Distributed Fourier Neural Operators

Thomas J. Grady[1], Rishi Khan[2], Felix J. Herrmann[1]

SLIM
Georgia Institute of Technology

ML4Seismic

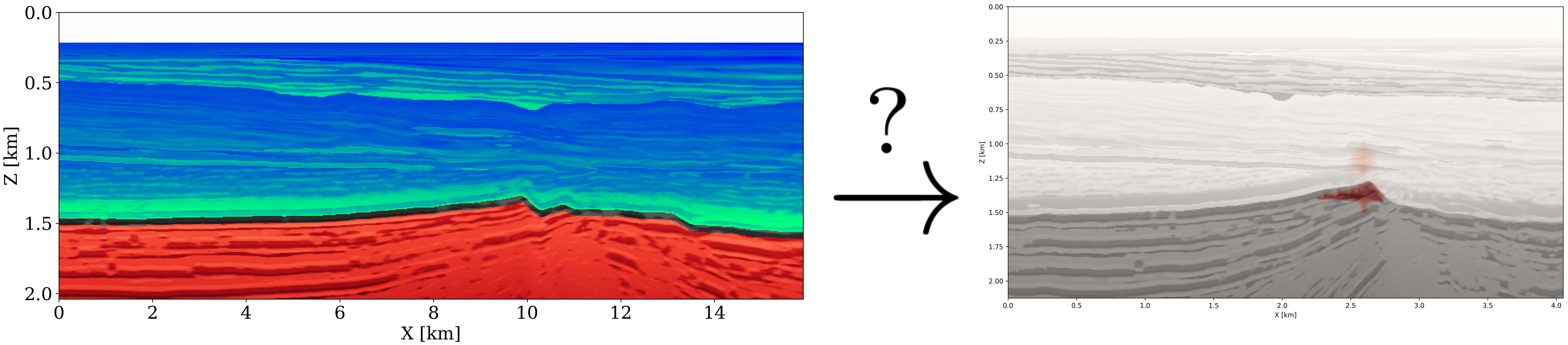
[1]Georgia Tech, [2]Extreme Scale Solutions

Motivation

Given an earth model (porosity/permeability), simulate CO_2 plumes

- over long periods of time
- for different scenarios (earth models, injection rates, etc.)

Important component of seismic monitoring for CCS

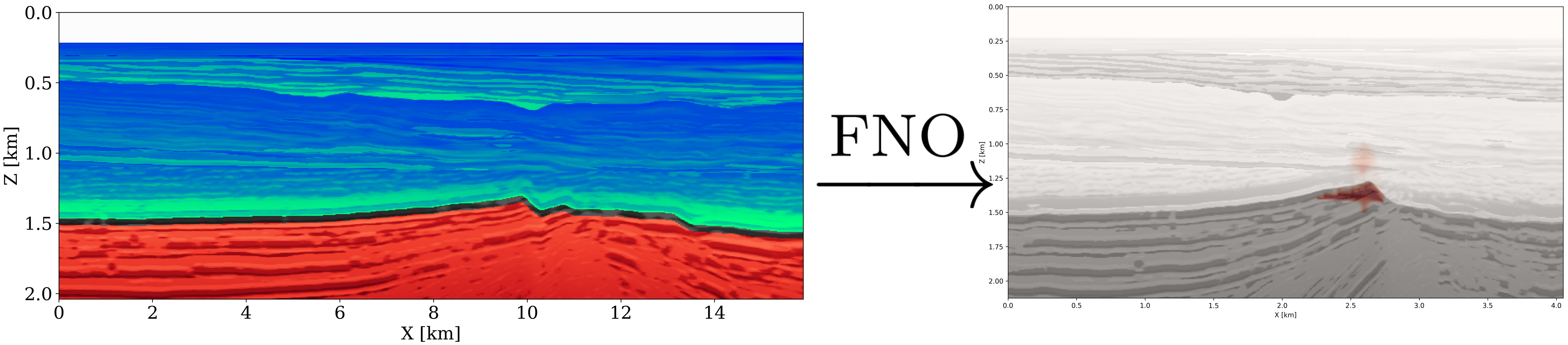


Zongyi Li et al. "Fourier neural operator for parametric partial differential equations." arXiv preprint arXiv:2010.08895 (2020).

Motivation

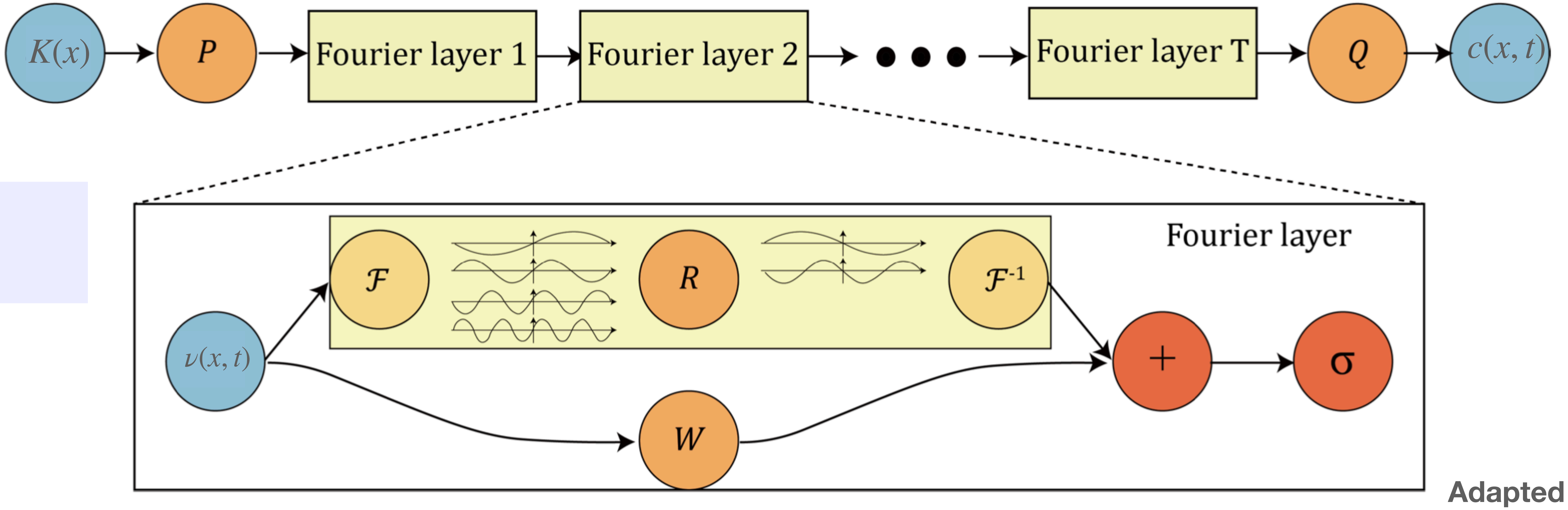
Use Fourier Neural Operators (FNOs) to learn development of CO_2 plumes

- Generalize to families of PDEs (i.e. differing parameters)
- Up to three orders of magnitude faster than numerical solvers once trained
- Discretization scale-invariance



Zongyi Li et al. "Fourier neural operator for parametric partial differential equations." arXiv preprint arXiv:2010.08895 (2020).
Ziyi Yin. Figure shown.

