HPC for ML
Magurele Summer School

GPU Accelerated Distributed Data Science

Marco Celoria – CINECA

Magurele 9 July 2025

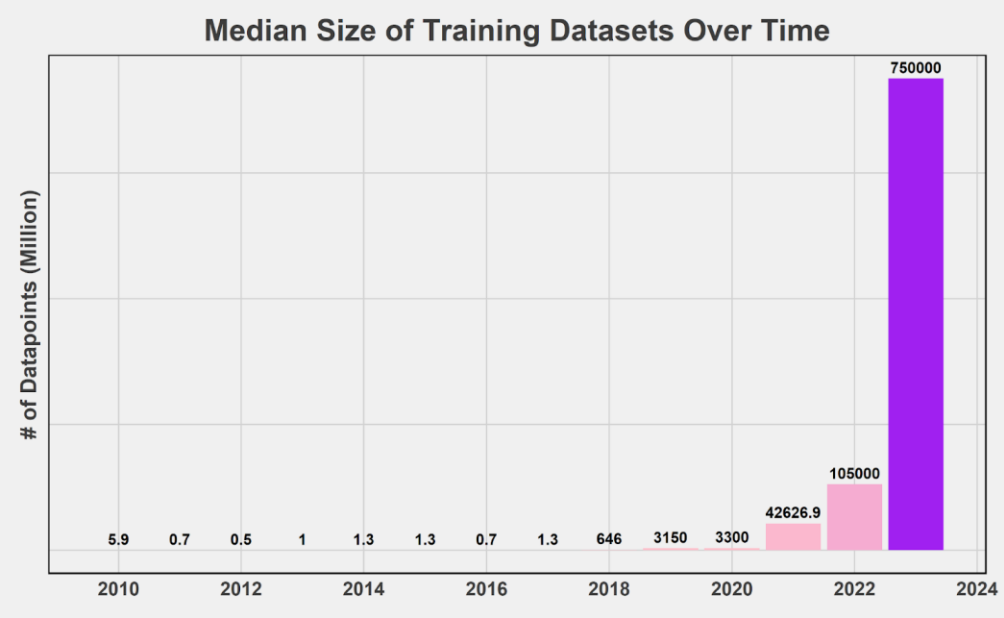
Machine Learning

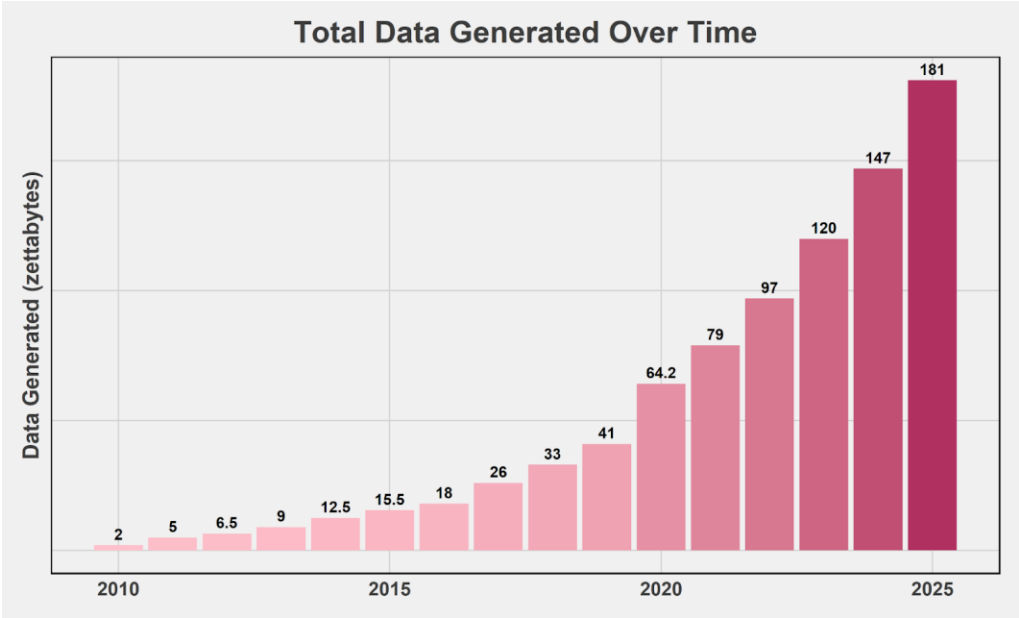
Machine Learning is building model from data using optimization.

High-dimensional non-convex optimization based on a growing wealth of data

- Larger and larger training sets
- GPUs used to speed up training times on such larger training sets
- New learning algorithms optimize convergence and generalization

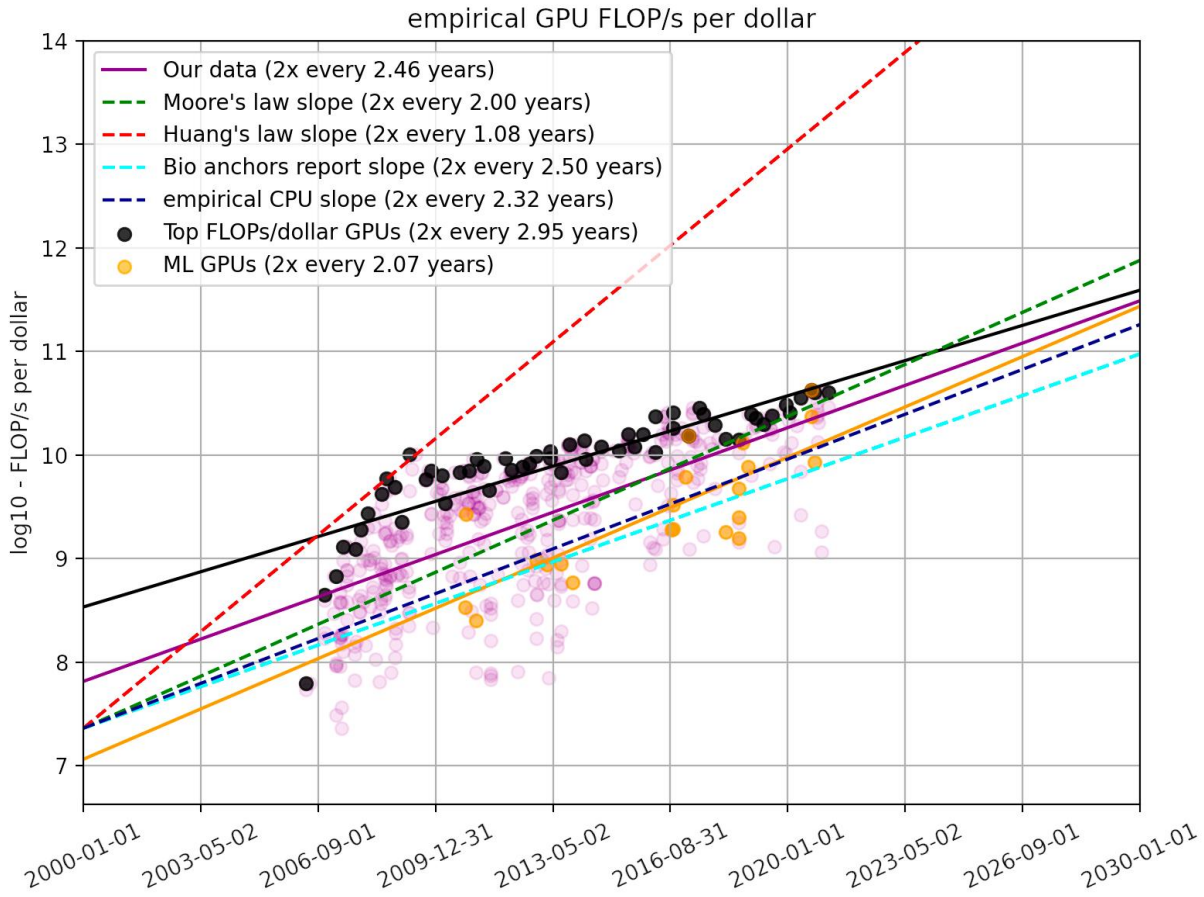
Machine Learning - Data





https://futuretech.mit.edu/news/what-drives-progress-in-ai-trends-in-data

Machine Learning - GPUs



Marius Hobbhahn and Tamay Besiroglu, "Trends in GPU Price-Performance".

