# Parallel Restarted SGD for with Faster Convergence and Less Communication: Demystifying Why Model Averaging Works for Deep Learning

Hao Yu
Machine Intelligence Technology
Alibaba Group (US) Inc.

# Distributed Non-Convex Optimization

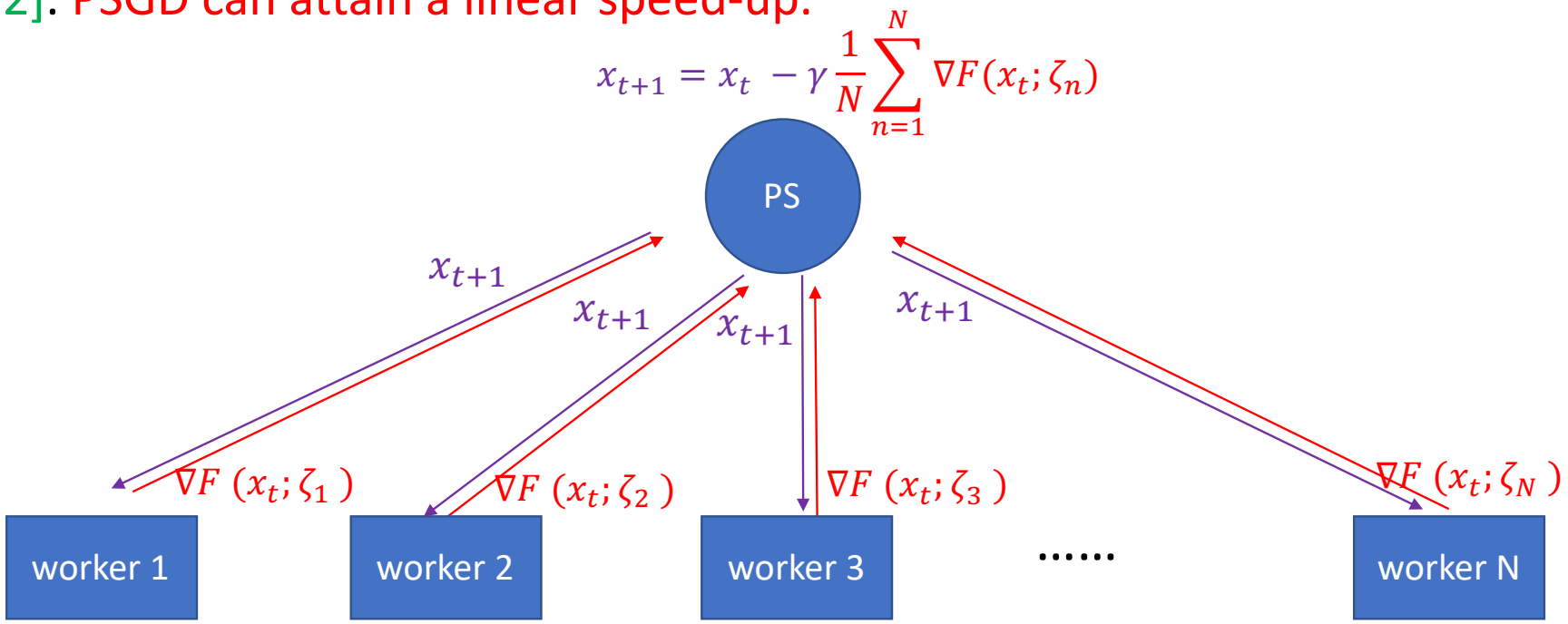
- Large scale non-convex stochastic optimization

$$\min_{x \in \boldsymbol{R}^d} f(x) = \mathrm{E}[F(x;\xi)]$$

- In challenging scenarios, e.g., training deep neural networks,...
  - complicated $F(x;\xi)$ with non-convexity
  - huge number of training samples
- Distributed/parallel optimization
  - With $\mathrm{N}$ machines, can we solve the same opt $\mathrm{N}$ times faster? If yes, then we say the linear speed-up (w.r.t. # of workers) is attained.

