Profiling DNN Workloads on a Volta-based DGX-1 System

Saiful A. Mojumder\textsuperscript{1}, Marcia S Louis\textsuperscript{1}, Yifan Sun\textsuperscript{2}, Amir Kavyan Ziabari\textsuperscript{3}, José L. Abellán\textsuperscript{4}, John Kim\textsuperscript{5}, David Kaeli\textsuperscript{2} and Ajay Joshi\textsuperscript{1}

\textsuperscript{1}ECE Department, BU; \textsuperscript{2}ECE Department, NEU; \textsuperscript{3}AMD; \textsuperscript{4}CS Department, UCAM; \textsuperscript{5}School of EE, KAIST;

\textsuperscript{\#}jlabellan@ucam.edu

AISTECS@HiPEAC January 21st, 2019
Networks-on-chip for GPUs

Asymmetric NoC Architectures for GPU Systems (NOCS’15)

(a) L1-to-L2 network in MeshX2  
(b) L2-to-L1 network in MeshX2  

(c) Networks in ButterflyX2.

Leveraging Silicon-Photonic NoC for Designing Scalable GPUs (ICS’15)

6.02 mm  11.31 mm  6.02 mm

25.01 mm

14.22 mm  13.14 mm

Saiful A. Mojumder, Marcia S Louis, Yifan Sun, Amir Kavyan Ziabari, José L. Abellán, John Kim, David Kaeli, and Ajay Joshi

AISTECS@HiPEAC January 21st
Memory Management for Multi-GPU Systems

UMH: A Hardware-Based Unified Memory Hierarchy for Systems with Multiple Discrete GPUs (TACO’16)
Motivation

Deep Learning is Popular

- Achieves high accuracy
- Solves complex problems
Motivation

Training of Deep Neural Networks is Time Consuming

- Efficient hardware and software are needed
- GPU and Multi-GPU System accelerate training

Retrieved from https://www.2work.com.br
Our Major Goals

Understand the Characteristics of DNN Workloads
- Training of DNNs
- Compute– and communication–intensiveness

Identify the Factors Affecting the Training of DNNs
- Hardware-level limitations
- Software-level limitations
- Scaling of DNN workloads
**Background: DNN**

**What is a DNN?**
- Multiple layers of neurons
- Two neighboring layers connected via weights
Background: Training Stages of a DNN

- **Forward Propagation (FP)**

  ![](image)

  - Input Layer
  - Hidden Layer
  - Output Layer

  **CO** = Calculated Output

- **Backward Propagation (BP)**

- **Weight Update (WU)**

  $\mathbf{N}_W = \mathbf{O}_W + \alpha \times f(G)$

  - $\mathbf{N}_W \rightarrow$ New Weight
  - $\mathbf{O}_W \rightarrow$ Old Weight
  - $\alpha \rightarrow$ Constant
  - $f(G) \rightarrow$ Averaged Gradients

- **Metric Evaluation (ME)**

  - Forward propagation
  - Simple arithmetic operation
Background: Training Stages of a DNN

**Forward Propagation (FP)**

- CO = Calculated Output

**Backward Propagation (BP)**

- EO~CO = Gradients

**Weight Update (WU)**

\[ N_W = O_W + \alpha \times f(G) \]

- N_W → New Weight
- O_W → Old Weight
- \( \alpha \) → Constant
- f(G) → Averaged Gradients

**Metric Evaluation (ME)**

- Forward propagation
- Simple arithmetic operation

---

Saiful A. Mojumder, Marcia S Louis, Yifan Sun, Amir Kavyan Ziabari, José L. Abellán, John Kim, David Kaeli, and Ajay Joshi

AISTECS@HiPEAC January 21st
Background: Training Stages of a DNN

- **Forward Propagation (FP)**
  - CO = Calculated Output

- **Backward Propagation (BP)**
  - EO ~ CO
  - EO = Expected Output

- **Weight Update (WU)**
  - NW = OW + α × f(G)
  - NW → New Weight
  - OW → Old Weight
  - α → Constant
  - f(G) → Averaged Gradients
Background: Training Stages of a DNN

- **Forward Propagation (FP)**

- **Backward Propagation (BP)**

- **Weight Update (WU)**
  \[ N_W = O_W + \alpha \times f(G) \]
  - \( N_W \rightarrow \) New Weight
  - \( O_W \rightarrow \) Old Weight
  - \( \alpha \rightarrow \) Constant
  - \( f(G) \rightarrow \) Averaged Gradients

- **Metric Evaluation (ME)**
  - Forward propagation
  - Simple arithmetic operation
Background: Multi-GPU DNN Training

