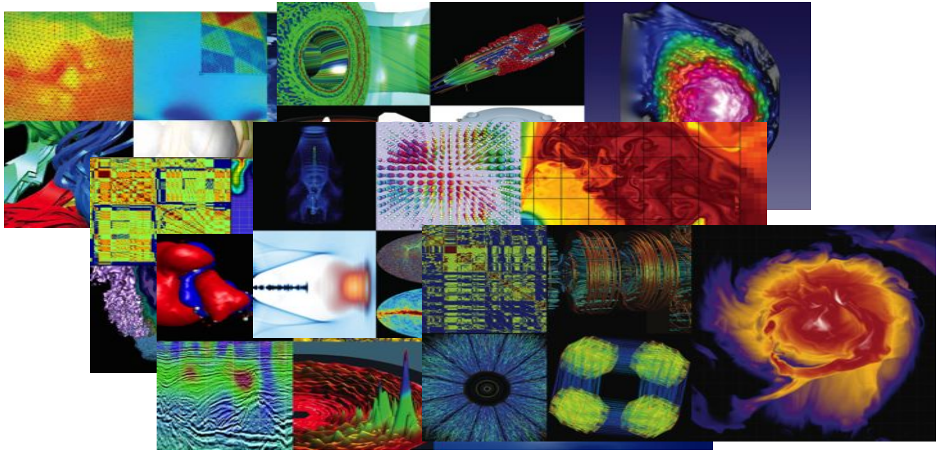
Enabling NCCL on Slingshot 11 at NERSC

NERSC

Jim Dinan, Josh Romero (NVIDIA)
Igor Gorodetsky, Ian Ziemba (HPE)
Peter Harrington, Steve Farrell, Wahid Bhimji, Shashank Subramanian
(Data & AI Services, NERSC)

Cray User Group (CUG)
May 2024

NERSC: Mission HPC for the Dept. of Energy Office of Science



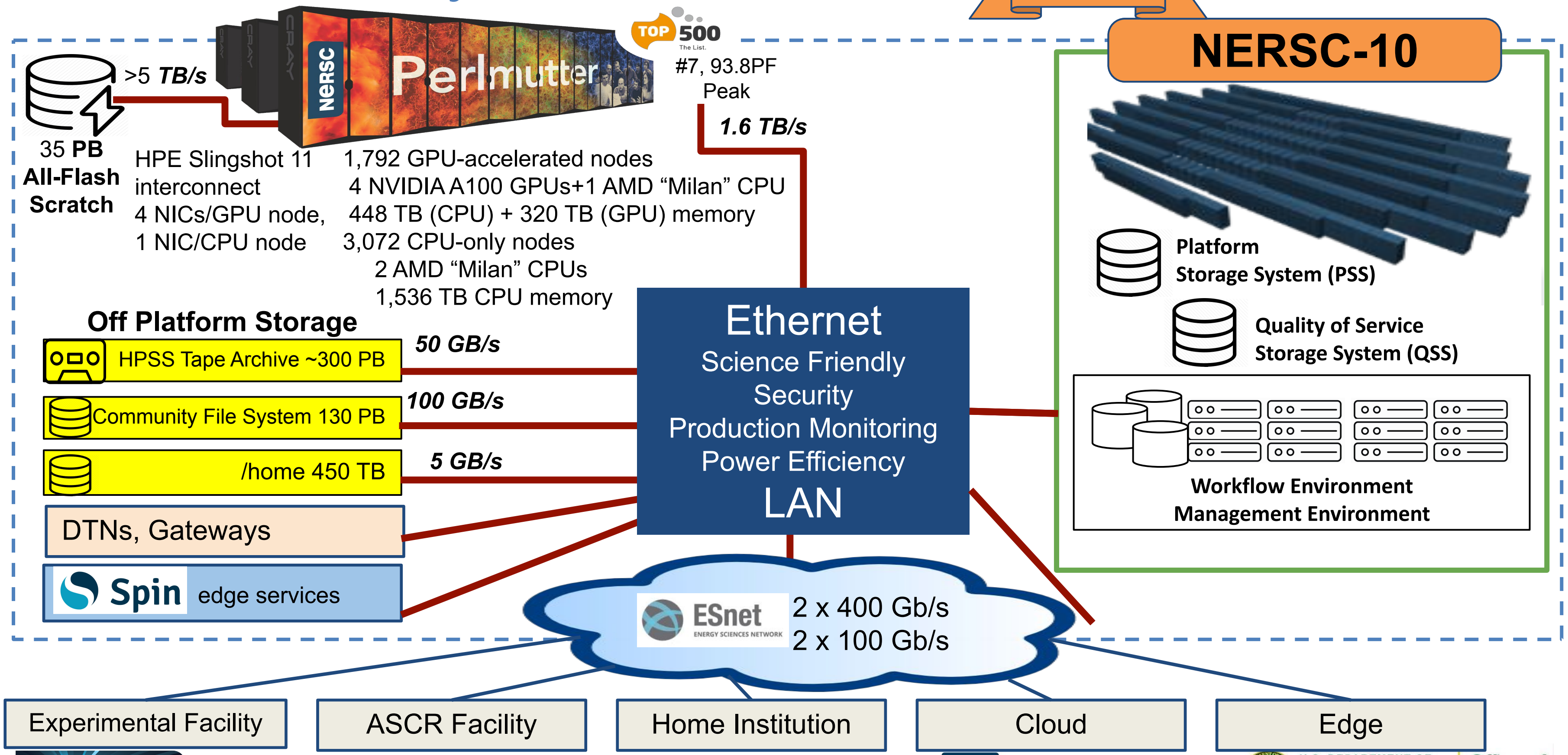
Large compute and data systems

- Perlmutter: ~7k A100 GPUs
- 128PB Community Filesystem ....

