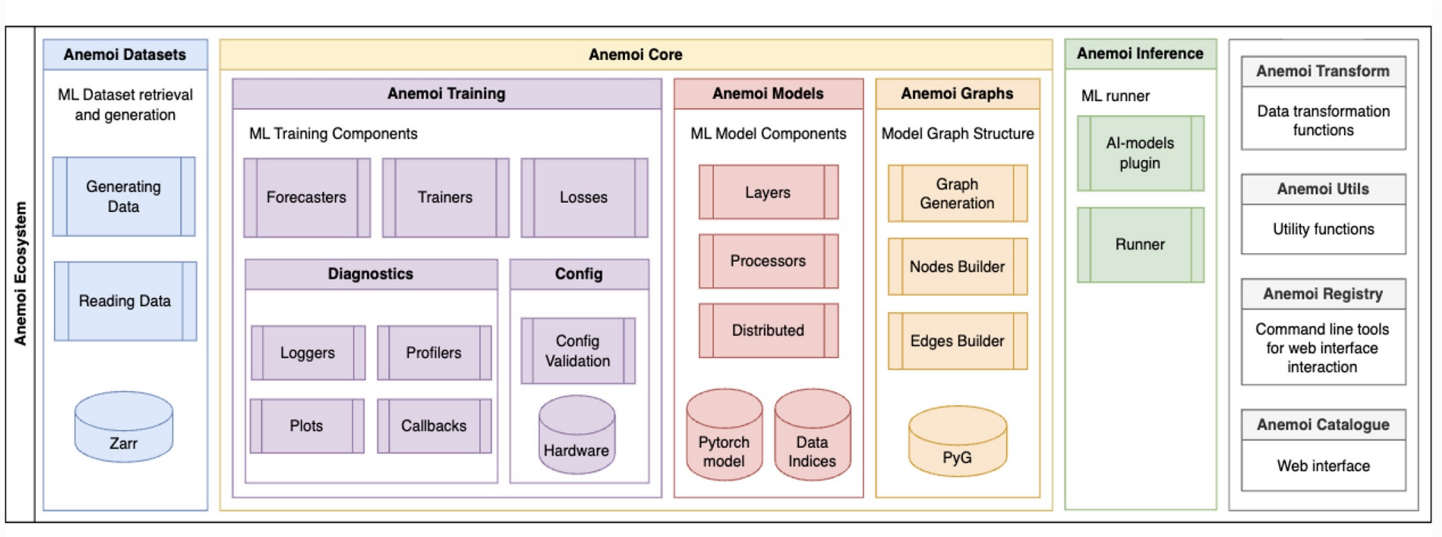
PROFILING DATA-DRIVEN WEATHER MODELS IN ANEMOI

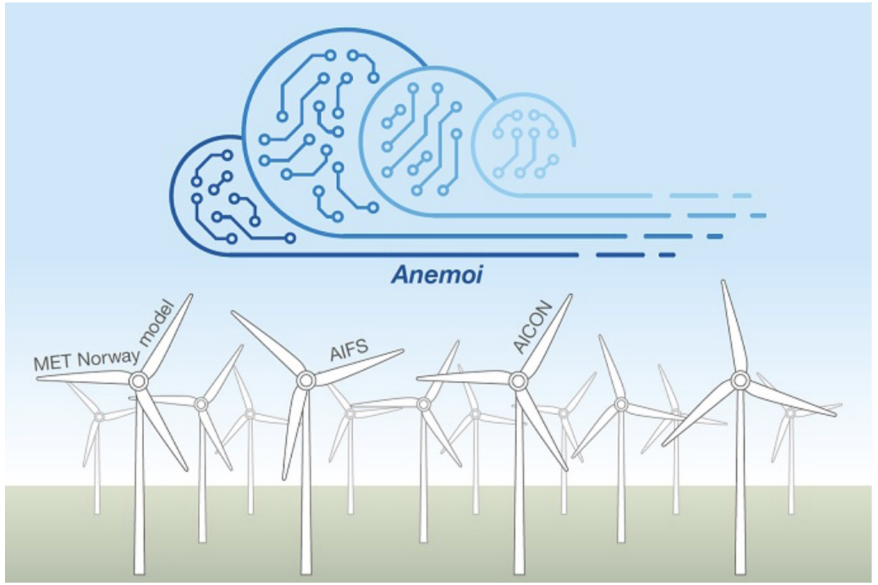
Marieke Plesske – HPC Applications Team (ECMWF)