# Classical Parallel mini-batch SGD

- The classical Parallel mini-batch SGD (PSGD) achieves $O(\frac{1}{\sqrt{NT}})$ convergence with N workers [Dekel et al. 12]. PSGD can attain a linear speed-up.




- But…each iteration of PSGD requires to aggregate gradients from every workers. Communication overhead is too large!

# Reduce Communication Overhead

- Can we reduce the communication complexity of PSGD while maintaining its linear speed-up?
- Some important attempts are
  - Gradient compression, e.g.,[Alistarh et al. 2017; Wen et al. 2017], uses fewer bits for gradient aggregations
  - Decentralized parallel SGD (DP-SGD) [Lian et al. 2017; Lian et al. 2018] eliminates the gradient aggregation step and exchanges gradients only between neighbors.
  - Neither reduces the number of communication rounds. Not suitable when network links have large latency.
- Question: Can we reduce the number of communication rounds?

# Model Averaging: common practice to reduce communication

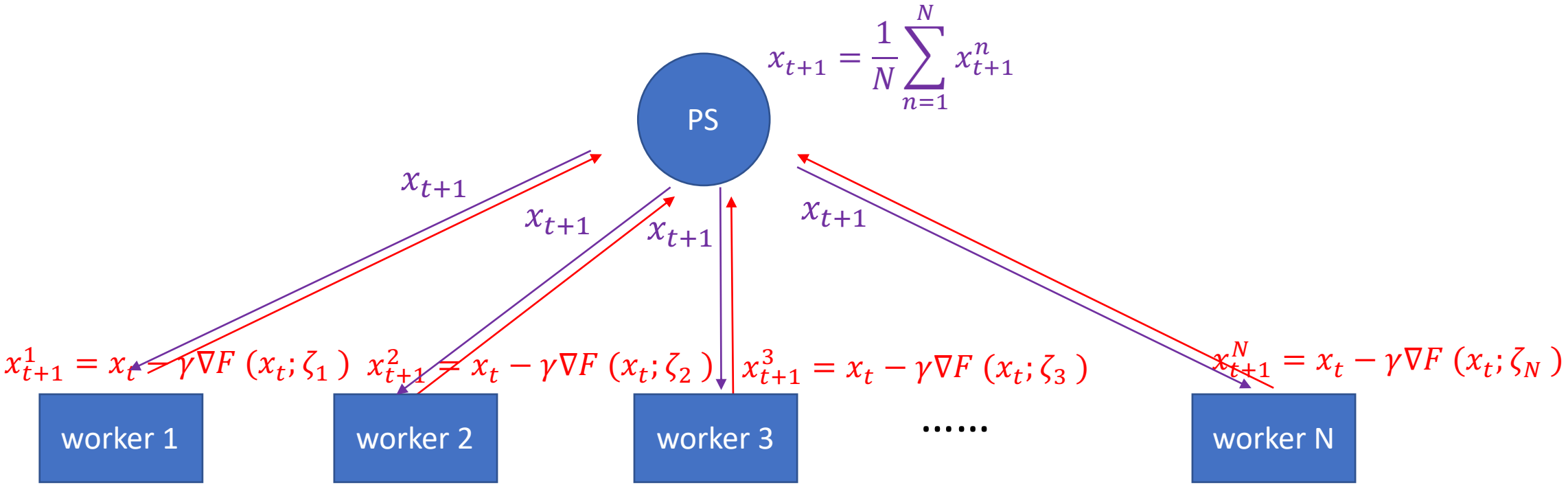
- Model Averaging: Each worker train its local model and used (periodically) averaged versions as the output.
  - **One-shot averaging**: [Zindevich et al. 2010, McDonalt et al. 2010] propose to average only once at the end.
  - [Zhang et al. 2016] shows averaging once can leads to poor solutions for non-convex opt and suggest more frequent averaging.
- Communication happens only when averaging. But how often is enough?
- There has been a long line of empirical works …

