GPU Accelerated Cloud-Native Geospatial

Tom Augspurger | CNG 2025 | May, 2025

Themes

- GPUs are fast
  - Useful for analytical, simulation, ML/DL, … workloads
- The cloud and CUDA / RAPIDS makes starting *much* easier
- Using GPUs *optimally* takes care
- Some heuristics to identify and optimize workloads that can benefit from GPU acceleration
- Figure out *together* where GPU-acceleration can be helpful

Definitions

- The cloud provides **programmable**
  - Compute
  - Storage
  - Networking
- Storage is separate from compute.
  - This affects many things!
- GPUs provide **accelerated compute**

```
$ coiled batch run \
    python job.py
```

```
$ coiled batch run \
    --n-tasks 100 \
    python job.py
```

```
$ coiled batch run \
    --n-tasks 100 \
    --vm-type Standard_ND96asr_A100_v4 \
    python job.py
```

Demo

Accelerated Computing Swim Lanes

RAPIDS makes accelerated computing more seamless while enabling specialization for maximum performance

Easier to Use

Zero Code Change: Acceleration Plugins *(no-code change)*

cudf.pandas: Accelerated Pandas, Accelerated NetworkX nx-cugraph, RAPIDS Spark Accelerator, Array-API backed Scikit-learn...

Hybrid CPU/GPU libraries *(minimal change)*

Pytorch, FAISS, Tensoflow, XGBoost, cuML-CPU ...

Dask, pySpark...

GPU Python Libraries *(GPU Python code)*

RAPIDS core libraries (cuDF, cuML, cuGraph, cuVS), RMM, CuPy, Numba, OpenAI Triton ...

Python/CUDA libraries *(Hybrid Python / CUDA code)*

CuPy RawKernels, Numba CUDA, Cython wrappers for CUDA ...

C++/CUDA high level *(High-level C++/CUDA code)*

RAFT, CCCL: Thrust, CUB ...

CUDA Toolkit *(C++/CUDA code and kernels)*

cuBLAS, cuDNN, cuSolver, cuSPARSE, ...

