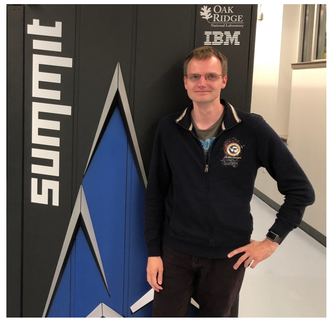
Exascale Deep Learning for Climate Analytics

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

GTC Silicon Valley
03/17/2019, San Jose, CA

The Team



Thorsten Kurth


Sean Treichler

Josh Romero


Mayur Mudigonda


Nathan Luehr


Everett Phillips


Ankur Mahesh


Michael Matheson


Jack Deslippe


Massimiliano Fatica


Prabhat

Michael Houston

Socio-Economic Impact of Extreme Weather Events

- tropical cyclones and atmospheric rivers have major impact on modern economy and society
- CA: 50% of rainfall through atmospheric rivers
- FL: flooding, influence on insurance premiums and home prices
- $200B worth of damage in 2017
- costs of ~$10B/event for large events

Katrina 2005

Berkeley 2019

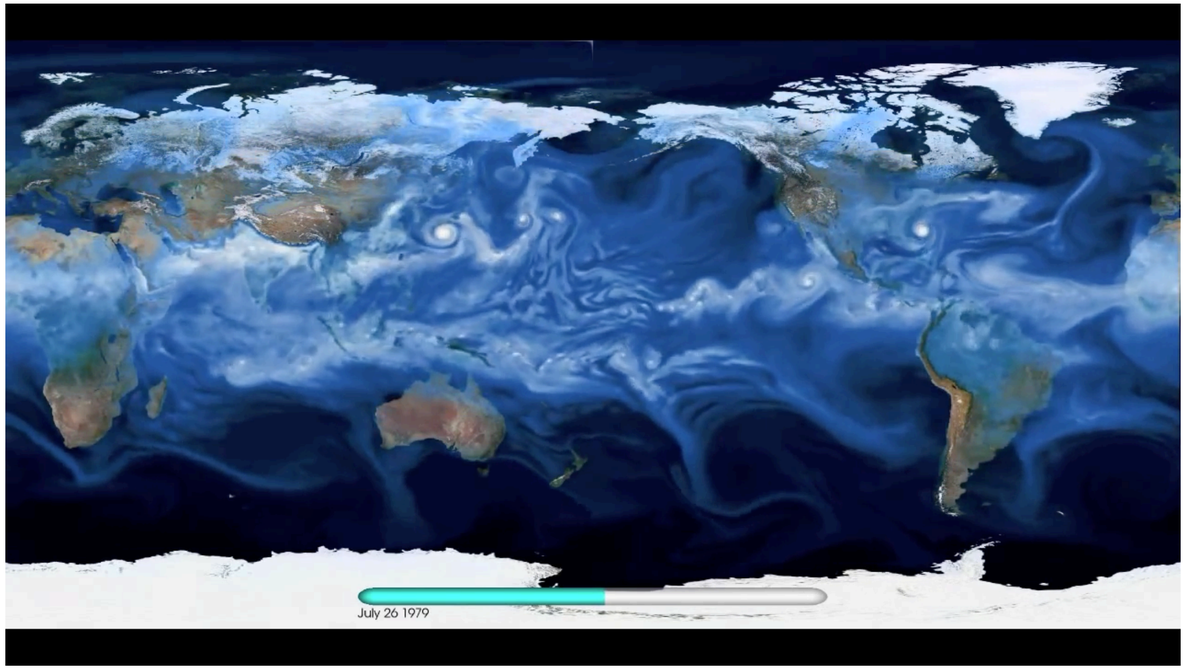
Harvey 2017

Santa Rosa 2018

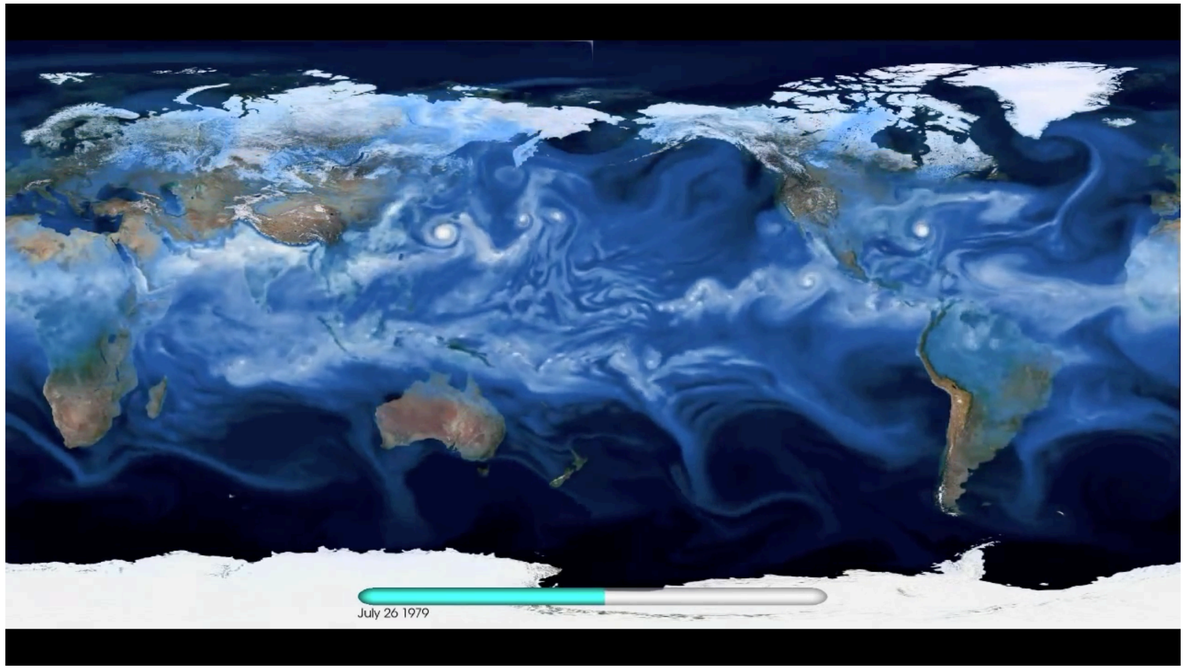
Understanding Extreme Weather Phenomena

- will there be more hurricanes?
- will they be more intense?
- will they make landfall more often?
- will atmospheric rivers carry more water?
- can they help mitigate droughts and decrease risk of forest fires?
- will they cause flooding and heavy precipitation?

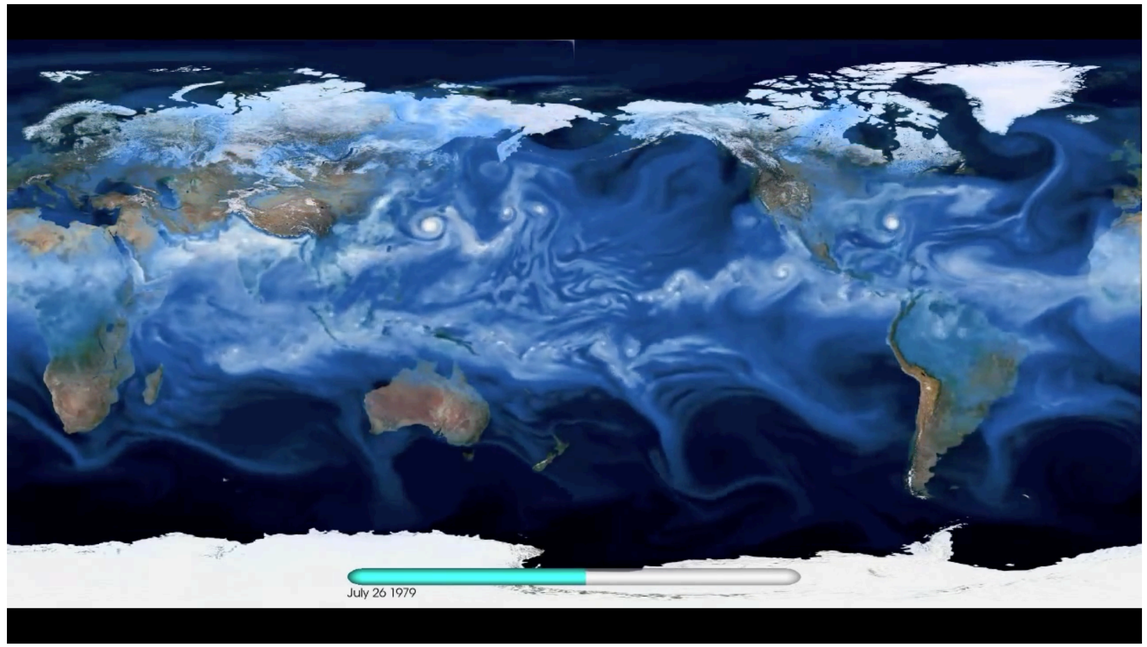
Typical Climate Analytics



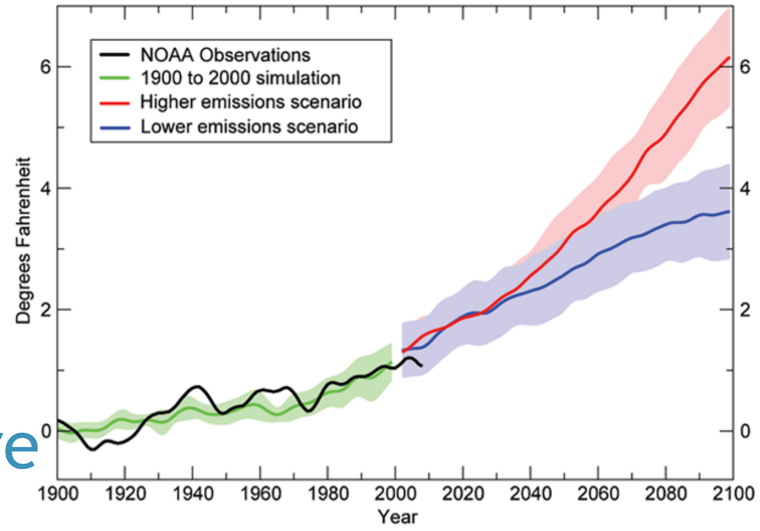
Typical Climate Analytics



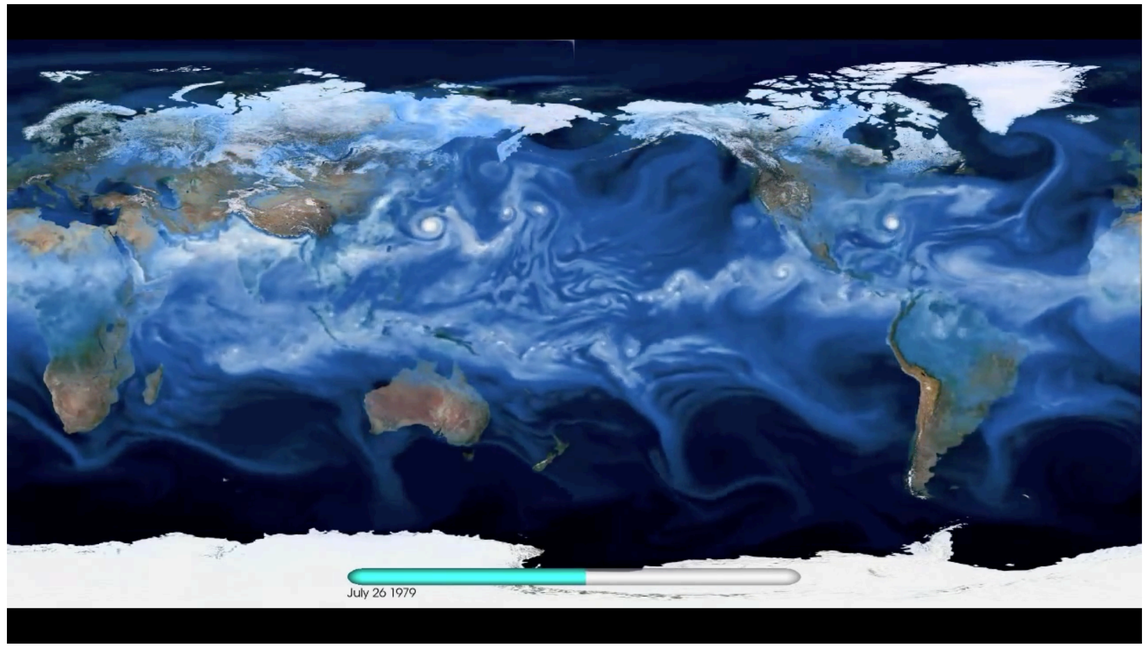
Typical Climate Analytics



mean/std Temperature

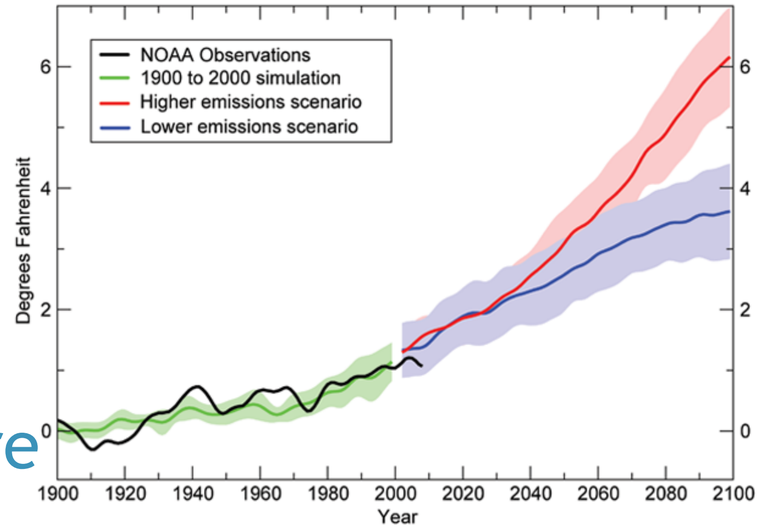


Typical Climate Analytics



14M variables/3h

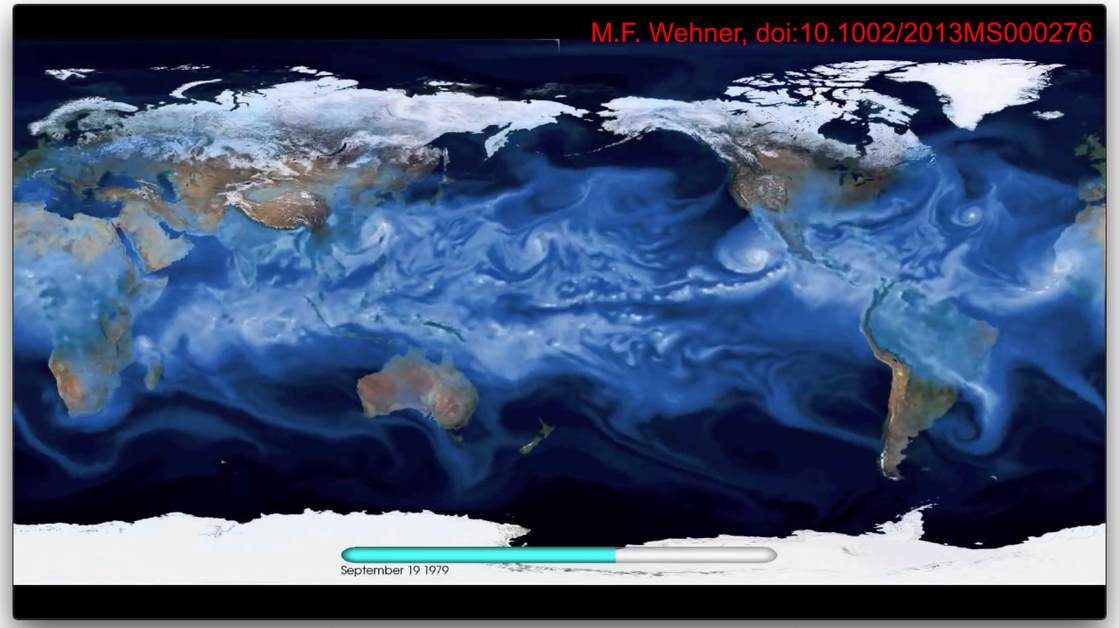
mean/std Temperature



O(1) variables/y

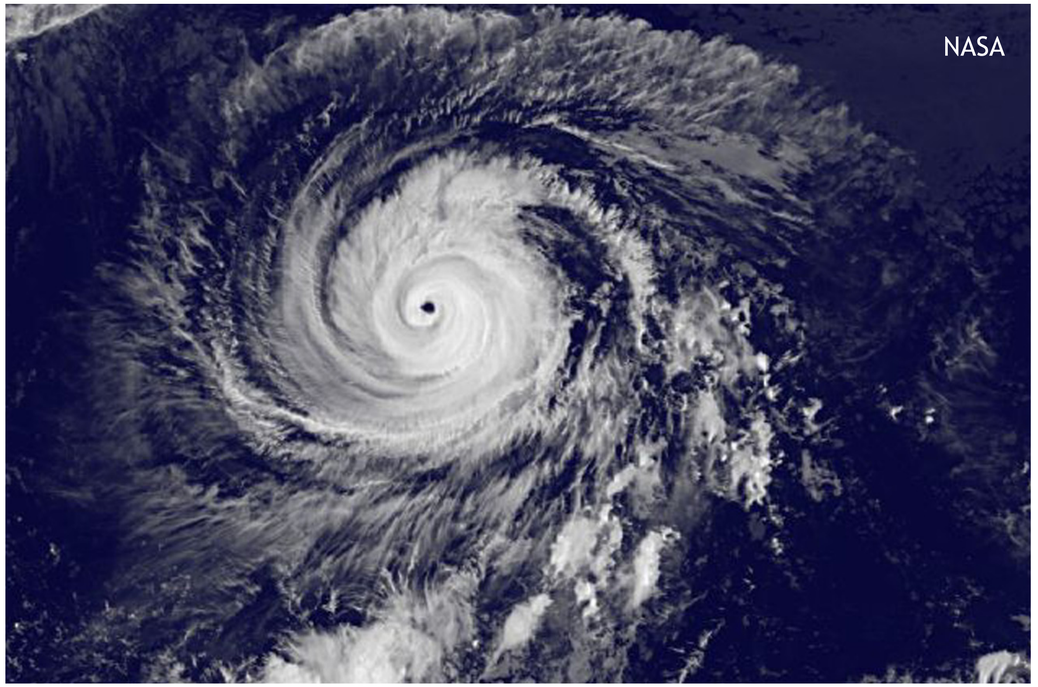
Impact Quantification of Extreme Weather Events

