SAILING LAB

Laboratory for Statistical Artificial InteLligence & INtegrative Genomics

Petuum: A New Platform for Distributed Machine Learning on Big Data

Eric Xing

epxing@cs.cmu.edu

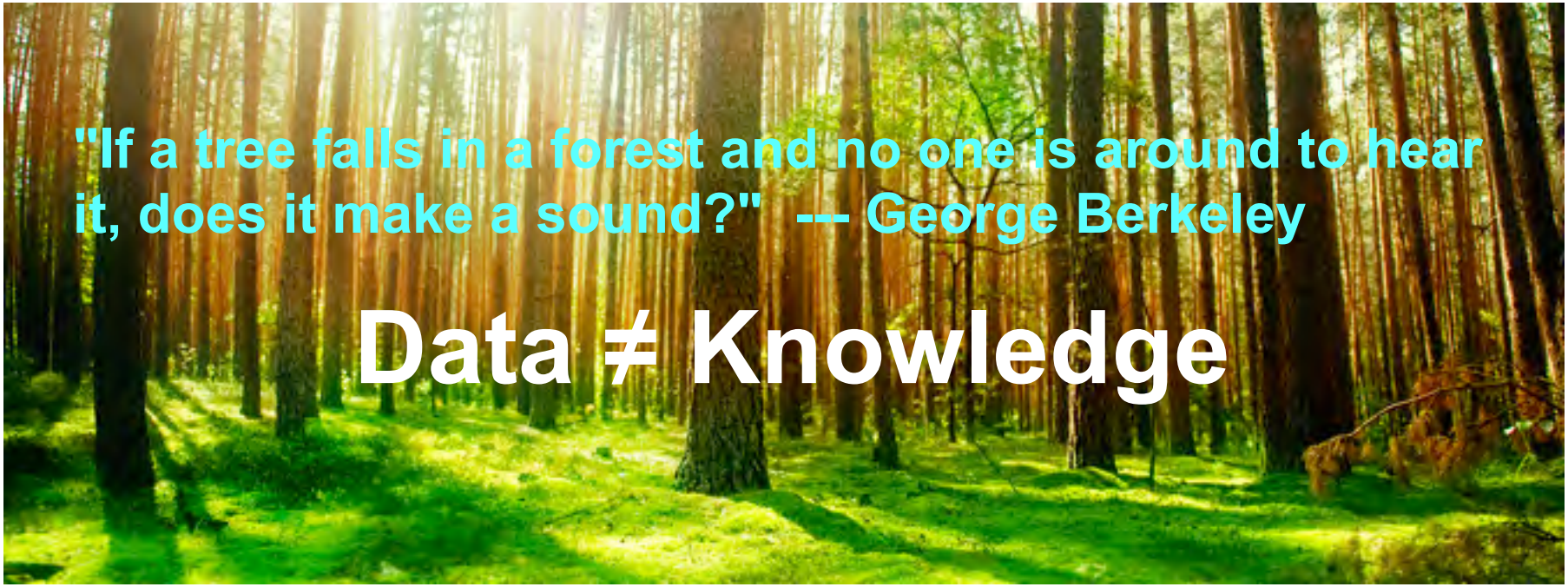
School of Computer Science

Carnegie Mellon University

Acknowledgement:

Wei Dai, Qirong Ho, Jin Kyu Kim, Abhimanu Kumar, Seunghak Lee, Jinliang Wei, Pengtao Xie, Xun Zheng
Zhiting Hu, Hao Zhang, James Cipar, Henggang Cui,
and, Phil Gibbons, Greg Ganger, Garth Gibson

Trees Falling in the Forest



- Nobody knows what's in data unless it has been processed and analyzed
  - Need a scalable way to automatically search, digest, index, and understand contents

Massive Data

facebook

1B+ USERS
30+ PETABYTES

WIKIPEDIA
The Free Encyclopedia

32 million
pages

YouTube

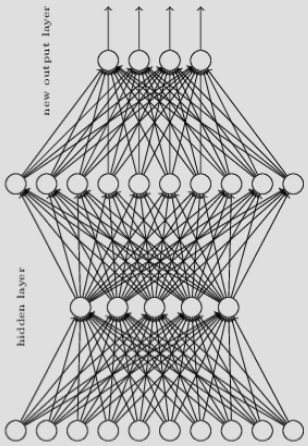
100+ hours video
uploaded every minute

twitter

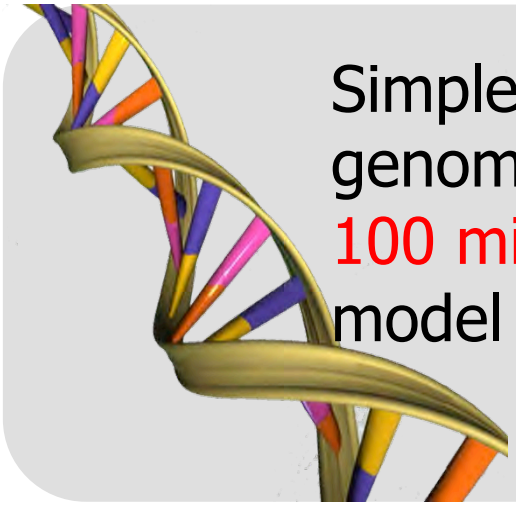
645 million users
500 million tweets / day

SAILING LAB
Laboratory for Statistical Artificial InteLligence & INtegrative Genomics

Growing Model Complexity



Google Brain Deep Learning for images:
10 Billion
model parameters



Simplest whole-genome analysis:
100 million
model parameters

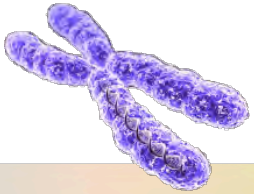
News article analysis:
1 Trillion
model parameters

NETFLIX

Video recommendation:
10 Billion
model parameters

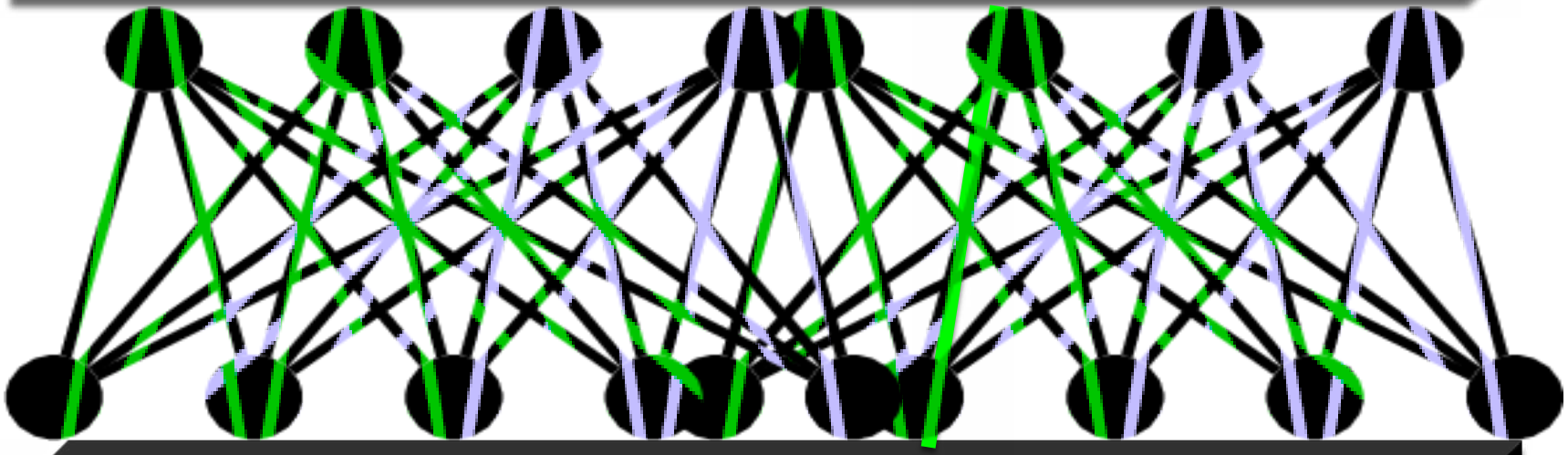
Solution:





Machine Learning Models/Algorithms

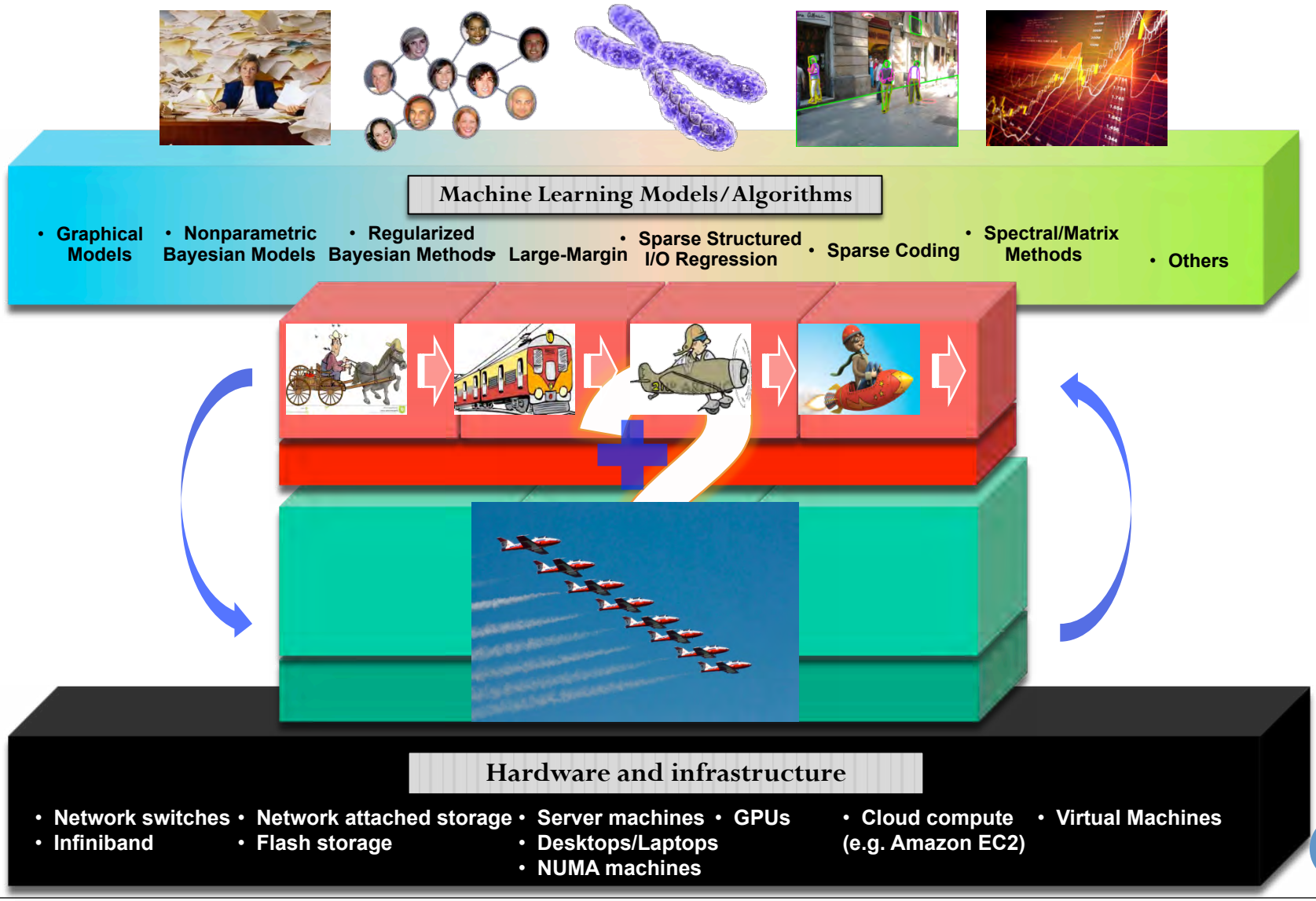
- Graphical Models
- Nonparametric Bayesian Models
- Regularized Bayesian Methods
- Large-Margin
- Sparse Structured I/O Regression
- Deep Learning
- Spectral/Matrix Methods
- Others



Hardware and infrastructure

- Network switches
- Infiniband
- Network attached storage
- Flash storage
- Server machines
- Desktops/Laptops
- NUMA machines
- GPUs
- Cloud compute (e.g. Amazon EC2)
- Virtual Machines

Solution: An Alg/Sys INTERFACE for Big ML



What does an ML programmer need?

First-timer's ideal view of ML

```
global model = (a,b,c,...)
global data = load(file)

Update(var a):
  a = doSomething(data,model)

Main:
  do Update() on all var in
    model until converged
```

High-performance implementation

Many considerations

- What data batch size?
- How to partition model?
- When to sync up model?
- How to tune step size?
- What order to Update()?

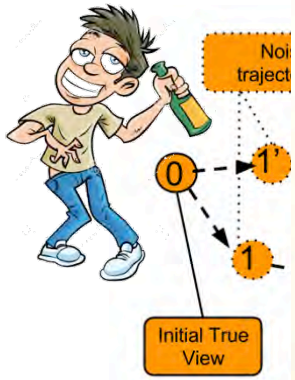
1000s of lines of extra code

Goal: system that can be programmed like "ideal view", yet yields state-of-the-art performance

SAILING LAB

Laboratory for Statistical Artificial InteLligence & INtegrative Genomics

ML Computing is not Traditional Computing



2

8

Sorting error: 2 after 5

ists and is not rected!