Background - FNOs



Adapted from Li

Input K in $\left(x, y, K(x, y)\right)$, output $c(x, y, t)$

P lifts to higher latent dimensions and Q projects to target dimension

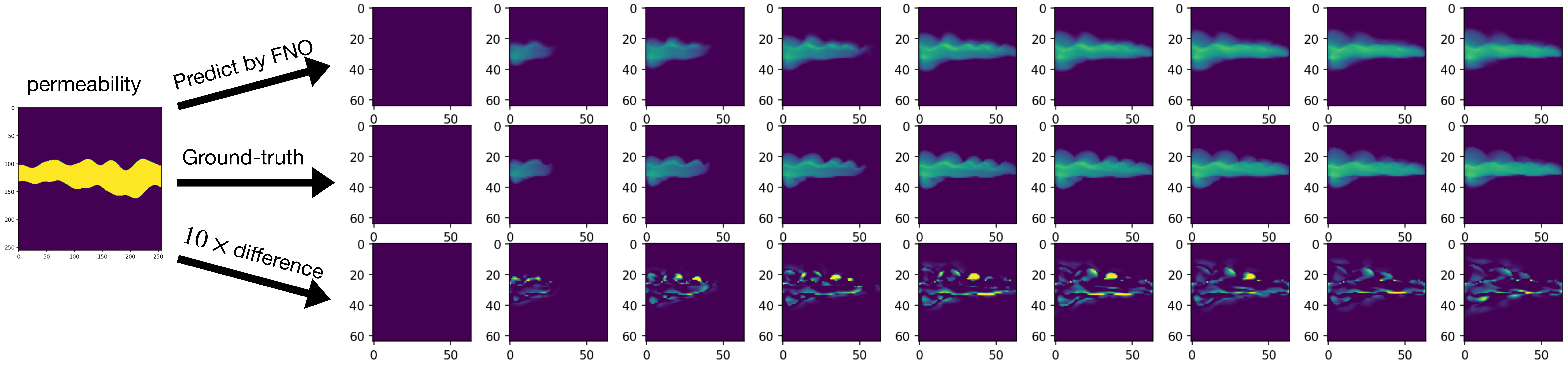
A Fourier layer reads $v_{j+1} = \sigma\left(Wv_j + \mathcal{F}^{-1}\left(R_\phi \cdot \left(\mathcal{F}v_j\right)\right)\right)$

https://github.com/lidongzh/FwiFlow.jl
https://github.com/zongyi-li/fourier_neural_operator

Ziyi Yin. Unpublished work.

Example - FNOs

learning two-phase flow

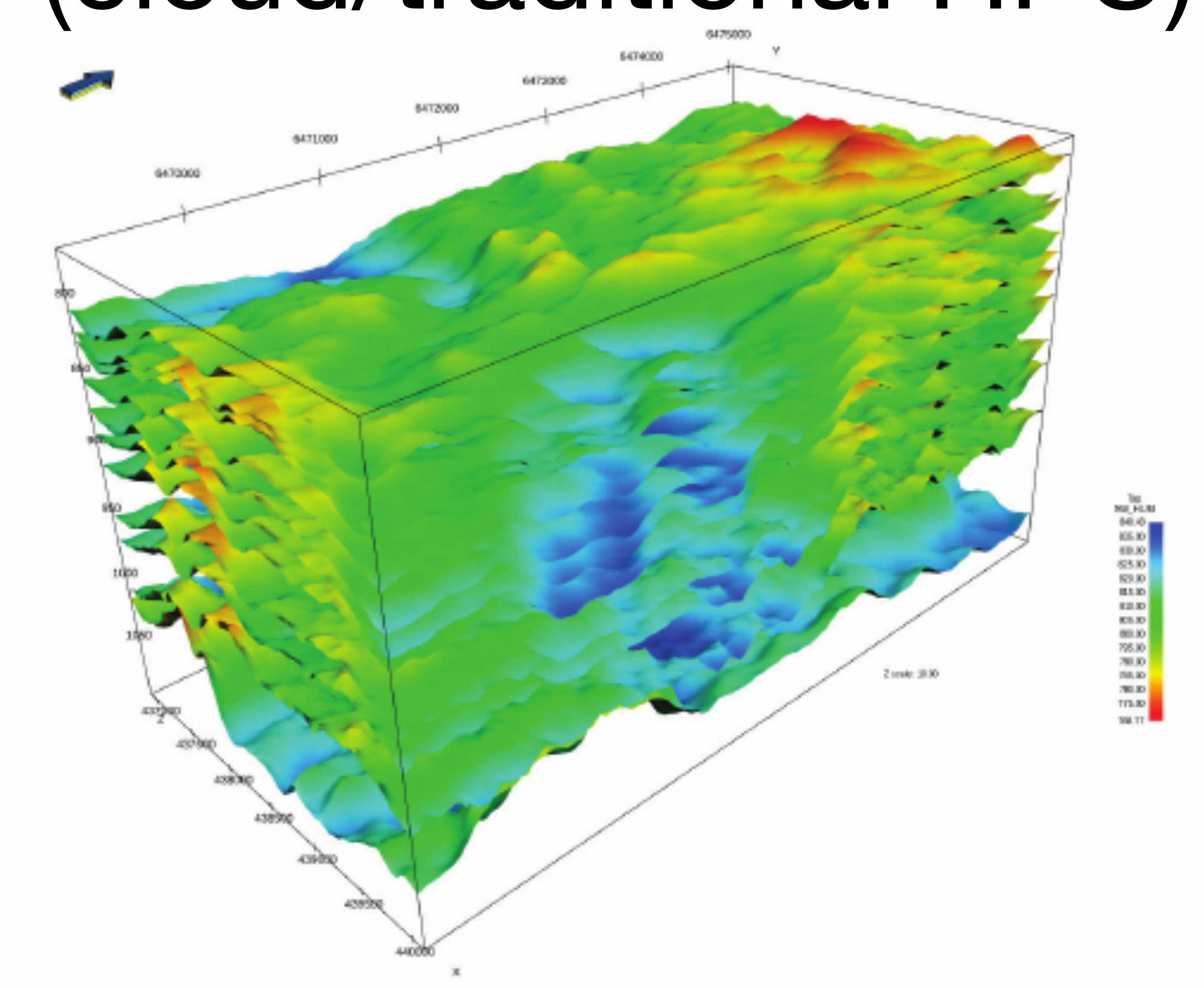


- ▶ Generate 1000 random tortuous channels
- ▶ Simulate 51 snapshots of CO_2 concentrations
- ▶ Map permeability to time evolution of CO_2 concentration

Motivation

Scaling FNOs to realistic problems (3D, large volumes) is a challenge

- problems beyond $64^3\ (x, y, z)$ do not fit w/i GPUs
- real problems are often much larger
  (e.g. Sleipner (low-resolution) is $64 \times 118 \times 263$)
- need high-dimensional model-parallelism on distributed-memory systems (cloud/traditional HPC)

