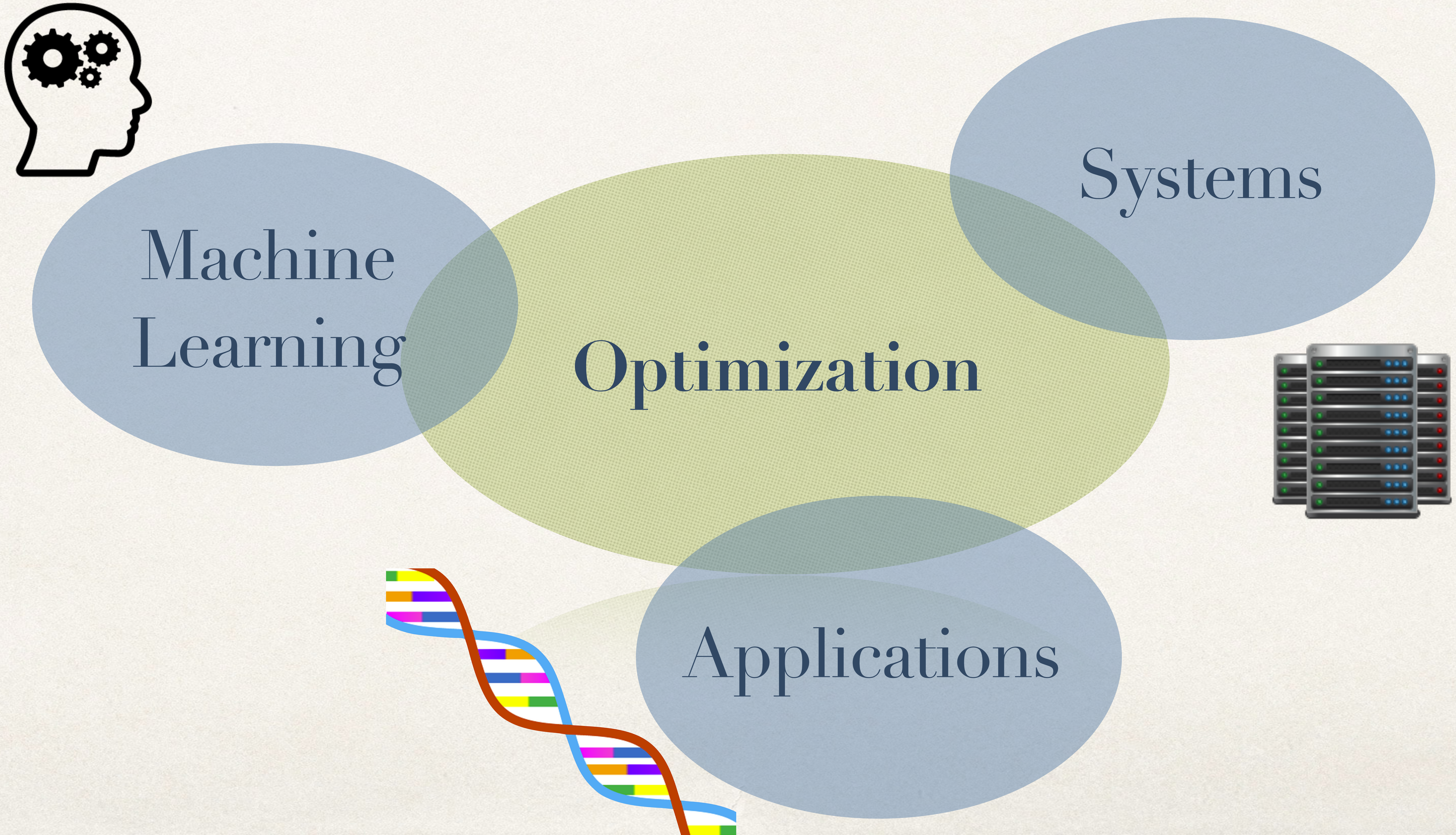
Optimization
for Machine Learning in Practice

Martin Jaggi
EPFL Machine Learning and Optimization Laboratory
mlo.epfl.ch

Where are we?



Trends - General

- **privacy in ML**
- **decentralized training**
- **ML for trust** (e.g. intrusion detection)
- **trust in ML** (provably secure against adversarial attacks)

Adversarial Attacks

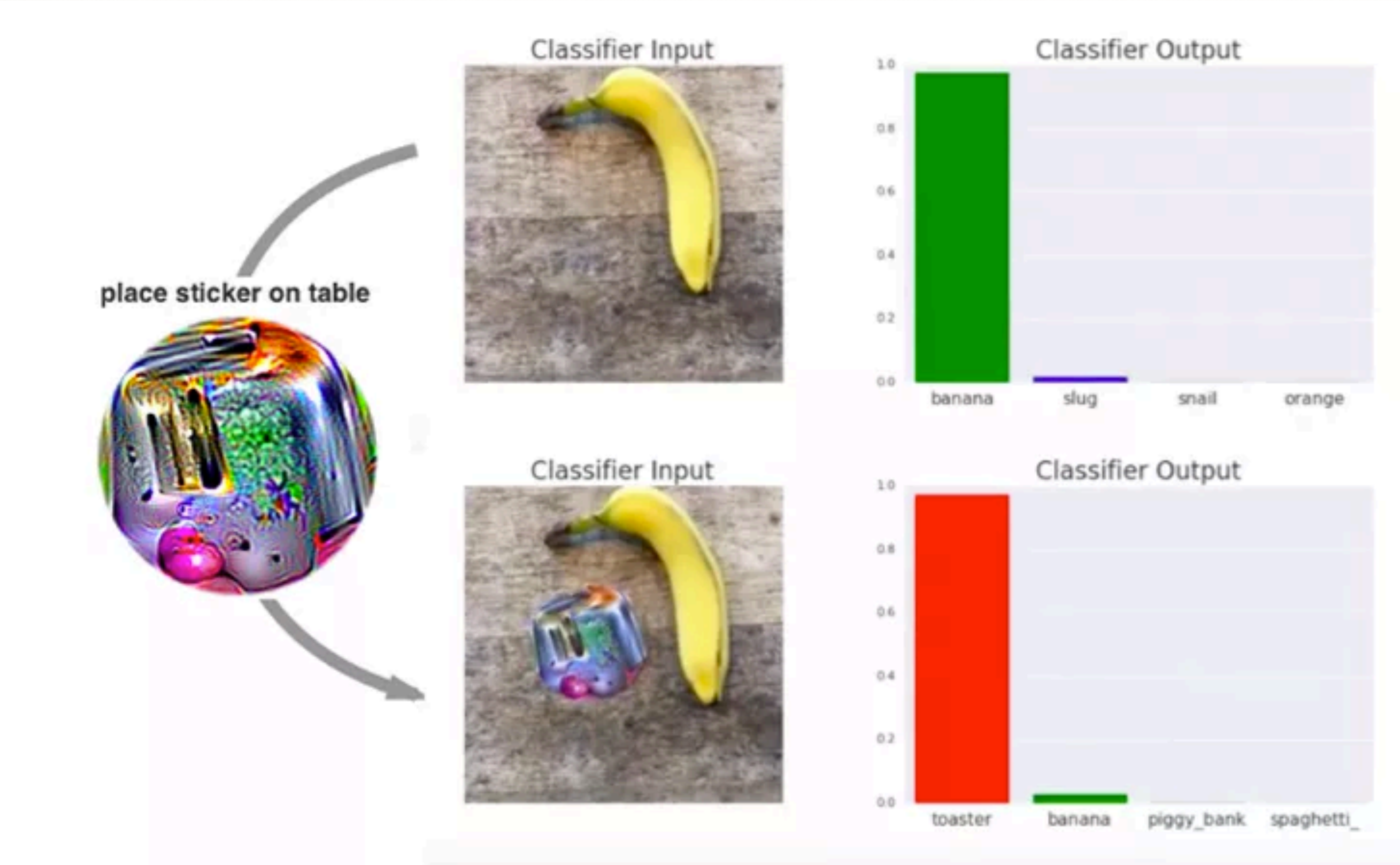


Image: [Tom B. Brown/Dandelion Mané]()

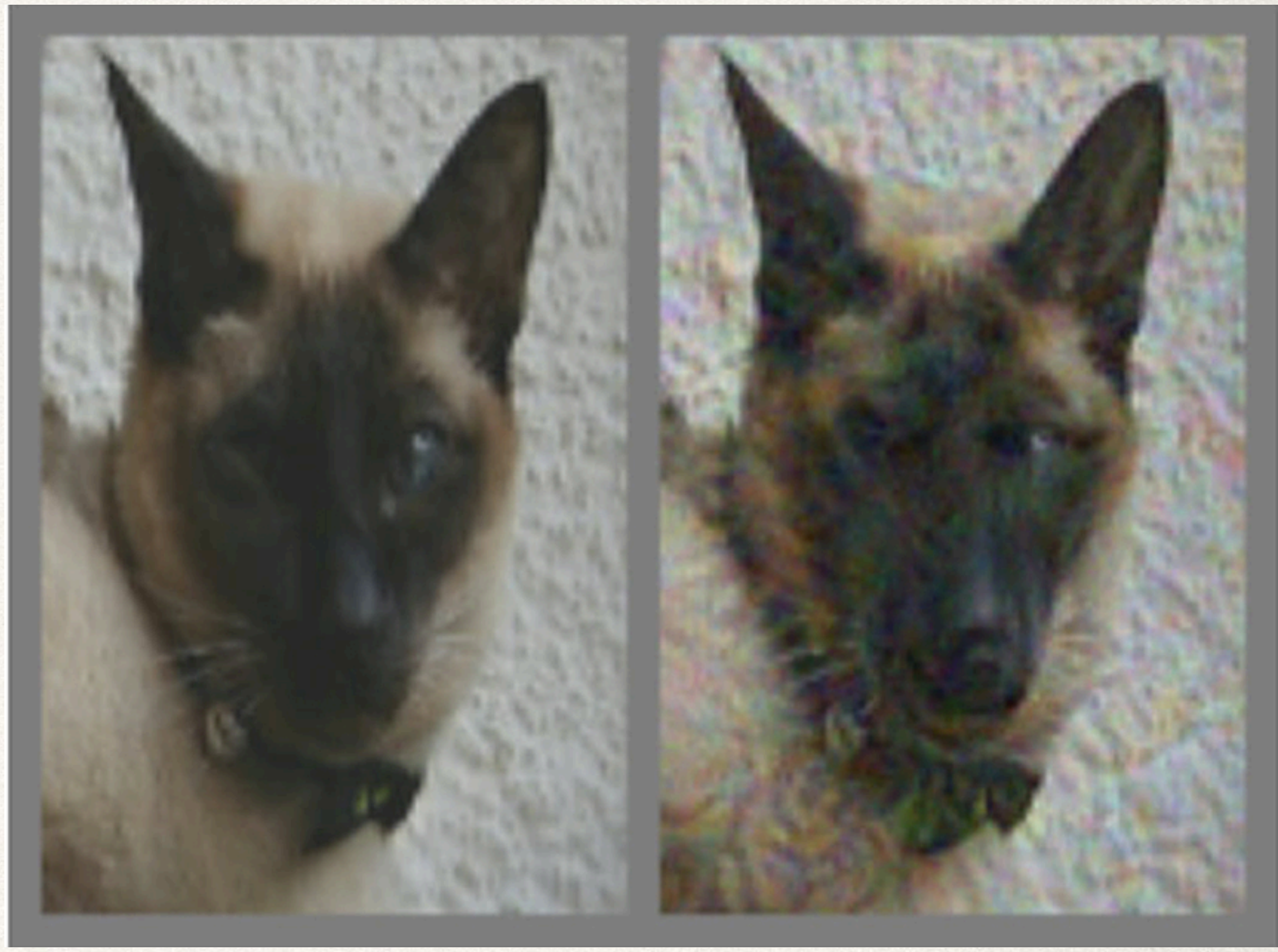


Image: Elsayed ,Papernot et al 2018

Adversarial Attacks

- **training**

$$\min_{\boldsymbol{w}} \left(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i\right)^2$$

