Berkeley
UNIVERSITY OF CALIFORNIA

Sparse Matrices in Biology and Machine Learning

Aydın Buluç
Lawrence Berkeley National Laboratory & UC Berkeley
ACDA Workshop, Aussois, France
Sep 5, 2022

PASSION Lab People

http://passion.lbl.gov

Aydın Buluç (Principal Investigator)

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

Undergraduate researchers:

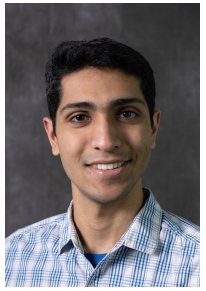
- Richard Lettich
- Ujjaini Mukhopadhyay

UC Berkeley PhD Students (co-advised)

Ben Brock

Giulia Guidi

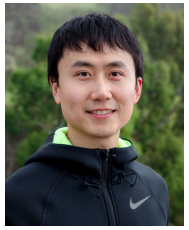
Alok Tripathy



Vivek Bharadwaj

LBNL Research Scientists, Engineers, Postdocs, Visiting Fellows

Oguz Selvitopi



Yu-Hang Tang

Can Kizilkale

Helen Xu

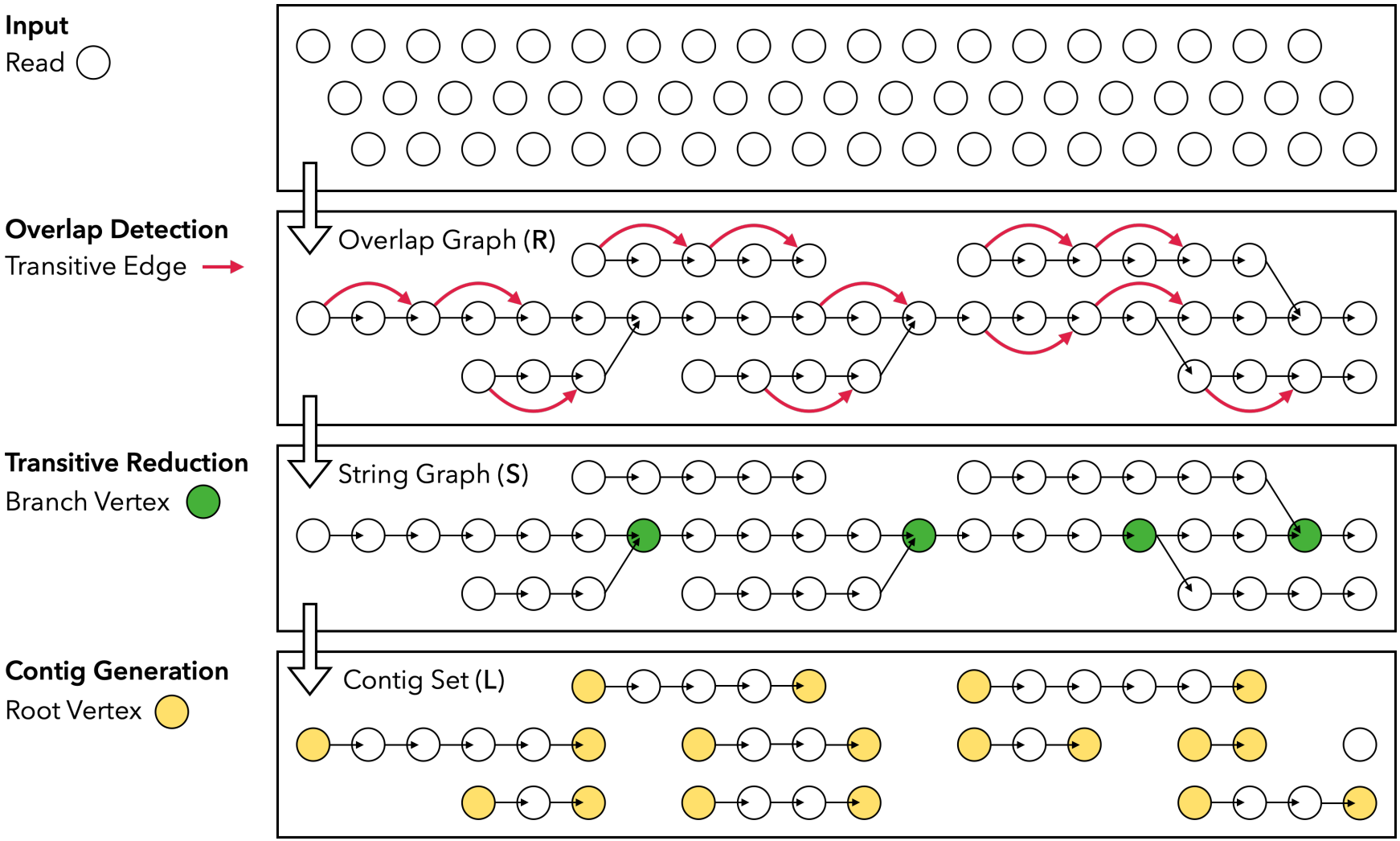
Gabriel Raulet

Koby Hayashi

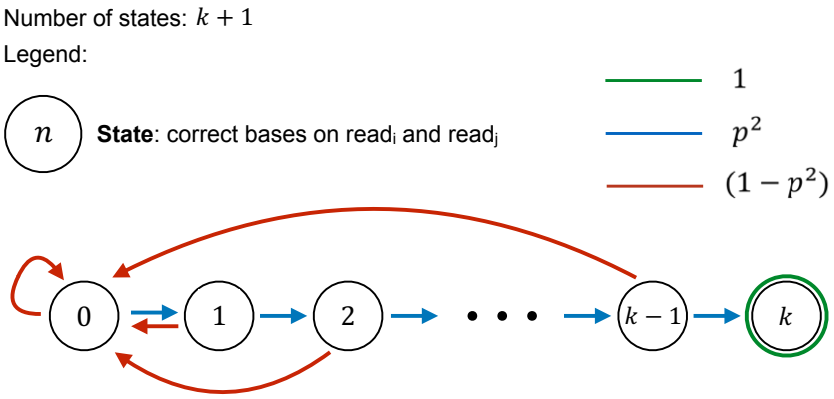
High-level outline

- **Sparse matrices for computational biology**
- Sparse matrices for machine learning

