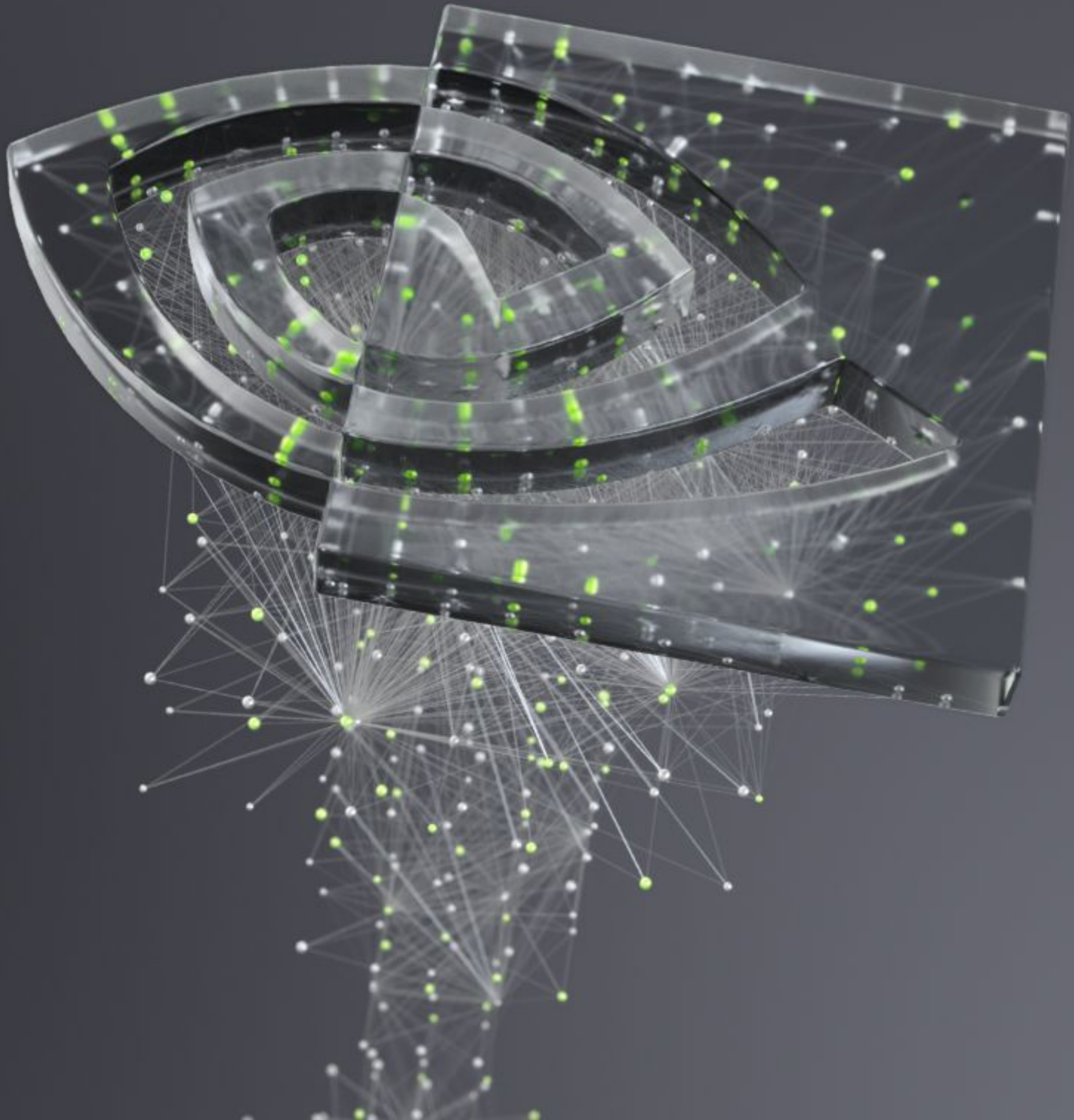


Intro to RAPIDS:

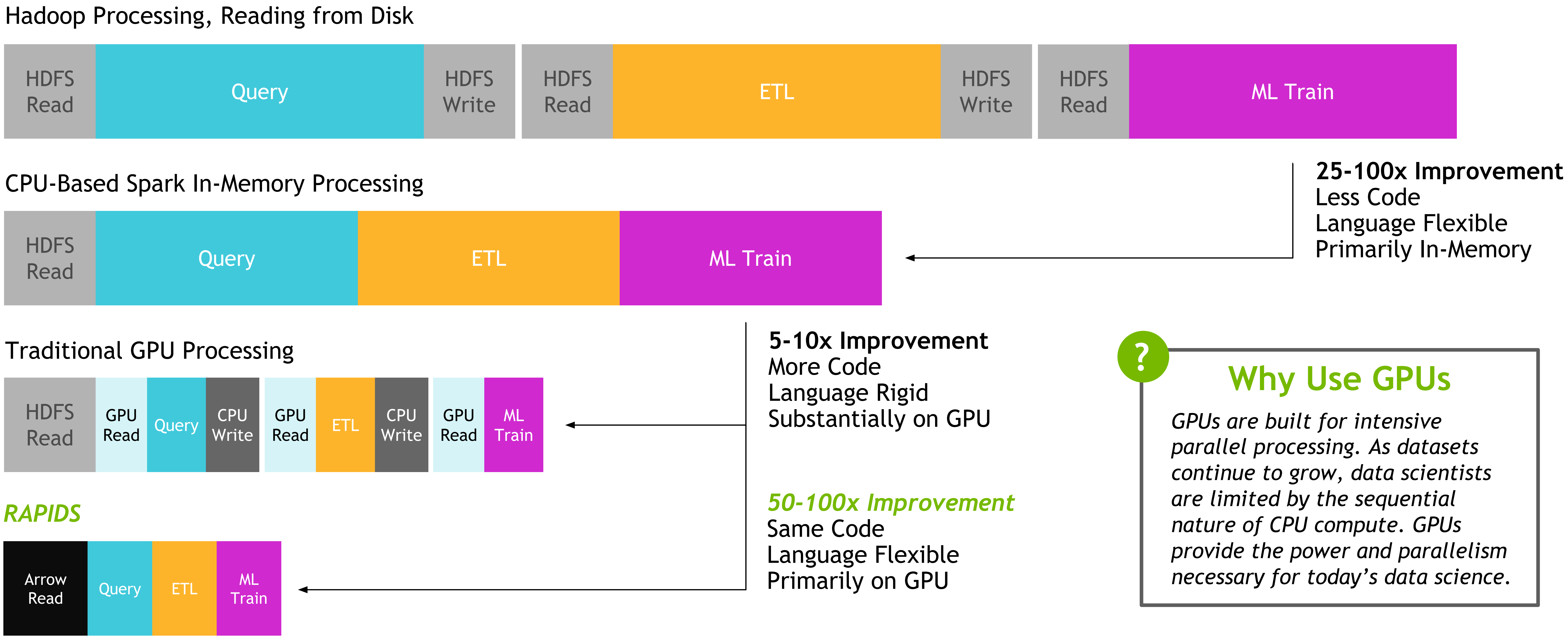
PyHEP 2021

Benjamin Zaitlen

RAPIDS Foundations

The Evolution of Data Processing

Faster Data Access, Less Data Movement



Hadoop Processing, Reading from Disk

CPU-Based 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

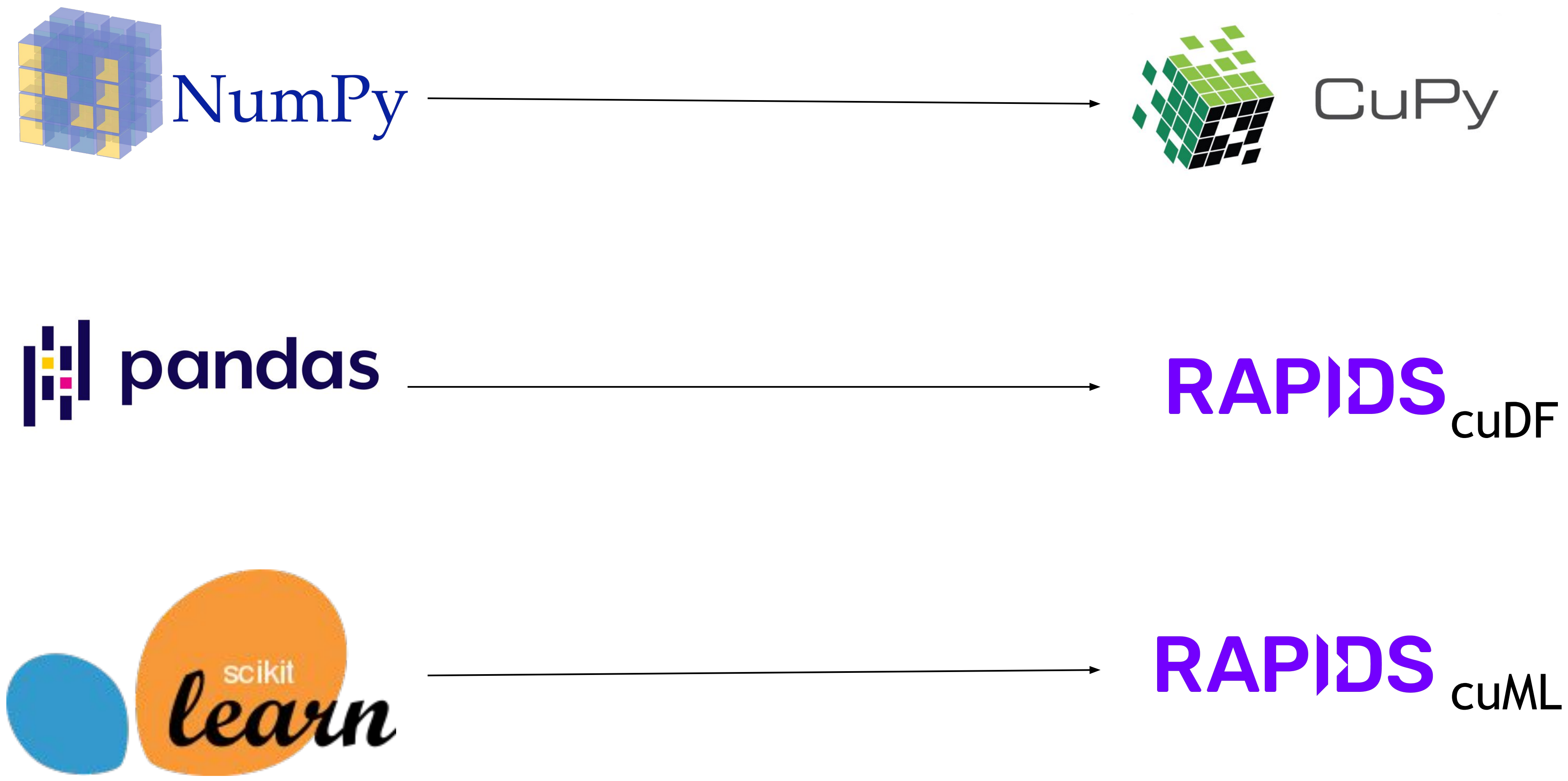
50-100x Improvement
Same Code
Language Flexible
Primarily on GPU

? Why Use GPUs

GPUs are built for intensive parallel processing. As datasets continue to grow, data scientists are limited by the sequential nature of CPU compute. GPUs provide the power and parallelism necessary for today's data science.

