Graphulo: Native Linear Algebra in a NoSQL DB

Dylan Hutchison[1] Vijay Gadepally[2,3] Jeremy Kepner[2,3,4] Bill Howe[1]
[1]University of Washington [2]MIT Lincoln Laboratory [3]MIT CSAIL [4]MIT Math Department

http://graphulo.mit.edu

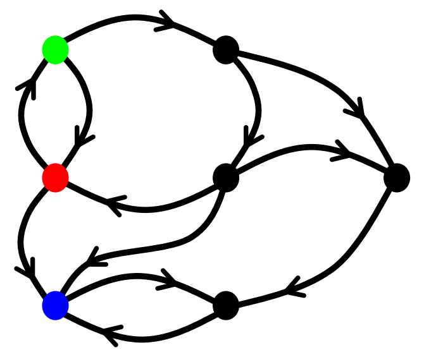
An open source library to orchestrate server-side graph processing in the Apache Accumulo database

- > **Problem:** to analyze graph and matrix data stored in Accumulo
- > **Non-solution:** always pull data from the DB before processing
  - ➢ MapReduce or an in-memory matrix library
- > **The Graphulo solution—** A tighter coupling: reuse Accumulo's native data access method, iterators, for query processing
  - ➢ Use Accumulo as a Big Index
  - ➢ Distribute with Accumulo's tablet servers
  - ➢ Generalize to BigTable NoSQL design

Data model suits Sparse Matrices:

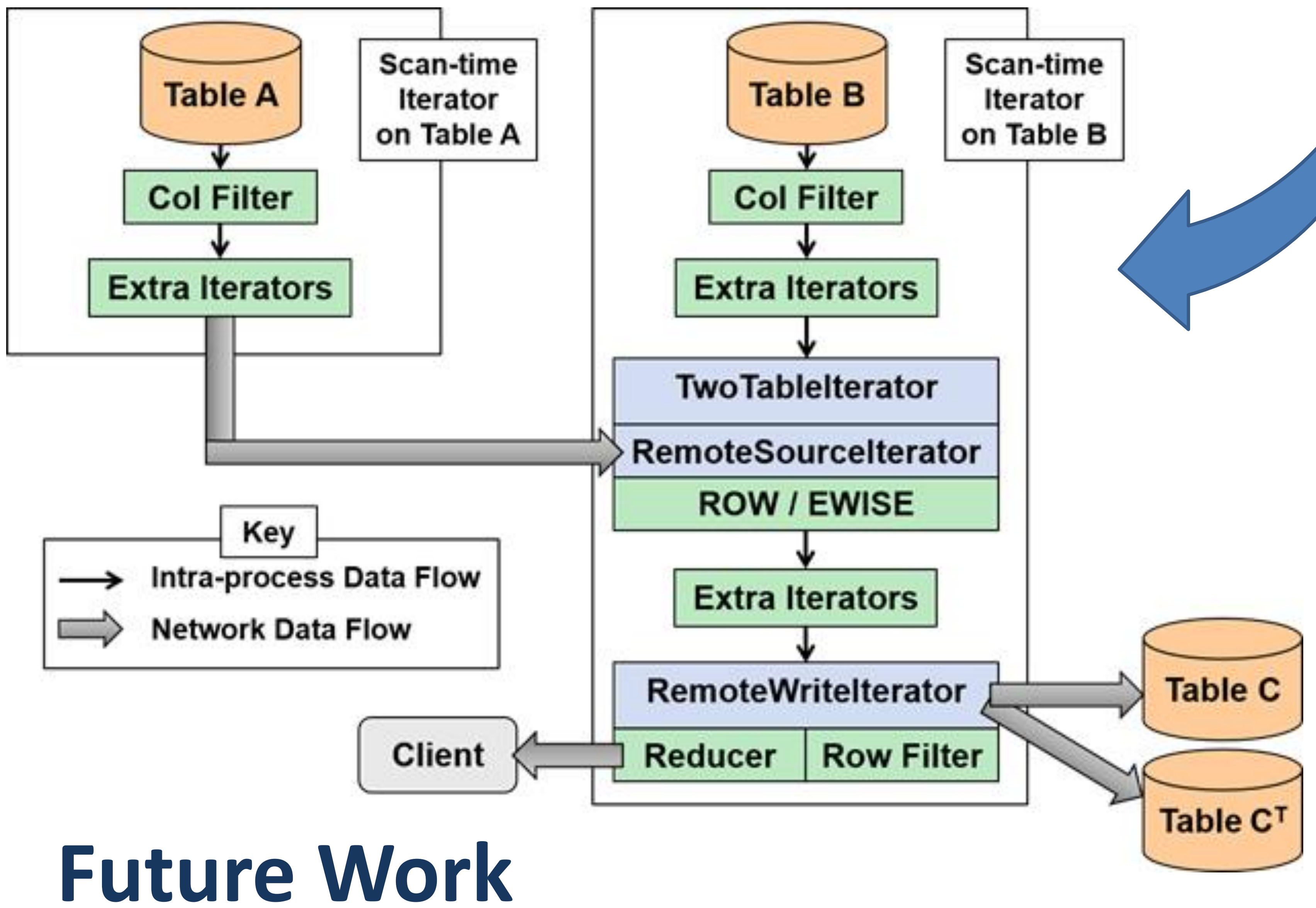
Key					Value
Row	Column			Timestamp	Value
	Family	Qualifier	Visibility		