$\nabla_{\boldsymbol{w}} f$

change model

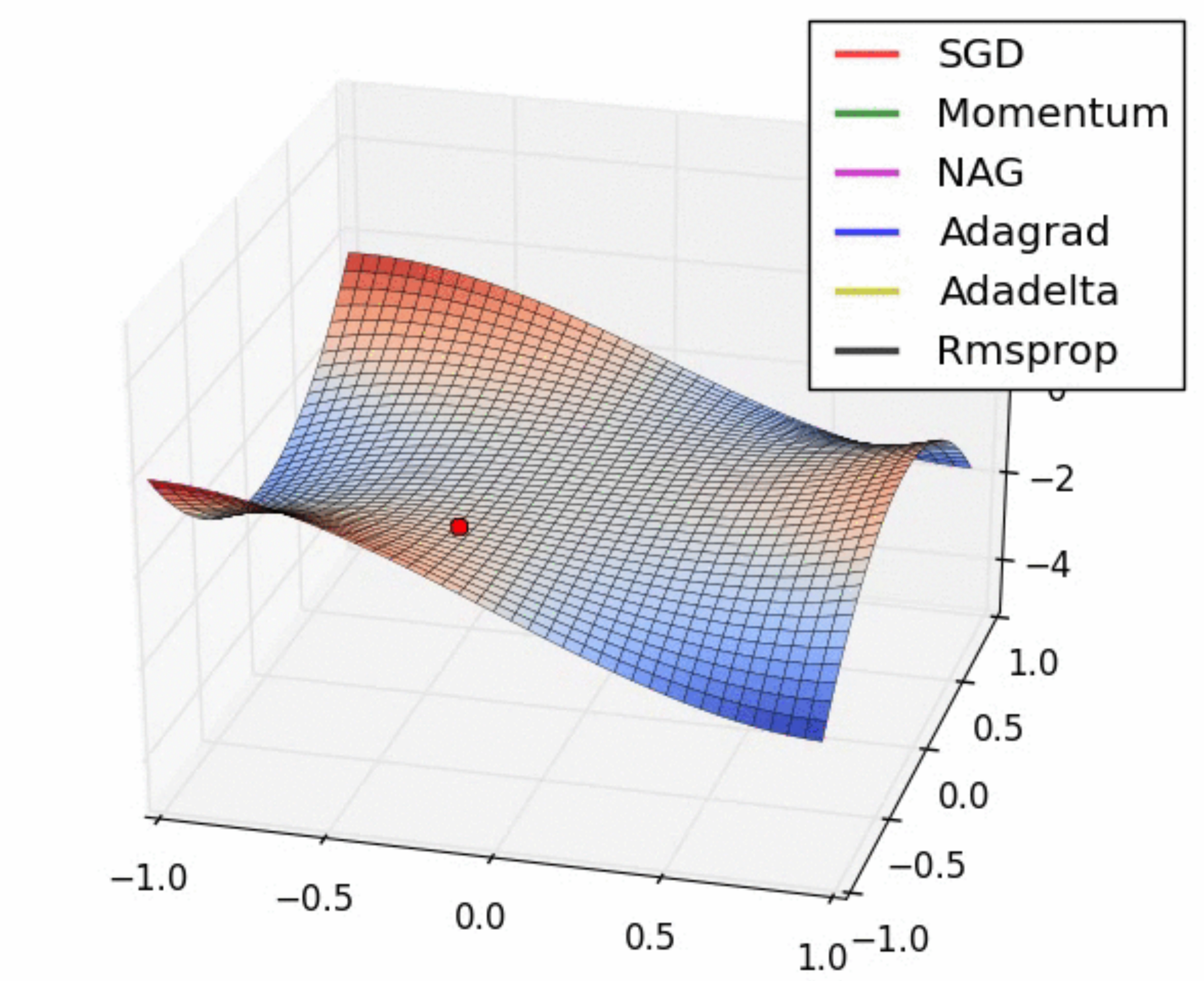
- **attacking**

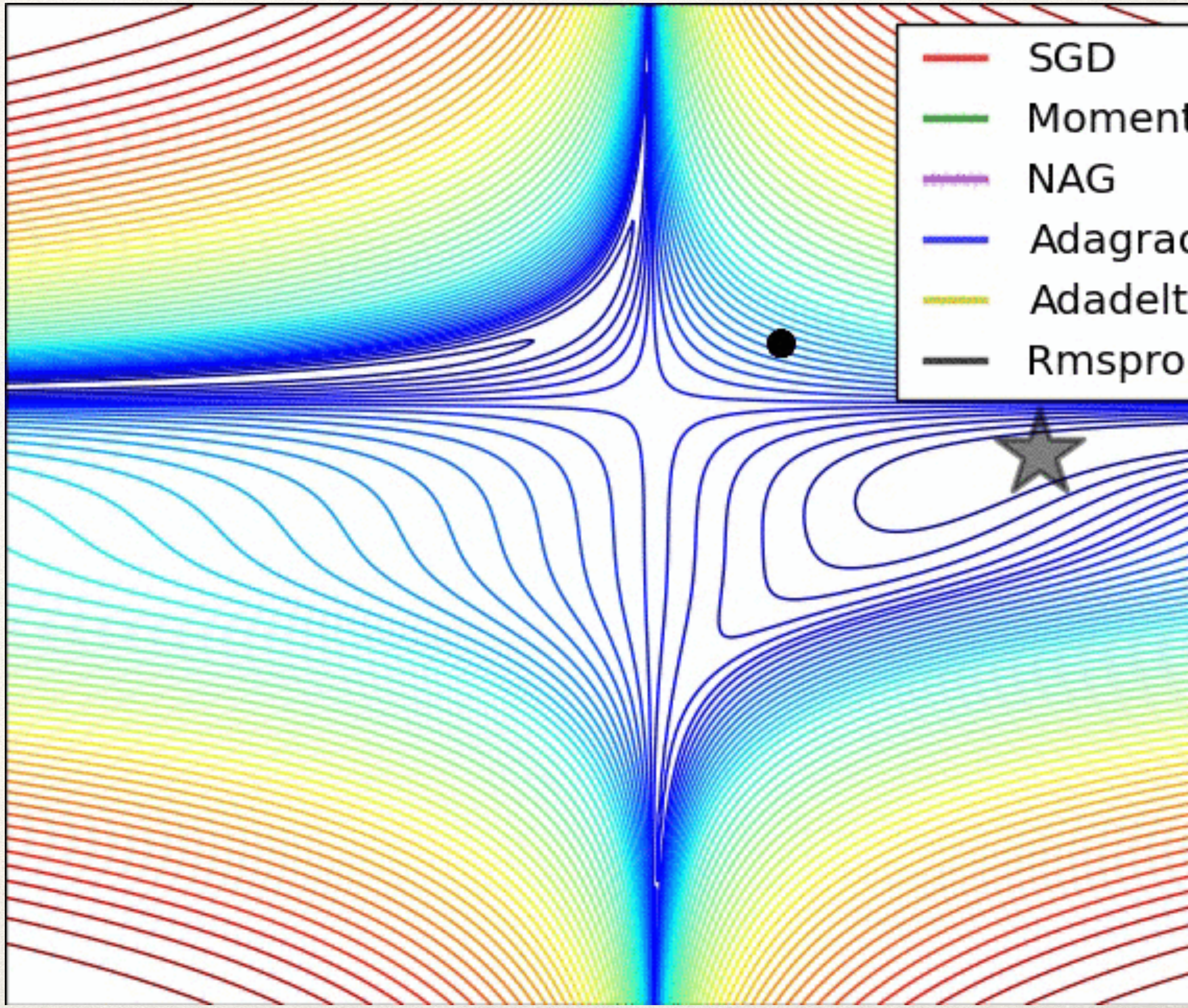
$$\min_{\boldsymbol{x}_i} \left(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i\right)^2$$

