MACHINE LEARNING
DEPARTMENT

Scaling Machine Learning

Three easy pieces

Alex Smola
CMU & Marianas Labs
github.com/dmlc

Marianas Labs

Carnegie Mellon University

Outline

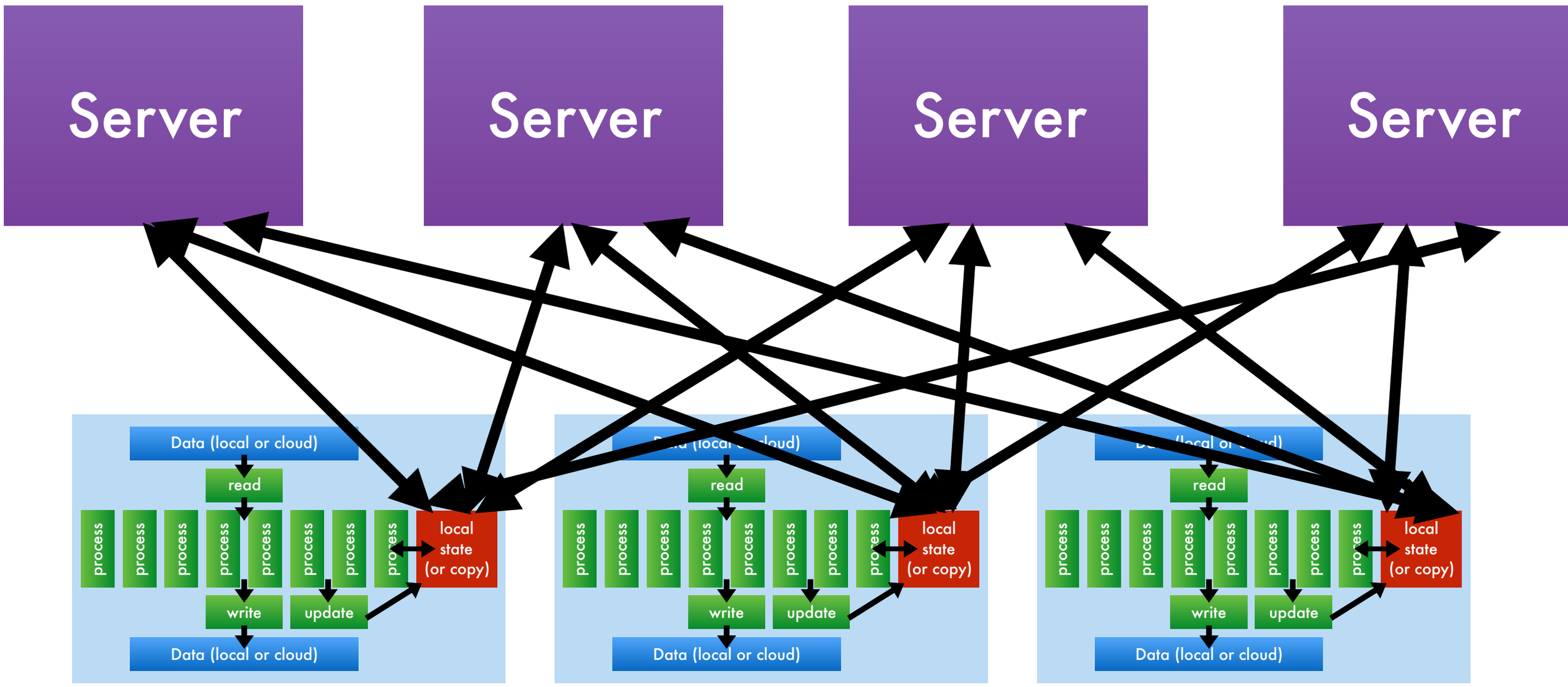
Main Contributors

- **Dave Andersen** (ParameterServer)
- **Mu Li** (MXNet, ParameterServer, FM)
- **Manzil Zaheer** (LDA)
- **Tianqi Chen** (MXNet)
- **Ziqi Liu** (Recommender, FM)
- **Jean-Baptiste Tristan** (LDA)
- **Yu-Xiang Wang** (Recommender, FM)

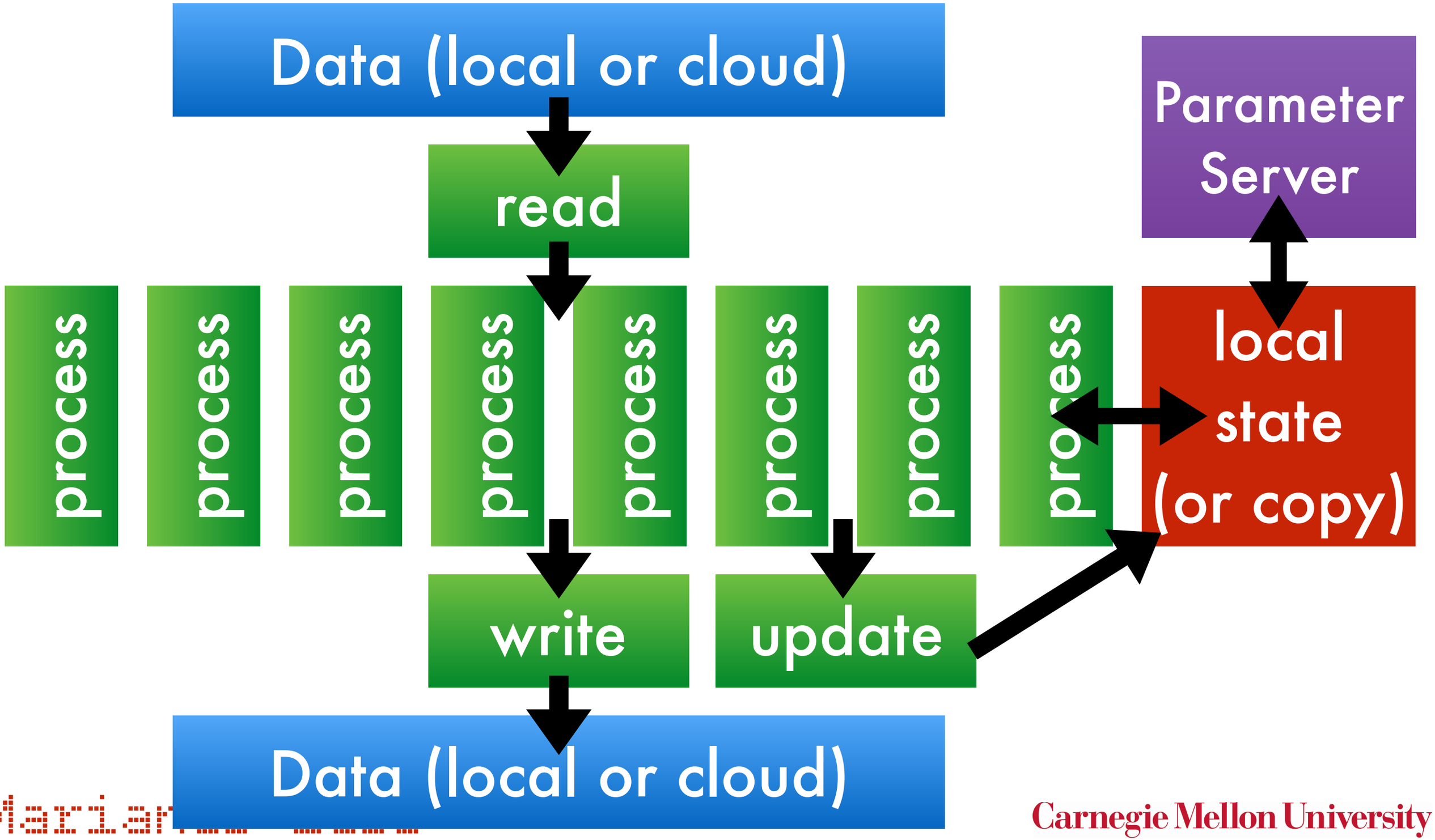
Topics

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **Large Distributed State**
  - Factorization Machines
  - Latent Variable Models
- **GPUs**
  MXNET and applications

Parameter Server



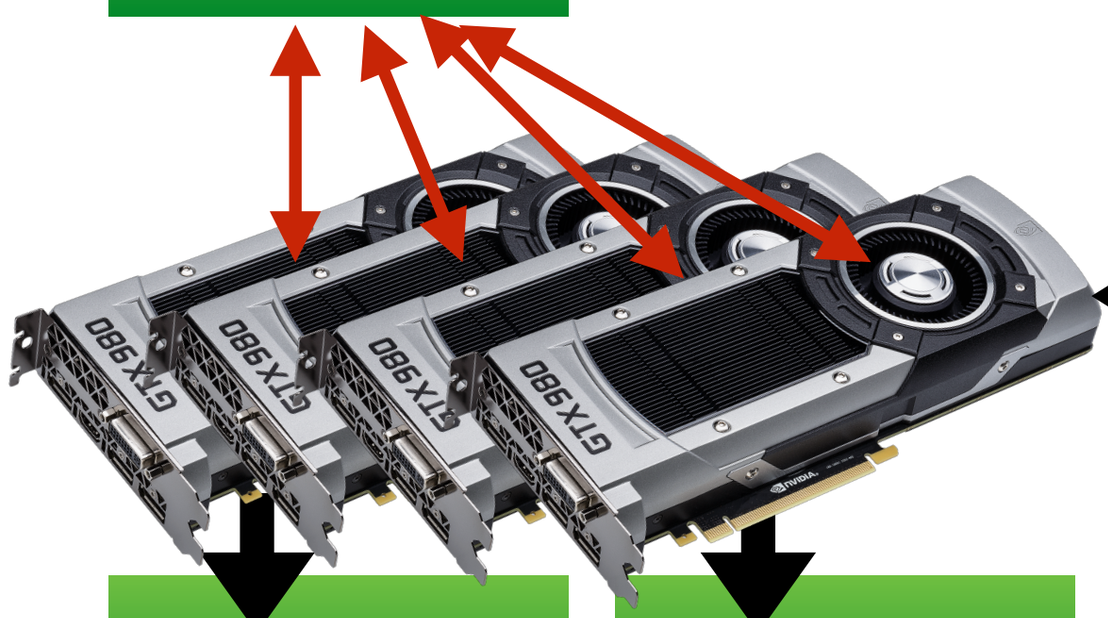
Multicore


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

GPUs (for Deep Learning)

Data (local or cloud)

read



write

update

Data (local or cloud)

Parameter Server

local state (or copy)

Details

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **Large Distributed State**
  - Factorization Machines (CTR)
  - Latent Dirichlet Allocation (LDA)
- **GPUs**
  MXNET and applications

Google machine learning

Web News Videos Books Images More Search tools

About 60,400,000 results (0.39 seconds)

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

Qualcomm Machine Learning - qualcomm.com
Ad www.qualcomm.com/WhyWait
4.3 ★★★★½ 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+

Estimate Click Through Rate

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!

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

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

sparse models for advertising

- 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

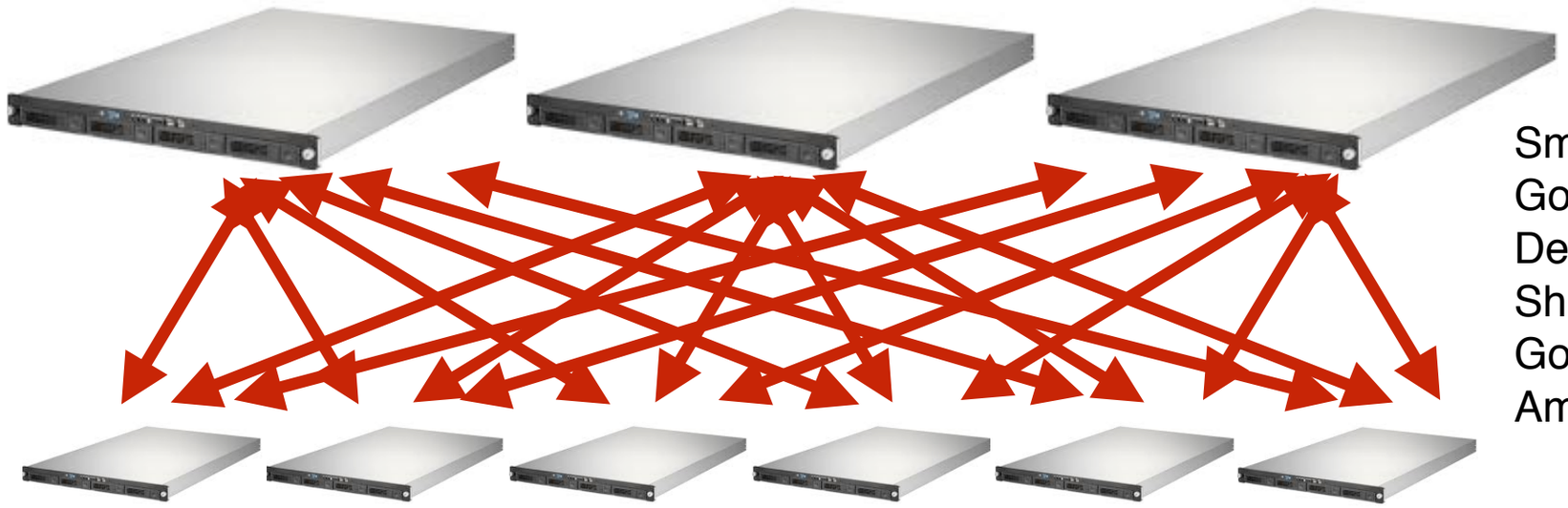
$$\operatorname*{argmin}_w \|w\|_1 + \frac{\gamma}{2} \|w - (w_t - \eta g_t)\|_2$$

- 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

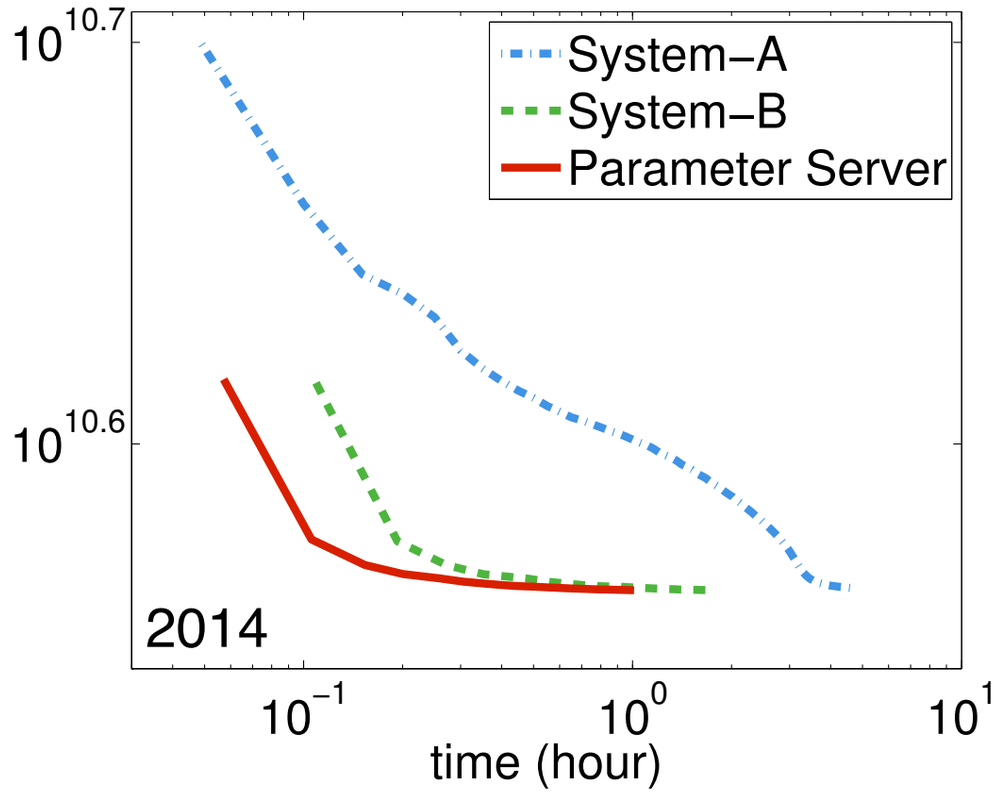


Smola & Narayanamurthy, 2010, VLDB
Gonzalez et al., 2012, WSDM
Dean et al, 2012, NIPS
Shervashidze et al., 2013, WWW
Google, Baidu, Facebook,
Amazon, Yahoo, Microsoft

