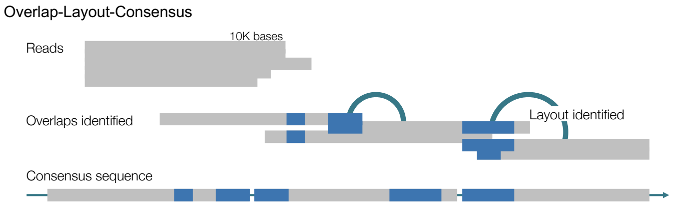
Parallel Sparse Matrix Algorithms for Data Analysis and Machine Learning

Aydın Buluç
Computational Research Division, LBNL
EECS Department, UC Berkeley

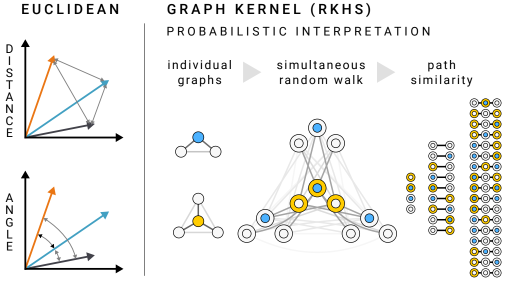
@SPCL_Bcast, March 24, 2022

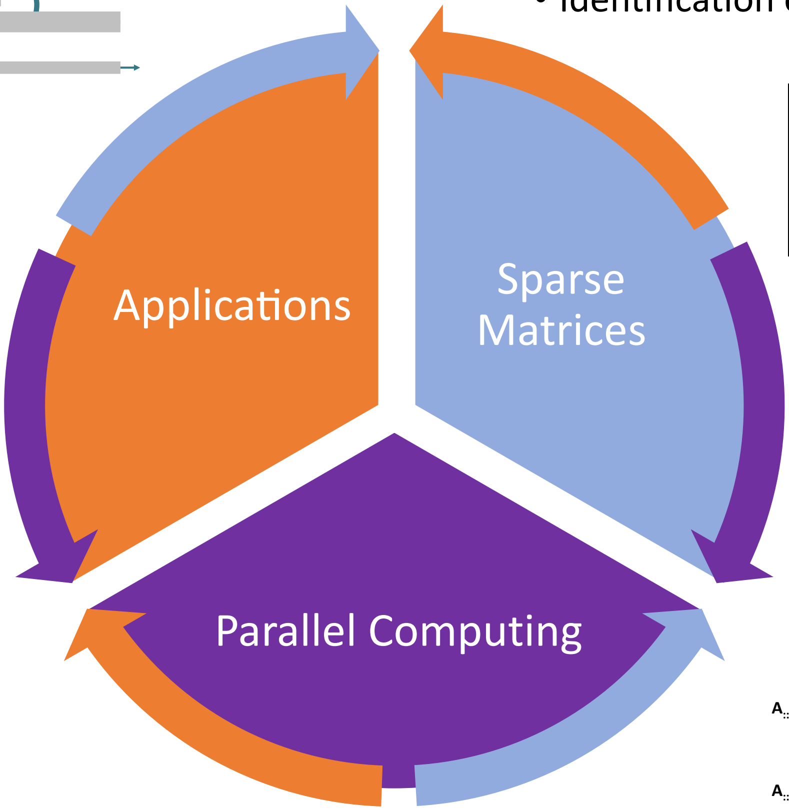
PASSION Lab Research Agenda

http://passion.lbl.gov



- Genomics
- Graph analysis
- Proteomics
- Machine learning



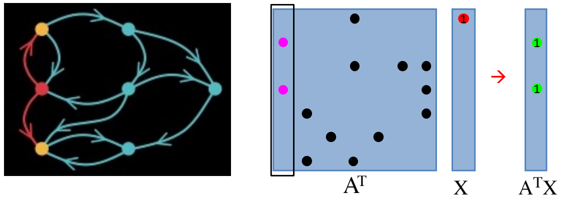


Applications

Sparse Matrices

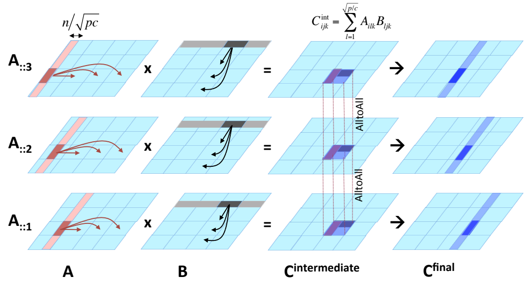
Parallel Computing

- New sparse data structures and algorithms
- Identification of computational primitives



GraphBLAS: graphs in the language of linear algebra
http://graphblas.org

Communication-avoiding algorithms for sparse matrices



- Parallel data structures
- Parallel programming
- Communication bounds

PASSION Lab People

Aydın Buluç (Principal Investigator)

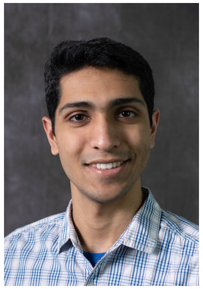
- Staff Scientist, AMCRD, Lawrence Berkeley National Laboratory
- Adjunct Assistant Professor, EECS Department (CS division), UC Berkeley

UC Berkeley PhD Students (co-advised)

Ben Brock

Giulia Guidi

Alok Tripathy



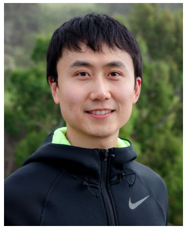
Vivek Bharadwaj

Undergraduate researchers:

- Richard Lettich
- Ujjaini Mukhopadhyay

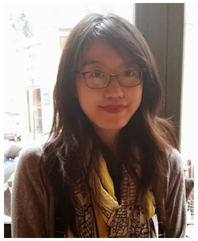
LBNL Research Scientists, Engineers, Postdocs, Visiting Fellows

Oguz Selvitopi



Yu-Hang Tang

Can Kizilkale



Helen Xu

Gabriel Raulet

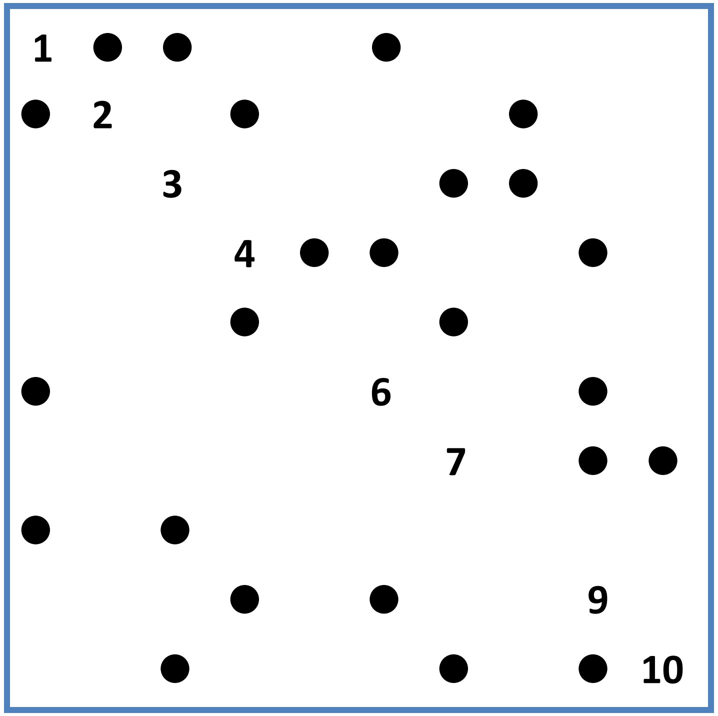
Koby Hayashi

Sparse Matrices

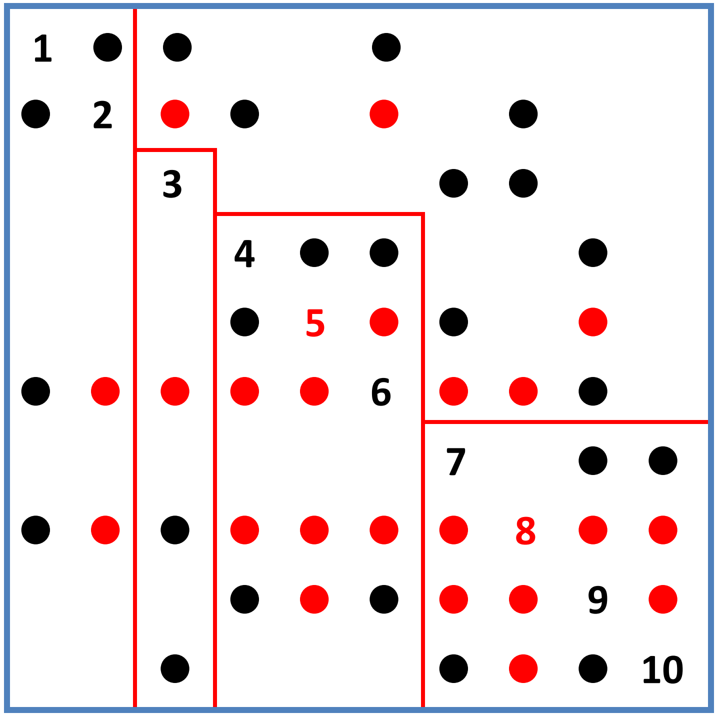
"I observed that most of the coefficients in our matrices were zero; i.e., the nonzeros were 'sparse' in the matrix, and that typically the triangular matrices associated with the forward and back solution provided by Gaussian elimination would remain sparse if pivot elements were chosen with care"

- Harry Markowitz, describing the 1950s work on portfolio theory that won the 1990 Nobel Prize for Economics

Sparse Matrices in Simulations



Original matrix A



Factors L+U

Original: Ax = b (hard to solve directly)

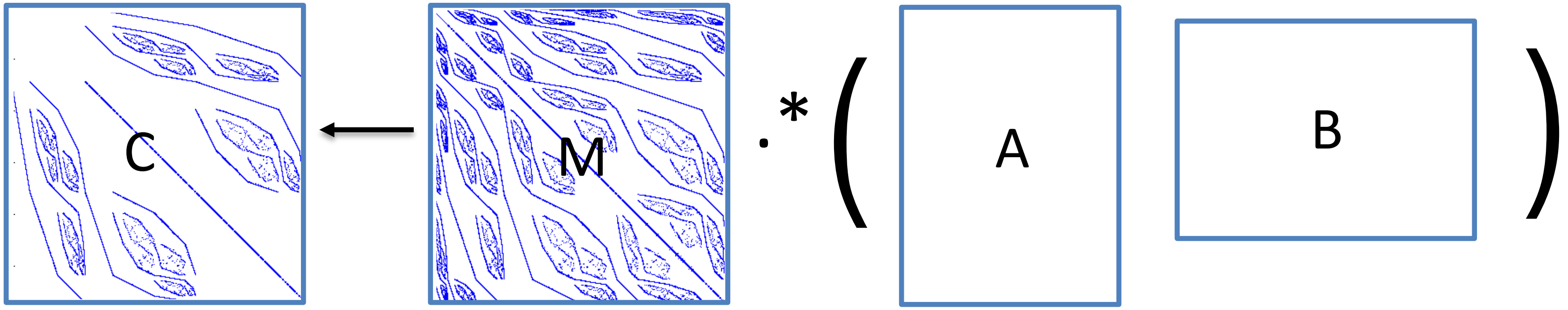
Factored: LUx = b (solvable by direct substitution)

Talk Outline

- Sparse matrix-matrix multiplication
  - SpGEMM use cases
  - Masked SpGEMM use case and new algorithms
  - SpMM (+SDDMM) use cases and new algorithms
- The GraphBLAS effort
  - Combinatorial BLAS 2.0
  - GraphBLAST

Sparse matrix-matrix multiplication

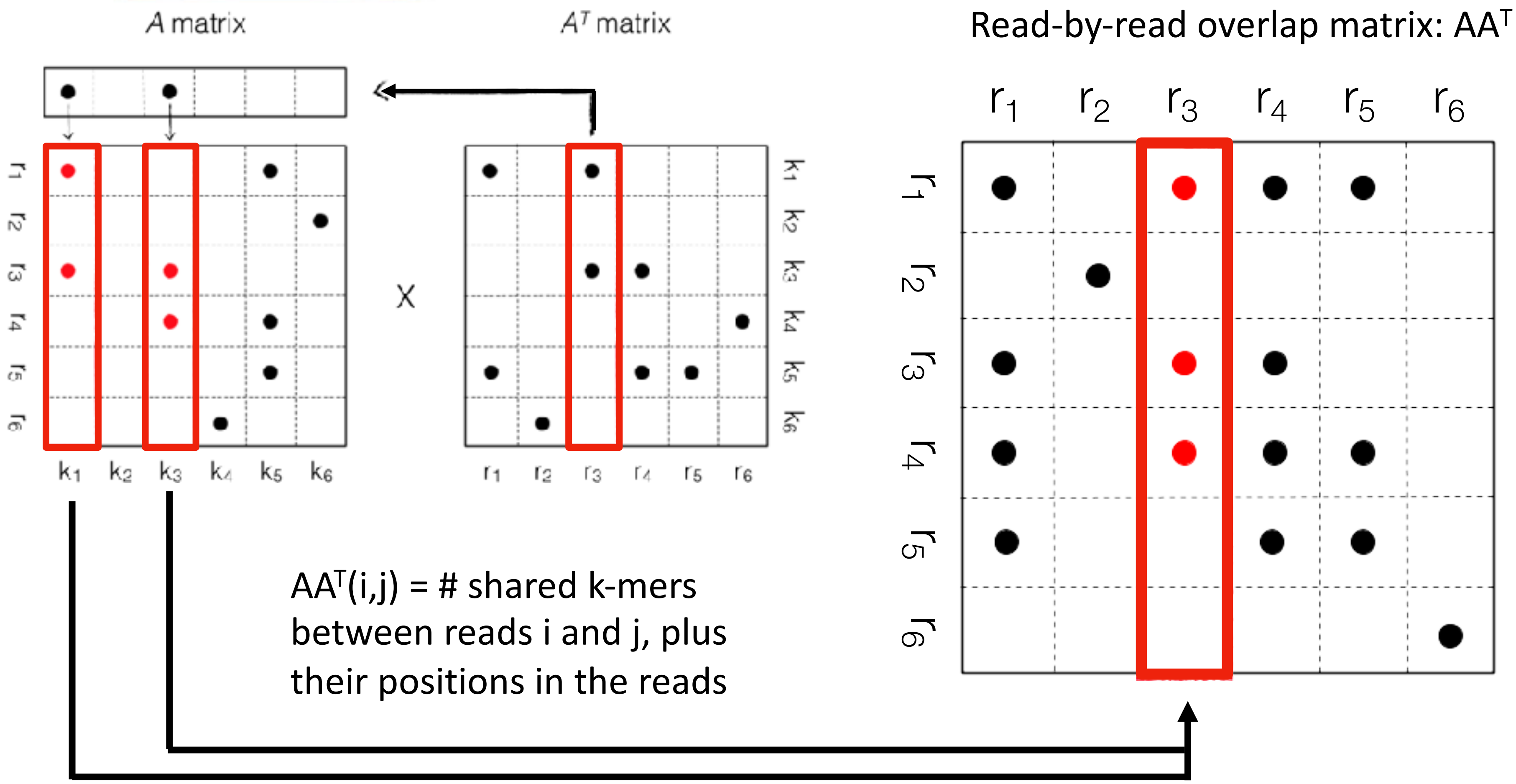
$$C(\neg M) \oplus= A^{T} \oplus . \otimes B^{T}$$



