Follow us on

https://twitter.com/mvapich

Scaling Single-Image Super-Resolution Training on Modern HPC Clusters: Early Experiences

Presentation at MUG '21

Quentin Anthony, Lang Xu,
Hari Subramoni, and Dhabaleswar K. Panda

Network Based Computing Laboratory (NBCL)
Dept. of Computer Science and Engineering , The Ohio State University
{anthony.301, xu.3304, subramoni.1, panda.2}@osu.edu

Q. Anthony, L. Xu, H. Subramoni, and D. K. Panda, Scaling Single-Image Super-Resolution Training on Modern HPC Clusters: Early Experiences, ScaDL Workshop, in conjunction with IPDPS '21, May 2021

Agenda

- **Introduction**
- Background
- Research Challenges
- Performance Evaluation
  - Evaluation Platforms and Software Libraries
  - Scaling Results
- Conclusion

Deep Learning Frameworks

- Easily implement and experiment with Deep Neural Networks
  – Several Deep Learning (DL) frameworks have emerged

- PyTorch, TensorFlow, and MXNet are the major DL frameworks
  – *Focus on **PyTorch** in this work*

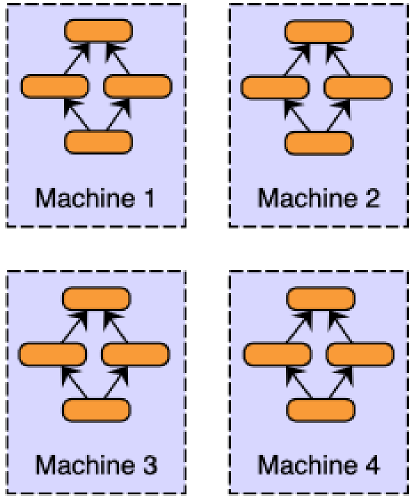
- Most frameworks are optimized for NVIDIA GPUs (for now!)

Agenda

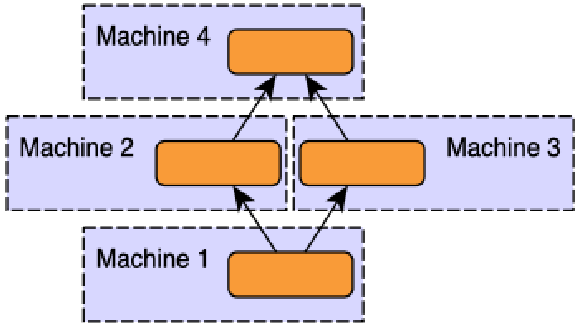
- Introduction
- **Background**
- Research Challenges
- Performance Evaluation
  - Evaluation Platforms and Software Libraries
  - Scaling Results
- Conclusion

Background: Distributed DNN Training

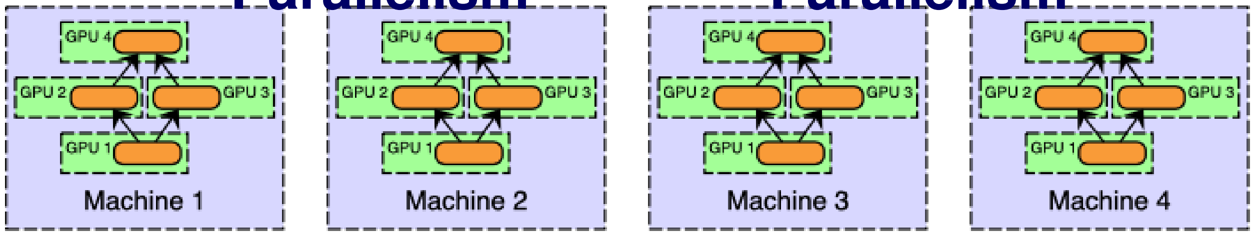
- Deep Neural Network training consists of two phases
  - Forward pass
  - Backward pass
- Training is a compute intensive task
  - Large datasets
  - Complex Deep Learning Models
  - MPI-driven training is on the rise
- Three approaches to Distribute DNN
  - **Data Parallelism (focus of this paper)**
  - Model Parallelism
  - Hybrid Parallelism



Data Parallelism



Model Parallelism



Hybrid Parallelism

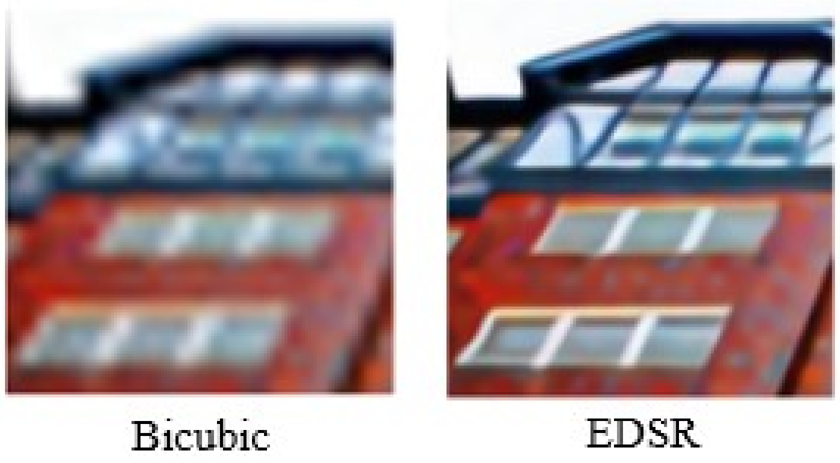
Courtesy: https://blog.skymind.ai/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks/