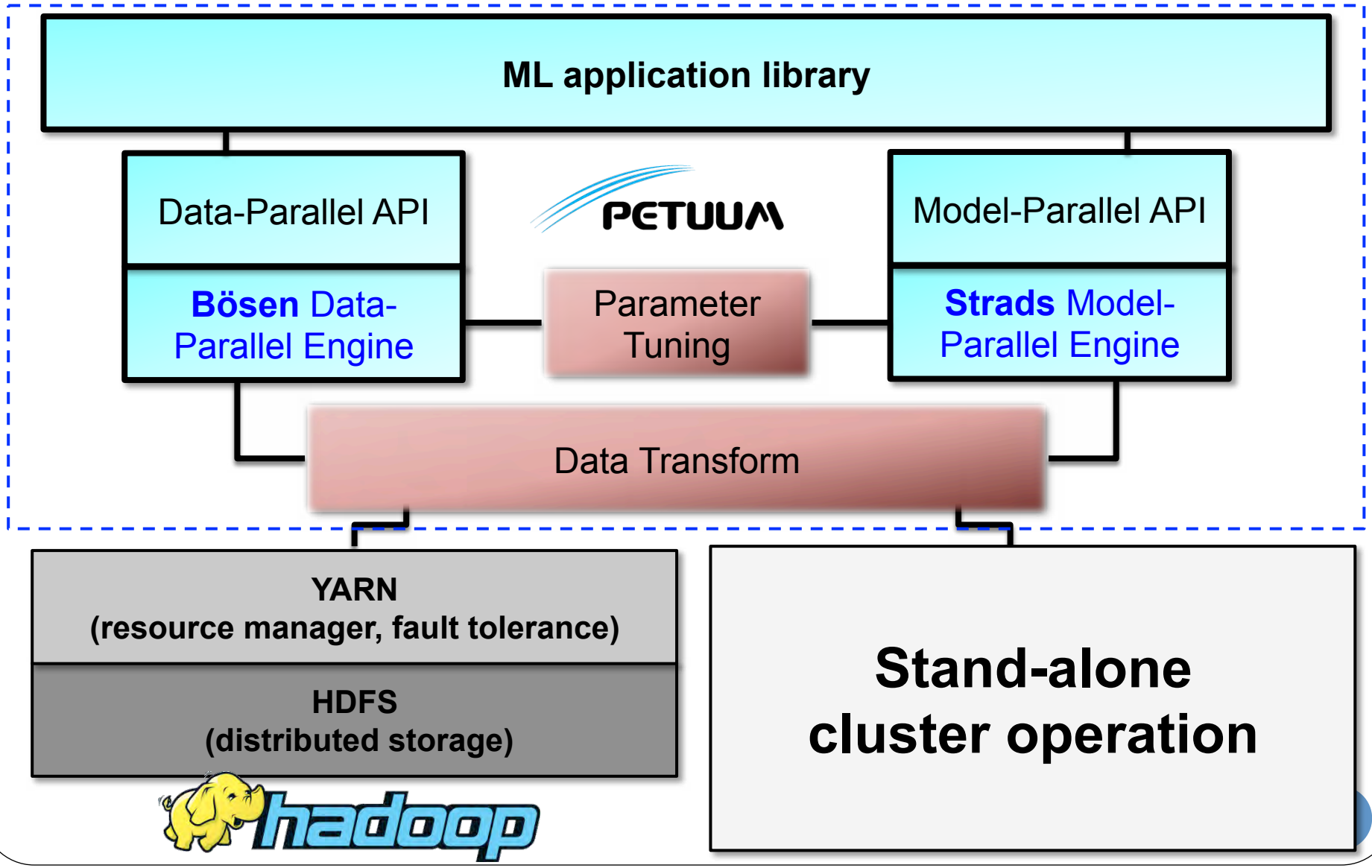
ML Comput
Self-healing

A New Framework for Large Scale Parallel Machine Learning

(Petuum.org)

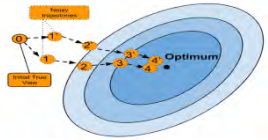
SAILING LAB
Laboratory for Statistical Artificial InteLligence & INtegrative Genomics

Petuum Architecture & Ecosystem

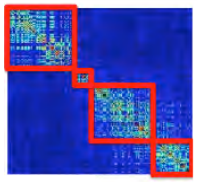


Intrinsic Properties of ML Programs

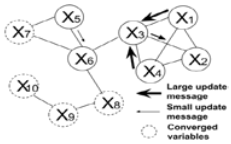
- ML is **optimization-centric**, and admits an **iterative convergent** algorithmic solution rather than a one-step closed form solution
  - **Error tolerance**: often robust against limited errors in intermediate calculations



  - **Dynamic structural dependency**: changing correlations between model parameters critical to efficient parallelization



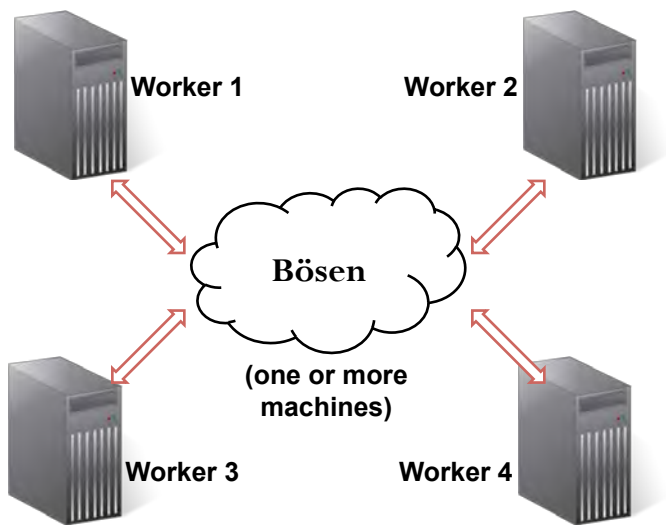
  - **Non-uniform convergence**: parameters can converge in very different number of steps



- Whereas traditional programs are **transaction-centric**, thus only guaranteed by **atomic correctness** at every step

Data-Parallel Engine

- Bösen: a bounded-asynchronous distributed key-value store
  - Data-parallel programming via distributed shared memory (DSM) abstraction
  - Managed communication for better parallel efficiency & guaranteed convergence

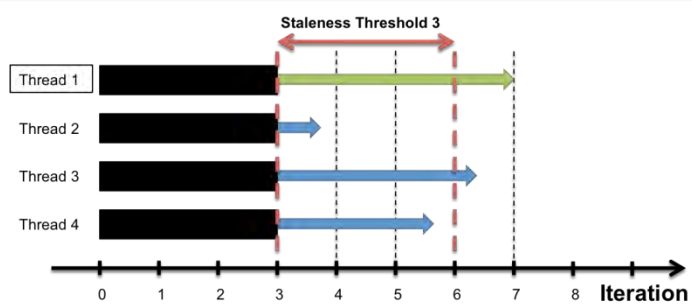


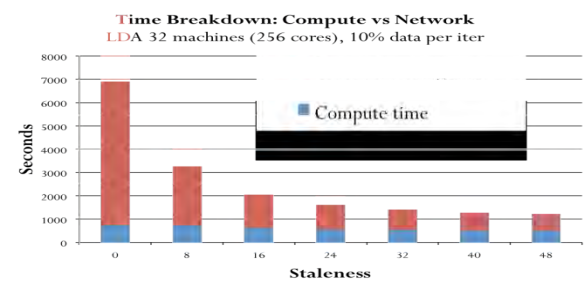
Single Machine Parallel

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

Distributed with Bösen

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





Data-Parallel Stochastic Gradient Descent

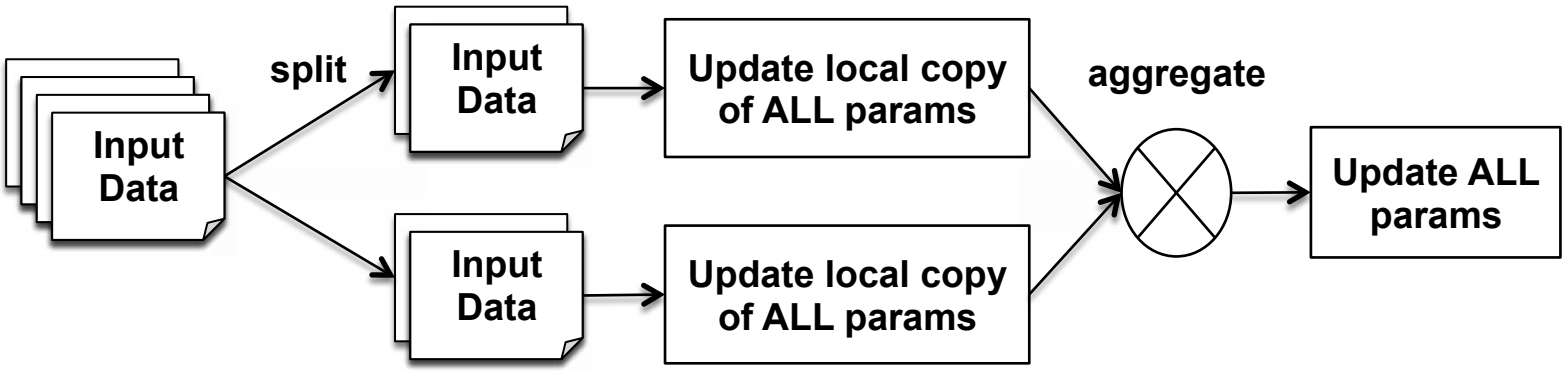
- Consider:

$$\min_x \mathbb{E}\{f(x,d)\}$$

- SPG:

$$x^{(t+1)} \leftarrow x^{(t)} - \gamma \nabla_x f(x^{(t)}, d_i)$$

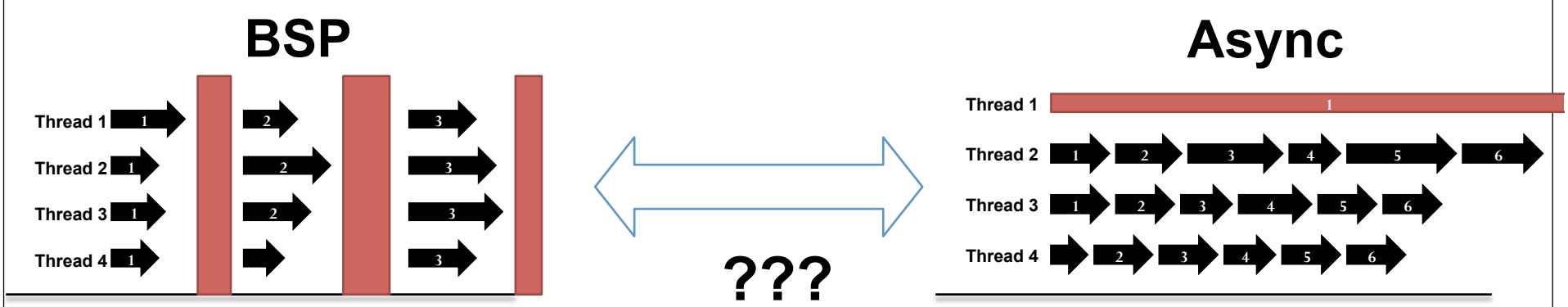
- Parallel SGD [Zinkevich et al., 2010]: Partition data to different workers; all workers update full parameter vector



- PSGD runs SGD on local copy of params in each machine

How to speed up Data-Parallelism?

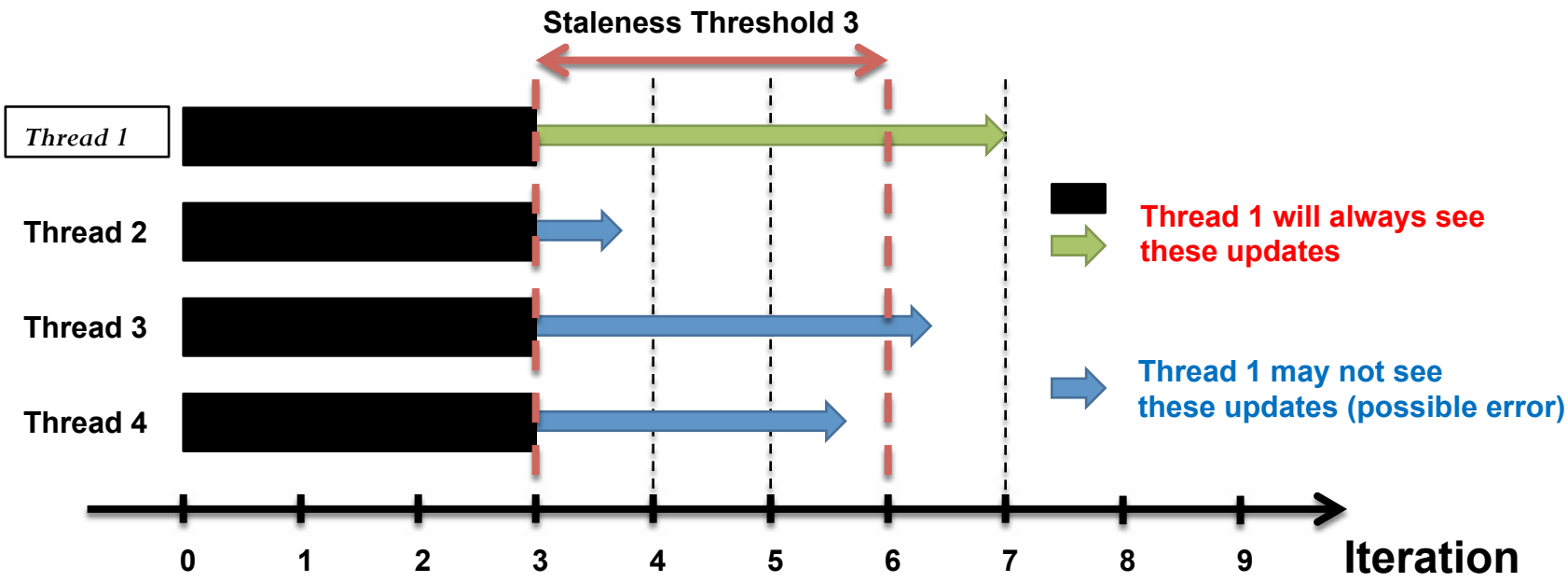
- **Existing ways are either safe/slow (BSP), or fast/risky (Async)**
- **Need "Partial" synchronicity: Bounded Async Parallelism (BAP)**
  - Spread network comms evenly (don't sync unless needed)
  - Threads usually shouldn't wait – but mustn't drift too far apart!
- **Need straggler tolerance**
  - Slow threads must somehow catch up



Is persistent memory really necessary for ML?

High-Performance Consistency Models for Fast Data-Parallelism

*Q. Ho, J. Cipar, H. Cui, J.-K. Kim, S. Lee, P. B. Gibbons, G. Gibson, G. R. Ganger and E. P. Xing. **More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server.** NIPS 2013.*



Stale Synchronous Parallel (SSP)

- Allow threads to run at their own pace, without synchronization
- Fastest/slowest threads not allowed to drift >S iterations apart
- Threads cache local (stale) versions of the parameters, to reduce network syncing

Consequence:

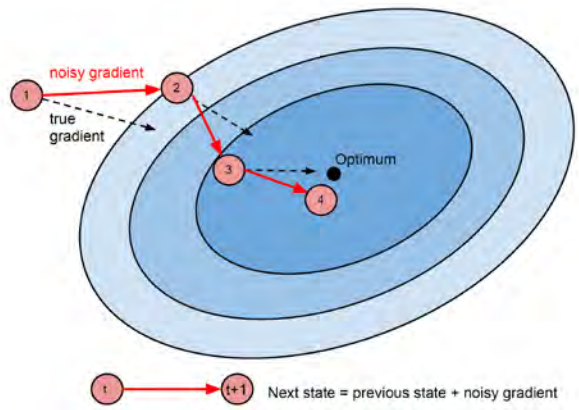
- Asynchronous-like speed, BSP-like ML correctness guarantees
- Guaranteed age bound (staleness) on reads
- Contrast: no-age-guarantee Eventual Consistency seen in Cassandra, Memcached

Convergence Theorem

*W. Dai, A. Kumar, J. Wei. Q. Ho, G. Gibson and E. P. Xing, **High-Performance Distributed ML at Scale through Parameter Server Consistency Models**. AAAI 2015.*

- **Goal:** minimize convex $f(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} f_t(\mathbf{x})$

  (Ex: Stochastic Gradient)
  - Max allowed staleness s, over P parallel workers
  - For accurate analysis, we must consider <u>real staleness observed by system</u>
    - average μ_{stale}, variance σ_{stale}



