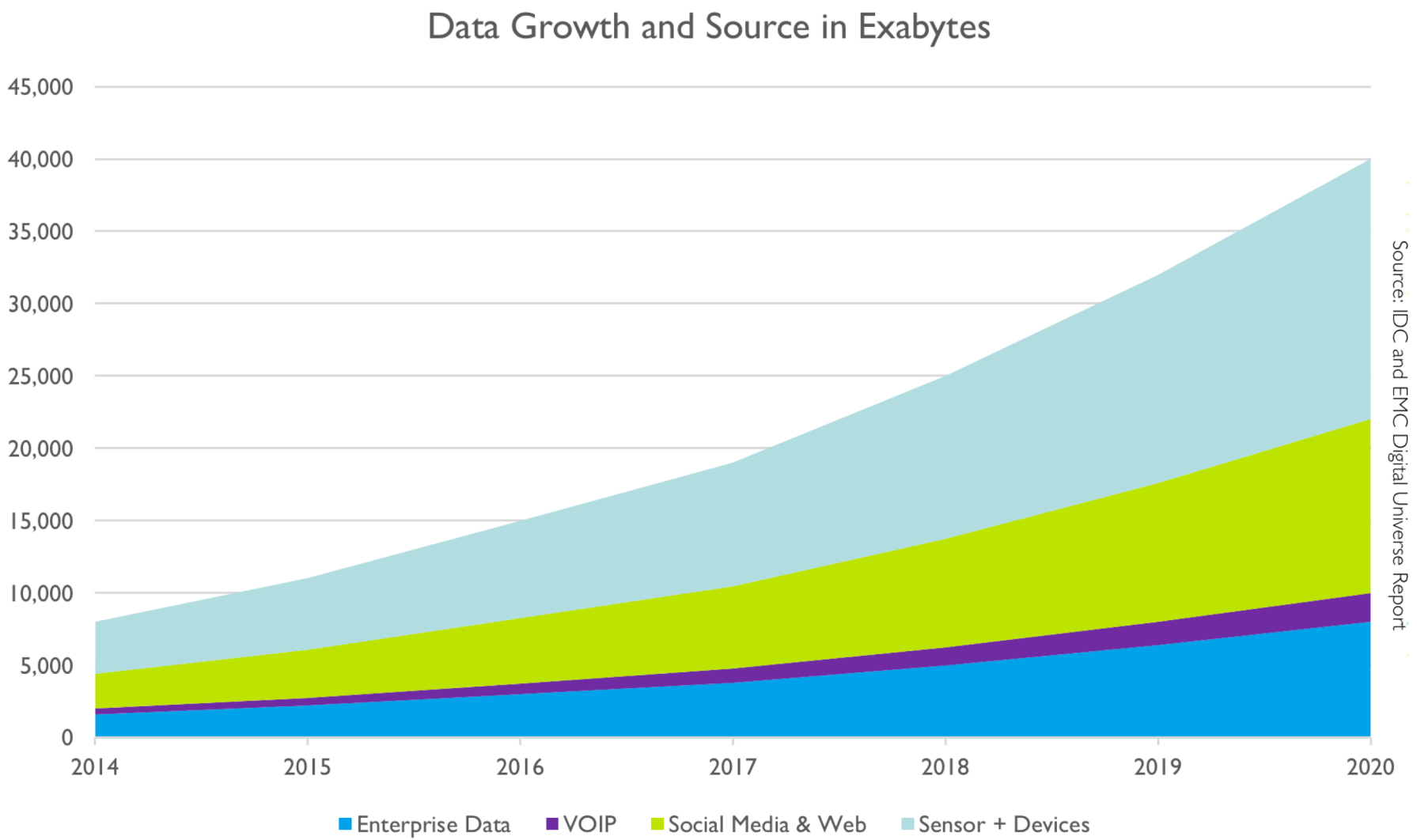
RAPIDS: THE PLATFORM INSIDE AND OUT

Josh Patterson 10-23-2018

REALITIES OF DATA



NVIDIA

DATA PROCESSING EVOLUTION

Faster Data Access Less Data Movement

Hadoop Processing, Reading from disk

DATA PROCESSING EVOLUTION

Faster Data Access Less Data Movement

WE NEED MORE COMPUTE!

Basic workloads are bottlenecked by the CPU

- In a simple benchmark consisting of aggregating data, the **CPU is the bottleneck**
- This is after the data is parsed and cached into memory which is another common bottleneck
- The CPU bottleneck is even worse in more complex workloads!

SELECT cab_type, count(*) FROM trips_orc GROUP BY cab_type;

```
top - 08:54:14 up  1:50,  4 users,  load average: 0.20, 1.64, 6.43
Tasks: 360 total,   2 running, 358 sleeping,   0 stopped,   0 zombie
%Cpu0  : 94.7 us,  1.7 sy,  0.0 ni,  3.3 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0
%Cpu1  : 95.0 us,  1.7 sy,  0.0 ni,  3.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu2  : 98.3 us,  0.3 sy,  0.0 ni,  1.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu3  : 87.3 us,  4.3 sy,  0.0 ni,  8.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu4  : 95.0 us,  1.3 sy,  0.0 ni,  3.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu5  : 98.3 us,  0.0 sy,  0.0 ni,  1.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu6  : 96.7 us,  1.3 sy,  0.0 ni,  2.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu7  : 92.7 us,  1.0 sy,  0.0 ni,  5.6 id,  0.3 wa,  0.0 hi,  0.3 si,  0.0
%Cpu8  : 93.7 us,  1.3 sy,  0.0 ni,  5.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu9  : 92.3 us,  0.7 sy,  0.0 ni,  7.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu10 : 97.3 us,  0.7 sy,  0.0 ni,  2.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu11 : 97.3 us,  0.7 sy,  0.0 ni,  2.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu12 : 92.0 us,  3.0 sy,  0.0 ni,  5.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu13 : 94.9 us,  1.0 sy,  0.0 ni,  4.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu14 : 88.3 us,  3.0 sy,  0.0 ni,  8.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu15 : 92.6 us,  2.3 sy,  0.0 ni,  4.7 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0
%Cpu16 : 94.7 us,  2.3 sy,  0.0 ni,  2.6 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0
%Cpu17 : 93.0 us,  0.7 sy,  0.0 ni,  6.0 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0
%Cpu18 : 93.0 us,  3.7 sy,  0.0 ni,  3.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
%Cpu19 : 91.2 us,  0.7 sy,  0.0 ni,  8.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
```

Source: Mark Litwintschik's blog: 1.1 Billion Taxi Rides: EC2 versus EMR

HOW CAN WE DO BETTER?

- Focus on the full Data Science workflow
  - Data Loading
  - Data Transformation
  - Data Analytics
- Python
  - Provide as close to a drop-in replacement for existing tools
- Performance - Leverage GPUs

