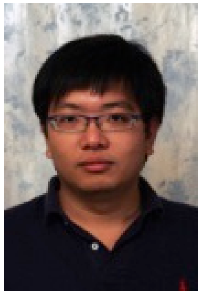
GraphLab$_2$

Machine Learning for Big Data in the Cloud

Carlos Guestrin


Joseph
Gonzalez


Yucheng
Low


Aapo
Kyrola


Haijie
Gu


Joseph
Bradley


Danny
Bickson

Needless to Say, We Need Machine Learning for Big Data

flickr

6 Billion
Flickr Photos

28 Million
Wikipedia Pages

facebook

1 Billion
Facebook Users

YouTube

72 Hours a Minute
YouTube

The New York Times

SundayReview

WORLD U.S. N.Y. / REGION BUSINESS TEC

NEWS ANALYSIS

The Age of Big Data

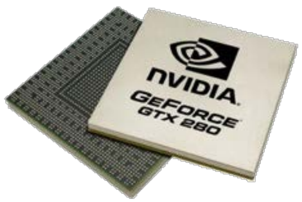
By STEVE LOHR
Published: February 11, 2012

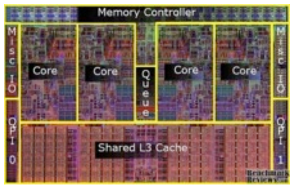
"… data a new class of economic asset, like currency or gold."

Big Learning

How will we
design and **implement**
parallel learning systems?

A Shift Towards Parallelism



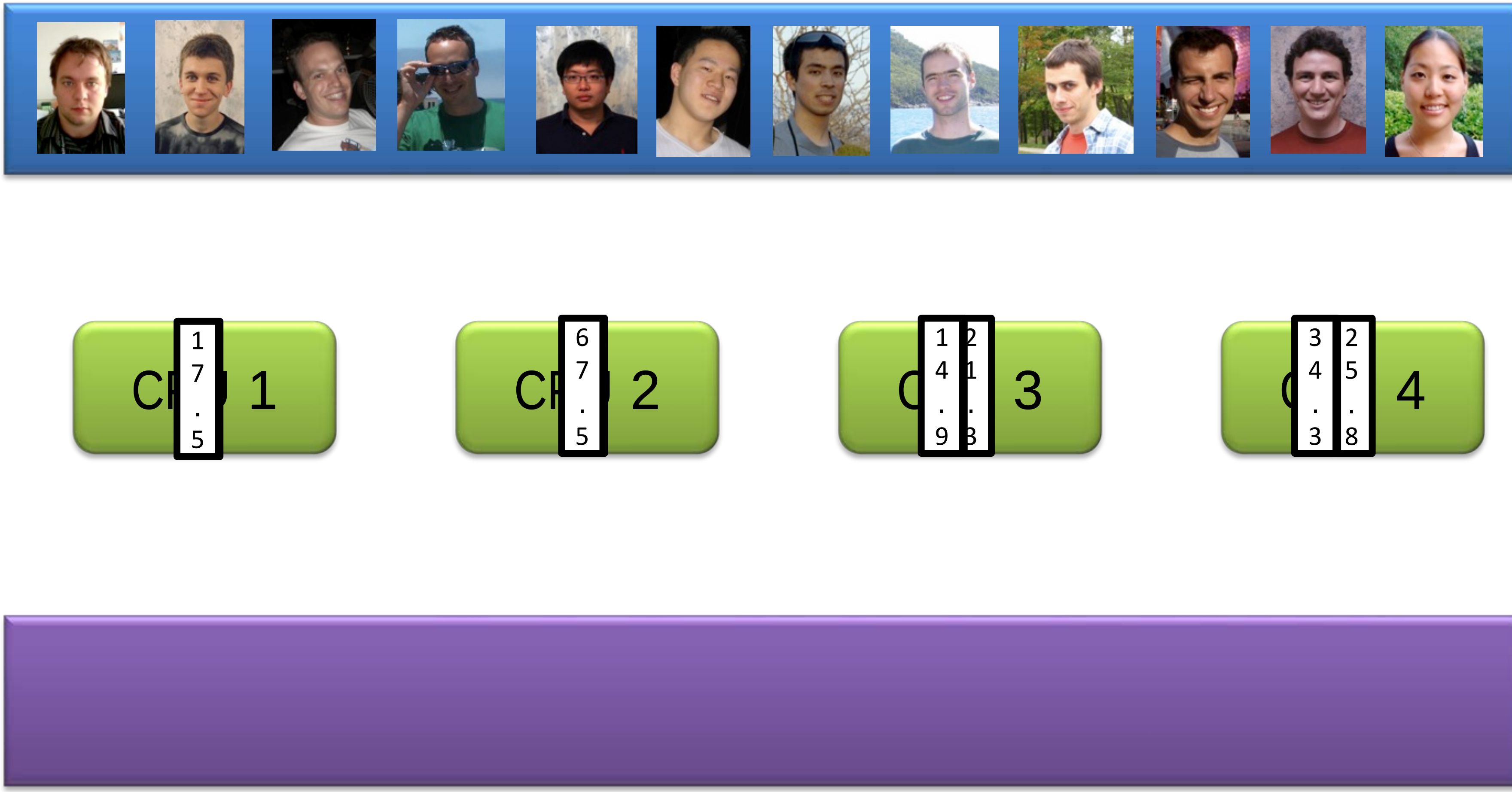


GPUs Multicore Clusters Clouds Supercomputers

- Graduate students **repeatedly** solve the same parallel design challenges:
  - Race conditions, distributed state, communication…
- The resulting code is:
  - difficult to maintain, extend, debug…

Avoid these problems by using **high-level abstractions**

Data Parallelism (MapReduce)



Solve a huge number of independent subproblems

MapReduce for Data-Parallel ML

Excellent for large data-parallel tasks!

Data-Parallel

MapReduce

Feature Extraction

Cross Validation

Computing Sufficient Statistics

Is there more to Machine Learning

?

What is this an image of?

It’s next to this…

The Power of Dependencies

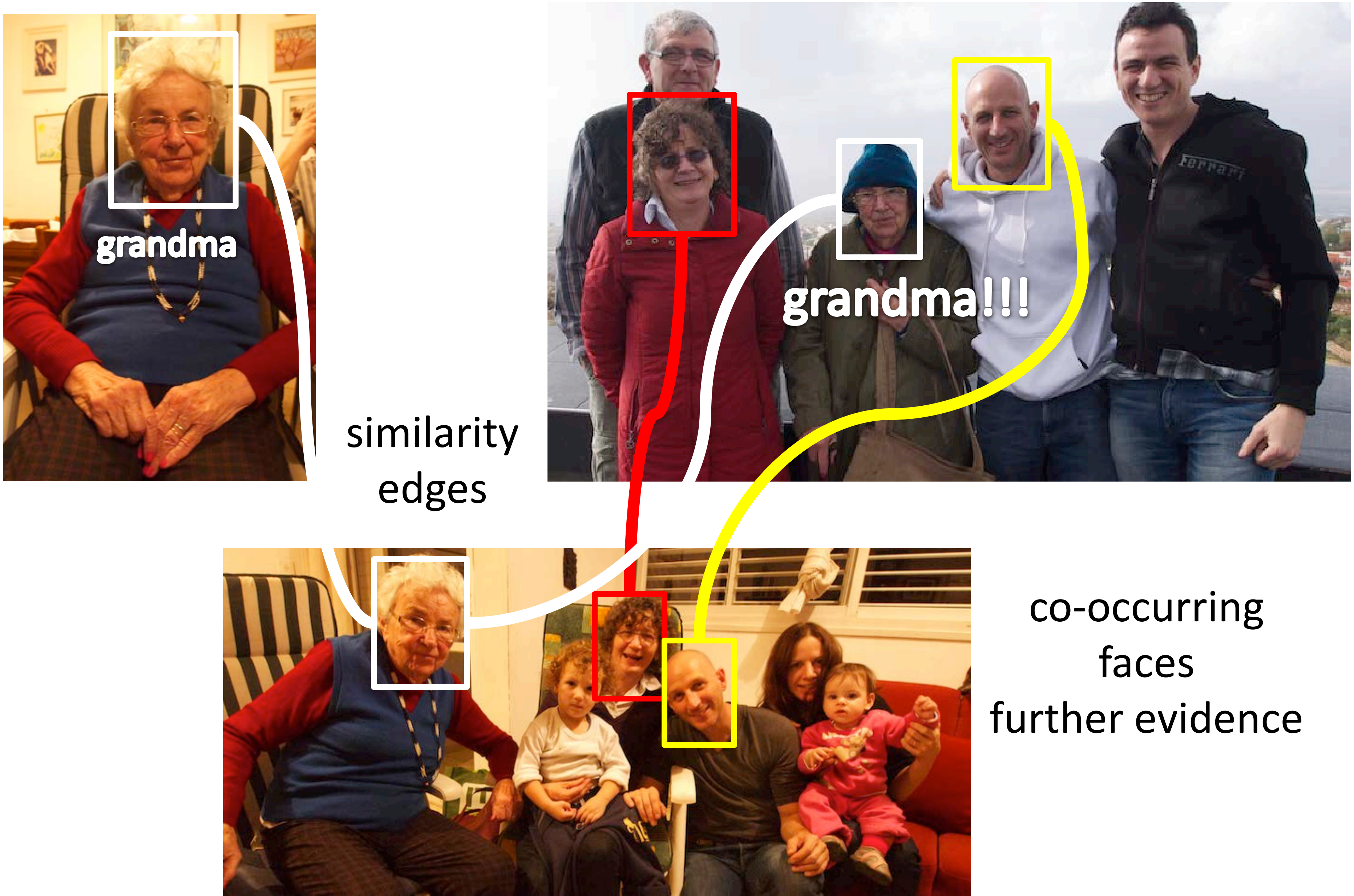
where the value is!

Examples of Graphs in Machine Learning

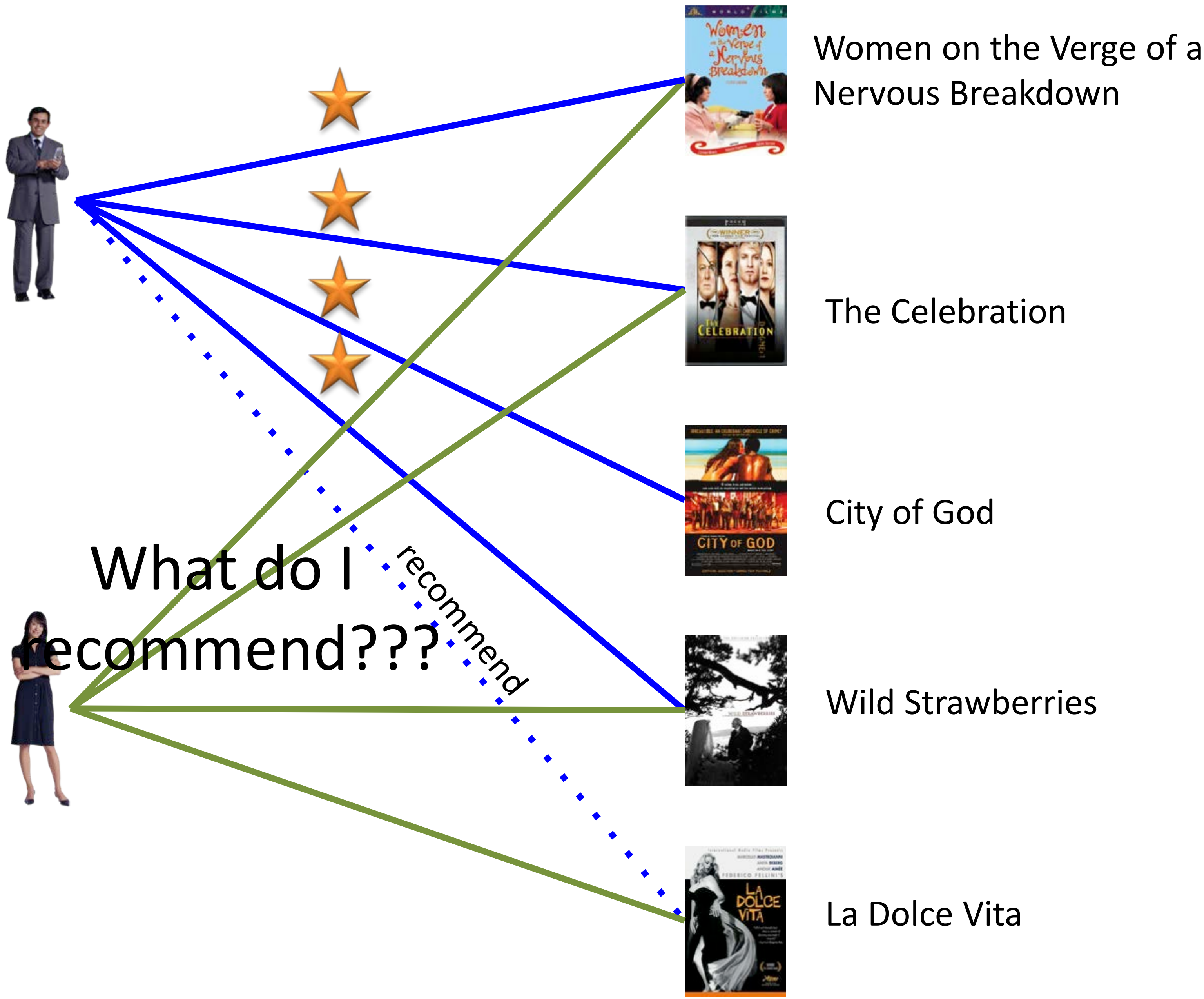
Label a Face and Propagate

Pairwise similarity not enough…

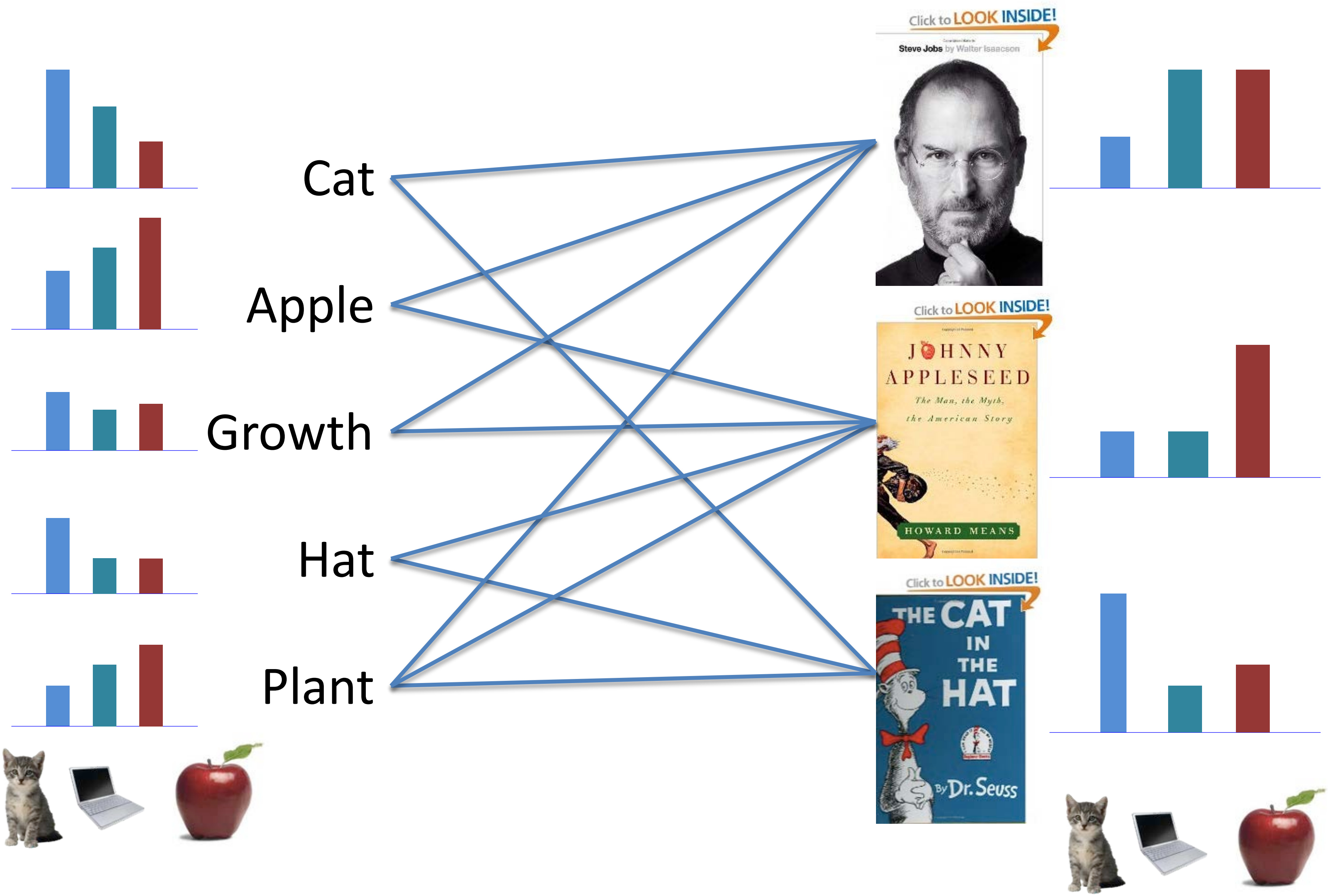
Propagate Similarities & Co-occurrences for Accurate Predictions



Collaborative Filtering: Exploiting Dependencies



Latent Topic Modeling (LDA)



Example Topics Discovered from Wikipedia

party law
government
election court
president elected
council general minister
political national members
committee united office federal
member house parliament vote
public elections democratic held
candidate congress senate district seat
commission supreme votes conservative

york county
american united
city washington john
texas served virginia
pennsylvania war moved ohio
chicago william carolina north
florida illinois george james died
massachusetts president
named jersey born boston south
union west company georgia smith began
michigan fort years philadelphia white

century king
roman empire greek
bc ancient emperor ii
kingdom period battle city
time great war ad early reign
kings iii son rule power greece
army centuries dynasty rome
modern history imperial medieval death
ottoman years led byzantine defeated
ruled year throne athens capital castle
military late iv middle control

engine car
design model cars
production built engines
vehicle class models
speed vehicles designed
produced power front system
version type series motor rear
standard gun company
introduced range ford sold fuel
drive wheel tank fitted factory machine
developed based replaced wheels time
powered small high weight electric body
mounted early

war army military
forces battle force british
command general navy ship
division ships troops corps
service naval regiment
commander infantry attack men
officer fleet soldiers units officers
operations unit june august brigade july fire
training march battalion april operation
captain september three enemy united
october sea royal german marine major

son died
married family
king daughter john
death william father
born wife royal ireland
irish henry house lord
charles sir prince brother
children england queen duke
thomas years marriage george
earl edward english second
elizabeth sons mary james mother
appointed year dublin lady title great
succeeded robert ii member castle

season team
game league games
played coach football
record teams baseball field year
second career play basketball
hockey three yards won bowl
points win series player head conference
championship seasons players draft high
time named national led nfl third major
finished stadium division lead playing ncaa
history runs touchdown signed

species family
birds small long large animals
bird plants genus plant natural
habitat tree fish tropical white black order
leaves brown common forests trees animal
flowers eggs worldwide feed occur
subtropical wild length male breeding
habitats range food female fruit short
insects endemic forest group including
include moist threatened tail

art museum work
works artists collection design
arts painting artist gallery
paintings exhibition style fine
including painted architecture york fashion
painter life early created sculpture artistic
history contemporary collections years
museums worked images time photography
figures academy exhibitions modern
portrait photographs began studio drawing
include exhibited produced designed
period visual

white red
black blue called
color will head green gold side
small hand long arms top flag
horse wear silver common light
dog wood body type large
yellow form worn dogs cut popular left
generally traditional ball front horses shape
hair feet colors time coat three typically
modern face cross

school students
university high college schools
education year program student
campus community programs training
center members science national years
public academic association courses arts
educational include class institute
department teachers colleges classes
offers activities universities district
engineering learning founded faculty girls
sports children boys international board
teaching academy secondary established

