Future of Computing II: What's So Special About Big Learning?

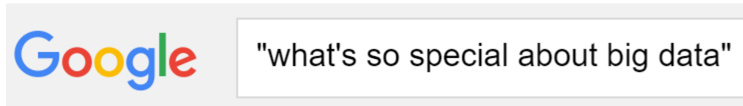
15-213: Introduction to Computer Systems

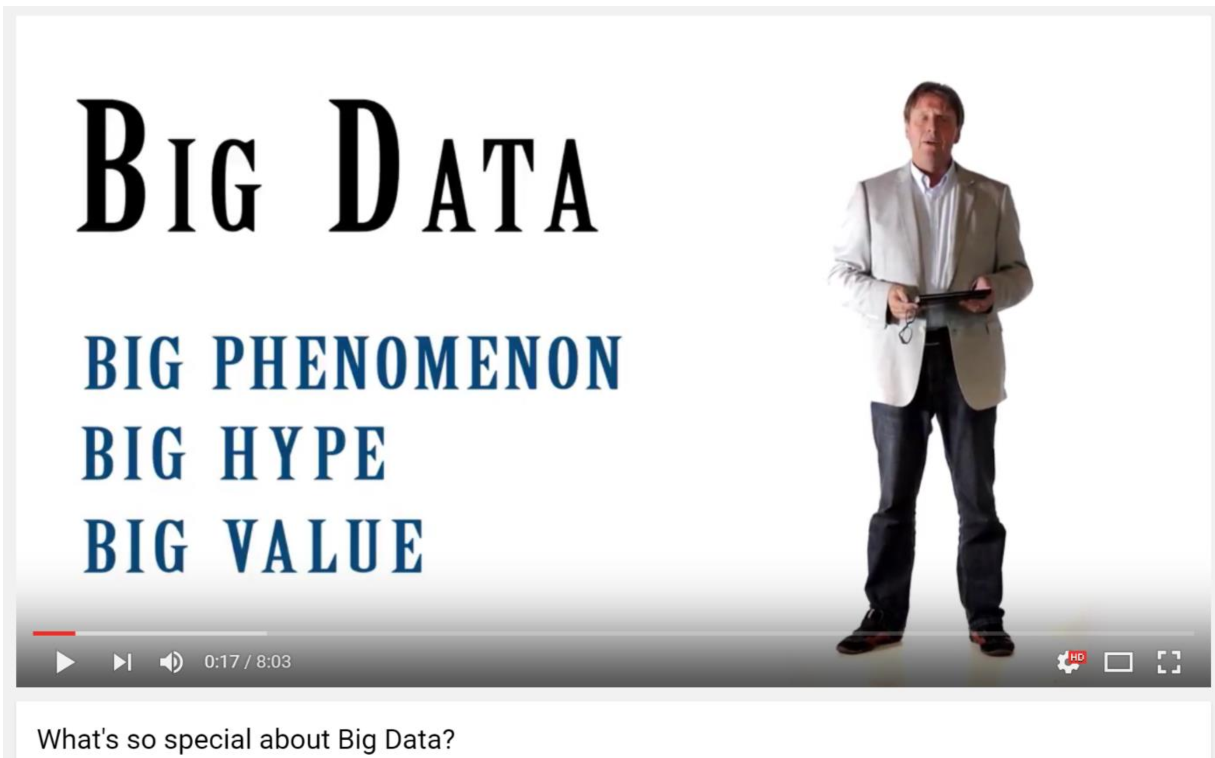
28th Lecture, Dec. 6, 2016

Instructor:

Phil Gibbons

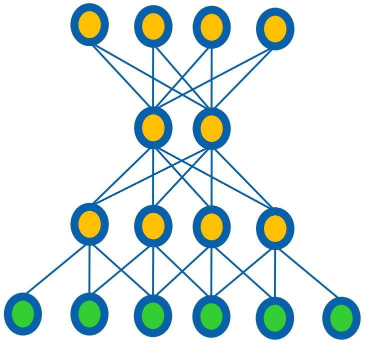
What’s So Special about…Big Data?





Focus of this Talk: Big Learning

- **Machine Learning over Big Data**
- **Examples:**
  - Collaborative Filtering (via Matrix Factorization)
    - Recommending movies
  - Topic Modeling (via LDA)
    - Clusters documents into K topics
  - Multinomial Logistic Regression
    - Classification for multiple discrete classes
  - Deep Learning neural networks:



  - Also: Iterative graph analytics, e.g. PageRank

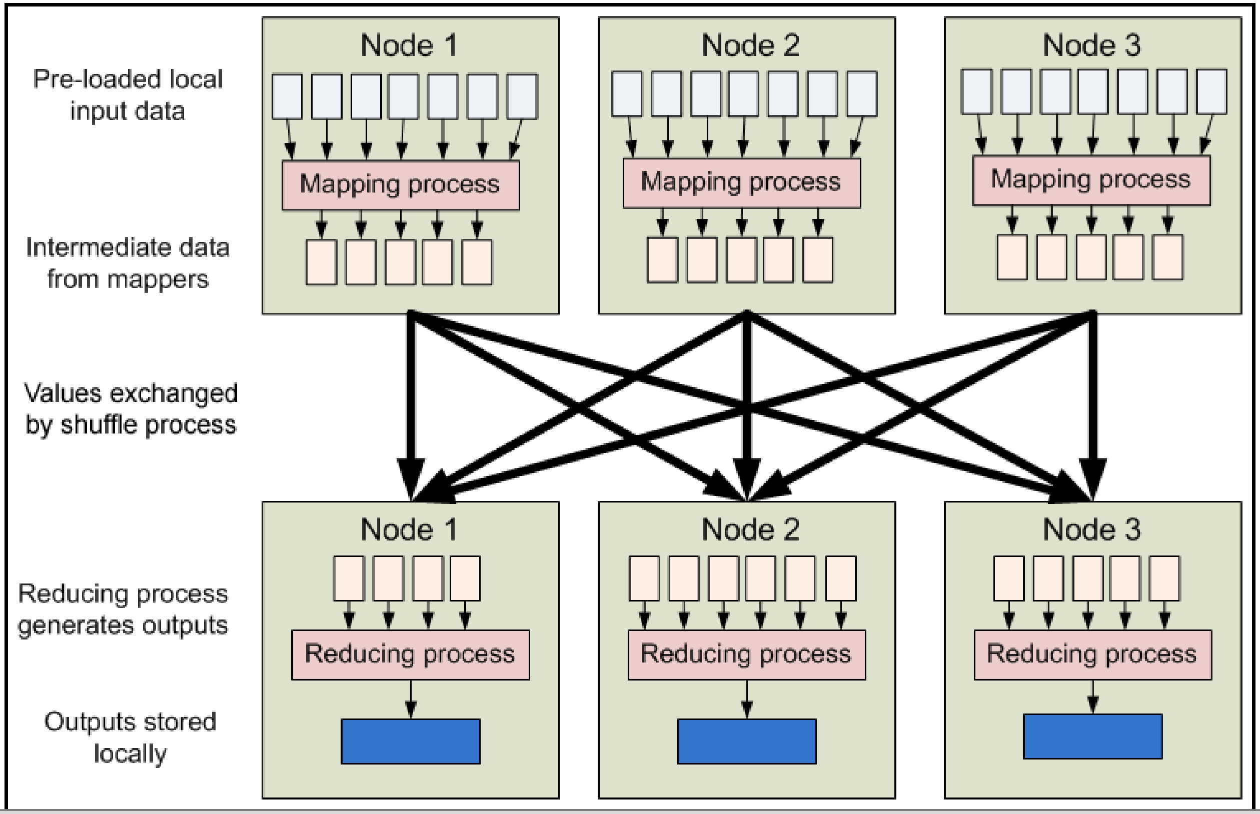
Big Learning Frameworks & Systems

- **Goal: Easy-to-use programming framework for Big Data Analytics that delivers good performance on large (and small) clusters**

- **A few popular examples (historical context):**
  - Hadoop (2006-)
  - GraphLab / Dato (2009-)
  - Spark / Databricks (2009-)

Hadoop

- **Hadoop Distributed File System (HDFS)**
- **Hadoop YARN resource scheduler**
- **Hadoop MapReduce**

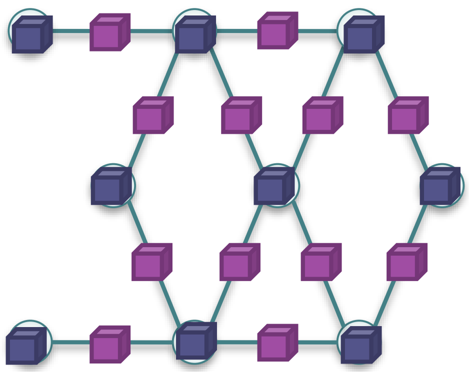


Key Learning: Ease of use trumps performance

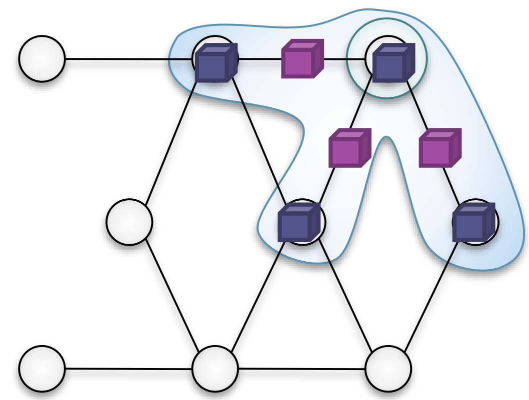
GraphLab

Graph Parallel: “Think like a vertex”