Background: MVAPICH2-GDR Existing Enhancements

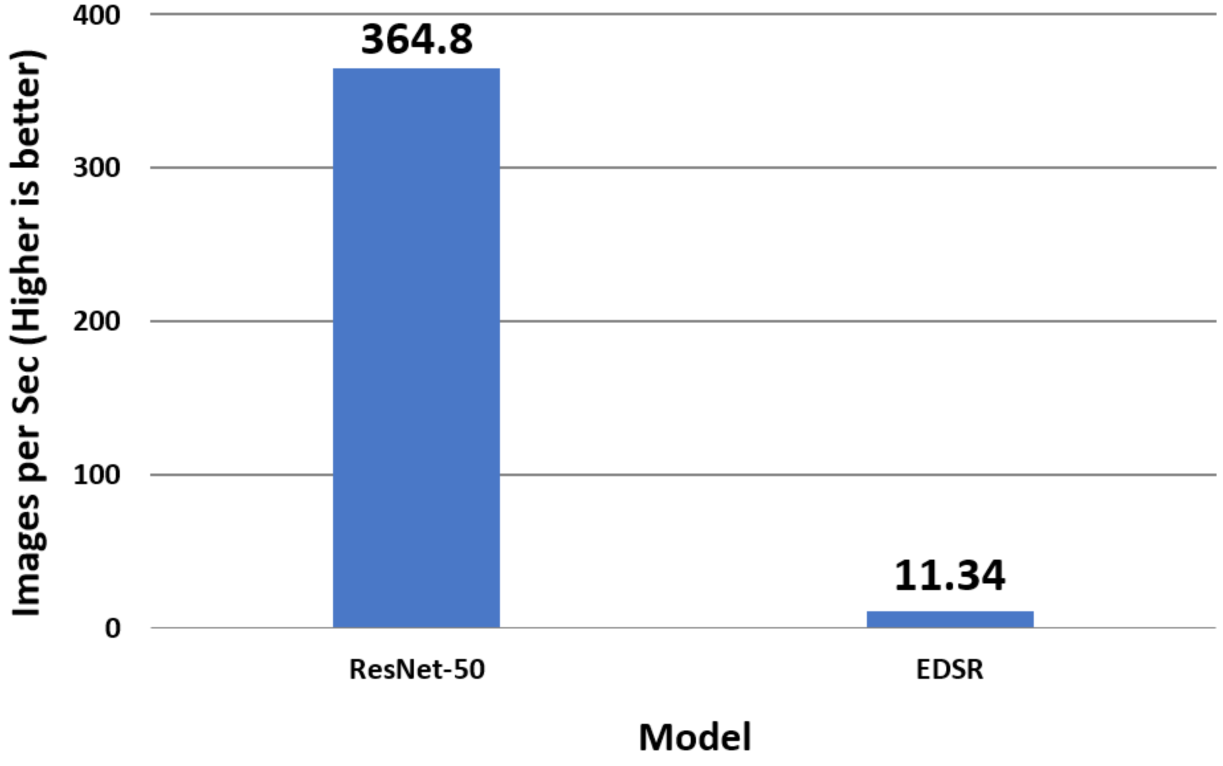
- CUDA IPC designs
  - Interface for communicating across GPUs within a node (since CUDA 4.1)
  - Many advanced collective and pt2pt designs in MVAPICH2-GDR use CUDA IPC
- Registration cache
  - Enables the zero-copy transfer of large messages across InfiniBand without requiring memory registration **on cache hits**
  - Previous MVAPICH2-GDR versions disabled registration cache for DL workloads due to conflict with TensorFlow's custom memory allocators

Background: Image Super-Resolution and EDSR

- Many emerging scientific and medical applications require high-resolution input
  - Examples: Computational pathology, satellite imagery, and climate analysis
- A super resolution (SR) model maps a low-resolution (LR) image to a high-resolution (HR) image
- Deep learning super resolution (DLSR) trains a DNN to learn the mapping LR --> HR
  - Datasets (i.e. DIV2K) consist of high-resolution (2K) imag counterparts)
  - Super-resolution models are much more compute-intens
- We chose EDSR as our representative DLSR moc
  - SOTA accuracy
  - Open-source implementation