Machine Learning - Algorithms

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

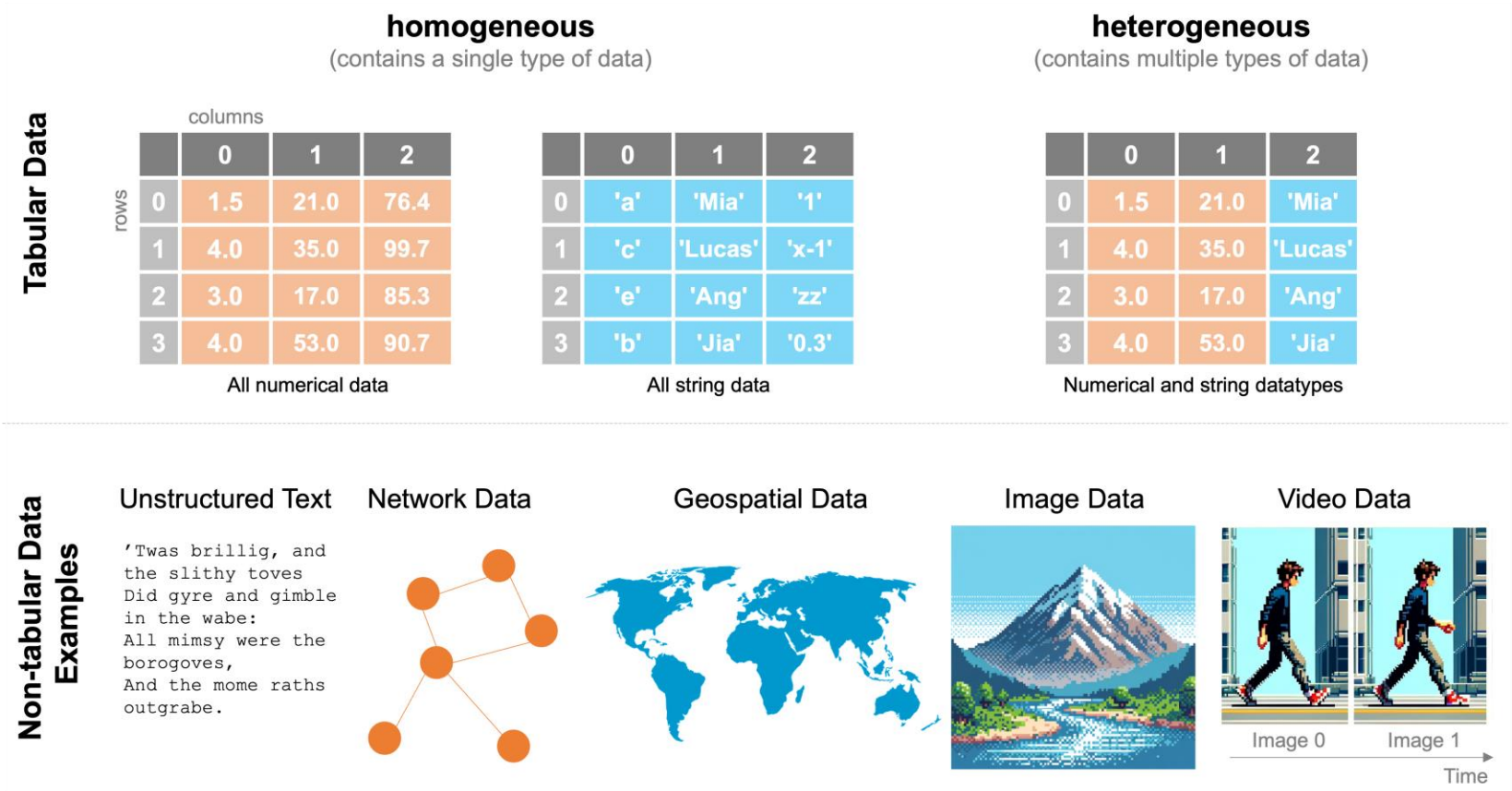
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations

M. Raissi [a], P. Perdikaris [b], G.E. Karniadakis [a]

Type of data

- Structured tabular data:
  - Data tables including discrete data types (numbers,short text,dates)
    Each column has an attribute, and each row is a single record with associated data values for each attribute.
    RandomForest and XGBoost excel for structured data
- Unstructured data:
  - Data that doesn’t fit into a data table because its size or nature.
    Unstructured data doesn’t follow any predetermined rules.
    Neural networks shine in tasks involving image, text, and speech data, which are difficult for traditional models like XGBoost.

Type of data

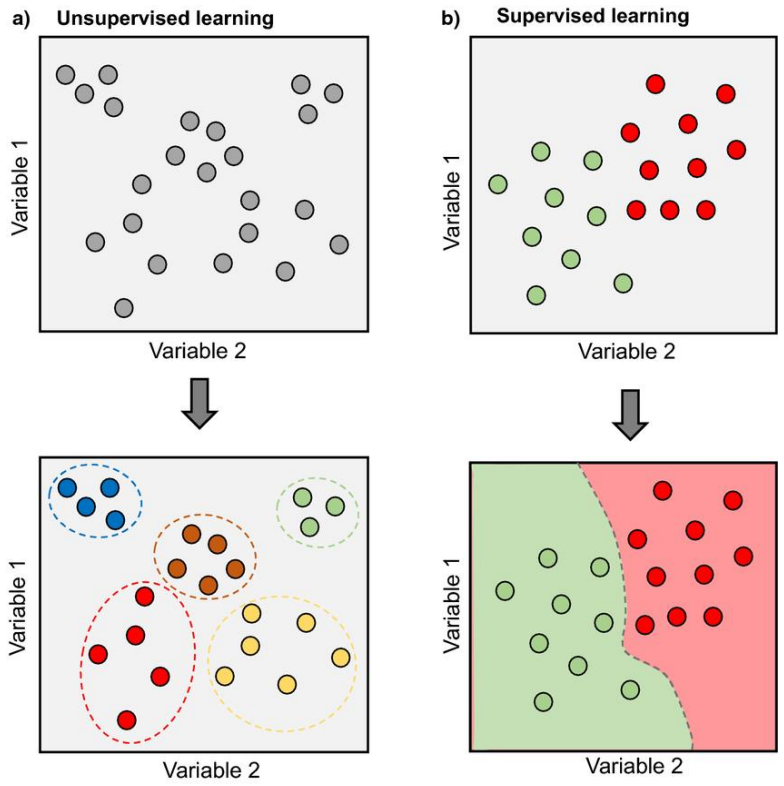


https://www.practicaldatascience.org/notebooks/class_3/week_2/00_intro_to_pandas.html

Structured tabular data and ML

- Typically, we deal with structured data starting from csv files, using pandas dataframes to load the datset into memory, and training ML algorithms using scikit-learn or XGBoost.
  - Unsupervised ML algorithms like PCA, clustering, … intend to infer an a priori probability distribution that has generated the data.
  - Supervised ML algorithms like decision trees, k-NN, neural networks, … intend to infer a conditional probability distribution conditioned on the label of input data.

