北京大学
PEKING UNIVERSITY

A Unified Algorithm for Non-IID Decentralized Learning

Kun Yuan

Center for Machine Learning Research @ Peking University

Nov. 26, 2023

Joint work with

Sulaiman A. Alghunaim

(Kuwait University)

Xinmeng Huang

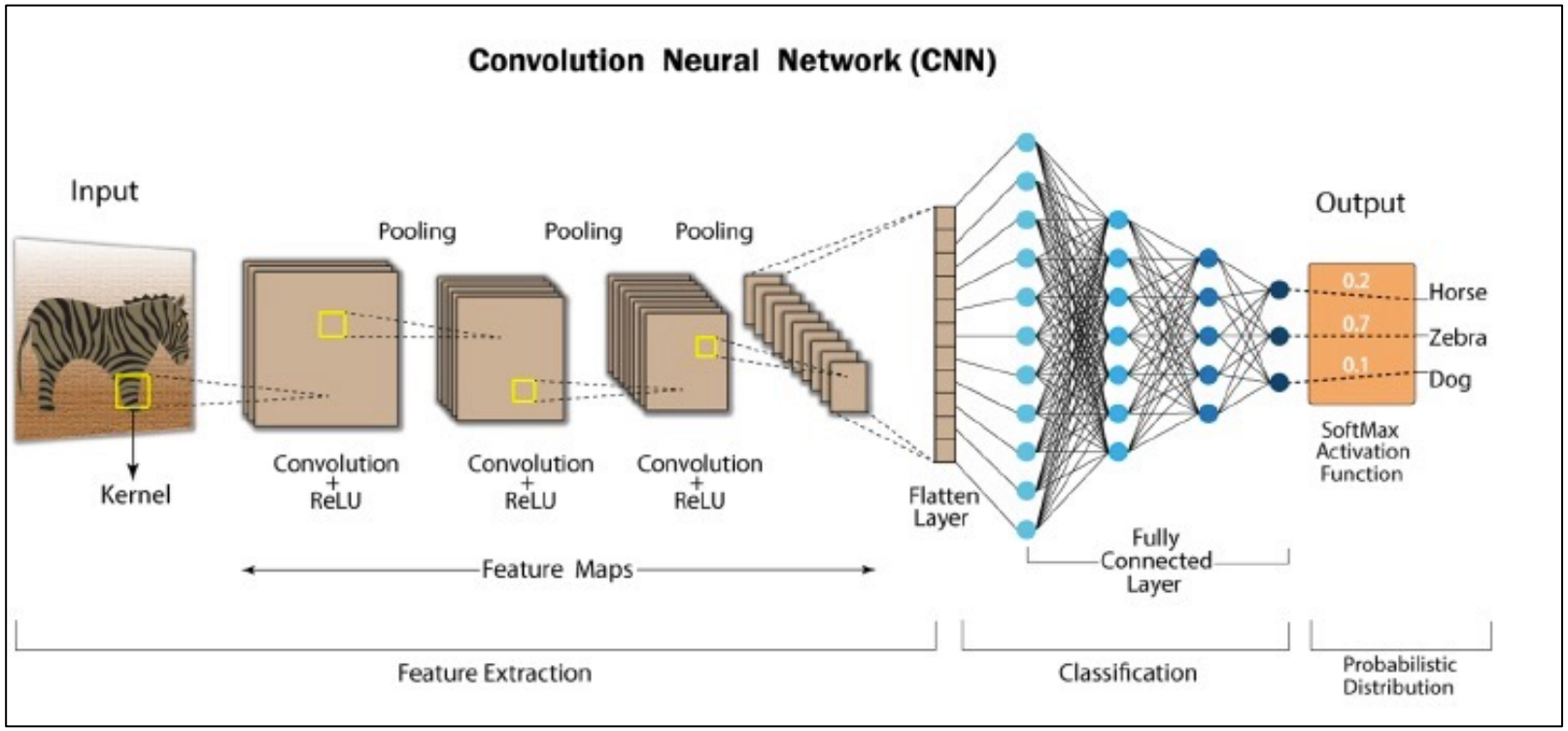
(University of Pennsylvania)

北京大学
PEKING UNIVERSITY

Preface

Basics and Motivation

Training deep neural network is notoriously difficult



DNN training = non-convexity + **massive dataset** + huge models

Distributed learning

北京大学 PEKING UNIVERSITY

- Training deep neural networks typically requires **massive** datasets; efficient and scalable distributed optimization algorithms are in urgent need

- A network of n nodes (devices such as GPUs) collaborate to solve the problem:

$$\min_{x\in\mathbb{R}^d} \quad f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad f_i(x)=\mathbb{E}_{\xi_i\sim D_i} F(x;\xi_i)$$

  - Each component $f_i : \mathbb{R}^d \to \mathbb{R}$ is local and private to node i
  - Random variable ξ_i denotes the local data that follows distribution D_i
  - Each local distribution D_i is different; data heterogeneity exists

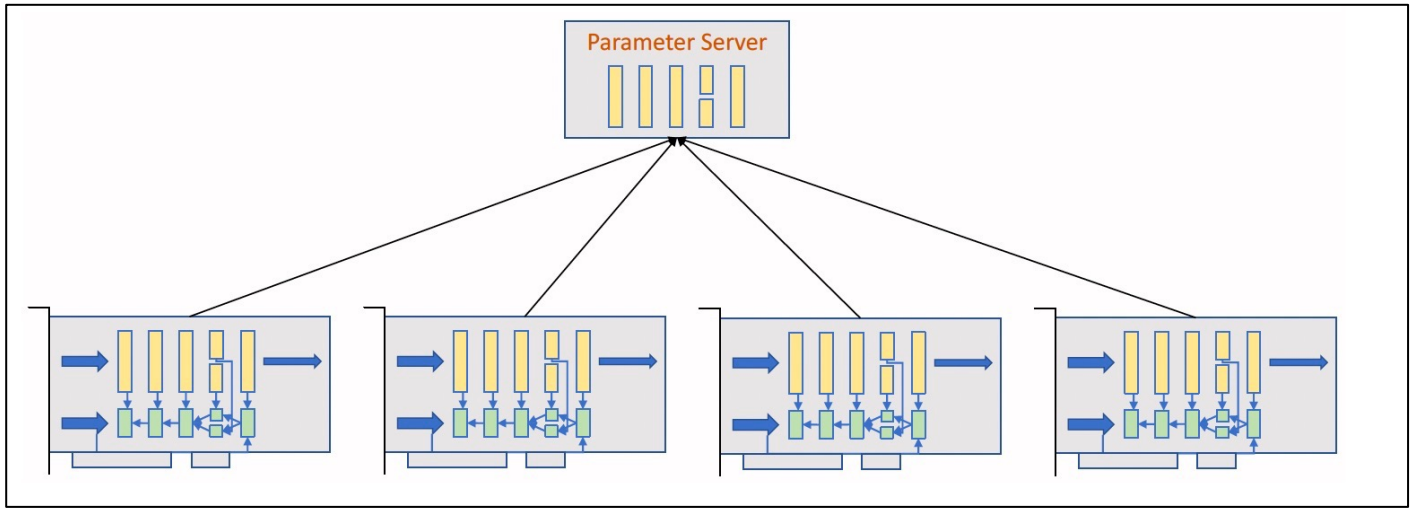
Vanilla parallel stochastic gradient descent (PSGD)

$$g_i^{(k)} = \nabla F(x^{(k)}; \xi_i^{(k)}) \qquad \text{(Local compt.)}$$

$$x^{(k+1)} = x^{(k)} - \frac{\gamma}{n} \sum_{i=1}^{n} g_i^{(k)} \qquad \text{(Global comm.)}$$

- Each node i samples data $\xi_i^{(k)}$ and computes gradient $\nabla F(x^{(k)}; \xi_i^{(k)})$
- All nodes synchronize (i.e. globally average) to update model x per iteration

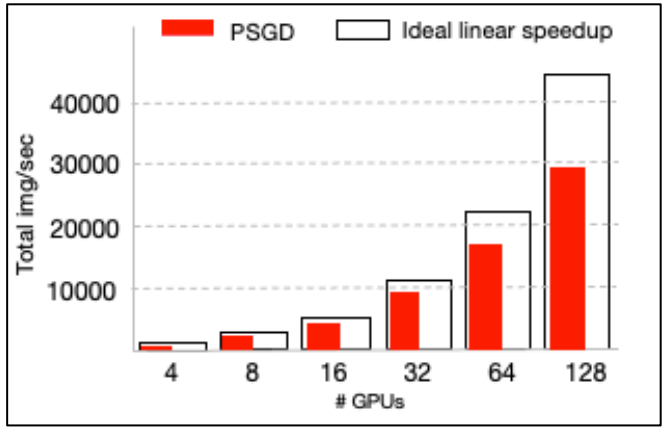
Vanilla parallel stochastic gradient descent (PSGD)



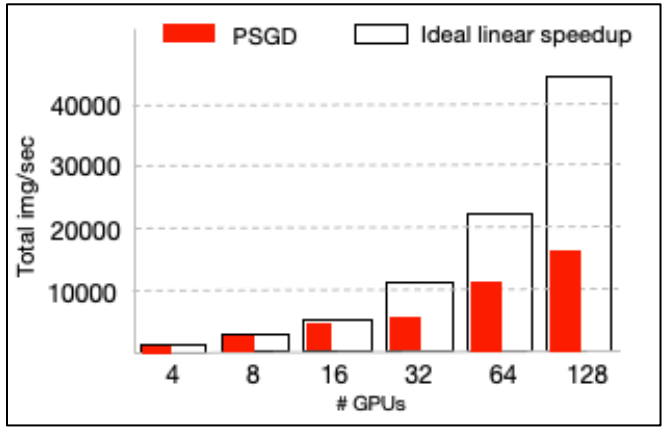
- Global average incurs $O(n)$ comm. overhead; **proportional to network size n**
- When network size n is large, PSGD suffers severe communication overhead

