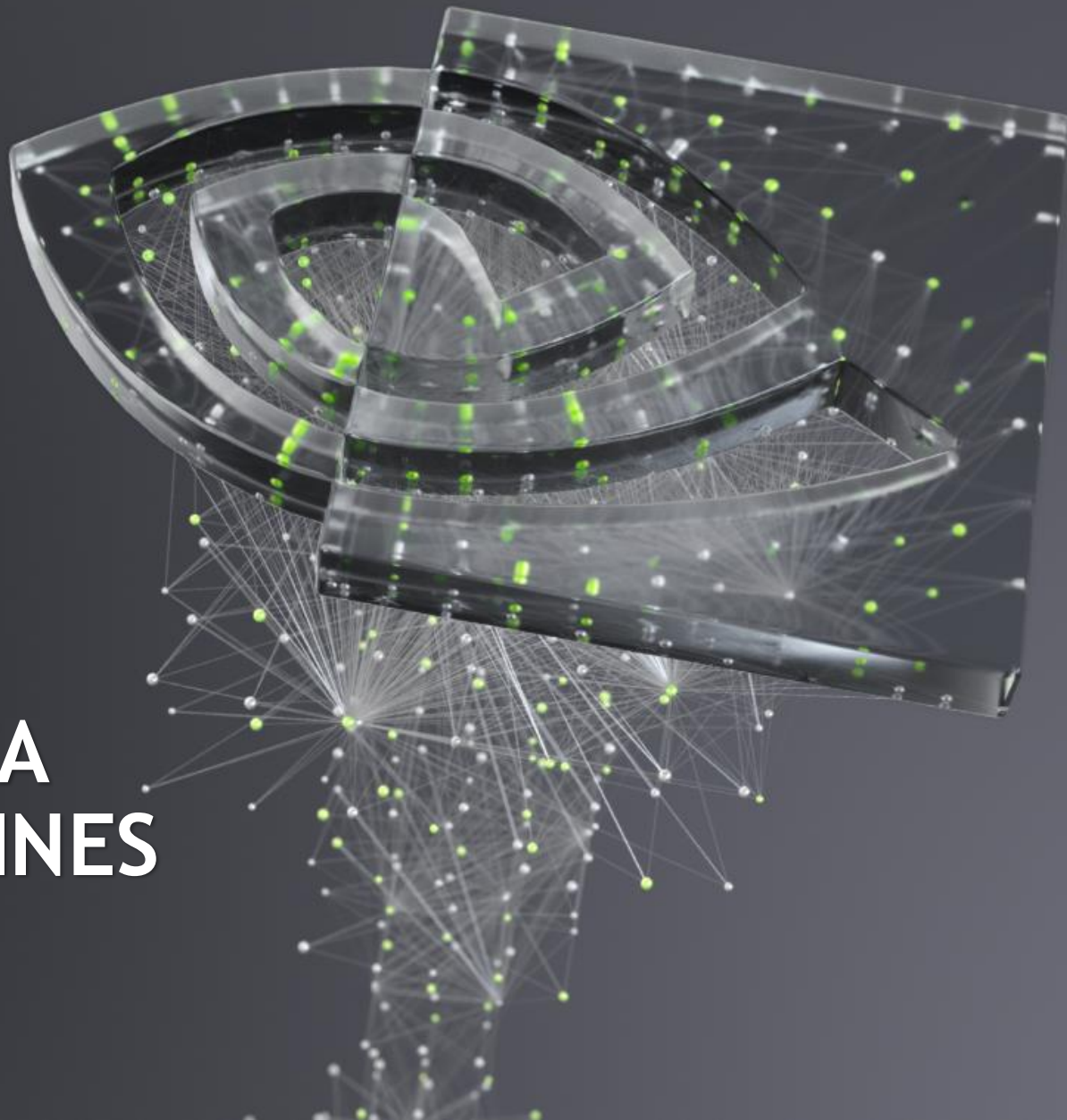




DEEP
LEARNING
INSTITUTE

ACCELERATING DATA ENGINEERING PIPELINES

Part 2: Extract, Transform, Load



AGENDA

Part 1: Data Formats

Part 2: ETL with NVTabular

Part 3: Data Visualization

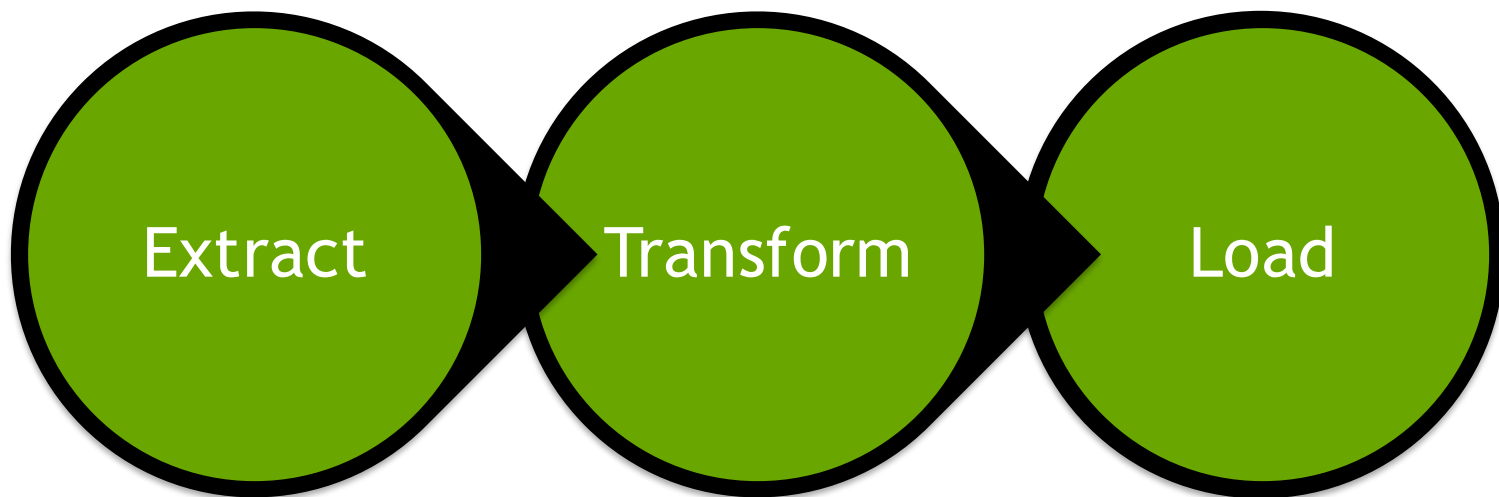
AGENDA – PART 2

- ETL Basics
- CUDA
- NVTabular
- Lab



ETL BASICS

DATA MANIPULATION IN 3 “EASY” STEPS



- Pull data from a database
 - SQL, Blob Storage, Image Archive

- Alter the data in some way
 - Cleaning, deduping, feature engineering

- Export transformed data to a new database location

NOT SO EASY TO OPTIMIZE

All parts of the technology stack come into play