Maximum Performance

Typical Workload

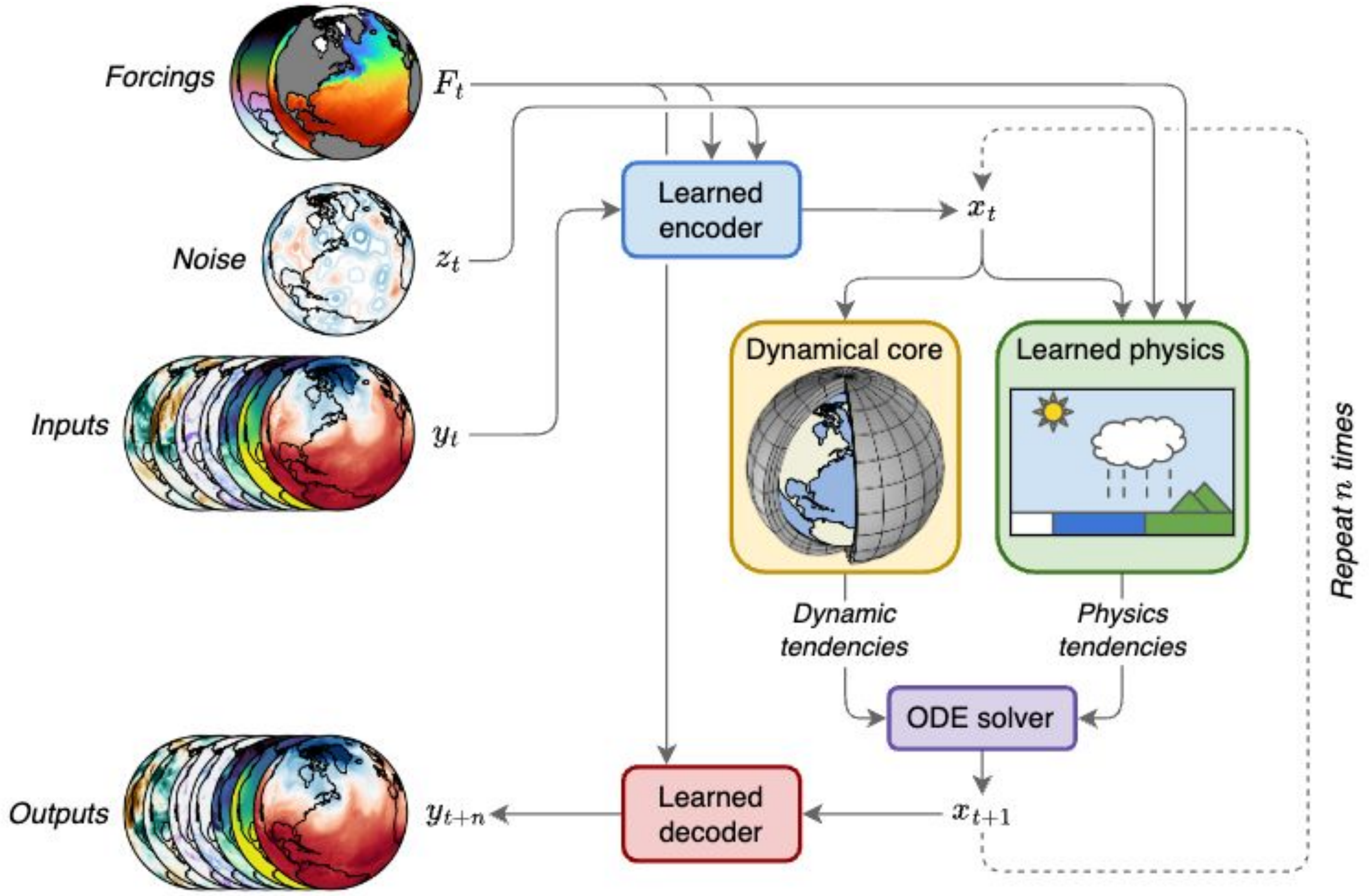
- Load inputs
- Compute
- Store result

Typical Workload: Cloud-free Mosaic

- Load inputs: timeseries of imagery over some
- Compute: Align pixels, median over time
- Store result: single cloud-free mosaic

