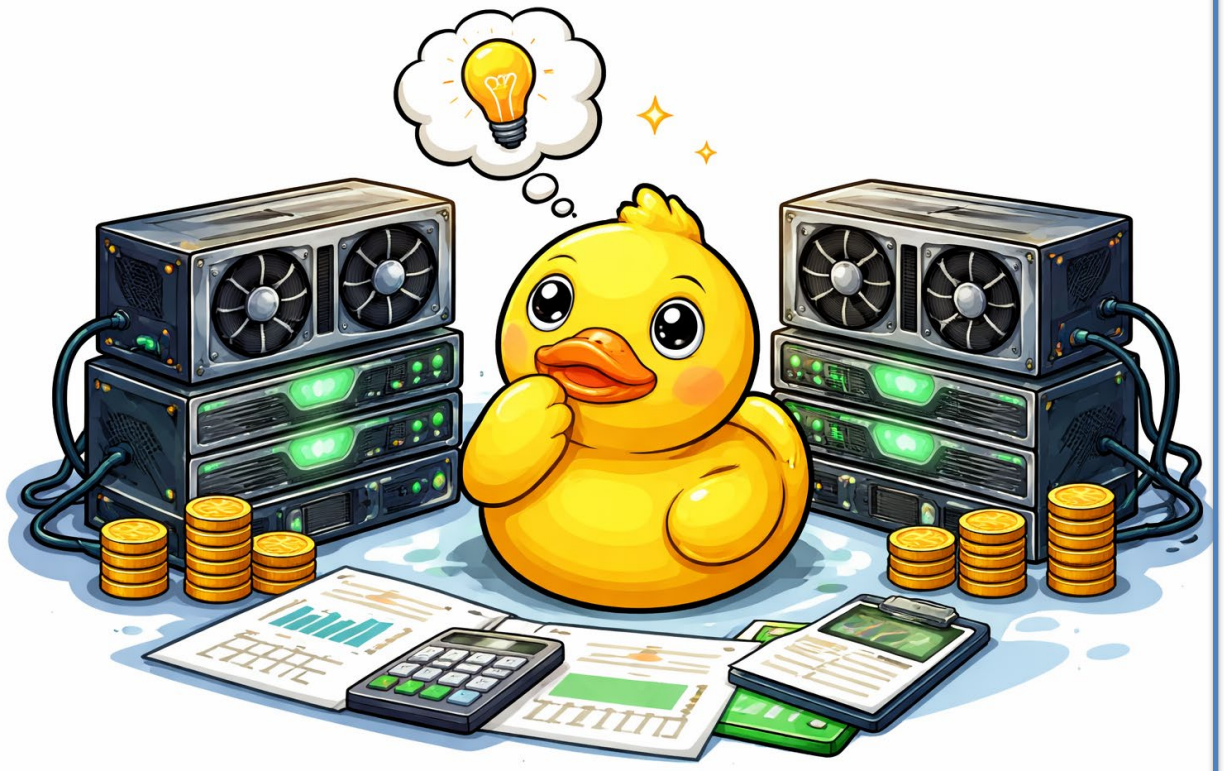
Improving the Efficiency and Scalability of Anemoi

5th ECMWF-ESA Machine Learning Workshop

Jan Polster, Cathal O'Brien, Marieke Plesske, Simon Lang
Ioan Hadade, Matthew Chantry

