Distributed Machine Learning: A Brief Overview

Dan Alistarh
IST Austria

Background

The Machine Learning “Cambrian Explosion”

Key Factors:

1. **Large Datasets**:
   - *Millions* of labelled images, *thousands of hours* of speech
2. **Improved Models and Algorithms**:
   - Deep Neural Networks: *hundreds* of layers, *millions* of parameters
3. **Efficient Computation for Machine Learning:**
   - Computational power for ML increased by ~100x since 2010 (Maxwell line to Volta)
   - **Gains *almost stagnant* in latest generations (GPU: <1.8x, CPU: <1.3x)**
   - **Computation times are extremely large anyway (days to weeks to months)**

Go-to Solution: **Distribute** Machine Learning Applications to Multiple Processors and Nodes

The Problem

CSCS: Europe's Top Supercomputer (World 3rd)

- 4500+ GPU Nodes, state-of-the-art interconnect

Task:

- Image Classification (ResNet-152 on ImageNet)
- Single Node time (TensorFlow): **19 days**
- 1024 Nodes: **25 minutes** (***in theory***)

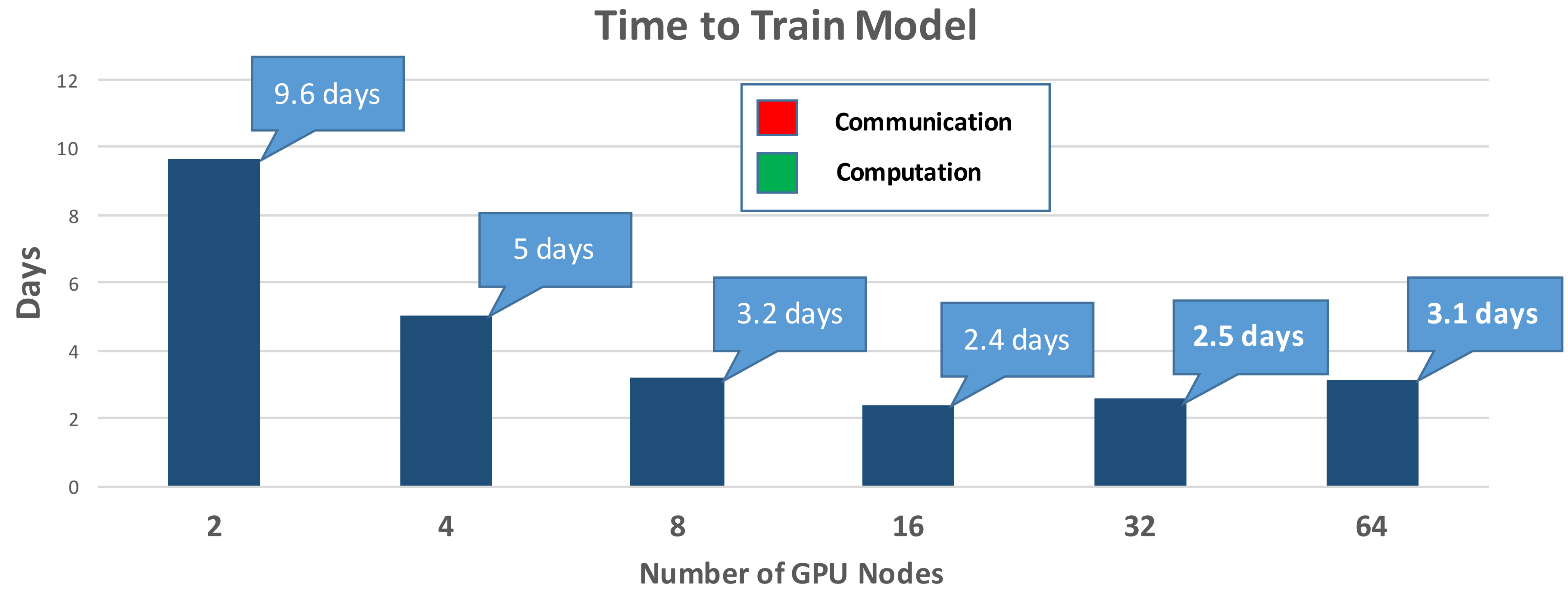
The Problem

CSCS: Europe's Top Supercomputer (World 3rd)

- 4500+ GPU Nodes, state-of-the-art interconnect

Task:

- Image Classification (ResNet-152 on ImageNet)
- Single Node time (TensorFlow): **19 days**
- 1024 Nodes: **25 minutes** (***in theory***)



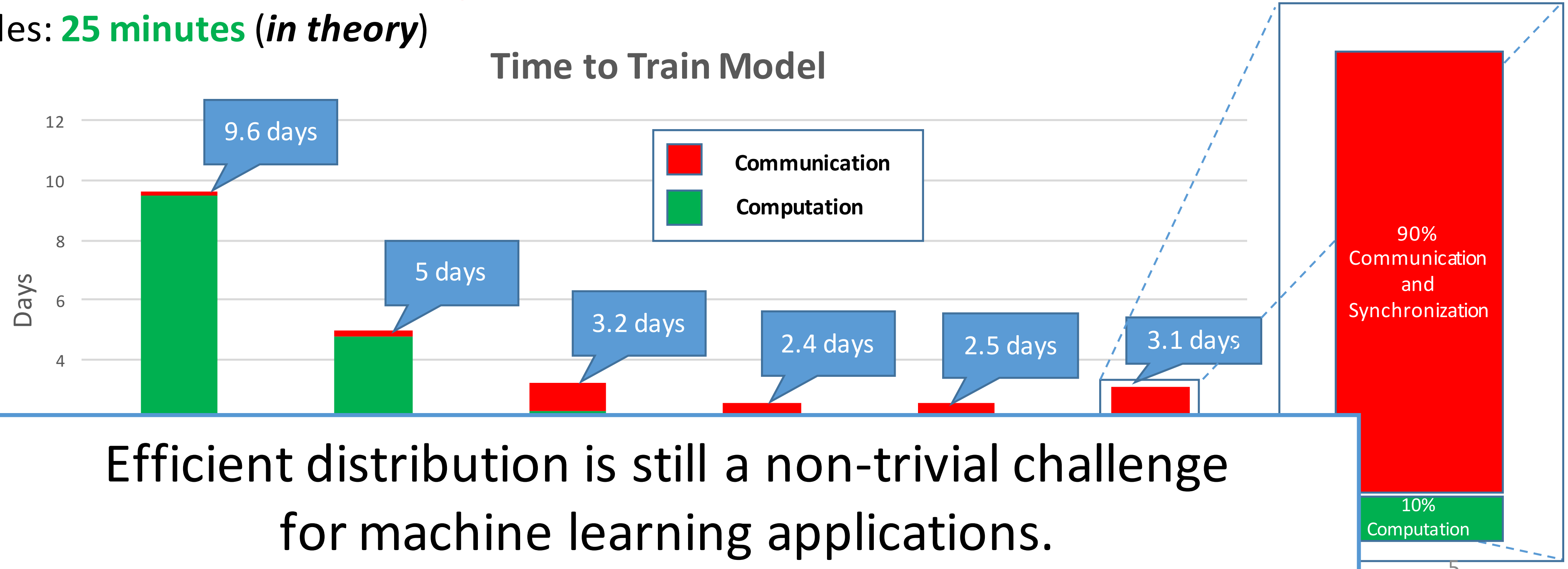
The Problem

CSCS: Europe's Top Supercomputer (World 3rd)

- 4500+ GPU Nodes, state-of-the-art interconnect

Task:

- Image Classification (ResNet-152 on ImageNet)
- Single Node time (TensorFlow): **19 days**
- 1024 Nodes: **25 minutes** (***in theory***)



Efficient distribution is still a non-trivial challenge for machine learning applications.

Part 1: Basics

Machine Learning in 1 Slide

$$\text{argmin}_{\boldsymbol{x}} f(\boldsymbol{x})$$

Solved via **optimization procedure**, e.g. **stochastic gradient descent** (SGD).

$$f(x) = \sum_{i=1}^{M} loss(x, exi)$$

Notion of “quality,” e.g. **squared distance**

$$\text{model } \boldsymbol{x}$$

E.g., **neural network** or **linear model**

Task

E.g., **classification**

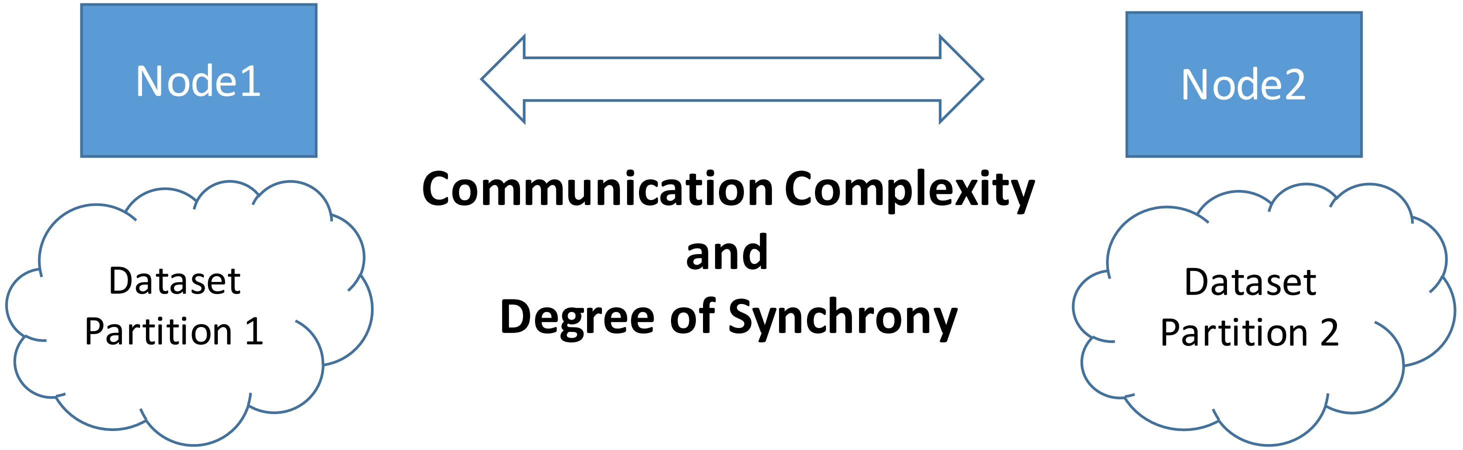
Data

Distributed Machine Learning in 1 Slide

$$\operatorname{argmin}_{\boldsymbol{x}} f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$$

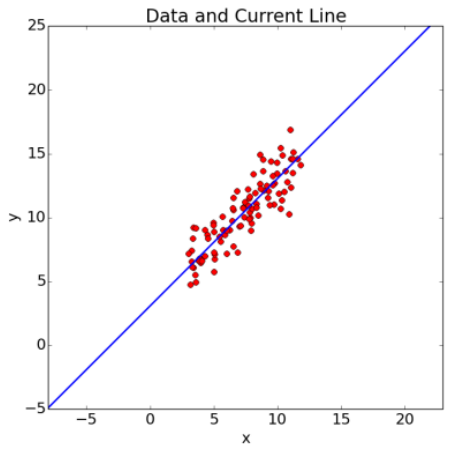
$$f_1(x) = \sum_{i=1}^{M/2} l(x, ei) \qquad f_2(x) = \sum_{i=\frac{M}{2}+1}^{M} l(x, ei)$$

model $\boldsymbol{x}$ model $\boldsymbol{x}$



This is the (somewhat standard) **data parallel** paradigm, but there are also **model parallel** or **hybrid** approaches.