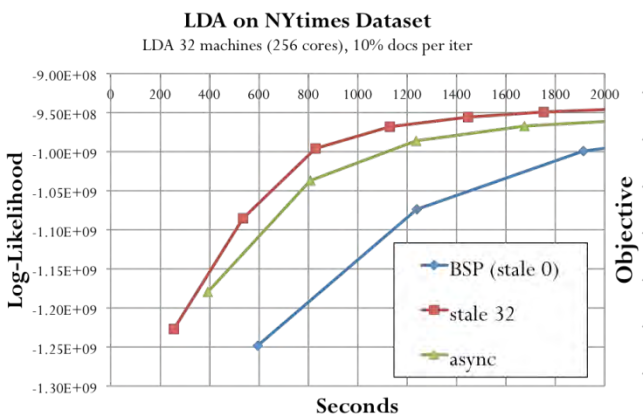
- **SSP converges according to**
  - Where T is # of iterations, τ is desired answer quality

$$\mathbb{P}\left[\underbrace{\frac{R[X]}{T}}_{\text{Gap between current and best answer}} \geq \tau + \underbrace{\frac{\mathcal{O}(\mu_{stale})}{\sqrt{T}}}_{\text{Negative effect from high avg. staleness } u_{stale}}\right] \leq \exp\left\{-\frac{T\tau^2}{\underbrace{o(T)\sigma_{stale} + \mathcal{O}(s\tau)}_{\text{Negative effect from max staleness } s\text{; staleness var. } \sigma_{stale}}}\right\}$$

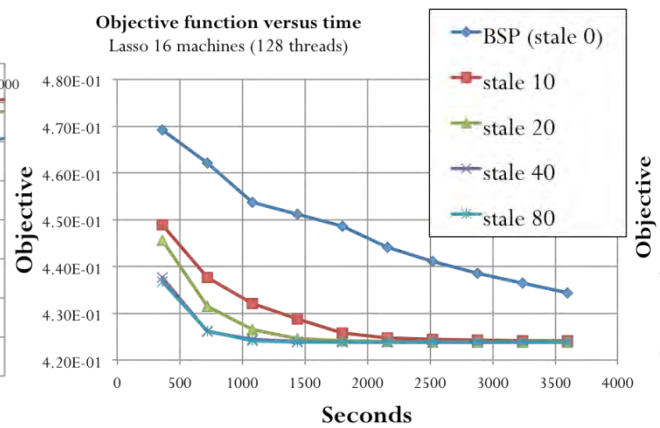
- Take-away: faster and more efficient ML requires SSP, which limits the staleness max s, and minimizes staleness mean μ_{stale}, variance σ_{stale}

Bösen advantages:

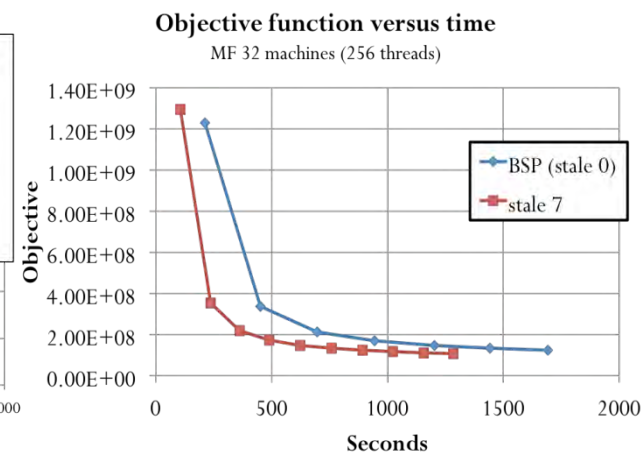
- Enjoys **async speed**, but **BSP guarantee** across algorithms



LDA



LASSO

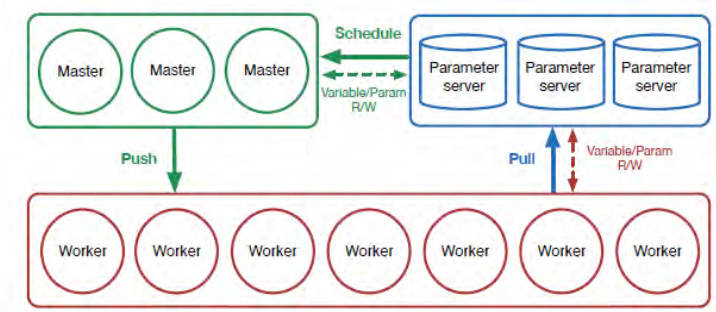


Matrix Fac.

- **Light network traffic:** Low-rank (sufficient factor) pre-updates
- **Auto-tuning:** for both parameter & step-size
- Unified data interface: support many **feature transforms**

Model-Parallel Engine

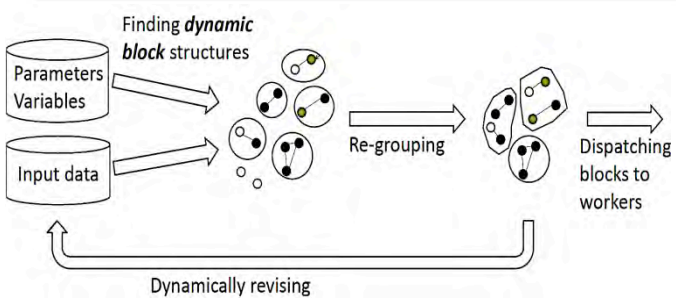
- Strads: a structure-aware load-balancer and task prioritizer
  - Model-parallel programming via a scheduler interface
  - Explore structural dependencies and non-uniform convergence within ML models for best execution order

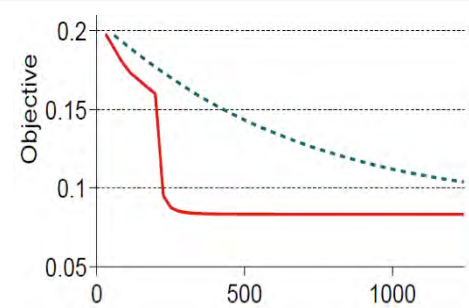



```
schedule() {
  // Select U vars x[j] to be sent
  // to the workers for updating
  ...
  return (x[j_1], ..., x[j_U])
}
```

```
push(worker = p, vars = (x[j_1],...,x[j_U])) {
  // Compute partial update z for U vars x[j]
  // at worker p
  ...
  return z
}
```

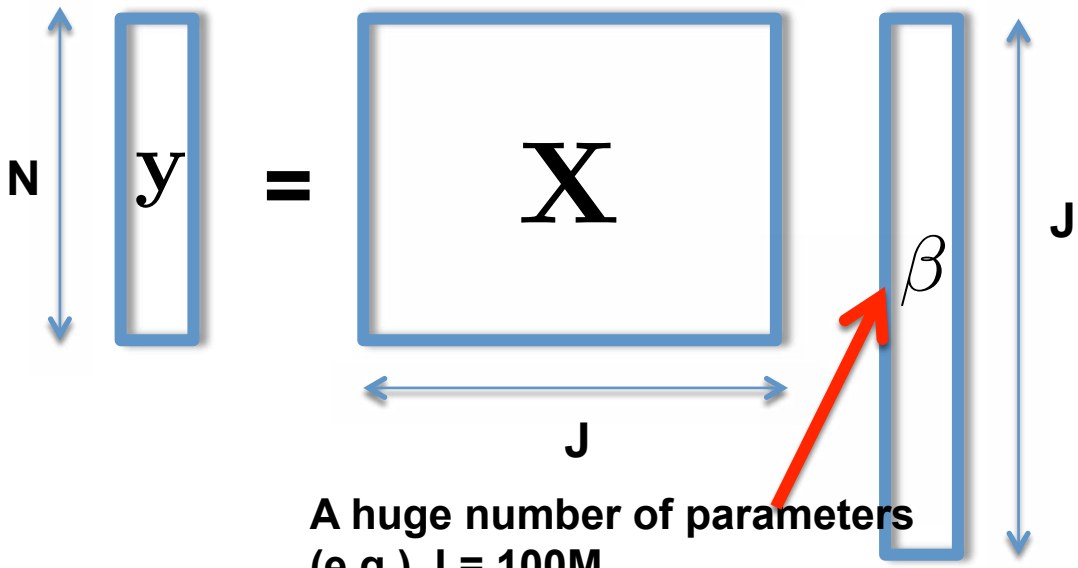
```
pull(workers = [p], vars = (x[j_1],...,x[j_U]),
     updates = [z]) {
  // Use partial updates z from workers p to
  // update U vars x[j]. sync() is automatic.
  ...
}
```





