Fine Grained Parallelism

Nat Shineman
Types of Parallelism

- Model parallelism – distribute model across processors
  - Different processors or processor groups have different weights
- Data parallelism – distribute training data in each minibatch
  - Each processor or group of processors gets a mini-minibatch
  - Usually takes the form of sample parallelism
Terminology

• Strong scaling – solving a fixed size problem faster
  - Same global minibatch size, more GPUs → smaller per-node minibatch size
  - Avoids dealing with minibatch size and generalization
  - Reaches a point where communication overhead is too high

• Weak scaling – solve a larger problem in fixed time
  - Same per-node minibatch size, more GPUS → higher global minibatch size
  - Allows local work to high communication overhead
  - Requires careful tuning of global minibatch
LBANN

- Livermore Big Artificial Neural Network toolkit
- Distributed ANN framework for use on HPCs
  - MPI + Threads
  - Distributed processing and communication
  - Node-local threading
  - Designed for robust model parallelism with some data parallelism
  - Utilizes low latency communication and storage
    - Node-local NVRAM for staging and parallel loading
Model Parallelism

(a) Processing mini-batch 0

(b) Processing mini-batch 1
Autoencoder Tests

- 50K, 100K, and 400K neuron models
  - 9.8, 19.7 78.6 billion parameters
  - 73, 147, 293 GB
More Parallel - CNNs

Sample Parallelism

Spatial Parallelism

Channel/Filter Parallelism
Fine Grained Data Parallelism

- Sample parallelism – samples are distributed across processors
  - Traditional data parallelism
  - Limited by minibatch size
  - What if data is too large?
- Spatial parallelism – distributing individual samples across height and width
  - Near perfect weak scaling, excellent strong scaling
  - Fits larger images
- Channel/Filter parallelism – distributing individual samples across channel/filter dimensions
  - Allows distributed weight matrix, less per-node memory
  - Fits more complicated data with more channels or models with larger filters
Expressing CNNs

- Input $x$: $(N \times C \times H \times W)$
- Weights $w$: $(F \times C \times K \times K)$
- Output $y$: $(N \times F \times H \times W)$
- Gradients: $\frac{dL}{dx}$, $\frac{dL}{dw}$, $\frac{dL}{dy}$
- Use * to represent a replicated dimension
- Use ° to represent a distributed dimension
- Sample parallelism – $x[^*,^*,^*,^*,^*]$
Sample Parallelism $x[^\circ, *, *, *]$

- Independent forward and back propagation
- Global allreduce to synchronize
- Good strong scaling but limited by minibatch size
  - Max of processor per image
- Cannot reduce memory usage
  - Consider mesh-tangling 2048x2048 18 channel images
  - ~ 288MB per image
Sample Parallelism

(a) Sample parallelism
Spatial Parallelism $x[*\, *, \, °\, , \, °]$ 

- Dividing $x$ and $y$ along height and width dimension 
  - And associated gradients $\frac{dL}{dx}, \frac{dL}{dy}$ 
- Additional parallelism 
  - Multiple GPUs per image 
- Requires additional communication 
- Allows less memory usage for samples 
- Spatial dimensions must be distributed in blocks 
  - Required for local convolution 
- Border cases become difficult 
- Can be combined with spatial parallelism
Implementation Details

- Most of forward propagation is local
  - Requires remote data at borders of partition
  - Halo exchange
- Symmetric for back prop
  - Local computation of $\frac{dL}{dy}$
  - Halo exchange to compute $\frac{dL}{dx}$
  - Global allreduce to sum components of $\frac{dL}{dw}$
- Can compute $\frac{dL}{dw}$ and $\frac{dL}{dx}$ in parallel
The Halo Exchange

• Exchange remote data on the border of convolution
  - Needed for local computation
• Based on size of convolutional kernel
• More parallelism → more halo exchanges
• Most efficient when kernel size << partition size

