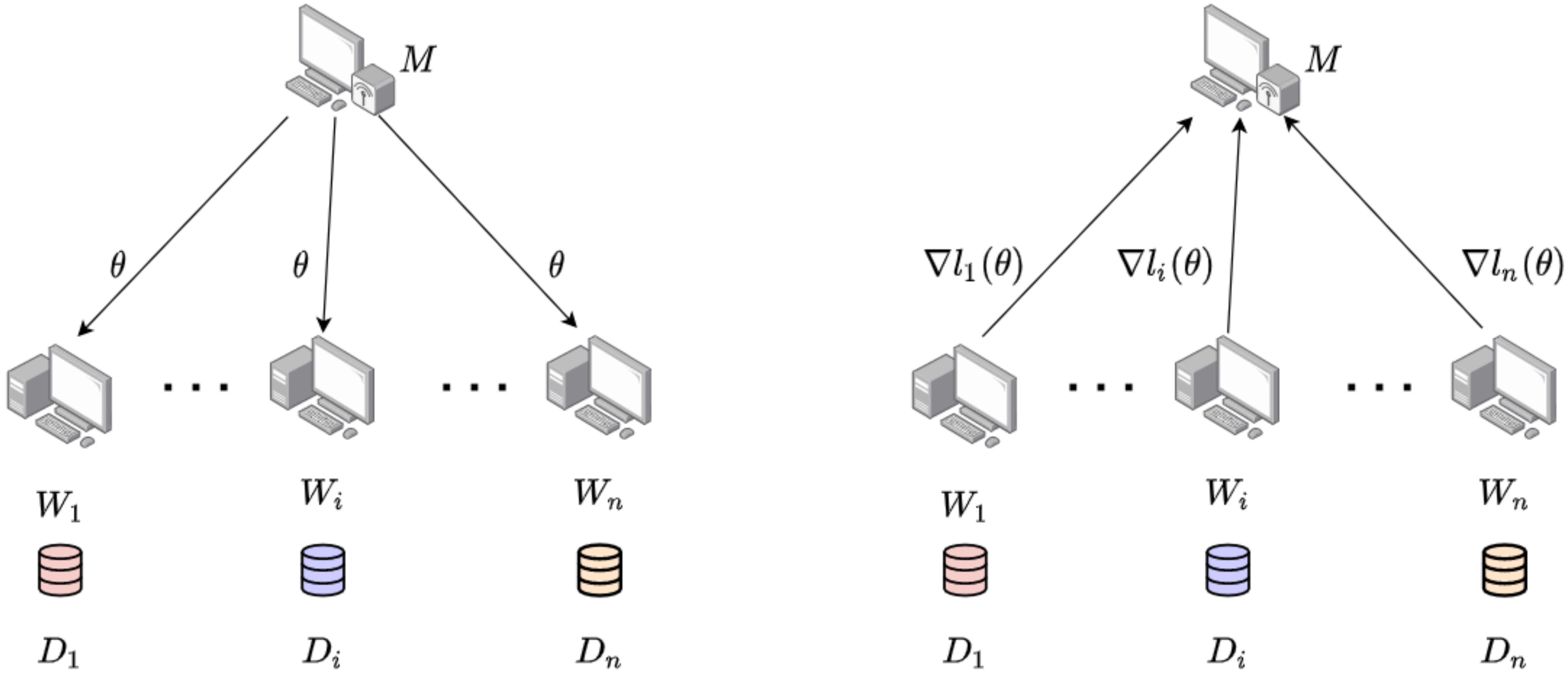
Lightweight Projective Derivative Codes for Compressed Asynchronous Gradient Descent

