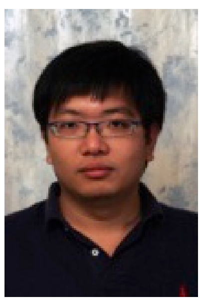
GraphLab

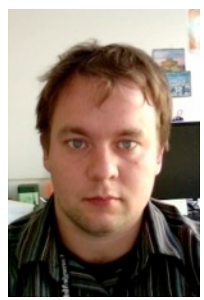
Large-Scale Machine Learning and Graphs

Carlos Guestrin

Joseph Gonzalez



Yucheng Low



Aapo Kyrola

Haijie Gu

Joseph Bradley

Danny Bickson

PHASE 1

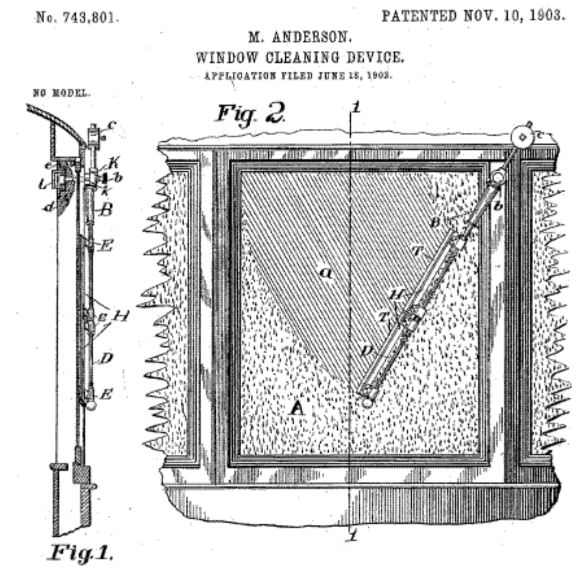
POSSIBILITY

PHASE 2

SCALABILITY

PHASE 3

USABILITY


No. 743,801.
PATENTED NOV. 10, 1903.
M. ANDERSON.
WINDOW CLEANING DEVICE.
APPLICATION FILED JUNE 18, 1903.
NO MODEL.
Fig. 2.
Fig.1.


a realm all its own... Cadillac


ABS


TEXACO
TWO GUNS
STORE ENTRANCE

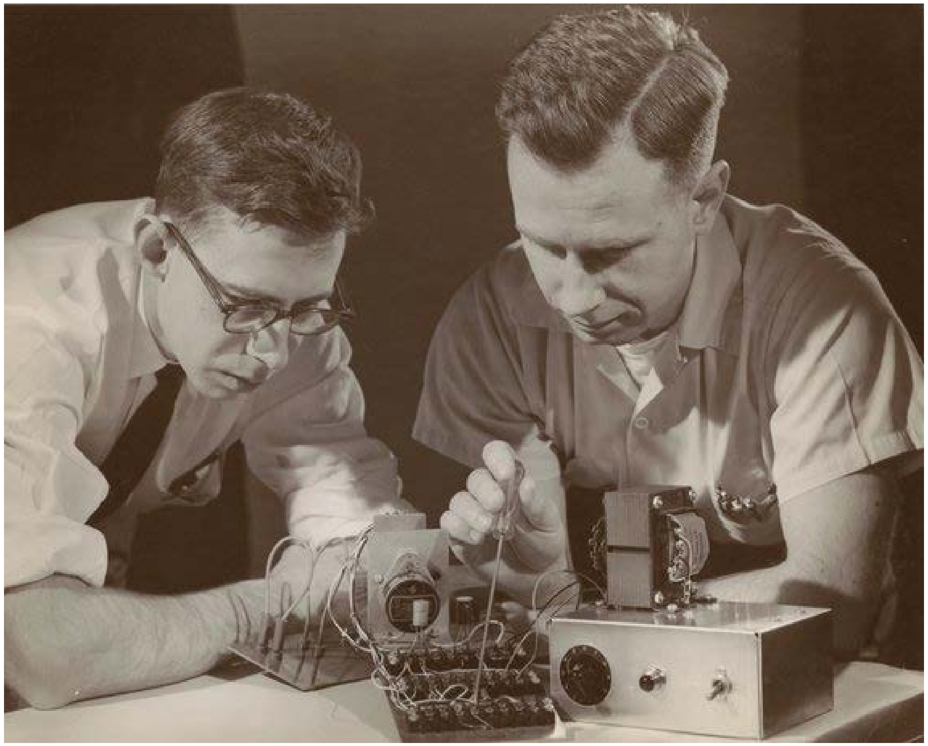
Three Phases in Technological Development

1. Possibility
2. Scalability
3. Usability

Wide Adoption Beyond Experts & Enthusiast

Machine Learning
PHASE 1

POSSIBILITY



Rosenblatt 1957

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."

The Power of Dependencies

where the value is!

Flashback to 1998

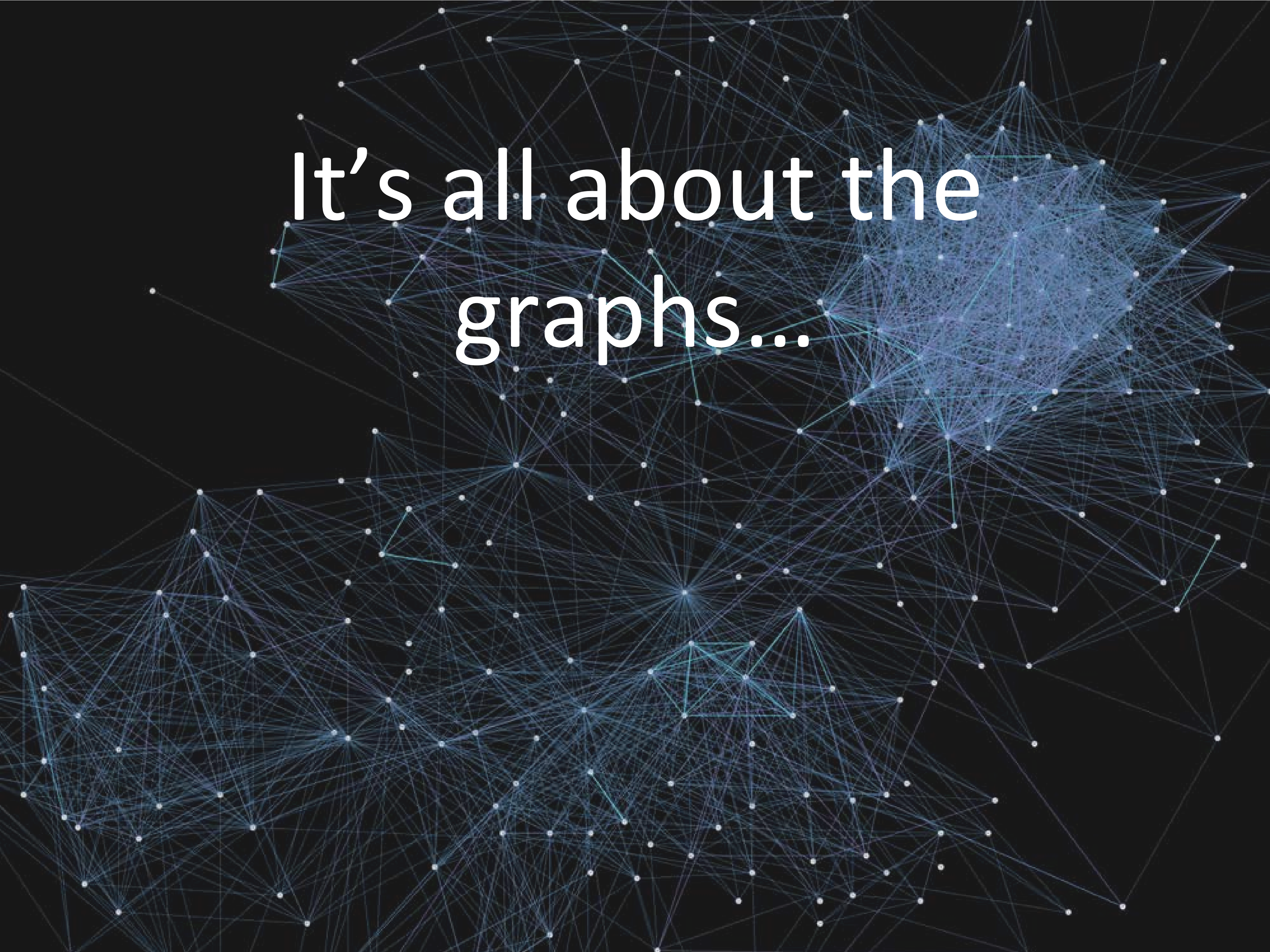
altavista™
SEARCH SOFTWARE

Why?

