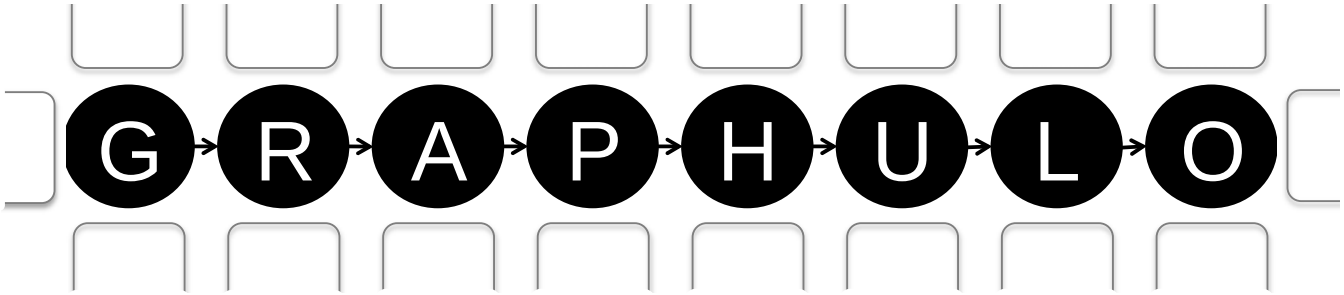


Graph Analytics in GraphBLAS

Vijay Gadepally (vijayg@ll.mit.edu)

Dylan Hutchison, Jeremy Kepner

MIT Lincoln Laboratory

July 2015

Massachusetts Institute of Technology

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

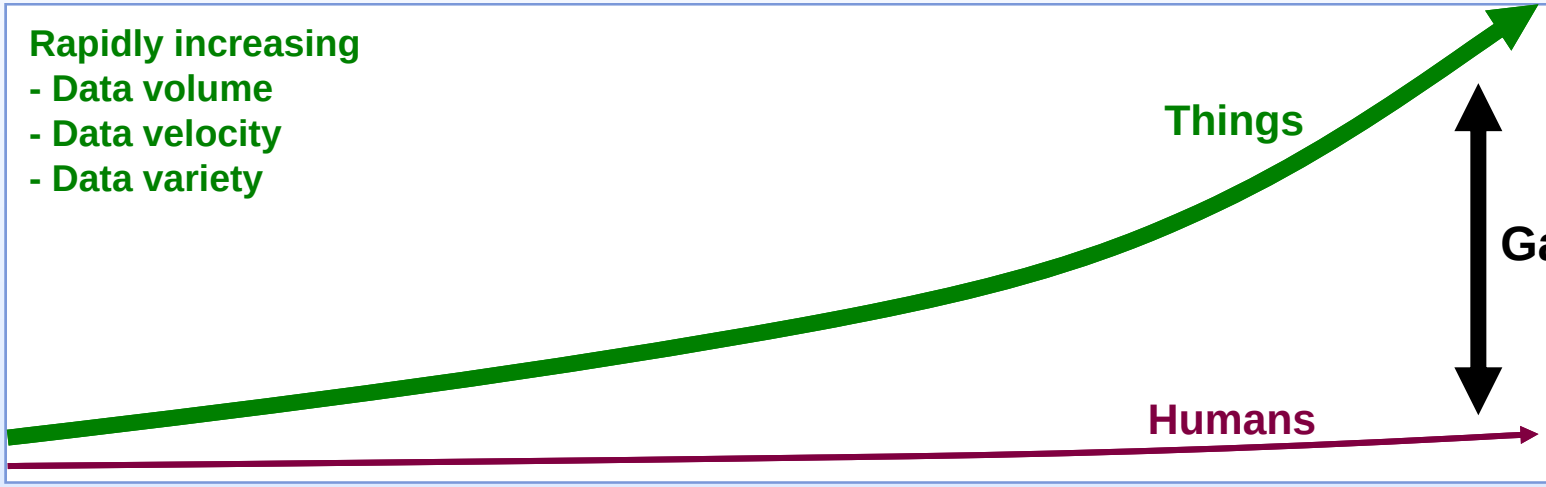
Humans
(deciders)

Kids

Adults

Elderly

Rapidly increasing
- Data volume
- Data velocity
- Data variety



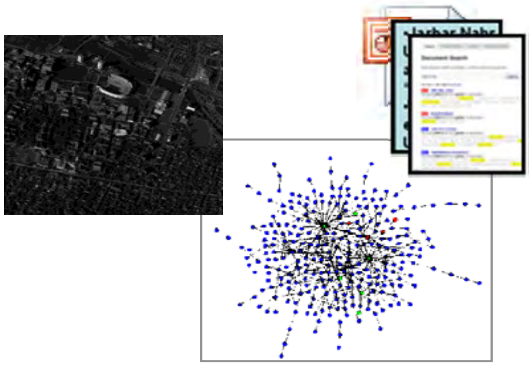
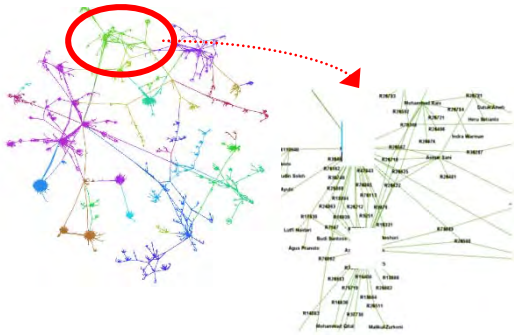
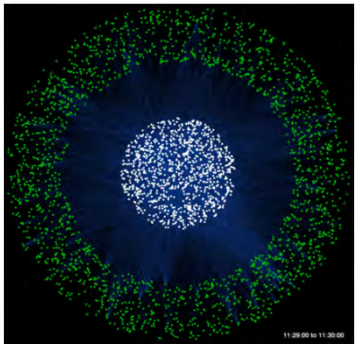
Things
(providers)

Building Security | Building Environment | Building Usage

Commuter Vehicles | Work Vehicles | Transport Vehicles

Student Smartphones | Classroom Tablets | Fitness Wearables

Graph Analytics

ISR	Social	Cyber
		
• Graphs represent entities and relationships detected through multi-INT sources • 1,000s – 1,000,000s tracks and locations • GOAL: Identify anomalous patterns of life	• Graphs represent relationships between individuals or documents • 10,000s – 10,000,000s individual and interactions • GOAL: Identify hidden social networks	• Graphs represent communication patterns of computers on a network • 1,000,000s – 1,000,000,000s network events • GOAL: Detect cyber attacks or malicious software

Graph analytics are used in many missions to solve a variety of problems

Outline

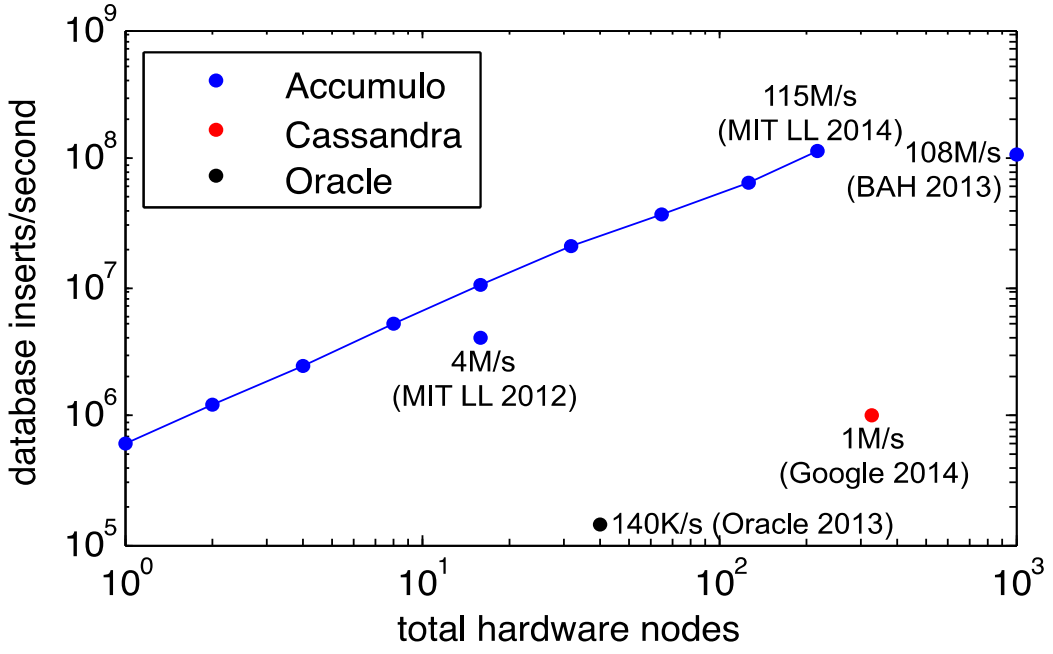
- Introduction
- Graphulo, Accumulo and the GraphBLAS
- Inner and Outer Products
- Graphulo Implementation and Performance
- Next Steps

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

Key					Value
Row ID	Column			Timestamp	
	Family	Qualifier	Visibility		

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**

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

GraphBLAS

- **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/**
- **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

