



CONFERENCE

—NORTH AMERICA 2026—

Scaling PyTorch on Intel XPU

Field Notes from AI for Science on Aurora

Sam Foreman · Panagiotis Kourdis · Tanim Dey

Scaling PyTorch on Intel XPU

Field notes from AI for Science on Aurora

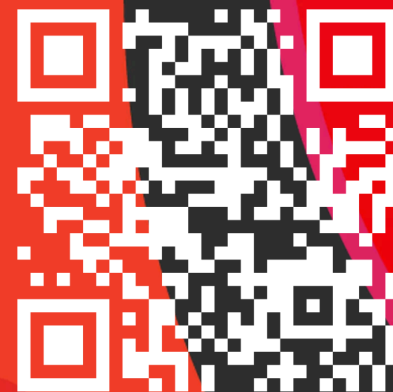
Sam Foreman · Argonne National Laboratory

Panagiotis Kourdis · Intel

Tanima Dey · Intel

PyTorch Conference · October 21, 2026 — San Jose

Interactive version:
live diagrams + links



samf.sh/talks/2026/10/21

Aurora at ALCF

ALCF	Argonne Leadership Computing Facility at Argonne National Lab : DOE open science on Aurora + Polaris
Aurora	10,624 nodes · 127,488 XPUs · system paper
This talk	Running PyTorch on Aurora : backend fixes, training, and recovery



Fig: Aurora at the ALCF. Photo: Argonne / ALCF.

Here, XPU = one GPU tile exposed as a `torch.xpu` device in FLAT mode. 12 XPUs per node.

Talk outline

Part	Question	Implementation
A • Needed	What does it take to build a GPU that runs PyTorch? <i>What about 100,000 of them, together?</i>	~~IPEX~~ (legacy) • torch-xpu-ops • XCCL / torch-ccl oneAPI • oneCCL • SYCL
B • Accomplished	How do you run AI for Science campaigns at scale? <i>On a brand-new, rapidly evolving software stack?</i>	ezpz • Megatron-DeepSpeed • torch/titan/ezpz DTensor • FSDP2
C • Remains	What keeps a 14-day job alive at 60k → 120k XPUs?	Nobody knows! Best bets: ezpz checkpoint-restart • watchdog • node exclusion • scheduler resubmit

Scientific AI on Aurora

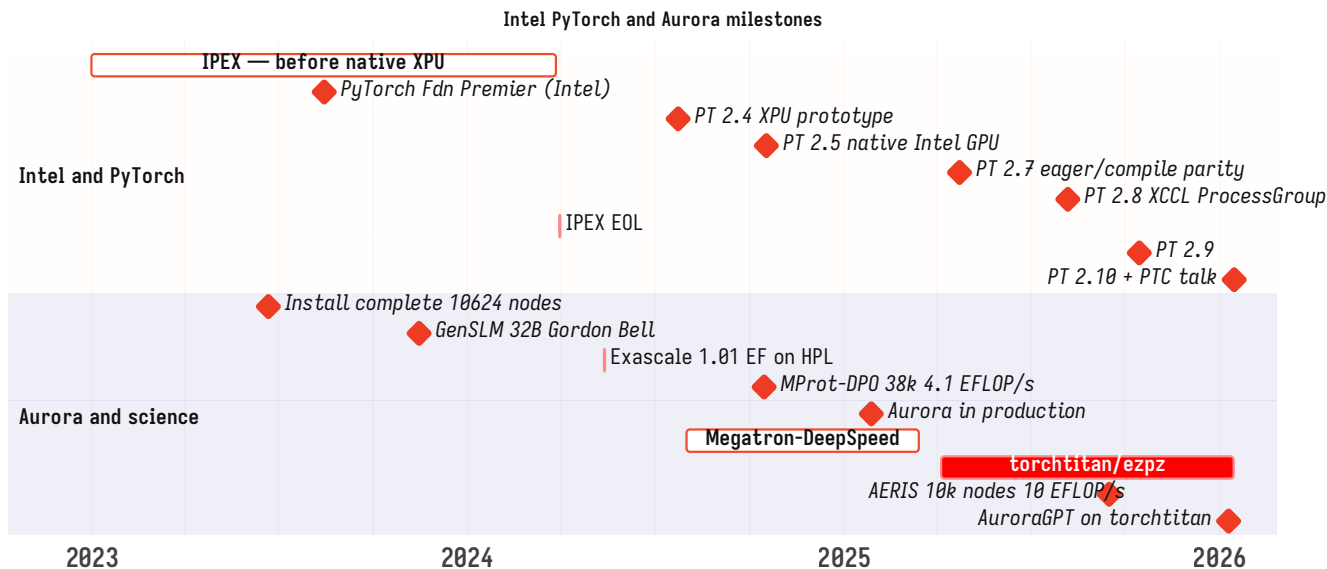
Workload	When	Scale / result
GenSLM: viral genomes	SC22 · Gordon Bell Prize ¹	Polaris + Cerebras
MProt-DPO: protein design	SC24 · Gordon Bell finalist	38,400 ranks · 4.11 EFL0P/s at 3,200 nodes
AERIS: weather diffusion	SC25 · Gordon Bell finalist	10,080 nodes · 10.21 EFL0P/s · 90-day stable rollouts
AuroraGPT: foundation LLM	ongoing	7.77T-token MDS reference → torchtitan

