Faculty of Engineering and Technology

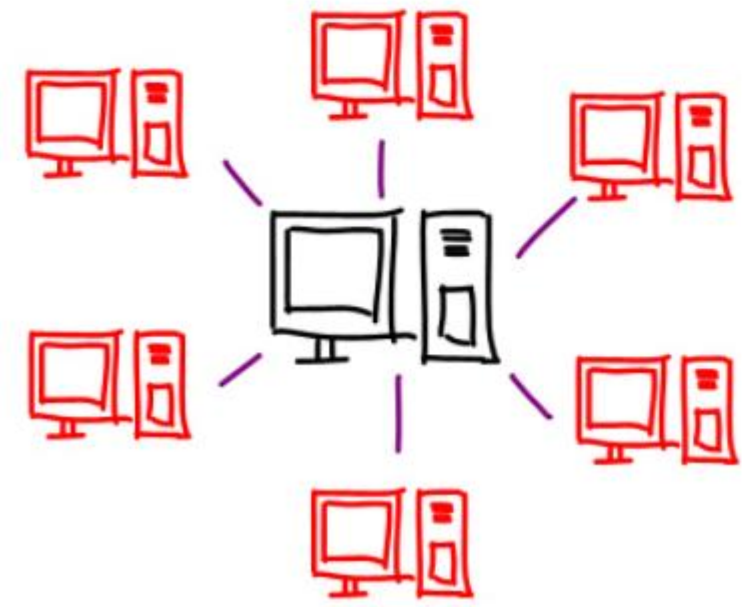
School of Computer Science and Mathematics

7145COMP Accelerated Machine Learning

Dr Paul Fergus, Dr Carl Chalmers

{p.fergus, c.chalmers}@ljmu.ac.uk

Room 714, 629 Byrom Street

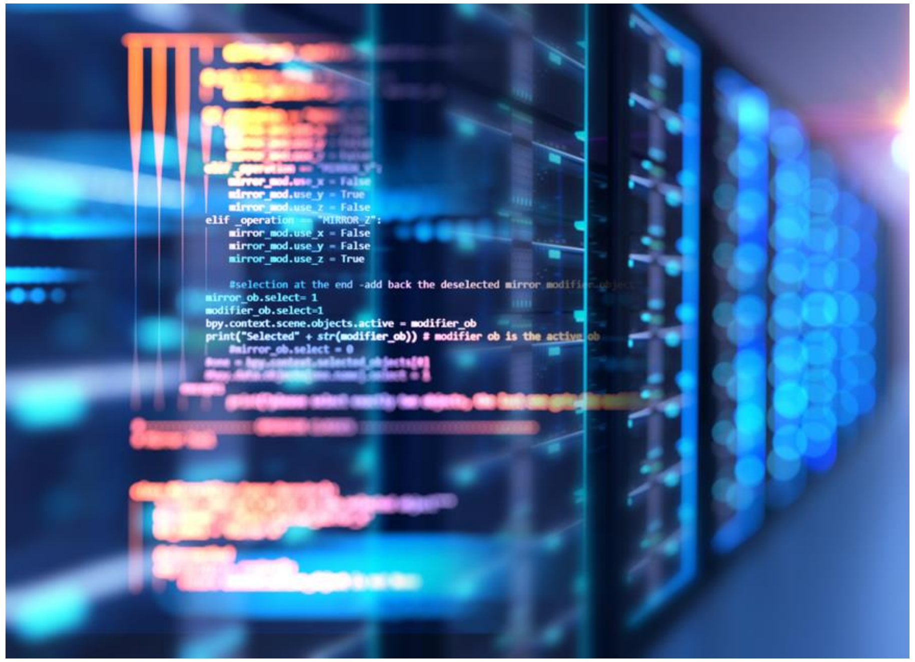


Lecture 1

Module Overview and Introduction

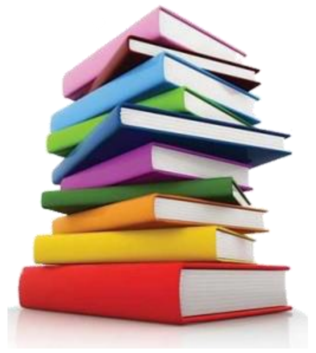
In this session…

- We will cover:
  - Module Overview
    - Aims & Assessment
    - Learning Activities & Learning Strategy
    - Achieving Success
    - Canvas Support
    - Recommended Reading
  - Introduction to Accelerated ML
    - CPU/GPU Based Clusters
    - Key Components (cuDF, DASK, cuML, and Spark)
    - Distributed Computing Frameworks
    - Scale-out, Scale-up



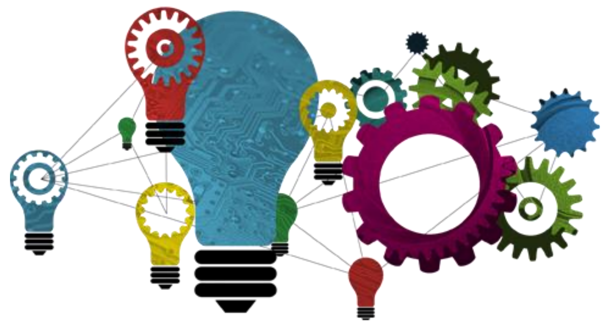
Learning outcomes

- At the end of this session you should understand:
  - The aims of the module
  - How you will be assessed
  - How you will learn on this module
  - How you can succeed on this module
  - The different forms of support available on this module
  - Have an understanding of accelerated ML and how to train models at scale



Module Overview: Aims of the Module

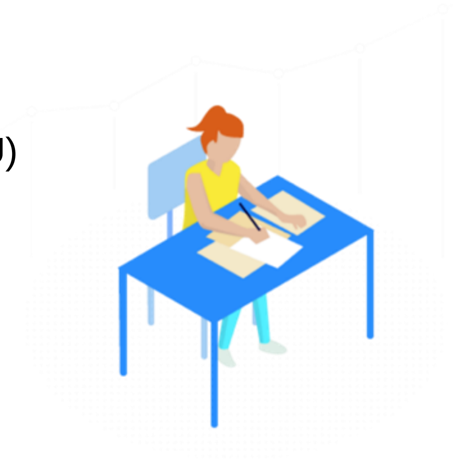
- To develop knowledge of accelerated machine learning at masters level and provide guidance on the purpose, design and development of accelerated machine learning projects.
- To provide an understanding of how the range of tools, techniques and algorithms can be applied for accelerated machine learning.
- To provide help on establishing accelerated machine learning design and development principles to successfully complete large scale machine learning projects.



