A Double Residual Compression Algorithm for Efficient Distributed Learning

Xiaorui Liu

Joint work with Yao Li, Jiliang Tang, Ming Yan

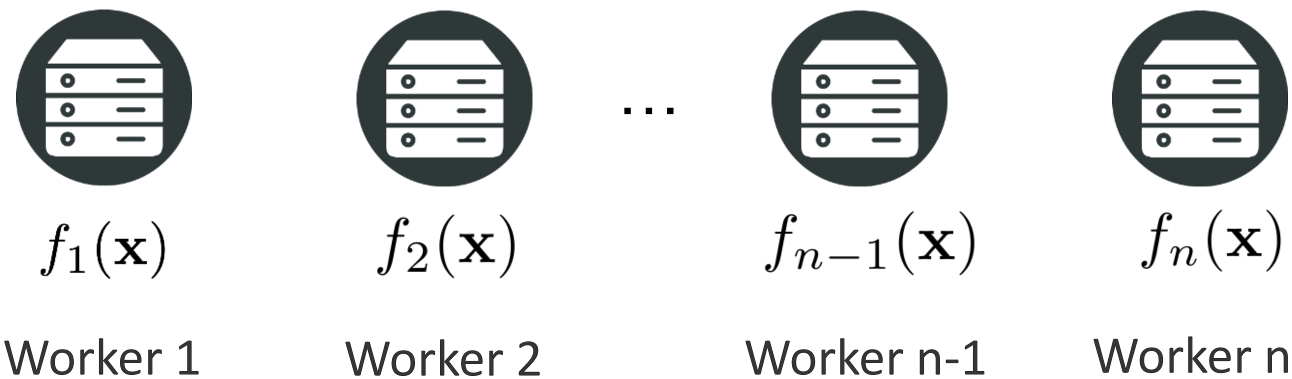
Michigan State University

June 15th, AISTATS 2020

Distributed Learning

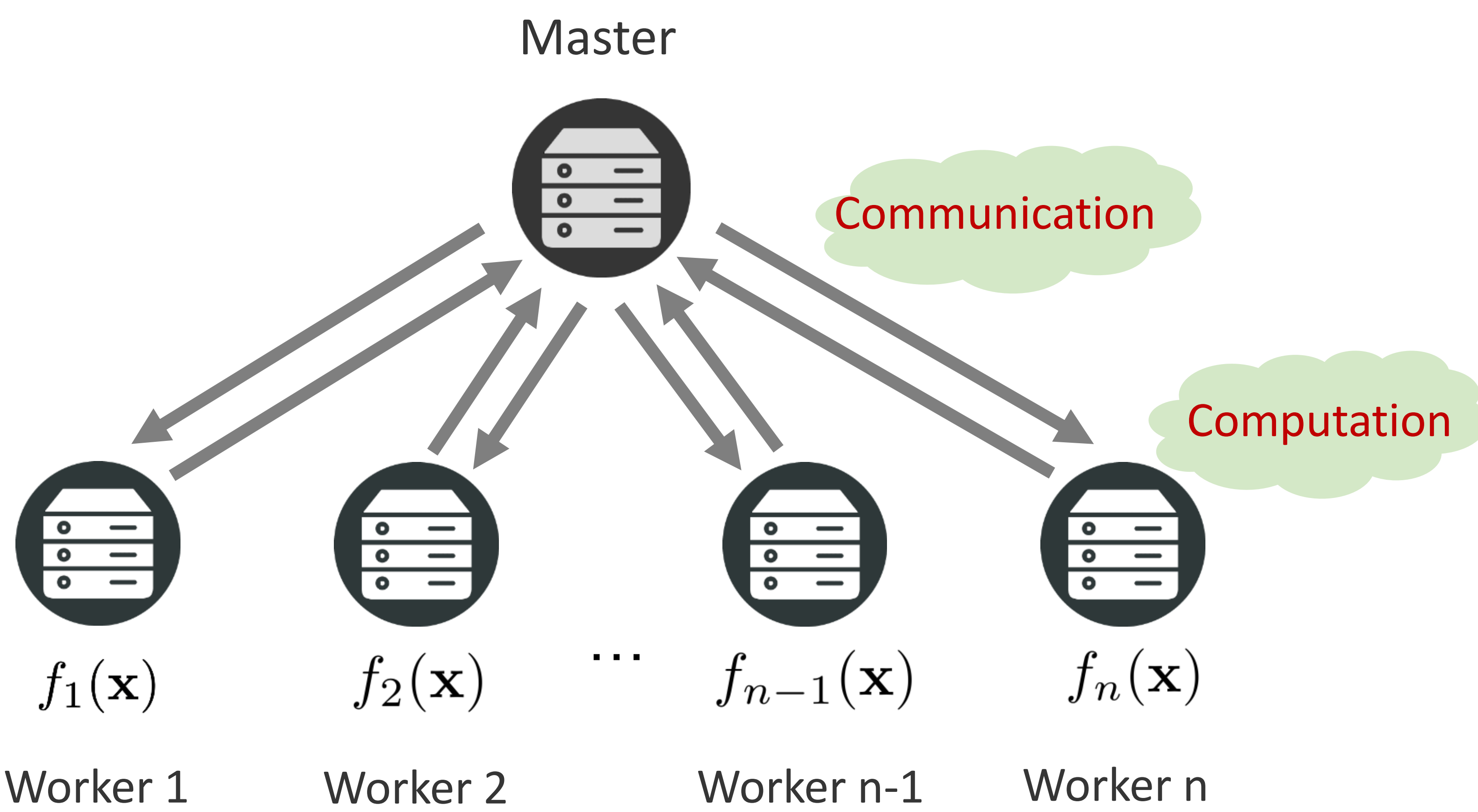
Problem

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}}\, f(\mathbf{x}) + R(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\mathbb{E}_{\xi\sim\mathcal{D}_i}[\ell(\mathbf{x},\xi)]}_{:=f_i(\mathbf{x})} + R(\mathbf{x})$$