I co-led MProt-DPO and AERIS, and co-lead AuroraGPT.

ezpz: same training program on Intel, NVIDIA, AMD.²

1. The ACM Gordon Bell Prize is awarded each year at SC for outstanding achievement in high-performance computing. GenSLM won the 2022 Special Prize for HPC-Based COVID-19 Research. MProt-DPO was a 2024 finalist; AERIS was a 2025 finalist for the Special Prize for Climate Modeling.
2. ALCF Aurora · ALCF PyTorch tile-as-device configuration · ALCF support: Aurora · torchtitan/ezpz · AuroraGPT companion.

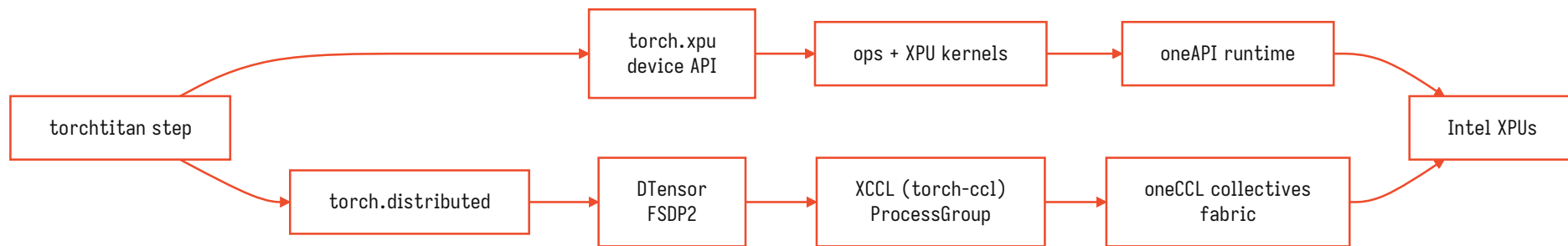
PyTorch XPU and Aurora timeline



- **Intel and PyTorch:** IPEX → PT 2.4–2.10+XPU (4 releases)
- **Aurora and science:** 10k nodes → 120k-XPU workloads

From **ezpz: distributed PyTorch across any hardware** plus verified Aurora dates.