PSGD cannot achieve linear speedup due to comm. overhead

- PSGD cannot achieve ideal linear speedup in throughput due to comm. overhead
- Larger comm-to-compt ratio leads to worse performance in PSGD



Small comm.-to-compt. ratio



Large comm.-to-compt. ratio

- How can we accelerate PSGD? **Decentralized SGD is a promising paradime**

B. Ying, K. Yuan, H. Hu, Y. Chen and W. Yin, “BlueFog: Make decentralized algorithms practical for optimization and deep learning”, arXiv: 2111. 04287, 2021

北京大学
PEKING UNIVERSITY

PART 01

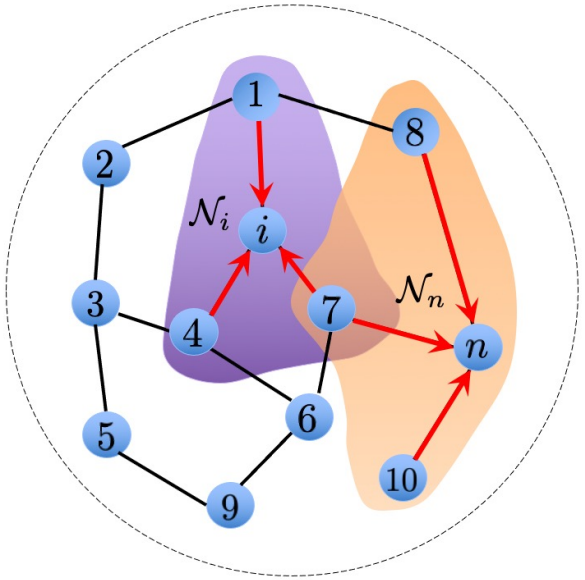
Decentralized SGD and Topology Effects

Decentralized SGD (DSGD)

- To break $O(n)$ comm. overhead, we replace global average with partial average

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) \quad \text{(Local update)}$$

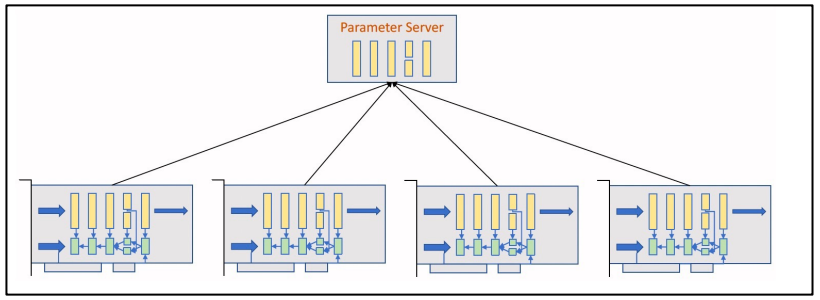
$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k+\frac{1}{2})} \quad \text{(Partial averaging)}$$



- DSGD = local SGD update + partial averaging [LS08]
- $\mathcal{N}_i$ is the set of neighbors at node i ; w_{ij} scales information from j to i and satisfies $\sum_{j \in \mathcal{N}_i} w_{ij} = 1$
- Incurs $O(d_{\max})$ comm. overhead per iteration where $d_{\max} = \max_i \{|\mathcal{N}_i|\}$ is the graph maximum degree

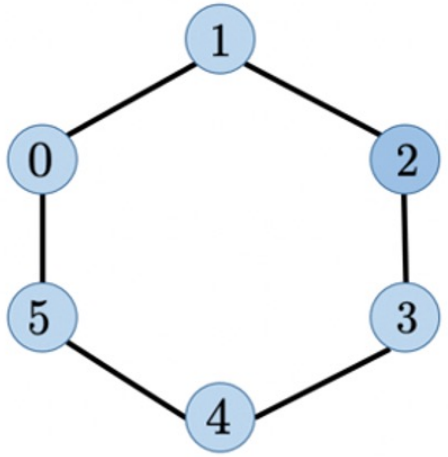
DSGD is more communication-efficient than PSGD

- Incurs $O(1)$ comm. overhead on **sparse** topologies; much less than global average $O(n)$



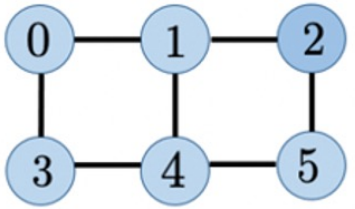
Centralized training

$O(n)$



Ring

$O(1)$



Grid

DSGD is more communication-efficient than PSGD

- A real experiment on a 256-GPUs cluster [CYZ+21]

Model	Ring-Allreduce	Partial average
ResNet-50 (25.5M)	278 ms	150 ms

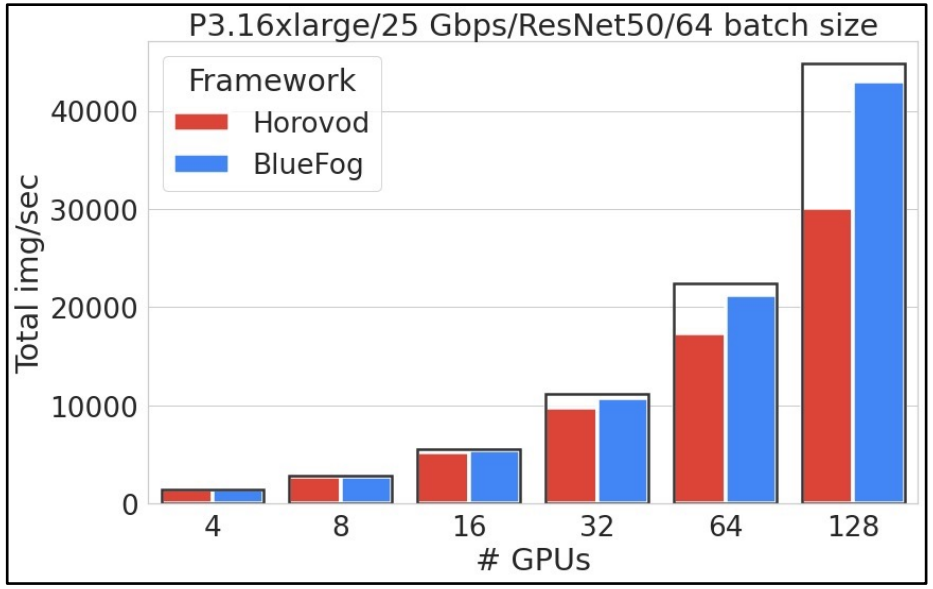
Table. Comparison of per-iter comm. time in terms of runtime with 256 GPUs

- DSGD saves more communications per iteration for larger models

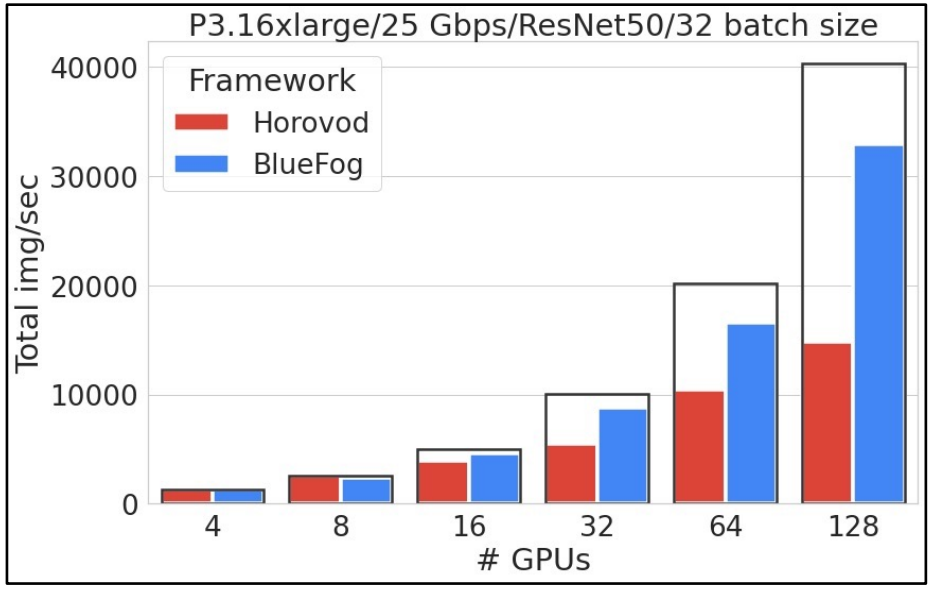
[CYZ+21] Y. Chen*, K. Yuan*, Y. Zhang, P. Pan, Y. Xu, and W. Yin, ``Accelerating Gossip SGD with Periodic Global Averaging", ICML 2021

DSGD is more communication-efficient than PSGD

- DSGD (BlueFog) has **better linear speedup** than PSGD (Horovod) due to its small comm. overhead



Small comm.-to-compt. ratio



Large comm.-to-compt. ratio

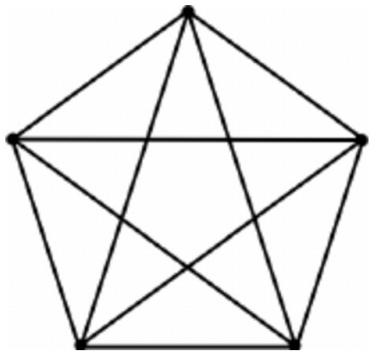
B. Ying, K. Yuan, H. Hu, Y. Chen and W. Yin, “BlueFog: Make decentralized algorithms practical for optimization and deep learning”, arXiv: 2111. 04287, 2021

