OAK RIDGE National Laboratory

OLCF User Conference Call
NVIDIA RAPIDS update at OLCF

Benjamín Hernández
Analytics & AI Methods at Scale (AAIMS) Group
OLCF
hernandezarb@ornl.gov

August 25th, 2021

U.S. DEPARTMENT OF ENERGY

Getting Started

- RAPIDS v21.08
  - RMM (RAPIDS Memory Manager)
  - cuDF
  - cuML
  - cuGraph
  - dask-cuda
- Preferred Networks' CuPy (since v0.18) - NumPy like library, compatible with RAPIDS ecosystem
- BlazingSQL (since v0.18) - high-performance distributed SQL engine in Python. No database needed.
- **New!** cuCIM - GPU based image processing and I/O.

Close to Python APIs

- cuDF -> pandas
- cuML -> scikit-learn
- cuGraph -> networkX
- cuCIM -> scikit-image, openslide
- CuPy -> numpy

Overview of Changes

- Arrow 4.0 support ! (more later)
- RMM (Rapids memory manager)
  - Fast memory allocation and deallocation
  - Regular and managed memory support.
  - Memory Pools.
  - Not only for RAPIDS!

Numpy to GPU memory - GPU memory to Numpy

```
import rmm
import numpy as np
a = np.array([1, 2, 3], dtype='float64')
buf = rmm.to_device(a.tobytes())
…
np.frombuffer(buf.tobytes())
array([1., 2., 3.])
```

RMM with CuPy

```
import rmm
import cupy
cupy.cuda.set_allocator(rmm.rmm_cupy_allocator)
cupy.ones(10)
```

https://docs.rapids.ai/overview/RAPIDS%2021.08%20Release%20Deck.pdf

Overview of Changes

- cuDF
  - Decimal support for CSV reader
  - List read and write, and experimental read support for structs in ORC
  - Supports multiple inputs in JSON and ORC reader
  - …

- Features to watch:
  - Parsing occurs on the GPU
  - I/O - local (csv, parquet, ORC, JSON) and remote data store (s3://, hdfs://, gcs://, http://...)
  - Word tokenizer
  - Windowing operations
  - https://docs.rapids.ai/api/cudf/stable/

