I/O Characterization and Performance Evaluation of BeeGFS for Deep Learning

Fahim Tahmid Chowdhury*, Yue Zhu*, Todd Heer+, Saul Paredes*,
Adam Moody+, Robin Goldstone+, Kathryn Mohror+, Weikuan Yu*

Florida State University*
Lawrence Livermore National Laboratory+

8/12/19

48th International Conference on Parallel Processing (ICPP 2019), Kyoto, Japan

Outline

- Overview
- BeeGFS Introduction
- Deep Learning I/O Pattern
- Experimental Results and Key Observations
- Conclusion and Future Works

Outline

- **Overview**
- BeeGFS Introduction
- Deep Learning I/O Pattern
- Experimental Results and Key Observations
- Conclusion and Future Works

Motivations

- BeeGFS is an emerging Parallel File System (PFS) that is interesting for a systematic performance analysis
- Deep Learning (DL) applications are very common HPC workloads on modern supercomputers
- Most of the time DL applications read files of a dataset kept on PFS during training phase
- **Our research aims at evaluating BeeGFS focusing on DL application workloads on HPC systems**

Key Contributions

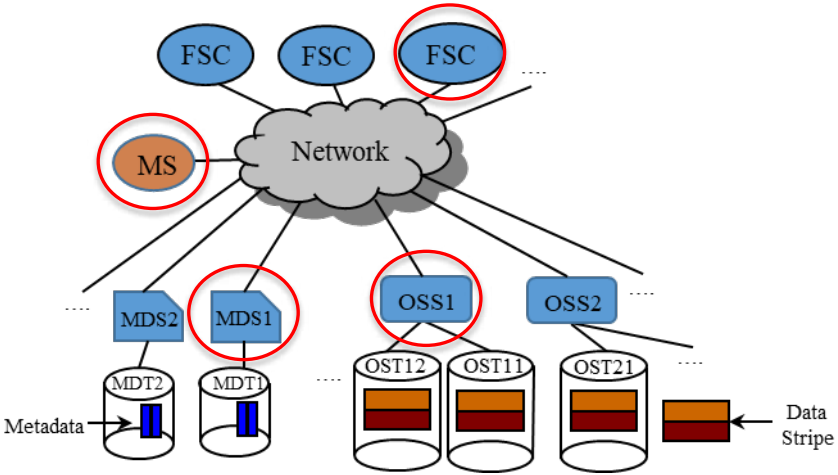
- Identify the I/O patterns posed by DL applications
- Evaluate the impact of read/write on BeeGFS
- Analyze metadata performance of BeeGFS
- Evaluate BeeGFS using DL application benchmarks

Outline

- Overview
- ➢ **BeeGFS Introduction**
- Deep Learning I/O Pattern
- Experimental Results and Key Observations
- Conclusion and Future Works

BeeGFS Software Architecture

- Management Server (MS)
  - Tracks initial connectivity information
- Metadata Server (MDS)
  - One Metadata Target per MDS
  - Maintains exclusive part of namespace
- Object Storage Server (OSS)
  - One or multiple Object Storage Targets (OST)
  - OST is generally RAID-set with POSIX compliant file system
- File System Client (FSC)
  - Kernel module to expose functions



Outline

- Overview
- BeeGFS Introduction
- ➢ **Deep Learning I/O Pattern**
- Experimental Results and Key Observations
- Conclusion and Future Works

Deep Learning Frameworks

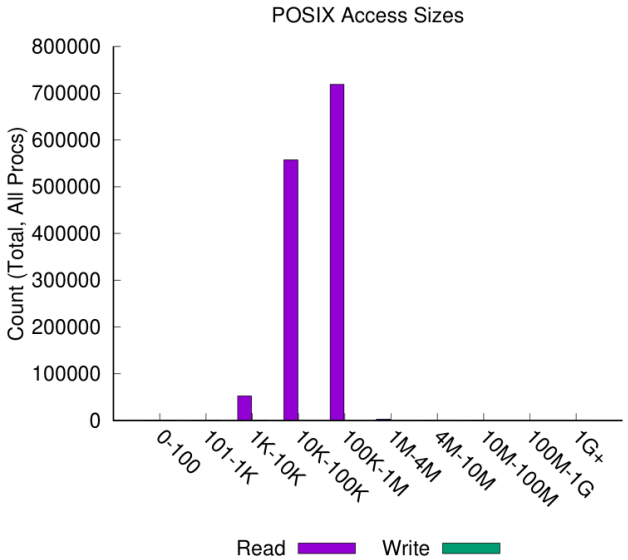
- Livermore Big Artificial Neural Network Toolkit (LBANN)
  - A neural network toolkit for HPC developed at LLNL
  - **Dataset specific specialized input pipeline**
- TensorFlow (TF) and Horovod
  - TF is a machine learning development platform by Google
  - Horovod helps enable HPC features in TF, Torch, etc.
  - **TF provides Dataset API for importing data while training**

