Why your machine learning code is slow

Designing algorithms for modern hardware
(disclaimer - this isn't a deep learning talk)

Alexander Smola, Amazon AWS
alex@smola.org

Most important slide - free AWS credits

- Register at AWS Educate for credit
  http://www.awseducate.com
- Use **PEK_DEEPLEARNING** for extra $25 credits
- **Use the AWS account option**
  Do **not** use the capped AWS Educate Starter Account.
- To apply for a position email CV/GitHub/short intro to
  aws-ai-event-recruiting@amazon.com

Thanks

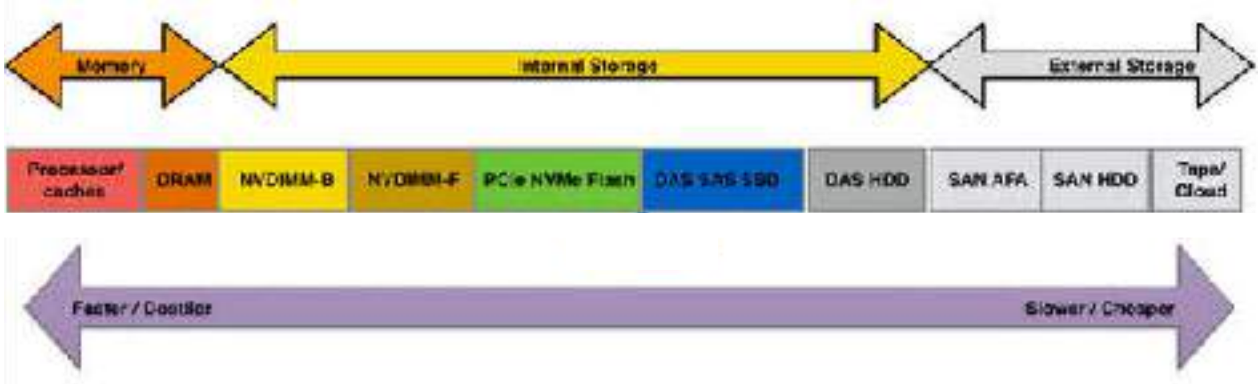
- **Amazon**
  Mu Li, Anima Anandkumar, Vishy Vishwanathan
- **CMU**
  Yu-Xiang Wang (+Amazon), Manzil Zaheer (+Amazon), Dave Andersen, Zichao Yang, Ziqi Liu
- **Google**
  Amr Ahmed, Vanja Josifovski, Steffen Rendle, Sujith Ravi

Outline

- **Memory**
  Cache, RAM, SSD, Harddisk, Network
- **Computation**
  vectorization, multicore
- **MxNet**
  language, parallelization, AWS Templates

Memory

Memory



http://www.theregister.co.uk/2016/04/04/memory_and_storage_boundary_changes/





10MB	10GB	500GB	5TB	100TB
1ns	100ns	10µs	10ms	100s

Numbers you should know

	Nanoseconds (ns)	Microseconds (us)	Milliseconds (msec)	If L1 access is 1 second
L1 cache reference	0.5			1 sec
L2 cache reference	7			14 secs
DRAM access	200			6 mins 40 secs
NVDIMM-N	5,000	5		2 hours 46 mins 40 secs
NVMe PCIe SSD write	30,000	30		16 hours 40 mins
Mangstor NX NVMeF array write	30,000	30		16 hours 40 mins
Zstor NVMe-F SSD array read	30,000	30		16 hours 40 mins
DSSD D5 NVMe-F array	[illegible]	[illegible]	[illegible]	55 hours 33 mins 20 secs
Mangstor NX NVMeF array read	[illegible]	[illegible]	[illegible]	61 hours 6 mins 40 secs
NVMe PCIe SSD read	[illegible]	[illegible]	[illegible]	61 hours 6 mins 40 secs
Zstor NVMe-F SSD array write	[illegible]	[illegible]	[illegible]	61 hours 6 mins 40 secs
Random 4K read from SSD	[illegible]	[illegible]	[illegible]	3 days, 11 hours, 20 mins
Sequential Read 1MB from DRAM	250,000	250		5 days, 18 hours, 53 mins, 20 secs
Round trip i datacenter	500,000	500	0.5	11 days, 13 hours, 46 mins, 40 secs
Sequential read 1MB from SSD	1,000,000	1,000	1	23 days, 3 hours, 33 mins, 20 secs
Disk seek	10,000,000	10,000	10	231 days, 11 hours, 33 mins, 20 secs
Sequential read 1MB from disk	20,000,000	20,000	20	1 year, 97 days, 23 hours, 6 mins, 40 secs
DAS disk access	100,000,000	100,000	100	5 years, 119 days, 19 hours, 33 mins, 20 secs
Send packet CA->Netherland->CA	150,000,000	150,000	150	9 years, 185 days, 5 hours, 20 mins, 0 secs
SAN array access	300,000,000	300,000	300	19 years, 5 days, 10 hours, 40 mins, 0 secs

ignorant code can kill performance

http://www.theregister.co.uk/2016/04/04/memory_and_storage_boundary_changes/

amazon web services

Use Case: 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)

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

Use Case: Recommender Systems

Regularized Objective

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

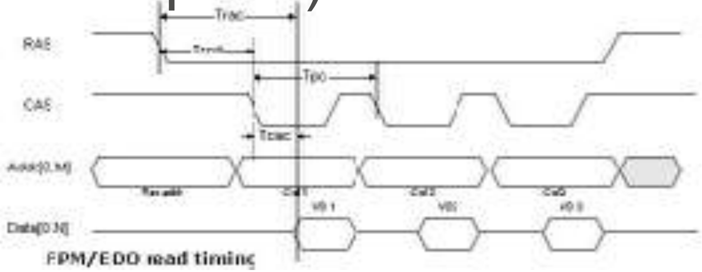
Update operations

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

Very simple SGD algorithm (random pairs)

This should be cheap …

memory subsystem



Not so cheap …

Netflix contest

- 100M samples, 2048 dimensions, 30 steps
- 100M × 2048 × 30 × 4 × 8byte burst reads
- 100M × 30 × 4 random reads

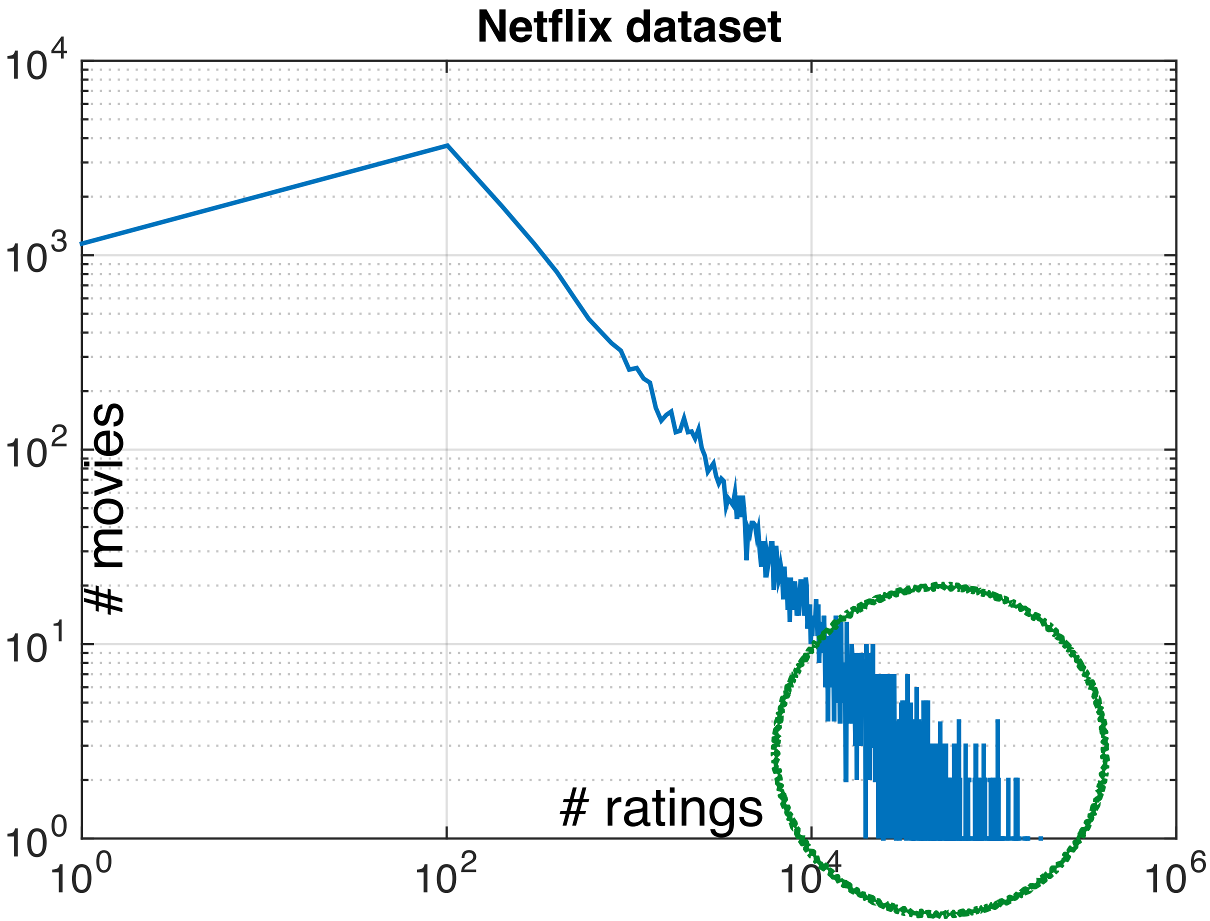
Runtime (1h 15 min)

- 3300s for burst reads (@60GB/s)
- 1200s for random reads (@10ns/read)

Better engineering gets 9.5 min. How?