# Model Averaging: common practice to reduce communication

- Some empirical works on model averaging
  - [Zhang et al. 2016]: CNN for MNIST
  - [Chen and Huo 2016] [Su, Chen, and Xu 2018] : DNN-GMM for speech recognition
  - [McMahan et al. 2017] :CNN for MNIST and Cifar10; LSTM for language modeling
  - [Kaamp et al. 2018] :CNN for MNIST
  - [Lin, Stich, and Jaggi 2018]: Res20 for Cifar10/100; Res50 for ImageNet
- These empirical works show that ”model averaging” is almost as good as PSGD with significantly less communication overhead!

# Why model averaging works?

- If we average models each iteration, then model averaging is equivalent to PSGD.




- Sounds good? But what if we average with an interval larger than 1?
  - Converge or not? What is convergence rate? lose linear speed-up of PSGD or not?

# Why model averaging works?

- It is mysterious why model averaging (with skipped communication) can work for non-convex opt, e.g., deep learning, as shown by empirical works?
- [Zhou and Cong 2017] shows if we average models every $I$ iterations, then the convergence is $I$ times slower for non-convex opt. 😂
- For strongly convex opt, [Stich 2018] shows the convergence (with linear speed-up w.r.t. # of workers) is maintained as long as the averaging interval is less than $O(\sqrt{T}/\sqrt{N})$.😀
- Still mysterious, why model averaging works for deep learning, which is non-convex! Where does the observed speed-up come from?🤯

# Parallel Restarted SGD (PR-SGD)

**Algorithm 1** Parallel Restarted SGD (PR-SGD)

**Input:** Initialize $\mathbf{x}_i^0 = \overline{\mathbf{y}} \in \mathbb{R}^m$. Set learning rate $\gamma > 0$ and node synchronization interval (integer) $I > 0$

**for** $t = 1$ to $T$ **do**

  Each node $i$ observes an unbiased stochastic gradient $\mathbf{G}_i^t$ of $f_i(\cdot)$ at point $\mathbf{x}_i^{t-1}$

  **if** $t$ is a multiple of $I$, i.e., $t \bmod I = 0$, **then**

    Calculate node average $\overline{\mathbf{y}} \overset{\Delta}{=} \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{t-1}$

    Each node $i$ in parallel updates its local solution

$$\mathbf{x}_i^t = \overline{\mathbf{y}} - \gamma \mathbf{G}_i^t, \quad \forall i \tag{2}$$

  **else**

    Each node $i$ in parallel updates its local solution

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} - \gamma \mathbf{G}_i^t, \quad \forall i \tag{3}$$

  **end if**

**end for**

# Parallel Restarted SGD

- PR-SGD abstracts “synchronous” model averaging.
- This work proves PR-SGD (with communication reduction) has the same convergence rate as the classical parallel mini-batch SGD for non-convex opt.

**Main result in this paper:**

If the averaging interval $I = O(T^{\frac{1}{4}}/N^{\frac{3}{4}})$ ,then PR-SGD has the convergence rate $O(\frac{1}{\sqrt{NT}})$ .

- 😛…now…no surprising why ”model averaging” works for deep learning. It is as fast as PSGD with significantly less communication.