- detect hurricanes and atmospheric rivers in climate model projections
- enable geospatial analysis of EW events and statistical impact studies for regions around the world
- flexible and scalable detection algorithm
- gear up for future simulations with ~1 km^2 spatial resolution



Unique Challenges for Climate Analytics

- interpret as segmentation problem
  - 3 classes - background (BG), tropical cyclones (TC), atmospheric rivers (AR)
- deep learning has proven successful for these tasks
- climate data is complex
  - high imbalance - more than 95% of pixels are background
  - high variance - shape of events change
  - many input channels w/ different properties
  - high resolution required
  - no static *background*, highly variable in space and time



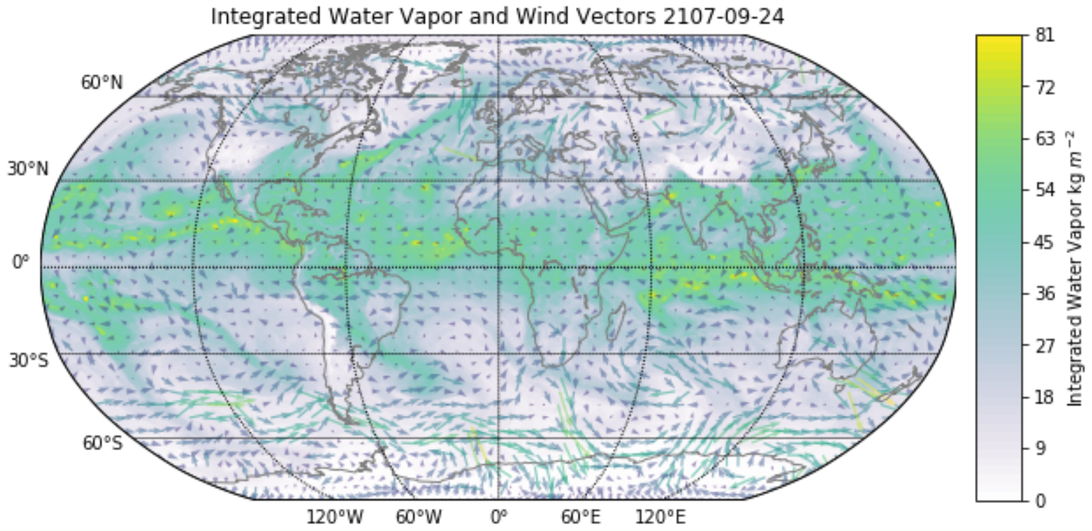
Unique Challenges for Deep Learning

- need labeled data for supervised approach
  - can be leveraged from existing heuristic-based approaches
- define neural network architecture
  - balance between compute performance and model accuracy
  - employ high productivity/flexibility frameworks for rapid prototyping
  - performance optimization requires holistic approach
- hyper parameter tuning (HPO)
  - necessary for convergence and accuracy

Unique Challenges for Deep Learning at Extreme Scale

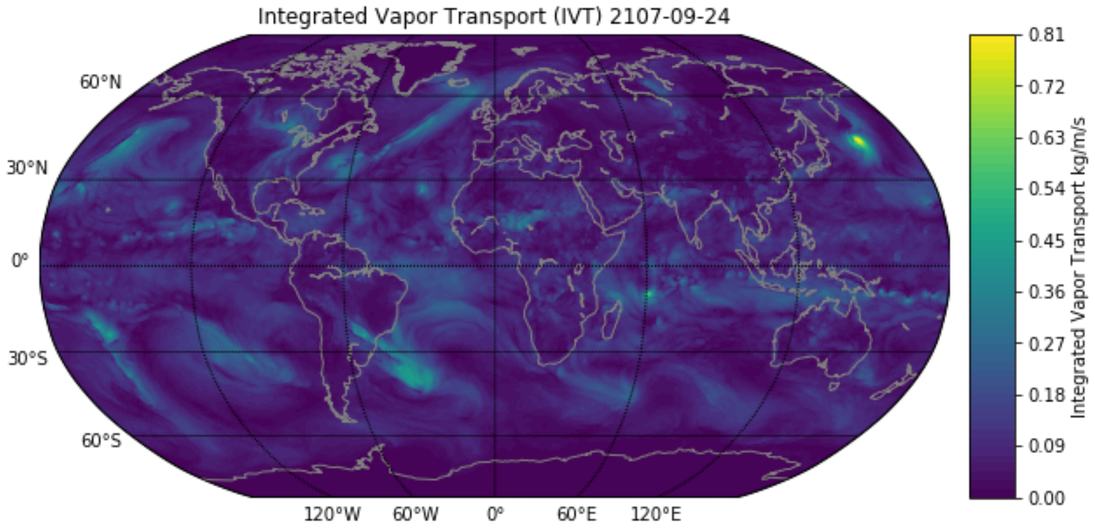
- data management
  - shuffling/loading/processing/feeding 20 TB dataset to keep GPUs busy
  - efficient use of remote filesystem
- multi-node coordination and synchronization
  - synchronous reduction of O(50)MB across 27360 GPUs after each iteration
- hyper parameter tuning (HPO)
  - convergence and accuracy challenging due to larger global batch sizes

Label Creation: Atmospheric Rivers

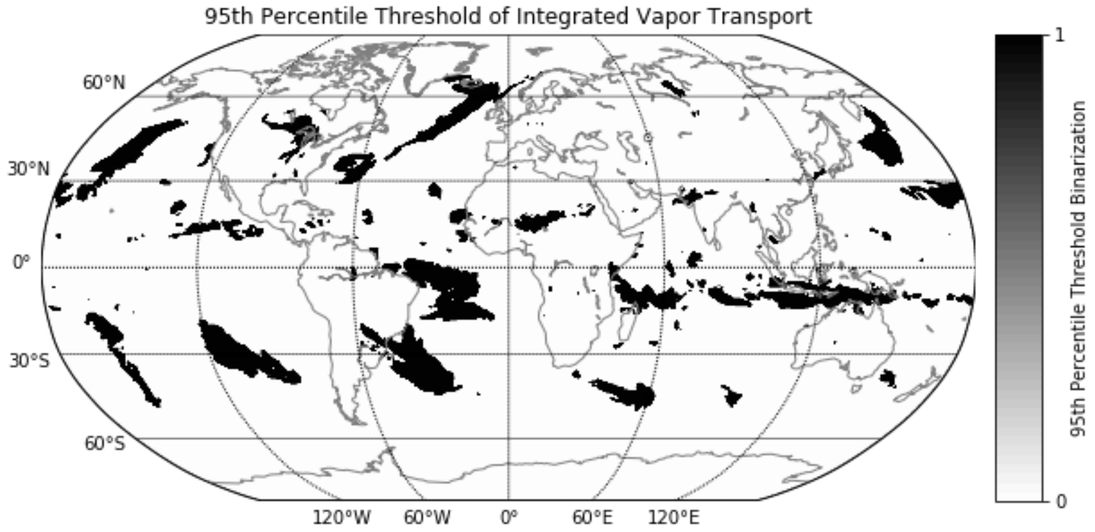


1. The climate model predicts *water vapor*, *wind speeds* and *humidity*

2. These observables are used to compute the *Integrated Water Vapor Transport*

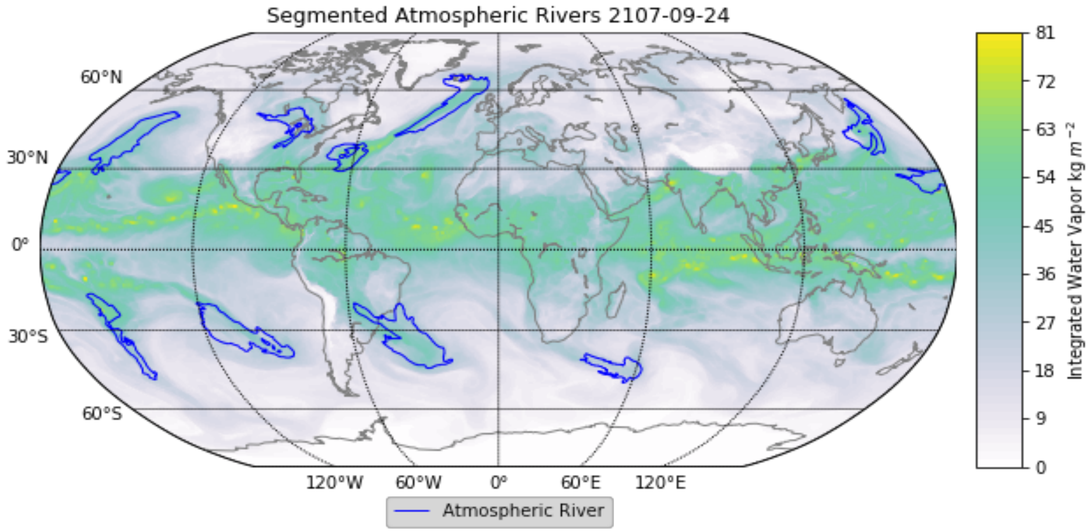


Label Creation: Atmospheric Rivers



3. Binarization by thresholding at 95th percentile

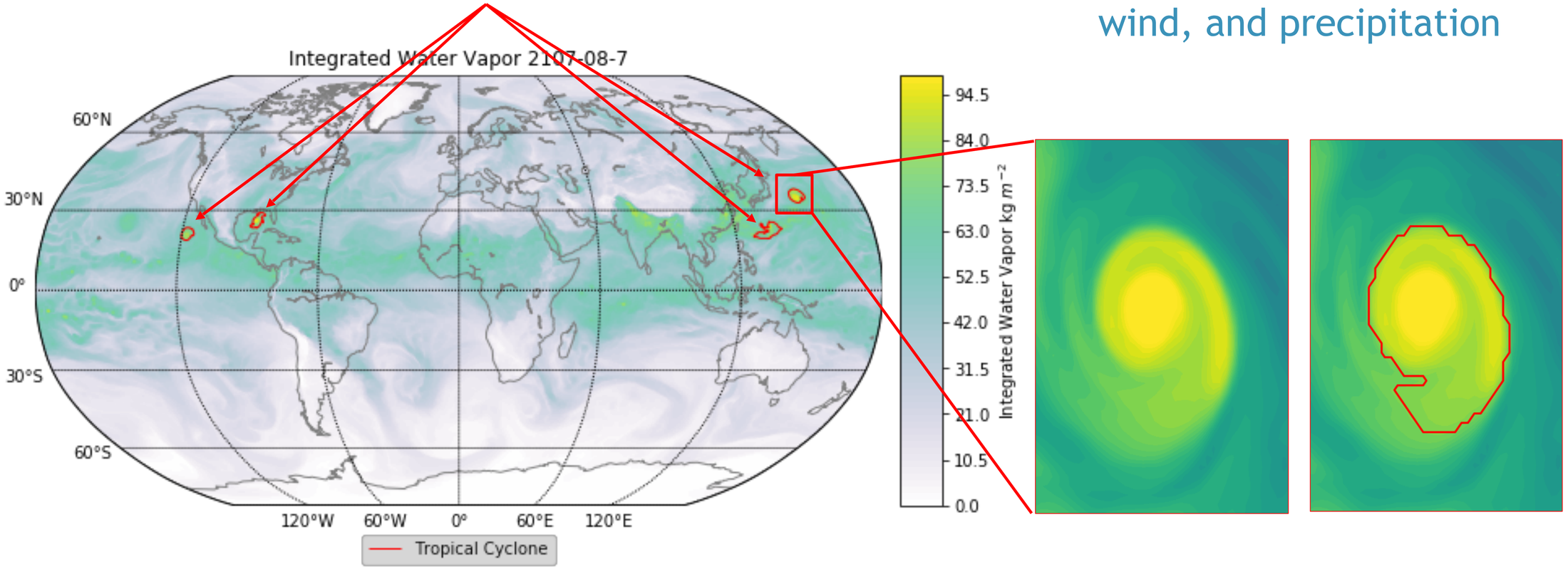
4. Flood fill algorithm generates AR candidates by masking out regions in mid-latitudes



Label Creation: Tropical Cyclones

1. Extract cyclone center and radius using thresholds for pressure, temperature, and vorticity

2. Binarize patch around cyclone center using thresholds for water vapor, wind, and precipitation



Software: TensorFlow

- high-productivity deep learning framework in Python with C++-backend, developed by Google
- leverages optimized cuDNN library for performance sensitive kernels (e.g. convolutions) on NVIDIA GPUs
- dataflow-style programming and asynchronous graph execution
- provides features for building I/O input pipeline
- can be combined with external Python modules to provide good flexibility

Software: Horovod

- library developed by Uber for distributed training
- provides straightforward wrappers to convert existing single process TensorFlow programs to multi-process data-parallel programs
- uses MPI for worker coordination/control
  - MPI is ubiquitous in HPC and available broadly on systems targeting HPC applications
- can leverage MPI or NCCL for collective communication on NVIDIA GPUs (e.g. allreduce)