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

GraphBLAS initial function list

Function	Parameters	Returns	Math Notation
SpGEMM	- sparse matrices **A** and **B** - unary functors (op)	sparse matrix	**C** = op(**A**) * op(**B**)
SpM{Sp}V **(Sp: sparse)**	- sparse matrix **A** - sparse/dense vector **x**	sparse/dense vector	**y** = **A** * **x**
SpEWiseX	- sparse matrices or vectors - binary functor and predicate	in place or sparse matrix/vector	**C** = **A** .* **B**
Reduce	- sparse matrix **A** and functors	dense vector	**y** = sum(**A**, op)
SpRef	- sparse matrix **A** - index vectors **p** and **q**	sparse matrix	**B** = **A**(**p**,**q**)
SpAsgn	- sparse matrices **A** and **B** - index vectors **p** and **q**	none	**A**(**p**,**q**) = **B**
Scale	- sparse matrix **A** - dense matrix or vector **X**	none	check manual
Apply	- any matrix or vector **X** - unary functor (op)	none	op(**X**)

GraphBLAS initial function list

Function	Parameters	Returns	Math Notation
SpGEMM	- sparse matrices **A** and **B** - unary functors (op)	sparse matrix	**C** = op(**A**) * op(**B**)
SpM{Sp}V **(Sp: sparse)**	- sparse matrix **A** - sparse/dense vector **x**	sparse/dense vector	**y** = **A** * **x**
SpEWiseX	- sparse matrices or vectors - binary functor and predicate	in place or sparse matrix/vector	**C** = **A** .* **B**
Reduce	- sparse matrix **A** and functors	dense vector	**y** = sum(**A**, op)
SpRef	- sparse matrix **A** - index vectors **p** and **q**	sparse matrix	**B** = **A**(**p**,**q**)
SpAsgn	- sparse matrices **A** and **B** - index vectors **p** and **q**	none	**A**(**p**,**q**) = **B**
Scale	- sparse matrix **A** - dense matrix or vector **X**	none	check manual
Apply	- any matrix or vector **X** - unary functor (op)	none	op(**X**)

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

	tod|0500	tod|0800	tod|0900	tod|1400
word|coffee	6	5	0	2
word|desert	0	4	0	0

×

	word|dew	word|hot
tod|0500	0	0
tod|0800	0	3
tod|0900	5	0
tod|1400	3	4

=

	word|dew	word|hot
word|coffee	6	23
word|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

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

×

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

=

	word|dew	word|hot
word|coffee	6	23
word|desert		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**

➔ **Associative Array Mathematics[1]**

Database Table Multiply

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

×

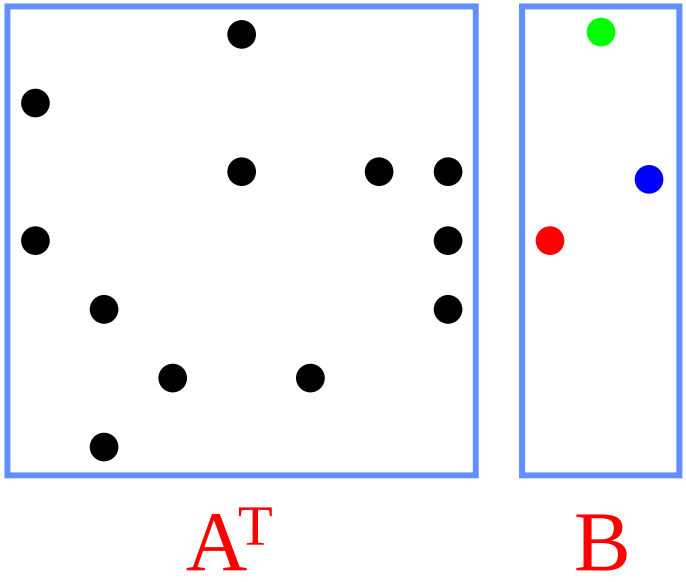
	word|dew	word|hot
tod|0800		3
tod|0900	5	
tod|1400	3	4

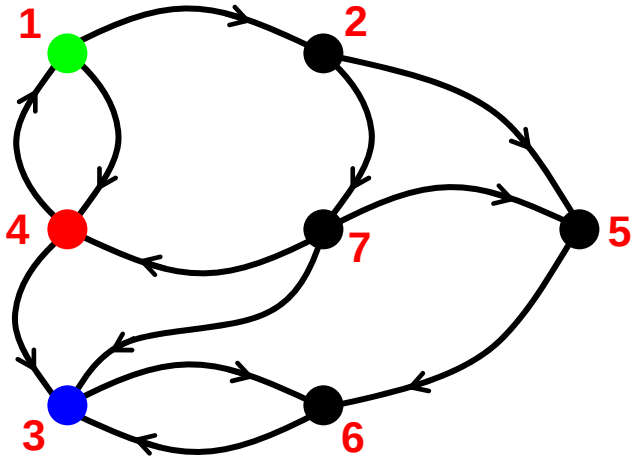
=

	word|dew	word|hot
word|coffee	6	23
word|desert		12

[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

Multiple-source breadth-first search

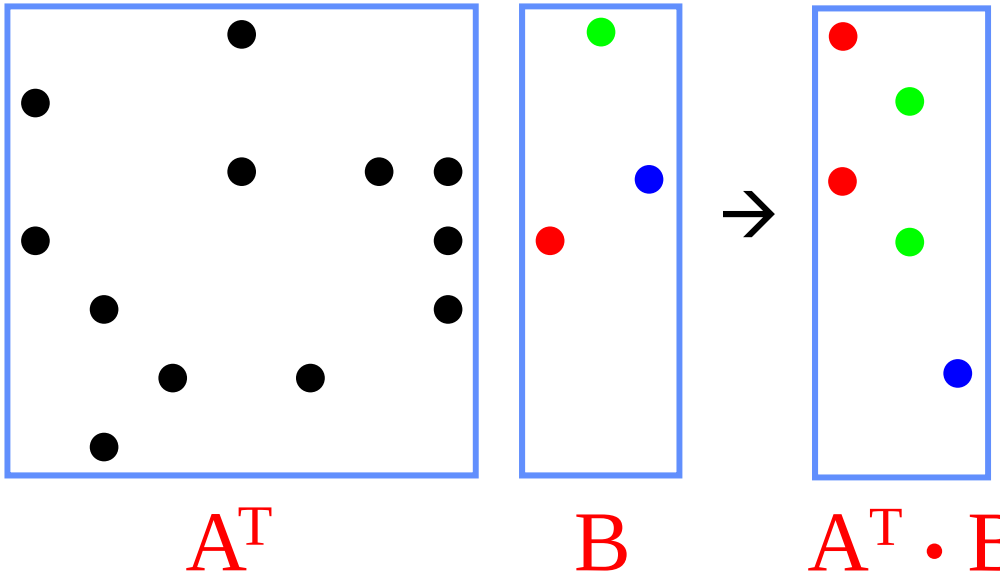


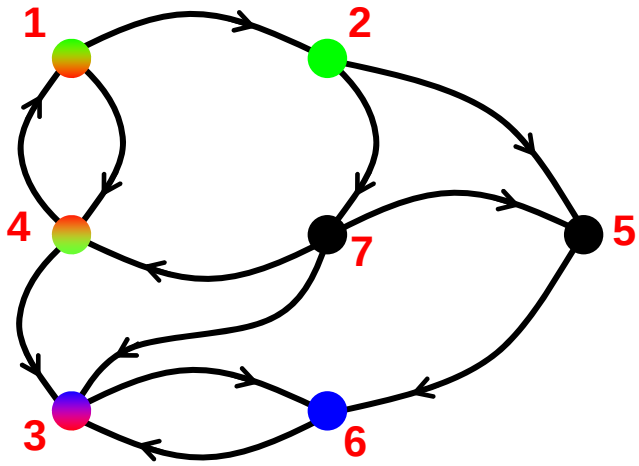


- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Highly-parallel implementation for Betweenness Centrality*

