Server-side Sparse Matrix Multiply in the Accumulo Database

Dylan Hutchison\(^1\)\(^2\)*, Vijay Gadepally\(^1\)*, Jeremy Kepner\(^1\)*, Adam Fuchs\(^3\)

\(^1\)MIT Lincoln Laboratory  \(^2\)University of Washington  \(^3\)Sqrrl Inc.

2015 August

*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.
This work is NOT
Creating the best system
for a particular task (matrix multiply)
This work is NOT
Creating the best system for a particular task (matrix multiply)

This work IS
Adding graph analytic capabilities (matrix multiply) to an all-around good system used in practice today (Accumulo)
Outline

- Intro to Graphulo
- Intro to Matrix Multiply
- Intro to Accumulo
- Matrix Multiply pre-Graphulo
- Inner Product
- Outer Product
- Accumulo Implementation
- Performance
- Conclusions
Many groups store graph data in Accumulo

Need tools for graph analysis in Accumulo
Why Accumulo?

Accumulo ingest performance is 100x greater than competing technologies

- MIT LL 2012: 4M/s
- MIT LL 2014: 115M/s
- BAH 2013: 108M/s
- Google 2014: 1M/s
- Oracle 2013: 140K/s

Accumulo ingest performance is 100x greater than competing technologies
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
  – "Queued" / 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?
# GraphBLAS initial function list

| Function   | Parameters                                      | Returns          | Math Notation          |
|------------|-------------------------------------------------|------------------|------------------------|
| SpGEMM     | - sparse matrices A and B <br> - unary functors (op) | sparse matrix    | $C = \text{op}(A) \times \text{op}(B)$ |
| SpM{Sp}V   | - sparse matrix A <br> - sparse/dense vector x   | sparse/dense vector | $y = A \times x$       |
| (Sp: sparse)|                                                |                  |                        |
| SpEWiseX   | - sparse matrices or vectors <br> - binary functor and predicate | in place or sparse matrix/vector | $C = A \times B$       |
| Reduce     | - sparse matrix A and functors                  | dense vector     | $y = \text{sum}(A, \text{op})$ |
| SpRef      | - sparse matrix A <br> - index vectors p and q   | sparse matrix    | $B = A(p,q)$           |
| SpAsgn     | - sparse matrices A and B <br> - index vectors p and q | none             | $A(p,q) = B$           |
| Scale      | - sparse matrix A <br> - dense matrix or vector X | none             | check manual           |
| Apply      | - any matrix or vector X <br> - unary functor (op) | none             | $\text{op}(X)$         |
• Intro to Graphulo
• Intro to Matrix Multiply
• Intro to Accumulo
• Matrix Multiply pre-Graphulo
• Inner Product
• Outer Product
• Accumulo Implementation
• Performance
• Conclusions
Matrix Multiply on Big Data

Traditional Matrix Multiply: $AB = C$

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

Database Table Multiply

| tod | word | dew | hot |
|-----|------|-----|-----|
| 0500 | coffee | 6 | 2 |
| 0800 | | 0 | 3 |
| 0900 | | 5 | 0 |
| 1400 | | 3 | 4 |

| tod | word | dew | hot |
|-----|------|-----|-----|
| 0500 | coffee | 6 | 23 |
| 0800 | desert | 0 | 12 |
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

\[
\begin{bmatrix}
\text{word|coffee} & 6 & 5 & 2 \\
\text{word|desert} & 4 & 2 & 1
\end{bmatrix}
\begin{bmatrix}
tod|0500 & \text{word|dew} & 3 \\
tod|0800 & \text{word|hot} & 4 \\
tod|0900 & \text{word|dew} & 3 \\
tod|1400 & \text{word|hot} & 4
\end{bmatrix} =
\begin{bmatrix}
\text{word|coffee} & 6 & 23 \\
\text{word|desert} & 12
\end{bmatrix}
\]
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

⇒ Associative Array Mathematics

Database Table Multiply

| word| coffee | word| desert |
|-----|--------|-----|--------|
| tod|0500    | tod|0800    | tod|1400    |
| word|coffee | word|dew    | word|hot    |
| 6   | 5      | 2   | 3      |
| 4   | 4      | 3   | 4      |

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

1. J. Kepner and V. Gadepally. "Adjacency matrices, incidence matrices, database schemas, and associative arrays" in International Parallel & Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2014
Application: Multi-Source Breadth-First Search

- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Basis for a wide range of graph algorithms
Application: Multi-Source Breadth-First Search

- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Basis for a wide range of graph algorithms
Outline

- Intro to Graphulo
- Intro to Matrix Multiply
- Intro to Accumulo
- Matrix Multiply pre-Graphulo
- Inner Product
- Outer Product
- Accumulo Implementation
- Performance
- Conclusions
Background on Accumulo

| Key | Value |
|-----|-------|
| Row ID | Column |
| | Family | Qualifier | Visibility |
| | Timestamp |

Use Transpose Tables see D4M Schema

Best for:

- Large, de-normalized tables (NoSQL)
- Hadoop HDFS / Java ecosystem
- Huge data volume – TBs to PBs
- Cell-level visibility
- Robust horizontal scaling

- Row store by default
  - Scan over rows for $O(\log n)$ lookup & sorted order
  - Log-structured Merge Tree design
- Iterator processing framework
Background on Accumulo

Best for:
- Large, de-normalized tables (NoSQL)
- Hadoop HDFS / Java ecosystem
- Huge data volume – TBs to PBs
- Cell-level visibility
- Robust horizontal scaling

- Row store by default
  - Scan over rows for $O(\log n)$ lookup & sorted order
  - Log-structured Merge Tree design
- Iterator processing framework

Use Transpose Tables
see D4M Schema\(^1\)

\(^1\)D4M 2.0 Schema: A General Purpose High Performance Schema for the Accumulo Database
Kepner et al, IEEE HPEC 2013
• Intro to Graphulo
• Intro to Matrix Multiply
• Intro to Accumulo
• Matrix Multiply pre-Graphulo
• Inner Product
• Outer Product
• Accumulo Implementation
• Performance
• Conclusions
Table Multiply Before Graphulo

Diagram showing a Client interacting with an Accumulo system.
Table Multiply Before Graphulo
Table Multiply Before Graphulo

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

Graphulo - TableMult-23
Table Multiply Before Graphulo

Old: DB = Indexed Storage

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

Old:  DB = Indexed Storage
New:  DB = Indexed Storage + Computation Engine

*Blocked algorithms exist for large tables at reduced efficiency
Outline

• Intro to Graphulo
• Intro to Matrix Multiply
• Intro to Accumulo
• Matrix Multiply pre-Graphulo
• Inner Product
• Outer Product
• Accumulo Implementation
• Performance
• Conclusions
Inner Product

\[
\begin{bmatrix}
6 & 5 & 2 \\
4 & 3 & 5 \\
2 & 3 & 4 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
1 \\
6 \\
\end{bmatrix}
\]

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

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

\[
\begin{align*}
\text{word|coffee} & \quad \text{word|desert} \\
\begin{bmatrix}
6 & 5 & 2 \\
4 & & \\
& & \\
& & \\
& & \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{tod|0500} & \quad \text{tod|0800} & \quad \text{tod|1400} \\
\begin{bmatrix}
& & \\
5 & 3 & 4 \\
& & \\
& & \\
& & \\
& & \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\text{word|dew} & \quad \text{word|hot} \\
\begin{bmatrix}
1 & 2 \\
6 & 23 \\
\end{bmatrix}
\end{align*}
\]

\[
\text{tod|0500} \quad \text{tod|0800} \quad \text{tod|1400}
\]

\[
\begin{align*}
\text{word|coffee} & \quad \text{word|dew} & \quad \text{word|hot} \\
\begin{bmatrix}
6 & 3 & 23 \\
\end{bmatrix}
\end{align*}
\]

1st Scan

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

**Algorithm**

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

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

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

```plaintext
for i = 1 : N = 2
    for j = 1 : L = 2
        for k = 1 : M = 4
            C(i, j) \oplus A(i, k) \otimes B(k, j)
```

- word|coffee
- word|desert
- tod|0500
- tod|0800
- tod|1400
- tod|0900
- tod|1400
- word|dew
- word|hot
- word|dew
- word|hot

\(\text{①} = 2\)
\(\text{②} = 4\)
\(\text{①} = 2\)
### Inner Product

For $i = 1: N = 2$

For $j = 1: L = 2$

For $k = 1: M = 4$

$$C(i, j) \oplus = A(i, k) \otimes B(k, j)$$

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

Word | Coffee | Desert
---|---|---
Word | 6 | 5 | 2
Word | 6 | 23 | 12
Word | 3 | 4

```
for i = 1: N = 2
for j = 1: L = 2
for k = 1: M = 4
C(i, j) \oplus = A(i, k) \otimes B(k, j)
```
Inner Product