Deep Learning Applications and Datasets

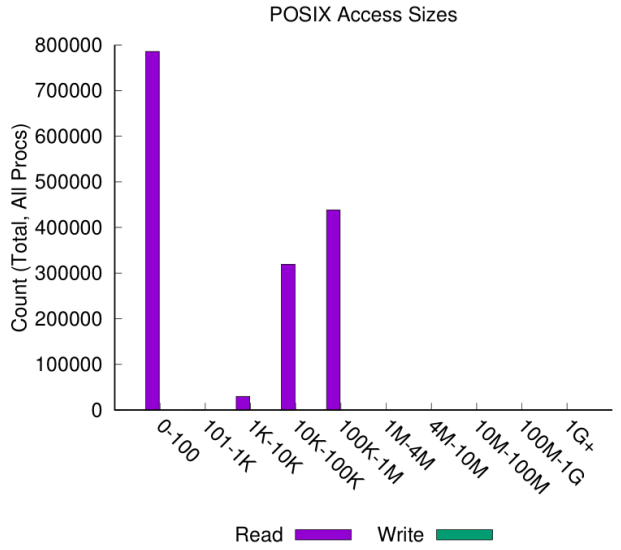
- AlexNet
  - 8 layer Convolutional Neural Network (CNN) for Image Recognition
- ResNet50
  - 50 layer Deep Residual Network for Image Classification
- ImageNet
  - Huge labelled image dataset with ~1.2 million images
  - Average file size is ~100 K

Deep Learning I/O Challenges on PFS

LBANN ImageNet Reader



TF Dataset API Input Pipeline



- Small I/O accesses for both LBANN and TF reported by Darshan-3.1.7
- Less data is read compared to huge metadata overhead
- No PFS caching due to randomization at each epoch

Outline

- Overview
- BeeGFS Introduction
- Deep Learning I/O Pattern
- ➢ **Experimental Results and Key Observations**
- Conclusion and Future Works

System Configuration

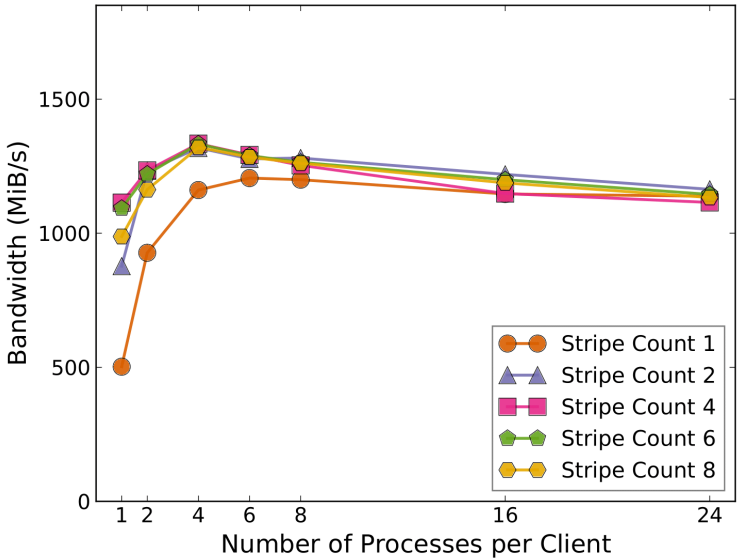
- All the experiments performed on **Catalyst** at LLNL
- **BeeGFS setup**
  - 338 TB BeeGFS-7.1.2 installation on Catalyst
  - 12 server hosts with QLogic Infiniband QDR interconnect
  - 12 OSS each with 2 OSTs
    - 1 OST: 4 HDD with ZFS RAID-Z and ZFS intent log on SSD
    - Level 2 Adjustable Replacement Cache on DRAM
  - 36 MDS each with 1 MDT
    - 3 MDS running per server host
    - 1 MDT: 4 SSD with ZFS RAID-Z formatting

Benchmarking Tools

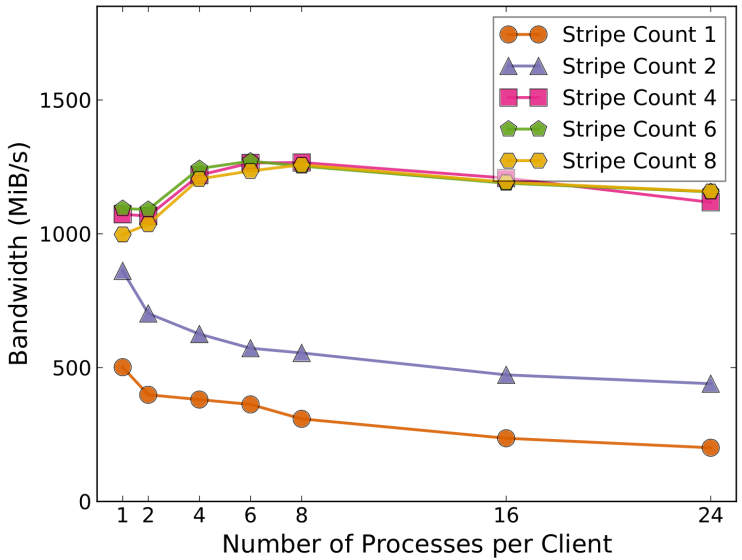
- Interleaved-Or-Random (IOR) version 2.10.3
  - File-per-process (N-N) and Single-shared-file (N-1) read/write with 240 GiB total file size
- MDTest-1.9.3
  - File create, stat and read operations
  - Shared-file, flat directory and single-depth hierarchical directory
- Deep Learning Benchmarks
  - AlexNet and ResNet-50 over ImageNet on LBANN
  - ImageNet reader pipeline on TF Dataset API and Horovod

