Slow and Stale Gradients Can Win the Race: Error-Runtime Trade-offs in Distributed SGD

Gauri Joshi

Carnegie Mellon University

Shannon YouTube Channel
1st February 2019

Joint work with

Sanghamitra Dutta, CMU

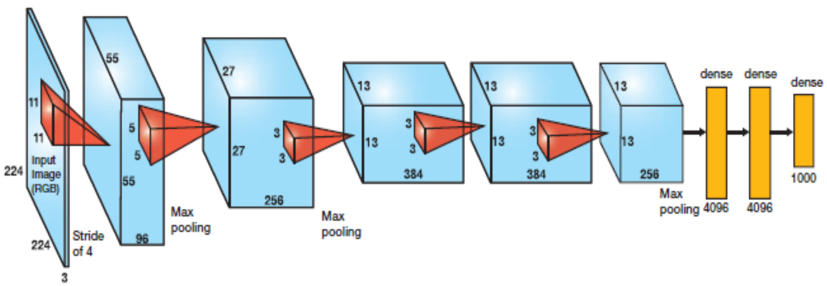
Jianyu Wang, CMU

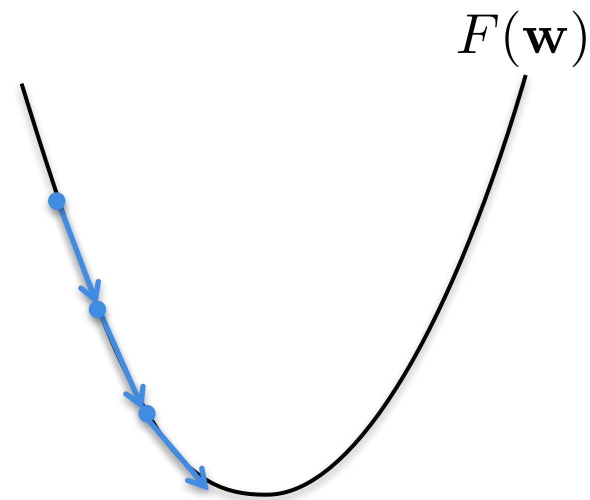
Soumyadip Ghosh, IBM

Parijat Dube, IBM

Priya Nagpurkar, IBM

Stochastic Gradient Descent is the backbone of ML

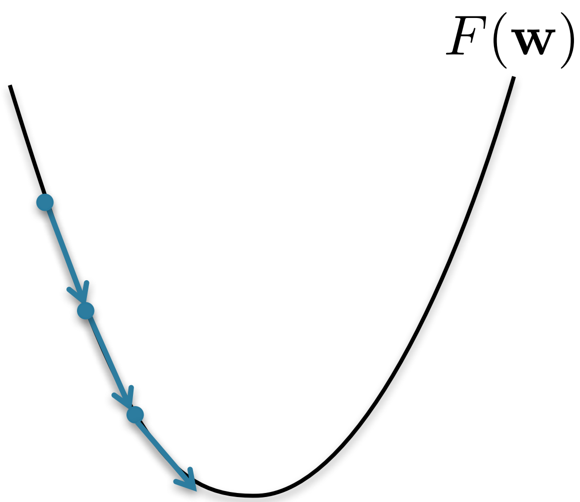




$$\mathbf{w_{n+1}} = \mathbf{w_n} - \eta \nabla F(\mathbf{w})$$

Speeding Up SGD convergence
is of critical importance!

Batch Gradient Descent



F(w) is the empirical risk function

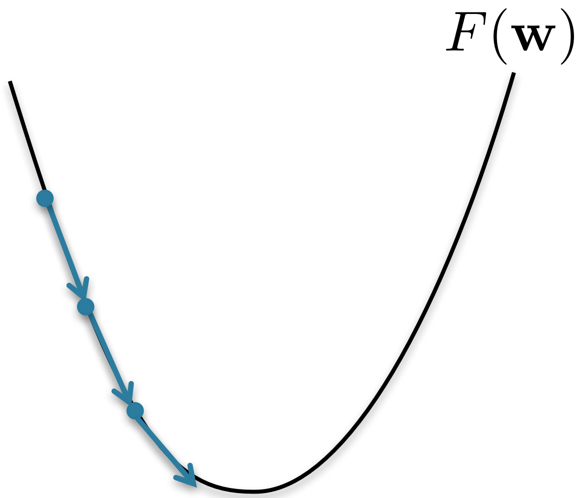
$$\min_{\mathbf{w}} \left\{ F(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{w}, \xi_n) \right\}$$

ξ_n is the n-th labeled sample

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \frac{\eta}{N} \sum_{i=1}^{N} \nabla f(\mathbf{w}_j, \xi_i)$$

Too expensive for large datasets

Mini-batch SGD



F(w) is the empirical risk function

$$\min_{\mathbf{w}} \left\{ F(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{w}, \xi_n) \right\}$$

ξ_n is the n-th labeled sample

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \frac{\eta}{m} \sum_{i=1}^{m} \nabla f(\mathbf{w}_j, \xi_i)$$

Noisier, but computationally tractable

Accelerating single-node SGD convergence

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \frac{\eta}{m} \sum_{n=1}^{m} \nabla f(\mathbf{w}_j, \xi_n)$$

Learning Rate Schedules: AdaGrad, Adam

Momentum Methods: Polyak, Nesterov

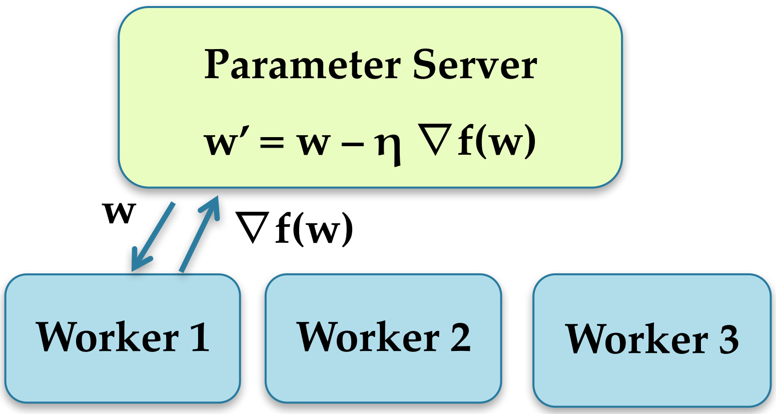
Variance Reduction Methods

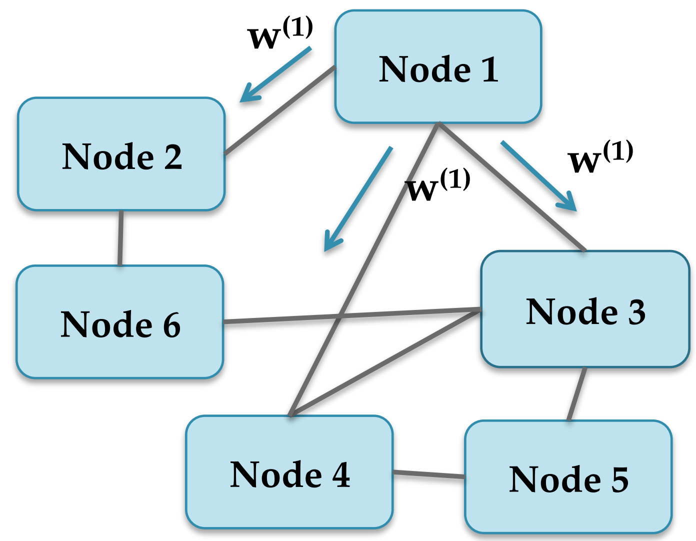
Second-Order Hessian Methods

For large training datasets single-node SGD can be prohibitively slow…

Our Work:

Speeding Up Error-Runtime Convergence of Distributed SGD

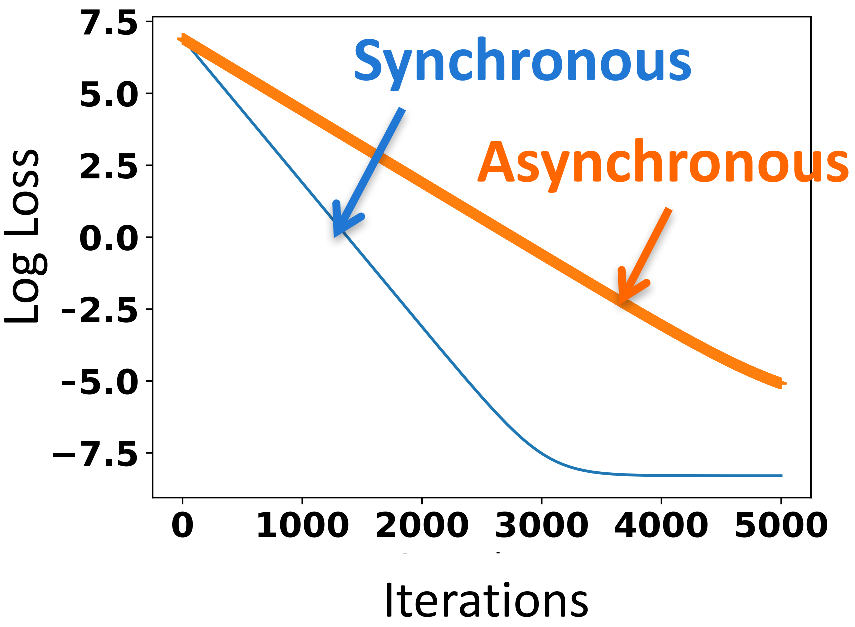


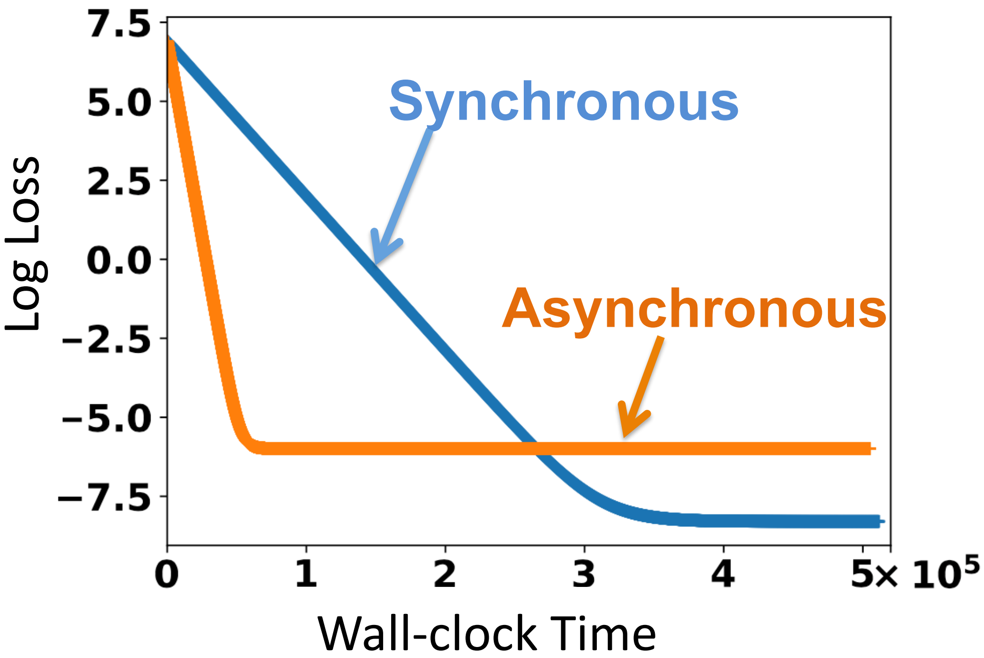


Key Issues

- Straggling Workers
- Gradient Staleness
- Communication between nodes

Our Approach:
Considering convergence w.r.t. *wall-clock time* instead of iterations

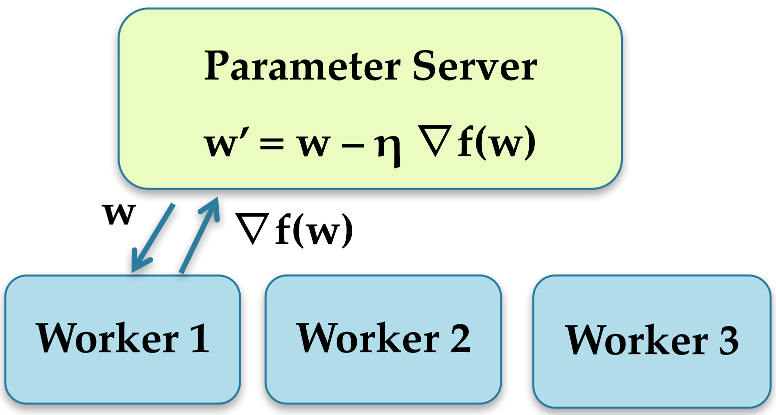




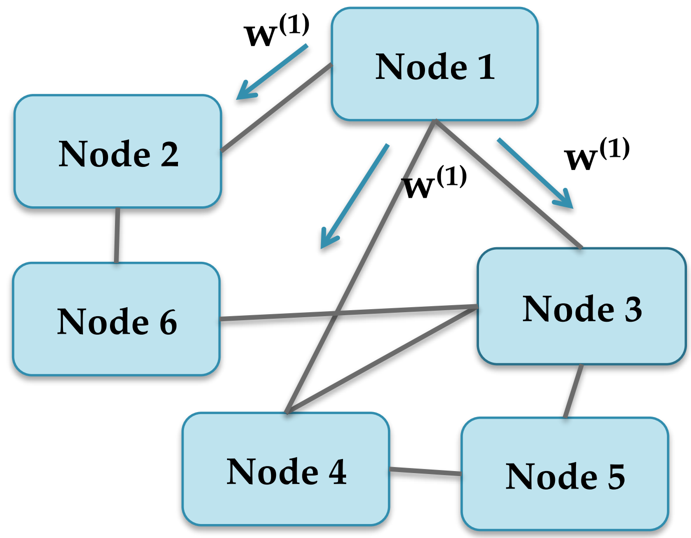
Need novel convergence analysis as well as runtime analysis

Our Work:

Speeding Up Error-Runtime Convergence of Distributed SGD



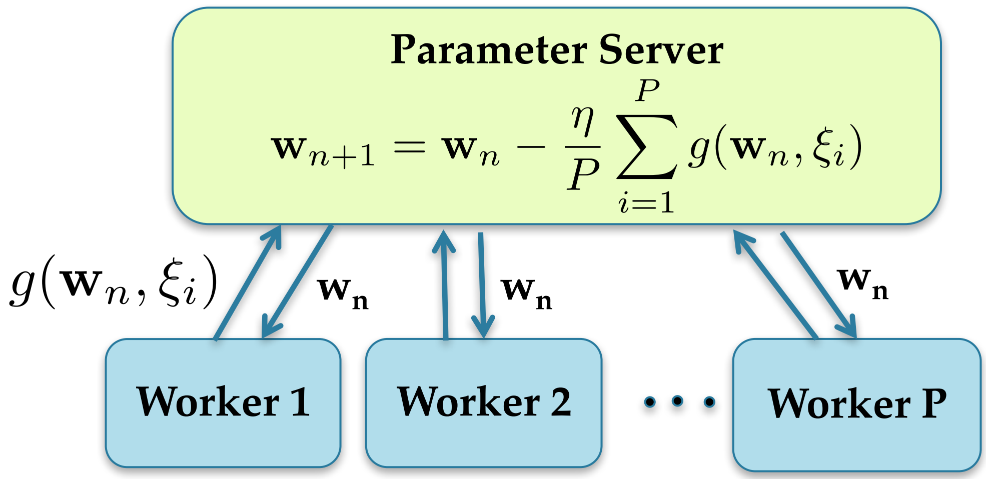
[Dutta, Joshi et al,
Slow and Stale Gradients,
AISTATS 2018]



