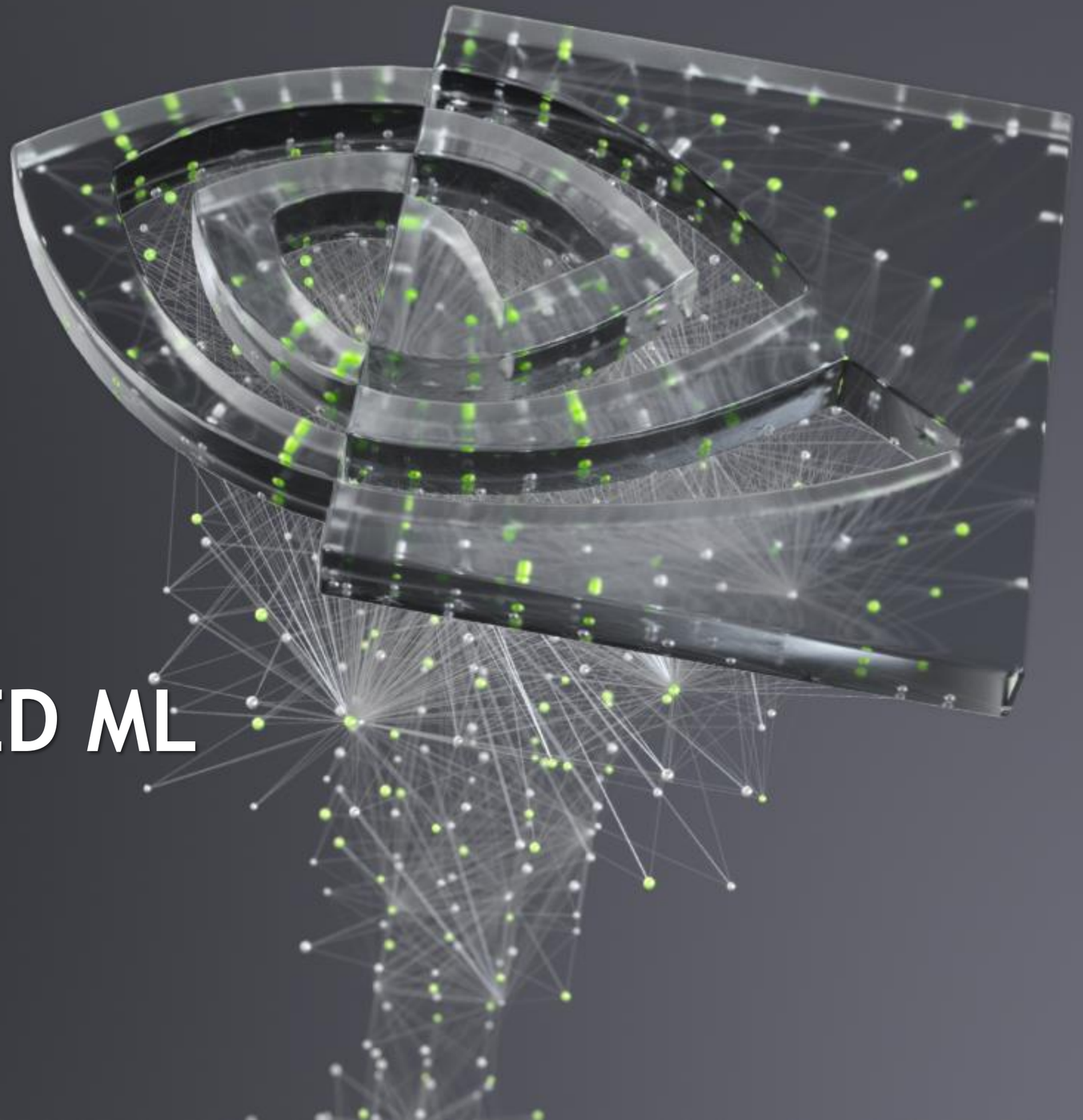


RAPIDS: GPU-POWERED ML

Miguel Martínez, Sr. Data Scientist at NVIDIA

Today's Agenda

- What is RAPIDS
- cuDF
- cuML
- cuGraph
- cuXFilter
- What is XGBoost
- Why XGBoost
- XGBoost + RAPIDS
- How to Start
- NVIDIA GPU Cloud
- Learn More
- Summary
- Q & A

What is RAPIDS

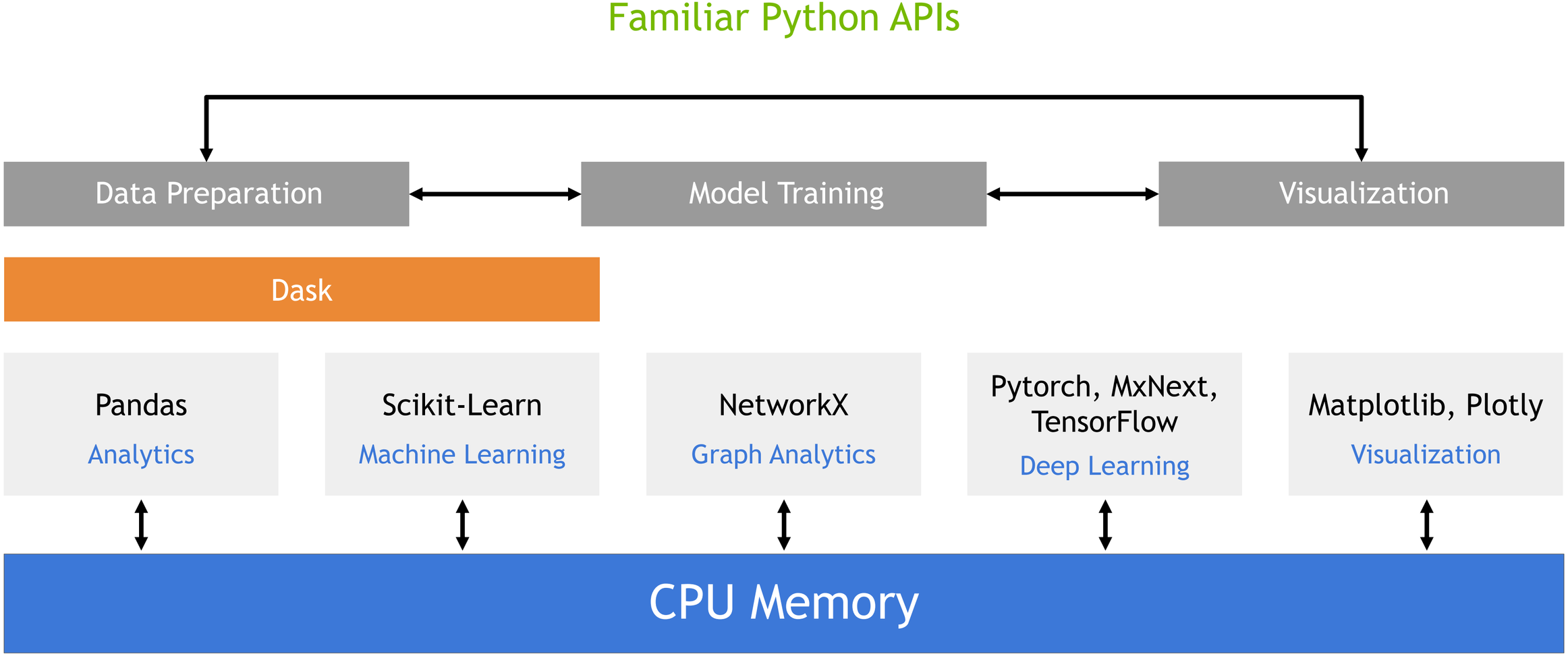
RAPIDS

GPU Accelerated Data Science

RAPIDS is a set of open-source software libraries which gives you the freedom to execute end-to-end data science and analytics pipelines entirely on GPUs.

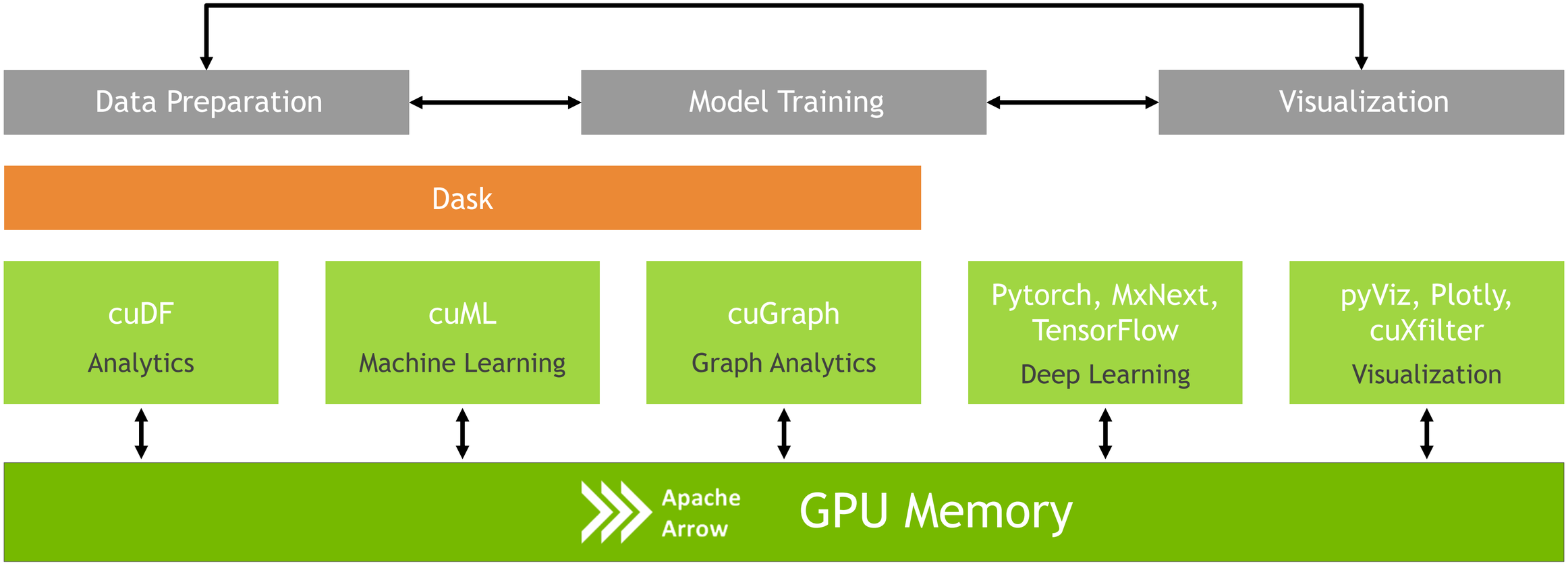
www.rapids.ai

Open Source Data Science Ecosystem



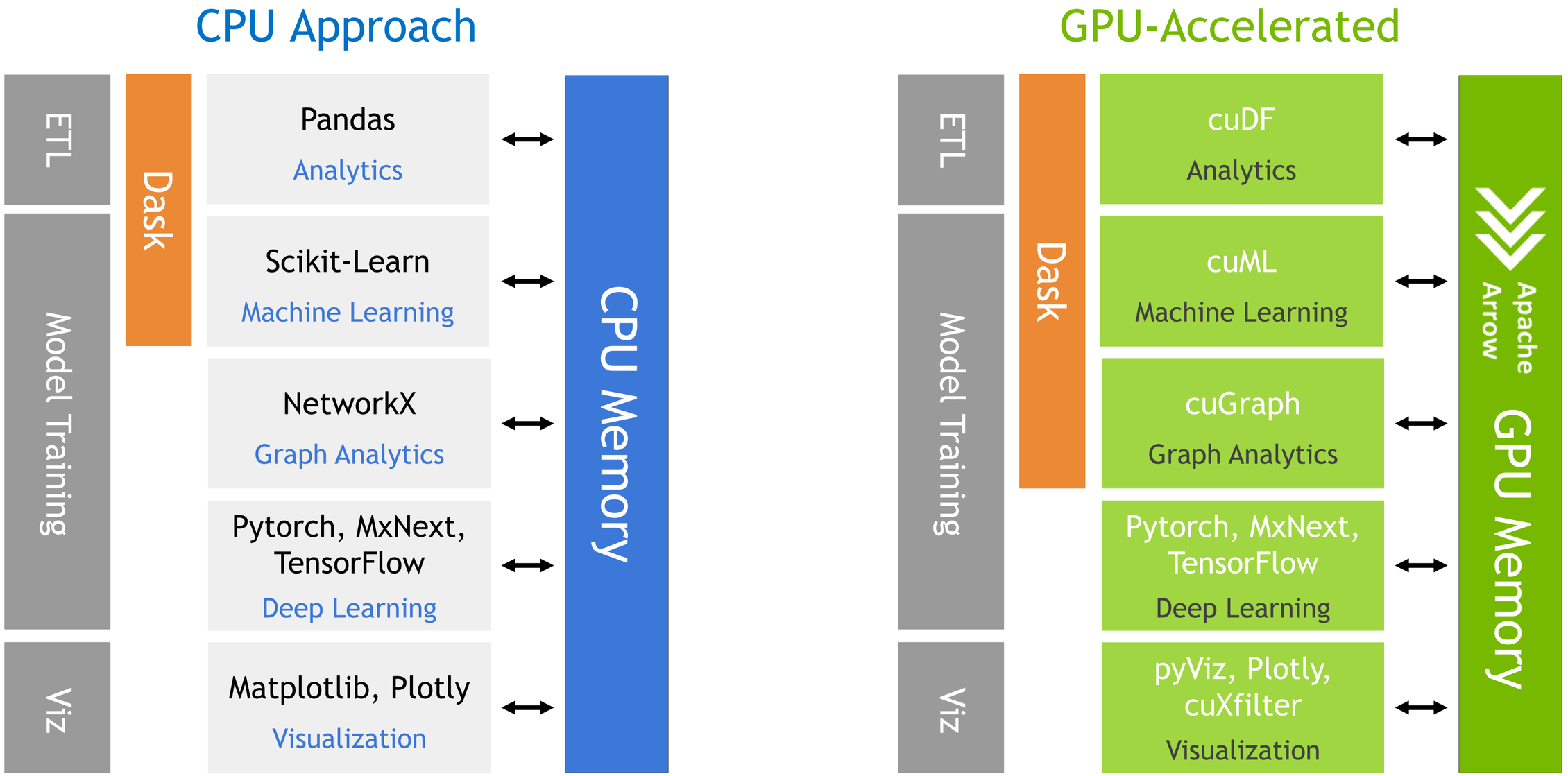
RAPIDS

End-to-End Accelerated GPU Data Science



Open Source Data Science Ecosystem

Same Familiar Python APIs – No/Low Code Change



EXTRACTION IS THE CORNERSTONE

cuDF for Faster Data Loading

- Follow Pandas APIs and provide >16x speedup:
  - CSV Reader/Writer
  - Parquet Reader/Writer
  - ORC Reader/Writer
  - JSON Reader
  - Avro Reader
- GPUDirect Storage integration in progress for bypassing PCIe bottlenecks!
- Key is GPU-accelerating both parsing and decompression wherever possible.