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

Multiple-source breadth-first search

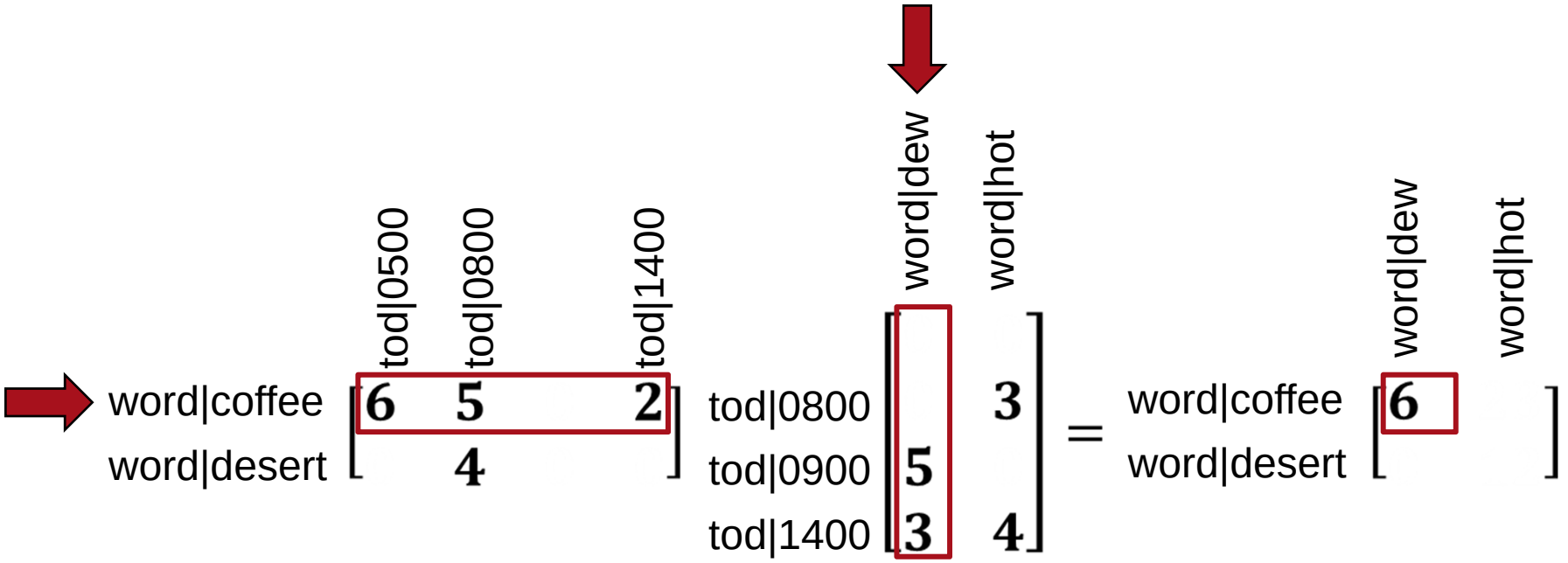




- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Highly-parallel implementation for Betweenness Centrality*