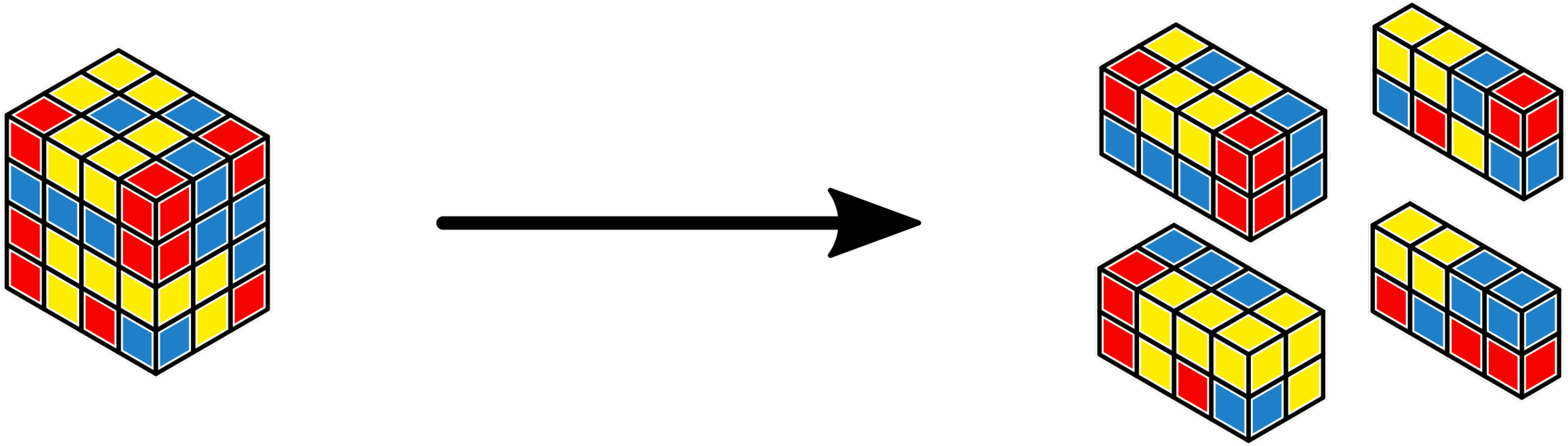


https://distdl.readthedocs.io/en/latest/user_guide/index.html#tensors
https://github.com/distdl/distdl

Solution

DistDL (Hewett, Grady, Merizian) framework provides parallelism

- partition data & model tensors onto different parallel workers along spacetime dimensions
- use advanced MPI functionality to perform parallel computation of neural net functions (convolution, pooling, etc.)

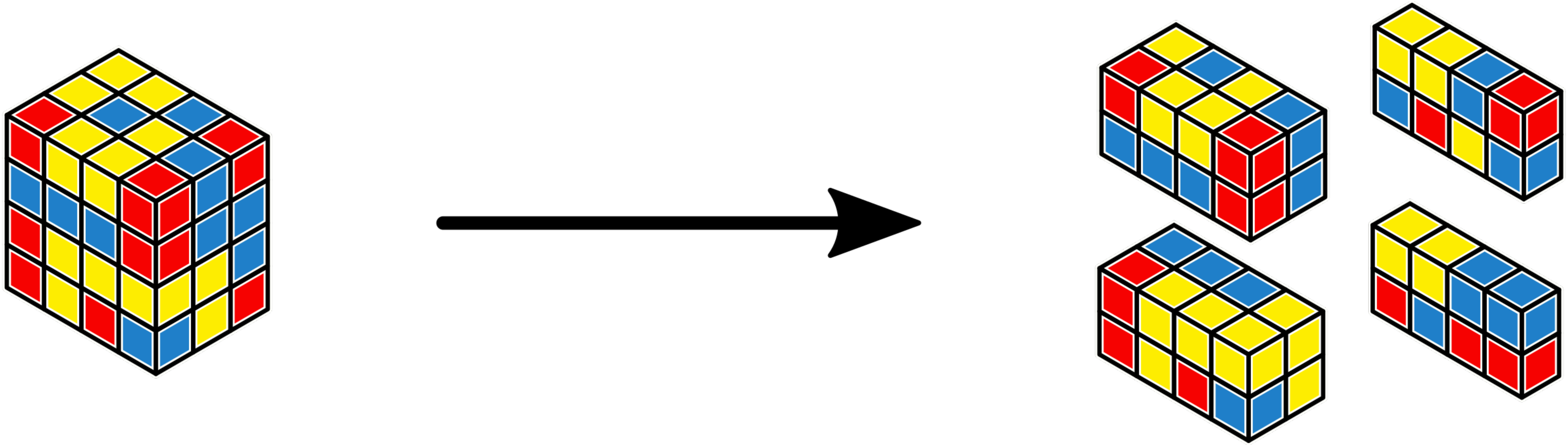


https://distdl.readthedocs.io/en/latest/user_guide/index.html#tensors
https://github.com/distdl/distdl

Solution

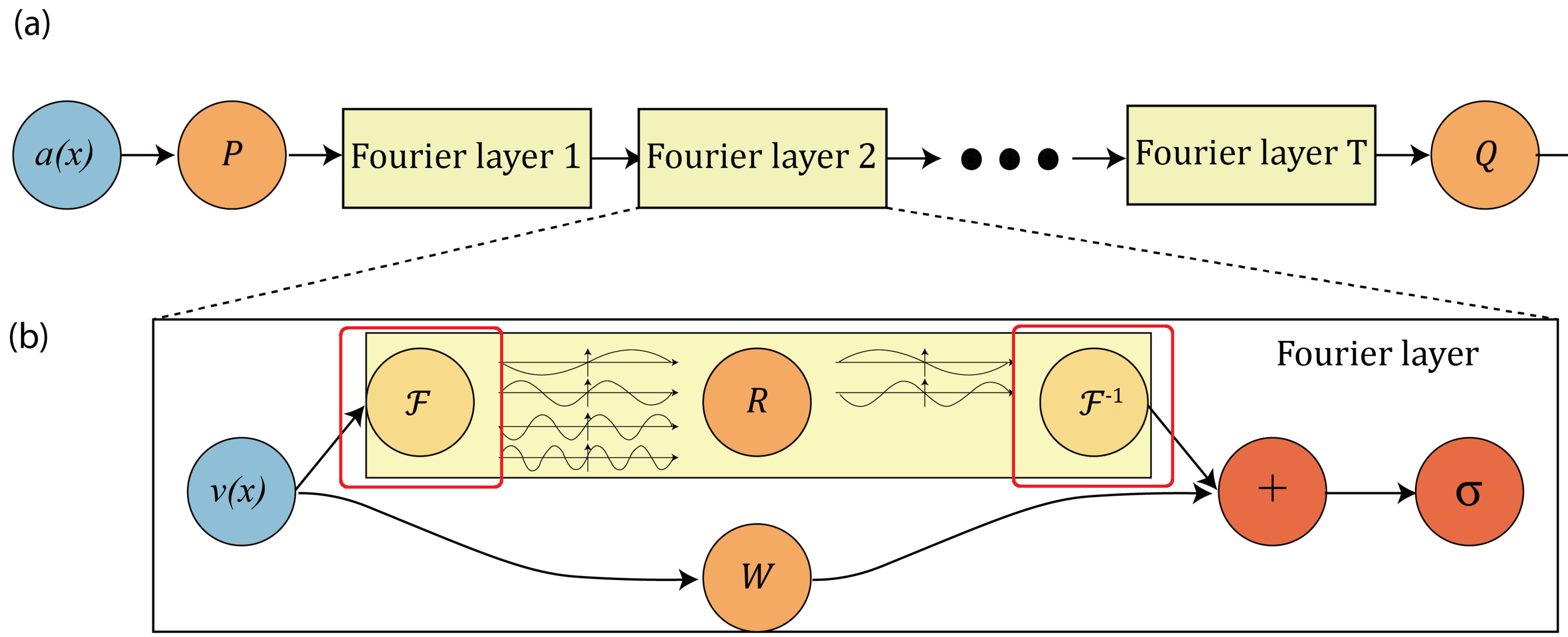
DistDL (Hewett, Grady, Merizian) framework provides parallelism

- implemented w/i PyTorch – differentiable parallelism
- runs on CPU/GPU clusters both cloud & traditional HPC
- for design philosophy & implementation specifics, Breakout Room 3