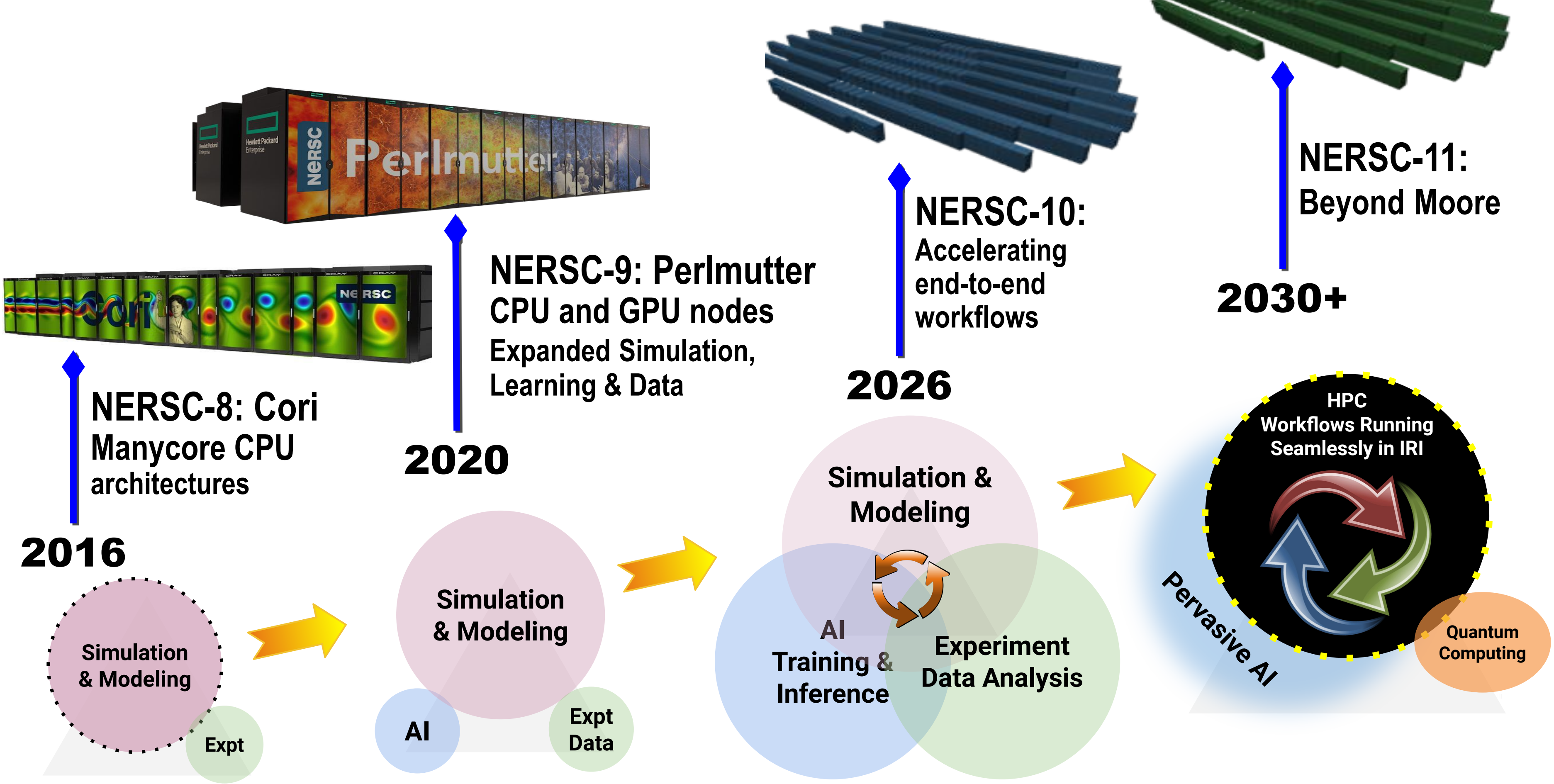
Broad science user base

- ~10,000 users,
- 1000 projects,

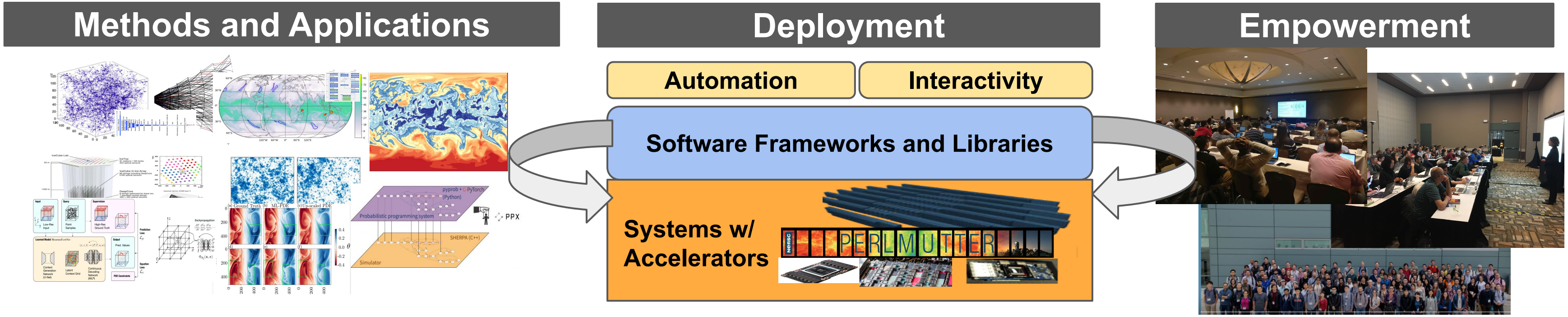
NERSC Facility



NERSC roadmap



NERSC AI Strategy



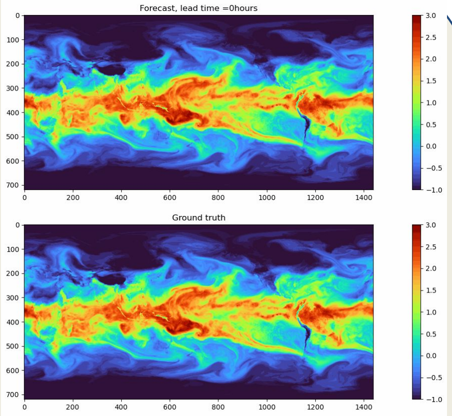
- ***Deploy*** optimized hardware and software systems
  - Work with vendors for optimized AI software (e.g. NCCL on Slingshot!)
- ***Apply*** AI for science using cutting-edge techniques
  - “NESAP” and strategic projects - leverage lessons learned to optimize ecosystem
- ***Empower*** and develop workforce through seminars, training and schools as well as staff, student intern and postdoctoral programs
  - Over 20 DL@Scale tutorials (e.g. SC18-23), 1000s of total participants

NESAP and Perlmutter are Enabling Adoption of Large-scale and Groundbreaking AI

FourCastNet

