MACHINE LEARNING DEPARTMENT

Fast, Cheap and Deep

Scaling machine learning

Alexander Smola
Machine Learning and Marianas Labs
github.com/dmlc

Marianas Labs
Carnegie Mellon University

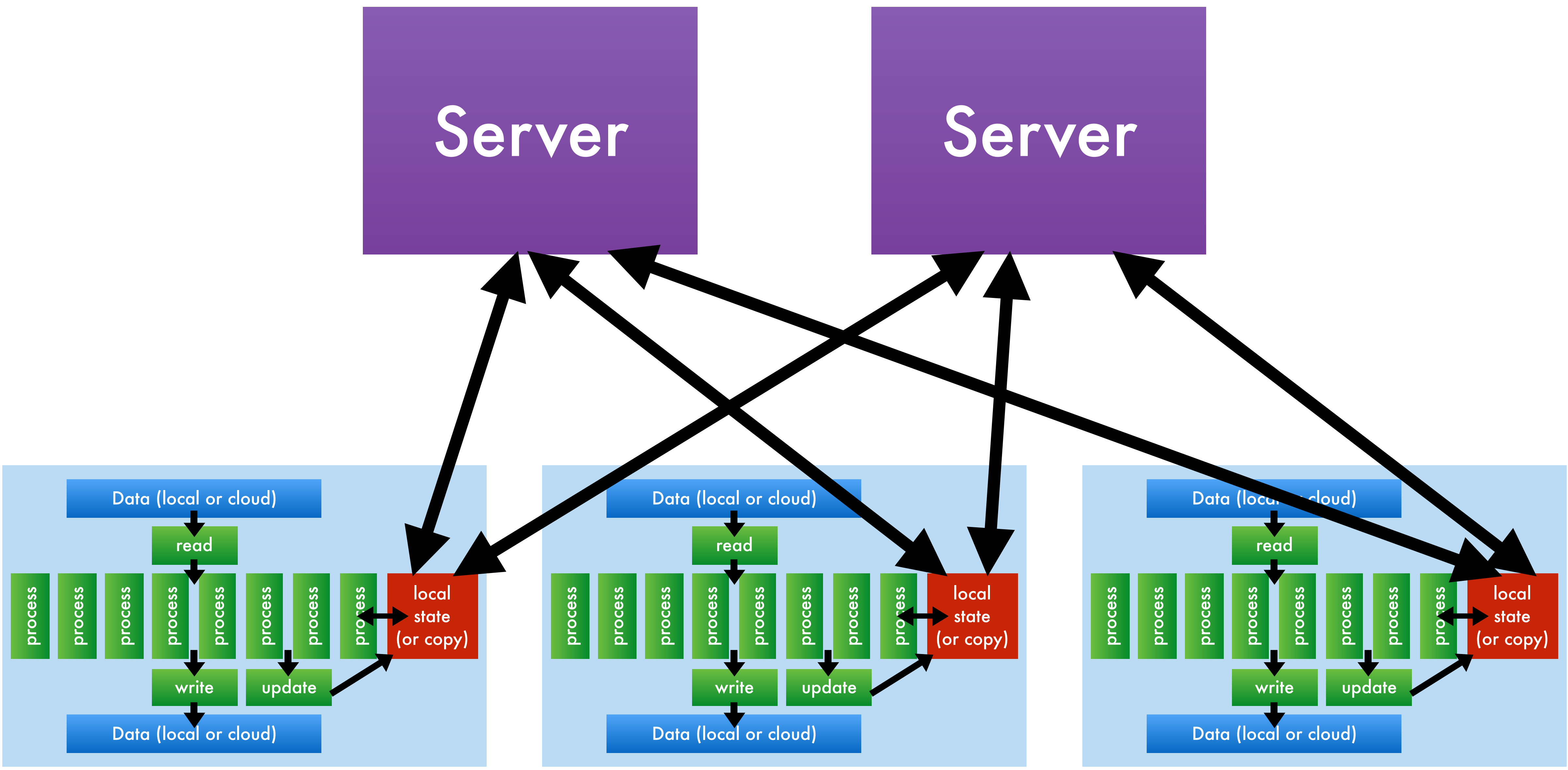
Many thanks to

- Mu Li
- Dave Andersen
- Chris Dyer
- Li Zhou
- Ziqi Liu
- Manzil Zaheer
- Qicong Chen
- Amr Ahmed (Google)
- Yu-Xiang Wang
- Jay Yoon Lee
- Ha Loc Do (SMU)

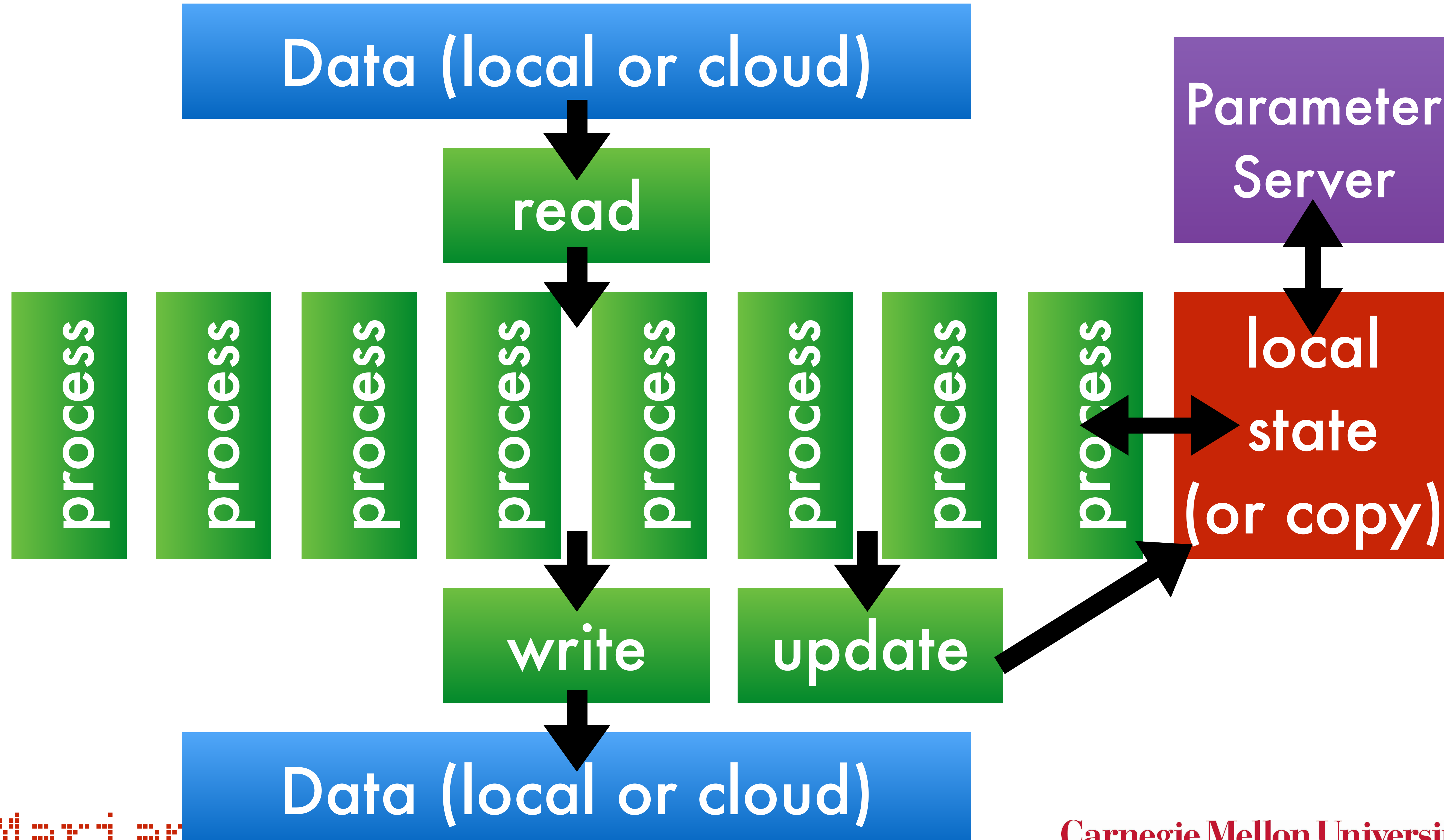
- **CXXNET Team**
  - Tianqi Chen (UW)
  - Bing Xu
  - Naiyang Wang
- **Minerva Team**
  - Minjie Wang
  - Tianjun Xiao
  - Jianpeng Li
  - Jiaxing Zhang

This talk in 3 slides

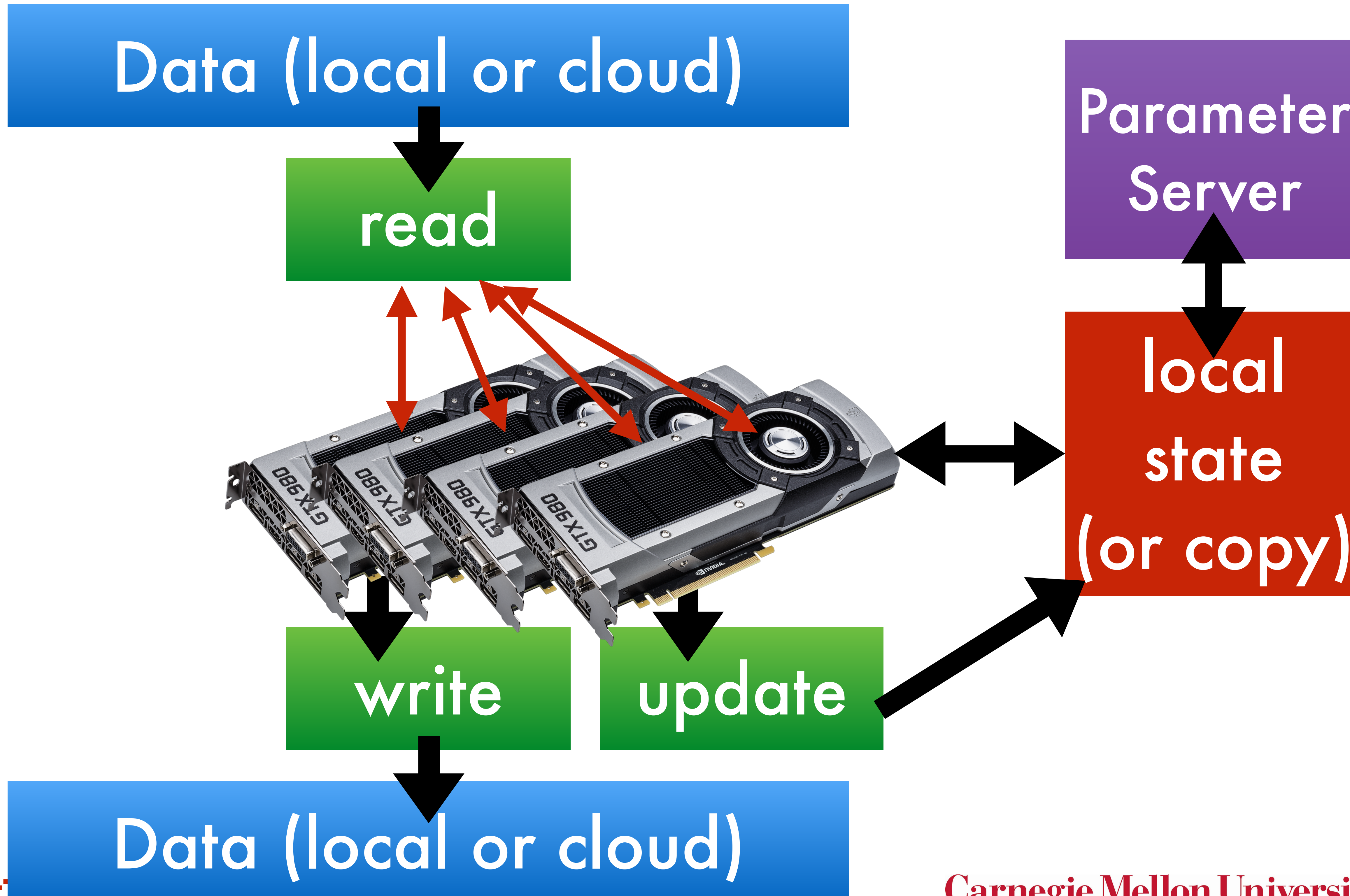
Parameter Server



Multicore


Data (local or cloud)
read
process
process
process
process
process
process
process
process
Parameter Server
local state (or copy)
write
update
Data (local or cloud)

