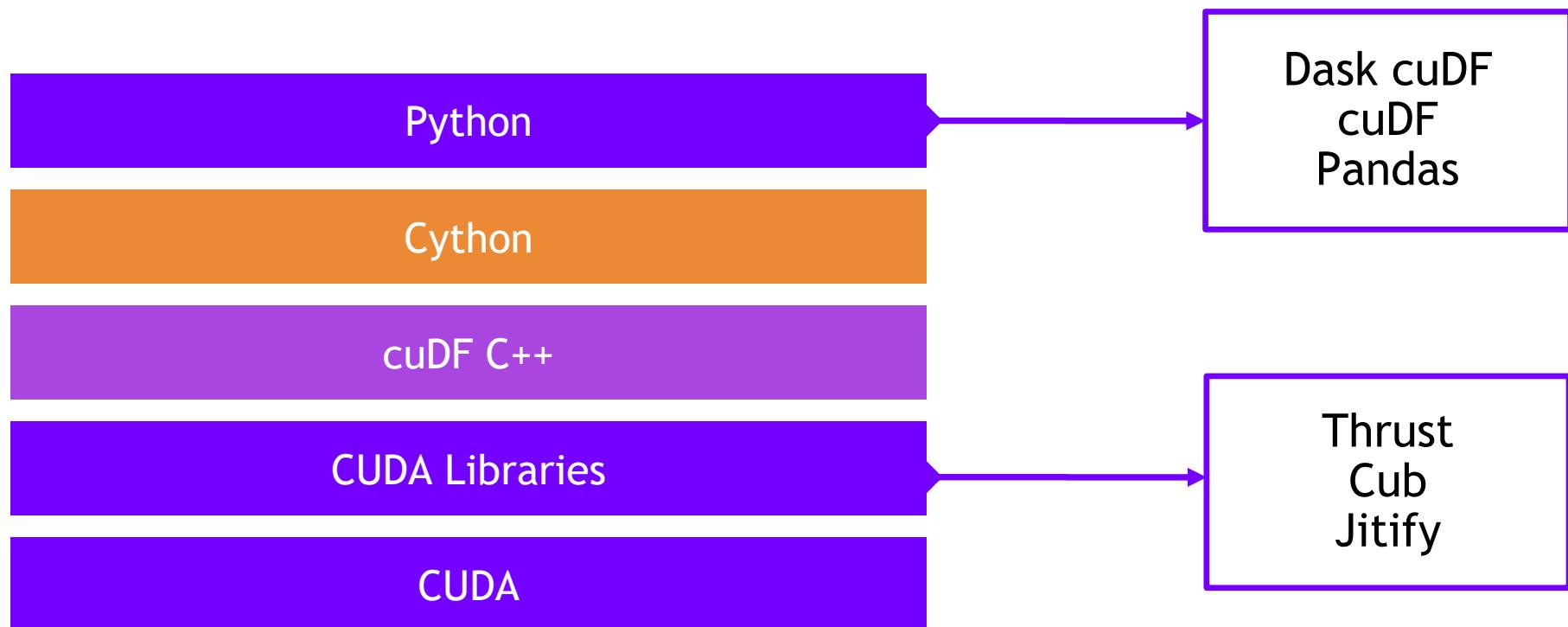
cuDF

RAPIDS

GPU Accelerated data wrangling and feature engineering



ETL Technology Stack



ETL - the Backbone of Data Science

cuDF is...

```
In [2]: #Read in the data. Notice how it decompresses as it reads the data into memory.  
gdf = cudf.read_csv('/rapids/Data/black-friday.zip')
```

```
In [3]: #Taking a look at the data. We use "to_pandas()" to get the pretty printing.  
gdf.head().to_pandas()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Cat
0	1000001	P00069042	F	0-17	10	A	2	0	3
1	1000001	P00248942	F	0-17	10	A	2	0	1
2	1000001	P00087842	F	0-17	10	A	2	0	12
3	1000001	P00085442	F	0-17	10	A	2	0	12
4	1000002	P00285442	M	55+	16	C	4+	0	8

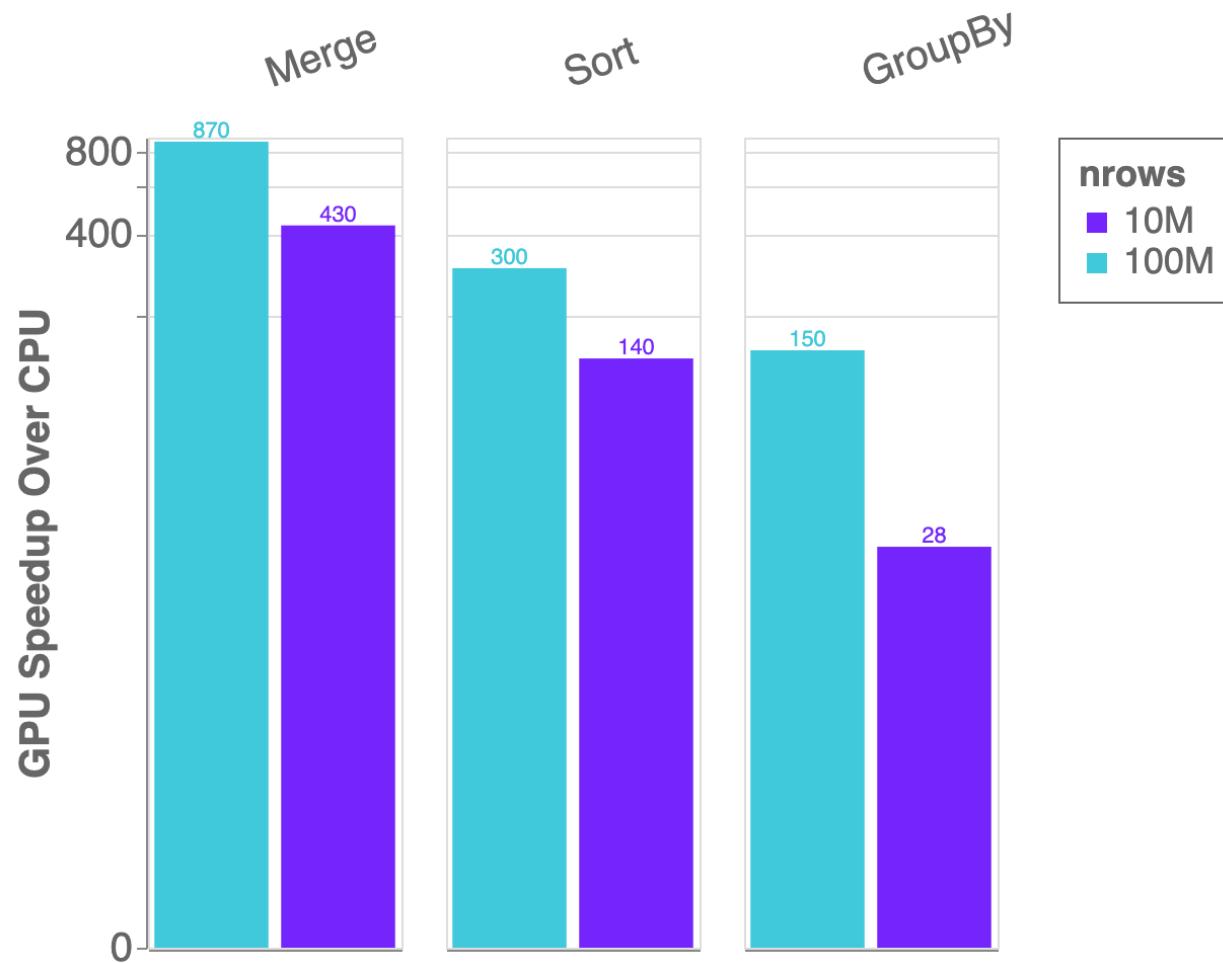
```
In [6]: #grabbing the first character of the years in city string to get rid of plus sign, and converting  
#to int  
gdf['city_years'] = gdf.Stay_In_Current_City_Years.str.get(0).stoi()
```

