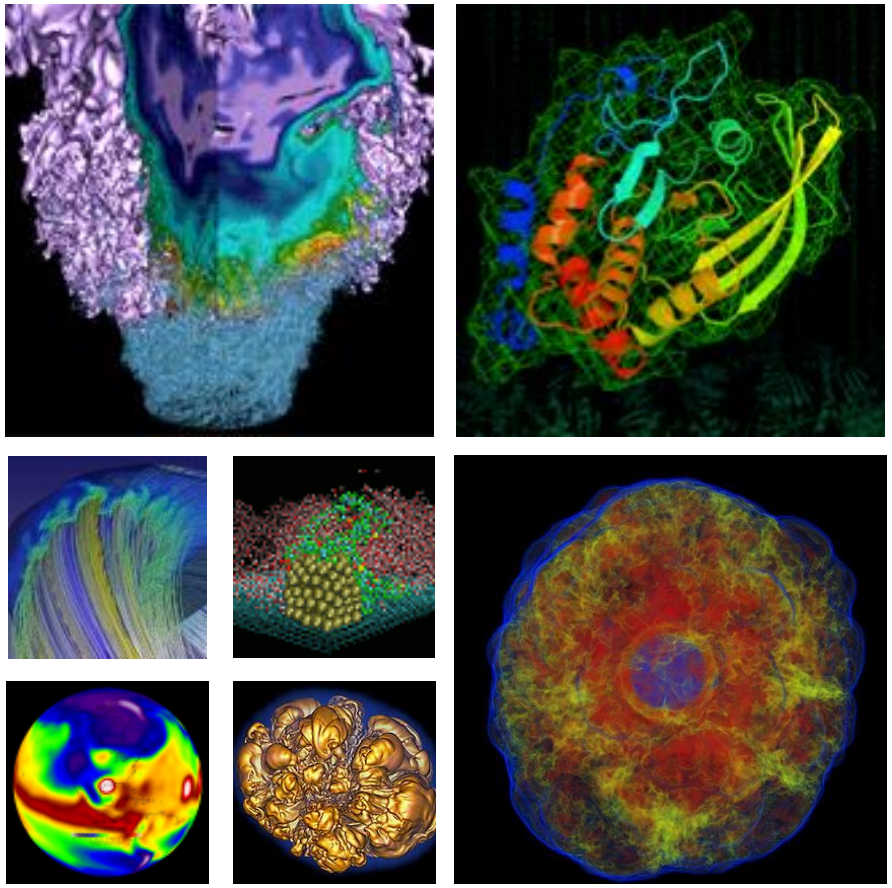
I/O for Deep Learning at Scale



Quincey Koziol
Principal Data Architect, NERSC
koziol@lbl.gov

IXPUG, September 24, 2019

Acknowledgments

- Prabhat, Wahid Bhimji, Debbie Bard, Thorsten Kurth, Jialin Liu (NERSC)
- Mike Houston, Sean Treichler, Josh Romero (NVIDIA)
- Lei Shao (Intel)
- Pete Mendygral, Mike Ringenburg (Cray)
- Gunes Baydin (Oxford)

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

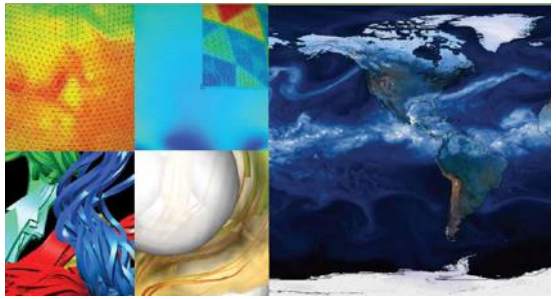
Outline

- **Introduction to NERSC**
- **Deep Learning for Science**
- **Case Studies**
  - Exascale Climate Analytics
  - etalumis
- **I/O Challenges for DL @ Scale**
- **Conclusions**

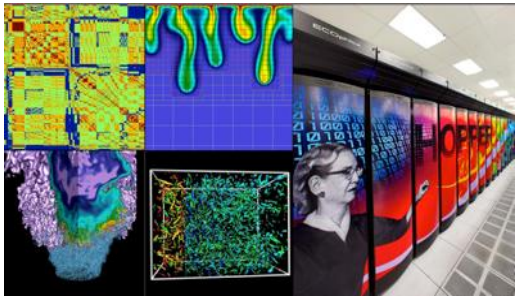
NERSC: the Mission HPC Facility for DOE Office of Science Research

U.S. DEPARTMENT OF ENERGY | Office of Science

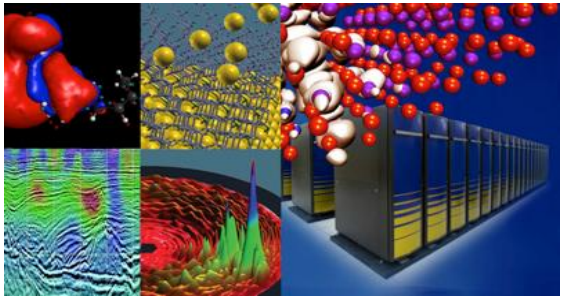
Largest funder of physical science research in the U.S.



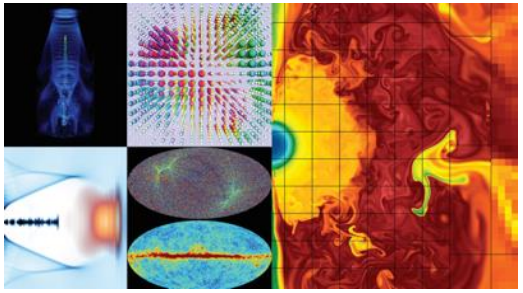
Bio Energy, Environment



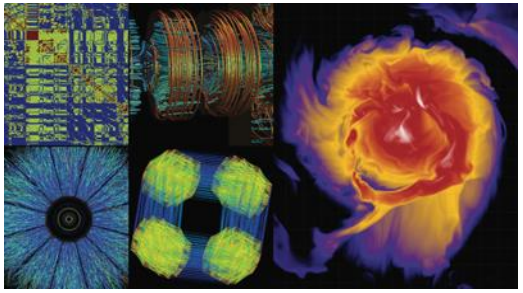
Computing



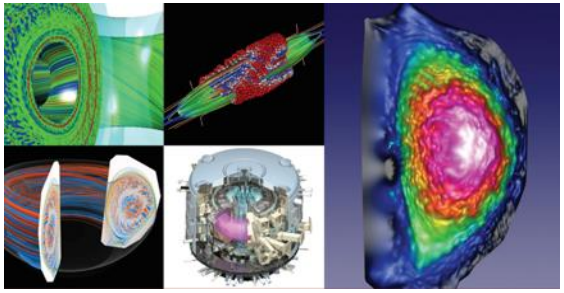
Materials, Chemistry, Geophysics



Particle Physics, Astrophysics



Nuclear Physics



Fusion Energy, Plasma Physics

7,000 users, 800 projects, 700 codes, 48 states, 40 countries, universities & national labs