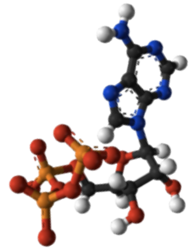
Google!
BETA

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

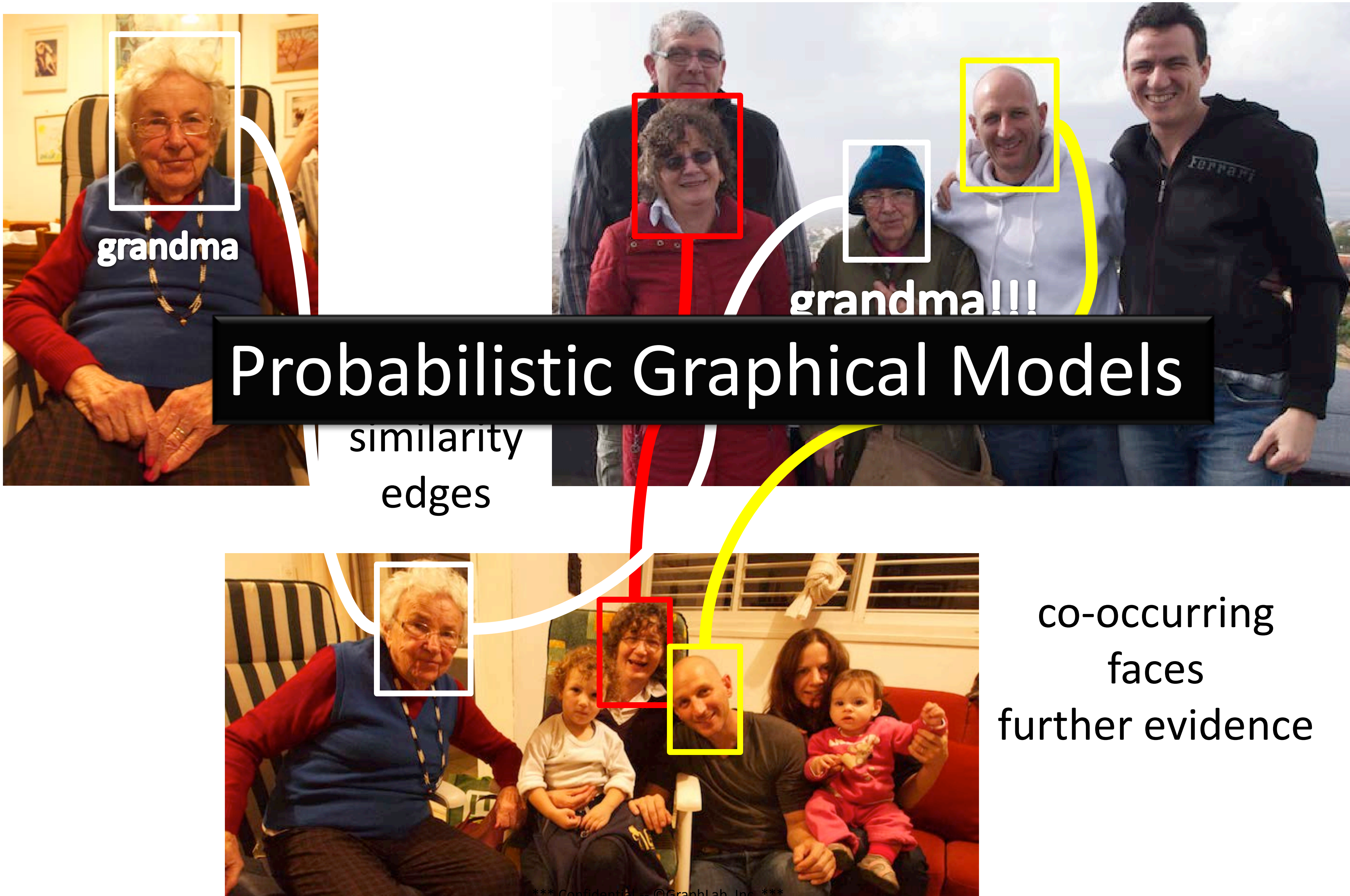
Examples of Graphs in Machine Learning

Label a Face and Propagate

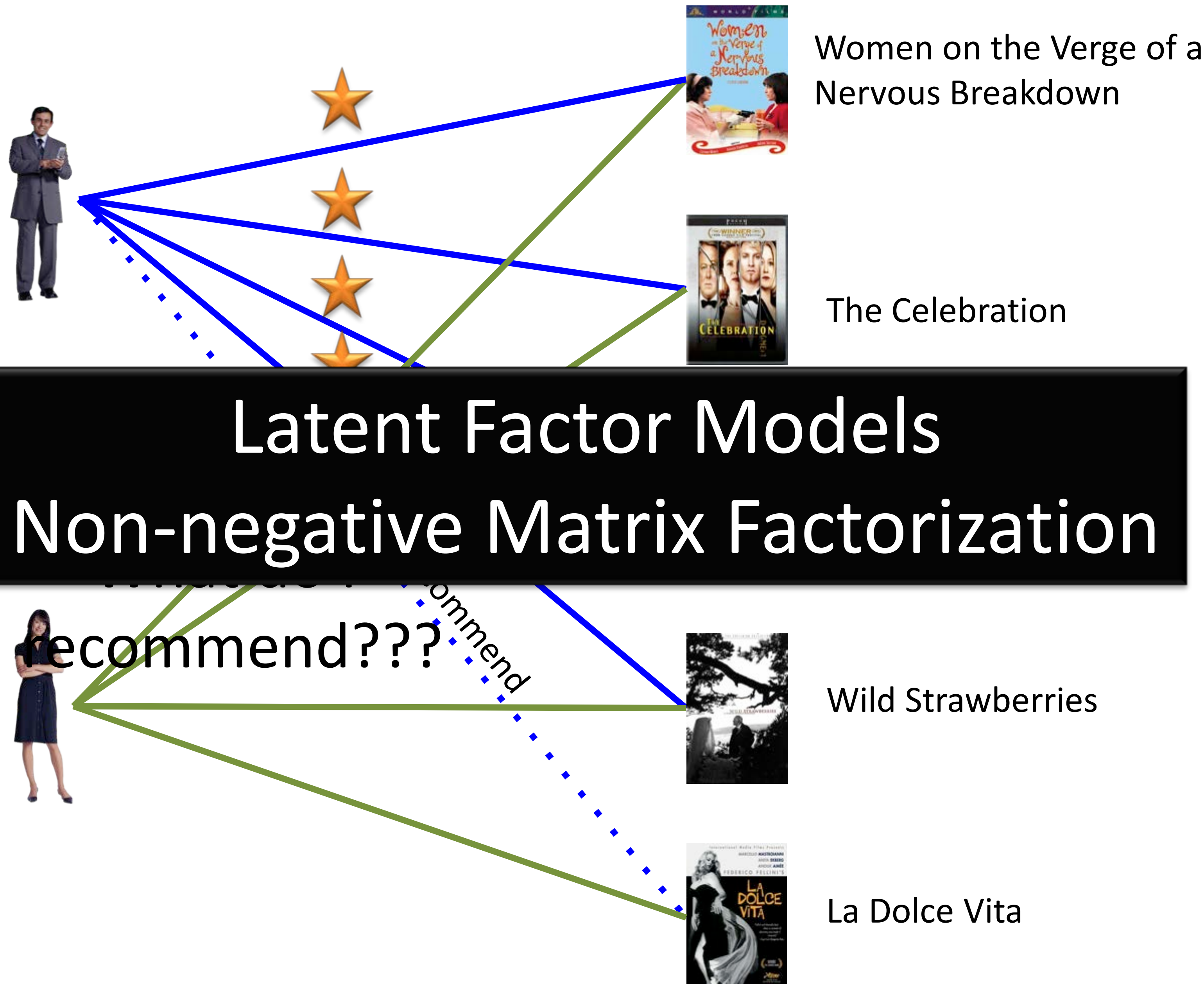
Pairwise similarity not enough…

Not similar enough
to be sure

Propagate Similarities & Co-occurrences for Accurate Predictions

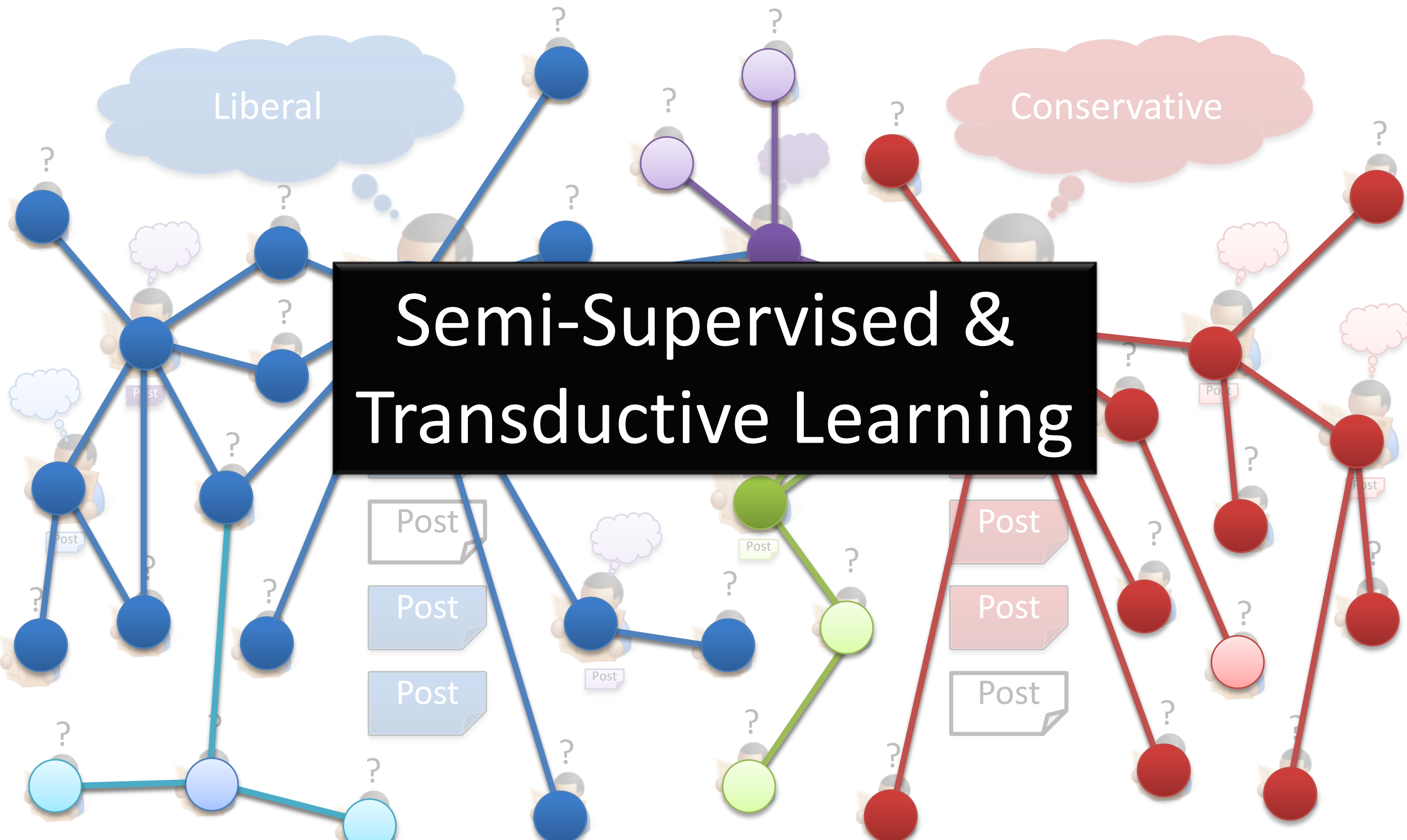