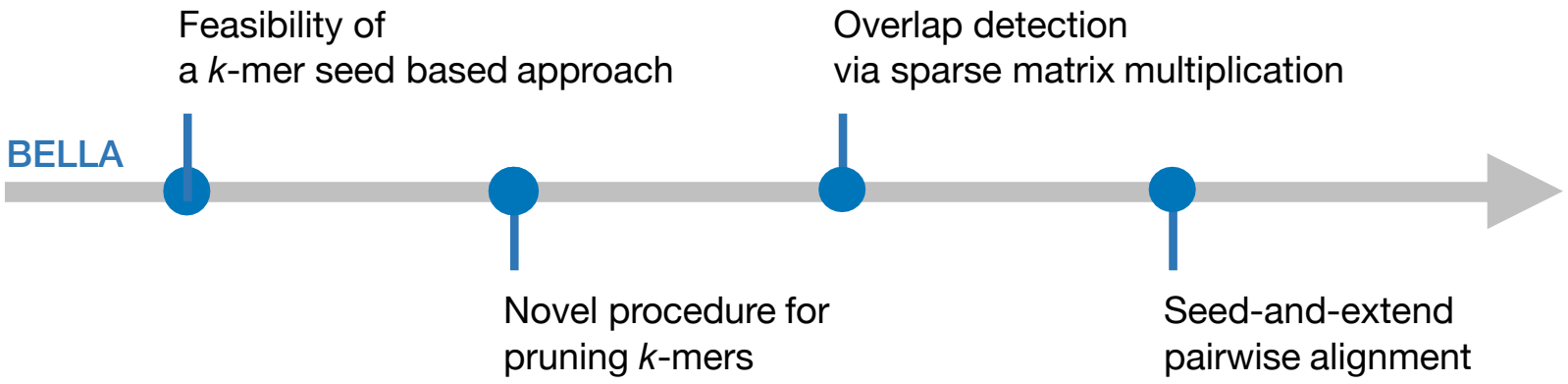
Genome assembly pipeline



BELLA: Berkeley Long-read to Long-read Aligner and Overlapper



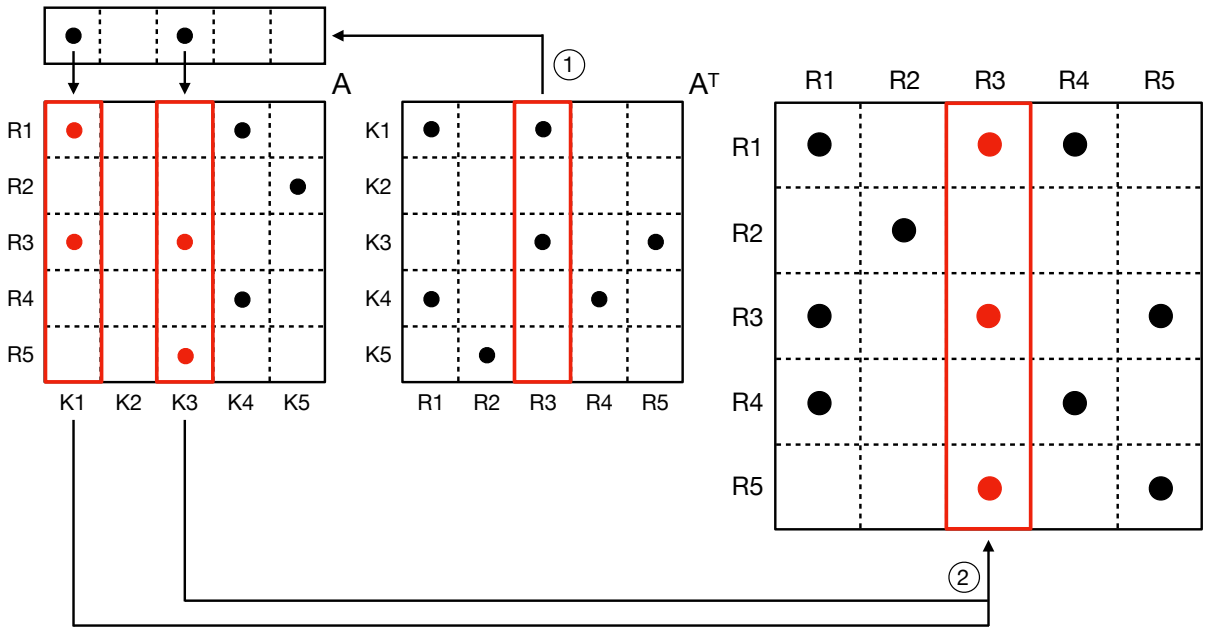
- How to choose the right set of k-mers, otherwise there are too many of them?
- How to use alignment score to tell true alignments from false positives?



Guidi G, Ellis M, Rokhsar D, Yelick K, Buluç A. BELLA: Berkeley Efficient Long-Read to Long-Read Aligner and Overlapper. SIAM Conference on Applied and Computational Discrete Algorithms (ACDA), 2021

SpGEMM use case #1: read overlapping

- **Overlapping** is the most computationally expensive step in the overwhelming majority of long read assemblers.
- Imagine each read is a sample, its k-mer profile is its feature set
- Create a reads-by-kmers (sparse) matrix

