GraphLab

Large-Scale Machine Learning and Graphs

Carlos Guestrin

PHASE 1

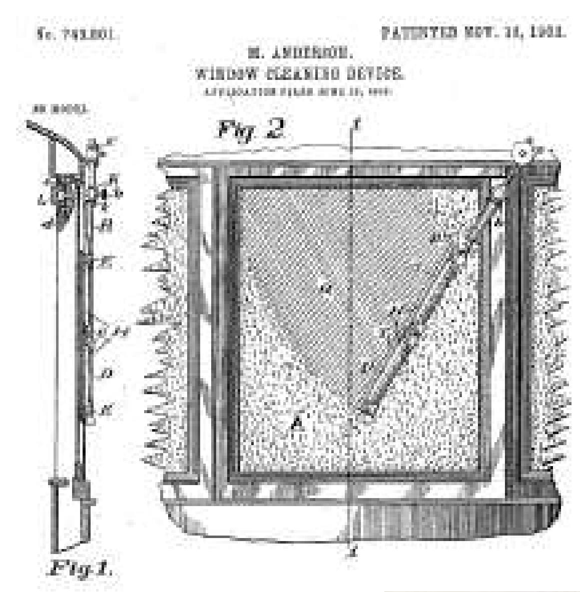
POSSIBILITY

PHASE 2

SCALABILITY

PHASE 3

USABILITY

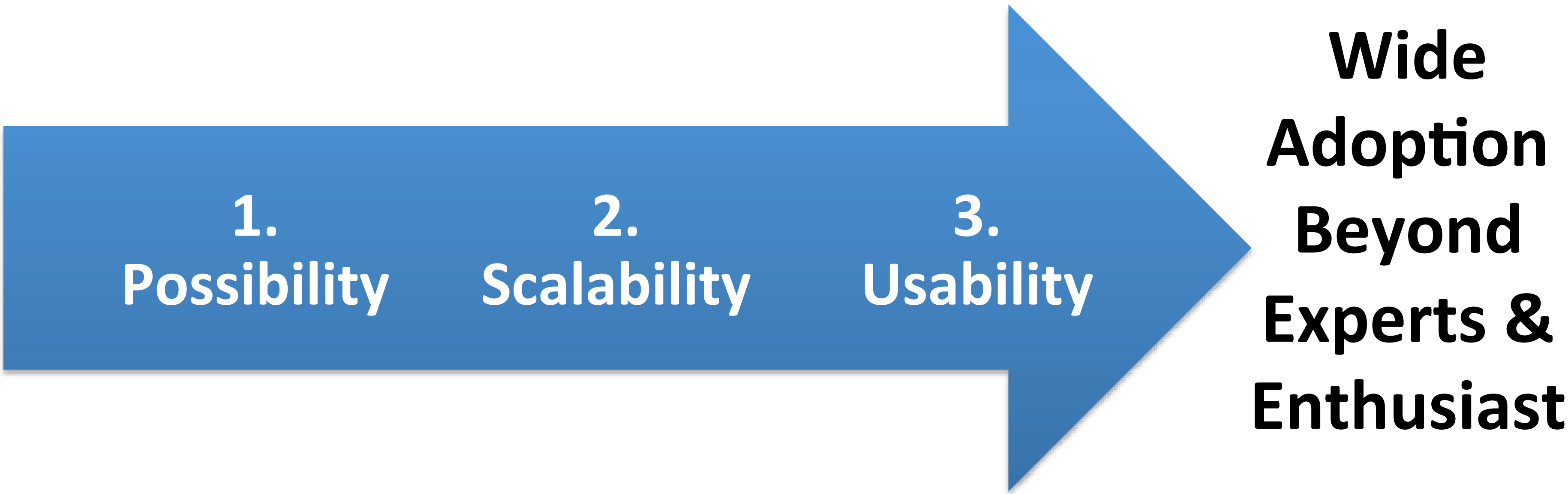

M. ANDERSON.
WINDOW CLEANING DEVICE.
Fig 2
Fig.1.


a realm all its own... Cadillac


ABS


TEXACO
TWO GUNS

Three Phases in Technological Development



Machine Learning PHASE 1

POSSIBILITY

Rosenblatt 1957


YAHOO!
1998 Winter Games
results, schedules, news
MegaMarketing
BENEFITS EXPOSED!
Academy Award
Nominations
Search
options
Yahoo! Chat with Wall Street guru Jim Cramer, supermodel Frederique
Yellow Pages - People Search - Maps - Classifieds - Personals - Chat - Free Email
Shopping - My Yahoo! - News - Sports - Weather - Stock Quotes - more...
Arts and Humanities
Architecture, Photography, Literature...
Business and Economy [Xtra!]
Companies, Finance, Employment...
Computers and Internet [Xtra!]
Internet, WWW, Software, Multimedia...
Education
Universities, K-12, College Entrance...
Entertainment [Xtra!]
Cool Links, Movies, Music, Humor...
Government
Military, Politics [Xtra!], Law, Taxes...
Health [Xtra!]
Medicine, Drugs, Diseases, Fitness...
News and Media [Xtra!]
Current Events, Magazines, TV, Newspapers...
Recreation and Sports [Xtra!]
Sports, Games, Travel, Autos, Outdoors...
Reference
Libraries, Dictionaries, Phone Numbers...
Regional
Countries, Regions, U.S. States...
Science
CS, Biology, Astronomy, Engineering...
Social Science
Anthropology, Sociology, Economics...
Society and Culture
People, Environment, Religion...


Google!
Search the web using Google!
Google Search
I'm feeling lucky
Index contains ~25 million pages (soon to be much bigger)
About Google!
Stanford Search Linux Search
Get Google! updates monthly!
your e-mail
Subscribe
Archive
Copyright ©1997-8 Stanford University

Machine Learning PHASE 2

SCALABILITY

Needless to Say, We Need Machine Learning for Big Data

6 Billion
Flickr Photos

28 Million
Wikipedia Pages

1 Billion
Facebook Users

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?

MapReduce for Data-Parallel ML

Excellent for large data-parallel tasks!

Data-Parallel

MapReduce

Feature Extraction

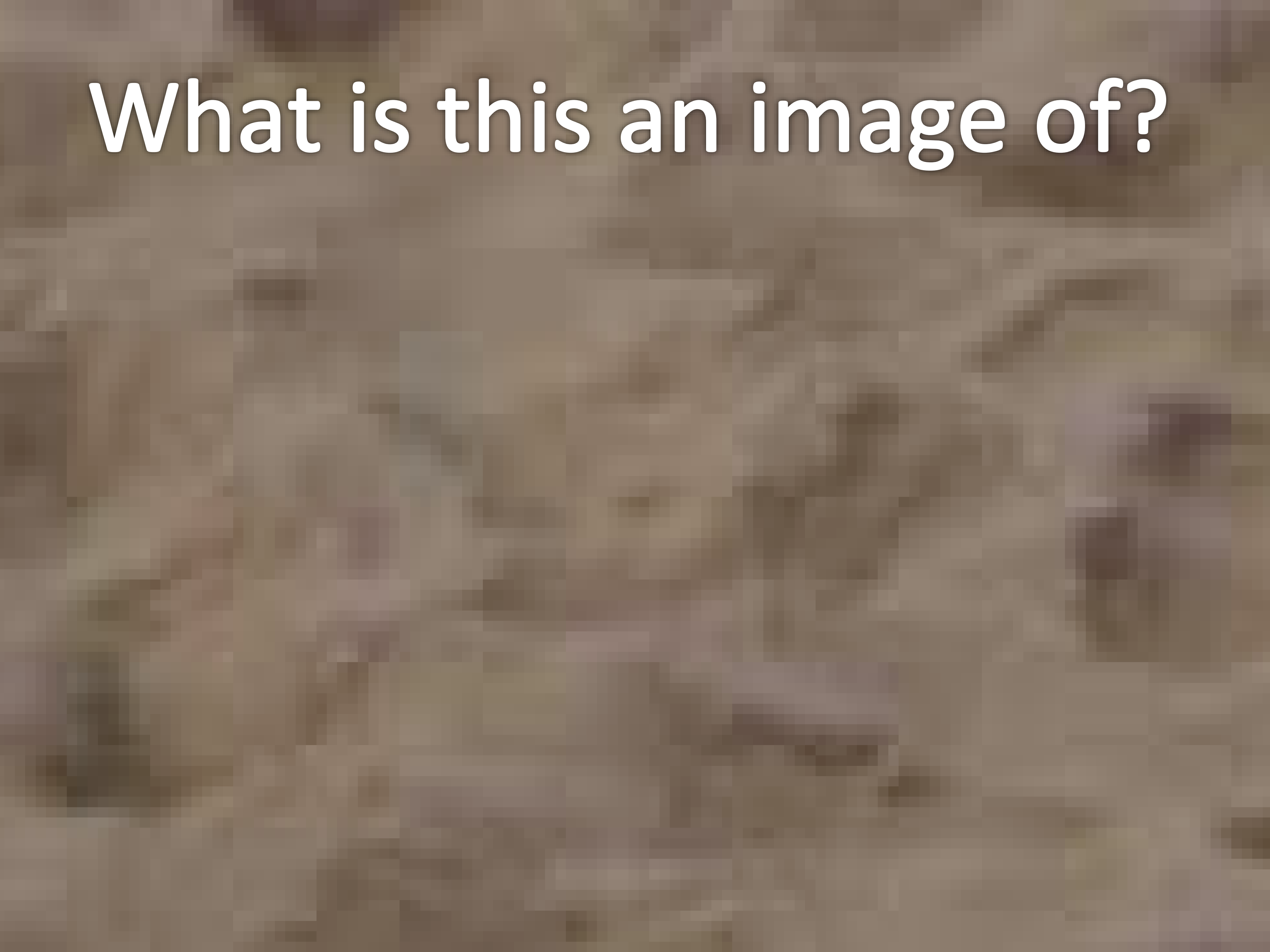
Cross Validation

Computing Sufficient Statistics

Is there more to Machine Learning

?

What is this an image of?



It’s next to this…

The Power of Dependencies

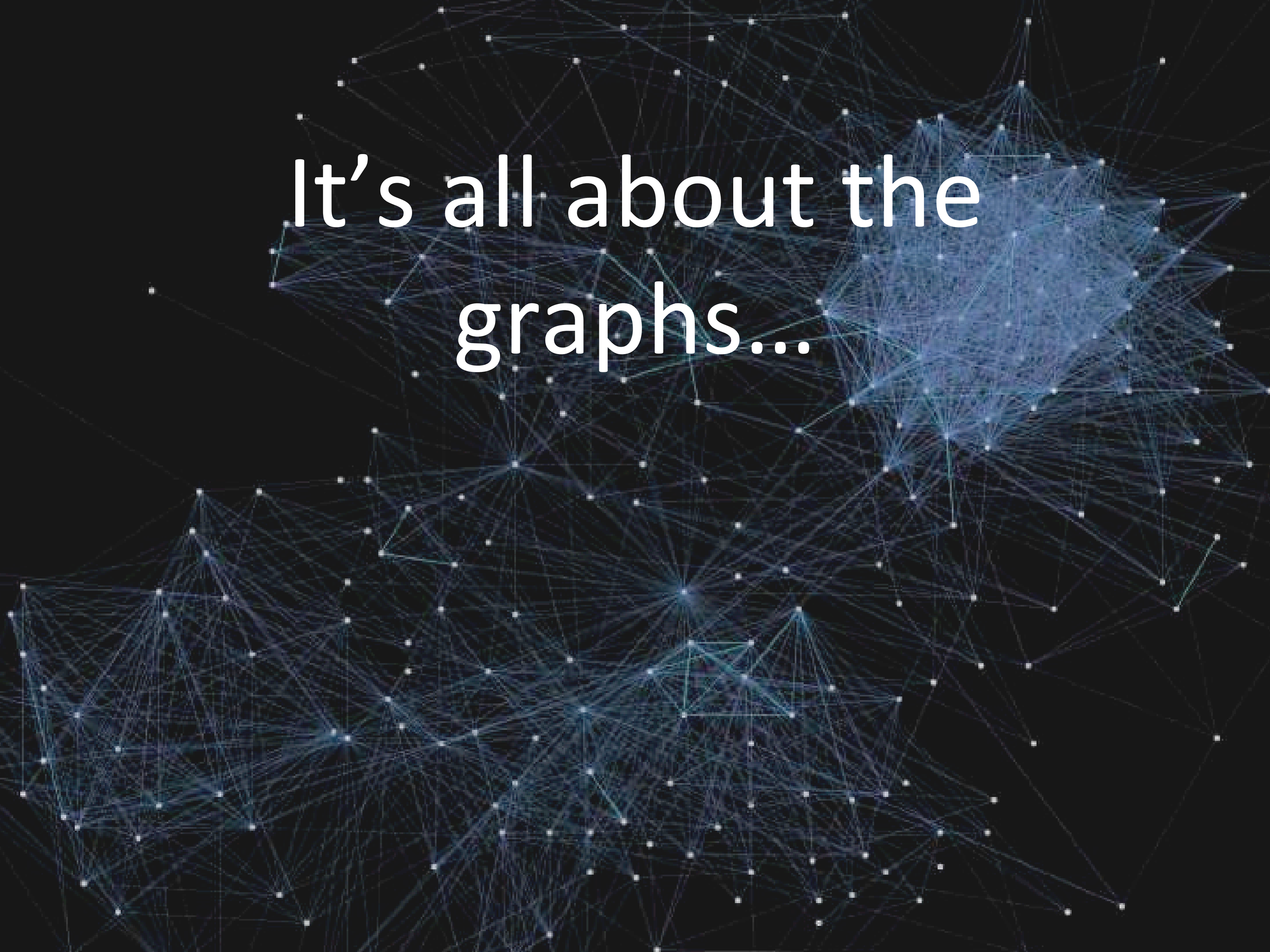
where the value is!