Single Client Read Performance

File-per-process (N-N)



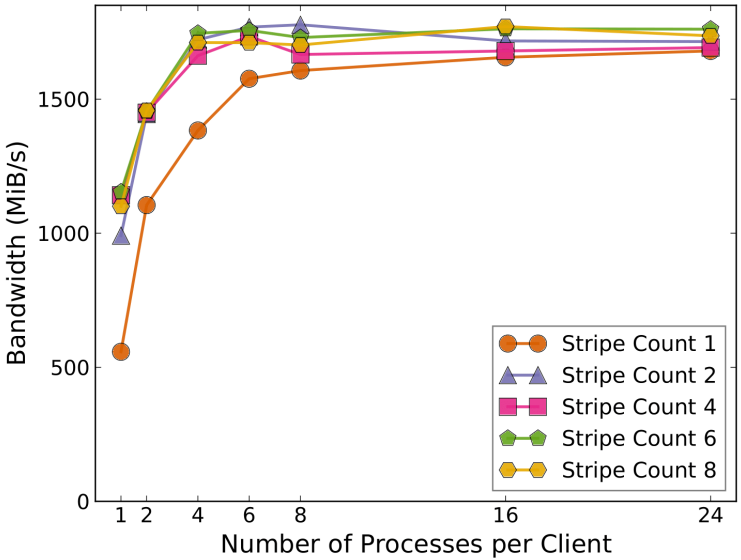
Single-shared-file (N-1)



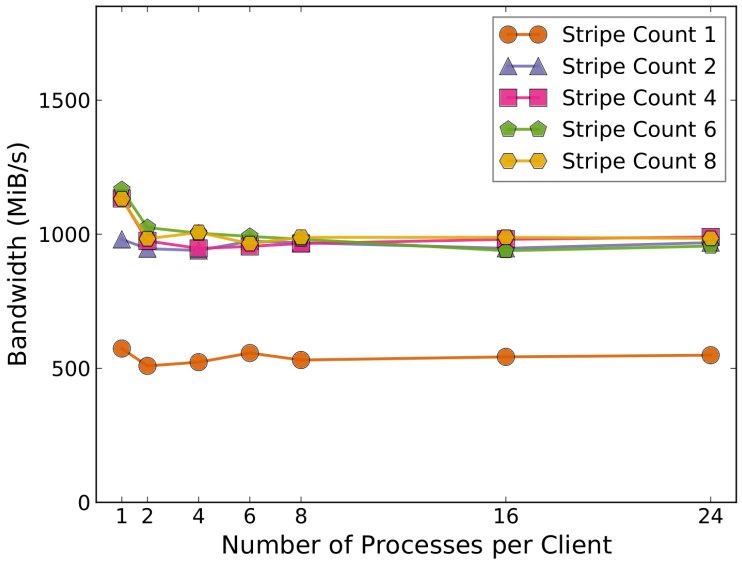
- N-N Read bandwidth reaches saturation point for network bandwidth
- N-1 Read bandwidth behaves differently for stripe count lower than 4

Single Client Write Performance

File-per-process (N-N)



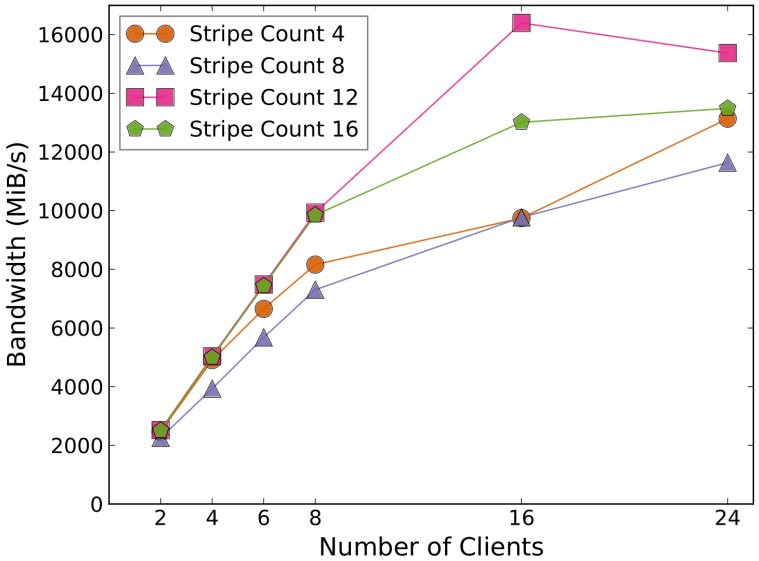
Single-shared-file (N-1)



