Quantized Stochastic Gradient Descent

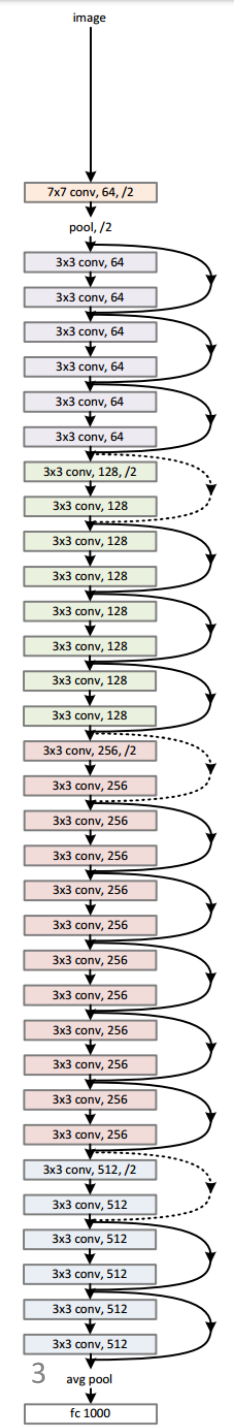
Dan Alistarh

ETH Zurich

The Practical Problem

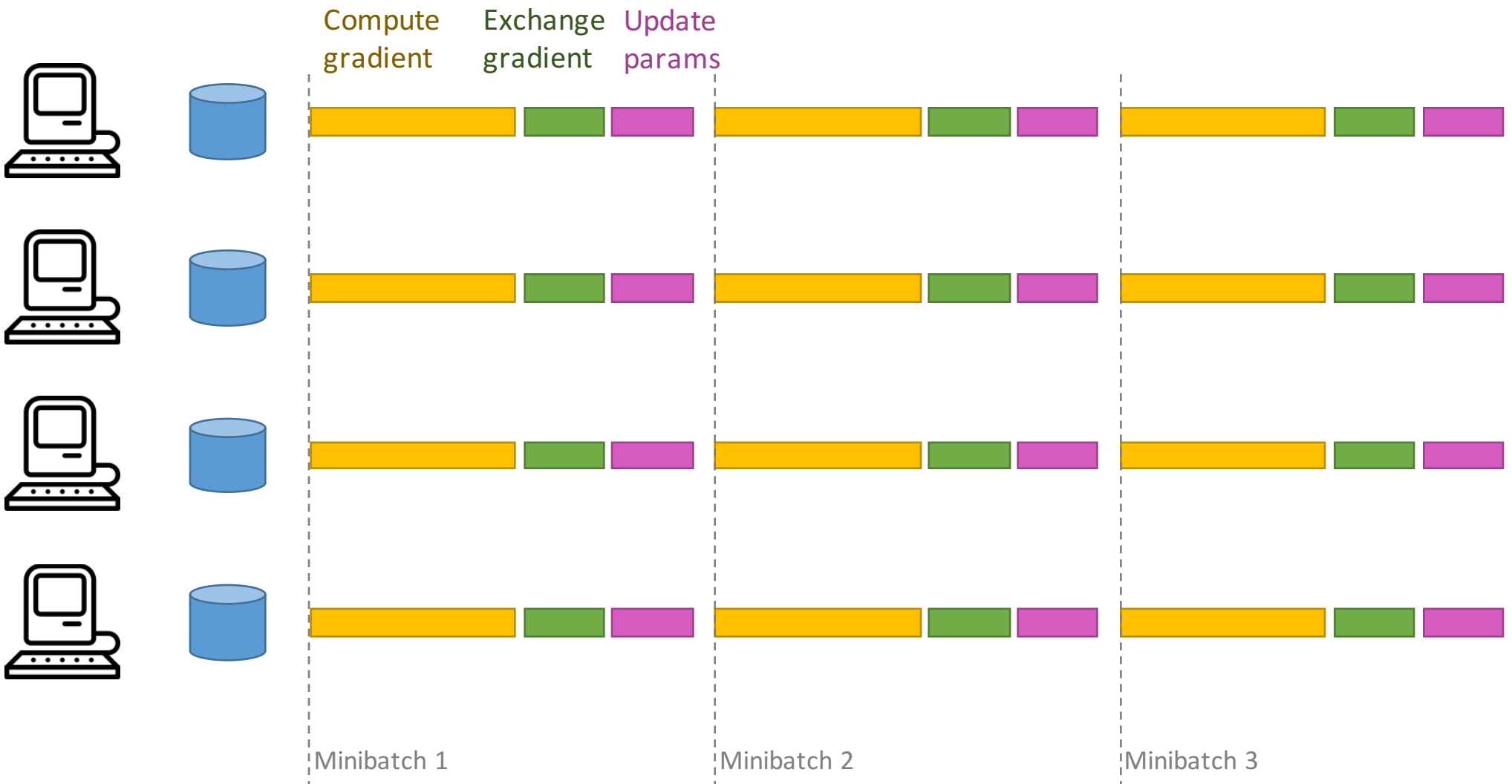
Training large machine learning models efficiently

- **Large Datasets**:
  - ImageNet: **1.6 million images (~300GB)**
  - NIST2000 Switchboard dataset: **2000 hours**
- **Large Models**:
  - **ResNet-152** [He et al. 2015]: 152 layers**, 60 million parameters**
  - **LACEA** [Yu et al. 2016]: 22 layers, **65 million parameters**

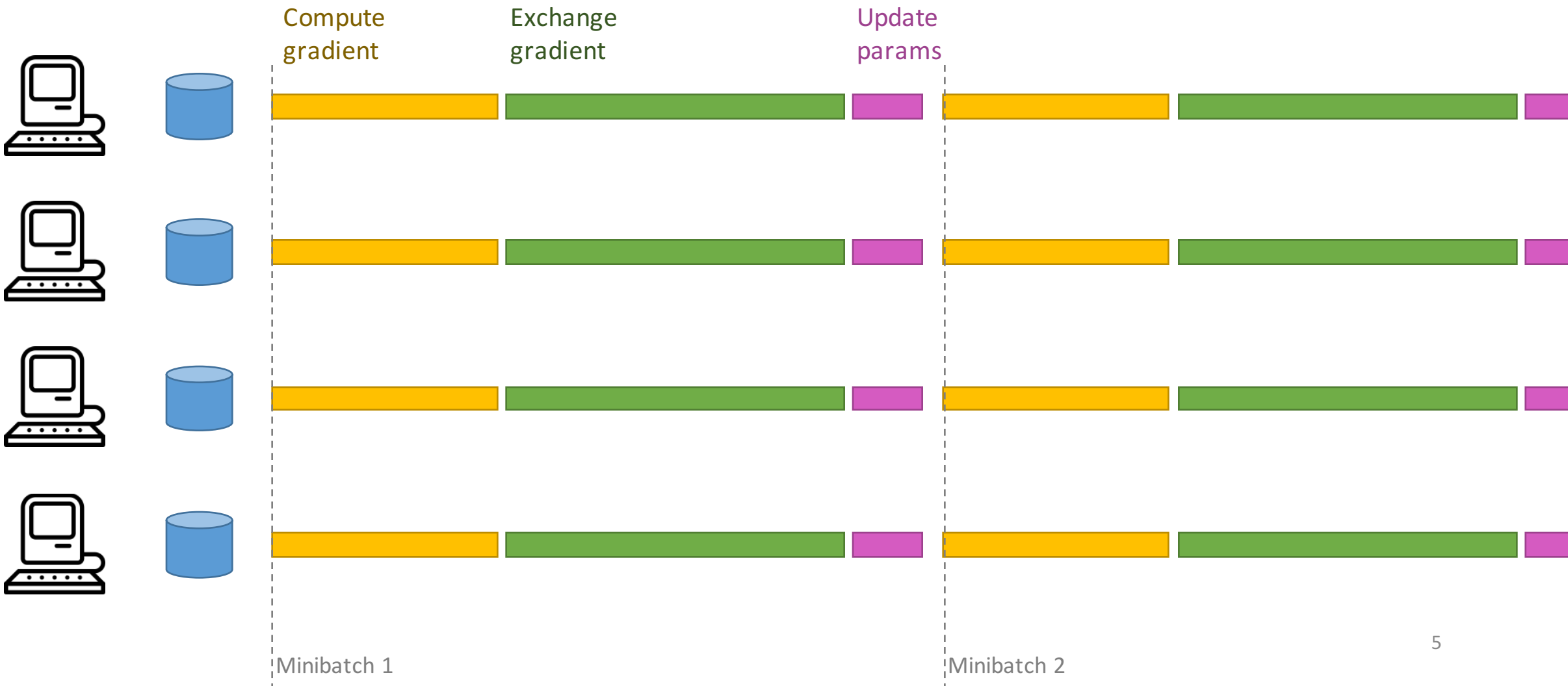


He et al. (2015) “Deep Residual Learning for Image Recognition”
Yu et al. (2016) “Deep convolutional neural networks with layer-wise context expansion and attention”