ECMWF

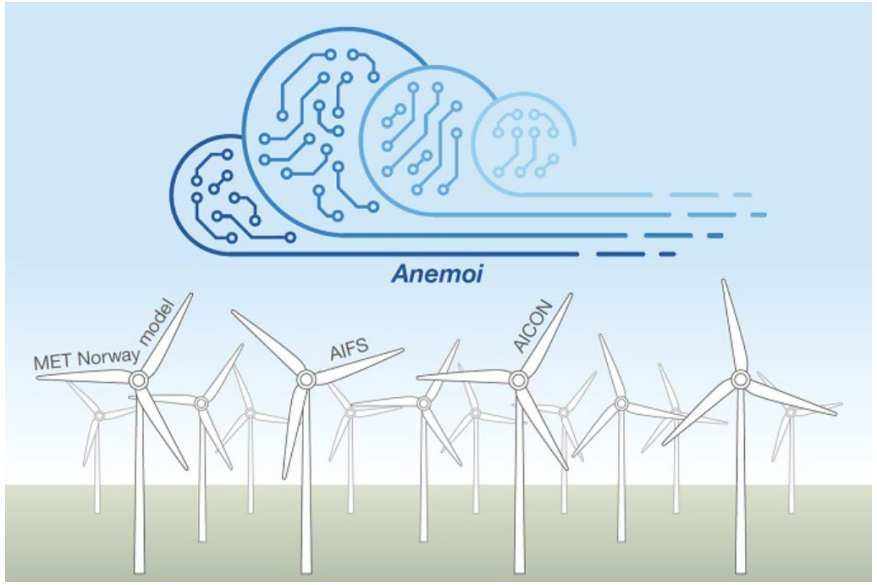
Efficiency vs Scalability

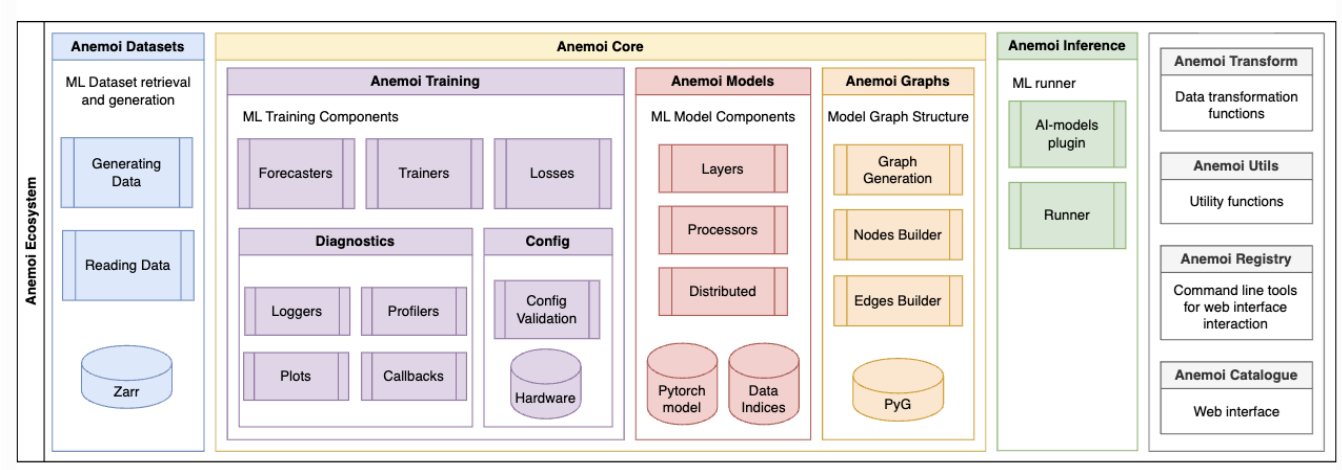


How do we achieve both across models, resolutions, and tasks?

Anemoi

- Anemoi is an open-source framework that provides a complete toolkit to develop data-driven weather models – from data preparation through to inference
- It's a highly modular framework organised into different Python packages
- The framework builds upon on established Python tool including PyTorch, Pytorch Lighting, Hydra, Pydantic, and earthkit.