RAPIDS

GPU Accelerated Data Science



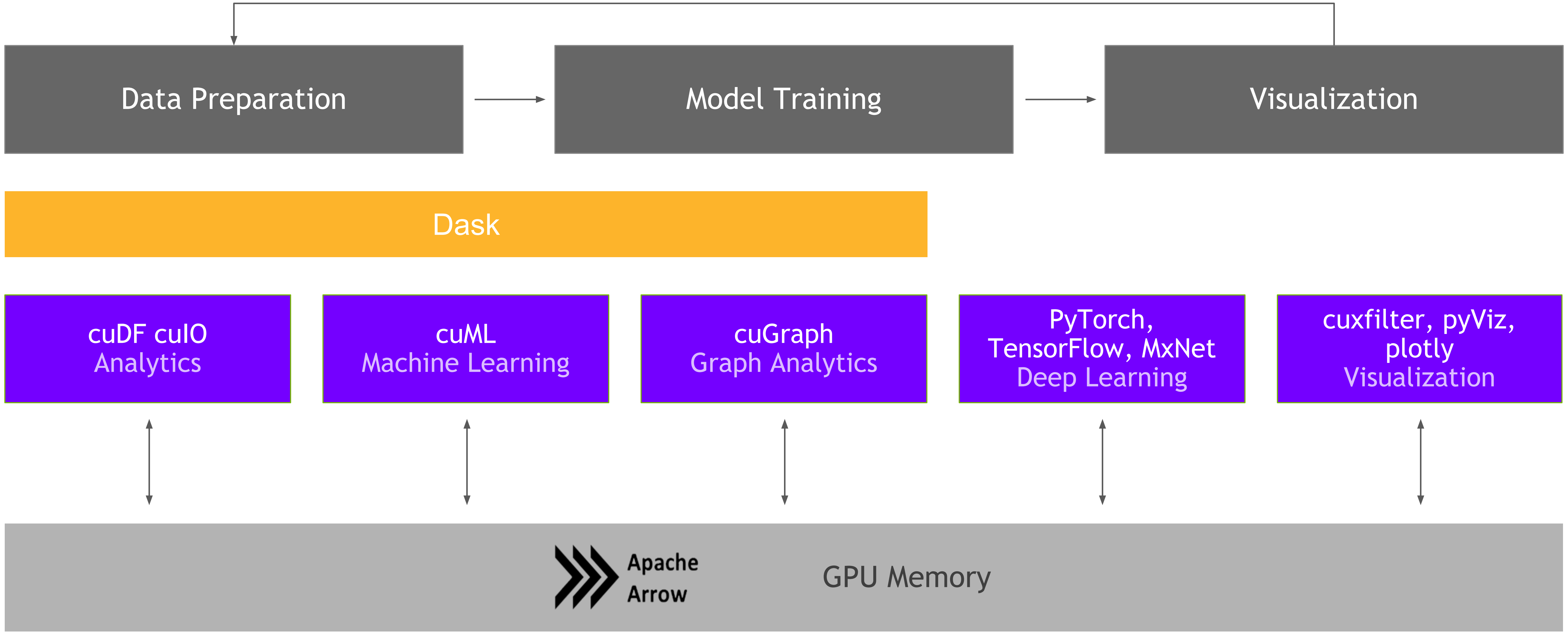
NumPy → CuPy

pandas → RAPIDS cuDF

scikit learn → RAPIDS cuML

RAPIDS

End-to-End GPU Accelerated Data Science



Data Preparation

Model Training

Visualization

Dask

cuDF cuIO
Analytics

cuML
Machine Learning

cuGraph
Graph Analytics

PyTorch,
TensorFlow, MxNet
Deep Learning

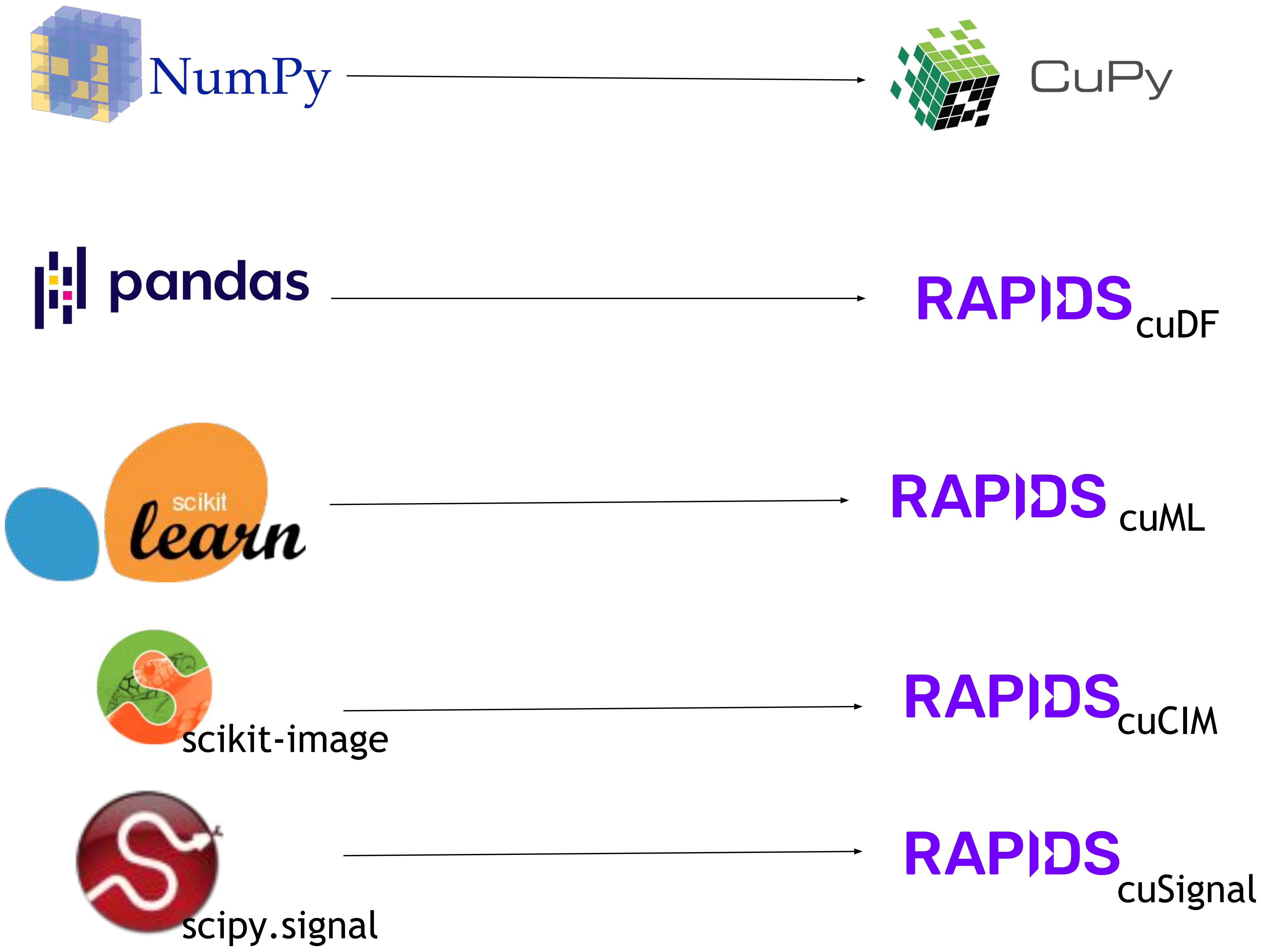
cuxfilter, pyViz,
plotly
Visualization

Apache
Arrow

GPU Memory

RAPIDS

GPU Accelerated Data Science



NumPy → CuPy

pandas → RAPIDS cuDF

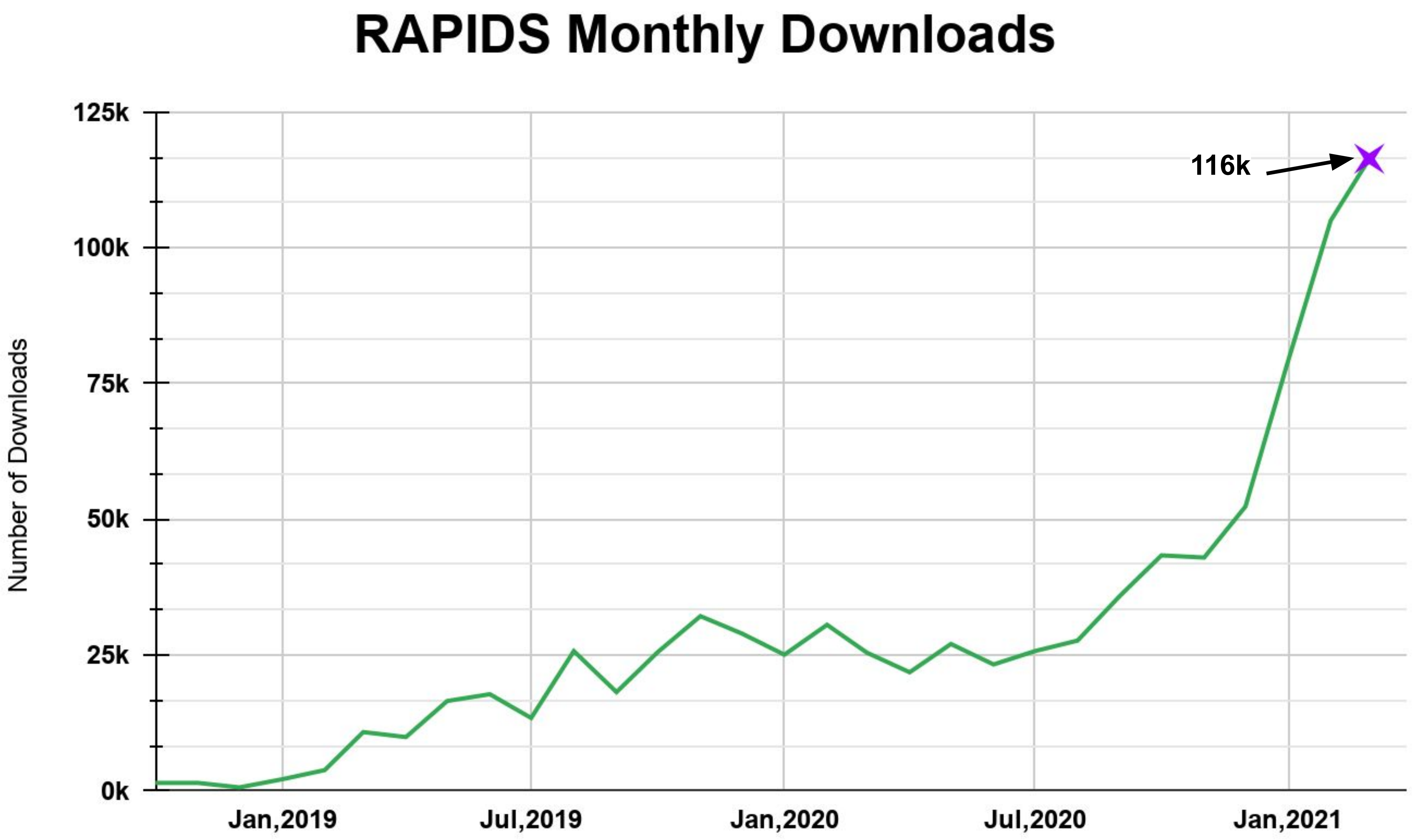
scikit learn → RAPIDS cuML

scikit-image → RAPIDS cuCIM

scipy.signal → RAPIDS cuSignal

Growth and Adoption

Growing community engagement



@RAPIDSai

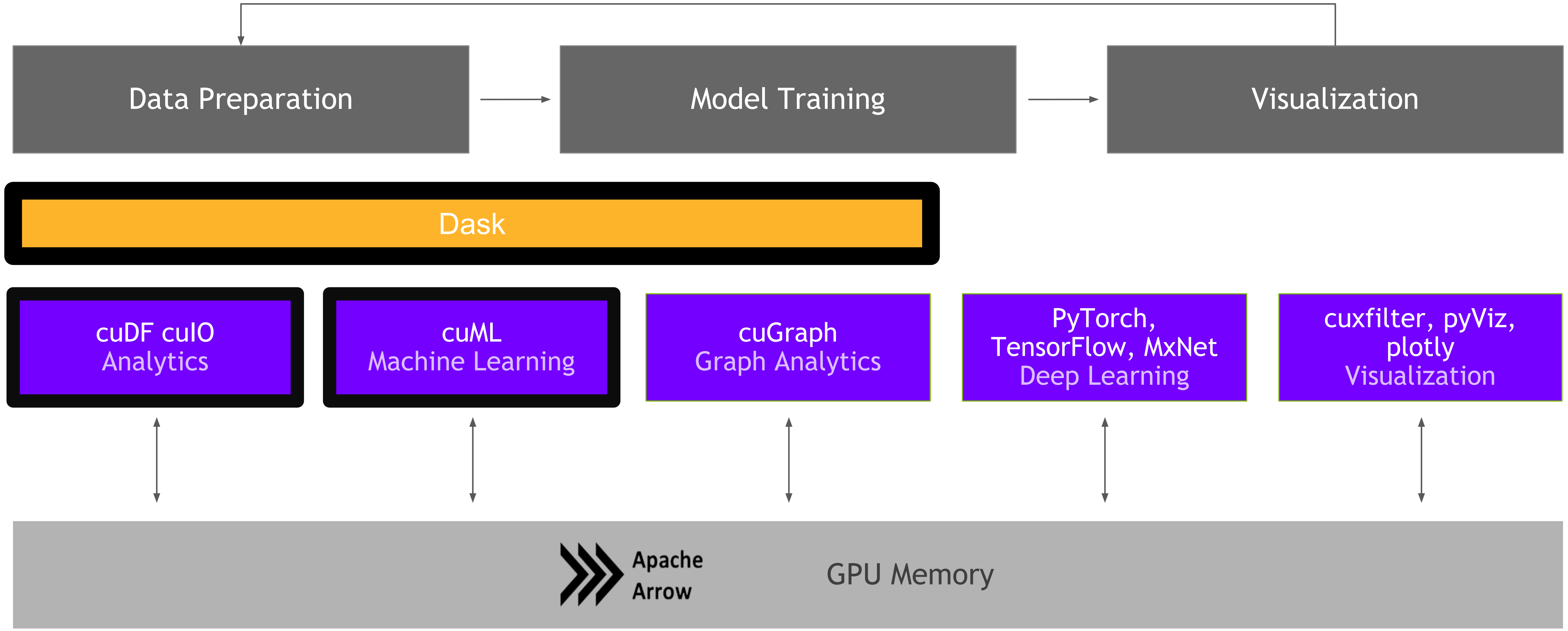
Followers: 10,345

github.com/rapidsai

Stars: 8K+ Contributors: 150+

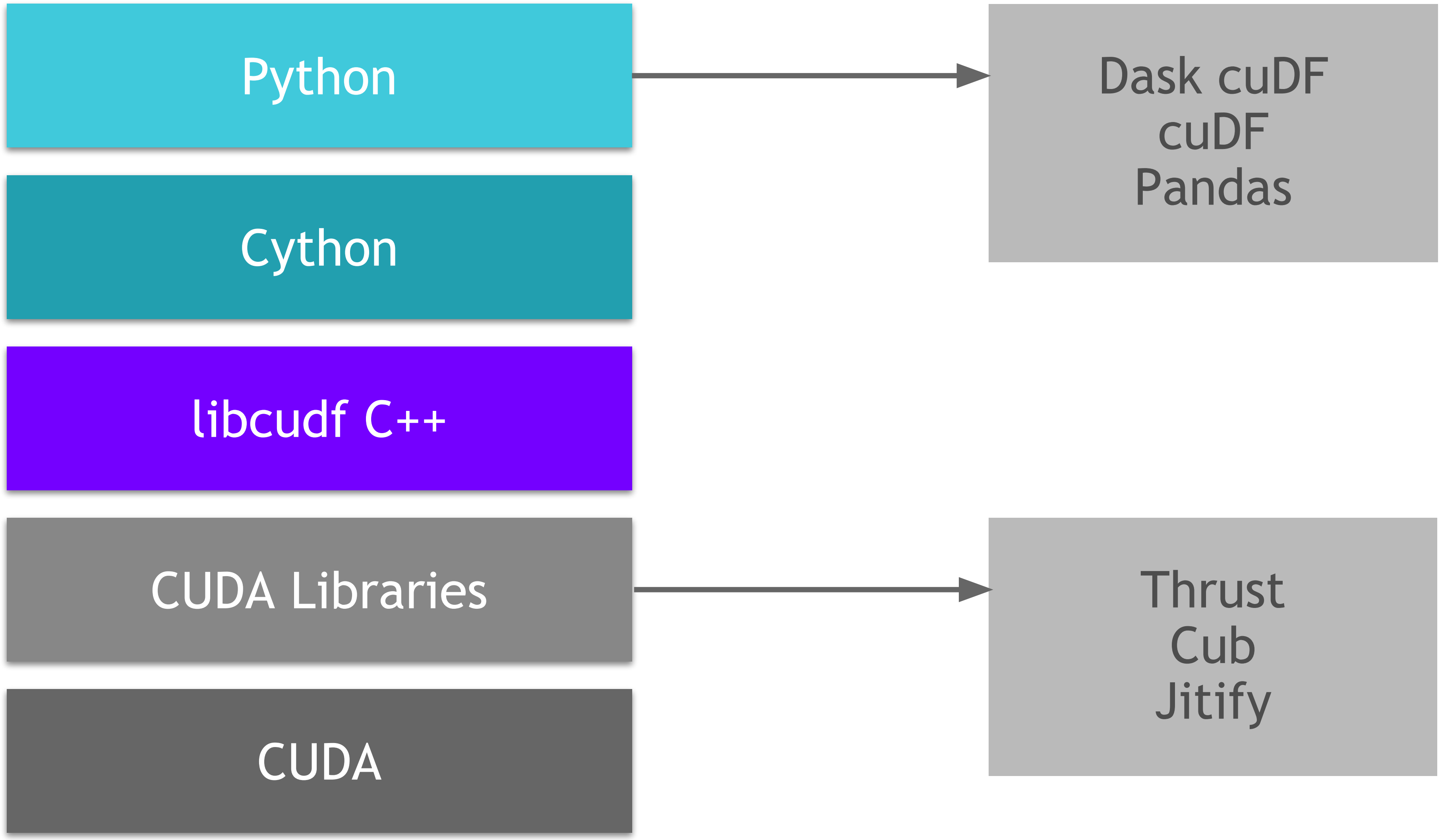
RAPIDS

End-to-End GPU Accelerated Data Science



What is cuDF?

Expandable platform for GPU data science

