MLSys·im Tutorial — Module 3

Scale, Dollars, and Carbon

Vijay Janapa Reddi

Conference Tutorial

Harvard University

Roadmap: Conference Tutorial

Module	Topic	Status
Module 1	Foundations & Architecture	✓Done
Module 2	Advanced Single-Node Analysis	✓Done
Module 3	**Scale, Dollars, and Carbon**	← **You are here**
Module 4	Design Space Exploration & Synthesis	

TUTORIAL | MODULE 3

Going Distributed

1

The Distributed Computing Taxonomy

When models outgrow a single GPU, we must partition them:

- **Data Parallelism (DP):** Replicate model, shard data.
- **Tensor Parallelism (TP):** Shard matrix multiplications. High communication bandwidth required (NVLink).
- **Pipeline Parallelism (PP):** Shard layers across GPUs. Leads to pipeline bubbles.

The Amdahl Trap

Scaling efficiency is never 100%. It degrades due to:

1. **Communication overhead:** Time spent passing tensors over InfiniBand (AllReduce, AllGather).
2. **Pipeline bubbles:** GPUs sitting idle waiting for previous pipeline stages to finish.
3. **Stragglers:** The cluster is only as fast as its slowest node.

The `DistributedModel` solver automatically calculates network transmission times and pipeline bubbles.

TUTORIAL | MODULE 3

Economics & TCO

The Total Cost of Ownership (TCO)

$$\text{TCO} = \underbrace{\text{CapEx}}_{\text{hw + facility}} + \underbrace{\text{OpEx}_{\text{energy}}}_{\text{electricity}} + \underbrace{\text{OpEx}_{\text{maint}}}_{\text{staff}}$$

The Infrastructure Multiplier: GPUs are only $\sim$40% of total CapEx. Networking (InfiniBand), power delivery, cooling, facility, and operations add 50–150%.

TUTORIAL | MODULE 3

Sustainability & Composition

3

Cross-Domain Carbon Accounting

Training time depends on hardware efficiency; carbon depends on grid intensity.

```python
from mlsysim.solvers import DistributedModel, SustainabilityModel
from mlsysim.systems.registry import Systems
from mlsysim.infrastructure.registry import Infrastructure
from mlsysim.models.registry import Models

fleet = Systems.Clusters.Research_256  # 32x DGX H100 nodes

perf = DistributedModel().solve(
    model=Models.Language.Llama3_70B, fleet=fleet,
    tp_size=8, pp_size=1, dp_size=32)
days = perf.latency.to("days").magnitude

sust = SustainabilityModel().solve(
    fleet=fleet, duration_days=days, mfu=perf.mfu,
    datacenter=Infrastructure.Datacenters.GCP_Iowa)
print(f"Total Carbon: {sust.carbon_footprint:.1f} tonnes CO2e")
```

Geography is the Biggest Lever

Region	Mix	Carbon Intensity (gCO_2/kWh)
Quebec	Hydro	20
Sweden	Hydro+Nuc	25
US Avg	Mixed	390
Poland	Coal	820

The 41× Gap

Training the exact same model on the exact same hardware in Poland emits **41× more carbon** than training it in Quebec.

Summary: Module 3

1. **Distributed Computing:** Tensor Parallelism requires high bandwidth (NVLink); Pipeline Parallelism incurs bubbles.
2. **Economics:** Hardware is only 40% of TCO. Do not ignore the Infrastructure Multiplier.
3. **Composition:** Output from performance solvers (latency) pipes directly into sustainability and economics solvers.
4. **Sustainability:** Geography (Carbon Intensity) is the single biggest lever in AI sustainability.

Next up: Module 4 - Design Space Exploration!