Write Quick, Run Fast: Sparse Deep Neural Network in 20 Minutes of Development Time via SuiteSparse:GraphBLAS

Tim Davis, Mohsen Aznaveh, Scott Kolodziej
Texas A&M University

Sept 25, 2019

HPEC'19 presentation

Sparse DNN in GraphBLAS

function name	description	GraphBLAS notation
`GrB_mxm`	matrix-matrix mult.	$\mathbf{C}\langle\mathbf{M}\rangle = \mathbf{C} \odot \mathbf{AB}$
`GrB_apply`	apply unary op.	$\mathbf{C}\langle\mathbf{M}\rangle = \mathbf{C} \odot f(\mathbf{A})$ $\mathbf{w}\langle\mathbf{m}\rangle = \mathbf{w} \odot f(\mathbf{u})$
`GxB_select`	apply select op.	$\mathbf{C}\langle\mathbf{M}\rangle = \mathbf{C} \odot f(\mathbf{A}, \mathbf{k})$

- Each GraphBLAS operation uses OpenMP parallelism, internally
- This took 20 minutes to write (almost as simple as MATLAB reference):

```
1     for (int layer = 0 ; layer < nlayers ; layer++)
2     {
3         // Y = Y * W [layer]
4         GrB_mxm (Y, NULL, NULL, GxB_PLUS_TIMES_FP32, Y, W [layer], NULL) ;
5         // Y(i,j) += Bias [layer] (j,j) for each Y(i,j)
6         GrB_mxm (Y, NULL, NULL, GxB_PLUS_PLUS_FP3 2, Y, Bias [layer], NULL) ;
7         // delete entries; keep only those > 0
8         GxB_select (Y, NULL, NULL, GxB_GT_ZERO, Y, NULL, NULL) ;
9         // threshold maximum values: Y (Y > 32) = 32
10        GrB_apply (Y, NULL, NULL, ymax, Y, NULL) ;
11    }
```

Sparse DNN in MATLAB: reference, and with GraphBLAS

```
function Y = dnn_matlab (W, bias, Y0)
% MATLAB reference solution
Y = Y0 ;
for i=1:length(W)
    % Propagate through layer.
    Z = Y * W {i} ;
    % Apply bias to non-zero entries.
    Y = Z + (double(logical(Z)).*bias{i}) ;
    % Threshold negative values.
    Y (Y < 0) = 0 ;
    % Threshold maximum values.
    Y (Y > 32) = 32 ;
end
```

```
function Y = dnn (W, bias, Y0)
% SuiteSparse:GraphBLAS MATLAB interface
Y = Y0 ;
for k = 1:length(W)
    % Propagate through layer, apply bias
    % and threshold negative values.
    Y = gb.select ('>0', gb.mxm ('+.+', ...
        Y * W {k}, bias {k})) ;
    Y (Y > 32) = 32 ;
end
```

SuiteSparse:GraphBLAS performance

- OpenMP parallelism inside SuiteSparse:GraphBLAS
- MPI: difficult to load balance
- Results for 4 machines:
  - NVIDIA DGX Station (20-core Intel)
    - MATLAB: reference solution
    - DGX+C: with C API to GraphBLAS
    - DGX+M: with MATLAB interface to GraphBLAS
  - IBM Power8 (20x8 cores)
  - IBM Power9 (40x4 cores)
  - Intel Xeon (34 cores, with MPI)

SuiteSparse:GraphBLAS single-threaded performance

Single threaded performance (rate, in billion edges/sec, higher is better)												
Neurons:	1K			4K			16K			64K		
Layers:	120	480	1920	120	480	1920	120	480	1920	120	480	1920
MATLAB	2	2	2	2	2	2	2	2	2	2	2	1.5
in SuiteSparse:GraphBLAS:												
DGX+C	11	16	19	10	13	14	5	6	6	4	5	5
DGX+M	10	14	16	10	13	14	5	6	6	4	4	4

GraphBLAS about 2x to 8x faster than pure MATLAB with a single thread. Little performance lost when using GraphBLAS via its MATLAB interface (DGX+M).

SuiteSparse:GraphBLAS parallel performance

OpenMP and/or MPI performance (rate; higher is better)												
DGX+C	**153**	**240**	**275**	100	127	143	75	83	88	60	68	65
DGX+M	86	101	110	95	123	129	65	75	77	59	62	63
Power8	111	139	150	125	164	183	102	112	114	**105**	**109**	**113**
Power9	119	177	190	**139**	**174**	**186**	**105**	**116**	101	90	-	-
Xeon	58	68	71	115	140	150	99	115	**120**	81	89	-

Highest performance in bold. GraphBLAS is 60x to 120x faster than the pure MATLAB reference solution, on the same platform (DGX). Normally little performance lost when using GraphBLAS via its MATLAB interface (DGX+M).