Sandia National Laboratories

*Exceptional service in the national interest*

# Federated LLM Training Across NNSA Labs


Max Carlson

*Sandia National Laboratories*


TPC, June 3, 2026


U.S. DEPARTMENT of ENERGY | NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2026-21930C

# MOTIVATION

- Numerous DOE data sets are not accessible to large LLMs
  - DOE proprietary data isn't on the internet.
  - Limits the usefulness of open-weight and proprietary models on DOE-relevant queries.
- Proprietary datasets are also not easily shareable, but the knowledge is useful.
  - This limits the utility of RAG-based approaches.
- Want to share the model without sharing the data
  - Can federated training be a path forward?

**DOE NNSA/Tri-lab:** Sandia National Laboratories (**SNL**), Lawrence Livermore National Laboratory (**LLNL**), and Los Alamos National Laboratory (**LANL**)

# BASE MODELS & DATASETS

- Base models: The Llama 3.X family at 1B, 8B & 70B sizes. Expanded to Nemotron 3 Nano (30B)
  - Text-only models (for now).
  - Data cut-off date (for Llama 3.x): December 2023.

- Dataset #1: Arxiv since January 2024 (~2B tokens)
  - Pain points: Equations, figures, references, macros.
  - Goal: Allows us to evaluate quality of federated model vs. monolithic approach.

- Dataset #2: Actual data from three labs (~150M tokens at SNL for our initial set).
  - This is *not* evenly balanced across the three labs.

# TECHNICAL APPROACH: LOCAL TRAINING (BACKEND)

- DOE has substantial compute resources, including El Capitan (#1 on the June 2025 Top 500).
  - LLNL's El Capitan uses AMD MI300A.
  - SNL / LANL machines use NVIDIA H100 accelerators.

- Initial backend approach: Meta's Torchtitan (https://github.com/pytorch/torchtitan)
  - Portable (works on NVIDIA & AMD).
  - Works with slurm/flux for multi-node HPC systems.
  - Supports FSDP2, tensor, pipeline and context parallelism.
  - Verified compatibility with Llama 3.X models (including 405B).
  - Meta reports scaling up to 512 nodes (8 GPUs/node).

- Expanded backend approach: Nvidia's Nemo-RL
  - Reinforcement learning using LLM judge
  - Better model performance but much higher training cost


Image from: https://hpc.llnl.gov/hardware/compute-platforms/el-capitan

# TECHNICAL APPROACH: FEDERATION

**Swarm Learning**

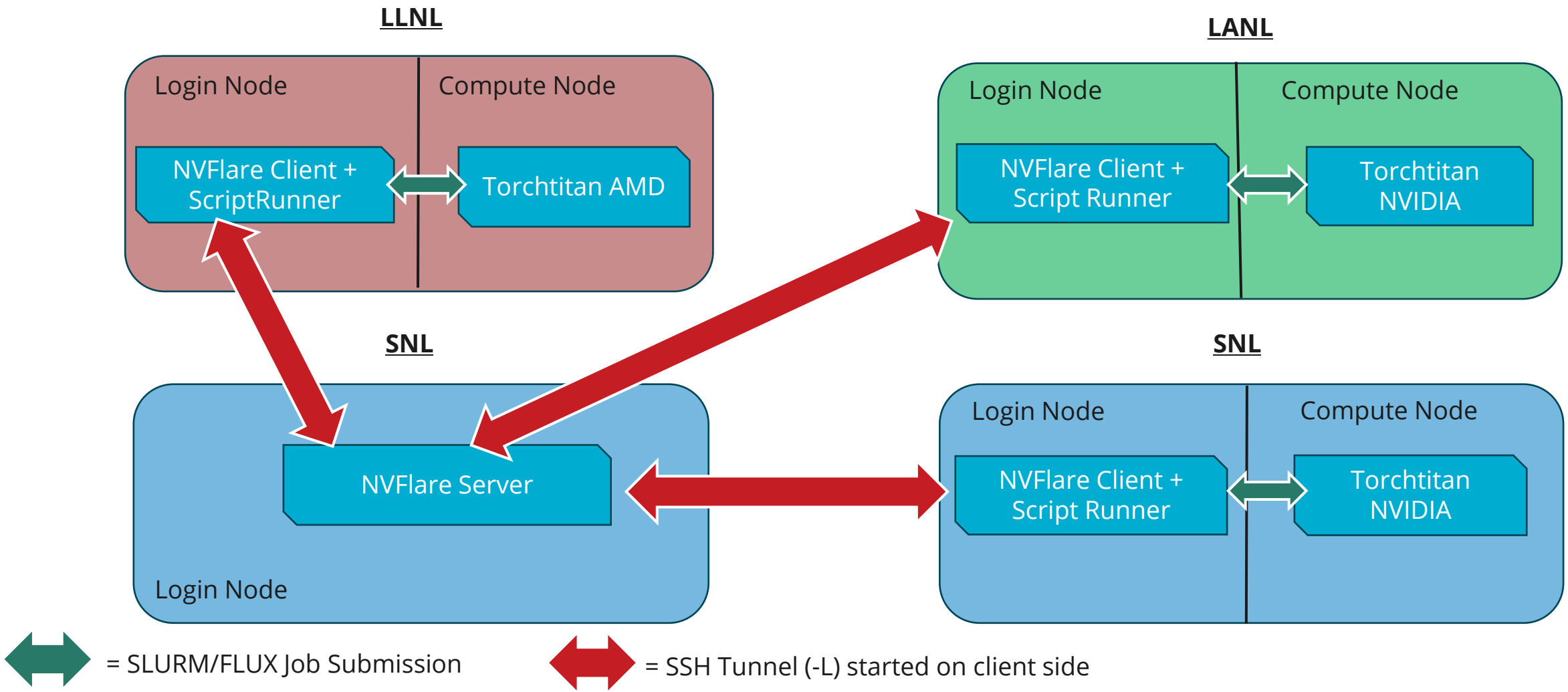
- Simultaneous work for clients.
- Client: Aggregation at a random client.
- Server: Orchestrates the clients.
- Comm: Peer-to-peer.