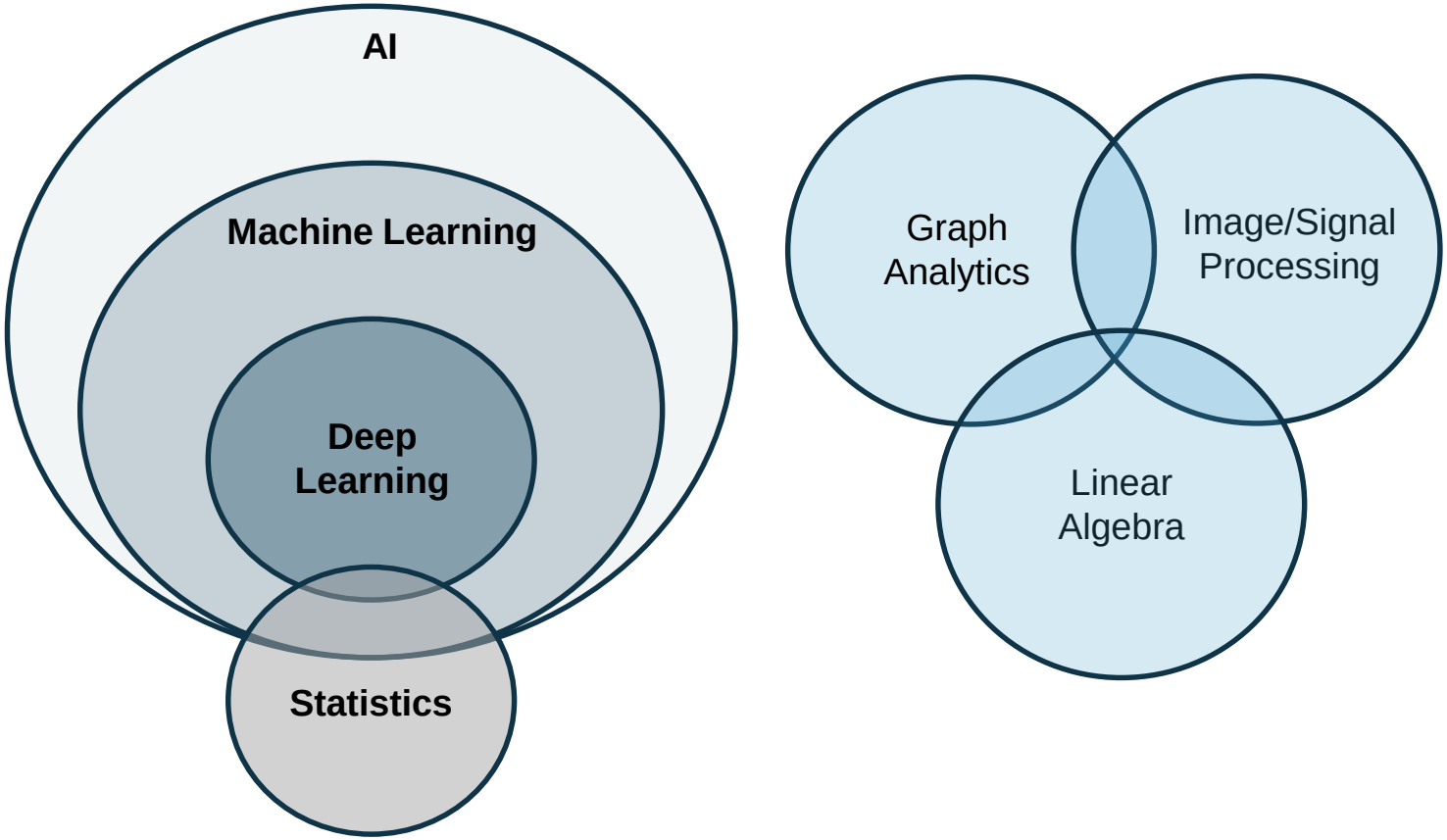
Cori supports Simulation and Data Workloads

- **Phase I: 2388 Intel Xeon “Haswell” nodes**
- **Phase II: 9688 Intel Xeon Phi “KNL” nodes**
- **1.5 PB NVRAM Burst Buffer, supporting 1.5TB/s I/O rates**

Outline

- Introduction to NERSC
- **Deep Learning for Science**
- Case Studies
  - Exascale Climate Analytics
  - etalumis
- I/O Challenges for DL @ Scale
- Conclusions

Data Analytics Methods



NERSC Big Data Stack

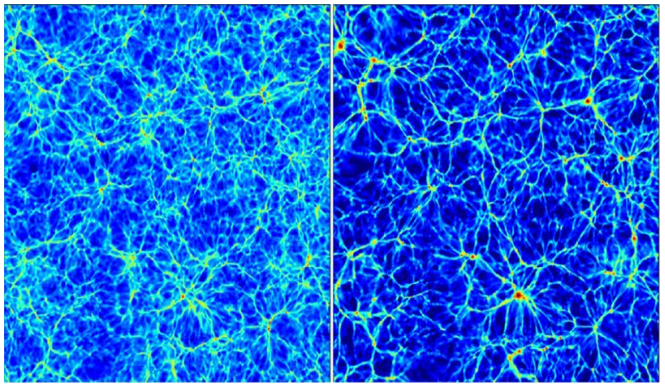
Capabilities	Technologies
Data Transfer + Access	globus online, GridFTP, jupyter, !m, python django, newt
Workflows	FireWorks, taskfarmer
Data Management	HDF, netCDF, ROOT Data Analysis Framework, mongoDB, MySQL, PostgreSQL
Data Analytics	python, Spark, TensorFlow, PYTORCH, R, julia, Caffe, K
Data Visualization	VisIt, ParaView

NERSC Deep Learning Stack

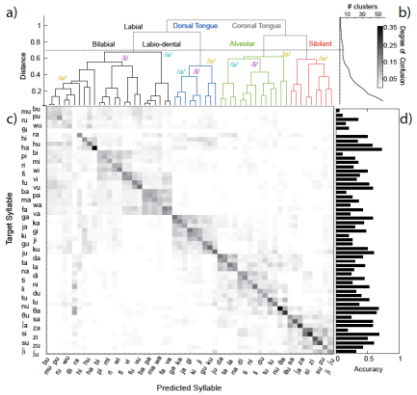
	Technologies
Deep Learning Frameworks	  Caffe PYTORCH CNTK, MXNet, Neon, …
Multi Node libraries	Cray Plugin Horovod GRPC MLSL MPI
Single Node libraries	MKL-DNN CuDNN
Hardware	CPUs (KNL) GPUs FPGAs Accelerators