CPU CODE

```
import pandas
```

```
%%time
pdf = pandas.read_csv('nyc_taxi.csv')
```

```
CPU times: user 25.8 s, sys: 2.71 s, total: 28.5 s
Wall time: 28.5 s
```

GPU CODE

```
import cudf
```

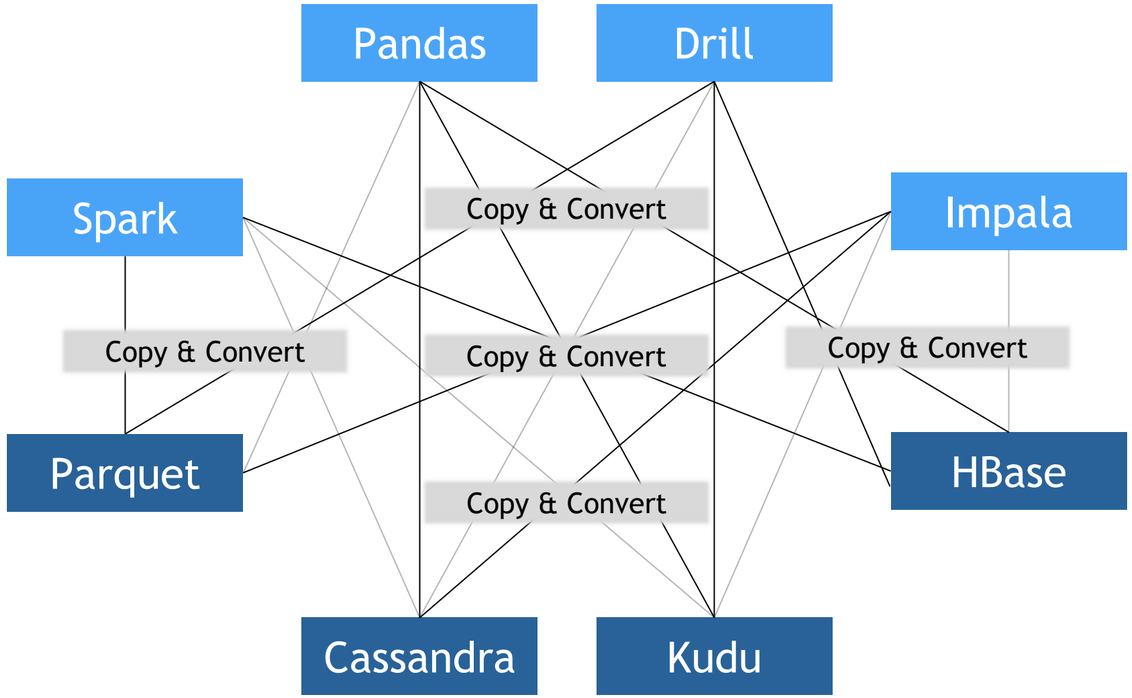
```
%%time
gdf = cudf.read_csv('nyc_taxi.csv')
```

```
CPU times: user 1.32 s, sys: 459 ms, total: 1.78 s
Wall time: 1.78 s
```

16 times speedup when loading a 2GB csv file.

DATA CONVERSION BOTTLENECK

Copy and convert is expensive



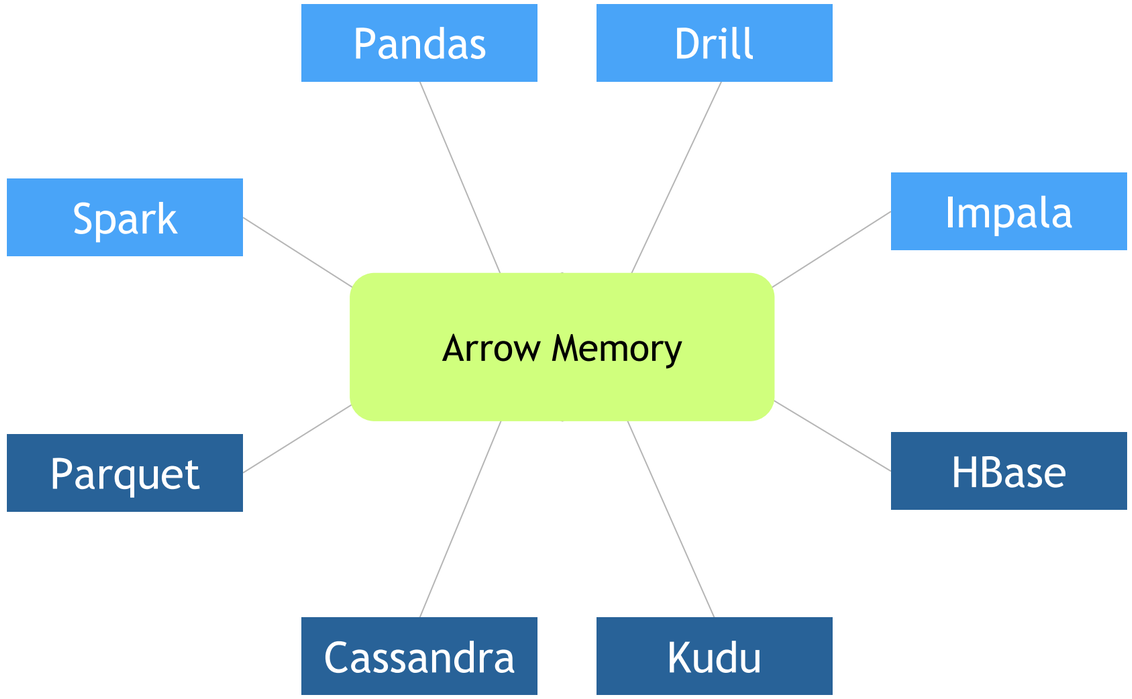
Each system has its own internal memory format.

Similar functionality implemented in multiple projects.

~70% computation wasted on serialization & deserialization.

DATA CONVERSION ~~BOTTLENECK~~

Learning from Apache Arrow



All systems utilize the same memory format.

Projects can share functionality.

No overhead for cross-system communication.

APACHE ARROW

A perfect match for data science

Columnar layout leverages GPU strengths.

CPU Single Instruction Multiple Data (SIMD) instructions also take advantage of it.

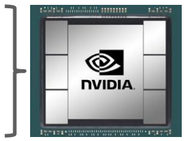
Consistency with CPU version simplifies development and conversion.

Emphasis on zero-copy and shallow-copy operations minimizes a core bottleneck.

	user_id	timestamp	source_ip
row_1	6939800	2021-04-12 05:31	42.155.123.142
row_2	4015666	2021-04-12 05:47	67.132.212.125
row_3	5456236	2021-04-12 05:52	81.205.129.121

	Row-wise data layout
row_1	6939800
	2021-04-12 05:31
	42.155.123.142
row_2	4015666
	2021-04-12 05:47
	67.132.212.125
row_3	5456236
	2021-04-12 05:52
	81.205.129.121

	Columnar data layout
user_id	6939800
	4015666
	5456236
timestamp	2021-04-12 05:31
	2021-04-12 05:47
	2021-04-12 05:52
source_ip	42.155.123.142
	67.132.212.125
	81.205.129.121



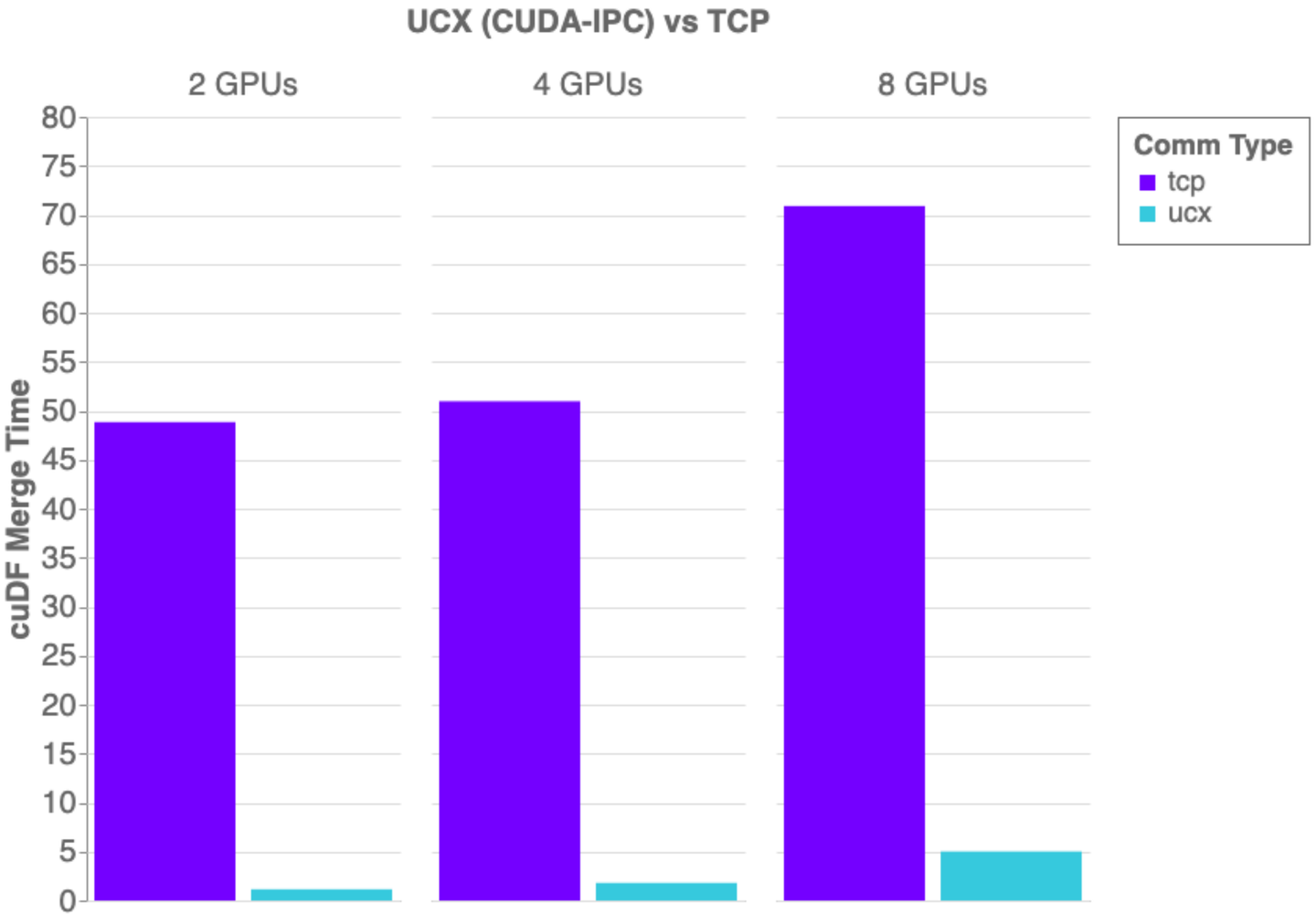
Why OpenUCX?

Bringing Hardware Accelerated Communications to Dask

- TCP sockets are slow!
- UCX provides uniform access to transports:
  - TCP, InfiniBand, Shared memory, NVLink
- Zero-copy GPU memory transfers over RDMA
- Provides best communication performance based on available hardware on nodes/cluster

BENCHMARKS

Distributed cuDF Random Merge



Benchmark Setup

- DataFrames:
  Left/Right 1x int64 column key column,
  1x int64 value columns.
- Inner Merge
- 30% of matching data balanced across each partition

Environment

- cuDF v0.11,
- UCX-PY 0.11
- Running on NVIDIA DGX-2:
- GPU NVIDIA Tesla V100 32GB
- CPU Intel(R) Xeon(R) CPU 8168 @ 2.70GHz

RAPIDS

cuDF

- GPU-accelerated data preparation and feature engineering
- Python drop-in Pandas replacement

cuML

- GPU-accelerated traditional machine learning libraries
- XGBoost, PCA, Kalman, K-means, k-NN, DBScan, tSVD…

cuGraph

- GPU-accelerated graph analytics libraries

cuXfilter

- Web Data Visualization library

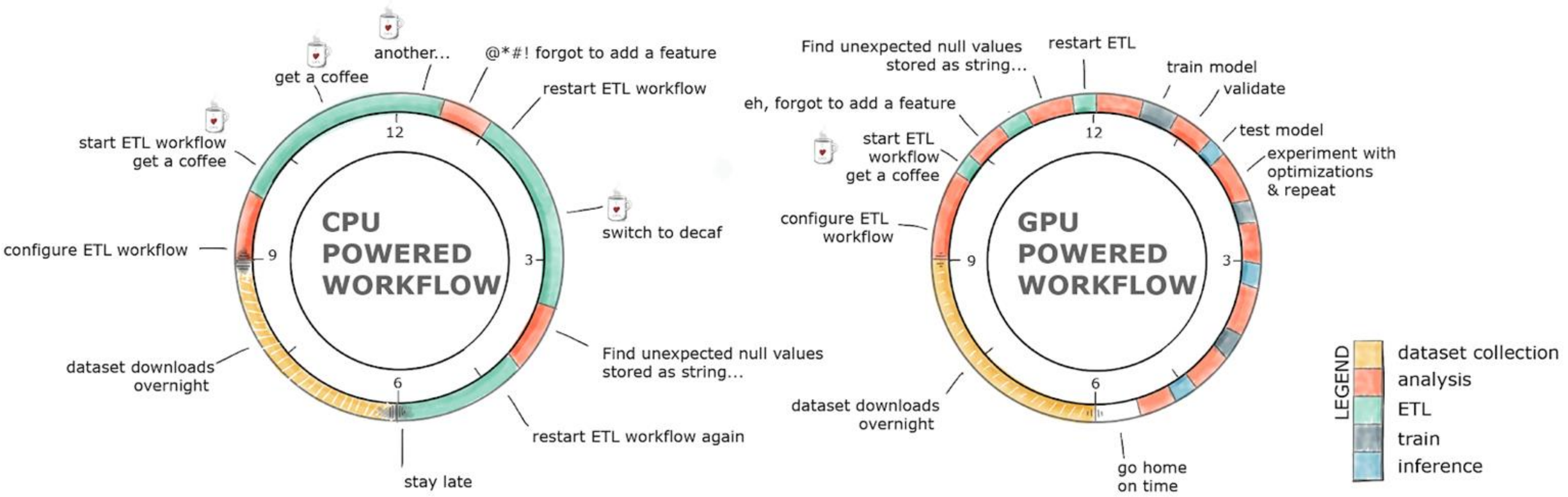
… and many others!

- cuSpatial, cuSignal, cuStreamz, …

cuDF

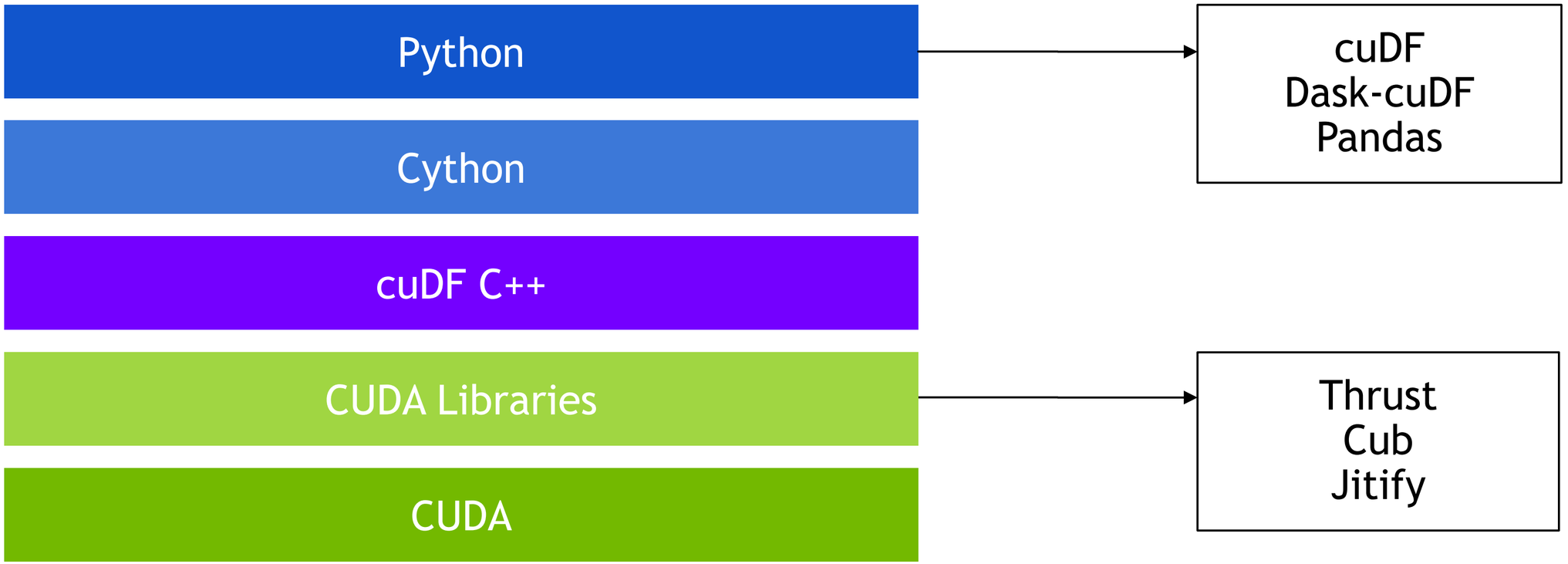
GPU-Accelerated ETL

The average data scientist spends up to 80% of their time in ETL, as opposed to training models



RAPIDS

ETL Technology Stack



EXTRACTION IS THE CORNERSTONE

cuDF for Faster Data Loading

- Follow Pandas APIs and provide >16x speedup:
  - CSV Reader/Writer
  - Parquet Reader/Writer
  - ORC Reader/Writer
  - JSON Reader
  - Avro Reader
- GPUDirect Storage integration in progress for bypassing PCIe bottlenecks!
- Key is GPU-accelerating both parsing and decompression wherever possible.

