Accelerating DNN Inference with GraphBLAS and the GPU

Xiaoyun Wang, Zhongyi Lin, Carl Yang, John Owens
UC Davis

Our Contributions

- We use GPU and GraphBLAST for Sparse DNN inference;
- Transposed FC layer:

$$Y_{(l+1)}^T = W^T Y_{(l)}^T + b \quad \text{Other than} \quad Y_{(l+1)} = Y_{(l)} W + b$$

- Filter out 0s after each layer, maintaining sparsity of activation matrices to 3% matrix fill.

Motivations

Why we want to use GPU

Graph problems are often bandwidth limited. GPUs provide greater achievable bandwidth on problem like sparse MxM. For example the V100 GPU has a peak bandwidth of 900 GB/s

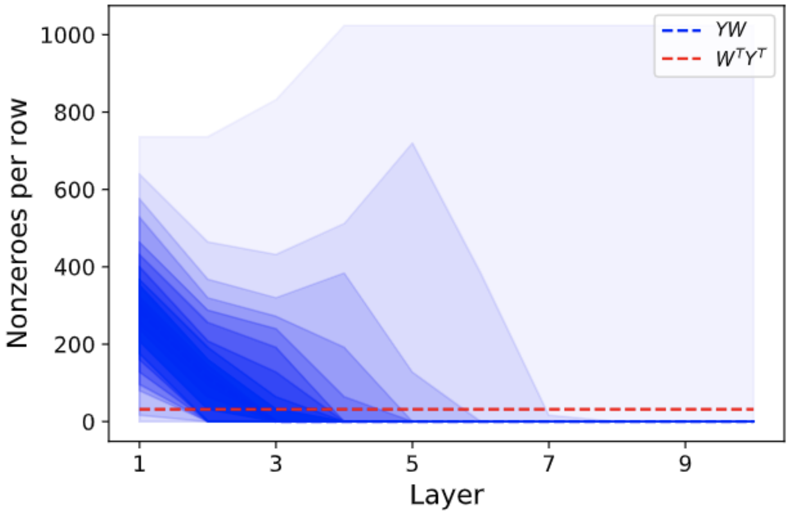
Why GraphBLAST

GraphBLAST is a GraphBLAS implementation on GPU, which is the first GraphBLAS implementation to match the state-of-art in high-performance graph processing.

Why it is a right tool

GraphBLAST provides high-performance operators required to implement Sparse DNN inference

Why transposed FC works



Nonzeroes per row in the activation matrix in each layer of a 1024-neuron, 120-layer neural network.

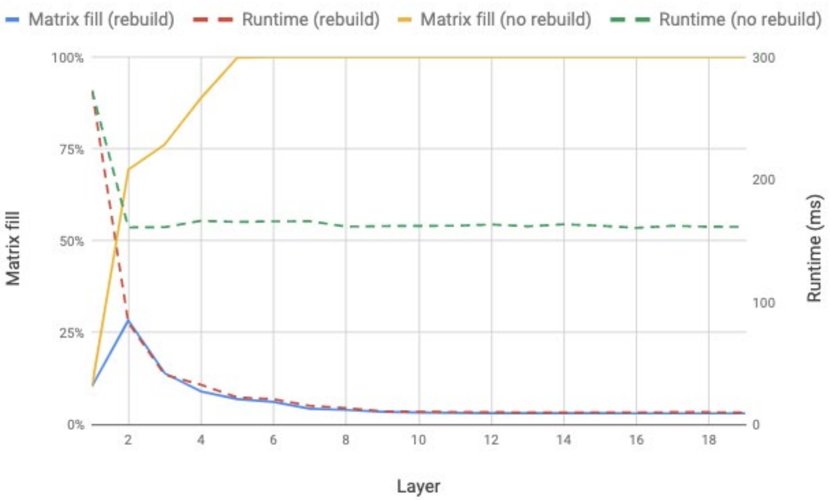
Load Balance:

W^T is always 32 nonzeroes per row, so there is no load imbalance

Why the weight matrix has such a characteristic:

- Kronecker Product and choice of W* in Radix-Net.
- The number of nonzeroes of sparse weight matrices is determined by the original dense matrix.

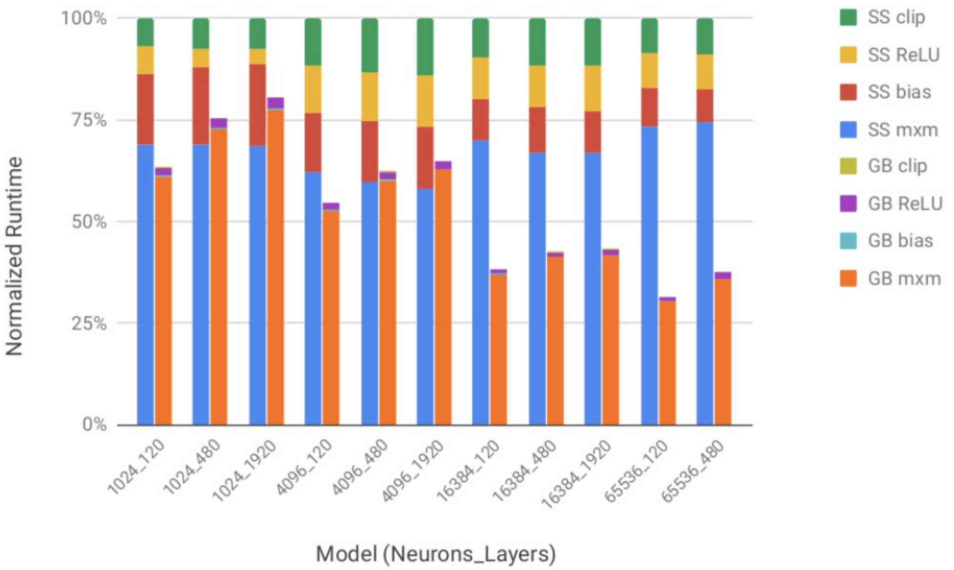
Why filtering out zeroes works



Filter Out Zeroes after each layer:

- Keep Sparsity of matrix;
- Reduce the computational cost of computing on zeroes in the next layer;

Performance Analysis of Each Operation



Non-matrix operators 16.6X speedup

- Add bias: 59.2X
- Clipping: 62.1X
- Relu: 5.44X

Sparse matrix multiplications

- 2X speedup over CPU
- Future works are needed investigate the performance