However, DSGD has slower convergence

- The efficient comm. comes with a cost: slower convergence
- Partial average $x_i^+ = \sum w_{ij} x_j$ is less effective to aggregate information than global average
- The average effectiveness can be evaluated by **graph spectral gap**:

$$\rho = \|W - \frac{1}{n}\mathbb{1}\mathbb{1}^T\|_2 \in (0,1) \text{ where } W = [w_{ij}] \in \mathbb{R}^{n\times n}$$

Fully-connected matrix



$$W = \frac{1}{5}\mathbb{1}\mathbb{1}^T = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$$

However, DSGD has slower convergence

- The efficient comm. comes with a cost: slower convergence
- Partial average $x_i^+ = \sum w_{ij} x_j$ is less effective to aggregate information than global average
- The average effectiveness can be evaluated by **graph spectral gap**:

$$\rho = \|W - \frac{1}{n}\mathbb{1}\mathbb{1}^T\|_2 \in (0,1) \text{ where } W = [w_{ij}] \in \mathbb{R}^{n\times n}$$

  - Well-connected topology has $\rho \to 0$, e.g. fully-connected topology
  - Sparsely-connected topology has $\rho \to 1$, e.g. ring has $\rho = O(1 - \frac{1}{n^2})$
  - ρ or $1-\rho$ essentially gauges the **graph connectivity**

DSGD convergence rate

- Convergence comparison (non-convex and **data-homogeneous** scenario) [KLB+20]:

$$\text{P-SGD}: \quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}}\right)$$

$$\text{D-SGD}: \quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}} + \underbrace{\frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}}}_{\text{extra overhead}}\right)$$

where σ^2 is the gradient noise, and T is the number of iterations

- D-SGD can asymptotically converge as fast as P-SGD when $T \to \infty$; the first term dominates; reach **linear speedup** asymptotically

- But D-SGD **requires more iteration** to reach that stage due to the overhead caused by partial average

Transient iterations

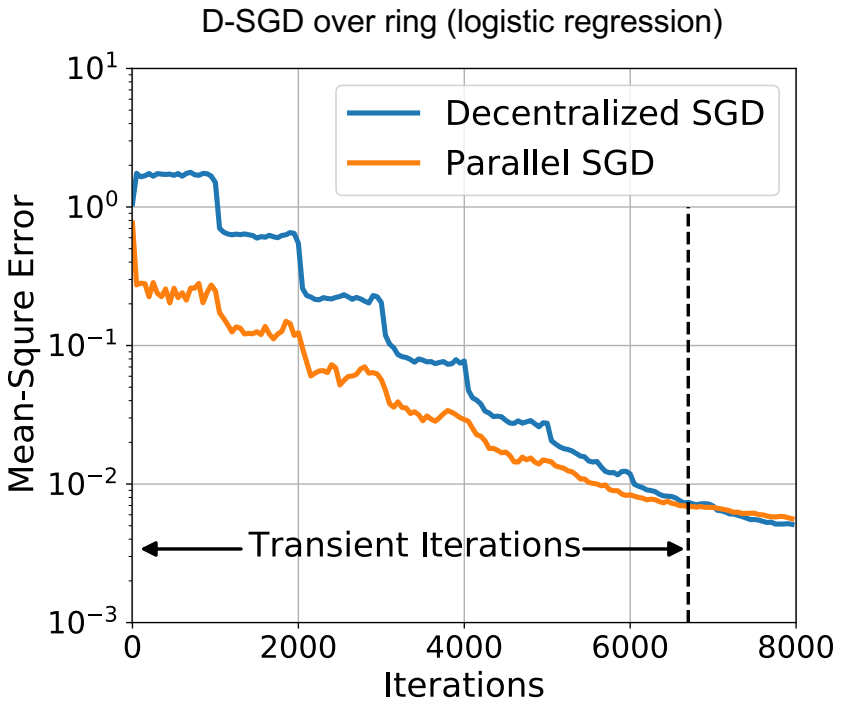
- Definition [POP21]: number of iterations before D-SGD achieves linear speedup

- D-SGD for non-convex and data-homogeneous scenario has $O(n^3(1-\rho)^{-2})$ transient iterations

$$\frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}} \leq \frac{\sigma}{\sqrt{nT}} \quad \Longrightarrow \quad O(\frac{\rho^4 n^3}{(1-\rho)^2})$$

- Topology significantly influence the trans. stage.

- Sparse topology $\rho \to 1$ incurs longer tran. Iters.



北京大学
PEKING UNIVERSITY

PART 02

Data Heterogeneity leads to longer transient stage

Data heterogeneity causes even longer transient stage

- Recall the distributed stochastic optimization problem

$$\min_{x\in\mathbb{R}^d} \quad f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad f_i(x)=\mathbb{E}_{\xi_i\sim D_i}F(x;\xi_i).$$

- If local data distribution $\mathcal{D}_i$ is different from each other, we have

$$f_i(x)\neq f_j(x) \quad \text{when} \quad i\neq j$$

- We assume the gradient dissimilarity is upper bounded

$$\frac{1}{n}\sum_{i=1}^{n}\|\nabla f_i(x)-\nabla f(x)\|^2\leq b^2, \qquad \forall\, x\in\mathbb{R}^d$$

We term b^2 as **data heterogeneity**. When $b^2=0$, we have $\nabla f_i(x)=\nabla f(x)$ which reduces to homogeneity

Data heterogeneity causes even longer transient stage

- D-SGD convergence (non-convex and data-heterogeneous scenario) [KLB20+]

$$\text{P-SGD:} \quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}}\right)$$

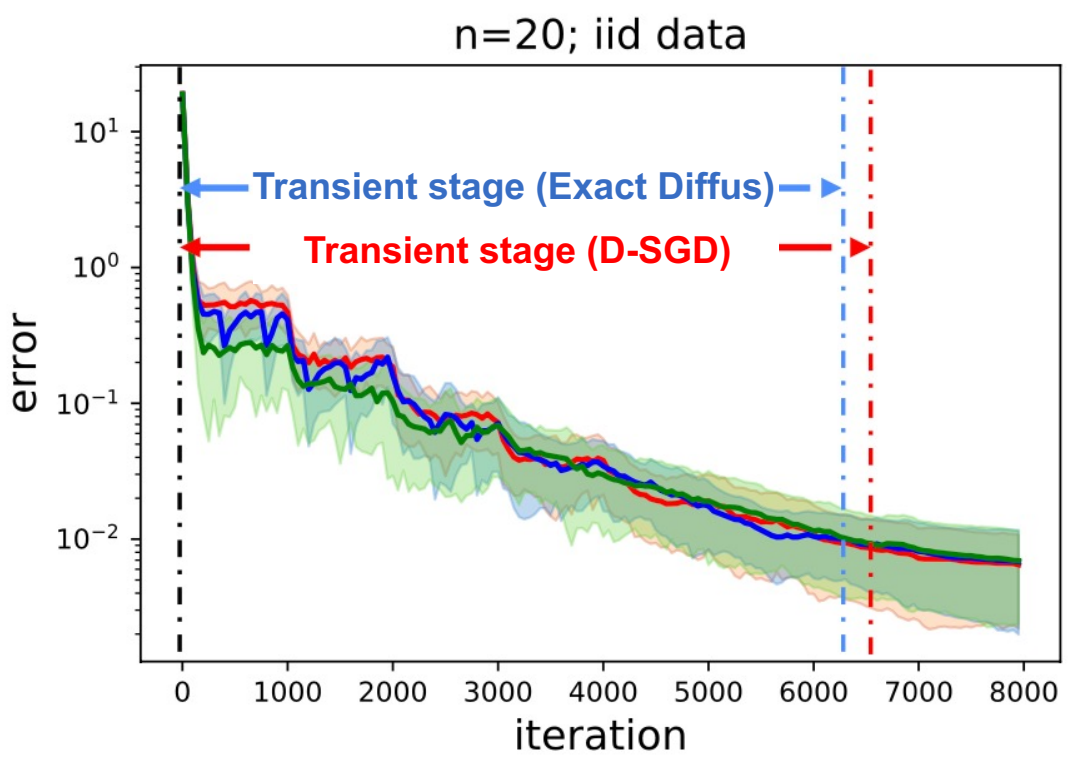
$$\text{D-SGD:} \quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}} + \frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}} + \frac{\rho^{2/3}b^{2/3}}{T^{2/3}(1-\rho)^{2/3}}\right)$$