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

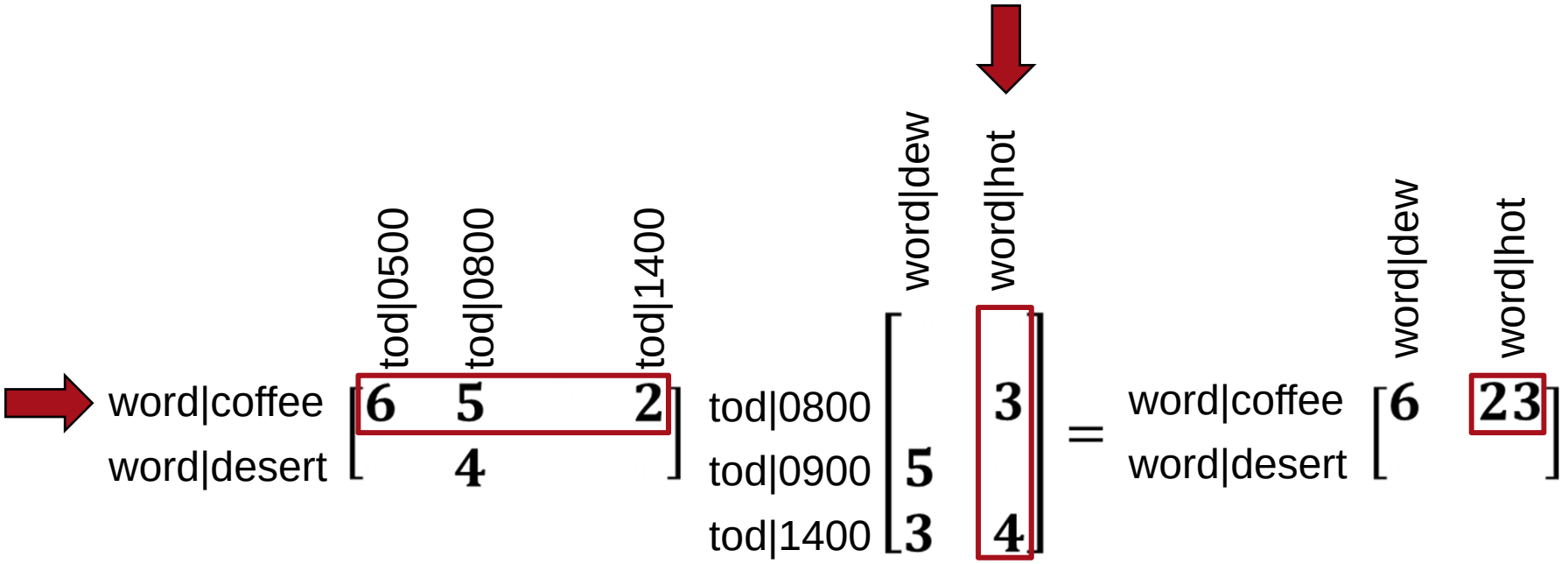
Inner Product



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

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

Inner Product

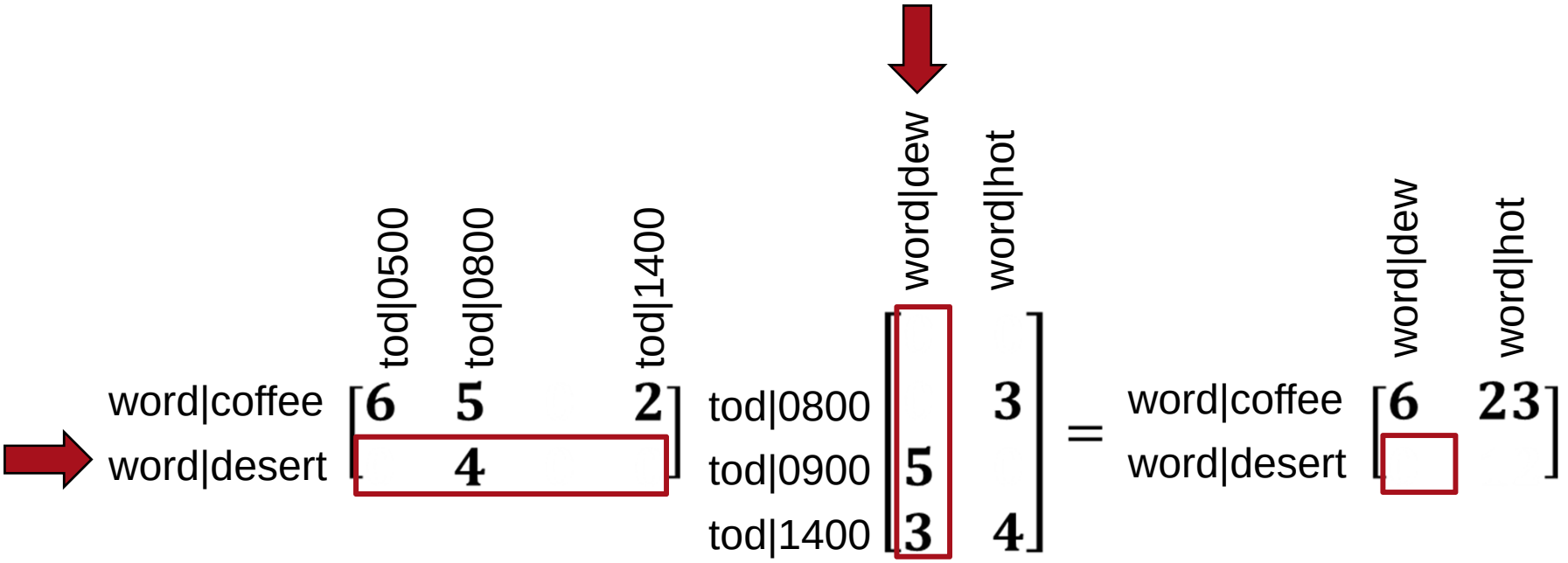


1st Scan

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

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

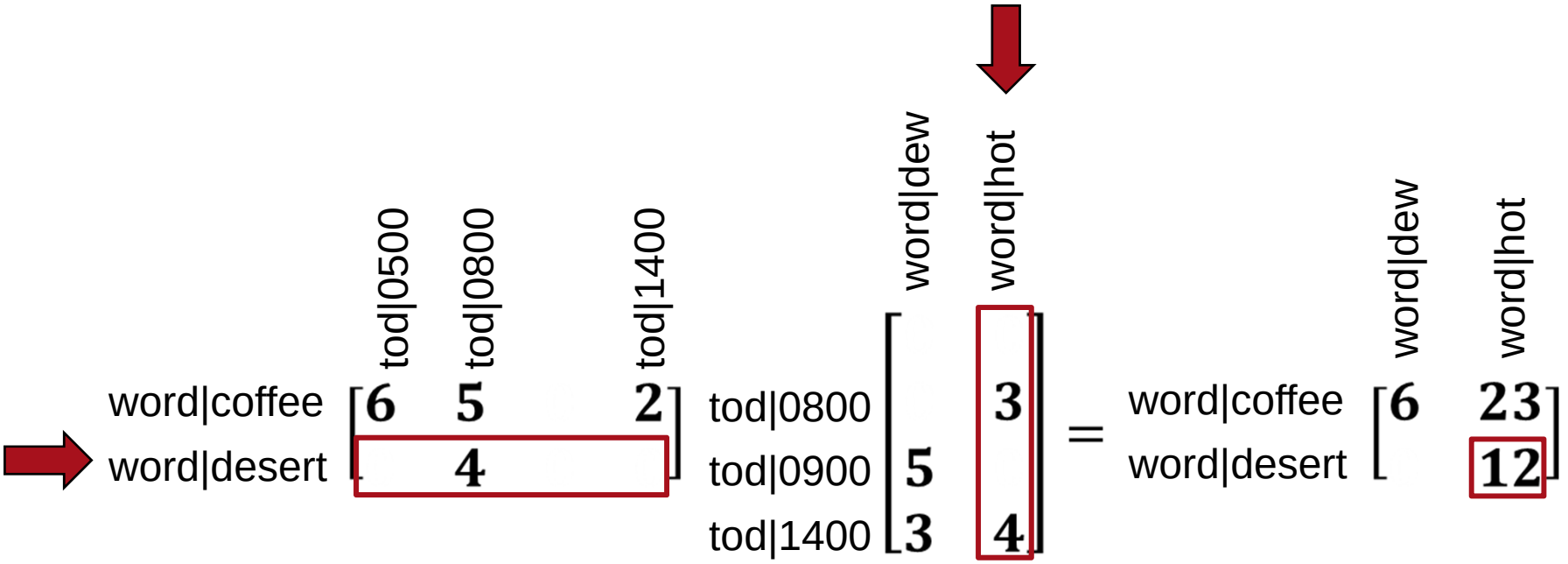
Inner Product



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

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

Inner Product



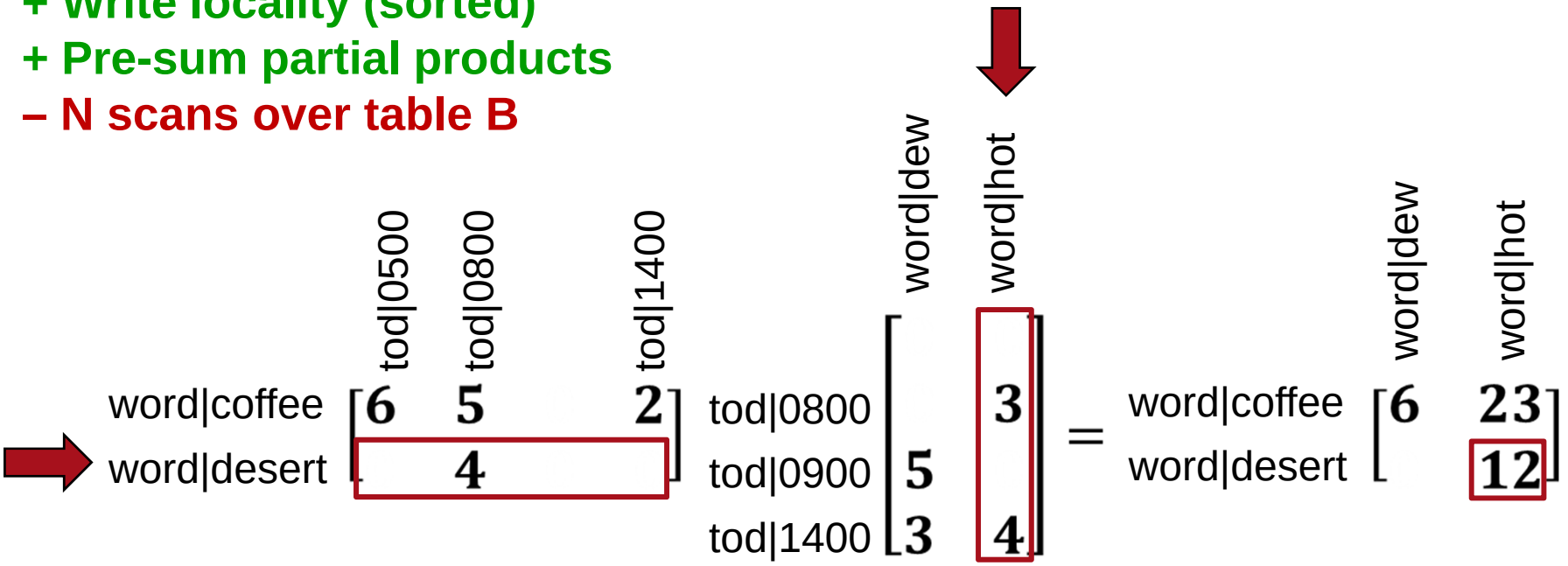
2nd Scan

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

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

Inner Product

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



2nd Scan

$$\textbf{for } i = 1 : N = 2$$
$$\quad \textbf{for } j = 1 : L = 2$$
$$\quad\quad \textbf{for } k = 1 : M = 4$$
$$\quad\quad\quad \textbf{emit } \mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$$

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

Outer Product

Now explicitly showing A^T

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

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

=

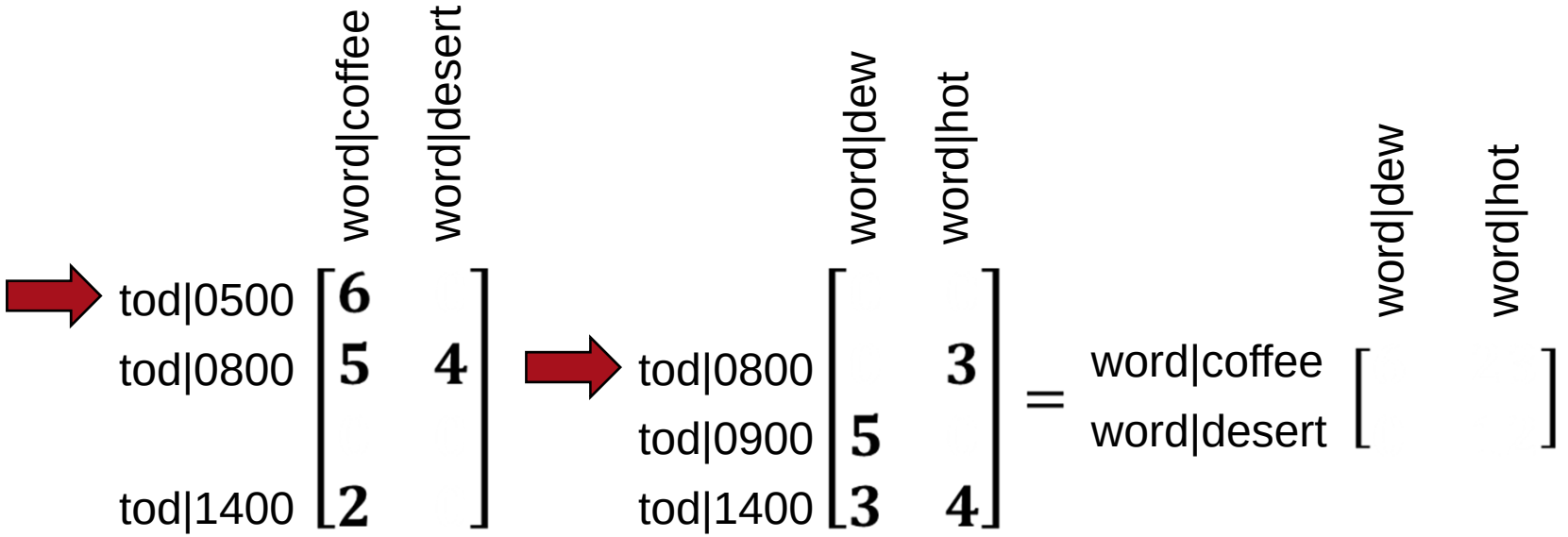
	word|dew	word|hot
word|coffee		
word|desert		

```
for k = 1:M = 4
    for i = 1:N = 2
        for j = 1:L = 2
            emit A(i,k) ⊗ B(k,j)
```

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

Outer Product

1. Align Rows

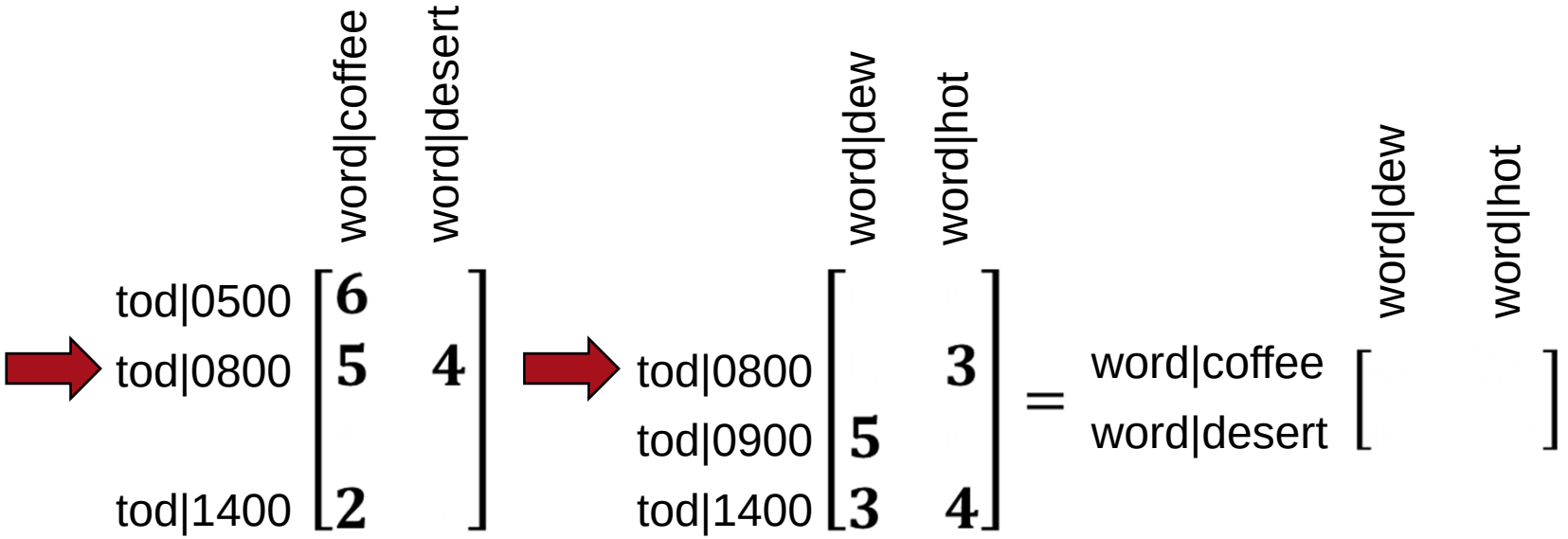


for $k = 1 : M$ = 4
  for $i = 1 : N$ = 2
    for $j = 1 : L$ = 2
      emit $\mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$