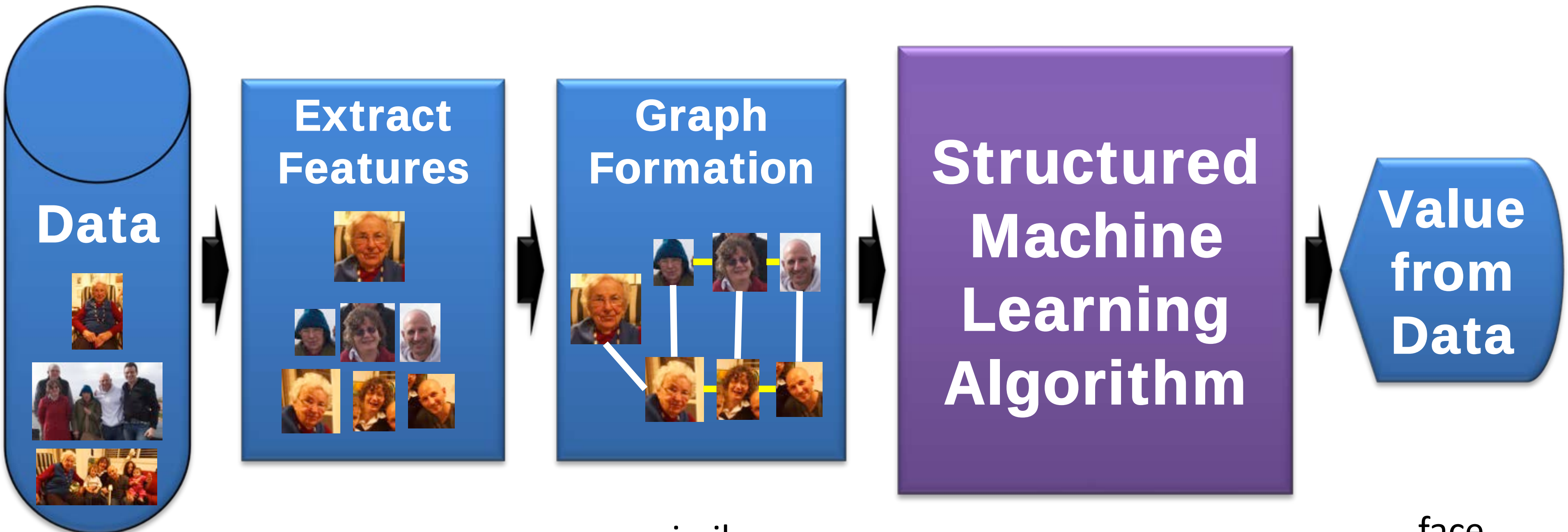
album band
song released
music songs single records
recorded rock bands release
live tour video record albums
label group recording guitar track
cover version tracks number featured time
chart hit uk top performed studio played
singles sound love pop artist solo cd debut
singer artists members included early
second bass

radio station
news television
channel broadcast
stations network media tv
broadcasting time format local
program bbc programming live
fm morning host began sports fox air cable
call hosted coverage music pm sunday
daily channels digital abc aired changed
current launched communications
programme day broadcasts moved cbs
years saturday talk night

age 18 population
income average years
median living 65 males
females households 100 family
people families older town size
city household miles density
american township total area county
races census 2000 square 45 25 64
children 24 44 white female land including
units housing bureau individuals located
poverty united village

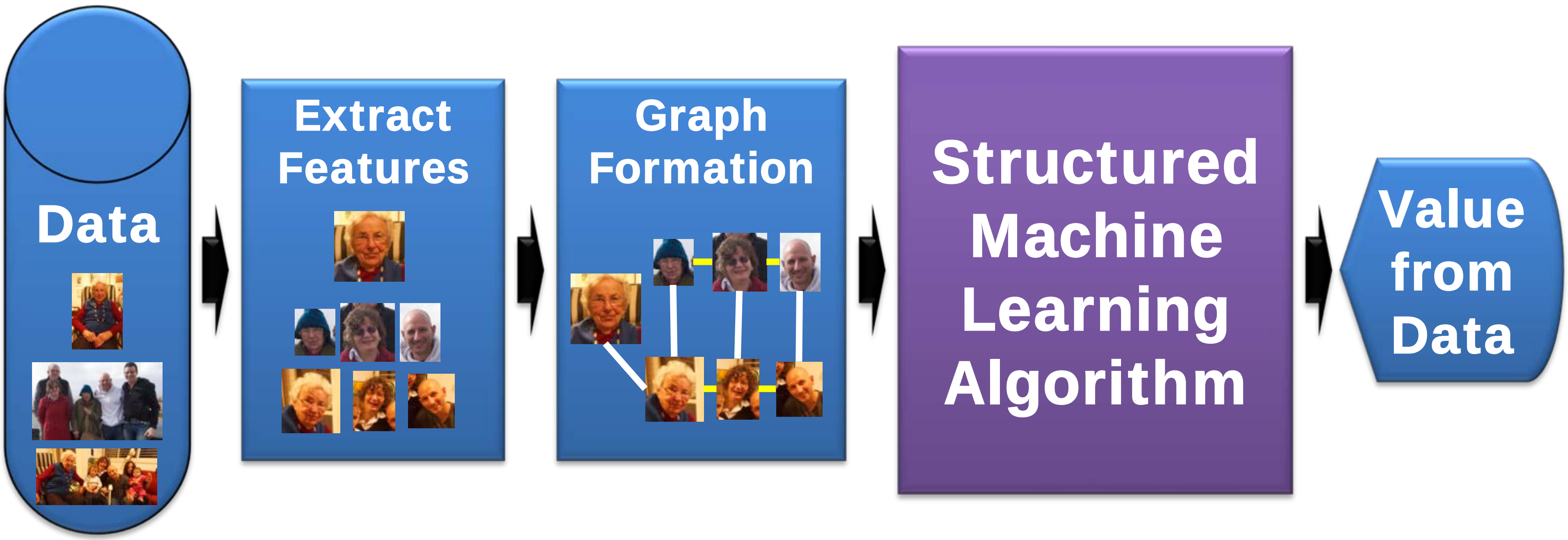
music musical opera
festival orchestra dance
performed jazz piano theatre
performance works concert
symphony composer played performances
instruments musicians classical including
work composed major singing songs folk
instrument ballet composition composers
play performing concerts playing stage
years include popular choir ensemble
sound style time violin hall piece chamber
recordings string

Machine Learning Pipeline



images	faces	similar faces	belief propagation	face labels
docs	important words	shared words	LDA	doc topics
movie ratings	side info	rated movies	collaborative filtering	movie recommend.

Parallelizing Machine Learning



Graph Ingress

mostly data-parallel

Graph-Structured Computation

graph-parallel

ML Tasks Beyond Data-Parallelism

Data-Parallel

Graph-Parallel

Map Reduce

Feature Extraction

Cross Validation

Computing Sufficient Statistics

Graphical Models

Gibbs Sampling

Belief Propagation

Variational Opt.

Semi-Supervised Learning

Label Propagation

CoEM

Collaborative Filtering

Tensor Factorization

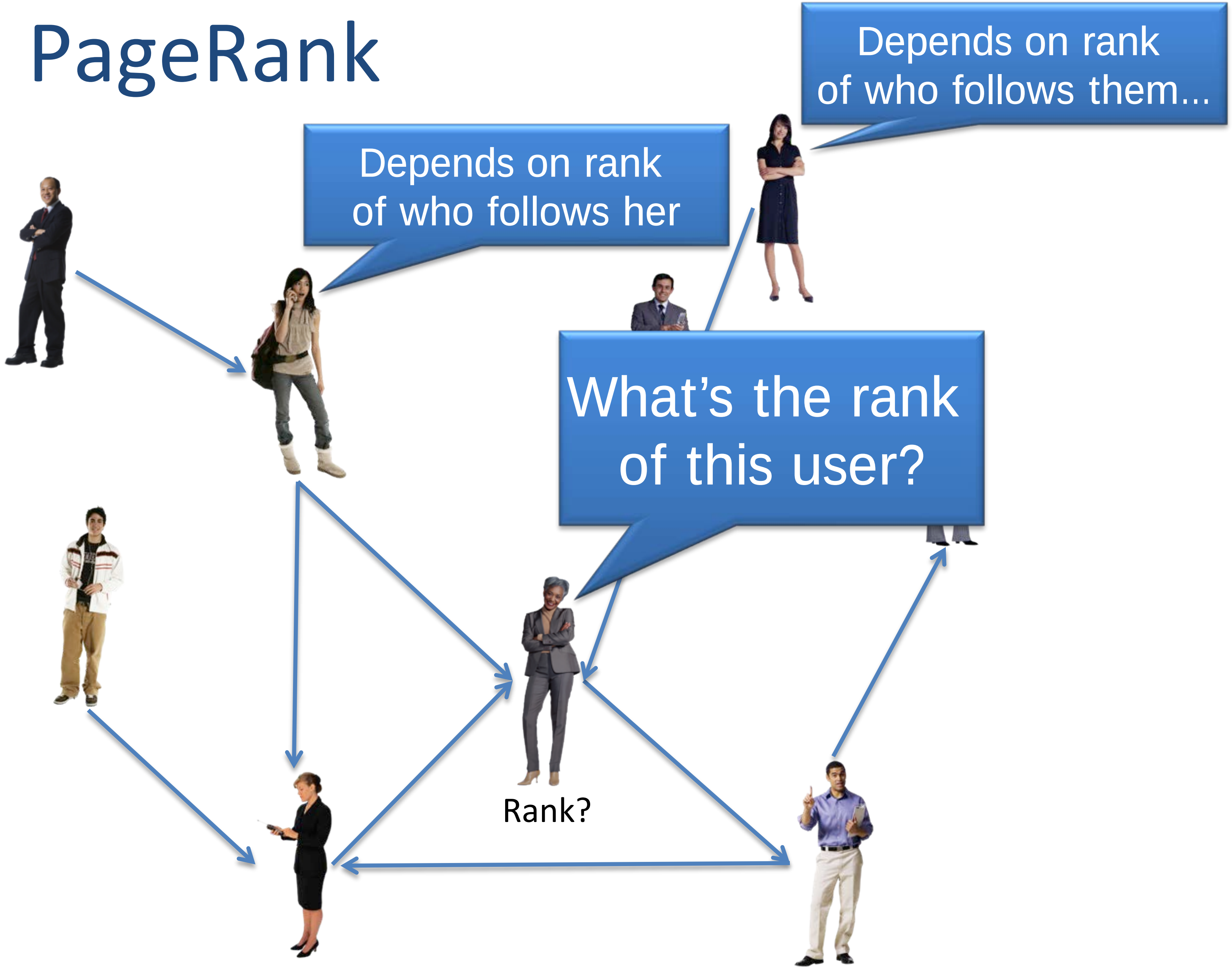
Graph Analysis

PageRank

Triangle Counting

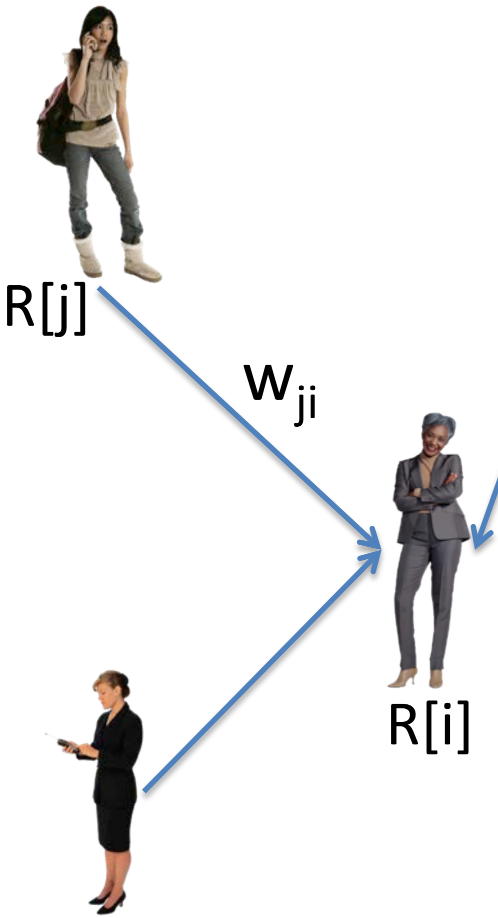
Example of a Graph-Parallel Algorithm

PageRank



Loops in graph → Must iterate!

PageRank Iteration



Iterate until convergence:

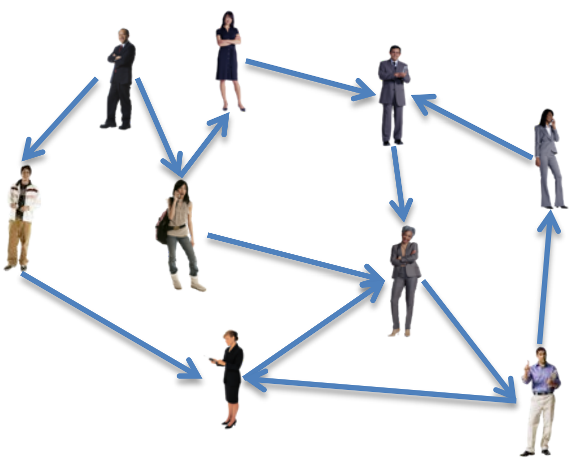
"My rank is weighted average of my friends' ranks"

$$R[i] = \alpha + (1 - \alpha) \sum_{(j,i) \in E} w_{ji} R[j]$$

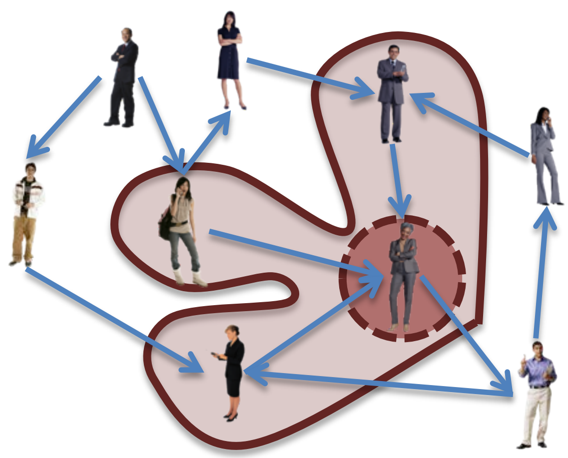
- α is the random reset probability
- w_{ji} is the prob. transitioning (similarity) from j to i

Properties of Graph Parallel Algorithms

Dependency Graph



Local Updates



Iterative Computation

Addressing Graph-Parallel ML

Data-Parallel

Graph-Parallel

Map Reduce

Feature Extraction

Cross Validation

Computing Sufficient Statistics

Graph-Parallel Abstraction

Graphical Models
Gibbs Sampling
Belief Propagation
Variational Opt.

Semi-Supervised Learning
Label Propagation
CoEM

Collaborative Filtering
Tensor Factorization

Data-Mining
PageRank
Triangle Counting

Graph Computation:

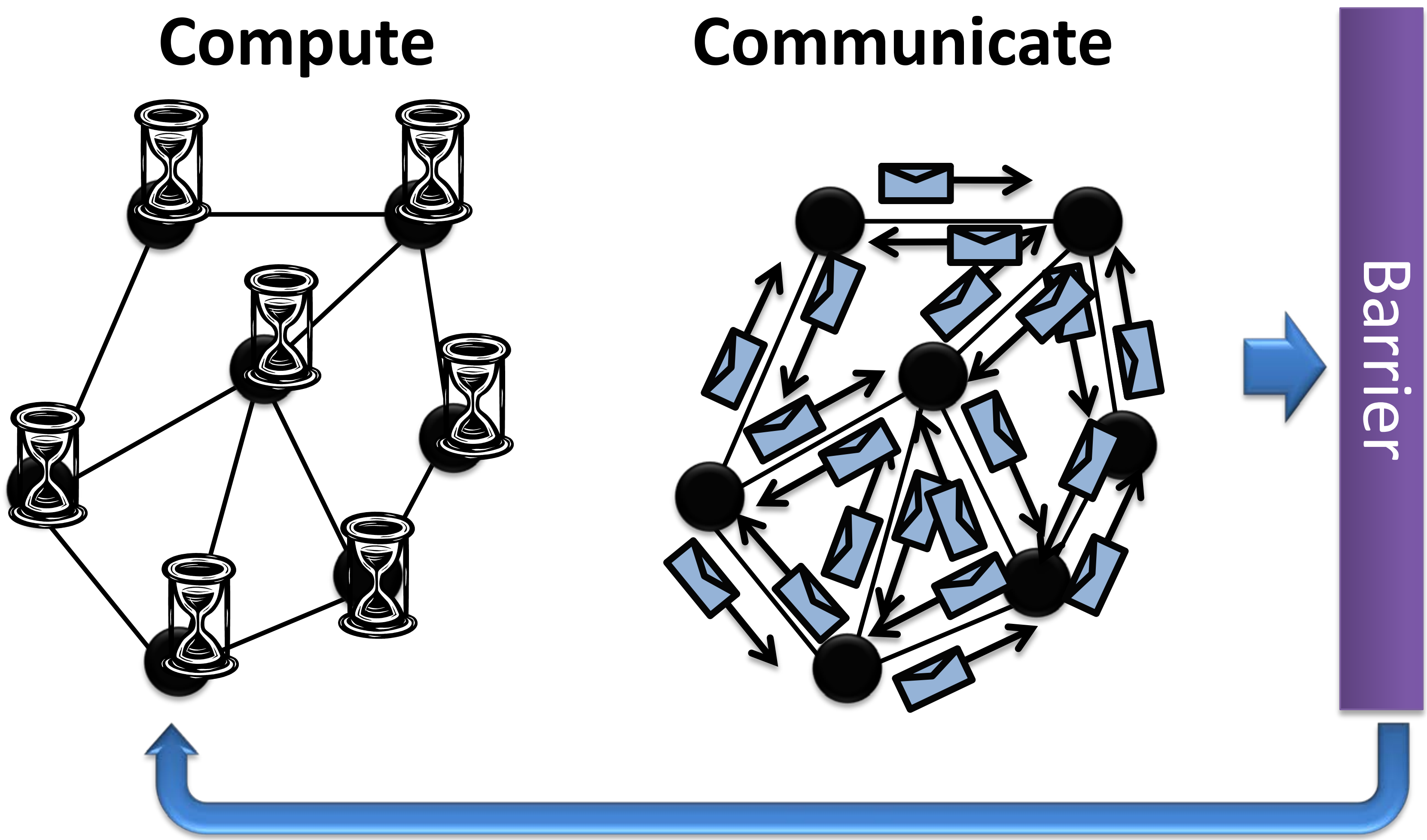
Synchronous

v.