GraphBLAS Matrix Math



GraphBLAS Kernel	Graphulo Implementation
BuildMatrix (⊕)	Accumulo BatchWriter
ExtracTuples	Accumulo BatchScanner
MxM (⊕, ⊗)	TwoTableIterator ROW mode, performing $\mathbf{A}^\top\mathbf{B}$
EwiseMult (⊗)	TwoTableIterator EWISE mode
EwiseAdd (⊕)	Similar to EwiseMult, with non-matching entries
Extract	Row and column filtering
Apply (f)	Extra Iterators
Assign	Apply with a key-transforming function
Reduce (⊕)	Reducer module on RemoteWriteIterator
Transpose	Transpose option on RemoteWriteIterator

Graphulo's TwoTable Iterator Stack



Future Work

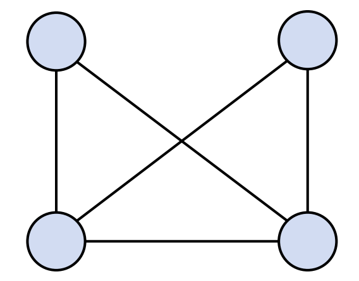
- ➢ More multi-node evaluation
- ➢ Expand to Relational Algebra
- ➢ Use an Optimizer to choose the best implementation

Reference

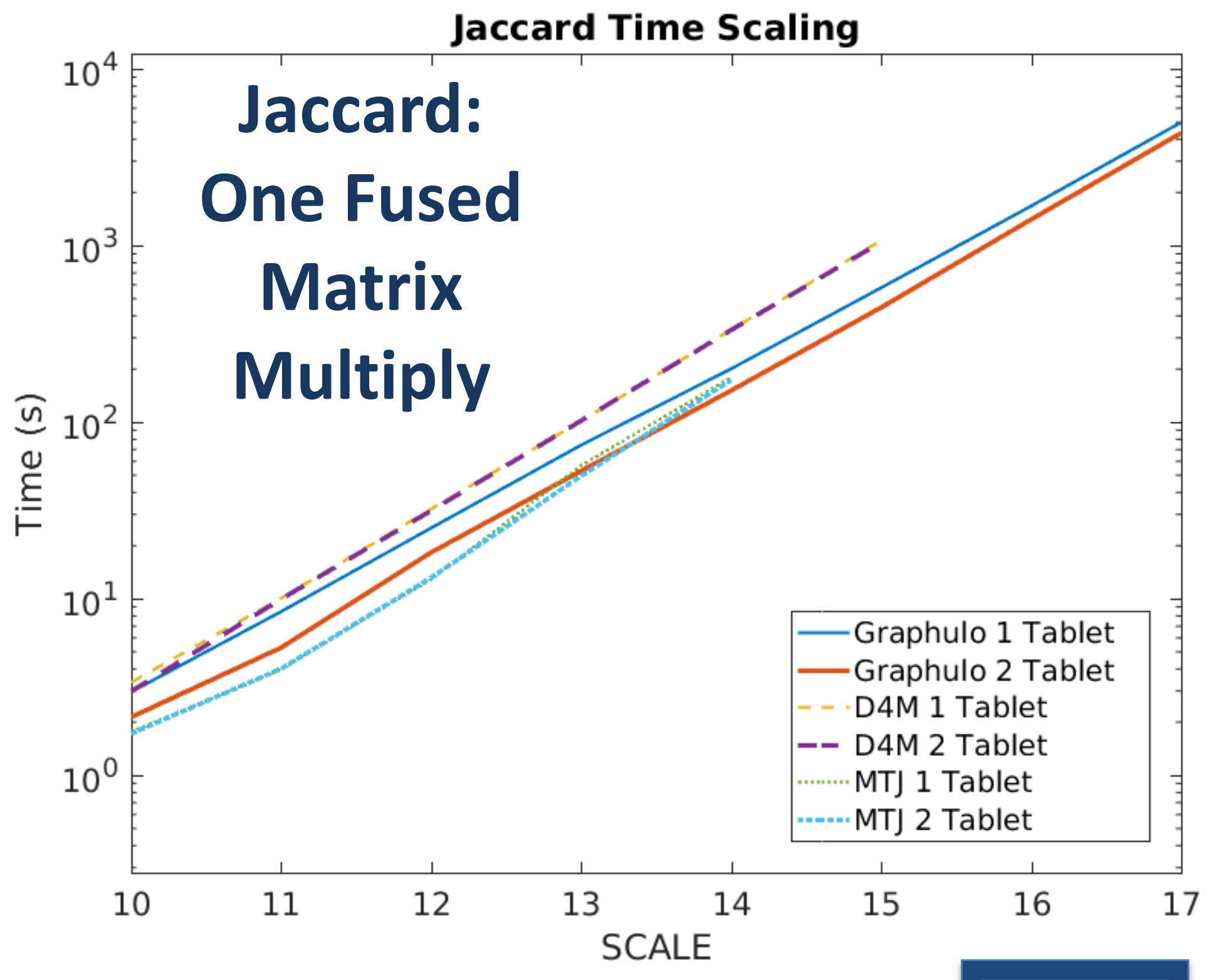
- > IPDPS '15
- > HPEC '15
- > HPEC '16 x2