M: the output mask (also called a sampling matrix), **always sparse if present**
A, B: input matrices, at least one is sparse unless the mask is present
C: output matrix

SpGEMM: A, B are sparse, C can be sparse or dense (depending on shape)
Masked-SpGEMM: Same as SpGEMM, with mask (M) present
SpMM: A sparse, B and C dense (tall skinny), often no mask (M)
SDDMM: A, B are dense, M present, C sparse
SpMV: degenerate case of SpMM with B and C having 1 column
SpMSpV: degenerate case of SpGEMM with B, C, (possibly M) having 1 column

SpGEMM use case #1: read overlapping



Read-by-read overlap matrix: AA^T

$AA^T(i,j)$ = # shared k-mers between reads i and j, plus their positions in the reads

Use any fast SpGEMM algorithm, as long as it runs on ***arbitrary semirings***

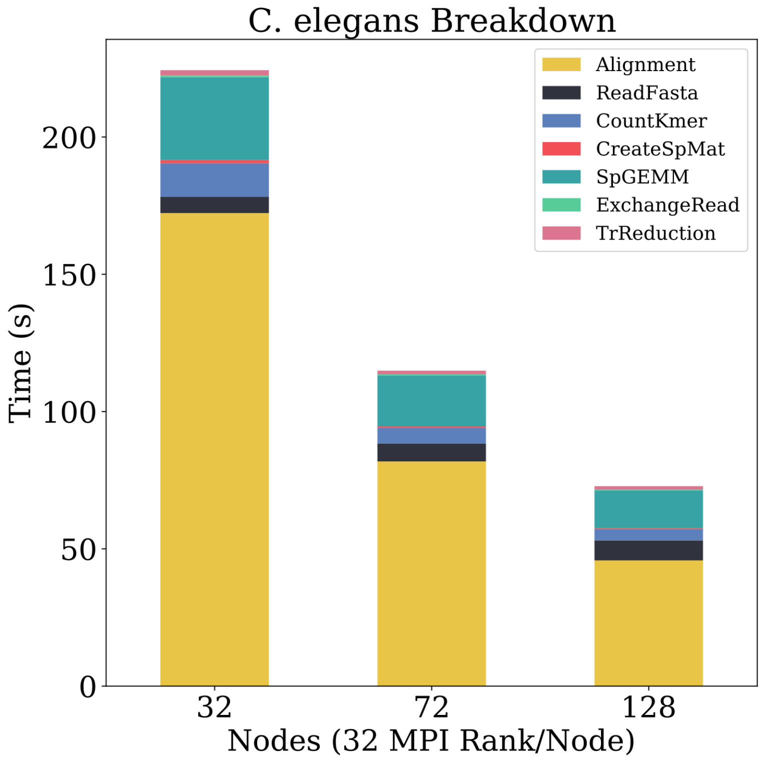
Giulia Guidi, Marquita Ellis, Daniel Rokhsar, Kathy Yelick, Aydın Buluç. BELLA: Berkeley Efficient Long-read to Long-Read Overlapper and Aligner. In SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21) 2021

diBELLA.2D performance results

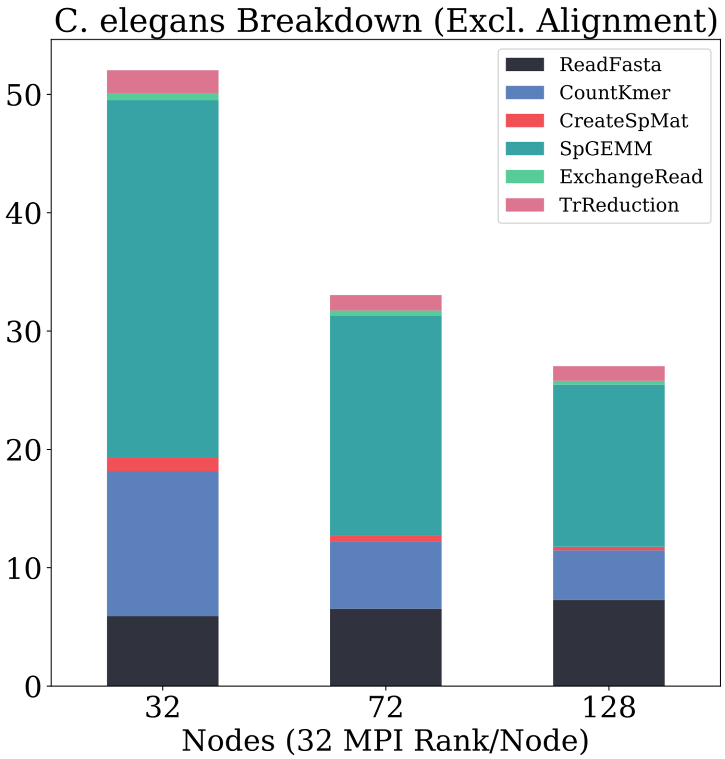
diBELLA.2D: distributed-memory version of BELLA on 2D process grid

Performs *overlap detection* plus *transitive reduction (overlap to string graph)*

https://github.com/PASSIONLab/diBELLA.2D



C. elegans Breakdown

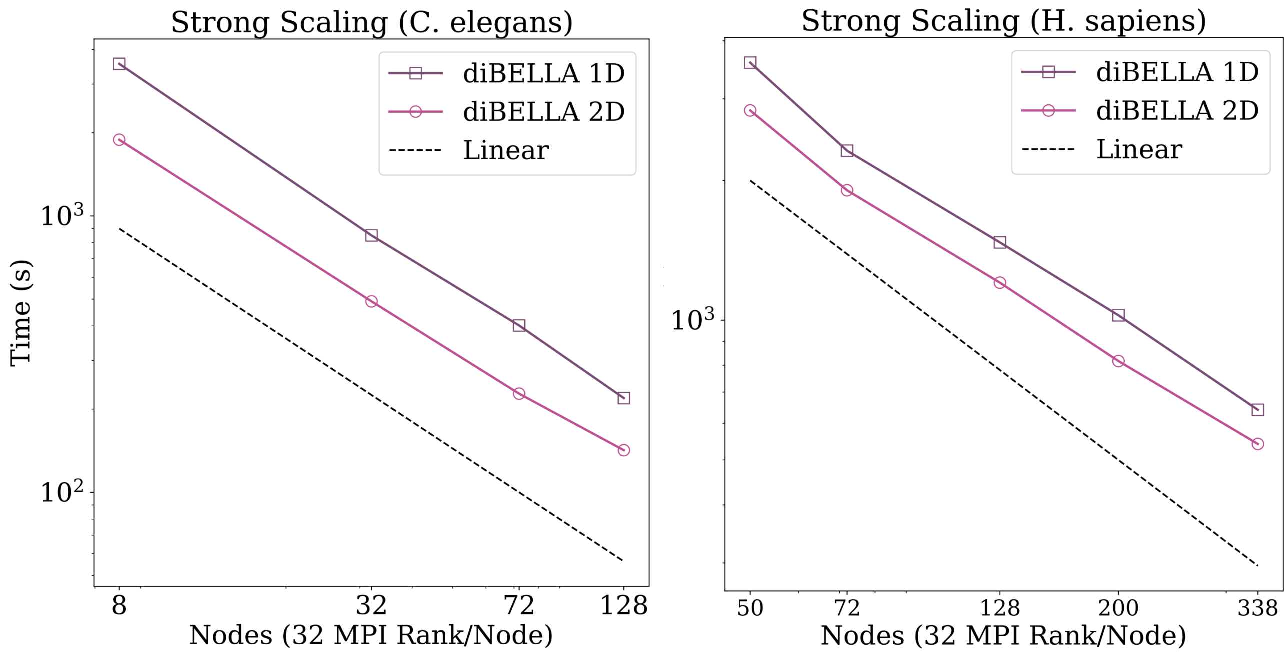


C. elegans Breakdown (Excl. Alignment)

Giulia Guidi, Oguz Selvitopi, Marquita Ellis, Leonid Oliker, Katherine Yelick, Aydin Buluç. Parallel String Graph Construction and Transitive Reduction for De Novo Genome Assembly. *IPDPS* 2021

Is the sparse matrix approach better?

- Comparing the sparse matrix abstraction (diBELLA 2D [2], **weeks** of effort) with a direct implementation (diBELLA 1D [1], **years** of effort). Both use MPI
- Sparse matrices reduce communication via 2D sparse SpGEMM

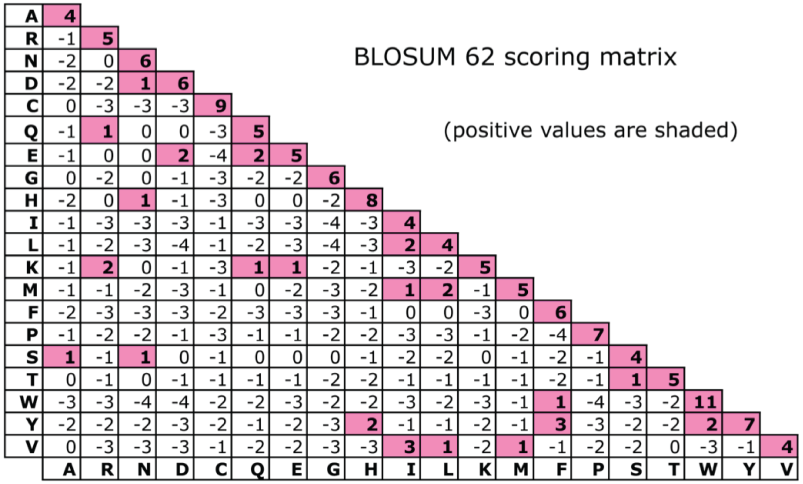


[1] Marquita Ellis, Giulia Guidi, Aydin Buluç, Leonid Oliker, and Katherine Yelick. "diBELLA: Distributed long read to long read alignment." ICPP 2019

[2] Giulia Guidi, Oguz Selvitopi, Marquita Ellis, Leonid Oliker, Katherine Yelick, Aydin Buluç. Parallel String Graph Construction and Transitive Reduction for De Novo Genome Assembly. *IPDPS* 2021

SpGEMM use case #2: many-to-many protein alignment

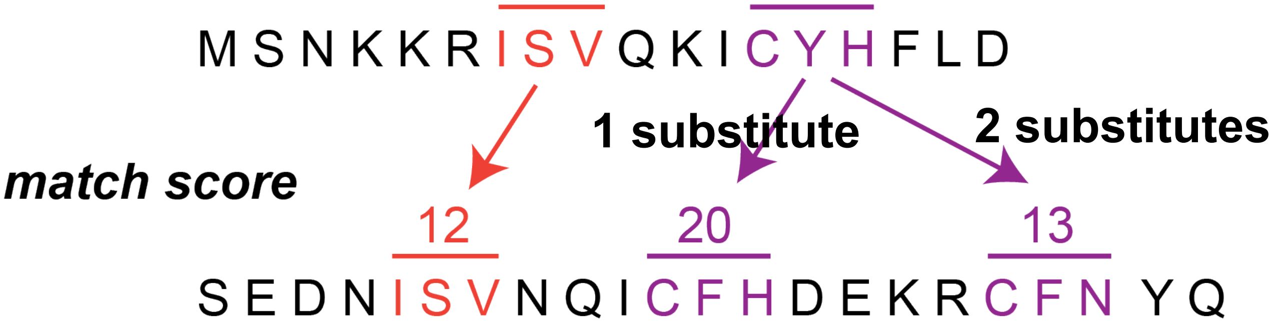
- Idea similar to BELLA, but removing the exact match restriction
- For homology detection, need to catch weaker signal (~30% ANI)
- K-mers with substitutes may be more valuable than exact matches!



BLOSUM 62 scoring matrix

(positive values are shaded)

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4																			
R	-1	5																		
N	-2	0	6																	
D	-2	-2	1	6																
C	0	-3	-3	-3	9															
Q	-1	1	0	0	-3	5														
E	-1	0	0	2	-4	2	5													
G	0	-2	0	-1	-3	-2	-2	6												
H	-2	0	1	-1	-3	0	0	-2	8											
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4



M S N K K R I S V Q K I C Y H F L D

1 substitute

2 substitutes

match score

12

20

13

S E D N I S V N Q I C F H D E K R C F N Y Q

SpGEMM for many-to-many protein alignment

PASTIS (https://github.com/PASSIONLab/PASTIS) does distributed many-to-many protein sequence similarity search using sparse matrices

Introduce new sparse matrix **S**

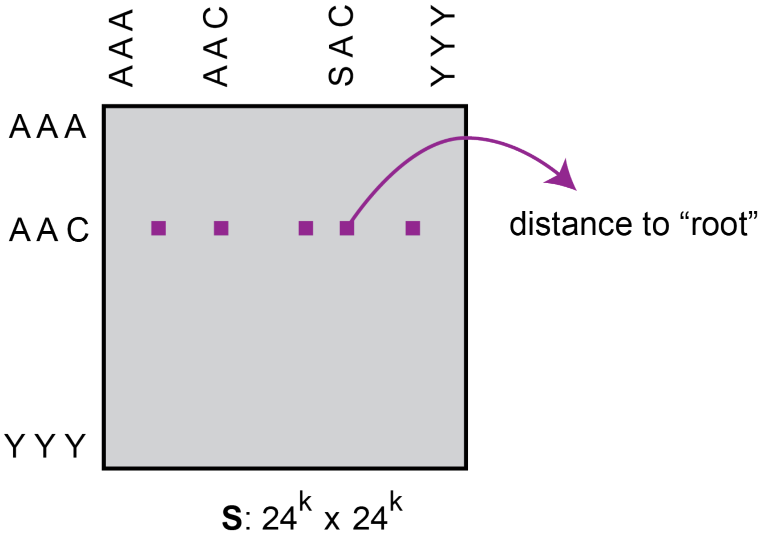
Contains substitution information

Each entry has **substitution cost**

Exact k-mers → C=AA^T

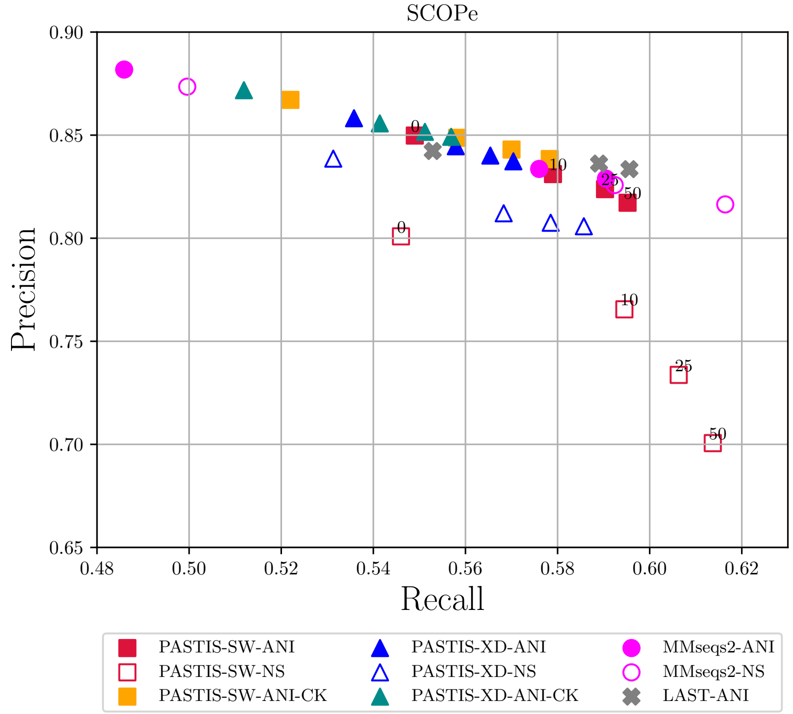
Substitute k-mers → C=ASA^T

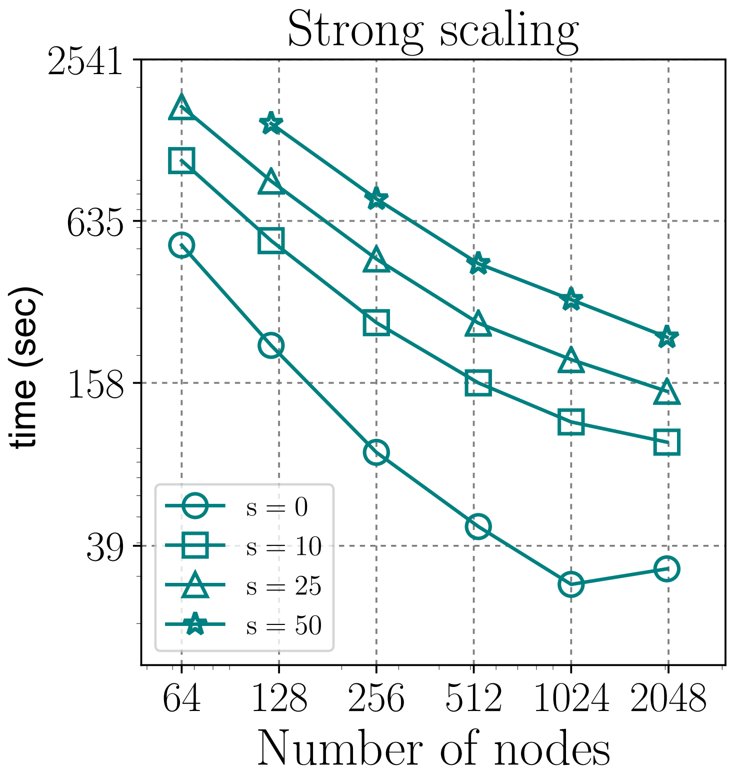
New semiring



Oguz Selvitopi, Saliya Ekanayake, Giulia Guidi, Georgios Pavlopoulos, Ariful Azad, and Aydın Buluç. Distributed Many-to-Many Protein Sequence Alignment Using Sparse Matrices. SC'20.

