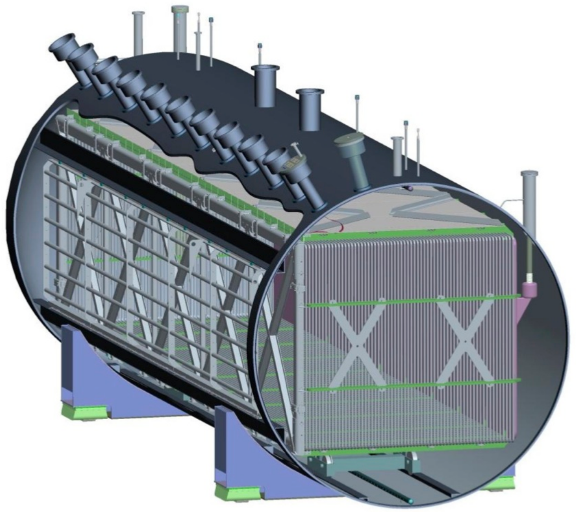
Scaling studies for deep learning in LArTPC event classification

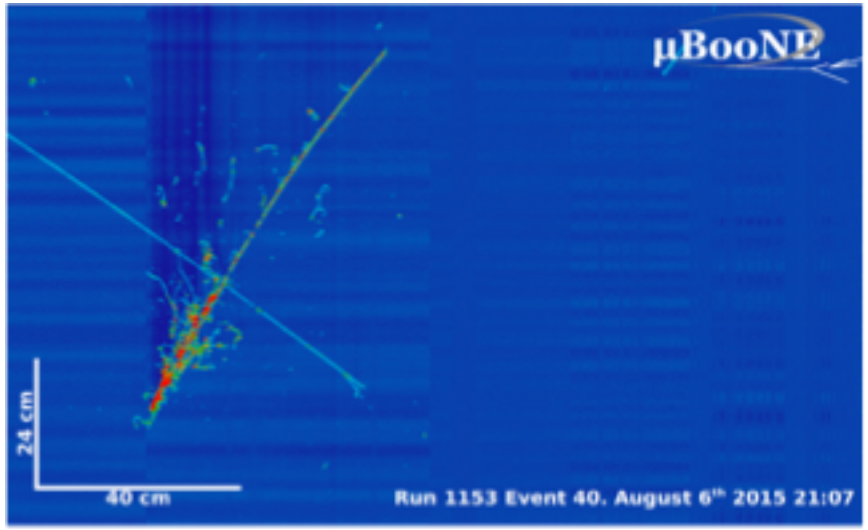
KOLAHAL BHATTACHARYA, ERIC CHURCH, MALACHI SCHRAM, JAN STRUBE, KEVIN WIERMAN, JEFF DAILY, CHARLES SIEGEL

Pacific Northwest National Laboratory

Introduction

- The MicroBooNE detector
  - 170 Tonne
    Liquid Argon Time Projection Chamber (LArTPC)
  - Readout:
    - 2 induction planes, 3256 wires
    - 1 collection plane, 3600 wires
    - 9600 digitizations ≅ 4.8 ms (~3x TPC drift length)
- The data
  - One event image is ~150 MB
    - Orders of magnitude larger than images for standard problems
  - We use simulated events for single particle interactions
- Disclaimer: Use of data is blessed by MicroBooNE, but this presentation is **not** on behalf of the collaboration