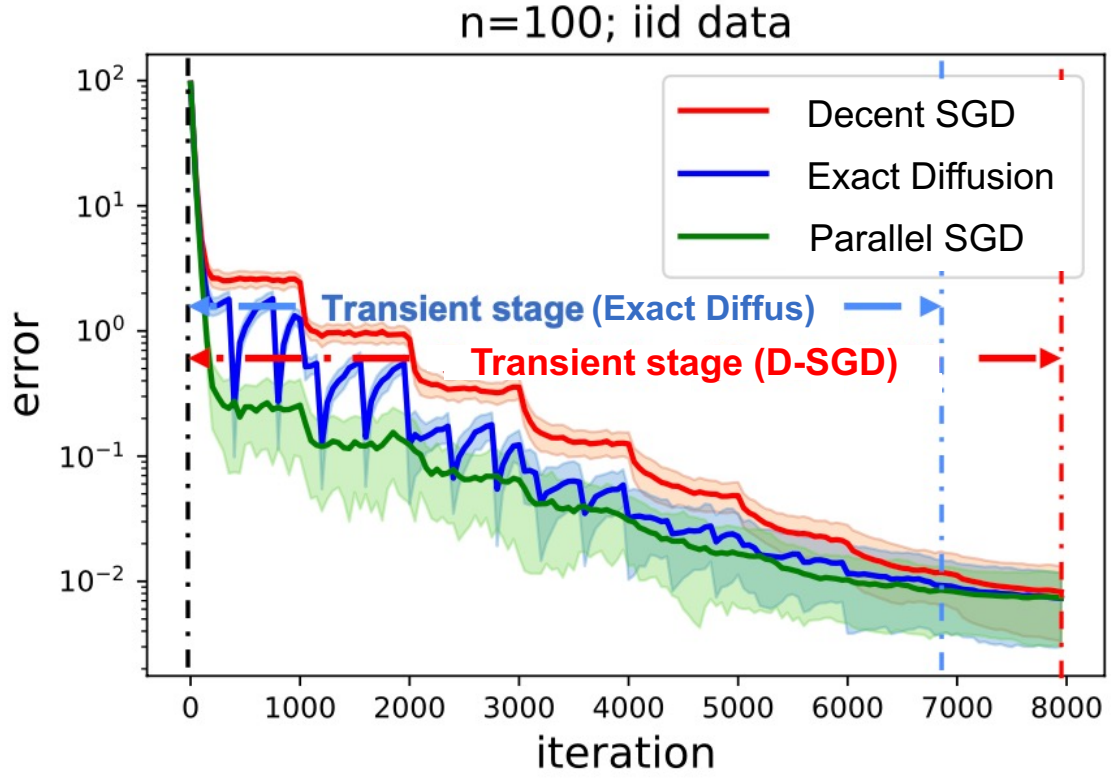
- Convergence gets slower due to additional data heterogeneity term
- For sparse topology in which $\rho \to 1$, transient stage gets significantly **worse**

Homogeneous: $O(\frac{n^3}{(1-\rho)^2})$ Heterogeneous: $O(\frac{n^3}{(1-\rho)^4})$

Data heterogeneity causes even longer transient stage

- Decentralized logistic regression with ring topology and **iid data**

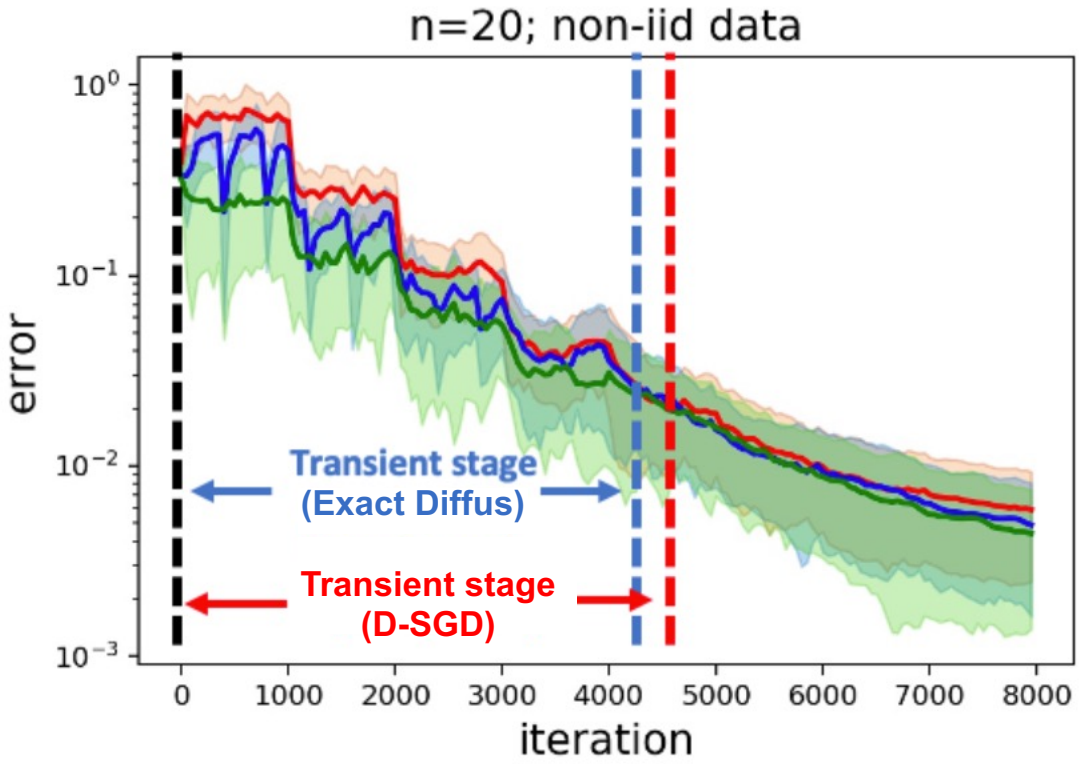


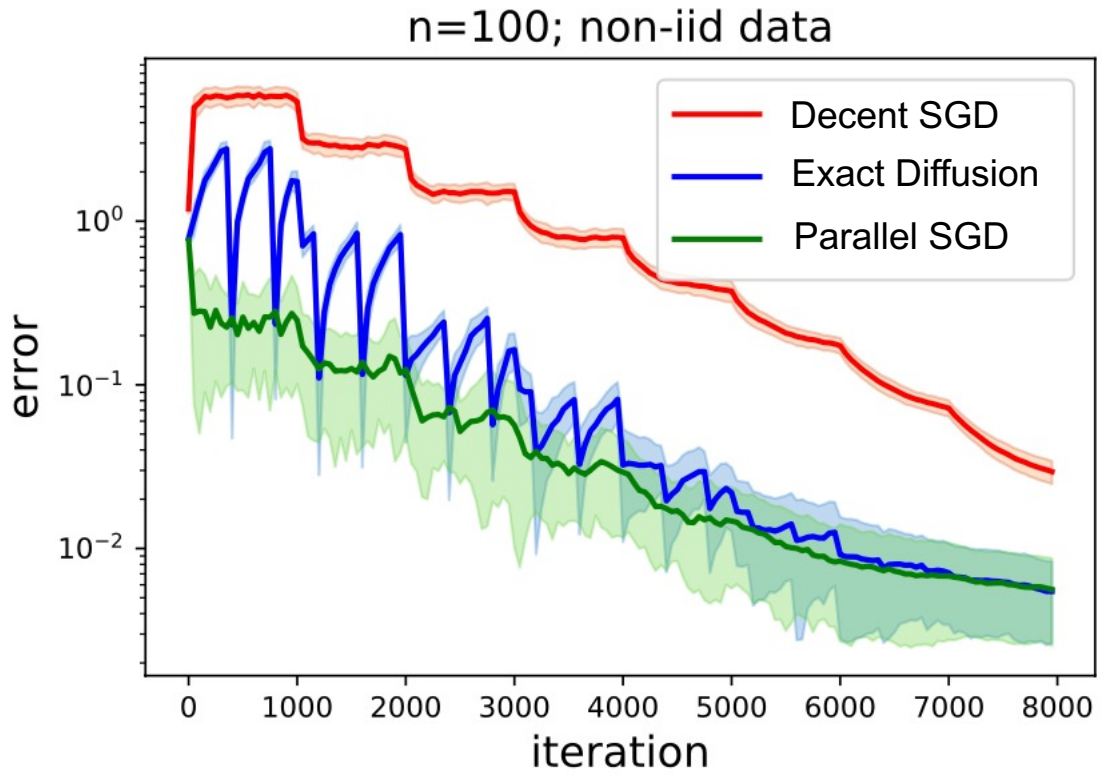


- When topology gets sparser, the transient stage of D-SGD gets longer due to $O(\frac{n^3}{(1-\rho)^2})$

Data heterogeneity causes even longer transient stage

- Decentralized logistic regression with ring topology with **non-iid data**





- When topology gets sparser, the transient stage of D-SGD gets **significantly longer** due to $O(\frac{n^3}{(1-\rho)^4})$

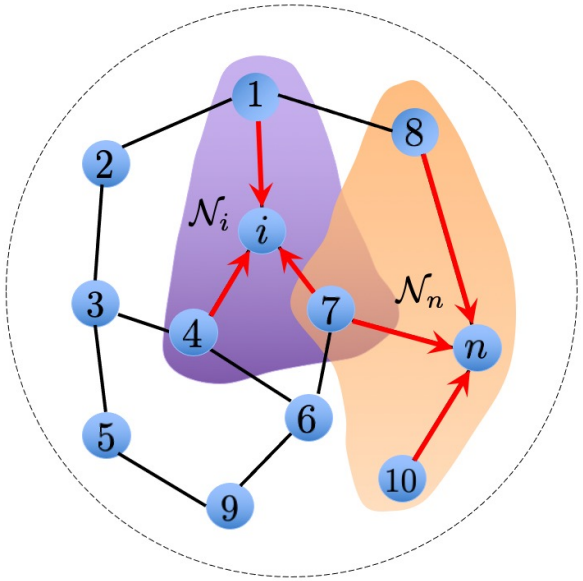
Why does D-SGD suffer from data heterogeneity?

- Recall the distributed stochastic optimization problem

$$\min_{x\in\mathbb{R}^d} \quad f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad f_i(x)=\mathbb{E}_{\xi_i\sim D_i}F(x;\xi_i).$$

- The D-SGD algorithm iterates as follows

$$\begin{aligned} x_i^{(k+\frac{1}{2})} &= x_i^{(k)} - \gamma\nabla F(x_i^{(k)};\xi_i^{(k)}) && \text{(Local update)} \\ x_i^{(k+1)} &= \sum_{j\in\mathcal{N}_i} w_{ij}x_j^{(k+\frac{1}{2})} && \text{(Partial averaging)} \end{aligned}$$



- We can write the above algorithm into one line

$$x_i^{(k+1)} = \sum_{j\in\mathcal{N}_i} w_{ij}\left(x_j^{(k)} - \gamma\nabla F(x_j^{(k)};\xi_j^{(k)})\right)$$

Why does D-SGD suffer from data heterogeneity?

