GraphLab Carnegie Mellon & GraphLab2 PowerGraph

Large–Scale Graph Machine Learning

Presenter: Susu Sun

Graduate School of Computer Science
Saarland University

May 29 , 2015

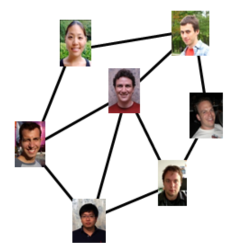
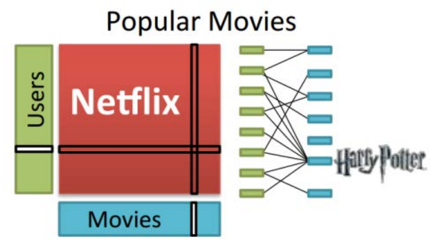
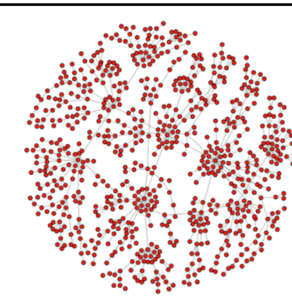
Outline

- **Properties** of Graph based machine learning
- ***Big*** : Data-parallel :Map Reduce
- ***Dependency***: Graph-parallel: Pregel
- ***Efficiency***: Asychronous Graph- Parallel: GraphLab
- ***PowerLaw Vertex*** : GraphLab 2.1- PowerGraph

Outline

- **Properties** of Graph based machine learning
- ***Big*** : Data-parallel :Map Reduce
- ***Dependency***: Graph-parallel: Pregel
- ***Efficiency***: Asychronous Graph- Parallel: GraphLab
- ***PowerLaw Vertex*** : GraphLab 2.1- PowerGraph

Characteristic of Graphs

Properties:	**Examples**
Graphs are *ubiquitous* Social media, science , advertising,web…	Social media Science Advertising Web
Graphs are *big* billions of vertices and edges and rich metadata	28 Million Wikipedia Pages 1 Billion Facebook Users 6 Billion Flickr Photos 72 Hours a Minute YouTube
***Dependency* is important** Graphs encode relationships between People, Facts , Products , Ideas, Interests	 
Vertices in natural graphs are ***Power-law***	 

MLDM(Machine learning and Data Mining) Algorithm

Requires :

Parallel Computation ----*Big*

- Run the computation in parallel

Graph Structured Computation ----*Dependency*

- Modeling dependencies between data

Asynchronous Iterative Computation ----*Efficiency*

- Asynchronous computation can accelerate convergence

Power-law Vertex ----*Power-Law vertex*

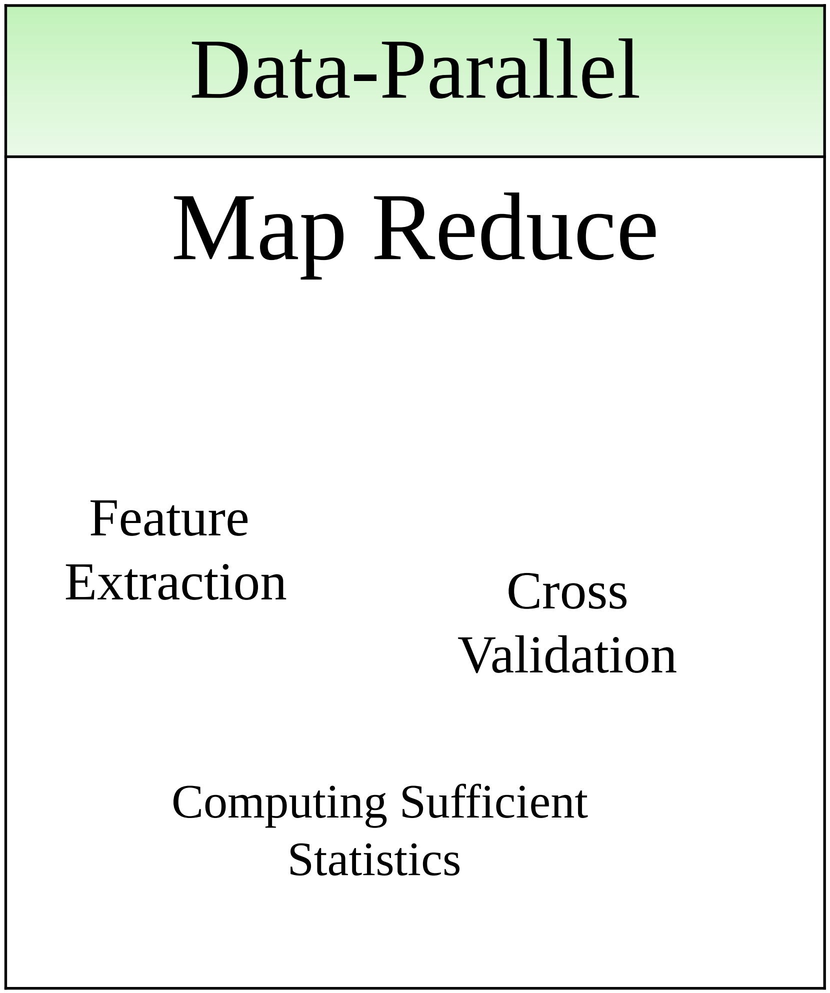
- Efficiently compute power-law degree vertex

Big Learning

Outline

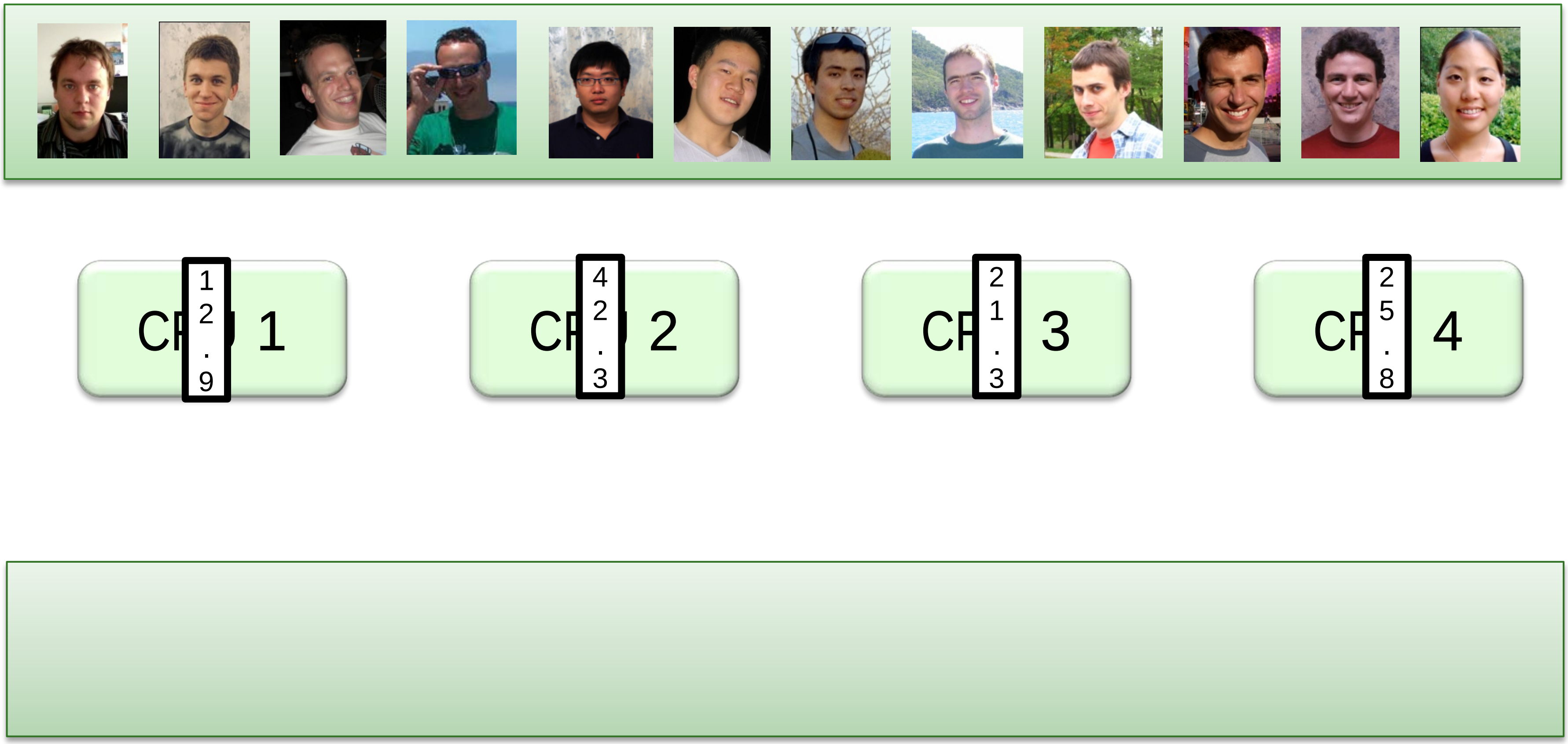
- **Properties** of Graph based machine learning
- ***Big*** : Data-parallel :Map Reduce
- ***Dependency***: Graph-parallel: Pregel
- ***Efficiency***: Asychronous Graph- Parallel: GraphLab
- ***PowerLaw Vertex*** : GraphLab 2.1- PowerGraph