Pathak et al. 2022 **arXiv:2202.11214**
Collab with Nvidia, Caltech, …

- Forecasts global weather at high-resolution.
- Prediction skill of numerical model; 10000s times faster



Jaideep Pathak
former NERSC
Postdoc now NVIDIA

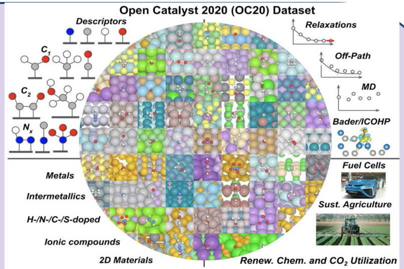
Shashank
Subramanian
Former NERSC
Postdoc now Staff

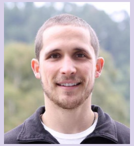
Jared Willard
NERSC Postdoc

CatalysisDL

Chanussot et al. 2021
Collab with CMU, MetaAI, …
arXiv:2010.09990

- NeurIPS 2021-23 Competitions
- Pre-trained models now used with DFT - e.g. FineTuna; AdsorbML





Brandon Wood
former NERSC
Postdoc now Meta AI

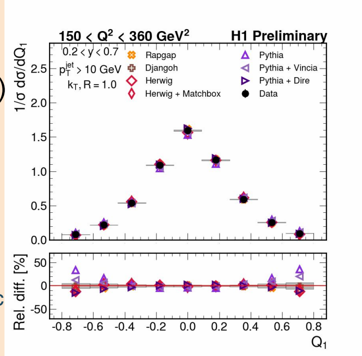
Wenbin Xu
NERSC postdoc

HEP-ML

Collab with LBL Physics division (and H1 Collaboration)

- AI "Unfolding" extracts new physics insights from data
    - Requires Perlmutter for 1000s of UQ runs

