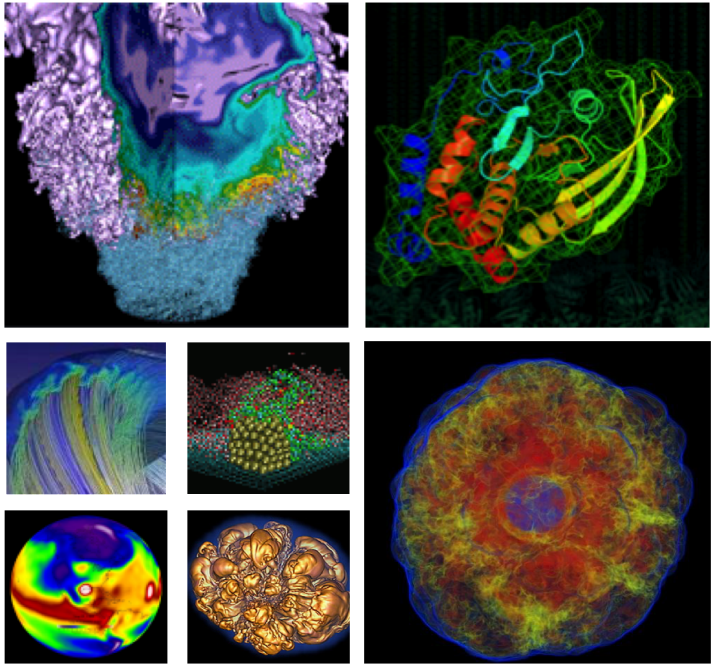
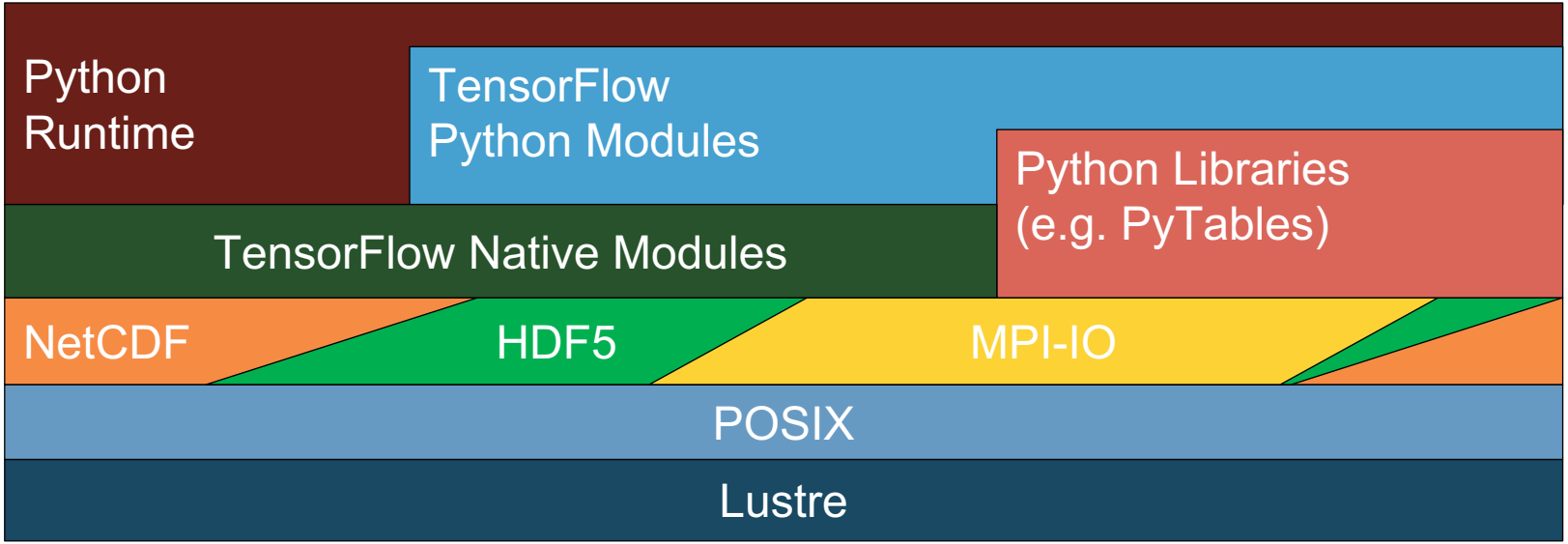
Big Data Center - Data Management Initiative



Quincey Koziol
BDC Summit
July 18, 2018

Data Management Technologies

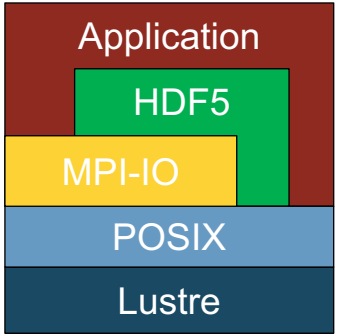
- **Scientific Data Analytics Software Stack**
  - Deep Learning on HPC IO Platforms



Data Management Challenges

- **Achieving good IO Performance is hard**
  - Tuning I/O is complex
    - Even along “the happy path”
    - Multi-layered stack
    - Scores of tunable parameters
  - What is not measured cannot be improved!
    - Determine baseline for every layer
    - Measure overhead for each component
    - Iterate: diff against baseline ☐ tune ☐ measure

HPC I/O



Data Management Challenges

- **“Big Data” adds new and different complexity**
  - More moving parts and layers
    - More permutations to analyze
    - Middleware/Runtime/Library implementation issues
  - Big Data >>> O (GB)
    - KNL Nodes on Cori:
      - 96 GB DDR4
      - 16 GB MCDRAM
    - Scientific Datasets Typically:
      - Very Large: O (TB, PB)
      - Very Complex: Hundreds of channels, Extreme Resolution