Flashback to 1998

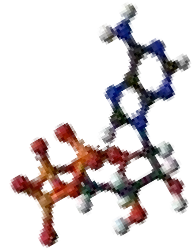
Why?

First Google advantage:
a **Graph Algorithm** & a **System to Support** it!

It’s all about the graphs…



Social Media **Science** **Advertising** **Web**



- **Graphs** encode the **relationships** between:

People Products Ideas

Facts Interests

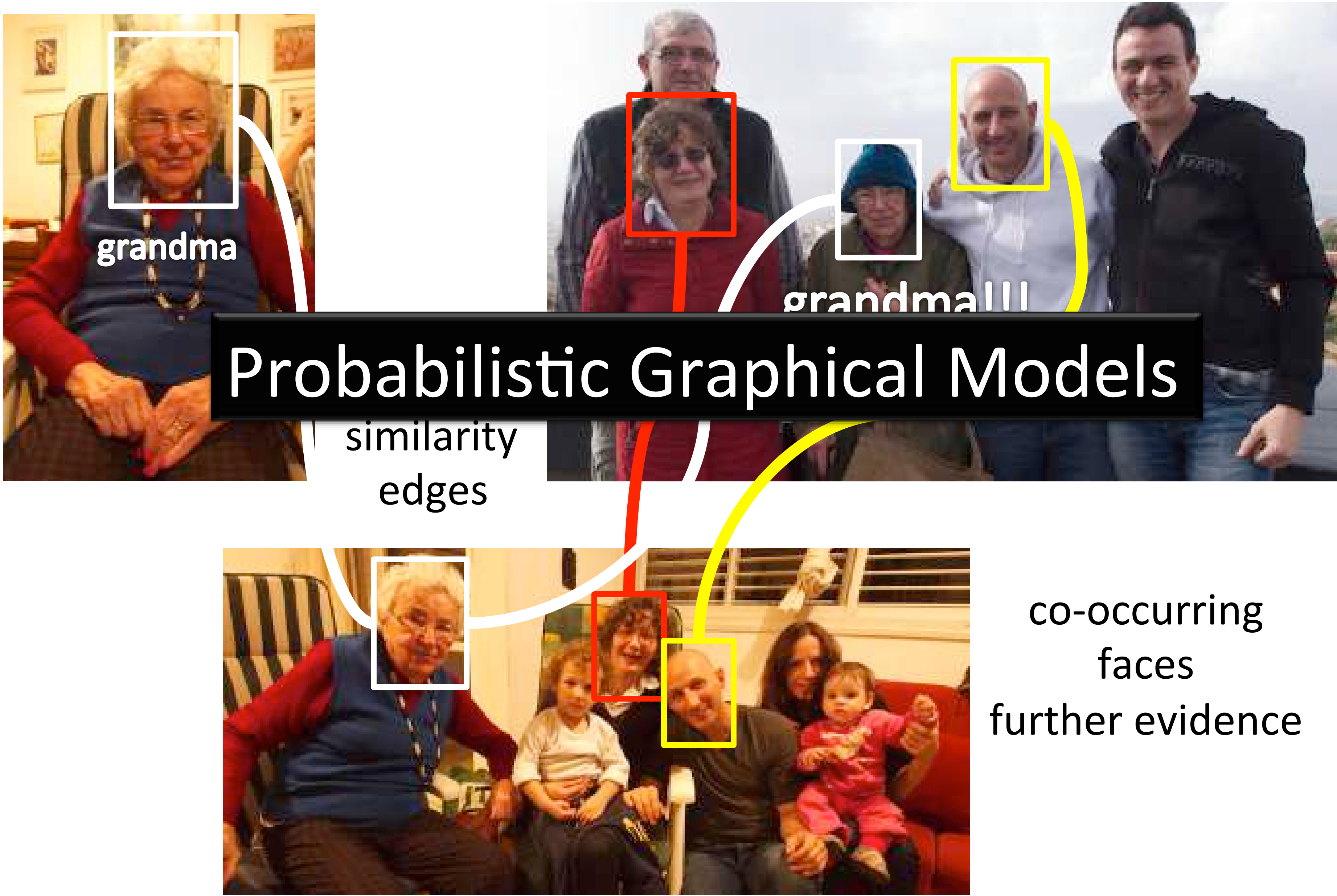
- **Big: 100 billions** of **vertices** and **edges** and rich metadata
  - Facebook (10/2012): 1B users, 144B friendships
  - Twitter (2011): 15B follower edges

Examples of Graphs in Machine Learning

Label a Face and Propagate

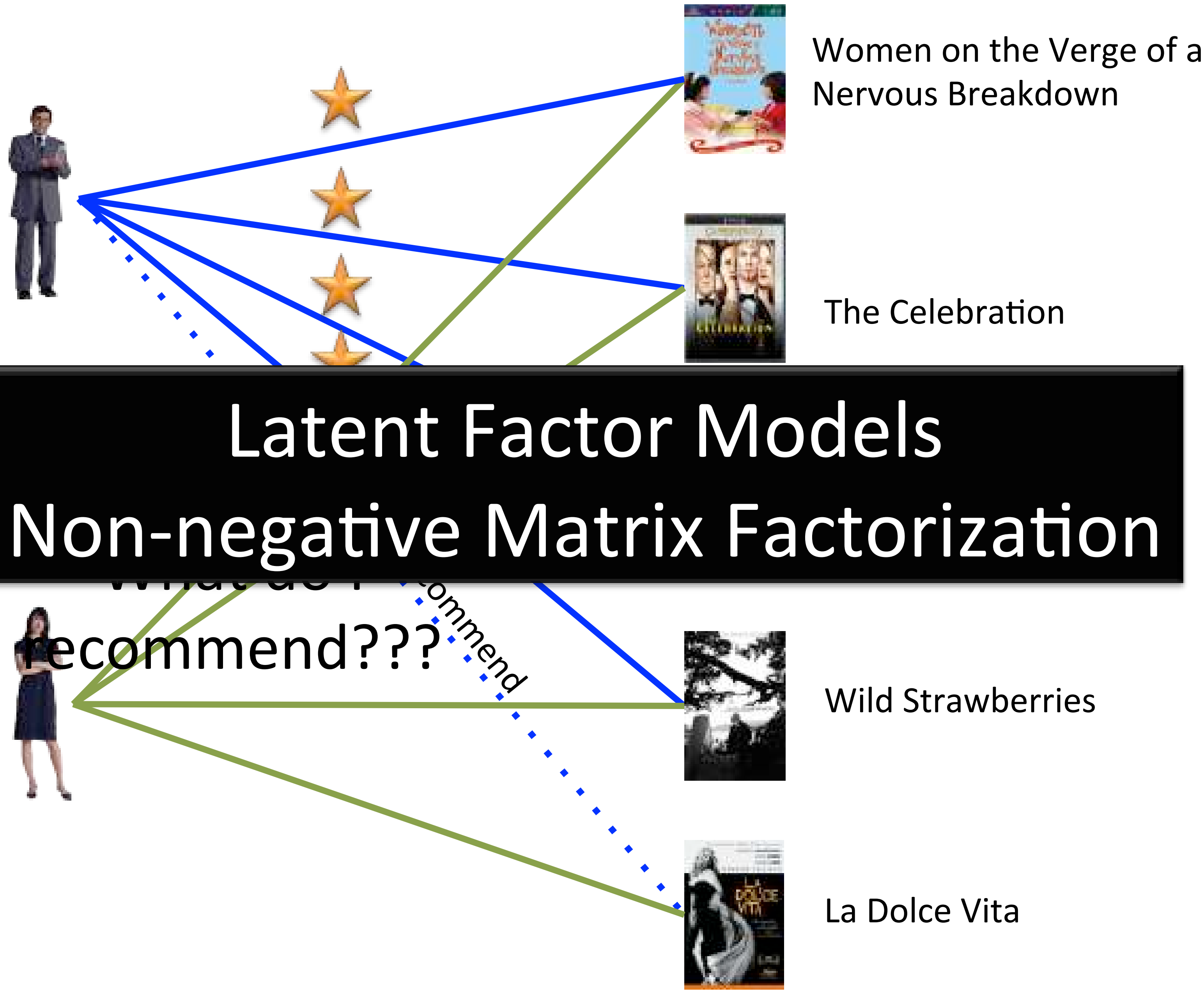
Pairwise similarity not enough…

Propagate Similarities & Co-occurrences for Accurate Predictions



Probabilistic Graphical Models

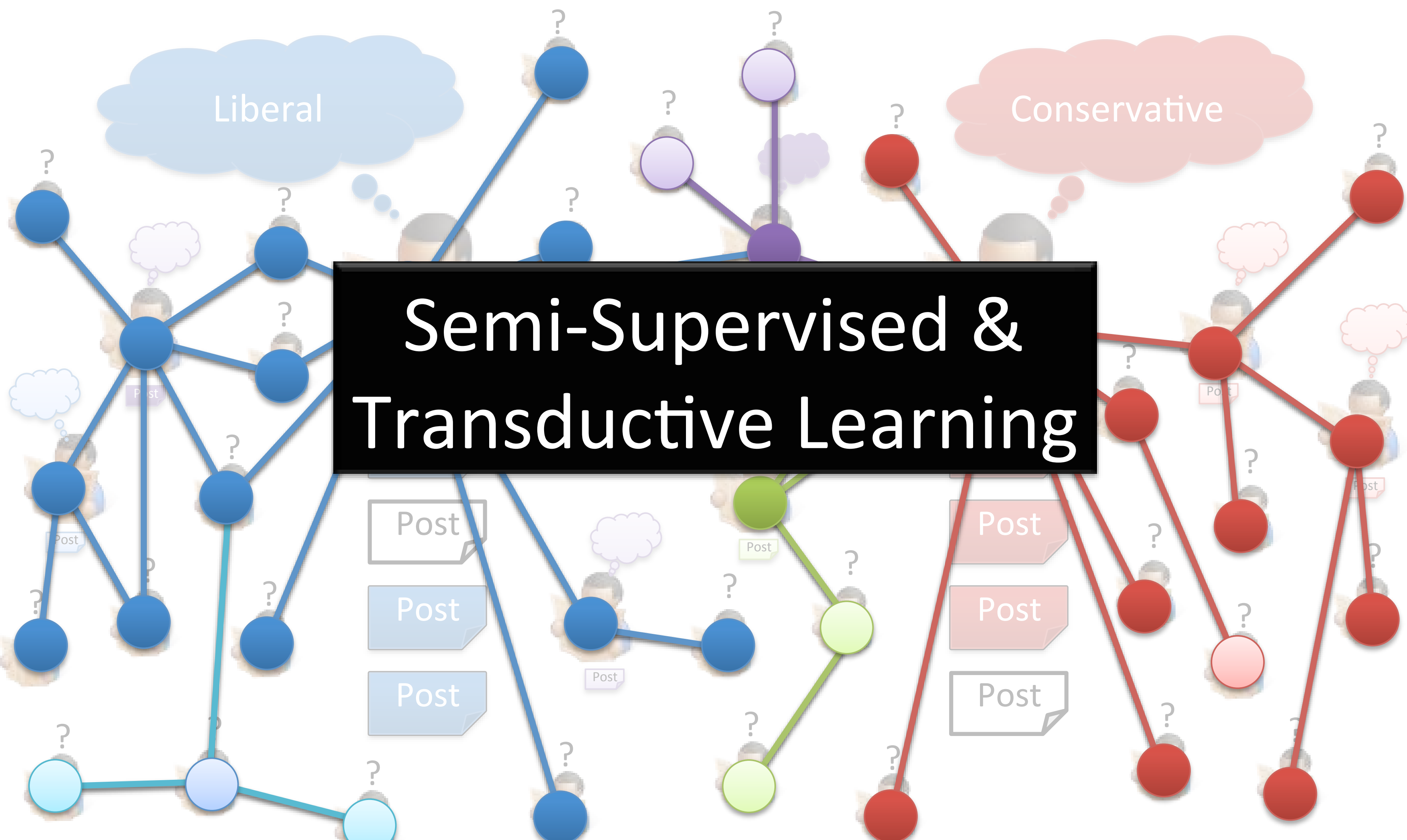
Collaborative Filtering: Exploiting Dependencies