Funded by the European Union
Destination Earth
implemented by
ECMWF
esa
EUMETSAT

ANEMOI

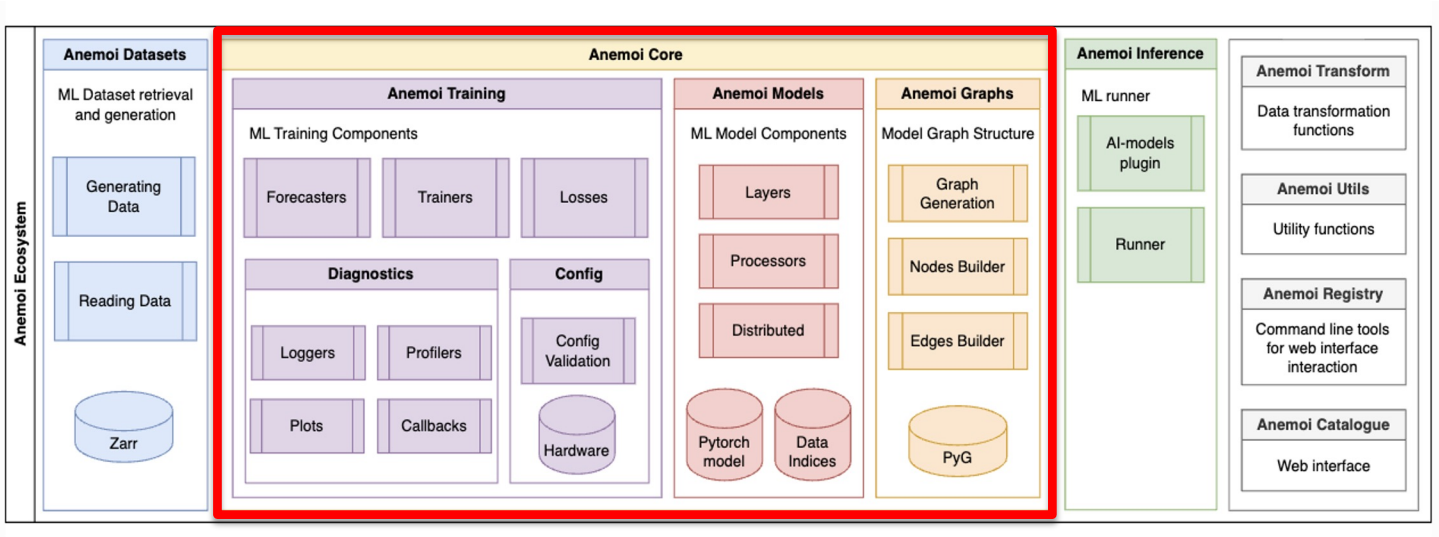
- open-source framework that provides a complete toolkit to develop data-driven weather models – from data preparation through to inference
- highly modular framework organised into different Python packages
- uses i.a. PyTorch, PyTorch Lightning, Hydra, Pydantic, and earthkit

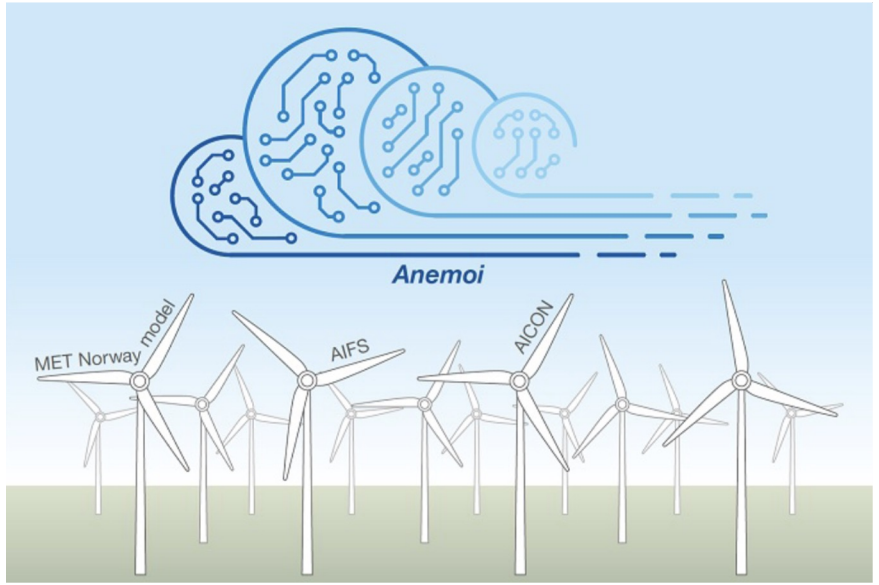


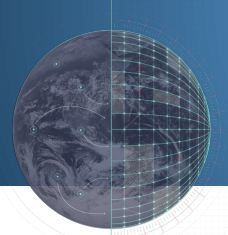


ANEMOI

- open-source framework that provides a complete toolkit to develop data-driven weather models – from data preparation through to inference
- highly modular framework organised into different Python packages
- uses i.a. PyTorch, PyTorch Lightning, Hydra, Pydantic, and earthkit







PROFILING DATA-DRIVEN WEATHER MODELS

What are the bottlenecks/hotspots?

shifting from pure compute → data movement, memory, scaling

- **Data Loading:** high-resolution datasets → increased data volume
- **Memory:** high-resolution global grids + transformer/GNN architectures → large tensors but limited batch size (GPU memory) → ***sharding !***
- **Communication:** (all-reduce, synchronisation) and interconnect bandwidth