Factors Affecting I/O Performance

- **Workload Type (I/O, Network, Memory, Compute)**
  - Is I/O even the bottleneck?
    - What proportion of application runtime spent doing I/O?
    - Use case may be compute/memory bound
  - Extract and run I/O kernel (i.e. “null” computation)
  - Compare against known baselines:
    - Realistic Peak Throughput on Cori
      - 3 GB/s per Lustre OST (Theoretical)
      - ~1 GB/s per Haswell Core

Factors Affecting I/O Performance

- **File System (Physical Layout)**
  - Parallelism: Serial vs Parallel
  - Concurrency: Exclusive vs Shared
    - How many Lustre clients per OST?
    - How many threads (or cores) per client? (KNL requires more)
  - Striping Layout: width, count, alignment
- **Data Format (Logical Layout)**
  - Text vs Binary, Raw vs Compressed
  - Row-major vs Column-Major order, Chunked

Factors Affecting I/O Performance

- **Access Pattern: Sequential vs Strided vs Random**
  - Does access pattern match file layout?
    - Reading columns from data stored row-wise
    - Misalignment of data transfers with stripe boundaries
  - Use I/O profiling tools to examine pattern
    - e.g. Darshan; but only works with MPI

HDF5 in TensorFlow

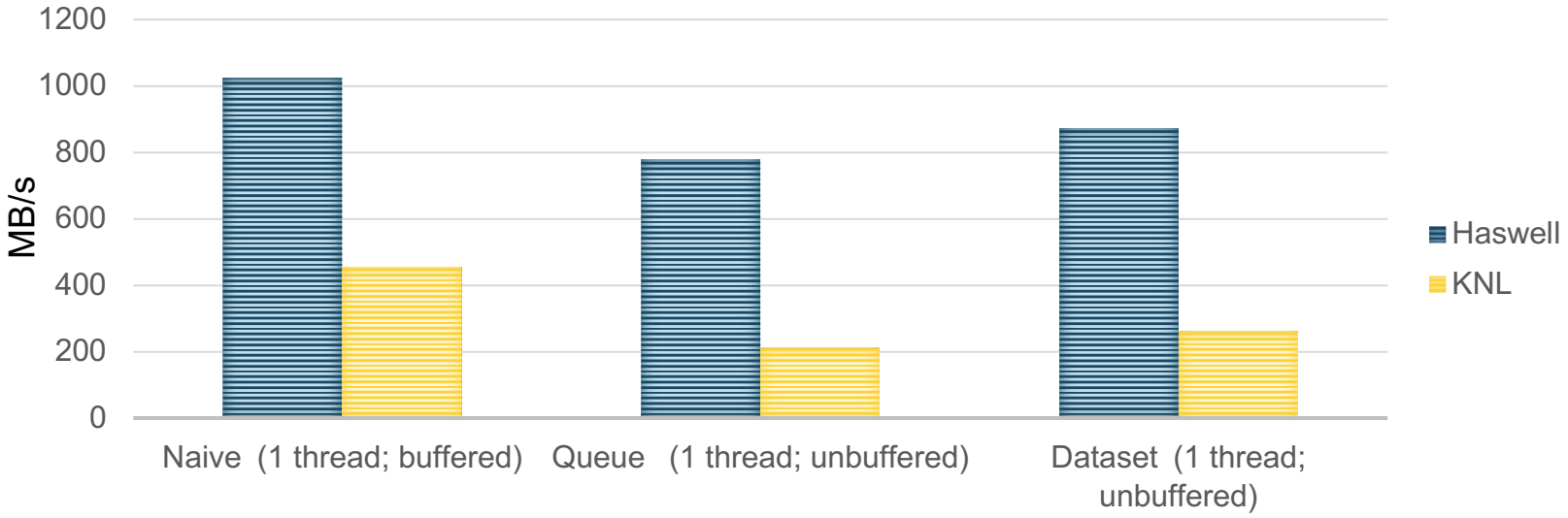
- **Requirements**
  - Feed data from HDF5 files to TensorFlow
    - And possibly other data analytics frameworks in the future
  - Needs to be fast and multi-threaded.
  - Buffer data for efficient shuffling
  - Allow overlapping I/O with computation
  - Flexible / Tunable with respect to
    - Number of files
    - HDF5 datatypes and array shapes

HDF5 in TensorFlow

- **Implementations**
  - Python-based File Iterator (Naïve)
    - Easy to implement and change; Customized for specific dataset
    - Not multi-threaded and cannot be used for shared files.
  - File Queue Runner (Deprecated)
    - Threaded I/O when working with multiple files; data prefetching
    - Tedious to implement, deadlock prone
  - Dataset API (Recommended)
    - Threaded I/O when working with multiple files; data prefetching
    - Must be implemented for every File System

HDF5 in TensorFlow

- **Preliminary Results**
  - Read 80 GiB from 4 HDF5 files stored on Lustre



Possible Next Steps

- **Benchmarking & Analysis**
  - Will need more real science applications to tune
  - Develop synthetic benchmarks from IO Kernels
- **Extending existing libraries for IO scaling**
  - Parallel I/O support for TensorFlow
    - FileSystem Interface?
    - MPI-IO over Lustre?
  - Multi-threading support for HDF5
- **Non-MPI Scaling & Parallel I/O**
  - Scale TensorFlow using novel exascale technologies?