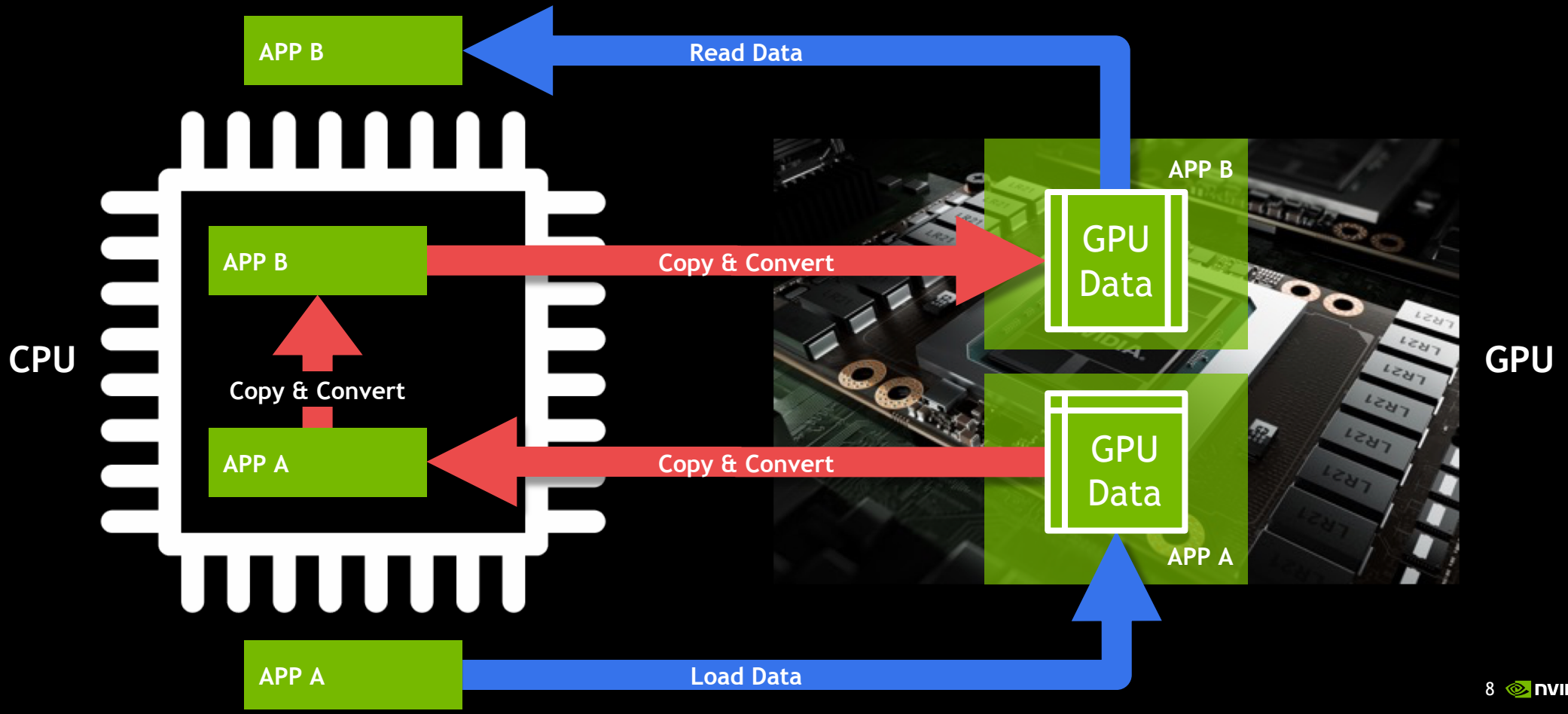
GPU ADOPTION BARRIERS

Yes GPUs are fast but …

- Too much data movement
- Too many makeshift data formats
- Writing CUDA C/C++ is hard
- No ***Python*** API for data manipulation

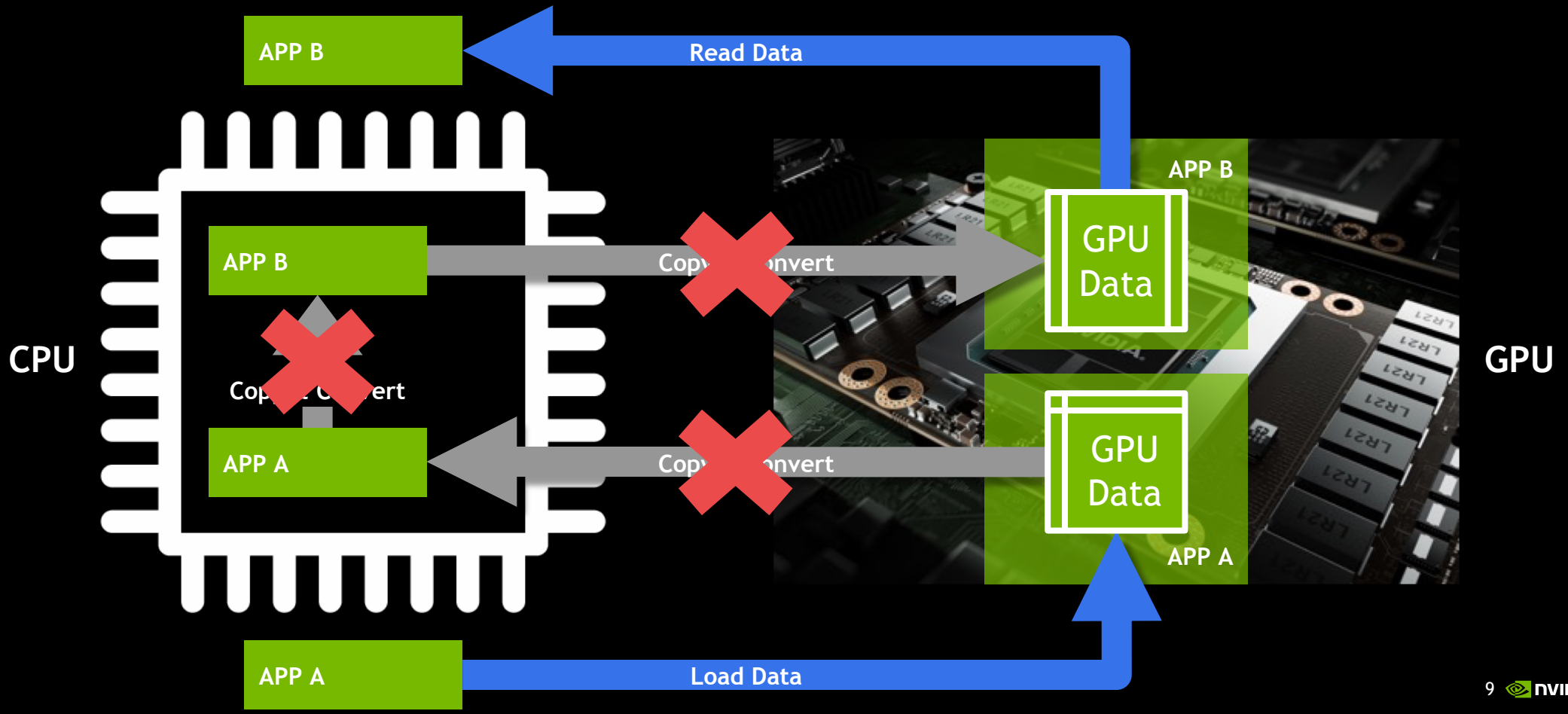
DATA MOVEMENT AND TRANSFORMATION

The bane of productivity and performance

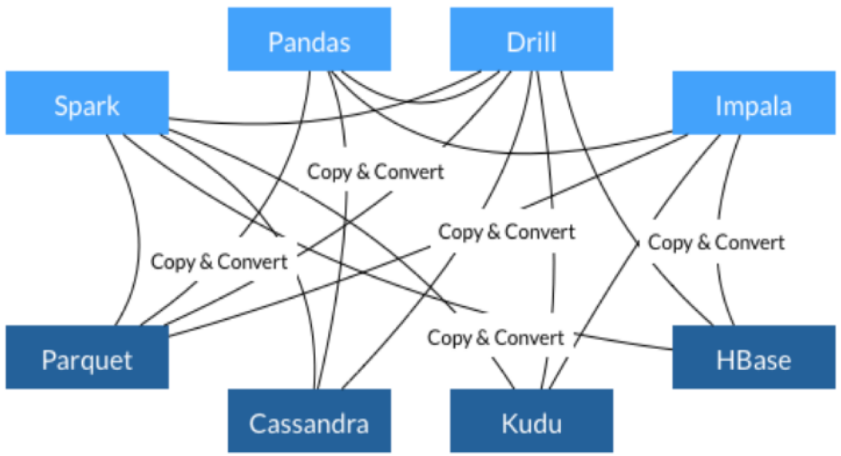


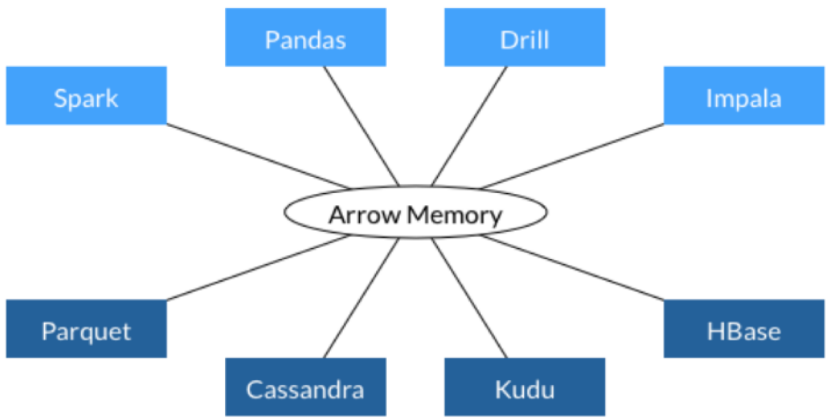
DATA MOVEMENT AND TRANSFORMATION

What if we could keep data on the GPU?