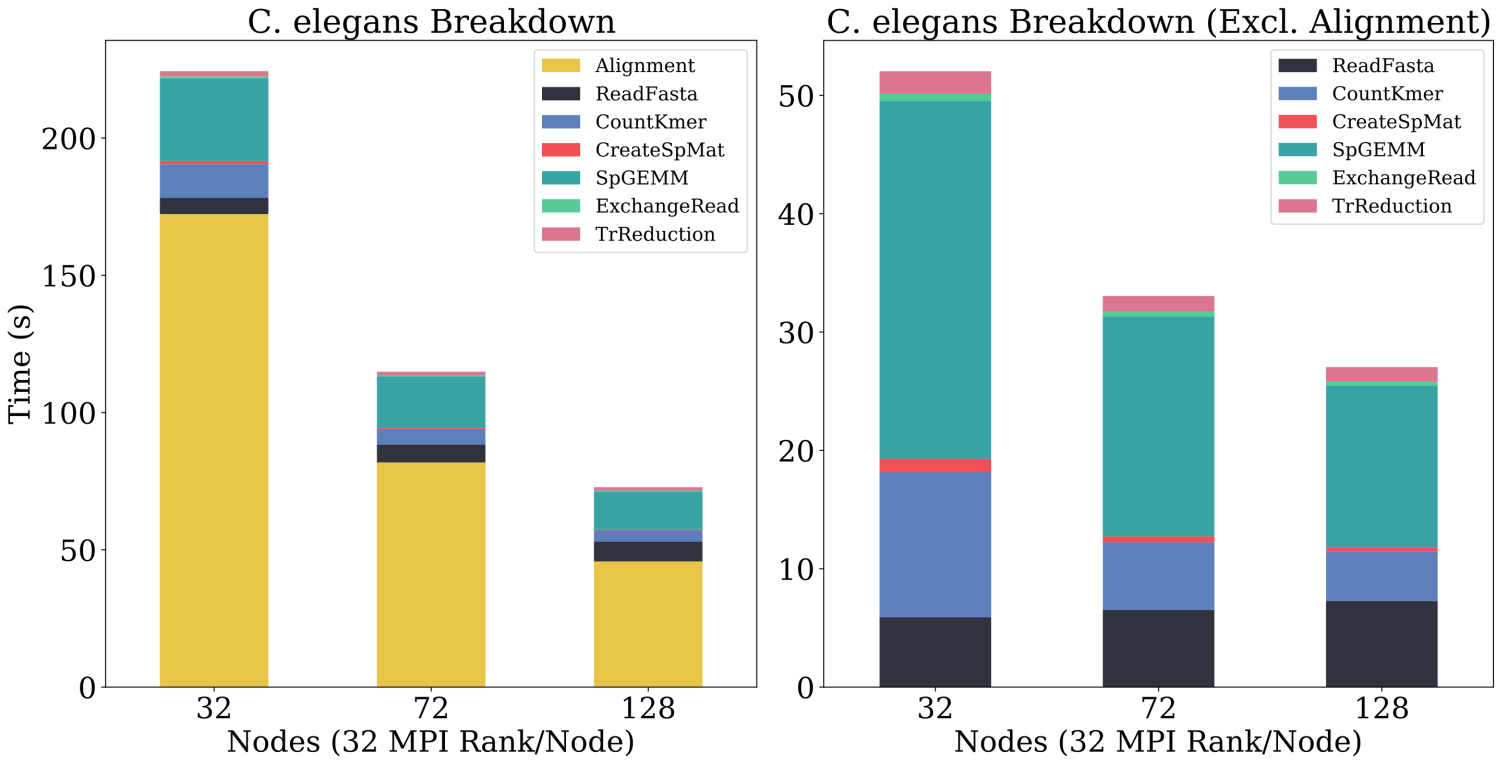


$AA^T(i,j)$ = # shared k-mers between reads i and j, plus their positions in the reads

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

diBELLA.2D performance results

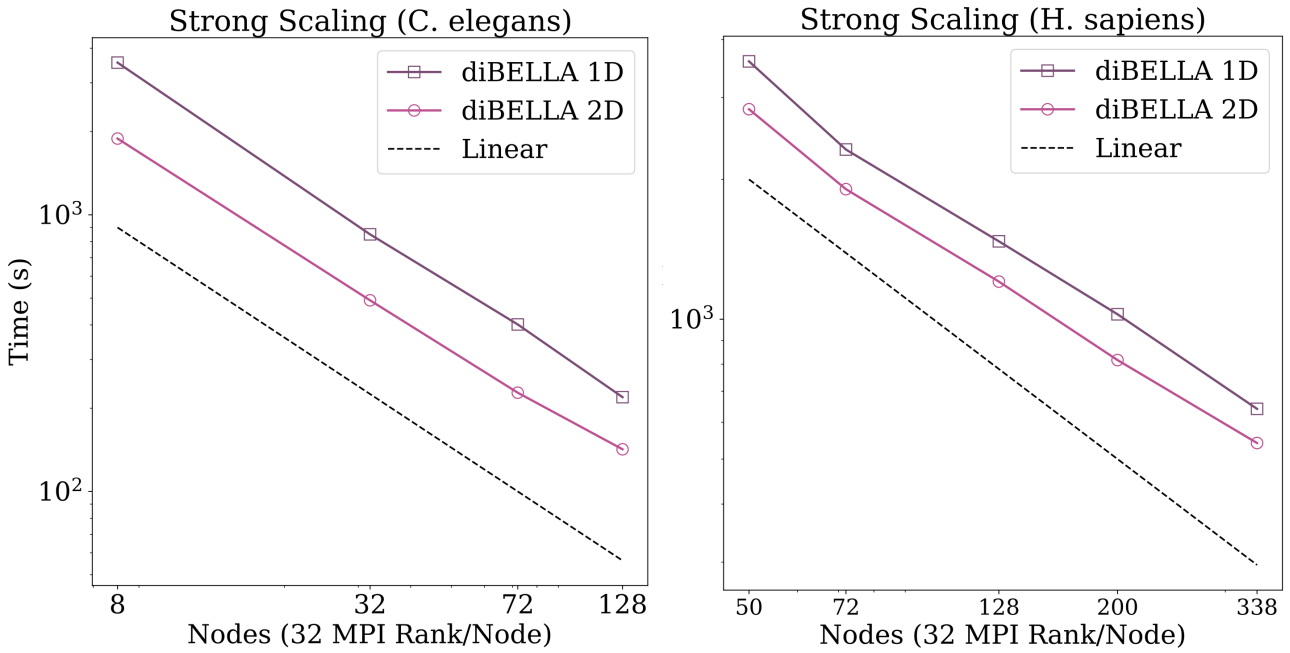
diBELLA.2D: distributed-memory version of BELLA on 2D process grid
Performs *overlap detection* plus *transitive reduction (overlap to string graph)*
https://github.com/PASSIONLab/diBELLA.2D



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?