Typical Workload: Model training

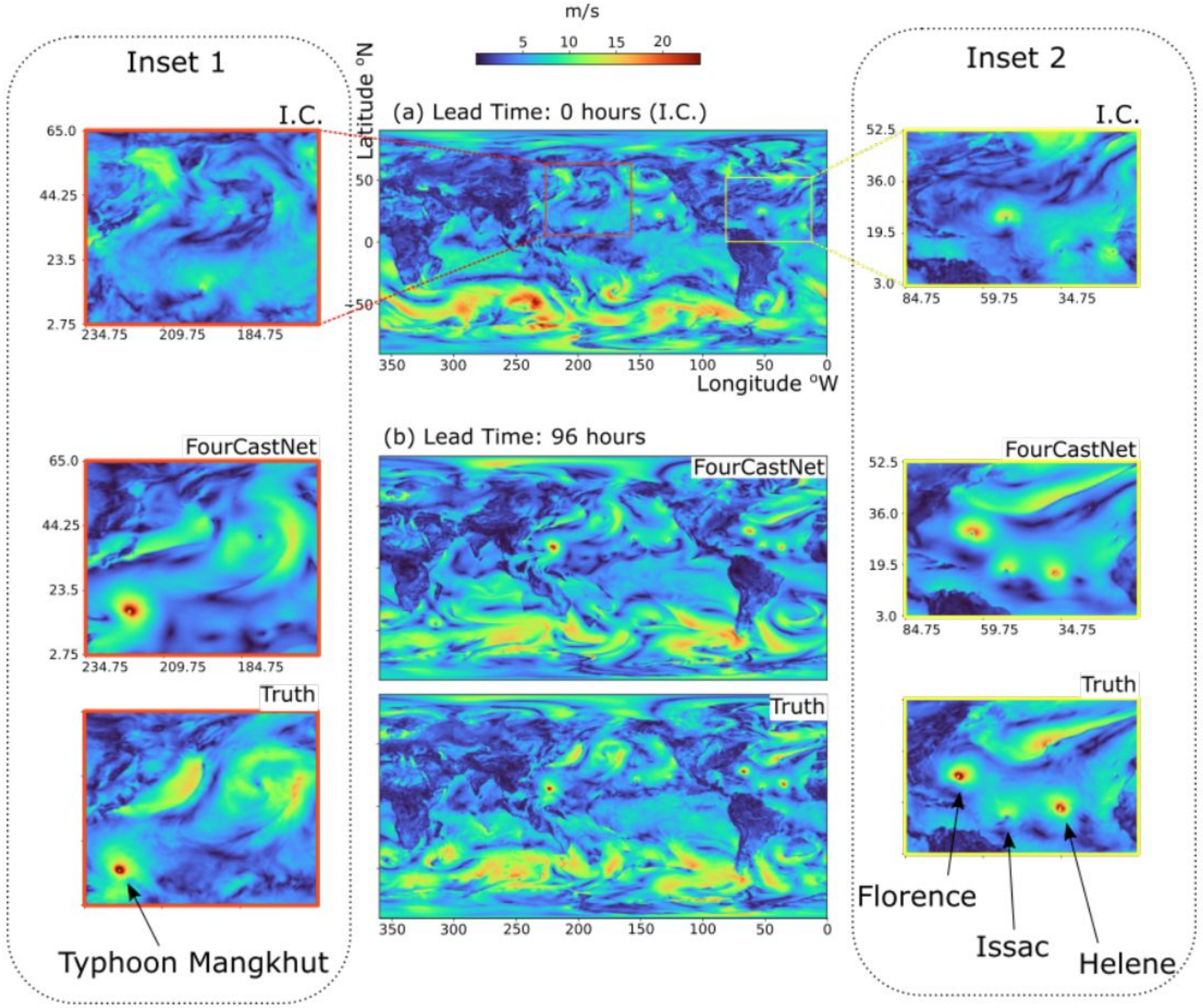
- Load inputs: imagery / labels / forcings / ...
- Compute: Update model weights
- Store result: Store model weights



NeuralGCM (Kochkov et. al. 2023)

Typical Workload: Forecast

- Load inputs: Model weights, initial conditions
- Compute: Predict next *N* time steps
- Store result: Store / serve forecasts



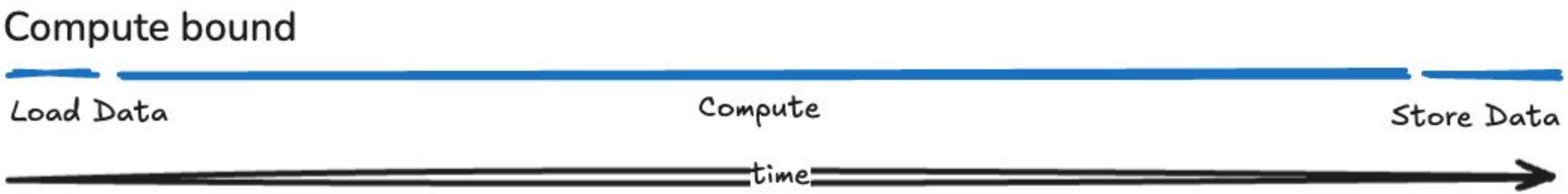
FourCastNet (Pathak et. al. 2022)

Where’s time spent?