PyTorch XPU backend components



Test the device API, kernels, and collectives together. IPEX shims →

native torch.xpu + torch-ccl / XCCL + oneCCL

XPU integration issues

Symptom	Root cause	What we run now
<code>torch 2.13</code> : <code>expected DTensor</code> , got <code>plain tensor</code> on FSDP wrap	Missing <code>pytorch#181519</code>	Fixed in <code>2.14.0+xpu</code>
<code>DeviceMesh</code> → <code>does not support splitting</code>	<code>ProcessGroupXCCL</code> inherits <code>supportsSplitting() == false</code> (see <code>pytorch#186548</code>)	Local shim · <code>pytorch#186548</code>
<code>train_timeout_seconds</code> is a no-op	<code>_set_pg_timeout</code> has no <code>xpu</code> branch: <code>XCCL</code> stores the timeout, never arms it	Watchdog heartbeat (Act C) + two-PR upstream

The `CUDA` path worked. The `XPU` path inherited defaults that needed fixing.

From Megatron-DeepSpeed to torchtitan

Megatron-DeepSpeed fork ³	<code>torchtitan</code> + DTensor + FSDP2
2B trained to 7.77T tokens (Aug talk)	Same run continues on <code>torchtitan</code> : 4.67T at 256 → 512 nodes
Parallelism baked into the fork	<code>DTensor</code> + <code>FSDP2</code> as composable primitives
Every PT release → rebase + patch audit	XPU integration in separate layers · <code>torchtitan/ezpz</code>

The fork trained the 2B model to 7.77T tokens. Each PyTorch upgrade required another rebase and patch audit.

3. Megatron-DeepSpeed · torchtitan upstream · AuroraGPT companion: MDS training.

One training program across vendors

Hello world:

python

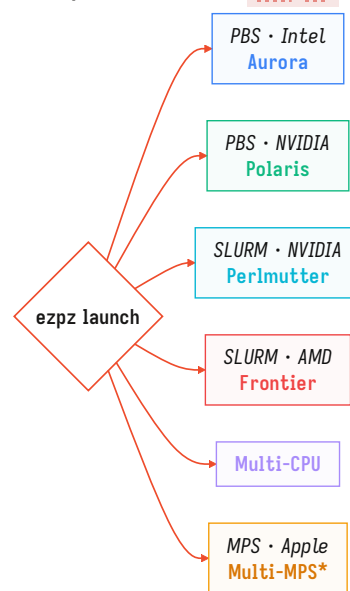
```
# example.py
import torch
import ezpz
rank = ezpz.setup_torch()
world_size = ezpz.get_world_size()
dev = ezpz.get_torch_device()      # {xpu, cuda, cpu, ...}
# FSDP2 by default - accepts DeviceMesh
model = ezpz.wrap_model(torch.nn.Linear(128, 10).to(dev))
print(f"Hello from {rank}/{world_size}!")
```

bash

```
$ ezpz launch python3 example.py
Hello from rank 0/24!
Hello from rank 1/24!
# ... etc
```

► Other examples

Same command on every machine: **ezpz** picks the backend.



* `mccl` (`mps-ddp/mccl`): MPS on Apple silicon, so `ezpz launch` works even on a laptop.

Benchmark the training step

1. Fix the build, model, batch size, precision, and node count.
2. Run the same training step on each backend.
3. Measure step time, tokens/s, collective time, and memory.
4. Increase node count. Rerun after kernel or collective changes.

An op test can pass while the distributed training step hangs.

Check loss and throughput at each node count. Test timeout and recovery behavior separately.⁴

4. PyTorch profiler (`torch.profiler`) • PyTorch `torch.xpu`

Changes in the torchtitan training stack

Layer	Before (MDS fork + <code>ezpz</code>)	After (<code>torchtitan/ezpz</code>)
Parallelism	Custom tensor-parallel + pipeline in fork	<code>FSDP2</code> + <code>DTensor</code> sharding
Collectives	DeepSpeed's <code>NCCL</code> assumptions	<code>XCCL</code> (<code>torch-ccl</code>) → <code>oneCCL</code>
Kernels	Fork-local fused kernels	XPU kernels via <code>torch.xpu</code> : <code>layernorm+residual</code> moved; <code>attn</code> / <code>MLP</code> / <code>optimizer</code> remain ⁵
Launch	Already <code>ezpz</code> : <code>scheduler</code> → <code>nodes</code> → <code>MPI</code>	Same <code>ezpz</code> ⁶ : <code>launch</code> + <code>checkpoint-restart</code>
Checkpoint	<code>DeepSpeed</code> engine save	<code>DCP</code> (<code>torch.distributed.checkpoint</code>)

`ezpz` stayed in place. Parallelism, collectives, and checkpoints moved out of the fork.