Latent Factor Models

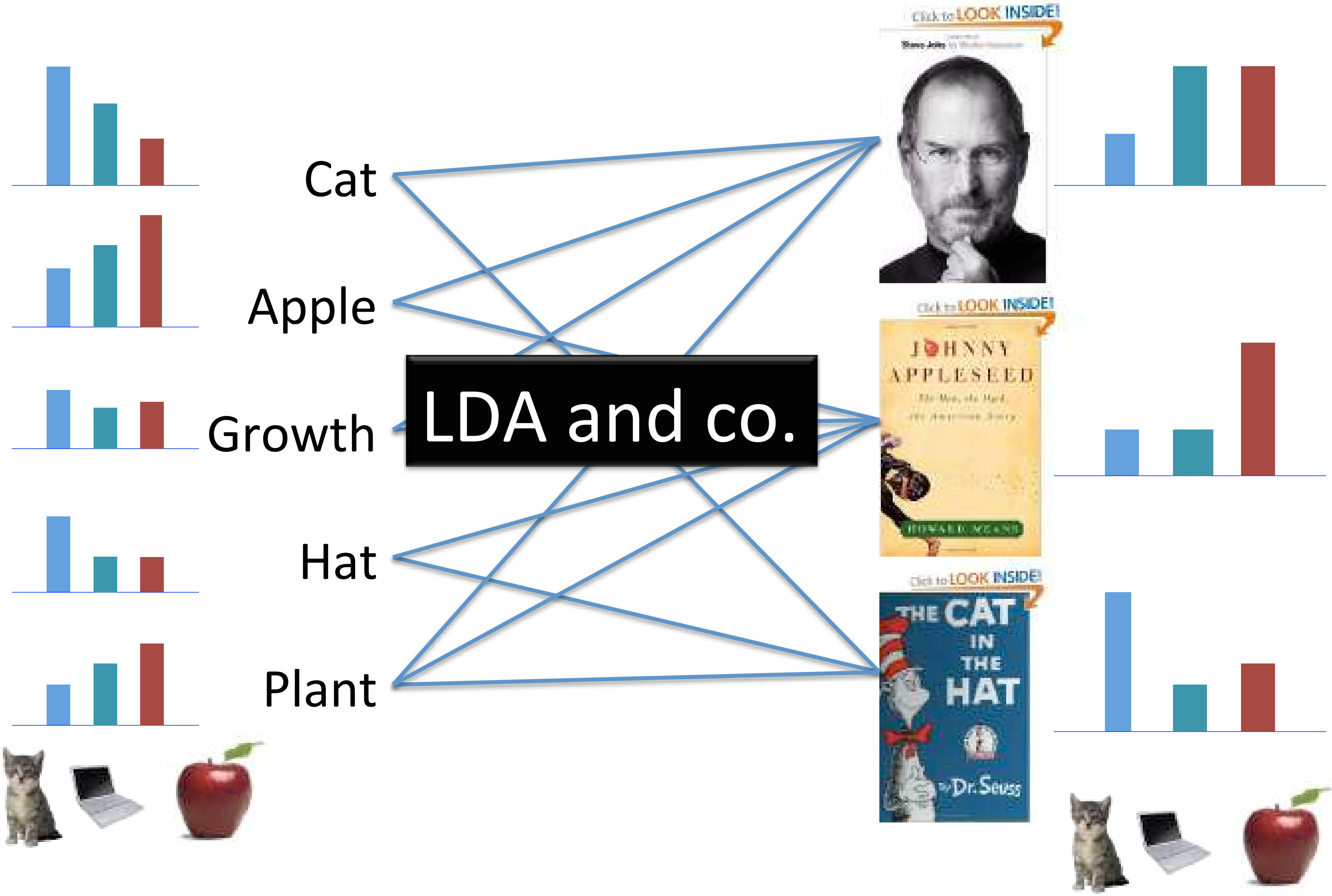
Non-negative Matrix Factorization

Estimate Political Bias

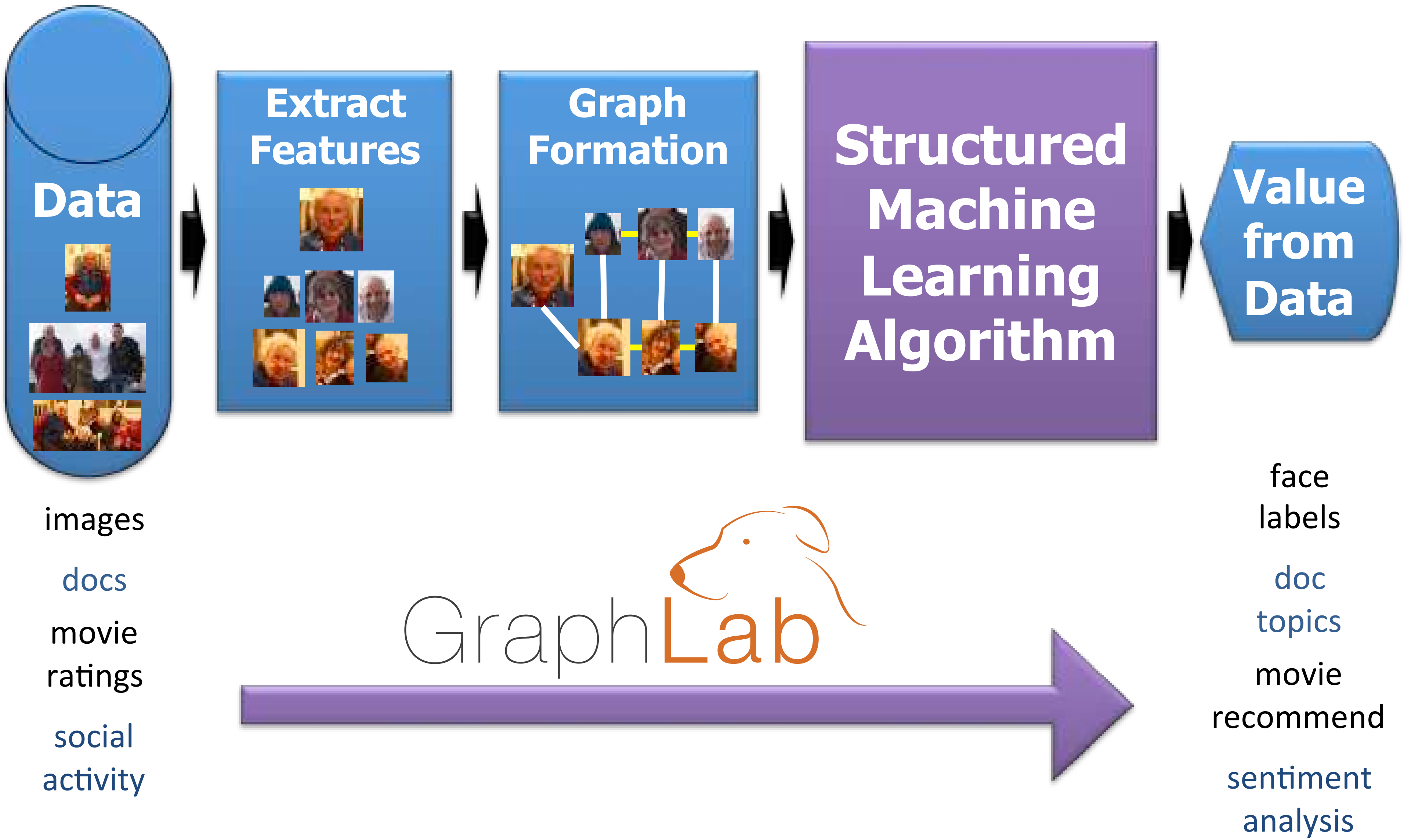


Semi-Supervised & Transductive Learning

Topic Modeling



Machine Learning Pipeline



ML Tasks Beyond Data-Parallelism

Map Reduce

Feature Extraction

Cross Validation

Computing Sufficient Statistics

Graphical Models
Gibbs Sampling
Belief Propagation
Variational Opt.

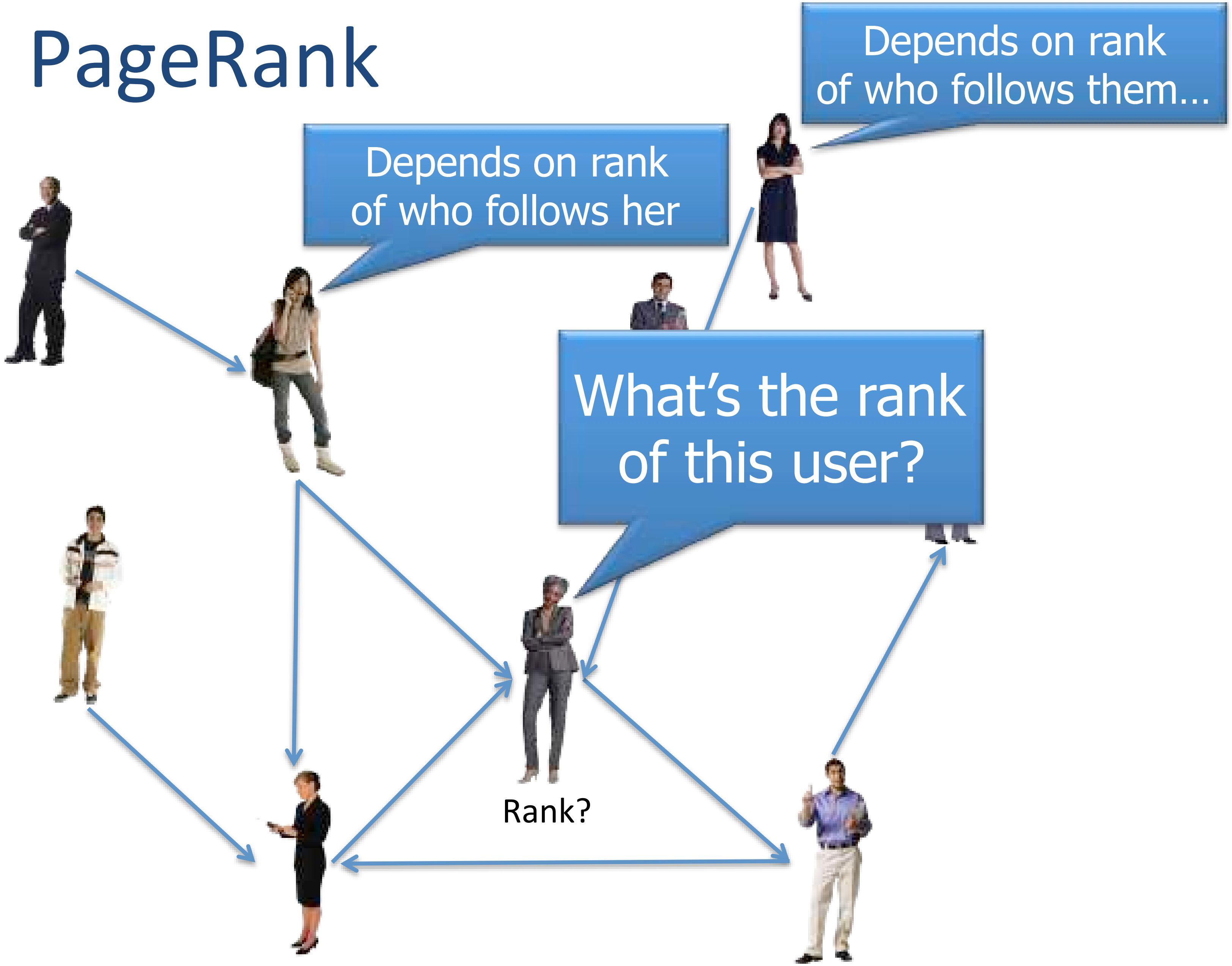
Semi-Supervised Learning
Label Propagation
CoEM

Collaborative Filtering
Tensor Factorization

Graph Analysis
PageRank
Triangle Counting

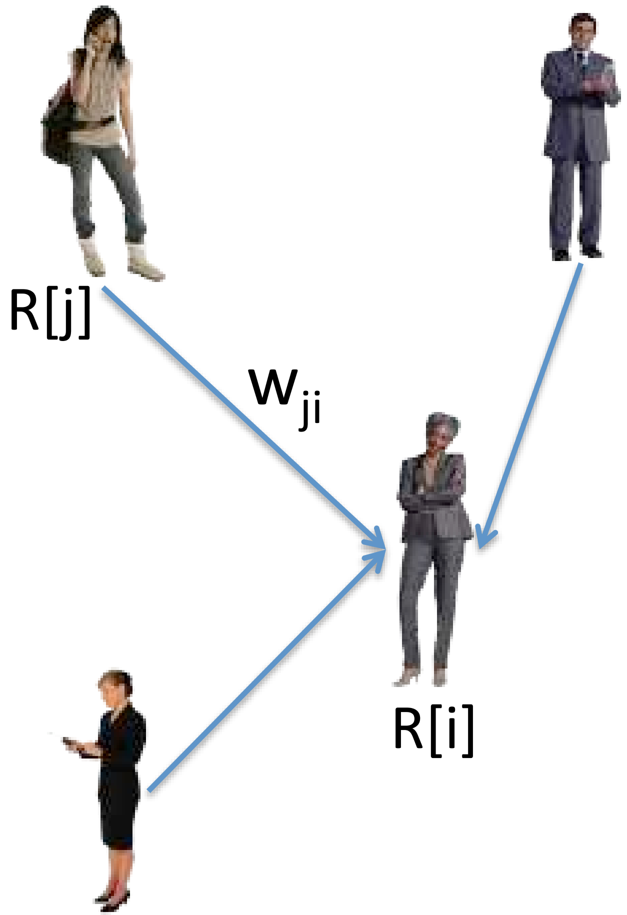
Example of a Graph-Parallel Algorithm

PageRank



Loops in graph → Must iterate!