- 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)`
- **Lots of tricks for bandwidth saving**

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 per machine**



Details

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **Large Distributed State**
  - Factorization Machines (CTR)
  - Latent Dirichlet Allocation (LDA)
- **GPUs**
  MXNET and applications

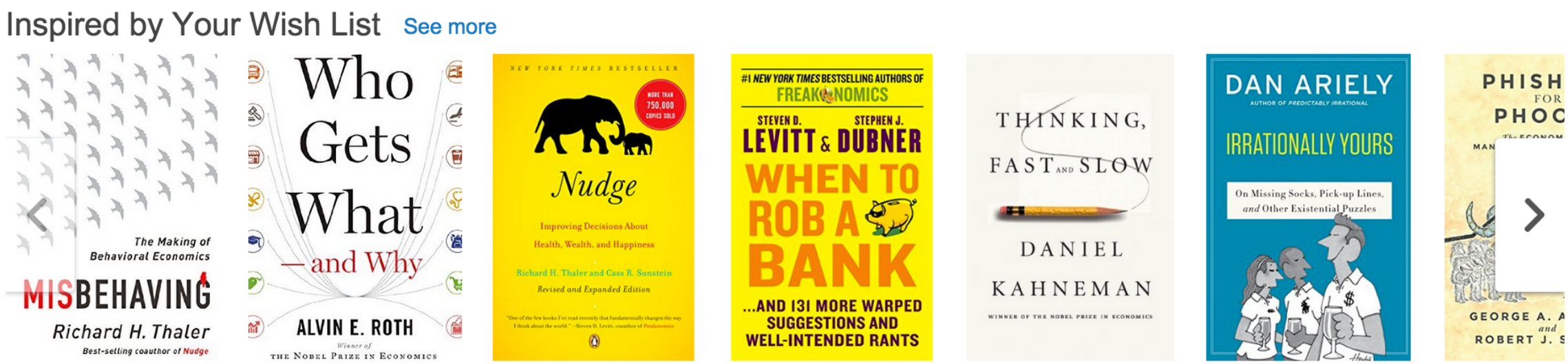
Recommender Systems

- 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

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

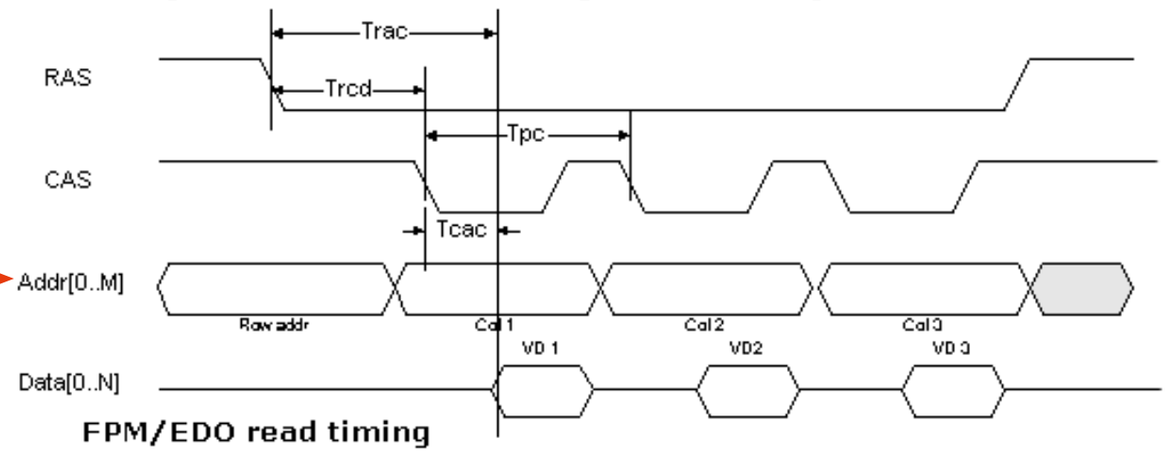
- Update operations

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

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

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

memory subsystem



FPM/EDO read timing

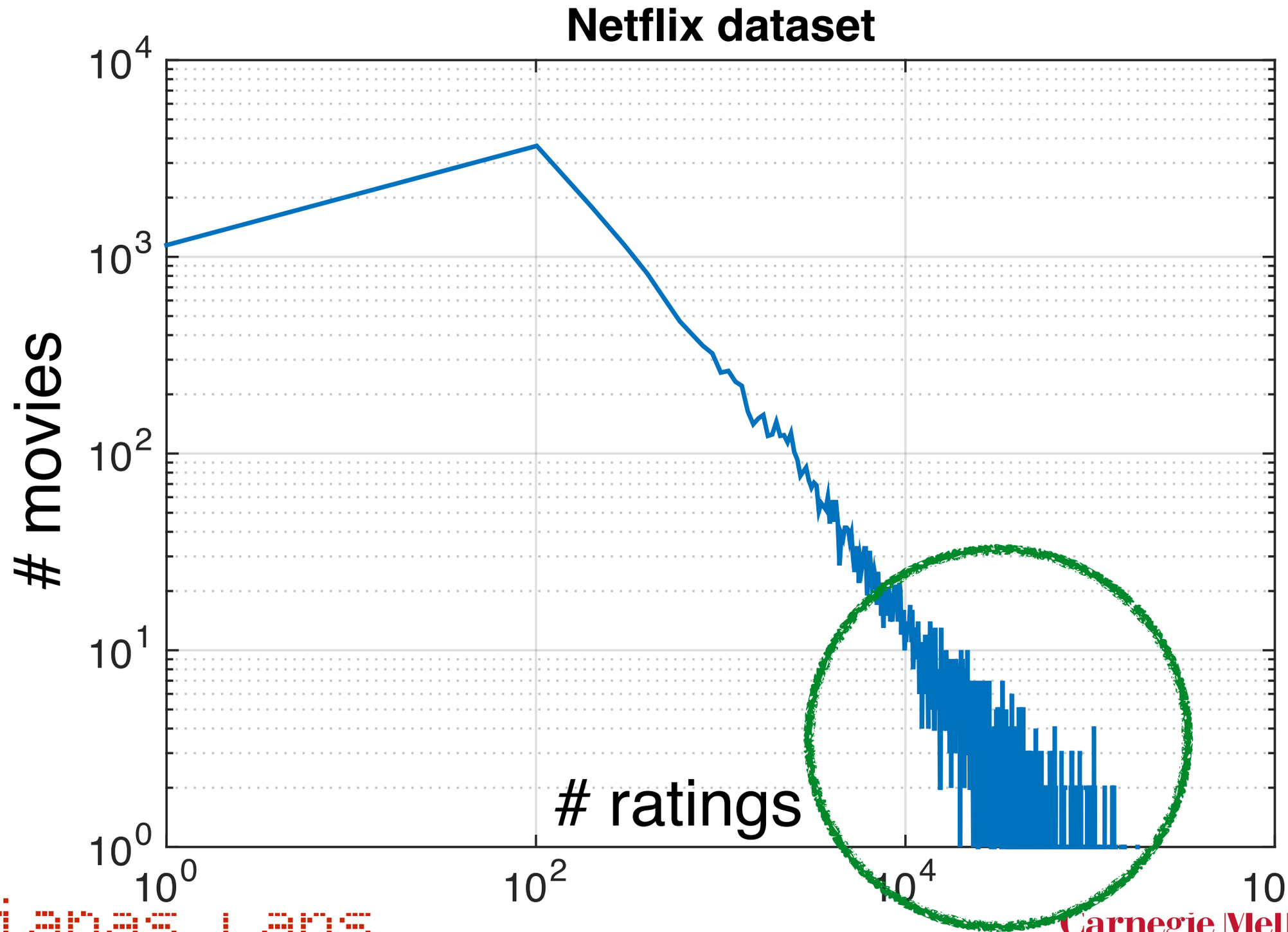
This should be cheap …

- 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



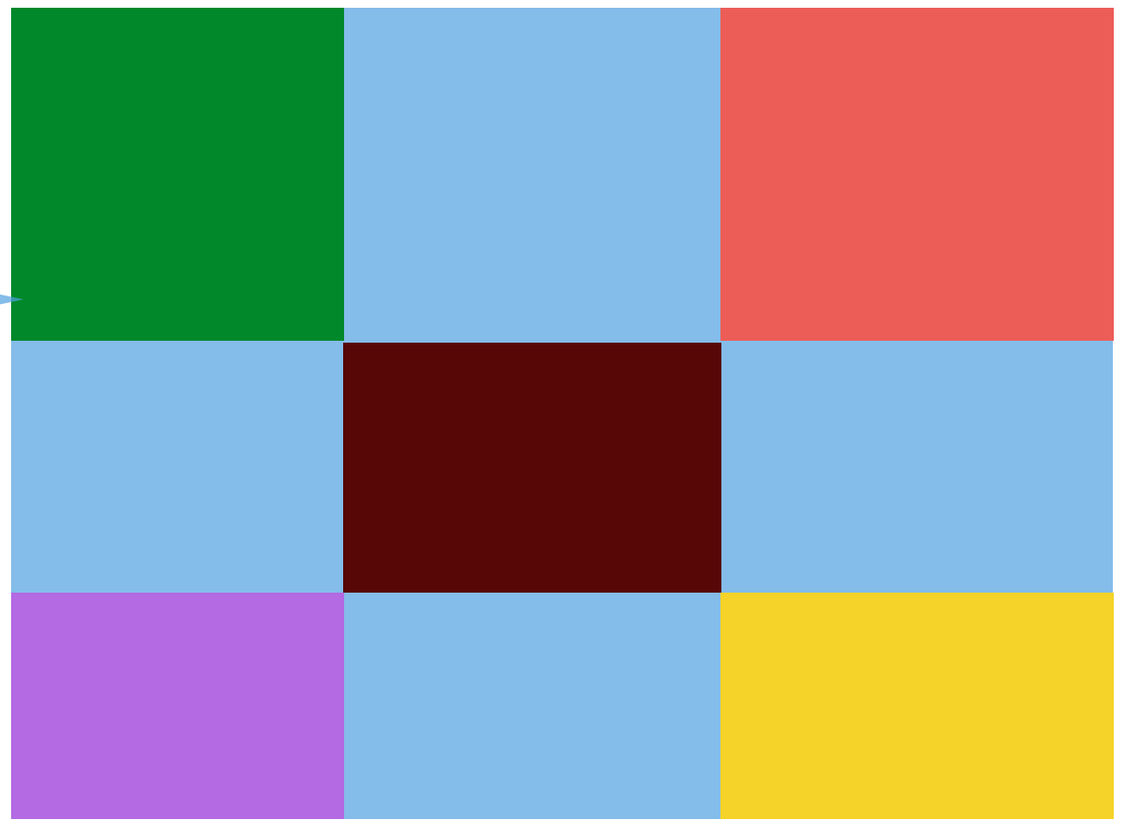
Key Ideas

- **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

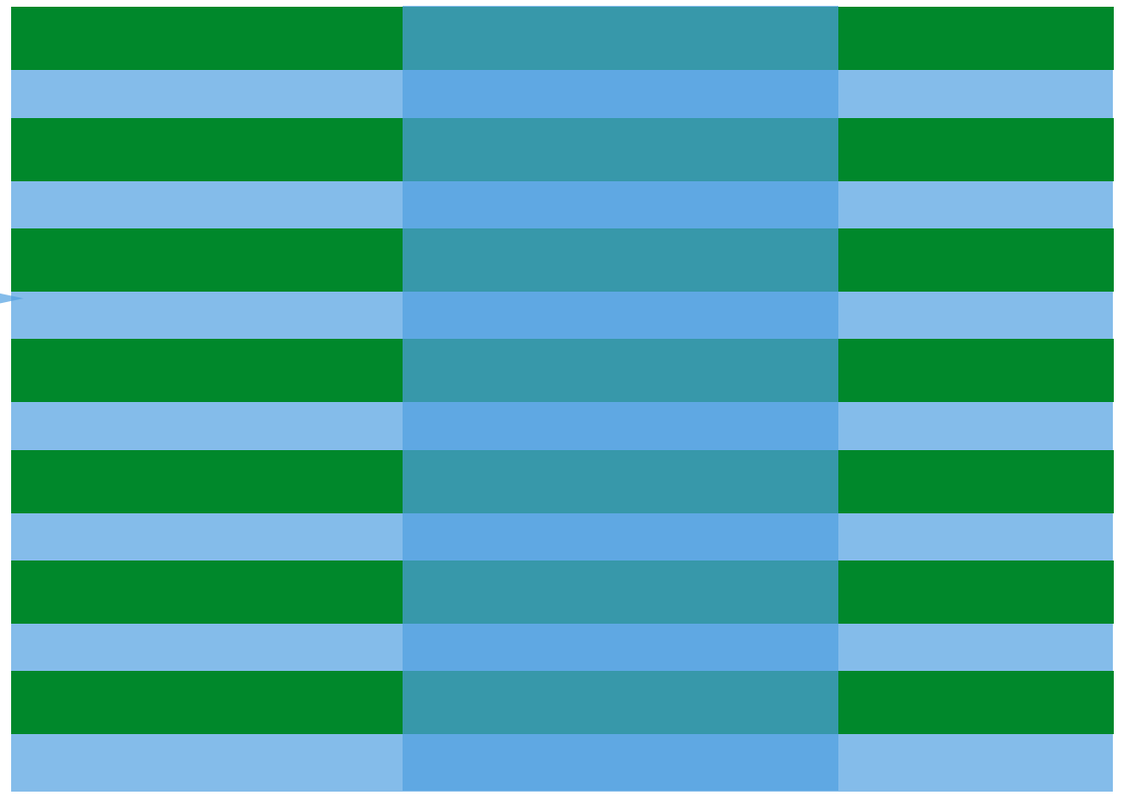
GraphChi Partitioning



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%

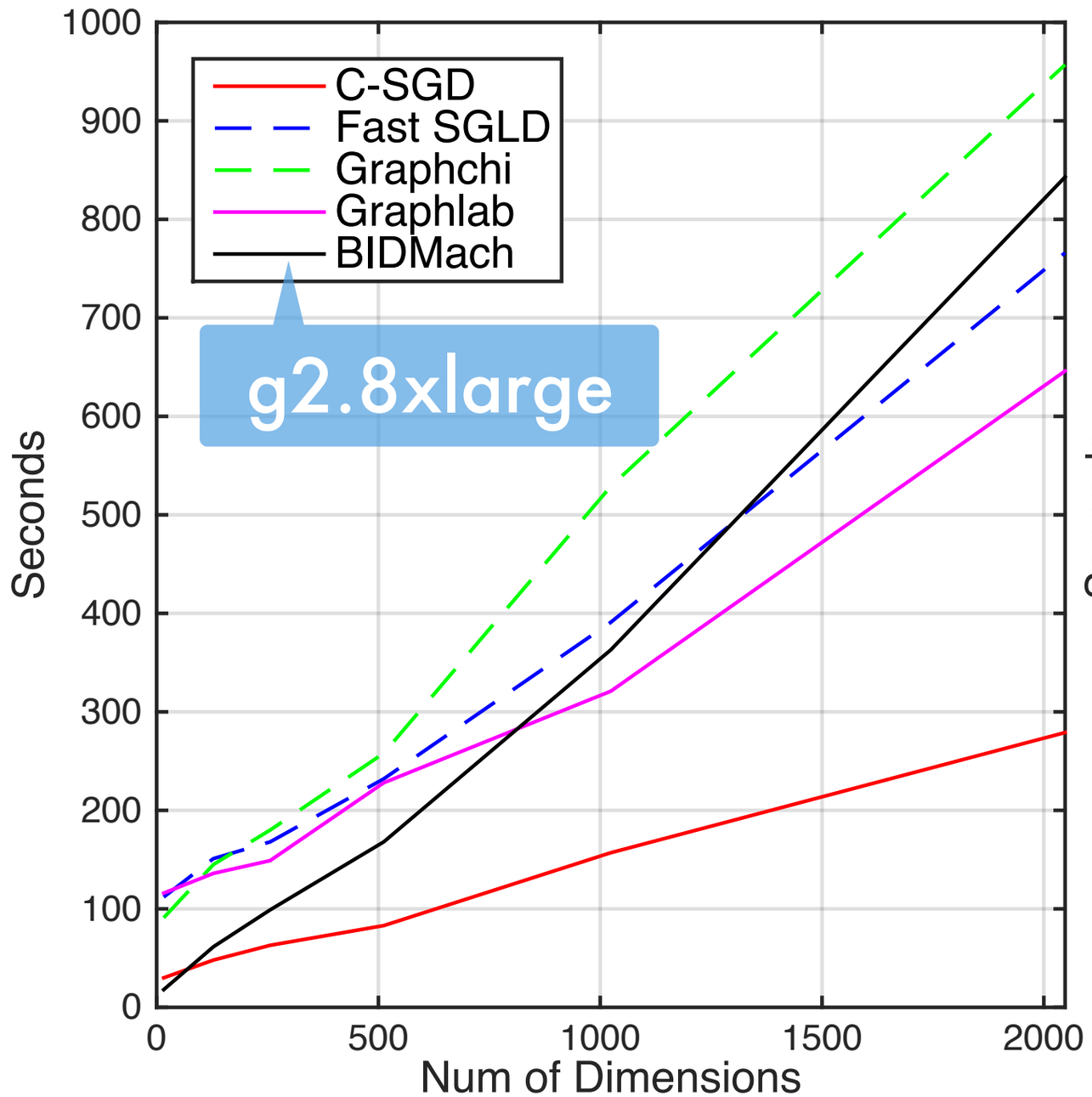
Key Ideas