```
In [7]: #Here we can see how we can control what the value of our dummies with the replace method and turn  
#strings to ints  
gdf['City_Category'] = gdf.City_Category.str.replace('A', '1')  
gdf['City_Category'] = gdf.City_Category.str.replace('B', '2')  
gdf['City_Category'] = gdf.City_Category.str.replace('C', '3')  
gdf['City_Category'] = gdf['City_Category'].str.stoi()
```

Python Library

- A Python library for manipulating GPU DataFrames following the Pandas API
- Python interface to CUDA C++ library with additional functionality
- Creating GPU DataFrames from Numpy arrays, Pandas DataFrames, and PyArrow Tables
- JIT compilation of User-Defined Functions (UDFs) using Numba

Benchmarks: single-GPU Speedup vs. Pandas



cuDF v0.10, Pandas 0.24.2

Running on NVIDIA DGX-1:

GPU: NVIDIA Tesla V100 32GB

CPU: Intel(R) Xeon(R) CPU E5-2698 v4
@ 2.20GHz

Benchmark Setup:

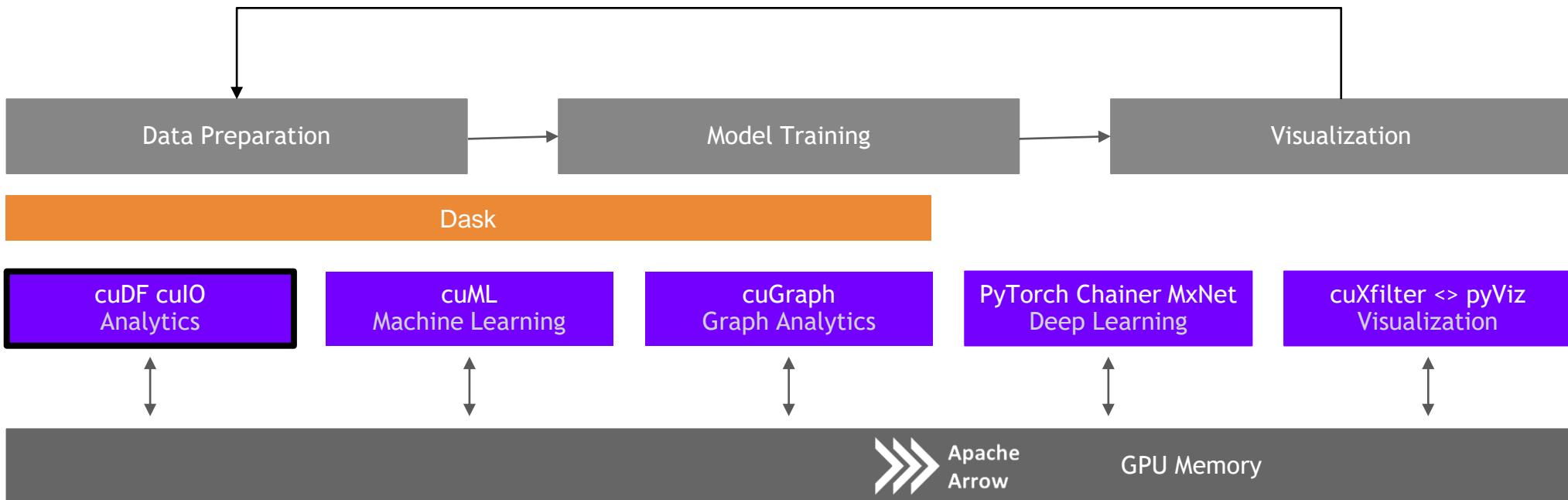
DataFrames: 2x int32 columns key columns,
3x int32 value columns

Merge: inner

GroupBy: count, sum, min, max calculated
for each value column

ETL - the Backbone of Data Science

cuDF is not the end of the story



Extraction is the Cornerstone

cudf for Faster Data Loading

- Follow Pandas APIs and provide >10x speedup
- CSV Reader - v0.2, CSV Writer v0.8
- Parquet Reader - v0.7, Parquet Writer v0.11
- ORC Reader - v0.7, ORC Writer v0.10
- JSON Reader - v0.8
- Avro Reader - v0.9
- GPU Direct Storage integration in progress for bypassing PCIe bottlenecks!
- Key is GPU-accelerating both parsing and decompression wherever possible

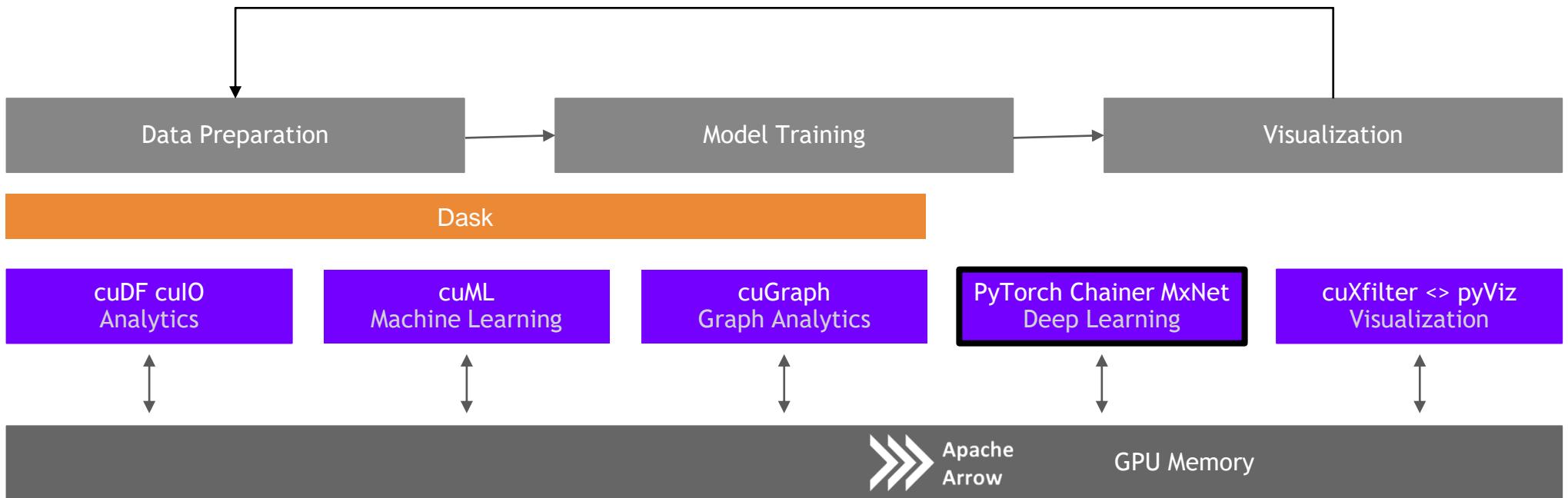
```
1]: import pandas, cudf
2]: %time len(pandas.read_csv('data/nyc/yellow_tripdata_2015-01.csv'))
CPU times: user 25.9 s, sys: 3.26 s, total: 29.2 s
Wall time: 29.2 s
2]: 12748986
3]: %time len(cudf.read_csv('data/nyc/yellow_tripdata_2015-01.csv'))
CPU times: user 1.59 s, sys: 372 ms, total: 1.96 s
Wall time: 2.12 s
3]: 12748986
4]: !du -hs data/nyc/yellow_tripdata_2015-01.csv
1.9G    data/nyc/yellow_tripdata_2015-01.csv
```

Source: Apache Crail blog: [SQL Performance: Part 1 - Input File Formats](#)

ETL is not just DataFrames!

RAPIDS

Building bridges into the array ecosystem

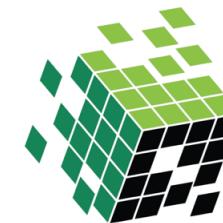


Interoperability for the Win

DLPack and __cuda_array_interface__



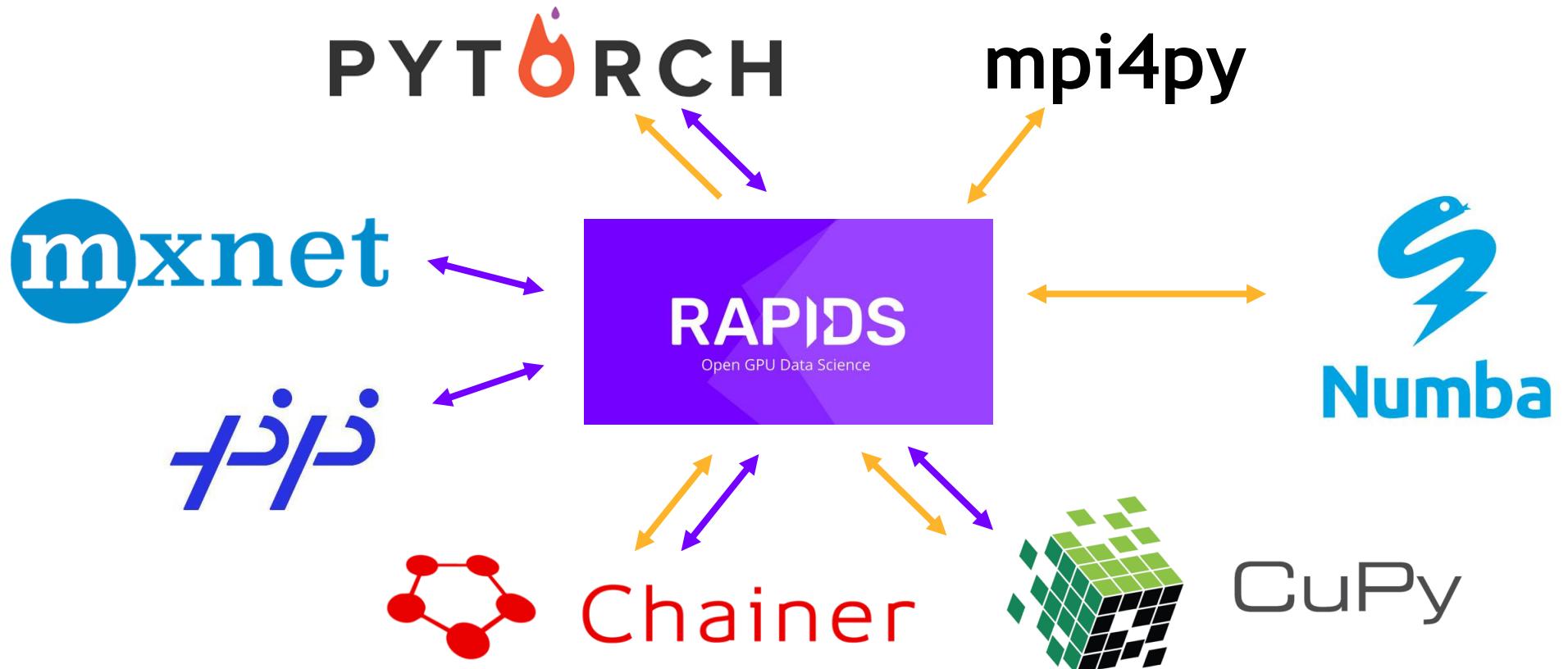
Chainer



CuPy

Interoperability for the Win

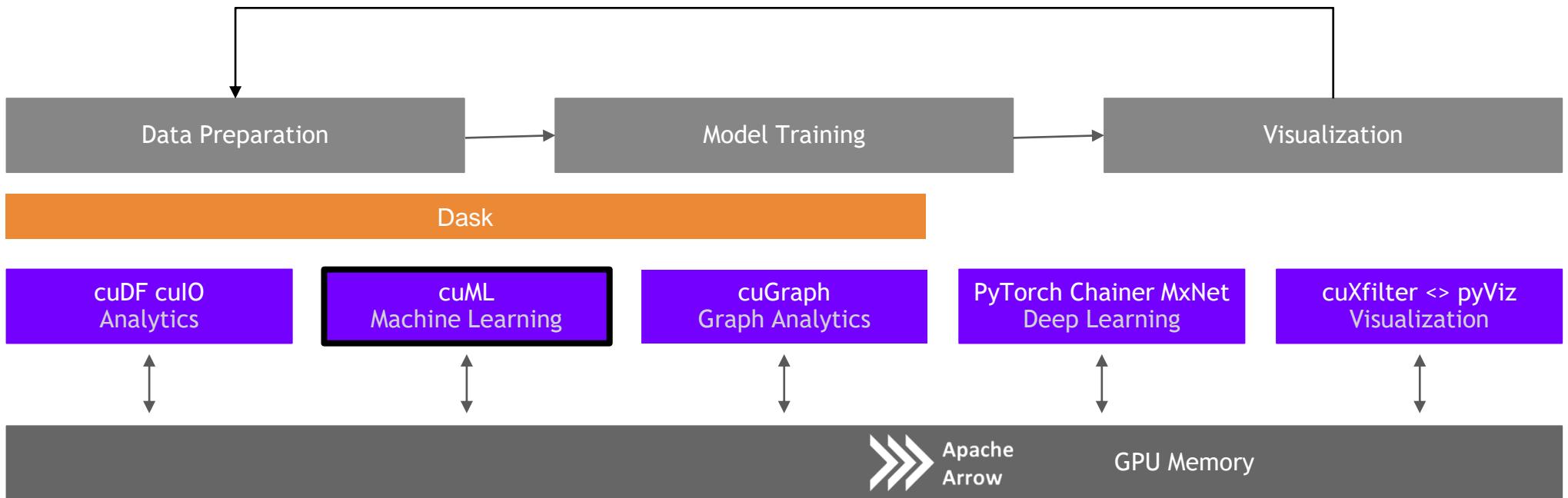
DLPack and `_cuda_array_interface_`



cuML

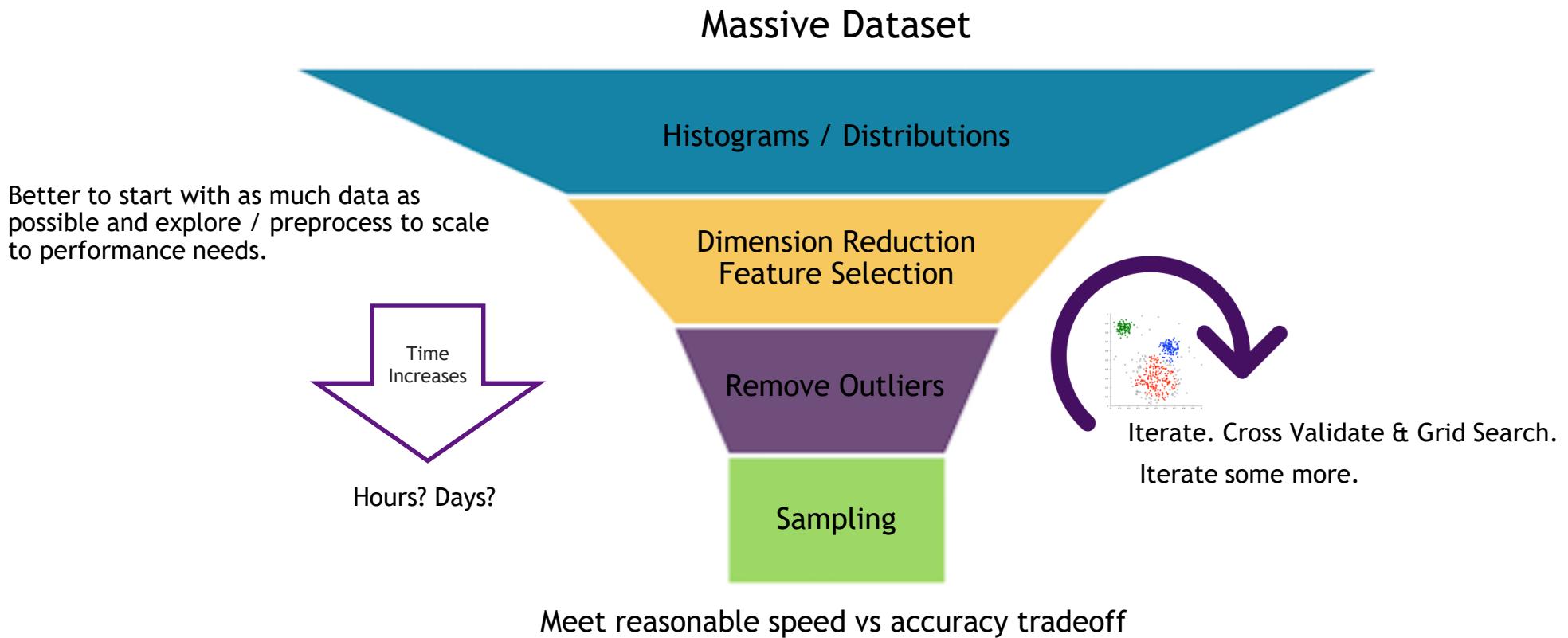
Machine Learning

More models more problems

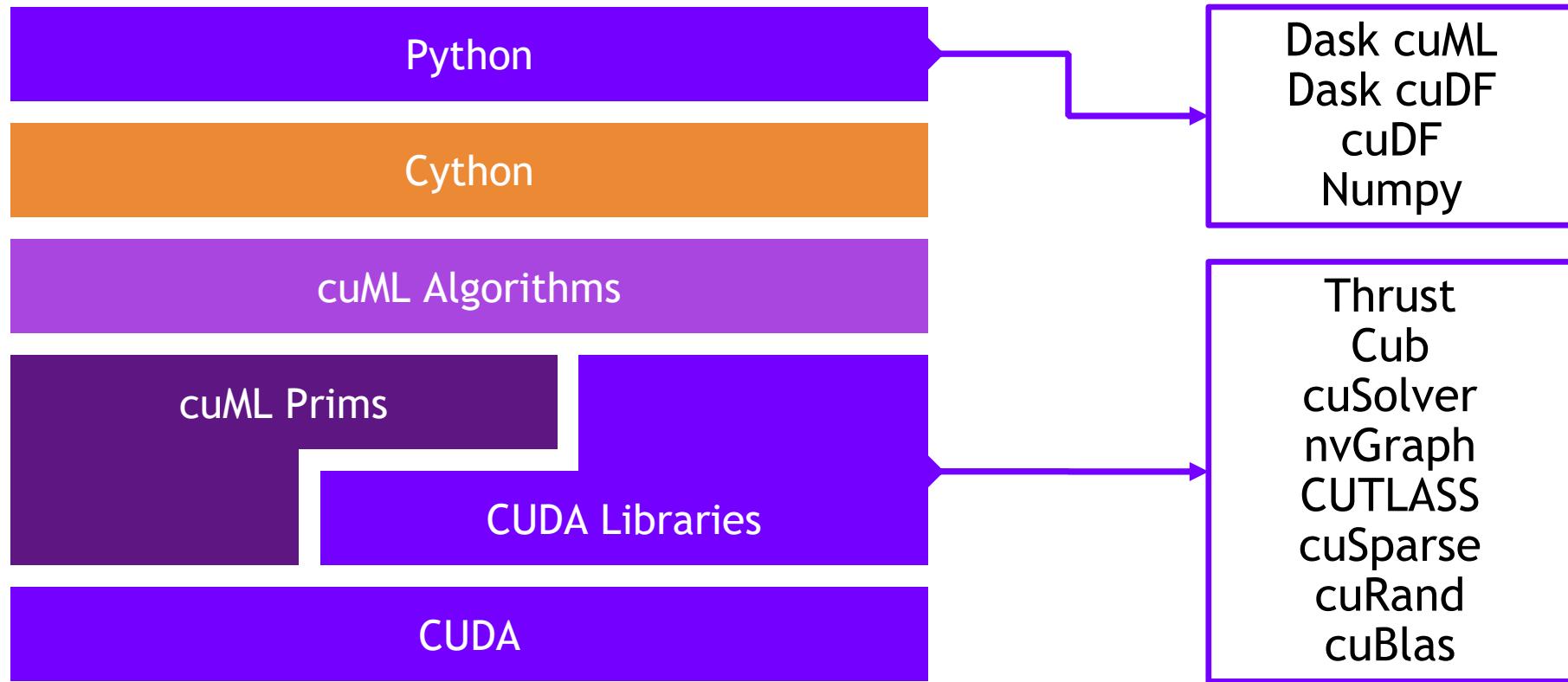


Problem

Data sizes continue to grow

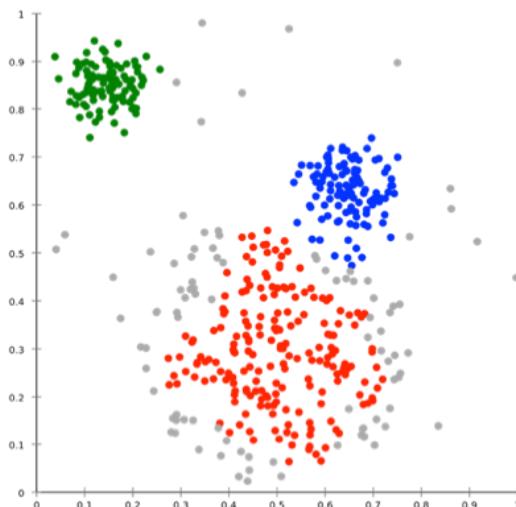


ML Technology Stack



Algorithms

GPU-accelerated Scikit-Learn



Cross Validation

Hyper-parameter Tuning

More to come!

Classification / Regression

Inference

Clustering

Decomposition & Dimensionality Reduction

Time Series

Decision Trees / Random Forests
Linear Regression
Logistic Regression
K-Nearest Neighbors
Support Vector Machine Classification

Random forest / GBDT inference

K-Means
DBSCAN
Spectral Clustering

Principal Components
Singular Value Decomposition
UMAP
Spectral Embedding
T-SNE

Holt-Winters
Kalman Filtering

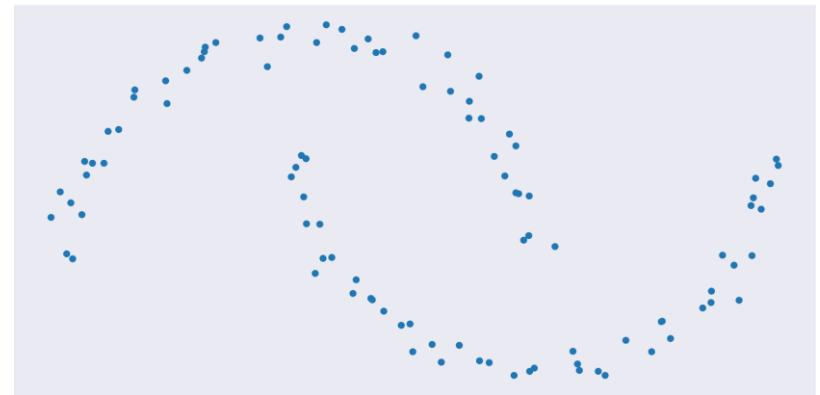
Key:

- Preexisting
- **NEW for 0.10**

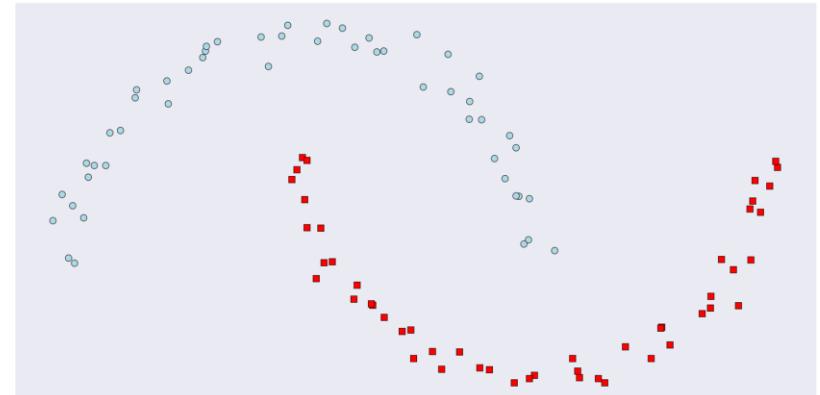
RAPIDS matches common Python APIs

CPU-Based Clustering