Module Overview: Assessment on the Module

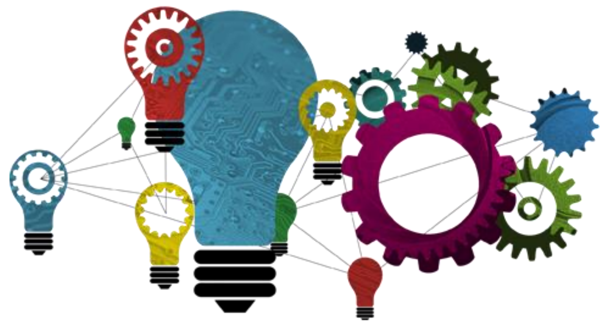
- This module is assessed using:
  - Coursework 1, is worth 40% of the final module mark which is individual essay on the theoretical Principles of Accelerated Machine Learning
  - Coursework 2, is worth 60% of the final module mark which is the development of an Accelerated Machine Learning Project
- The module has five learning outcomes:
  1. Describe in depth and detail the theoretical principles and objectives of accelerated Machine Learning (ML) using the Python-based NVIDIA RAPIDS framework
  2. Demonstrate deep understanding of relevant RAPIDS ML concepts and techniques
  3. Critically select appropriate RAPIDS ML algorithms to solve particular tasks
  4. Evaluate RAPIDS ML algorithms to determine their strengths and weaknesses
  5. Implement and test different RAPIDS ML algorithms using a suitable language, e.g. Python
  6. Evaluate the suitability of different processing architectures for specific computational tasks (CPU/GPU)
- The learning outcomes are mapped to the assessment components as follows:

Report	Prototype
1	3
2	4
6	5



Module Overview: Learning Activities

- Each week there is
  - 1 x 1 hour lecture followed by a 1 x 1 hour tutorial and a 1 x 1 hour lab
- The coursework is a **PRACTICAL** assessment that requires you to ***gain experience*** via:
  - The lab exercises
    - Applying what you have learned to new problem-based scenarios
  - The lecture and tutorial sessions
    - Observing the tutor

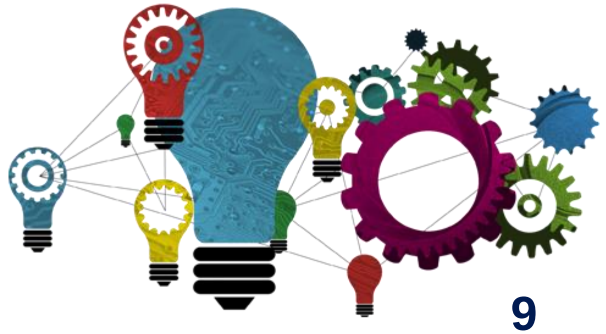


Module Overview: Learning Strategy

- Each week the theory and experience you gain will **build** upon the previous weeks
- So the module is **NOT** designed for you to dip in and out of
- It is designed for you to follow a sequence of learning experiences
  - If you miss a lecture or lab, YOU need to work on them in your own time before the next learning session
  - If you get stuck, simply ask for help at the next learning session
    - The ‘moral of story’ is ‘keep up so you don’t fall behind’

Module Overview: Learning Strategy

- There are 167 hours of private study time specified on the module specification
- In that time you should ‘play’ with the technologies in preparation for the assessment.
- You should also read through the designated reading material which is available on Canvas



Module Overview: Canvas Support

- Each week on Canvas you will find the following materials in the ‘Modules’ area:
  - Lecture slides
    - Bring your note books as more will typically be discussed than is provided in the slides
  - Lab exercises
    - The lab exercises can be found on your dedicated lab computer
  - Web links and further reading
    - Links to websites and further reading that will support you in your learning
- There will be a checkpoint assessment on Canvas (week 5)
- This quiz will assess how well you have learned the concepts covered to date

Module Overview: Canvas Support

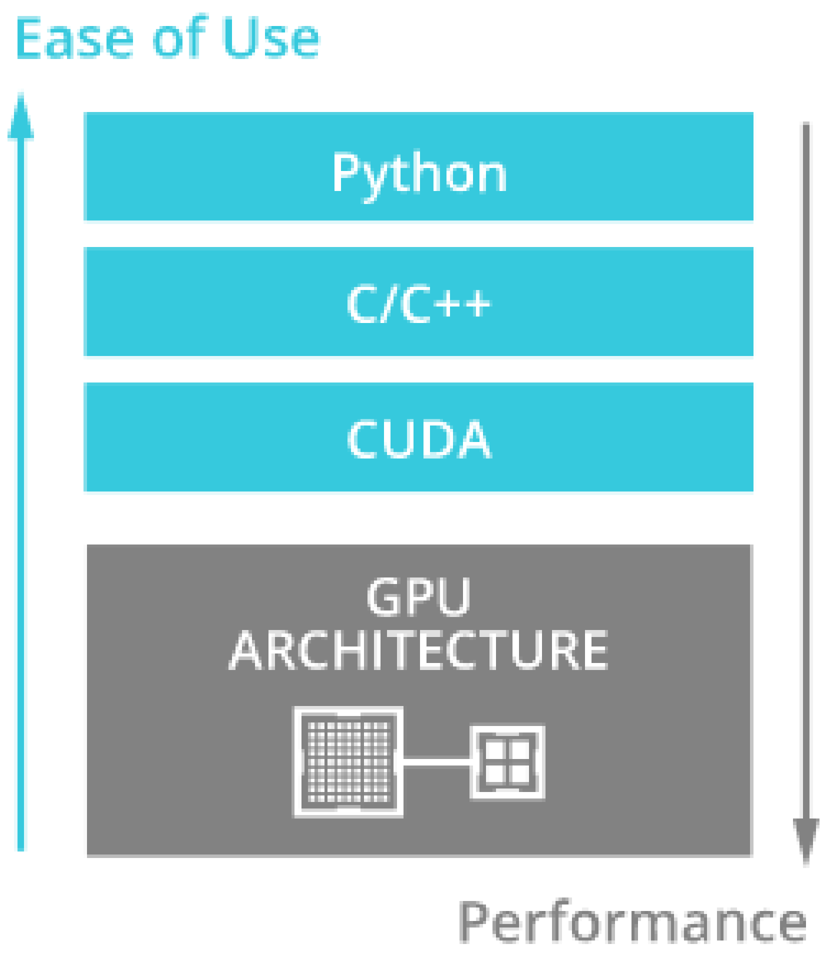
- When we start to learn the TensorFlow Object Detection API you will also find a set of additional resources in the ’Modules’ area to help you with your learning
- In the ‘Modules’ area you will also find:
  - Module Proforma
    - The ‘specification’ for the module
  - Module Handbook
    - An informative ‘design’ of the module
  - Reading List
    - Access to books available in the library and some as eBooks
- ‘Working outside the lab’ Area
  - You can remote into your designated lab computer using the instructions provided
  - Although all of the module software is open source it computationally expensive so do as much as you can on the lab computer