Possible Next Steps

- **Explore I/O strategies for scaling Deep Learning**
  - Hyperparameter Optimization
    - Single data stream, no partitioning, multiple models
    - I/O Strategy: Embarrassingly Parallel, Shuffle (e.g. MPI_Allgatherv)
  - Data Parallelism
    - Multiple data streams, vertical partitioning, replicated model
    - I/O Strategy: Embarrassingly parallel, Stripe aligned
  - Model Parallelism
    - Multiple data streams, horizontal partitioning, distributed model
    - I/O Strategy: Read Coalescing, Chunking

Summer Project – Fahim Chowdhury

- **Evaluate Applications**
  - Instrument Applications and Frameworks
  - Find I/O Bottlenecks
  - Determine Solutions and Implement Them

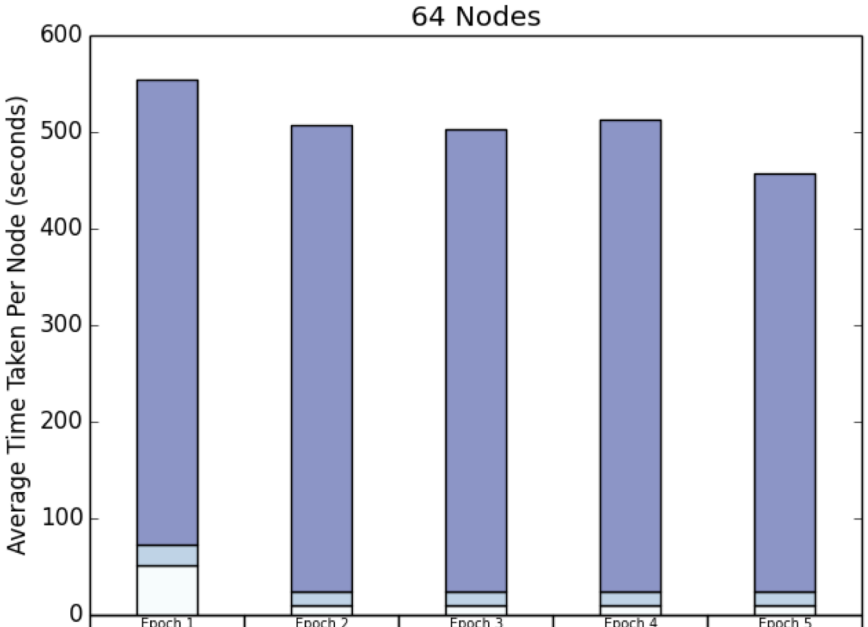
Outline

- HEP Benchmark
  - Time Breakdown and Bandwidth
- Climate Data Benchmark
  - Data Format
  - Time Breakdown and Bandwidth
- Future Exploration Scopes

Outline

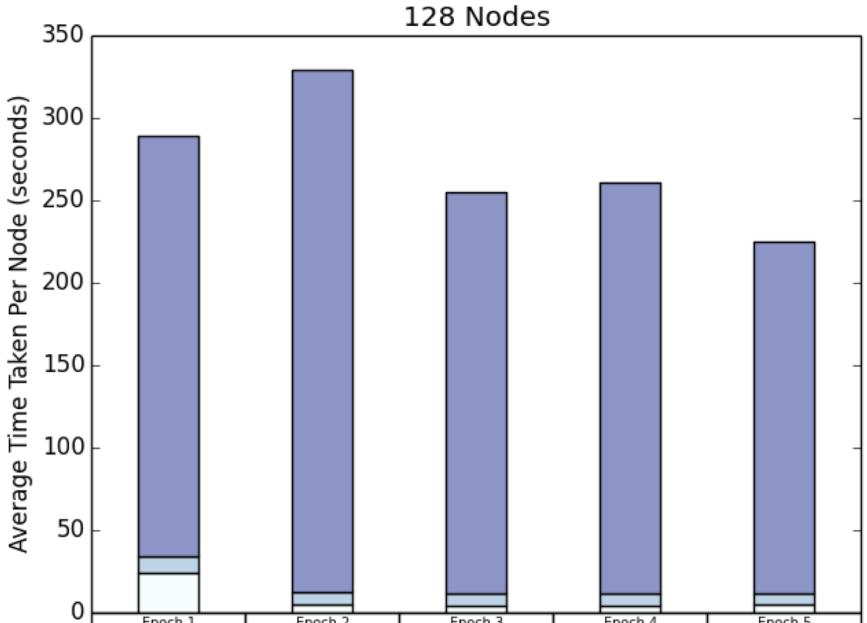
- HEP Benchmark
  - **Time Breakdown and Bandwidth**
- Climate Data Benchmark
  - Data Format
  - Time Breakdown and Bandwidth
- Future Exploration Scopes

64 Nodes



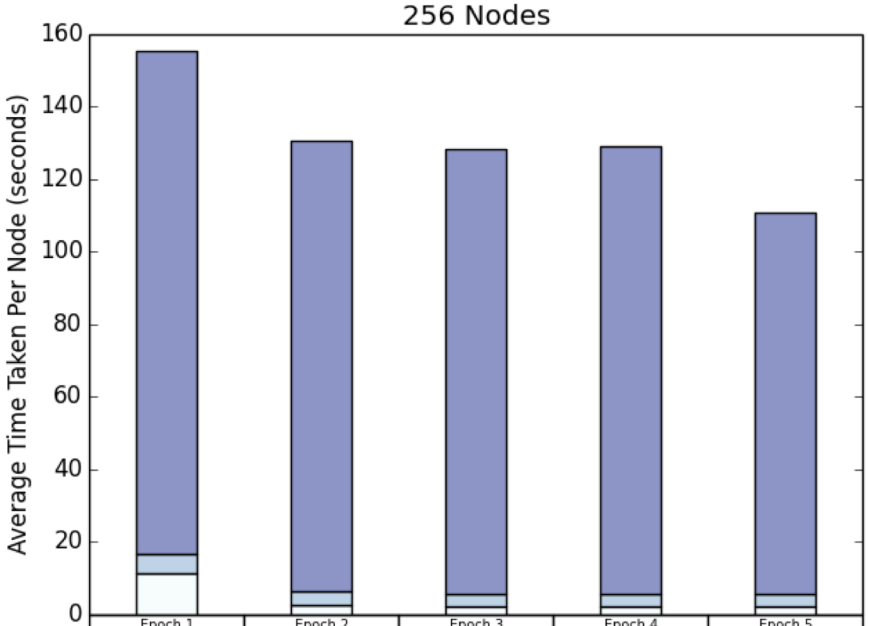
	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Read	51.88 (9.35 %)	9.32 (1.84 %)	9.76 (1.94 %)	9.58 (1.87 %)	10.67 (2.34 %)
Load File	21.19 (3.82 %)	14.46 (2.85 %)	14.00 (2.78 %)	14.01 (2.73 %)	14.02 (3.07 %)
Training Iteration	481.80 (86.83 %)	483.54 (95.31 %)	479.41 (95.28 %)	488.67 (95.40 %)	431.76 (94.59 %)

