NVIDIA | DEEP LEARNING INSTITUTE

FUNDAMENTALS OF ACCELERATED DATA SCIENCE WITH RAPIDS

COURSE GOALS

Learn the core tools to use RAPIDS for everyday data science

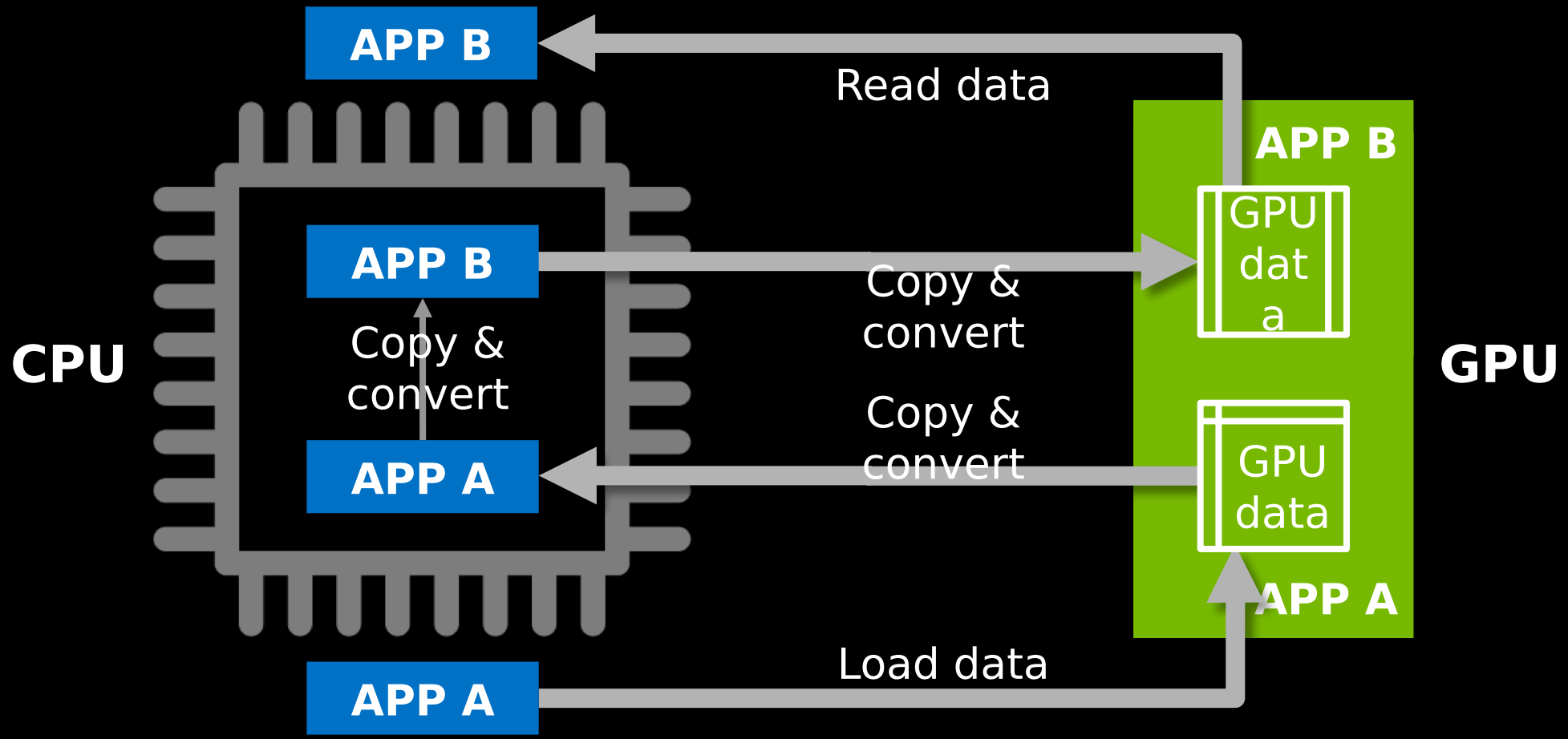
Understand RAPIDS' scalability from workstation and cluster to cloud and HPC

Build the foundations for you to learn RAPIDS capabilities now and in the future

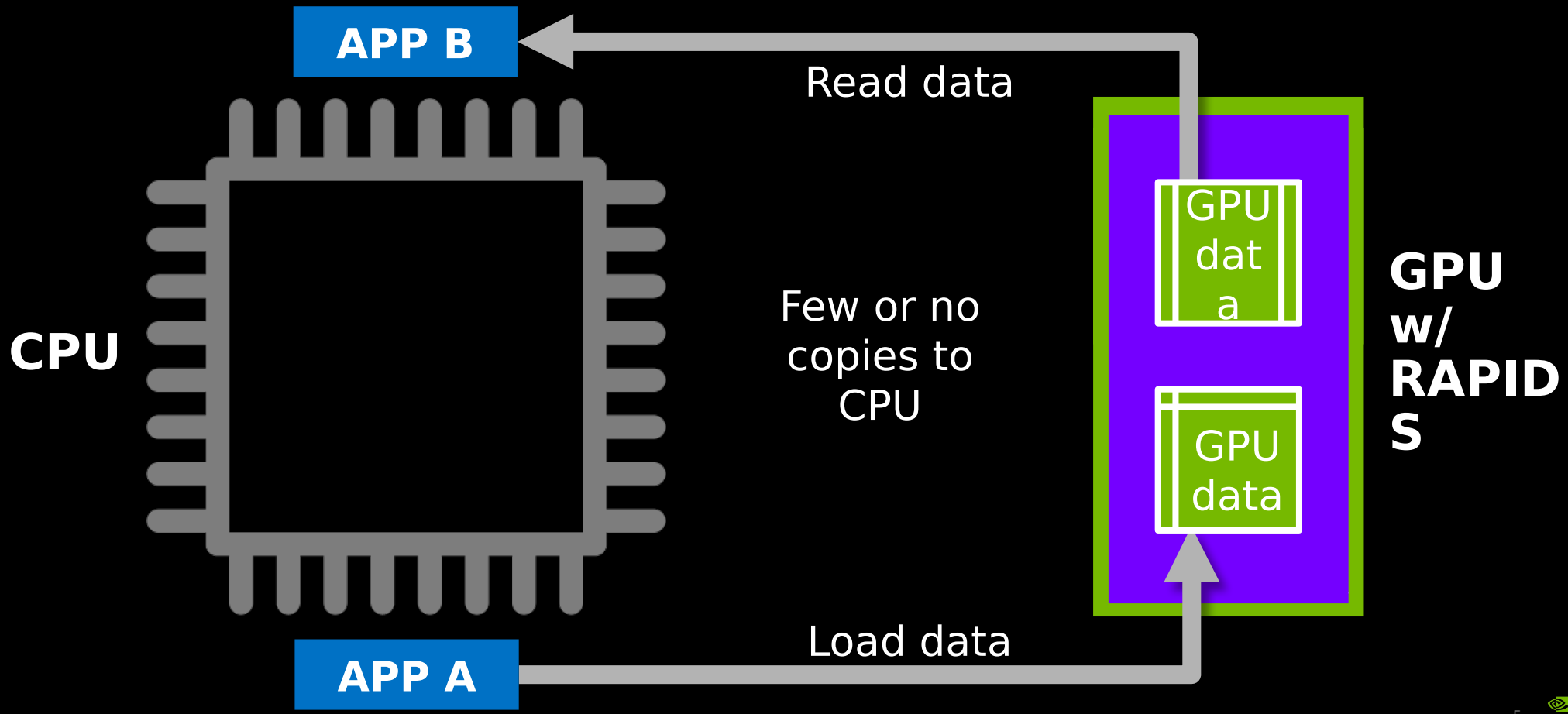
RAPIDS

FUNDAMENTALS

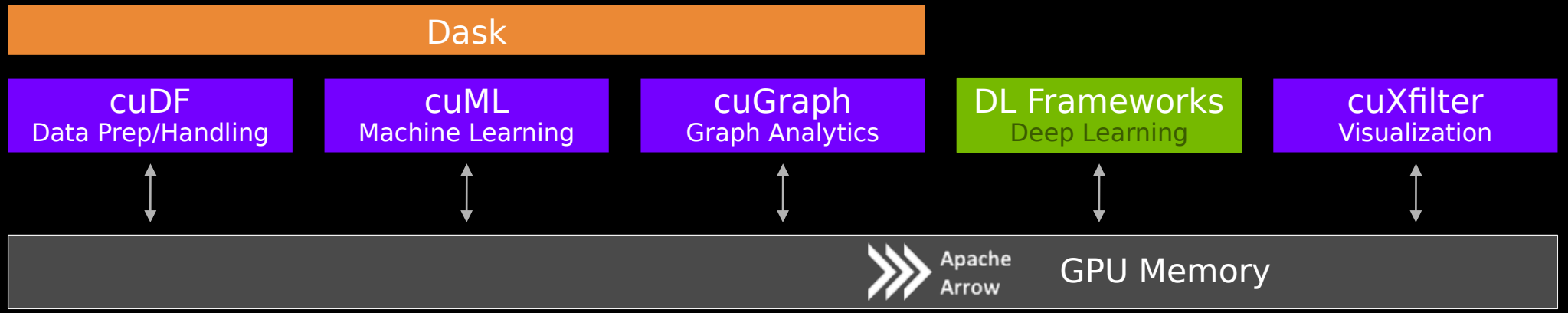
TRADITIONAL MODEL



RAPIDS MODEL



RAPIDS PLATFORM



Specialized package examples

- cuSpatial — Geospatial Analytics
- cuSignal — Signal Processing
- CLX — Cyber Analytics