Collaborative Filtering: Exploiting Dependencies

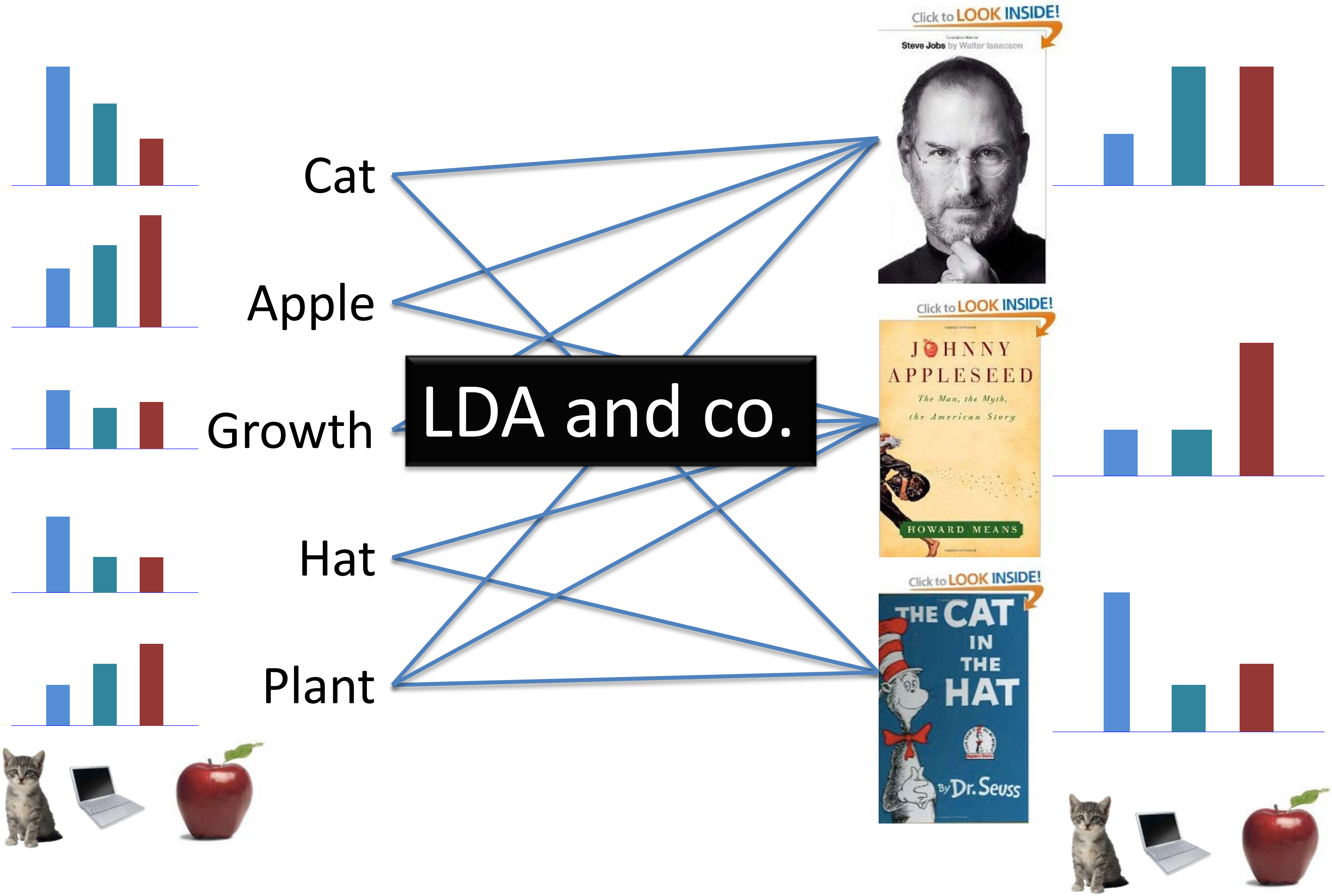


Latent Factor Models
Non-negative Matrix Factorization

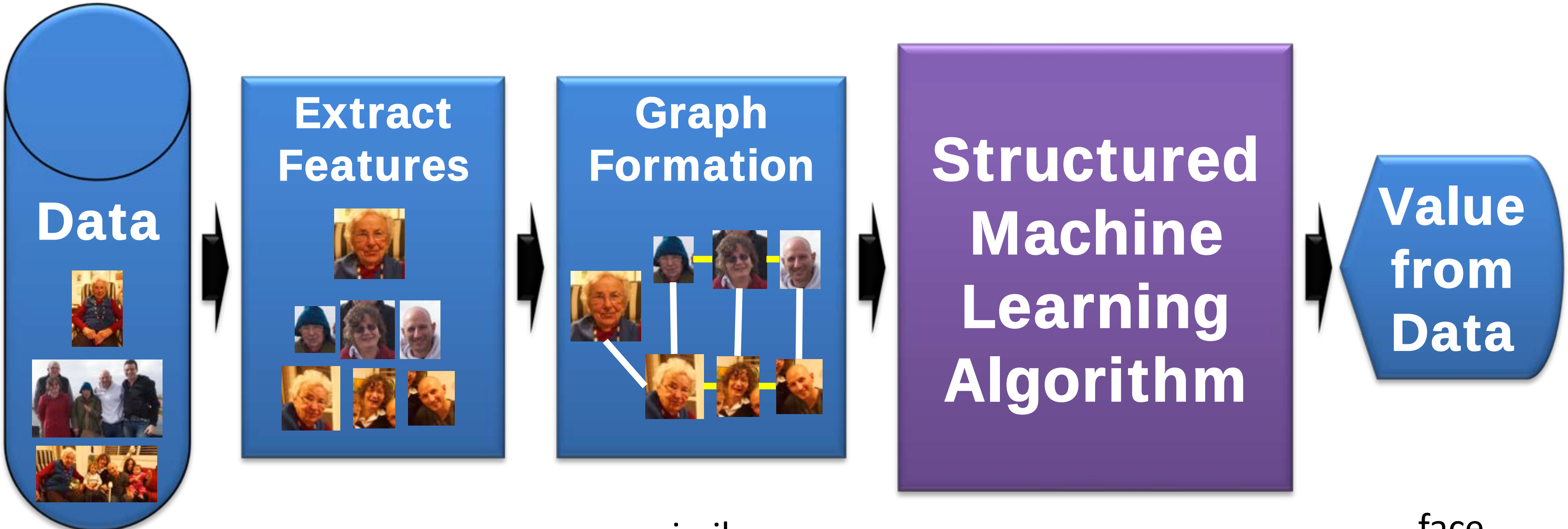
Estimate Political Bias



Topic Modeling

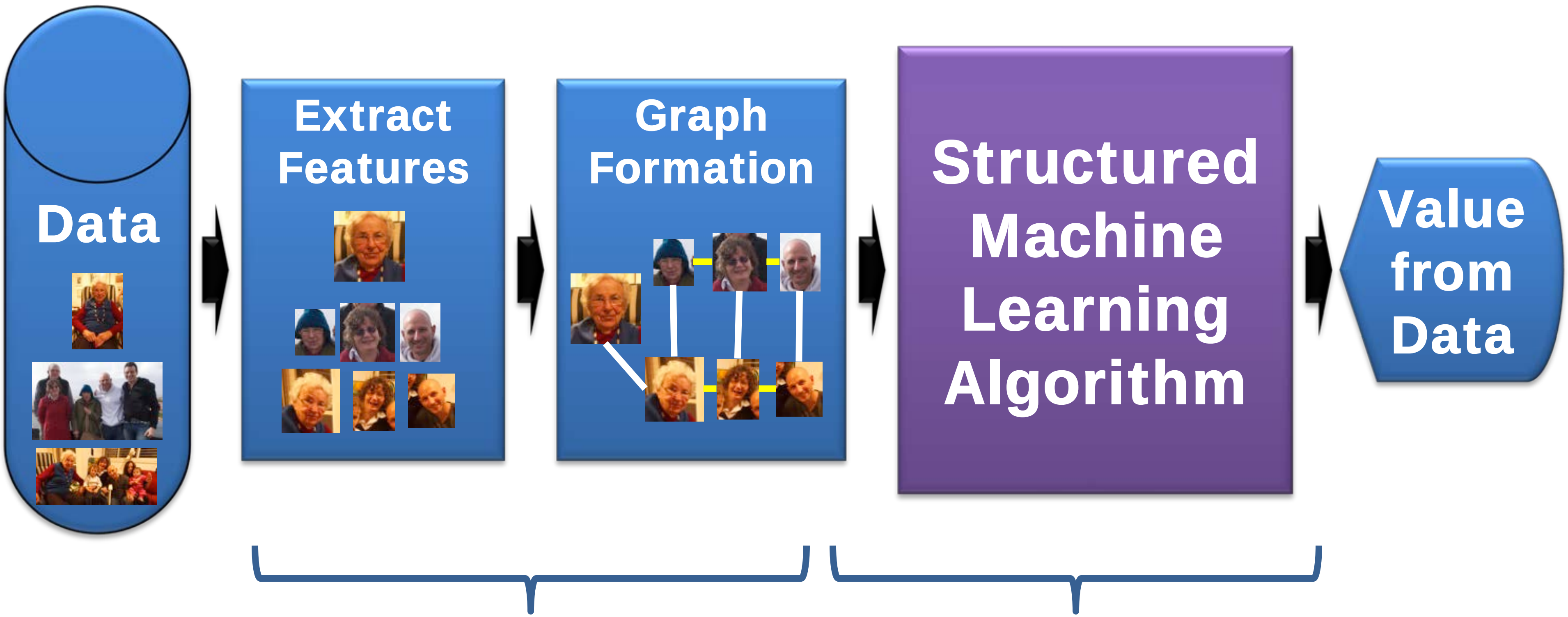


Machine Learning Pipeline



Data	Extract Features	Graph Formation	Structured Machine Learning Algorithm	Value from Data
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