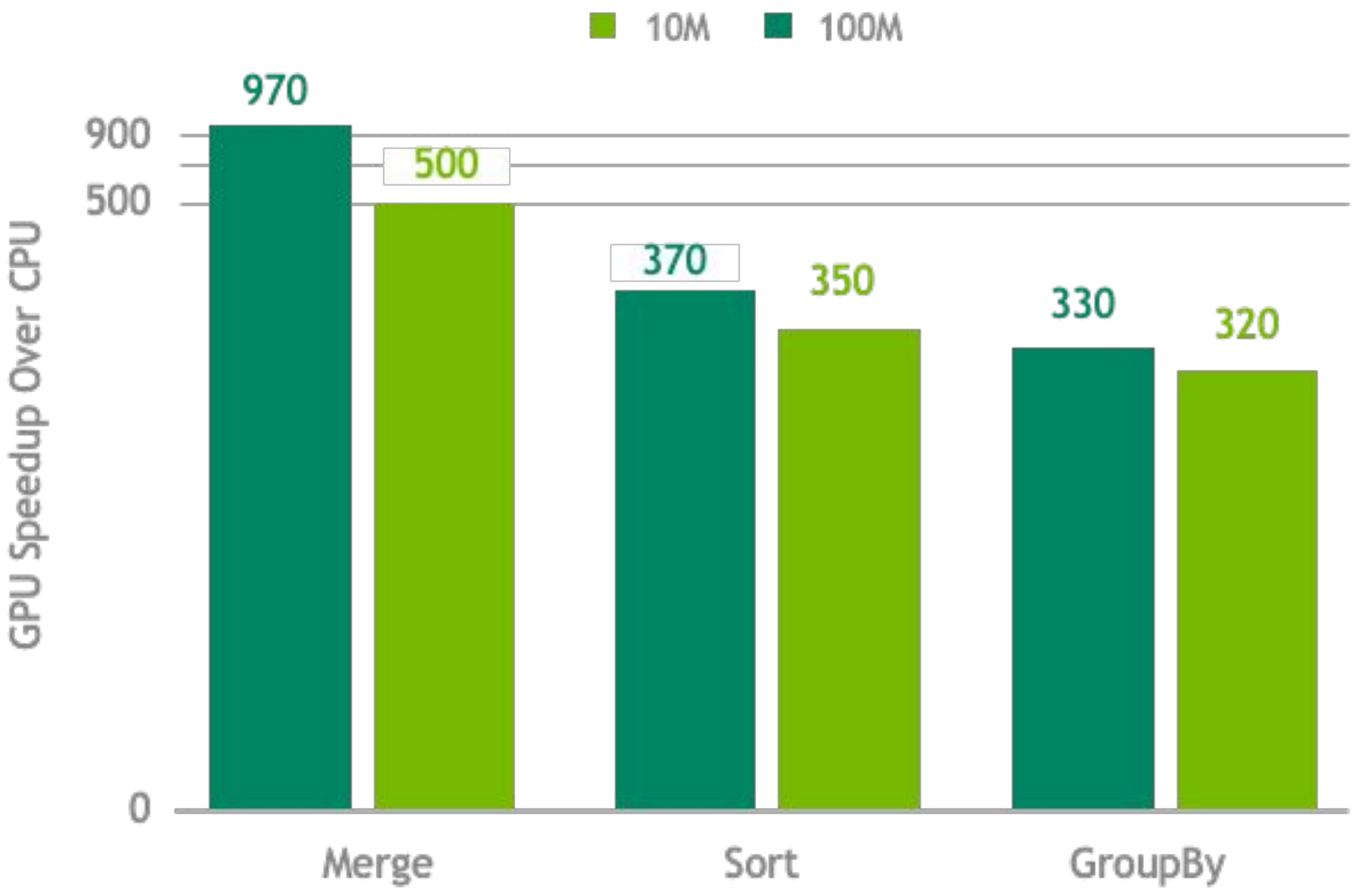


- Familiar pandas-like Python API
- Table (dataframe) and column types and algorithms
- High-performance C++ layer provides GPU-optimized CUDA kernels, data types, operations, and primitives
- CUDA/C++ is top level supported and used by many for integrating RAPIDS

ACCELERATED PRE-PROCESSING

A FAMILIAR EXPERIENCE FOR DATA ENGINEERS

RAPIDS provides a GPU DataFrame library with a pandas-like API while providing *significant* performance improvements.



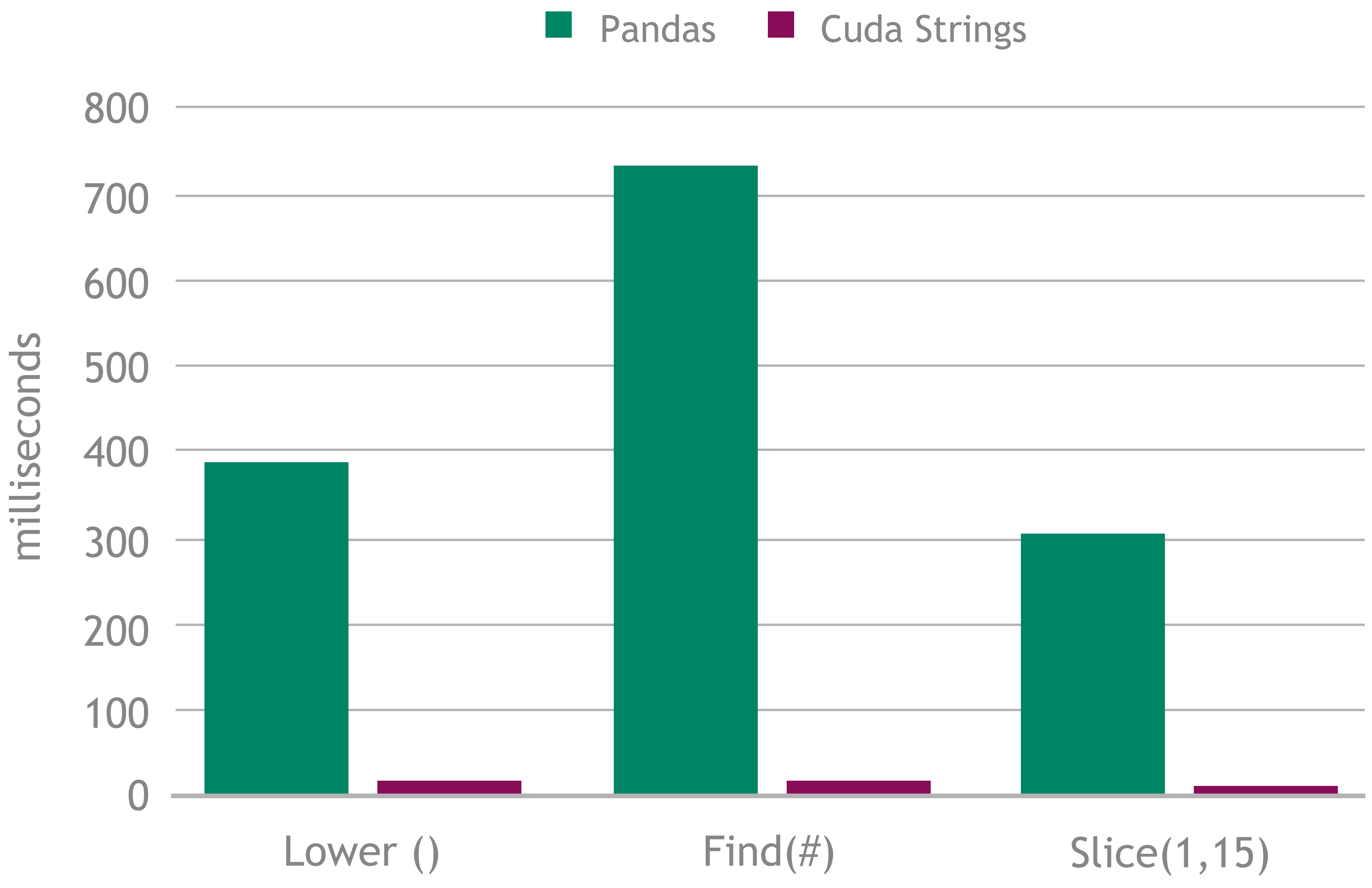
Single GPU Speed-Ups vs pandas

GPU: NVIDIA Tesla V100 32GB on DGX-1
CPU: Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz

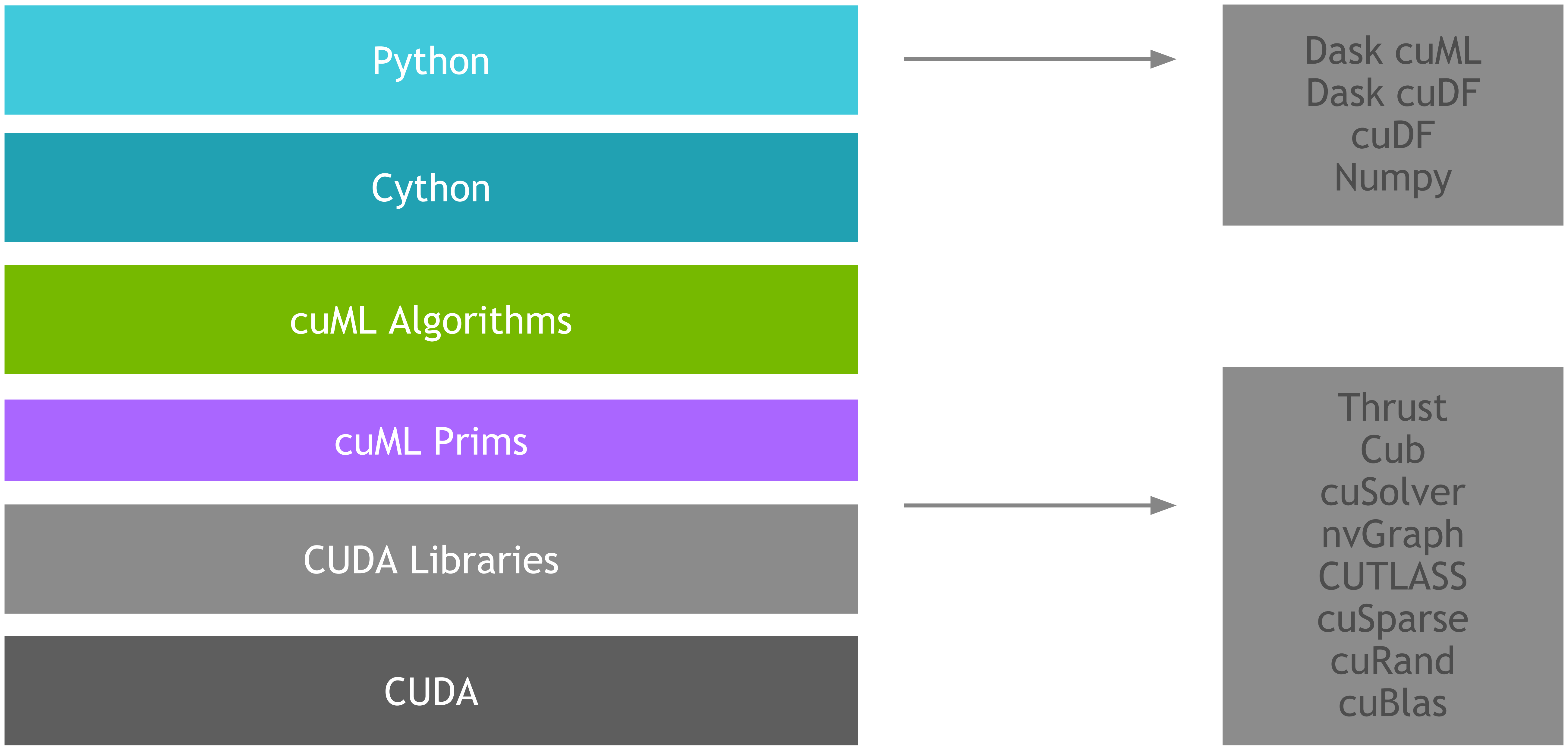
Comprehensive String Support

Backbone of ETL: Strings

- Regular Expressions
- Element-wise operations
    - Split, Find, Extract, Cat, Typecasting, etc…
- String GroupBys, Joins, Sorting, etc.
- Categorical columns fully on GPU
- NLP Preprocessors
    - Tokenizers, Normalizers, Edit Distance, Porter Stemmer, etc.