Data is partitioned at different worker machines

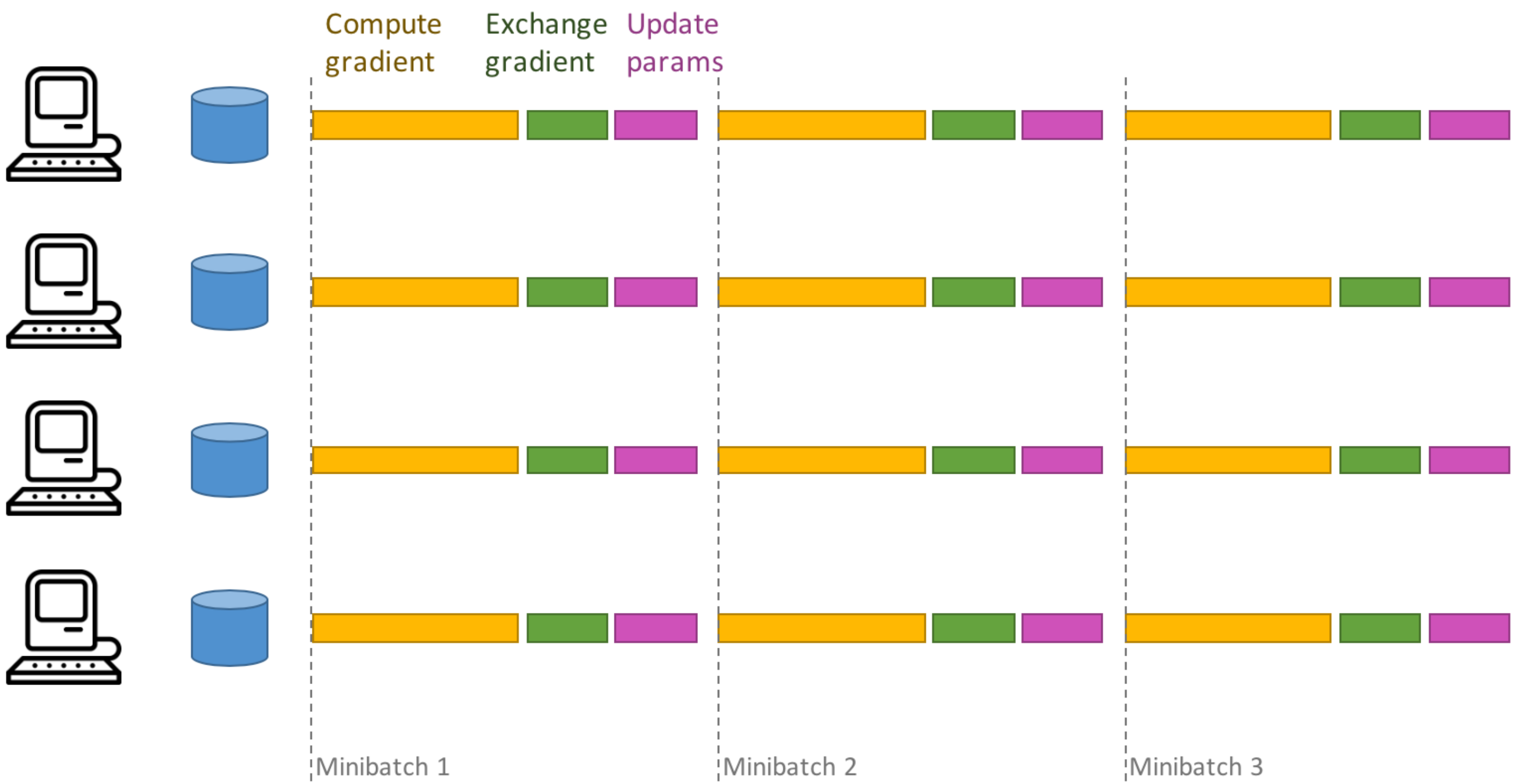
Parallel SGD



- Gradient reduce & model (or averaged gradient) broadcasting
- Widely supported and used in PyTorch/TensorFlow/MXNET …