Data Parallel ML: MapReduce



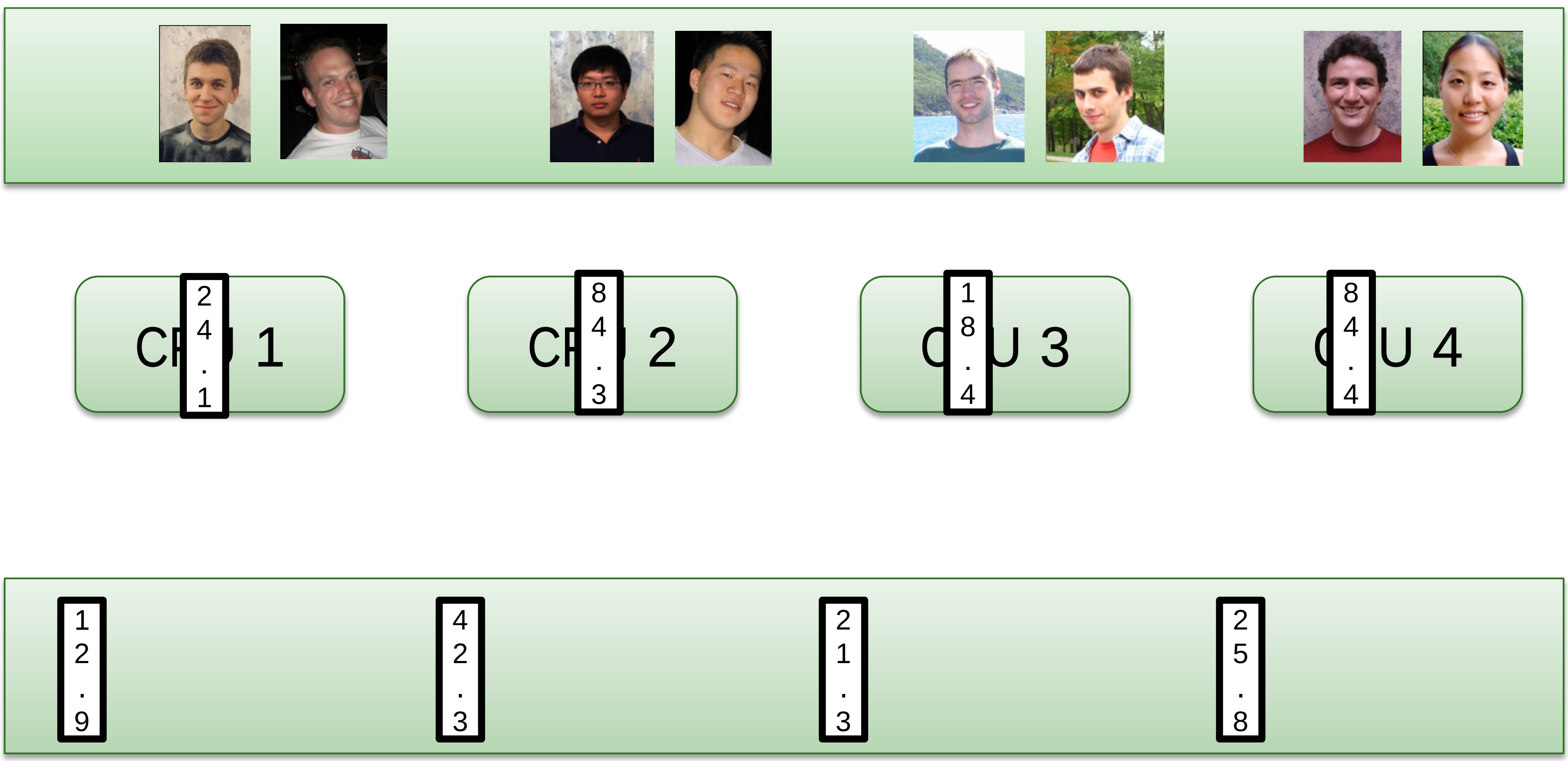
Data Parallel ML: MapReduce

Map phrase: Image Features Extraction



Data Parallel ML: MapReduce

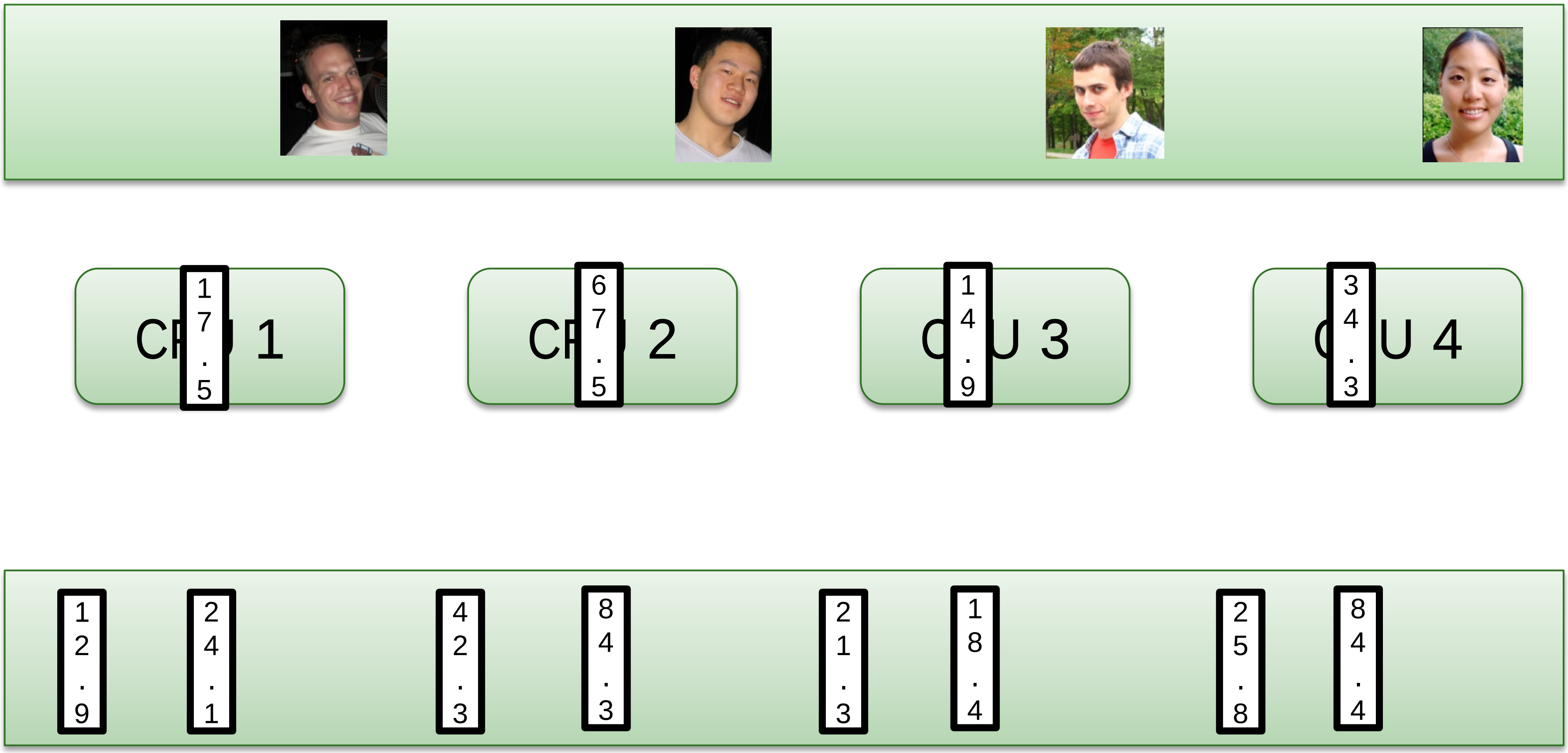
Map phrase: Image Features Extraction



cloud.berkeley.edu/data/graphlab

Data Parallel ML: MapReduce

Map phrase: Embarrassingly Parallel independent computation

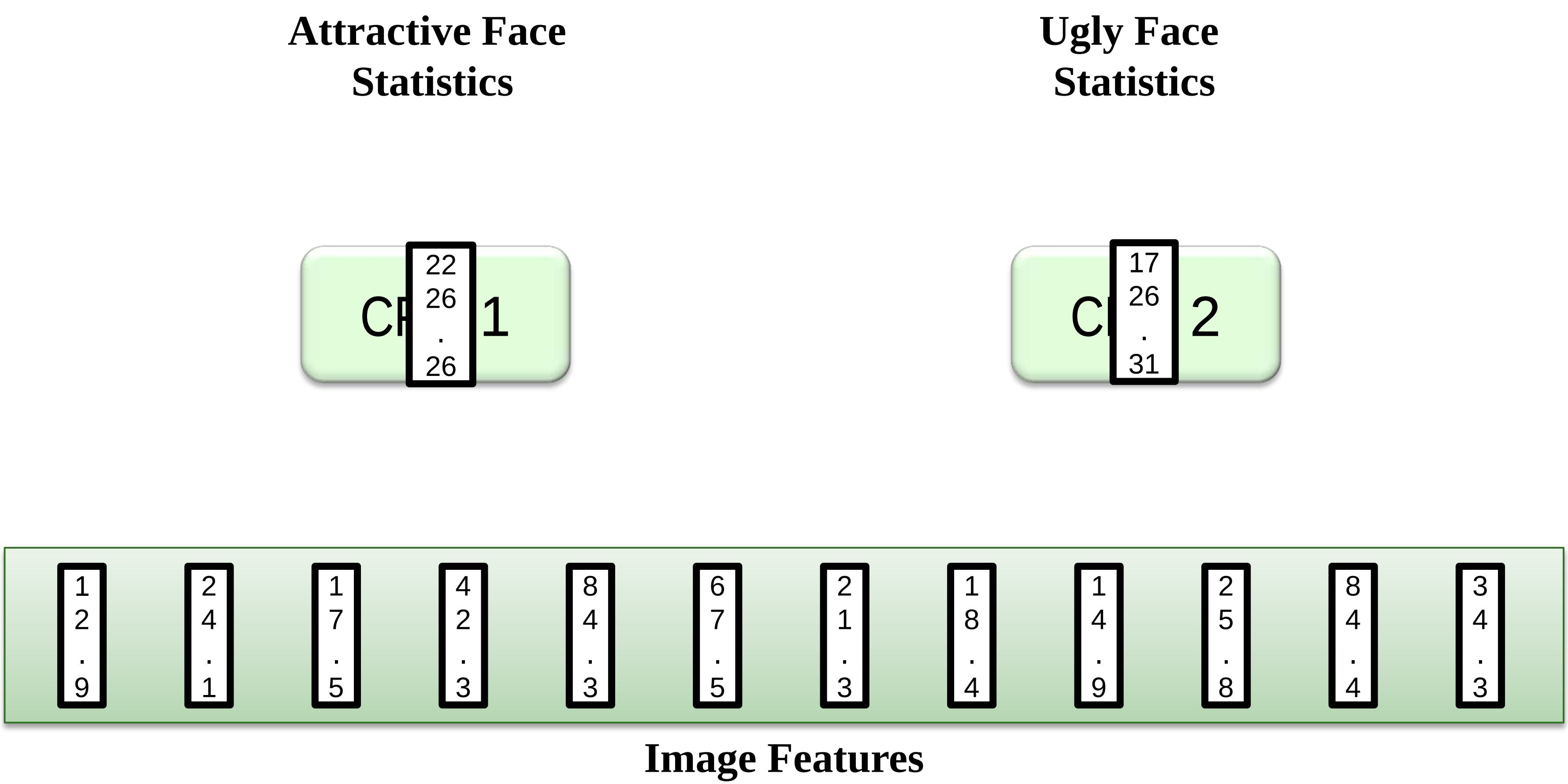


No Communication needed !

cloud.berkeley.edu/data/graphlab

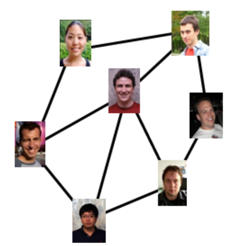
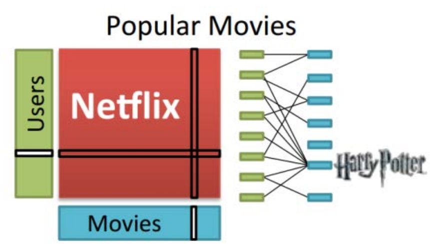
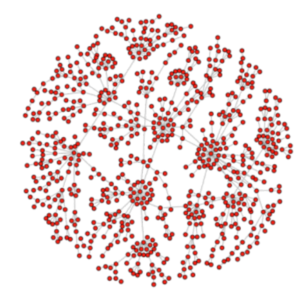
Data Parallel ML: MapReduce

Reduce phrase: Image classification

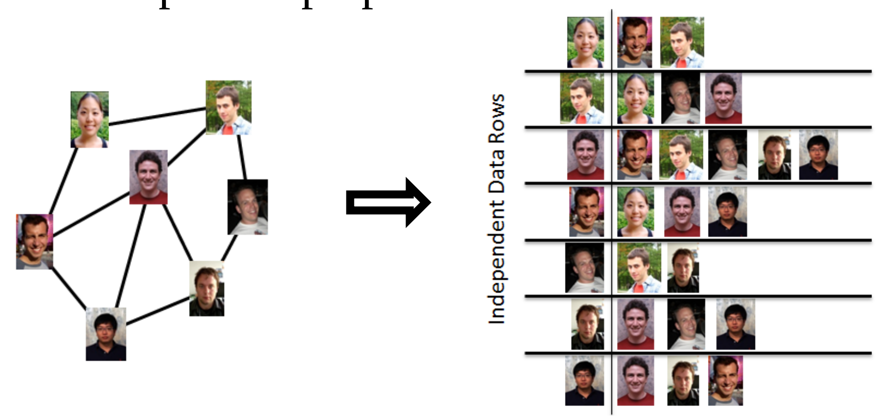


cloud.berkeley.edu/data/graphlab

Characteristic of Graphs

Properties:	**Examples**
Graphs are *ubiquitous* Social media, science , advertising,web…	Social media Science Advertising Web
Graphs are *big* billions of vertices and edges and rich metadata SOLVED	28 Million Wikipedia Pages 1 Billion Facebook Users 6 Billion Flickr Photos 72 Hours a Minute YouTube
***Dependency* is important** Graphs encode relationships between People, Facts , Products , Ideas, Interests	 
Vertices in natural graphs are ***Power-law***	 

MapReduce: Pros & Cons

Pros	Cons
Simplicity of the model: Programmers specifies few simple methods that focuses on the functionality not on the parallelism	**Restricted programming constructs:** only map & reduce
Scalability: Scales easily for large number of clusters with thousands of machines	**Does not scale well for dependent tasks:** for example Graph problems 
Applicability: Applicable to many different systems and a wide variety of problems	**Does not scale well for iterative algorithms:** iteration is very common in machine learning

ML Tasks Beyond Data-Parallelism

Data-Parallel ⇨	Graph-Parallel
Map Reduce Feature Extraction Cross Validation Computing Sufficient Statistics	Lasso Label Propagation Kernel Methods Belief Propagation Tensor Factorization PageRank Deep Belief Networks Neural Networks

Outline

- **Properties** of Graph based machine learning
- ***Big*** : Data-parallel :Map Reduce
- ***Dependency***: Graph-parallel: Pregel
- ***Efficiency***: Asychronous Graph- Parallel: GraphLab
- ***PowerLaw Vertex***: GraphLab 2.1- PowerGraph

Graph-Parallel computation?

"Think like a vertex."
-Malewicz et al. [SIGMOD'10]