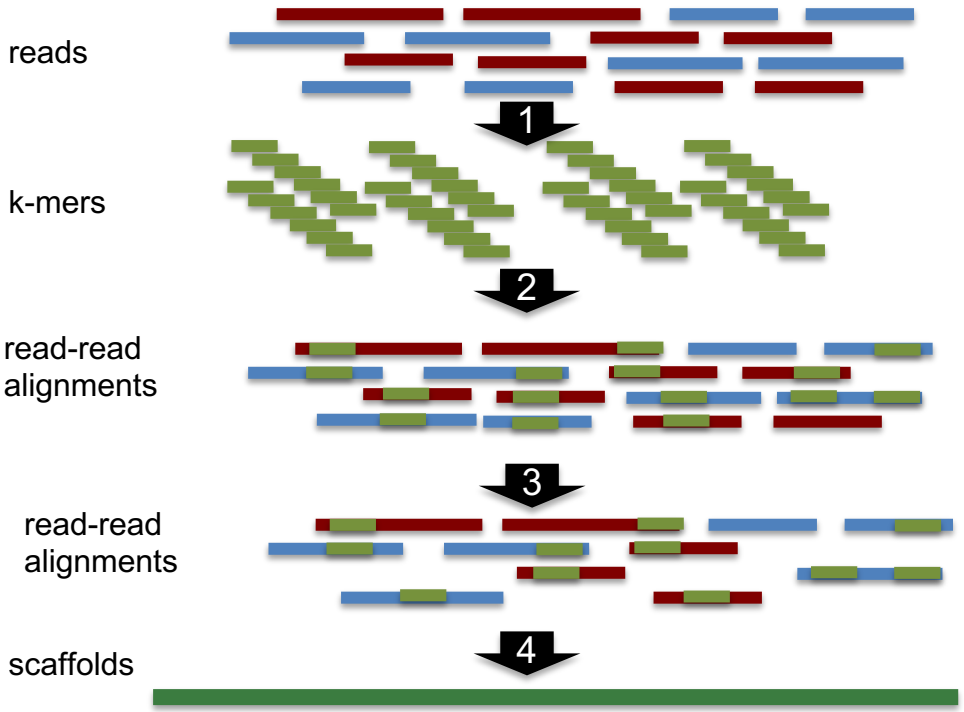
- Comparing the sparse matrix abstraction (diBELLA 2D [2], **weeks** of effort) with a direct implementation (diBELLA 1D [1], **years** of effort). Both use MPI
- Sparse matrices reduce communication via 2D sparse SpGEMM



[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

Sparse matrix approach for assembly with long reads



1) K-mer Analysis
K-mer histogram

2) Sparse matrix building
A: reads-by-kmers

3) Overlapping via SpGEMM C
= AA^T : reads-by-reads

4) X-drop alignments
M = filter(C, alignment_score)

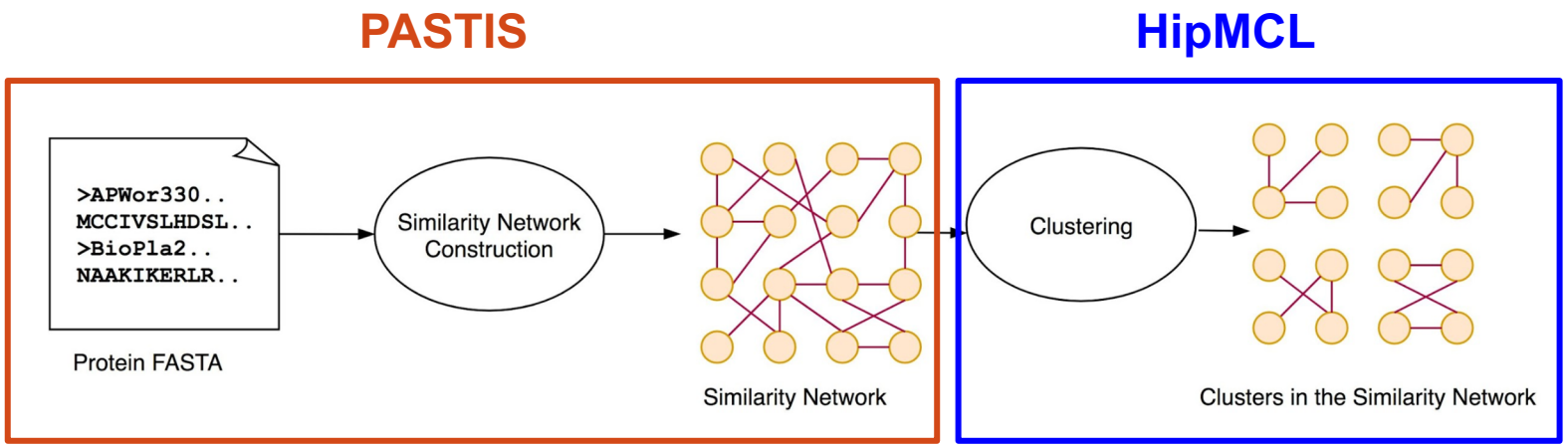
5) Transitive Reduction
$M^{i+1} = \text{Prune}(M^i M^i \odot M^i)$

6) Contig generation [3]
- Remove forks
- Find connected components (CCs)
- Local traversal of CCs

[3] Giulia Guidi, Gabriel Raulet, Daniel Rokhsar, Leonid Oliker, Katherine Yelick, and Aydın Buluç. Distributed-memory parallel contig generation for de novo long-read genome assembly. In ICPP, 2022