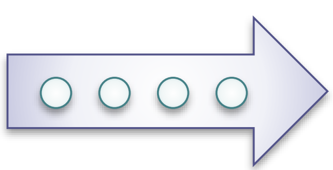
Graph Based
Data Representation



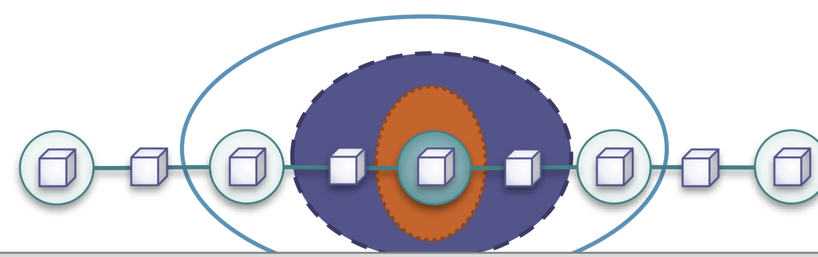
Update Functions
User Computation



Scheduler



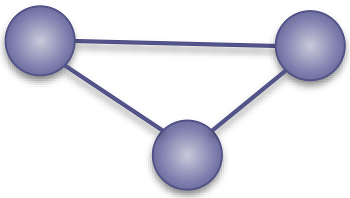
Consistency Model



Slide courtesy of Carlos Guestrin

Key Learning: Graph Parallel is quite useful

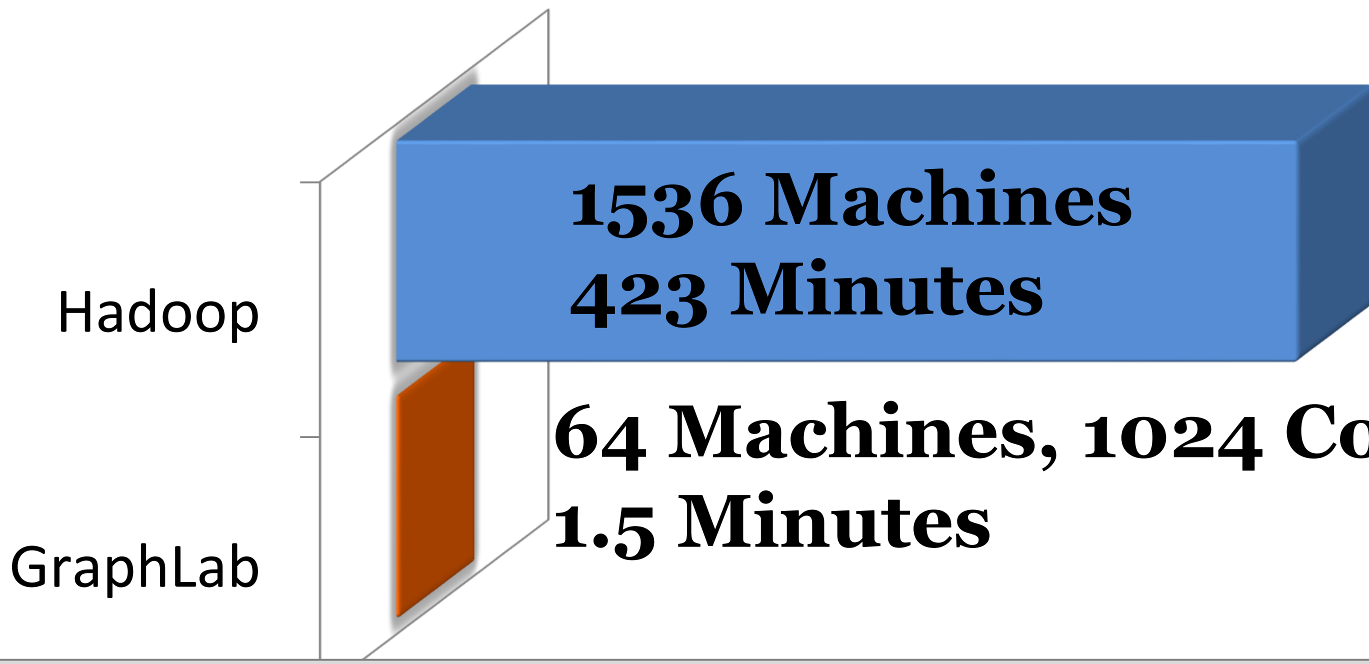
Triangle Counting* in Twitter Graph



40M Users
1.2B Edges

***How often are two of a user's friends also friends?**

Total: 34.8 Billion Triangles



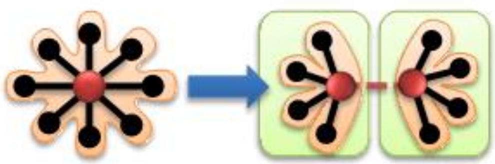
Key Learning:
Graph Parallel is MUCH faster than Hadoop!

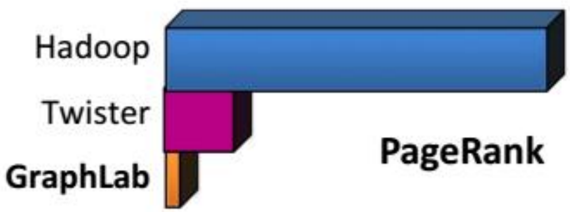
GraphLab & GraphChi

GraphLab 2

Distributed Graph Processing System

How Fast Can we Go?





Disk/SSD Graph Processing System

How Large Can we Go?

20B edges on one Laptop

How to handle high degree nodes: GAS approach

Can do fast BL on a machine w/SSD-resident data

GraphLab Create

Data scientist: inspiration to production

Prototype

Use my laptop
Variety of data
Not toy data scales

Language I love
Iterate quickly

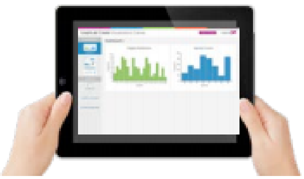
Production

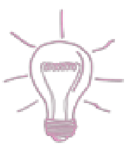
Clean → Learn → Deploy
data pipeline

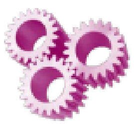
predictive service

EC2

Monitor





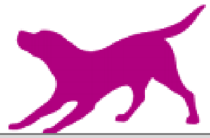


GraphLab Create

Analyze **big data** on **one machine**
graphs, tables, text, images
in **Python**
doesn't have to fit in memory

Distribute in production
with same code
on EC2, Yarn,…

GraphLab Canvas: Monitor & visualize
from prototype to production



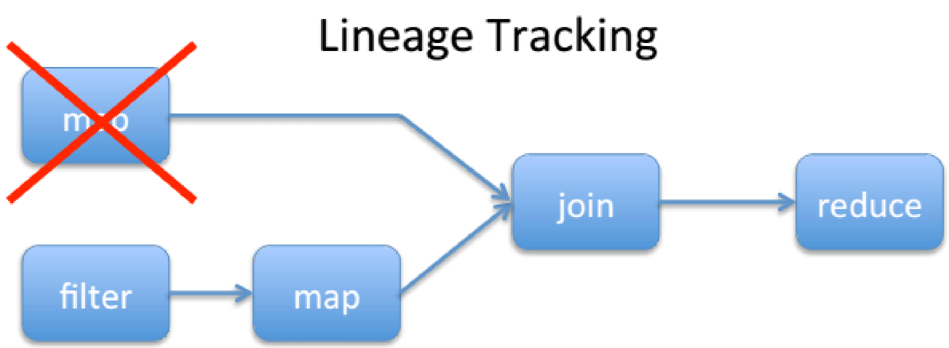
User experience is paramount for customers

Spark: Key Idea

Features:
- In-memory speed w/fault tolerance via lineage tracking
- Bulk Synchronous

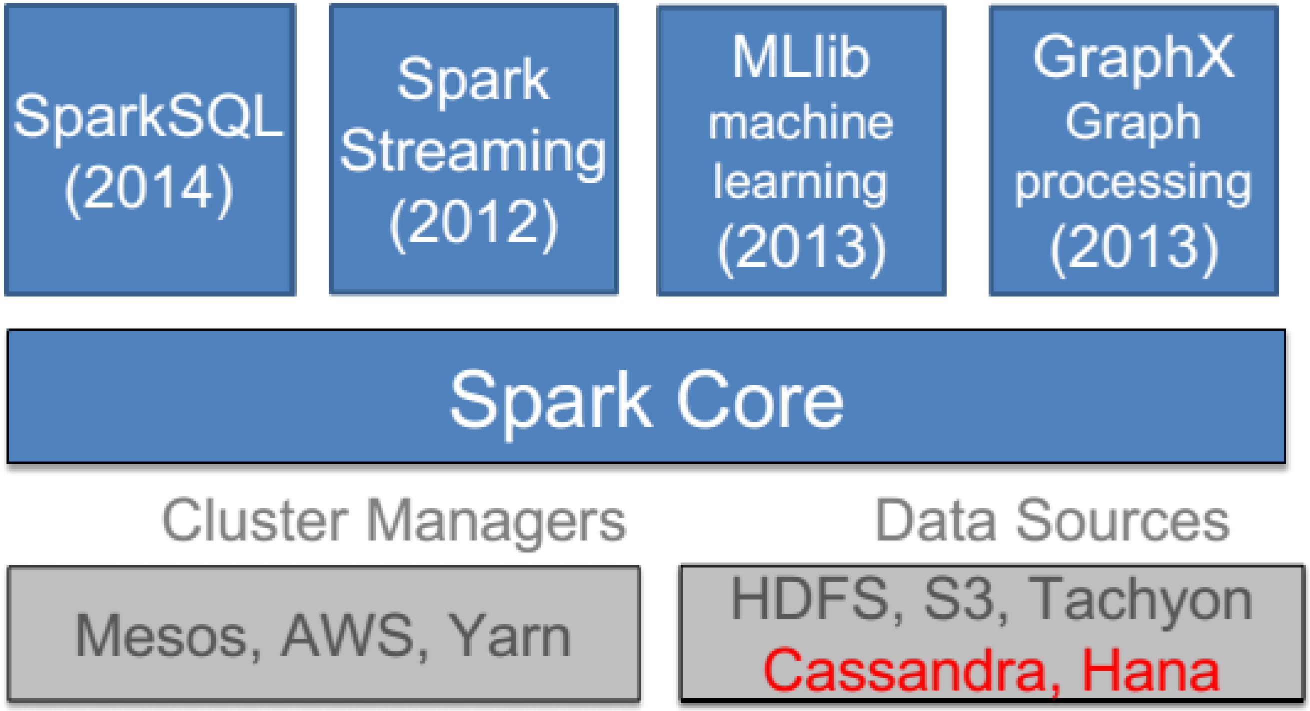
Resilient Distributed Datasets: A Fault-Tolerant Abstraction for InMemory Cluster Computing, [Zaharia et al, NSDI'12, best paper]

A restricted form of shared memory, based on coarse-grained deterministic transformations rather than fine-grained updates to shared state: expressive, efficient and fault tolerant



In-memory compute can be fast & fault-tolerant

Spark Stack continued innovations



(Start to) Build it and they will come (help build it)
1000+ companies use Spark & many contribute

A Brave New World

Spark Timeline

- Research breakthrough in 2009
- First open source release in 2011
- Into Apache Incubator in 2013
- In all major Hadoop releases by 2014

- Pipeline of research breakthroughs (publications in best conferences) fuel continued leadership & uptake
- Start-up (Databricks), Open Source Developers, and Industry partners (IBM, Intel) make code commercial-grade

Fast path for Academics impact via Open Source: Pipeline of research breakthroughs into widespread commercial use in 2 years!