The Optimization Procedure:
Stochastic Gradient Descent



- Gradient descent (GD): $$\boldsymbol{x_{t+1}} = \boldsymbol{x_t} - \boldsymbol{\eta_t} \nabla f(x_t).$$
- ***Stochastic*** gradient descent:
  Let $\widetilde{\boldsymbol{g}}(\boldsymbol{xt})$ = gradient at ***randomly chosen*** point.

$$\boldsymbol{x_{t+1}} = \boldsymbol{x_t} - \boldsymbol{\eta_t} \widetilde{\boldsymbol{g}}(\boldsymbol{xt}), \quad \text{where } \boldsymbol{E}[\tilde{g}(xt)] = \nabla f(x_t).$$

- Let $\boldsymbol{E}\left[||\widetilde{\boldsymbol{g}}(\boldsymbol{x}) - \boldsymbol{\nabla f}(\boldsymbol{x})||^2\right] \leq \boldsymbol{\sigma^2}$ (variance bound)

<u>Theorem</u> [classic]**:** Given f **convex** and ***L*-smooth**, and $R^2 = ||x_0 - x^*||^2$.
If we run SGD for $\boldsymbol{T} = \boldsymbol{\mathcal{O}}(\boldsymbol{R^2} \frac{\boldsymbol{2\sigma^2}}{\boldsymbol{\varepsilon^2}})$ iterations, then

$$\boldsymbol{E}\left[f\left(\frac{1}{T}\sum_{t=0}^{T} x_t\right)\right] - f(x^*) \leq \varepsilon.$$

A Compromise

- ***Mini-batch*** SGD:
  Let $\widetilde{\boldsymbol{g}}_{\boldsymbol{B}}(\boldsymbol{xt})$ = stochastic gradient with respect to a set of ***B randomly chosen*** points.

$$\boldsymbol{x}_{\boldsymbol{t+1}} = \boldsymbol{x}_{\boldsymbol{t}} - \boldsymbol{\eta}_{\boldsymbol{t}}\widetilde{\boldsymbol{g}}_{\boldsymbol{B}}(\boldsymbol{xt}), \quad \text{where } \boldsymbol{E}[\tilde{g}_B(xt)] = \nabla f(x_t).$$

- Why is this better?
  - The variance $\boldsymbol{\sigma}^{\mathbf{2}}$ of $\widetilde{\boldsymbol{g}}_{\boldsymbol{B}}(\boldsymbol{xt})$ is reduced linearly by ***B*** with respect to $\widetilde{\boldsymbol{g}}(\boldsymbol{xt})$
  - By the previous Theorem, the algorithm will converge in ***B*** times less iterations (in the **convex** case)

Note: Convergence is less well understood for **non-convex** optimization objectives (e.g., neural nets). In this case, it's known that SGD converges to a **local optimum** (point where gradient = 0).

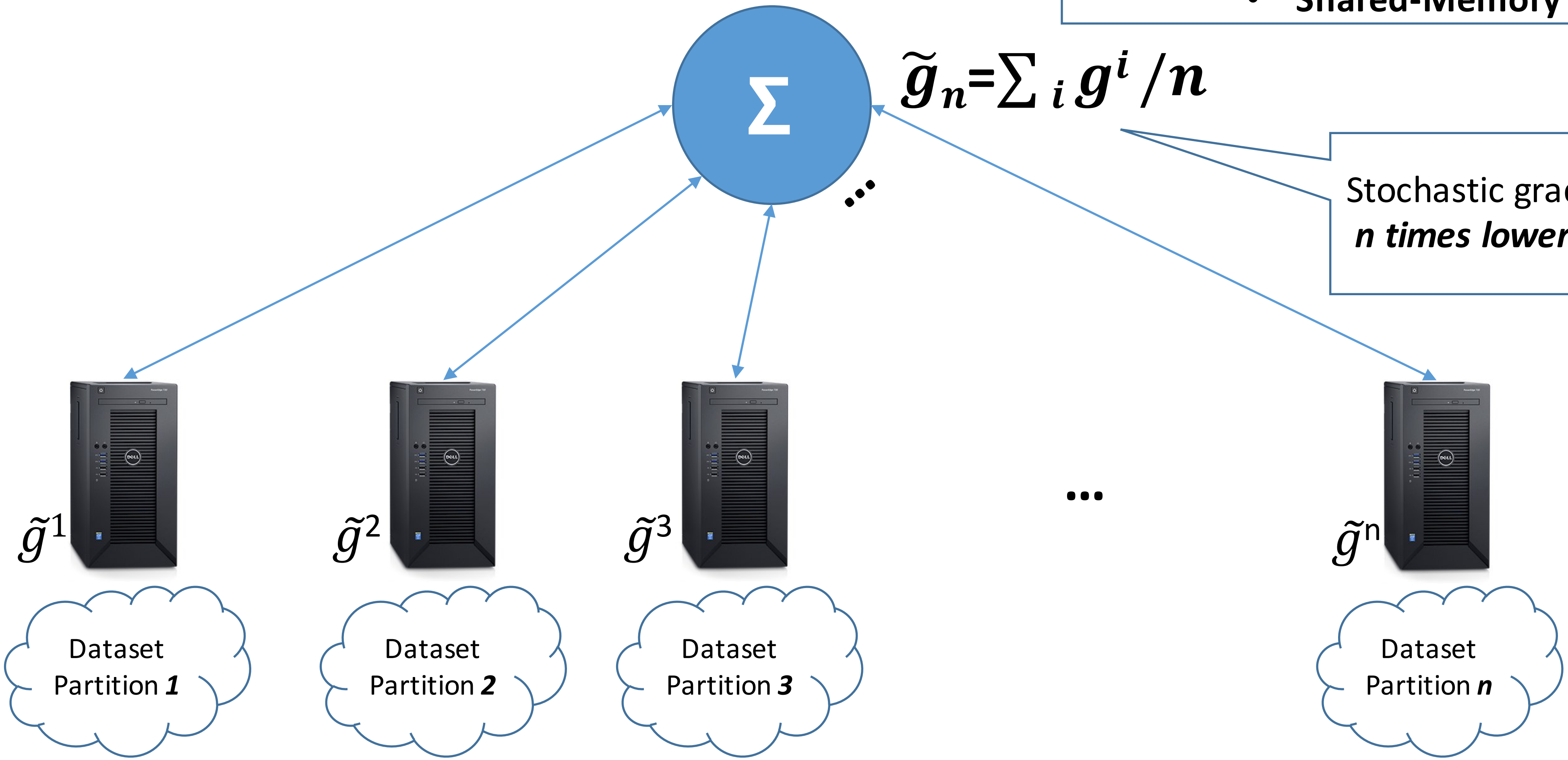
SGD Parallelization

Aggregation can be performed via:

- **Master node ("parameter server")**
- **MPI All-Reduce ("decentralized")**
- **Shared-Memory**

$$\tilde{g}_n = \sum_i g^i / n$$

Stochastic gradient with ***n times lower variance***



Theory: by distributing, we can perform ***P*** times more work per "clock step."
Hence, we should converge ***P*** times faster in terms of **wall-clock time**.

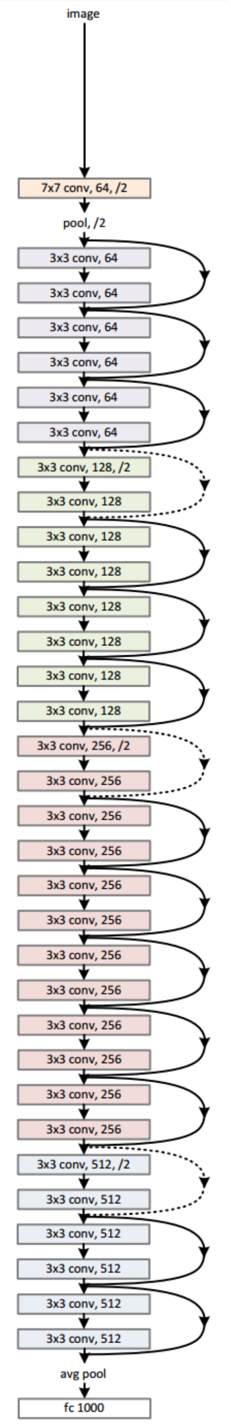
Embarrassingly parallel?

The Practice

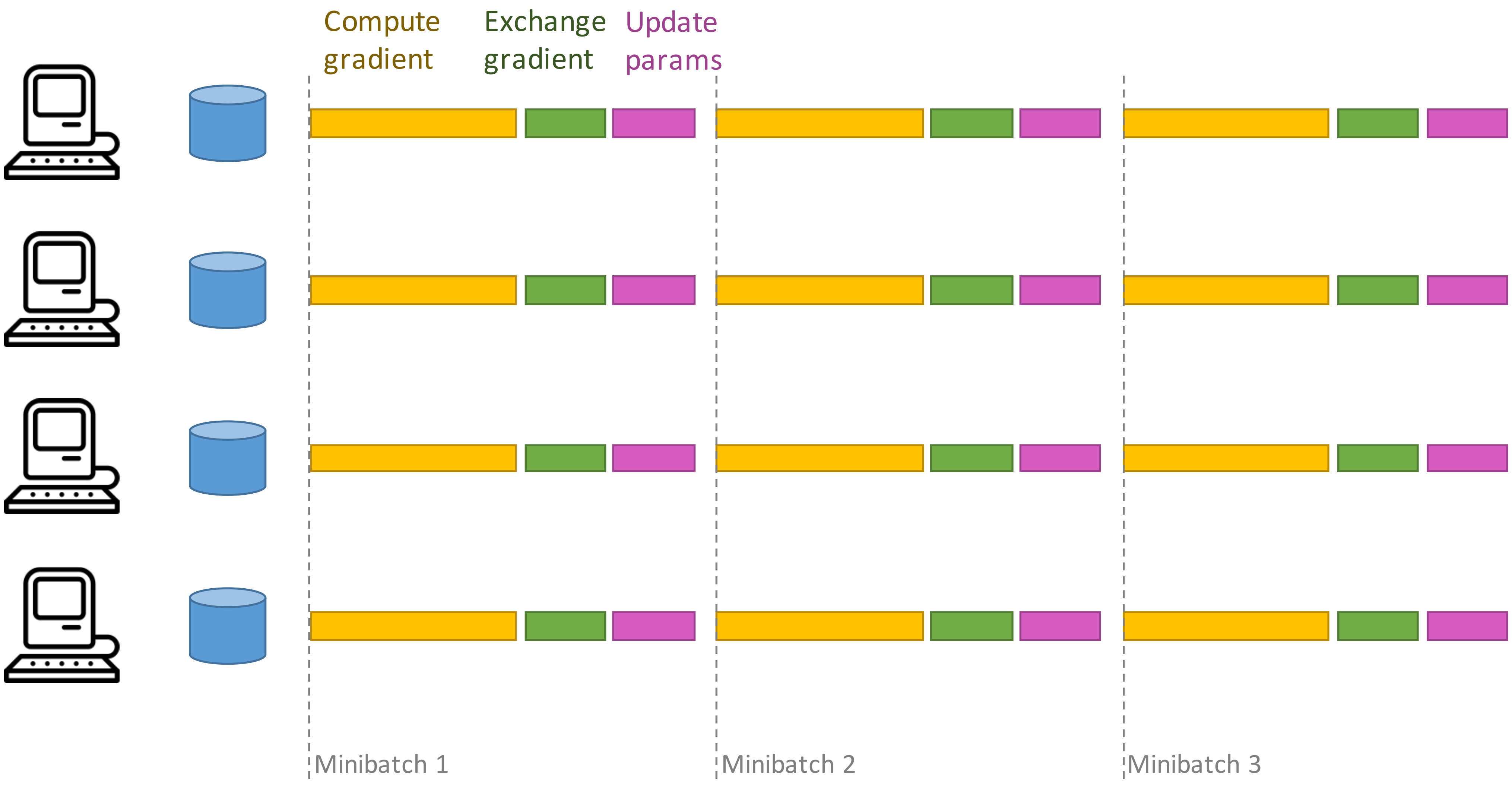
Training very large models efficiently

- Vision
  - ImageNet: **1.3 million images**
  - ResNet-152 [He+15]: **152 layers, 60 million parameters**
  - **Model/update size: approx. 250MB**
- Speech
  - NIST2000 Switchboard dataset: **2000 hours**
  - LACEA [Yu+16]: **22 LSTM (recurrent) layers**, **65 million parameters** (w/o language model)
  - **Model/update size: approx. 300MB**