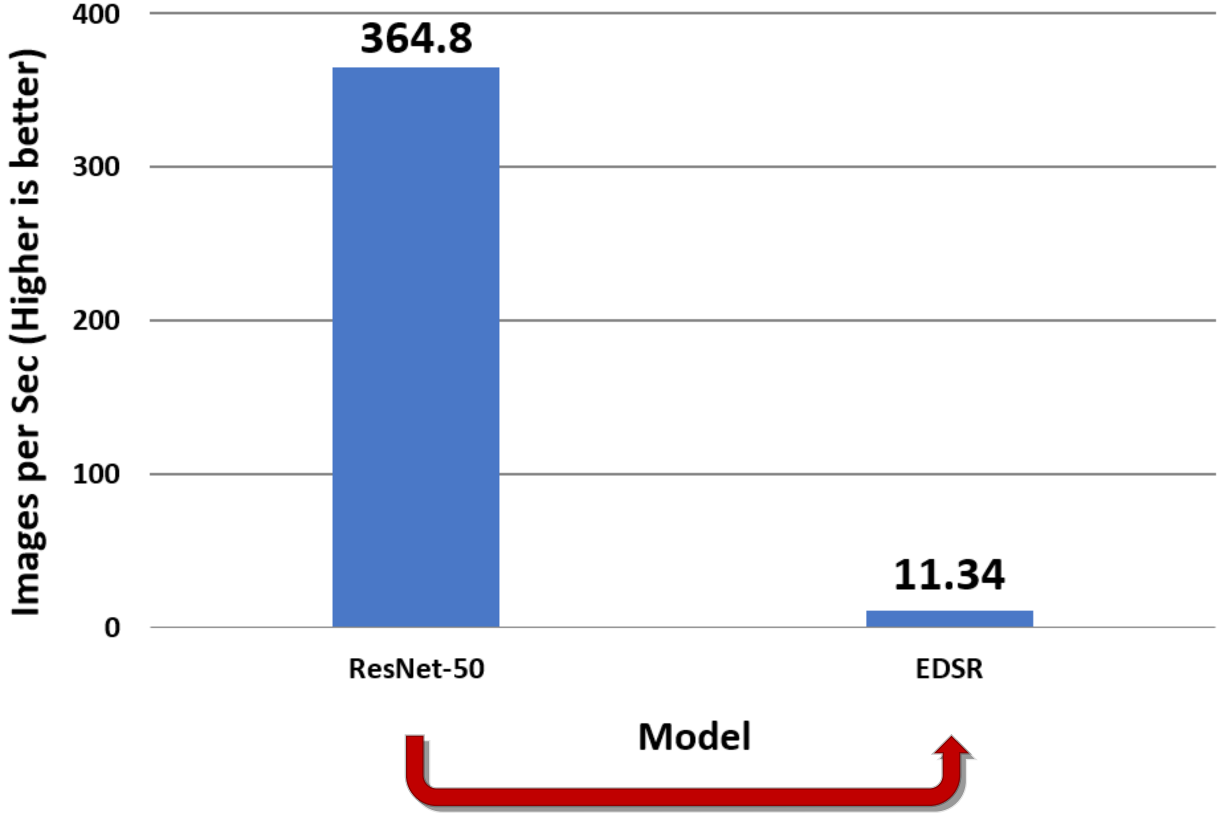
Super-resolution vs. Classification



- We compare Resnet-50 (Classification) with EDSR [1] (DLSR) on synthetic benchmark with single V100 GPU

[1] Bee Lim et al. “Enhanced Deep Residual Networks for Single Image Super-Resolution”

Super-resolution vs. Classification



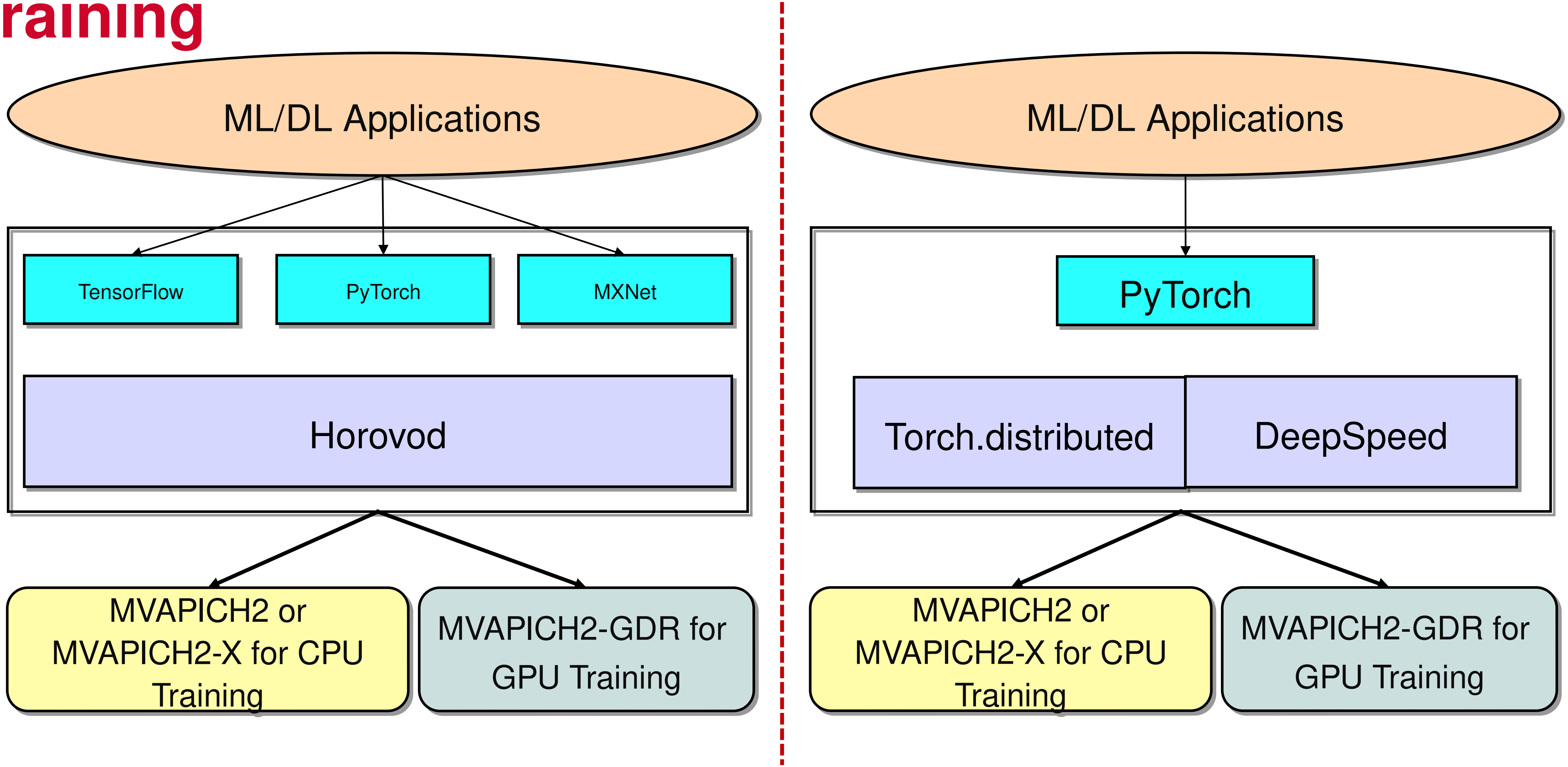
- We compare Resnet-50 (Classification) with EDSR [1] (DLSR) on synthetic benchmark with single V100 GPU
- **Super-resolution DNNs require more computation per image**
- Super-resolution DNNs could benefit greatly from HPC!