DATA SCIENCE TOOLSETS

	CPU	GPU/ RAPIDS
Data handling	pandas	cuDF
Machine learning	scikit-learn	cuML
Graph analytics	NetworkX	cuGraph

	CPU	GPU/ RAPIDS
Viz	Bokeh/ Datashader	cuXfilter
Geospatial	GeoPandas / SciPy.spatial	cuSpatial
Signals	SciPy.signal	cuSignal
Cyber	cyberpandas	CLX

REQUIREMENTS

Appropriate OS: Ubuntu 16.04/18.04, CentOS/RHEL 7, Windows with WSL (preview)

NVIDIA Pascal™ GPU architecture or newer

CUDA 10.1.2/10.2/11.0, drivers, etc. (see rapids.ai)

Open source/flexible mindset

- Using v0.15 in this class
- New versions released regularly
- v1.0 TBD

RAPIDS

Source code on GitHub

https://github.com/rapidsai

Containers on NGC & Docker Hub

https://ngc.nvidia.com

Conda packages

https://anaconda.org/rapidsai

On-premises

rapids.ai

In the cloud

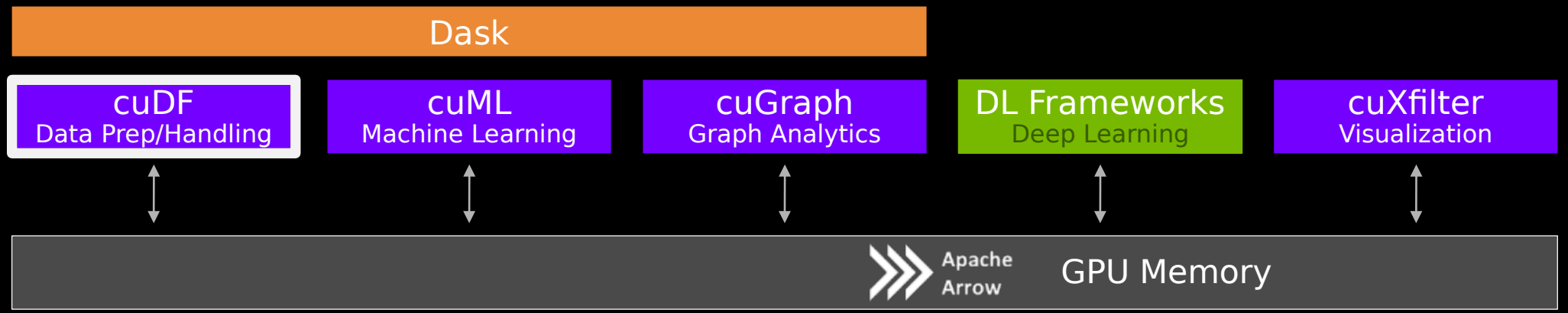
EXERCISE DATA

Fused and simulated from several sources

- Population data
  - Simulated from UK Census data on England and Wales, both from details (age, sex, given name, county) and aggregate statistics (geographic coordinates, employment)
- Road network data
  - Nodes (endpoints/junctions) and edges of the entire road network of Great Britain
- Epidemic data
  - Detailed hospital/clinic data from the UK National Health Service
  - Spread modeled on academic research on Ebolavirus risk factors

SECTION 1
01 - 04

RAPIDS PLATFORM



CUDF DATAFRAMES

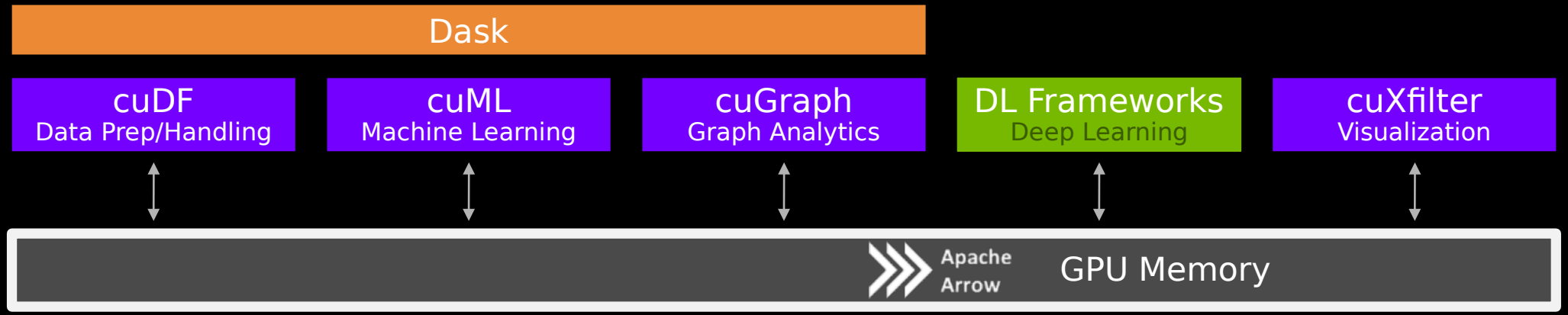
Pandas model: observations/records (rows) of features (columns)