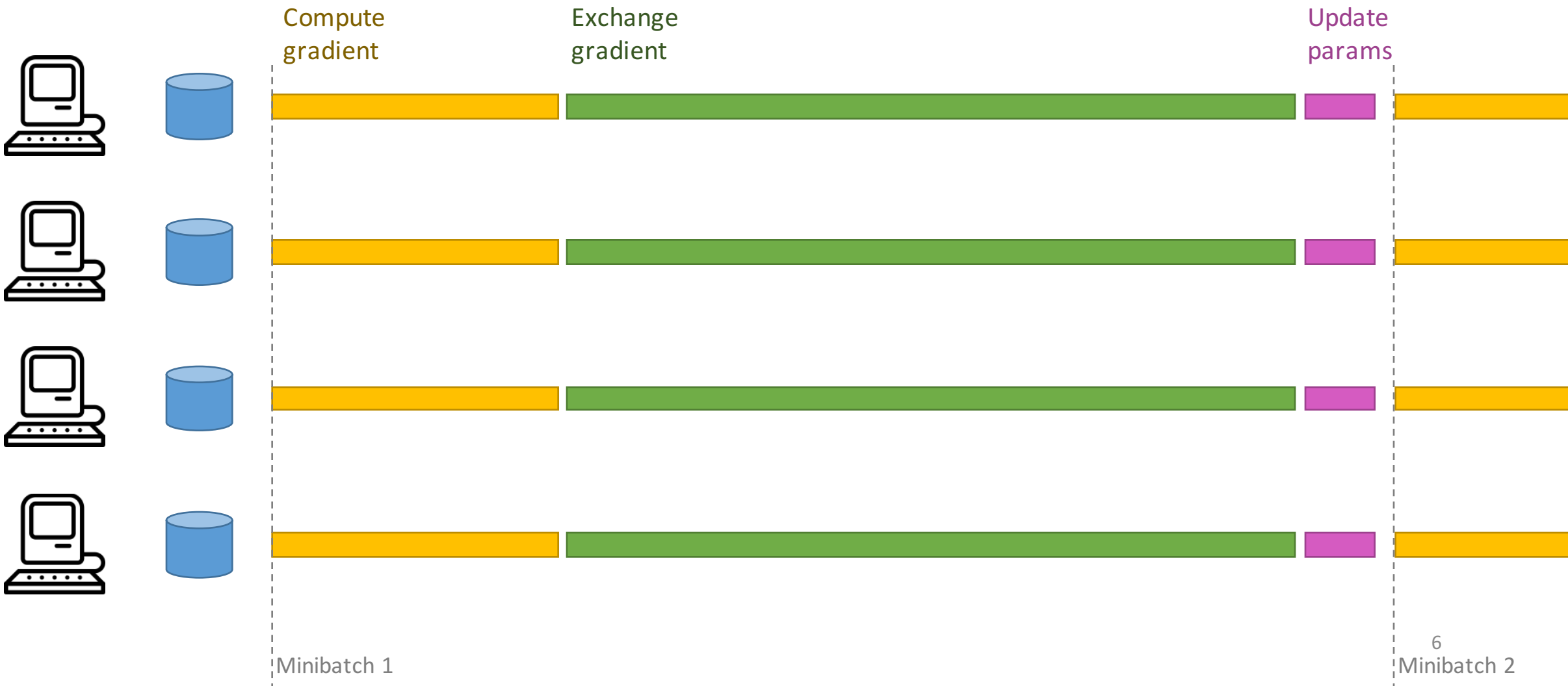
Data-parallel Stochastic Gradient Descent



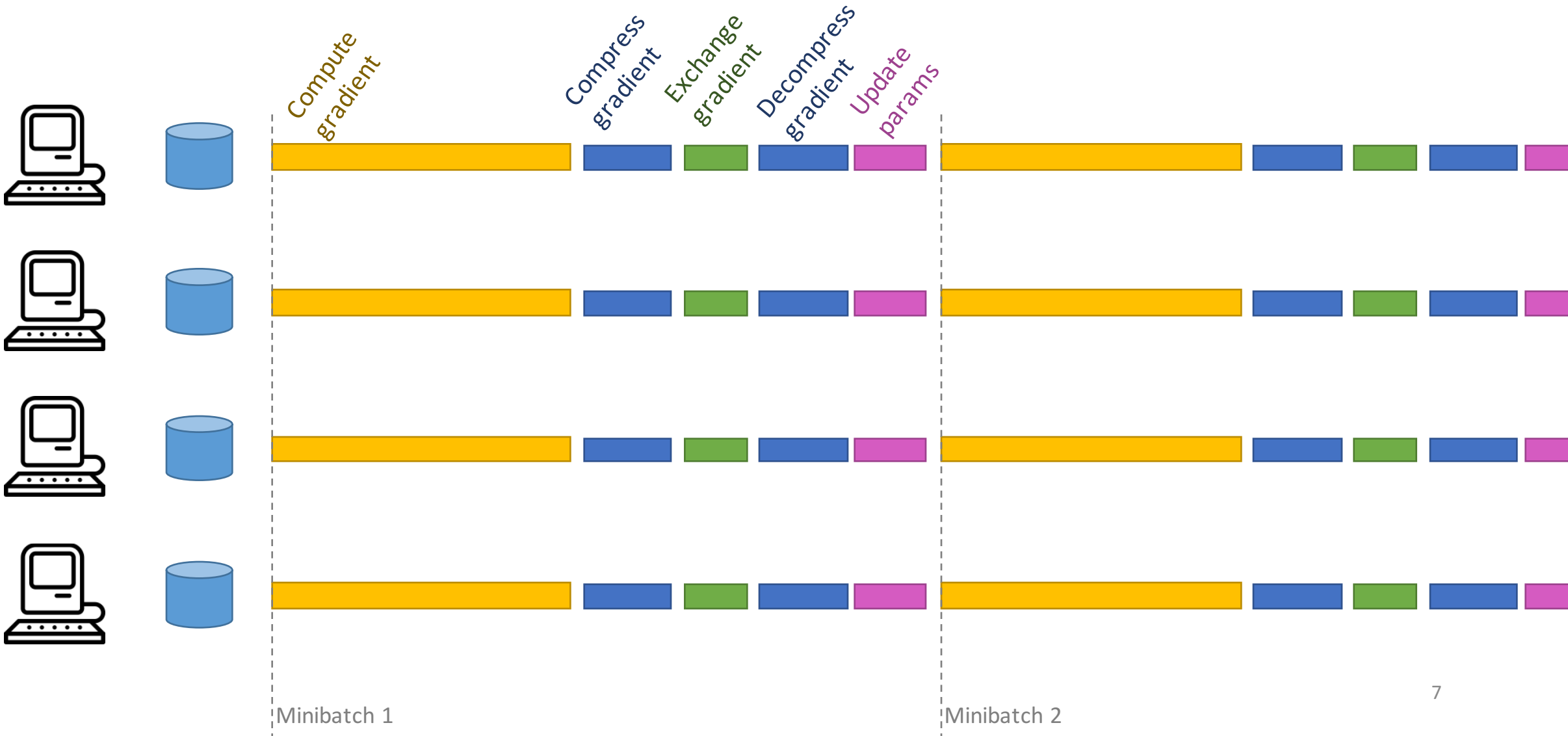
Data parallel SGD (**bigger models**)



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



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



Background on SGD

- Start from the ***gradient descent*** iteration

$$\boldsymbol{x_{t+1} = x_t - \eta_t \nabla f(x_t).}$$

- Let $\boldsymbol{\tilde{g}(xt)}$ = gradient at ***randomly chosen*** data point.

