Graphulo

Graph Analytics in GraphBLAS
Vijay Gadepally (vijayg@ll.mit.edu)
Dylan Hutchison, Jeremy Kepner
MIT Lincoln Laboratory
July 2015

This material is based upon work supported by the National Science Foundation under Grant No. DMS-1312831. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Outline

• Introduction
• Graphulo, Accumulo and the GraphBLAS
• Inner and Outer Products
• Graphulo Implementation and Performance
• Next Steps
Big Data Challenge

Humans (deciders)
- Kids
- Adults
- Elderly

Things (providers)
- Building Security
- Building Environment
- Building Usage
- Commuter Vehicles
- Work Vehicles
- Transport Vehicles
- Student Smartphones
- Classroom Tablets
- Fitness Wearables

Rapidly increasing
- Data volume
- Data velocity
- Data variety

Gap

10 Years Ago
5 Years Ago
Today
In 5 Years
Graph analytics are used in many missions to solve a variety of problems.
Outline

• Introduction
• Graphulo, Accumulo and the GraphBLAS
• Inner and Outer Products
• Graphulo Implementation and Performance
• Next Steps
Graphulo Overview

• Primary Goal
  – Open source Apache Accumulo Java library that enables many graph algorithms in Accumulo

• Core primitives: GraphBLAS

• 3 Graph Schemas
  – Adjacency, Incidence, Single-Table

• 4 Demonstration Graph Algorithms
  – Degree-filtered Breadth First Search, Jaccard coefficients, k-Truss subgraph, Non-negative Matrix Factorization

• Focus on Interactive Computing
  – “Cued” / Localized analytics within a neighborhood, as opposed to whole table analytics
  – Low latency more important than high throughput
  – Progress monitoring for user sanity (Is the library working or stuck?)
Apache Accumulo

- Highest performance open source database
- Contributed to Apache project by the USG in 2011
- Used extensively for government applications
- Requires a schema for storing and organizing data to obtain full benefits
Background on Accumulo

Best for:

- Large, de-normalized tables (NoSQL)
- In Hadoop HDFS / Java ecosystem
- Huge data volume – TBs to PBs
- Need cell-level visibility
- CP part of CAP
  - Consistency and Partition tolerance at the expense of Availability
  - Zookeeper, FATE
- Horizontal scaling

- Row store by default
  ➔ Scan over rows for O(log n) lookup & sorted order
- Iterator processing framework

<table>
<thead>
<tr>
<th>Key</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Row ID</td>
<td>Column</td>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td>Family</td>
<td>Qualifier</td>
<td>Visibility</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th></th>
<th></th>
</tr>
</thead>
</table>
Accumulo Graph Schema Variants

- **Adjacency Matrix (directed/undirected/weighted graphs)**
  - row = start vertex; column = vertex; value = edge weight

- **Incidence Matrix (multi-hyper-graphs)**
  - row = edge; column = vertices associated with edge; value = weight

- **D4M Schema**
  - Standard: main table, transpose table, column degree table, row degree table, raw data table
  - Multi-Family: use 1 table with multiple column families
  - Many-Table: use different tables for different classes of data

- **Single-Table**
  - use concatenated v1|v2 as a row key, and isolated v1 or v2 row key implies a degree

Graphulo should work with as many of Accumulo graph schemas as is possible
The GraphBLAS is an effort to define standard building blocks for graph algorithms in the language of linear algebra.

- More information about the group: [http://istc-bigdata.org/GraphBlas/](http://istc-bigdata.org/GraphBlas/)

Background material in book by J. Kepner and J. Gilbert: Graph Algorithms in the Language of Linear Algebra. SIAM, 2011

Draft GraphBLAS functions:

- SpGEMM, SpM{Sp}V, SpEWiseX, Reduce, SpRef, SpAsgn, Scale, Apply

Goal: show that these functions can perform the types of analytics that are often applied to data represented in graphs.

