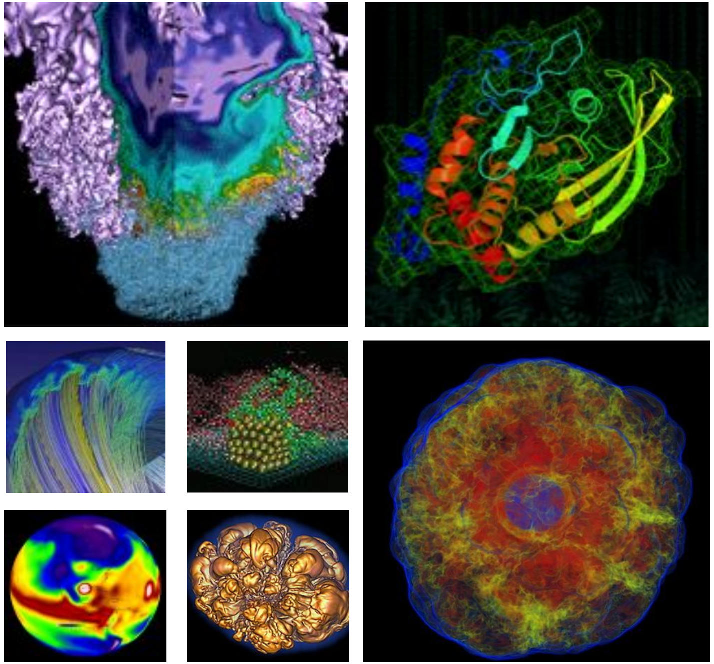
Deep Learning at NERSC
Usability, Capability, and Everything in Between

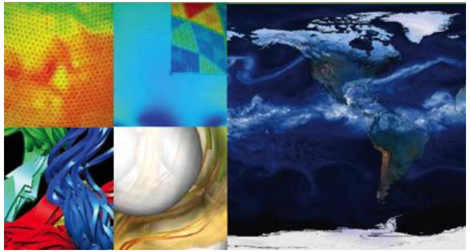


Steven Farrell,
on behalf of many folks at LBNL

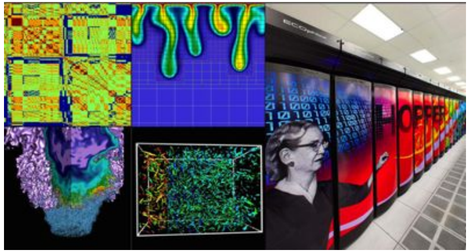
SC18 Deep Learning on Supercomputers Workshop

NERSC

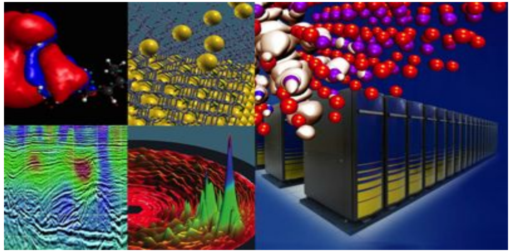
- ***National Energy Research Scientific Computing Center***
  - Established in 1974 as the first unclassified supercomputer center
- **Now the Mission HPC facility for the DOE Office of Science**
  - 7000 users, 800 projects, 700 codes