CPU CODE

```
import pandas
```

```
%%time
pdf = pandas.read_csv('nyc_taxi.csv')
```

```
CPU times: user 25.8 s, sys: 2.71 s, total: 28.5 s
Wall time: 28.5 s
```

GPU CODE

```
import cudf
```

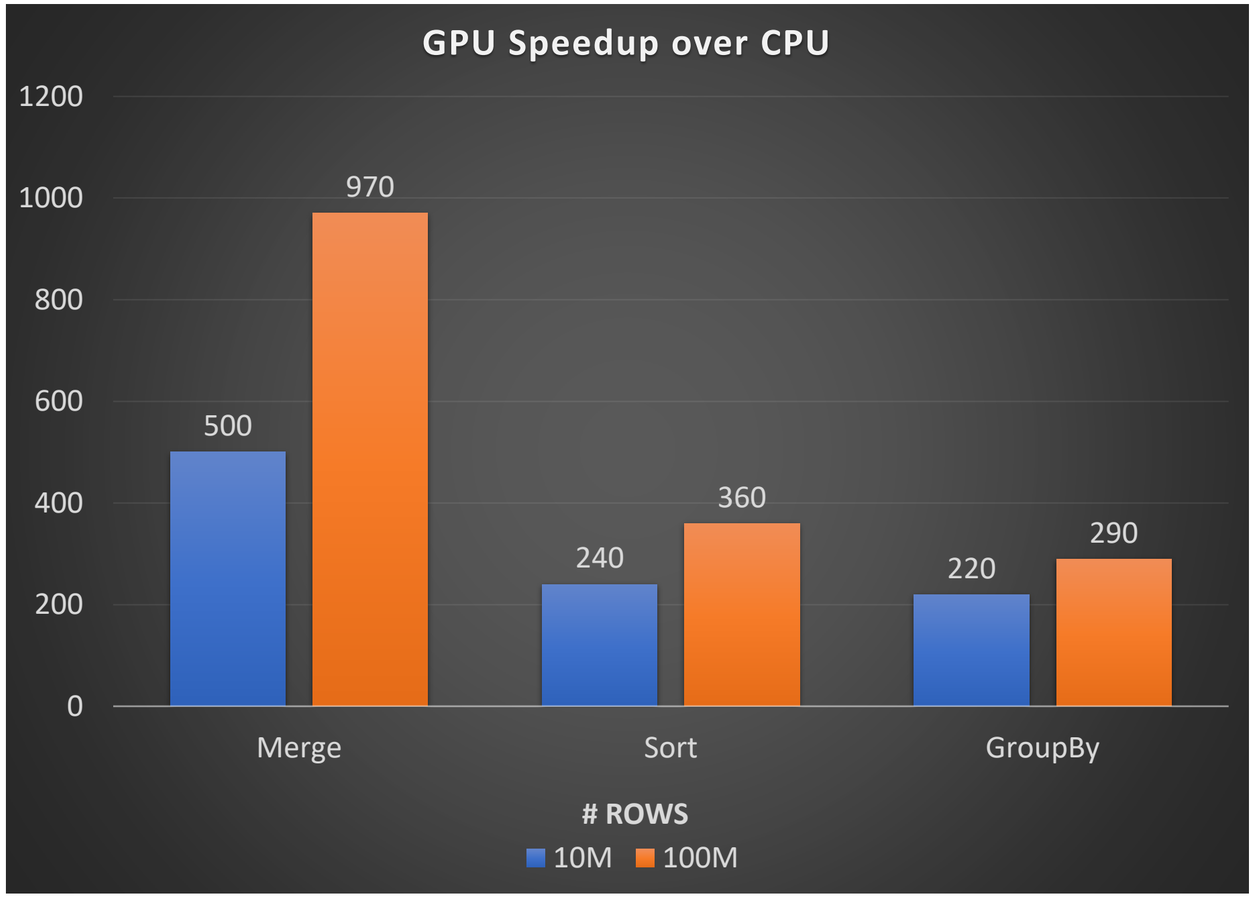
```
%%time
gdf = cudf.read_csv('nyc_taxi.csv')
```

```
CPU times: user 1.32 s, sys: 459 ms, total: 1.78 s
Wall time: 1.78 s
```

16 times speedup when loading a 2GB csv file.

BENCHMARKS

Single-GPU Speedup vs Pandas



Environment

- cuDF v0.13
- Pandas v0.25.3

- 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.

- Inner Merge

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

cuDF code examples

Loading data into a GPU DataFrame

```
import cudf

gdf = cudf.DataFrame()
gdf['my_column'] = [6, 7, 8]
```

Create an empty DataFrame, and add a column

```
import cudf

dataset = './apartments.csv'
gdf = cudf.read_csv(dataset, delimiter=';')
```

Load a CSV file into a GPU DataFrame

```
import cudf

gdf = cudf.DataFrame({'a': [1, 2, 3],
                      'b': [4, 5, 6],
                      'c': [7, 8, 9]})
```

Create a DataFrame with three columns

```
import pandas
import cudf

# Load a CSV file with Pandas
pdf = pandas.read_csv('./apartments.csv')

# Create a cuDF dataframe from Pandas
gdf = cudf.DataFrame.from_pandas(pdf)
```

Create a cuDF DataFrame from a Pandas DataFrame

cuDF code examples

Working with GPU DataFrames

```
head_values = gdf.head(3)
```

Create a new DataFrame with the first three rows

```
print(gdf.loc[2:5, 'zipcode', 'street_name'])
```

Row slicing with column selection

```
population_mean = gdf['population'].mean()
population_std  = gdf['population'].std()
```

Find the mean and standard deviation of a column

```
city_counts = gdf['city'].value_counts()
city_unique = gdf['city'].unique()
```

Count the number of occurrences per value, and the number of unique values

```
def double(income):
    return 2 * income

gdf['income'] = gdf['income'].applymap(double)
```

Transform column values with a custom function

```
gdf['income'] = gdf['income'].astype('float64')

print(gdf['income'].dtype)
```

Change the data type of a column

cuDF code examples

Query, sort, group, join, …

```
results = gdf.query('year_built < 1930')
```

Query a DataFrame with a boolean expression

```
results = gdf.nsmallest(3, columns=['income'])
```

Return ‘n’ smallest rows sorted by ‘columns’

```
results = gdf.sort_values(by='population')
```

Sort a column by its values

```
gdf[‘c_codes'] = gdf.city.cat.codes
cats = gdf.codes.unique()

enc = cudf.get_dummies(gdf, column='c_codes',
                       cats=cats, prefix=‘c_')
```

One-hot encoding

```
# Difference to Pandas:
#   - aggr. column names are prefixed with
#     the aggr. function name.
grouped = dgf.groupby(['city'])
                  .agg({'zipcode': 'count'})
```

Group by column with aggregate function

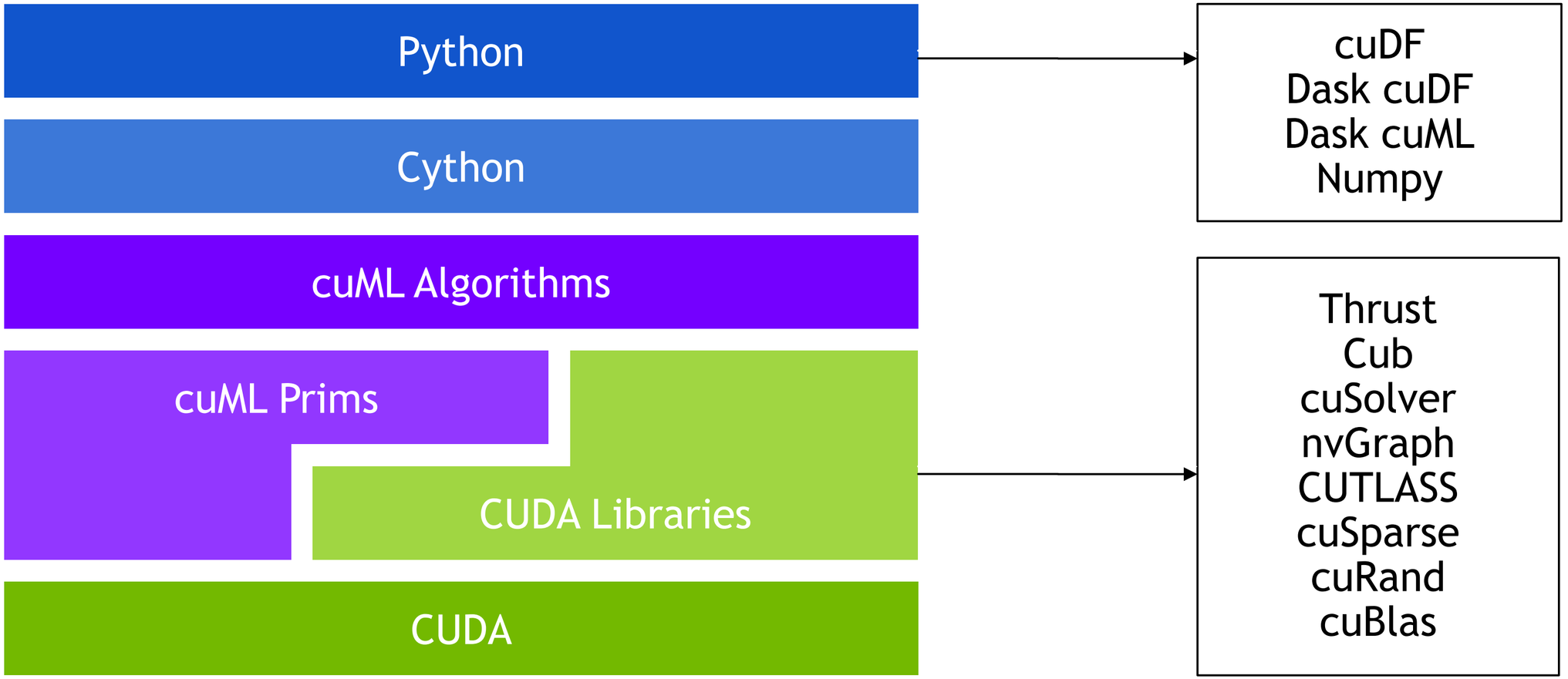
```
# join() uses the index.
left_j = left.set_index('count_zipcode')
right_j = right.set_index('zipcode')

# Different join styles are supported.
joined = left_j.join(right_j, how='right')

# Merge DataFrames with inner join.
merged = left.merge(right, on=['zipcode'])
```

Join and merge DataFrames

cuML

RAPIDS

ML Technology Stack



RAPIDS

cuML 21.06

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

CPU vs GPU

PRINCIPAL COMPONENT ANALYSIS (PCA)

Training results

CPU: 57.1 seconds
GPU: 4.28 seconds

System: AWS p3.8xlarge
CPU: Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, 32 vCPU cores, 244 GB RAM
GPU: Tesla V100 SXM2 16GB

Specific: Import CPU algorithm

```
from sklearn.decomposition import PCA
```

Common: Data loading and algo params

```
# Timer, load_data ...
from helper import *

# Data loading
nrows = 2**22
ncols = 400

X = load_data(nrows, ncols)

# Algorithm parameters
n_components = 8
svd_solver = 'full'
```

Common: Model training

```
pca = PCA(n_components, svd_solver)
_ = pca.fit_transform(X)
```

Specific: Import GPU algorithm

```
from cuml import PCA
```

Common: Data loading and algo params

```
# Timer, load_data ...
from helper import *

# Data loading
nrows = 2**22
ncols = 400

X = load_data(nrows, ncols)

# Algorithm parameters
n_components = 8
svd_solver = 'full'
```

Specific: DataFrame from Pandas to GPU

```
import cudf
X = cudf.DataFrame.from_pandas(X)
```

Common: Model training

```
pca = PCA(n_components, svd_solver)
_ = pca.fit_transform(X)
```

CPU vs GPU

PRINCIPAL COMPONENT ANALYSIS (PCA)

Training results

CPU: 57.1 seconds
GPU: 4.28 seconds

System: AWS p3.8xlarge
CPU: Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, 32 vCPU cores, 244 GB RAM
GPU: Tesla V100 SXM2 16GB

Specific: Import CPU algorithm

```
from sklearn.decomposition import PCA
```

Common: Data loading and algo params

```
# Timer, load_data ...
from helper import *

# Data loading
nrows = 2**22
ncols = 400

X = load_data(nrows, ncols)

# Algorithm parameters
n_components = 8
svd_solver = 'full'
```

Common: Model training

```
pca = PCA(n_components, svd_solver)
_ = pca.fit_transform(X)
```

Minimal changes, huge performance boost!