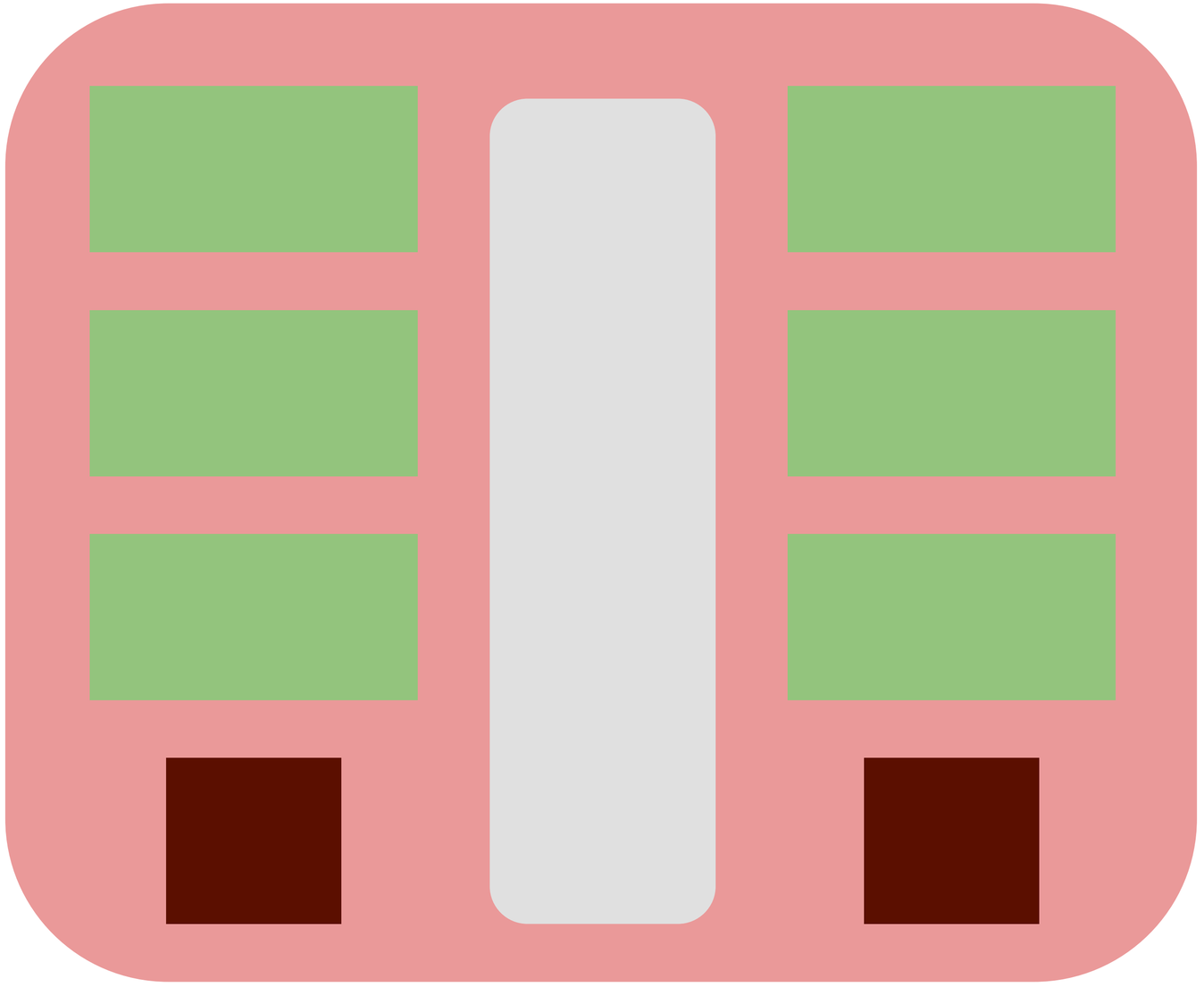
Systems

Piz Daint

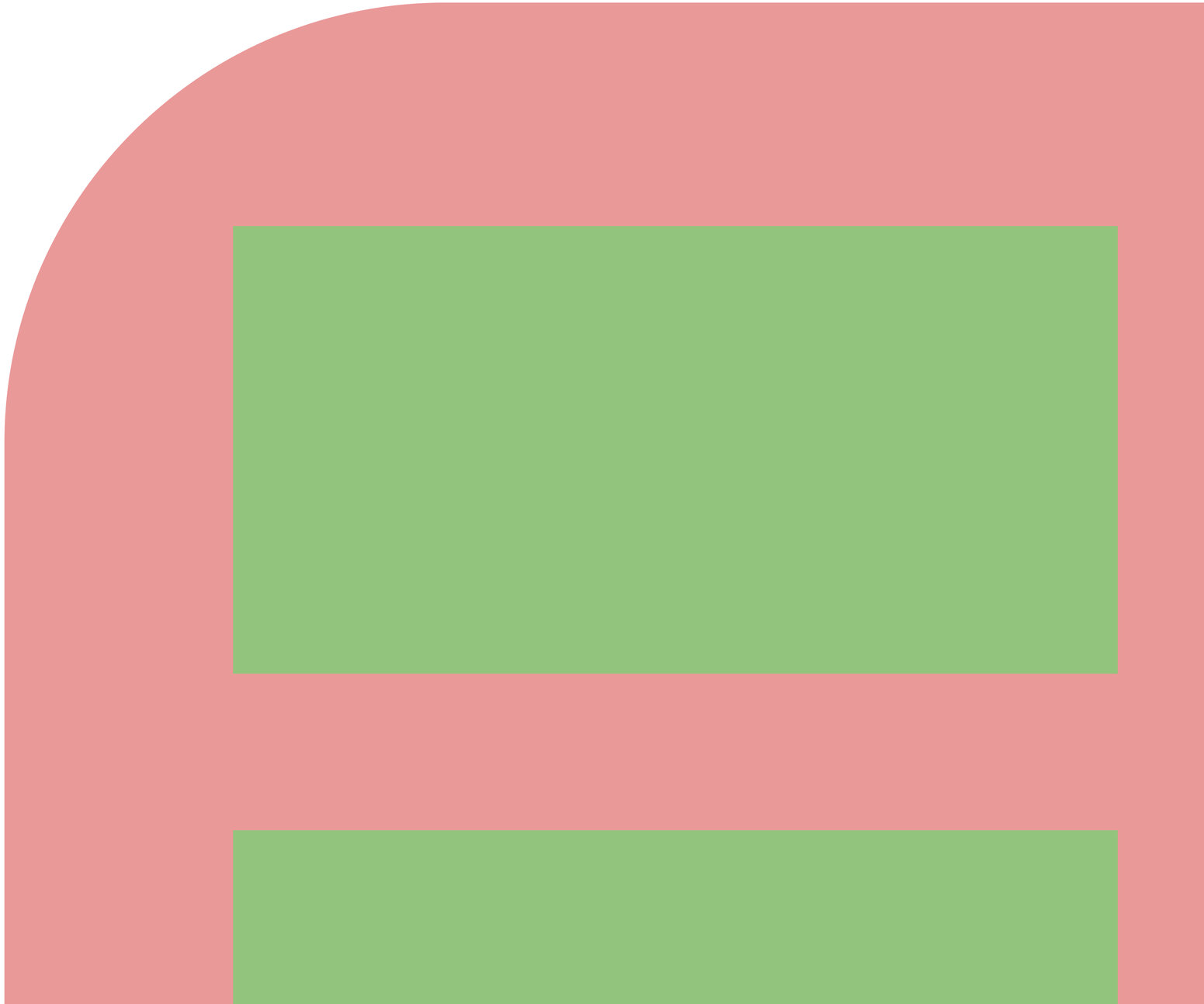
- Cray XC50 HPC system at CSCS, 5th on top500
- 5320 nodes with Intel Xeon E5-2695v3 and 1 NVIDIA P100 GPU
- Cray Aries interconnect in diameter 5 dragonfly topology
- ~54.4 PetaFlop/s peak performance (FP32)

Summit

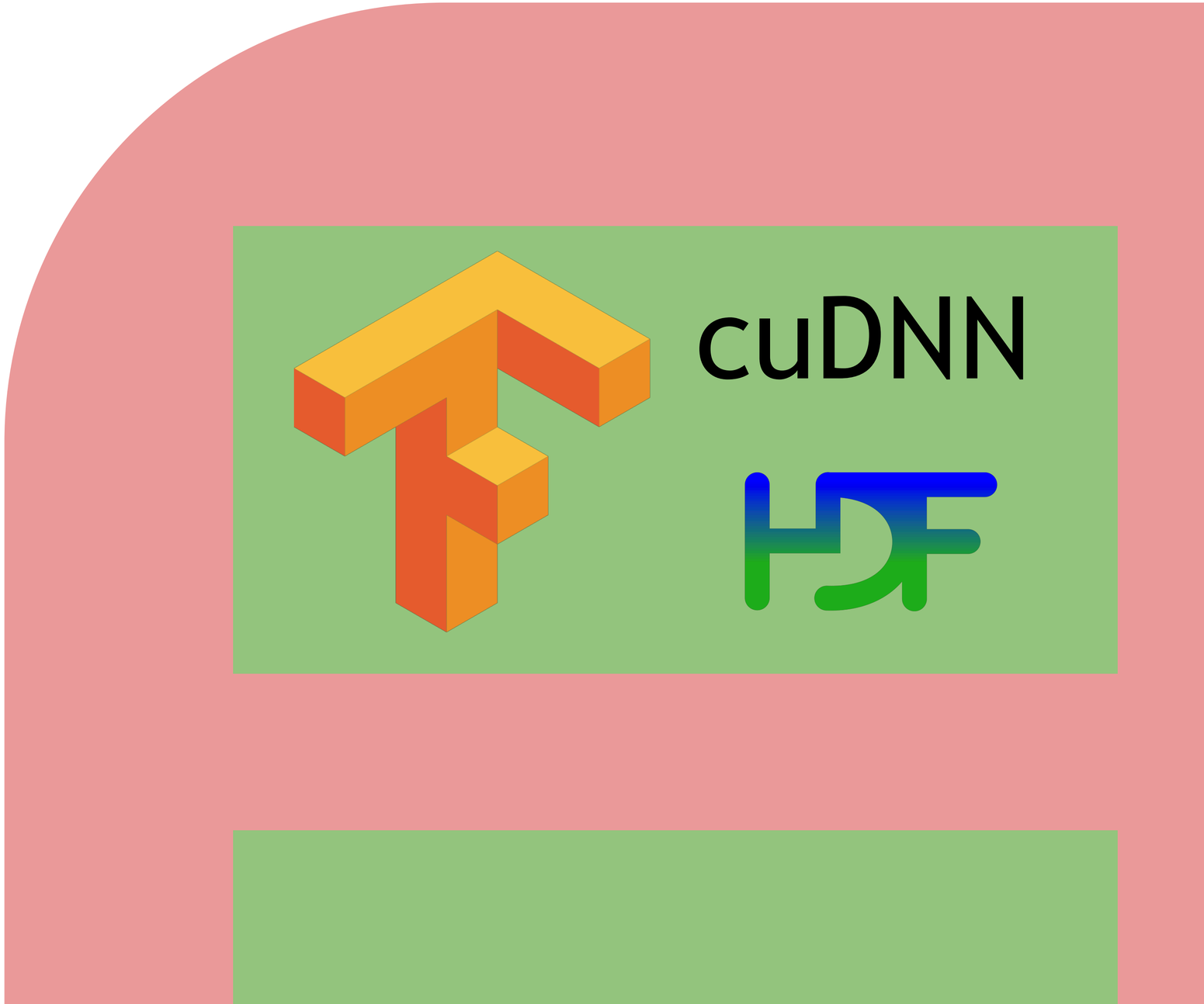
- leadership class HPC system at OLCF, 1st on top500
- 4609 nodes with 2 IBM P9 CPU and 6 NVIDIA V100 GPU
- 300 GB/s NVLink connection btw. 3 GPUs in a group
- 800 GB available NVMe storage/node
- dual-rail EDR Infiniband in fat-tree topology
- ~3.45 ExaFlop/s theoretical peak performance (FP16)