[1] Bee Lim et al. "Enhanced Deep Residual Networks for Single Image Super-Resolution"

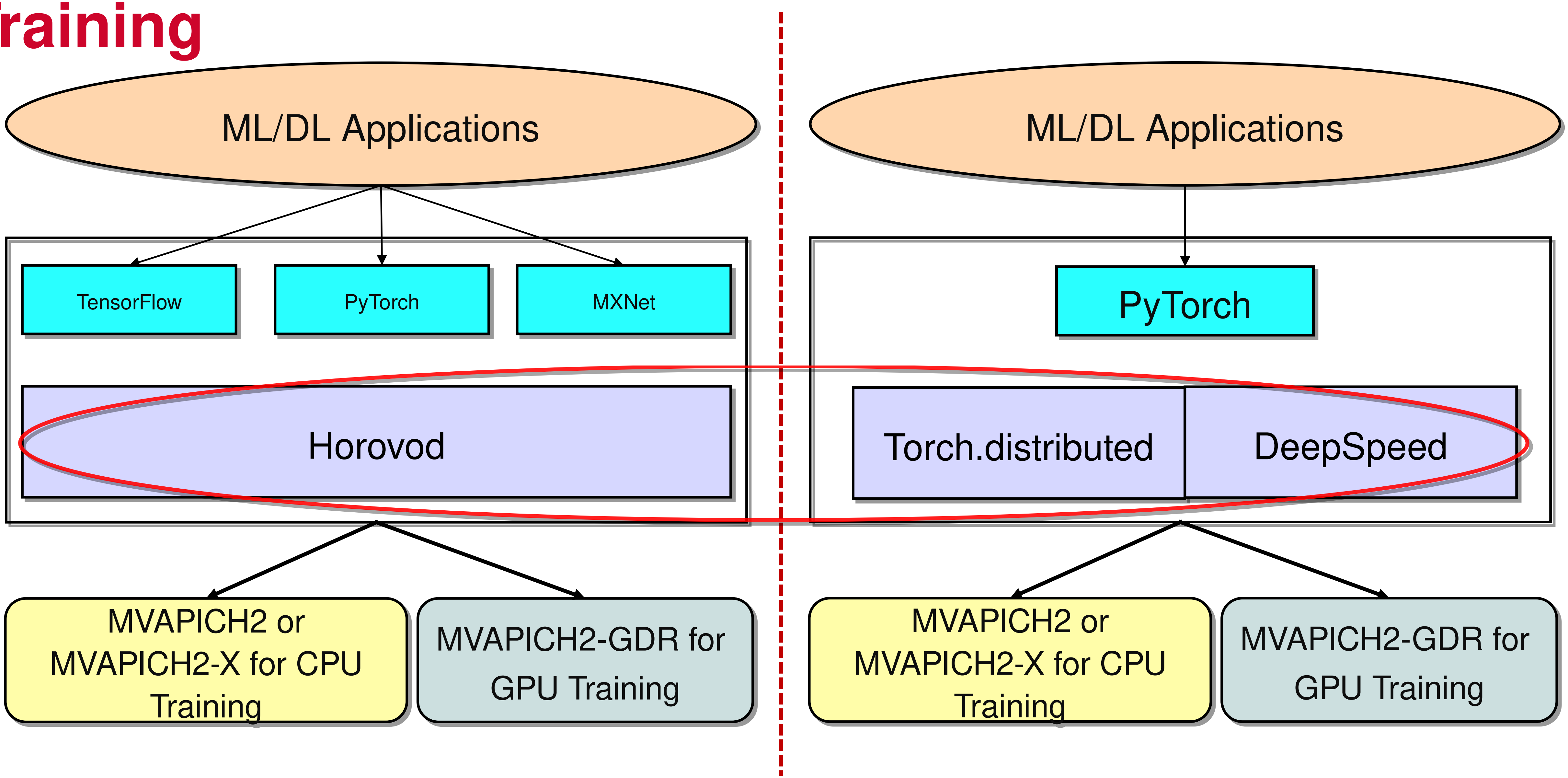
Horovod

- Horovod is a distributed DNN training framework that employs data parallelism
  - **Acts as middleware** between DL framework (Tensorflow, Pytorch, etc) and communication backend (MPI, NCCL, etc)
  - Performance is strongly dependent on **Allreduce**
- Before carrying out evaluations, we have added full distributed training support to EDSR via Horovod