Parallel SGD

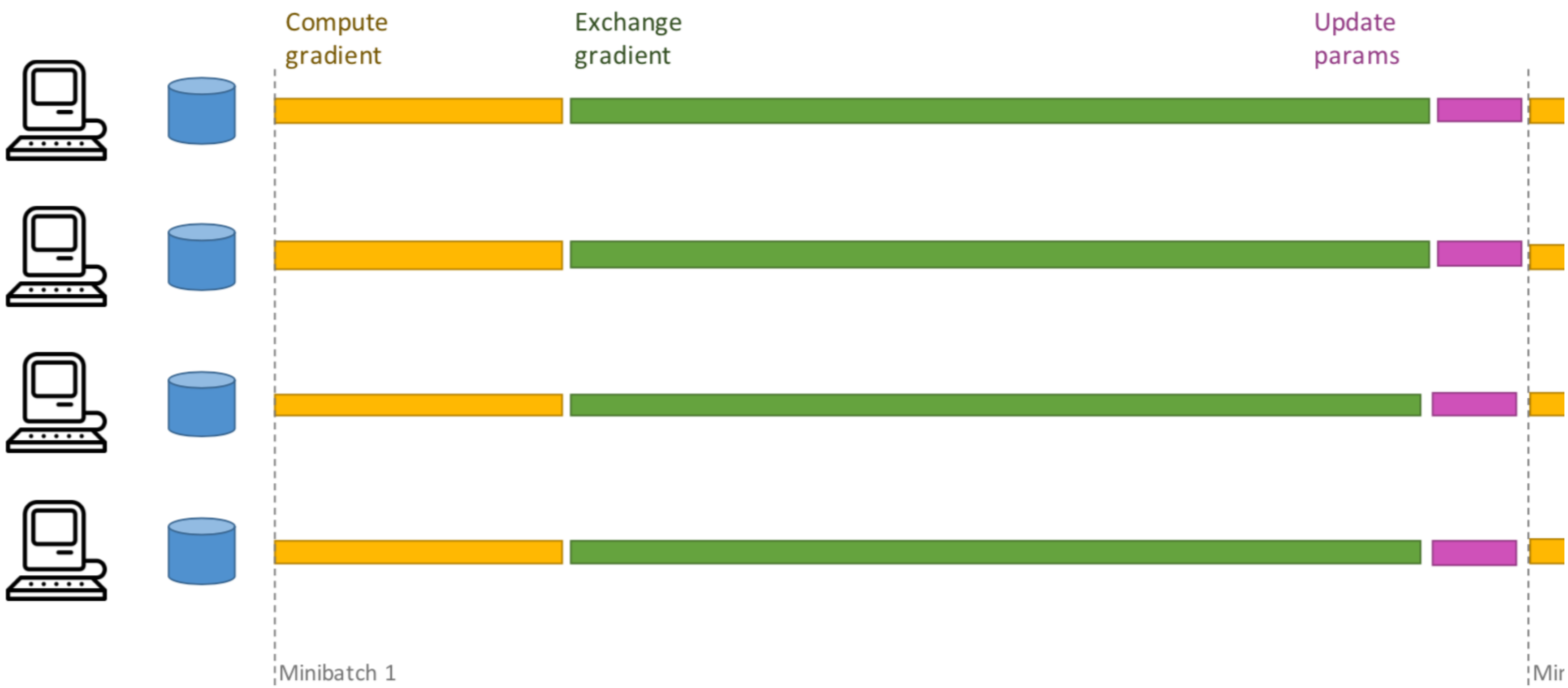
Hopefully



D. Alistarh's Tutorial at PODC 2018

Parallel SGD

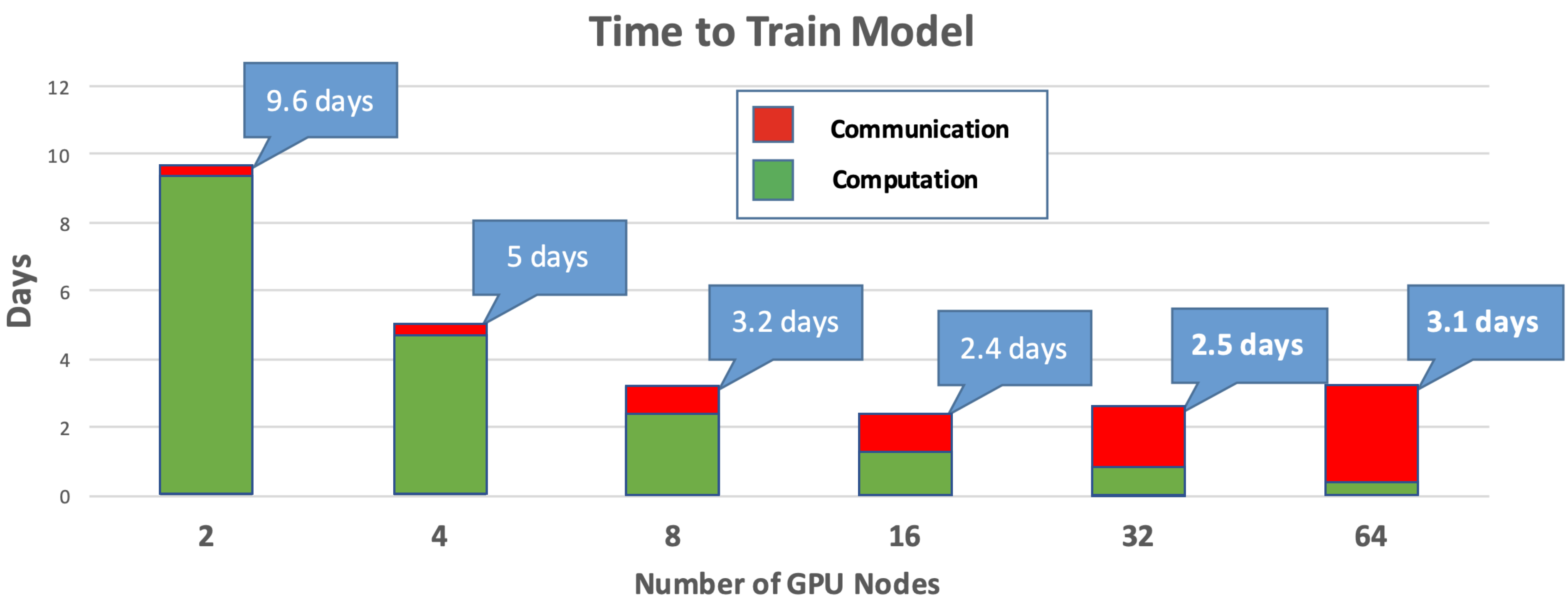
Big model



D. Alistarh's Tutorial at PODC 2018

Parallel SGD

Big network

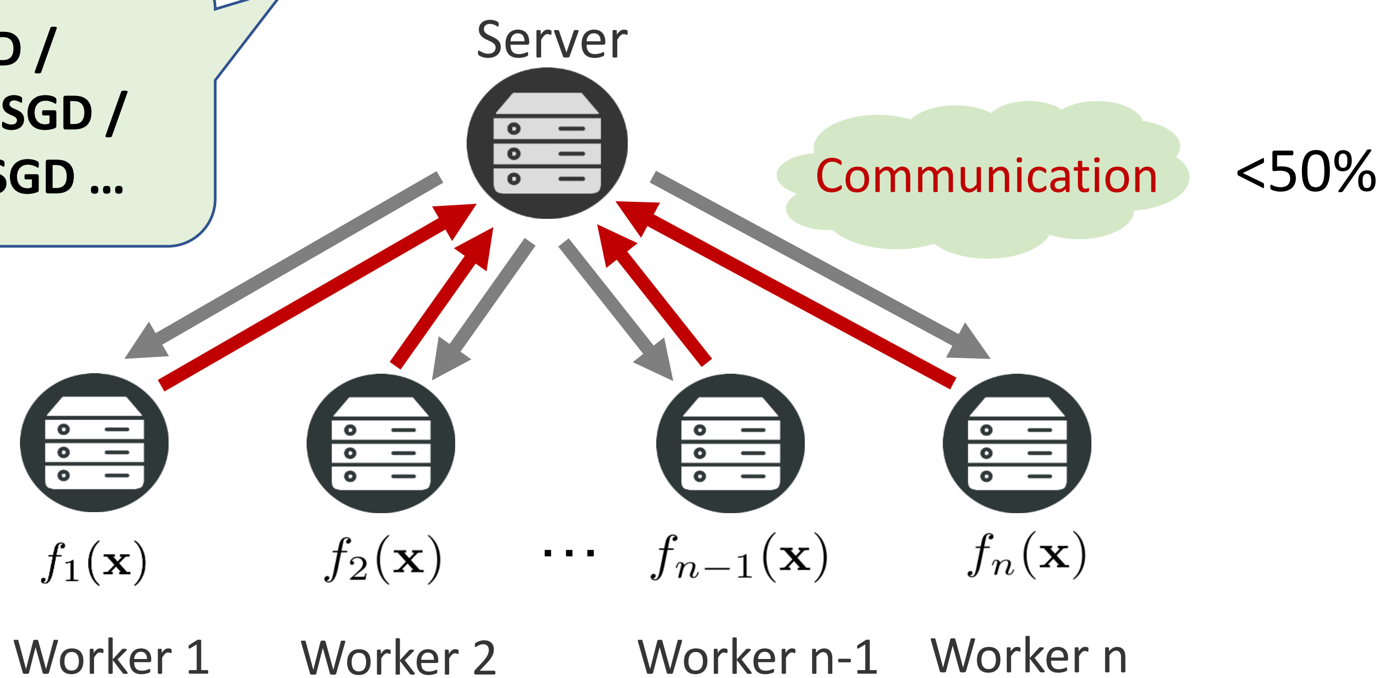


D. Alistarh's Tutorial at PODC 2018

Parallel SGD

Gradient compression

1Bit SGD / QSGD / Terngrad / ECQ-SGD / DIANA / MEM-SGD ...

