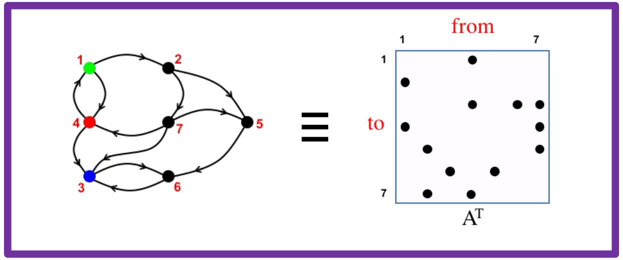



# Graphs (GraphBLAS) and storage (TileDB) as Sparse Linear algebra

http://graphblas.org

**Tim Mattson**

**Intel Labs**

# Notices and Disclaimers

Forward-Looking Statements. Statements in this presentation that refer to business outlook, plans, and expectations are forward-looking statements that involve risks and uncertainties. Words such as "anticipate," "expect," "intend," "goal," "plans," "believe," "seek," "estimate," "continue,“ “committed,” “on-track,” ”positioned,” “ramp,” “momentum,” “roadmap,” “path,” “pipeline,” “progress,” “schedule,” “forecast,” “likely,” “guide,” “potential,” “next gen,” “future,” "may," "will," “would,” "should," “could,” and variations of such words and similar expressions are intended to identify such forward-looking statements. Statements that refer to or are based on estimates, forecasts, projections, uncertain events or assumptions, including statements relating to Intel’s strategy and its anticipated benefits; business plans; financial projections and expectations; total addressable market (TAM) and market opportunity; manufacturing expansion and investment plans; future manufacturing capacity; future products, technology, and services, and the expected availability and benefits of such products, technology, and services, including product and manufacturing plans, goals, timelines, ramps, progress, and future product and process leadership and performance; future economic conditions; future impacts of the COVID-19 pandemic; plans and goals related to Intel’s foundry business; future legislation; future capital offsets; pending or future transactions; the proposed Mobileye IPO; supply expectations including regarding industry shortages; future external foundry usage; future use of EUV and other manufacturing technologies; expectations regarding customers, including designs, wins, orders, and partnerships; projections regarding competitors; ESG goals; and anticipated trends in our businesses or the markets relevant to them, including future demand, market share, industry growth, and technology trends, also identify forward-looking statements. Such statements involve many risks and uncertainties that could cause actual results to differ materially from those expressed or implied in these forward-looking statements. Important factors that could cause actual results to differ materially are set forth in Intel's earnings release dated January 26, 2022, which is included as an exhibit to Intel’s Form 8-K furnished to the SEC on such date, and in Intel's SEC filings, including the company's most recent reports on Forms 10-K and 10-Q. Copies of Intel’s SEC filings may be obtained by visiting our Investor Relations website at www.intc.com or the SEC's website at www.sec.gov. All information in this presentation reflects management’s views as of April 20, unless an earlier date is indicated. Intel does not undertake, and expressly disclaims any duty, to update any statement made in this presentation, whether as a result of new information, new developments or otherwise, except to the extent that disclosure may be required by law.

Intel technologies may require enabled hardware, software or service activation. No product or component can be absolutely secure. Your costs and results may vary. Product and process performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex and www.Intel.com/ProcessInnovation. Future product and process performance and other metrics are projections and are inherently uncertain.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

# Human Readable Disclaimer

- The views expressed in this talk are those of the speaker and not his employer.

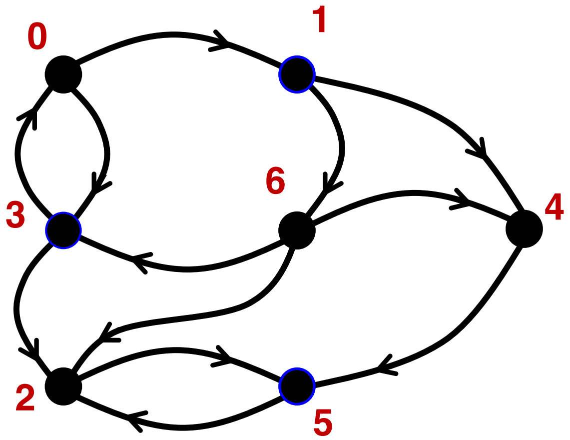
- This is a speculative, academic style talk … I am not describing or even suggesting <u>ANYTHING</u> about future products from Intel!!!