Individual Machines

- CPU
- GPU
- RAM
- PCIe



Software

- Python vs C vs SQL
- Algorithm efficiency
- MapReduce
- GPU support

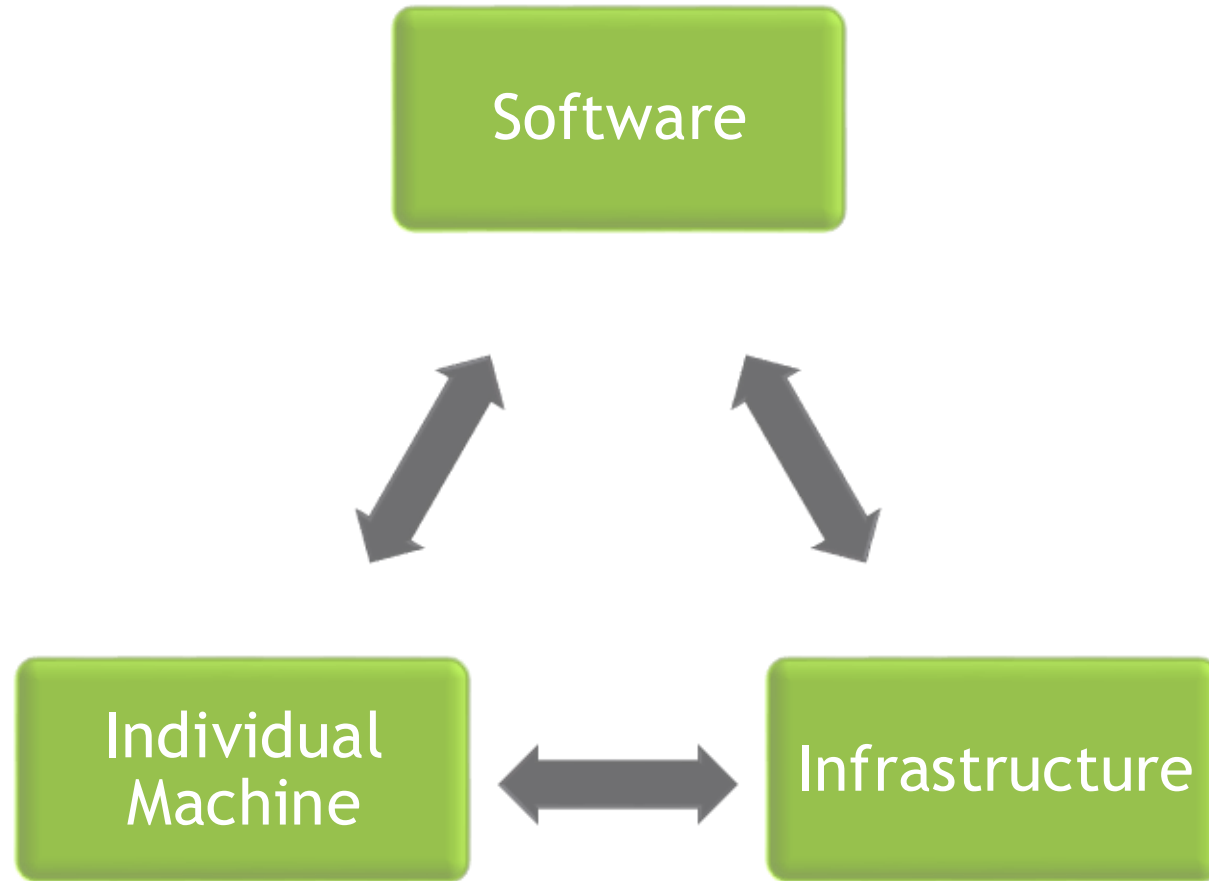


Infrastructure

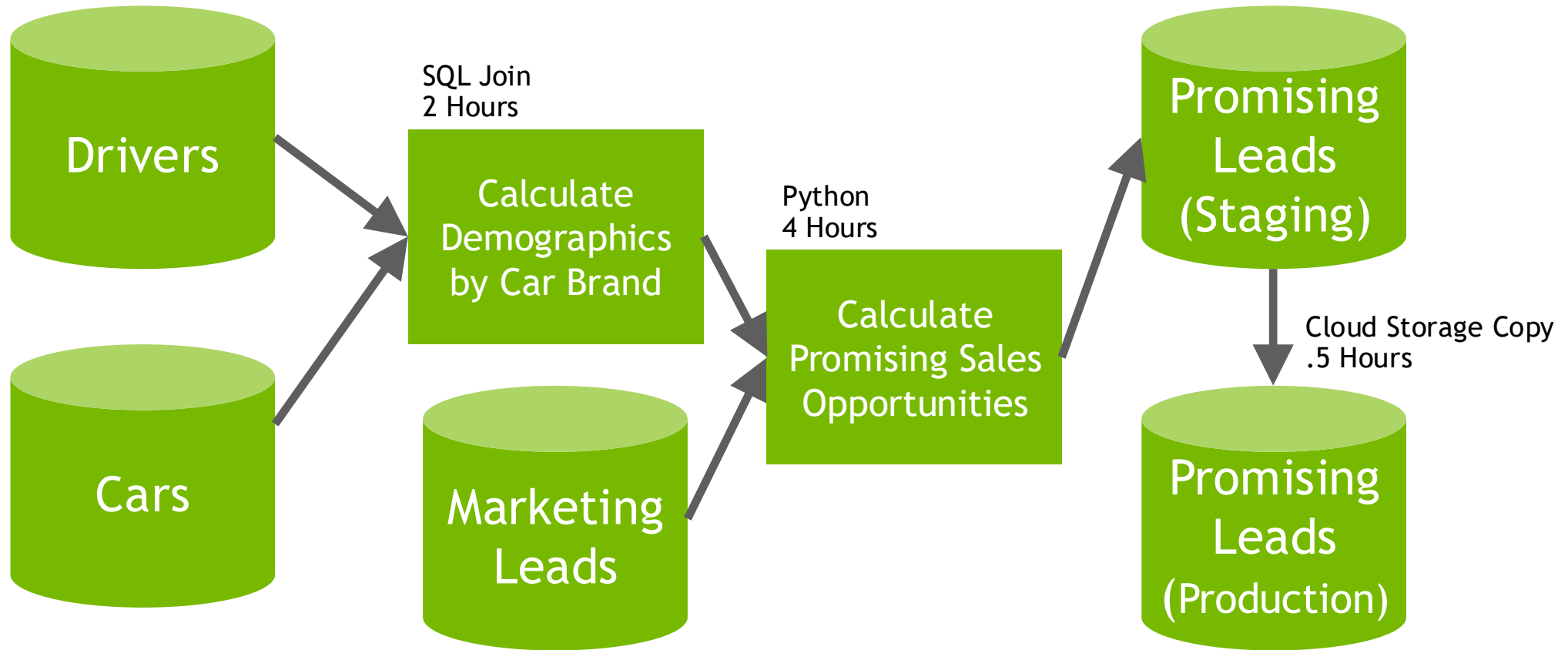
- Number of servers
- Bandwidth
- Database location
- Client location



THE MOST EFFICIENT PIPELINES CONSIDER ALL AT ONCE



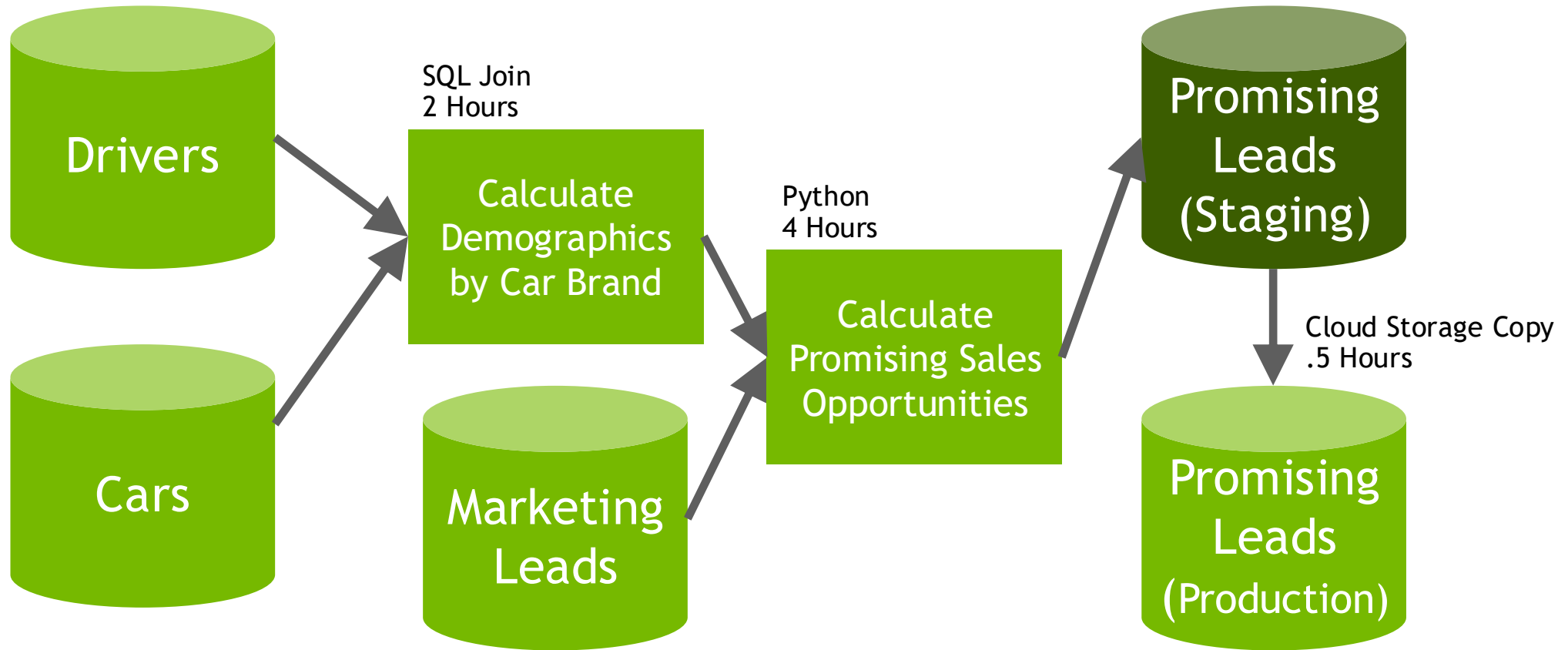
ETL SYSTEMS ENGINEERING



ETL SYSTEMS ENGINEERING



ETL SYSTEMS ENGINEERING

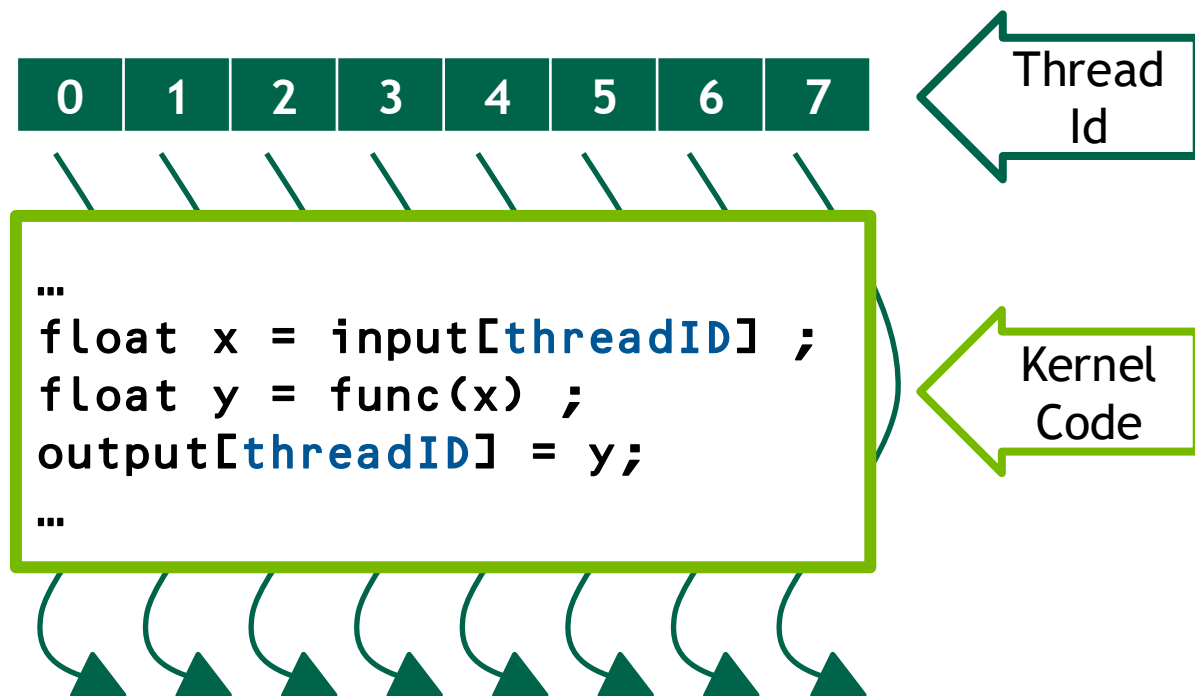




CUDA BASICS

CUDA COMPONENTS

Kernels and Threads



Kernel

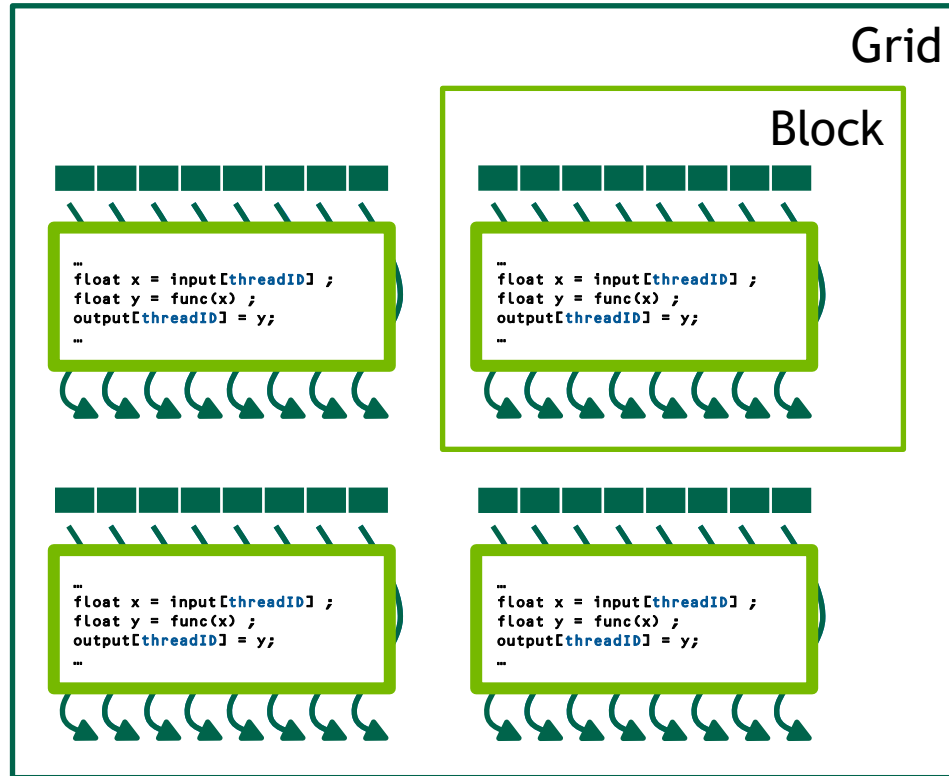
- A function to run in parallel on the GPU