GPUs (for Deep Learning)

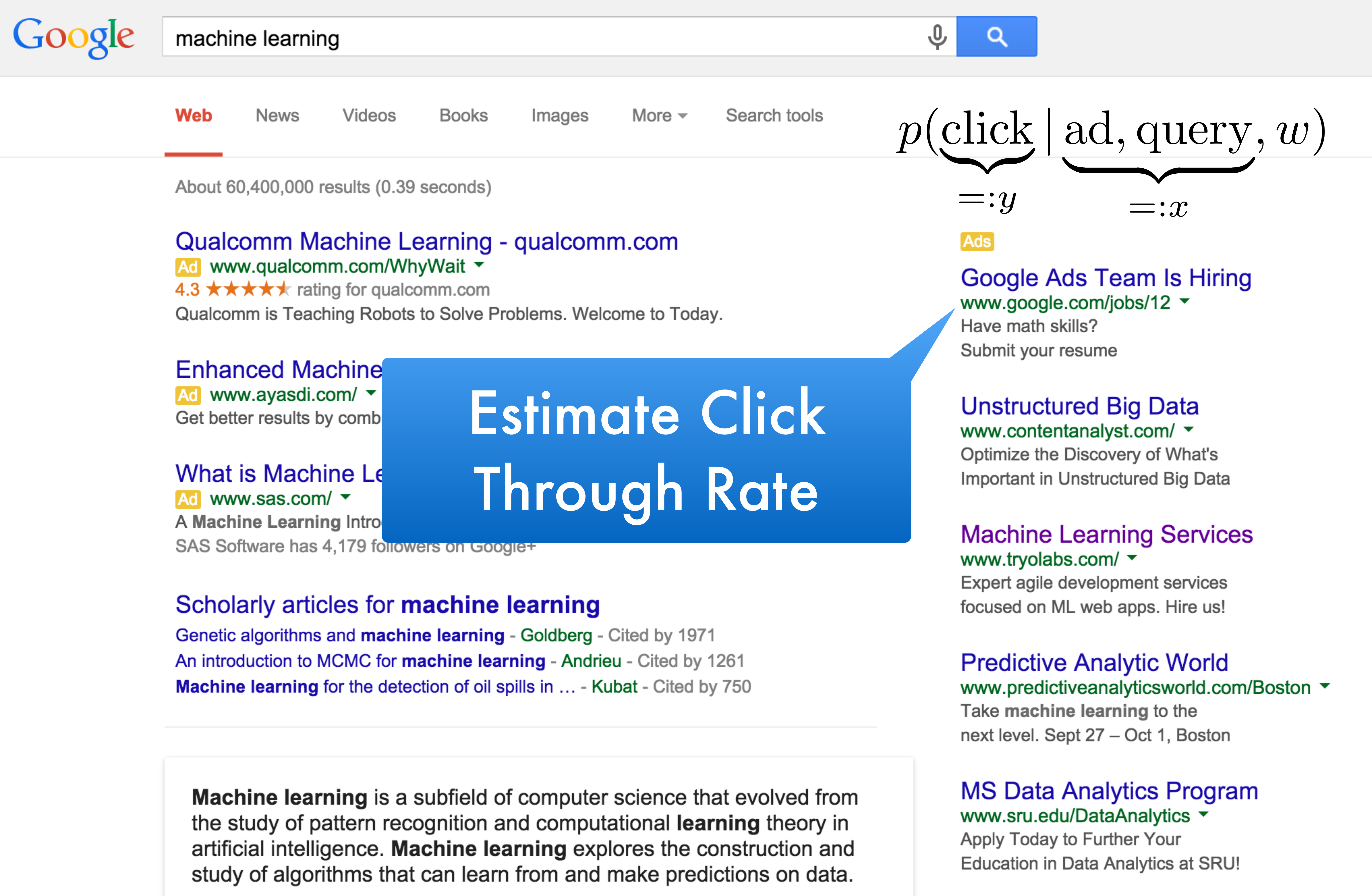


Details

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Large Distributed State**
  Factorization Machines (CTR)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **GPUs**
  Deep Learning (Images)

$$p(\underbrace{\text{click}}_{=:y} \mid \underbrace{\text{ad}, \text{query}}_{=:x}, w)$$

Estimate Click Through Rate



Click Through Rate (CTR)

- Linear function class

$$f(x) = \langle w, x \rangle$$

- Logistic regression

$$p(y|x, w) = \frac{1}{1 + \exp\left(-y \langle w, x \rangle\right)}$$

- Optimization Problem

sparse models for advertising

$$\underset{w}{\text{minimize}} \sum_{i=1}^{m} \log(1 + \exp(-y_i \langle w, x_i \rangle)) + \lambda \|w\|_1$$

- Solve distributed over many machines **(typically 1TB to 1PB of data)**

Optimization Algorithm

- **Compute gradient on data**
- l_1 norm is nonsmooth, hence proximal operator

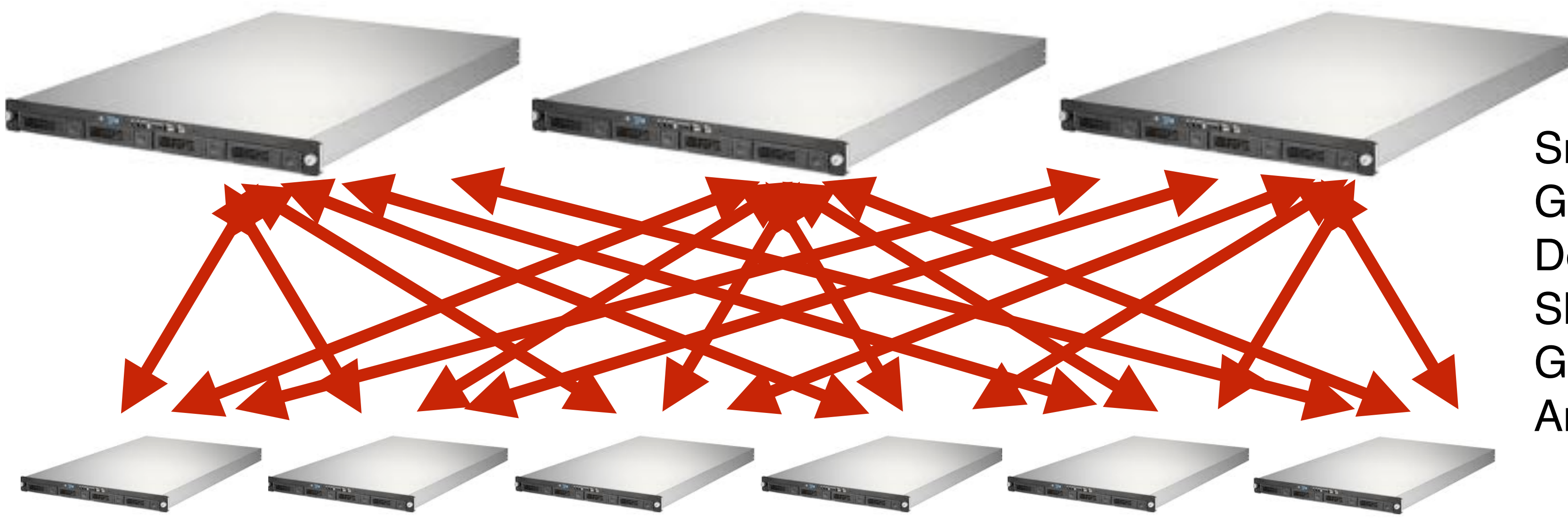
$$\underset{w}{\operatorname{argmin}} \|w\|_1 + \frac{\gamma}{2}\|w - (w_t - \eta g_t)\|^2$$

- Updates for l_1 are very simple

$$w_i \leftarrow \operatorname{sgn}(w_i) \max(0, |w_i| - \epsilon)$$

- **All steps decompose by coordinates**
- Solve in parallel (and asynchronously)

Parameter Server Template

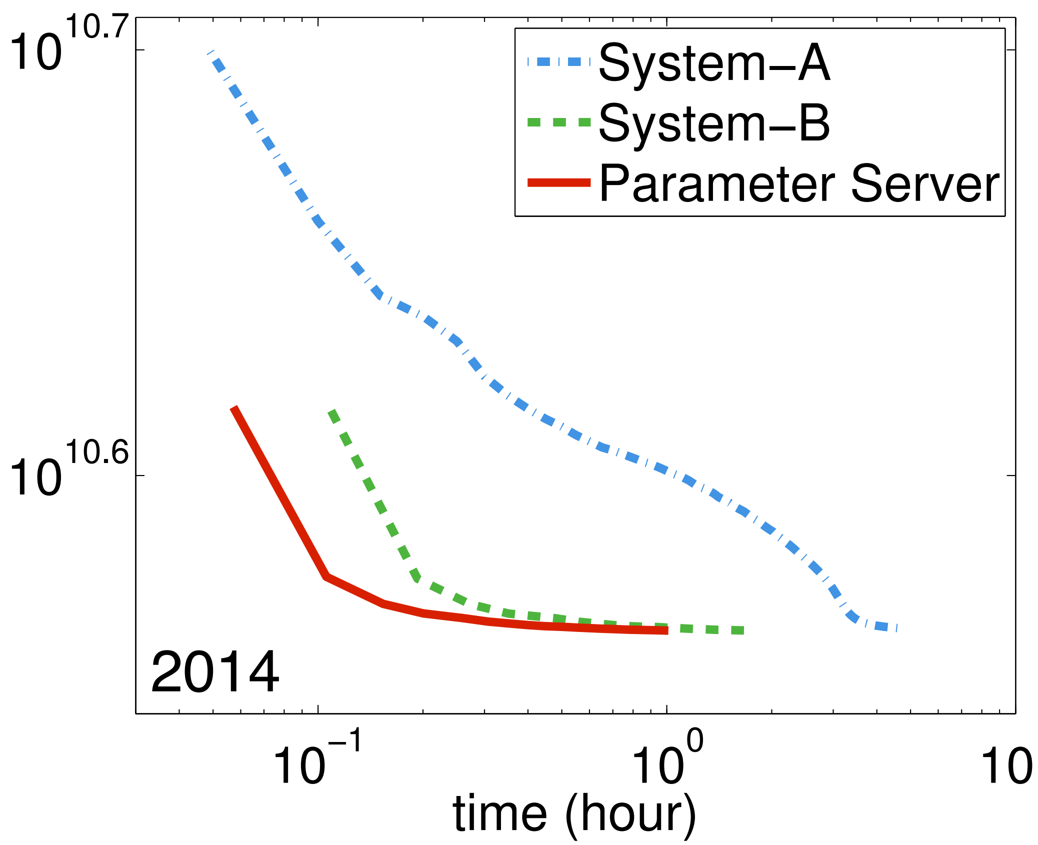


Smola & Narayanamurthy, 2010, VLDB
Gonzalez et al., 2012, WSDM
Dean et al, 2012, NIPS
Shervashidze et al., 2013, WWW
Google, Baidu, Facebook,
Amazon, Yahoo, Microsoft