MVAPICH2 (MPI)-driven Infrastructure for ML/DL Training



MVAPICH2 (MPI)-driven Infrastructure for ML/DL Training

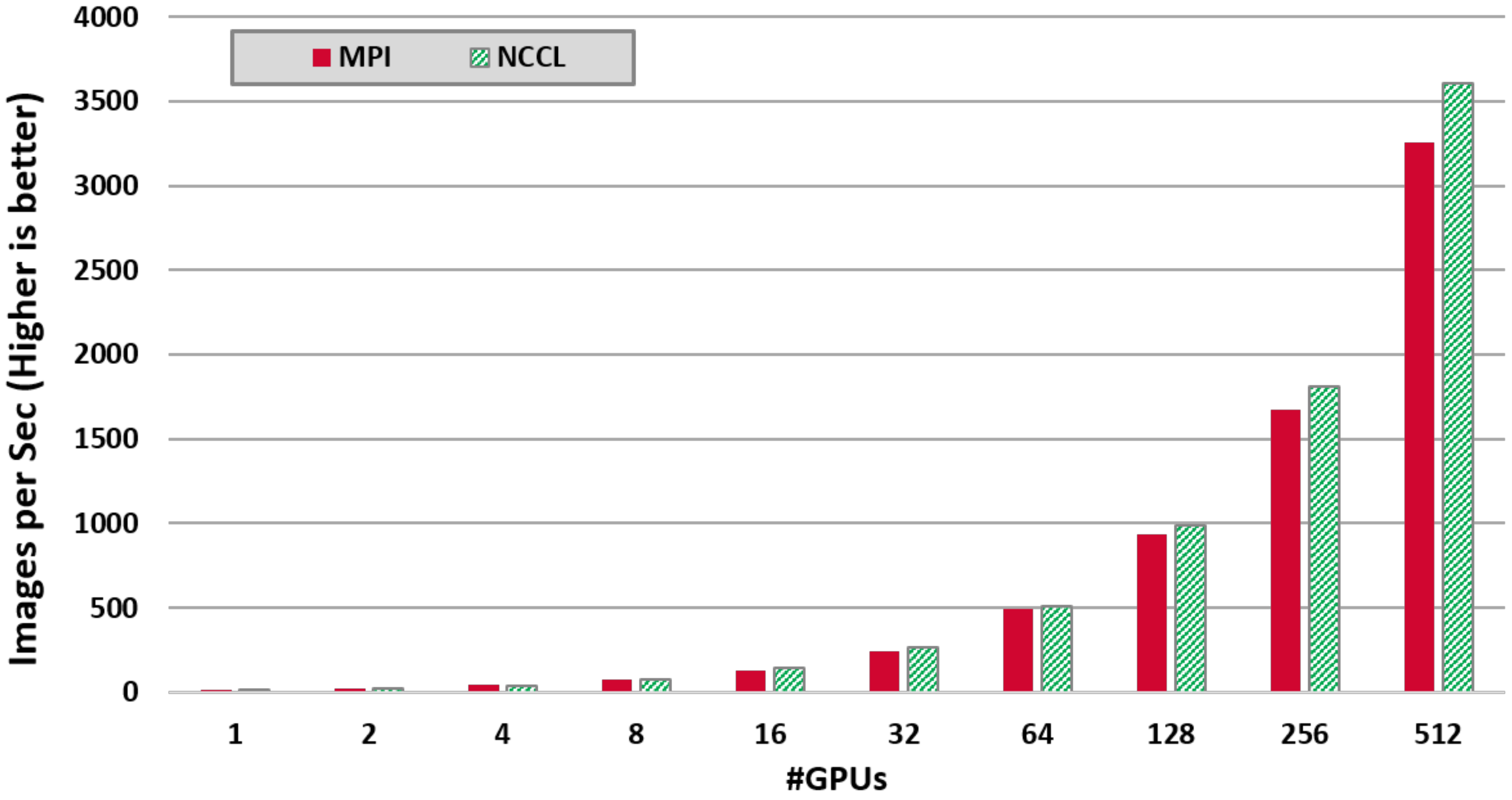


Agenda

- Introduction
- Background
- **Research Challenges**
- Performance Evaluation
  - Evaluation Platforms and Software Libraries
  - Scaling Results
- Conclusion

Scaling Shortcomings

- Default Horovod performance for EDSR is inefficient
- Only **≈ 55-65%** scaling efficiency for both MVAPICH2-GDR and NCCL backends at scale