**Federated Averaging**

- Simultaneous work for clients.
- Client: No aggregation.
- Server: Aggregation, weight management.
- Comm: Through server.

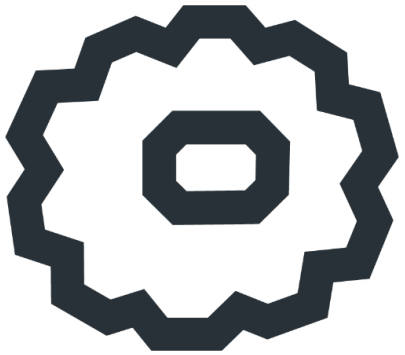
- Aggregation requires multiple copies of the model.
- This creates *substantial* CPU memory pressure on whoever does it, e.g., Llama 3.1 70B takes ~130GB to store weights (float32) per copy!
- Using federated averaging for larger models due to increased memory usage of swarm learning on memory-limited client machines

# TECHNICAL APPROACH: FEDERATED LEARNING



# TECHNICAL APPROACH: WORKFLOW MANAGEMENT

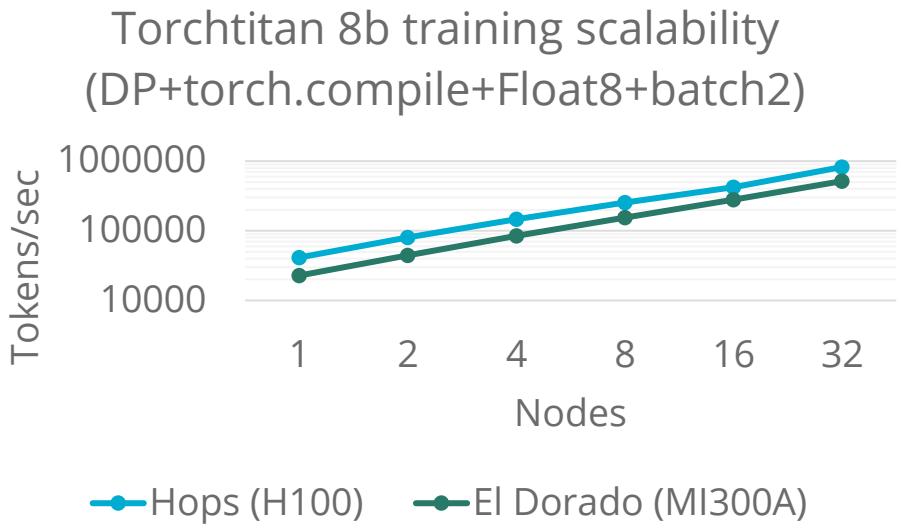
- We evaluated three publicly available federated training workflow frameworks:
  - Nvidia FLARE
  - Flower
  - APPFL




- Federated training frameworks provide tools for model aggregation, secure connection and communication of model weights, and overall workflow management
- All frameworks initially struggled with the size of LLM model weights and communication speed for Flower and APPFL is still somewhat slow, but all frameworks have been quick to respond and develop solutions for these issues
- While we are still evaluating each framework, FLARE has been the main workhorse for this project's training tasks

# BACKEND PERFORMANCE

- Working with Llama 3.1 8b/70b (H100/MI300A)
  - Continued pretraining works
  - Supervised fine tuning evaluation in progress
  - Context parallel evaluation in progress
- Issues
  - TorchTitan under active development
    - Difficult to maintain working version
  - Conversion/checkpointing not stable
  - Frequent timeouts on MI300A




**Good scaling for continued pretraining on NVIDIA and AMD GPUs**

| 70B Training | 16 H100 Nodes DP8TP8batch1 | 16 MI300A Nodes DP64batch1 |
|---|---|---|
| Tokens/sec/GPU | 390.6 | 637.8 |