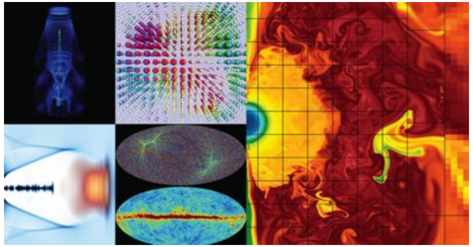
Bio Energy, Environment



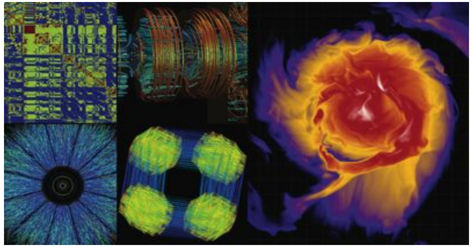
Computing



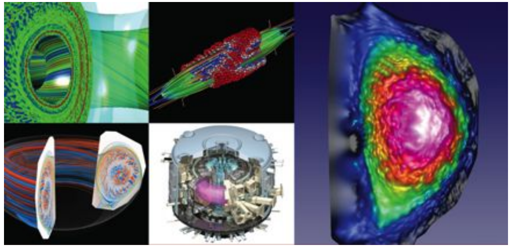
Materials, Chemistry, Geophysics



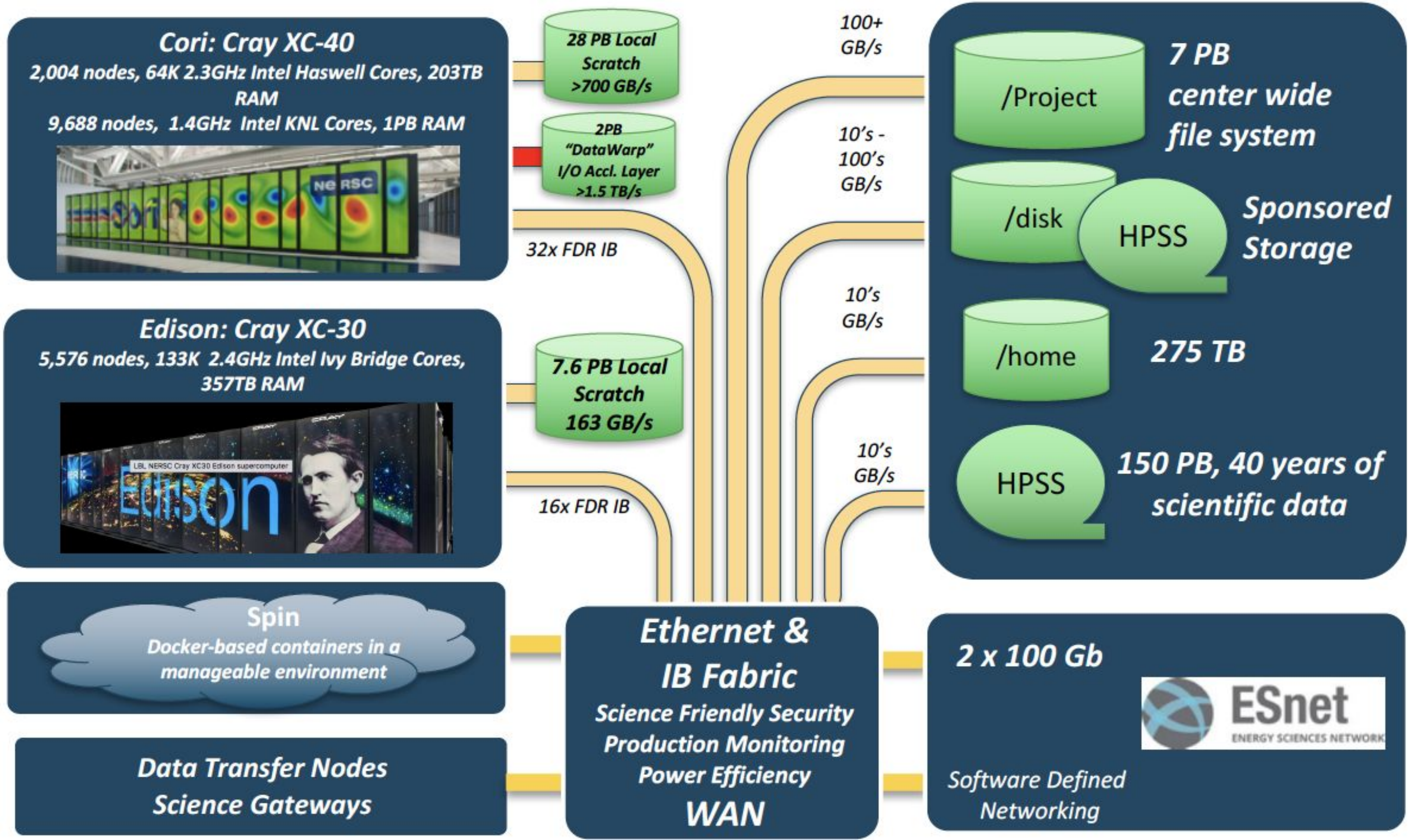
Particle Physics, Astrophysics



Nuclear Physics

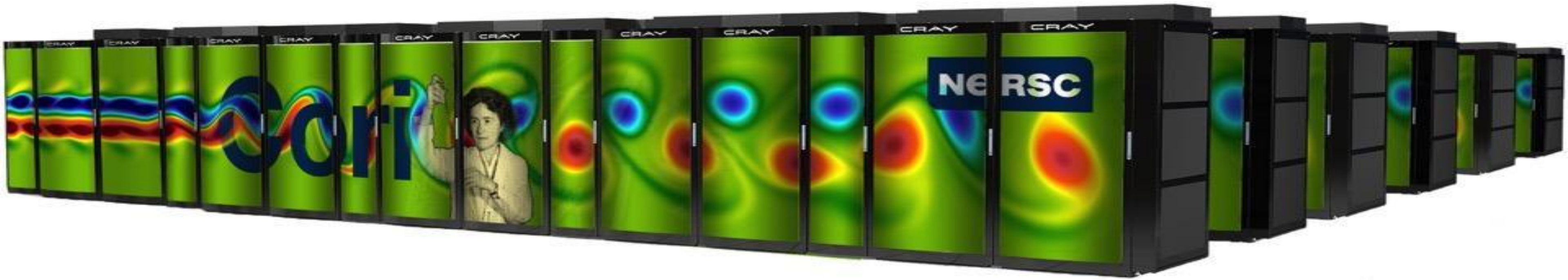


Fusion Energy, Plasma Physics


Cori: Cray XC-40
2,004 nodes, 64K 2.3GHz Intel Haswell Cores, 203TB RAM
9,688 nodes, 1.4GHz Intel KNL Cores, 1PB RAM
NERSC
28 PB Local Scratch >700 GB/s
2PB "DataWarp" I/O Accl. Layer >1.5 TB/s
32x FDR IB
Edison: Cray XC-30
5,576 nodes, 133K 2.4GHz Intel Ivy Bridge Cores, 357TB RAM
LBL NERSC Cray XC30 Edison supercomputer
Edison
7.6 PB Local Scratch 163 GB/s
16x FDR IB
100+ GB/s
10's - 100's GB/s
10's GB/s
10's GB/s
/Project
7 PB center wide file system
/disk
HPSS
Sponsored Storage
/home
275 TB
HPSS
150 PB, 40 years of scientific data
Spin
Docker-based containers in a manageable environment
Ethernet & IB Fabric
Science Friendly Security
Production Monitoring
Power Efficiency
WAN
2 x 100 Gb
ESnet
ENERGY SCIENCES NETWORK
Software Defined Networking
Data Transfer Nodes
Science Gateways

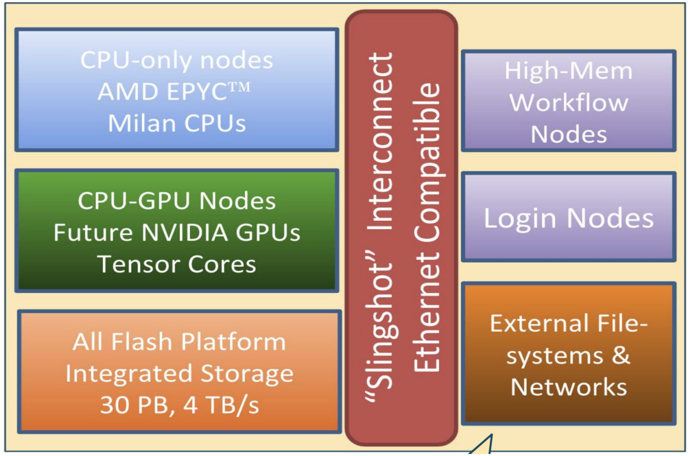
The Cori supercomputer

- 2,388 Intel Xeon ("Haswell") nodes
- 9,688 Intel Xeon Phi ("KNL") nodes (29.5 PFlops)
- 1.8 PB "burst buffer" (1,700 GB/s I/O bandwidth)
- 30 PB Lustre (700 GB/s I/O bandwidth)
- Entered Top500 list at #5 in Nov 2016, now #12



The Perlmutter supercomputer

- Next-gen system optimized for science
- Cray Shasta system with 3-4x capability of Cori
- GPU-accelerated (4x NVIDIA) nodes and CPU-only (AMD) nodes
- Cray Slingshot high performance network
- Single-tier All-Flash Lustre based file system



Delivered late 2020

Deep Learning at NERSC

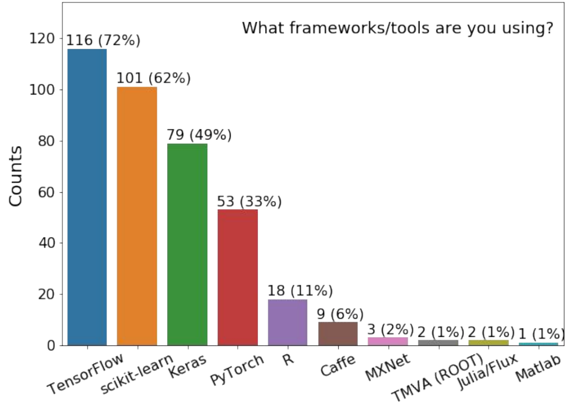
Optimized DL software stack

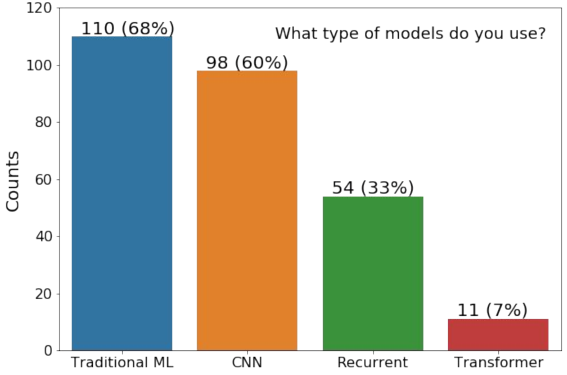
- **Frameworks**: TensorFlow, Keras, PyTorch, Caffe, …
- **Multi-node libraries**: Cray PE ML Plugin, Horovod, PyTorch distributed
- **150-200 users** at NERSC

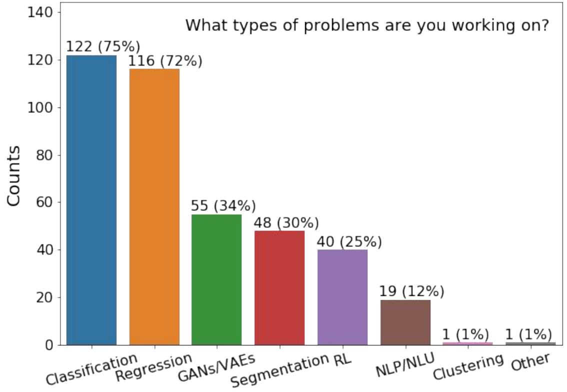
Big Data Center collaborations

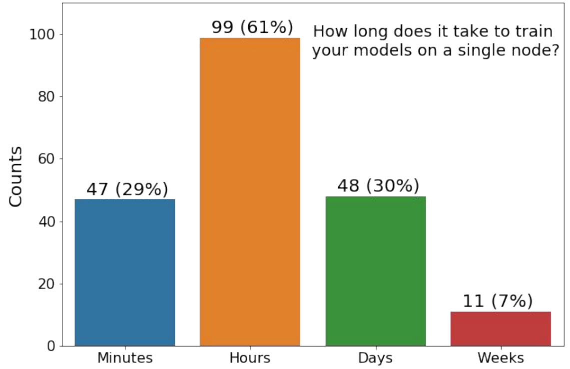
- With **Intel** optimizing TensorFlow and PyTorch for CPU with MKL
- With **Cray** optimizing scaling, workflows, data management and I/O

