An In-depth Performance Characterization of CPU- and GPU-based DNN Training on Modern Architectures

Ammar Ahmad Awan, Hari Subramoni, and Dhableswar K. (DK) Panda
Dept. of Computer Science and Engineering
The Ohio State University
CPU based Deep Learning is not as bad as you think!

• Introduction
  – CPU-based Deep Learning
  – Deep Learning Frameworks

• Research Challenges

• Design Discussion

• Performance Characterization

• Conclusion
GPUs are great for Deep Learning

• NVIDIA GPUs have been the main driving force for faster training of Deep Neural Networks (DNNs)
• GPUs: A natural fit for DL due to the throughput-oriented nature
• GPUs are also growing in the HPC arena!
But what about CPUs?

- Intel CPUs are everywhere and many-core CPUs are emerging according to Top500.org
- Host CPUs exist even on the GPU nodes
  - Many-core Xeon Phis are increasing
- Xeon Phi 1st generation: a many-core co-processor
- Xeon Phi 2nd generation (KNL): a self-hosted many-core processor!
- Usually, we hear CPUs are \textbf{10x – 100x} slower than GPUs?
  - \textit{But can we do better?}
Deep Learning Frameworks – CPUs or GPUs?

• There are several Deep Learning (DL) or DNN Training frameworks
  – Caffe, Cognitive Toolkit, TensorFlow, MXNet, and counting....

• Every (almost every) framework has been optimized for NVIDIA GPUs

• **But every framework is able to execute on a CPU as well**
  – So why are we not using them?
  – Performance has been “terrible” and several studies have reported significant degradation when using CPUs

• But there is hope :-)  
  – Coupled with Intel Xeon Phi (Knights Landing or KNL) and MC-DRAM, the landscape for CPU-based DL looks promising..
The DL Framework(s) in discussion: Caffe

- Caffe is a popular and widely used framework
- NVIDIA-Caffe and BVLC-Caffe (Official Caffe) are almost similar
- Intel-Caffe is optimized for CPU-based Deep Learning
- OSU-Caffe is a multi-node multi-GPU variant that we have worked on at OSU

<table>
<thead>
<tr>
<th>Caffe Variant</th>
<th>Multi-GPU Support</th>
<th>Multi-node Support</th>
<th>Multi-node Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>BVLC-Caffe</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>NVIDIA-Caffe</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Intel-Caffe</td>
<td>N/A</td>
<td>Yes</td>
<td>Intel MLSL 2017.1.016 (with Intel MPI 2017)</td>
</tr>
<tr>
<td>OSU-Caffe</td>
<td>Yes</td>
<td>Yes</td>
<td>MVAPICH2-GDR 2.2</td>
</tr>
</tbody>
</table>
Agenda

- Introduction
- **Research Challenges**
- Design Discussion
- Performance Characterization
- Conclusion
Can we provide a holistic yet comprehensive view of DNN training performance for a diverse set of hardware architectures including Intel Xeon Phi (KNL) processors and NVIDIA Pascal GPUs?
Agenda

• Introduction

• Research Challenges

• Design Discussion
  – Caffe Architecture
  – Understanding the Impact of Execution Environments

• Performance Characterization

• Conclusion
Caffe Architecture

1. Data Propagation

2. Forward Backward Pass

3. Gradient Aggregation

Loop {}
Understanding the Impact of Execution Environments

Performance is dependent on:

1. Hardware Architectures
   - GPUs
   - Multi-/Many-core CPUs

2. Software Libraries
   - cuDNN (for GPUs)
   - MKL-DNN/MKL 2017 (for CPUs)

3. Hardware/Software co-design
   - Software libraries optimized for one platform will not help the other!
Agenda

• Introduction
• Research Challenges
• Design Discussion

• Performance Characterization
  – Single-node Performance
  – Multi-node Performance

• Conclusion
Performance Characterization

• Several GPU generations and CPU architectures

• Single-node Results for AlexNet and ResNet-50
  – Impact of MKL engine
  – Impact of MC-DRAM
  – Layer-wise breakdown
  – P100 vs. KNL

• Multi-node results using Intel-Caffe and OSU-Caffe
  – Weak scaling
  – ResNet-50 and AlexNet
## Performance Characterization: Various Architectures