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

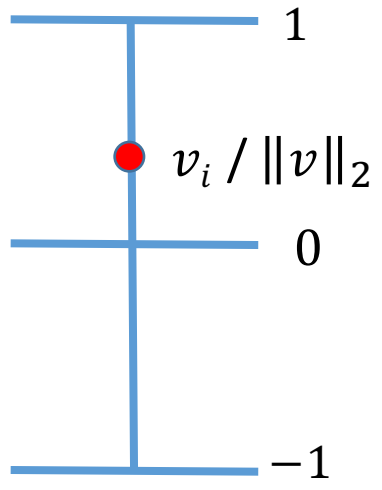
- Let $\boldsymbol{E\left[\left|\left|\tilde{g}(x) - \nabla f(x)\right|\right|^2\right] \leq \sigma^2}$ (variance bound)

<u>Theorem</u> (standard)**:** Given f **convex** and **smooth**, and $R^2 = ||x_0 - x^*||^2$.

If we run SGD for $\boldsymbol{T = \mathcal{O}\left(R^2 \frac{2\sigma^2}{\varepsilon^2}\right)}$ iterations, then

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

Basic Idea: **QSGD** [Alistarh et al., NIPS17]



- Quantization function

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

where $\xi_i(v_i) = \mathbf{1}$ with probability $|\boldsymbol{v_i}|/\|\boldsymbol{v}\|_2$ and **0**, otherwise.

Properties:

1. **Unbiasedness**:

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

Convergence: $\boldsymbol{E}[Q(\boldsymbol{v})] = \boldsymbol{v}$.

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

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

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

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

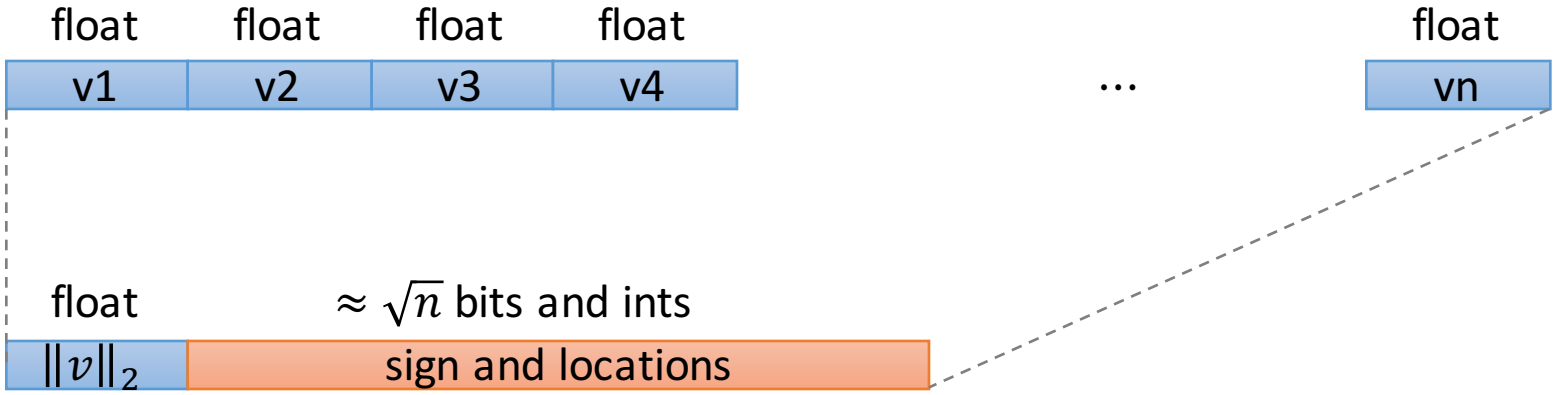
$$\boldsymbol{E}[\textbf{non-zeroes in } \boldsymbol{Q(v)}\ ] = \boldsymbol{E}[\textstyle\sum_i \boldsymbol{\xi_i}(\boldsymbol{v})] \leq \|\boldsymbol{v}\|_1/\|\boldsymbol{v}\|_2 \leq \sqrt{\boldsymbol{n}}$$

QSGD Compression

- Quantization function

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

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



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

QSGD Compression

- Quantization function

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

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

Moral: We're not too happy, since **the $\sqrt{n}$ increase in number of iterations is quite large**.

In the paper: We reduce $\sqrt{n}$ to a **small constant (2)** by increasing the **number of quantization levels**.

In practice, 4-8 bits is enough.

Experimental Setup

Where?

- **Amazon p16xLarge** (**16 x NVIDIA K80 GPUs**), with **NVIDIA DirectConnect**
- Microsoft **CNTK v2.0**, with fast **MPI-based communication**

What?

- **Tasks: image classification** (ImageNet, CIFAR) and **speech recognition** (CMU AN4)
- **Nets**: **ResNet**, **VGG**, **AlexNet ,** respectively **LSTM**, with **default parameters**

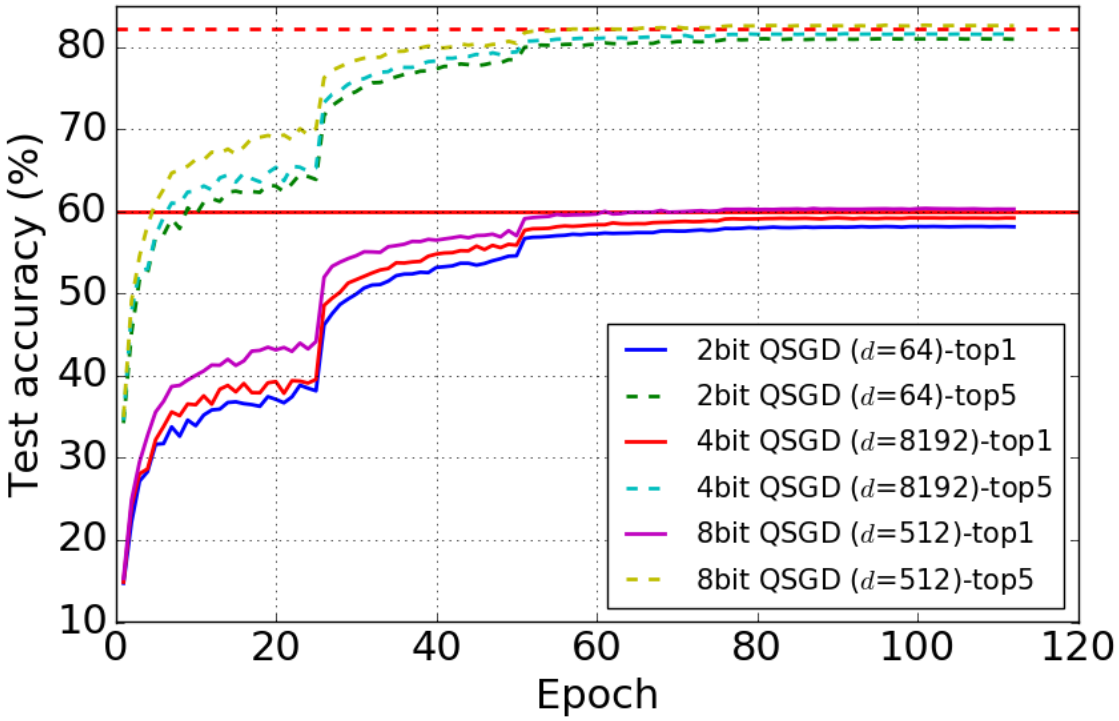
Why?

- **Accuracy vs. Speed/Scalability**

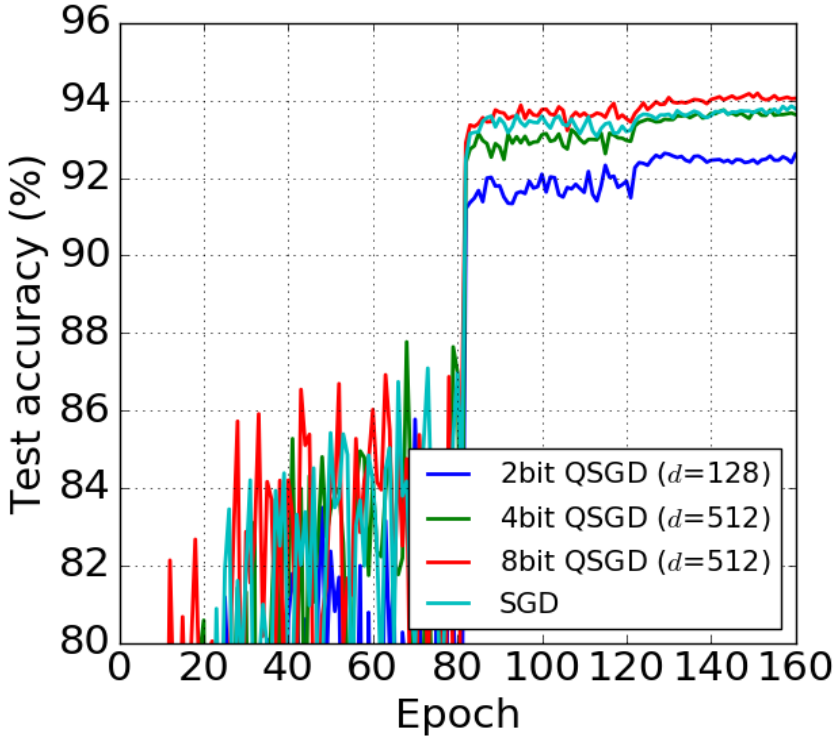
How long?