PASTIS performance and accuracy





- *Protein similarity search* is the first and most time-consuming step in discovering protein families (proteins evolved from a common ancestor and who likely have the same function)
- *Protein family identification* is a key step in protein function discovery and taxonomic assignment of newly sequenced organisms

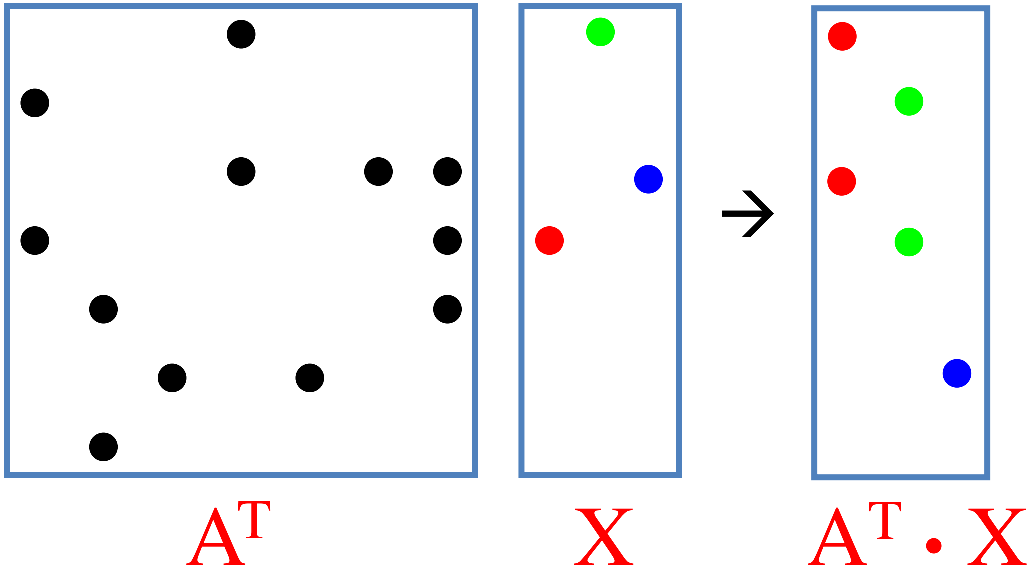
Masked SpGEMM use case: graph traversal

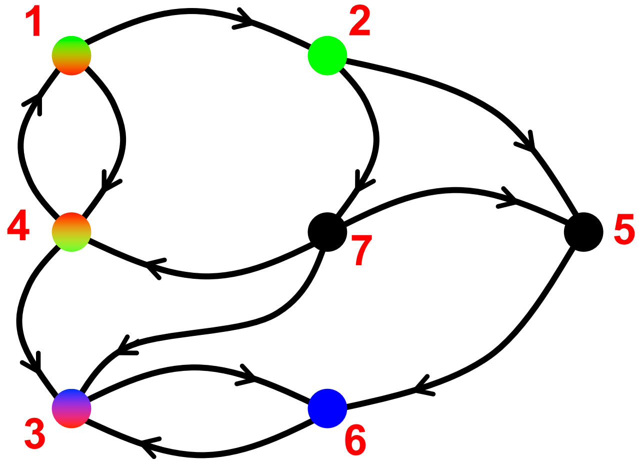
Multi-source traversal:

Ex: multi-source BFS, betweenness centrality, triangle counting*, Markov clustering*

`GrB_mxm(Y, P, <semiring>, A, X, <desc>)`

A: sparse adjacency matrix
X: sparse input matrix (previous frontier), n-by-b where b is the #sources
P: mask (already discovered vertices), multi-vector version of p from previous slide

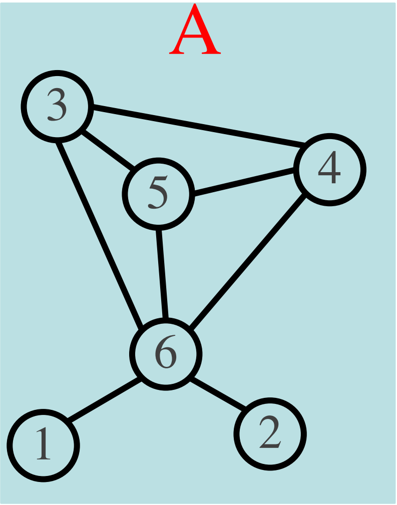




Masked SpGEMM use case: graph traversal

Triangle counting is also multi-source(in fact, all sources) traversal:
It just stops after one traversal iteration only, discovering all wedges

GrB_mxm(C, A, <semiring>, L, U, <desc>)

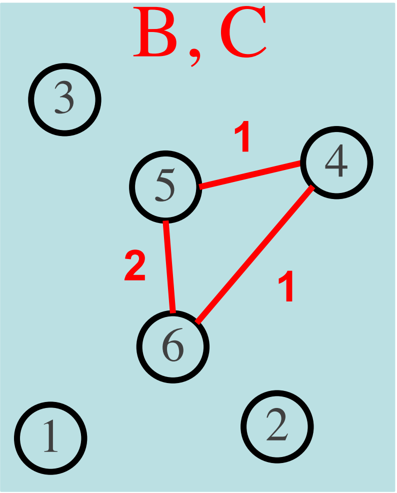


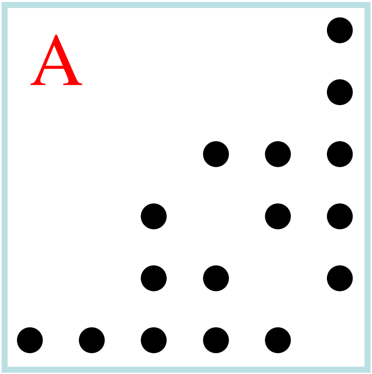
A = L + U (hi->lo + lo->hi)

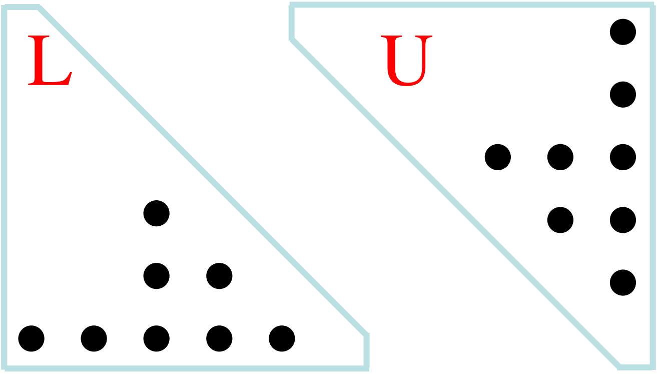
L × U = B (wedge, low hinge)

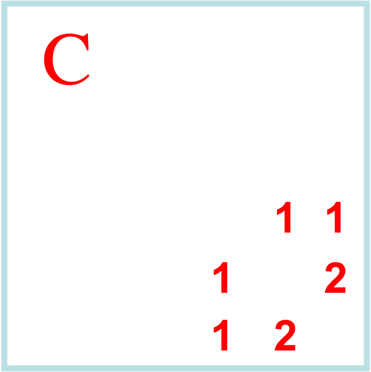
A ^ B = C (closed wedge)

sum(C)/2 = **4 triangles**



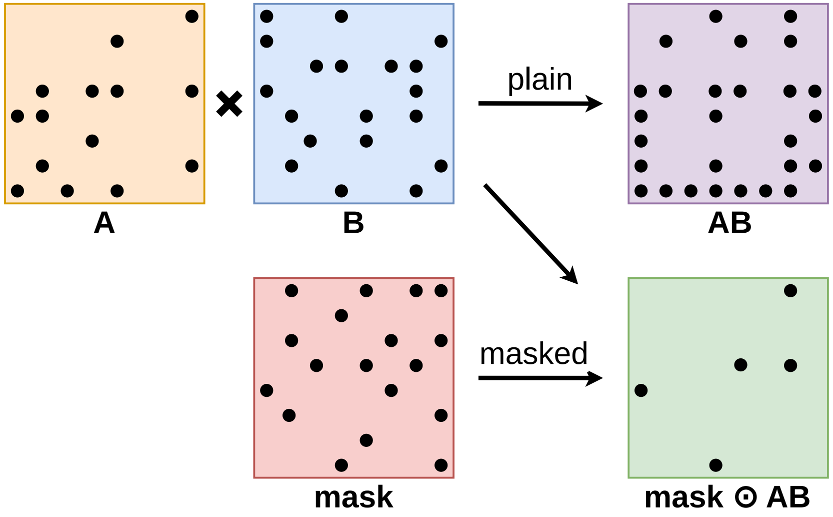


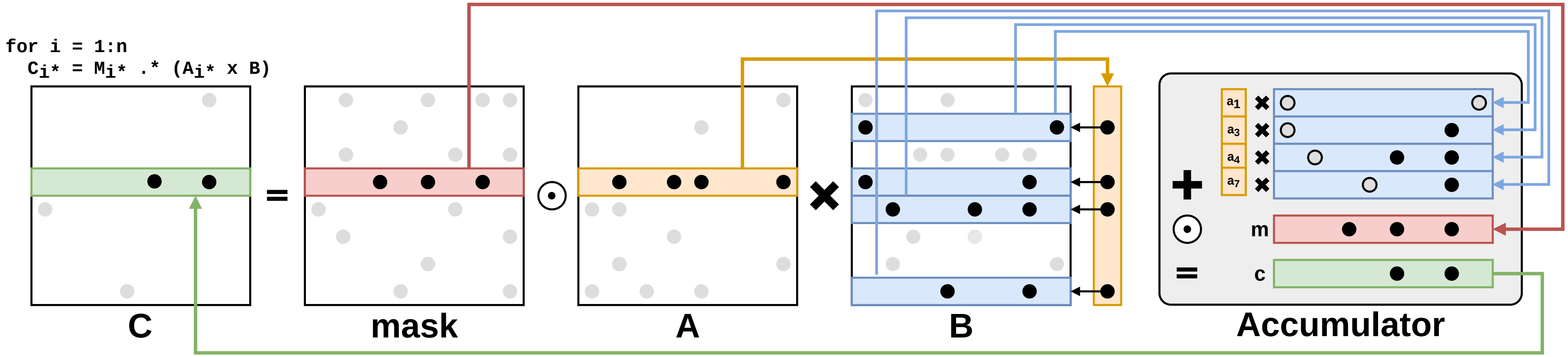




New algorithms for Masked SpGEMM

Main Idea: When certain output entries of SpGEMM are not needed (masked out), it is wasteful to materialize/compute the product first and then to mask out entries





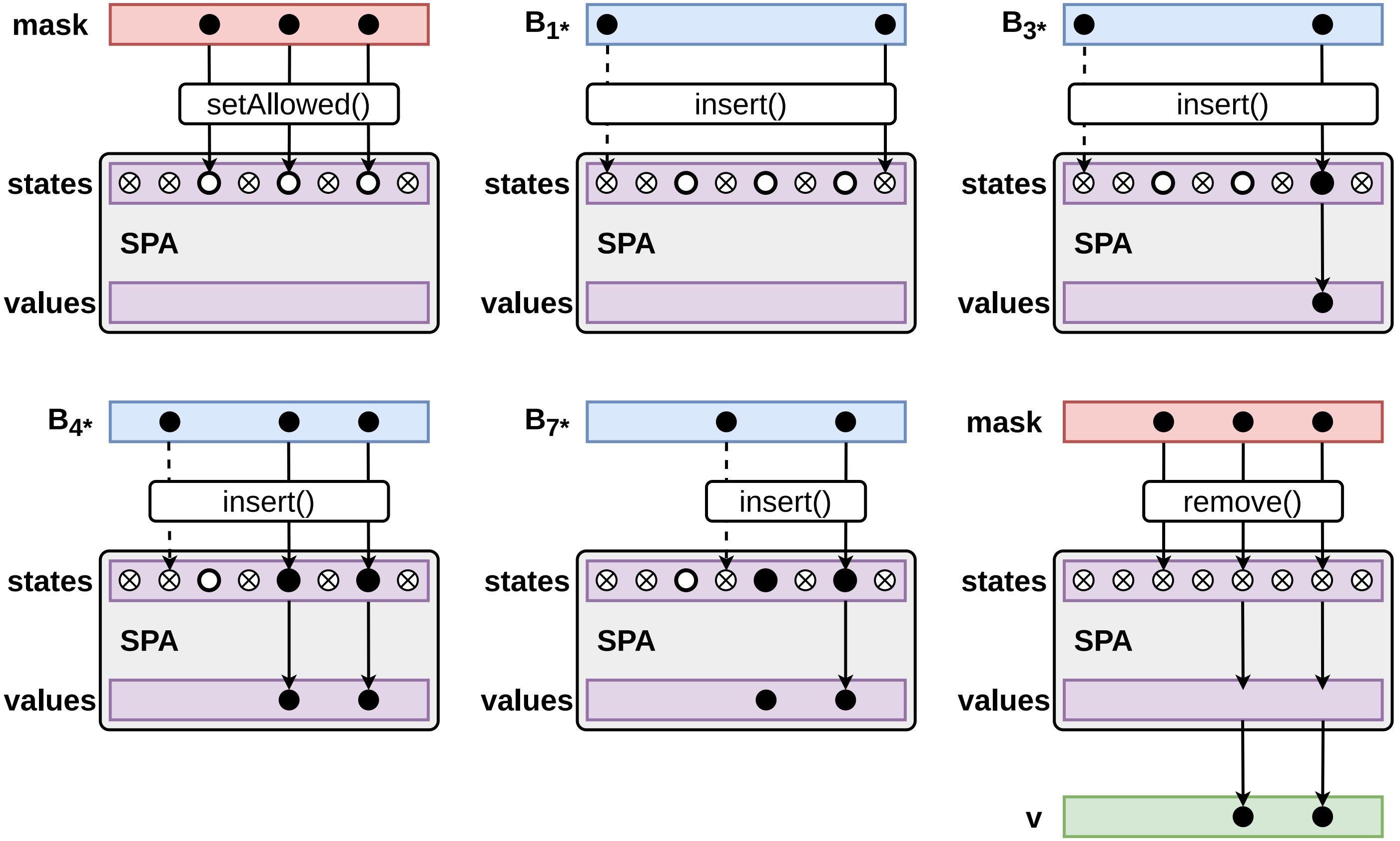
```
for i = 1:n
  Ci* = Mi* .* (Ai* x B)
```

- Row-wise Masked SpGEMM using an accumulator to compute output row C_{i*}.
- The rows corresponding to the column indices of entries in row A_{i*} are merged and filtered through the respective mask entries to compute C_{i*}.
- This merging and filtering process can be performed in a number of ways.