LEARNING FROM APACHE ARROW



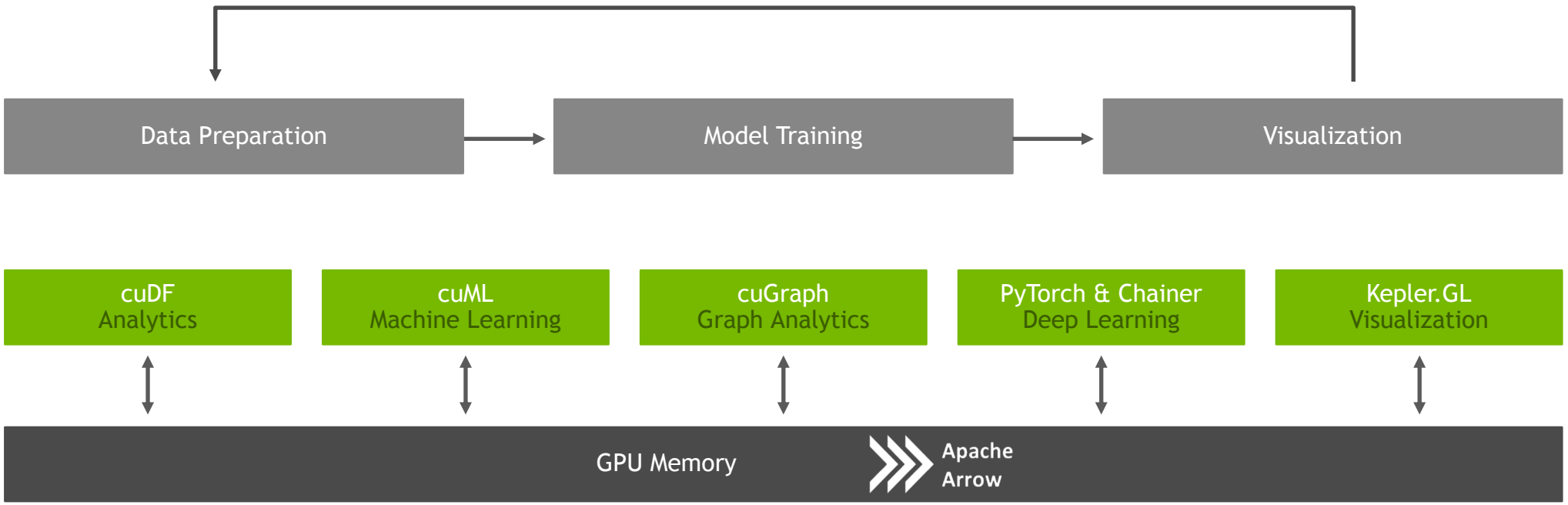


- Each system has its own internal memory format
- 70-80% computation wasted on serialization and deserialization
- Similar functionality implemented in multiple projects

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

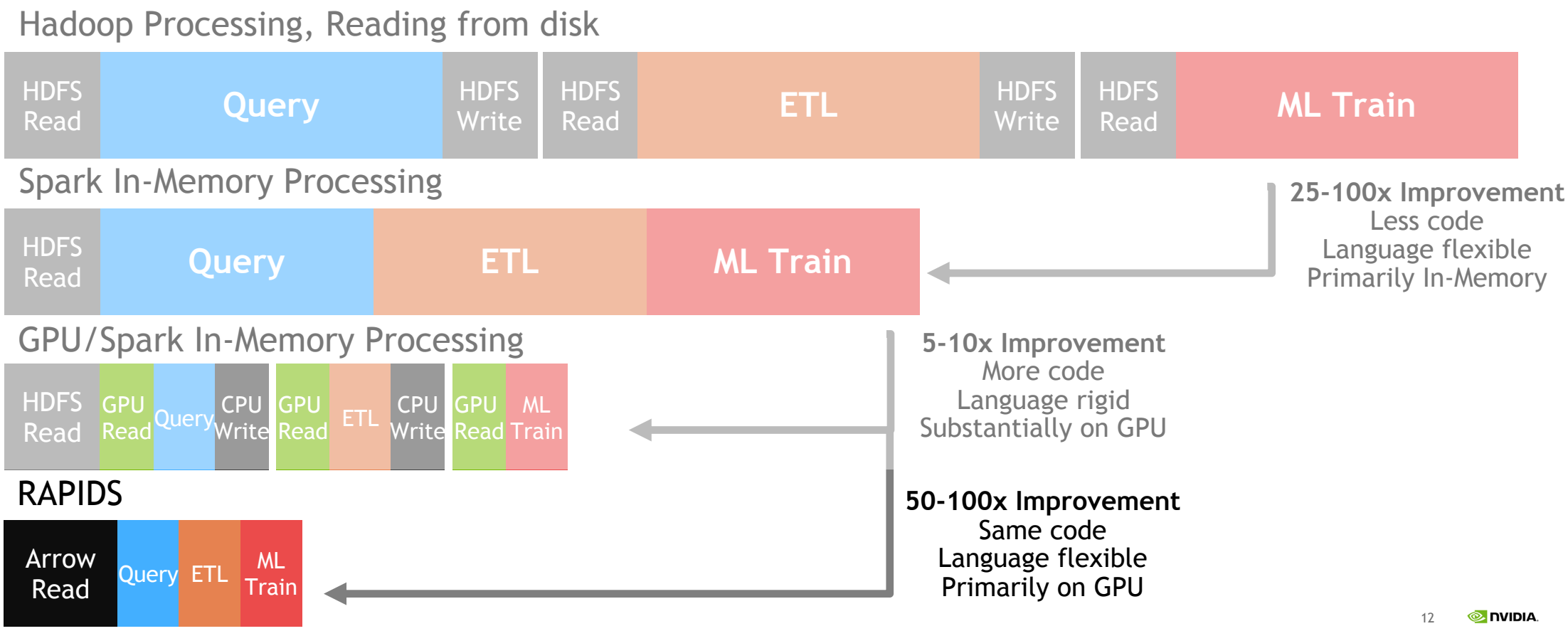
From Apache Arrow Home Page - https://arrow.apache.org/

RAPIDS OPEN SOURCE SOFTWARE



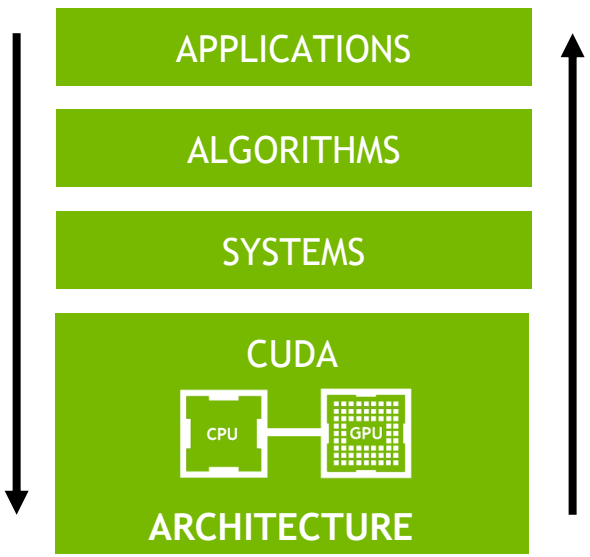
DATA PROCESSING EVOLUTION

Faster Data Access Less Data Movement



RAPIDS

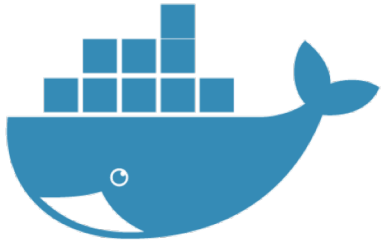
Open GPU Data Science



- Learn what the data science community needs
- Use best practices and standards
- Build scalable systems and algorithms
- Test Applications and workflows
- Iterate

RAPIDS

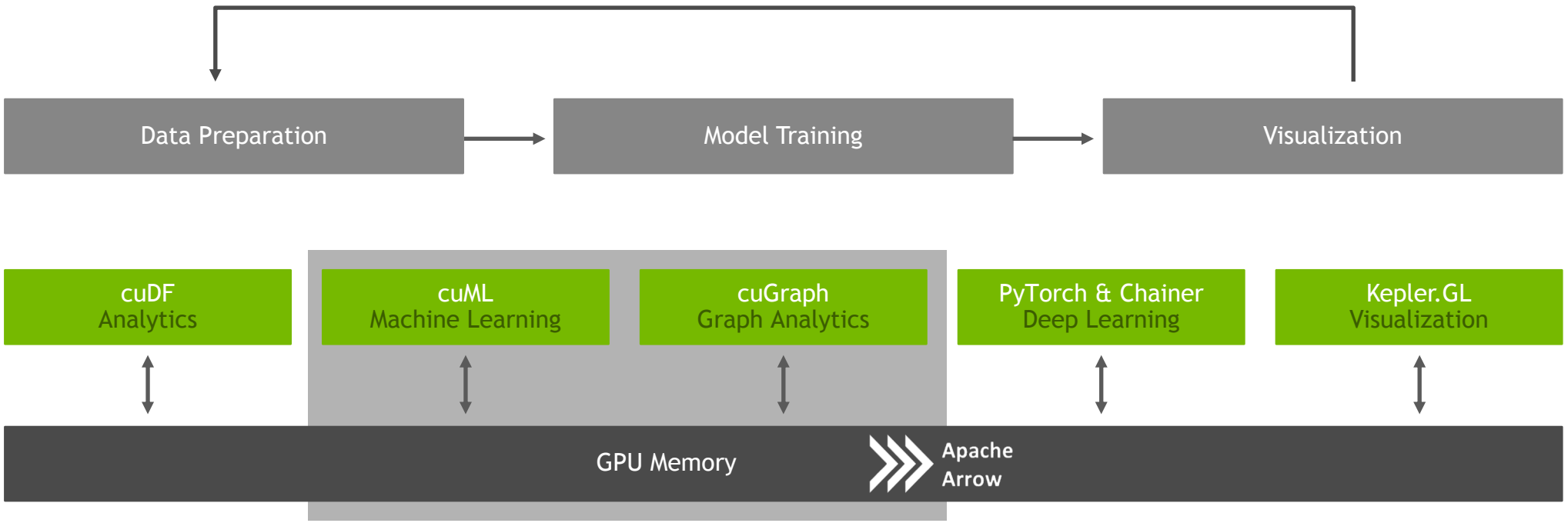
How do I get the software?