# Technical analysis

- Focus on

$$\bar{x}^t = \frac{1}{N}\sum_{i=1}^{N} x_i^t$$

average of local solution over all $N$ workers

- Note…

$$\bar{x}^t = \bar{x}^{t-1} - \gamma \frac{1}{N}\sum_{i=1}^{N} G_i^t$$

$G_i^t$ : independent gradients sampled at different points $x_i^{t-1}$

- In PSGD, need iid gradients at $\bar{x}^{t-1}$, which is unavailable at each local worker without communication

# Technical analysis

- A simple technical lemma relating deviations between $\bar{x}^t$ and $x_i^t$ to averaging interval $I$

Our Algorithm 1 ensures $E||\bar{x}^t - x_i^t||^2 \leq 4\gamma^2 I^2 G^2, \forall i, \forall t$

- Using the smoothness and following kind of "routine" analysis:

$$\mathbb{E}[f(\overline{\mathbf{x}}^t)] \leq \mathbb{E}[f(\overline{\mathbf{x}}^{t-1})] + \mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1} \rangle] + \frac{L}{2}\mathbb{E}[\|\overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1}\|^2]$$

$$\mathbb{E}[\|\overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1}\|^2] \leq \tfrac{1}{N}\gamma^2\sigma^2 + \gamma^2\mathbb{E}[\|\tfrac{1}{N}\textstyle\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2]$$