Module Overview: Software

- We will use a variety of different frameworks and packages in this module
- Python is the default programming language and is used throughout the module

Module Overview: Labs

- Python/Jupyter Notebooks to test and implement your code
- cuDF and cuML
- Pre-processing with cuDF
- Training with cuML
- RAPIDS framework
- Distributed learning using DASK



Module Overview: Lecture Structure

Lecture 2: Distributed Accelerated Computing Frameworks Part 1

- Local vs Distributed
- Benefits of Scaling Out (CPU vs GPU)
- Job Execution Model (Master and Worker Nodes)
- Legacy Frameworks (Hadoop, MapReduce)
- Computation on Large Data sets
- Locality
- Introduction to Apache Spark

Lecture 3: Distributed Accelerated Computing Frameworks Part 2

- Benefits of RDDs over Map Reduce
- Spark Architecture
- Spart Transformations and Actions
- Spark Programming Languages
- Spark ML Lib
- Introduction to DASK (CPU)
- DASK Dataframes
- DASK Arrays
- DASK BAGS
- DASK Clusters

Lecture 4: Distributed Accelerated Computing Frameworks Part 3

- Scikit Learn and DASK Integration (DASK ML)
- Scikit Learn Joblib
- Setting up DASK
- Exploiting the DASK UI Dashboard
- Best Practices for DASK
- Limitations of DASK

Lecture 5: Introduction to GPU Computing

- Introduction to GPU Hardware
- Introduction to NVIDIA Accelerated Computing
- GPU vs CPU
- Introduction to CUDA
- CUDA Libraries
- Numba

Module Overview: Lecture Structure

Lecture 6 Introduction to RAPIDS

- What is RAPIDS
- Why is RAPIDS Used
- Libraries and APIs Overview
- cuDF
- cuML
- cuGraph
- cuML Algorithms
- RAPIDS Architecture
- cuDF Analytics
- cuML Machine Learning
- cuGraph Analytics

Lecture 7: Apache Arrow and DASK GPU Accelerated Computing

- In-Memory Data Processing
- Benefits
- cuDF and DASK Dataframes
- RAPIDS and DASK Integration
- Worker and Core Utilisation in GPUs
- Benefits of using DASK with cuML

Lecture 8: AutoML Part 1

- Introduction to AutoML
- AutoML Pipeline
- AutoML Solutions
- AutoWEKA
- Auto-sklearn
- TPOT
- DataRobot
- H20 AutoML
- AutoKeras

Lecture 9: AutoML Part 2

- Driverless AI
- Google AutoML
- Microsoft Azure AutoML
- Amazon SageMaker Autopilot
- AutoML Challenges and Opportunities

Lecture 10: Next Generation Software and Hardware for Accelerated Computing

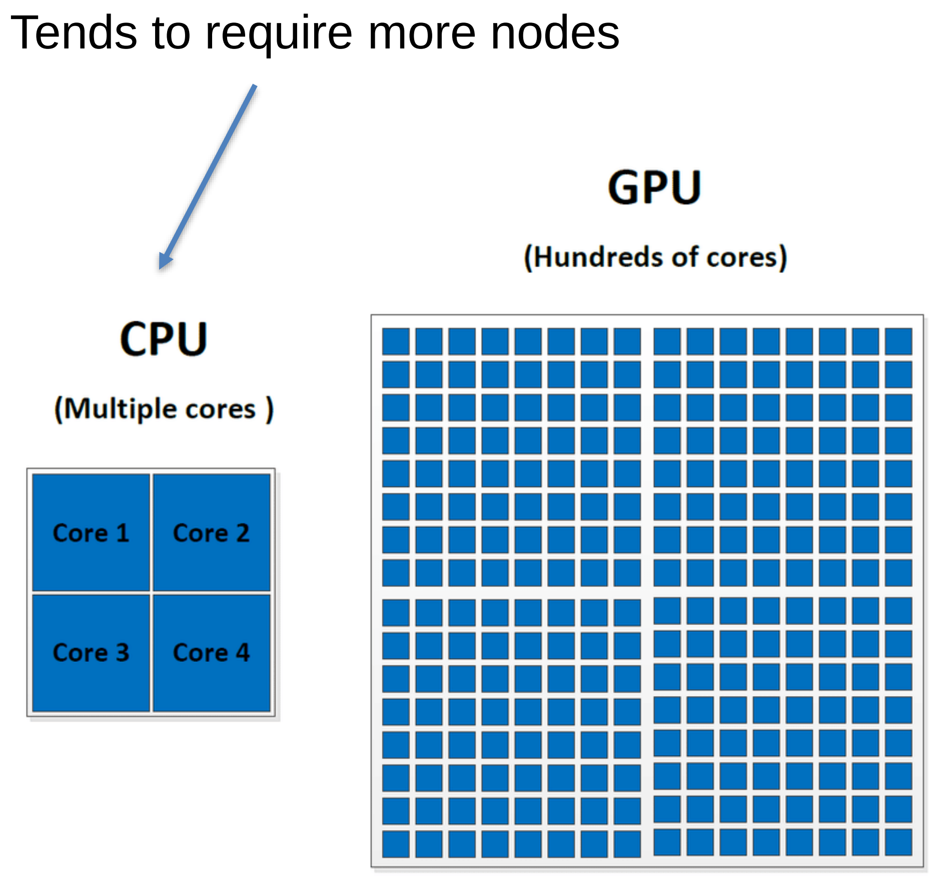
- What is next for Accelerated Computing Software (RAPIDS, DASK, ARROW)
- Current Hardware Limitations in ML
- ML on a Chip (TPU, FPGA GPGPU etc)

Lecture 11: Future Directions in Accelerated Computing and AutoML

- Master Algorithm
- Machine Perception
- Re-engineering ML Hardware
- Responsible AI Development

