Practical Efficiency of Asynchronous Stochastic Gradient Descent

Onkar Bhardwaj, *Guojing Cong*

IBM TJ Watson Research Center
1101 Kitchawan Road, Yorktown Heights, NY, 10598

November, 2016

ASGD is popular in current studies and applications

- Derived from stochastic gradient descent (SGD)
- Reduces synchronization and communication overhead by tolerating stale gradient updates
- Recent analyses show ASGD converges with linear asymptotic speedup over SGD
- Examples: *Downpour* and *EAMSGD*

ASGD in action

Each learner asynchronously repeats the following:

- **Pull:** Get the parameters from the server
- **Compute:** Compute the gradient with respect to randomly selected mini-batch (i.e., a certain number of samples from the dataset)
- **Push and update:** Communicate the gradient to the server. Server then updates the parameters by subtracting this newly communicated gradient multiplied by the learning rate

Practical efficiency

- Communication overhead
- Practical learning rates (and other parameters)
- Number of samples needed to reach target accuracies

Experiments and observations of ASGD

Datasets: ASGD with two different data sets: *CIFAR-10* and an in-house natural language processing data set from the finance industry – *NLC-F*.
Platform: IBM Power8 with an OSS high-density compute accelerator – 8 NVIDIA Tesla K80 GPUs connected by PCIe switches forming a binary tree. The host contains two Power8 chips, each with 12 cores
Implementation: *Downpour*: the learners are run on the GPUs, and the (sharded) parameter server is run on the host Power8 CPUs

Communication overhead

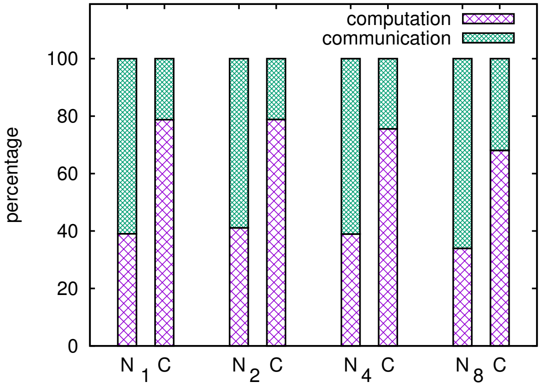


Figure: Breakdown of epoch time

Convergence

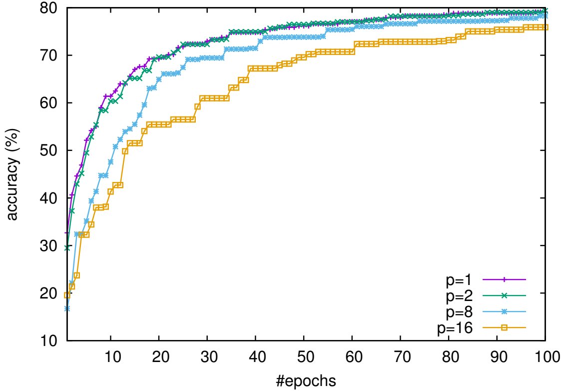


Figure: *Downpour* convergence for CIFAR10 with $\gamma = 0.1$

Notations

x_1	$:=$	The initial parameter vector for ASGD
x^*	$:=$	A local optima towards which ASGD proceeds
D_f	$:=$	$f(x_1) - f(x^*)$
M	$:=$	Mini-batch size
z	$:=$	a randomly selected minibatch
K	$:=$	No. of mini-batches processed (or ASGD updates)
p	$:=$	Number of learners
γ	$:=$	Learning rate

Why slower convergence?

Perhaps the convergence assumptions do not hold ?:

- Unbiased gradient: partial gradient $G(x, z)$ of $f(\cdot)$ is an unbiased estimator of true gradient, i.e., $\mathbb{E}(G(x, z)) = \nabla f(x)$
- Bounded variance: the variance of partial gradient with respect to randomly selected mini-batches is bounded, i.e., $\mathbb{E}(\|G(x, z) - \nabla f(x)\|^2) \leq \sigma^2$
- Lipschitzian gradient: there exists a constant L such that $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|$ for any two parameter vectors x, y

Learning rate!