Each feature/column has a single datatype

Simple, flexible interface to complex, performant datastructure

Special emphasis on columnar structure

RAPIDS PLATFORM

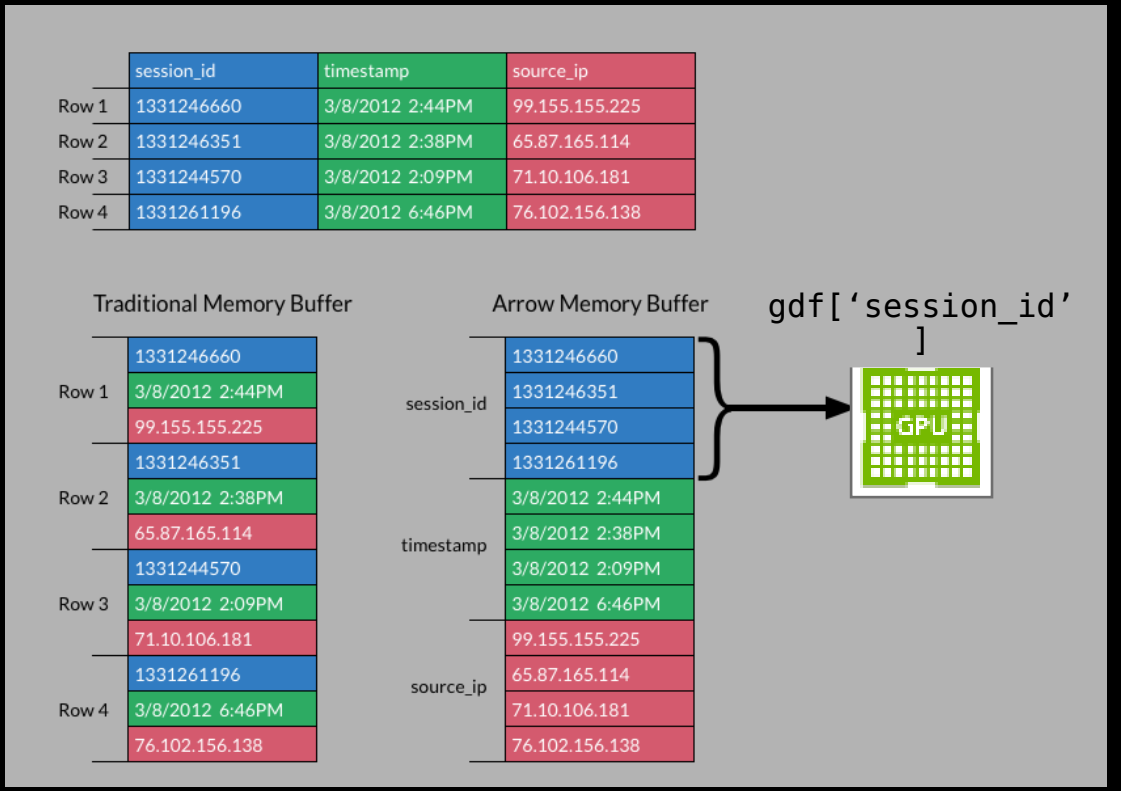


APACHE ARROW

Columnar layout leverages GPU strengths

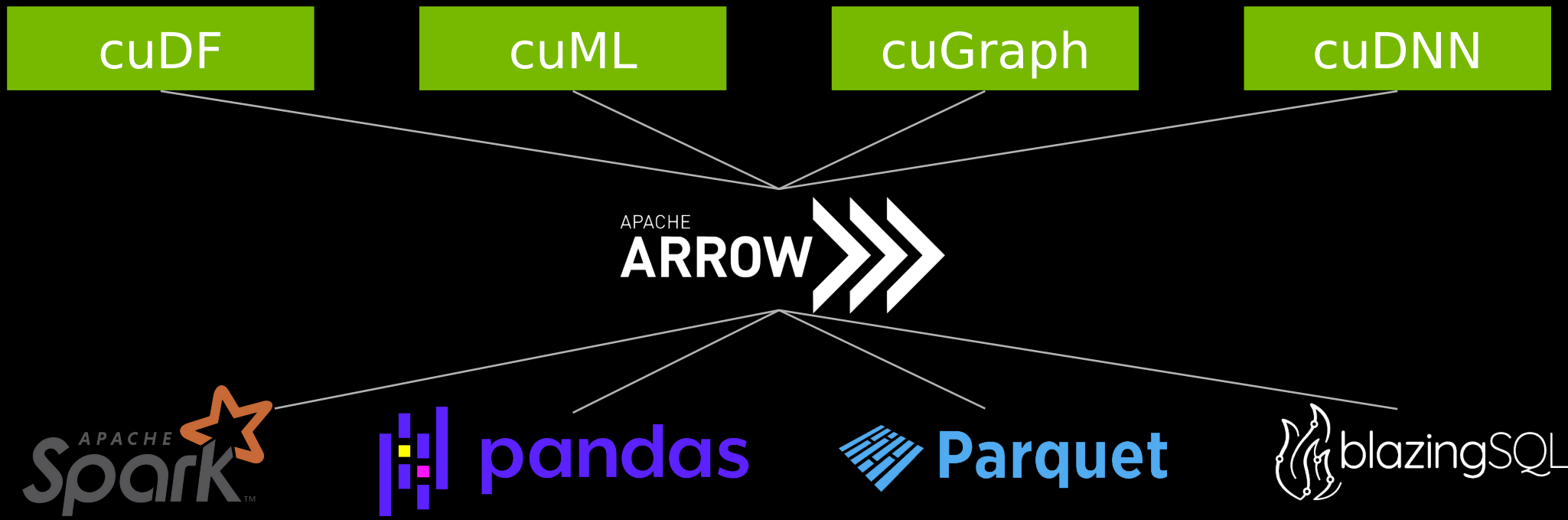
Emphasis on zero-copy and shallow-copy operations minimizes a key bottleneck

Consistency with CPU version simplifies development and conversion



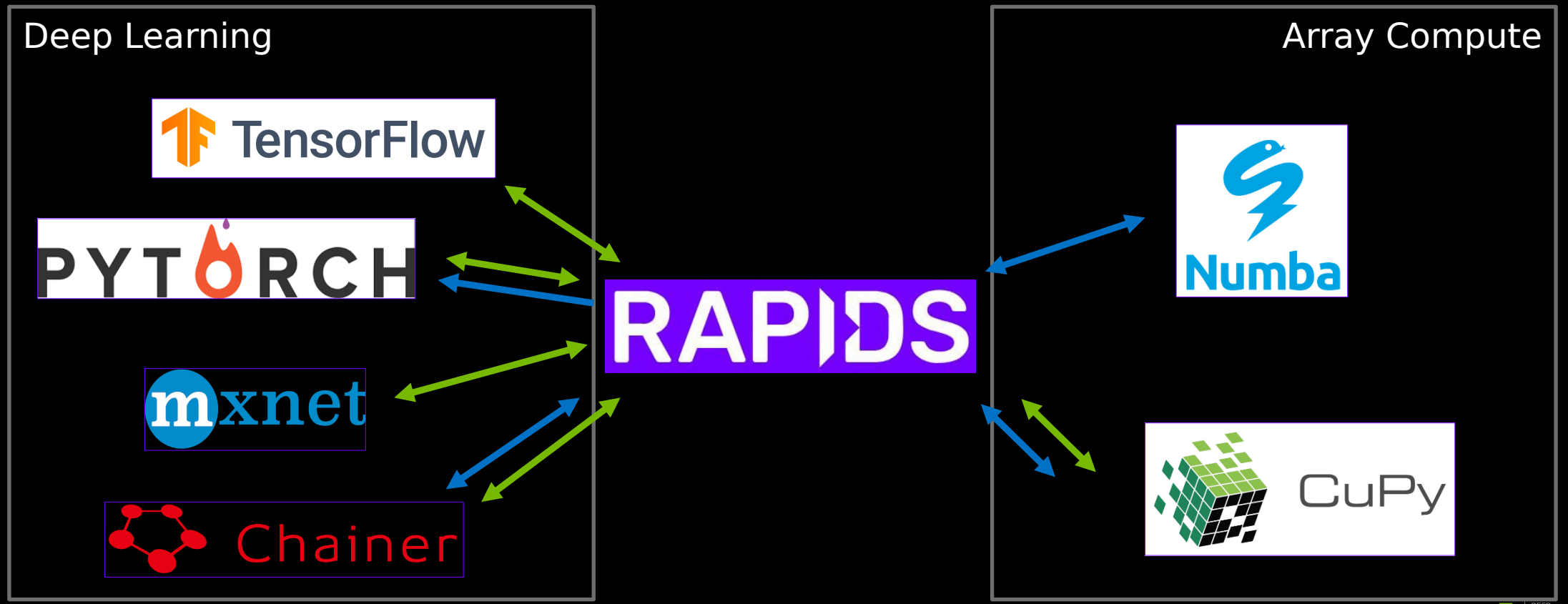
APACHE ARROW

One format for interoperability and efficiency



INTEROPERABILITY

DLPack and __cuda_array_interface__



TRY NOTEBOOKS 01 – 04 NOW

docs.rapids.ai/api

SECTION 1
05

INTEROPERATING WITH CUPY

CuPy:cuDF :: numpy:pandas

Not as fast as an optimized CUDA kernel, but very efficient for coding

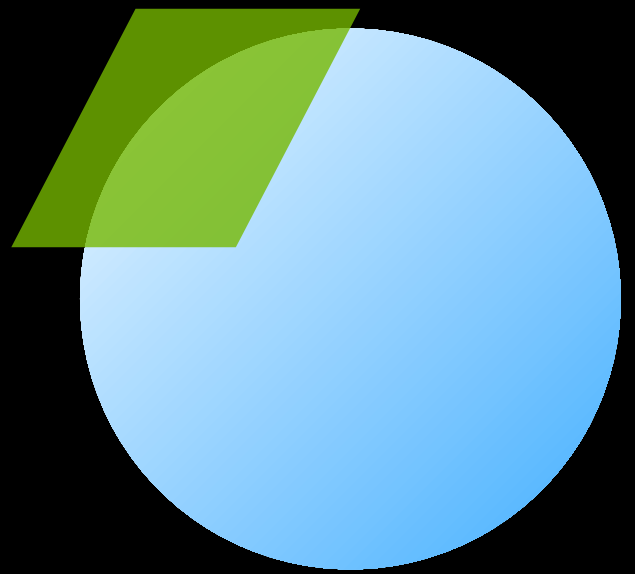
Important to keep track of data type requirements (e.g. contiguity)

COORDINATE SYSTEMS

We will be using data that was provided in both ellipsoidal and grid coordinate formats

Grid coordinates make distance calculations more convenient within a specific area

Fusing geospatial datasets like this requires complex coordinate conversions—a perfect job for GPU acceleration!