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

Outer Product

1. Align Rows

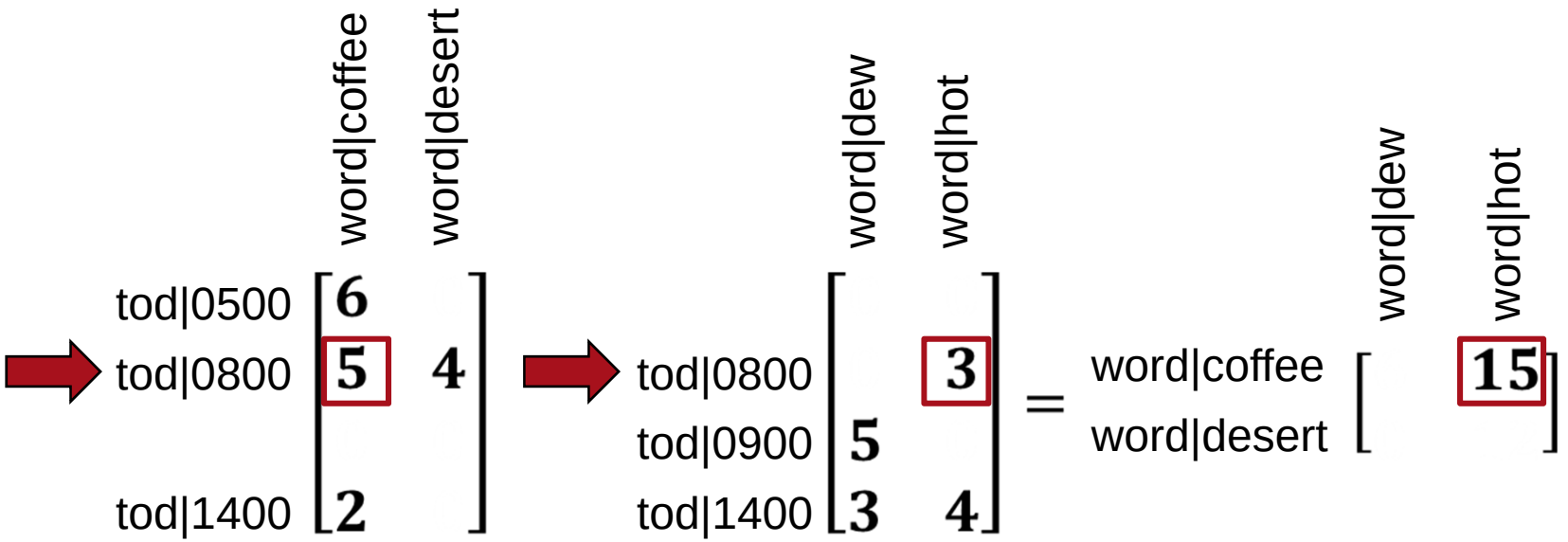


for $k = 1 : M$ = 4
  for $i = 1 : N$ = 2
    for $j = 1 : L$ = 2
      emit $\mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$

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

Outer Product

2. Cartesian Product

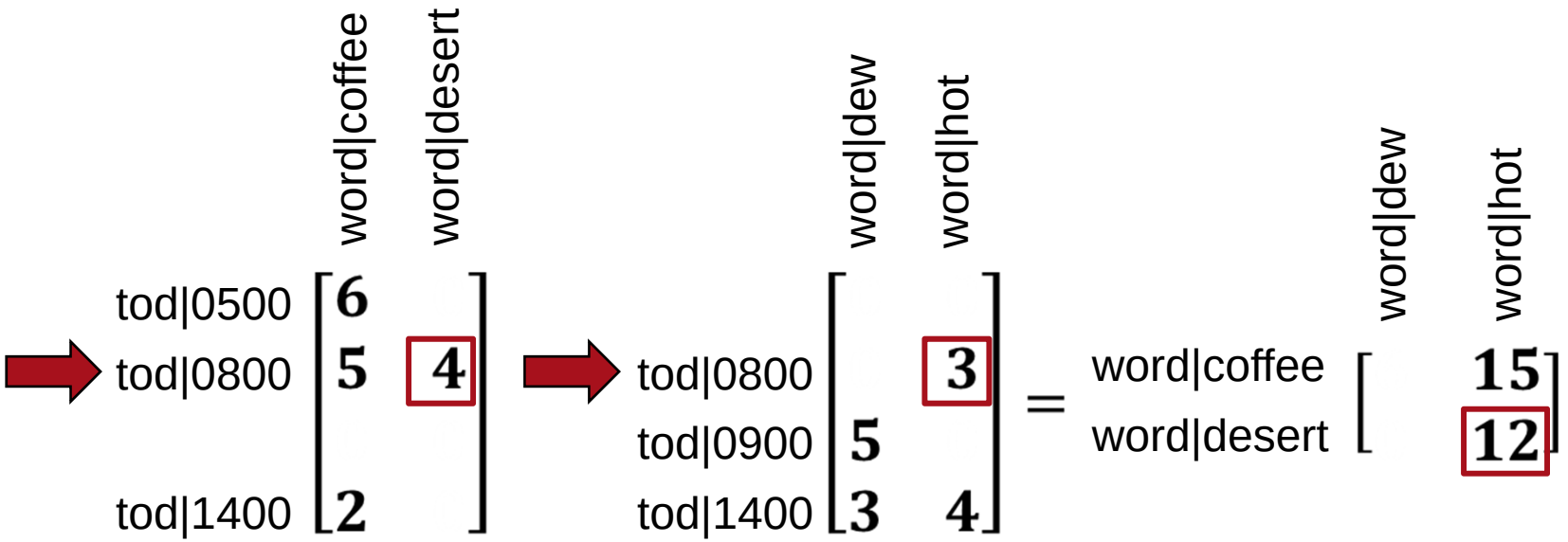


$$\textbf{for } k = 1 : M = 4$$
$$\quad \textbf{for } i = 1 : N = 2$$
$$\quad\quad \textbf{for } j = 1 : L = 2$$
$$\quad\quad\quad \textbf{emit } \mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$$

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

Outer Product

2. Cartesian Product

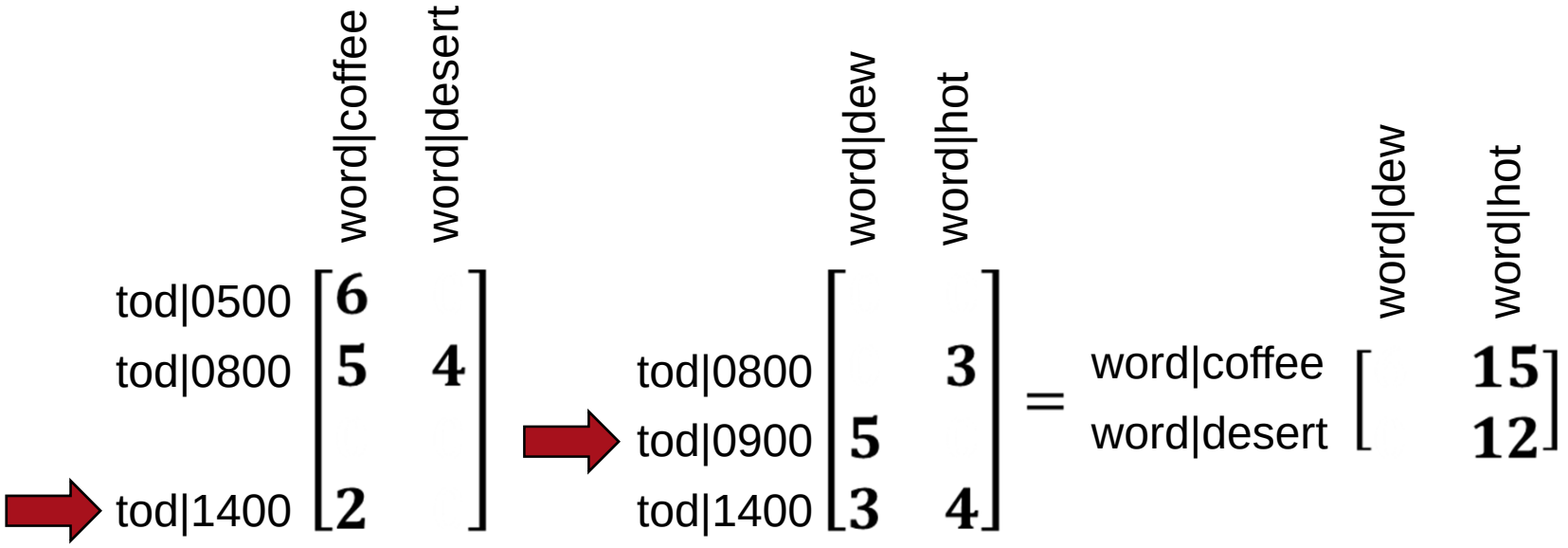


for $k = 1 : M$ = 4
  for $i = 1 : N$ = 2
    for $j = 1 : L$ = 2
      emit $\mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$