$\nabla_{\boldsymbol{x}_i} f$

change data

Practical comparison of algorithms

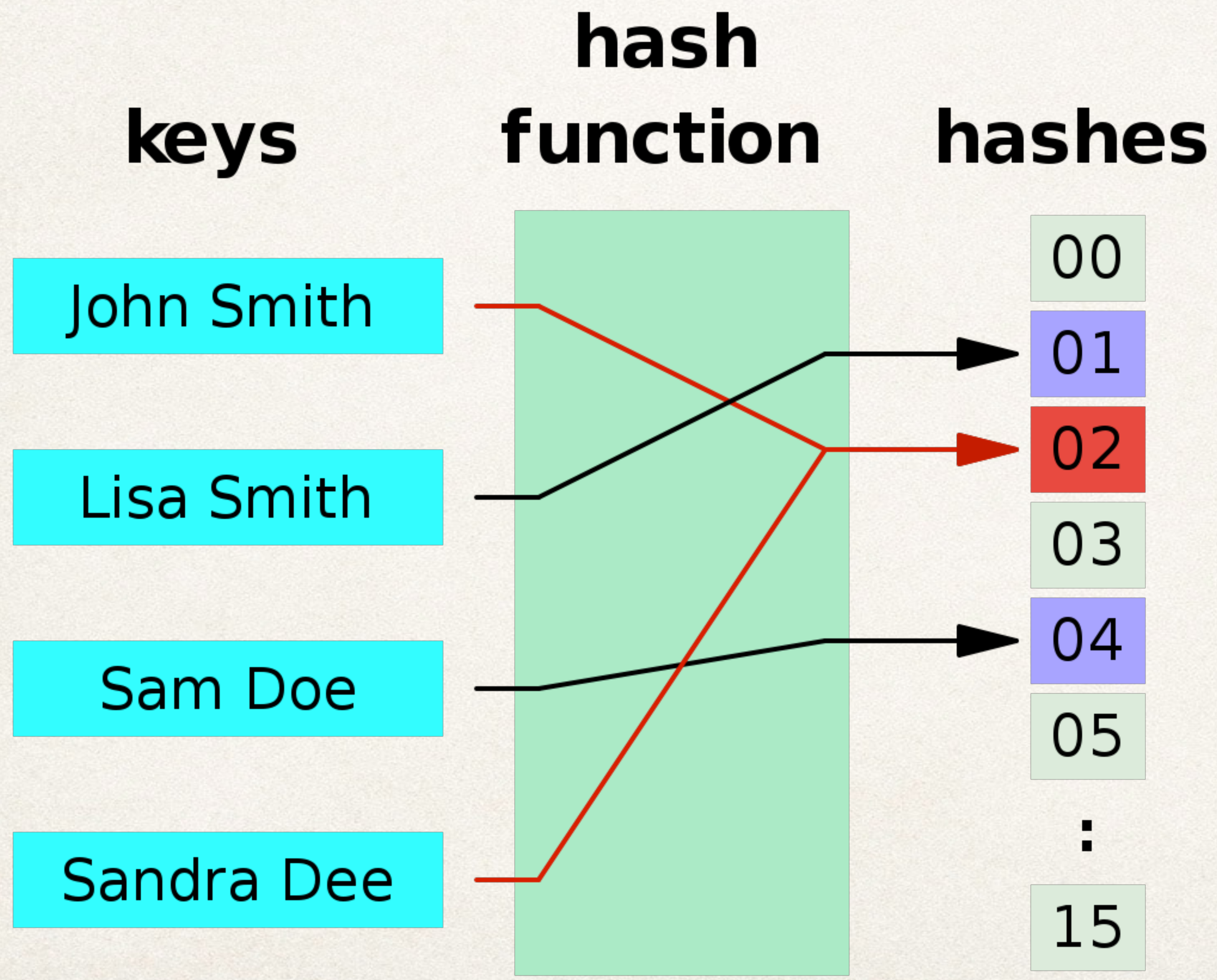




https://imgur.com/a/Hqolp#2dKCQHh

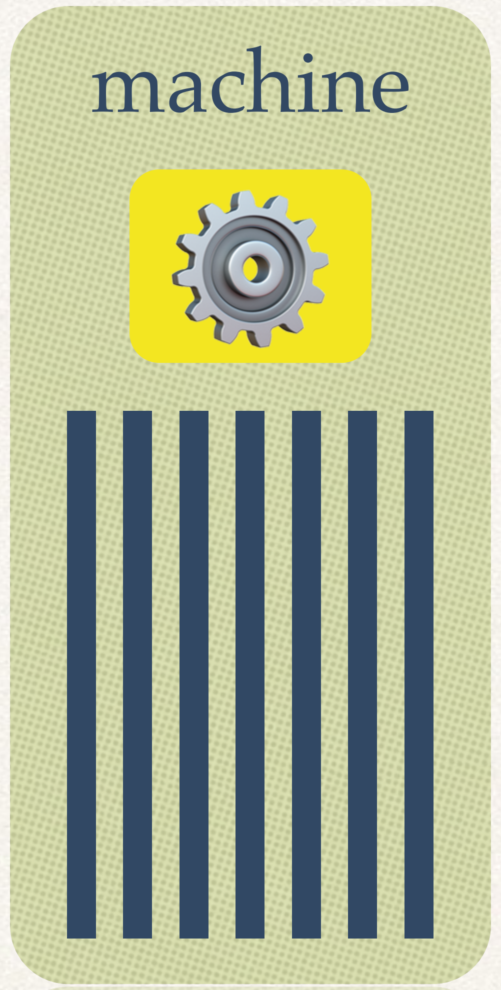
Practical tricks

- **feature hashing**

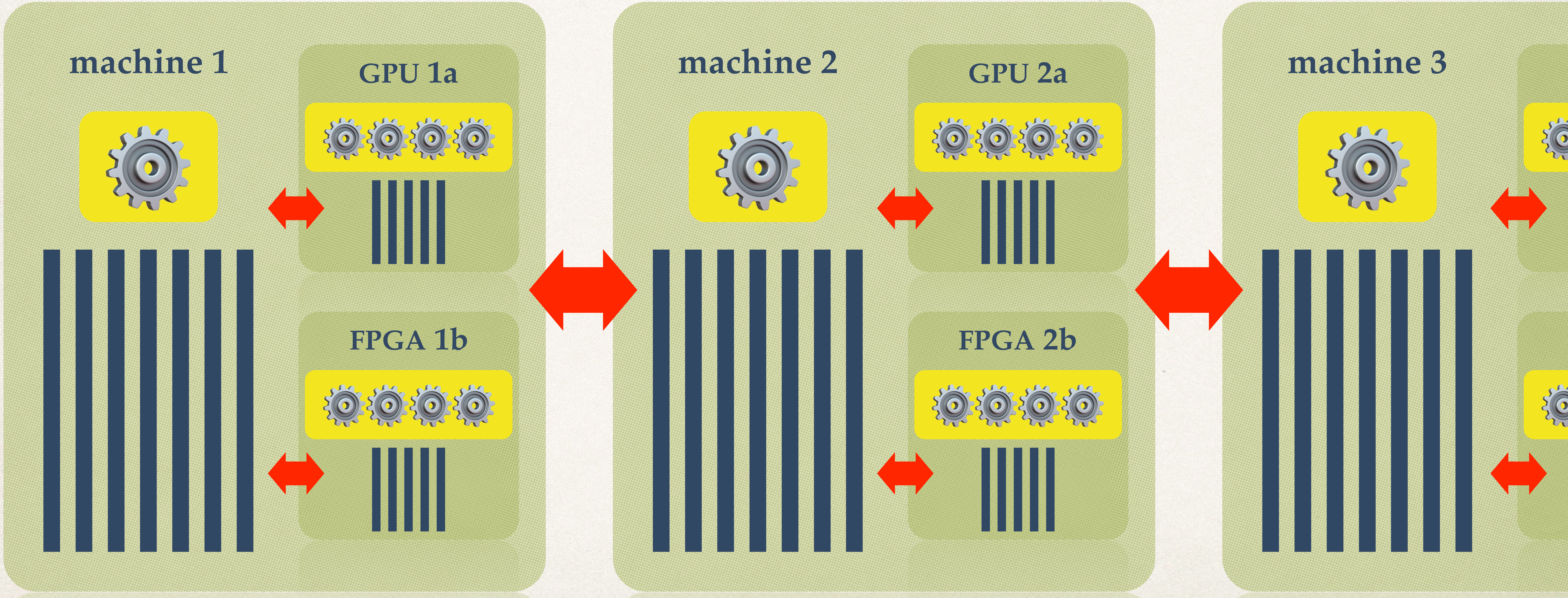


- **limited precision operations**

Systems …then

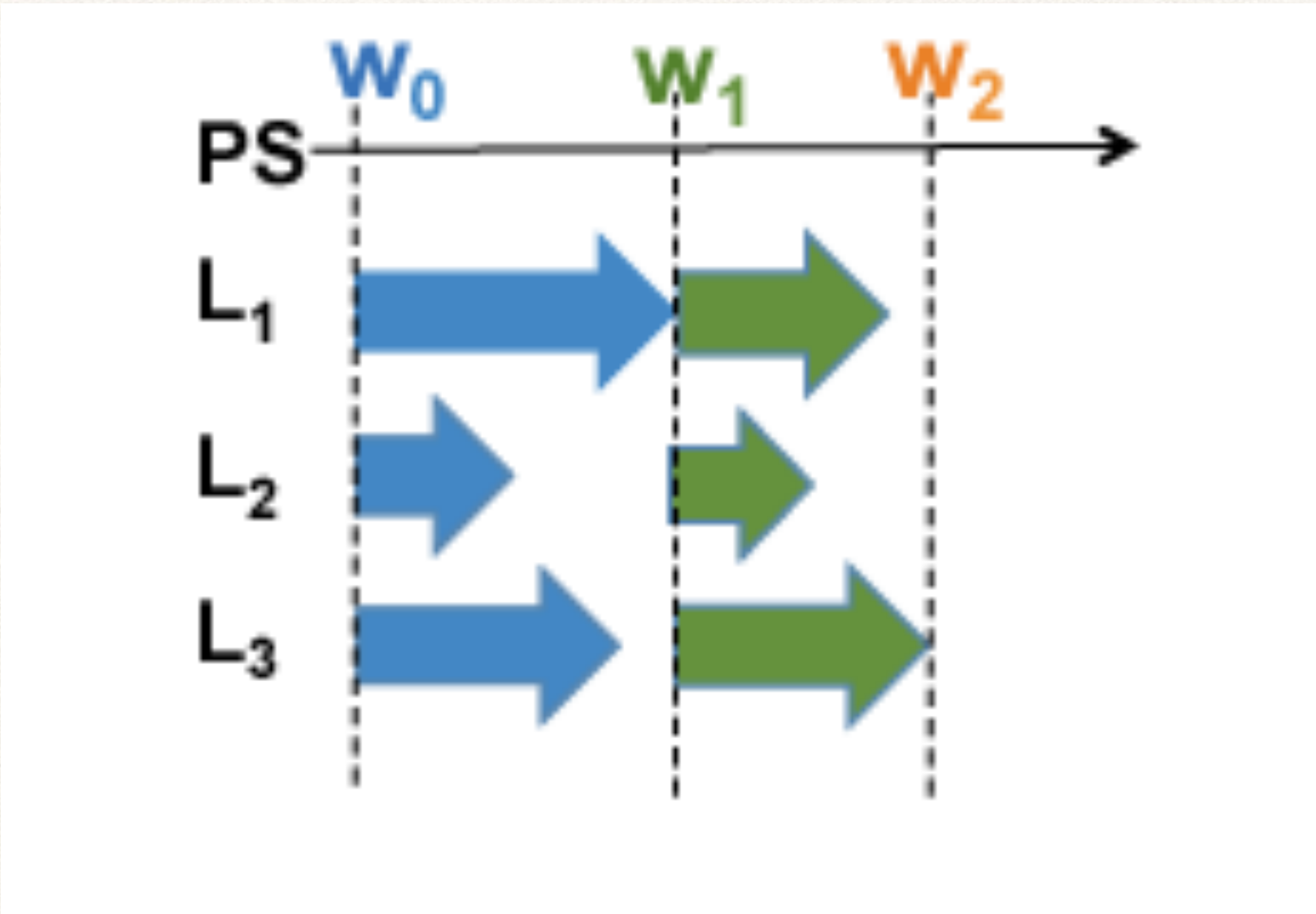


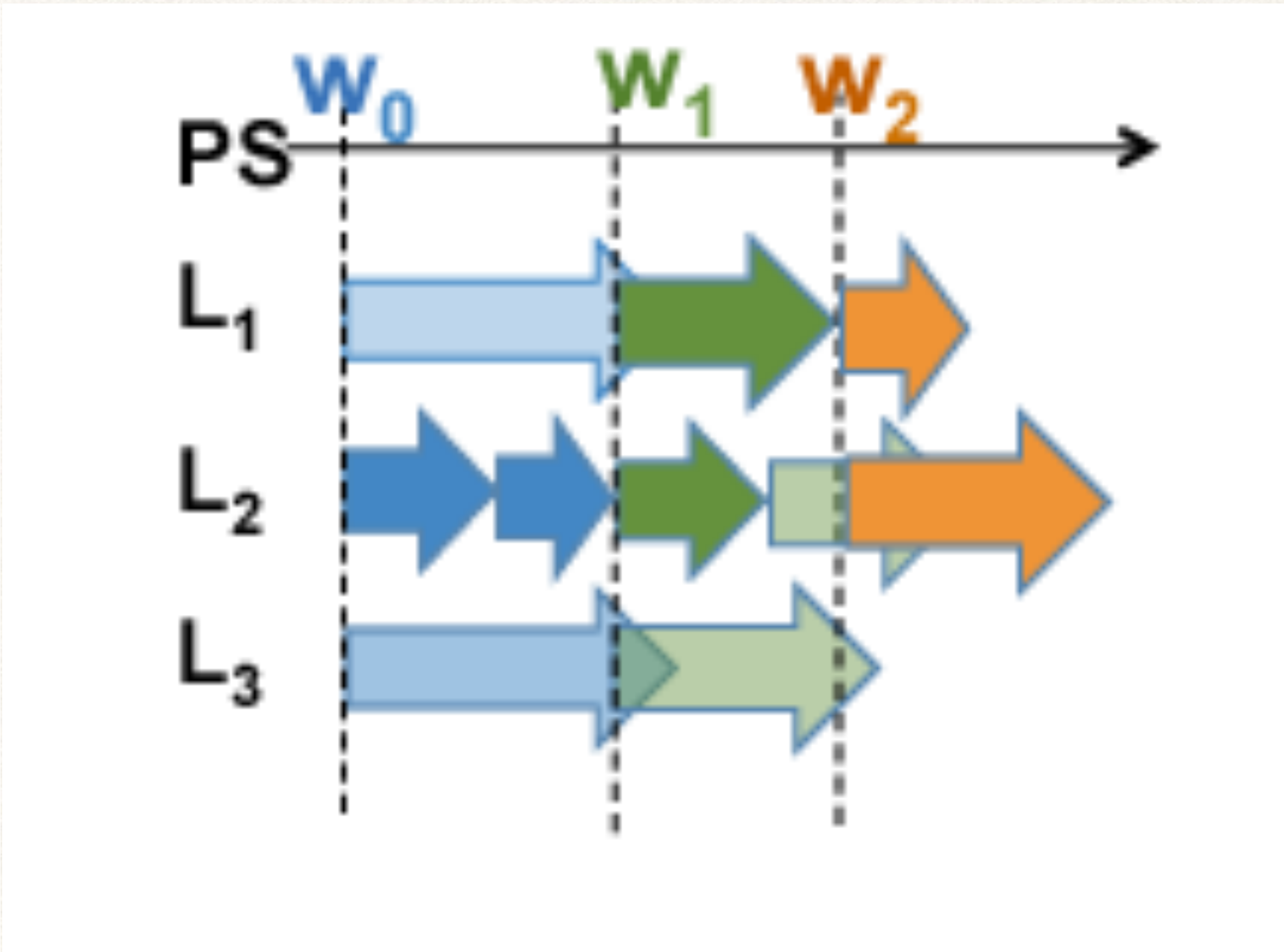
Systems …now



Parallel SGD

- **Synchronous**
- **Asynchronous**



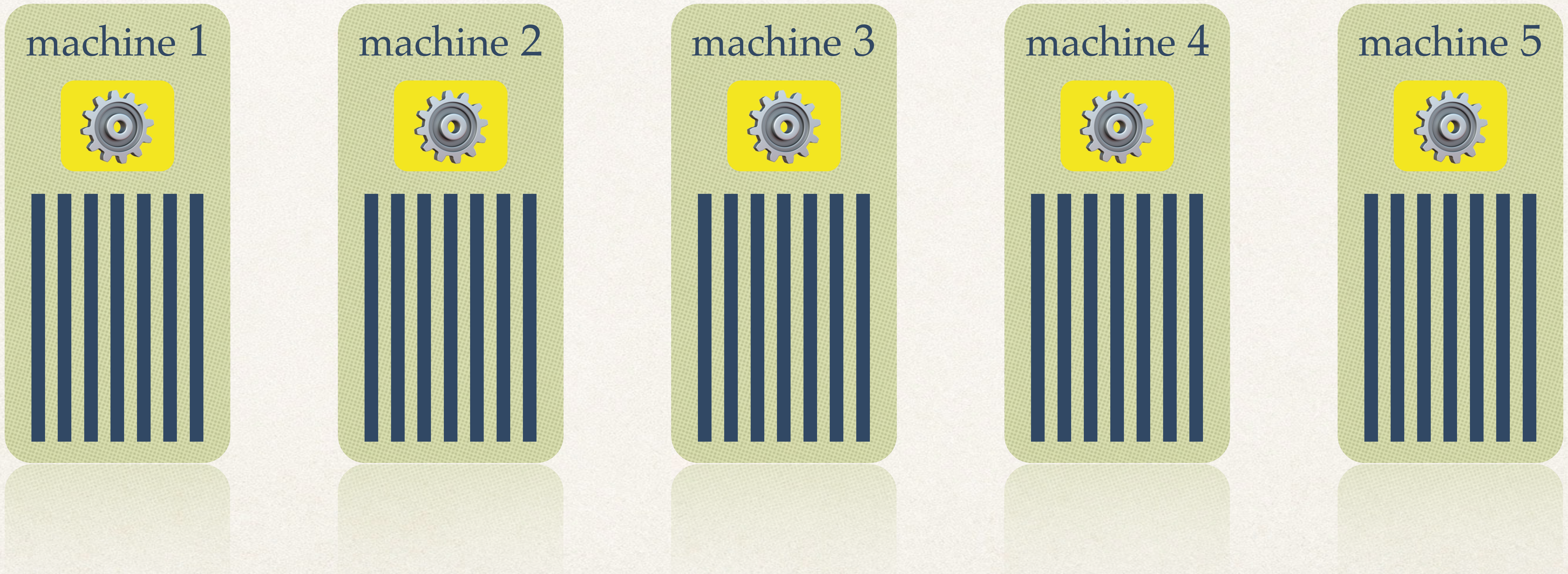


Mini-Batch!