SC-SGD partitioning

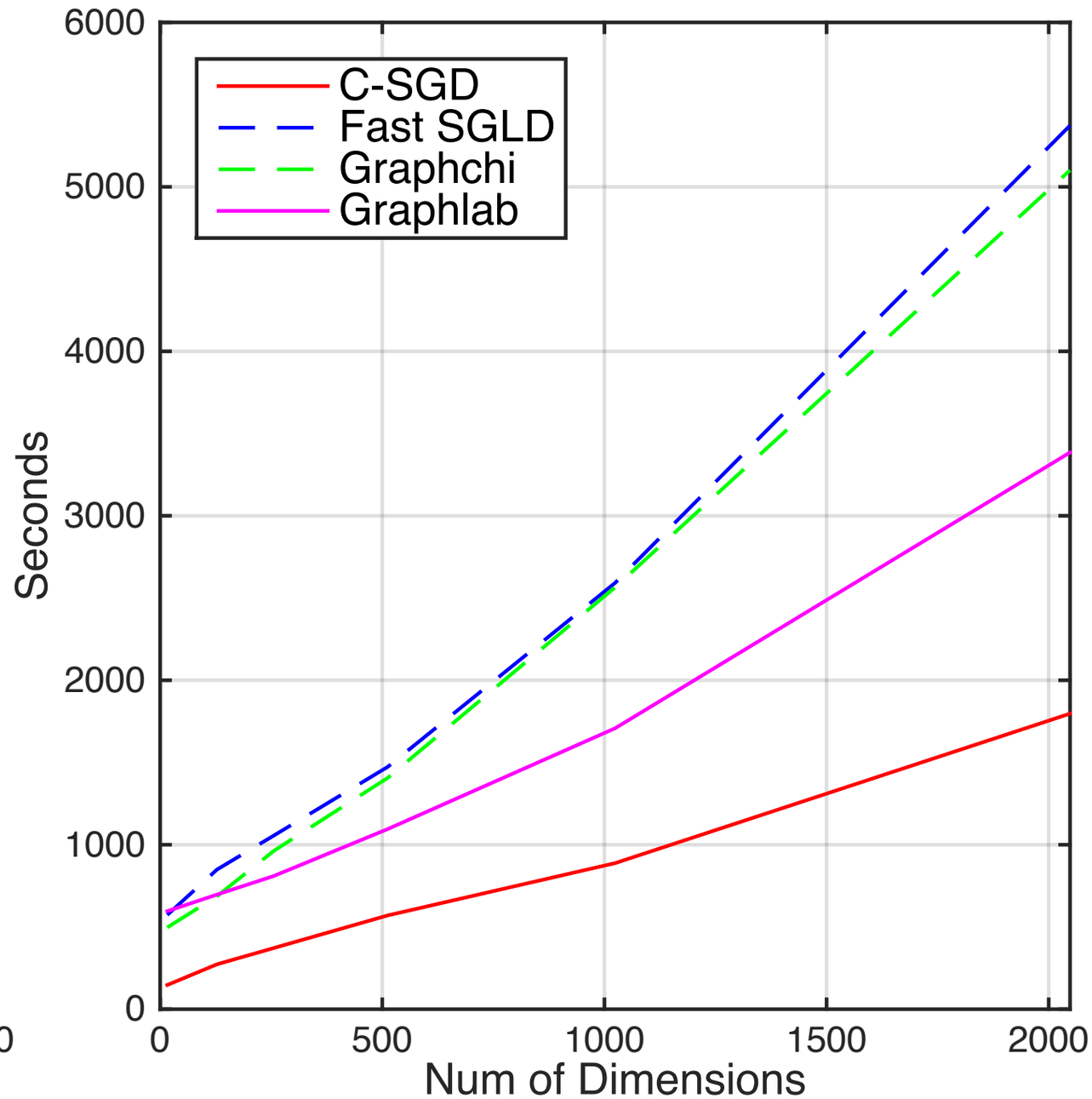


K	SC-SGD		GraphChi	
	L1 Cache	L3 Cache	L1 Cache	L3 Cache
16	2.84%	0.43%	12.77%	2.21%
256	2.85%	0.50%	12.89%	2.34%
2048	3.3%	1.7%	15%	9.8%

Speed (c4.8xlarge)



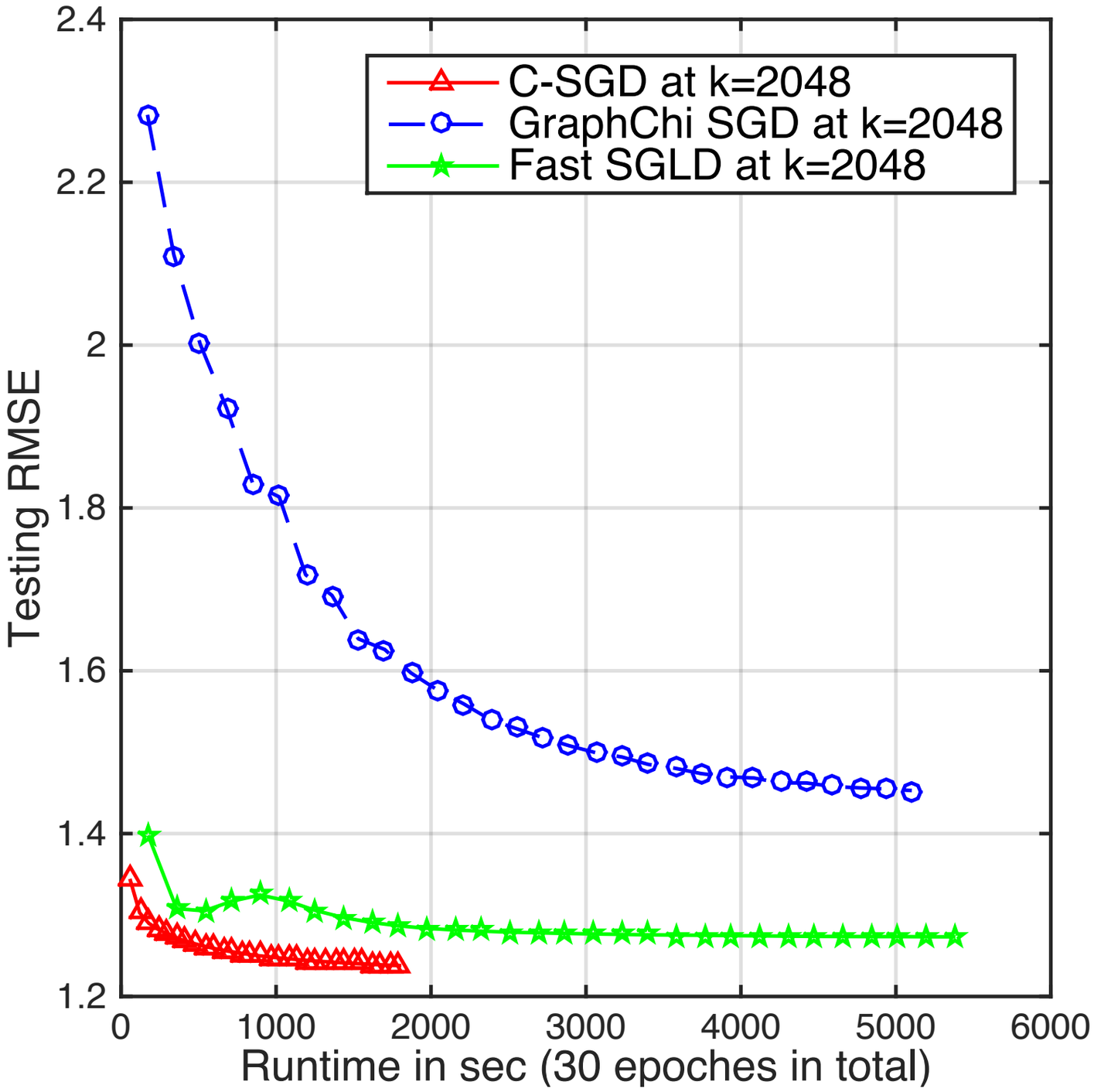
Netflix - 100M, 15 iterations



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)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **Large Distributed State**
  - Factorization Machines (CTR)
  - Latent Dirichlet Allocation (LDA)
- **GPUs**
  MXNET and applications

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 ★★★★½ 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+

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.

A Linear Model is not enough

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)

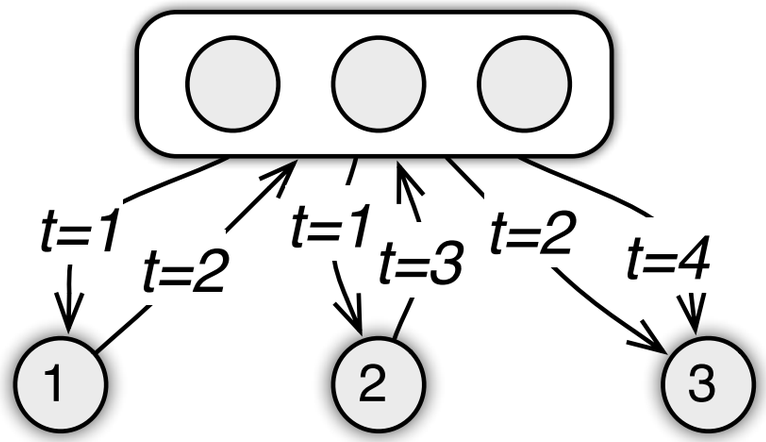
$$f(x) = \langle w, x \rangle + \sum_{i<j} x_i x_j \operatorname{tr}\left(V_i^{(2)} \otimes V_j^{(2)}\right) + \sum_{i<j<k} x_i x_j x_k \operatorname{tr}\left(V_i^{(3)} \otimes V_j^{(3)} \otimes V_k^{(3)}\right) + \dots$$

memory hog

too large for individual machine

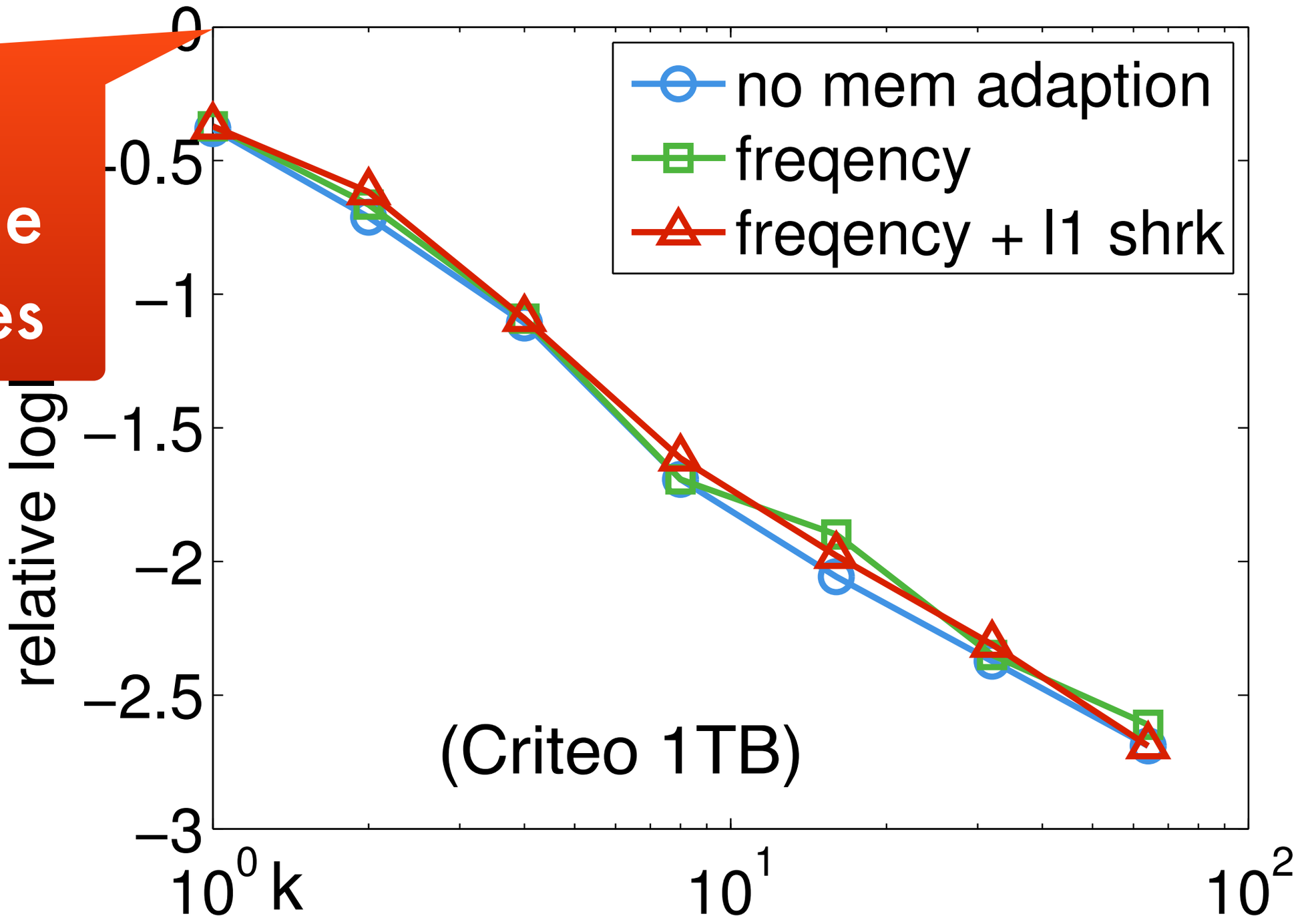
Prefetching to the rescue

- **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)