```
from sklearn.datasets import make_moons  
import pandas  
  
X, y = make_moons(n_samples=int(1e2),  
                   noise=0.05, random_state=0)  
  
X = pandas.DataFrame({'fea%d'%i: X[:, i]  
                      for i in range(X.shape[1])})
```



```
from sklearn.cluster import DBSCAN  
dbSCAN = DBSCAN(eps = 0.3, min_samples = 5)  
  
dbSCAN.fit(X)  
  
y_hat = dbSCAN.predict(X)
```



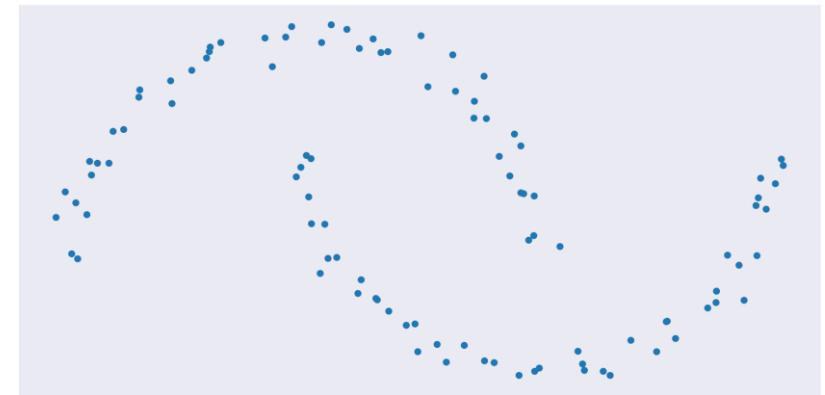
RAPIDS matches common Python APIs

GPU-Accelerated Clustering

```
from sklearn.datasets import make_moons
import cudf

X, y = make_moons(n_samples=int(1e2),
                   noise=0.05, random_state=0)

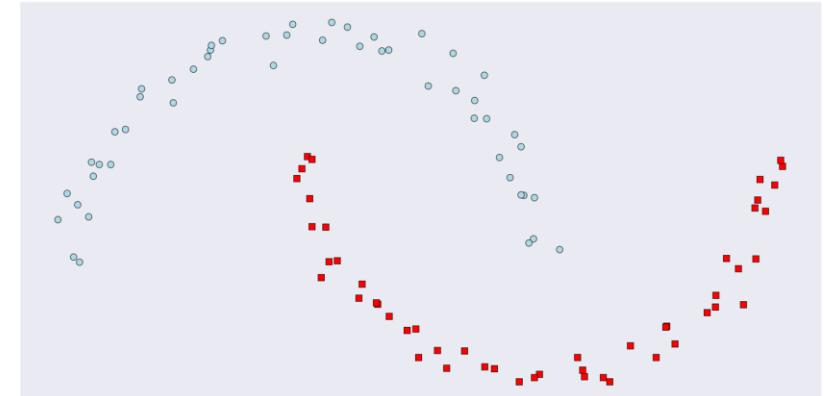
X = cudf.DataFrame({'fea%d'%i: X[:, i]
                    for i in range(X.shape[1])})
```



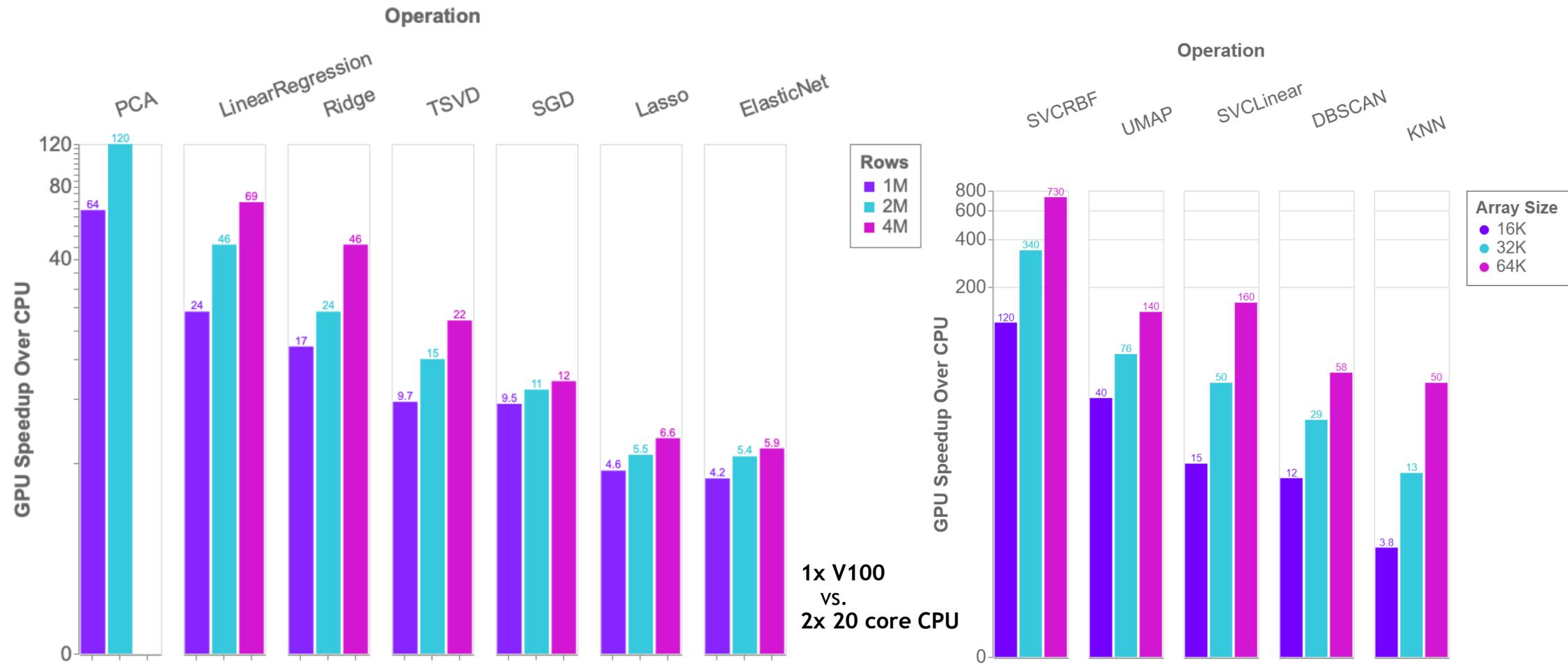
```
from cuml import DBSCAN
dbSCAN = DBSCAN(eps = 0.3, min_samples = 5)

dbSCAN.fit(X)