Pedro Soto, Ilia Ilmer, Haibin Guan, Jun Li

Presented at ICML 2022

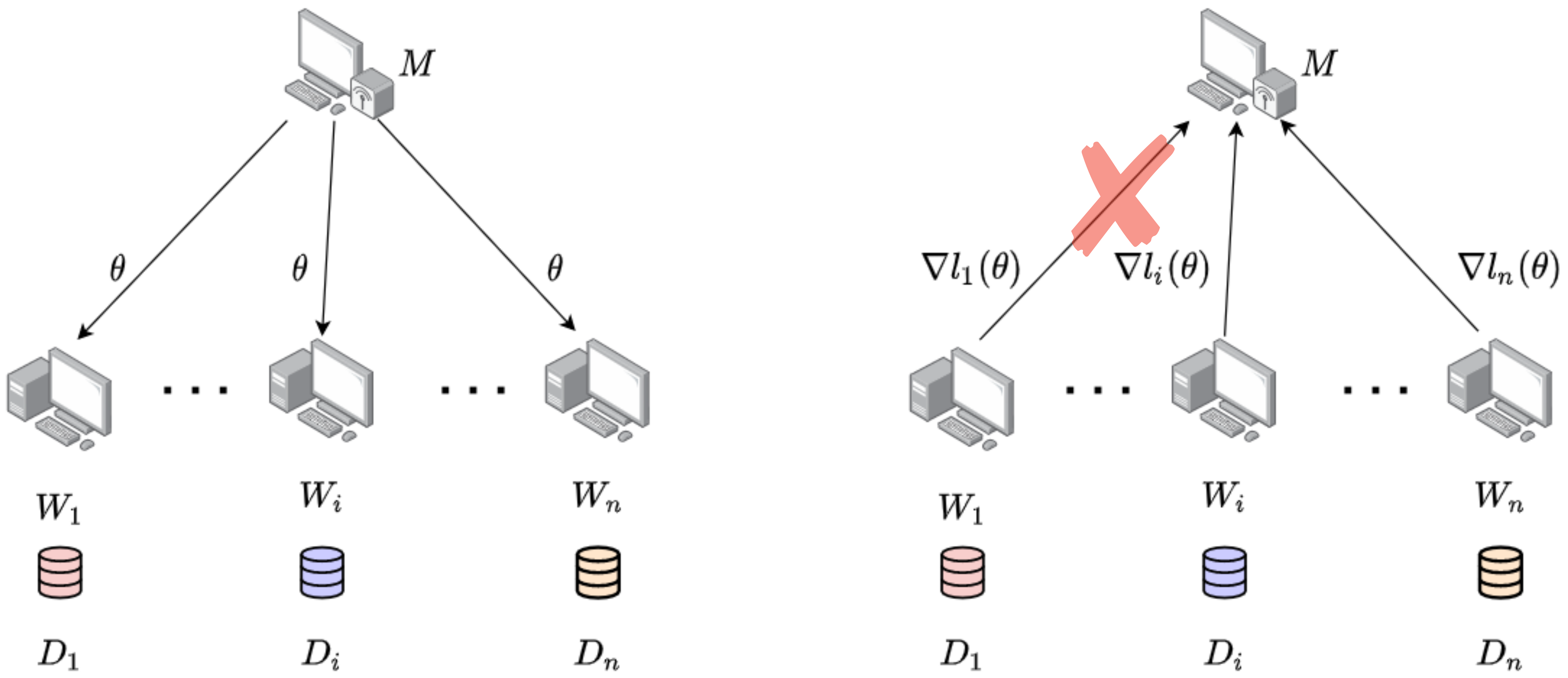
Distributed Gradient Descent



- The task $\nabla \mathscr{L}_D(\theta)$ is too large to do on machine M so it partitions the job into some smaller tasks $f_1(\theta) = \nabla l_1(\theta), \ldots, f_k(\theta) = \nabla l_k(\theta)$ and distributes it amongst the workers.