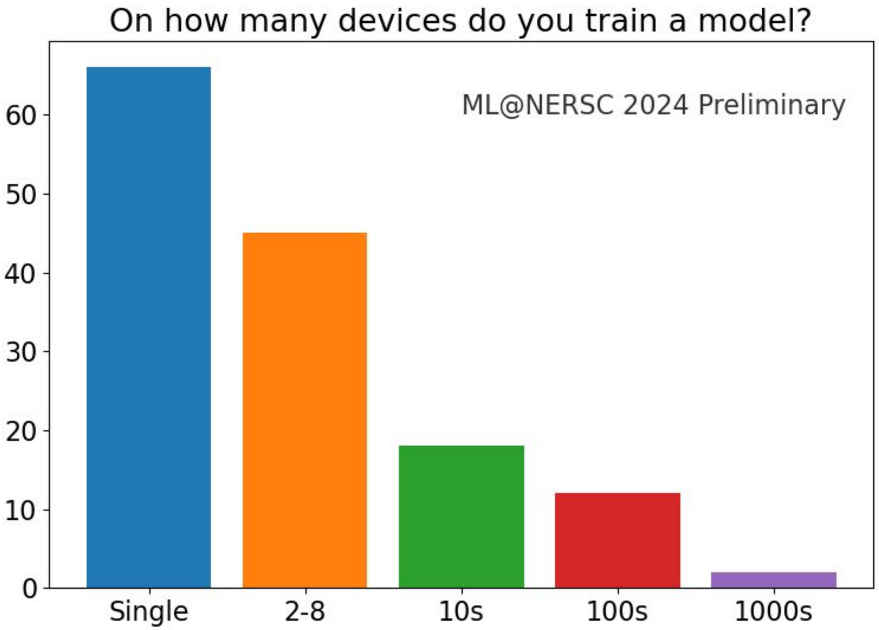
Vinicius Mikuni
NERSC Postdoc

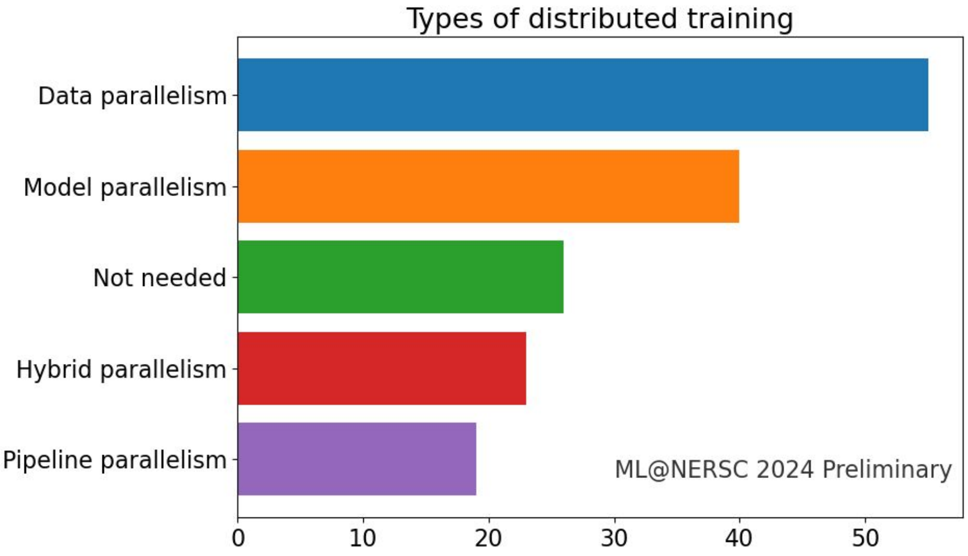


- All use Perlmutter at-scale
- Have complex workflows
- So ***need*** **for network (NCCL+SS) performance**

Broad AI userbase needs distributed DL too…

- NERSC ML Biyearly Survey 2024 currently in progress
  - Results not final





- Continues to show need for complex distributed DL

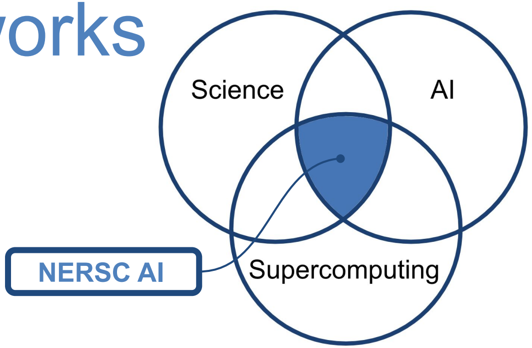
BERKELEY LAB Bringing Science Solutions to the World

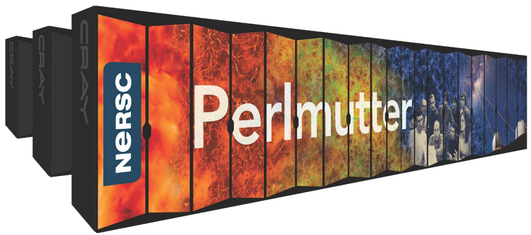
U.S. DEPARTMENT OF ENERGY | Office of Science

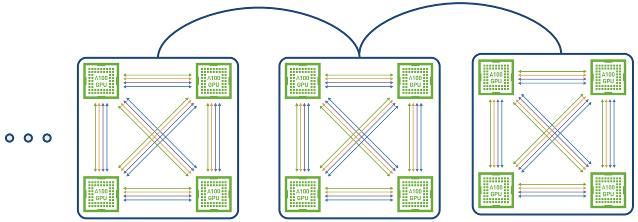
NCCL underlies distributed AI frameworks

See Jesse Tager and Caio Davi's talk for details

- NCCL: NVIDIA Collective Communication Library
- Critical for high-performance distributed training in major deep-learning (DL) frameworks
- Need high-bandwdith, low-latency P2P allreduce between GPUs
    - NCCL able to use NVLink within a node, then interconnect across nodes



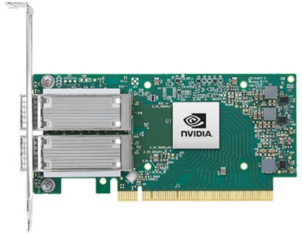




Perlmutter deployment

System was delivered in multiple phases

- Phase I used HPE/Cray Slingshot 10 interconnect between GPU nodes
  - 2 ConnectX-5 NICs (100 Gbps each) per GPU node
  - **RoCE for RDMA, supported by NCCL “out-of-the-box”**



- In Q2 2022 we began Phase II integration of Slingshot 11 interconnect
  - 4 Cassini NICs (200 Gbps each) per GPU node
  - Overall 4x increase in ‘speed-of-light’ bandwidth
  - **Required libfabric implementation for NCCL**

Putting together a testing workload

- NCCL tests: Standard suite of tests
  - Primary performance metric: bus bandwidth
  - Sweep over message sizes - see backup slides for more details

Also *need to test real deep learning workflows* - we saw issues arise even if nccl-tests succeed with e.g. forking, dataloaders, processes using mpi etc.