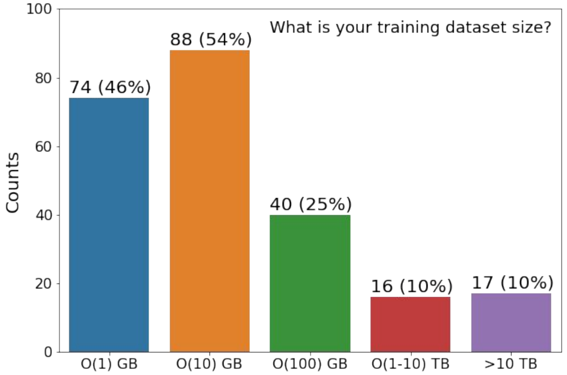
DL users at NERSC

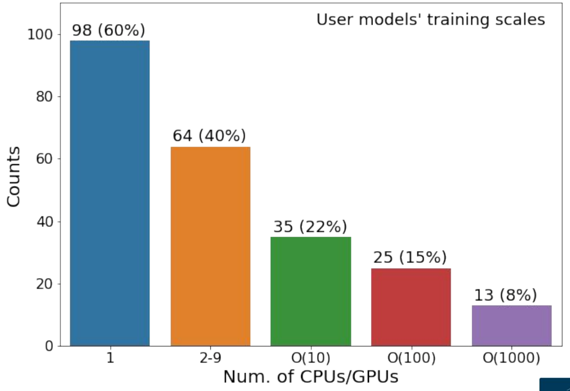










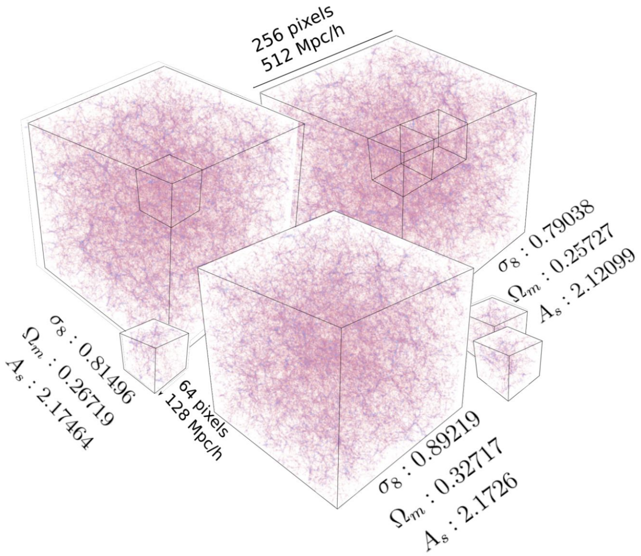


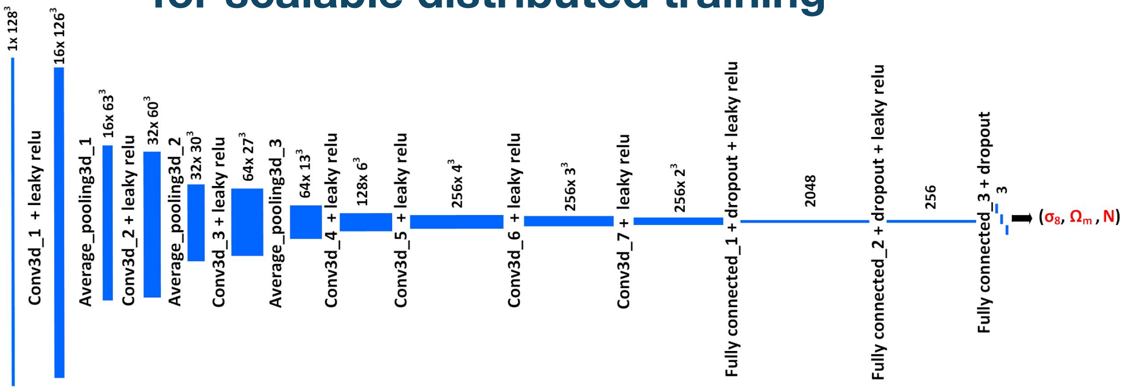
CosmoFlow

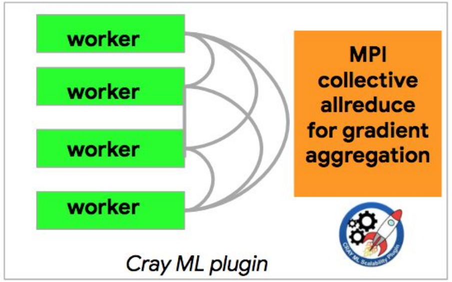
Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arnemann, Lei Shao, Siyu He, Tuomas Karna, Daina Moise, Simon J. Pennycook, Kristyn Maschoff, Jason Sewall, Nalini Kumar, Shirley Ho, Mike Ringenburg, Prabhat, Victor Lee

NERSC

- **Collaboration between NERSC, Cray, Intel**
- **Predicting cosmological parameters from 3D voxels of Dark Matter simulations**
- **Extended a previous study to larger scale and more parameters (2 -> 3)**
- **Uses TensorFlow and Cray PE ML Plugin for scalable distributed training**





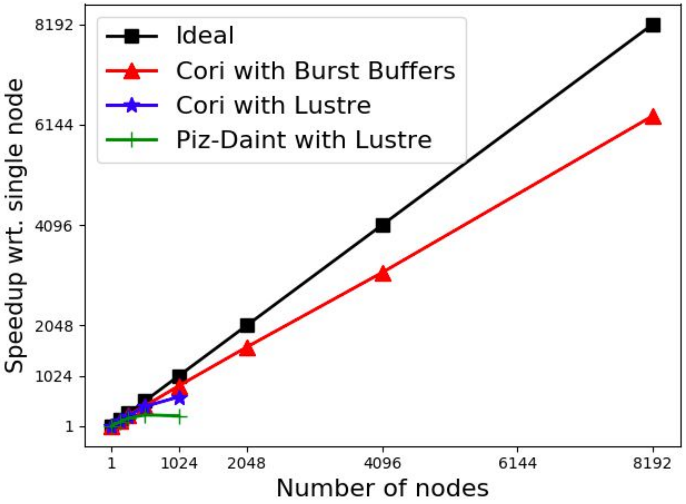


U.S. DEPARTMENT OF ENERGY | Office of Science

[arXiv:1808.04728]