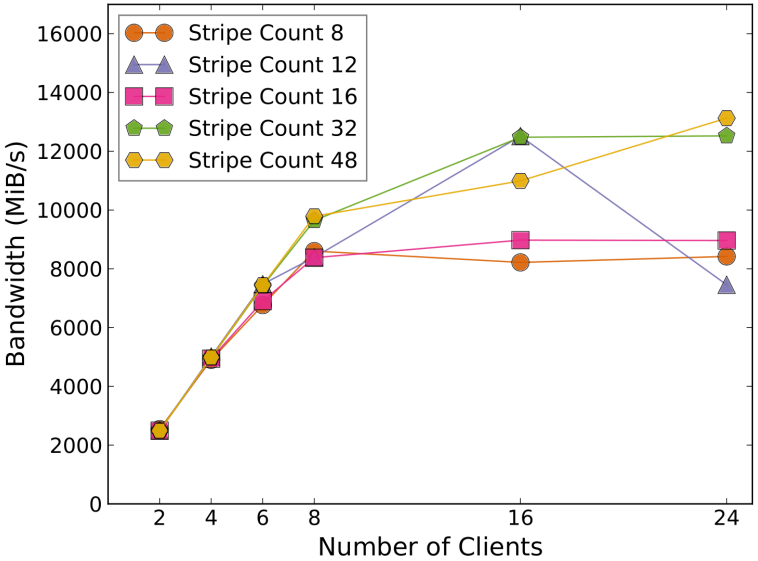
- N-N write bandwidth increases with process count and saturates
- N-1 write bandwidth does not scale as good as N-N write

Scale-out Read Performance

File-per-process (N-N)



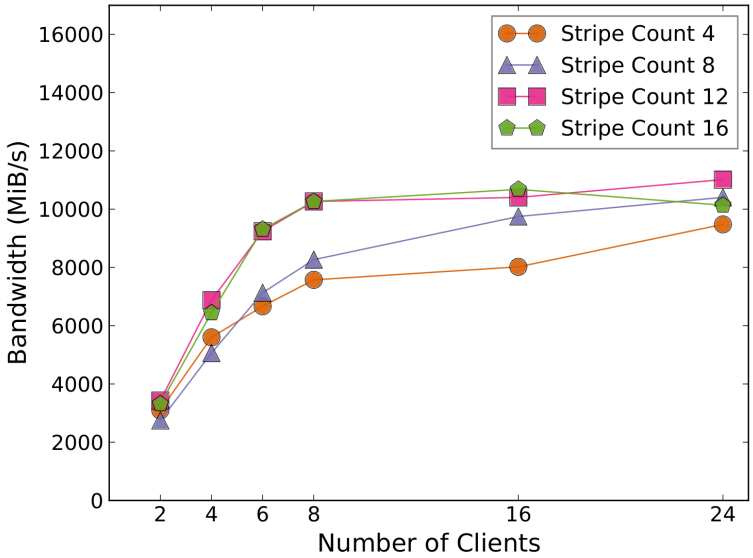
Single-shared-file (N-1)



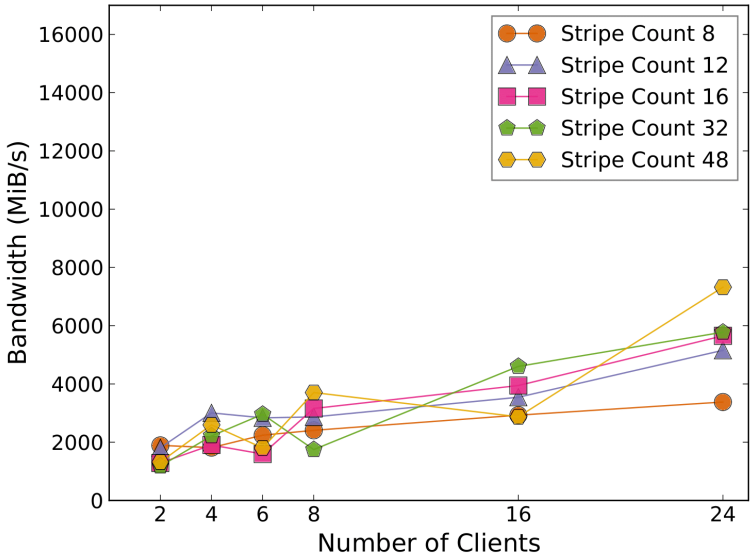
- N-N read bandwidth scales smoothly with increasing number of clients
- Bandwidth of N-1 read is slightly lower that of N-N read