Big Learning Frameworks & Systems

- **Goal: Easy-to-use programming framework for Big Data Analytics that delivers good performance on large (and small) clusters**

- **A few popular examples (historical context):**
  - Hadoop (2006-)
  - GraphLab / Dato (2009-)
  - Spark / Databricks (2009-)

- **Our Idea: Discover & take advantage of distinctive properties ("what's so special") of Big Learning training algorithms**

What's So Special about Big Learning? ...A Mathematical Perspective

- **Formulated as an optimization problem**
  - Use training data to learn model parameters
- **No closed-form solution, instead algorithms iterate until convergence**
  - E.g., **Stochastic Gradient Descent** for Matrix Factorization or Multinomial Logistic Regression, LDA via Gibbs Sampling, Deep Learning, Page Rank

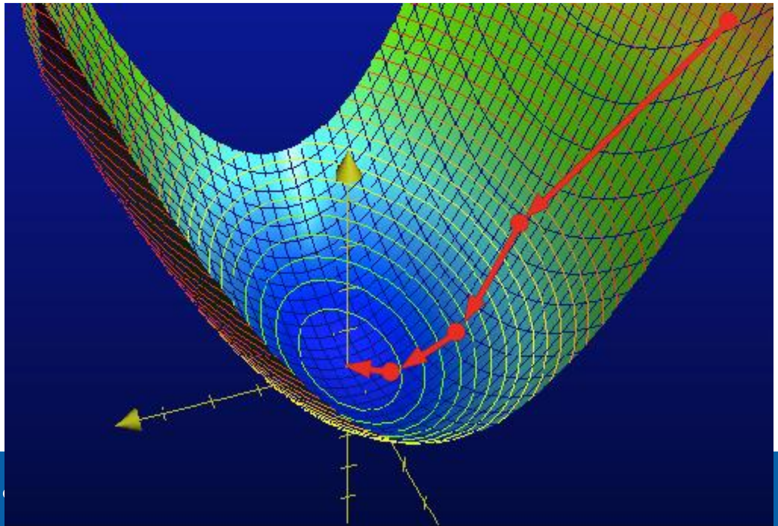


Image from charlesfranzen.com

What's So Special about Big Learning? ...A Distributed Systems Perspective

The Bad News

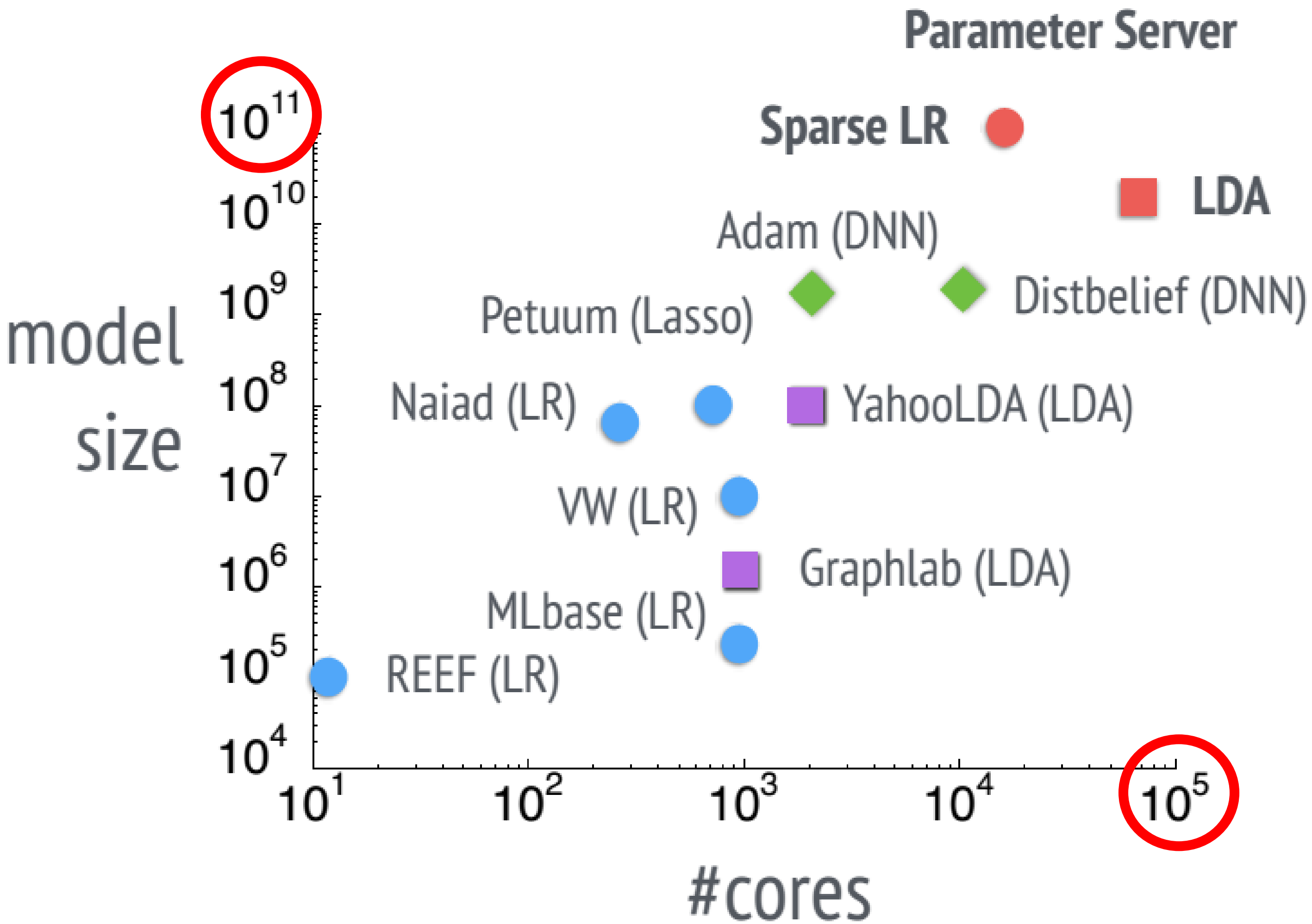
- **Lots of Computation / Memory**
  - Many iterations over Big Data
  - Big Models
  - ➡ Need to distribute computation widely
- **Lots of Communication / Synchronization**
  - Not readily "partitionable"

➡ **Model Training is SLOW**

- hours to days to weeks, even on many machines

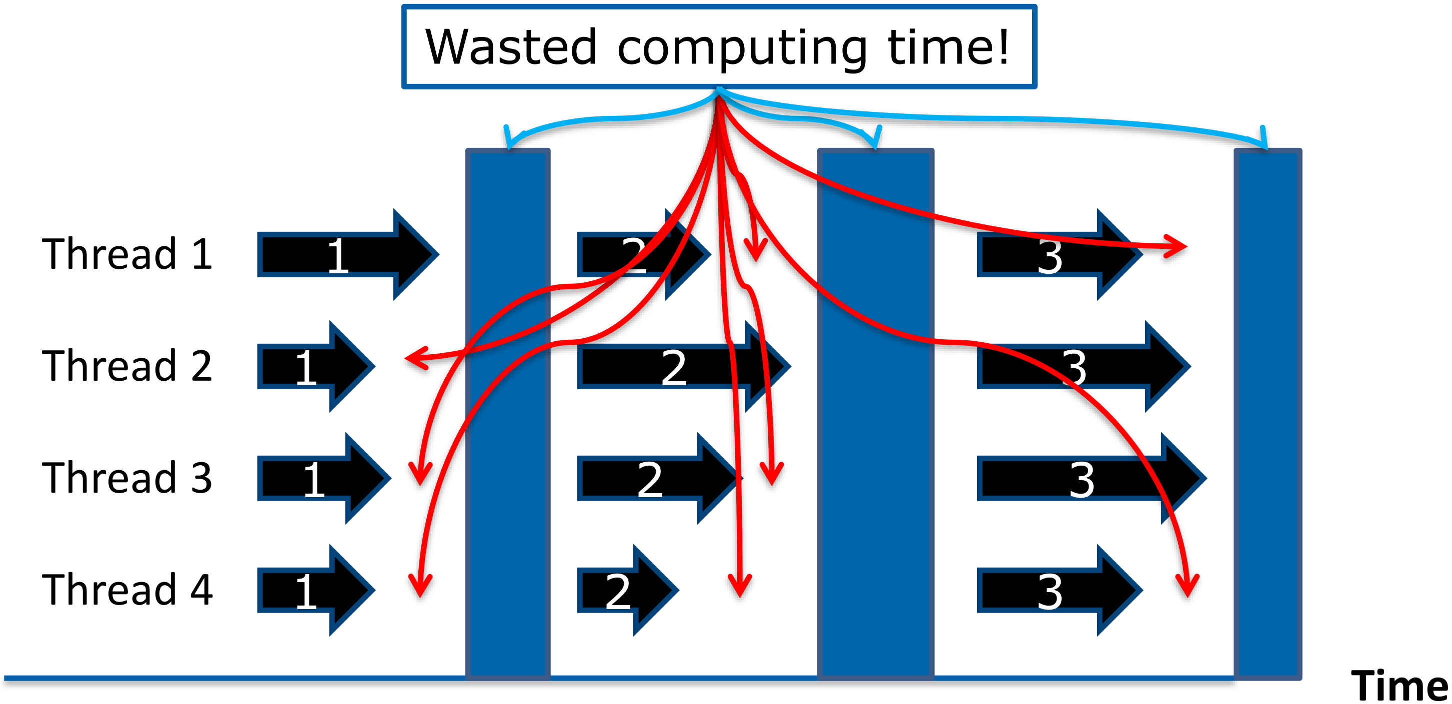
...why good distributed systems research is needed!

Big Models, Widely Distributed



[Li et al, OSDI’14]

Lots of Communication / Synchronization e.g. in BSP Execution (Hadoop, Spark)



- **Exchange ALL updates at END of each iteration**
  - ➡ Frequent, bursty communication
- **Synchronize ALL threads each iteration**
  - ➡ Straggler problem: stuck waiting for slowest

What's So Special about Big Learning? …A Distributed Systems Perspective

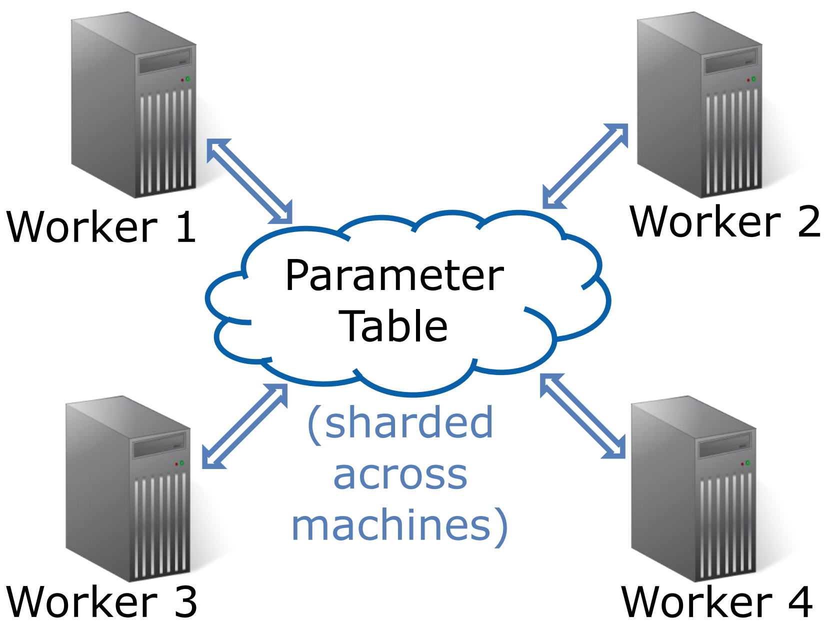
The Good News

1. Commutative/Associative parameter updates
2. Tolerance for lazy consistency of parameters
3. Repeated parameter data access pattern
4. Intra-iteration progress measure
5. Parameter update importance hints
6. Layer-by-layer pattern of deep learning

…can exploit to run orders of magnitude faster!