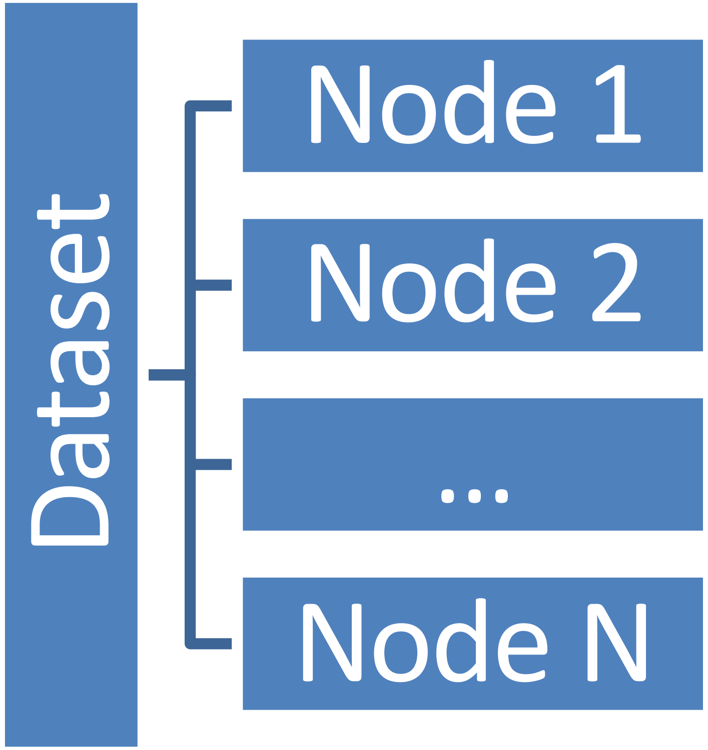
Technology choices

- Large event images lead to small batch sizes
  → Very slow gradient descent

- **MaTEx** (https://github.com/matex-org/matex) enables distributed training in TensorFlow / Keras with minimal code modfications
  - MPI for inter-node communication
- Distributed training allows to effectively scale the batch size with the number of nodes
  - More nodes → larger batch size → more efficient gradient descent (up to optimal value of batch size)
- Except for 3 lines of MaTEx setup, code is 100% valid Keras 2.0

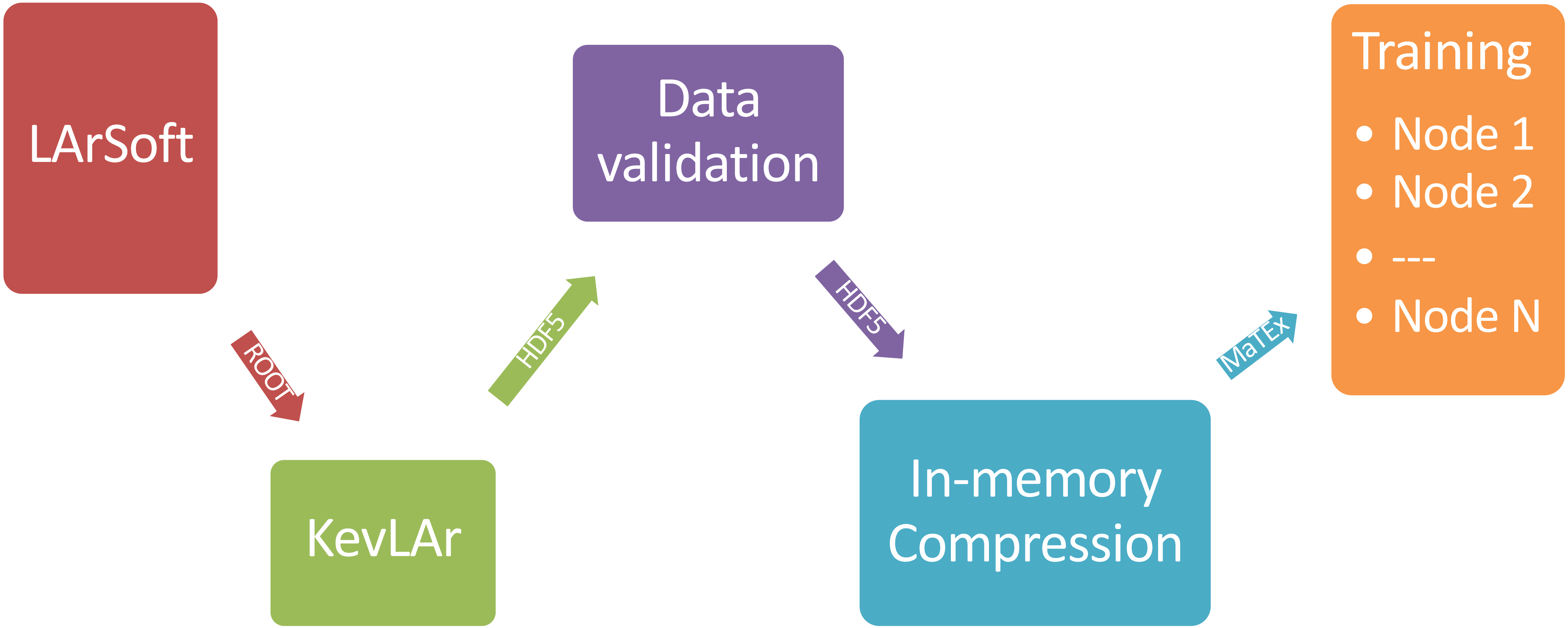
- In-memory compression: http://blosc.org/
  - We are using the python implementation: `pip install blosc`