ML Technology Stack



Python

Cython

cuML Algorithms

cuML Prims

CUDA Libraries

CUDA

Dask cuML
Dask cuDF
cuDF
Numpy

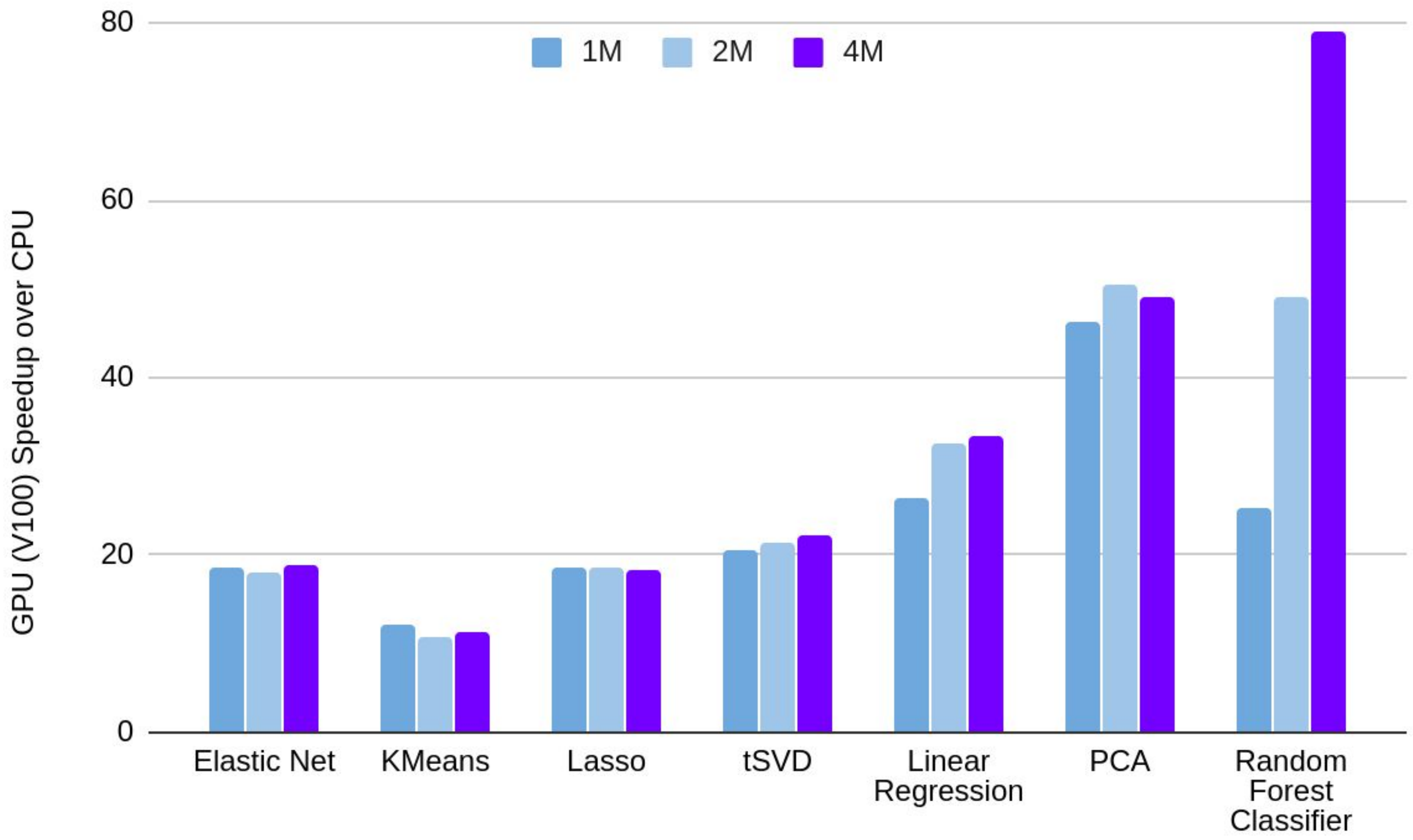
Thrust
Cub
cuSolver
nvGraph
CUTLASS
cuSparse
cuRand
cuBlas

ACCELERATED MACHINE LEARNING

GPU-POWER WITH THE FEEL OF SCIKIT-LEARN

RAPIDS provides a GPU ML library with a scikit-learn API while providing *significant* performance improvements.

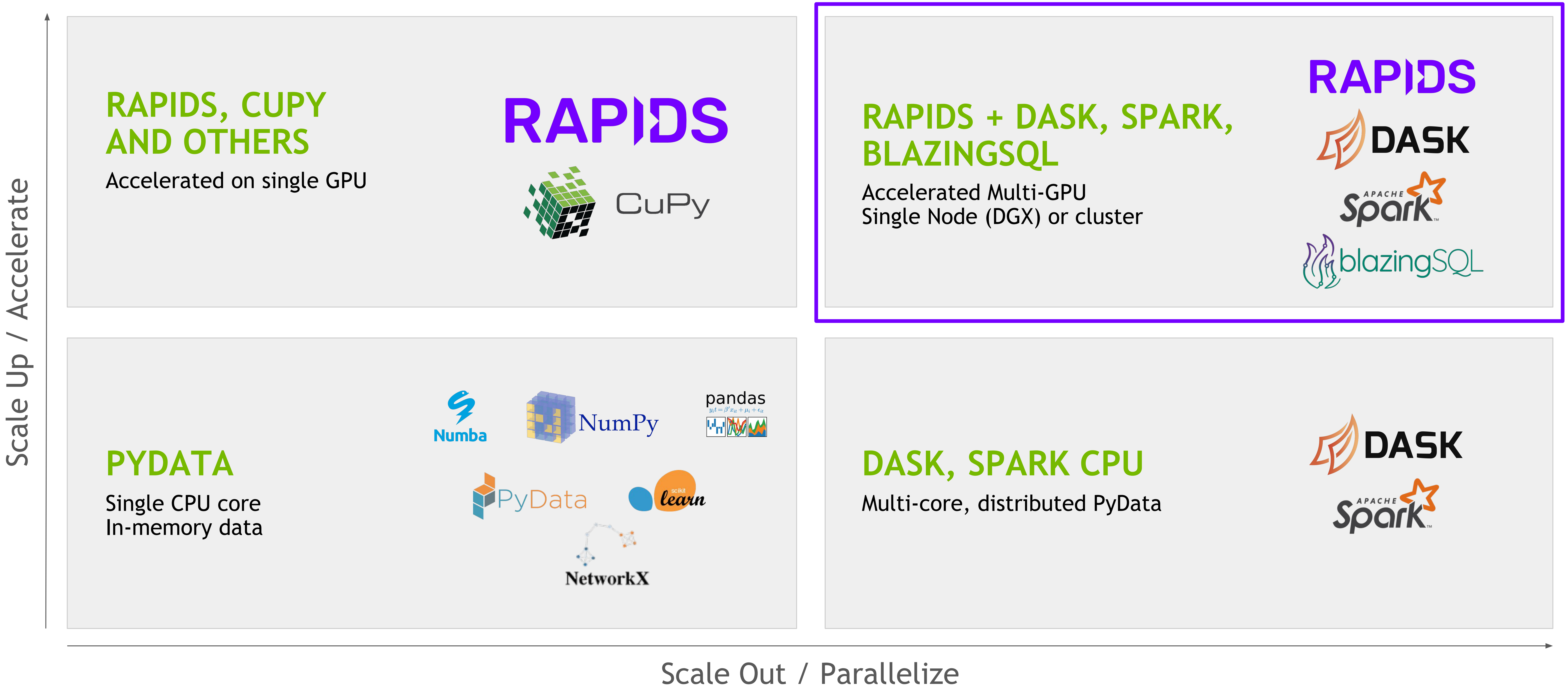
26 GPU-Accelerated Algorithms & Growing



1x V100 vs. 2x 20 Core CPUs (DGX-1, RAPIDS 0.15)

Scale Out with RAPIDS

Multiple options to scale, from multi-GPU to a whole cluster



RAPIDS, CUPY AND OTHERS

Accelerated on single GPU

RAPIDS + DASK, SPARK, BLAZINGSQL

Accelerated Multi-GPU
Single Node (DGX) or cluster

PYDATA

Single CPU core
In-memory data

DASK, SPARK CPU

Multi-core, distributed PyData

DASK

Python library for parallel computing

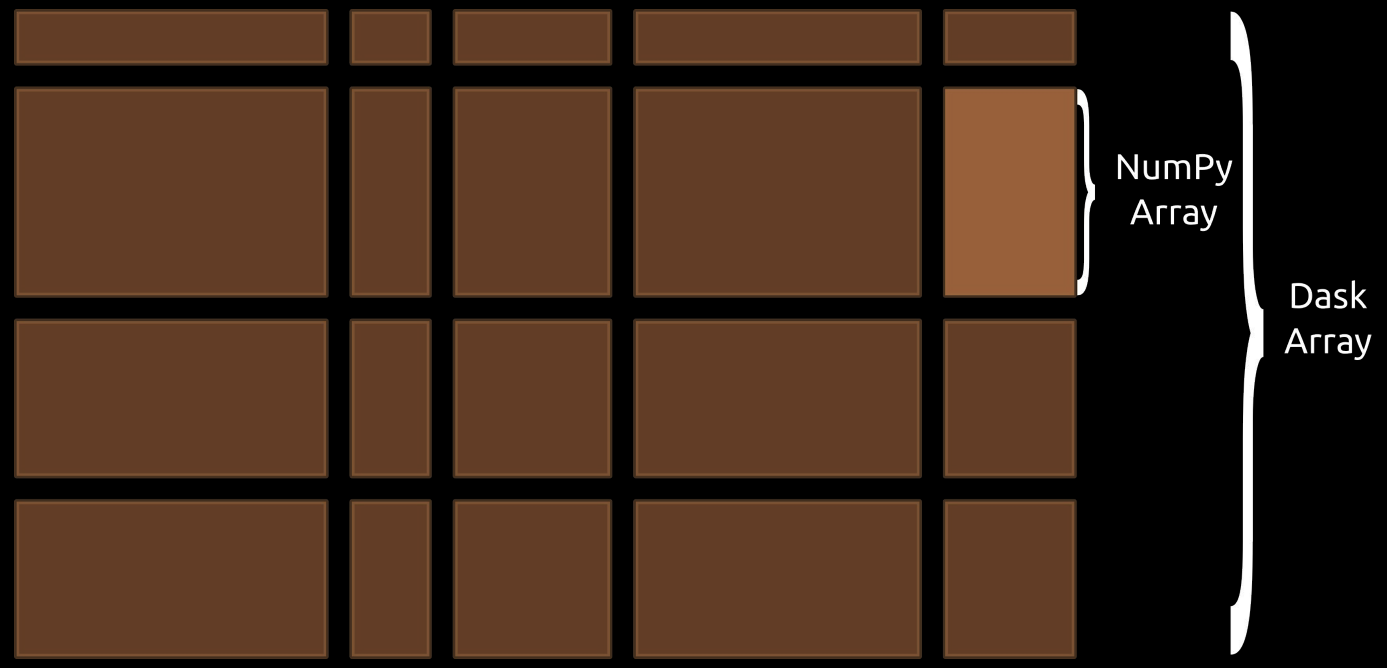
Scales Numpy, Pandas, and Scikit-Learn
Accelerates custom systems