GraphBLAS is a natural starting point Graphulo mathematics
# Examples of Graph Problems

<table>
<thead>
<tr>
<th>Algorithm Class</th>
<th>Description</th>
<th>Algorithm Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exploration &amp; Traversal</td>
<td>Algorithms to traverse or search vertices</td>
<td>Depth First Search, Breadth First Search</td>
</tr>
<tr>
<td>Centrality &amp; Vertex Nomination</td>
<td>Finding important vertices or components within a graph</td>
<td>Betweenness Centrality, K-Truss sub graph detection</td>
</tr>
<tr>
<td>Similarity</td>
<td>Finding parts of a graph which are similar in terms of vertices or edges</td>
<td>Graph Isomorphism, Jaccard Index, Neighbor matching</td>
</tr>
<tr>
<td>Community Detection</td>
<td>Look for communities (areas of high connectedness or similarity) within a graph</td>
<td>Topic Modeling, Non-negative matrix factorization, Principle Component Analysis</td>
</tr>
<tr>
<td>Prediction</td>
<td>Predicting new or missing edges</td>
<td>Link Prediction</td>
</tr>
<tr>
<td>Shortest Path</td>
<td>Finding the shortest distance between two vertices</td>
<td>Floyd Warshall, Bellman Ford, A* algorithm, Johnson’s algorithm</td>
</tr>
</tbody>
</table>
# GraphBLAS initial function list

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Returns</th>
<th>Math Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpGEMM</td>
<td>- sparse matrices $A$ and $B$</td>
<td>sparse matrix</td>
<td>$C = \text{op}(A) \times \text{op}(B)$</td>
</tr>
<tr>
<td></td>
<td>- unary functors (op)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpM{Sp}V</td>
<td>- sparse matrix $A$</td>
<td>sparse/dense vector</td>
<td>$y = A \times x$</td>
</tr>
<tr>
<td>(Sp: sparse)</td>
<td>- sparse/dense vector $x$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpEWiseX</td>
<td>- sparse matrices or vectors</td>
<td>in place or sparse</td>
<td>$C = A \times B$</td>
</tr>
<tr>
<td></td>
<td>- binary functor and predicate</td>
<td>matrix/vector</td>
<td></td>
</tr>
<tr>
<td>Reduce</td>
<td>- sparse matrix $A$ and functors</td>
<td>dense vector</td>
<td>$y = \text{sum}(A, \text{op})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpRef</td>
<td>- sparse matrix $A$</td>
<td>sparse matrix</td>
<td>$B = A(p,q)$</td>
</tr>
<tr>
<td></td>
<td>- index vectors $p$ and $q$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpAsgn</td>
<td>- sparse matrices $A$ and $B$</td>
<td>none</td>
<td>$A(p,q) = B$</td>
</tr>
<tr>
<td></td>
<td>- index vectors $p$ and $q$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scale</td>
<td>- sparse matrix $A$</td>
<td>none</td>
<td>check manual</td>
</tr>
<tr>
<td></td>
<td>- dense matrix or vector $X$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apply</td>
<td>- any matrix or vector $X$</td>
<td>none</td>
<td>$\text{op}(X)$</td>
</tr>
<tr>
<td></td>
<td>- unary functor (op)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### GraphBLAS Initial Function List

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Returns</th>
<th>Math Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SpGEMM</strong></td>
<td>- sparse matrices ( A ) and ( B )</td>
<td>sparse matrix</td>
<td>( C = \text{op}(A) \times \text{op}(B) )</td>
</tr>
<tr>
<td></td>
<td>- unary functors (op)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SpM{Sp}V</strong></td>
<td>- sparse matrix ( A )</td>
<td>sparse/dense</td>
<td>( y = A \times x )</td>
</tr>
<tr>
<td>(Sp: sparse)</td>
<td>- sparse/dense vector ( x )</td>
<td>vector</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SpEWiseX</strong></td>
<td>- sparse matrices or vectors</td>
<td>in place or sparse matrix/vector</td>
<td>( C = A \times B )</td>
</tr>
<tr>
<td></td>
<td>- binary functor and predicate</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reduce</strong></td>
<td>- sparse matrix ( A ) and functors</td>
<td>dense vector</td>
<td>( y = \text{sum}(A, \text{op}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SpRef</strong></td>
<td>- sparse matrix ( A )</td>
<td>sparse matrix</td>
<td>( B = A(p,q) )</td>
</tr>
<tr>
<td></td>
<td>- index vectors ( p ) and ( q )</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SpAsgn</strong></td>
<td>- sparse matrices ( A ) and ( B )</td>
<td>none</td>
<td>( A(p,q) = B )</td>
</tr>
<tr>
<td></td>
<td>- index vectors ( p ) and ( q )</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Scale</strong></td>
<td>- sparse matrix ( A )</td>
<td>none</td>
<td>check manual</td>
</tr>
<tr>
<td></td>
<td>- dense matrix or vector ( X )</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>- any matrix or vector ( X )</td>
<td>none</td>
<td>( \text{op}(X) )</td>
</tr>
<tr>
<td></td>
<td>- unary functor (op)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Graphulo, Accumulo and the GraphBLAS
• Inner and Outer Products
• Graphulo Implementation and Performance
• Next Steps
### Matrix Multiply on Big Data

