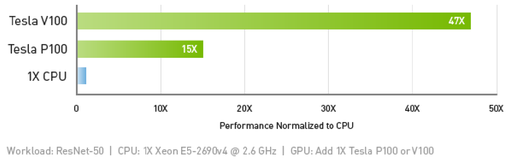
Gradient Descent Algorithms on Modern Hardware

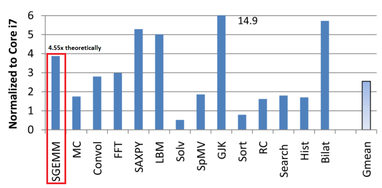
Yujing Ma, Florin Rusu, Martin Torres
University of California Merced

Motivation

- 47x higher than CPU on deep learning inference (https://www.nvidia.com/en-us/data-center/tesla-v100/)



- In open-source TensorFlow simple placer: GPU are preferred over CPU.
- GPU is not always superior and GPU has lower speedup than the theoretical value. [Lee, Victor W., et al. "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU."]



Problem & Contribution

- in-depth comparison on *which GD algorithm performs better on which architecture and with what type of data*
  1. GD algorithm (model update): sync, async
  2. computing architecture: CPU, GPU
  3. data sparsity: dense, sparse

Gradient Descent

- Batch Gradient Descent (Parallel BGD)

  1. Compute gradient: **in parallel**
     for $i = 1$ **to** N **do** $\vec{g} \leftarrow \vec{g} + \overrightarrow{\nabla f}(\vec{w}; \vec{x}_i, y_i)$
  2. Update model: $\vec{w} \leftarrow \vec{w} - \alpha\vec{g}$

- Mini-Batch Gradient Descent (Hogbatch)

  for each batch: **in parallel**
  1. Compute gradient: **in parallel**
     for $i = 1$ **to** B **do** $\vec{g} \leftarrow \vec{g} + \overrightarrow{\nabla f}(\vec{w}; \vec{x}_i, y_i)$
  2. Update model: $\vec{w} \leftarrow \vec{w} - \alpha\vec{g}$

- Stochastic Gradient Descent (Hogwild)

  1. **for** $i = 1$ **to** N **do** **in parallel**
  2. Compute gradient estimate: $\vec{g} \leftarrow \overrightarrow{\nabla f}(\vec{w}; \vec{x}_i, y_i)$
  3. Update model: $\vec{w} \leftarrow \vec{w} - \alpha\vec{g}$
  4. **end for**

Datasets

dataset	#examples	#features	#NNZ/example (avg)	*sparse size	dense size
covtype	581,012	54	54 (54)	485 MB	485 MB
w8a	64,700	300	0 to 114 (12)	4.4 MB	155 MB
rcv1	677,399	47,236	4 to 1,224 (73)	1.2 GB	256 GB
news	19,996	1,355,191	1 to 16,423 (455)	134 MB	217 GB

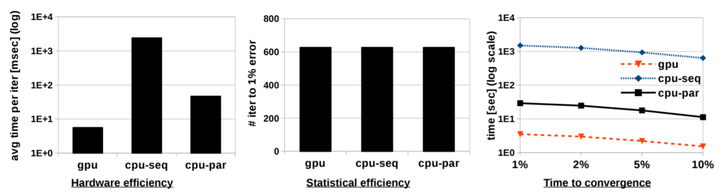
* compressed sparse row (CSR) used for sparse representation

System Specification

- CPU: 2 x Intel Xeon E5-2660 v4
  2*14 cores, 2*28 threads, max turbo frequency: 3.20 GHz
- GPU: NVIDIA Tesla K80
  2*2,496 cores (13 MP * 192 cores/MP), boost clock: 876 MHz

Synchronous GD

- *news, logistic regression*



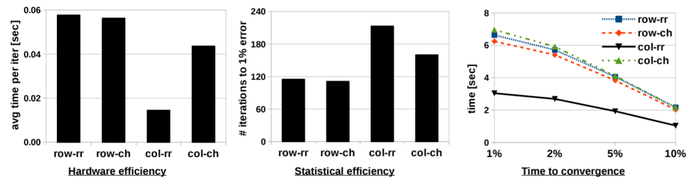
- synchronous GD performance to 1% error

task	dataset	time-to-convergence (sec) gpu	cpu-seq	cpu-par	time-per-iteration (msec) gpu	cpu-seq	cpu-par	epochs	speedup cpu-seq/cpu-par	cpu-par/gpu
LR	covtype	1.05	145.11	1.29	15	2,073	18.42	70	112.54	1.23
	w8a	0.37	148.88	0.46	4.87	1,959	6.05	76	323.80	1.24
	real-sim	3.10	1,537.90	7.67	4.43	2,197	10.96	700	200.46	2.47
	rcv1	31.69	2,227.05	48.06	44.82	3,150	67.98	707	46.34	1.52
	news	0.65	240.21	3.68	6.37	2,355	36.08	102	65.27	5.66
SVM	covtype	10.22	1,344.65	13.50	14.27	1,878	18.85	716	99.63	1.32
	w8a	0.78	342.85	0.80	4.13	1,814	4.23	189	428.84	1.02
	real-sim	0.23	75.59	0.46	6.22	2,043	12.43	37	164.36	2.00
	rcv1	1.13	111.61	2.61	29.74	2,937	68.69	38	42.76	2.31
	news	0.30	98.42	1.69	6.67	2,187	37.56	45	58.23	5.63
MLP	covtype	1,498	19,398	10,009	919	11,908	6,145	1,629	1.94	6.68
	w8a	83.57	909	388	107	1,161	495	783	2.34	4.64
	real-sim	21.99	229	93.98	130	1,365	556	168	2.46	4.26
	rcv1	48.91	1,146	241	1,193	16,960	5,880	41	2.89	4.93
	news	4.03	35.04	16.08	40.23	357	164	98	2.17	4.08