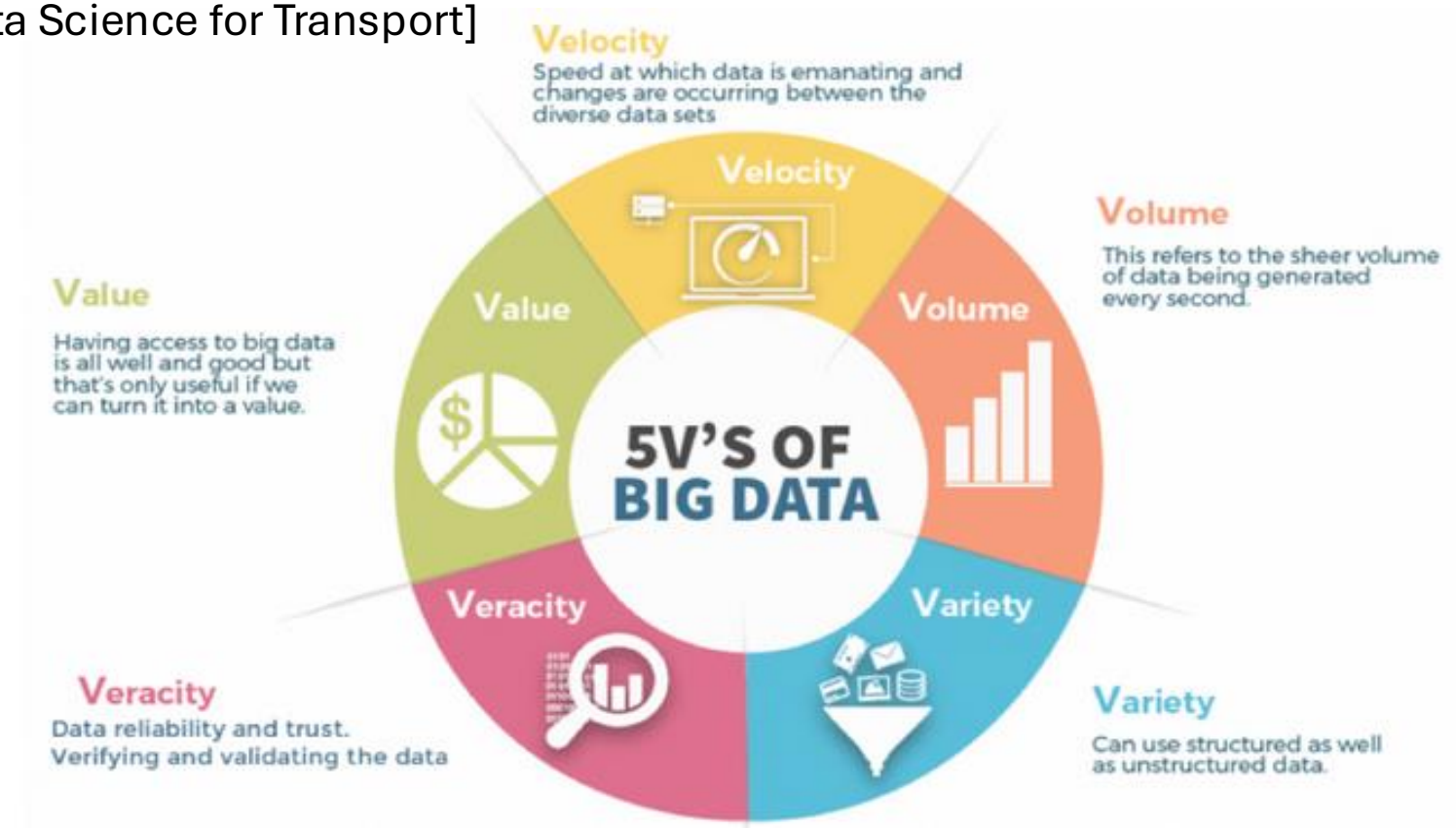
Supervised and unsupervised learning



Morimoto, Juliano & Ponton, Fleur. (2021). Virtual reality in biology: could we become virtual naturalists?

Big Data

- A 2018 definition states "Big data is where parallel computing tools are needed to handle data" [Fox, Data Science for Transport]



https://www.devopsschool.com/blog/what-is-big-data-processing-tools-and-use-cases-of-big-data-processing-tools/

Hardware limitations

- On Leonardo, we have Lustre (open-source parallel filesystem) and the High Performance Storage is 5.7 PB (DDN Exascaler ES400NVX2)
- RAM of the compute node is 512 GiB

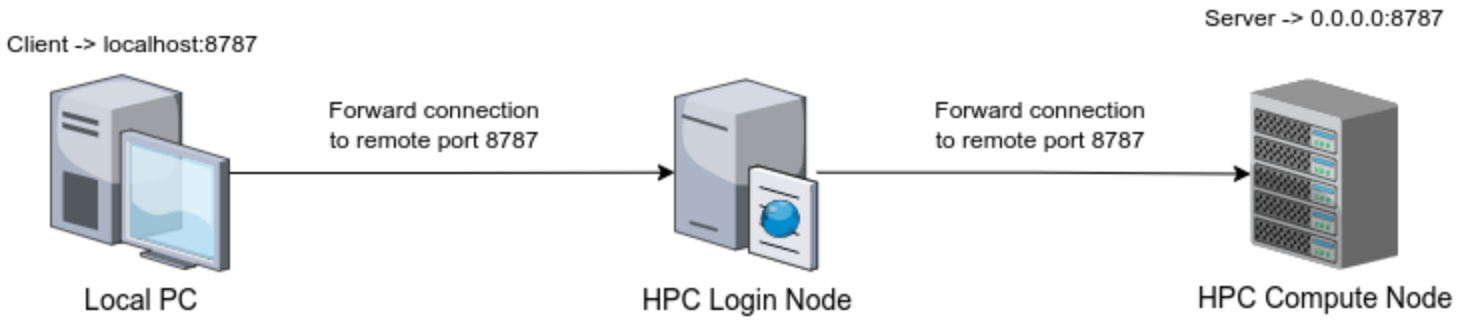
- Big data cannot fit in the memory of a computational node
- Most Pandas operations run on a single thread, while some scikit-learn estimators and utilities are parallelized using multi-cores CPU

Feature of Big Data Processing Tools

- **Distributed Computing:** Big Data Processing Tools distribute data and computation across multiple nodes or clusters to achieve scalability.
- **Fault Tolerance:** They provide mechanisms to handle node failures and ensure data integrity.
- **Parallel Processing:** Tools leverage parallel processing techniques to speed up data processing tasks.
- **Data Replication:** Data replication ensures data availability and resilience.
- **Data Partitioning:** Tools partition data for efficient processing and distribution across the cluster.
- **Data Compression:** Compression techniques reduce storage and transmission overhead.

HPC infrastructure

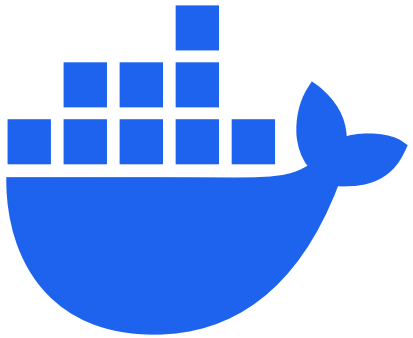
- Login nodes are the only nodes accessible from external networks.
- On Leonardo no connections from the computational nodes to the outside world are permitted.
- To use interactive tools like Jupyter Notebook using the client-server paradigm in HPC compute nodes:



- Finally, we typically use SLURM as the job scheduler

Coud infrastructure

- By containerizing the model with Docker and orchestrating it with Kubernetes, we can ensure consistency, scalability, and reliability
- The key principles in modern MLOps workflows.
- Kubernetes clusters can deploy, manage, and scale AI and ML:
  - Infrastructure orchestration that supports GPUs for training at scale.
  - Flexible integration with distributed computing and data processing



Distributed DataFrames

- The idea is to use the memory of several computational nodes that share the filesystem and communicate via the network
- In addition, we parallelize the work on several multicore nodes
- The most popular frameworks for distributed computing are
  - Apache Spark
  - Ray
  - Dask

