
Tutorial at GTC ’19

by

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~panda

Ammar Ahmad Awan
The Ohio State University
E-mail: awan.10@osu.edu
http://www.cse.ohio-state.edu/~awan.10

Hari Subramoni
The Ohio State University
E-mail: subramon@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~subramon
Outline

• Introduction
  – The Past, Present, and Future of Deep Learning
  – What are Deep Neural Networks?
  – Diverse Applications of Deep Learning
  – Deep Learning Frameworks

• Overview of Execution Environments

• Parallel and Distributed DNN Training

• Latest Trends in HPC Technologies

• Challenges in Exploiting HPC Technologies for Deep Learning

• Solutions and Case Studies

• Open Issues and Challenges

• Conclusion
Brief History of Deep Learning (DL)

Milestones in the Development of Neural Networks

1940 - S. McCulloch and W. Pitts: Electronic Brain
1957 - M. Minsky and S. Papert: Perceptron
1960 - F. Rosenblatt: ADALINE
1969 - M. Widrow and M. Hoff: XOR Problem
1969 - D. Rumelhart, G. Hinton, R. Williams: Multi-layered Perceptron (Backpropagation)
1995 - V. Vapnik, C. Cortes: SVM
2006 - G. Hinton, S. Ruslan: Deep Neural Network (Pretraining)

Golden Age - 1970
Dark Age (“AI Winter”) - 1980

Understanding the Deep Learning Resurgence

• Deep Learning is a sub-set of Machine Learning
  – But, it is perhaps the most radical and revolutionary subset
  – Automatic feature extraction vs. hand-crafted features

• Deep Learning
  – A renewed interest and a lot of hype!
  – Key success: Deep Neural Networks (DNNs)
  – Everything was there since the late 80s except the “computability of DNNs”

Courtesy: http://www.deeplearningbook.org/contents/intro.html
Deep Learning, Many-cores, and HPC

- NVIDIA GPUs are the main driving force for faster training of DL models
  - The ImageNet Challenge - (ILSVRC)
  - 90% of the ImageNet teams used GPUs in 2014*
  - Deep Neural Networks (DNNs) like AlexNet, GoogLeNet, and VGG are used
  - A natural fit for DL due to the throughput-oriented nature

- In the High Performance Computing (HPC) arena
  - 126/500 Top HPC systems use NVIDIA GPUs (Nov ’18)
  - CUDA-Aware Message Passing Interface (MPI)
  - NVIDIA Fermi, Kepler, and Pascal architecture
  - DGX-1 (Pascal) and DGX-2 (Volta)
    - Dedicated DL supercomputers

*https://blogs.nvidia.com/blog/2014/09/07/imagenet/

Performance Share
www.top500.org
Deep Learning Use Cases and Growth Trends

1.1 Artificial Intelligence Revenue, World Markets: 2016-2025

1.2 Artificial Intelligence Revenue, Top 10 Use Cases, World Markets: 2025

Outline

• Introduction
  – The Past, Present, and Future of Deep Learning
  – What are Deep Neural Networks?
    – Diverse Applications of Deep Learning
    – Deep Learning Frameworks

• Overview of Execution Environments
• Parallel and Distributed DNN Training
• Latest Trends in HPC Technologies
• Challenges in Exploiting HPC Technologies for Deep Learning
• Solutions and Case Studies
• Open Issues and Challenges
• Conclusion
So what is a Deep Neural Network?

- Example of a 3-layer Deep Neural Network (DNN) – (input layer is not counted)

Graphical/Mathematical Intuitions for DNNs

Drawing of a Biological Neuron

The Mathematical Model

Key Phases of Deep Learning

• Deep Learning has two major tasks
  1. Training of the Deep Neural Network
  2. Inference (or deployment) that uses a trained DNN

• DNN Training
  – Training is a compute/communication intensive process – can take days to weeks
  – Faster training is necessary!

• Faster training can be achieved by
  – Using Newer and Faster Hardware – But, there is a limit!
  – Can we use more GPUs or nodes?
    • The need for Parallel and Distributed Training
DNN Training and Inference

[Diagram showing the process of training and inference for deep neural networks]

Courtesy: [Link to the presentation]
To actually train a network, please visit: http://playground.tensorflow.org
Outline

• Introduction
  – The Past, Present, and Future of Deep Learning
  – What are Deep Neural Networks?
  – Diverse Applications of Deep Learning
  – Deep Learning Frameworks

• Overview of Execution Environments
• Parallel and Distributed DNN Training
• Latest Trends in HPC Technologies
• Challenges in Exploiting HPC Technologies for Deep Learning
• Solutions and Case Studies
• Open Issues and Challenges
• Conclusion
Caption Generation, Translation, Style Transfer, and many more..