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

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

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

\text{for } i = 1: N = 2
\text{for } j = 1: L = 2
\text{for } k = 1: M = 4
\text{for } k = 1: M = 4
\text{for } k = 1: M = 4
Outline

- Intro to Graphulo
- Intro to Matrix Multiply
- Intro to Accumulo
- Matrix Multiply pre-Graphulo
- Inner Product
- Outer Product
- Accumulo Implementation
- Performance
- Conclusions
Now explicitly showing $A^T$

$$\begin{align*}
tod|0500 & \begin{bmatrix} 6 \\
5 \\
2 \end{bmatrix} \\
tod|0800 & \begin{bmatrix} 4 \\
3 \\
4 \end{bmatrix} \\
tod|1400 & \begin{bmatrix} 0 \\
0 \\
1 \end{bmatrix}
\end{align*}$$

$$\begin{align*}
tod|0800 & \begin{bmatrix} 3 \\
5 \\
3 \end{bmatrix} \\
tod|0900 & \begin{bmatrix} 0 \\
0 \\
4 \end{bmatrix} \\
tod|1400 & \begin{bmatrix} 0 \\
0 \\
0 \end{bmatrix}
\end{align*}$$

$$\begin{bmatrix} 6 & 2 \end{bmatrix} \begin{bmatrix} 0 & 12 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

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

### 1. Align Rows

| Word | Coffee | Desert |
|------|--------|--------|
| Tod  | 0500   |        |
| Tod  | 0800   | 6      |
| Tod  | 1400   | 2      |

| Word | Dew    | Hot    |
|------|--------|--------|
| Tod  | 0800   | 5      |
| Tod  | 0900   | 3      |
| Tod  | 1400   | 4      |

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

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

1. Align Rows

|   | word | coffee |   | word | desert |   |
|---|------|--------|---|------|--------|---|
| tod| 0500 | 6      |   | tod| 0800 | 5  |
| tod| 0800 | 5      |   | tod| 0900 | 3  |
| tod| 1400 | 2      |   | tod| 1400 | 3  |

\[
\begin{bmatrix}
6 \\
5 \\
2
\end{bmatrix} \begin{bmatrix}
5 \\
4 \\
3
\end{bmatrix} = 
\begin{bmatrix}
\end{bmatrix}
\]

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

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

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

\[
C(i, j) \oplus = A(i, k) \otimes B(k, j)
\]

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

2. Cartesian Product

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

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

\[
\begin{align*}
\text{word|coffee} & \quad \text{word|hot} \\
\end{align*}
\]

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

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

2. Cartesian Product

| tod | 0500 | tod | 0800 | tod | 0900 | tod | 1400 |
|-----|------|-----|------|-----|------|-----|------|
| word | coffee | word | desert |
| 6     | 5     | 4     |
| tod | 0800 | tod | 0900 | tod | 1400 |
| word | dew | word | hot |
| 5     | 3     | 4     |

\[
\text{word|coffee} \quad \text{word|desert} \\
6 \quad 5 \quad 4 \\
\text{word|dew} \quad \text{word|hot} \\
5 \quad 3 \quad 4
\]

\[
\text{word|hot} = 15 \\
\text{word|hot} = 12
\]

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

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

\[
\begin{align*}
\text{tod|0500} & \begin{bmatrix} 6 \\ 5 \\ 2 \end{bmatrix} & \text{tod|0800} & \begin{bmatrix} 0 \\ 4 \\ 3 \end{bmatrix} & \text{tod|0900} & \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} & \text{tod|1400} & \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}
\end{align*}
\]

for \( k = 1: M = 4 \)
for \( i = 1: N = 2 \)
for \( j = 1: L = 2 \)
\[
C(i, j) \oplus A(i, k) \otimes B(k, j)
\]

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

## 1. Align Rows

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

| word  | dew | word  | hot |
|-------|-----|-------|-----|
| tod|0800 | 5 | 3 |  |
| tod|0900 | 3 | 4 |  |
| tod|1400 | 3 | 4 |  |

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

\[
= \begin{bmatrix}
\text{word|coffee} & 15 \\
\text{word|desert} & 12 
\end{bmatrix}
\]
## Outer Product

### 2. Cartesian Product

For $k = 1: M = 4$

For $i = 1: N = 2$

For $j = 1: L = 2$

$C(i, j) \oplus A(i, k) \otimes B(k, j)$
## Outer Product

### 2. Cartesian Product

Accumulo stores both 15 and 8 until next scan or compaction.