PageRank Iteration



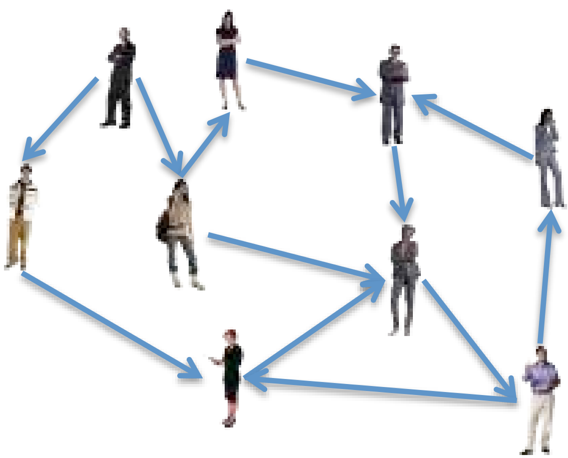
Iterate until convergence:
"My rank is weighted average of my friends' ranks"

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

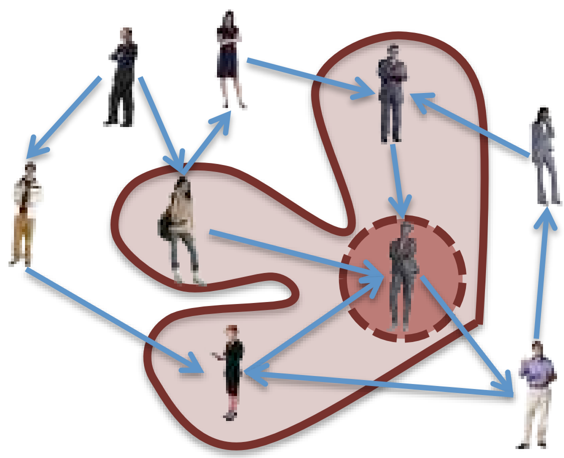
- α is the random reset probability
- w_{ji} is the prob. transitioning (similarity) from j to i

Properties of Graph Parallel Algorithms

Dependency Graph



Local Updates



Iterative Computation

The Need for a New Abstraction

- Need: Asynchronous, Dynamic Parallel Computations

Data-Parallel ⟷ Graph-Parallel

Map Reduce

Feature Extraction

Cross Validation

Computing Sufficient Statistics

GraphLab

Carnegie Mellon

Graphical Models
Gibbs Sampling
Belief Propagation
Variational Opt.

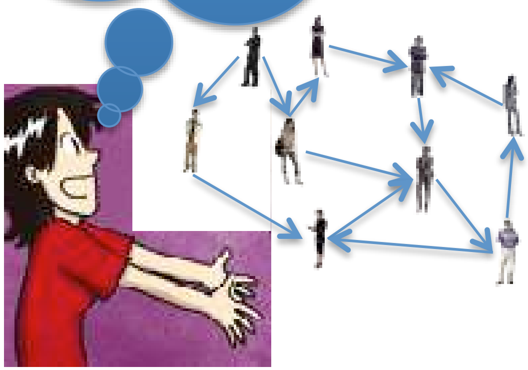
Semi-Supervised Learning
Label Propagation
CoEM

Collaborative Filtering
Tensor Factorization

Data-Mining
PageRank
Triangle Counting

The **GraphLab** Goals

Know how to solve ML problem on 1 machine



Efficient parallel predictions

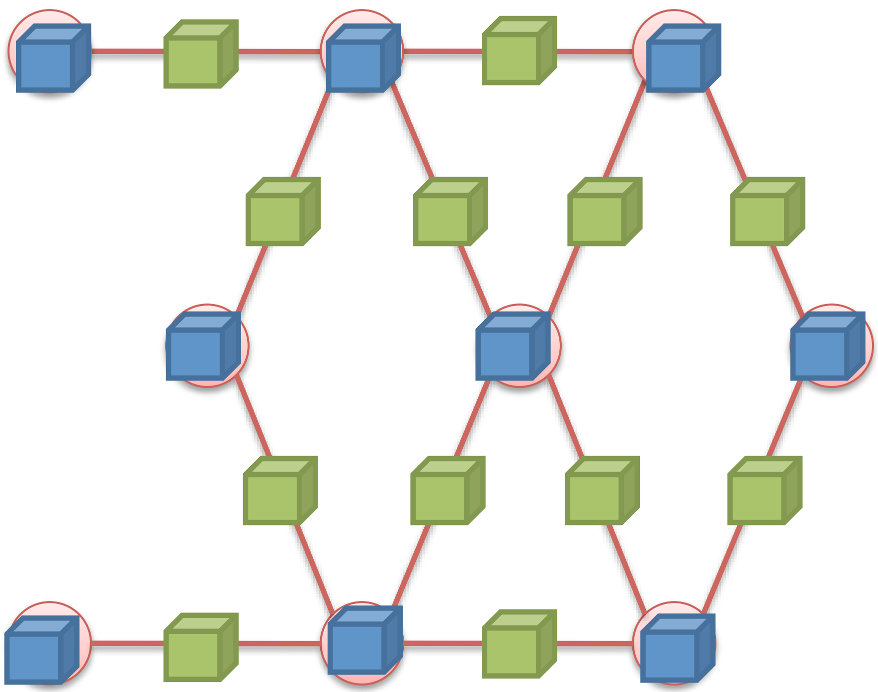
GraphLab
Carnegie Mellon

1

POSSIBILITY

Data Graph

Data associated with vertices and edges



Graph:
- Social Network

Vertex Data:
- User profile text
- Current interests estimates

Edge Data:
- Similarity weights

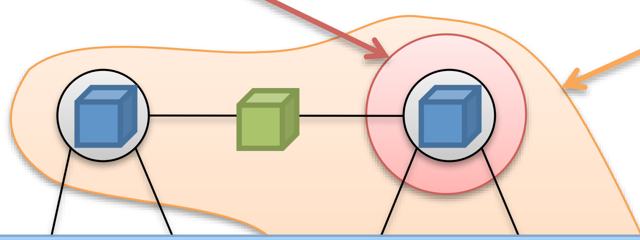
How do we *program* **graph** computation?

“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

$(R[i], w_{ij}, R[j]) \leftarrow$ scope;

Update function applied (asynchronously) in parallel until convergence

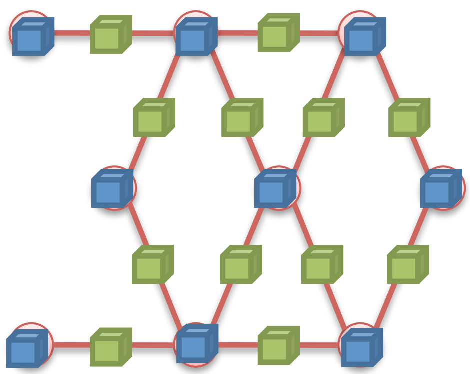
Many schedulers available to prioritize computation

}

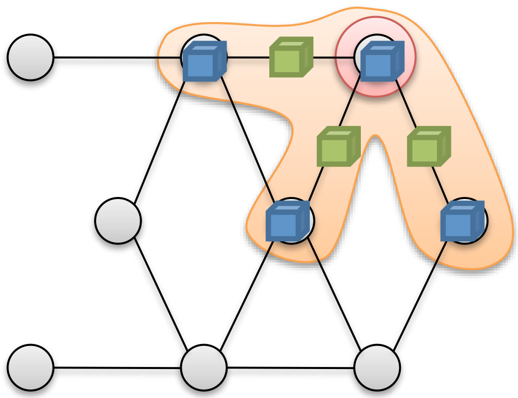
Dynamic computation

The GraphLab Framework

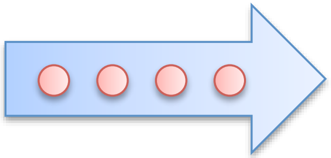
Graph Based
Data Representation



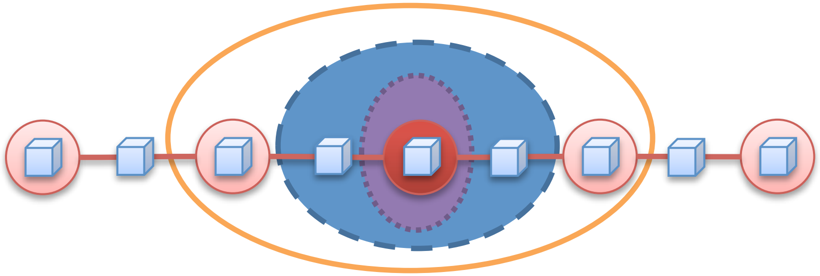
Update Functions
User Computation



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

GraphLab 1

- ML algorithms as vertex programs
- Asynchronous execution and consistency models

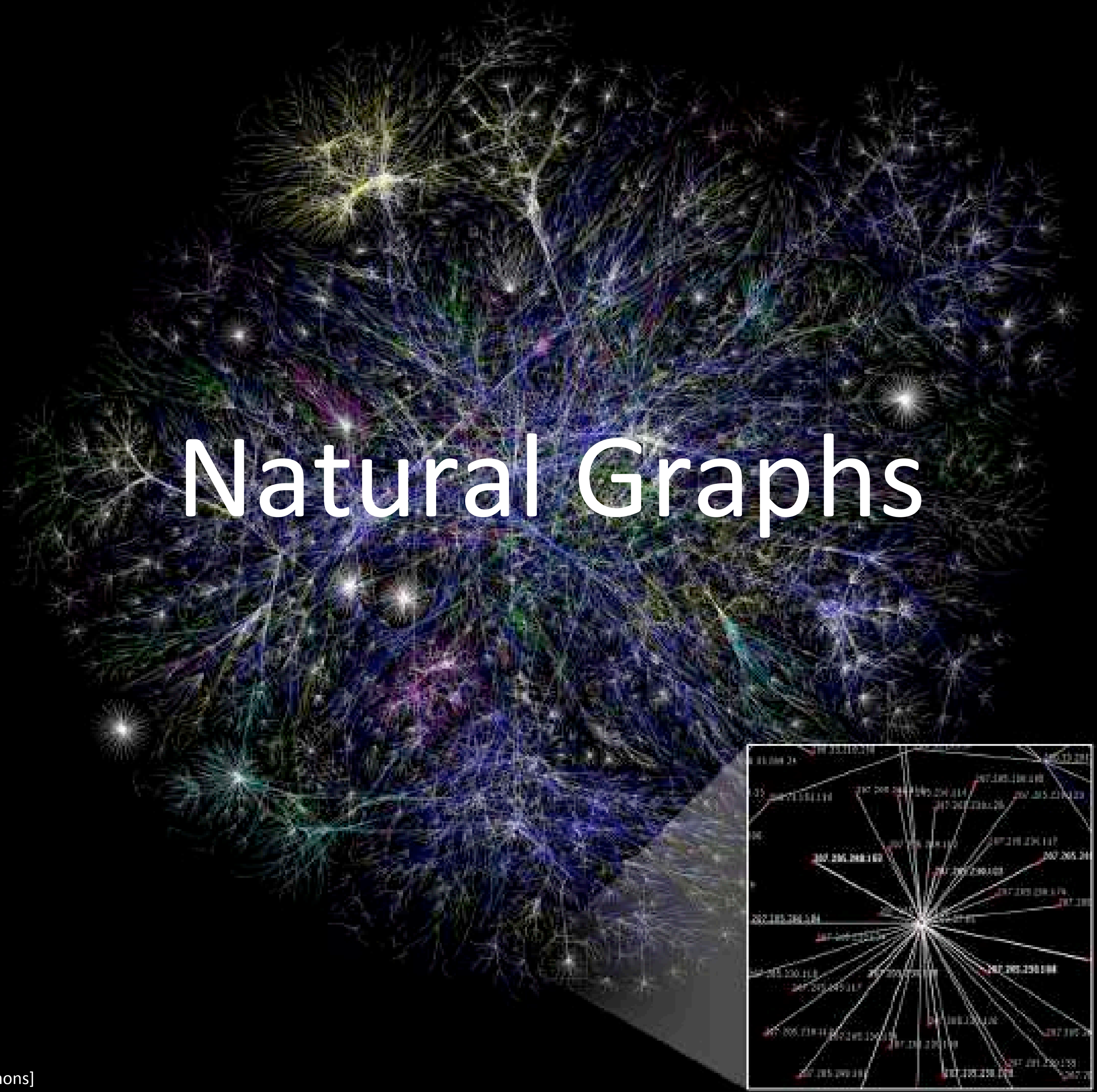
Thus far…

GraphLab 1 provided exciting scaling performance

But…

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

Natural Graphs



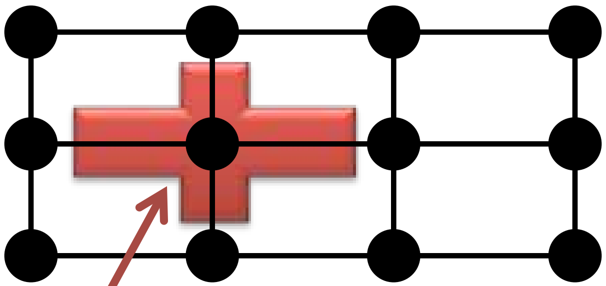
[Image from WikiCommons]

Problem:

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

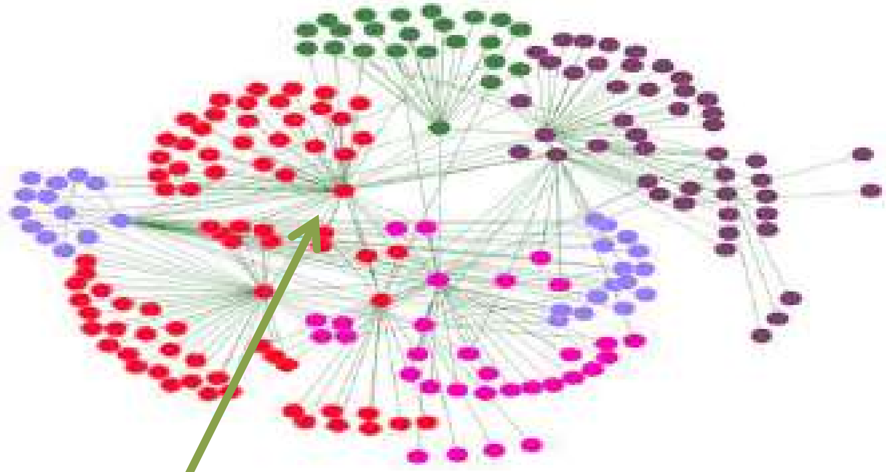
Achilles Heel: Idealized Graph Assumption

Assumed…



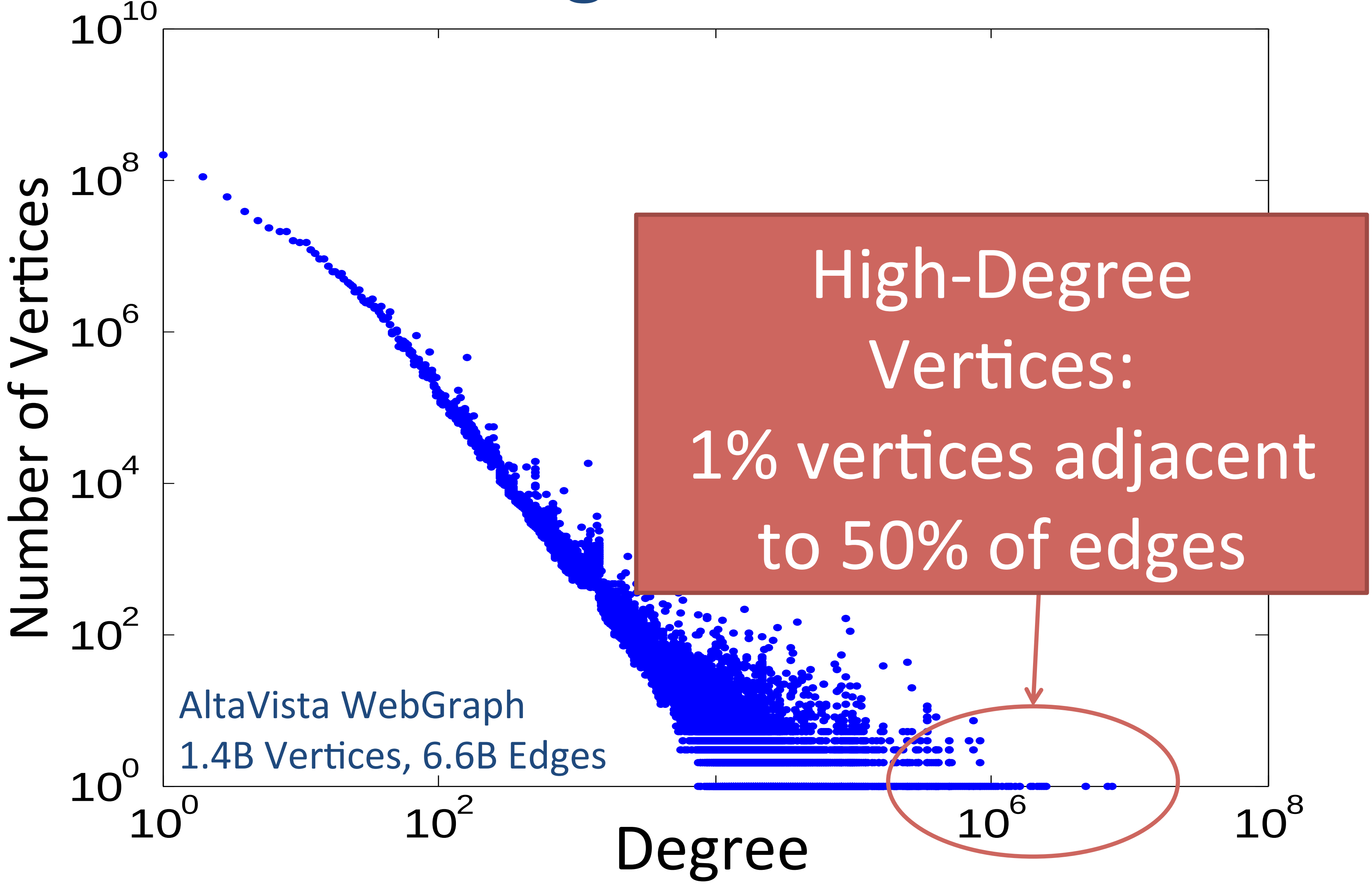
Small degree → Easy to partition

But, Natural Graphs…



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

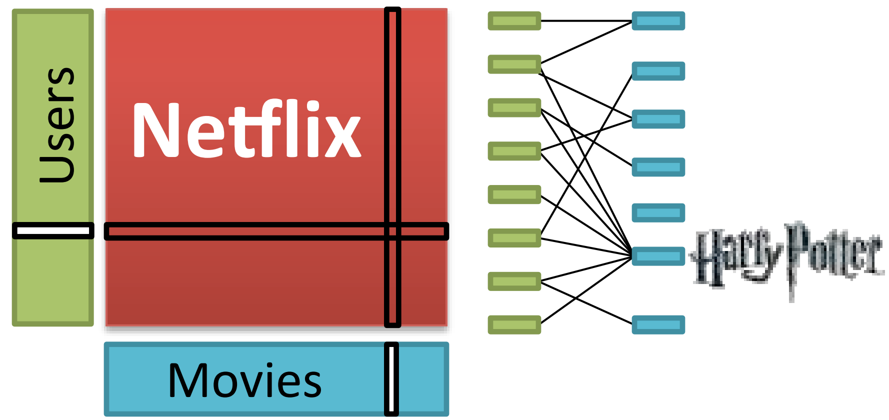
Power-Law Degree Distribution



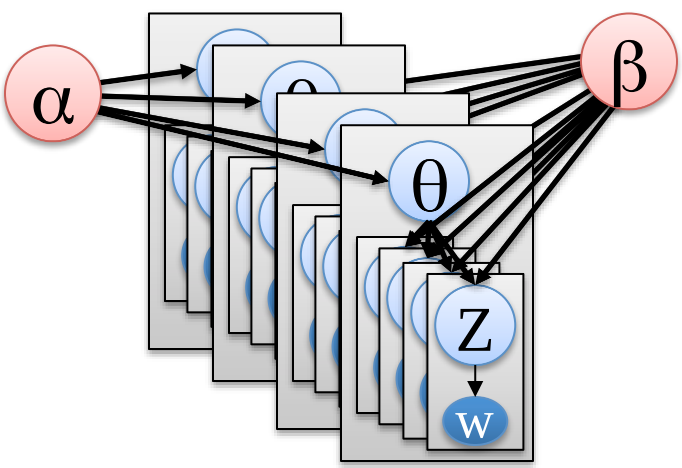
High Degree Vertices are Common

“Social” People

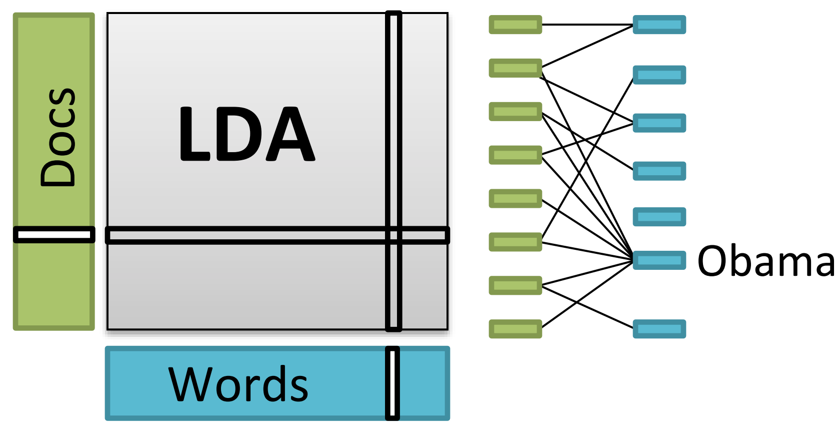
Popular Movies



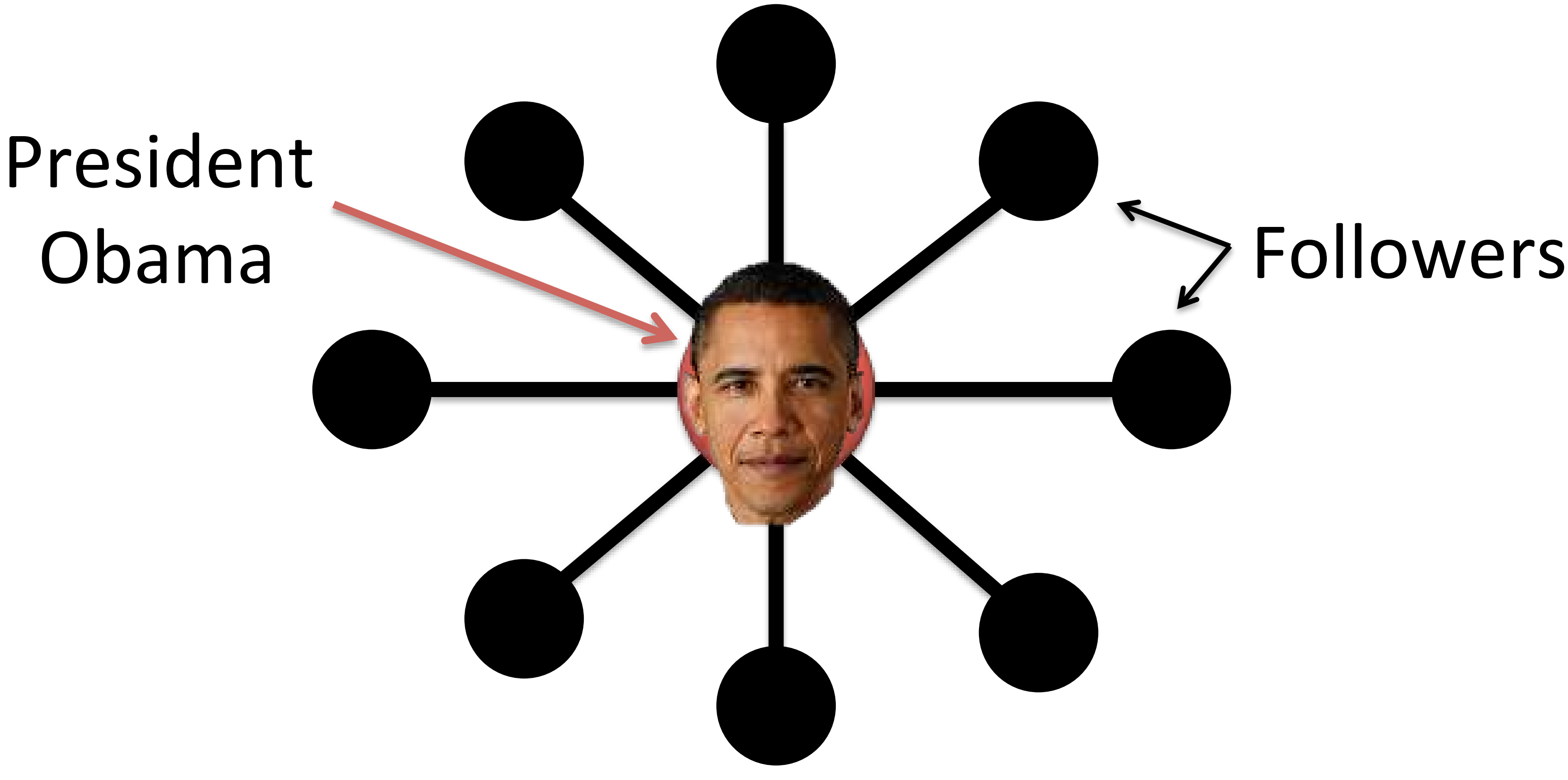
Hyper Parameters



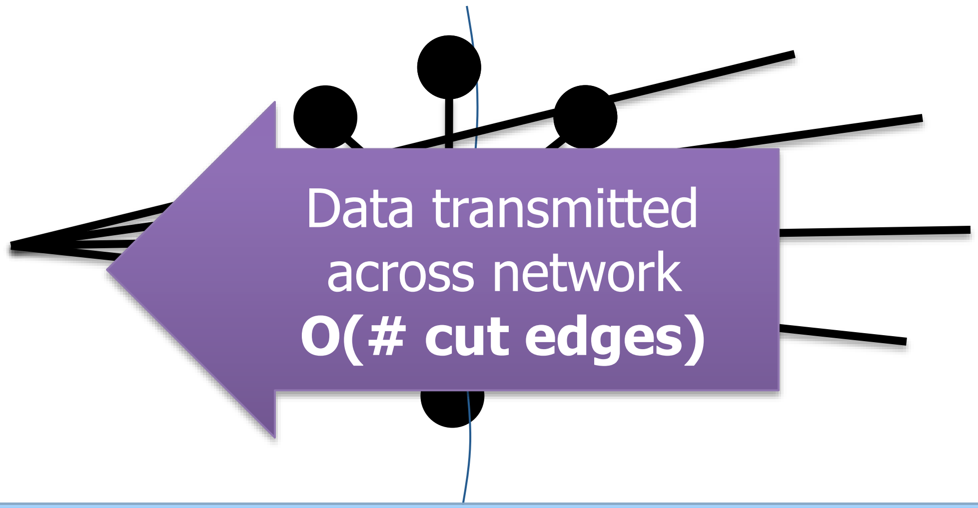
Common Words



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]