Liu, Wang, Smola, RecSys 2015

Power law in Collaborative Filtering

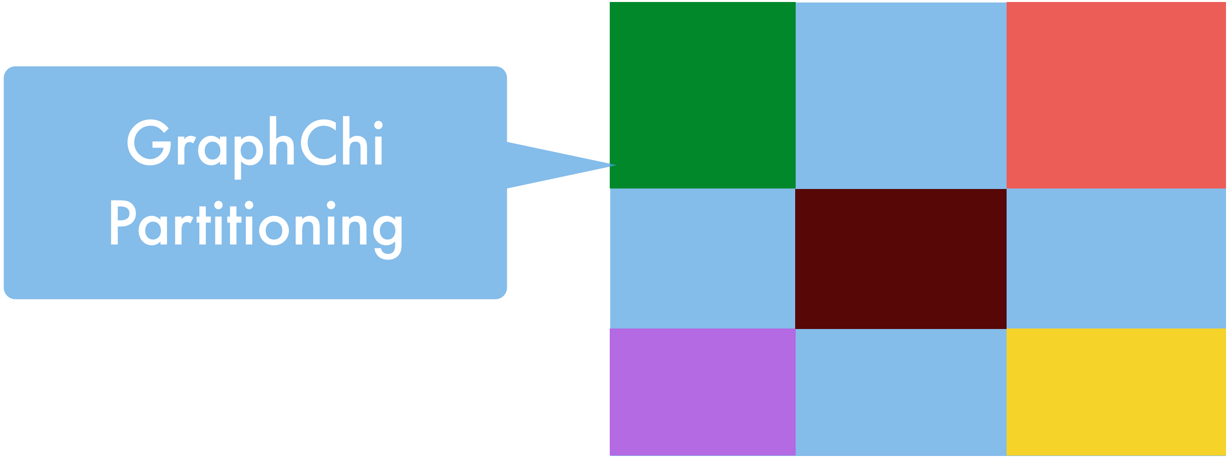


Key Ideas

- **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%

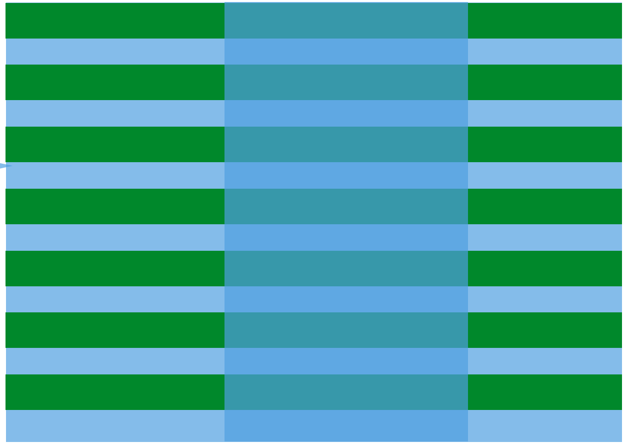
Key Ideas



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%

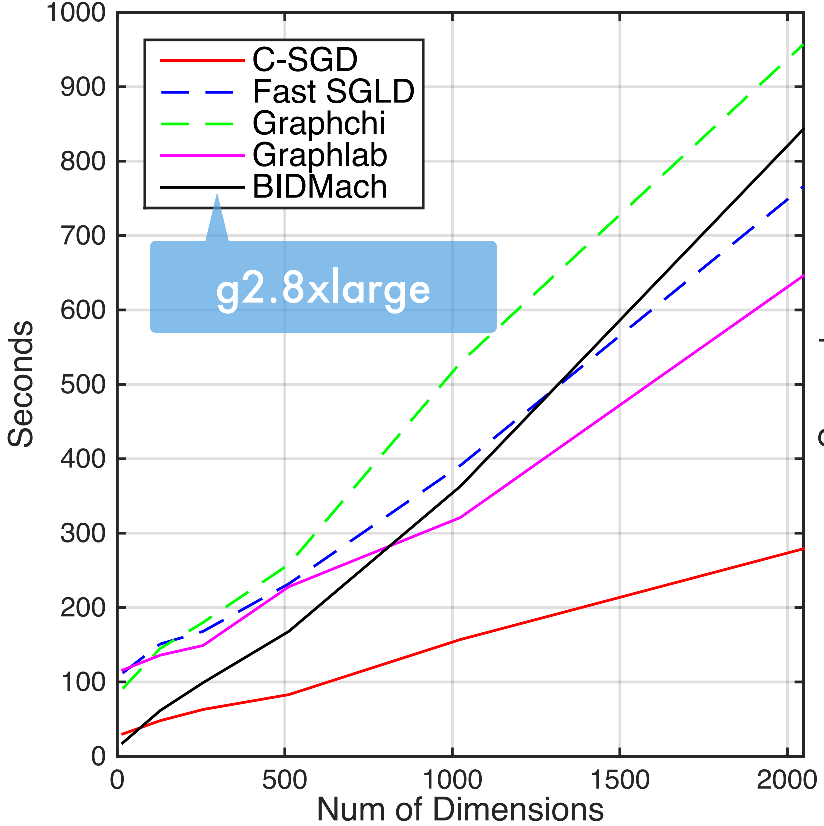
Key Ideas

SC-SGD
partitioning

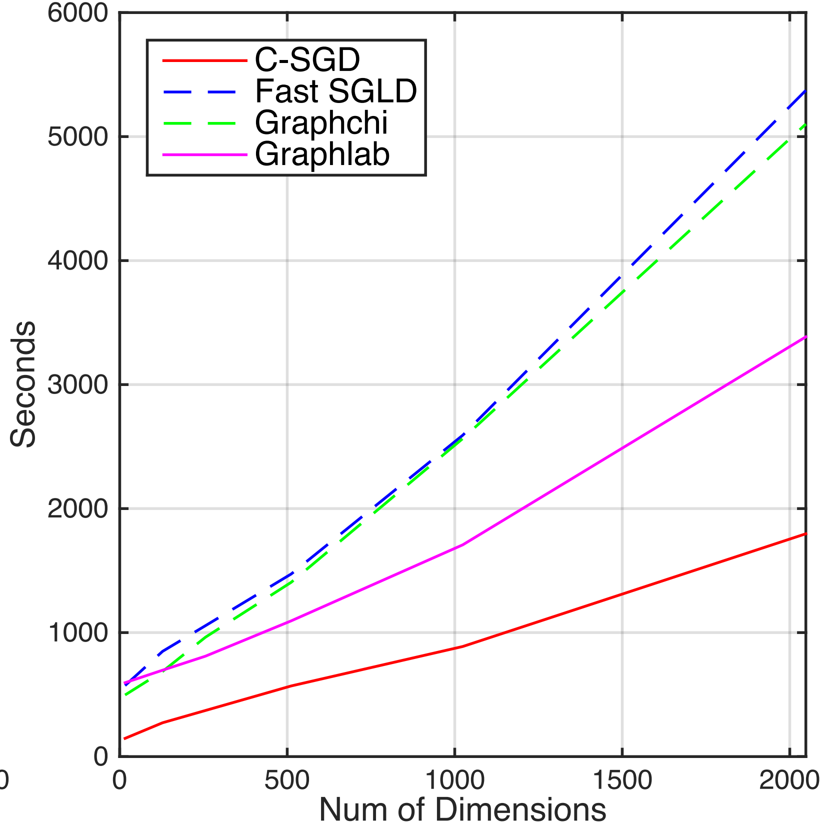


K	SC-SGD		GraphChi	
	L1 Cache	L3 Cache	L1 Cache	L3 Cache
16	2.84%	0.43%	12.77%	2.21%
256	2.85%	0.50%	12.89%	2.34%
2048	3.3%	1.7%	15%	9.8%

Speed (c4.8xlarge)

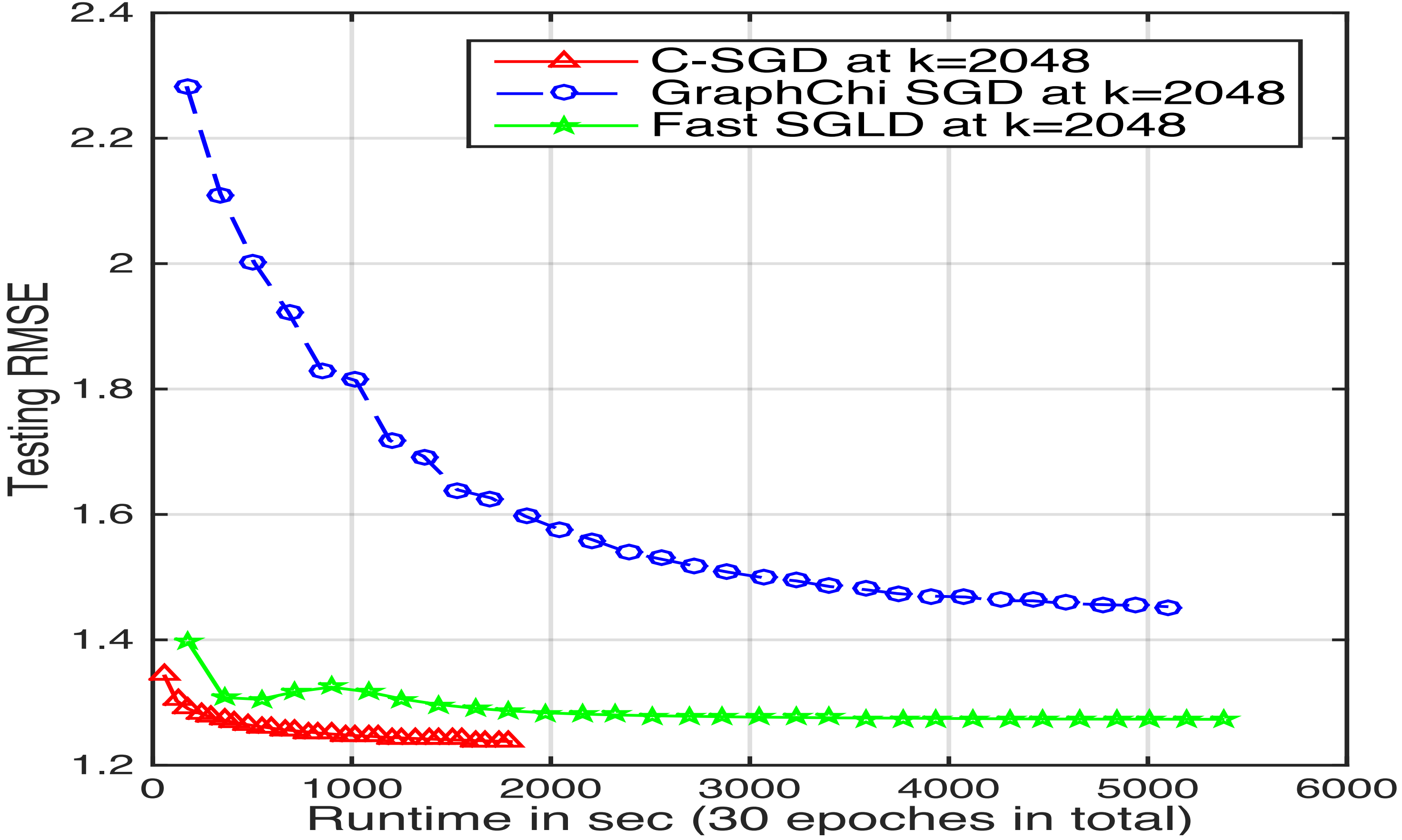


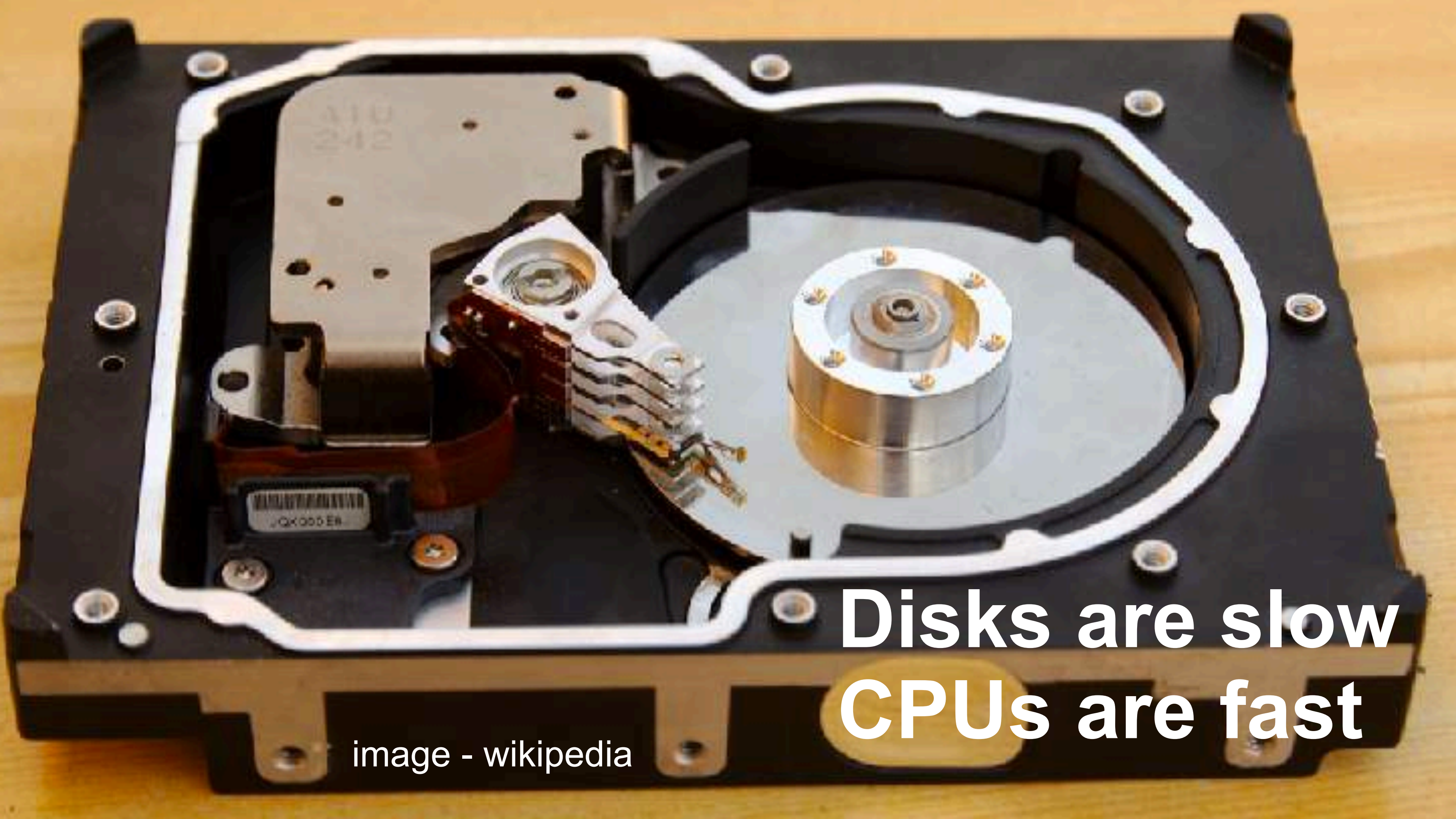
Netflix - 100M, 15 iterations



Yahoo - 250M, 30 iterations

Convergence



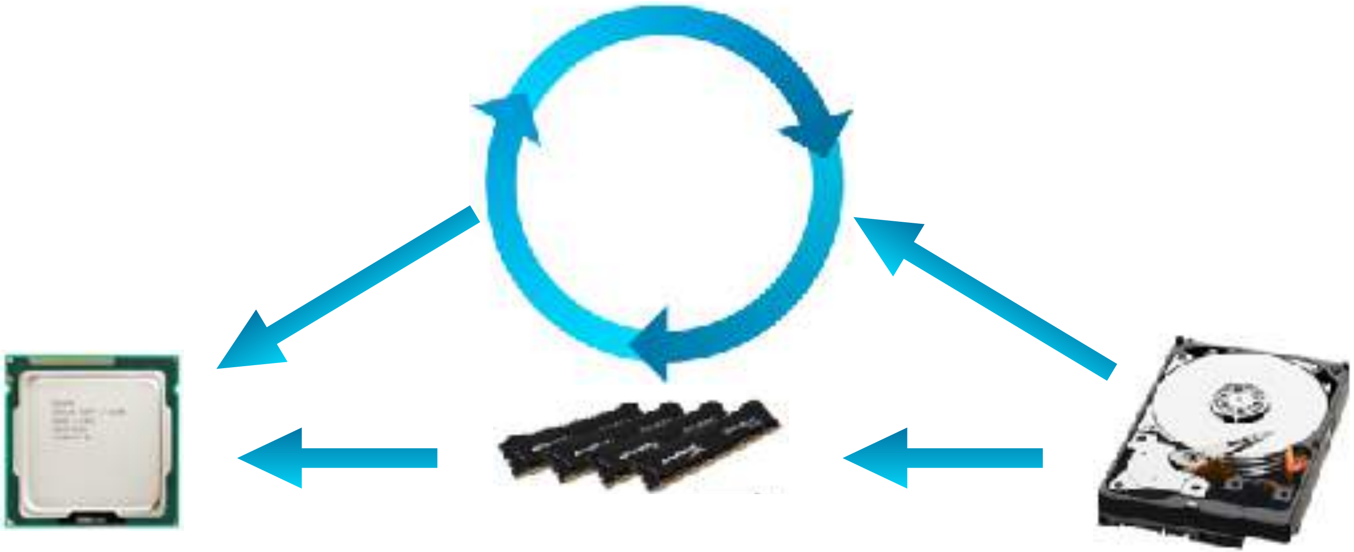

Disks are slow
CPUs are fast
image - wikipedia

Use Case: SVM Optimization

- LibLinear / SMO style optimization
  - Read data from disk
  - Update parameters w in memory (soft margin, logistic)
- CPU is much faster than HDD