Asynchronous

Bulk Synchronous Parallel Model: Pregel (Giraph)

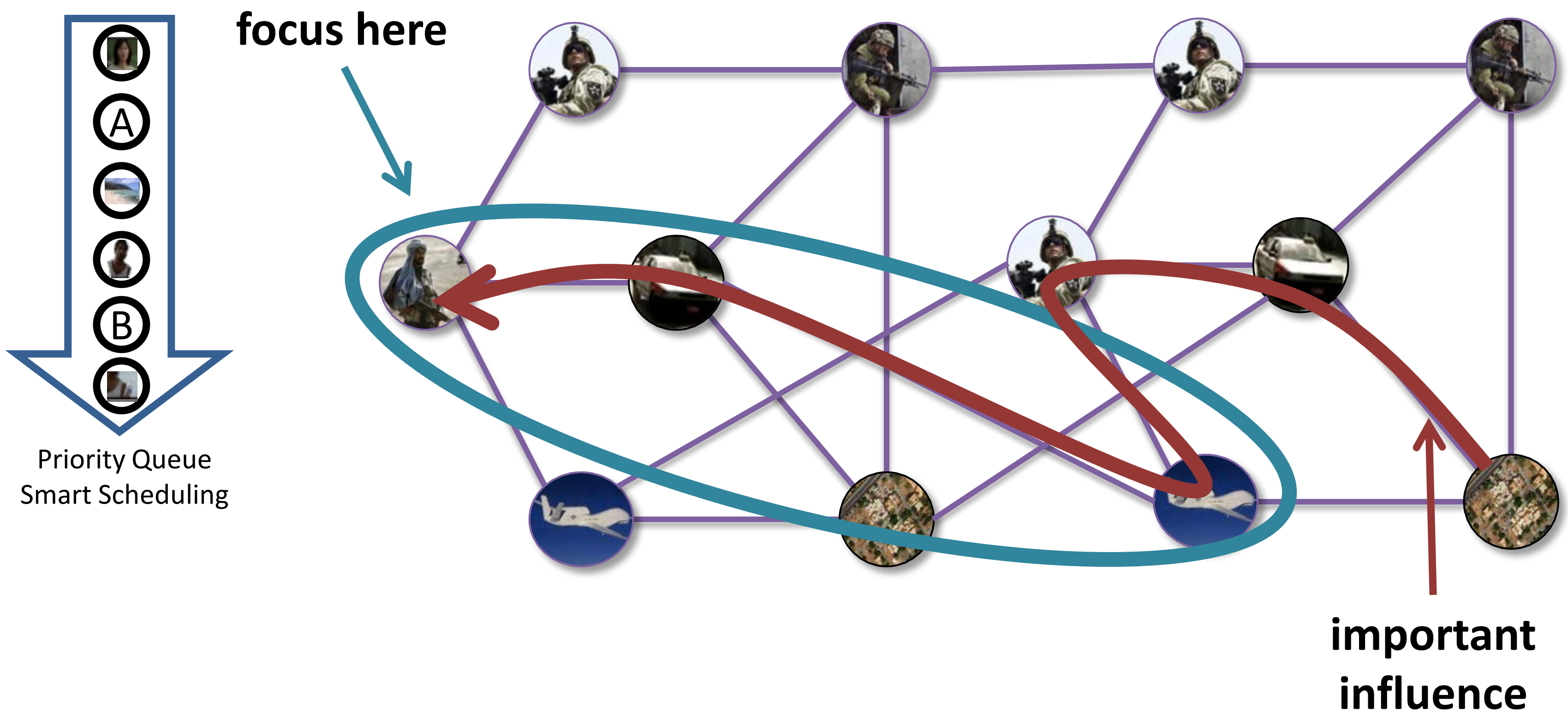
[Valiant ‘90]



Bulk synchronous parallel model ***provably inefficient*** *for some ML tasks*

Analyzing Belief Propagation

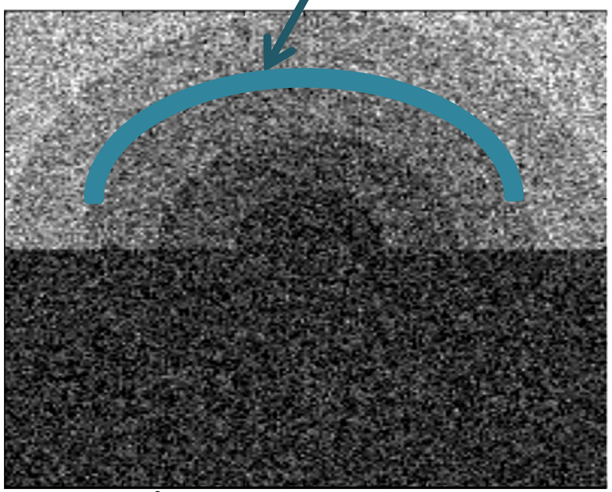
[Gonzalez, Low, G. '09]



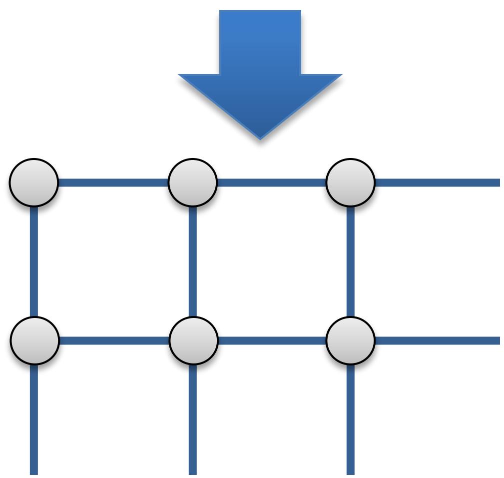
Asynchronous Parallel Model (rather than BSP) fundamental for efficiency

Asynchronous Belief Propagation

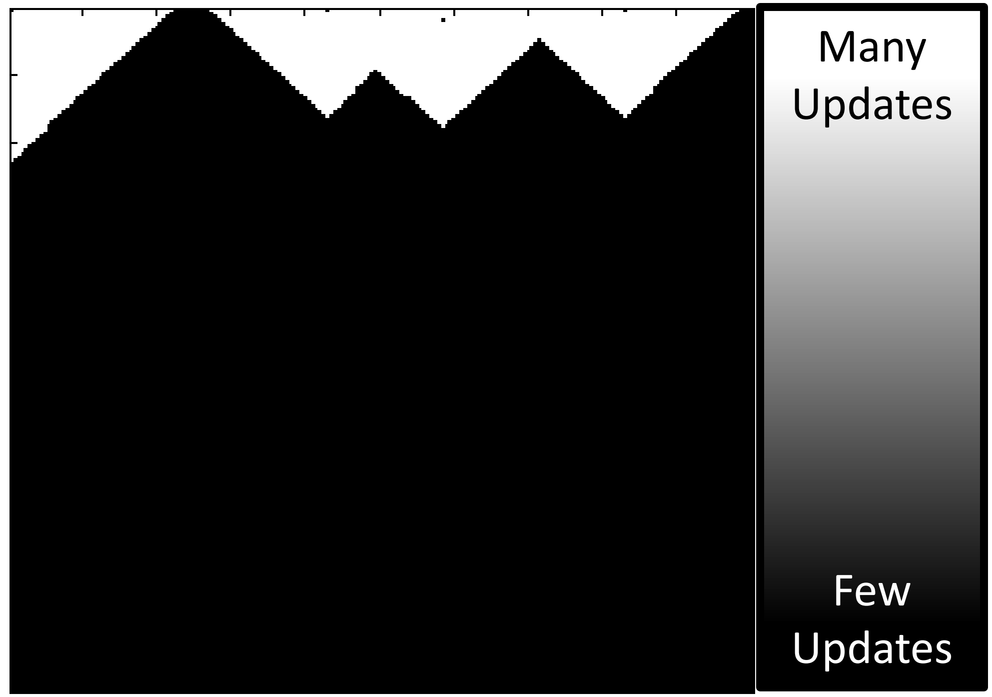
Challenge = Boundaries



Synthetic Noisy Image



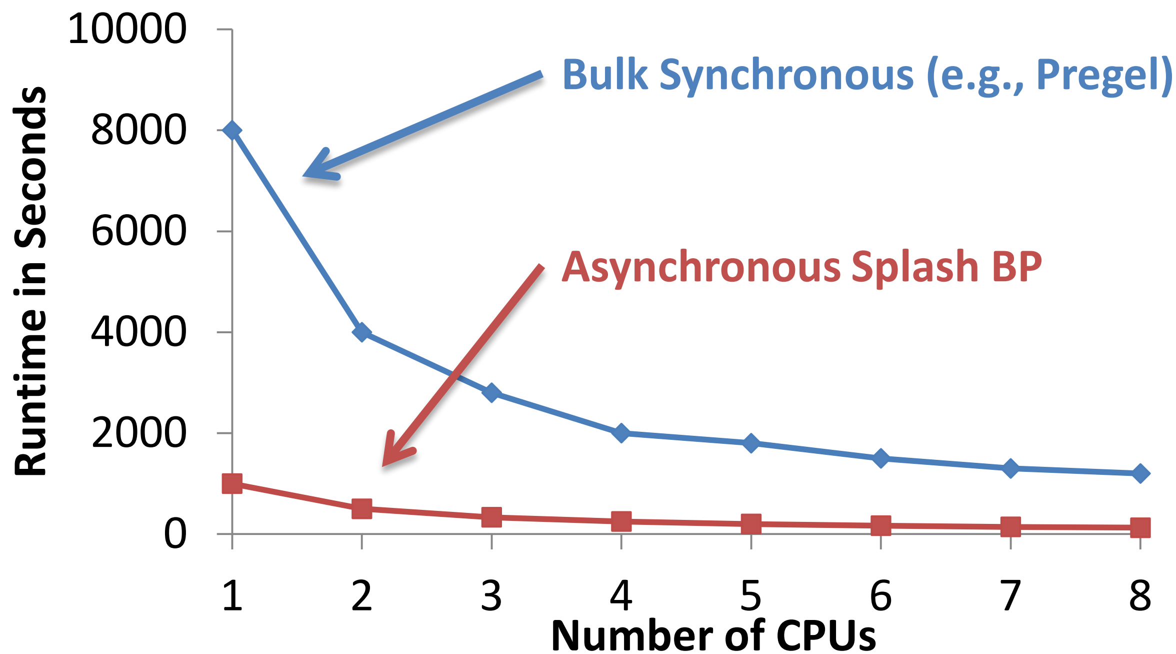
Graphical Model



Cumulative Vertex Updates

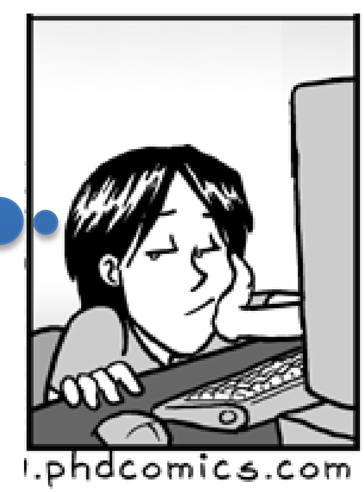
Algorithm identifies and focuses on hidden sequential structure

BSP ML Problem:
Synchronous Algorithms can be **Inefficient**



Theorem:
Bulk Synchronous BP
O(#vertices) slower
than Asynchronous BP

Efficient parallel implementation was painful, painful, painful…



The Need for a New Abstraction

- Need: Asynchronous, Dynamic Parallel Computations

Data-Parallel ⟷ Graph-Parallel

Map Reduce

Feature Extraction

Cross Validation

Computing Sufficient Statistics

BSP, e.g., Pregel

Carnegie Mellon

Graphical Models
Gibbs Sampling
Belief Propagation
Variational Opt.

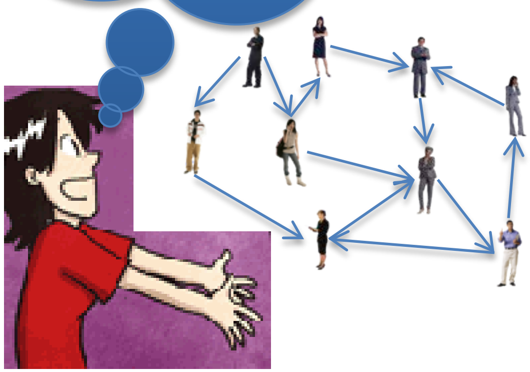
Semi-Supervised Learning
Label Propagation
CoEM

Collaborative Filtering
Tensor Factorization

Data-Mining
PageRank
Triangle Counting

The **GraphLab** Goals

Know how to solve ML problem on 1 machine



GraphLab
Carnegie Mellon

+

Efficient parallel predictions

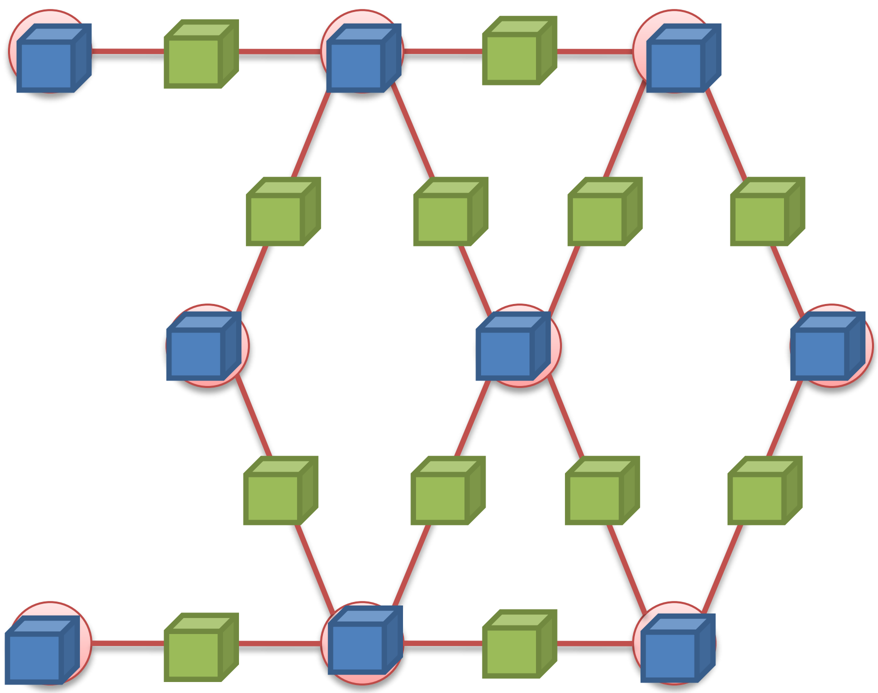
GraphLab

Carnegie Mellon

1

Data Graph

Data associated with vertices and edges



Graph:

- Social Network

Vertex Data:

- User profile text
- Current interests estimates

Edge Data:

- Similarity weights

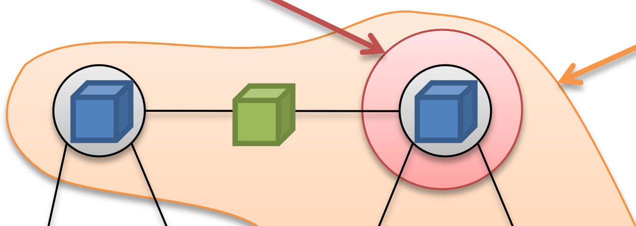
How do we *program* **graph** computation?

"Think like a Vertex."

-Malewicz et al. [SIGMOD'10]

Update Functions

User-defined program: applied to **vertex** transforms data in **scope** of vertex



pagerank(i, scope){
  // Get Neighborhood data
  ($R[i]$, w_{ij}, $R[j]$) ←scope;

}

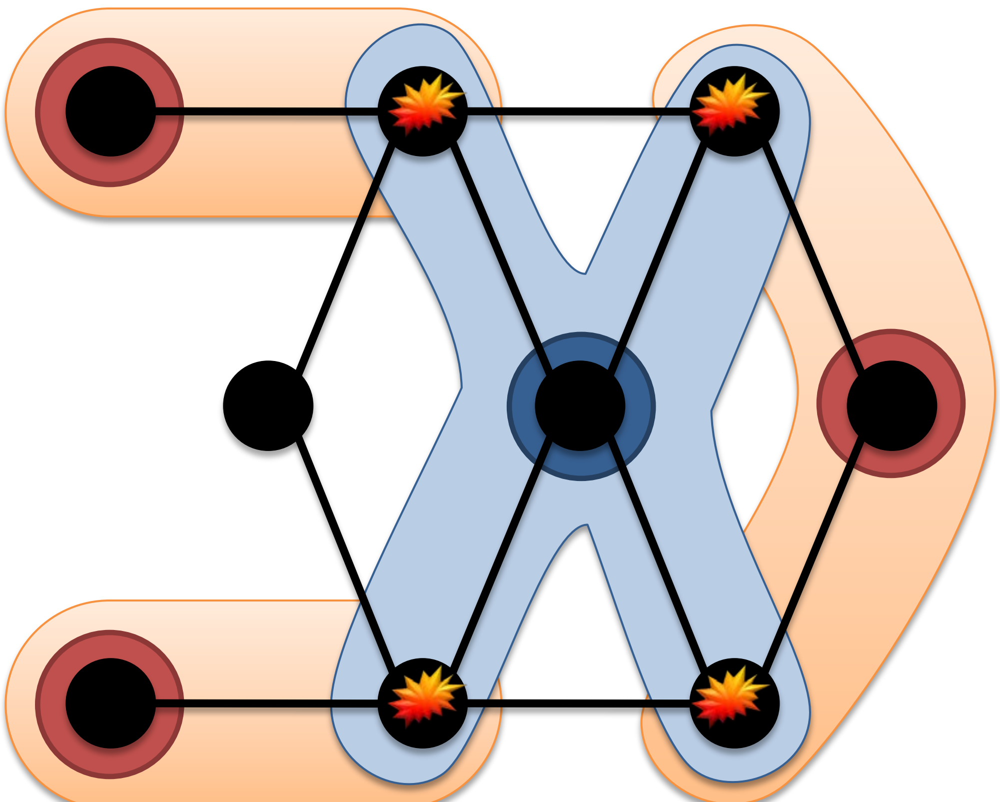
Update function applied (asynchronously) in parallel until convergence