- `layernorm+residual` landed in `torch.xpu`; `attn` / `MLP` / `optimizer` remain — composable kernels must survive `torch.compile`.
- `ezpz` is the cross-workload launcher: `qsub` / `sbatch` → `node list` → `MPI` on Aurora+Sunspot/Polaris. Same before and after the framework switch.

Testing at every layer

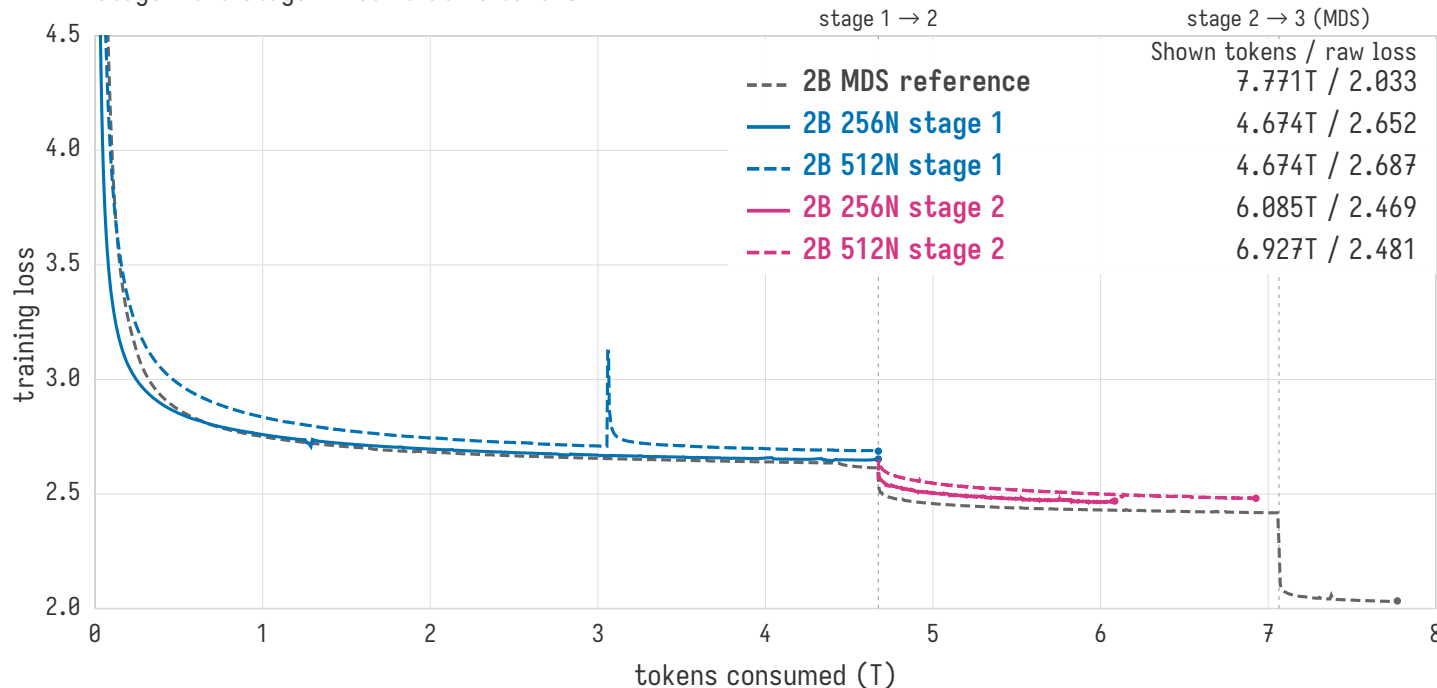
Layer	Gate	Link
Numerics	50-step loss: ± 0.10 on final + last-10 mean	Production loss figure
Collectives	XCCL ladder 48 $\rightarrow$ 768 ranks, value-checked	ezpz/test_dist.py
Parallelism	TP1 / TP2 / MoE on Aurora; parity on Sunspot	experiments/ezpz
Portability	One gate on Aurora + Sunspot + Polaris + Perlmutter	ezpz
Orchestration	Failover harness: 9 fixtures	Failover writeup