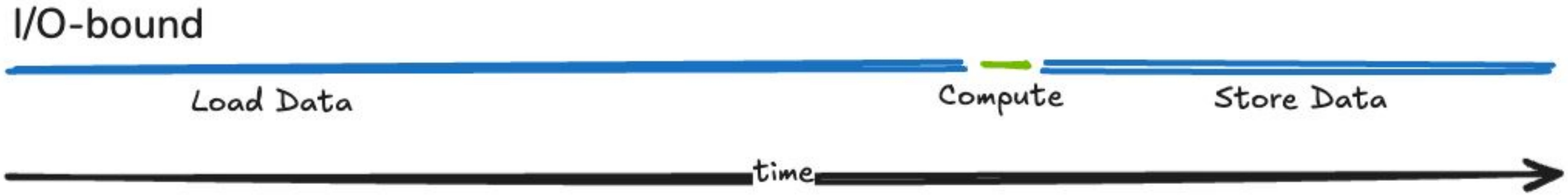
- Load inputs: *Mostly* network, but some compute
- Compute: Mostly compute, with caveats
- Store result: *Mostly* network, but some compute

Workflow timeline: I/O-bound

Workflow timeline: Compute-bound

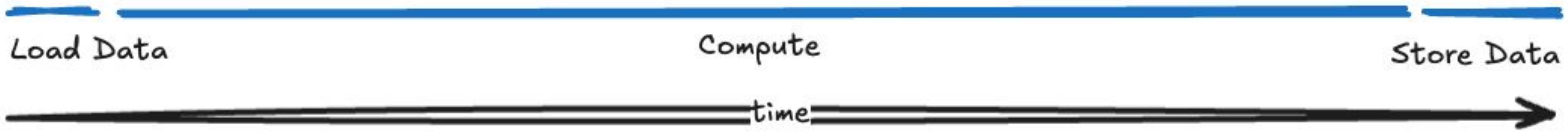


Accelerated Workflow: I/O-bound



Accelerated Workflow: Compute-bound

Compute bound



Compute bound

Compute *Accelerator*

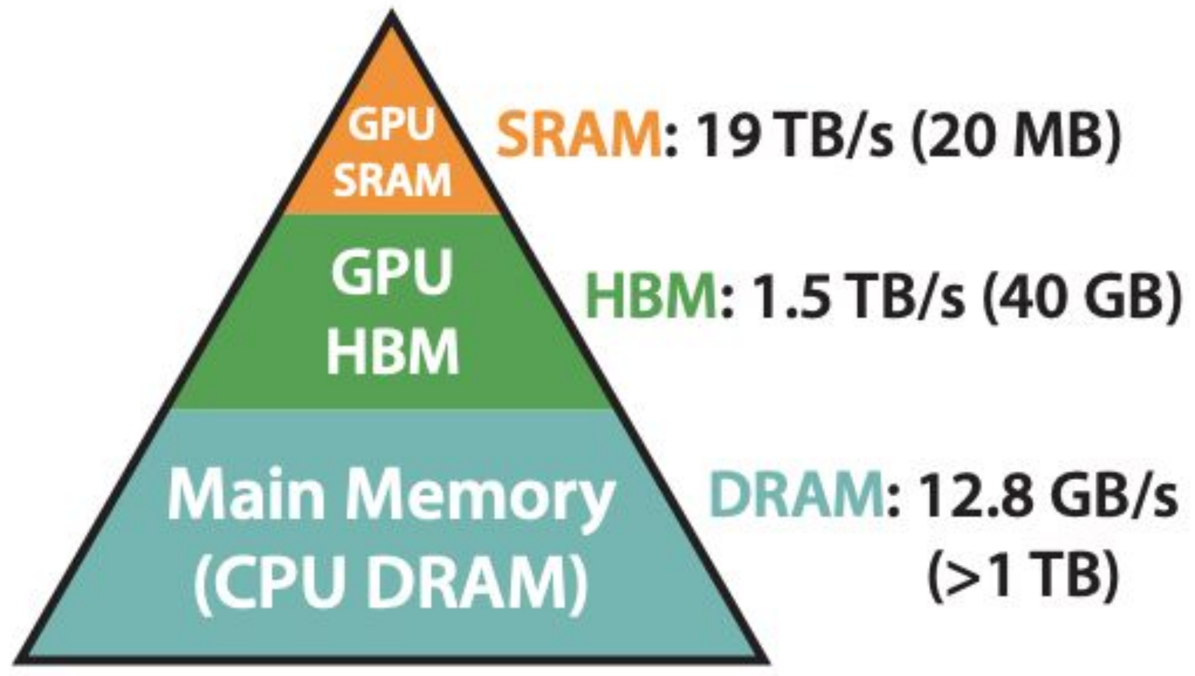
- GPUs are part of a larger system
- Use your network, CPUs, RAM, GPU interconnects, efficiently