Many schedulers available to prioritize computation

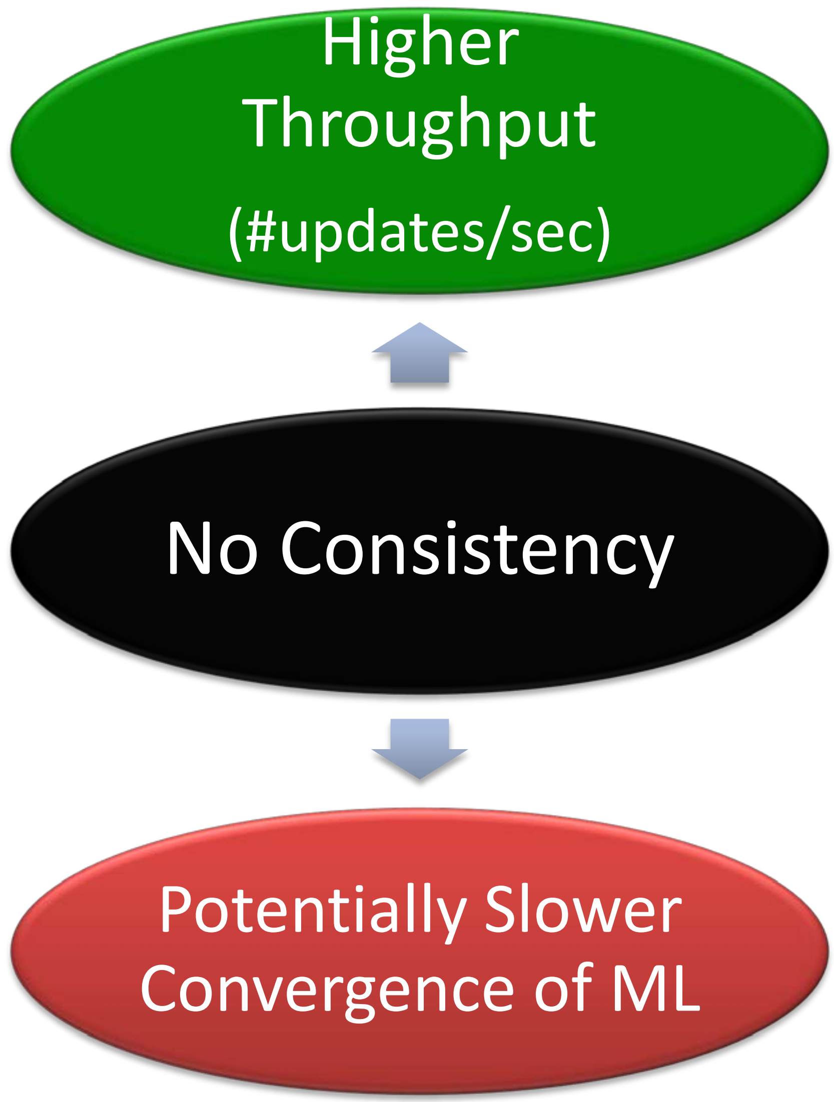
Dynamic computation

Ensuring Race-Free Code

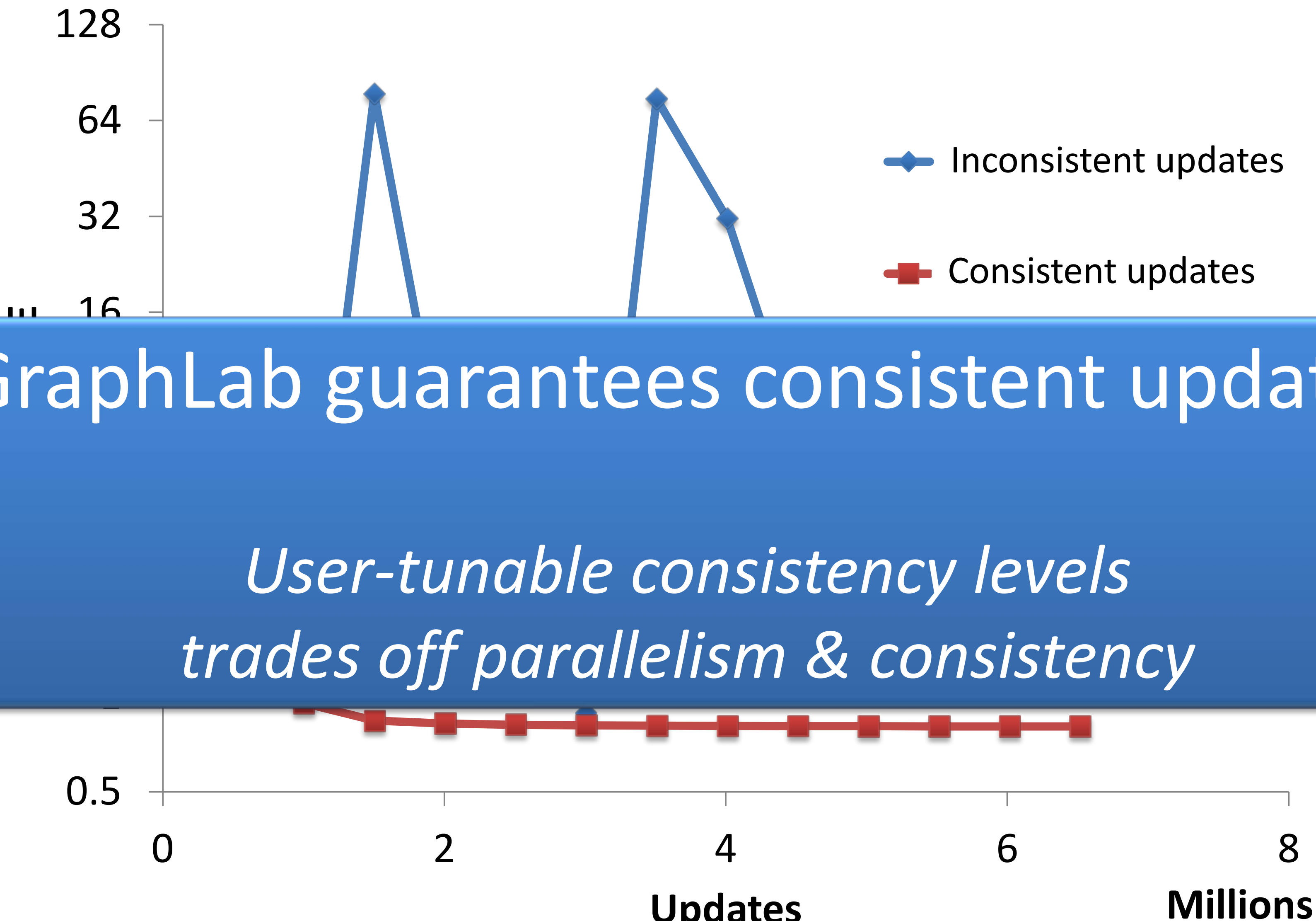
How much can computation **overlap**?



Need for Consistency?



Consistency in Collaborative Filtering



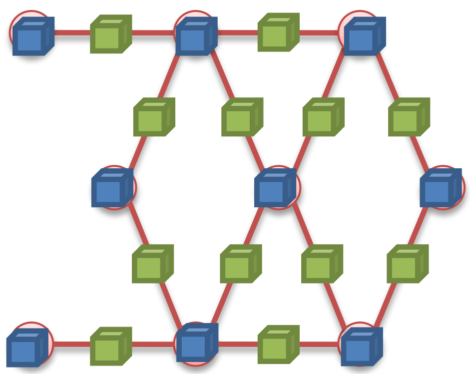
GraphLab guarantees consistent updates

User-tunable consistency levels
trades off parallelism & consistency

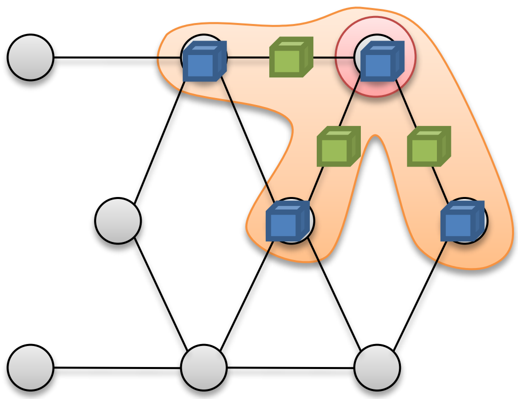
Netflix data, 8 cores

The GraphLab Framework

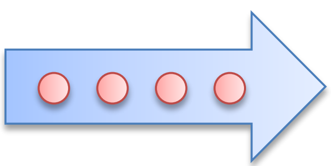
Graph Based
Data Representation



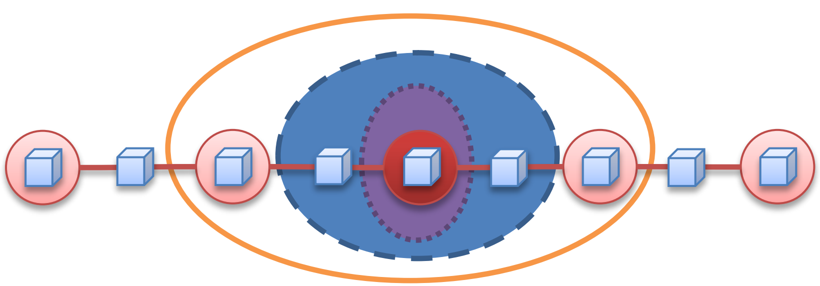
Update Functions
User Computation



Scheduler



Consistency Model



Alternating Least Squares

SVD

Splash Sampler

CoEM

Bayesian Tensor Factorization

Lasso

Belief Propagation

PageRank

LDA

GraphLab
Carnegie Mellon

SVM

Gibbs Sampling

Dynamic Block Gibbs Sampling

K-Means

Matrix Factorization

...Many others...

Linear Solvers

Never Ending Learner Project (CoEM)

Hadoop	95 Cores	7.5 hrs
Distributed GraphLab	**32 EC2 machines**	**80 secs**

0.3% of Hadoop time

2 orders of mag faster → 2 orders of mag cheaper

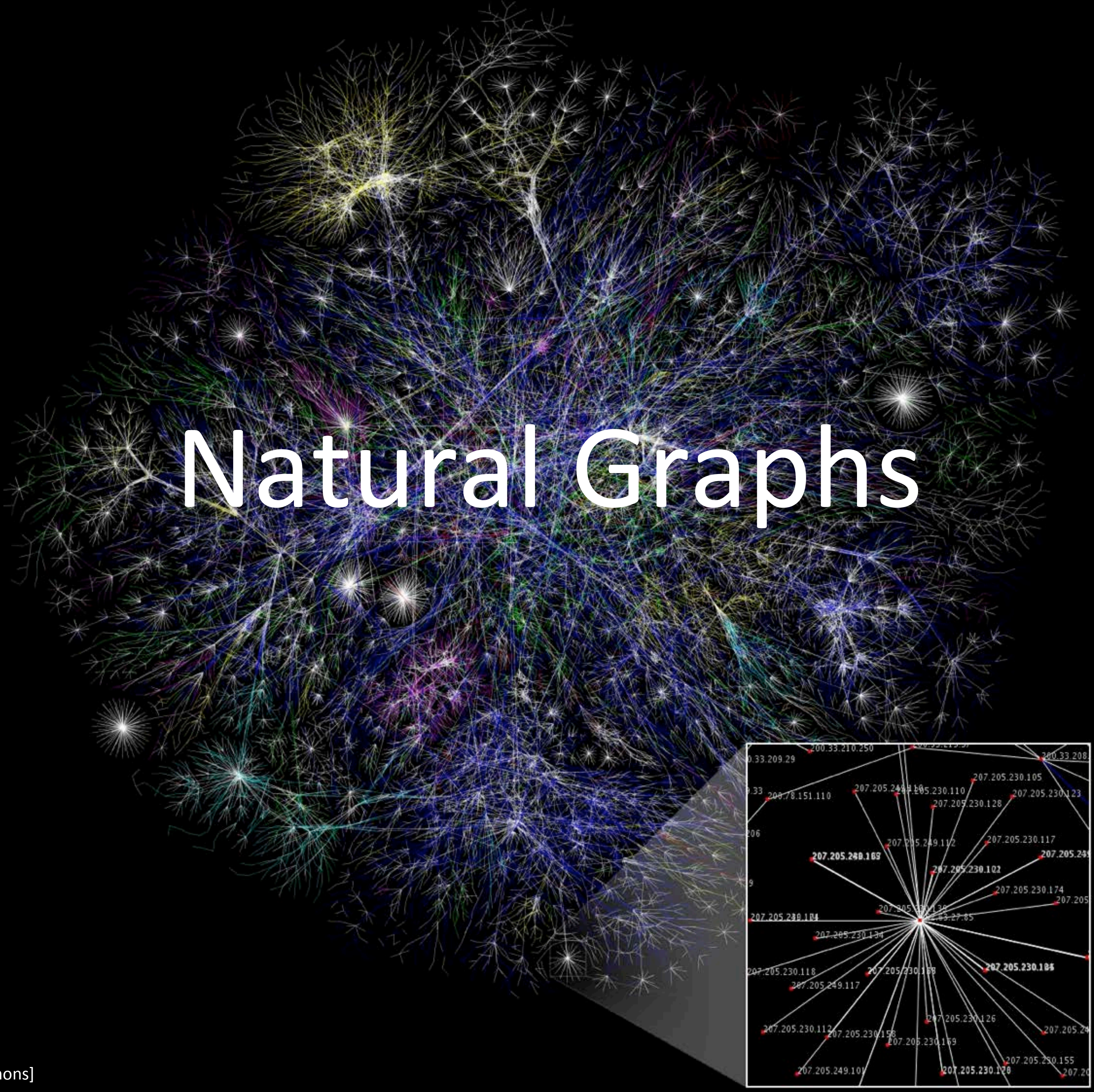
Thus far…

GraphLab 1 provided exciting scaling performance

But…

We couldn't scale up to Altavista Webgraph 2002 1.4B vertices, 6.7B edges

Natural Graphs



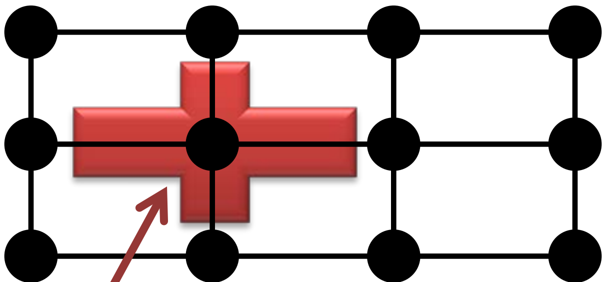
[Image from WikiCommons]

Problem:

Existing ***distributed*** graph computation systems perform poorly on **Natural Graphs**

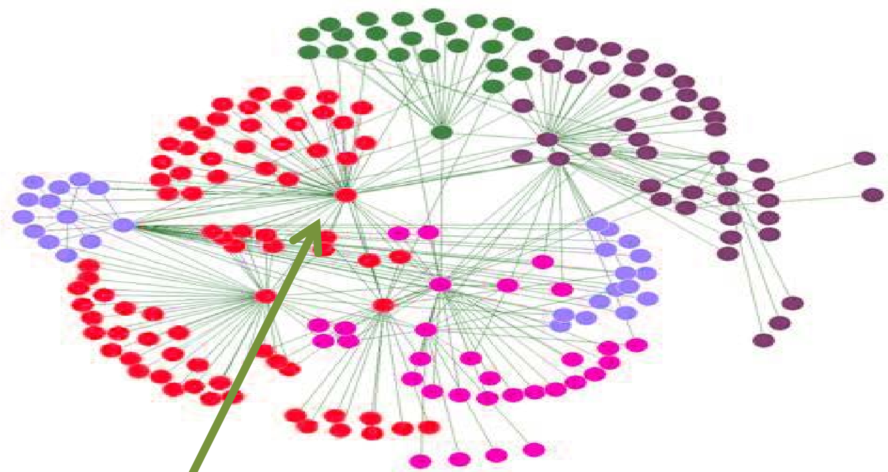
Achilles Heel: Idealized Graph Assumption

Assumed…



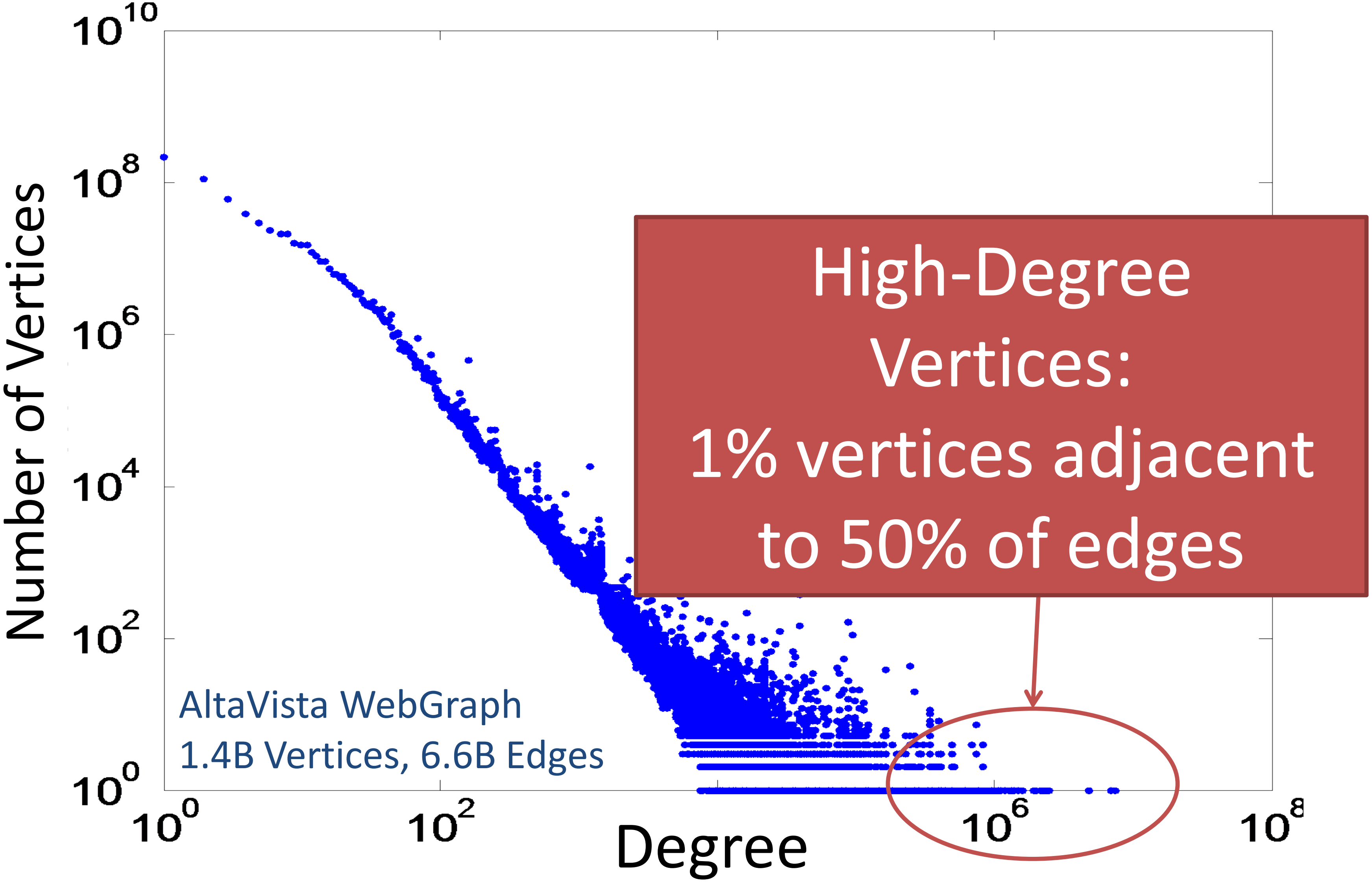
Small degree → Easy to partition

But, Natural Graphs…



Many high degree vertices (power-law degree distribution) → Very hard to partition

