Asynchronous Stochastic Gradient Descent on GPU: Is It Really Better than CPU?

Florin Rusu, Yujing Ma
University of California Merced

Stochastic Gradient Descent (SGD)

Algorithm 1 Stochastic Gradient Descent (SGD)

Require:

Training examples $\vec{\vec{X}} \in \mathbb{R}^{N \times d}$ and their labels $\vec{Y} \in \mathbb{R}^N$

Loss function f and its gradient $\overrightarrow{\nabla f}$

Initial model $\vec{w} \in \mathbb{R}^d$ and step size $\alpha \in \mathbb{R}$

Number of epochs t and batch size B

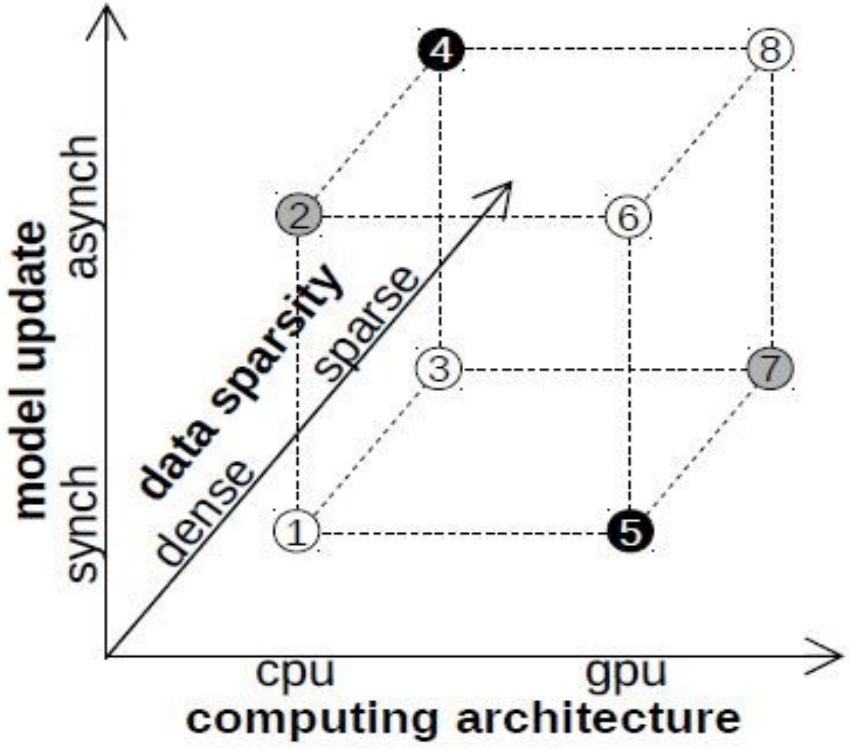
1. **for** $k = 1$ **to** t **do**

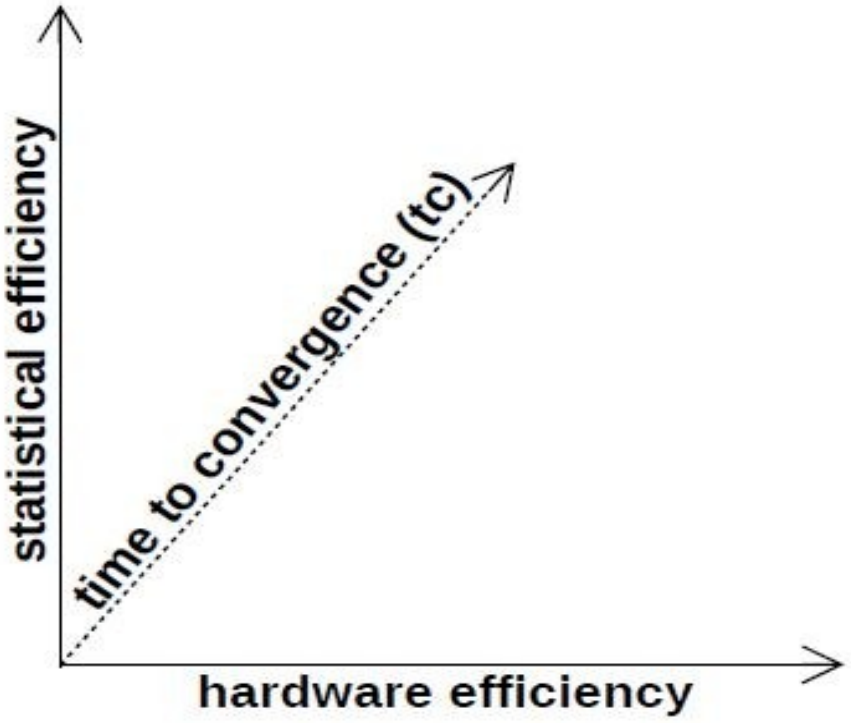
OPTIMIZATION EPOCH

2. Select a random subset of B examples $\vec{\vec{X}}_k = \{\vec{x}_{i_1}, \ldots, \vec{x}_{i_B}\}$ and their labels $\vec{Y}_k = \{y_{i_1}, \ldots, y_{i_B}\}$
3. Compute gradient estimate: $\vec{g} \leftarrow \sum_{\vec{\vec{X}}_k, \vec{Y}_k} \overrightarrow{\nabla f}(\vec{w})$
4. Update model: $\vec{w} \leftarrow \vec{w} - \alpha \vec{g}$
5. **end for**
6. **return** $\vec{w}$

SGD Study

Yujing Ma, Florin Rusu, and Martin Torres: *Stochastic Gradient Descent on Highly-Parallel Architectures*, arXiv abs/1802.08800, 2018.





Exploratory axes **Performance axes**

NUMA CPU vs GPU Architecture

	NUMA	GPU
CPU/MP	2	13
cores	14 per CPU	192 per MP
blocks	-	16 per MP
threads	28 per CPU	2048 per MP
L1 cache	32+32 KB	48 KB
L2 cache	256 KB	1.5 MB