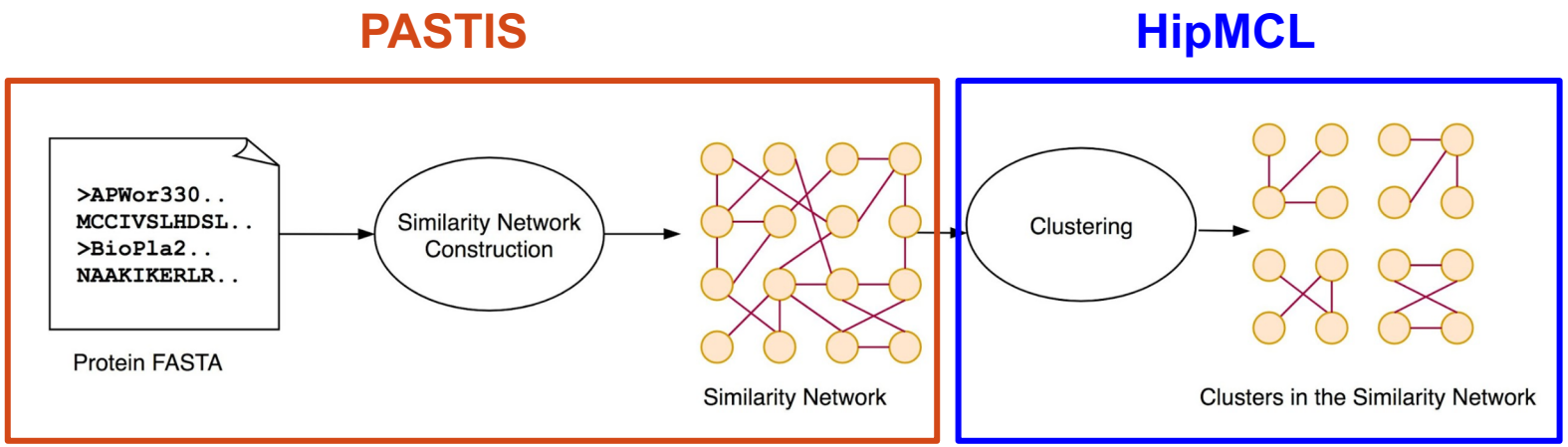
Protein Family Identification

- Problem: Given a large collection of proteins, identify groups of proteins that are homologous (i.e. descended from a common ancestor).
- Homologous proteins often have the same function (think of different variants of hemoglobin in many species)
- Often, only sequences (and not structure) of the proteins are available, so we infer homology via sequence similarity



Protein Family Identification

- Many one-step approaches are possible that trade accuracy for lower memory consumption and faster execution (e.g., CD-HIT).
- The approach that seems to lead to highest accuracy:
  - Construct a similarity network over protein sequences using many-to-many sequence search (PASTIS)
  - Cluster this network to discover possible protein families (HipMCL)



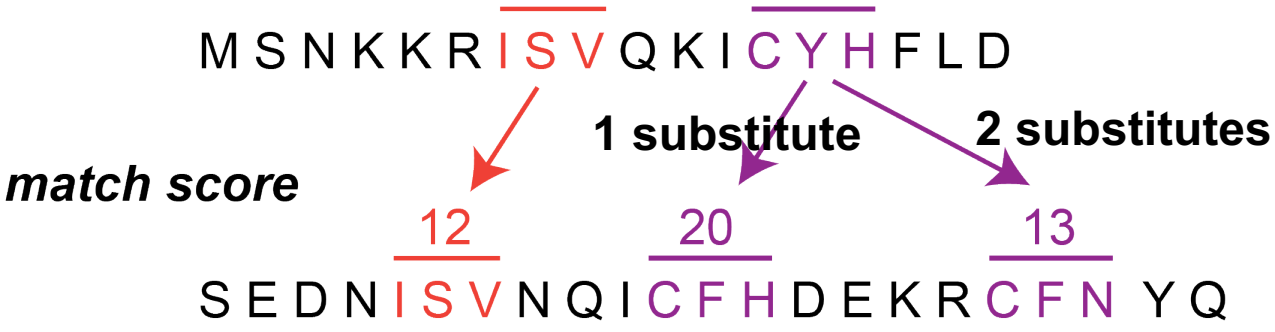
SpGEMM use case #2: many-to-many protein alignment

- 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



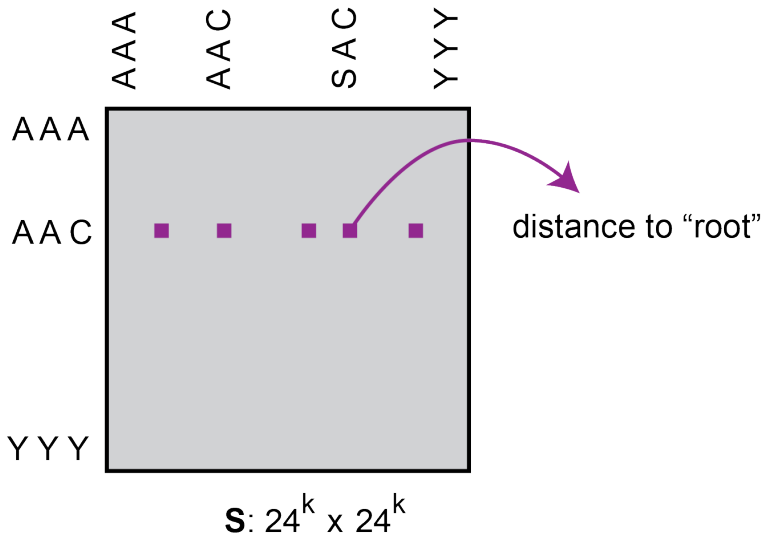
SpGEMM for many-to-many protein alignment