- https://github.com/rapidsai
- WIP: https://anaconda.org/rapidsai/
- WIP:
  - https://pypi.org/project/cudf
  - https://pypi.org/project/cuml

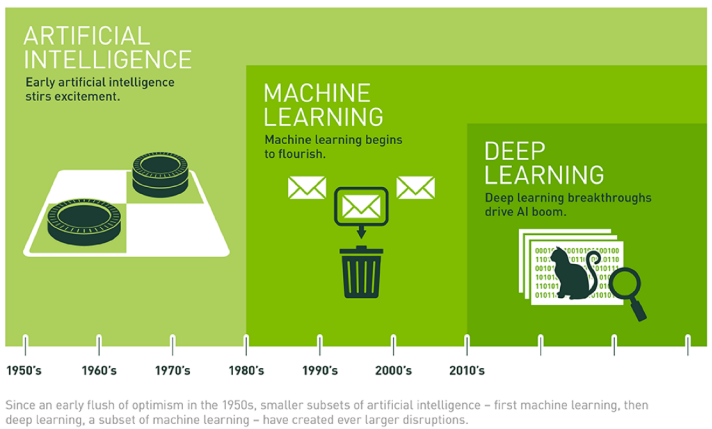
- https://ngc.nvidia.com/registry/nvidia-rapidsai-rapidsai
- https://hub.docker.com/r/rapidsai/rapidsai/

CUML & CUGRAPH



AI LIBRARIES

cuML & cuGraph



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Accelerating more of the AI ecosystem

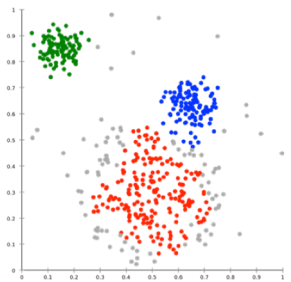
Graph Analytics is fundamental to network analysis

Machine Learning is fundamental to prediction, classification, clustering, anomaly detection and recommendations.

Both can be accelerated with NVIDIA GPU

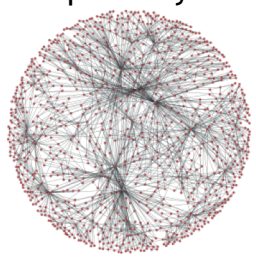
8x V100 20-90x faster than dual socket CPU

Machine Learning



Decisions Trees
Random Forests
Linear Regressions
Logistics Regressions
K-Means
K-Nearest Neighbor
DBSCAN
Kalman Filtering
Principal Components
Single Value Decomposition
Bayesian Inferencing

Graph Analytics



PageRank
BFS
Jaccard Similarity
Single Source Shortest Path
Triangle Counting
Louvain Modularity

Time Series

ARIMA
Holt-Winters

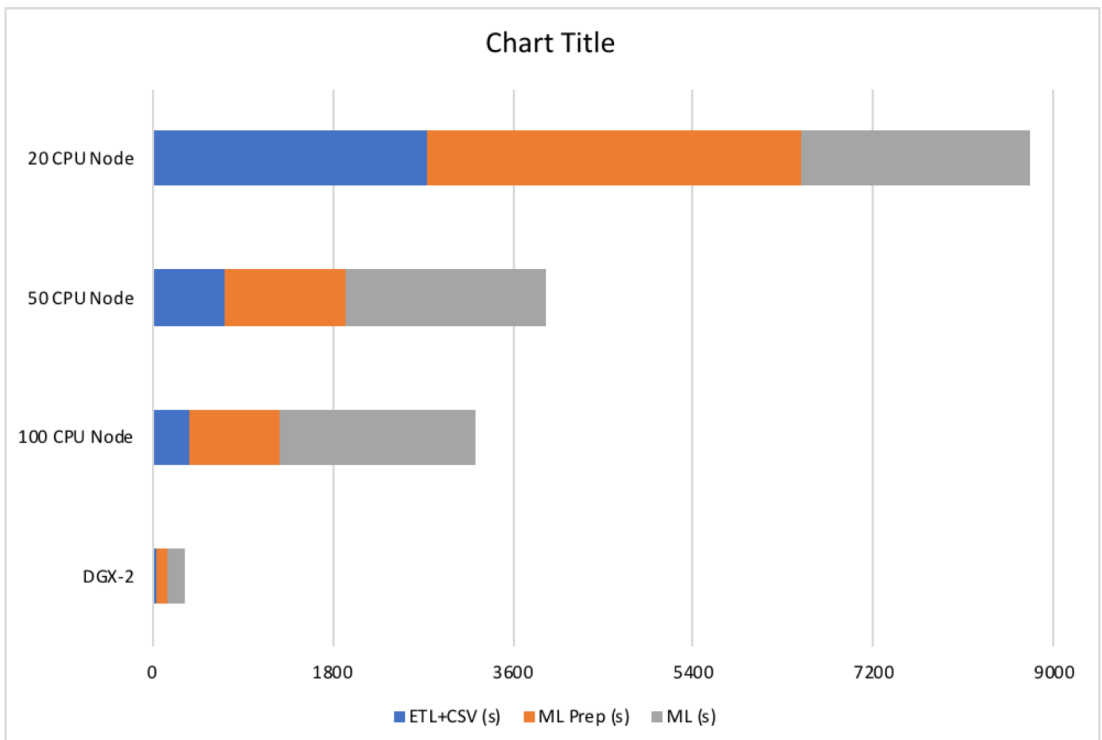
XGBoost, Criteo Dataset, 90x

3 Hours to 2 mins on 1 DGX-1

NVIDIA

CUDF + XGBOOST

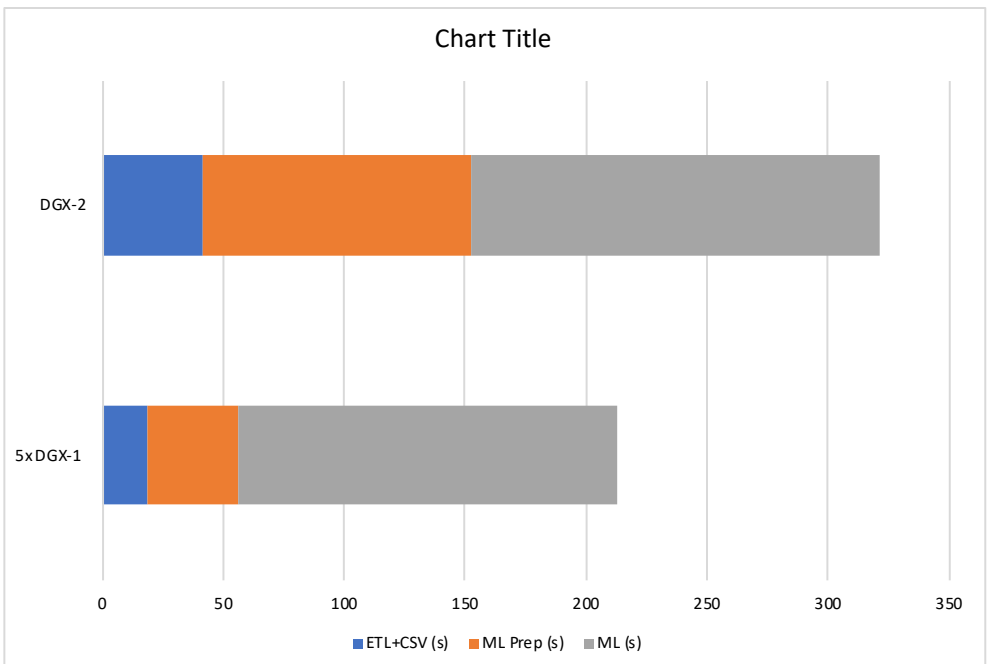
DGX-2 vs Scale Out CPU Cluster



- Full end to end pipeline
- Leveraging Dask + PyGDF
- Store each GPU results in sys mem then read back in
- Arrow to Dmatrix (CSR) for XGBoost