Bad pipelining

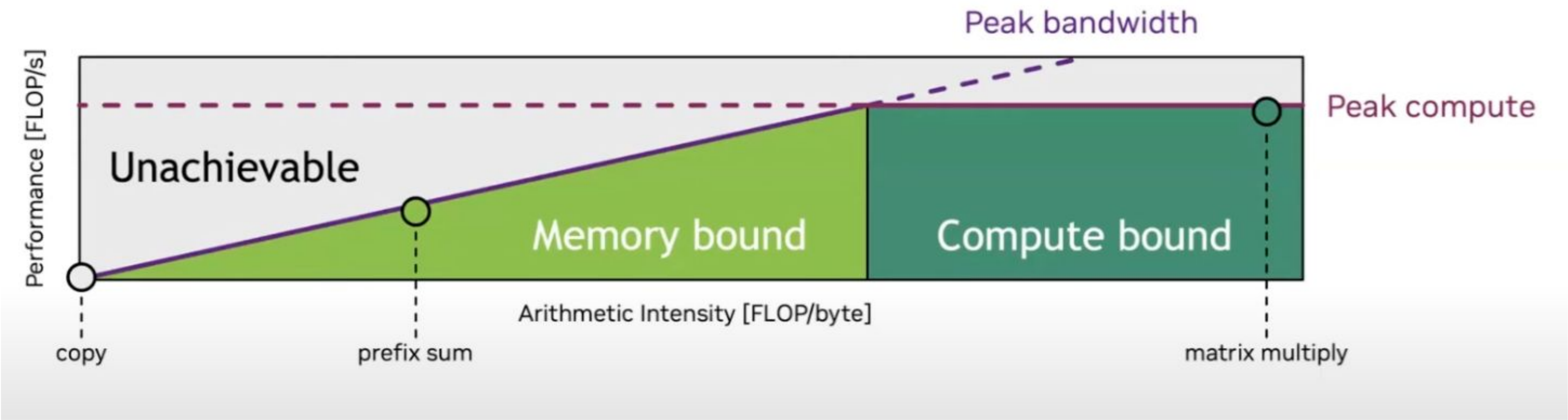
Good pipelining

Caveat: Memory Bandwidth

- Copying data from host memory to device memory is relative slow
- Minimize the amount of transfers
- Maximize the compute operations per byte read / stored



Memory Hierarchy with Bandwidth & Memory Size



Summary

- Profile your workloads
- Focus on compute bound problems
- Use pipelining to avoid I/O bottleneck
- Start with RAPIDS / cupy / torch / jax / Earth2 / Zarr / …
- Please reach out if you have issues or use-cases

Thanks!

toaugspurger@nvidia.com | @TomAugspurger

Smooth Installation and Packaging

Meeting a wide array of needs with easy to install and use distribution methods

Standard binary distributions

- One line installation methods
  - ***Conda***:
    - With conda-forge CUDA packages, and rapidsai RAPIDS packages
  - ***Pip*** installation
    - Custom index to support all RAPIDS ecosystem.

Containers

- Thin **CUDA** containers
- ***RAPIDS*** and other ***application-level*** containers
- ***Devcontainers*** for developers wanting to contribute
- ***Specialized containers***, for example Vector Search benchmarks
- ***NVIDIA AI Enterprise***