https://docs.rapids.ai/overview/RAPIDS%2021.08%20Release%20Deck.pdf

Overview of Changes

cuML - Multi-GPU, multi-node support

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

https://docs.rapids.ai/overview/RAPIDS%2021.08%20Release%20Deck.pdf

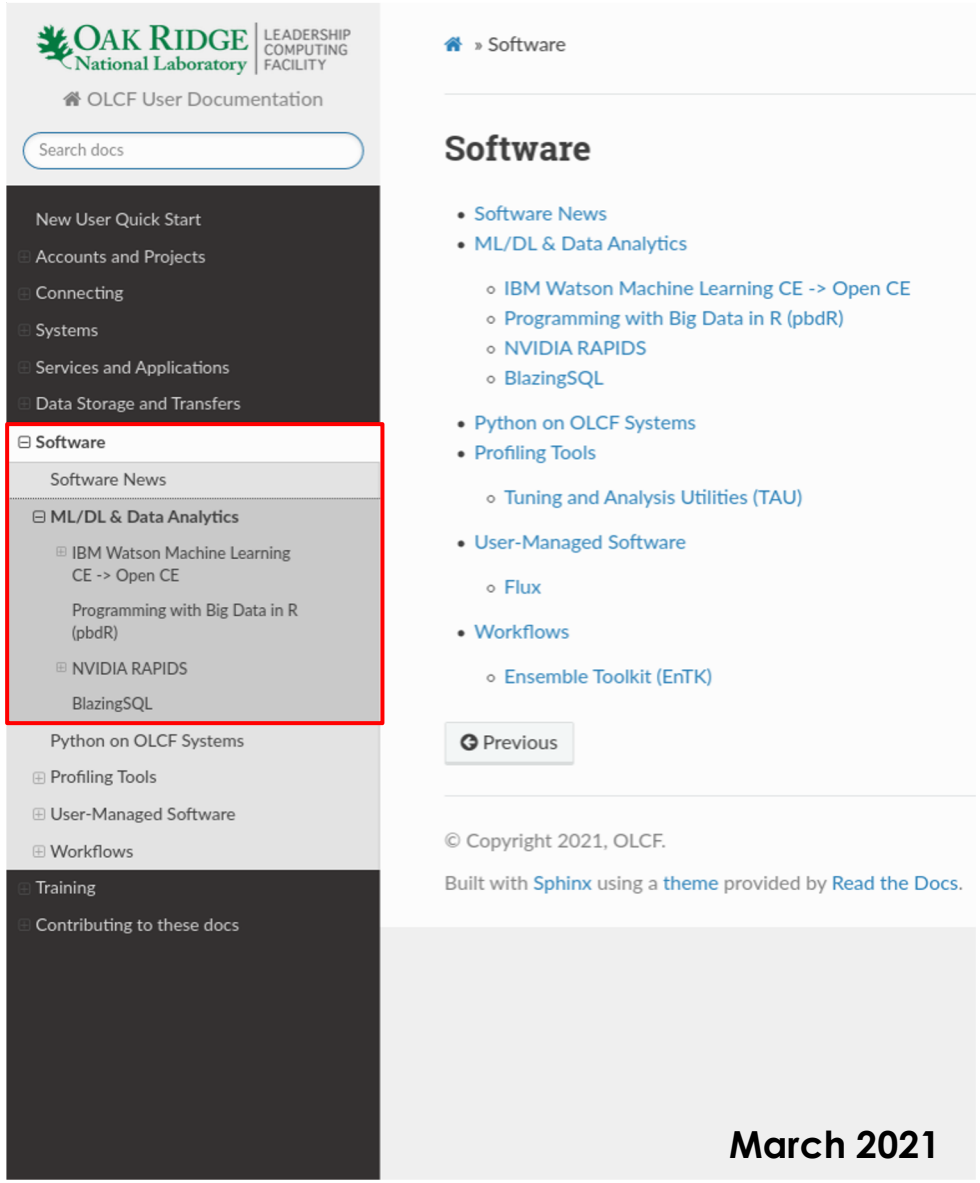
Overview of Changes

cuGraph - Multi-GPU, multi-node support

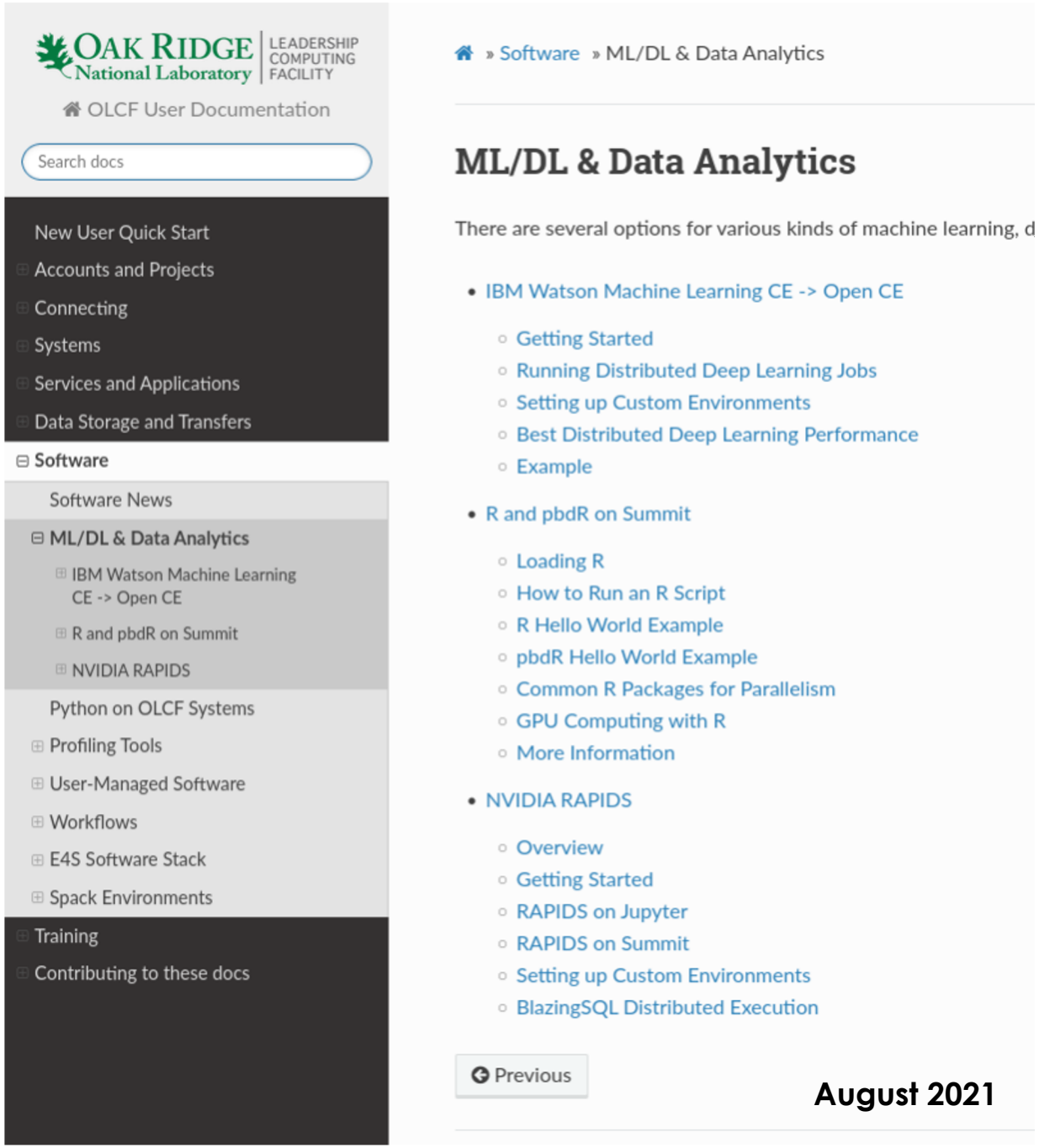
Algorithm	Target Release	Percent Done	Status
PageRank		100%	Complete
Personal PagePank		100%	Complete
BFS		100%	Complete
SSSP		100%	Complete
Louvain		100%	Complete
Katz		100%	Complete
WCC		100%	Complete

https://docs.rapids.ai/overview/RAPIDS%2021.08%20Release%20Deck.pdf

Overview of Changes



March 2021



August 2021

Getting Started

RAPIDS on Jupyter@OLCF

https://docs.olcf.ornl.gov/services_and_applications/jupyter/overview.html#example-creating-a-conda-environment-for-rapids

RAPIDS/BlazingSQL

```
module load ums
module load ums-gen119
module load nvidia-rapids/21.08
```