CUDF + XGBOOST

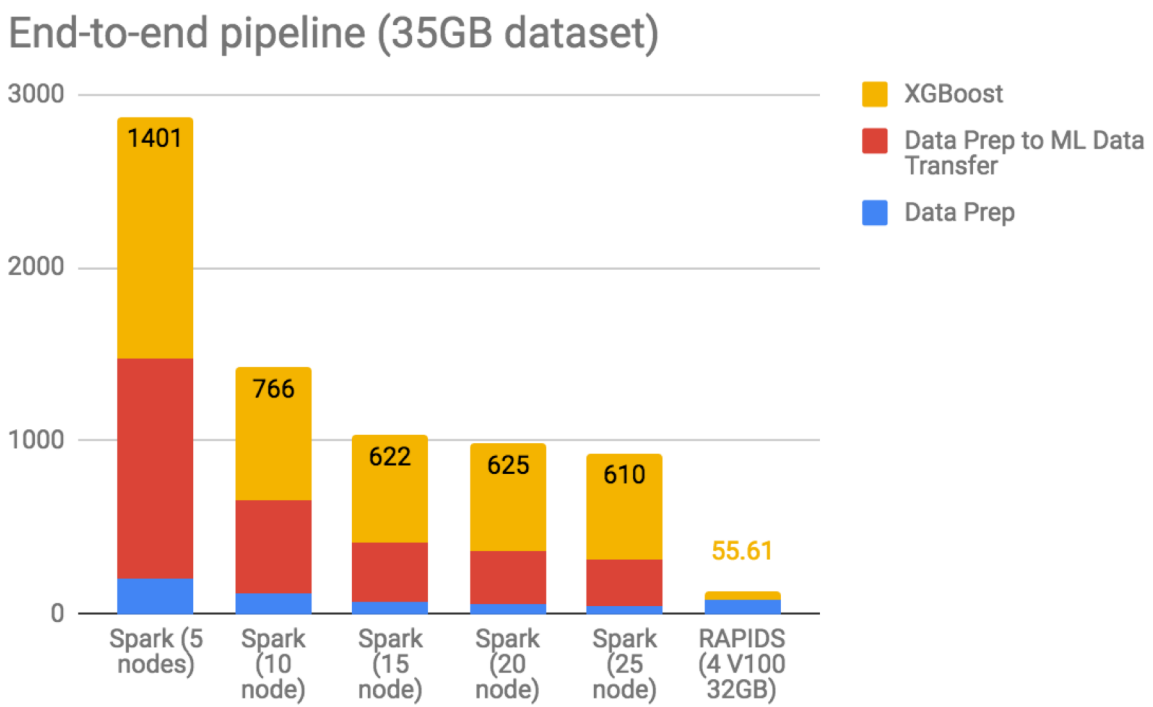
Scale Out GPU Cluster vs DGX-2



- Full end to end pipeline
- Leveraging Dask for multi-node + PyGDF
- Store each GPU results in sys mem then read back in
- Arrow to Dmatrix (CSR) for XGBoost

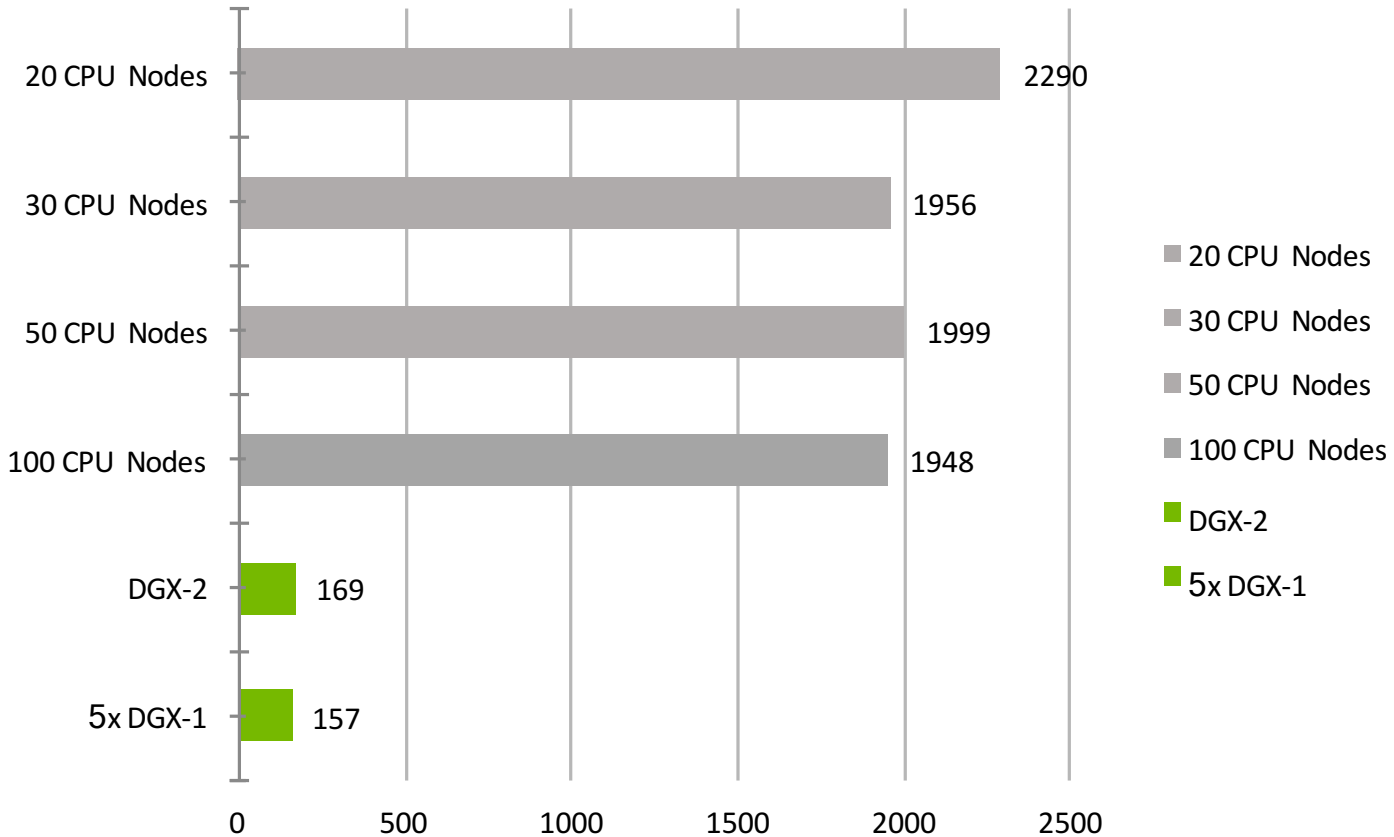
CUDF + XGBOOST

Fully In- GPU Benchmarks



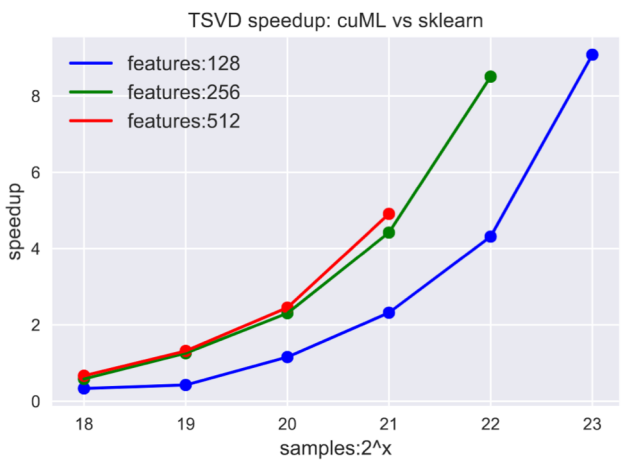
- Full end to end pipeline
- Leveraging DaskGDF
- No Data Prep time all in memory
- Arrow to Dmatrix (CSR) for XGBoost

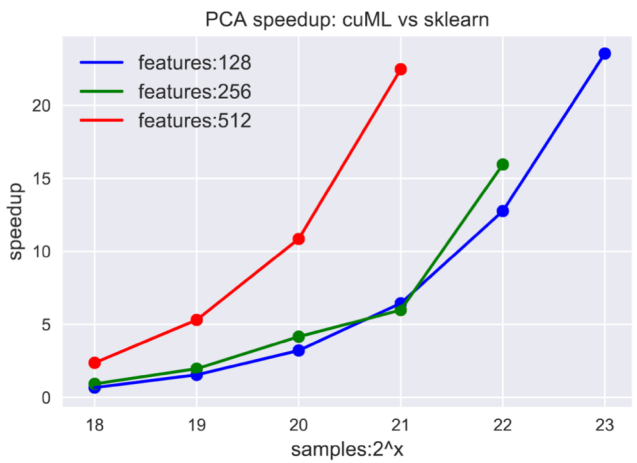
XGBOOST
MULTI-NODE, MULTI-GPU PERFORMANCE