[Wang-Joshi, Cooperative-SGD, 2018]

[Wang-Joshi, Adaptive Comm, SysML 2018]

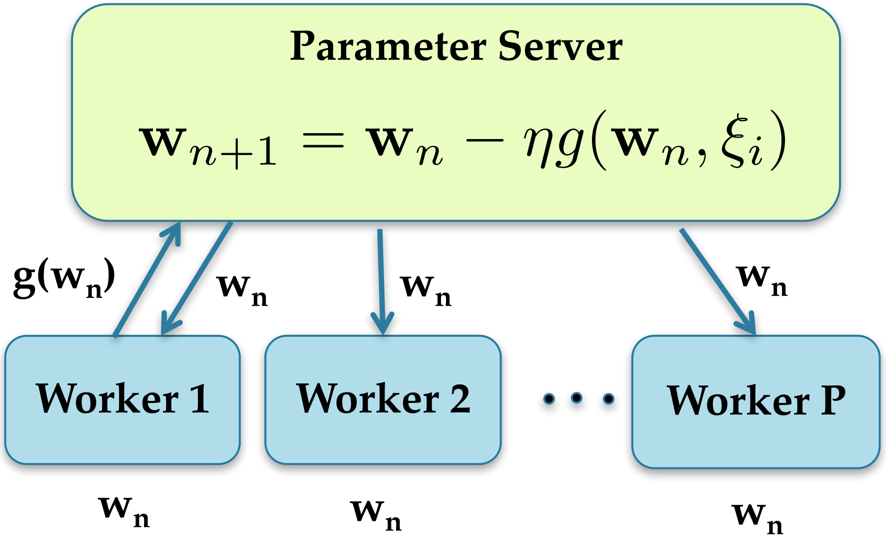
Parameter Server Model: Synchronous SGD



Can process a P-times larger mini-batch in each iteration

Bottlenecked by one or more slow workers

Parameter Server Model: Asynchronous SGD

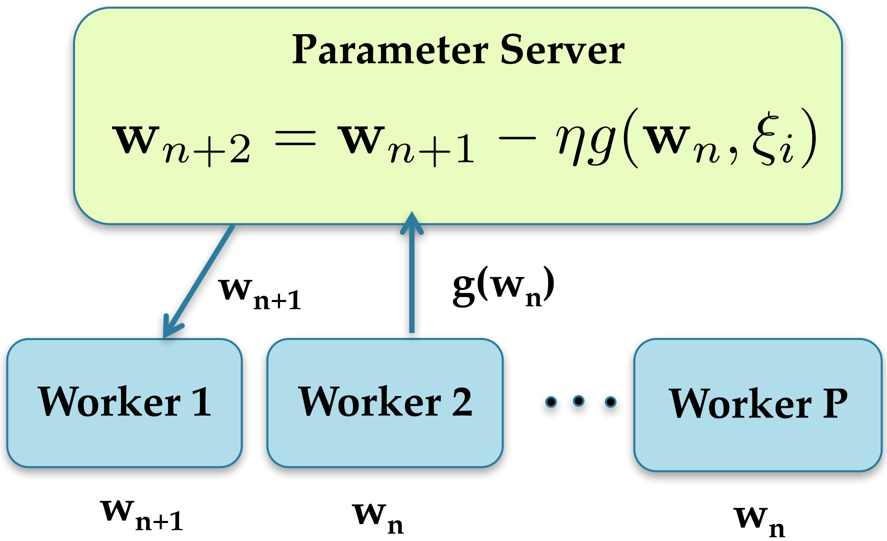


[Recht 2011,
Dean 2012,
Cipar 2013 …]

Don't have to wait for straggling workers

Gradient Staleness can increase error

Parameter Server Model: Asynchronous SGD

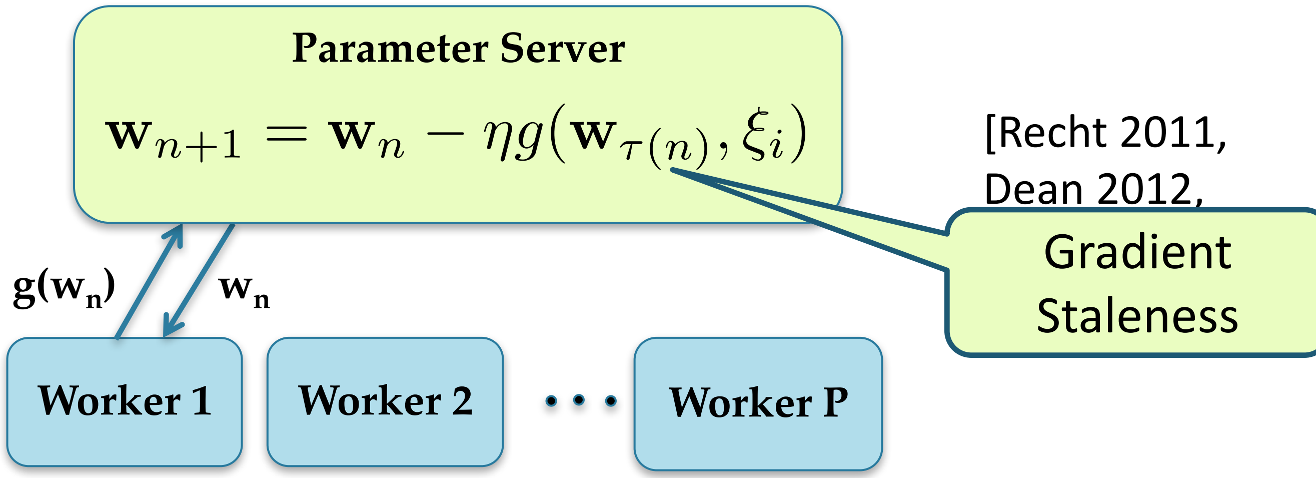


[Recht 2011, Dean 2012, Cipar 2013 …]

Don’t have to wait for straggling workers

Gradient Staleness can increase error

Parameter Server Model: Asynchronous SGD



[Recht 2011, Dean 2012,

Don't have to wait for straggling workers

Gradient Staleness can increase error

Outline

Error Analysis of Sync, Async SGD

Runtime Analysis of Sync, Async SGD

Straggler Mitigation via SGD variants

Staleness Compensation in Async SGD

Sync SGD: Error Analysis

Update Rule: Equivalent to mini-batch SGD with batch size Pm

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \frac{\eta}{P}\sum_{i=1}^{P} g(\mathbf{w}_n, \xi_i)$$

For c-strongly convex, L-smooth functions [Bottou, 2016]

$$\mathbb{E}[F(\mathbf{w}_J) - F^*] \leq \frac{\eta L \sigma^2}{2c(Pm)} + (1-\eta c)^J \left(F(\mathbf{w}_0) - F^* - \frac{\eta L \sigma^2}{2c(Pm)}\right)$$

Error Floor

Decay Rate

Async SGD: Error Analysis

Update Rule $\mathbf{w}_{n+1} = \mathbf{w}_n - \eta g(\mathbf{w}_{\tau(n)}, \xi_i)$

Hard to analyze due to stale gradients

Assumptions in Previous works

- Upper Bound on Staleness $\tau(n) \leq B$ [Hogwild 2014, Lian et al 2015]
- Geometric staleness distribution

$$P(\tau(n) = j) = p(1-p)^{j-1}$$

[Mitiliagkas et al 2016]

- Independently drawn gradient staleness

We remove these assumptions, and instead consider

$$\mathbb{E}[||\nabla F(\mathbf{w}_j) - \nabla F(\mathbf{w}_{\tau(j)})||_2^2] \leq \gamma \mathbb{E}[||\nabla F(\mathbf{w}_j)||_2^2] \qquad \gamma \leq 1$$

Async SGD: Error Analysis

For c-strongly convex, L-smooth functions,

$$\mathbb{E}[F(\mathbf{w}_J) - F^*] \leq \frac{\eta L \sigma^2}{2c\gamma' m} + (1 - \eta c \gamma')^J \left(\mathbb{E}[F(\mathbf{w}_0) - F^*] - \frac{\eta L \sigma^2}{2c\gamma' m} \right)$$

Larger than Sync-SGD

Can be faster than Sync SGD if $p_o / 2 > \gamma$

where $\gamma' = 1 - \gamma + p_0/2$

γ is the staleness bound,

and p_0 is the probability of getting a fresh gradient

Analysis can be generalized to non-convex objectives

Outline

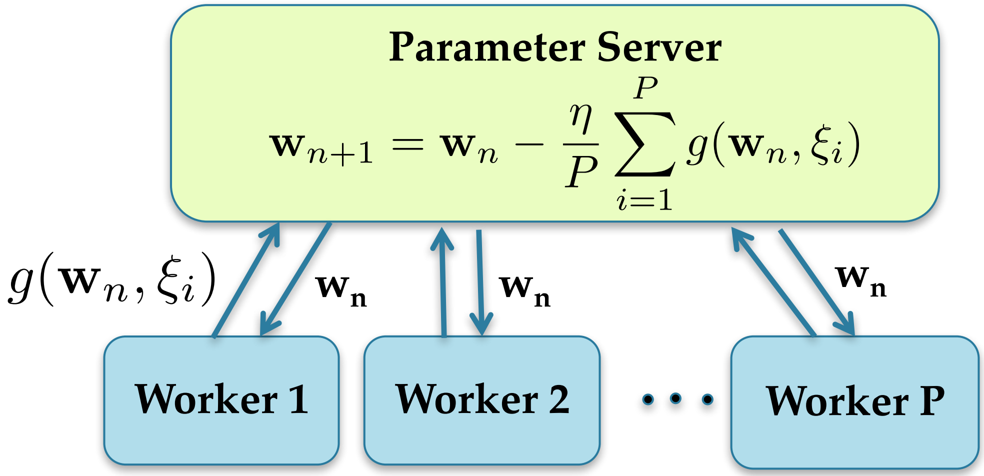
Error Analysis of Sync, Async SGD

Runtime Analysis of Sync, Async SGD

Straggler Mitigation via SGD variants

Staleness Compensation in Async SGD

Expected Time Per Iteration

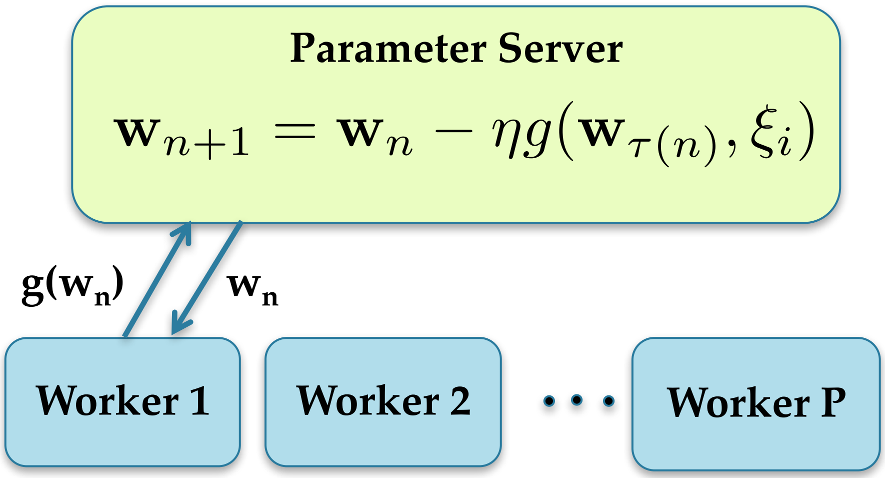


Each worker takes time Y~ exp(μ)

Synchronous SGD

$$\mathbb{E}[T] = \mathbb{E}[Y_{P:P}]$$
$$\approx \frac{1}{\mu}\log P$$

Expected Time Per Iteration



Each worker takes time Y~ exp(μ)

Synchronous SGD

$$\mathbb{E}[T] = \mathbb{E}[Y_{P:P}]$$
$$\approx \frac{1}{\mu} \log P$$

Asynchronous SGD

$$\mathbb{E}[T] = \frac{1}{\mu P}$$

P log P times smaller!