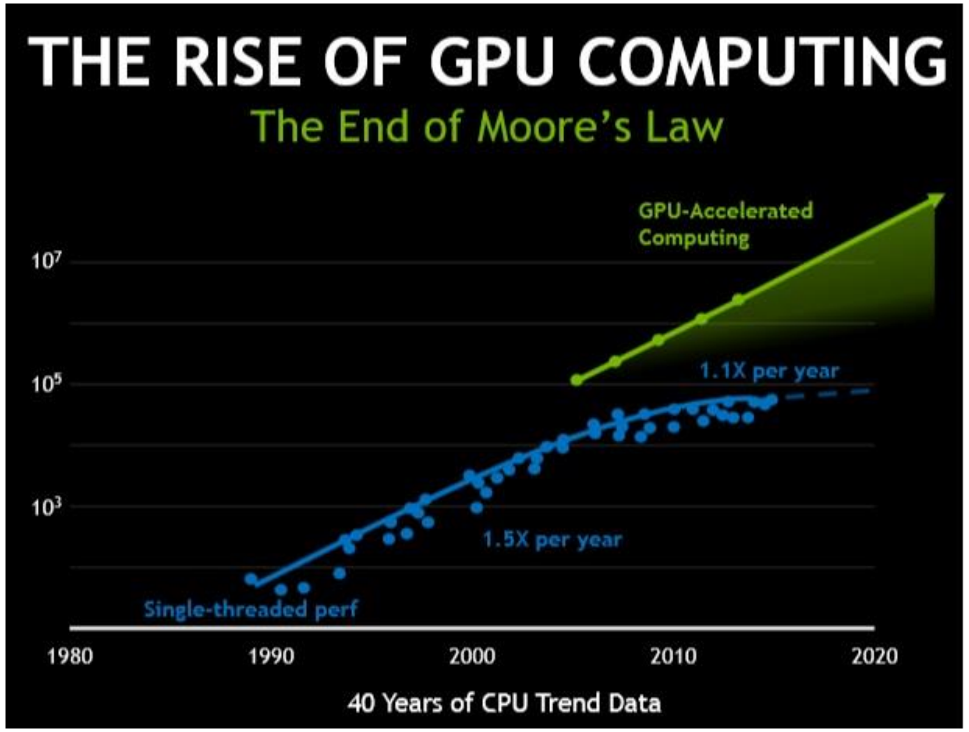
Introduction to Accelerated Computing

- Accelerated machine learning is exciting new paradigm in the field of artificial intelligence which aims improve the efficiency of training machine learning models
- This is achieved by providing users with the ability to execute end-to-end data science pipelines on GPU's or large-scale CPU based clusters
- Historically the training of traditional machine learning models such as Support Vector Machines and Random Forests has been restricted to the CPU compute (usually on a single core)



Introduction to Accelerated Computing

- Architectures based on single CPU’s which are found in all types of computer hardware (Intel/AMD) have started to stall in terms of their performance
- This is due to restrictions on the manufacturing process and heat dissipation issues
- The performance of CPU’s has seen limited improvement in recent years which is in stark contrast to the early decades which saw CPU performance double every year (Moore's Law)

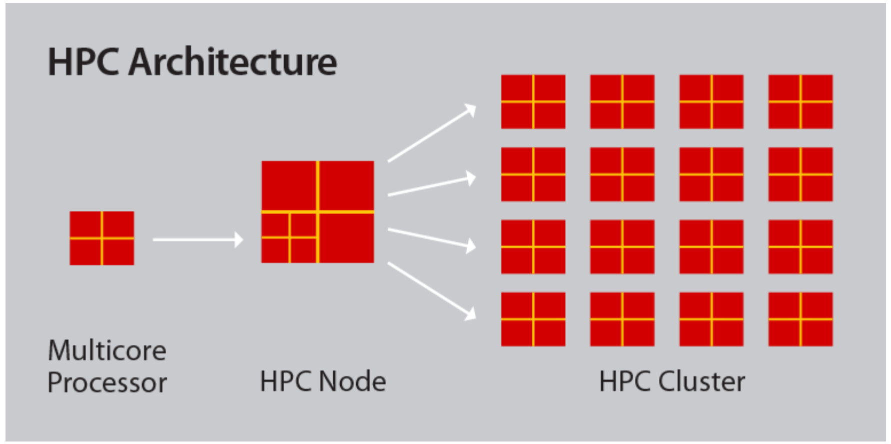


Introduction to Accelerated Computing

- To combat some of these limitations' CPU vendors have switched to the development and manufacture of multi-core CPUs which enable multithreading tasks
- These processors facilitated rapid performance and enabled giga floating-point operations per second (GFLOPS)
- However traditionally applications were designed to run sequentially and therefor dramatically slowing down the execution of the running process
- Although multithread and multicore CPU’s address some of these limitations as sequences of the application can be run and scheduled on the CPU in parallel the effect on time execution is limited

Introduction to Accelerated Computing

- Until recently high-performance parallel computing also known as (HPC) focused on adding additional CPU’s into each compute node and joining multiple nodes through InfiniBand connections to allow large parallel processing tasks to be run at scale
- HPC developers have been designing and implementing parallel programs for years which can run on large scale HPC infrastructures
- In the domain of data processing and AI resides some of the most computational demanding applications which has led significant interest in large parallel computing architectures