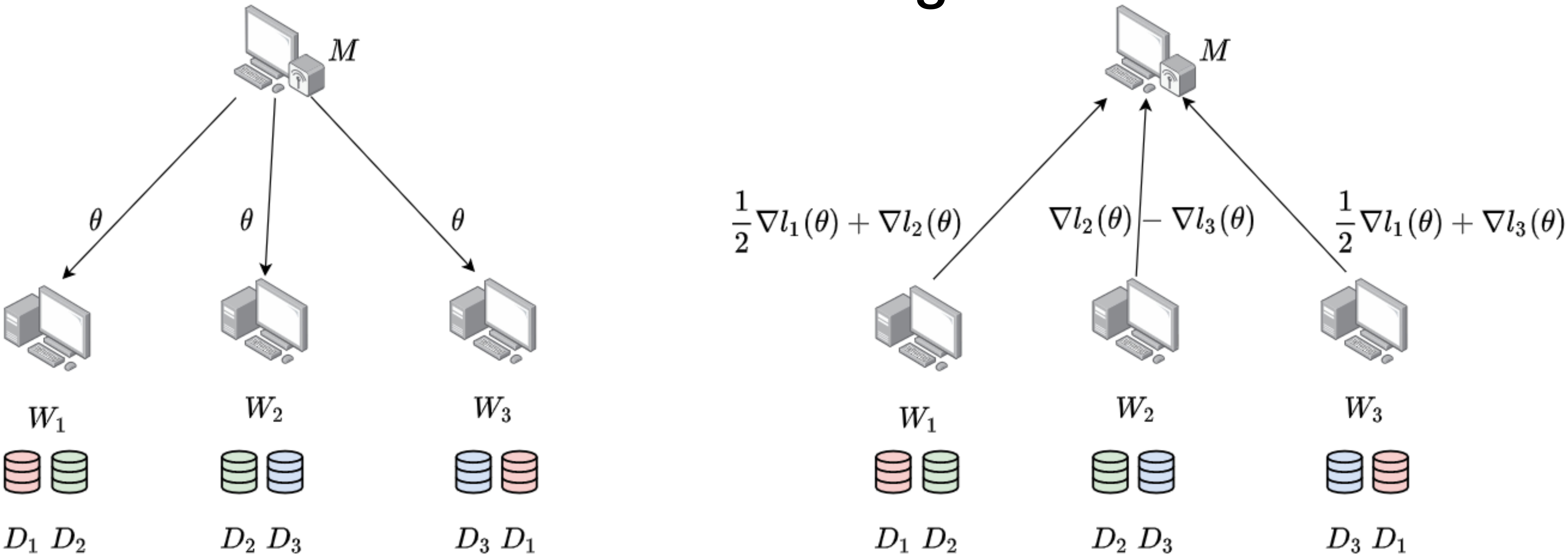
- Unlike other forms of distributed computing, the data being sent to the workers; *i.e.*, θ, is the same.

Distributed Gradient Descent



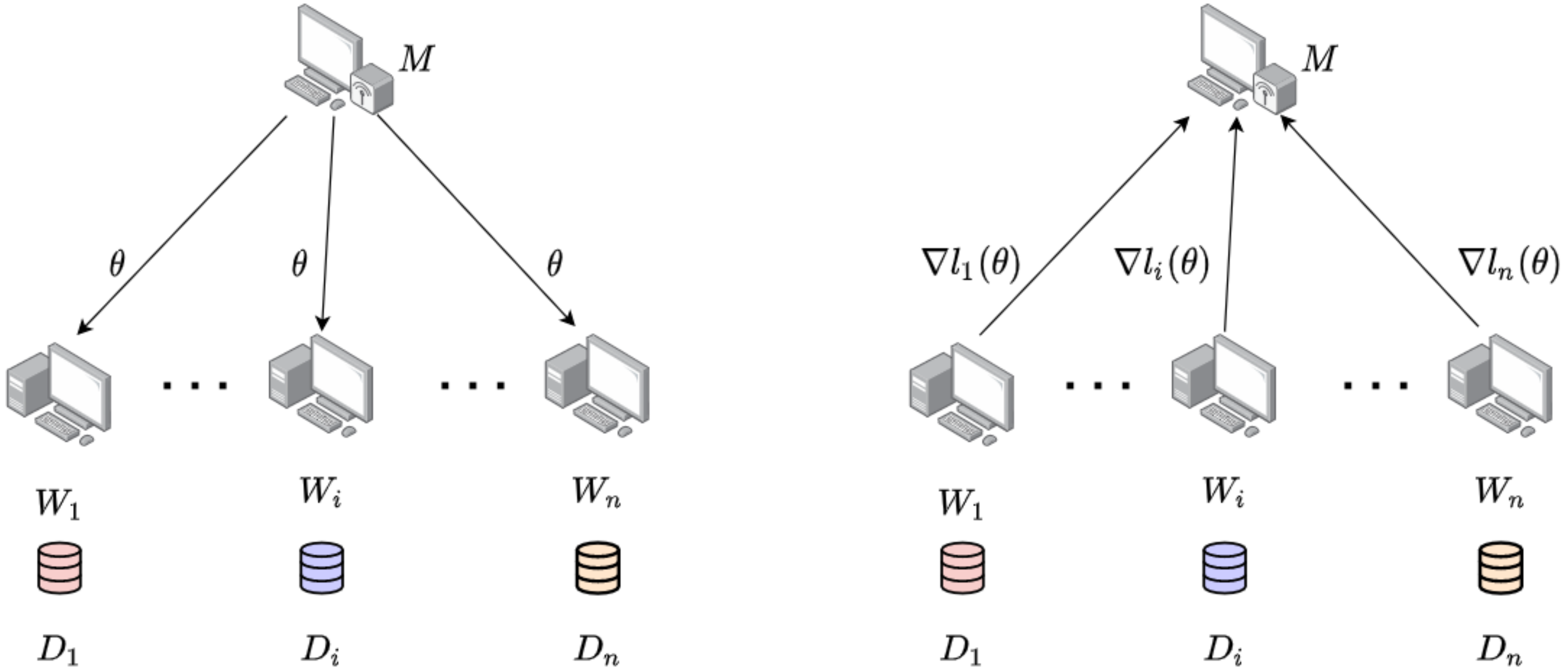
- The overall job can only be completed as fast as the slowest worker
- One solution is to use linear coding techniques