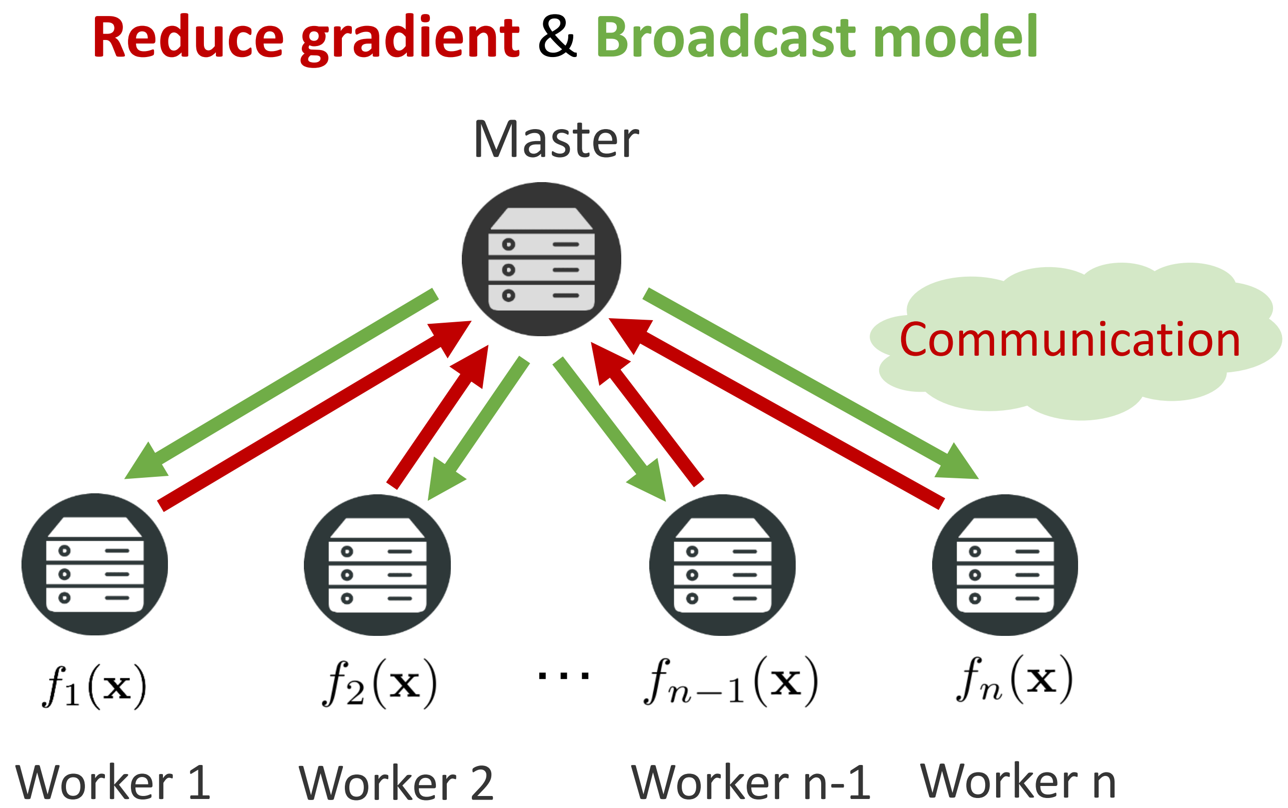


Most algorithms either

(1) directly broadcast the full-precision model or

(2) allgather the compressed gradients

DOuble REsidual compression (DORE)



- **Worker side**: gradient residual compression + running average
- **Server side**: model residual compression + error compensation

Algorithm DORE

DORE with R(X)

Input: Stepsize $\alpha, \beta, \gamma, \eta$, initialize $\mathbf{h}^0 = \mathbf{h}_i^0 = \mathbf{0}^d$, $\hat{\mathbf{x}}_i^0 = \hat{\mathbf{x}}^0,\ \forall i \in \{1, \ldots, n\}$.

for $k = 1, 2, \cdots, K-1$ **do**

For each worker $\{i = 1, 2, \cdots, n\}$:

Sample $\mathbf{g}_i^k$ such that $\mathbb{E}[\mathbf{g}_i^k | \hat{\mathbf{x}}_i^k] = \nabla f_i(\hat{\mathbf{x}}_i^k))$

Gradient residual: $\Delta_i^k = \mathbf{g}_i^k - \mathbf{h}_i^k$

Compression: $\hat{\Delta}_i^k = Q(\Delta_i^k)$

$\mathbf{h}_i^{k+1} = \mathbf{h}_i^k + \alpha \hat{\Delta}_i^k$

$\{\ \hat{\mathbf{g}}_i^k = \mathbf{h}_i^k + \hat{\Delta}_i^k\ \}$

Sent $\hat{\Delta}_i^k$ to the master

Receive $\hat{\mathbf{q}}^k$ from the master

$\hat{\mathbf{x}}_i^{k+1} = \hat{\mathbf{x}}_i^k + \beta \hat{\mathbf{q}}^k$

For the master:

Receive $\{\hat{\Delta}_i^k\}$ from workers

$\hat{\Delta}^k = 1/n \sum_i^n \hat{\Delta}_i^k$

$\hat{\mathbf{g}}^k = \mathbf{h}^k + \hat{\Delta}^k \{= \frac{1}{n} \sum_i^n \hat{\mathbf{g}}_i^k\}$

$\mathbf{x}^{k+1} = \mathbf{prox}_{\gamma R}(\hat{\mathbf{x}}^k - \gamma \hat{\mathbf{g}}^k)$

$\mathbf{h}^{k+1} = \mathbf{h}^k + \alpha \hat{\Delta}^k$

Model residual: $\mathbf{q}^k = \mathbf{x}^{k+1} - \hat{\mathbf{x}}^k + \eta \mathbf{e}^k$

Compression: $\hat{\mathbf{q}}^k = Q(\mathbf{q}^k)$

$\mathbf{e}^{k+1} = \mathbf{q}^k - \hat{\mathbf{q}}^k$

$\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \beta \hat{\mathbf{q}}^k$

Broadcast $\hat{\mathbf{q}}^k$ to workers

end for

Output: $\hat{\mathbf{x}}^K$ or any $\hat{\mathbf{x}}_i^K$

Algorithm DORE

➢ **Worker side**: gradient residual compression + running average

Intuition 1: issue of simple gradient compression

$$\begin{aligned}\mathbf{x} =& \ \mathbf{x}^* - \frac{\gamma}{n}\sum_{i=1}^{n} Q(\nabla f_i(\mathbf{x}^*)) \\ =& \ \mathbf{x}^* - \frac{\gamma}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x}^*) + \frac{\gamma}{n}\sum_{i=1}^{n}(\nabla f_i(\mathbf{x}^*) - Q(\nabla f_i(\mathbf{x}^*))).\end{aligned}$$

- The convergence requires either
  (1) diminishing stepsize γ or (2) diminishing compression error