Specific: Import GPU algorithm

```
from cuml import PCA
```

Common: Data loading and algo params

```
# Timer, load_data ...
from helper import *

# Data loading
nrows = 2**22
ncols = 400

X = load_data(nrows, ncols)

# Algorithm parameters
n_components = 8
svd_solver = 'full'
```

Specific: DataFrame from Pandas to GPU

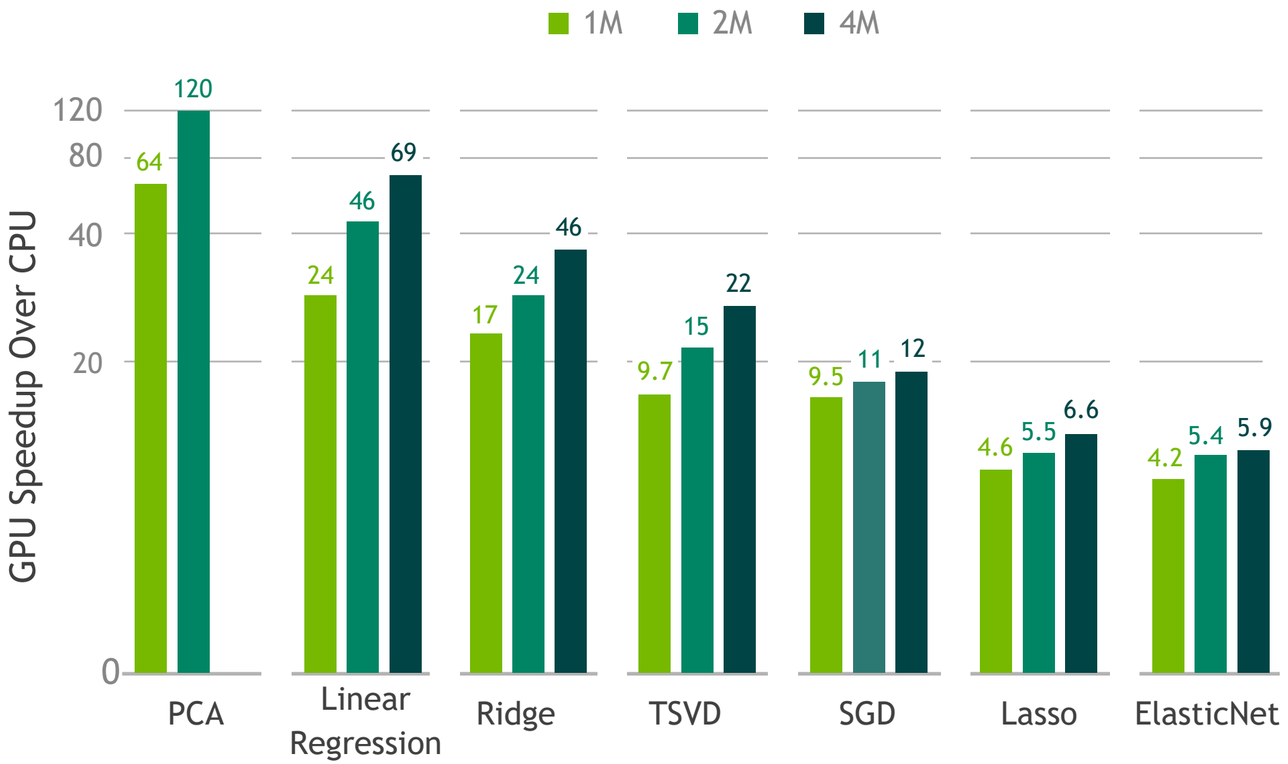
```
import cudf
X = cudf.DataFrame.from_pandas(X)
```

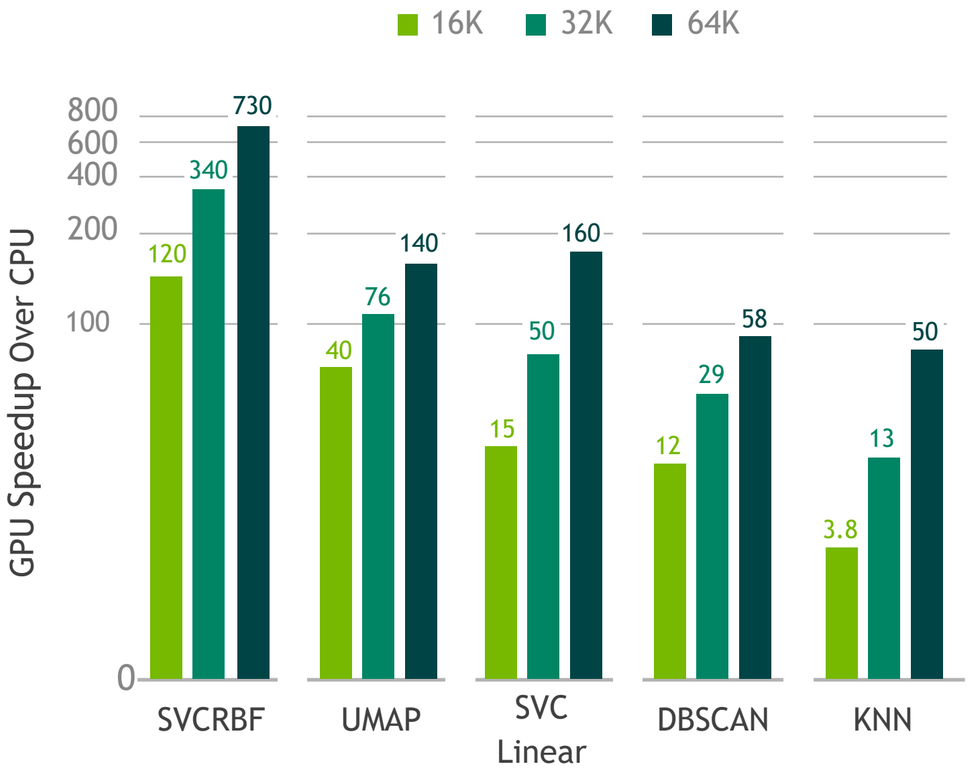
Common: Model training

```
pca = PCA(n_components, svd_solver)
_ = pca.fit_transform(X)
```

Benchmarks

cuML vs Scikit-learn





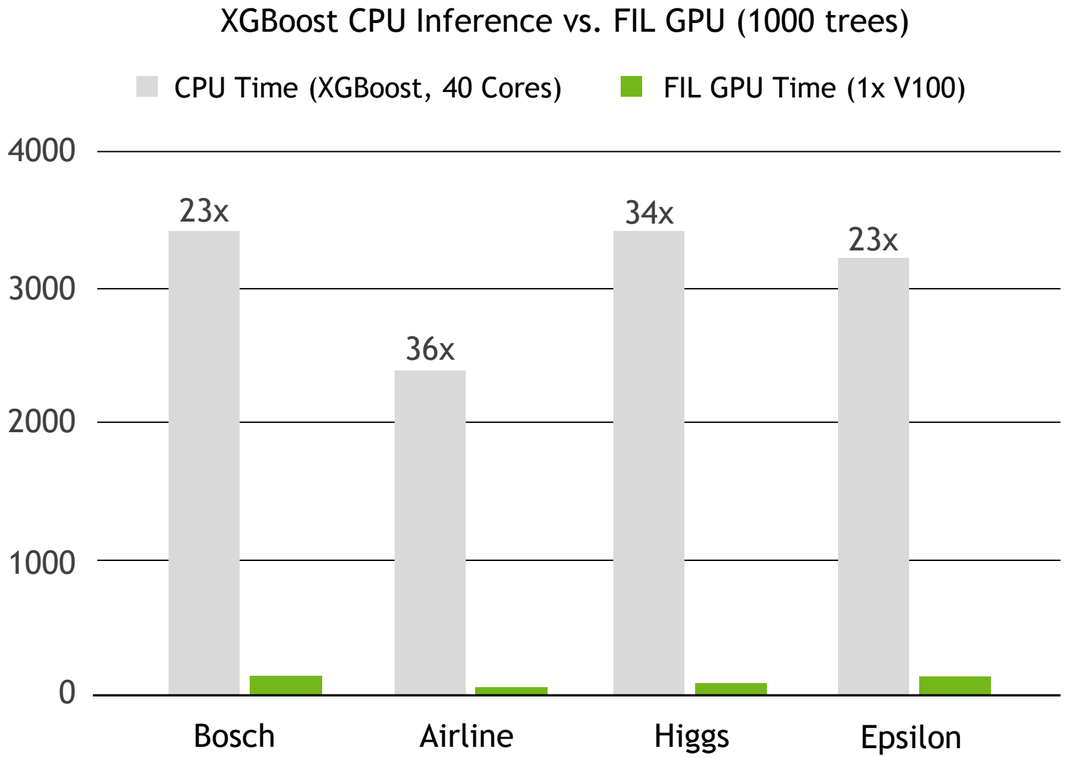
1xV100 vs 2x20 Core CPU

Forest Inference Library (FIL)

Taking Models From Training to Production

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

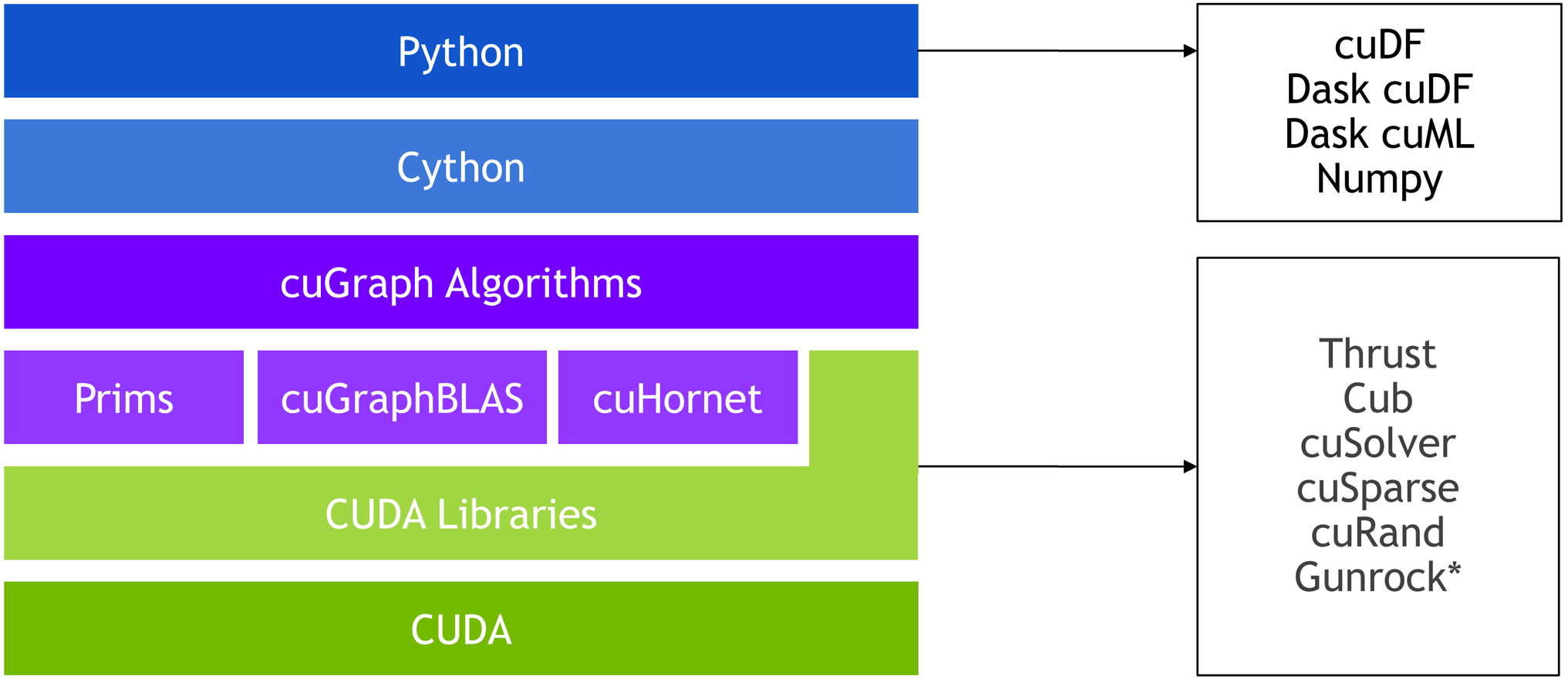
- Works with existing saved models:
  (XGBoost, LightGBM, cuML RF, scikit-learn RF)
- Lightweight Python API.
- Single V100 GPU can infer up to 34x faster than XGBoost dual-CPU node.
- Over 100 million forest inferences.



cuGraph

RAPIDS

Graph Technology Stack



RAPIDS

cuGraph 21.06

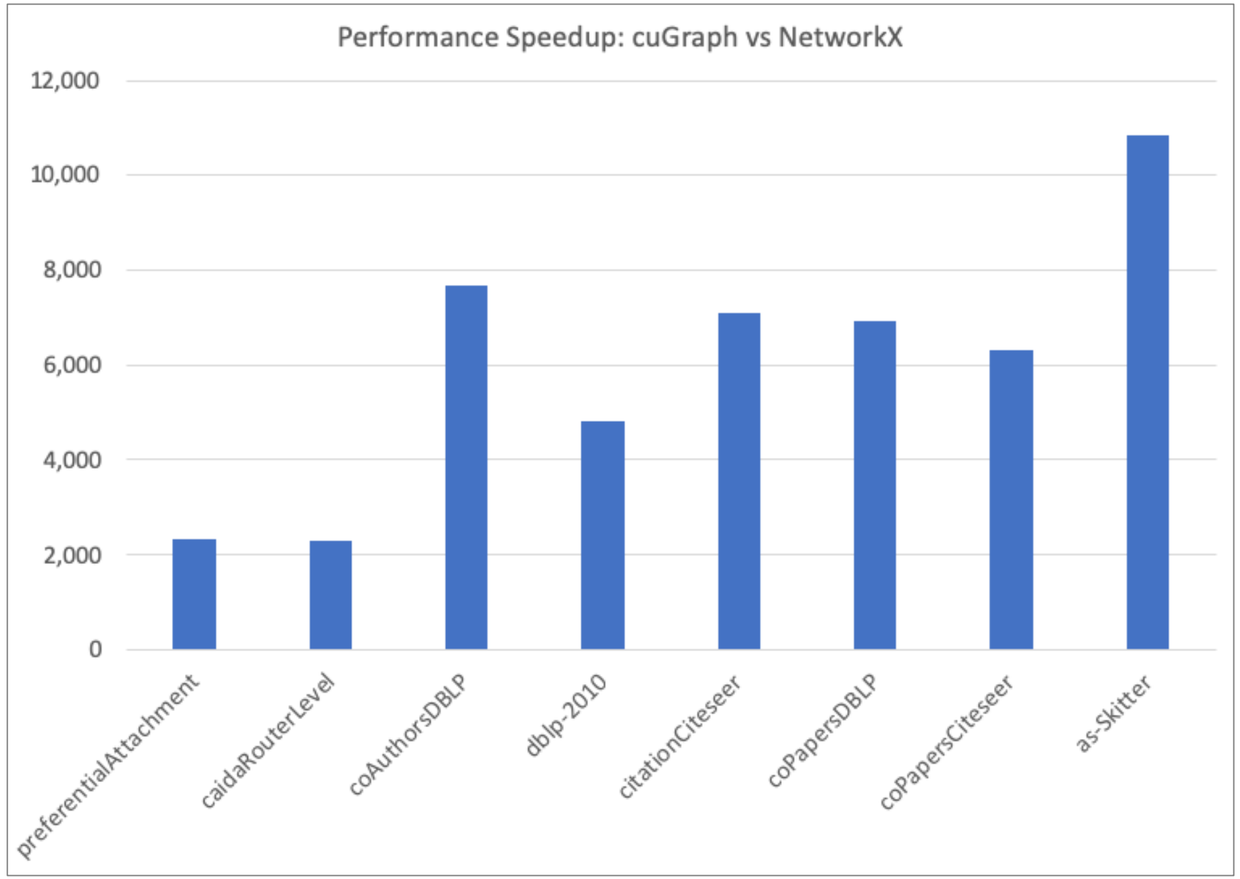
cuGraph	Single-GPU	Multi-Node Multi-GPU
PageRank	✓	✓
Personal Page Rank	✓	✓
Katz	✓	✓
Betweenness Centrality	✓	
Spectral Clustering	✓	
Louvain	✓	✓
Ensemble Clustering for Graphs	✓	
K-Truss & K-Core	✓	
Connected Components (Weak and Strong)	✓	
Triangle Counting	✓	
Single Source Shortest Path (SSSP)	✓	✓
Breadth First Search (BFS)	✓	✓
Jaccard & Overlap Coefficient	✓	
Force Atlas 2	✓	
Hungarian Algorithm & Leiden	✓	