Outline

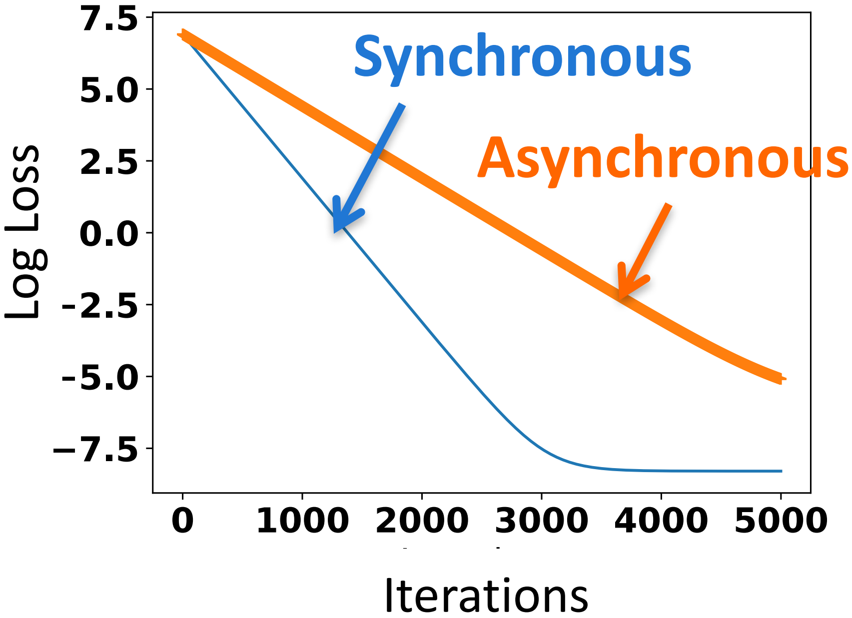
Error Analysis of Sync, Async SGD

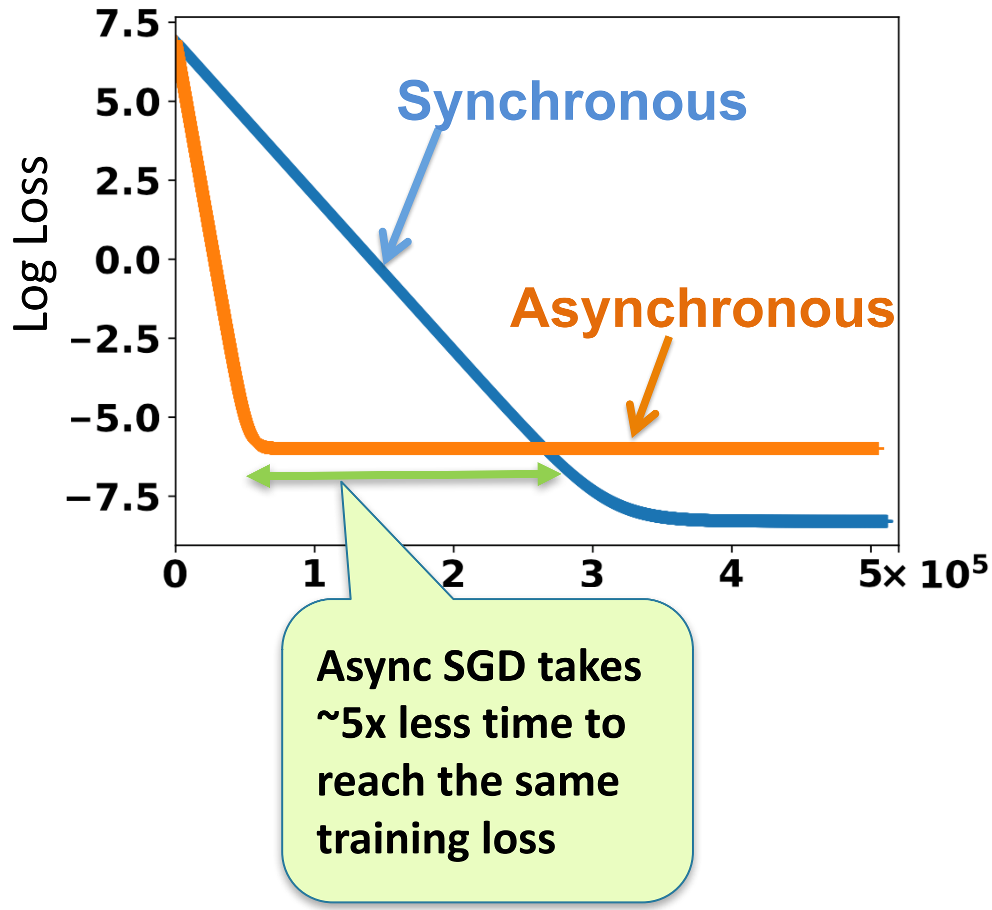
Runtime Analysis of Sync, Async SGD

Straggler Mitigation via SGD variants

Staleness Compensation in Async SGD

Need to compare convergence w.r.t. *wall-clock time* instead of iterations





Outline

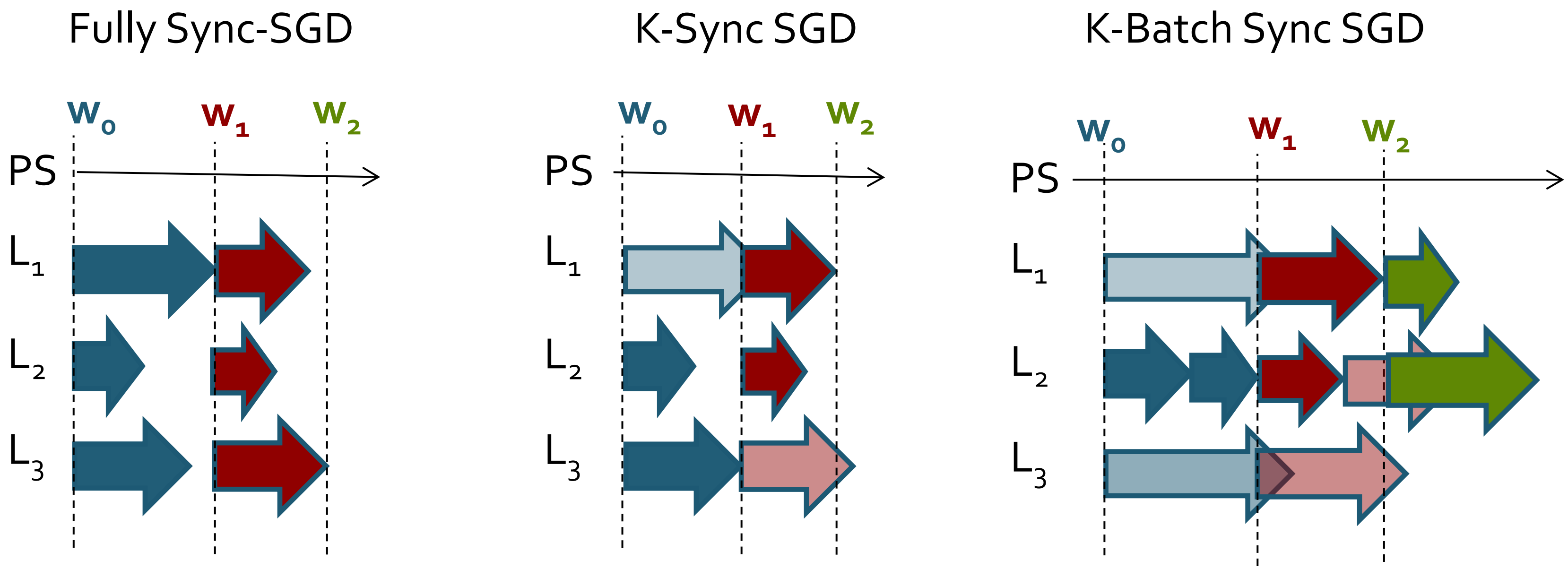
Error Analysis of Sync, Async SGD

Runtime Analysis of Sync, Async SGD

Straggler Mitigation via SGD variants

Staleness Compensation in Async SGD

Sync SGD Variants

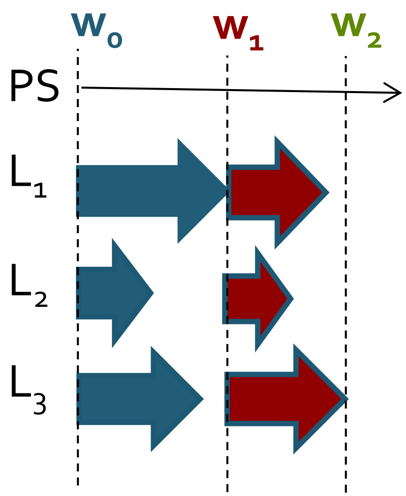


Related Work: Revisiting Distributed SGD [Chen, Monga et al]

Instead of erasure coding [Tandon et al], we ignore the slow gradients

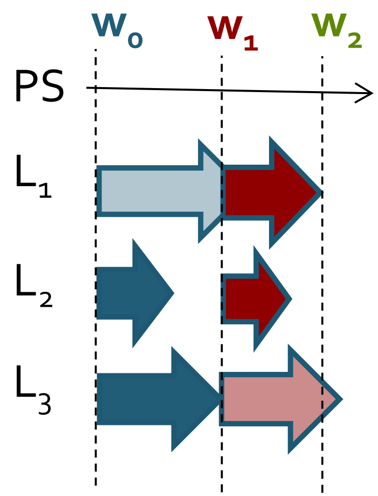
Sync SGD: Expected Time Per Iteration

Fully Sync-SGD



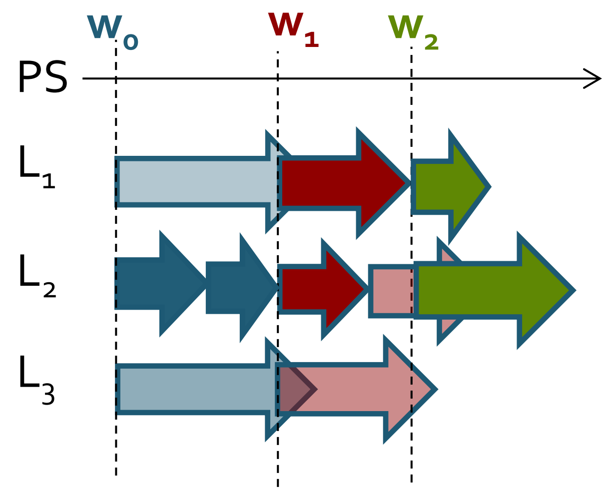
$$\mathbb{E}[T] = \mathbb{E}[Y_{P:P}] \approx \frac{1}{\mu} \log P$$

K-Sync SGD



$$\mathbb{E}[T] = \mathbb{E}[Y_{K:P}] \approx \frac{1}{\mu} \log \frac{P}{P-K}$$

K-Batch Sync SGD

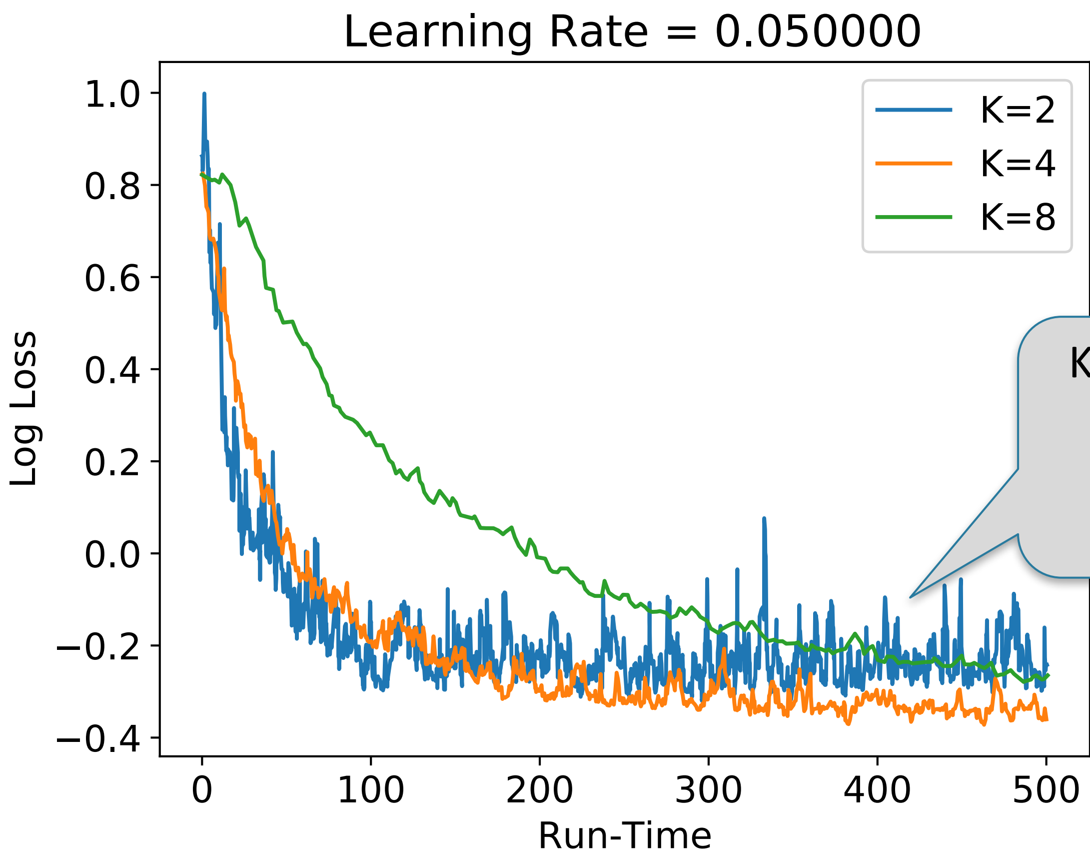


$$\mathbb{E}[T] = \frac{K}{\mu P}$$

for exponential Y

Sync SGD: Choosing the best K

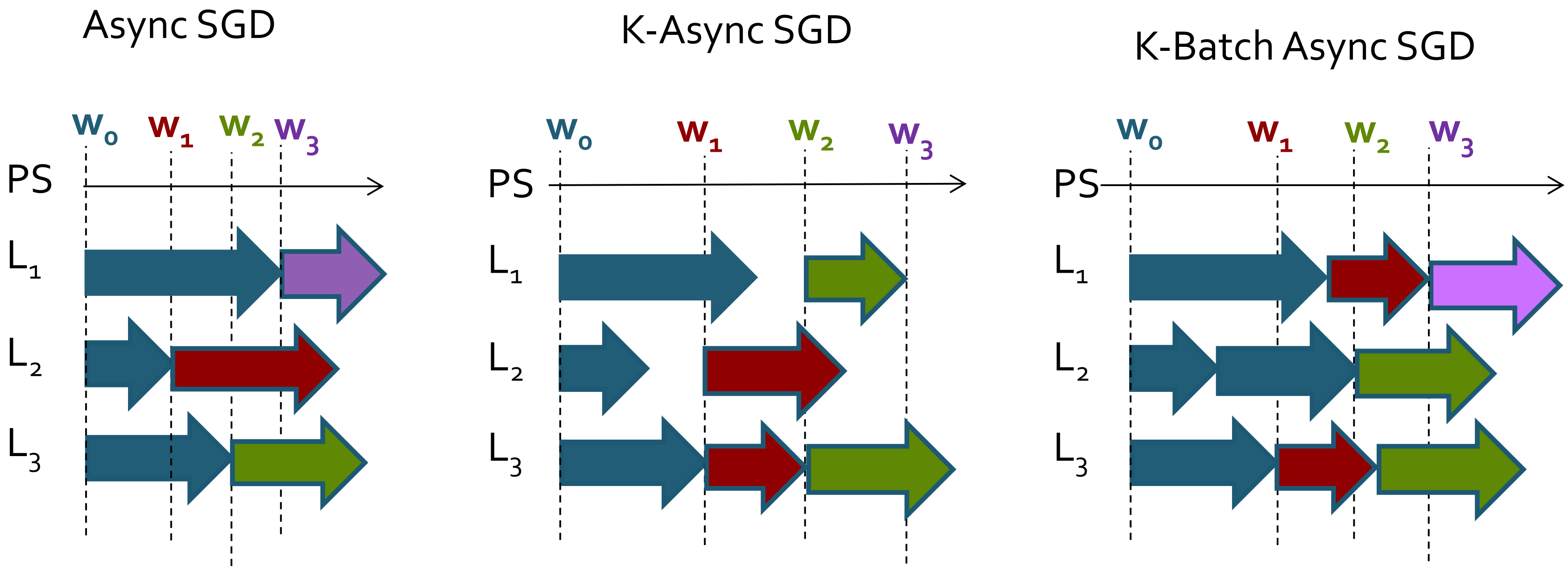
Error is equivalent to mini-batch SGD with batch size Km