Parallelism - FNOs

Fourier transform



Lisandro Dalic, Mikael Mortensen, and David E. Keyes. "Fast parallel multidimensional FFT using advanced MPI". arXiv preprint arXiv:1804.09536 (2018).
https://github.com/mpi4py/mpi4py-fft

Parallelism - FNOs

Fourier transform

Challenges

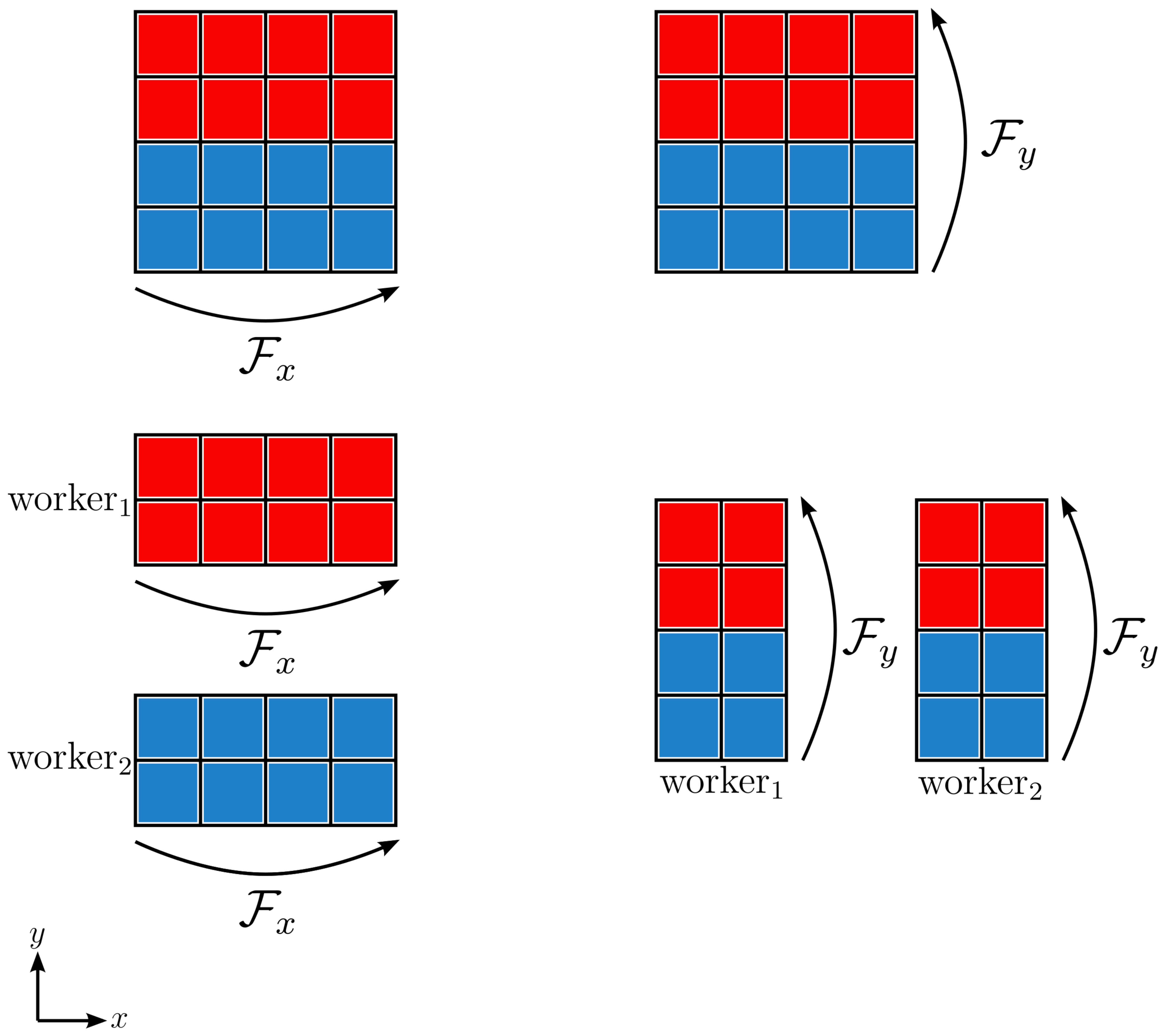
- FFT is a global operation – inherently difficult to parallelize

Distributed FFT algorithm by Dalcin et al.

- multi-dimensional FFT equivalent to repeated FFTs of lower dimension
- switch data partition to keep FFTs along sequential dimension

Parallelism - FNOs

Fourier transform



Parallelism - FNOs

Fourier Transform

Distributed FFT is part of a neural network

- must be differentiable
- must work on tensors of arbitrary size & partition

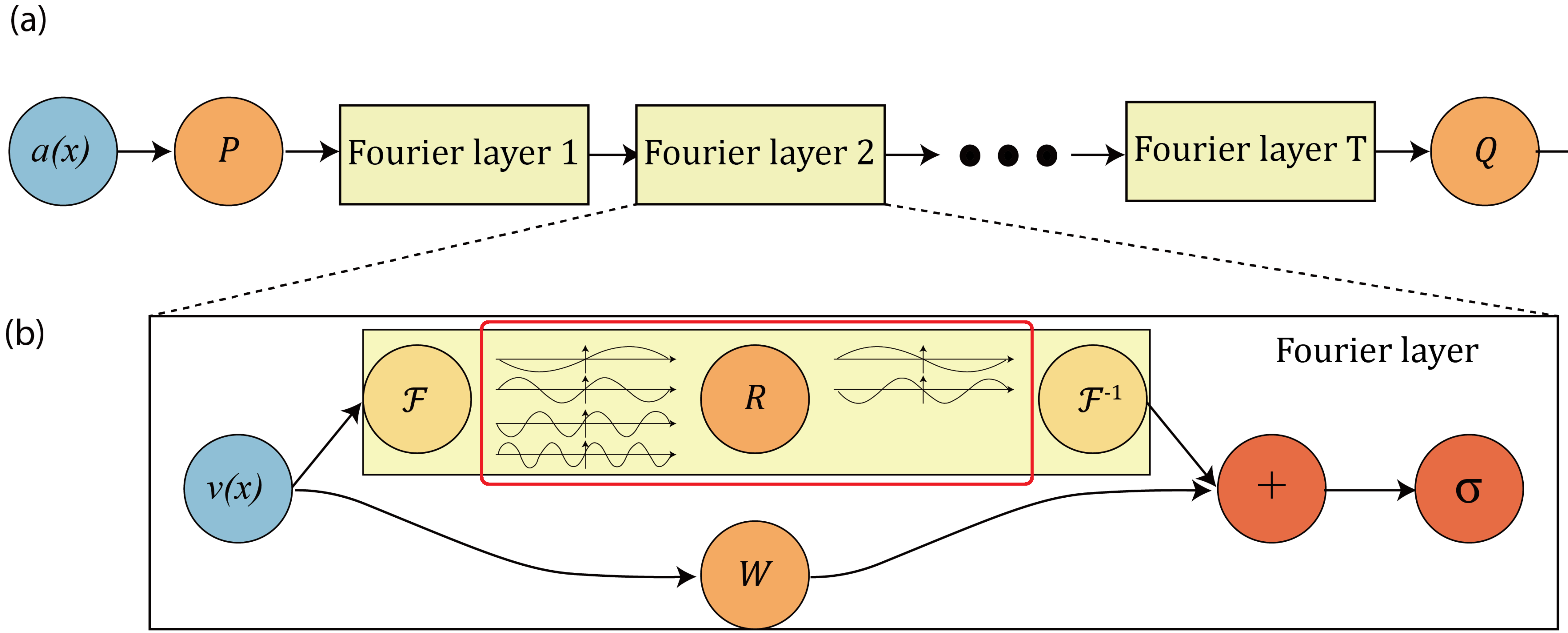
Custom implementation needed, made simple by DistDL

```
def forward(self, x):
    for T, f, kwargs in zip(self.transposes, self.transforms, self.transform_kwargs):
        x = T(x)
        x = f(x, **kwargs)
    x = self.transpose_out(x)
    return x
```