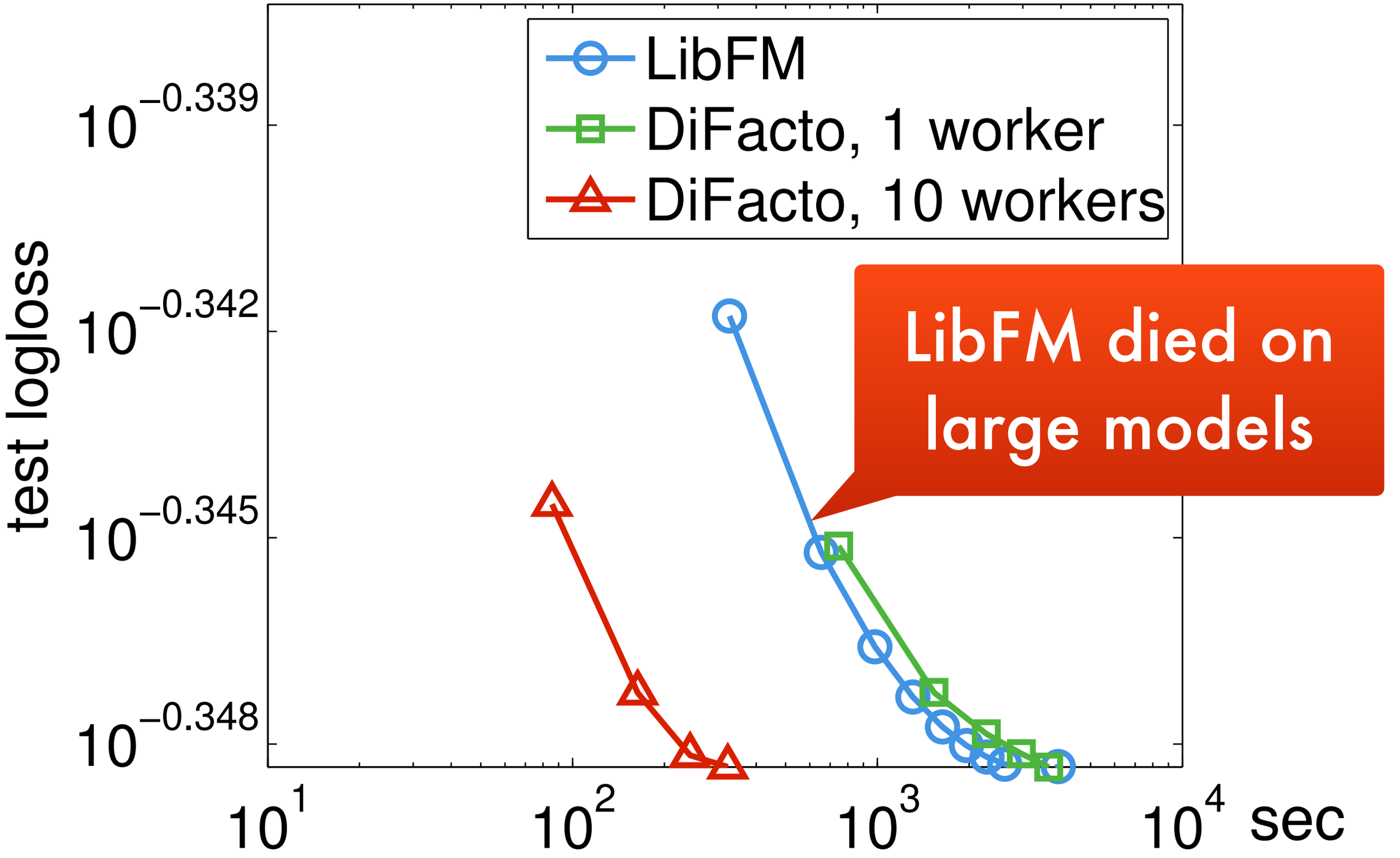


Better Models

what everyone else does

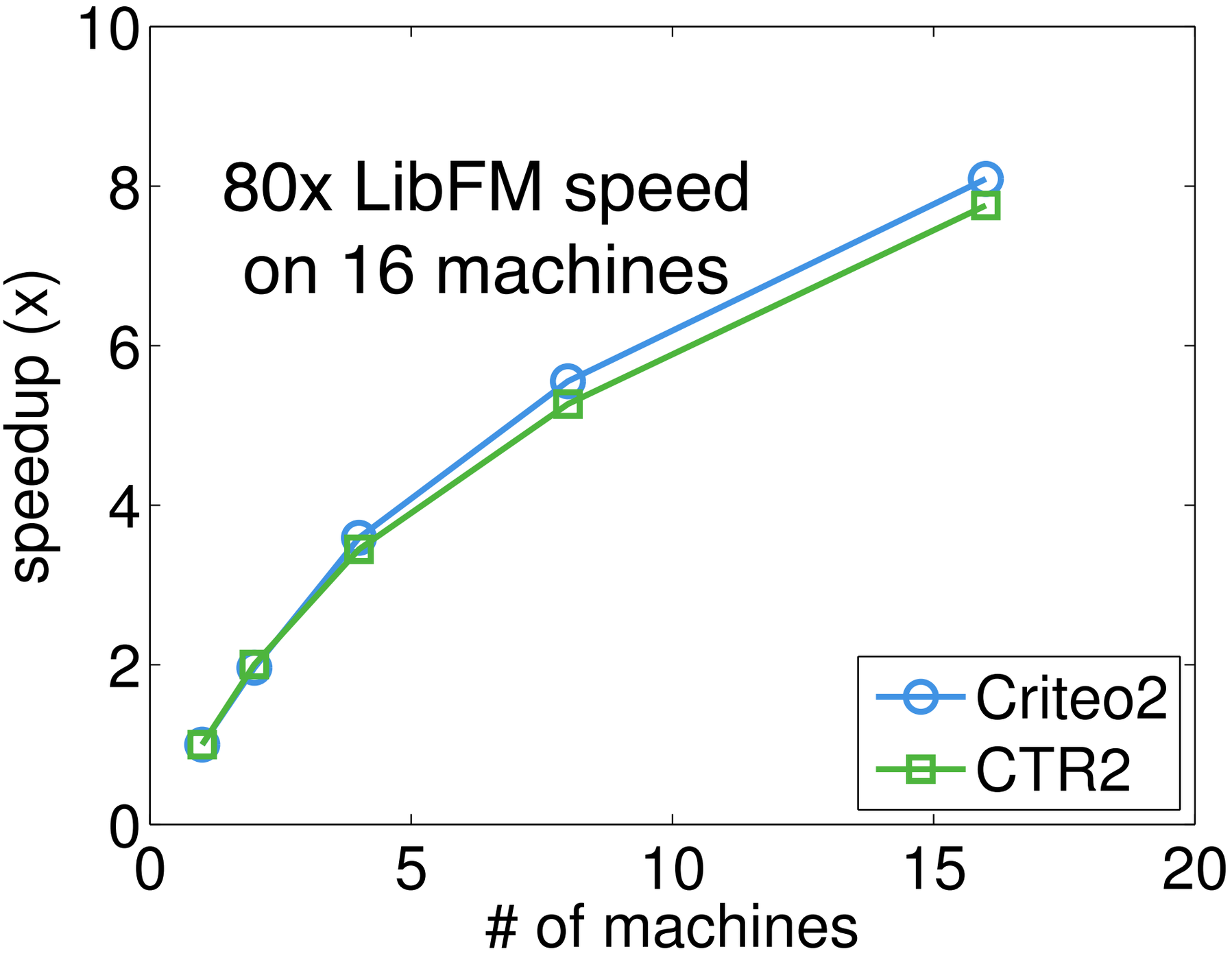


Faster Solver (small Criteo)



Multiple Machines

Li, Wang, Liu, Smola, WSDM'16

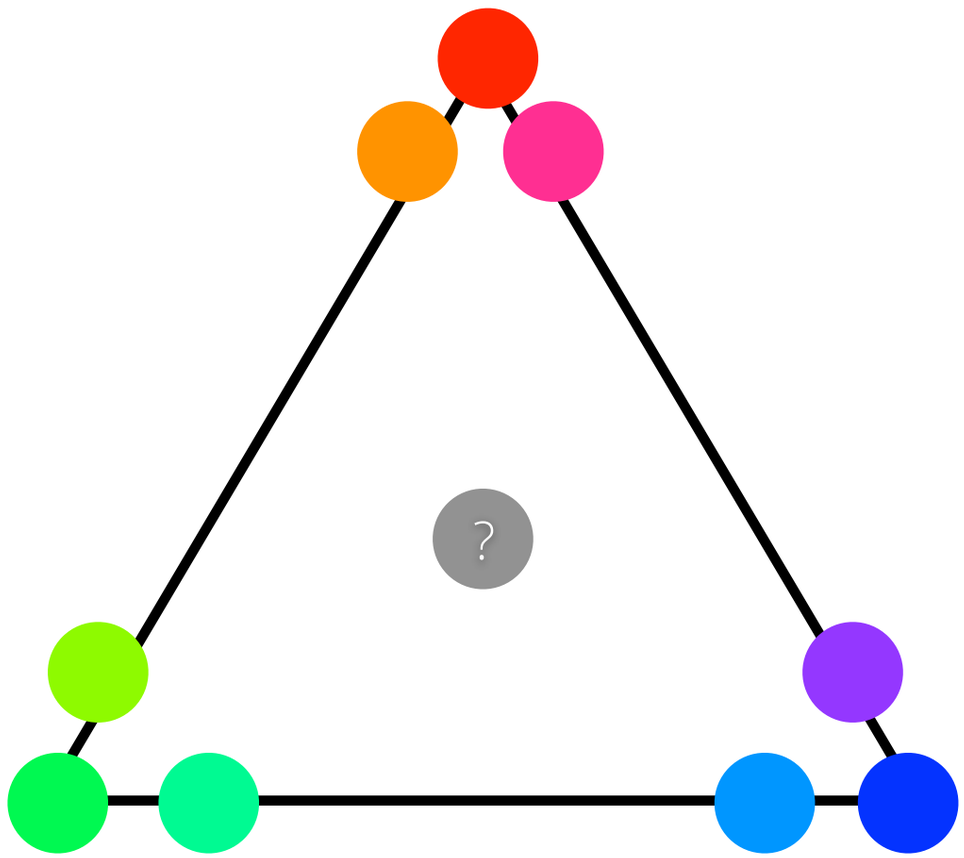


Details

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **Large Distributed State**
  - Factorization Machines (CTR)
  - Latent Dirichlet Allocation (LDA)
- **GPUs**
  MXNET and applications