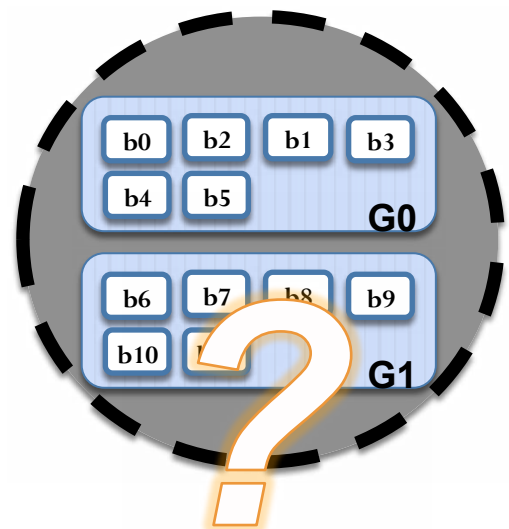
Challenges in Model Parallelism

$$\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_j |\beta_j|$$

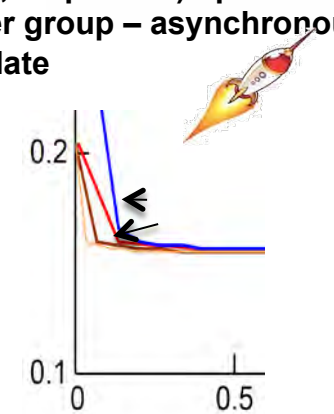


A huge number of parameters
(e.g.) J = 100M

$$\boxed{\beta_1^{(t)} \leftarrow S(\mathbf{x}_1^T\mathbf{y} - \mathbf{x}_1^T\mathbf{x}_2\beta_2^{(t-1)}, \lambda)}$$



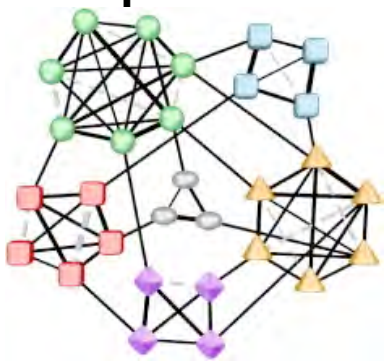
- Within group – synchronous (i.e., sequential) update
- Inter group – asynchronous update



How to Model-Parallel?

- **Again, existing ways are either safe but slow, or fast but risky**
- **Need to avoid processing whole-data just for optimal distribution**
  - i.e., build expensive data representation on the whole data
  - Compute all variable dependencies
- **Dynamic load balance**

Graph Partition



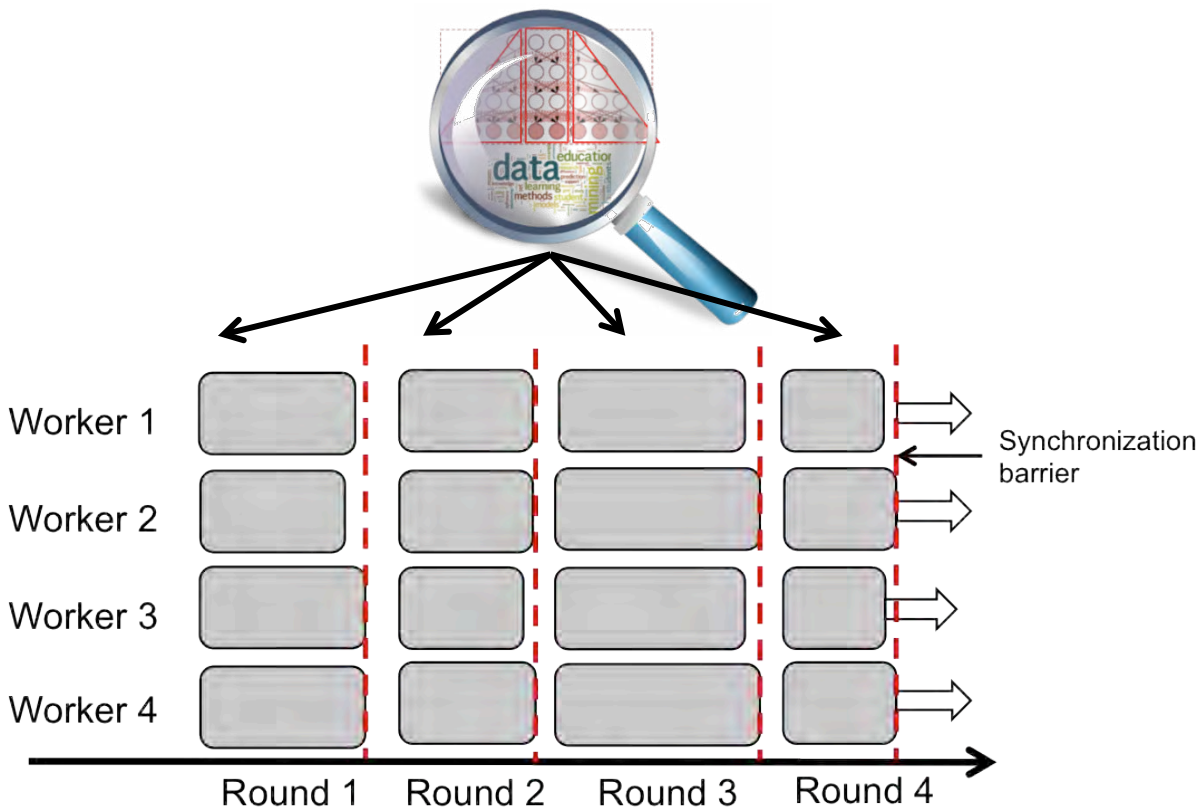
???

Random Partition

GraphLab

Is correct global dependency really necessary for ML?

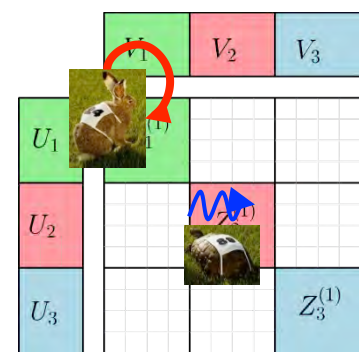
Structure-Aware Parallelization (SAP)



- **Priority Scheduling**

$$\{\beta_j\} \sim \left(\delta\beta_j^{(t-1)}\right)^2 + \eta$$

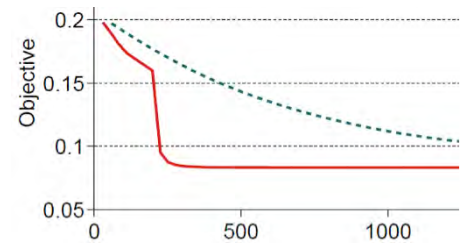
- **Block scheduling**



- ❑ **Smart model-parallel execution:**
  - ❑ Structure-aware scheduling
  - ❑ Variable prioritization
  - ❑ Load-balancing
- ❑ **Simple programming:**
  - ❑ Schedule()
  - ❑ Push()
  - ❑ Pull()

SAILING LAB
Laboratory for Statistical Artificial InteLligence & INtegrative Genomics