CUDA Compatibility

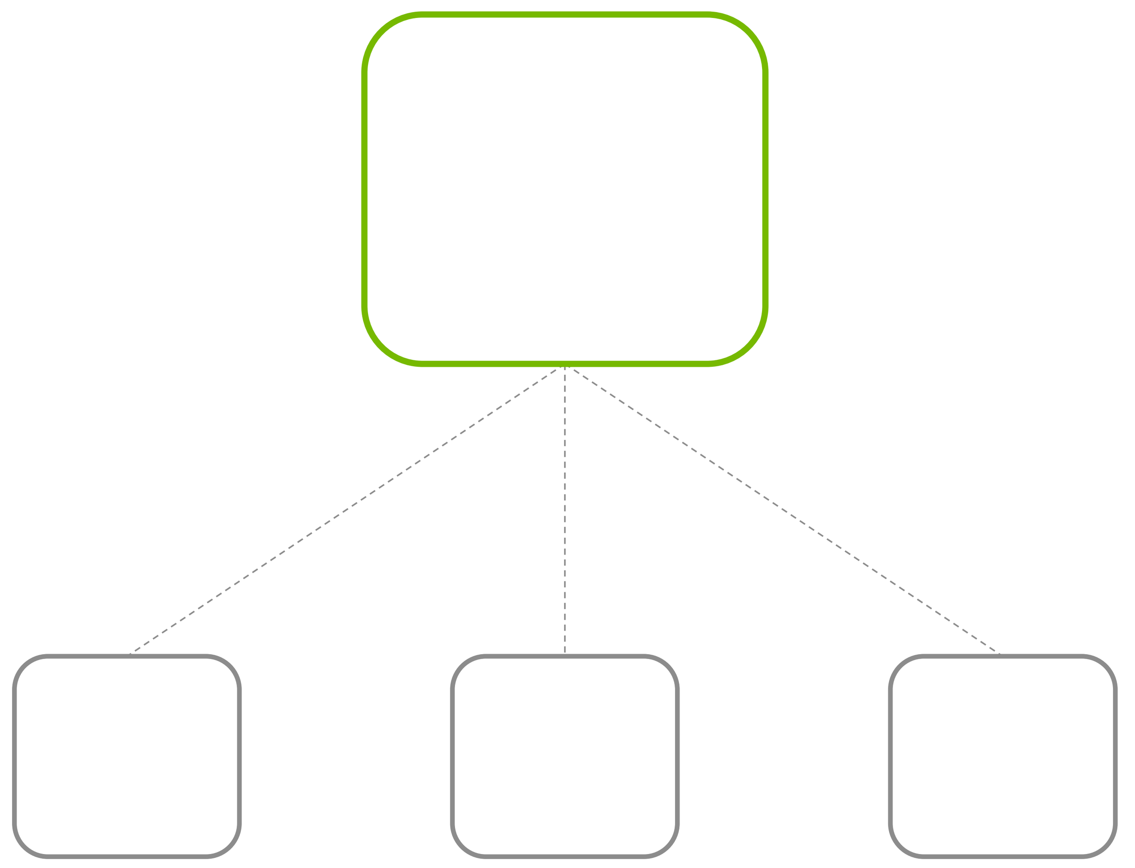
- One package works across major CUDA version
  - True for all distribution methods
- CUDA Toolkit available entirely on conda-forge
  - No need for system CUDA Toolkit, just driver

JIT compilation

- Available for writing CUDA code at runtime
- Adds capability of running GPU code in latest GPU architectures