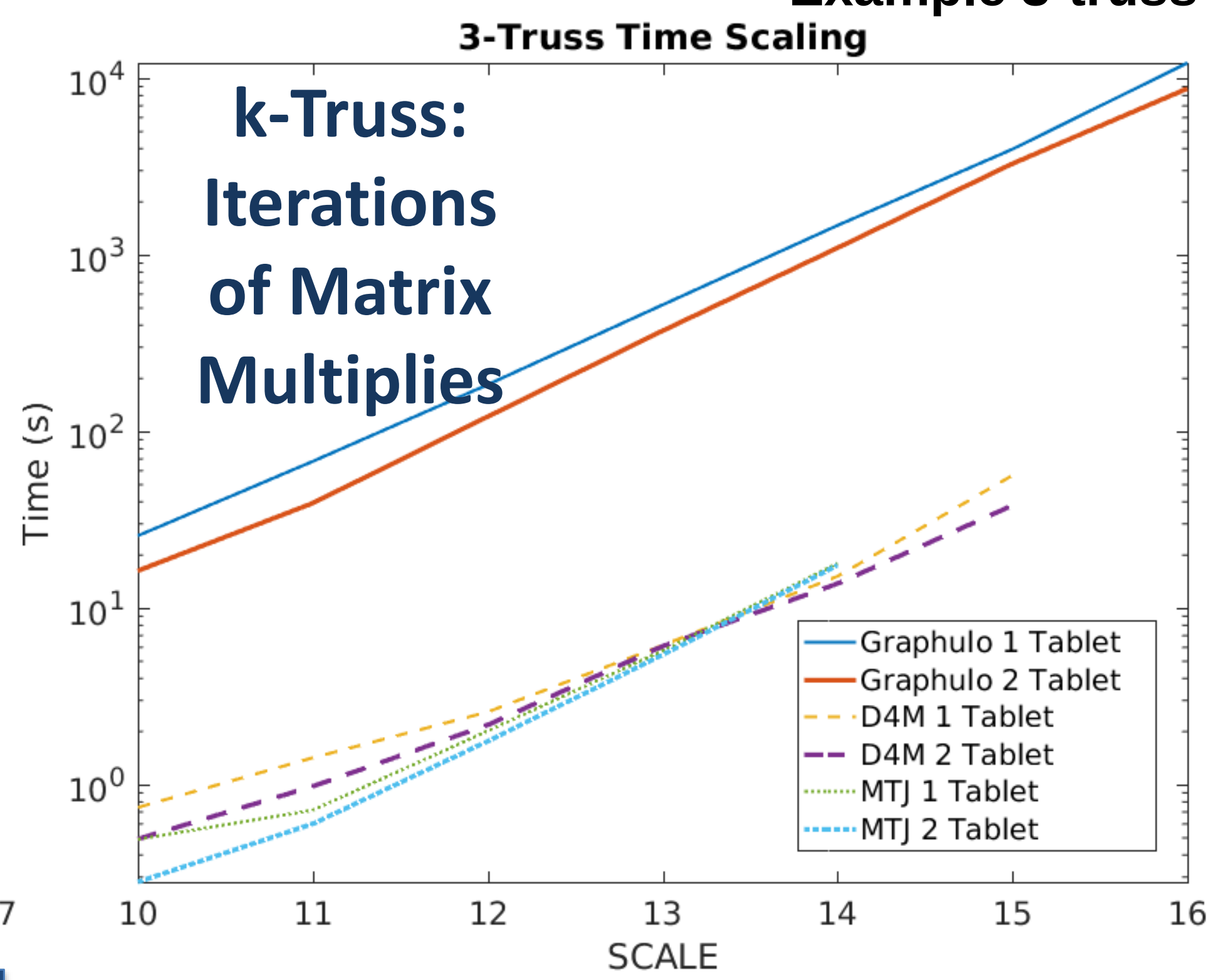
Performance Comparison

- ➢ D4M: Sparse Matrix Library for MATLAB
- ➢ MTJ: Dense Matrix Library for Java



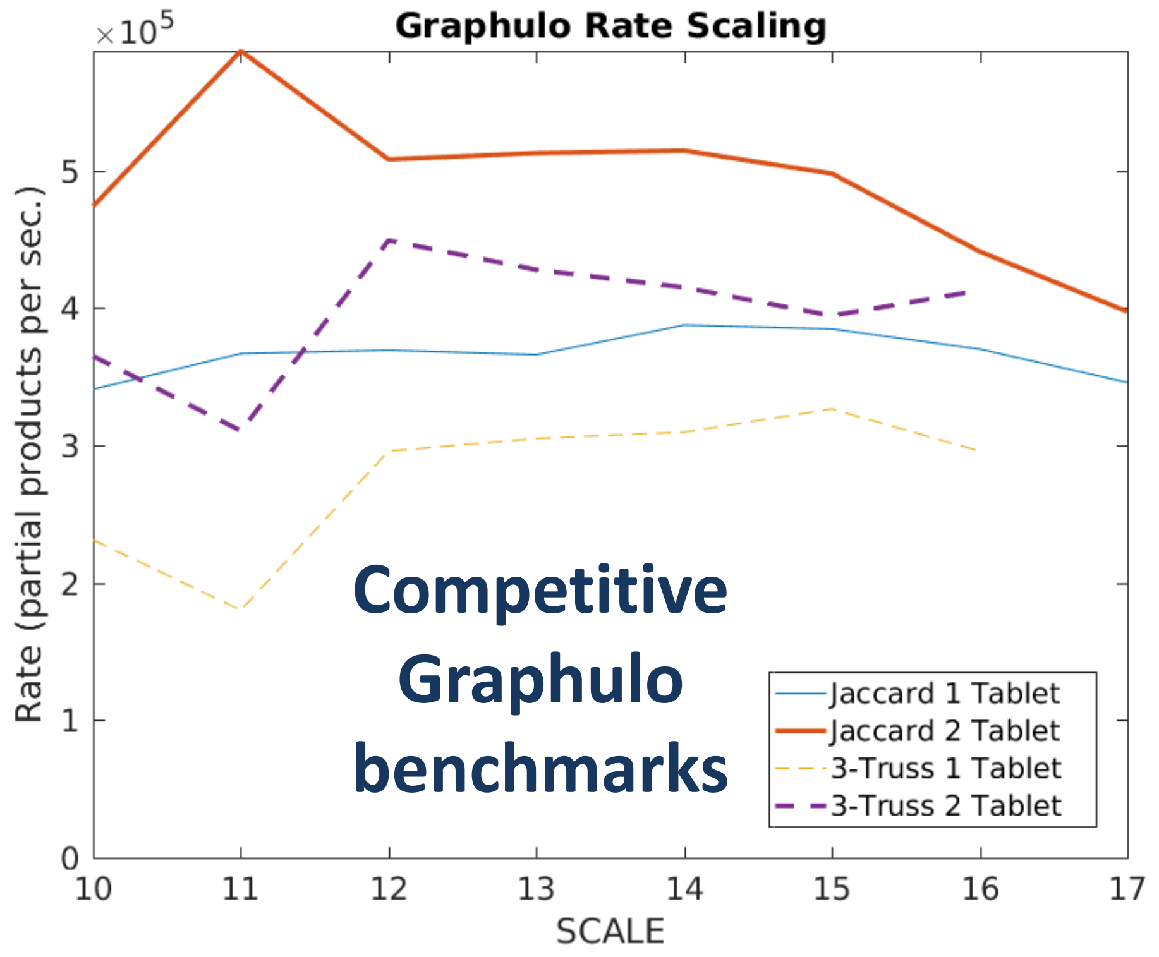
Example 3-truss





SCALE	nnz($\mathbf{A}$)	nnz(Jaccard($\mathbf{A}$))	Partial Products	Graphulo Overhead
10	2.10×10^4	2.15×10^5	1.01×10^6	4.7x
11	4.52×10^4	7.07×10^5	3.10×10^6	4.4x
12	9.67×10^4	2.18×10^6	9.29×10^6	4.3x
13	2.04×10^5	6.75×10^6	2.71×10^7	4.0x
14	4.26×10^5	2.02×10^7	7.77×10^7	3.8x
15	8.83×10^5	6.07×10^7	2.22×10^8	3.7x
16	1.82×10^6	1.77×10^8	6.20×10^8	3.5x
17	3.73×10^6	5.16×10^8	1.72×10^9	3.3x

SCALE	nnz($\mathbf{A}$)	nnz(3Truss($\mathbf{A}$))	Partial Products	Graphulo Overhead
10	2.10×10^4	2.03×10^4	5.94×10^6	293.3x
11	4.52×10^4	4.35×10^4	1.22×10^7	280.7x
12	9.67×10^4	9.20×10^4	5.45×10^7	592.7x
13	2.04×10^5	1.93×10^5	1.59×10^8	825.5x
14	4.26×10^5	3.99×10^5	4.55×10^8	1140.6x
15	8.83×10^5	8.20×10^5	1.30×10^9	1582.5x
16	1.82×10^6	1.67×10^6	3.62×10^9	2167.0x



Results

- > Jaccard coefficient algorithm is ideal for Graphulo
- > k-Truss subgraph algorithm is better in an external matrix library, assuming sufficient memory

Guideline

Use an in-DB solution when I/O is within an order of magnitude of alternative solutions