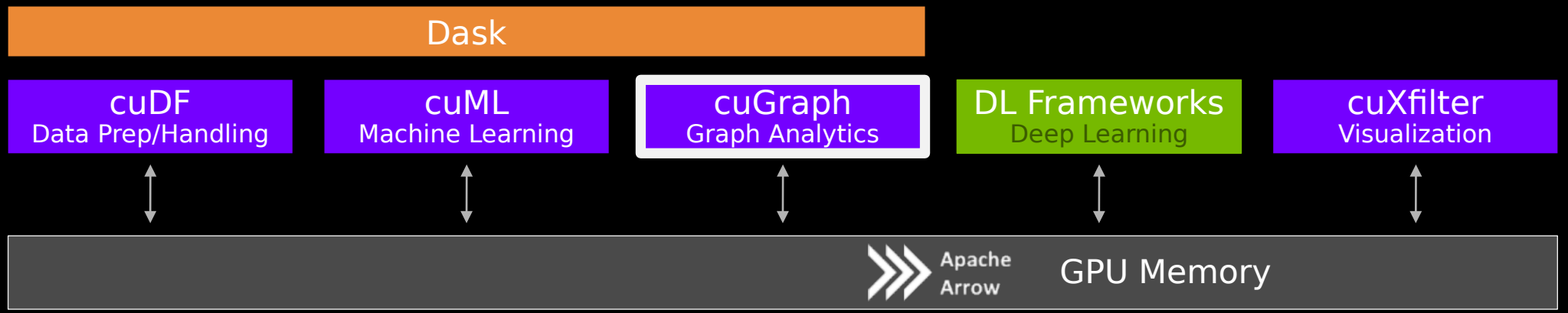


TRY NOTEBOOK 05 NOW

docs.rapids.ai/api

SECTION 1
06

RAPIDS PLATFORM



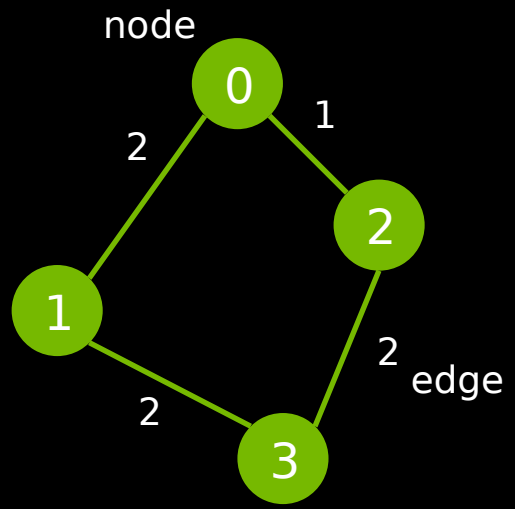
CUGRAPH

Follows NetworkX convention for graph object

Key differences to take advantage of GPU power

Exercises

- Now: steps to build a graph with `from_cudf_edgelist`
- Later: traversing the graph with single-source shortest path



Not shown today: analyzing a graph for centralities, communities, link prediction…

BUILDING A GRAPH

With `from_cudf_edgelist`

Undirected (Graph) vs directed (DiGraph)

Single vs Multi graphs

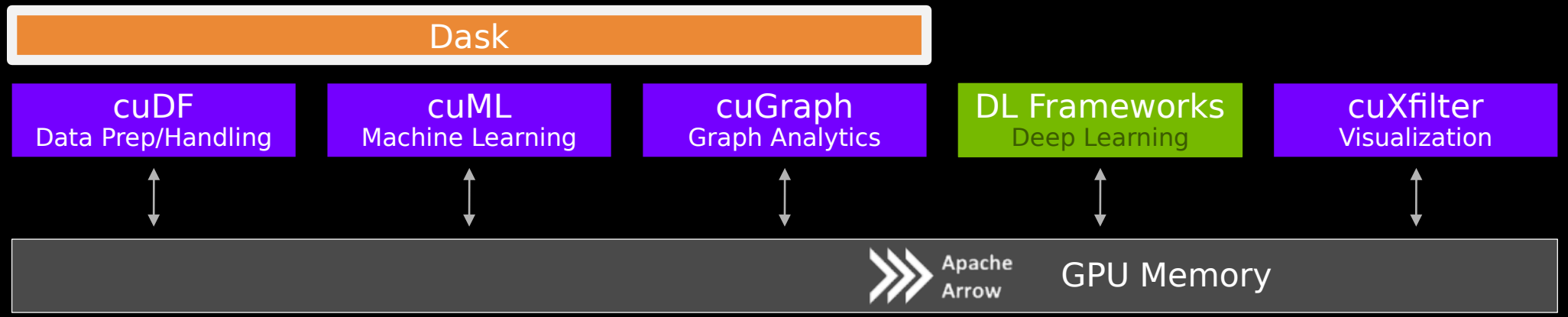
One source column, one destination column, one edge weight column

TRY NOTEBOOK 06 NOW

docs.rapids.ai/api

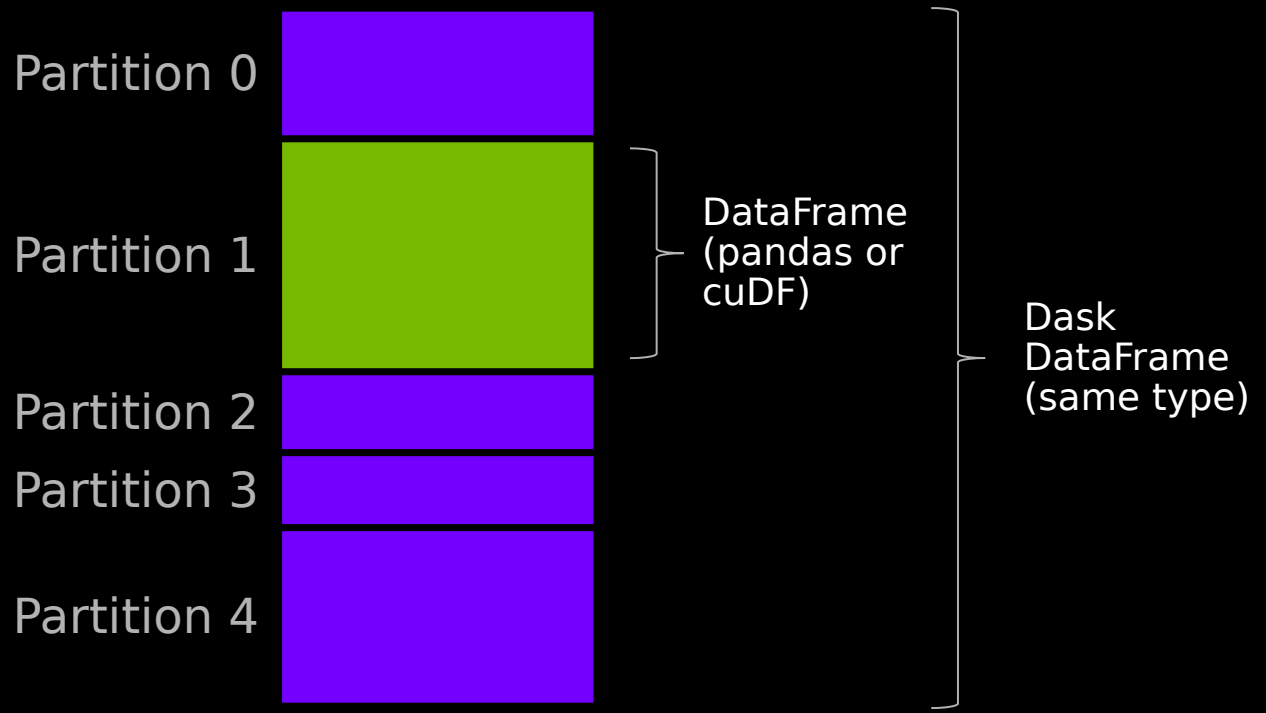
SECTION 1
07 - 08

RAPIDS PLATFORM



DISTRIBUTED DATAFRAMES

Scaling seamlessly



WORKING WITH PARTITIONS

No intrinsic row ordering, so no `.iloc` row selection, and index is essential

Key methods operate on whole dataframe partitions

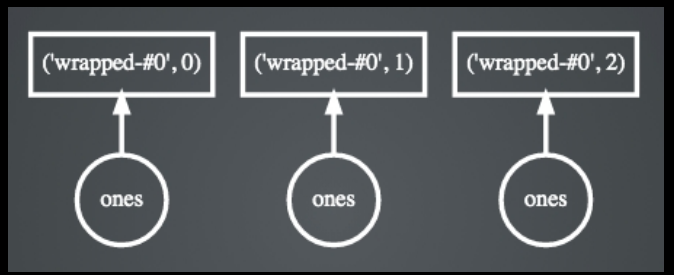
Remember distinction between multi-GPU and multi-node/multi-GPU algorithms