Single GPU

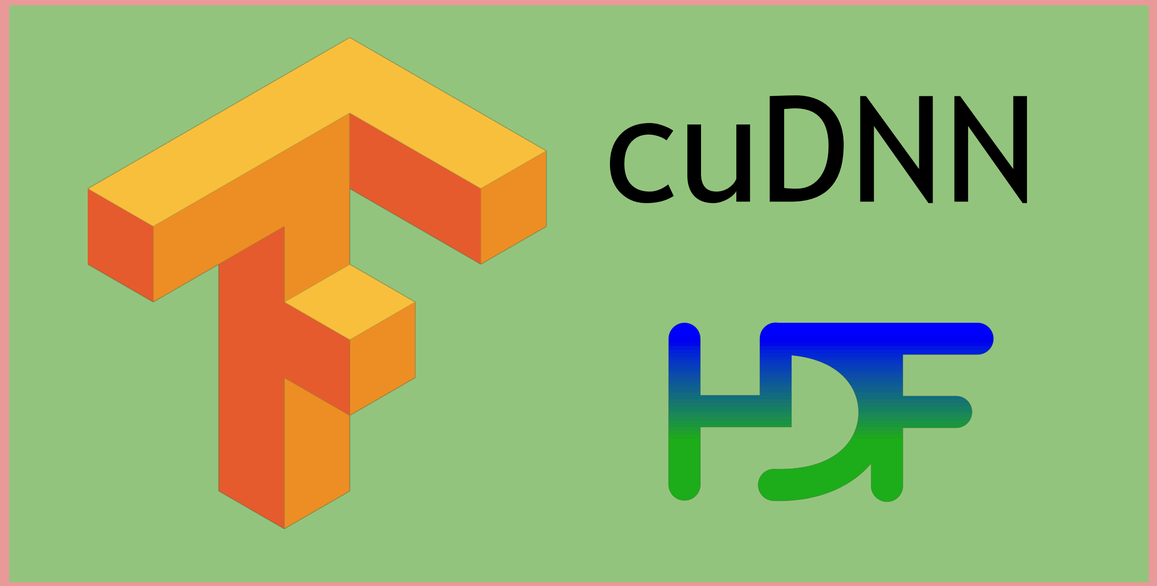


Single GPU

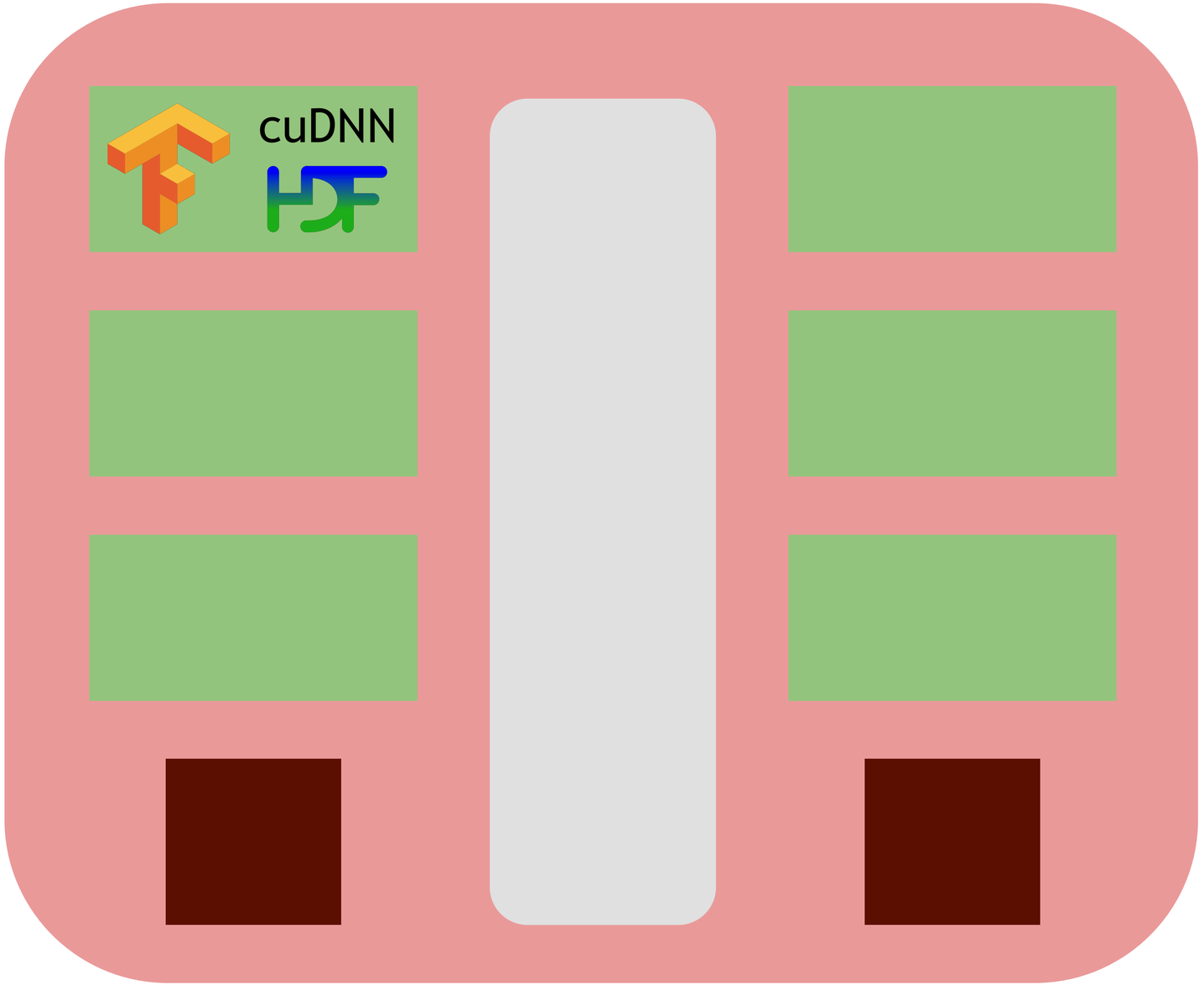


Single GPU

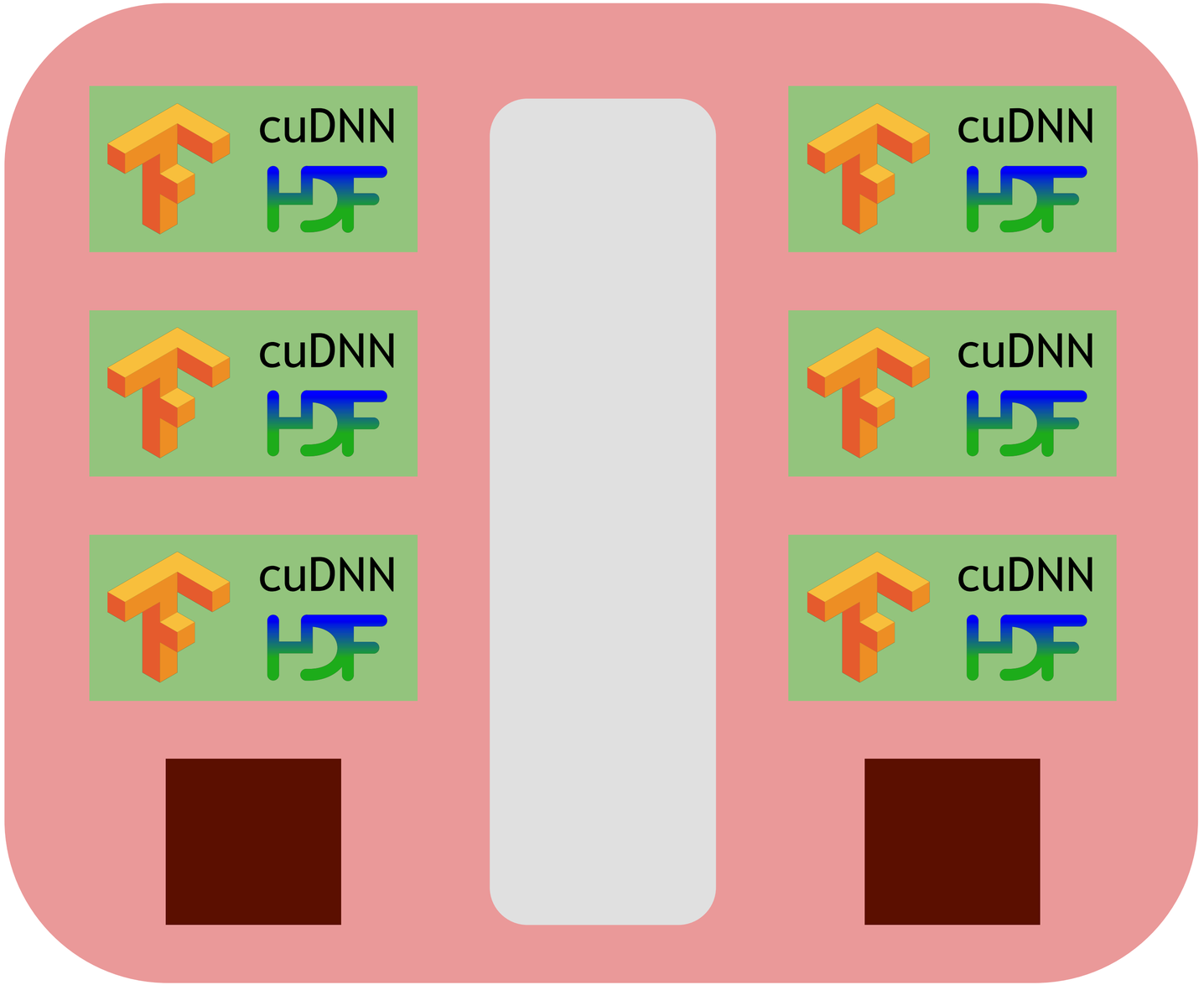
- Things to consider:
  - Is my TensorFlow model efficiently using GPU resources?
  - Is my data input pipeline keeping up?
  - Is my TensorFlow model providing reasonable results?



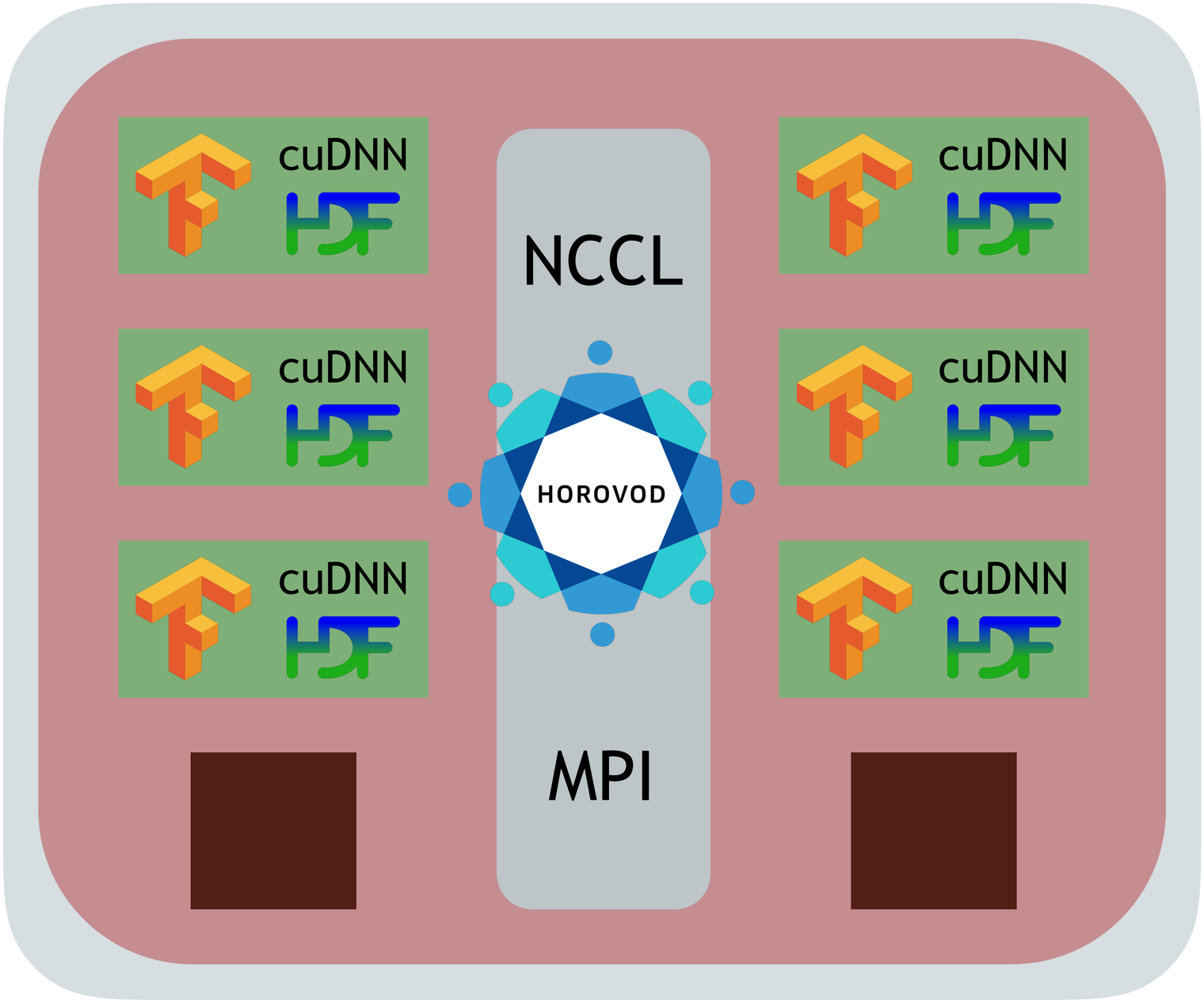
Single Node



Single Node

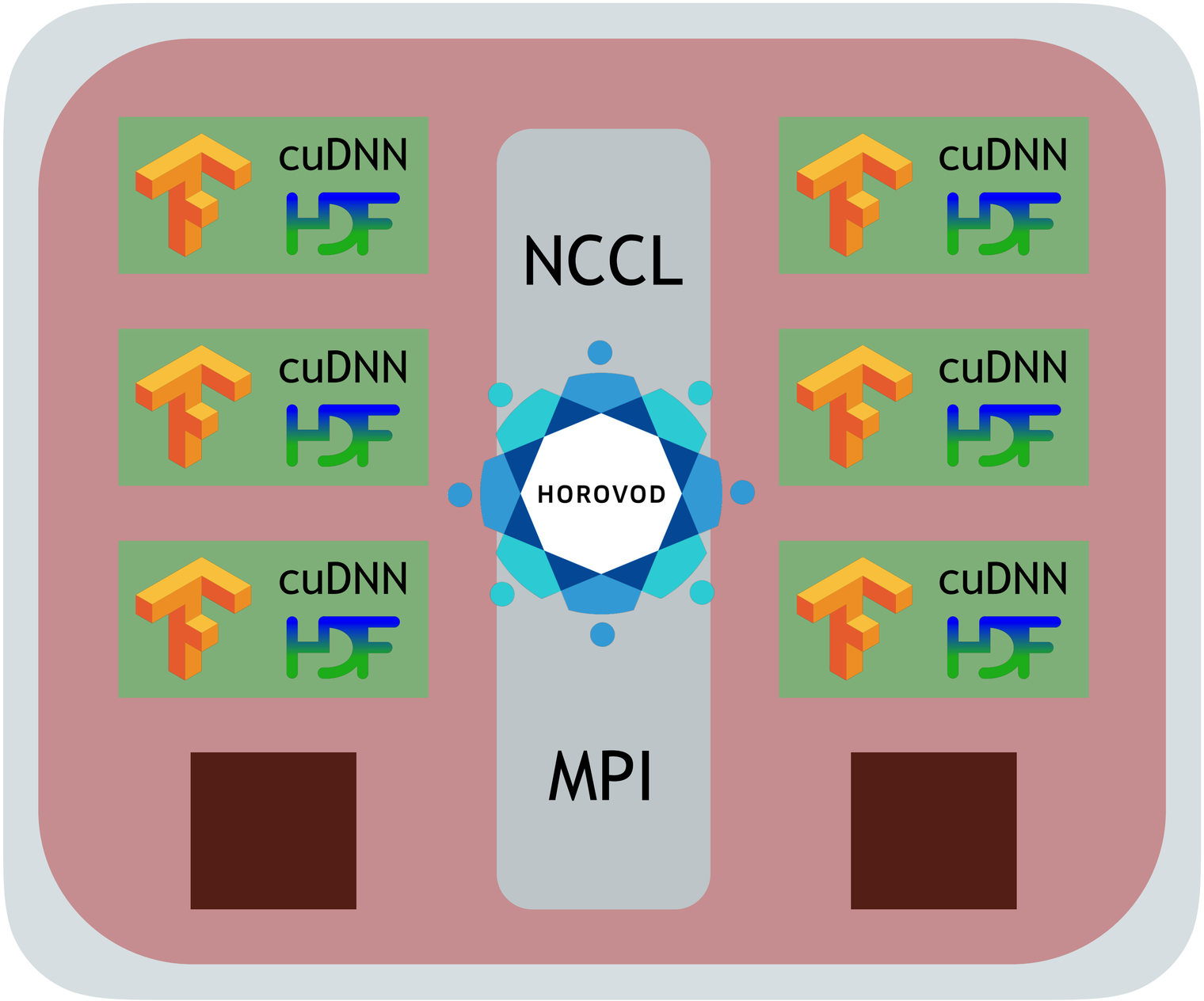


Single Node

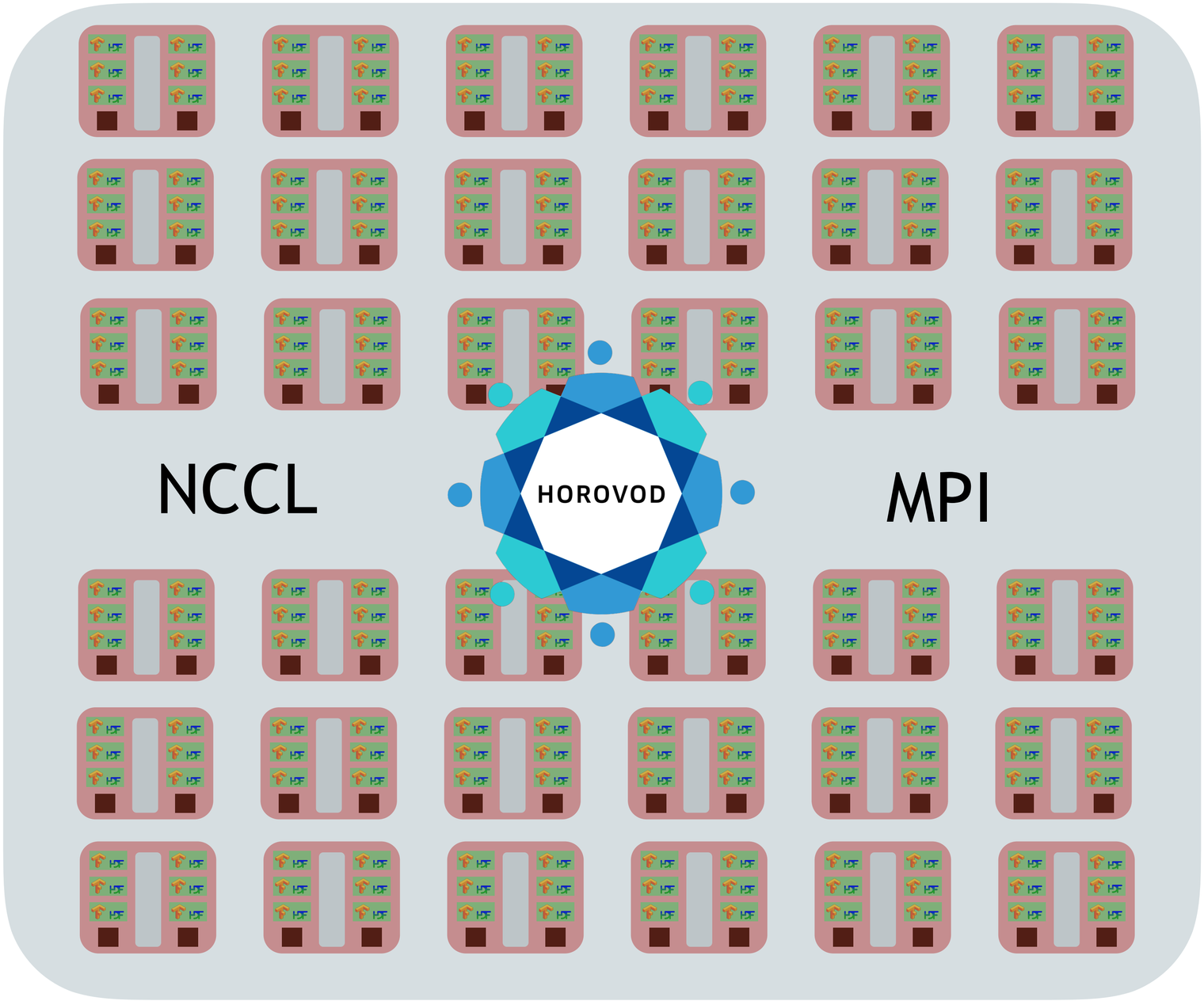


Single Node

- Things to consider:
  - Is my data input pipeline *still* keeping up?
  - Is my *data-parallel* TensorFlow model providing reasonable results?
  - How is my performance scaling over PCIe/NVLink using Horovod?