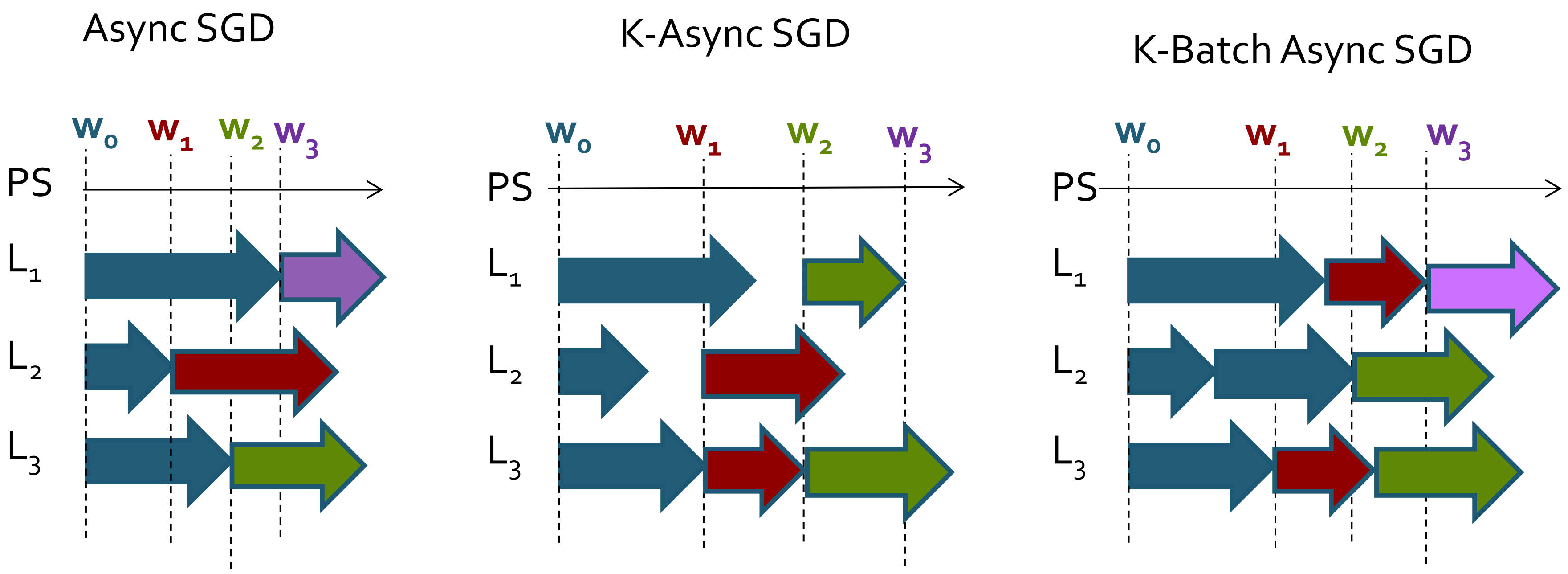
MNIST dataset

K = 4 strikes a good balance between conv. speed and error floor

Async SGD Variants

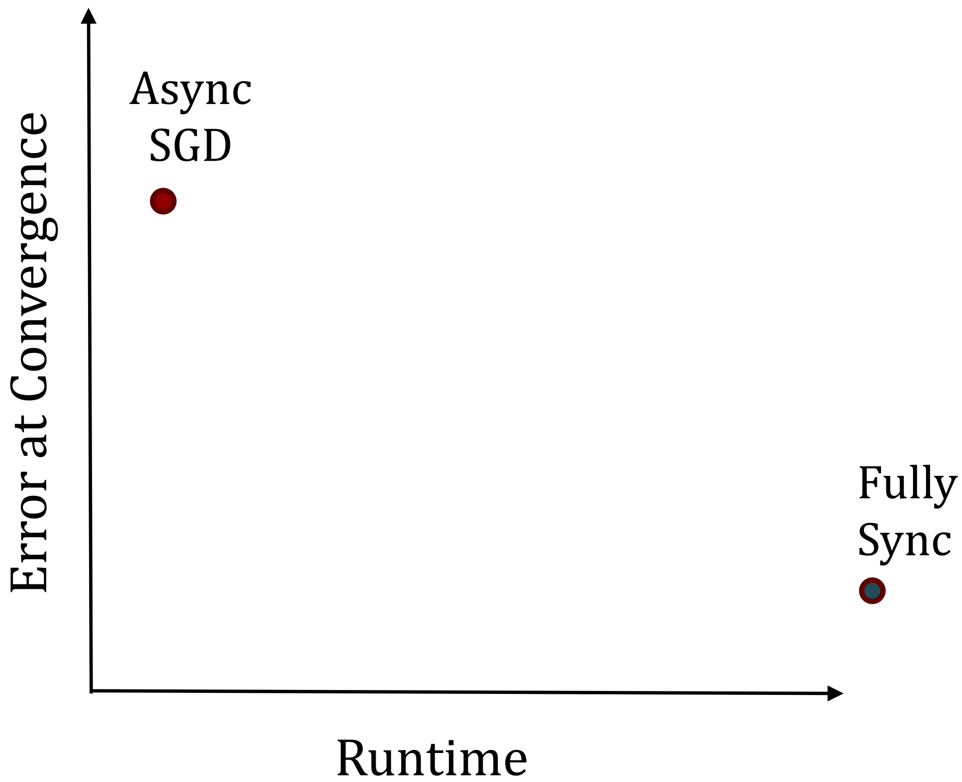


Async SGD Variants

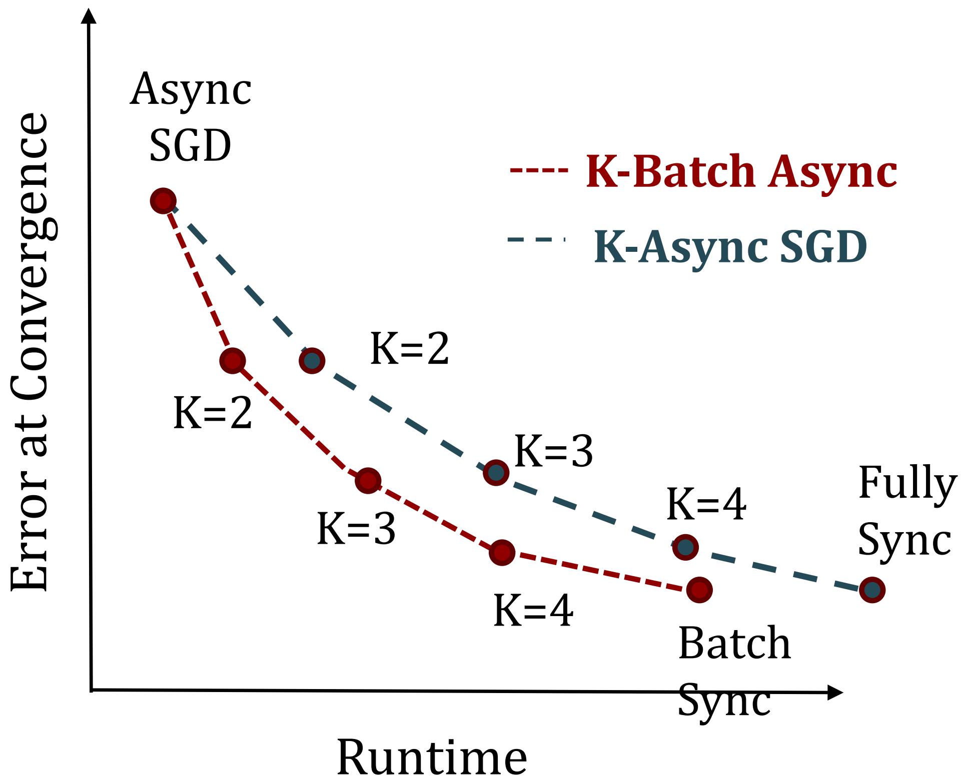


Our runtime and error analysis for Async SGD can be generalized to these variants

Spanning the spectrum between Synchronous and Asynchronous SGD

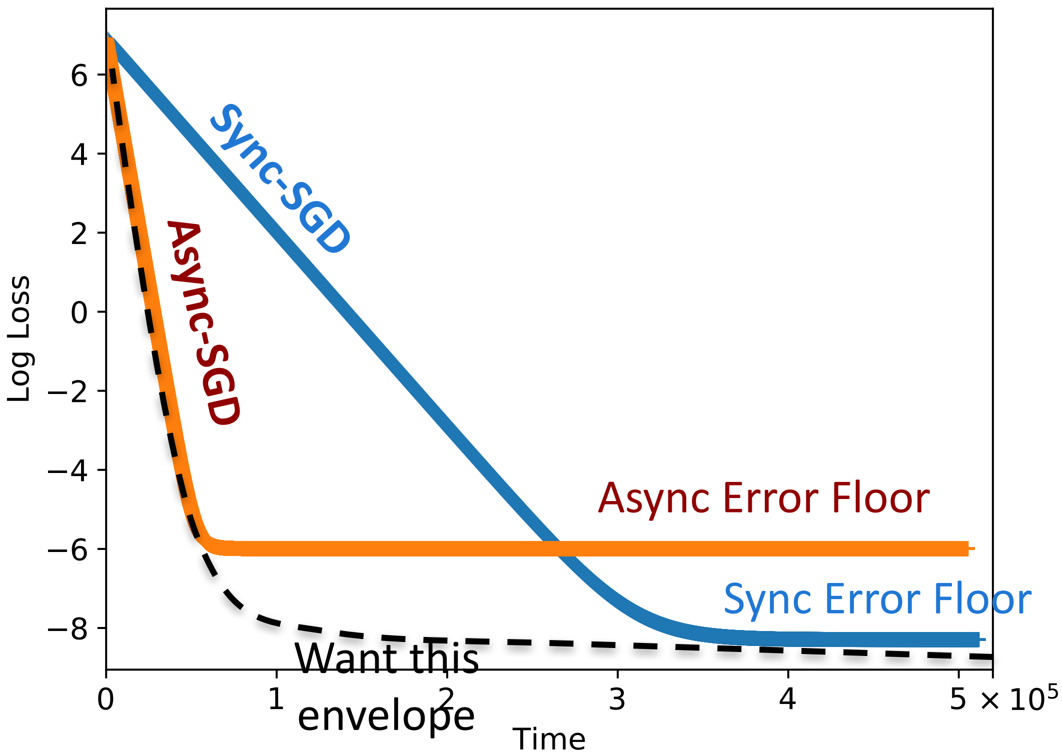


Spanning the spectrum between Synchronous and Asynchronous SGD



Ongoing Research Direction

Gradually increasing synchrony



Outline

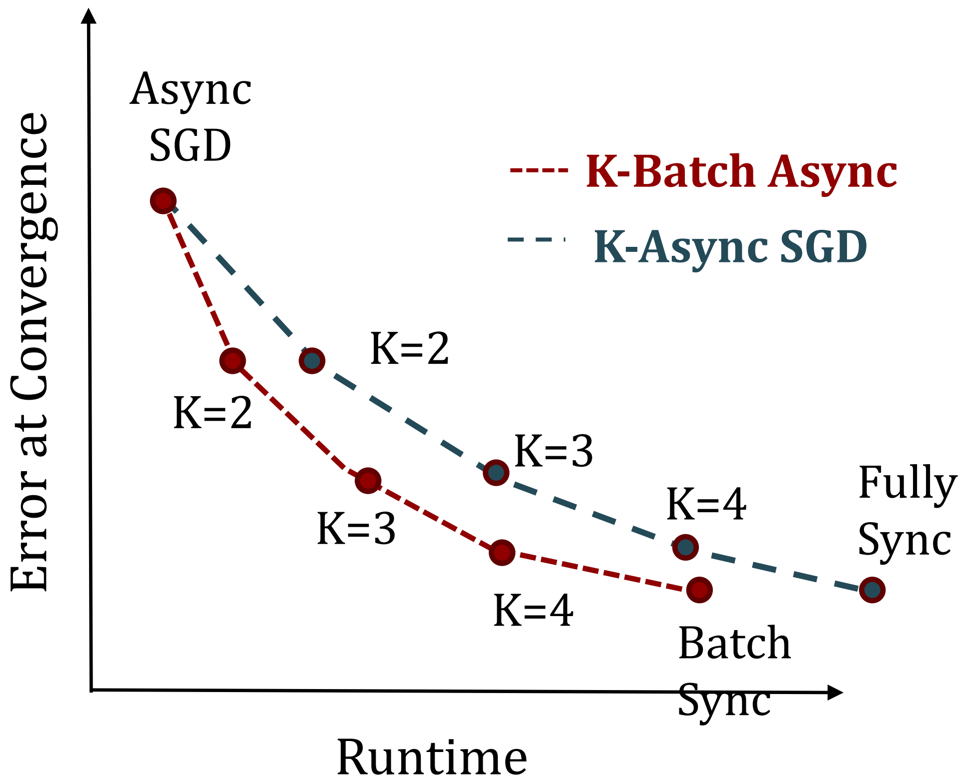
Error Analysis of Sync, Async SGD

Runtime Analysis of Sync, Async SGD

Straggler Mitigation via SGD variants

Staleness Compensation in Async SGD

Spanning the spectrum between Synchronous and Asynchronous SGD

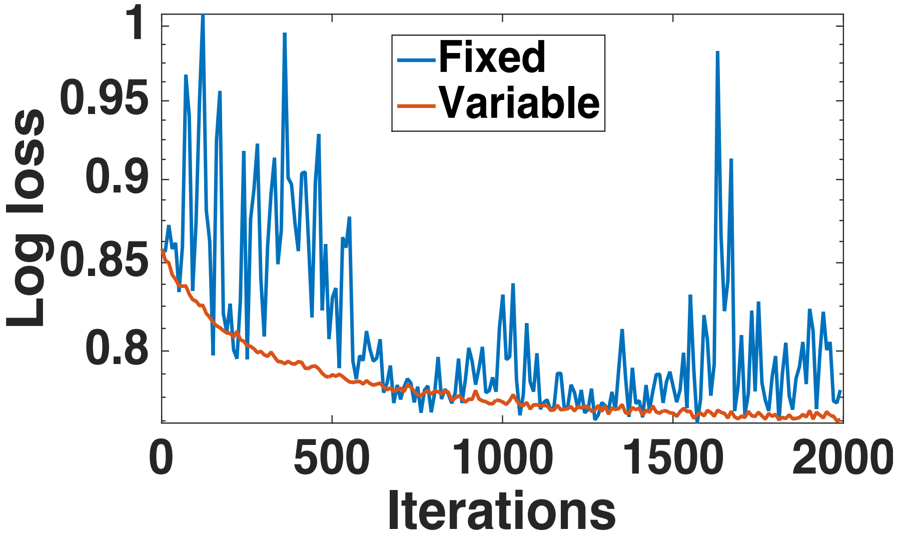


Adapting the Learning Rate to Tame Gradient Staleness

Proposed Learning Rate Schedule

$$\eta_j = \min \left\{ \frac{C}{||\mathbf{w}_j - \mathbf{w}_{\tau(j)}||_2^2}, \eta_{max} \right\}$$

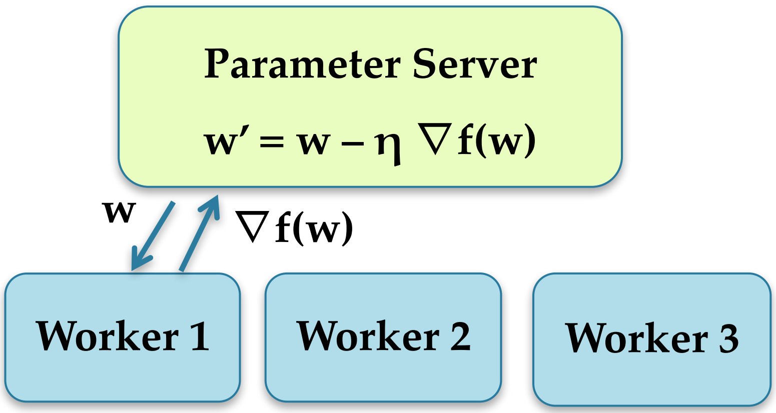
helps eliminate the bounded staleness assumption in our analysis