GraphLab

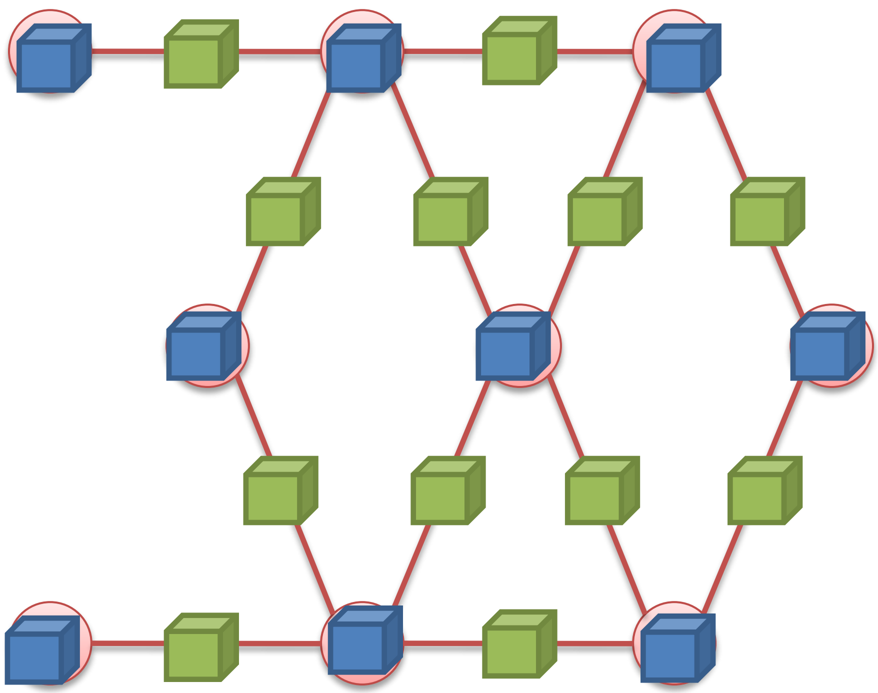
GraphLab 1

Carnegie Mellon

POSSIBILITY

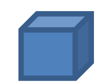
Data Graph

Data associated with vertices and edges

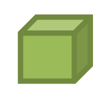


Graph: 

- Social Network

Vertex Data: 

- User profile text
- Current interests estimates

Edge Data: 

- Similarity weights

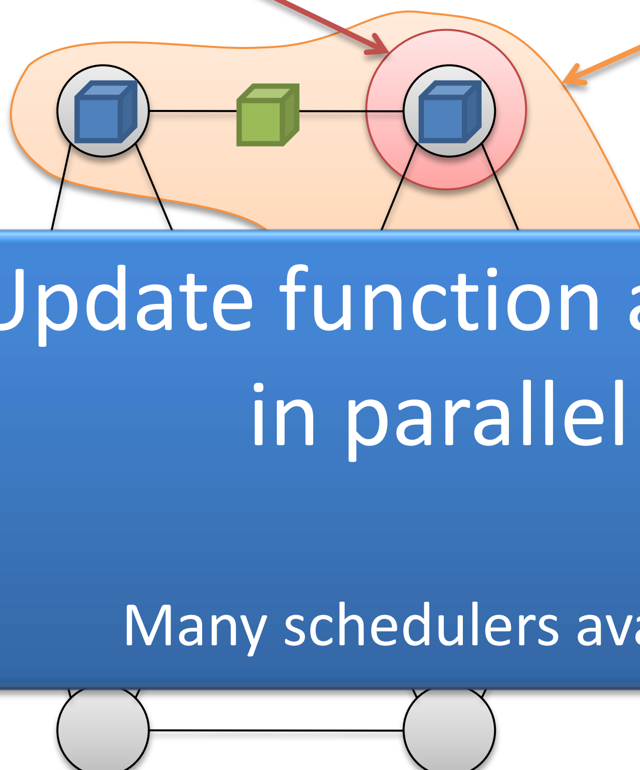
How do we *program* **graph** computation?

“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]) ←scope;
```

Update function applied (asynchronously) in parallel until convergence

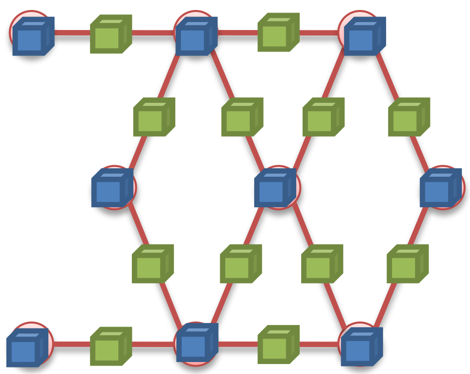
Many schedulers available to prioritize computation

