Asynchronous Parallel Stochastic Gradient For Nonconvex Optimization

UNIVERSITY of ROCHESTER

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu

Motivation

- **Nonconvex** optimization is quite common in Machine Learning (Deep Learning, Natural Language Processing, Recommendation System, etc.)
- **Asy**nchronous **S**tochastic **G**radient (**AsySG**) is a powerful method in solving large scale machine learning problems.
- However, the theoretical analysis is still limited for **nonconvex** optimization.

Background

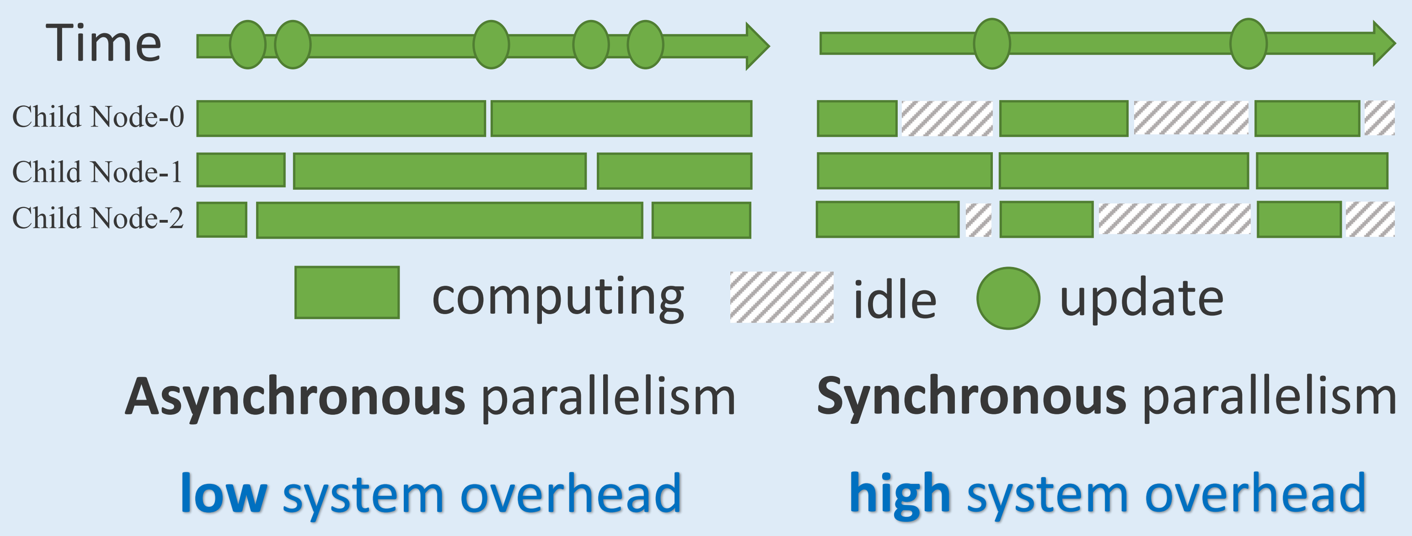
Consider the nonconvex optimization (deep learning, NLP, Recommendation):

$$\min_{x\in\mathbb{R}^n} f(x) = \mathbb{E}_\xi[F(x,\xi)].$$

- $\xi \in \Xi$ is a random variable.
- $f(x)$ is a smooth but not necessarily convex function.

Example: $\Xi = \{1,2,\cdots,N\}$ is an index set of all training samples and $F(x;\xi)$ is the corresponding loss function.