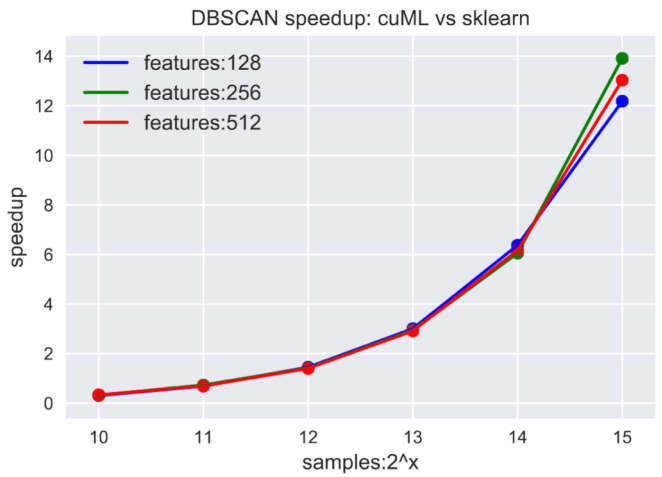


CUML

Benchmarks of initial algorithms







NEAR FUTURE WORK ON CUML

Additional algorithms in development right now

K-means

K-NN

Kalman filter

Collaborative filtering

GLM

Random Forests

FUTURE WORK ON CUML

Roadmapped algorithms for development

ARIMA

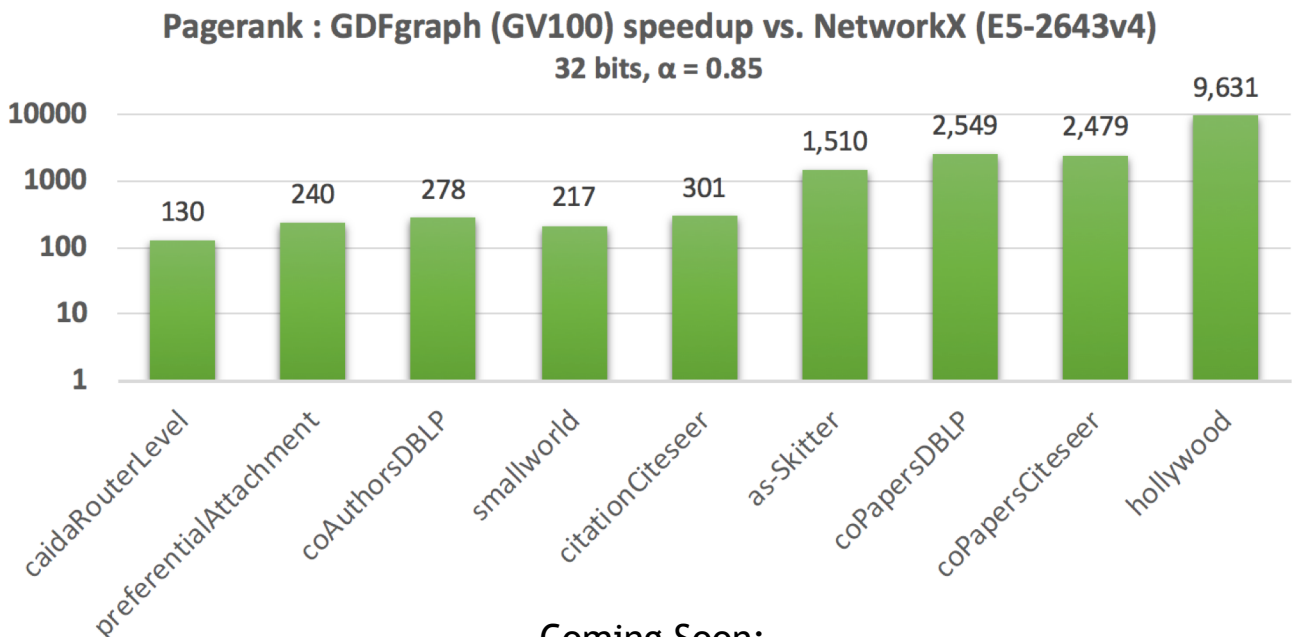
Holt-Winters (triple exponential) smoothing for time series