y_hat = dbSCAN.predict(X)
```



Benchmarks: single-GPU cuML vs scikit-learn

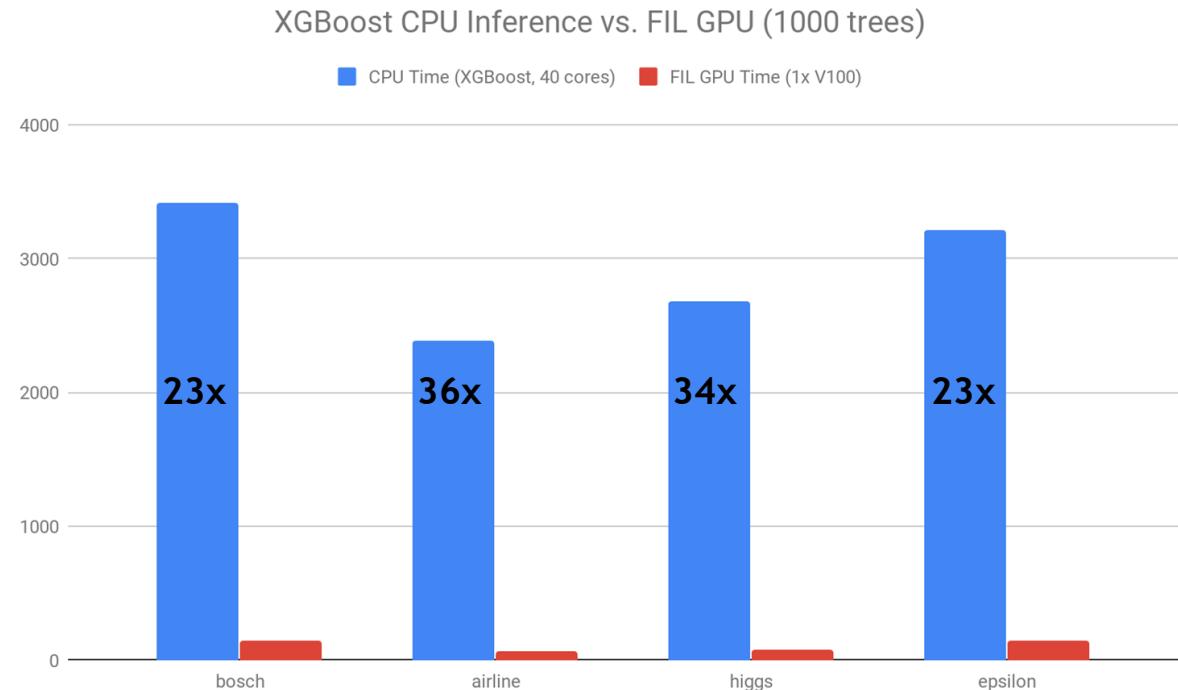


Forest Inference

Taking models from training to production

cuML's **Forest Inference Library** accelerates prediction (inference) for random forests and boosted decision trees:

- Works with existing saved models (XGBoost and LightGBM today, scikit-learn RF and cuML RF soon)
- Lightweight Python API
- Single V100 GPU can infer up to 34x faster than XGBoost dual-CPU node
- Over 100 million forest inferences per sec (with 1000 trees) on a DGX-1



Road to 1.0

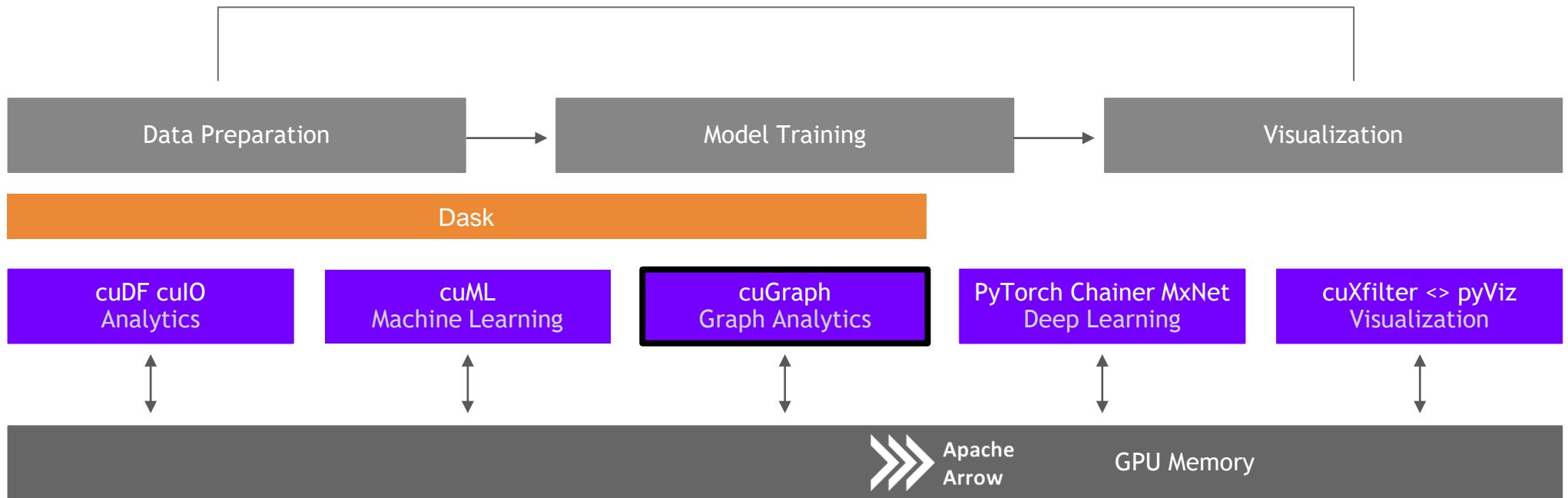
March 2020 - RAPIDS 0.13

cuML	Single-GPU	Multi-Node-Multi-GPU
Gradient Boosted Decision Trees (GBDT)		
GLM		
Logistic Regression		
Random Forest		
K-Means		
K-NN		
DBSCAN		
UMAP		
ARIMA & Holt-Winters		
Kalman Filter		
t-SNE		
Principal Components		
Singular Value Decomposition		
SVM		

cuGraph

Graph Analytics

More connections more insights



GOALS AND BENEFITS OF CUGRAPH

Focus on Features and User Experience

Breakthrough Performance

- Up to 500 million edges on a single 32GB GPU
- Multi-GPU support for scaling into the billions of edges

Multiple APIs

- Python: Familiar NetworkX-like API
- C/C++: lower-level granular control for application developers

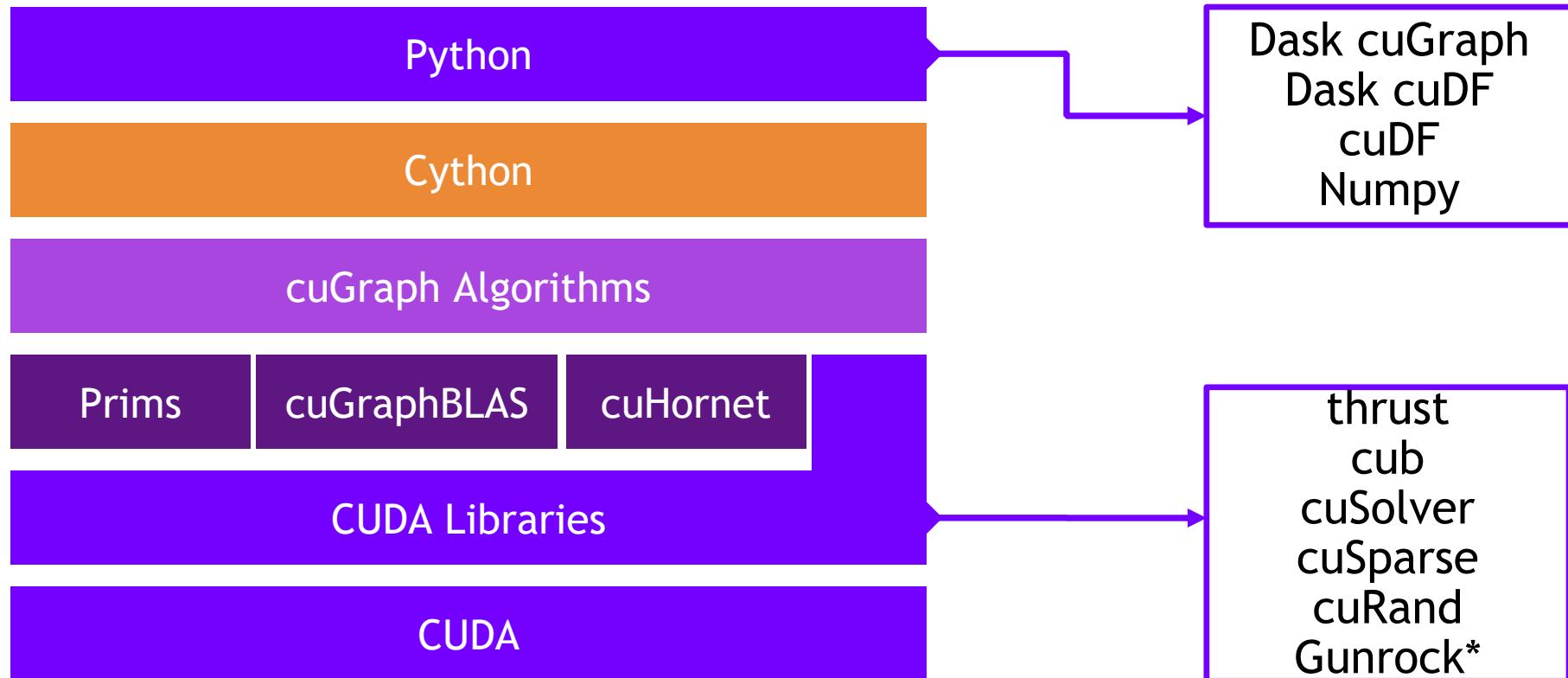
Seamless Integration with cuDF and cuML

- Property Graph support via DataFrames

Growing Functionality

- Extensive collection of algorithm, primitive, and utility functions

Graph Technology Stack

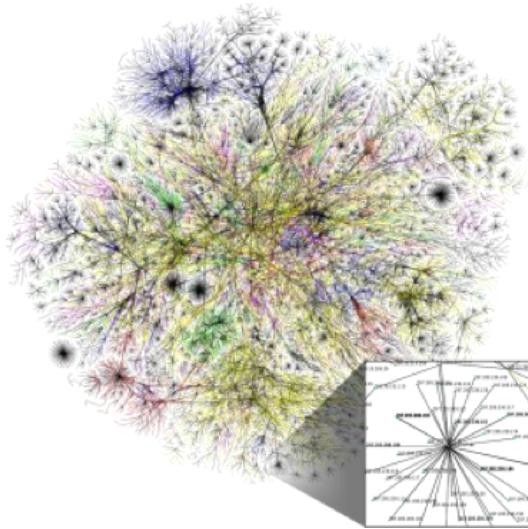


nvGRAPH has been Opened Sourced and integrated into cuGraph. A legacy version is available in a RAPIDS GitHub repo

* Gunrock is from UC Davis

Algorithms

GPU-accelerated NetworkX



Query Language

Multi-GPU

Utilities

More to come!

Renumbering

Community

Components

Link Analysis

Link Prediction

Traversal

Structure

Spectral Clustering
Balanced-Cut
Modularity Maximization
Louvain
Subgraph Extraction
KCore

Weakly Connected Components
Strongly Connected Components

Page Rank (Multi-GPU)
Personal Page Rank
Katz

Jaccard
Weighted Jaccard
Overlap Coefficient

Single Source Shortest Path (SSSP)
Breadth First Search (BFS)

Triangle Counting
COO-to-CSR (Multi-GPU)
Transpose

Multi-GPU PageRank Performance

PageRank portion of the HiBench benchmark suite

HiBench Scale	Vertices	Edges	CSV File (GB)	# of GPUs	# of CPU Threads	PageRank for 3 Iterations (secs)
Huge	5,000,000	198,000,000	3	1		1.1
BigData	50,000,000	1,980,000,000	34	3		5.1
BigData x2	100,000,000	4,000,000,000	69	6		9.0
BigData x4	200,000,000	8,000,000,000	146	12		18.2
BigData x8	400,000,000	16,000,000,000	300	16		31.8
BigData x8	400,000,000	16,000,000,000	300		800*	5760*

*BigData x8, 100x 8-vCPU nodes, Apache Spark GraphX ⇒ 96 mins!

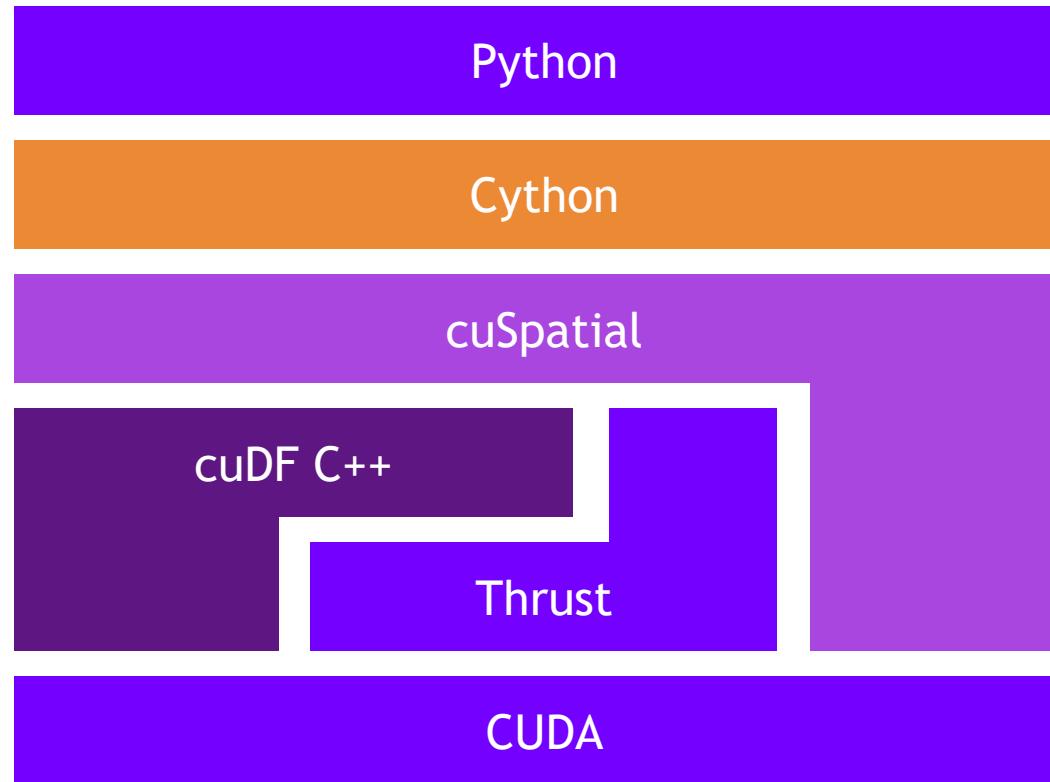
Road to 1.0

March 2020 - RAPIDS 0.13

cuGraph	Single-GPU	Multi-Node-Multi-GPU
Jaccard and Weighted Jaccard		
Page Rank		
Personal Page Rank		
SSSP		
BFS		
Triangle Counting		
Subgraph Extraction		
Katz Centrality		
Betweenness Centrality		
Connected Components (Weak and Strong)		
Louvain		
Spectral Clustering		
K-Cores		

cuSpatial

cuSpatial Technology Stack



cuSpatial 0.10

Breakthrough Performance & Ease of Use

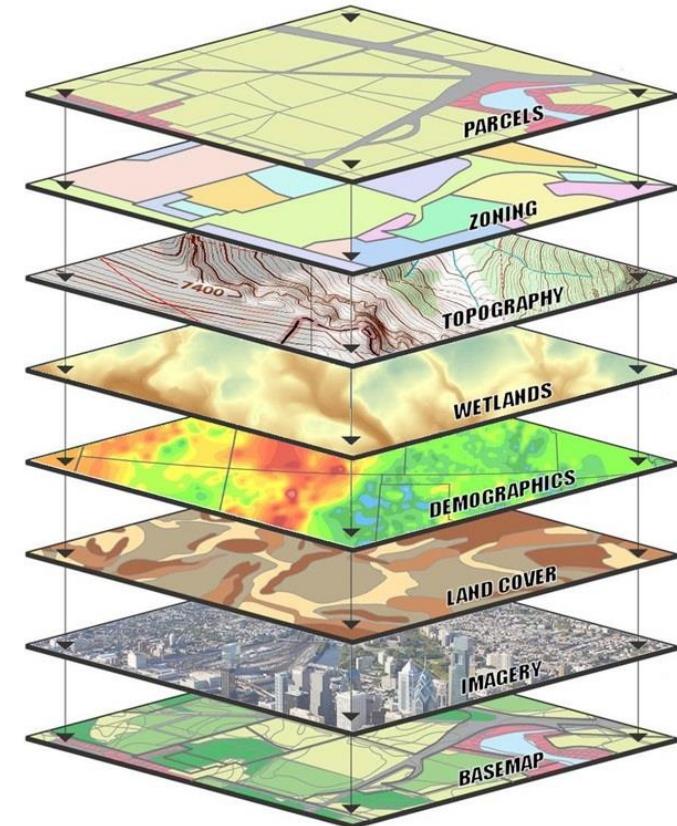
- Up to 1000x faster than CPU spatial libraries
- Python and C++ APIs for maximum usability and integration

Growing Functionality

- Extensive collection of algorithm, primitive, and utility functions for spatial analytics

Seamless Integration into RAPIDS

- cuDF for data loading, cuGraph for routing optimization, and cuML for clustering are just a few examples



cuSpatial

0.10 and Beyond

Layer	0.10/0.11 Functionality	Functionality Roadmap (2020)
High-level Analytics	C++ Library w. Python bindings enabling distance, speed, trajectory similarity, trajectory clustering	C++ Library w. Python bindings for additional spatio-temporal trajectory clustering, acceleration, dwell-time, salient locations, trajectory anomaly detection, origin destination, etc.
Graph layer	cuGraph	Map matching, Djikstra algorithm, Routing
Query layer	Spatial Window	Nearest Neighbor, KNN, Spatiotemporal range search and joins
Index layer		Grid, Quad Tree, R-Tree, Geohash, Voronoi Tessellation
Geo-operations	Point in polygon (PIP), Haversine distance, Hausdorff distance, lat-lon to xy transformation	Line intersecting polygon, Other distance functions, Polygon intersection, union
Geo-representation	Shape primitives, points, polylines, polygons	Additional shape primitives

RAPIDS in Enterprise

Apache Spark

Flexible, in-memory data processing for Big Data

Easy Development

- Rich APIs for Scala, Java, and Python
- Interactive shell

Flexible Extensible API

- APIs for different types of workloads:
 - Batch
 - Streaming
 - Machine Learning
 - Graph

Fast Batch & Stream Processing

- In-Memory processing and caching

Easy Development

High Productivity Language Support

Python

```
lines = sc.textFile(...)  
lines.filter(lambda s: "ERROR" in s).count()
```

Scala

```
val lines = sc.textFile(...)  
lines.filter(s => s.contains("ERROR")).count()
```

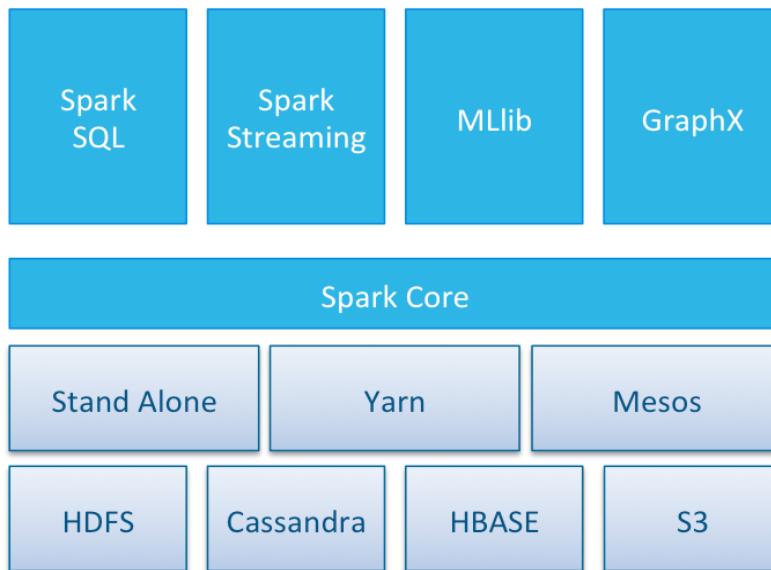
Java