Gradient Coding



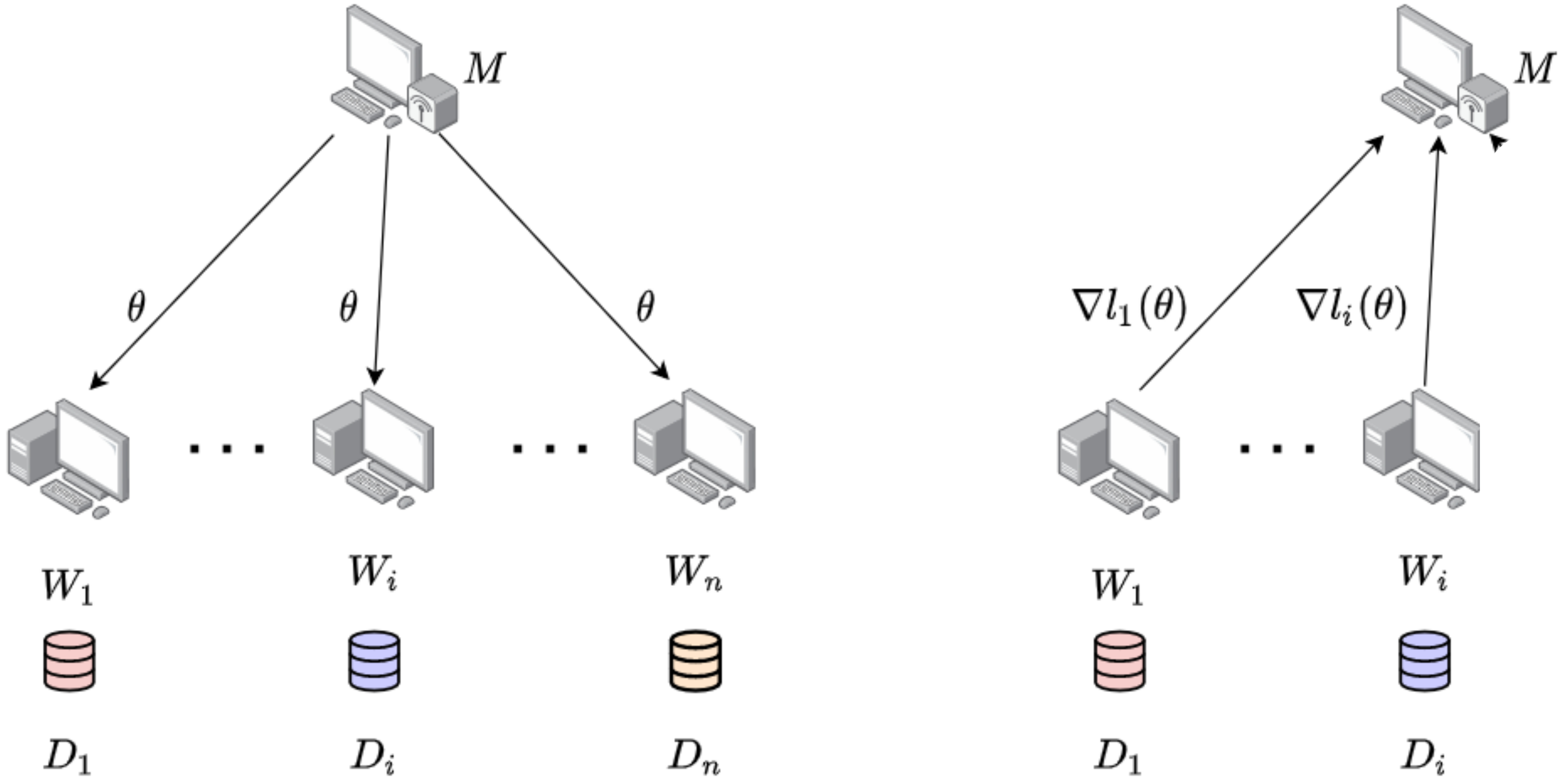
- A small example of a gradient code [Tandon et al. 17] where the master desires the computation of $f(\theta) = \nabla l_1(\theta) + \nabla l_2(\theta) + \nabla l_3(\theta)$
- The current assignment creates a $[2,1]$ code since any two workers can recreate the task of the third
- For example, if the second worker becomes a straggler, then the task can be recreated by computing $f(\theta) = f_1 + f_3$ at M