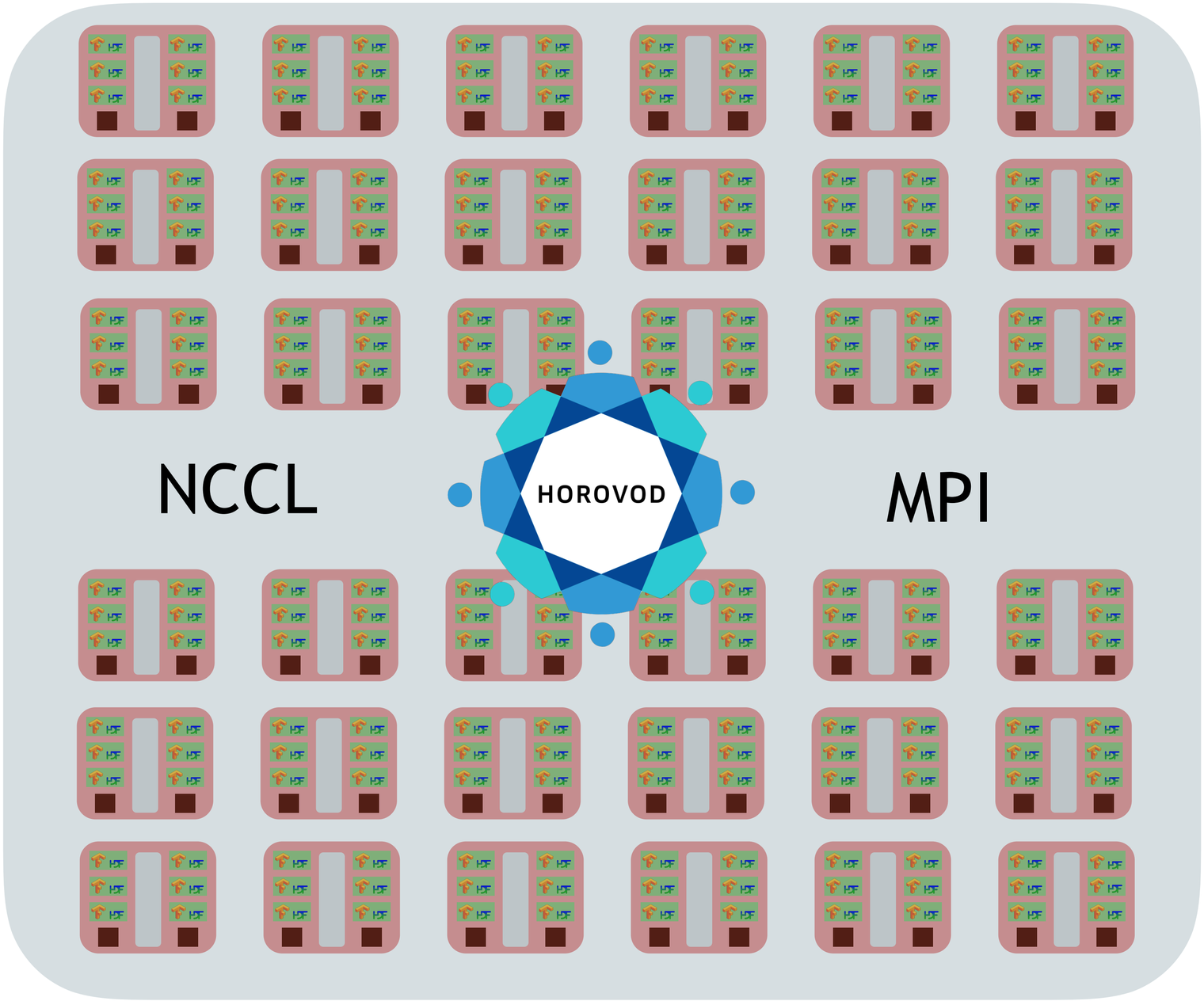


Multi Node

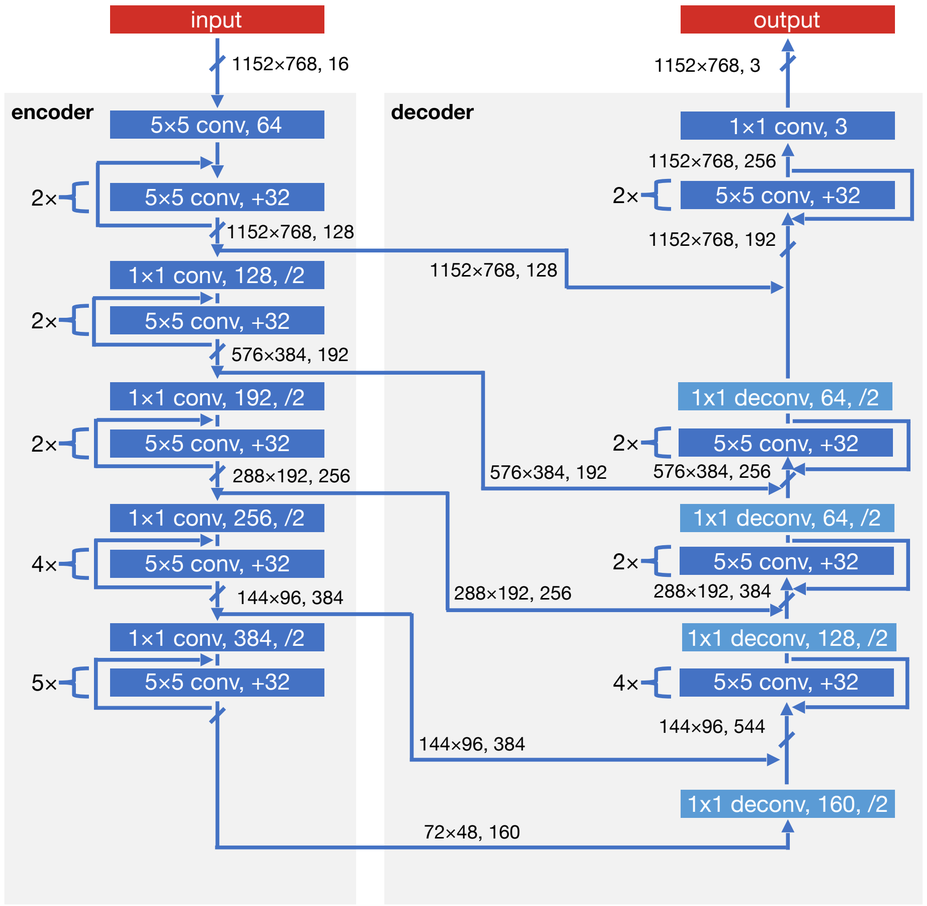


Multi Node

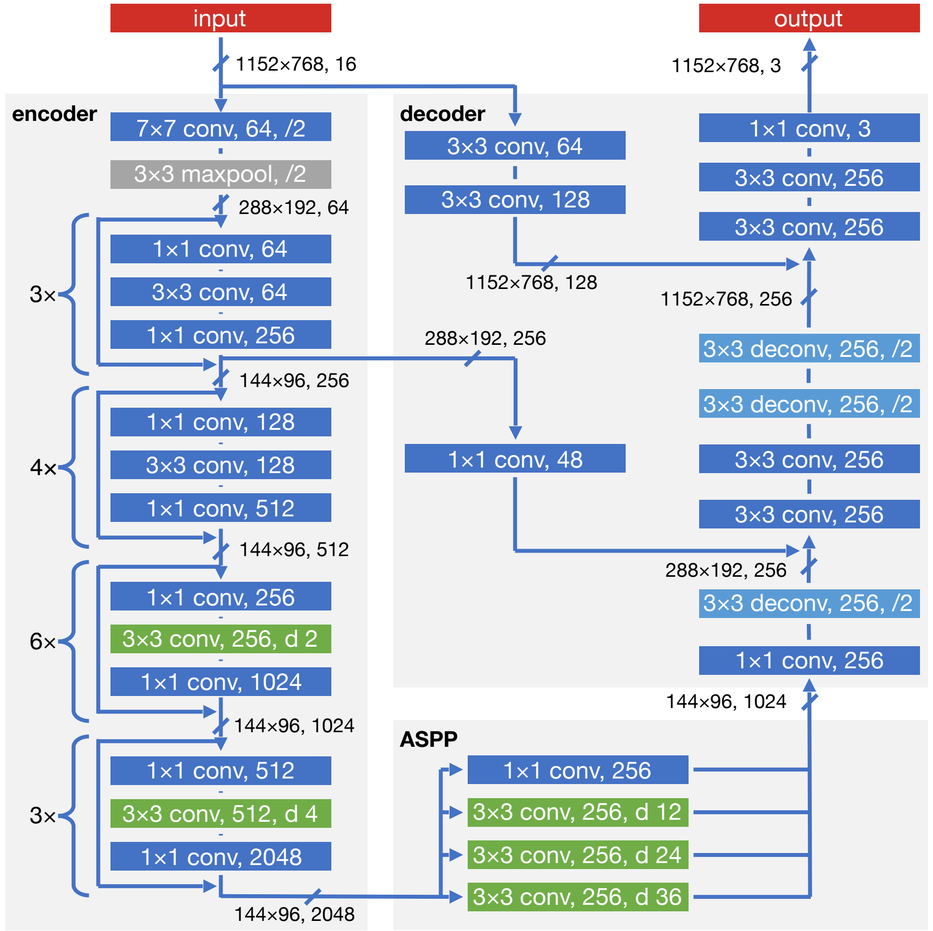
- Things to consider:
  - Is my data-parallel TensorFlow model *still* providing reasonable results?
  - How is my performance scaling over NVLink + *InfiniBand* using Horovod?
  - How do I distribute my data across so many nodes?



Deep Learning Models for Extreme Weather Segmentation



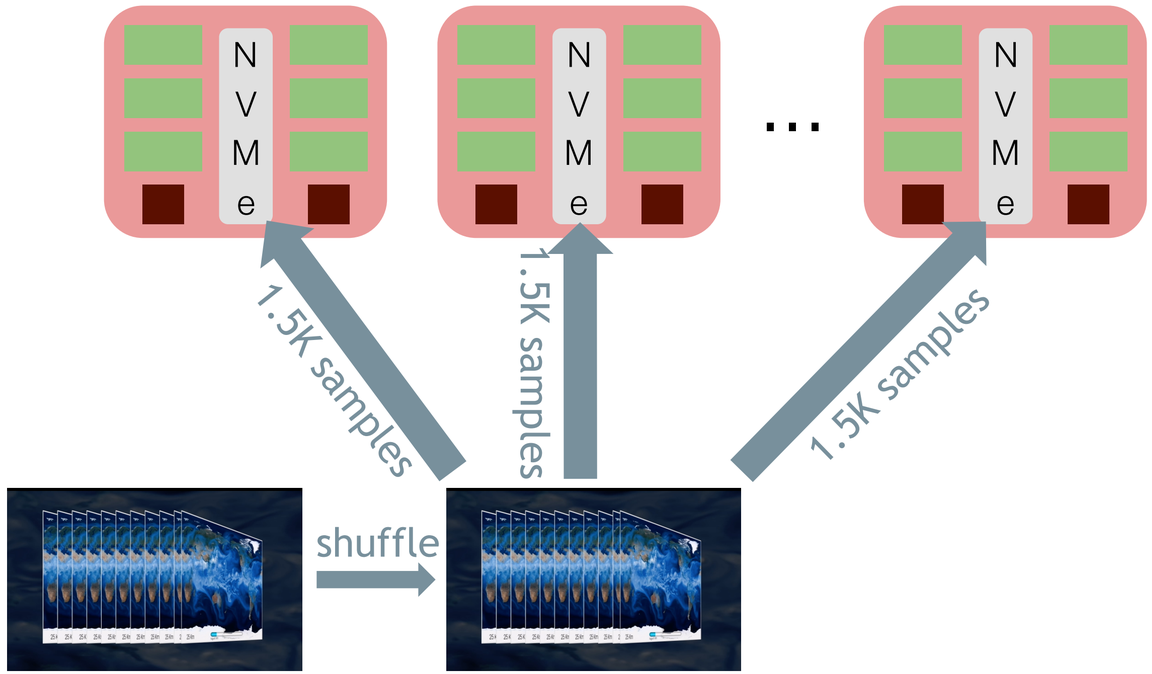
Tiramisu, 35 layers,
7.8M parameters, 4.2 TF/sample



DeepLabv3+, 66 layers,
43.7M parameters, 14.4 TF/sample

Data Staging

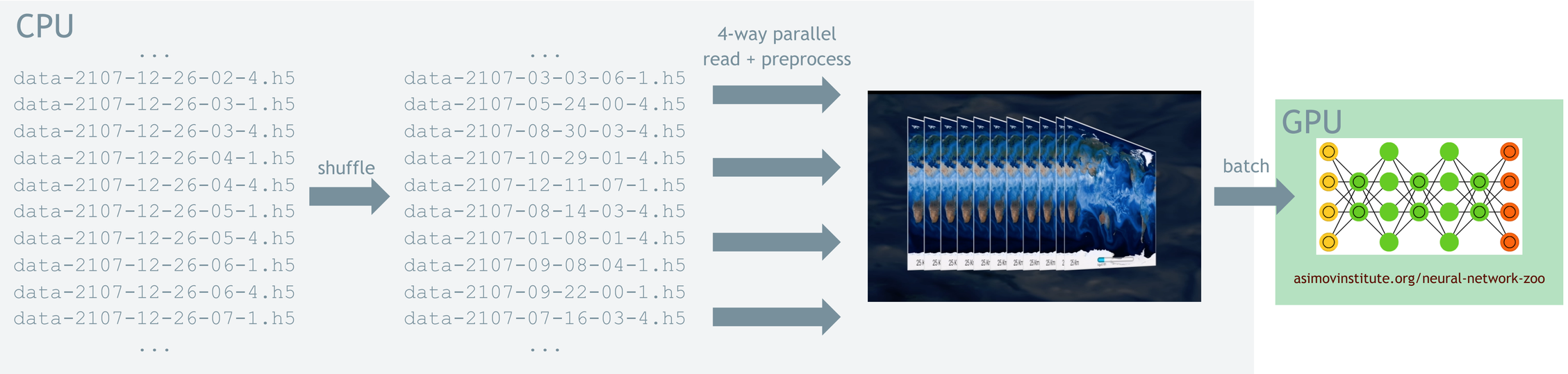
Dataset Size	Required BW (27K GPUs)	GPFS/LUSTRE	BurstBuffer	NVM/e or DRAM
20 TB (~63K samples)	3.8 TB/s	~400 GB/s	~2 TB/s	~26 TB/s



- 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`)
- preprocess and feed data to GPU asynchronously using `tf.data` and python `multiprocess`

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 + h5py`
- extract and batch using `tf.data` input pipeline