Parameter Servers for Distributed ML

- **Provides all workers with convenient access to global model parameters**
- **Easy conversion of single-machine parallel ML algorithms**
  - "Distributed shared memory" programming style
  - Replace local memory access with PS access



Single Machine Parallel

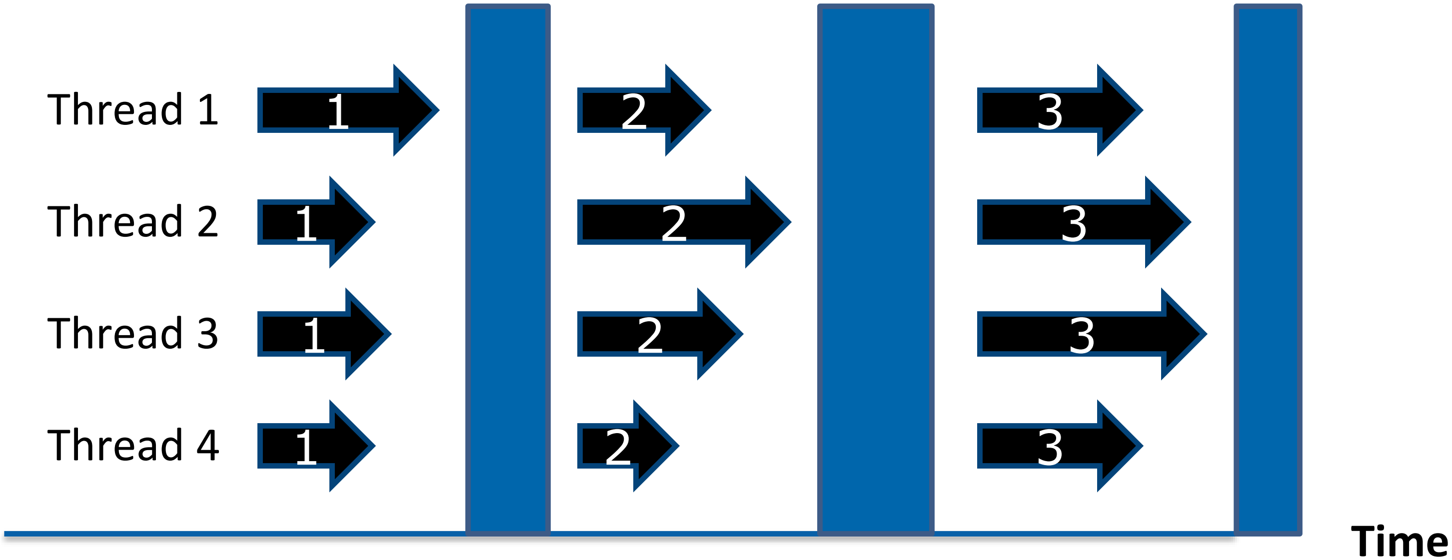
```
UpdateVar(i) {
  old = y[i]
  delta = f(old)
  y[i] += delta }
```

Distributed with PS

```
UpdateVar(i) {
  old = PS.read(y,i)
  delta = f(old)
  PS.inc(y,i,delta) }
```

[Power & Li, OSDI'10], [Ahmed et al, WSDM'12], [NIPS'13], [Li et al, OSDI'14], Petuum, MXNet, TensorFlow, etc

Cost of Bulk Synchrony (e.g., in Spark)



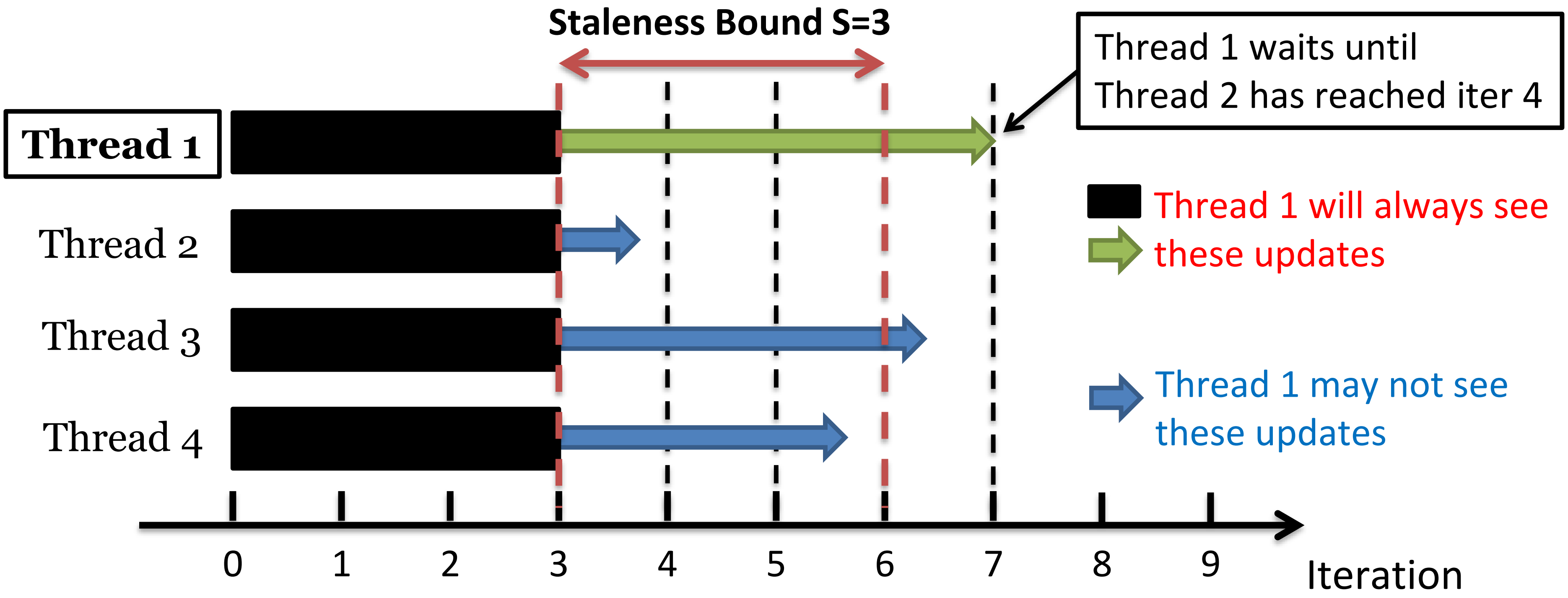
- **Exchange ALL updates at END of each iteration**
- **Synchronize ALL threads each iteration**

Bulk Synchrony => Frequent, bursty communication & stuck waiting for stragglers

But: **Fully asynchronous** => No algorithm convergence guarantees

Better idea: Bounded Staleness: All threads within S iterations

Stale Synchronous Parallel (SSP)



Fastest/slowest threads not allowed to drift >S iterations apart

Allow threads to *usually* run at own pace

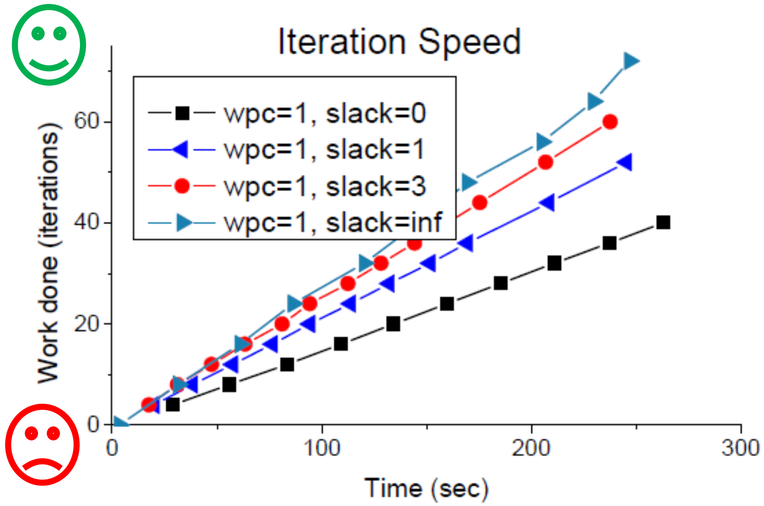
Protocol: check cache first; if too old, get latest version from network

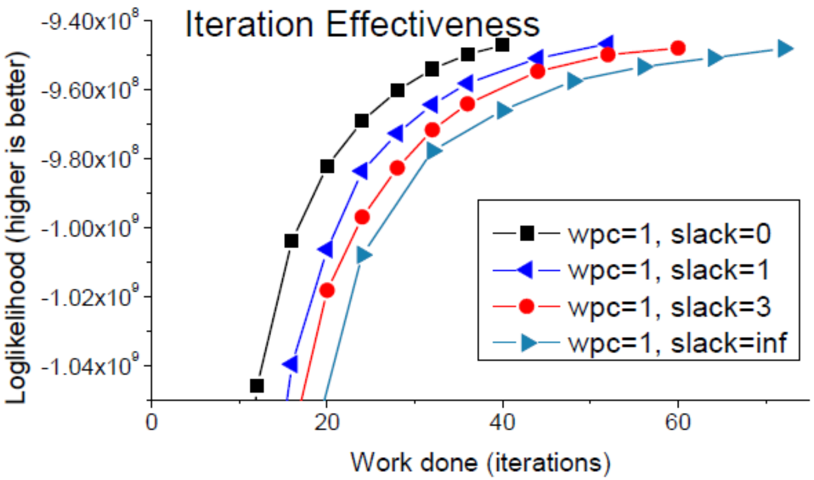
Slow threads check only every S iterations – fewer network accesses, so catch up!

Exploits: 1. commutative/associative updates &

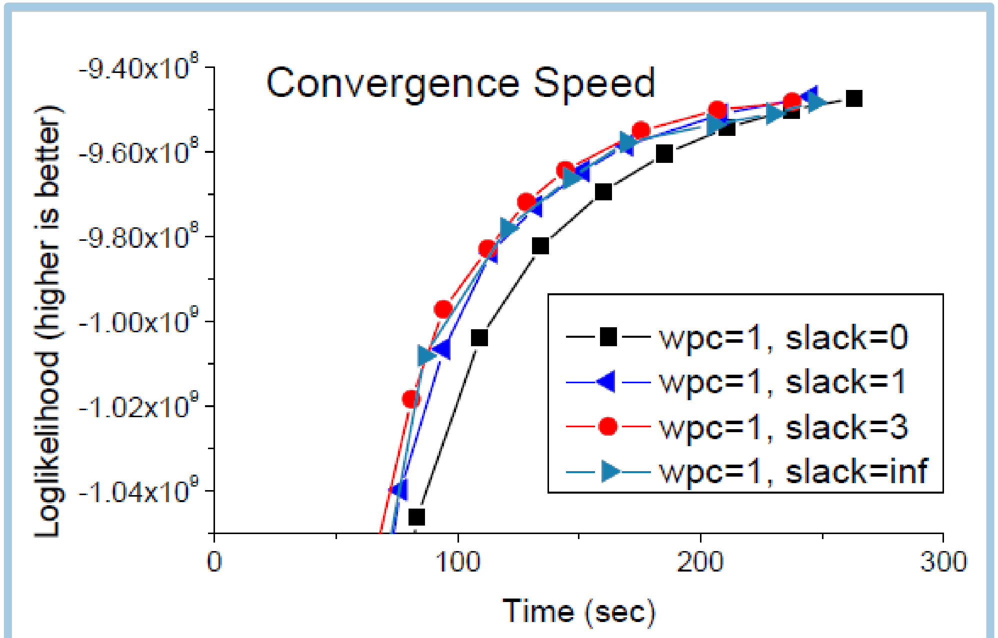
2. tolerance for lazy consistency (bounded staleness) [NIPS'13]