Rebalance across workers when necessary

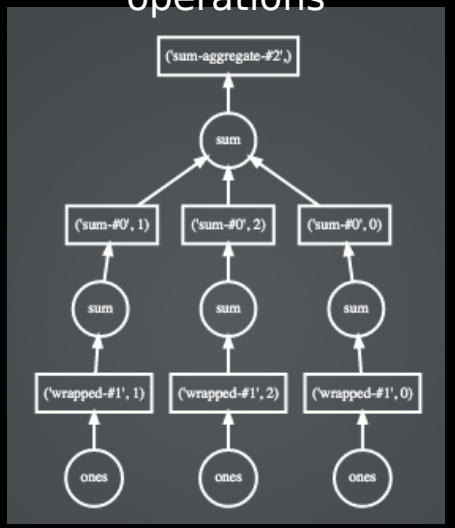
TASK SCHEDULER

Enabling efficient compute

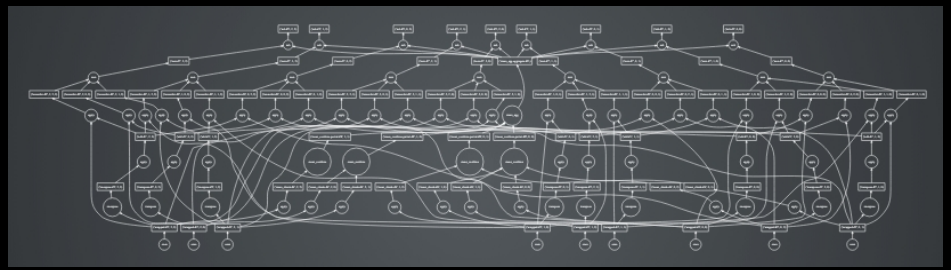
Simple operations



Compound operations



Complex DAG task chains



WORKING WITH THE SCHEDULER

Let Dask help you overcome your storage I/O barriers

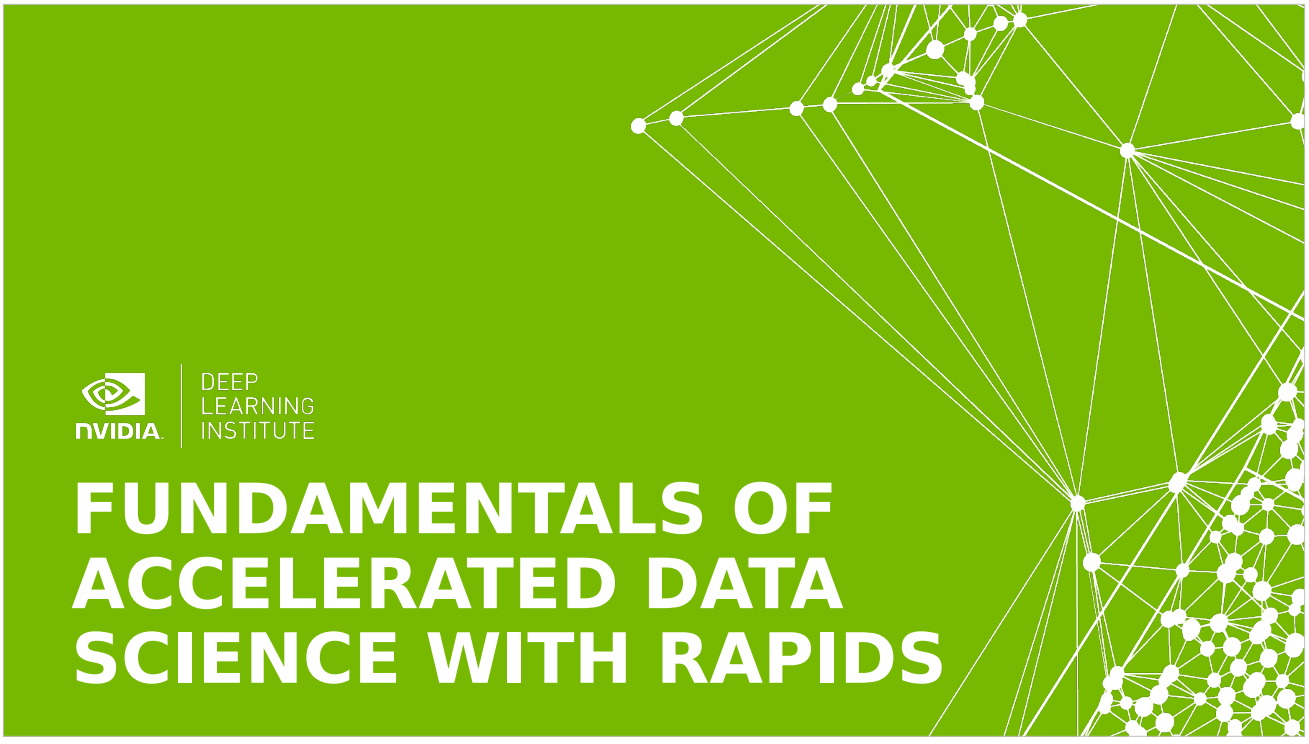
Limit `.compute` (stay in Dask) until necessary

For exploratory and experimental data science, don't be afraid to `.persist`

Remember that everything in a graph will be rerun
without `.persist`/`.compute`—
including random number generation