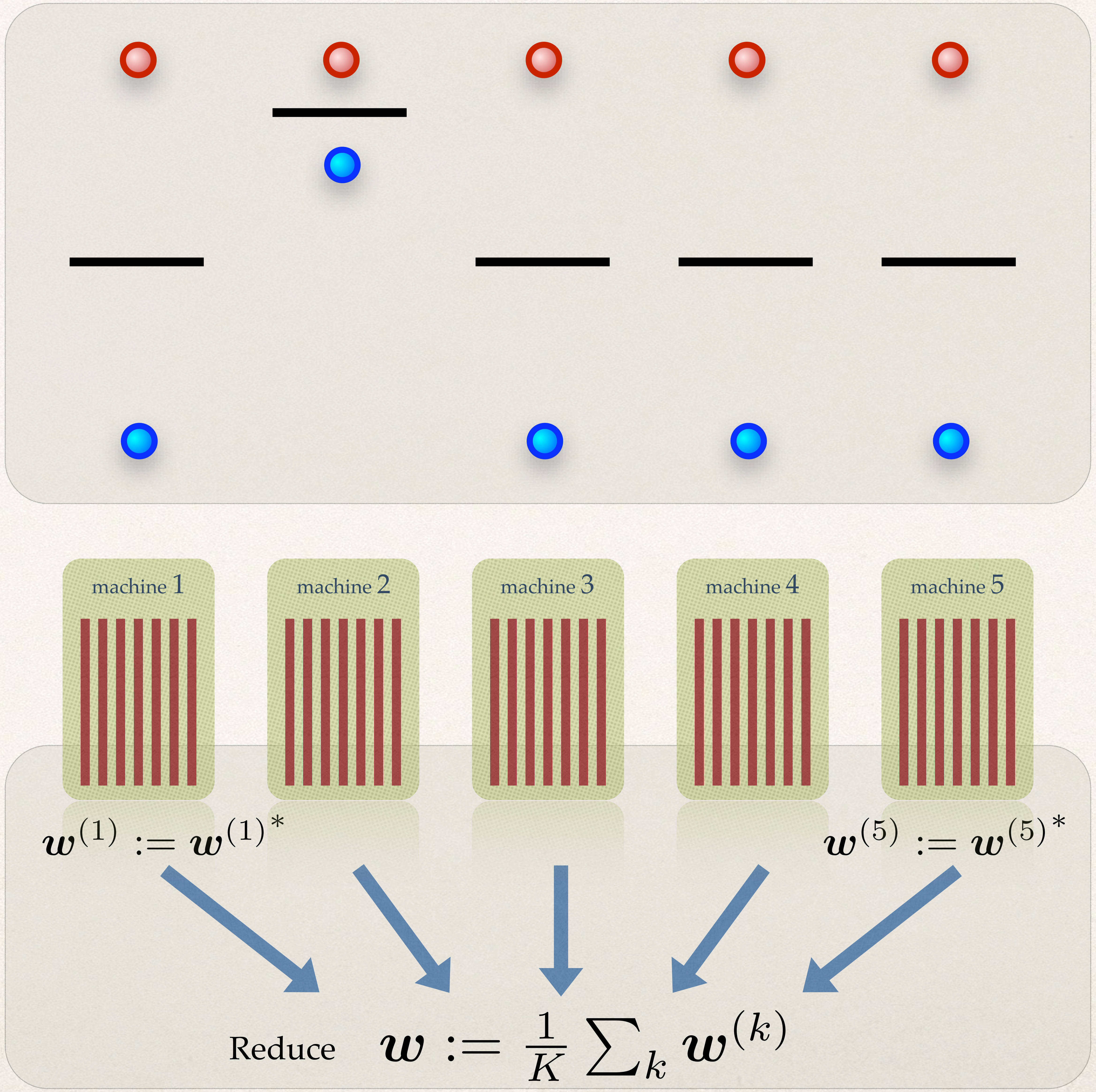
1

Distributed

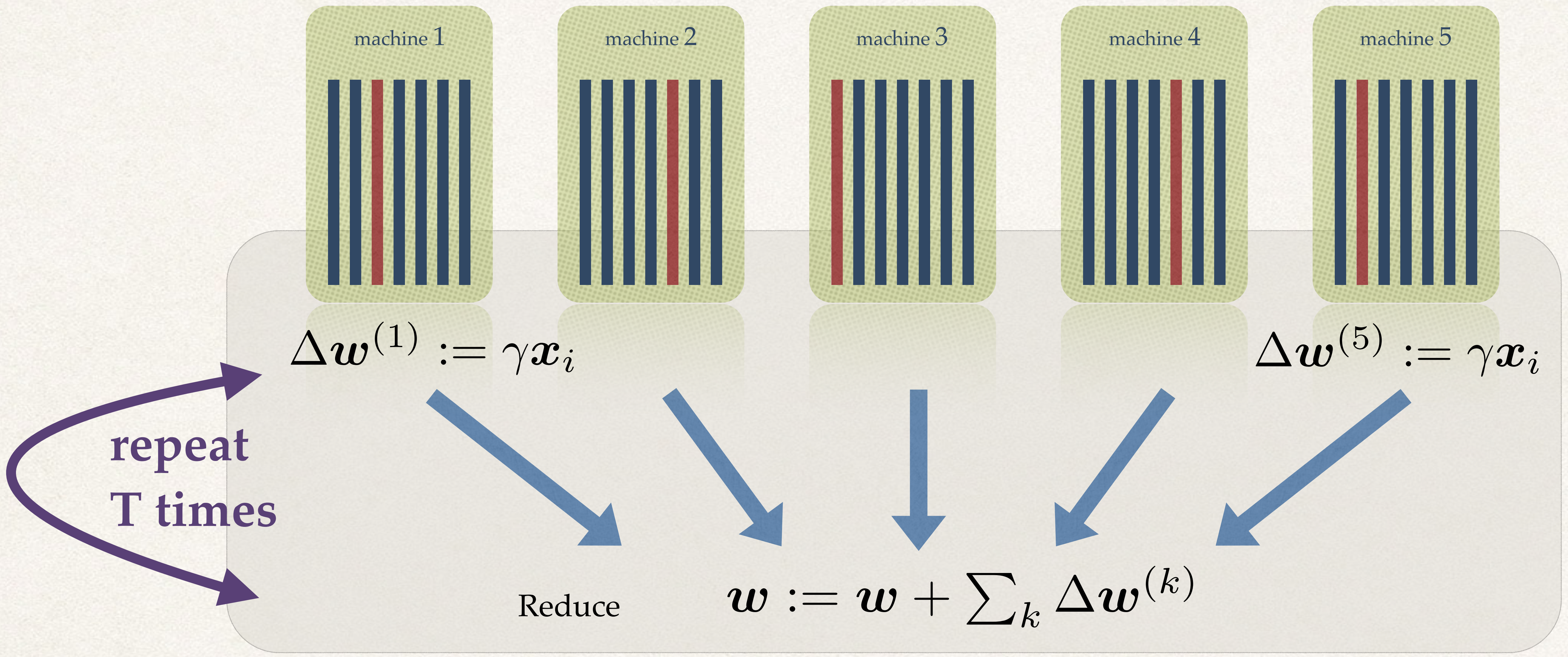
What if the data does not fit onto one device anymore?



One-Shot Averaging Does Not Work



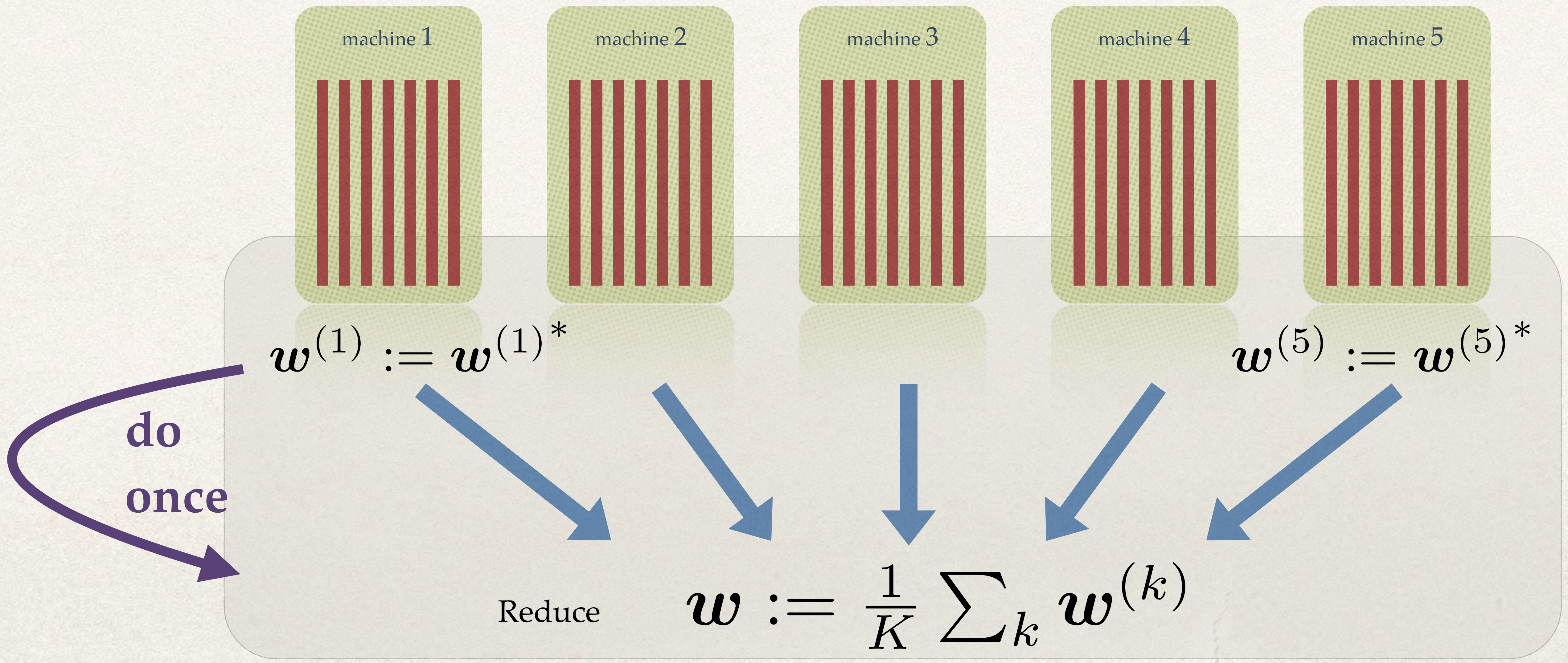
Communication: Always / Never



Naive Distributed SGD

 # local datapoints read: T
 # communications: T
convergence: ✓

"always communicate"



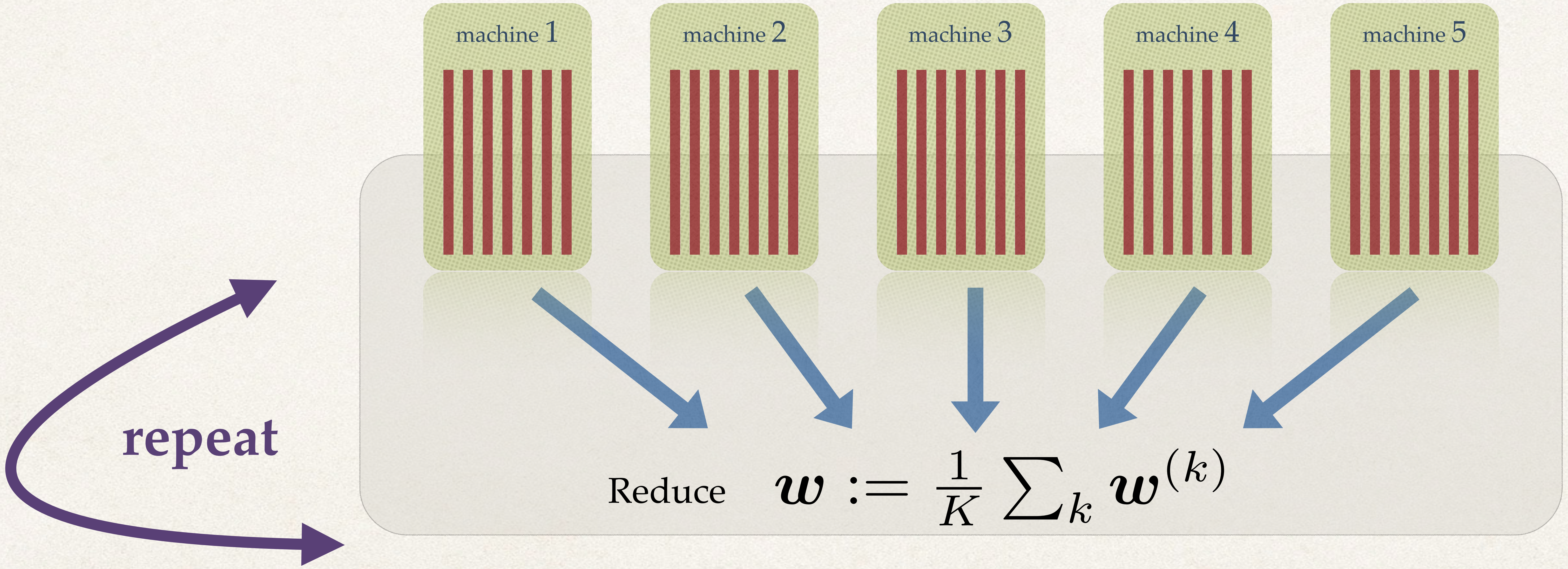
One-Shot Averaged Distributed Optimization

 # local datapoints read: T
 # communications: 1
convergence: ✗

"never communicate"

Distributed Full Gradient, L-BFGS

(just distribute the full gradient computation)



Problem class

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad f(A\boldsymbol{\alpha}) \; + \; g(\boldsymbol{\alpha})$$

Optimization: Primal-Dual Context

$$A_{\text{loc}} \Delta \boldsymbol{\alpha}_{[k]} + \boldsymbol{w}$$

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left[\mathcal{O}_A(\boldsymbol{\alpha}) := f(A\boldsymbol{\alpha}) + g(\boldsymbol{\alpha}) \right]$$