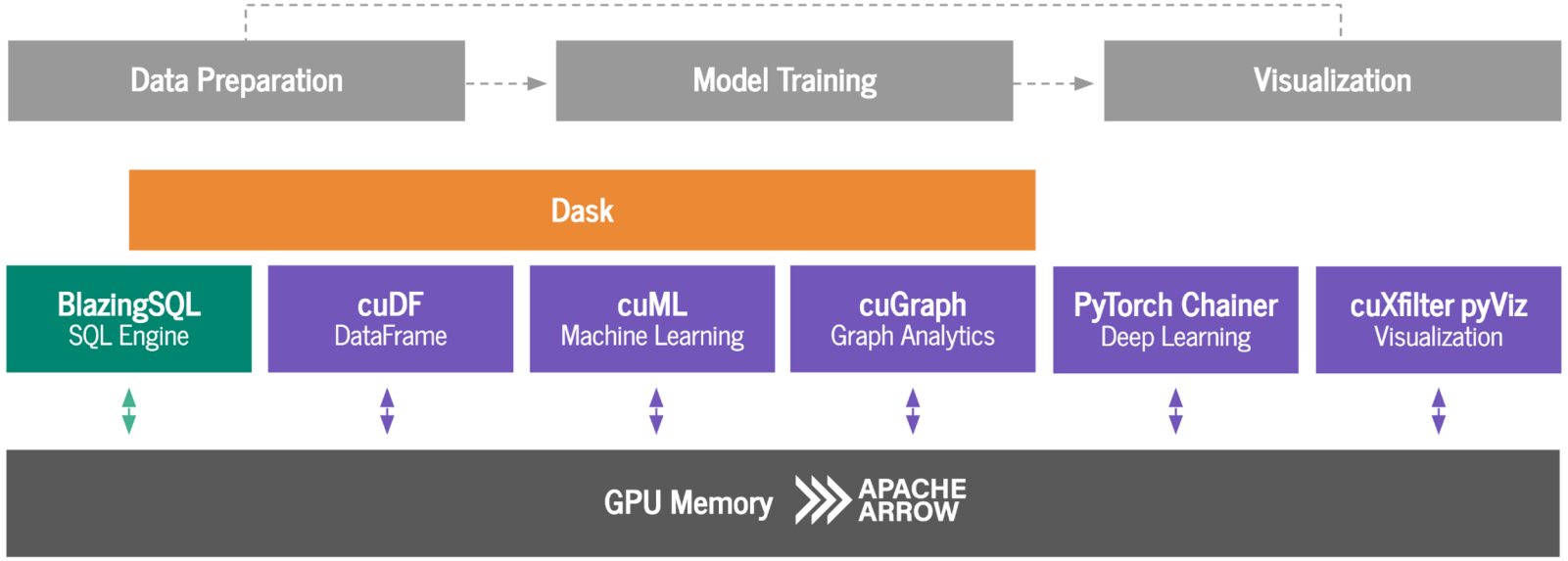
The module include RMM, cuDF, cuML and cuGraph C/C++ CUDA libraries !

cuCIM

```
module load ums
module load ums-gen119
module load nvidia-rapids/cucim_21.08
```

Refer to https://docs.olcf.ornl.gov/software/analytics/nvidia-rapids.html to learn about how to submit jobs

BlazingSQL



- Open source community effort.
- Single GPU, multi GPU and distributed SQL engine
  - Operates on Pandas, cudf and dask_cudf dataframes
  - I/O module supports various file formats (text delimited, Apache Orc, Apache Parquet, JSON)
  - Various filesystems (HDFS, AWS S3, GCS) <- not on Summit, let us know if needed.

BlazingSQL - an example

INPUT: pandas, cuDF or directly from file

```
import blazingsql
import cudf

bc = blazingsql.BlazingContext()
gdf = cudf.read_csv('taxi.csv')
bc.create_table('taxis', gdf)
```

```
import blazingsql
import pandas

bc = blazingsql.BlazingContext()
df = pandas.read_csv('taxi.csv')
bc.create_table('taxis', df)
```

```
import blazingsql
import cudf

bc = blazingsql.BlazingContext()
bc.create_table('taxis', taxi.parquet)
```

QUERY: on GPU no matter where the data is coming

```
gdf = bc.sql('SELECT count(*) FROM taxi GROUP BY year(key)')
```

OUTPUT: result of the query is a cuDF dataframe, i.e. result is kept on GPU for further analysis with RAPIDS

```
print(gdf)
```

or it can be transferred to CPU

```
df = gdf.to_pandas()
```

For a distributed example refer to: https://docs.olcf.ornl.gov/software/analytics/nvidia-rapids.html#blazingsql-distributed-execution

cuCIM

- Volumetric or general n-dimensional data typical on scientific fields such as bioimaging (microscopy), medical imaging (CT/PET/MRI), materials science, and remote sensing.
- cuCIM provides CUDA-accelerated image processing operations.
- Closely mirror the scikit-image API for image manipulation and OpenSlide for image loading.
- Visit https://docs.rapids.ai/api/cucim/stable/ for current functionality

cuCIM - operations

Feature Extraction

- canny
- corner_harris
- corner_shi_thomasi
- daisy
- match_templated
- shape_index
- structure_tensor
- ...

Restoration

- calibrate_denoiser
- denoise_tv_chambolle
- richardson_lucy
- wiener
- unsupervised_wiener

Registration

- optical_flow_ilk
- optical_flow_tvl1
- phase_cross_correlation

Morphology

- binary_erosion
- binary_dilation
- erosion (greyscale)
- opening (greyscale)
- remove_small_objects
- ...