10MB	10GB	5TB
1ns	100ns	10ms
1TB/s	50GB/s	100MB/s

Use Case: SVM Optimization



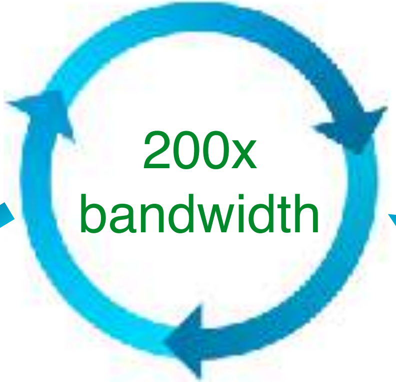
10MB
1ns
1TB/s

10GB
100ns
50GB/s

5TB
10ms
100MB/s

Use Case: SVM Optimization

trainer - read data
repeatedly from RAM
update model



reader - get data
from disk and
write to memory

10MB
1ns
1TB/s

10GB
100ns
50GB/s

5TB
10ms
100MB/s

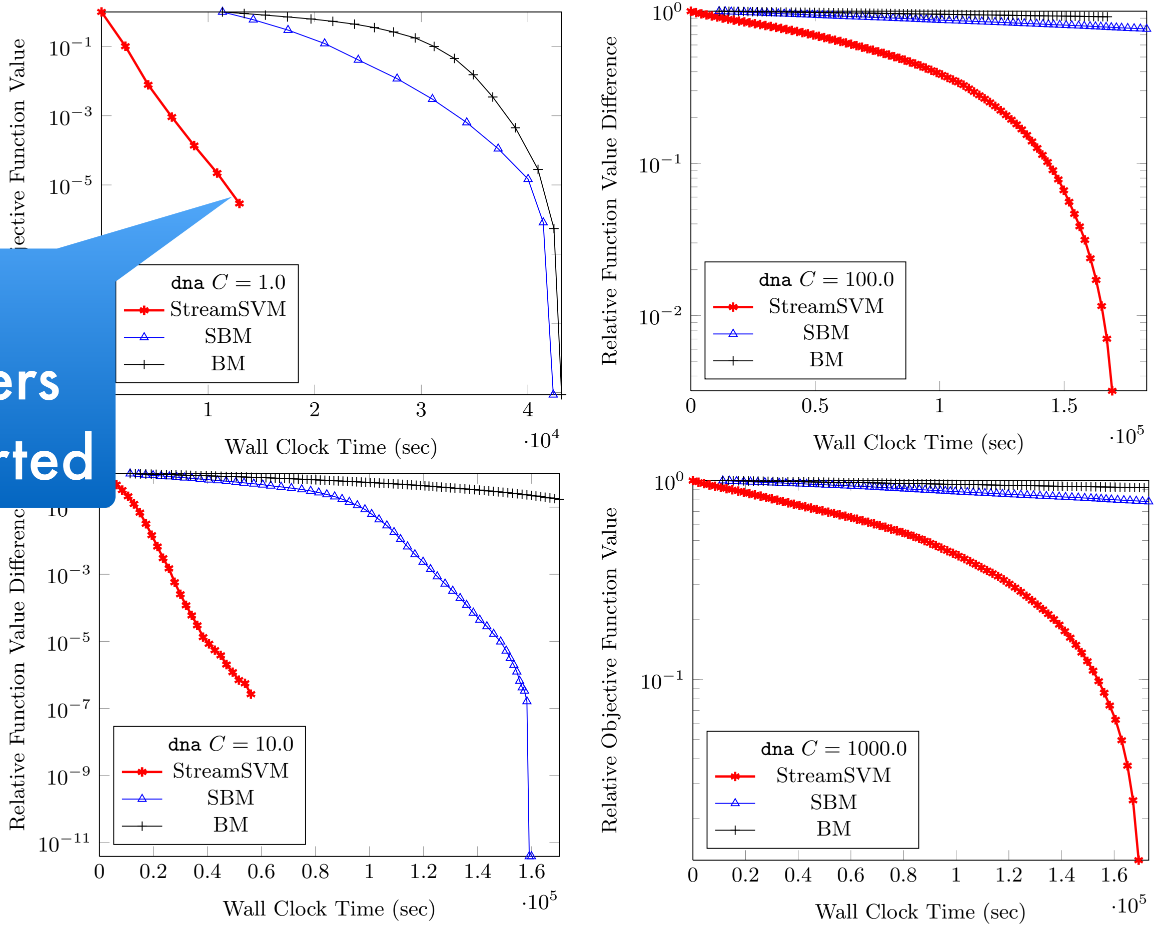
Technical challenge

- If we just reuse observations we might overweight some data relative to the rest

$$\sum_i l(x_i, y_i, f(x_i)) \longrightarrow \sum_j \sum_{i \in S_j} l(x_i, y_i, f(x_i))$$

- Primal descent impossible (without bookkeeping)
- Dual ascent is accurate (leaves objective unchanged)

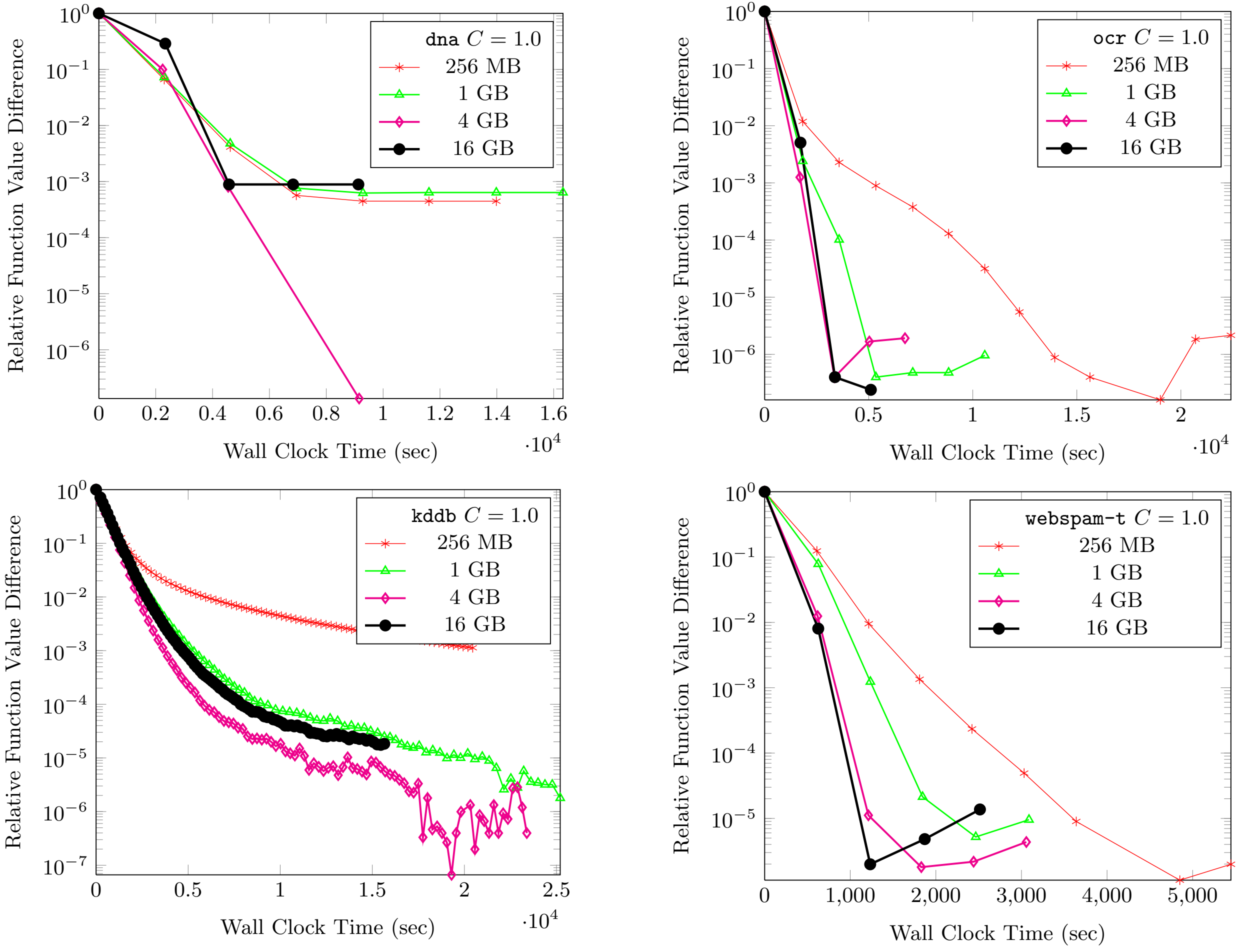
Speed (DNA dataset, different C)