Masked Sparse Accumulator (MSA)

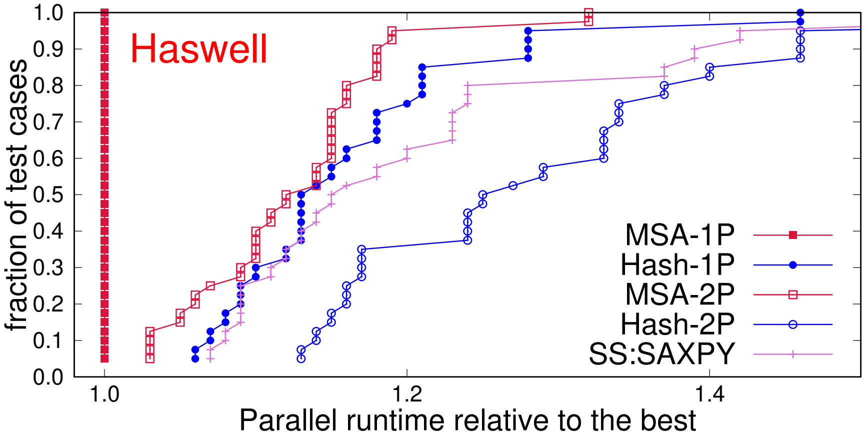
Execution of 1 row of SpGEMM with Masked Sparse Accumulator (MSA)

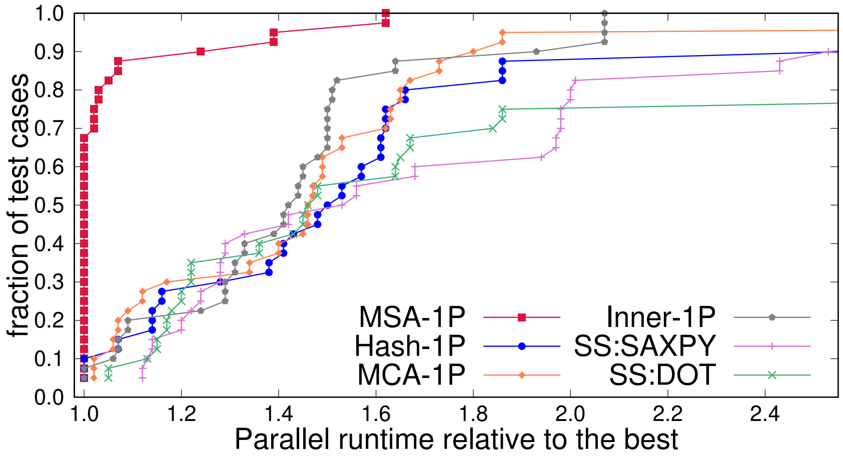
(a) initialize (b) MSA+=u_1 B_{1*} (c) MSA+=u_3 B_{3*} (d) MSA+=u_4 ×B_{4*} (e) MSA+=u_7 ×B_{7*} (f) output

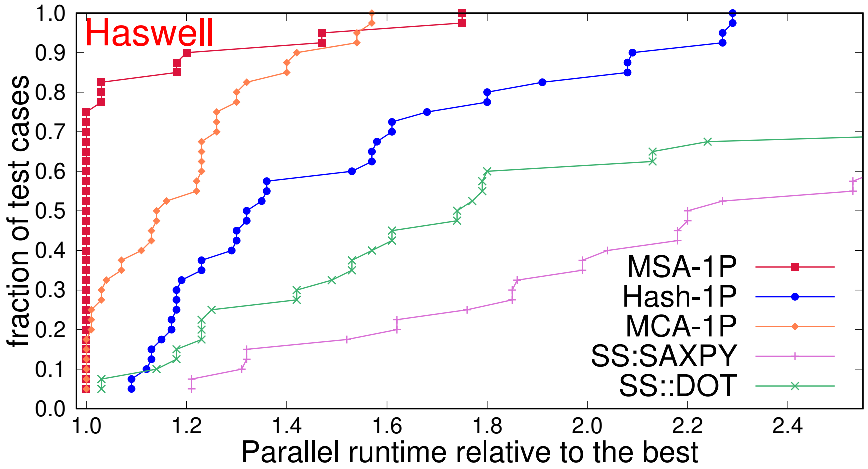


Srdjan Milaković , Oguz Selvitopi, Israt Nisa, Zoran Budimlić , and Aydın Buluç. Parallel algorithms for masked sparse matrix-matrix products. *arXiv preprint arXiv:2111.09947*, 2021 (Poster at PPOPP’22)

Performance of Masked SpGEMM algorithms







Top (left): Betweenness Centrality

Top (right): k-truss

Bottom: Triangle counting

SS is the Suitesparse:GraphBLAS

SS:DOT and Inner-1P do sparse dot products

Srdjan Milaković, Oguz Selvitopi, Israt Nisa, Zoran Budimlić, and Aydın Buluç. Parallel algorithms for masked sparse matrix-matrix products. *arXiv preprint arXiv:2111.09947*, 2021 (Poster at PPOPP'22)

Motivation for Graph Neural Networks

"GNNs are among the most general class of deep learning architectures currently in existence, […] and most other deep learning architectures can be understood as a special case of the GNN with additional geometric structure"

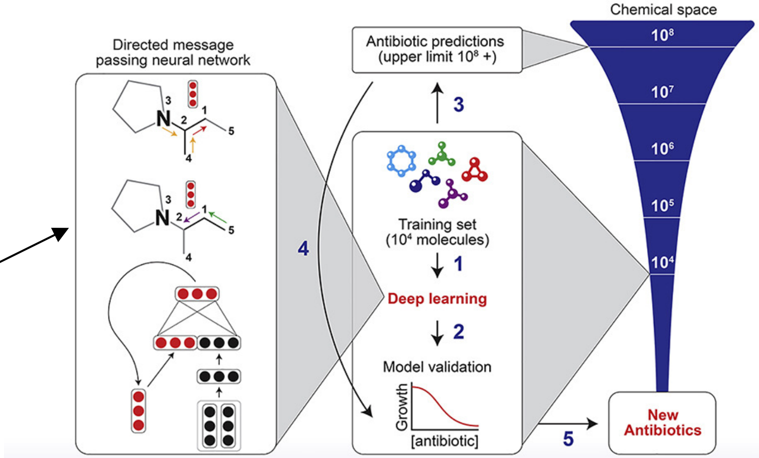
Bronstein, Michael M., et al. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges." (2021)

NEWS · 20 FEBRUARY 2020

Powerful antibiotics discovered using AI

Machine learning spots molecules that work even against 'untreatable' strains of bacteria.

This is a graph neural network



Article | Published: 09 June 2021

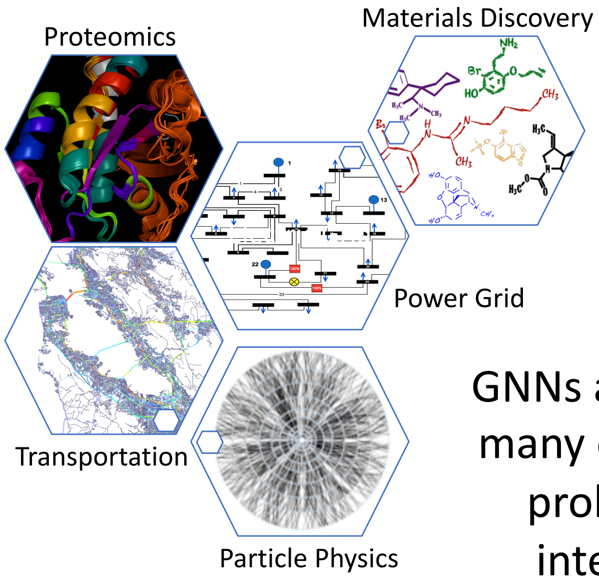
A graph placement methodology for fast chip design

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter & Jeff Dean

Nature **594**, 207–212 (2021) | Cite this article

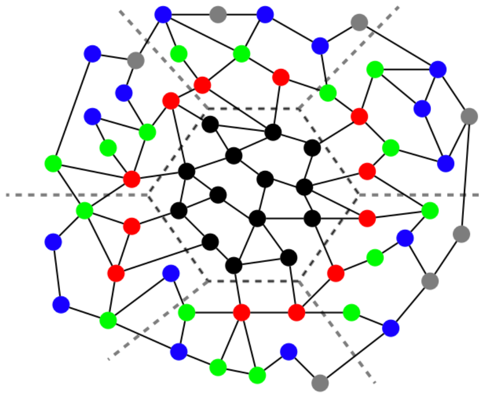
… we pose chip floorplanning as a reinforcement learning problem, and develop an **edge-based graph convolutional neural network** architecture…

Graph Neural Networks (GNNs)



GNNs are finding success in many challenging scientific problems that involve interconnected data.

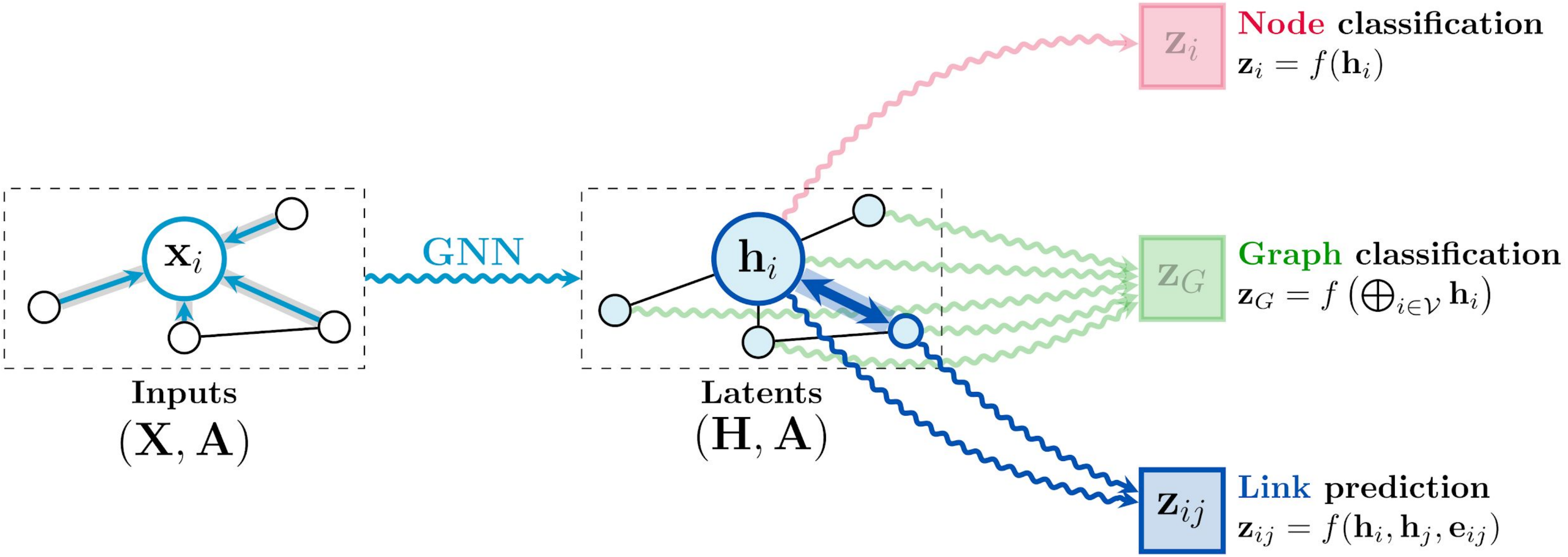
Interdependencies between samples (nodes of the graph) make stochastic gradient non-trivial without graph sampling



- GNNs are computationally intensive to train. Distributed training need to scale to large GPU/node counts despite challenging sparsity.
- CAGNET (Communication-Avoiding Graph Neural nETworks) full gradient descent to avoid inaccurate (and expensive) graph sampling

https://github.com/PASSIONLab/CAGNET/

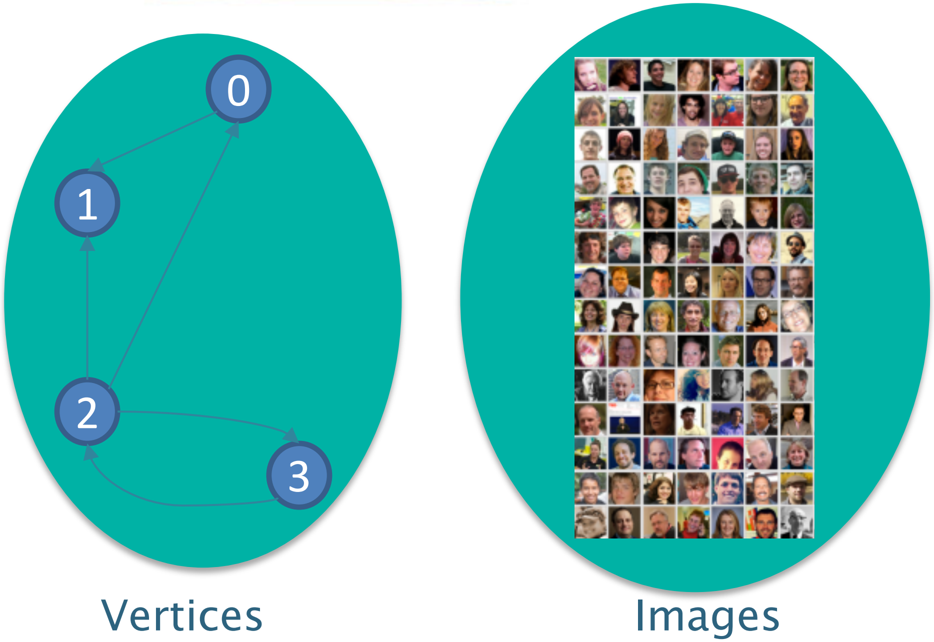
What can I do with a GNN?