Synthesized Image

 Courtesy: https://github.com/alexjc/neural-doodle

Courtesy: https://machinelearningmastery.com/inspirational-applications-deep-learning/

Courtesy: https://research.googleblog.com/2015/07/how-google-translate-squeezes-deep.html
Google Translate

Courtesy: https://www.theverge.com/2015/1/14/7544919/google-translate-update-real-time-signs-conversations
Self Driving Cars

Courtesy: http://www.teslarati.com/teslas-full-self-driving-capability-arrive-3-months-definitely-6-months-says-musk/
Outline

• **Introduction**
  – The Past, Present, and Future of Deep Learning
  – What are Deep Neural Networks?
  – Diverse Applications of Deep Learning
  – **Deep Learning Frameworks**

• Overview of Execution Environments
• Parallel and Distributed DNN Training
• Latest Trends in HPC Technologies
• Challenges in Exploiting HPC Technologies for Deep Learning
• Solutions and Case Studies
• Open Issues and Challenges
• Conclusion
**Why we need DL frameworks?**

- Deep Learning frameworks have emerged
  - hide most of the *nasty mathematics*
  - focus on the *design* of neural networks
- Distributed DL frameworks are being designed
  - We have saturated the peak potential of a single GPU/CPU/KNL
  - Parallel (multiple processing units in a single node) and/or Distributed (usually involves multiple nodes) frameworks are emerging
- Distributed frameworks are being developed along two directions
  - The HPC Eco-system: MPI-based Deep Learning
  - Enterprise Eco-system: BigData-based Deep Learning

Statement and its dataflow fragment. The data and computing vertexes with different colors reside on different processes.

*Courtesy: https://web.stanford.edu/~rezab/nips2014workshop/submits/minerva.pdf*
DL Frameworks and GitHub Statistics

- AI Index report offers very detailed trends about AI and ML
- It also provides interesting statistics about open source DL frameworks and related GitHub statistics

Courtesy: http://cdn.aiindex.org/2017-report.pdf
Are Define-by-run frameworks easier than Define-and-run?

- Define-and-run: TensorFlow, Caffe, Torch, Theano, and others
- Define-by-run
  - PyTorch and Chainer
  - TensorFlow 1.5 introduced Eager Execution (Define-by-run) mode

Google TensorFlow (Most Popular)

• The most widely used framework open-sourced by Google
• Replaced Google’s DistBelief\[^1\] framework
• Runs on almost all execution platforms available (CPU, GPU, TPU, Mobile, etc.)
• Very flexible but performance has been an issue
• Certain Python peculiarities like `variable_scope` etc.
• \url{https://github.com/tensorflow/tensorflow}

Courtesy: \url{https://www.tensorflow.org/}

\[^1\] Jeffrey Dean et al., “Large Scale Distributed Deep Networks”
Facebook Torch/PyTorch - Catching up fast!

- Torch was written in Lua
  - Adoption wasn’t wide-spread
- PyTorch is a Python adaptation of Torch
  - Gaining lot of attention
- Several contributors
  - Biggest support by Facebook
- There are/maybe plans to merge the PyTorch and Caffe2 efforts
- Key selling point is ease of expression and “define-by-run” approach
Preferred Networks Chainer/ChainerMN

- ChainerMN provides multi-node parallel/distributed training using MPI
  - MVAPICH2 MPI library is being used by Preferred Networks
  - http://mvapich.cse.ohio-state.edu

- ChainerMN is geared towards performance
  - Uses **Define-by-run** (Chainer, PyTorch) approach instead of **Define-and-run** (Caffe, TensorFlow, Torch, Theano) approach
  - https://github.com/chainer/chainer
  - Focus on Speed as well as multi-node Scaling
  - Beats CNTK, MXNet, and TensorFlow for training ResNet-50 on 128 GPUs [1]

Many Other DL Frameworks...

- Keras - https://keras.io
- MXNet - http://mxnet.io
- Theano - http://deeplearning.net/software/theano/
- The list keeps growing and the names keep getting longer and weirder ;-) 
  - Livermore Big Artificial Neural Network Toolkit (LBANN) - https://github.com/LLNL/lbann
Outline

• Introduction

• Overview of Execution Environments
  • Parallel and Distributed DNN Training
  • Latest Trends in HPC Technologies
  • Challenges in Exploiting HPC Technologies for Deep Learning
  • Solutions and Case Studies
  • Open Issues and Challenges
  • Conclusion
So where do we run our DL framework?

• Early (2014) frameworks used a single fast GPU
  – As DNNs became larger, faster and better GPUs became available
  – At the same time, parallel (multi-GPU) training gained traction as well

• Today
  – Parallel training on multiple GPUs is being supported by most frameworks
  – Distributed (multiple nodes) training is still upcoming
    • A lot of fragmentation in the efforts (MPI, Big-Data, NCCL, Gloo, etc.)
  – On the other hand, DL has made its way to Mobile and Web too!
    • Smartphones - OK Google, Siri, Cortana, Alexa, etc.
    • DrivePX – the computer that drives NVIDIA’s self-driving car
    • Deeplearn.js – a DL framework in a web-browser
    • TensorFlow playground - [http://playground.tensorflow.org/](http://playground.tensorflow.org/)
Conventional Execution on GPUs and CPUs

- My framework is faster than your framework!
- This needs to be understood in a holistic way.
- Performance depends on the entire execution environment (the full stack)
- Isolated view of performance is not helpful

DL Frameworks and Underlying Libraries

• BLAS Libraries – the heart of math operations
  – Atlas/OpenBLAS
  – NVIDIA cuBlas
  – Intel Math Kernel Library (MKL)

• Most compute intensive layers are generally optimized for a specific hardware
  – E.g. Convolution Layer, Pooling Layer, etc.