- Learning assumed in convergence (linear speedup) analysis is: $\sqrt{\frac{D_f}{MKL\sigma^2}}$
- Compute this projected learning rate:
  - Upper bound on gradient variance is estimated as the maximum of observed gradient variance
  - D_f as $f(x_1)$ and used $MK = 500,000$ (MK is the total number of samples processed)
  - γ=0.005, not 0.1

With the predicted learning rate

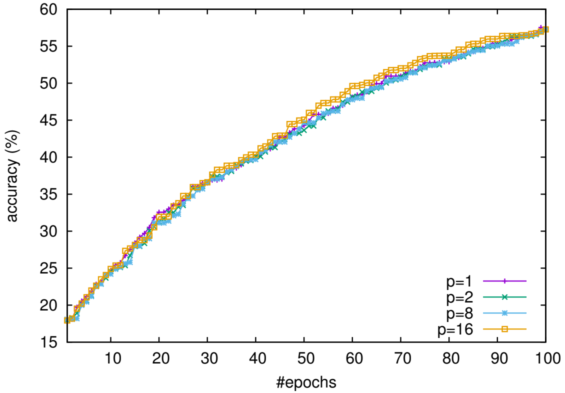


Figure: ASGD convergence for *CIFAR-10* with $\gamma = 0.005$

Sample complexity

Let $\bar{R}_K$ denote the average expected gradient norm after the first K updates of ASGD, then from Theorem 1 in [**LHL15**]

$$\begin{aligned} \bar{R}_K &\leq \frac{2D_f}{MK\gamma} + \sigma^2 L\gamma + 2\sigma^2 L^2 M p \gamma^2 && (1) \\ \text{s.t.} \quad & LM\gamma + 2L^2M^2p^2\gamma^2 \leq 1 && (2) \end{aligned}$$

Theorem

Let $p > 1$ be the number of learners and let $\alpha = \sqrt{\frac{K\sigma^2}{MLD_f}} \leq p$, then the optimal ASGD convergence rate guarantee for 1 learner and p learners can differ by a factor of approximately $\frac{p}{\alpha}$.

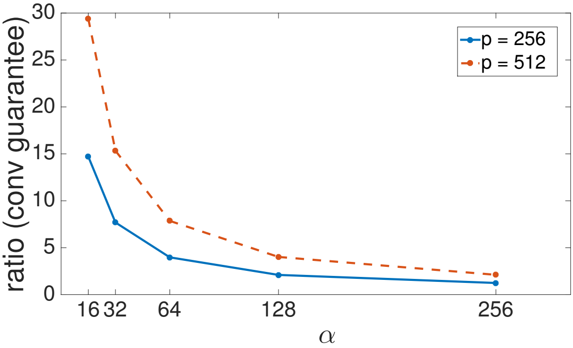
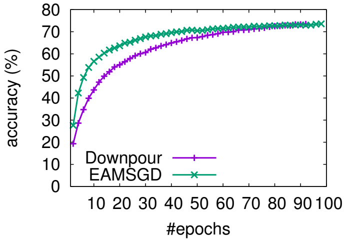


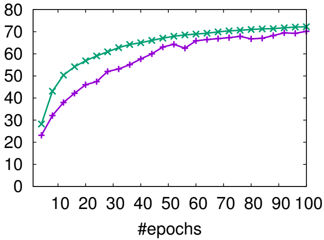
Figure: The ratio of convergence rate guarantees obtained for various values of α and p

α is a measure of (square root of) the number of mini-batches processed.