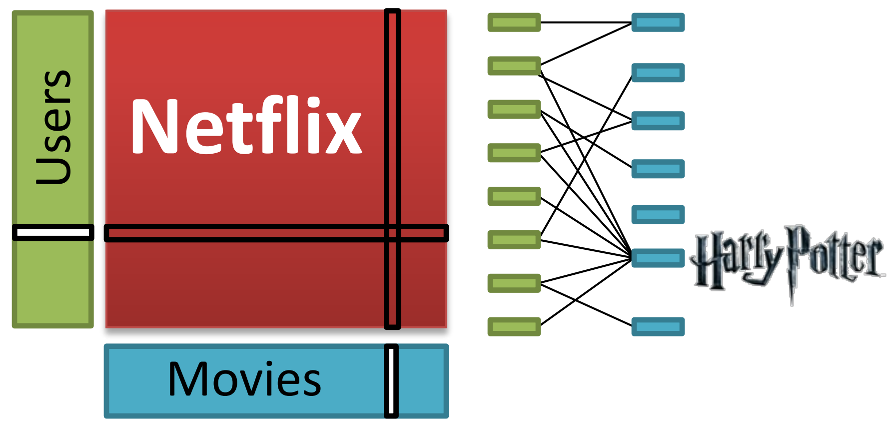
Power-Law Degree Distribution



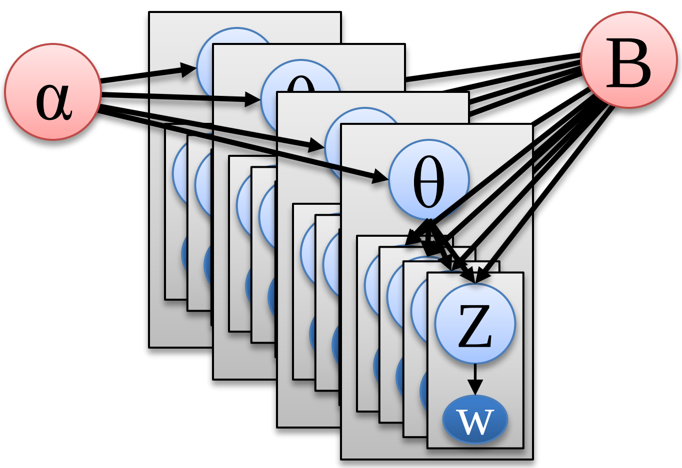
High Degree Vertices are Common

“Social” People

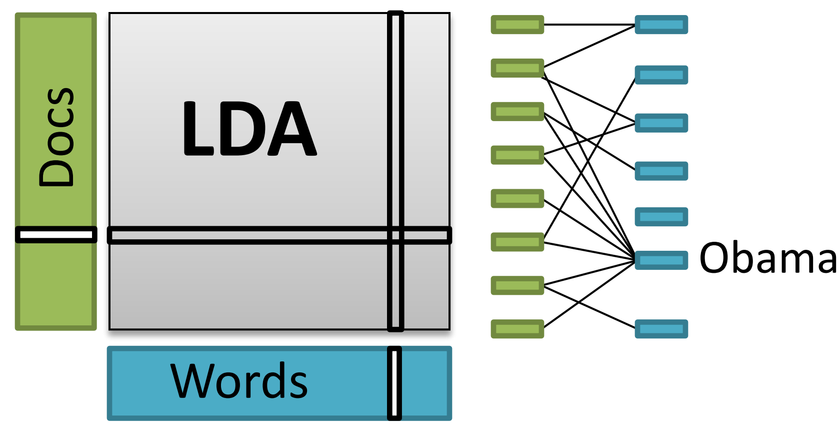
Popular Movies



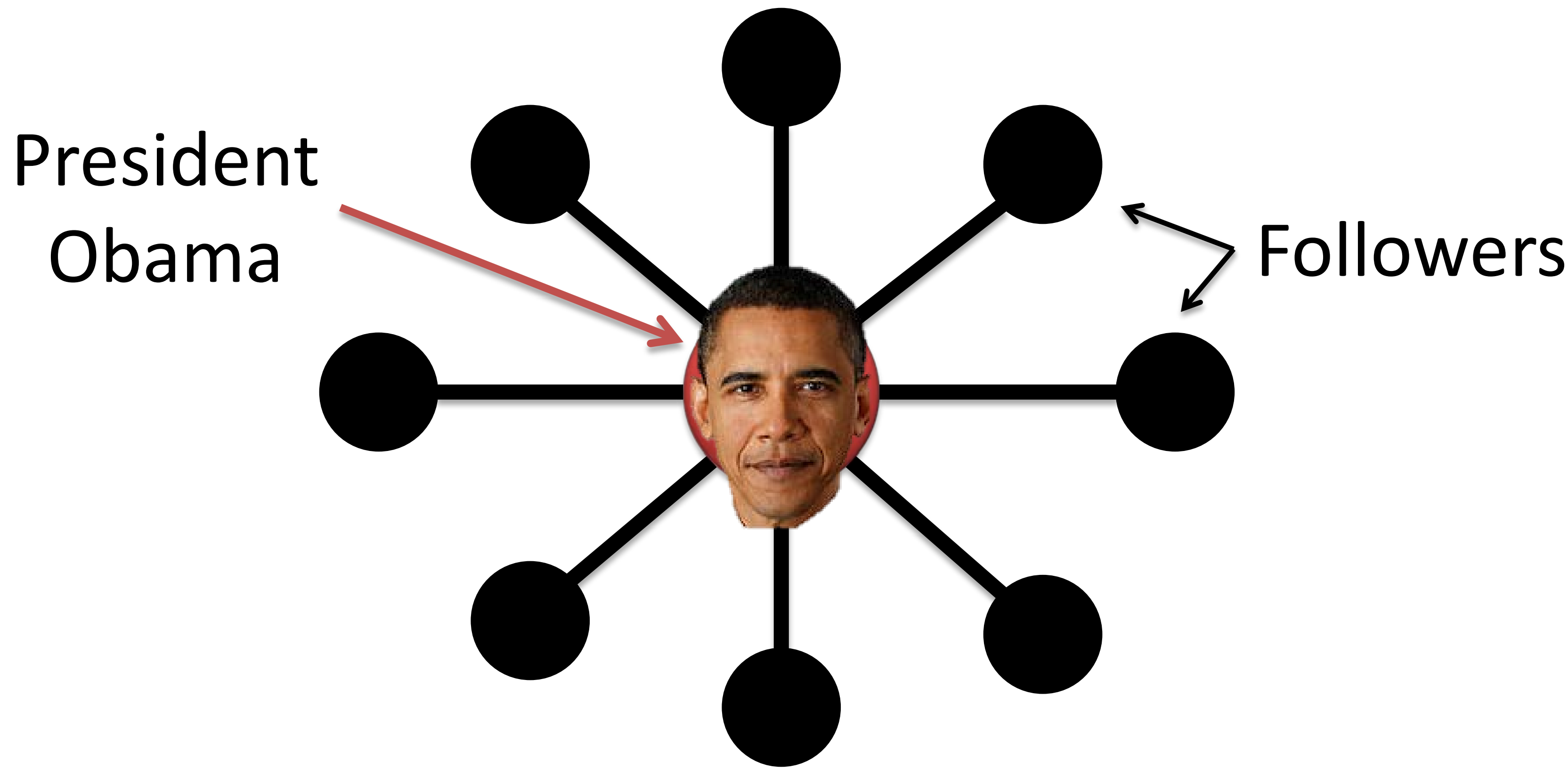
Hyper Parameters



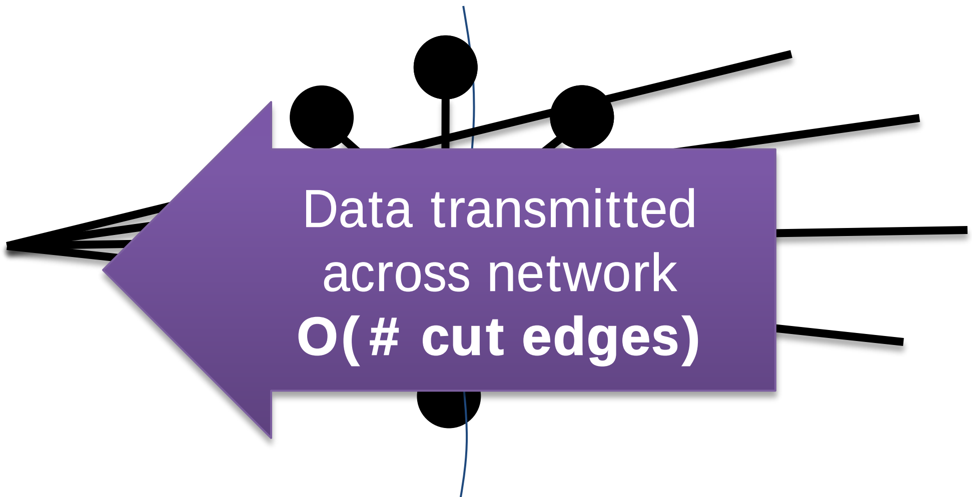
Common Words



Power-Law Degree Distribution
“Star Like” Motif



Problem:
High Degree Vertices ➔ High Communication for Distributed Updates



Natural graphs do not have low-cost balanced cuts

[Leskovec et al. 08, Lang 04]

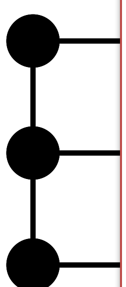
Popular partitioning tools (Metis, Chaco,...) perform poorly

[Abou-Rjeili et al. 06]

Extremely slow and require substantial memory

Random Partitioning

- Both GraphLab 1 and Pregel proposed Random (hashed) partitioning for Natural Graphs



For *p* Machines:

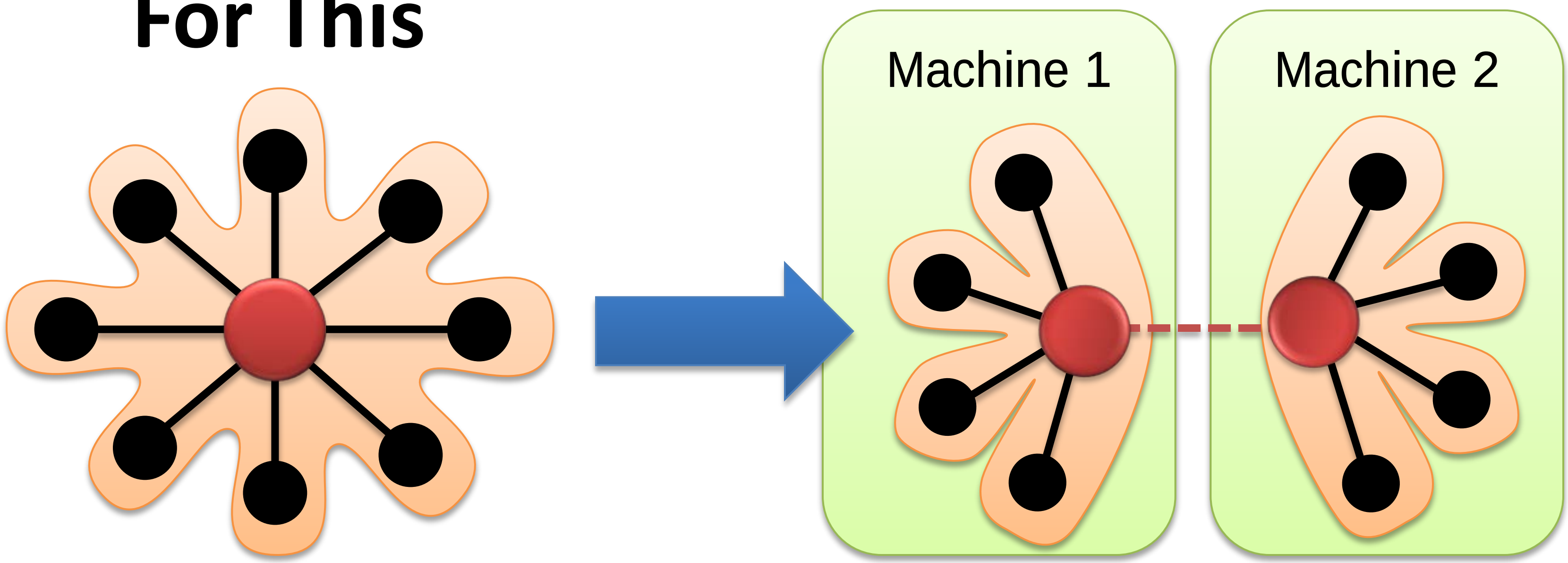
10 Machines → 90% of edges cut

100 Machines → 99% of edges cut!

All data is communicated… Little advantage over MapReduce

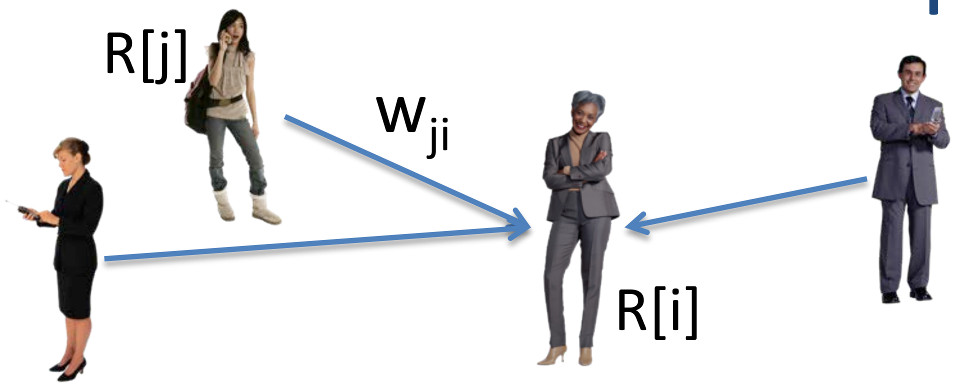
Program For This

Run on This



- Split **High-Degree** vertices
- **New Abstraction →** *Leads to this Split Vertex Strategy*

Common Pattern for Update Fncs.



GraphLab_PageRank(i)

```
// Compute sum over neighbors
total = 0
foreach( j in in_neighbors(i)):
  total = total + R[j] * w_ji
```

Gather **Information About Neighborhood**

```
// Update the PageRank
R[i] = 0.1 + total
```

Apply **Update to Vertex**

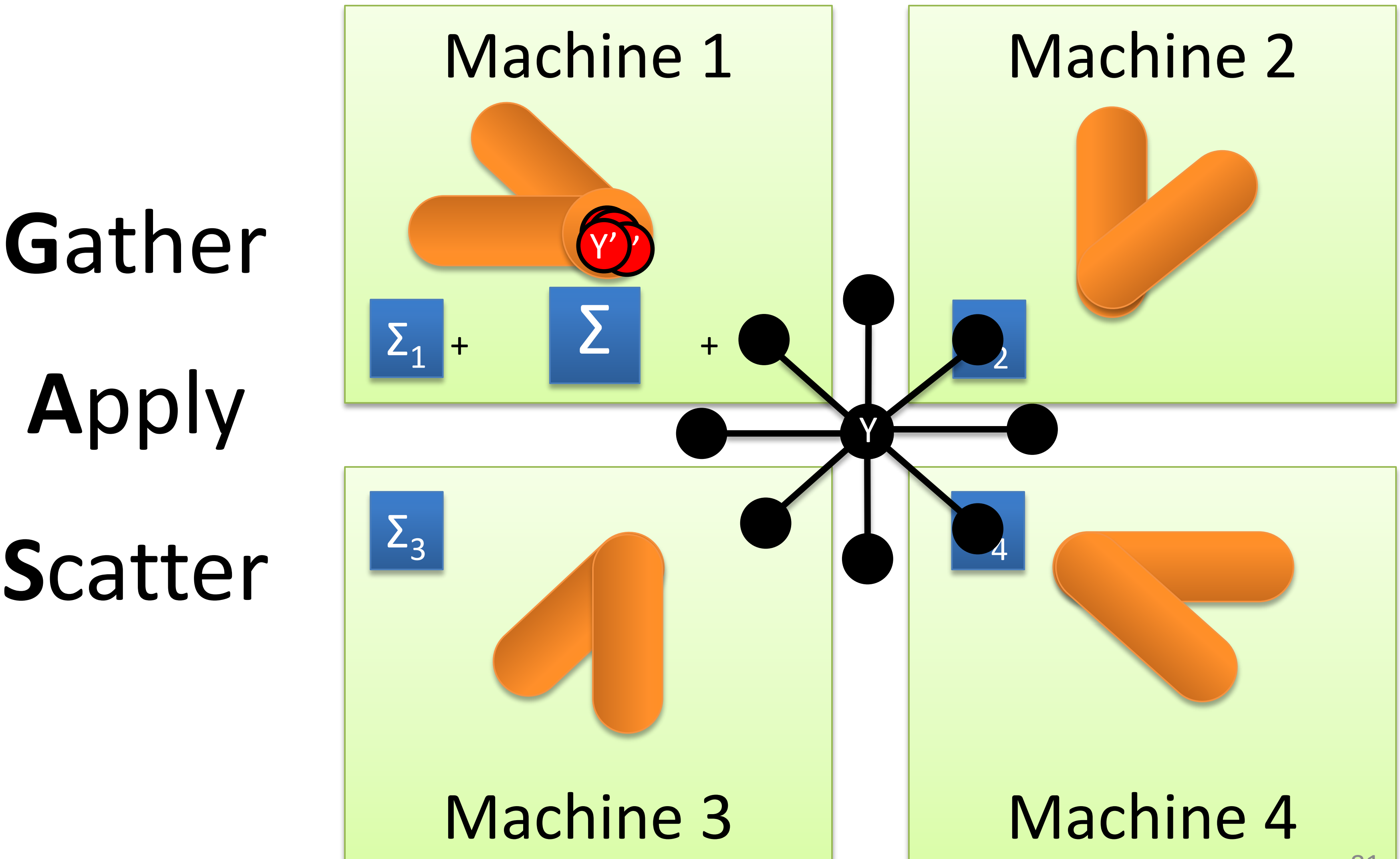
```
// Trigger neighbors to run again
if R[i] not converged then
  foreach( j in out_neighbors(i))
    signal vertex-program on j
```