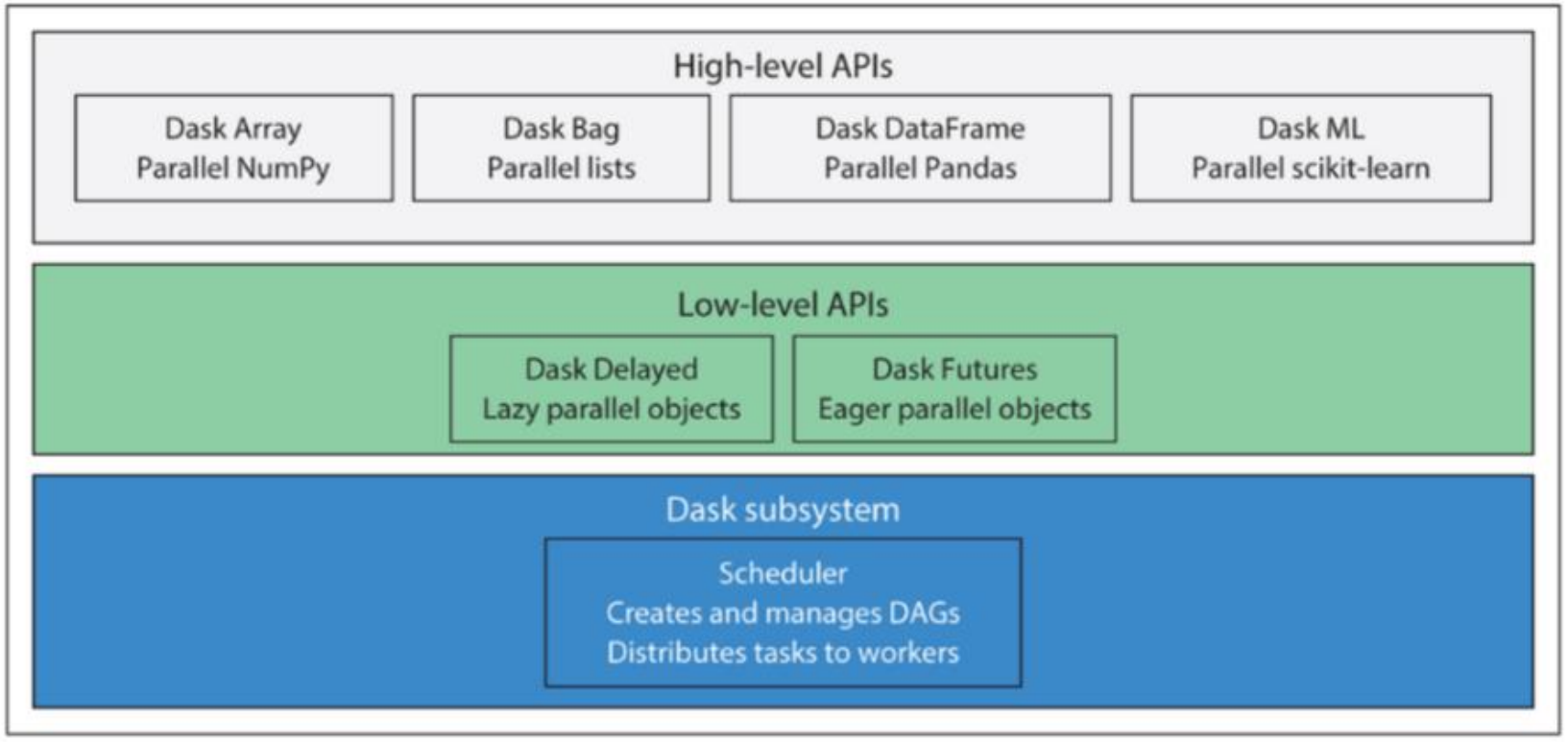
What is Dask?

- Dask provides multi-core and distributed+parallel execution on larger-than-memory datasets
- Dask allow to work with larger datasets making it possible to parallelize computation
- Dask is in Python and scales NumPy, Pandas, and scikit-learn.
- Dask is a framework for parallelizing most Python objects.

When Dask?

- Dask is not helpul for small size datasets as it generates overhead
- Dask might be useful for medium size dataset as it allows to work in parallel on a local multi-core (or multi-GPU) machine.
- Dask is essential for large datasets as Pandas, NumPy, and scikit-learn are not inherently built to operate on distributed datasets.

Dask APIs



Adapted from Data Science with Dask

Dask overview

Collections

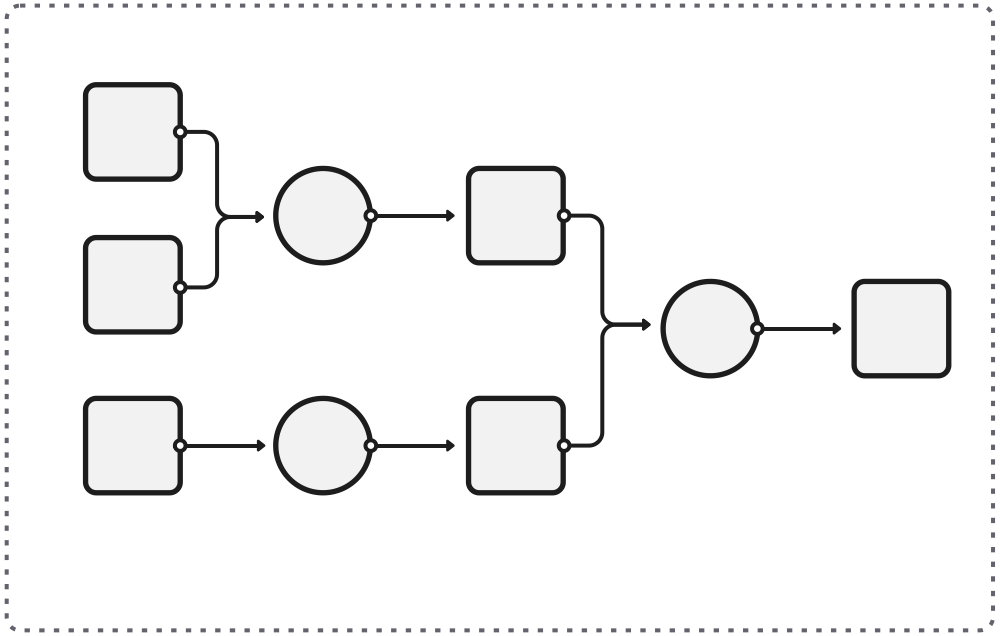
(create task graphs)

Task Graph

Schedulers

(execute task graphs)

- Dask Array
- Dask DataFrame
- Dask Bag
- Dask Delayed
- Futures

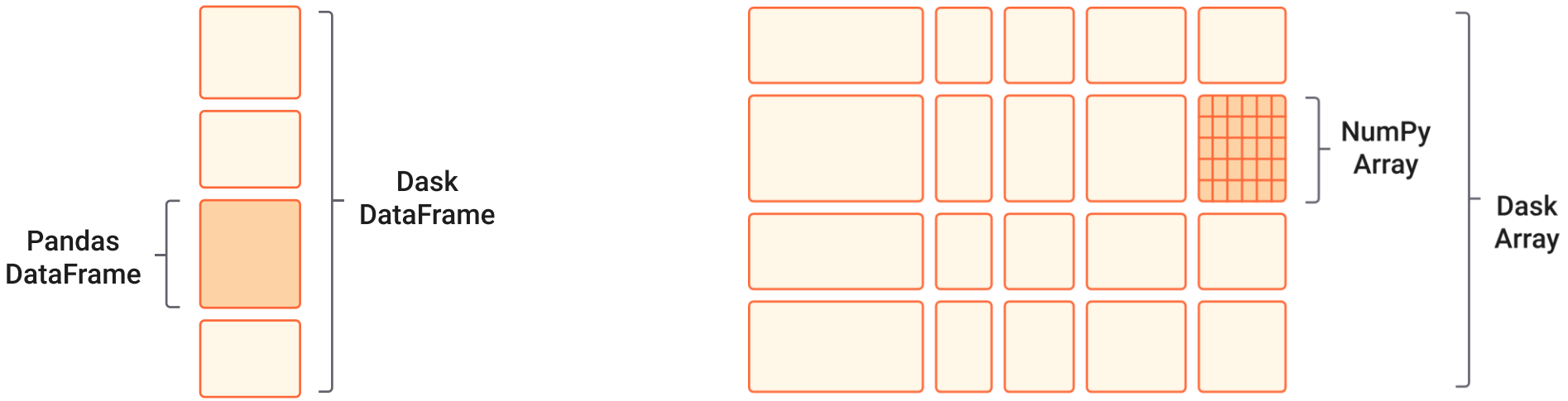


- Single-machine (threads, processes, synchronous)
- Distributed

High level collections are used to generate task graphs which can be executed by schedulers

