Large Scale Artificial Neural Network Training Using Multi-GPUs

Introduction

1. Artificial Neural Network (ANN) is widely used in a variety of data analysis

Motivation

1. Benefit from large scale
   • Reducing the approximation error
     In the field of function approximation, the integrated squared error of approximation, integrating on a bounded subset of variables, is bounded by $\int f \, dc$, where $c$ depends on a norm of the Fourier transform of the function being approximated [1]. Therefore, the approximation error can be reduced by increasing the network size.
   • Increasing the model capabilities
     In the field of natural language processing, the output nodes in a neural network are usually required to be equal to the size of vocabulary [2], which often exceeds 10,000. A larger network enables itself to recognize bigger vocabulary.
   • Improving the accuracy of forecasting
     In the field of forecasting, the number of input nodes in an ANN tie to the number of variables in a predicting model. A larger network can improve the forecasting accuracy by incorporating more variables into the model.

2. The challenge of training a large scale ANN
   • Large scale ANN training is compute intensive For a network with 10^4 output nodes and 10^8 hidden nodes, it has at least 10^4 tuning parameters. Training of such a network may take hundreds of hours on a CPU.
   • Large scale ANN training is parameter intensive Although a GPU can accelerate the training phase 10x faster, the ANN parameters have to fit into the fixed and limited GPU memory.

Reducing ANN Training to Matrix Multiplication

1. Forward pass
   $A_{L-1}(BATCH, n) = W_{L-1}(BATCH, n) X_{L-1}(BATCH, n) + B_{L-1}(BATCH, n)$
   $O_{L-1}(BATCH, n) = f(A_{L-1}(BATCH, n))$

2. Backward pass
   $\frac{dE}{dA_{L-1}(BATCH, n)} = \frac{dE}{dO_{L-1}(BATCH, n)} \cdot \frac{dO_{L-1}(BATCH, n)}{dA_{L-1}(BATCH, n)}$
   $\frac{dE}{dW_{L-1}(BATCH, n)} = \frac{dE}{dA_{L-1}(BATCH, n)} \cdot X_{L-1}(BATCH, n)$
   $\frac{dE}{dB_{L-1}(BATCH, n)} = \frac{dE}{dA_{L-1}(BATCH, n)} \cdot 1$

3. Amortizing high frequency memory allocation and de-allocation

4. A simplified dynamic task scheduling runtime for matrix multiplication

5. Tile reuse by multi-GPU cache coherence

Experimental Setup

<table>
<thead>
<tr>
<th>System Configuration</th>
<th>GPU</th>
<th>CPU</th>
<th>RAM</th>
<th>CPU BLAS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 NVIDIA TESLA K40c + 2 TITAN X</td>
<td>Xeon ES 2637 V3</td>
<td>128 GB DDR5</td>
<td>OpenBLAS + 1.13</td>
</tr>
</tbody>
</table>

Performance:

- Speed Up W.R.T Caffe’s CPU
- Speed Up W.R.T Caffe’s in-core GPU
- Speed Up W.R.T Caffe’s in-core GPU

- Caffe’s CPU | 20x |
- Caffe’s GPU | 10x |
- Our 2 K40c + 2 TITAN X | 120x |
- Caffe’s GPU | N/A |

- Caffe’s GPU | 60x |
- Our 2 K40c + 2 TITAN X | 120x |
- Caffe’s GPU | N/A |

Citation