128 Nodes



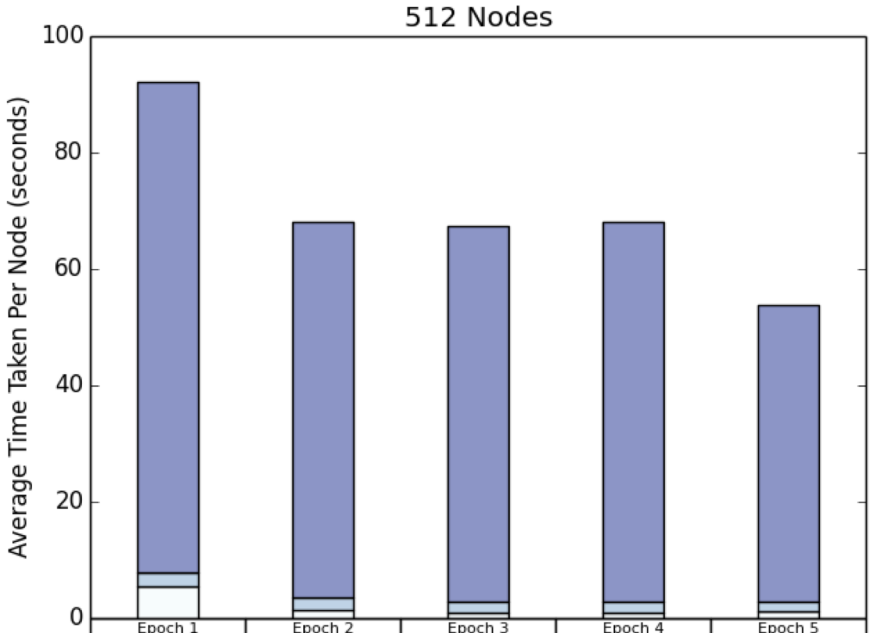
	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Read	23.81 (8.24 %)	4.81 (1.46 %)	4.24 (1.66 %)	4.25 (1.63 %)	4.76 (2.12 %)
Load File	10.45 (3.62 %)	7.47 (2.27 %)	7.01 (2.74 %)	7.01 (2.69 %)	7.01 (3.12 %)
Training Iteration	254.60 (88.14 %)	316.98 (96.27 %)	244.05 (95.59 %)	249.18 (95.68 %)	212.98 (94.76 %)

256 Nodes



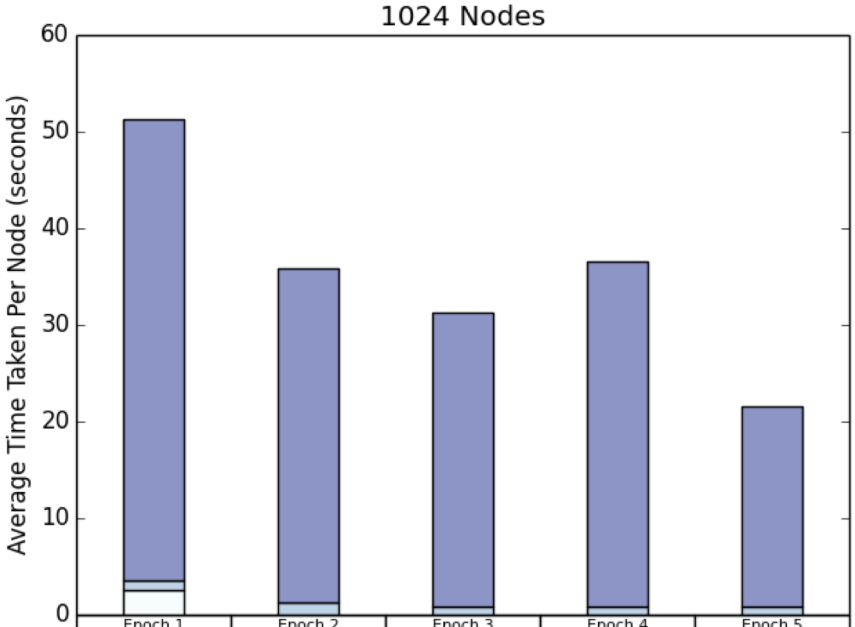
	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Read	11.57 (7.45 %)	2.55 (1.95 %)	2.34 (1.82 %)	2.11 (1.64 %)	2.13 (1.92 %)
Load File	5.03 (3.24 %)	3.96 (3.03 %)	3.50 (2.72 %)	3.50 (2.71 %)	3.50 (3.16 %)
Training Iteration	138.78 (89.31 %)	124.17 (95.02 %)	122.51 (95.45 %)	123.37 (95.65 %)	105.15 (94.92 %)