I work in Intel's research labs. I don't build products. Instead, I get to poke into dark corners and think silly thoughts… just to make sure we don't miss any great ideas.

I have a really GREAT Job!!!!

# A graph as a matrix

- Adjacency Matrix: A square matrix (usually sparse) where rows and columns are labeled by vertices and non-empty values are edges from a row vertex to a column vertex




To vertex (columns)

A =

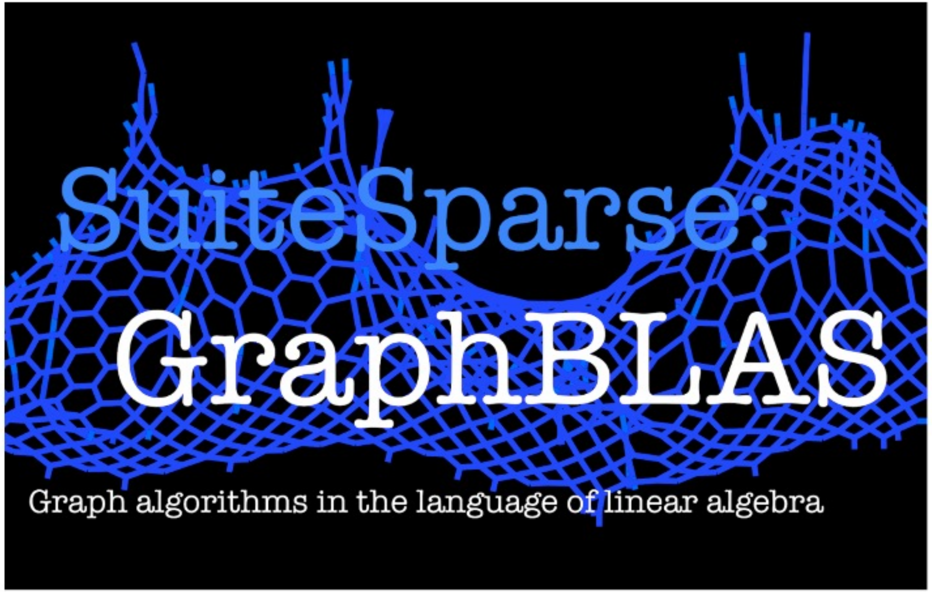
From vertex (rows)

| | | | | | | |
|---|---|---|---|---|---|---|
| - | ★ | - | ★ | - | - | - |
| - | - | - | - | ★ | - | ★ |
| - | - | - | - | - | ★ | - |
| ★ | - | ★ | - | - | - | - |
| - | - | - | - | - | ★ | - |
| - | - | ★ | - | - | - | - |
| - | - | ★ | ★ | ★ | - | - |

By using a matrix, I can turn graph algorithms into linear algebra.

**GraphBLAS Math is a lot of fun, but without a software ecosystem the impact from all this cool math is negligible.**

# The Foundation of our GraphBLAS ecosystem: SuiteSparse … C libraries for GraphBLAS and LAGraph




Tim Davis
Texas A&M
University


- Open-Source C library (Apache 2.0) conforms to the v2.0 C GraphBLAS specification.
- High performance, internal parallelism (OpenMP) for easy-to-code, fast Graph Algorithms
- Support from NSF, MIT Lincoln Labs, Intel, Nvidia, IBM, MathWorks, Redis Labs, and Julia Computing

https://people.engr.tamu.edu/davis/GraphBLAS.html

# GraphBLAS Implementations

SuiteSparse library (Texas A&M): First fully conforming GraphBLAS release

- http://faculty.cse.tamu.edu/davis/suitesparse.html

GraphBLAS C (IBM): the second fully conforming release

- https://github.com/IBM/ibmgraphblas

GBTL: GraphBLAS Template Library (CMU/SEI/IU/PNNL): GraphBLAS C++ implementation

- https://github.com/cmu-sei/gbtl

GraphBLAST: A C++ implementation for GraphBLAS for GPUs (UC Davis)

- https://github.com/gunrock/graphblast

Python bindings:

- PyGB: A python wrapper around GBTL (UW/PNNL/CMU)
  - https://github.com/jessecoleman/gbtl-python-binding
- pygraphblas: A python wrapper around SuiteSparse GraphBLAS
  - https://github.com/michelp/pygraphblas
- Python-graphblas: Anaconda's python wrapper around SuiteSparse GraphBLAS
  - https://github.com/python-graphblas/python-graphblas

pggraphblas: A PostgreSQL wrapper around SuiteSparse GraphBLAS

- https://github.com/michelp/pggraphblas

Julia wrapper around SuiteSparse

- SuiteSparseGraphBLAS.jl

Matlab and Julia wrappers around SuiteSparse GraphBLAS

- https://aldenmath.com

Implementations in progress:

- Intel and SEI/CMU are working on a C++ implementation. We will have a preliminary release running on clusters of CPUs, GPUs, and multiple CPUs
- And soon Intel will have a Go implementation (wrapping SuiteSparse)

Third Party names are the property of their owners

# Multilanguage support by wrapping SuiteSparse GraphBLAS

**The GraphBLAS in Julia and Python: the PageRank and Triangle Centralities** (HPEC'21)

Michel Pelletier, Will Kimmerer, Timothy A. Davis, Timothy G. Mattson

$$\mathbf{c} = \frac{(3\mathbf{A} - 2\check{\mathbf{T}} + \mathbf{I})\mathbf{T}\mathbf{1}}{\mathbf{1}^T\mathbf{T}\mathbf{1}}$$

- $\mathbf{T}$ matrix of triangles of A
- $\check{\mathbf{T}}$ all zero, but 1 where T is non-zero
- $\mathbf{1}$ Matrix of all ones; $\mathbf{I}$ identity matrix

**The math**

**Julia**

```julia
function triangle_centrality1(A)
    T = mul(A, A', mask=A, desc=S)
    y = reduce(+, T, dims=2)
    k = reduce(+, y)
    return (3.*mul(A,y)-2.*mul(one.(T),y).+y) ./ k
end
```

**pygraphblas**

```python
def triangle_centrality1(A):
    T = A.mxm(A, mask=A, desc=ST1)
    y = T.reduce_vector()
    k = y.reduce_float()
    return (3 * (A@y) - 2 * (T.one()@y) + y) / k
```

# Multilanguage support by wrapping SuiteSparse GraphBLAS

The GraphBLAS in Julia and Python:
the PageRank and Triangle Centralities (HPEC'21)

Michel Pelletier Will Kimmerer Timothy A. Davis Timothy G. Mattson

$$\mathbf{c} = \frac{(3\mathbf{A} - 2\check{\mathbf{T}} + \mathbf{I})\mathbf{T}\mathbf{1}}{\mathbf{1}^T\mathbf{T}\mathbf{1}}$$

$\mathbf{T}$ matrix of triangles of A
$\check{\mathbf{T}}$ all zero, but 1 where T is non-zero
$\mathbf{1}$ Matrix of all one

**The math**

Expressivity/productivity in programming languages should be measured by how clear "the math" maps onto code.

These interfaces are highly productive by that measure

**Julia**

```julia
function triangle_centrality1(A)
    T = mul(A, A', mask=A, desc=S)
    y = reduce(+, T, dims=2)
    k = reduce(+, y)
    return (3.*mul(A,y)-2.*mul(one.(T),y).+y) ./ k
end
```

**pygraphblas**

```python
def triangle_centrality1(A):
    T = A.mxm(A, mask=A, desc=ST1)
    y = T.reduce_vector()
    k = y.reduce_float()
    return (3 * (A@y) - 2 * (T.one()@y) + y) / k
```

Third Party names are the property of their owners

# LAGraph: A curated collection of high level Graph Algorithms

Graph Algorithms built on top of the GraphBLAS.

