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**: Asynchronous 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**: Asynchronous Graph-Parallel: GraphLab
  - **PowerLaw Vertex**: GraphLab 2.1- PowerGraph
## Characteristic of Graphs

<table>
<thead>
<tr>
<th>Properties:</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs are ubiquitous</strong></td>
<td>Social media, science, advertising, web…</td>
</tr>
<tr>
<td><strong>Graphs are big</strong></td>
<td>Billions of vertices and edges and rich metadata</td>
</tr>
<tr>
<td><strong>Dependency is important</strong></td>
<td>Graphs encode relationships between People, Facts, Products, Ideas, Interests</td>
</tr>
<tr>
<td>Vertices in natural graphs are <strong>Power-law</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Social media**
- 28 Million Wikipedia Pages
- 1 Billion Facebook Users
- 6 Billion Flickr Photos
- 72 Hours a Minute YouTube
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**: Asynchronous Graph-Parallel: GraphLab
- **PowerLaw Vertex**: GraphLab 2.1- PowerGraph
Data Parallel ML: MapReduce

<table>
<thead>
<tr>
<th>Data-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map Reduce</td>
</tr>
<tr>
<td>Feature</td>
</tr>
<tr>
<td>Extraction</td>
</tr>
<tr>
<td>Cross</td>
</tr>
<tr>
<td>Validation</td>
</tr>
<tr>
<td>Computing</td>
</tr>
<tr>
<td>Sufficient</td>
</tr>
<tr>
<td>Statistics</td>
</tr>
</tbody>
</table>
Data Parallel ML: MapReduce

Map phrase: Image Features Extraction

cloud.berkeley.edu/data/graphlab
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

Attractive Face Statistics

Ugly Face Statistics

Image Features

cloud.berkeley.edu/data/graphlab
## Characteristic of Graphs

<table>
<thead>
<tr>
<th>Properties:</th>
<th>Examples</th>
</tr>
</thead>
</table>
| **Graphs are ubiquitous**  
Social media, science, advertising, web… | ![Social media](facebook.png), ![Science](aeml.png), ![Advertising](billboard.png), ![Web](google.png) |
| **Graphs are big**  
billions of vertices and edges, rich metadata | ![Social media](facebook.png), ![Science](aeml.png), ![Advertising](billboard.png), ![Web](google.png)  
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 | ![Dependency](netflix.png)  
Popular Movies: Harry Potter, Popular Movies: Netflix |
| Vertices in natural graphs are **Power-law** | ![Power-law](power-law.png) |
# MapReduce: Pros & Cons

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simplicity of the model:</strong> Programmers specifies few simple methods that focuses on the functionality not on the parallelism</td>
<td><strong>Restricted programming constructs:</strong> only map &amp; reduce</td>
</tr>
<tr>
<td><strong>Scalability:</strong> Scales easily for large number of clusters with thousands of machines</td>
<td><strong>Does not scale well for dependent tasks:</strong> for example Graph problems</td>
</tr>
<tr>
<td><strong>Applicability:</strong> Applicable to many different systems and a wide variety of problems</td>
<td><strong>Does not scale well for iterative algorithms:</strong> iteration is very common in machine learning</td>
</tr>
</tbody>
</table>
### ML Tasks Beyond Data-Parallelism

<table>
<thead>
<tr>
<th>Data-Parallel</th>
<th>Graph-Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Map Reduce</strong></td>
<td><strong>Lasso</strong></td>
</tr>
<tr>
<td>Feature Extraction</td>
<td>Label Propagation</td>
</tr>
<tr>
<td>Cross Validation</td>
<td>Kernel Methods</td>
</tr>
<tr>
<td>Computing Sufficient Statistics</td>
<td>Belief Propagation</td>
</tr>
<tr>
<td>Tensor Factorization</td>
<td>PageRank</td>
</tr>
<tr>
<td>Deep Belief Networks</td>
<td>Neural Networks</td>
</tr>
</tbody>
</table>
Properties of Graph based machine learning

Big: Data-parallel :Map Reduce

Dependency: Graph-parallel: Pregel

Efficiency: Asynchronous 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
What's the rank of this user?

- Depends on rank of her followers
- Depends on the rank of their followers

PageRank
Page Rank Iteration

Iterate until convergence:
“My rank is weighted average of my friends’ rank”

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

- \( \alpha \) is the random reset probability
- \( w_{ji} \) is the probability 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

<table>
<thead>
<tr>
<th>Properties:</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs are ubiquitous</strong></td>
<td><img src="facebook.png" alt="Social media" /> <img src="amazon.png" alt="Science" /> <img src="bfi.png" alt="Advertising" /> <img src="google.png" alt="Web" /></td>
</tr>
<tr>
<td>Social media, science, advertising, web…</td>
<td></td>
</tr>
<tr>
<td><strong>Graphs are big</strong></td>
<td><img src="wikipedia.png" alt="Wikipedia" /> <img src="facebook.png" alt="Facebook" /> <img src="flickr.png" alt="Flickr" /> <img src="youtube.png" alt="YouTube" /></td>
</tr>
<tr>
<td>billions of vertices and edges, rich metadata</td>
<td></td>
</tr>
<tr>
<td><strong>Dependency is important</strong></td>
<td><img src="netflix.png" alt="Netflix" /> <img src="harry-potter.png" alt="Popular Movies" /></td>
</tr>
<tr>
<td>Graphs encode relationships between People, Facts, Products, Ideas, Interests</td>
<td></td>
</tr>
<tr>
<td><strong>Vertices in natural graphs are Power-law</strong></td>
<td><img src="complex-graph.png" alt="Complex graph" /> <img src="power-law.png" alt="Power-law distribution" /></td>
</tr>
</tbody>
</table>
Pregel: Bulk Synchronous Parallel (BSP) Model