<table>
<thead>
<tr>
<th>Name (Label)</th>
<th>Processor Architecture (Description)</th>
<th>No. of Cores</th>
<th>No. of Sockets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haswell1</td>
<td>Intel Xeon CPU E5-2660 v3 @ 2.60 GHz</td>
<td>20 (2*10)</td>
<td>2</td>
</tr>
<tr>
<td>Haswell2</td>
<td>Intel Xeon CPU E5-2687 v3 @ 3.10 GHz</td>
<td>20 (2*10)</td>
<td>2</td>
</tr>
<tr>
<td>Broadwell</td>
<td>Intel Xeon CPU E5-2680 v4 @ 2.40 GHz</td>
<td>28 (2*14)</td>
<td>2</td>
</tr>
<tr>
<td>KNL</td>
<td>Intel Xeon Phi CPU 7250 @ 1.40 GHz</td>
<td>68 (1*68)</td>
<td>1</td>
</tr>
<tr>
<td>K40</td>
<td>NVIDIA Tesla K40 11.8GB @ 0.75 GHz</td>
<td>2880 CUDA Cores</td>
<td>N/A</td>
</tr>
<tr>
<td>K80</td>
<td>NVIDIA Tesla K80 11.8GB @ 0.82 GHz</td>
<td>2496 CUDA Cores</td>
<td>N/A</td>
</tr>
<tr>
<td>P100</td>
<td>NVIDIA Tesla P100-PCIE 1 6GB @ 1.33 GHz</td>
<td>3584 CUDA Cores</td>
<td>N/A</td>
</tr>
</tbody>
</table>
The comparison of optimized MKL engine and the default Caffe engine

MKL engine is up to 3X better than default Caffe engine

Biggest gains for Intel Xeon Phi (KNL) (many-core) architecture

Both Haswell and Broadwell architectures get significant speedups (up to 1.5X)
Single-node: Impact of Utilizing MCDRAM

- “MCDRAM as Cache” and “MCDRAM-All” offer very similar performance
- **MCDRAM as Cache was chosen** for all the subsequent results
- On average, DDR-All is up to **1.5X slower** than MCDRAM
• The full landscape for AlexNet: Forward and Backward Pass

• **Faster Convolutions** → **Faster Training**

• Most performance gains are based on **conv2** and **conv3** for AlexNet
Diving Deeper: P100 vs. KNL (AlexNet)

- Fully connected layers are much slower on KNL compared to P100
- \textit{conv1} and \textit{conv3} also contribute to degradation on KNL
- \textit{conv2} is faster on KNL compared to P100
Multi-node Results: ResNet-50

- All results are \textit{weak scaling}.

- Images/second is a derived metric but more meaningful for understanding scalability.

![Graph showing training time and images/second vs. number of nodes for ResNet-50 on Intel-Caffe.](image-url)
Multi-node Results: AlexNet Comparison

- OSU-Caffe vs. Intel-Caffe
  - Different frameworks so not directly comparable
  - A rough comparison can still help in understanding scalability trends
  - Design of framework can affect performance for distributed training
    - **MPI (or the communication runtime) can cause a marked difference**
Agenda

• Introduction
• Research Challenges
• Design Comparisons
• Performance Characterization
• Conclusion
Conclusion

• CPU is very comparable to GPU for DNN Training workloads if appropriate optimizations are exploited
• GPUs are still faster than CPUs in general
• KNL beats P100 for one case but P100 beats KNL for most cases
• Evaluating the performance of a DL framework
  – The hardware architecture matters
  – But software stack has a higher and more significant impact than hardware
  – The full execution environment and communication runtime needs to be evaluated to ensure fairness in comparisons
Performance Characterization of DNN Training using TensorFlow and PyTorch on Modern Clusters

Arpan Jain, Ammar Ahmad Awan, Quentin Anthony, Hari Subramoni, and Dhableswar K. (DK) Panda
Dept. of Computer Science and Engineering
The Ohio State University

Credits: http://nowlab.cse.ohio-state.edu/static/media/talks (Arpan Jain)
Agenda

• Introduction
• Background
• Research Challenges
• Characterization Strategy
  – Evaluation Platforms and Software Libraries
  – Experimental Setup
• Performance Evaluation
• Conclusion
Deep Learning Frameworks

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

• Caffe, PyTorch, TensorFlow, MXNet, and counting....
  – Focus on TensorFlow and PyTorch