- Suppose no gradient noise exists and all nodes stay at the stationary solution, i.e.,

$$\nabla F(x_i^{(k)};\xi_i^{(k)}) = \nabla f(x_i^{(k)}) \quad \text{and} \quad x_1^{(k)} = x_2^{(k)} = \cdots = x_n^{(k)} = x^\star$$

- The D-SGD algorithm becomes

$$x_i^{(k+1)} = \sum_{j\in\mathcal{N}_i} w_{ij}\left(x^\star - \gamma\nabla f_j(x^\star)\right) = x^\star - \gamma\sum_{j\in\mathcal{N}_i} w_{ij}\nabla f_j(x^\star)$$

- In homogeneous-data scenario where $f_i(x) = f(x)$ for any node i, we have $\nabla f_i(x^\star) = \nabla f(x^\star) = 0$ and

$$x_i^{(k+1)} = x^\star - \gamma\sum_{j\in\mathcal{N}_i} w_{ij}\nabla f_j(x^\star) = x^\star$$

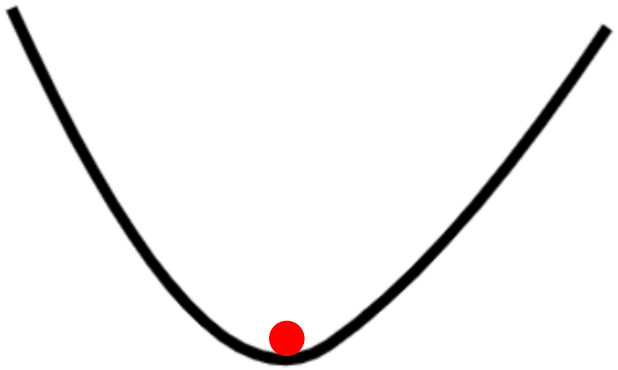
  Therefore, the stationary solution is stable in homogeneous-data scenario

Why does D-SGD suffer from data heterogeneity?

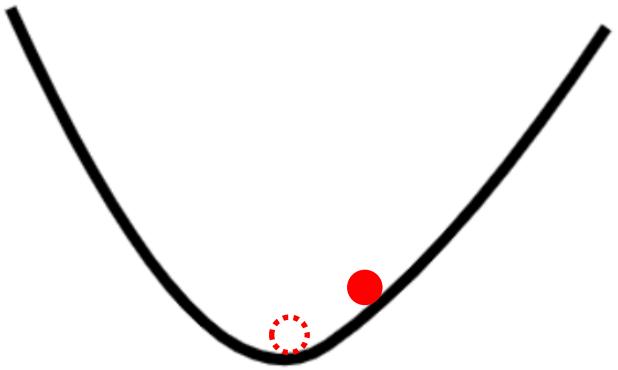
- In heterogeneous-data scenario where $\nabla f_i(x) \neq \nabla f(x)$ for any node i, we have $\nabla f_i(x^\star) \neq \nabla f(x^\star)$ and

$$x_i^{(k+1)} = x^\star - \gamma \sum_{j \in \mathcal{N}_i} w_{ij} \nabla f_j(x^\star) \neq x^\star - \gamma \nabla f(x^\star) = x^\star$$

Therefore, the stationary solution is NOT stable in heterogeneous-data scenario



$x_i^{(k)} = x^\star$



$x_i^{(k+1)} \neq x^\star$

- This non-stability is caused by the algorithmic structure, not the gradient noise

PART 03

Exact-Diffusion: An Effective Method to Remove Data Heterogeneity

Exact-diffusion corrects data heterogeneity

北京大学 PEKING UNIVERSITY

- Various efforts have been made to correct data heterogeneity, such as EXTRA [SLWY15] and gradient tracking [XZSX15, NOS17]

- We propose a new method called **Exact-Diffusion** [YYZS19]

$$\psi_i^{(k)} = x_i^{(k)} - \gamma \nabla f_i(x_i^{(k)}) \qquad \text{(Local update)}$$

$$\phi_i^{(k)} = \psi_i^{(k)} + x_i^{(k)} - \psi_i^{(k-1)} \qquad \text{(Correction)}$$

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} \phi_j^{(k)} \qquad \text{(Partial averaging)}$$

- If we remove the correction term $x_i^k - \psi_i^{(k-1)}$, Exact-Diffusion reduces to D-SGD

K. Yuan, B. Ying, X. Zhao, A. Sayed, "Exact Diffusion for Distributed Optimization and Learning - Part I: Algorithm Development", IEEE TSP 2019

Exact-diffusion corrects data heterogeneity

$$\psi_i^{(k)} = x_i^{(k)} - \gamma \nabla f_i(x_i^{(k)}) \qquad \text{(Local update)}$$

$$\phi_i^{(k)} = \psi_i^{(k)} + x_i^{(k)} - \psi_i^{(k-1)} \qquad \text{(Correction)}$$

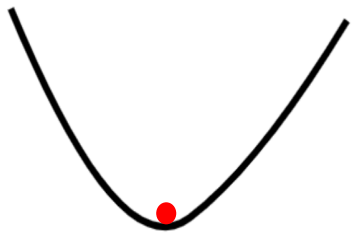
$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} \phi_j^{(k)} \qquad \text{(Partial averaging)}$$

- Suppose $x_i^{(k-1)} = x_i^{(k)} = x^\star$, we have $\psi_i^{(k-1)} = \psi_i^{(k)} = x^\star - \gamma \nabla f_i(x^\star)$, which implies that

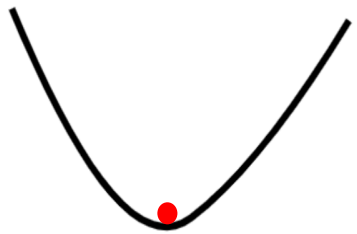
$$\phi_i^{(k)} = \psi_i^{(k)} + x_i^{(k)} - \psi_i^{(k-1)} = x^\star \quad \Longrightarrow \quad x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} \phi_j^{(k)} = x^\star$$

- $x^\star$ is **stable** for Exact-Diffusion

- Exact-Diffusion **corrects data heterogeneity!**



$x_i^{(k)} = x^\star$



$x_i^{(k+1)} = x^\star$

Exact-Diffusion has a shorter transient stage with non-iid data

- Exact-Diffusion with stochastic gradient

$$
\begin{aligned}
\psi_i^{(k)} &= x_i^{(k)} - \gamma \nabla F(x_i^{(k)}; \xi_i^{(k)}) && \text{(Local update)} \\
\phi_i^{(k)} &= \psi_i^{(k)} + x_i^{(k)} - \psi_i^{(k-1)} && \text{(Correction)} \\
x_i^{(k+1)} &= \sum_{j \in \mathcal{N}_i} w_{ij} \phi_j^{(k)} && \text{(Partial averaging)}
\end{aligned}
$$

- Exact-Diffusion with stochastic gradient is also called D2 which was studied in [TLYZ+18]

Exact-Diffusion has a shorter transient stage with non-iid data

北京大学 PEKING UNIVERSITY

Assumption 1. Each $f_i(x)$ is L-smooth

Assumption 2. The stochastic gradient satisfies

$$\mathbb{E}[\nabla F(x;\xi_i)] = \nabla f_i(x)$$
$$\mathbb{E}\|\nabla F(x;\xi_i) - \nabla f_i(x)\|^2 \le \sigma^2$$

Assumption 3. The weight matrix satisfies $W = W^T$, $W \succ 0$ and

$$\|W - \frac{1}{n}\mathbf{1}\mathbf{1}^T\| \le \rho$$

Exact-Diffusion has a shorter transient stage with non-iid data

Theorem 1. (Informal) Under Assumption 1-3, Exact-Diffusion with heterogenous data will converge as follows

$$\frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}} + \frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}}\right)$$

- Recall D-SGD with heterogeneous data will converge as follows

$$\text{D-SGD:}\quad \frac{1}{T}\sum_{k=1}^{T}\mathbb{E}\|\nabla f(\bar{x}^{(k)})\|^2 = O\left(\frac{\sigma}{\sqrt{nT}} + \frac{\rho^{2/3}\sigma^{2/3}}{T^{2/3}(1-\rho)^{1/3}} + \frac{\rho^{2/3}b^{2/3}}{T^{2/3}(1-\rho)^{2/3}}\right)$$

- Exact-Diffusion eliminates the influence of data heterogeneity and shorten the transient stage

D-SGD: $O(\frac{n^3}{(1-\rho)^4})$ ⟹ Exact-Diffusion: $O(\frac{n^3}{(1-\rho)^2})$

Exact-Diffusion has a shorter transient stage with non-iid data

北京大学 PEKING UNIVERSITY

- We establish similar results in both strongly-convex and generally-convex scenarios [AY22, YAH23]

	D-SGD	Exact-Diffusion
Non-convex	$O(\frac{n^3}{(1-\rho)^4})$	$O(\frac{n^3}{(1-\rho)^2})$
Convex	$O(\frac{n^3}{(1-\rho)^4})$	$O(\frac{n^3}{(1-\rho)^2})$
Strongly convex	$O(\frac{n^3}{(1-\rho)^2})$	$O(\frac{n^3}{1-\rho})$