Asynchronous Distributed Gradient Descent



- In Asynchronous Distributed Gradient Descent [Dutta et al. 2021], the master performs the update before waiting for all workers to return
- In particular, for *K-Asynchronous* Distributed Gradient Descent the master performs the update every time that k workers return
- The workers may have different models at any given moment

Asynchronous Distributed Gradient Descent

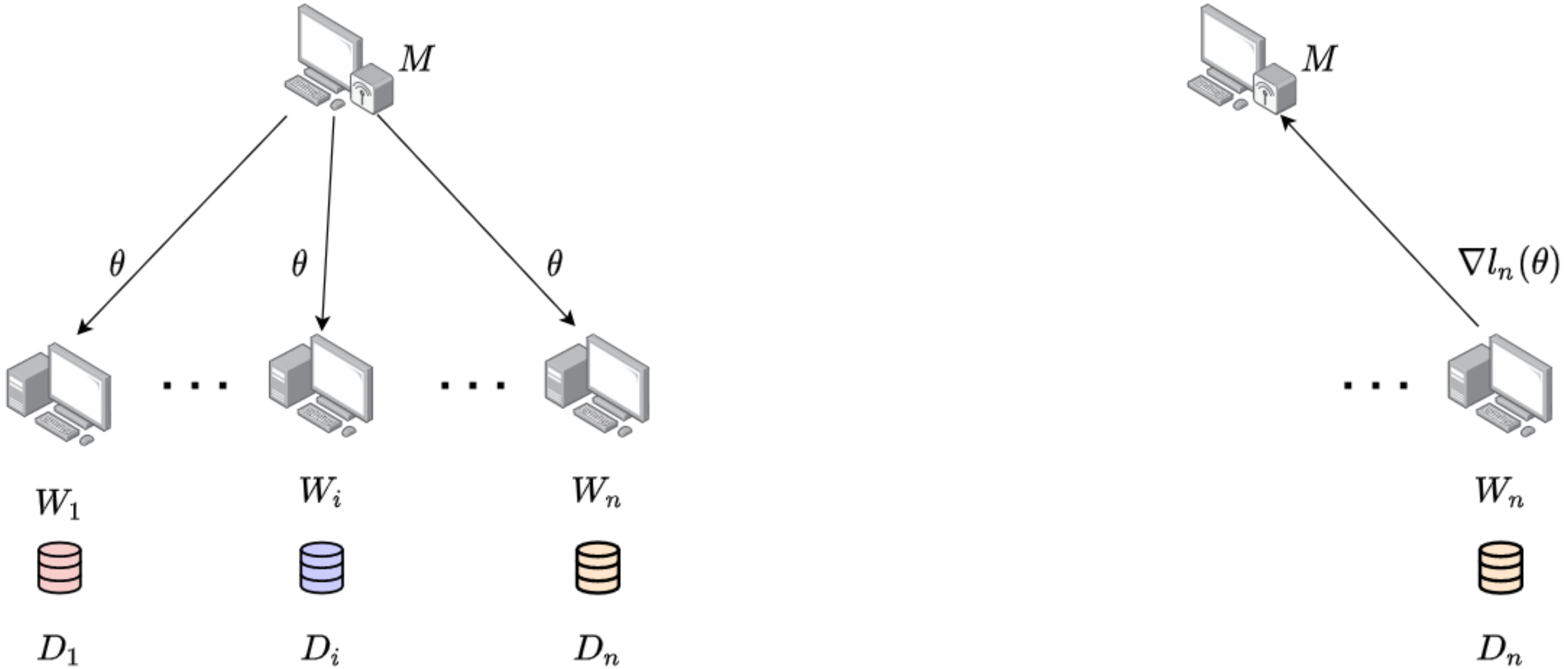


- In Asynchronous Distributed Gradient Descent [Dutta et al. 2021], the master performs the update before waiting for all workers to return

- In particular, for *K-Asynchronous* Distributed Gradient Descent the master performs the update every time that k workers return

- The workers may have different models at any given moment

Asynchronous Distributed Gradient Descent



- In Asynchronous Distributed Gradient Descent [Dutta et al. 2021], the master performs the update before waiting for all workers to return

- In particular, for *K-Asynchronous* Distributed Gradient Descent the master performs the update every time that k workers return