Scale-out Write Performance

File-per-process

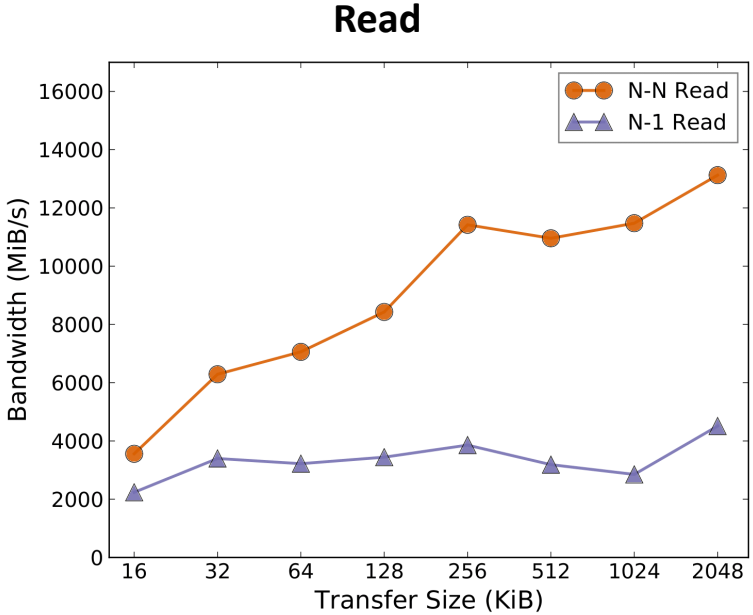


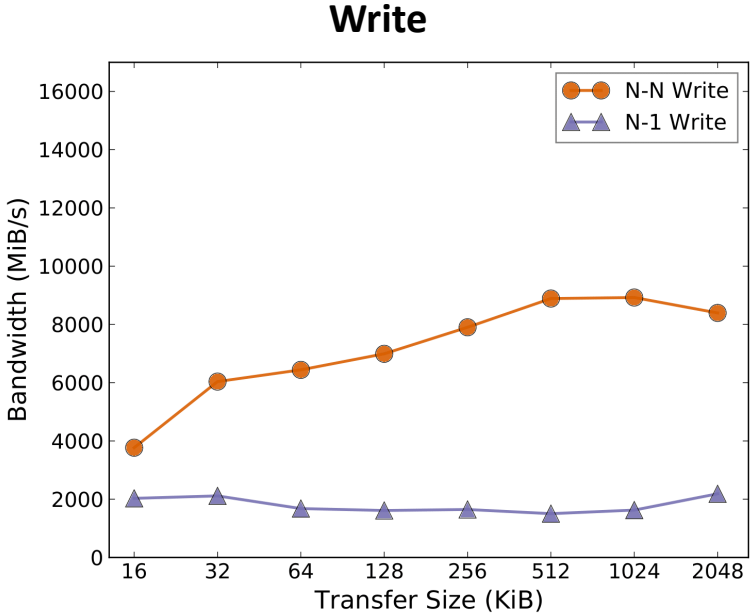
Single-shared-file



- N-N write bandwidth shows good scalability
- N-1 write bandwidth is almost half of N-N write bandwidth

Multi Client Varying Transfer Sizes



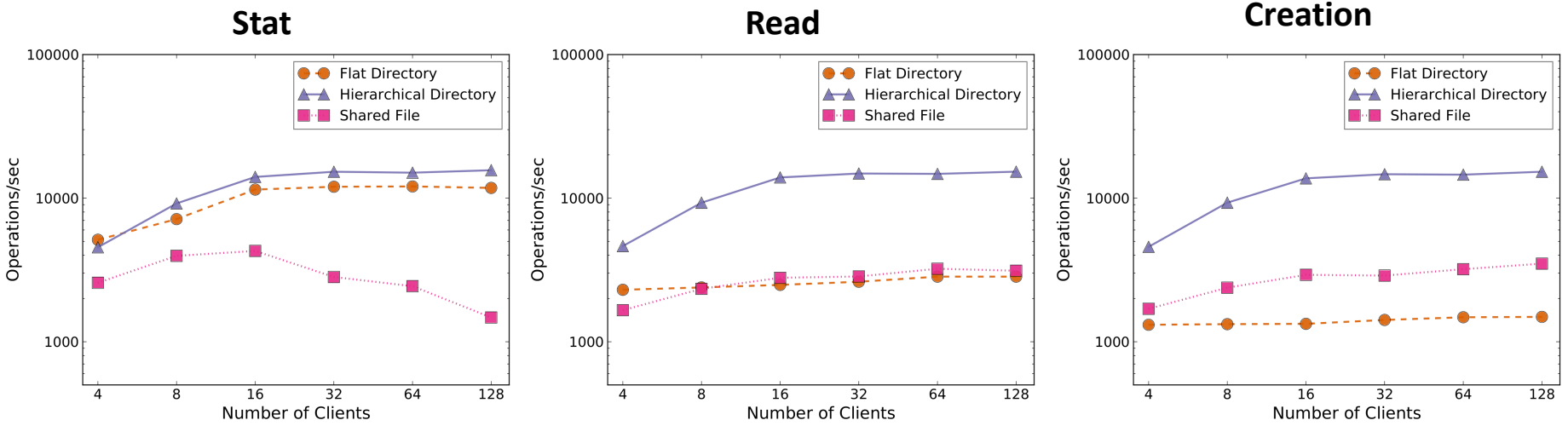


- Vary the transfer size for 16 clients and 8 processes per client
- N-N workloads handled better with increasing transfer size
- N-N bandwidths are better than N-1 as usual

I/O Observations

- N-N workload is almost always better than N-1
  - N-N read workload is closer to actual DL training I/O
  - Checkpointing and logging should avoid N-1 write
- Increasing stripe count is not always necessary
  - Need to balance resource utilization and performance
  - Lower striping leaves more devices available with reasonable performance