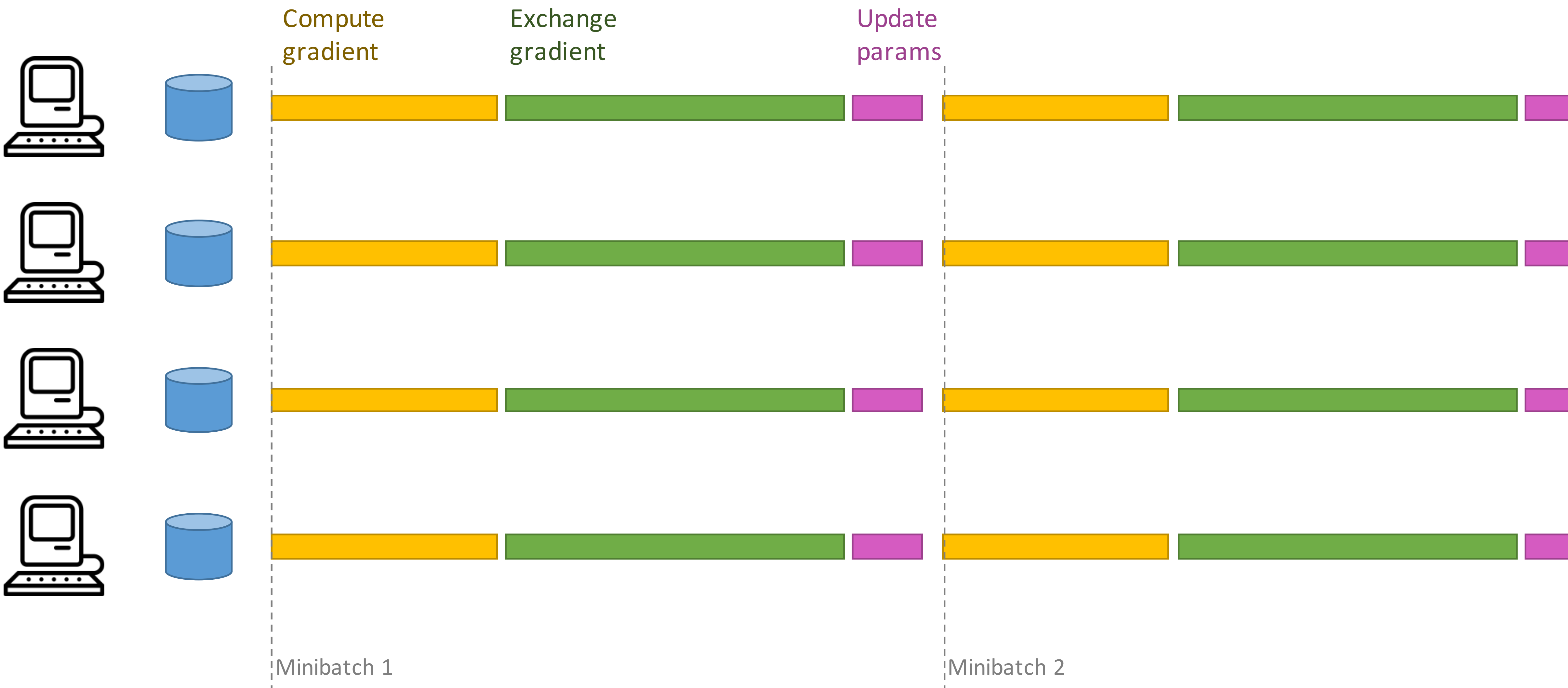
He et al. (2015) “Deep Residual Learning for Image Recognition”
Yu et al. (2016) “Deep convolutional neural networks with layer-wise context expansion and attention”



Data parallel SGD

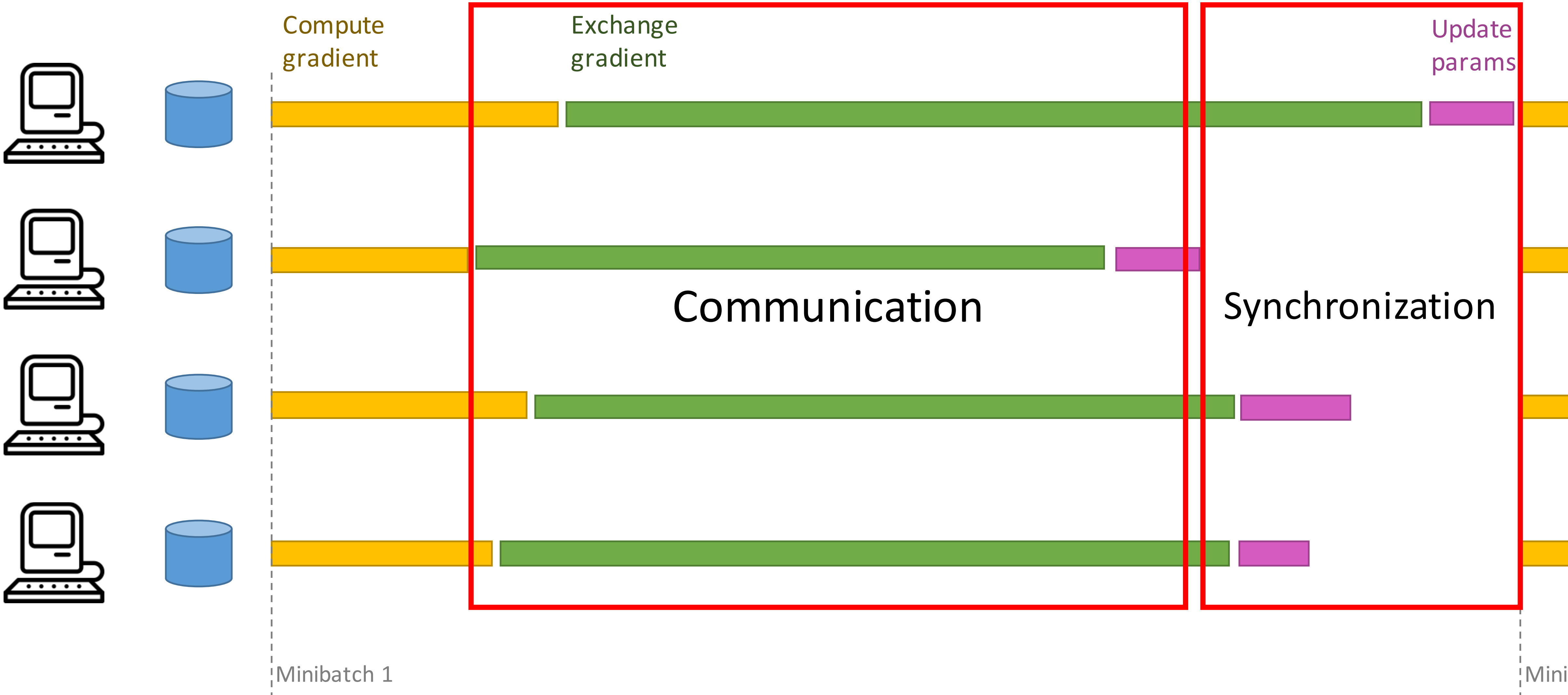


Data parallel SGD (**bigger models**)



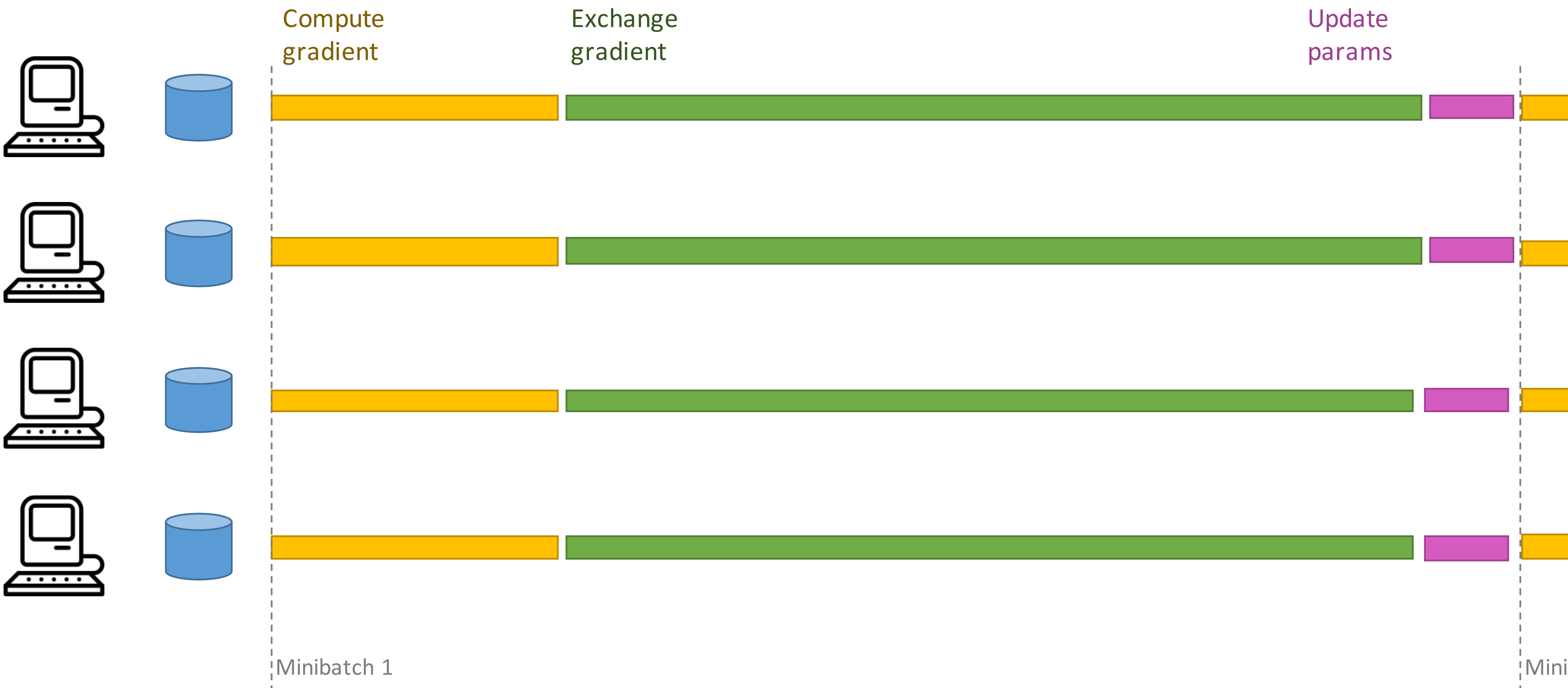
Data parallel SGD (***biggerer*** model)