• DNN Libraries – the heart of Convolutions!
  – NVIDIA cuDNN (already reached its 7th iteration – cudnn-v7.5)
  – Intel MKL-DNN (MKL 2018) – recent but a very promising development
Where does the Performance come from?

- The full landscape: Forward and Backward Pass -- **Faster Convolutions → Faster Training**
- Performance of Intel KNL == NVIDIA P100 for AlexNet Training – **Volta is in a different league!**
- Most performance gains are based on improvements in layer **conv2** and **conv3** for AlexNet

Outline

• Introduction
• Overview of Execution Environments

• Parallel and Distributed DNN Training
  • Latest Trends in HPC Technologies
  • Challenges in Exploiting HPC Technologies for Deep Learning
  • Solutions and Case Studies
  • Open Issues and Challenges
• Conclusion
The Need for Parallel and Distributed Training

• Why do we need Parallel Training?
• Larger and Deeper models are being proposed
  – *AlexNet* to *ResNet* to *Neural Machine Translation (NMT)*
  – DNNs require a lot of memory
  – Larger models cannot fit a GPU’s memory
• Single GPU training became a bottleneck
• As mentioned earlier, community has already moved to multi-GPU training
• Multi-GPU in one node is good but there is a limit to Scale-up (8 GPUs)
• **Multi-node (Distributed or Parallel) Training is necessary!!**
Batch-size, Model-size, Accuracy, and Scalability

- Increasing model-size generally increases accuracy
- Increasing batch-size requires tweaking hyper-parameters to maintain accuracy
  - Limits for batch-size
  - Cannot make it infinitely large
  - Over-fitting
- **Large batch size generally helps scalability**
  - More work to do before the need to synchronize
- Increasing the model-size (no. of parameters)
  - Communication overhead becomes bigger so scalability decreases
  - GPU memory is precious and can only fit finite model data

Benefits of Distributed Training: An Example with Caffe

- Strong scaling CIFAR10 Training with OSU-Caffe (1 → 4 GPUs) – **Batch Size 2K**
- Large batch size is needed for scalability.
- Adding more GPUs will degrade the scaling efficiency

Run Command - (change $np from 1—4)

```bash
mpirun_rsh -np $np ./build/tools/caffe train -solver examples/cifar10/cifar10_quick_solver.prototxt -scal strong
```


OSU-Caffe is available from the HiDL project page [http://hidl.cse.ohio-state.edu](http://hidl.cse.ohio-state.edu)
Parallelization Strategies

- What are the Parallelization Strategies
  - Model Parallelism
  - Data Parallelism (Received the most attention)
  - Hybrid Parallelism
  - Automatic Selection

Communication in Distributed Frameworks

• What are the Design Choices for Communication?
  – Established paradigms like Message Passing Interface (MPI)
  – Develop specific communication libraries like NCCL, Gloo, Baidu-allreduce, etc.
  – Use Big-Data frameworks like Spark, Hadoop, etc.
    • Still need some form of external communication for parameters (RDMA, IB, etc.)

• Focus on Scale-up and Scale-out
  – What are the challenges and opportunities?
Scale-up and Scale-out

- **Scale-up**: Intra-node Communication
  - Many improvements like:
    - NVIDIA cuDNN, cuBLAS, NCCL, etc.
    - CUDA 9 Co-operative Groups

- **Scale-out**: Inter-node Communication
  - DL Frameworks – most are optimized for single-node only
  - Distributed (Parallel) Training is an emerging trend
    - **OSU-Caffe** – MPI-based
    - Microsoft CNTK – MPI/NCCL2
    - Google TensorFlow – gRPC-based/MPI/NCCL2
    - Facebook Caffe2 – Hybrid (NCCL2/Gloo/MPI)
Data Parallel Deep Learning and MPI Collectives

- **Major MPI Collectives** involved in Designing distributed frameworks
- **MPI_Bcast** – required for DNN parameter exchange
- **MPI_Reduce** – needed for gradient accumulation from multiple solvers
- **MPI_Allreduce** – use just one Allreduce instead of Reduce and Broadcast

Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
  - Challenges in Exploiting HPC Technologies for Deep Learning
  - Solutions and Case Studies
  - Open Issues and Challenges
- Conclusion
Drivers of Modern HPC Cluster Architectures

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs)
- Available on HPC Clouds, e.g., Amazon EC2, NSF Chameleon, Microsoft Azure, etc.
HPC Technologies

• Hardware
  – Interconnects – InfiniBand, RoCE, Omni-Path, etc.
  – Processors – GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.