```
JavaRDD<String> lines = sc.textFile(...);  
lines.filter(new Function<String, Boolean>() {  
    Boolean call(String s) {  
        return s.contains("error");  
    }  
}).count();
```

- Native support for multiple languages with identical APIs
 - Scala, Java, Python
- Use of closures, iterations, and other common language constructs to minimize code
 - 2-5x less code

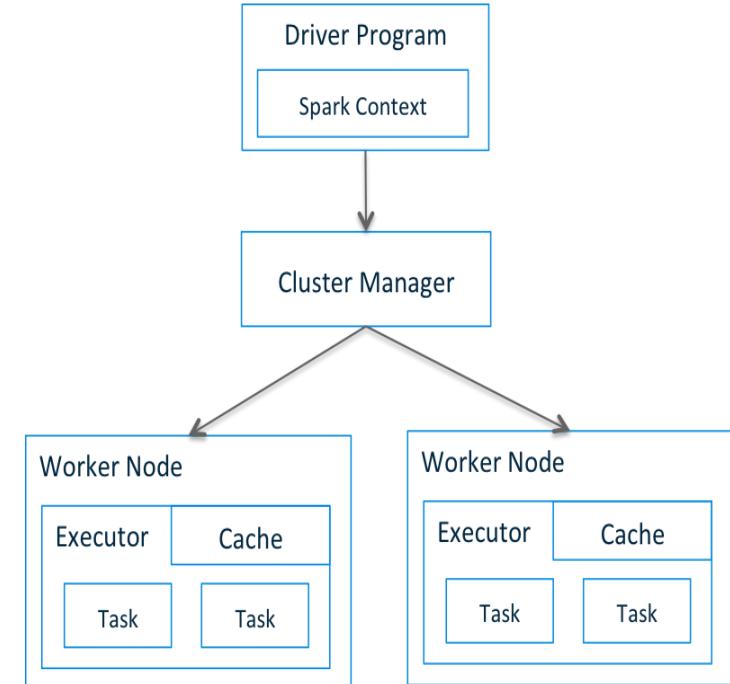
Overview: Spark Components

Spark is a fast, general purpose cluster computing platform.
Spark takes advantage of parallelism, by distributing processing across a cluster of nodes, in order to provide fast processing of data.



Each Spark application gets its own executor processes which stay up for the duration of the application. The executor runs tasks in multiple threads.

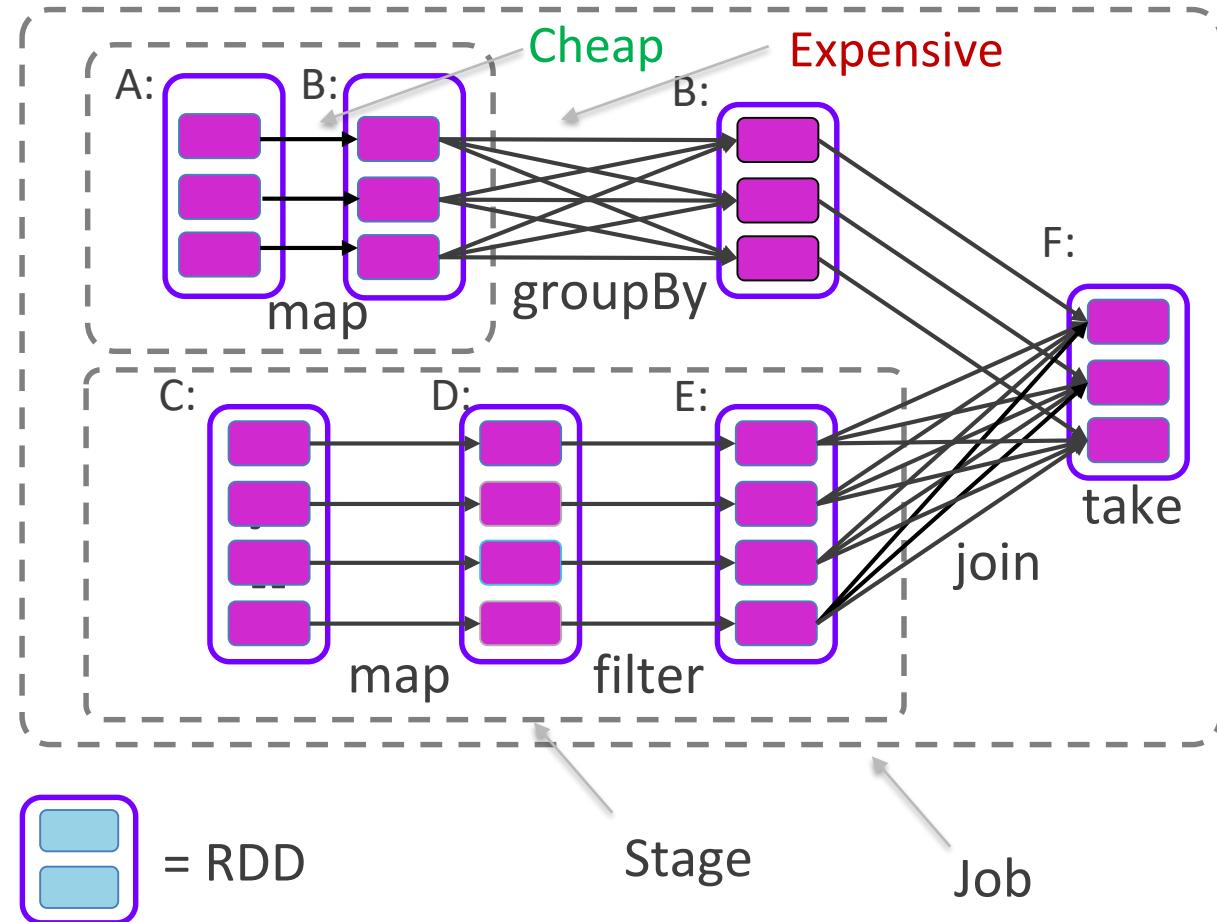
The driver program coordinates tasks and handles resource requests to the cluster manager. The driver distributes application code to each executor. Normally, each task takes 1 core.



Example: When Spark is run interactively via `pyspark` or `spark-shell` executors are assigned to the shell (driver) until the shell is exited. Each time the user invokes an action on the data the driver invokes that action on each executor as a task.

Another Quick Overview: Spark Execution

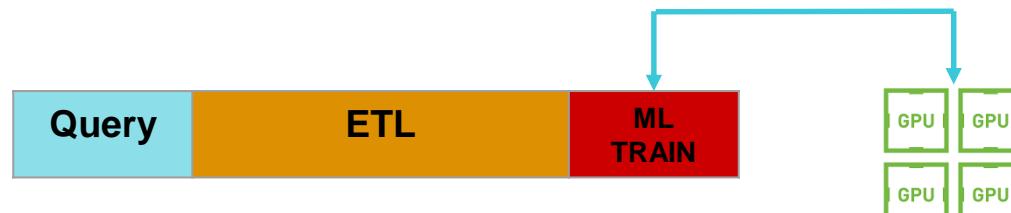
- Contrary to what some believe, Spark does not do everything in memory.
- Operations like `groupBy`, `join`, etc force a break in stages and shuffle data.
- Shuffles write data to disk.
- Shuffles can cause problems and also performance degradations
- Necessary Evil



Spark: Data Preparation Still on CPU's



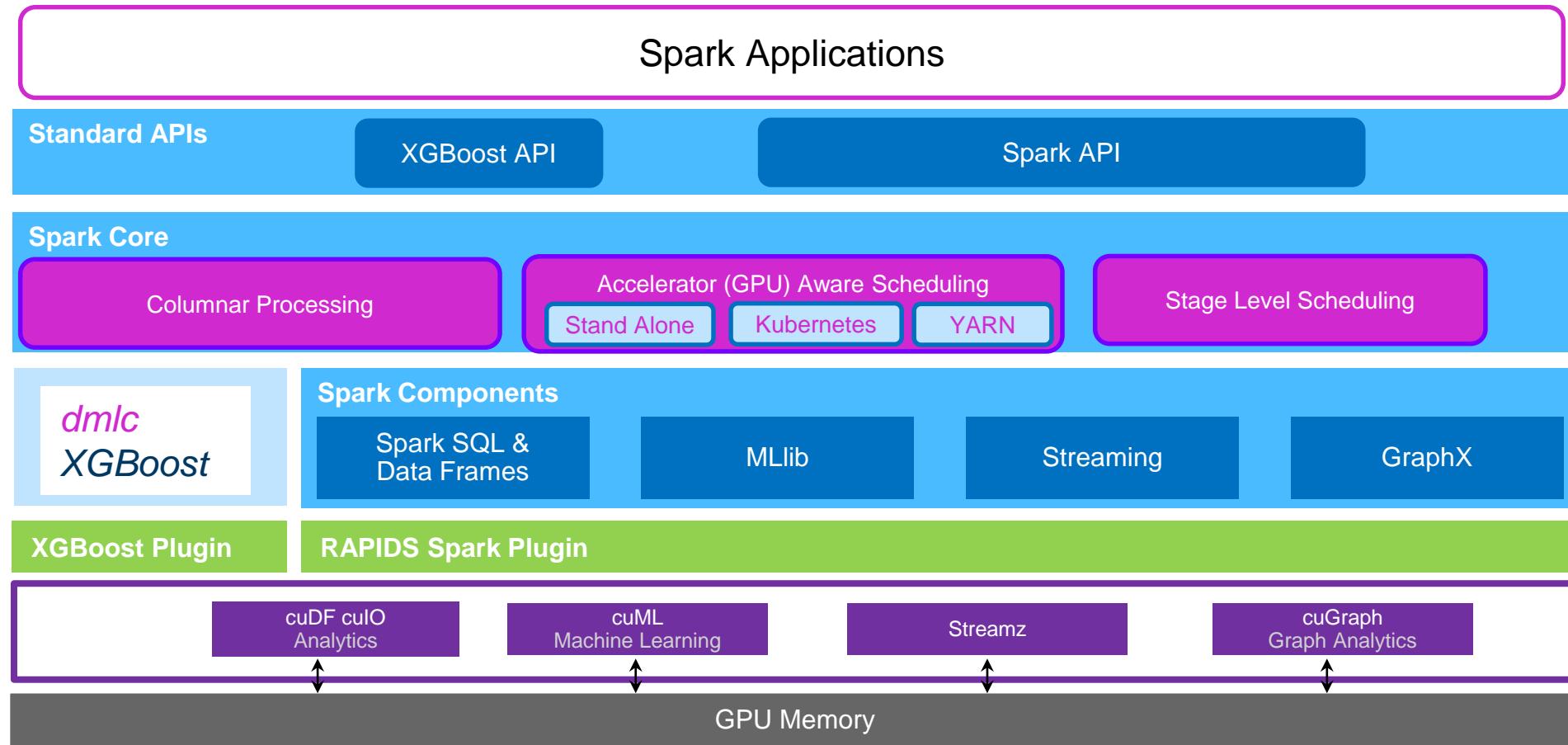
- Apache Spark – With CPU Only Setup



- Apache Spark – With GPU

Enterprise GPU users find it challenging to “Feed the Beast”.

Spark+GPU: Accelerated Data Science at Scale

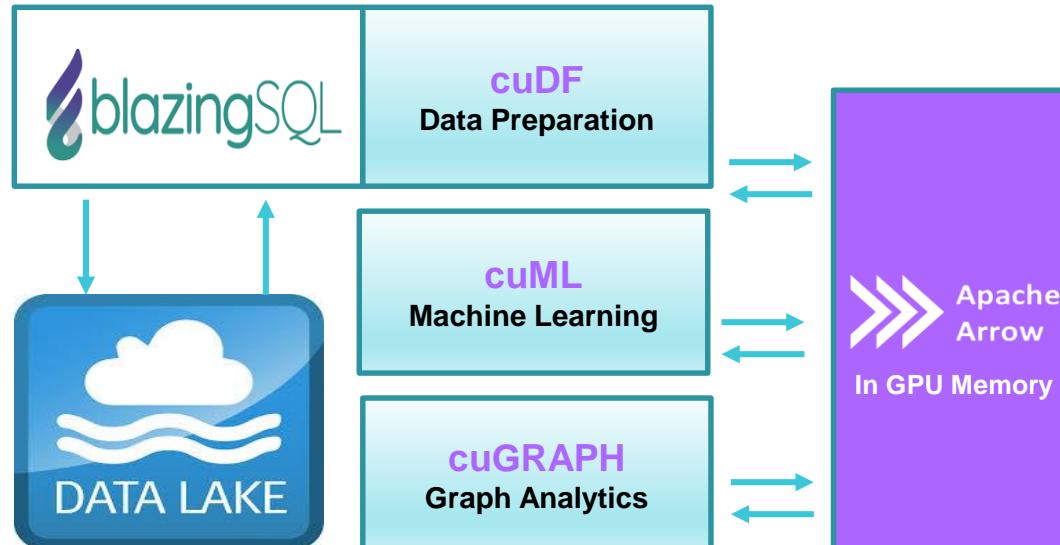


SQL Engines with RAPIDS

BlazingSQL

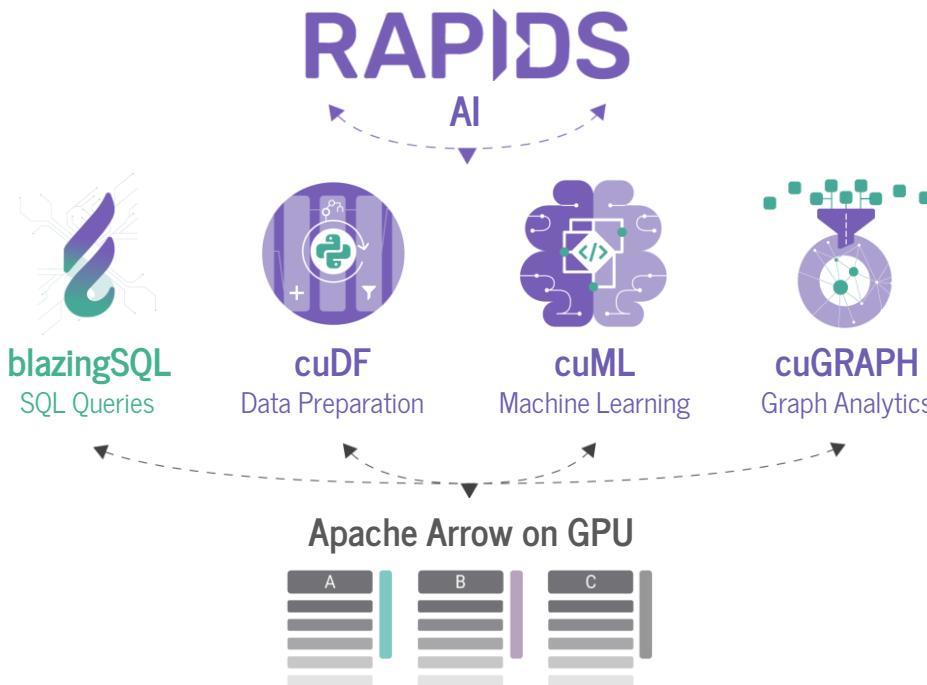


A **SQL** engine built on RAPIDS. Query **enterprise data lakes** lightning fast with full **interoperability** with RAPIDS stack.

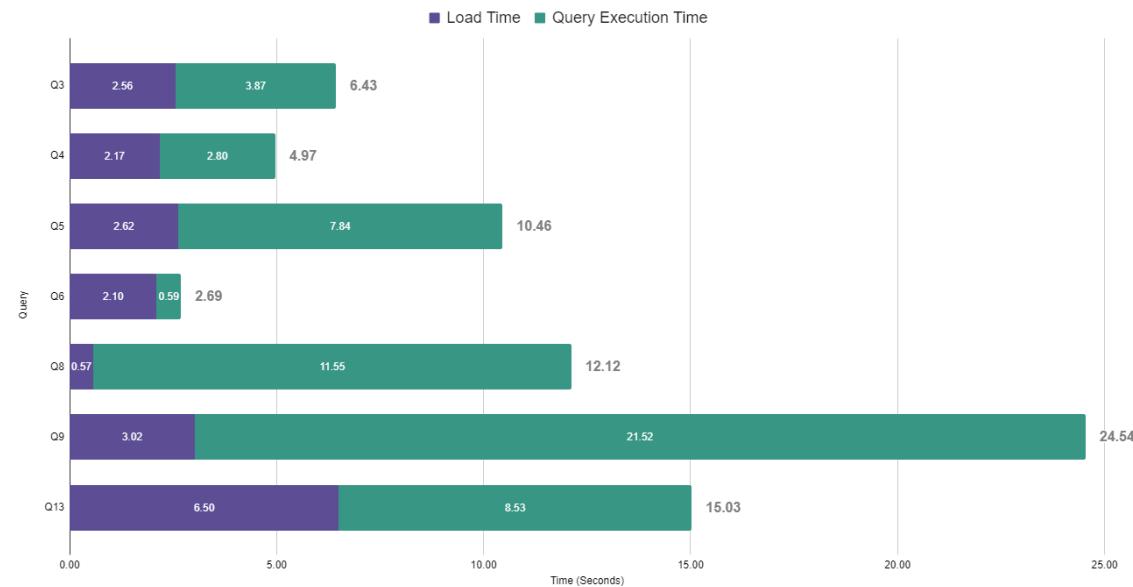