MVAPICH2 Enhancements

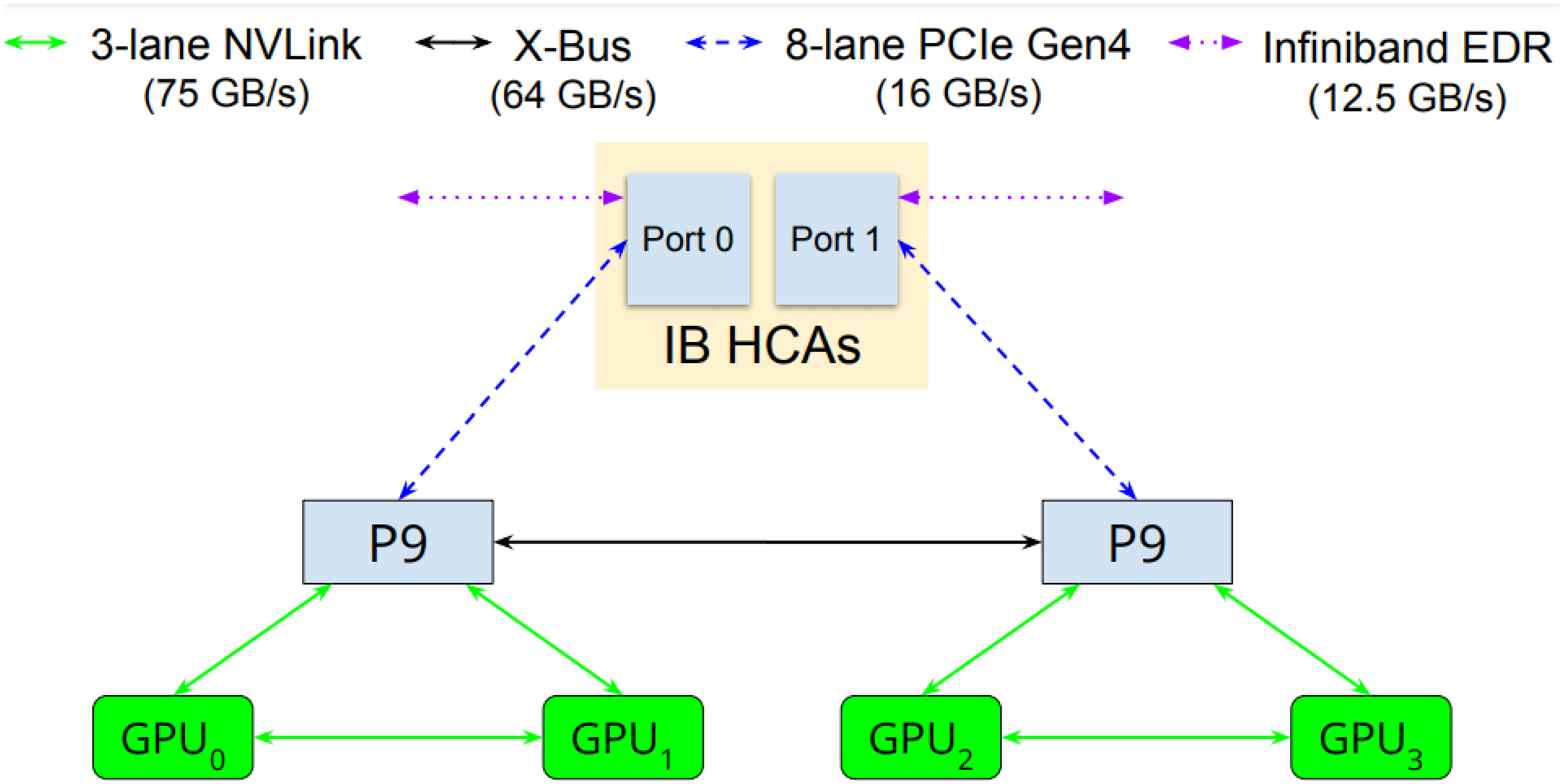
- CUDA IPC
  - Since PyTorch allocates all available GPUs, CUDA IPC must be disabled for most distributed PyTorch training runs.
  - The method recommended by PyTorch is to set CUDA_VISIBLE_DEVICES=${LOCAL_RANK} (i.e. set each GPU's visible device list to itself)
  - **Primary Enhancement:** Add an intermediate visible device list (i.e. MV2_VISIBLE_DEVICES) that allows MVAPICH2-GDR to view GPUs while hiding them from PyTorch

- Registration cache
  - **Secondary Enhancement:** Evaluate the effect of the registration cache on DL training performance

Agenda

- Introduction
- Background
- Research Challenges
- **Performance Evaluation**
  - **Evaluation Platforms and Software Libraries**
  - Scaling Results
- Conclusion

Evaluation Platform



Courtesy: Performance Evaluation of MPI Libraries on GPU-enabled OpenPOWER Architectures: Early Experiences, IWOPH '19