L3 /shared	35 MB	48 KB
RAM/global	256 GB	12 GB

- CPU: 2 x Intel Xeon E5-2660 (14 cores, 28 threads)
- GPU: Tesla K80 (use only one multiprocessor, ~K40)

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	1 to 114 (11.65)	4.4 MB	155 MB
real-sim	72,309	20,958	1 to 3,484 (51.30)	87 MB	12.1 GB
rcv1	677,399	47,236	4 to 1,224 (73.16)	1.2 GB	256 GB
news	19,996	1,355,191	1 to 16,423 (454.99)	134 MB	217 GB

Speedup of Synchronous SGD

task	dataset	cpu-seq/cpu-par	cpu-par/gpu
LR	covtype	112.54	1.23
	w8a	323.80	1.24
	real-sim	200.46	2.47
	rcv1	46.34	1.52
	news	65.27	5.66
SVM	covtype	99.63	1.32
	w8a	428.84	1.03
	real-sim	164.36	2.00
	rcv1	42.76	2.31
	news	58.23	5.63

Speedup of Asynchronous SGD

task	dataset	cpu-seq/cpu-par	cpu-par/gpu
LR	covtype	0.60	5.80
	w8a	2.54	2.11
	real-sim	3.09	0.30
	rcv1	4.86	0.31
	news	6.09	0.13
SVM	covtype	0.69	5.13
	w8a	0.39	2.15
	real-sim	1.45	0.54
	rcv1	3.18	0.72
	news	5.60	0.17

Takeaways

- Synchronous SGD
  - GPU is faster than CPU, however the gap is only 1-2X and at most 5X on sparse data
  - Results are dependent on linear algebra library, e.g., ViennaCL has inefficient matrix transpose on GPU
- Asynchronous SGD
  - GPU is faster only on dense data, while CPU is as much as 7X faster on sparse data and large models
  - Efficient execution on GPU requires in-depth optimization across data access path, and data and model replication