Measure

- label
- centroid
- moments_central
- moments_hu
- shannon_entropy
- ...

Metrics

- peak_signal_noise_ratio
- structural_similarity
- mean_square_error
- normalized_root_mse

Transforms

- resize
- rotate
- warp
- integral_image
- pyramid_gaussian
- ...

Exposure

- histogram
- equalize_hist
- equalize_adaptive
- adjust_gamma
- match_histogram
- ...

Segmentation

- random_walker
- morphological_chan_vese
- join_segmentations
- ...

Color Conversions

- rgb2gray
- rgb2hsv
- rgb2yuv
- combine_stains
- separate_stains
- ...

Filters

- gabor
- gaussian
- median
- sobel
- frangi
- hessian
- unsharp_mask
- threshold_local
- threshold_otsu
- threshold_niblack
- threshold_sauvola
- ...

Quansight

Source GTC 2021 cuCIM: A GPU Image I/O and Processing Toolkit [S32194]

cuCIM - an example

Gamma Correction

```
from skimage import data
from cucim.skimage import exposure, img_as_float

image = img_as_float(cp.array(data.moon()))
gamma_corrected = exposure.adjust_gamma(image, 2)
```

Color conversion

```
import cupy as cp
from cucim.skimage import data

img = cp.array(data.astronaut())
img_hsv = convert_colorspace(img, 'RGB', 'HSV')
```

scikit-image

```
import numpy as np

from skimage import data
from skimage import exposure

# Load an example image
img = data.moon()




# Contrast stretching
p2, p98 = np.percentile(img, (2, 98))
img_rescale = exposure.rescale_intensity(img, in_range=(p2, p98))

# Equalization
img_eq = exposure.equalize_hist(img)

# Adaptive Equalization
img_adapteq = exposure.equalize_adapthist(img, clip_limit=0.03)
```

cuCIM

```
import cupy as cp

from skimage import data
from cucim.skimage import exposure

# Load an example image
img = data.moon()

# transfer to the GPU
img = cp.asarray(img)

# Contrast stretching
p2, p98 = cp.asnumpy(cp.percentile(img, (2, 98)))
img_rescale = exposure.rescale_intensity(img, in_range=(p2, p98))

# Equalization
img_eq = exposure.equalize_hist(img)

# Adaptive Equalization
img_adapteq = exposure.equalize_adapthist(img, clip_limit=0.03)
```

A note about interoperability

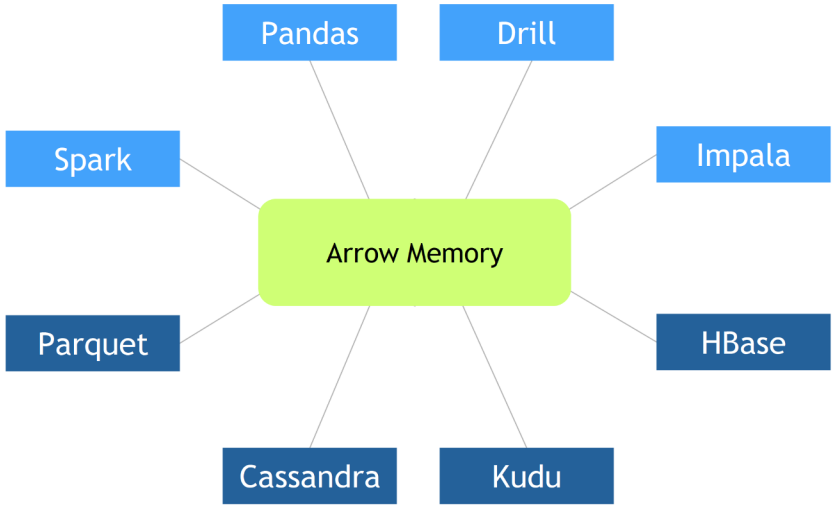
- RAPIDS ecosystems is not isolated
  - It can interact with CPU based libraries
  - It can interact with GPU based libraries
- Iterop. enabled by
  - **Arrow**: columnar data layout good for SIMD like processing -> Arrow ecosystem
  - **DLpack**: sharing tensors -> ML/DL frameworks
  - **`__cuda_array_interface__`** -> interop. with different GPU array-like objects in various projects

```
# Convert a CuPy ndarray to a PyTorch Tensor
src = cp.array([[1, 2], [3, 4]])
dst = torch.utils.dlpack.from_dlpack(src.toDlpack())

print(type(dst), "\n", dst)
```

```
# Convert a cuDF DataFrame to a CuPy ndarray
src = cudf.DataFrame({'x': [1, 2], 'y': [3, 4]})
dst = cp.fromDlpack(src.to_dlpack())

print(type(dst), "\n", dst)
```