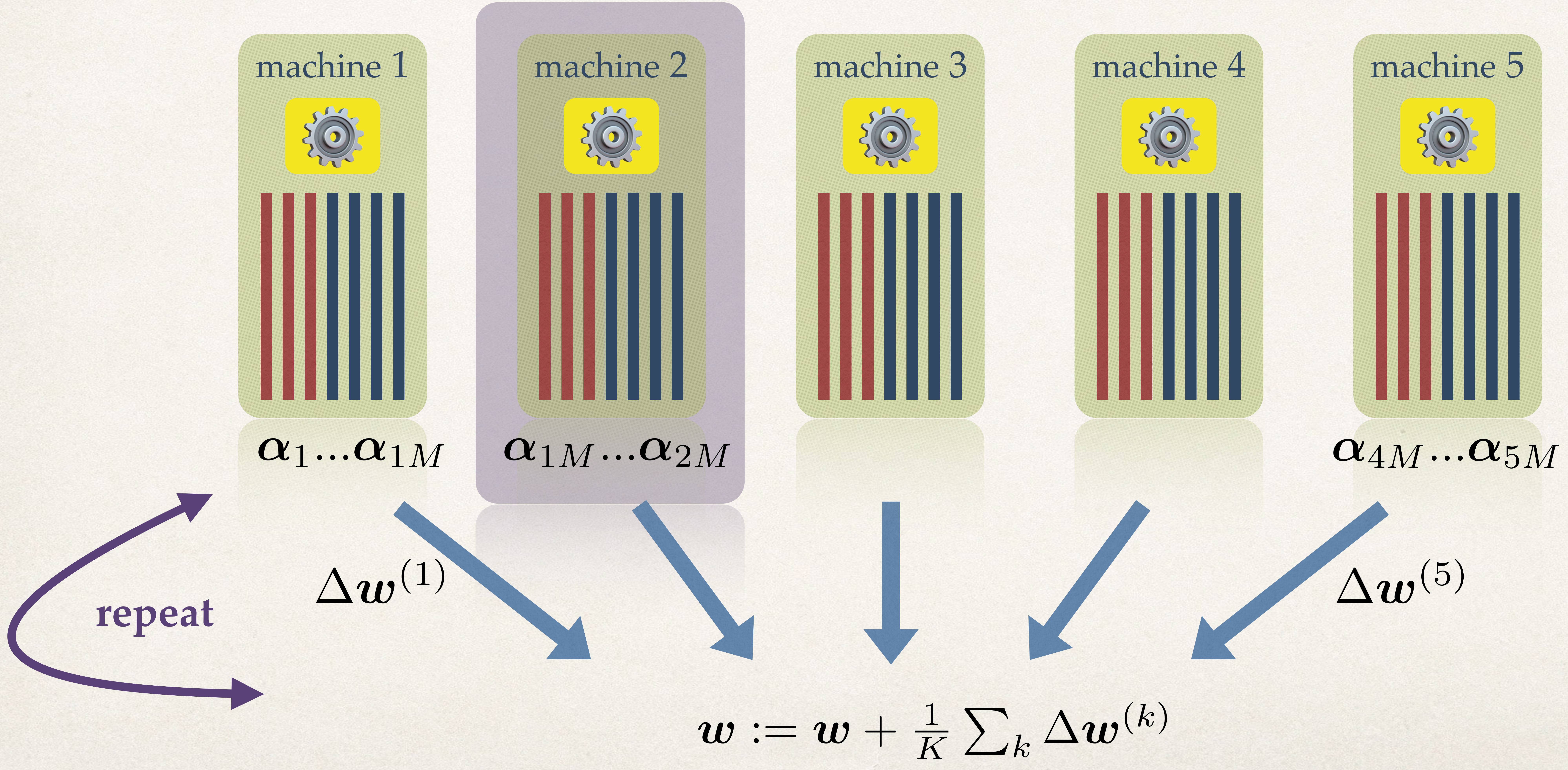
primal Lasso
dual L2-reg SVM/Log-Regr
primal L1-reg SVM/Log-Reg

correspondence

$$\boldsymbol{w} := \nabla f(A\boldsymbol{\alpha})$$

$$\min_{\mathbf{w} \in \mathbb{R}^d} \left[\mathcal{O}_B(\mathbf{w}) := g^*(-A^\top \mathbf{w}) + f^*(\mathbf{w}) \right]$$

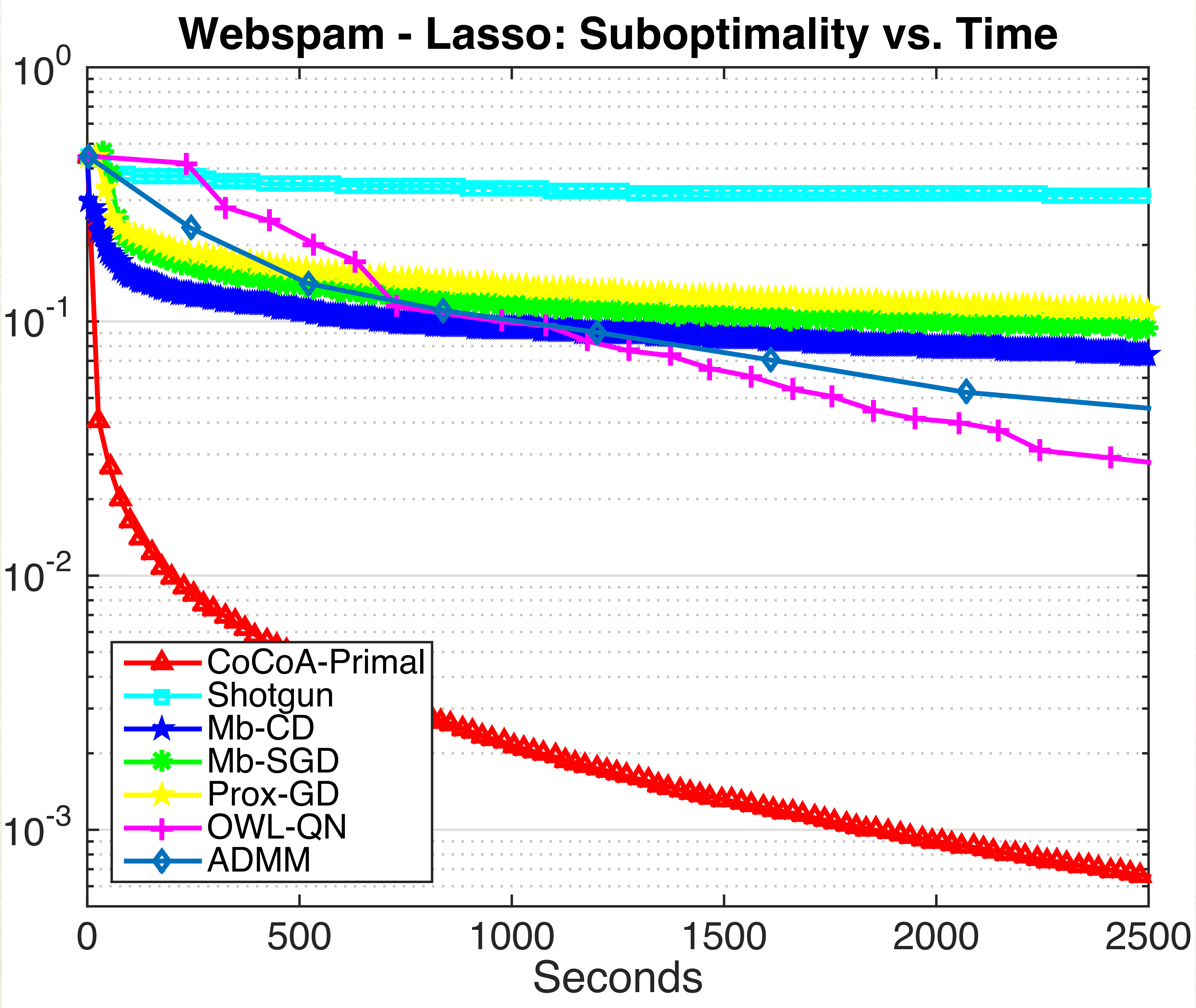
CoCoA - Communication Efficient Distributed Optimization



Distributed Experiments

Dataset	Training	Features	Sparsity
url	2,396,130	3,231,961	3.5e-3%
epsilon	400,000	2,000	100%
kddb	19,264,097	29,890,095	9.8e-5%
webspam	350,000	16,609,143	0.02%

L1-Regularized Linear Regression



NIPS 2014, ICML 2015, JMLR 2018
arxiv.org/abs/1611.02189

- part of TensorFlow core (L2)
- *code in pytorch, TF, spark, C (also L1)*

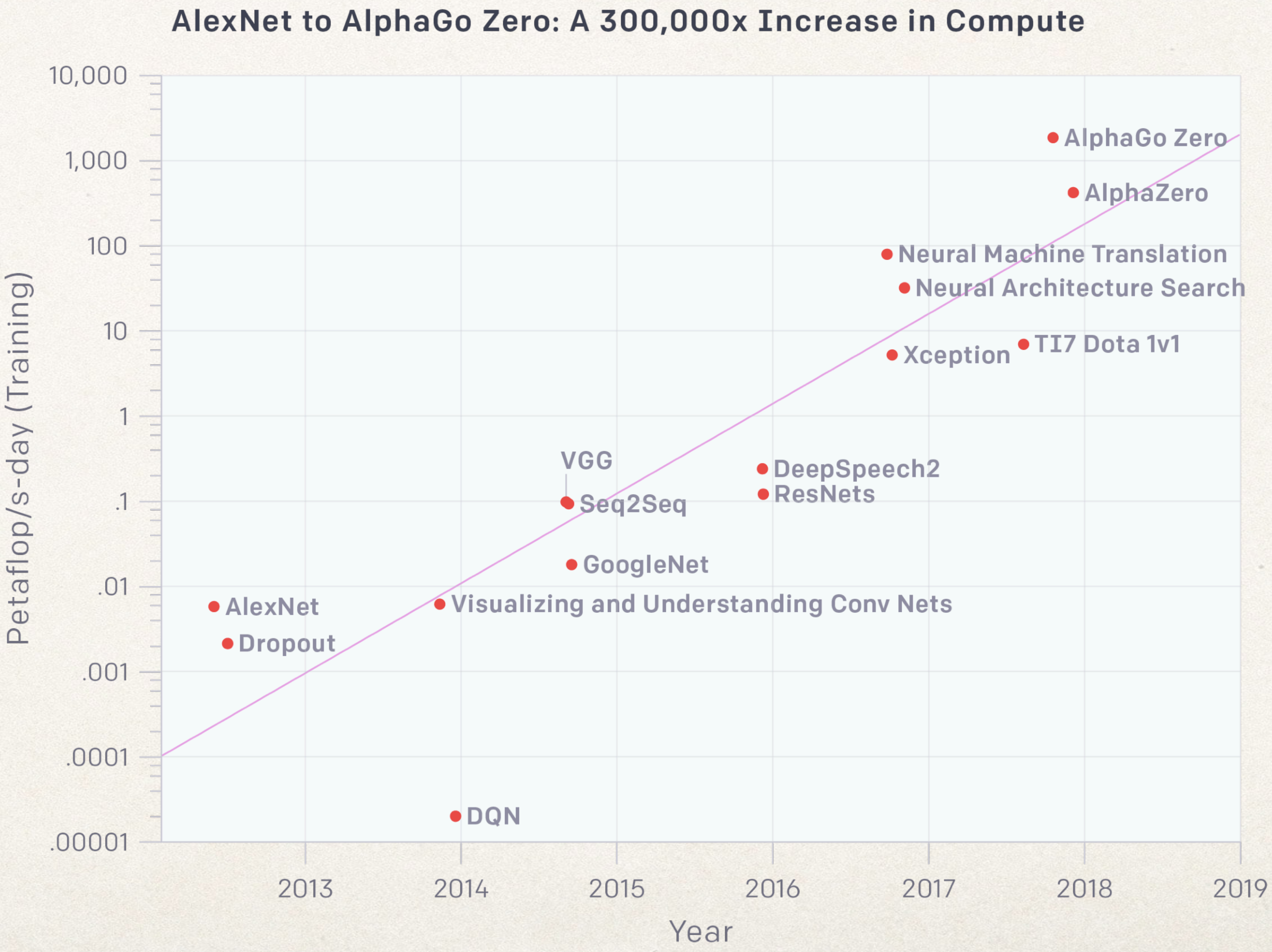
Summary

- **adaptivity** to the communication cost
- **re-usability** of good existing solvers
- **accuracy** certificates
- **second-order** and **trust-region** version (local Hessian)

Next Steps

- **adaptivity** to the degree of separability
- generalization to **deep learning, SGD**
- **decentralized version** (communication on graph)
- **benchmarking** & code