BERKELEY LAB

CosmoFlow results

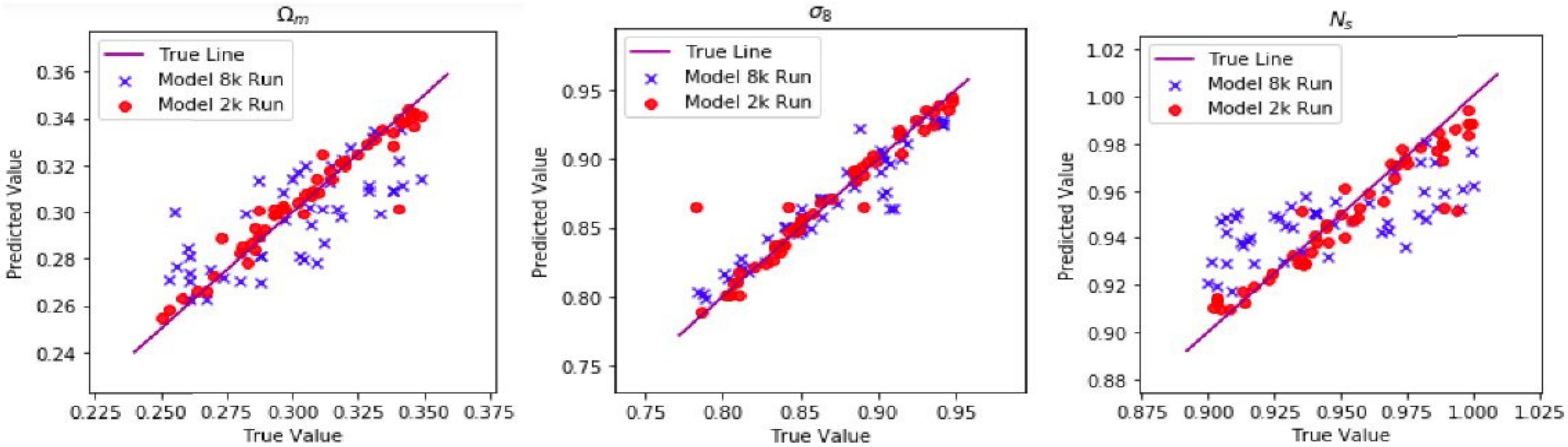


Successful prediction of 3 cosmological parameters

- Comparable to experimental uncertainties for Ω_m and σ_8, almost 5x better for N_s.

Scaled to 3.5PF on Cori with 8k KNL nodes

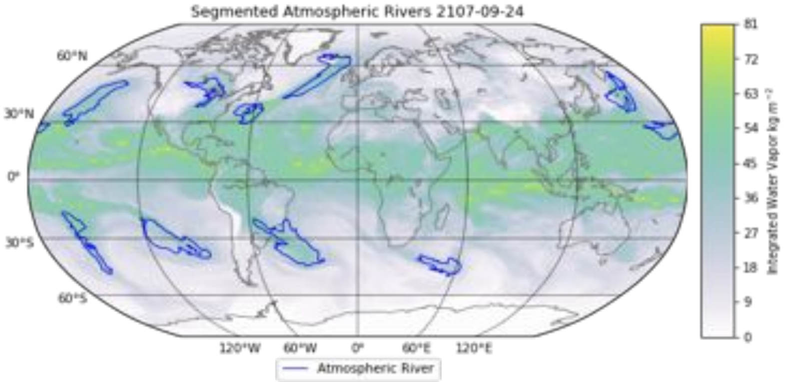
- Convergence issues at scale for further study



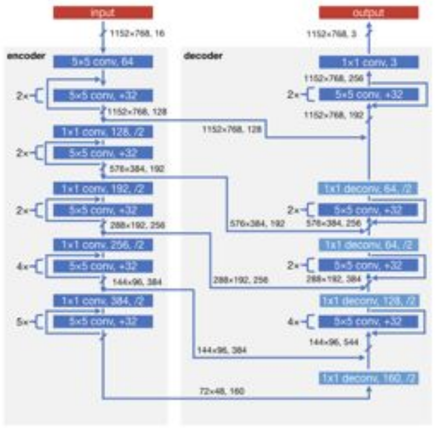
DL climate analytics

Thorsten Kurth, Sean Treichler, Joshua Romero, Mayur Mudigonda, Nathan Luehr, Everett Phillips, Ankur Mahesh, Michael Matheson, Jack Deslippe, Massimiliano Fatica, Prabhat, Michael Houston

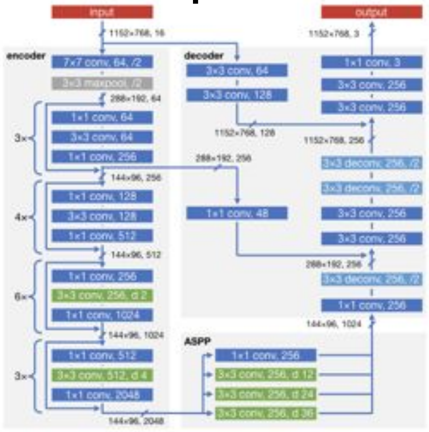
- **Collaboration between NERSC, NVIDIA, OLCF**
- **Identify extreme weather phenomena (tropical cyclones, atmospheric rivers) in climate simulations**
- **Uses TensorFlow and Horovod, with some scaling improvements**
  - Hierarchical control plane, hybrid all-reduce, gradient lag
  - Data staged to local NVMe
- **Uses modified Tiramisu and DeepLabv3+ architectures for image segmentation**



Tiramisu



DeepLabv3+



[arXiv:1810.01993](https://arxiv.org/abs/1810.01993)

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

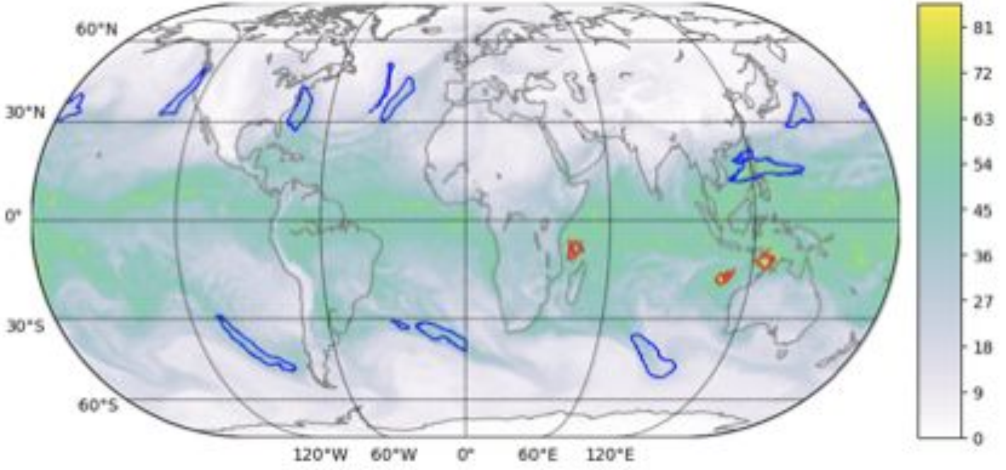
DL climate results

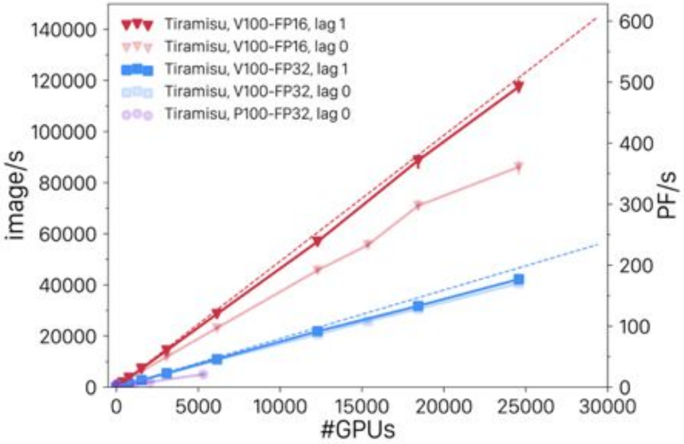
Best IoU achieved: ~73%

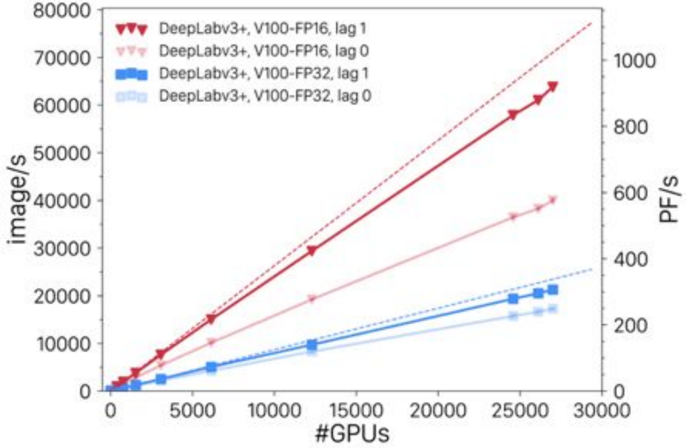
Excellent scaling on Summit:

DeepLabv3+, 4560 nodes (27360 GPU)

- **1.13 ExaFlop/s (FP16) peak**
- **999 PetaFlop/s (FP16) sustained**







U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

DL climate results

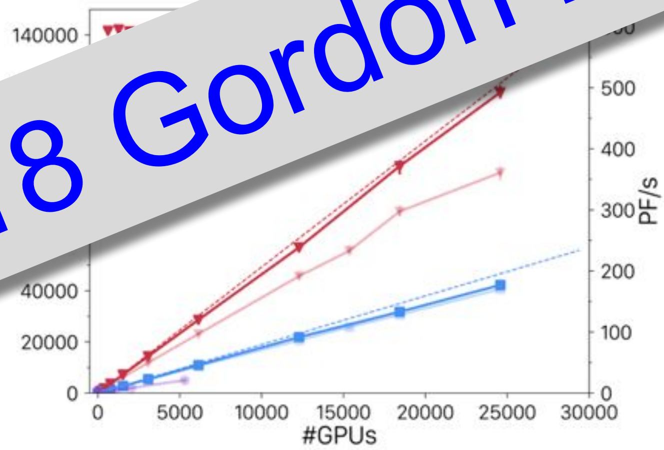
Best IoU achieved: ~73%

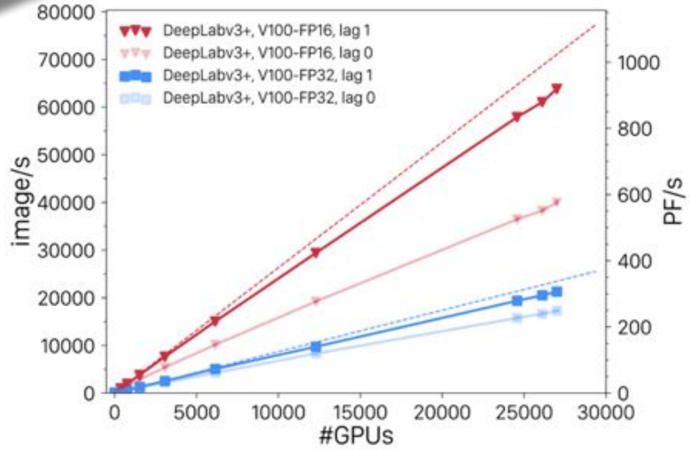
Excellent scaling on Summit:

DeepLabv3+, 4560 nodes (27360 GPU)

- **1.13 ExaFlop/s (FP16) peak**
- **999 PetaFlop/s (FP16) s**

2018 Gordon Bell Prize Winner





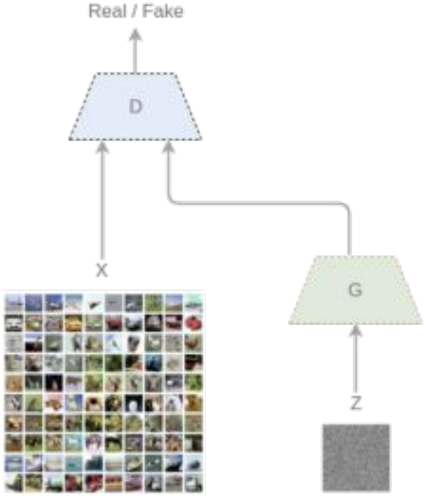
U.S. DEPARTMENT OF ENERGY

BERKELEY LAB

Generative models

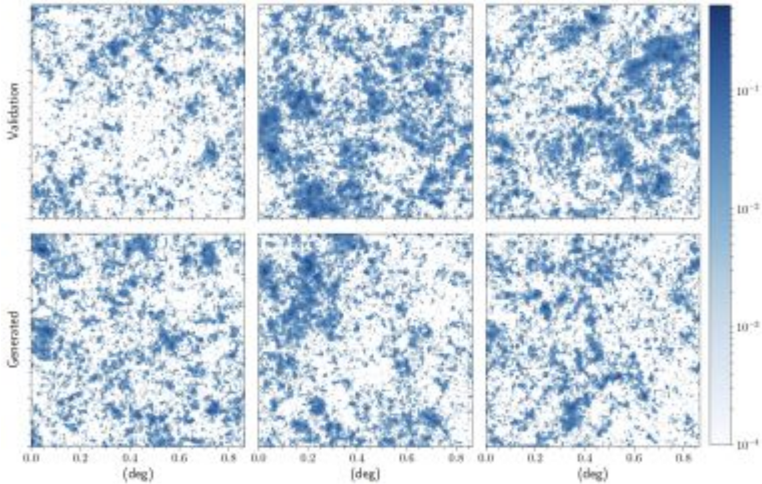
Mustafa Mustafa, Deborah Bard, Wahid Bhimji, Zarija Lukić, Rami Al-Rfou, Jan Kratochvil, Steve Farrell, Ben Nachman, Thorsten Kurth, Harley Patton

Generative Adversarial Networks (GANs)



CosmoGAN

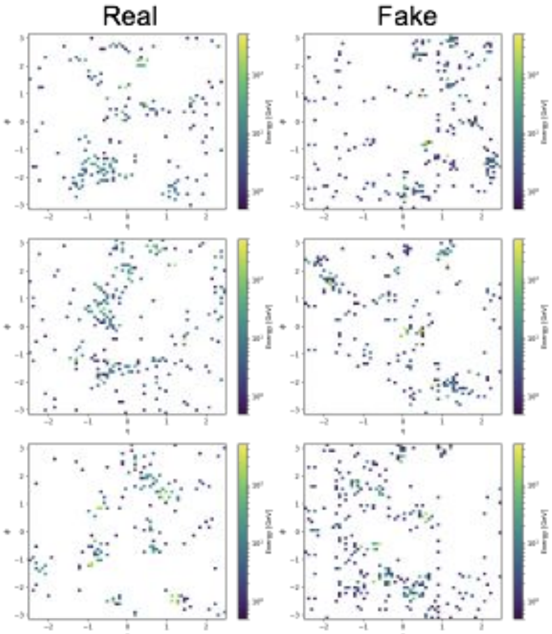
arXiv:1706.02390



We demonstrate ability of GANs to learn complex scientific data distributions, with powerful possible applications

- Augment/replace expensive simulations
- Interpolate between existing simulations

HEP-GAN



Shown at CHEP 2018

Jupyter

Steve Farrell, Wahid Bhimji, Shreyas Cholia, Matt Henderson, Oliver Evans, Aaron Vose, Rollin Thomas, Shane Canon