- Compute gradient on (subset of data) on each client
- Send gradient from client to server asynchronously
  `push(key_list,value_list,timestamp)`
- Proximal gradient update on server per coordinate
- Server returns parameters
  `pull(key_list,value_list,timestamp)`

Solving it at scale

- **2014 - Li et al., OSDI'14**
  - 500 TB data, 10^{11} variables
  - Local file system stores files
  - 1000 servers (corp cloud),
  - 1h time, 140 MB/s learning
- **2015 - Online solver**
  - 1.1 TB (Criteo), $8 \cdot 10^8$ variables, $4 \cdot 10^9$ samples
  - S3 stores files (no preprocessing) - **better IO library**
  - 5 machines (c4.8xlarge),
  - 1000s time, 220 MB/s learning



Details

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Large Distributed State**
  Factorization Machines (CTR)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **GPUs**
  Deep Learning (Images)

Google

machine learning

Web News Videos Books Images More Search tools

$$p(y|x, w)$$

About 60,400,000 results (0.39 seconds)

Qualcomm Machine Learning - qualcomm.com
Ad www.qualcomm.com/WhyWait
4.3 ★★★★★ rating for qualcomm.com
Qualcomm is Teaching Robots to Solve Problems. Welcome to Today.

Enhanced Machine
Ad www.ayasdi.com/
Get better results by comb

What is Machine Le
Ad www.sas.com/
A **Machine Learning** Intro
SAS Software has 4,179 followers on Google+

A Linear Model is not enough

Scholarly articles for **machine learning**
Genetic algorithms and **machine learning** - Goldberg - Cited by 1971
An introduction to MCMC for **machine learning** - Andrieu - Cited by 1261
Machine learning for the detection of oil spills in … - Kubat - Cited by 750

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational **learning** theory in artificial intelligence. **Machine learning** explores the construction and study of algorithms that can learn from and make predictions on data.

Ads

Google Ads Team Is Hiring
www.google.com/jobs/12
Have math skills?
Submit your resume

Unstructured Big Data
www.contentanalyst.com/
Optimize the Discovery of What's
Important in Unstructured Big Data

Machine Learning Services
www.tryolabs.com/
Expert agile development services
focused on ML web apps. Hire us!

Predictive Analytic World
www.predictiveanalyticsworld.com/Boston
Take **machine learning** to the
next level. Sept 27 – Oct 1, Boston

MS Data Analytics Program
www.sru.edu/DataAnalytics
Apply Today to Further Your
Education in Data Analytics at SRU!

Factorization Machines

- Linear Model

$$f(x) = \langle w, x \rangle$$

- Polynomial Expansion (Rendle, 2012)

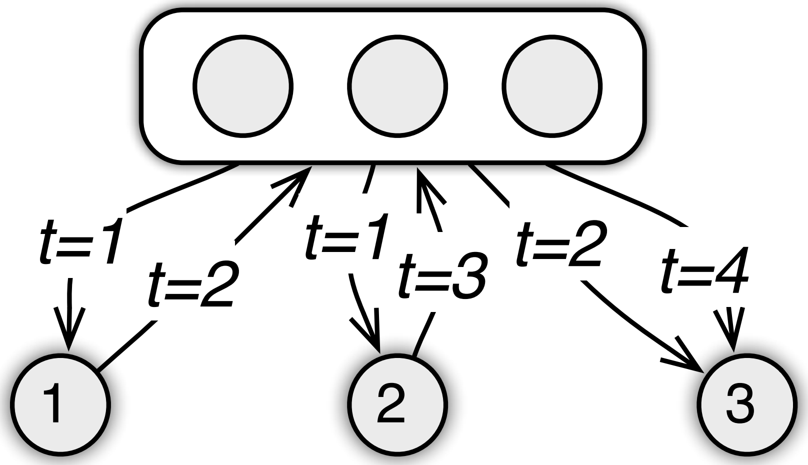
$$f(x) = \langle w, x \rangle + \sum_{i<j} x_i x_j \operatorname{tr}\left(V_i^{(2)} \otimes V_j^{(2)}\right) + \sum_{i<j<k} x_i x_j x_k \operatorname{tr}\left(V_i^{(3)} \otimes V_j^{(3)} \otimes V_k^{(3)}\right) + \ldots$$

memory hog

too large for individual machine

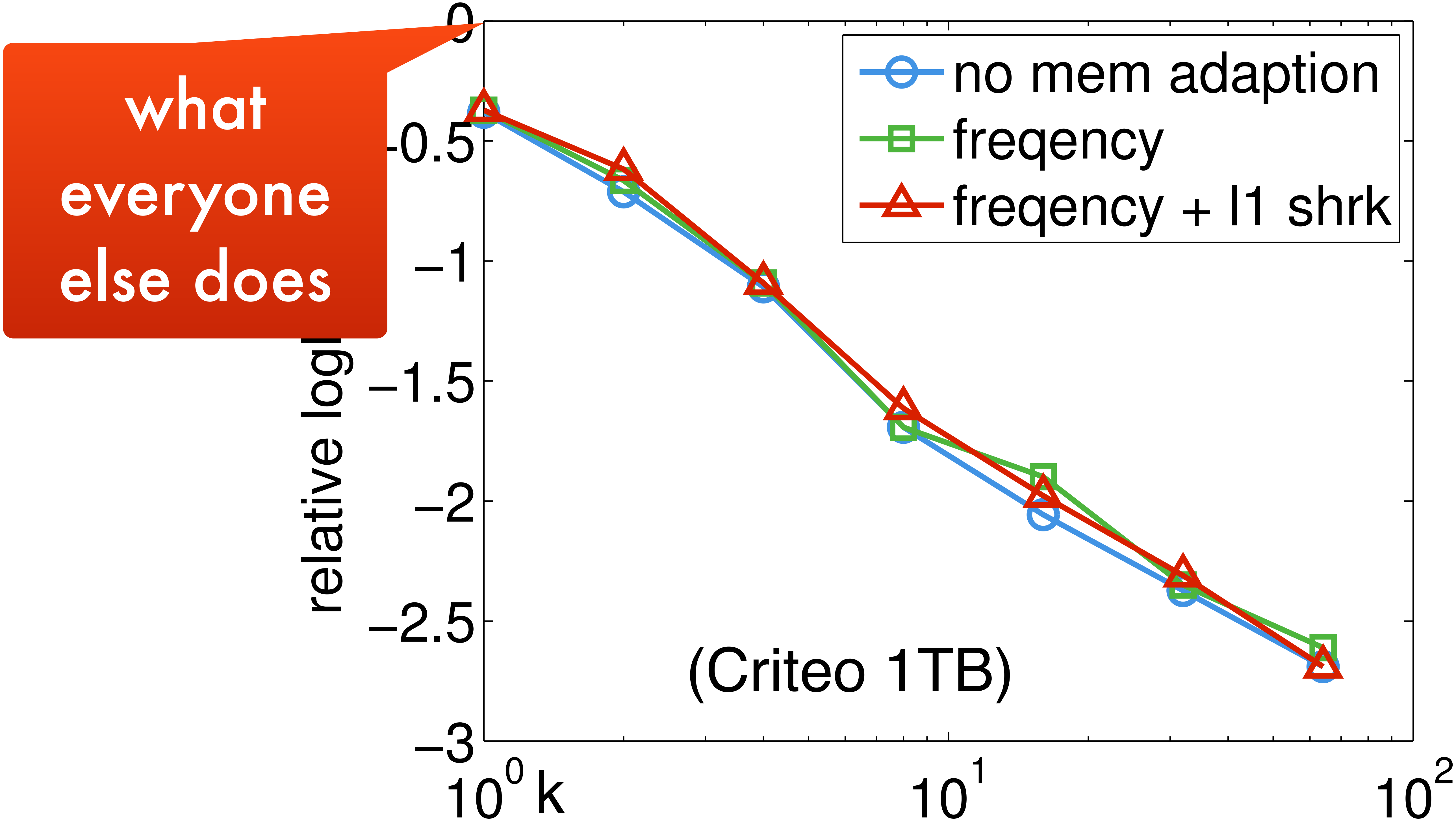
Prefetching to the rescue

- **Most keys are infrequent** (power law distribution)
- **Prefetch** the embedding **vectors** for a **minibatch** from parameter server