#### Traditional Matrix Multiply: $AB = C$

$$\begin{bmatrix} 6 & 5 & 0 & 2 \\ 0 & 4 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 3 \\ 5 & 0 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 6 & 23 \\ 0 & 12 \end{bmatrix}$$

- **Row & Column Labels**

### Database Table Multiply

<table>
<thead>
<tr>
<th>tod</th>
<th>word|coffee</th>
<th>word|desert</th>
</tr>
</thead>
<tbody>
<tr>
<td>0500</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>0800</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>0900</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1400</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tod</th>
<th>word|dew</th>
<th>word|hot</th>
</tr>
</thead>
<tbody>
<tr>
<td>0500</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0800</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0900</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>1400</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

$$\begin{bmatrix} \text{word}\|\text{dew} \\ \text{word}\|\text{hot} \end{bmatrix} = \begin{bmatrix} 6 & 23 \\ 0 & 12 \end{bmatrix}$$
### Matrix Multiply on Big Data

**Traditional Matrix Multiply:** \( AB = C \)

\[
\begin{bmatrix}
6 & 5 & 0 & 2 \\
0 & 4 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
0 & 0 \\
0 & 3 \\
5 & 0 \\
3 & 4
\end{bmatrix}
= \begin{bmatrix}
6 & 23 \\
0 & 12
\end{bmatrix}
\]

- **Row & Column Labels**
- **Sparse**

### Database Table Multiply

| tod|0500 | tod|0800 | tod|1400 |
|-----|------|------|------|
| word|coffee | 6 | 5 | 2 |
| word|desert | 4 |

| tod|0800 | tod|0900 | tod|1400 |
|-----|------|------|------|
| word|dew | 3 |
| word|hot | 5 |

| tod|1400 |
|-----|
| word|dew | 3 |
| word|hot | 4 |

\[
\begin{bmatrix}
6 & 4 \\
4 & 2
\end{bmatrix}
\begin{bmatrix}
6 & 23 \\
0 & 12
\end{bmatrix}
= \begin{bmatrix}
6 & 23 \\
0 & 12
\end{bmatrix}
\]
Matrix Multiply on Big Data

Traditional Matrix Multiply: $AB = C$

$$\begin{bmatrix} 6 & 5 & 0 & 2 \\ 0 & 4 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 3 \\ 5 & 0 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 6 & 23 \\ 0 & 12 \end{bmatrix}$$

- Row & Column Labels
- Sparse

$\longrightarrow$ Associative Array Mathematics$^1$

Database Table Multiply

$$\begin{array}{ccc}
\text{word|coffee} & \text{word|dew} & \text{word|hot} \\
\text{tod|0500} & 6 & 2 \\
\text{tod|0800} & 5 & 3 \\
\text{tod|1400} & 2 & \text{tod|0800} \\
\end{array}$$

$$\begin{array}{ccc}
\text{word|dew} & \text{word|hot} \\
\text{word|dew} & \text{word|hot} \\
\end{array}$$

$$\begin{bmatrix} 6 & 23 \\ 0 & 12 \end{bmatrix}$$

Multiple-source breadth-first search

- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Highly-parallel implementation for Betweenness Centrality*

*: A measure of influence in graphs, based on shortest paths
Multiple-source breadth-first search

- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Highly-parallel implementation for Betweenness Centrality*