- **>25K GPU hours**

Experiments: Accuracy



AlexNet on ImageNet



ResNet-110 on CIFAR-10

More Accuracy: please see paper

AlexNet ImageNet

Setup		Accuracy			Samples per second (MPI)				
Precision	Bucket size	56	89	112	1 GPU	2 GPUs	4 GPUs	8 GPUs	16 GPUs
32bit	/	/	/	59.90%	240.80	301.45	328.00	272.90	192.10
QSGD 16bit	8192	/	/	/	/	388.80	508.80	500.90	335.60
QSGD 8bit	512	59.63%	60.36%	60.37%	/	424.90	544.60	739.10	535.00
QSGD 4bit	512	59.20%	59.97%	60.05%	/	466.50	598.70	964.90	748.50
QSGD 2bit	128	57.09%	58.09%	58.17%	/	449.20	609.15	1076.50	889.80
1bitSGD	/	59.57%	60.23%	60.31%	/	424.05	564.30	971.10	849.40
1bitSGD*	64	59.26%	59.92%	59.99%	/	370.80	476.50	761.20	712.70
1bitSGD*	512	58.44%	59.25%	59.29%	/	/	/	/	/

ResNet50 ImageNet

Setup		Accuracy			Samples per second (MPI)				
Precision	Bucket size	60	96	120	1 GPU	2 GPUs	4 GPUs	8 GPUs	16 GPUs
32bit	/	70.75%	74.60%	74.68%	47.20	80.80	142.40	247.90	272.30
QSGD 16bit	8192	/	/	/	/	90.20	156.30	275.80	348.70
QSGD 8bit	512	70.66%	75.07%	75.19%	/	92.60	162.70	313.70	416.80
QSGD 4bit	512	70.66%	74.62%	74.76%	/	93.90	165.70	326.10	461.20
QSGD 2bit	128	/	/	/	/	93.30	178.35	330.45	472.25
1bitSGD	/	/	/	/	/	45.10	81.70	160.15	155.20
1bitSGD*	64	70.78%	74.93%	75.02%	/	88.10	156.50	296.70	442.40

ResNet110 CIFAR10

Setup		Accuracy			Samples per second (MPI)				
Precision	Bucket size	80	128	160	1 GPU	2 GPUs	4 GPUs	8 GPUs	16 GPUs
32bit	/	87.09%	93.65%	93.86%	343.70	555.00	957.70	1229.10	831.60
QSGD 16bit	8192	/	/	/	/	551.00	942.70	1164.20	763.40
QSGD 8bit	512	86.92%	93.97%	94.19%	/	550.20	960.10	1193.10	759.70
QSGD 4bit	512	87.77%	93.52%	93.76%	/	571.10	957.40	1257.10	784.30
QSGD 2bit	128	85.77%	92.59%	92.64%	/	557.20	973.10	1227.90	780.40
1bitSGD	/	87.13%	94.02%	94.15%	/	465.60	643.30	610.90	406.90
1bitSGD*	64	/	/	/	/	550.40	884.80	1156.70	757.70

ResNet152 ImageNet

Setup		Accuracy			Samples per second (MPI)				
Precision	Bucket size	60	96	120	1 GPU	2 GPUs	4 GPUs	8 GPUs	16 GPUs
32bit	/	/	/	77.00%	16.90	26.10	45.00	73.90	113.50
QSGD 16bit	8192	/	/	/	/	31.20	54.50	95.50	151.00
QSGD 8bit	512	72.95%	76.66%	76.74%	/	32.80	62.70	109.20	182.50
QSGD 4bit	512	/	/	/	/	33.60	60.20	121.90	203.20
QSGD 2bit	128	/	/	/	/	33.50	64.35	123.55	208.50
1bitSGD	/	/	/	/	/	10.55	22.10	41.40	63.15
1bitSGD*	64	/	/	/	/	30.40	55.50	108.10	193.50

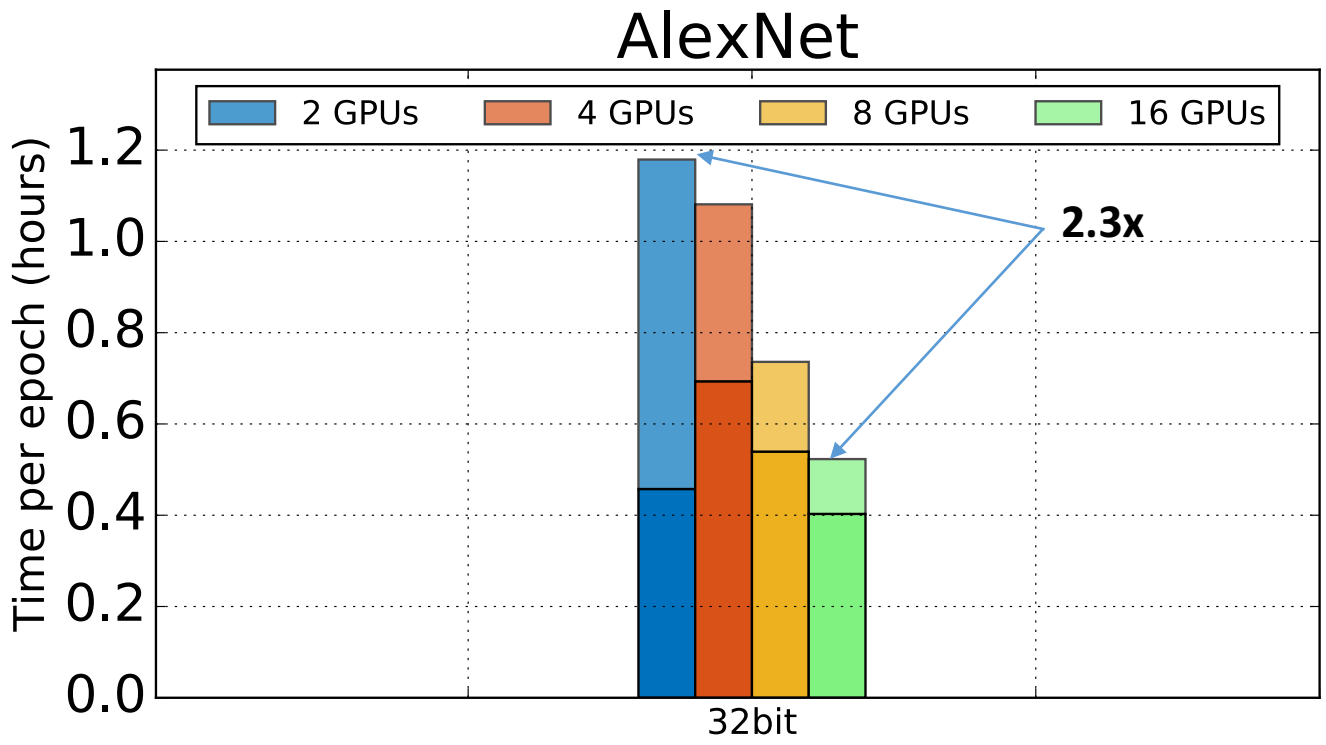
VGG19 ImageNet

Setup		Accuracy			Samples per second (MPI)				
Precision	Bucket size	40	64	80	1 GPU	2 GPUs	4 GPUs	8 GPUs	16 GPUs
32bit	/	/	/	/	12.40	20.40	36.30	53.95	40.60
QSGD 16bit	8192	/	/	/	/	24.80	46.40	35.80	67.80
QSGD 8bit	512	/	/	/	/	24.20	47.50	119.50	106.60
QSGD 4bit	512	/	/	/	/	27.00	52.30	151.65	143.80
QSGD 2bit	128	/	/	/	/	24.60	49.35	160.35	170.50
1bitSGD	/	/	/	/	/	22.20	43.15	117.35	120.60
1bitSGD*	64	/	/	/	/	22.90	44.80	99.15	134.30