- **Workflow imbalance:** mismatch between data loading, compute, and output/postprocessing leads to idle resources
- **Compute hotspots:** expensive ops dominate forward/backward passes

PROFILING TOOLS

- **Anemoi Dashboard:** nightly tests on current anemoi-core main, logging throughput and peak memory usage
- **ML Flow:** system metrics per experiment (disk, cpu, gpu, power usage and utilisation)

- **Nsight/ROCm Systems:** timeline view of the entire application (CPU, GPU, Communication)
- **Nsight/ROCm Compute:** deep per-kernel performance analysis

➢ ***user/developer friendly, anemoi specific, and portable***

- **Anemoi Profiler:** timeline trace of PyTorch- and user-annotated kernels and communication on CPU and GPU, memory snapshots of PyTorch-kernel wise memory allocations on GPU

ANEMOI PROFILER

- wrapper around PyTorch Lightning Profiler
- instances: SimpleProfiler, PyTorchProfiler

- **SimpleProfiler:** reports the mean duration of each action/section and the total time spent over the entire training run

- **PyTorchProfiler**: PyTorch's Autograd Profiler, generates a timeline trace for operations on CPU and GPU

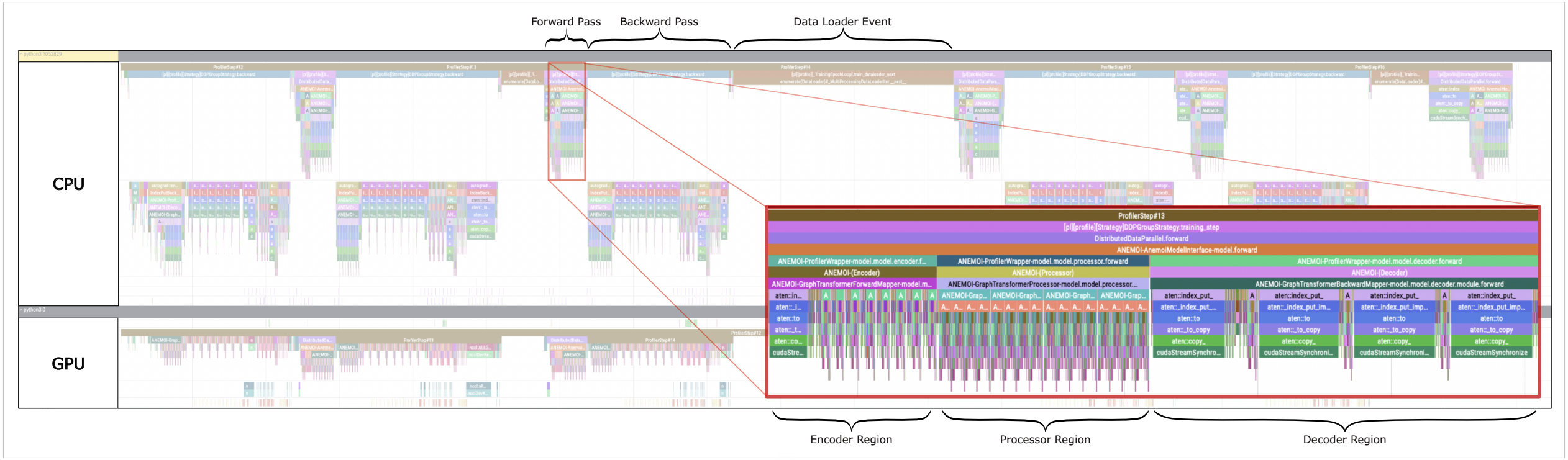
Disclaimer:
All shown time and memory measurements are based on an **artificial benchmark** experiment and **DO NOT** represent real **ANEMOI** or **AIFS** performance!

```
anemoi-training profile –config-name=...
```

timeline traces per rank

TIMELINE TRACE – USER ANNOTATIONS

https://ui.perfetto.dev/



ANEMOI PROFILER – TRACE ANALYSER

```
trace-analyser -i [TRACE DIR PATH] --detailed --plot
```

TRACE ANALYSIS · number of ranks 1 · number of recorded batches 5

Performance overview per rank

Rank	Total (s)	Batches	Throughput (it/s)	GPU idle (%)	Comm (%)
0	14.079	5	0.36	23.79	14.37