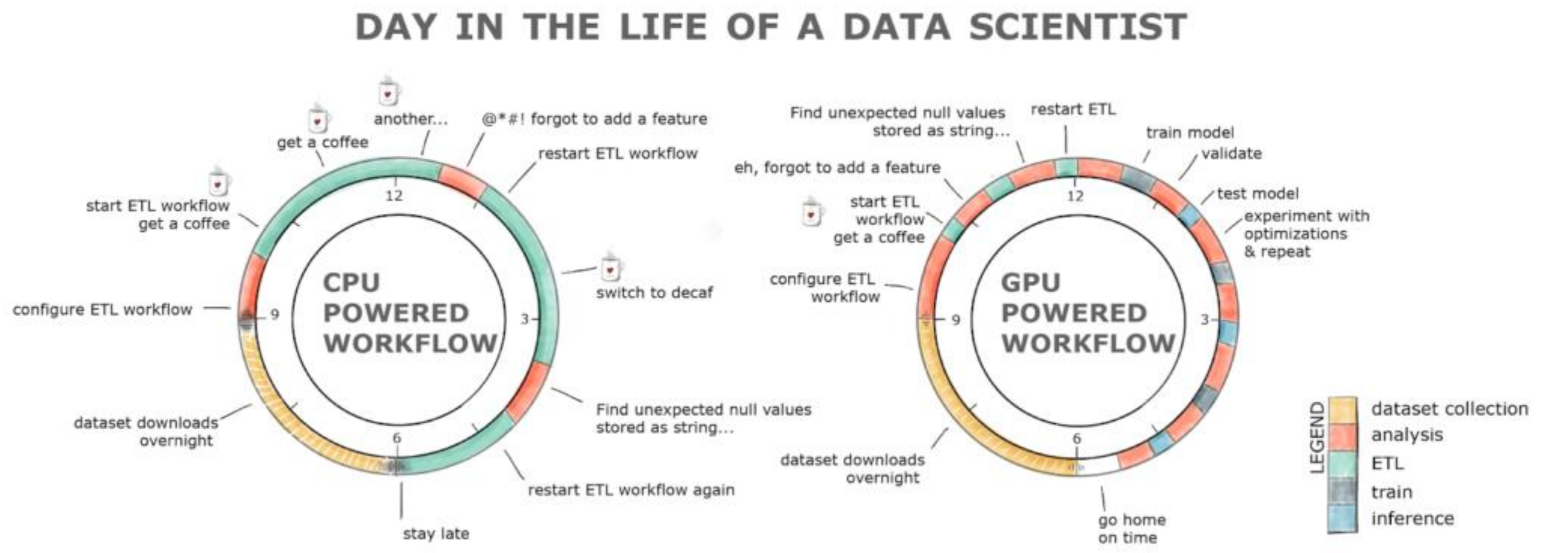


Introduction to Accelerated Computing

- Over the last few years hardware vendors have come to realise that bottlenecks in multicore and multithreaded CPU’s are starting to limit the types of computational jobs we are able to undertake
- This has led to an increase in the use of GPU’s within HPC to try and remove processing bottle necks and provide the compute needed in high end applications such as data analytics and AI
- GPU’s are designed for high throughput parallel processing which can be 10X+ faster the CPU’s for parallel code execution
- As such new frameworks and libraires (discussed later in this module) have been developed to exploit the underlying architecture of GPU’s and are a key component in many data processing pipelines

Introduction to Accelerated Computing

- With the ever-increasing volume of data, traditional approaches to the machine learning pipeline introduces signification bottles necks and delays from to training to inference
- With the introduction of GPU accelerated machine learning more data can be processed at scale which in turn delivers results significantly quicker

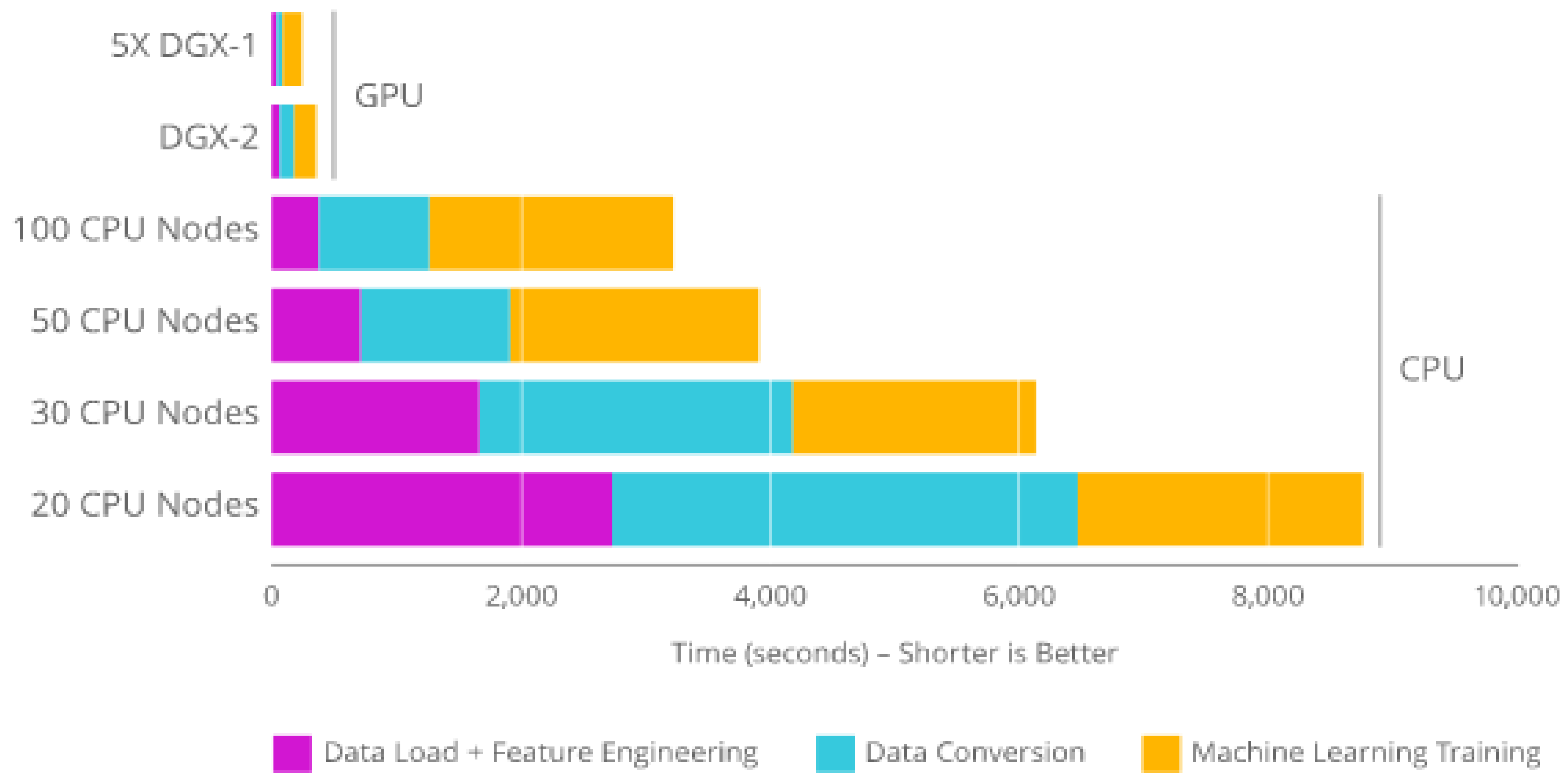


Introduction to Accelerated Computing

- Machine learning practice has traditionally resulted extended periods of wasted time due to slow stages in the Extract Transform and Load Process (ETL)
- This has led to underutilisation in data science teams while delaying the time to extract intelligence from the data science pipeline and deliver new products and services
- However, with the utilisation of GPU’s and the associated frameworks and libraries to exploit them we can now utilise GPU’s for general purpose data analytics and machine learning tasks
- This approach provides a much more efficient workflow through GPU accelerated data processing therefore giving organisations a competitive edge

Introduction to Accelerated Computing

From a datacentre management point of view this means there are typically less nodes to manage (firmware/software patching) and facilitates a more scaleup up approach instead of scaling out



CPU/GPU based clusters

- The CPU is composed of just a few cores with lots of cache memory that can handle a few software threads at a time
- In contrast, a GPU is composed of hundreds of cores that can handle thousands of threads simultaneously
- Due to its ability to perform multidimensional processing GPUs are utilised greatly for machine learning and AI technology
- Although 64 cores are a lot for CPUs, the number pales in comparison to the number of cores in a GPU
- While GPU cores are simpler and target more specific applications, general-purpose GPUs (GPGPUs) are becoming more flexible