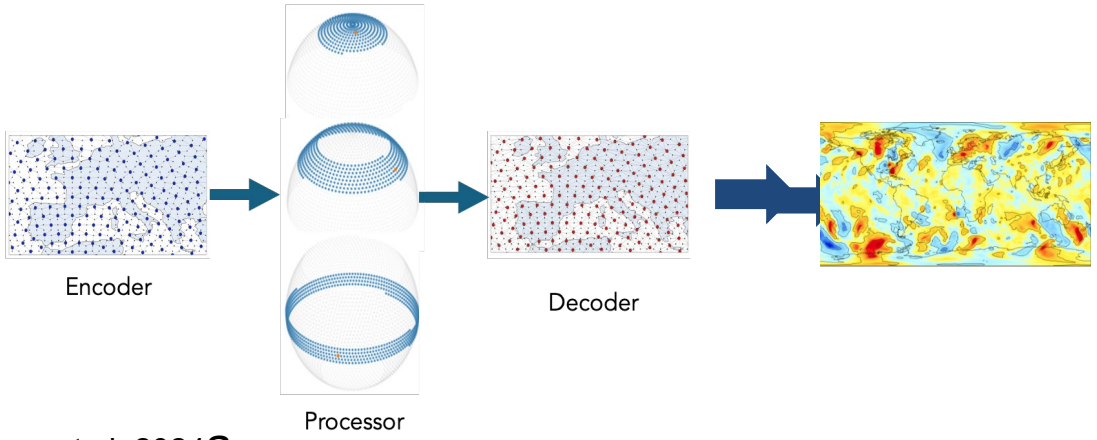




Many Models, One Framework

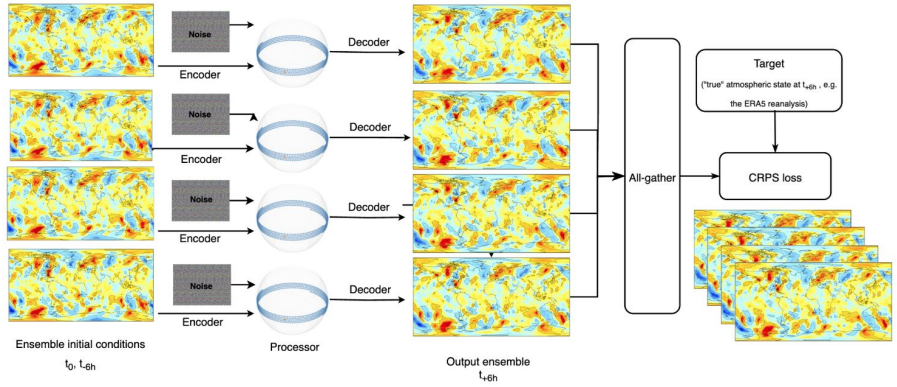
- Deterministic and Probabilistic Models

Deterministic Model



Lang et al. 2024a

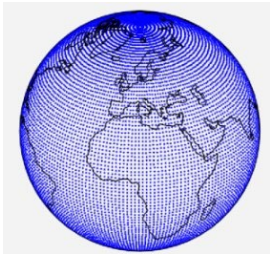
Probabilistic Model - Learn an ensemble via optimization of a probabilistic score



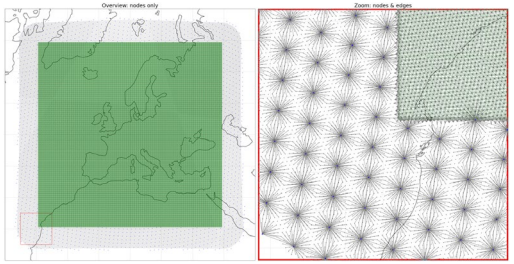
Lang et al. 2024b

https://arxiv.org/abs/2406.01465
https://arxiv.org/abs/2412.15832
https://arxiv.org/abs/2409.02891