- Error compensation on the worker side doesn't solve this issue

Algorithm DORE

➢ **Worker side**: gradient residual compression + running average

Intuition 2: Gradient for smooth function changes smoothly

- Keep a state **h** to track the local gradient
- Residual between current gradient and **h** vanishes
- Recover the estimated gradient on server side

$$\mathbf{x} = \mathbf{x}^* - \frac{\gamma}{n}\sum_{i=1}^{n}\left(\mathbf{h}_i + Q(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i)\right)$$

$$= \mathbf{x}^* - \underbrace{\frac{\gamma}{n}\sum_{i=1}^{n}\nabla f_i(\mathbf{x}^*)}_{=\mathbf{0}} + \frac{\gamma}{n}\sum_{i=1}^{n}\left(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i - Q(\nabla f_i(\mathbf{x}^*) - \mathbf{h}_i)\right)$$

C-contraction compressor: $\mathbb{E}\|\mathbf{v} - Q(\mathbf{v})\|^2 \leq C\|\mathbf{v}\|^2, \quad \forall \mathbf{v} \in \mathbb{R}^d$

Mishchenko et al. 19'

Algorithm DORE

➢ **Worker side**: gradient residual compression + running average

Intuition 3: Achieve vanishing residual by running average

$$\begin{aligned}\mathbf{h}_i^{k+1} &= (1-\alpha)\mathbf{h}_i^k + \alpha\left(\mathbf{h}_i^k + Q(\nabla f_i(\mathbf{x}^k) - \mathbf{h}_i^k)\right)\\ &= \mathbf{h}_i^k + \alpha Q(\nabla f_i(\mathbf{x}^k) - \mathbf{h}_i^k)\end{aligned}$$

With unbiasedness compression $\mathbb{E}Q(\mathbf{v}) = \mathbf{v}$, have

$$\mathbb{E}_Q \mathbf{h}_i^{k+1} = (1-\alpha)\mathbf{h}_i^k + \alpha \nabla f_i(\mathbf{x}^k)$$

such that $\mathbf{h}_i^k \to f_i(\mathbf{x}^*)$ once $\mathbf{x}^k \to \mathbf{x}^*$

Mishchenko et al. 19'

Algorithm DORE

➢ **Server side**: model residual compression + error compensation

Intuition 1: model changes slowly when approaching optima

- Model residual compression will only incur diminishing error

Intuition 2: compensate the compression error to next iteration

- Consider the error as delay and maintain it for faster convergence

Remark:

To prove the convergence, most works using error compensation require the bounded gradient assumption, but DORE doesn't.

Convergence analysis

Assumption on the compression

Assumption

The stochastic compression operator $Q : \mathbb{R}^d \to \mathbb{R}^d$ *is unbiased, i.e.,* $\mathbb{E}Q(\mathbf{x}) = \mathbf{x}$ *and satisfies*

$$\mathbb{E}\|Q(\mathbf{x}) - \mathbf{x}\|^2 \leq C\|\mathbf{x}\|^2,$$

for a nonnegative constant C *that is independent of* $\mathbf{x}$.

- Random Quantization
- Random Sparsification
- P-norm Quantization
- …

Convergence analysis

Assumption

Each worker node samples an unbiased estimator of the gradient stochastically with bounded variance, i.e., for $i = 1, 2, \cdots, n$ *and* $\forall \mathbf{x} \in \mathbb{R}^d$,

$$\mathbb{E}[\mathbf{g}_i | \mathbf{x}] = \nabla f_i(\mathbf{x}), \quad \mathbb{E}\|\mathbf{g}_i - \nabla f_i(\mathbf{x})\|^2 \leq \sigma_i^2,$$

where $\mathbf{g}_i$ *is the estimator of* ∇f_i *at* $\mathbf{x}$*. In addition, we define* $\sigma^2 = \frac{1}{n}\sum_{i=1}^n \sigma_i^2$.

Assumption

Each f_i *is* L*-Lipschitz differentiable, i.e., for* $i = 1, 2, \cdots, n$ *and* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$f_i(\mathbf{x}) \leq f_i(\mathbf{y}) + \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|^2.$$

Convergence analysis

Theorem

Choose parameters such that

$$\beta = \frac{1}{C+1}, \quad \alpha = \frac{1}{2(C+1)},$$
$$\gamma = \eta = \frac{1}{12L(1+2C/n)(1+\sqrt{K/n})}$$

Then we have

$$\frac{1}{K}\sum_{k=1}^{K} \mathbb{E}\|\nabla f(\hat{\mathbf{x}}^k)\|^2 \lesssim \frac{1}{K} + \frac{1}{\sqrt{Kn}}.$$

- Sublinear convergence to stationary points for non-convex cases
- Linear speedup w.r.t. number of workers

Convergence analysis

Assumption

Each f_i is μ-strongly convex ($\mu \geq 0$), i.e., for $i = 1, 2, \cdots, n$ and $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$f_i(\mathbf{x}) \geq f_i(\mathbf{y}) + \langle \nabla f_i(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \tfrac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2.$$

Theorem

Choose parameters such that

$$\begin{aligned}
0 < \beta &\leq \tfrac{1}{C+1} \\
\tfrac{1-\sqrt{1-\delta}}{2(C+1)} \leq \alpha &\leq \tfrac{1+\sqrt{1-\delta}}{2(C+1)}, \\
\eta < \min &\left(\tfrac{\sqrt{C^2+4(1-(C+1)\beta)}-C}{2C}, \ \tfrac{4\mu L}{(\mu+L)^2\left(1+\frac{4C(C+1)}{n\delta}\alpha\right)-4\mu L} \right), \\
\tfrac{\eta(\mu+L)}{2(1+\eta)\mu L} \leq \gamma &\leq \tfrac{2}{\left(1+\frac{4C(C+1)}{n\delta}\alpha\right)(\mu+L)}.
\end{aligned}$$

Then we have

$$\mathbf{V}^{k+1} \leq \rho^k \mathbf{V}^1 + \frac{(1+\eta)\left(1+n\frac{4C(C+1)}{n\delta}\alpha\right)}{n(1-\rho)}\beta\gamma^2\sigma^2,$$

for some $\rho < 1$, and $\mathbf{V}^k$ measures the convergence of $\mathbf{q}^k \to \mathbf{0}$, $\hat{\mathbf{x}}^k \to \mathbf{x}^*$, and $\mathbf{h}_i^k \to \nabla f_i(\mathbf{x}^*)$.