Thread

- Runs an instance of the kernel

CUDA COMPONENTS

Blocks and Grids



Block

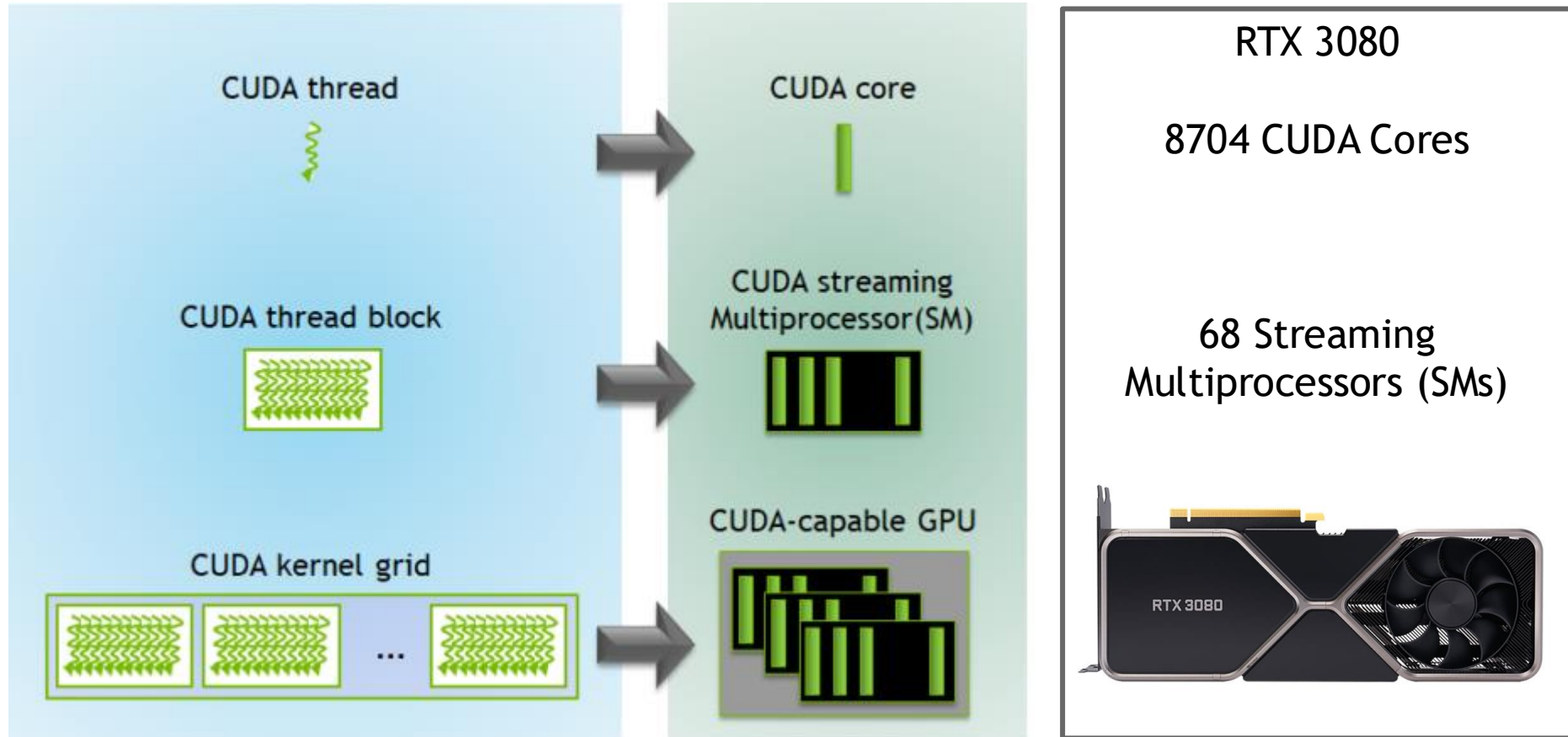
- A group of threads

Grid

- All blocks mapped on the GPU

CUDA COMPONENTS

Hardware to Software



MORE CORES MORE PERFORMANCE?

A Hardware Integration Example

Motherboard

CPU

RAM

Memory Drive

GPU

PSU

- If hardware is not balanced, bottlenecks occur
- The right hardware depends on the task

HOW DATA MOVES IN A COMPUTER

A Hardware Integration Example

Motherboard

CPU

1 - 2 Cores
per GPU.

RAM

Should be at least a little
bigger than GPU RAM. PCIe
lane speed can be a factor
with massive datasets.

Memory Drive

If the data is too large to be loaded
into RAM, then reading speed from the
memory drive can be a bottleneck.

GPU

GPUs excel on matrix multiplication computation which is used
frequently in ETL and ML model training. RAM speed, GPU speed
and Memory Bandwidth can all be a potential bottleneck for
computation. Model and data size will impact the efficacy of the
GPU.

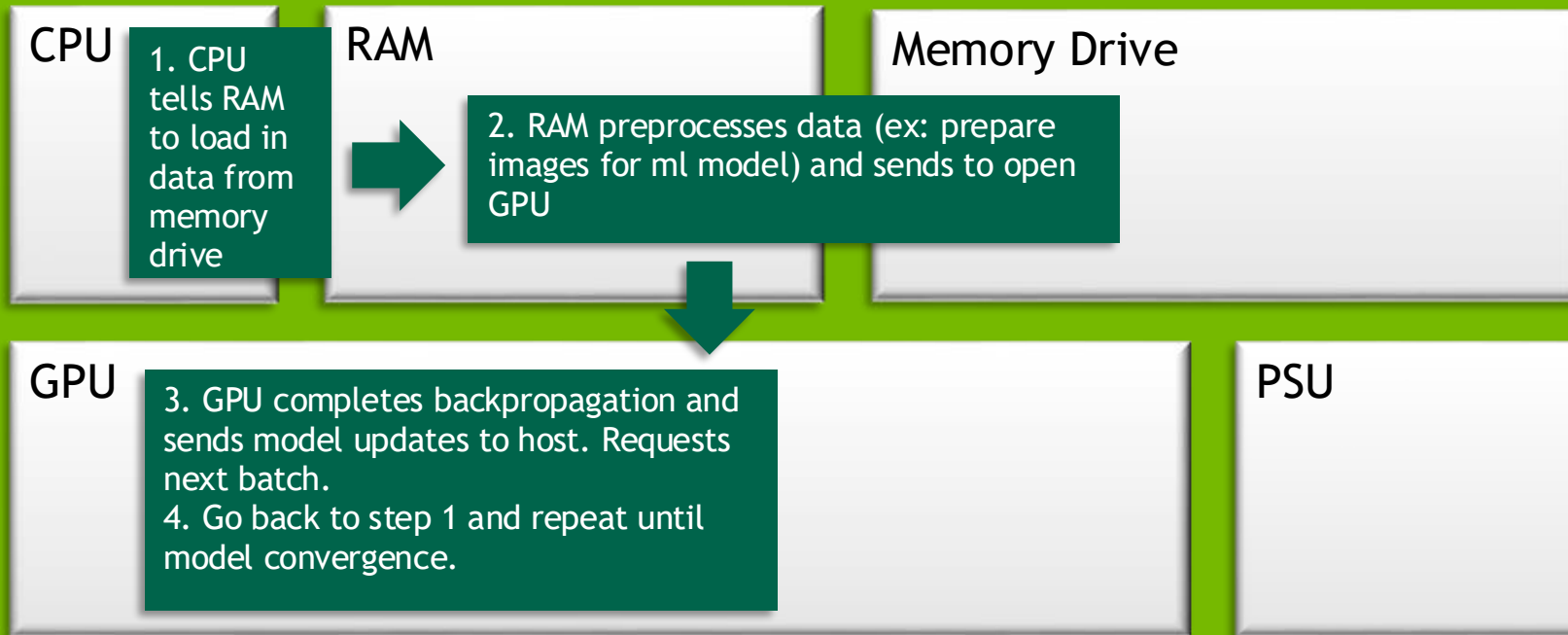
PSU

There should be
enough juice to
power all the cool
hardware. If it's
not cool, try
adding more fans.