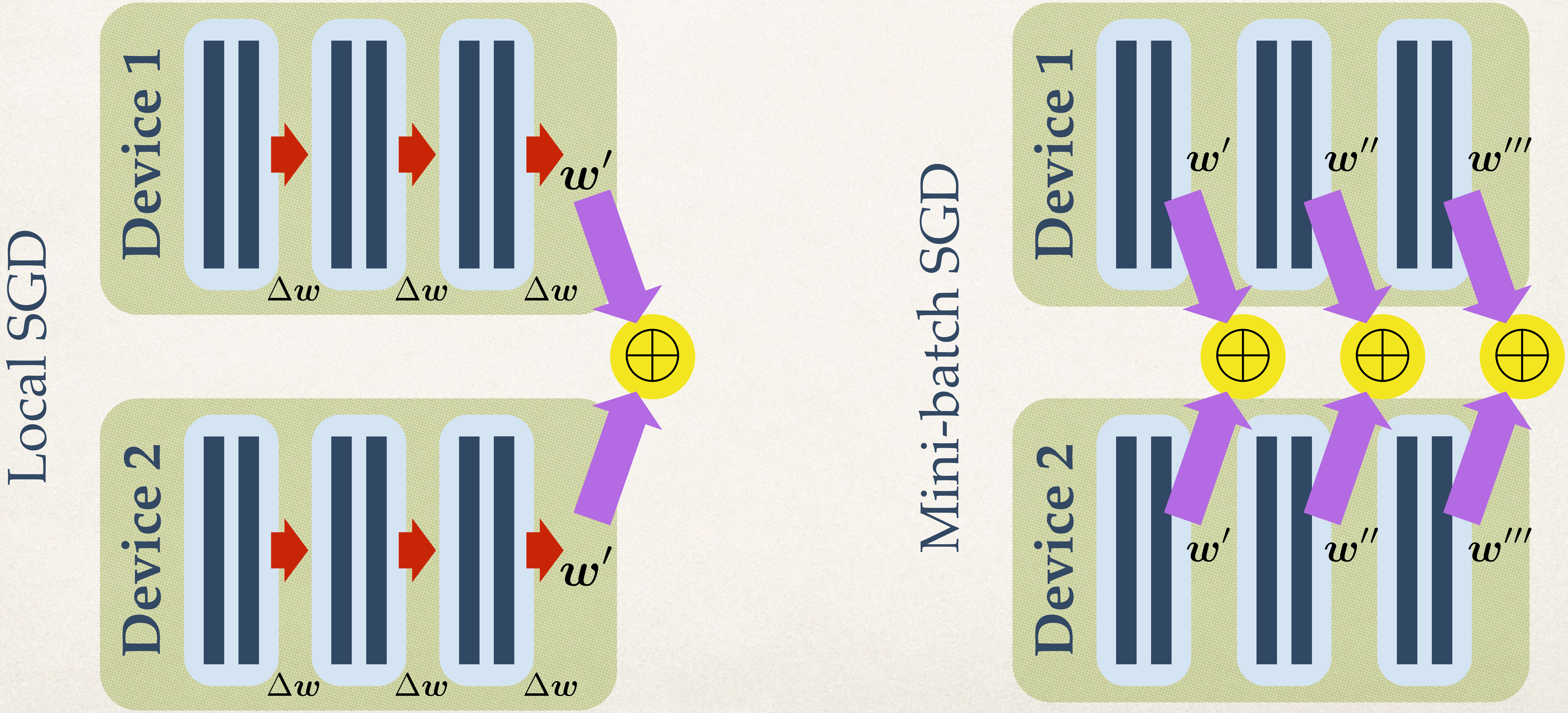
Deep Learning



https://blog.openai.com/ai-and-compute/

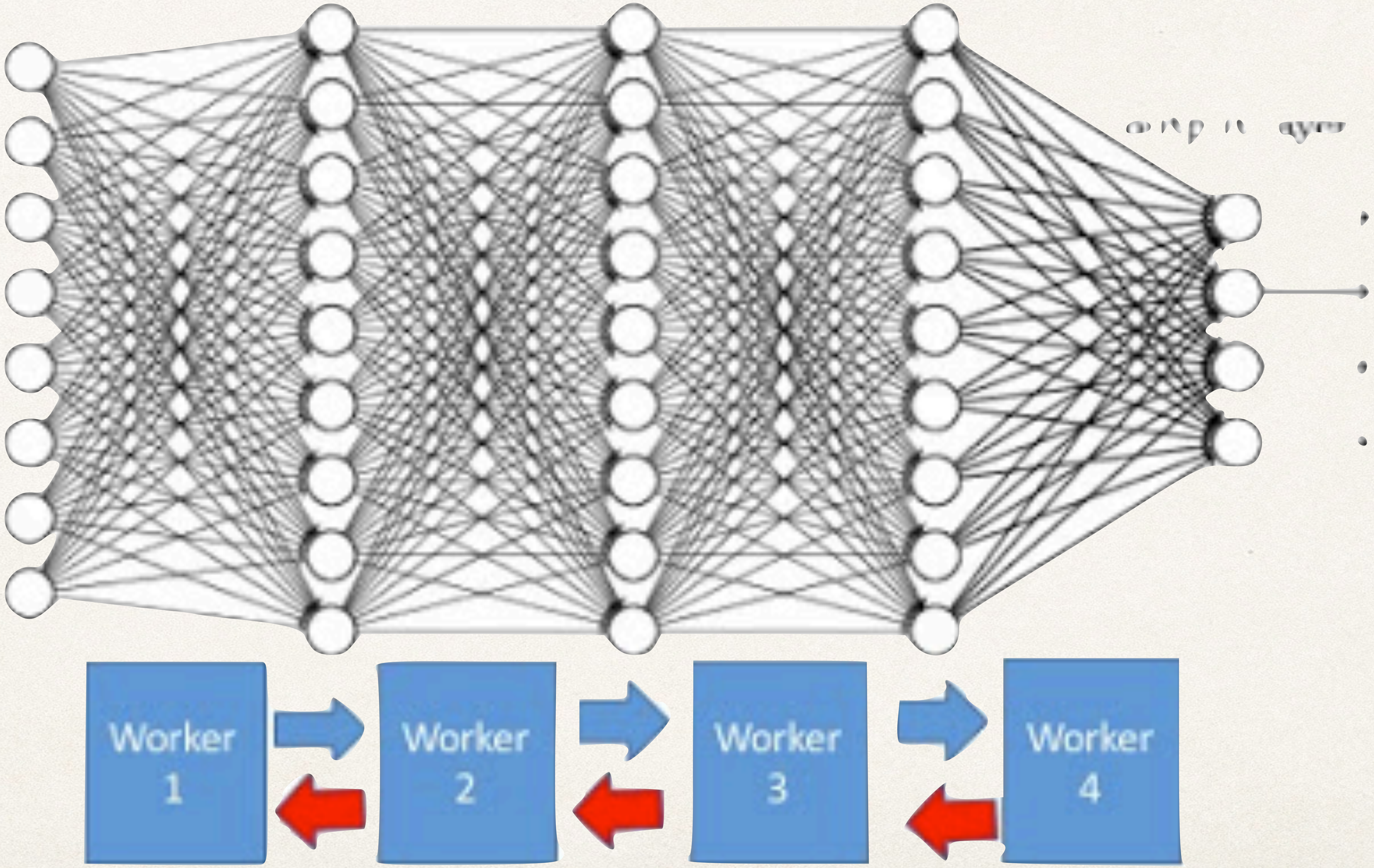
(Data Parallel)

Distributed DL



(Model Parallel)

Distributed DL

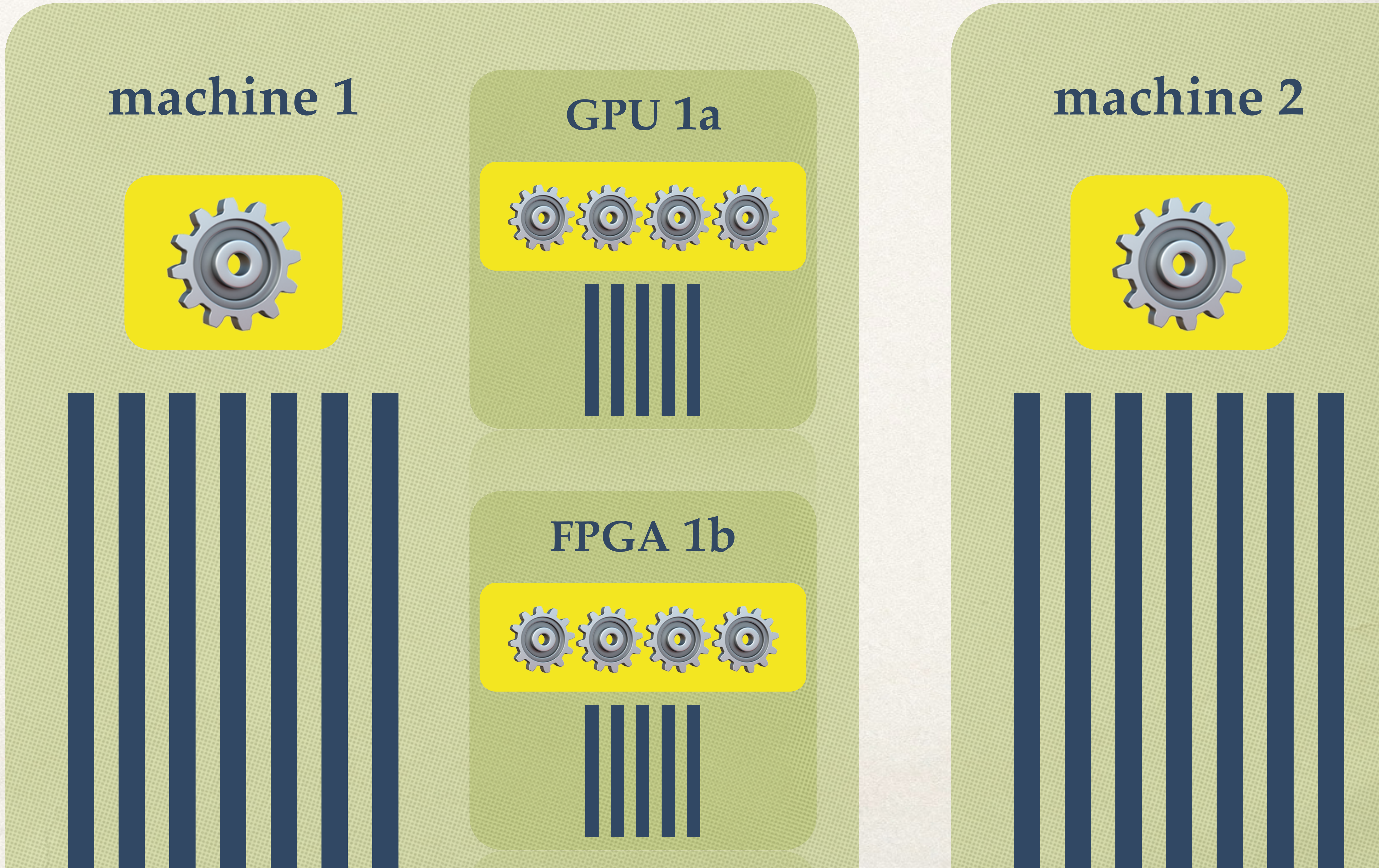


http://kuozhangub.blogspot.com/2017/08/data-parallel-and-model-parallel.html

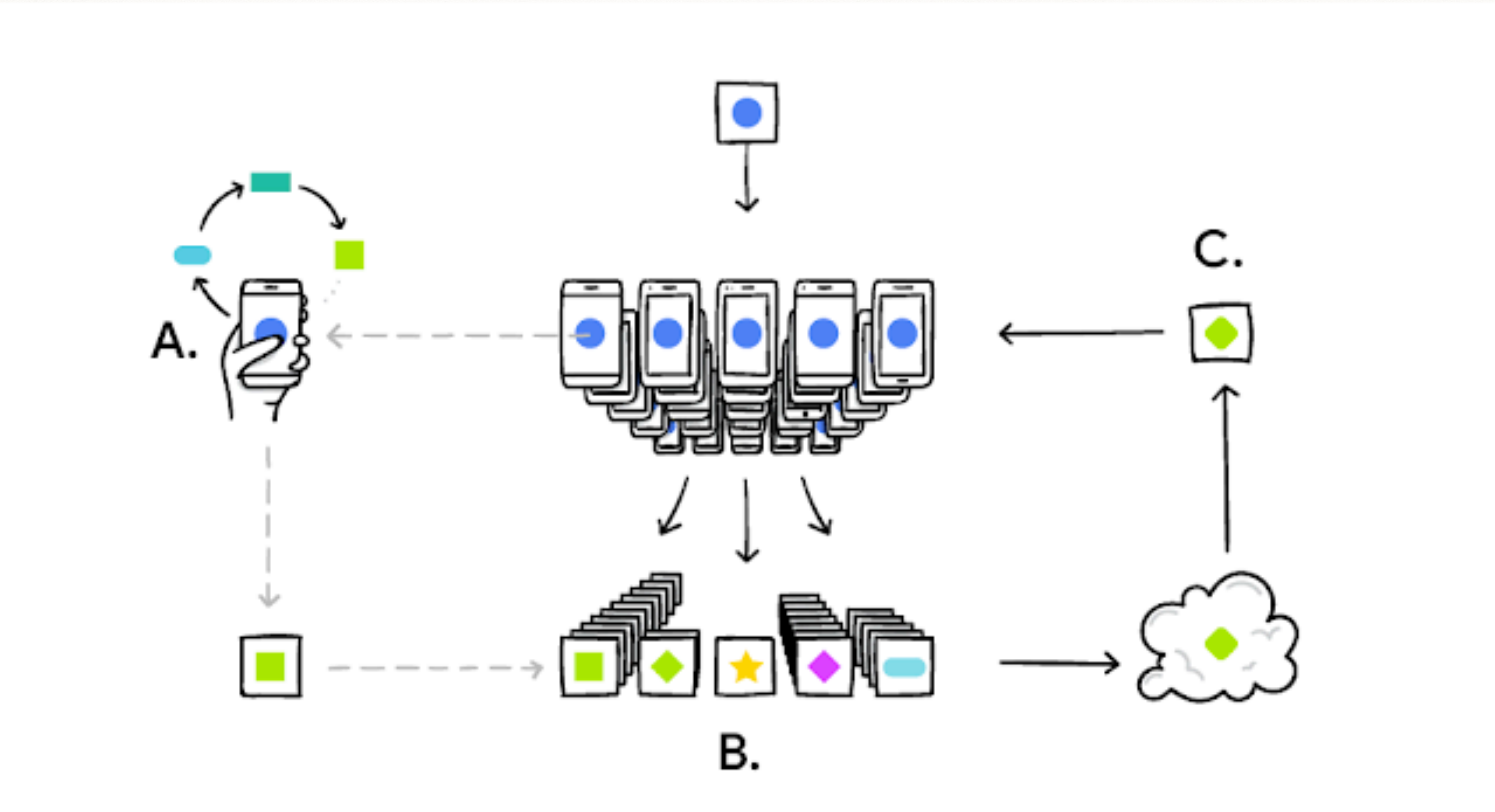
2

Leveraging Heterogenous Systems

Compute & Memory Hierarchy: Which data to put in which device?