RAPIDS Deployment Models

Scales from sharing GPUs to leveraging many GPUs at once

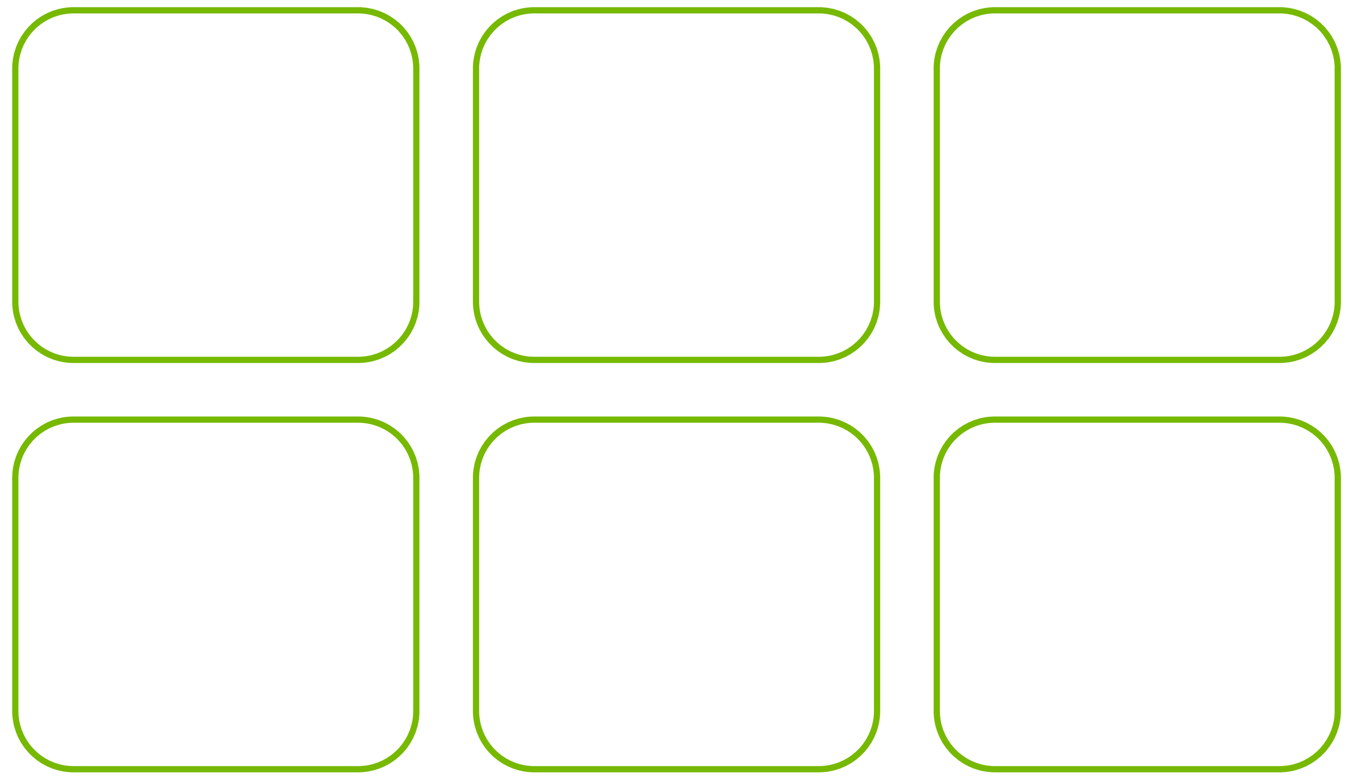


Shared Node

Scale out AI/ML APIs and model serving with NVIDIA Triton Inference Server and the Forest Inference Library

Single Node

Scale up interactive data science sessions with NVIDIA accelerated tools like cudf.pandas