Deep Learning for Science

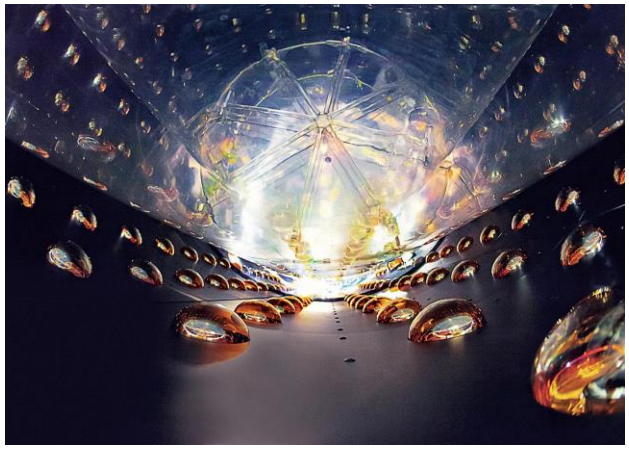
Modeling galaxy shapes



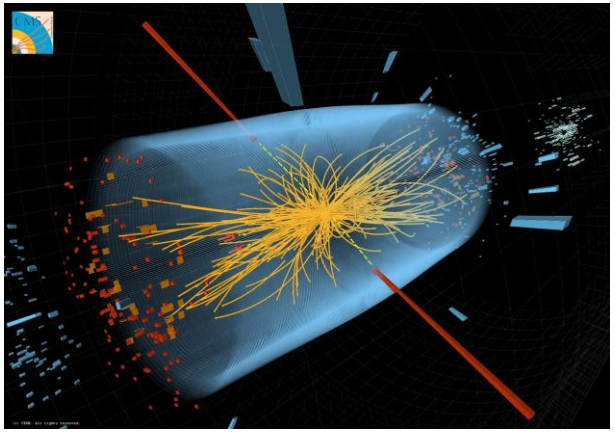
Generating cosmology mass maps



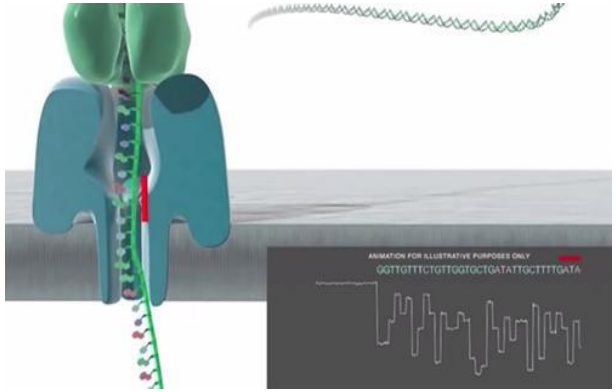
Decoding speech from ECoG



Clustering Daya Bay events



LHC Signal/Background classification



Oxford Nanopore sequencing

U.S. DEPARTMENT OF ENERGY | Office of Science

https://www.oreilly.com/ideas/a-look-at-deep-learning-for-science

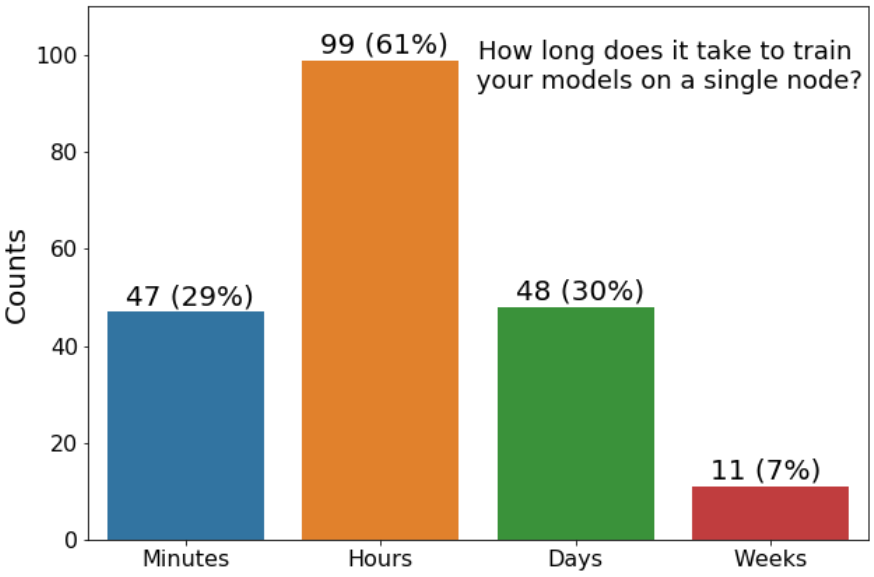
BERKELEY LAB

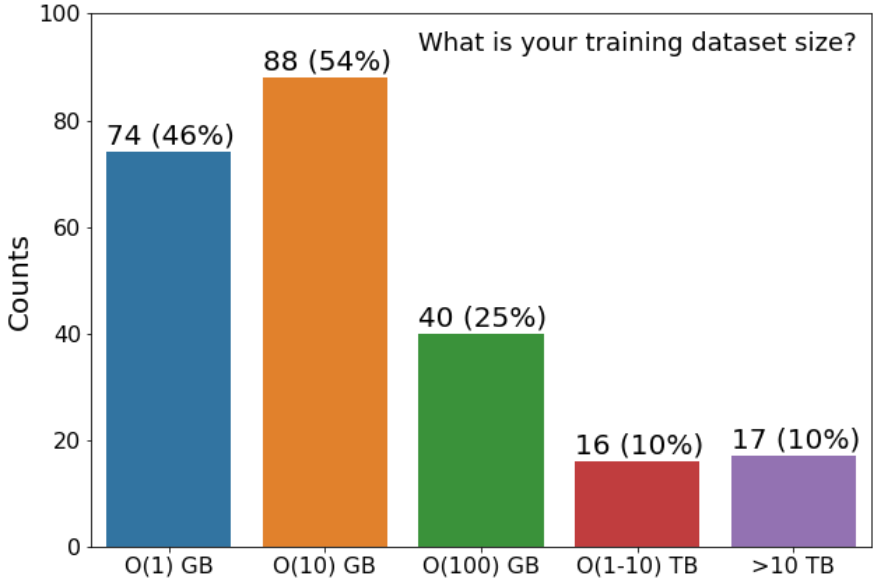
	HEP			BER		BES		NP	FES
	Astronomy	Cosmology	Particle Physics	Climate	Genomics	Light Sources	Materials	Heavy Ion Colliders	Plasma Physics
Classification	X		X	X	X	X	X	X	X
Regression	X	X	X	X	X	X	X	X	X
Clustering	X	X	X	X	X	X	X	X	X
Dimensionality Reduction				X				X	
Surrogate Models	X	X	X	X			X	X	X
Design of Experiments		X		X		X	X		X
Feature Learning	X	X	X	X	X	X	X	X	X
Anomaly Detection	X		X	X		X		X	

	HEP			BER		BES		NP	FES
	Astronomy	Cosmology	Particle Physics	Climate	Genomics	Light Sources	Materials	Heavy Ion Colliders	Plasma Physics
Classification	CNNs, Graph NNs, RNNs								
Regression									
Clustering	Auto-encoders								
Dimensionality Reduction									
Surrogate Models	VAEs, GANs								
Design of Experiments	RL								
Feature Learning	✓								
Anomaly Detection	?								

DL adoption by NERSC user community

- ~200 users exercising the DL stack
- 160 respondents to 2018 'ML@NERSC' survey





Why Scale Deep Learning?

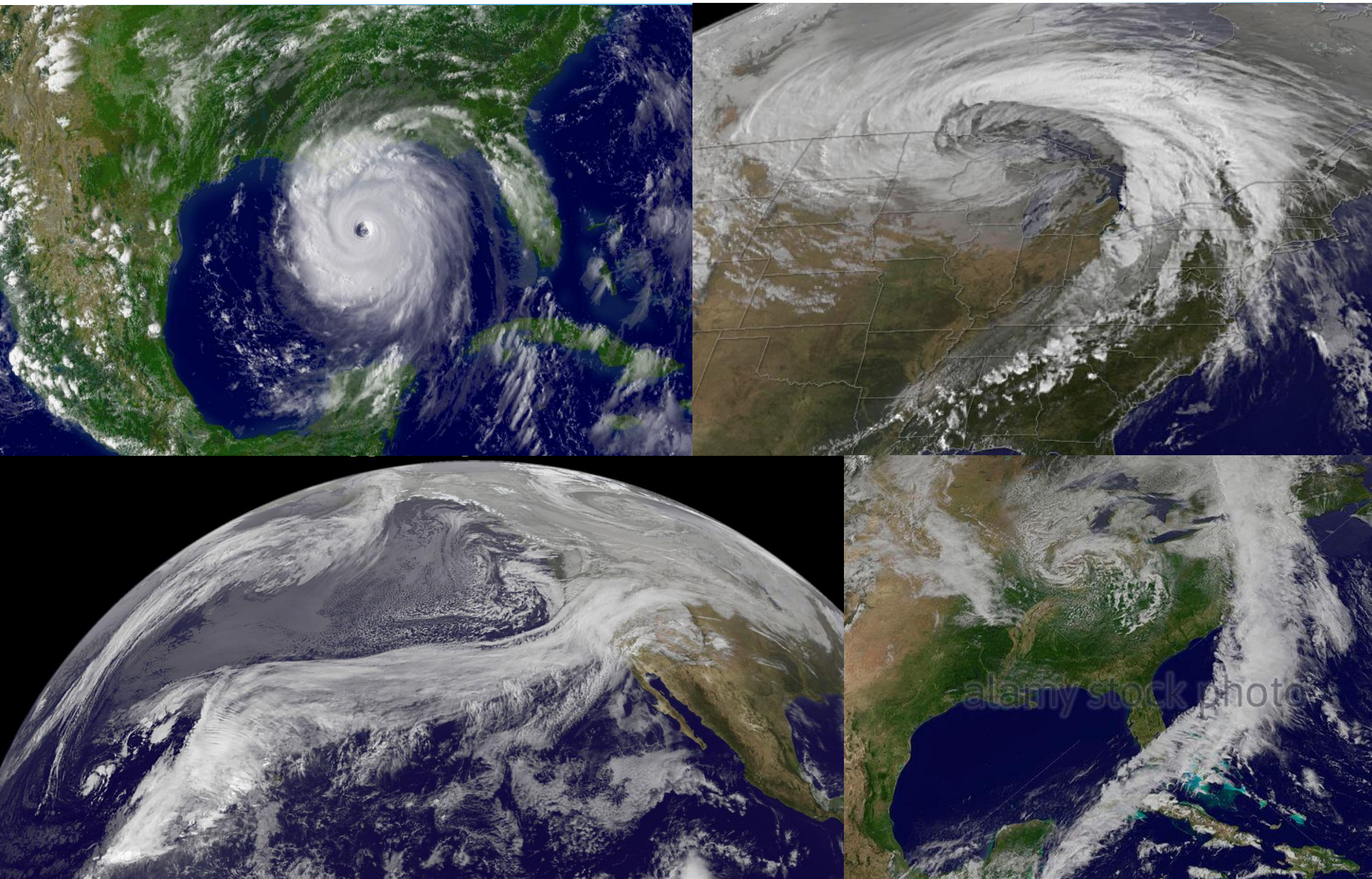
- **Day / Week-long runtimes for O(100) GB - O(1) TB sized datasets**
  - ‘Classical’ convolutional architectures
  - More advanced architectures (Hybrid CNN + LSTM, spacetime convolutions)
- **Hyper-Parameter optimization is important**
- **Large computational demands**