Asynchronous vs Synchronous



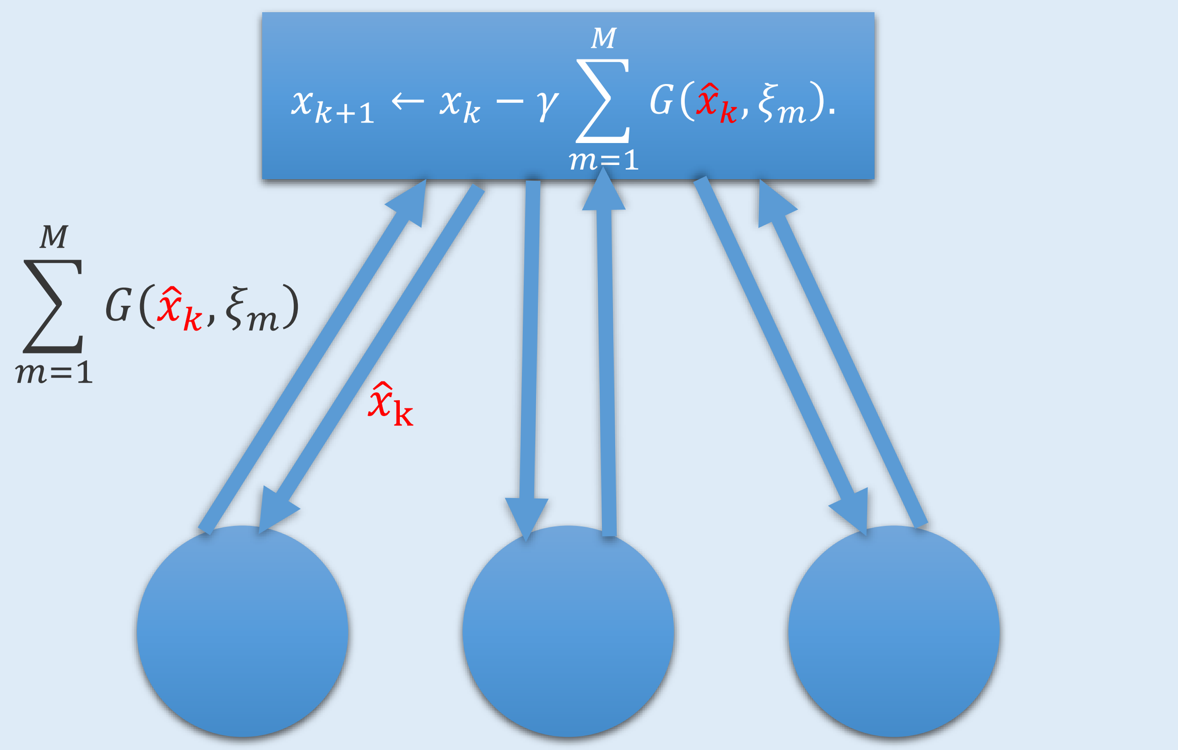
Two Implementations of AsySG (AsySG-con & AsySG-incon)

The procedure of AsySG

A central node or a shared memory maintains the optimization variable x.

All child nodes/threads run the following procedure concurrently:

1. (**Read**): read the parameter $\hat{x}_k$ from the central node/shared memory.
2. (**Compute**): sample M training data $\xi_1,\cdots,\xi_M$ and compute a batch of the stochastic gradient $G(\hat{x}_k,\xi_m) = \nabla f(\hat{x}_k,\xi_m), m = 1,\cdots,M$ locally.
3. (**Update**): Update parameter x in the central node/shared memory ***without* locks**.



Key challenges in analysis

1) $\hat{x}_t \neq x_t$;

2) Different implemenations => Different forms of $\hat{x}_t$.