HOW DATA MOVES IN A COMPUTER

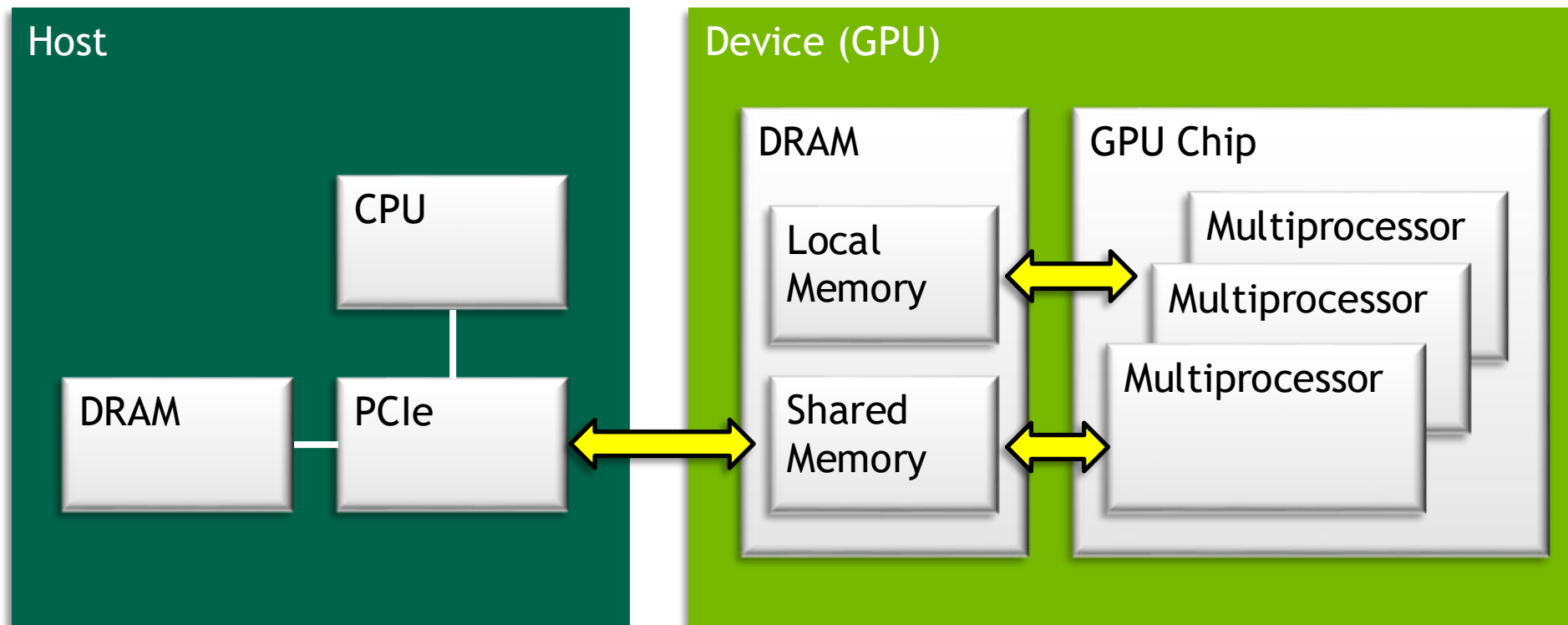
A Hardware Integration Example

Motherboard



DEBUGGING: WHY IS MY CPU FASTER THAN MY GPU?