- The workers may have different models at any given moment

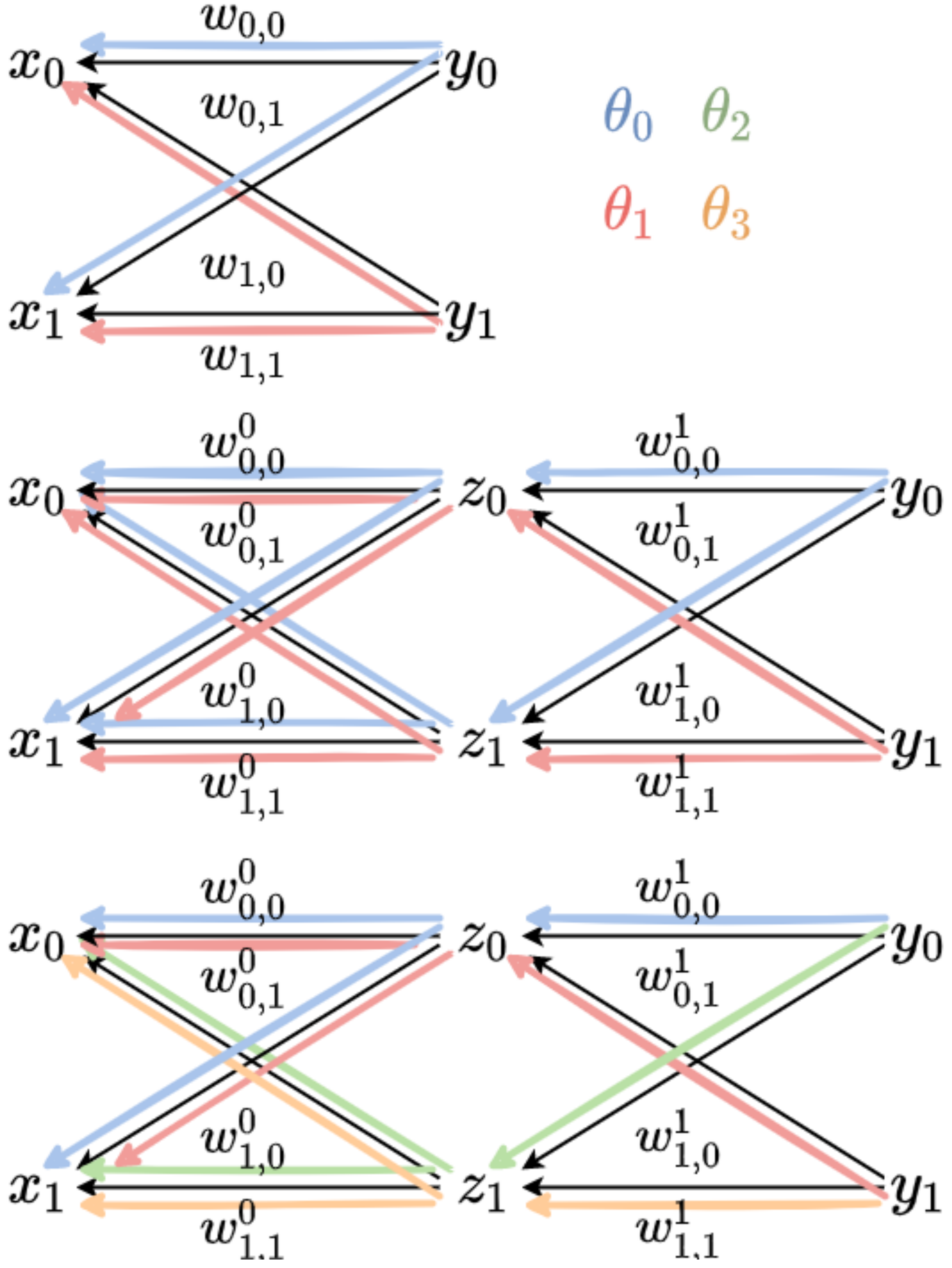
The LWPD Code $\mathscr{C}^{(8,4,2)}$

- $\dfrac{\partial}{\partial\theta_0}$ = ``the derivative of the first half of the output nodes with respect to the first half of the dataset'',
- $\dfrac{\partial}{\partial\theta_1}$ = ``the derivative of the second half of the output nodes with respect to the first half of the dataset'',
- $\dfrac{\partial}{\partial\theta_2}$ = ``the derivative of the first half of the output nodes with respect to the second half of the dataset'', and
- $\dfrac{\partial}{\partial\theta_3}$ = ``the derivative of the second half of the output nodes with respect to the second half of the dataset''.

$$
\begin{array}{c} \\ \tilde{\theta}_0 \\ \tilde{\theta}_1 \\ \tilde{\theta}_2 \\ \tilde{\theta}_3 \\ \tilde{\theta}_4 \\ \tilde{\theta}_5 \\ \tilde{\theta}_6 \\ \tilde{\theta}_7 \end{array}
\begin{array}{c}
\begin{array}{cccc} \theta_0 & \theta_1 & \theta_2 & \theta_3 \end{array} \\
\begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\
\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\
0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\
0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\
0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\
\frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\
-\frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}}
\end{bmatrix}
\end{array}
$$

Parameter Compression

- The different ways to partition the backpropagation gradient.
- The first partition shows how to partition the gradient for a simple neural network with no hidden nodes.
- The two other partition corresponds to a more general deep-neural network with hidden nodes.
- The last partition depicts the recursive construction for larger n