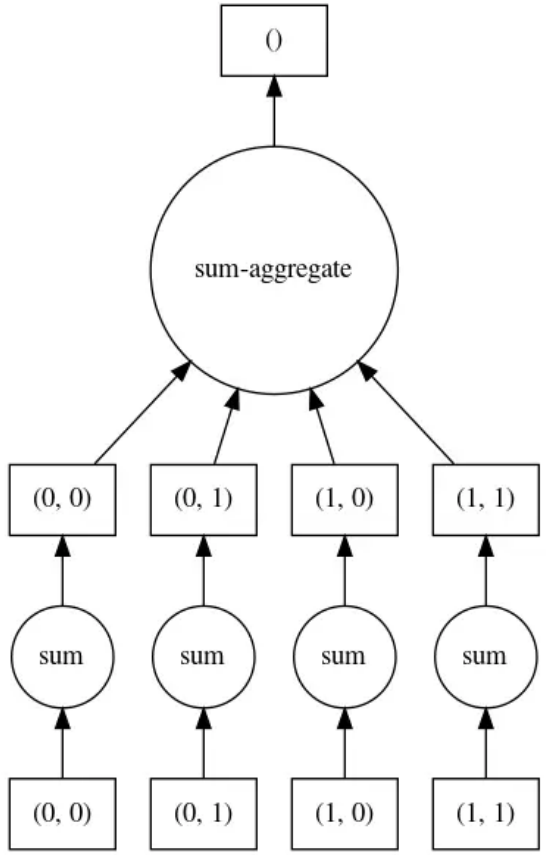
Dask Collections

- Dask can be used to scale popular Python libraries such as Pandas and NumPy allowing to analyze dataset with greater size



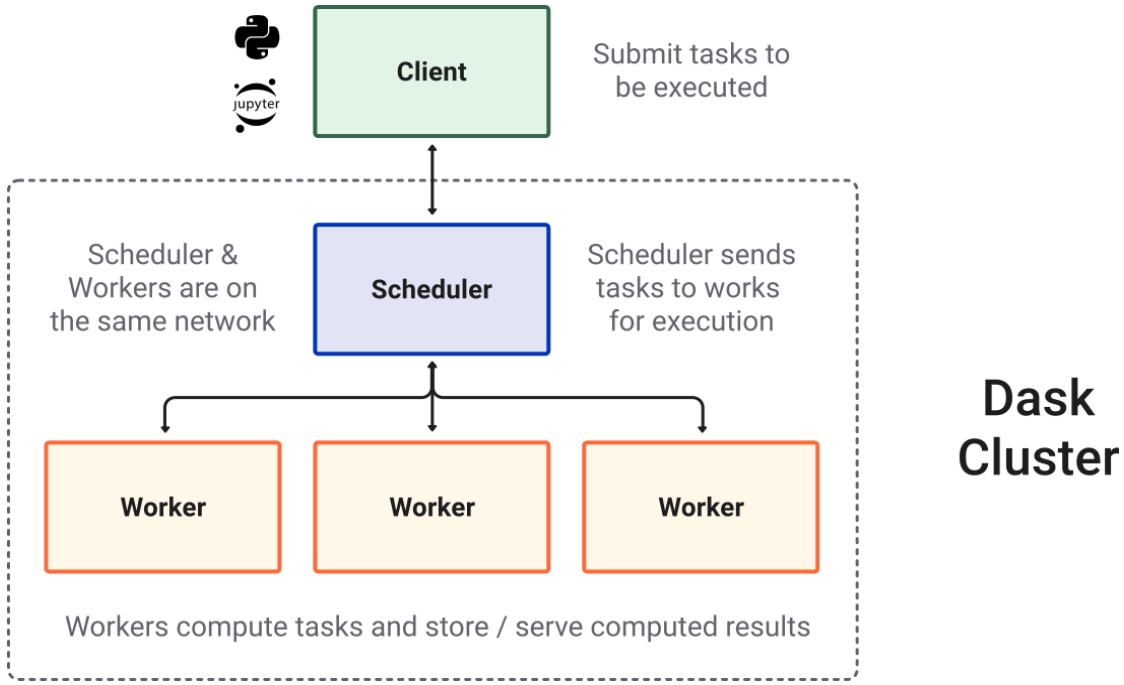
Task Directed Acyclical Graph (DAG)

- A graph is a representation of a set of objects that have a relationship with one another, consisted by:
  - node: a function, an object or an action
  - line: symbolize the relationship among nodes
- In DAG there is one logical way to traverse the graph. No node is visited twice.
- Dask uses DAG to coordinate execution of parallelized code across processors.
- Upstream actions are completed before downstream nodes.



Dask Cluster

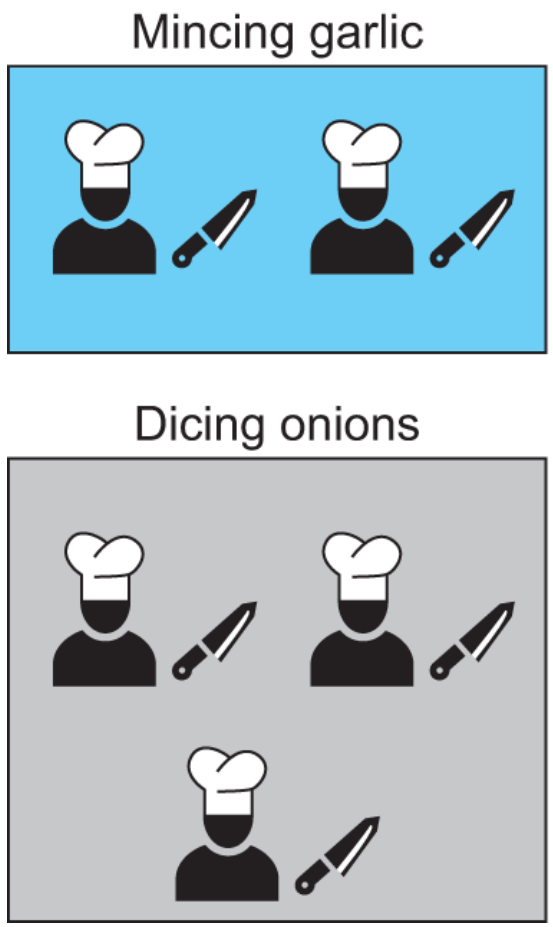
- Usually, when using Dask, we will be using a distributed scheduler, which exists in the context of a Dask cluster.

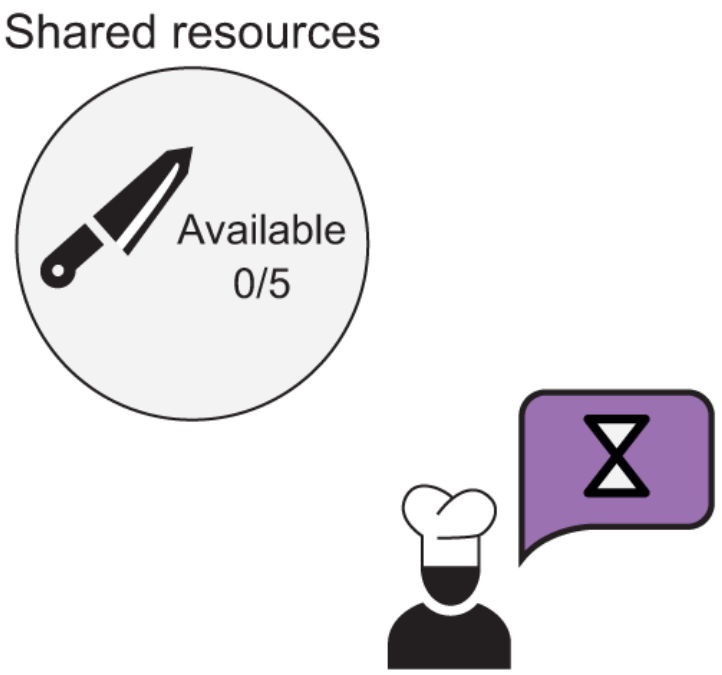


Dask Scheduler and Workers

- Dask uses a central scheduler to orchestrate the work.
- Adding workers can improve performances of complex workloads, however, create overhead that can reduces gains.
- Workers will unlikely be perfectly balanced with respect to load, power and data access.
- Some resources might be not fully exploited or might be idling because of insufficient shared resources (resource starvation).
- Due to these conditions, scheduler needs to promptly react to avoid bottlenecks that will affect overall runtime.