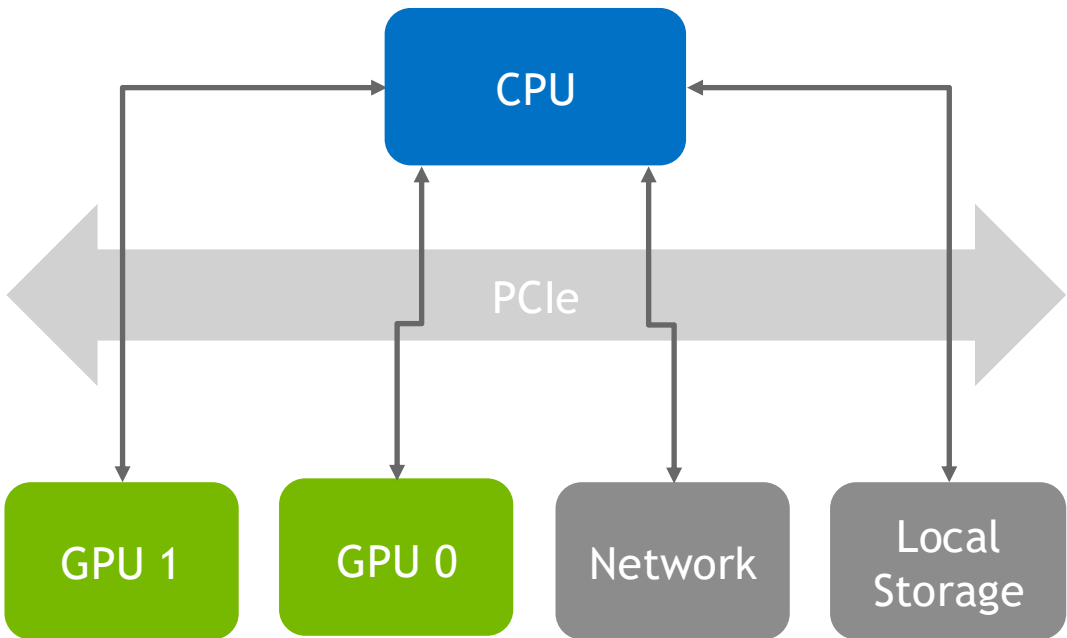
How data moves to the GPU with CUDA



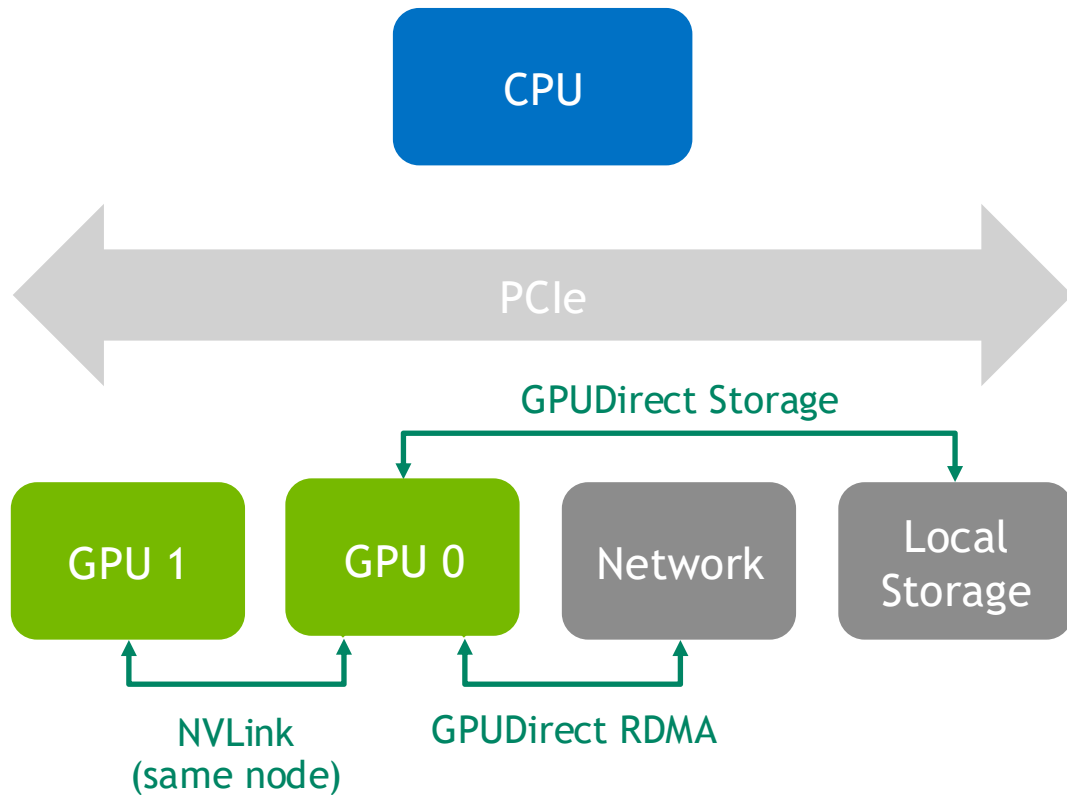
DATA MOVEMENT

CPU vs GPU

CPU-Centric Data Movement

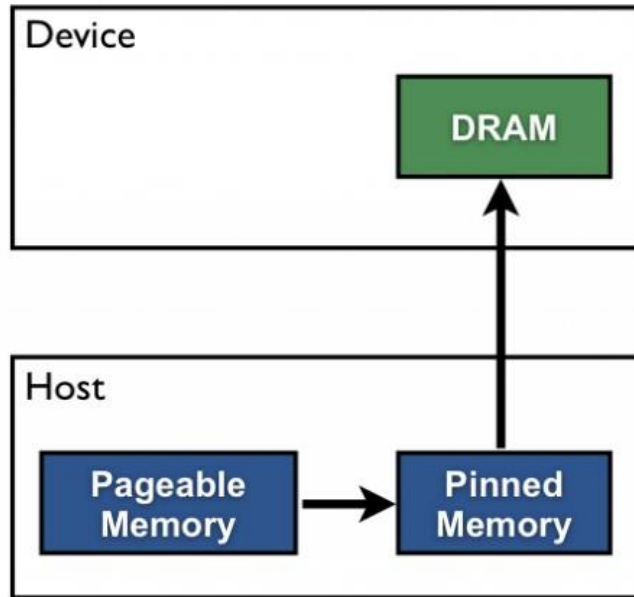


GPU-Centric Data Movement

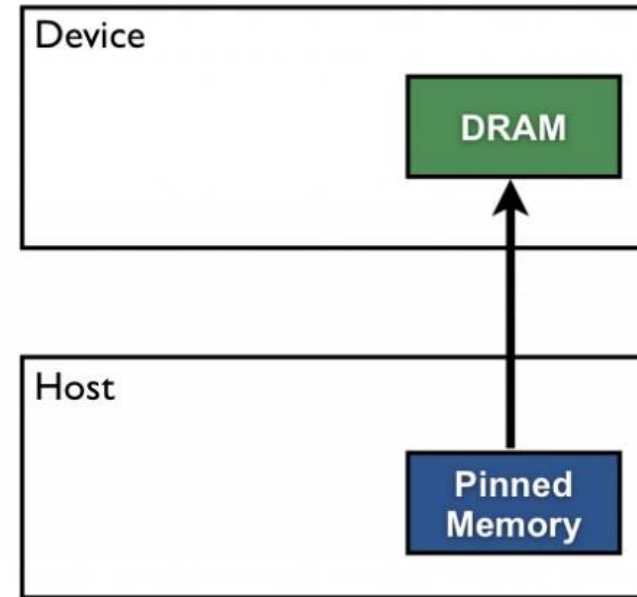


HOW TO OPTIMIZE DATA TRANSFERS

Pageable Data Transfer



Pinned Data Transfer

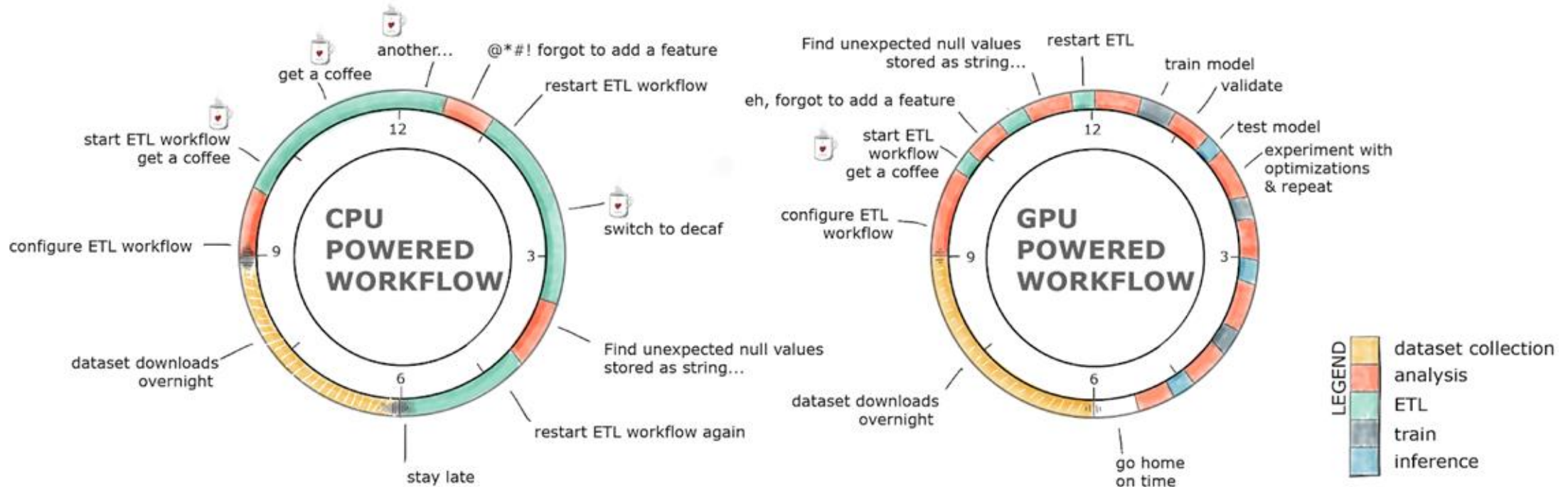




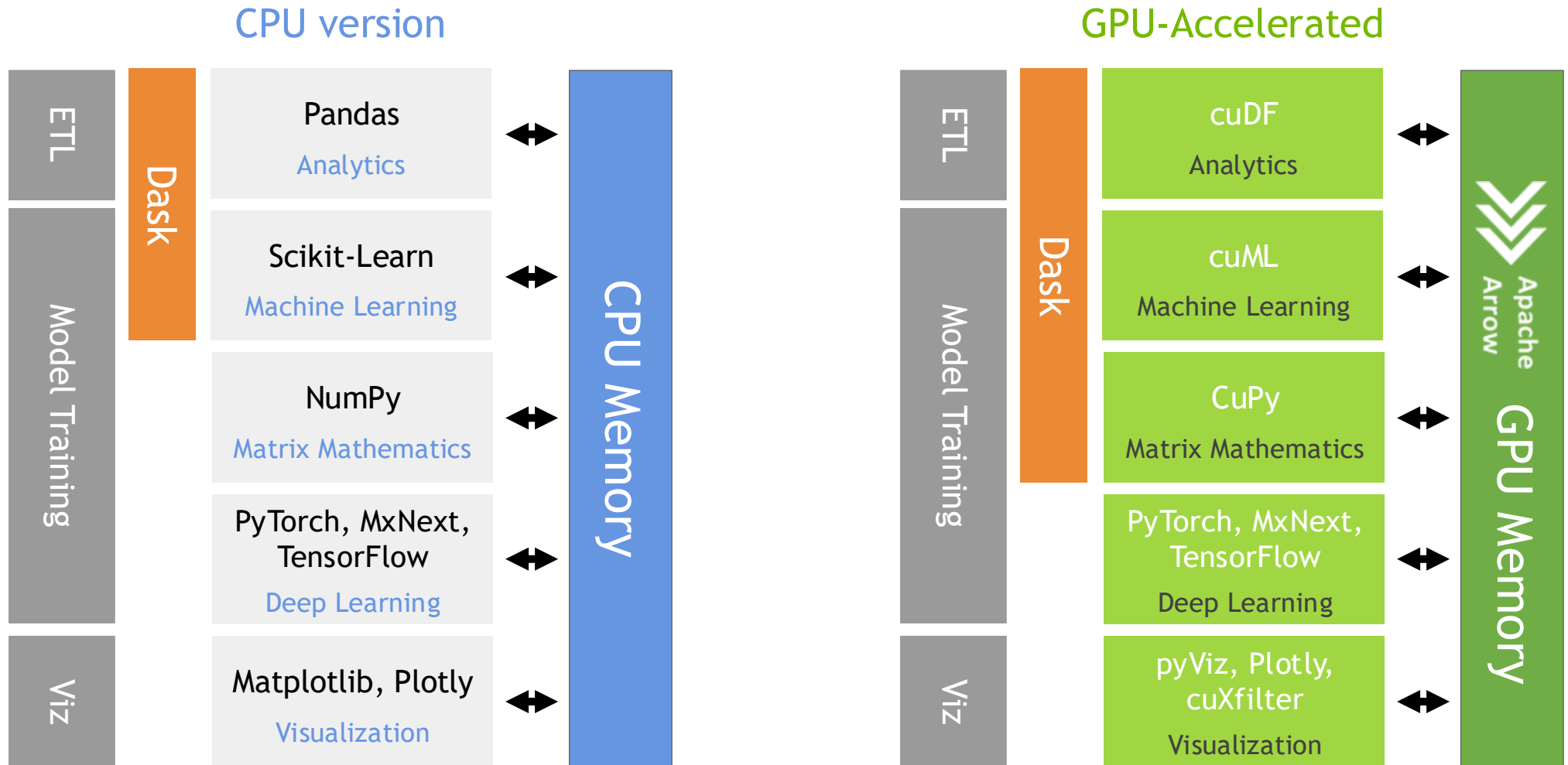
NVTABULAR

GPU-ACCELERATED ETL

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



Built on top of RAPIDS



NVTABULAR KEY FEATURES

Faster and Easier GPU-based ETL

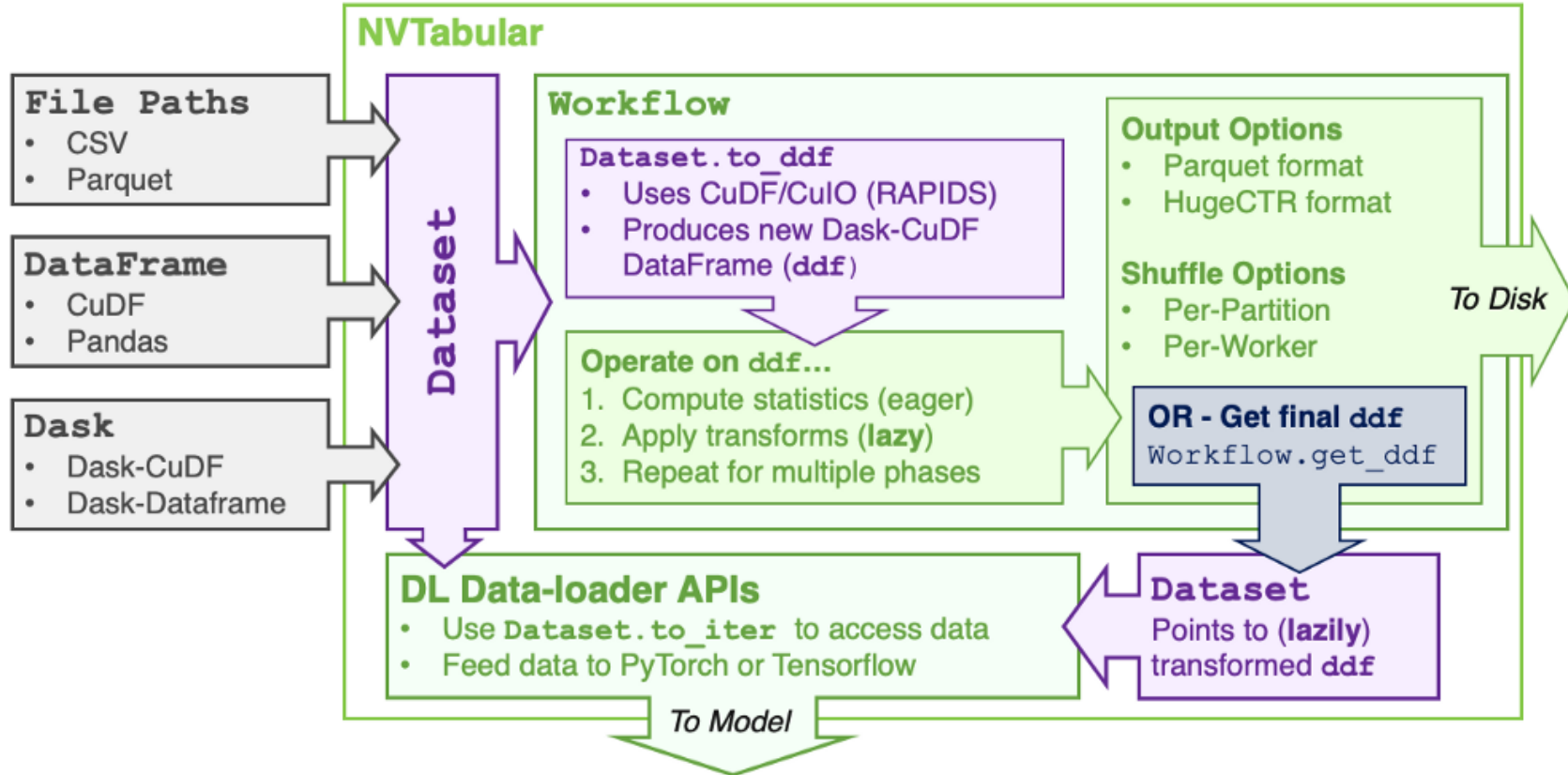
- GPU-accelerated, eliminating CPU bottlenecks.
- Out-of-core execution. No GPU memory limits and reduced I/O through lazy execution.
- PyTorch, TensorFlow and HugeCTR compatible.
- Filtering outliers or missing values.
- Inputting and filling in missing data.
- Discretization or bucketing of continuous features.
- Creating features by splitting or combining existing features.
- Normalizing numerical features to have zero mean and unit variance.
- Encoding discrete features using one-hot vectors or converting them to continuous integer indices.
- **More to come 😊**

NVTabular 

Dataset size limitation	Unlimited	CPU Memory
Code complexity	Simple	Moderate
Lines of code	10 - 20	100 - 1000
Flexibility	Domain specific	General
Data loading Transforms	Yes	No
Inference Transforms	Yes	No

NVTABULAR

Integration with RAPIDS/DASK



NVTabular vs Pandas code

100x fewer lines of code required

```
import glob
import nvtabular as nvt
```

```
# Create datasets from input files
```

```
train_files = glob.glob("./dataset/train/*.parquet")
valid_files = glob.glob("./dataset/valid/*.parquet")
```

```
train_ds = nvt.Dataset(train_files, gpu_memory_frac=0.1)
valid_ds = nvt.Dataset(valid_files, gpu_memory_frac=0.1)
```

```
# Initialise workflow
```

```
cat_names = ["C" + str(x) for x in range(1, 27)] # Specify categorical feature names
cont_names = ["I" + str(x) for x in range(1, 14)] # Specify continuous feature names
label_name = ["label"] # Specify target feature
```

```
proc = nvt.Workflow(cat_names=cat_names, cont_names=cont_names, label_name=label_name)
```

```
# Add feature engineering and pre-processing ops to workflow
```

```
proc.add_cont_feature([nvt.ops.ZeroFill(), nvt.ops.LogOp()])
proc.add_cont_preprocess(nvt.ops.Normalize())
proc.add_cat_preprocess(nvt.ops.Categorify(use_frequency=True, freq_threshold=15))
```

```
# Compute statistics, transform data, and export to disk
```

```
proc.apply(train_dataset, shuffle=True, output_path="./processed_data/train", num_out_files=len(train_files))
proc.apply(valid_dataset, shuffle=False, output_path="./processed_data/valid", num_out_files=len(valid_files))
```

Import libraries.