Convergence analysis

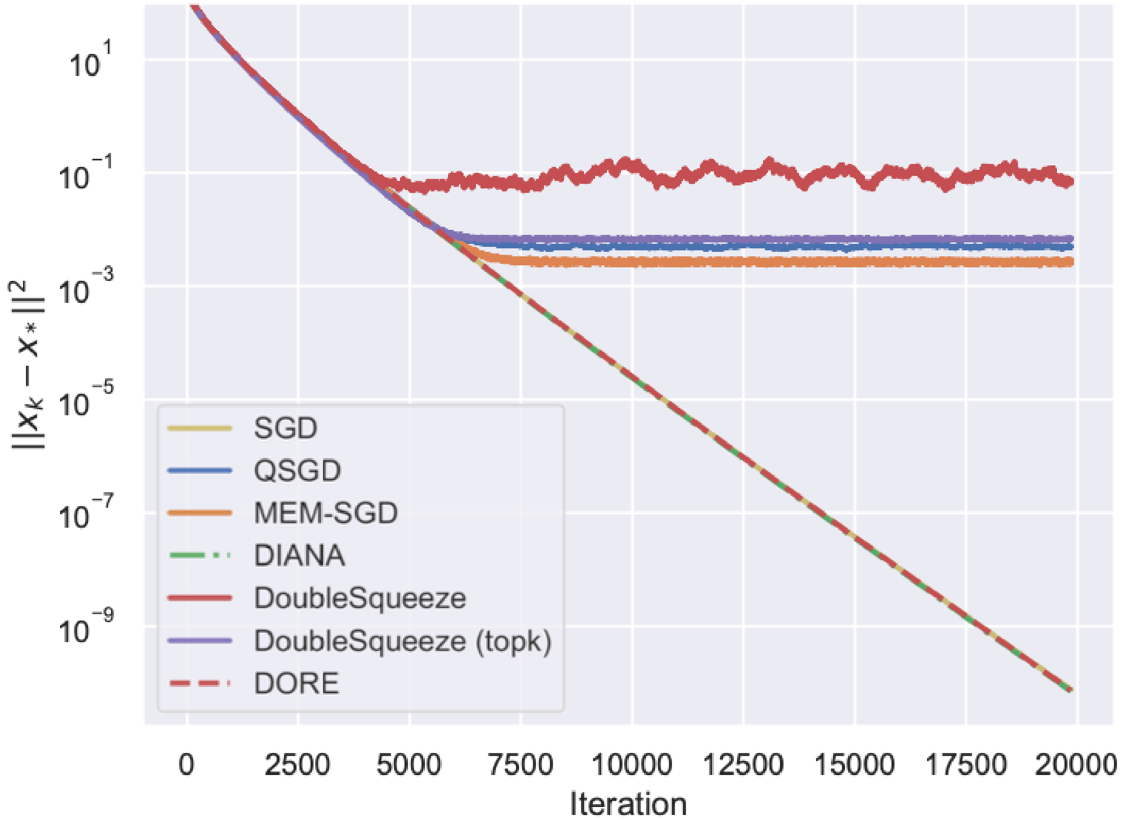
Theoretical comparison with related works

Algorithm	Compression	Compress. Model	Linear	Nonconvex Rate
SGD	No	No	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$
QSGD	Grad	2-norm	N/A	$\frac{1}{K} + B$
MEM-SGD	Grad	k-contraction	N/A	N/A
DIANA	Grad	p-norm	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$
DoubleSqueeze	Grad+Model	Bdd Variance	N/A	$\frac{1}{\sqrt{Kn}} + \frac{1}{K^{2/3}} + \frac{1}{K}$
DORE	Grad+Model	Assum. 1	✓	$\frac{1}{\sqrt{Kn}} + \frac{1}{K}$

Most algorithms, except DIANA and DORE, requires bounded gradient assumption and incur extra error.

Numerical experiment

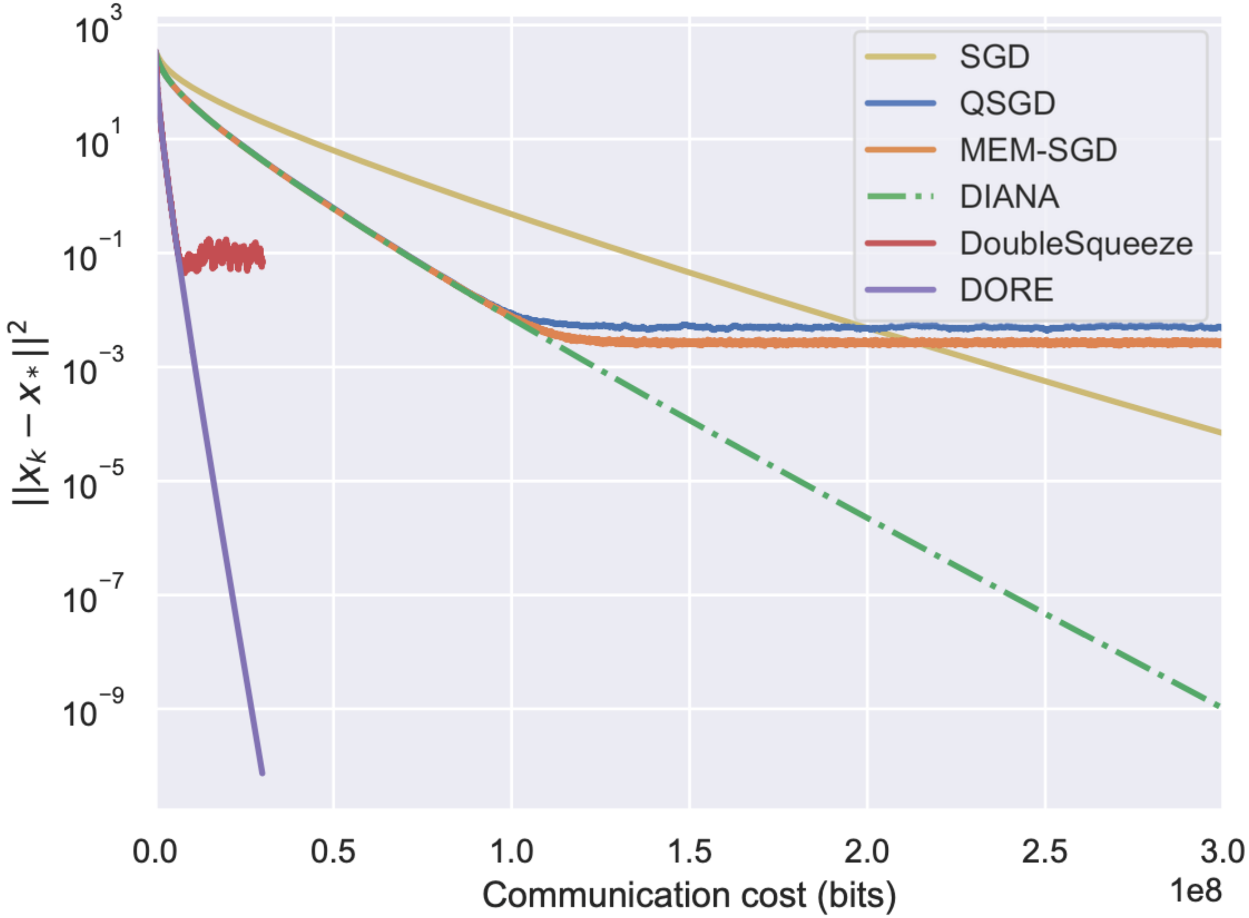
Regularized Least square problem



$$\mathbf{V}^{k+1} \leq \rho^k \mathbf{V}^1 + \frac{(1+\eta)\left(1+n\frac{4C(C+1)}{n\delta}\alpha\right)}{n(1-\rho)}\beta\gamma^2\sigma^2$$ full-gradient where $\sigma = 0$