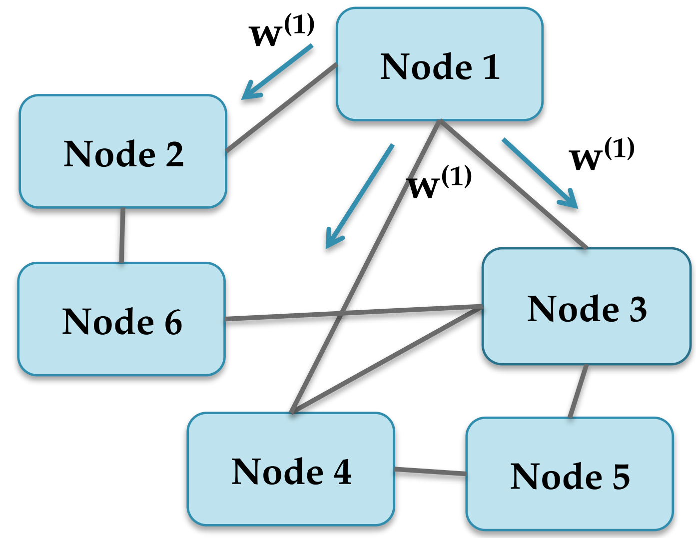
η = 0.01,
CIFAR10 dataset

Related to momentum tuning in [Mitliagkas 2016]

Our Work:
Speeding Up Error-Runtime Convergence of Distributed SGD



[Dutta et al, AISTATS 2018]



[Wang-Joshi, Cooperative-SGD, 2018]

[Wang-Joshi, Adaptive Comm, 2018]

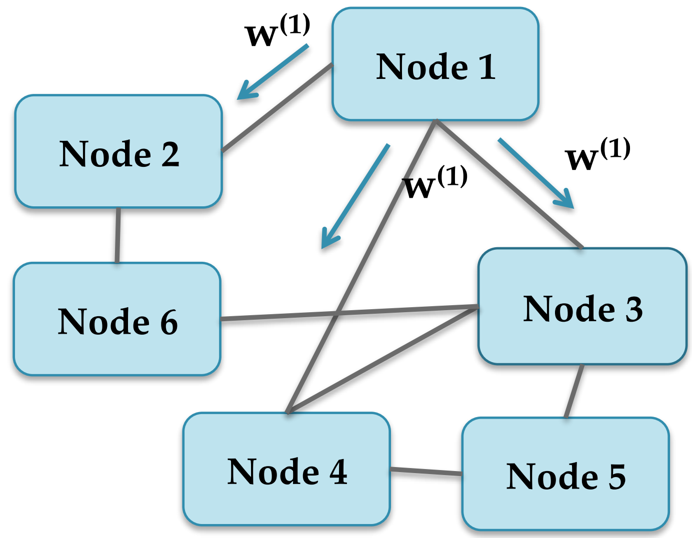
Key Issues

- Straggling Workers
- Gradient Staleness

Two Ways of Reducing Communication

1. Compressing or quantizing gradients sent by nodes to the parameter server

2. Performing local updates at the nodes and averaging periodically to encourage consensus



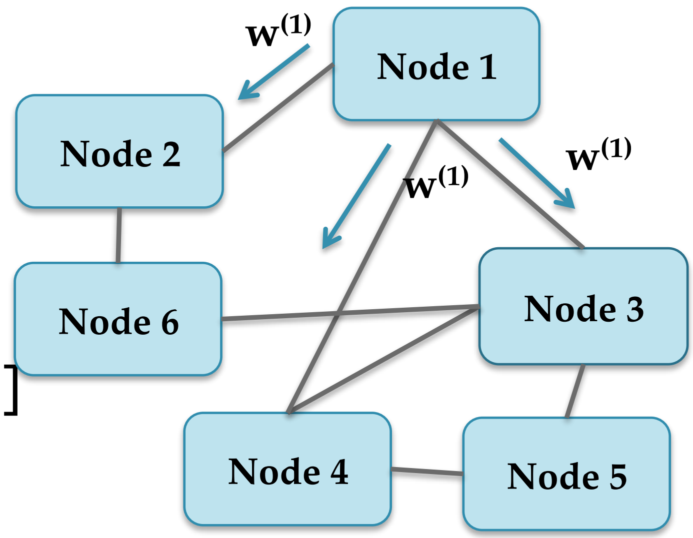
[Wang-Joshi, Cooperative-SGD, 2018]

[Wang-Joshi, Adaptive Comm, 2018]

Distributed SGD with Local Updates

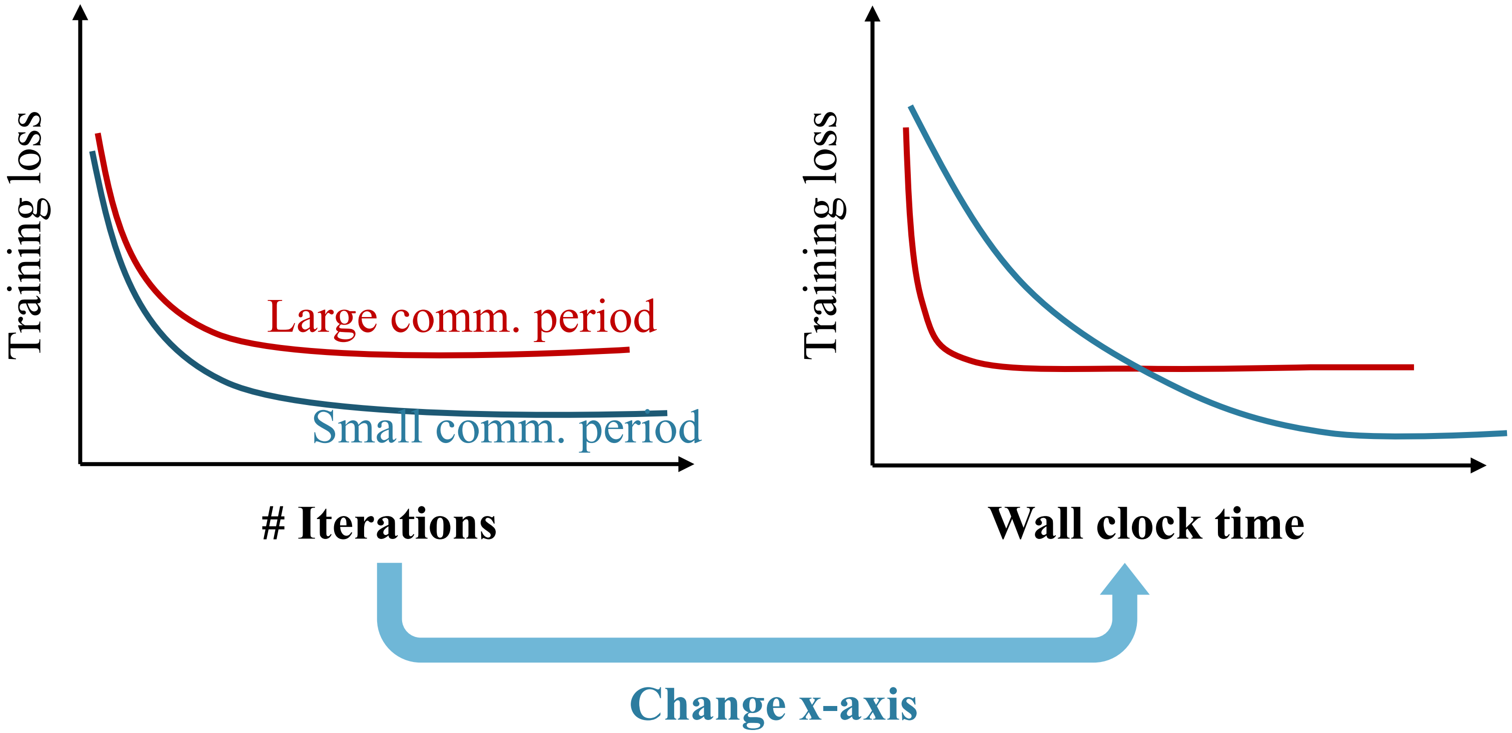
DESIGN PARAMETERS

1. Number of local updates, τ, the communication period
2. Model-averaging Method
   - Federated Avg, [McMahan 2015]
   - Elastic Avg, [Zhang et al 2015]
   - Decentralized Avg, [Lian et al 2017]



Error convergence analysis with local updates for non-convex objectives was mostly unexplored

Error-Runtime Trade-off in Local-Update SGD



Model discrepancies gives inferior error-convergence

Large τ or sparse averaging reduces communication delay

Outline

Error Analysis via the Cooperative SGD Framework

Runtime Analysis

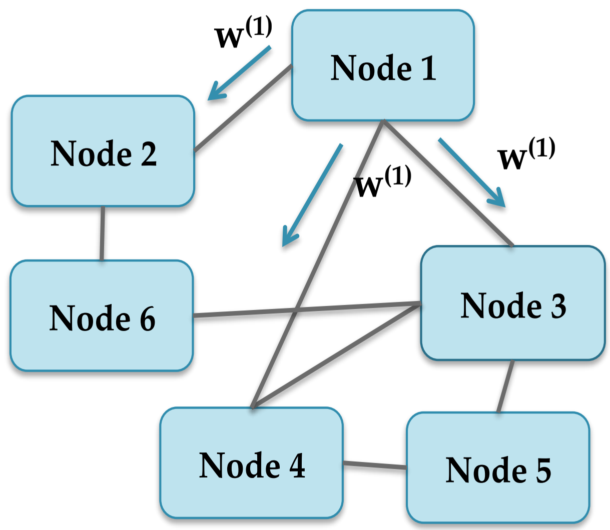
Adaptive Communication Strategies

The Cooperative SGD Framework

KEY ELEMENTS

1. Model Versions at m workers and v auxiliary nodes

$$\mathbf{X}_k = [\mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(p)}, \mathbf{z}_k^{(1)}, \ldots, \mathbf{z}_k^{(v)}]$$



2. τ local updates at m workers, no updates at auxiliary nodes

3. Mixing Matrix $\mathbf{W_k}$

$$\mathbf{W}_k = \begin{cases} \mathbf{W}, & k \bmod \tau = 0 \\ \mathbf{I}_{(p+v)\times(p+v)}, & \text{otherwise} \end{cases}.$$

$$\mathcal{A}(\tau, \mathbf{W}, v)$$

4. Update Rule

$$\mathbf{X}_{k+1} = (\mathbf{X}_k - \eta \mathbf{G}_k)\mathbf{W}_k$$

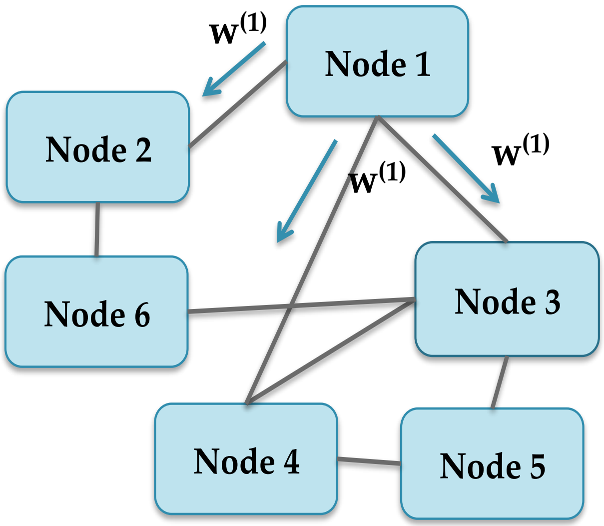
Cooperative SGD: Special Cases

Fully Synchronous $\mathcal{A}(1, \mathbf{1}\mathbf{1}^T/m, 0)$

Periodic/Federated Avg $\mathcal{A}(\tau, \mathbf{1}\mathbf{1}^T/m, 0)$

Elastic Averaging SGD $\mathcal{A}(1, \mathbf{W}_\alpha, 1)$

Decentralized SGD $\mathcal{A}(1, \mathbf{W}, 0)$



and many more variants..

Cooperative SGD: Assumptions

1. Lipschitz smooth

$$||\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})|| \leq L||\mathbf{x} - \mathbf{y}||$$

2. Unbiased Gradients

$$\mathbb{E}_{\xi|\mathbf{x}}[g(\mathbf{x})] = \nabla F(\mathbf{x})$$

3. Bounded Variance

$$\mathbb{E}_{\xi|\mathbf{x}}[||g(\mathbf{x}) - \nabla F(\mathbf{x})||^2] \leq \beta||g(\mathbf{x})||^2 + \sigma^2$$

Cooperative SGD: Error Analysis

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}||\nabla F(\mathbf{u}_k)||^2\right]$$

Average of all the local models

Cooperative SGD: Error Analysis

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}||\nabla F(\mathbf{u}_k)||^2\right] \leq \frac{2(F(\mathbf{x}_1) - F_{\text{inf}})}{\eta_{\text{eff}} K} + \frac{\eta_{\text{eff}} L \sigma^2}{m} + \eta^2 L^2 \sigma^2 \left(\frac{1+\zeta^2}{1-\zeta^2}\tau - 1\right)$$