More Precisely: Two major costs



Part 2: Communication-Reduction Techniques

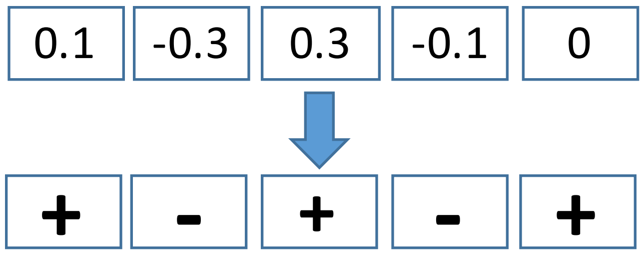
Data parallel SGD (***biggerer*** model)



Idea [Seide et al., 2014]: *compress* **the gradients…**

1BitSGD Quantization

[Microsoft Research, Seide et al. 2014]



$avg_+ = +0.2$

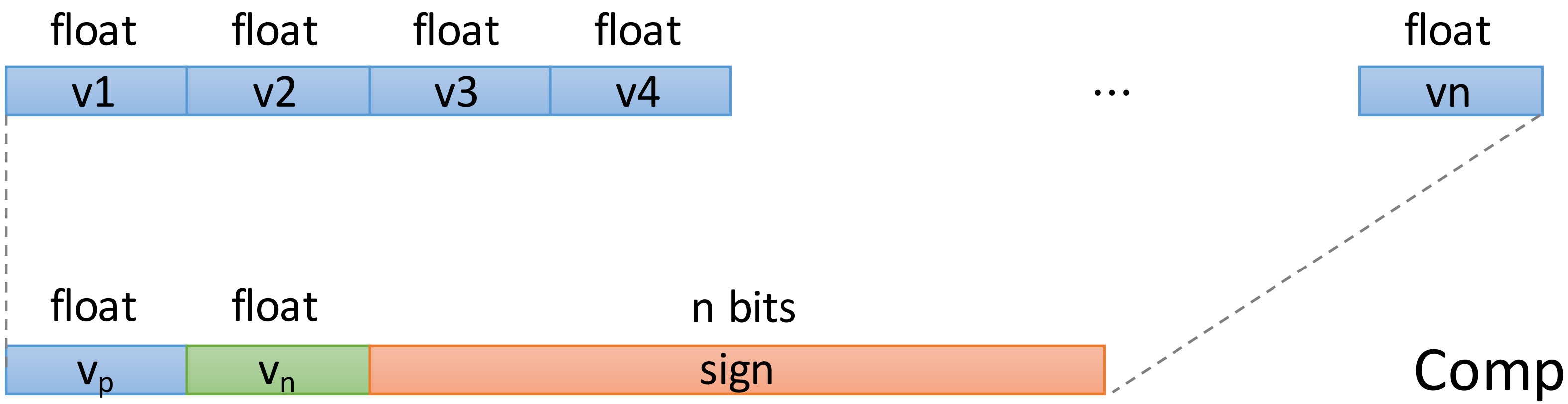
$avg_- = -0.2$

Quantization function

$$Q_i(v) = \begin{cases} avg_+ & \text{if } v_i \geq 0, \\ avg_- & \text{otherwise} \end{cases}$$

where $avg_+ = \text{mean}([v_i \text{ for } i: v_i \geq 0])$, $avg_- = \text{mean}([v_i \text{ for } i: v_i < 0])$

Accumulate the error locally, and apply to next gradient!



Compression rate ≈ 32x

Does not always converge!

Seide et al (2014) "1-Bit Stochastic Gradient Descent and its Application to Data-Parallel Distributed Training of Speech DNNs"

Why this shouldn't work

Let $\boldsymbol{Q(x)}$ be the gradient quantization function.

- Iteration:

$$\boldsymbol{x_{t+1} = x_t - \eta_t \tilde{g}(x_t)}$$

$$\boldsymbol{x_{t+1} = x_t - \eta_t Q(\tilde{g}(x_t))}$$

where $\boldsymbol{E}[\tilde{g}(xt)] = \nabla f(x_t)$.

No longer unbiased in 1BitSGD!

$$\boxed{\boldsymbol{E}[\tilde{g}(xt)] \neq \nabla f(x_t)}$$

- Let:
  - $\boldsymbol{E}\left[||\tilde{\boldsymbol{g}}(\boldsymbol{x}) - \boldsymbol{\nabla f(x)}||^2\right] \leq \boldsymbol{\sigma^2}$ (variance bound)

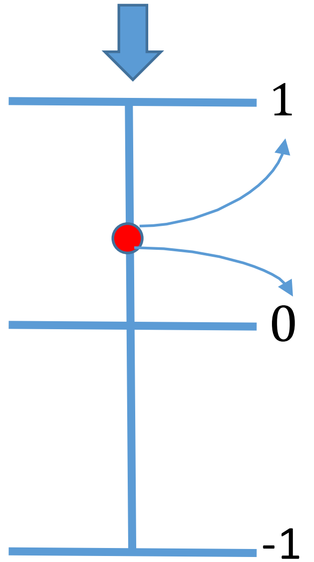
Theorem [classic]**:** Given f convex and L-smooth, and $R^2 = ||x_0 - x^*||^2$.
If we run SGD for $\boldsymbol{T} = \boldsymbol{\mathcal{O}}(\, \boldsymbol{R^2 \frac{2\sigma^2}{\varepsilon^2}})$ iterations, then

$$\boldsymbol{E}\left[f\left(\frac{1}{T}\sum_{t=0}^{T} x_t\right)\right] - f(x^*) \leq \varepsilon.$$

Take One: **Stochastic Quantization**

0.1	-0.3	0.3	-0.1	0

$||v||_2$=0.447



- Quantization function

$$Q(vi) = \|v\|_2 \cdot \operatorname{sgn}(v_i) \cdot \xi_i(v_i)$$

where $\xi_i(v_i) = 1$ with probability $|v_i|/\|v\|_2$ and 0 otherwise.

Properties:

1. **Unbiasedness**:

$$\boldsymbol{E}\left[\boldsymbol{Q}[\boldsymbol{v}_i]\right] = \|v\|_2 \cdot \operatorname{sgn}(v_i) \cdot |v_i|/\|v\|_2 = \operatorname{sgn}(v_i) \cdot |v_i|$$

2. **Second moment** (variance) bound:

$$\boldsymbol{E}\left[\|\boldsymbol{Q}[\boldsymbol{v}]\|^2\right] \leq \|v\|_2 \|v\|_1 \leq \sqrt{\boldsymbol{n}}\|\boldsymbol{v}\|^2$$

3. **Sparsity**: If ***v*** has dimension ***n***, then

$$\textbf{\textit{E}[non-zeroes in \textit{Q(v)}]} = \boldsymbol{E}\left[\sum_i \xi_i(\boldsymbol{v})\right] \leq \|\boldsymbol{v}\|_1/\|\boldsymbol{v}\|_2 \leq \sqrt{\boldsymbol{n}}$$

Convergence:

$$\boldsymbol{E}[Q[\tilde{g}(x_t)]] = \boldsymbol{E}[\tilde{g}(x_t)] = \nabla f(x_t)$$

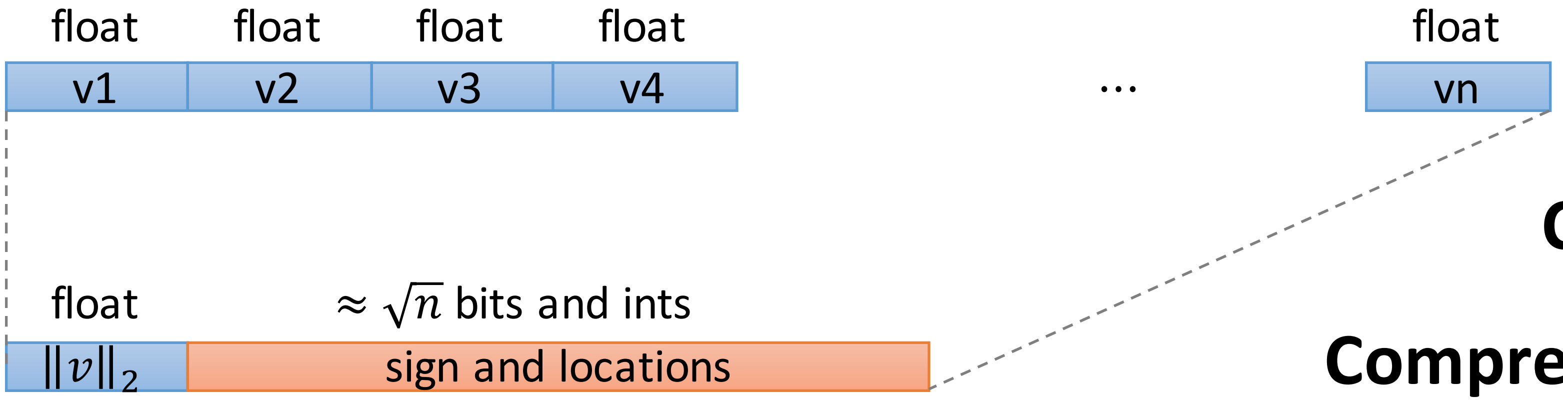
Runtime $\leq \sqrt{n}$ more iterations

Compression

- Quantization function

$$Q(vi) = \|v\|_2 \cdot \mathrm{sgn}(v_i) \cdot \xi_i(v_i)$$

where $\xi_i(v_i) = 1$ with probability $|v_i|/\|v\|_2$ and 0 otherwise.



Original: 32n bits

Compression $\approx \sqrt{n}/\log n$.

Compressed: 32 + $\sqrt{n}\log n$ **bits**

<u>Moral</u>: We're not too happy:

the $\sqrt{n}$ **increase in number of iterations offsets the** $\frac{\sqrt{n}}{\log n}$ compression.

Take Two: **QSGD**

[Alistarh, Grubic, Li, Tomioka, Vojnovic, NIPS17]