512 Nodes



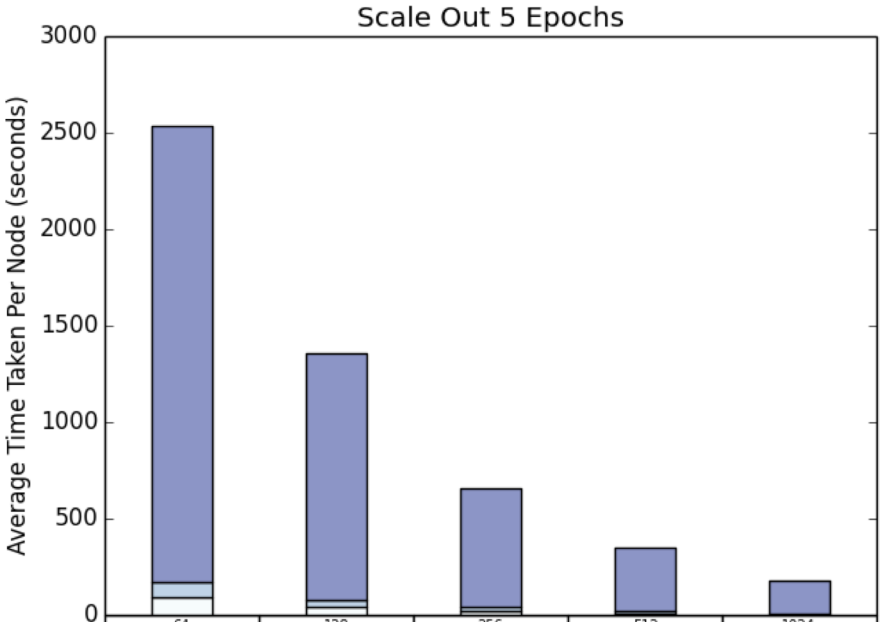
	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Read	5.49 (5.97 %)	1.46 (2.14 %)	1.06 (1.56 %)	1.06 (1.56 %)	1.09 (2.02 %)
Load File	2.35 (2.56 %)	2.21 (3.25 %)	1.75 (2.59 %)	1.75 (2.57 %)	1.75 (3.25 %)
Training Iteration	84.19 (91.48 %)	64.33 (94.60 %)	64.69 (95.84 %)	65.37 (95.88 %)	50.99 (94.73 %)

1024 Nodes



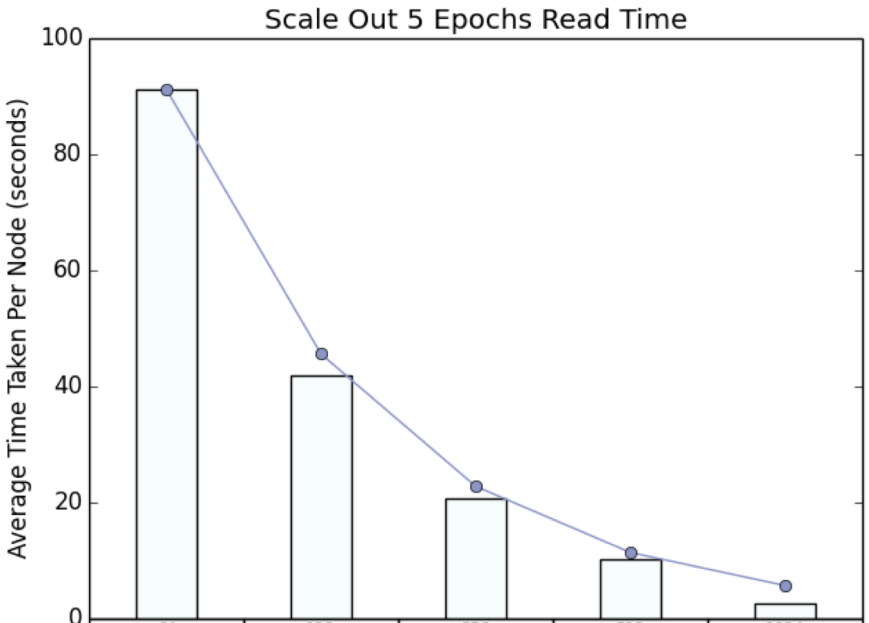
	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
Read	2.54 (4.95 %)	0.00 (0.00 %)	0.00 (0.00 %)	0.00 (0.00 %)	0.00 (0.00 %)
Load File	0.99 (1.93 %)	1.28 (3.58 %)	0.84 (2.68 %)	0.84 (2.29 %)	0.84 (3.90 %)
Training Iteration	47.77 (93.12 %)	34.56 (96.42 %)	30.44 (97.32 %)	35.72 (97.71 %)	20.73 (96.10 %)

Total Time Scale Out



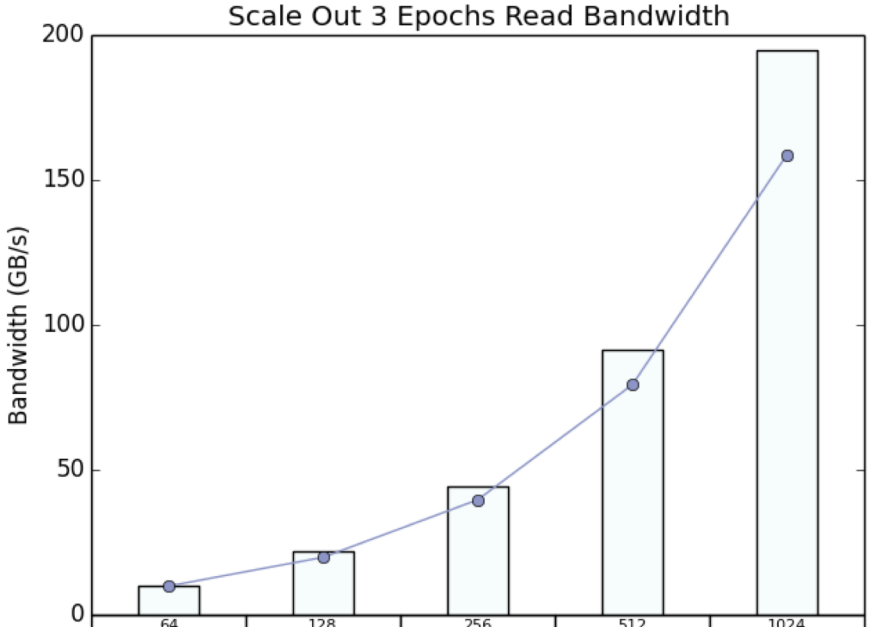
	64	128	256	512	1024
Read	91.21 (3.60 %)	41.87 (3.08 %)	20.71 (3.17 %)	10.16 (2.91 %)	2.54 (1.44 %)
Load File	77.68 (3.07 %)	38.95 (2.87 %)	19.49 (2.98 %)	9.81 (2.81 %)	4.79 (2.71 %)
Training Iteration	2365.17 (93.34 %)	1277.79 (94.05 %)	613.98 (93.86 %)	329.56 (94.29 %)	169.22 (95.85 %)