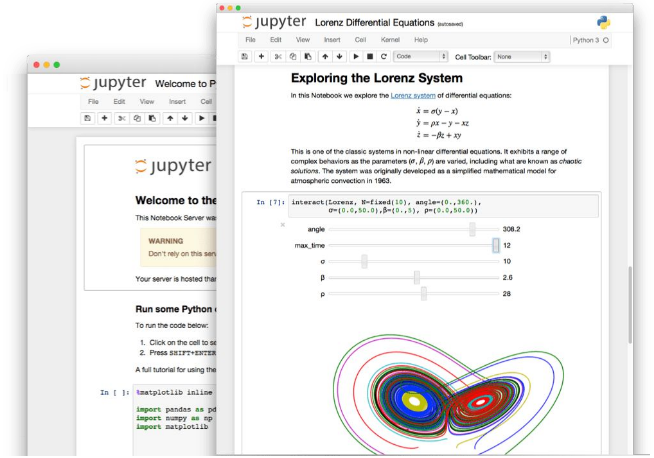
We are leveraging Jupyter as a platform for distributed Deep Learning on our systems

- Lower barrier of entry for ML on HPC
- Enable new, interactive workflows for ML HPC research

Notebook solutions we’re developing

- Distributed training
- Distributed hyper-parameter optimization
- Widgets for live, interactive monitoring

https://github.com/sparticlesteve/cori-intml-examples

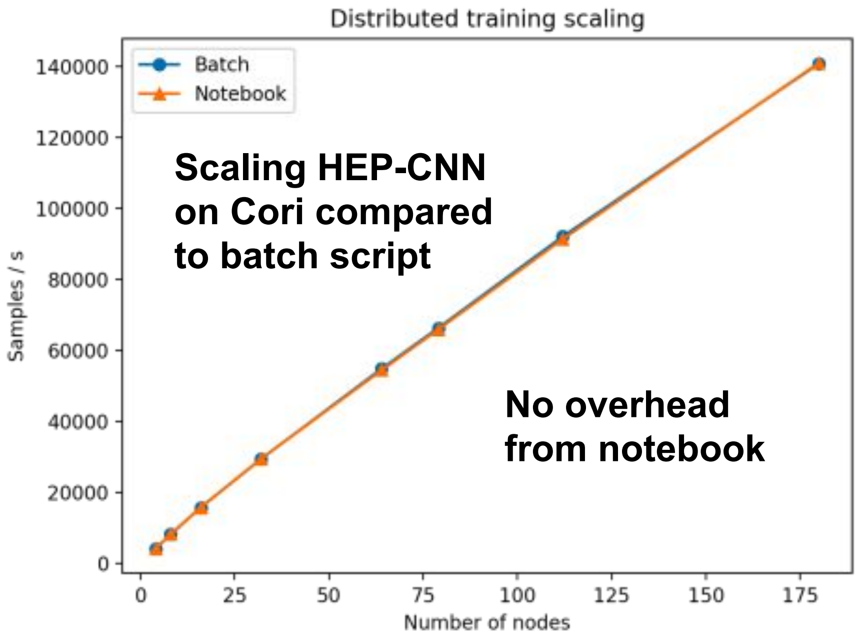


Distributed training from notebooks

With IPyParallel, notebook cells can be marked for parallel execution via cell magic

- MPI in a notebook, but *interactive!*

```
In [10]: %%px

         # Train the model
         history = train_model(model, train_input=train_input, train_labels=train_labels,
                               valid_input=valid_input, valid_labels=valid_labels,
                               batch_size=batch_size, n_epochs=n_epochs,
                               use_horovod=True)

[stdout:0]
Train on 64000 samples, validate on 32000 samples
Epoch 1/4
 - 31s - loss: 0.2363 - acc: 0.9068 - val_loss: 0.1882 - val_acc: 0.9268
Epoch 2/4
 - 30s - loss: 0.0518 - acc: 0.9812 - val_loss: 0.0598 - val_acc: 0.9805
Epoch 3/4
 - 31s - loss: 0.0226 - acc: 0.9922 - val_loss: 0.0464 - val_acc: 0.9860
Epoch 4/4
 - 30s - loss: 0.0157 - acc: 0.9944 - val_loss: 0.0548 - val_acc: 0.9862
[stdout:1]
Train on 64000 samples, validate on 32000 samples
Epoch 1/4
 - 31s - loss: 0.2369 - acc: 0.9056 - val_loss: 0.1882 - val_acc: 0.9268
Epoch 2/4
 - 30s - loss: 0.0513 - acc: 0.9817 - val_loss: 0.0598 - val_acc: 0.9805
Epoch 3/4
 - 30s - loss: 0.0223 - acc: 0.9922 - val_loss: 0.0464 - val_acc: 0.9860
Epoch 4/4
 - 30s - loss: 0.0149 - acc: 0.9950 - val_loss: 0.0548 - val_acc: 0.9862
```

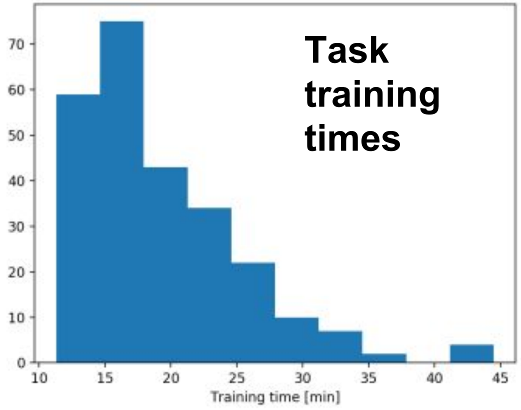


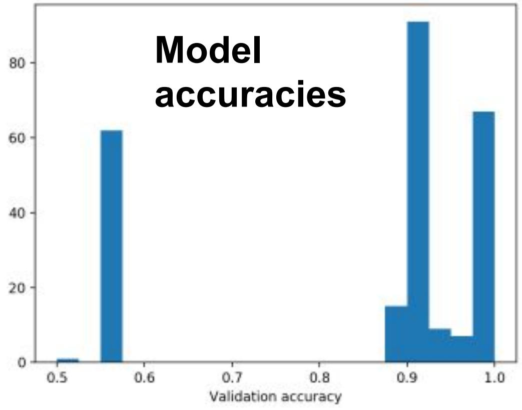
Distributed HPO from notebooks

Use IPyParallel's load-balanced scheduler to farm training tasks out to the cluster

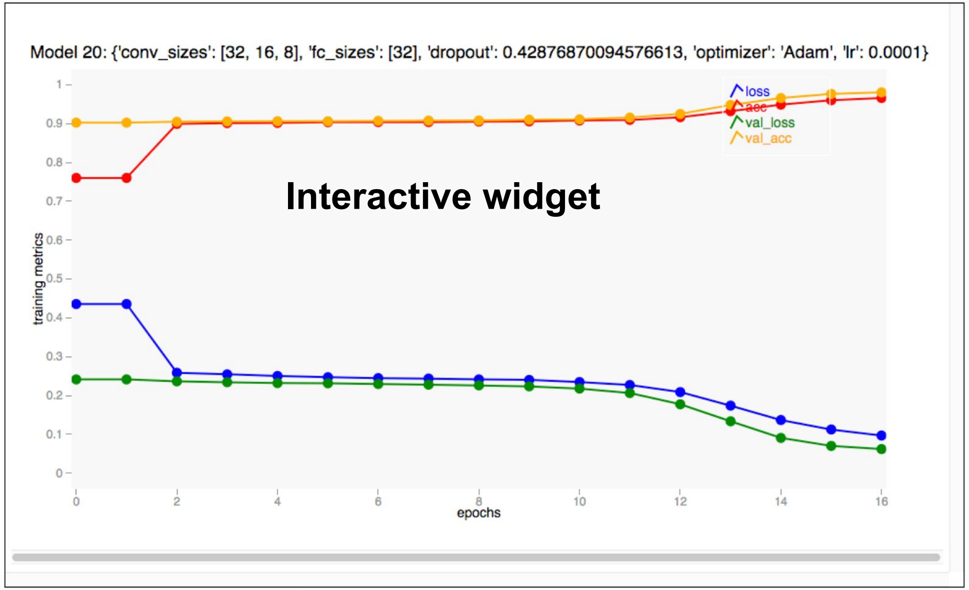
```
# Load-balanced scheduler
lv = client.load_balanced_view()

# Loop over hyper-param sets and queue tasks
for params in range(my_param_sets):
    results.append(lv.apply(my_train_function, params))
```