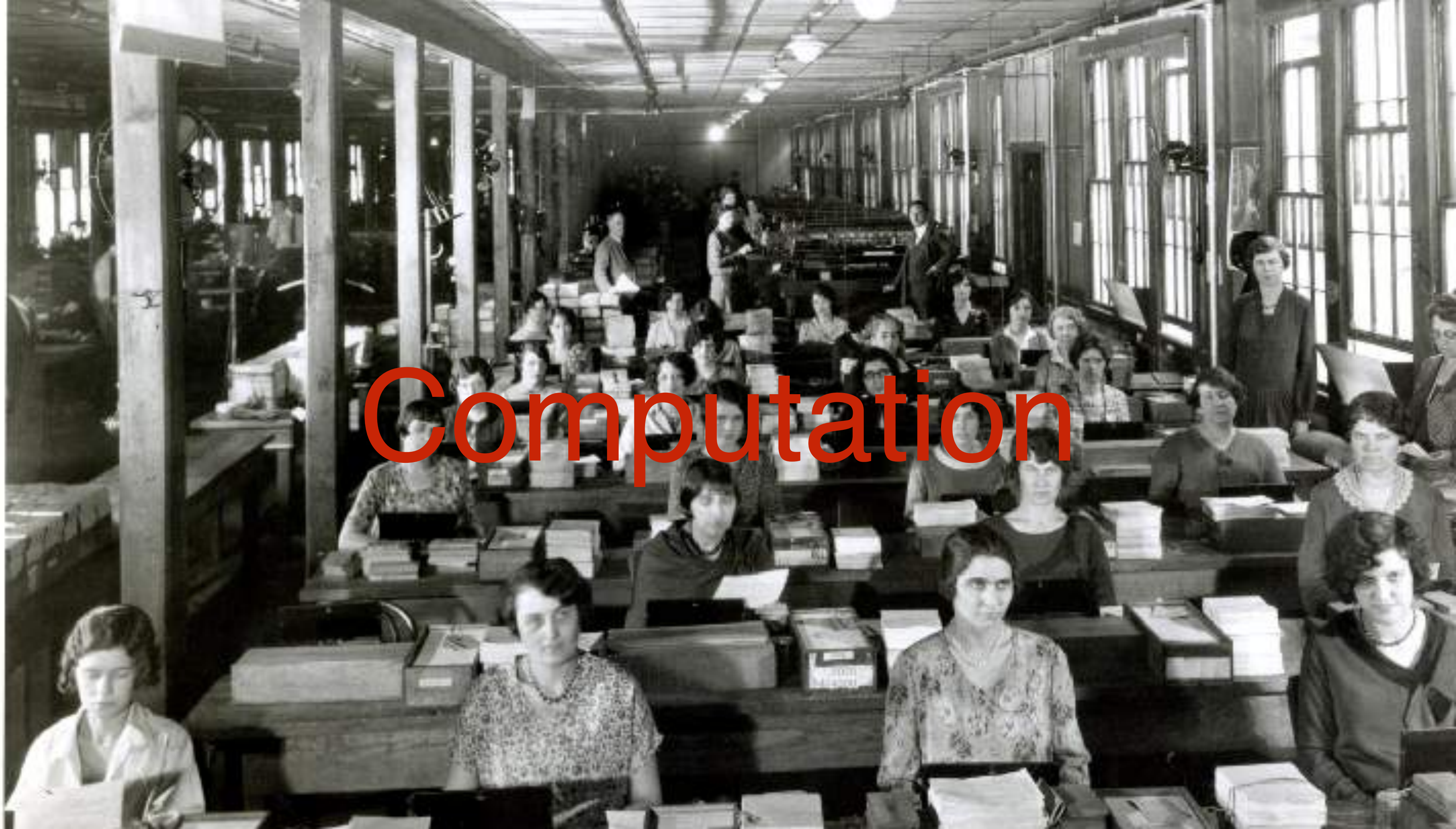


we finish before others even get started

Effect of caching

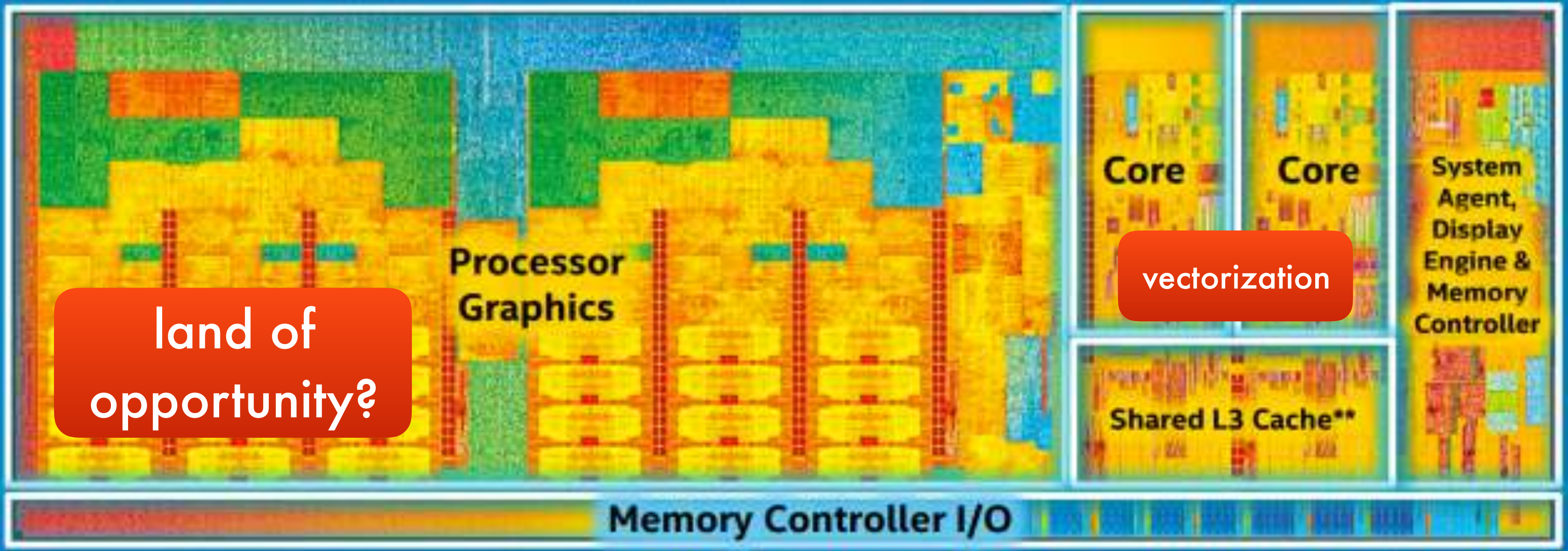
Matsushima, Vishwanathan, Smola, KDD'12




Computation

5th Gen Intel® Core™ Processor Die Map

Intel® HD Graphics 6000 or Intel® Iris™ Graphics 6100



Dual Core Die Shown Above | Transistor Count: 1.9 Billion | Die Size: 133 mm²

4th Gen Core Processor (U series): 1.3B

4th Gen Core Processor (U series): 181mm²

** Cache is shared across both cores and processor graphics

Vectorization

Per Core (up to 18 per chip - Xeon E5)

- Multiple integer and FP units
  (each one needs to be fed with data)
- Naive calculation (i7-5960X)
  8 × 3 GHz = 24 GFlops but benchmark has 183 GFlops

AVX Instructions (256 bit or 512 bit wide on Xeon)

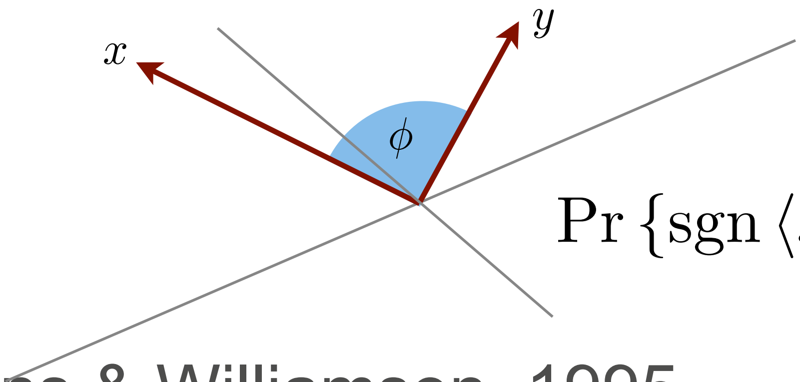
- Works on arrays of numbers (8 floats = 256 bit)
- **In one clock cycle**
- Improved calculation (i7-5960X)
  8 × 3 × 8 GHz = 192 GFlops (much better)

BLAS/LAPACK/EISPACK/whateverPACK libraries
highly optimized - don't write your own code **needlessly**

amazon web services

Use Case: SimHash

- Basic Idea

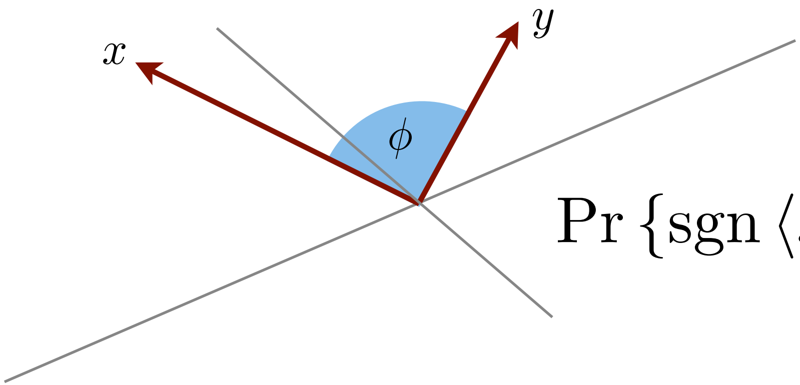


$$\Pr\left\{\operatorname{sgn}\langle x, w\rangle \neq \operatorname{sgn}\langle y, w\rangle\right\} = \frac{\phi}{\pi}$$

- Goemans & Williamson, 1995
  Use this for an SDP relaxation of graph cut
- Charikar, 2003
  Use this for hashing angles between vectors

Use Case: SimHash

- Basic Idea



$$\Pr\left\{\operatorname{sgn}\langle x, w\rangle \neq \operatorname{sgn}\langle y, w\rangle\right\} = \frac{\phi}{\pi}$$

- Hash map is very memory efficient (n bits)

$$x \to h(x) = \left(\operatorname{sgn}\langle x, w_1\rangle, \ldots, \operatorname{sgn}\langle x, w_n\rangle\right)$$

  Compute with matrix-vector product

- Inner product estimation

$$\langle x, y\rangle \approx \|x\| \, \|y\| \cos n^{-1} \|h(x) - h(y)\|_1$$

Cosine similarity

- Similarity measure for many problems (search, retrieval, recommendation)
- Very expensive if we need to compute it many times for high dimensional data
- Good enough approximation for first step

$$\langle x, y \rangle \approx \|x\| \, \|y\| \cos n^{-1} \|h(x) - h(y)\|_1$$

compute once

negligible

vectorize with AVX

Clustering

- Text clustering via exponential family distributions
- Discrete distribution (approximate with SimHash)
  Once per cluster is expensive for 10k clusters