## LAGraph: A Community Effort to Collect Graph Algorithms Built on Top of the GraphBLAS

Tim Mattson[‡], Timothy A. Davis[◦], Manoj Kumar[¶], Aydın Buluç[†], Scott McMillan[§], José Moreira[¶], Carl Yang[*,†]

[‡]*Intel Corporation* [†]*Computational Research Division, Lawrence Berkeley National Laboratory*
[◦]*Texas A&M University* [¶]*IBM Corporation* [§]*Software Engineering Institute, Carnegie Mellon University*
[*]*Electrical and Computer Engineering Department, University of California, Davis*

GrAPL 2019

Official release of LAGraph library v1.0 late 2021

Third Party names are the property of their owners

# Integration of GraphBLAS with NetworkX

Jim Kitchen (Anaconda) and Erik Welch (Nvida)

```python
import networkx as nx

G = nx.erdos_renyi_graph(8000, 0.02)

k = nx.k_truss(G, 5)
```

8000 nodes, ~ 640_000 edges

This takes 10.7 seconds

*The k-truss is the maximal induced subgraph of G with each edge belonging to at least k-2 triangles.*

```python
import networkx as nx
import graphblas_algorithms as ga

G = nx.erdos_renyi_graph(8000, 0.02)
G2 = ga.Graph.from_networkx(G)

k = nx.k_truss(G2, 5)
```

**conda install -c conda-forge graphblas-algorithms**

-or-

**pip install graphblas-algorithms (Linux Only)**

This takes 0.5 seconds

This takes 0.28 seconds

# Benchmarks: GraphBLAS vs NetworkX

**Hardware:** NVIDIA DGX-1
**CPU:** Dual 20 Core Intel Xeon E5-2698 v4 2.2GHz
**RAM:** 512 GB 2133 MHz DDR4 RDIMM

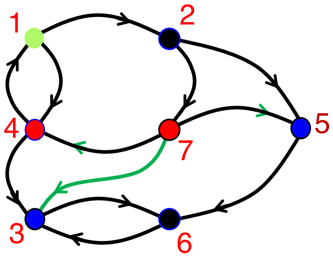
## Speed-up

| | amazon | google | pokec | enron | preferentialAttachment | caidaRouterLevel | dblp | citationCiteseer | coAuthorsDBLP | as-Skitter | coPapersCiteseer | coPapersDBLP | Network X run times |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # of vertices | 262,111 | 916,428 | 1,632,804 | 36,692 | 100,000 | 192,244 | 326,186 | 268,495 | 299,067 | 1,696,415 | 434,102 | 5,404,486 | |
| # of edges | 1,234,877 | 5,105,039 | 30,622,564 | 367,662 | 999,970 | 1,218,132 | 1,615,400 | 2,313,294 | 1,955,352 | 22,190,596 | 32,071,440 | 30,491,458 | |
| degree centrality | 32 | 48 | 31 | 29 | 60 | 140 | 65 | 180 | 200 | 530 | 190 | 220 | 0.25-1 s |
| reciprocity | 290 | 370 | 470 | 230 | 600 | 840 | 1600 | 1000 | 1400 | 1700 | 2200 | 2200 | 3-5 min |
| generalized degree | N/A (Requires Undirected Graph) | | | 140 | 160 | 190 | 150 | 220 | 150 | 1700 | 500 | 360 | 10-30 min |
| k-truss(k=5) | | | | 53 | 800 | 140 | 130 | 150 | 170 | 350 | 2000 | 1100 | 30-100 min |
| pagerank | 130 | 340 | 930 | 50 | 240 | 250 | 390 | 580 | 810 | 1800 | 3900 | 4200 | 1 min |
| eigenvector centrality | 53 | 120 | 150 | 61 | 650 | 740 | 1300 | 1100 | 1300 | 2000 | 5200 | 5300 | 30-100 min |
| katz centrality | 420 | 530 | 830 | 300 | 1100 | 1400 | 1700 | 2100 | 2300 | 3400 | 7500 | 7600 | hours-days |
| clustering | 160 | 900 | 620 | 370 | 370 | 290 | 280 | 540 | 380 | 11000 | 2600 | 2100 | 10-30 min |
| transitivity | 180 | 270 | 440 | 830 | 970 | 900 | 730 | 1600 | 970 | 20000 | 6600 | 5000 | 10-30 min |
| square clustering | N/A | | | 1200 | 950 | 1400 | 1800 | 1100 | 1300 | DNF | DNF | 21000 | days-weeks? |
| pagerank (scipy) | 3.4 | 14 | 23 | 2.1 | 3.3 | 3.8 | 6.3 | 9.8 | 11 | 20 | 23 | 27 | 0.25-1 s |