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

Outer Product

1. Align Rows

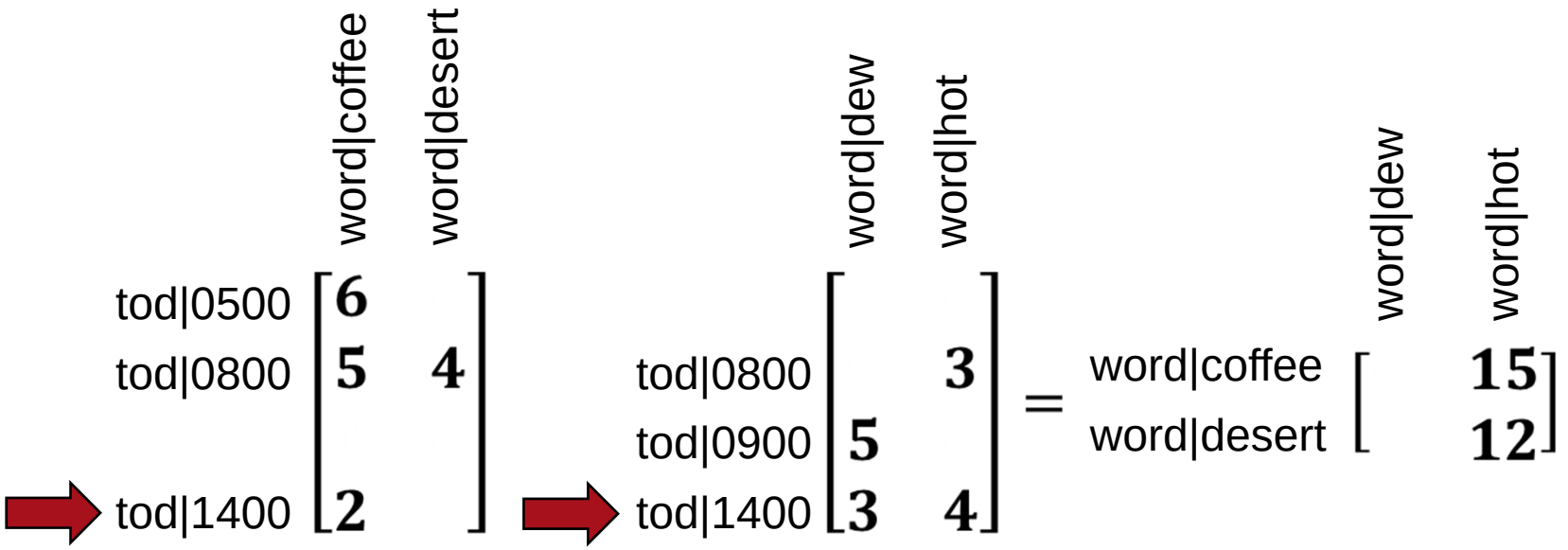


for $k = 1 : M$ = 4
  for $i = 1 : N$ = 2
    for $j = 1 : L$ = 2
      emit $\mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$

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

Outer Product

1. Align Rows

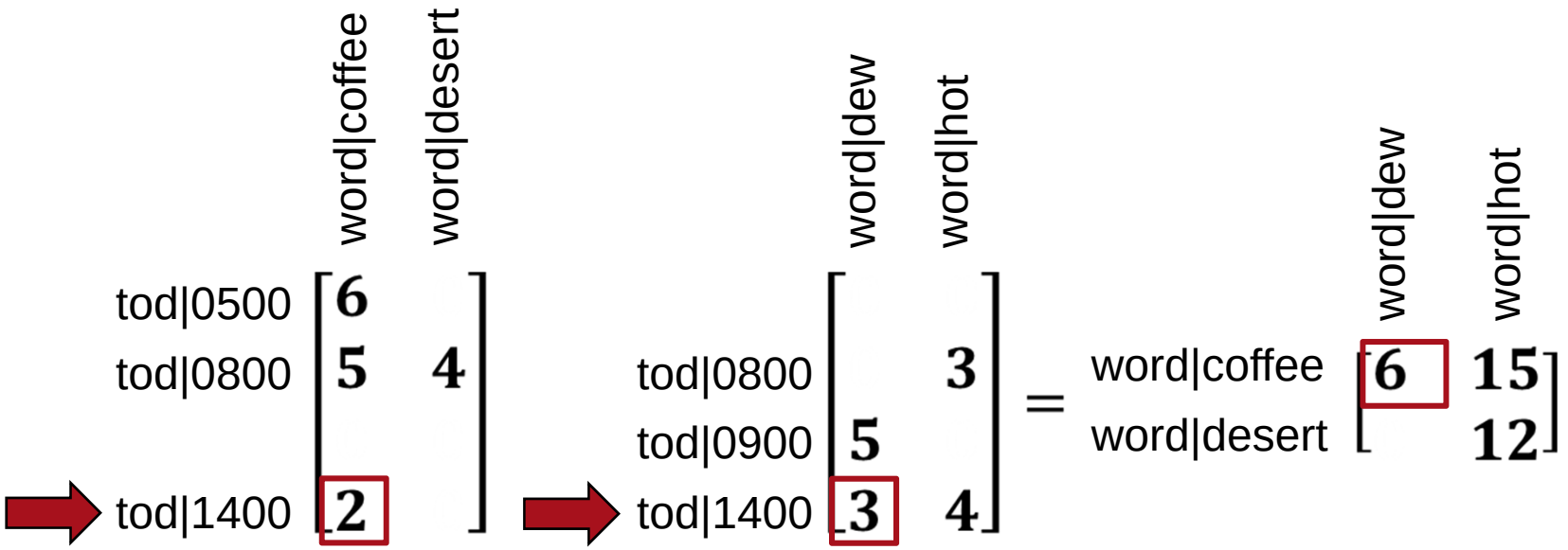


for $k = 1 : M$ = 4
  for $i = 1 : N$ = 2
    for $j = 1 : L$ = 2
      emit $\mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$

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

Outer Product

2. Cartesian Product

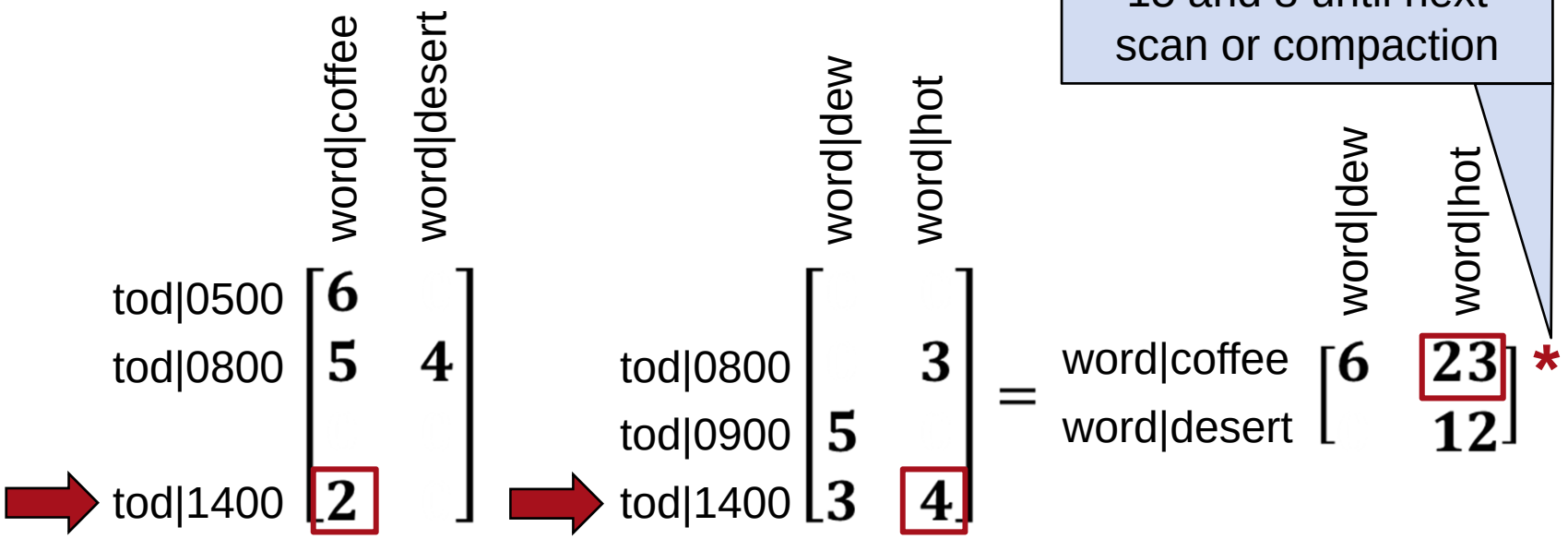


for $k = 1 : M$ = 4
  for $i = 1 : N$ = 2
    for $j = 1 : L$ = 2
      emit $\mathbf{A}(i,k) \otimes \mathbf{B}(k,j)$