$\mathbf{x}_i$
Inputs
$(\mathbf{X}, \mathbf{A})$
GNN
$\mathbf{h}_i$
Latents
$(\mathbf{H}, \mathbf{A})$
$\mathbf{z}_i$
Node classification
$\mathbf{z}_i = f(\mathbf{h}_i)$
$\mathbf{z}_G$
Graph classification
$\mathbf{z}_G = f\left(\bigoplus_{i \in \mathcal{V}} \mathbf{h}_i\right)$
$\mathbf{z}_{ij}$
Link prediction
$\mathbf{z}_{ij} = f(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij})$

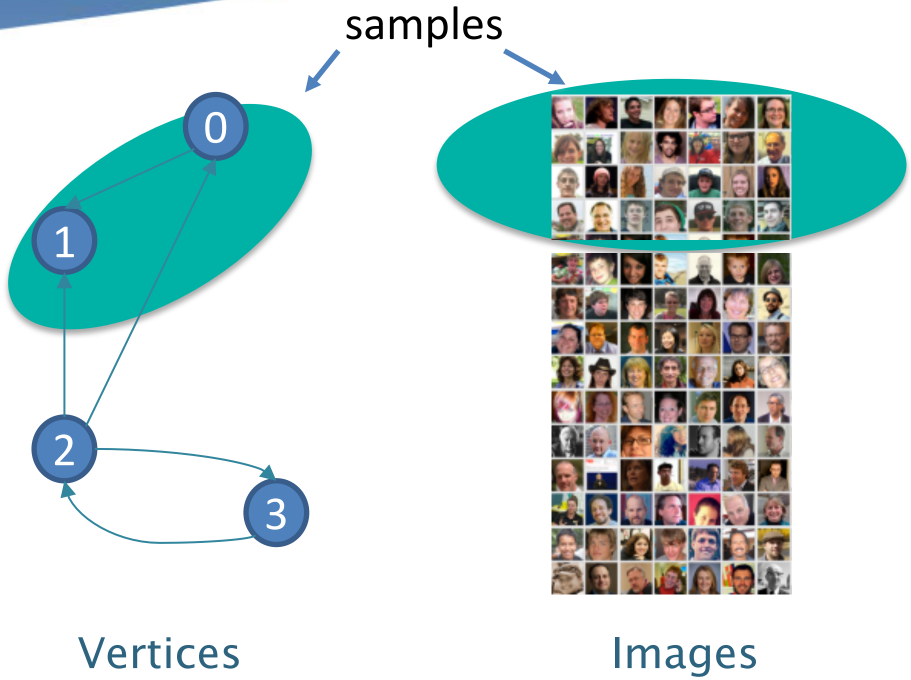
Figure source: Petar Veličković

Full-graph vs. mini-batch SGD



Full-graph training:

- Train on **entire** training set
- Slower convergence per epoch
- Faster training per epoch
- **Focus of this work**



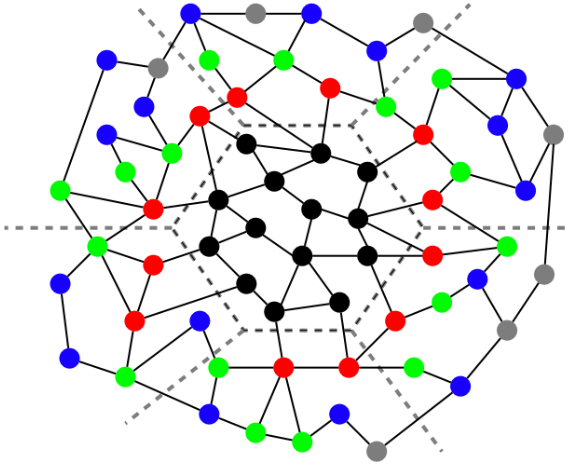
Mini-batch SGD:

- Train on multiple **samples** from training set
- Faster convergence per epoch
- Slower training per epoch
- Requires graph sampling, which effects accuracy and performance

Full-graph vs. mini-batch SGD

sample

No dependencies

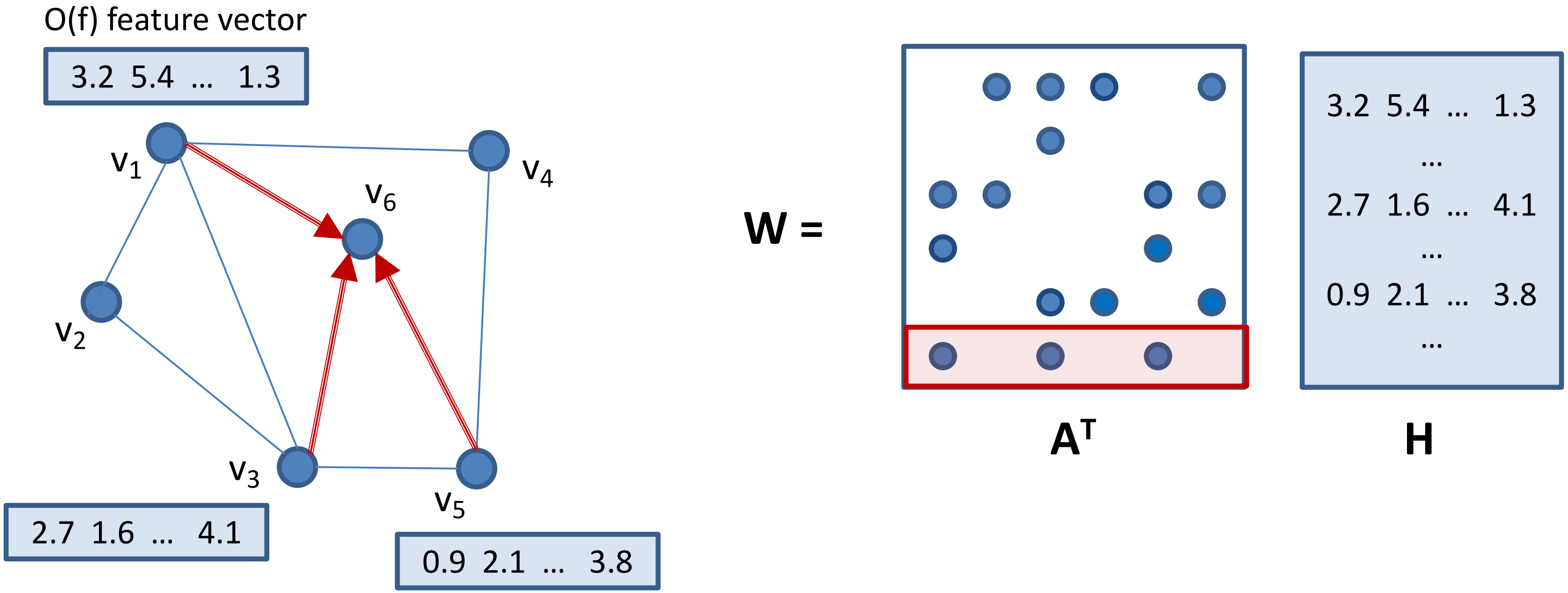


Layered dependencies

- Vertices (unlike images) are dependent on each other
- L-layer GNN uses L-hop neighbors for vertices in batch
- Even for small L, must store ~whole graph for any minibatch for power-law graphs
- How to subsample from aggregated L-hop neighborhood and keep accuracy?
- CAGNET (Communication-Avoiding Graph Neural nETworks) full gradient descent to avoid such issues: https://github.com/PASSIONLab/CAGNET/

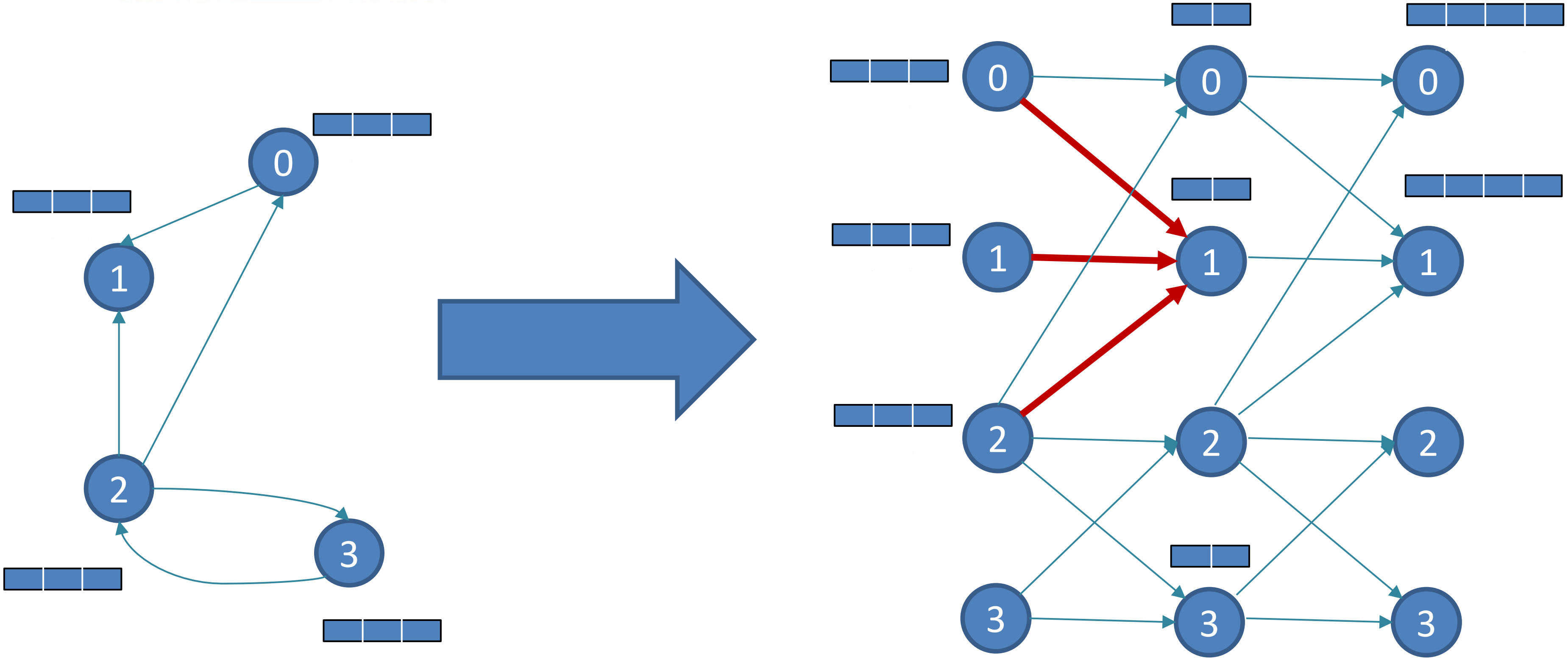
Graph convolutions

Graph convolution: Feature aggregation from neighbors



- GNN is an umbrella term for any neural network that performs graph representation learning.
- CAGNET focuses on Graph Convolutional Networks (GCNs)
- We are working on adding graph attention layers

Graph convolutions

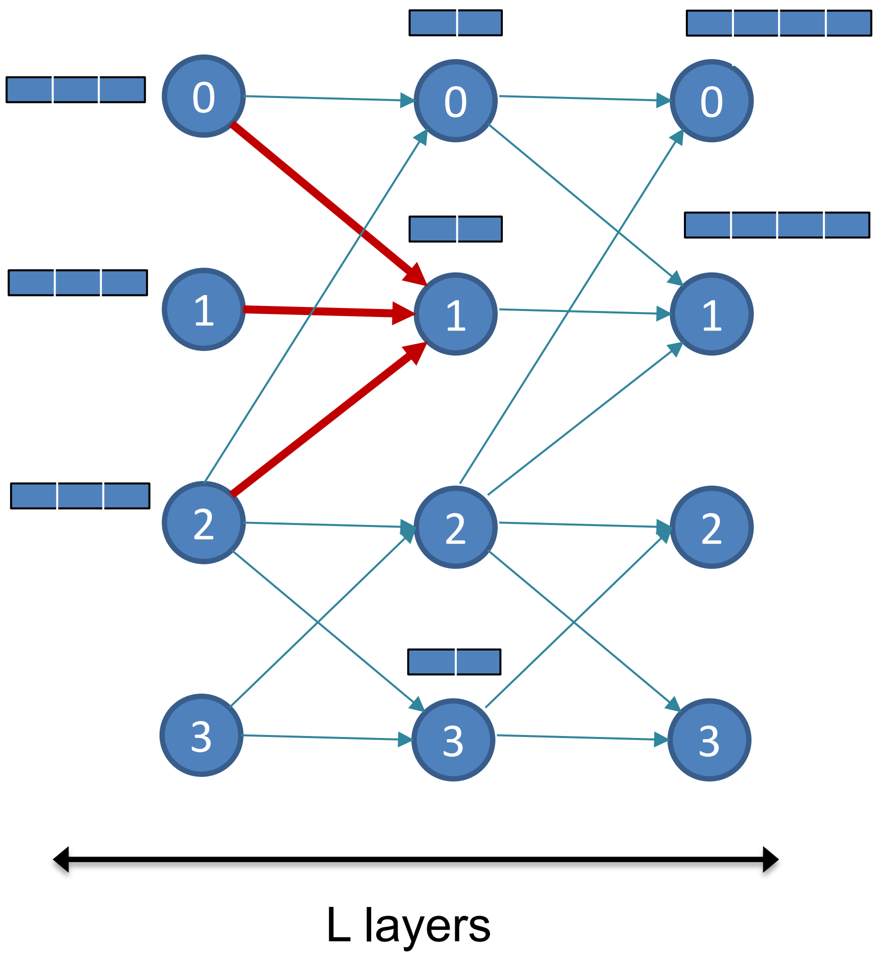


Input Graph

GNN of Input Graph

- Recall that a CNN can have different *channel* dimension at each layer.
- GNNs also have different embedding dimension at each layer

Memory cost of full-batch GCN training



Storage= $\sum_{i=1}^{L} nf^{i}$

$\approx O(nLf)$

Where $f = \frac{\sum_{i=1}^{L} f^{i}}{L}$

Say n = 100M, L = 4, f = 256, we are looking at 100B words, or 800GB

GNN Training

- Each node is initialized with a feature vector
  - H^0 has initial feature vector per node $(n\ x\ f)$
- Each node aggregates vectors of its neighbors, applies a weight
- Each layer computes gradients

```
for i = 1 … E
    for l = 1 … L
        $Z^l = A^T * H^{l-1} * W^l$
        $H^l = \sigma(Z^l)$
    ...
    for l = L-1 … 1
        $G^l = A * G^{l+1} * (W^{l+1})^T \odot \sigma'(Z^l)$
       $dH/dW = (H^{l-1})^T * A * G^l$
```

