ML/DL Micro-benchmark Suite

Contact: Prasanna Balaprakash (pbalapra@anl.gov), Venkat Vishwanath (venkat@anl.gov), Kalyan Kumaran (kumaran@anl.gov), ALCF, Argonne National Laboratory

Summary Version

1.0

Purpose of Benchmark

Convolutions, single and half precision GEMM, FFT, and other machine/deep learning math algorithms not included in other CORAL benchmark suites.

Characteristics of Benchmark

ML/DL micro-benchmark suite consists of a subset of kernels and input sizes from DeepBench. This comprises a set of basic operations (dense, sparse matrix multiplications, convolutions as well as some recurrent layer types) for training and inference. In addition, the benchmark suite also includes FFT. FFTs consists of 1D and 2D FFT kernels.

Mechanics of Building and running the Deepbench Benchmark

Get the code by doing:
git clone https://github.com/baidu-research/DeepBench

See the detailed build and run instructions at the following URL:
https://github.com/baidu-research/DeepBench#getting-the-code

Mechanics of Building and Running FFT

Vendors are free to use their own FFT source code and library of their choice.

Reporting Rules

For the DeepBench Benchmarks:
Base single precision run: Suite is run in single precision. Calls to vendor’s optimized library is allowed.

Optimized reduced precision run: Vendors can run in reduced precision to obtain better results. The precision must be reported and the version of the routines must be supported in a library.

For FFT:
Base run: Report the time and Gflops to compute the DFT using FFT for R2C for various 1D and 2D FFT kernel. Report the performance for batch size of 1 for single precision and double precision. Report the performance for the largest batch size one can run on a node. Calls to vendor’s optimized library is allowed.

Optimized reduced precision run: Vendors can run in reduced precision to obtain better results. The precision must be reported and the version of the routines must be supported in a library.

A. Deepbench Training

1) Dense Matrix Multiplication					
M	N	K	A Transpose	B Transpose	Time (usec)
4096	7000	4096	N	N	
35	8457	4096	T	N	
6144	16	2048	T	N	

2) Convolution												
W (input - time)	H (input)	C (channels)	N (batch size)	K (number of filters)	S (filter width)	R (filter height)	pad_w	pad_h	Horizontal Stride	Vertical Stride	fwd_time (usec)	bwd_inputs_time (usec)
224	224	3	8	64	3	3	1	1	1	1		

224	224	3	16	64	7	7	3	3	2	2		
14	14	1024	16	2048	1	1	0	0	2	2		
3) Recurrent Layers - LSTM												

Input	N	Timesteps	Time Forward (usec)	Time Backward (usec)	Geomean Time(usec)
4096	128	25			
4) Recurrent Layers - GRU					
Hidden units	N	Timesteps	Time Forward (usec)	Time Backward (usec)	Geomean Time(usec)
1024	64	1500			

B. DeepBench Inference

1) Dense Matrix Multiplication					
M	N	K	A Transpose	B Transpose	Time (usec)
5124	1500	2560	N	N	
8448	4	2816	N	N	

1024	4	512	N	N	

2) Sparse Matrix Multiplication

M	N	K	A Transpose	B Transpose	Sparsity	Sparse time (usec)	Dense time (usec)	Geomean Time (usec)
10752	2	3584	N	N	0.95			
10752	3000	3584	N	N	0.95			
7680	1500	2560	N	N	0.9			

3) Convolution

W (input - time)	H (input)	C (channels)	N (batch size)	K (number of filters)	S (filter width)	R (filter height)	pad_w	pad_h	Horizontal Stride	Vertical Stride	Time Forward (usec)
700	161	1	4	32	20	5	0	0	2	2	
224	224	3	1	64	7	7	3	3	2	2	
14	14	1024	2	2048	1	1	0	0	2	2	

4) Recurrent Layers - LSTM

Input	N	Timesteps	Time Forward (usec)
1536	4	50	

5) Recurrent Layers - GRU

Hidden units	N	Times teps	Time Forward (usec)
2816	4	1500	

C. FFT for DFT (r2c)

Dims	Floating Point Precision	Batch Size	Time (s)	Gflop/s
1024	Single	1		
	Single	X (enter the largest batch size on a node)		
1024	Double	1		
	Double	X		
4096	Single	1		
	Single	X		
	Double	1		
	Double	X		
32x32	Single	1		
	Single	X		
	Double	1		
	Double	X		
1024 X 1024	Single	1		

	Single	X		
	Double	1		
	Double	X		
4096x4096	Single	1		
	Single	X		
	Double	1		
	Double	X		