*DP=Data parallel, TP=Tensor parallel

Credit: Jerry Watkins

# END-TO-END TRAINING PERFORMANCE

**Dataset:** arXiv three-way split (SNL/LANL/LLNL)

| Dataset | Monolithic | SNL |
|---|---|---|
| Tokens | 2 Billion | 709 Million |
| Examples | 124,459 | 41,472 |
| Largest Example | 40 Million | 40 Million |

**Training:** continued pretraining (2 epochs)

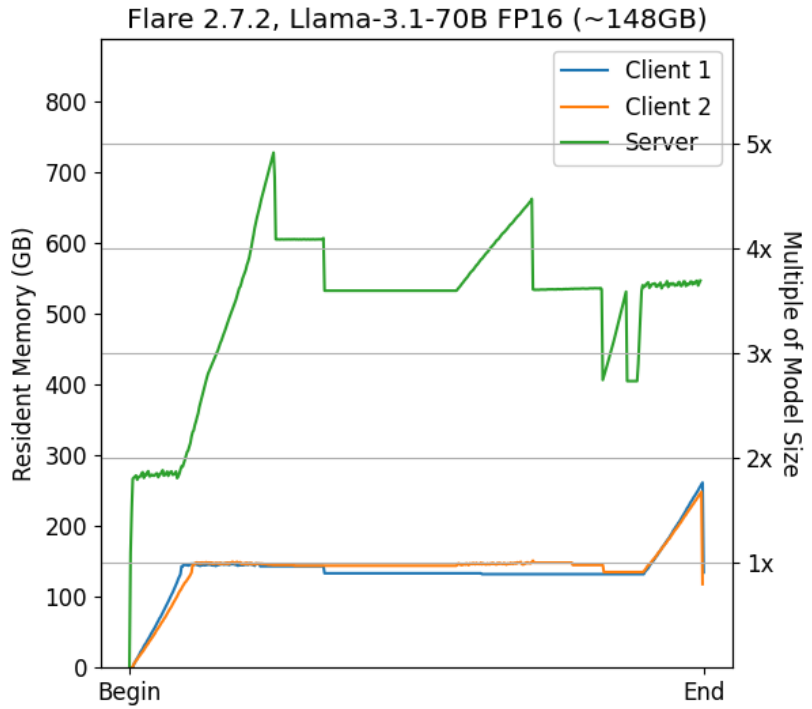
- **Models:**
  - Llama 3.2 1B (4 MI300a)
  - Llama 3.1 8B (4 H100)
  - Llama 3.1 70B (64 MI300a)
- **Federated:**
  - 8 rounds (hops/eldorado/rzadams)
  - Total training steps equivalent to non-federated training
  - Federated training time can include time in queue

| | 1B Mono | 1B SNL | 1B Federated | 8B Mono | 8B SNL | 8B Federated | 70B Mono | 70B SNL | 70B Federated |
|---|---|---|---|---|---|---|---|---|---|
| Training time | 20 hrs | 7 hrs | 11 hrs | 45 hrs | 15 hrs | 28 hrs | 38 hrs | 12 hrs | WIP |

Credit: Jerry Watkins

# EVOLUTION OF NVFLARE

| Feature | Project Impact |
|---|---|
| Message Quantization | Reduced communication overhead |
| Native Tensor Transfer | Reduced model serialization overhead, significant memory reduction |
| Client-Controlled Workflow Improvements | Reduced communication contention in swarm learning jobs when training with an external process |
| File-based Streaming | Significant memory reduction |




**Memory Reduction:** Our NVFlare clients run on cluster login nodes, critical to have minimal memory footprint due to limited memory and contention with other users

**Communication Overhead Reduction:** Moving multiple copies of large language models across the network can cause stress on the network

# LESSONS IN LARGE-SCALE FEDERATED TRAINING

- **Memory**: For models with hundreds of billions of parameters, federated training frameworks need to be very efficient with their memory usage.
  - This has been improved significantly over the course of this project and is no longer a bottleneck
- **Reliability:** As models increase in size and training increases in complexity and cost, so does the likelihood of failures to occur during federated training.
  - Training frameworks need to keep a persistent training state across the entire process for reliable restart capability.
- **Synchronization:** Existing federated training frameworks rely on synchronous server/client communication which is a source of training failure especially at large scale.
  - For complex LLM training approaches, server and clients are only actually doing anything a very small percentage of the time but must remain in constant synchronization
  - An asynchronous model or relaxed synchronization requirements would be beneficial for doing large scale federated training

# CONCLUSIONS AND FUTURE WORK

- **Improvements in memory usage** has enabled much larger training tasks thanks to direct collaboration with partners including Nvidia, Flower, and Argonne but there are still barriers to widespread adoption at the labs
- Model performance improved with more complex and expensive backend training approaches like **reinforcement learning**
- We are repurposing this workflow to do federated training of similarly large-scale **Foundation PDE models** as part of Genesis