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**



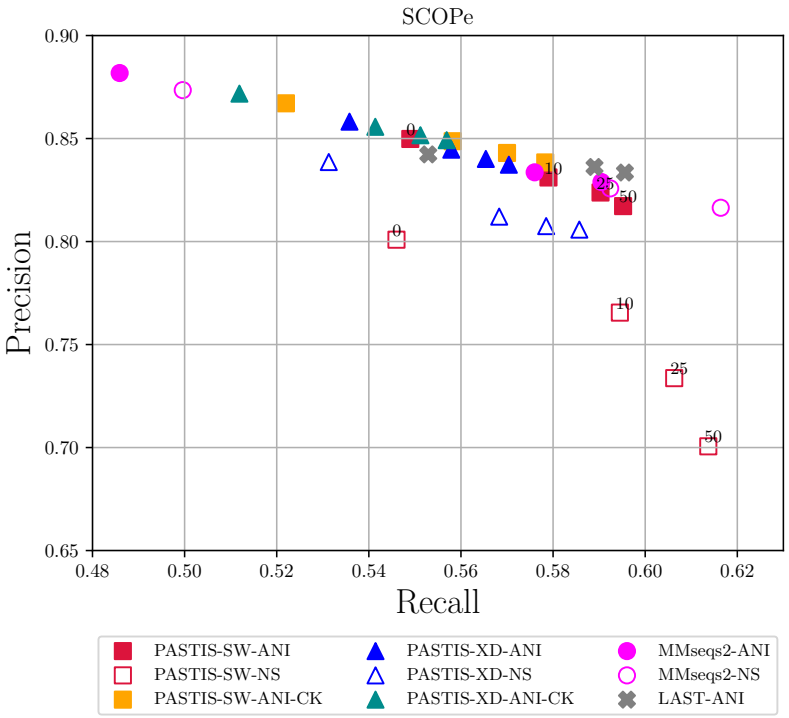
Exact k-mers → $C=AA^T$

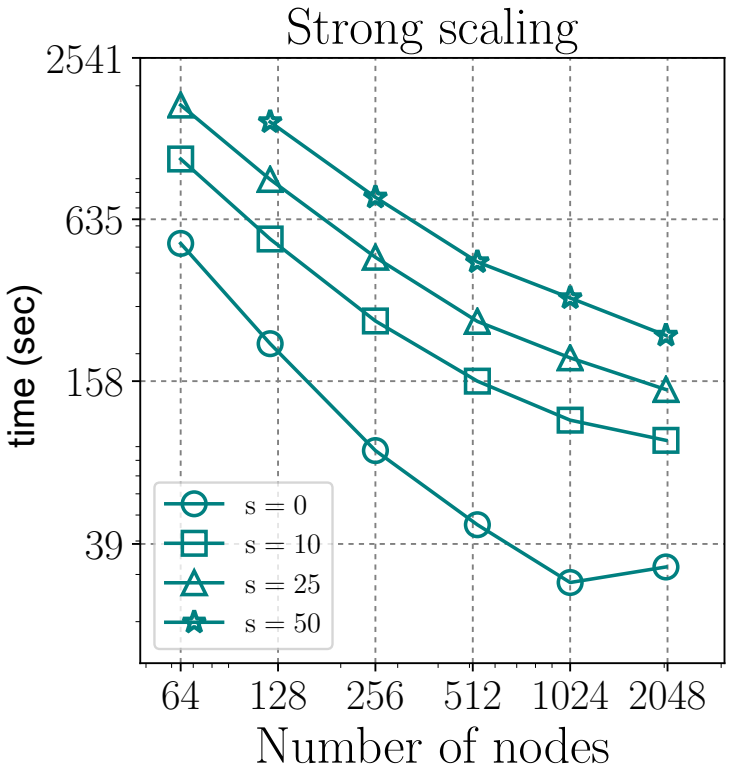
Substitute k-mers → $C=ASA^T$

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

PASTIS in 2022

Hot off the press: Finalist for the 2022 ACM Gordon Bell Prize
https://en.wikipedia.org/wiki/Gordon_Bell_Prize

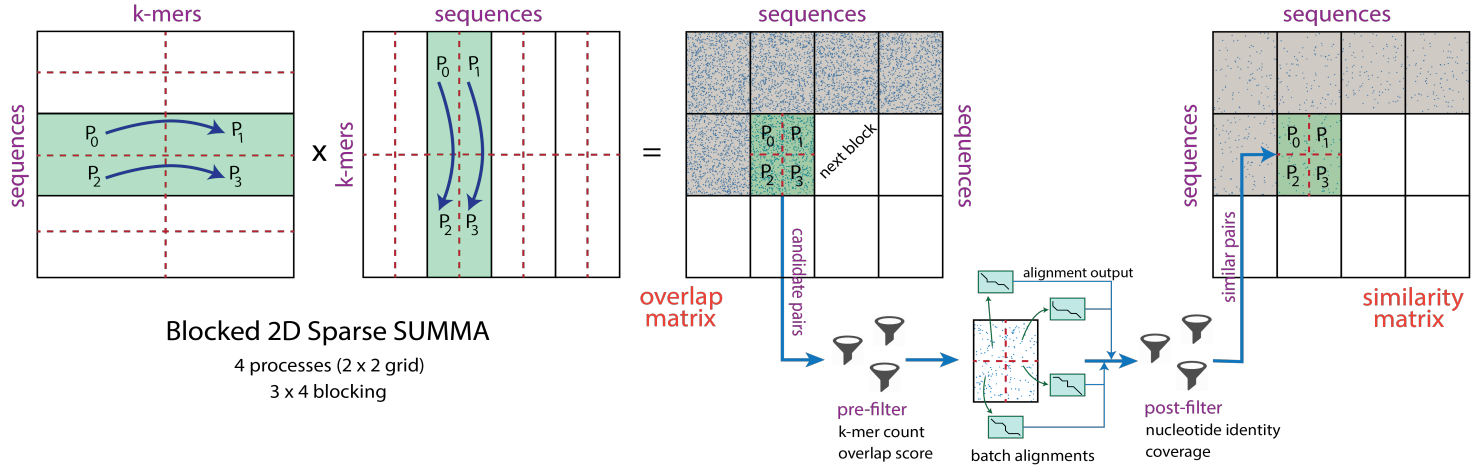
Extreme-scale many-against-many protein similarity search

Oguz Selvitopi*, Saliya Ekanayake†, Giulia Guidi‡, Muaaz G. Awan§, Georgios A. Pavlopoulos¶, Ariful Azad‖, Nikos Kyrpides**, Leonid Oliker*, Katherine Yelick‡*, Aydın Buluç*‡

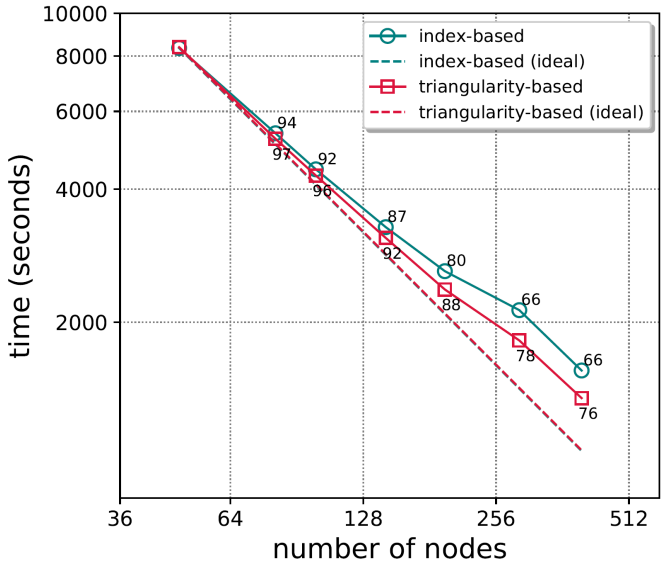
*Applied Mathematics & Computational Research Division, Lawrence Berkeley National Laboratory, USA
†Microsoft Corporation, USA
‡University of California, Berkeley, USA
§NERSC, Lawrence Berkeley National Laboratory, USA
¶Institute for Fundamental Biomedical Research, BSRC "Alexander Fleming", 34 Fleming Street, 16672, Vari, Greece
‖Indiana University, USA
**Joint Genome Institute, Lawrence Berkeley National Laboratory, USA
roselvitopi@lbl.gov