Staleness Sweet Spot





Topic Modeling
Nytimes dataset
400k documents
100 topics
LDA w/Gibbs sampling
8 machines x 64 cores
40Gbps Infiniband



[ATC'14]

What's So Special about Big Learning? …A Distributed Systems Perspective

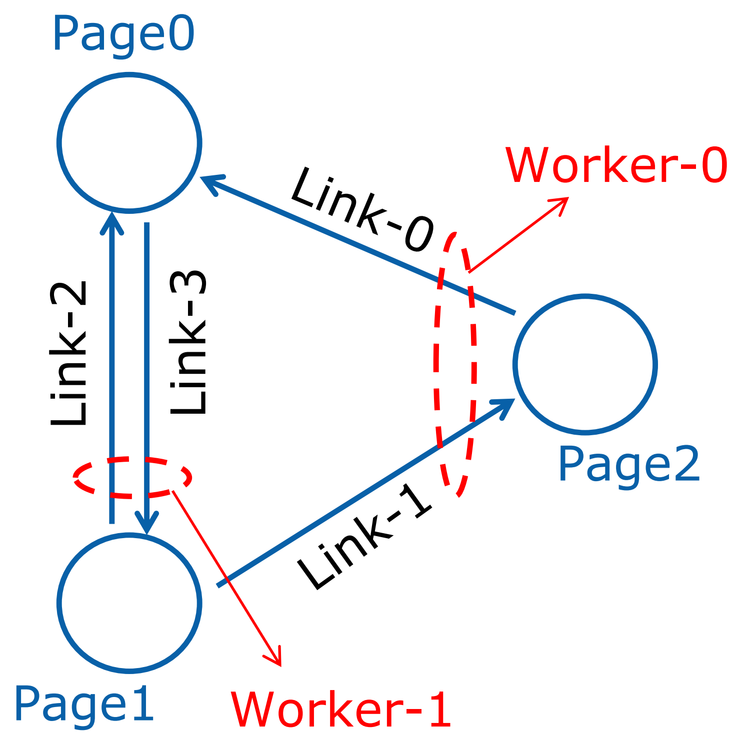
The Good News

1. Commutative/Associative parameter updates
2. Tolerance for lazy consistency of parameters
3. Repeated parameter data access pattern ⬅
4. Intra-iteration progress measure
5. Parameter update importance hints
6. Layer-by-layer pattern of deep learning

…can exploit to run orders of magnitude faster!

Repeated Data Access in PageRank

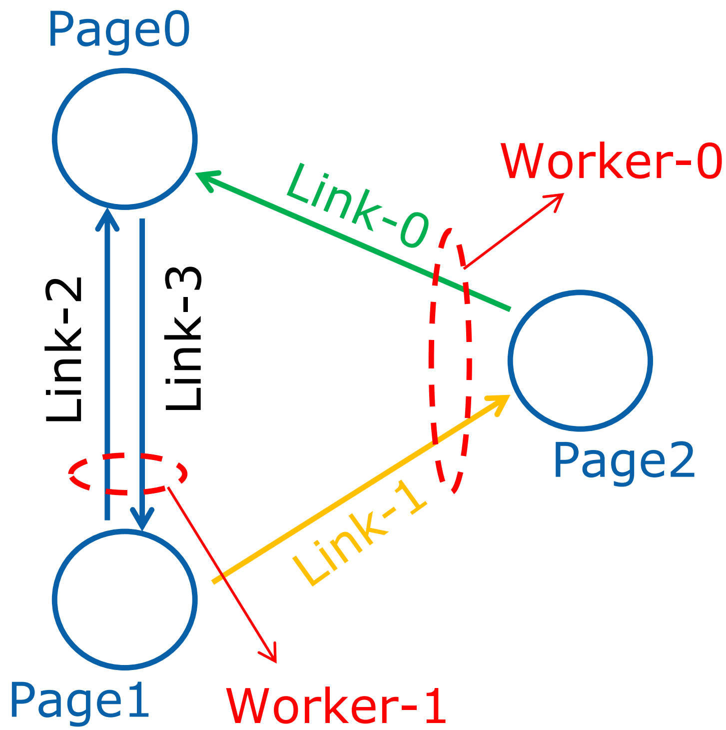
Input data: a set of links, stored locally in workers
Parameter data: ranks of pages, stored in PS




```
Init ranks to random value
loop
   foreach link from i to j {
      read Rank(i)
      update Rank(j)
   }
while not converged
```

Repeated Data Access in PageRank

Input data: a set of links, stored locally in workers
Parameter data: ranks of pages, stored in PS



Worker-0

```
loop
    # Link-0
    read page[2].rank
    update page[0].rank
    # Link-1
    read page[1].rank
    update page[2].rank
    clock()
while not converged
```

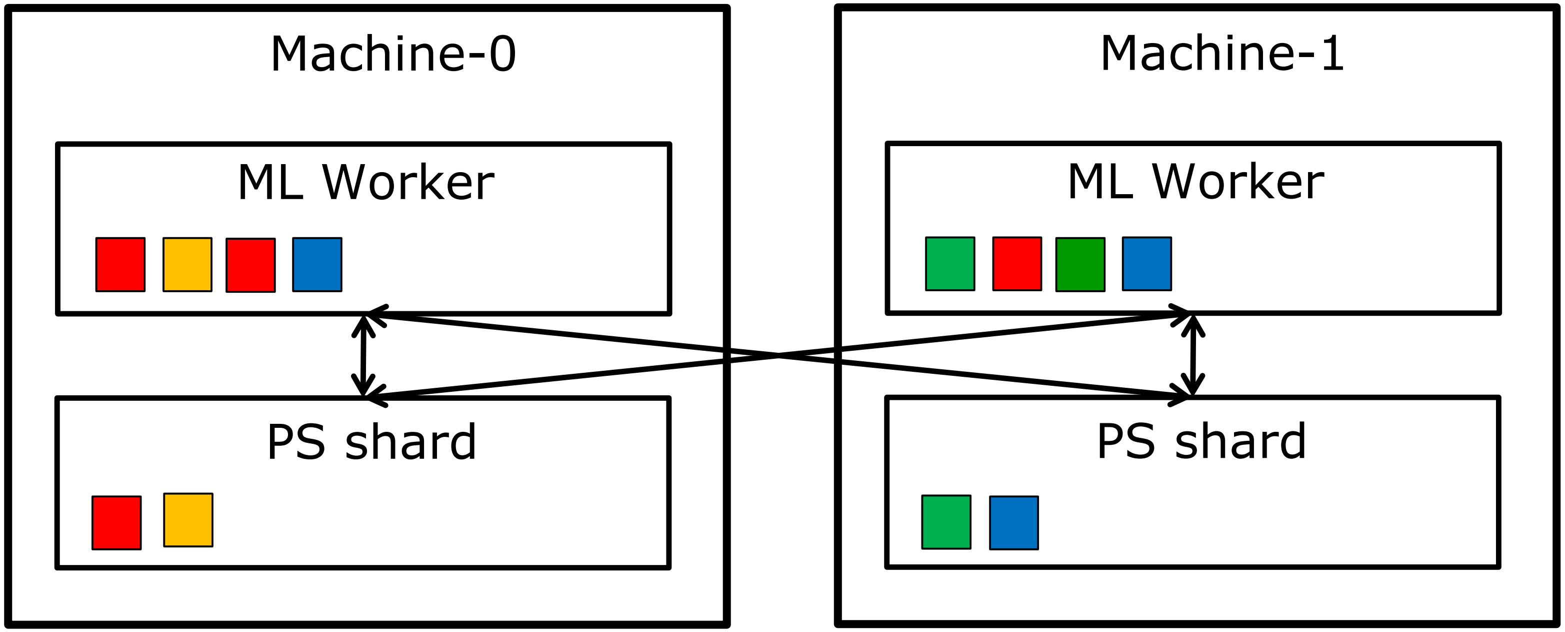
Repeated access sequence depends only on input data (not on parameter values)

Exploiting Repeated Data Access

Collect access sequence in "virtual iteration"

Enables many optimizations:

1. **Parameter data placement across machines**



Exploiting Repeated Data Access

Collect access sequence in "virtual iteration"

Enables many optimizations:

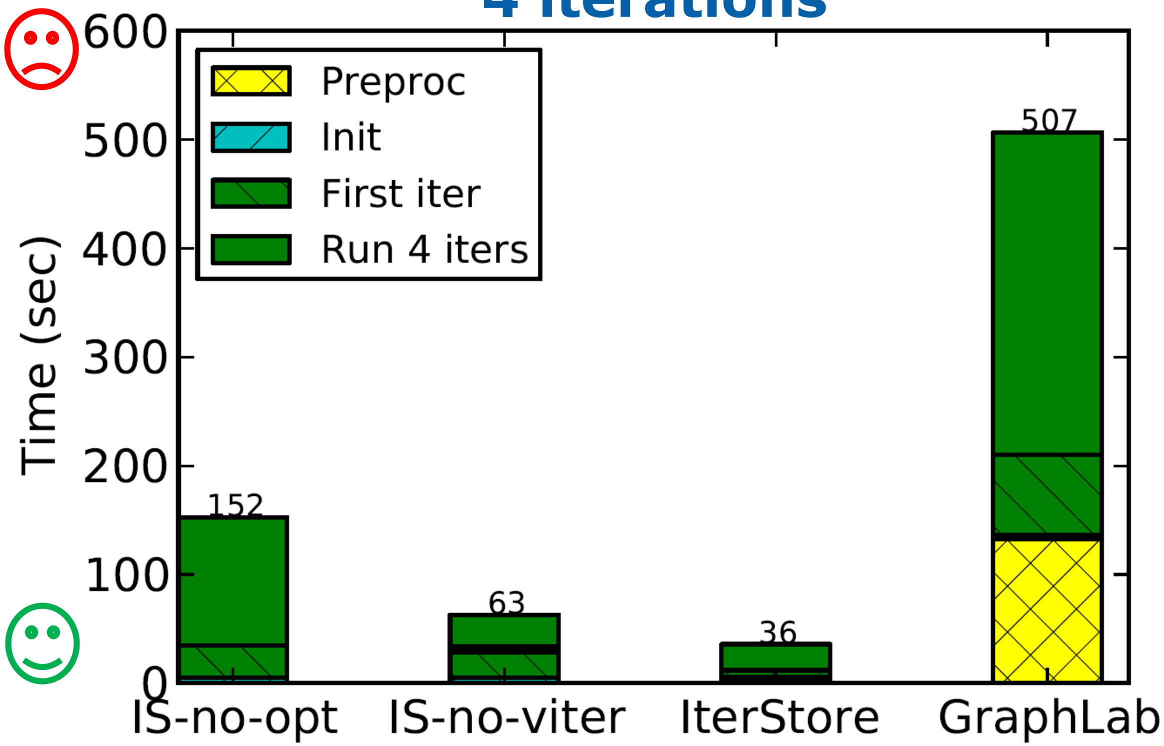
1. Parameter data placement across machines
2. Prefetching
3. Static cache policies
4. More efficient marshalling-free data structures
5. NUMA-aware memory placement

- **Benefits are resilient to moderate deviation in an iteration's actual access pattern**