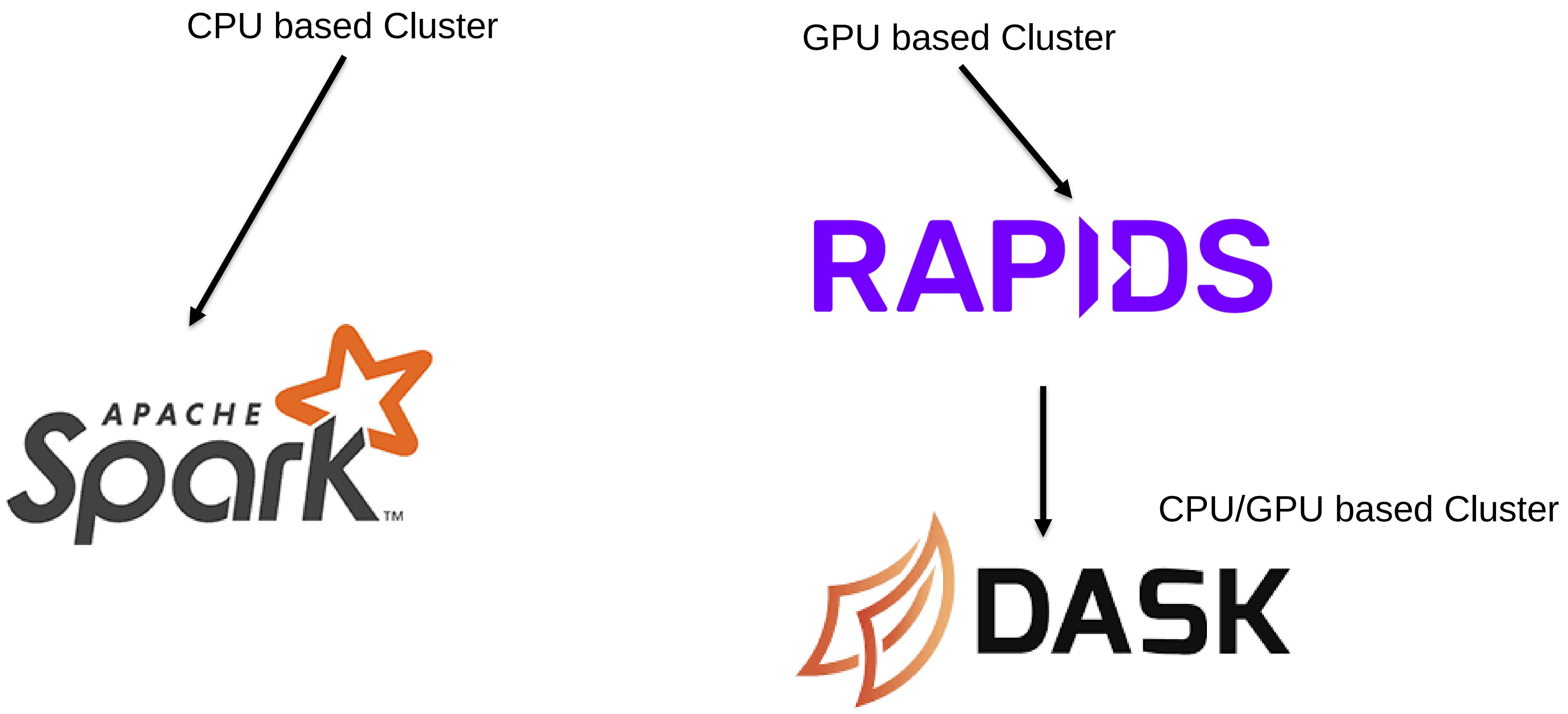
CPU/GPU based clusters

- Programming platforms like OpenCL have pulled GPGPUs into the programming mainstream and they were the initial workhorses in the rise of AI/ML applications
- Accelerated computing can be achieved in a number of ways
- Big data frameworks are designed to utilise specific hardware types based on infrastructure configuration
- For example, HPC data centres in most organisations will have large CPU availability and this has led to the implementation of frameworks such as Spark
- In contrast some organisations have a blend of hardware containing both CPU’s and GPU’s

CPU/GPU based clusters

- As a result, frameworks such as Apache Arrow and RAPDIS can be used to speed up data analytics by taking advantage of GPU hardware and the paralisation it offers

CPU/GPU based clusters



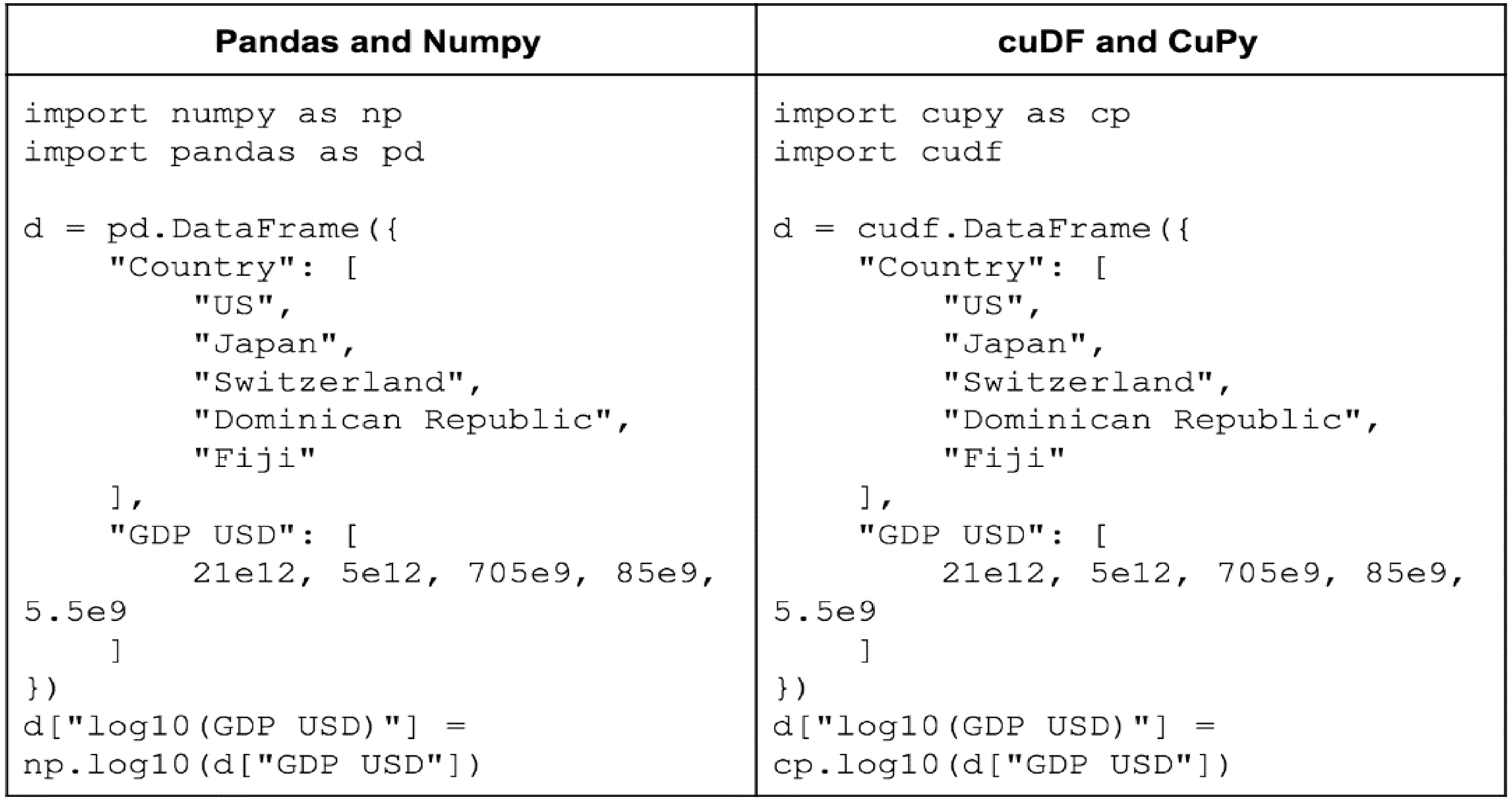
Key Components (cuDF, cuML and DASK)