• Most frameworks - optimized for NVIDIA GPUs–
  – but CPU optimized implementations are also emerging as we saw in
    the previous paper
Deep Learning and TensorFlow

- The most widely used framework open-sourced by Google
- Replaced Google’s DistBelief framework
- Runs on almost all execution platforms available (CPU, GPU, TPU, Mobile, etc.)
- [https://github.com/tensorflow/tensorflow](https://github.com/tensorflow/tensorflow)
Agenda

- Introduction
- **Background**
  - Research Challenges
  - Characterization Strategy
    - Evaluation Platforms and Software Libraries
    - Experimental Setup
- Performance Evaluation
- Conclusion
Deep Neural Network training consists of two phases
- Forward pass
- Backward pass

Two approaches to Distribute DNN training
- Data Parallelism (focus of this paper)
- Model Parallelism
DL Frameworks and Communication Libraries

• Most ML/DL frameworks – started single-node/single-GPU design
  – Various multi-node design schemes have emerged since then!
• Distributed Training needs **communication libraries** to synchronize across nodes
• **DL Frameworks**
  – Caffe
  – **TensorFlow and PyTorch with Horovod (focus of this paper)**
• **Communication Libraries** for DL
  – MPI Libraries: MVAPICH2, IntelMPI, OpenMPI
  – NVIDIA NCCL (GPU only)
What is Allreduce? And How DL frameworks use it?

• A generic group communication pattern – element-wise vector sum available to all participants in the group
• In the MPI world, we call it MPI_Allreduce
• Needed in DNN Training during gradient aggregation from different workers
Agenda

• Introduction

• Background

• **Research Challenges**

  • Characterization Strategy
    – Evaluation Platforms and Software Libraries
    – Experimental Setup

• Performance Evaluation

• Conclusion
How to systemically characterize CPU-based DNN Training using TensorFlow and PyTorch at scale? And how to achieve best possible performance for different HPC systems?
Key Contributions

- Describe single-process (SP), multi-process (MP), and multi-node (MN) approach
- Highlight up to $1.47 \times$ better performance for MP approach over SP approach
- Evaluate five DNN architectures at scale (128 Xeon Skylake nodes)
- Report $125 \times$ speedup on 128 nodes for ResNet-152 with MVAPICH2
- Summarize key insights gained from the systematic characterization
Agenda

• Introduction
• Background
• Research Challenges

• **Characterization Strategy**
  – Evaluation Platforms and Software Libraries
  – Experimental Setup

• Performance Evaluation
• Conclusion
## Evaluation Platforms

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Cluster</th>
<th>Speed (GHz)</th>
<th>Cores</th>
<th>Threads per core</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skylake</td>
<td>RI2</td>
<td>2.6</td>
<td>28</td>
<td>1</td>
<td>Skylake-1</td>
</tr>
<tr>
<td>Skylake</td>
<td>Pitzer</td>
<td>2.4</td>
<td>40</td>
<td>1</td>
<td>Skylake-2</td>
</tr>
<tr>
<td>Skylake</td>
<td>Stampede2</td>
<td>2.1</td>
<td>48</td>
<td>2</td>
<td>Skylake-3</td>
</tr>
<tr>
<td>Broadwell</td>
<td>RI2</td>
<td>2.4</td>
<td>28</td>
<td>1</td>
<td>Broadwell</td>
</tr>
<tr>
<td>EPYC</td>
<td>AMD-Cluster</td>
<td>2.0</td>
<td>32</td>
<td>4</td>
<td>EPYC</td>
</tr>
<tr>
<td>K80</td>
<td>RI2</td>
<td>-</td>
<td>4992 (Dual socket)</td>
<td>-</td>
<td>K80</td>
</tr>
<tr>
<td>P100</td>
<td>Owens</td>
<td>-</td>
<td>3584</td>
<td>-</td>
<td>P100</td>
</tr>
<tr>
<td>V100</td>
<td>Pitzer</td>
<td>-</td>
<td>Cuda: 5120 Tensor: 640</td>
<td>-</td>
<td>V100</td>
</tr>
</tbody>
</table>
Software Libraries

• Deep Learning Frameworks
  – Intel optimized TensorFlow (v1.12), -- details on the next slide
  – TensorFlow v1.12 (for GPUs and AMD processors)
  – PyTorch (v1.1)