Numerical experiment

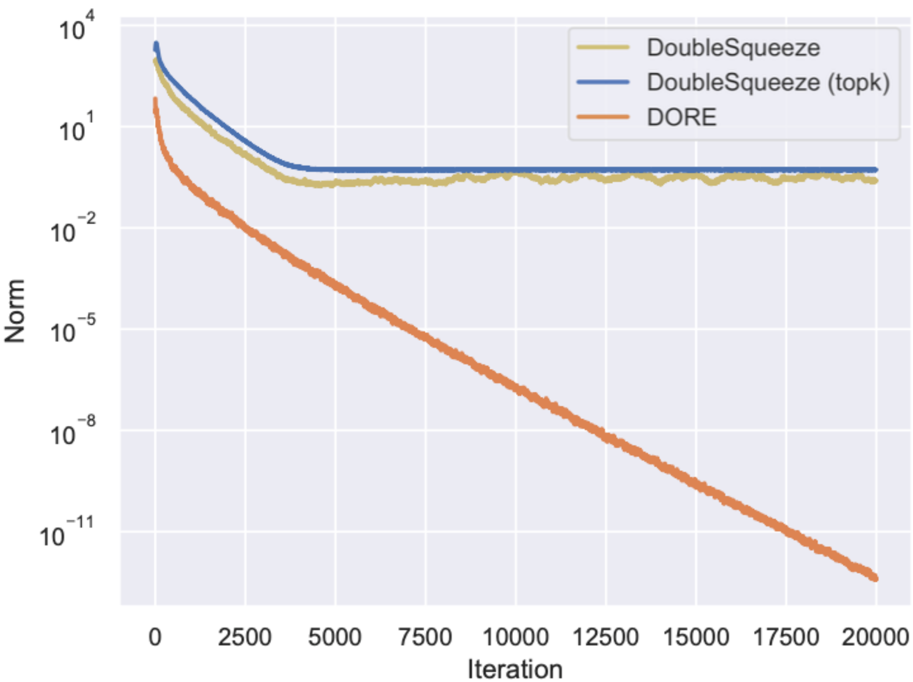
Regularized Least square problem



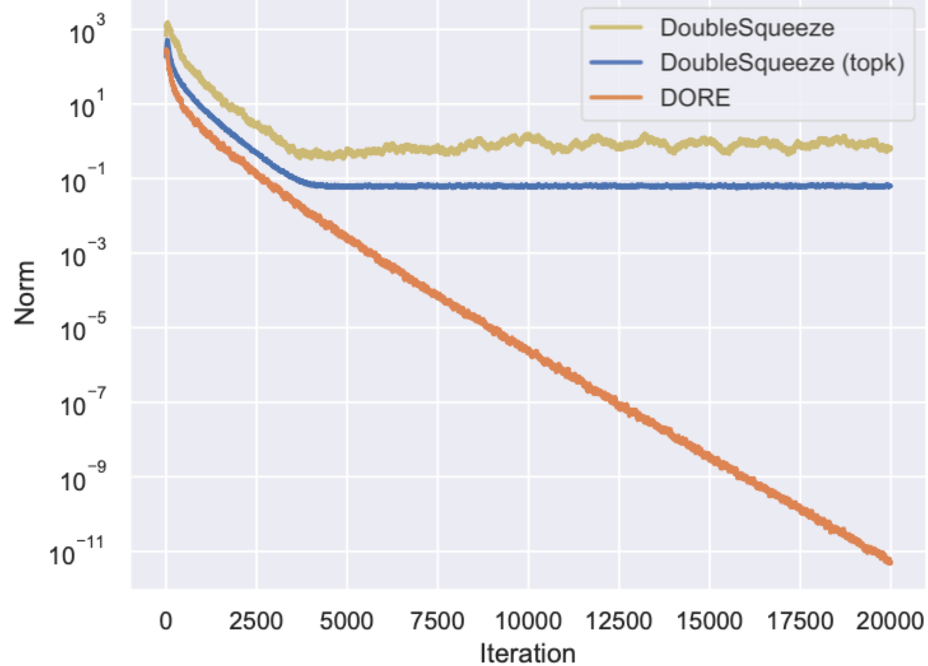
Distance to optimum vs communication bits