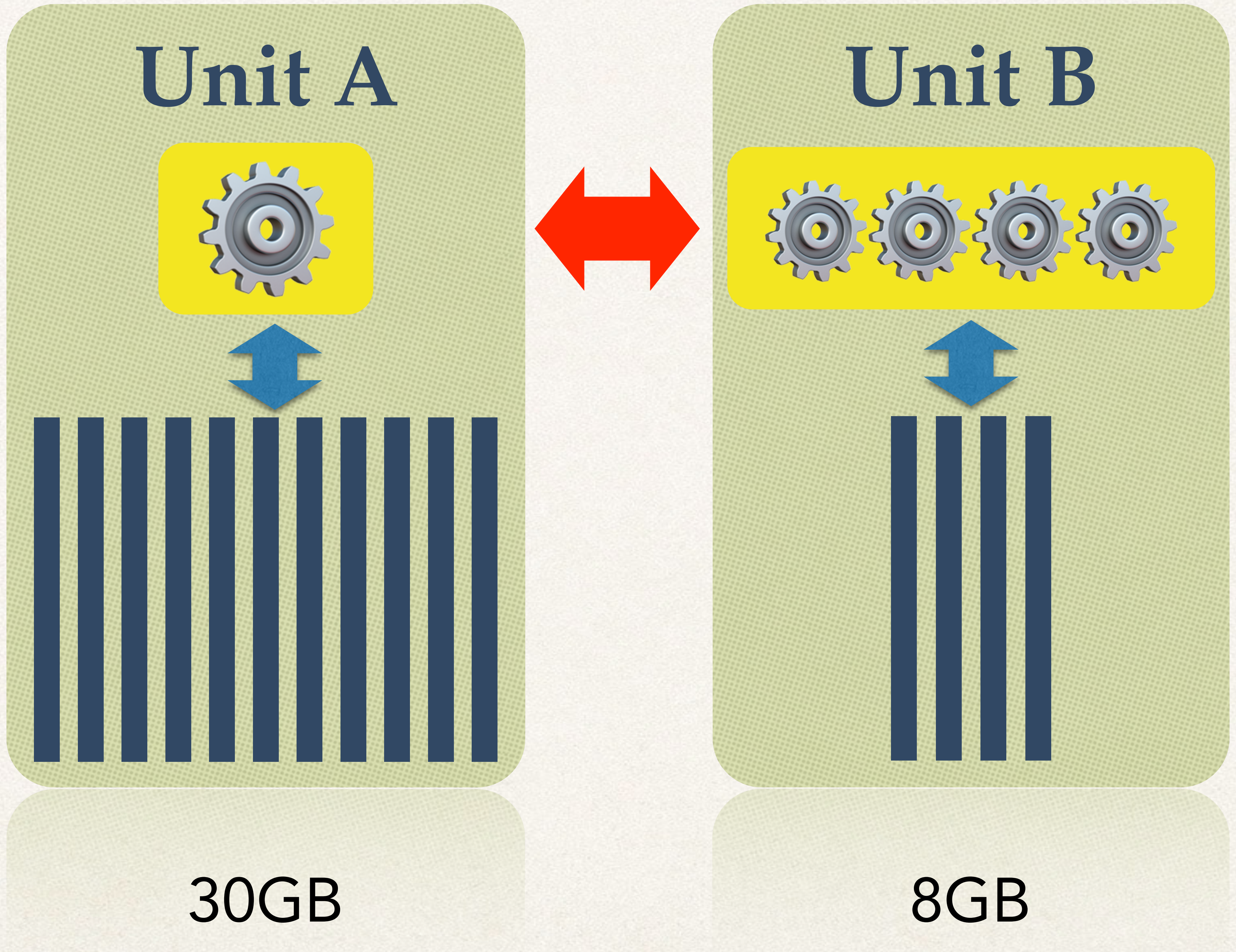


Decentralized / Federated Training



Leveraging Heterogenous Systems

duality gap as selection criterion

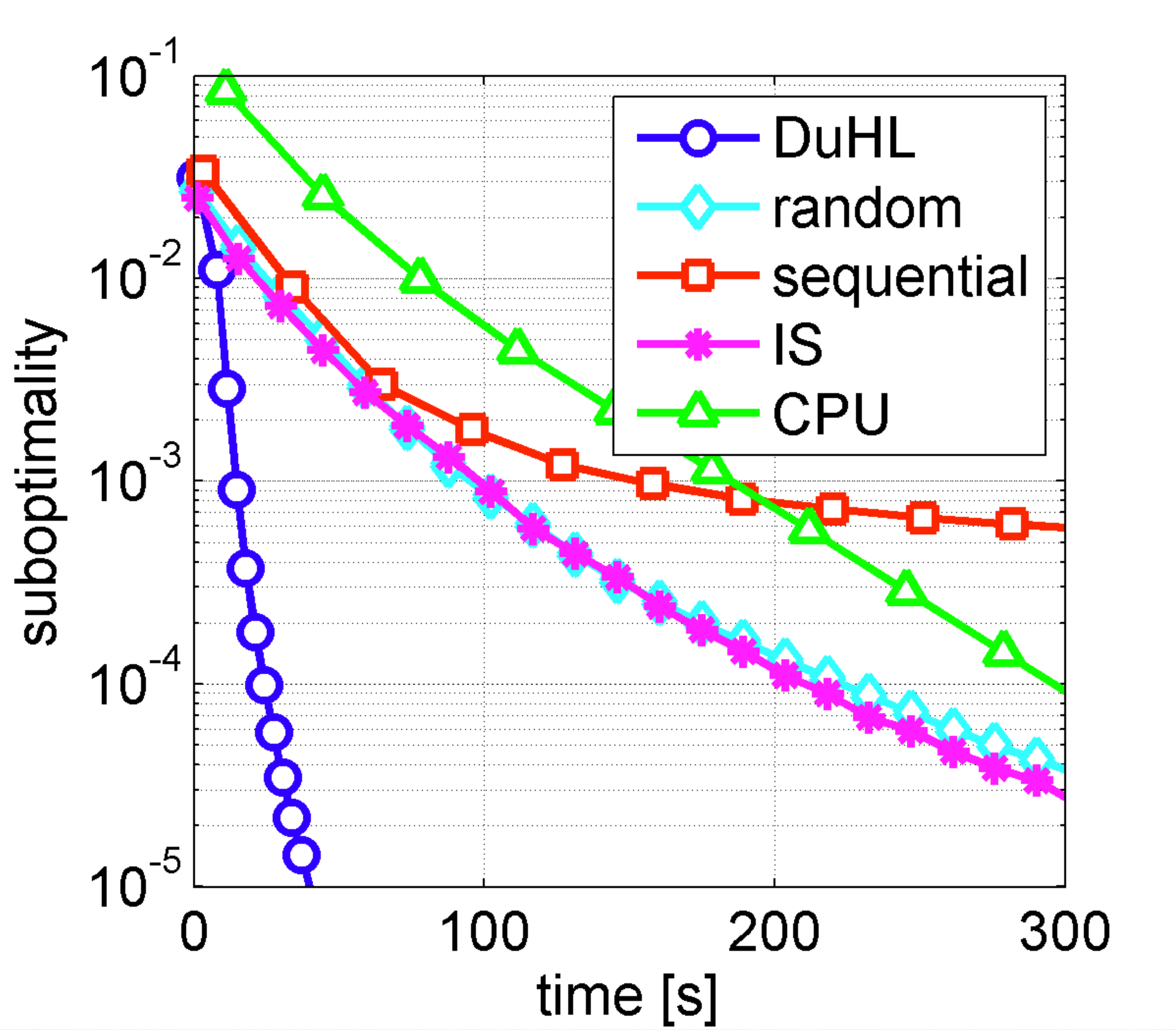


adaptive importance sampling

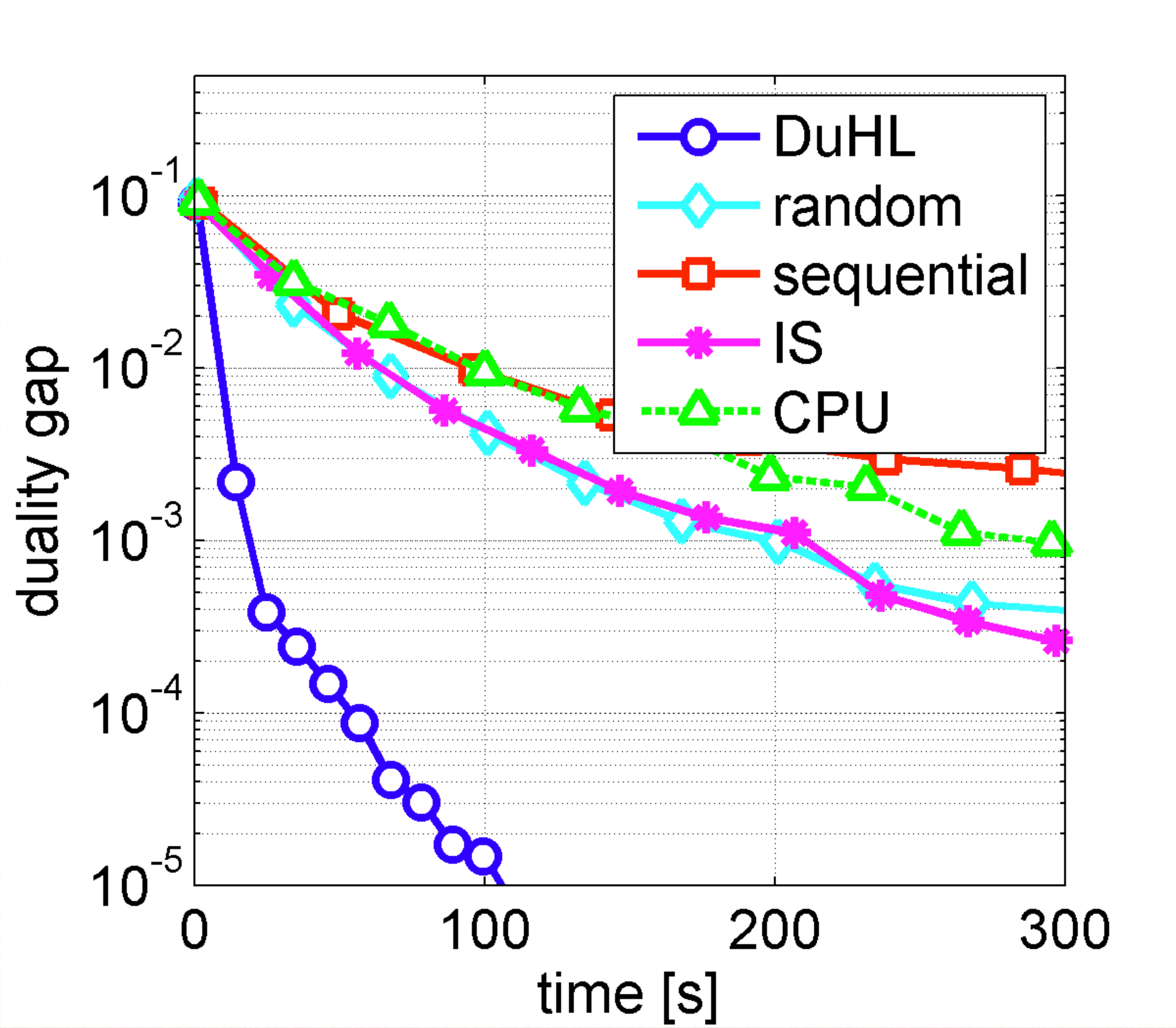
AISTATS 2017, 2018
NIPS 2017a,b

Experiments

RAM ⟷ GPU, 30GB dataset



Lasso



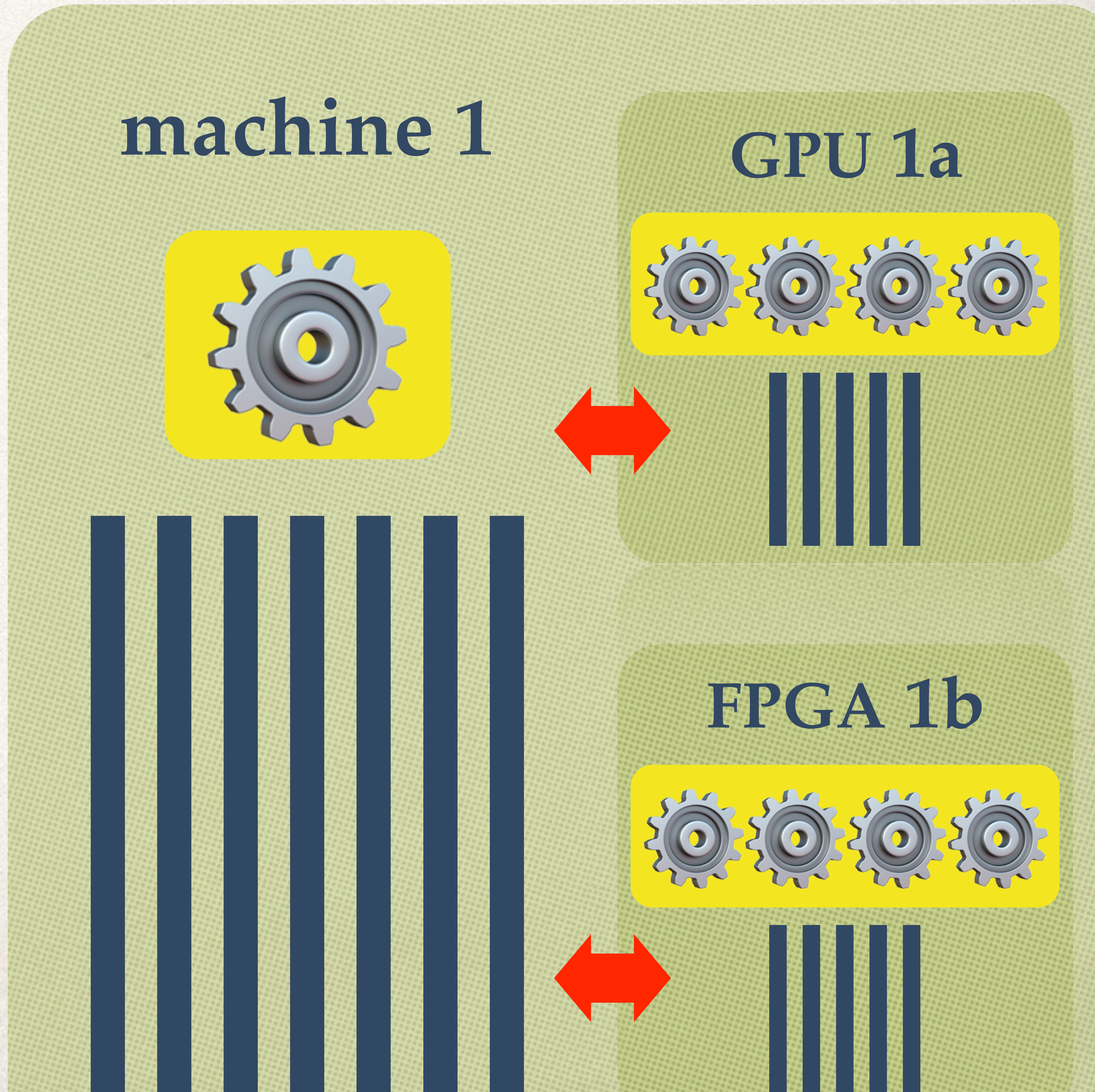
SVM

Experiments

terabyte click log dataset, IBM cloud implementation *[arXiv]*

Open Research

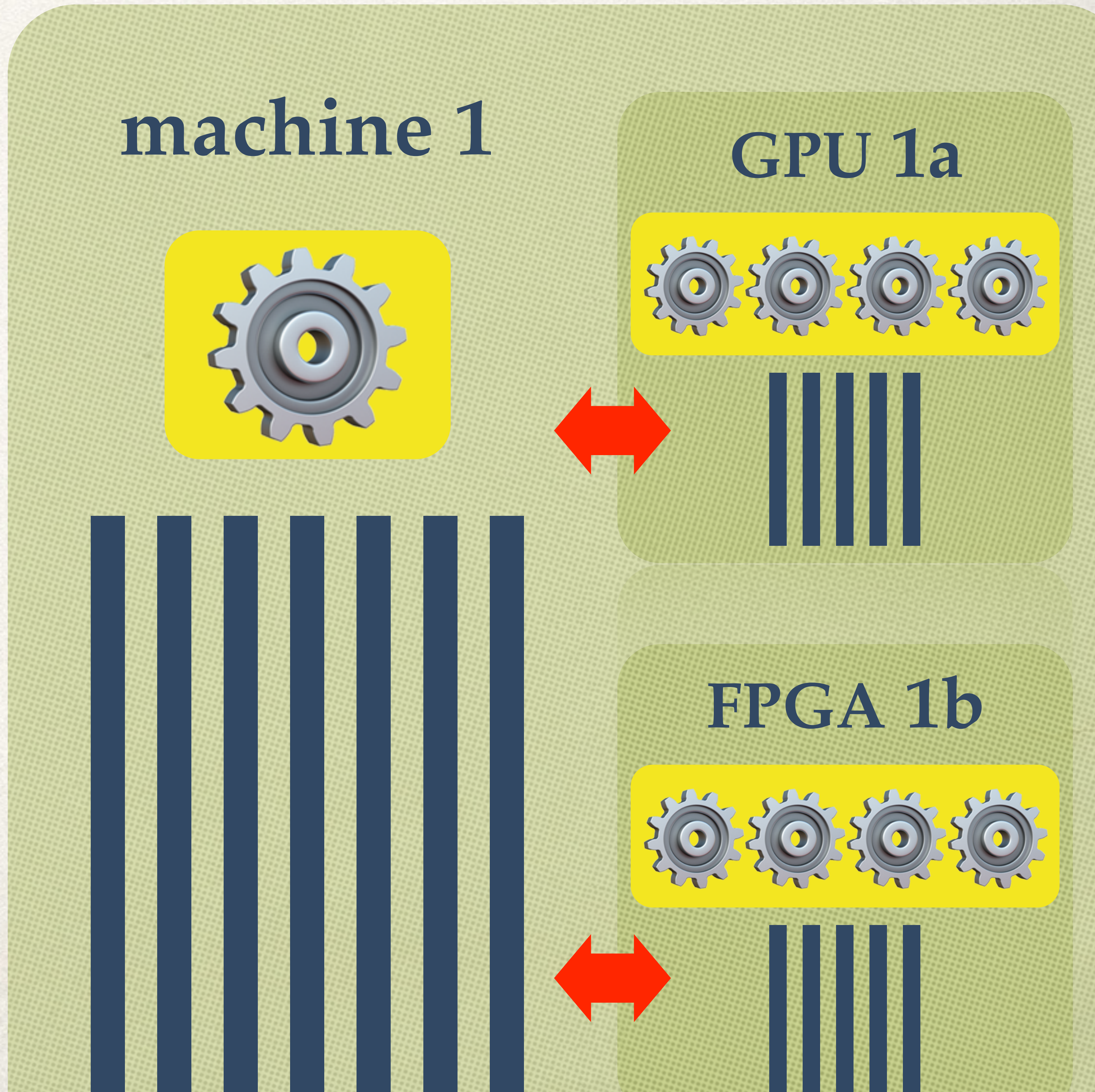
- **limited precision operations** for efficiency of communication and computation
- **asynchronous** and **fault tolerant** algorithms
- heterogenous systems
- more **re-usable** algorithmic building blocks
  - for more systems and problems



Trends - Systems

- new **hardware**
    - TPU, GraphCore
    - sparse ops?

- **Software** frameworks
    - AutoGrad (Tensorflow, PyTorch, etc)
    - Communication?



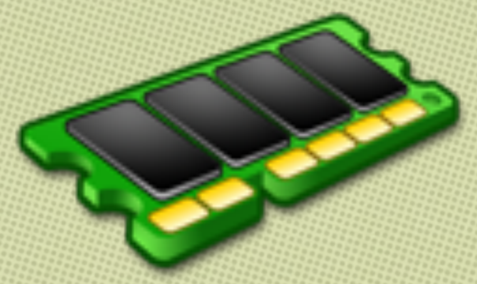
Challenge

The Cost of Communication

$\boldsymbol{v} \in \mathbb{R}^{100}$

- Reading $\boldsymbol{v}$ from memory (RAM)

  100 ns



- Sending $\boldsymbol{v}$ to another machine

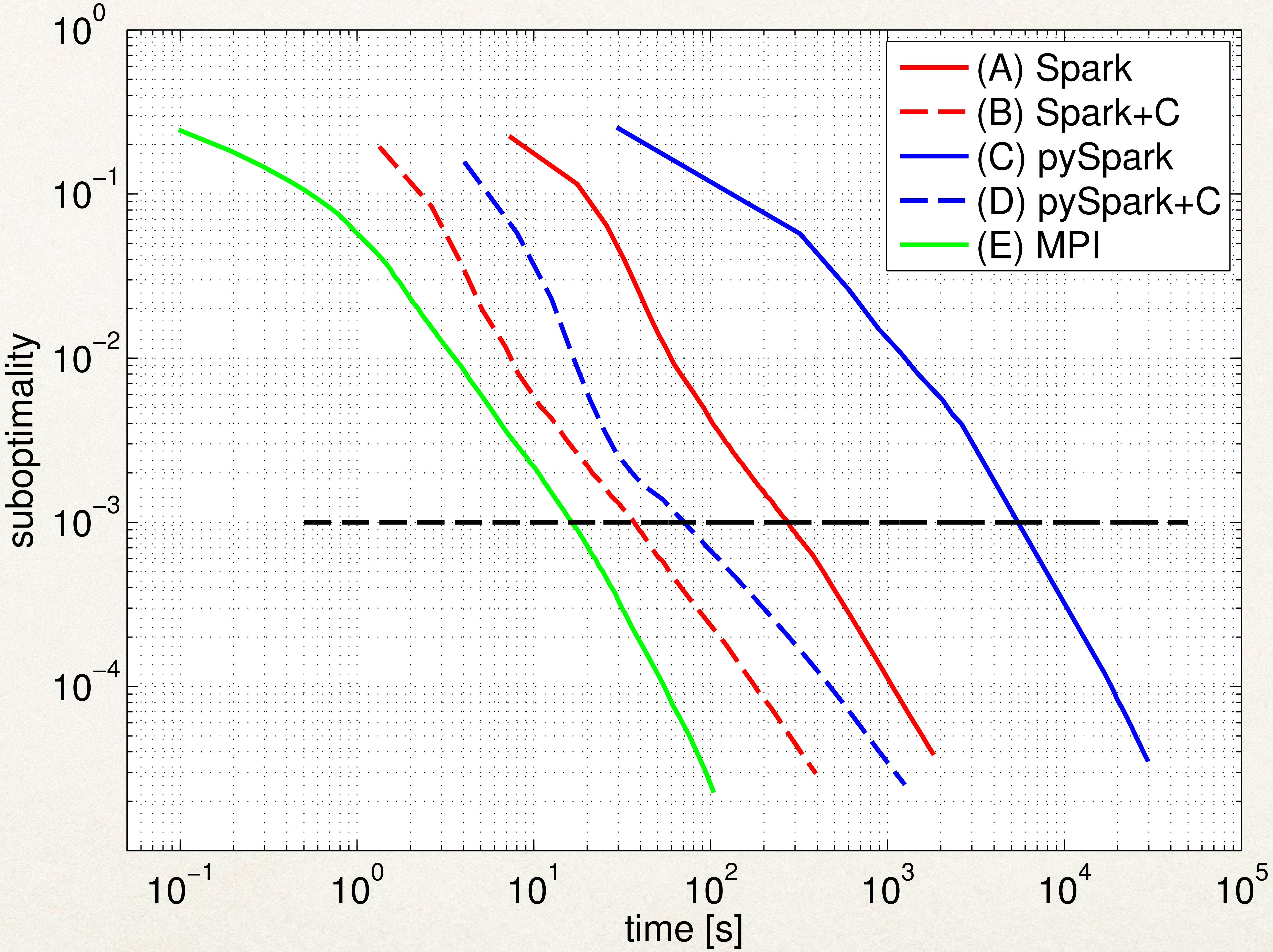
  500’000 ns

- Typical Map-Reduce iteration

  10’000’000’000 ns

Challenge

The Cost of Communication



High-Performance Distributed Machine Learning using Apache Spark
Dünner et al. 2016, arxiv.org/abs/1612.01437

Challenge

Usability

Good distributed and parallel code is hard