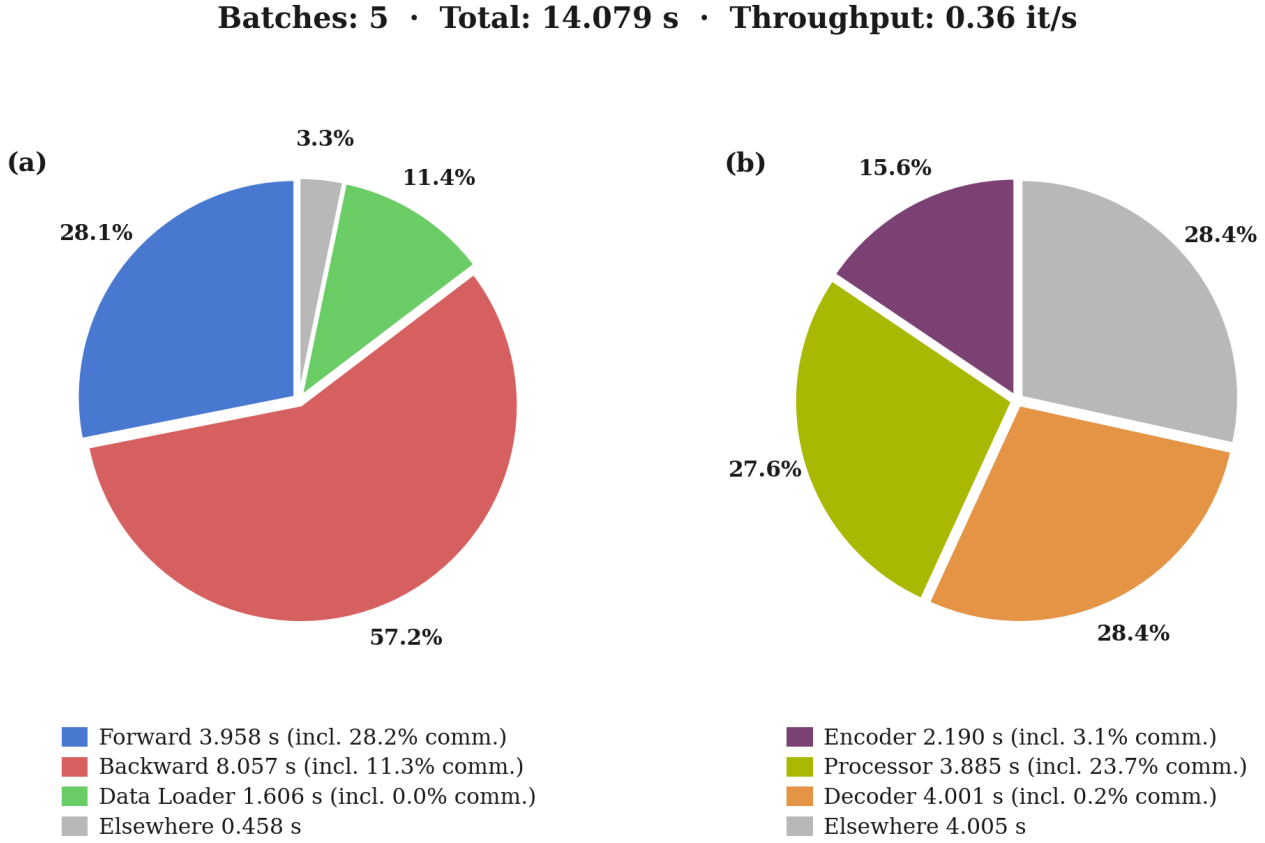
ARCHITECTURE

Trace metadata

Field	Value
Framework	pytorch-lightning
Trace ID	6BC8B85D4FAC4315A02C94D9211F92FE
CUPTI	26
CUDA runtime	12.8
CUDA driver	13.0
Distributed	rank 0, backend=nccl
Devices	4 (NVIDIA GH200 120GB)

Device details

ID	Name	Mem (GB)	CC	SMs	Warp
0	NVIDIA GH200 120GB	95.1	9.0	132	32
1	NVIDIA GH200 120GB	95.1	9.0	132	32
2	NVIDIA GH200 120GB	95.1	9.0	132	32
3	NVIDIA GH200 120GB	95.1	9.0	132	32



ANEMOI PROFILER – TRACE ANALYSER

```
trace-analyser -i [TRACE DIR PATH] --detailed --plot
```