The streaming loader contributes ~ 0.05 loss variation from shuffle order. The 50-step check uses a ± 0.10 tolerance.

AuroraGPT: production training progress

AuroraGPT 2B: production training progress

Stage 1 and stage 2 · cumulative tokens



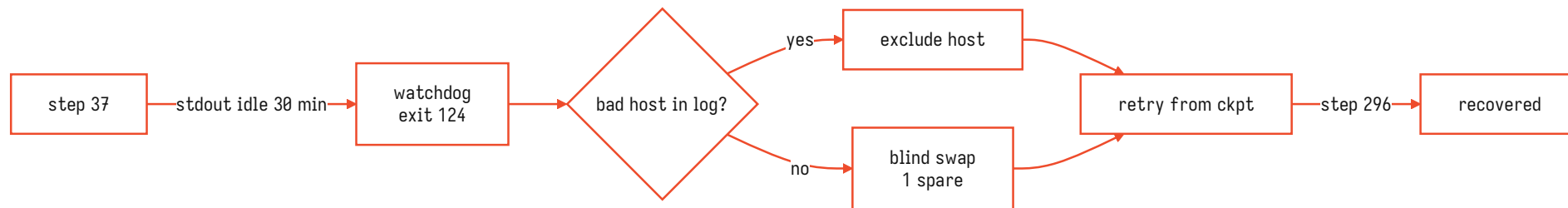
Solid: 256 nodes · dashed: 512 nodes. Stage 2 includes stage-1 tokens; training mixes differ.

Stage 2: **6.927T** cumulative tokens on 512 nodes · **6.085T** on 256 nodes.

Production runs and training logs · Paper forthcoming at the TPC workshop, SC26.

Detecting and recovering from stalled jobs

Job 8505298 : step 37, then silence. No traceback. Watchdog at 30 min (exit 124), wrapper swapped a node, relaunched. Attempt 2 reached step 296.



`_set_pg_timeout` has no `xpu` branch: timeout is stored, never enforced.⁷

7. Failover incident: 8505298

Failure modes at ~120k XPU

Mode	Signal	What helps	What doesn't
Silent hang	Stdout stalls, no error	Watchdog on log progress	<code>_set_pg_timeout</code> (<code>xpu</code> never arms)
Lost collective	Rank times out in <code>XCCL</code>	Node exclude + retry	Retrying on the same host
Crash / OOM	Traceback + exit code	Process restart from <code>ckpt</code>	Scheduler resubmit alone
Stale FS / DCP	Checkpoint read stalls	Async save + local staging	Sync save on shared <code>FS</code> at 10k ranks