- Dual Intel Broadwell E5-2620 v4 @ 2.10GHz CPUs
- Dual NVIDIA P100 12GB PCI-e based GPUs



each node gets an N-th chunk of the data

Putting it all together



Network and data

30k events for training

5k for validation

Aggregate weights

gamma: 873.00 e+-: 1622.00 mu+-: 856.00

pi+-: 826.00 K+: 823.00

Truth	Prediction: gamma	e+-	mu+-	pi+-	K+	highest score was correct:
gamma	851.47	21.53	0.00	0.00	0.00	**852**
e+-	8.19	1611.79	0.00	0.01	2.00	**1613**
mu+-	0.00	0.00	853.93	0.00	2.07	**854**
pi+-	2.90	3.87	3.00	483.68	332.54	**482**
K+	1.00	1.00	9.43	307.19	504.37	**508**

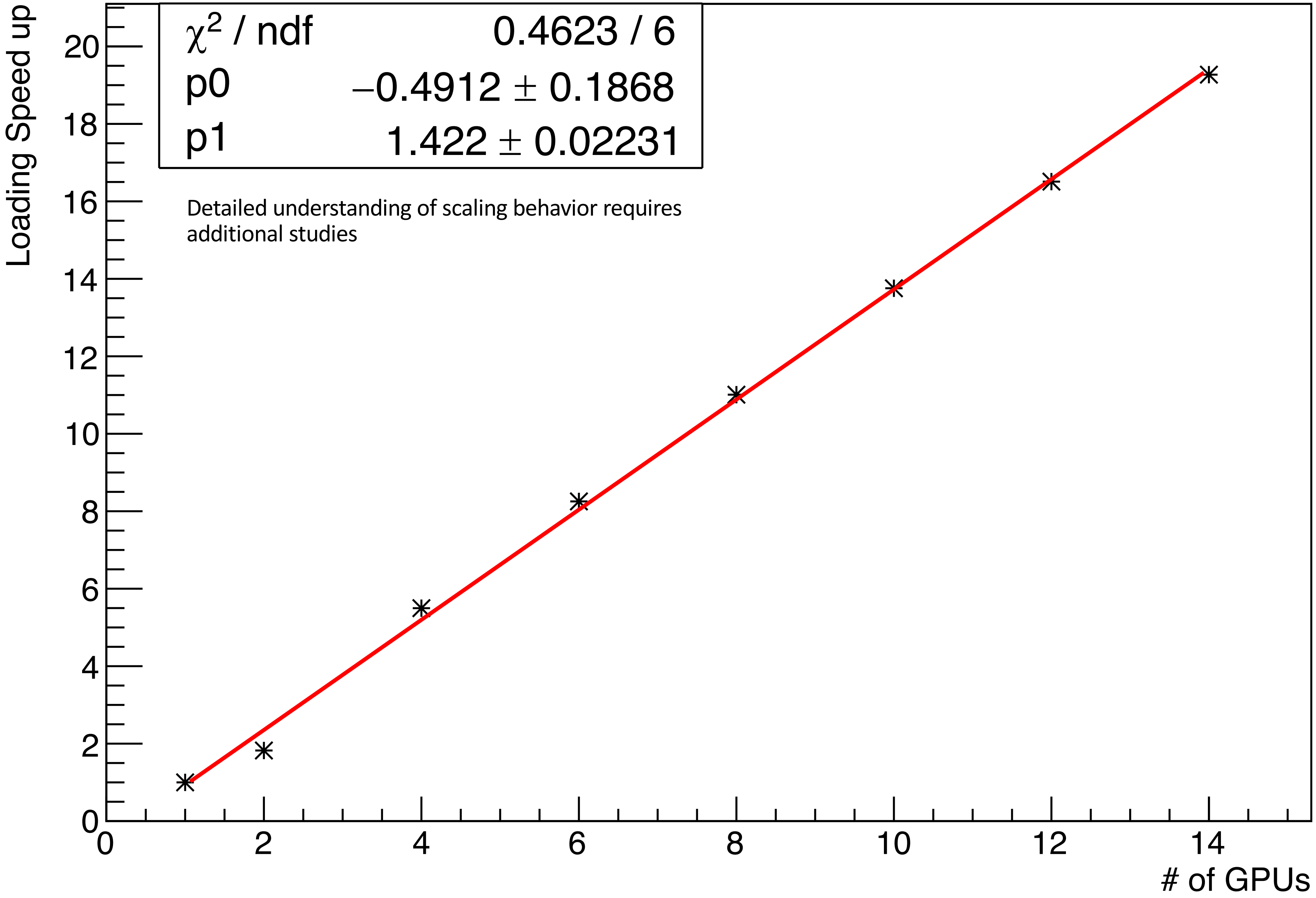
Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 3600, 3600, 10)	260
elu_1 (ELU)	(None, 3600, 3600, 10)	0
block1_pool (MaxPooling2D)	(None, 720, 720, 10)	0
block2_conv1 (Conv2D)	(None, 720, 720, 64)	16064
elu_2 (ELU)	(None, 720, 720, 64)	0
block2_pool (MaxPooling2D)	(None, 144, 144, 64)	0
block3_conv1 (Conv2D)	(None, 144, 144, 128)	204928
elu_3 (ELU)	(None, 144, 144, 128)	0
block3_pool (MaxPooling2D)	(None, 28, 28, 128)	0
block4_conv1 (Conv2D)	(None, 28, 28, 256)	819456
elu_4 (ELU)	(None, 28, 28, 256)	0
block4_pool (MaxPooling2D)	(None, 5, 5, 256)	0
flatten (Flatten)	(None, 6400)	0
fc1 (Dense)	(None, 32)	204832
elu_5 (ELU)	(None, 32)	0
predictions (Dense)	(None, 5)	165