- **Problem is well suited for HPC systems**

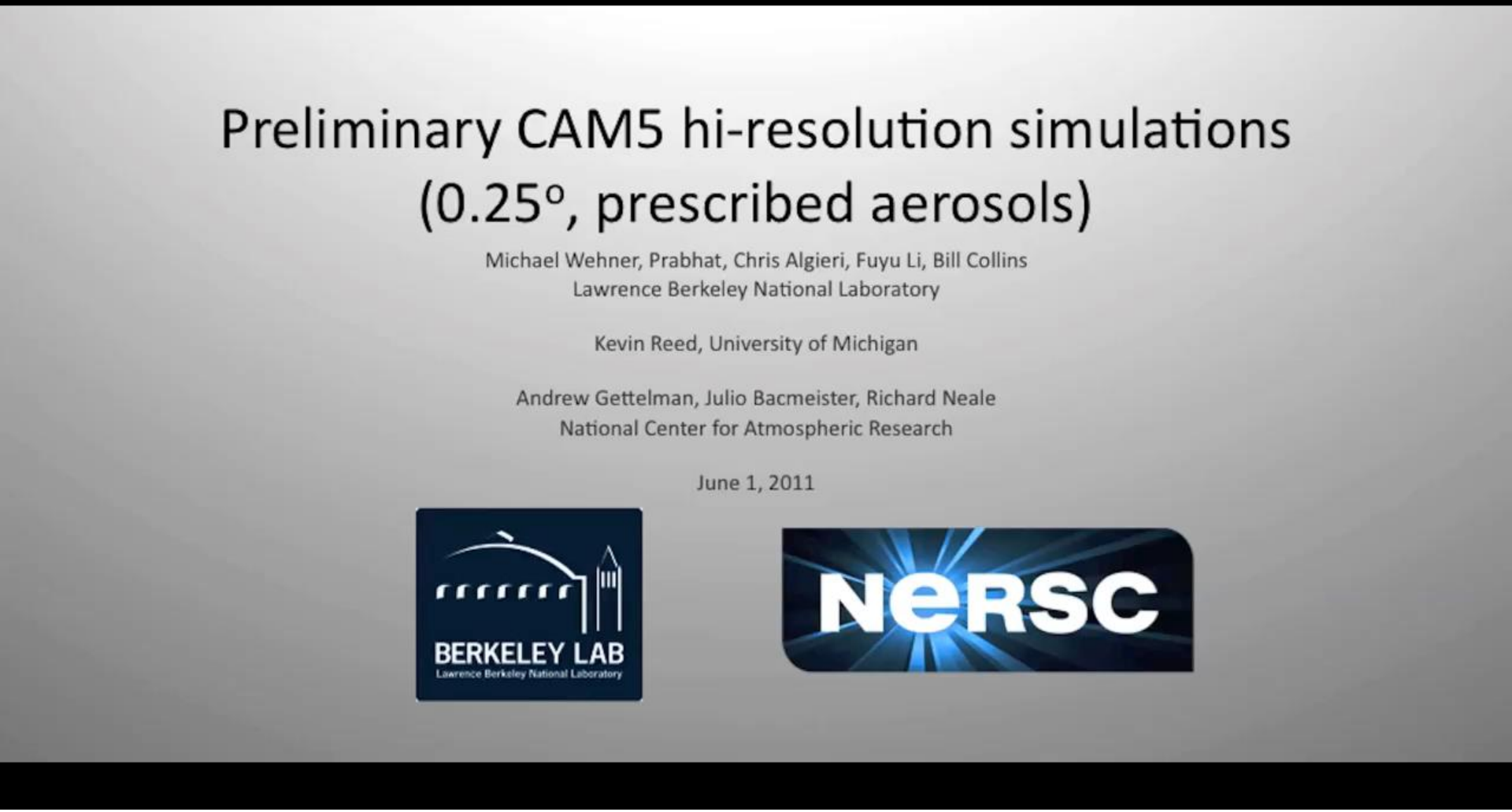
Outline

- Introduction to NERSC
- Deep Learning for Science
- **Case Studies**
  - Exascale Climate Analytics
  - etalumis
- I/O Challenges for DL @ Scale
- Conclusions

Characterizing Extreme Weather…

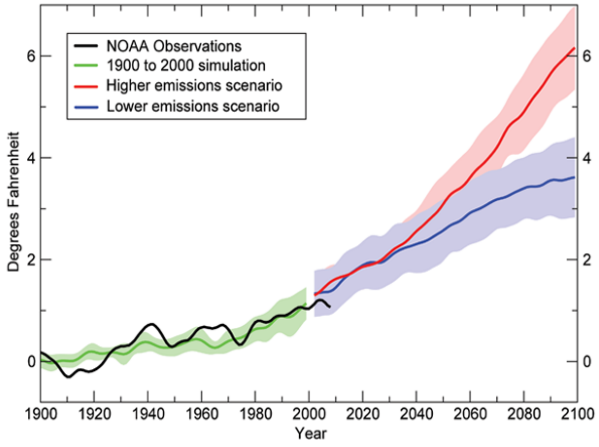


… in a changing Climate

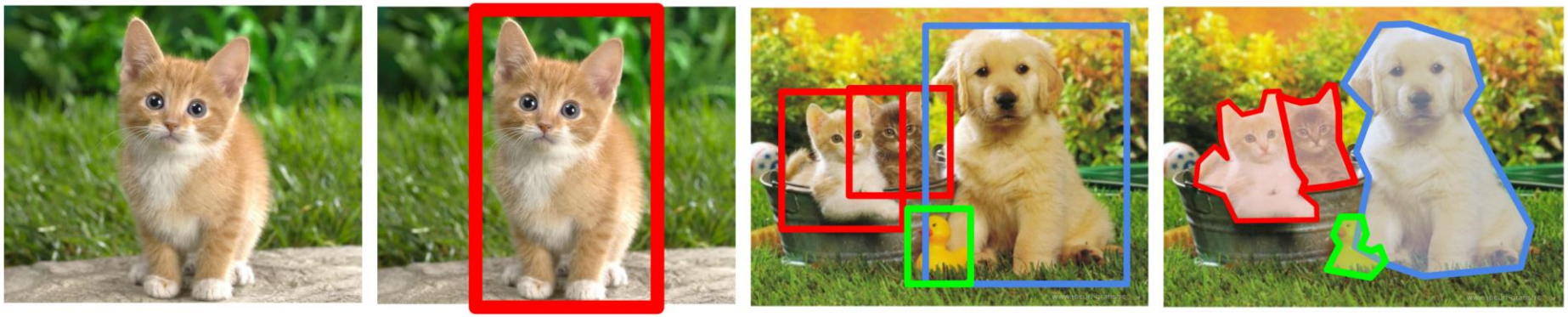


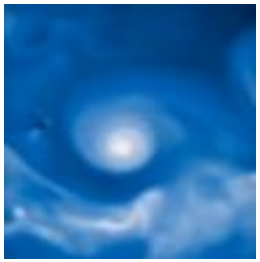
Understanding Climate Change

- **How will the global weather change by 2100?**
  - Will the Earth warm up by 1.5 or 2.0 C?
  - Will the sea level rise by 1 or 2 feet?
- **How will extreme weather change by 2100?**
  - Will there be more hurricanes?
  - Will they become more intense?
  - Will they make landfall more often?
  - Will atmospheric rivers carry more water?
  - Will they make landfall over California?
  - Will they mitigate droughts?
  - Will they cause heavy precipitation and flooding?