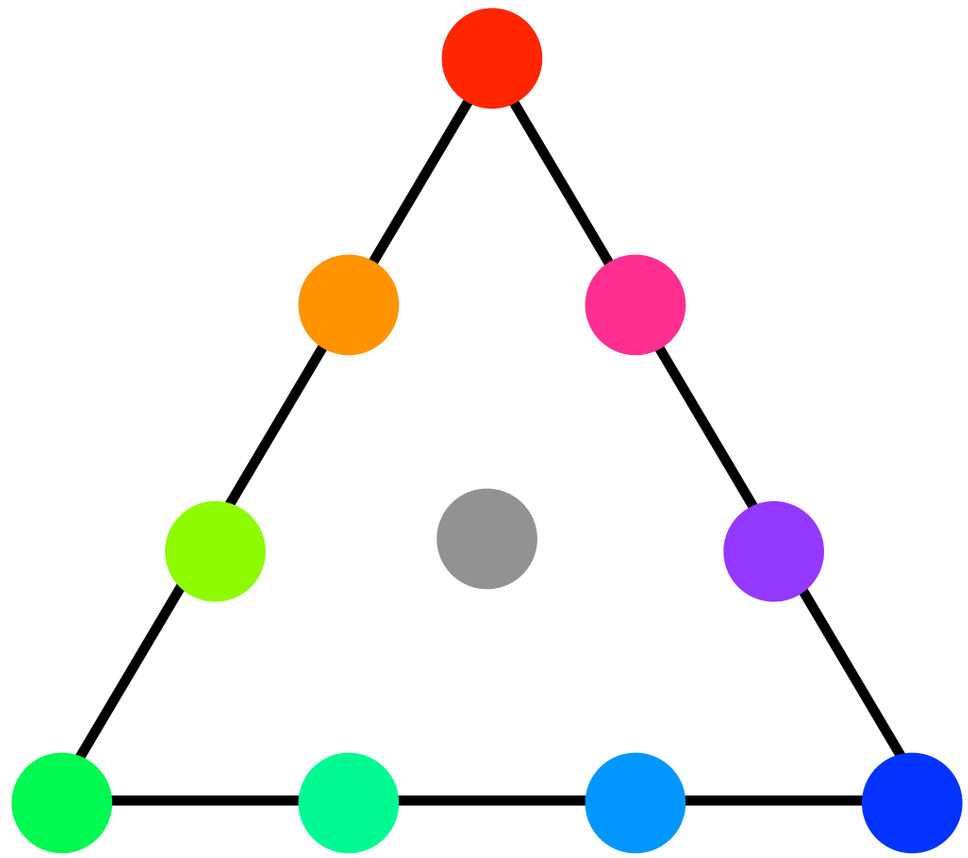
Clustering & Topic Models

Clustering



group objects
by prototypes

Topics



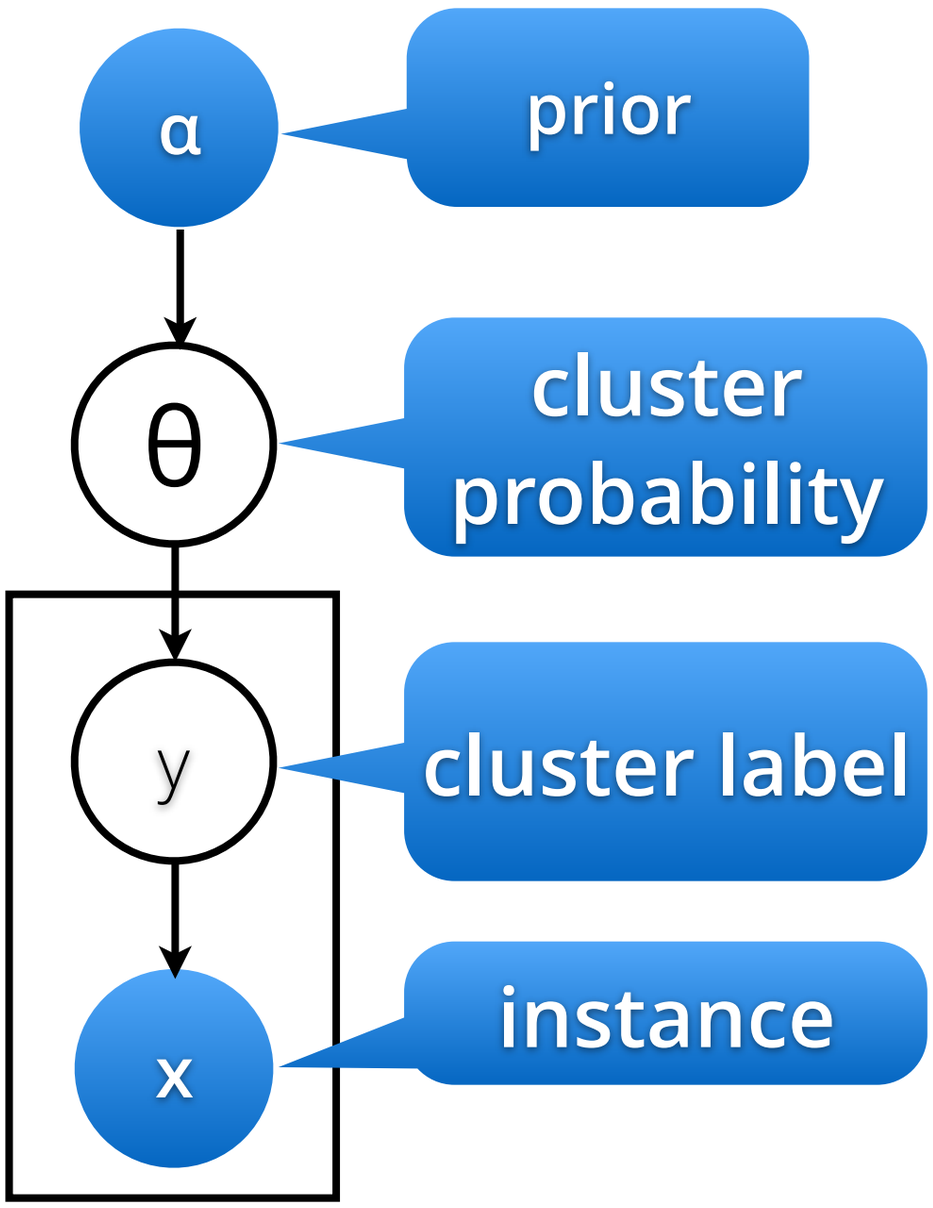
decompose objects
into prototypes

Google

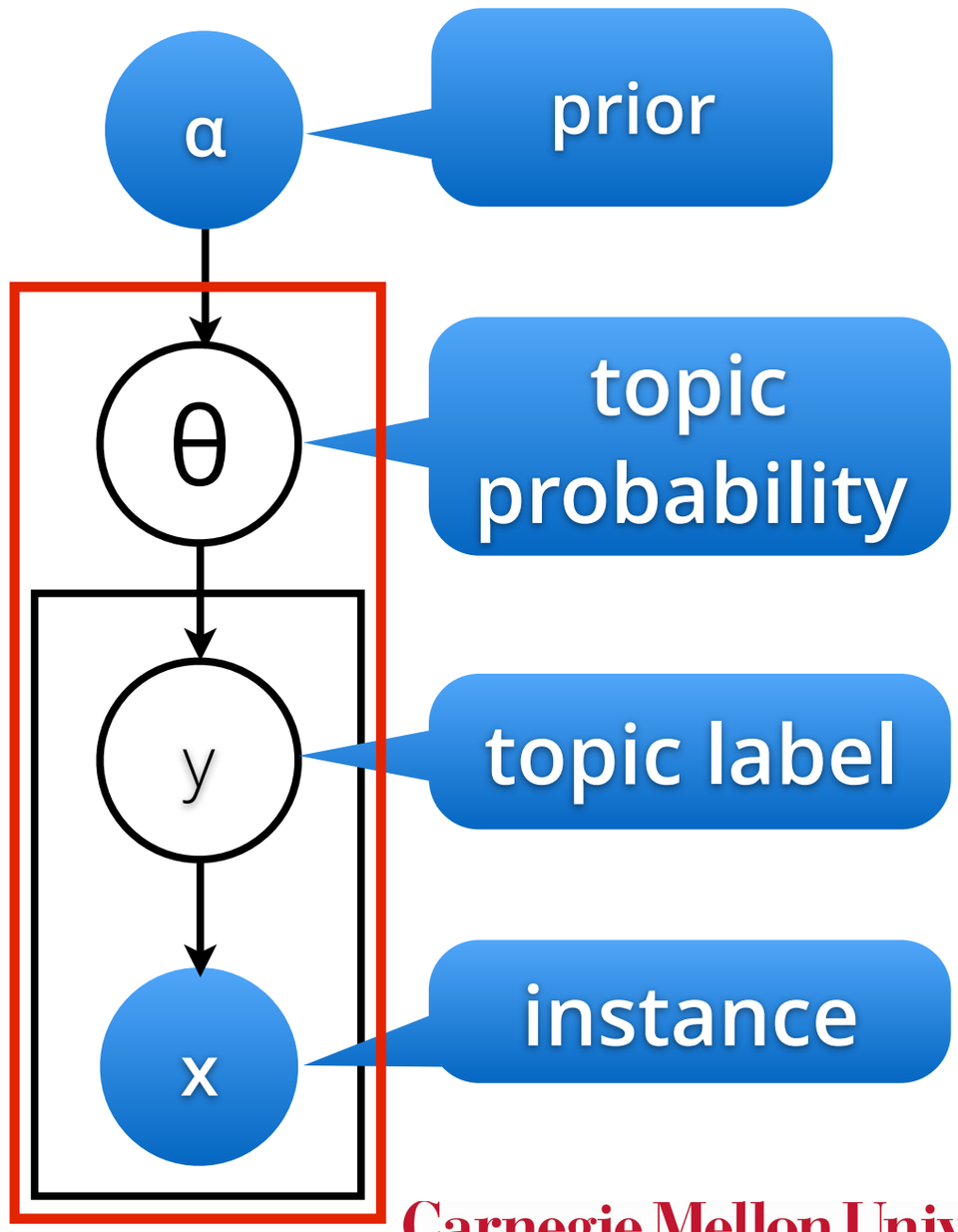
Carnegie Mellon University

Clustering & Topic Models

clustering



Latent Dirichlet Allocation



Google

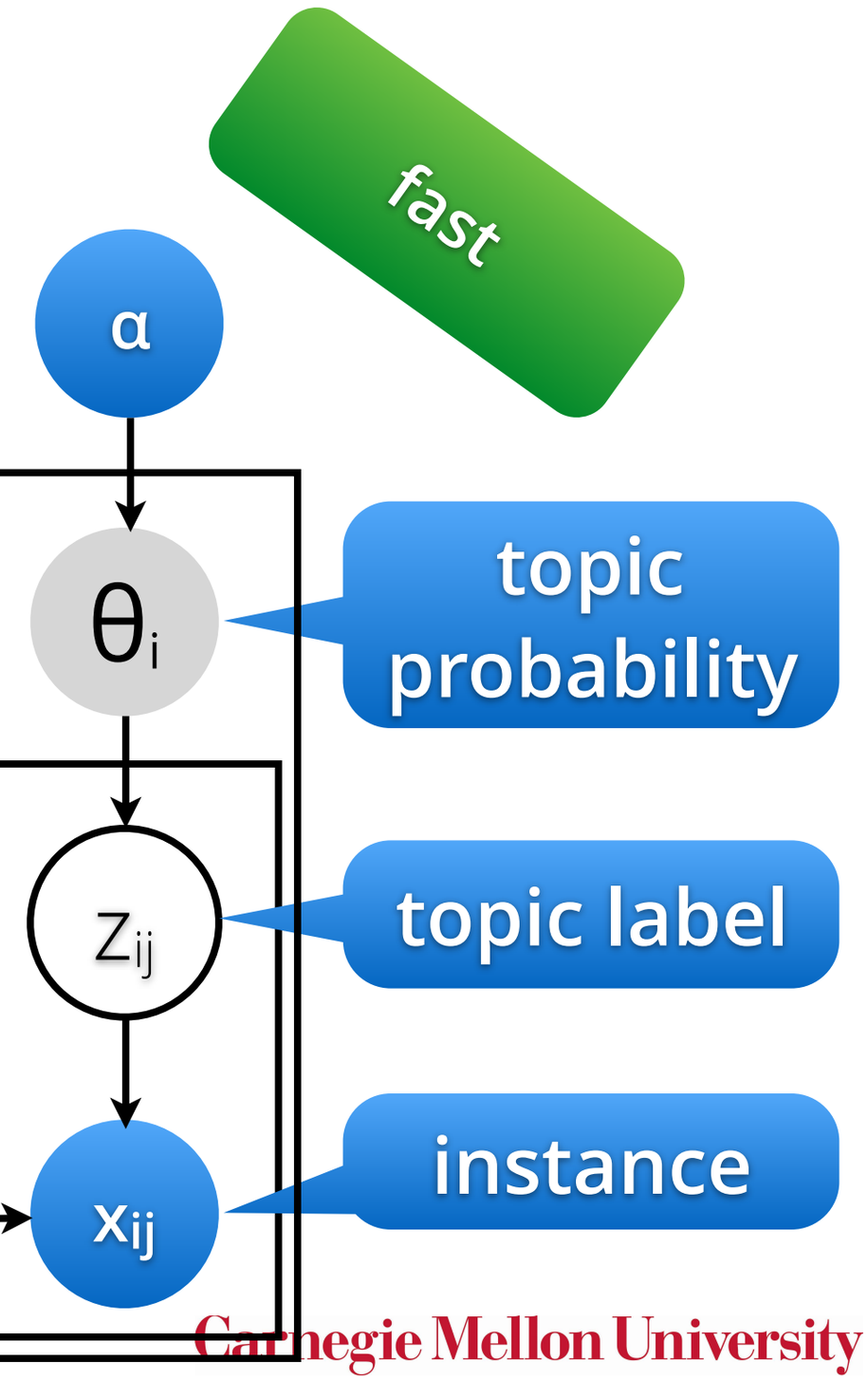
Carnegie Mellon University

Collapsed Sampler

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\}|\beta)$$

sample z sequentially



Collapsed Sampler

Griffiths & Steyvers, 2005

$$p(z, x|\alpha, \beta)$$

$$= \prod_{i=1}^{m} p(z_i|\alpha) \prod_{k=1}^{k} p(\{x_{ij}|z_{ij} = k\}|\beta)$$

$$\frac{n^{-ij}(t,d) + \alpha_t}{n^{-i}(d) + \sum_t \alpha_t}$$

$$\frac{n^{-ij}(t,w) + \beta_t}{n^{-i}(t) + \sum_t \beta_t}$$

fast mixing

α

θ_i

topic probability

z_{ij}

topic label

x_{ij}

instance

β

ψ_k

language prior

Google

Carnegie Mellon University

Gibbs Sampler

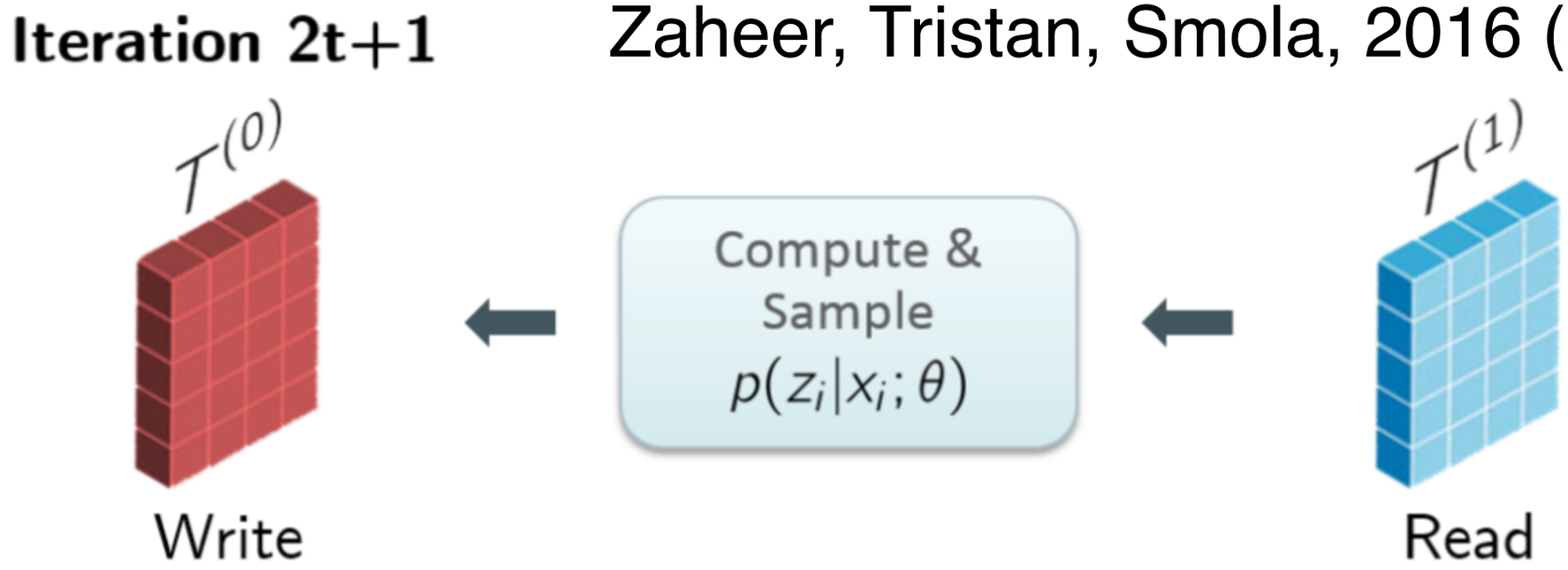
- For 1000 iterations do
  - For each document do
    - For each word in the document do
      - Resample topic for the word
      - Lock (word,topic) table
      - Update local (document, topic) table
      - Update (word,topic) table
      - Unlock (word,topic) table