Read Time Scale Out



	64	128	256	512	1024
Read Time	91.21 (1.00X)	41.87 (1.09X)	20.71 (1.10X)	10.16 (1.12X)	2.54 (2.25X)
Ideal Read Time	91.2054123171	45.6027061585	22.8013530793	11.4006765396	5.70033826982

Read Bandwidth Scale Out



	64	128	256	512	1024
Read Count	9280	9344	9472	9728	2048
Read Time	91.2054123171	41.8738500997	20.7105450658	10.1580018685	2.53911591787
Read Bandwidth	9.917 (1.00X)	21.686 (1.09X)	44.195 (1.11X)	91.534 (1.15X)	194.994 (1.23X)
Ideal Case	9.91656529215	19.8331305843	39.6662611686	79.3325223372	158.665044674

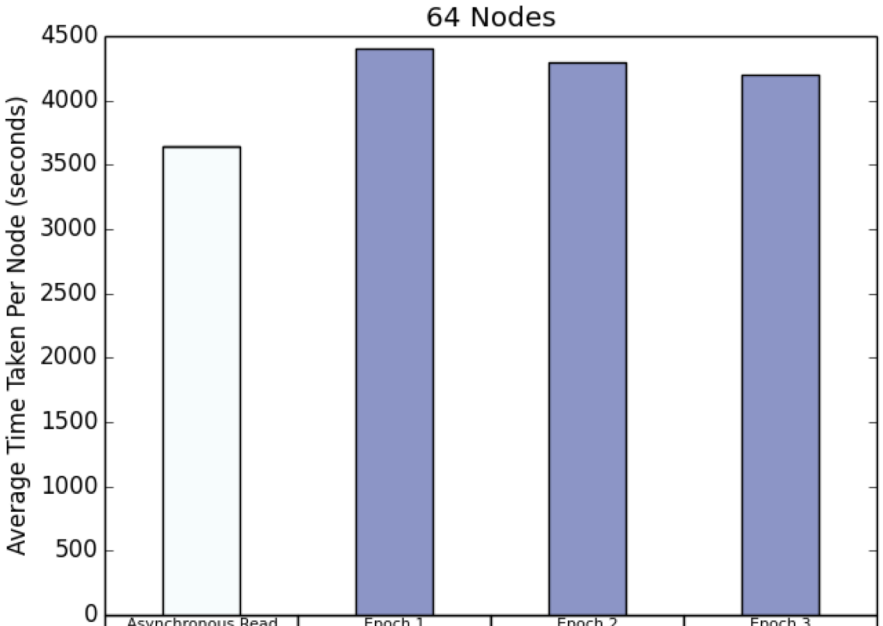
Outline

- HEP Benchmark
  - Time Breakdown and Bandwidth
- Climate Data Benchmark
  - **Data Format**
  - Time Breakdown and Bandwidth
- Future Exploration Scopes

Outline

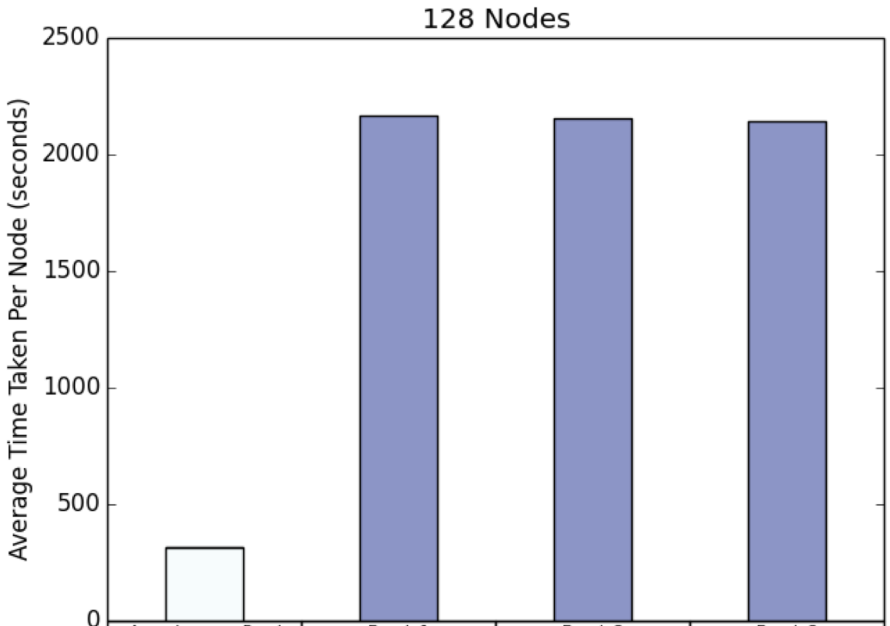
- HEP Benchmark
  - Time Breakdown and Bandwidth
- Climate Data Benchmark
  - Data Format
  - **Time Breakdown and Bandwidth**
- Future Exploration Scopes