*: A measure of influence in graphs, based on shortest paths
Inner Product

\[
\begin{align*}
\text{for } i = 1: N &= 2 \\
&\quad \text{for } j = 1: L = 2 \\
&\quad \quad \text{for } k = 1: M = 4 \\
&\quad \quad \quad \text{emit } A(i, k) \otimes B(k, j) \\
C(i, j) &= \bigoplus_{k=1}^{M} A(i, k) \otimes B(k, j)
\end{align*}
\]
Inner Product

\[
\begin{align*}
\text{word} | \text{coffee} & \quad \text{word} | \text{desert} \\
\text{tod} | 0500 & \quad \text{tod} | 0800 & \quad \text{tod} | 1400 \\
6 & \quad 5 & \quad 2 \\
\text{tod} | 0800 & \quad \text{tod} | 0900 & \quad \text{tod} | 1400 \\
5 & \quad 3 & \quad 4 \\
\text{word} | \text{dew} & \quad \text{word} | \text{hot} \\
\end{align*}
\]

\[
C(i, j) = \bigoplus_{k=1}^{M} A(i, k) \otimes B(k, j)
\]

1st Scan

\[
\begin{align*}
\text{for } i = 1: N = 2 \\
\quad \text{for } j = 1: L = 2 \\
\quad \quad \text{for } k = 1: M = 4 \\
\quad \quad \quad \text{emit } A(i, k) \otimes B(k, j)
\end{align*}
\]
Inner Product

\[
\begin{bmatrix}
6 & 5 & 2 \\
4 & & \\
\end{bmatrix}
\begin{bmatrix}
tod|0500 & tod|0800 & tod|1400 \\
tod|0800 & tod|0900 & tod|1400 \\
\end{bmatrix}
= 
\begin{bmatrix}
3 \\
5 & 4 \\
\end{bmatrix}
\begin{bmatrix}
word|dew & word|hot \\
word|dew & word|dew \\
\end{bmatrix}
\]

\[
C(i, j) = \bigoplus_{k=1}^{M} A(i, k) \otimes B(k, j)
\]

\[
\text{for } i = 1:\ N = 2 \\
\text{for } j = 1:\ L = 2 \\
\text{for } k = 1:\ M = 4 \\
\text{emit } A(i, k) \otimes B(k, j)
\]
Inner Product

\[
\begin{bmatrix}
6 & 5 & 2 \\
4 & & \\
\end{bmatrix}
\begin{bmatrix}
\text{word}\mid\text{dew} \\
\text{word}\mid\text{hot} \\
\end{bmatrix}
= \\
\begin{bmatrix}
6 & 23 \\
2 & 12 \\
\end{bmatrix}
\]

\[
\text{for } i = 1 : N = 2 \\
\text{for } j = 1 : L = 2 \\
\text{for } k = 1 : M = 4 \\
\text{emit } A(i,k) \otimes B(k,j)
\]

\[
C(i,j) = \bigoplus_{k=1}^{M} A(i,k) \otimes B(k,j)
\]
Inner Product

+ Write locality (sorted)
+ Pre-sum partial products
– N scans over table B

\[
\begin{bmatrix}
\text{word|coffee} & 6 & 5 & 2 \\
\text{word|desert} & 4 & & \\
\end{bmatrix}
\begin{bmatrix}
\text{tod|0500} & \text{tod|0800} & \text{tod|1400} \\
\end{bmatrix}
\begin{bmatrix}
\text{word|dew} & 3 \\
\text{word|hot} & 5 & 3 \\
\end{bmatrix}
\begin{bmatrix}
\text{tod|0800} & \text{tod|0900} & \text{tod|1400} \\
\end{bmatrix}
\begin{bmatrix}
\text{word|dew} & 6 & 23 \\
\text{word|hot} & 12 \\
\end{bmatrix}
\]

\[
C(i, j) = \sum_{k=1}^{M} A(i, k) \otimes B(k, j)
\]

\[
\begin{array}{c}
\text{for } i = 1 : N = 2 \\
\text{for } j = 1 : L = 2 \\
\text{for } k = 1 : M = 4 \\
\text{emit } A(i, k) \otimes B(k, j)
\end{array}
\]
Outer Product