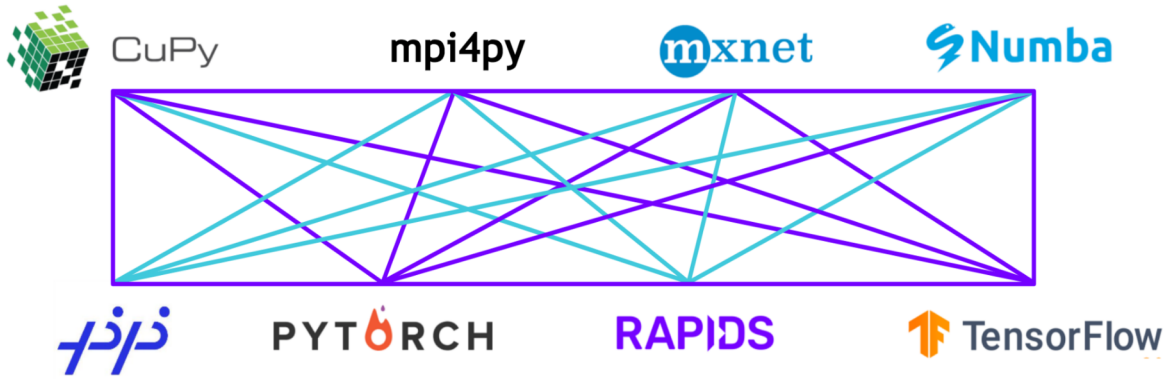


Image Source: GTC 2021 Machine Learning Frameworks Interoperability[E32273]

A note about interoperability

- No functionality available on a GPU library ?
  - Use Numba to implement such functionality
- -OR- copy ~~and convert~~ to CPU
  - Optimized on POWER9 via NVLINK !