64 Nodes



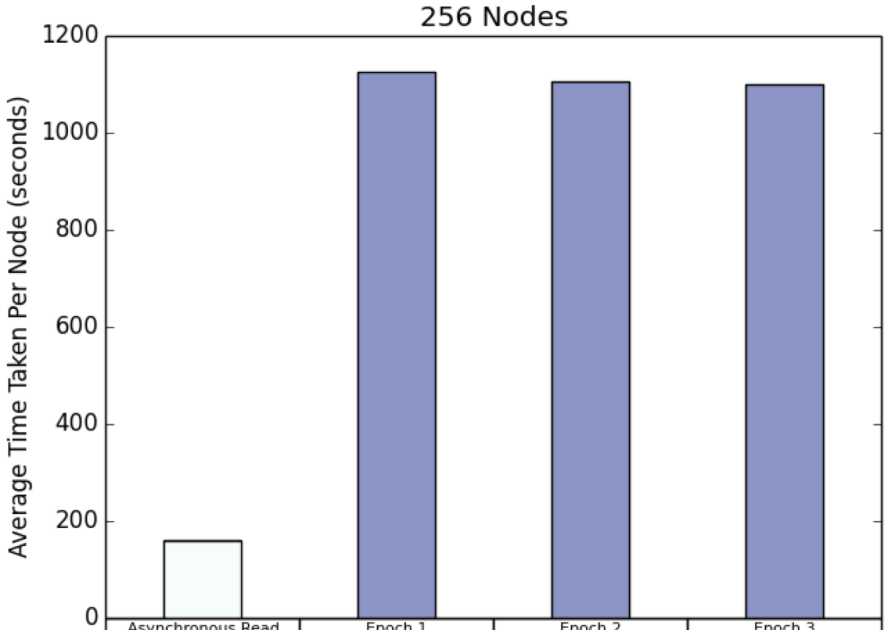
	Asynchronous Read	Epoch 1	Epoch 2	Epoch 3
Read	3640.89 (100.00 %)	0.00 (0.00 %)	0.00 (0.00 %)	0.00 (0.00 %)
Training Iteration	0.00 (0.00 %)	4407.48 (100.00 %)	4291.96 (100.00 %)	4199.03 (100.00 %)

128 Nodes



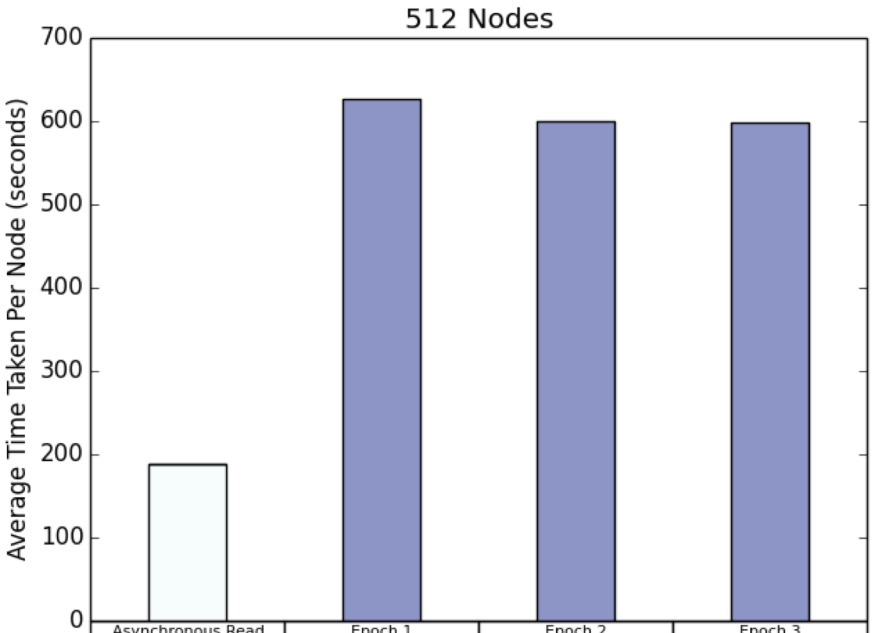
	Asynchronous Read	Epoch 1	Epoch 2	Epoch 3
Read	313.76 (100.00 %)	0.00 (0.00 %)	0.00 (0.00 %)	0.00 (0.00 %)
Training Iteration	0.00 (0.00 %)	2169.07 (100.00 %)	2153.28 (100.00 %)	2141.81 (100.00 %)

256 Nodes



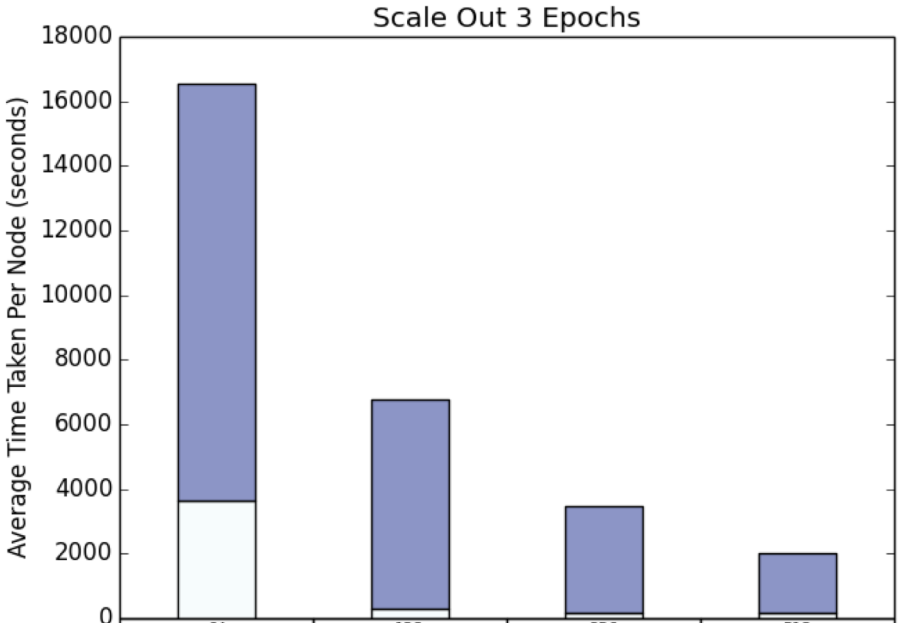
	Asynchronous Read	Epoch 1	Epoch 2	Epoch 3
Read	160.26 (100.00 %)	0.00 (0.00 %)	0.00 (0.00 %)	0.00 (0.00 %)
Training Iteration	0.00 (0.00 %)	1124.79 (100.00 %)	1104.39 (100.00 %)	1099.94 (100.00 %)