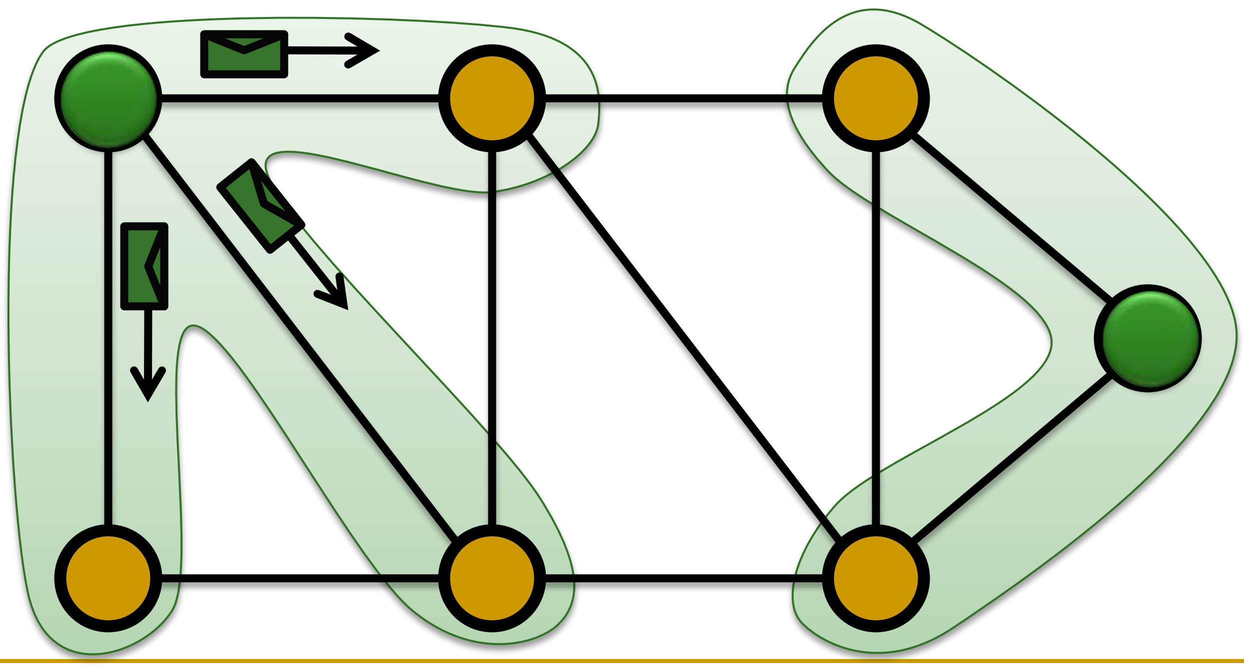
The Graph-Parallel Abstraction

A user-defined **Vertex-Program** runs on each vertex

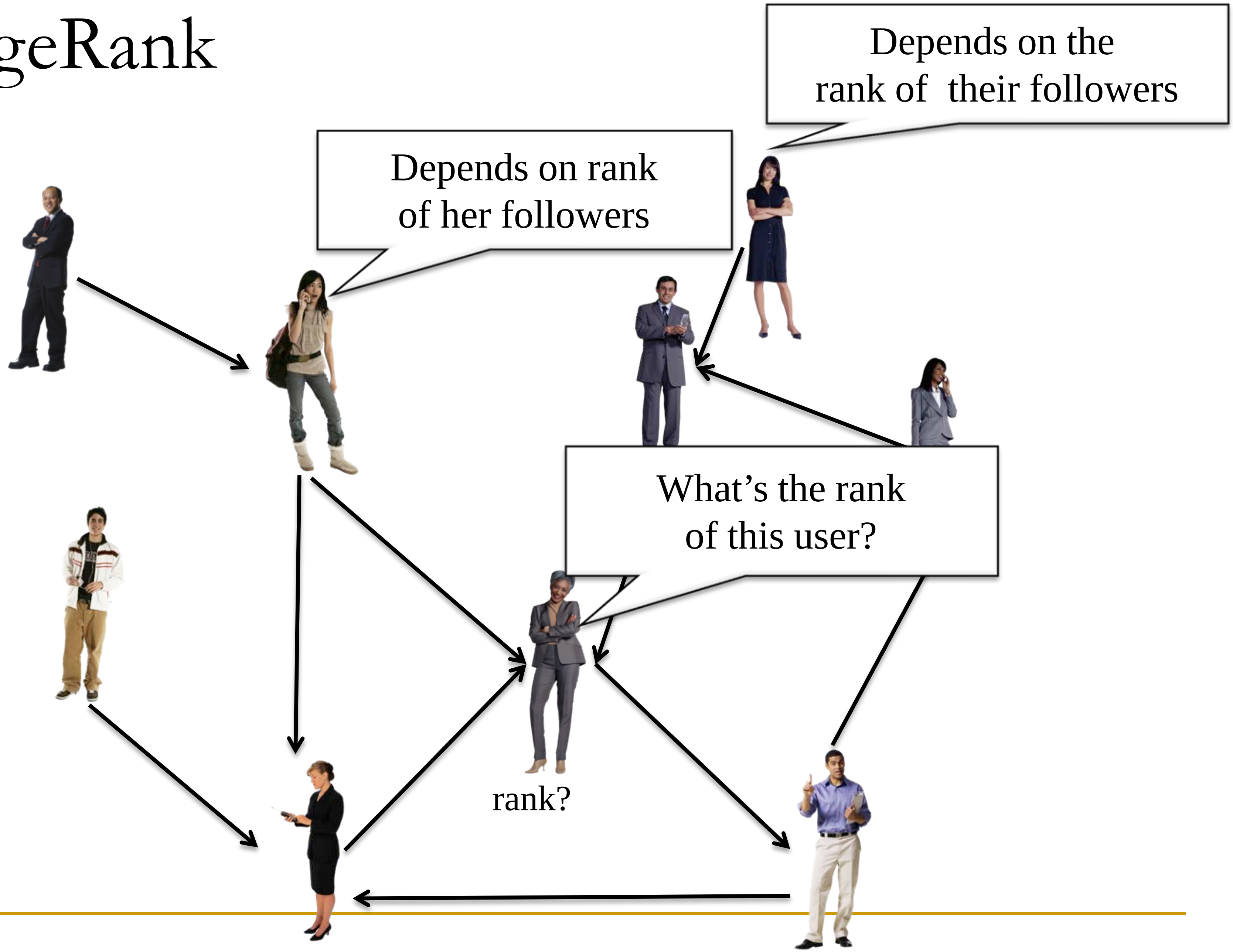
Graph constrains **interaction** along edges

- Using **messages** (e.g. **Pregel** [PODC’09, SIGMOD’10])
- Through **shared state** (e.g., **GraphLab** [UAI’10, VLDB’12])

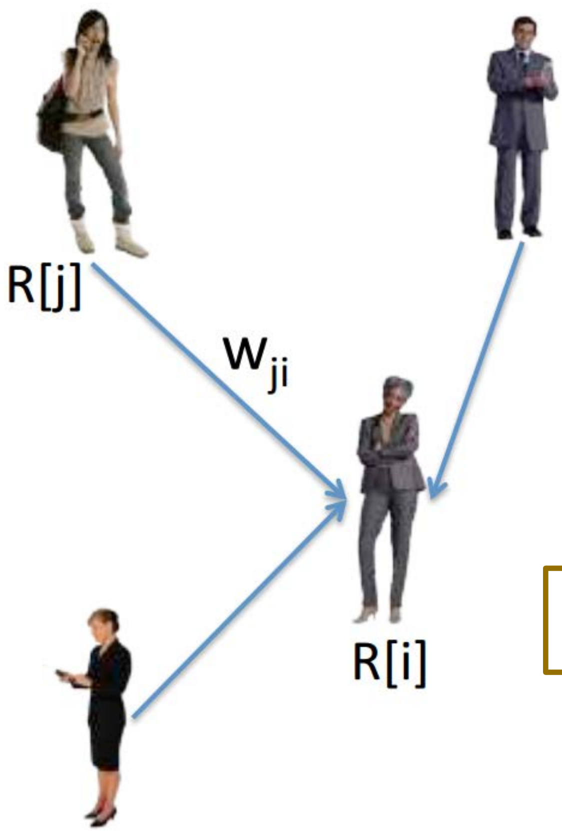
Parallelism: run multiple vertex programs simultaneously



PageRank


Depends on the
rank of their followers
Depends on rank
of her followers
What's the rank
of this user?
rank?

Page Rank Iteration



Iterate until convergence:

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

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

Rank of user *i*

Weighted sum of neighbors' ranks

α is the random reset probability

w_{ji} Is the prob. transitioning from j to i

Pregel Abstraction

Vertex-Programs interact by sending **messages**

Pregel_PageRank(i, **messages**) :

// Receive all the messages

total = 0

foreach(msg in **messages**) :

  total = total + msg

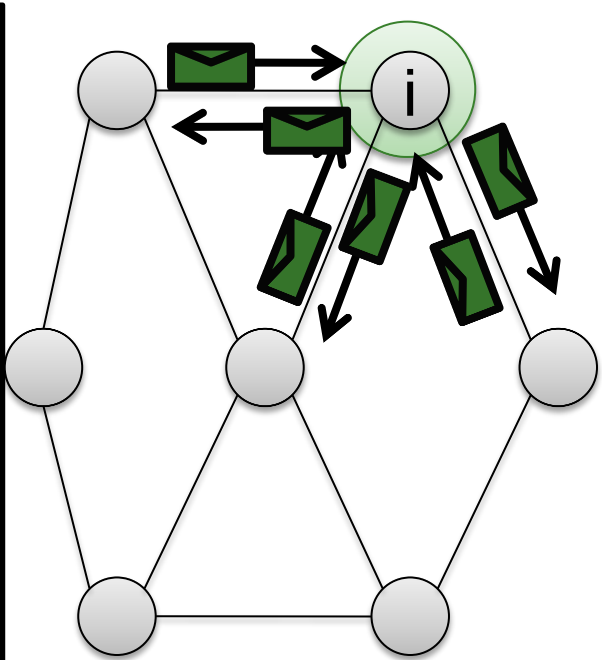
// Update the rank of this vertex

R[i] = 0.15 + total

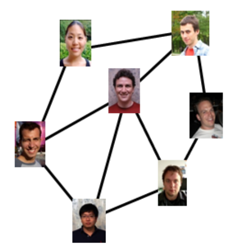
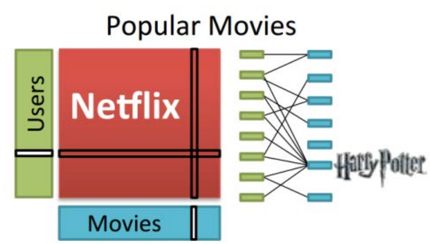
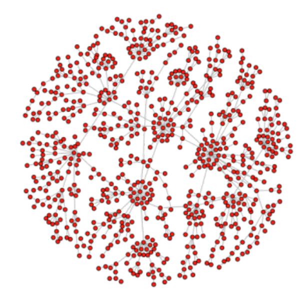
// Send new messages to neighbors

foreach(j in out_neighbors[i]) :

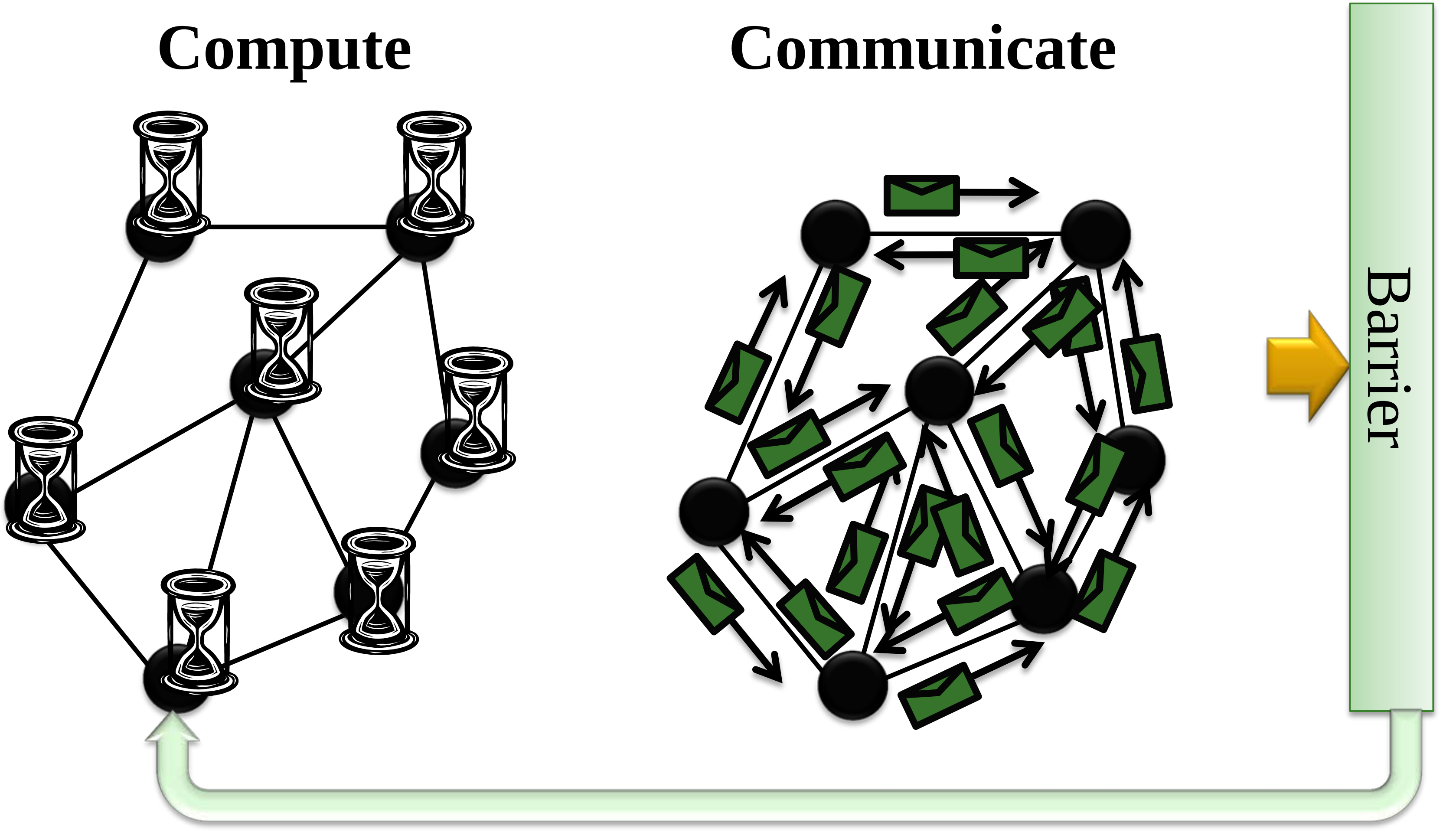
  Send msg(R[i] * w_{ij}**) to vertex j**



Characteristic of Graphs

Properties:	**Examples**
Graphs are *ubiquitous* Social media, science , advertising,web…	Social media · Science · Advertising · Web
Graphs are *big* billions of vertices and edges and rich metadata SOLVED	28 Million Wikipedia Pages · 1 Billion Facebook Users · 6 Billion Flickr Photos · 72 Hours a Minute YouTube
***Dependency* is important** Graphs encode relationships between People, Facts , Products , Ideas, Interests SOLVED	 
Vertices in natural graphs are ***Power-law***	 