```
}
```

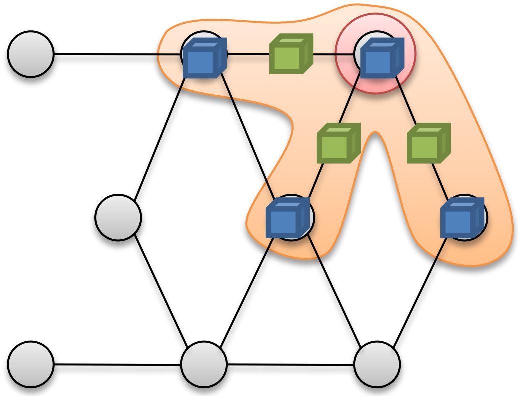
Dynamic computation

The GraphLab Framework

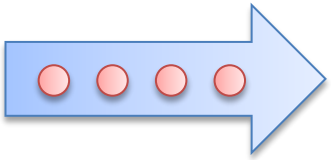
Graph Based
Data Representation



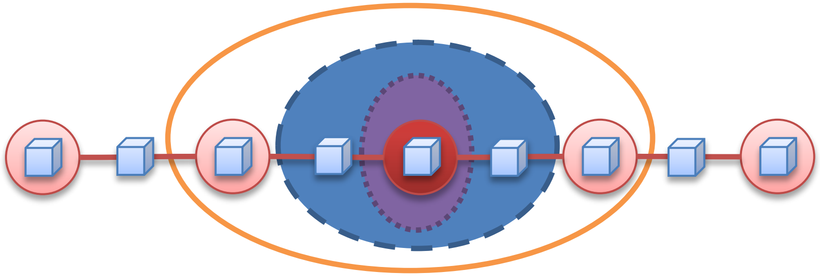
Update Functions
User Computation



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

...Many others...

Matrix Factorization

Linear Solvers

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

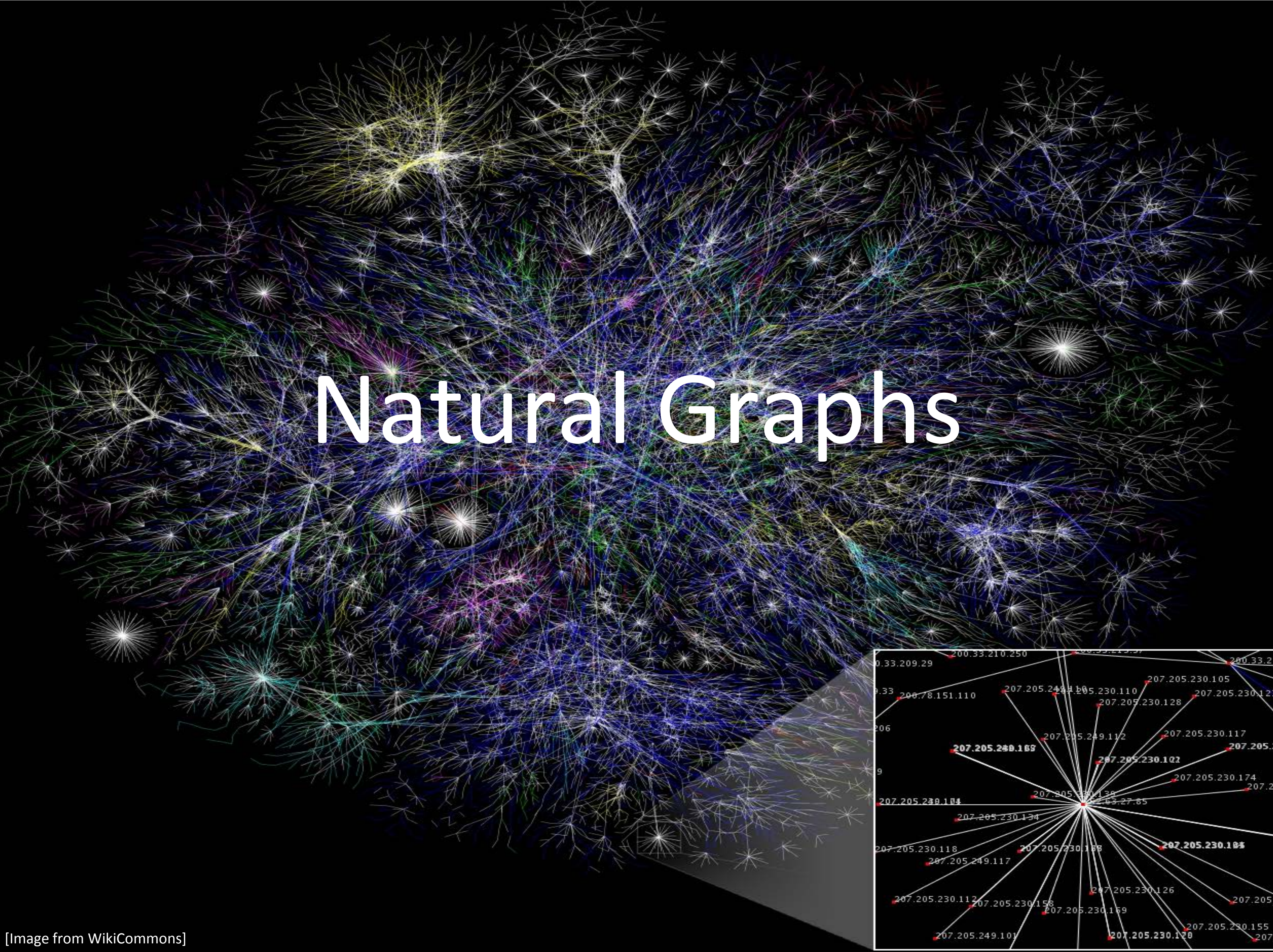
Thus far…

GraphLab 1 provided exciting scaling performance

But…

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

Natural Graphs



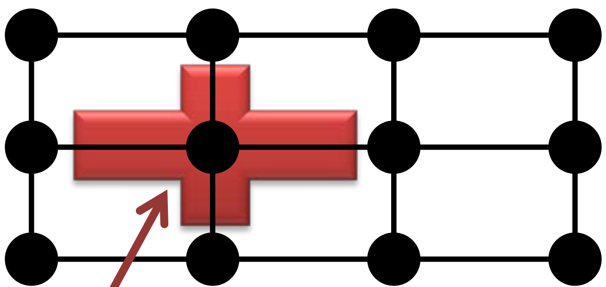
[Image from WikiCommons]

Problem:

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

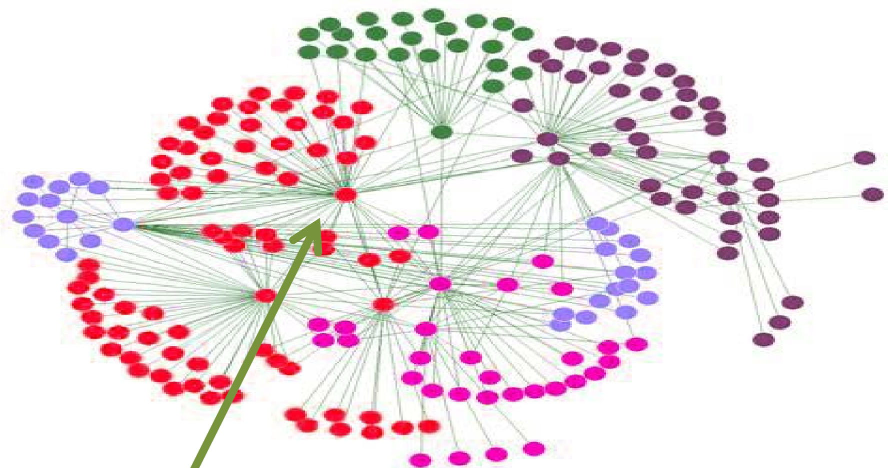
Achilles Heel: Idealized Graph Assumption

Assumed…



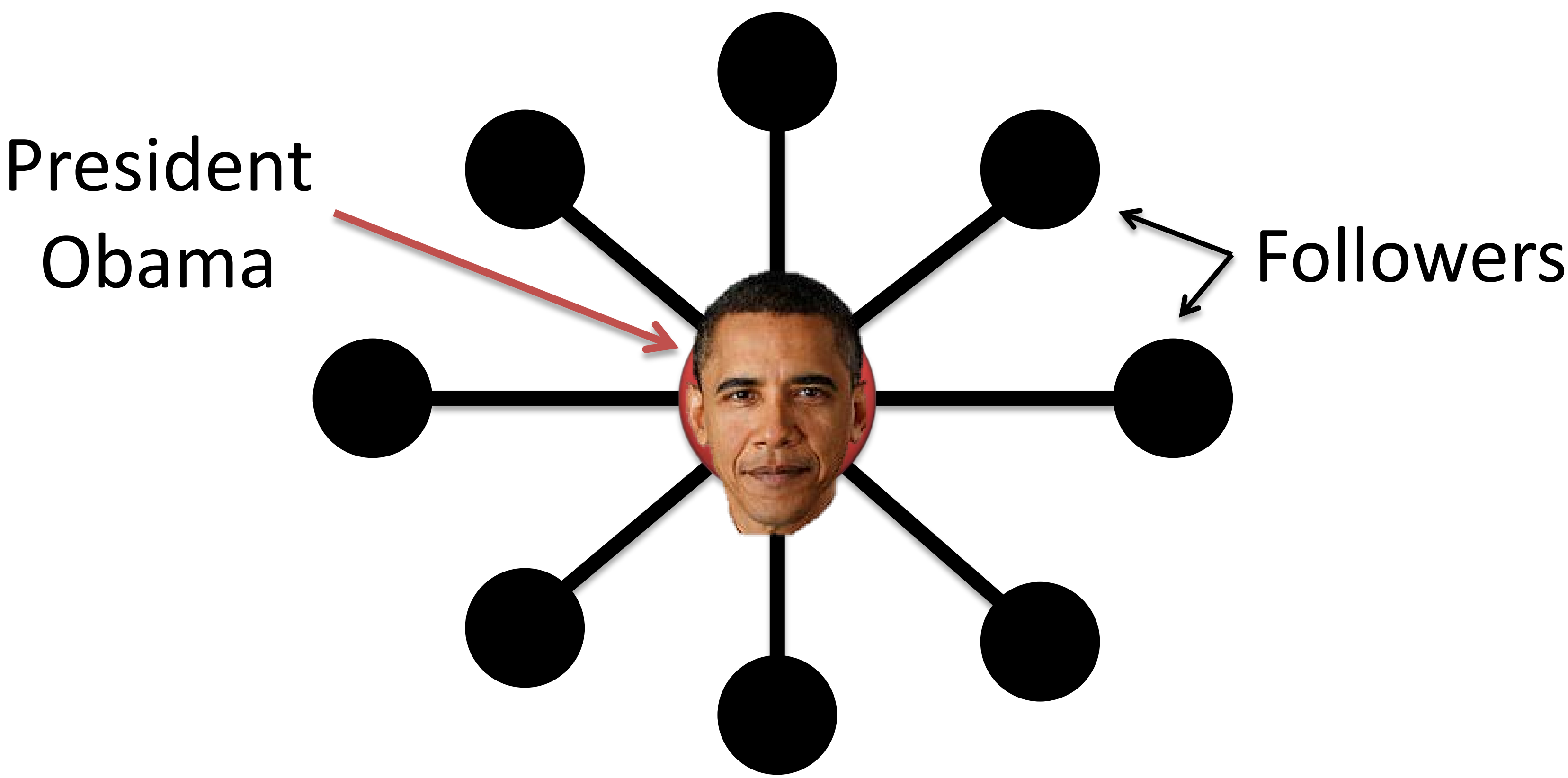
Small degree → Easy to partition

But, Natural Graphs…

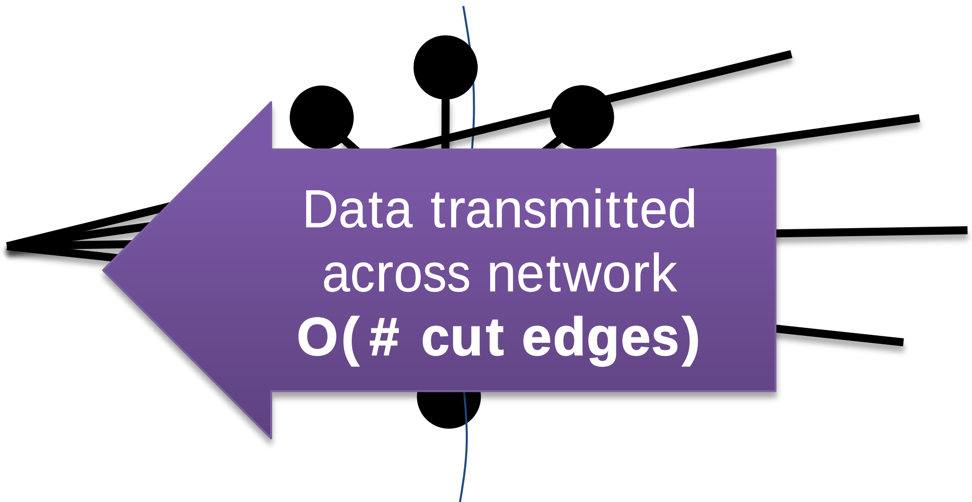


Many high degree vertices (power-law degree distribution) → Very hard to partition

Power-Law Degree Distribution
"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

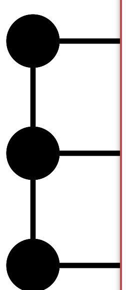
[Abou-Rjeili et al. 06]

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:



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

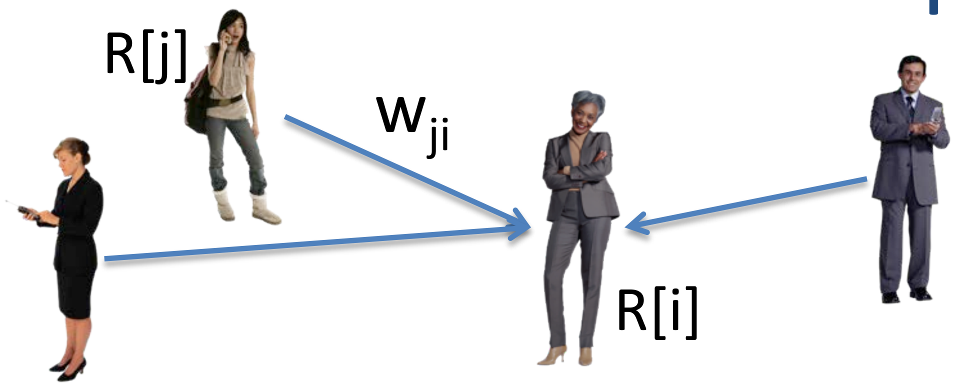
- Poor performance on high-degree vertices
- Low Quality Partitioning

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**

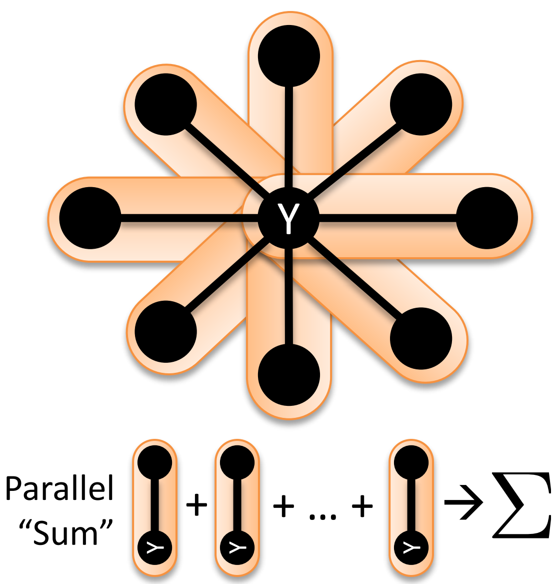
```
// Trigger neighbors to run again
if R[i] not converged then
  foreach( j in out_neighbors(i))
    signal vertex-program on j
```

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

GAS Decomposition

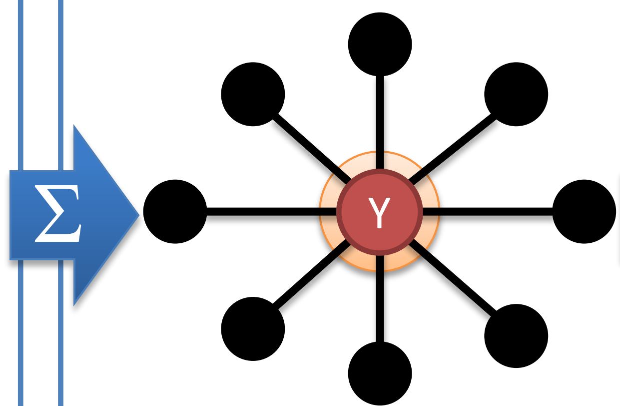
Gather (Reduce)

Accumulate information about neighborhood



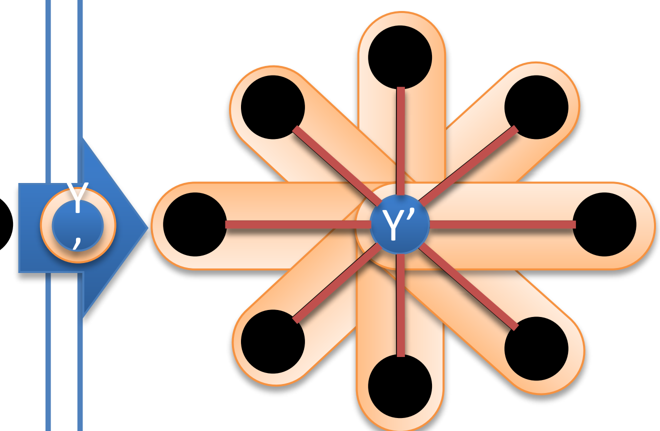
Apply

Apply the accumulated value to center vertex



Scatter

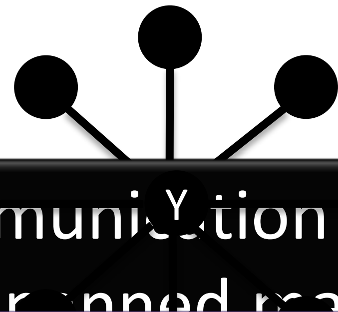
Update adjacent edges and vertices.



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**



Communication linear in # spanned machines