- no **reusability** of good single machine algorithms & code
- no portability: model-specific and system-specific code

Cloud ML

Microsoft Azure Machine Learning
Home
Studio
Gallery
Binary Classification: Direct marketing
In draft
Search experiment items
Saved Datasets
Data Format Conversions
Data Input and Output
Data Transformation
Feature Selection
Machine Learning
OpenCV Library Modules
Python Language Modules
R Language Modules
Statistical Functions
Text Analytics
Web Service
Deprecated
Reader
Metadata Editor
Project Columns
remove columns that are part of the label
Split
Two-Class Boosted Decision T...
Two-Class Support Vector Ma...
Split
Split
Sweep Parameters
Sweep Parameters
Score Model
Score Model
Evaluate Model
1:1
Properties
Two-Class Boosted Decision Tree
Create trainer mode
Single Parameter
Maximum number of leav...
20
Minimum number of sam...
10
Learning rate
0.2
Number of trees construct...
100
Random number seed
0
Allow unknown categ...
Quick Help
Creates a binary classifier using a boosted decision tree algorithm
(more help...)

Project:

Distributed Machine Learning Benchmark

Goal:

Public and Reproducible
Comparison of Distributed Solvers

github.com/mlbench/mlbench

PyTorch

PyTorch

Google

TensorFlow™

Apache

Spark

HPC

MPI

Auto ML

- **hyper-parameter optimization**
  zero-order methods
- **learning to learn**
  adaptive methods
- **neural architecture search**
  zero-order, warm-start

Thanks!

mlo.epfl.ch