Now explicitly showing $A^T$

\[
\begin{bmatrix}
\text{tod|0500} & 6 \\
\text{tod|0800} & 5 \\
\text{tod|1400} & 2 \\
\end{bmatrix} \quad \begin{bmatrix}
\text{word|dew} & 3 \\
\text{word|hot} & 4 \\
\end{bmatrix}
= \begin{bmatrix}
\text{word|coffee} \\
\text{word|desert} \\
\end{bmatrix}
\]

for $k = 1: M = 4$

for $i = 1: N = 2$

for $j = 1: L = 2$

emit $A(i, k) \otimes B(k, j)$

\[
C = \bigoplus_{k=1}^{M} A(:, k) \otimes B(k, :)
\]
Outer Product

1. Align Rows

\[
\begin{array}{c|ccc}
\text{word} & \text{coffee} & \text{desert} \\
\hline
\text{tod|0500} & 6 & 4 \\
\text{tod|0800} & 5 & 4 \\
\text{tod|1400} & 2 & 4 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
\text{word} & \text{dew} & \text{hot} \\
\hline
\text{tod|0800} & 3 & 4 \\
\text{tod|0900} & 3 & 4 \\
\text{tod|1400} & 3 & 4 \\
\end{array}
\]

\[
\begin{align*}
\text{for} & \ k = 1: M = 4 \\
\text{for} & \ i = 1: N = 2 \\
\text{for} & \ j = 1: L = 2 \\
\text{emit} & \ A(i, k) \otimes B(k, j)
\end{align*}
\]

\[
C = \bigoplus_{k=1}^{M} A(:, k) \otimes B(k, :)
\]
Outer Product

1. Align Rows

\[
\begin{pmatrix}
\text{tod|0500} \\
\text{tod|0800} \\
\text{tod|1400}
\end{pmatrix}
\begin{bmatrix}
6 \\
5 \\
2
\end{bmatrix}
\begin{pmatrix}
\text{word|coffee} \\
\text{word|desert}
\end{pmatrix}
\begin{pmatrix}
\text{tod|0800} \\
\text{tod|0900} \\
\text{tod|1400}
\end{pmatrix}
\begin{bmatrix}
3 \\
5 \\
3
\end{bmatrix}
\begin{pmatrix}
\text{word|dew} \\
\text{word|hot}
\end{pmatrix}
= \begin{pmatrix}
\text{word|coffee} \\
\text{word|dew} \\
\text{word|dew}
\end{pmatrix}
\begin{bmatrix}
3 \\
5 \\
3
\end{bmatrix}
\begin{pmatrix}
\text{word|hot} \\
\text{word|hot}
\end{pmatrix}
\]

\[C = \bigoplus_{k=1}^{M} A(:, k) \otimes B(k, :)\]
Outer Product

2. Cartesian Product

\[
\begin{bmatrix}
6 & 4 \\
5 & 4 \\
2 & 2 \\
\end{bmatrix}
\begin{bmatrix}
3 \\
5 \\
3 \\
\end{bmatrix}
= \begin{bmatrix}
15 \\
\text{word|coffee} \\
\text{word|dew} \\
\text{word|hot} \\
\text{word|desert} \\
\end{bmatrix}
\]

\[C = \bigoplus_{k=1}^{M} A(:,k) \otimes B(k,:)
\]

for \(k = 1: M = 4\)

for \(i = 1: N = 2\)

for \(j = 1: L = 2\)

emit \(A(i,k) \otimes B(k,j)\)
Outer Product

2. Cartesian Product

\[
\begin{align*}
\text{word|coffee} & \quad \text{word|desert} \\
\text{tod|0500} & \quad 6 \\
\text{tod|0800} & \quad 5 \\
\text{tod|1400} & \quad 2 \\
\end{align*}
\]

\[
\begin{align*}
\text{word|dew} & \quad \text{word|hot} \\
\text{tod|0800} & \quad 3 \\
\text{tod|0900} & \quad 5 \\
\text{tod|1400} & \quad 3 \\
\end{align*}
\]

\[
\begin{align*}
\text{word|coffee} & \quad \text{word|desert} \\
\text{tod|0800} & \quad 15 \\
\text{tod|0900} & \quad 12 \\
\end{align*}
\]

for \( k = 1 : M = 4 \)

\[
\begin{align*}
\text{for } i = 1 : N = 2 \\
\text{for } j = 1 : L = 2 \\
\text{emit } A(i, k) \otimes B(k, j)
\end{align*}
\]

\[
C = \bigoplus_{k=1}^{M} A(:, k) \otimes B(k, :)
\]
Outer Product

1. Align Rows

\[
\begin{bmatrix}
\text{word}\mid\text{coffee} \\
\text{word}\mid\text{desert}
\end{bmatrix}
= \begin{bmatrix}
6 \\
5 \\
2
\end{bmatrix}
\begin{bmatrix}
\text{word}\mid\text{dew} \\
\text{word}\mid\text{hot}
\end{bmatrix}
= \begin{bmatrix}
\text{word}\mid\text{coffee} \\
\text{word}\mid\text{desert}
\end{bmatrix}
\begin{bmatrix}
15 \\
12
\end{bmatrix}
\]

\[
C = \bigoplus_{k=1}^{M} A(:, k) \otimes B(k, :)
\]

for \( k = 1 : M = 4 \)

for \( i = 1 : N = 2 \)

for \( j = 1 : L = 2 \)