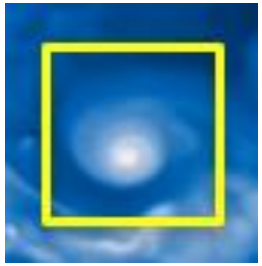


Climate Science Deep Learning Tasks

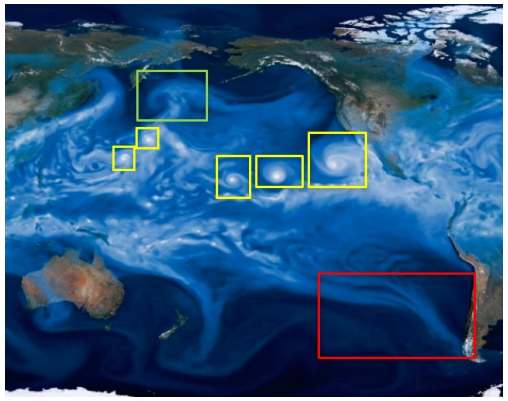




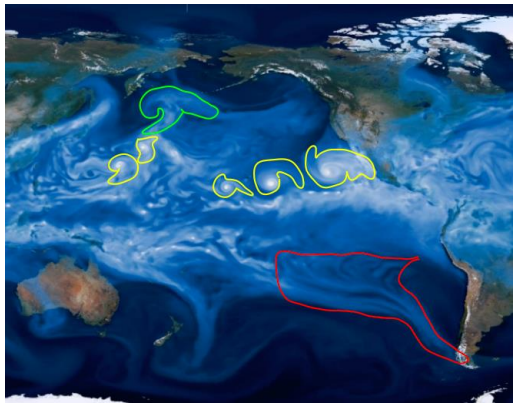
Liu, et al
ABDA'16



Racah, et al
NIPS'17

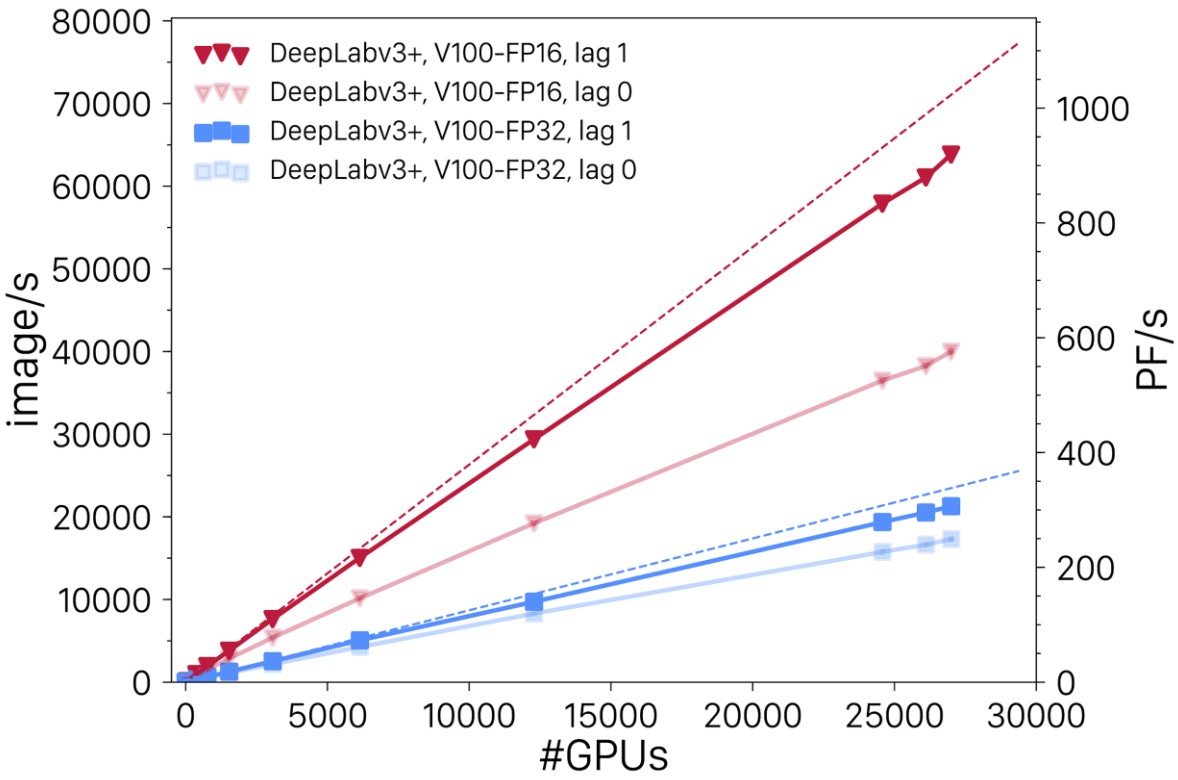


Racah, et al, NIPS'17
Kurth, et al, SC'17



Kurth, et al, SC'18

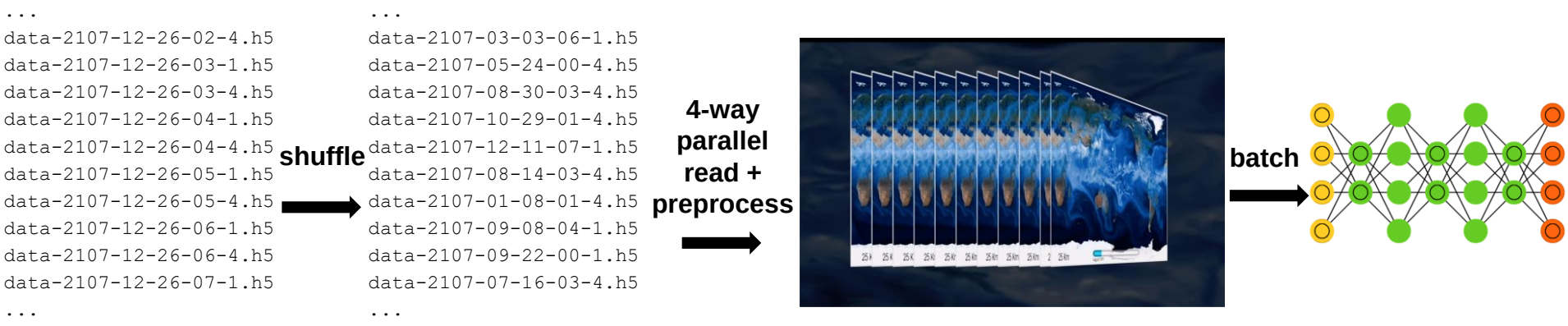
Extreme Scaling



- 4560 Summit nodes, 27,360 Volta GPUs, @ ORNL
- **1.13 EF** peak performance (16-bit)

On-Node I/O Pipeline

- Files are in HDF5 with single sample + label/file
- List of filenames passed to TensorFlow Dataset API (tf.data)
- HDF5 serialization bottleneck addressed with multiprocessing
- Extract and batch using tf.data input pipeline

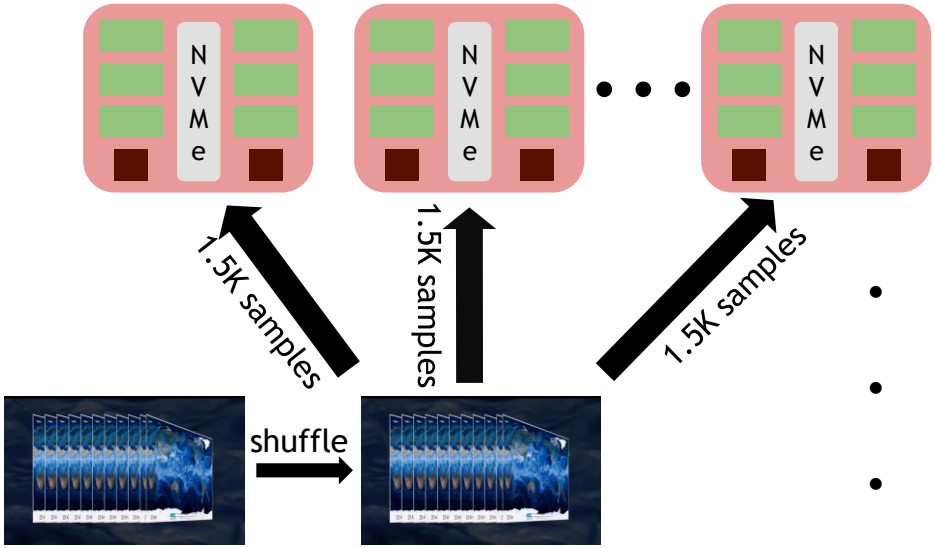


Data Management Overview

- **Shuffling / loading / preprocessing / feeding 20 TB dataset**
  - Ensure that composition of a batch is random
- **Sustained Bandwidth**
  - ~61 MB/sample × ~65,000 samples/s @ 27K GPUs → ~3.8 TB/s
  - Typical distributed FS bandwidth: ~400 GB/s → ~8x performance gap
  - Typical Burst Buffer bandwidth: ~2 TB/s → ~2x performance gap
- **Random reads / no writes:**
  - Modern HPC file systems are not optimized for this!
- **Must work around HDF5 library limitations**
  - No threading support ☹
- **Use available tools/packages to achieve this along with recommended TensorFlow data ingestion method**