Total params: 1,245,705

Training workflow

- Load the (modified) MaTEx dataset
  - Splits dataset into equal size chunks, one per MPI rank
- In each rank (node / GPU):
  - Load images into RAM
  - one at a time, compress, store in dictionary
- Load the Keras model, start training
- For each batch
  - Retrieve compressed images from datastore
  - Uncompress
  - move to GPU memory
  - learn
- Aggregate weights across nodes, average, update all nodes
- Rinse, repeat

Data throughput vs. number of nodes

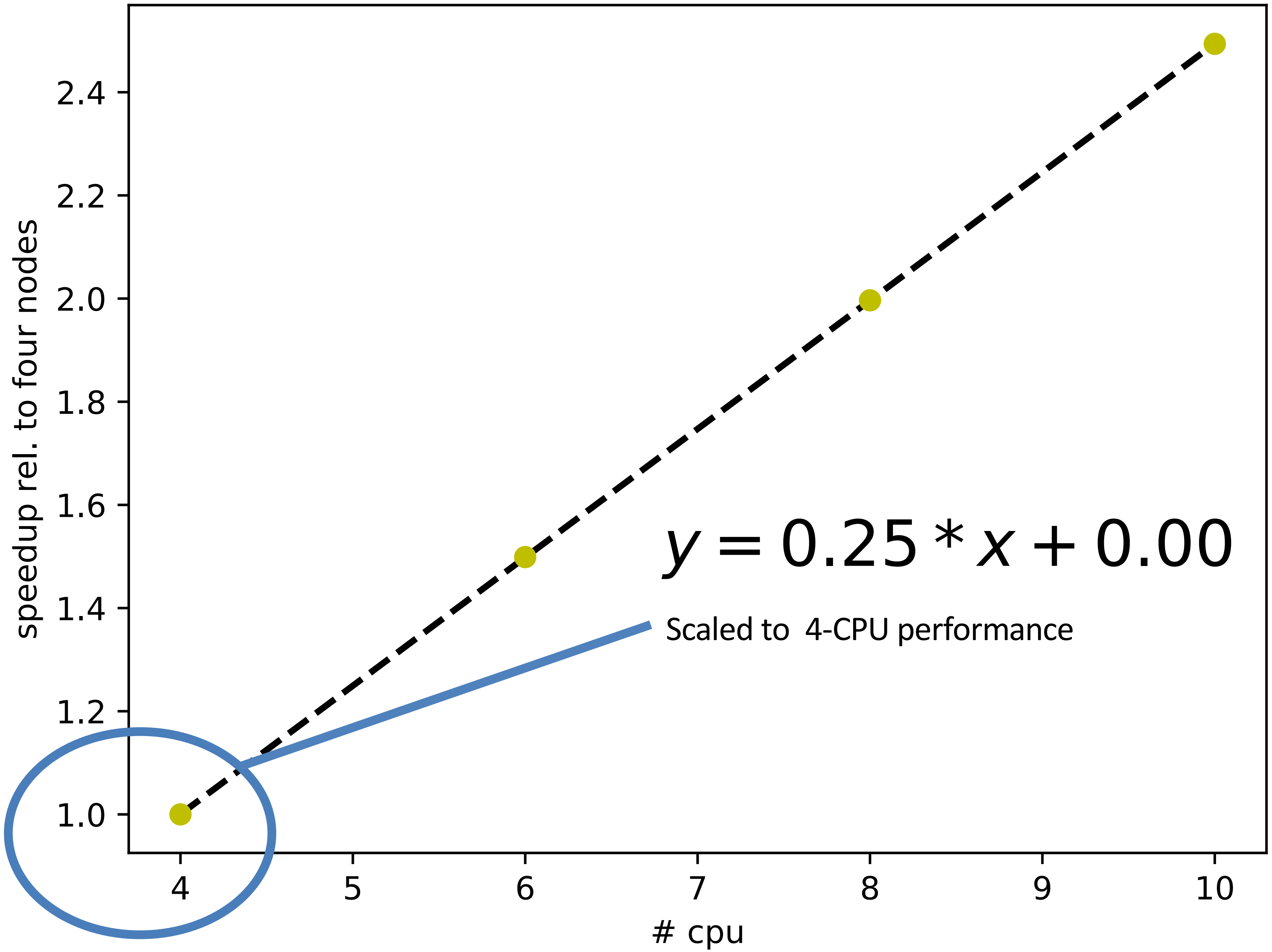


Measurement of the data distribution: Time to load all data (30k events) into memory
(2 GPUs share memory on the same node)

➞ More nodes == less work / node

Taking advantage of built-in multiprocessing capabilities.