IterStore: Exploiting Iterativeness

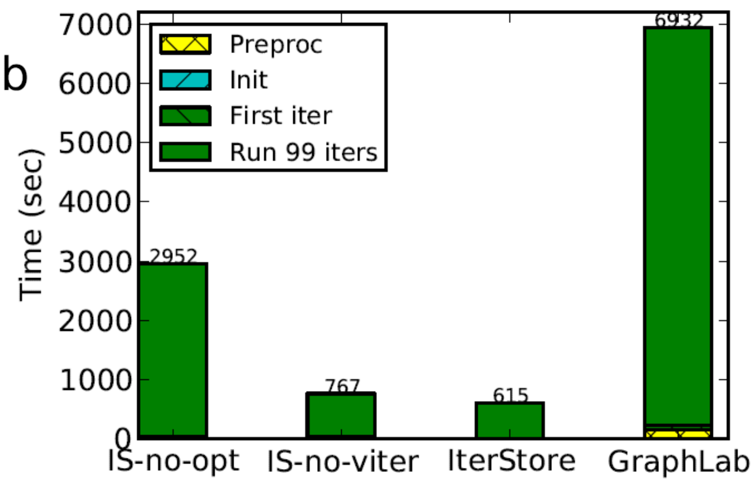
[SoCC'14]

4 iterations



Collaborative Filtering
(Matrix Factorization)
NetFlix data set
8 machines x 64 cores
40 Gbps Infiniband

99 iterations



4-5x faster than baseline
11x faster than GraphLab

What's So Special about Big Learning? …A Distributed Systems Perspective

The Good News

1. Commutative/Associative parameter updates
2. Tolerance for lazy consistency of parameters
3. Repeated parameter data access pattern
4. Intra-iteration progress measure ⬅
5. Parameter update importance hints
6. Layer-by-layer pattern of deep learning

…can exploit to run orders of magnitude faster!

Addressing the Straggler Problem

- **Many sources of transient straggler effects**
  - Resource contention
  - System processes (e.g., garbage collection)
  - Slow mini-batch at a worker

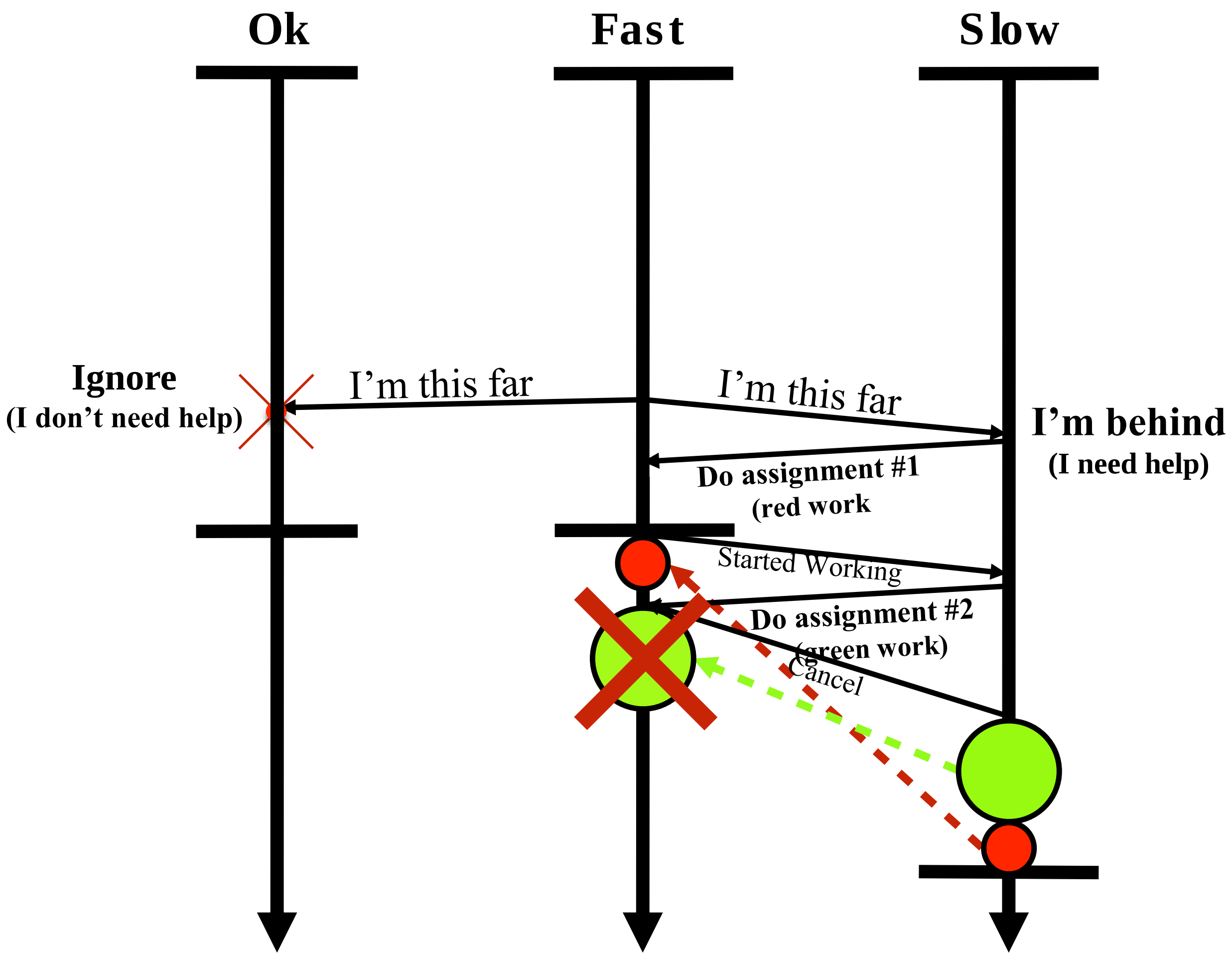
  Causes significant slowdowns for Big Learning

- **FlexRR: SSP + Low-overhead work migration (RR) to mitigate transient straggler effects**
  - Simple: Tailored to Big Learning's special properties

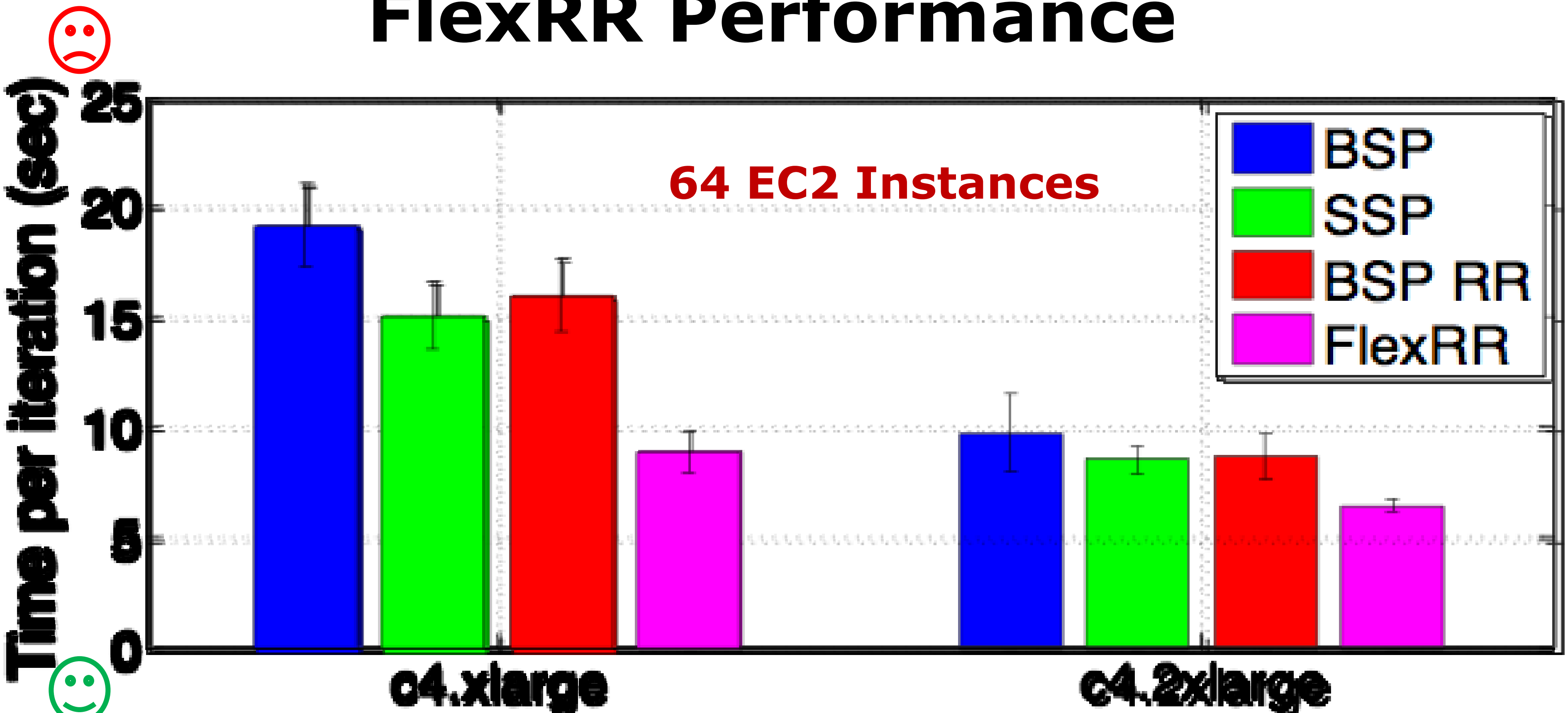
    E.g., cloning (used in MapReduce) would break the algorithm (violates idempotency)!
  - Staleness provides slack to do the migration

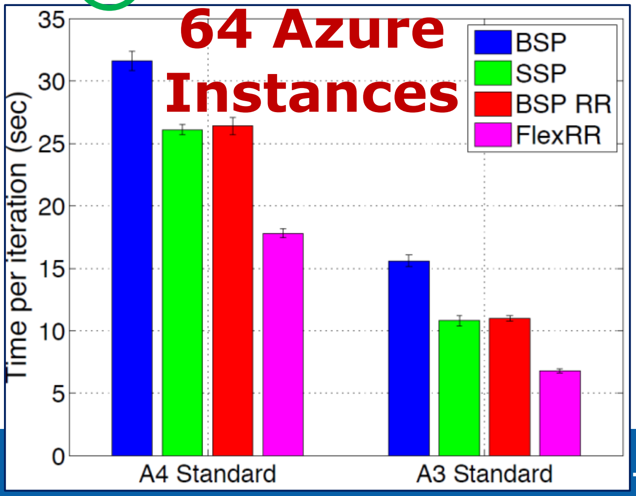
Rapid-Reassignment Protocol

- Multicast to preset possible helpees (has copy of tail of helpee's input data)
- Intra-iteration progress measure: percentage of input data processed
- Can process input data in any order
- Assignment is percentage range
- State is only in PS
- Work must be done exactly once



FlexRR Performance





Matrix Factorization
Netflix dataset

[SoCC'16]