Data Staging

Dataset Size	Required BW (27K GPUs)	GPFS/LUSTRE	BurstBuffer	NVMe or DRAM
20 TB (~63K samples)	3.8 TB/s	~400 GB/s	~2 TB/s	~26 TB/s



- 250 training samples/GPU (~15 GB), sample w/replacement
- Each file will be read at most once from FS
- Files shared between nodes via MPI (mpi4py)

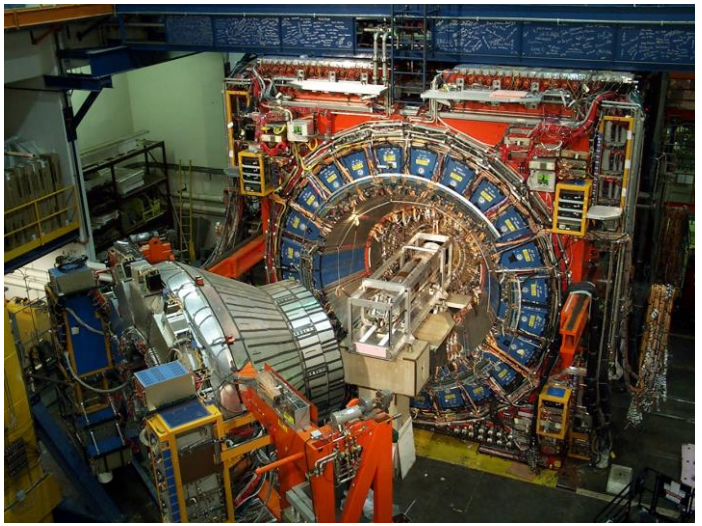
Outline

- Introduction to NERSC
- Deep Learning for Science
- **Case Studies**
  - Exascale Climate Analytics
  - etalumis
- I/O Challenges for DL @ Scale
- Conclusions

Probabilistic Programming and High-Energy Physics - “etalumis”

Baydin, A.G., Heinrich, L., Bhimji, W., Gram-Hansen, B., Louppe, G., Shao, L., Prabhat, Cranmer, K., Wood, F. 2018. ***Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model*** *arXiv preprint arXiv:1807.07706.*

https://arxiv.org/abs/1807.07706



Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model

Atilim Gunes Baydin, Lukas Heinrich, Wahid Bhimji, Bradley Gram-Hansen, Gilles Louppe, Lei Shao, Prabhat, Kyle Cranmer, Frank Wood

(Submitted on 20 Jul 2018)

We present a novel framework that enables efficient probabilistic inference in large-scale scientific models by allowing the execution of existing domain-specific simulators as probabilistic programs, resulting in highly interpretable posterior inference. Our framework is general purpose and scalable, and is based on a cross-platform probabilistic execution protocol through which an inference engine can control simulators in a language-agnostic way. We demonstrate the technique in particle physics, on a scientifically accurate simulation of the tau lepton decay, which is a key ingredient in establishing the properties of the Higgs boson. High-energy physics has a rich set of simulators based on quantum field theory and the interaction of particles in matter. We show how to use probabilistic programming to perform Bayesian inference in these existing simulator codebases directly, in particular conditioning on observable outputs from a simulated particle detector to directly produce an interpretable posterior distribution over decay pathways. Inference efficiency is achieved via inference compilation where a deep recurrent neural network is trained to parameterize proposal distributions and control the stochastic simulator in a sequential importance sampling scheme, at a fraction of the computational cost of Markov chain Monte Carlo sampling.

etalumis

HEP packages like

- SHERPA
- GEANT
- PYTHIA
- Herwig++
- MadGraph

are essentially very accurate probabilistic algorithms

We focus our attention to SHERPA (C++)

We run etalumis code on Cori at NERSC using Shifter:

shifterimg -v pull docker:etalumis/sherpa:latest

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Common trace types in SHERPA

440 trace types (address sequences) encountered over 1.6M executions

Trace type: unique sequencing of addresses (with different sampled values)

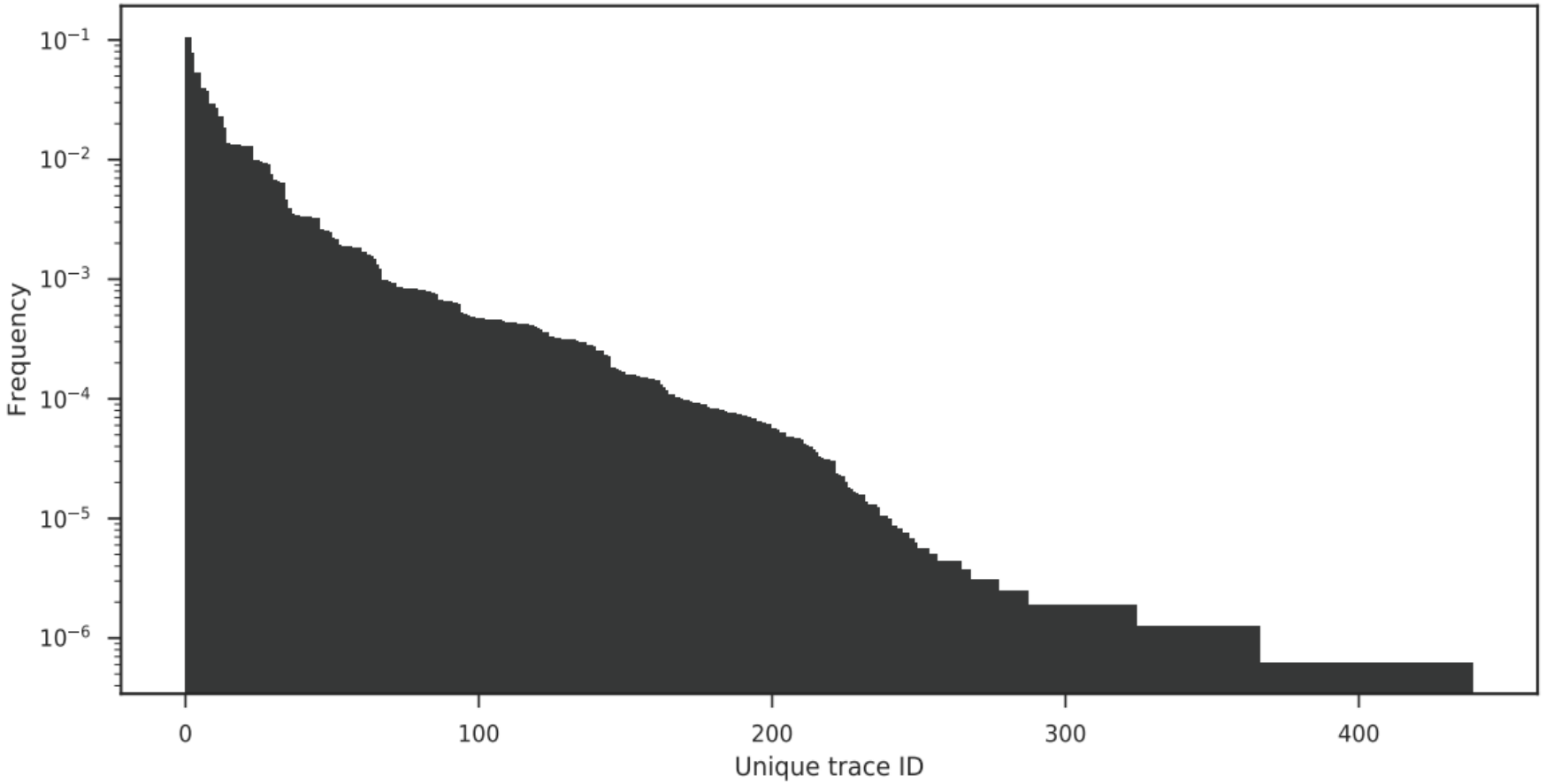
Freq.	Length	Addresses (showing controlled only)
0.106	72	A1, A2, A3, A5, A6, A32, A33, A31
0.105	41	A1, A2, A3, A5, A6, A499, A31
0.078	1,780	A1, A2, A3, A5, A6, A7, A8, A9, A10, A31
0.053	188	A1, A2, A3, A5, A6, A7, A8, A9, A10, A17, A18, A26, A31
0.053	100	A1, A2, A3, A5, A6, A7, A8, A9, A10, A17, A18, A99, A100, A101, A102, A31
0.039	56	A1, A2, A3, A5, A6, A499, A17, A18, A26, A31
0.039	592	A1, A2, A3, A5, A6, A499, A17, A18, A99, A100, A101, A102, A31
0.038	162	A1, A2, A3, A5, A6, A7, A8, A9, A10, A17, A500, A99, A100, A101, A102, A31
0.030	240	A1, A2, A3, A5, A6, A7, A8, A9, A10, A17, A18, A20, A21, A41, A42, A26, A99, A100, A101, A102, A31
0.029	836	A1, A2, A3, A5, A6, A7, A8, A9, A10, A17, A18, A20, A21, A41, A42, A99, A100, A101, A102, A26, A31
0.027	643	A1, A2, A3, A5, A6, A7, A8, A9, A10, A17, A507, A99, A100, A101, A102, A31
0.023	135	A1, A2, A3, A5, A6, A7, A8, A9, A10, A17, A18, A20, A21, A41, A42, A44, A45, A26, A99, A100, A101, A102, A31
0.023	485	A1, A2, A3, A5, A6, A7, A8, A9, A10, A17, A18, A20, A21, A41, A42, A44, A45, A99, A100, A101, A102, A26, A31
...		