$$\mathbb{E}[\langle \nabla f(\overline{\mathbf{x}}^{t-1}), \overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1} \rangle] = -\frac{\gamma}{2}\mathbb{E}\big[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2 + \|\frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2 - \|\nabla f(\overline{\mathbf{x}}^{t-1}) - \frac{1}{N}\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2\big]$$

$$\mathbb{E}[\|\nabla f(\overline{\mathbf{x}}^{t-1}) - \tfrac{1}{N}\textstyle\sum_{i=1}^N \nabla f_i(\mathbf{x}_i^{t-1})\|^2] \leq L^2 \tfrac{1}{N}\textstyle\sum_{i=1}^N \mathbb{E}[\|\overline{\mathbf{x}}^{t-1} - \mathbf{x}_i^{t-1}\|^2]$$

# Technical analysis

- Thm1: If $0 < \gamma \leq \frac{1}{L}$ , then for $T \geq 1$, Algorithm 1 ensures

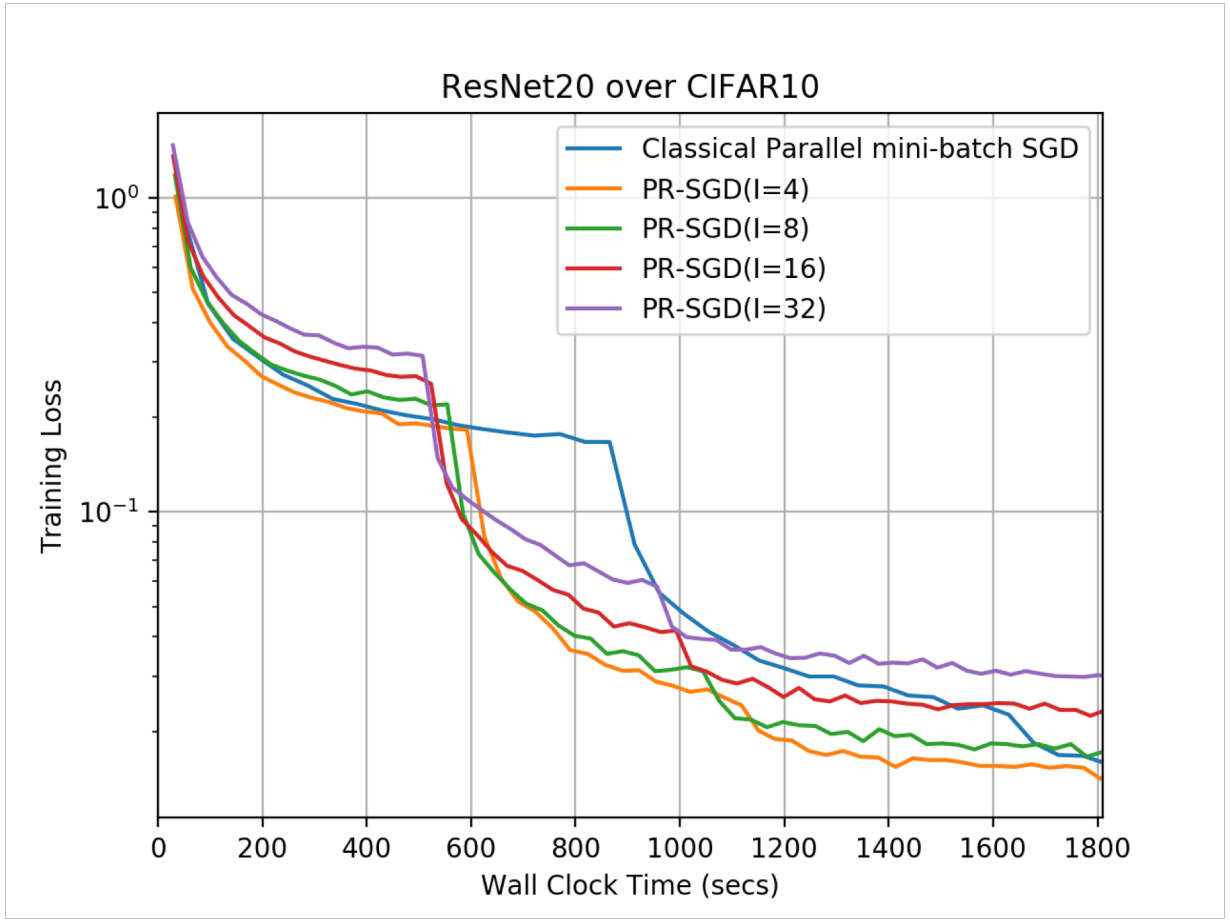
$$\frac{1}{T}\sum_{t=1}^{T} E[\left|\left|\nabla f(\bar{x}^{t-1})\right|\right|^2] \leq \frac{2}{\gamma T}(f(\bar{x}^0) - f^*) + 4\gamma^2 I^2 G^2 L + \frac{L}{N}\gamma\sigma^2$$