Both SSP & RR required.
Nearly ideal straggler mitigation

What's So Special about Big Learning? ...A Distributed Systems Perspective

The Good News

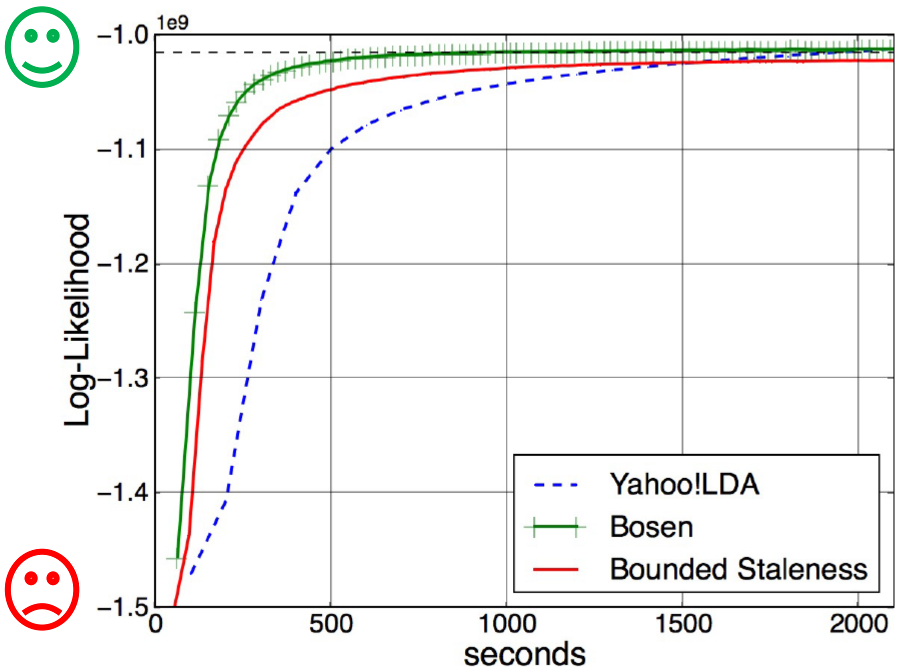
1. Commutative/Associative parameter updates
2. Tolerance for lazy consistency of parameters
3. Repeated parameter data access pattern
4. Intra-iteration progress measure
5. Parameter update importance hints ⬅
6. Layer-by-layer pattern of deep learning

...can exploit to run orders of magnitude faster!

Bosen: Managed Communication

- **Combine SSP's lazy transmission of parameter updates with:**
  - early transmission of larger parameter changes

  (Idea: larger change likely to be an important update)
  - up to bandwidth limit & staleness limit



LDA Topic Modeling
Nytimes dataset
16x8 cores

[SoCC'15]

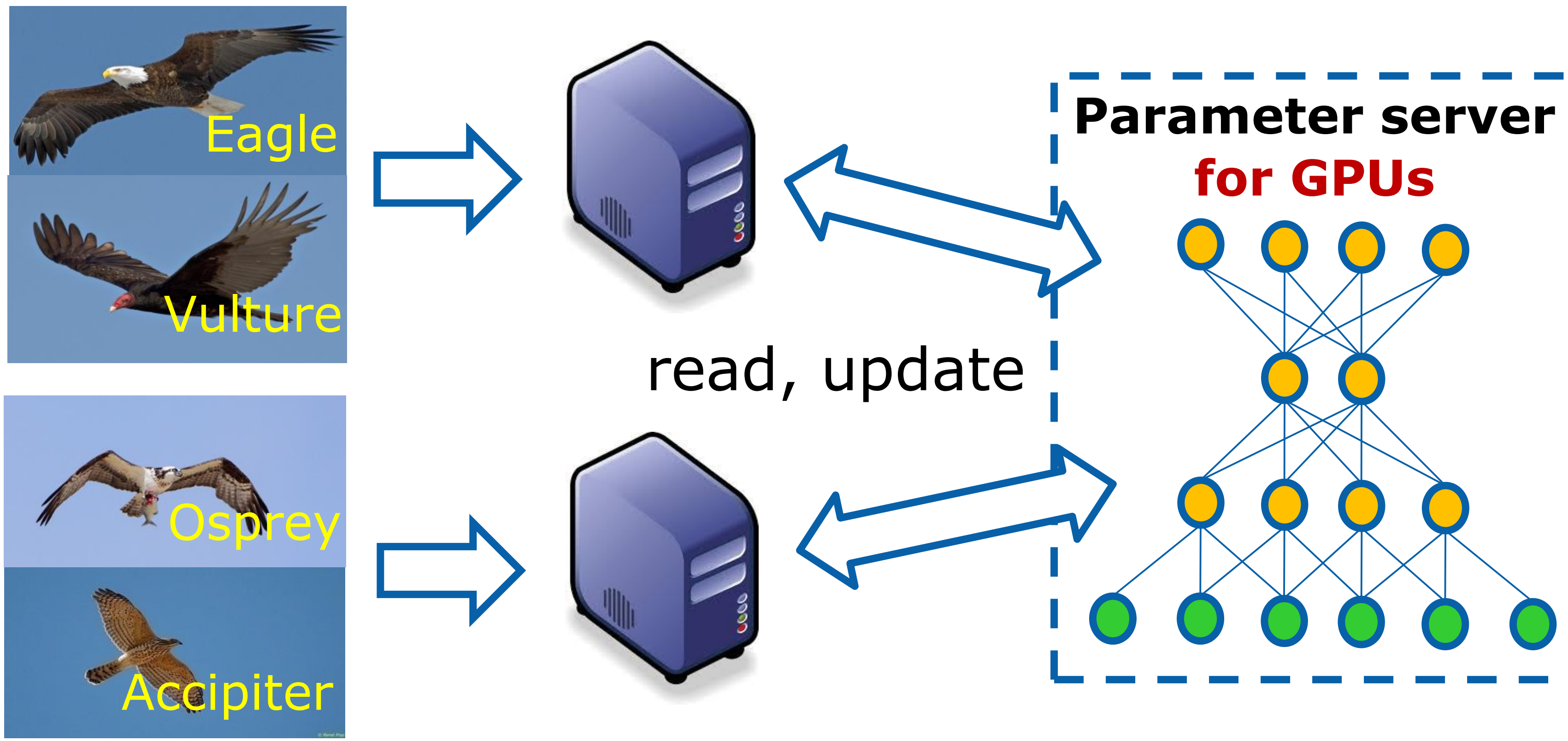
What's So Special about Big Learning? …A Distributed Systems Perspective

The Good News

1. Commutative/Associative parameter updates
2. Tolerance for lazy consistency of parameters
3. Repeated parameter data access pattern
4. Intra-iteration progress measure
5. Parameter update importance hints
6. Layer-by-layer pattern of deep learning ⬅

…can exploit to run orders of magnitude faster!

Distributed Deep Learning

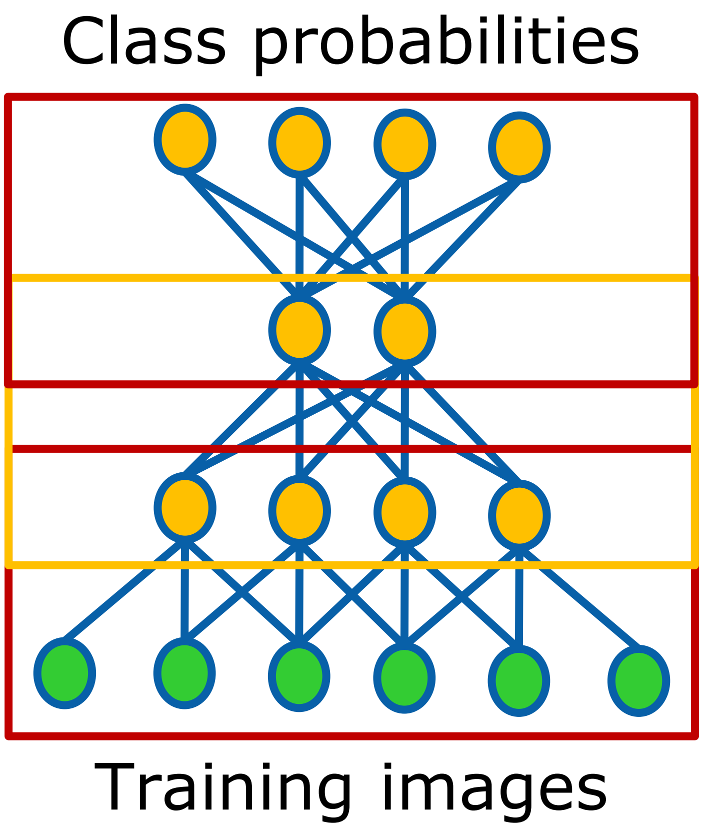


Partitioned training data

Distributed ML workers

Shared model parameters

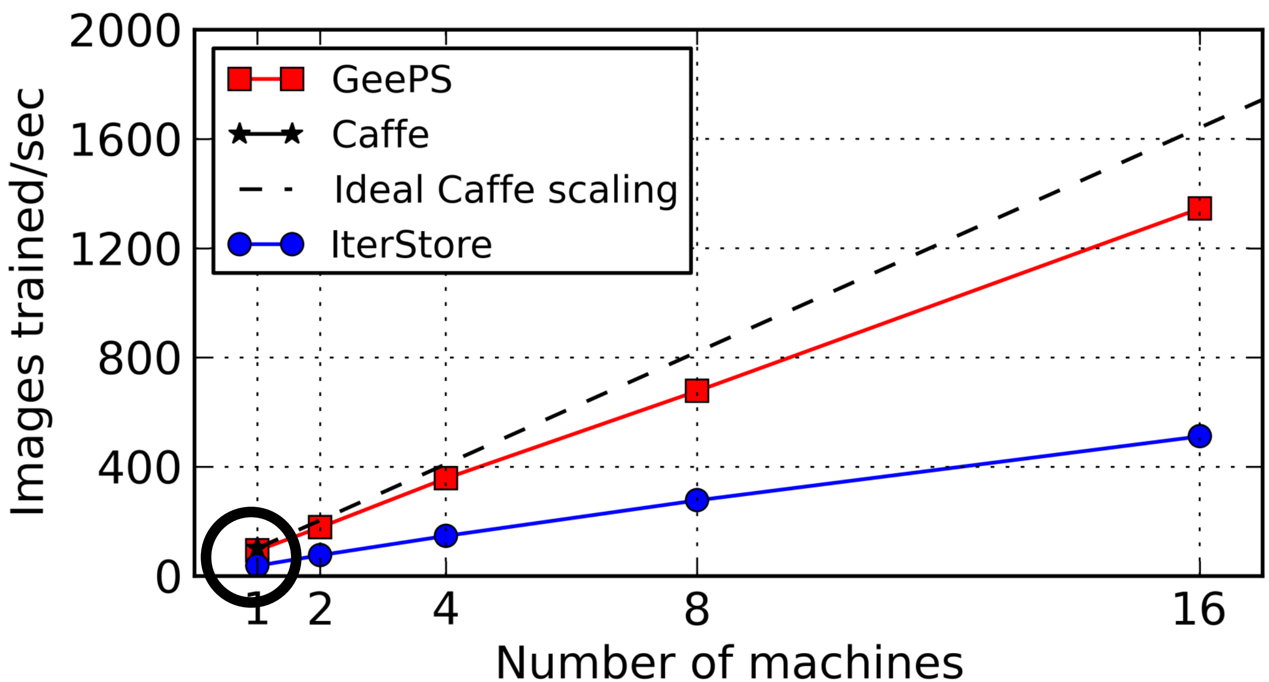
Layer-by-Layer Pattern of DNN



- For each iteration (mini-batch)
  - A forward pass
  - Then a backward pass
- Pairs of layers used at a time

GeePS: Parameter Server for GPUs

- **Careful management of GPU & CPU memory**
  - Use GPU memory as cache to hold pairs of layers
  - Stage remaining data in larger CPU memory



ImageNet22K
Adam model

[EuroSys'16]

GeePS is 13x faster than Caffe (1 GPU) on 16 machines,
2.6x faster than IterStore (CPU parameter server)

What’s So Special about Big Learning? …A Distributed Systems’ Perspective

The Good News

1. Commutative/Associative parameter updates
2. Tolerance for lazy consistency of parameters
3. Repeated parameter data access pattern
4. Intra-iteration progress measure
5. Parameter update importance hints
6. Layer-by-layer pattern of deep learning

…can exploit to run orders of magnitude faster!