Accessing the Trace Training Data

- etalumis' *15m test dataset*
  - 1.7TB, with 15 million trace files, each averaging 110KB
  - Stored on Lustre file system on Cori, with another copy in Burst Buffer
- For each training iteration, each process reads in a local-batch # of traces, e.g., 64 traces
- For each iteration, the global batch size is <# of ranks> * <local batch size>, e.g., 1024 * 64 = 64k
- Initially, I/O in etalumis is similar to HPC file-per-process access

Common trace types in SHERPA

440 trace types (address sequences) encountered over 1.6M executions



(c) Distribution of trace types, sorted in decreasing frequency.

Data and I/O Challenges

I/O Challenges:

- Random access due to shuffling each iteration and epoch
- Number of input files is large
- No parallel I/O support in current DL system, e.g., PyTorch

File Format Challenges:

- Complex data and file structure
- Data duplication

Metadata Optimization

Merge Many Small Files into Few Large Files

- Original: 15 million files, w/1 trace per file
- After Merging: 150 files, w/100k traces per file

File Handle Caching

- Maintain cache of file handles
- Keep file open during training

Data I/O Optimization

Trace Structure Pruning

- Remove unnecessary data structures
  - Disk space and memory consumption savings

Sorting

- Offline sorting based on controlled address length
  - Random access → *sequential* access

Distributed I/O Loading

- Implementation based on PyTorch's *Sampler*
- Round-robin assign local batches to each worker
- Shuffle within each worker's local batch list

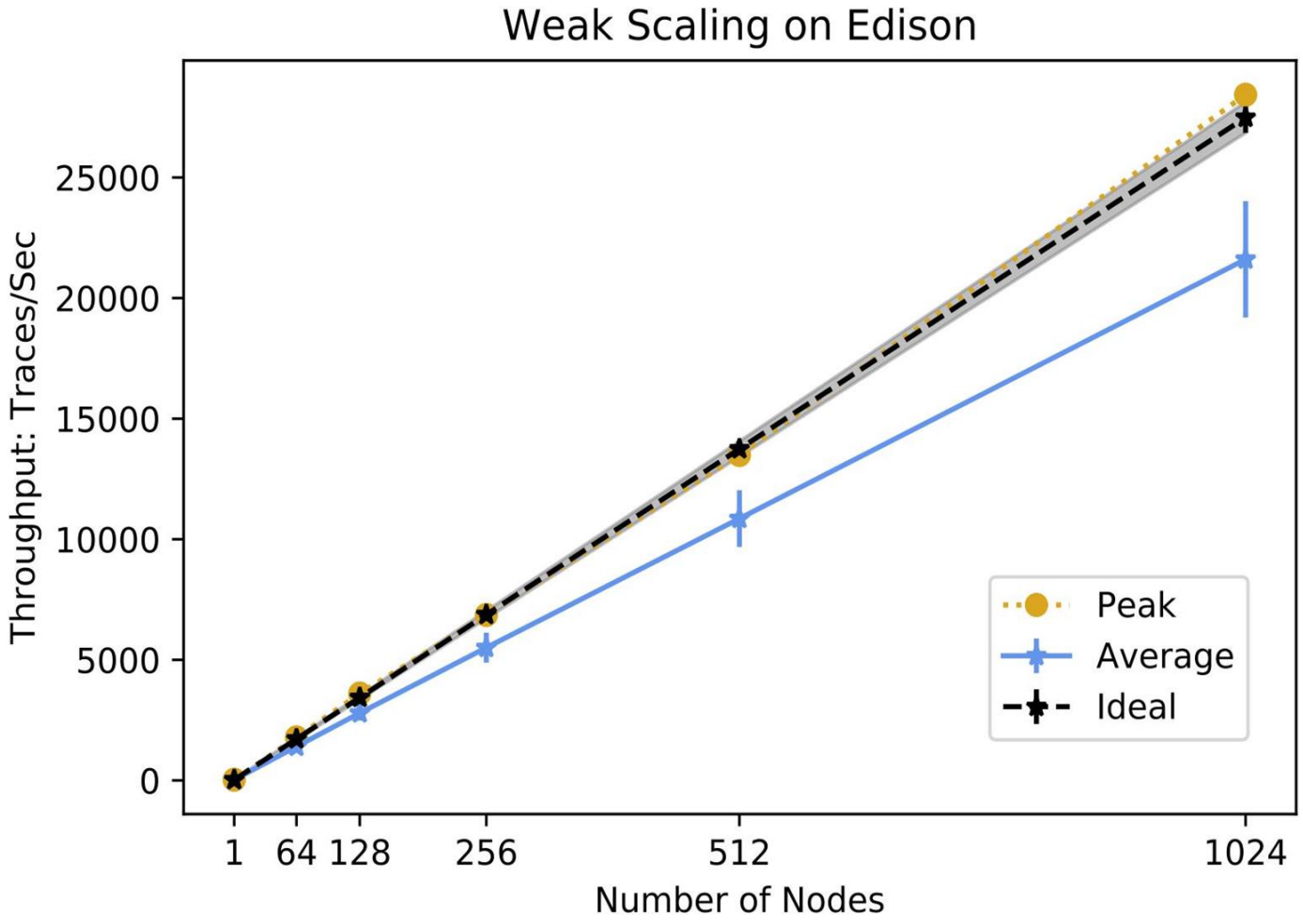
Efficient I/O ➔ Scalable Training

Before:

>75% of total runtime spent in I/O

After:

I/O reduced to <5%



Outline

- Introduction to NERSC
- Deep Learning for Science
- Case Studies
  - Exascale Climate Analytics
  - etalumis
- **I/O Challenges for DL @ Scale**
- Conclusions

I/O Challenges for DL @ Scale

- **DL I/O workloads are extremely demanding, with both Data and Metadata Issues:**

HPC Simulation	Deep Learning
Write-Once / ***Read-Never***	Write-Once / ***Read-Always***
Contiguous Large I/O to Sequence of Files	**Random Small I/O to Random Files**
O(10) of TBs in O(1000) files	O(10) of TBs in **O(100,000) files**

- **Lustre and GPFS file systems typically can't keep up**
  - Burst Buffer and node-local NVMe storage have been critical