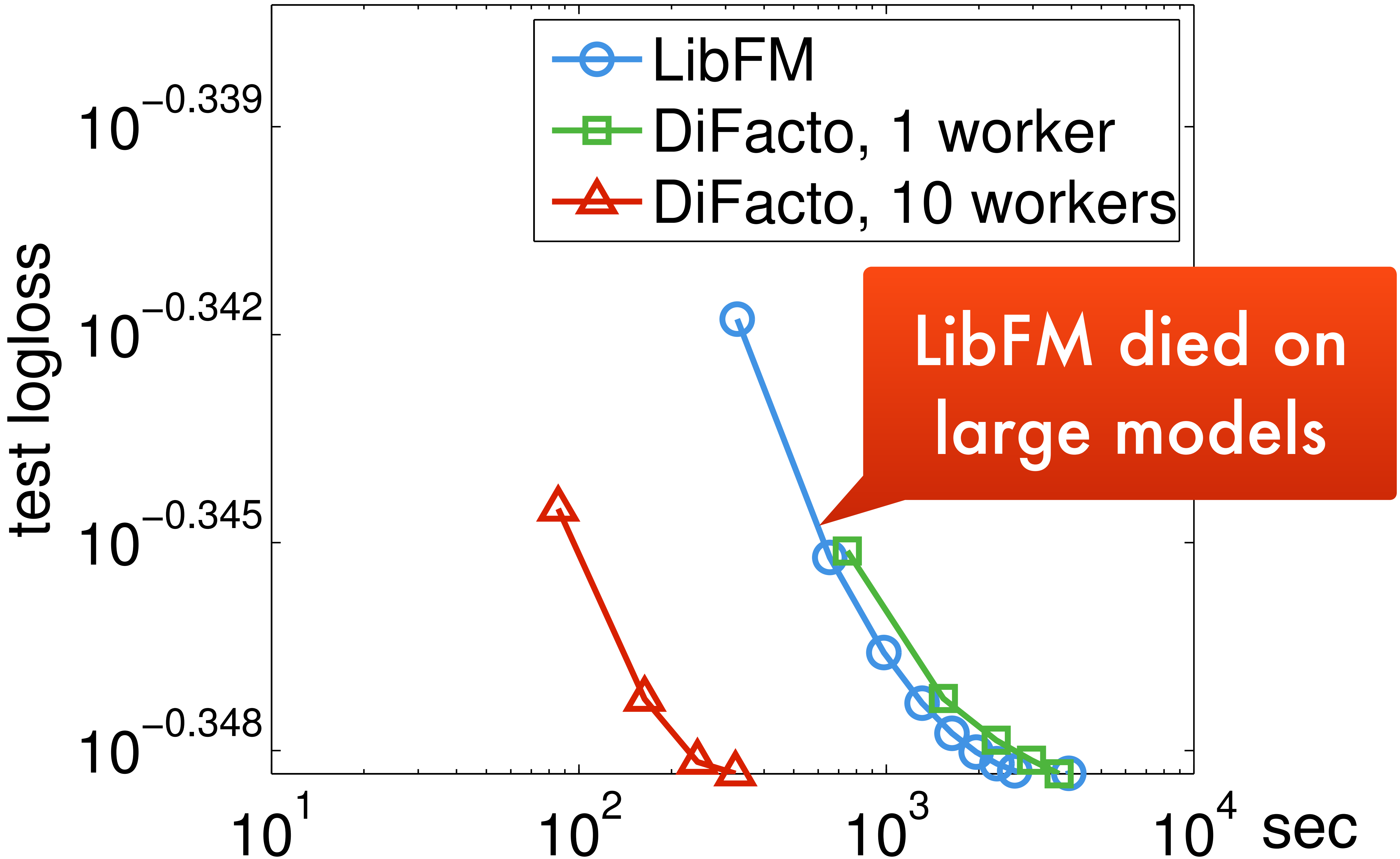


- **Compute gradients** and **push to server**
    - Variable dimensionality embedding
    - Enforcing sparsity (ANOVA style)
    - Adaptive gradient normalization
    - Frequency adaptive regularization (CF style)

Better Models

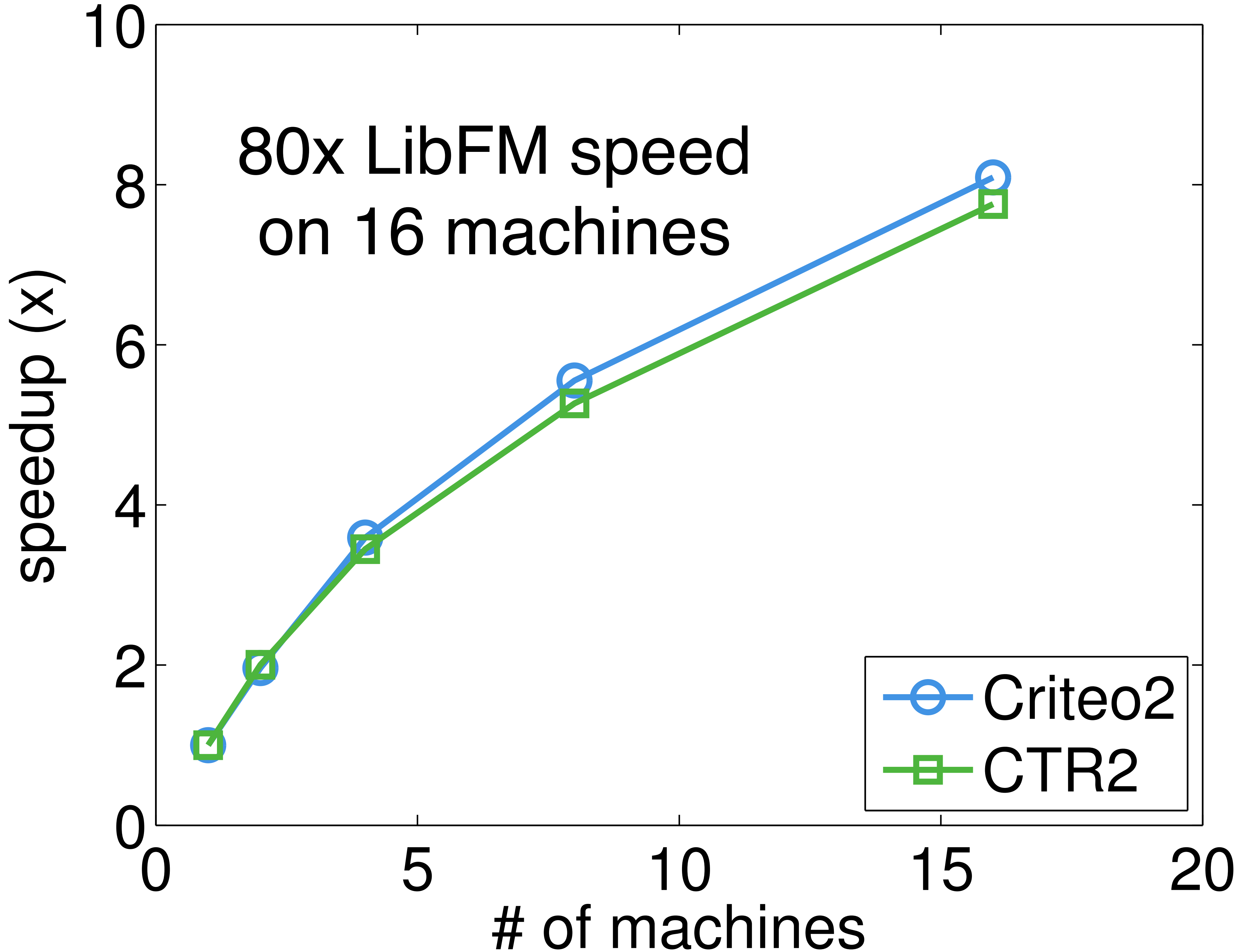


Faster Solver (small Criteo)



Multiple Machines

Li, Wang, Liu, Smola, WSDM'16, submitted



Details

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Large Distributed State**
  Factorization Machines (CTR)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **GPUs**
  Deep Learning (Images)

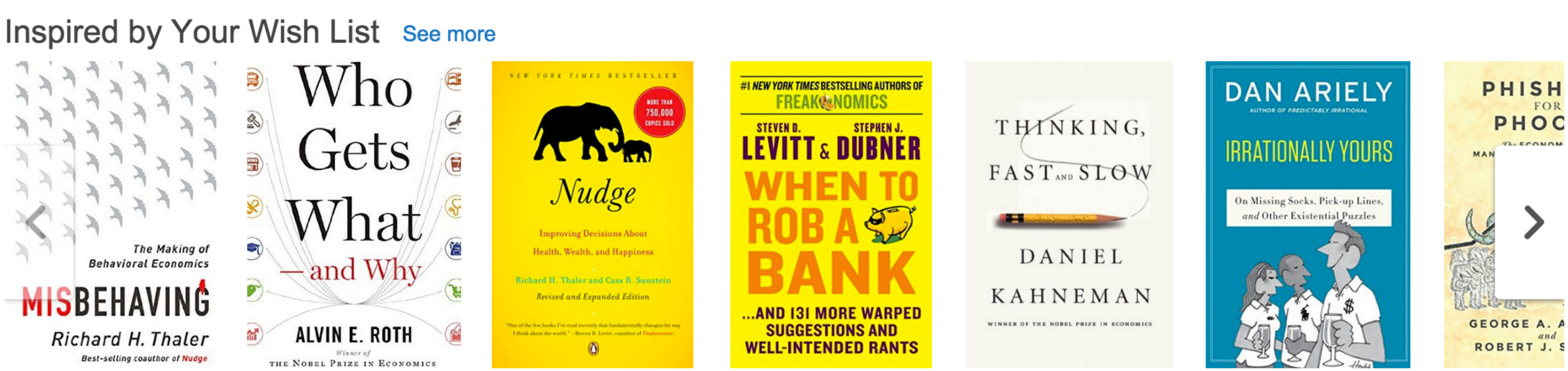
Recommender Systems

- Users u, movies m (or projects)
- Function class

$$r_{um} = \langle v_u, w_m \rangle + b_u + b_m$$

- Loss function for recommendation (Yelp, Netflix)

$$\sum_{u \sim m} \left(\langle v_u, w_m \rangle + b_u + b_m - y_{um} \right)^2$$



Recommender Systems

- Regularized Objective

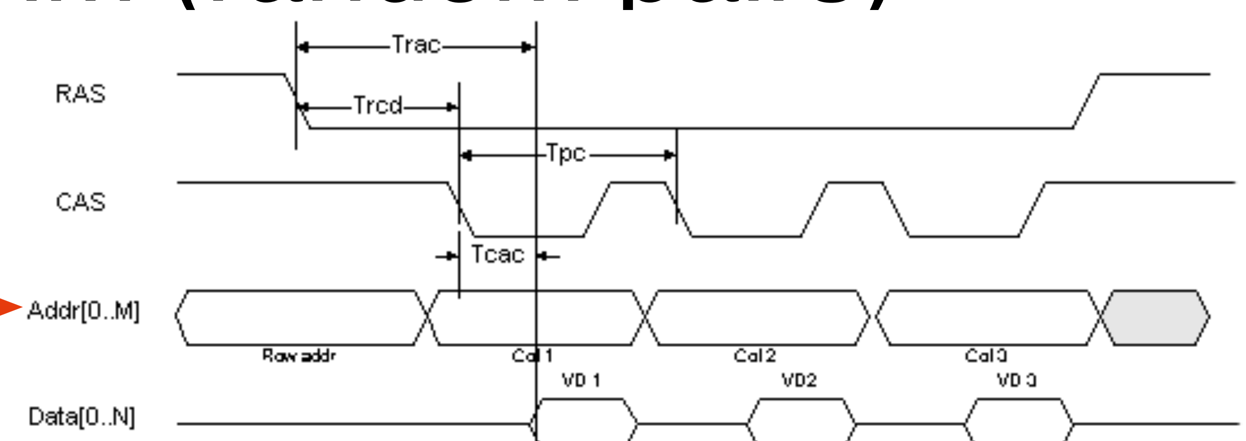
$$\sum_{u \sim m} \left(\langle v_u, w_m \rangle + b_u + b_m + b_0 - r_{um}\right)^2 + \frac{\lambda}{2}\left[\|U\|^2_{\mathrm{Frob}} + \|V\|^2_{\mathrm{Frob}}\right]$$