Pregel: Bulk Synchronous Parallel(BSP) Model



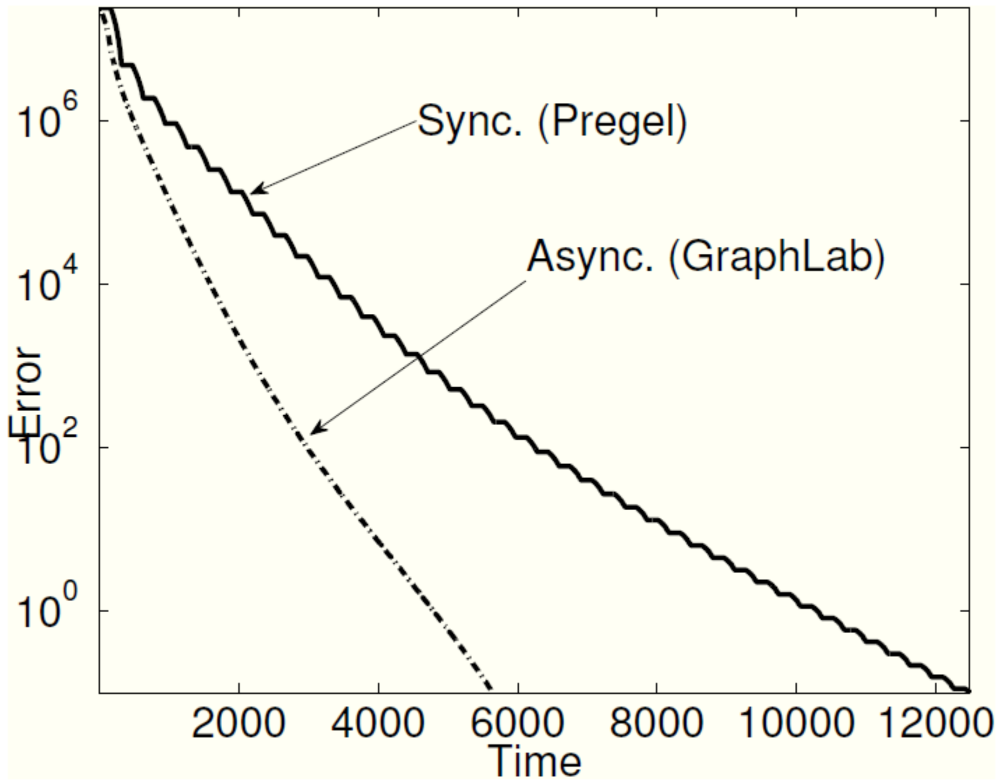
Bulk synchronous computation can be highly inefficient

[Malewicz et al. ‘2010] http://graphlab.org/powergraph-presented-at-osdi/

Synchronous vs Asynchronous

Synchronous computation can be inefficient

Graph: 25M vertex,
355 edge
16 processors



Async vs Sync PageRank

Tradeoffs of the BSP Model

Pros

- Scales better than Map Reduce for Graphs
- Relatively easy to build

Cons

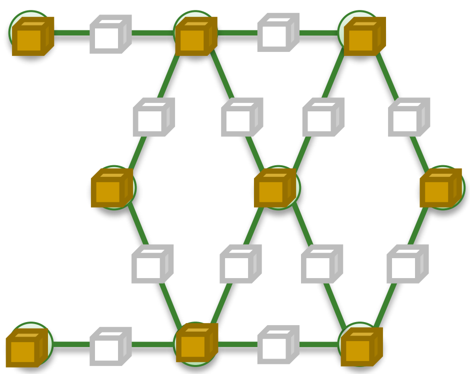
- Inefficient if different regions of the graph converge at different speed
- Runtime of each phase is determined by the slowest machine
- Synchronous computation can be inefficient!

Outline

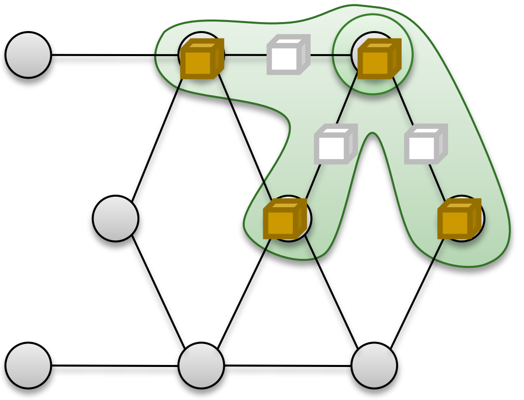
- **Properties** of Graph based machine learning
- ***Big*** : Data-parallel :Map Reduce
- ***Dependency***: Graph-parallel: Pregel
- ***Efficiency***: Asychronous Graph-Parallel: GraphLab
- ***PowerLaw Vertex*** : GraphLab 2.1- PowerGraph

GraphLab Framework

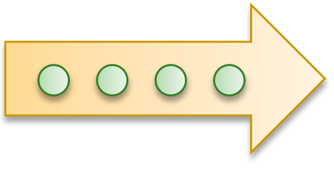
Graph Based
Data Representation



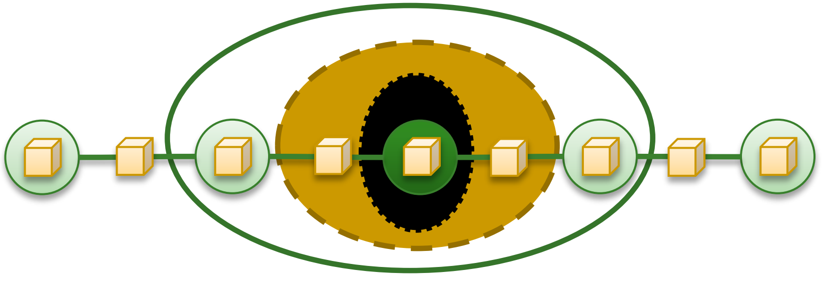
Update Functions
User Computation



Scheduler



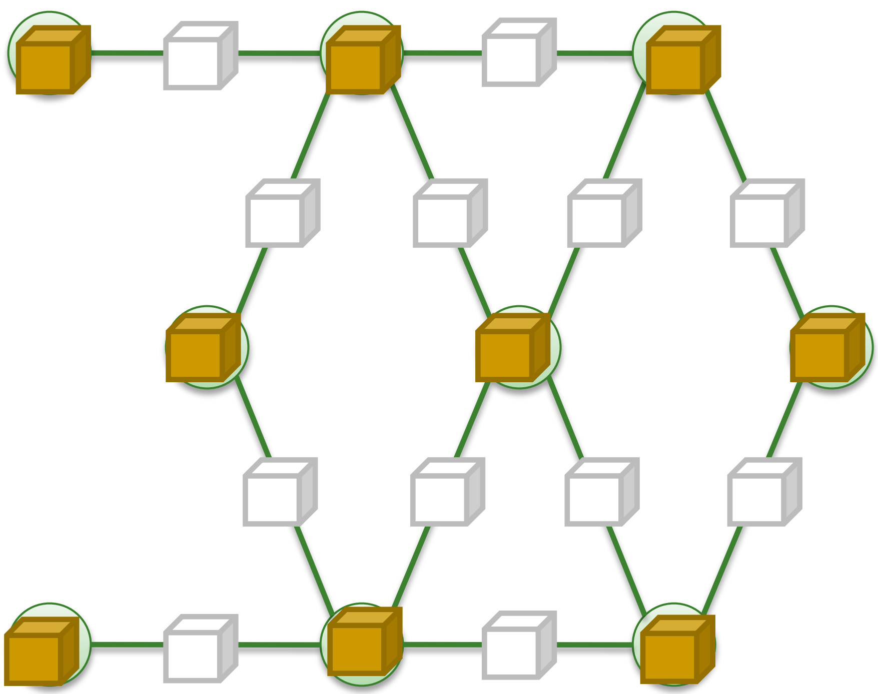
Consistency Model



Data Graph

Date Graph: a directed graph G=(V, E, D)

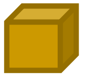
Data D refers to model parameters, algorithm states and other related data.



Graph:

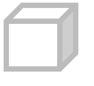
- Social Network

Vertex Data:



- User profile text
- Current interests estimates

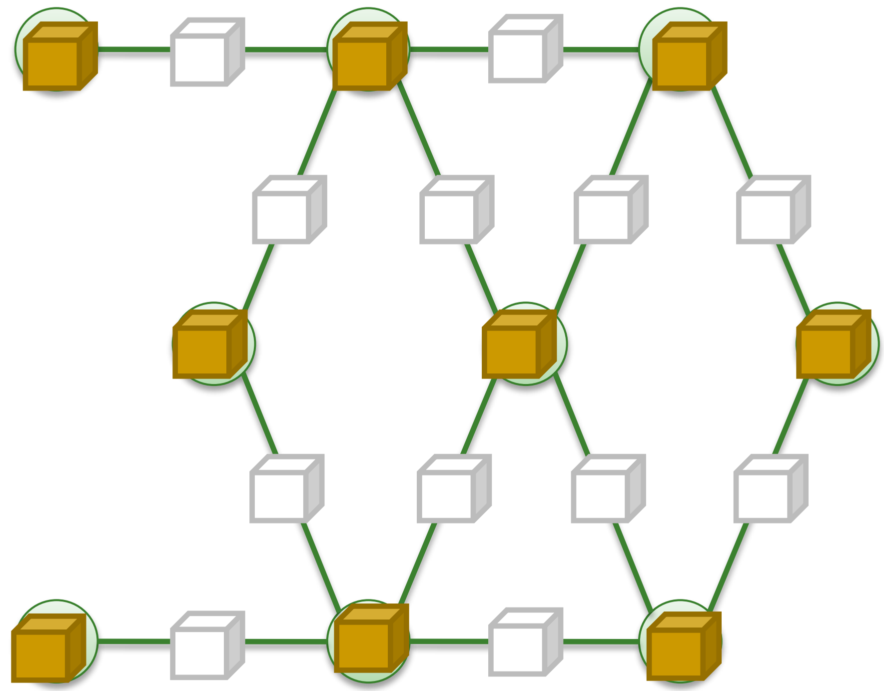
Edge Data:



- Similarity weights

Data Graph

PageRank : G=(V, E, D)



• Each vertex (*V*) corresponds to a webpage

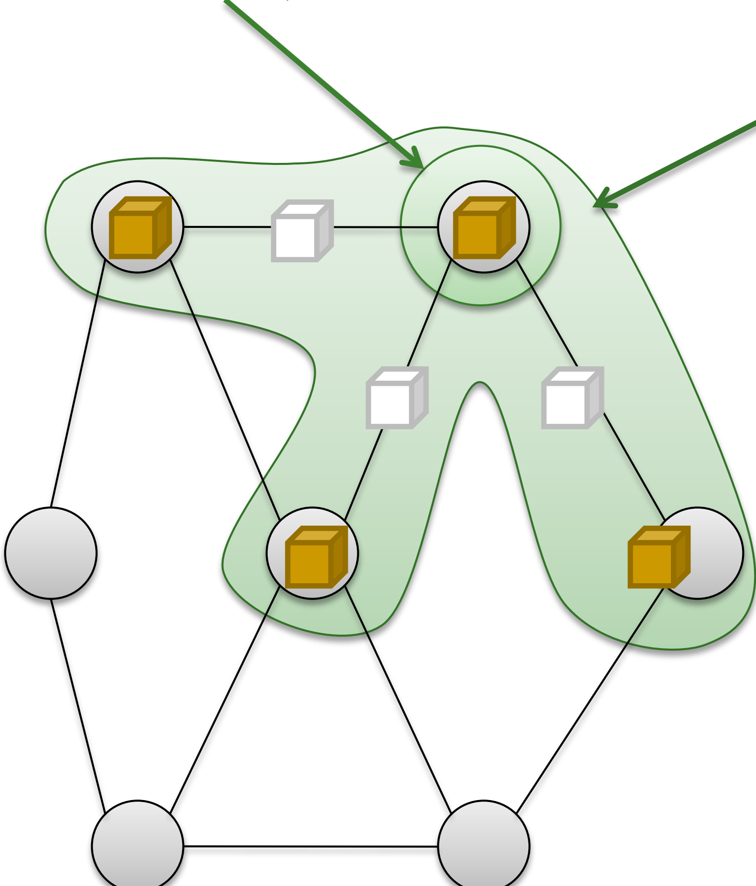
• Each edge (*U*,*V*) corresponds to a link from (*U* -> *V*)

•Vertex data D_V stores the rank of the webpage R(V)

•Edge data $D_{U\text{->}V}$ stores the weight of the link (U->V)

Update Functions

update function : user defined program which when applied to a **vertex**, transforms the data in the **scope**of the vertex



pagerank(i, scope){

// Get Neighborhood data

(R[i], W_{ij}, R[j]) ←scope;

// Update the vertex data

$$R[i] \leftarrow \alpha + (1-\alpha)\sum_{j\in N[i]} W_{ij}\times R[j]$$

// Reschedule Neighbors if needed

if R[i] changes then

reschedule_neighbors_of(i);

}

Scheduling

The **scheduler** determines the order that vertices are updated.

GraphLab Execution Model (i, messages) :

Input: Data Graph G=(V,E,D)
Input: Update Function f
Input: Initial vertex set $\mathrm{T} = \{v_1, v_2, \ldots\}$

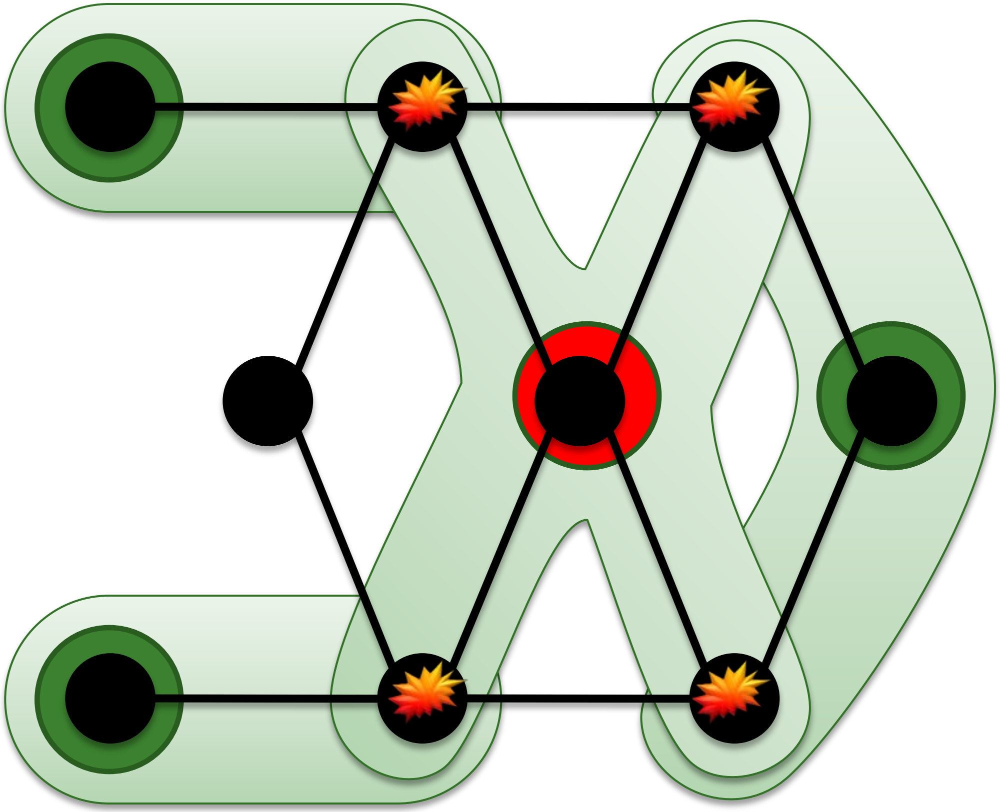
While T is not Empty do
(1) $v \leftarrow \mathrm{Re}\,moveNext(T)$
(2) $(T', S_v) \leftarrow f(v, S_v)$
(3) $T \leftarrow T \cup T'$

Output: Modified Data Graph $G = (V, E, D')$

Only Requirement :All vertex in T are eventually executed

Ensuring Race-Free Code

How much can computation **overlap**?