Resources





This cook must wait and remain idle until either a knife becomes available or a new task that doesn't require a knife is available. This is an example of a resource-starved worker.

Adapted from
Data Science with Dask

The importance of being lazy

- Dask performs a lazy computation
- Until we run the method .compute(), Dask only splits the process into smaller logical pieces
- Even though the process is defined, the resources assigned and the place where the result will be stored are not assigned
- The scheduler assigns them dynamically, allowing to recover from possible worker failure
- In case of a failure, Dask reaches a node and repeat the action without disturbing the rest of the process.

We want more FLOPS/s!

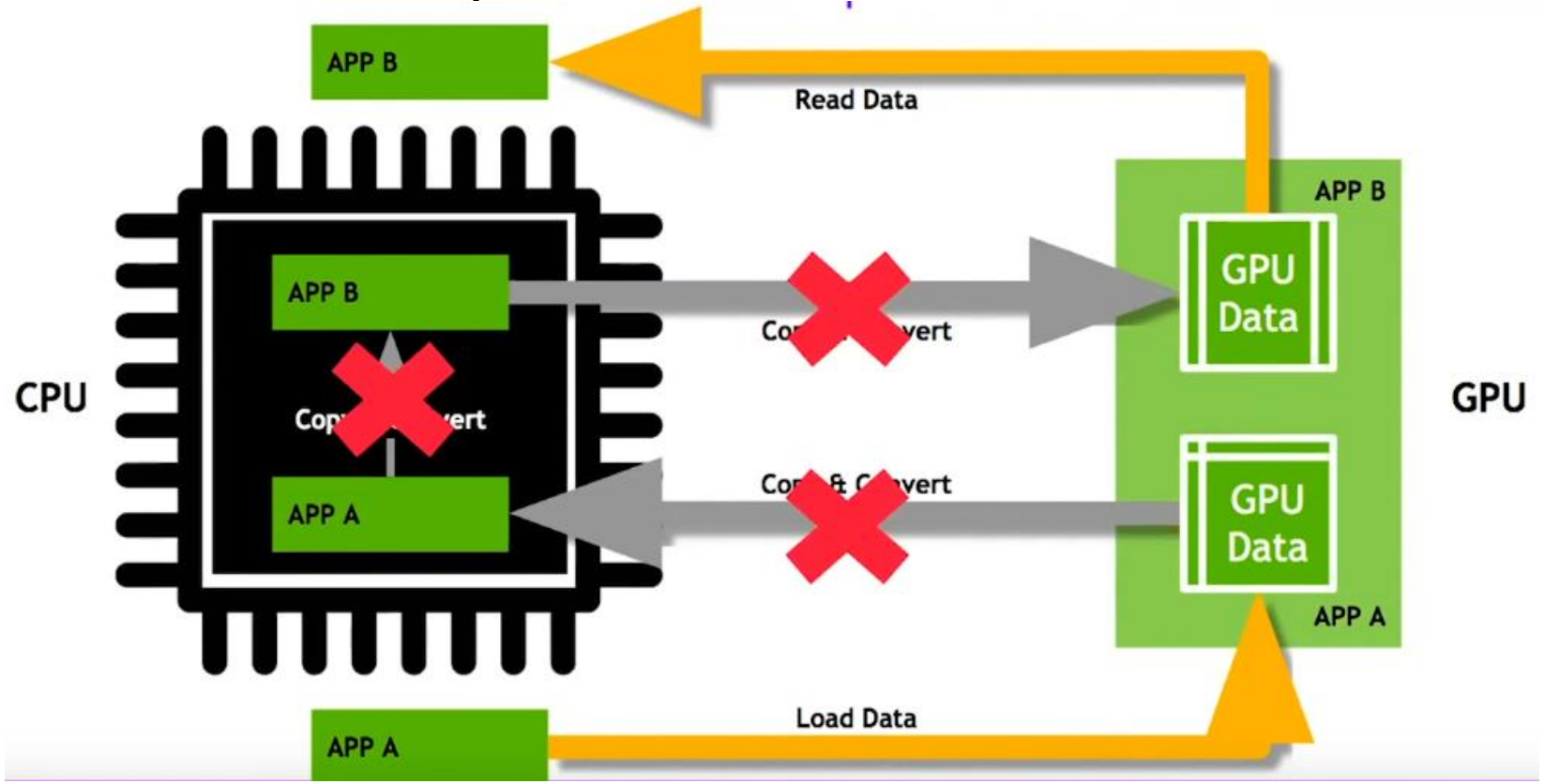
- CPUs are designed for more general-purpose tasks and have fewer, more powerful cores that excel at low-latency operations.
- GPUs are designed for highly parallel computations, with many-cores optimized for executing the same instructions on many data elements simultaneously (SIMT).

Source: NVIDIA

Data Movement and Transformation

- What if we could keep data on the GPU?

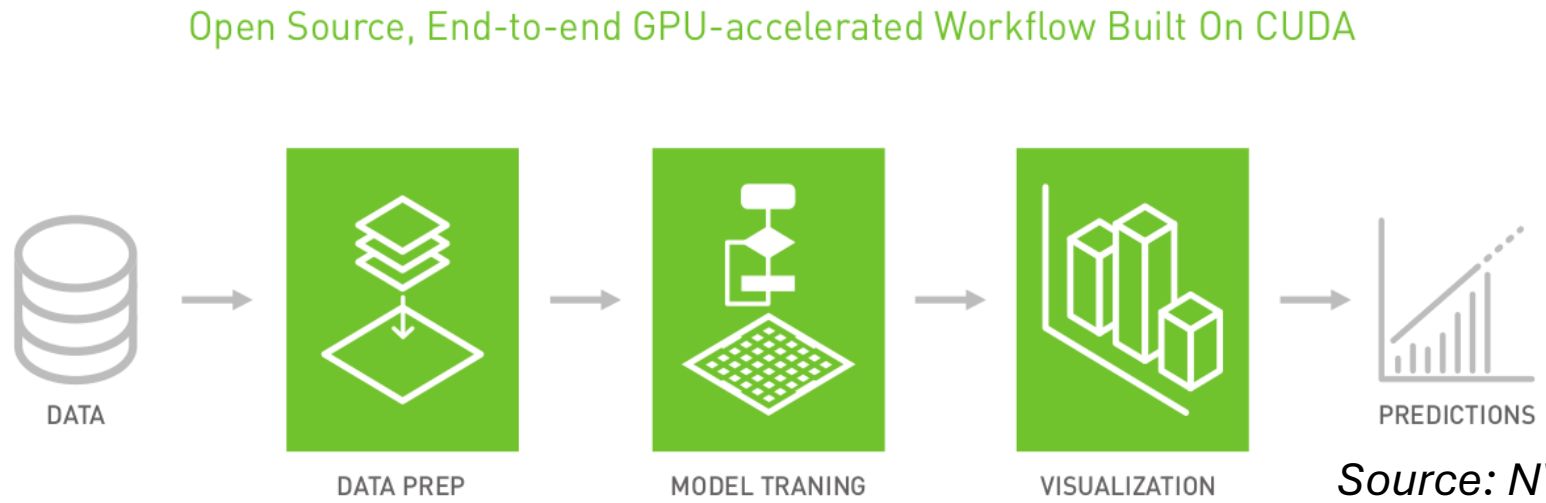
Source: NVIDIA RAPIDS



RAPIDS

RAPIDS by NVIDIA is a suite of open-source software libraries and APIs designed to accelerate all data science