Abstract-- … We unleash the power of over 12,000 GPUs to perform all-vs-all protein similarity search on one of the largest publicly available datasets with 313 million proteins, in less than 4 hours, cutting the time-to-solution for many use cases from weeks. The variability of protein sequence lengths, as well as the sparsity of the space of pairwise comparisons, make this a challenging problem in distributed memory. …

Extreme-scale many-against-many protein similarity search



The ExaGraph and ExaBiome teams of ECP were selected as a Gordon Bell Award Finalist for their joint work entitled “Extreme-scale many-against-many protein similarity search”. The paper describes significant improvements to the accuracy and scale of protein sequence similarity search capabilities developed under ECP. Thanks to memory-consumption optimizations, new parallel algorithms taking advantage of the symmetry in the sequence similarity matrix, GPU acceleration, and the ability to address load imbalance issues, they were able to perform many-against-many protein search on a dataset containing 405 million protein sequences with the PASTIS code on 3364 compute nodes of ORNL Summit Supercomputer in 3.4 hours, sustaining a rate of 691 million alignments per second and attaining ~176 TCUPs (Tera Cell Updates per second). The output, represented as a protein-protein sequence similarity graph, is an impressive 27 Terabytes. This is an improvement over the team's work from their SC'20 paper of nearly two orders of magnitude, and a significant advance over the rest of the field.



U.S. DEPARTMENT OF ENERGY | Office of Science

Extreme-scale many-against-many protein similarity search. O. Selvitopi, S. Ekanayake, G. Guidi, M. G. Awan, G. A. Pavlopoulos, A. Azad, N. Kyrpides, L. Oliker, K. Yelick, A. Buluc. SC 2022

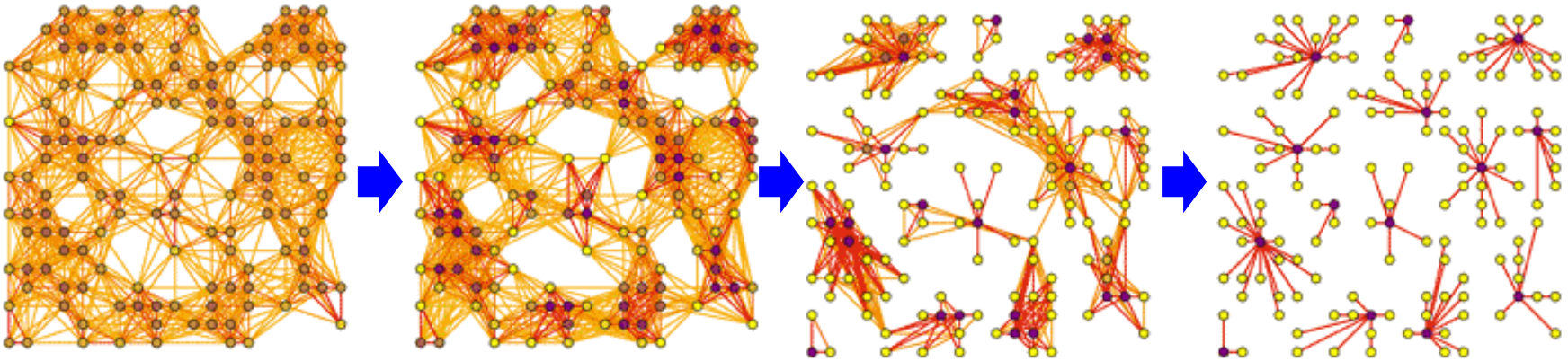
ECP EXASCALE COMPUTING PROJECT | BERKELEY LAB Lawrence Berkeley National Laboratory

SpGEMM use case #3: Markov Clustering

Markov clustering is also multi-source (in fact, all sources) traversal:
It alternates between SpGEMM and element-wise or column-wise pruning

GrB_mxm(C, GrB_NULL, <semiring>, A, A, <desc>)

A: sparse normalized adjacency matrix
C: denser (but still sparse) pre-pruned matrix for next iteration

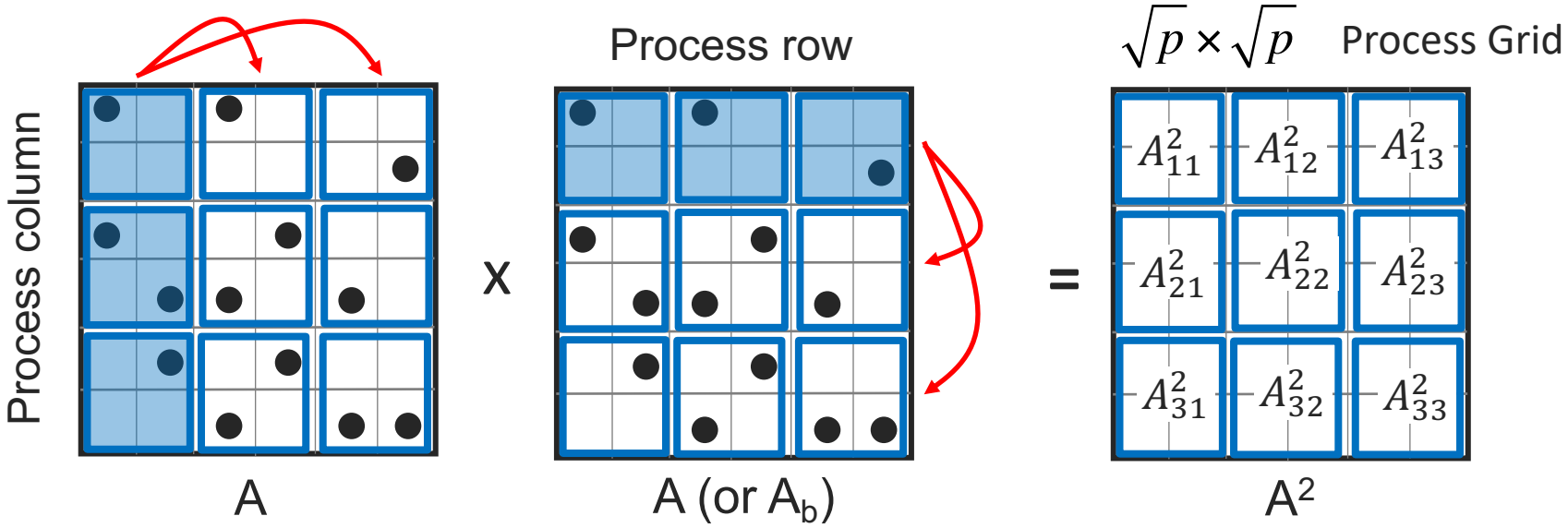


Initial network | Iteration 1 | Iteration 2 | Iteration 3

At each iteration:
Step 1 (Expansion): Squaring the matrix while pruning (a) small entries, (b) denser columns
Naïve implementation: sparse matrix-matrix product (SpGEMM), followed by column-wise top-K selection and column-wise pruning
Step 2 (Inflation) : taking powers entry-wise