Easy for beginners,
Secure and trusted for institutions

Dask accelerates the existing Python ecosystem

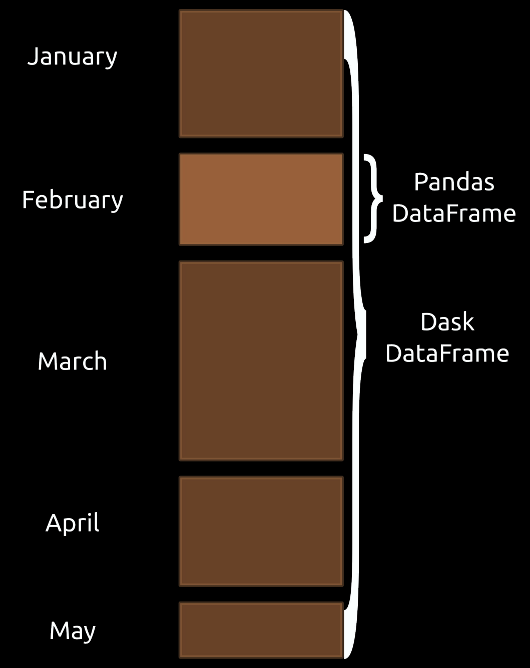
Built alongside the current community

Numpy



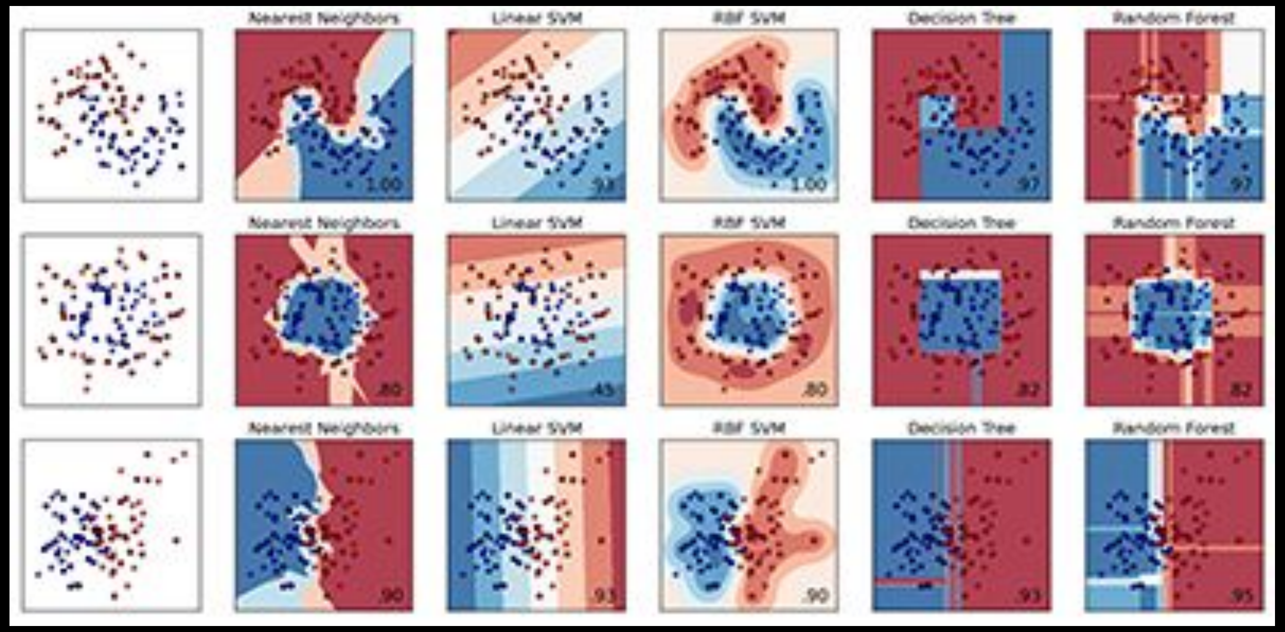
```
import numpy as np

x = np.ones((1000, 1000))
x + x.T - x.mean(axis=0)
```

Pandas



```
import pandas as pd

df = pd.read_csv("file.csv")
df.groupby("x").y.mean()
```

Scikit-Learn

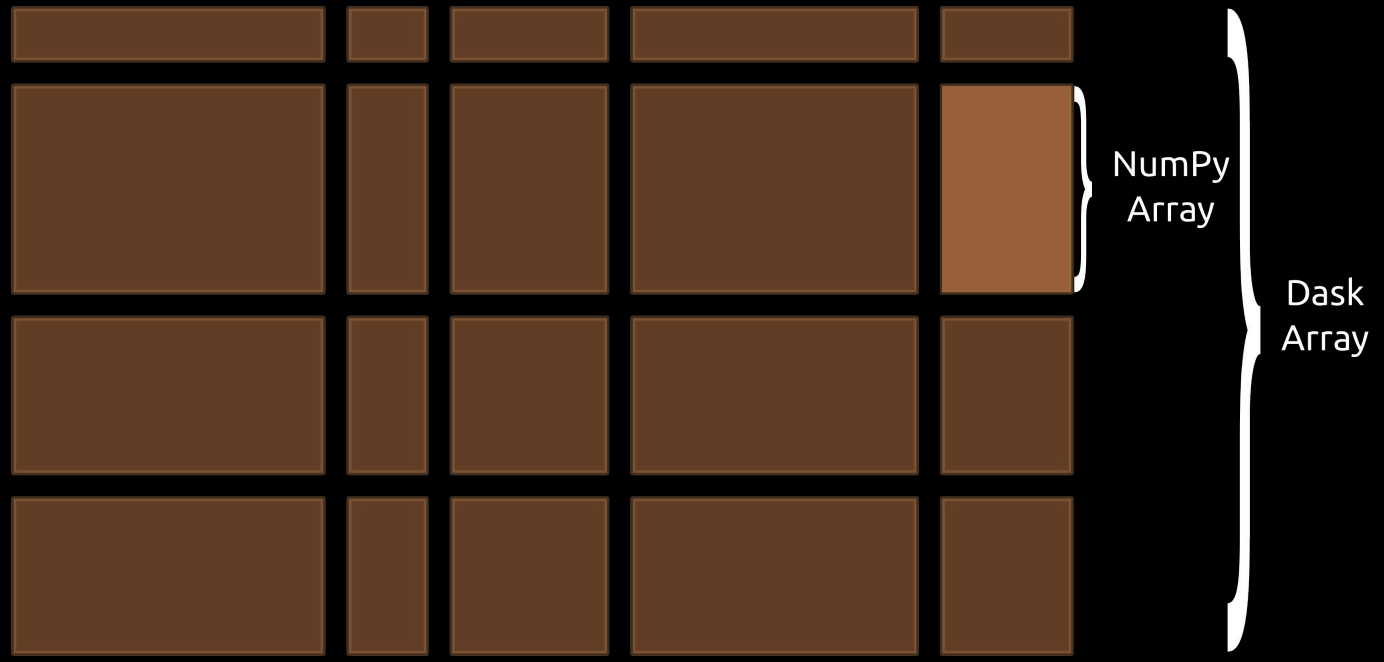


```
from scikit_learn.linear_model \
    import LogisticRegression

lr = LogisticRegression()
lr.fit(data, labels)
```

Dask accelerates the existing Python ecosystem

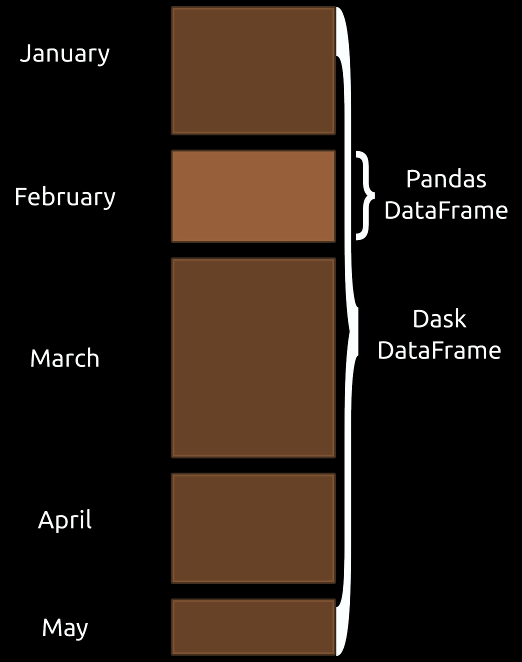
Built alongside the current community

Numpy



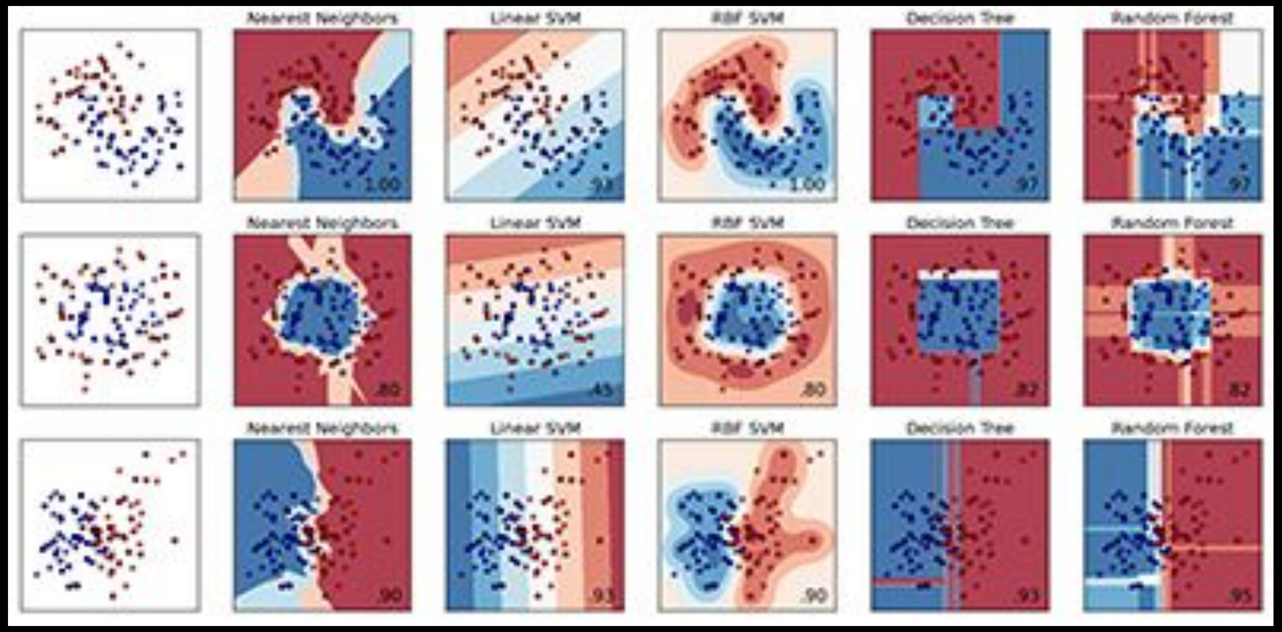
```
import dask.array as da

x = da.ones((10000, 10000))
x + x.T - x.mean(axis=0)
```

Pandas



```
import dask.dataframe as dd

df = dd.read_csv(“s3://*.csv”)
df.groupby(“x”).y.mean()
```

Scikit-Learn



```
from dask_ml.linear_model \
    import LogisticRegression

lr = LogisticRegression()
lr.fit(data, labels)
```

Parallelize existing complex code

Dask scales existing codebases with modest changes

```
def f(data, model) -> pd.DataFrame:
    …

def g(data, model) -> pd.DataFrame:
    …


results = []

for x in A:
  for y in B:
    if x < y:
       results.append(f(x, y))
    else:
       results.append(g(x, y))
```

Many codebases have opportunities for parallelism

But the problem doesn’t look like a big array or big dataframe

Parallelize existing complex code

Dask scales existing codebases with modest changes

```
@dask.delayed
def f(data, model) -> pd.DataFrame:
    …
@dask.delayed
def g(data, model) -> pd.DataFrame:
    …


results = []

for x in A:
  for y in B:
    if x < y:
       results.append(f(x, y))
    else:
       results.append(g(x, y))

results = dask.compute(results)
```

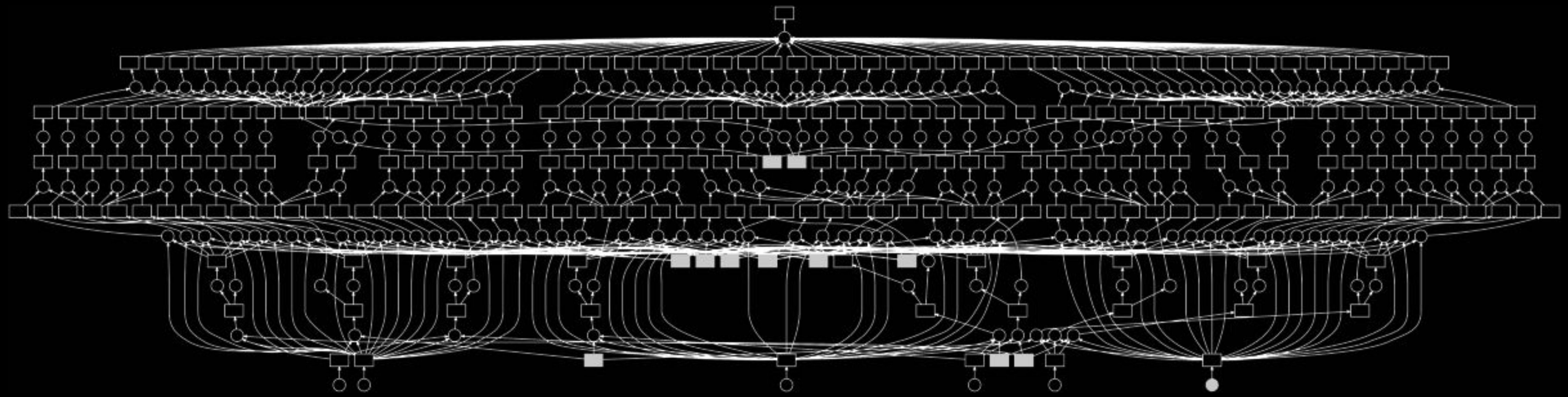
Dask Delayed adds parallelism without changing existing logic.

Dask lazily traverses your code to build a recipe for future execution.

Parallelize existing complex code

Dask scales existing codebases with modest changes

```
@dask.delayed
def f(data, model) -> pd.DataFrame:
    …
@dask.delayed
def g(data, model) -> pd.DataFrame:
    …


results = []

for x in A:
  for y in B:
    if x < y:
       results.append(f(x, y))
    else:
       results.append(g(x, y))

results = dask.compute(results)
```



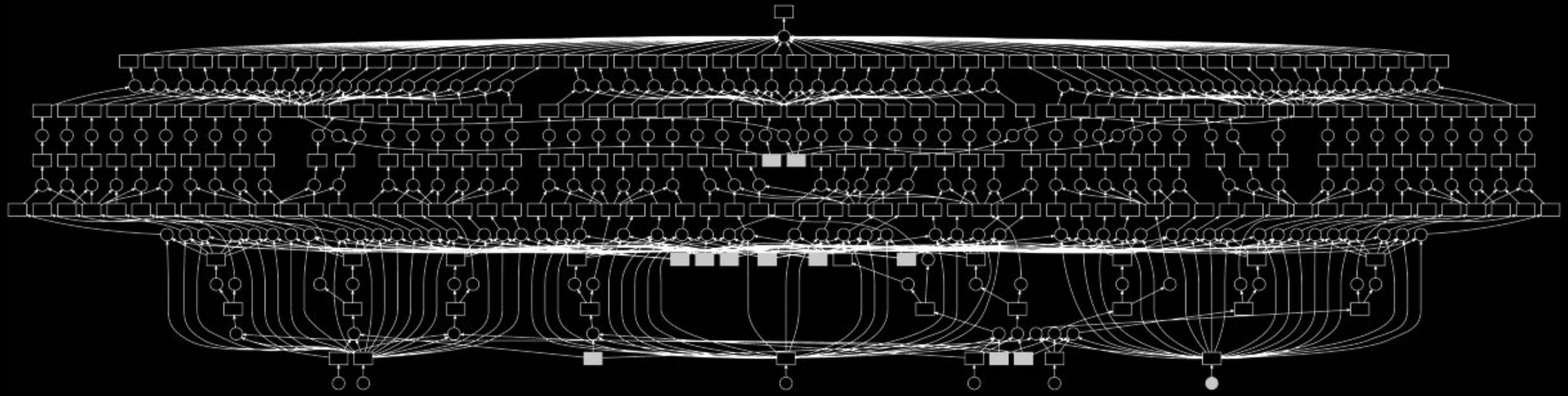
Your code creates a task graph for future execution.
Each node is one Python function.