With everything in-notebook, seamless steering and analysis of tasks



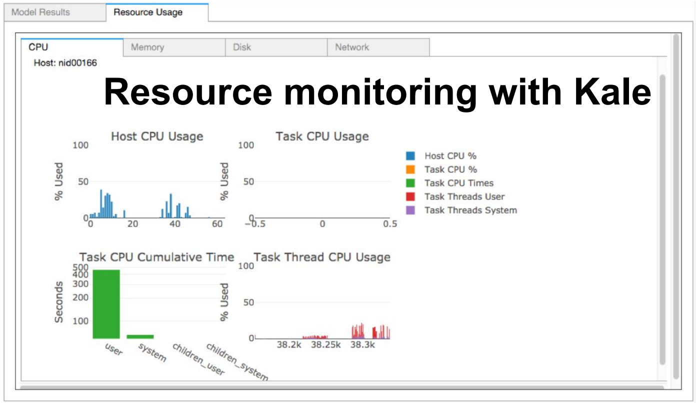


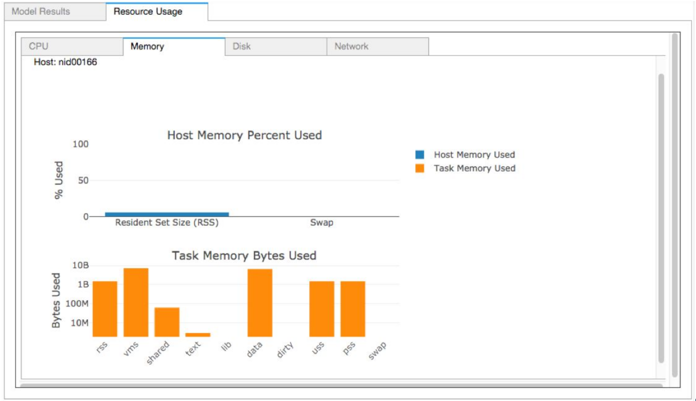
Interactive widgets and monitoring



Stop selected
Restart selected

index	status	epoch	conv_sizes	fc_sizes	dropout	optimizer	lr	loss	val_loss	acc	val_acc
15	Ended Traini…	15	[8, 64, 32]	[256]	0.01919	Adam	0.001	0.0067703…	0.058146…	0.99775	0.98596875
19	Ended Traini…	15	[64, 16, 64]	[256]	0.61802	Adam	0.001	0.0497609…	0.039992…	0.982625	0.98496875
21	Ended Traini…	15	[4, 4, 8]	[64]	0.13547	Adam	0.0001	0.0735504…	0.051296…	0.9740625	0.982
20	Ended Traini…	15	[32, 16, 8]	[32]	0.42877	Adam	0.0001	0.0971096…	0.062174…	0.96575	0.98021875
18	Ended Traini…	15	[8, 8, 16]	[128]	0.29008	Adam	0.01	0.0700781…	0.073561…	0.975390…	0.969625
16	Ended Traini…	15	[8, 8, 8]	[256]	0.30157	Nadam	0.01	0.1863622…	0.162577…	0.927140…	0.93790625





https://github.com/Jupyter-Kale/kale

Conclusion

At NERSC, we strive to provide capability and usability for Deep Learning on our supercomputers

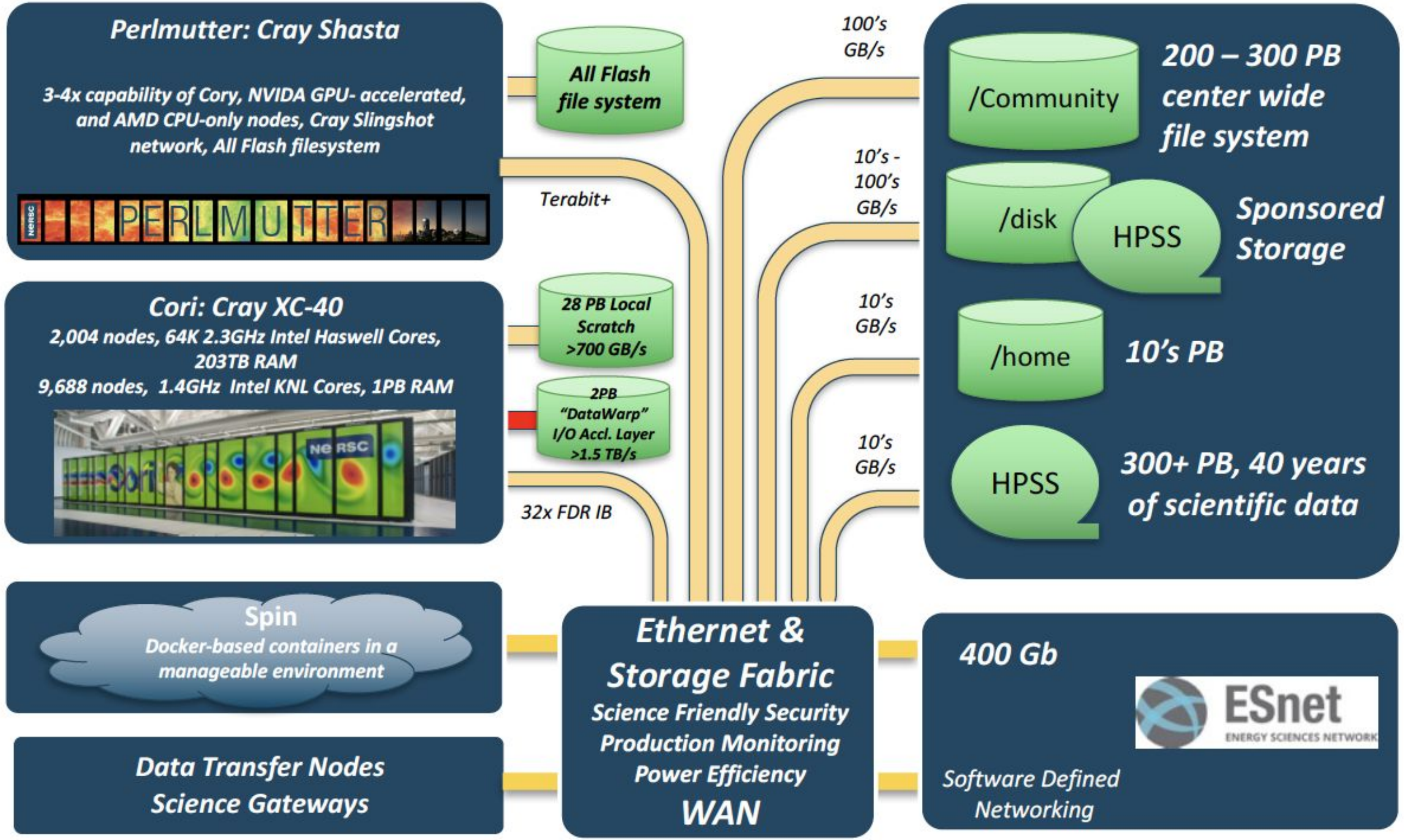
- Scalable, optimized software and methods
- Interactive workflows with Jupyter