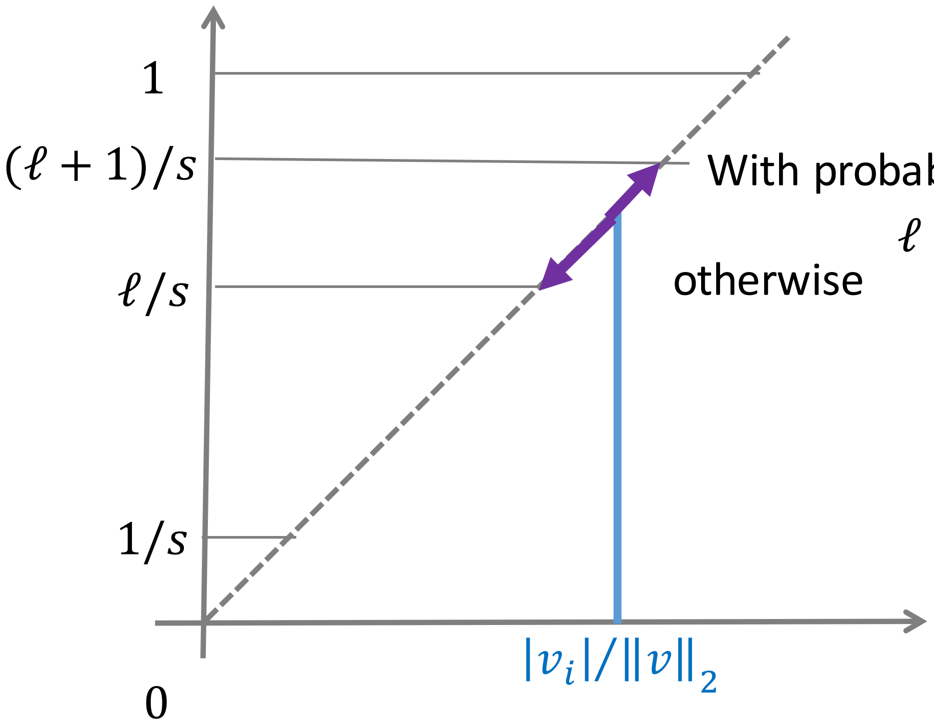
- Quantization function

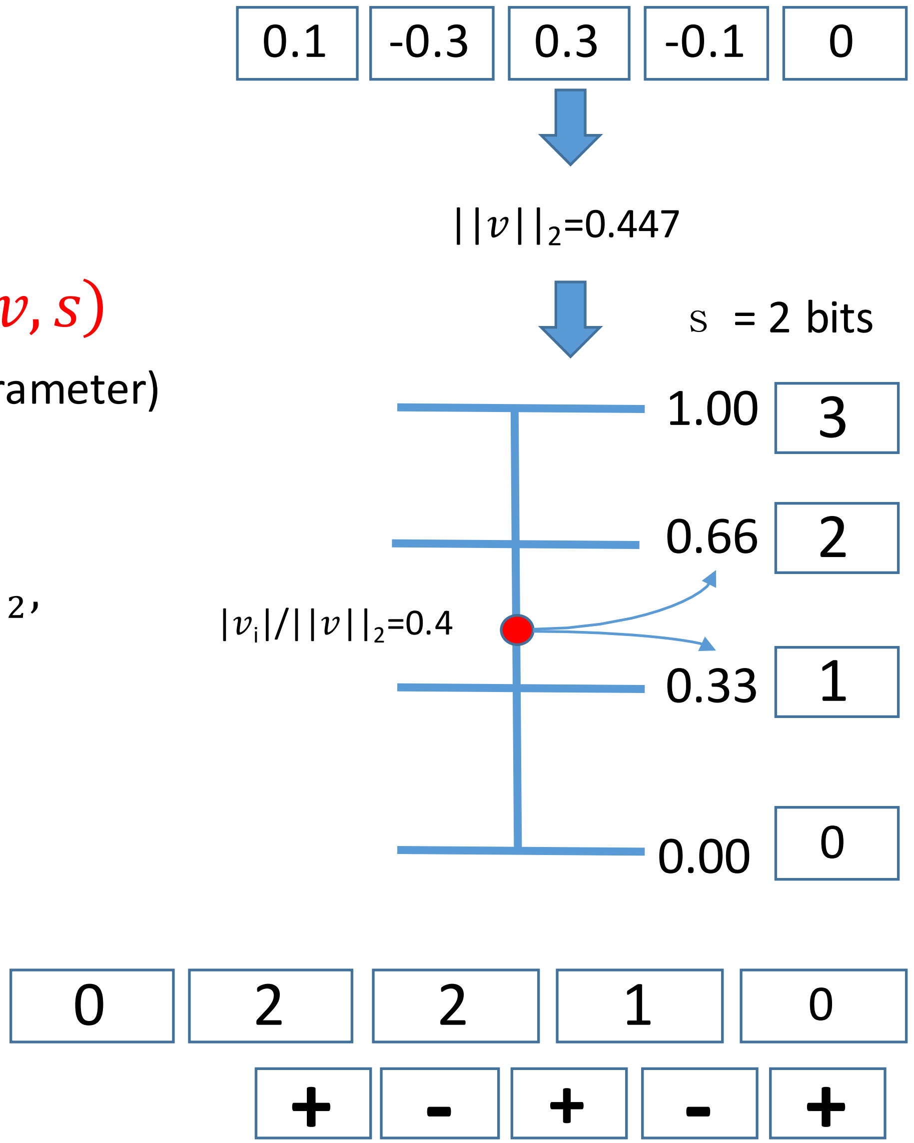
$$Q[v; s] = \|v\|_2 \cdot \mathrm{sgn}(v_i) \cdot \xi_i(v, s)$$

(s is a tuning parameter)

where

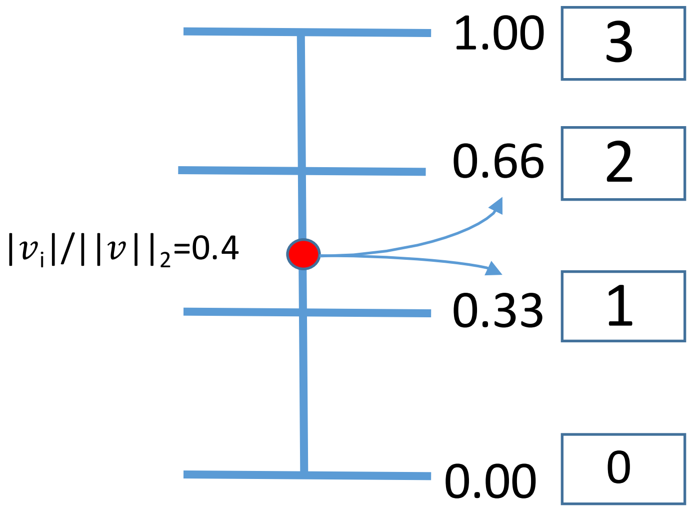
$$\xi_i(v,s) = \begin{cases} (\ell+1)/s & \text{With probability } \ell + 1 - s \cdot |v_i|/\|v\|_2, \\ \ell/s & \text{otherwise} \end{cases} \quad \ell = \mathrm{floor}(s \cdot |v_i|/\|v\|_2)$$





- Note: *s*=1 reduces to the two-bit quantization function.

QSGD Properties



- Quantization function

$$Q[v_i; s] = \|v\|_2 \cdot \mathrm{sgn}(v_i) \cdot \xi_i(v, s)$$

- Properties

1. Unbiasedness

$$E\big[Q[v_i; s]\big] = v_i$$

2. Sparsity

$$E[\|Q(v,s)\|_0] \leq s^2 + \sqrt{n}$$

3. Second moment bound

$$E[\|Q[v;s]\|_2^2] \leq \left(1 + \min\left(\frac{n}{s^2}, \frac{\sqrt{n}}{s}\right)\right) \cdot \|v\|_2^2$$

(Multiplier only 2 for $s = \sqrt{n}$)

Two Regimes

<u>Theorem 1</u> (constant s): The expected bit length of the quantized gradient is $32 + \left(\boldsymbol{s}^2 + \sqrt{\boldsymbol{n}}\right)\log \boldsymbol{n}$.

<u>Theorem 2</u> (large s): For $s = \sqrt{n}$, the expected bit length of the quantized gradient is $32 + 2.8 \cdot \boldsymbol{n}$, and the added variance is **constant**.

- **Idea1**: there can be **few large integer values** encoded
- **Idea2**: Use **Elias recursive coding** to code integers efficiently

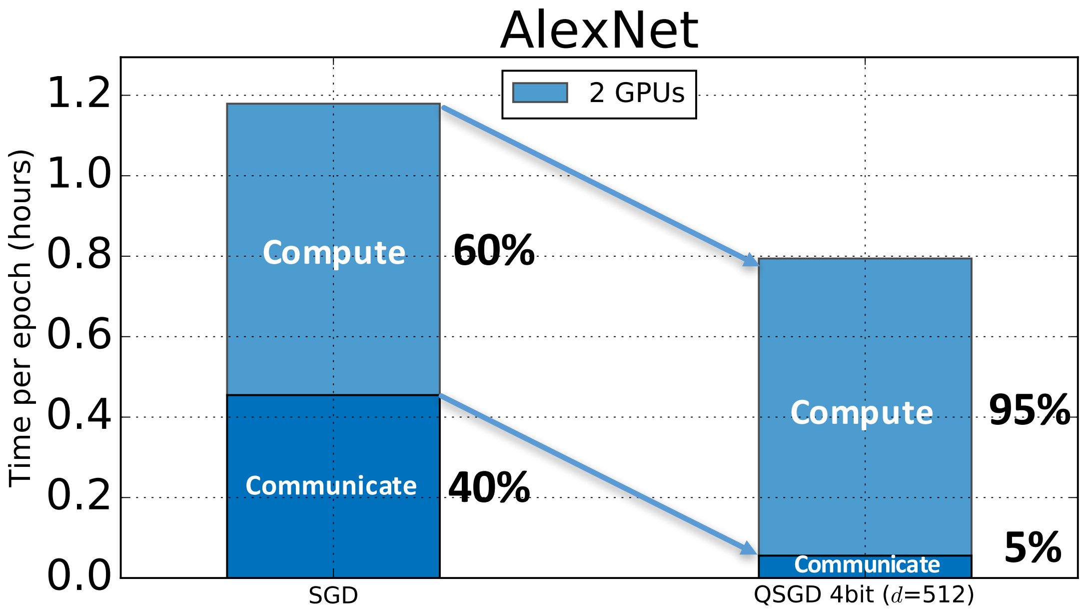
Original: 32*n bits*.

<u>Theorem</u> [Tsitsiklis&Luo, '86]: Given dimension **n,** the necessary number of bits for approximating the minimum within *ε* is **Ω (n (log n + log (1 / *ε*)))**.

Matches Thm 2.

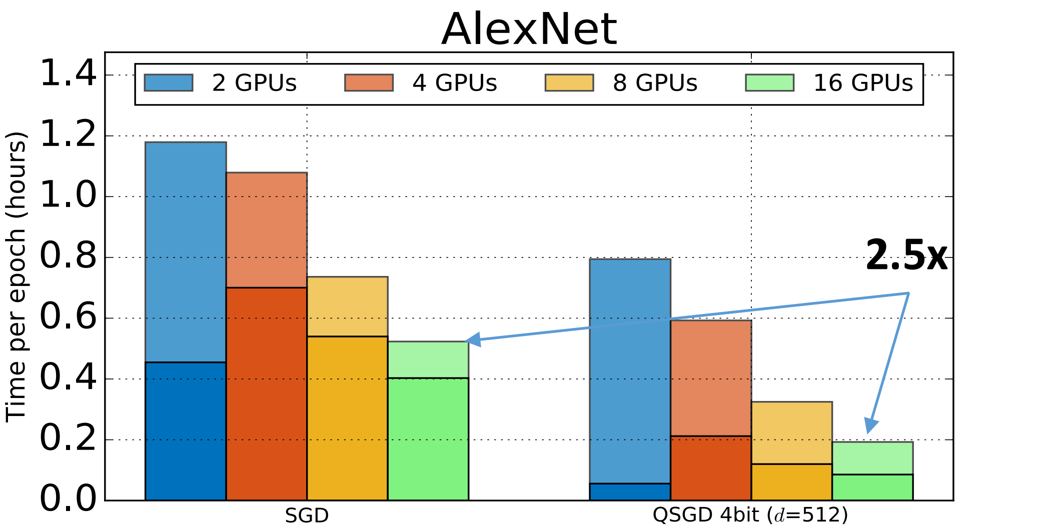
Does it actually work?