- Standard DL workloads - e.g. ResNet/ImageNet and Megatron LLMs
- MLPerf HPC - Science HPC part of MLPerf benchmark suite
  - CosmoFlow - 3D CNN predicting cosmological parameters
  - DeepCAM - segmentation of phenomena in climate sims
  - OpenCatalyst - GNN modeling atomic catalyst systems
- FourCastNet - Large scale weather/climate training - model/data parallel

Initial Slingshot 11 performance without libfabric

- Initially, NCCL performance on SS11 at Perlmutter did not use libfabric
- Forced to ***fall back to TCP*** for inter-node communications
- **2-3x reduction in bandwidth**: *impact on even small-scale DL workloads*
- Began measuring and tracking performance

Benchmarks	**Phase I, SS10**	**Phase II, SS11**
NCCL-Tests AllReduce (32 MB) 2 Node (GB/s) (**higher is better**)	26	9.5
Tensorflow 2 + Horovod (ResNet/ImageNet) 2 Nodes (samples/second) (**higher is better**)	4700	3900
DeepCam-4k 8 Node Runtime (min) (**lower is better**)	5.2	7.0

libfabric plugin

- AWS already provide an open-sourced libfabric plugin for NCCL for their EFA network

AWS OFI NCCL

AWS OFI NCCL is a plug-in which enables EC2 developers to use libfabric as a network provider while running NVIDIA's NCCL based applications.

aws

https://github.com/aws/aws-ofi-nccl

- Provider needs network-specific implementation for SS11
- Strategy: leverage/adapt this plugin for SS11 libfabric on Perlmutter
    - o Initial efforts led by Josh Romero, Jim Dinan (NVIDIA)

Early integration of libfabric plugin

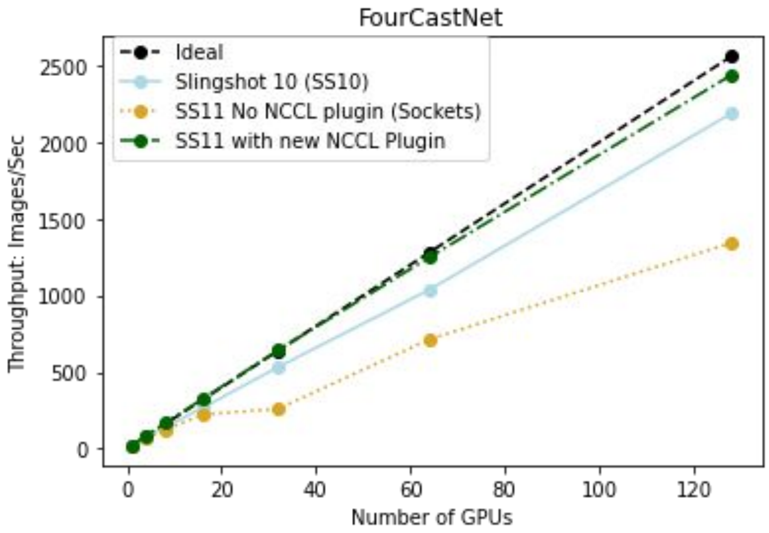
- Early days (2022 Q3-4): focus on mid-scale functionality and performance
  - Multiple rounds of iterating on the custom NCCL plugin, HPE's libfabric implementation, Slingshot software versions, etc.
  - Debugging challenges: hangs (sometimes intermittent), segfaults, variable performance; rapidly moving software as SS11 was hardened in general
- Initial deployment Q4 2022

```
> module load nccl/2.15.5-ofi
Warning: This is an experimental release of NCCL with an OFI plugin for use with libfabric on Perlmutter.
In case of issues, please refer to our known issues: https://docs.nersc.gov/current/
    and open a help ticket if your issue is not listed: https://help.nersc.gov/
```

- Also integrated as a plugin for Shifter, our HPC container runtime
- Small-scale runs perform well, for both NCCL benchmarks and DL workloads (adopted by some non-DL workloads using NCCL, e.g. VASP)

Initial deployment: performance results

- Already saw performance and scaling improvements



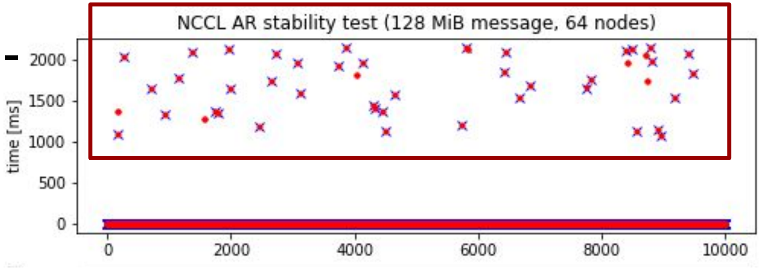
- However larger-scale runs saw hangs
  - More frequent as scale increased