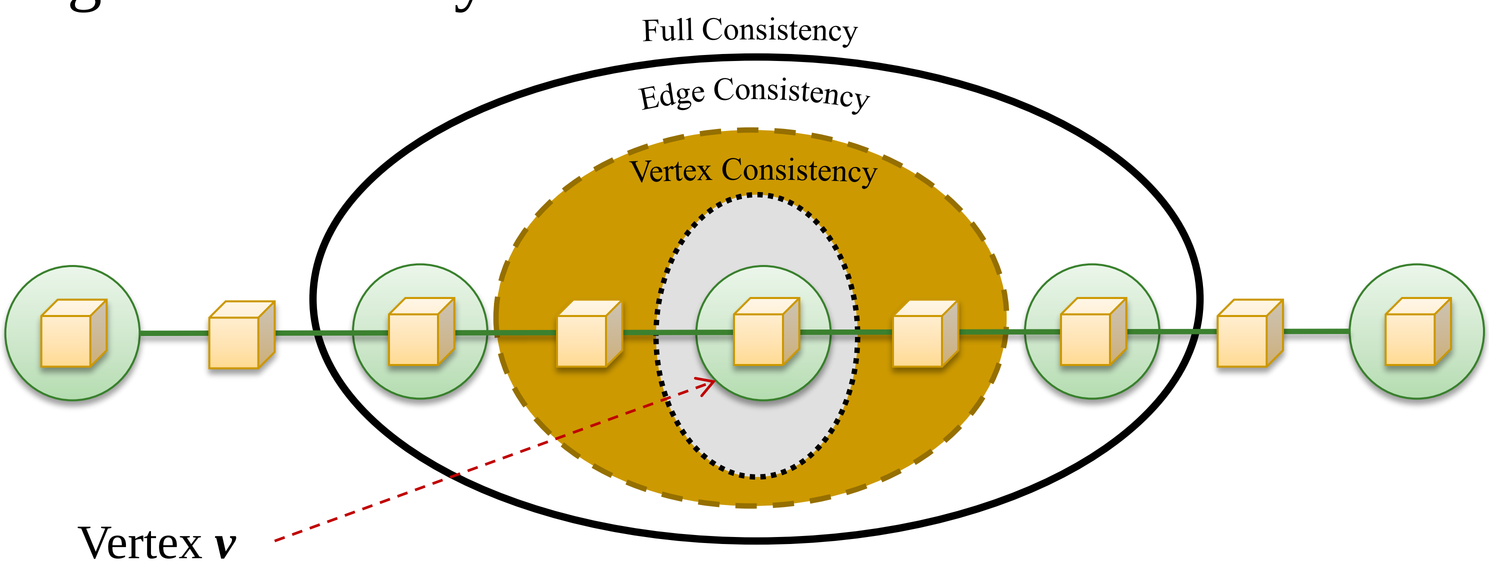


Consistency Models

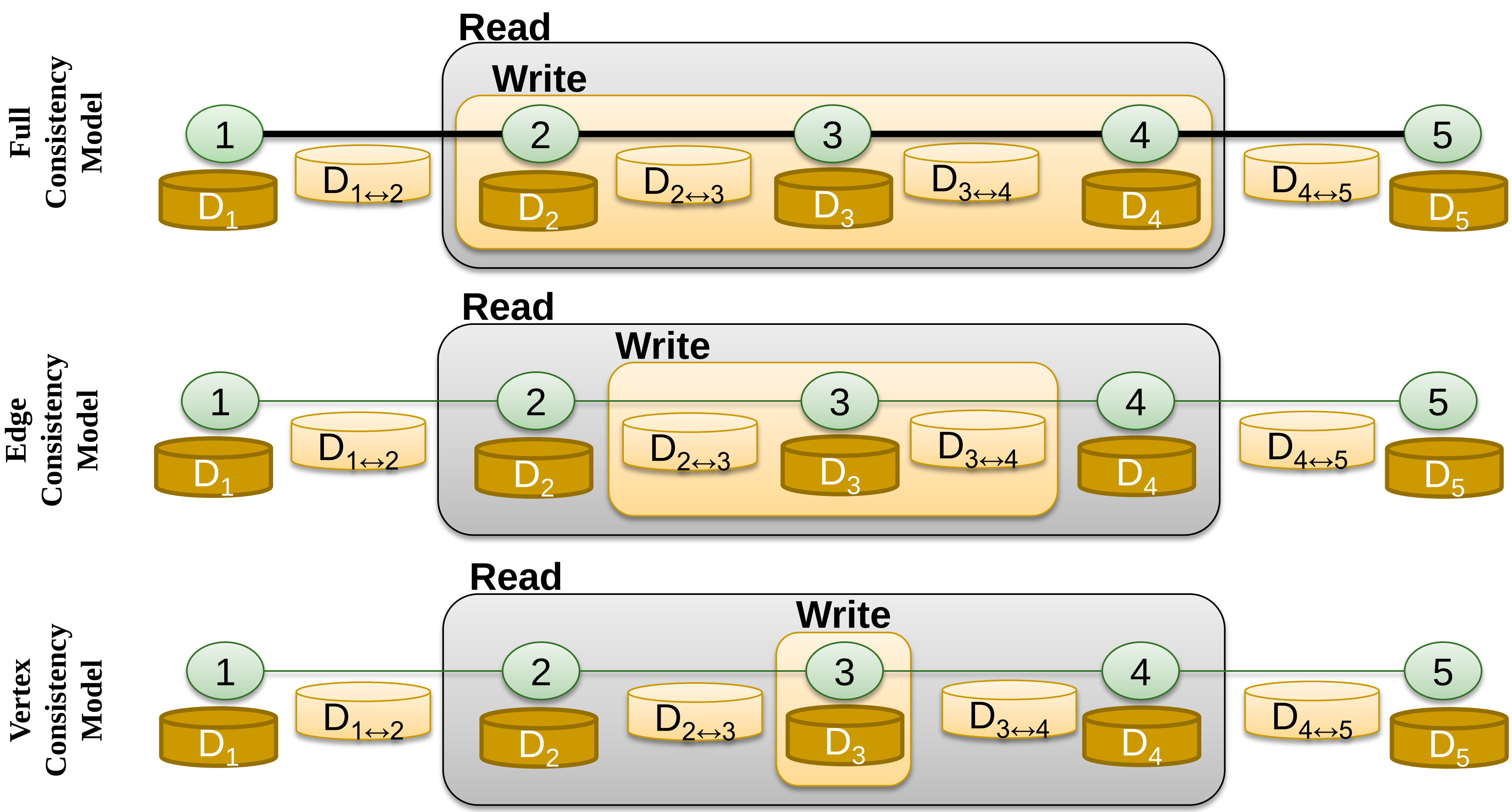
Guarantee sequential consistency for all update functions

User –defined consistency models:

- Full consistency
- Vertex Consistency
- Edge Consistency



Consistency Model in GraphLab



Consistency vs Parallelism

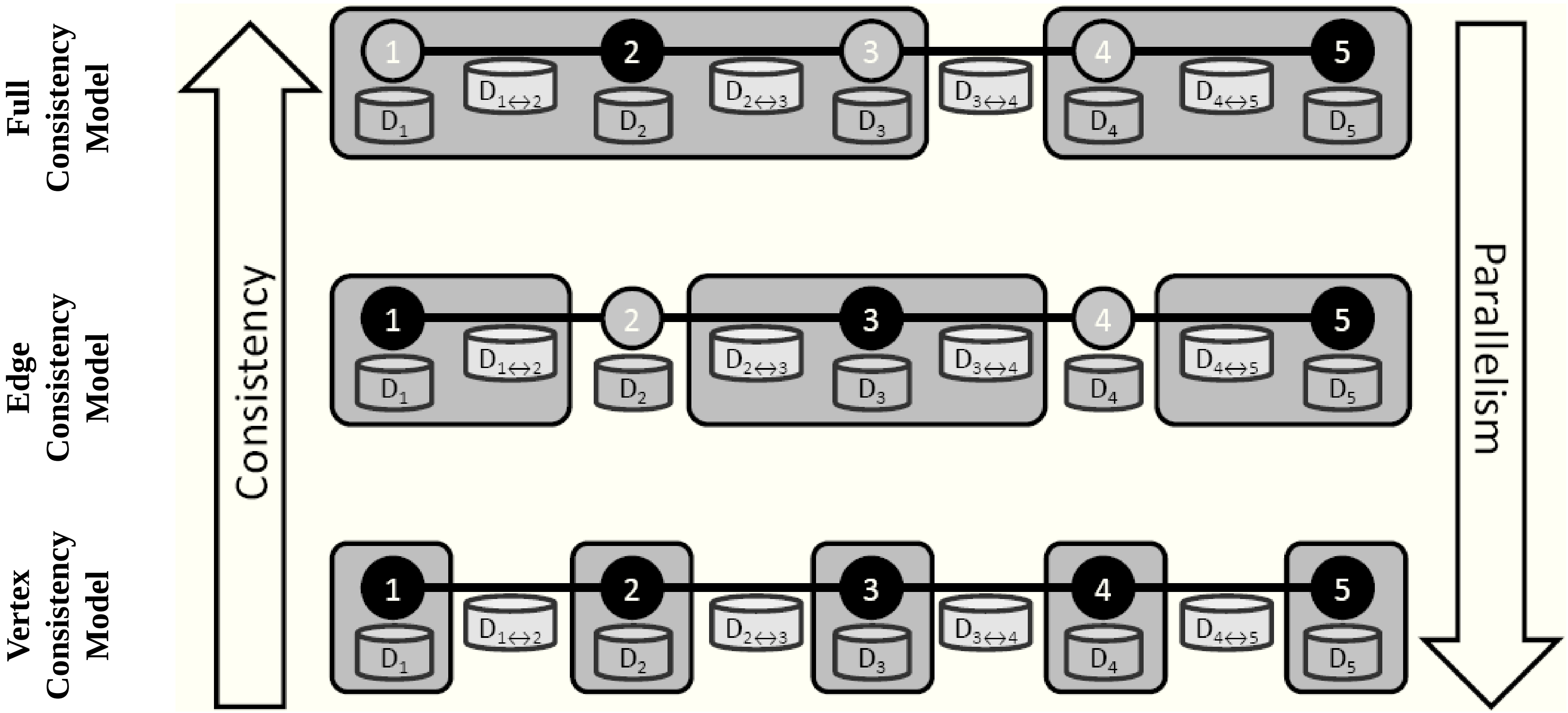


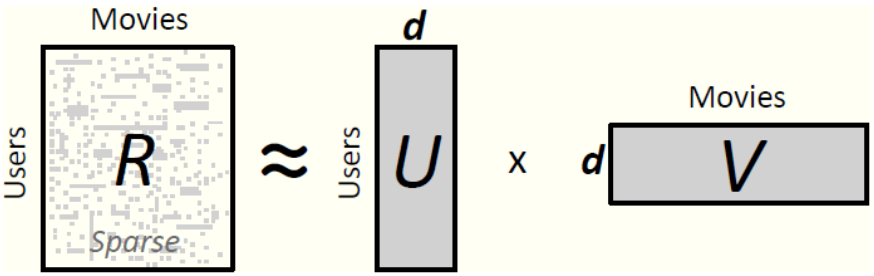
Figure 2: Consistency and Parallelism

GraphLab System Evaluation

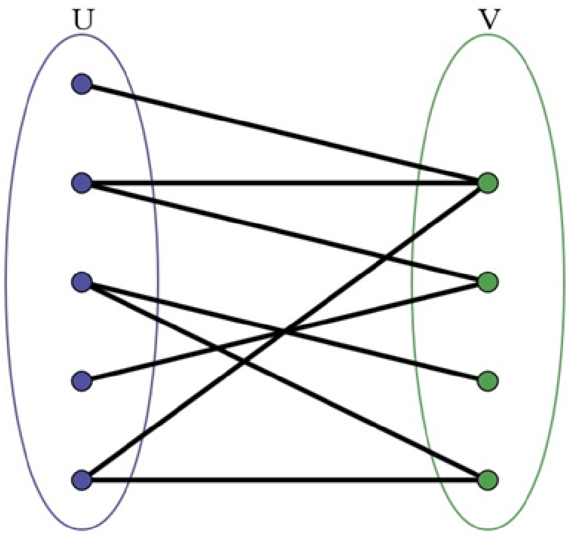
Experiments---Netflix Movie Recommendation

Task: collaborative filtering
Recommend movies based on the ratings of similar user.