Louvain Single Run

Performance Speedup: cuGraph vs NetworkX

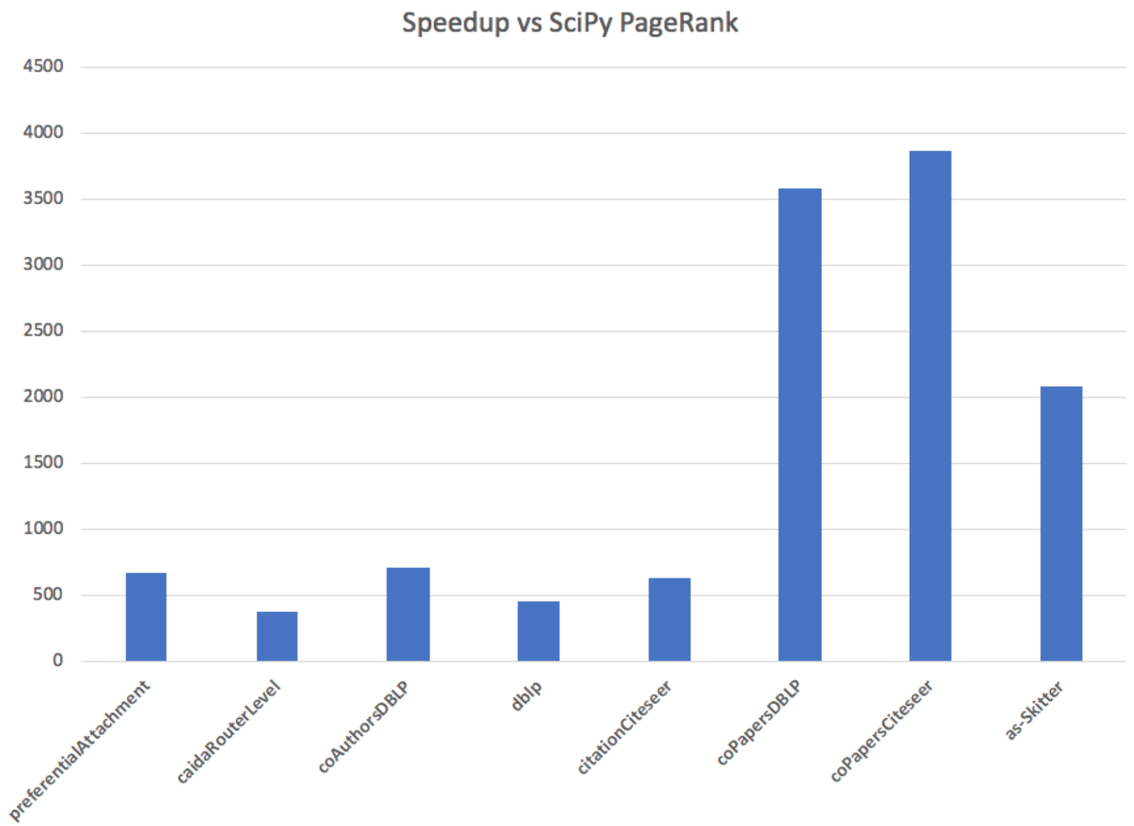
```
G = cugraph.Graph()
G.add_edge_list(gdf["src_0"], gdf["dst_0"], gdf["data"])
df, mod = cugraph.nvLouvain(G)
```

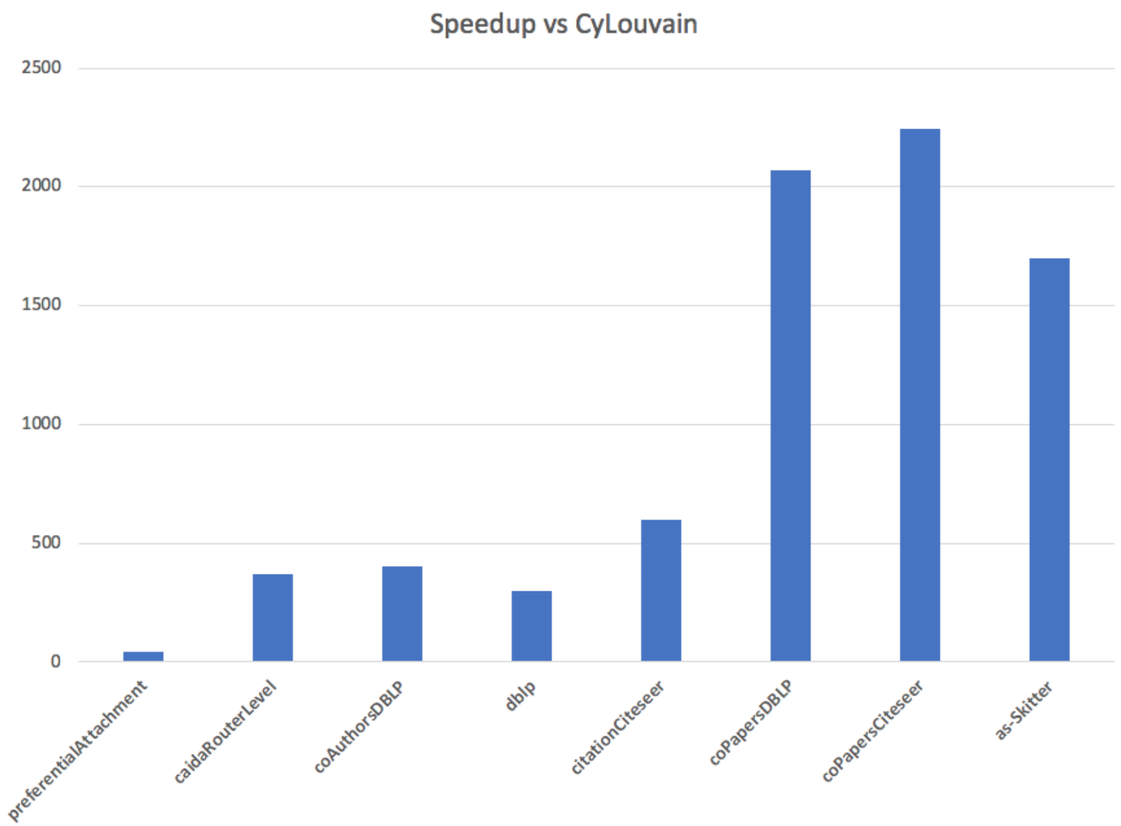
Dataset	Nodes	Edges
preferentialAttachment	100,000	999,970
caidaRouterLevel	192,244	1,218,132
coAuthorsDBLP	299,067	299,067
dblp-2010	326,186	1,615,400
citationCiteseer	268,495	2,313,294
coPapersDBLP	540,486	30,491,458
coPapersCiteseer	434,102	32,073,440
as-Skitter	1,696,415	22,190,596



Benchmarks

Performance Speedup vs Scipy PageRank and cyLouvain





GPU-Accelerated Visualization

RAPIDS cuxfilter

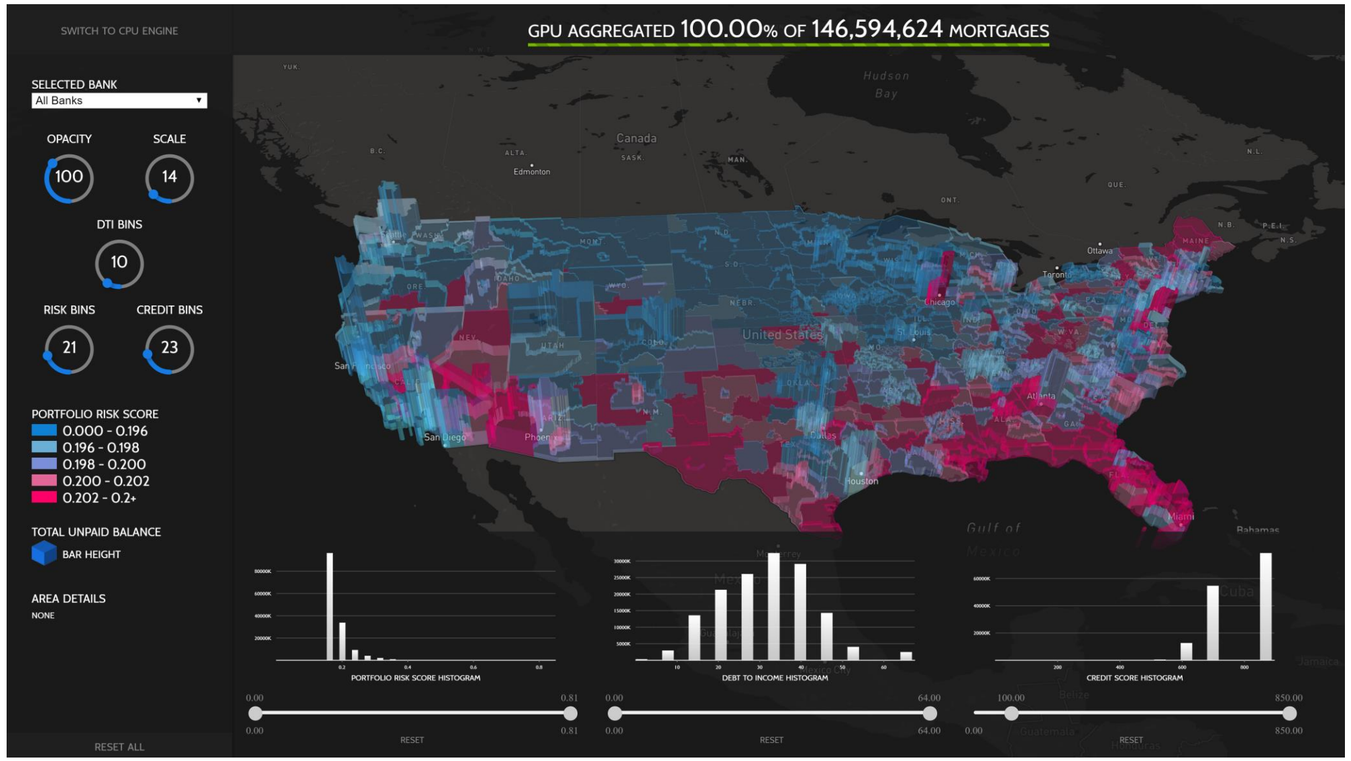
GPU-Accelerated cross filtering

STREAMLINED FOR NOTEBOOKS

Cuxfilter allows you to visually explore your cuDF dataframes through fully cross-filtered dashboards in less than 10 lines of notebook code.

MINIMAL COMPLEXITY & FAST RESULTS

Select from vetted chart libraries, pre-designed layouts, and multiple themes to find the best fit for a use case, at speeds typically 20x faster per chart than Pandas.



https://github.com/rapidsai/cuxfilter

Plotly Dash

GPU-Accelerated Visualization Apps with Python

plotly | Dash

Plotly Dash is RAPIDS accelerated, combining the ease-of-use development and deployment of Dash apps with the compute speed of RAPIDS. Work continues to optimize Plotly's API for even deeper RAPIDS integration.

300 MILLION DATAPOINT CENSUS EXAMPLE

Interact with data points of every individual in the United States, in real time, with the 2010 Census visualization. Get the code on GitHub and read about details here.

https://github.com/rapidsai/plotly-dash-rapids-census-demo

NVIDIA

pyViz Community

GPU-Accelerated Python Visualization

HoloViews

Uses cuDF to easily annotate and interactively explore data with minimal syntax. Work continues to optimize its API for deeper RAPIDS integration. holoviews.org

hvPlot

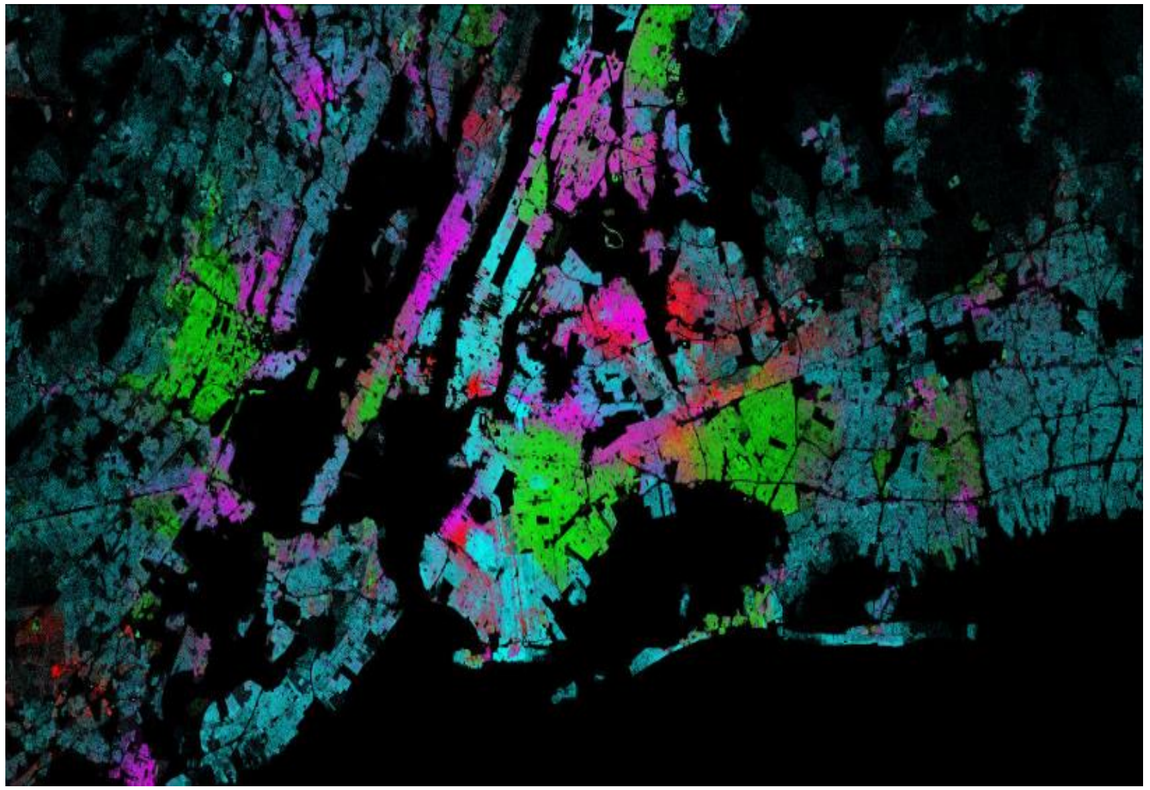
A higher-level plotting API built on HoloViews, work is ongoing to ensure its features can utilize the RAPIDS integration with HoloViews. hvplot.holoviz.org

bokeh

A backbone chart library used by the pyViz community. RAPIDS is actively supporting integration development to further its use in GPU-accelerated libraries and enhance rendering performance. bokeh.org

Datashader

Uses cuDF / dask cuDF for accelerated server-side rendering of extremely high density visualizations. Work continues to optimize more of its features for GPUs. datashader.org



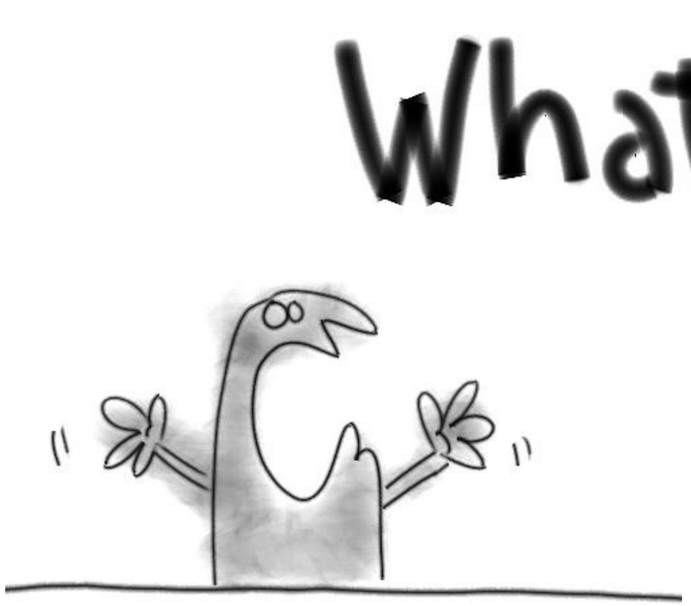
https://datashader.org/

What is XGBoost

XGBoost

DEFINITION

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

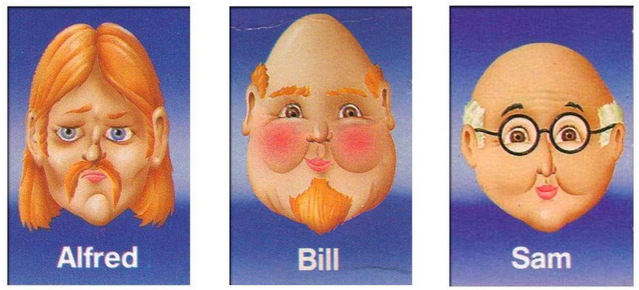


It is a powerful tool for solving classification and regression problems in a supervised learning setting.