- Global and Regional Area Models



Global Model

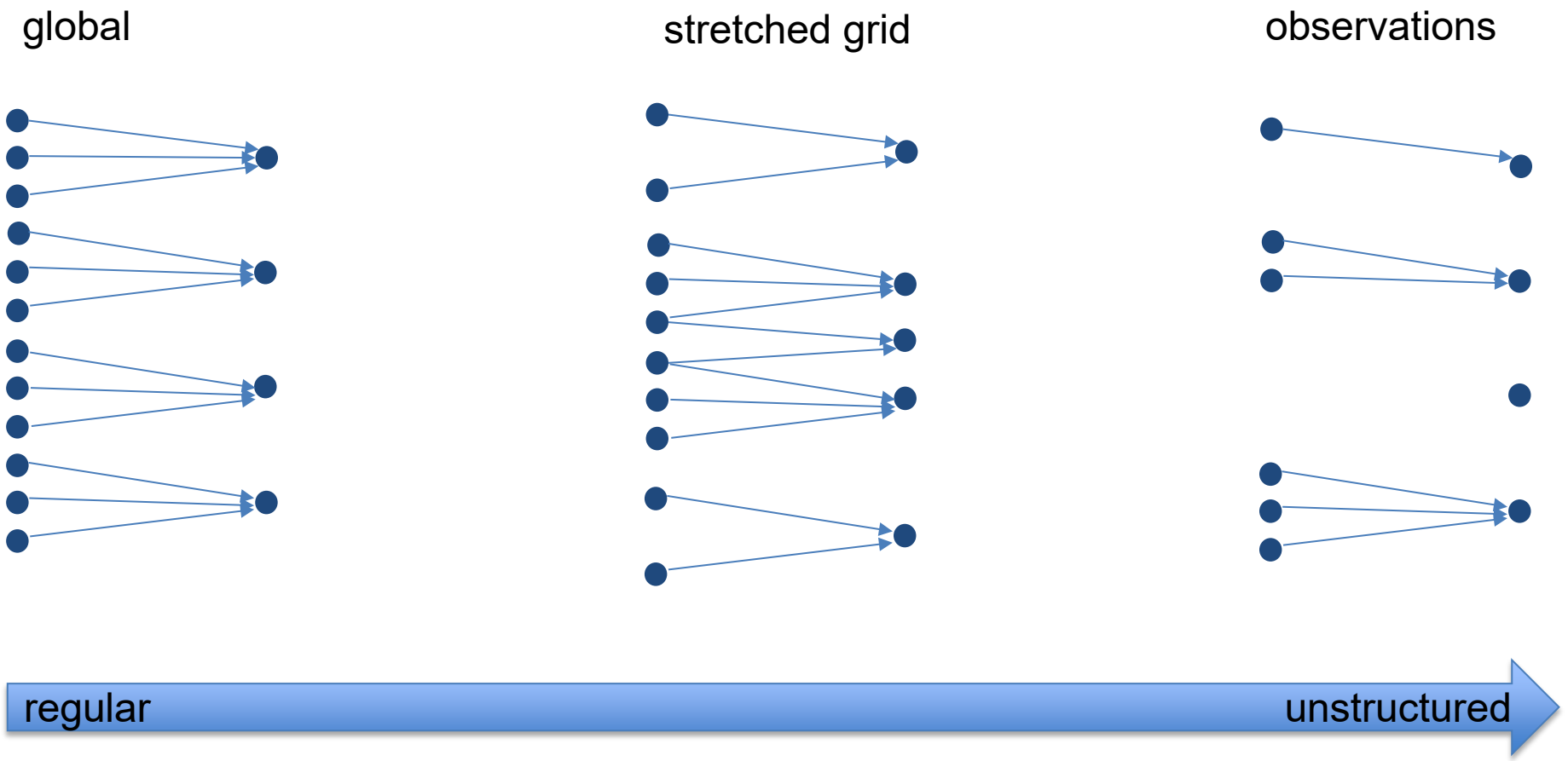


LAM

Nipen et al. 2024

Stretched Grid Model