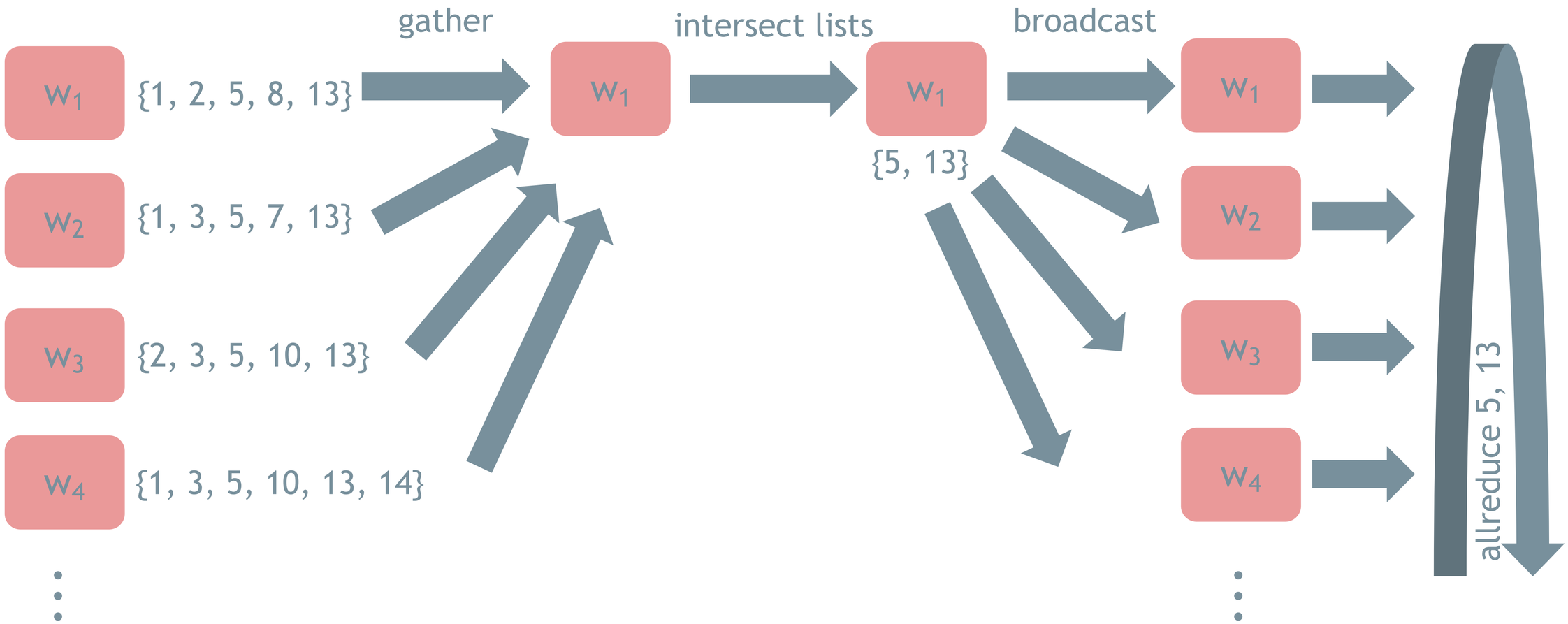


Single Node Performance

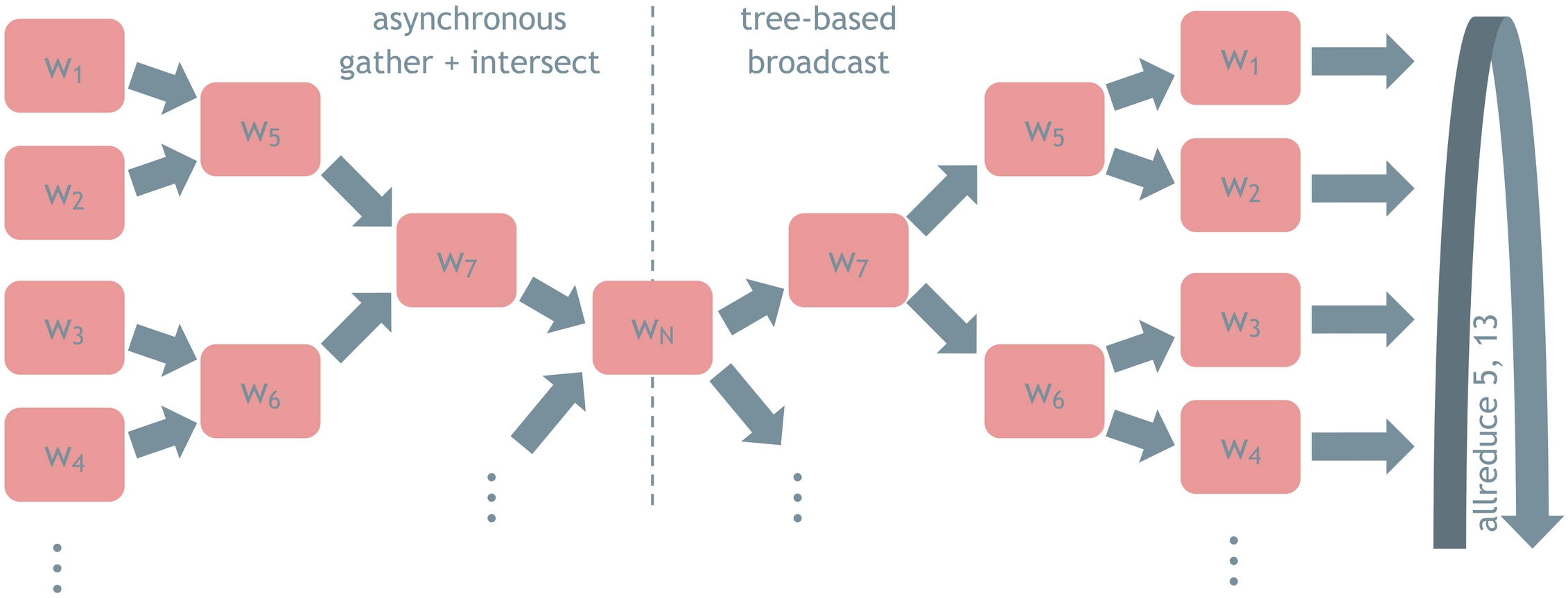
- GPU execution profiled with CUDA profiler, kernels grouped by category
- convolution kernels: use latest cuDNN, favor higher computational intensity
- pay attention to memory layout to reduce transposes and copies
- tuning input pipeline on CPU to keep off critical path

Category		DeepLabv3+ FP16 Training						
		# Kern	Time (ms)	Math (TF)	Mem (GB)	% Time	% Math	% Mem
Forward	Convolutions	158	147.9	9.61	27.6	18.1	52.0	20.7
	Point-wise	829	52.3	< 0.1	24.3	6.4		51.6
Backward	Convolutions	195	300.2	19.21	50.5	36.7	51.2	18.7
	Point-wise	157	25.6	< 0.1	6.3	3.1		27.3
Optimizer		1219	3.9	< 0.1	1.1	0.5		31.3
Copies / Transposes		708	213.2	-	92.6	26.1		48.3
Allreduce (NCCL)		30	58.7	< 0.1	0.6	7.2		1.1
Type Conversions		201	1.3	-	0.6	0.2		51.3
GPU Idle			14.2			1.7		
Total		3497	817.3	28.82	203.6		28.2	27.7

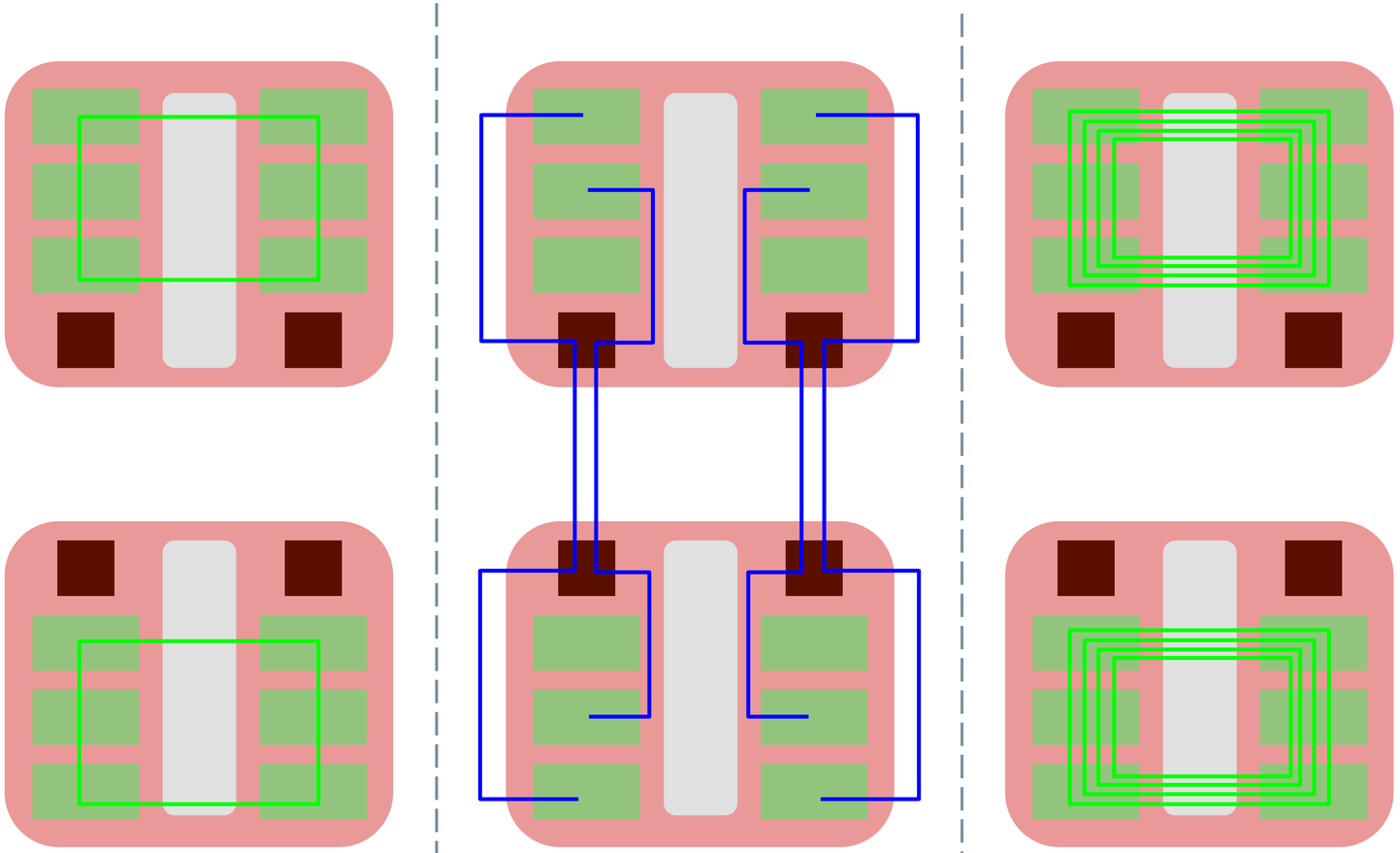
Improvements to Horovod: Original Control Plane



Improvements to Horovod: Tree-based Control Plane



Improvements to Horovod: Hybrid All-Reduce



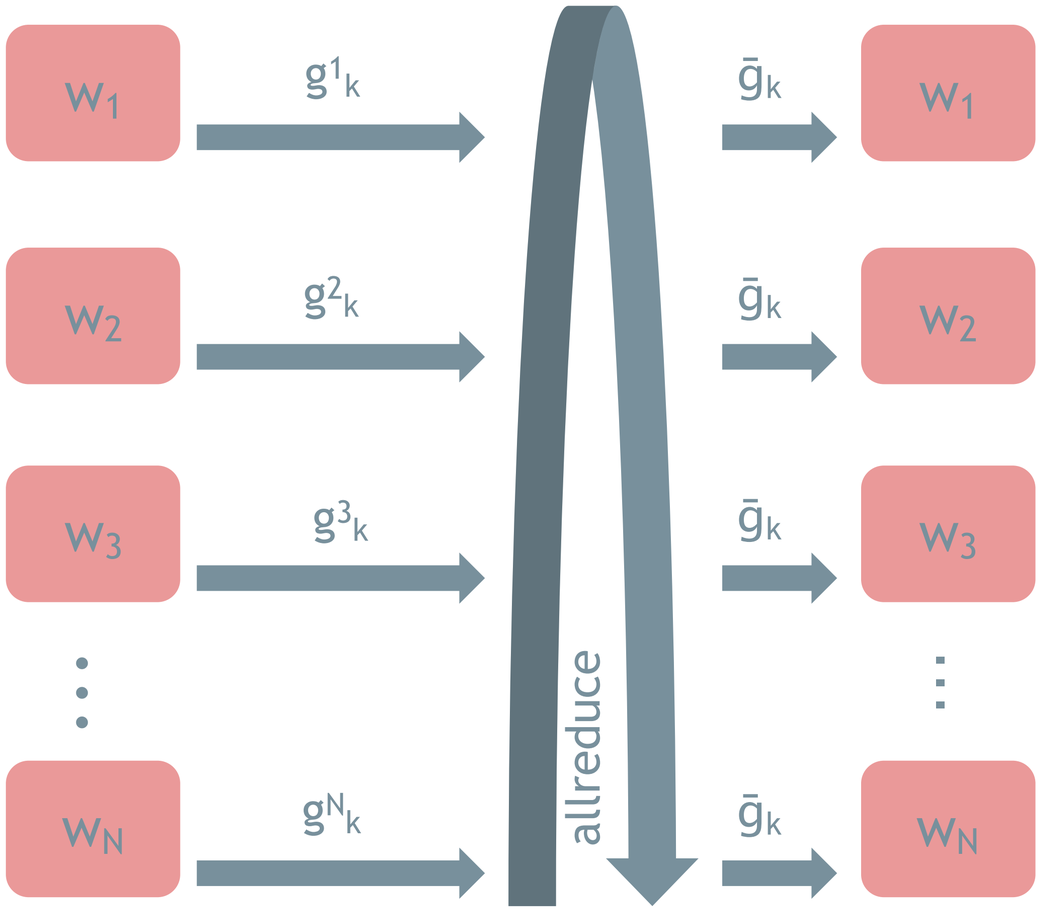
intra-node allreduce (NCCL)

4x inter-node allreduce (MPI)

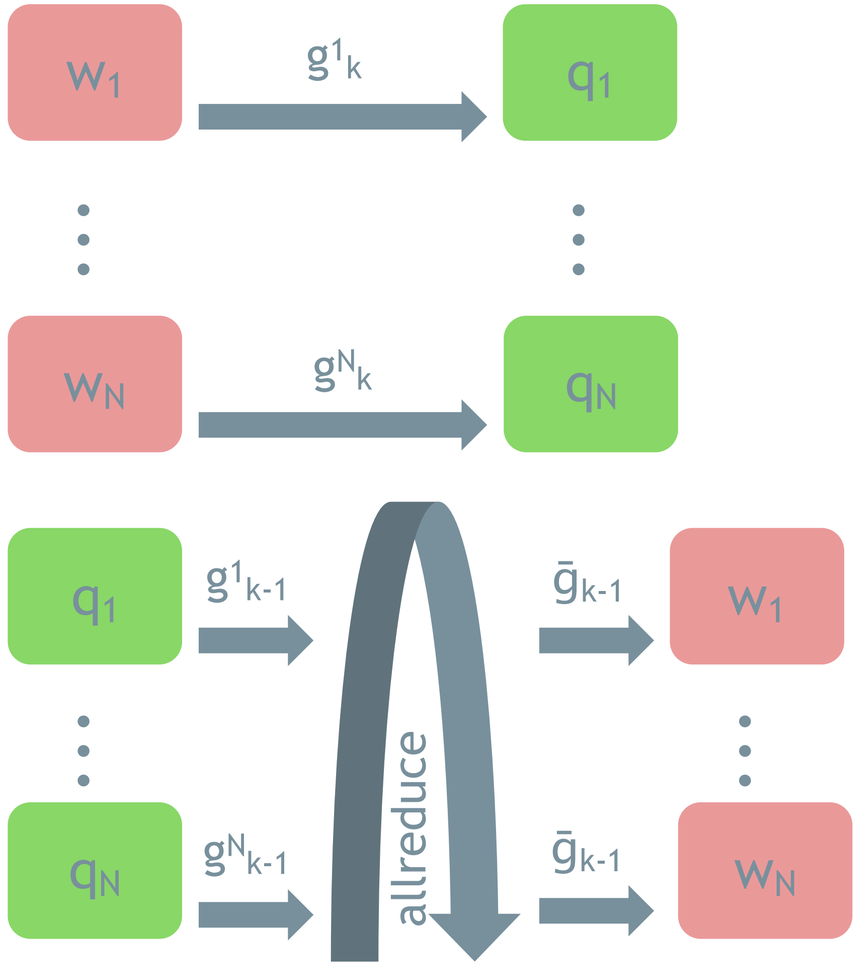
4x intra-node broadcast (NCCL)

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

Gradient Pipelining (Lag)