UMAP - Uniform Manifold Approximation and Projection for Dimension Reduction

Bayesian methods

CUGRAPH

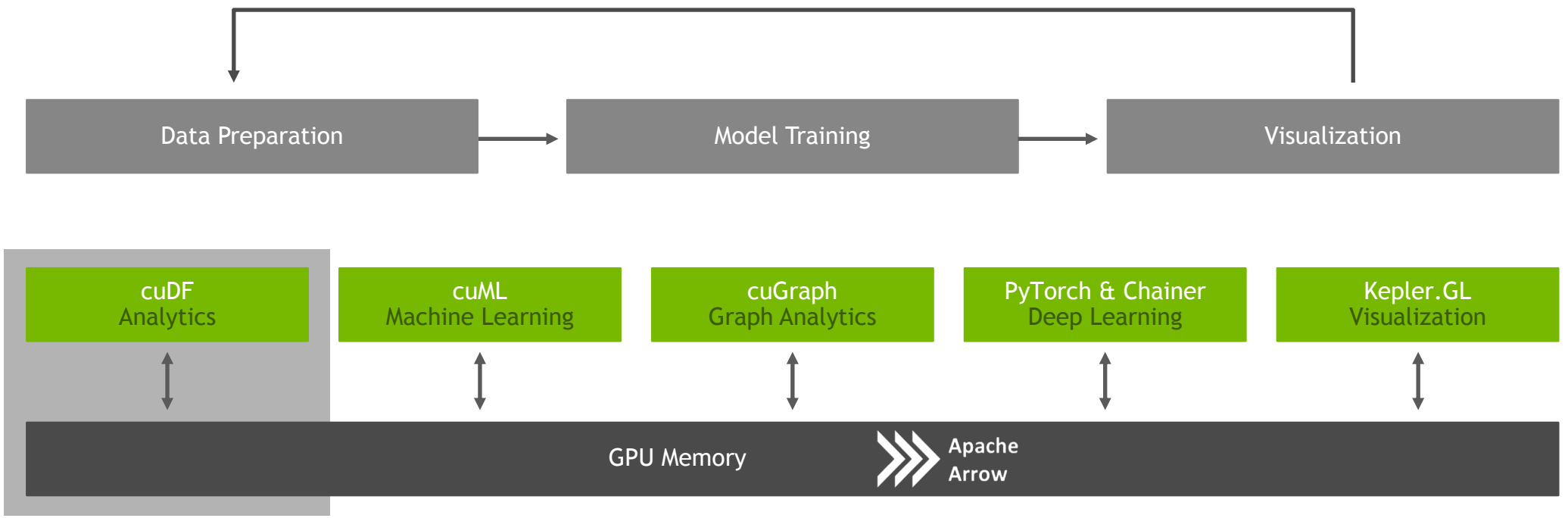
GPU-Accelerated Graph Analytics Library



Coming Soon:
PageRank, BFS, Triangle Counting with NetworkX-like API

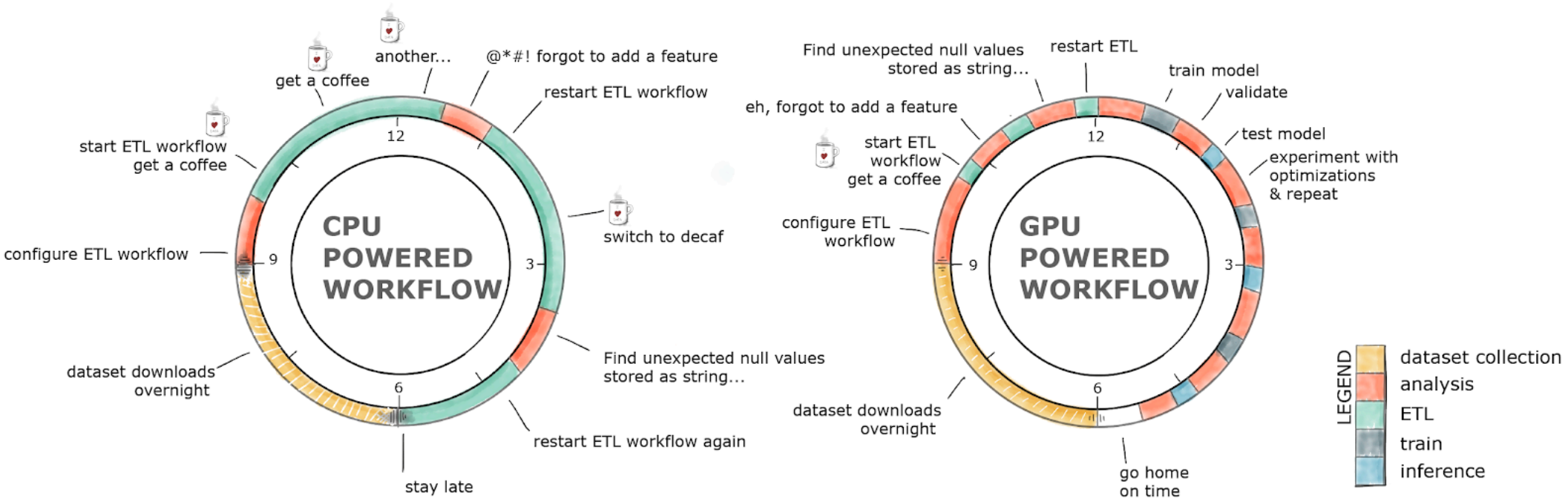
DC8110 - RAPIDS: Benchmarking Graph Analytics on the DGX-2
Brad Rees & Joe Eaton

CUDF


Data Preparation
Model Training
Visualization
cuDF
Analytics
cuML
Machine Learning
cuGraph
Graph Analytics
PyTorch & Chainer
Deep Learning
Kepler.GL
Visualization
GPU Memory
Apache
Arrow

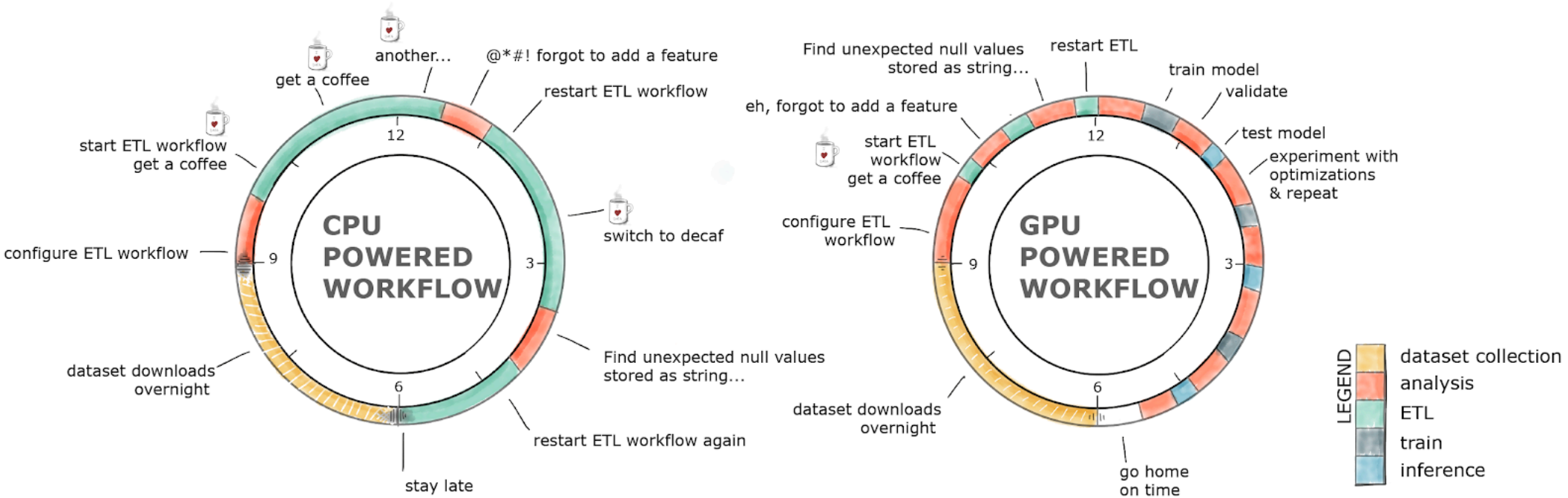
GPU-ACCELERATED ETL

Is GPU-acceleration really needed?



GPU-ACCELERATED ETL

The average data scientist spends 90+% of their time in ETL as opposed to training models



CUDF

GPU DataFrame library

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0
5	AF	2	Afghanistan	2514	Maize and products	5142	Food	1000 tonnes	33.94	67.71	...	231.0	67.0	82.0	67.0	69.0
6	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	...	15.0	21.0	11.0	19.0	21.0
7	AF	2	Afghanistan	2520	Cereals, Other	5142	Food	1000 tonnes	33.94	67.71	...	2.0	1.0	1.0	0.0	0.0
8	AF	2	Afghanistan	2531	Potatoes and products	5142	Food	1000 tonnes	33.94	67.71	...	276.0	294.0	294.0	260.0	242.0
9	AF	2	Afghanistan	2536	Sugar cane	5521	Feed	1000 tonnes	33.94	67.71	...	50.0	29.0	61.0	65.0	54.0
10	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	...	0.0	0.0	0.0	0.0	0.0

- Apache Arrow data format
- Pandas-like API

- Unary and Binary Operations
- Joins / Merges
- GroupBys
- Filters
- User-Defined Functions (UDFs)
- Accelerated file readers
- Etc.

CUDF

Today

LibGDF

- Low level library containing function implementations and C/C++ API
- Importing/exporting a GDF using the CUDA IPC mechanism
- CUDA kernels to perform element-wise math operations on GPU DataFrame columns
- CUDA sort, join, groupby, and reduction operations on GPU DataFrames

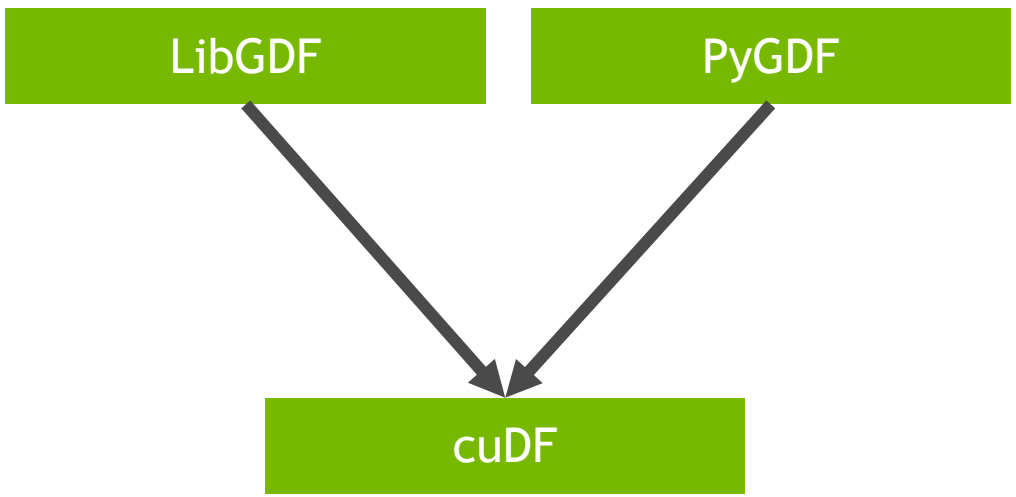
PyGDF

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

CUDF

Refactor in Progress

cuDF



- Single repository containing both the low level implementation and high level wrappers and APIs
- Future high level language bindings based on community demand, feedback, and contributions
- Moving from CFFI to Cython for Python bindings to better integrate into the PyData community

PANDAS-LIKE API

Python GPU DataFrame library

In [2]:
```
import numpy as np
import pandas as pd

import pygdf
```

```
time: 591 ms
```

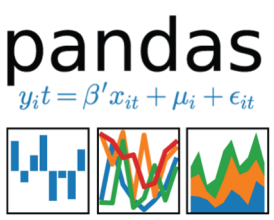
PANDAS-LIKE API

Pandas ↔ PyGDF

```
In [2]: import numpy as np
        import pandas as pd

        import pygdf
```

time: 591 ms



Convert to GPU DataFrame

```
In [12]: pa_df = pygdf.DataFrame.from_pandas(pa_df)
         pg_df = pygdf.DataFrame.from_pandas(pg_df)
```

time: 480 ms

```
In [13]: pa_df.dtypes
```

```
Out[13]: Arrival_Time        int64
         Creation_Time       int64
         x                 float64
         y                 float64
         z                 float64
         User             category
         Model            category
         Device           category
         gt               category
         dtype: object
```

time: 4.1 ms

PANDAS-LIKE API

Built-In Functions

Filter the data

In [9]:
```
# how many records do we have?
print("{} = Original # of records".format(len(gdf)))

# filter out
gdf = gdf.query('INCEARN >= 0')

# how many records do we have left?
print("{} = New # of records".format(len(gdf)))

# sanity check...
gdf.head(5)
```

Manipulate data with a user-defined function (UDF)

INCEARN is not a true representation of income earned. Let's adjust it by multiplying it by the ADJUST constant.