```
from blazingsql import BlazingContext  
  
bc = BlazingContext()  
  
#Register Filesystem  
bc.hdfs('data', host='129.13.0.12',  
port=54310)  
  
# Create Table  
bc.create_table('performance',  
file_type='parquet',  
path='hdfs://data/performance/')  
  
#Execute Query  
result_gdf = bc.run_query('SELECT * FROM  
performance WHERE  
YEAR(maturity_date)>2005')  
  
print(result_gdf)
```

BlazingSQL



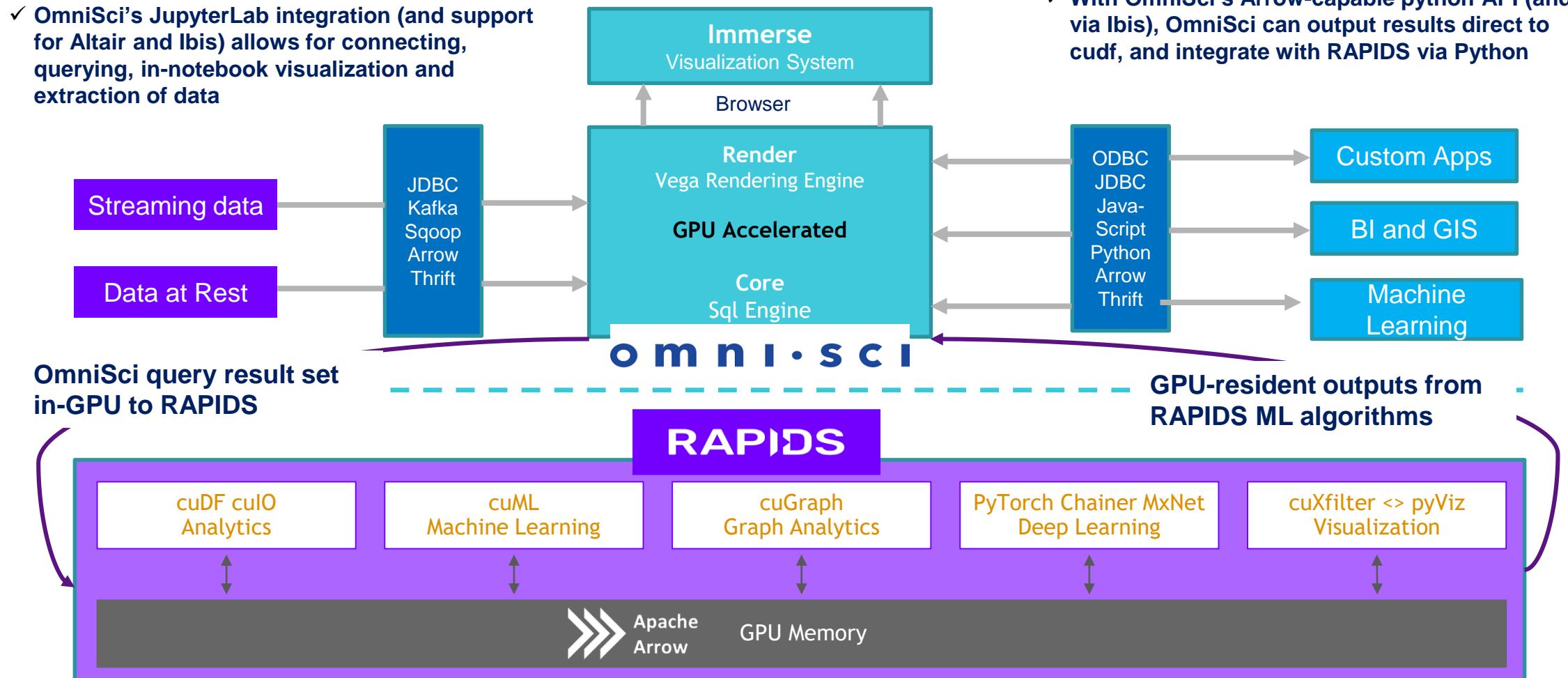
TPC-H SF100 Query Times - NVME Storage



SQL Engines with RAPIDS

OmniSci

- ✓ OmniSci's JupyterLab integration (and support for Altair and Ibis) allows for connecting, querying, in-notebook visualization and extraction of data



Community

Ecosystem Partners

CONTRIBUTORS



ADOPTERS

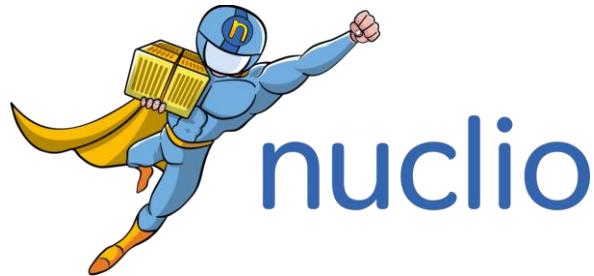


OPEN SOURCE



Building on top of RAPIDS

A bigger, better, stronger ecosystem for all



High-Performance
Serverless event and
data processing that
utilizes RAPIDS for GPU
Acceleration

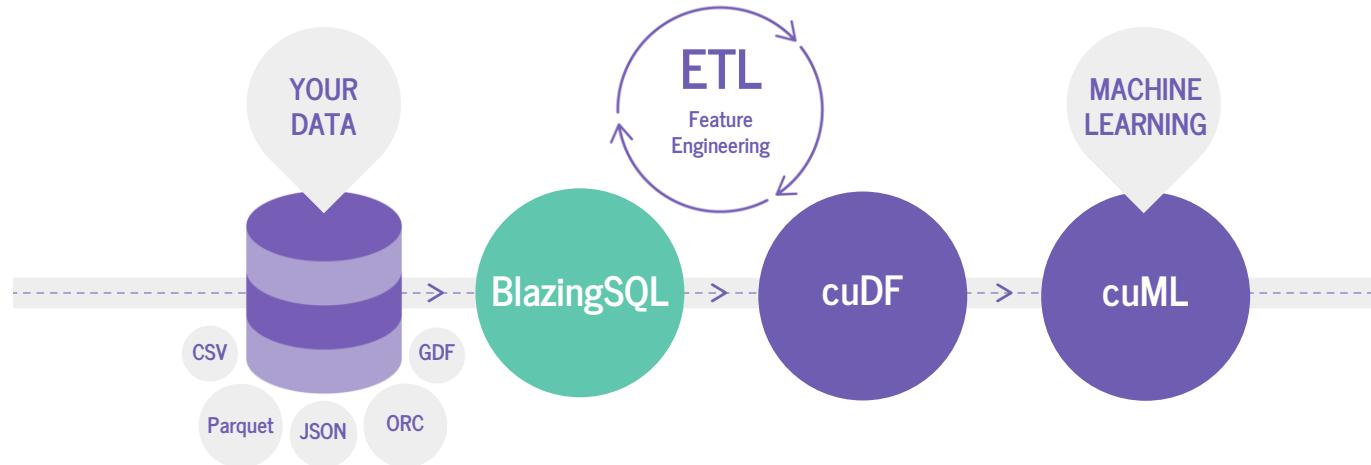


GPU accelerated SQL
engine built on top of
RAPIDS

Streamz

Distributed stream
processing using
RAPIDS and Dask

BlazingSQL



```
from blazingsql import BlazingContext
import cudf

bc = BlazingContext()

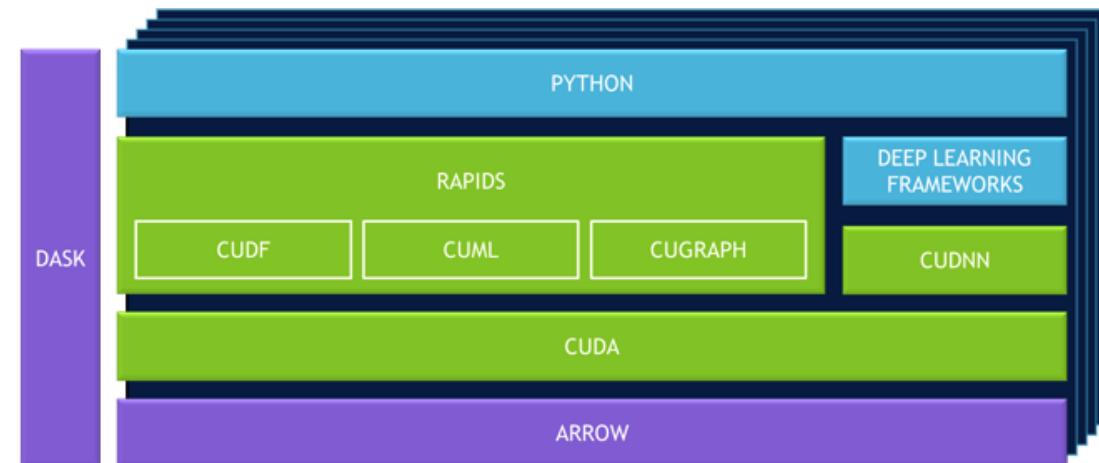
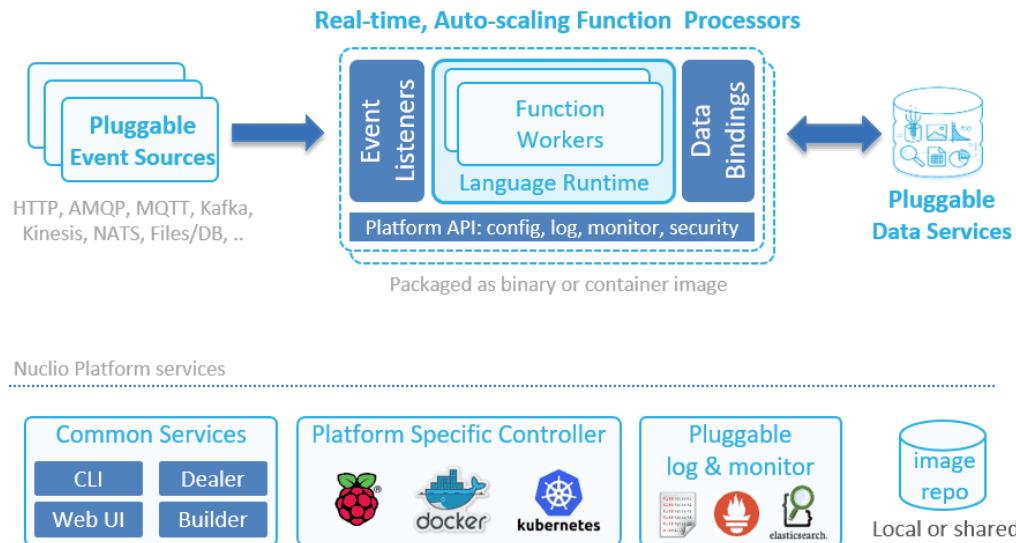
bc.s('bsql', bucket_name='bsql', access_key_id='<access_key>', secret_key='<secret_key>')

bc.create_table('orders', s3://bsql/orders/')