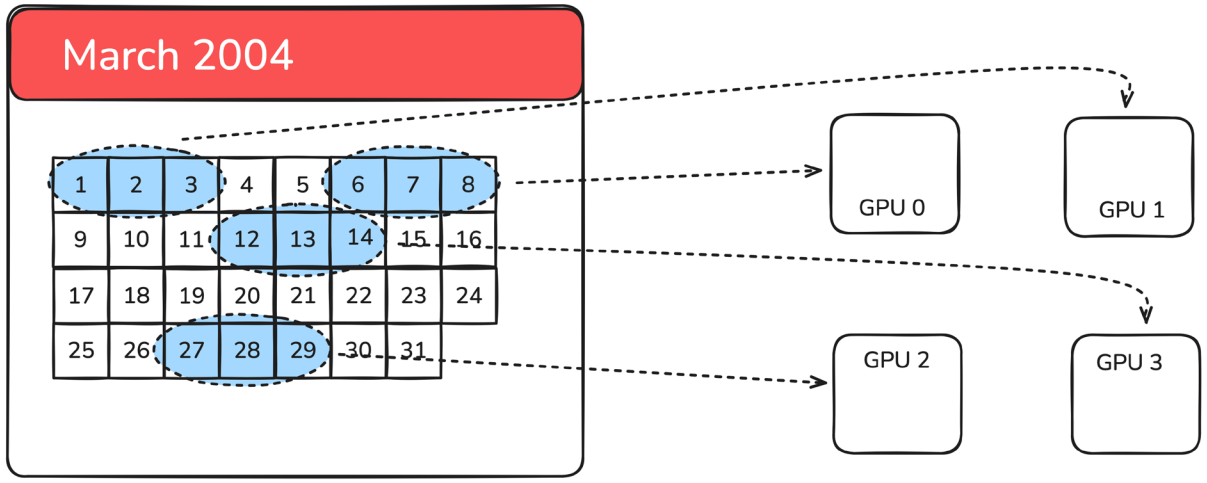
Everything is a Graph



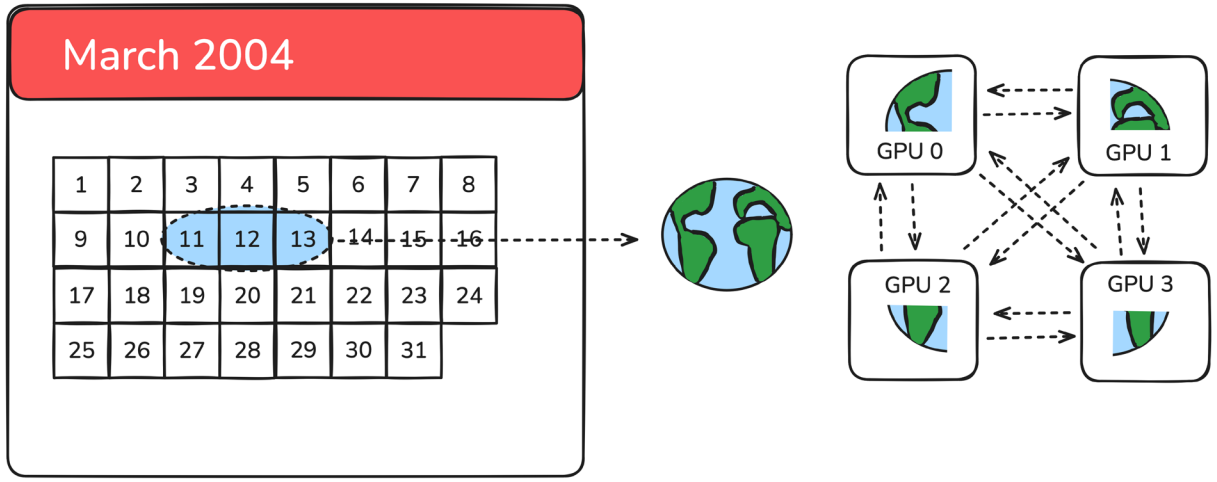
optimise the graph operations -> improvements across models