- Most packages including Pandas only run-on CPU’s and due to the limited number of cores available processing large dataframes can be problematic
- Large datasets can have millions, billions, or even trillions of data points that need to be processed in timely manner
- CuDF is a python based dataframe library which allows you perform a wide variety of pre-processing tasks such as loading, joining, aggregating and filtering data at scale
- By using cuDF we gain access to a much wider array of graphics cores which dramatically speed up data processing tasks

Key Components (cuDF, cuML and DASK)

- cuDF’s API is a mirror of Pandas and can in most cases be used as a direct replacement
- As a result, it makes it easy for data scientists, analysts and engineers to integrate it into their existing workflows
- The framework allows you to convert your existing Pandas dataframes into a cuDF dataframe which gives you instant access to the GPU
- cuDF supports most of the common DataFrame operations that Pandas does meaning that code can be accelerated with little effort
- cuDF provides a pandas-like API that will be familiar to data engineers & data scientists, so they can use it to easily accelerate their workflows without going into the details of CUDA programming

Key Components (cuDF, cuML and DASK)



Pandas and Numpy

```
import numpy as np
import pandas as pd

d = pd.DataFrame({
    "Country": [
        "US",
        "Japan",
        "Switzerland",
        "Dominican Republic",
        "Fiji"
    ],
    "GDP USD": [
        21e12, 5e12, 705e9, 85e9,
5.5e9
    ]
})
d["log10(GDP USD)"] =
np.log10(d["GDP USD"])
```