GPU TIME BREAKDOWN

Rank 0

GPU time breakdown

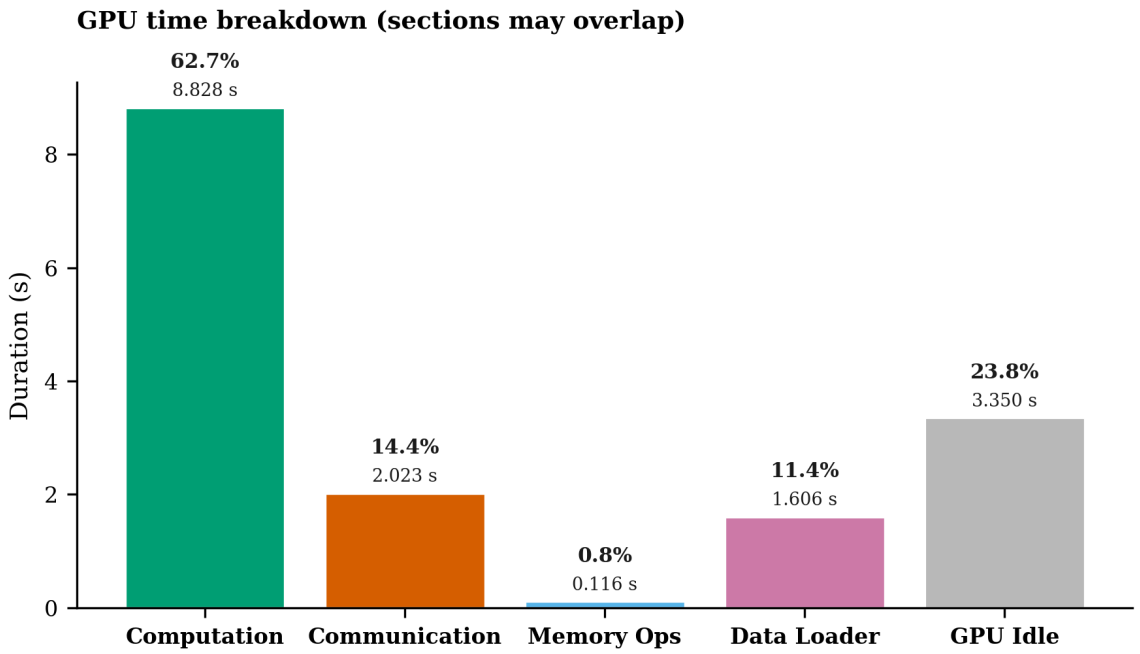
Section	Duration (s)	Per batch (ms)	% of Total
Computation	8.828	1765.68	62.71%
Communication	2.023	404.53	14.37%
Memory Ops	0.116	23.16	0.82%
Data Loader	1.606	321.25	11.41%

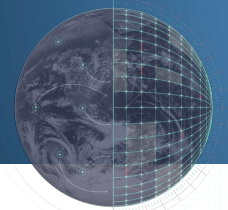
GPU Active: 10.729 s (76.21%) GPU Idle: 3.350 s (23.79%) of total wall time 14.079 s (100%)

Communication overlap: 0.226 s (1.61% of total, 11.18% of comm, 2.56% of comp)

ℹ Sections overlap across streams – percentages may sum to >100%.

Batches: 5 · Total: 14.079 s · Throughput: 0.36 it/s





ANEMOI PROFILER – TRACE ANALYSER

```
trace-analyser -i [TRACE DIR PATH] --detailed --plot
```

DETAILED TRACE ANALYSIS · number of ranks 1 · number of recorded batches 5

Rank 0

GPU time - SUMMARY - rank 0

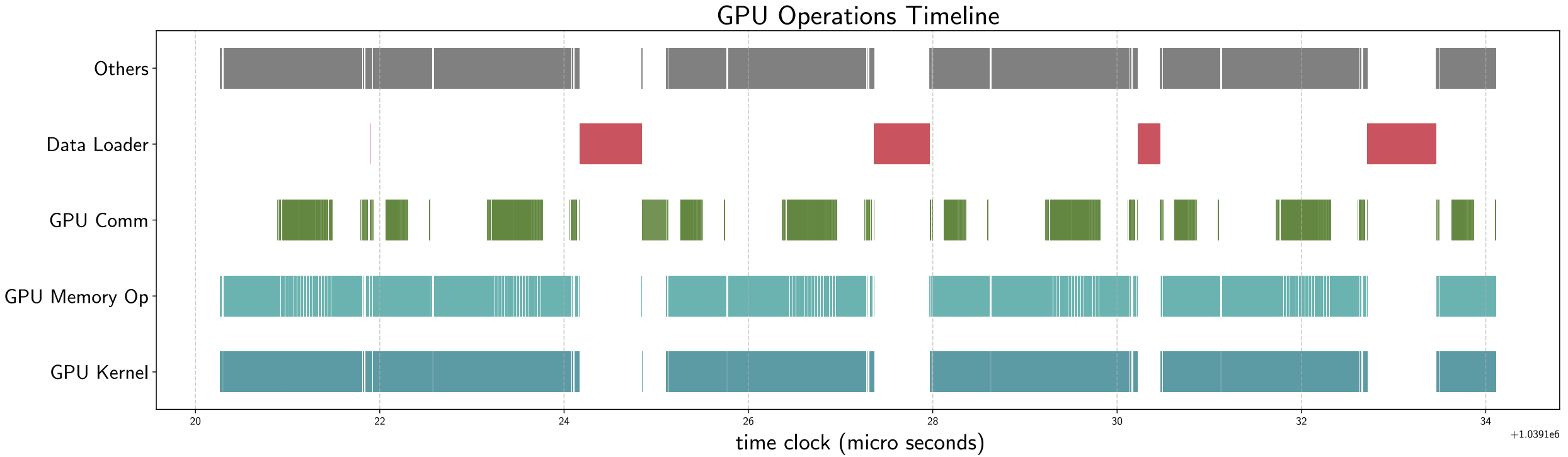
Total active GPU time (user annotated): 9.866 s (70.08%) of total wall time 14.079 s