Parallelism in Anemoi

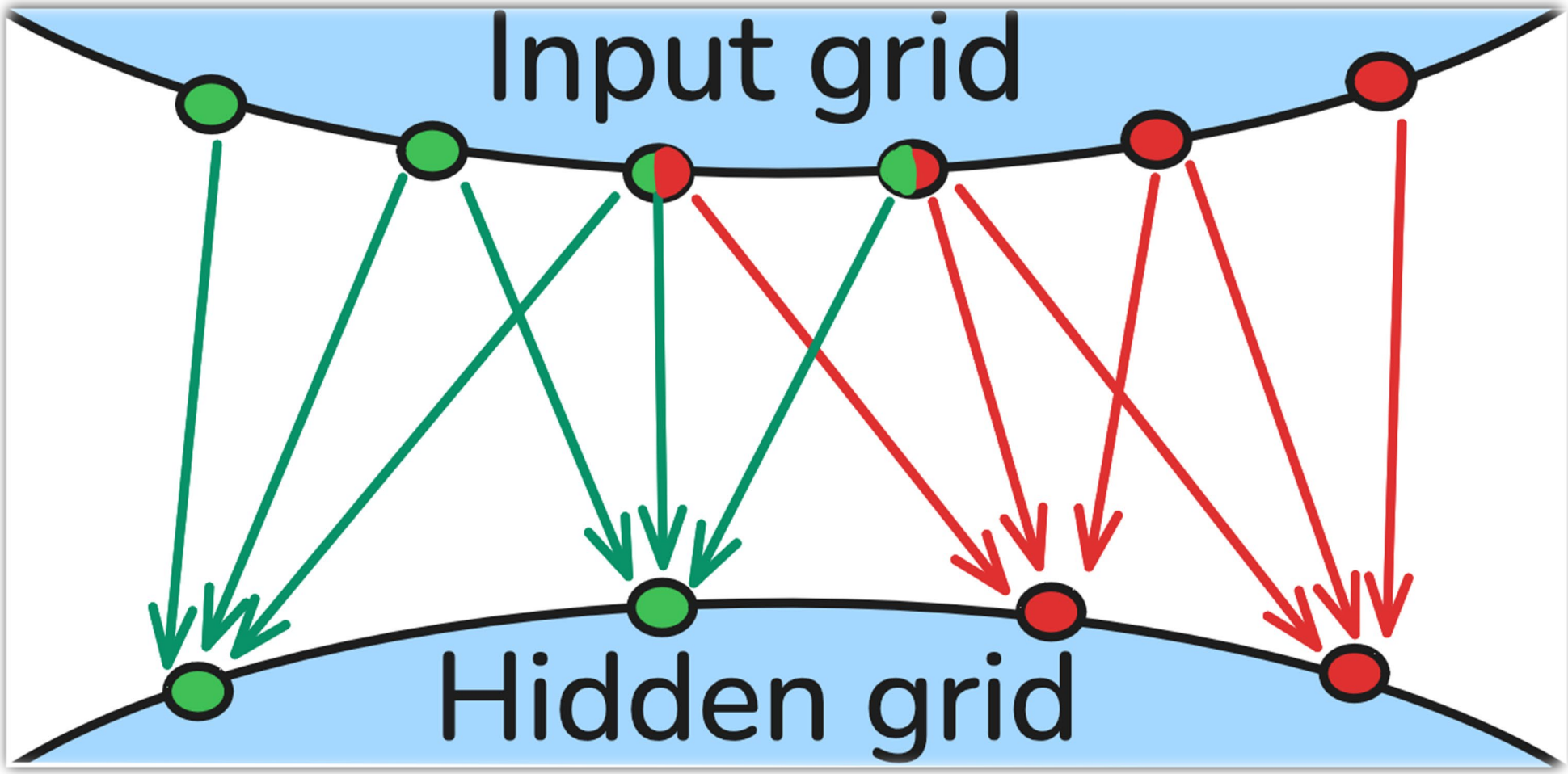
- Data Parallelism: DistributedDataParallel (PyTorch)
  - Distribute training batch across model replicas
  - Aggregate gradients via all-reduce, good scaling :)
  - Limited by batch size :(



- Model parallelism: domain-specific sharding
  - Distribute input data and activations across GPUs
  - Collective communication to handle synchronisation
  - Limited by communication overheads

