Exploring the Hidden Dimension in Graph Processing

Mingxing Zhang, Yongwei Wu, Kang Chen,
*Xuehai Qian, Xue Li, and Weimin Zheng

Tsinghua University
*University of Southern California
Graph is Ubiquitous

Google: > 1 trillion indexed pages

Web Graph

31 billion RDF triples in 2011

Information Network

Google: > 1 trillion indexed pages

Facebook: > 800 million active users

Social Graph

De Bruijn: $4^k$ nodes ($k = 20, \ldots, 40$)

Biological Network

Acknowledgement: Arijit Khan, Sameh Elnikety [VLDB '14]
Graph Computing is Even More Ubiquitous

Traditional Graph Analysis

Many graph applications aim at analyzing the property of graphs

Examples

- Shortest Path
- Triangle Counting
- PageRank
- Connecting Component
Graph Computing is Even More Ubiquitous

Traditional Graph Analysis
- Many graph applications aim at analyzing the property of graphs

Examples
- Shortest Path
- Triangle Counting
- PageRank
- Connecting Component

MLDM as Graph Computing
- Many MLDM applications can be modeled and computed as graph problems

Examples
- Collaborative Filtering
- SpMM
- Neural Network
- Matrix Factorization
1D Partitioning: each vertex is assigned to one worker with all its incoming/outcoming edges attached.

Original Graph

Partition into 4 nodes with 1D partitioning

4 Sub-graphs & 6 Replicas (denoted by dashed circle)

Node 0

Node 1

Node 2

Node 3
1D → Each Vertex is a Task

Granularity of task dispatching is vertex, which is the SAME as the granularity of data dispatching.

Vertex Program
Each vertex program can read and modify the neighborhood of a vertex

Communication
- by messages (e.g. Pregel)
- by shared state (e.g. GraphLab)

Parallelism
by running multiple vertex programs simultaneously

Problem
SKEW!!!
2D Partitioning: each edge is assigned to one worker. Certain heuristics can be used for the assigning.

Original Graph

Partition into 4 nodes with 2D partitioning

4 Sub-graphs & 6 Replicas (denoted by dashed circle)
Each Edge is a Task

**Gather**

*User Defined:*
- Gather\((Y)\) → \(\Sigma\)
- \(\Sigma_1 \oplus \Sigma_2 \rightarrow \Sigma_3\)

**Apply**

*User Defined:*
- Apply\((Y, \Sigma)\) → \(Y'\)

**Scatter**

*User Defined:*
- Scaber\((Y)\)

Granularity of task dispatching is edge, **which is also the SAME as** the granularity of data dispatching.
Is this the END of task partitioning?

- YES
  - Vertex can be attached with multiple edges
  - But an edge is connected to only two vertices
  - Indivisible!
Is this the END of task partitioning?

- YES
  - Vertex can be attached with multiple edges
  - But an edge is connected to only two vertices
  - Indivisible! \(\leftrightarrow\) NOT true for many problems

- NO
  - Each vertex/edge can be further partitioned!
Example: Collaborative Filtering

Matrix View

$$R_{u,v} \approx \langle P[u], Q[v] \rangle$$

Graph View

In graph view:
Each vertex is attached with a feature vector with size $D$.
A typical pattern of modelling MLDM problem as graph problem.

Collaborative Filtering:
estimating missing ratings based on a given incomplete set of (user, item) ratings.
3D Partitioning: each vertex is split into sub-vertices and a 2D partitioner is used in each layer.
Motivation

• We found a **NEW** dimension, shouldn’t we partition it?

• Observation
  - These vector properties are usually operated by element-wise operators
  - Easy to parallelize
    - With no communication!
Two Kinds of Communications

Intra-layer Communication
- Less sub-graphs
- Less replicas
- Reduced!
Two Kinds of Communications

Intra-layer Communication
- Less sub-graphs
- Less replicas
- Reduced!

Inter-layer Communication
- Not exist before
- Increased!
Two Kinds of Communications

Intra-layer Communication
- Less sub-graphs
- Less replicas
- Reduced!

Inter-layer Communication
- Not exist before
- Increased!