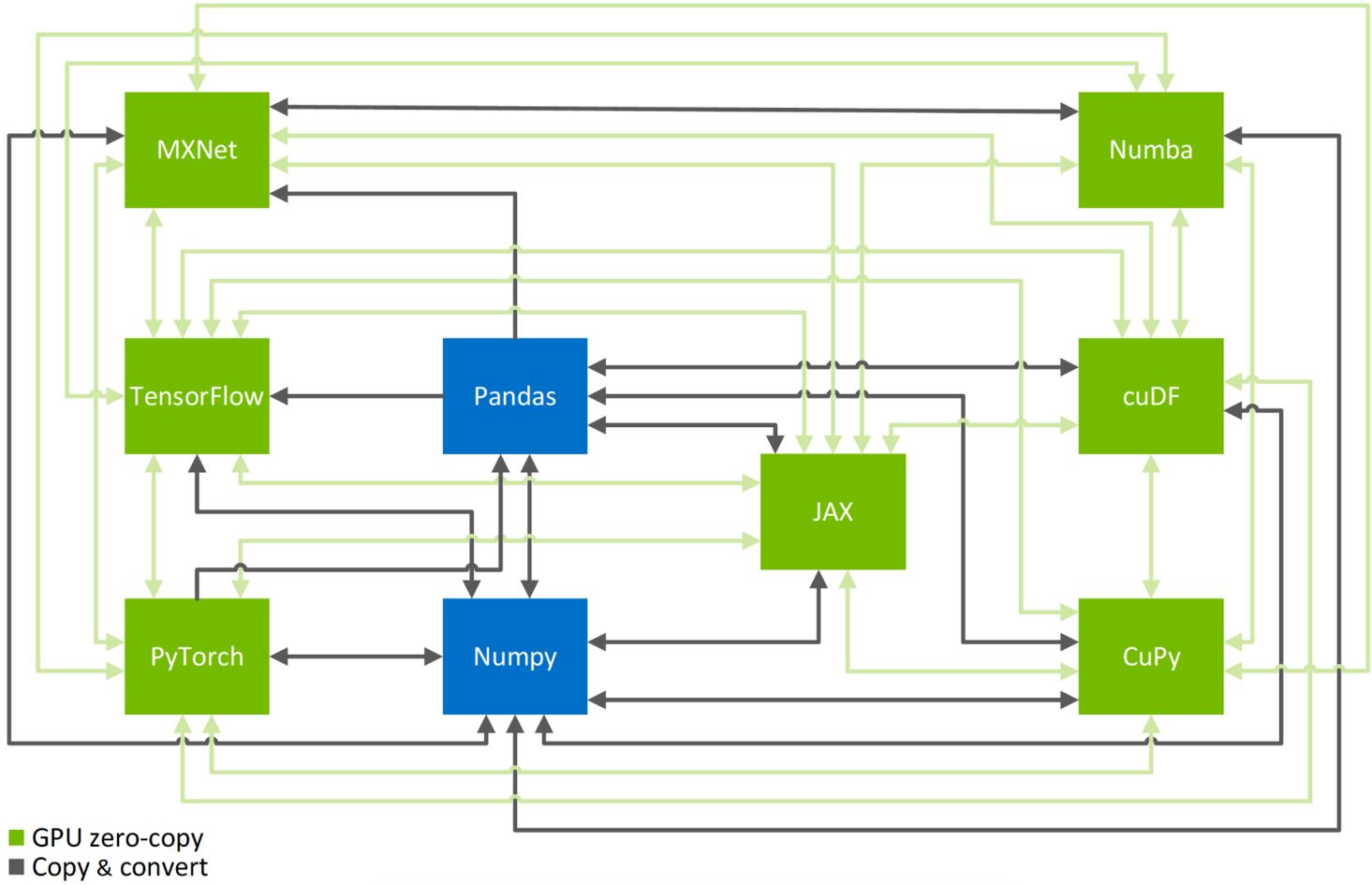


Image Source: GTC 2021 Machine Learning Frameworks Interoperability[E32273]

Further reading

- 10 Minutes to cuDF and Dask-cuDF.
  - https://docs.rapids.ai/api/cudf/stable/10min.html#
- cuML's Multi-Node, Multi-GPU Algorithms.
  - https://docs.rapids.ai/api/cuml/stable/api.html#multi-node-multi-gpu-algorithms
- Multi-GPU with cuGraph.
  - https://docs.rapids.ai/api/cugraph/stable/dask-cugraph.html
- DataAPI
  - https://data-apis.org
- Machine Learning Frameworks Interoperability Series:
  - https://developer.nvidia.com/blog/machine-learning-frameworks-interoperability-part-1-memory-layouts-and-memory-pools/

Questions ?

help@olcf.ornl.gov

- We are interested in knowing more about your use cases.
- Feedback is important
  - New features
  - New RAPIDS Libraries
  - https://forms.gle/rqKRvUsWbXXr5qes7
- Stay tuned for training announcements.

Early Adopters

- Several teams from the COVID-19 HPC Consortium
  - Data wrangling (DASK+cuDF)
  - Classification e.g. K-means, Random Forest (DASK+cuML)
  - GPU accelerated Queries (DASK+RAPIDS with BlazingSQL)

- A finalist of the ACM Gordon Bell Special Prize for High Performance Computing-Based COVID-19 Research 2020
  - "Screening Drug Compounds for COVID-19 via a Virtual Laboratory with AutoDock-GPU on Summit"

https://www.olcf.ornl.gov/2020/11/18/multi-institutional-team-earns-gordon-bell-special-prize-finalist-nomination-for-rapid-covid-19-molecular-docking-simulations/

https://www.olcf.ornl.gov/2020/10/28/in-a-groundbreaking-move-summit-joins-forces-with-blazingsql-to-speed-up-data-query-processing-on-supercomputers/