Initial deployment issues

- When things did run at scale, saw intermittent **substantial drops** in NCCL bandwidth (>10-100x reduction); worst at large scale

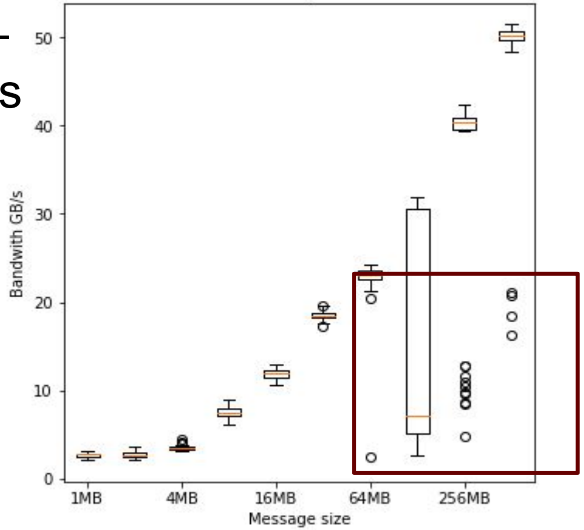
64 nodes

Time
2000 ms -



512 nodes

50 –
GB/s



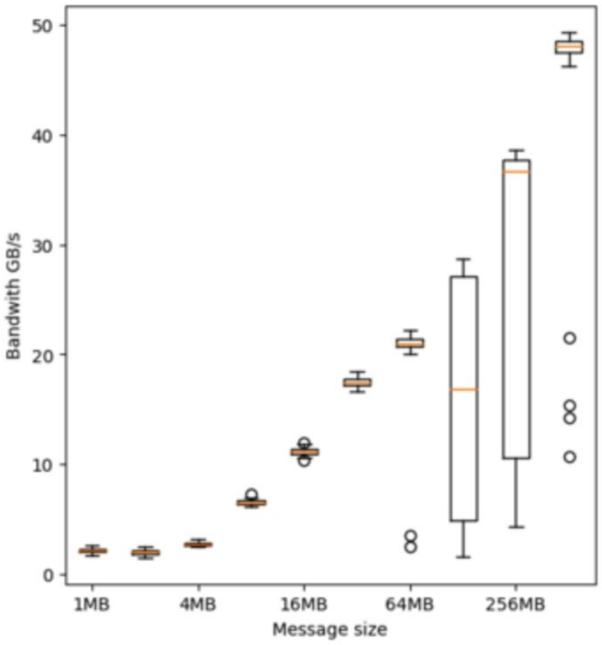
Improvements to the initial deployment required NVIDIA/HPC (and NERSC) collaboration

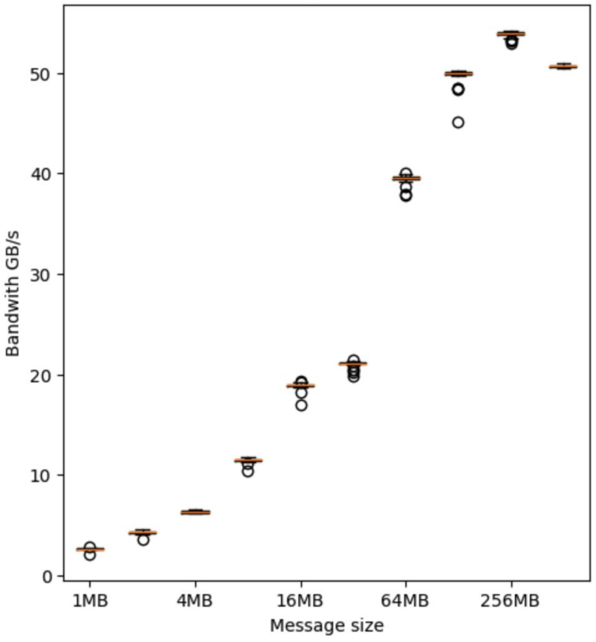
See Jesse Tager and Caio Davi's talk for more details

- **Performance drops**: fairly quickly root-caused to the protocol used in Slingshot for queueing messages
    - NVIDIA devs worked with Igor Gorodetsky (HPE) to resolve
    - Fix integrated into Slingshot Host Software (SHS) 2.1.0 Q2 2023
- **Hangs**: Multiple causes across hardware/software stack
    - Some intermittent/only emergent at very large scale
    - E.g. Pytorch multiprocessing data loader and need for "FI_MR_CACHE_MONITOR=userfaultfd"
    - "Final" issue traced to undetected GPU nvlink hardware error
        - "Solved" by adding new nvbandwidth test to node health checker

Performance measurements: NCCL Tests

- Before-and-after of allreduce bandwidth over **4,096 GPUs**:

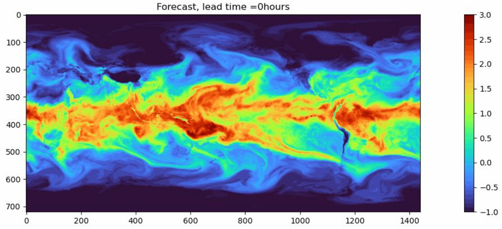


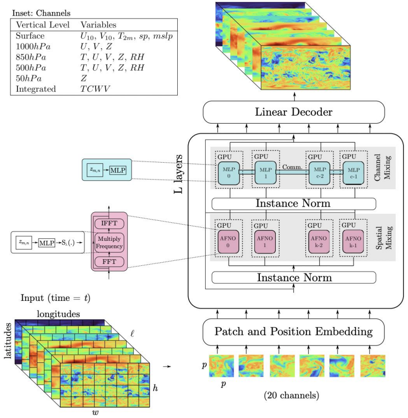


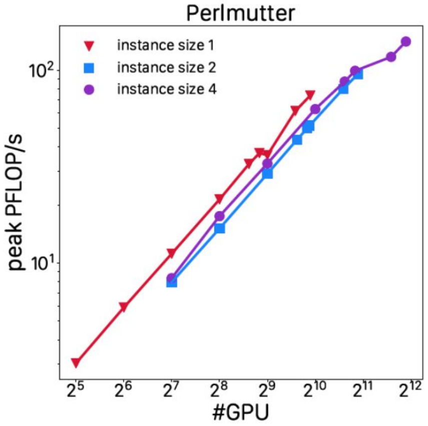
Stable
>=50 GB/s,
without hanging

Performance measurements: real workloads

- Impact on real workload: FourCastNet++ ([PASC 2023]())
  - Hybrid data-model parallel DL weather model training
- Scaling runs for PASC paper were done on Phase I, so they provide a strong SS10 baseline
- Taking largest-scale config from the paper (~4k GPUs) and re-running, now see **60% end-to-end speedup from SS10**







Performance measurements: MLPerf HPC v3.0

- **NERSC partnered with HPE and NVIDIA to submit results using Perlmutter**
  - Previously submitted with Phase 1 system (and slingshot 10)

Achieved highly competitive results:

- Includes other performance improvements:
  - NGC containers; JIT complilation; Optimized data movement and DALI
- Excellent improvements over NERSC's previous results
  - **CosmoFlow 2x on 0.5*GPUs**
  - **DeepCam 1.5x**
  - **OpenCatalyst 5x**

Nodes	GPUs	CosmoFlow	DeepCAM	OpenCatalyst	OpenFold
128	512	4.73		21.04	
224	896				16.11
512	2048		1.81		

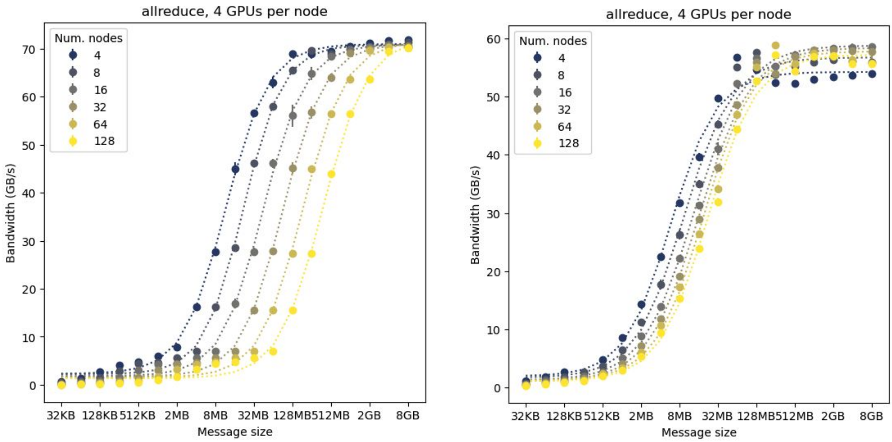
Full results: https://mlcommons.org/benchmarks/training-hpc/
Code and log files: https://github.com/mlcommons/hpc_results_v3.0/

NERSC

More detailed NCCL measurements

Can now reliably characterize NCCL collective performance over a wide range of settings:

- Node counts/topology/message size
- NCCL algorithm & settings
- Slingshot protocols & other settings
- See backup slides for more plots and validated software versions



Ring (left) vs. tree (right) allreduce bus bandwidth as a function of message size

- See **largely expected & consistent behavior across these settings**
- Enables **empirically-based performance modeling** for NCCL workloads

Conclusions and next steps

- Transformative AI for Science at NERSC requires HPC-scale Deep Learning and so NCCL on Slingshot 11
- Great collaboration with NVIDIA and HPE to develop libfabric plugin
- Required extensive testing at scale,
  - Broad sweeps of low-level tests and real (scientific) DL workloads
- Now achieve significant improvements over SS10

Future work:

- Continuing to harden and add regular performance testing and tracking
  - E.g. adding NCCL all-reduce into "Reframe" testing suite
- Extend NCCL plugin integration into Perlmutter user software env:
  - Beyond 'core' features: non-DL applications; podman-hpc and 'containerizability' of the plugin
- Helping and encouraging HPE and Nvidia production support