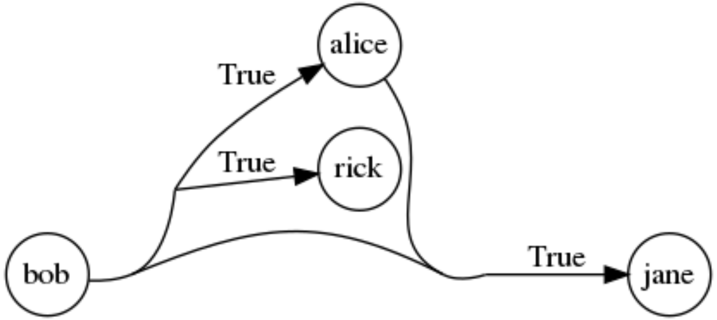
# Moving Beyond Simple Graphs

# Different Types of Graphs

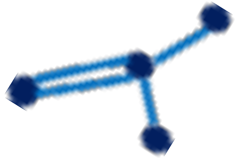
- Simple Graphs: an edge connects one source to one destination.




- Hypergraphs: at least one edge from one source to multiple destinations.

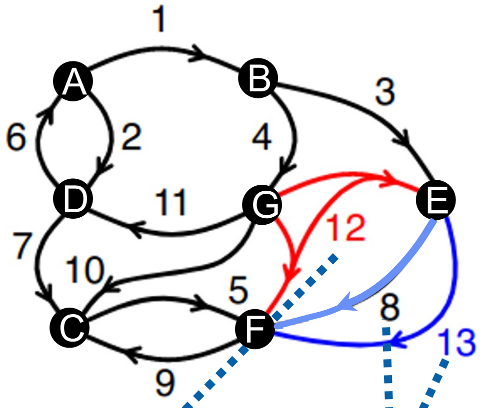



- Multigraphs: Includes edges that share end points.



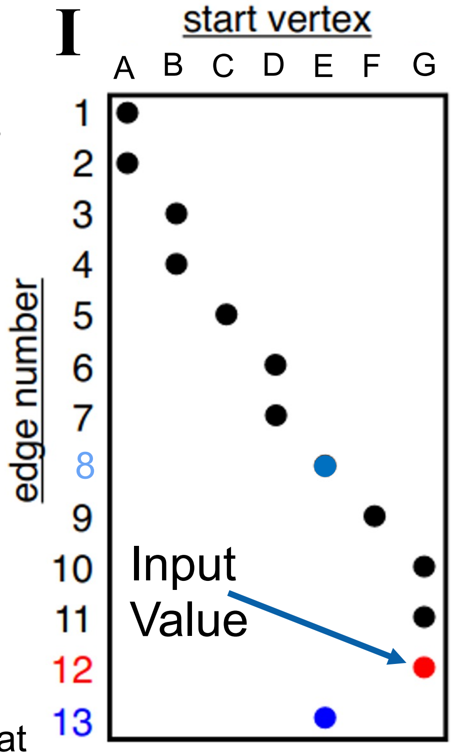
# All these graph types can be handled with the GraphBLAS