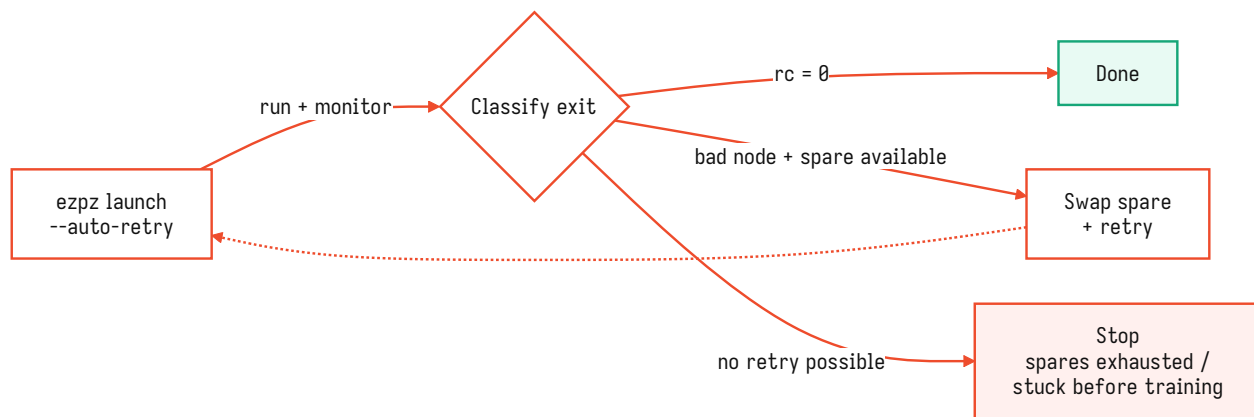
Test node exclusion, checkpoint reload, and scheduler resubmission before a long run.⁸

8. Failover incident: 8505298 • [ezpz docs](#)

Process, node, and allocation restarts

Layer	Scope	Trigger	Cost of being wrong
Process	1 rank	Crash / OOM	Tokens since last ckpt
Node	12 XPU	Hang on one host	1 node's work + re-shard
Allocation	10k nodes	Bad FS / scheduler	Whole-job walltime

ezpz handles node and allocation restarts:
checkpoint-restart + node exclude + scheduler resubmit. DCP writes the checkpoints.
Test each restart path before committing a full allocation.



Exit trailer + crash patterns + idle watchdog (exit 124) determine the next action.

Remaining XPU gaps

Layer	Gap	What's needed
Kernels	Fused attn + MLP + optimizer across XPU	Composable kernels compatible with <code>torch.compile</code>
Collectives	Tuning, diagnostics, real timeouts	<code>XCCL</code> timeout enforcement + <code>DTensor</code> split fix upstream
Checkpoints	Async <code>DCP</code> at 10k ranks without <code>FS</code> collapse	Local staging → async flush
Operations	Detect bad nodes during the run	Heartbeat at the <code>ProcessGroup</code> level, not <code>stdout grep</code>

Run the training step, save a checkpoint, interrupt the job, and resume it.

Bringing up the next accelerator

1. Test compute and collectives together.
2. Use the same training program on each machine.
3. Compare loss against the reference run.
4. Measure full training steps at scale.
5. Interrupt, restart, and check the resumed loss.

Links and further reading

Stack	Link
ezpz	Launch · Checkpoint · All docs
torchtitan/ezpz	Experiments · Failover writeup
PyTorch XPU	torch.xpu · DTensor · FSDP2 · DCP
Distributed	torch-ccl / XCCL · oneCCL · torch.distributed
System	Aurora · Aurora system paper · Scaling MPI on Aurora · ALCF support: Aurora · ALCF news: 120k-tile scale
Science	AuroraGPT companion · AERIS · MProt-DPO · GenSLM

Further reading: [ezpz](#) portability · [Launching 50k processes](#) · [Data and training at ALCF](#)

Acknowledgements

This work used resources of the **Argonne Leadership Computing Facility**, a DOE Office of Science user facility at Argonne National Laboratory, supported under contract **DE-AC02-06CH11357**.

Aurora was built in partnership with **HPE** and **Intel**: Intel Max 1550 XPU (`PVC`), `oneAPI` / `oneCCL` / `XCCL` , and the upstream `torch.xpu` work tracked in Act A. Thanks to the ALCF operations and support teams, the `PyTorch` and `torchtitan` maintainers, and the science teams behind `GenSLM` , `MProt-DPO` , `AERIS` , and `AuroraGPT` .