Training speedup vs. number of CPUs



Relative speedup to train for 10(!) epochs

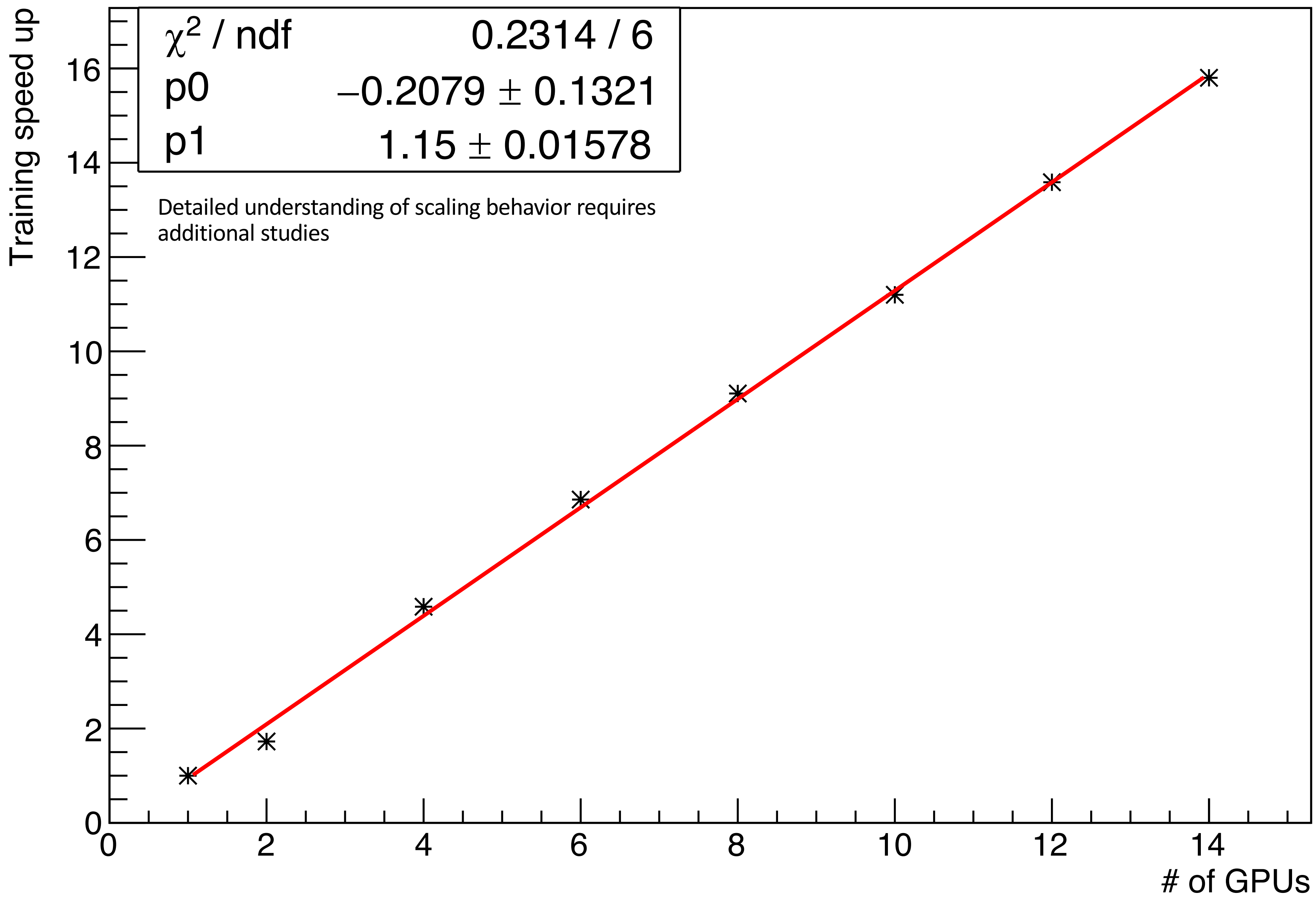
Jobs on 1 and 2 nodes did not complete in 4 days, hence omitted.

Jobs submitted to separate nodes (16-core).

Dual Intel Broadwell E5-2620 v4 @ 2.10GHz CPUs
64 GB 2133Mhz DDR4 memory per node

No work done to improve multi-core utilization over vanilla python / keras / tensorflow