expensive

normalization

$$p(x) = \exp\left(\langle w, x\rangle - g(w)\right)$$

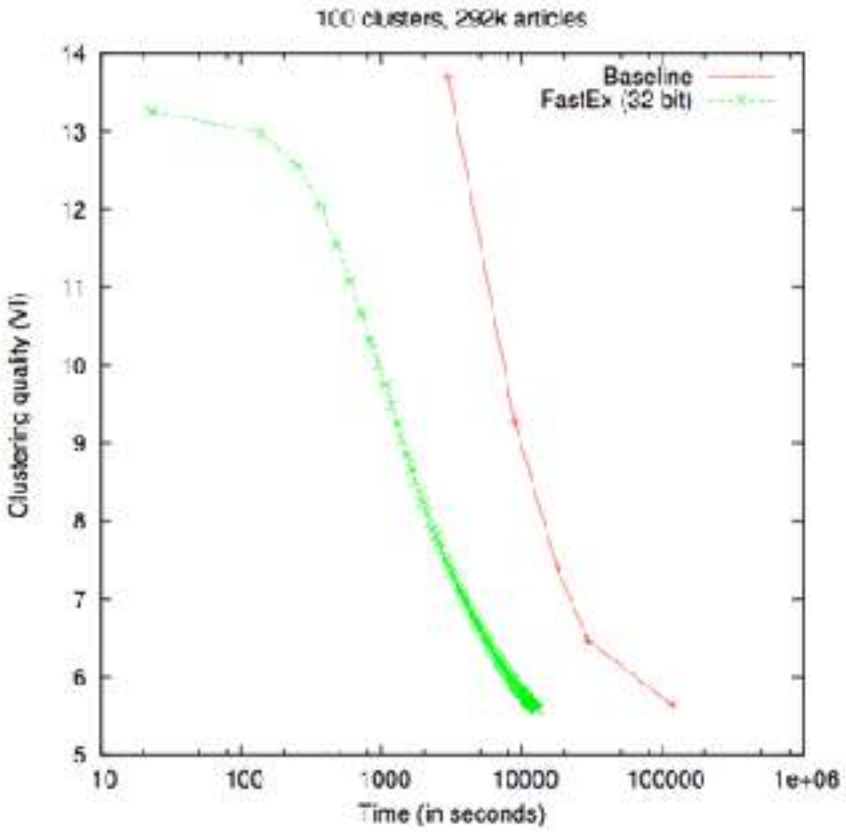
exponential family

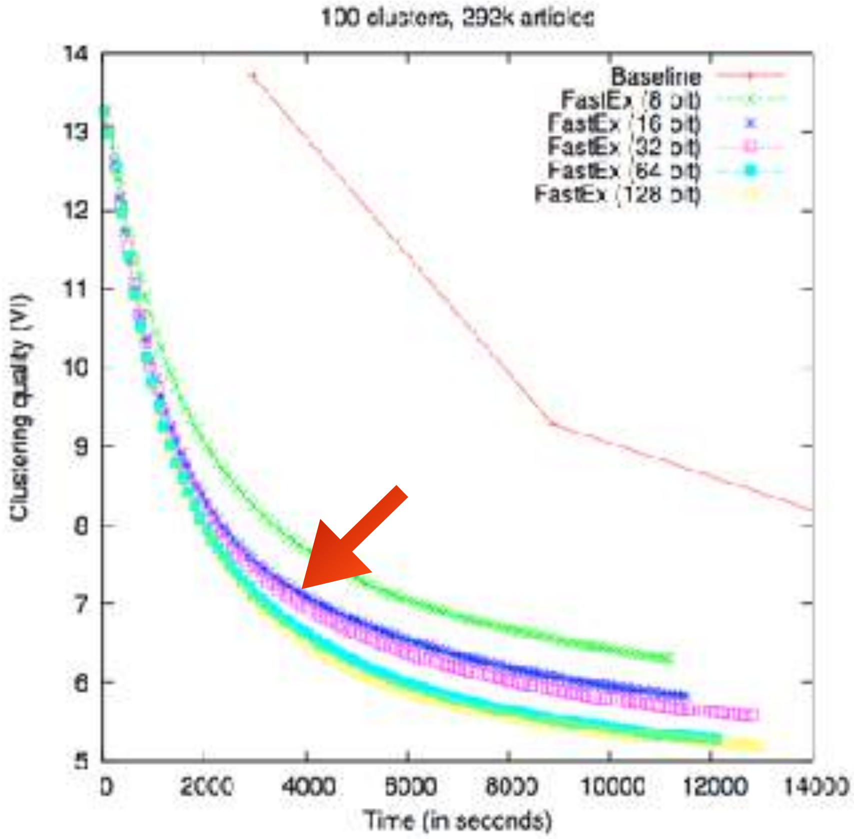
natural parameter

sufficient statistic

amazon web services

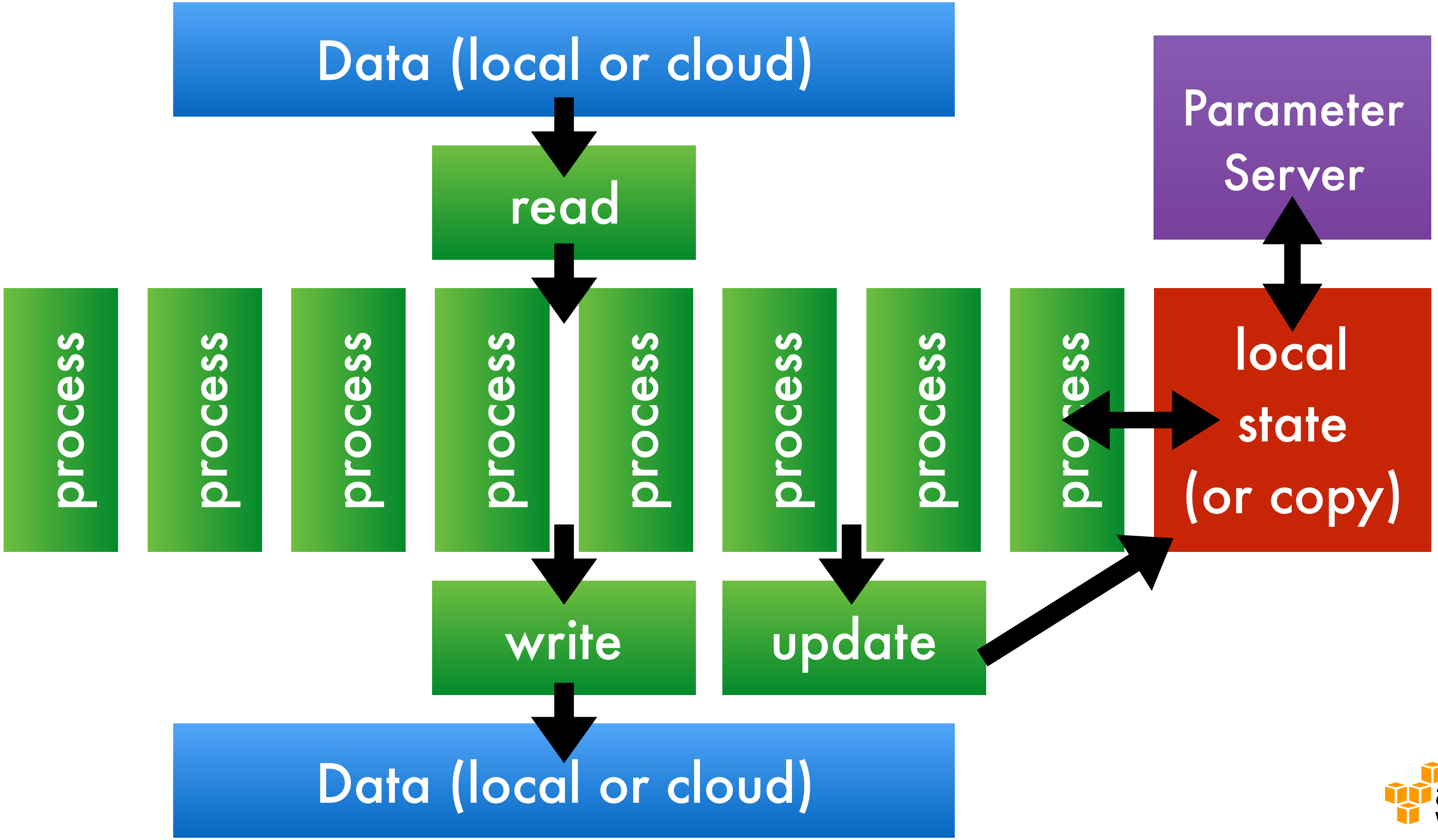
Results



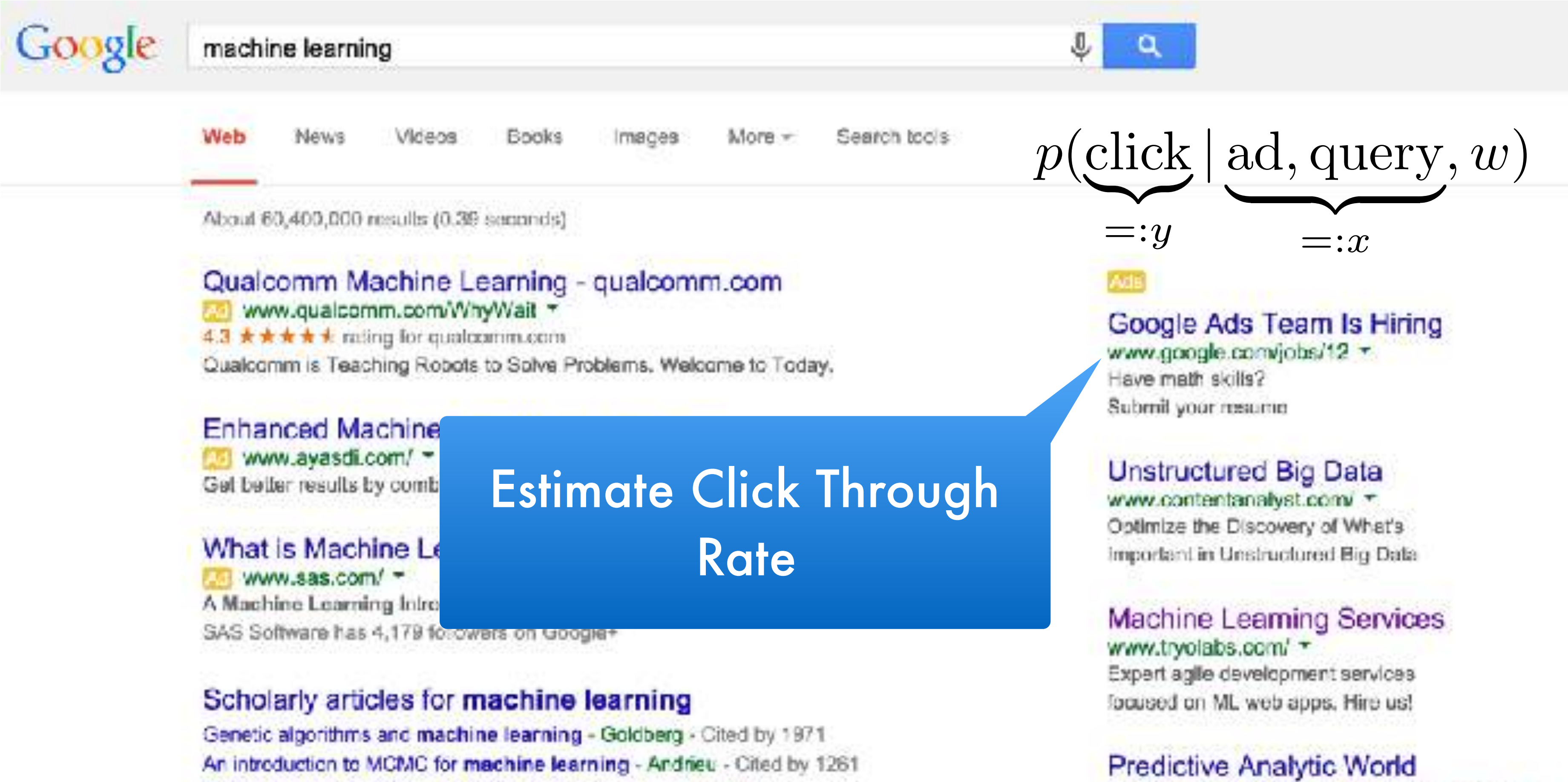


Ahmed, Narayanamurthy, Ravi, Smola, NIPS 2012

No Locks for Multicore



A boring problem (worth $100B)



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

Estimate Click Through Rate

Logistic Regression

- 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

- Small network
  100M users, 10 days = 1B examples, 1 big server

Stochastic gradient descent

- **Compute gradient on data**

$$g_i = \partial_w l(x_i, y_i, w) \text{ e.g. } \partial_w \log\left(1 + \exp\left(-y_i \langle x_i, w \rangle\right)\right)$$

- Update parameter with gradient (for l_2 penalty)

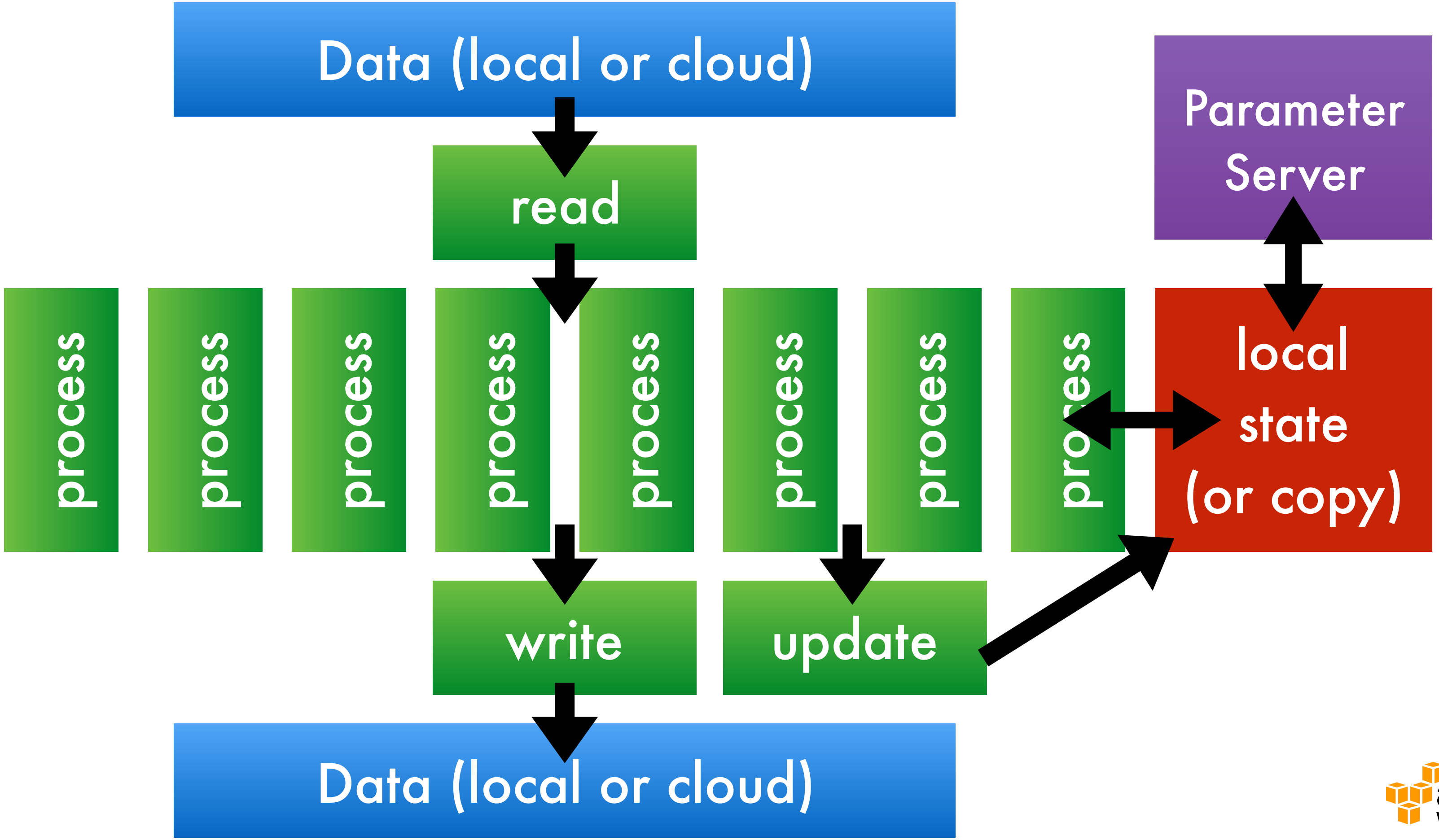
$$w \leftarrow (1 - \lambda\eta)w - \eta g_i$$

- Update parameter with prox operator (l_1 penalty)

$$w \leftarrow \operatorname*{argmin}_w \|w\|_1 + \frac{\gamma}{2} \|w - (w_i - \eta g_i)\|^2$$

- This is **sequential**. Most cores will be idle
- But most updates are sparse. **Discard locks!**
  (Hogwild - Recht, Re, Wright, 2014)

Dataflow



amazon web services

2. MXNet

- Imperative and Declarative Programming
- Language Support
- Backend and Automatic Parallelization

Caffe(2)
(Py)Torch
Theano
Tensorflow
CNTK
Keras
Paddle
Chainer
SINGA
DL4J

image credit - Banksy/wikipedia

Why yet another deep networks tool?