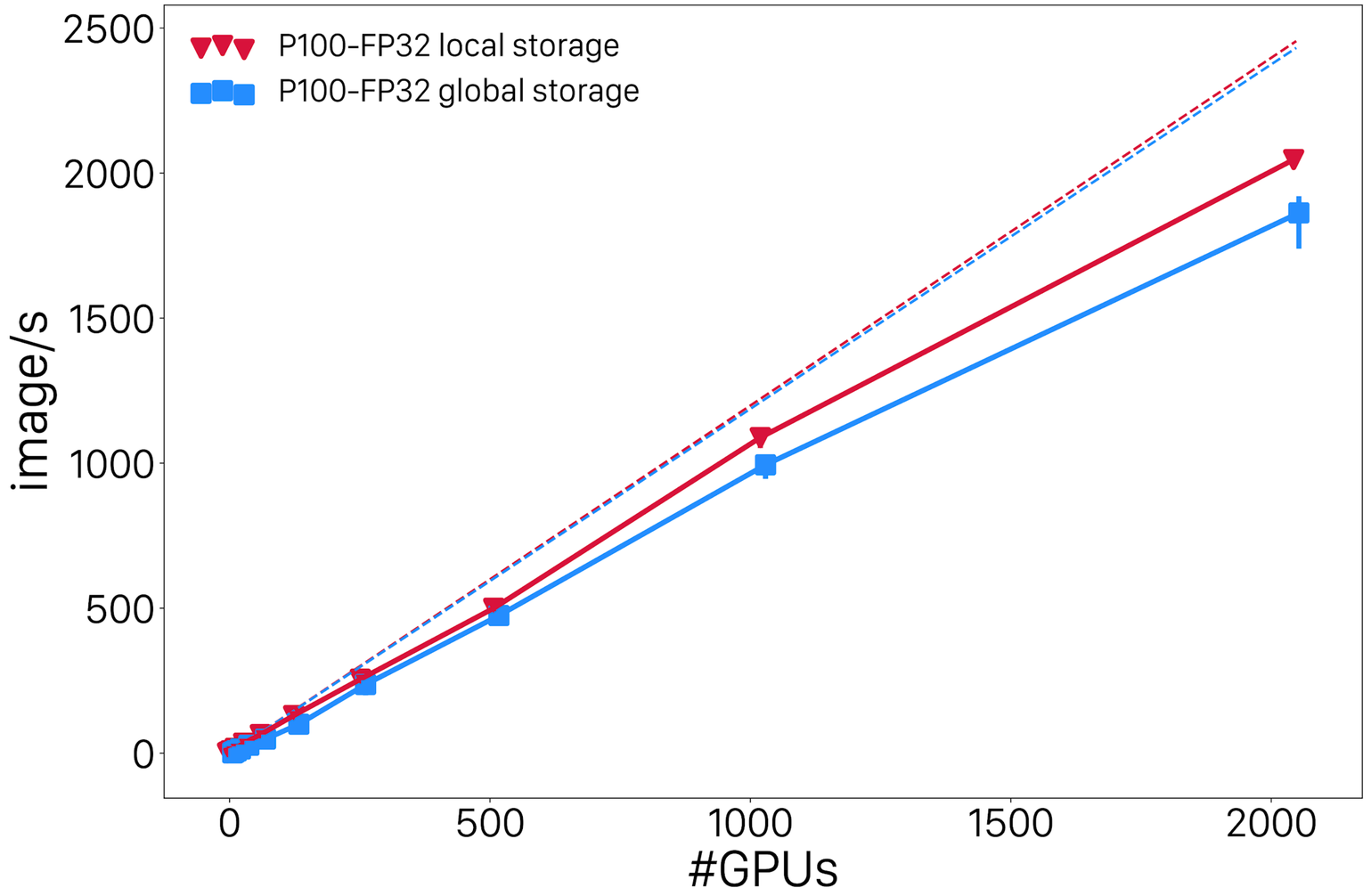


lag-0 (fully synchronous)



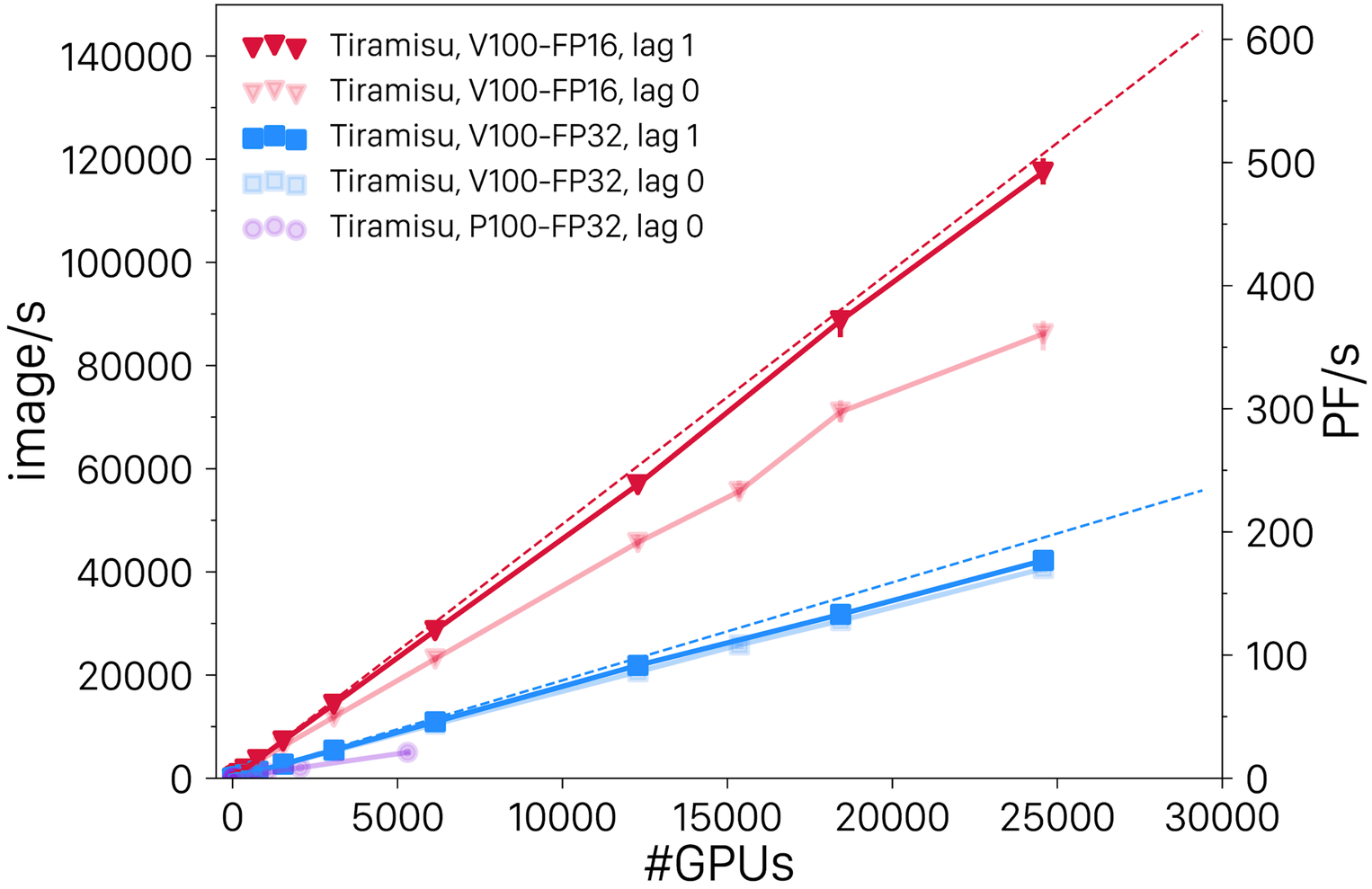
lag-1

Importance of Node-Local Memory



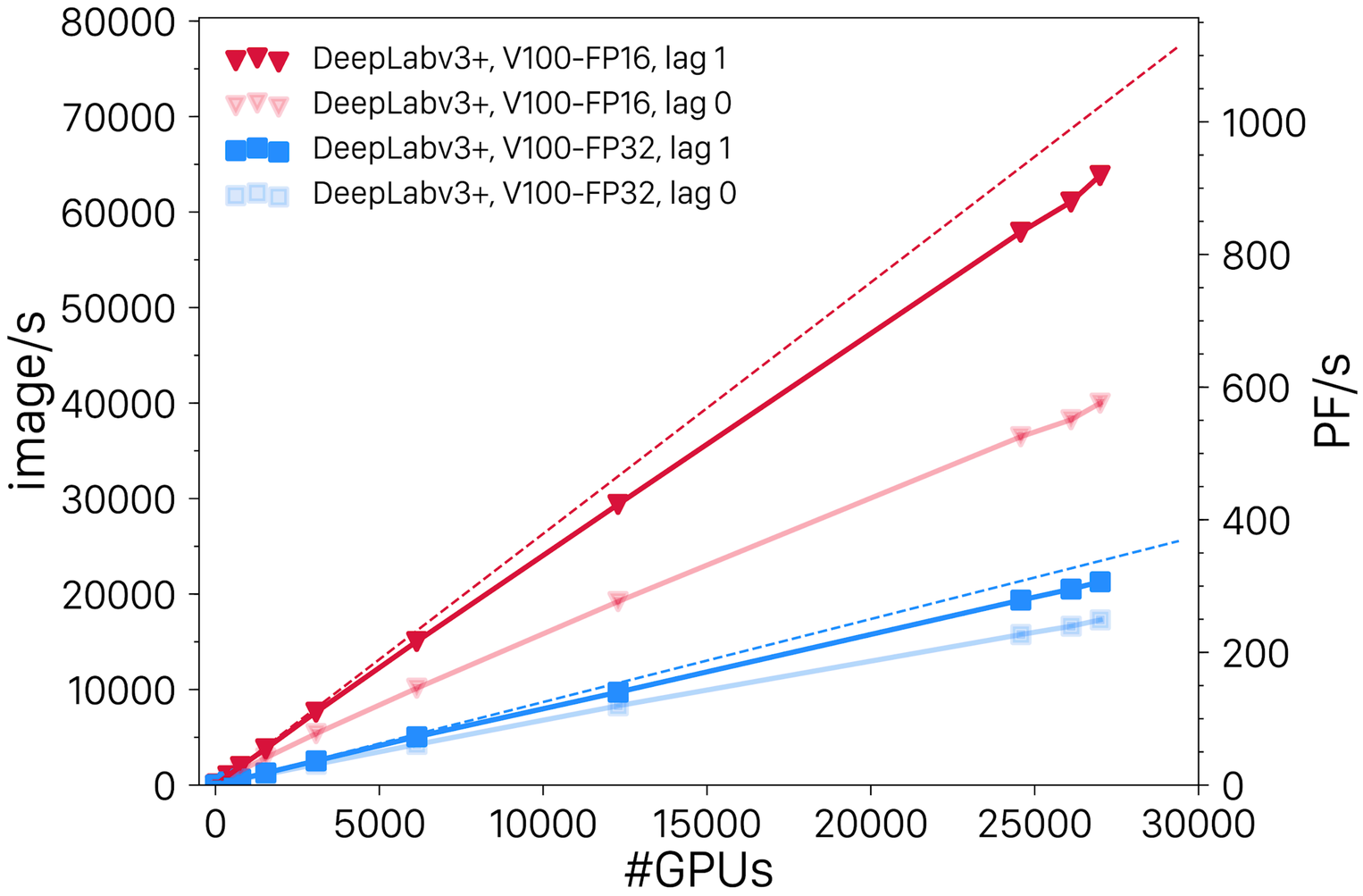
}
~10% performance penalty for reading from global file system

Scaling Tiramisu



- FP16-model sensitive to communication
- FP16-model BW-bound (only 2.5x faster than FP32)
- almost ideal scaling for both precisions on Summit when gradient lag is used

Scaling DeepLabv3+



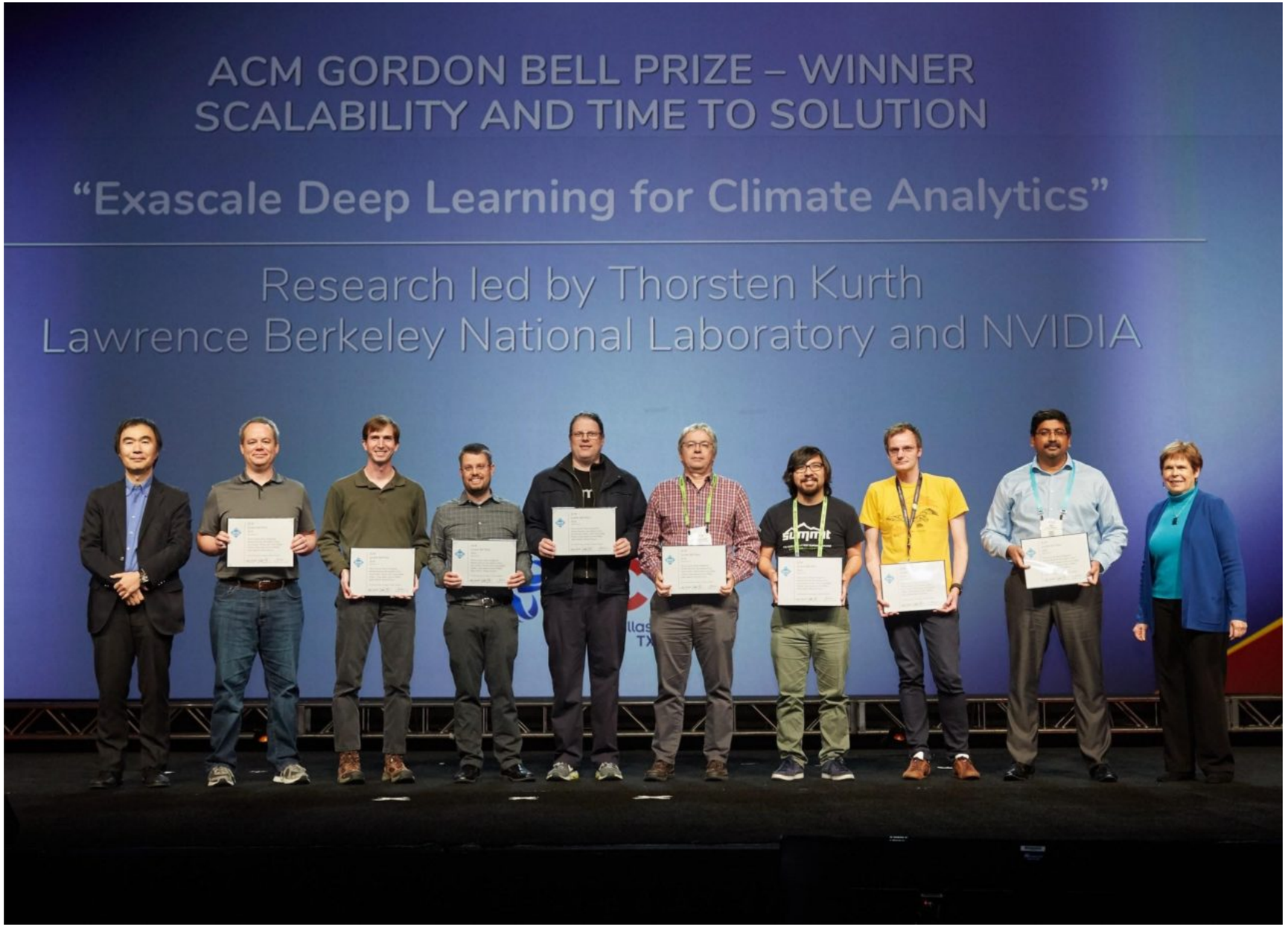
- FP16-model sensitive to communication
- FP16-model BW-bound (only 2.5x faster than FP32)
- excellent scaling for both precisions on Summit when gradient lag is used