$A \in n\ x\ n$

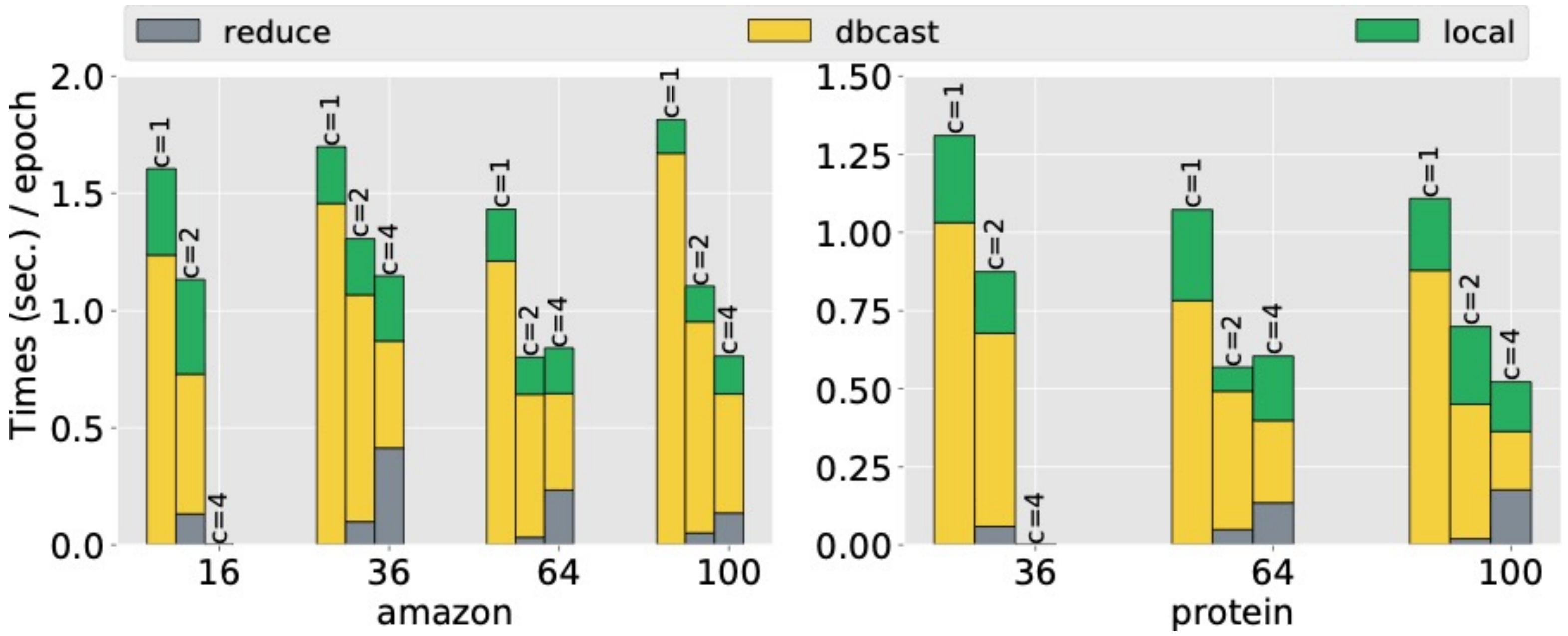
$H^l \in n\ x\ f^l$

$G^l \in n\ x\ f^l$

$W^l \in f^{l-1}\ x\ f^l$

- A is sparse and f << n, so the main workhorse is SpMM (sparse matrix times tall-skinny dense matrix)

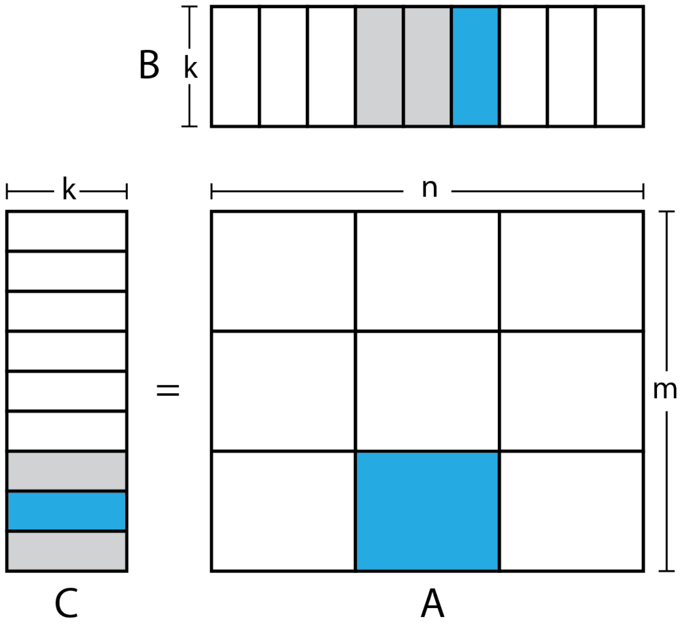
Communication avoidance (CA) In GNN Training



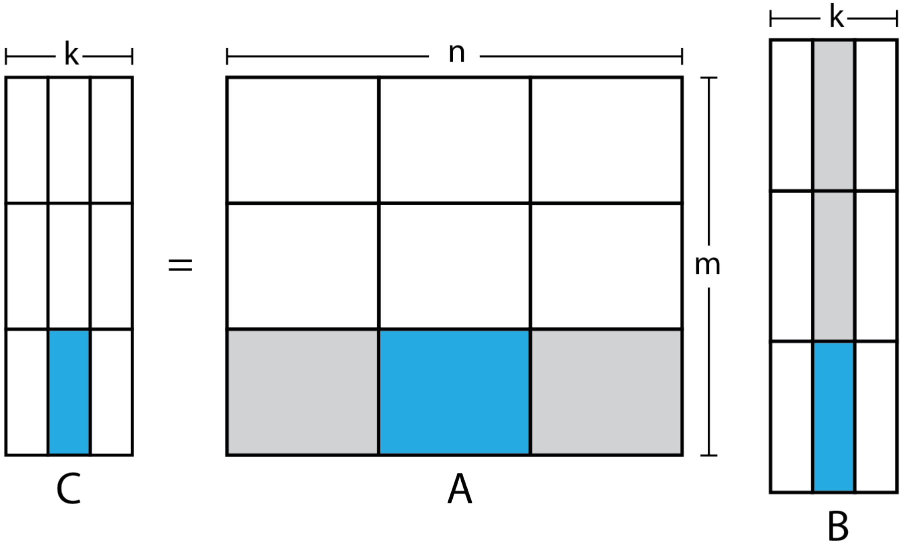
- Scales with both P (GPUs – x axis) and c (replication layers in CA algorithms)
- This is 1 GPU/node on Summit (all GPUs per node results in paper)
- Expect to scale with all GPUs / node with future architectures (e.g. Perlmutter)
- More results (2D and 3D algorithm) and 6 GPUs/node in the paper

Alok Tripathy, Katherine Yelick, Aydın Buluç. Reducing Communication in Graph Neural Network Training. SC'20

Distributed SpMM algorithms



A is sparse, **B** and **C** are dense



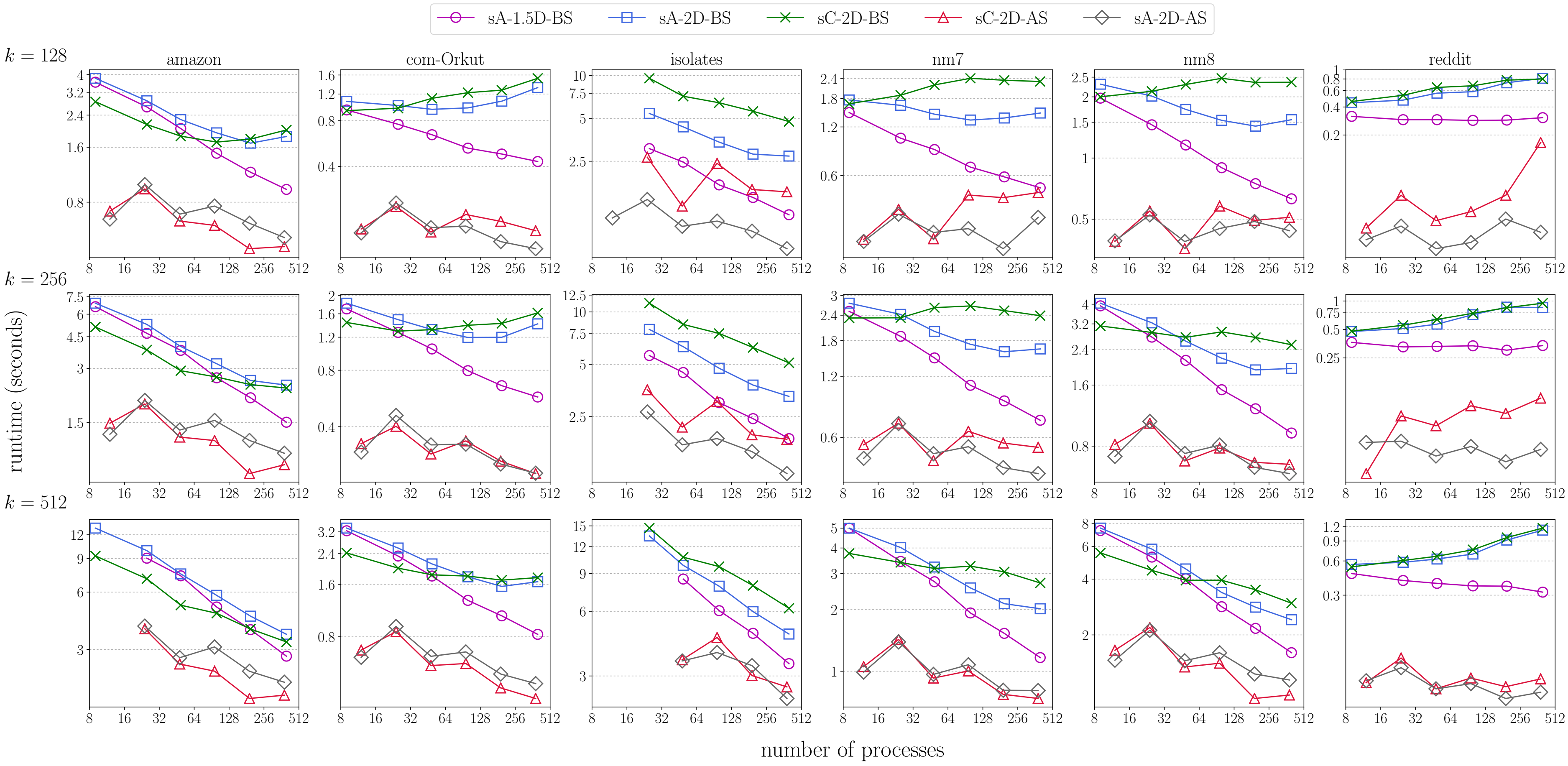
- Stationary A, 1.5D algorithm
- **A** is split on a p/c-by-c grid

- Stationary C, 2D algorithm
- Memory optimal

- 1D algorithm not shown, degeneration of sA-1.5D for the c=1 case
- Right before reduction, sA-1.5D uses c times more dense-matrix memory

Could we do SpMM differently?

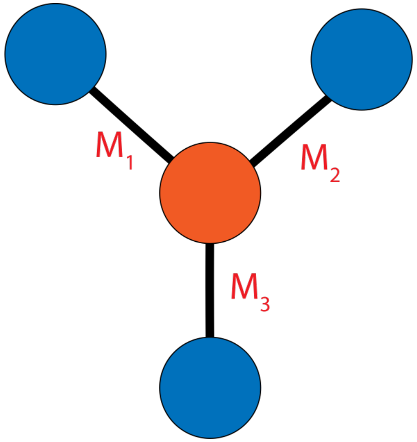
BS: bulk-synchronous (MPI)
AS: asynchronous (RDMA)



Oguz Selvitopi , Benjamin Brock, Israt Nisa, Alok Tripathy, Katherine Yelick, Aydın Buluç. Distributed-Memory Parallel Algorithms for Sparse Times Tall-Skinny-Dense Matrix Multiplication. ICS'21

Sparse Kernels in Machine Learning

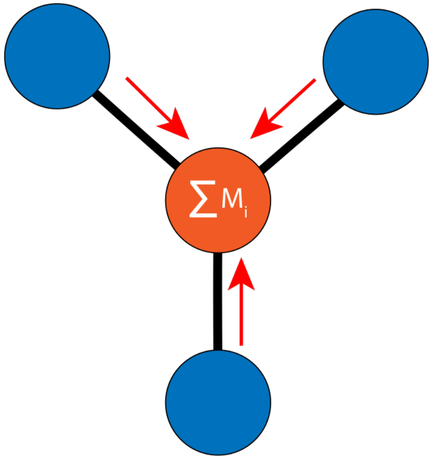
- Sampled Dense-Dense Matrix Multiplication (SDDMM) and Sparse-times-Dense Matrix Multiplication (SpMM) appear in a variety of applications:
  - Graph Neural Networks with Self-Attention
  - Collaborative Filtering with Alternating Least Squares
  - Document Clustering by Wordmover's Distance



Message Generation

- Both kernels involve a single sparse matrix and two (typically tall-skinny) dense matrices. Typically, applications use both operations in sequence.

- When the sparse matrix is the adjacency matrix of a graph, we interpret the kernels as follows:
  - SDDMM generates a message on each edge
  - SpMM aggregates messages from incident edges



Message Aggregation

SpMM and SDDMM algorithmic duality

SDDMM and SpMM have **identical data access patterns**.
Consider serial algorithms for both kernels:

$$\mathrm{R} \coloneqq \mathrm{SDDMM}(S, A, B)$$

```
for $(i, j) \in S$
    $R_{ij} \coloneqq S_{ij}(A_{i:} \cdot B_{j:}^T)$
```

$$\mathrm{A} \coloneqq \mathrm{SpMMA}(S, B)$$

```
for $(i, j) \in S$
    $A_{i:} \mathrel{+}= S_{ij} B_{j:}$
```

Every nonzero (i, j) requires an interaction between row i of A and row j of B.

As a result:

Every distributed algorithm for SpMM can be converted to an algorithm for SDDMM with identical communication characteristics, and vice-versa.

V. Bharadwaj, A. Buluc, J. Demmel, "Distributed Memory Sparse Kernels for Machine Learning," 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2022

Creating a parallel SDDMM algorithm from an SpMM algorithm

Consider any distributed algorithm for SpMMA that performs no replication. For all indices $k \in [1, r]$, the algorithm must (at some point)

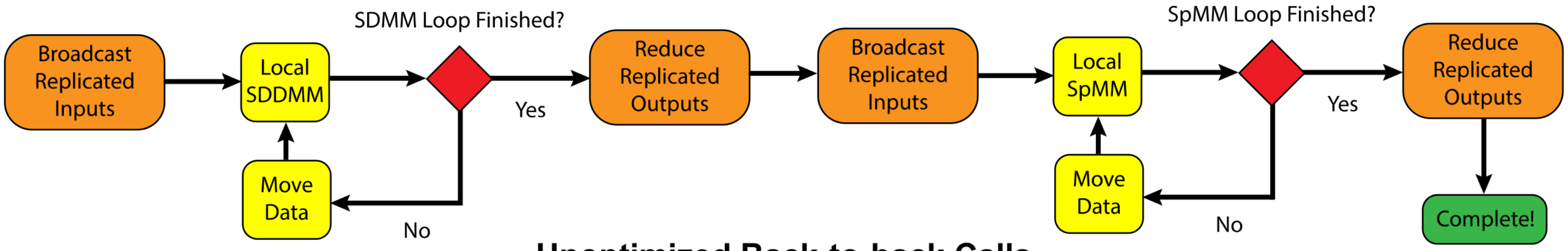
- Co-locate S_{ij}, A_{ik}, B_{jk} on a single processor
- Perform the update $A_{ik} \mathrel{+}= S_{ij} B_{jk}$

Transform this algorithm as follows:

1. Change the input sparse matrix S to an output that is initialized to 0.
2. Change A from an output to an input.
3. Have each processor execute the local update: $S_{ij} \mathrel{+}= A_{ik} B_{jk}$

The resulting algorithm performs SDDMM (up to multiplication with the values initially in S) with communication characteristics and data layout identical to the original.

Communication Eliding Strategies for FusedMM: SDDMM+SpMM



Broadcast Replicated Inputs

Local SDDMM

SDMM Loop Finished?

Yes

No

Move Data

Reduce Replicated Outputs

Broadcast Replicated Inputs

Local SpMM

SpMM Loop Finished?