- Data analysis, modelization and visualization entirely on GPUs
- Accelerated machine learning (K-means, PCA, XGBoost, …)



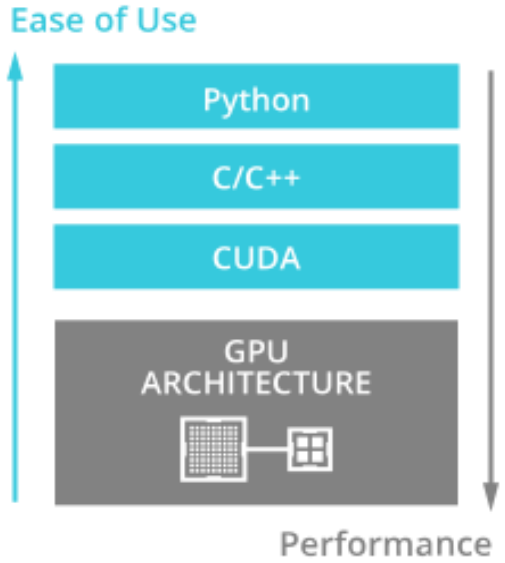
Source: NVIDIA RAPIDS

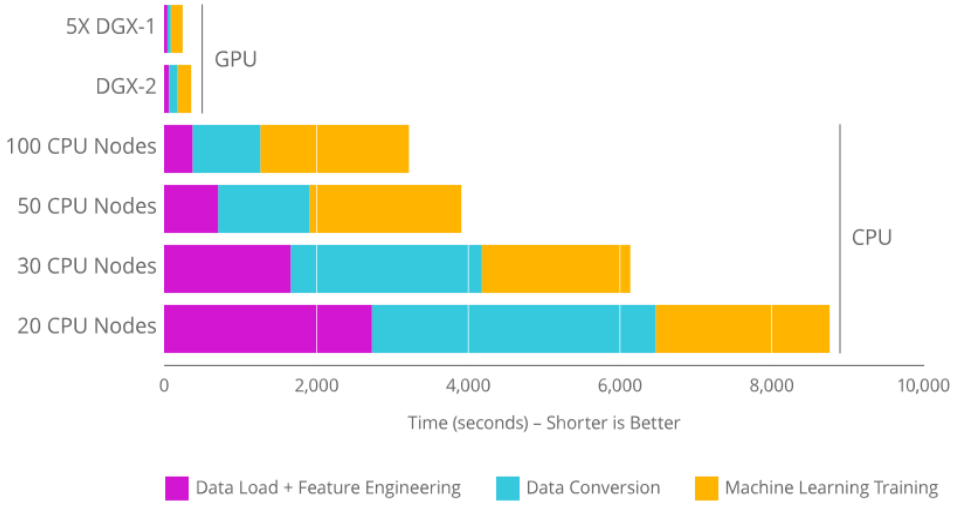
Why RAPIDS?

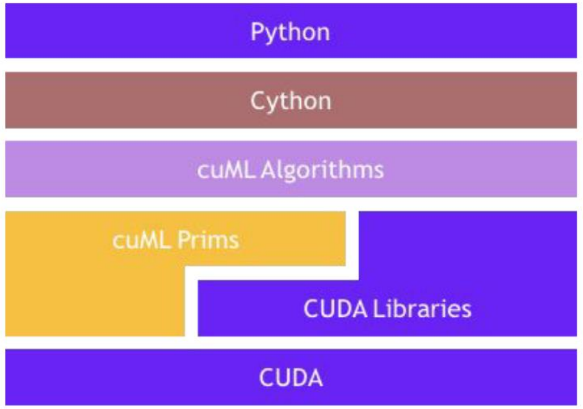
- **Speed**: RAPIDS leverages the power of GPUs to process data, often delivering performance gains of 10x or more compared to traditional CPU-based methods.
- **Familiarity**: Data scientists can continue using familiar tools and workflows, with RAPIDS offering drop-in replacements for many common Python data science libraries.
- **Ecosystem**: RAPIDS integrates with popular data science frameworks like Apache Spark and Dask, making it easy to incorporate GPU acceleration into your existing infrastructure.
- **Scale**: RAPIDS scales seamlessly to handle massive datasets, enabling you to tackle big data problems that were previously impractical or computationally expensive.

RAPIDS is easy and fast

- RAPIDS provides a similar interface to Python libraries like pandas, scikit-learn and NetworkX
- RAPIDS becomes advantageous, when the data being manipulated reaches multiple GiBs