• Communication Middleware
  – Message Passing Interface (MPI)
    • CUDA-Aware MPI, Many-core Optimized MPI runtimes (KNL-specific optimizations)
  – NVIDIA NCCL
Overview of High Performance Interconnects

• High-Performance Computing (HPC) has adopted advanced interconnects and protocols
  – InfiniBand (IB)
  – Omni-Path
  – High Speed Ethernet 10/25/40/50/100 Gigabit Ethernet/iWARP
  – RDMA over Converged Enhanced Ethernet (RoCE)

• Very Good Performance
  – Low latency (few micro seconds)
  – High Bandwidth (200 Gb/s with HDR InfiniBand)
  – Low CPU overhead (5-10%)

• OpenFabrics software stack with IB, Omni-Path, iWARP and RoCE interfaces are driving HPC systems

• Many such systems in Top500 list
## Network Speed Acceleration with IB and HSE

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet (1979 - )</td>
<td>10 Mbit/sec</td>
</tr>
<tr>
<td>Fast Ethernet (1993 -)</td>
<td>100 Mbit/sec</td>
</tr>
<tr>
<td>Gigabit Ethernet (1995 -)</td>
<td>1000 Mbit/sec</td>
</tr>
<tr>
<td>ATM (1995 -)</td>
<td>155/622/1024 Mbit/sec</td>
</tr>
<tr>
<td>Myrinet (1993 -)</td>
<td>1 Gbit/sec</td>
</tr>
<tr>
<td>Fibre Channel (1994 -)</td>
<td>1 Gbit/sec</td>
</tr>
<tr>
<td>InfiniBand (2001 -)</td>
<td>2 Gbit/sec (1X SDR)</td>
</tr>
<tr>
<td>10-Gigabit Ethernet (2001 -)</td>
<td>10 Gbit/sec</td>
</tr>
<tr>
<td>InfiniBand (2003 -)</td>
<td>8 Gbit/sec (4X SDR)</td>
</tr>
<tr>
<td>InfiniBand (2005 -)</td>
<td>16 Gbit/sec (4X DDR)</td>
</tr>
<tr>
<td>InfiniBand (2007 -)</td>
<td>24 Gbit/sec (12X SDR)</td>
</tr>
<tr>
<td>InfiniBand (2010 -)</td>
<td>32 Gbit/sec (4X QDR)</td>
</tr>
<tr>
<td>40-Gigabit Ethernet (2010 -)</td>
<td>40 Gbit/sec</td>
</tr>
<tr>
<td>InfiniBand (2011 -)</td>
<td>54.6 Gbit/sec (4X FDR)</td>
</tr>
<tr>
<td>InfiniBand (2012 -)</td>
<td>2 x 54.6 Gbit/sec (4X Dual-FDR)</td>
</tr>
<tr>
<td>25-/50-Gigabit Ethernet (2014 -)</td>
<td>25/50 Gbit/sec</td>
</tr>
<tr>
<td>100-Gigabit Ethernet (2015 -)</td>
<td>100 Gbit/sec</td>
</tr>
<tr>
<td>Omni-Path (2015 -)</td>
<td>100 Gbit/sec</td>
</tr>
<tr>
<td>InfiniBand (2015 -)</td>
<td>100 Gbit/sec (4X EDR)</td>
</tr>
<tr>
<td>InfiniBand (2018 -)</td>
<td>200 Gbit/sec (4X HDR)</td>
</tr>
</tbody>
</table>

*100 times in the last 17 years*
Intel Neural Network Processor (NNP)

- Intel® Nervana™ Neural Network Processors (NNP)
  - formerly known as “Lake Crest”
- Recently announced as part of Intel’s strategy for next-generation AI systems
- Purpose built architecture for deep learning
- 1 TB/s High Bandwidth Memory (HBM)
- Spatial Architecture
- FlexPoint format
  - Similar performance (in terms of accuracy) to FP32 while using 16 bits of storage

GraphCore – Intelligence Processing Unit (IPU)

- New processor that’s the first to be specifically designed for machine intelligence workloads – an Intelligence Processing Unit (IPU)
  - Massively parallel
  - Low-precision floating-point compute
  - Higher compute density
- UK-based Startup
- Early benchmarks show 10-100x speedup over GPUs
  - Presented at NIPS 2017

Courtesy: https://www.graphcore.ai/posts/preliminary-ipu-benchmarks-providing-previously-unseen-performance-for-a-range-of-machine-learning-applications
HPC Technologies

• Hardware
  – Interconnects – InfiniBand, RoCE, Omni-Path, etc.
  – Processors – GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.

