Poster ID: **P61916** | Topic / Category: **AI Models & Deployment - Deep Learning Frameworks**

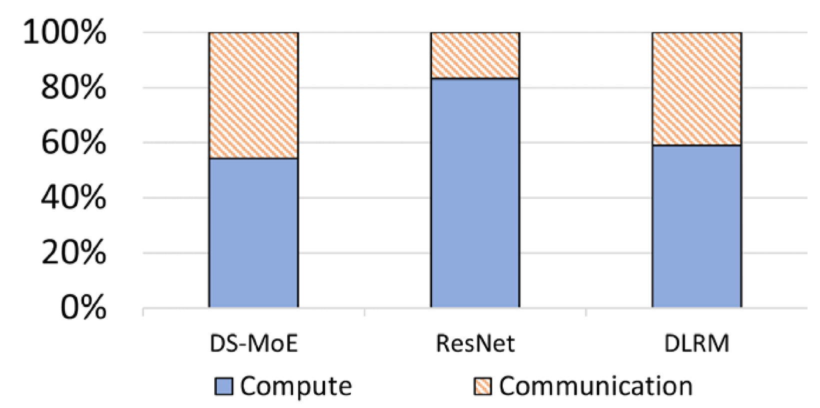
Quentin Anthony: anthony.301@osu.edu | **Dhabaleswar K (DK) Panda:** panda@cse.ohio-state.edu

MCR-DL: Mix-and-Match Communication Runtime for Deep Learning

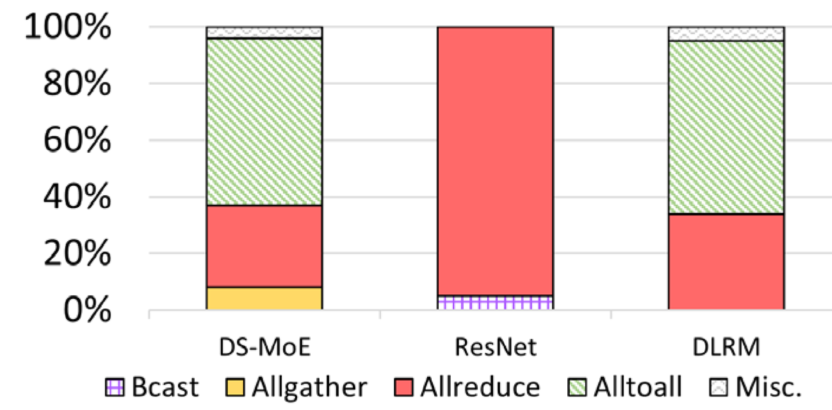
Quentin G. Anthony, Ammar Awan, Jeff Rasley, Yuxiong He, Aamir Shafi, Mustafa Abduljabbar, Hari Subramoni and Dhabaleswar K. Panda
{anthony.301, shafi.16, abduljabbar.1, subramoni.1, panda.2}@osu.edu, {ammar.awan, jeff.rasley, yuxhe}@microsoft.com

MOTIVATION

- Deep Learning applications require significantly more communication than previous data-parallel applications
- Deep Learning workloads are no longer purely using AllReduce and Broadcast communication operations
  - **Tensor Parallelism:** AllReduce/AllGather
  - **Pipeline Parallelism:** Send/Recv
  - **Data Parallelism:** AllReduce/Broadcast
  - **Pipeline Parallelism:** Point-to-Point
  - **ZeRO:** AllGather/ReduceScatter
  - **MoE:** AlltoAll