\[
\begin{array}{ccc}
\text{word|coffee} & \text{word|desert} \\
\text{tod|0500} & \begin{bmatrix} 6 \\ 5 \end{bmatrix} & \begin{bmatrix} 4 \end{bmatrix} \\
\text{tod|0800} & \begin{bmatrix} 3 \end{bmatrix} & \begin{bmatrix} 3 \end{bmatrix} \\
\text{tod|1400} & \begin{bmatrix} 2 \end{bmatrix} & \begin{bmatrix} 4 \end{bmatrix}
\end{array}
\]

\[
\begin{bmatrix}
\text{word|dew} \\
\text{word|hot}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{word|coffee} \\
\text{word|desert}
\end{bmatrix}
\]

\[
\begin{bmatrix}
6 \\
12
\end{bmatrix}
\]

\[
\begin{bmatrix}
23 \\
4
\end{bmatrix}
\]

\[
\begin{bmatrix}
3 \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
2 \\
4
\end{bmatrix}
\]

\[
\begin{array}{l}
\text{for } k = 1: M = 4 \\
\text{for } i = 1: N = 2 \\
\text{for } j = 1: L = 2 \\
\quad C(i, j) \oplus A(i, k) \otimes B(k, j)
\end{array}
\]

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

- No write locality; unsorted writes
- Hard to pre-sum partial products (4 entries written)

+ Single scan over table B

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

\[
\begin{array}{c}
\text{For } k = 1: M = 4 \\
\text{For } i = 1: N = 2 \\
\text{For } j = 1: L = 2 \\
C(i, j) \oplus A(i, k) \otimes B(k, j)
\end{array}
\]

*Lazy \(\oplus\): Accumulo stores both 15 and 8 until next scan or compaction
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
Outline

• Intro to Graphulo
• Intro to Matrix Multiply
• Intro to Accumulo
• Matrix Multiply pre-Graphulo
• Inner Product
• Outer Product
• Accumulo Implementation
• Performance
• Conclusions
Outer Product in Graphulo Iterators
Accumulo Distributes Graphulo Iterators

- Tablets can be hosted on any tablet server
  - Accumulo load balances tablet allocation
- Matrix multiply iterators run on B's tablets in parallel
  - Scan from A's tablets in parallel
  - BatchWrite to C's tablets in parallel
Outline

• Intro to Graphulo
• Intro to Matrix Multiply
• Intro to Accumulo
• Matrix Multiply pre-Graphulo
• Inner Product
• Outer Product
• Accumulo Implementation
• Performance
• Conclusions
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
  - Unpermuted 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

TableMult Rate Scaling

- Graphulo 1 Tablet
- Graphulo 2 Tablets
- D4M 1 Tablet
- D4M 2 Tablets

Rate (partial products/s) vs SCALE

Graphulo-TableMult-49
Outline

• Intro to Graphulo
• Intro to Matrix Multiply
• Intro to Accumulo
• Matrix Multiply pre-Graphulo
• Inner Product
• Outer Product
• Accumulo Implementation
• Performance
• Conclusions
Conclusion

• Promising performance
  – Write rates near 400k / sec, near highest single-node recorded rates
  – Experiments on a larger cluster will confirm weak & strong scaling

• Outer product better suited to Accumulo
  – Hybrid inner-outer product algorithms worth studying

• Current Graphulo research is
  – implementing remaining GraphBLAS
  – developing graph algorithms
## TABLE I: Output Table C Sizes and Experiment Timings