Two **Incidence Matrices** can represent a wide range of graphs

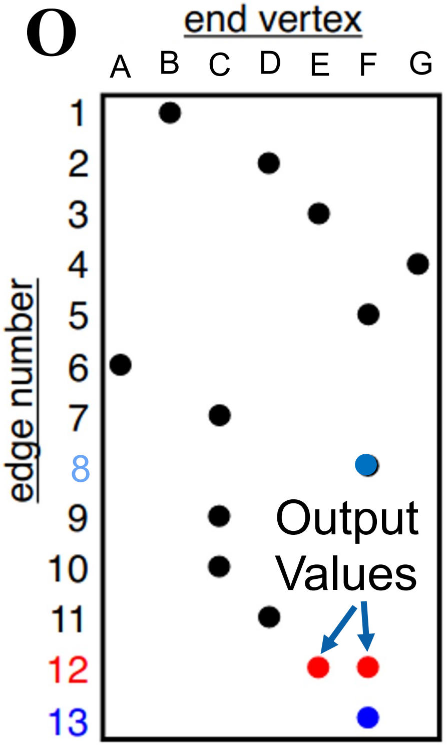



**Hypergraph**: Multiple Outputs from edge 12

**Multigraph**: Multiple edges that share end points




**UINT64**




**UINT64**

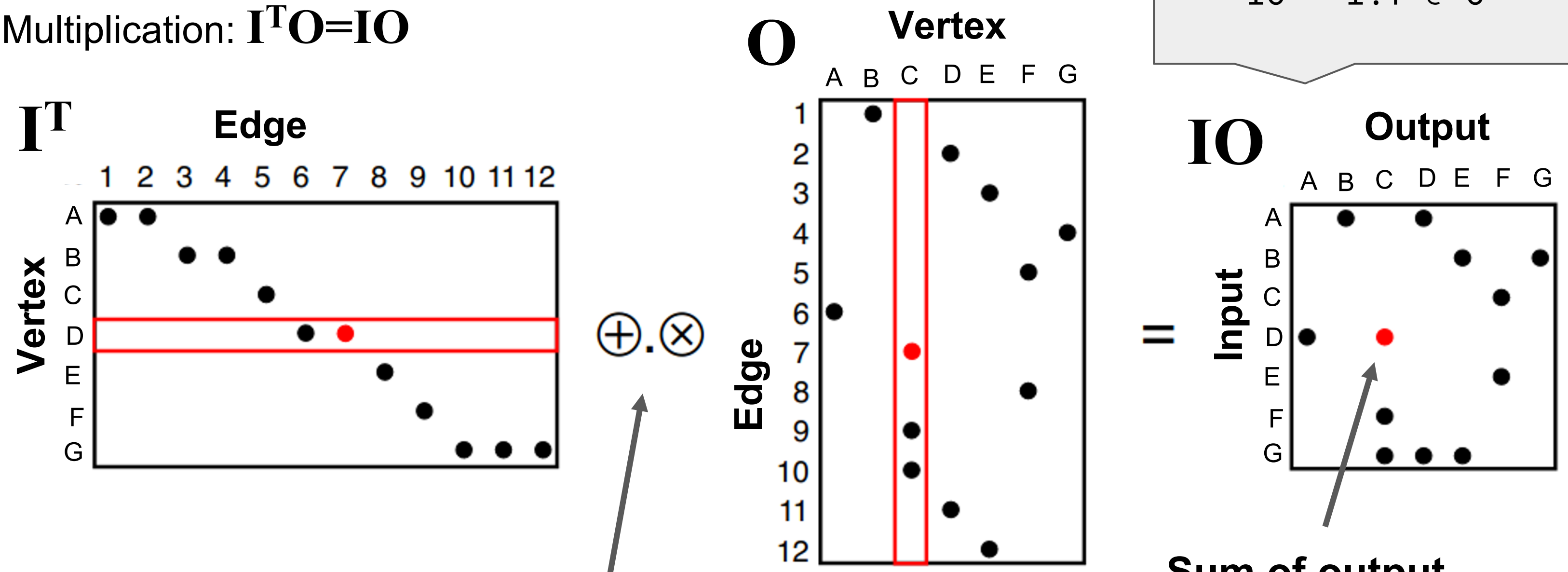


Graph/Array source: Jananthan, Dibert, Kepner, Constructing Adjacency Arrays from Incidence Arrays, GABB'2017

# Adjacency matrices from incidence matrices

Adjacency can be **projected** from two Incidence Matrices with Matrix Multiplication: $\mathbf{I}^{\mathbf{T}}\mathbf{O}=\mathbf{IO}$

```python
#!python
with PLUS_SECOND:
    IO = I.T @ O
```