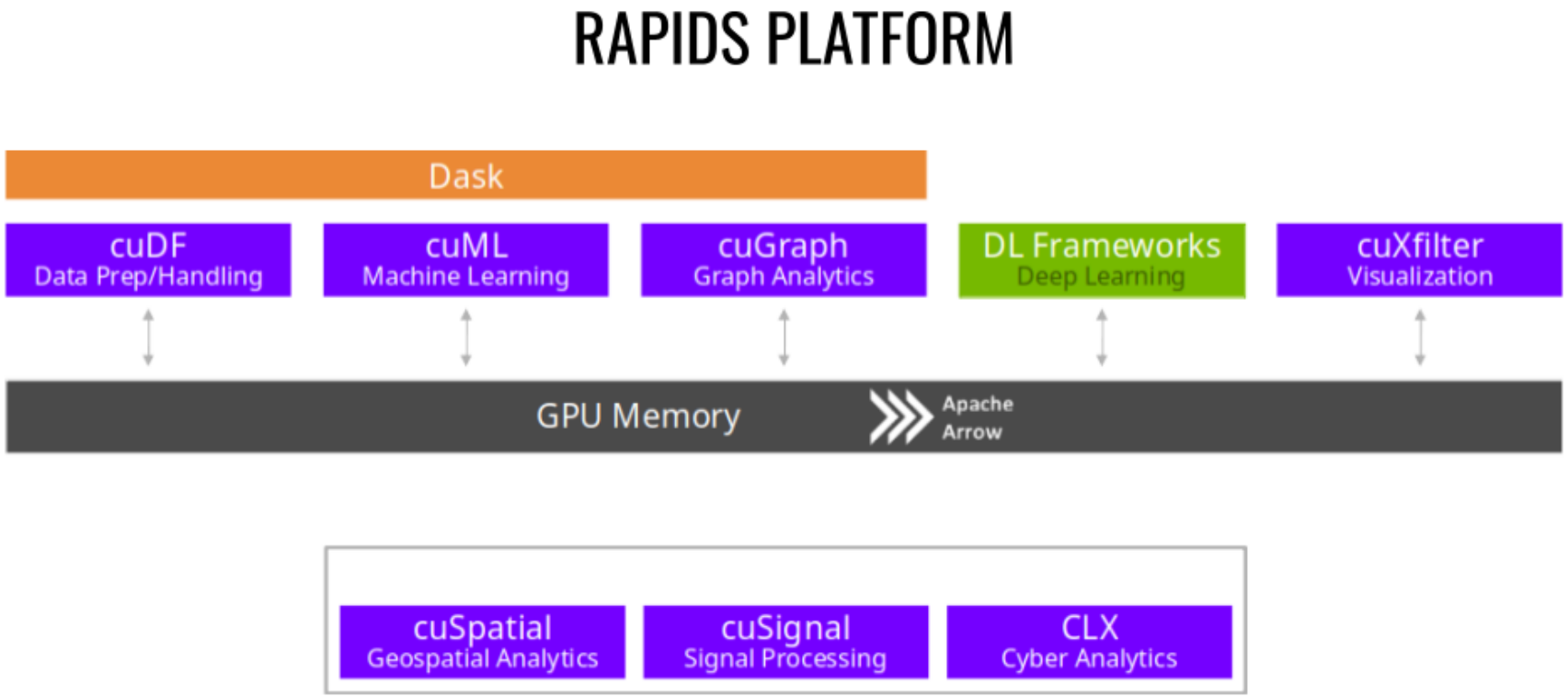






Source:
Documentation RAPIDS

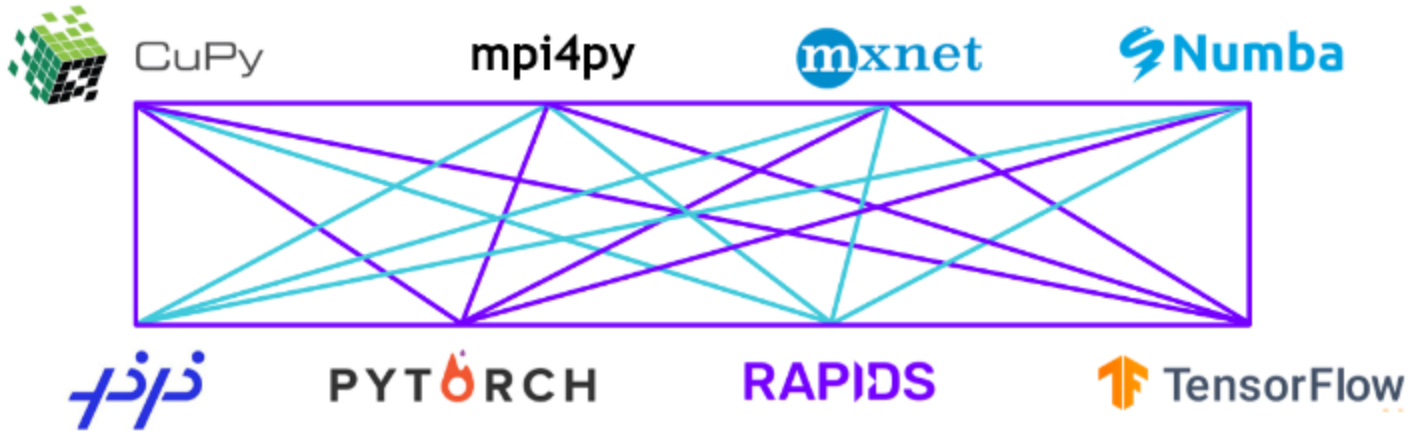
RAPIDS Platform



Source: Documentation RAPIDS

RAPIDS Interoperability

- Real-world workflows often need to share data between libraries
- RAPIDS supports device memory sharing between data science and deep learning libraries
- Keeps data on the GPU avoids costly copies to host memory
- Any library that support __cuda_array_interface__ will allow for sharing of memory buffers with RAPIDS



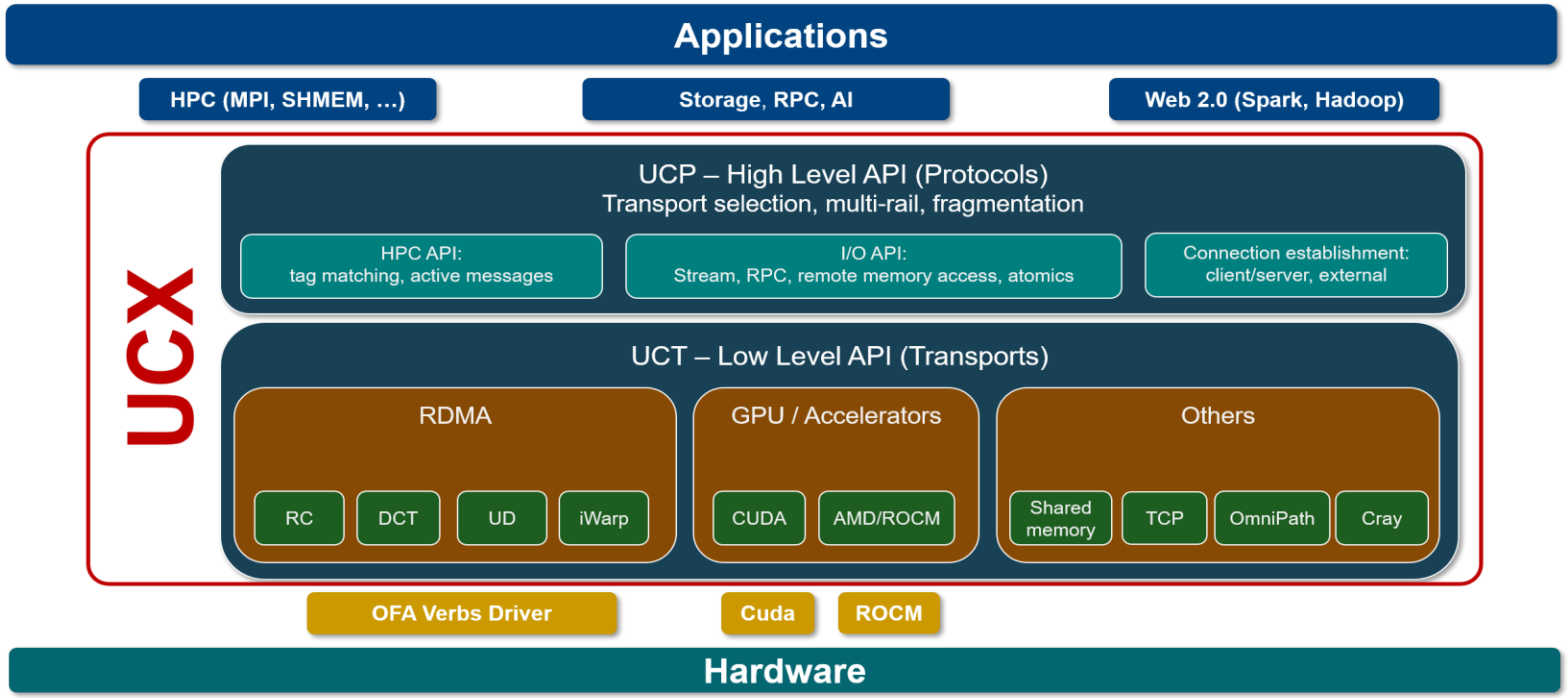
Dask-CUDA

- Dask allows users to create a cluster with scheduler and workers
- Each Dask-CUDA worker is attached to a dedicated GPU
- Dask client can connect to the cluster and distribute data on it
- Once the data has been distributed to the workers,
  the workers can start the distributed operations
- Dask-CUDA is an addition to Dask that allows the creation of GPU clusters while optionally configuring networking via the high-performance UCX-Py

Unified Communication X

UCX is an optimized communication framework for high-bandwidth and low-latency networks

UCX exposes abstract communication primitives that utilize the best of available hardware resources and offload, including RDMA, TCP, GPUs, shared memory, and atomics operations



Install RAPIDS on Leonardo

- **pip Issues:** Infiniband is not supported yet.
- **Docker Issues:** On Leonardo, we use Apptainer and not Docker
- **Conda Issues:**
  If the Conda is older than `22.11`, update to the latest version.
  This will include libmamba, a Mamba-powered Conda solver to significantly accelerate environment solving.
  If the Conda installation is version `22.11` or newer, run:
  ```
  $> conda install -n base conda-libmamba-solver
  $> conda create --solver=libmamba ...
  ```
- RAPIDS can be used with any conda distribution e.g. miniforge
- Also available as module: module load rapids/2023.12

Failure is an option

- In case of a failure, Dask reaches a node and repeat the action without disturbing the rest of the process.
- There are two types of failures:
    - work failures: a worker might fail, and we know that we must assign another one to their task. This can slow down the execution. However, it won't affect previous work as data have not been lost.
    - data loss: accident happens, and we have to start from the beginning. The scheduler stops and restarts from the beginning the whole process.

RAPIDS RAFT

RAFT contains fundamental widely-used CUDA-accelerated algorithms and primitives for ML and information retrieval forming building blocks for more easily writing high-performance applications