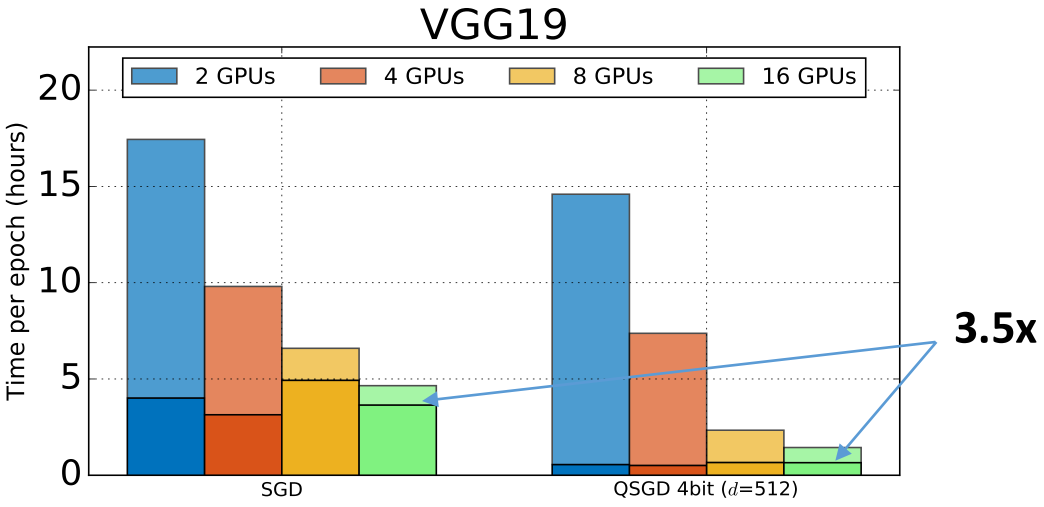
- **Amazon EC2 p2.xlarge machine**
- **AlexNet model (60M params)** x **ImageNet dataset** x **2 GPUs**
- **QSGD 4bit quantization (s = 16)**
- **No additional hyperparameter tuning**

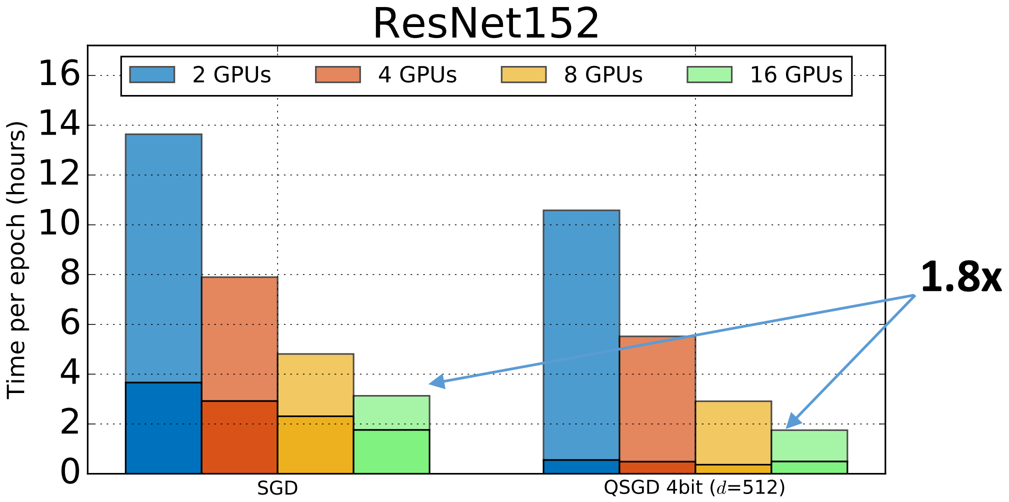


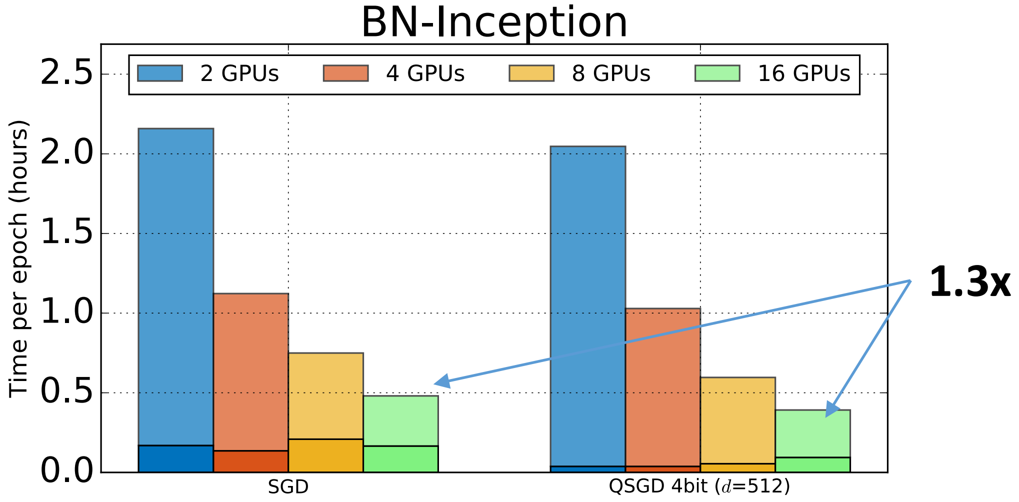
SGD vs QSGD on AlexNet.

Experiments: "Strong" Scaling

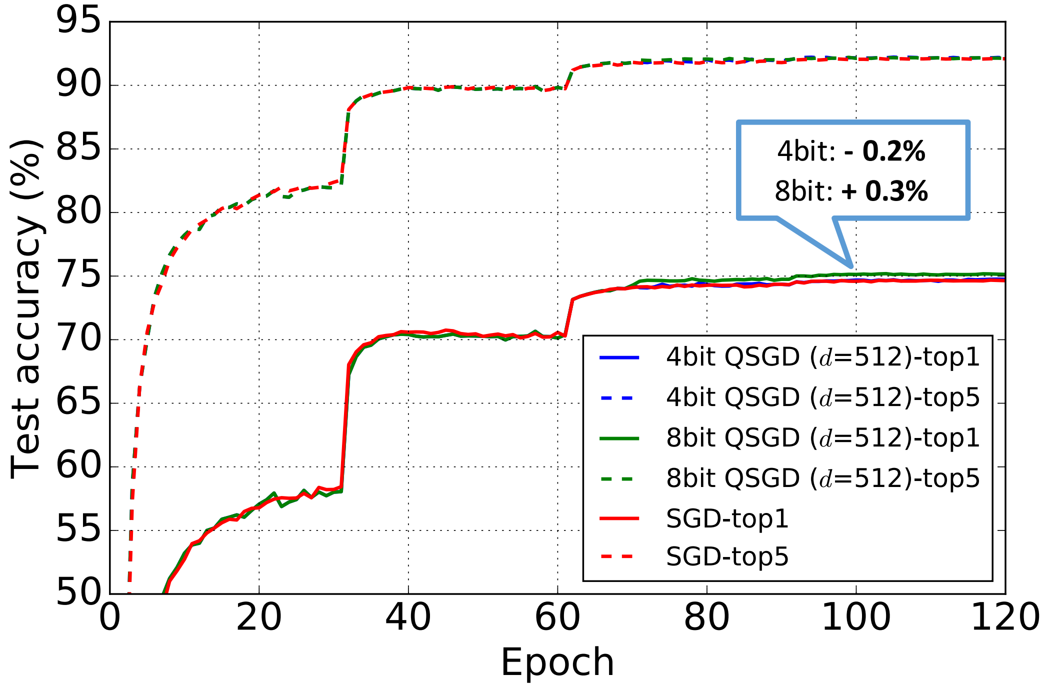




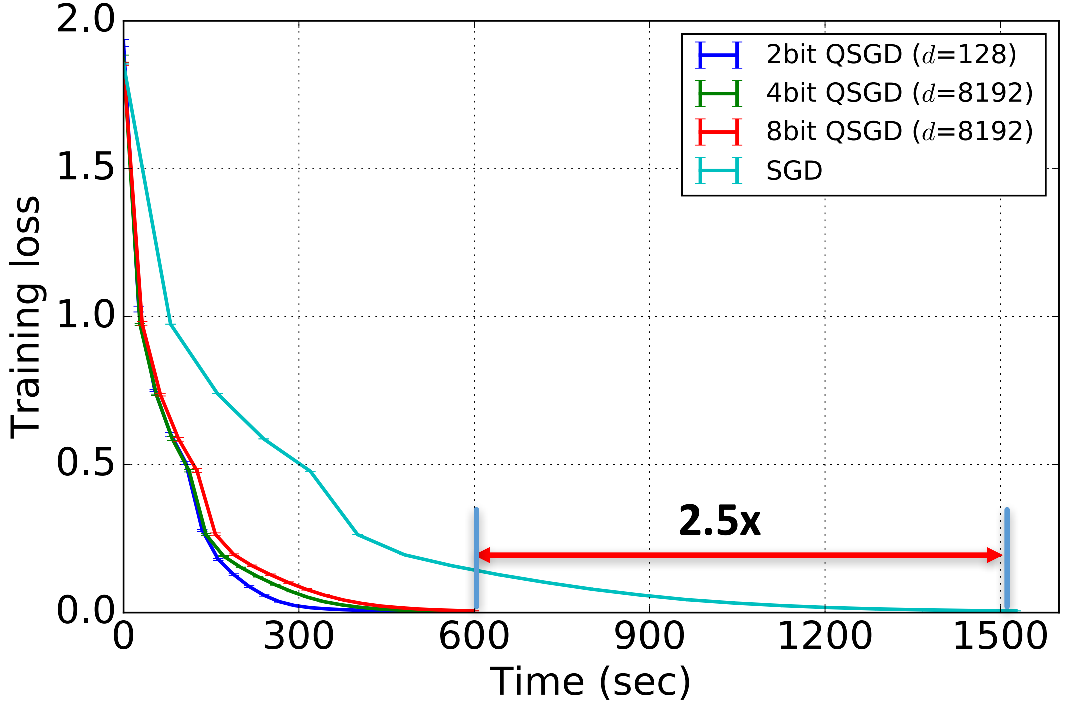




Experiments: Accuracy



ResNet50 on ImageNet
8 GPU nodes



3-Layer LSTM on CMU AN4 (Speech)
2 GPU Nodes

Across all networks we tried, 4 bits are sufficient.
(QSGD report contains full numbers and comparisons.)

Other Communication-Efficient Approaches

Quantization-based methods yield stable, but limited gains in practice

- Usually **< 32x compression**, since it's just **bit width reduction**
- Can't do much better without large variance [QSGD, NIPS17]

The "Engineering" approach [NVIDIA NCCL]

- Increase network bandwidth, decrease network latency
- New interconnects (NVIDIA, CRAY), better protocols (NVIDIA)

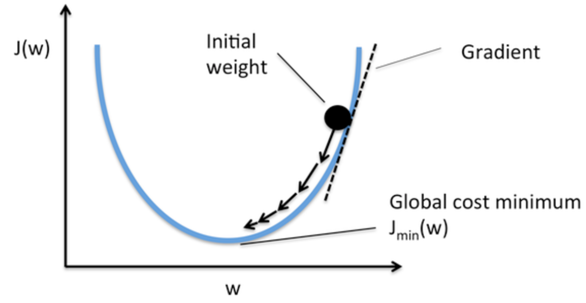
The "Sparsification" approach [Dryden et al., 2016; Aji et al., 2018]

- **Send the "important" components of each gradient, sorted by magnitude**
- Empirically gives much higher compression (up to **800x** [Han et al., ICLR 2018])

"Large-Batch" approaches [Goyal et al., 2017; You et al., 2018]