Multi Client Metadata Performance



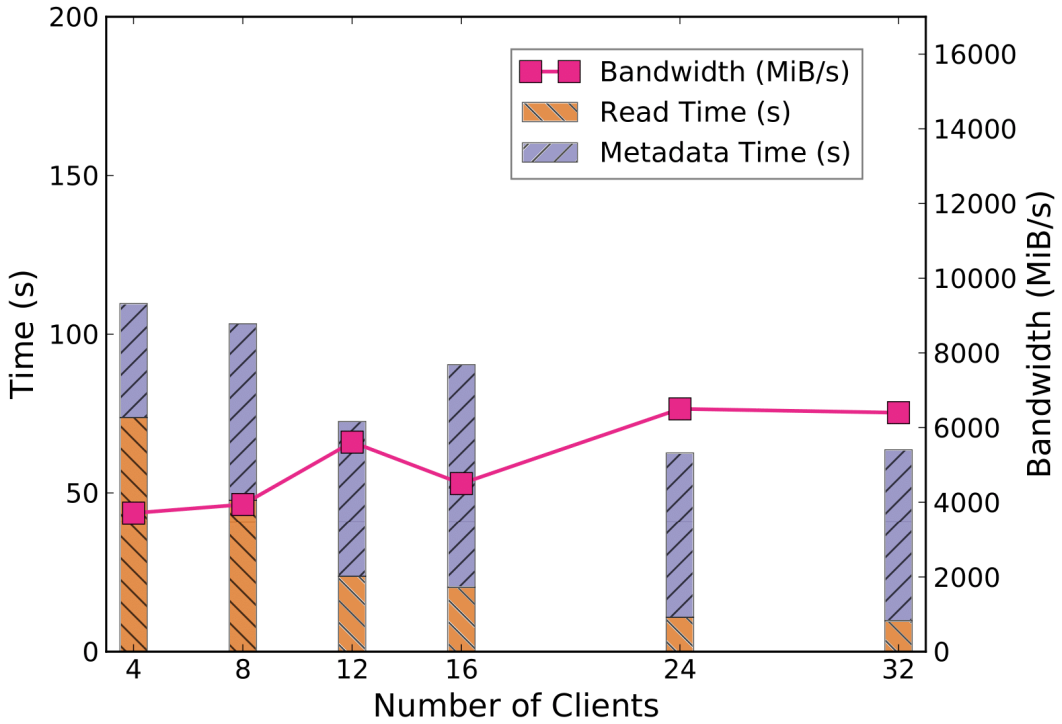
- File stat performance is good for both flat and hierarchical directory structures
- Single-depth hierarchical directory structure is the best for file open and close
- File creation metadata operation can leverage hierarchical directory structure

Metadata Observations

- Hierarchical file organization is beneficial to metadata management in BeeGFS
  - Dataset files should be arranged in hierarchical directories
  - ImageNet files are arranged in single-depth subdirectories

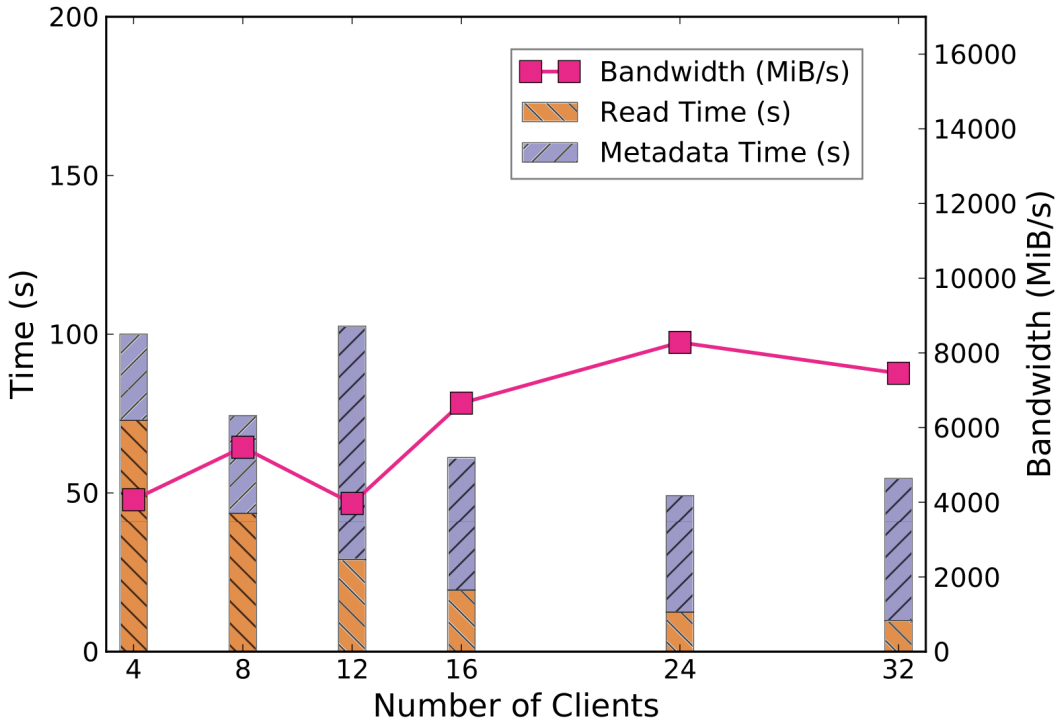
AlexNet over ImageNet on LBANN

- Read time decreases with increasing nodes
- Read bandwidth does not scale much
- Metadata overhead governs the overall throughput