- Update operations

$$v_u \leftarrow (1 - \eta_t \lambda) v_u - \eta_t w_m \left(\langle v_u, w_m \rangle + b_u + b_m + b_0 - r_{um}\right)$$
$$w_m \leftarrow (1 - \eta_t \lambda) w_m - \eta_t v_u \left(\langle v_u, w_m \rangle + b_u + b_m + b_0 - r_{um}\right)$$

- Very simple SGD algorithm (random pairs)
- This should be cheap …

memory subsystem



FPM/EDO read timing

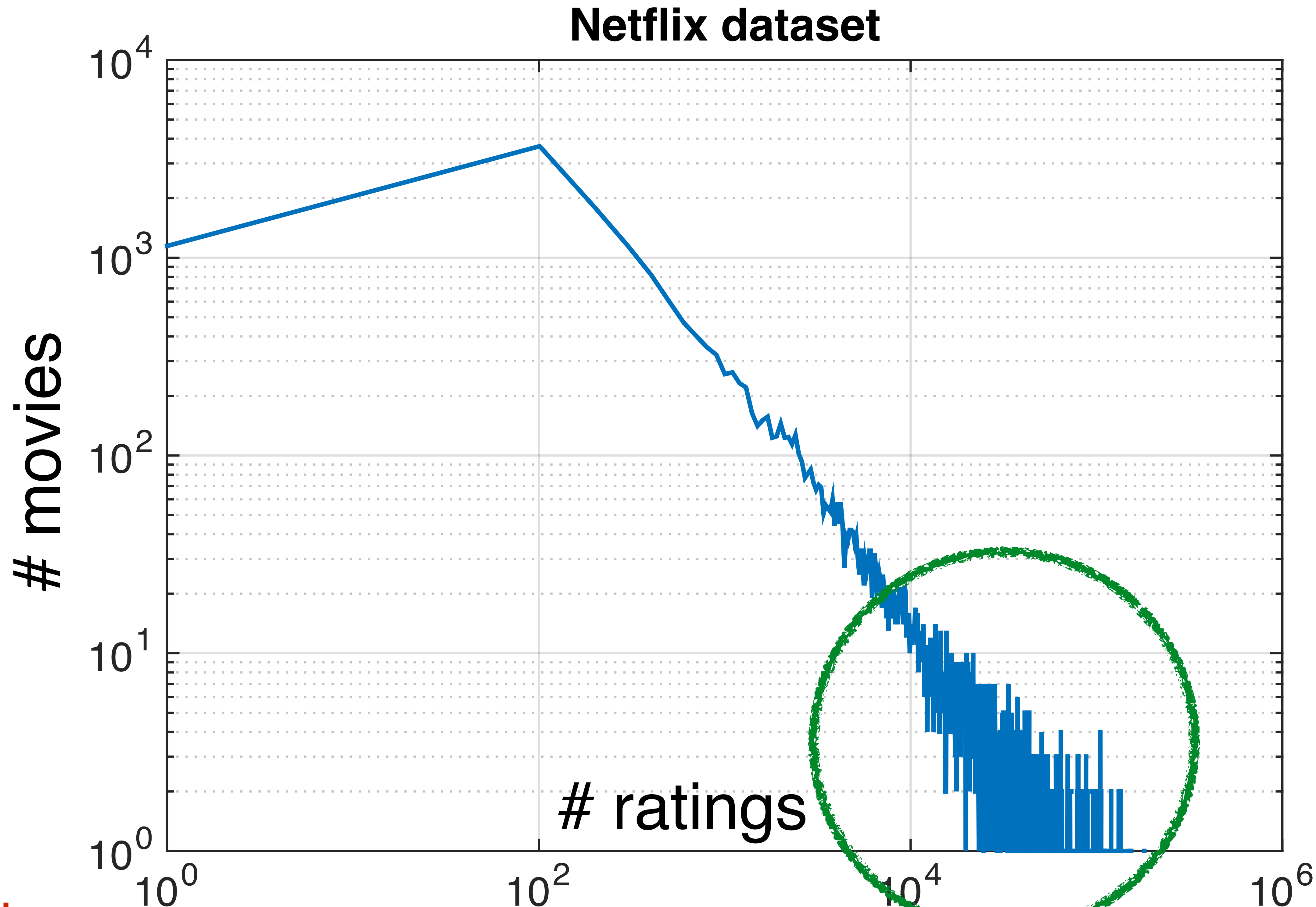
This should be cheap …

- O(md) burst reads and O(m) random reads
- Netflix dataset
  m = 100 million, d = 2048 dimensions, 30 steps
- Runtime should be > 4500s
  - 60 GB/s memory bandwidth = 3300s
  - 100 ns random reads = 1200s

We get 560s. Why?

Liu, Wang, Smola, RecSys 2015

Power law in Collaborative Filtering

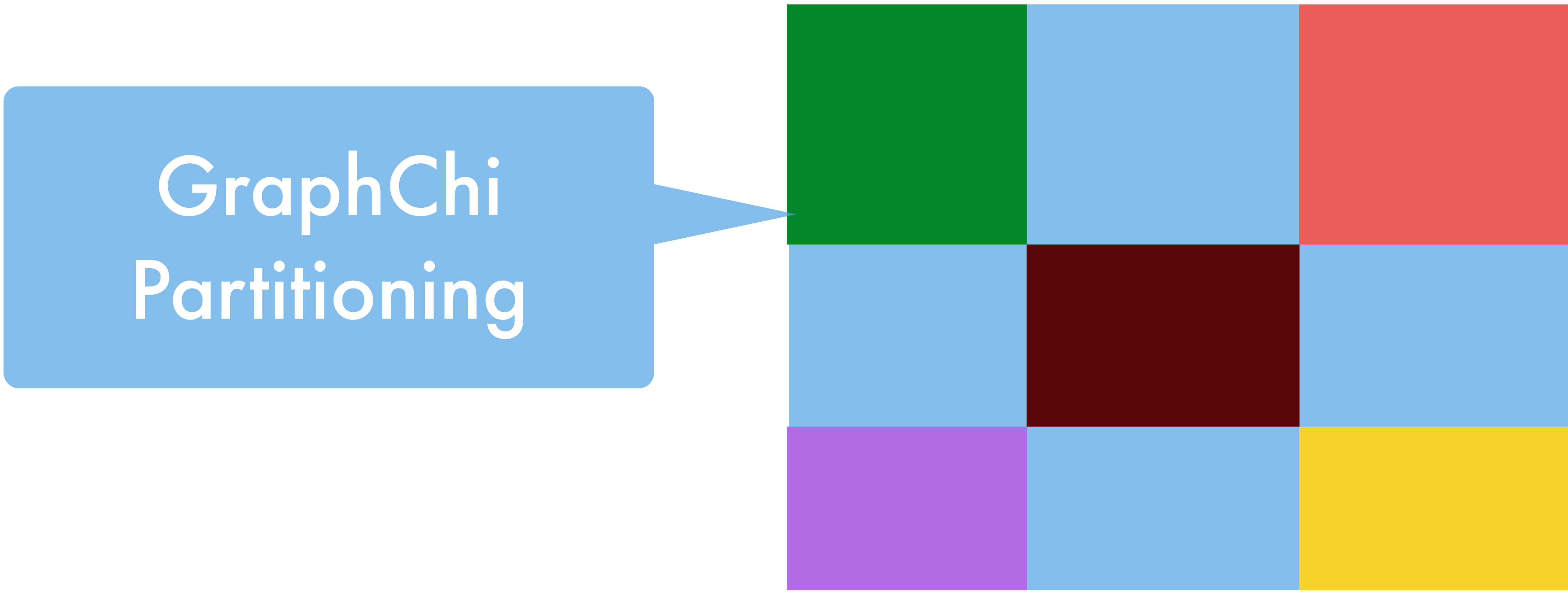


Key Ideas

- **Stratify ratings by users**
  (only 1 cache miss / read per user / out of core)
- **Keep frequent movies in cache**
  (stratify by blocks of movie popularity)
- **Avoid false sharing between sockets**
  (key cached in the wrong CPU causes miss)

K	SC-SGD		GraphChi	
	L1 Cache	L3 Cache	L1 Cache	L3 Cache
16	2.84%	0.43%	12.77%	2.21%
256	2.85%	0.50%	12.89%	2.34%
2048	3.3%	1.7%	15%	9.8%

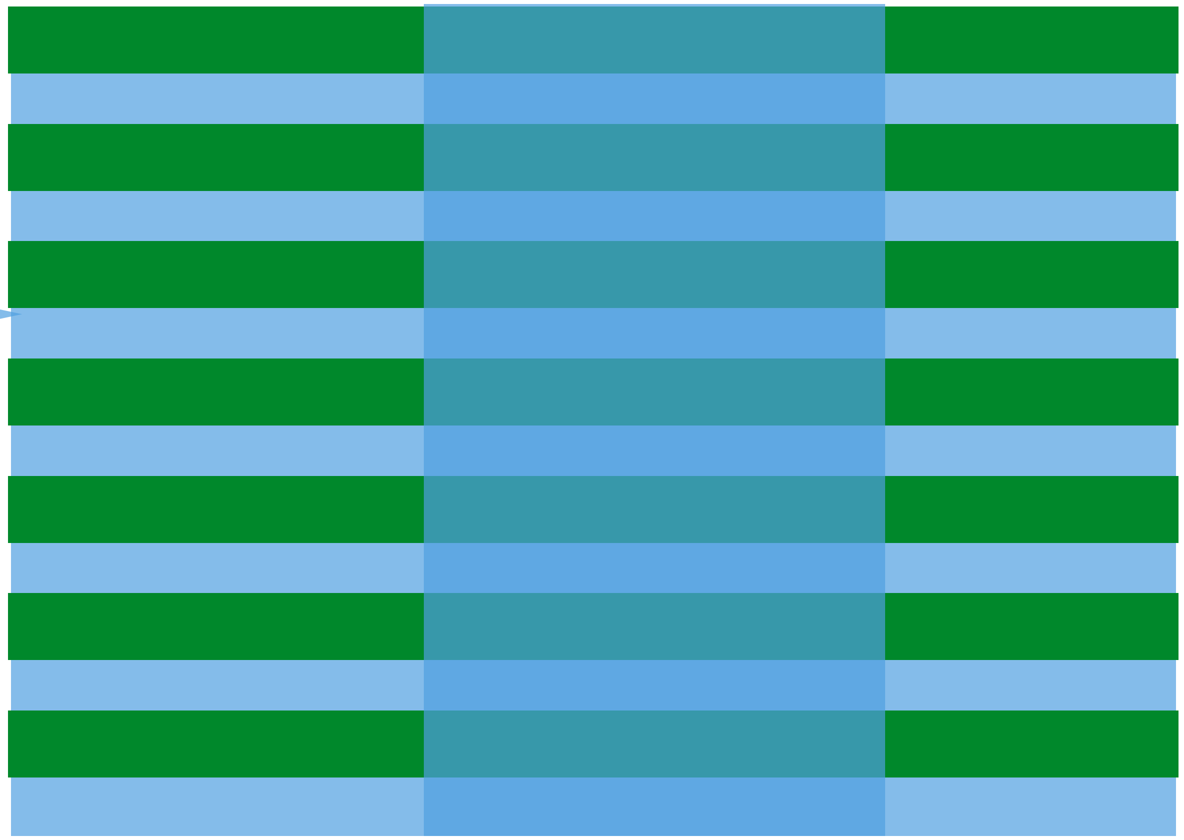
Key Ideas



K	SC-SGD		GraphChi	
	L1 Cache	L3 Cache	L1 Cache	L3 Cache
16	2.84%	0.43%	12.77%	2.21%
256	2.85%	0.50%	12.89%	2.34%
2048	3.3%	1.7%	15%	9.8%