this kills parallelism

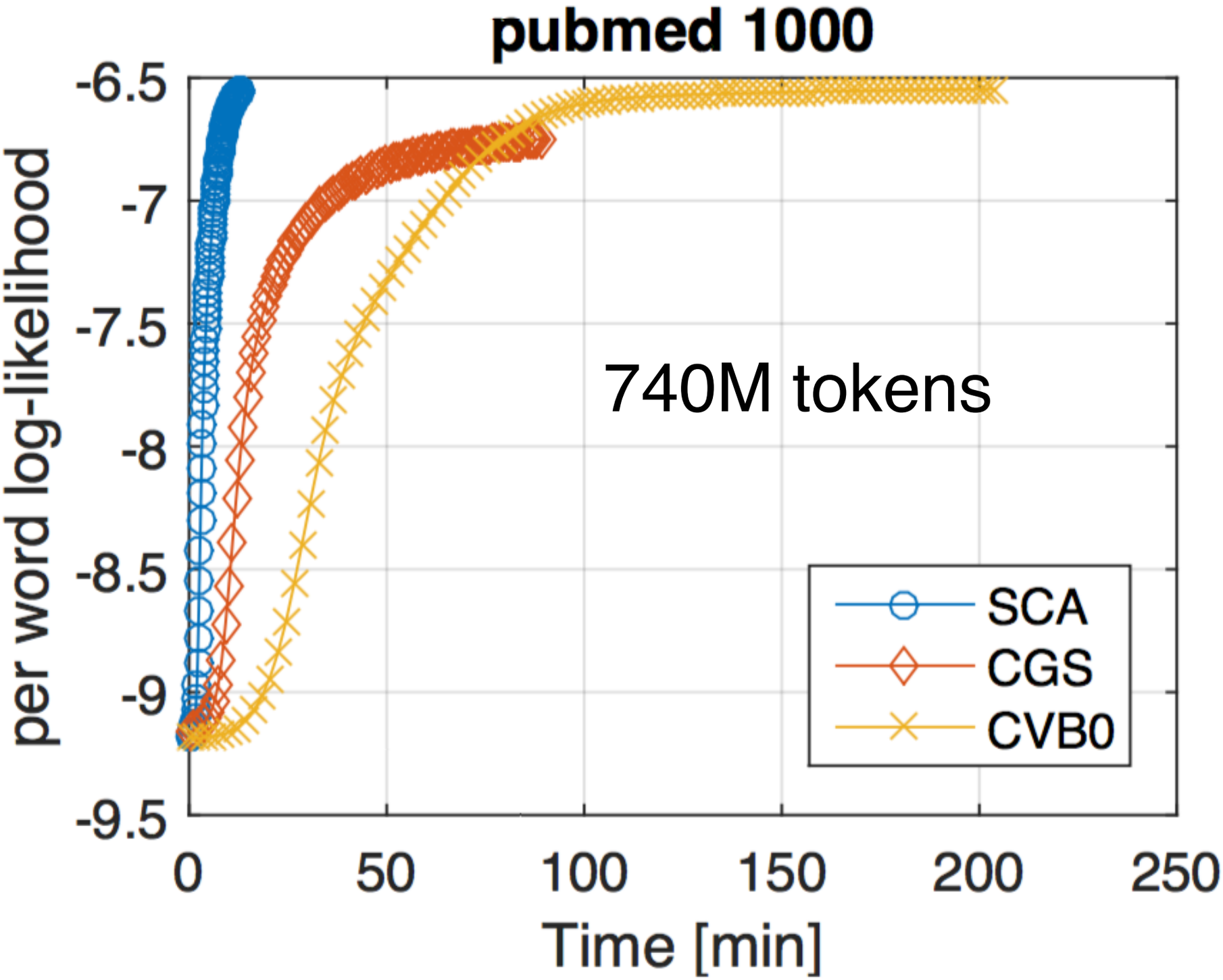
Stochastic Cellular Automata

- **Locks are evil**
- Collapsing everything is not necessary
  - Collapse documents
  - Mix and alternate language model
- Exchange between machines

Zaheer, Tristan, Smola, 2016 (submitted)



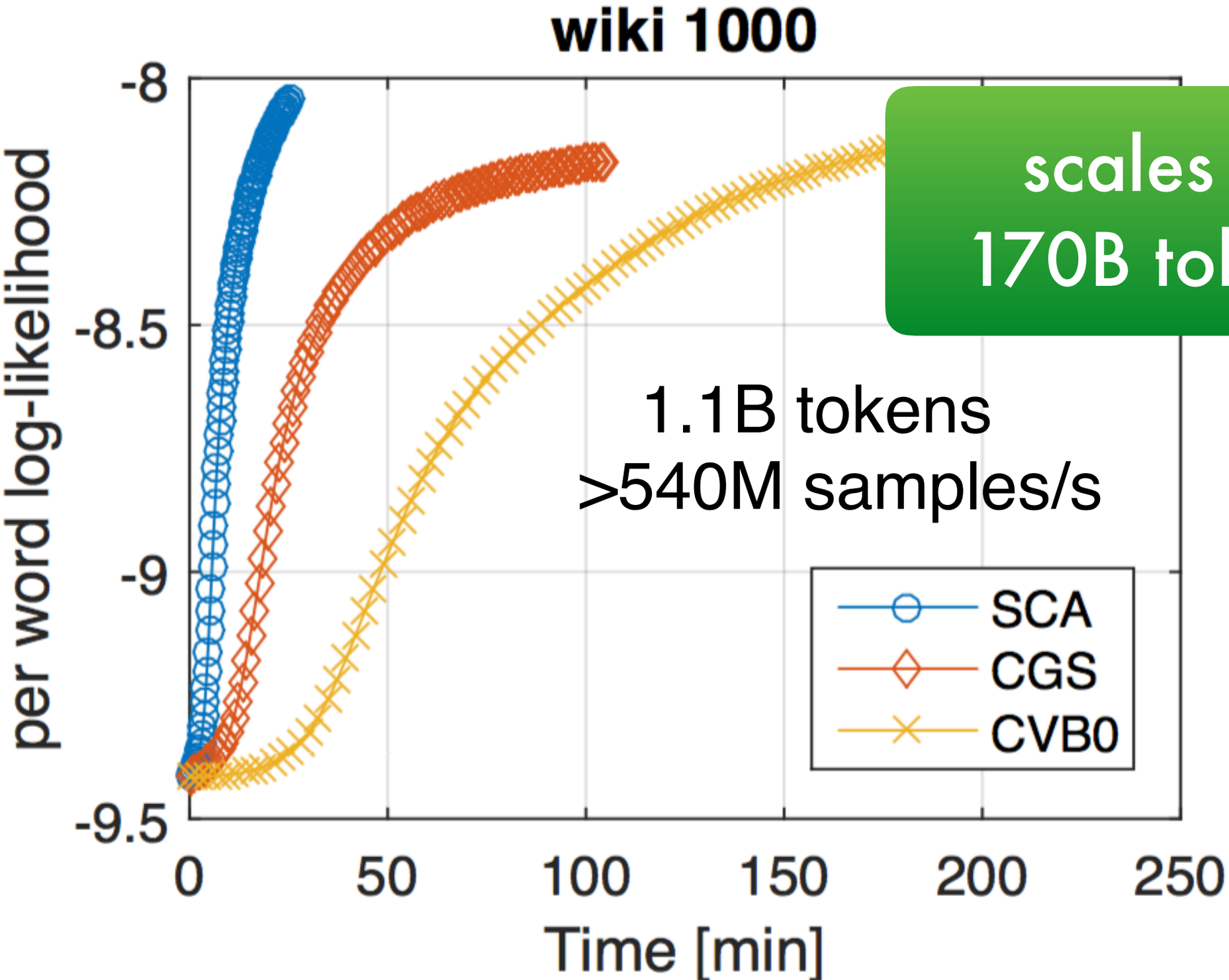
Speed (4 machines, 1000 topics)



Mallet

on University

Speed (4 machines, 1000 topics)



scales to
170B tokens

Mallet

n University

Even more Speed

Algorithm	Dataset	Number of topics	Size of vocabulary	Number of documents	Number of tokens	Hardware	Year	Speed (tokens/s)
Yahoo! LDA	PubMed data	2K	140K	8.2M	797M	10 machines Hadoop cluster	2010	< 13M
lightLDA	Bing “web chunk” data	1,000K	50K	1,200M	200,000M	24 machines (480 cores)	2014	60M
F+Nomad LDA	Amazon product reviews (from SNAP)	1K	1,680K	30M	1,500M	32 Xeon E5 CPUs (640 cores)	2014	110M
ESCA	**100 copies of Wikipedia (kept on solid state disks)**	**1K**	**290K**	**677M**	**128,000M**	**20 nodes Amazon Cloud c4.8xlarge**	**2015**	**1,000M**

Details

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **Large Distributed State**
  - Factorization Machines (CTR)
  - Latent Dirichlet Allocation (LDA)
- **GPUs**
  MXNET and applications

MXNET - Distributed Deep Learning

- Several deep learning frameworks
  - **Torch7**: matlab-like tensor computation in Lua
  - **Theano**: symbolic programming in Python
  - **Caffe**: configuration in Protobuf
- **Problem - need to learn another system for a different programing flavor**
- **MXNET**
  - provides programming interfaces in a consistent way
  - multiple languages: C++/Python/R/Julia/…
  - fast, memory efficient, and distributed training

Programming Interfaces

- **Tensor Algebra Interface**
  - compatible (e.g. NumPy)
  - GPU/CPU/ARM (mobile)
- **Symbolic Expression**
  - Easy Deep Network design
  - Automatic differentiation
- Mixed programming
  - CPU/GPU
  - symbolic + local code

Python

```
>>> import mxnet as mx
>>> a = mx.nd.ones((2, 3),
... mx.gpu(2))
>>> print (a * 2).asnumpy()
[[ 2.  2.  2.]
 [ 2.  2.  2.]]
```

Julia

```
using MXNet
mlp = @mx.chain mx.Variable()          =>
  mx.FullyConnected(num_hidden=128)    =>
  mx.Activation(act_type=:relu)        =>
  mx.FullyConnected(num_hidden=64)     =>
  mx.Activation(act_type=:relu)        =>
  mx.FullyConnected(num_hidden=10)     =>
  mx.Softmax()
```