Method	Total (s)	# Calls	Avg (µs)	Max (µs)	Min (µs)	Overlap %
Linear | model.model.processor.proc.lin_edge | bwd	2.564	160	46700.252	513015.850	19.298	65.689
Sequential | model.model.decoder.data.node_data_extractor | bwd	1.314	155	9396.149	17120.145	1882.209	9.759
Sequential | model.model.encoder.data.proc.node_dst_mlp | bwd	0.698	155	4849.642	11135.357	591.512	7.106
Linear | model.model.decoder.data.proc.lin_edge | bwd	0.655	155	4237.172	8476.076	5.410	0.198
Linear | model.model.encoder.data.proc.lin_edge | bwd	0.555	155	3593.382	7513.788	6.242	0.382
Linear | model.model.decoder.data.proc.lin_value | bwd	0.542	155	3508.381	7055.013	7.554	0.253
Sequential | model.model.decoder.data.proc.node_dst_mlp | bwd	0.500	155	4955.386	6503.087	3335.079	34.915
Linear | model.model.decoder.data.proc.node_dst_mlp | fwd	0.418	320	1306.296	1460.904	1163.533	0.000
Linear | model.model.decoder.data.proc.lin_key | bwd	0.411	155	5093.088	5619.454	4969.673	47.979
Linear | model.model.encoder.data.proc.lin_value | bwd	0.388	155	2616.215	5401.538	8.066	4.219
Top-10 active total (65.73%)	6.485					

ECPEAK

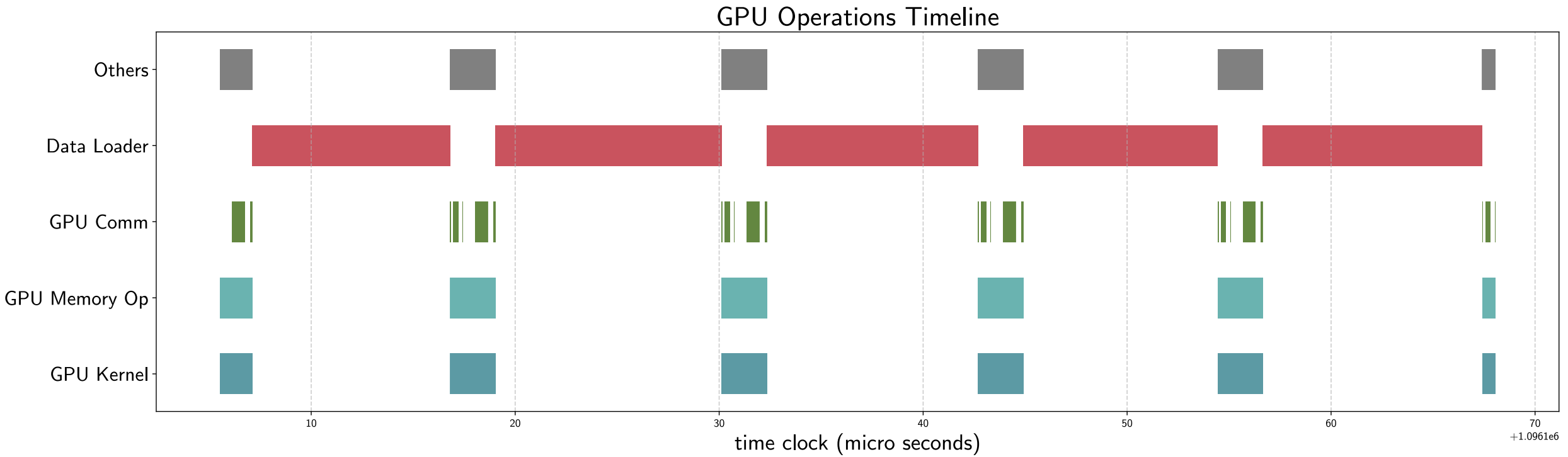
- Luzie Hugger

```
ecpeak batches -i [PATH TO TRACE] -a anemoi -e gpu_batches
```



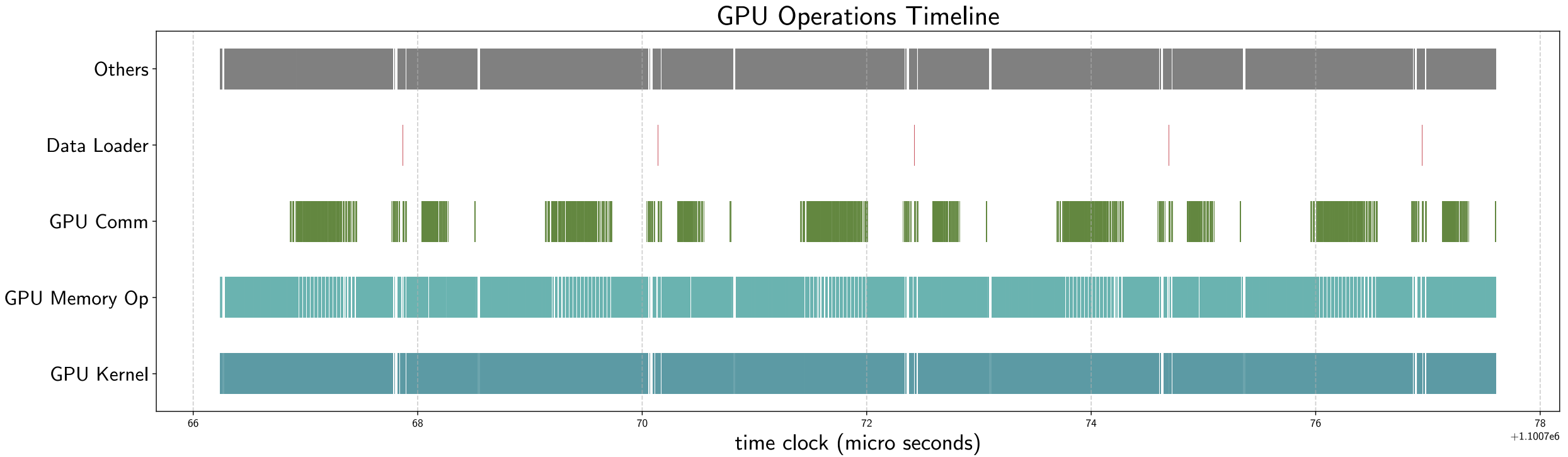
4 data loader

```
ecpeak batches -i [PATH TO TRACE] -a anemoi -e gpu_batches
```