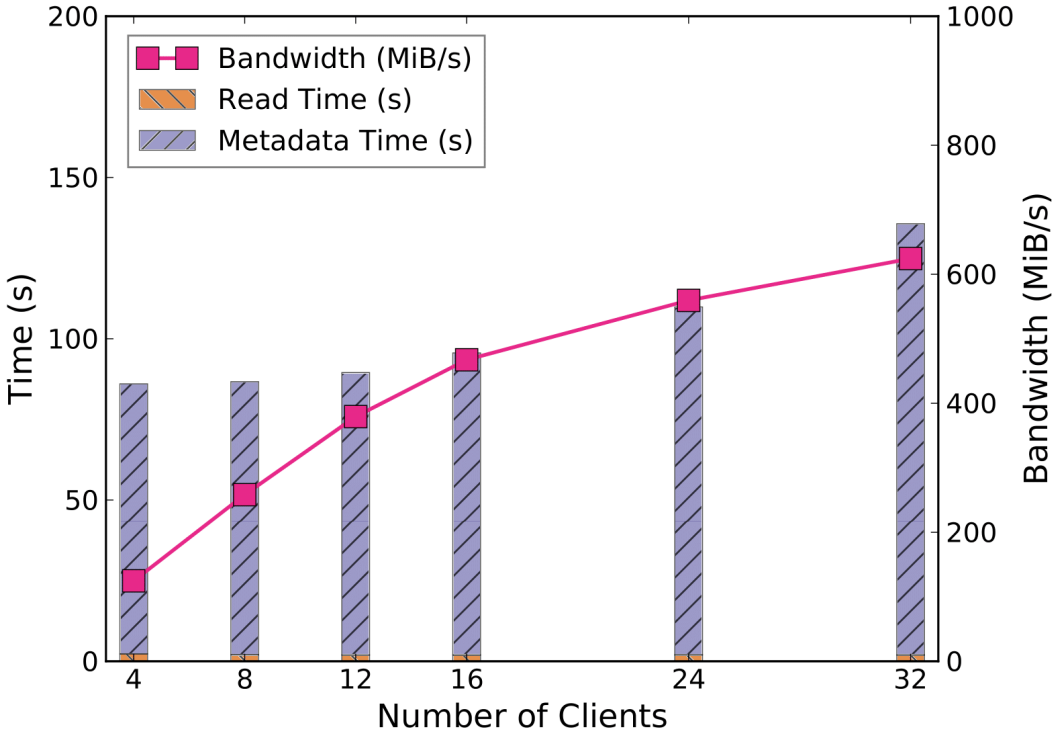
ResNet50 over ImageNet on LBANN

- AlexNet and ResNet50 both uses the same pipeline
- ImageNet dataset specific input pipeline in LBANN
- Similar trends for read throughput



ImageNet Data Reader on TF Dataset API

- Initial level tests on distributed TensorFlow
- Less data read compared to LBANN and bandwidth is low
- Metadata is a notable bottleneck



DL Application Benchmark Observations

- BeeGFS can reasonably handle LBANN DL I/O pattern
  - Read time lessens with more clients
  - Metadata overhead increases less steeply
- The directory structure of ImageNet is helpful for BeeGFS
  - Single-depth hierarchical directory structure in ImageNet assists metadata manager
- TF offers internal optimizations for file reading
  - tf.read_file through python wrapper invokes tiny file reads
- Metadata handling is a notable bottleneck in both LBANN and TF

Outline

- Overview
- BeeGFS Introduction
- Deep Learning I/O Pattern
- Experimental Results and Key Observations
- **Conclusion and Future Works**

Conclusion and Future Works

- We perform microscopic data and metadata I/O performance evaluation of BeeGFS
- We discuss all the results and observations in the context of Deep Learning I/O patterns on BeeGFS
- We plan to perform more in-depth analysis on I/O workload by applications built atop TensorFlow
- We want to compile characterization data on more BeeGFS features, e.g., Storage Pool, BeeOND, etc.

Sponsors

This work is performed under the auspices of the U.S. Depart- ment of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-PRES-784219. This work is also supported in part by the National Science Foundation awards 1561041, 1564647, 1744336, 1763547, and 1822737.

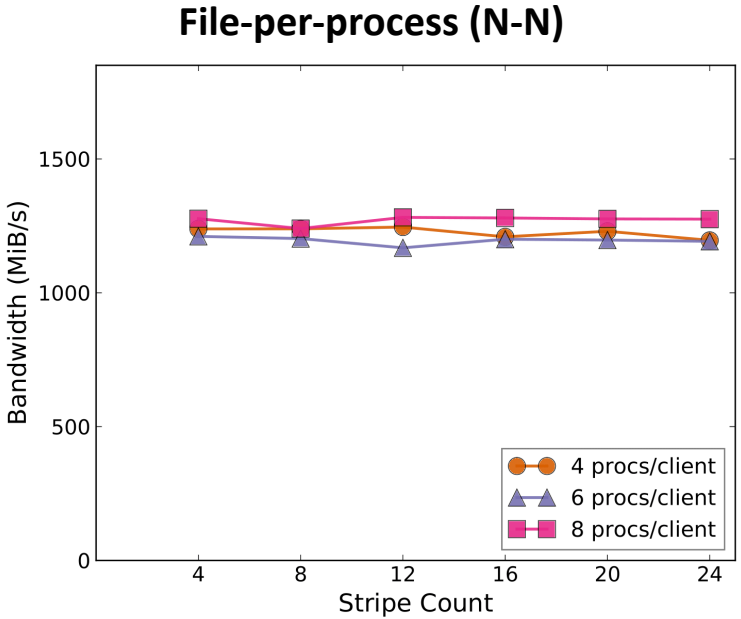
Disclaimer
This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