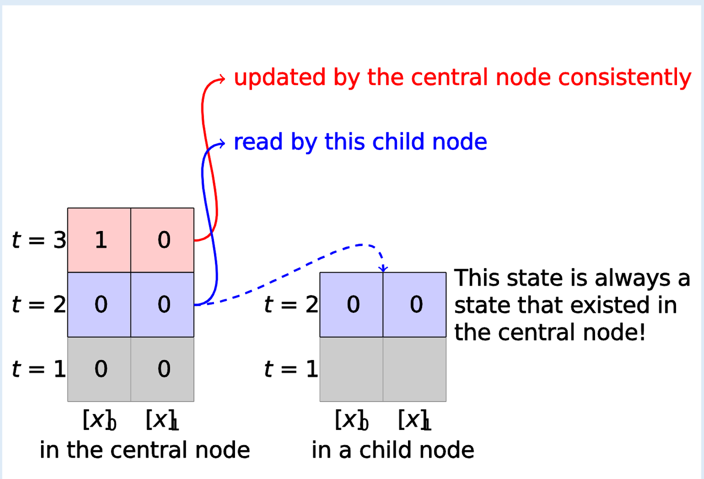


*Figure 1 Consistent read in computing cluster (**AsySG-con**).*

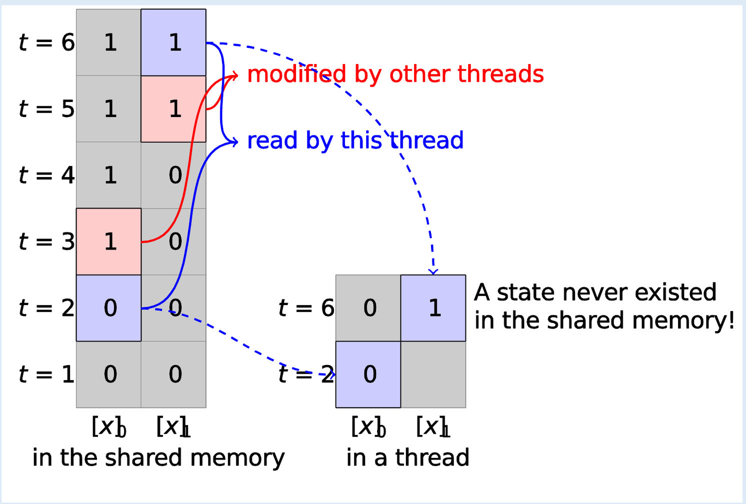


*Figure 2 Inconsistent read in multicore machine (**AsySG-incon**).*

$$x_{k+1} = x_k - \gamma\sum_{m=1}^{M} G(x_{k-\tau_k},\xi_{k,m}).$$

T := upper bound of staleness.

For example, in AsySG-con: $\max_k \tau_k \leq T$.

In AsySG-incon: $J(k) \subset \{k-1,\cdots,k-T\}$.

In practice T is proportional to the number of workers.

$$[x_{k+1}]_{i_k} = [x_k]_{i_k} - \gamma\sum_{m=1}^{M}[G(\hat{x}_k,\xi_{k,m})]_{i_k}.$$

$$\hat{x}_k = x_k - \sum_{j\in J(k)}(x_{j+1} - x_j).$$

M is the mini-batch size,

γ is the steplength.

References

[1] F. Niu, B. Recht, C. Re, and S. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. NIPS, 2011.

[2] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. NIPS, 2011.

[3] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. NIPS, 2012.

[4] J. Liu, S. J. Wright, C. Re, V. Bittorf, and S. Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. ICML, 2014.

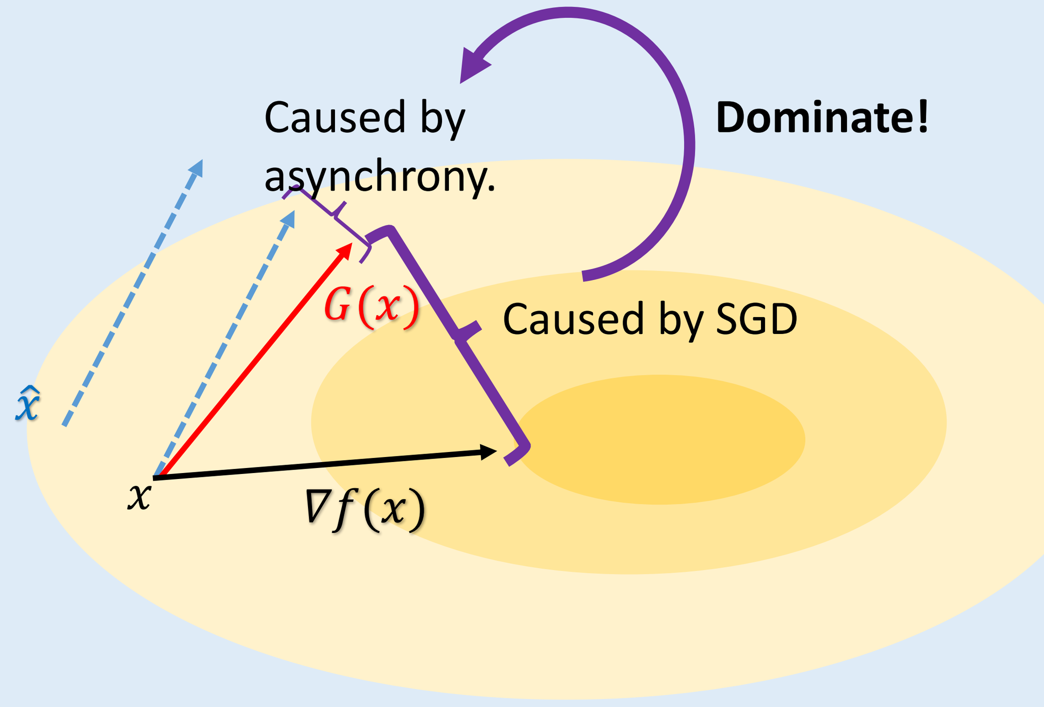
Convergence Rate for AsySG

Theorem

Assume that certain assumptions hold and $\mathbb{E}_\xi(\|G(x,\xi) - \nabla f(x)\|^2) \leq \sigma^2$. Set the steplength to be a constant $\gamma = O\left(\sqrt{\frac{1}{MK\sigma^2}}\right)$. If the delay parameter T is bounded by $K \geq O\left(\frac{MT^2}{\sigma^2}\right)$, then the output of AsySG-con satisfies the following ergodic convergence rate:

$$\min_{k\in\{1,\cdots,K\}} \mathbb{E}(\|\nabla f(x_k)\|^2) \leq \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}(\|\nabla f(x_k)\|^2) \leq O\left(\frac{\sigma}{\sqrt{MK}}\right).$$