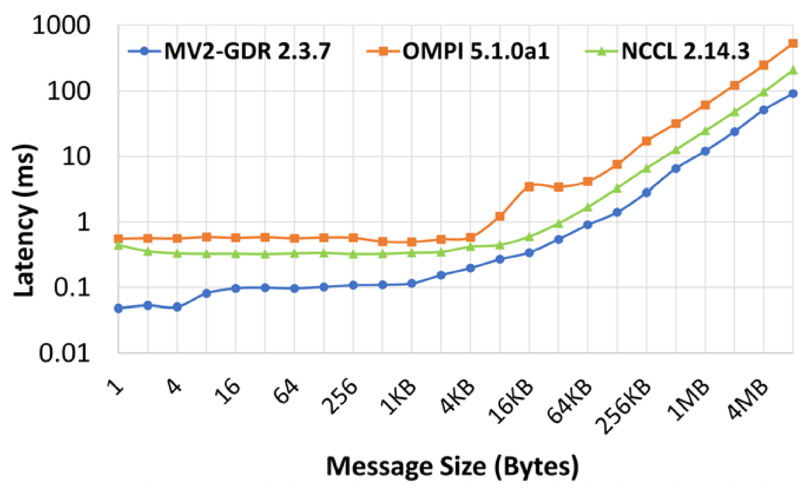


Profile of compute vs communication for each DL application

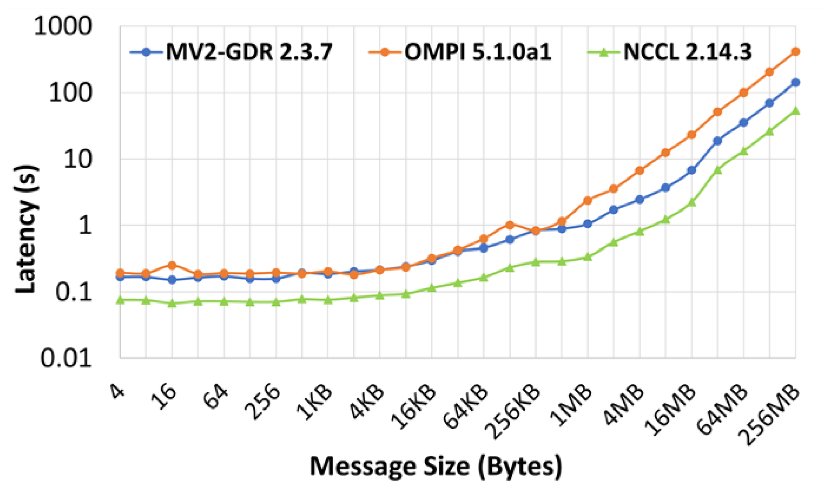


Profile of communication operation breakdown for each DL application

- Given that we need a mixture of communication operations, which communication library should we use?



AlltoAll performance on OSU Micro-Benchmark



iAllReduce performance on OSU Micro-Benchmark

- Best AlltoAll is MVAPICH2-GDR, and best iAllReduce is NCCL

CHALLENGES

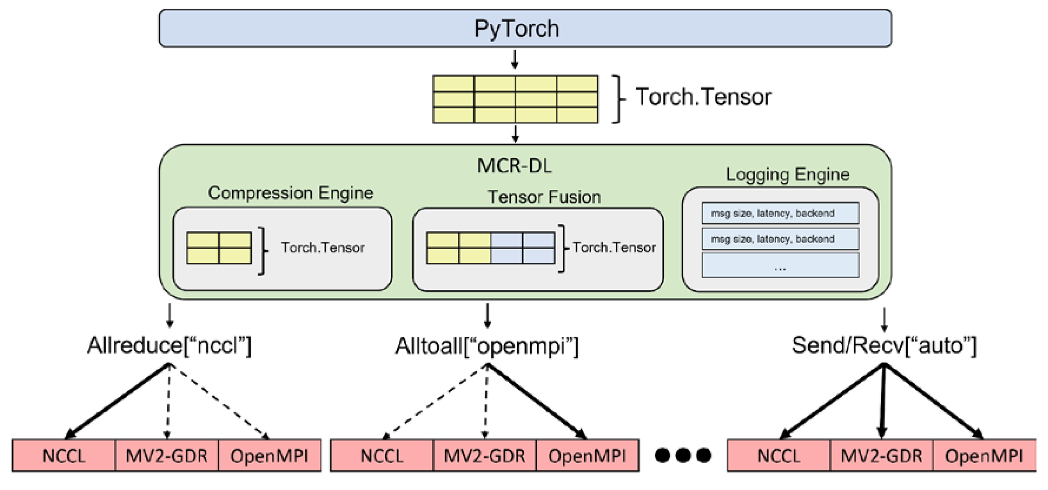
How to use the best communication library for a given operation?

- How do we avoid deadlocks when mixing communication backends?
- Can we choose the best library dynamically based on the message size?
- How do we define a unified interface for all communication calls?