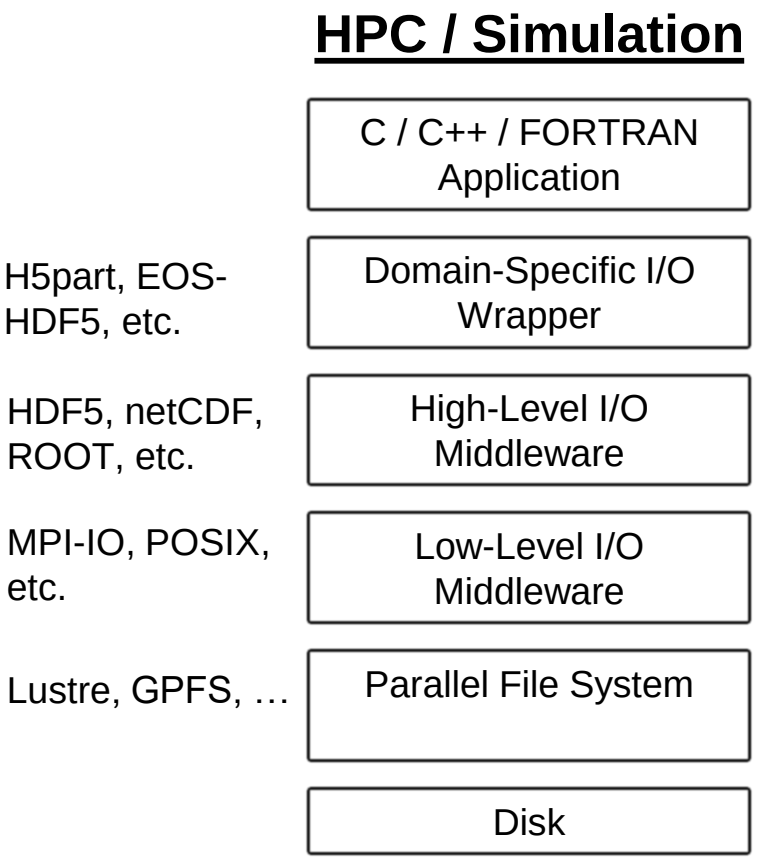
I/O Challenges for DL @ Scale

- **Applications are very young and unstable**
  - DL frameworks only 2-3 years old, may not last another 2-3
  - Many load imbalances in compute, communication, and I/O
  - Come from a culture of academia & industry, not HPC centers
    - Still ”learning to scale”
- **Data Management & I/O are not “hot topics”**
  - I/O is typically last consideration for application developer
  - I/O isn’t “interesting”, just “infrastructure”
- **Ingest pipelines for loading scientific data (HDF5, NetCDF, …) into DL frameworks are not optimized**
  - Need multi-threaded support, different tuning, etc.

Outline

- Introduction to NERSC
- Deep Learning for Science
- Case Studies
  - Exascale Climate Analytics
  - etalumis
- I/O Challenges for DL @ Scale
- **Conclusions**

Conclusion


HPC / Simulation
C / C++ / FORTRAN Application
H5part, EOS-HDF5, etc.
Domain-Specific I/O Wrapper
HDF5, netCDF, ROOT, etc.
High-Level I/O Middleware
MPI-IO, POSIX, etc.
Low-Level I/O Middleware
Lustre, GPFS, …
Parallel File System
Disk

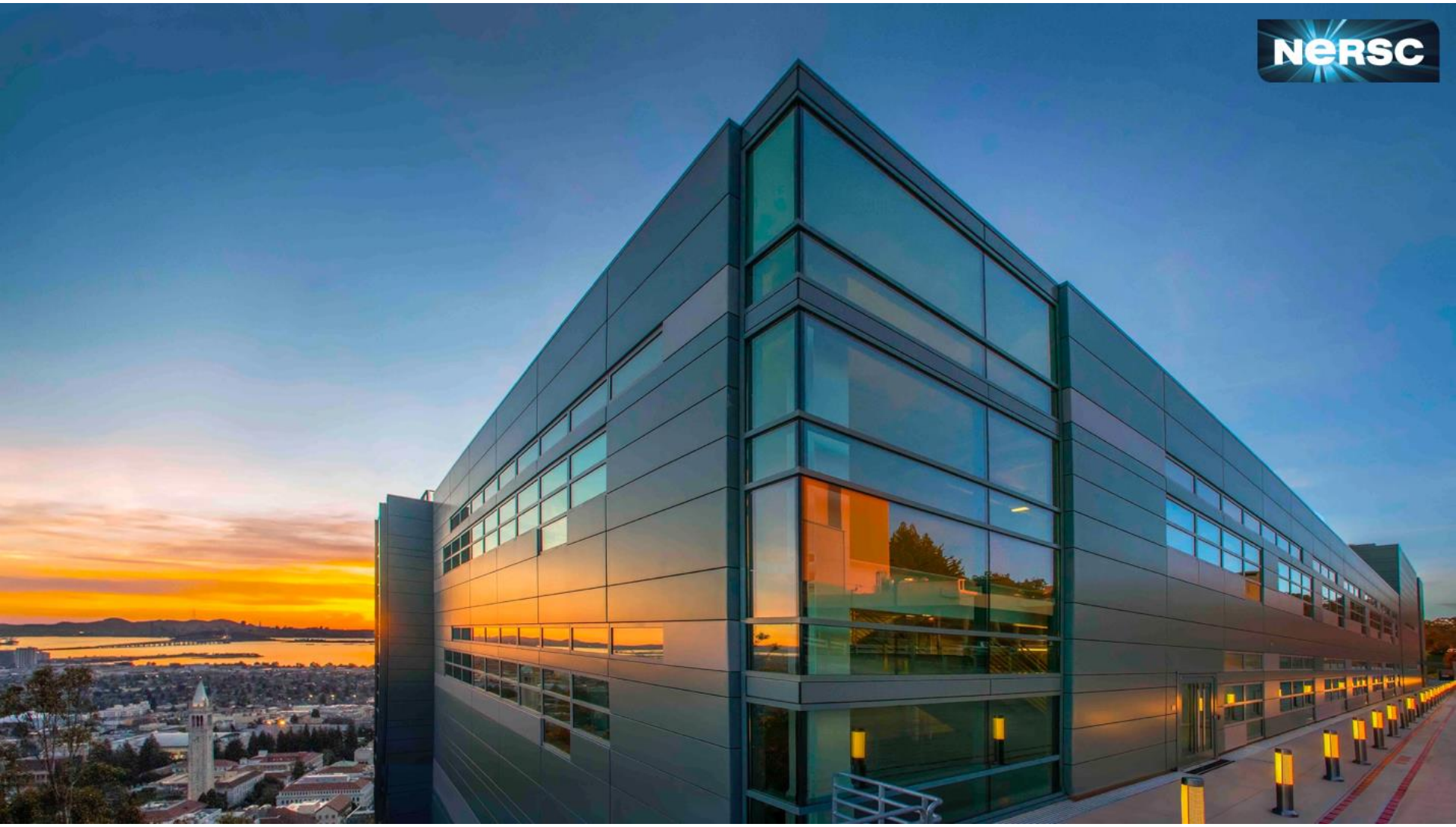
Conclusion

	HPC / Simulation	Deep Learning / Analytics	
	C / C++ / FORTRAN Application	Python / Julia / … Application	
H5part, EOS-HDF5, etc.	Domain-Specific I/O Wrapper	Domain-Specific Wrapper	TensorFlow, PyTorch, Caffe, …
HDF5, netCDF, ROOT, etc.	High-Level I/O Middleware	???	
MPI-IO, POSIX, etc.	Low-Level I/O Middleware	???	
Lustre, GPFS, …	Parallel File System	???	
	Disk	???	

Conclusion

We need a new I/O Middleware Stack for DL workloads!

	HPC / Simulation	Deep Learning / Analytics	
	C / C++ / FORTRAN Application	Python / Julia / … Application	
H5part, EOS-HDF5, etc.	Domain-Specific I/O Wrapper	Domain-Specific Wrapper	TensorFlow, PyTorch, Caffe, …
HDF5, netCDF, ROOT, etc.	High-Level I/O Middleware	Deep Learning I/O Middleware	Adapt HDF5 or create new?
MPI-IO, POSIX, etc.	Low-Level I/O Middleware	Low-Level I/O Middleware	Need something like MPI-IO
Lustre, GPFS, …	Parallel File System	Object Store (DAOS, Rados, …)	Read-heavy; high data and metadata rates
	Disk	NVRAM	IOPs & Random I/O



Questions?

koziol@lbl.gov