1 data loader

```
ecpeak batches -i [PATH TO TRACE] -a anemoi -e gpu_batches
```



7 data loader

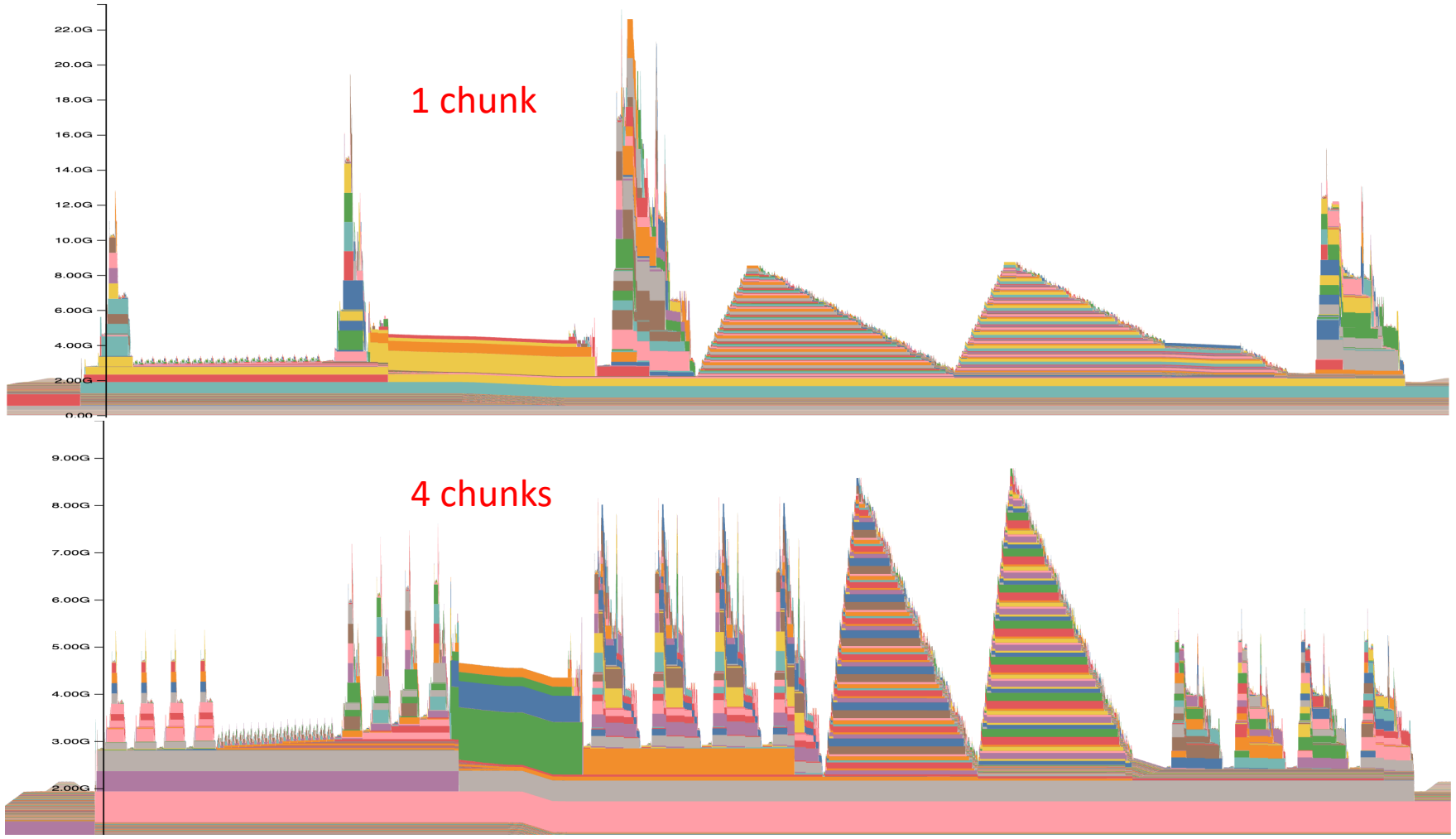
CONCLUSION

- **Multi-level tooling (system → kernel)**
  system profilers, dashboards, internal application/framework profilers, and kernel tools
- **Strong visibility at individual levels**
  e.g. CPU/GPU/Memory/Power usage, time breakdowns, imbalances, GPU idle time, data loader stalling, communication overlaps, bottleneck detection
- **Key gap: missing cross-level integration**