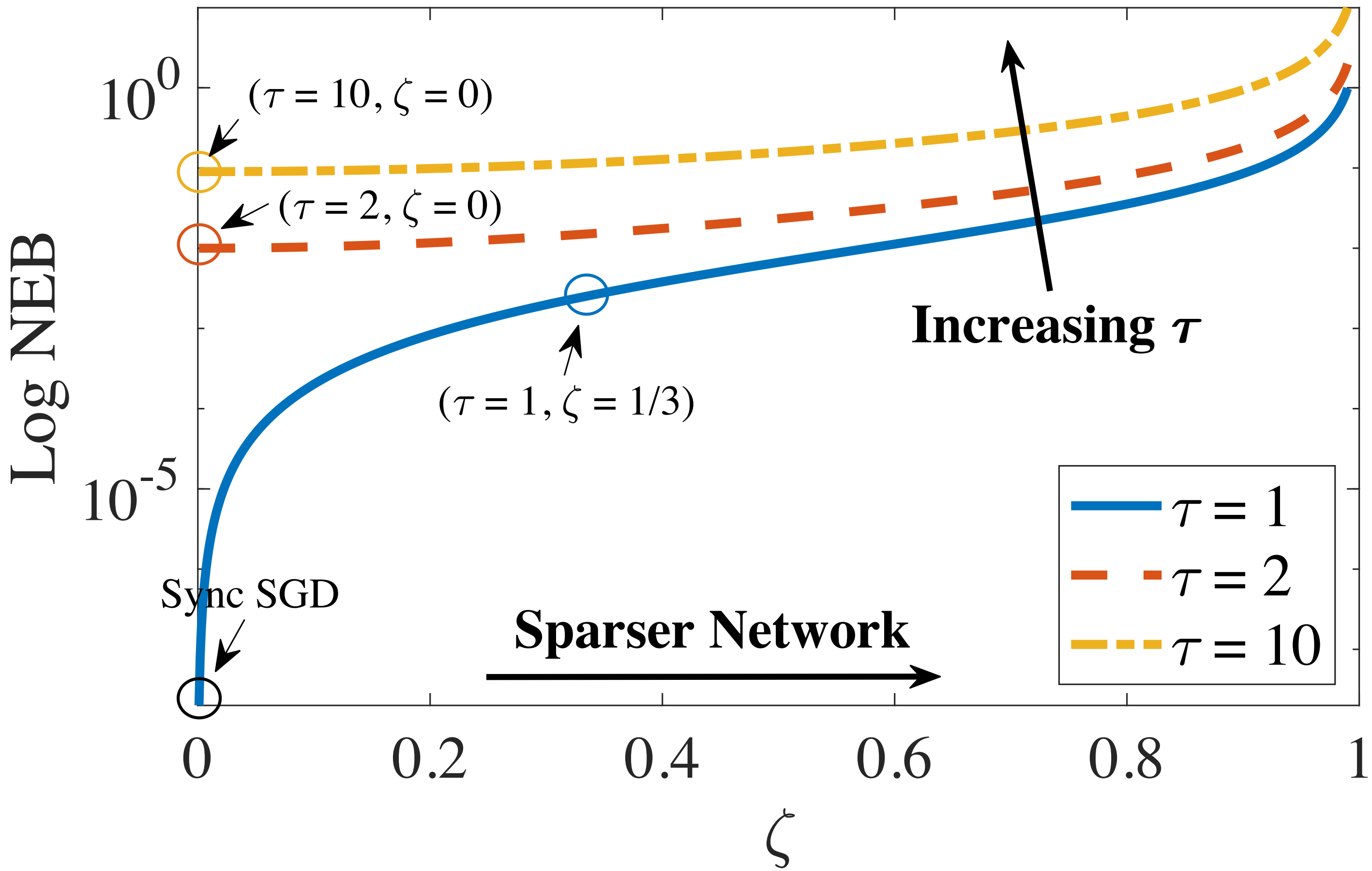
Fully Sync Error

Network Error

$\zeta = \max\{|\lambda_2(\mathbf{W})|, |\lambda_{m+v}(\mathbf{W})|\}$, the spectral Gap of W, which is larger for sparser networks

$\eta_{\text{eff}} = \eta \frac{m}{m+v}$, more auxiliary variables gives slower convergence, but a lower error floor

Cooperative SGD: Error Analysis



Cooperative SGD: Error Analysis

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}||\nabla F(\mathbf{u}_k)||^2\right] \leq \frac{2(F(\mathbf{x}_1) - F_{\text{inf}})}{\eta_{\text{eff}} K} + \frac{\eta_{\text{eff}} L \sigma^2}{m} + \eta^2 L^2 \sigma^2 \left(\frac{1+\zeta^2}{1-\zeta^2}\tau - 1\right)$$

MAIN CONTRIBUTIONS

- First analysis of Elastic Averaging SGD for non-convex objectives. Can show that α=m/m+2 gives minimum error
- Allows comparison of periodic averaging (controlling τ) and decentralized SGD (controlling ζ)

Outline

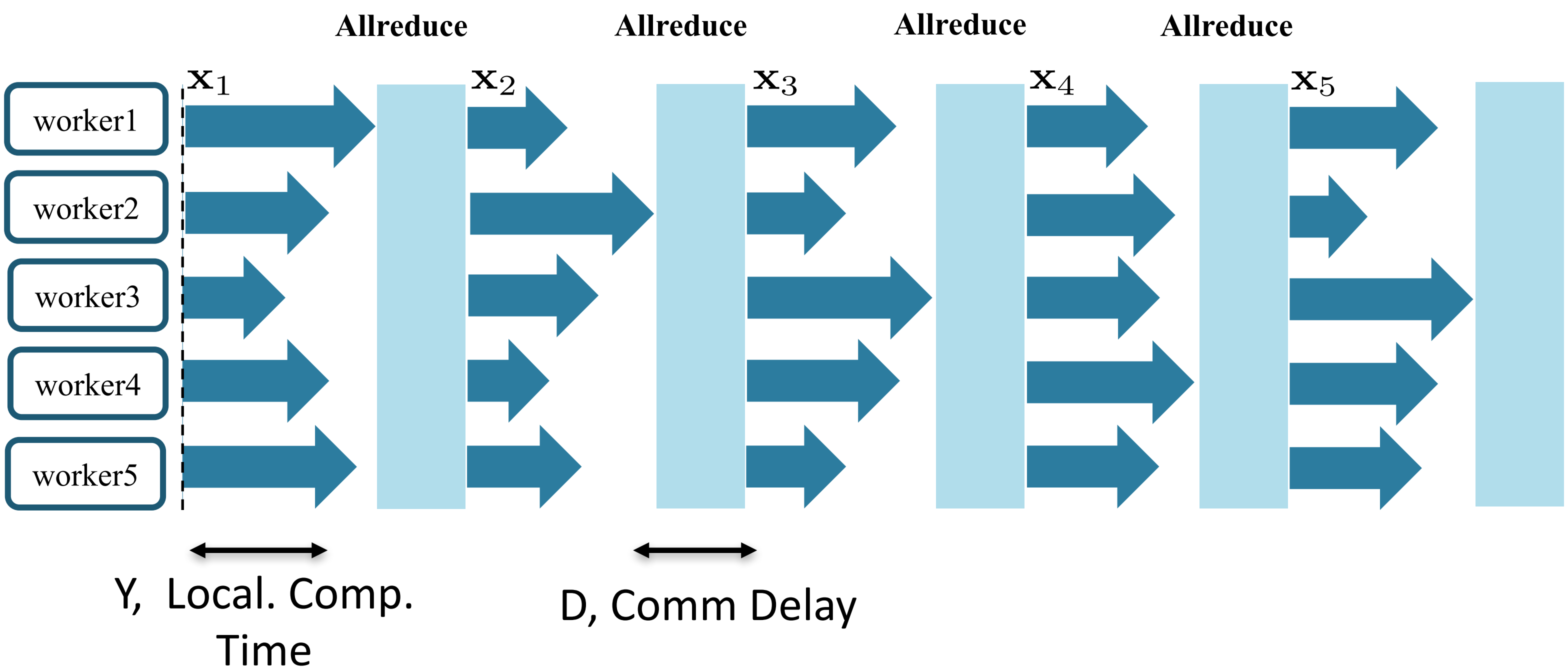
Error Analysis via the Cooperative SGD Framework

Runtime Analysis

Adaptive Communication Strategies

Cooperative SGD: Runtime Per Iteration

Fully Synchronous SGD

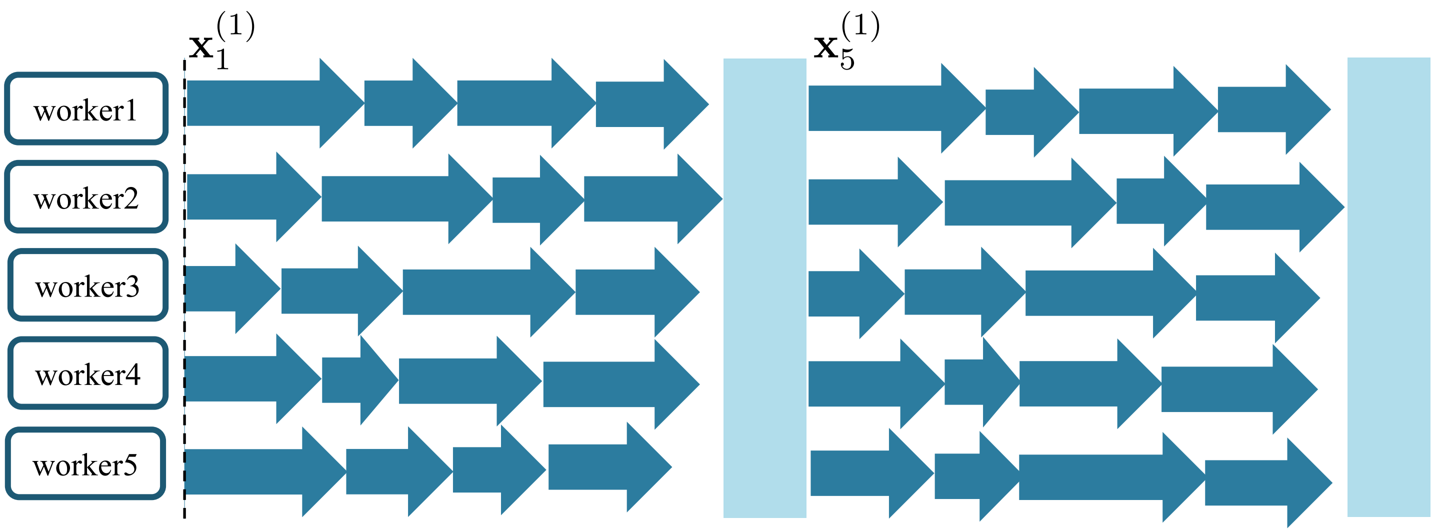


$$T_{\text{sync}} = \max(Y_{1,1}, Y_{2,1}, \ldots, Y_{m,1}) + D$$

$$\mathbb{E}[T_{\text{sync}}] = \mathbb{E}[Y_{m:m}] + \mathbb{E}[D]$$

Cooperative SGD: Runtime Per Iteration

Periodic Averaging SGD



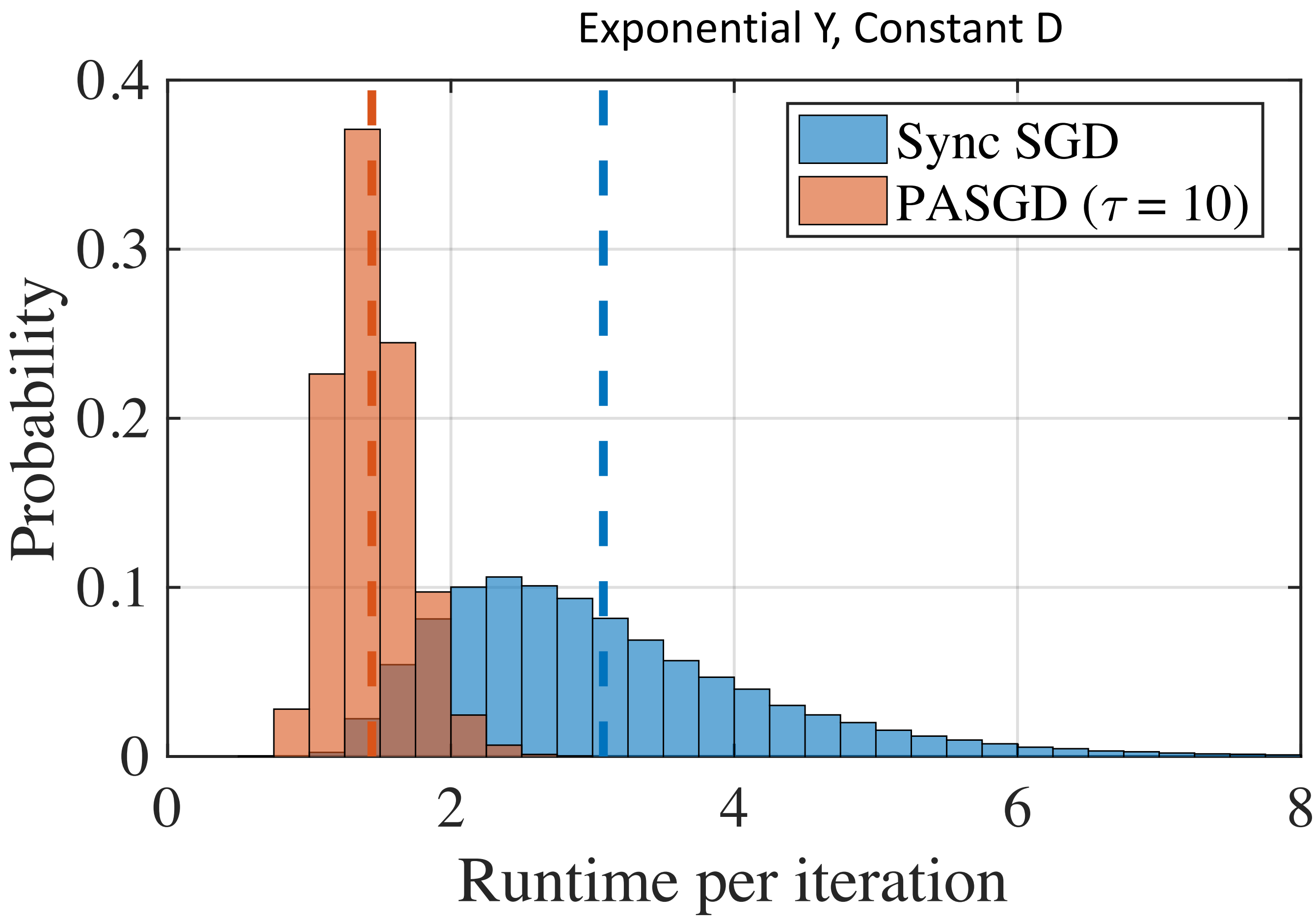
$$T_{\text{P-Avg}} = \max(\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_m) + \frac{D}{\tau}$$

$$\mathbb{E}[T_{\text{P-Avg}}] = \mathbb{E}[\overline{Y}_{m:m}] + \frac{\mathbb{E}[D]}{\tau}$$

Straggler Mitigation due to averaging

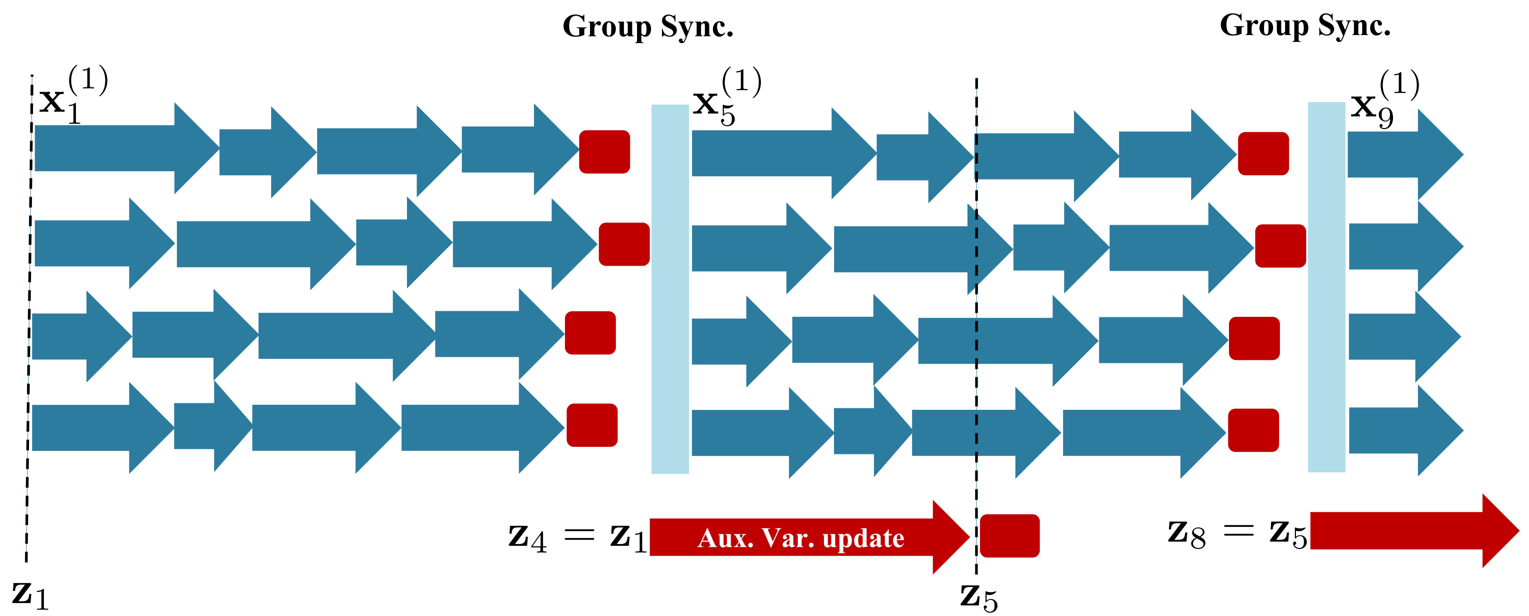
Comm. Delay amortized over τ slots

Cooperative SGD: Runtime Per Iteration



Cooperative SGD: Runtime Per Iteration

Analyzing the effect of mixing matrix W and auxiliary variables is non-trivial and is still open.