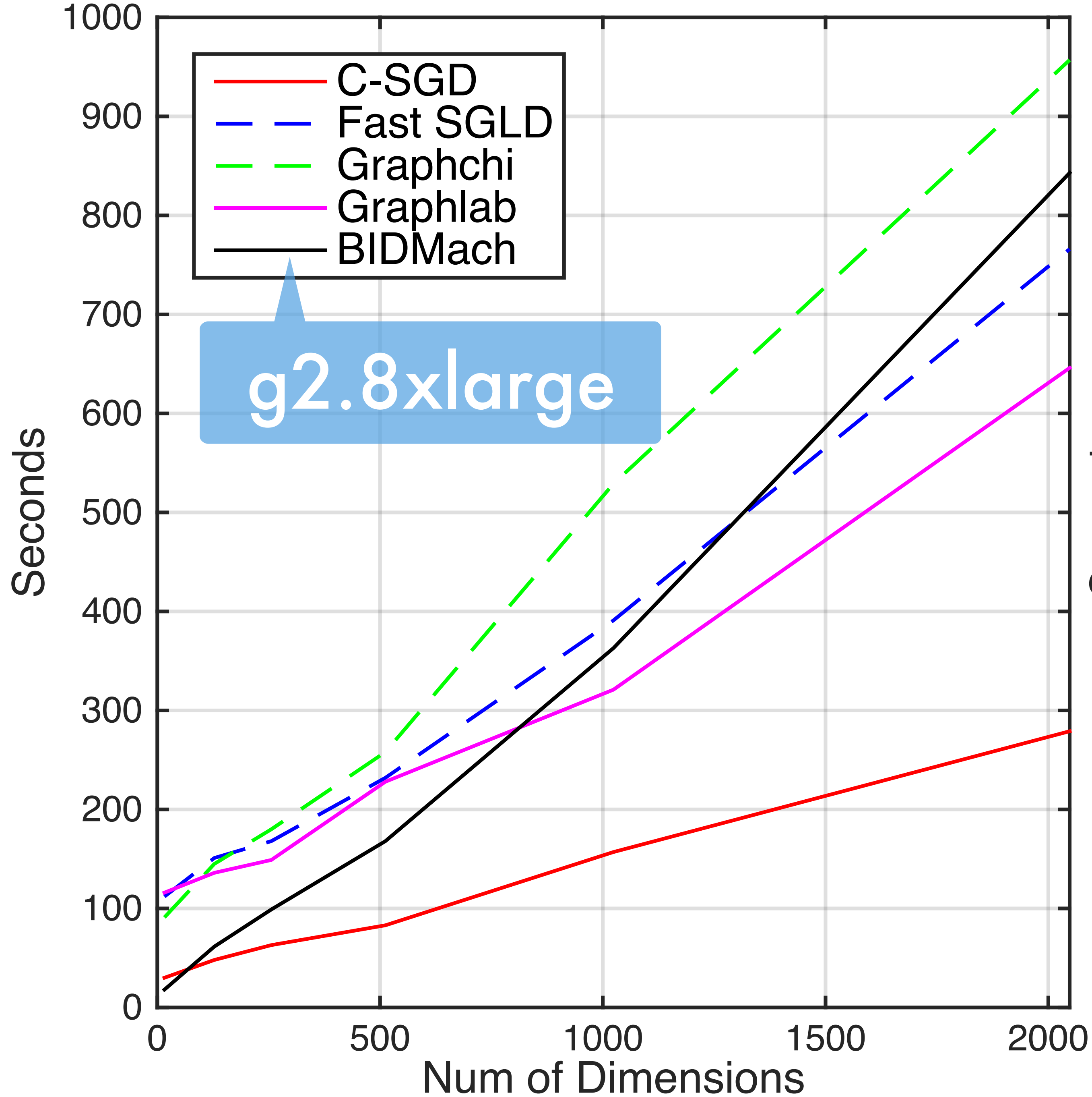
Key Ideas

SC-SGD partitioning

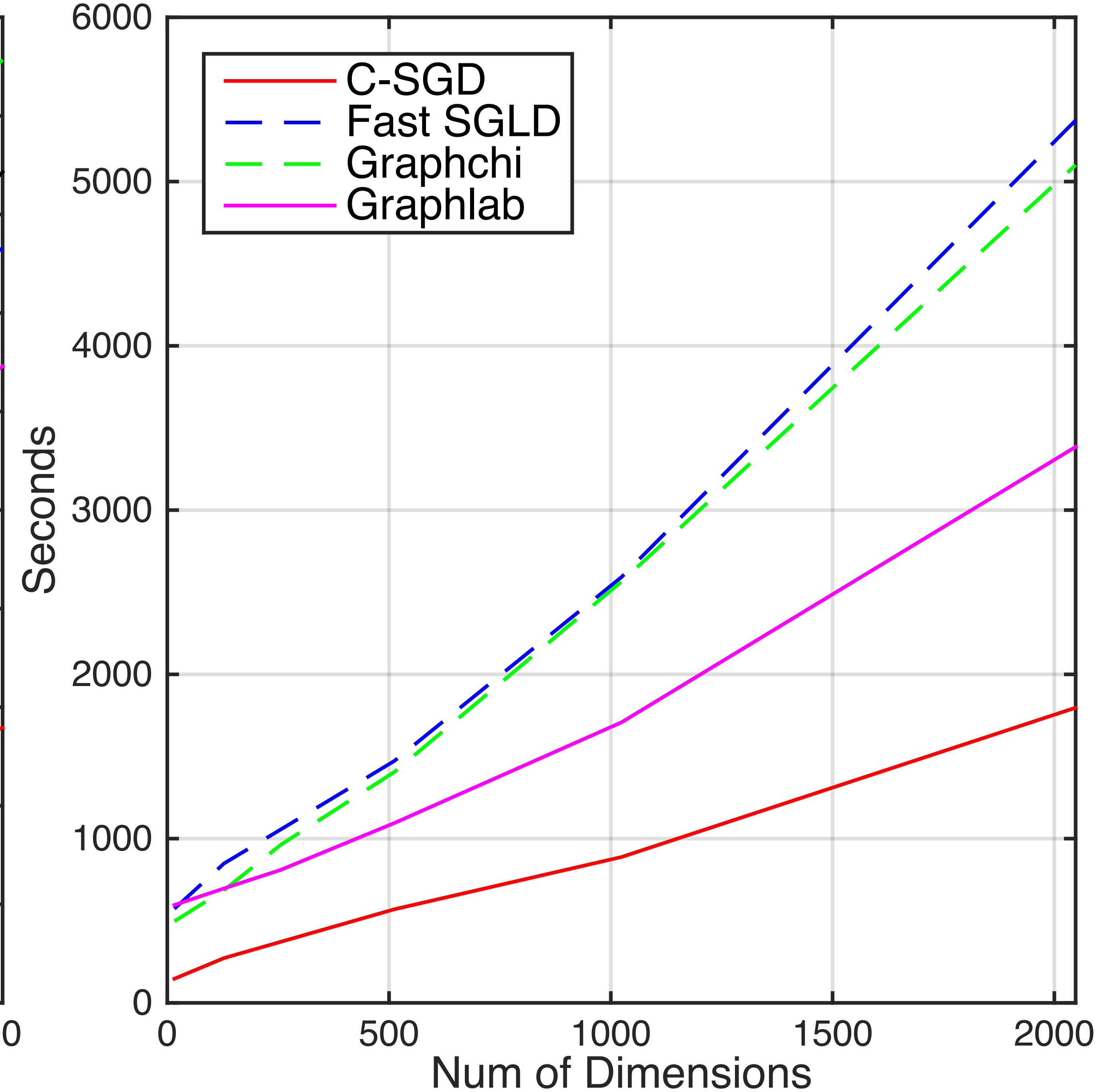


K	SC-SGD		GraphChi	
	L1 Cache	L3 Cache	L1 Cache	L3 Cache
16	2.84%	0.43%	12.77%	2.21%
256	2.85%	0.50%	12.89%	2.34%
2048	3.3%	1.7%	15%	9.8%

Speed (c4.8xlarge)