DASK

Parallelize existing complex code

Dask scales existing codebases with modest changes

```
@dask.delayed
def f(data, model) -> pd.DataFrame:
    …
@dask.delayed
def g(data, model) -> pd.DataFrame:
    …


results = []

for x in A:
  for y in B:
    if x < y:
       results.append(f(x, y))
    else:
       results.append(g(x, y))

results = dask.compute(results)
```



Dask then executes that graph on parallel hardware

DASK

Dask deploys on all major resource managers

Cloud, HPC, or Yarn, it’s all the same to Dask

Cloud

aws

Google Cloud Platform

Microsoft Azure

```
cluster = KubeCluster()
cluster = ECSCluster()

df = dd.read_parquet(...)
```

HPC

```
cluster = PBSCluster()
cluster = LSFCluster()
cluster = SLURMCluster()
...
df = dd.read_parquet(...)
```

Hadoop/Spark

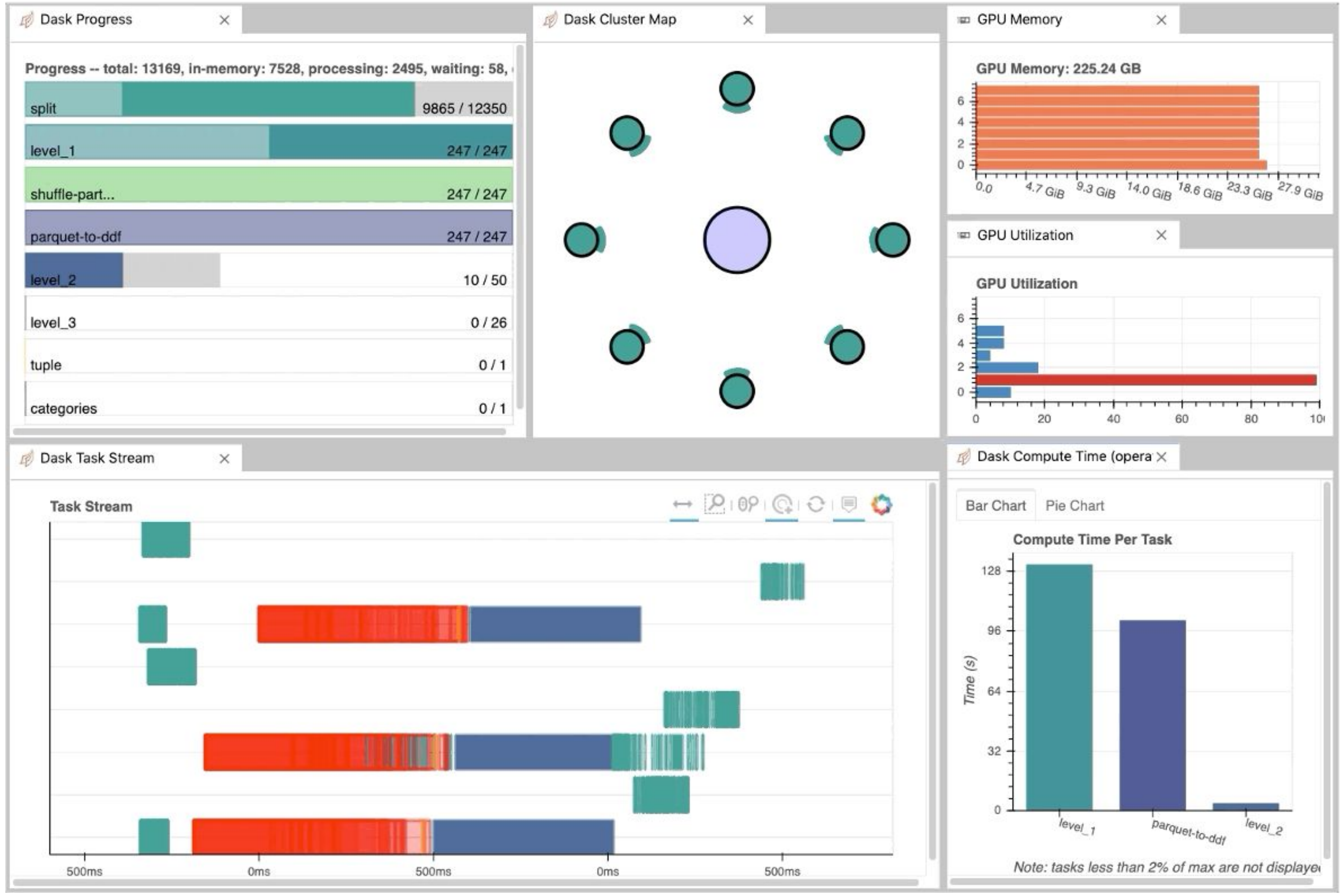
hadoop

```
cluster = YarnCluster()




df = dd.read_parquet(...)
```

Dask + RAPIDS

PyData-native scalable analytics

- **Deployable**: Kubernetes, Yarn, SLURM
- **PyData native**: Easy migration, built on top of NumPy, Pandas, Scikit-learn
- **Easy scalability**: Easy to install; scales to thousands of nodes
- **Popular**: most Common parallelism framework in PyData and SciPy community



Progress -- total: 13169, in-memory: 7528, processing: 2495, waiting: 58,

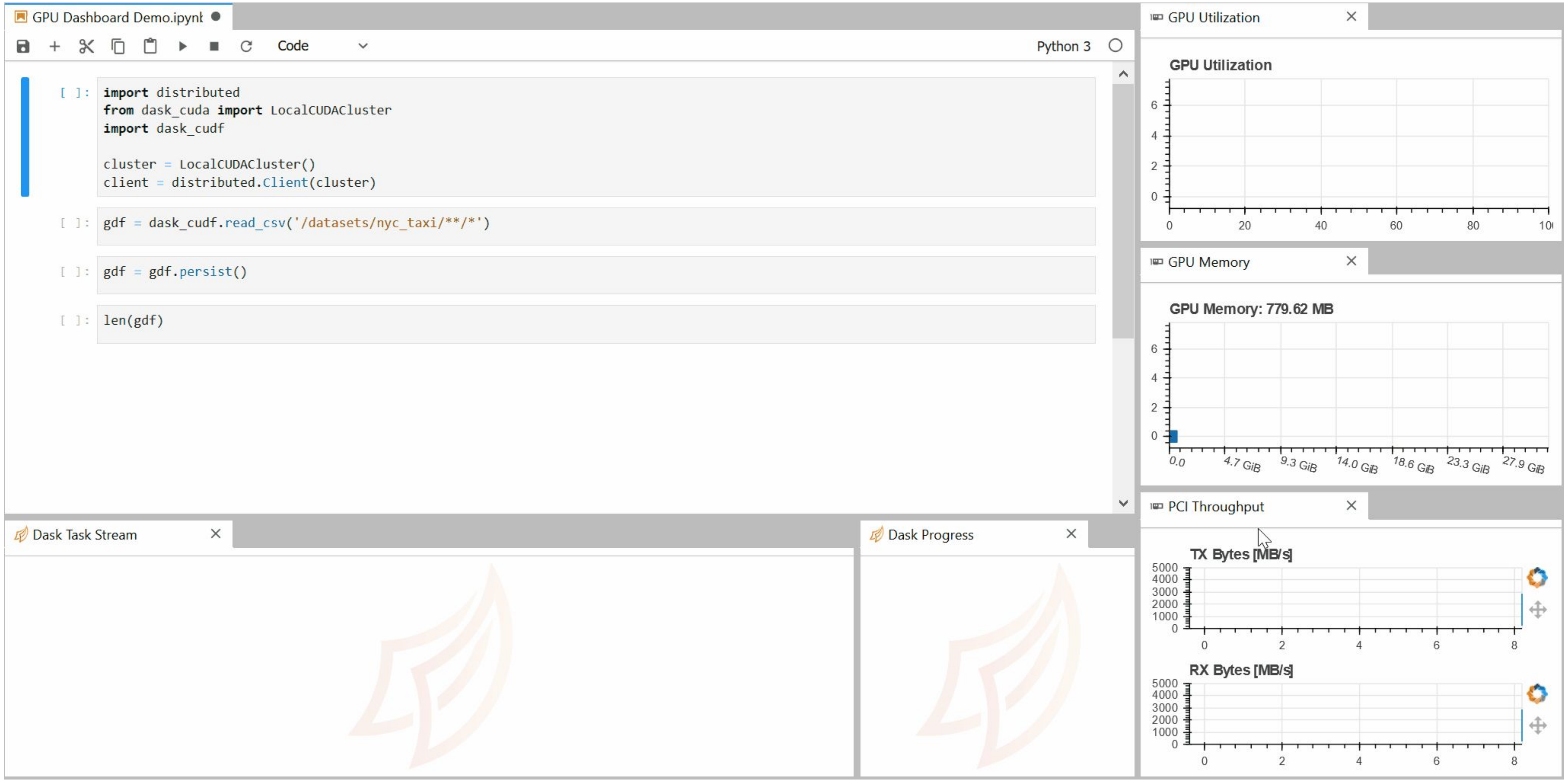
split	9865 / 12350
level_1	247 / 247
shuffle-part...	247 / 247
parquet-to-ddf	247 / 247
level_2	10 / 50
level_3	0 / 26
tuple	0 / 1
categories	0 / 1

GPU Memory: 225.24 GB

Note: tasks less than 2% of max are not displaye

RAPIDS Dev Environment

JupyterLab + Friends



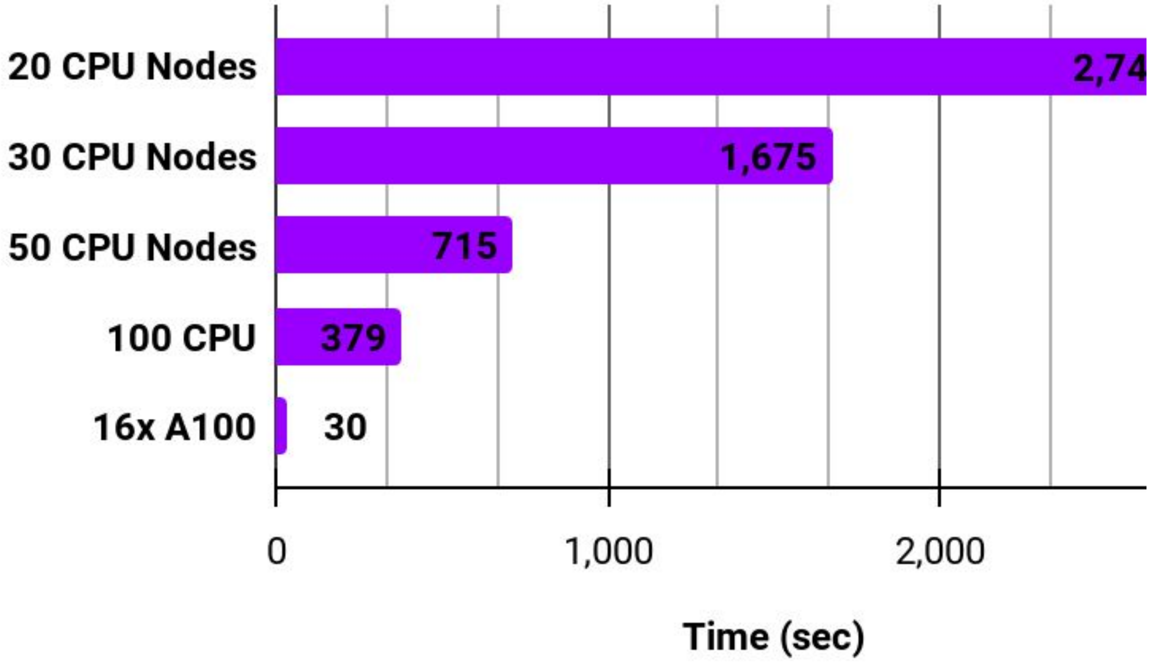
```
[ ]: import distributed
     from dask_cuda import LocalCUDACluster
     import dask_cudf

     cluster = LocalCUDACluster()
     client = distributed.Client(cluster)

[ ]: gdf = dask_cudf.read_csv('/datasets/nyc_taxi/**/*')

[ ]: gdf = gdf.persist()

[ ]: len(gdf)
```