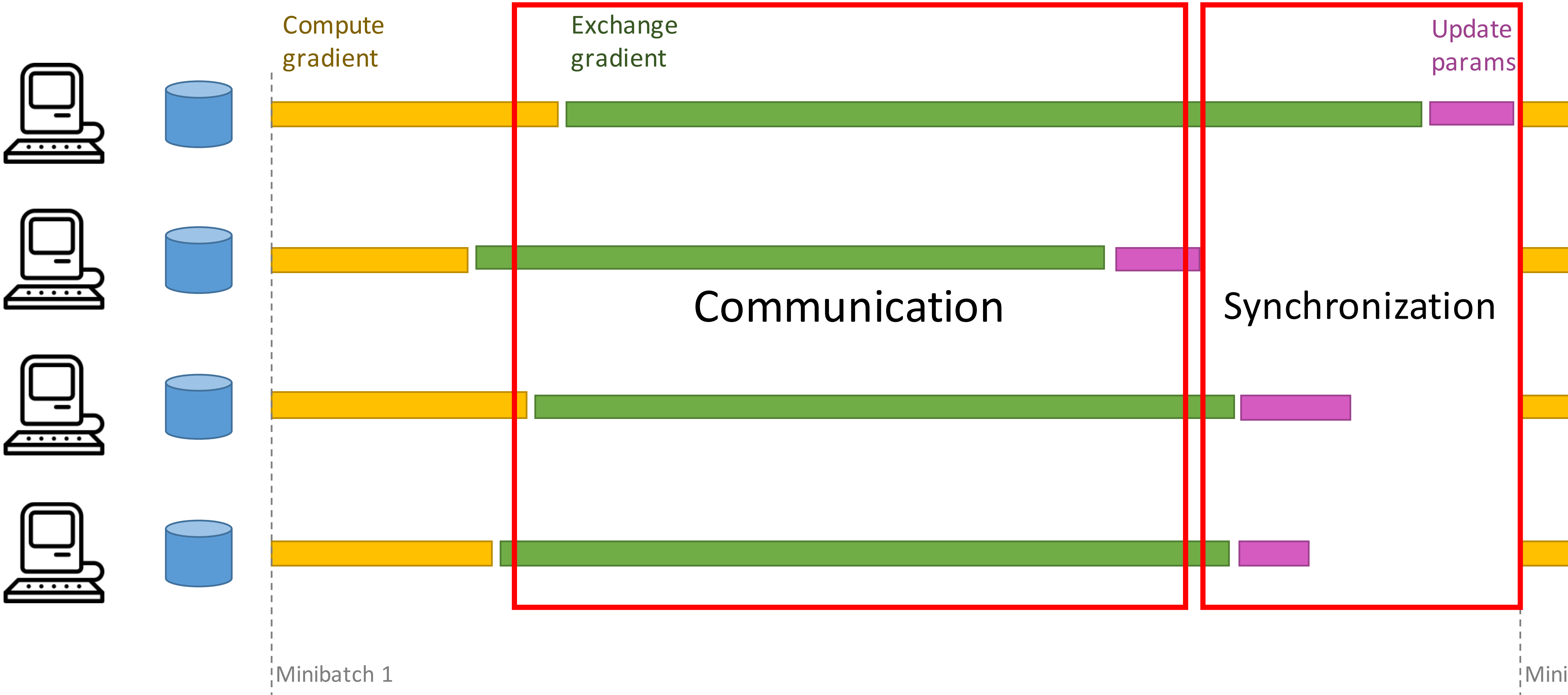
- **Run more computation locally before communicating (large "batches")**
- **Need extremely careful parameter tuning in order to work without accuracy loss**

Roadmap

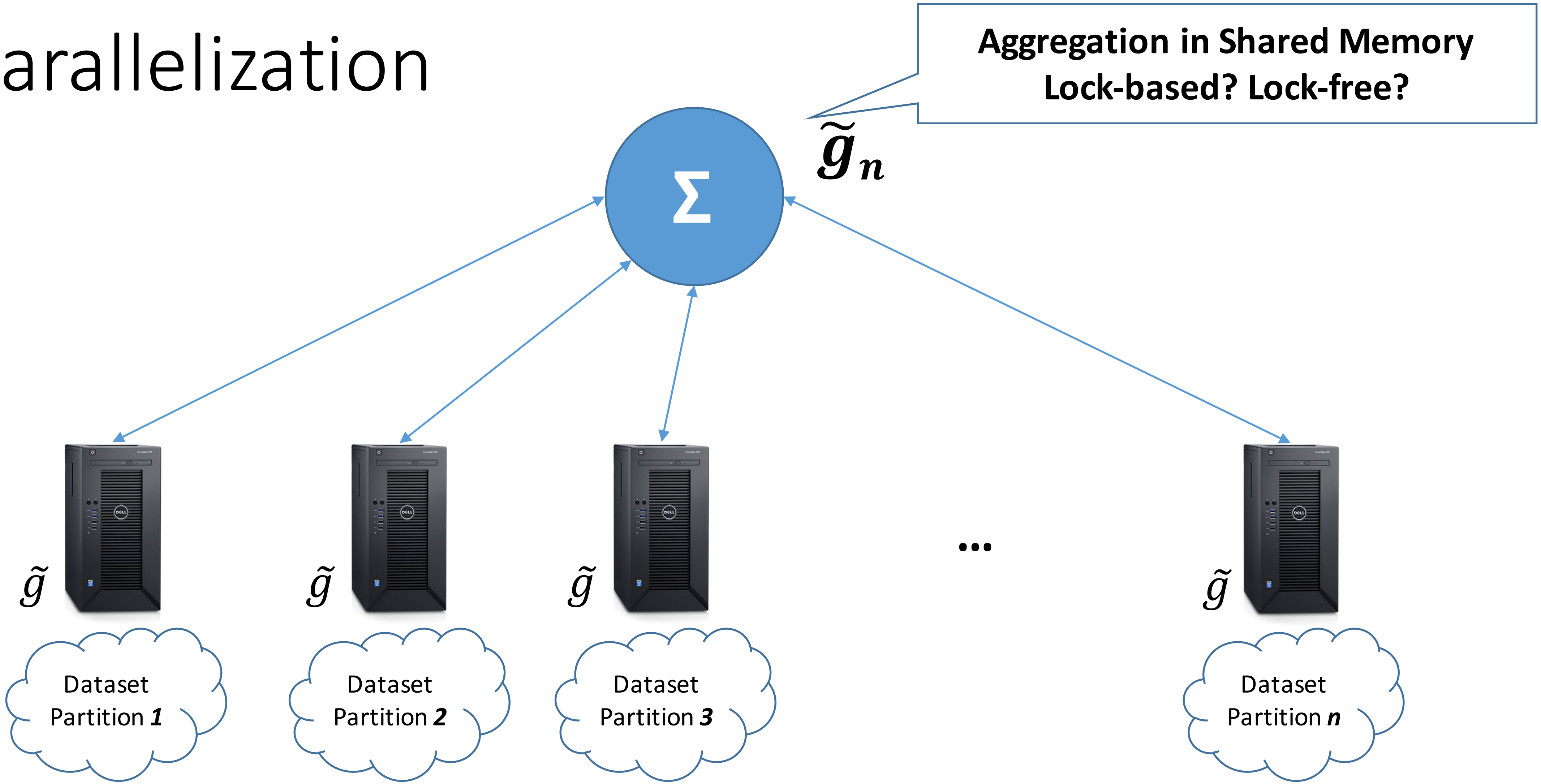


- **Introduction**
- **Basics**
  - **Distributed Optimization and SGD**
- **Communication-Reduction**
  - **Stochastic Quantization and QSGD**
- Asynchronous Training
  - **Asynchronous SGD**
- **Recent Work**

Two major costs



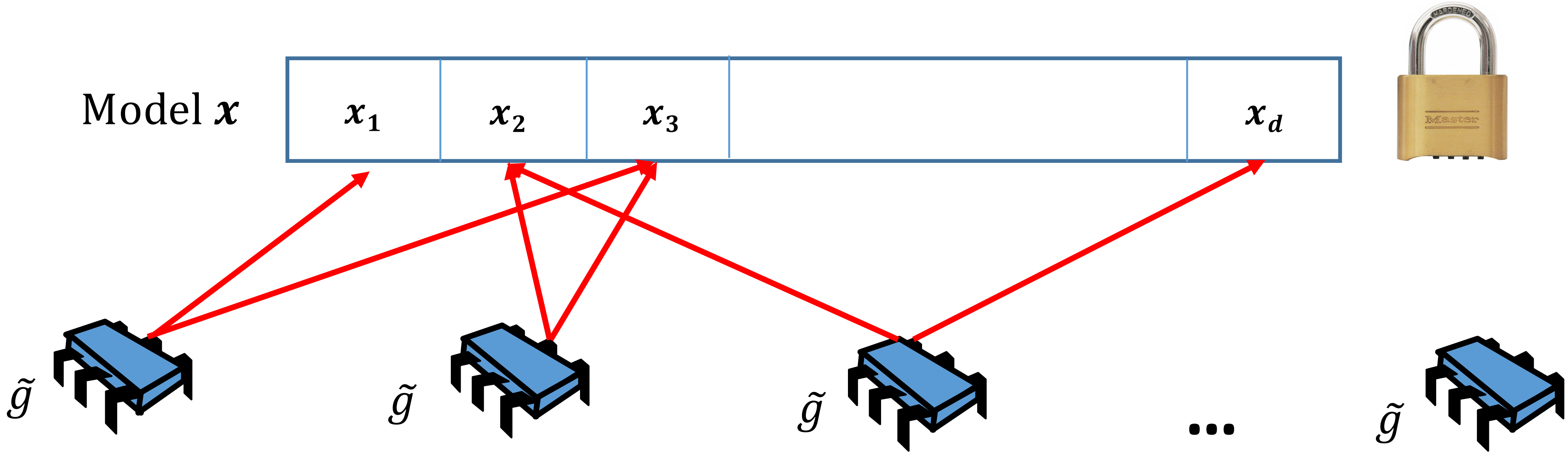
SGD Parallelization



SGD in Asynchronous Shared Memory

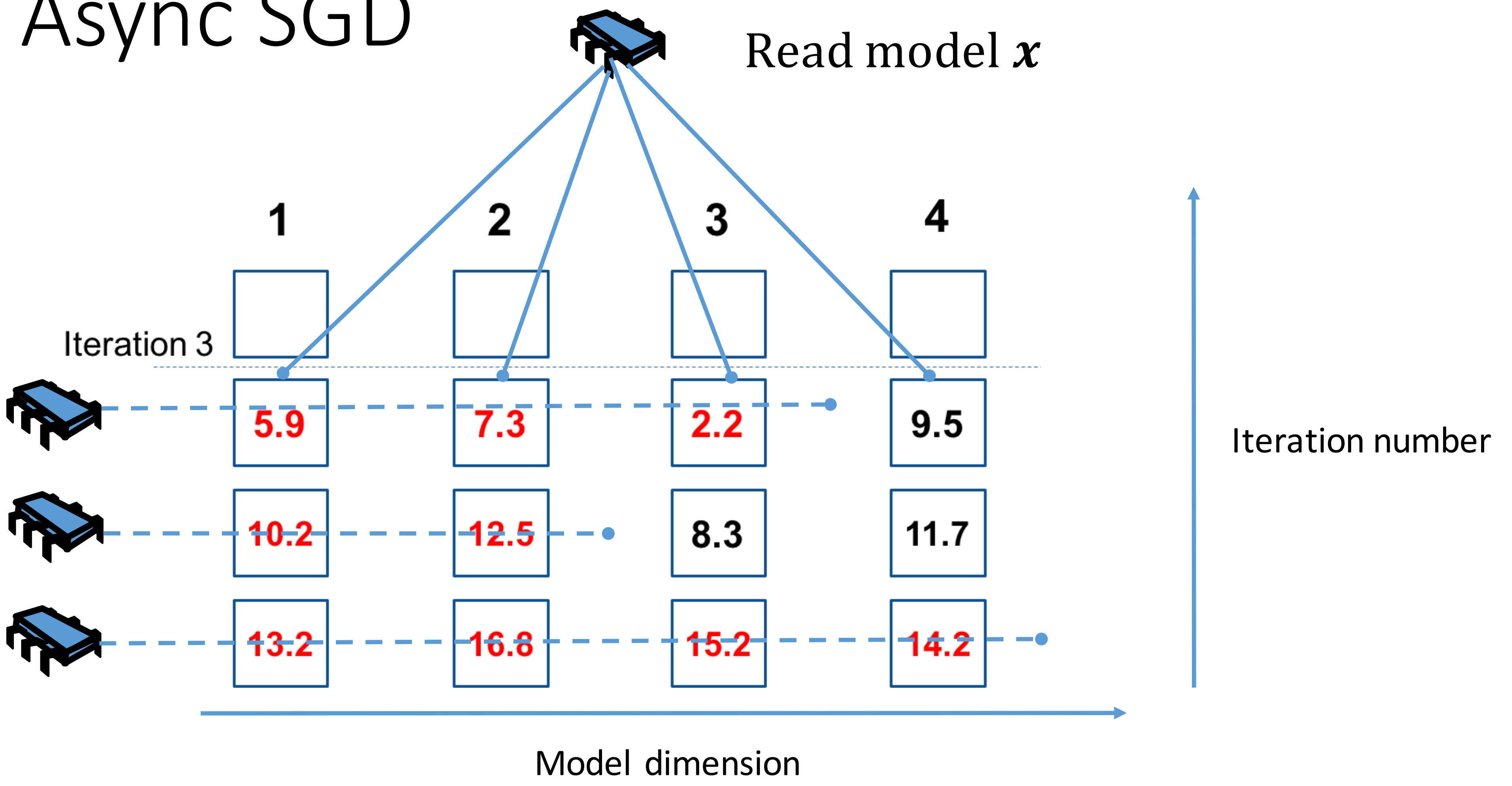
P threads, adversarial scheduler

- Model updated using atomic operations (read, CAS/Fetch-and-add)