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

Outer Product

2. Cartesian Product



Lazy ⊕:
Accumulo stores both 15 and 8 until next scan or compaction

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

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

Outline

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

Table Multiply Before Graphulo

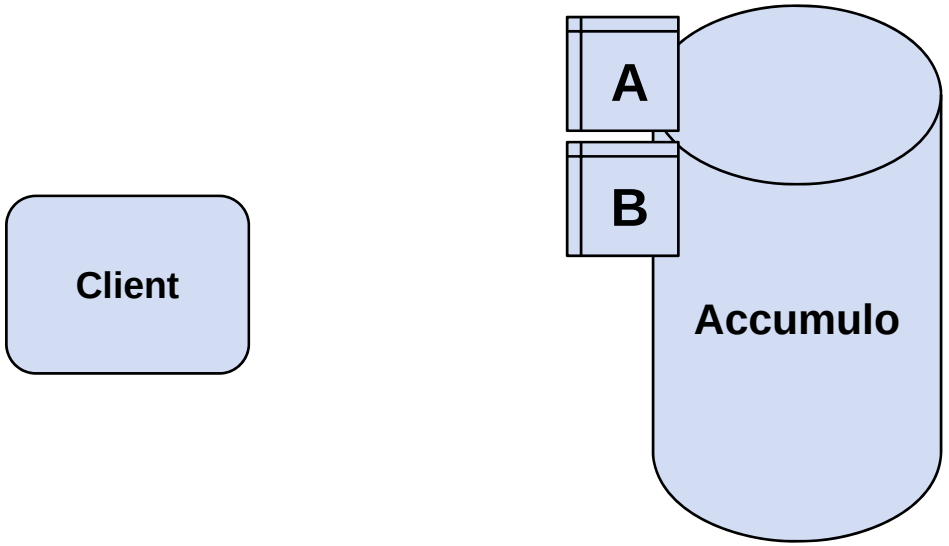


Table Multiply Before Graphulo

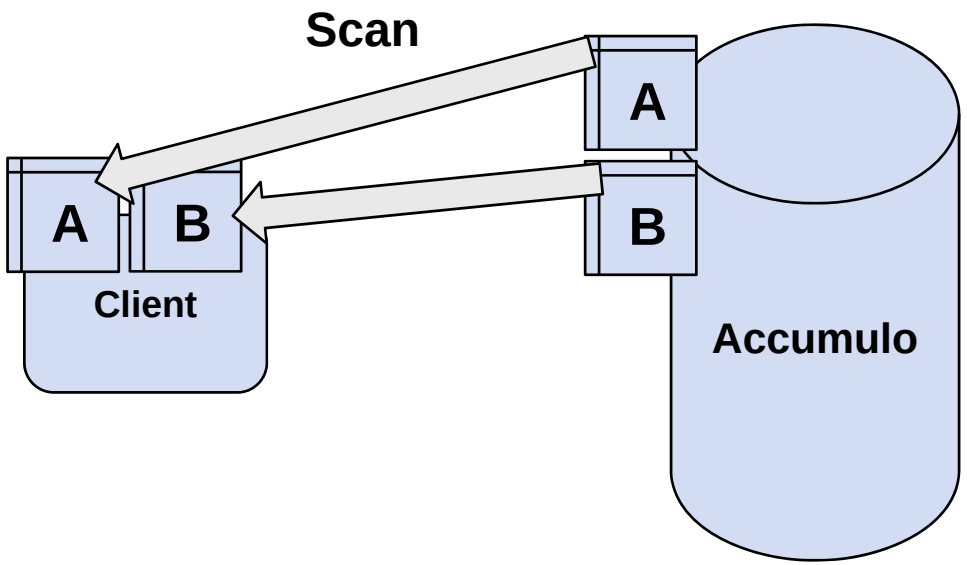
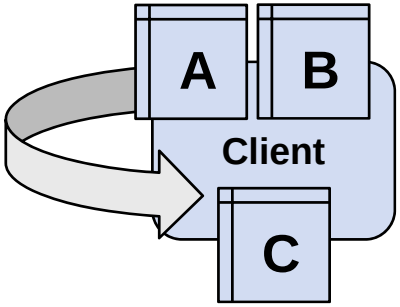
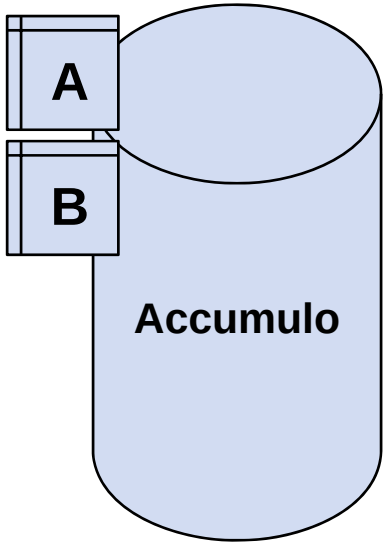


Table Multiply Before Graphulo

Multiply in-memory*





*Blocked algorithms exist for large tables at reduced efficiency

Table Multiply Before Graphulo

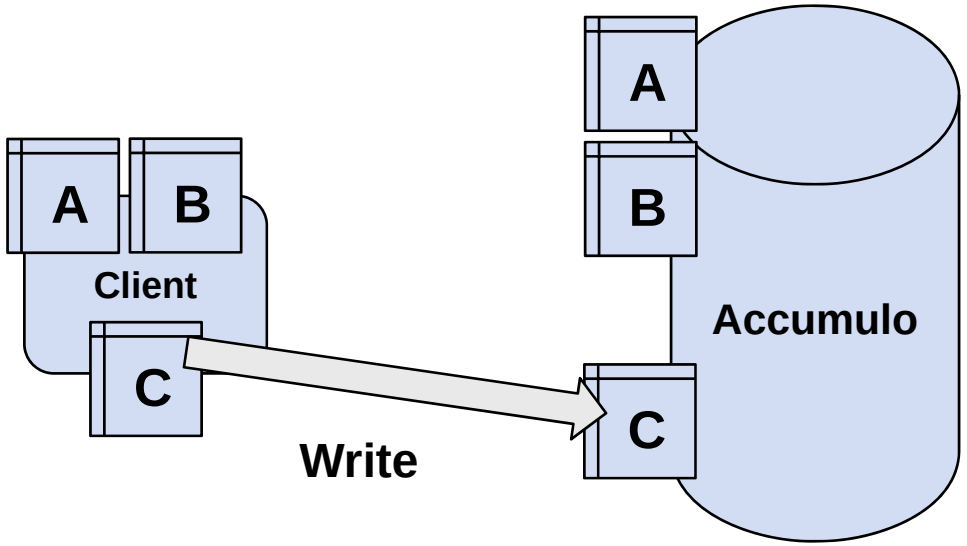
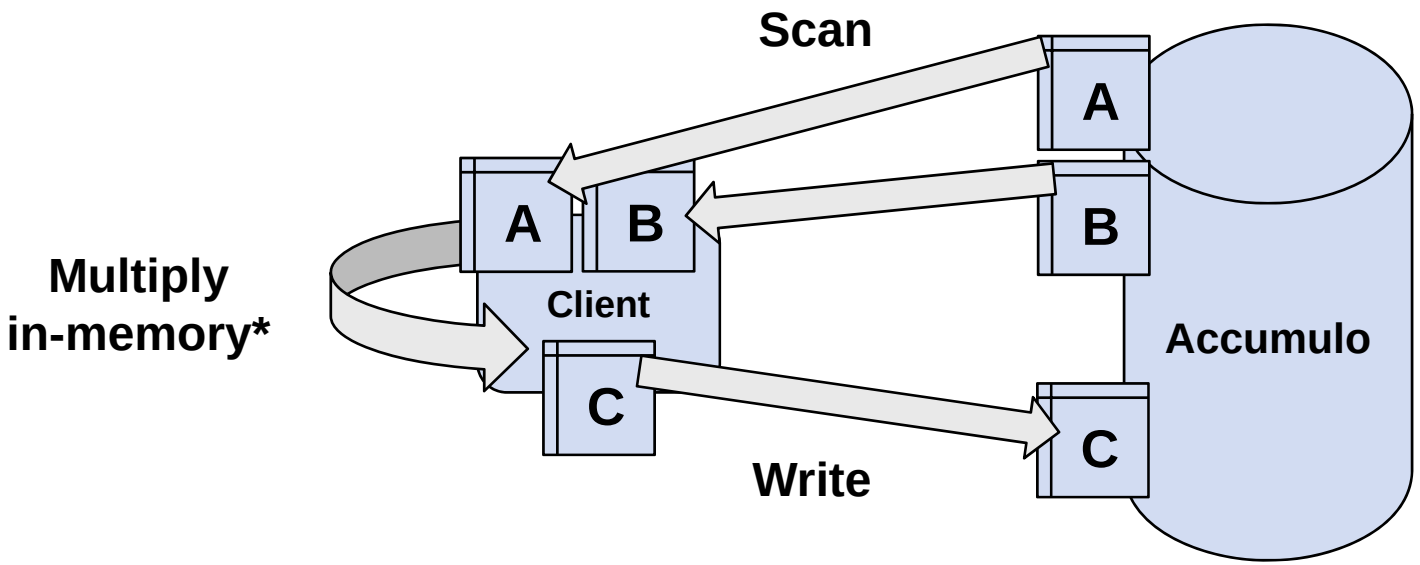