Facility	ALCF · Aurora · Polaris
Partners	HPE · Intel (Max 1550 / <code>oneAPI</code> / <code>oneCCL</code>) · PyTorch Foundation
Teams	<code>GenSLM</code> · <code>MProt-DPO</code> · <code>AERIS</code> · <code>AuroraGPT</code> — see Links for papers

Aurora allocation programs

Anyone (!) can apply for time on any of the **ALCF** systems, including our **AI Testbed**!

Program	Best for	Cadence
INCITE	Large, peer-reviewed science campaigns	Annual
ALCC	DOE-mission-aligned projects	Annual
Director's Discretionary (DD)	Porting, benchmarking, getting started — start here	Rolling, weeks

For a **DD** proposal, find me after the talk: foremans@anl.gov / [@saforem2](https://twitter.com/saforem2).

[ALCF allocations overview](#) · [DD program details](#) · [ALCF support center](#)

Next at ALCF: Solstice and Equinox

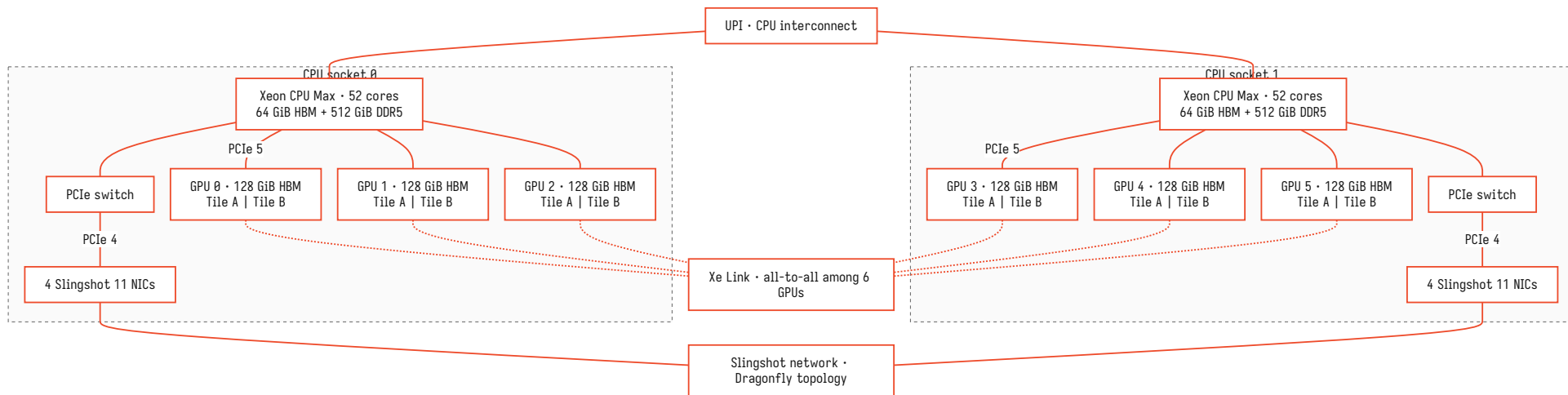
Announced with NVIDIA and Oracle:

System	Planned accelerators
Equinox	10,000 NVIDIA Blackwell GPUs
Solstice	100,000 NVIDIA Blackwell GPUs

The hardware, fabrics, launchers, networks, and file systems keep changing.
The same training code still needs to launch, communicate, checkpoint, and recover at scale.

NVIDIA / Oracle / DOE announcement • October 2025

Backup: inside an Aurora node



6 GPUs × 2 tiles = 12 torch.xpu devices in **FLAT** mode. GPU Max 1550 (PVC), 768 GiB GPU HBM per node.

Logical schematic, not a wiring diagram. Xe Link connections grouped. [ALCF node topology](#) · [Aurora system paper](#) · [Machine overview](#) · [PyTorch tile configuration](#)

Session feedback

Scan to share feedback with us and the organizers.



Scaling PyTorch on Intel XPU: Field Notes from AI for Science
on Aurora