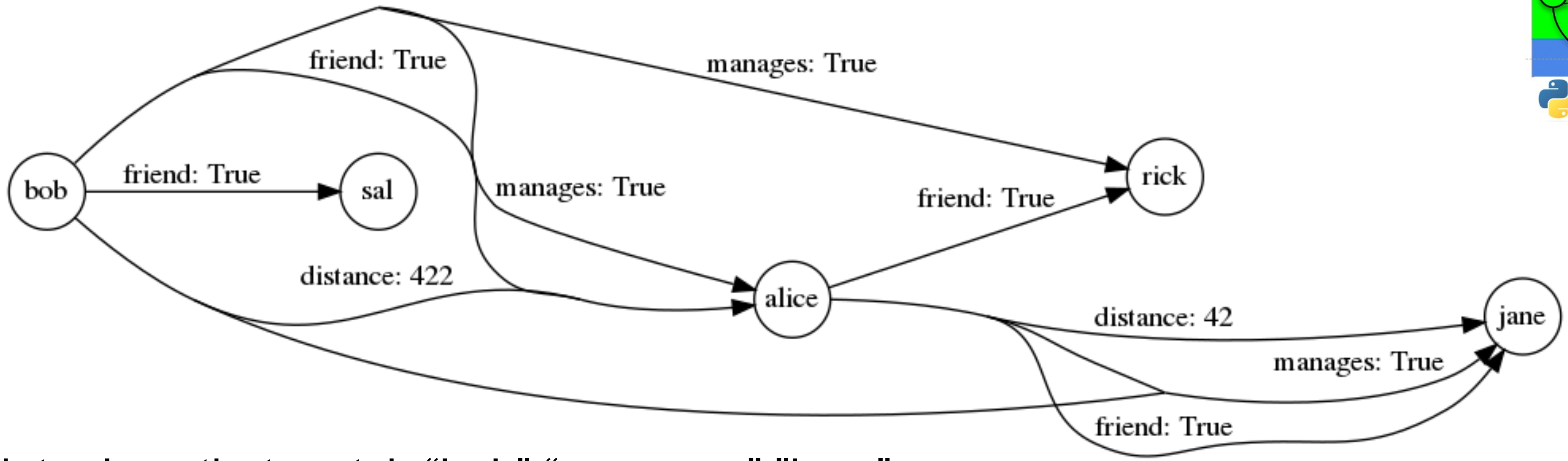

**PLUS_SECOND** Semiring

**Sum of output from ($I_D$ → $O_C$)**



Graph/Array source : Jananthan, Dibert, Kepner, Constructing Adjacency Arrays from Incidence Arrays, GABB’2017

# Graphony: Queries over property graphs




Print edges that match “bob” “manages” ”jane”:

```
>>> p(G(source='bob', property='manages', destination='jane'))
[manages((bob, alice), (jane), (True))]
```

Michel Pelletier

Source: https://github.com/Graphegon/Graphony

# Speaking of property graphs …

- A graph database built on top of GrapghBLAS … one of our major, commercial success stories for the GraphBLAS

- Supports a subset of the Cypher query language … mapping elements of the language onto linear algebra operations.

# RedisGraph

A Graph database built on Redis

chat 480 online | repository

RedisGraph is a graph database built on Redis. This graph database uses GraphBlas under the hood for its sparse adjacency matrix graph representation.

## Primary features

- Based on the property graph model
- Nodes can have any number of labels
- Relationships have a relationship type
- Graphs represented as sparse adjacency matrices
- Cypher as the query language
- Cypher queries translate into linear algebra expressions

Third Party names are the property of their owners

https://redis.io/docs/stack/graph/

**The lesson from Edgar Codd so long ago was the power of an algebra to unify disparate approaches to a problem.**

**Relational algebras are great at data management, but they suck at computation. It would be stupid to build a PDE solver around a relational algebra.**

**So if we want "one algebra to rule them all", what should be our algebra?**

PDE: Partial Differential Equation

# Linear Algebra: One Algebra to rule them all

- Computational physics is basically applied linear algebra
  - We create differential equations from the physics, discretize domains to replace derivatives with differences, and solve resulting algebraic equations.
  - Since the differential operators are replaced by modest sized stencils, the arrays in physics problems are sparse (with a small number of exceptions such as in ab initio quantum chemistry).

- Graphs are linear algebra, databases map onto linear algebra, science and engineering is linear algebra … if you go deep enough, in almost any field, you end up doing linear algebra.

- All we need is a good library for Sparse Linear Algebra.