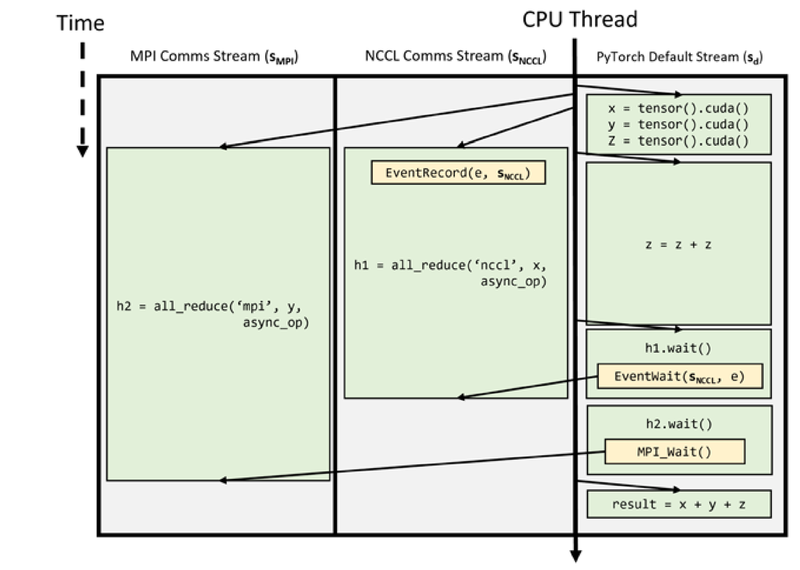
DESIGN – MCR-DL

We define a Mix-and-Match Communication Runtime for Deep Learning (MCR-DL)

- Unified interface for all communication calls based on PyTorch distributed module's API
- Backend-aware communication synchronization to avoid deadlocks
  - MPI_Wait() for MPI, CUDA events for NCCL/SCCL
- Easy to add communication features like compression, tensor fusion, and logging.



MCR-DL: A unified communication interface



Synchronization in MCR-DL

PERFORMANCE - Applications

Performance of DL applications using proposed designs on the **Lassen@LLNL** (V100 GPUs) and **ThetaGPU@ALCF** (A100 GPUs) systems

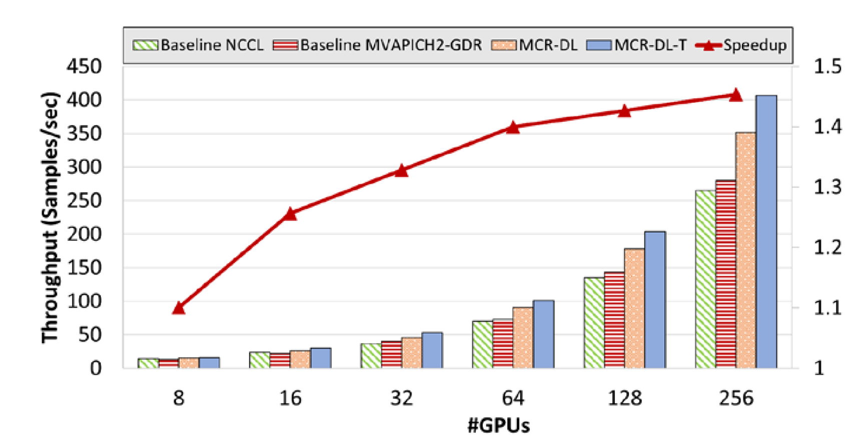
- **Baseline:** Single communication library for all operations
- **MCR-DL:** Coarse library mixing (NCCL AllReduce, MVAPICH2-GDR AlltoAll)
- **MCR-DL-T:** "Tuned" library mixing using a static table mapping message sizes to communication backend (see below table for details)

MCR-DL supports creating **tuning tables** (see right)

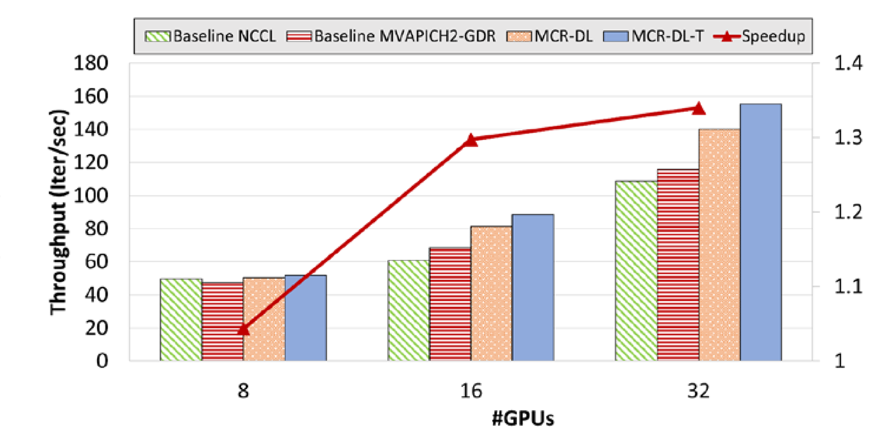
- Define table before DL applications are run
- Map messages to libraries based on message size
- One table per #GPUs and communication operation