Contributions

- The main contribution of this work is to construct a gradient coding scheme that is asynchronous
  - In particular, not all the information is needed to decode
  - Previous works have considered both separately but not simultaneously
  - To achieve this we construct the code so that it can be iteratively decoded
  - If the code has rank k we can update the gradient with less than k workers
- We can further lower communication complexity by compressing the partial gradients sent back to the master
- A theoretical contribution is to come up with the correct definition of distance between code-words that maximizes information returned by subsets of coded gradients

Comparison of Main Algorithms

Code Scheme	Encoding Complexity	Communication Complexity	Decoding Complexity
LPDC	0	$\mathcal{O}(\frac{k}{t})$	0
GC	$\mathcal{O}(nk)$	$\mathcal{O}(k)$	$\mathcal{O}(k^{\omega}) \leq \mathcal{O}(k^{2.38})$
K-AC	0	$\mathcal{O}(k)$	0

Code Scheme	Weight Range	Asynchronous ?	Parameter Compression?
LPDC	$t \in [2, \frac{n}{4}]$	$\surd$	$\surd$
GC	$t = n - k + 1$	$\times$	$\times$
K-AC	$t \in [1, n]$	$\surd$	$\times$

- (LPDC) Lightweight Projective Derivative Codes
- GC) Gradient Coding \cite{pmlr-v70-tandon17a}
- (K-AC) K-Asynchronous Gradient Descent

Experimental Results

References

- [Tandon et al. 17] Rashish Tandon et al. “Gradient Coding: Avoiding Stragglers in Distributed Learning”. In: Proceedings of the 34th International Conference on Machine Learning. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3368–3376. URL: https : //proceedings.mlr.press/v70/tandon17a.html.

- [Dutta 21] Sanghamitra Dutta et al. “Slow and Stale Gradients Can Win the Race”. In: IEEE Journal on Selected Areas in Information Theory 2 (2021), pp. 1012–1024.