• Horovod Distributed Training middleware
• MPI Library: MVAPICH2
• Scripts: tf_cnn_benchmarks and Horovod’s pytorch_synthetic_benchmarks
Intel Optimized TensorFlow

- Optimized by Intel for Intel Xeon CPUs
- Uses Math Kernel Library for Deep Neural Networks – (MKL-DNN) primitives
- Can be installed easily using conda and pip
- [https://github.com/Intel-tensorflow](https://github.com/Intel-tensorflow)
Agenda

• Introduction
• Background
• Research Challenges
• Characterization Strategy
  – Evaluation Platforms and Software Libraries
  – Experimental Setup
• Performance Evaluation
• Conclusion
Experimental Setup

Four different types of experiments were performed

4. GPU vs. CPU Comparisons
Agenda

• Introduction
• Background
• Research Challenges
• Characterization Strategy
  – Evaluation Platforms and Software Libraries
  – Experimental setup
• **Performance Evaluation**
• Conclusion
ResNet-50 Training performance

- Different configurations lead to different performance trends
- **Key Message**: Process per node (PPN), Batch Size, and number of threads are tunable parameters
- Parameters need to be determined and tuned properly!
SP: Effect of Hyper-Threading

ResNet-50 Training performance

- Skylake-3 on Stampede2 is hyper-threaded (two threads per core)
- Possible to run TF on 96 threads
- But, performance degrades beyond 48 threads
  - Why?
    - Depends on the size and type of DNN
Single Node Multi-Process (MP) Experiments

ResNet-152 Training performance

• BS=64, 4ppn is better
• BS=32, 8ppn is slightly better
• However, keeping effective batch size (EBS) low is more important! – Why? (DNN does not converge to SOTA when batch size is large)

ResNet-152 (SP vs. MP)

• MP is better for all effective batch sizes
• Up to $1.35X$ better performance for MP compared to SP for BS=64.
Multi-Node Multi-Process (MN) Experiments

- We use the best SP configuration to run Multi-node experiments
- Evaluate five models to identify common trends
  - All models give near-linear scaling on both platforms

Skylake-1 (28 cores)

Skylake-2 (40 cores)
Multi-Node Multi-Process (MN): MP vs. SP?

Skylake-3 (48 cores, 96 threads)
- Scale—32 nodes
- MP-Tuned—up to 1.5X better than SP
- MP-Tuned—10% better than MP-Default
- **Why MP-Tuned is better?**
  - Uses the best possible number of inter-op and intra-op threads
Multi-Node Multi-Process (MN): TF vs. PyTorch

- This is an early experience with PyTorch

- TensorFlow is up to **2.5X faster** than PyTorch for 128 Nodes.

- TensorFlow: up to **125X** speedup for ResNet-152 on 128 nodes

- PyTorch: Scales well but overall lower performance than TensorFlow
Multi-Node Multi-Process (MN): AMD Platform

EPYC for TensorFlow
- TensorFlow is **4X slower** on EPYC compared to Skylake-3
- For EPYC, there is no optimized TensorFlow

EPYC for PyTorch
- PyTorch—better than TensorFlow
- Up to **19% better** than TensorFlow on 8 nodes.
TensorFlow and PyTorch: CPU vs. GPU

TensorFlow on GPUs vs. CPUs
- Inception-v4: Skylake-3 up to 2.35X faster than K80s
- ResNet-101: V100s up to 3.32X faster than Skylake-3

Multi-Node: TensorFlow (TF) vs. PyTorch (PT)
- ResNet-50: PT slightly better than TF
- ResNet-152, PT up to 12% better than TF
Agenda

- Introduction
- Background
- Research Challenges
- Characterization Strategy
  - Evaluation Platforms and Software Libraries
  - Experimental setup
- Performance Evaluation
- Conclusion
Conclusion

• In-depth Characterization for Distributed Training with TensorFlow and early results for PyTorch
  – Experiments on **five HPC clusters** including **Stampede2** and three different CPU architectures: Skylake, Broadwell, and AMD EPYC
  – Single Node Single Process (SP) and Single Node Multi Process (MP) to determine best performance for single node experiments
  – Use best single-node configuration for multi-Node experiments
  – Up to 128 nodes to show DNN training scaling
  – GPU vs. CPU comparisons for both TensorFlow and PyTorch
• Guidelines for the DL Researchers to get best performance on CPU platforms