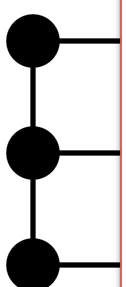
Popular partitioning tools (Metis, Chaco,...) perform poorly

[Abou-Rjeili et al. 06]

Extremely slow and require substantial memory

Random Partitioning

- Both GraphLab 1, Pregel, Twitter, Facebook,… rely on Random (hashed) partitioning for Natural Graphs



For *p* Machines:

$$\mathbb{E}\left[\frac{|Edges\ Cut|}{|E|}\right] = 1 - \frac{1}{p}$$

10 Machines → 90% of edges cut

100 Machines → 99% of edges cut!

All data is communicated… Little advantage over MapReduce

In Summary

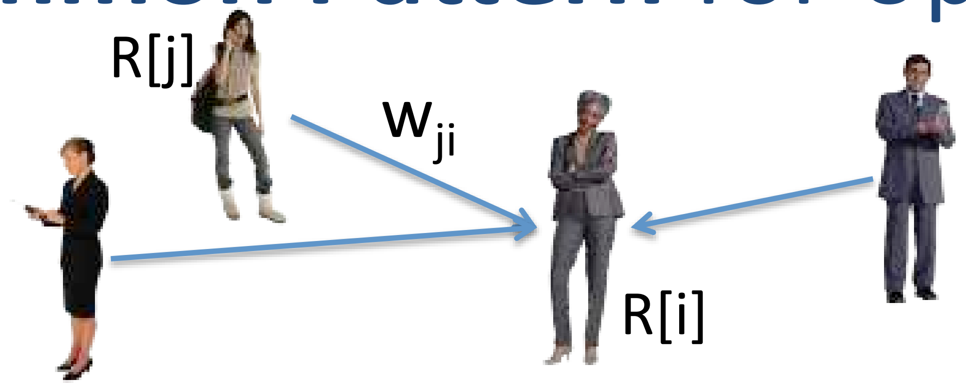
GraphLab 1 and Pregel are not well suited for natural graphs

- Poor performance on high-degree vertices
- Low Quality Partitioning

GraphLab2 PowerGraph

SCALABILITY

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**

GAS Decomposition

Gather (Reduce)

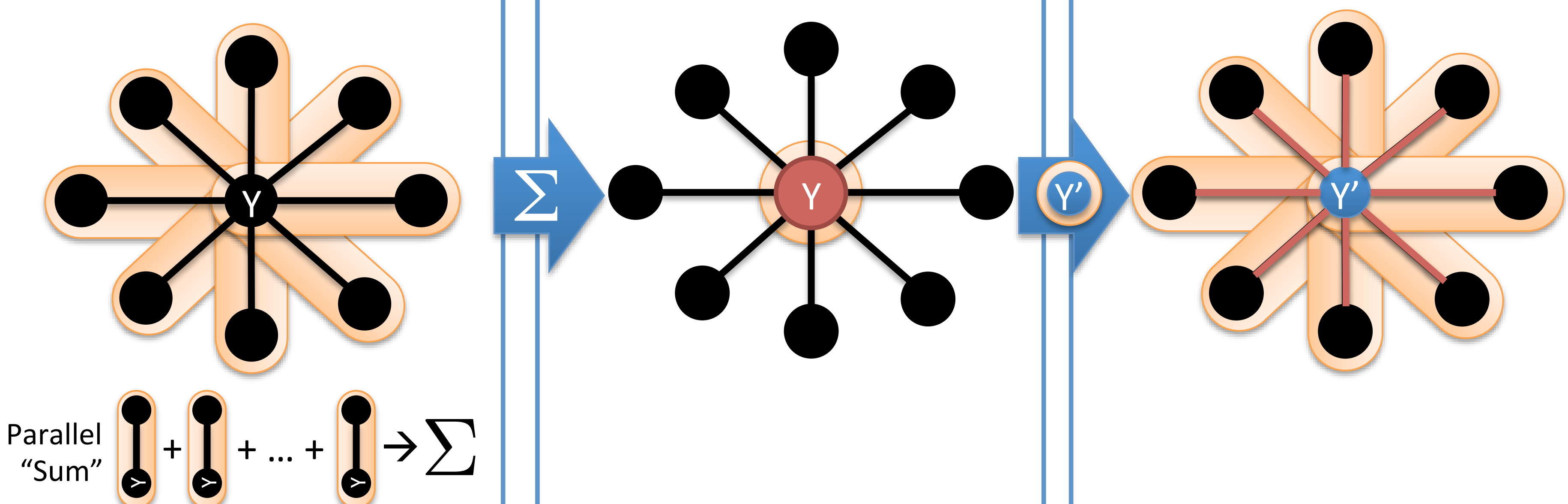
Accumulate information about neighborhood

Apply

Apply the accumulated value to center vertex

Scatter

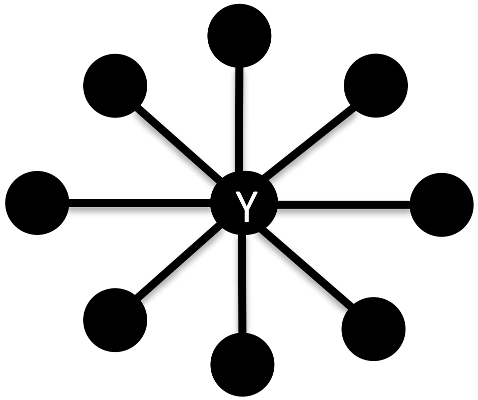
Update adjacent edges and vertices.



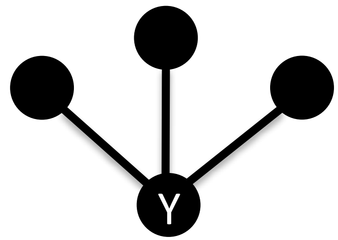
Many ML Algorithms fit into GAS Model

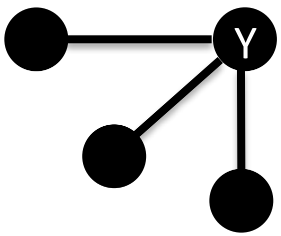
graph analytics, inference in graphical models, matrix factorization, collaborative filtering, clustering, LDA, …

Minimizing Communication in GL2 PowerGraph: **Vertex Cuts**

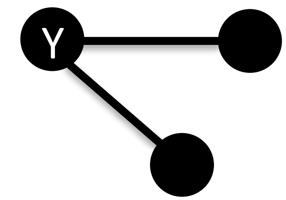


Minimizing Communication in GL2 PowerGraph: **Vertex Cuts**





Communication linear in # spanned machines



A **vertex-cut** minimizes # 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!

Minimizing Communication in GL2 PowerGraph: **Vertex Cuts**

Communication linear

GL2 PowerGraph 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

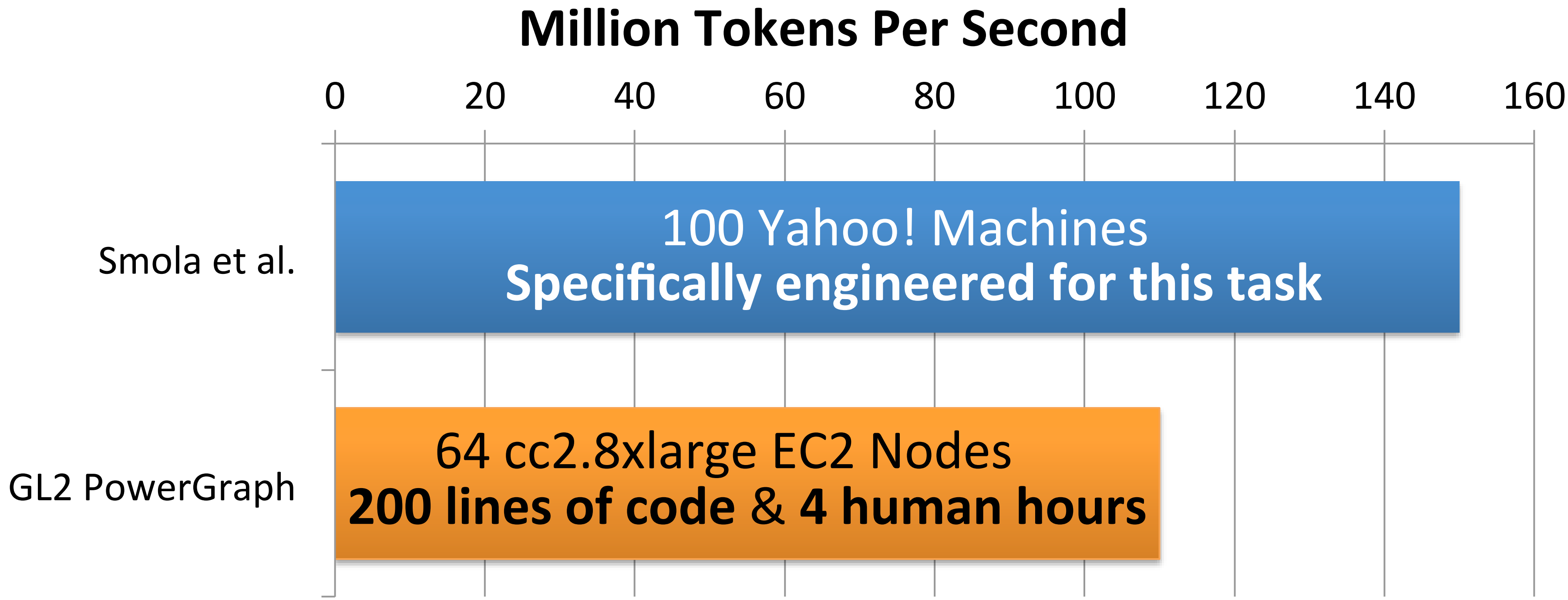
Triangle Counting on Twitter Graph

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

How well does GraphLab scale?

Yahoo Altavista Web Graph (2002):

One of the largest publicly available webgraphs

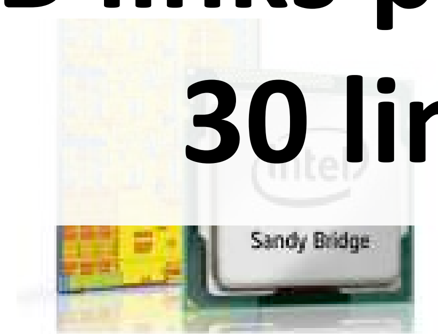
1.4B Webpages, 6.7 Billion Links

7 seconds per iter.

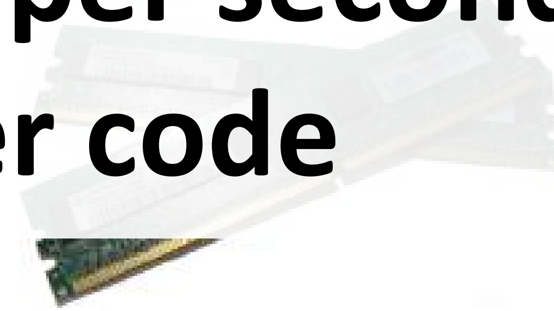
64 HPC Nodes

1B links processed per second

30 lines of user code



1024 Cores (2048 HT)

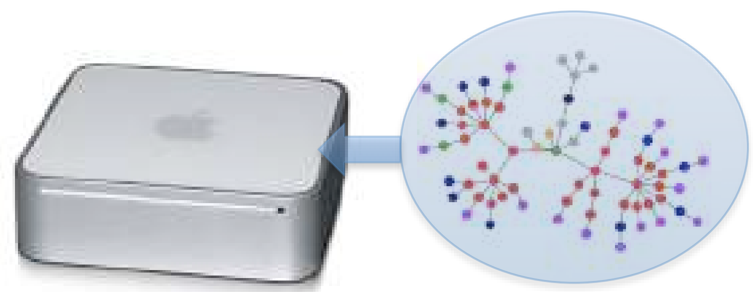


4.4 TB RAM

GraphChi: Going small with GraphLab

GraphLab

Solve huge problems on small or embedded devices?