Yes

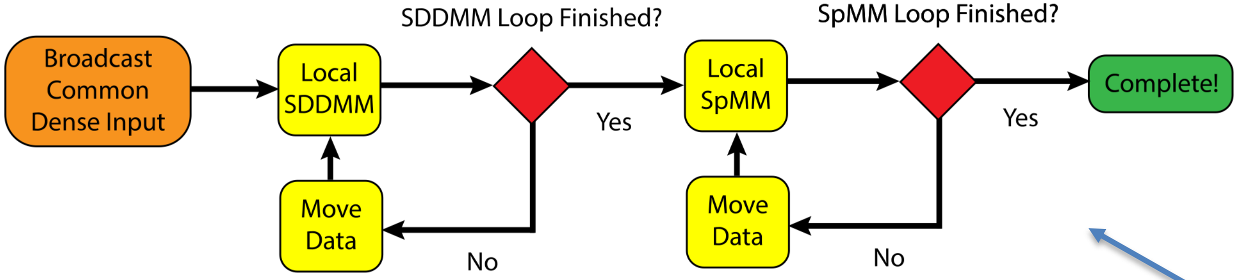
No

Move Data

Reduce Replicated Outputs

Complete!

Unoptimized Back-to-back Calls



Broadcast Common Dense Input

Local SDDMM

SDDMM Loop Finished?

Yes

No

Move Data

Local SpMM

SpMM Loop Finished?

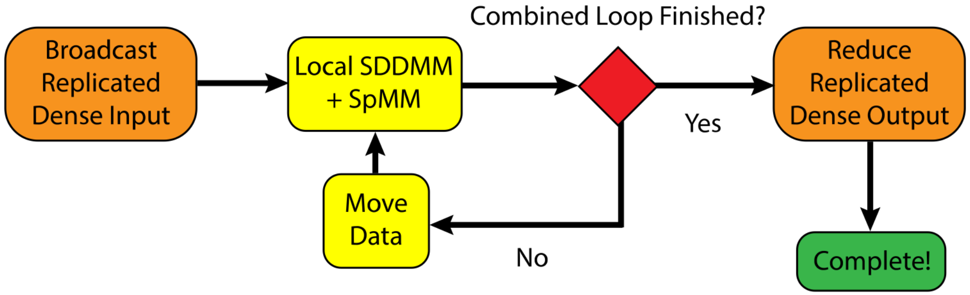
Yes

No

Move Data

Complete!

Replication Reuse



Broadcast Replicated Dense Input

Local SDDMM + SpMM

Combined Loop Finished?

Yes

No

Move Data

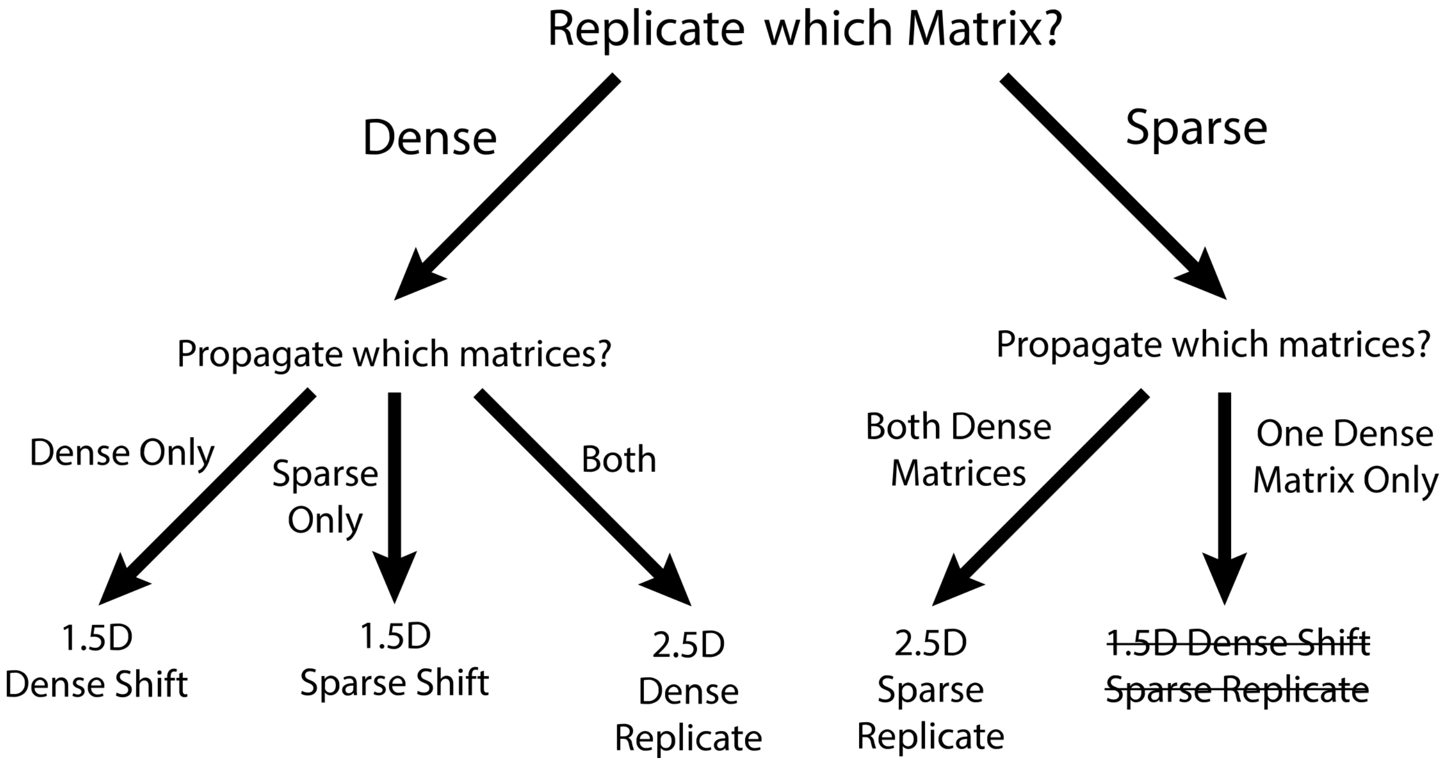
Reduce Replicated Dense Output

Complete!

Local Kernel Fusion

Mutually exclusive optimizations

Replication and Propagation Choices



Replicate which Matrix?

Dense

Sparse

Propagate which matrices?

Dense Only

Sparse Only

Both

1.5D Dense Shift

1.5D Sparse Shift

2.5D Dense Replicate

Propagate which matrices?

Both Dense Matrices

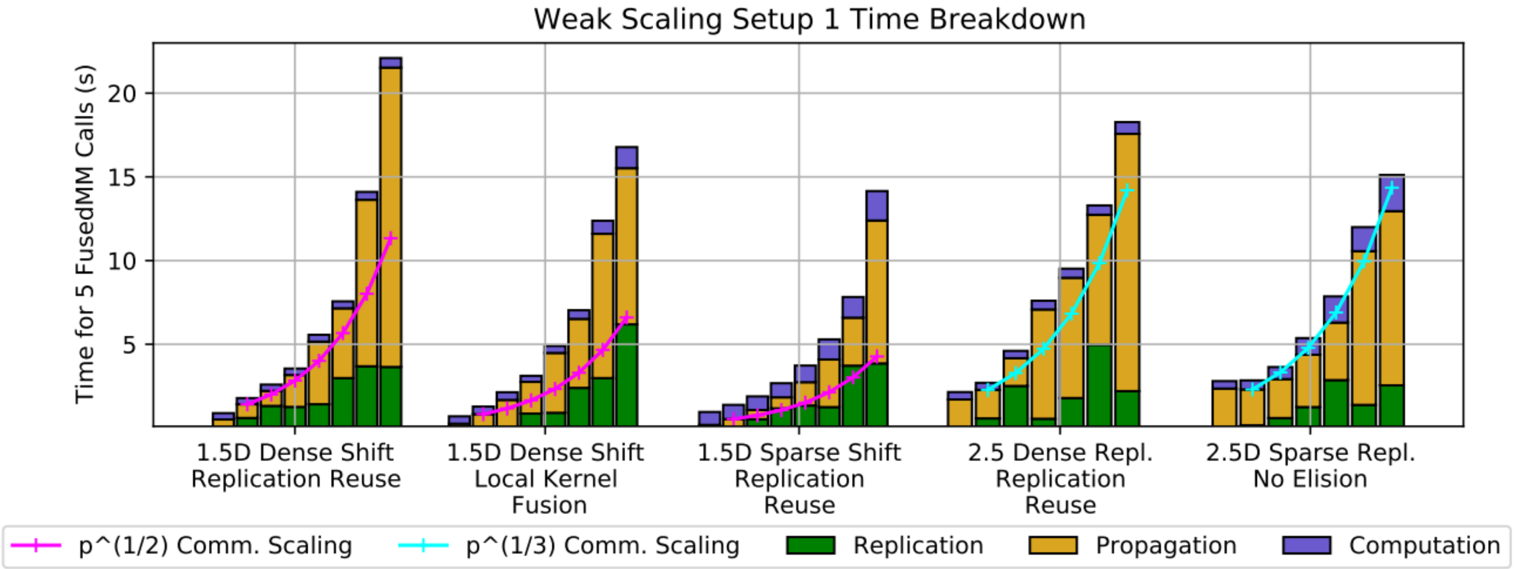
One Dense Matrix Only

2.5D Sparse Replicate

~~1.5D Dense Shift~~
~~Sparse Replicate~~

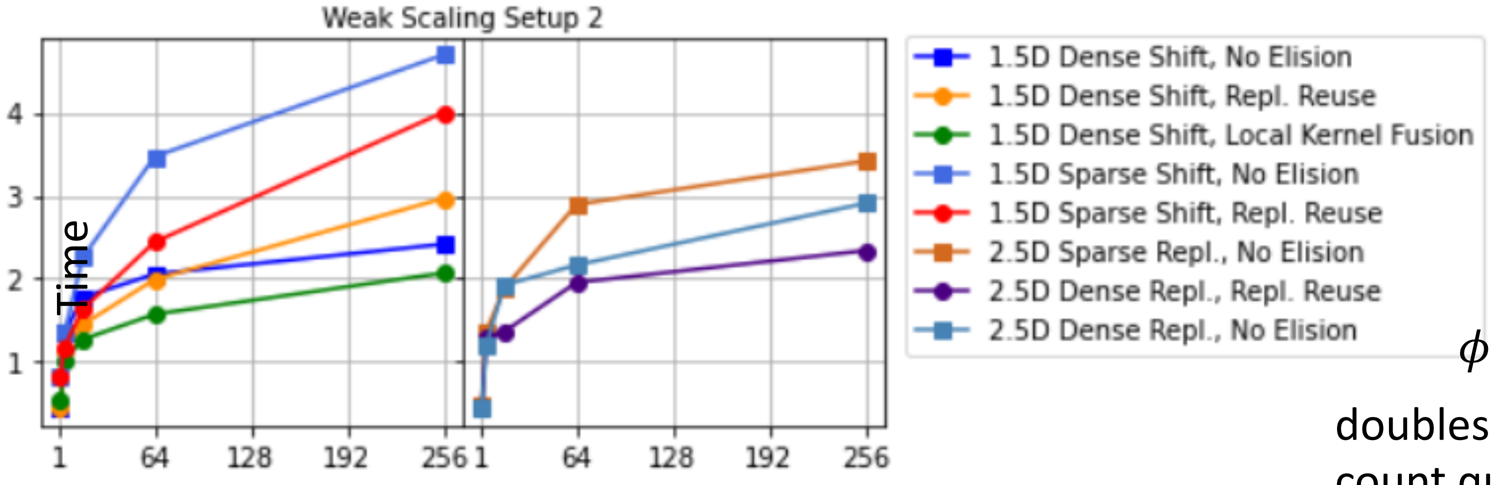
The optimal algorithm choice depends on the ratio between the **nonzero count of the sparse matrix** and the **total entries in either dense matrix**.

Distributed FusedMM performance



$$\phi = \frac{\mathrm{nnz}(S)}{nr}$$

remains constant



$$\phi = \frac{\mathrm{nnz}(S)}{nr}$$

doubles at each process count quadrupling

GraphBLAS C API Spec (http://graphblas.org)

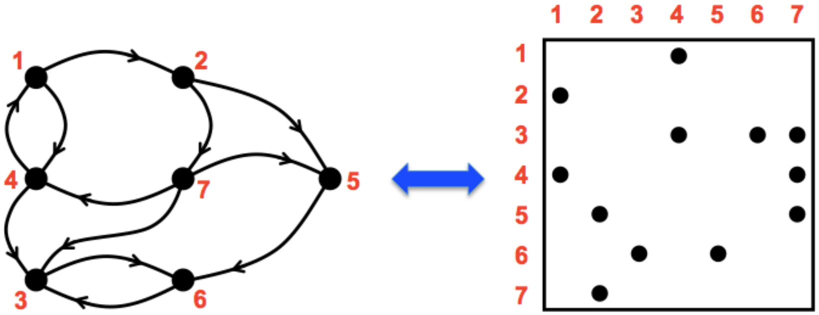
- **Goal:** A crucial piece of the GraphBLAS effort is to translate the mathematical specification to an actual Application Programming Interface (API) that
  - i. is faithful to the mathematics as much as possible, and
  - ii. enables efficient implementations on modern hardware.
- **Impact:** All graph and machine learning algorithms that can be expressed in the language of linear algebra
- **Innovation:** Function signatures (e.g. mxm, vxm, assign, extract), parallelism constructs (blocking v. non-blocking), fundamental objects (masks, matrices, vectors, descriptors), a hierarchy of algebras (functions, monoids, and semiring)

```
GrB_info GrB_mxm(GrB_Matrix               *C,      // destination
                 const GrB_Matrix          Mask,
                 const GrB_BinaryOp        accum,
                 const GrB_Semiring        op,
                 const GrB_Matrix          A,
                 const GrB_Matrix          B
              [, const Descriptor          desc]);
```

$$C(\neg M) \oplus= A^{T} \oplus.\otimes B^{T}$$

A.Buluç, T. Mattson, S. McMillan, J. Moreira, C. Yang. “The GraphBLAS C API Specification”, version 1.3.0

Combinatorial BLAS (historical slide)

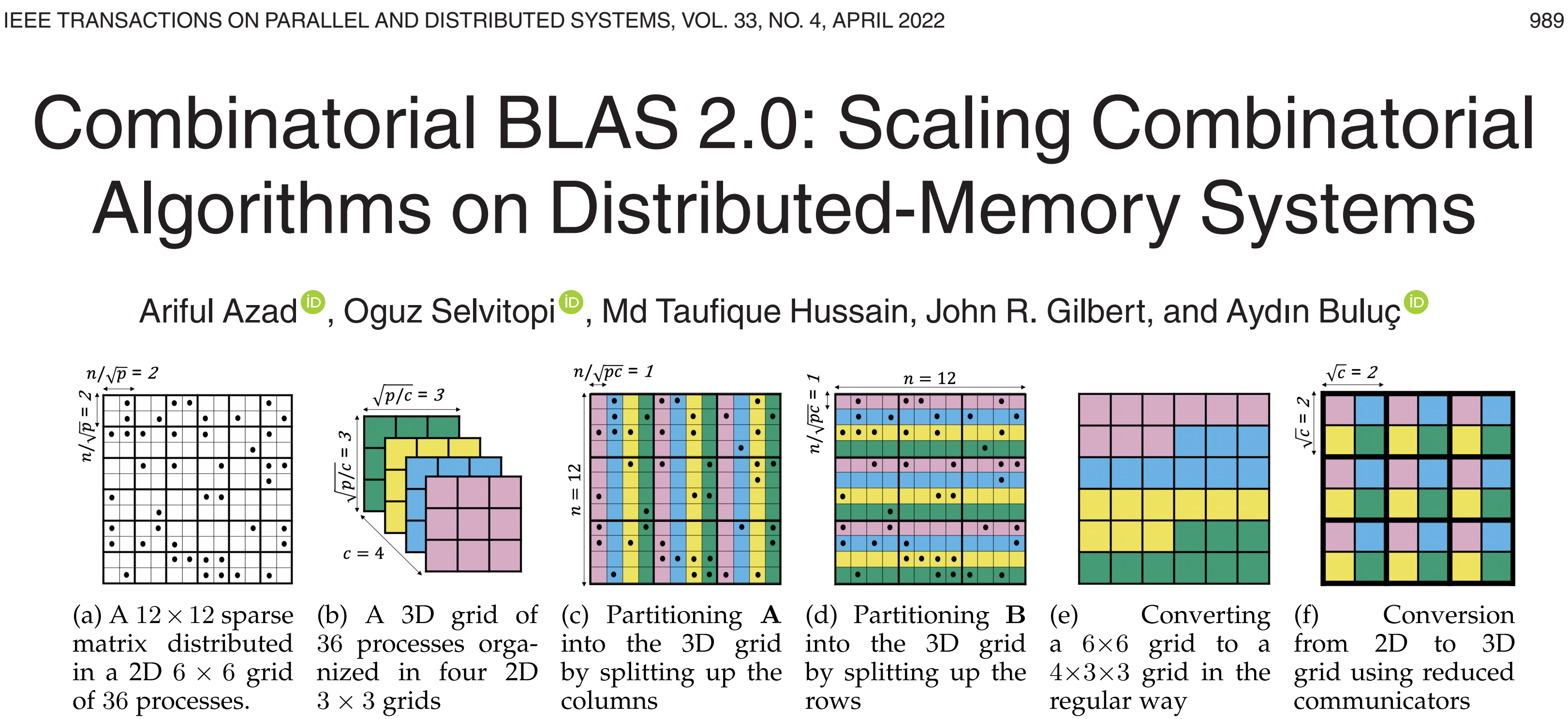


	1	2	3	4	5	6	7
1				●			
2	●						
3				●		●	●
4	●						●
5		●					●
6			●		●		
7		●					

An extensible distributed-memory library offering a small but powerful set of linear algebraic operations specifically targeting graph analytics.