6 layers definition

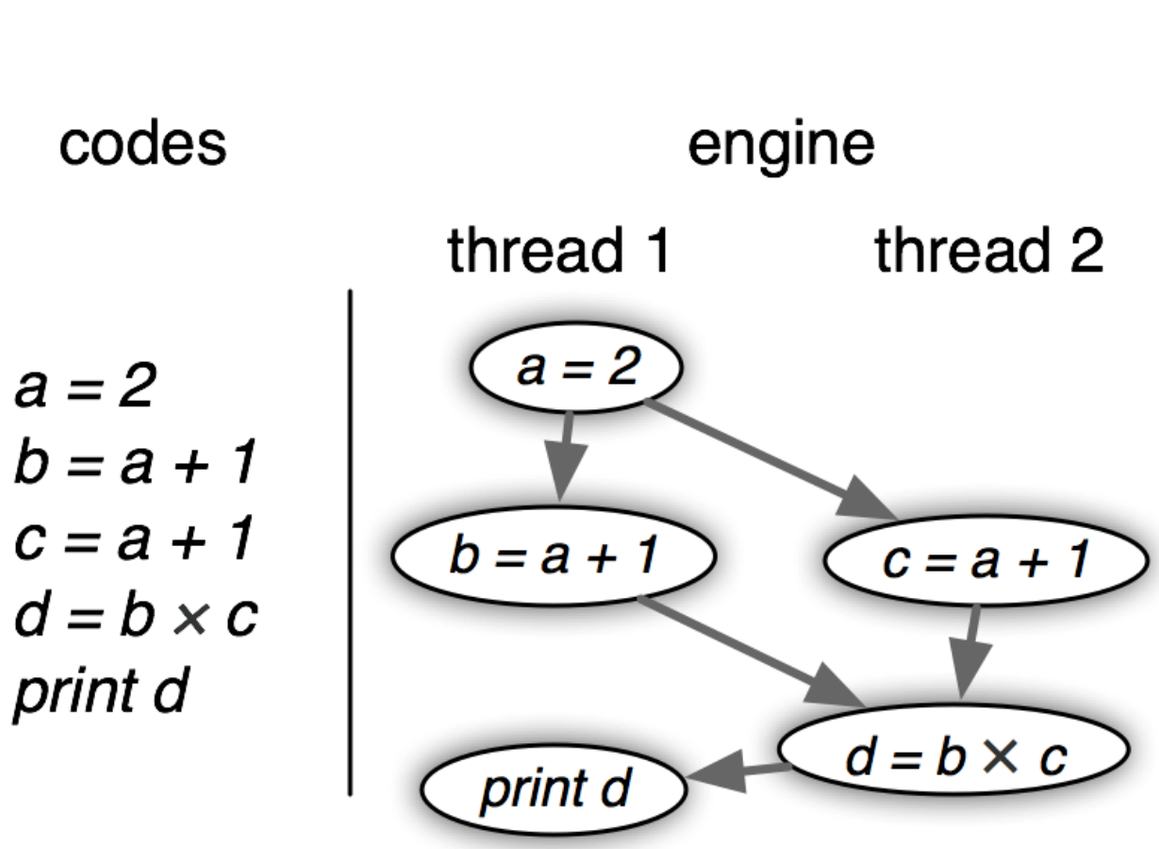
multi GPU

train it

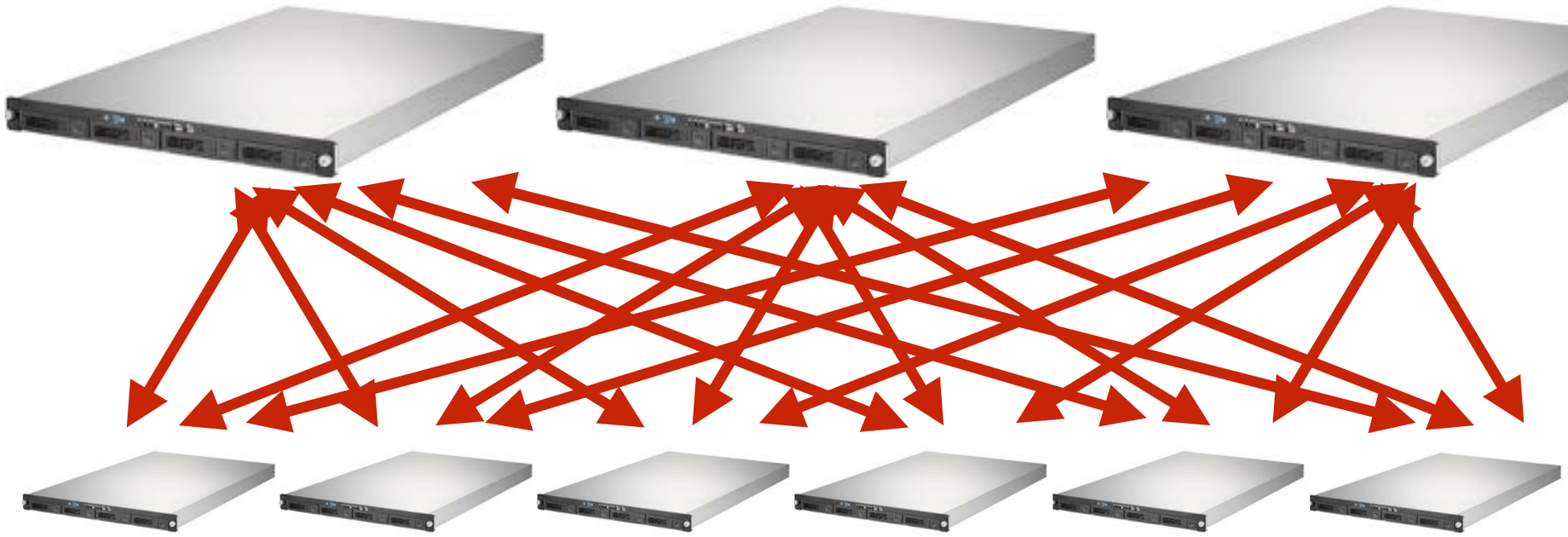
```
from data import ilsvrc12_iterator
import mxnet as mx

## define alexnet
input_data = mx.symbol.Variable(name="data")
# stage 1
conv1 = mx.symbol.Convolution(data=input_data, kernel=(11, 11), stride=(4, 4), num_filter=96)
relu1 = mx.symbol.Activation(data=conv1, act_type="relu")
pool1 = mx.symbol.Pooling(data=relu1, pool_type="max", kernel=(3, 3), stride=(2,2))
lrn1 = mx.symbol.LRN(data=pool1, alpha=0.0001, beta=0.75, knorm=1, nsize=5)
# stage 2
conv2 = mx.symbol.Convolution(data=lrn1, kernel=(5, 5), pad=(2, 2), num_filter=256)
relu2 = mx.symbol.Activation(data=conv2, act_type="relu")
pool2 = mx.symbol.Pooling(data=relu2, kernel=(3, 3), stride=(2, 2), pool_type="max")
lrn2 = mx.symbol.LRN(data=pool2, alpha=0.0001, beta=0.75, knorm=1, nsize=5)
# stage 3
conv3 = mx.symbol.Convolution(data=lrn2, kernel=(3, 3), pad=(1, 1), num_filter=384)
relu3 = mx.symbol.Activation(data=conv3, act_type="relu")
conv4 = mx.symbol.Convolution(data=relu3, kernel=(3, 3), pad=(1, 1), num_filter=384)
relu4 = mx.symbol.Activation(data=conv4, act_type="relu")
conv5 = mx.symbol.Convolution(data=relu4, kernel=(3, 3), pad=(1, 1), num_filter=256)
relu5 = mx.symbol.Activation(data=conv5, act_type="relu")
pool3 = mx.symbol.Pooling(data=relu5, kernel=(3, 3), stride=(2, 2), pool_type="max")
# stage 4
flatten = mx.symbol.Flatten(data=pool3)
fc1 = mx.symbol.FullyConnected(data=flatten, num_hidden=4096)
relu6 = mx.symbol.Activation(data=fc1, act_type="relu")
dropout1 = mx.symbol.Dropout(data=relu6, p=0.5)
# stage 5
fc2 = mx.symbol.FullyConnected(data=dropout1, num_hidden=4096)
relu7 = mx.symbol.Activation(data=fc2, act_type="relu")
dropout2 = mx.symbol.Dropout(data=relu7, p=0.5)
# stage 6
fc3 = mx.symbol.FullyConnected(data=dropout2, num_hidden=1000)
softmax = mx.symbol.SoftmaxOutput(data=fc3, name='softmax')

## data
batch_size = 256
train, val = ilsvrc12_iterator(batch_size=batch_size, input_shape=(3,224,224))

## train
num_gpus = 4
gpus = [mx.gpu(i) for i in range(num_gpus)]
model = mx.model.FeedForward(
    ctx           = gpus,
    symbol        = softmax,
    num_round     = 20,
    learning_rate = 0.01,
    momentum      = 0.9,
    wd            = 0.00001)
model.fit(X = train, eval_data = val, batch_end_callback = mx.callback.Speedometer(batch_size=batch_size))
```

Engine

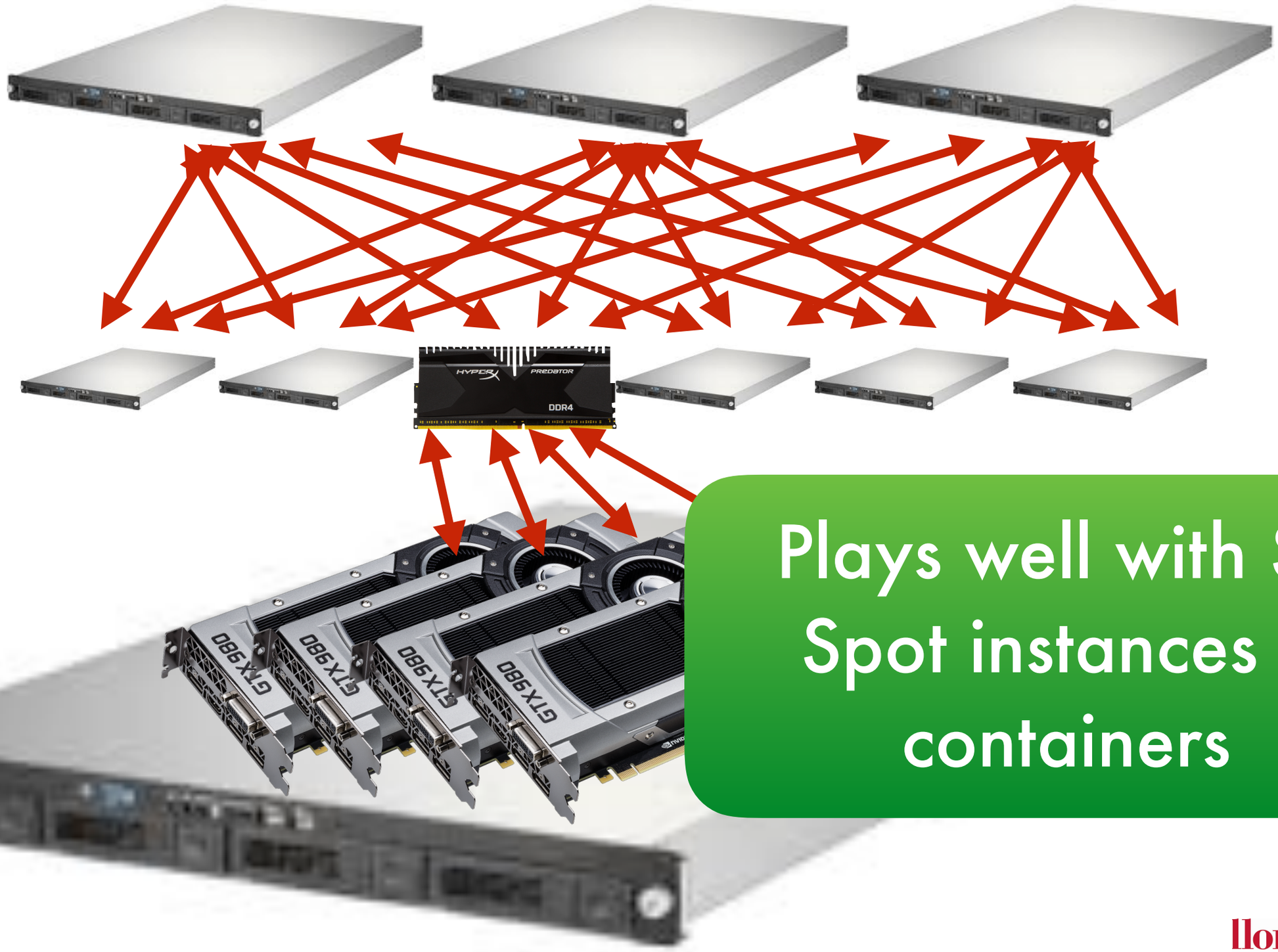
- All operations issued into backend C++ engine
  Binding languages' performance does not matter
- Engine builds the read/write dependency graph
  - lazy evaluation
  - parallel execution
  - sophisticated memory allocation



Distributed Deep Learning



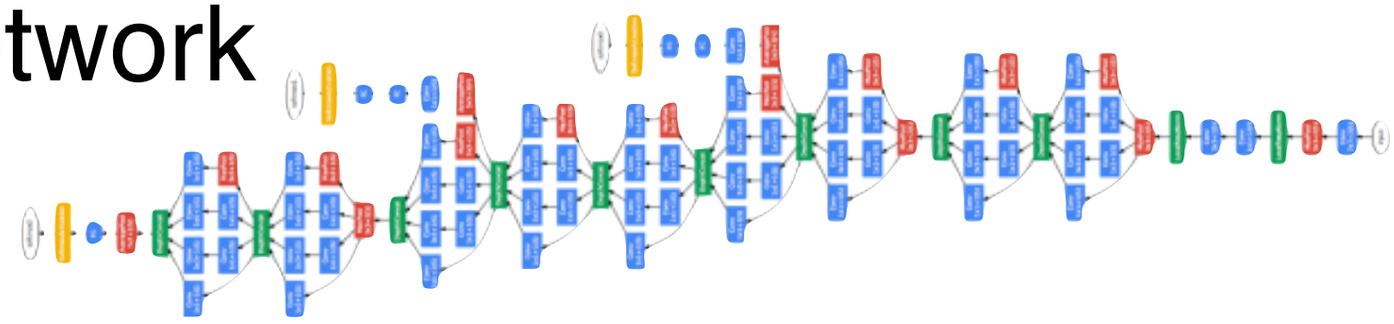
Distributed Deep Learning



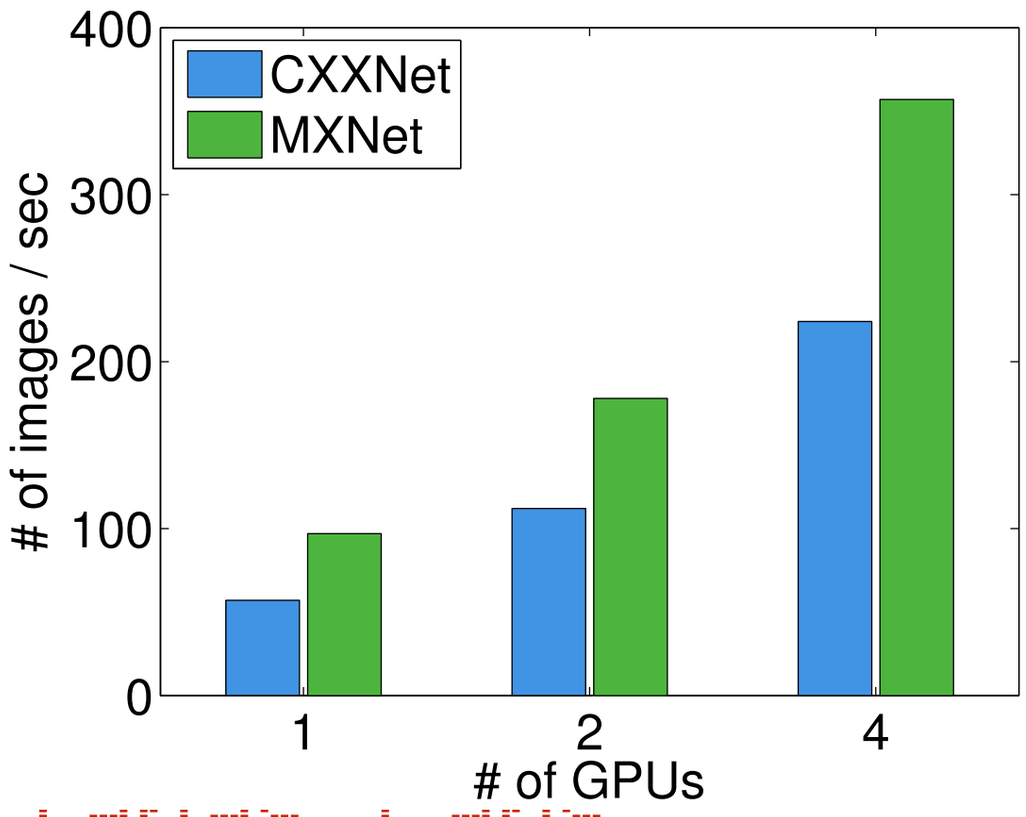
Plays well with S3
Spot instances &
containers

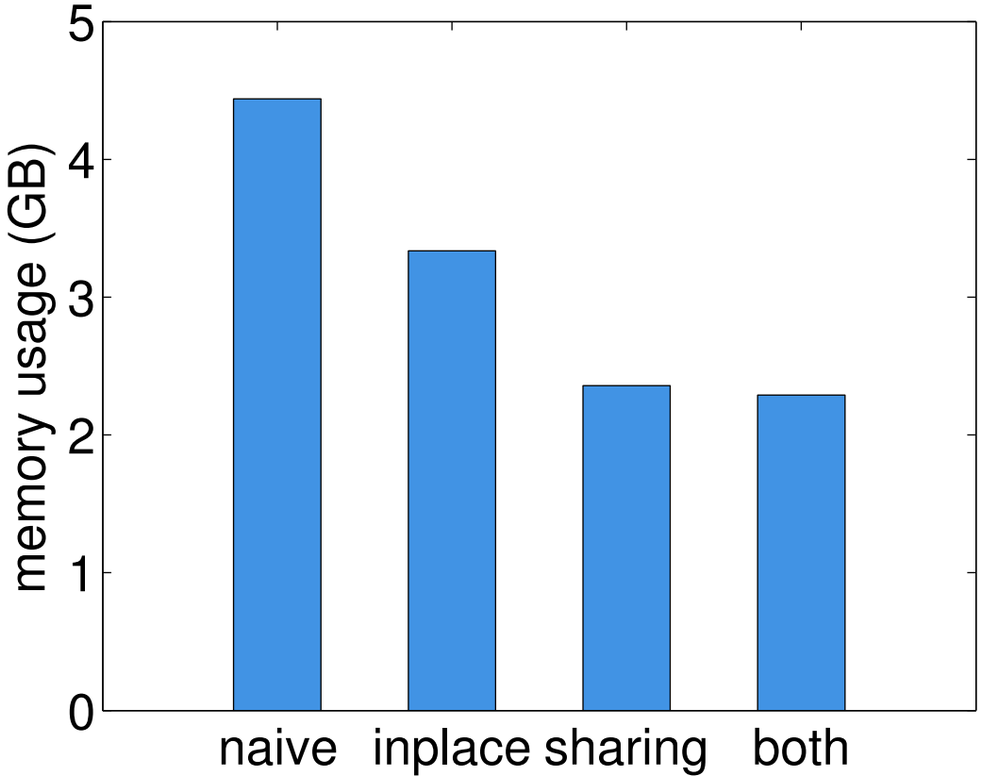
Results

- Imagenet datasets (1m images, 1k classes)
- Google Inception network (88 lines of code)