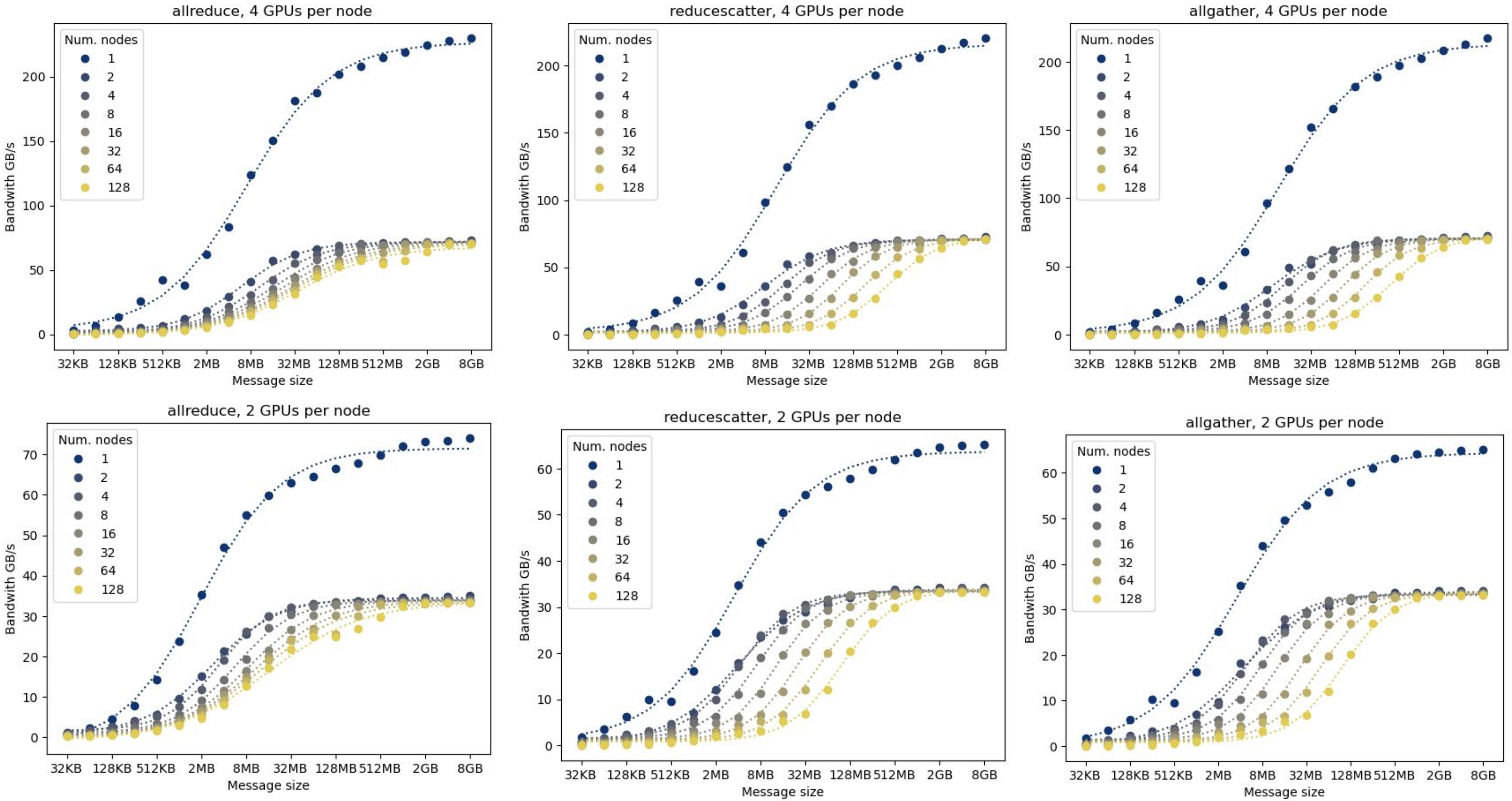
Thank You!

This was a major effort, thanks to all collaborators across NERSC, NVIDIA, HPE, and AWS! A non-exhaustive list below…

- Thorsten Kurth (NVIDIA)
- Brian Barrett (AWS)
- Lisa Gerhardt (NERSC)
- Brian Friesen (NERSC)
- Christopher Samuel (NERSC)
- Daniel Margala (NERSC)

Backup

Extended benchmark measurements



NCCL Testing configuration

- Tests run w/ 4 GPUs per node, sweeping over message sizes from 1MB to ~1GB
- Primary performance metric: bus bandwidth
    - Better number to compare against hardware peak bandwidth "speed of light"
    - (message size / time) * correction factor
        - factor depends on communicator size & which collective algorithm is being performed (allreduce, allgather, etc)
- In-place and out-of-place measurements averaged to get final number; multiple trials run per job for error bars
    - In-place: same buffer is used for comms and result
    - Out-of-place: additional buffer used for comms, result updated after comms complete
    - The two should be about the same in performance

NCCL Testing configuration

Validated configurations, Slingshot Host Software (SHS) v2.1.x:

- CUDA 11.7, NCCL 2.15.5, aws-ofi-nccl v1.6.0-hcopy
- CUDA 11.7, NCCL 2.17.1, aws-ofi-nccl v1.6.0-hcopy
  - Used for most results presented here
- CUDA 12.0, NCCL 2.18.3, aws-ofi-nccl v1.6.0
- CUDA 12.2, NCC 2.19.4, aws-ofi-nccl v1.6.0

Newer CUDA/NCCL pairs have also been working w/ our container setup:

- CUDA, NCCL from NVIDIA NGC image
- Inject plugin and slingshot dependencies via `$LD_LIBRARY_PATH`