**… to address data management, we need a storage engine to work with GraphBLAS**

**Hardware:** NVIDIA DGX-1
**CPU:** Dual 20 Core Intel Xeon E5-2698 v4 2.2GHz
**RAM:** 512 GB 2133 MHz DDR4 RDIMM

# SuiteSparse to the rescue:

SuiteSparse versus the Intel MKL sparse library

| computation | format | MKL method | MKL time (sec) 1st | MKL time (sec) 2nd | SuiteSparse time (sec) | speedup 1st | speedup 2nd |
|---|---|---|---|---|---|---|---|
| y+=S*x | S by row | mkl_sparse_d_mv | 2.54 | 1.27 | 1.21 | 2.10 | 1.05 |
| y+=S*x | S by col | mkl_sparse_d_mv | 7.22 | 7.22 | 1.98 | 3.65 | 3.65 |
| C+=S*F | S by row, F by row | mkl_sparse_d_mm | 2.95 | 1.90 | 1.98 | 1.49 | **.96** |
| C+=S*F | S by row, F by col | mkl_sparse_d_mm | 6.12 | 4.99 | 1.48 | 4.13 | 3.37 |
| C+=S*F | S by col, F by row | mkl_sparse_d_mm | 28.82 | 28.82 | 13.78 | 2.09 | 2.09 |
| C+=S*F | S by col, F by col | mkl_sparse_d_mm | 78.82 | 5.17 | 9.38 | 8.40 | **.55** |
| C=S+B | S by row | mkl_sparse_d_add | 30.77 | 30.77 | 1.44 | 21.37 | 21.37 |
| C=S'+B | S by row | mkl_sparse_d_add | 102.09 | 27.30 | 16.29 | 6.26 | 1.67 |
| C=S' | S by row | mkl_sparse_convert_csr | 77.27 | 77.27 | 14.80 | 5.22 | 5.22 |

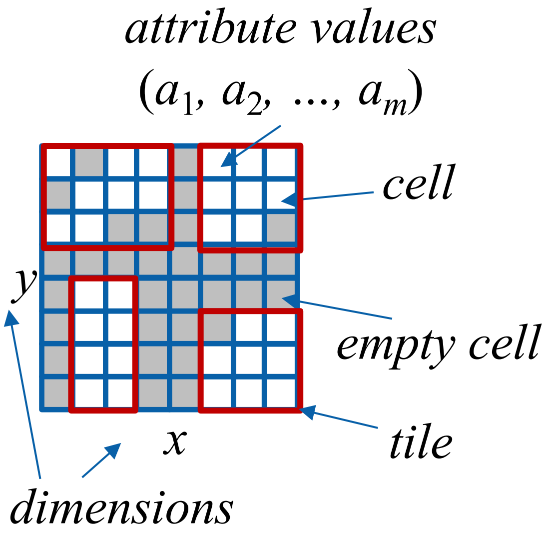
Table 4. SuiteSparse vs MKL 2022 with the GAP-Twitter matrix

SuiteSparse GraphBLAS traditional sparse linear algebra as well as graphs.

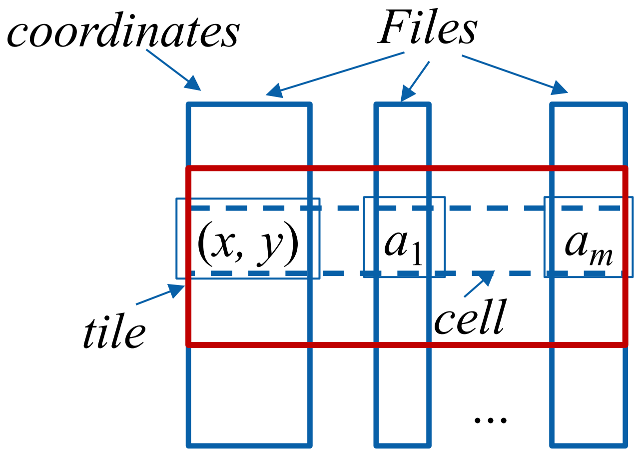
All we need is a good storage engine for an end-to-end solution

# TileDB an data storage manager:  Optimized for Sparse Arrays

## Logical representation




## Physical representation




**Tile:** *Atomic unit of processing*

Manage array storage as tiles of different shape/size in the index space, but with ~equal number of non-empty cells

TileDB Inc website:   https://tiledb.io

The TileDB Array Data Storage Manager, Stavros Papadopoulos, Kushal Datta, Samuel Madden, Tim Mattson, VLDB 2017

Third Party names are the property of their owners

# Conclusion

- SuiteSparse GraphBLAS + TileDB as a storage engine is the foundation of an end-to-end framework for data analytics.
- All that's missing is a query engine supporting GQL that maps onto GraphBLAS
- I am looking for collaborators to implement the above (interface GraphBLAS to TileDB and combine with a GQL query engine). This would be fun and impactful. Let me know if you want to get involved.

Third Party names are the property of their owners 