Scatter **Signal to Neighbors & Modify Edge Data**

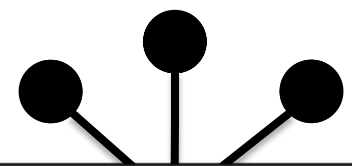
Many ML Algorithms fit into GAS Model

graph analytics, inference in graphical models, matrix factorization, collaborative filtering, clustering, LDA, …

Distributed Execution of a GraphLab 2 Vertex-Program



Minimizing Communication in GraphLab 2: **Vertex Cuts**



Communication linear in # spanned machines

GraphLab 2 includes novel vertex cut algorithms

↓

Provides order of magnitude gains in performance

machines per vertex

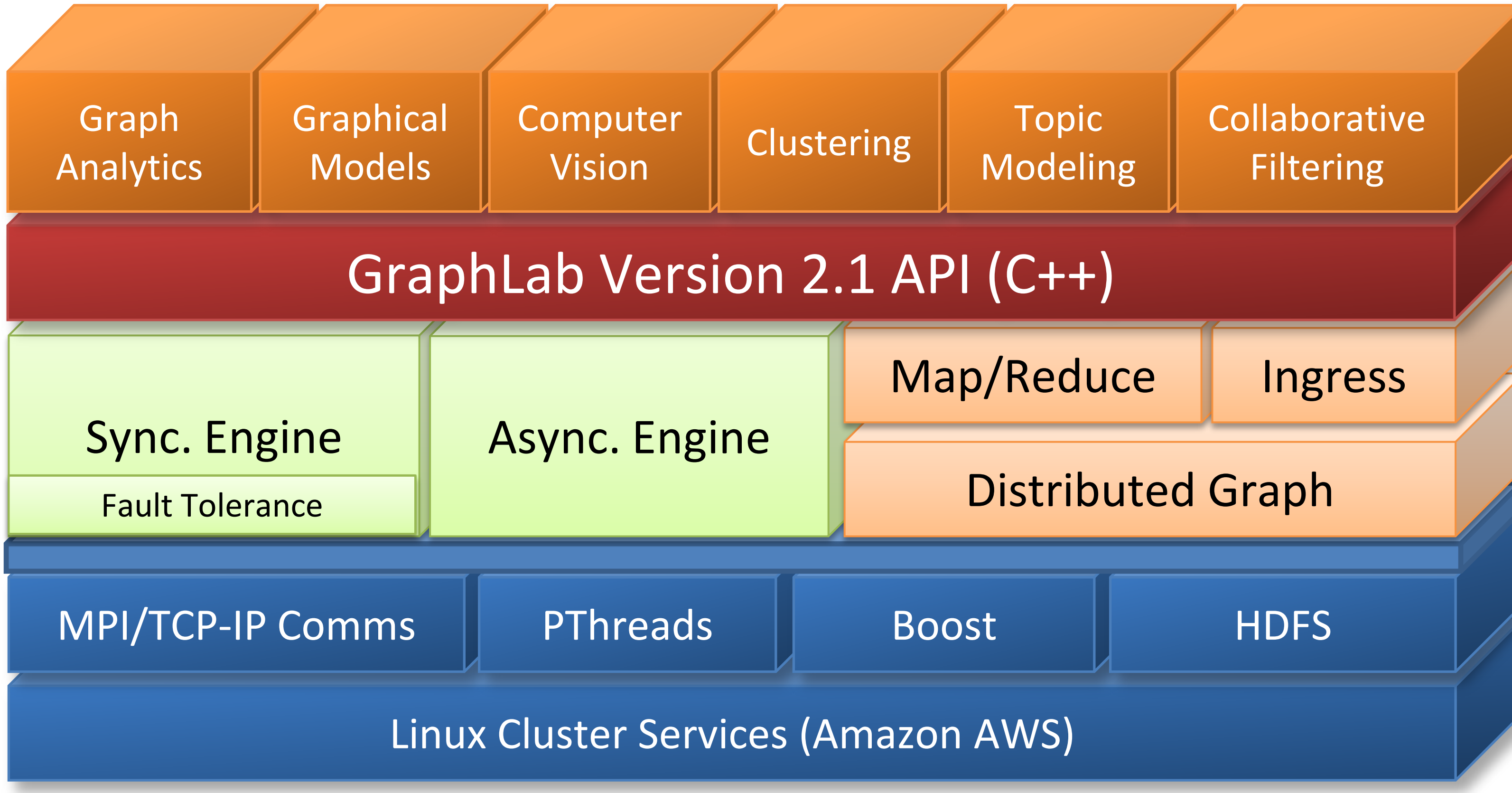
Percolation theory suggests Power Law graphs can be split by removing only a small set of vertices [Albert et al. 2000]

➔

Small vertex cuts possible!

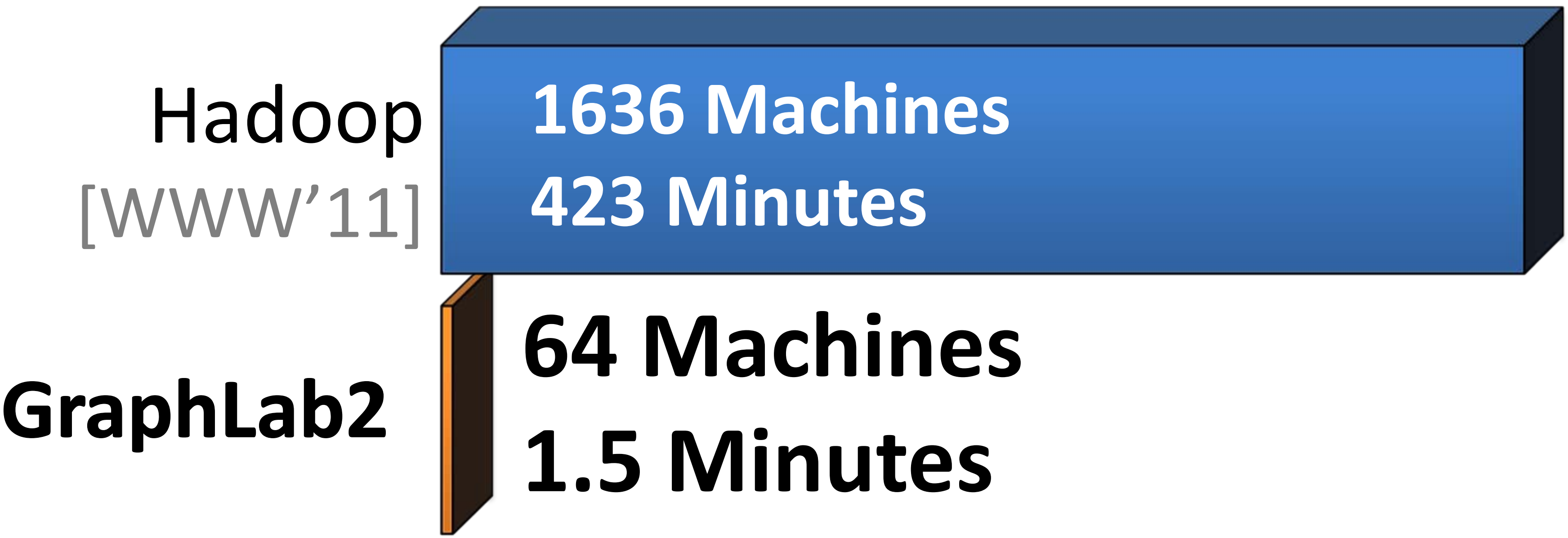
GraphLab2

From the Abstraction to a System


Graph Analytics
Graphical Models
Computer Vision
Clustering
Topic Modeling
Collaborative Filtering
GraphLab Version 2.1 API (C++)
Sync. Engine
Fault Tolerance
Async. Engine
Map/Reduce
Ingress
Distributed Graph
MPI/TCP-IP Comms
PThreads
Boost
HDFS
Linux Cluster Services (Amazon AWS)

Triangle Counting on Twitter Graph

34.8 Billion Triangles



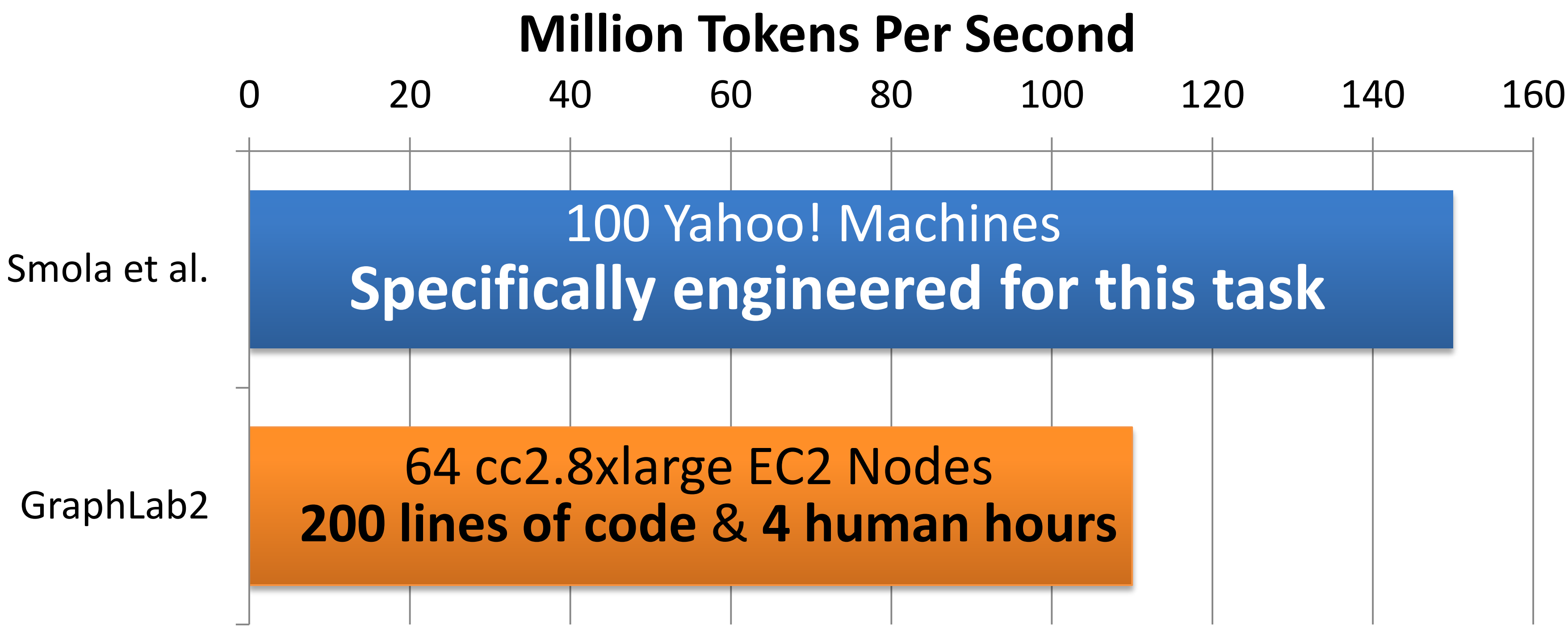
Why? **Wrong Abstraction** →

Broadcast O(degree^2) messages per Vertex

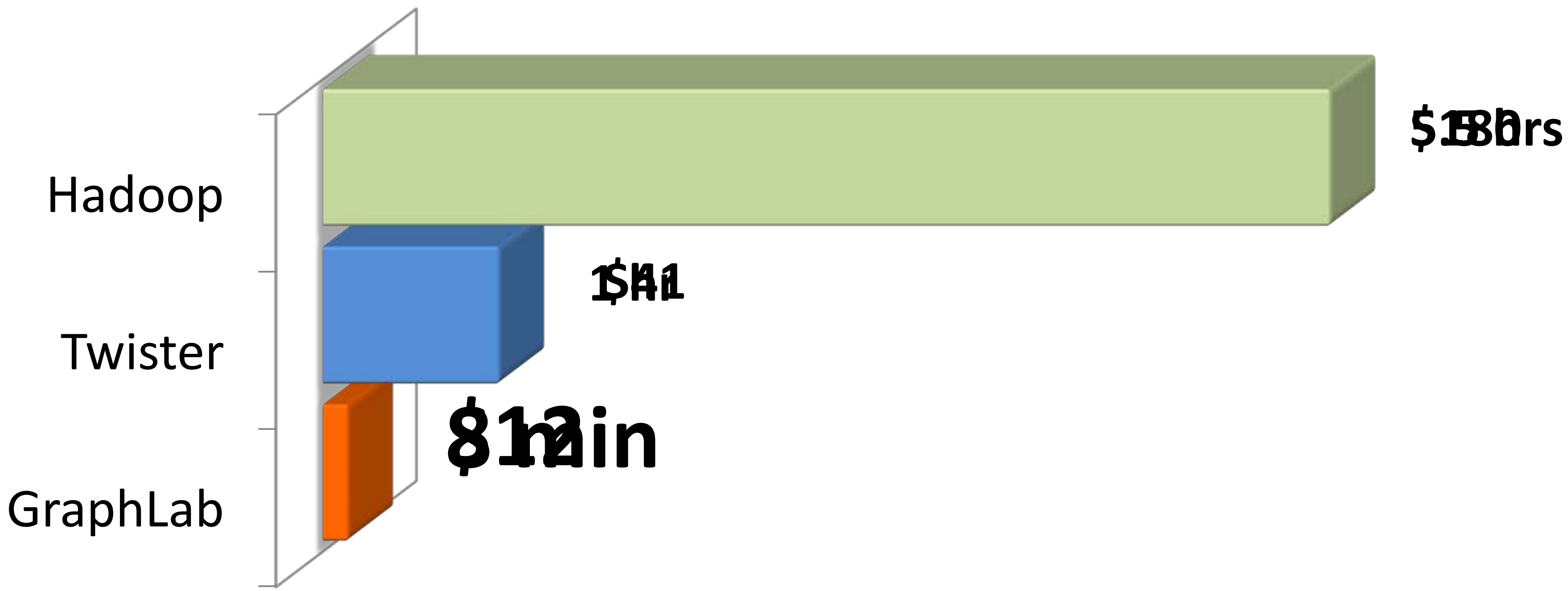
S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," WWW'11

Topic Modeling (LDA)

- English language Wikipedia
  - 2.6M Documents, 8.3M Words, 500M Tokens
  - Computationally intensive algorithm



PageRank



40M Webpages, 1.4 Billion Links

Hadoop results from [Kang et al. '11]
Twister (in-memory MapReduce) [Ekanayake et al. ‘10]

How well does GraphLab scale?

Yahoo Altavista Web Graph (2002):

One of the largest publicly available webgraphs

1.4B Webpages, 6.7 Billion Links

7 seconds per iter.

1B links processed per second

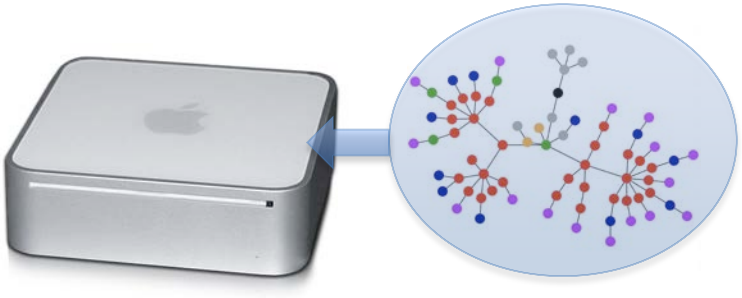
30 lines of user code

1024 Cores (2048 HT)

4.4 TB RAM

GraphChi: Going small with GraphLab

Solve huge problems on small or embedded devices?

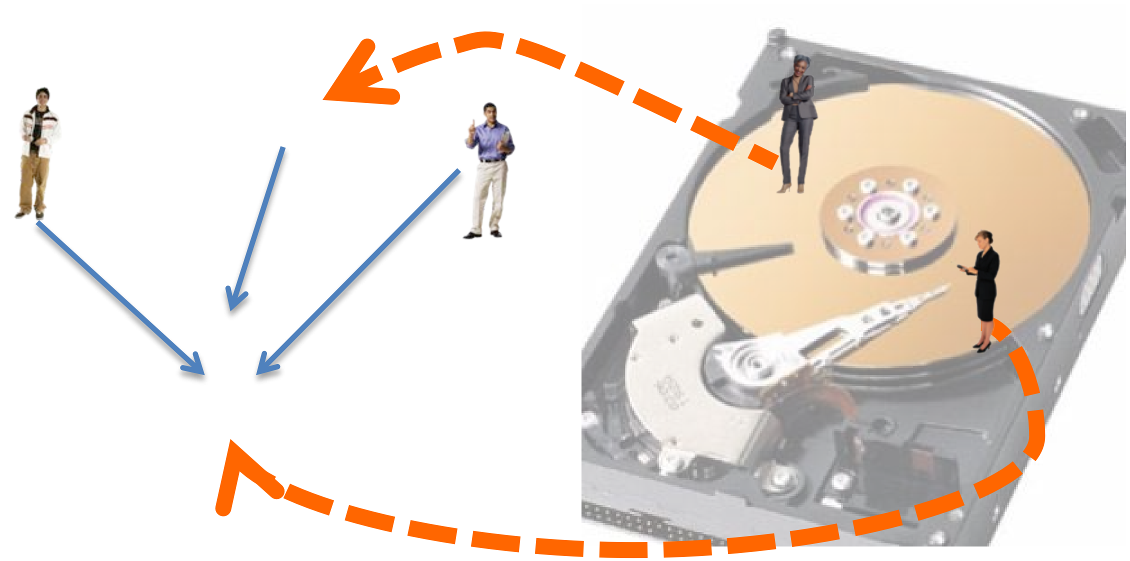


Key: Exploit non-volatile memory (starting with SSDs and HDs)

GraphChi – disk-based GraphLab

Challenge:

Random Accesses



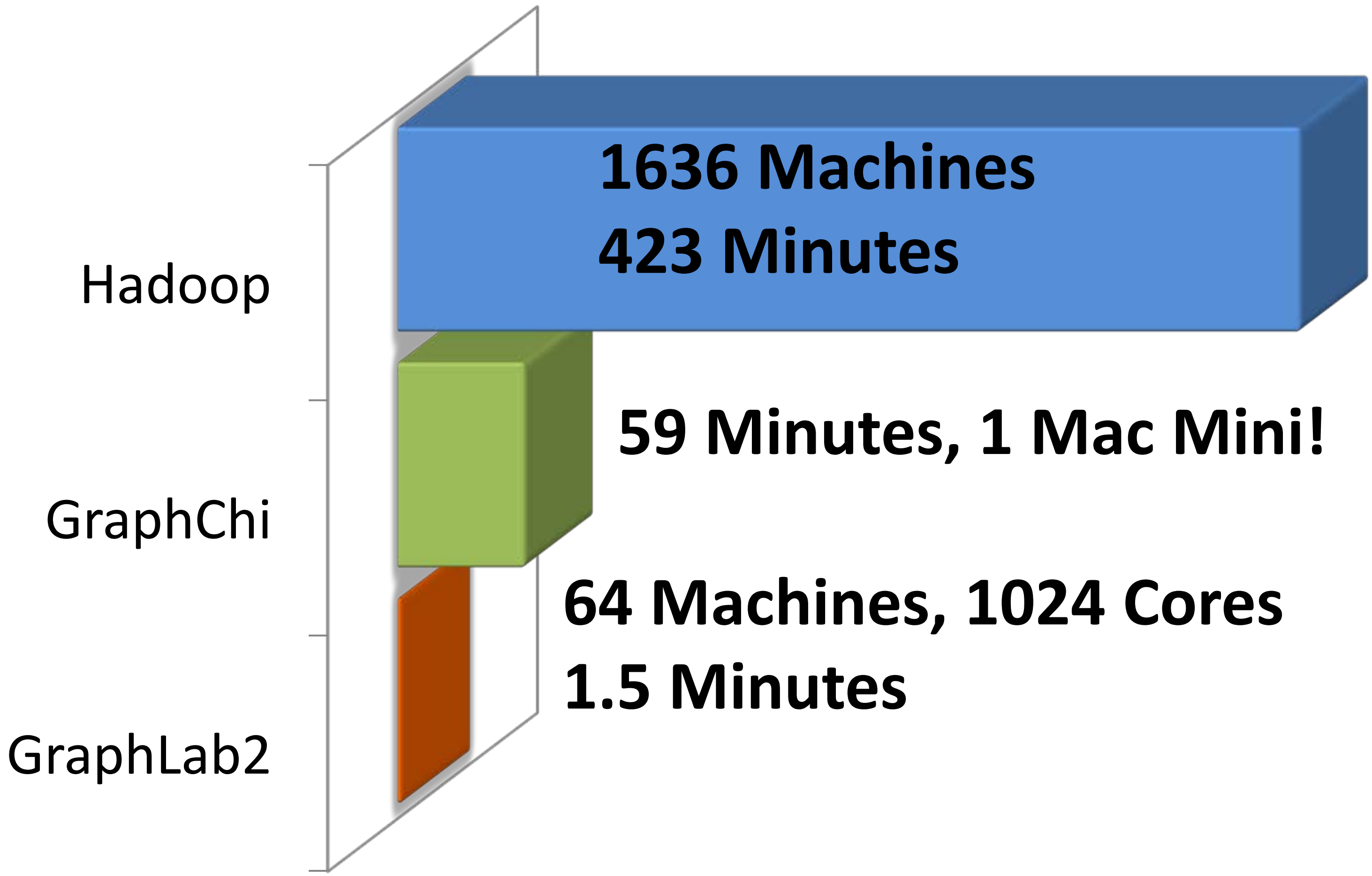
Novel GraphChi solution:

Parallel sliding windows method ➔ *minimizes number of random accesses*

Triangle Counting on Twitter Graph

40M Users
1.2B Edges

Total: 34.8 Billion Triangles

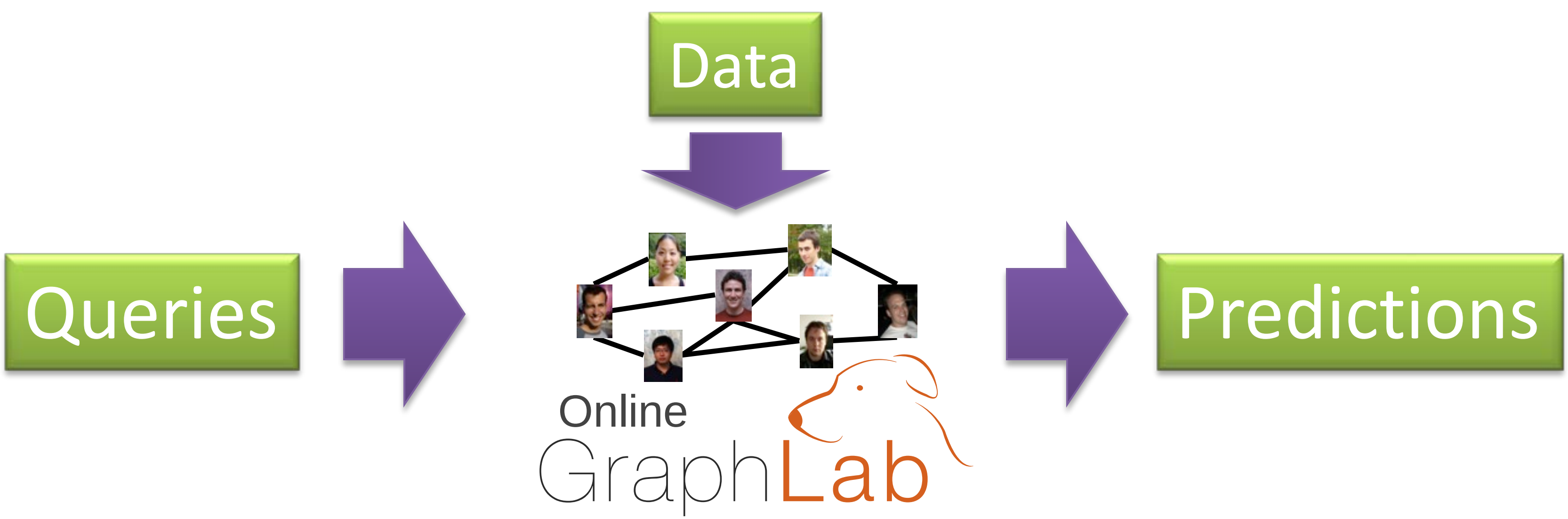


Hadoop results from [Suri & Vassilvitskii '11]

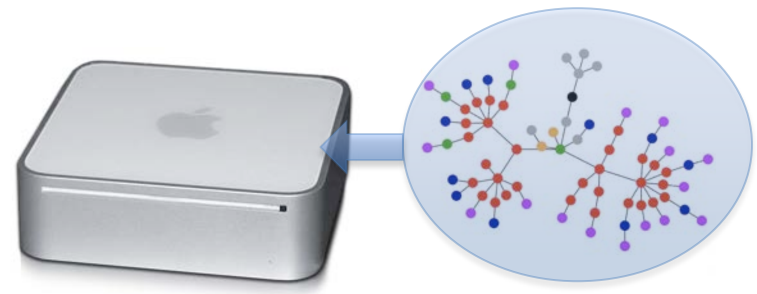
Next: Online GraphLab

Today, batch computation:

But, must continuously make predictions in presence of changing data (new users, friends, de-friending, …)



GraphChi: Streaming Graph Updates



Stream of Twitter social graph updates

Ingest 100,000 graph updates / sec

While **simultaneously computing** Pagerank on a Mac Mini, sustaining throughput of 200K updates/second

GraphLab

Release 2.1 available now

http://graphlab.org

Documentation... Code... Tutorials... (more on the way)

GraphChi 0.1 available now

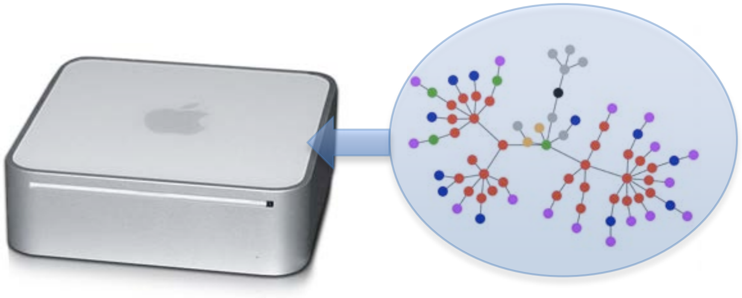
http://graphchi.org

GraphChi: Going small with GraphLab

Kyrola+al OSDI12

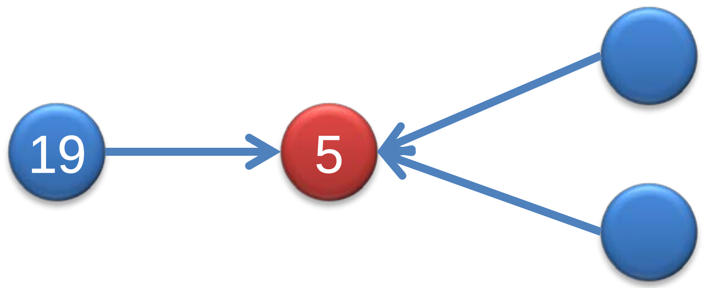
GraphLab

Solve huge problems on small or embedded devices?



Key: Exploit non-volatile memory (starting with SSDs and HDs)

Naive Graph Disk Layouts



- Symmetrized adjacency file with values,

vertex	in-neighbors	out-neighbors
5	**3**:2.3, **19**: 1.3, **49**: 0.65,...	**781**: 2.3, **881**: 4.2..
....		
19	**3**: 1.4, **9**: 12.1, ...	**5**: 1.3, 28: 2.2, ...

synchronize

Random write

- ... or with file index pointers

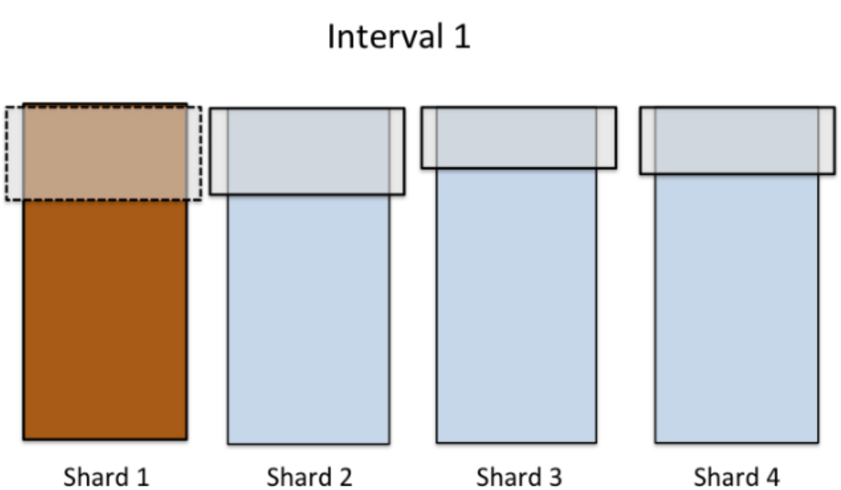
vertex	in-neighbor-ptr	out-neighbors
5	**3**: <u>881</u>, **19**: <u>10092</u>, **49**: <u>20763</u>,...	**781**: 2.3, **881**: 4.2..
....		
19	**3**: <u>882</u>, **9**: <u>2872</u>, ...	**5**: 1.3, 28: 2.2, ...

read

Random read/write

GraphChi – disk-based GraphLab

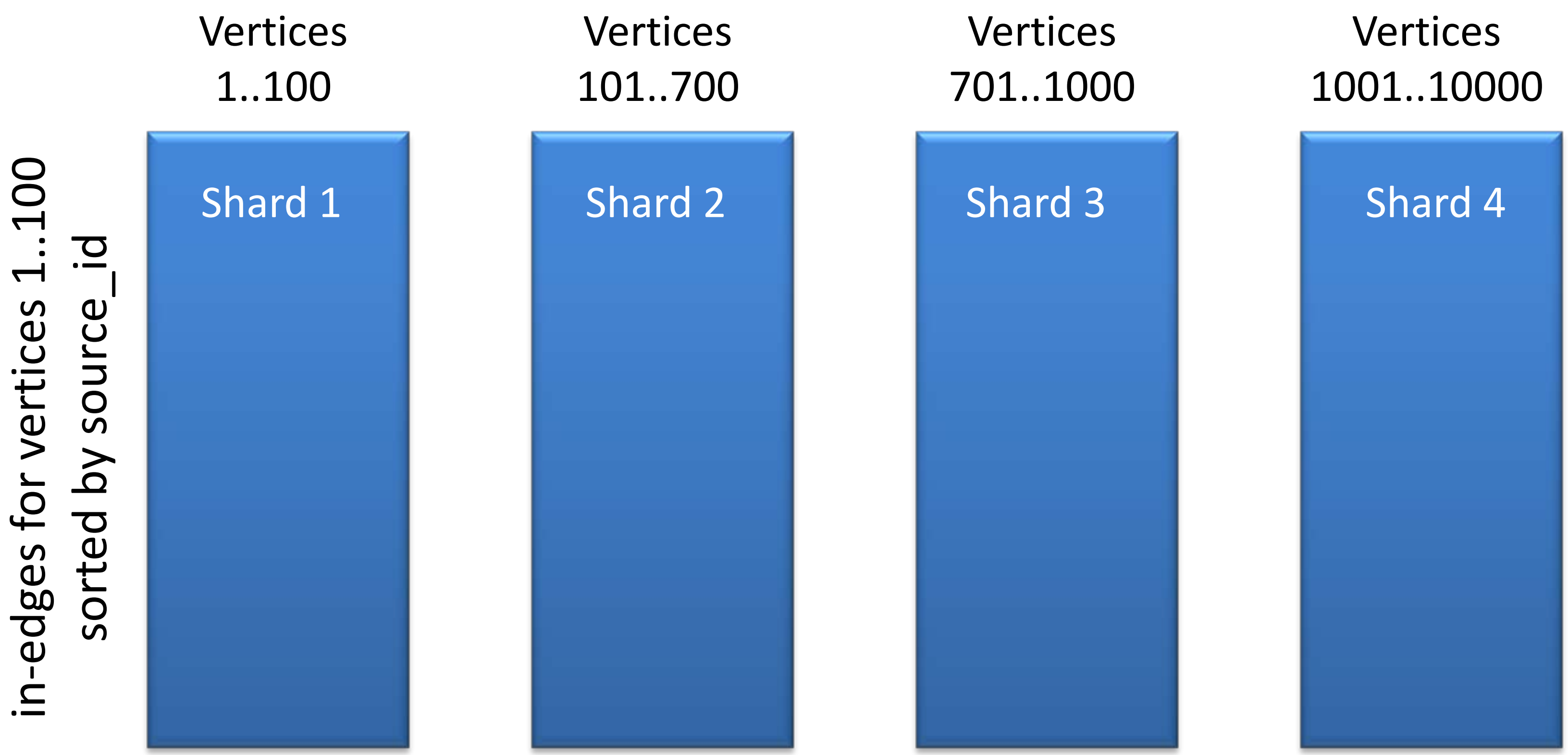
Novel Parallel Sliding Windows algorithm



- Fast ☺
- Solves tasks as large as current distributed systems
- Minimizes non-sequential disk accesses
  - Efficient on *both* SSD and hard-drive
- Parallel, asynchronous execution

Parallel Sliding Windows Layout

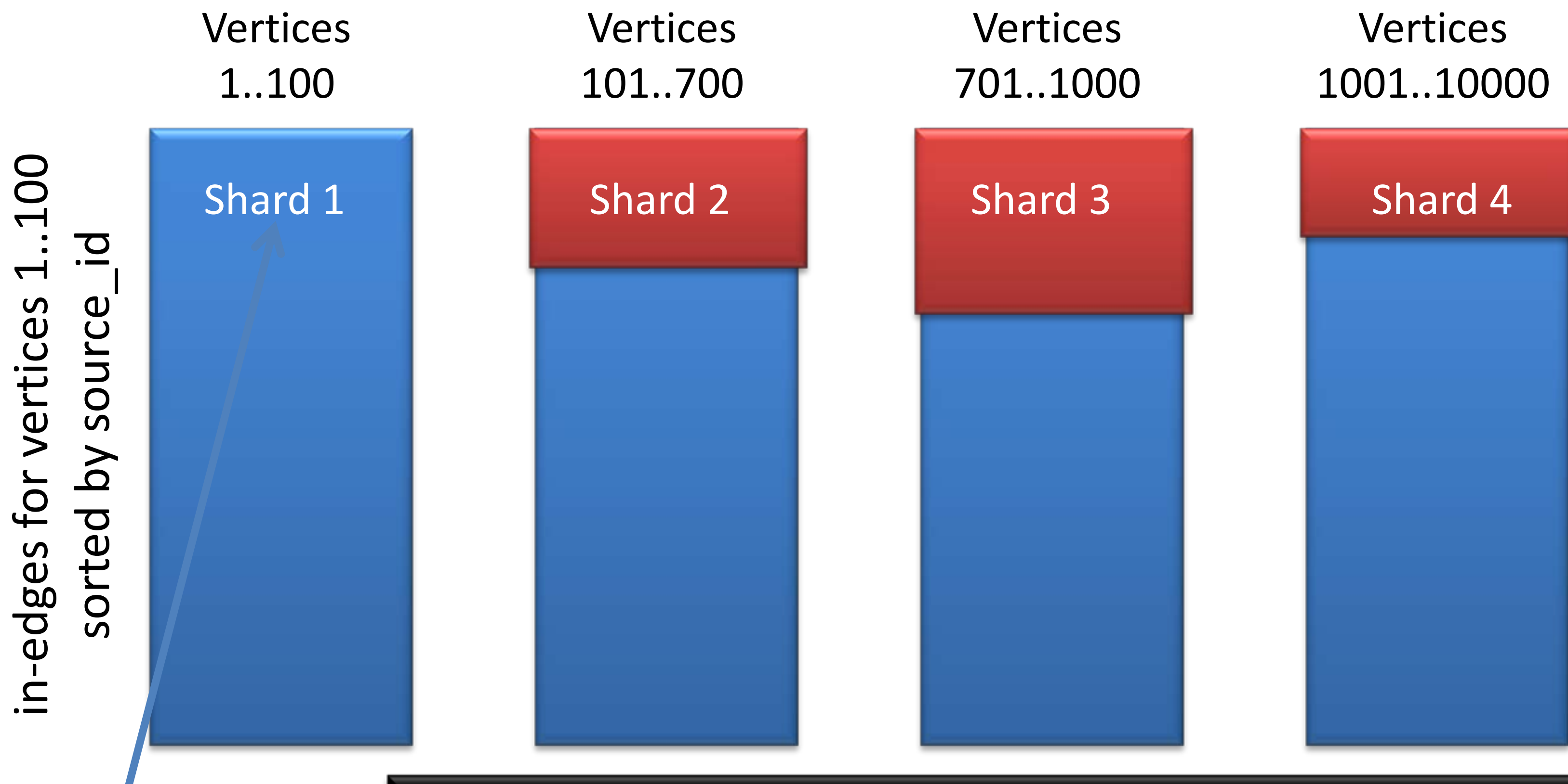
Shard: in-edges for subset of vertices; sorted by source_id