• Communication Middleware
  – Message Passing Interface (MPI)
    • CUDA-Aware MPI, Many-core Optimized MPI runtimes (KNL-specific optimizations)
  – NVIDIA NCCL
Parallel Programming Models Overview

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

Shared Memory Model
SHMEM, DSM

Distributed Memory Model
MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)
OpenSHMEM, UPC, Chapel, X10, CAF, ...
Allreduce Collective Communication Pattern

- Element-wise Sum data from all processes and sends to all processes

```
int MPI_Allreduce (const void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype,
                  MPI_Op operation, MPI_Comm comm)
```

### Input-only Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sendbuf</td>
<td>Starting address of send buffer</td>
</tr>
<tr>
<td>recvbuf</td>
<td>Starting address of recv buffer</td>
</tr>
<tr>
<td>type</td>
<td>Data type of buffer elements</td>
</tr>
<tr>
<td>count</td>
<td>Number of elements in the buffers</td>
</tr>
<tr>
<td>operation</td>
<td>Reduction operation to be performed (e.g. sum)</td>
</tr>
<tr>
<td>comm</td>
<td>Communicator handle</td>
</tr>
</tbody>
</table>

### Input/Output Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recvbuf</td>
<td>Starting address of receive buffer</td>
</tr>
</tbody>
</table>

Sendbuf (Before):

- T1: 1 2 3 4
- T2: 1 2 3 4
- T3: 1 2 3 4
- T4: 1 2 3 4

Recvbuf (After):

- T1: 4 8 12 16
- T2: 4 8 12 16
- T3: 4 8 12 16
- T4: 4 8 12 16
Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.1), Started in 2001, First version available in 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
- Used by more than 2,975 organizations in 86 countries
- More than 528,000 (> 0.5 million) downloads from the OSU site directly
- Empowering many TOP500 clusters (Nov ‘18 ranking)
  - 3rd ranked 10,649,640-core cluster (Sunway TaihuLight) at NSC, Wuxi, China
  - 14th, 556,104 cores (Oakforest-PACS) in Japan
  - 17th, 367,024 cores (Stampede2) at TACC
  - 27th, 241,108-core (Pleiades) at NASA and many others
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, and OpenHPC)
- Partner in the upcoming TACC Frontera System
- http://mvapich.cse.ohio-state.edu
- Empowering Top500 systems for over a decade
GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GDR

- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

\[
\text{MPI\_Send(s\_devbuf, size, ...)};
\]

At Receiver:

\[
\text{MPI\_Recv(r\_devbuf, size, ...)};
\]

High Performance and High Productivity
Optimized MVAPICH2-GDR Design

**GPU-GPU Inter-node Latency**
- **MV2-(NO-GDR)**
- **MV2-GDR 2.3**
- **1.85us**
- **10x**

**GPU-GPU Inter-node Bandwidth**
- **MV2-(NO-GDR)**
- **MV2-GDR-2.3**
- **9x**

**GPU-GPU Inter-node Bi-Bandwidth**
- **MV2-(NO-GDR)**
- **MV2-GDR-2.3**
- **11X**

**Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores**
- **NVIDIA Volta V100 GPU**
- **Mellanox Connect-X4 EDR HCA**
- **CUDA 9.0**
- **Mellanox OFED 4.0 with GPU-Direct-RDMA**
NCCL Communication Library

• Collective Communication with a caveat!
  – GPU buffer exchange
  – Dense Multi-GPU systems
    (Cray CS-Storm, DGX-1)
  – MPI-like – but not MPI standard compliant

• NCCL (pronounced Nickel)
  – Open-source Communication Library by NVIDIA
  – Topology-aware, ring-based (linear) collective
    communication library for GPUs
  – Divide bigger buffers to smaller chunks
  – Good performance for large messages
    • Kernel-based threaded copy (Warp-level Parallel)
      instead of cudaMemcpy

Outline

• Introduction
• Overview of Execution Environments
• Parallel and Distributed DNN Training
• Latest Trends in HPC Technologies
• Challenges in Exploiting HPC Technologies for Deep Learning
  • Solutions and Case Studies
  • Open Issues and Challenges
• Conclusion
Broad Challenge: Exploiting HPC for Deep Learning

**How to efficiently scale-out a Deep Learning (DL) framework and take advantage of heterogeneous High Performance Computing (HPC) resources?**
1. What are the fundamental issues in designing DL frameworks?
   - Memory Requirements
   - Computation Requirements
   - Communication Overhead

2. Why do we need to support distributed training?
   - To overcome the limits of single-node training
   - To better utilize hundreds of existing HPC Clusters
3. What are the **new design challenges** brought forward by DL frameworks for Communication runtimes?
   - Large Message **Collective Communication** and Reductions
   - GPU Buffers (**CUDA-Awareness**)  

4. Can a **Co-design** approach help in achieving Scale-up and Scale-out efficiently?
   - **Co-Design** the support at **Runtime level** and Exploit it at the DL **Framework level**
   - What performance benefits can be observed?
   - What needs to be fixed at the communication runtime layer?
Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in HPC Technologies
- Challenges in Exploiting HPC Technologies for Deep Learning
  - **Solutions and Case Studies**
    - Open Issues and Challenges
    - Conclusion
Solutions and Case Studies: Exploiting HPC for DL