- Cor: Taking $\gamma = \frac{\sqrt{N}}{L\sqrt{T}}$ and $I \leq \frac{T^{1/4}}{N^{3/4}}$ yields

$$\frac{1}{T}\sum_{t=1}^{T} E[\left|\left|\nabla f(\bar{x}^{t-1})\right|\right|^2] \leq \frac{1}{\sqrt{NT}}$$

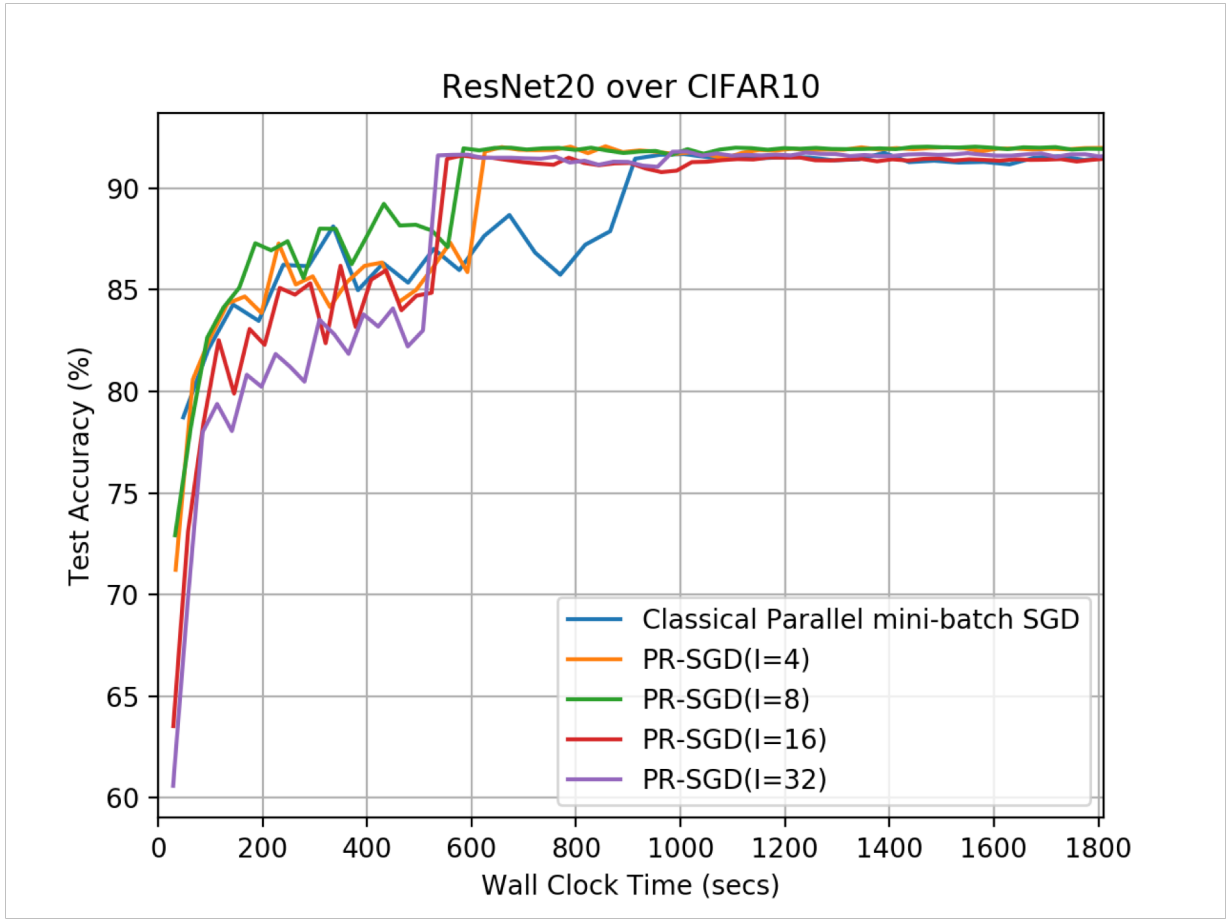
# Experiment

- Run all schemes over a machine with 8 P100 GPUs
- ResNet20 over CIFAR10
- Each GPU uses BS=32, momentum=0.9
- lr=0.1 for 0-150 epochs; =0.01 for 150-225 epochs; =0.001 afterwards



# Experiment

- Run all schemes over a machine with 8 P100 GPUs
- ResNet20 over CIFAR10
- Each GPU uses BS=32, momentum=0.9
- lr=0.1 for 0-150 epochs; =0.01 for 150-225 epochs; =0.001 afterwards



# Extension1: PR-SGD with time-varying learning rate

- Need to develop a new version with possibly time-varying parameters such that the alg's performance improves as it runs

**Algorithm 2** PR-SGD with Time-Varying Learning Rates

**Input:** Set time-varying epoch learning rates $\gamma^s > 0$.
**Initialize:** Initialize $\mathbf{x}_i^{0,K^0} = \overline{\mathbf{x}}^0 \in \mathbb{R}^m$.
**for** epoch index $s = 1$ to $S$ **do**
    Set epoch length $K^s$ and initialize $\mathbf{x}_i^{s,0} = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{s-1,K^{s-1}}$ to be the node average of local worker solutions from the last epoch.
    **for** $k = 1$ to $K^s$ **do**
        Each node $i$ observes an unbiased gradient $\mathbf{G}_i^{s,k}$ of $f_i(\cdot)$ at point $\mathbf{x}_i^{s,k-1}$ and in parallel updates