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!

Triangle Counting on Twitter Graph

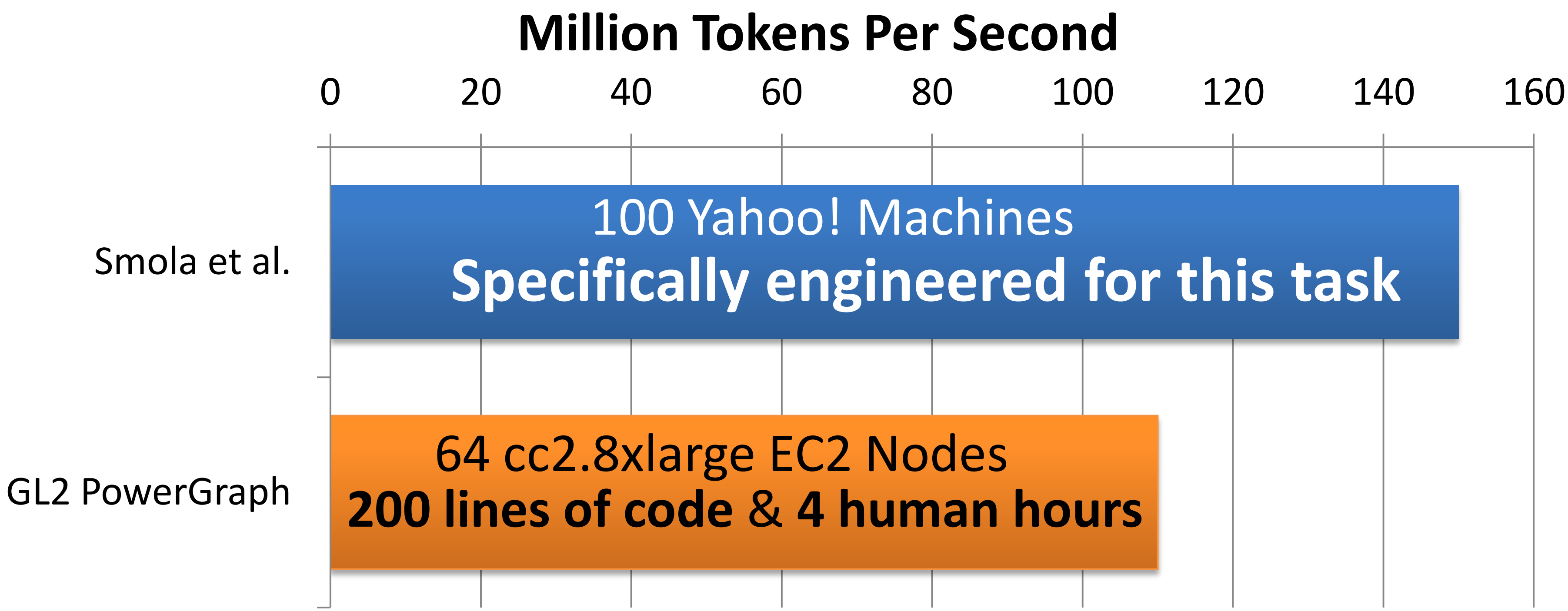
34.8 Billion Triangles

Why? **Hadoop Wrong Abstraction for Graphs** → 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

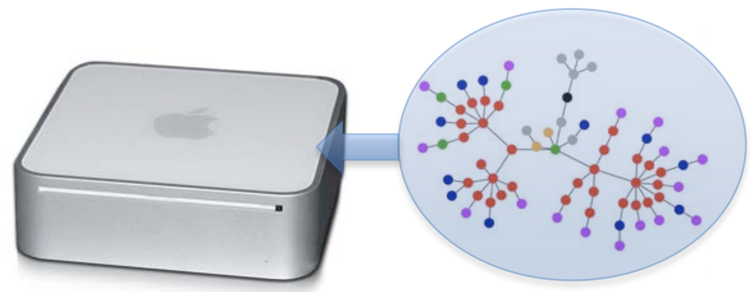
1024 Cores (2048 HT)

4.4 TB RAM

GraphChi: Going small with GraphLab

GraphLab

Solve huge problems on small or embedded devices?

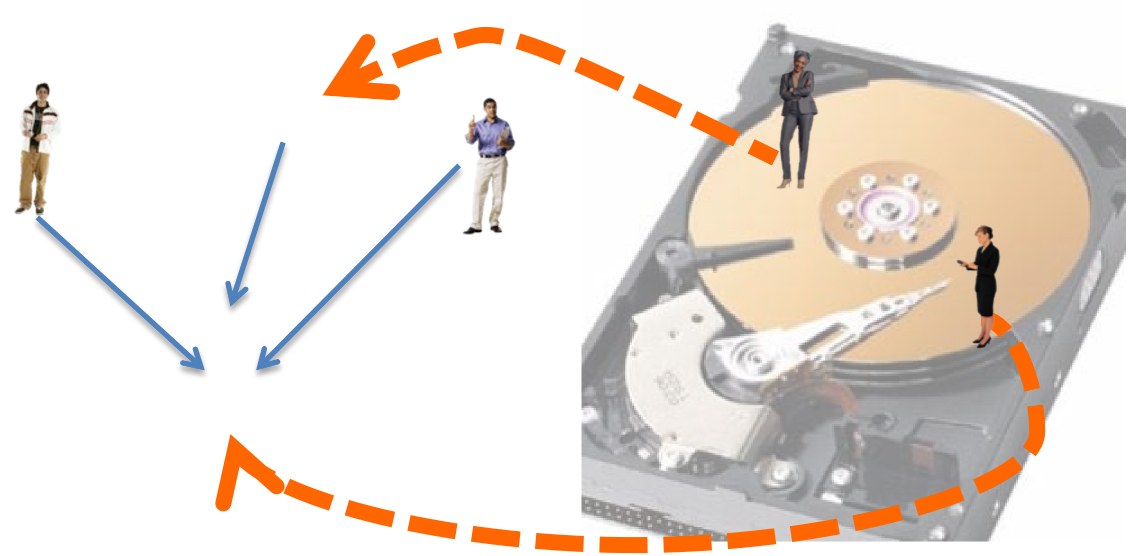


Key: Exploit non-volatile memory (starting with SSDs and HDs)

GraphChi – disk-based GraphLab

Challenge:

Random Accesses



Novel GraphChi solution:

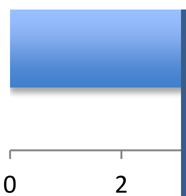
Parallel sliding windows method ➔ *minimizes number of random accesses*

See the paper for more comparisons.

Performance Comparison

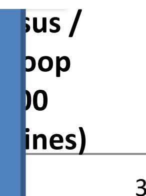
PageRank

Twitter-2010 (1.5B edges)



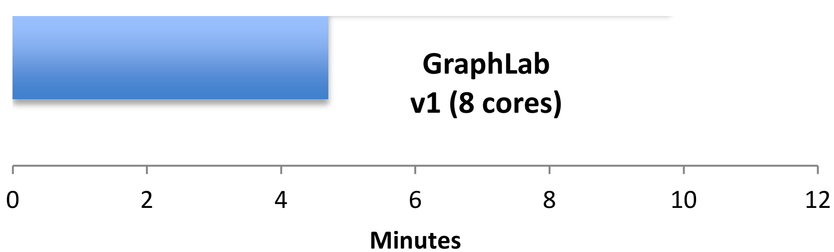
WebGraph Belief Propagation (U Kang et al.)

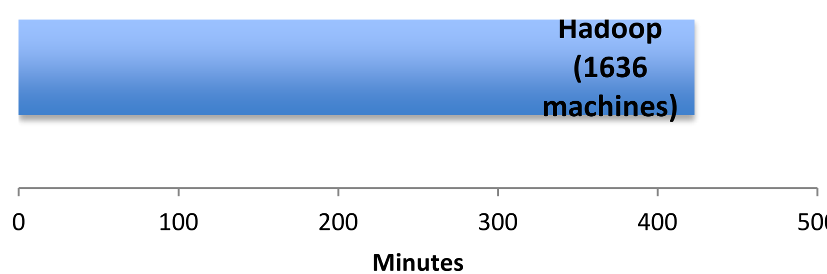
Yahoo-web (6.7B edges)



- ✓GraphChi can solve problems as big as existing large-scale systems
- ✓Comparable performance

Matrix





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

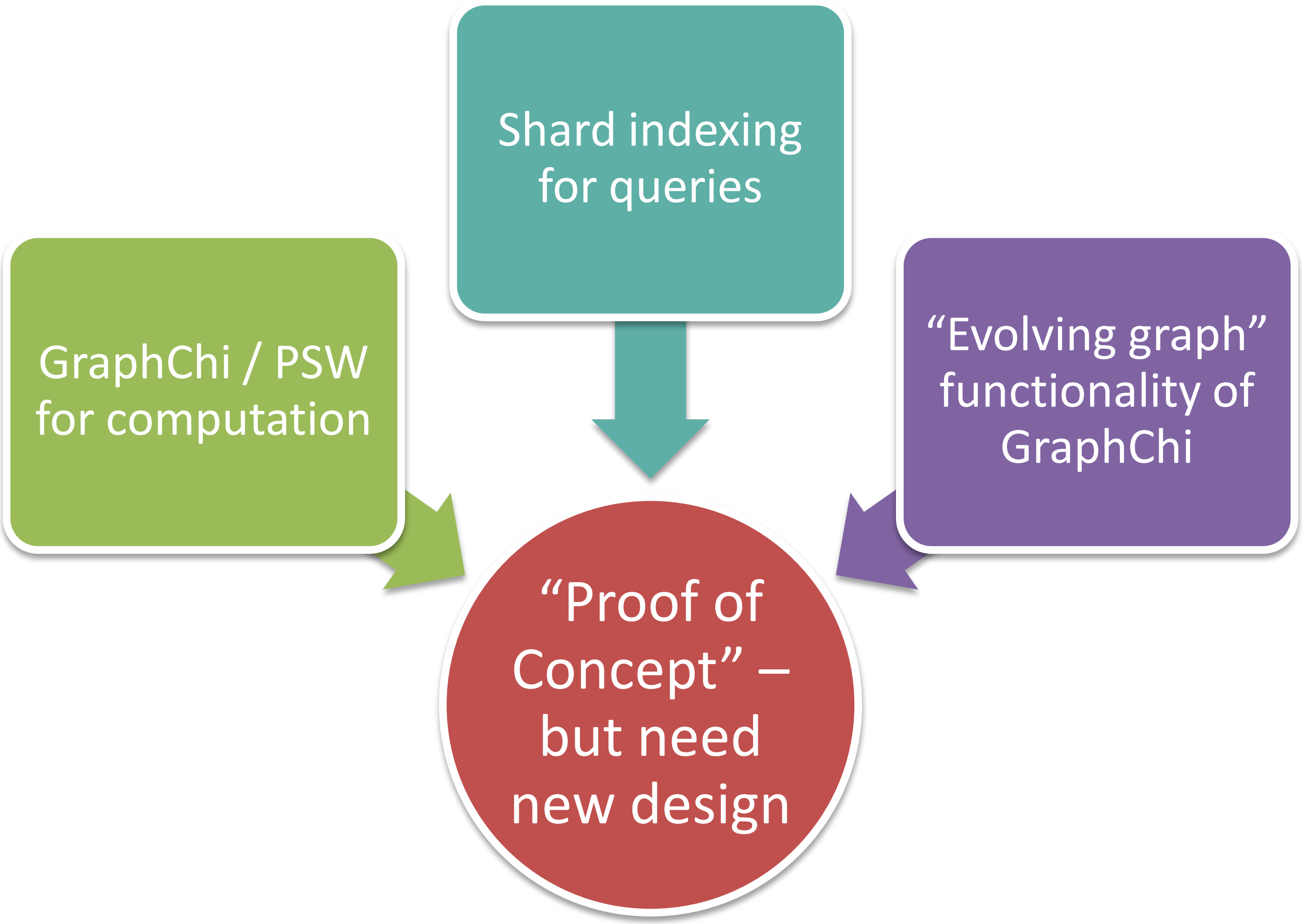
WHERE NEXT?

GRAPHCHI-DB:
LARGE-SCALE GRAPH COMPUTATION + GRAPH DATABASE ON JUST A PC

Graph Databases vs. Graph Computation Systems

- Current graph databases do not provide large-scale graph computation capabilities
  - E.g., Titan: graph computation executed outside database, using Hadoop
- **Can we have a graph database on a single machine that can store billions of edges and vertices, and do efficient graph computation?**
  - Challenge is to handle graph queries and updates while executing computation

Existing Pieces



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: Highly Visible Open-Source Project

320 ATTENDEES IN FIRST WORKSHOP

570 ATTENDEES IN SECOND WORKSHOP

100+ COMPANIES

100+ NSF PROPOSALS MENTIONING GRAPHLAB

$6.75M VC FUNDING FOR SPINOFF EFFORT

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
}
```