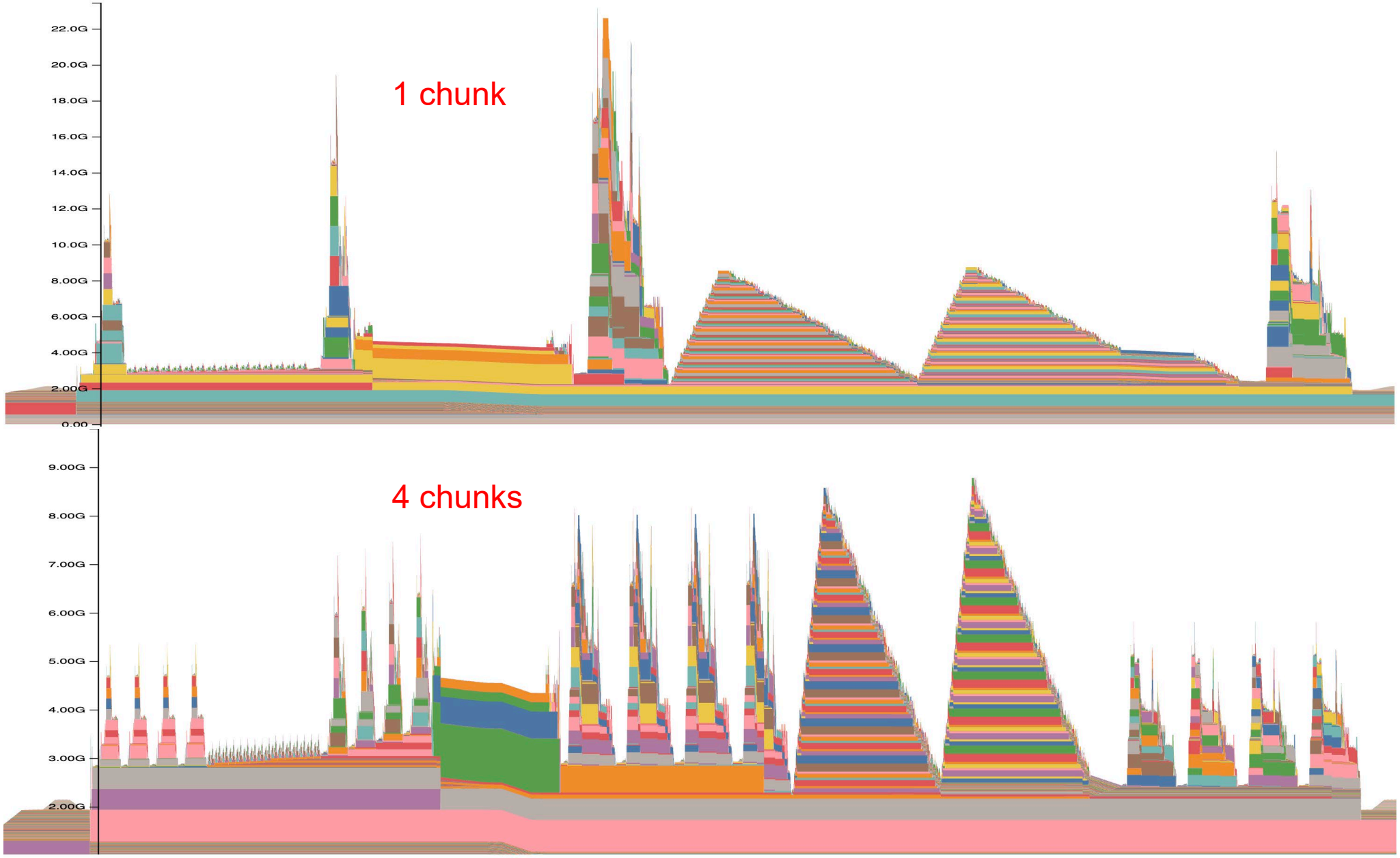


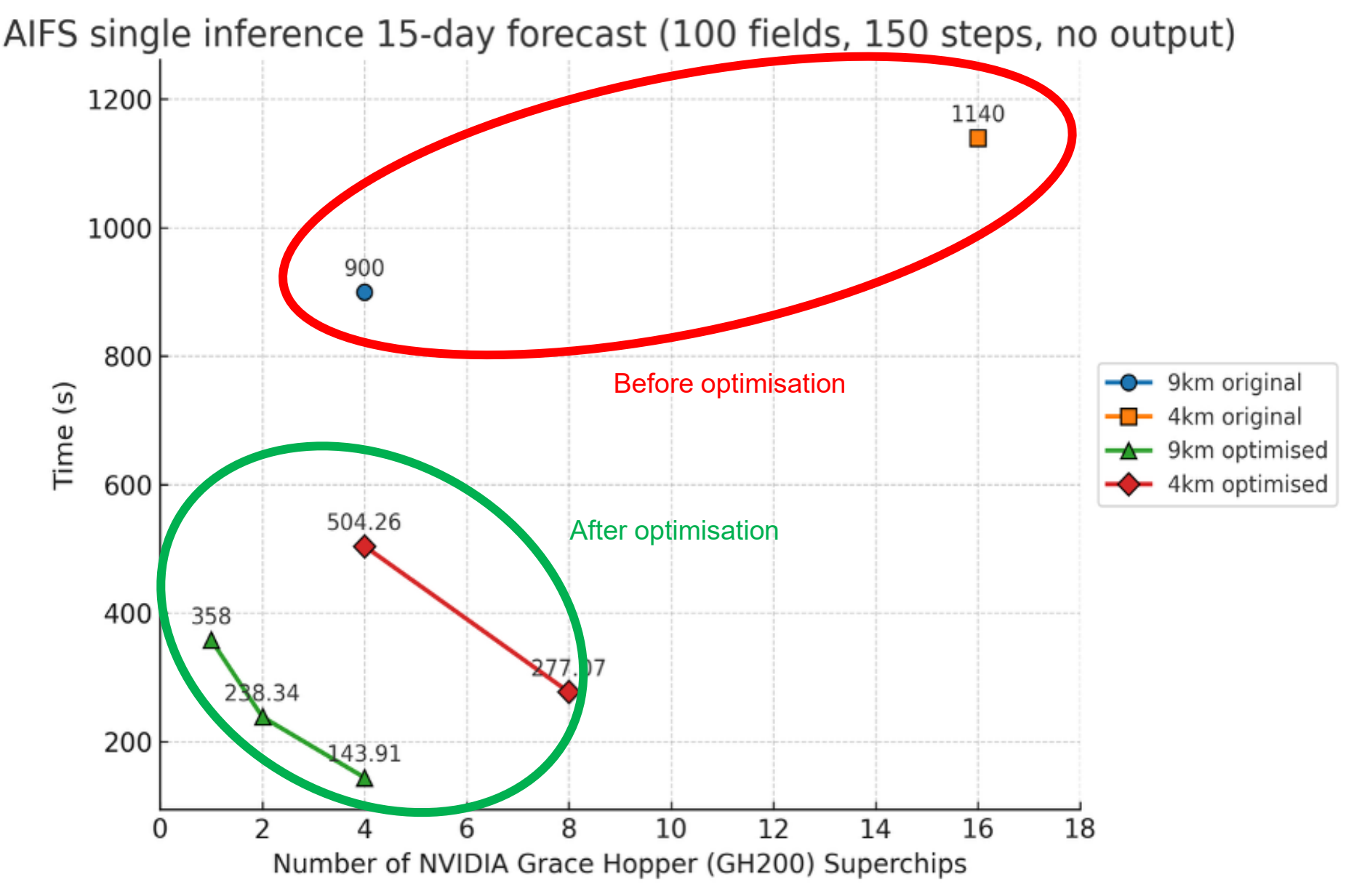
Partitioning the Graph



Same idea enables both multi-GPU parallelism and sequential processing

Reducing Activation Memory




AIFS single inference 15-day forecast (100 fields, 150 steps, no output)
Time (s)
Number of NVIDIA Grace Hopper (GH200) Superchips
1200
1000
800
600
400
200
0
2
4
6
8
10
12
14
16
18
1140
900
Before optimisation
After optimisation
504.26
358
238.34
143.91
277.07
9km original
4km original
9km optimised
4km optimised

Efficient Graph Kernels in Triton

- GraphTransformer kernel
  - flash-attn style online softmax
  - no atomics, aggregate per-node
  - CSC edge layout for optimized memory access patterns
- Fast KNN on regular grids (WIP)
  - fast dynamic graph construction for obs
  - ~10x vs torchcluster

GraphTransformer Triton Kernel

H200

Graph type	PyG	PyG (compiled)	Triton
era_to_h	23.97	6.02	**2.77**
h_to_h	9.64	1.69	**0.83**
h_to_era	46.99	8.16	**4.87**
dop_enc	26.62	6.85	**3.86**

Exemplary impact on training for an o800 lvl7 GraphTransformer model in bf16 on H200s:
pyg compiled: 3.90 s/it → Triton: 2.62 s/it (×1.5 faster)

2x speedup across different graphs

Summary:

Optimising graph operations enables efficiency and scalability across models and tasks.

Next Steps

- fully local communication via halo exchange
- support for observations (dynamic, unstructured)
- explore parameter sharding