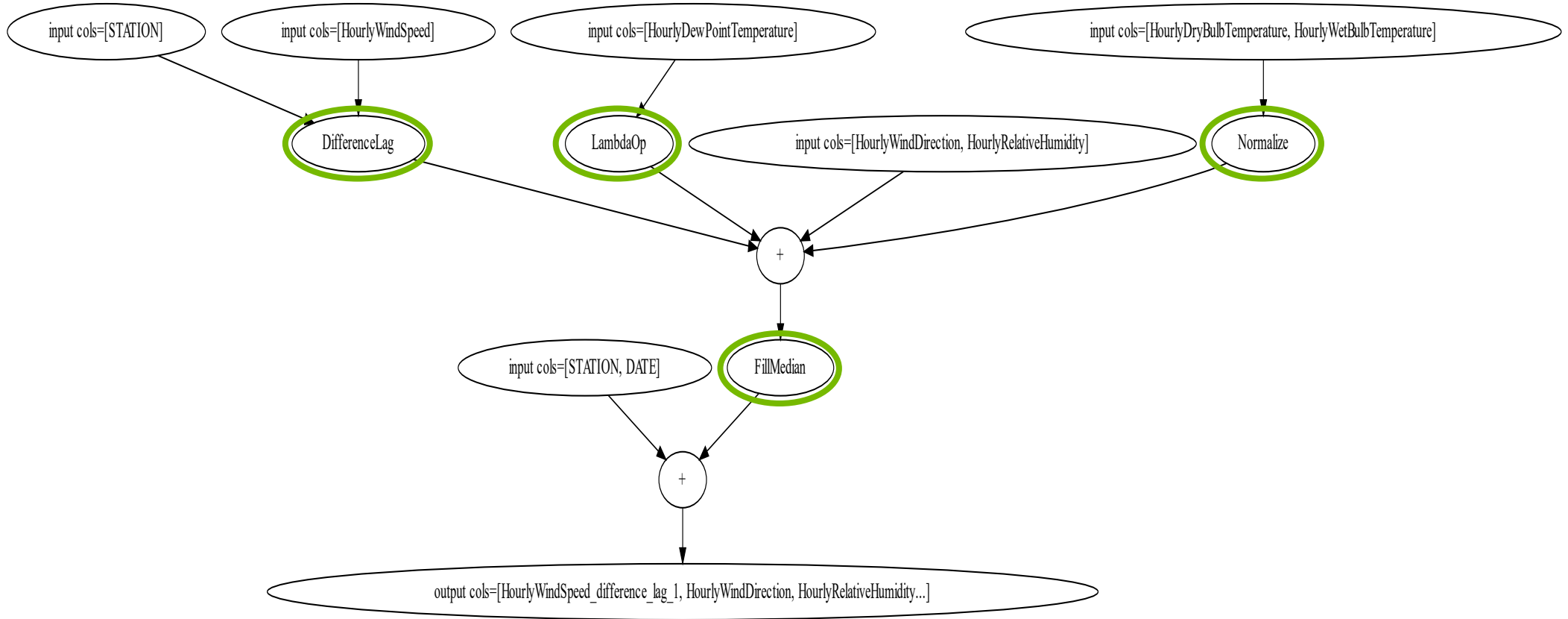
Create training and validation datasets.


Initialise workflow specifying categorical, and continuous data.

Zero fill any nulls, log transform and normalize continuous variables. Encode categorical data.

Apply the operations, creating new shuffled training and validation datasets.