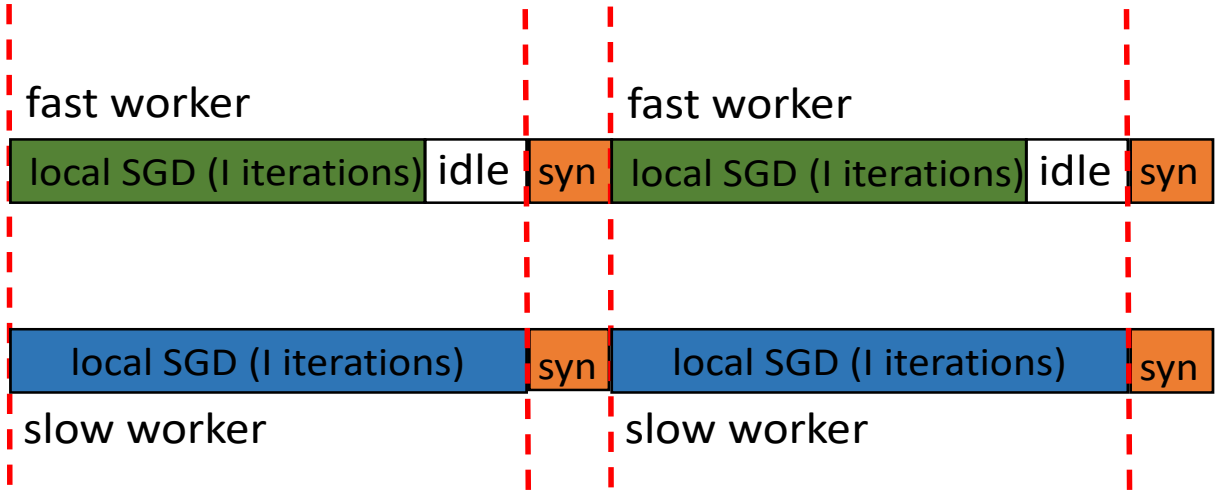
$$\mathbf{x}_i^{s,k} = \mathbf{x}_i^{s,k-1} - \gamma^s \mathbf{G}_i^{s,k}, \quad \forall i$$

    **end for**
**end for**

- Thm: Choosing $K^s = \lfloor \frac{s^{1/3}}{N} \rfloor$ and $\gamma^s = \frac{N}{s^{2/3}}$ in Algorithm 2 ensures $O(\frac{1}{\sqrt{NT}})$ convergence.

# Extesion2: PR-SGD in heterogeneous systems

- Model averaging in heterogeneous systems.
  - Some workers with more advanced hardware are faster.
  - Some workers shared with other ML tasks can be unfortunately slower.
- (Synchronous) Model Averaging:
  - Average all models every $I$ iterations
  - Faster workers have to stay idle to wait for slow workers.



# Extesion2: PR-SGD in heterogeneous systems ("asynchronous" model averaging)

**Algorithm 3** PR-SGD in Heterogeneous Networks

1: **Input:** Set learning rate $\gamma > 0$ and epoch length of each worker $i$ as $I_i$.
2: **Initialize:** Initialize $\mathbf{x}_i^{0,I_i} = \overline{\mathbf{x}}^0 \in \mathbb{R}^m$.
3: **for** epoch index $s = 1$ to $S$ **do**
4: Initialize $\mathbf{x}_i^{s,0} = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{s-1,I_i}$ as the node average of local worker solutions from the last epoch.
5: Each worker $i$ in parallel runs its local SGD for $I_i$ iterations via:

$$\mathbf{x}_i^{s,k} = \mathbf{x}_i^{s,k-1} - \gamma \mathbf{G}_i^{s,k}, \quad \forall i$$

where $\mathbf{G}_i^{s,k}$ is an unbiased stochastic gradient at point $\mathbf{x}_i^{s,k-1}$.
6: **end for**

- We show if let faster worker run more iterations and slower worker run fewer iterations before model averaging, such "asynchronous" model averaging can be at least as fast as "synchronous" model averaging. If the initial solution is poor, Algorithm 3 is faster than Algorithm 1.