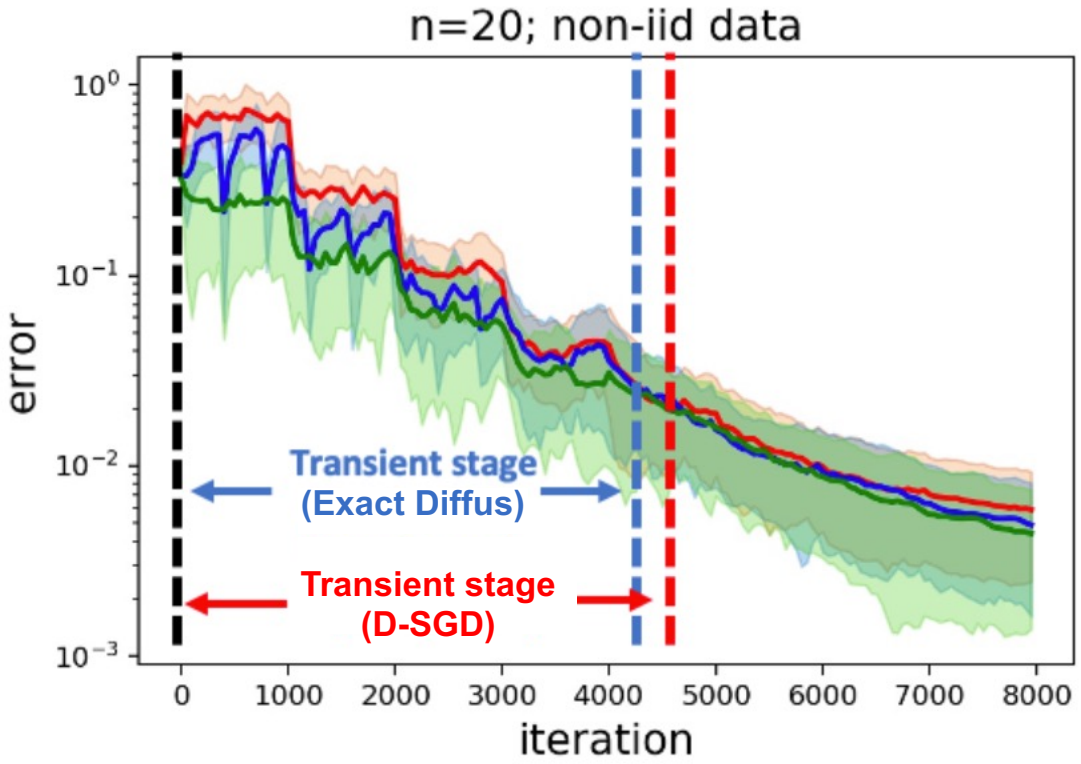
- Exact-Diffusion always improves the transient stage and the dependence on network topology

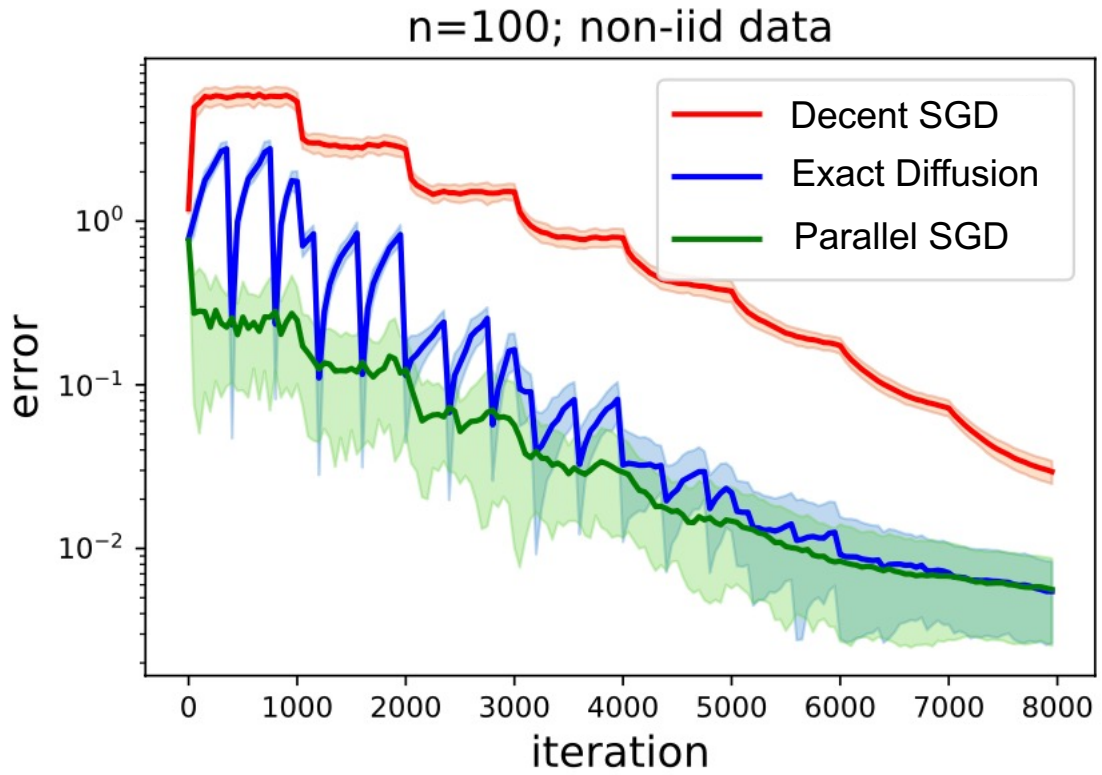
K. Yuan, S. Alghunaim, X. Huang, "Removing Data Heterogeneity Influence Enhances Network Topology Dependence of Decentralized SGD", JMLR 2023

S. Alghunaim and K. Yuan, "A Unified and Refined Convergence Analysis for Non-Convex Decentralized Learning", IEEE TSP 2022

Empirical studies

- Decentralized logistic regression with ring topology with **non-iid data**





- Exact-Diffusion has much shorter transient stage than D-SGD for sparse topology

Experiments in deep training (image classification)

ImageNet-1K dataset

1.3M training images

50K test images

1K classes

DNN model: ResNet-50 (25.5M parameters)

GPU: Up to 256 Tesla V100 GPUs

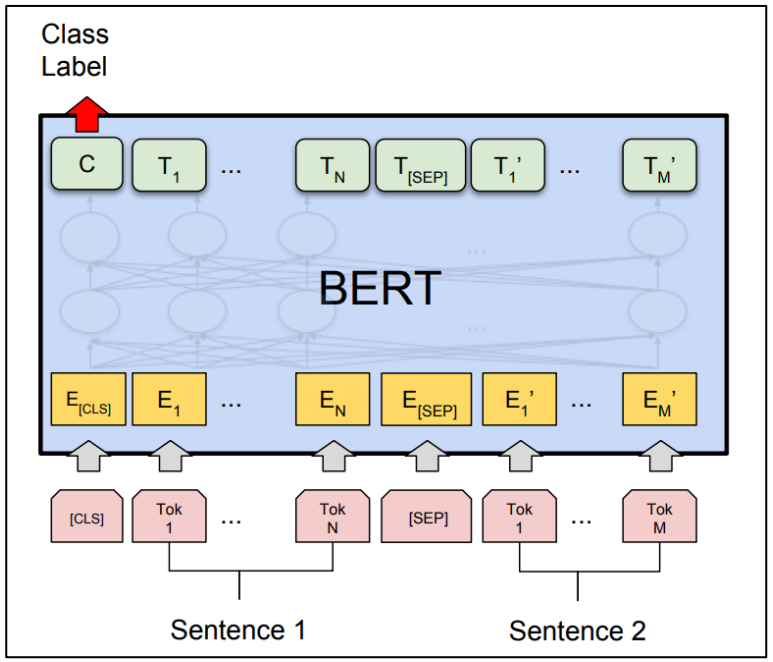
- **Wall-clock time** to finish 90 epochs of training; measures per-iter communication
- **Validation accuracy** after 90 epochs of training; measures convergence rate

DSGD achieves better linear speedup

nodes topology	4(4x8 GPUs) acc.	 time	8(8x8 GPUs) acc.	 time	16(16x8 GPUs) acc.	 time	32(32x8 GPUs) acc.	 time
P-SGD	76.32	11.6	76.47	6.3	76.46	3.7	76.25	2.2
Decentralized	76.34	11.1	76.52	5.7	76.47	2.8	76.27	1.5

By removing data heterogeneity and using proper topologies, Decentralized algorithms **reduces the training time without hurting the performance**

Experiments in deep learning (language modeling)



Model: BERT-Large (330M parameters)

Dataset: Wikipedia (2500M words) and BookCorpus (800M words)

Hardware: 64 GPUs

Table. Comparison in loss and training time [CYZ+21]

Method	Final Loss	Wall-clock Time (hrs)
P-SGD	1.75	59.02
D-SGD	1.77	30.4

北京大学
PEKING UNIVERSITY

PART 04

A Stochastic Unified Decentralized Algorithm

A stochastic unified decentralized algorithm

- Recall the distributed stochastic optimization problem

$$\min_{x\in\mathbb{R}^d} \quad f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x), \quad \text{where} \quad f_i(x)=\mathbb{E}_{\xi_i\sim D_i}F(x;\xi_i).$$

- We propose a stochastic unified decentralized algorithm (SUDA)

$$\begin{aligned}
\mathbf{z}^{k+1} &= \mathbf{C}\mathbf{x}^k - \alpha\nabla\mathbf{F}(\mathbf{x}^k,\boldsymbol{\xi}^k) - \mathbf{B}\mathbf{y}^k\\
\mathbf{y}^{k+1} &= \mathbf{y}^k + \mathbf{B}\mathbf{z}^{k+1}\\
\mathbf{x}^{k+1} &= \mathbf{A}\mathbf{z}^{k+1}
\end{aligned}$$

  - $\mathbf{A}$ and $\mathbf{C}$ are doubly stochastic matrices
  - $\mathbf{B}\mathbf{x}=0 \Longleftrightarrow x_1=x_2=\cdots=x_n$

A stochastic unified decentralized algorithm

- SUDA reduces to various well-known algorithms for difference choices of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$