- Aimed at graph algorithm designers/programmers who are not expert in mapping algorithms to parallel hardware.
- Flexible templated C++ interface; 2D data decomposition
- Scalable performance from laptop to 100,000-processor HPC.
- Open source software (v1.4.0 released January, 2014)

Combinatorial BLAS 2.0 innovations



IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 33, NO. 4, APRIL 2022 989

Combinatorial BLAS 2.0: Scaling Combinatorial Algorithms on Distributed-Memory Systems

Ariful Azad, Oguz Selvitopi, Md Taufique Hussain, John R. Gilbert, and Aydın Buluç

(a) A 12 × 12 sparse matrix distributed in a 2D 6 × 6 grid of 36 processes.

(b) A 3D grid of 36 processes organized in four 2D 3 × 3 grids

(c) Partitioning **A** into the 3D grid by splitting up the columns

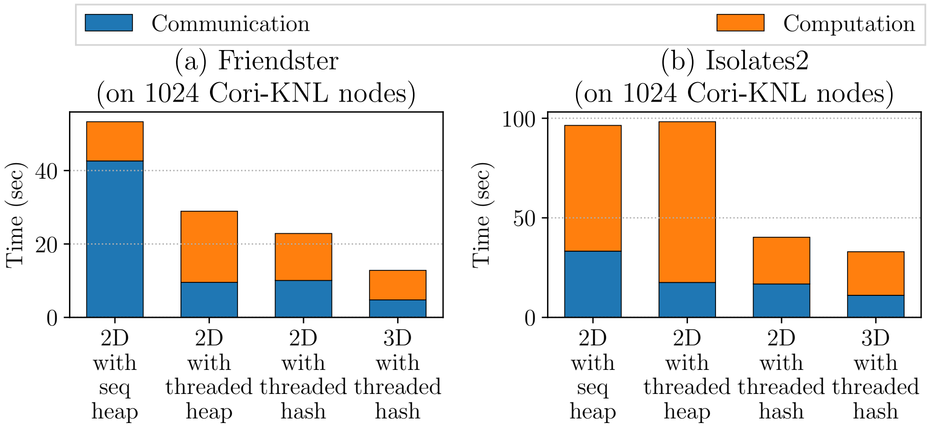
(d) Partitioning **B** into the 3D grid by splitting up the rows

(e) Converting a 6×6 grid to a 4×3×3 grid in the regular way

(f) Conversion from 2D to 3D grid using reduced communicators

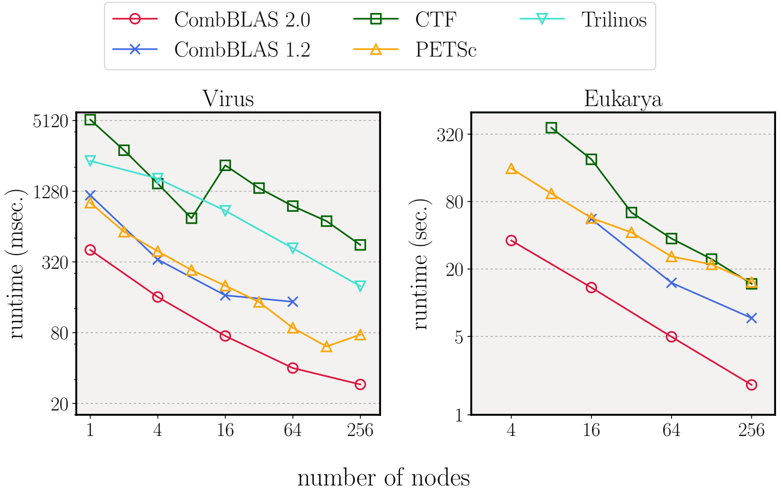
- communication avoiding algorithms,
- hierarchical parallelism via in-node multithreading,
- accelerator support via GPU kernels,
- generalized semiring support,
- implementations of key data structures and functions,
- scalable distributed I/O operations for human-readable files

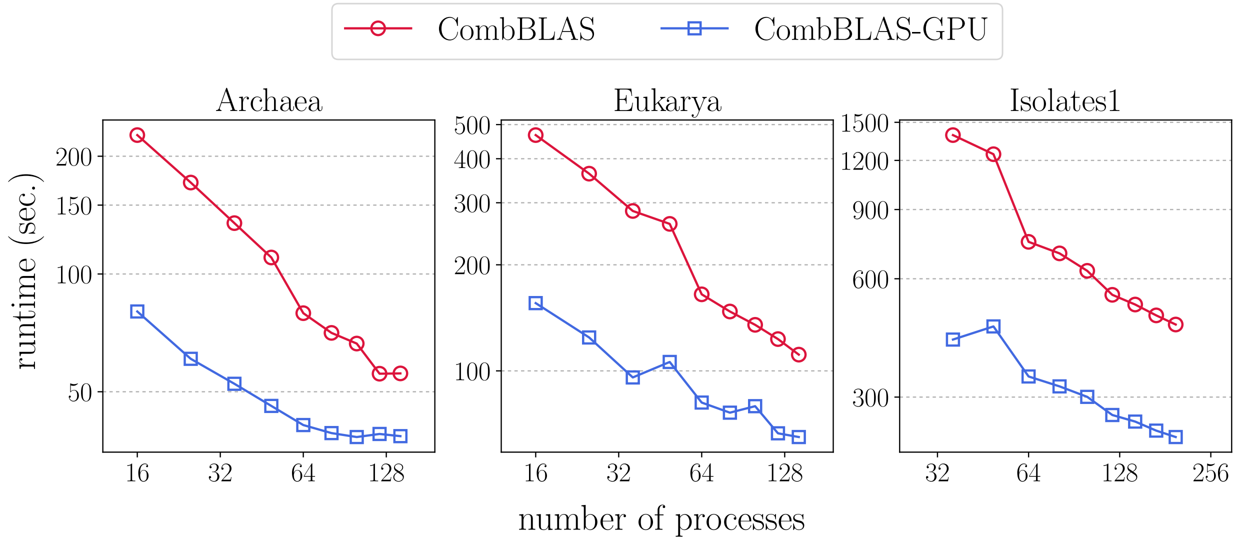
Combinatorial BLAS 2.0 performance



Distributed SpGEMM performance evolution

Parallel SpGEMM runtime of CombBLAS 1.0, 2.0, and other popular parallel sparse linear algebra libraries





Impact of GPU-enabled and disabled CombBLAS backends for HipMCL

GraphBLAST

- First “high-performance” GraphBLAS implementation on the GPU
- Optimized to take advantage of both input and output sparsity
- Automatic direction-optimization through the use of masks
- Competitive with fastest GPU (Gunrock) and CPU (Ligra) codes
- Outperforms multithreaded SuiteSparse::GraphBLAS

Design principles:

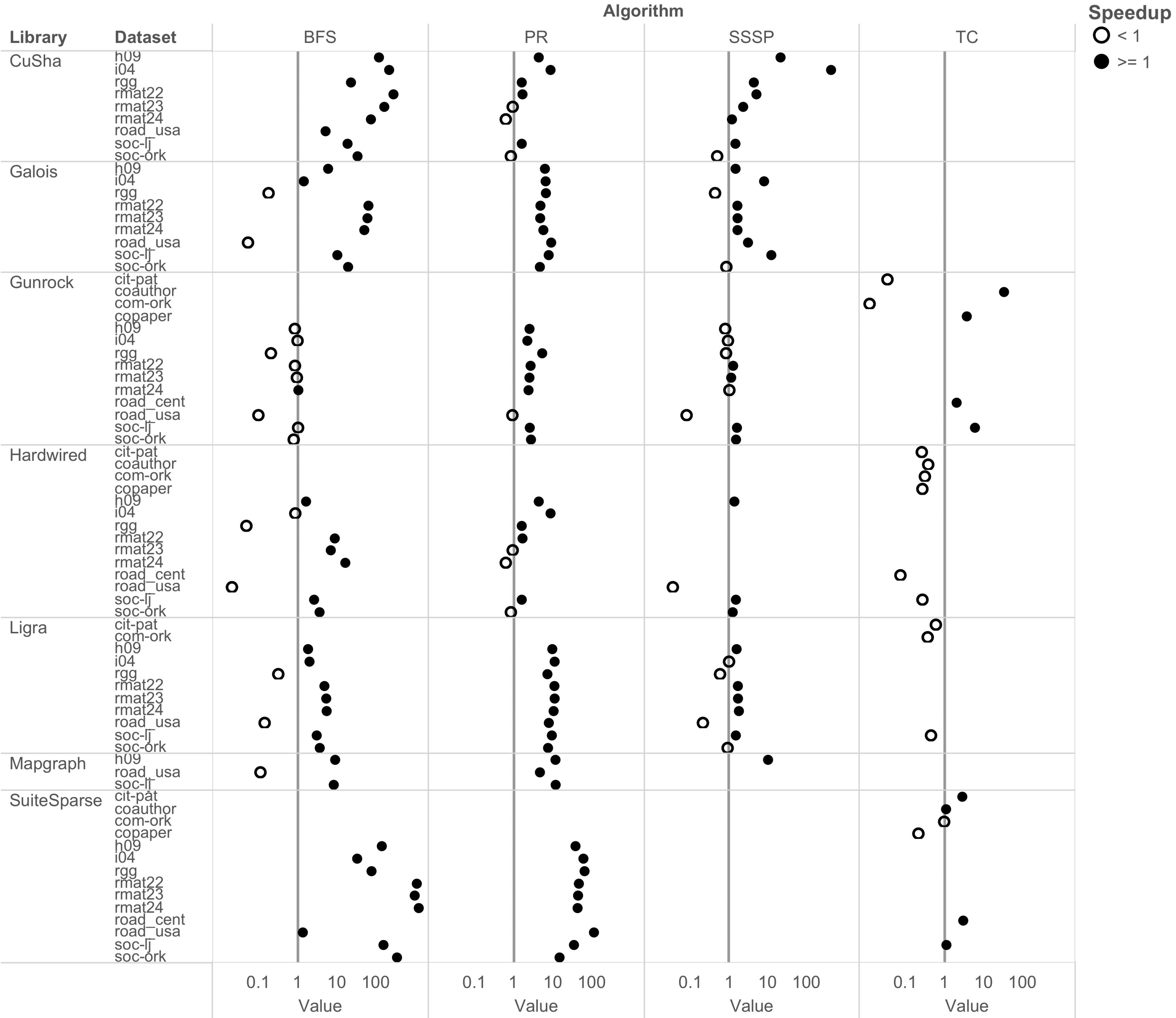
1. Exploit input sparsity => direction-optimization
2. Exploit output sparsity => masking
3. Proper load-balancing => key for GPU implementations

Extensively evaluated on (more implemented, google for github repo)

- Breadth-first-search (BFS)
- Single-source shortest-path (SSSP)
- PageRank (PR)
- Triangle counting (TC)

https://github.com/gunrock/graphblast

Yang, Buluc, Owens, “GraphBLAST: A High-Performance Linear Algebra-based Graph Framework on the GPU”, ACM Transactions on Mathematical Software (TOMS), 2022


Algorithm
Library
Dataset
BFS
PR
SSSP
TC
Speedup
< 1
>= 1
CuSha
h09
i04
rgg
rmat22
rmat23
rmat24
road_usa
soc-lj
soc-ork
Galois
h09
i04
rgg
rmat22
rmat23
rmat24
road_usa
soc-lj
soc-ork
Gunrock
cit-pat
coauthor
com-ork
copaper
h09
i04
rgg
rmat22
rmat23
rmat24
road_cent
road_usa
soc-lj
soc-ork
Hardwired
cit-pat
coauthor
com-ork
copaper
h09
i04
rgg
rmat22
rmat23
rmat24
road_cent
road_usa
soc-lj
soc-ork
Ligra
cit-pat
com-ork
h09
i04
rgg
rmat22
rmat23
rmat24
road_usa
soc-lj
soc-ork
Mapgraph
h09
road_usa
soc-lj
SuiteSparse
cit-pat
coauthor
com-ork
copaper
h09
i04
rgg
rmat22
rmat23
rmat24
road_cent
road_usa
soc-lj
soc-ork
0.1
1
10
100
Value

Conclusions

- Sparse matrix techniques underlie computations from disparate fields:
  a. Scientific computing
  b. Machine learning
  c. Graph analysis
  d. Bioinformatics
- GraphBLAS already seem to have the right abstraction with its flexible **masks** and **semirings** to be the default backend of many of these computations
- Extreme parallelism and data, and hence **the need for distributed memory parallelism** is here to stay and will get worse
- **Communication-avoiding algorithms, and novel data structures for sparse matrices** will be the key to overcome these adverse technological trends

Acknowledgments

Ariful Azad, Vivek Bharadwaj, Ben Brock, Zoran Budimlić, Tim Davis, James Demmel, Saliya Ekanayake, Marquita Ellis, John Gilbert, Giulia Guidi, Md Taufique Hussain, Jeremy Kepner, Nikos Krypides, Tim Mattson, Scott McMillan, Srđan Milaković, Jose Moreira, Israt Nisa, John Owens, Georgios Pavlopoulos, Doru Popovici, Dan Rokhsar, Oguz Selvitopi, Yu-Hang Tang, Alok Tripathy, Carl Yang, Kathy Yelick.

Our Research Team: http://passion.lbl.gov

Our work is funded by U.S. DEPARTMENT OF ENERGY | Office of Science