TRY NOTEBOOKS 07 - 08 NOW

docs.rapids.ai/api

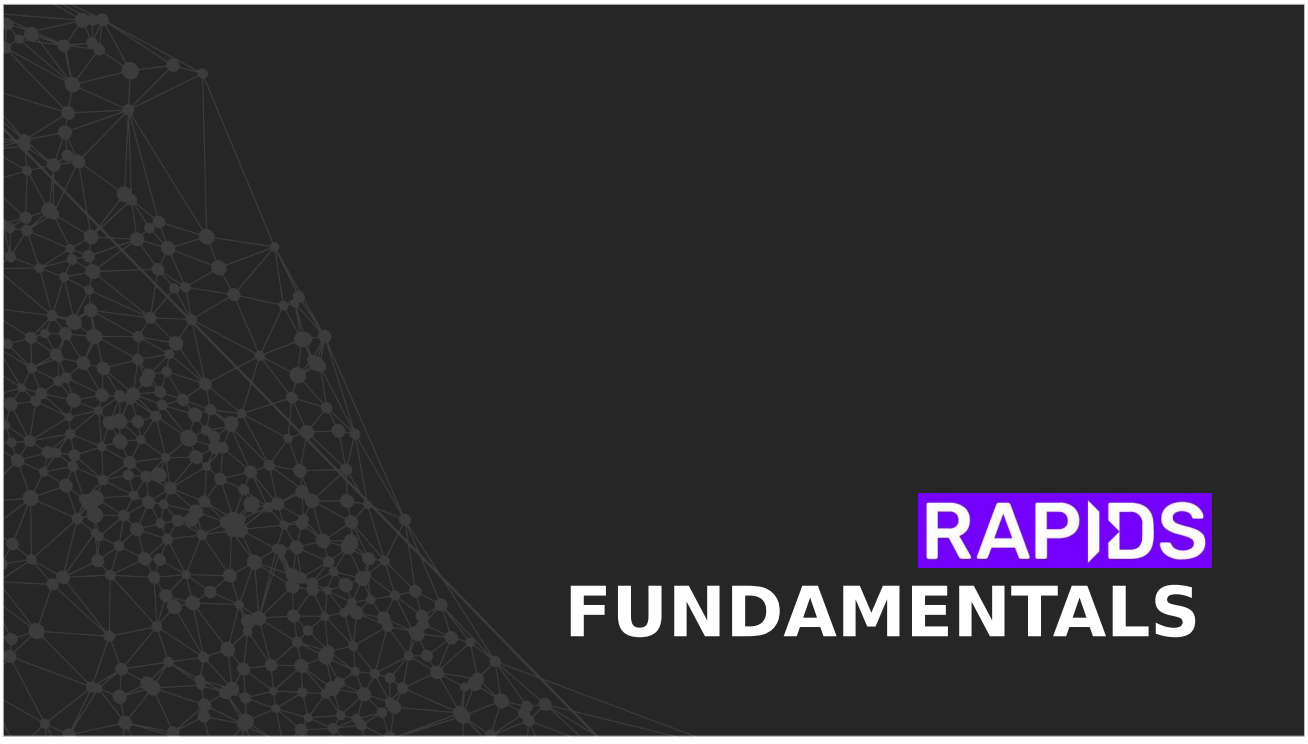

DEEP
LEARNING
INSTITUTE
FUNDAMENTALS OF
ACCELERATED DATA
SCIENCE WITH RAPIDS

COURSE GOALS

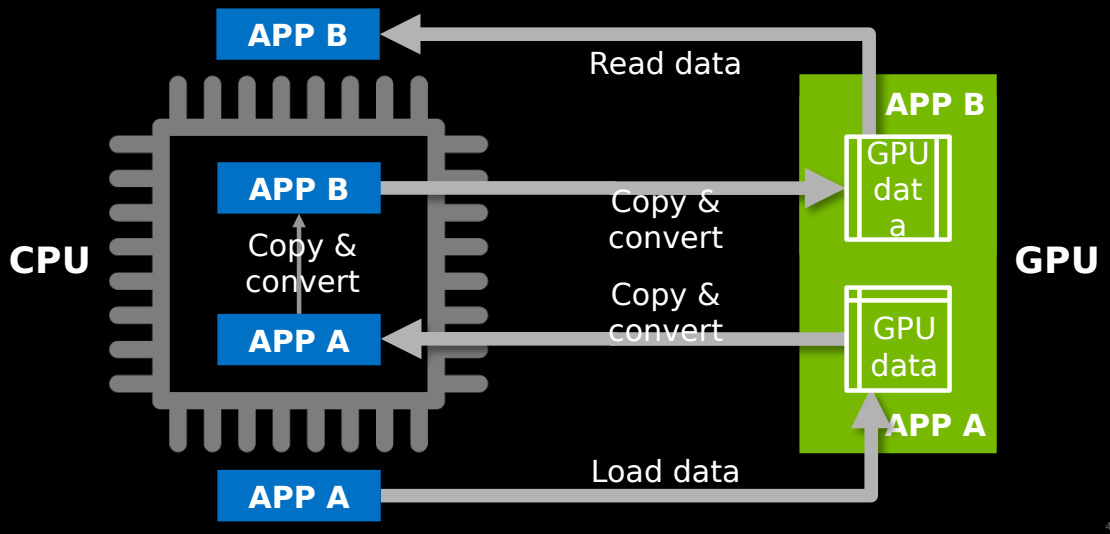
Learn the core tools to use RAPIDS for everyday data science

Understand RAPIDS' scalability from workstation and cluster to cloud and HPC

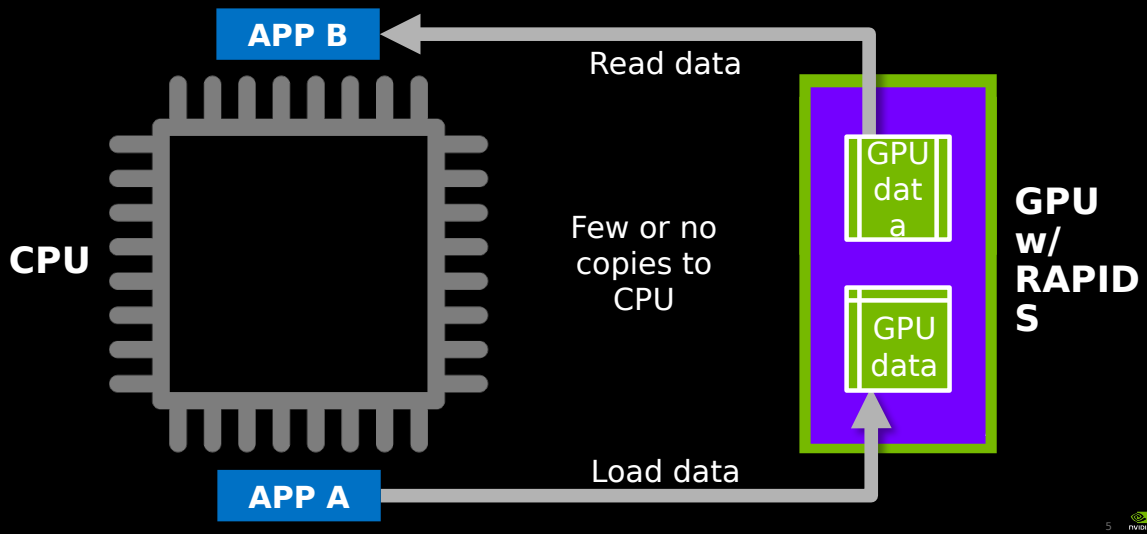
Build the foundations for you to learn RAPIDS capabilities now and in the future


RAPIDS
FUNDAMENTALS

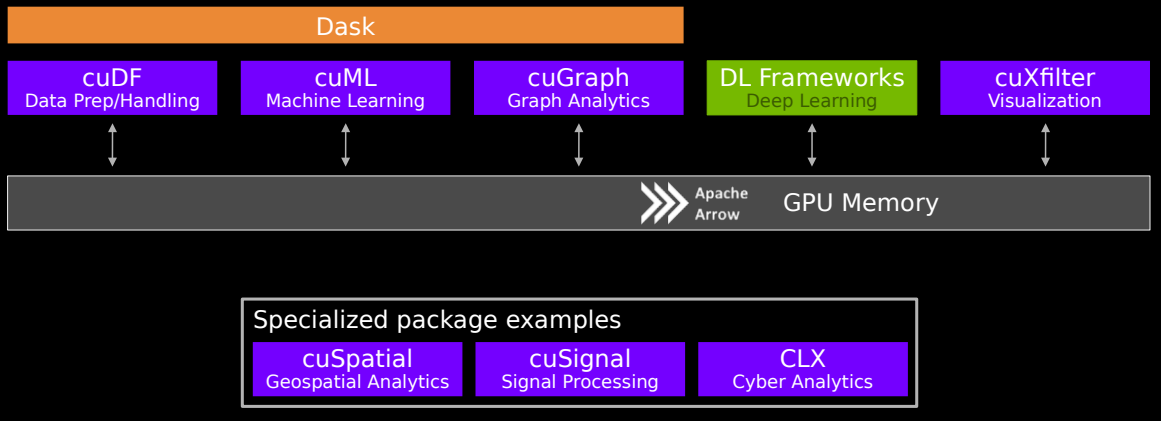
TRADITIONAL MODEL



RAPIDS MODEL



RAPIDS PLATFORM



DATA SCIENCE TOOLSETS

	CPU	GPU/ RAPIDS
Data handling	pandas	cuDF
Machine learning	scikit-learn	cuML
Graph analytics	NetworkX	cuGraph

	CPU	GPU/ RAPIDS
Viz	Bokeh/ Datashader	cuXfilter
Geospatial	GeoPandas / SciPy.spatial	cuSpatial
Signals	SciPy.signal	cuSignal
Cyber	cyberpandas	CLX

REQUIREMENTS

Appropriate OS: Ubuntu 16.04/18.04, CentOS/RHEL 7, Windows with WSL (preview)

NVIDIA Pascal™ GPU architecture or newer

CUDA 10.1.2/10.2/11.0, drivers, etc. (see rapids.ai)

Open source/flexible mindset

- Using v0.15 in this class
- New versions released regularly
- v1.0 TBD

RAPIDS

Source code on GitHub

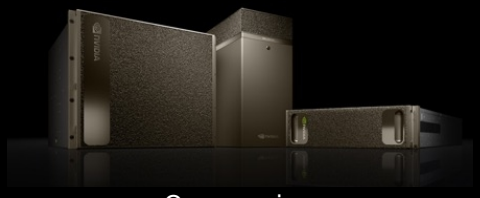
https://github.com/rapidsai

Containers on NGC & Docker Hub

https://ngc.nvidia.com

Conda packages

https://anaconda.org/rapidsai



On-premises

rapids.ai

In the cloud

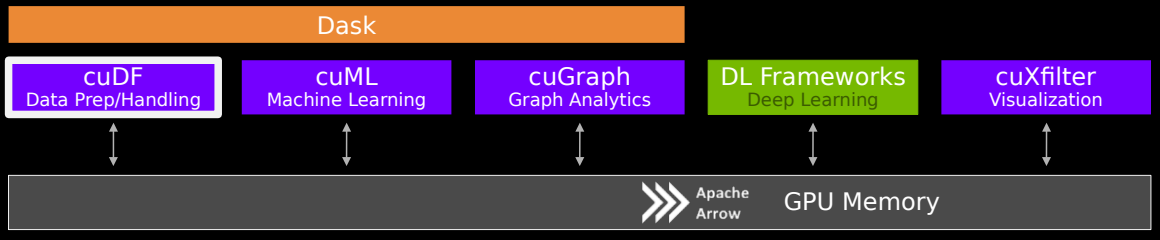
EXERCISE DATA

Fused and simulated from several sources

- Population data
  - Simulated from UK Census data on England and Wales, both from details (age, sex, given name, county) and aggregate statistics (geographic coordinates, employment)
- Road network data
  - Nodes (endpoints/junctions) and edges of the entire road network of Great Britain
- Epidemic data
  - Detailed hospital/clinic data from the UK National Health Service
  - Spread modeled on academic research on Ebolavirus risk factors