https://arxiv.org/pdf/1804.09536.pdf

Parallelism - FNOs

Spectral Convolution

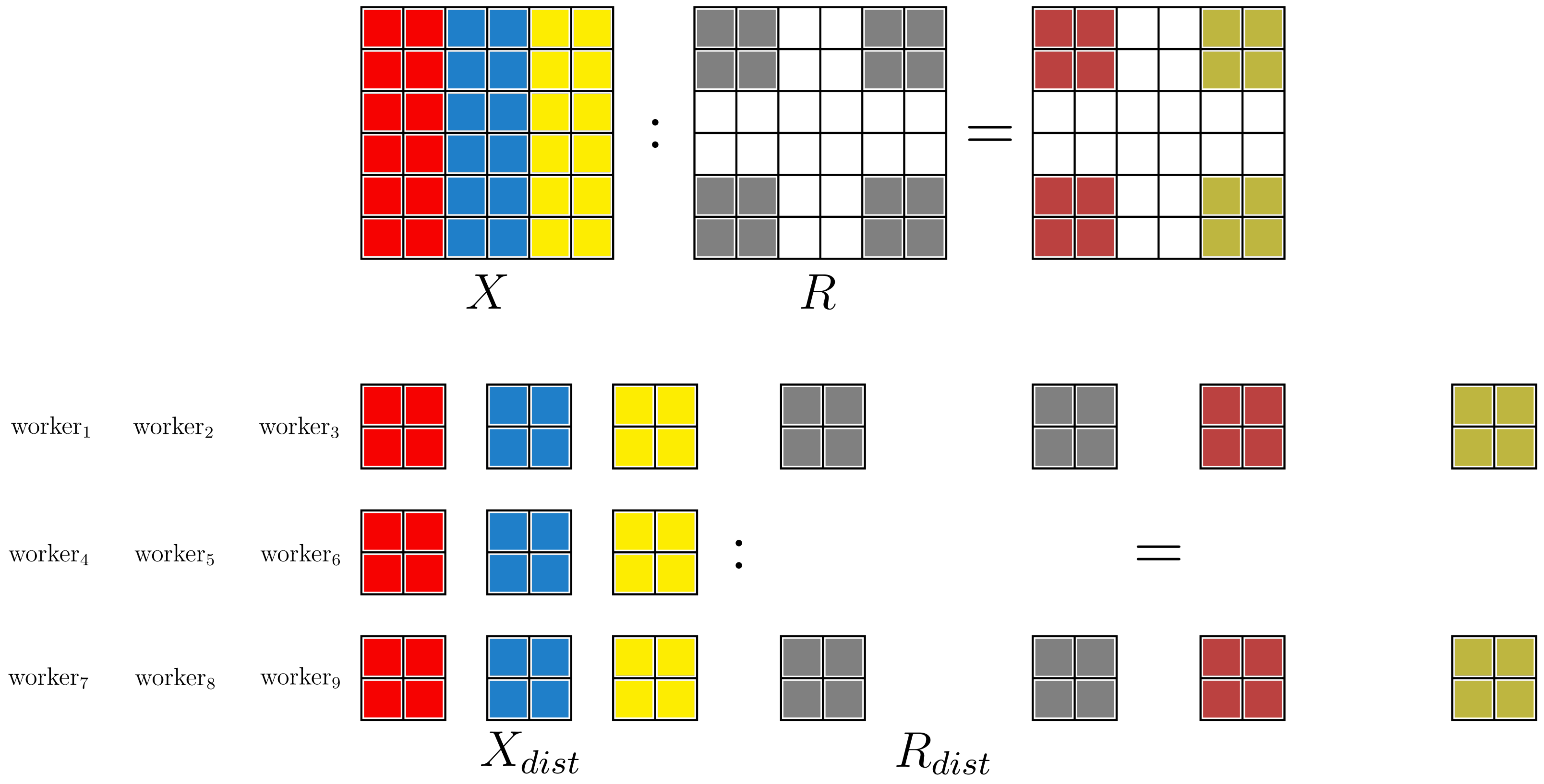


Parallelism - FNOs

Spectral Convolution

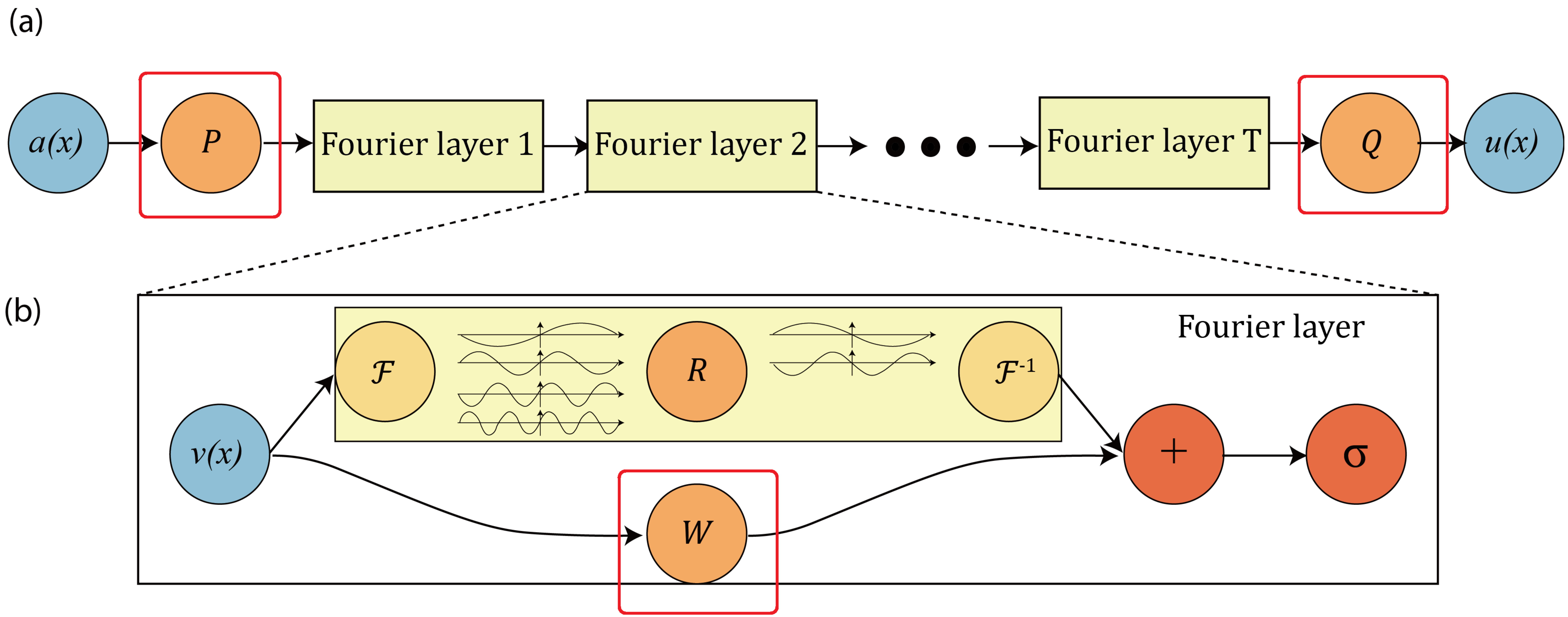
Determine the size of weights on each worker by complex indexing tricks