- JupyterLab
- Dask Extension
- NVDashboard Extension

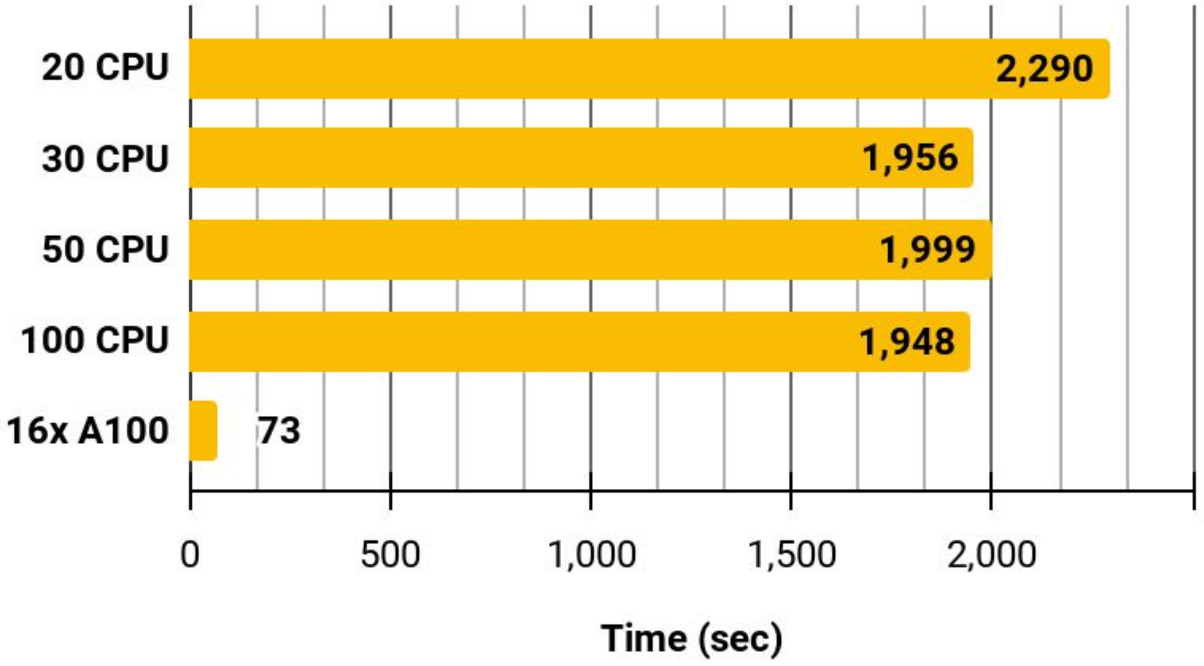
Faster Speeds, Real World Benefits

Faster Data Access, Less Data Movement

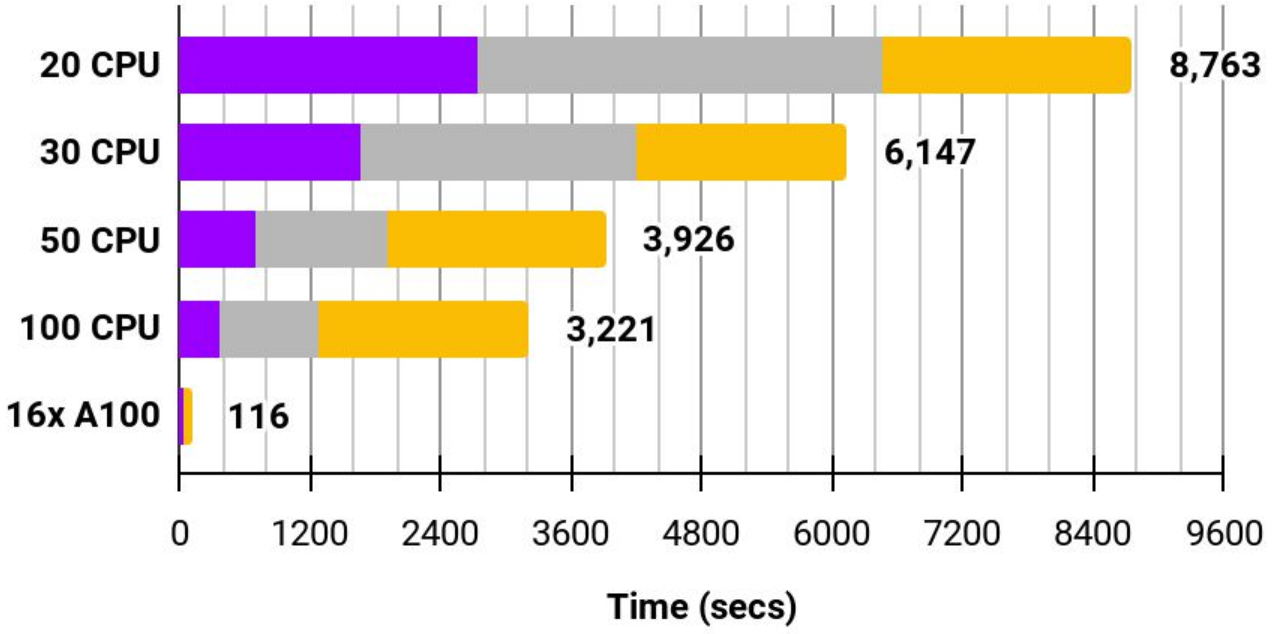
cuIO/cuDF - Load and Data Preparation



XGBoost Machine Learning



End-to-End



Time in seconds (shorter is better)

■ cuIO/cuDF (Load and Data Prep) ■ Data Conversion ■ XGBoost

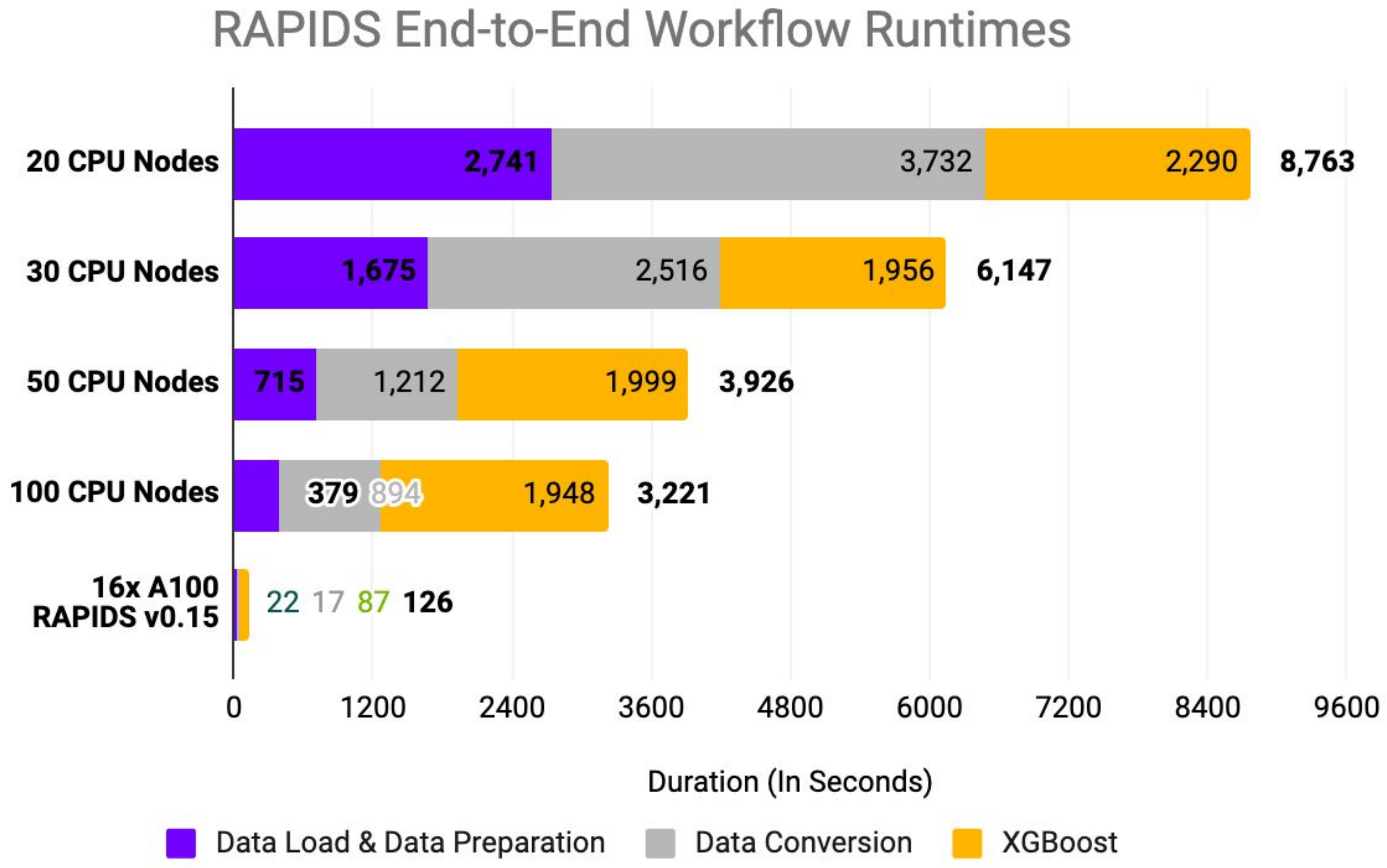
Benchmark	CPU Cluster Configuration	A100 Cluster Configuration	RAPIDS Version
200GB CSV dataset; Data prep includes joins, variable transformations	CPU nodes (61 GiB memory, 8 vCPUs, 64-bit platform), Apache Spark	16 A100 GPUs (40GB each)	RAPIDS 0.19

RAPIDS/Dask End-to-End Performance

Reducing Data Science Processes from Hours to Seconds

RAPIDS delivers massive speed-ups across the end-to-end data science lifecycle. Conducting benchmarks in a commercial cloud environment, we're able to get incredible performance running a common ML model training pipeline.

Between loading and cleansing data, engineering features, and training a classifier using a 200GB CSV dataset, a RAPIDS-based pipeline completed these operations in ***just over two minutes***. The same process takes two and half hours on a similar CPU-configuration.



16

A100s Provide More Power than 100 CPU Nodes

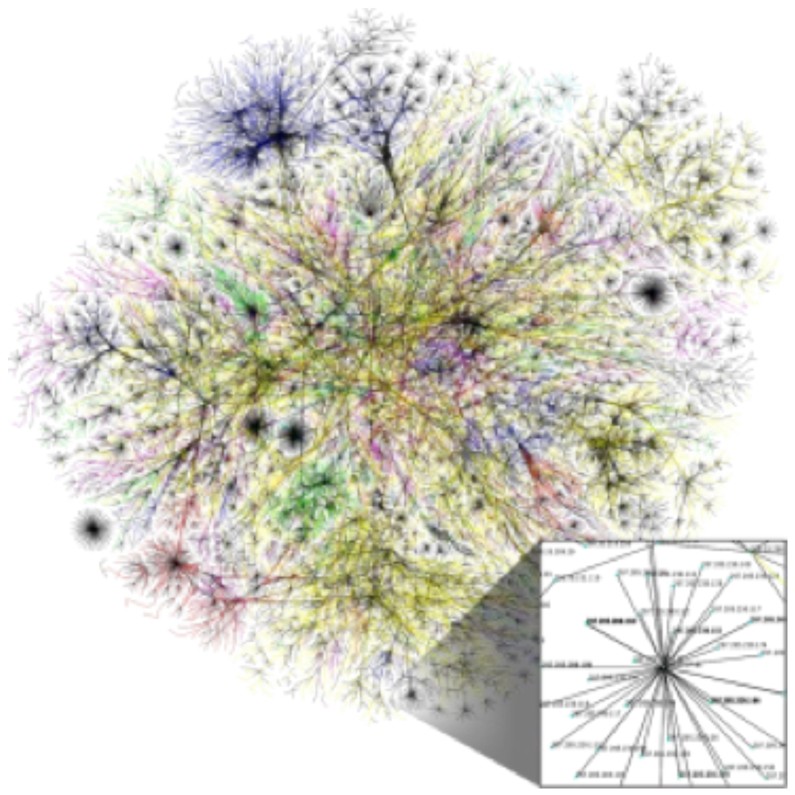
70x

Faster Performance than Similar CPU Configuration

20x

More Cost-Effective than Similar CPU Configuration

*CPU approximate to n1-highmem-8 (8 vCPUs, 52GB memory) on Google Cloud Platform. TCO calculations-based on Cloud instance costs.



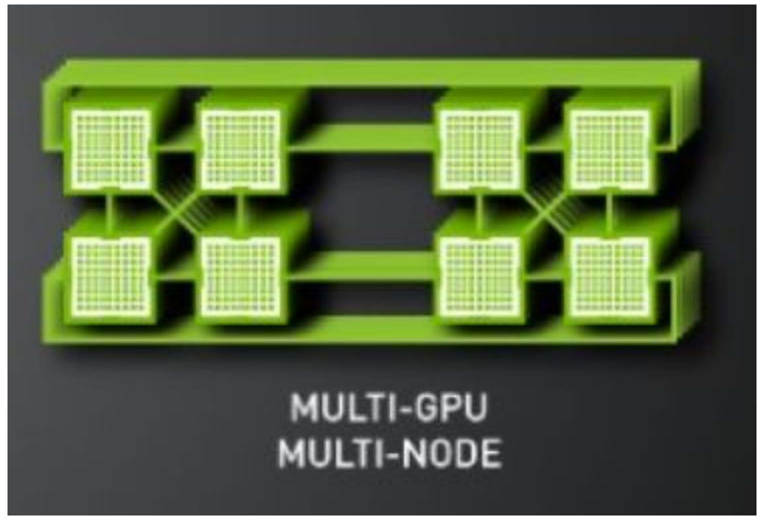
cuGraph - Algorithms

GPU-accelerated NetworkX

Traveling Salesman	Routing
Minimum Spanning Tree Maximum Spanning Tree	Tree
Graph Classes Subgraph Extraction **Egonet**	Structure
Force Atlas 2 Hungarian Algorithm	Other
Renumbering Auto-Renumbering NetworkX converters	Utilities

Community	Spectral Clustering - Balanced Cut and Modularity Maximization Louvain (Multi-GPU) and Leiden Ensemble Clustering for Graphs KCore and KCore Number Triangle Counting K-Truss
Components	Weakly Connected Components Strongly Connected Components
Link Analysis	Page Rank (Multi-GPU) Personal Page Rank (Multi-GPU) HITS
Link Prediction	Jaccard Weighted Jaccard Overlap Coefficient
Traversal	Single Source Shortest Path (SSSP) (Multi-GPU) Breadth First Search (BFS) (Multi-GPU)
Centrality	Katz (Multi-GPU) Betweenness Centrality (Vertex and Edge)