SECTION 1
01 - 04

RAPIDS PLATFORM



CUDF DATAFRAMES

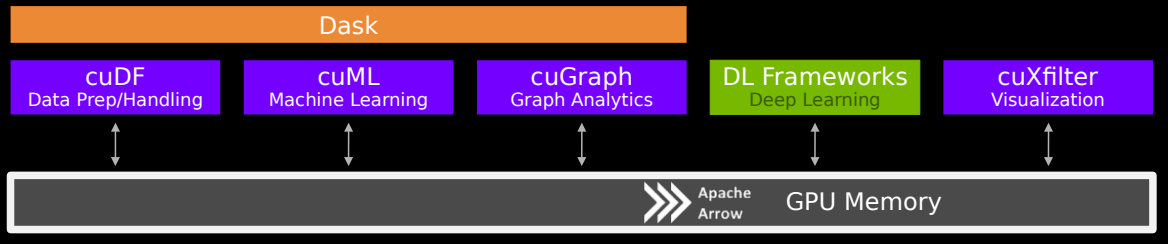
Pandas model: observations/records (rows) of features (columns)

Each feature/column has a single datatype

Simple, flexible interface to complex, performant datastructure

Special emphasis on columnar structure

RAPIDS PLATFORM

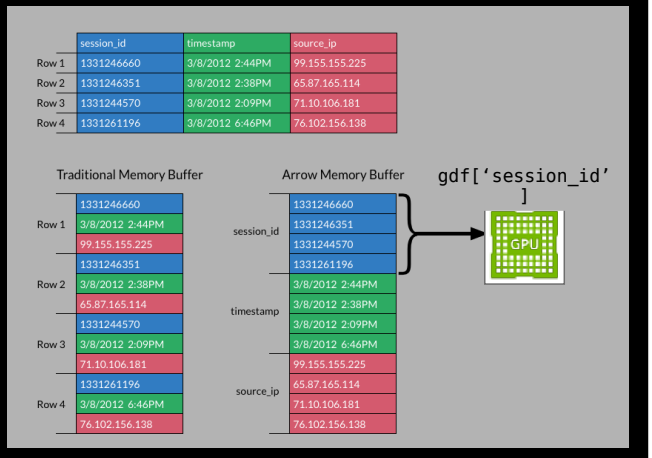


APACHE ARROW

Columnar layout leverages GPU strengths

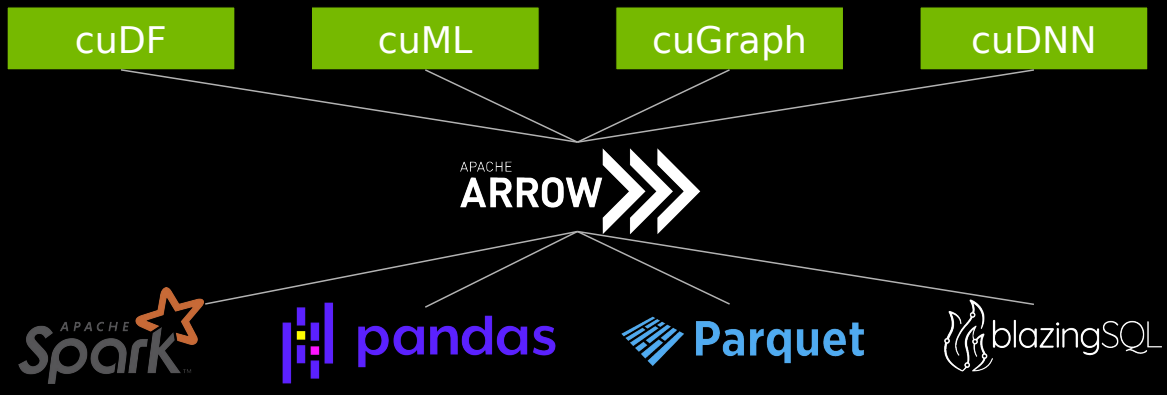
Emphasis on zero-copy and shallow-copy operations minimizes a key bottleneck

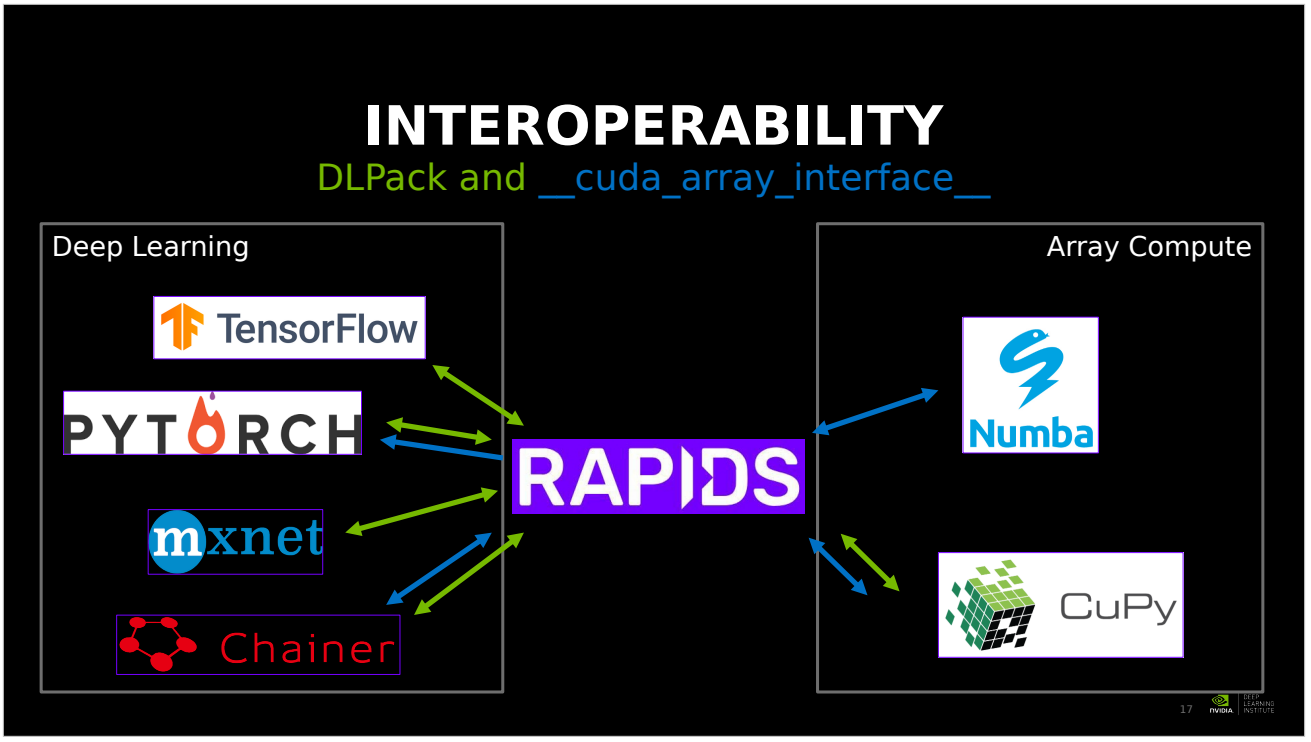
Consistency with CPU version simplifies development and conversion



APACHE ARROW

One format for interoperability and efficiency




INTEROPERABILITY
DLPack and __cuda_array_interface__
Deep Learning
Array Compute
TensorFlow
PYTORCH
mxnet
Chainer
RAPIDS
Numba
CuPy
17

TRY NOTEBOOKS 01 – 04 NOW

docs.rapids.ai/api

SECTION 1
05

INTEROPERATING WITH CUPY

CuPy:cuDF :: numpy:pandas

Not as fast as an optimized CUDA kernel, but very efficient for coding

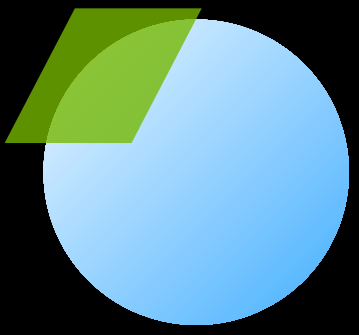
Important to keep track of data type requirements (e.g. contiguity)

COORDINATE SYSTEMS

We will be using data that was provided in both ellipsoidal and grid coordinate formats

Grid coordinates make distance calculations more convenient within a specific area

Fusing geospatial datasets like this requires complex coordinate conversions—a perfect job for GPU acceleration!