Table Multiply Before Graphulo



*Blocked algorithms exist for large tables at reduced efficiency

Outer Product in Graphulo Iterators

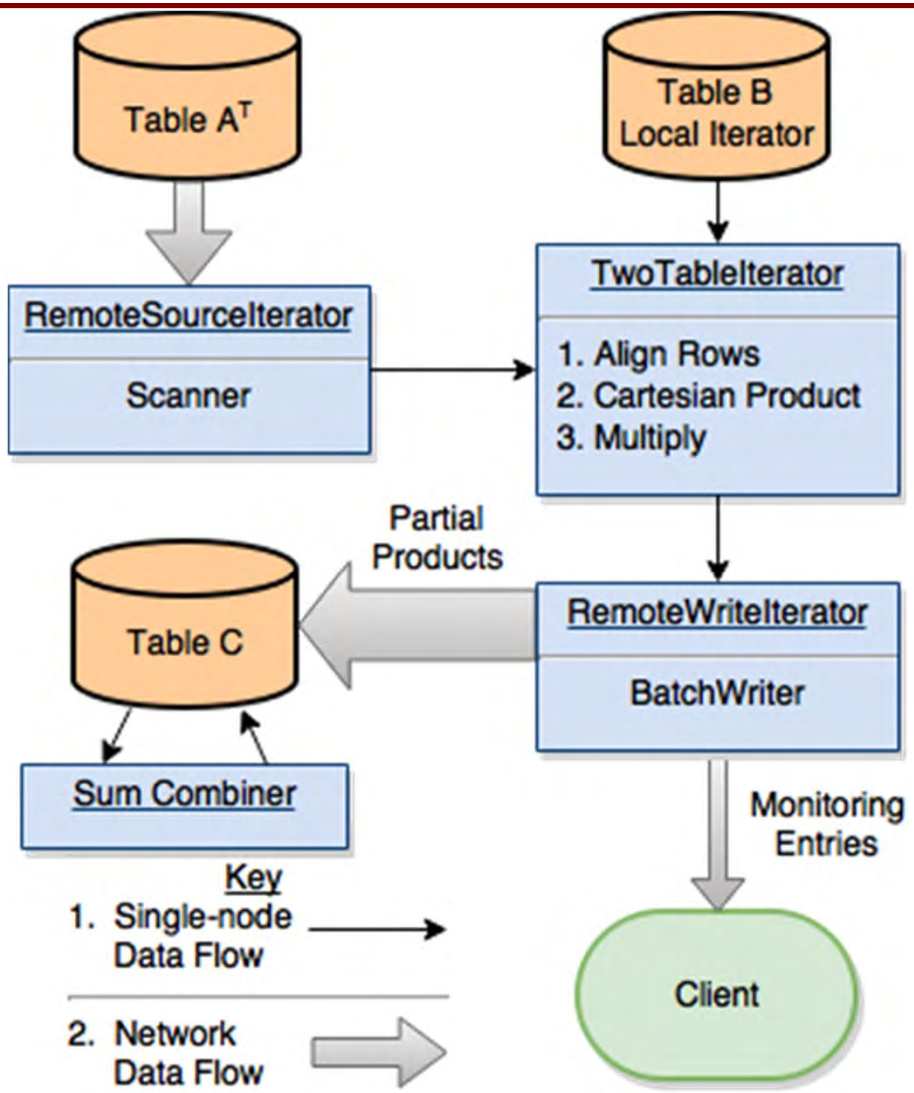
Assumes Tables A and B are in the database

Custom Iterator performs Outer Product

Partial Products to Table C

Lazy Addition to Sum Partial Products

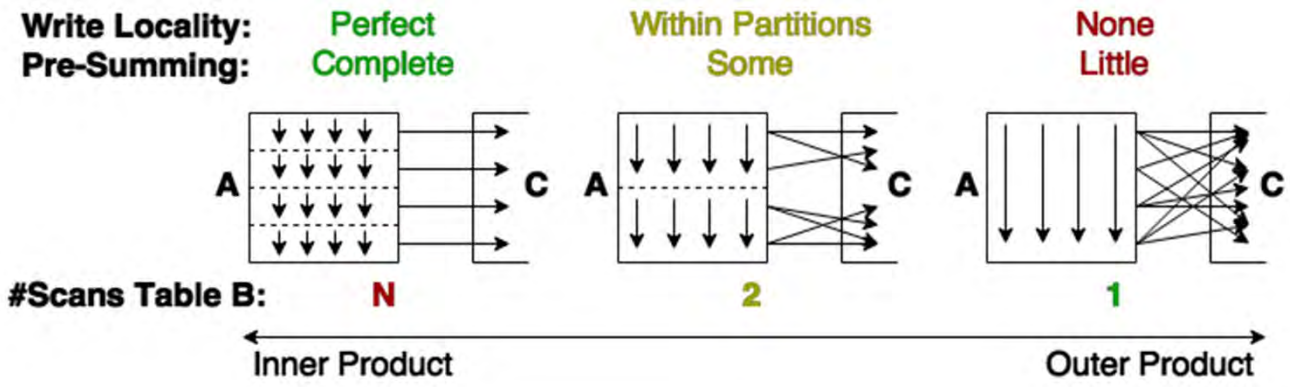
Entries sent to client if requested



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**

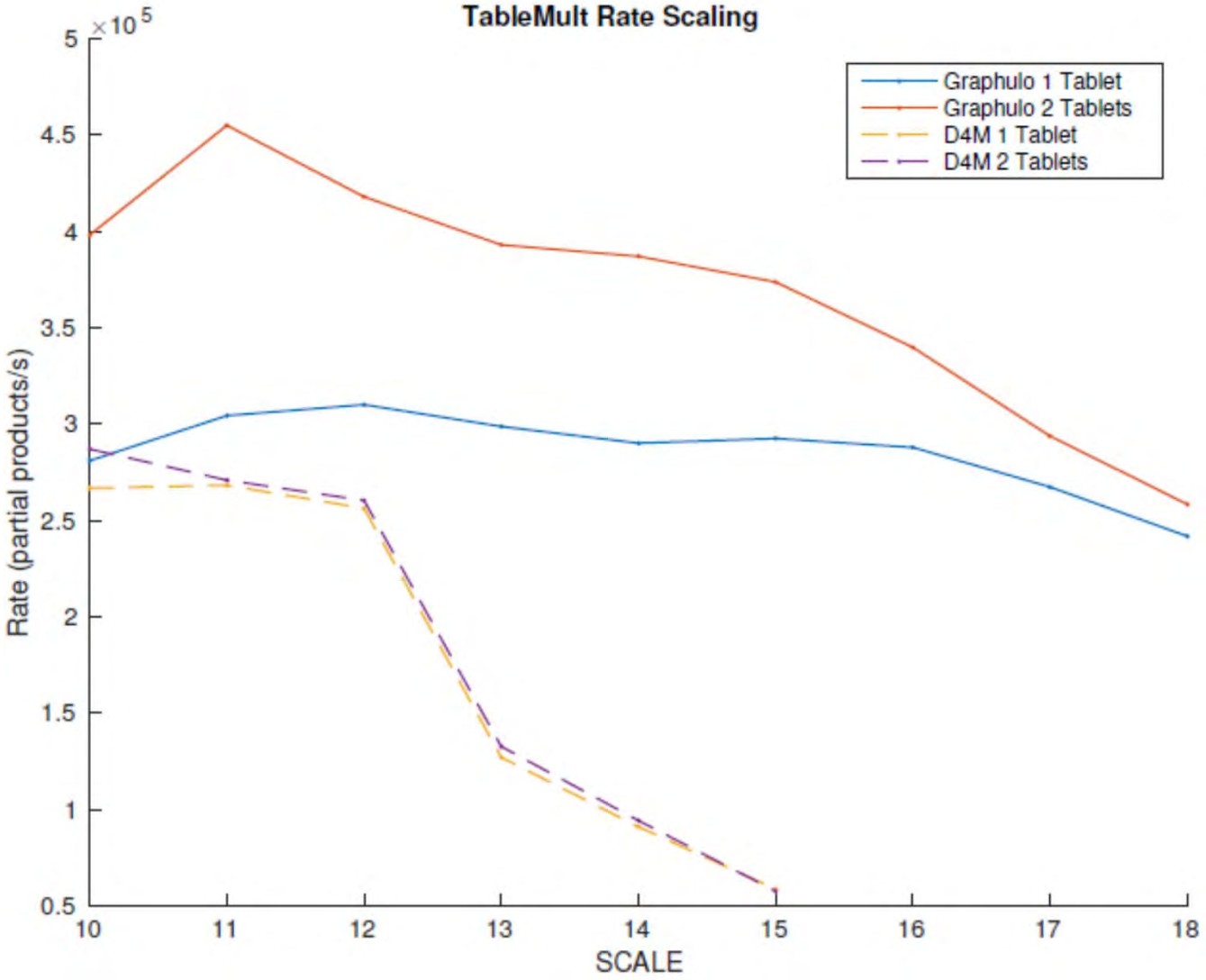
- **Inner product still has merit**
  - **Better for dense data**
  - **Hybrid 2D-like algorithm possible**



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^{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!