Implicit aggregation

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

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

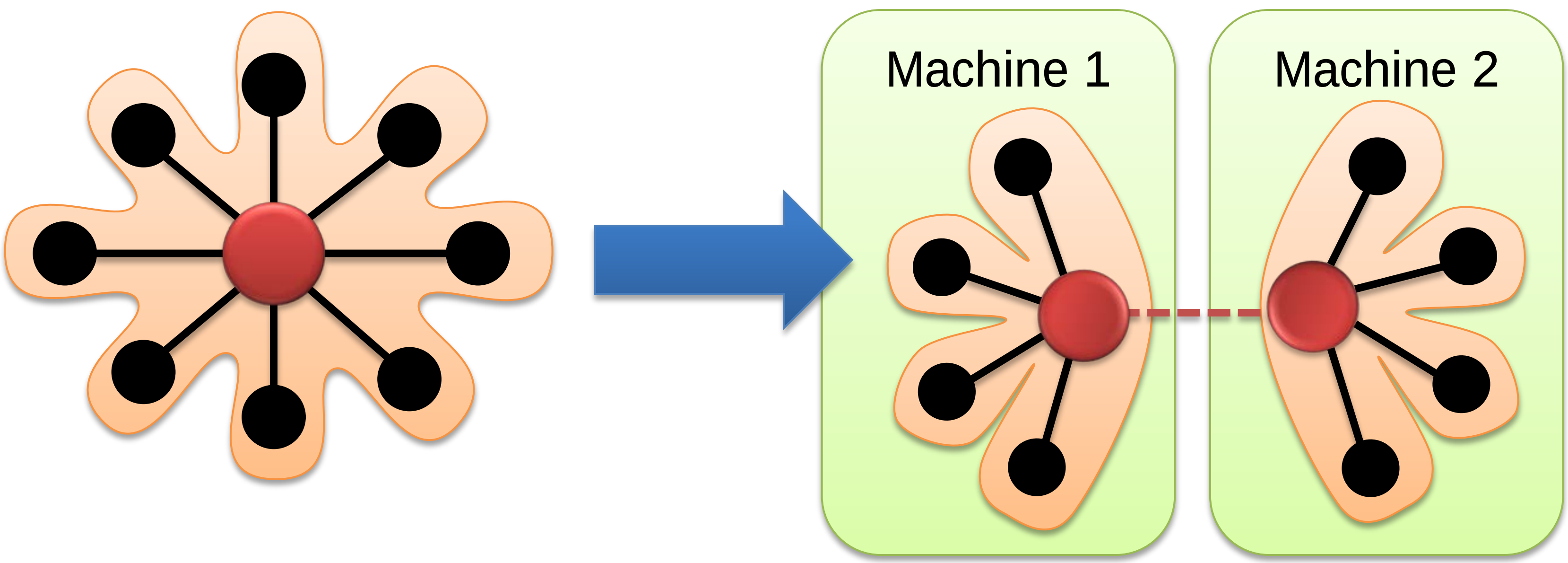
WarpGraph

USABILITY

GL3 WarpGraph Goals

Program Like GraphLab 1

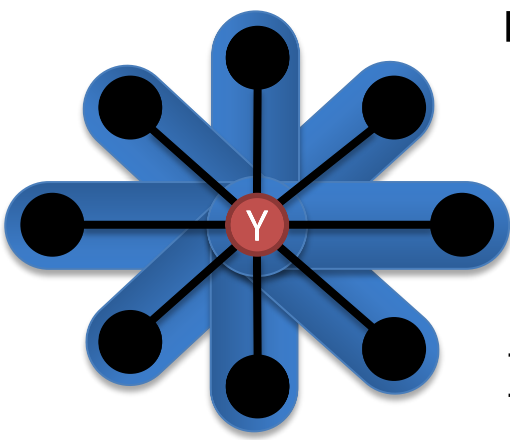
Run Like GraphLab 2



Fine-Grained Primitives

Expose Neighborhood Operations through Parallelizable Iterators

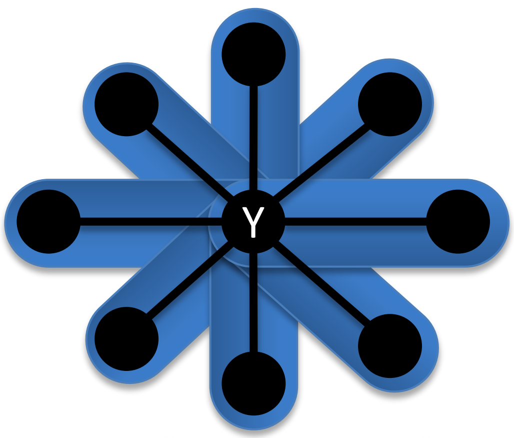
$$R[i] = 0.15 + 0.85 \boxed{\sum_{(j,i) \in E} \mathrm{w}[j,i] * R[j]}$$



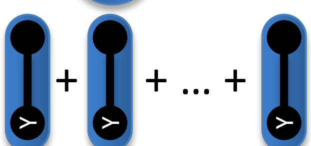

```
PageRankUpdateFunction(Y) {
    Y.pagerank = 0.15 + 0.85 *





}
```