- Consistent convergence rate with SGD.
- ***Linear speedup*** up to $O(\sqrt{K})$ machines.
- Better linear speedup property than existing work (linear speedup up to $O(K^{1/4})$ machines in [2]).
- For AsySG-incon, we have a similar convergence rate.



Experiment

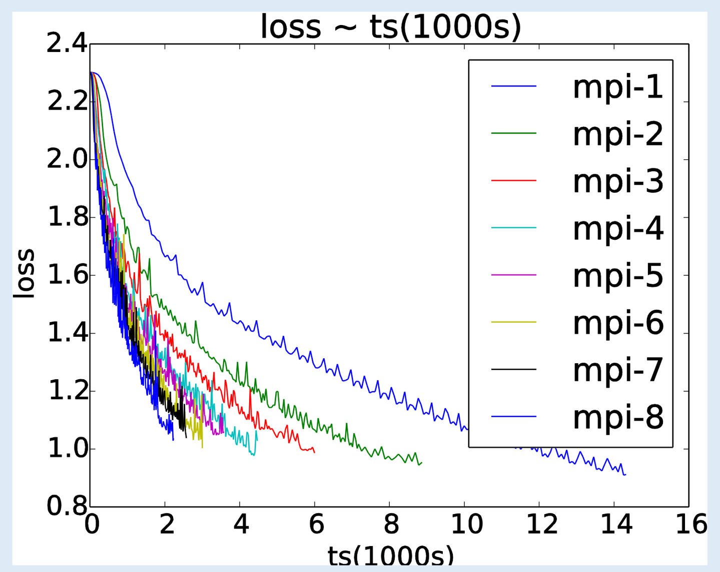


Figure 3 AsySG-con algorithm run on various numbers of mahicnes.

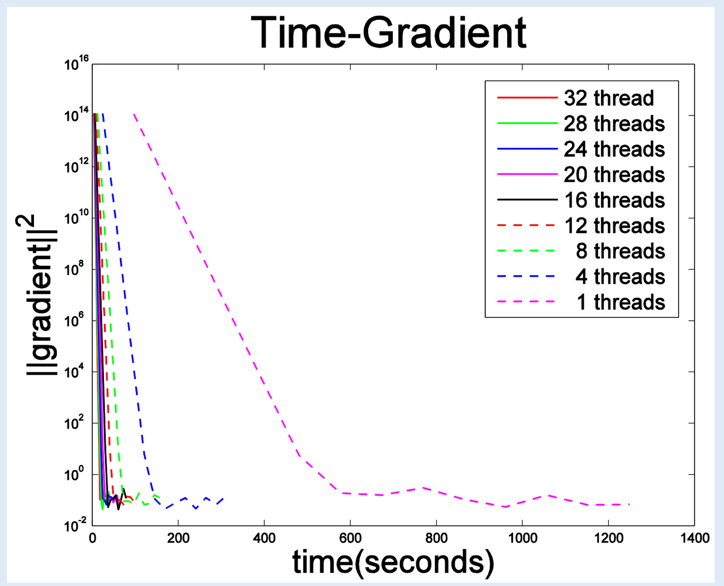


Figure 4 AsySG-incon algorithm run on various numbers of cores.

AsySG-con (CIFAR10-FULL)

	mpi-1	mpi-2	mpi-3	mpi-4	mpi-5	mpi-6	mpi-7	mpi-8
iter speedup	1.01	1.93	2.65	3.42	4.27	4.92	5.36	5.96
time speedup	1.00	1.73	2.28	2.88	3.56	4.07	4.41	5.00

AsySG-incon (Synthetic data)

	thr-1	thr-4	thr-8	thr-12	thr-16	thr-20	thr-24	thr-28	thr-32
iter speedup	1	3.9	7.8	11.6	15.4	19.9	24.1	28.7	31.6
time speedup	1	4.0	8.1	11.9	16.3	19.2	22.7	26.1	29.2