gdf = bc.sql('select * from orders').get()
```

RAPIDS + Nuclio

Serverless meets GPUs



<https://towardsdatascience.com/python-pandas-at-extreme-performance-912912b1047c>

Deploy RAPIDS Everywhere

Focused on robust functionality, deployment, and user experience



Google Cloud



Kubeflow



Azure



Azure Machine Learning



Cloud Dataproc



Amazon SageMaker



Alibaba Cloud

Integration with major cloud providers

Both containers and cloud specific machine instances
Support for Enterprise and HPC Orchestration Layers

5 Steps to getting started with RAPIDS

1. Install RAPIDS on using [Docker](#), [Conda](#), or [Colab](#)
2. Explore our [walk through videos](#), [blog content](#), our [github](#), the [tutorial notebooks](#), and our [examples workflows](#),
3. Build your own data science workflows.
4. Join our community conversations on [Slack](#), [Google](#), and [Twitter](#)
5. Contribute back. Don't forget to ask and answer questions on [Stack Overflow](#)

Easy Installation

Interactive Installation Guide

The screenshot shows the RAPIDS Release Selector tool. At the top, there's a purple header bar with the RAPIDS logo and navigation links: HOME, ABOUT, GET STARTED, COMMUNITY, BLOG, DOCS, and GITHUB. Below the header is a large title "RAPIDS RELEASE SELECTOR". A descriptive text block explains that RAPIDS is available as conda packages, docker images, and from source builds, and encourages users to use the tool to select their preferred method, packages, and environment. It notes that certain combinations may not be possible and are dimmed automatically. It also links to prerequisites and details.

The main interface consists of several dropdown menus and buttons:

- METHOD:** Conda (Preferred), Docker + Examples, Docker + Dev Env, Source
- RELEASE:** Stable (0.9), Nightly (0.10a)
- PACKAGES:** cuDF, cuML, cuGraph, All Packages
- LINUX:** Ubuntu 16.04, Ubuntu 18.04, CentOS 7
- PYTHON:** Python 3.6, Python 3.7
- CUDA:** CUDA 9.2, CUDA 10.0
- COMMAND:** A code editor containing the command: `conda install -c rapidsai -c nvidia -c numba -c conda-forge -c anaconda \ cudf=0.9 cuml=0.9 cugraph=0.9 python=3.6 anaconda::cudatoolkit=9.2`

At the bottom right of the command area is a "COPY COMMAND" button with a clipboard icon.

Explore: RAPIDS Github

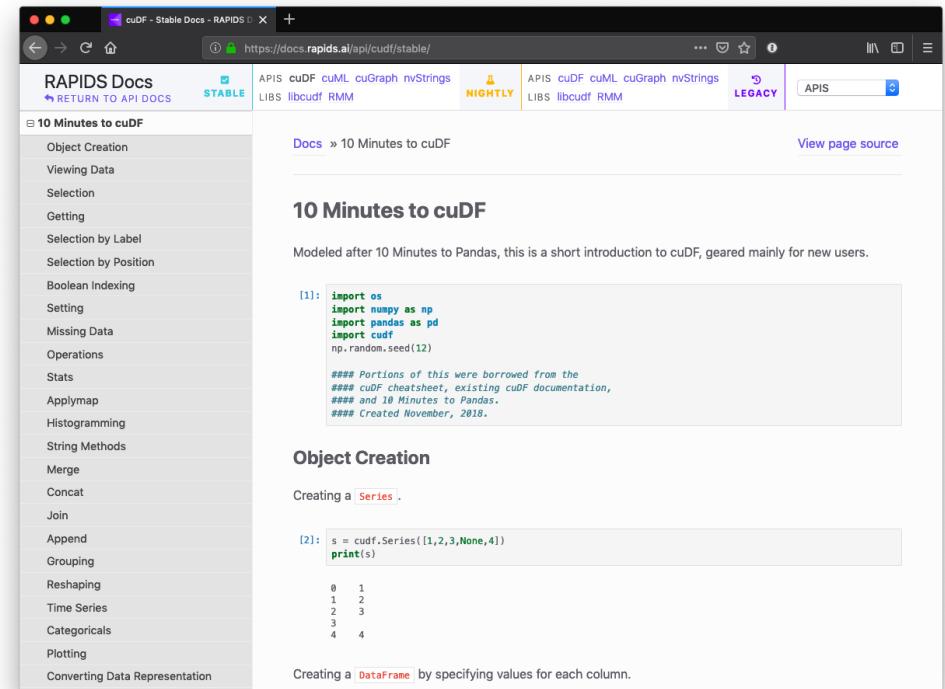
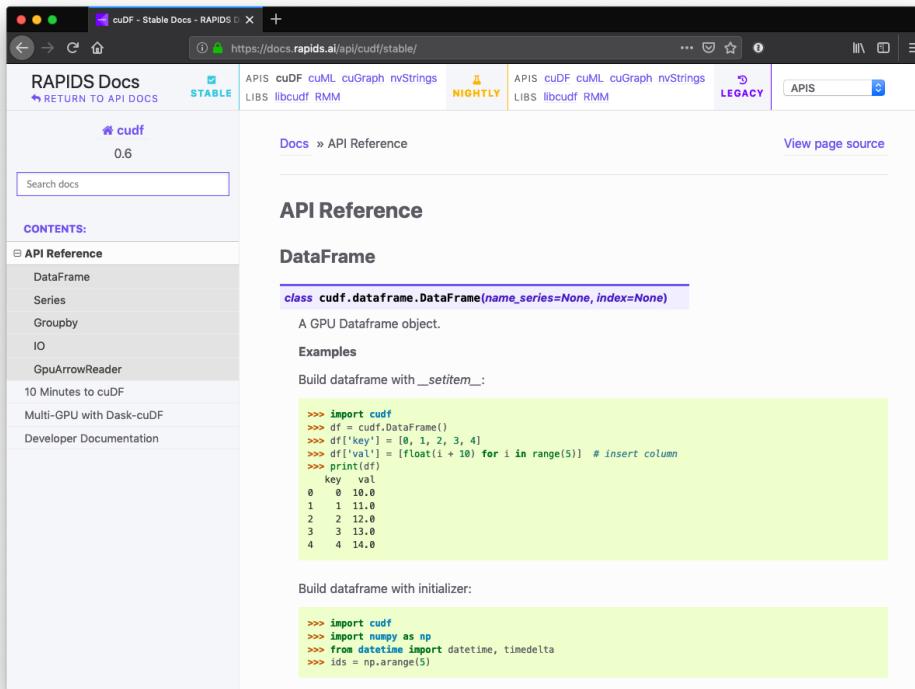
<https://github.com/rapidsai>

The screenshot shows the GitHub homepage for the RAPIDS repository. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. The repository card for RAPIDS displays its logo (a purple square with 'RAPIDS' in white), its name, a brief description 'Open GPU Data Science', and a link to its website (<http://rapids.ai>). Below the repository card, there are tabs for Repositories (67), Packages, People (118), Teams (91), and Projects (6). A section titled 'Pinned repositories' lists six projects:

- cudf**: cuDF - GPU DataFrame Library. Cuda, 1.9k stars, 270 forks.
- cuml**: cuML - RAPIDS Machine Learning Library. C++ (indicated by a pink dot), 665 stars, 119 forks.
- cugraph**: cuGraph - RAPIDS Graph Analytics Library. Cuda, 204 stars, 52 forks.
- notebooks**: RAPIDS Sample Notebooks. Jupyter Notebook, 204 stars, 94 forks.
- notebooks-contrib**: RAPIDS Community Notebooks. Jupyter Notebook, 106 stars, 76 forks.
- cuxfilter**: GPU accelerated cross filtering. Python, 31 stars, 14 forks.

Explore: RAPIDS Docs

Improved and easier to use!



<https://docs.rapids.ai>

Explore: RAPIDS Code and Blogs

Check out our code and how we use it

README.md

RAPIDS cuDF - GPU DataFrames

build running

NOTE: For the latest stable README.md ensure you are on the master branch.

Built based on the Apache Arrow columnar memory format, cuDF is a GPU DataFrame library for loading, joining, aggregating, filtering, and otherwise manipulating data.

cuDF provides a pandas-like API that will be familiar to data engineers & data scientists, so they can use it to easily accelerate their workflows without going into the details of CUDA programming.

For example, the following snippet downloads a CSV, then uses the GPU to parse it into rows and columns and run calculations:

```
import cudf, io, requests
from io import StringIO

url="https://github.com/plotly/datasets/raw/master/tips.csv"
content = requests.get(url).content.decode('utf-8')

tips_df = cudf.read_csv(StringIO(content))
tips_df['tip_percentage'] = tips_df['tip']/tips_df['total_bill']*100