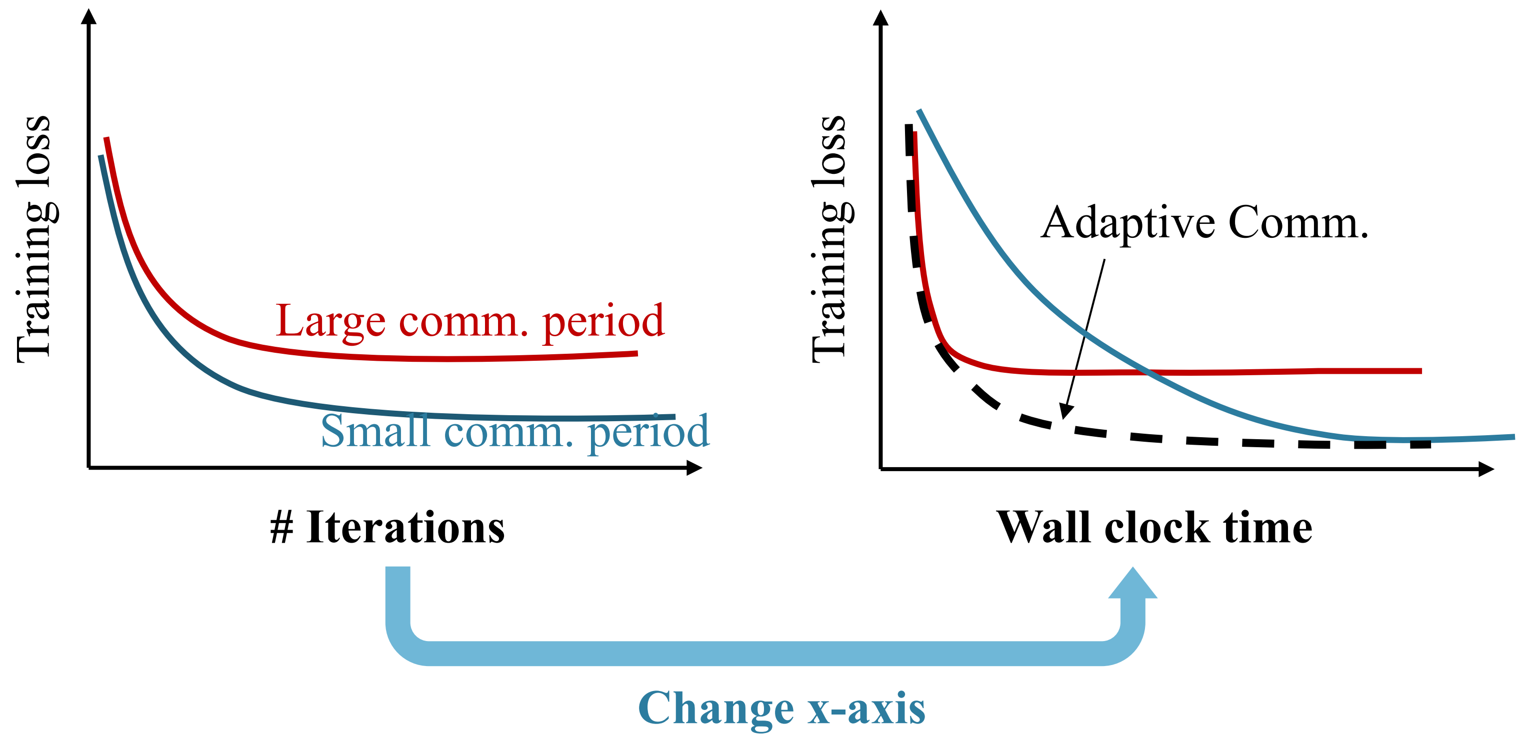
Outline

Error Analysis via the Cooperative SGD Framework

Runtime Analysis

Adaptive Communication Strategies

Error-Runtime Trade-off in Local-Update SGD



Large τ or sparse averaging reduces communication delay

Model discrepancies gives inferior error-convergence

Outline

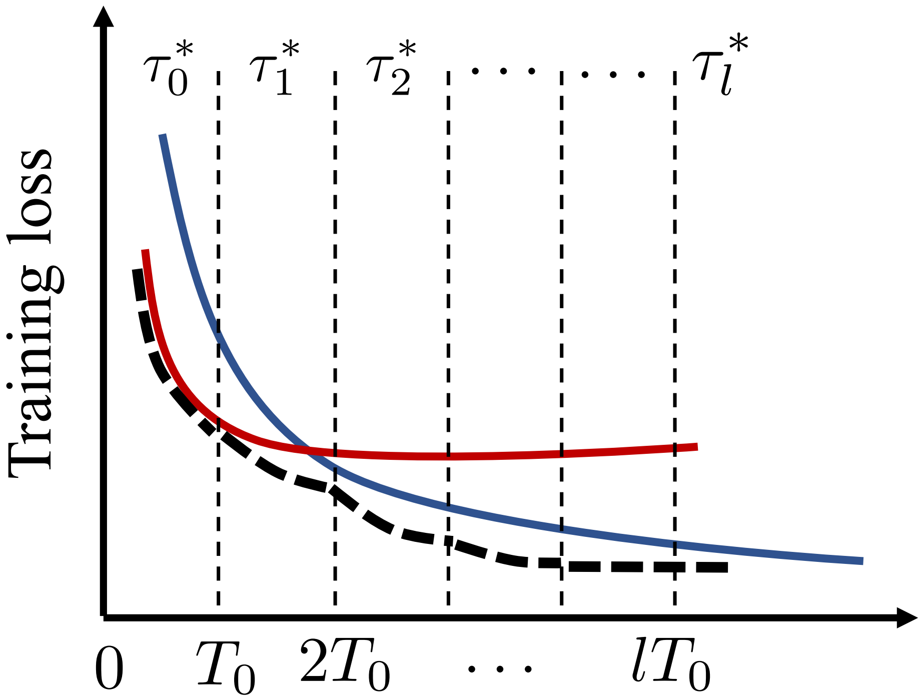
Error Analysis via the Cooperative SGD Framework

Runtime Analysis

Adaptive Communication Strategies

Adaptive Communication Strategy

When to Switch to a Different τ?



Our Approach: Use the error-runtime analysis to decide switching points

Error-Runtime Trade-off

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}||\nabla F(\mathbf{u}_k)||^2\right] \leq \frac{2(F(\mathbf{x}_1) - F_{\text{inf}})}{\eta_{\text{eff}} K} + \frac{\eta_{\text{eff}} L\sigma^2}{m} +$$

$$\eta^2 L^2 \sigma^2 \left(\frac{1+\zeta^2}{1-\zeta^2}\tau - 1\right)$$

Set ζ = 0 to focus on all-reduce communication

Error-Runtime Trade-off

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}||\nabla F(\mathbf{u}_k)||^2\right] \leq \frac{2(F(\mathbf{x}_1) - F_{\text{inf}})}{\eta_{\text{eff}} K} + \frac{\eta_{\text{eff}} L\sigma^2}{m} + \eta_{\text{eff}}^2 L^2 \sigma^2 (\tau - 1)$$

Replace iteration index by wall-clock time

$$\frac{T}{K} = \max(\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_m) + \frac{D}{\tau}$$

$$\approx Y + \frac{D}{\tau}, \text{ for const.} Y, D$$

Error-Runtime Trade-off

$$\text{Error at time } T \leq \frac{2(F(\mathbf{x}_1) - F_{\inf})}{\eta_{\text{eff}} T} \left(Y + \frac{D}{\tau} \right) + \frac{\eta_{\text{eff}} L \sigma^2}{m} + \eta_{\text{eff}}^2 L^2 \sigma^2 (\tau - 1)$$

A heuristic choice of τ is to take the derivative and set to 0

$$\tau * = \sqrt{\frac{2(F(\mathbf{x}_1) - F_{\inf}) D}{\eta^3 L^2 \sigma^2 T}}.$$

Decreases with T

AdaComm Strategy

Can we directly use this in practice?

$$\tau* = \sqrt{\frac{2(F(\mathbf{x}_1) - F_{\text{inf}})D}{\eta^3 L^2 \sigma^2 T}}.$$

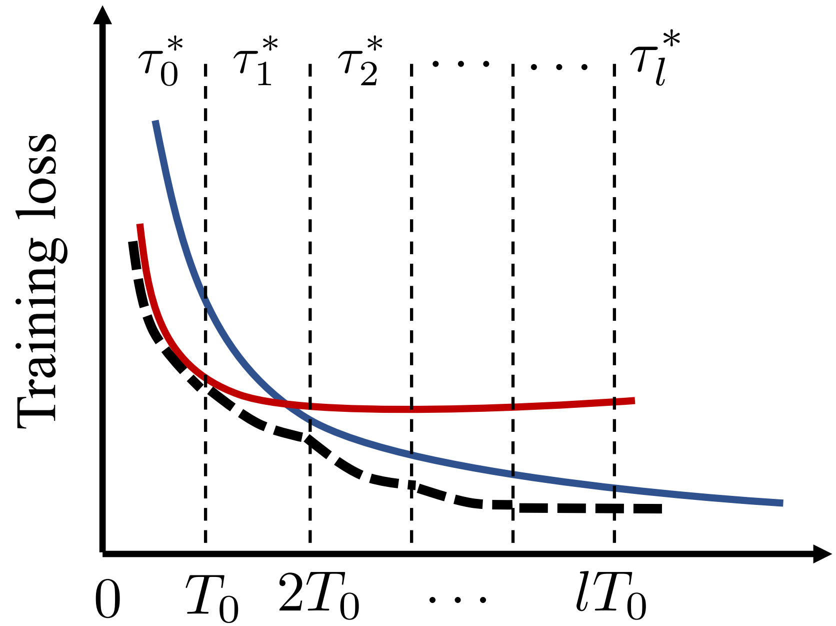
AdaComm Strategy

Unfortunately, no.

We don't know $F_{\text{inf},}$ L, σ in most ML problems

Also, we cannot switch at each time T

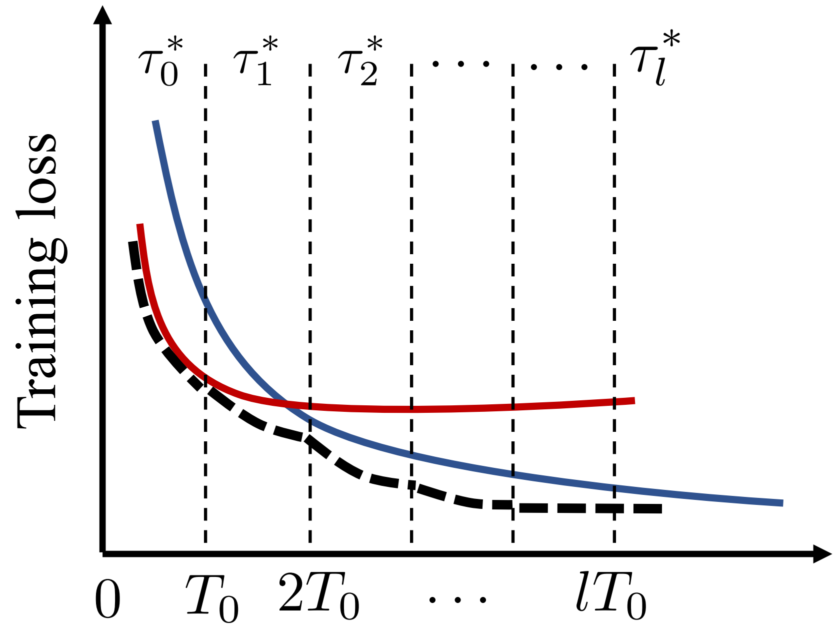
Modifying AdaComm to Account for Practical Constraints



$$\tau_l = \left\lceil \sqrt{\frac{F(\mathbf{x}_{t=lT_0})}{F(\mathbf{x}_{t=0})}}\,\tau_0 \right\rceil$$