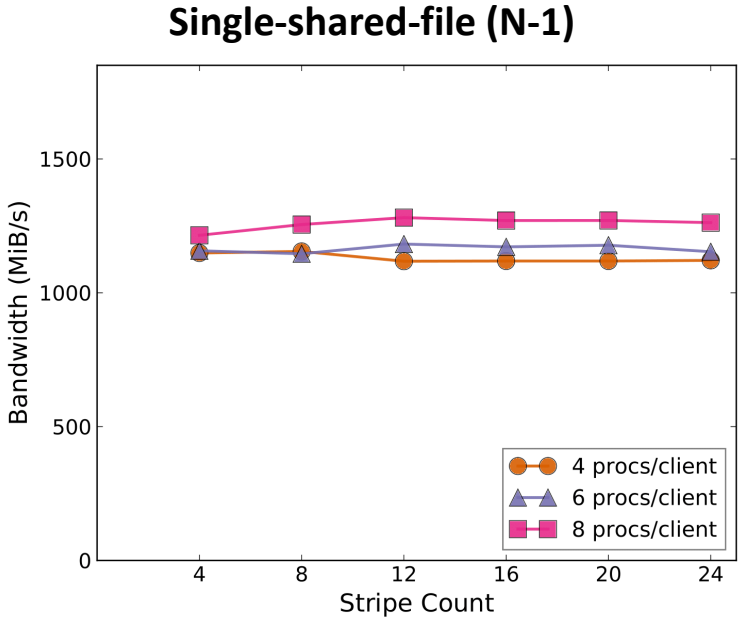
Thank you

Questions?

Backup

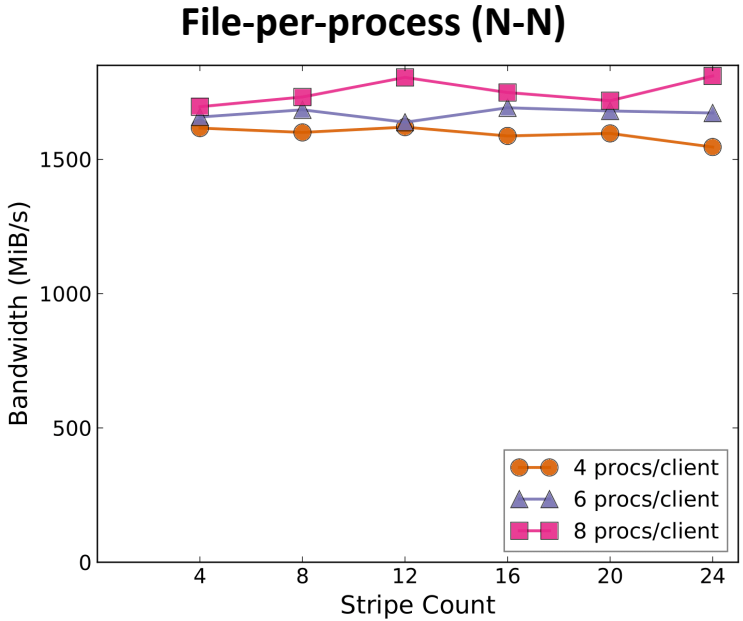
Single Client Read Performance (Contd.)

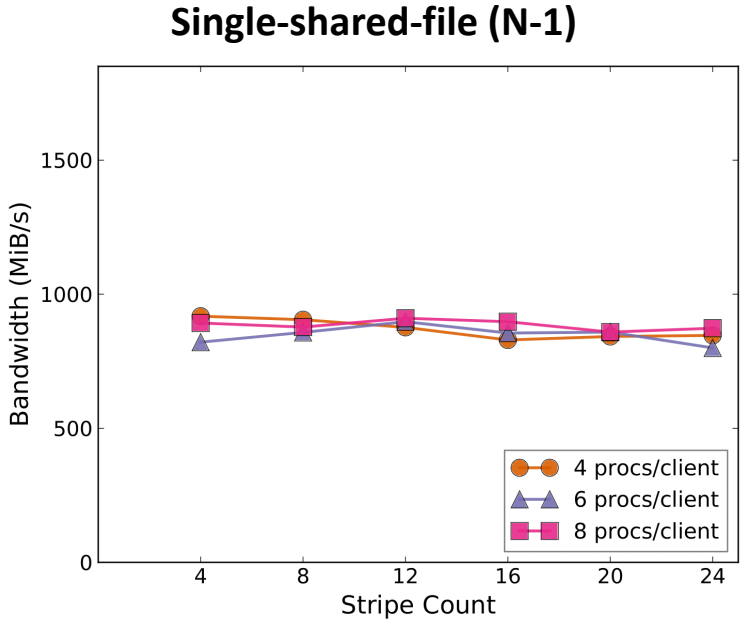




- N-N and N-1 read behave similarly with increasing stripes
- Lower stripe count ensures cost-efficient resource utilization

Single Client Write Performance (Contd.)





- N-N and N-1 write bandwidth both do not change much with stripe count
- Lower stripe count ensures more available storage devices