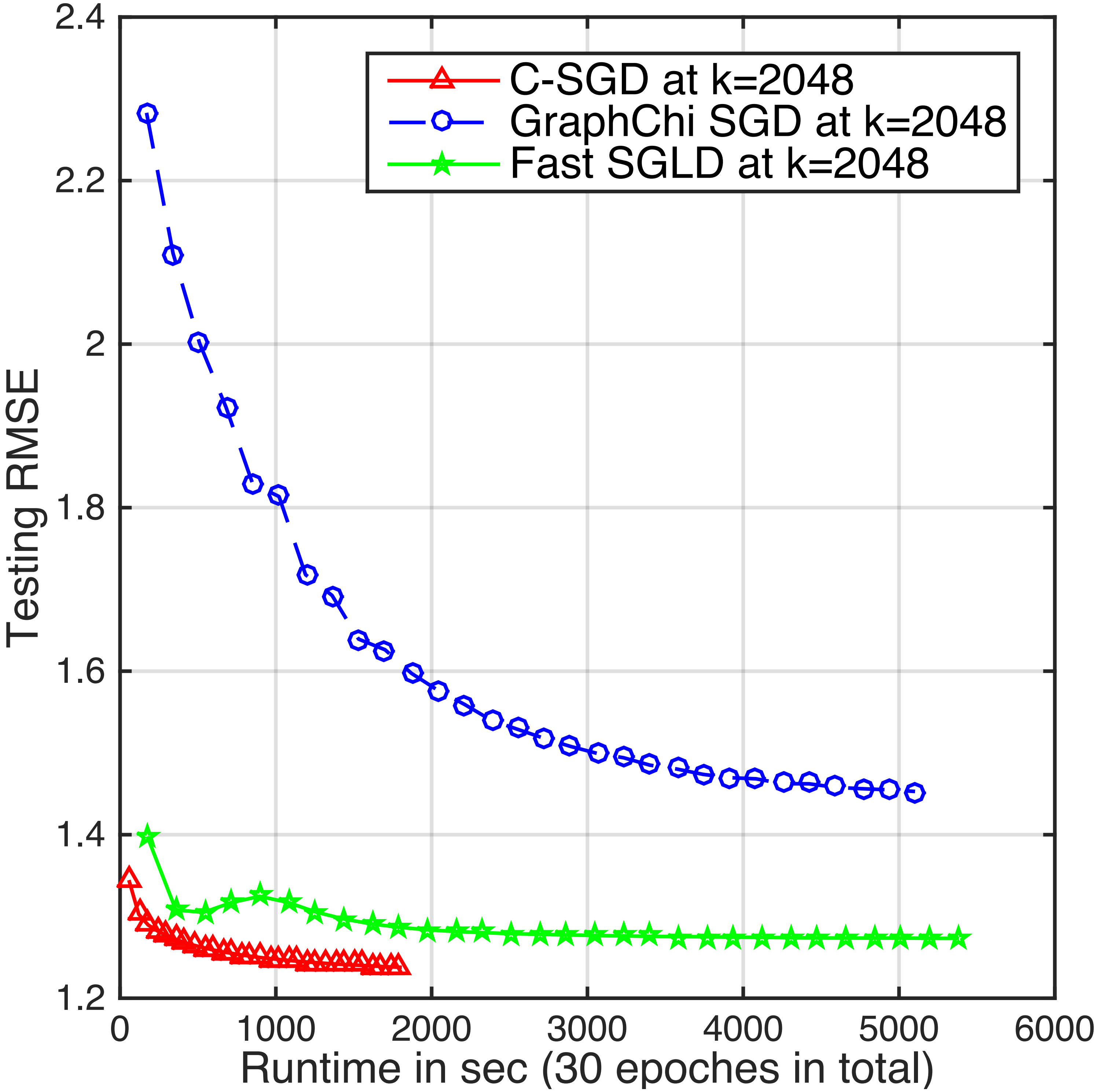
Netflix - 100M, 15 iterations



Yahoo - 250M, 30 iterations

Marianas Labs

Convergence



- GraphChi blocks (users, movies) into random groups
- Poor mixing
- Slow convergence

Details

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Large Distributed State**
  Factorization Machines (CTR)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **GPUs**
  Deep Learning (Images)

github.com/dmlc

The Challenge

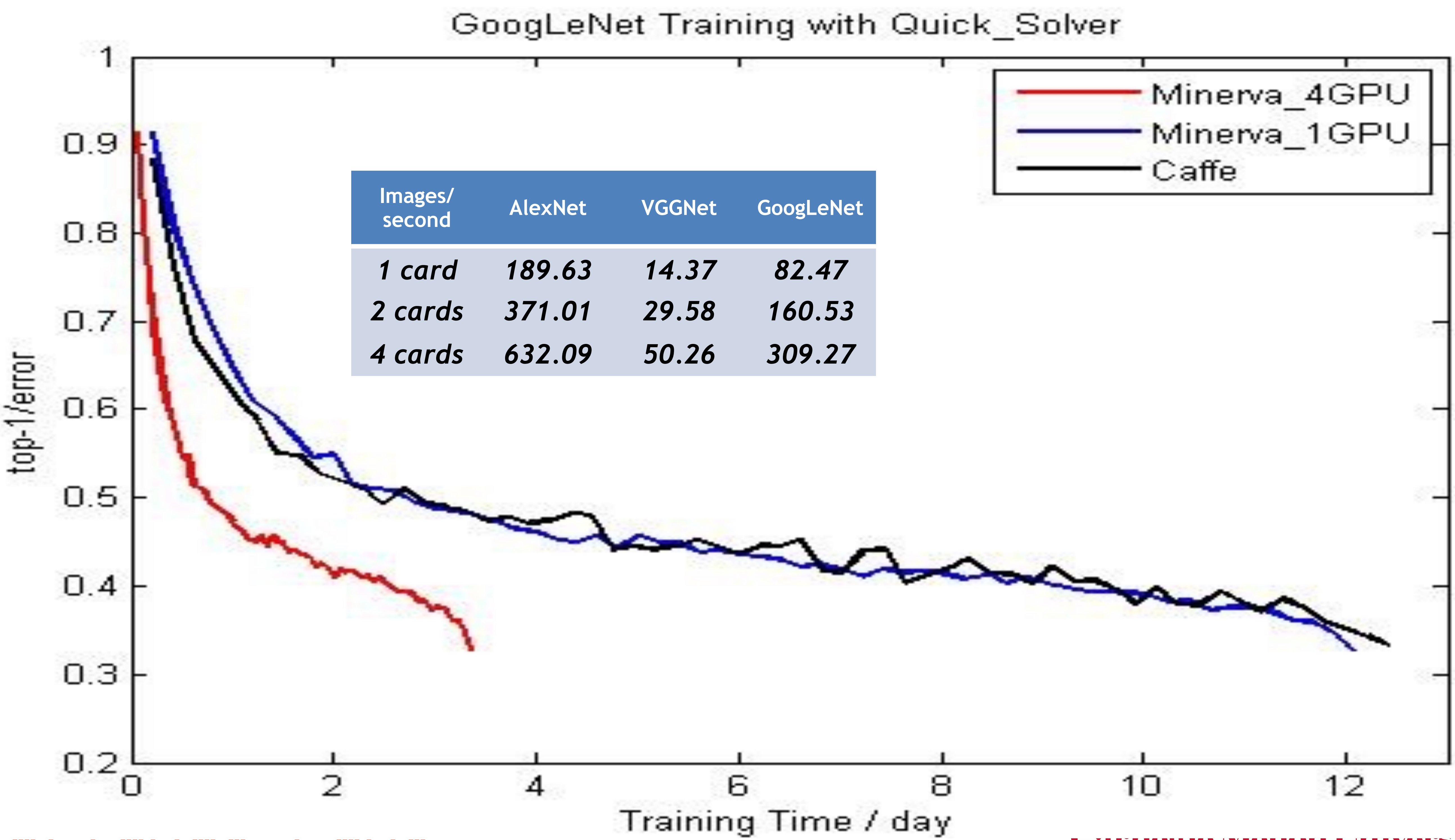
- Multiple good single-machine toolkits
  - **Caffe** - convolution optimized (images)
  - **CXXNET** - good tensor library
  - **Minerva** - Scheduler & Layout on CPU/GPU
  - Torch - Lua + interesting C preprocessor (very very popular, though)
  - Theano - Deep network compiler built by ML
- **Don’t reinvent the wheel for deep learning**
- Integrate with parameter server

Minerva (dmlc/minerva)

- Tensor interface in python (similar to numpy)
- Dataflow engine
- Auto parallel execution
  - On multi-core CPU
  - On multi-GPU
- Optimizes layout automatically

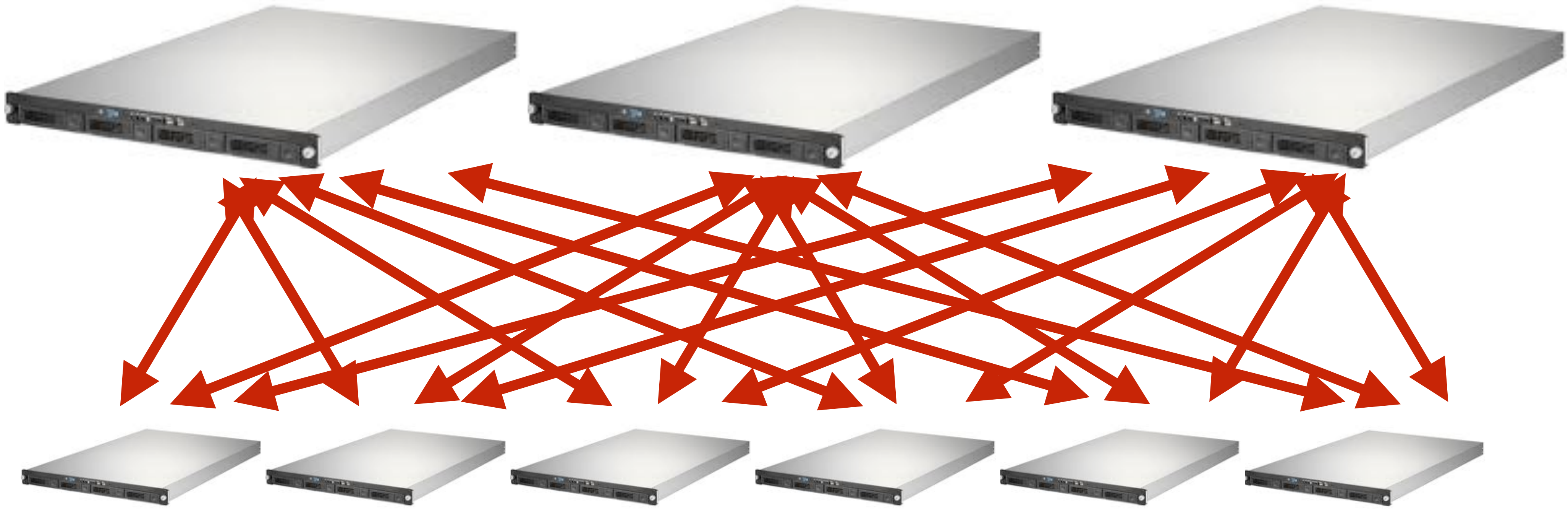
Zhang et al, '14 (NIPS workshop)