Scaling and Expanding Graph Analytics

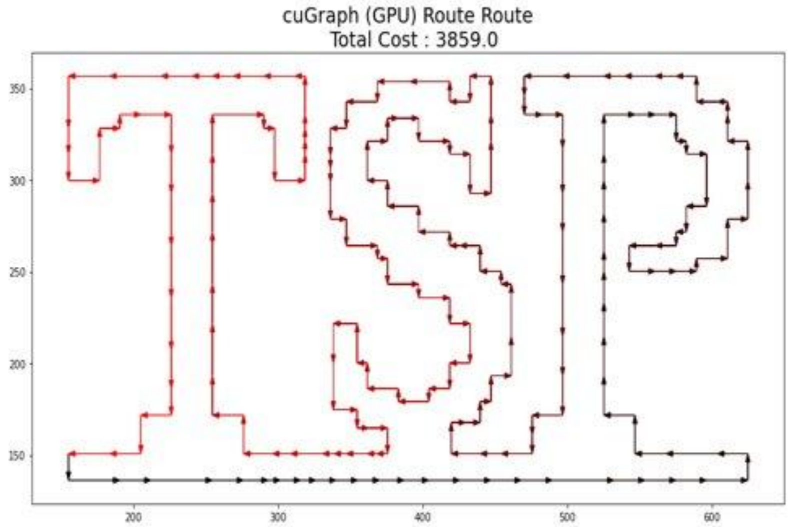


Multi-node, Multi-GPU Scaling

New graph primitives will underpin all algorithms

PageRank performance up to 180x faster than CPU

New 2d partitioning methods for large graphs

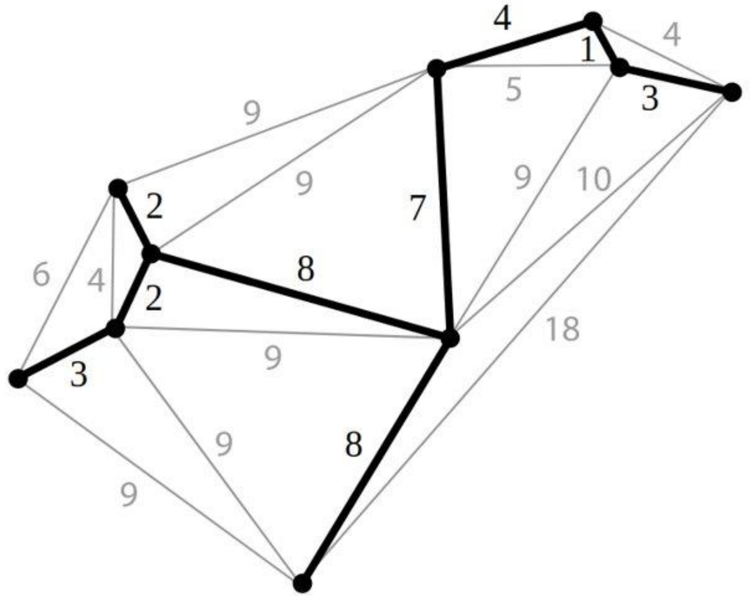


Traveling Salesperson Solver

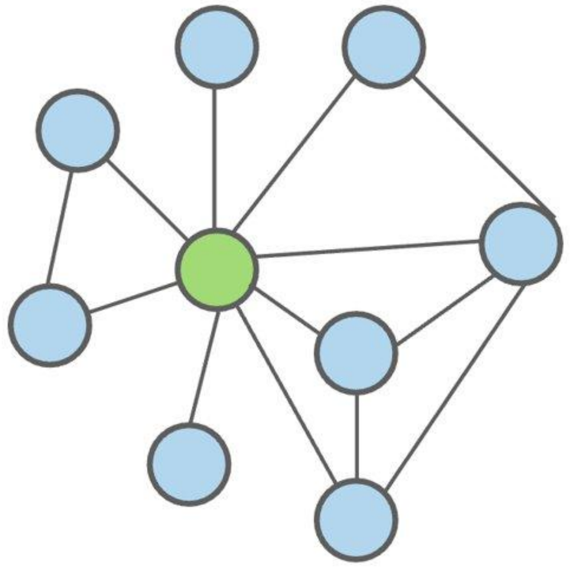
Up to 32x faster than CPU alternatives

NetworkX
Network Analysis in Python

Improved NetworkX API Compatibility



Minimum Spanning Tree



EGONet

Visualization and NodeJS

Plot.ly Dash

A Python visualization framework able to use RAPIDS libraries for notebooks and hosted dashboard applications.

Ideal for purpose built analytics applications, but also useful for notebook workflows.

Integrated RAPIDS backend for large datasets.

RAPIDS cuXFilter

A Python notebook based crossfilter dashboard library, using cuDF. Incorporates many chart libraries such as Datashader, HvPlot, Holoviews, Bokeh, and Deck.gl.

Easy integration with RAPIDS notebook based workflows.

RAPIDS Node.js (early alpha)

Experimental Node.js Javascript bindings for RAPIDS and related GPU libraries. Usable for both visualization and general-purpose compute on Node.js platforms.

Build End-to-End Data Science Applications

Leverage RAPIDS Core Libraries to Build Custom Solutions

	Description	Similar To	Problem Domain	Maturity	Performance	Example User	API Docs
cuDF	Dataframes & ETL	pandas	Data Preparation	◕	◕	Walmart	Read the Docs
Apache Spark 3.0 Plugin	ETL	Apache Spark	Data Preparation	◔	◑	CLOUDERA	Read the Docs
BlazingSQL	ANSI SQL	SQL	Data Preparation	◕	◕	Oak Ridge National Laboratory	Read the Docs
cuML	Machine Learning	scikit-learn	Model Training	◕	◕	Capital One	Read the Docs
cuGraph	Graph Analytics	NetworkX	Model Training	◑	◕	VISA	Read the Docs
XGBoost	GBMs	XGBoost	Model training	●	●	Scotiabank	Read the Docs
RAPIDSViz	Large-Scale Visualization	Bokeh, DataShader, HoloViews	Visualization	◑	◕	plotly	Read the Docs

Use RAPIDS-Enabled Tools & Frameworks

High-Performance Solutions for a Wide Variety of Domains

	Description	Similar To	Problem Domain	Maturity	Performance	Example User	API Docs
CLX	Cyber log parsing & analytics	N/A	Cybersecurity	◐	◕	BEST BUY	Read the Docs
cuCIM	Image processing & analytics	scikit-image	Image Processing	◔	◐	QUANSIGHT	Read the Docs
cuSignal	Signal processing & analytics	N/A	Signal Processing	◐	◕	LOCKHEED MARTIN	Read the Docs
cuSpatial	Spatial processing & analytics	N/A	Spatial Data Processing	◔	◕	tsmc	Read the Docs
cuStreamz	Stream processing & analytics	Streamz & Kafka	Stream Processing	◐	◕	NVIDIA	Read the Docs
Node-RAPIDS	Server-side JavaScript	Node.js	Web Development	◔	◕	Technical Preview	Read the Docs
NVTabular	Feature engineering and data loading	N/A	Recommender Systems	◐	●	Postmates	Read the Docs

cuDF

Updates + Improvements

Optimizations

- Complex Hash Aggregations
- Character Parallel String Algorithms
- Parquet GPU Direct Storage Support

New Features

- List, Struct, Dictionary, And Decimal types and operations
- Expanded GroupBy Operations
- Improved API and Developer Docs

Build Infrastructure

- CUDA 11.2 Support And CUDA Enhanced Compatibility
- CMake Refactored for easier source builds

Upcoming Improvements

- Abstract Syntax Tree Evaluation
- ORC GPU Direct Storage
- Reduce Python Overheads

- Improved CUDA Stream Support
- Time Series Support
- Conditional Joins

- Upgrade to C++17
- CUDA 11.4

The Rapidly Growing RAPIDS Ecosystem

Supported, Used, & Extended by a Wide Variety of Partners

CONTRIBUTORS

ANACONDA. · Capital One · CuPy · Chainer · Deepwave Digital · UNROCK · NVIDIA. · QUANSIGHT · Walmart

ADOPTERS

anyscale · BEST BUY · BMW GROUP · Booz | Allen | Hamilton · Capital One · CLOUDERA · databricks · Deepwave Digital · DOMINO

graphistry · H2O.ai · IBM · iguazio · Informatica · kinetica · Inria · MAPR. · NASA · omni·sci

Preferred Networks · PyTorch · RAY · SaturnCloud · Spotify · splunk> · Uber · URSA LABS · VW · Walmart Global Tech

OPEN SOURCE

APACHE ARROW · blazingSQL · CuPy · DASK · HoloViz · node JS · nuclio · Numba · scikit learn · dmlc XGBoost

Dask/RAPIDS in HPC

Supported, Used, and Extended in Research and Academia

ATLAS EXPERIMENT
cnes
RIKEN
OAK RIDGE National Laboratory
Met Office
SLAC
NCAR
NERSC

Dask in HPC (Recording)

Dask in HEP (Recording)

Deploy RAPIDS Everywhere

Focused on Robust Functionality, Deployment, and User Experience

aws

Amazon SageMaker

Azure

Azure Machine Learning

Google Cloud

Alibaba Cloud

Cloud Dataproc

Bai du 百度

SaturnCloud

Kubeflow

COILED

Integration with major cloud providers | Both containers and cloud specific machine instances
Support for Enterprise and HPC Orchestration Layers

How to Get Started with RAPIDS

A Variety of Ways to Get Up & Running

More about RAPIDS

- Learn more at RAPIDS.ai
- Read the API docs
- Check out the RAPIDS blog
- Read the NVIDIA DevBlog

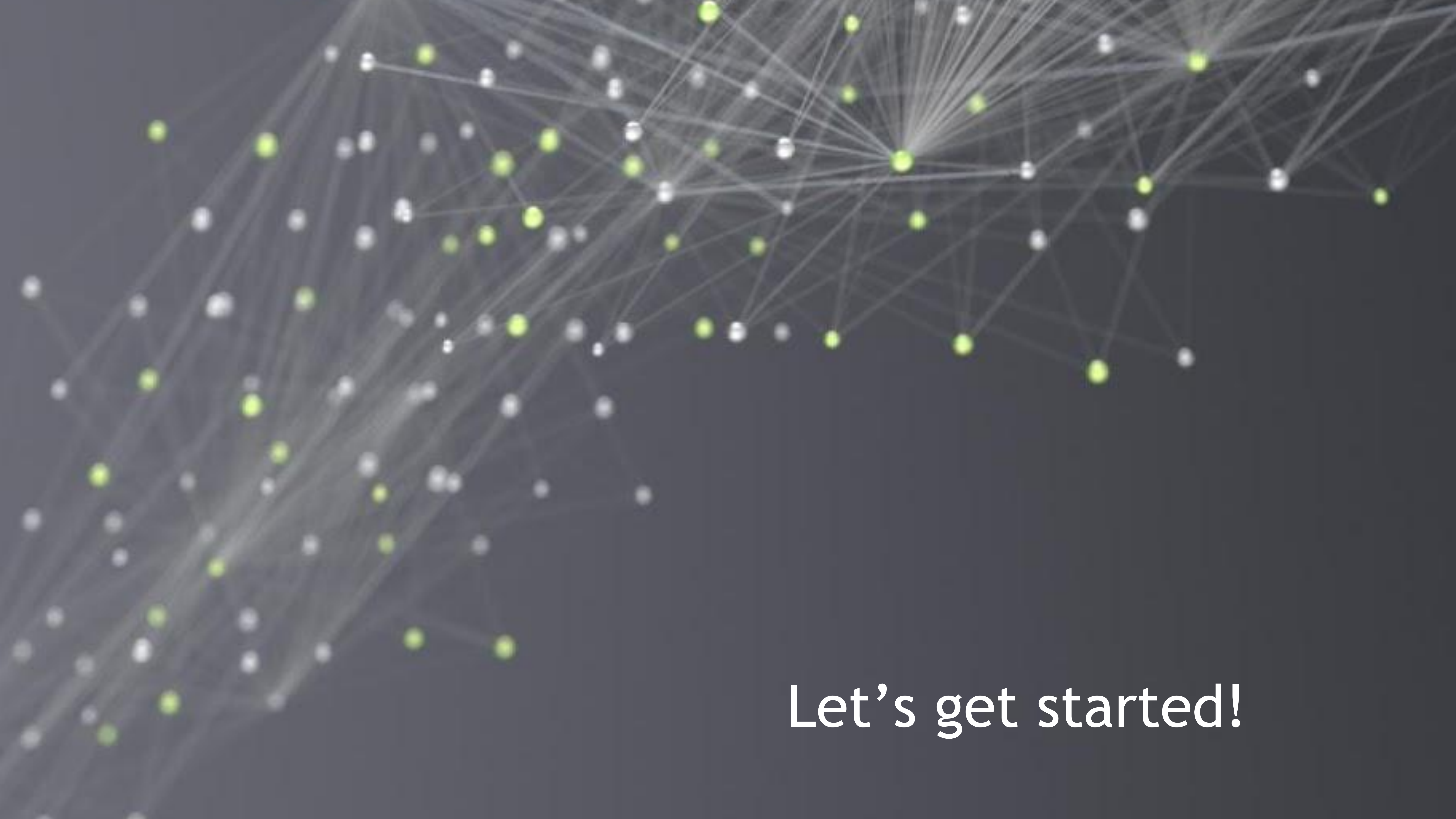
Self-Start Resources

- Get started with RAPIDS
- Deploy on the Cloud today
- Start with Google Colab
- Look at the cheat sheets

Discussion & Support

- Check the RAPIDS GitHub
- Use the NVIDIA Forums
- Reach out on Slack
- Talk to NVIDIA Services

Keep in touch with us on the Dask Slack workspace: link here



Let’s get started!