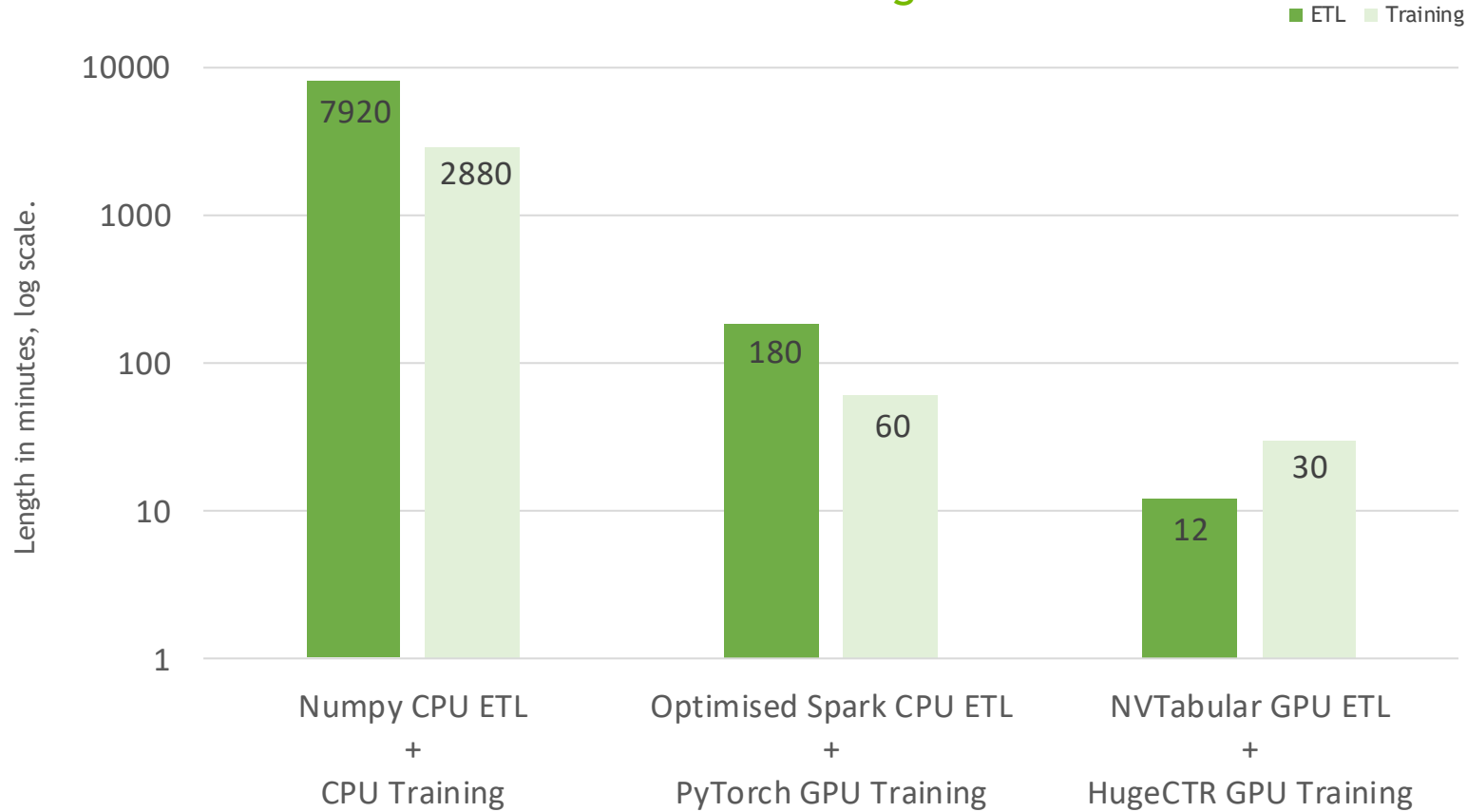
NVTABULAR DAG



 = NVTabular Operations

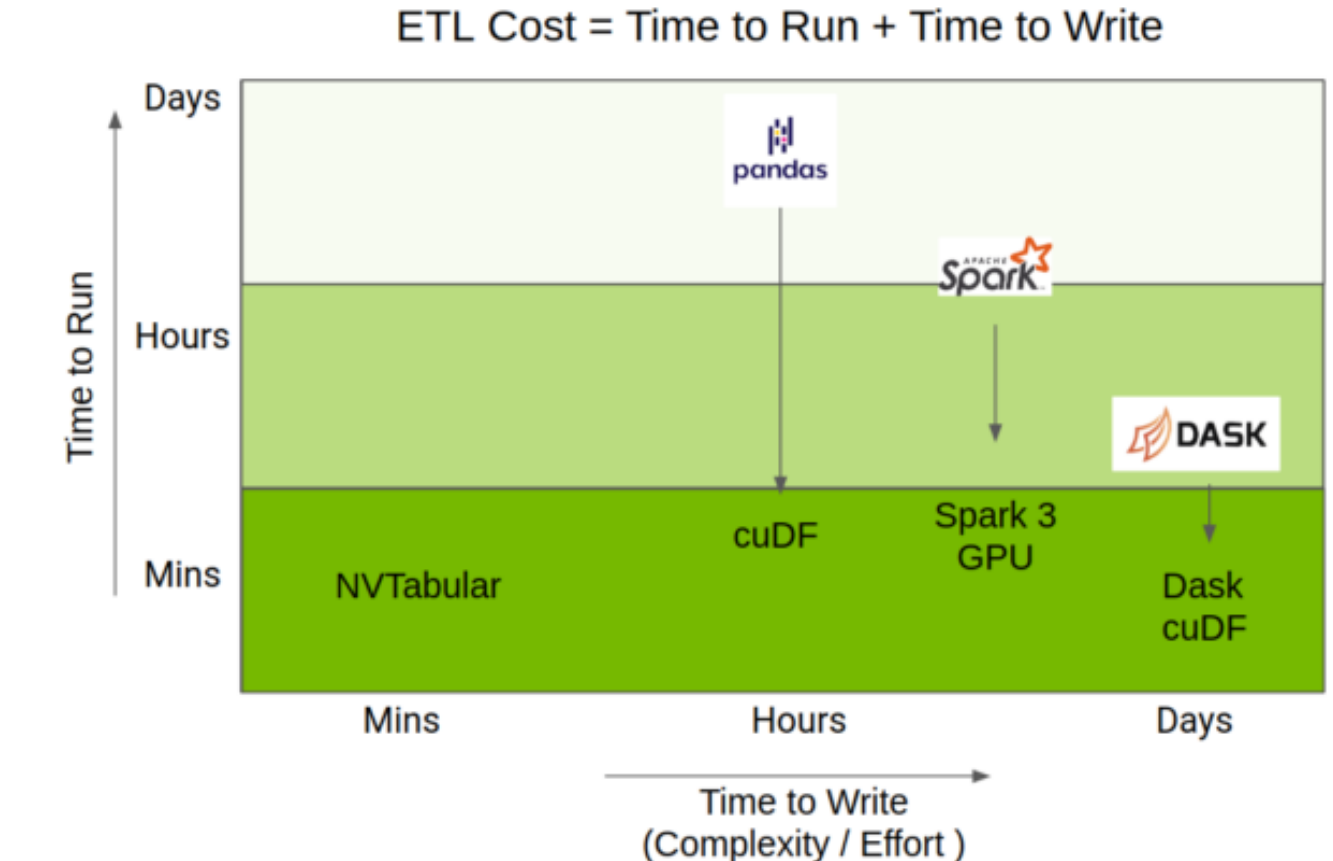
Case Study: 1TB Ads Dataset

ETL 660x faster. Training 96x faster.



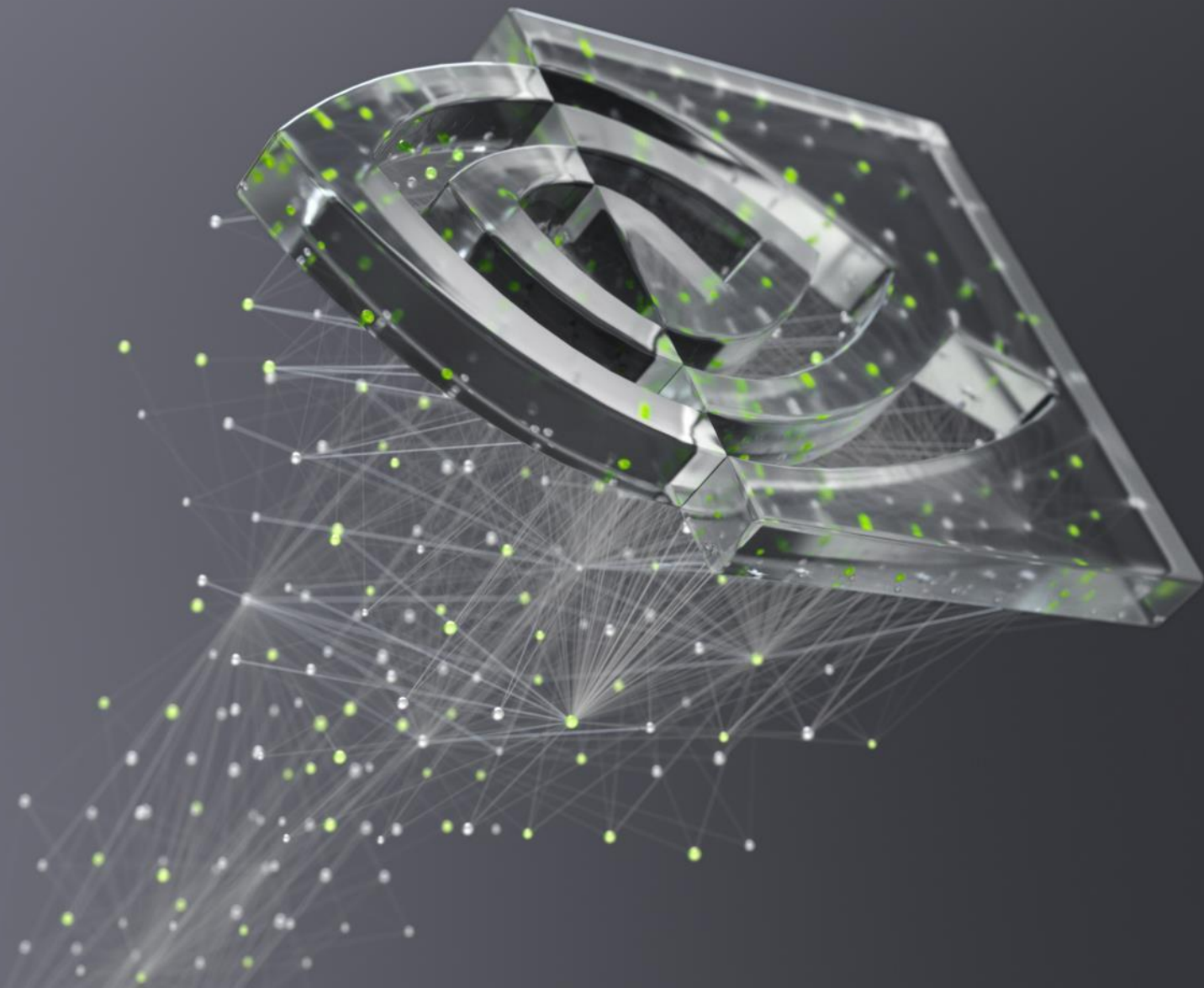
NVTABULAR

Position compared to other popular DataFrame libraries





LET'S GO!



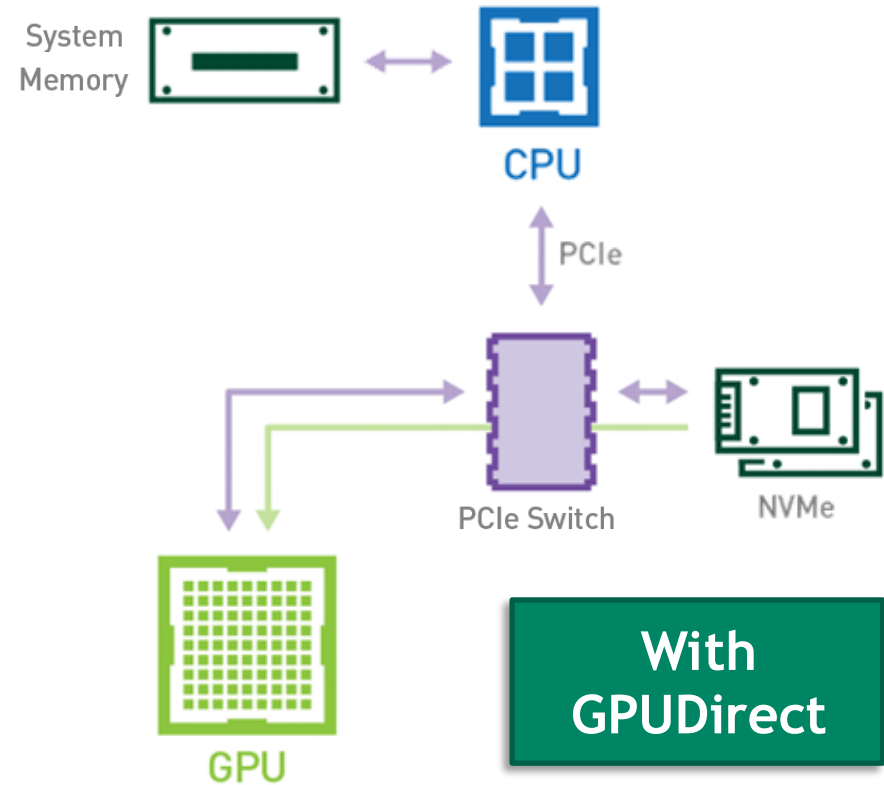
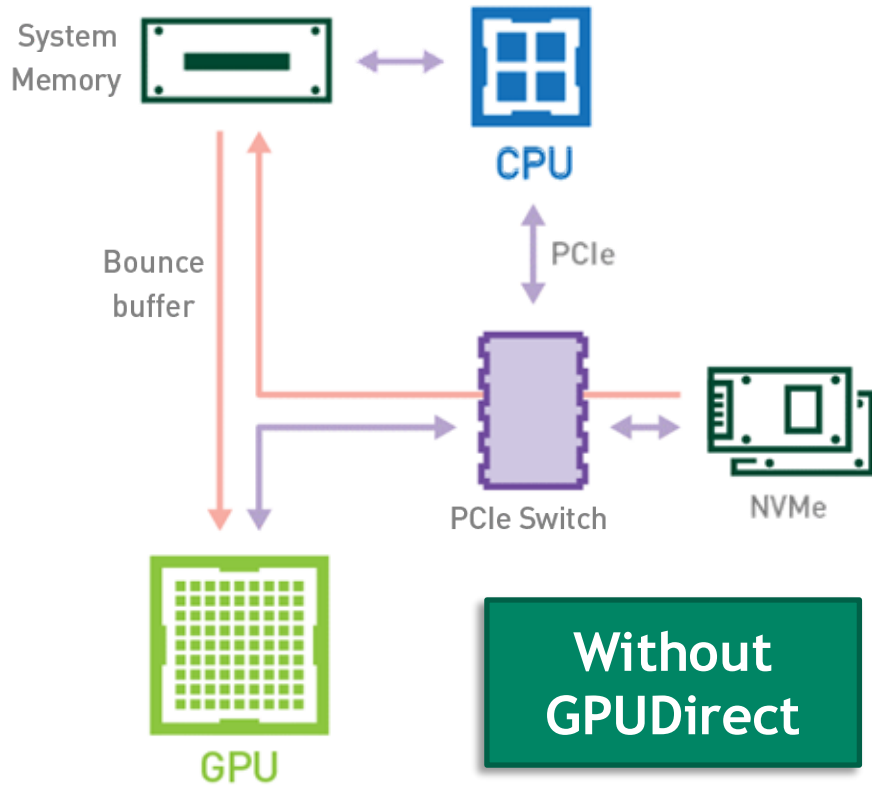
DEEP
LEARNING
INSTITUTE