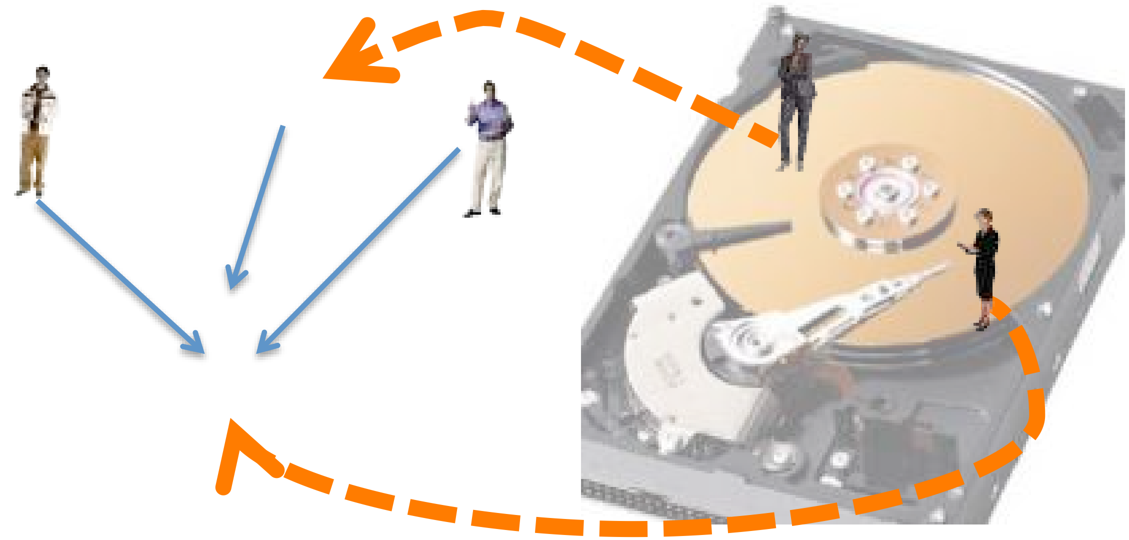


Key: Exploit non-volatile memory (starting with SSDs and HDs)

GraphChi – disk-based GraphLab

Challenge:

Random Accesses



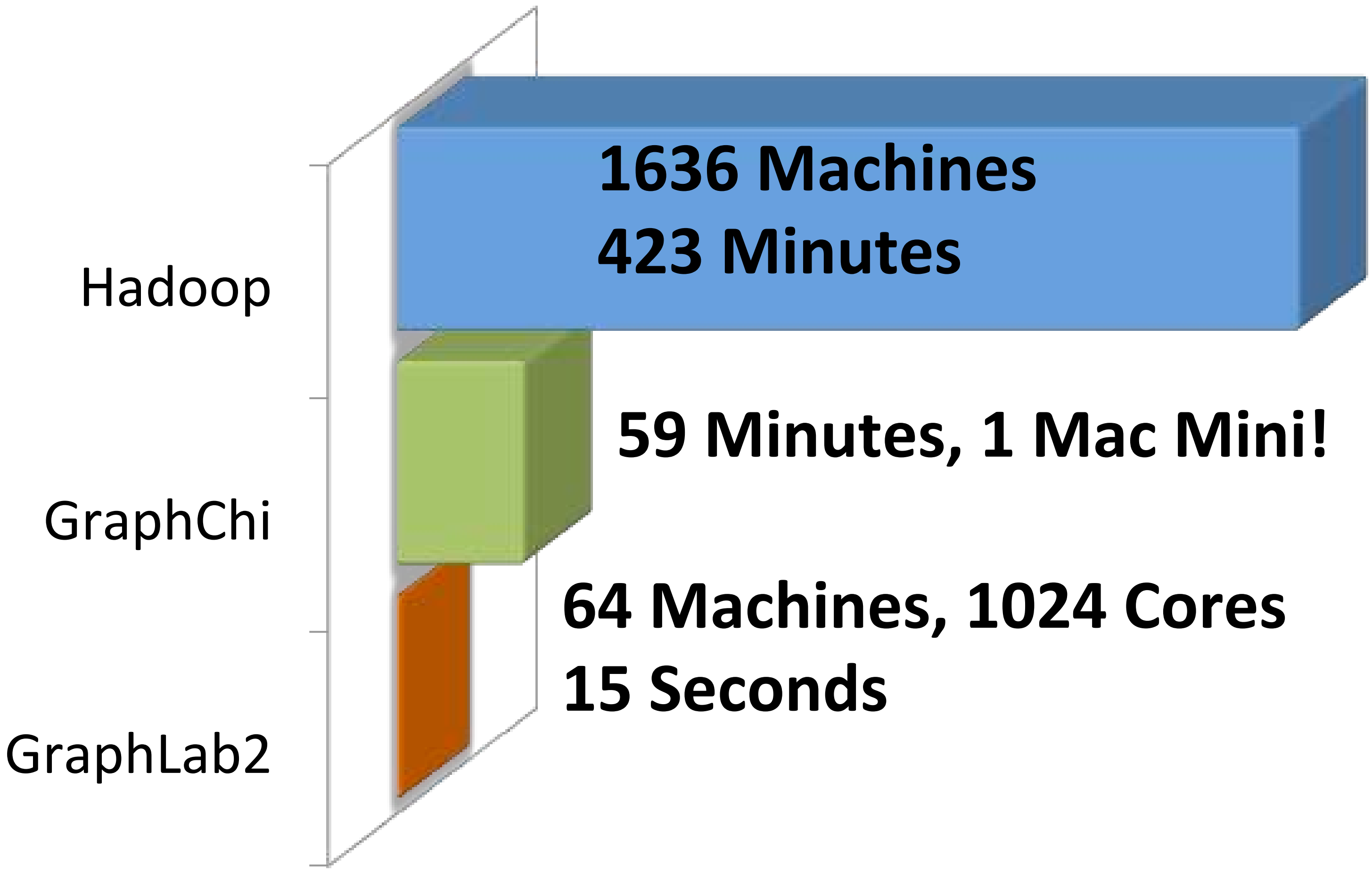
Novel GraphChi solution:

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]

GraphLab 1
Carnegie Mellon

- ML algorithms as vertex programs
- Asynchronous execution and consistency models

GraphLab 2
PowerGraph

- Natural graphs change the nature of computation
- Vertex cuts and gather/apply/scatter model

GL2 PowerGraph
focused on
Scalability

at the loss of
Usability

GraphLab 1

```
PageRank(i, scope){
  acc = 0
  for (j in InNeighbors) {
    acc += pr[j] * edge[j].weight
  }
  pr[i] = 0.15 + 0.85 * acc
}
```

Explicitly described operations

Code is intuitive

GraphLab 1

```
PageRank(i, scope){
  acc = 0
  for (j in InNeighbors) {
    acc += pr[j] * edge[j].weight
  }
  pr[i] = 0.15 + 0.85 * acc
}
```

Explicitly described operations

Code is intuitive

GL2 PowerGraph

Implicit operation

```
gather(edge) {
  return edge.source.value *
         edge.weight
}

merge(acc1, acc2) {
        return accum1 + accum2
}

apply(v, accum) {
  v.pr = 0.15 + 0.85 * acc
}
```

Implicit aggregation

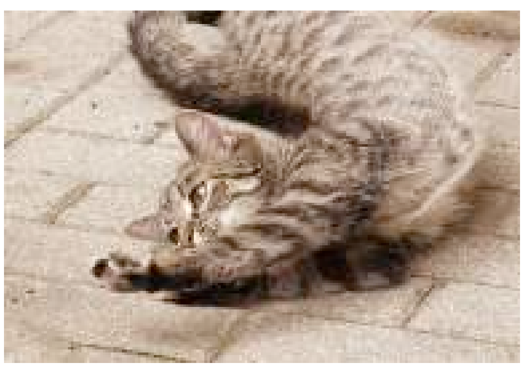
Need to understand engine to understand code

GraphLab 1
Carnegie Mellon

Great flexibility,
but hit scalability wall

GraphLab 2
PowerGraph

Scalability,
but very rigid abstraction
(many contortions needed to implement SVD++, Restricted Boltzmann Machines)



What now?

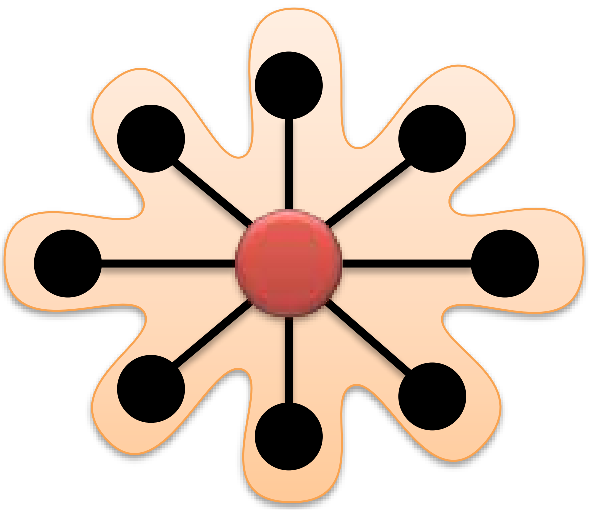
GraphLab 3

WarpGraph

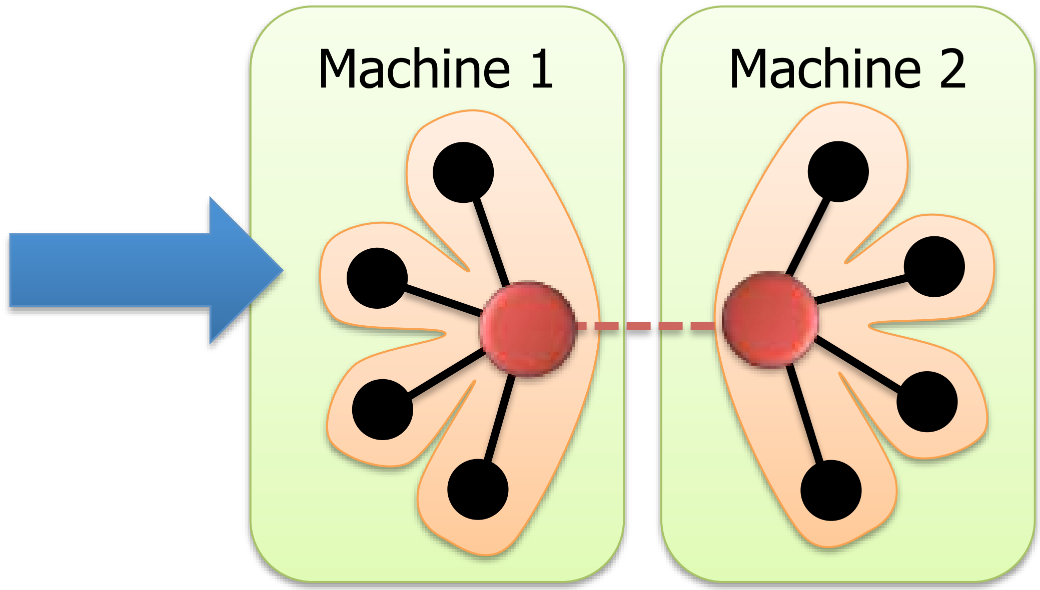
USABILITY

GL3 WarpGraph Goals

Program Like GraphLab 1



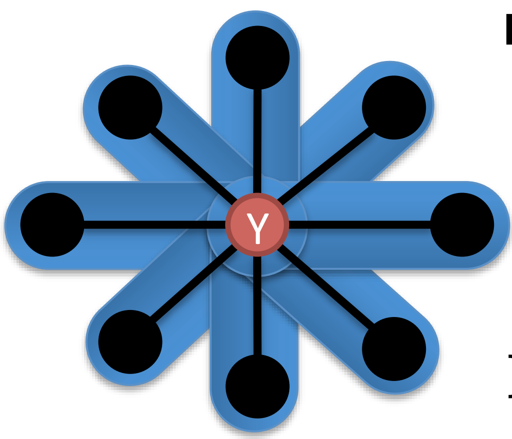
Run Like GraphLab 2



Fine-Grained Primitives

Expose Neighborhood Operations through Parallelizable Iterators

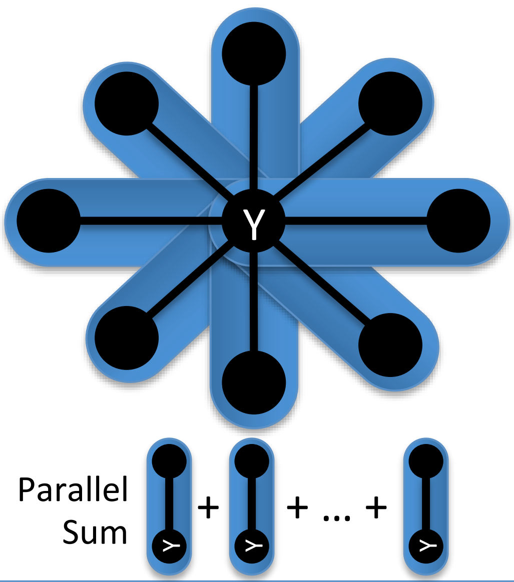
$$R[i] = 0.15 + 0.85 \boxed{\sum_{(j,i) \in E} \mathrm{w}[j,i] * R[j]}$$



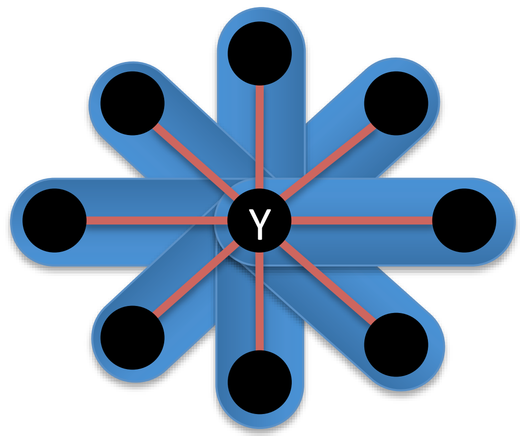

```
PageRankUpdateFunction(Y) {
    Y.pagerank = 0.15 + 0.85 *




}
```