# display average tip by dining party size
print(tips_df.groupby('size').tip_percentage.mean())

Output:
size
    tip_percentage
1      1.614333
2      1.666667
3      1.785714
4      2.000000
5      2.222222
6      2.444444
7      2.777778
8      3.111111
9      3.444444
10     3.777778
11     4.111111
12     4.444444
13     4.777778
14     5.111111
15     5.444444
16     5.777778
17     6.111111
18     6.444444
19     6.777778
20     7.111111
21     7.444444
22     7.777778
23     8.111111
24     8.444444
25     8.777778
26     9.111111
27     9.444444
28     9.777778
29     10.111111
30     10.444444
31     10.777778
32     11.111111
33     11.444444
34     11.777778
35     12.111111
36     12.444444
37     12.777778
38     13.111111
39     13.444444
40     13.777778
41     14.111111
42     14.444444
43     14.777778
44     15.111111
45     15.444444
46     15.777778
47     16.111111
48     16.444444
49     16.777778
50     17.111111
51     17.444444
52     17.777778
53     18.111111
54     18.444444
55     18.777778
56     19.111111
57     19.444444
58     19.777778
59     20.111111
60     20.444444
61     20.777778
62     21.111111
63     21.444444
64     21.777778
65     22.111111
66     22.444444
67     22.777778
68     23.111111
69     23.444444
70     23.777778
71     24.111111
72     24.444444
73     24.777778
74     25.111111
75     25.444444
76     25.777778
77     26.111111
78     26.444444
79     26.777778
80     27.111111
81     27.444444
82     27.777778
83     28.111111
84     28.444444
85     28.777778
86     29.111111
87     29.444444
88     29.777778
89     30.111111
90     30.444444
91     30.777778
92     31.111111
93     31.444444
94     31.777778
95     32.111111
96     32.444444
97     32.777778
98     33.111111
99     33.444444
100    33.777778
101    34.111111
102    34.444444
103    34.777778
104    35.111111
105    35.444444
106    35.777778
107    36.111111
108    36.444444
109    36.777778
110    37.111111
111    37.444444
112    37.777778
113    38.111111
114    38.444444
115    38.777778
116    39.111111
117    39.444444
118    39.777778
119    40.111111
120    40.444444
121    40.777778
122    41.111111
123    41.444444
124    41.777778
125    42.111111
126    42.444444
127    42.777778
128    43.111111
129    43.444444
130    43.777778
131    44.111111
132    44.444444
133    44.777778
134    45.111111
135    45.444444
136    45.777778
137    46.111111
138    46.444444
139    46.777778
140    47.111111
141    47.444444
142    47.777778
143    48.111111
144    48.444444
145    48.777778
146    49.111111
147    49.444444
148    49.777778
149    50.111111
150    50.444444
151    50.777778
152    51.111111
153    51.444444
154    51.777778
155    52.111111
156    52.444444
157    52.777778
158    53.111111
159    53.444444
160    53.777778
161    54.111111
162    54.444444
163    54.777778
164    55.111111
165    55.444444
166    55.777778
167    56.111111
168    56.444444
169    56.777778
170    57.111111
171    57.444444
172    57.777778
173    58.111111
174    58.444444
175    58.777778
176    59.111111
177    59.444444
178    59.777778
179    60.111111
180    60.444444
181    60.777778
182    61.111111
183    61.444444
184    61.777778
185    62.111111
186    62.444444
187    62.777778
188    63.111111
189    63.444444
190    63.777778
191    64.111111
192    64.444444
193    64.777778
194    65.111111
195    65.444444
196    65.777778
197    66.111111
198    66.444444
199    66.777778
200    67.111111
201    67.444444
202    67.777778
203    68.111111
204    68.444444
205    68.777778
206    69.111111
207    69.444444
208    69.777778
209    70.111111
210    70.444444
211    70.777778
212    71.111111
213    71.444444
214    71.777778
215    72.111111
216    72.444444
217    72.777778
218    73.111111
219    73.444444
220    73.777778
221    74.111111
222    74.444444
223    74.777778
224    75.111111
225    75.444444
226    75.777778
227    76.111111
228    76.444444
229    76.777778
230    77.111111
231    77.444444
232    77.777778
233    78.111111
234    78.444444
235    78.777778
236    79.111111
237    79.444444
238    79.777778
239    80.111111
240    80.444444
241    80.777778
242    81.111111
243    81.444444
244    81.777778
245    82.111111
246    82.444444
247    82.777778
248    83.111111
249    83.444444
250    83.777778
251    84.111111
252    84.444444
253    84.777778
254    85.111111
255    85.444444
256    85.777778
257    86.111111
258    86.444444
259    86.777778
260    87.111111
261    87.444444
262    87.777778
263    88.111111
264    88.444444
265    88.777778
266    89.111111
267    89.444444
268    89.777778
269    90.111111
270    90.444444
271    90.777778
272    91.111111
273    91.444444
274    91.777778
275    92.111111
276    92.444444
277    92.777778
278    93.111111
279    93.444444
280    93.777778
281    94.111111
282    94.444444
283    94.777778
284    95.111111
285    95.444444
286    95.777778
287    96.111111
288    96.444444
289    96.777778
290    97.111111
291    97.444444
292    97.777778
293    98.111111
294    98.444444
295    98.777778
296    99.111111
297    99.444444
298    99.777778
299    100.111111
300    100.444444
301    100.777778
302    101.111111
303    101.444444
304    101.777778
305    102.111111
306    102.444444
307    102.777778
308    103.111111
309    103.444444
310    103.777778
311    104.111111
312    104.444444
313    104.777778
314    105.111111
315    105.444444
316    105.777778
317    106.111111
318    106.444444
319    106.777778
320    107.111111
321    107.444444
322    107.777778
323    108.111111
324    108.444444
325    108.777778
326    109.111111
327    109.444444
328    109.777778
329    110.111111
330    110.444444
331    110.777778
332    111.111111
333    111.444444
334    111.777778
335    112.111111
336    112.444444
337    112.777778
338    113.111111
339    113.444444
340    113.777778
341    114.111111
342    114.444444
343    114.777778
344    115.111111
345    115.444444
346    115.777778
347    116.111111
348    116.444444
349    116.777778
350    117.111111
351    117.444444
352    117.777778
353    118.111111
354    118.444444
355    118.777778
356    119.111111
357    119.444444
358    119.777778
359    120.111111
360    120.444444
361    120.777778
362    121.111111
363    121.444444
364    121.777778
365    122.111111
366    122.444444
367    122.777778
368    123.111111
369    123.444444
370    123.777778
371    124.111111
372    124.444444
373    124.777778
374    125.111111
375    125.444444
376    125.777778
377    126.111111
378    126.444444
379    126.777778
380    127.111111
381    127.444444
382    127.777778
383    128.111111
384    128.444444
385    128.777778
386    129.111111
387    129.444444
388    129.777778
389    130.111111
390    130.444444
391    130.777778
392    131.111111
393    131.444444
394    131.777778
395    132.111111
396    132.444444
397    132.777778
398    133.111111
399    133.444444
400    133.777778
401    134.111111
402    134.444444
403    134.777778
404    135.111111
405    135.444444
406    135.777778
407    136.111111
408    136.444444
409    136.777778
410    137.111111
411    137.444444
412    137.777778
413    138.111111
414    138.444444
415    138.777778
416    139.111111
417    139.444444
418    139.777778
419    140.111111
420    140.444444
421    140.777778
422    141.111111
423    141.444444
424    141.777778
425    142.111111
426    142.444444
427    142.777778
428    143.111111
429    143.444444
430    143.777778
431    144.111111
432    144.444444
433    144.777778
434    145.111111
435    145.444444
436    145.777778
437    146.111111
438    146.444444
439    146.777778
440    147.111111
441    147.444444
442    147.777778
443    148.111111
444    148.444444
445    148.777778
446    149.111111
447    149.444444
448    149.777778
449    150.111111
450    150.444444
451    150.777778
452    151.111111
453    151.444444
454    151.777778
455    152.111111
456    152.444444
457    152.777778
458    153.111111
459    153.444444
460    153.777778
461    154.111111
462    154.444444
463    154.777778
464    155.111111
465    155.444444
466    155.777778
467    156.111111
468    156.444444
469    156.777778
470    157.111111
471    157.444444
472    157.777778
473    158.111111
474    158.444444
475    158.777778
476    159.111111
477    159.444444
478    159.777778
479    160.111111
480    160.444444
481    160.777778
482    161.111111
483    161.444444
484    161.777778
485    162.111111
486    162.444444
487    162.777778
488    163.111111
489    163.444444
490    163.777778
491    164.111111
492    164.444444
493    164.777778
494    165.111111
495    165.444444
496    165.777778
497    166.111111
498    166.444444
499    166.777778
500    167.111111
501    167.444444
502    167.777778
503    168.111111
504    168.444444
505    168.777778
506    169.111111
507    169.444444
508    169.777778
509    170.111111
510    170.444444
511    170.777778
512    171.111111
513    171.444444
514    171.777778
515    172.111111
516    172.444444
517    172.777778
518    173.111111
519    173.444444
520    173.777778
521    174.111111
522    174.444444
523    174.777778
524    175.111111
525    175.444444
526    175.777778
527    176.111111
528    176.444444
529    176.777778
530    177.111111
531    177.444444
532    177.777778
533    178.111111
534    178.444444
535    178.777778
536    179.111111
537    179.444444
538    179.777778
539    180.111111
540    180.444444
541    180.777778
542    181.111111
543    181.444444
544    181.777778
545    182.111111
546    182.444444
547    182.777778
548    183.111111
549    183.444444
550    183.777778
551    184.111111
552    184.444444
553    184.777778
554    185.111111
555    185.444444
556    185.777778
557    186.111111
558    186.444444
559    186.777778
560    187.111111
561    187.444444
562    187.777778
563    188.111111
564    188.444444
565    188.777778
566    189.111111
567    189.444444
568    189.777778
569    190.111111
570    190.444444
571    190.777778
572    191.111111
573    191.444444
574    191.777778
575    192.111111
576    192.444444
577    192.777778
578    193.111111
579    193.444444
580    193.777778
581    194.111111
582    194.444444
583    194.777778
584    195.111111
585    195.444444
586    195.777778
587    196.111111
588    196.444444
589    196.777778
590    197.111111
591    197.444444
592    197.777778
593    198.111111
594    198.444444
595    198.777778
596    199.111111
597    199.444444
598    199.777778
599    200.111111
600    200.444444
601    200.777778
602    201.111111
603    201.444444
604    201.777778
605    202.111111
606    202.444444
607    202.777778
608    203.111111
609    203.444444
610    203.777778
611    204.111111
612    204.444444
613    204.777778
614    205.111111
615    205.444444
616    205.777778
617    206.111111
618    206.444444
619    206.777778
620    207.111111
621    207.444444
622    207.777778
623    208.111111
624    208.444444
625    208.777778
626    209.111111
627    209.444444
628    209.777778
629    210.111111
630    210.444444
631    210.777778
632    211.111111
633    211.444444
634    211.777778
635    212.111111
636    212.444444
637    212.777778
638    213.111111
639    213.444444
640    213.777778
641    214.111111
642    214.444444
643    214.777778
644    215.111111
645    215.444444
646    215.777778
647    216.111111
648    216.444444
649    216.777778
650    217.111111
651    217.444444
652    217.777778
653    218.111111
654    218.444444
655    218.777778
656    219.111111
657    219.444444
658    219.777778
659    220.111111
660    220.444444
661    220.777778
662    221.111111
663    221.444444
664    221.777778
665    222.111111
666    222.444444
667    222.777778
668    223.111111
669    223.444444
670    223.777778
671    224.111111
672    224.444444
673    224.777778
674    225.111111
675    225.444444
676    225.777778
677    226.111111
678    226.444444
679    226.777778
680    227.111111
681    227.444444
682    227.777778
683    228.111111
684    228.444444
685    228.777778
686    229.111111
687    229.444444
688    229.777778
689    230.111111
690    230.444444
691    230.777778
692    231.111111
693    231.444444
694    231.777778
695    232.111111
696    232.444444
697    232.777778
698    233.111111
699    233.444444
700    233.777778
701    234.111111
702    234.444444
703    234.777778
704    235.111111
705    235.444444
706    235.777778
707    236.111111
708    236.444444
709    236.777778
710    237.111111
711    237.444444
712    237.777778
713    238.111111
714    238.444444
715    238.777778
716    239.111111
717    239.444444
718    239.777778
719    240.111111
720    240.444444
721    240.777778
722    241.111111
723    241.444444
724    241.777778
725    242.111111
726    242.444444
727    242.777778
728    243.111111
729    243.444444
730    243.777778
731    244.111111
732    244.444444
733    244.777778
734    245.111111
735    245.444444
736    245.777778
737    246.111111
738    246.444444
739    246.777778
740    247.111111
741    247.444444
742    247.777778
743    248.111111
744    248.444444
745    248.777778
746    249.111111
747    249.444444
748    249.777778
749    250.111111
750    250.444444
751    250.777778
752    251.1
```

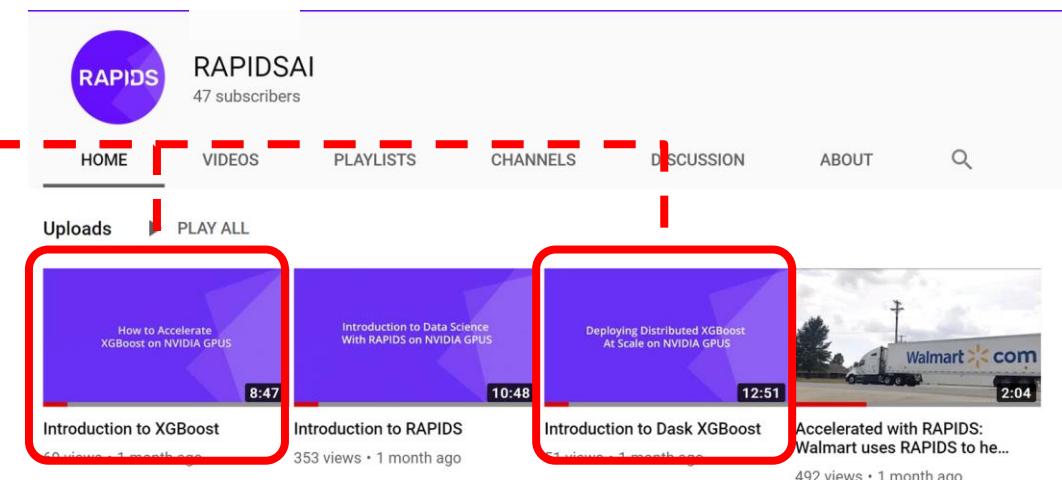
Explore: Notebooks Contrib

Notebooks Contrib Repo has tutorials and examples, and various E2E demos. RAPIDS Youtube channel has explanations, code walkthroughs and use cases.

intro_tutorials	05_Introduction_to_Dask_cuDF	This notebook shows how to work with cuDF DataFrames distributed across multiple GPUs using Dask.
intro_tutorials	06_Introduction_to_Supervised_Learning	This notebook shows how to do GPU accelerated Supervised Learning in RAPIDS.
intro_tutorials	07_Introduction_to_XGBoost	This notebook shows how to work with GPU accelerated XGBoost in RAPIDS.
intro_tutorials	08_Introduction_to_Dask_XGBoost	This notebook shows how to work with Dask XGBoost in RAPIDS.
intro_tutorials	09_Introduction_to_Dimensionality_Reduction	This notebook shows how to do GPU accelerated Dimensionality Reduction in RAPIDS.
intro_tutorials	10_Introduction_to_Clustering	This notebook shows how to do GPU accelerated Clustering in RAPIDS.

Intermediate Notebooks:

Folder	Notebook Title	Description
examples	DBSCAN_Demo_FULL	This notebook shows how to use DBSCAN algorithm and its GPU accelerated implementation present in RAPIDS.
examples	Dask_with_cuDF_and_XGBoost	In this notebook we show how to quickly setup Dask and train an XGBoost model using cuDF.



Join the Conversation



[Google Groups](#)



[Docker Hub](#)



[Slack Channel](#)



[Stack Overflow](#)

Contribute Back

Issues, feature requests, PRs, Blogs, Tutorials, Videos, QA...bring your best!

The screenshot shows four GitHub repository cards:

- cuml**: cuML - RAPIDS Machine Learning Library. Tags: machine-learning, gpu, machine-learning-algorithms, cuda, nvidia. Languages: C++, Apache-2.0. Stars: 608. Issues: 186 (26 need help). Updated 9 minutes ago.
- cuDF**: cuDF - GPU DataFrame Library. Tags: anaconda, gpu, arrow, machine-learning-algorithms, h2o, cuda, pandas. Languages: Cuda, Apache-2.0. Stars: 1,699. Issues: 325 (6 need help). Updated 31 minutes ago.
- notebooks-contrib**: RAPIDS Community Notebooks. Tags: Jupyter Notebook, Apache-2.0. Stars: 70. Issues: 10 (1 needs help). Updated 40 minutes ago.
- cugraph**: cuGraph. Tags: Cuda, Apache-2.0. Stars: 172. Issues: 58 (1 needs help). Updated 1 hour ago.

TECH BLOG Walmart Labs | ENGINEERING | DATA SCIENCE | INFOSEC | UX DESIGN | LEADERSHIP | ABOUT

How GPU Computing literally saved me at work?

Python+GPU = Power, 2 Days to 20 seconds

Abhishek Mungoli [Follow](#) May 9 · 9 min read

Getting Started with cuDF (RAPIDS)

Darren Ramsook [Follow](#) Jun 9 · 3 min read

John Murray @MurrayData [Follow](#)

Comparison CPU vs GPU @rapidsai to project 100 million x,y points to lat/lon to 0.01mm accuracy. CPU 1 core c 65 mins, multicore c 13 mins, GPU #RAPIDS 2 seconds. I optimised the code since previous run. Dell T7910 Xeon E5-2640V4x2/NVIDIA Titan Xp cc @NvidiaAI @marc_stampfli

```
john@plato:~/Source/Python/misc$ python crs_test.py
Generating Data
CPU Iterative
4005.0377202 seconds
CPU mapped
3957.19386101 seconds
CPU multiprocessing
788.550751209 seconds
GPU Rapids
2.103230476 seconds
```

Getting Started

RAPIDS Docs

New, improved, and easier to use

The screenshot shows a web browser window for the cuDF - Stable Docs - RAPIDS Docs API Reference. The URL is <https://docs.rapids.ai/api/cudf/stable/>. The page title is "cuDF - Stable Docs - RAPIDS Docs". The top navigation bar includes links for "APIS", "cuDF", "cuML", "cuGraph", "nvStrings", "LIBS", "libcudf", and "RMM". Below this, there are tabs for "STABLE" (selected), "NIGHTLY", and "LEGACY". A dropdown menu for "APIS" is open. On the left, a sidebar titled "RAPIDS Docs" has a "RETURN TO API DOCS" link. It contains a search bar and a "CONTENTS:" section with a tree view of API Reference, including "DataFrame", "Series", "Groupby", "IO", "GpuArrowReader", "10 Minutes to cuDF", "Multi-GPU with Dask-cuDF", and "Developer Documentation". The main content area is titled "API Reference" and "DataFrame". It defines the `cudf.dataframe.DataFrame` class as a GPU Dataframe object. It includes examples for building a dataframe with `__setitem__` and an initializer. The code examples are highlighted in green.

API Reference

DataFrame

`class cudf.dataframe.DataFrame(name_series=None, index=None)`

A GPU Dataframe object.

Examples

Build dataframe with `__setitem__`:

```
>>> import cudf
>>> df = cudf.DataFrame()
>>> df['key'] = [0, 1, 2, 3, 4]
>>> df['val'] = [float(i + 10) for i in range(5)] # insert column
>>> print(df)
   key    val
0    0  10.0
1    1  11.0
2    2  12.0
3    3  13.0
4    4  14.0
```

Build dataframe with initializer:

```
>>> import cudf
>>> import numpy as np
>>> from datetime import datetime, timedelta
>>> ids = np.arange(5)
```

<https://docs.rapids.ai>

RAPIDS Docs

Easier than ever to get started with cuDF

The screenshot shows a web browser window for the cuDF - Stable Docs - RAPIDS Docs website. The URL is https://docs.rapids.ai/api/cudf/stable/. The page title is "10 Minutes to cuDF". The left sidebar contains a table of contents for the guide, including sections like Object Creation, Viewing Data, Selection, Getting, Selection by Label, Selection by Position, Boolean Indexing, Setting, Missing Data, Operations, Stats, Applymap, Histogramming, String Methods, Merge, Concat, Join, Append, Grouping, Reshaping, Time Series, Categoricals, Plotting, and Converting Data Representation. The main content area starts with a heading "10 Minutes to cuDF" and a sub-section "Object Creation". It includes code snippets and explanatory text. The first code snippet is:

```
[1]: import os
import numpy as np
import pandas as pd
import cudf
np.random.seed(12)

### Portions of this were borrowed from the
### cuDF cheatsheet, existing cuDF documentation,
### and 10 Minutes to Pandas.
### Created November, 2018.
```

The second code snippet is:

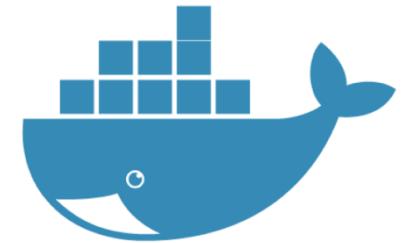
```
[2]: s = cudf.Series([1,2,3,None,4])
print(s)
```

0	1
1	2
2	3
3	
4	4

Below the code snippets, there is explanatory text: "Creating a DataFrame by specifying values for each column."

RAPIDS

How do I get the software?



- <https://github.com/rapidsai>
- <https://anaconda.org/rapidsai/>
- <https://ngc.nvidia.com/registry/nvidia-rapidsai-rapidsai>
- <https://hub.docker.com/r/rapidsai/rapidsai/>

Join the Movement

Everyone can help!



APACHE ARROW

<https://arrow.apache.org/>

@ApacheArrow



RAPIDS

<https://rapids.ai>

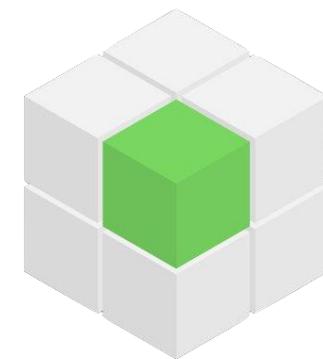
@RAPIDSAI



Dask

<https://dask.org>

@Dask_dev



GPU Open Analytics Initiative

<http://gpuopenanalytics.com/>

@GPUOAI

Integrations, feedback, documentation support, pull requests, new issues, or code donations welcomed!

THANK YOU

RAPIDS