Algorithm:
Alternating Least Squares Matrix Factorization(**ALS**)

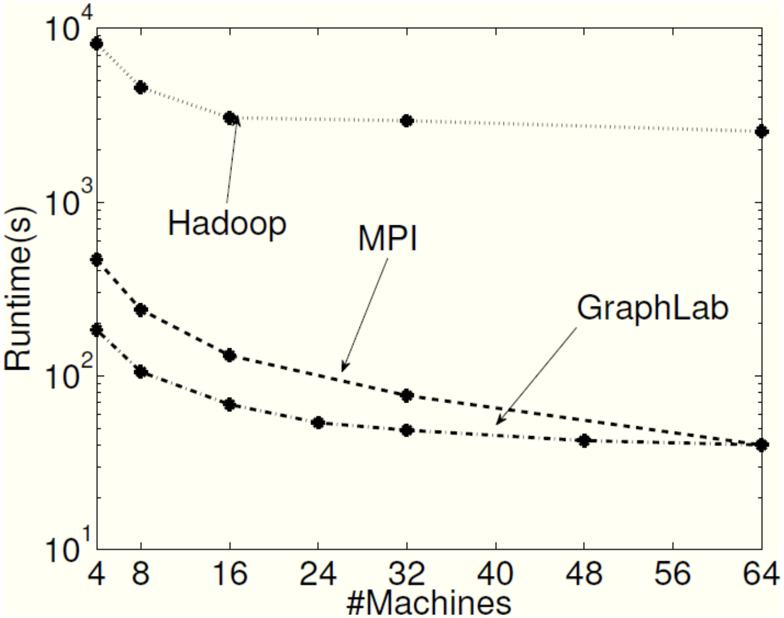


GraphLab Model:
R: bipartite graph connecting each user and the moves they rated.
Edge: rating for a movie-user pair
Vertex: user and movie data corresponding to row in U and column in V
Update Function: recompute the d length vector for each each vertex by reading the d length vectors on adjacent vertices and predict the edge value

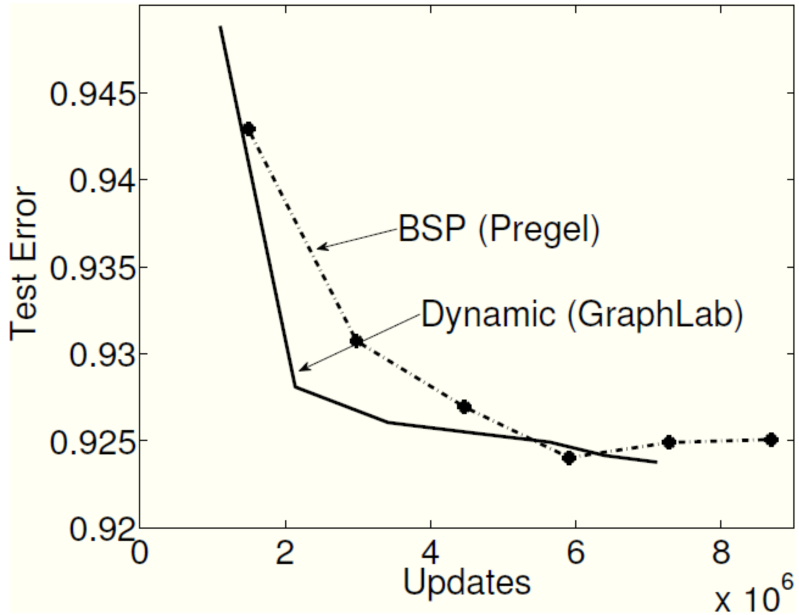


Experiments---Netflix Movie Recommendation

Exp.	#Verts	#Edges	Vertex Data	Edge Data	Update Complexity	Shape	Partition	Engine
Netflix	0.5M	99M	$8d + 13$	16	$O\left(d^3 + deg.\right)$	bipartite	random	Chromatic



Netflix comparisions



Dynamic Netflix

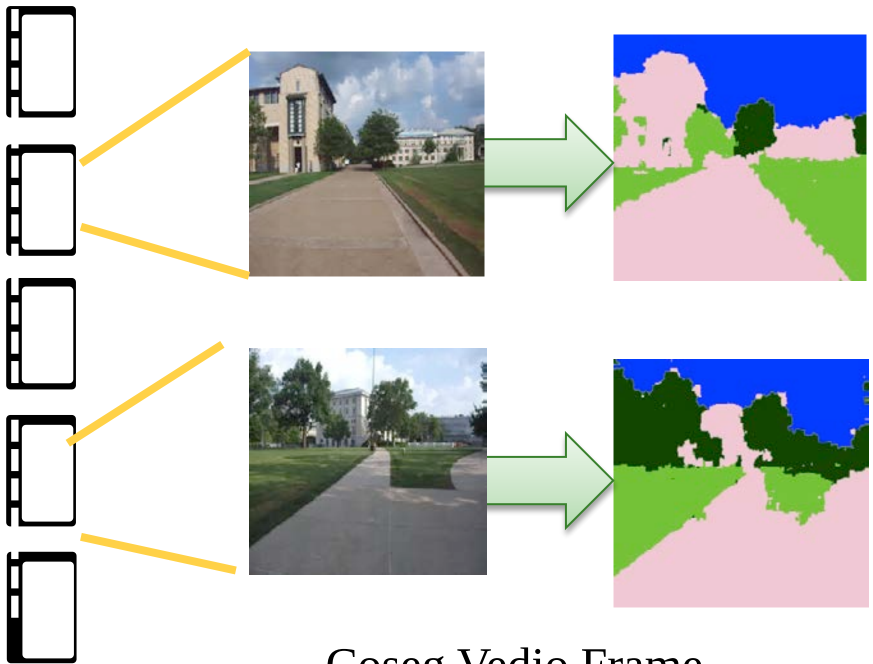
GraphLab outperforms Hadoop by 40~60 times and is comparable to MPI implementation

Dynamic computation can converge to equivalent test error in about half the number of updates

Experiments---Video Co-segmentation(CoSeg)

Task: Joint co-segmentation
Identify and cluster spatio -temporal segments with similar texture in video.
Algorithm:
Gaussian Misture Model (**GMM**)
Loopy Belief Propagation (**LBP**)

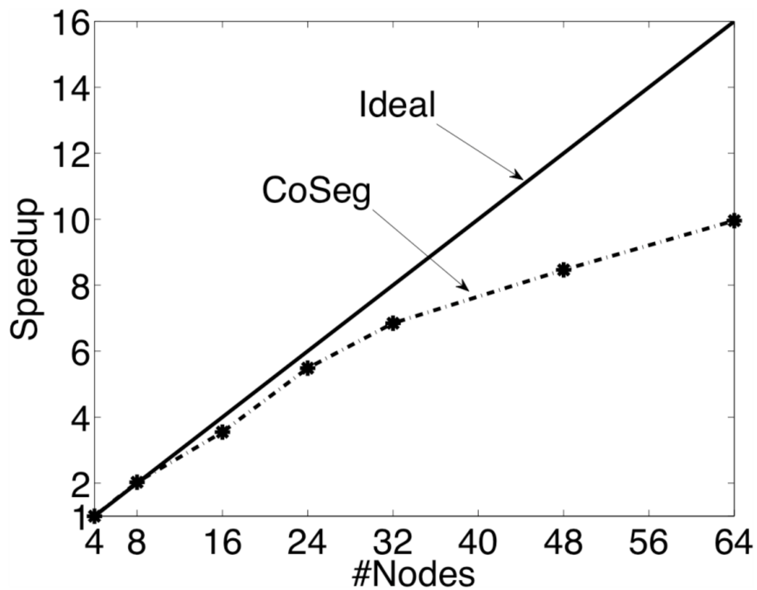
GraphLab Model:
Graph: a grid of 120*50 rectangular super-pixels
Edge: indicating the neighboring super-pixel
Vertex: super-pixel, stores the color and texture statistics for all the raw pixels in its domain
Update Function: alternating GMM and LBP to predict the best label for each super-pixel.
Schedule: adaptive update schedule



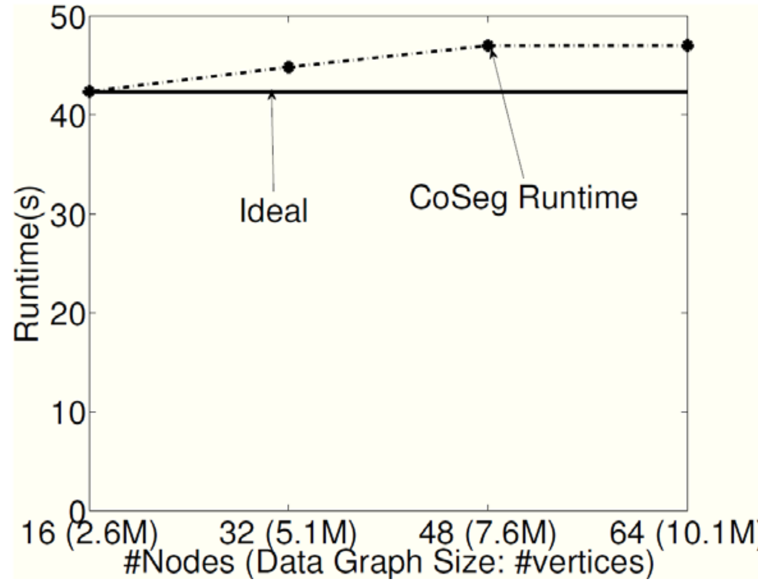
Coseg Vedio Frame

Experiments---Video Co-segmentation(CoSeg)

Exp.	#Verts	#Edges	Vertex Data	Edge Data	Update Complexity	Shape	Partition	Engine
CoSeg	10.5M	31M	392	80	$O\,(deg.)$	3D grid	frames	Locking



Coseg Scalability



Coseg Weak Scaling

GraphLab can achieve scalability and performance on large vertex graph

GraphLab provides nearly optimal weak scaling

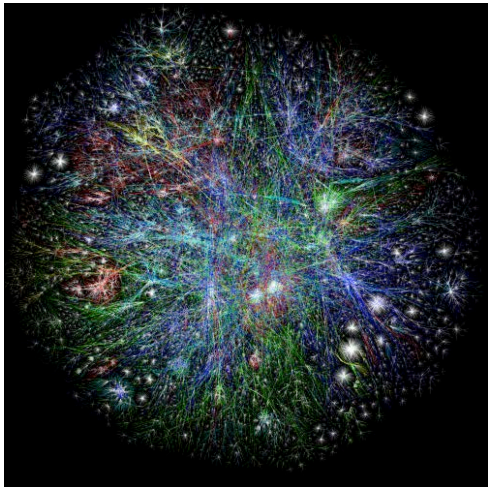
Summary of GraphLab

- An abstraction tailored to Machine Learning
  - Targets Graph-Parallel Algorithms
- Naturally expresses
  - Data/computational dependencies
  - Dynamic iterative computation
- Simplifies parallel algorithm design
- Automatically ensures data consistency
- Achieves state-of-the-art parallel performance on a variety of problems
- ***But, GraphLab is not sufficient to handle Natural Graphs!***

Outline

- **Properties** of Graph based machine learning
- ***Big*** : Data-parallel :Map Reduce
- ***Dependency***: Graph-parallel: Pregel
- ***Efficiency***: Asychronous Graph- Parallel: GraphLab
- ***PowerLaw Vertex***: GraphLab 2.1- PowerGraph

Natural Graph



The Internet

Human Brain

LinkedIn Social Network

Natural Graphs

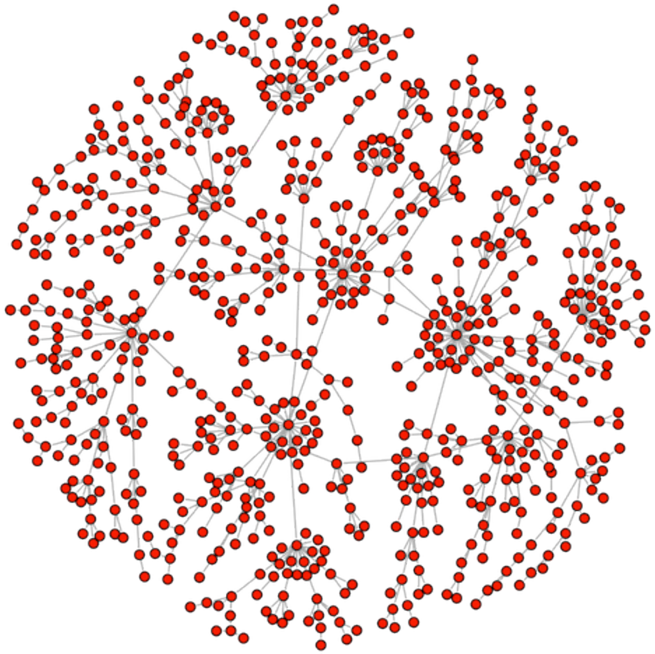
Graphs derived from natural phenomena

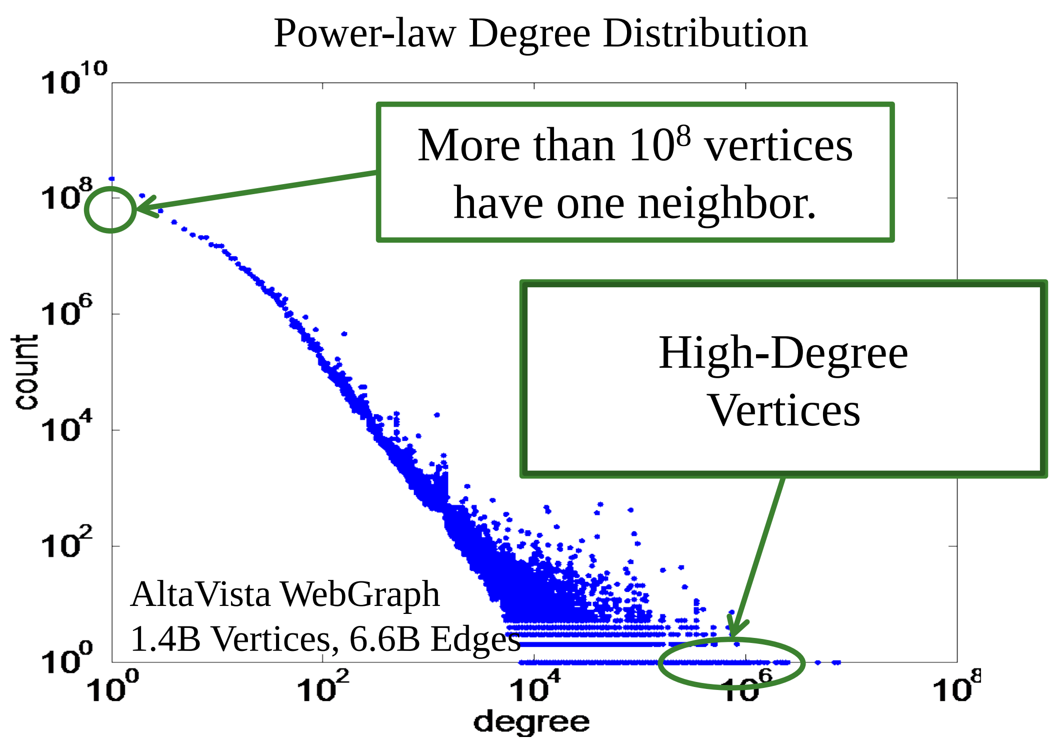
Problems:

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

Natural Graph Properties

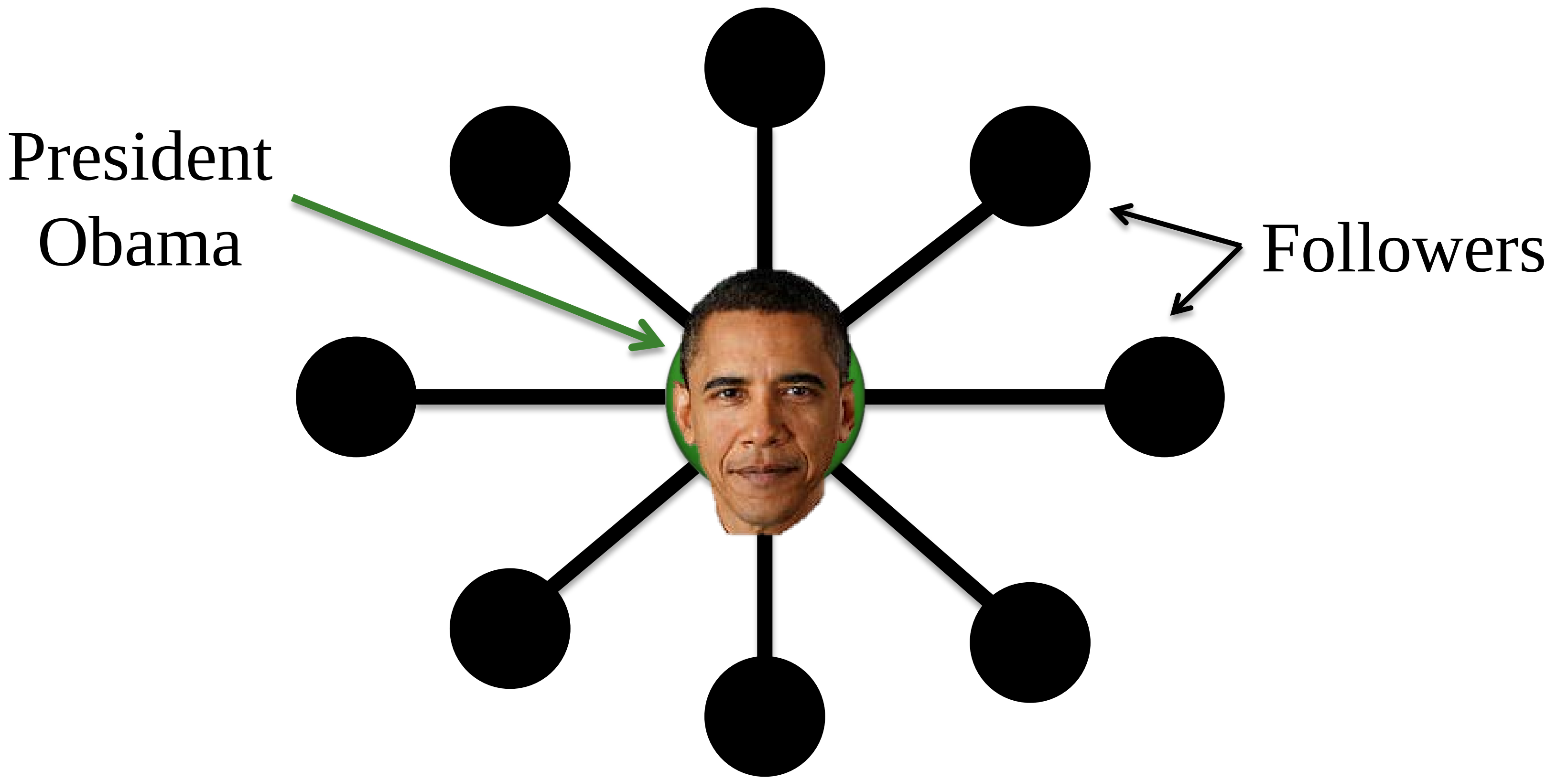
Power-Law Degree Distribution





Natural Graph Properties

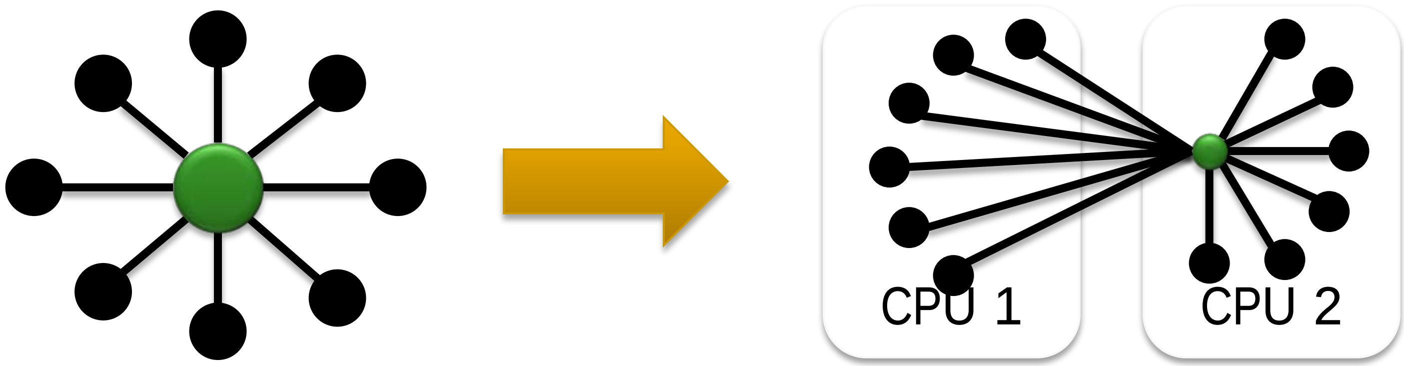
Power-Law Degree Distribution
"Star Like" Motif



https://www.usenix.org/conference/osdi12

Natural Graph Properties

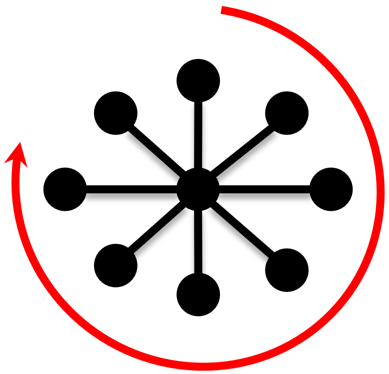
Power-Law Graphs are Difficult to Partition



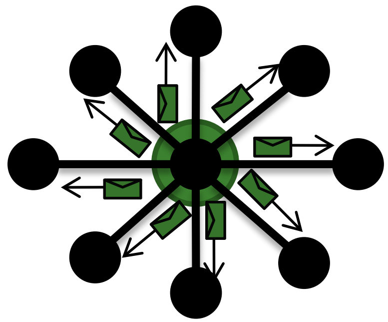
Power-Law graphs do not have **low-cost** balanced cuts *[Leskovec et al. 08, Lang 04]*

Traditional graph-partitioning algorithms perform **poorly** on Power-Law Graphs. *[Abou-Rjeili et al. 06]*

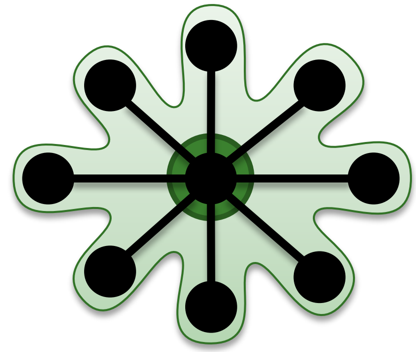
Pregel and GraphLab for High-Degree Vertices



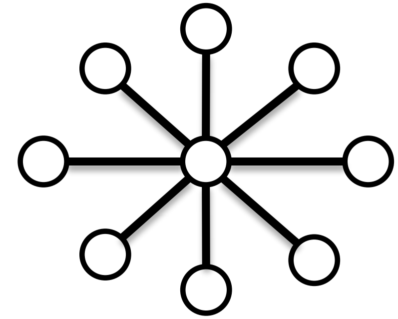
Sequentially process
edges



Sends many
messages
(Pregel)

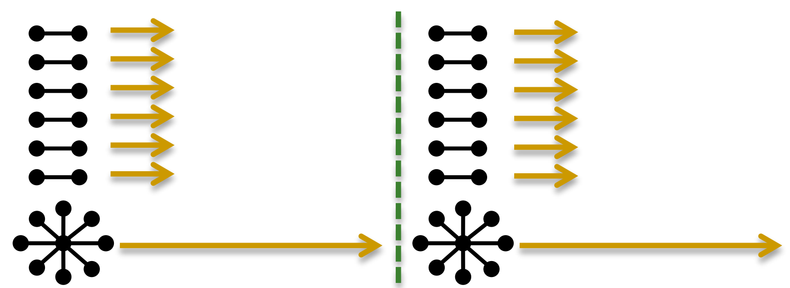


Touches a large
fraction of graph
(GraphLab)



Edge meta-data
too large for single
machine

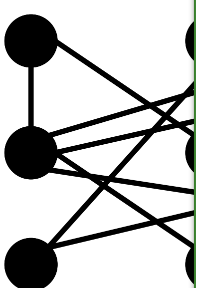
Asynchronous Execution
requires heavy locking (GraphLab)



Synchronous Execution
prone to stragglers (Pregel)

Graph & Pregel: Random Partitioning

Both GraphLab and Pregel resort to **random** (hashed) partitioning on **natural graphs**



10 Machines → 90% of edges cut

100 Machines → 99% of edges cut!

https://www.usenix.org/conference/osdi12

PowerGraph is Needed

GraphLab and **Pregel** are not well suited for **natural graphs**

Challenges of **high-degree vertices**

Low quality **partitioning**

GraphLab$_2$

PowerGraph

PowerGraph

GAS Decomposition: distribute vertex-programs

- Parallelize **high-degree** vertices

Vertex Partitioning:

- Efficiently distribute large **power-law** graphs.

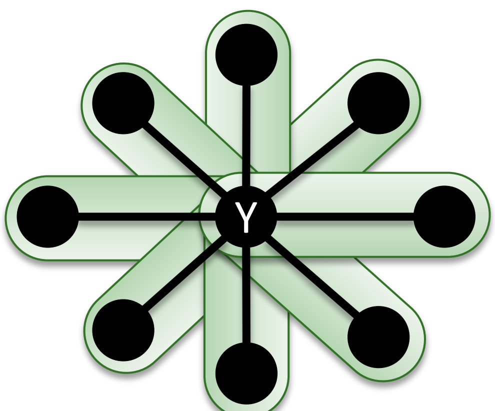
GAS Decomposition

Gather (Reduce)

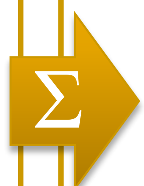
Accumulate information about neighborhood

User Defined:

- **Gather**( ) → Σ
- $\Sigma_1 \oplus \Sigma_2 \rightarrow \Sigma_3$



Parallel Sum

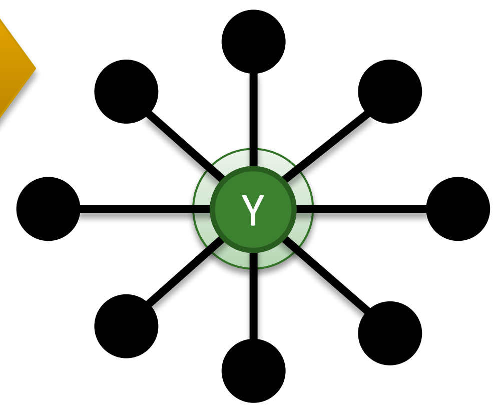


Apply

Apply the accumulated value to center vertex

User Defined:

- **Apply**(Y , Σ) → Y'

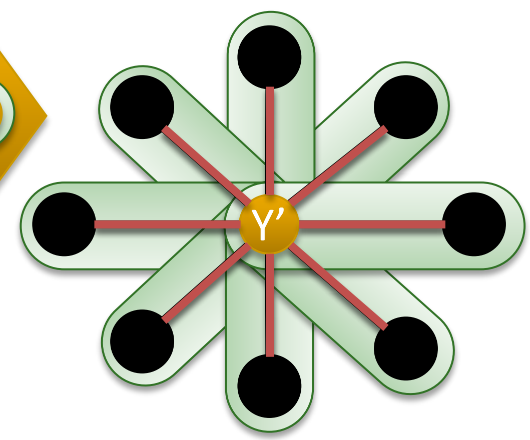


Scatter

Update adjacent edges and vertices.