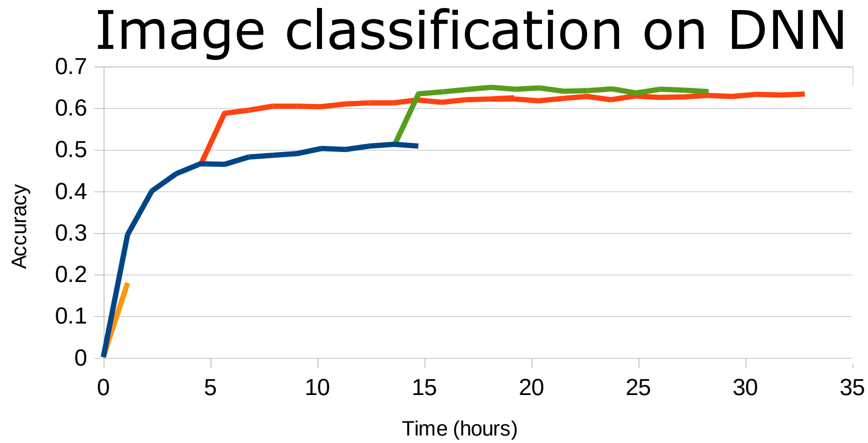
What's So Special about Big Learning? …A Distributed Systems' Perspective

<u>**More Bad News**</u>

- **Sensitivity to tunables**
- **Costly: can we use spot instances?**
- **Geo-distributed data (with skew)**

Sensitivity to Tunables

- **Many tunables in ML algorithms:**
  - Coefficients in optimization function,
    e.g., weights on regularization terms
  - Configuration tunables in optimization algorithm,
    e.g., learning rate, mini-batch size, staleness
- **Quality of solution & rate of convergence are highly sensitive to these tunables**
  - Today, mostly human trial-and-error



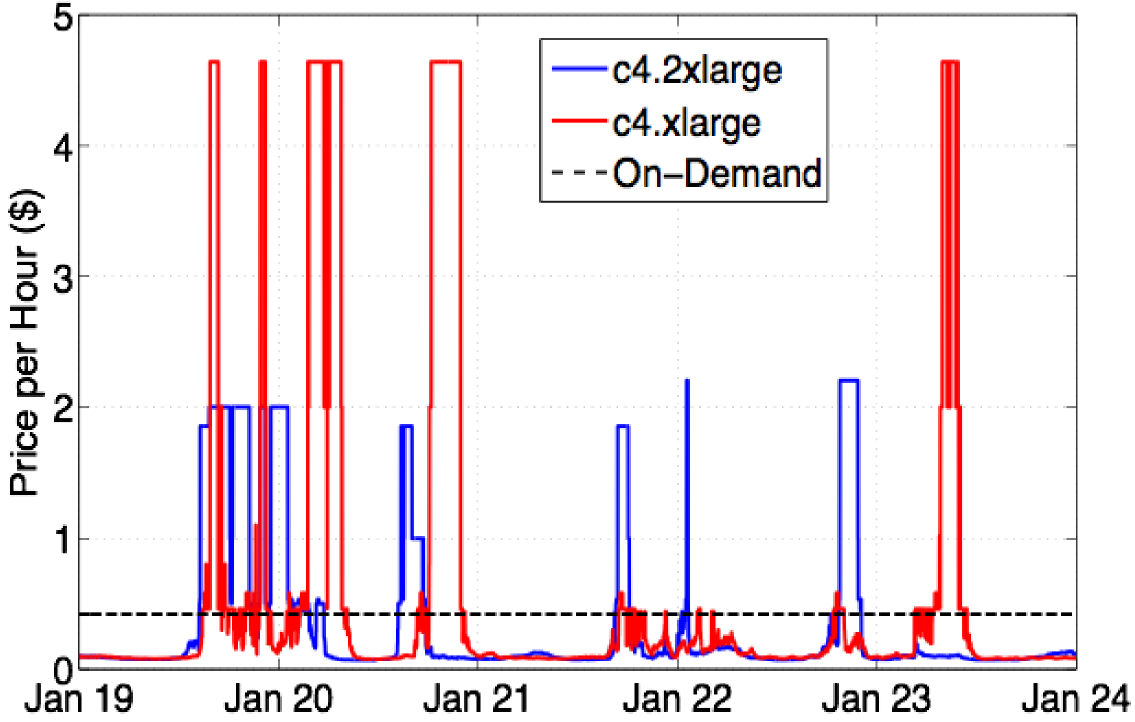
Orange: lr = 0.001
Blue: lr = 0.01
Red: lr = 0.001
Green: lr = 0.001

Ongoing Research: How to automate?

[submitted]

Costly => Use Spot Instances?

- **Spot Instances are often 85%-90% cheaper, but can be taken away at short notice**



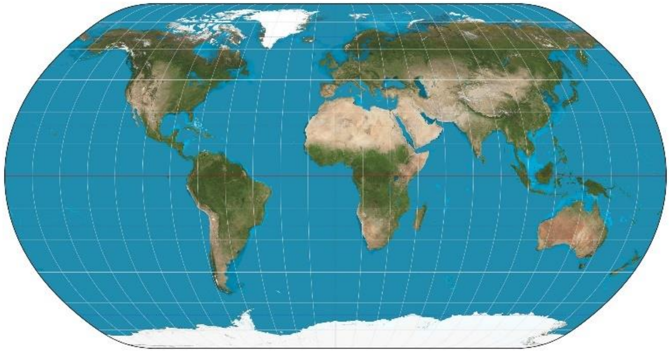
Each machine class is a bidding market

[submitted]

Ongoing Research: Effective, elastic, "Spot Dancing" Big Learning

Geo-Distributed Data (with Skew)

- **Data sources are everywhere (geo-distributed)**
  - Too expensive (or not permitted) to ship all data to single data center
- **Big Learning over geo-distributed data**
  - **Low Bandwidth & High Latency** of Inter-data-center communication relative to Intra-data-center
  - Geo-distributed data may be **highly skewed**
  - **Regional answers** also of interest



Ongoing Research: Effective Big Learning systems for Geo-distributed data

[NSDI'17]

What's So Special about Big Learning? …A Distributed Systems' Perspective

The Bad News: Model Training is SLOW

- **Lots of Computation / Memory**
  - Many iterations over Big Data
  - Big Models

  => Need to distribute computation widely
- **Lots of Communication / Synchronization**
  - Not readily "partitionable"

More Bad News:

Sensitivity to tunables
Costly=>spot instances?
Geo-distributed data (with skew)

What's So Special about Big Learning? …A Distributed Systems' Perspective

The Good News

- Commutative/Associative parameter updates
- Tolerance for lazy consistency of parameters
- Repeated parameter data access pattern
- Intra-iteration progress measure
- Parameter update importance hints
- Layer-by-layer pattern of deep learning
- Others to be discovered ⬅

…can exploit to run orders of magnitude faster!

Thanks to Collaborators & Sponsors

- **CMU Faculty:** **Greg Ganger**, **Garth Gibson**, **Eric Xing**
- **CMU/ex-CMU Students:** James Cipar, **Henggang Cui**, Wei Dai, Jesse Haber-Kucharsky, **Aaron Harlap**, **Qirong Ho**, **Kevin Hsieh**, Jin Kyu Kim, Dimitris Konomis, Abhimanu Kumar, Seunghak Lee, Aurick Qiao, Alexey Tumanov, Nandita Vijaykumar, **Jinliang Wei**, Lianghong Xu, Hao Zhang (**Bold**=first author)

  (Many of these slides adapted from slides by the students)

- **Sponsors:**
  - **Intel** (via ISTC for Cloud Computing & new ISTC for Visual Cloud Systems)
  - **PDL Consortium:** Avago, Citadel, EMC, Facebook, Google, Hewlett-Packard Labs, Hitachi, Intel, Microsoft Research, MongoDB, NetApp, Oracle, Samsung, Seagate, Symantec, Two Sigma, Western Digital
  - **National Science Foundation**

References

(in order of first appearance)

[Zaharia et al, NSDI'12] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. Usenix NSDI, 2012.

[Li et al, OSDI'14] M. Li, D. G. Anderson, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. Usenix OSDI, 2014.

[Power & Li, OSDI'10] R. Power and J. Li. Piccolo: Building Fast, Distributed Programs with Partitioned Tables. Usenix OSDI, 2010.

[Ahmed et al, WSDM'12] A. Ahmed, M. Aly, J. Gonzalez, S. M. Narayanamurthy, and A. J. Smola. Scalable inference in latent variable models. ACM WSDM, 2012.

[NIPS'13] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. Gibson, G. Ganger, and E. Xing. More effective distributed ML via a state synchronous parallel parameter server. NIPS, 2013.

[Petuum] petuum.org

[MXNet] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv:1512.01274, 2015.

[TensorFlow] tensorflow.org

[ATC'14] H. Cui, J. Cipar, Q. Ho, J. K. Kim, S. Lee, A. Kumar, J. Wei, W. Dai, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing. Exploiting Bounded Staleness to Speed Up Big Data Analytics. Usenix ATC, 2014.

[SoCC'14] H. Cui, A. Tumanov, J. Wei, L. Xu, W. Dai, J. Haber-Kucharsky, Q. Ho, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing. Exploiting iterative-ness for parallel ML computations. ACM SoCC, 2014.

[SoCC'16] A. Harlap, H. Cui, W. Dai, J. Wei, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing. Addressing the straggler problem for iterative convergent parallel ML. ACM SoCC, 2016.

References (cont.)

[SoCC'15] J. Wei, W. Dai, A. Qiao, Q. Ho, H. Cui, G. R. Ganger, P. B. Gibbons, G. A. Gibson, and E. P. Xing. Managed Communication and Consistency for Fast Data-Parallel Iterative Analytics. ACM SoCC, 2015.

[EuroSys'16] H. Cui, H. Zhang, G. R. Ganger, P. B. Gibbons, E. P. Xing. GeePS: Scalable deep learning on distributed GPUs with a GPU-specialized parameter server. EuroSys, 2016.

[NSDI'17] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. NSDI, 2017.