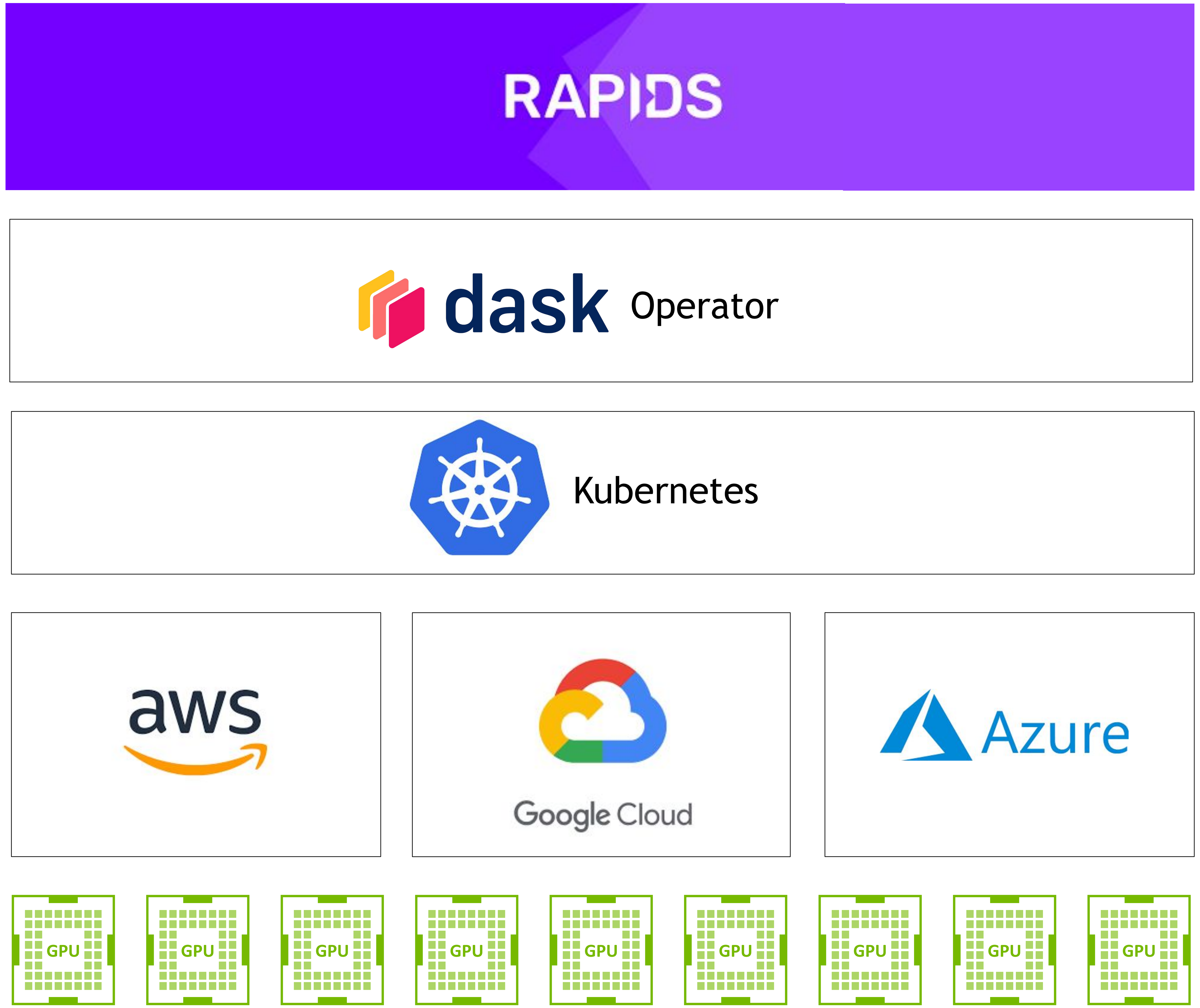


Multi Node

Scale out processing and training by leveraging GPU acceleration in distributed frameworks like Dask and Spark

RAPIDS on Kubernetes

Unified Cloud Deployments



RAPIDS runs your workloads faster

How do you want to spend those gains?

Reduce cost

Reduce the amount of time you need to run servers. Beneficial for reducing cloud costs.

Do more work

Run more workloads for the same time/cost. Process things that were not possible before.

Performance boost

Get work done faster. May help give a competitive advantage or reduce pressure on SLAs.

Environment impact

Reduce power needed to perform the same calculation. Using less power produces less CO_2.

Reduce context switching

Reduce time people need to wait for calculations to complete which helps avoid switching to a different task.

Improve accuracy

Acceleration could allow for more iterations or to process more data leading to improved model accuracy

Overview

- Data science needs accelerated computing
- NVIDIA RAPIDS is an end-to-end data science platform
  - The RAPIDS cuDF Python package provides a pandas-like API that accelerates common analytics workloads
- NVIDIA is committed to meeting data scientists and engineers where they are (pandas, NetworkX, Spark ML)
  - cudf.pandas provides zero-code-change acceleration for existing pandas scripts / notebooks
- Accelerated computing is powering the next wave of AI applications such as LLMs, large scale graphs