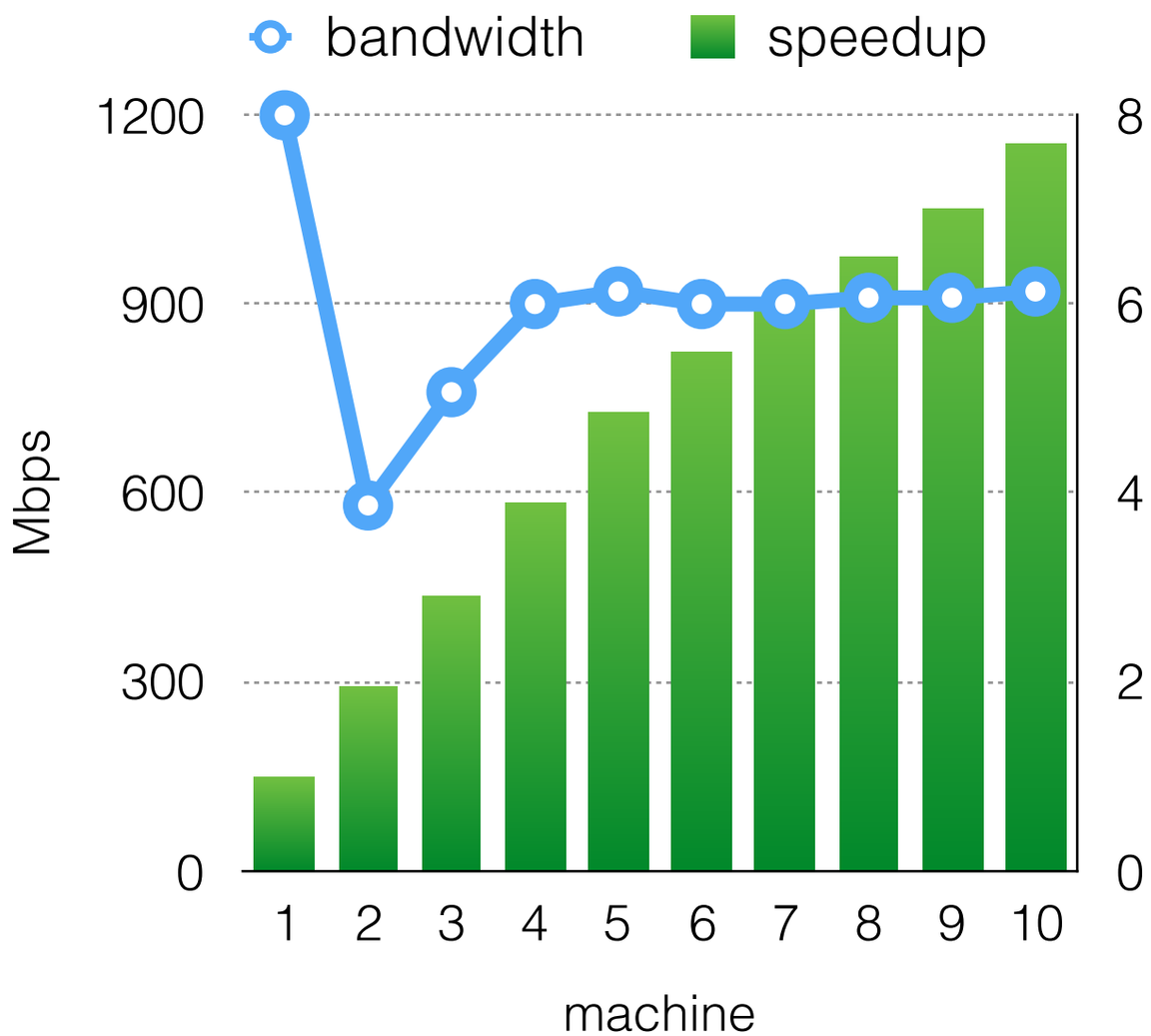
40% faster due to parallelization 50% GPU memory reduction



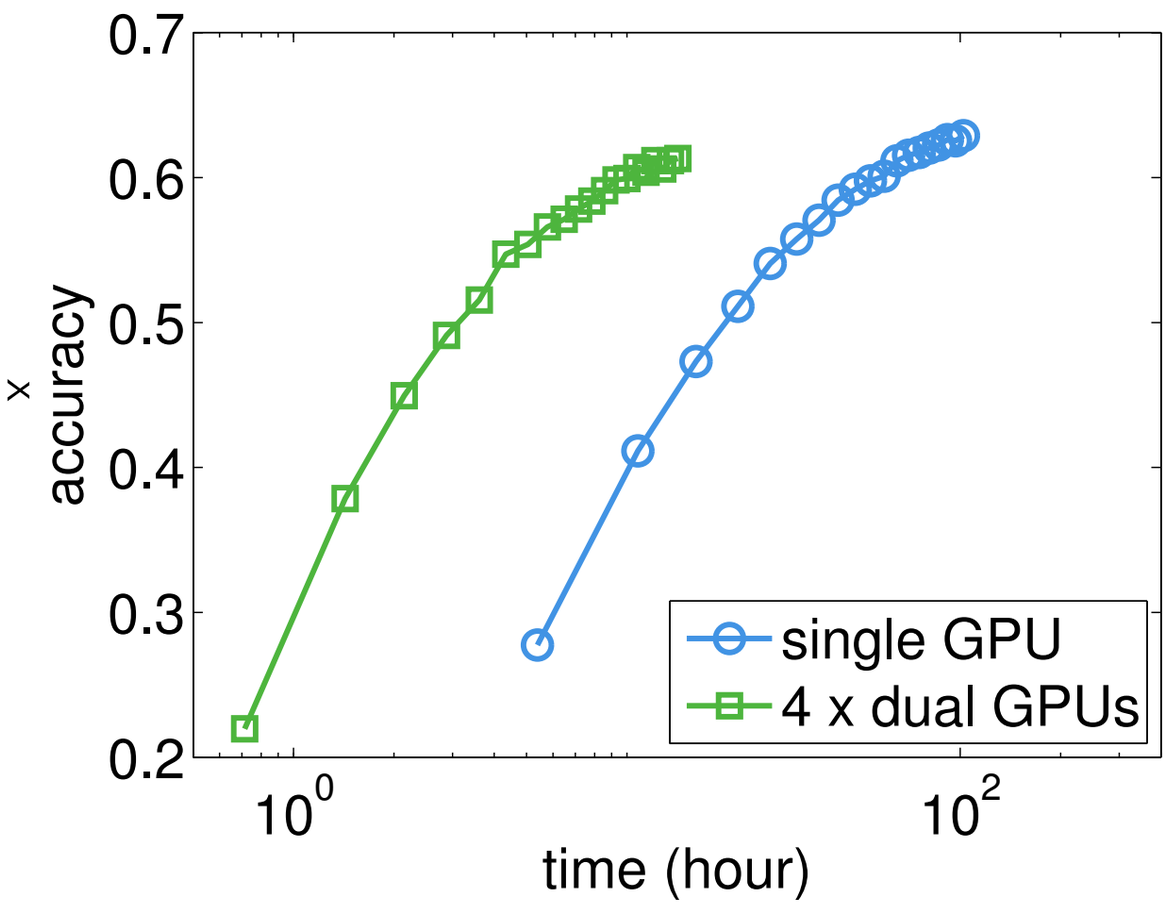


Distributed Results

g2.2xlarge network limit



Convergence on 4 x dual GTX980



Marianas Labs

Carnegie Mellon University

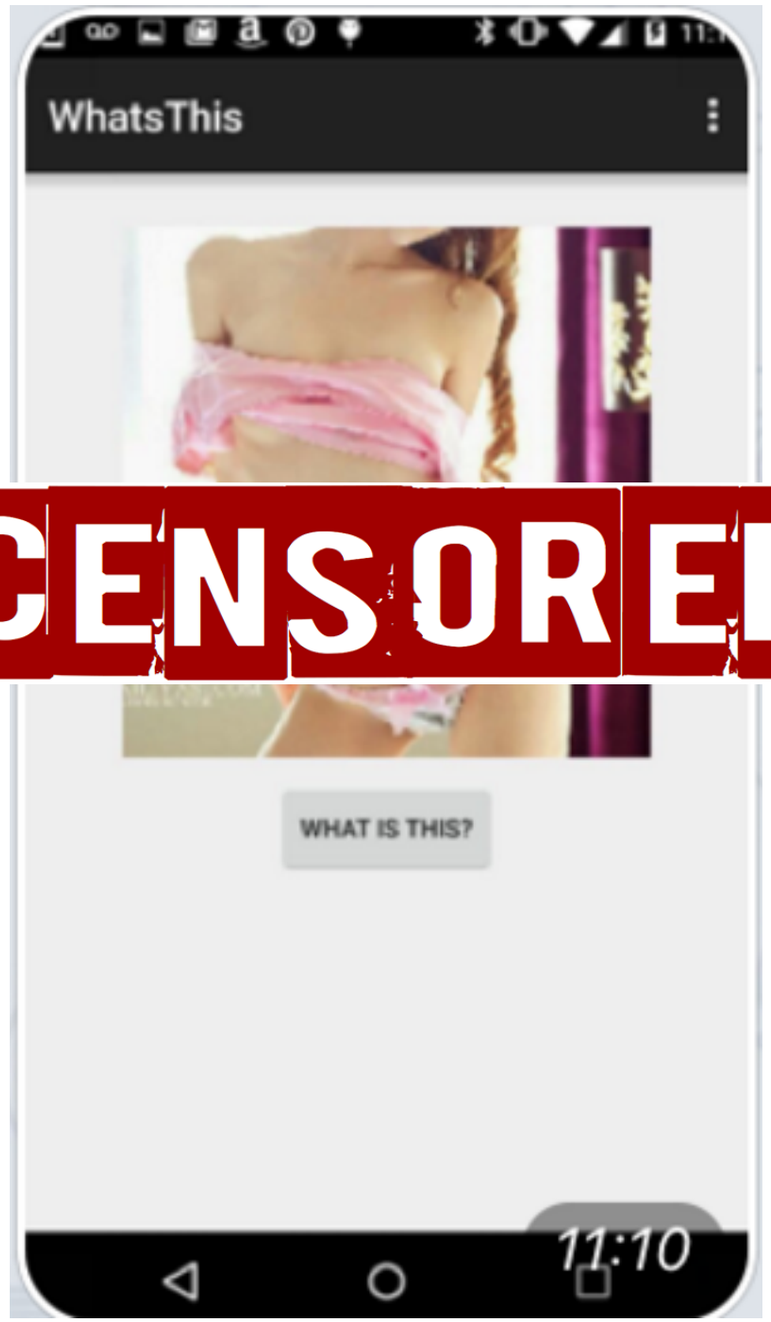
Amazon g2.8xlarge

- 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)

Mobile, too



- 10 hours on 10 GTX980
- Deployed on Android

Side-effects of using MXNET

Alex Smola <alex@smola.org> 1:29 PM (1 hour ago)

to David, Dan, Hyeontaek, FAWN, maas-users

Haha. Zichao's code is getting more efficient over time.

...

Zichao Yang 1:33 PM (57 minutes ago)

to Alex, David, FAWN, maas-users

sorry... I was indeed running experiment using 8 gpus last night and suddenly found the cluster was down :(

...

Mu Li

to Alex, David, FAWN, maas-users

sometime zichao and I are using gpus at the same time. though it looks like mxnet can get more powers due to the multiple streams for each gpu. (didn't see an apparent slow down for mxnet even zichao is also using the gpus)

...

Zichao was using Theano

Tensorflow vs. MXNET

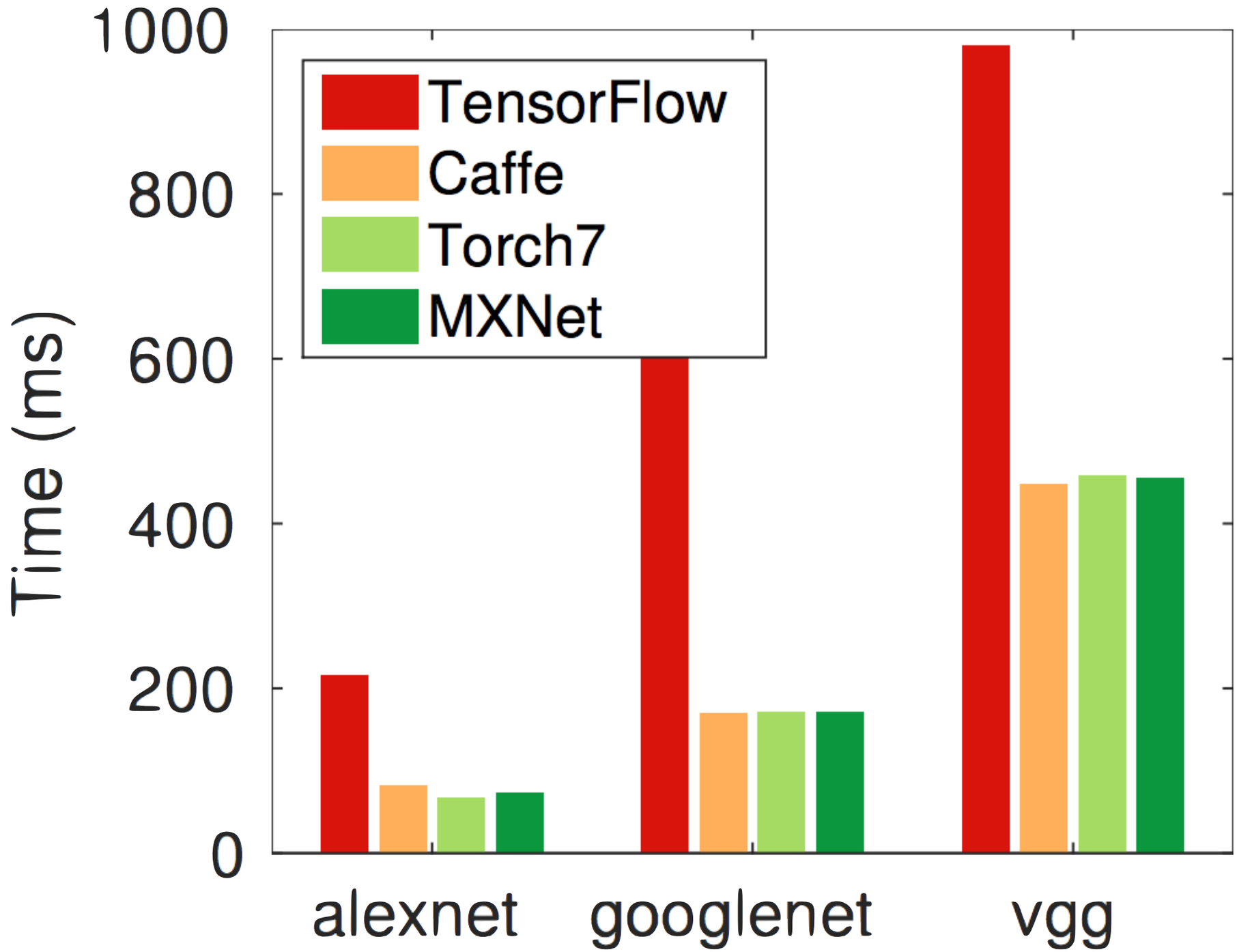
	Languages	MultiGPU	Distributed	Mobile	Runtime Engine
Tensorflow	Python	**Yes**	No	**Yes**	?
MXNET	**Python, R, Julia, Go, Javascript**	**Yes**	**Yes**	**Yes**	**Yes**

dmlc.ml

Some numbers

	Googlenet batch 32 (24)		Alexnet batch 128		VGG batch 32 (24)	
Torch7	172ms	2.1GB	**135ms**	1.6GB	459ms	**3.4gb**
Caffe	**170ms**	2.2GB	165ms	1.34GB	448ms	**3.4gb**
MXNET	172ms (new and improved)	**1.5GB**	147ms	**1.38GB**	456ms	3.6gb
Tensor flow	940ms	-	433ms	-	980ms	-

Some numbers



Summary

Main Contributors

- **Dave Andersen** (ParameterServer)
- **Mu Li** (MXNet, ParameterServer, FM)
- **Manzil Zaheer** (LDA)
- **Tianqi Chen** (MXNet)
- **Ziqi Liu** (Recommender, FM)
- **Jean-Baptiste Tristan** (LDA)
- **Yu-Xiang Wang** (Recommender, FM)

Topics

- **Parameter Server Basics**
  Logistic Regression (Classification)
- **Memory Subsystem**
  Matrix Factorization (Recommender)
- **Large Distributed State**
  - Factorization Machines
  - Latent Variable Models
- **GPUs**
  MXNET and applications