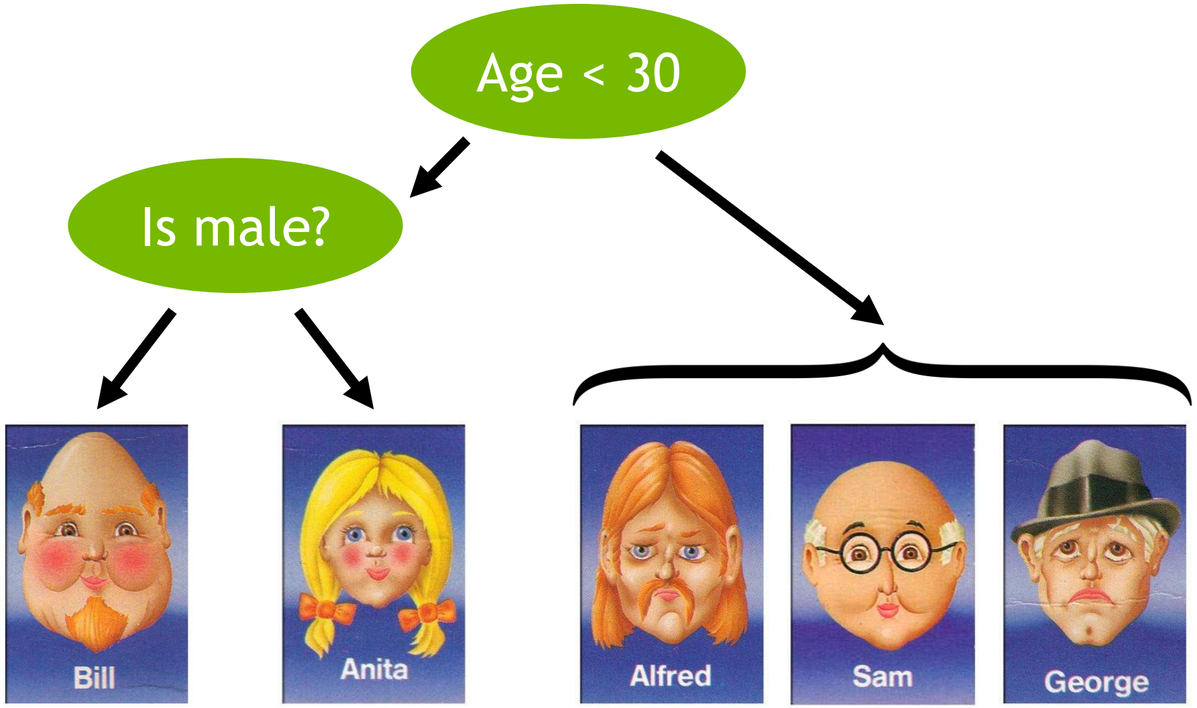
Who Enjoys Computer Games

Single Decision Tree

Input: age, gender, hair colour, …



Does the person like computer games?