Only perform computation where restriction operator R is nonzero globally



Parallelism - FNOs

Affine Transformation



Parallelism - FNOs

Affine Transformation

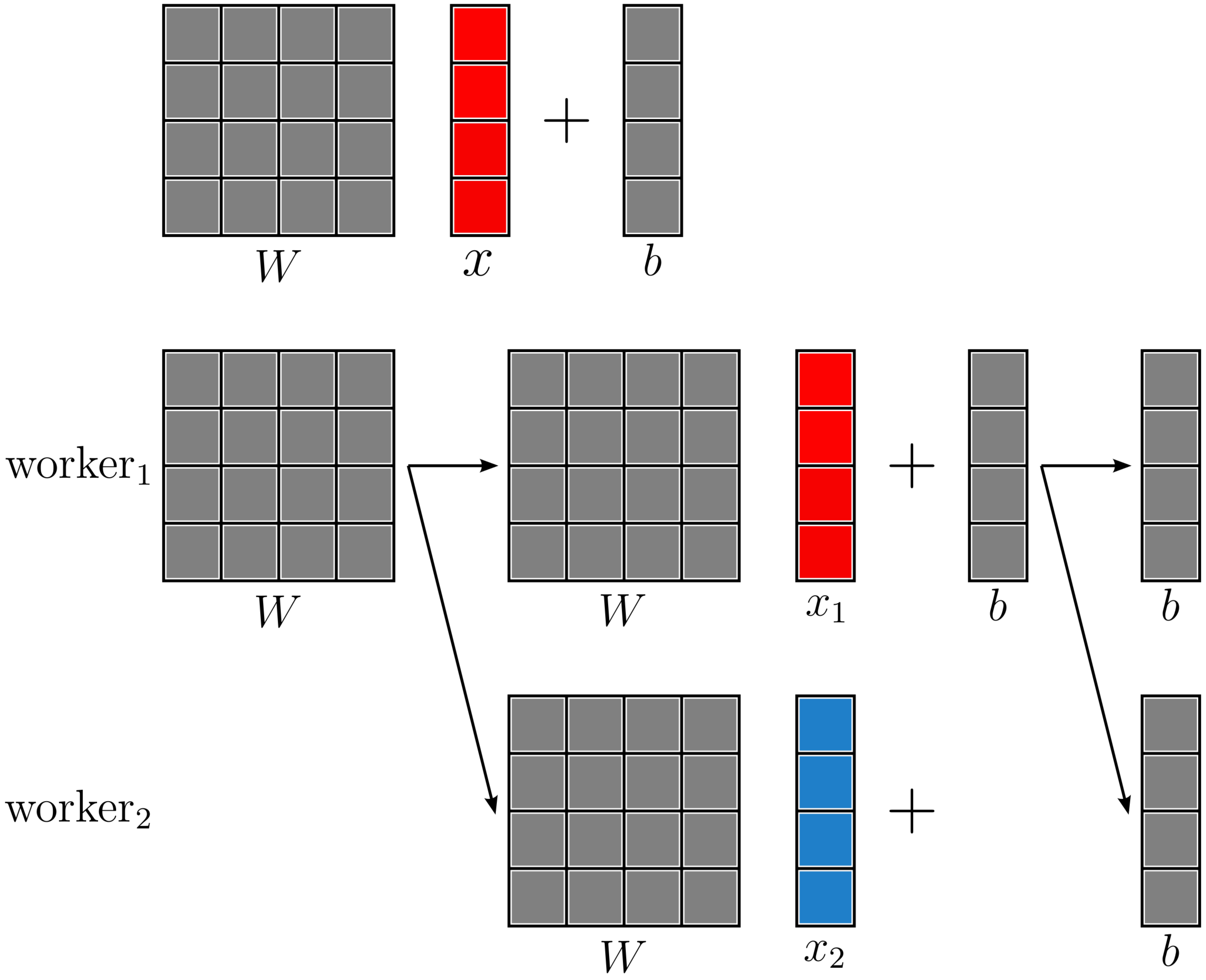
Input network P, output network Q, and weights W all are affine transformations along channel dimension

$$Wx + b$$

Want the action of W to be the same everywhere, so **broadcast** weights & biases before multiplication/addition

Parallelism - FNOs

Affine Transformation



Results

Distributed FNO running on Azure & NERSC Perlmutter

- 64^3 barrier surpassed
- Gradient computed for random input up to $512 \times 512 \times 256$ in spatial dimensions
- Capability to train FNOs on real-world data on distributed memory systems

https://www.nersc.gov/systems/perlmutter/

Results

gradient timing experiment

Run on NERSC Perlmutter cluster

- 10 gradient computations per run, take the average
- Max problem size reached $512 \times 512 \times 256$ in spatial dimensions
- Simultaneous usage of 10TB of A100 GPU memory for a single gradient computation – true HPC scale