\emph{emit} \( A(i, k) \otimes B(k, j) \)
**Outer Product**

1. **Align Rows**

\[
\begin{bmatrix}
\text{ tod|0500 } & 6 \\
\text{ tod|0800 } & 5 \\
\text{ tod|1400 } & 2
\end{bmatrix}
\begin{bmatrix}
\text{ word|coffee} \\
\text{ word|desert}
\end{bmatrix}
\vdots
\begin{bmatrix}
\text{ tod|0800 } & 3 \\
\text{ tod|0900 } & 5 \\
\text{ tod|1400 } & 3
\end{bmatrix}
\begin{bmatrix}
\text{ word|dew} \\
\text{ word|hot}
\end{bmatrix}
= \begin{bmatrix}
\text{ word|coffee} \\
\text{ word|desert}
\end{bmatrix}
\begin{bmatrix}
15 \\
12
\end{bmatrix}
\]

\[
C = \bigoplus_{k=1}^{M} A(:, k) \otimes B(k, :) 
\]
Outer Product

2. Cartesian Product

\[
\text{for } k = 1: M = 4 \\
\text{for } i = 1: N = 2 \\
\text{for } j = 1: L = 2 \\
\text{emit } A(i, k) \otimes B(k, j)
\]

\[
C = \bigoplus_{k=1}^{M} A(:, k) \otimes B(k, :)
\]
Outer Product

2. Cartesian Product

Lazy $\oplus$:
Accumulo stores both 15 and 8 until next scan or compaction

\[
C = \bigoplus_{k=1}^{M} A(:,k) \otimes B(k,:)\]
Outline

- Introduction
- Graphulo, Accumulo and the GraphBLAS
- Inner and Outer Products
- Graphulo Implementation and Performance
- Next Steps
Table Multiply Before Graphulo

Client

Accumulo

A
B
Table Multiply Before Graphulo
Multiply in-memory*

*Blocked algorithms exist for large tables at reduced efficiency
Table Multiply Before Graphulo
Table Multiply Before Graphulo

Multiply in-memory*

*Blocked algorithms exist for large tables at reduced efficiency
Outer Product in Graphulo Iterators

Assumes Tables A and B are in the database

Custom Iterator performs Outer Product

Partial Products to Table C

Lazy Addition to Sum Partial Products

Entries sent to client if requested
Inner vs. Outer Product

- Outer product best for Accumulo
  - Single pass over table B = single disk read
  - BatchWriter ingest handles unsorted writes
  - Combiners handle $\oplus$
  - Less extra partial products written for sparse data

- Inner product still has merit
  - Better for dense data
  - Hybrid 2D-like algorithm possible
Performance Experiment

• Compare to pre-Graphulo alternative:
  – D4M Matlab client as Middleman

• Scaled / Weak scaling study:
  – How multiply rate varies with increasing problem size at fixed resources
  – Ideal: constant multiply rate

• Fixed / Strong scaling study:
  – How multiply rate varies with increasing resources at fixed problem size
  – Ideal: multiply rate scales linearly with increasing resources

• Environment:
  – Laptop, 16GB RAM, 2 Dual-core i7 processors, Accumulo 1.6.1

• Vary problem size between SCALE 10 and 18
  – Power law graph generator
  – # of nodes in each input table is $2^{\text{SCALE}}$. Used 16 edges/node

• Vary resources with # Accumulo Tablets (Varies # Threads)
Performance Experiment
Outline

• Introduction
• Graphulo, Accumulo and the GraphBLAS
• Inner and Outer Products
• Graphulo Implementation and Performance
• Next Steps
Next Steps

- Continue to implement GraphBLAS functions in Accumulo
- Incorporate greater set of examples in Graphulo release
  - Currently, Degree Filtered BFS, K-Truss, Jaccard, and NMF
- Extended scalability and performance testing

We will have a Graphulo release by Fall 2015!