# Reinforcement Learning for LLMs with Intel XPU and PyTorch on ALCF Aurora

Guoqiong Song[2] Filippo Simini[1] Sam Foreman[1] Nathan Nichols[1] Varuni Sastry[1] Samuel Wheeler[1] Khalid Hossain[1] Huihuo Zheng[1] Murali Emani[1] Marieme Ngom[1] Ethan Wong[1] Venkat Vishwanath[1]

[1]Argonne Leadership Computing Facility, Argonne National Laboratory [2]Intel Corporation

## Introducing ALCF Aurora




The **Aurora** exascale supercomputer at Argonne National Laboratory is built on **63,744 Intel Data Center GPU Max 1550 accelerators (XPUs)**: 10,624 HPE Cray-Ex nodes × 6 XPUs, each with **128 GB HBM2e** (8 PB total) and ∼1 TB/s per tile, linked by **8 HPE Slingshot-11 NICs** per node in a dragonfly topology.

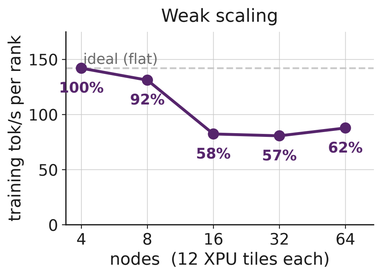
- Aurora is open to **any large-scale open science and engineering project**, and is currently pre-training Argonne's LLM **AuroraGPT**. Hear more in tomorrow's session:

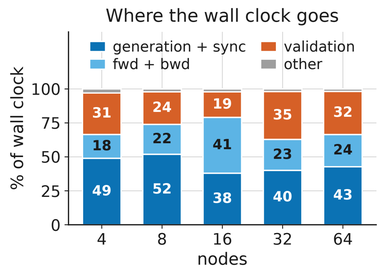
  *Scaling PyTorch on Intel XPU: Field Notes from AI for Science on Aurora*
  17 Oct 21 ⏰ 5:30–5:55 PM 📍 LL20C

- Here we show Aurora's PyTorch XPU ecosystem can run RL **post-training today, at scale, on real science**, with **TRL** and **torchtitan**.

## TRL: scaling GRPO

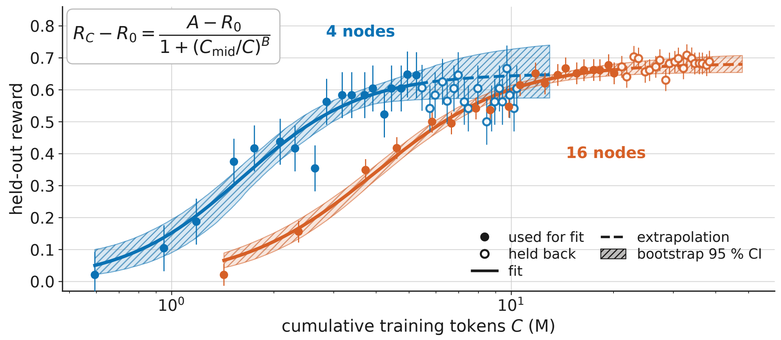
| | |
|---|---|
| model | Qwen3-4B bf16, untied embeddings → 4.4 B trainable |
| task | Countdown — *"use 1, 2, 3 to make 7"* → `1 + 2 * 3` |
| parallelism | FSDP2 `full_shard`, 1 rank per XPU tile (12/node), vLLM generation **co-located on the same ranks** |
| config | `BATCH_SIZE=4`/device, `num_iterations=2`, 50 steps |
| scaling | **weak** — per-device batch is fixed, so global batch and held-out set grow 16× from 4 to 64 nodes |




other = everything after `training_step` returns (optimizer, gradient-norm all-reduce, metric all-gather, dataloader).

## Predicting RL scaling




Reward vs compute is a sigmoid in $\log C$ (*The Art of Scaling RL Compute for LLMs*, arXiv:2510.13786). Following that protocol we fit $A, B, C_{\text{mid}}$ on the **first 100 steps** of the two 200-step runs and predict the rest.

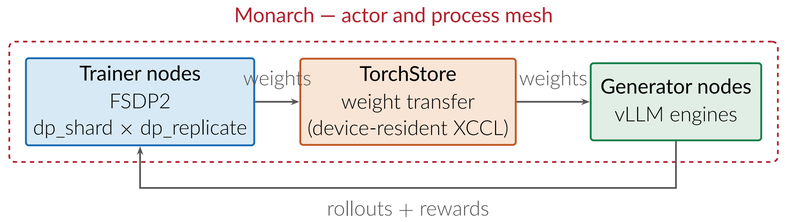
- The reward at 200 steps is predictable at step 100, and the final reward $A$ is the same at both batch sizes: 16 n $A = 0.683$ $[0.655, 0.714]$, 4 n $A = 0.651$ $[0.575, 0.761]$.
- $C_{\text{mid}}$ **grows linearly with nodes**: smaller batches reach the final reward on less compute.
- **Larger batches neither raise the ceiling nor reach it sooner**, for this task and model.

## Porting torchtitan's RL stack to Intel GPUs

**torchtitan's RL core needed no rewrite** — it already routes through device-agnostic helpers. The port concentrated in **three narrow surfaces**: its vLLM worker, TorchStore's transport, Monarch's launch.

| | NVIDIA | Intel XPU |
|---|---|---|
| Collectives | NCCL | oneCCL / XCCL |
| vLLM worker | `GPUWorker` rejects non-CUDA | XPU worker and model-runner subclasses, isolated so CUDA installs never import them |
| Attention | `CUSTOM` ≈ FlashAttention | overridable config, so vLLM picks its XPU-native backend |
| Device mask | CUDA_VISIBLE_DEVICES | ZE_AFFINITY_MASK, resolved before torch is importable |
| Weight transfer | Gloo / NIXL / RDMA | device-resident XCCL transport on ProcessGroupXCCL |

## torchtitan + Monarch + TorchStore

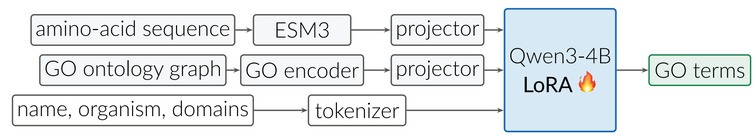



- **Disaggregated**: trainer and generators sit on *different* nodes and are sized independently.
- **torchtitan's full parallelism mesh** is available (`dp_shard`, `dp_replicate`, `tp`, `cp`, `pp`).
- **Asynchronous rollouts**: the controller keeps several rollout groups in flight, overlapping generation with training.

## Science case: RL for a *multimodal* model

*BioReason-Pro (Advancing Protein Function Prediction with Multimodal Biological Reasoning, bioRxiv 2026.03.19.712954)* predicts **Gene Ontology** (GO) terms for a protein:

*"protein* `MSSSAGSGHQ…`*, human, 120 residues"* → *cytoplasm* (`GO:0005737`), *neuron projection* (`GO:0043005`)




- It is **not a text model**: two encoders feed the LLM *non-text vectors* alongside tokens: frozen **ESM3** encodes the sequence, and a constant 200-row tensor describes the ontology.
- The spliced embedding matrix reaches the vLLM engine as `prompt_embeds`, and each step syncs only the LoRA weights.

## Need compute for science? Run on Aurora!

Get time on Aurora

Start with a DD award

**Acknowledgment.** This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.



fsimini@anl.gov, guoqiong.song@intel.com