Minerva Scaling

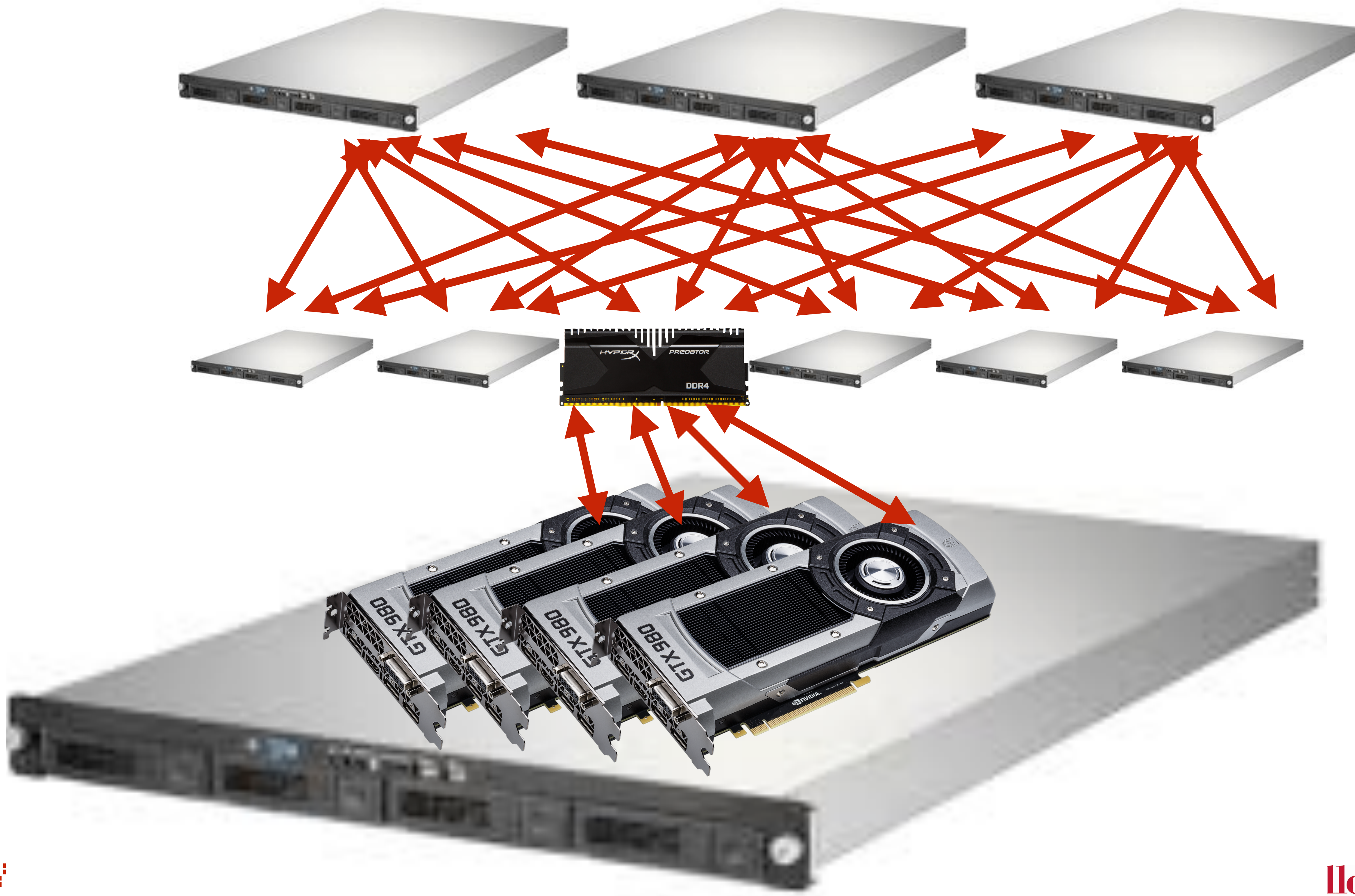


Images/ second	AlexNet	VGGNet	GoogLeNet
1 card	*189.63*	*14.37*	*82.47*
2 cards	*371.01*	*29.58*	*160.53*
4 cards	*632.09*	*50.26*	*309.27*

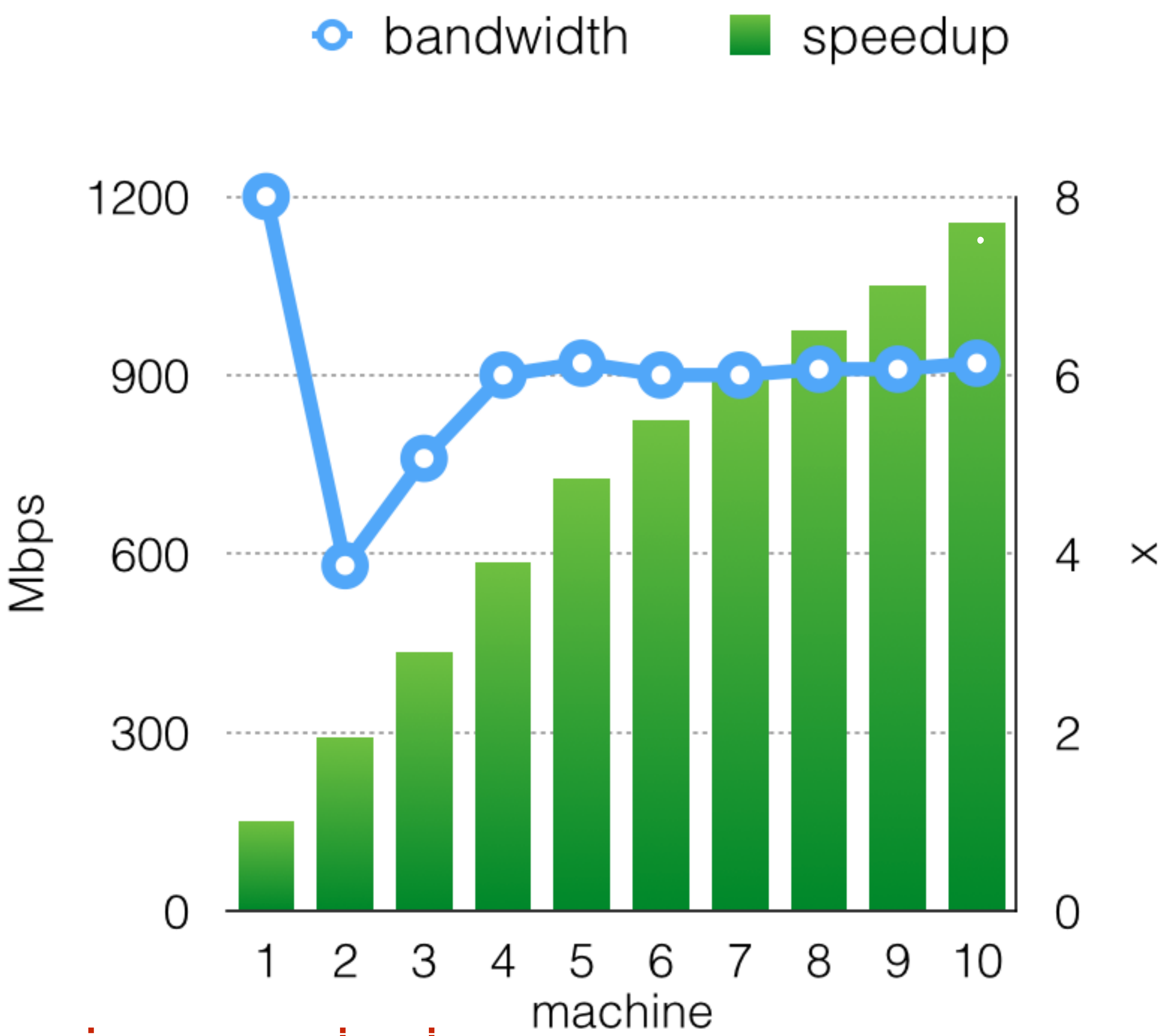
Distributed Deep Learning



Distributed Deep Learning



Scaling on AWS g2.2xlarge



1Gbit network limit
(alexnet scaling)

Amazon just released g2.8xlarge …

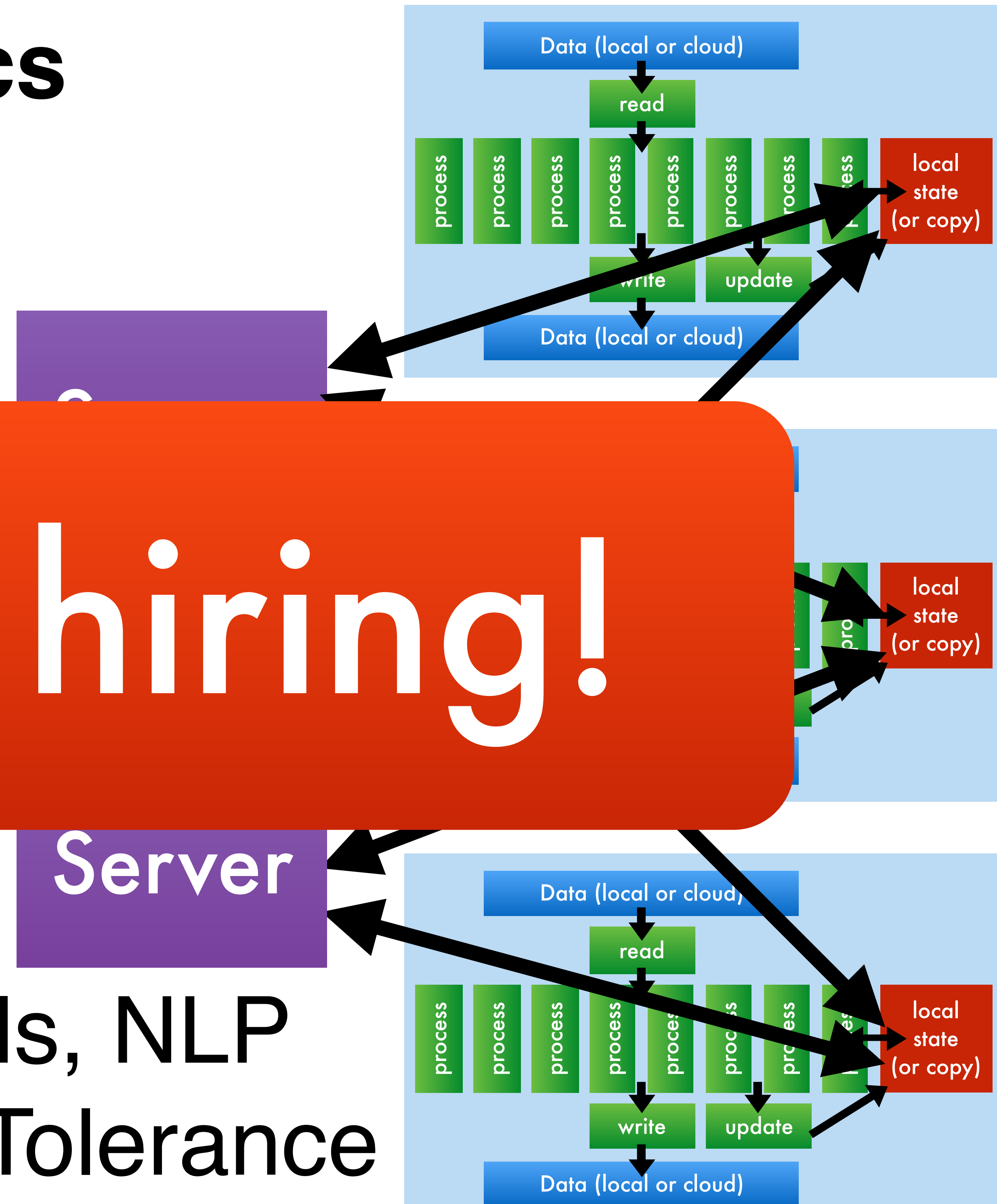
- 12 instances (48 GPUs) @ \$0.50/h spot
- Minibatch size 512
- BSP with 1 delay between machines
- 2 GB/s bandwidth between machines (awful)

10.113.170.187,10.157.109.227, 10.169.170.55, 10.136.52.151, 10.45.64.250, 10.166.137.100, 10.97.167.10, 10.97.187.157, 10.61.128.107, 10.171.105.160, 10.203.143.220, 10.45.71.20 (all over the place in availability zone)

- Compressing to 1 byte per coordinate helps a bit but adds latency due to extra pass (need to fix)
- **37x speedup on 48 GPUS**
- Imagenet'12 dataset in trained in 4h, i.e. \$24 (with alexnet; googlenet even better for network)

Summary

- **Parameter Server Basics**
  Logistic Regression
- **Large Distributed State**
  Fac
- **Me**
  Ma
- **GP**
  Deep Learning
- Much more - Topic Models, NLP
  Docker, Sketches, Fault Tolerance



We are hiring!