- **NVIDIA NCCL/NCCL2**
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL
NVIDIA NCCL

- NCCL is a collective communication library
  - NCCL 1.x is only for Intra-node communication on a single-node
- NCCL 2.0 supports inter-node communication as well
- Design Philosophy
  - Use Rings and CUDA Kernels to perform efficient communication
- NCCL is optimized for dense multi-GPU systems like the DGX-1 and DGX-1V

Fully connected quad
120 GB/s per GPU bidirectional for peer traffic
40 GB/s per GPU bidirectional to CPU
Direct Load/store access to CPU Memory
High Speed Copy Engines for bulk data movement

Courtesy: https://www.nextplatform.com/2016/05/04/nvlink-takes-gpu-acceleration-next-level/
NCCL 2: Multi-node GPU Collectives

Optimized designs in MVAPICH2-GDR 2.3 offer better/comparable performance for most cases.

MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 16 GPUs

*Available since MVAPICH2-GDR 2.3

Platform: Intel Xeon (Broadwell) nodes equipped with a dual-socket CPU, 1 K-80 GPUs, and EDR InfiniBand Inter-connect
MVAPICH2-GDR vs. NCCL2 – Allreduce on DGX-2

- Optimized designs in MVAPICH2-GDR 2.3.1 offer better/comparable performance for most cases
- MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 1 DGX-2 node (16 Volta GPUs)

*Available with MVAPICH2-GDR 2.3.1

Platform: Nvidia DGX-2 system (16 Nvidia Volta GPUs connected with NVSwitch), CUDA 9.2
Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL
Baidu’s Ring-Allreduce in TensorFlow

Scaling with TensorFlow

• Run many independent TensorFlow processes

• Insert allreduce as a node in the graph:

![Diagram showing TensorFlow process flow](http://on-demand.gputechconf.com/gtc/2017/presentation/s7543-andrew-gibiansky-effectively-scakubg-deep-learning-frameworks.pdf)
MVAPICH2-GDR: Allreduce Comparison with Baidu and OpenMPI

- 16 GPUs (4 nodes) MVAPICH2-GDR vs. Baidu-Allreduce and OpenMPI 3.0

*Available since MVAPICH2-GDR 2.3a*
Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL
Facebook Caffe2

- Caffe2 (by Facebook) allows the use of multiple communication back-ends
  - Gloo – Multi-node design from the beginning
  - NCCL – Multi-node support added recently in v2
- Gloo – Performance evaluation studies not available yet
- Design principles are similar to MPI and NCCL
- In essence, Gloo is an application level implementation of collective algorithms for Reduce, Allreduce, etc.
- Details and code available from: https://github.com/facebookincubator/gloo
Facebook: Training ImageNet in 1 Hour

- Near-linear Scaling for ~256 Pascal GPUs (Facebook Big Basin Servers with 8 GPUs/node)
- Explored large batch-size training with ResNet-50
  - 8K batch-size seems to be the sweet-spot.

Courtesy: https://research.fb.com/publications/imagenet1kin1h/
Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL
S-Caffe: Proposed Co-Design Overview

- To address the limitations of Caffe and existing MPI runtimes, we propose the **OSU-Caffe (S-Caffe)** framework

- At the application (DL framework) level
  - Develop a fine-grain workflow – i.e. layer-wise communication instead of communicating the entire model

- At the runtime (MPI) level
  - Develop support to perform reduction of very-large GPU buffers
  - Perform reduction using GPU kernels

OSU-Caffe is available from the HiDL project page
[http://hidl.cse.ohio-state.edu](http://hidl.cse.ohio-state.edu)
OSU-Caffe: Scalable Deep Learning

- Benefits and Weaknesses
  - Multi-GPU Training within a single node
  - Performance degradation for GPUs across different sockets
  - Limited Scale-out
- OSU-Caffe: MPI-based Parallel Training
  - Enable Scale-up (within a node) and Scale-out (across multi-GPU nodes)
  - Scale-out on 64 GPUs for training CIFAR-10 network on CIFAR-10 dataset
  - Scale-out on 128 GPUs for training GoogLeNet network on ImageNet dataset

Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL
Distributed Training using TensorFlow (TF)

- TensorFlow is the most popular DL framework
- gRPC is the official distributed training runtime
  - Many problems for HPC use-cases
- Community efforts - Baidu and Uber’s Horovod have added MPI support to TF across nodes
- Need to understand several options currently available

https://arxiv.org/abs/1810.11112
Scalable TensorFlow using Horovod, MPI, and NCCL

- Efficient Allreduce is crucial for Horovod’s overall training performance
  - Both MPI and NCCL designs are available
- We have evaluated Horovod extensively and compared across a wide range of designs using gRPC and gRPC extensions
- MVAPICH2-GDR achieved up to 90% scaling efficiency for ResNet-50 Training on 64 Pascal GPUs

https://arxiv.org/abs/1810.11112
Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL
# Caffe2 Performance Optimization with Intel MKL