- **Frugality & resource efficiency**
  Engineered for cheap GPUs with smaller memory, slow networks
- **Speed**
  - Linear scaling with #machines and #GPUs
  - High efficiency on single machine, too (C++ backend)
- **Simplicity**
  Mix declarative and imperative code

frontend

backend

single implementation of backend system and common operators

performance guarantee regardless which frontend language is used

Ease of coding & efficiency (imperative & declarative)

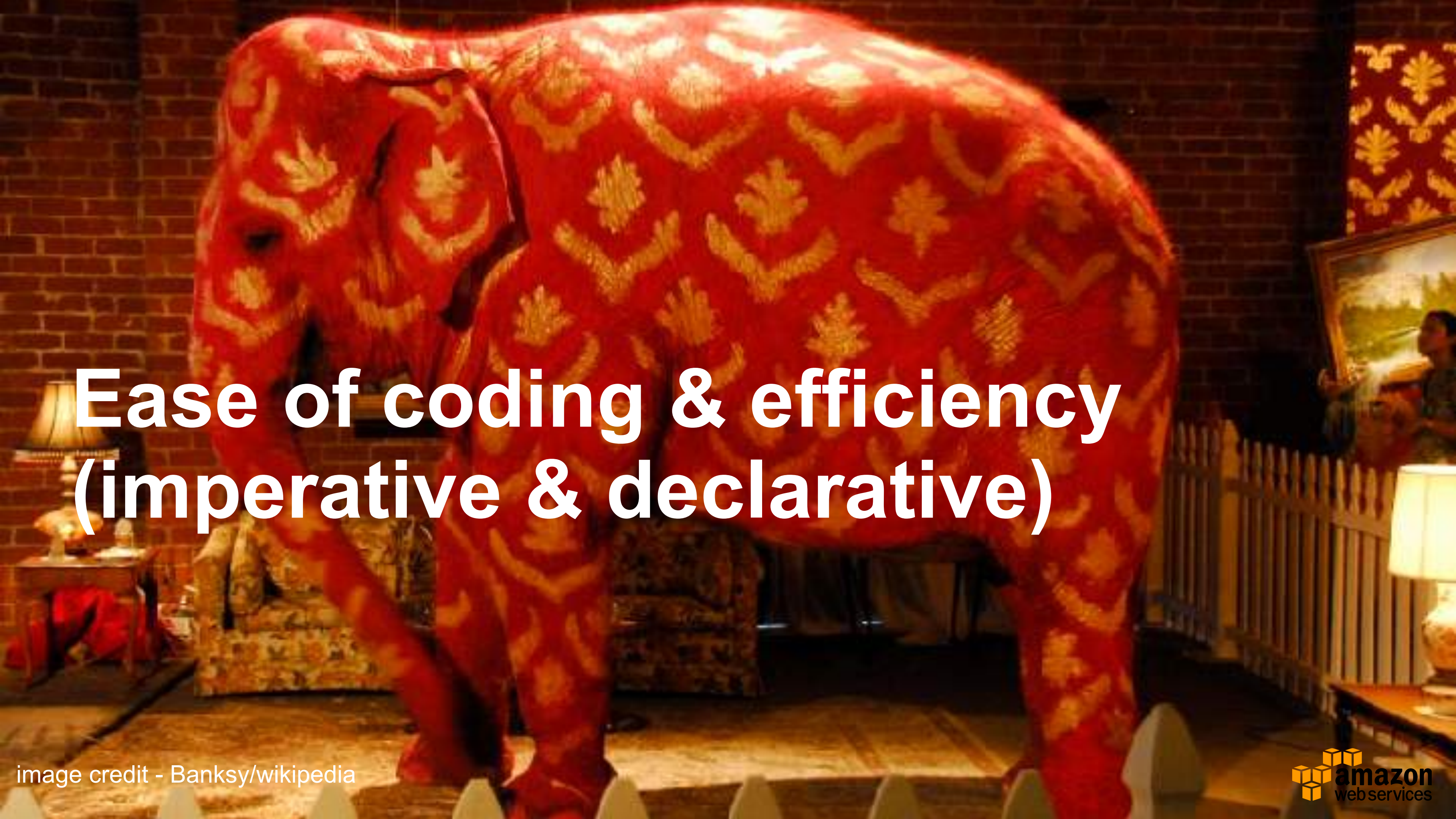


image credit - Banksy/wikipedia

amazon web services

Imperative Programs

```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
print c
d = c + 1
```

Easy to tweak with python codes

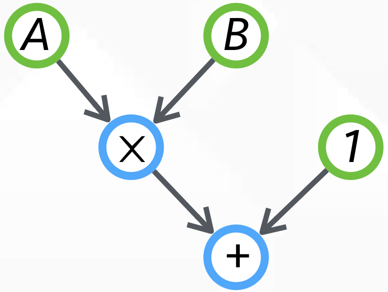
Pro

- Straightforward and flexible.
- Take advantage of language native features (loop, condition, debugger)

Con

- Hard to optimize

Declarative Programs




```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + 1
f = compile(D)
d = f(A=np.ones(10),
      B=np.ones(10)*2)
```

C can share memory with *D*, because *C* is deleted later

Pro

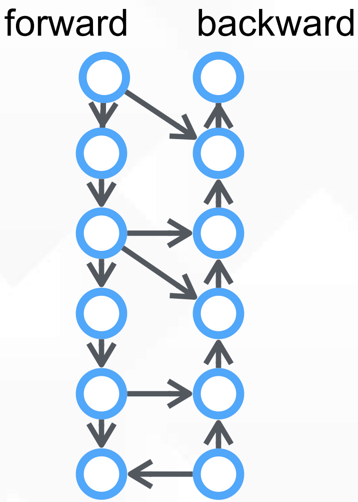
- More chances for optimization
- Cross different languages

Con

- Less flexible

Imperative vs. Declarative for Deep Learning

Computational Graph of the Deep Architecture



Needs heavy optimization, fits **declarative** programs

Updates and Interactions with the graph

- Iteration loops
- Parameter update

$$w \leftarrow w - \eta \partial_w f(w)$$

- Beam search
- Feature extraction …

Needs mutation and more language native features, good for **imperative** programs

MXNet: Mix the Flavors Together

Imperative NDArray API

```
import mxnet as mx
a = mx.nd.zeros((100, 50))
b = mx.nd.ones((100, 50))
c = a + b
c += 1
print(c)
```

Declarative Symbolic Executor

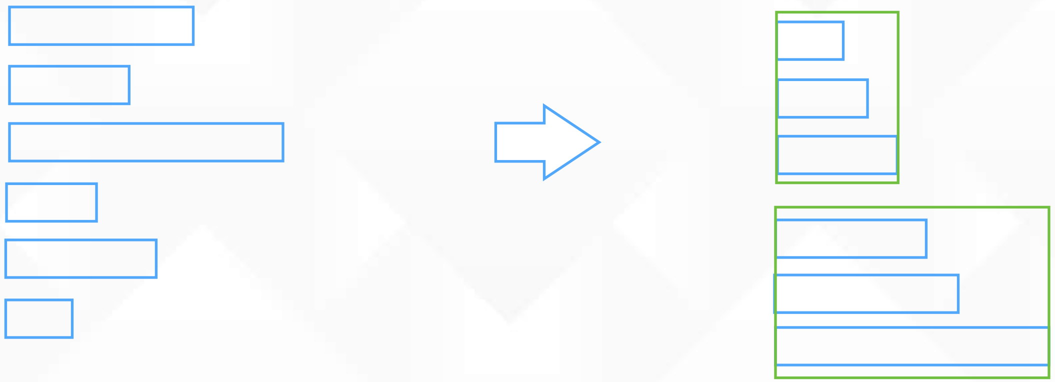
```
import mxnet as mx
net = mx.symbol.Variable('data')
net = mx.symbol.FullyConnected(data=net, num_hi
net = mx.symbol.SoftmaxOutput(data=net)
texec = mx.module.Module(net)
texec.forward(data=c)
texec.backward()
```

Imperative NDArray can be set as input to the graph

Mixed API for Quick Extensions

Variable length sentences

Bucketing



- Runtime switching between different graphs depending on input
- Useful for sequence modeling and image size reshaping
- Use of imperative code in Python, 10 lines of additional Python code

3D Image Construction

https://github.com/piiswrong/deep3d

Deep3D

100 lines of Python code

amazon web services

Multiple Languages
Multiple Toolkits

image credit - Banksy/wikipedia

What We Heard from Users

Programming Languages:

- Python is nice, but I like R/Julia/Matlab more
- I want Scala to work with the Spark pipeline
- I need C++ interface to run on embedded systems
- I prefer Javascript to run on user browsers

Frameworks:

- I used Torch for 7 years
- All my codes are in Caffe
- I like Keras
- I started deep learning with Tensorflow
- I only used Numpy before, how should I start?

Multiple Programming Languages

frontend

backend

single implementation of backend system and common operators

performance guarantee regardless of which frontend language is used

Bringing Caffe to MXNet

Caffe is widely used in computer vision

Call Caffe Operators in MXNet

```
import mxnet as mx
data = mx.symbol.Variable('data')
fc1  = mx.symbol.CaffeOp(data_0=data, num_weight=2, prototxt=
       "layer{type:\"InnerProduct\" inner_product_param{num_output: 128} }")
act1 = mx.symbol.CaffeOp(data_0=fc1, prototxt="layer{type:\"TanH\"}")
fc2  = mx.symbol.CaffeOp(data_0=act1, num_weight=2, prototxt=
       "layer{type:\"InnerProduct\" inner_product_param{num_output: 10}}")
mlp  = mx.symbol.SoftmaxOutput(data=fc3)
```

Bringing Torch to MXNet

Torch is a popular Lua framework for both scientific computing and deep learning

Tensor Computation

```
import mxnet as mx
x = mx.th.randn(2, 2, ctx=mx.gpu(0))
y = mx.th.abs(x)
print y.asnumpy()
```

Modules (Layers)

```
import mxnet as mx
data = mx.symbol.Variable('data')
fc   = mx.symbol.TorchModule(data_0=data, lua_string='nn.Linear(784, 128)',…
mlp  = mx.symbol.TorchModule(data_0=fc, lua_string='nn.LogSoftMax()',…
```

MinPy: NumPy in MxNet

Printing & Debugging

```python
import tensorflow as tf
x = tf.zeros((2, 3))
with tf.control_dependencies([x]):
    tf.Print(x, [x])
sess = ... # Create session.
sess.run([x], ...)
```

Tensorflow Program

```python
import minpy.numpy as np
x = np.zeros((2, 3))
print x
```

MinPy Program

Data-dependent execution (with AutoGrad)

```python
import tensorflow as tf
x = ... # create x array
y = ... # create y array
z = tf.cond(x < y,
            lambda: tf.add(x, y),
            lambda: tf.square(y))
```

Tensorflow Program

```python
import minpy.numpy as np
x = ... # create x array
y = ... # create y array
if x < y:
    z = x + y
else:
    z = y ** 2
```

MinPy Program

… Keras coming very soon …

Efficiency & Parallelization

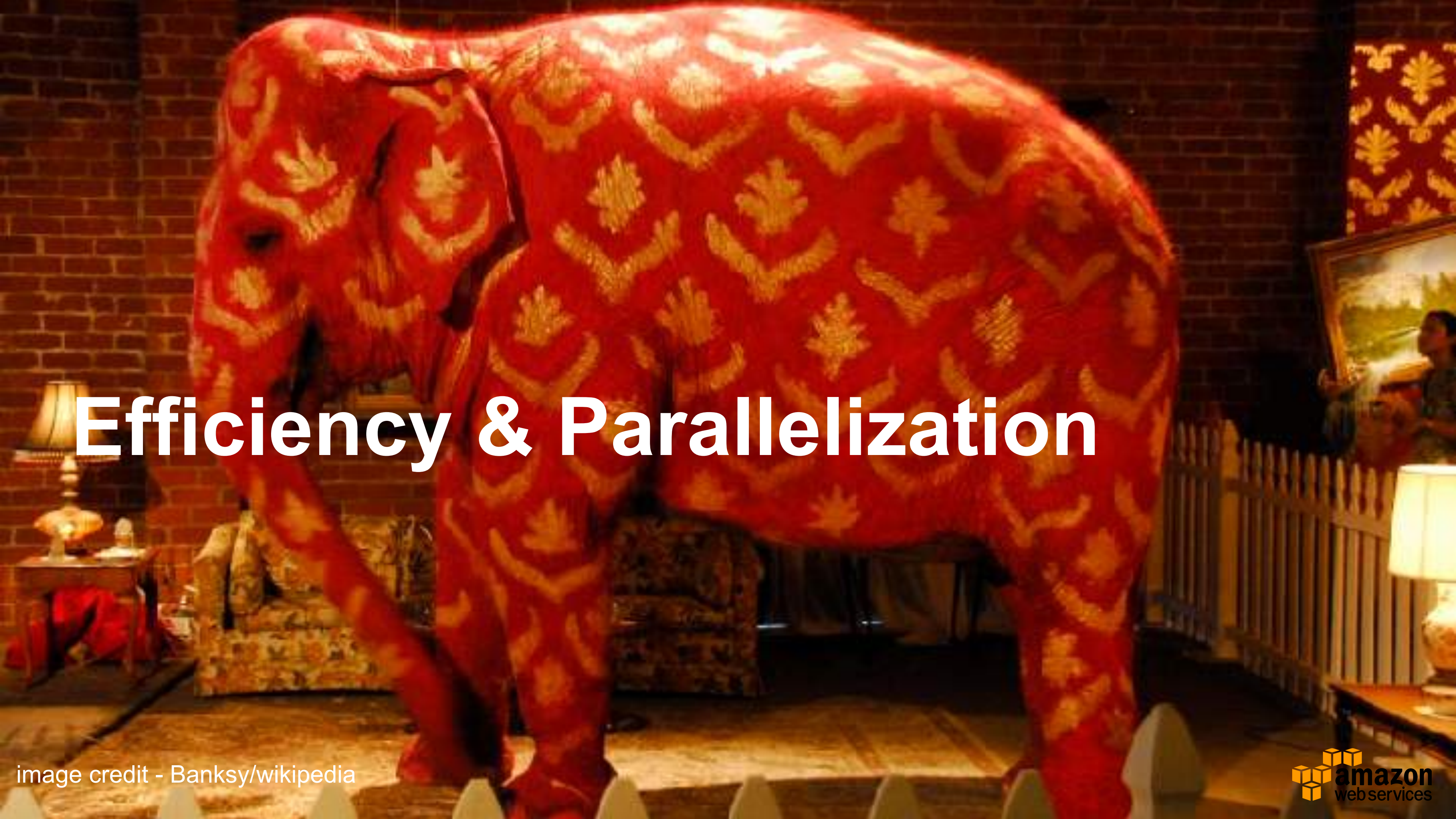
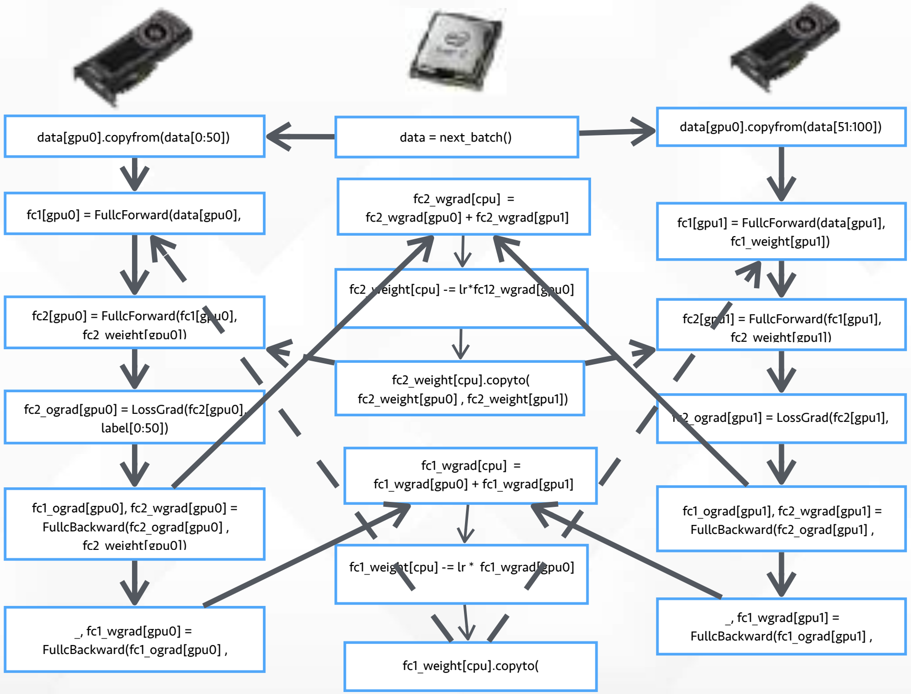