Software Libraries

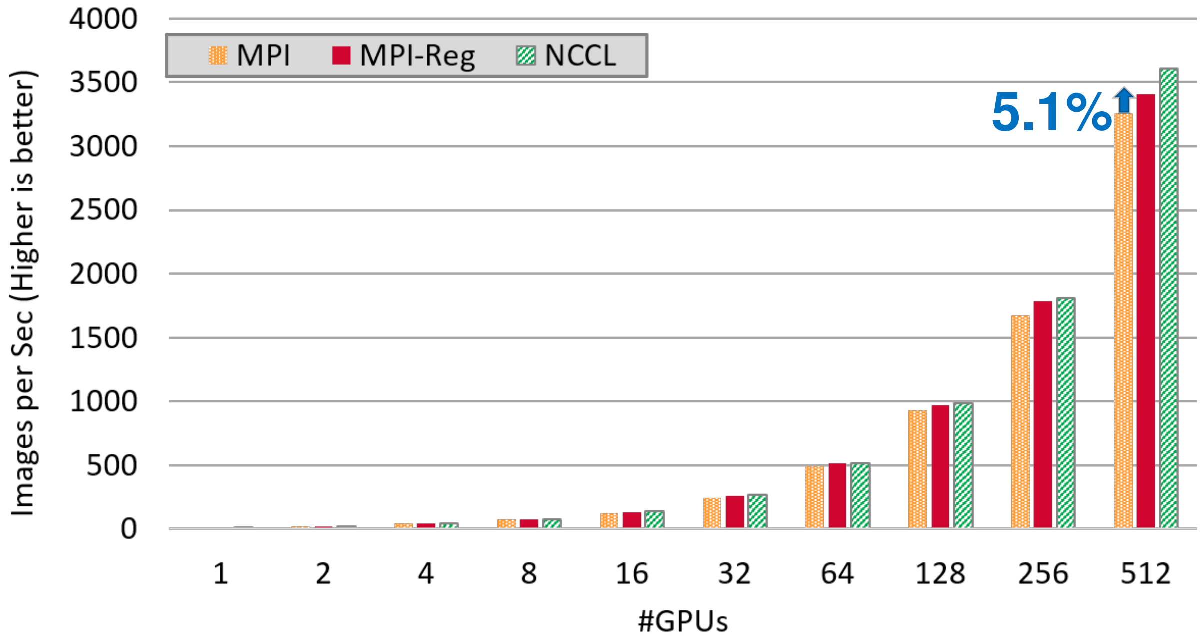
- Deep Learning Frameworks
  - PyTorch v1.8.0
- CUDA 10.2
- cuDNN 7.6.5
- Horovod Distributed Training middleware (0.19.1)
- MPI Library: MVAPICH2-GDR 2.3.5
- DL Model: EDSR from publicly-available github
  - https://github.com/sanghyun-son/EDSR-PyTorch

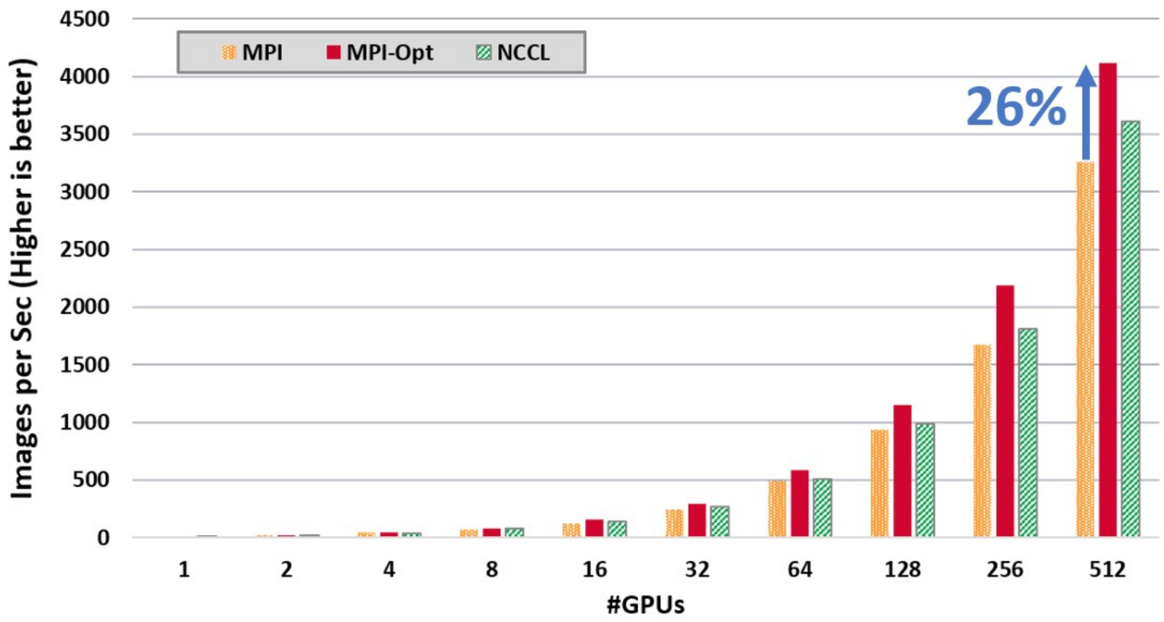
Agenda

- Introduction
- Background
- Research Challenges
- Methodologies
- Performance Evaluation
  - Evaluation Platforms and Software Libraries
  - Scaling Results
- Conclusion

Performance Improvement: Throughput

- **MPI**: (No CUDA IPC, No Registration Cache)
- **MPI-Reg**: (No CUDA IPC, Registration Cache)
- **MPI-Opt**: (CUDA IPC, Registration Cache)
- We evaluate:
  - MPI with the registration cache enabled (average of **5.1%** improvement and a **93%** hit rate)
  - MPI with the CUDA IPC advanced designs (up to **26%** improvement)