- **EXTRA** [SLWY15]: when $\mathbf{A} = \mathbf{I}, \mathbf{B} = (\mathbf{I} - \mathbf{W})^{1/2}, \mathbf{C} = \mathbf{W},$ we recover EXTRA

$$\mathbf{x}^{k+2} = \mathbf{W}\left(2\mathbf{x}^{k+1} - \mathbf{x}^k\right) - \gamma\left(\nabla \mathbf{F}(\mathbf{x}^{k+1}; \boldsymbol{\xi}^{k+1}) - \nabla \mathbf{F}(\mathbf{x}^k; \boldsymbol{\xi}^k)\right)$$

- **Exact-Diffusion/D2** [YYZS19, TLYZ+18] : when $\mathbf{A} = \mathbf{W}, \mathbf{B} = (\mathbf{I} - \mathbf{W})^{1/2}, \mathbf{C} = \mathbf{I},$ we recover Exact-Diffusion

$$\mathbf{x}^{k+2} = \mathbf{W}\left(2\mathbf{x}^{k+1} - \mathbf{x}^k - \gamma\left(\nabla \mathbf{F}(\mathbf{x}^{k+1}; \boldsymbol{\xi}^{k+1}) - \nabla \mathbf{F}(\mathbf{x}^k; \boldsymbol{\xi}^k)\right)\right)$$

A stochastic unified decentralized algorithm

- SUDA reduces to various well-known algorithms for difference choices of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$

- **ATC Gradient Tracking** [XZSX15]: when $\mathbf{A} = \mathbf{W}^2$, $\mathbf{B} = (\mathbf{I} - \mathbf{W})$, and $\mathbf{C} = \mathbf{I}$, we recover

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{W}\left(\mathbf{x}^k - \gamma \mathbf{G}^k\right)\\ \mathbf{G}^{k+1} &= \mathbf{W}\left(\mathbf{G}^k + \nabla \mathbf{F}(\mathbf{x}^{k+1}; \boldsymbol{\xi}^{k+1}) - \nabla \mathbf{F}(\mathbf{x}^k; \boldsymbol{\xi}^k)\right)\end{aligned}$$

- **AWC Gradient Tracking** [NOS17]: when $\mathbf{A} = \mathbf{I}, \mathbf{B} = \mathbf{I} - \mathbf{W}, \mathbf{C} = \mathbf{W}^2$, we recover

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{W}\mathbf{x}^k - \gamma \mathbf{G}^k\\ \mathbf{G}^{k+1} &= \mathbf{W}\mathbf{G}^k + \nabla \mathbf{F}(\mathbf{x}^{k+1}; \boldsymbol{\xi}^{k+1}) - \nabla \mathbf{F}(\mathbf{x}^k; \boldsymbol{\xi}^k)\end{aligned}$$

A stochastic unified decentralized algorithm

北京大学 PEKING UNIVERSITY

A unified algorithm is important because

- One analysis **fits all** existing popular decentralized algorithms
- Any new trick on SUDA **applies to all** existing popular decentralized algorithms
  - Proximal extensions to non-smooth problem [AYS19; ARYS 20]
  - Accelerated extensions to non-smooth problem

Convergence analysis

When the step size satisfies α is sufficiently small, then, the iterates $\{\mathbf{x}^k\}$ of SUDA with $\mathbf{x}^0 = \mathbf{1} \otimes x^0$ ($x^0 \in \mathbb{R}^m$) satisfy

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\,\|\overline{\nabla \mathbf{f}}(\mathbf{x}^k)\|^2 \le \frac{8(f(x^0)-f^\star)}{\alpha K} + \frac{4\alpha L\sigma^2}{N} + \frac{12\alpha^2 L^2 v_1^2 v_2^2 \zeta_0^2}{\underline{\lambda_b}^2(1-\gamma)K} + \frac{16\alpha^2 L^2 v_1^2 v_2^2 \lambda_a^2 \sigma^2}{1-\gamma} + \frac{16\alpha^4 L^4 v_1^2 v_2^2 \lambda_a^2 \sigma^2}{\underline{\lambda_b}^2(1-\gamma)^2 N},$$

where $v_1 \triangleq \|\hat{\mathbf{V}}\|$, $v_2 \triangleq \|\hat{\mathbf{V}}^{-1}\|$ and

$$\gamma \triangleq \|\mathbf{\Gamma}\| < 1, \quad \underline{\lambda_b} \triangleq \frac{1}{\|\mathbf{\Lambda}_b^{-1}\|}, \quad \lambda_a \triangleq \|\mathbf{\Lambda}_a\|,$$

$$\zeta_0^2 \triangleq \tfrac{1}{N}\|(\mathbf{A} - \tfrac{1}{N}\mathbf{1}^T\mathbf{1} \otimes I_m)\big(\nabla \mathbf{f}(\mathbf{x}^0) - \mathbf{1} \otimes \nabla f(x^0)\big)\|^2$$

Refined rate in special scenarios

METHOD	WORK	CONVERGENCE RATE
ED/D^2	[prior]	$O\left(\frac{1}{\sqrt{NK}}+\frac{N\lambda^2}{(1-\lambda)^3K}+\frac{N\varsigma_0^2}{(1-\lambda)^2K^2}\right)$
	SUDA	$O\left(\frac{1}{\sqrt{NK}}+\frac{N\lambda^2}{(1-\lambda)K}+\frac{N\lambda^2\varsigma_0^2}{(1-\lambda)^2K^2}\right)$
ATC-GT	[prior]	$O\left(\frac{1}{\sqrt{NK}}+\frac{N\lambda^2}{(1-\lambda)^3K}+\frac{\lambda^4\sum_{i=1}^N\|\nabla f_i(0)\|^2}{(1-\lambda)^3K^2}\right)$
	SUDA	$O\left(\frac{1}{\sqrt{NK}}+\frac{N\lambda^4}{(1-\lambda)K}+\frac{N\lambda^4}{(1-\lambda)^4K^2}+\frac{N\lambda^4\varsigma_0^2}{(1-\lambda)^3K^2}\right)$

Refined transient stage in special scenarios

北京大学 PEKING UNIVERSITY

METHOD	WORK	TRANSIENT TIME
DSGD	[prior]	$\mathcal{O}\left(\frac{N^3}{(1-\lambda)^4}\right)$
ED/D^2	[prior]	$\mathcal{O}\left(\frac{N^3}{(1-\lambda)^6}\right)$
	SUDA	$\mathcal{O}\left(\frac{N^3}{(1-\lambda)^2}\right)$
ATC-GT	[prior]	$\mathcal{O}\left(\frac{N^3}{(1-\lambda)^6}\right)$
	SUDA	$\mathcal{O}\left(\max\left\{\frac{N^3}{(1-\lambda)^2}, \frac{N}{(1-\lambda)^{8/3}}\right\}\right)$

Final summary

北京大学 PEKING UNIVERSITY

- Decentralized algorithms save remarkable communication compared to centralized ones
- Data heterogeneity results in long transient stage in decentralized SGD
- We propose Exact-Diffusion to remove data heterogeneity, and improves the transient stage
- We introduced a stochastic unified decentralized algorithm that involves many popular algorithms

北京大学
PEKING UNIVERSITY

Thank you!