Find τ_0 by a grid search

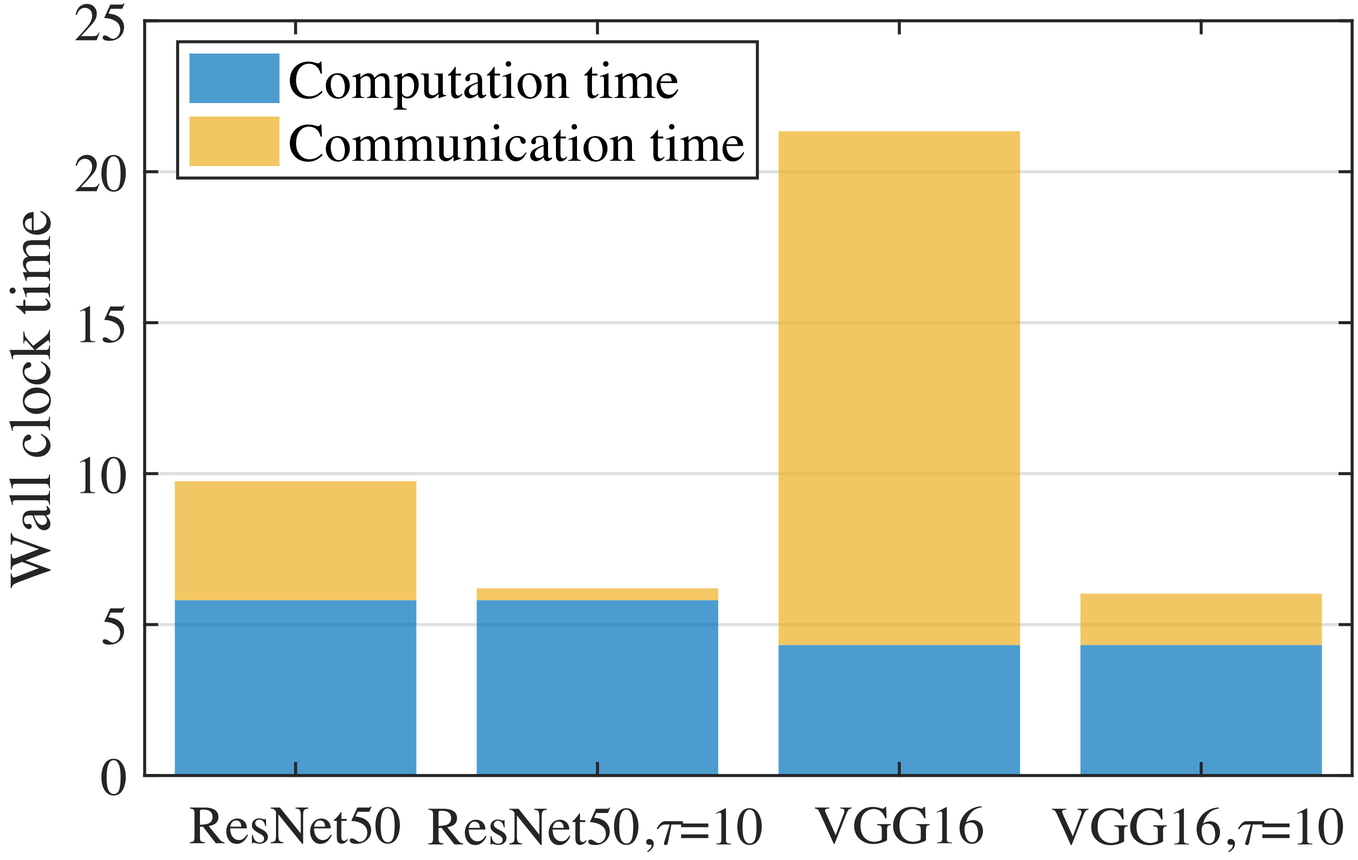
What about learning rate schedules like AdaGrad, Adam etc. ?



$$\tau_l = \left\lceil \sqrt{\frac{\eta_0^3}{\eta_l^3} \frac{F(\mathbf{x}_{t=lT_0})}{F(\mathbf{x}_{t=0})}} \tau_0 \right\rceil$$

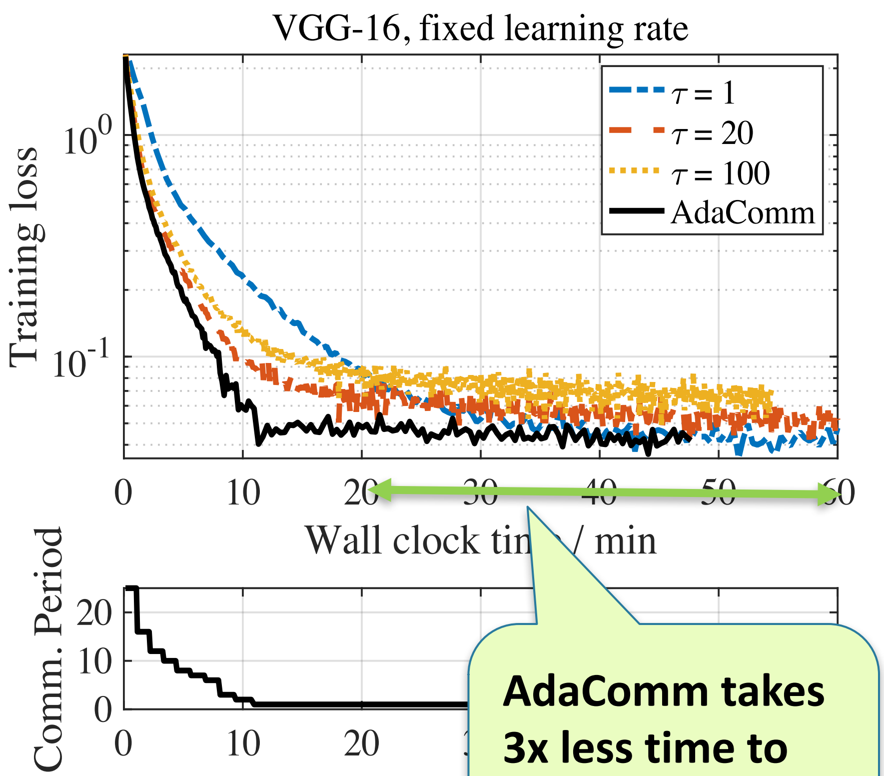
Can increase when η decreases

Experiments on VGG16 and ResNet50

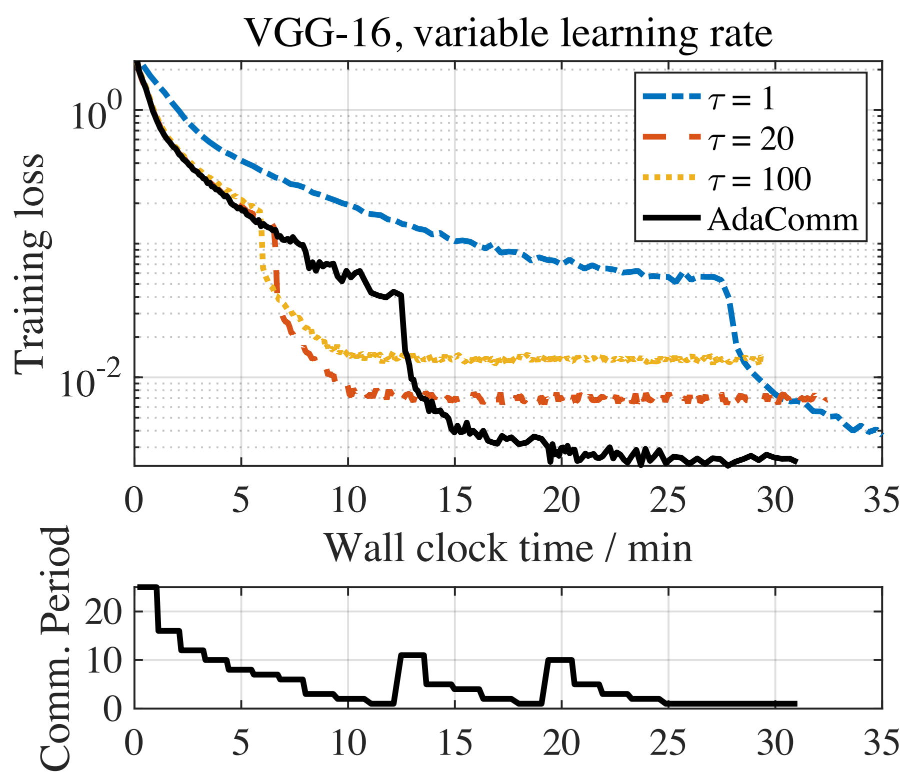


Adaptive Communication Strategy

Experiments on CIFAR10/ 100, VGG16/ResNet-50 with 4 nodes

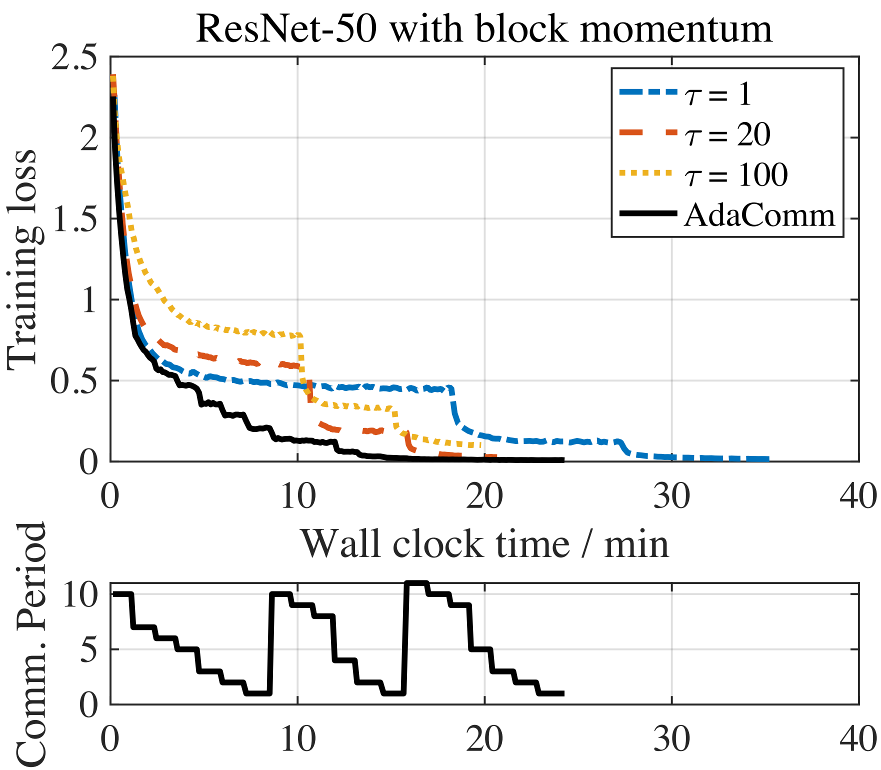


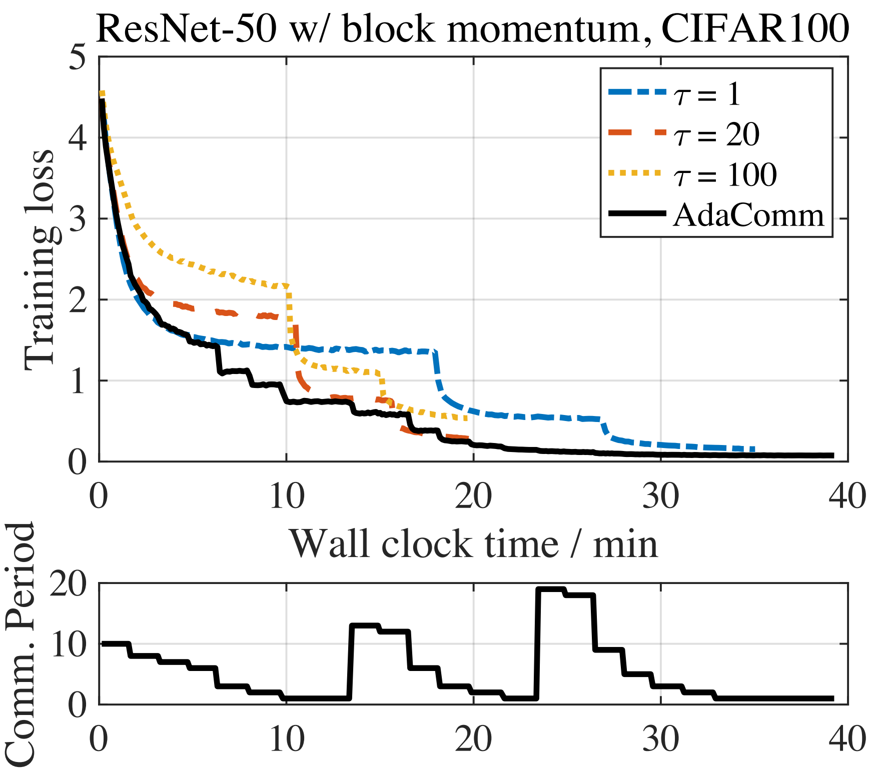
AdaComm takes 3x less time to reach the same training loss



Adaptive Communication Strategy

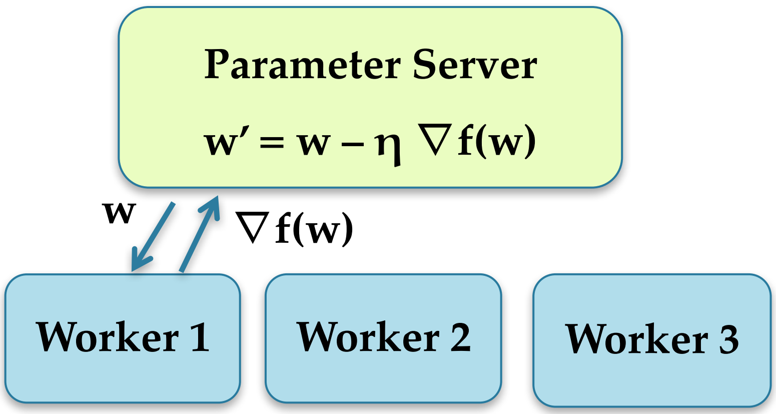
Experiments on CIFAR10/ 100, VGG16/ResNet-50 with 4 nodes

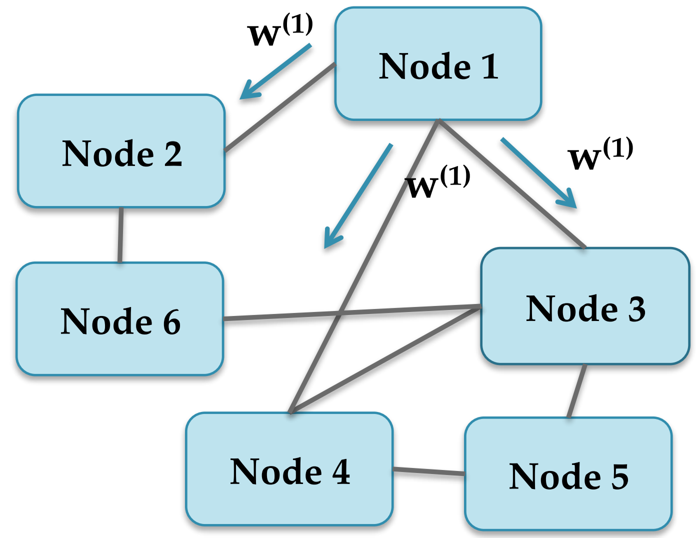




Key Takeaways

Speeding Up Error-Runtime Convergence of Distributed SGD





True SGD convergence is w.r.t. the wall-clock time

Integration of error and runtime reduction strategies

Many Other Interesting Directions in Distributed Machine Learning

- Asynchronous Local-Update SGD Algorithms
- Unevenly distributed and non i.i.d. data
- Model-parallel distributed SGD
- Gradient Compression or Quantization

ArXiV Links to Our Papers

Asynchronous/Synchronous SGD

https://arxiv.org/abs/1803.01113, AISTATS 2018

S. Dutta, G. Joshi, S. Ghosh, P. Dube, P. Nagpurkar

Cooperative SGD Framework

https://arxiv.org/abs/1808.07576, preprint

J. Wang, G. Joshi

Adaptive Communication Strategies for Local-Update SGD

https://arxiv.org/abs/1810.08313, SysML 2019

J. Wang, G. Joshi

Cooperative SGD: Runtime Analysis