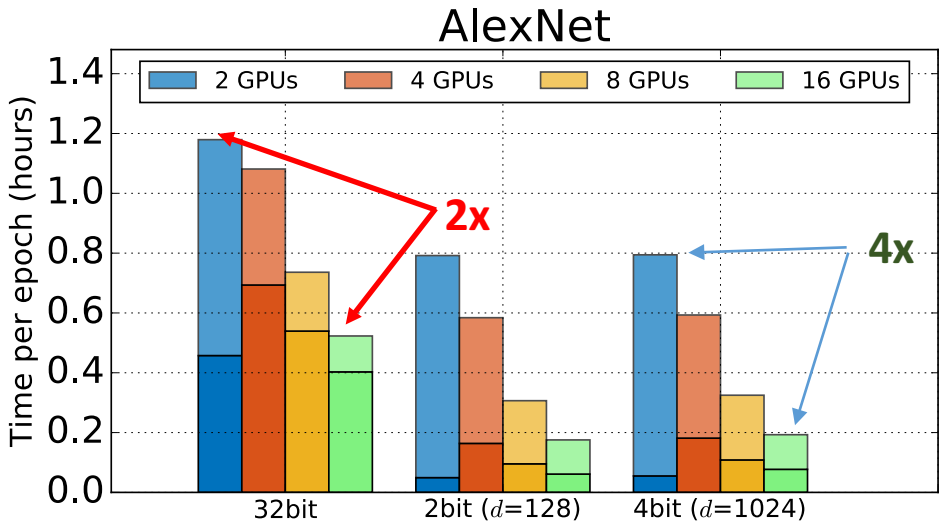
Generally, 4bits are sufficient for recovering or improving accuracy at same rate (also theoretically sound).

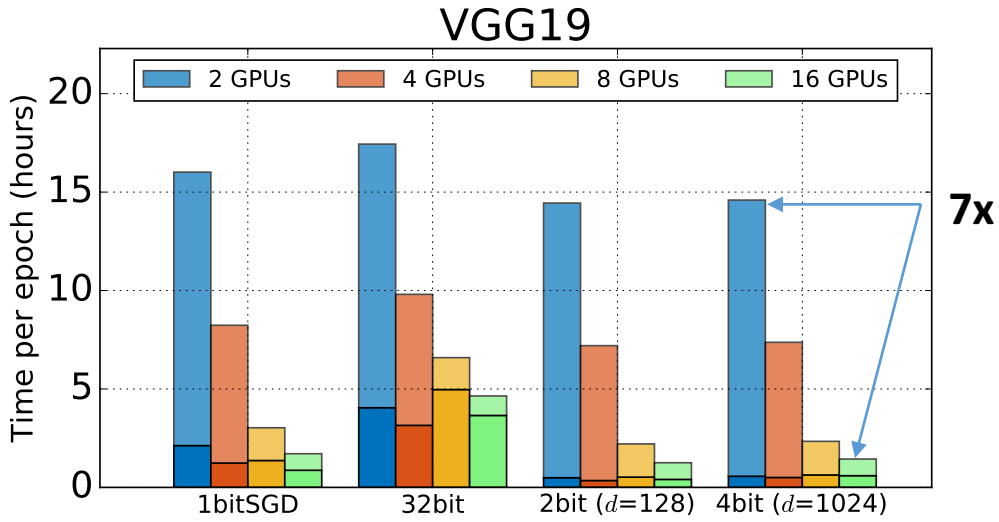
Experiments: Scalability

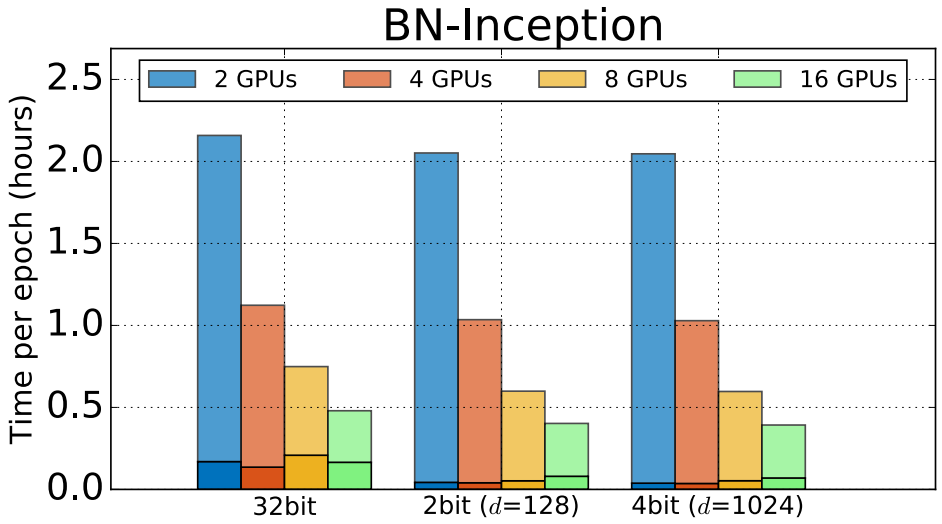
- All experiments ran on ImageNet
- Amazon baseline:

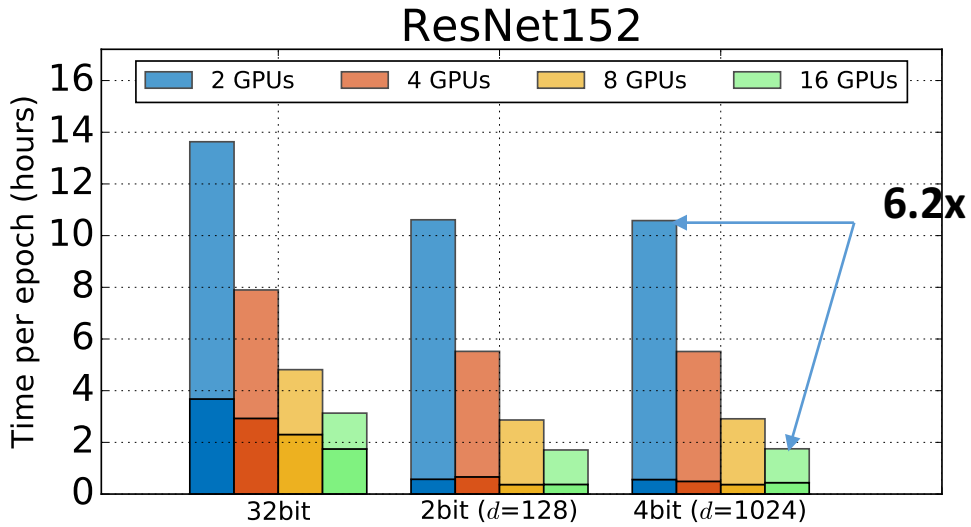


Experiments: Scalability



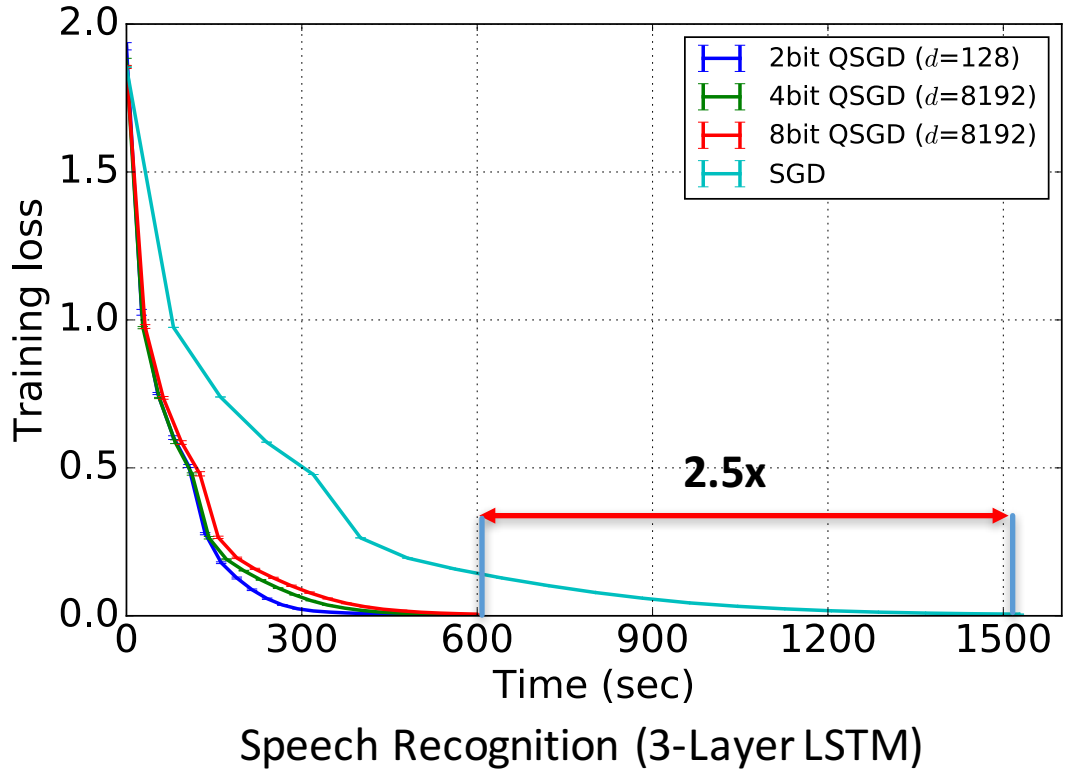






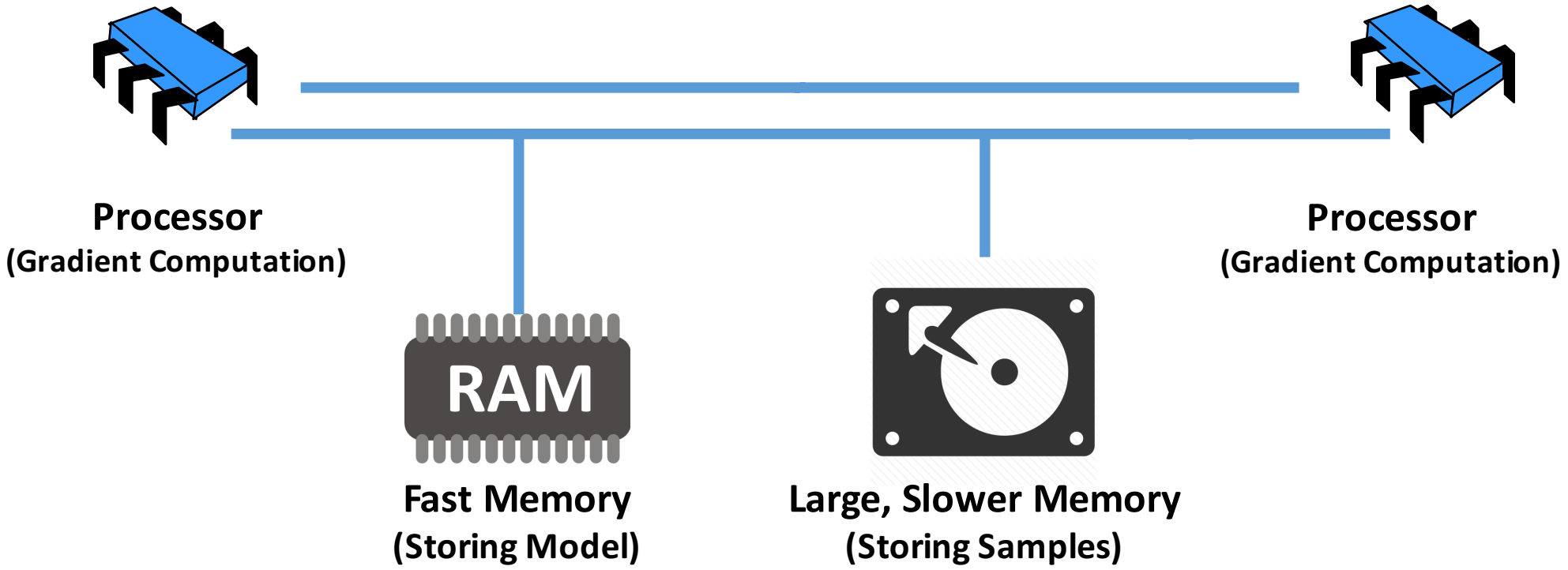
Accuracy vs Time on LSTM

- **AN4 dataset, 2 x Titan X** GPUs



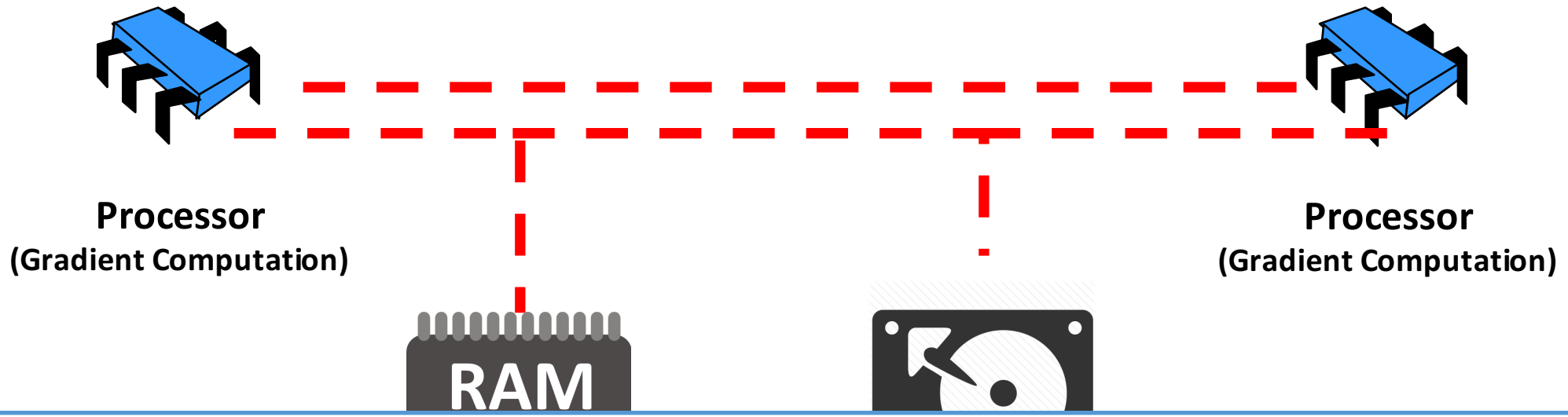
What Else Can We Quantize?

- Iteration: $x_{t+1} = x_t - \eta_t \tilde{g}(xt, et)$



Everything: *Gradient*, *Samples*, and *Model*

- Iteration: $x_{t+1} = x_t - \eta_t \boldsymbol{Q}_g(\tilde{g}(\boldsymbol{Q}_m(xt), \boldsymbol{Q}_s(et)))$



Can we quantize end-to-end and still get convergence guarantees?

ZipML: End-to-end Quantization for Linear Models

[Alistarh, Li, Liu, Kara, Zhang & Zhang, ICML 2017]

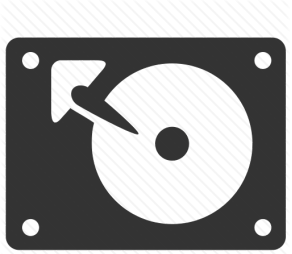
- ***Samples*** are problematic:

  Original gradient: $\tilde{g}(x_t) = at(atTx_t - bt)$

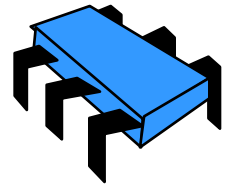
  Naïve quantization: $\tilde{g}(x_t) = \boldsymbol{Q}(\boldsymbol{at})(\boldsymbol{Q}(\boldsymbol{at})^T x_t - bt)$ ← **Not unbiased!**

- We can fix this by ***double sampling***

  $\tilde{g}(x_t) = \boldsymbol{Q}_1(\boldsymbol{at})(\boldsymbol{Q}_2(\boldsymbol{at})^T x_t - bt)$ ← **If samples are independent, the gradient is again unbiased!**