Kun Yuan homepage: https://kunyuan827.github.io/

We have openings for PostDocs

北京大学 PEKING UNIVERSITY

PART 05

BlueFog: An open-source and high-performance python library

https://github.com/Bluefog-Lib/bluefog

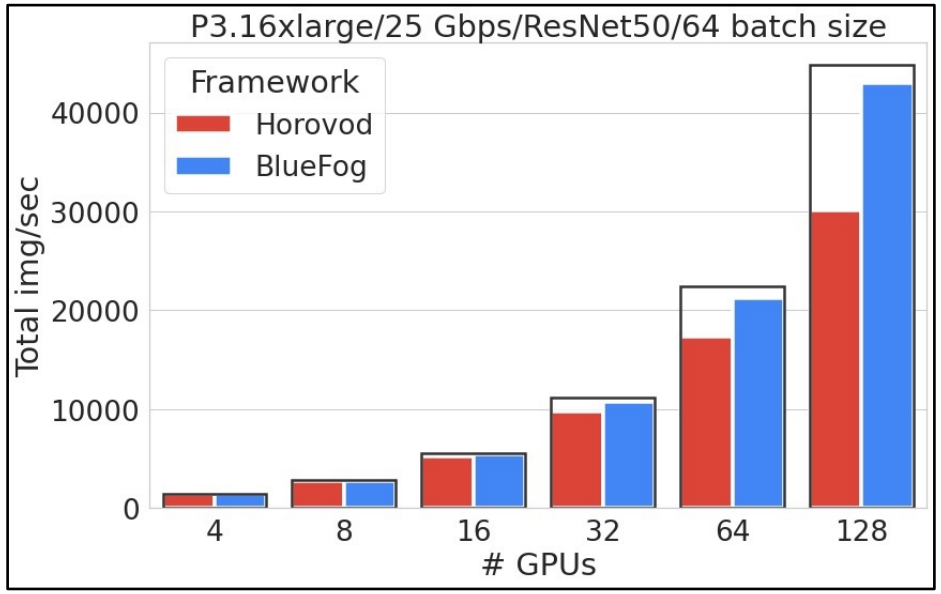
BlueFog

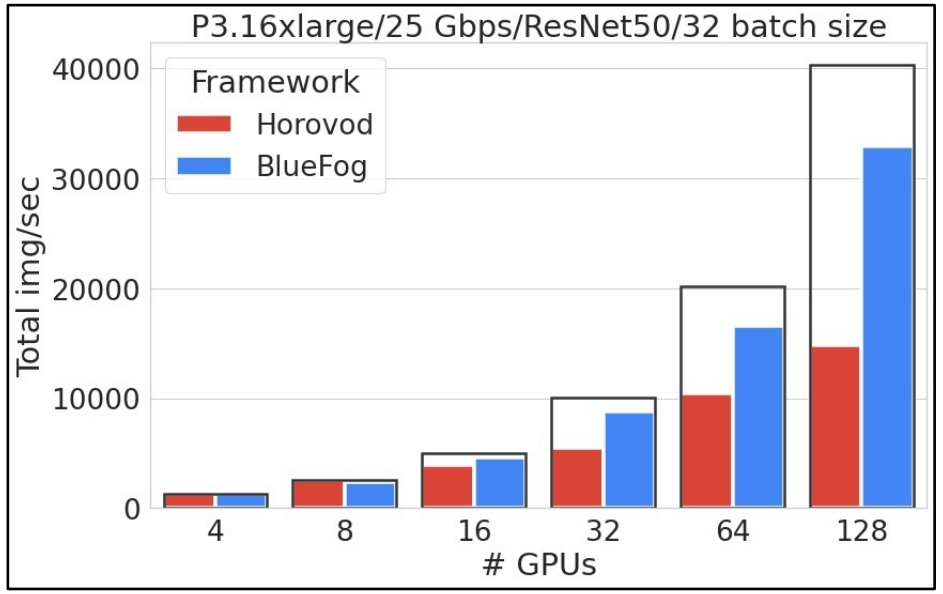
北京大学 PEKING UNIVERSITY

- An open-source library to support decentralized communication in optimization and deep learning
- High-performance
- Easy-to-use

High-performance

- BlueFog has larger throughput than Horovod (the SOTA DL system implementing PSGD) [YYH+21]





- All our research progresses are involved in BlueFog

[YYH+21] B. Ying, K. Yuan, H. Hu, Y. Chen, and W. Yin, ``BlueFog: Make Decentralized Algorithms Practical for Optimization and machine learning", arXiv:2111.04287 [GitHub site: github.com/Bluefog-Lib/bluefog]

Easy-to-use

北京大学 PEKING UNIVERSITY

- Writing codes for decentralized methods is as easy as writing equations

Decentralized least-square algorithms

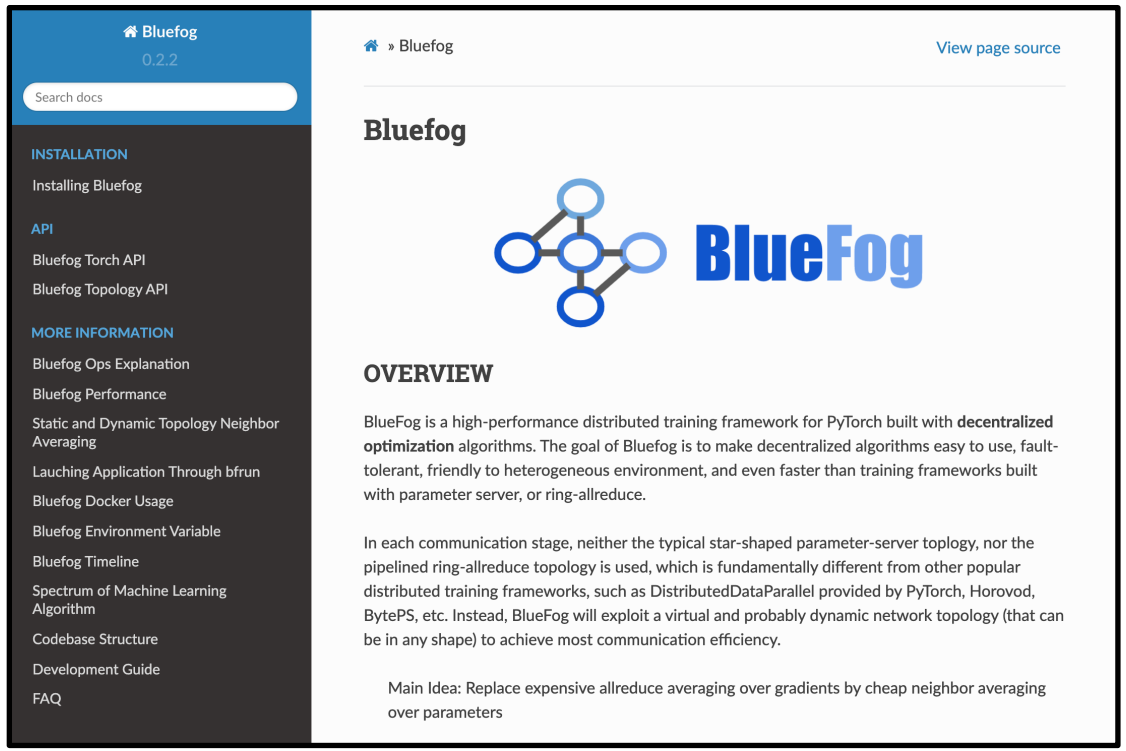
$$y_i^{(k)} = x_i^{(k)} - \gamma A_i^T (A_i x_i^{(k)} - b_i)$$
$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i} w_{ij} y_j^{(k)}$$

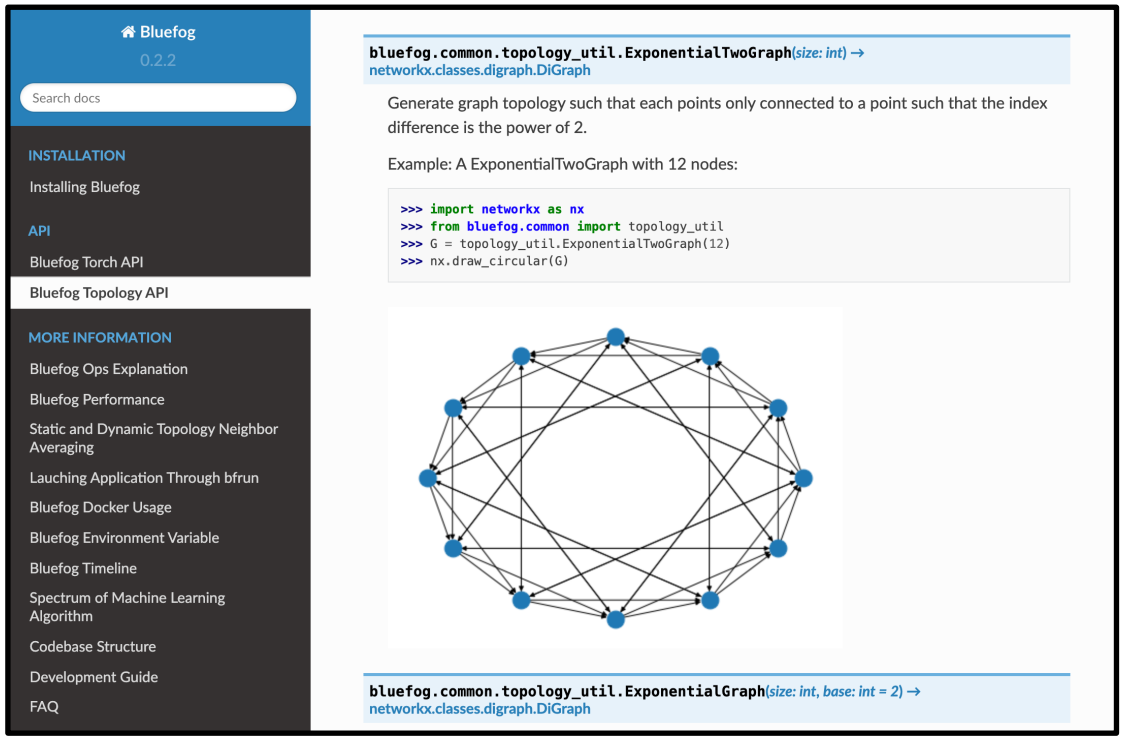
```
import bluefog.torch as bf
bf.init()  # Initialize the BlueFog

# Set topology as static exponential graph.
G = bf.ExponentialTwoGraph(bf.size())
bf.set_topology(G)

# DGD implementation
for ite in range(maxite):
    grad_local = A.t().mm(A.mm(x) - b)   # compute local grad
    y = x - gamma * grad_local           # local update
    x = bf.neighbor_allreduce(y)         # partial averaging
```

Easy-to-use

北京大学 PEKING UNIVERSITY

Abundant documents





Easy-to-use

北京大学 PEKING UNIVERSITY

Detailed tutorials

Contents

2.1.3 Initialize BlueFog and test it

All contents in this section are displayed in Jupyter notebook, and all experimental examples are written with BlueFog and iParallel. Readers not familiar with how to run BlueFog in ipython notebook environment is encouraged to read Sec. [HelloWorld section] first. In the following codes, we will initialize BlueFog and test whether it works normally.

The output of `rc.ids` should be a list from 0 to the number of processes minus one. The number of processes is the one you set in the `ibfrun start -np {X}`.

In [1]:
```
import ipyparallel as ipp

rc = ipp.Client(profile="bluefog")
rc.ids
```

Let each agent import necessary modules and then initialize BlueFog. You should be able to see the printed information like:

> [stdout:0] Hello, I am 1 among 4 processes
>
> ...

In [2]:
```
%%px
import numpy as np
import bluefog.torch as bf
import torch
from bluefog.common import topology_util
import networkx as nx

bf.init()
print(f"Hello, I am {bf.rank()} among {bf.size()} processes")
```

Push seed to each agent so that the simulation can be reproduced.

In [3]:
```
dview = rc[:]  # A DirectView of all engines
dview.block = True

# Push the data into all workers
#   `dview.push({'seed': 2021}, block=True)`
# Or equivalently
dview["seed"] = 2021
```

After running the following code, you should be able to see the printed information like

> [stdout:0] I received seed as value: 2021