999 PetaFlop/s (FP16) sustained

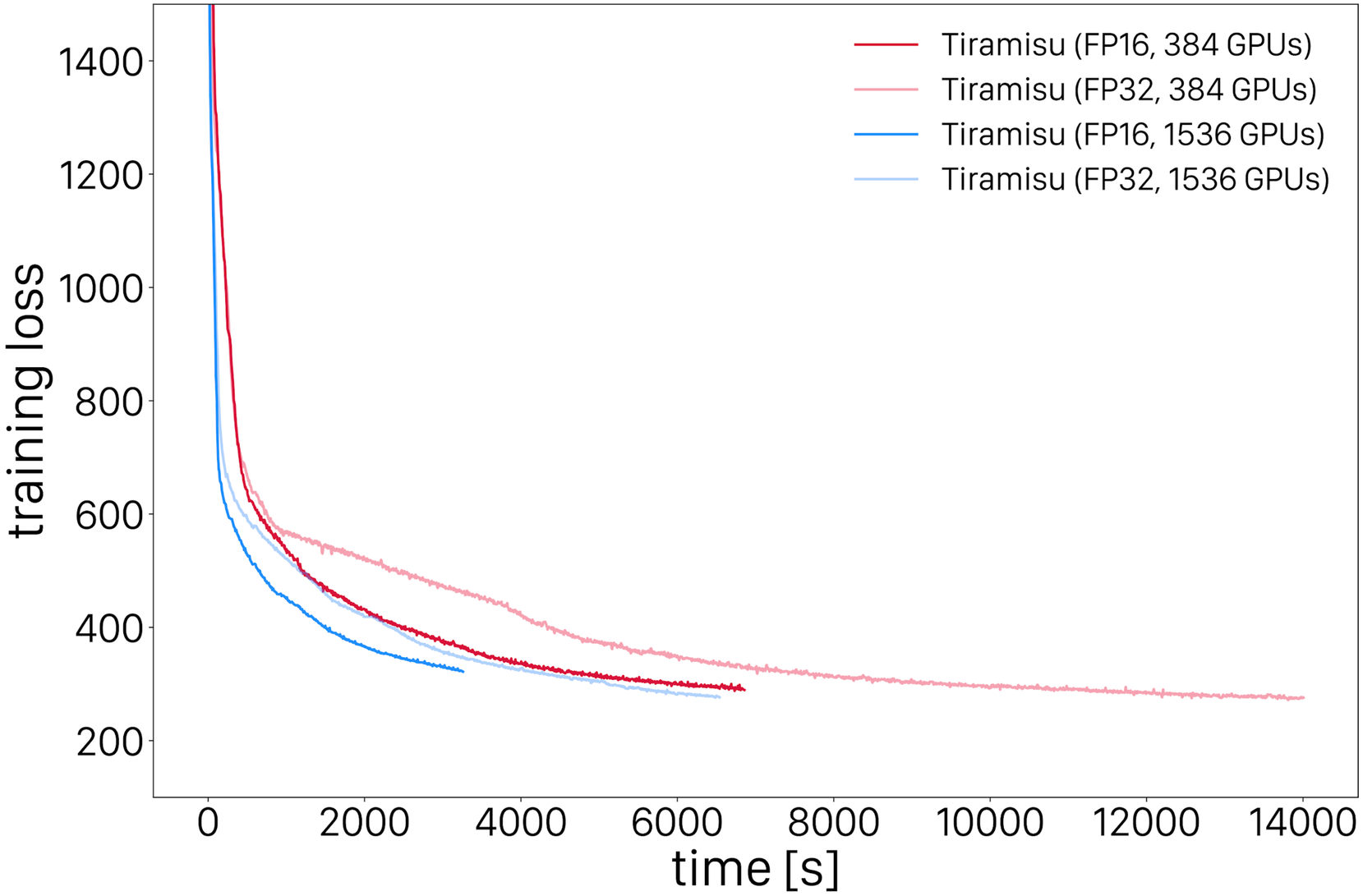
DeepLabv3+, 4560 nodes (27360 GPU)

1.13 ExaFlop/s (FP16) peak

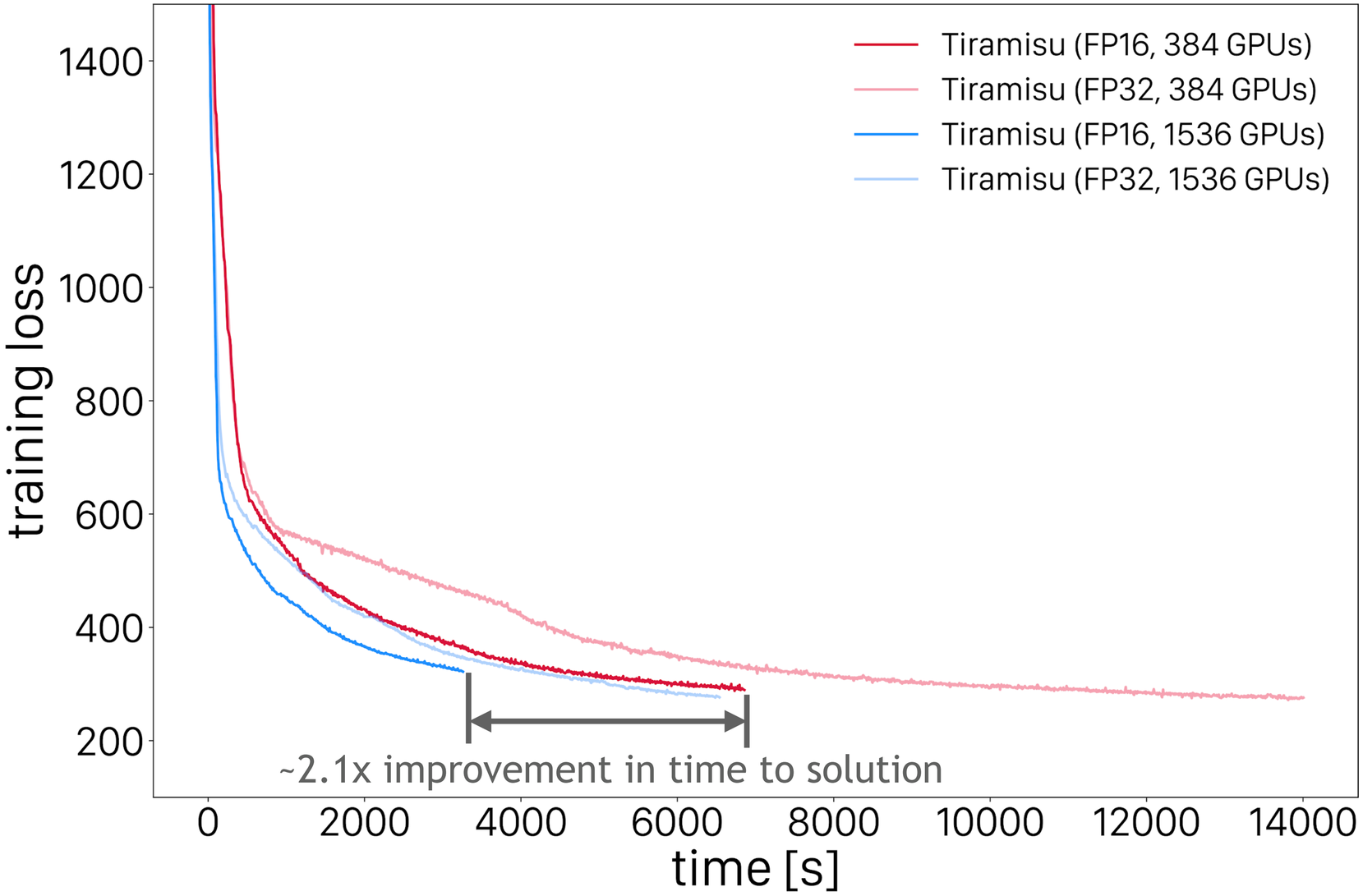
DeepLabv3+, 4560 nodes (27360 GPU)


ACM GORDON BELL PRIZE – WINNER
SCALABILITY AND TIME TO SOLUTION
“Exascale Deep Learning for Climate Analytics”
Research led by Thorsten Kurth
Lawrence Berkeley National Laboratory and NVIDIA

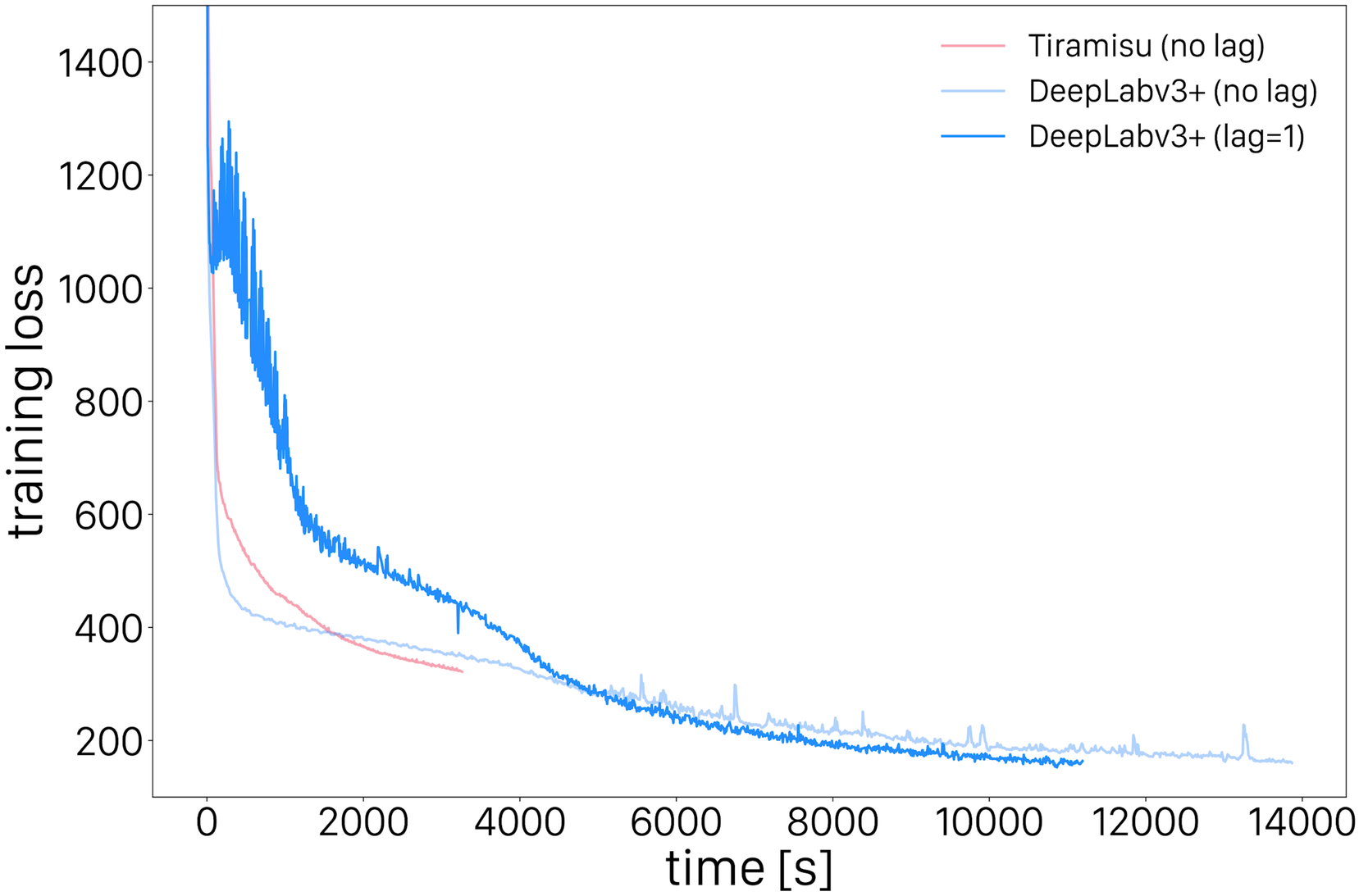
Concurrency/Precision and Convergence



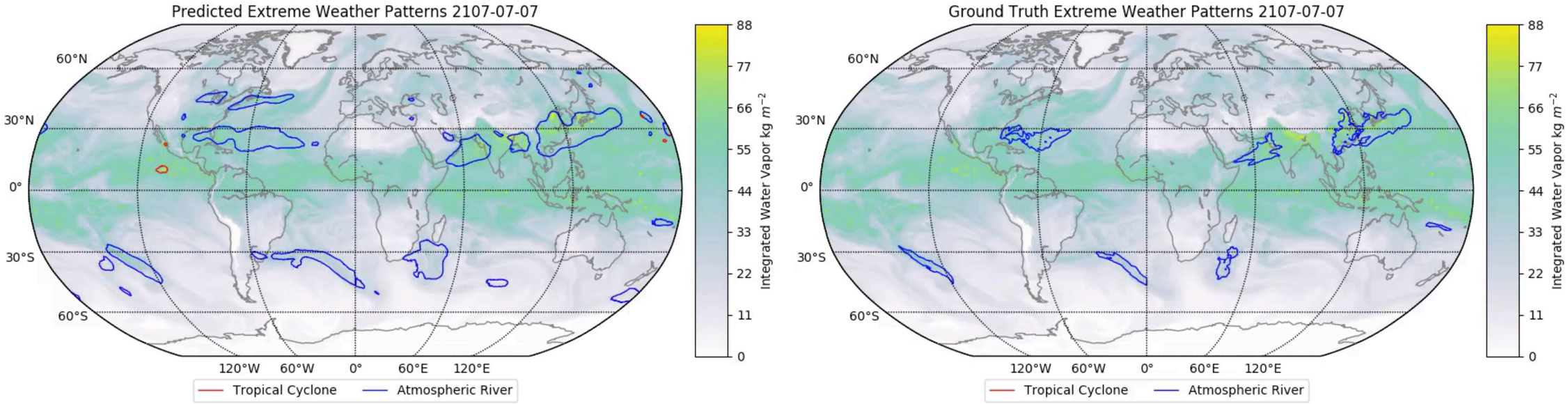
Concurrency/Precision and Convergence



Model/Lag and Convergence

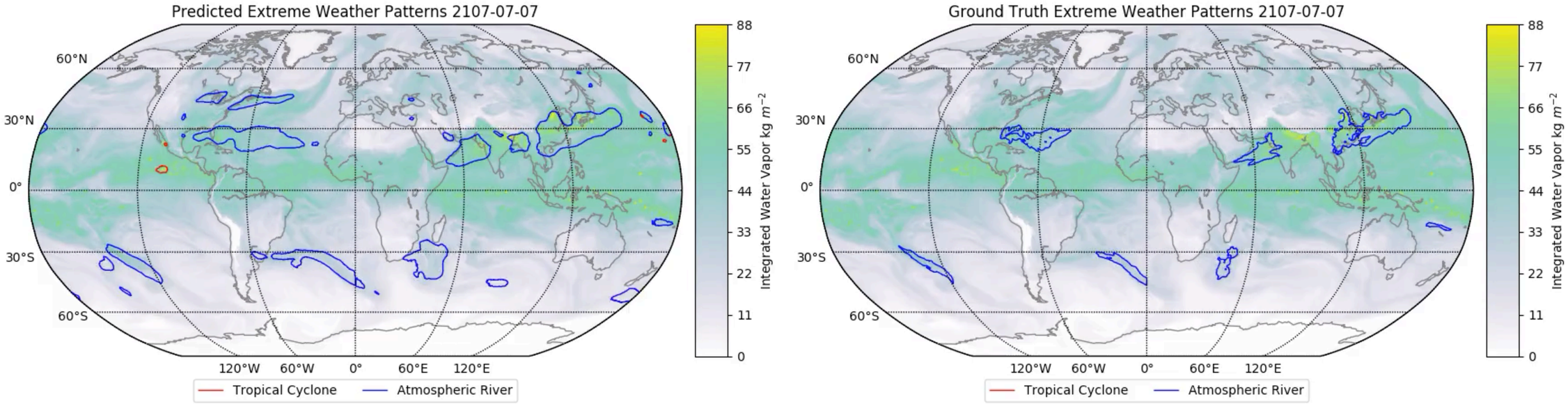


Segmentation Animation



- best result for intersection-over-union (IoU) obtained: ~73%
- result at large scale (batch-size > 1500): IoU ~55%

Segmentation Animation



- best result for intersection-over-union (IoU) obtained: ~73%
- result at large scale (batch-size > 1500): IoU ~55%

What has happened since Gordon Bell?

- Horovod:
  - Hierarchical (hybrid) allreduce available in upstream code (NVIDIA/Amazon)
  - Control plane bypass via caching available soon (NVIDIA)
- NCCL:
  - Version 2.4.x introduced tree-based reduction algorithms for improved performance at scale
- NERSC:
  - ClimateNet: human annotated labels from domain experts
  - project serves as science problem for the ISC-HPC19 student cluster AI challenge
  - Gained insights for design and evaluation of upcoming NERSC Perlmutter system

Conclusions

- deep learning and HPC converge, achieving *exascale* performance
- compute capabilities of contemporary HPC systems can be utilized to tackle challenging scientific deep learning problems
- HPO and convergence at scale still an open problem — but now we can do it
- software enhancements benefit deep learning community at large
- deep learning-powered techniques usher in a new era of precision analytics for various science areas

Thank You