Sample Source

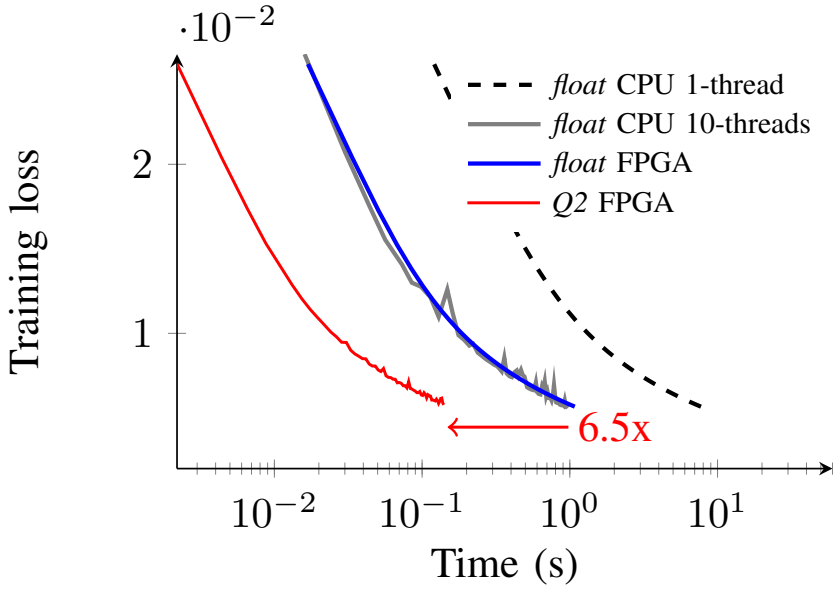


Processor

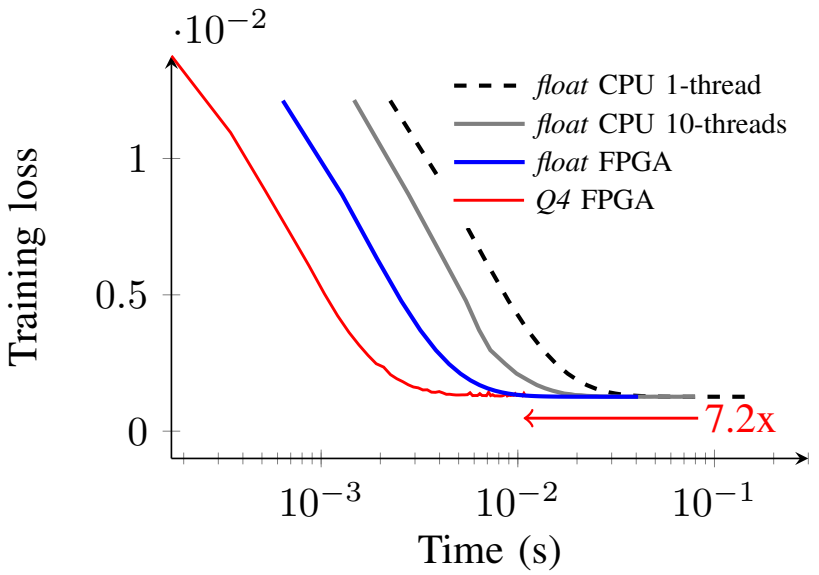
$(at, bt) \rightarrow (Q_1(a_t), Q_2(a_t), bt)$ → $\tilde{g}(x_t)$

Generalization & Implementation

- We can do **end-to-end quantized linear regression** with **guarantees**
- General **classification loss** functions:
  1. For **smooth** losses, we can do **polynomial approximation,** and then **multi-sampling**
  2. For **non-smooth** losses (e.g. **SVM**), we can do "**re-fetch sampling**"
- Implemented on an **Xeon+FPGA platform (Intel-Altera HARP)**
  - **Quantized data -> 8-16x more data per transfer**



Classification (gisette dataset, n = 5K).



Regression (synthetic dataset, n = 100).

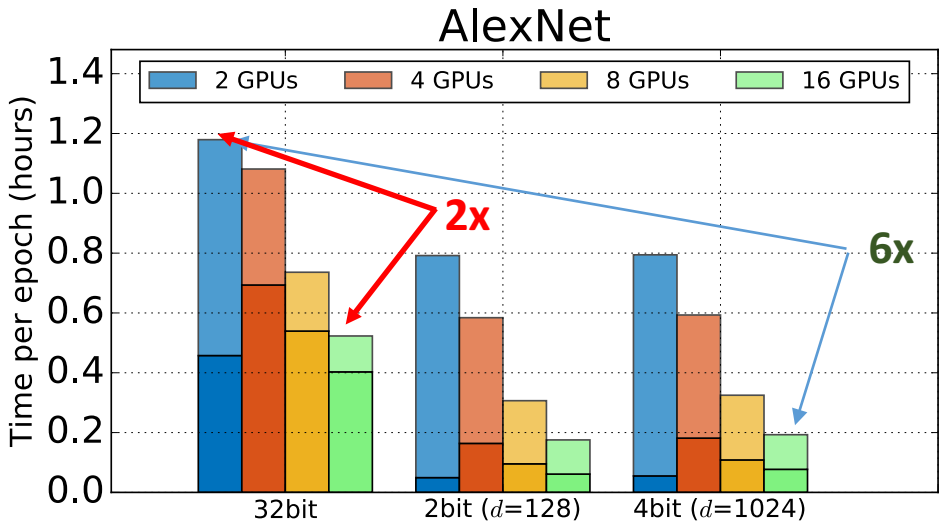
Summary

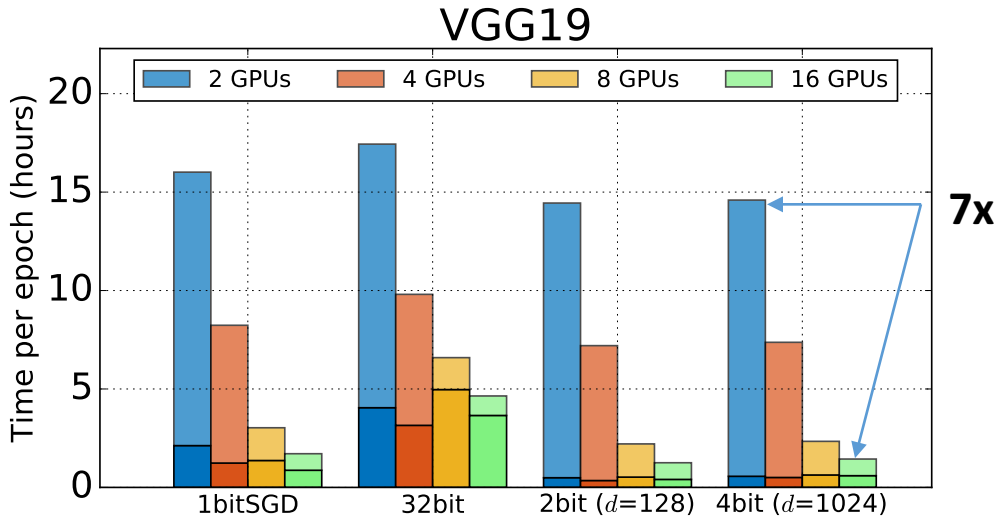
QSGD is a family of **lossy compression schemes** which trade off **low bit complexity** and **convergence time**.

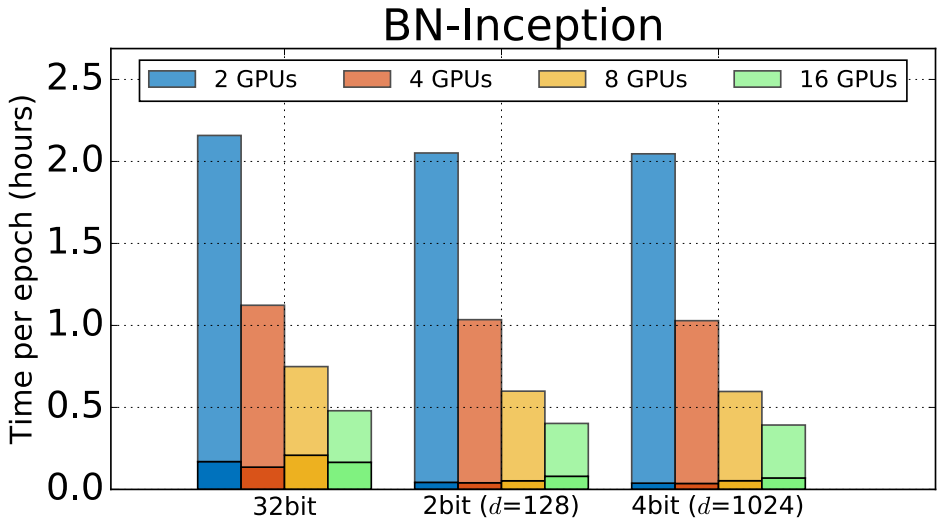
QSGD is **theoretically optimal**, can be **practical**, and extensible to **end-to-end compression**.