Numerical experiment

Compression error



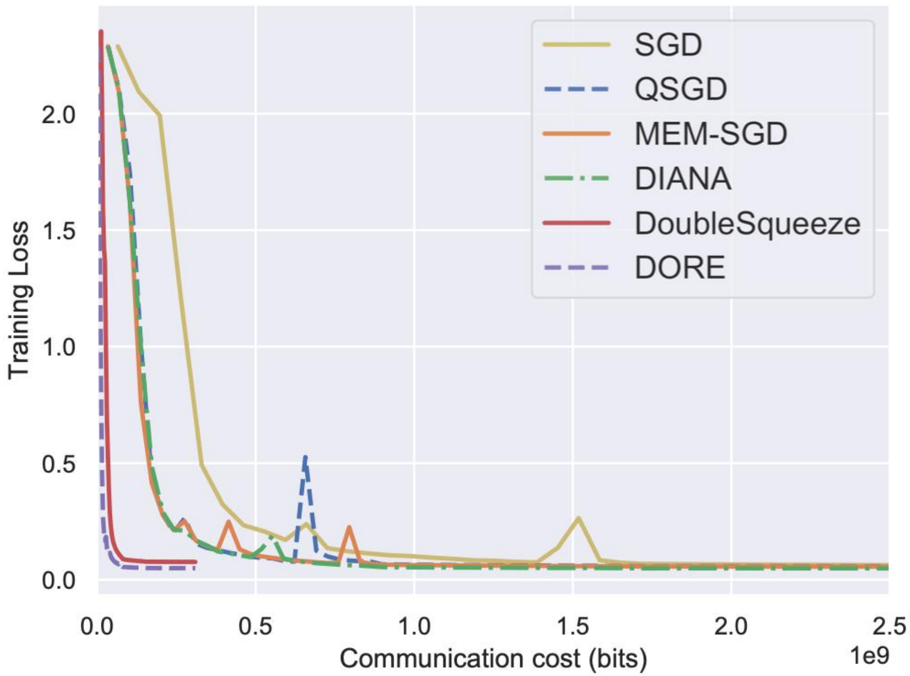
Worker side



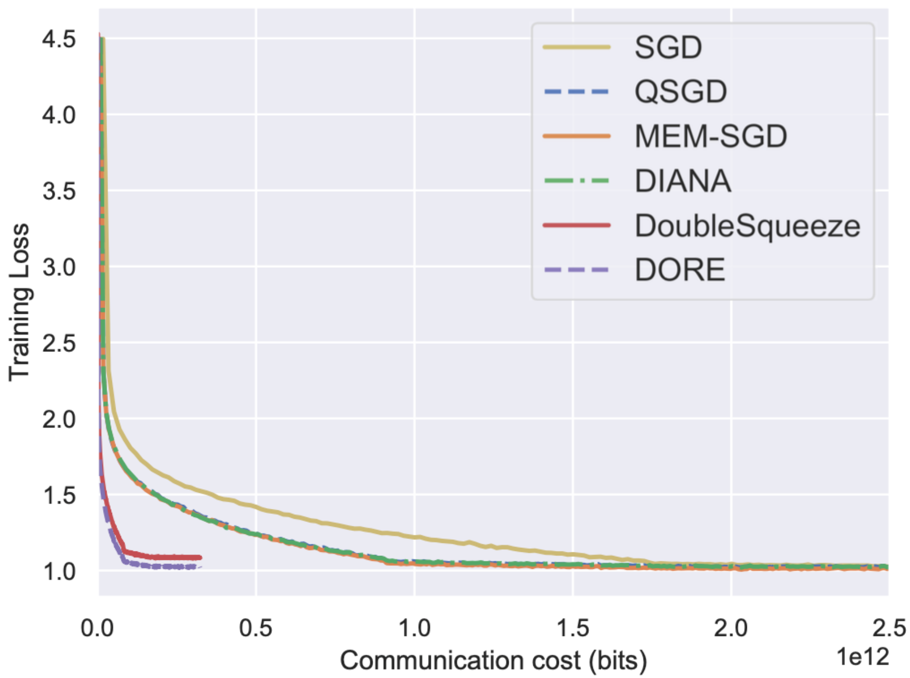
Server side

Numerical experiment

LeNet trained on MNIST

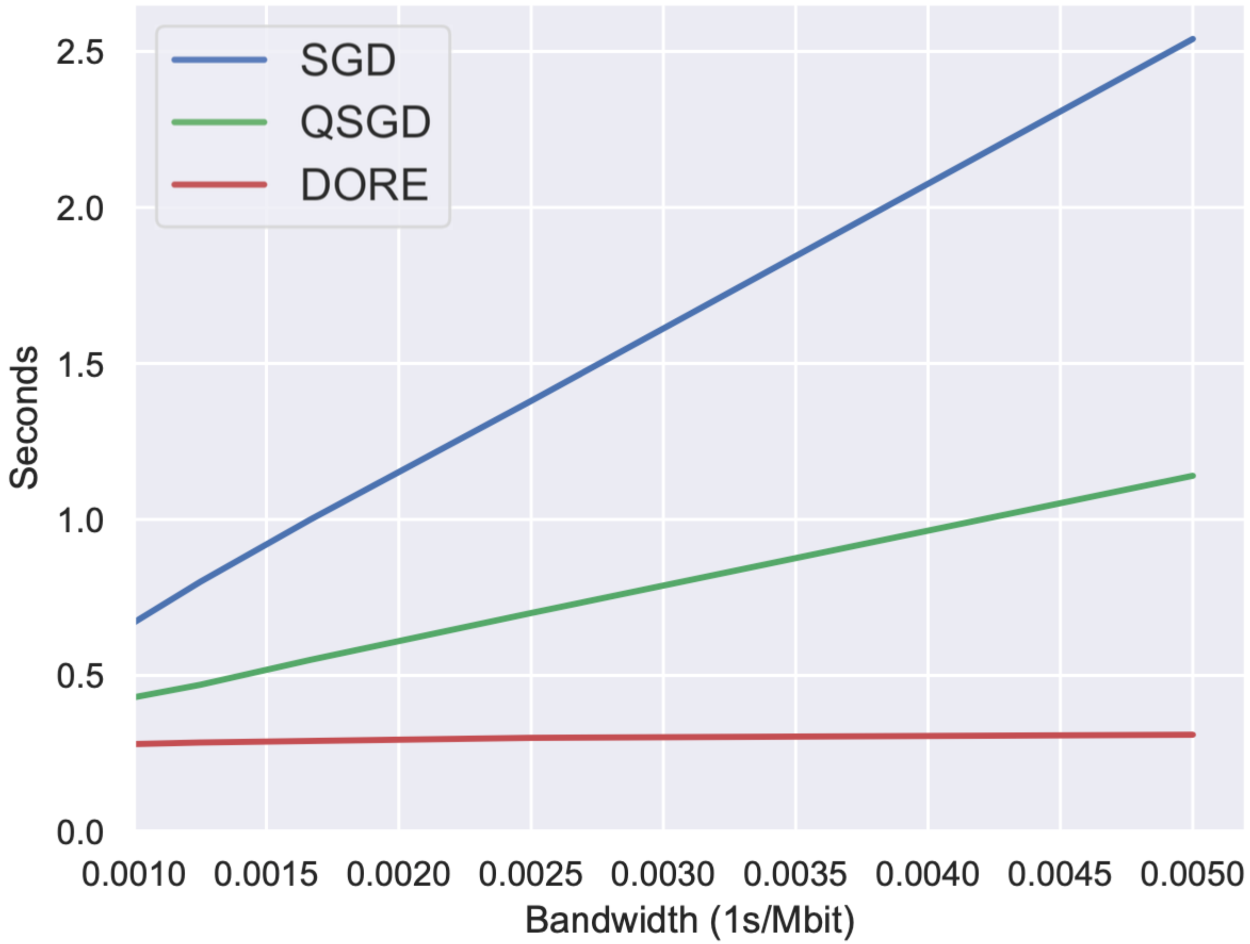


ResNet18 trained on CIFAR10



Train loss vs communication bits

Numerical experiment



Time cost per iteration vs network bandwidth

Conclusion

- DORE reduces over 95% of the communication cost through the double residual compression;
- Provide a sublinear convergence rate in the nonconvex case and achieve linear speedup;
- Provide a linear convergence analysis to the neighborhood of the optimum for smooth and strongly convex functions;
- DORE achieves state-of-art performance both theoretically and empirically;
- We hope to see more applications or extensions of DORE in bandwidth limited settings such as federated learning.