  system - application – framework – kernel
  → e.g. which module in the application causes the most GPU idle time? has the highest communication cost?
- **Incomplete coverage**
  user annotations in the backward pass;
  missing high-level memory summaries (application level)

→ **Future direction:**
extend coverage, connection of all levels, automatic bottleneck detections

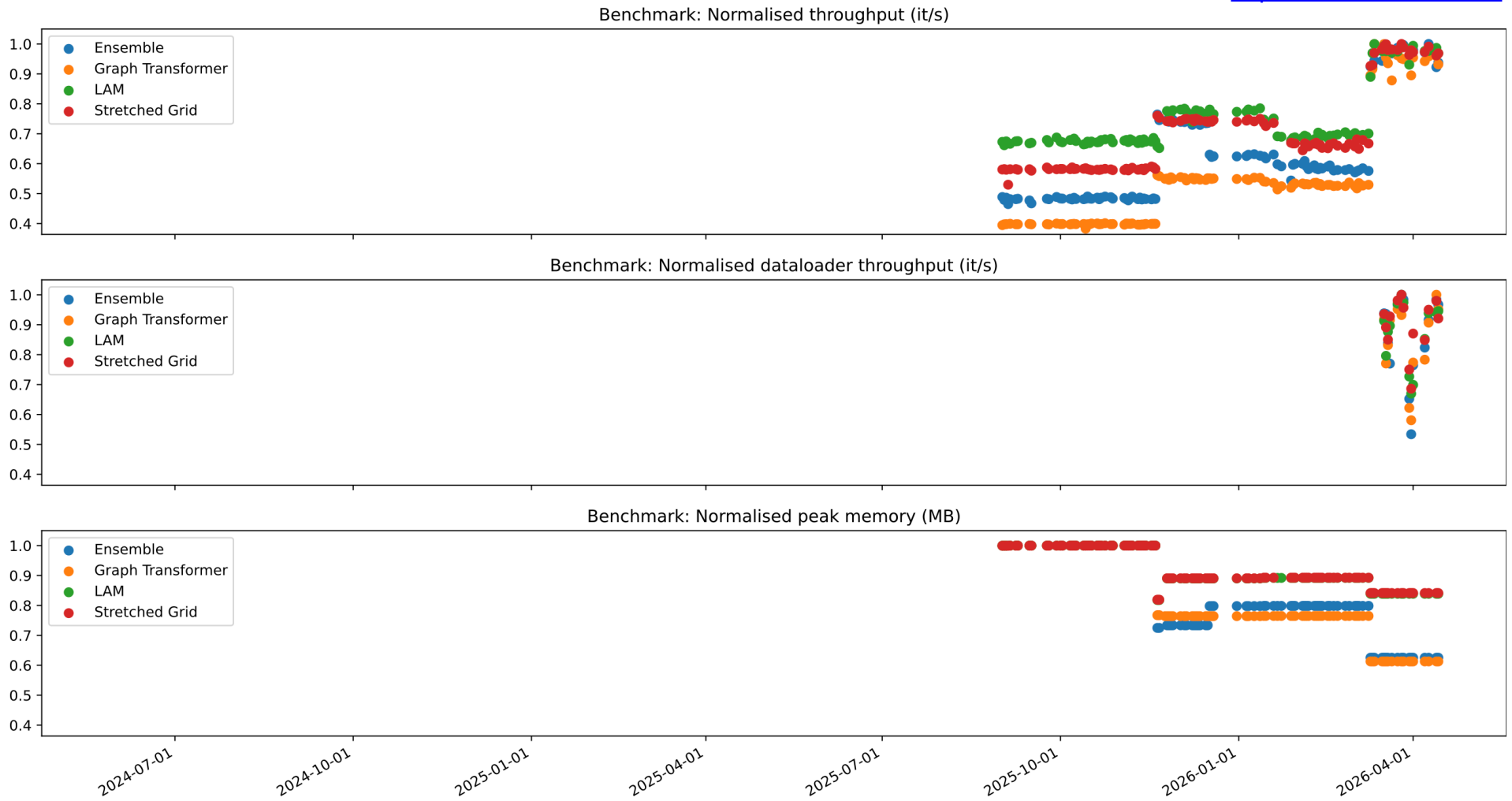
MEMORY SNAPSHOTS

https://docs.pytorch.org/memory_viz



ANEMOI DASHBOARD – BENCHMARKS

https://anemoi.ecmwf.int



ML FLOW

https://mlflow.ecmwf.int