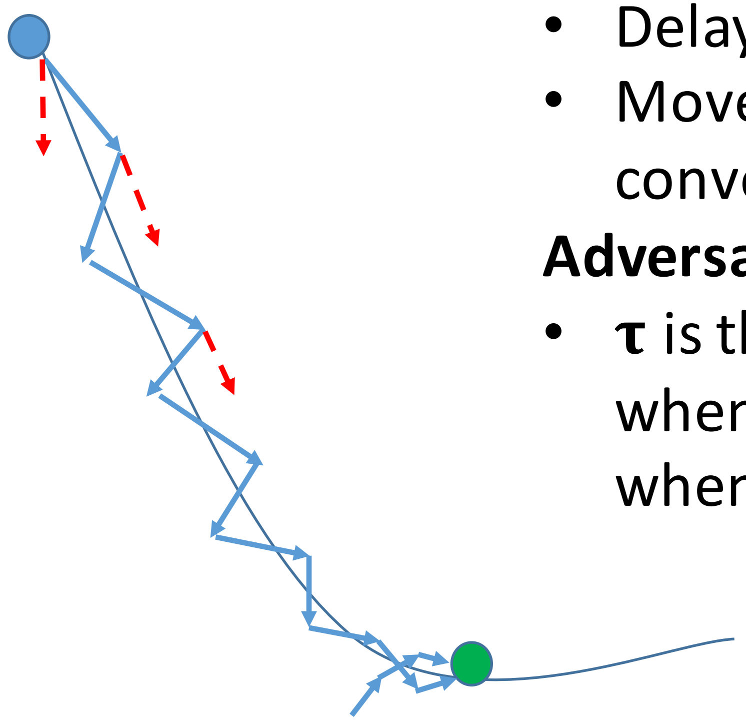
Does SGD still converge under asynchronous (inconsistent) iterations?

Modeling Async SGD



Define **τ** = **maximum** number of previous updates a scan **may miss**.
Note that **τ** ≤ maximum **interval contention** for an operation.

Convergence Intuition



Legend:

- **Blue** = original minibatch SGD
- **Red** dotted = delayed updates

Adversary's power:

- Delay a subset of gradient updates
- Move the delayed updates to delay convergence to the **optimum**

Adversary's limitation:

- **τ** is the **maximum delay** between when the step is generated and when it has to be applied

Theorem [Recht et al., '11]: Under analytic assumptions, asynchronous SGD still converges, but at a rate that is **O (τ)** times slower than serial SGD.

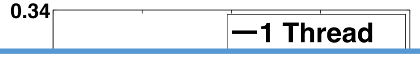
Convergence of Asynchronous SGD (**“Hogwild”**)

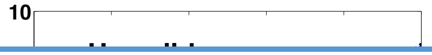
<u>Theorem</u> [Recht et al., ‘11]: Under analytic assumptions, asynchronous SGD still converges, but at a rate that is **O (τ)** times slower than serial SGD.

Lots of follow-up work, tightening assumptions.

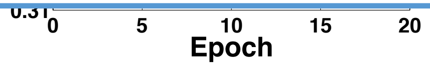
The linear dependency on **τ** is **tight** in general, but can be reduced to $\sqrt{P\tau}$ by simple modifications [PODC18].

This is a **worst-case bound**: in practice, asynchronous SGD sometimes converges at **the same rate** as the serial version.





Theoretical gains come from the fact that the **τ** slowdown due to async is compensated by the speedup of **P** due to parallelism**.**



More details in Nikola’s talk on Wednesday morning!

Recht et al. “HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent”, NIPS 2011

Asynchronous Approaches

The Convex Case:

- By now, lock-free is the standard implementation of SGD in shared memory
- Exploit the fact that many large datasets are sparse, so conflicts are rare
- **NUMA** case is much less well understood

The Non-Convex Case:

- Requires careful hyperparameter tuning to work, and is less popular
- Convergence of SGD in the non-convex case is less well understood, and very little is known analytically [Lian et al, NIPS 2015]

Lian et al: Asynchronous Parallel Stochastic Gradient for Nonconvex Optimization, NIPS 2015

Summary

Most medium-to-large-scale machine learning is **distributed**.

Communication-efficient and **asynchronous learning techniques** are fairly common, and are starting to have a **sound theoretical basis**.

Lots of exciting new questions!

A Sample of Open Questions

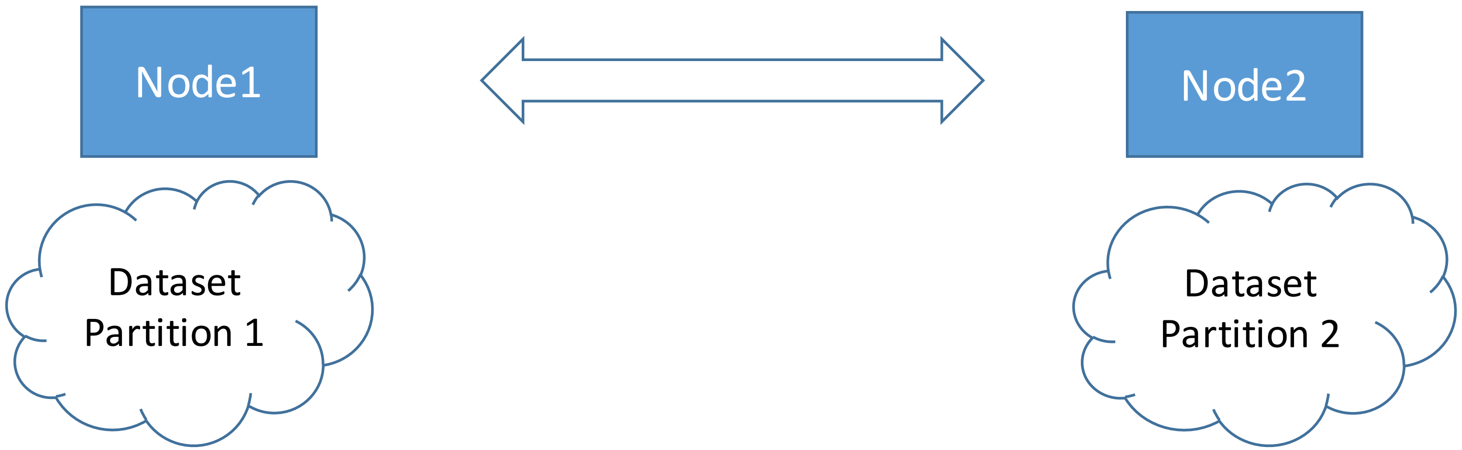
What are the notions of ***consistency*** required by distributed machine learning algorithms in order to converge?

At first sight, *much weaker* than standard notions .

$$\operatorname{argmin}_{\boldsymbol{x}} f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$$

model $\boldsymbol{x_1} = \boldsymbol{x} + \boldsymbol{noise}$

model $\boldsymbol{x_1} = \boldsymbol{x} + \boldsymbol{noise}$

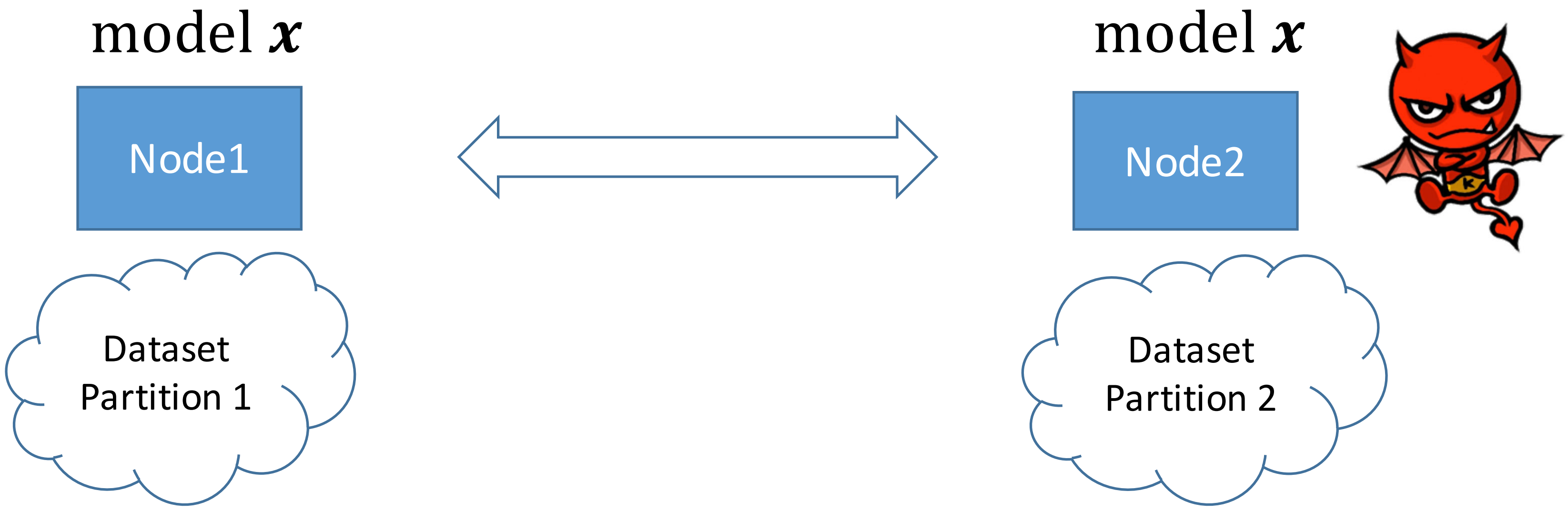


A Sample of Open Questions

Can distributed Machine Learning algorithms be **Byzantine-resilient**?
Early work by **[Su, Vaidya]**, **[Blanchard, El Mhamdi, Guerraoui, Steiner]**

Non-trivial ideas from both ML and distributed computing sides.

$$\mathrm{argmin}_{\boldsymbol{x}} f(\boldsymbol{x}) = f_1(\boldsymbol{x_1}) + f_2(\boldsymbol{x_2})$$



A Sample of Open Questions

Can distributed Machine Learning algorithms be **completely decentralized**?
Early work by e.g. [Lian et al., NIPS 2017], for SGD.