image credit - Banksy/wikipedia

Writing Parallel Programs is Painful

Dependency graph for 2-layer neural networks with 2 GPUs



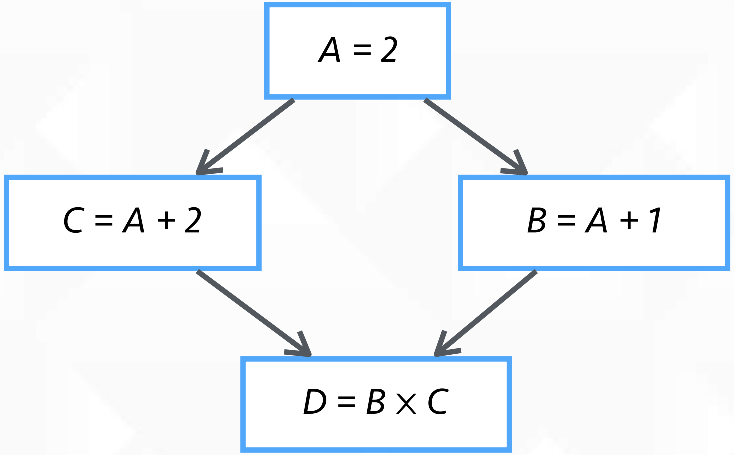
Each forward-backward-update involves O(num_layer), which is often 100—1,000, tensor computations and communications

Auto Parallelization

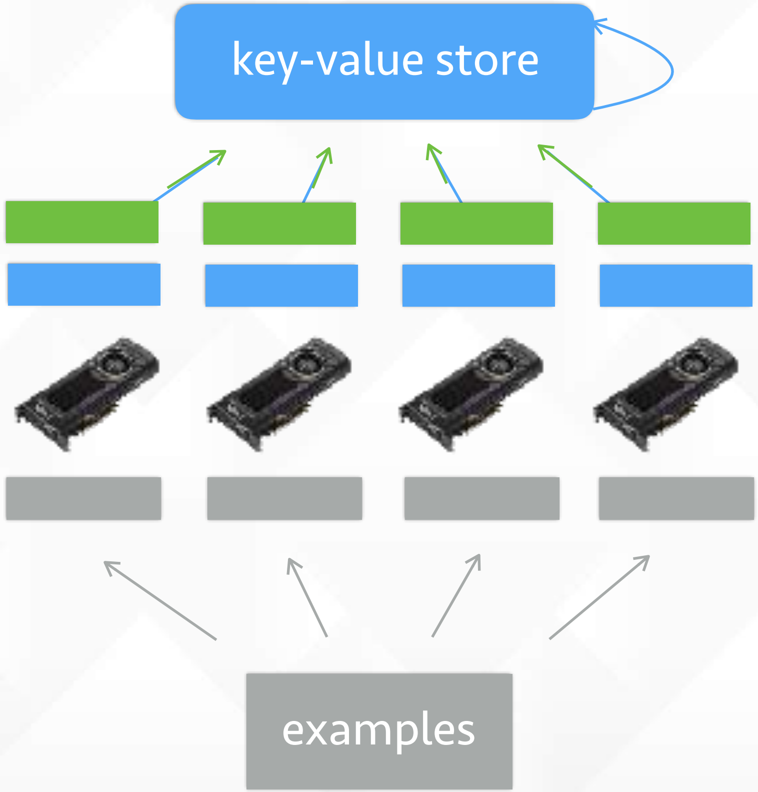
Write **serial** programs

```
import mxnet as mx
A = mx.nd.ones((2,2)) *2
C = A + 2
B = A + 1
D = B * C
```

Run in **parallel**



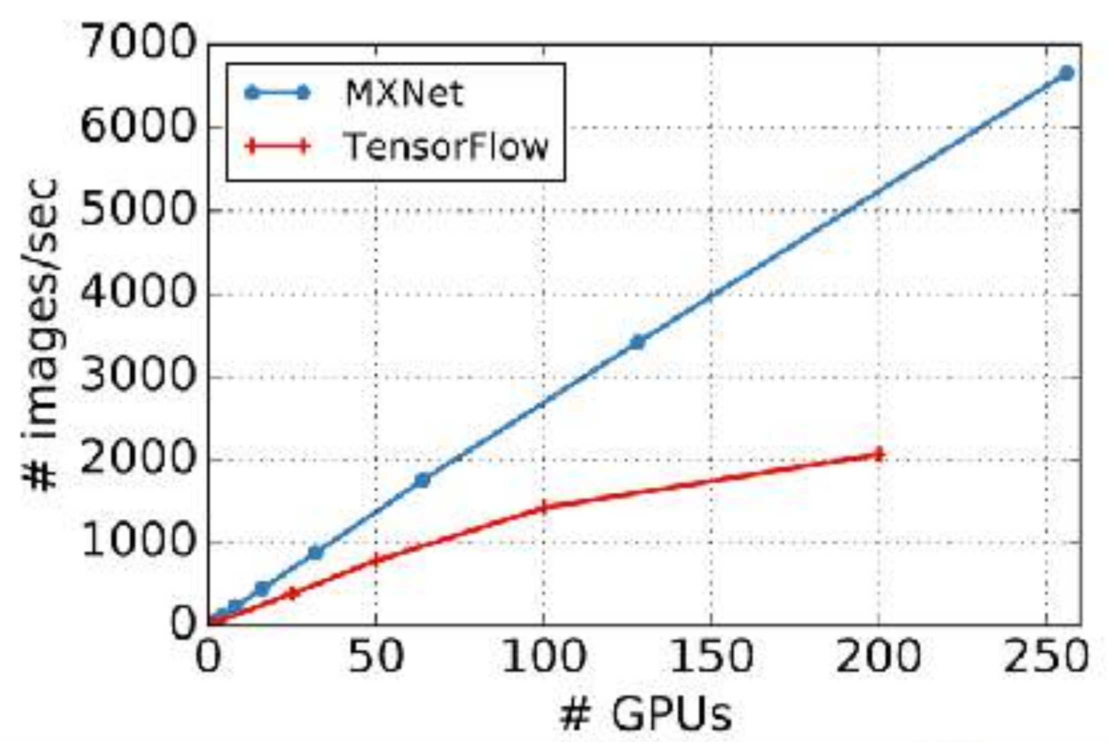
Data Parallelism



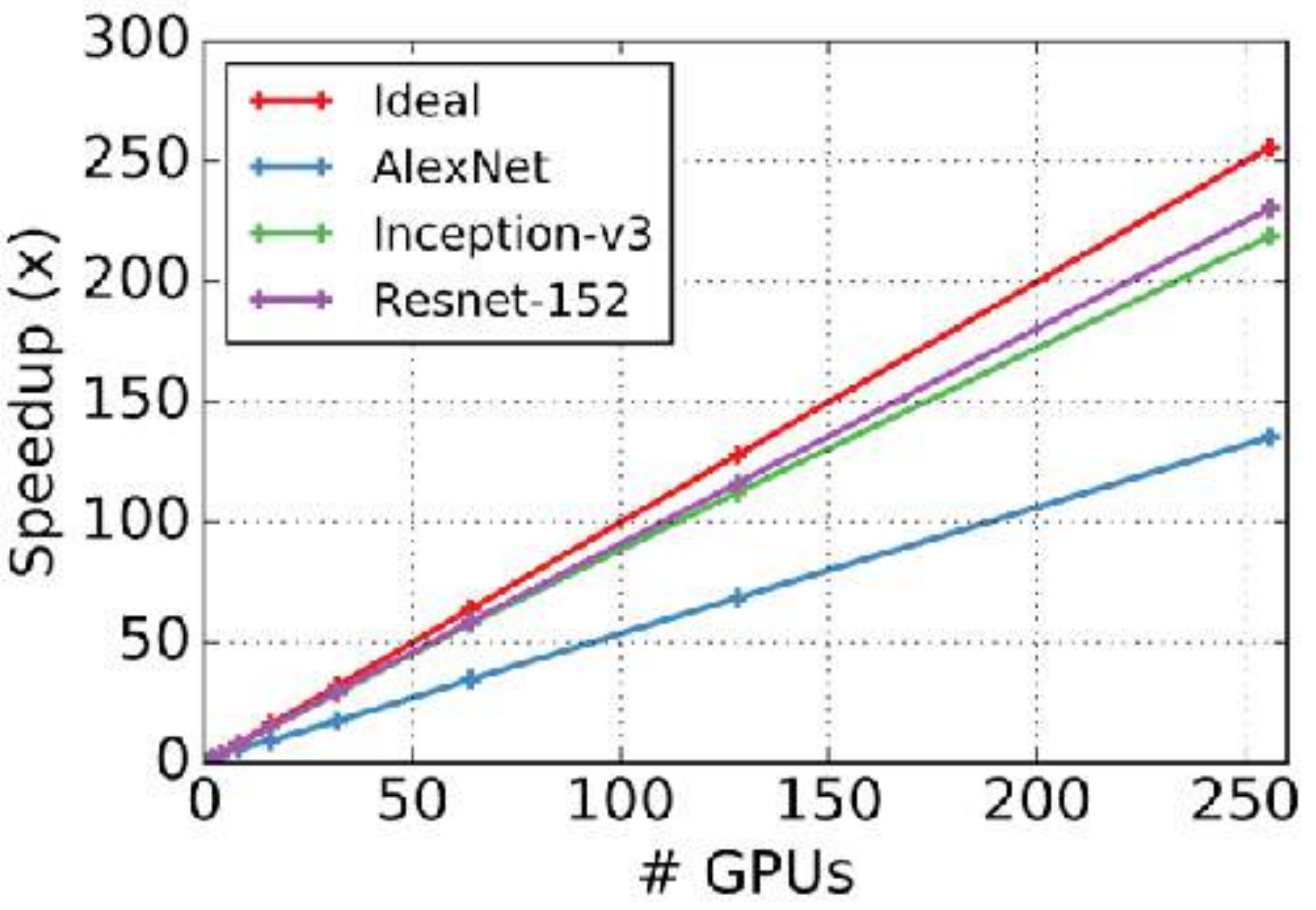
1. Read a data partition
2. Pull the parameters
3. Compute the gradient
4. Push the gradient
5. Update the parameters

Distributed Experiments

- Google Inception v3
- Increasing machines from 1 to 47
- 2x faster than TensorFlow if using more than 10 machines