Expressive, Extensible Neighborhood API

MapReduce over Neighbors

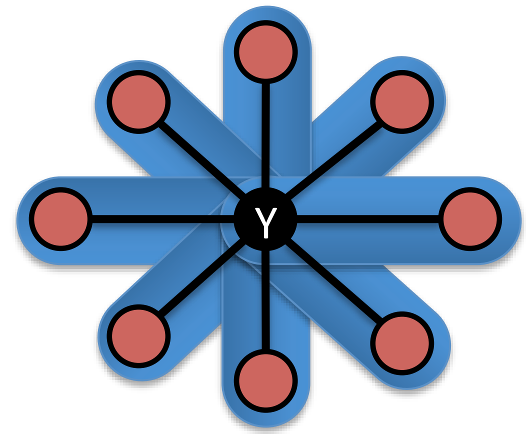


Parallel Transform Adjacent Edges



Modify adjacent edges

Broadcast



Schedule a selected subset of adjacent vertices

Can express every GL2 PowerGraph program (more easily) in GL3 WarpGraph

But GL3 is more expressive

```
UpdateFunction(v) {
  if (v.data == 1)
    accum = MapReduceNeighs(g,m)
  else ...
}
```

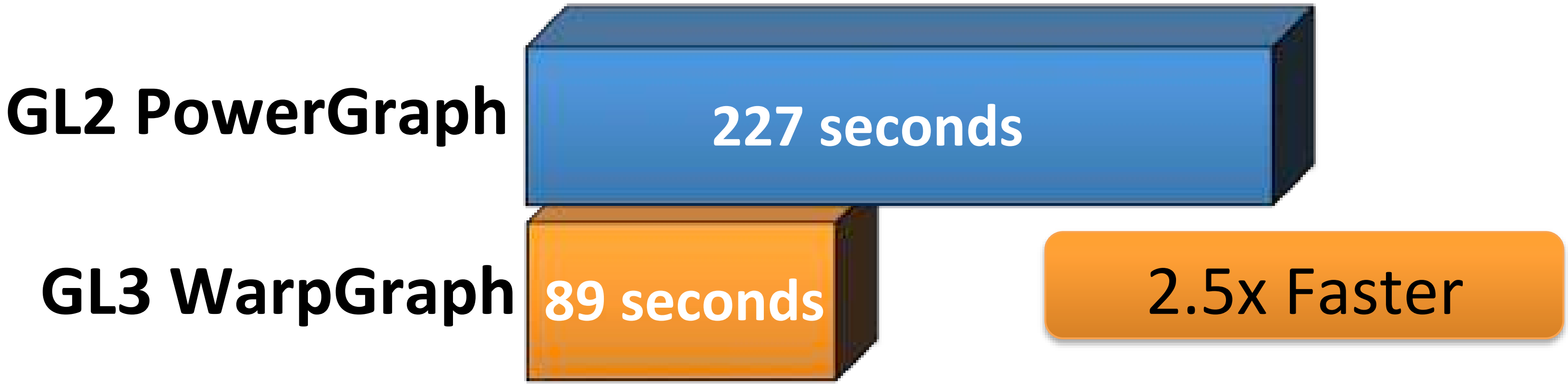
Multiple gathers

Scatter before gather

Conditional execution

Graph Coloring

Twitter Graph: 41M Vertices 1.4B Edges



WarpGraph outperforms PowerGraph with simpler code

32 Nodes x 16 Cores (EC2 HPC cc2.8x)

GraphLab 1

- ML algorithms as vertex programs
- Asynchronous execution and consistency models

GraphLab 2 PowerGraph

- Natural graphs change the nature of computation
- Vertex cuts and gather/apply/scatter model

GraphLab 3 WarpGraph

- Usability is key
- Access neighborhood through parallelizable iterators and latency hiding

Usability

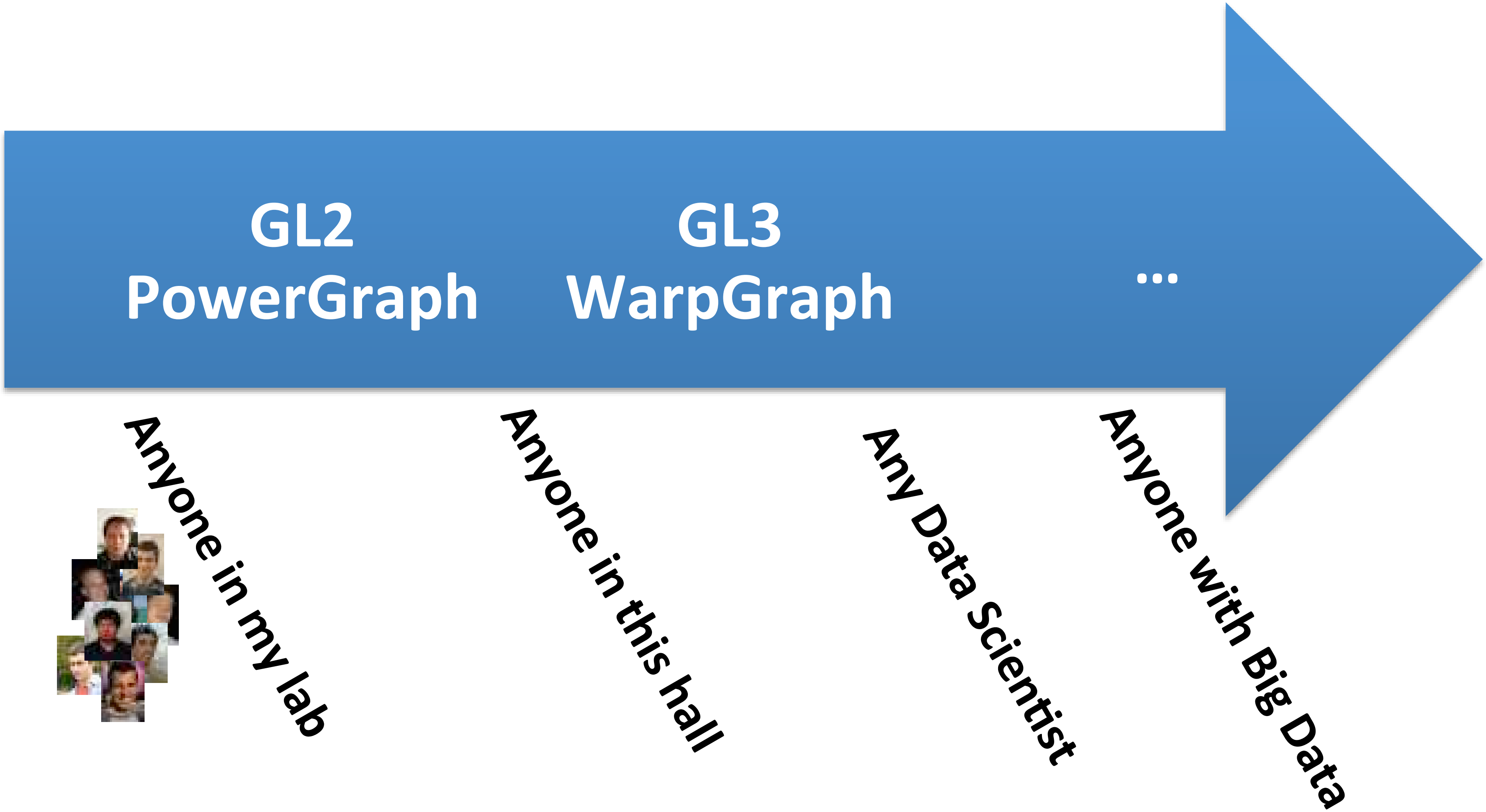
RECENT RELEASE: GRAPHLAB 2.2, INCLUDING WARPGRAPH ENGINE

And support for
streaming/dynamic graphs!

Consensus that WarpGraph is much easier to use than PowerGraph

“User study” group biased… :-)

Usability for Whom???



Machine Learning
PHASE 3

USABILITY

Exciting Time to Work in ML

Unique opportunities to change the world!! ☺
But, **every deployed system** is an one-off solution,
and **requires PhDs** to make work… ☹

But…

ML key to any new service we want to build

Even basics of scalable ML can be challenging

6 months from R/Matlab to production, at best

State-of-art ML algorithms trapped in research papers

Goal of GraphLab 3:
Make huge-scale ***machine learning*** accessible to all! ☺

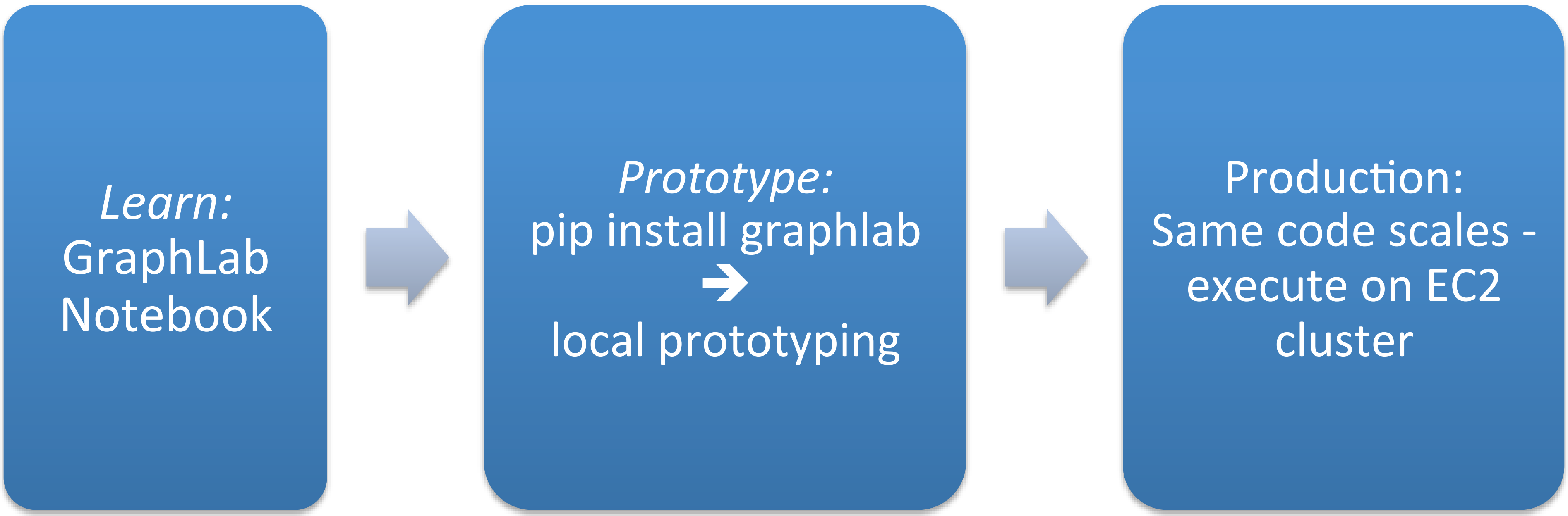
Step 1

Learning ML in Practice with **GraphLab Notebook**

Step 2

GraphLab+Python:

ML Prototype to Production


Learn:
GraphLab Notebook
Prototype:
pip install graphlab
➔
local prototyping
Production:
Same code scales - execute on EC2 cluster

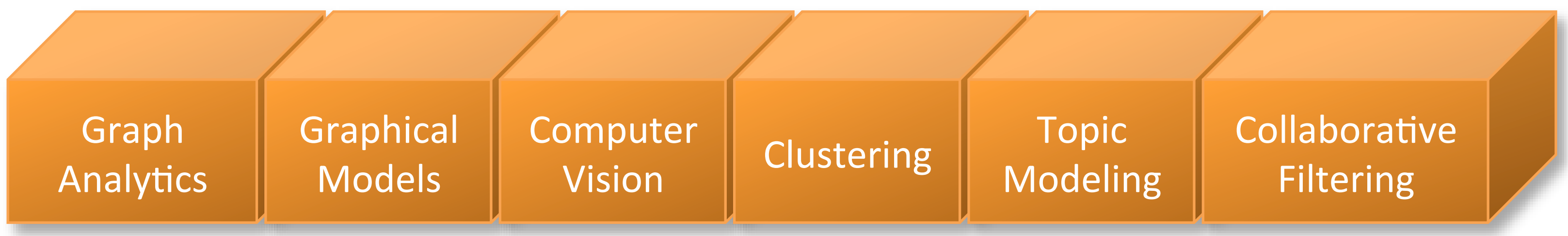
Step 3

GraphLab Toolkits:

Integrated State-of-the-Art ML in Production

GraphLab Toolkits

Highly scalable, state-of-the-art machine learning straight from python



Now with GraphLab: Learn/Prototype/Deploy

Even basics of scalable ML can be challenging

Learn ML with GraphLab Notebook

6 months from R/Matlab to production, at best

pip install graphlab then deploy on EC2

State-of-art ML algorithms trapped in research papers

Fully integrated via GraphLab Toolkits

We're selecting strategic partners

Help define our strategy & priorities
And, get the value of GraphLab in your company

partners@graphlab.com

v1 Possibility

v2 Scalability

v3 Usability

GraphLab 2.2 available now: **graphlab.com**

Define our future: **partners@graphlab.com**

Needless to say: **jobs@graphlab.com**