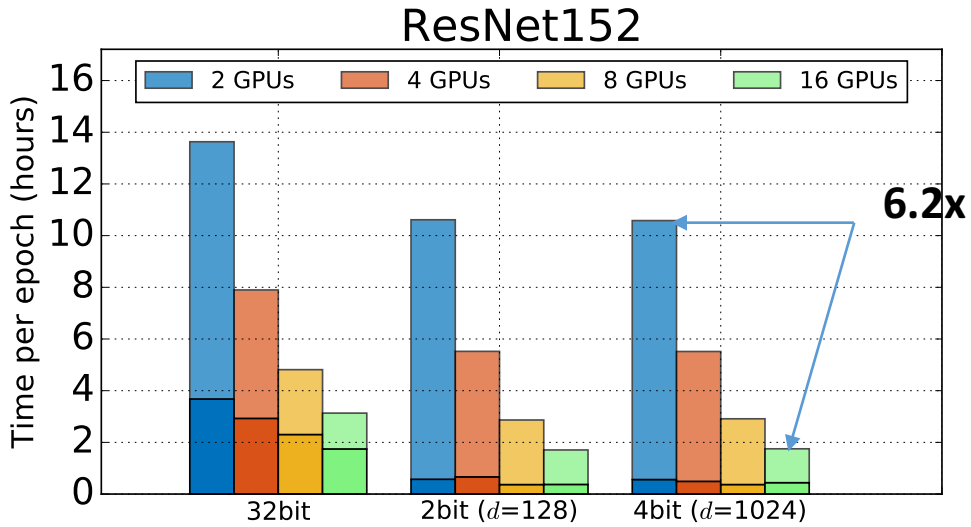
Got Applications?

Experiments: More Scalability

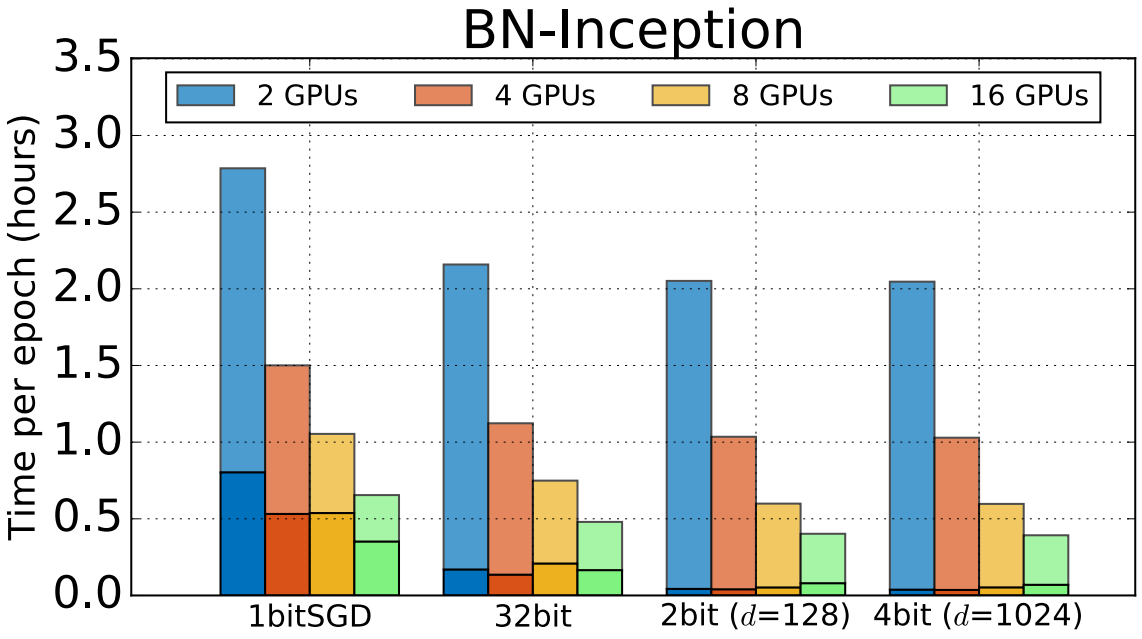


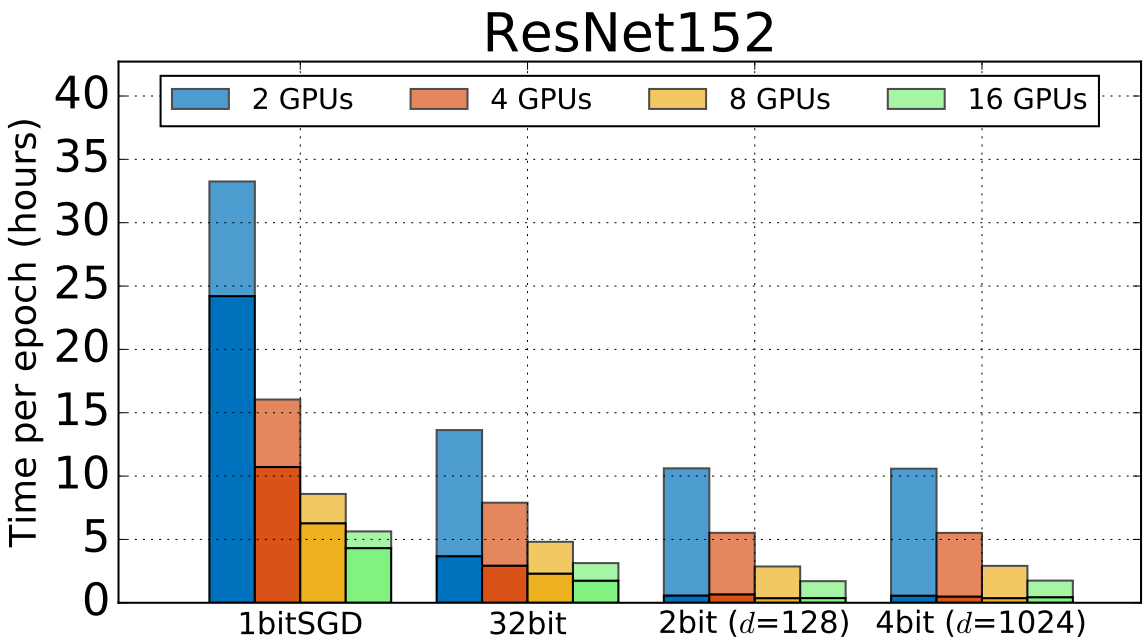


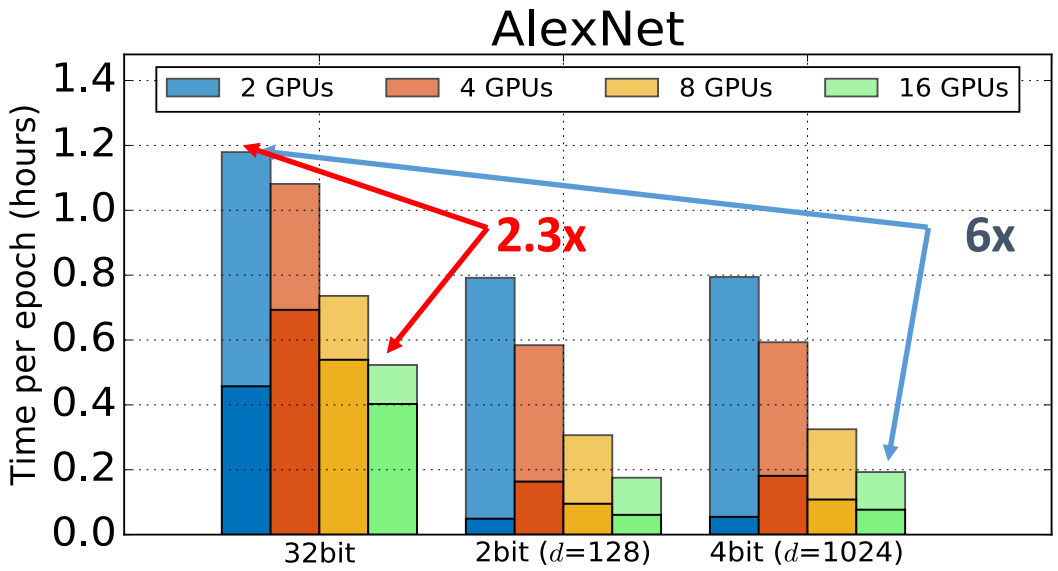




Backup 1: Comparison with 1BitSGD






AlexNet
2 GPUs
4 GPUs
8 GPUs
16 GPUs
Time per epoch (hours)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2.3x
6x
32bit
2bit (d=128)
4bit (d=1024)