Bulk synchronous computation can be highly inefficient

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**: Asynchronous Graph-Parallel: GraphLab
- **PowerLaw Vertex** : GraphLab 2.1- PowerGraph
GraphLab Framework

Graph Based Data Representation

Scheduler

Update Functions User Computation

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 ($U \rightarrow V$)
- Vertex data $D_{v}$ stores the rank of the webpage $R(V)$
- Edge data $D_{U \rightarrow V}$ stores the weight of the link ($U \rightarrow V$)
**Update Functions**

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

```plaintext
pagerank(i, scope) {
  // Get Neighborhood data
  (R[i], W_{ij}, R[j]) \leftarrow \text{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 \( T = \{v_1, v_2, \ldots\} \)

While \( T \) is not Empty do

1. \( v \leftarrow \text{RemoveNext}(T) \)
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

Full Consistency Model

Edge Consistency Model

Vertex Consistency Model

Read

Write
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 vertex by reading the $d$ length vectors on adjacent vertices and predict the edge value
Experiments---Netflix Movie Recommendation

<table>
<thead>
<tr>
<th>Exp.</th>
<th>#Verts</th>
<th>#Edges</th>
<th>Vertex Data</th>
<th>Edge Data</th>
<th>Update Complexity</th>
<th>Shape</th>
<th>Partition</th>
<th>Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netflix</td>
<td>0.5M</td>
<td>99M</td>
<td>8d + 13</td>
<td>16</td>
<td>$O(d^3 + \text{deg.})$</td>
<td>bipartite</td>
<td>random</td>
<td>Chromatic</td>
</tr>
</tbody>
</table>

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 Mixture 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
Experiments---Video Co-segmentation (CoSeg)

<table>
<thead>
<tr>
<th>Exp.</th>
<th>#Verts</th>
<th>#Edges</th>
<th>Vertex Data</th>
<th>Edge Data</th>
<th>Update Complexity</th>
<th>Shape</th>
<th>Partition</th>
<th>Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoSeg</td>
<td>10.5M</td>
<td>31M</td>
<td>392</td>
<td>80</td>
<td>$O (deg.)$</td>
<td>3D grid</td>
<td>frames</td>
<td>Locking</td>
</tr>
</tbody>
</table>

**Coseg Scalability**

GraphLab can achieve scalability and performance on large vertex graph

**Coseg Weak Scaling**

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!
Properties of Graph based machine learning

Big: Data-parallel: Map Reduce

Dependency: Graph-parallel: Pregel

Efficiency: Asynchronous Graph-Parallel: GraphLab

PowerLaw Vertex: GraphLab 2.1- PowerGraph
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

More than $10^8$ vertices have one neighbor.

High-Degree Vertices

Altavista WebGraph
1.4B Vertices, 6.6B Edges
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 $\rightarrow$ 90% of edges cut
100 Machines $\rightarrow$ 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
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:*
- \( \text{Gather}(Y) \rightarrow \Sigma \)
- \( \Sigma_1 + \Sigma_2 \rightarrow \Sigma_3 \)

**Apply**
Apply the accumulated value to center vertex

*User Defined:*
- \( \text{Apply}(Y, \Sigma) \rightarrow \) 

**Scatter**
Update adjacent edges and vertices.

*User Defined:*
- \( \text{Scatter}(Y') \rightarrow \)

Parallel Sum
\[ \sum \]

Update Edge Data & Activate Neighbors

https://www.usenix.org/conference/osdi12
// Compute sum over neighbors
total = 0
foreach (j in in_neighbors(i)):
    total = total + R[j] * w_{ji}

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

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

- Rather than cut edges:

  ![Diagram](image1)

  Must synchronize many edges

- PowerGraph cut vertices:

  ![Diagram](image2)

  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:

\( \alpha: \text{Power-Law Constant} \), higher \( \alpha \) imply lower density (majority of vertices are low degree)

PowerGraph is robust to high-degree vertices

PowerGraph is robust to **high-degree** vertices
# Characteristic of Graphs

<table>
<thead>
<tr>
<th>Properties:</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs are ubiquitous</strong></td>
<td>![Social media](facebook, twitter, web, science)</td>
</tr>
<tr>
<td>Social media, science, advertising, web…</td>
<td></td>
</tr>
<tr>
<td><strong>Graphs are big</strong></td>
<td>![Wikipedia Pages](28 Million), ![Facebook Users](1 Billion), ![Flickr Photos](6 Billion), ![YouTube](72 Hours a Minute)</td>
</tr>
<tr>
<td>billions of vertices and edges, rich metadata</td>
<td></td>
</tr>
<tr>
<td><strong>Dependency is important</strong></td>
<td>![Dependency Diagram](Netflix, Popular Movies, Movies)</td>
</tr>
<tr>
<td>Graphs encode relationships between People, Facts, Products, Ideas, Interests</td>
<td></td>
</tr>
<tr>
<td><strong>Vertices in natural graphs are Power-law</strong></td>
<td>![Power-law Distribution](Power-law Graph)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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?