Results

gradient timing experiment

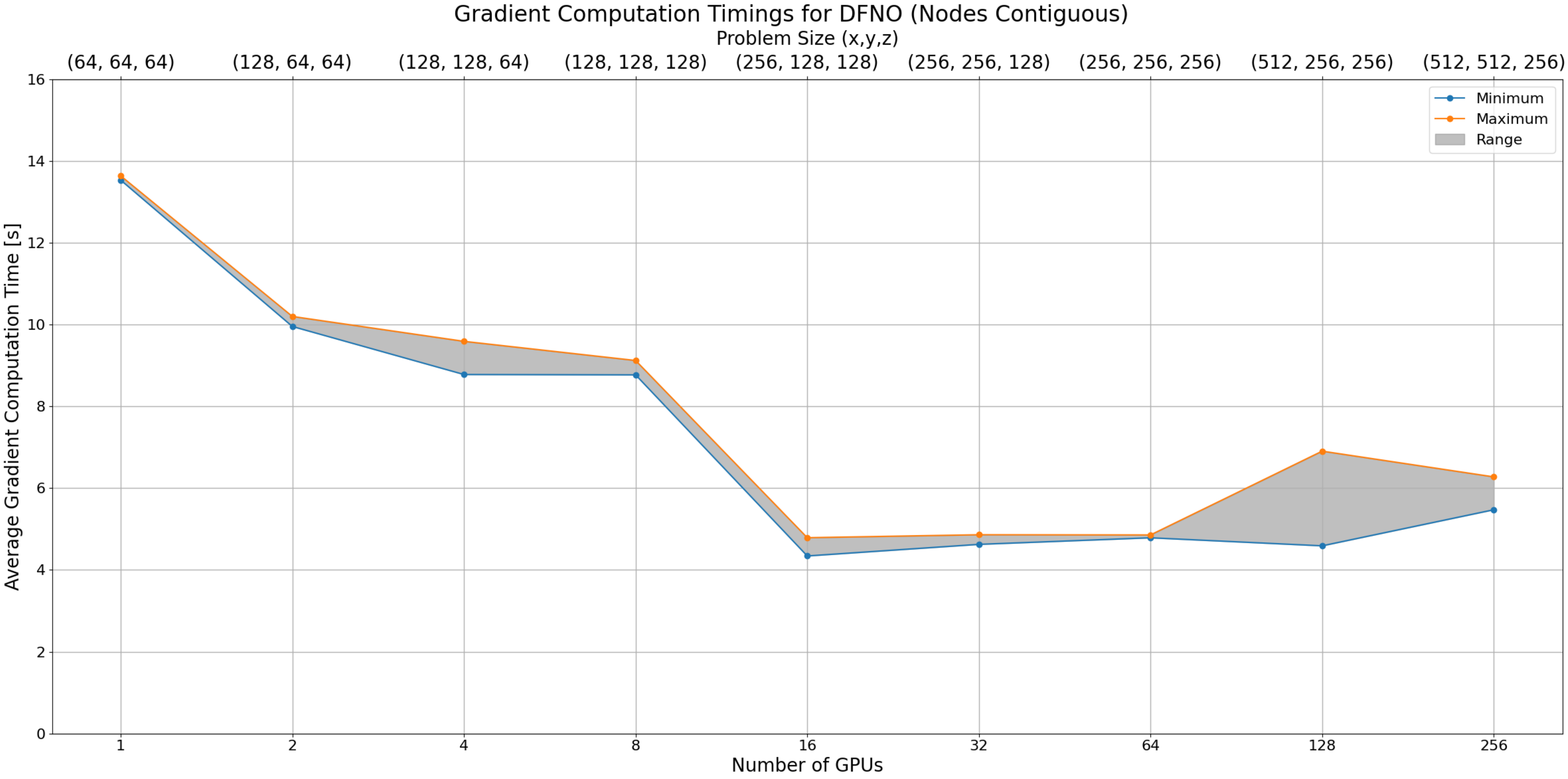
Weak scaling experiment

- data size & weight sizes scale with number of GPUs

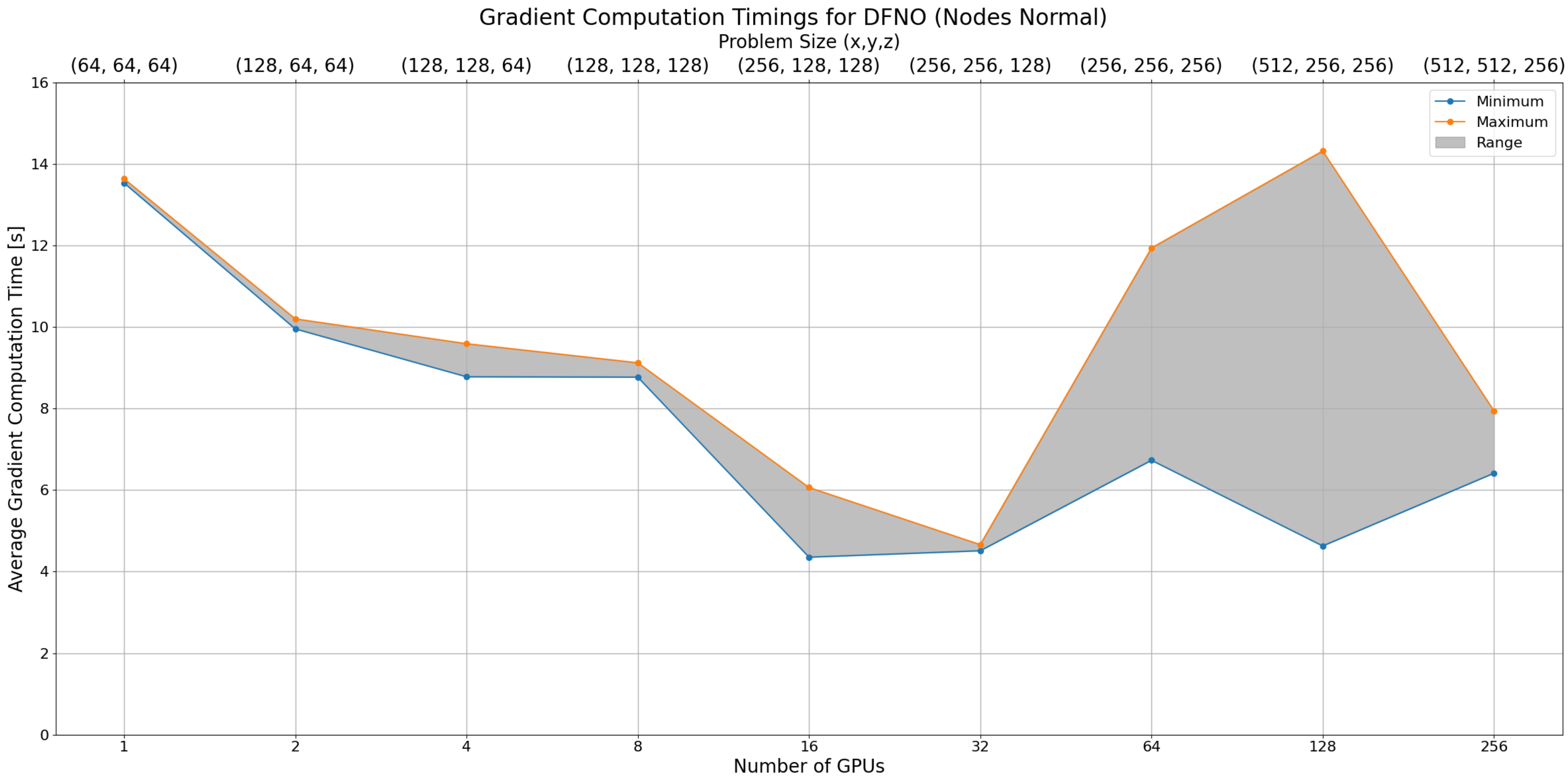
Performance

- speedup due to structure of the network and performance of A100s
- contiguous assignment of workers essential