(b) Spatial parallelism halo exchange
Evaluation

• Focused on strong scaling
• ResNet-50
  – 1.4x to 1.8x speedup
  – Expected since most layers have small spatial domain
  – Near linear weak scaling
• Artificial Mesh tangling
  – 1K and 2K images
Strong Scaling

## TABLE I

<table>
<thead>
<tr>
<th>( N )</th>
<th>1 GPU/sample</th>
<th>2 GPUs/sample</th>
<th>4 GPUs/sample</th>
<th>8 GPUs/sample</th>
<th>16 GPUs/sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.403s</td>
<td>0.2s (2.0x)</td>
<td>0.121s (3.3x)</td>
<td>0.0969s (4.4x)</td>
<td>0.066s (6.1x)</td>
</tr>
<tr>
<td>8</td>
<td>0.399s</td>
<td>0.201s (2.0x)</td>
<td>0.124s (3.2x)</td>
<td>0.0829s (4.8x)</td>
<td>0.0681s (5.9x)</td>
</tr>
<tr>
<td>16</td>
<td>0.4s</td>
<td>0.201s (2.0x)</td>
<td>0.121s (3.3x)</td>
<td>0.085s (4.7x)</td>
<td>0.0739s (5.4x)</td>
</tr>
<tr>
<td>32</td>
<td>0.401s</td>
<td>0.207s (1.9x)</td>
<td>0.123s (3.3x)</td>
<td>0.0874s (4.6x)</td>
<td>0.0794s (5.1x)</td>
</tr>
<tr>
<td>64</td>
<td>0.407s</td>
<td>0.208s (2.0x)</td>
<td>0.124s (3.3x)</td>
<td>0.0911s (4.5x)</td>
<td>0.0839s (4.8x)</td>
</tr>
<tr>
<td>128</td>
<td>0.407s</td>
<td>0.209s (1.9x)</td>
<td>0.125s (3.3x)</td>
<td>0.0931s (4.4x)</td>
<td>0.0902s (4.5x)</td>
</tr>
<tr>
<td>256</td>
<td>0.401s</td>
<td>0.209s (1.9x)</td>
<td>0.127s (3.2x)</td>
<td>0.0977s (4.1x)</td>
<td>n/a</td>
</tr>
<tr>
<td>512</td>
<td>0.393s</td>
<td>0.209s (1.9x)</td>
<td>0.126s (3.1x)</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>1024</td>
<td>0.4s</td>
<td>0.211s (1.9x)</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Note that 8 and 16 way necessitate both intra-node and inter-node communication on halo exchanges.

- 4 GPUs per node
Weak Scaling
Filter and Channel Parallelism

- Partitioning $x$ along channel dimension
- Partitioning $y$ along filter dimension
- Partition weights accordingly
- Less per node memory to store model
- Useful for complicated input data
  - Many channels or filters
- Allows segmented allreduce instead of global
  - Only across subset of processors
Stationary-\(x\)

- Input data distributed on channel and sample/spacial dimensions
  - \(x[^{\circ}, ^{\circ}, ^{\circ}, ^{\circ}]\)

- Output data distributed with relation to channels
  - \(y[^{\circ}, ^{\circ}C, ^{\circ}, ^{\circ}]\)

- Weights distributed by channel
  - \(W[^*, ^{\circ}]\)

- Communication of \(x\) avoided
  - All gather needed for \(\frac{dL}{dy}\)