We also get to work on some pretty fun DL projects

- Many didn’t have time to mention!
- Stay tuned for future projects
- Let us know if you’re interested in collaborating on something

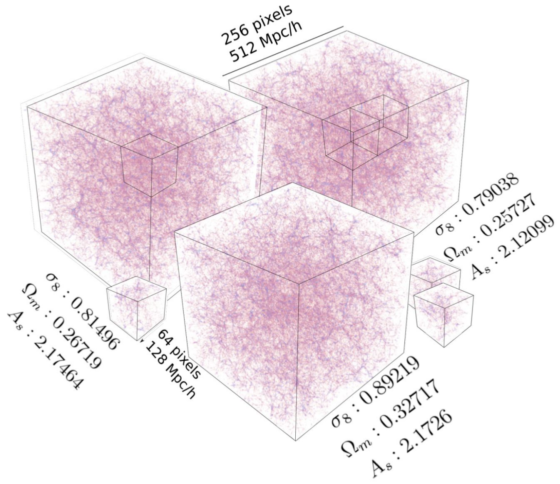
Thank You

NERSC in 2020



CosmoFlow goals

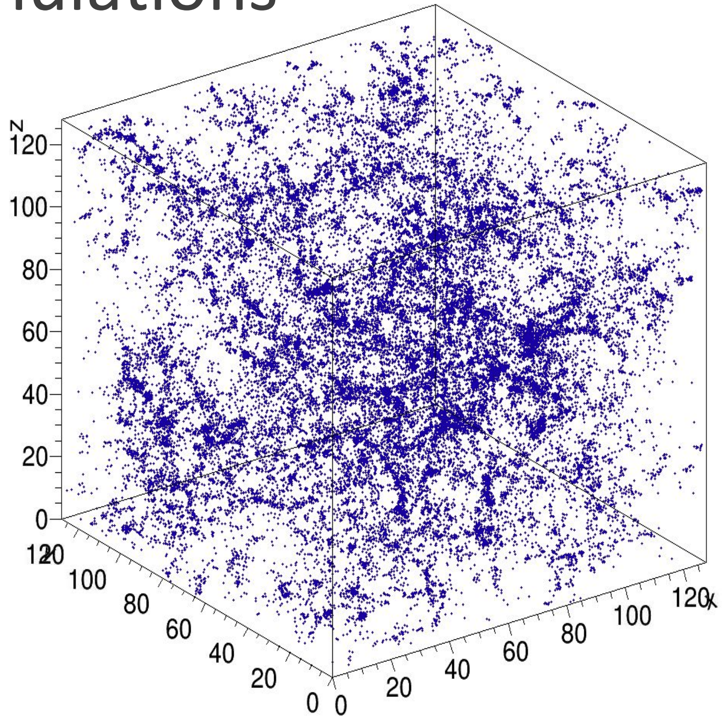
- Based on pioneering work by CMU team, scale up:
  - The dataset: 62GB $\longrightarrow$ 1.4TB
  - The problem size: 64^3 voxel training data $\longrightarrow$ 128^3 voxel training data
  - The number of parameters predicted: 2$\longrightarrow$ 3



- **Optimise performance of TensorFlow for 3D volumes on Cori.**
- **Run fast, effective multi-node training on all of Cori**
- **Predict 3 cosmological parameters from the dark matter distribution**

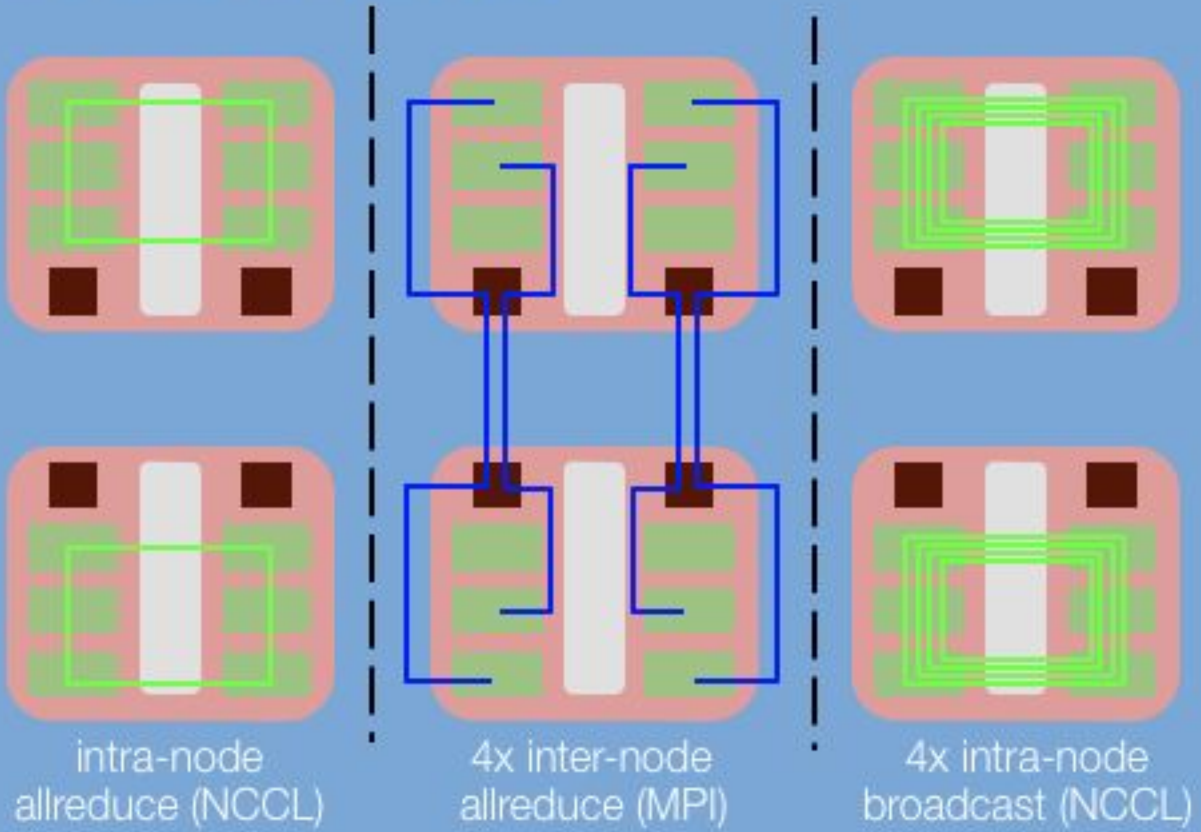
Training data: 3D dark matter simulations

- Ran fast 512h^{-1} Mpc^3 simulation volumes containing 512^3 dark matter particles
- Vary 3 cosmological parameters
  - $\boldsymbol{\Omega_m}$: proportion of matter in the universe (assuming $\Omega_m + \Omega_{DE} = 1$)
  - $\boldsymbol{\sigma_8}$: amplitude of mass fluctuations in the universe at 8Mpc/h scale
  - $\boldsymbol{N_s}$: scalar spectral index of spatial curvature of spacetime
- Total amount of data produced during simulation runs: ~100TB
- Training data used: ~1.4TB in TFRecord format



Example 128^3-voxel training sample showing 3D dark matter distribution.

Hybrid All-Reduce



- NCCL uses NVLink for high throughput, but ring-based algorithms latency-limited at scale
- hybrid NCCL/MPI strategy uses strengths of both
- one inter-node allreduce per virtual NIC
- MPI work overlaps well with GPU computation