NEW TRADE-OFF!
New System: CUBE

- Existing models does not formalize the inter-layer communication
- CUBE
  - Adopts 3D partitioning
  - Implements a new programming model UPPS
  - Use a matrix backend
Data Model of CUBE

• The data graph model is also used in CUBE, but each vertex/edge can be attached with two kinds of data.

• Two classes of data

  - DShare: an indivisible property
    - represented by a single variable.
  - DColle: a divisible collection of property vector
    - stored as a vector of variables
    - \( \text{len(DColle)} == \text{“collection size” Sc} \)
3D Partitioner (L, \(\mathcal{P}\)): where L is the number of layers, and \(\mathcal{P}\) is a 2D partitioner.
Programming Model

• Update:--- **inter-layer** communication
  - UpdateVertex: read/write *all the data* of a vertex
  - UpdateEdge: read/write *all the data* of an edge

• Push, Pull, Sink:--- **intra-layer** communication
  - Variations of GAS, for an edge *edg* that connects *(src, dst)*
  - Push: use *src* and *edg* to update *dst*.
  - Pull: use *dst* and *edg* to update *src*.
  - Sink: use *src* and *dst* to update *edg*. 
Matrix Backend

Vertex Frontend

- Easy to program
- Similar to existing works
- Less efficiency

Matrix Backend

- Hard to program
- High efficiency
- Simple indexing
- Hilbert order

Mapping from vertex frontend to matrix backend can significantly speed up the system; a useful method pioneered by GraphMat.
Evaluation

• **Baseline:** PowerGraph and PowerLyra
  - the default partitioner *oblivious* is used for PowerGraph
  - every possible partitioner is tested for PowerLyra and the *best* is used
  - $\mathcal{P}$ of CUBE is the same as the partitioner used by PowerLyra

• **Platform:** 8-node system; connected with 1Gb Ethernet

• **Dataset**

| Name    | $|U|$ | $|V|$ | $|E|$ | Best 2D Partitioner |
|---------|-----|-----|-----|-------------------|
| Libimseti | 135,359 | 168,791 | 17,359,346 | Hybrid-cut |
| Last.fm   | 359,349 | 211,067 | 17,559,530 | Bi-cut          |
| Netflix   | 17,770  | 480,189 | 100,480,507 | Bi-cut          |
Micro Benchmarks: SpMM

**SpMM**

\[ Q[v] = R[u,v]\cdot P[u] + R[w,v]\cdot P[w] \]

**Execution Time of SpMM**

- Libimseti, Sc = 256
- Lastfm, Sc = 256
- Libimseti, Sc = 1024

Communication cost is proportional to the number of replicas and inversely proportional to the number of layers and the number of subgraphs.

The graph shows the execution time of SpMM for different datasets and number of layers.
Micro Benchmarks: SumV & SumE

**SumV**: summing up each vertex’s vector

**SumE**: summing up each edge’s vector

Communication cost proportional to \( \frac{L-1}{L} \), where \( L \) is the number of layers.
### Compare to PowerLyra & PowerGraph

<table>
<thead>
<tr>
<th>Dataset</th>
<th>D</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PowerGraph</td>
</tr>
<tr>
<td>Libimseti</td>
<td>64</td>
<td>6.82</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>11.64</td>
</tr>
<tr>
<td>Lastfm</td>
<td>64</td>
<td>10.4</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>18.6</td>
</tr>
<tr>
<td>Netflix</td>
<td>64</td>
<td>18.3</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>30.6</td>
</tr>
</tbody>
</table>

- **Up to 4.7x speedup (V.S. PowerLyra)**
- **Up to 7.3x speedup (V.S. PowerGraph)**
About half of the speedup is from 3D partitioning (up to 2x) for Lastfm and Libimseti.

Not useful for Netflix.

Can achieve 2.5x speedup on Netflix if D is set to 2048.

Almost all the speedup is from network reduction.

Computation of ALS is dominated by DSYSV, an O(N^3) computation kernel.
Total memory consumption for running ALS on 64 workers with $D=32$, which means that $Sc=D^2+D=1056$.
Conclusion

• We found that, for certain graph problems, vertices and edges are not indivisible.

• We propose CUBE, which

  ➢ adopts 3D partitioning;
  ➢ uses matrix backend;
  ➢ achieves up to 4.7x speedup.
Thank You!!