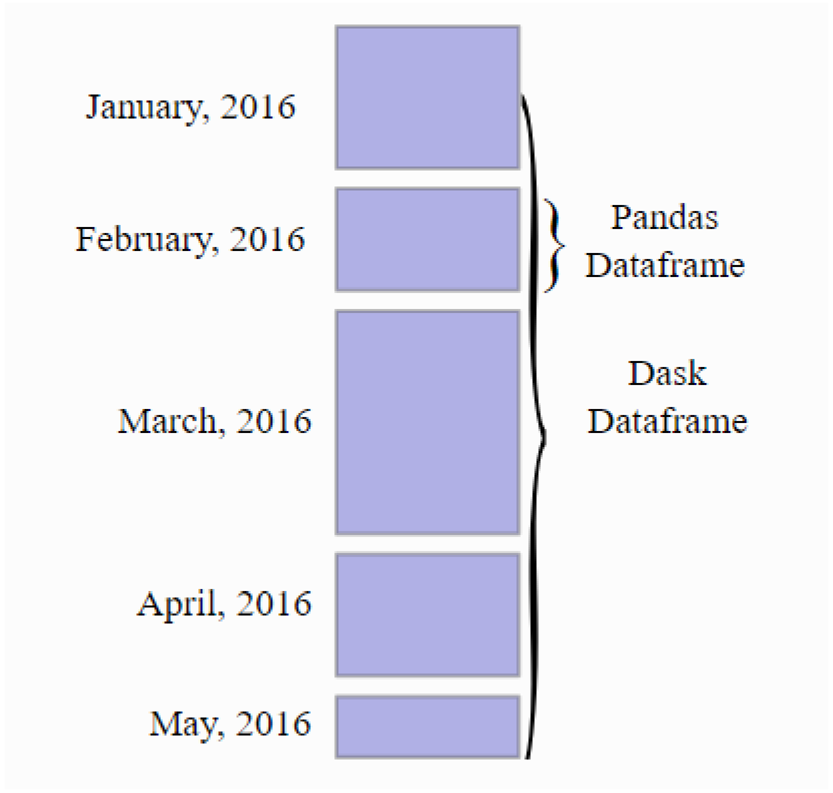
cuDF and CuPy

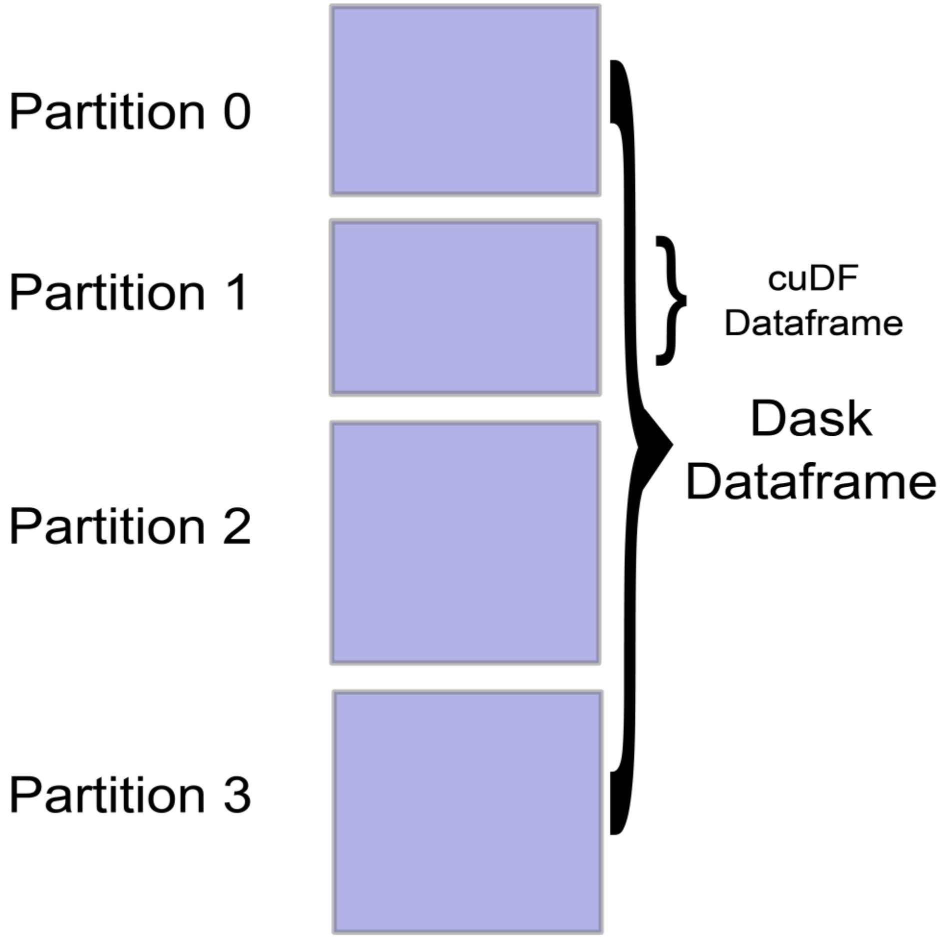
```
import cupy as cp
import cudf

d = cudf.DataFrame({
    "Country": [
        "US",
        "Japan",
        "Switzerland",
        "Dominican Republic",
        "Fiji"
    ],
    "GDP USD": [
        21e12, 5e12, 705e9, 85e9,
5.5e9
    ]
})
d["log10(GDP USD)"] =
cp.log10(d["GDP USD"])
```

Key Components (cuDF, cuML and DASK)

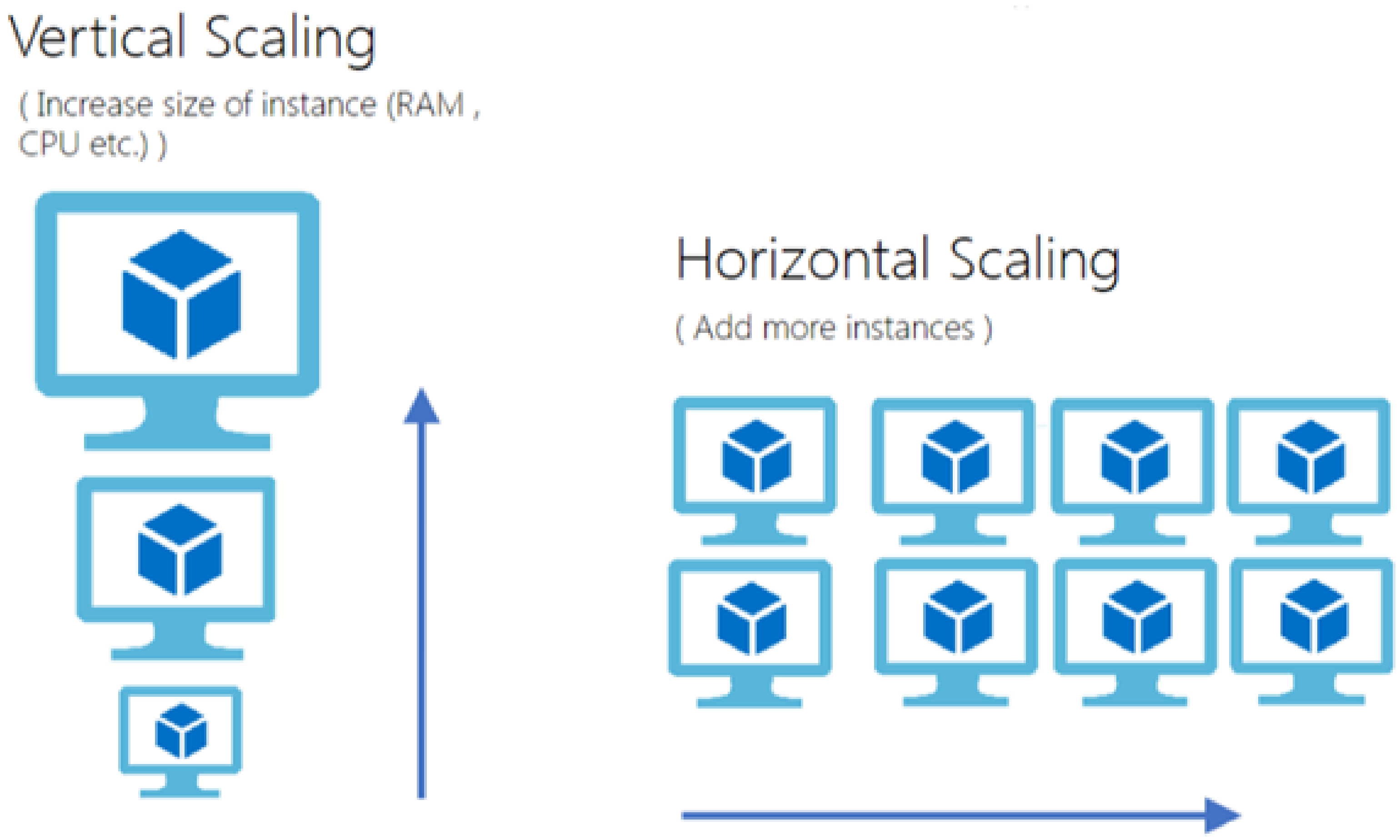
- cuML is a suite of fast, GPU-accelerated machine learning algorithms designed for data science and analytical tasks
- Its API is similar to Sklearn’s. This means you can use the same code you use to train Sklearn’s model to train cuML’s model
- CuML drastically reduces the amount of training time required to train traditional ML models
- Although in many cases, you don’t need to install cuDF to use cuML, cuDF is a nice complement for cuML since it is a GPU DataFrame
- Sklearn is a great library with a variety of machine learning models that you can use to train your data. But if your data is big, it might take you a long time to train your data, especially when you experiment with different hyperparameters to find the best model

Key Components (cuDF and DASK)





Scale-out, Scale-up



Next Session

- Local vs Distributed
- Benefits of Scaling Out (CPU vs GPU)
- Job Execution Model (Master and Worker Nodes)
- Legacy Frameworks (Hadoop, MapReduce)
- Computation on Large Data sets
- Locality
- Introduction to Apache Spark