SGD Algorithm

CPU
GPU 0
GPU 1
GPU 2
GPU 3

Sending
Training Data
**Background: Multi-GPU DNN Training**

**SGD Algorithm**

- **CPU**
- **GPU 0**
- **GPU 1**
- **GPU 2**
- **GPU 3**

![Sending Training Data](image-url)
Background: Multi-GPU DNN Training

SGD Algorithm

CPU FP BP
GPU 0 FP BP
GPU 1 FP BP
GPU 2 FP BP
GPU 3 FP BP

Sending Training Data
Background: Multi-GPU DNN Training

SGD Algorithm

<table>
<thead>
<tr>
<th>CPU</th>
<th>FP</th>
<th>BP</th>
<th>WU</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU 0</td>
<td>FP</td>
<td>BP</td>
<td>AVG</td>
</tr>
<tr>
<td>GPU 1</td>
<td>FP</td>
<td>BP</td>
<td>AVG</td>
</tr>
<tr>
<td>GPU 2</td>
<td>FP</td>
<td>BP</td>
<td>AVG</td>
</tr>
<tr>
<td>GPU 3</td>
<td>FP</td>
<td>BP</td>
<td>AVG</td>
</tr>
</tbody>
</table>

Time

Sending Training Data

Sending Gradient Data
Background: Inter-GPU Communication

P2P Direct Transfer:
cudaMemcpy: DMA copy (data size)
Background: Inter-GPU Communication

P2P Direct Transfer:
\texttt{cudaMemcpy}: DMA copy (data size)

\begin{itemize}
  \item GPU 0
  \item GPU 1
  \item NVLink
\end{itemize}

P2P Direct Access:
Simpler but Cache-line data transfer

\begin{itemize}
  \item GPU 0
  \item GPU 1
  \item NVLink
\end{itemize}
Background: Inter-GPU Communication

**P2P Direct Transfer:**
cudaMemcpy: DMA copy (data size)

**P2P Direct Access:**
Simpler but Cache-line data transfer

---

**NVIDIA Collective Communication Library (NCCL)**
- AllReduce and Broadcast (WU-stage friendly)
- P2P Direct Access
Methodology: Evaluation Platform

DGX-1 System with 8 Tesla V100 GPUs

- Asymmetric interconnect
- Lack of direct NVLink connectivity between all GPUs
- PCIe or Two-hop communication
Methodology: Evaluation Platform

DGX-1 System with 8 Tesla V100 GPUs

- Asymmetric interconnect
- Lack of direct NVLink connectivity between all GPUs
- PCIe or Two-hop communication
DGX-1 System with 8 Tesla V100 GPUs

- Asymmetric interconnect
- Lack of direct NVLink connectivity between all GPUs
- PCIe or Two-hop communication
Methodology: Evaluation Platform

DGX-1 System with 8 Tesla V100 GPUs

- Asymmetric interconnect
- Lack of direct NVLink connectivity between all GPUs
- PCIe or Two-hop communication
Methodology: Frameworks, Workloads and Datasets

Testbed

- MXNet, CUDA 9, cuDNN 7.1, cuBLAS 9.0, NCCL 2.1
- DNNs: LeNet, AlexNet, GoogLeNet, Inception-v3, and ResNet
- nvprof + nvidia-smi
- ImageNet Data set: 256K images
- Batch sizes: 16, 32 and 64

<table>
<thead>
<tr>
<th>Network</th>
<th>Layers</th>
<th>Conv Layers</th>
<th>Incep Layers</th>
<th>FC Layers</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>60K</td>
</tr>
<tr>
<td>AlexNet</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>60M</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>22</td>
<td>3</td>
<td>9</td>
<td>1</td>
<td>4M</td>
</tr>
<tr>
<td>Inception-v3</td>
<td>48</td>
<td>7</td>
<td>11</td>
<td>1</td>
<td>24M</td>
</tr>
<tr>
<td>ResNet</td>
<td>110</td>
<td>107</td>
<td>0</td>
<td>1</td>
<td>55M</td>
</tr>
</tbody>
</table>
Questions We Address

- Do the workloads scale as **GPU count** increases?
- Does **P2P** always perform worse than **NCCL**?
- What is the impact of **network size** on training time?
- What is the impact of **batch size** on training time?
- How do different **stages** in the training process scale with GPU count, batch size and network size?
- What is the impact of **GPU memory** on training?
Do the Workloads Scale with GPU Count?

**LeNet: Batch Size of 16**
- P2P: $1.62\times$, $2.37\times$, and $3.36\times$ for 2, 4, and 8 GPUs, respectively
- NCCL: $1.56\times$, $2.27\times$, and $2.77\times$ for 2, 4, and 8 GPUs, respectively

**LeNet Does Not Scale Well!**
- Why? Small number of layers!
Do the Workloads Scale with GPU Count?

- Not linearly
- How well do they scale?
  - Depends on:
    - DNN model
    - Communication Method (P2P vs NCCL)
Does NCCL Always Outperform P2P?

LeNet: Batch Size of 16
- P2P: $1.62 \times$, $2.37 \times$, and $3.36 \times$ for 2, 4, and 8 GPUs, respectively
- NCCL: $1.56 \times$, $2.27 \times$, and $2.77 \times$ for 2, 4, and 8 GPUs, respectively

LeNet Does Not Scale Well!
- Why? Small number of layers!

