- Varun Batra
- Why PipeDream?
- Pipeline Parallelism
  - Partitioning
  - Scheduling
  - Learning
- Implementation
- Experimentation
- Distbelief and Adam – Using Commodity Machines

- TensorFlow – Generalization and giving user the power to code

- Problem - Time and Resource consumption. Imagine billions of parameters in a word imbedding/image processing task.
- Solution – Parallelism! 10 points to Gryffindor!

- Naïve parallelism can be detrimental, as quality matters and also can blow up computation or communication overheads down the road.

- Time per pass can decrease, but number of passes increase! Accuracy/Convergence impacted.

- Total Time = Time per epoch * Number of epochs for a given accuracy.
- Training contains multiple epochs over the entire data.

- In each epoch, model trains over all the inputs in the dataset using steps.

- In each step, the current model makes a prediction from a small set of training samples called minibatch. This process is called forward pass.

- Minibatch fed to layer 1, each layer computes a function using learned parameters and passes to next layer. The final output class prediction is compared to actual value and the error is propagated back in a Backward Pass to update the weights.
- Under-Utilization
- Unknown Model Splitting Technique
As number of workers increase, the communication overhead increases.
- PipeDream

- Pipeline Parallelism = MP + DP + Pipelining
• Entire Model broken into Stages

• Each Stage mapped to a Machine that performs both backward and forward pass

• Multiple minibatches inserted together to make use of all machines.
Benefits over Data Parallelism:

- Pipelining communicates less
  - output of layer much smaller than parameter size

- Pipelining overlaps computation and communication
  - forward and backward pass has a lot of communication and computation overlap for subsequent minibatches, so, better hardware efficiency.
- Automatic Partitioning
- Scheduling
- Effective Learning
Goals

1. Each Stage performs roughly same amount of work

2. Inter-stage data communication is minimum
Profiling: Dry run the model on a single machine to estimate for each layer:

- Total Forward and Backward Computation time.
- Size of output activation and input gradients.
- Size of parameters
Partitioning Algorithm:
Computes:
- Partitioning of layers into stages
- Replication Factor for each stage
- Minibatches to keep pipeline busy

Goal is Minimize the Overall Time in the Pipeline System
ie. Minimizing the time for the slowest stage.
Let $T(i \rightarrow j, m)$ denote the time taken by a single stage spanning layers $i$ through $j$, replicated over $m$ machines.

Let $A(j, m)$ denote the time taken by the slowest stage between layers 1 and $j$ using $m$ machines.

Goal – Find $A(N, M)$, and the corresponding partitioning where $N$ is the number of layers and $M$ is the number of Machines.

\[
T(i \rightarrow j, m) = \frac{1}{m} \max \left( \sum_{l=i}^{j} T_l, \sum_{l=i}^{j} W^m_l \right)
\]

1. $A(j, m) = T(1 \rightarrow j, m)$
2. $A(j, m) = \min_{1 \leq i < j} \min_{1 \leq m' < m} \max \left\{ A(i, m-m'), 2 \cdot C_i \right\}$

Initialization. $A(1, m) := T(1 \rightarrow 1, m)$, where $T(.)$ is as defined above, and $m$ is varied from 1 through $M$ (the total number of machines). $A(i, 1) := T(1 \rightarrow i, 1)$, where $i$ is varied from 1 through $N$ (the total number of layers in the model).
Alternate between Forward and Backward Work – 1F1B
- Mixing of Forward and Backward passes with different versions of parameters can lead to incorrect/slow learning.

- Weight Stashing – Maintaining multiple versions of weight for Forward and Backward pass in a stage. In Forward – Use latest version, in Backward – use the corresponding version

- Vertical Sync – After performing the backward pass of a minibatch using an older version, each stage applies latest updates to use new weights.
- Initialization Step
- Parameter State
- Intermediate State
- Checkpointing
- Cluster A – Fast Network, Slow GPU
- Cluster B – Fast GPU, Slow Network

(a) VGG16

(b) Inception-v3
<table>
<thead>
<tr>
<th>DNN Model</th>
<th># Machines (Cluster)</th>
<th>BSP speedup over 1 machine</th>
<th>PipeDream speedup over 1 machine</th>
<th>PipeDream speedup over BSP</th>
<th>PipeDream communication reduction over BSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>4 (A)</td>
<td>1.47×</td>
<td>3.14×</td>
<td>2.13×</td>
<td>90%</td>
</tr>
<tr>
<td></td>
<td>8 (A)</td>
<td>2.35×</td>
<td>7.04×</td>
<td>2.99×</td>
<td>95%</td>
</tr>
<tr>
<td></td>
<td>16 (A)</td>
<td>3.28×</td>
<td>9.86×</td>
<td>3.00×</td>
<td>91%</td>
</tr>
<tr>
<td></td>
<td>8 (B)</td>
<td>1.36×</td>
<td>6.98×</td>
<td>5.12×</td>
<td>95%</td>
</tr>
<tr>
<td>Inception-v3</td>
<td>8 (A)</td>
<td>7.66×</td>
<td>7.66×</td>
<td>1.00×</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>8 (B)</td>
<td>4.74×</td>
<td>6.88×</td>
<td>1.45×</td>
<td>47%</td>
</tr>
<tr>
<td>S2VT</td>
<td>4 (A)</td>
<td>1.10×</td>
<td>3.34×</td>
<td>3.01×</td>
<td>95%</td>
</tr>
</tbody>
</table>