User Defined:

- **Scatter**(Y') →



Update Edge Data & Activate Neighbors

https://www.usenix.org/conference/osdi12

GAS for PageRank

GraphLab_PageRank (i)

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

Gather: gather information about neighborhood

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

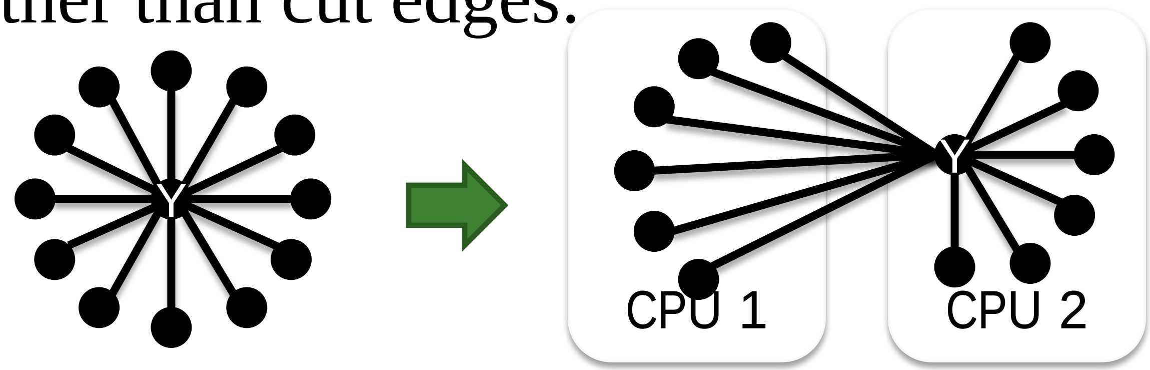
Apply: update vertex

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

Scatter: signal neighbors

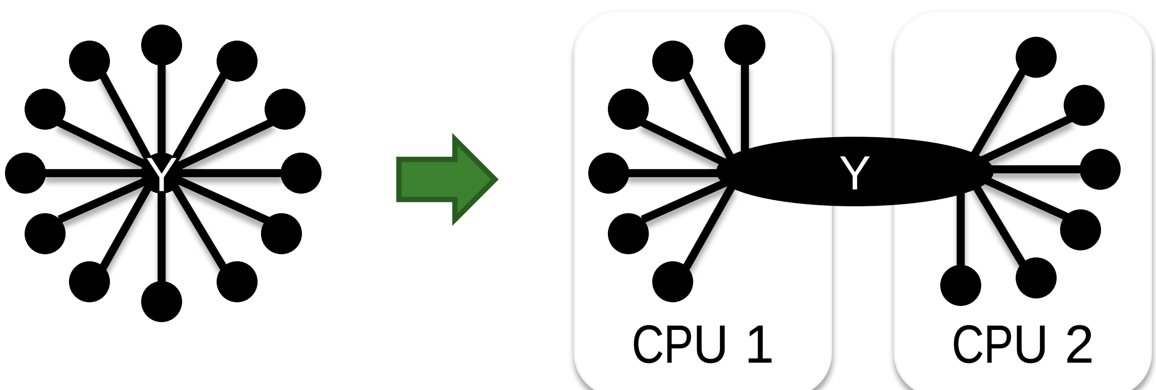
Graph partition

- Rather than cut edges:



Must synchronize **many** edges

- PowerGraph cut vertices:



Must synchronize a **single** vertex

*Percolation theory suggests that power law graphs have **good vertex cuts**. [Albert et al. 2000]*

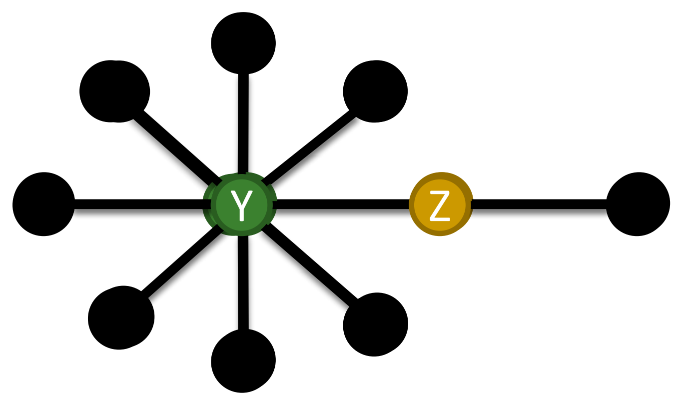
Constructing Vertex-Cut

Evenly assign **edges** to machines

Machine 1

Machine 2

Machine 3

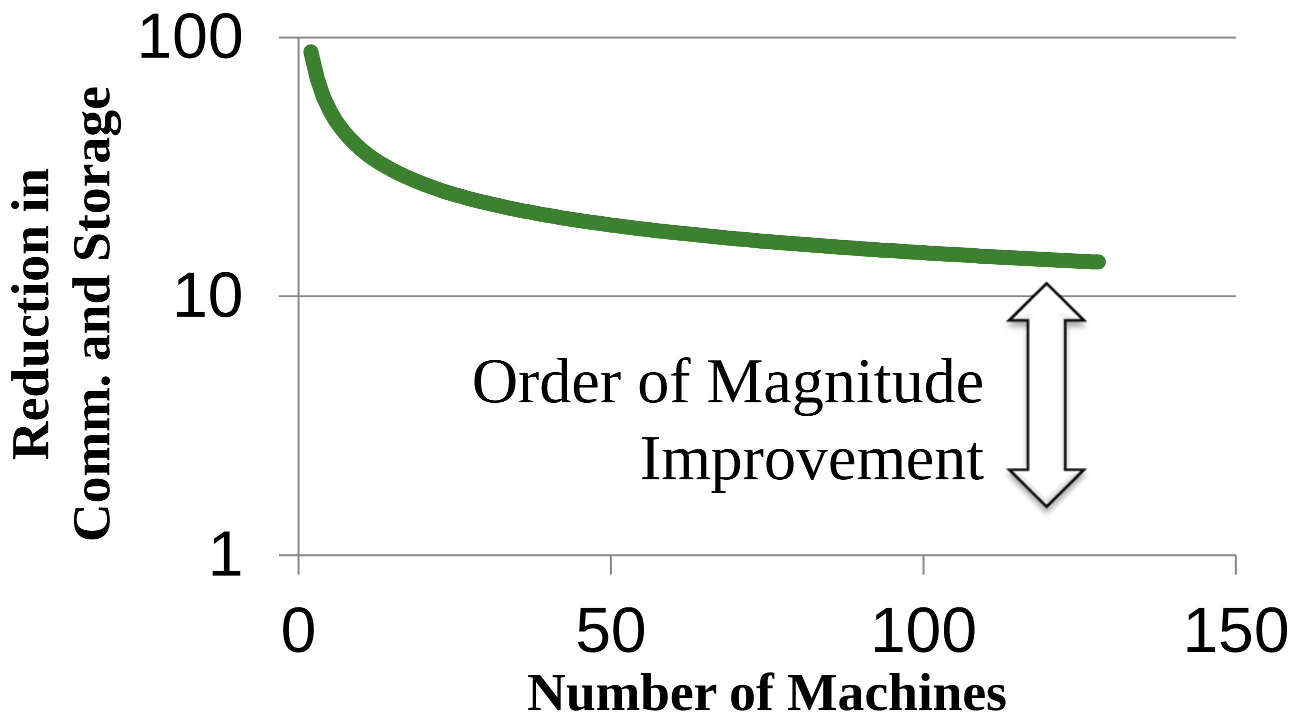


Vertex spans according to its **adjacent edges**

https://www.usenix.org/conference/osdi12

Vertex-Cut vs Edge-Cut

Expected improvement from vertex-cuts:

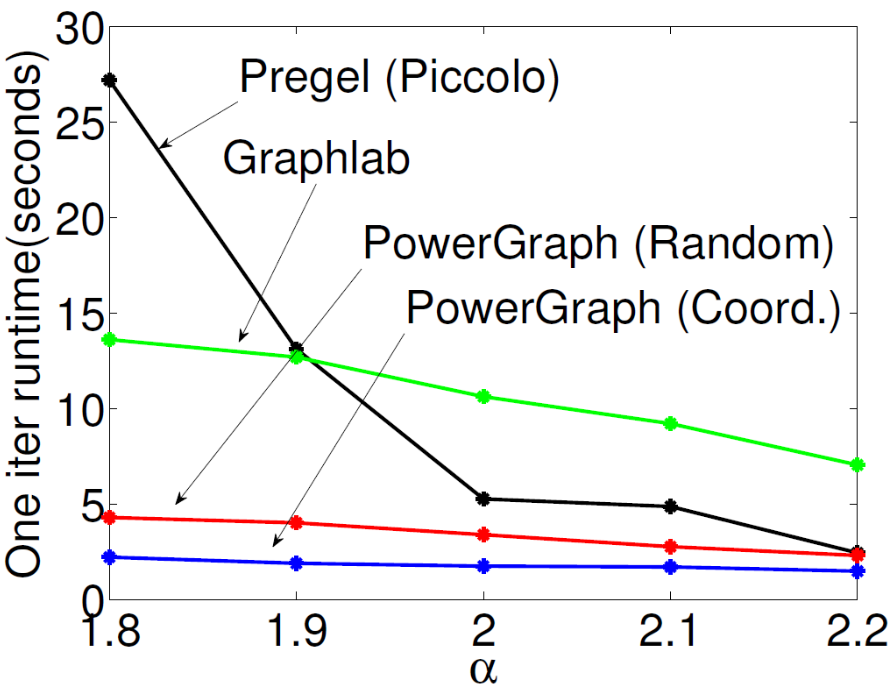


PowerGraph System Evaluation

PowerGraph vs GraphLab & Pregel

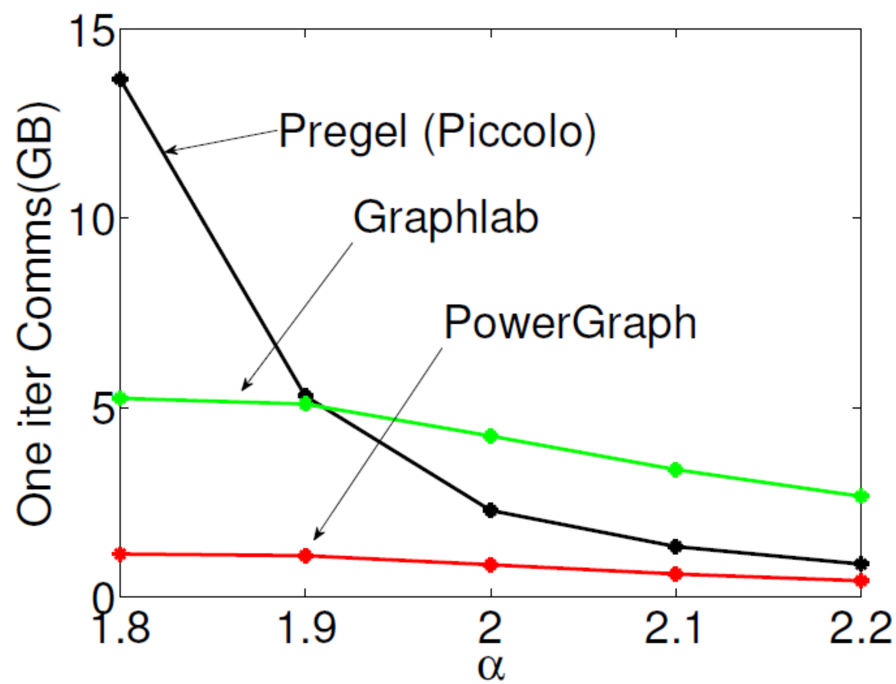
PageRank on Synthetic Power-Law Graphs:

α: Power-Law Constant, **higher α imply lower density (majority of vertices are low degree)**



←High-degree vertices

Runtime

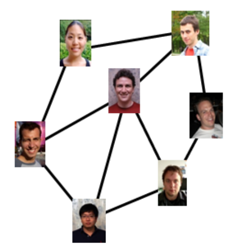
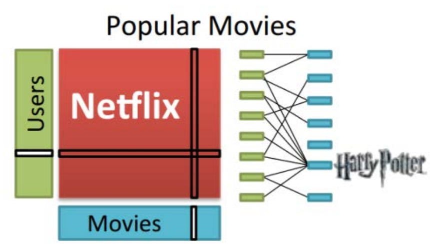
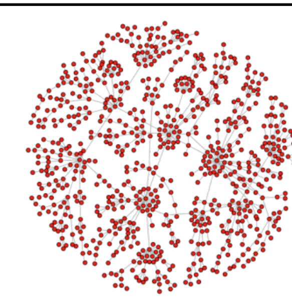


←High-degree vertices

Communication

PowerGraph is robust to **high-degree** vertices

Characteristic of Graphs

Properties:	**Examples**
Graphs are *ubiquitous* Social media, science , advertising,web…	Social media · Science · Advertising · Web
Graphs are *big* billions of vertices and edges and rich metadata SOLVED	28 Million Wikipedia Pages · 1 Billion Facebook Users · 6 Billion Flickr Photos · 72 Hours a Minute YouTube
***Dependency* is important** Graphs encode relationships between People, Facts , Products , Ideas, Interests SOLVED	 
Vertices in natural graphs are ***Power-law*** SOLVED	 

Summary of PowerGraph

- *Problem*: Computation on **Natural Graphs** is challenging
  - High-degree vertices
  - Low-quality edge-cuts
- *Solution*: **PowerGraph System**
  - **GAS Decomposition**: split vertex programs
  - **Vertex-partitioning**: distribute natural graphs
- PowerGraph **theoretically** and **experimentally** outperforms existing graph-parallel systems.

Future work

- Time evolving graphs
  - Support **structural changes** during computation
- Out-of-core storage (GraphChi)
  - Support graphs that don't fit in memory

Thank you!

Question?