P2P Outperforms NCCL!
- Why? NCCL overhead!
How Much is the NCCL Overhead?

### Measurement
- From 16% to 32% additional overhead for NCCL compared to P2P
- Smaller workload → More overhead

### NCCL Overhead
- Different source codes: ReduceKernel and BroadcastKernel in NCCL
- Different data transfer mechanisms: P2P Direct Access in NCCL

<table>
<thead>
<tr>
<th>Network</th>
<th>Batch Size</th>
<th>(%) NCCL Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet</td>
<td>16</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>26.7</td>
</tr>
<tr>
<td>AlexNet</td>
<td>16</td>
<td>21.8</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>21.8</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>31.8</td>
</tr>
<tr>
<td>ResNet</td>
<td>16</td>
<td>20.1</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>22.9</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>19.3</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>16</td>
<td>18.7</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>17.5</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>16.2</td>
</tr>
<tr>
<td>Inception-v3</td>
<td>16</td>
<td>16.9</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>19.4</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>18.9</td>
</tr>
</tbody>
</table>
What is the Impact of Network Size on Training Time?

GoogLeNet, Inception-v3 and ResNet: Batch size of 16
- P2P: $<1.5 \times, <2.3 \times, <3 \times$ for 2, 4, and 8 GPUs, respectively
- NCCL: $<1.8 \times, <2.9 \times, <4.4 \times$ for 2, 4, and 8 GPUs, respectively

Scale Better Than LeNet!
- Why? Significantly larger!

NCCL Outperforms P2P!
- Why? Amortization of Overhead!
Why Does P2P Perform Worse Than NCCL for 8 GPU Cases?

P2P Performs Poorly!
- Asymmetric link distribution
- 2-hop data copy
- Copy + Fetch
Why Does P2P Perform Worse Than NCCL for 8 GPU Cases?

P2P Performs Poorly!
- Asymmetric link distribution
- 2-hop data copy
- Copy + Fetch
Why Does P2P Perform Worse Than NCCL for 8 GPU Cases?

P2P Performs Poorly!
- Asymmetric link distribution
- 2-hop data copy
- Copy + Fetch
What is the Impact of Batch Size on Training Time?

For Both P2P and NCCL

- Linear reduction in training time
- True for all GPU counts
- Why? The larger the batch size:
  - Fewer batches per GPU
  - More computation per batch
  - Fewer data transfers
How Do Different Stages in the Training Process Scale?

FP+BP and WU Breakdown

- **FP+BP**
  - Compute-intensive
  - Only computation and no GPU-to-GPU data transfer

- **WU**
  - Communication-intensive
  - Transfer of gradients and weights
  - Negligible amount of computation
What Is the Impact of GPU Count on FP+BP and WU?

Impact on LeNet and AlexNet

- GPU count 1 → 2: >2× improvement in FP+BP time
- GPU count 2 → 4→ 8: Non-linear decrease in the FP+BP time
  - Why? Low GPU compute utilization!
- ∼Linear decrease in WU time
What Is the Impact of Network Size on FP+BP and WU?

Impact on Larger Workloads
- Near linear speedup of FP+BP stages
  - Why?
    - Increased GPU compute utilization!
- Better speedup in WU!
  - Why?
    - More weights per layer
    - Better NVLink BW utilization!
Memory Usage

- ≤5% difference between P2P and NCCL
- GPU0 consumes additional memory!
- Pre-training Memory Usage ≈ Memory for Network Model
- Training Memory Usage ≈ Memory for Network Model + Memory for outputs
What Is the Impact of Batch Size and Network Size on Memory Usage?

Impact of Batch Size
- Negligible increase in pre-training memory usage
- A limit on the maximum batch size
  - Inception-v3: No more than 64!
  - ResNet: No more than 128!

Impact of Network Size
- Larger network $\rightarrow$ More memory
Accelerating DNN Training

Hardware-Level Improvements

- More powerful GPUs!
- More efficient interconnect network!
- More memory capacity!

Software-Level Improvements

- Reduction in NCCL overhead
- Efficient data distribution (e.g., remove data transfers in P2P)
- Improvement of high-level frameworks (such as MXNet)
- Improvement in algorithm (e.g., synchronization in SGD)
Contributions

- Comparison between two different multi-GPU communication methods for training DNNs
- Breakdown of training time into computation– and communication–intensive portion
- Demonstration of the impact of GPU memory
- Guidelines for designing future hardware and software
Profiling DNN Workloads on a Volta-based DGX-1 System

Saiful A. Mojumder¹, Marcia S Louis¹, Yifan Sun², Amir Kavyan Ziabari³, José L. Abellán⁴, John Kim⁵, David Kaeli² and Ajay Joshi¹

¹ECE Department, BU; ²ECE Department, NEU; ³AMD; ⁴CS Department, UCAM; ⁵School of EE, KAIST;

jlabellan@ucam.edu

AISTECS@HiPEAC January 21st, 2019