DYAD: Accelerating Deep Learning Training via inter-node access coordination over node-local storages

Hariharan Devarajan[1], Ian Lumsden[2], Chen Wang[1], Konstantia Georgouli[1], Tom Scogland[1], Jae-Seung Yeom[1], and Michela Taufer[2]

Overview

- Deep Learning (DL) impacts many complex scientific problems.
- Training DL for consume terabyte scale datasets every epoch.
- The data access behavior during DL training exposes unique optimization opportunities for caching systems.
- Current caching solutions employ near-compute storage accelerators primarily as exclusive caches.
  - Limiting the effectiveness of cache access locality.

DYAD *is a system designed to maximize sample locality in the cache, thereby increasing I/O throughput in HPC systems.*

- DYAD boosts inter-node access speeds by using a novel streaming RPC with RDMA protocol, achieving a 2.5x performance gain over state-of-the-art solutions.
- DYAD further enhances inter-node access by coordinating data movement, which mitigates network congestion and increases throughput for inter-node accesses by up to 8.78x.
- DYAD uses smart metadata caching that outperforms traditional global metadata access methods by 55x.

DYAD accelerates large-scale DL training on Corona with 512 GPUs by up to 10.82x faster epochs compared to UnifyFS by performing locality-aware caching on NVMe.

Contrasting requirements of Deep Learning Workloads

Execution Runtime
- Data is read using asynchronous workers using a task-based runtime.

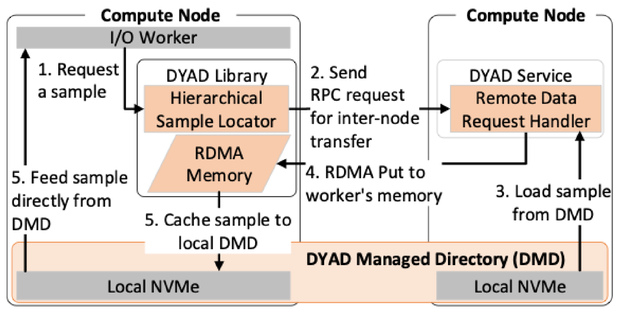
Dataset Reuse
- DL training iterates over the entire dataset multiple times.

Data parallel training
- DP training shards datasets across multiple GPUs and reduces the results.

Data Access Pattern
- Random sampling is a norm to increase model generality which randomizes sample reading.

File-Sample Distribution
- Different scientific domains have different distribution of samples within file system.

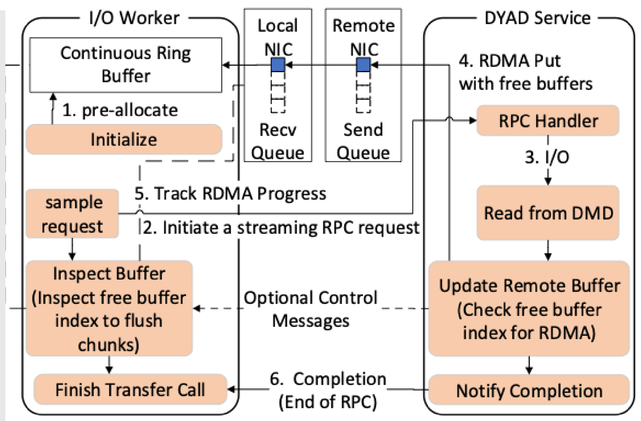
DYAD High-Level Design



- The first epoch in DL training caches data into DMD.
- Future epochs, load data from local or remote DMD and cache it locally.
- Eventually all accesses become local.

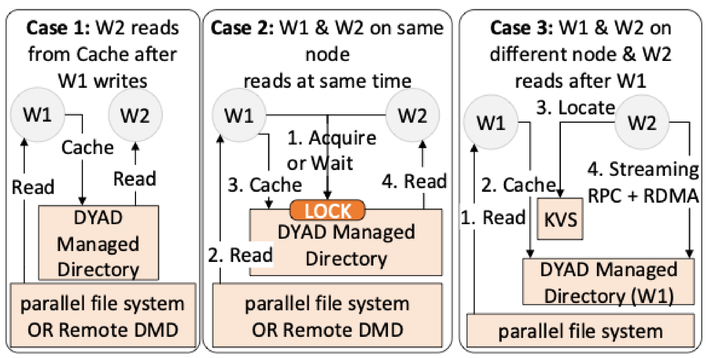
Novel Data Movement Protocol (Streaming RPC over RDMA)

- DYAD pre-allocates and page aligns memory buffers used for transferring sample.
- Control plane message are passed through streaming RPC.
- Data plane messages uses RDMA protocol.
- A ring buffer is utilized to manage dynamically sized data.