Message Size	Backend
256	MVAPICH2-GDR
512	MVAPICH2-GDR
1024	MVAPICH2-GDR
2048	MVAPICH2-GDR
4096	NCCL
8192	NCCL
16384	SCCL
32768	SCCL

Tuning table for AllGather on 16 GPUs



MCR-DL throughput for DS-MoE on V100s

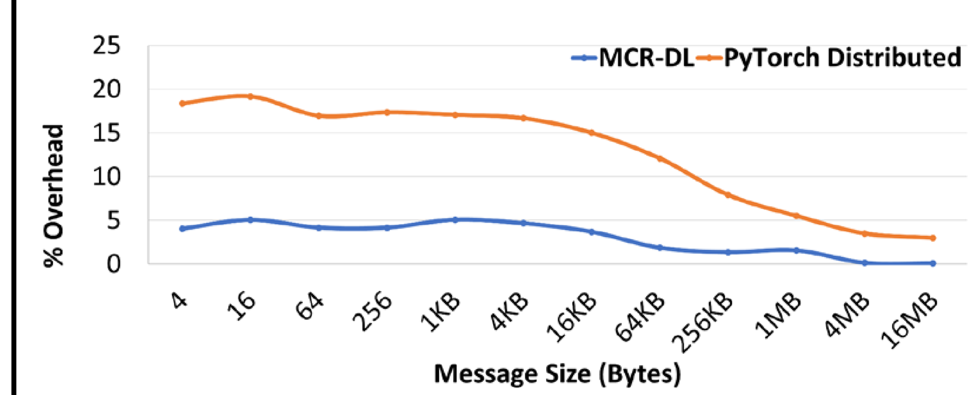


MCR-DL throughput for DLRM on A100s

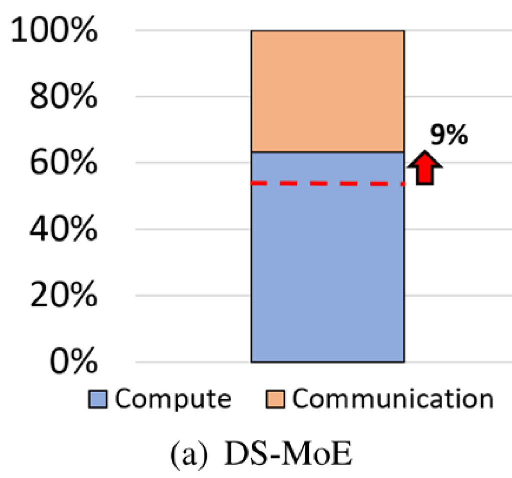
- **1.45x** throughput on 256 Lassen V100 GPUs using DS-MoE
- **1.33x** throughput on 32 ThetaGPU A100 GPUs using DLRM

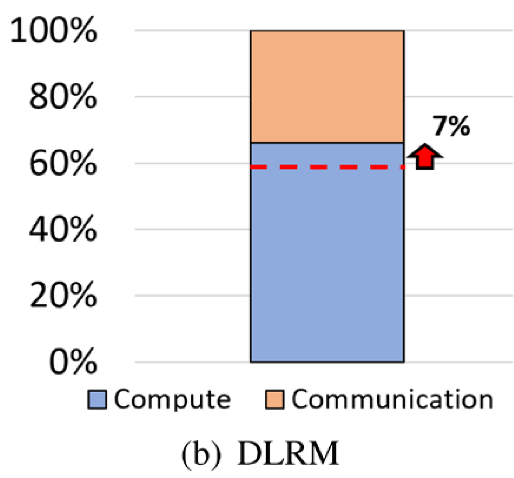
PERFORMANCE – Secondary Results

- MCR-DL's reduced Python logic for communication operations improves small-message overhead (see below)
- MCR-DL reduces the proportion of time spent in tensor communication (see above)



Overhead of MCR-DL (Python) over OMB (C)





Communication overhead reduction with MCR-DL at 256 Lassen V100 GPUs (DS-MoE) and 32 ThetaGPU A100 GPUs (DLRM)

CONTRIBUTIONS

- Comprehensive profiling and evaluation of modern deep learning applications
- Introduce MCR-DL, a unified communication interface that supports mixing libraries so that **the best library is used for a given communication operation**
- Evaluate proposed designs compared to existing libraries using **DeepSpeed Mixture-of-Experts (DS-MoE) on up to 256 V100 GPUs, and Deep Learning Recommendation Models (DLRM) on up to 32 A100 GPUs**