In [7]:
```
# define a function to adjust the incearn var
# so it more accurately represents income earned
adjust = gdf['ADJUST'][0]
def adjust_incearn(incearn):
    return adjust * incearn;

# apply it to the 'population' column
gdf['INCEARN'] = gdf['INCEARN'].applymap(adjust_incearn)

# drop the ADJUST column
gdf.drop_column('ADJUST')

# compute the mean
gdf['INCEARN'].mean()
```

Out[7]: 18637.0999154208

DASK

What is Dask and why does RAPIDS use it for scaling out?

- Dask is a distributed computation scheduler built to scale Python workloads from laptops to supercomputer clusters.
- Extremely modular with scheduling, compute, data transfer, and out-of-core handling all being disjointed allowing us to plug in our own implementations.
- Can easily run multiple Dask workers per node to allow for an easier development model of one worker per GPU regardless of single node or multi node environment.

DASK

Scale up and out with cuDF

- Use cuDF primitives underneath in map-reduce style operations with the same high level API
- Instead of using typical Dask data movement of pickling objects and sending via TCP sockets, take advantage of hardware advancements using a communications framework called OpenUCX:
  - For intranode data movement, utilize NVLink and PCIe peer-to-peer communications
  - For internode data movement, utilize GPU RDMA over Infiniband and RoCE

https://github.com/rapidsai/dask_gdf

DASK

Scale up and out with cuML

- Native integration with Dask + cuDF
- Can easily use Dask workers to initialize NCCL for optimized gather / scatter operations
  - Example: this is how the dask-xgboost included in the container works for multi-GPU and multi-node, multi-GPU
- Provides easy to use, high level primitives for synchronization of workers which is needed for many ML algorithms

LOOKING TO THE FUTURE

GPU DATAFRAME

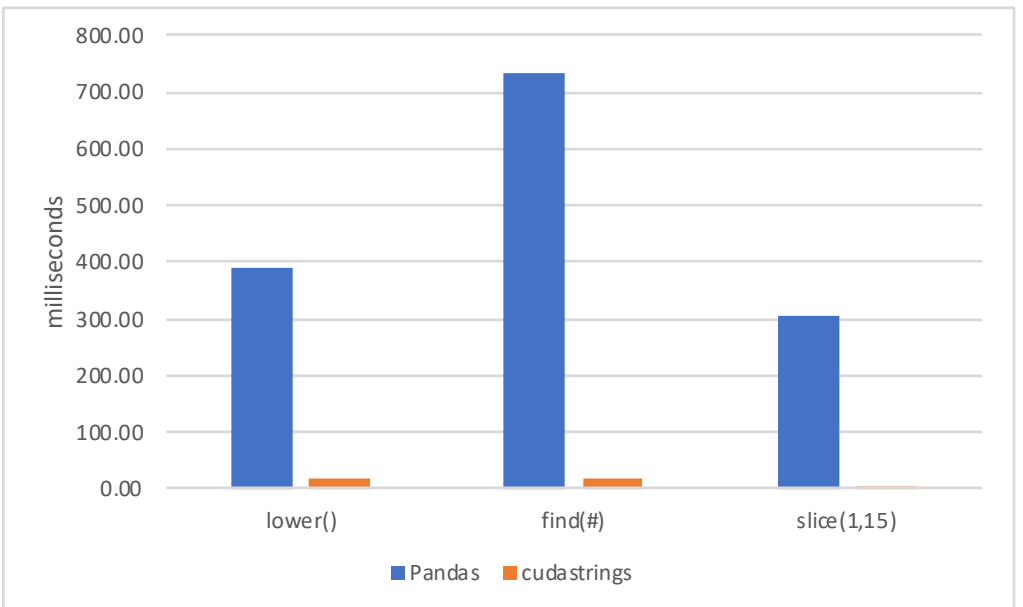
Next few months

- Continue improving performance and functionality
  - Single GPU
  - Single node, multi GPU
  - Multi node, multi GPU
- String Support
  - Support for specific “string” dtype with GPU-accelerated functionality similar to Pandas
- Accelerated Data Loading
  - File formats: CSV, Parquet, ORC – to start

CUSTRING

GPU-Accelerated string functions with a Pandas-like API

- API and functionality is following Pandas: https://pandas.pydata.org/pandas-docs/stable/api.html#string-handling
- lower()
  - ~22x speedup
- find()
  - ~40x speedup
- slice()
  - ~100x speedup

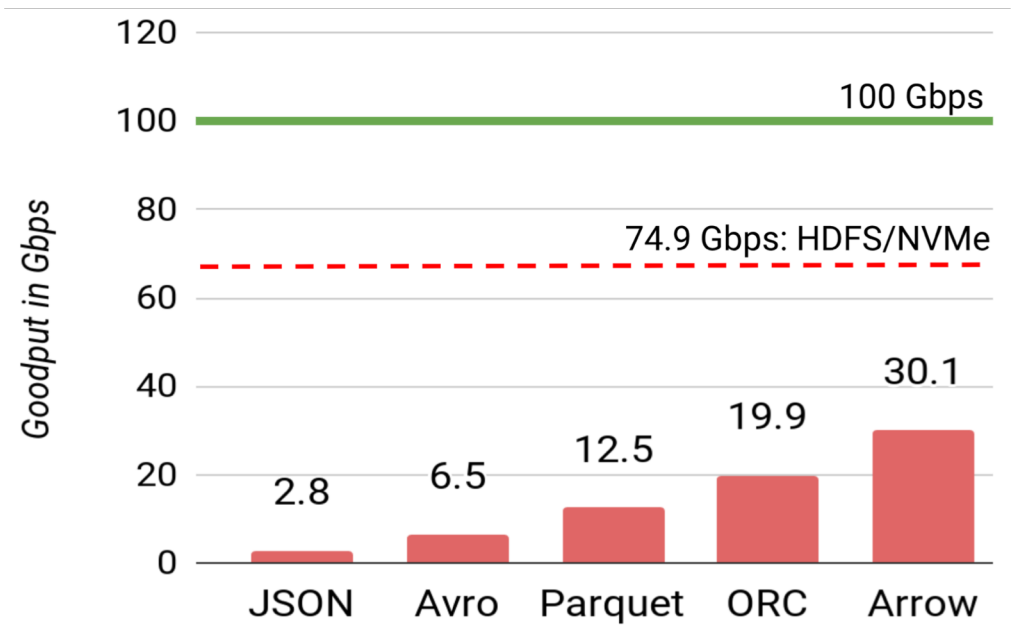


ACCELERATED DATA LOADING

CPUs bottleneck data loading in high throughput systems

- CSV Reader
  - Follows API of pandas.read_csv
  - Current implementation is >10x speed improvement over pandas
- Parquet Reader
  - Work in progress: https://github.com/gpuopenanalytics/libgdf/pull/85
  - Will follow API of pandas.read_parquet
- ORC Reader

- Additionally looking towards GPU-accelerating decompression for common compression schemes



Source: Apache Crail blog: SQL Performance: Part 1 - Input File Formats

PYTHON CUDA ARRAY INTERFACE

Interoperability for Python GPU Array Libraries

- The CUDA array interface is a standard format that describes a GPU array to allow sharing GPU arrays between different libraries without needing to copy or convert data

- Numba, CuPy, and PyTorch are the first libraries to adopt the interface:
  - https://numba.pydata.org/numba-doc/dev/cuda/cuda_array_interface.html
  - https://github.com/cupy/cupy/releases/tag/v5.0.0b4
  - https://github.com/pytorch/pytorch/pull/11984

Numba

CuPy

JOIN THE MOVEMENT

Everyone Can Help!

APACHE ARROW

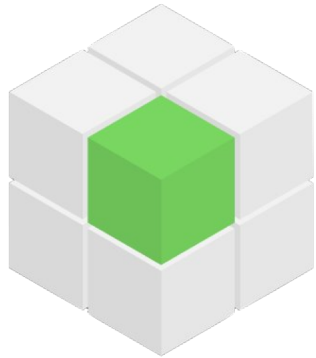
https://arrow.apache.org/

@ApacheArrow

RAPIDS

https://rapids.ai

@RAPIDSAI



GPU Open Analytics Initiative

http://gpuopenanalytics.com/

@GPUOAI

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

THANK YOU

Joshua Patterson

@datametrician

NVIDIA