Convergence Theorem



E. P. Xing et al., Petuum: A New Platform for Distributed Machine Learning on Big Data. KDD 2015.

- **Goal:** solve sparse regression problem $\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_j |\beta_j|$
  - Via coordinate descent over $X^{(1)}, X^{(2)}, \ldots, X^{(T)}$
    - where $X^{(t)}$ is the data columns (features) chosen for updating at iteration (t)
  - P parallel workers, M-dimensional data
  - $\rho = \max_t (X^{(t)})^T X^{(t)}$, i.e. the maximum "data difficulty" (technically, spectral radius) across all data subsets $X^{(1)}, X^{(2)}, \ldots, X^{(t)}$
- **SAP converges according to**
  - Where t is # of iterations

Gap between current and best answer

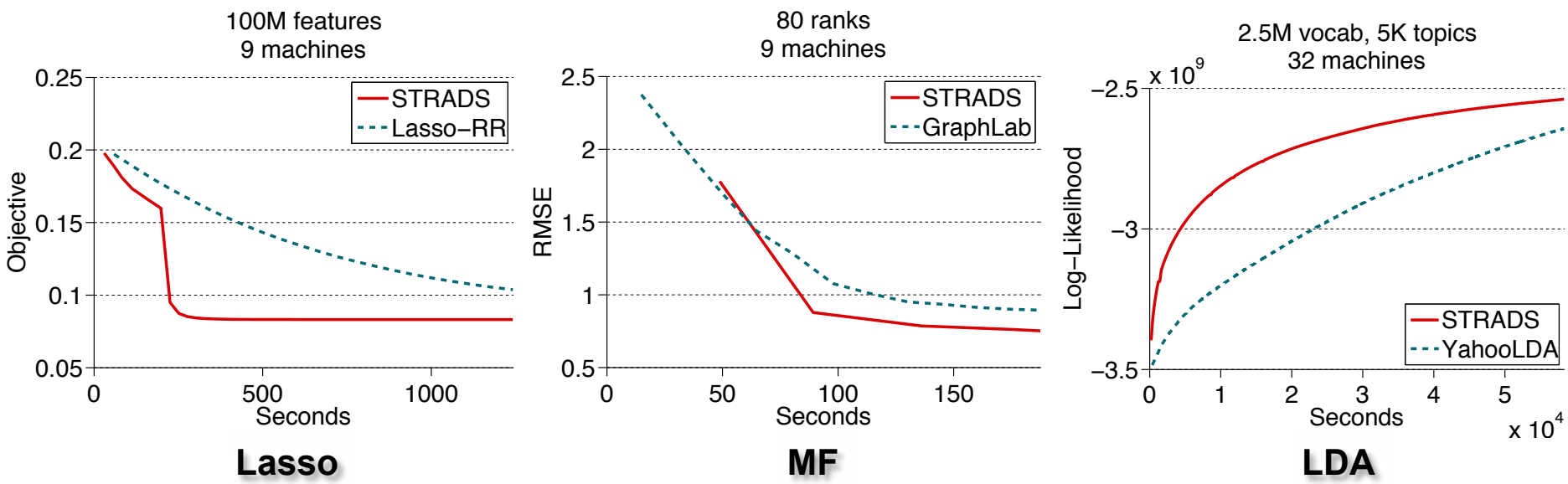
SAP scheduling minimizes ρ, ensuring close to $1/P$ convergence, i.e. near-perfect scaling with P workers

$$\mathbb{E}\left[f(X^{(t)}) - f(X^*)\right] \leq \frac{\mathcal{O}(M)}{P - \frac{\mathcal{O}(P^2\rho)}{M}} \frac{1}{t} = \mathcal{O}\left(\frac{1}{Pt}\right)$$

- Take-away: faster and more efficient ML requires SAP, which minimizes the difficulty ρ of the problem by searching for uncorrelated feature subsets $X^{(1)}, X^{(2)}, \ldots, X^{(t)}$

Strads advantages:

- Faster, **near-ideal** convergence speed across algorithms



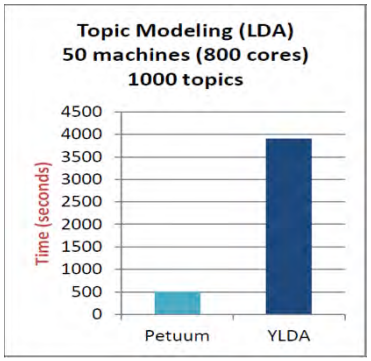
- **Structure-aware parallelization:** auto-find best update order
- **Pipelining:** overlap `schedule()` with `push/pull()` for speed
- Parameter & step-size **auto-tuning**
- Library of **feature transforms**

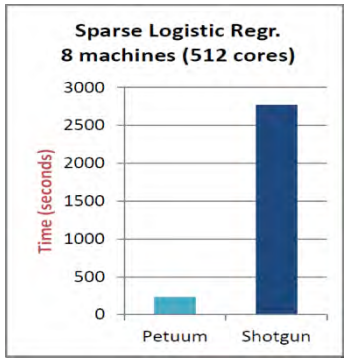
Efficiency

- Petuum automatically makes ML apps more efficient

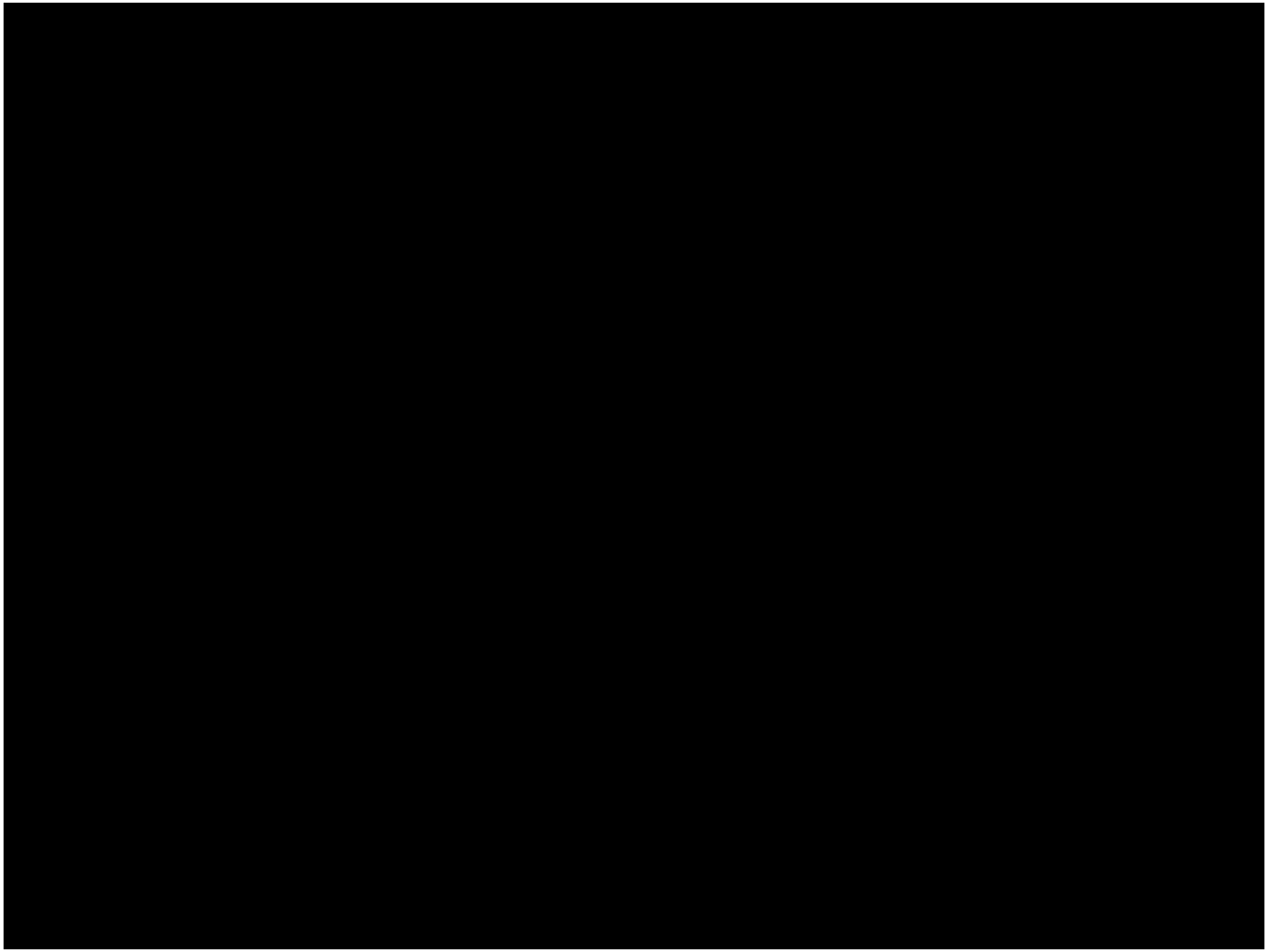
- Versus Spark MLlib v1.3, Petuum is faster by
  - 8x on Logistic Regression for **CTR and Event Prediction**
  - 100x on Topic Modeling for **User Profiling**
  - 20x on Lasso Regression for **Genetic Assay Analysis**
  - Scale: 10-100 machines, GBs-TBs of data