\[
\begin{align*}
(1) \quad & T_1[^{\circ}, ^{*}, ^{\circ}, ^{\circ}] \leftarrow \text{ConvForward}(x[^{\circ}, ^{\circ}, ^{\circ}, ^{\circ}], w[^{*}, ^{\circ}]) \\
(2) \quad & y[^{\circ}, ^{\circ}C, ^{\circ}, ^{\circ}] \leftarrow \sum_CT_1[^{\circ}, ^{*}, ^{\circ}, ^{\circ}] \\
(3) \quad & \frac{dL}{dy}[^{\circ}, ^{*}, ^{\circ}, ^{\circ}] \leftarrow \frac{dL}{dy}[^{\circ}, ^{\circ}C, ^{\circ}, ^{\circ}] \\
(4) \quad & \frac{dL}{dx}[^{\circ}, ^{\circ}, ^{\circ}, ^{\circ}] \leftarrow \text{ConvBackData}(\frac{dL}{dy}[^{\circ}, ^{*}, ^{\circ}, ^{\circ}], w[^{*}, ^{\circ}]) \\
(5) \quad & T_3[^{*}, ^{\circ}] \leftarrow \text{ConvBackFilt}(\frac{dL}{dy}[^{\circ}, ^{*}, ^{\circ}, ^{\circ}], x[^{\circ}, ^{\circ}, ^{\circ}, ^{\circ}]) \\
(6) \quad & \frac{dL}{dw}[^{*}, ^{\circ}] \leftarrow \sum_{N,H,W}T_3[^{*}, ^{\circ}] 
\end{align*}
\]
Stationary-x
Stationary-y

- Distribute $x$ with relation to filters
  - $x^{\circ, \circ F, \circ, \circ}$
- Distribute $y$ *along* filter dimension
  - $y^{\circ, \circ, \circ, \circ}$
- Distribute weights accordingly
  - $w^{\circ, \circ}$
- Avoids communicating output
  - Allgath needed for $\frac{dL}{dx}$

\begin{align*}
(1) & \quad x^{\circ, *, \circ, \circ} \leftarrow x^{\circ, \circ \mathcal{F}, \circ, \circ} \\
(2) & \quad y^{\circ, \circ, \circ, \circ} \leftarrow \text{ConvForward}(x^{\circ, *, \circ, \circ}, w^{\circ, *}) \\
(3) & \quad T_2^{\circ, *, \circ, \circ} \leftarrow \text{ConvBackData}(\frac{dL}{dy}^{\circ, \circ, \circ, \circ}, w^{\circ, *}) \\
(4) & \quad \frac{dL}{dx}^{\circ, \circ \mathcal{F}, \circ, \circ} \leftarrow \sum_{\mathcal{F}} T_2^{\circ, *, \circ, \circ} \\
(5) & \quad T_3^{\circ, *} \leftarrow \text{ConvBackFilt}(\frac{dL}{dy}^{\circ, \circ, \circ, \circ}, x^{\circ, *, \circ, \circ}) \\
(6) & \quad \frac{dL}{dw}^{\circ, *} \leftarrow \sum_{\mathcal{N}, \mathcal{H}, \mathcal{W}} T_3^{\circ, *}. 
\end{align*}
Stationary-$w$

- Generalization of both
- Distribute both channel and filter
- Requires communication of both $x$ and $y$
Evaluation

• Training ResNet-50
  - Wide ResNet-50-2
  - Wide ResNet-50-4
    • Previously impossible to train

• ImageNet Dataset
Weak Scaling

- Sample parallelism degrades with more allreduces
- Higher channel parallelism reduces communication overhead
- Adding spatial parallelism reduces early overhead
- Combining stationary-x and -y improves performance
Large Models

Wide ResNet-50-2 scaling

Wide ResNet-50-4 scaling
Summary

- Sample parallelism
  - Has least overhead
  - Applicable in many situations

- Spatial parallelism
  - Reduces sample memory
  - Provides near perfect weak scaling
  - Can provide good strong scaling

- Channel/Filter Parallelism
  - Typically stationary-x preferred
  - Significantly increases communication overhead
  - Relieves memory pressure by distributing weights
Sources

• LBANN: livermore big artificial neural network HPC toolkit. https://dl.acm.org/citation.cfm?id=2834897

• Nikoli Dryden et al., Improving Strong-Scaling of CNN Training by Exploiting Finer-Grained Parallelism (IPDPS '19) https://arxiv.org/abs/1903.06681

• Dryden et al., Channel and Filter Parallelism for Large-Scale CNN Training (SC19) https://ndryden.com/data/papers/sc2019-chanfilt.pdf