Training speedup vs. number of GPUs

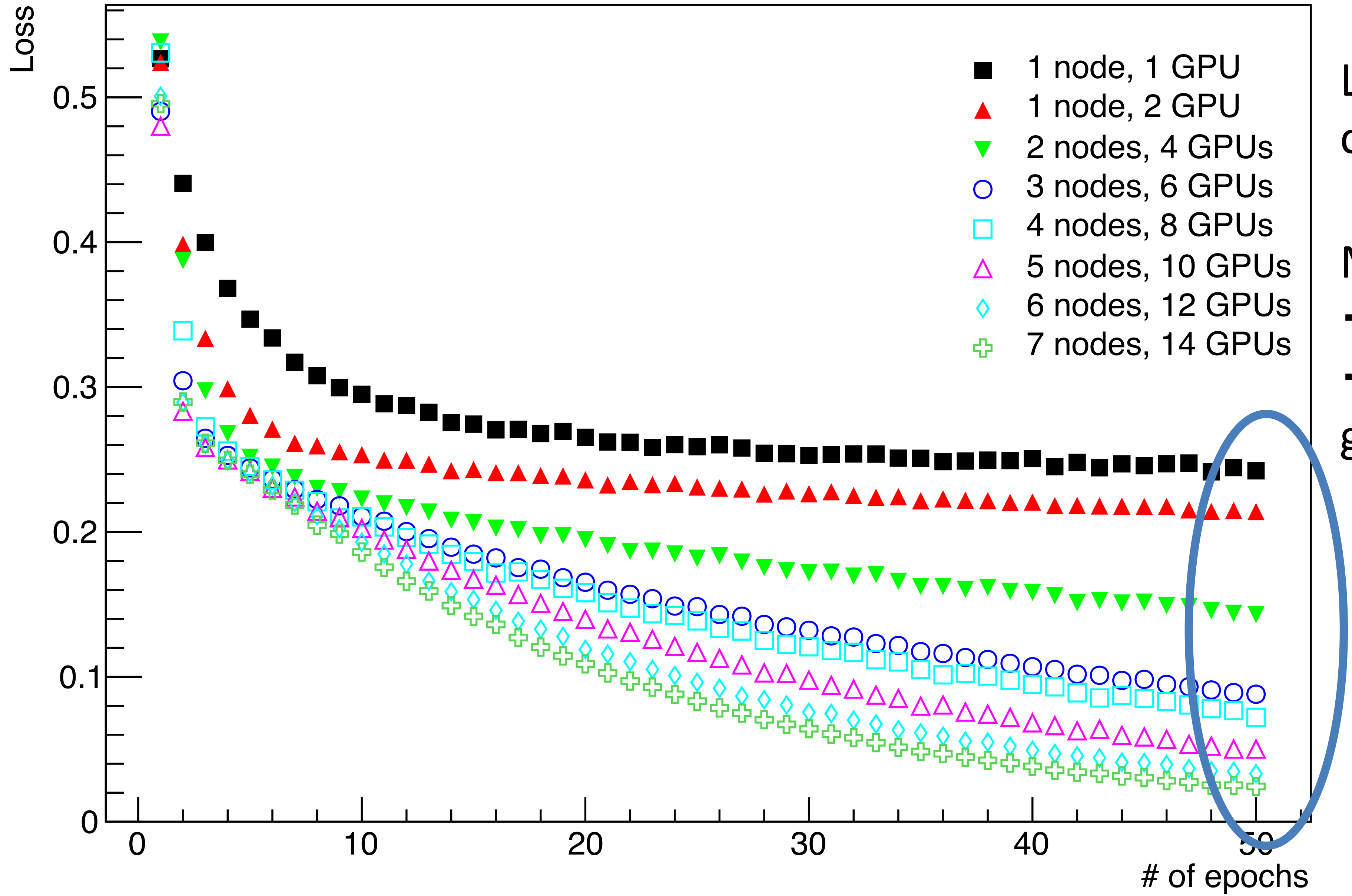


Time to train for 50 epochs

Time includes decompression of images and movement to the GPU

Relative speedup over CPU ~35
➞ I/O dominated

Loss performance for multiple nodes



Loss = categorical cross-entropy

More nodes
➞ larger batch size
➞ more efficient gradient updates.

In addition to training speedup, large difference in training performance with larger batch sizes

Conclusions

- Multi-node setup enables training on full-fidelity MicroBooNE event images.
  - This allows comparisons with the reduced images to evaluate the information loss in the size reduction.
  - linear scaling with the number of CPUs
  - (slightly better than) linear scaling on GPUs (up to the maximal number of 14 in our tests).
    - Detailed understanding of deviation from linear scaling would need further studies
    - Data loading mechanism and MPI behavior as bottlenecks on single node are possible sources.
- Multi-node training allows to effectively increase the batch size for convolutional networks of large event images.
  - demonstrated using MaTEx.
  - More efficient gradient updates require fewer epochs to arrive at the same loss (or lead to better loss after the same number of epochs)
  - LarTPC experiments clearly benefit from an HPC workflow.
- The project cycle is now completed.
  - Additional studies on OLCF's Summit (allocation available) pending HEP-ASCR funding.