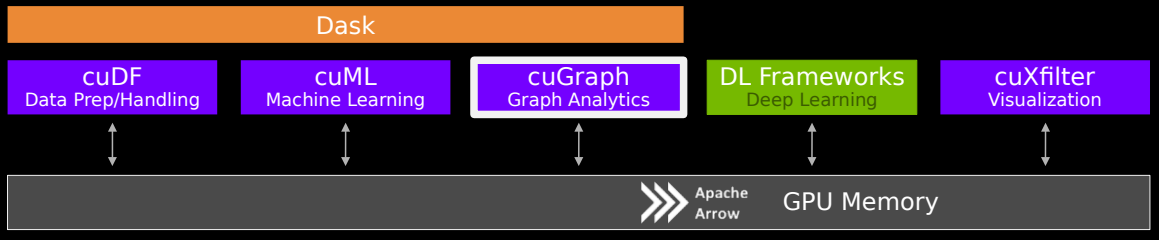


TRY NOTEBOOK 05 NOW

docs.rapids.ai/api

SECTION 1
06

RAPIDS PLATFORM



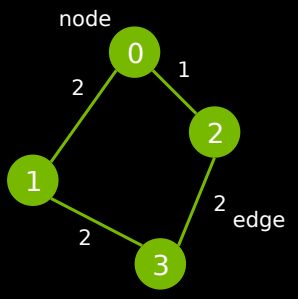
CUGRAPH

Follows NetworkX convention for graph object

Key differences to take advantage of GPU power

Exercises

- Now: steps to build a graph with `from_cudf_edgelist`
- Later: traversing the graph with single-source shortest path



Not shown today: analyzing a graph for centralities, communities, link prediction…

BUILDING A GRAPH

With `from_cudf_edgelist`

Undirected (Graph) vs directed (DiGraph)

Single vs Multi graphs

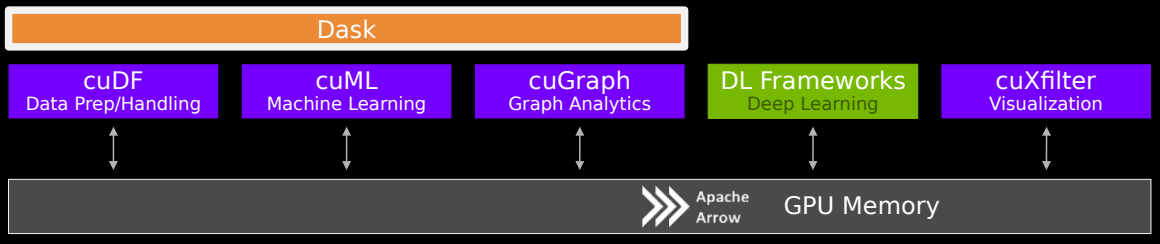
One source column, one destination column, one edge weight column

TRY NOTEBOOK 06 NOW

docs.rapids.ai/api

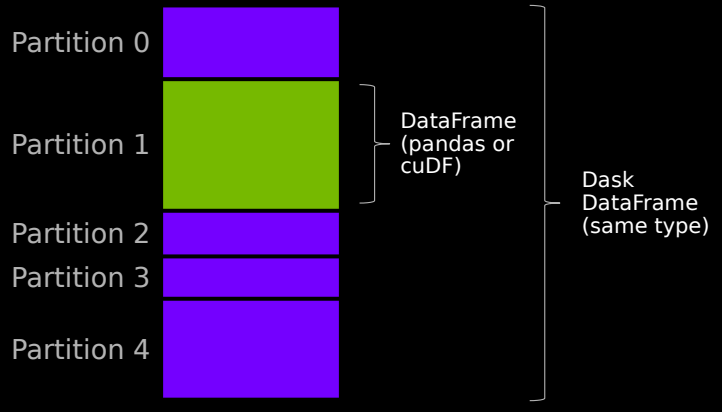
SECTION 1
07 - 08

RAPIDS PLATFORM



DISTRIBUTED DATAFRAMES

Scaling seamlessly



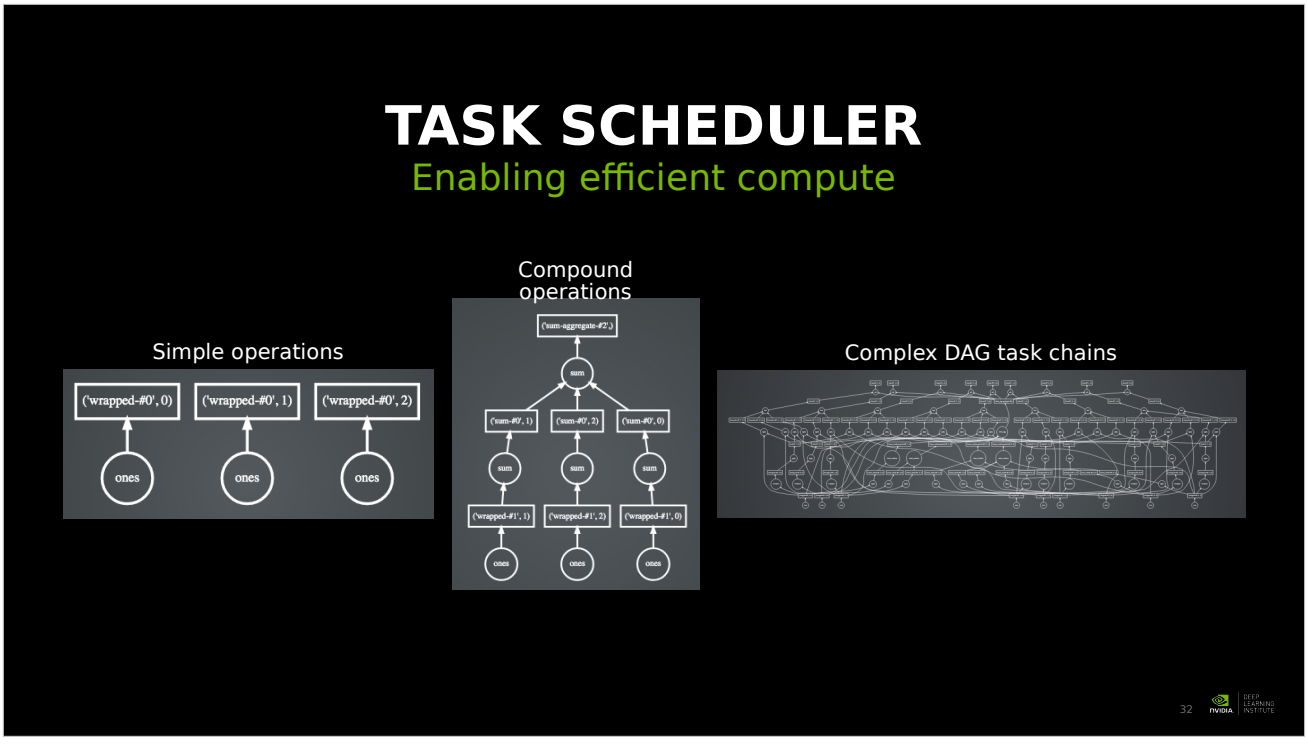
WORKING WITH PARTITIONS

No intrinsic row ordering, so no `.iloc` row selection, and index is essential

Key methods operate on whole dataframe partitions

Remember distinction between multi-GPU and multi-node/multi-GPU algorithms

Rebalance across workers when necessary



Dask parallelizes PyData natively.

Dask.distributed is a centrally managed, distributed, dynamic task scheduler. The central dask-scheduler process coordinates the actions of several dask-worker processes spread across multiple machines and the concurrent requests of several clients.

The scheduler is asynchronous and event driven, simultaneously responding to requests for computation from multiple clients and tracking the progress of multiple workers. The event-driven and asynchronous nature makes it flexible to concurrently handle a variety of workloads coming from multiple users at the same time while also handling a fluid worker population with failures and additions. Workers communicate amongst each other for bulk data transfer over TCP.

Internally the scheduler tracks all work as a constantly changing directed acyclic graph of tasks. A task is a Python function operating on Python objects, which can be the results of other tasks. This graph of tasks grows as users submit more computations, fills out as workers complete tasks, and shrinks as users leave or become disinterested in previous results.

WORKING WITH THE SCHEDULER

Let Dask help you overcome your storage I/O barriers

Limit `.compute` (stay in Dask) until necessary

For exploratory and experimental data science, don't be afraid to `.persist`

Remember that everything in a graph will be rerun
without `.persist`/`.compute`—
including random number generation

TRY NOTEBOOKS 07 - 08 NOW

docs.rapids.ai/api