<table>
<thead>
<tr>
<th>batch size</th>
<th>OMP_NUM_THREADS=44</th>
<th>OMP_NUM_THREADS=1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Intel® MKL (images/sec)</td>
<td>Eigen BLAS (images/sec)</td>
</tr>
<tr>
<td>1</td>
<td>173.4</td>
<td>5.2</td>
</tr>
<tr>
<td>32</td>
<td>1500.2</td>
<td>29.3</td>
</tr>
<tr>
<td>64</td>
<td>1596.3</td>
<td>35.3</td>
</tr>
<tr>
<td>256</td>
<td>1735.2</td>
<td>44.9</td>
</tr>
</tbody>
</table>

TensorFlow Optimization for Intel CPUs

Optimized Perf_Alexnet on different batch size

72x Speedup From New Optimizations – available through Google's TensorFlow Git

Intel Machine Learning Scaling Library (MLSL)

- Intel MLSL is built on top of MPI primitives
  - [https://github.com/01org/MLSL](https://github.com/01org/MLSL)
- Works across various interconnects: Intel(R) Omni-Path Architecture, InfiniBand*, and Ethernet
- Common API to support Deep Learning frameworks (Caffe*, Theano*, Torch*, etc.)

<table>
<thead>
<tr>
<th>MLSL::Activation</th>
<th>A wrapper class for operation input and output activations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLSL::CommBlockInfo</td>
<td>A class to hold block information for activations packing/unpacking</td>
</tr>
<tr>
<td>MLSL::Distribution</td>
<td>A class to hold the information about the parallelism scheme being used</td>
</tr>
<tr>
<td>MLSL::Environment</td>
<td>A singleton object that holds global Intel MLSL functions</td>
</tr>
<tr>
<td>MLSL::Operation</td>
<td>A class to hold information about learnable parameters (parameter sets) and activations corresponding to a certain operation of the computational graph</td>
</tr>
<tr>
<td>MLSL::OperationRegInfo</td>
<td>A class to hold Operation registration information</td>
</tr>
<tr>
<td>MLSL::ParameterSet</td>
<td>A wrapper class for operation parameters</td>
</tr>
<tr>
<td>MLSL::Session</td>
<td>A class to represent a collection of Operation objects with the same global mini-batch size</td>
</tr>
<tr>
<td>MLSL::Statistics</td>
<td>A class to measure and store performance statistics of communication among processes that perform computation in the computational graph</td>
</tr>
</tbody>
</table>

Courtesy: [https://github.com/01org/MLSL](https://github.com/01org/MLSL)
Solutions and Case Studies: Exploiting HPC for DL

- NVIDIA NCCL
- LLNL Aluminum
- Baidu-allreduce
- Facebook Gloo
- Co-design MPI runtimes and DL Frameworks
- Distributed Training for TensorFlow
- Scaling DNN Training on Multi-/Many-core CPUs
- PowerAI DDL
IBM PowerAI DDL

IBM Power System for HPC, with NVLink
Breakthrough performance for GPU accelerated applications, including Deep Learning and Machine Learning.

PowerAI DDL Performance

IBM Distributed Deep Learning Scaling Efficiency

Caffe with PowerAI DDL on ResNet-50 model using the ImageNet-1K data set on 64 Power8 servers

Courtesy:
Outline

• Introduction
• Overview of Execution Environments
• Parallel and Distributed DNN Training
• Latest Trends in HPC Technologies
• Challenges in Exploiting HPC Technologies for Deep Learning
• Solutions and Case Studies
• **Open Issues and Challenges**
• Conclusion
Open Issues and Challenges

• Convergence of DL and HPC
• Scalability and Large batch-size training?
• DL Benchmarks and Thoughts on Standardization
Convergence of DL and HPC

• Is Deep Learning an HPC Problem?
  – Distributed DNN Training is definitely an HPC problem
  – Inference – not yet an HPC problem

• Why HPC can help?
  – Decades of research for communication models and performance optimizations
  – MPI, PGAS, and other upcoming programming models and communication runtimes can help for “data-parallel” training

• Some of the needs for DNN training are an exact match
  – Compute intensive problem

• Some needs are new for distributed/parallel communication runtimes
  – Large Message Communication
  – CUDA-Aware Communication
Scalability and Large batch-size training?

• Large batch-size helps improve the scalability
  – Lesser communication and more compute before synchronization
  – Limits to large batch-size
    • DL community is actively exploring this area
    • HPC community can also investigate overlap and latency-hiding techniques

• Is there a limit to DNN size?
  – Noam Shazeer’s Outrageously Large Model (137 Billion Parameters)

• Out-of-core Training for GPUs?
  – Prune the network or selectively allocate/de-allocate memory on GPUs
  – OC-DNN and OC-Caffe
Scalability and Large (Out-of-core) Models?