512 Nodes



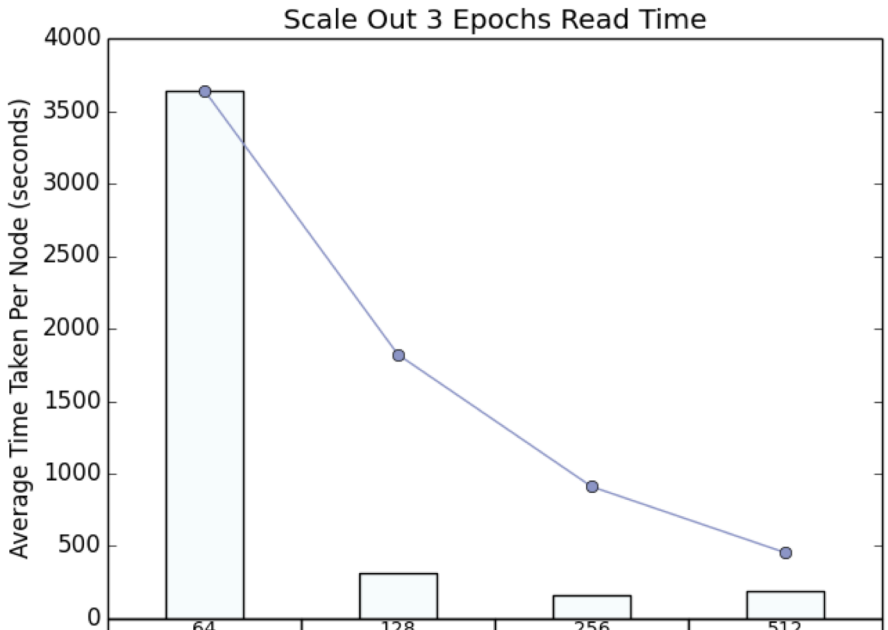
	Asynchronous Read	Epoch 1	Epoch 2	Epoch 3
Read	188.77 (100.00 %)	0.00 (0.00 %)	0.00 (0.00 %)	0.00 (0.00 %)
Training Iteration	0.00 (0.00 %)	626.99 (100.00 %)	599.85 (100.00 %)	598.40 (100.00 %)

Total Time Scale Out



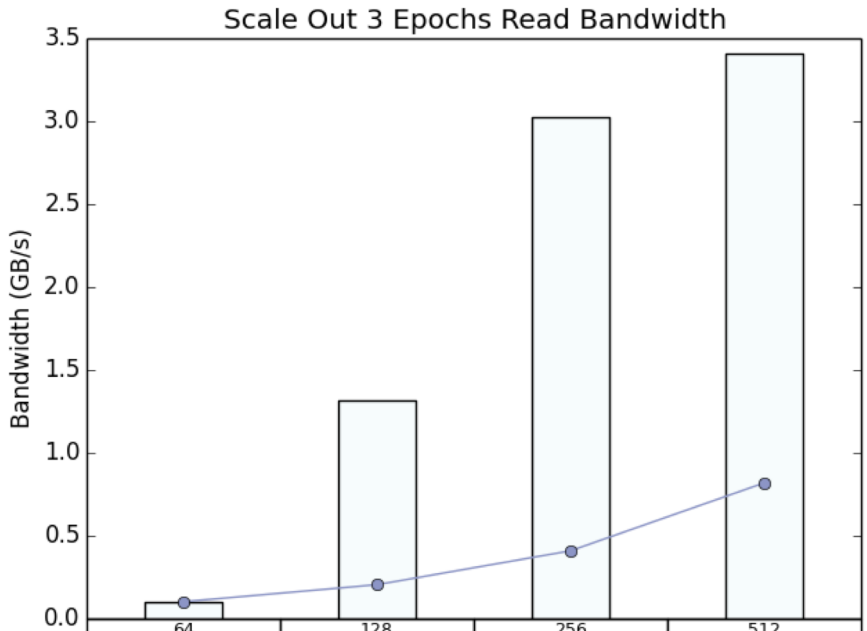
	64	128	256	512
Read	3640.89 (22.01 %)	313.76 (4.63 %)	160.26 (4.59 %)	188.77 (9.37 %)
Training Iteration	12898.48 (77.99 %)	6464.17 (95.37 %)	3329.12 (95.41 %)	1825.23 (90.63 %)

Read Time Scale Out



	64	128	256	512
Read Time	3640.89 (1.00X)	313.76 (5.80X)	160.26 (5.68X)	188.77 (2.41X)
Ideal Read Time	3640.89109441	1820.4455472	910.222773602	455.111386801

Read Bandwidth Scale Out



	64	128	256	512
Read Count	21056	23296	27392	36352
Read Time	3640.89109441	313.761952868	160.264374966	188.769511697
Read Bandwidth	0.102 (1.00X)	1.314 (6.42X)	3.024 (7.39X)	3.408 (4.16X)
Ideal Case	0.102332283073	0.204664566147	0.409329132293	0.818658264586

Outline

- HEP Benchmark
  - Time Breakdown and Bandwidth
- Climate Data Benchmark
  - Data Format
  - Time Breakdown and Bandwidth
- **Future Exploration Scopes**

Next Exploration Scopes

- Added Flag to Ignore Training
  - Can Try to Run the Tests with Training Disabled
- Added Provision to Test Checkpointing by *TimeLogger*
  - Can Try to Run the Tests with this feature
- Have to Perform Deeper Analysis on the Bandwidth for Parallel IO in Climate Data Benchmark
- To Have a Look into the *PyTorch Benchmark*
- To Try Running the Climate Data Tests Using Burst Buffer