- Streaming RPC Protocol allows light weight request with multiple responses to efficiently manage bulk RDMA calls.

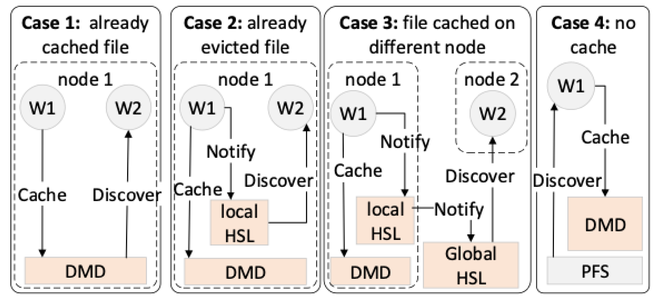
Data Movement Coordination



- Samples from Local DMD do not require any explicit coordination from DYAD.
- Samples from remote DMD are served using Streaming RPC over RDMA protocol.

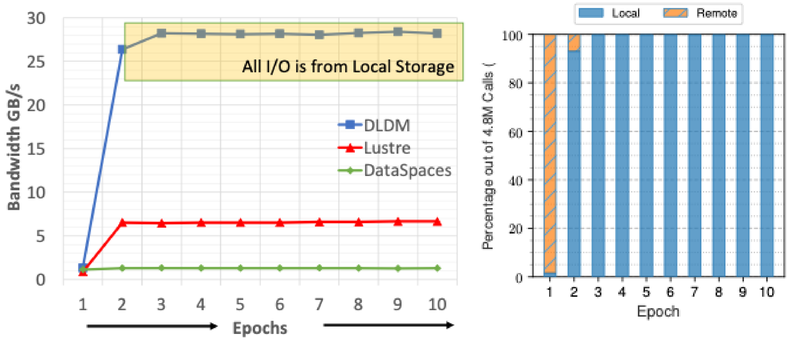
- Request for same files across multiple samples are coordinated by using fs locks.
- Levels of sample access from DMD allows DYAD to optimize data movement by 8.78x.

Hierarchical Sample Locator



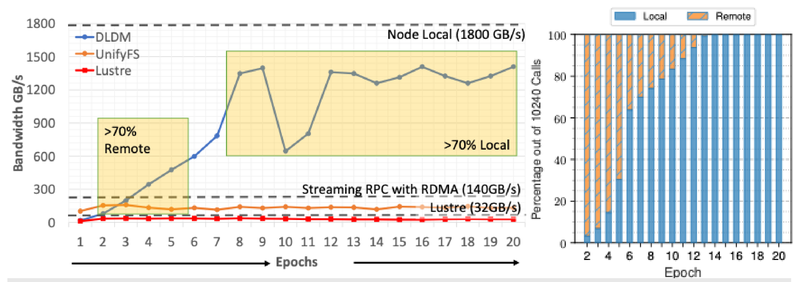
Hierarchical sample locator allows for multiple caching opportunities for sample reuse across epochs. This improves performance by up to 55x.

Accelerating MuMMI Training



MuMMI DL training reads the simulation dataset using PyTorch framework. The dataset is cached in first epoch and then reused in future epochs.

Accelerating Unet3D Training



Unet3D has many files in its dataset. Training across multiple epochs increases the % of local accesses and thus optimizes I/O bandwidth.

Conclusions

1. DYAD introduces a novel streaming RPC over RDMA protocol to accelerate data movement in DL workloads by 2.5x
2. DYAD increases locality of sample for DL training optimizing MuMMI and Unet3D by up to 10.82x on Corona.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by C-CSSE Nextgen NGCE and the LLNL-LDRD Program under Project No. 24-SI-005 and 23-ERD-045. This research used resources from the Oak Ridge Leadership Computing Facility (OLCF), a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. For computing time, the authors thank the Advanced Scientific Computing Research Leadership Computing Challenge (ALCC) for time on Summit and the Livermore Institutional Grand Challenge for time on Lassen. LLNL-CONF-862383. IL and MT acknowledge the support of NSF #2138811 and #2331152.