- Versus specialized implementations

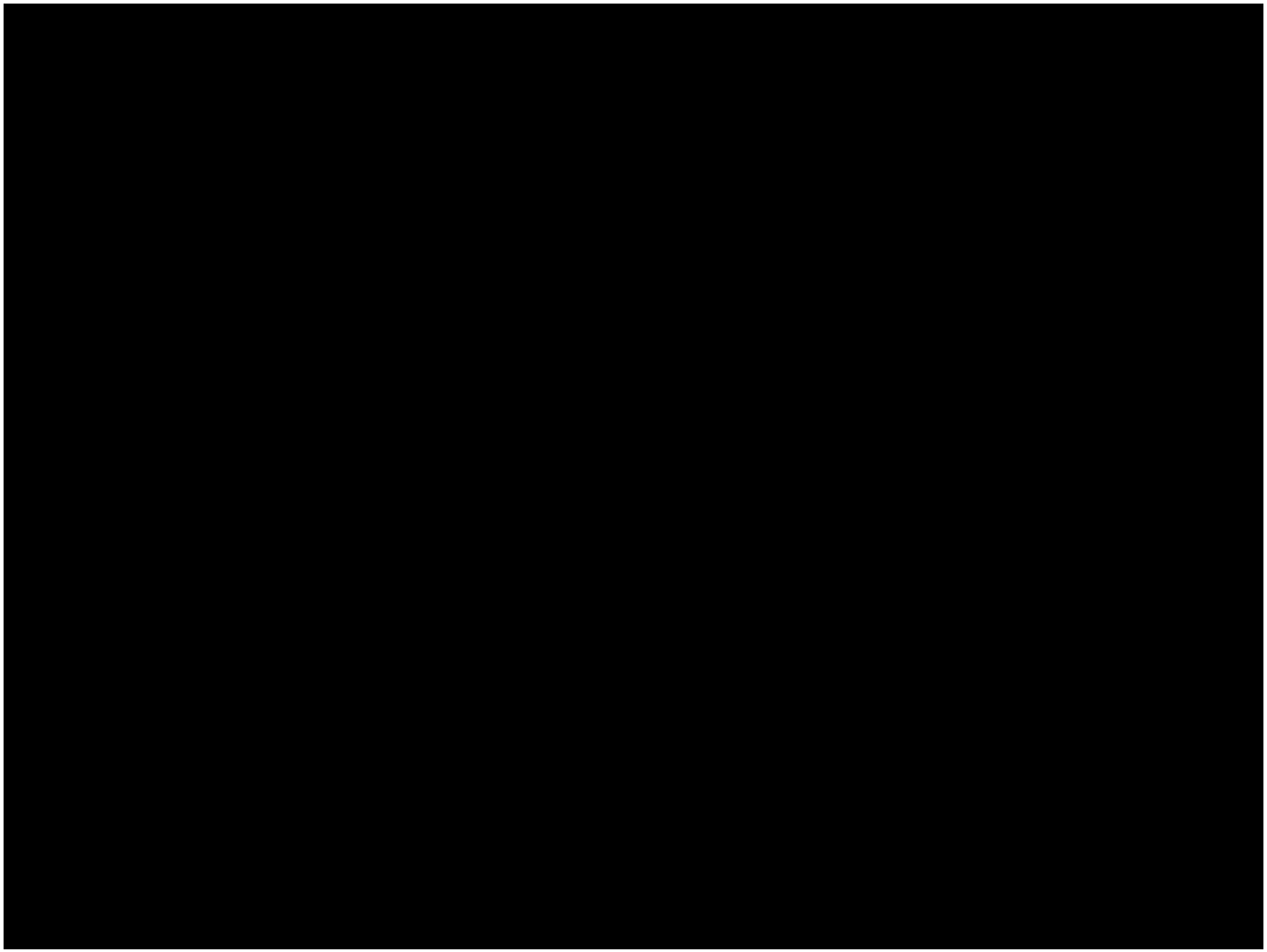




Efficiency Demo - MatrixFact



Efficiency Demo - CNN



High-Speed Model Building and Prediction

High-volume model building for real-time data streams

- **CNN Deep Learning:** ***200 images / second / GPU machine***
- **Topic Model:** ***5 million words / second / machine***
- **Multiclass LR:** ***1 million events / second / machine***
- **Random Forest:** ***500k events / second / machine***
- **Metric Learning:** ***24 million ops / second / machine***

High-speed analytics predictions

- **CNN Deep Learning:** ***350 images / second / machine***
- **Matrix Factorization:** ***1.2m recommendations / second / machine***
- **Random Forest:** ***0.5m classification / second / machine***

SAILING LAB
Laboratory for Statistical Artificial InteLligence & INtegrative Genomics

Petuum Verticals

PETUUM
petuum.org

Video Summarization with **deep learning & parallel GPU system**
1000s of cameras, millions of frames: find traffic accidents, suspicious activity

User Profiling with **supervised topic models**
Turn blogs, tweets, videos, pics, shopping => customer profile

Mobile Call Data Record (CDR) Analysis with **graph miners**
Predict device adoption, churners, influence spread in a telco setting

Click-through-rate and Event Prediction with **sparse regression**
~1B events/hour for mobile apps, e.g. streaming video, taxi services, mobile advertising

Recommender Systems with **collaborative filtering**
~100M users, ~10M products, even with ever-changing ratings, growing userbase/product catalogue

SAILING LAB
Laboratory for Statistical Artificial InteLligence & INtegrative Genomics

Acknowledgements

SAILING LAB
Laboratory for Statistical Artificial InteLligence & INtegrative Genomics


Jin Kyu Kim


Seunghak Lee


Jinliang Wei


Wei Dai


Pengtao Xie

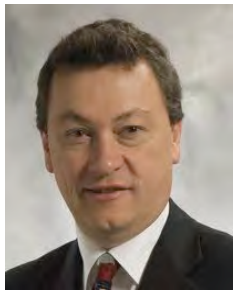

Xun Zheng


Abhimanu Kumar


Qirong Ho

www.sailing.cs.cmu.edu

$$$:


Garth Gibson

Greg Ganger


Phillip Gibbons


James Cipar