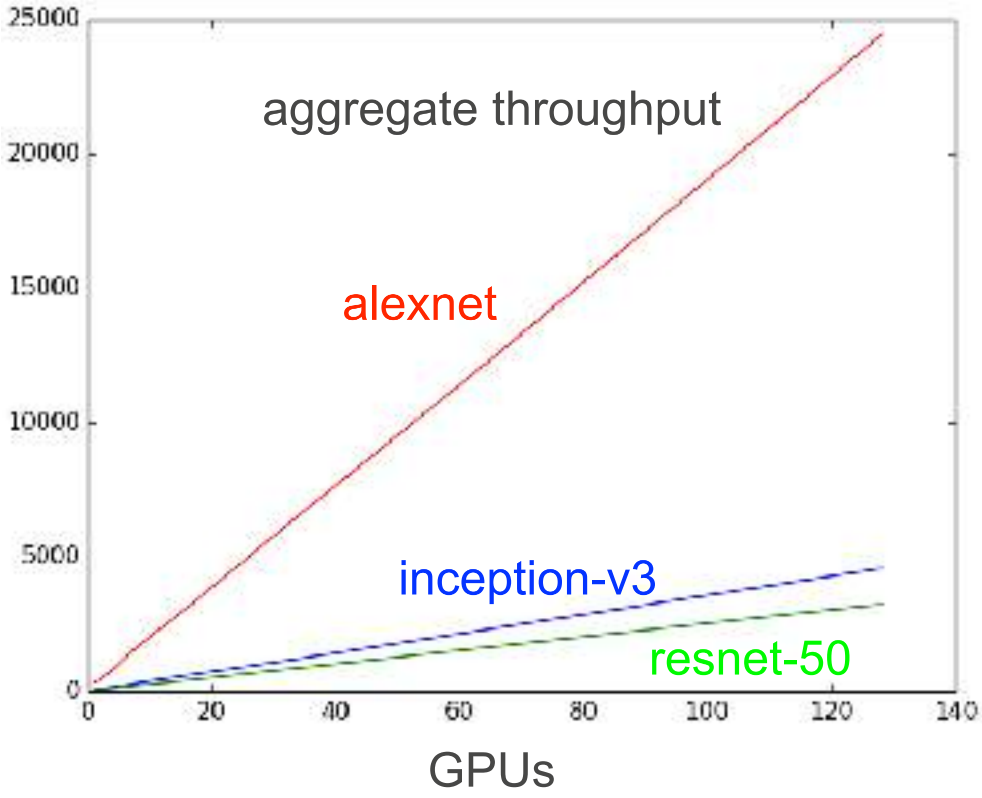


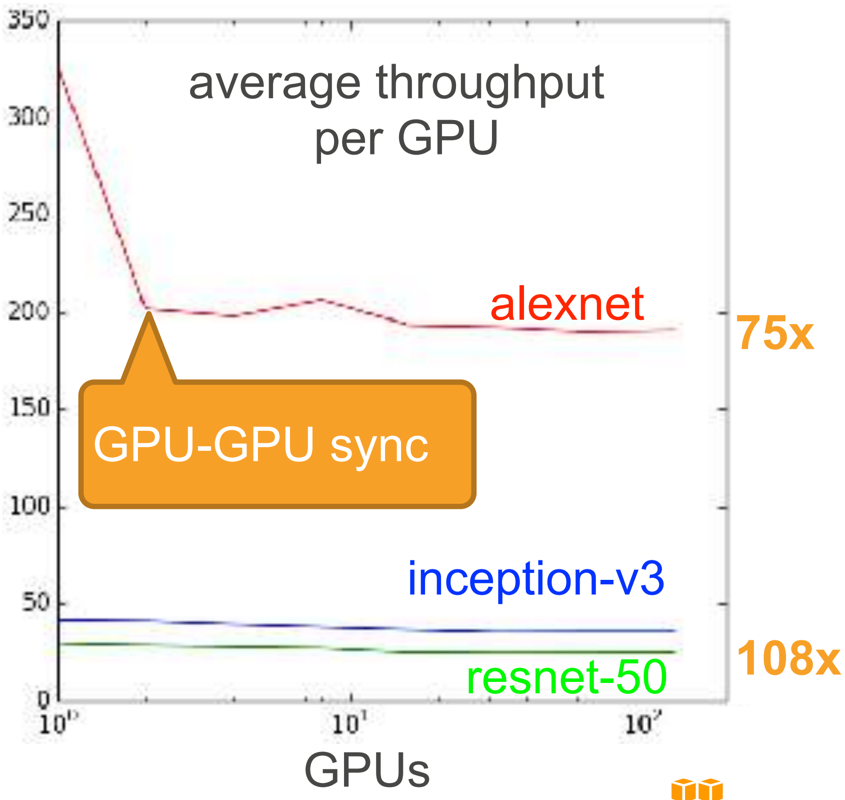
Distributed Training Speedup



- Cloud formation with Deep Learning AMI
- 16x P2.16xlarge
- Mounted on EFS
- ImageNet
  1.2M images
  1K classes
- 152-layer ResNet
  5.4d on 4x K80s
  (1.2h per epoch)
  0.22 top-1 error

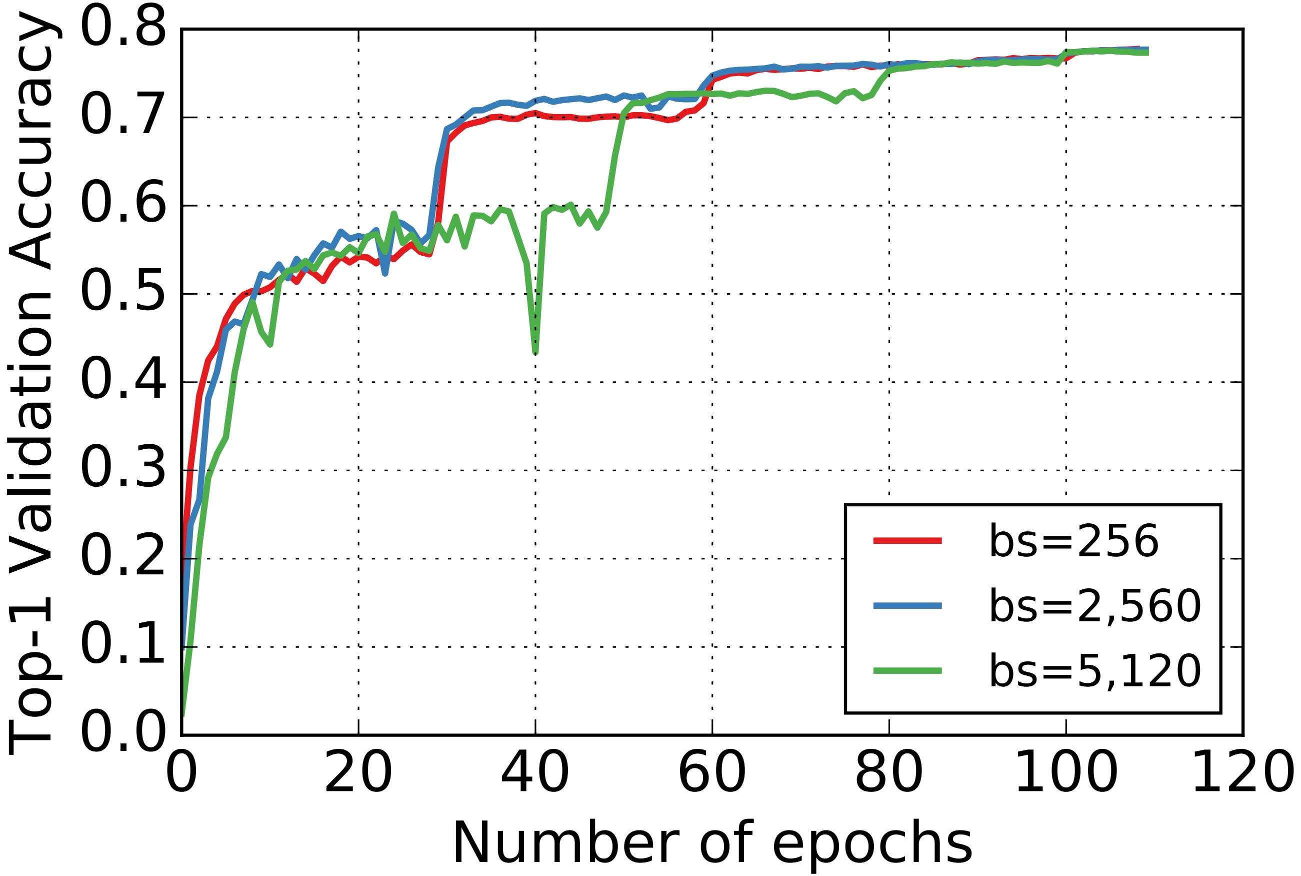
Scaling on p2.16xlarge





amazon web services

Distributed Training Convergence



amazon web services

3. AMIs and Cloud Formation Templates

- Amazon Machine Images (AMI)
- Deep Learning Frameworks
- Cloud Formation Templates

image credit - publicdomainpibtures

amazon web services

Amazon Machine Image for Deep Learning

bit.ly/deepami bit.ly/deepubuntu

- Tool for data scientists and developers
- Setting up a DL system takes (install) time & skill
  - Keep packages up to date and compiled (MXNet, TensorFlow, Caffe, Torch, Theano, Keras)
  - Anaconda, Jupyter, Python 2 and 3
  - **NVIDIA** Drivers for G2 and P2 instances
  - **Intel MKL** Drivers for all other instances (C4, M4, …)

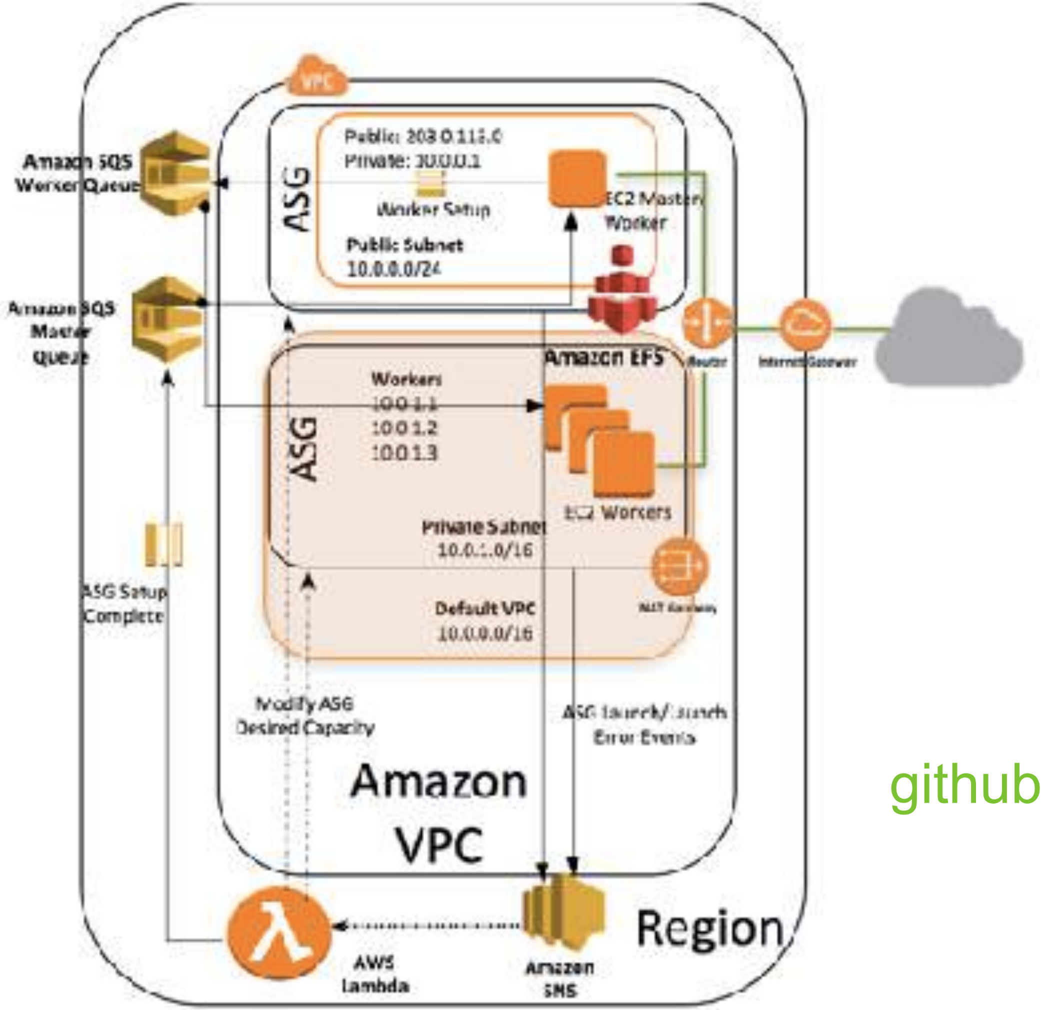
Getting started

```
acbc32cf4de3:image-classification smola$ ssh ec2-user@54.210.246.140
Last login: Fri Nov 11 05:58:58 2016 from 72-21-196-69.amazon.com
=============================================================================
       __|  __|_  )
       _|  (     /   Deep Learning AMI for Amazon Linux
      ___|\___|___|

This is beta version of the Deep Learning AMI for Amazon Linux.

The README file for the AMI ➜➜➜➜➜➜➜➜➜➜➜➜➜➜➜➜➜➜➜➜  /home/ec2-user/src/README.md
Tests for deep learning frameworks ➜➜➜➜➜➜➜➜➜➜➜➜   /home/ec2-user/src/bin
=============================================================================

7 package(s) needed for security, out of 75 available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2016.09 is available.
[ec2-user@ip-172-31-55-21 ~]$ cd src/
[ec2-user@ip-172-31-55-21 src]$ ls
anaconda2  bazel  caffe   cntk   keras  mxnet                 OpenBLAS  README.md   Theano
anaconda3  bin    caffe3  demos  logs   Nvidia_Cloud_EULA.pdf opencv    tensorflow  torch
```



AWS CloudFormation Template for Deep Learning

github.com/awslabs/deeplearning-cfn

AWS CloudFormation Components

- **VPC** in the customer account.
- The requested number of **worker instances** in an Auto Scaling group within the VPC. Workers are launched in a **private subnet**.
- **Master instance** in a separate Auto Scaling group that acts as a proxy to enable connectivity to the cluster via **SSH**.
- Two security groups that open ports on the **private subnet** for communication between the master and workers.
- **IAM role** that allows users to access and query Auto Scaling groups and the private IP addresses of the EC2 instances.
- **NAT gateway** used by instances within the VPC to talk to the outside.

Summary

- **Memory**
  - Recommender Systems
  - SVM optimization
- **Computation**
  - Hashing and samplers
  - Lock free optimization
- **MxNet**
  language, parallelization, AWS Templates

We are hiring!

aws-ai-event-recruiting@amazon.com

amazon web services