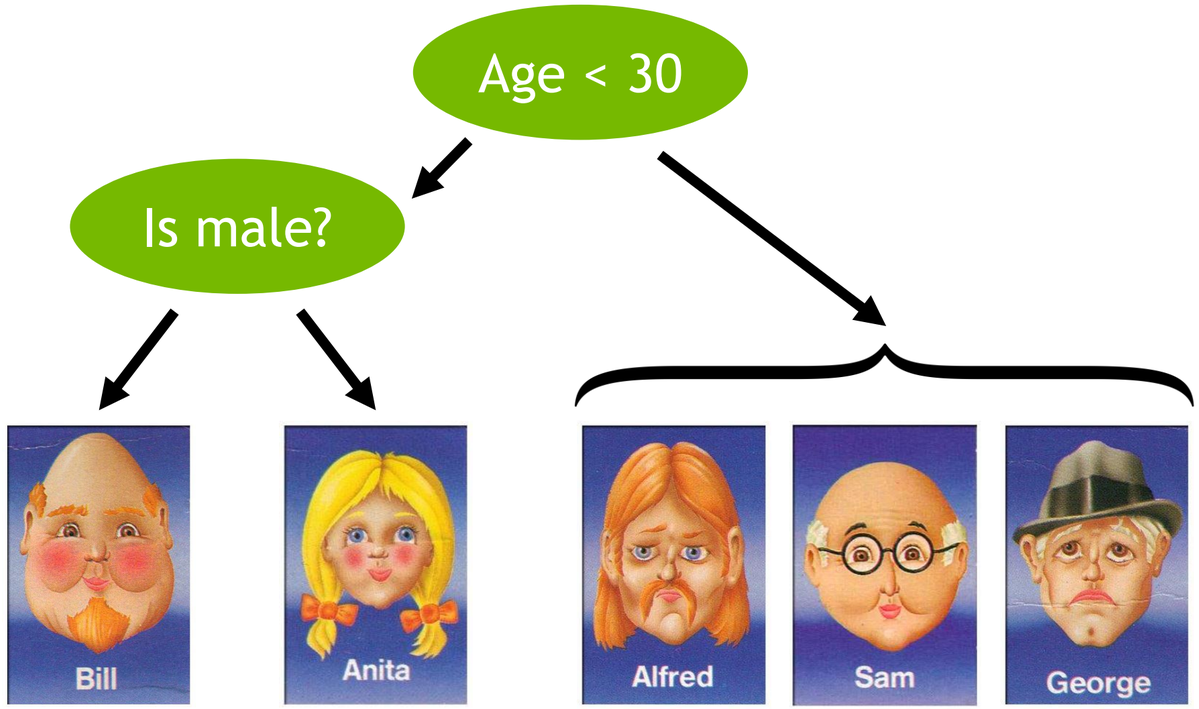


Prediction score in each leaf → +2 +1 -1 -1 -1

Combine Trees For Better Predictions

Ensembled Decision Trees

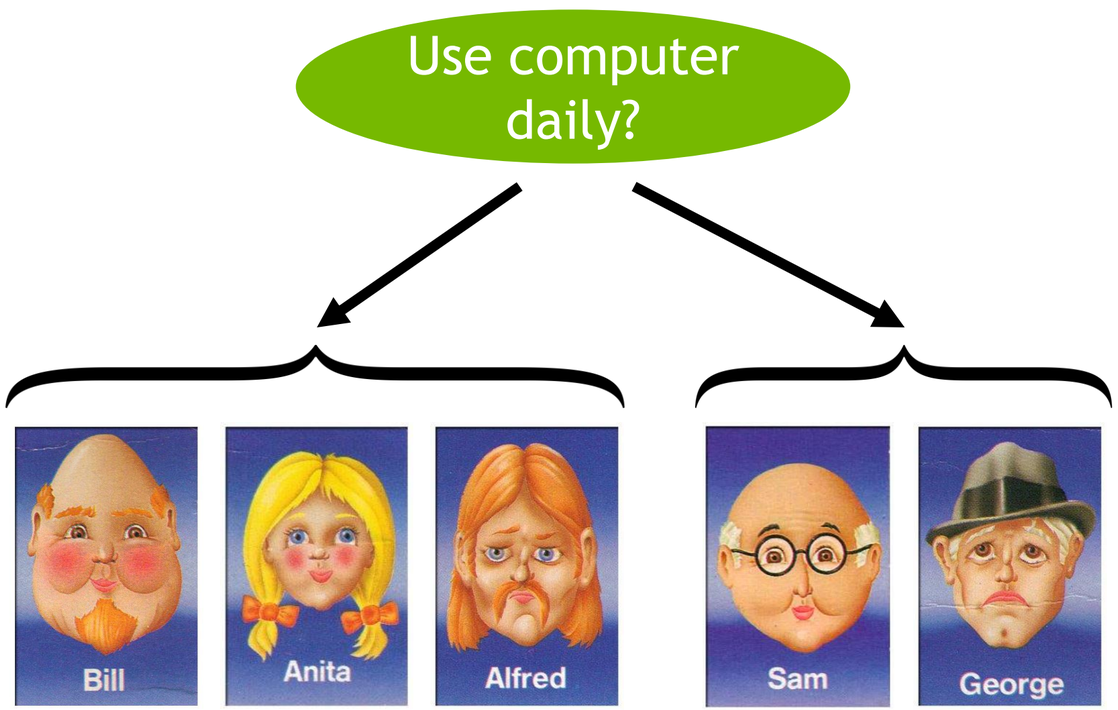
Tree 1



+2 +1 -1 -1 -1

f('Bill') = 2 + 0.9 = 2.9

Tree 2

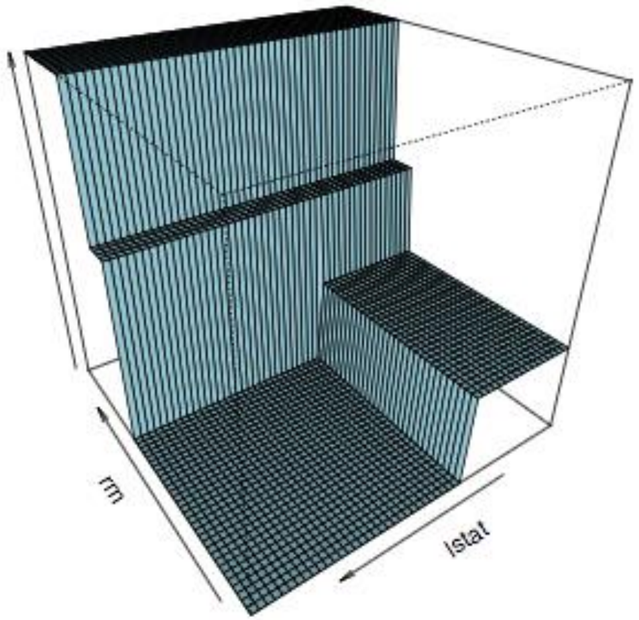


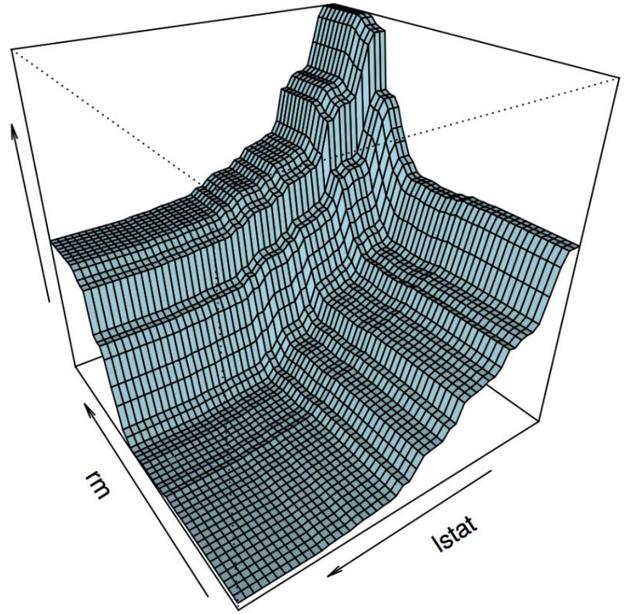
+0.9 +0.9 +0.9 -0.9 -0.9

f('Sam') = -1 - 0.9 = -1.9

Trained Models Visualization

Single Decision Tree vs Ensembled Decision Trees





Source: https://goo.gl/GWNdEm

Why XGBoost

A Strong History of Success

On a Wide Range of Problems

Winner of ACM RecSys Challenge 2017

- Job posting recommendation

Winner of KDD Cup 2016

- Research institutions' impact on the acceptance of submitted academic papers

Winner of Caterpillar Kaggle Contest 2015

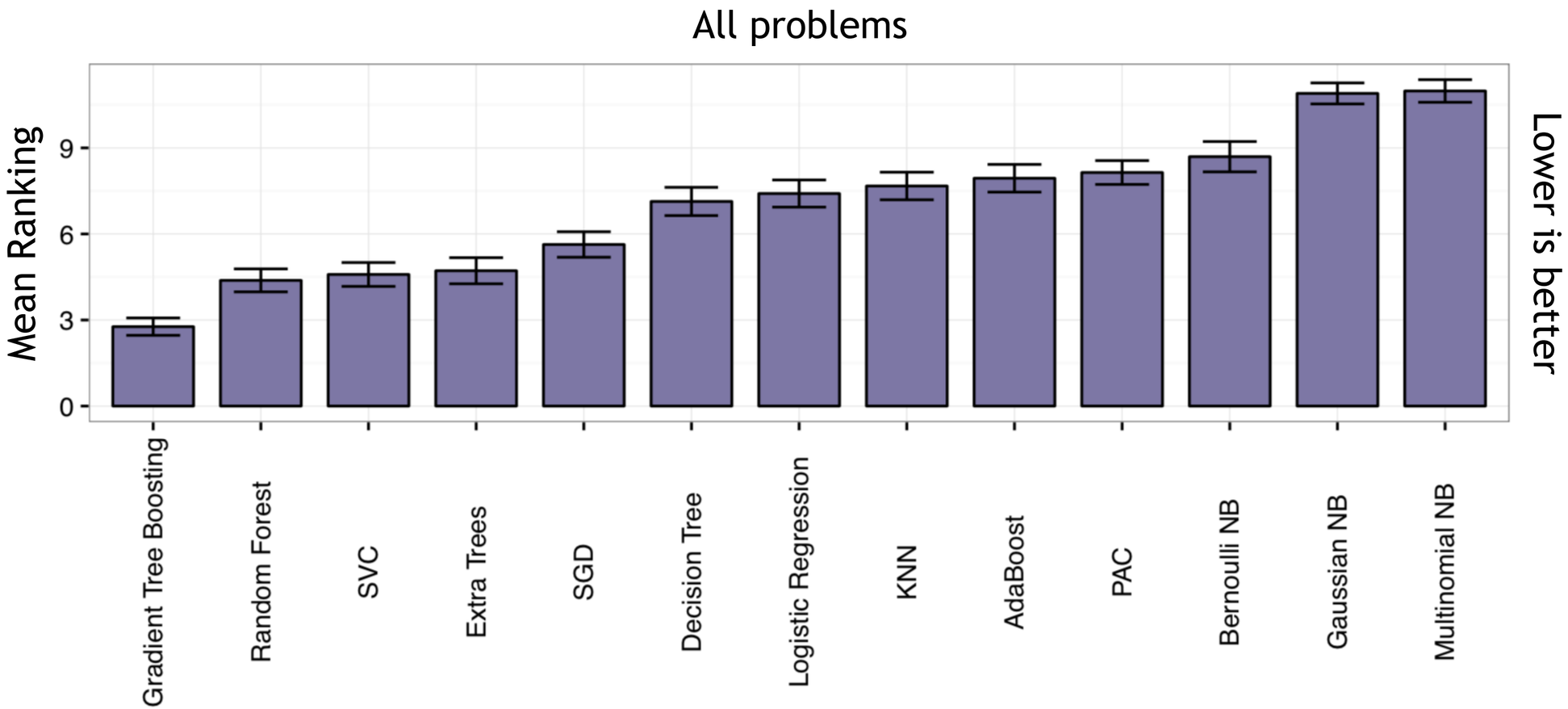
- Machinery component pricing

Winner of CERN Large Hadron Collider Kaggle Contest 2015

- Classification of rare particle decay phenomena

Which ML Algorithm Performs Better

Average Rank Across 165 Datasets



Source: https://goo.gl/R8Y8Pp

XGBoost + RAPIDS

XGBoost + RAPIDS

XGBoost

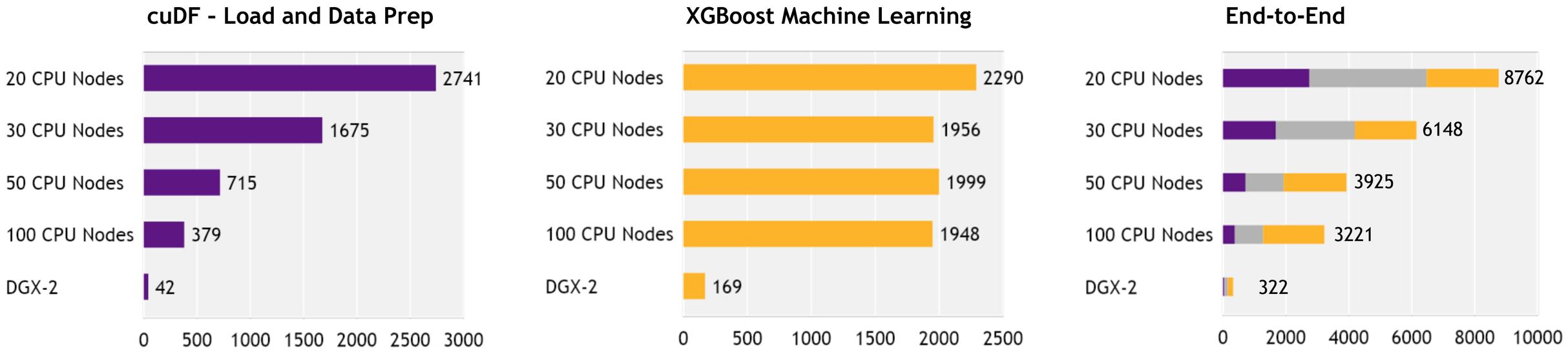
- Tuned for extreme performance and high efficiency
- Multi-GPU and Multi-Node Support

RAPIDS

- E2E data science & analytics pipeline entirely on GPU
- User-friendly Python interfaces
- Relies on CUDA primitives, exposes parallelism and high-memory bandwidth
- Dask integration for managing workers and data in distributed environments

Benchmarks

Time in seconds – Shorter is better



Benchmark

200GB CSV dataset; Data preparation includes joins, variable transformations.

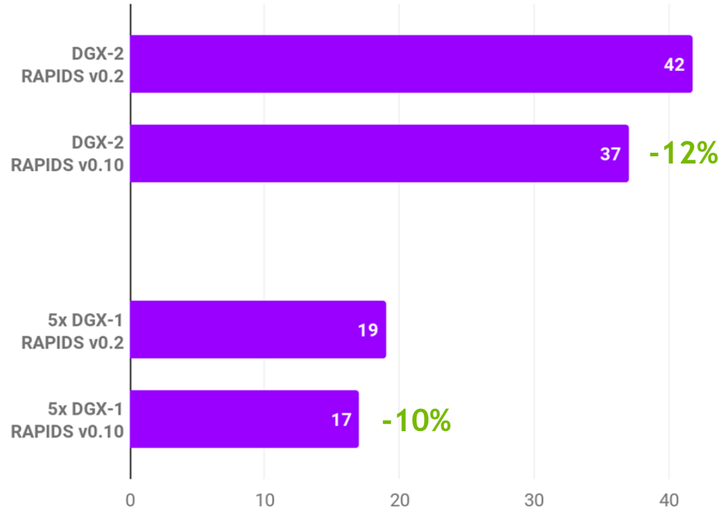
CPU Cluster Configuration

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

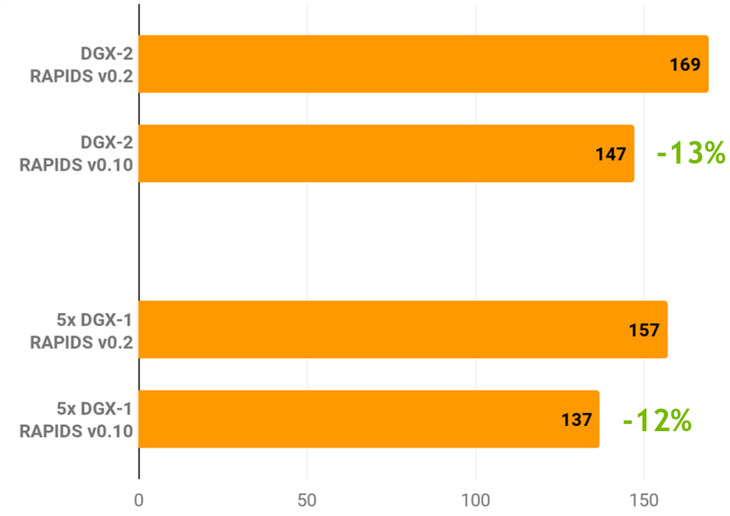
Improving Over time

RAPIDS 0.2 vs 0.10 speedup

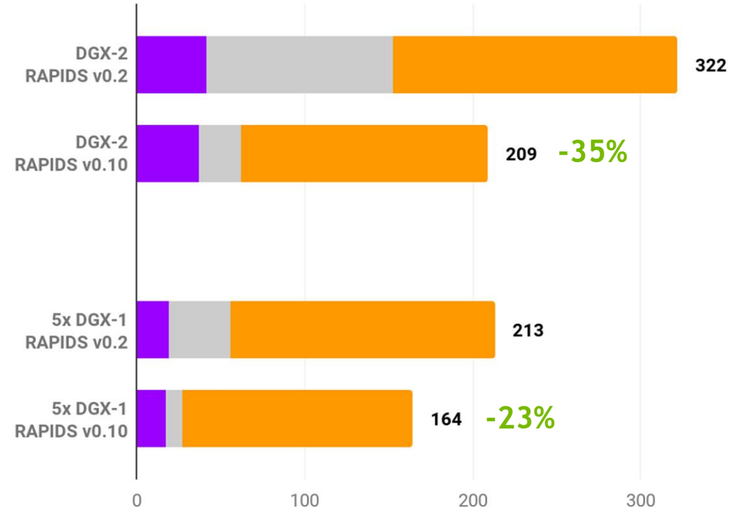
cuDF - Load and Data Prep



XGBoost Machine Learning



End-to-End



cuDF (Load and Data Preparation) Data Conversion XGBoost

Benchmark

200GB CSV dataset; Data preparation includes joins, variable transformations.

CPU Cluster Configuration

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

DGX Cluster Configuration

5x DGX-1 on InfiniBand network

How to start

RAPIDS

Source code on GitHub

https://github.com/rapidsai

GitHub

Containers on NGC & Docker Hub

https://ngc.nvidia.com

NVIDIA NGC

docker

Conda packages

https://anaconda.org/rapidsai

CONDA®

On-premise

In the cloud

Pascal architecture or better
CUDA 10.1.2, 10.2, 11
Ubuntu 16.04, 18.04, 20.04,
CentOS 7, 8 & RHEL 7, 8

RAPIDS RELEASE SELECTOR

RAPIDS is available as conda packages, docker images, and from source builds. Use the tool below to select your preferred method, packages, and environment to install RAPIDS. Certain combinations may not be possible and are dimmed automatically. Be sure you've met the required **prerequisites above** and see the **details below**.

	☑ Preferred		⚗ Advanced	
METHOD	**Conda**	Docker + Examples	Docker + Dev Env	Source
RELEASE	**Stable (0.17)**		Nightly (0.18a)	
TYPE	**RAPIDS and BlazingSQL**		RAPIDS Core (w/o BlazingSQL)	
PACKAGES	**All Packages**	cuDF	cuML	cuGraph	cuSignal	cuSpatial	cuxfilter
LINUX	**Ubuntu 16.04**	Ubuntu 18.04	CentOS 7	RHEL 7
PYTHON	**Python 3.7**		Python 3.8	
CUDA	**CUDA 10.1.2**	CUDA 10.2	CUDA 11.0	

ℹ **NOTE:** Ubuntu 16.04/18.04 & CentOS 7 use the same `conda create` commands.

COMMAND

```
conda create -n rapids-0.17 -c rapidsai -c nvidia -c conda-forge \
    -c defaults rapids-blazing=0.17 python=3.7 cudatoolkit=10.1
```

COPY COMMAND RESET SELECTOR

https://rapids.ai/start.html

RAPIDS RELEASE SELECTOR

RAPIDS is available as conda packages, docker images, and from source builds. Use the tool below to select your preferred method, packages, and environment to install RAPIDS. Certain combinations may not be possible and are dimmed automatically. Be sure you've met the required **prerequisites above** and see the **details below**.

	☑ Preferred		⚗ Advanced	
METHOD	Conda	**Docker + Examples**	Docker + Dev Env	Source
RELEASE	**Stable (0.17)**		Nightly (0.18a)	
TYPE	**RAPIDS and BlazingSQL**		RAPIDS Core (w/o BlazingSQL)	
PACKAGES	**All Packages**	cuDF	cuML	cuGraph	cuSignal	cuSpatial	cuxfilter
LINUX	**Ubuntu 16.04**	Ubuntu 18.04	CentOS 7	RHEL 7
PYTHON	**Python 3.7**		Python 3.8	
CUDA	**CUDA 10.1.2**	CUDA 10.2	CUDA 11.0	

COMMAND

```
docker pull rapidsai/rapidsai:cuda10.1-runtime-ubuntu16.04-py3.7
docker run --gpus all --rm -it -p 8888:8888 -p 8787:8787 -p 8786:8786 \
    rapidsai/rapidsai:cuda10.1-runtime-ubuntu16.04-py3.7
```

COPY COMMAND **RESET SELECTOR**

https://rapids.ai/start.html

NVIDIA GPU CLOUD

ACCELERATED SOFTWARE

ALL CONTENT TYPES | CONTAINERS | MODELS | MODEL SCRIPTS | HELM CHARTS

Search containers

Sort: Last Modified

TensorFlow

TensorFlow is an open-source software library for high-performance numerical computation. Its flexible architecture allows easy deployment of computation a...

20.02-tf1-py3
built by NVIDIA 02/27/20

TensorRT Inference Serv...

TensorRT Inference Server provides a data center inference solution optimized for NVIDIA GPUs. It maximizes inference utilization and performance on GPUs via ...

20.02-py3-clientsdk
built by NVIDIA 02/26/20

NVCaffe

NVIDIA Caffe, also known as NVCaffe, is an NVIDIA-maintained fork of Berkeley Vision and Learning Center (BVLC) Caffe tuned for NVIDIA GPUs, particularly in multi-GPU co...

20.02-py3
built by NVIDIA 02/26/20

TensorRT

NVIDIA TensorRT is a C++ library that facilitates high-performance inference on NVIDIA graphics processing units (GPUs). TensorRT takes a trained network, which ...

20.02-py2
built by NVIDIA 02/26/20

DIGITS

The NVIDIA Deep Learning GPU Training System (DIGITS) puts the power of deep learning into the hands of engineers and data scientists.

20.02-tensorflow-py3
built by NVIDIA 02/26/20

MXNet

MXNet is a deep learning framework that allows you to mix the flavors of symbolic programming and imperative programming to maximize efficiency and ...

20.02-py3
built by NVIDIA 02/26/20

Kaldi

Kaldi is an open-source software framework for speech processing.

20.02-py3
built by NVIDIA 02/26/20

PyTorch

PyTorch is a GPU accelerated tensor computational framework with a Python front end. Functionality can be easily extended with common Python libraries s...

20.02-py3
built by NVIDIA 02/25/20

ParaView IndeX

ParaView is one of the most popular visualization software for analyzing HPC datasets. NVIDIA IndeX delivers real-time, interactive visualization of large volumetr...

5.7.0-egl-pvw
built by NVIDIA 02/21/20

Isaac Sim

Isaac Sim Leonardo Preview is a cloud-based simulator supporting Leonardo, a robotic manipulation research application.

2020.1_preview
built by NIVIDA 02/21/20

OpenACC Training Mater...

These training materials have been developed as a collaboration between the University of Delaware and NVIDIA Corporation and are provided free of cha...

20.1.1
built by PGI Compilers ... 02/20/20

Clara-Train-SDK

NVIDIA Clara is a python based SDK. It includes the following components:Annotation Server for AI Assisted Annotation, Training framework ...

v2.0
built by NVIDIA 02/18/20

RAPIDS

The RAPIDS suite of software libraries gives you the freedom to execute end-to-end data science and analytics pipelines entirely on GPUs.

cuda10.1-runtime-ubuntu18.04
built by NVIDIA 02/06/20

RELION

RELION (for REgularized LIkelihood OptimizatioN) implements an empirical Bayesian approach for analysis of electron cryo-microscopy (Cryo-EM). Specifically it...

2.1.b1
built by Stockholm Univ... 01/27/20

vmd

VMD is designed for modeling, visualization, and analysis of biomolecular systems such as proteins, nucleic acids, lipid membranes, carbohydrate structure...

cuda9-ubuntu1604-egl-1.9.4a17
built by UIUC 01/08/20

ACCELERATED SOFTWARE

ALL CONTENT TYPES | CONTAINERS | MODELS | MODEL SCRIPTS | HELM CHARTS

Search models

Sort: Last Modified

Unsupervised Latent La...
Pytorch | FP32
1
built by unknown 02/21/20

Transformer-BIG-en-de...
PyTorch with NeMo | FP16
1
built by okuchaiev@nvid... 02/14/20

JasperNetDr 10x5 for N...
PyTorch with NeMo | FP16
1
built by jasoli@nvidia.... 02/14/20

BertLargeUncasedForN...
PyTorch with NeMo | FP32
4
built by Nvidia 02/11/20

BertBaseCasedForNeMo
PyTorch with NeMo | FP32
3
built by Nvidia 02/11/20

BertBaseUncasedForNe...
PyTorch with NeMo | FP32
2
built by Nvidia 02/11/20

Multidataset-QuartzNet...
PyTorch with NeMo | fp32
1
built by unknown 01/08/20

QuartzNet 15x5 LibriSpe...
PyTorch with NeMo | FP32
1
built by unknown 01/08/20

TensorRT 7.0 Resnet50 ...
TensorRT | FP32
1
built by NVIDIA 01/07/20

TensorRT 7.0 Resnet50 ...
TensorRT | FP16
1
built by NVIDIA 01/07/20

TensorRT 7.0 Resnet50 ...
TensorRT | INT8
1
built by NVIDIA 01/07/20

TensorRT 7.0 Mobilenet_...
TensorRT | FP32
1
built by NVIDIA 01/07/20

TensorRT 7.0 Mobilenet_...
TensorRT | FP16
1
built by NVIDIA 01/07/20

TensorRT 7.0 Mobilenet_...
TensorRT | INT8
1
built by NVIDIA 01/07/20

TensorRT 7.0 Inception_...
TensorRT | FP32
1
built by NVIDIA 01/07/20

ACCELERATED SOFTWARE

ALL CONTENT TYPES | CONTAINERS | MODELS | MODEL SCRIPTS | HELM CHARTS

Search model scripts

Sort: Last Modified

V-Net Medical for Tenso...
TensorFlow | FP16, FP32
built by unknown 02/21/20

BERT for PyTorch
PyTorch | FP16, FP32
built by unknown 02/21/20

BERT for TensorFlow
TensorFlow | FP16, FP32
built by NVIDIA 02/21/20

Transformer-XL for PyTo...
PyTorch | FP16, FP32
built by unknown 02/21/20

NCF for PyTorch
PyTorch | FP16, FP32
built by NVIDIA 02/21/20

Jasper for PyTorch
PyTorch | FP16, FP32
built by unknown 01/27/20

Clara Deploy SDK
TensorFlow | FP16
built by NVIDIA 01/17/20

SE-ResNeXt101-32x4d f...
PyTorch | FP16, FP32
built by unknown 01/07/20

Tacotron2 and Waveglow...
PyTorch | FP16, FP32
built by NVIDIA 01/07/20

ResNeXt101-32x4d for P...
PyTorch | FP16, FP32
built by unknown 01/07/20

ResNet v1.5 for PyTorch
PyTorch | FP16, FP32
built by NVIDIA 01/07/20

GNMT v2 for TensorFlow
TensorFlow | FP16, FP32
built by NVIDIA 01/07/20

SSD v1.1 for PyTorch
PyTorch | FP16, FP32
built by NVIDIA 01/07/20

Transformer for PyTorch
PyTorch | FP16, FP32
built by NVIDIA 12/13/19

BioBERT for TensorFlow
TensorFlow | FP16, FP32
built by unknown 12/13/19

ACCELERATED SOFTWARE

ALL CONTENT TYPES | CONTAINERS | MODELS | MODEL SCRIPTS | HELM CHARTS

Search helm charts

Sort: Last Modified

Conundrum Aircraft-En...