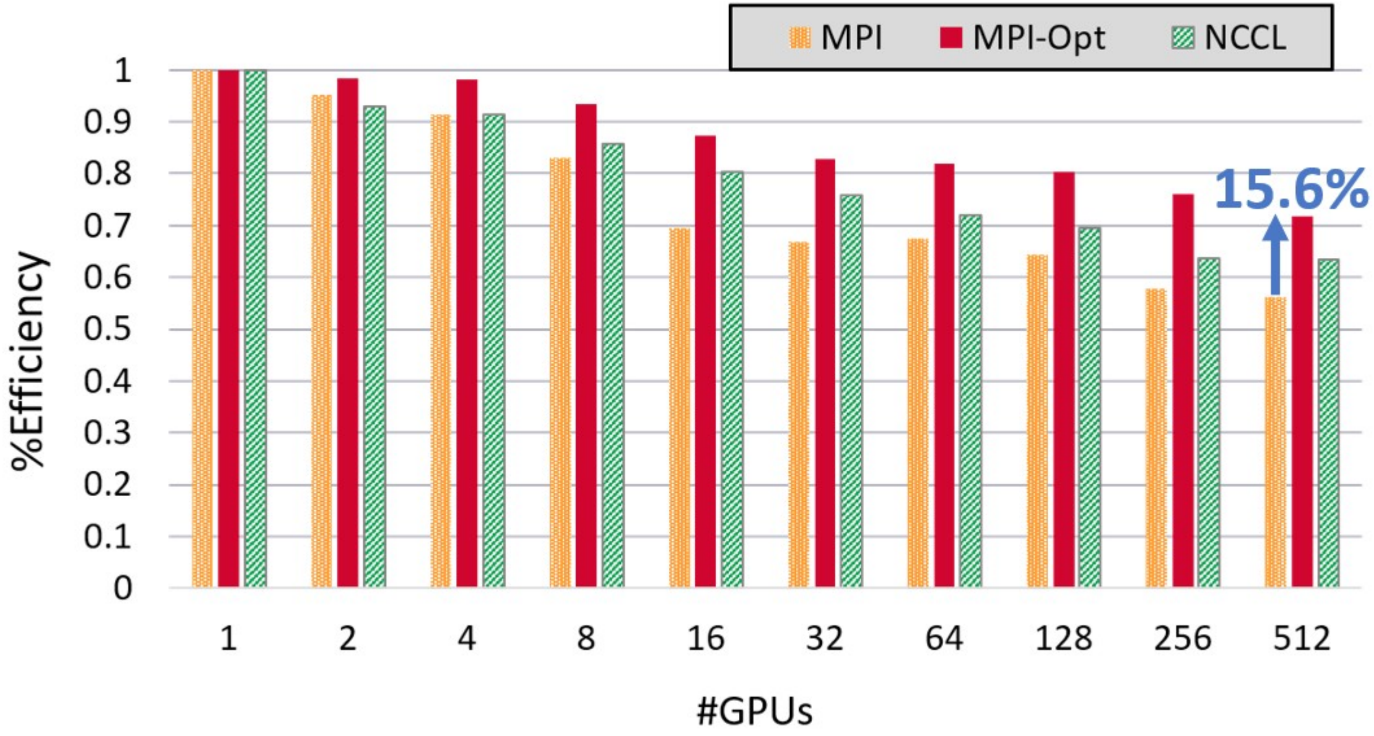
Performance Improvement: Throughput

- Throughput is improved at all scales
- MPI-Opt performs better than **both** NCCL and default MPI

#GPUs	Throughput (Img/sec)			Percentage Improvement (MPI-Opt over default MPI)
	NCCL	MPI	MPI-Opt	
1	11	11	11	≈ 0
2	20	21	22	2.08
4	40	41	44	6.28
8	76	75	83	11.15
16	142	126	156	24.26
32	269	242	296	22.32
64	511	490	588	20.05
128	989	933	1152	23.49
256	1807	1675	2185	30.41
512	3608	3258	4116	**26.33**

Performance Improvement: Scaling Efficiency

- We compare scaling efficiency of EDSR between the optimized and default Horovod runs
- Demonstrated a **15.6%** improvement in scaling efficiency



Profiling Results

- We applied our internal profiling tool *hvprof* [1], to profile Allreduce before and after enabling the registration cache and CUDA IPC designs
- Note that most messages are near 64MB, because Horovod batches tensor into a "fusion buffer" of default size 64MB
- Latency is significantly improved for large message sizes

[1] A. A. Awan, et al."Communication Profiling and Characterization of Deep-Learning Workloads on Clusters With High-Performance Interconnects,"

Profiling Results

- Latency is significantly improved for large message sizes
  - CUDA IPC and the registration cache is only enabled in MVAPICH2-GDR for **large messages**
- It's critical that users first profile the DL workload before applying MPI designs

Message Size (Bytes)	Time (ms)		Percentage Improvement
	Default	Optimized	
1-128 KB	392	391	≈ 0
128 KB - 16 MB	320	342	≈ 0
16 MB - 32 MB	1321	619	53.1
32 MB - 64 MB	5145	2587	49.7
Total Time	7179	3918	**45.4**

Profiling Results

- Latency is significantly improved for large message sizes
  - CUDA IPC and the registration cache is only enabled in MVAPICH2-GDR for **large messages**
- It's critical that users first profile the DL workload before applying MPI designs
- Up to **45% improvement** in total Allreduce time after applying MVAPICH2-GDR designs

Message Size (Bytes)	Time (ms)		Percentage Improvement
	Default	Optimized	
1-128 KB	392	391	≈ 0
128 KB - 16 MB	320	342	≈ 0
16 MB - 32 MB	1321	619	53.1
32 MB - 64 MB	5145	2587	49.7
Total Time	7179	3918	**45.4**

Agenda

- Introduction
- Background
- Research Challenges
- Methodologies
- Performance Evaluation
  - Evaluation Platforms and Software Libraries
  - Scaling Results
- **Conclusion**

Conclusion

- Image super resolution is growing in popularity
  - Deep learning super resolution models (e.g. EDSR) require long training times
  - Novel DNNs bring about novel communication bottlenecks

- Detailed profiling of middleware is critical to identifying bottlenecks

- In order to effectively scale to hundreds or thousands of GPUs, we need to ensure DL frameworks are using advanced MPI features

Thank You!

Anthony.301@osu.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/