| SCALE | Entries in Table C | Graphulo 1 Tablet | D4M 1 Tablet | Graphulo 2 Tablets | D4M 2 Tablets |
|-------|--------------------|-------------------|--------------|-------------------|--------------|
|       | PartialProducts    | AfterSum          | Time (s)     | Rate (pp/s)       | Time (s)     | Rate (pp/s)  | Time (s)     | Rate (pp/s)     | Time (s)     | Rate (pp/s)  |
| 10    | $8.05 \times 10^5$ | $2.69 \times 10^5$| 2.87         | $2.81 \times 10^5$ | 3.02         | $2.67 \times 10^5$ | 2.02         | $3.98 \times 10^5$ | 2.80         | $2.87 \times 10^5$ |
| 11    | $2.36 \times 10^6$ | $8.15 \times 10^5$| 7.76         | $3.04 \times 10^5$ | 8.80         | $2.68 \times 10^5$ | 5.19         | $4.55 \times 10^5$ | 8.72         | $2.71 \times 10^5$ |
| 12    | $6.82 \times 10^6$ | $2.43 \times 10^6$| $2.20 \times 10^1$ | $3.10 \times 10^5$ | $2.66 \times 10^1$ | $2.56 \times 10^5$ | $1.63 \times 10^1$ | $4.18 \times 10^5$ | $2.62 \times 10^1$ | $2.60 \times 10^5$ |
| 13    | $1.91 \times 10^7$ | $7.04 \times 10^6$| $6.40 \times 10^1$ | $2.99 \times 10^5$ | $1.50 \times 10^2$ | $1.27 \times 10^5$ | $4.86 \times 10^1$ | $3.93 \times 10^5$ | $1.44 \times 10^2$ | $1.33 \times 10^5$ |
| 14    | $5.27 \times 10^7$ | $2.00 \times 10^7$| $1.82 \times 10^2$ | $2.90 \times 10^5$ | $5.79 \times 10^2$ | $9.09 \times 10^4$ | $1.36 \times 10^2$ | $3.87 \times 10^5$ | $5.59 \times 10^2$ | $9.42 \times 10^4$ |
| 15    | $1.47 \times 10^8$ | $5.83 \times 10^7$| $5.03 \times 10^2$ | $2.93 \times 10^5$ | $2.51 \times 10^3$ | $5.86 \times 10^4$ | $3.94 \times 10^2$ | $3.74 \times 10^5$ | $2.56 \times 10^3$ | $5.75 \times 10^4$ |
| 16    | $4.00 \times 10^8$ | $1.63 \times 10^8$| $1.39 \times 10^3$ | $2.88 \times 10^5$ |                 |                 | $1.18 \times 10^3$ | $3.40 \times 10^5$ |                 |                 |
| 17    | $1.09 \times 10^9$ | $4.59 \times 10^8$| $4.06 \times 10^3$ | $2.67 \times 10^5$ |                 |                 | $3.70 \times 10^3$ | $2.94 \times 10^5$ |                 |                 |
| 18    | $2.94 \times 10^9$ | $1.28 \times 10^9$| $1.21 \times 10^4$ | $2.42 \times 10^5$ |                 |                 | $1.14 \times 10^4$ | $2.58 \times 10^5$ |                 |                 |
Inner-Outer Hybrid Algorithm

\[
\text{for } p = 1 : P \\
\quad \text{for } k = 1 : M \\
\quad \quad \text{for } i = \left( \left\lfloor \frac{(p-1)N}{P} \right\rfloor + 1 \right) : \left\lfloor \frac{pN}{P} \right\rfloor \\
\quad \quad \quad \text{for } j = 1 : L \\
\quad \quad \quad \quad C(i, j) \oplus= A(i, k) \otimes B(k, j)
\]

\[P = N \quad \text{– Inner Product}\]
\[P = 1 \quad \text{– Outer Product}\]
### D4M Schema for Sparse Arrays in Key/Value Databases (Accumulo)

#### Input Data

| Time     | Col1 | Col2 | Col3 |
|----------|------|------|------|
| 2001-01-01 | a    |      | a    |
| 2001-01-02 | b    | b    |      |
| 2001-01-03 | c    | c    |      |

#### Accumulo Table: \( T_{\text{transpose}} \)

|        | 01-01-2001 | 02-01-2001 | 03-01-2001 |
|--------|------------|------------|------------|
| Col1|a         | 1          |            |
| Col1|b         | 1          |            |
| Col2|b         | 1          |            |
| Col2|c         |            | 1          |
| Col3|a         |            | 1          |
| Col3|c         |            |            |

#### Accumulo Table: \( T \)

|        | Col1|a | Col1|b | Col2|b | Col2|c | Col3|a | Col3|c |
|--------|-----|---|-----|---|-----|---|-----|---|-----|---|-----|---|
| 01-01-2001 | 1   |   |     |   |     |   |     |   | 1    |   |     |   |
| 02-01-2001 |     | 1 |     | 1 |     |   |     |   |       |   |     |   |
| 03-01-2001 |     |   |     |   | 1    |   |     | 1 |       |   |     |   |

- Tabular data expanded to create many type/value columns
- Transpose pairs allows quick look up of either row or column

---

1. **D4M 2.0 Schema: A General Purpose High Performance Schema for the Accumulo Database**

Kepner et al, IEEE HPEC 2013