Shards small enough to fit in memory; balance size of shards

Parallel Sliding Windows Execution

Load subgraph for vertices 1..100



Load all in-edges in memory

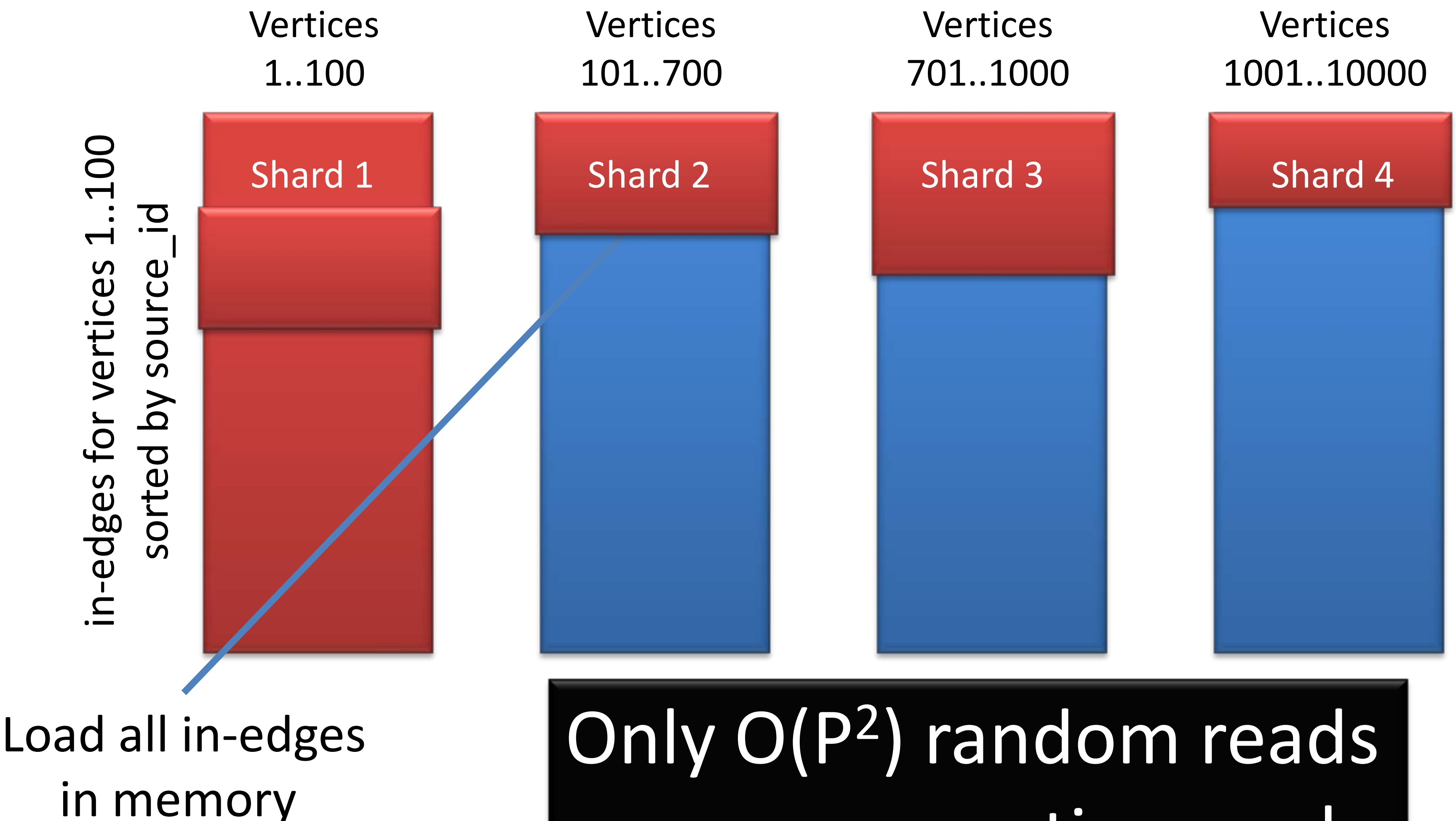
What about out-edges?

Arranged in sequence in other shards!

And sequential writes!

Parallel Sliding Windows Execution

Load subgraph for vertices 101..700

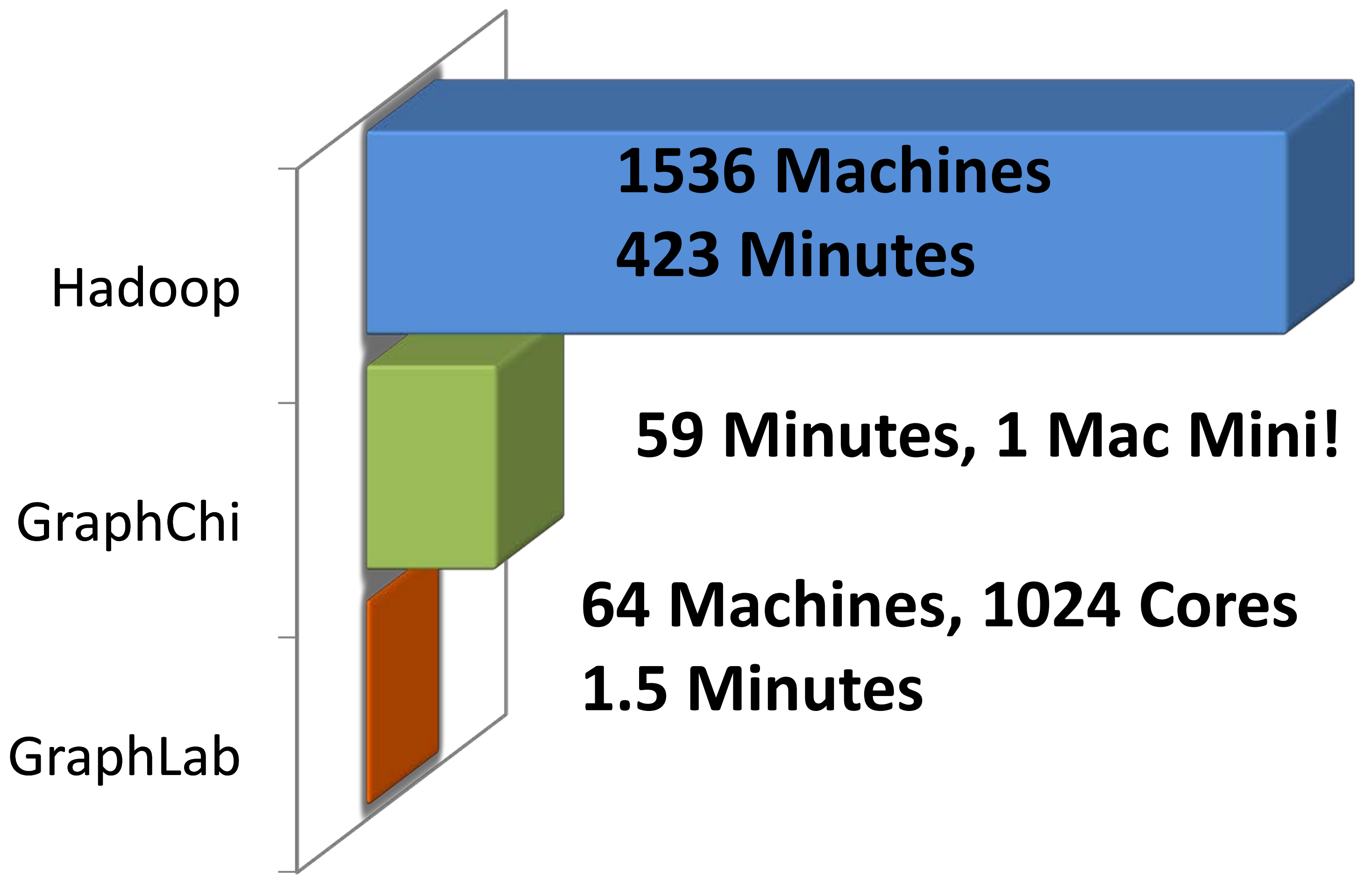


Only $O(P^2)$ random reads per pass on entire graph

Triangle Counting in Twitter Graph

40M Users
1.2B Edges

Total: 34.8 Billion Triangles



Hadoop results from [Suri & Vassilvitskii '11]

Apps & Performance

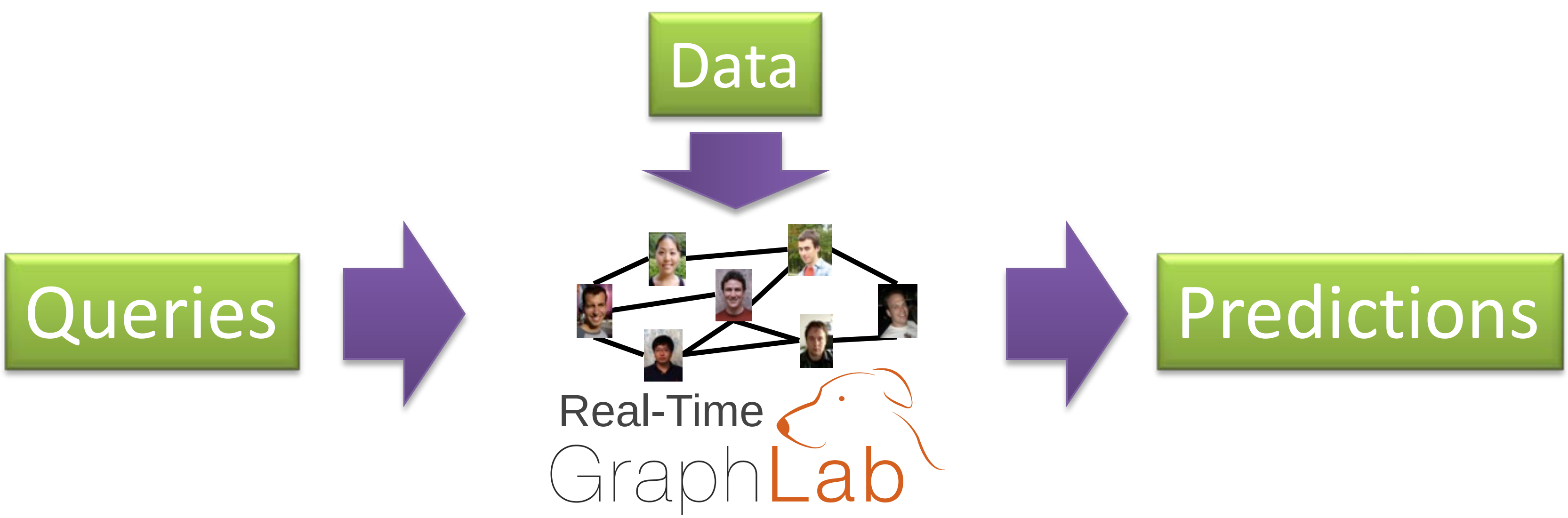
Application	Graph	Comparison	GraphChi on Mac Mini (SSD)
Pagerank (3 iter.)	Twitter-2010 (1.5B edges)	SPARK, 50 machines 8.1 min	13 min
Pagerank (100 iter.)	Uk-union (3.7B edges)	STANFORD GPS (PREGEL), 30 machines 144 min	581 min
WebGraph-Belief-Propagation (U Kang et al.)	Yahoo-web (6.7B edges)	PEGASUS, 100 machines 22 min	27 min
Matrix factorization (ALS) (10 iter.)	Netflix movies (99M edges)	GRAPHLAB, 8-core machine 4.7 min	9.8 min
Triangle counting	Twitter-2010	HADOOP, 1636 machines 423 min	45 min

Node, comparison results do not include time to transfer the data to cluster, or the time to load the graph from disk.

Goal: Real-Time GraphLab

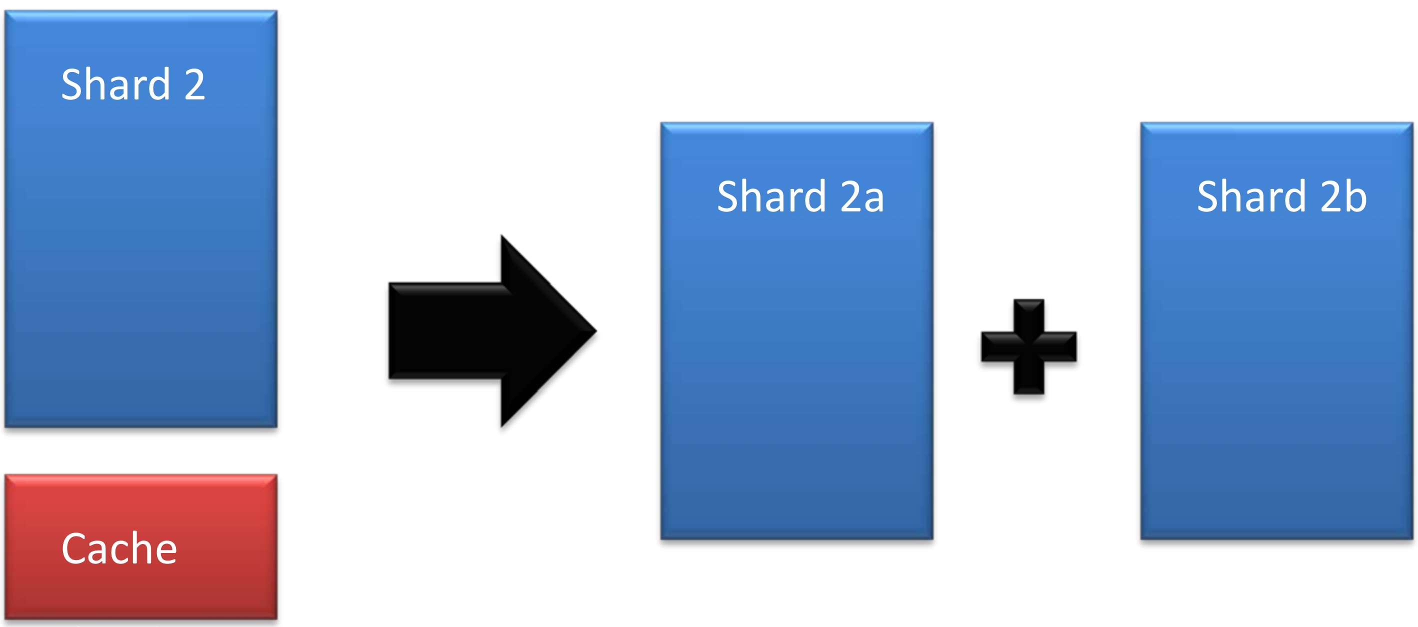
Today, batch computation:

But, must continuously make predictions in presence of changing data (new users, friends, de-friending, sensors…)

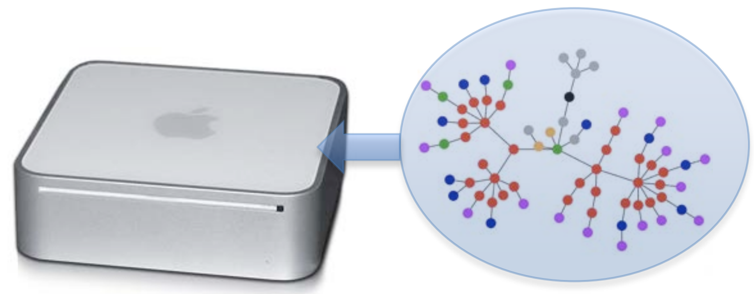


GraphChi with Streaming Graphs

- Keep edge additions and deletions in-memory cache, per shard
- When cache too large, split shard
  - Or merge as needed
  - Resort shard in memory, since small enough



Streaming Graph Updates

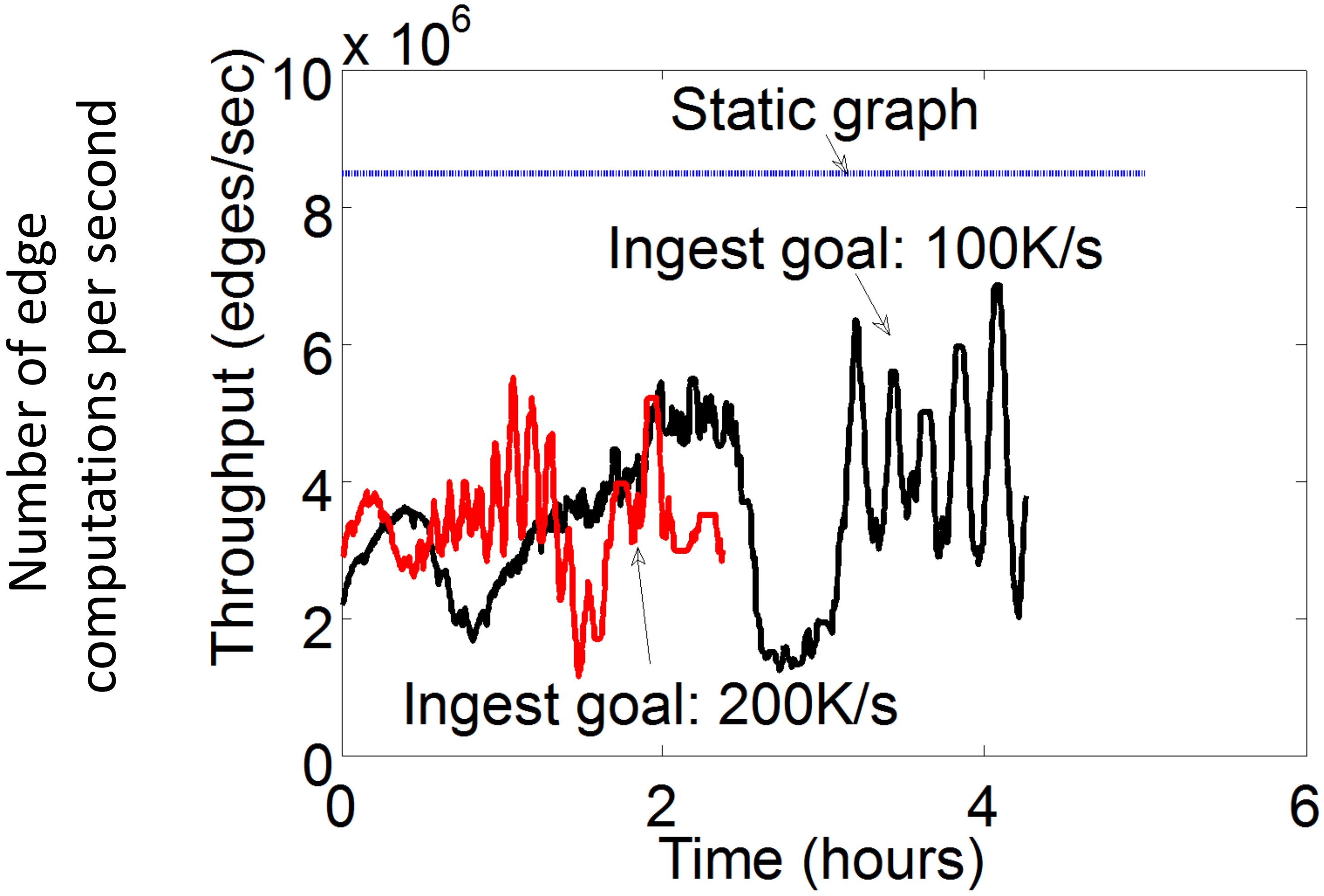


Stream of Twitter social graph updates

Ingest 100,000 graph updates / sec

While **simultaneously computing** Pagerank on a Mac Mini, sustaining throughput of 200K updates/second

GraphChi: Dynamic Graphs Evaluation



Mac Mini / SSD: streaming of Twitter graph (1.5B edges) from the hard drive with gapped rate of 100K or 200K edges/sec.