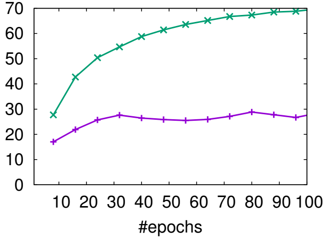
CIFAR-10 train accuracy



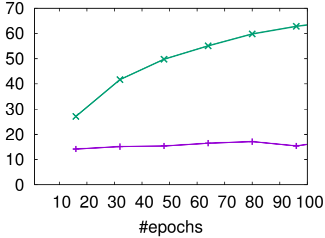
(a) $p = 2$



(b) $p = 4$

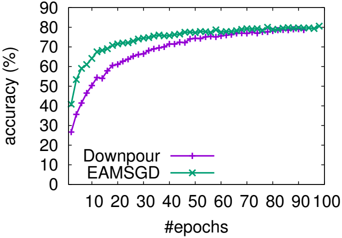


(c) $p = 8$

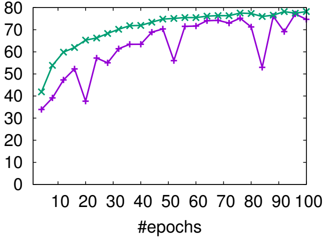


(d) $p = 16$

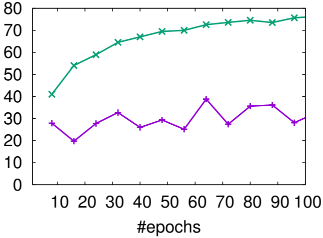
CIFAR-10 test accuracy



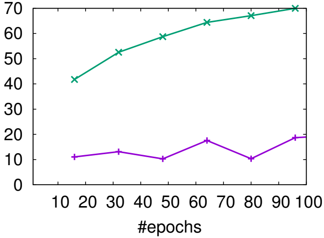
(e) $p = 2$



(f) $p = 4$

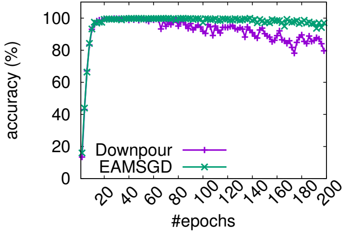


(g) $p = 8$

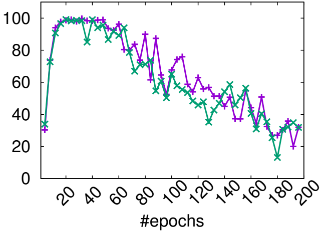


(h) $p = 16$

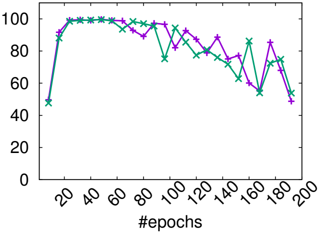
NLC-F train accuracy



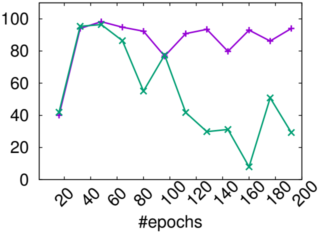
(i) $p = 2$



(j) $p = 4$

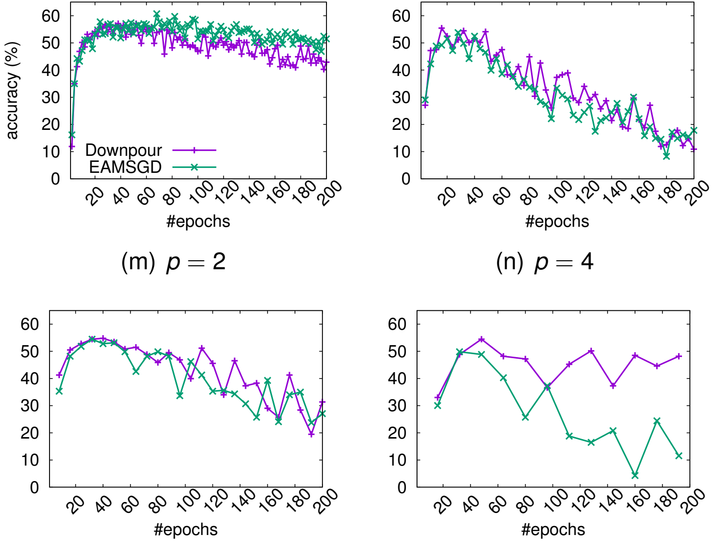


(k) $p = 8$



(l) $p = 16$

NLC-F test accuracy



(m) $p = 2$ (n) $p = 4$

(o) $p = 8$ (p) $p = 16$

Challenges on current and emerging platforms

- Centralized parameter server becomes a bottleneck
- Sharded parameter server suffers inconsistency
- Narrow channel between learners (on GPUs) and parameter server (on CPU)

Conclusion and future work

- ASGD faces challenges on HPC systems with a large number of learners
- Other approaches need to be explored