Expressive, Extensible Neighborhood API

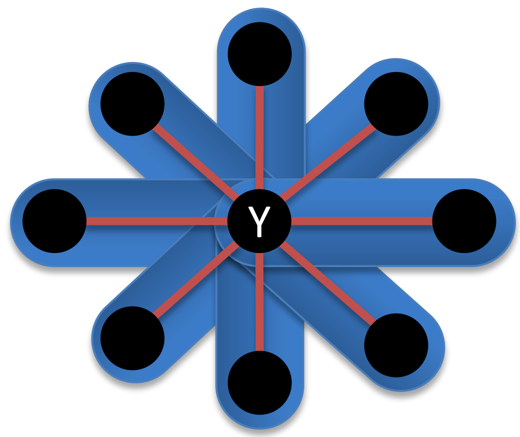
MapReduce over Neighbors



Parallel Sum

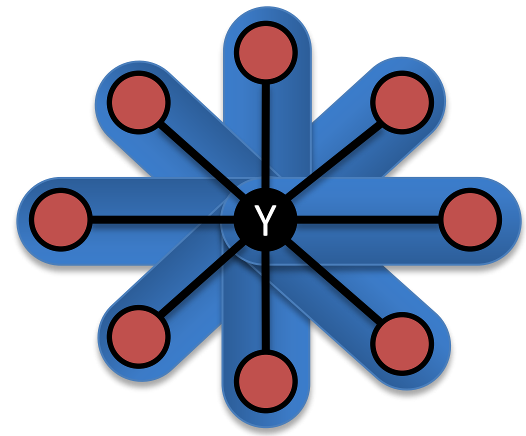


Parallel Transform Adjacent Edges



Modify adjacent edges

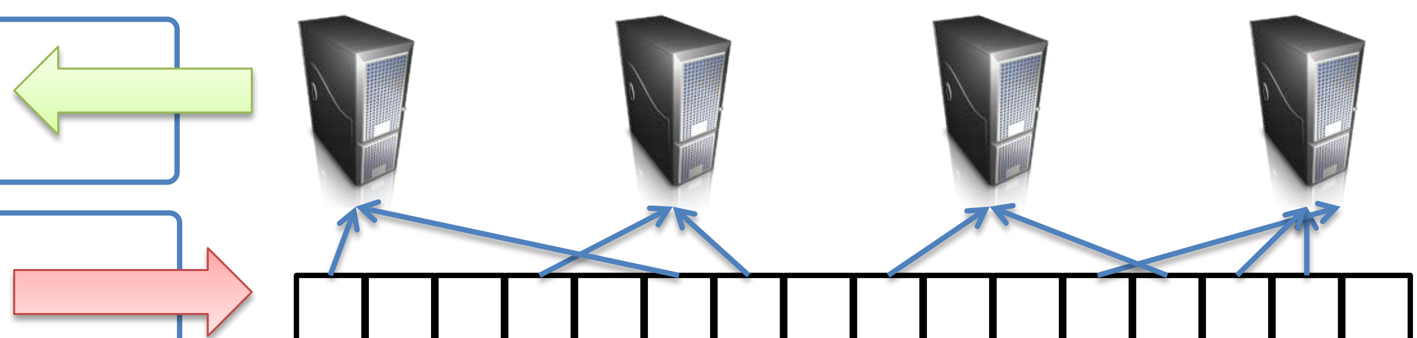
Broadcast



Schedule a selected subset of adjacent vertices

DHT Get Keys

DHT Update Keys



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

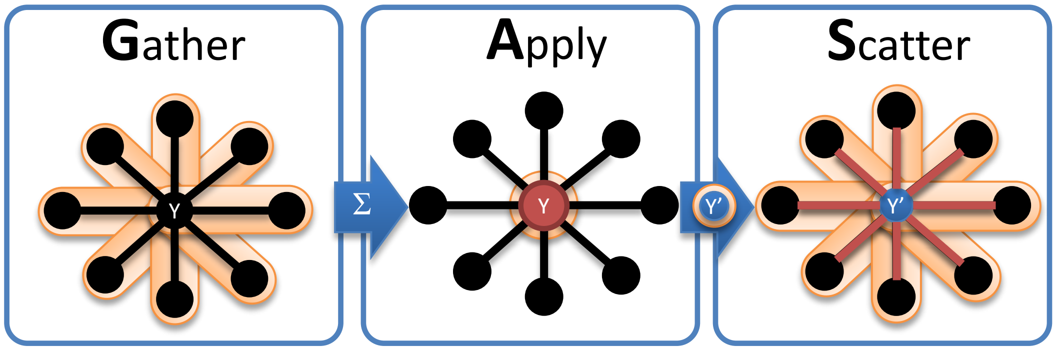
But GL3 is more expressive

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

- Multiple gathers
- Scatter before gather
- Conditional execution

GL2 PowerGraph:

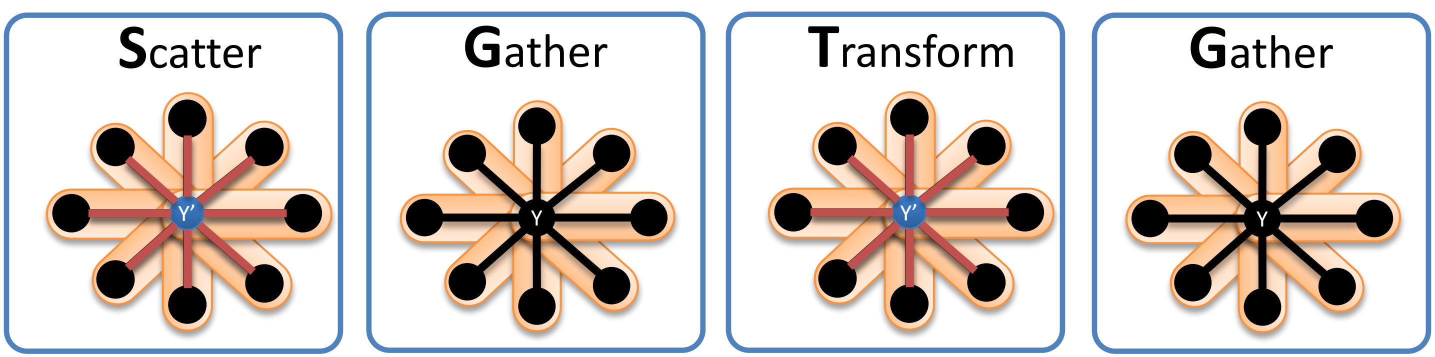
Fast because **communication** phases are very **predictable**



… repeat

GL3 WarpGraph:

Communication highly **unpredictable**



…

Risk: **High Latency**

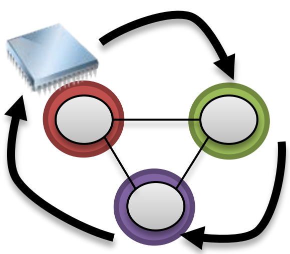
(spend all our time waiting for a reply…)

Hide Latency

Do Something Else while Waiting

Create 1000s of threads, each running an update function on a different vertex

Performance Bottleneck: Context Switching



Every cycle used in context switching is wasted

(OS context switch is slow requiring 10K-100k cycles)

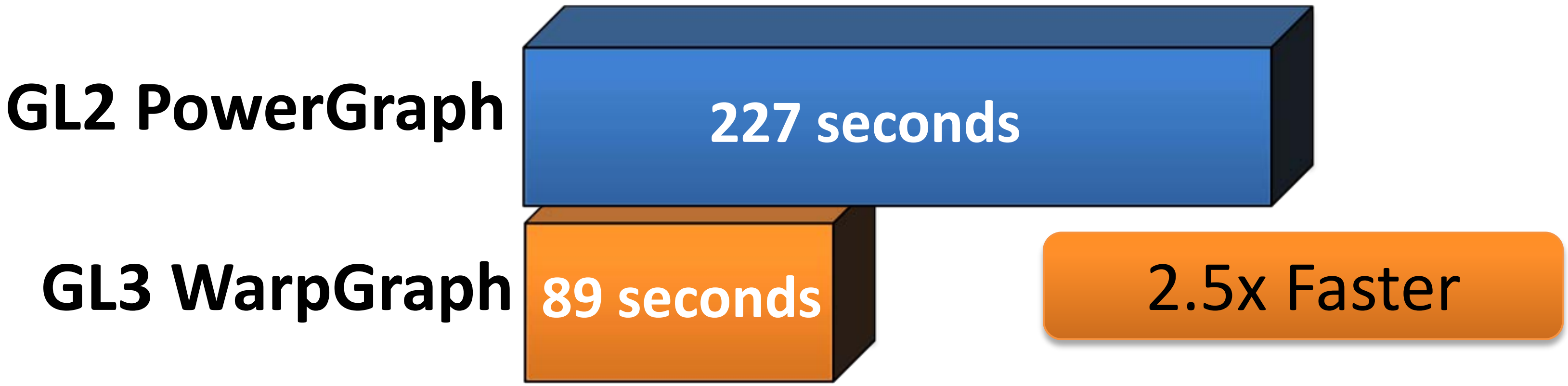
GL3 WarpGraph: Novel user-mode threading

8M context switches per second

100x faster than OS

Graph Coloring

Twitter Graph: 41M Vertices 1.4B Edges



WarpGraph outperforms PowerGraph with simpler code

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

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… :-)

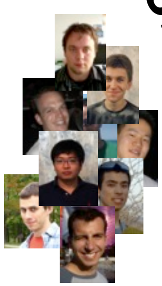
Usability for Whom???

GL2
PowerGraph

GL3
WarpGraph

…

Anyone in my lab



Anyone in this hall

Any Data Scientist

Anyone with Big Data

Machine Learning
PHASE 3

USABILITY

Exciting Time to Work in ML

With Big Data, I'll take over the world!!!

We met because of Big Data

Why won't Big Data read my mind???

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

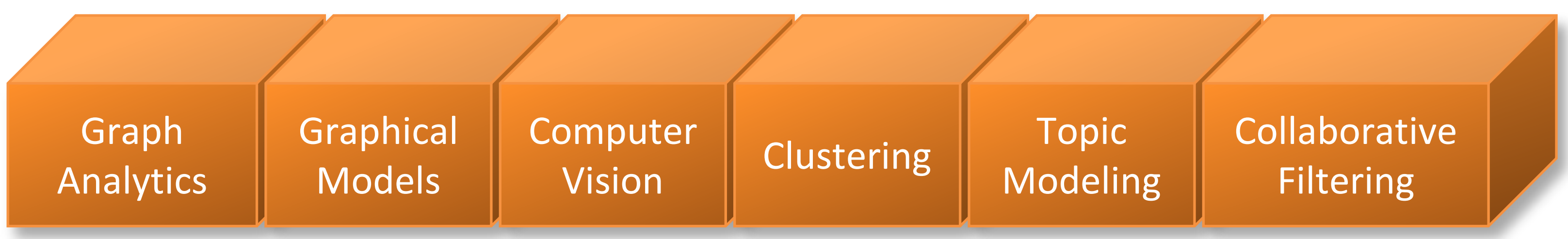
Step 3

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

GraphLab

v1 Possibility

v2 Scalability

v3 Usability

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