- Large DNNs cannot be trained on GPUs due to memory limitation!
  - ResNet-50 for Image Recognition but current frameworks can only go up to a small batch size of 45
  - Next generation models like Neural Machine Translation (NMT) are ridiculously large, consists of billions of parameters, and require even more memory
  - Can we design Out-of-core DNN training support using new software features in CUDA 8/9 and hardware mechanisms in Pascal/Volta GPUs?

- General intuition is that managed allocations “will be” slow!
  - The proposed framework called OC-Caffe (Out-of-Core Caffe) shows the potential of managed memory designs that can provide performance with negligible/no overhead.

- OC-Caffe-Opt: up to 80% better than Intel-optimized CPU Caffe for ResNet-50 training on the Volta V100 GPU with CUDA9 and CUDNN7

DL Benchmarks and Thoughts on Standardization

• Can we have a standardized interface?
  – Are we there yet?
  – Deep Learning Interface (DLI)? Inspired by Message Passing Interface (MPI)
    • What can be a good starting point?
    • Will it come from the HPC community or the DL community?
    • Can there be a collaboration across communities?

• What about standard benchmarks? Is there a need?
  – State-of-the-art
    • HKBU benchmarks - http://dlbench.comp.hkbu.edu.hk
    • Soumith Chintala’s benchmarks - https://github.com/soumith/convnet-benchmarks
    • MLPerf – https://www.mlperf.org -- Latest and Widely Promoted now!
Outline

• Introduction
• Overview of Execution Environments
• Parallel and Distributed DNN Training
• Latest Trends in HPC Technologies
• Challenges in Exploiting HPC Technologies for Deep Learning
• Solutions and Case Studies
• Open Issues and Challenges

• Conclusion
Conclusion

• Exponential growth in Deep Learning frameworks

• Provided an overview of issues, challenges, and opportunities for communication runtimes
  – Efficient, scalable, and hierarchical designs are crucial for DL frameworks
  – Co-design of communication runtimes and DL frameworks will be essential
    • OSU-Caffe
    • TensorFlow (Baidu, Uber’s Horovod, etc.)
    • Neon and Nervana Graph

• Need collaborative efforts to achieve the full potential

• Standardization may help remove fragmentation in DL frameworks
Please join us for more events..

<table>
<thead>
<tr>
<th>Monday, March 18</th>
<th>Tuesday, March 19</th>
<th>Wednesday, March 20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Research Poster</strong></td>
<td><strong>Talk</strong></td>
<td><strong>Instructor-Led Training</strong></td>
</tr>
<tr>
<td>1. <strong>P9243</strong> - Exploiting CUDA Unified Memory for Efficient Out-of-Core DNN Training</td>
<td><strong>S9476</strong> - MVAPICH2-GDR: High-Performance and Scalable CUDA-Aware MPI Library for HPC and AI</td>
<td><strong>L9121</strong> - How to Boost the Performance of HPC/AI Applications Using MVAPICH2 Library</td>
</tr>
<tr>
<td>2. <strong>P9242</strong> - Exploiting GPUDirect Technology and Hardware Multicast for Streaming and Deep Learning Applications</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SJCC Upper Concourse</strong></td>
<td><strong>SJCC Room 211A</strong> (Concourse Level)</td>
<td><strong>SJCC Room LL21D</strong> (Lower Level)</td>
</tr>
<tr>
<td><strong>06:00 PM - 08:00 PM</strong></td>
<td><strong>03:00 PM - 03:50 PM</strong></td>
<td><strong>08:00 AM - 10:00 AM</strong></td>
</tr>
</tbody>
</table>
Funding Acknowledgments

**Funding Support by**

[Logos of various sponsors]

**Equipment Support by**

[Logos of various sponsors]
Personnel Acknowledgments

Current Students (Graduate)
- A. Awan (Ph.D.)
- M. Bayatpour (Ph.D.)
- S. Chakraborty (Ph.D.)
- C.-H. Chu (Ph.D.)
- S. Guganani (Ph.D.)

Current Students (Undergraduate)
- J. Hashmi (Ph.D.)
- A. Jain (Ph.D.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- D. Shankar (Ph.D.)

Current Students (Undergraduate)
- V. Gangal (B.S.)
- M. Haupt (B.S.)
- N. Sarkauskas (B.S.)
- A. Yeretzian (B.S.)

Current Research Asst. Professor
- X. Lu

Current Post-doc
- A. Ruhela
- K. Manian

Current Research Scientist
- H. Subramoni

Past Students
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- R. Biswas (M.S.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharpappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- R. Rajachandrasekar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)

Past Research Scientist
- K. Hamidouche
- S. Sur

Past Programmers
- D. Bureddy
- J. Perkins

Past Research Specialist
- M. Arnold

Past Post-Docs
- D. Banerjee
- X. Besseron
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne
- H. Wang

Current Research Specialist
- J. Smith

Current Research Assistant Professor
- X. Lu
Thank You!

panda@cse.ohio-state.edu
awan.10@osu.edu
subramon@cse.ohio-state.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/