Results
gradient timing experiment

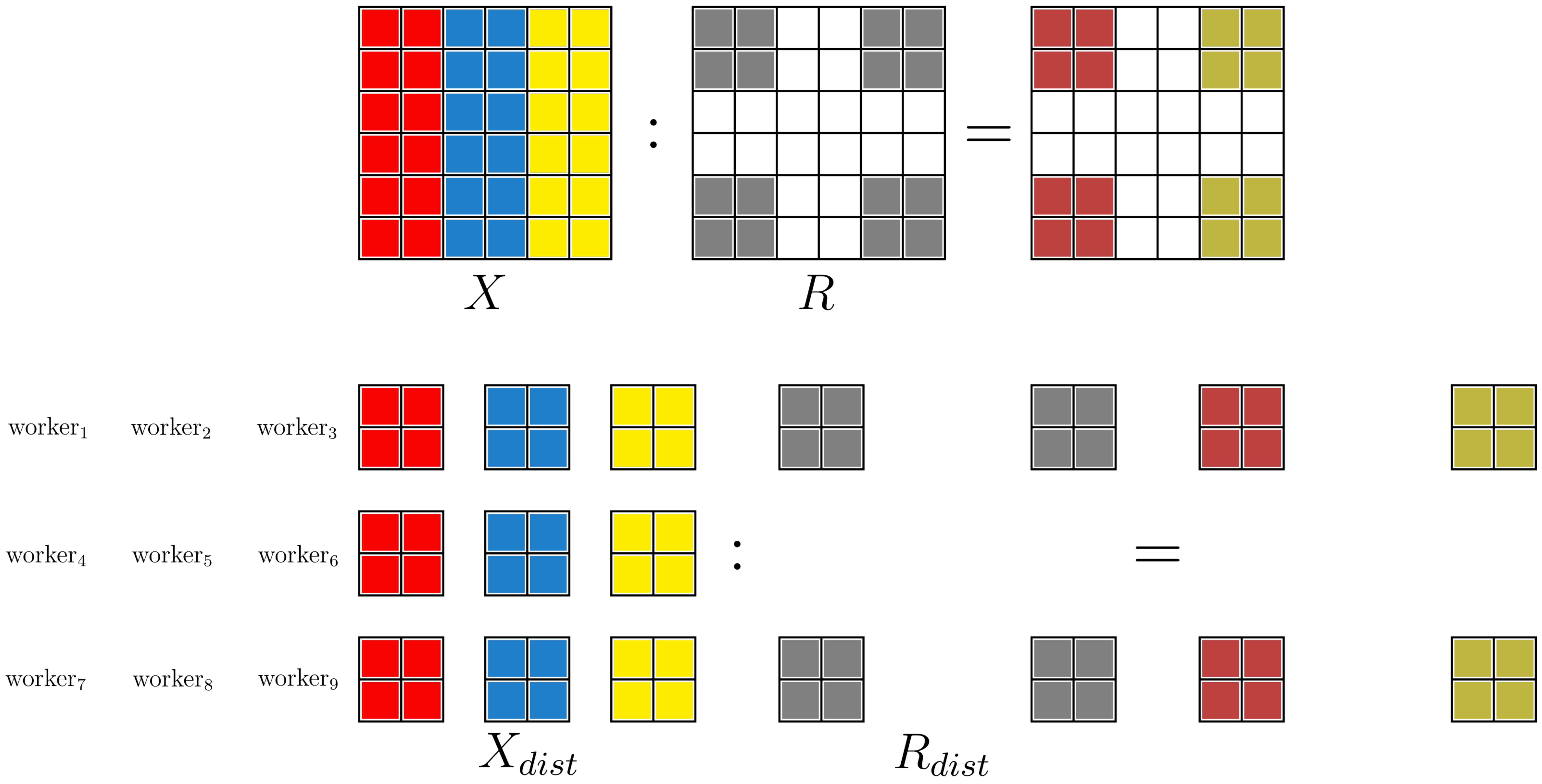


Results
gradient timing experiment



Results

gradient timing experiment - speedup explanation



Conclusions

Distributed-memory parallelism of FNOs is difficult

- high-dimensional data/network
- large memory consumption
- complex network components

Using HPC-oriented deep learning tools (i.e. DistDL) solves the problem

- good abstraction of data movement in HPC systems
- integration with PyTorch allows concise expression & differentiation

FNOs scale well, due to structure of network

https://swimburger.net/media/ppnn3pcl/azure.png
https://www.hdfgroup.org/wp-content/uploads/2017/11/stackedlogo-RGB.jpg

SLIM

Future Work

Scaling:

- fully train network on realistic volume sizes
- remove communication bottlenecks (e.g. GPU offload)

Cloud integration:

- full data pipeline (e.g. Azure Blob w/ HSDS, CycleCloud)
- packaging & deployment of pre-trained models

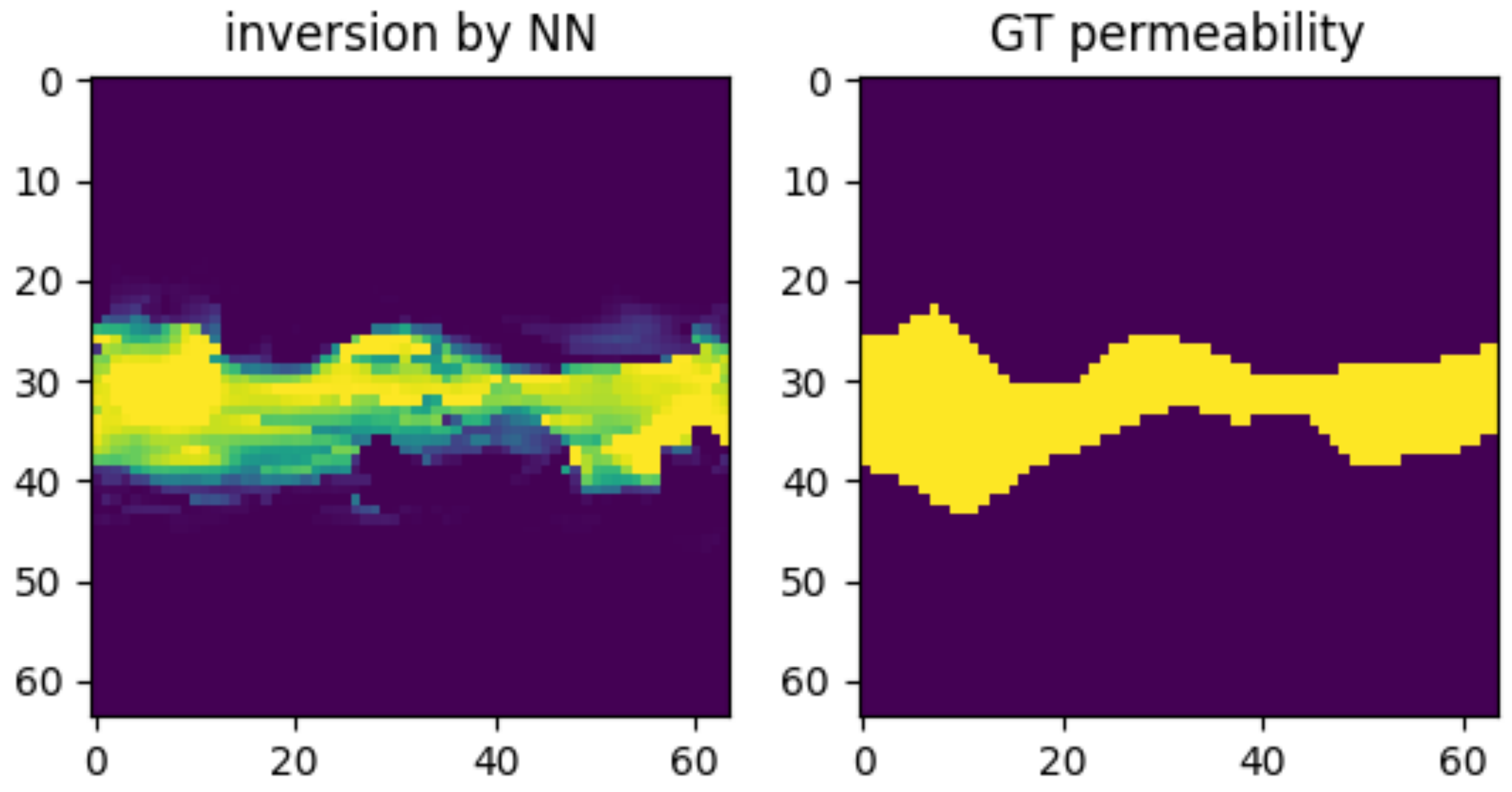
+

The HDF Group

Future Work

CCS Integration

- wave-based monitoring of CCS
- inversion for permeability
- uncertainty quantification



Related Work

DFNO Implementation - https://github.com/slimgroup/dfno

Original FNO - https://github.com/zongyi-li/fourier_neural_operator

DistDL - https://github.com/distdl/distdl

Acknowledgements

Rishi Khan, Extreme Scale Solutions, & US DOE for HPC resources and development guidance

Phillip Witte & Microsoft Research for training data & cloud resources/support

Georgia Research Alliance & partners of the ML4Seismic Center