GPU has better hardware efficiency and faster time to convergence. When the dataset is more sparse, the speedup of gpu over cpu-par is higher.

Asynchronous GD

- *covtype, logistic regression*



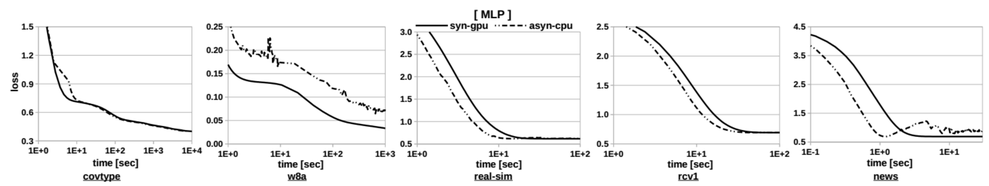
Asynchronous GD (cont.)

- asynchronous GD performance to 1% error (logistic regression, support vector machine, multilayer perceptron)

task	dataset	time-to-convergence (sec) gpu	cpu-seq	cpu-par	time-per-iteration (msec) gpu	cpu-seq	cpu-par	epochs gpu	cpu-seq	cpu-par	speedup cpu-seq/cpu-par	gpu/cpu-par
LR	covtype	1.97	0.60	1.51	15	150	251	135	4	6	0.60	0.06
	w8a	0.22	0.27	0.18	2.8	15	5.9	80	18	27	2.54	0.47
	real-sim	2.48	1.35	0.52	27	25	8.1	92	54	61	3.09	3.33
	rcv1	18.29	20.37	4.64	226	345	71	81	59	65	4.86	3.18
	news	∞	5.47	∞	65	53	8.7	∞	103	∞	6.09	7.47
SVM	covtype	0.96	0.16	0.35	15	53	77	63	3	4	0.69	0.19
	w8a	∞	0.54	1.89	2.6	2.2	5.6	∞	239	333	0.39	1.18
	real-sim	3.46	1.82	1.28	14	11	7.6	247	164	166	1.45	1.84
	rcv1	10.25	22.71	7.57	94	216	68	109	105	111	3.18	1.38
	news	∞	20.01	1.79	50	47	8.4	∞	425	211	5.60	5.95
MLP	covtype	2,106	6,365	288	6,056	19,058	814	344	334	354	23.42	7.44
	w8a	495	1,284	986	635	1,668	92.61	776	770	10,635	18.01	6.85
	real-sim	140	317	11.14	715	1,925	107	196	165	108	18.04	6.70
	rcv1	352	724	34.47	8,326	17,234	858	42	42	40	20.08	9.70
	news	18.25	47.35	1.12	234	512	34.04	78	91	32	15.06	6.87

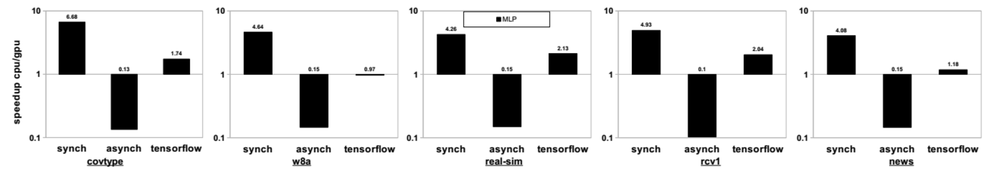
CPU is faster than GPU in time to convergence. On dense and low-dimensional data, cpu-seq is faster. On sparse data, cpu-par dominates.

Sync GD vs. Async GD



comparison between BGD and SGD: The best choice is particular to the task and the dataset.

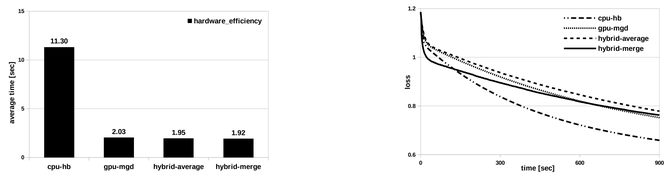
CPU vs. GPU



On sync GD, optimized GPU kernels provide better hardware efficiency than TensorFlow. On async GD, the performance of GPU is worse than CPU about 10 times.

Hybrid GD

- *covtype, multilayer perceptron*



{yma33, frusu, mtorres58}@ucmerced.edu