Helm chart for Conundrum Deep Learning based Predictive Maintenance Demo using NASA-Turbofan Engine Degradation Simulation dataset

latest
built by unknown 02/21/20

Ironyun - ainvr-app

IronYun is the leading AI deep learning video search and real-time video analytics solutions company. We provide enterprise customers with hyper-converged, all-in-o...

1.0.1
built by NVIDIA 01/07/20

DeepVison - Vehicle Insp...

Advanced computer vision technology to understand images and video automatically, turning visual content into real-time analytics and valuable insights

1.0.1
built by NVIDIA 01/07/20

TensorRT Inference Serv...

TensorRT Inference Server Helm Chart

1.0.0
built by NVIDIA 12/09/19

Kinetica

Kinetica is a very fast, distributed, GPU-accelerated database with advanced analytics, visualization, geospatial, and machine learning functionality.

0.1.0
built by NVIDIA 10/28/19

GPU Operator

Deploy and Manage NVIDIA GPU resources in Kubernetes.

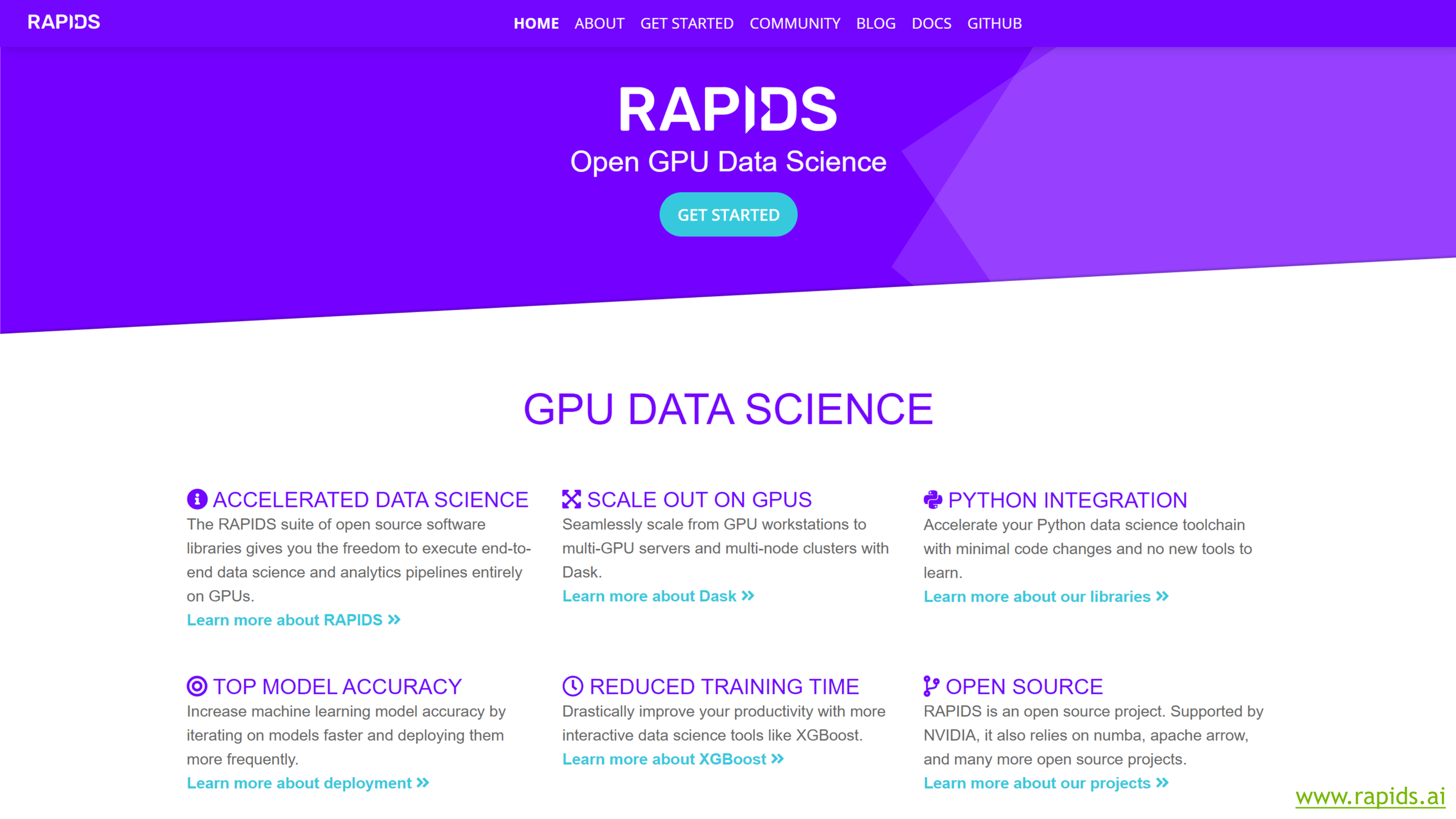
1.0.0-techpreview.1
built by NVIDIA 10/21/19

DeepStream - Intelligent...

This is an easy to deploy a video analytics demo that allows you to demo GPU accelerated video analytics. The container is based on the NVIDIA DeepStream cont...

0.1.2
built by NVIDIA 10/21/19

Learn more


RAPIDS
HOME ABOUT GET STARTED COMMUNITY BLOG DOCS GITHUB
RAPIDS
Open GPU Data Science
GET STARTED
GPU DATA SCIENCE
ACCELERATED DATA SCIENCE
The RAPIDS suite of open source software libraries gives you the freedom to execute end-to-end data science and analytics pipelines entirely on GPUs.
Learn more about RAPIDS »
SCALE OUT ON GPUS
Seamlessly scale from GPU workstations to multi-GPU servers and multi-node clusters with Dask.
Learn more about Dask »
PYTHON INTEGRATION
Accelerate your Python data science toolchain with minimal code changes and no new tools to learn.
Learn more about our libraries »
TOP MODEL ACCURACY
Increase machine learning model accuracy by iterating on models faster and deploying them more frequently.
Learn more about deployment »
REDUCED TRAINING TIME
Drastically improve your productivity with more interactive data science tools like XGBoost.
Learn more about XGBoost »
OPEN SOURCE
RAPIDS is an open source project. Supported by NVIDIA, it also relies on numba, apache arrow, and many more open source projects.
Learn more about our projects »
www.rapids.ai

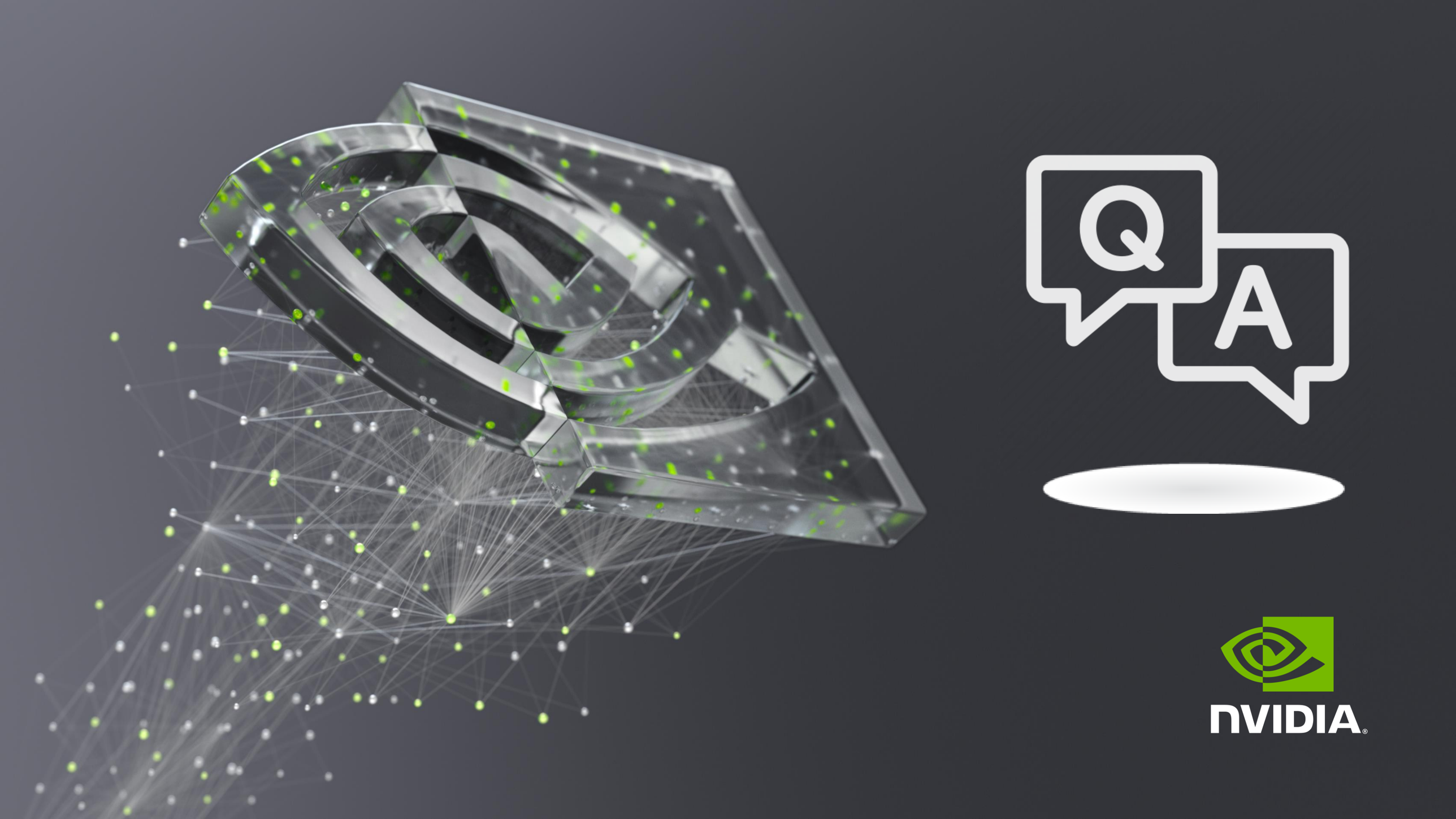
Summary

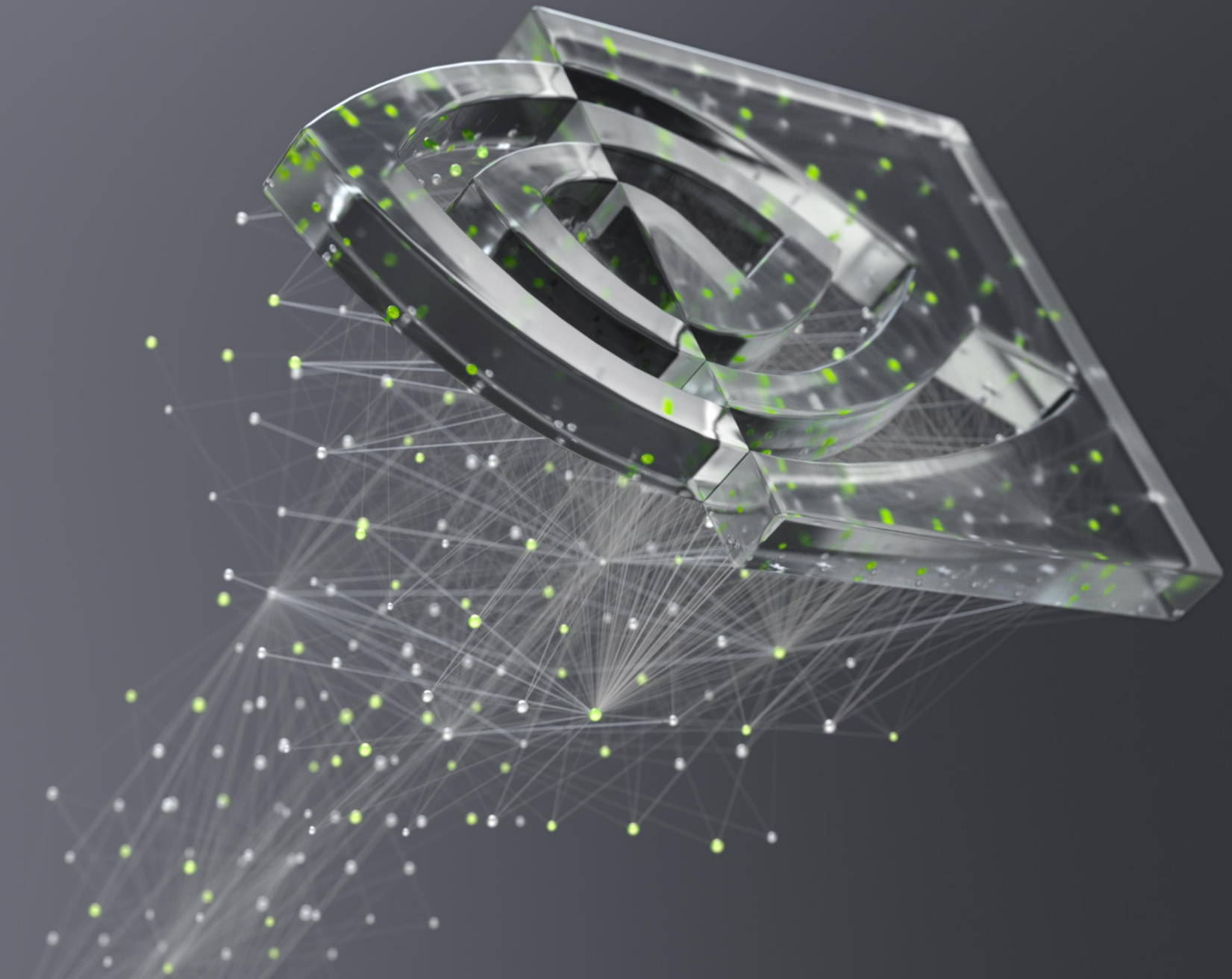
RAPIDS

GPU Accelerated Data Science

RAPIDS is a set of open-source software libraries which gives you the freedom to execute end-to-end data science and analytics pipelines entirely on GPUs.

www.rapids.ai


Q
A
NVIDIA



Thank You!

NVIDIA