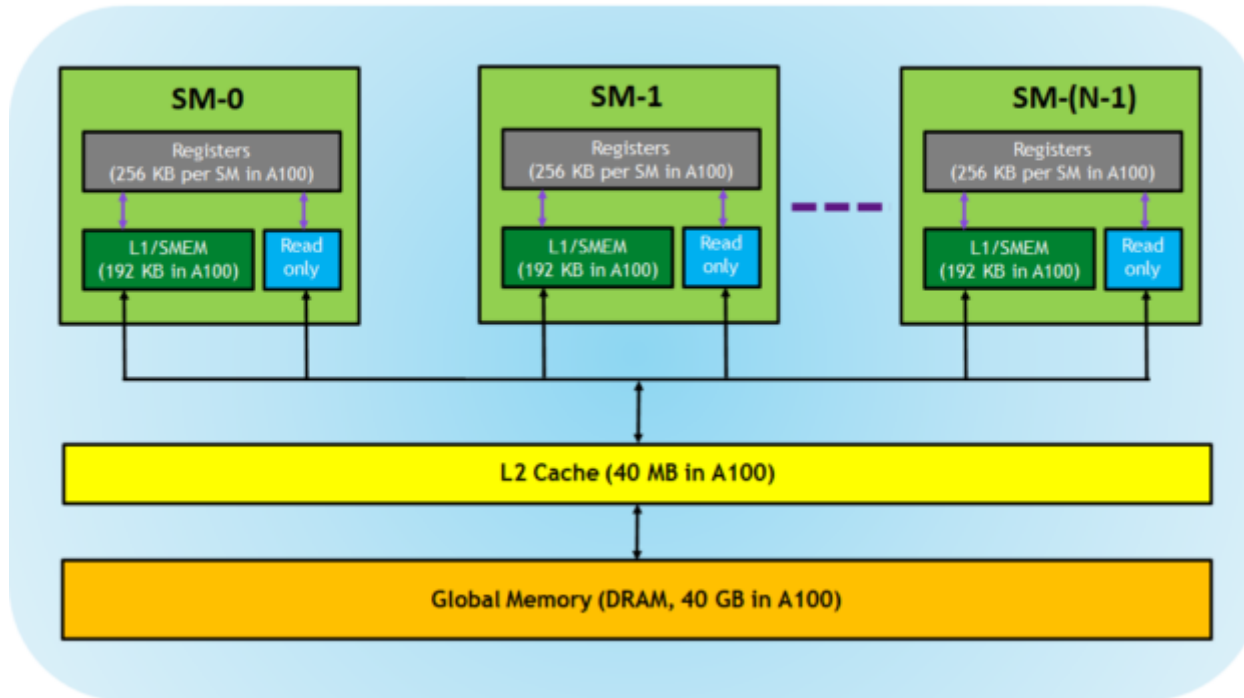
APPENDIX

GPU

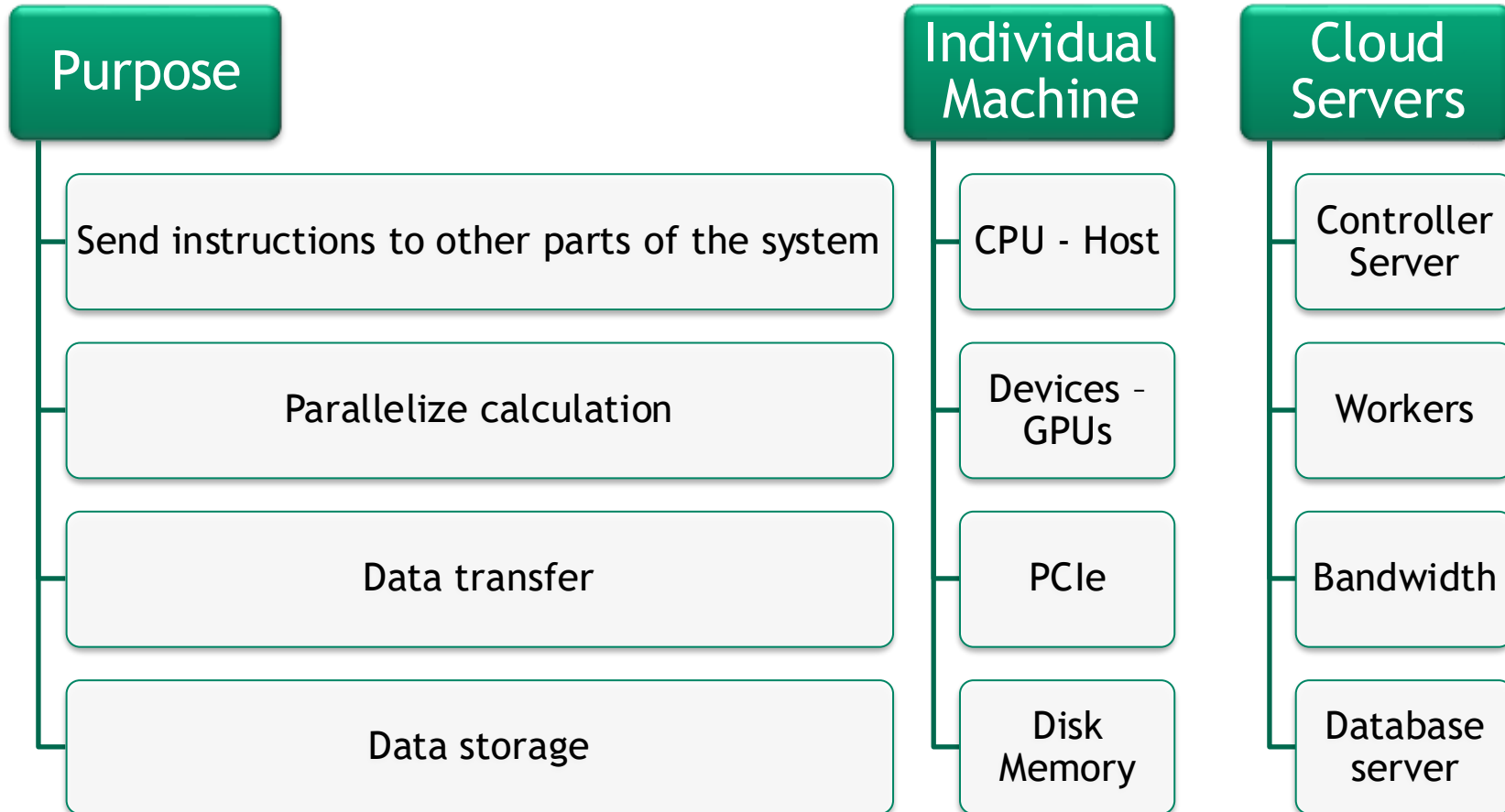
Another Hardware Integration Example



MEMORY HIERARCHY IN GPUS



PARTS OF AN ETL PIPELINE



KERNELS AND PAGEABLE MEMORY

Why they do not work together



Kernel requests pageable data from disk memory



Data does not exist.



Kernel asks page fault handler to fetch data.



Kernel restarts user-defined code.

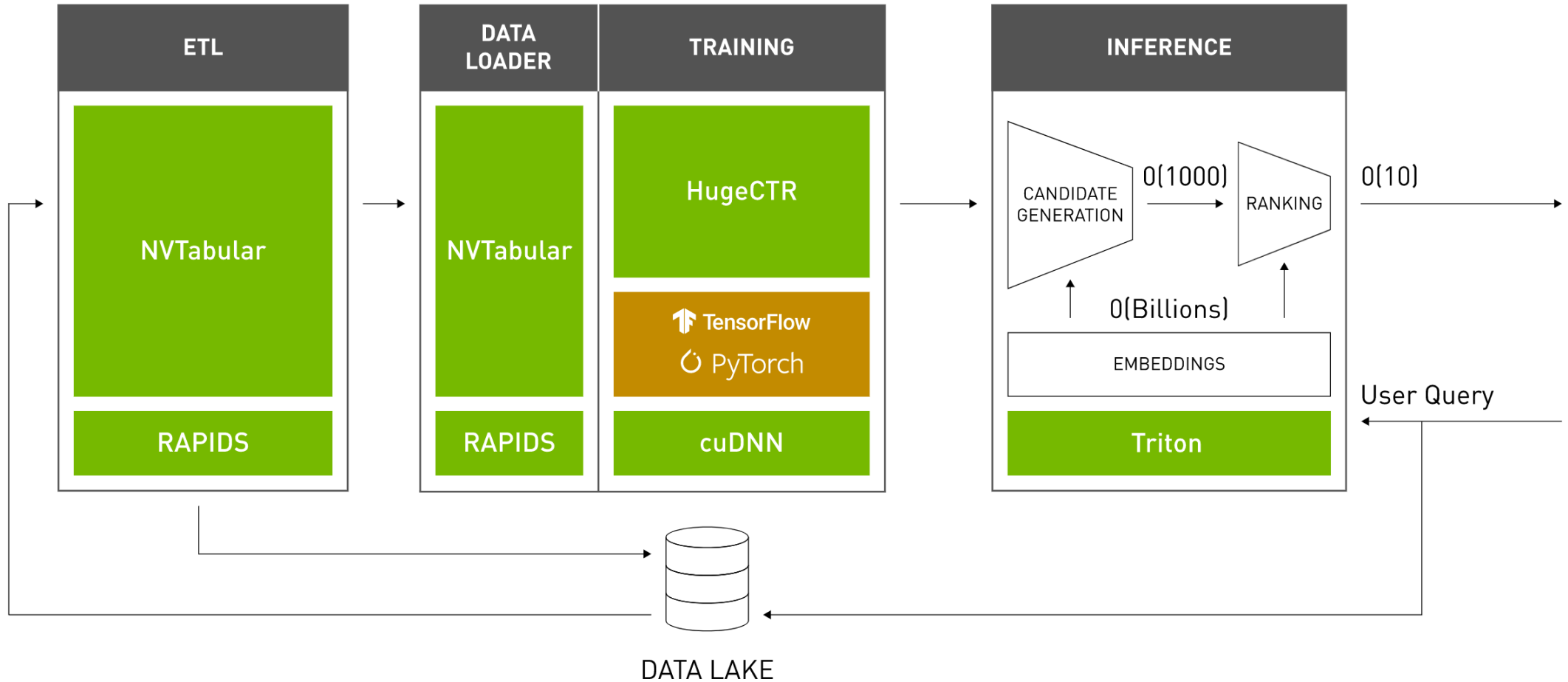


Page fault handler not loaded, fault not handled



Infinite loop.

BIG DATA FOR RETAIL



WORKING WITH NVTABULAR

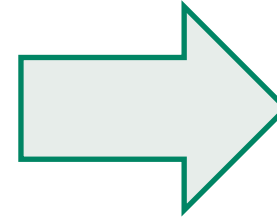
Workflow

Step 1:
Define Column
Group

```
cols = ["col_a", "col_b"]
```

Step 2:
Transform
Columns with Ops

```
cleaned_cols =  
cols >> ops.FillMissing()
```



Step 3:
Define Graph as
Workflow



Step 4:
Run Data through
Workflow