HipMCL: High-performance MCL

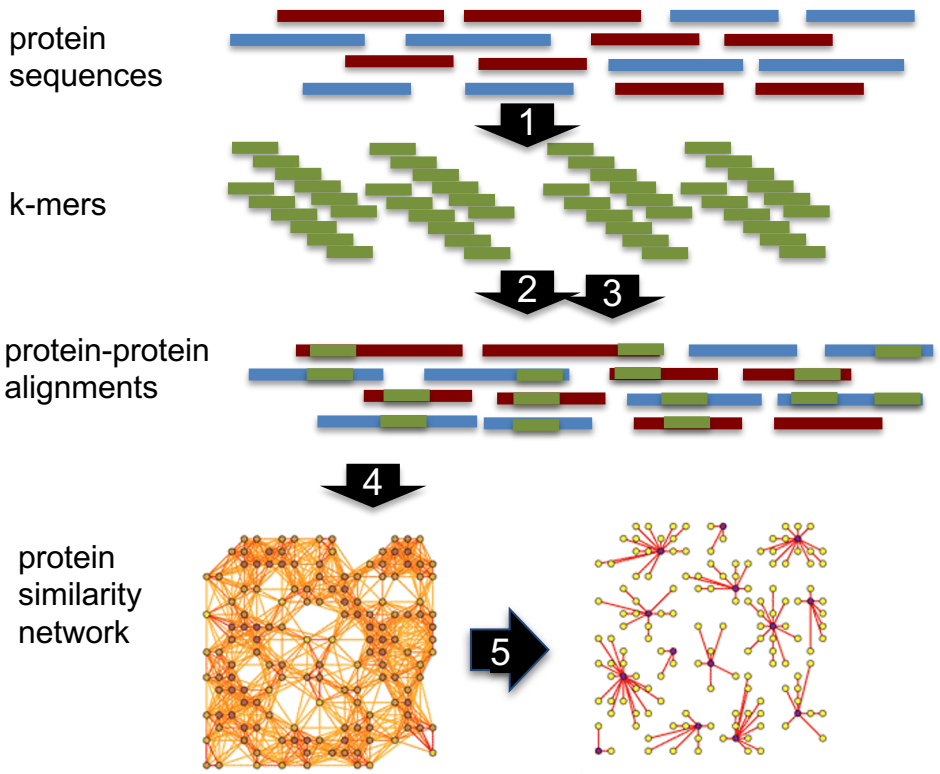
- HipMCL uses the most popular variant of Sparse SUMMA
- Both input matrices are broadcasted in stages and owners of output submatrices perform local sparse matrix multiplications
- When the number of phases increase (b decreases), A is re-broadcasted for each phase, increasing communication



A. Azad, G. Pavlopoulos, C. Ouzounis, N. Kyrpides, A. Buluç; HipMCL: a high-performance parallel implementation of the Markov clustering algorithm for large-scale networks, *Nucleic Acids Research, 2018*

Recap: Protein family identification using sparse matrices

PASTIS + HipMCL approach for protein family identification



1) K-mer Analysis
K-mer histogram

2) Sparse matrix building
A: proteins-by-kmers

3) Overlapping via SpGEMM C
$= AA^T$ (or ASA^T)

4) Pairwise alignments
M = filter(C, alignment_score)

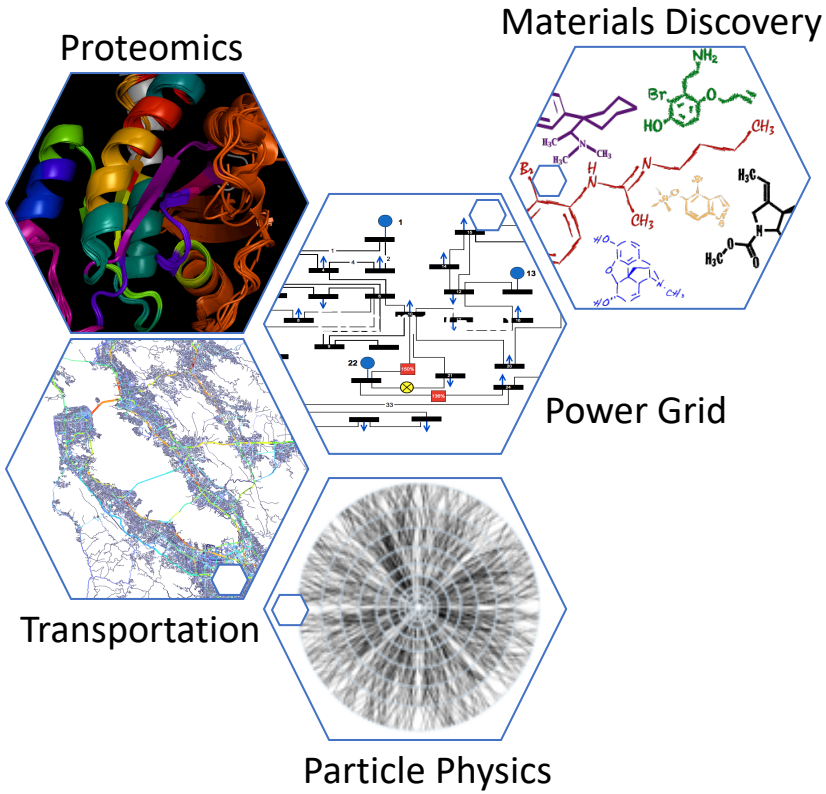
5) MCL iteration via SpGEMM
$M^{i+1} = \text{Prune}(M^i M^i)$

SpGEMM: Sparse matrix times sparse matrix

High-level outline

- Sparse matrices for computational biology
- **Sparse matrices for machine learning**

Graph Neural Networks (GNNs)



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

- Graph classification
- Edge classification
- **Node classification**

GNNs are computationally intensive to train. Distributed training need to scale to large GPU/node counts despite challenging sparsity.

What can I do with a GNN?

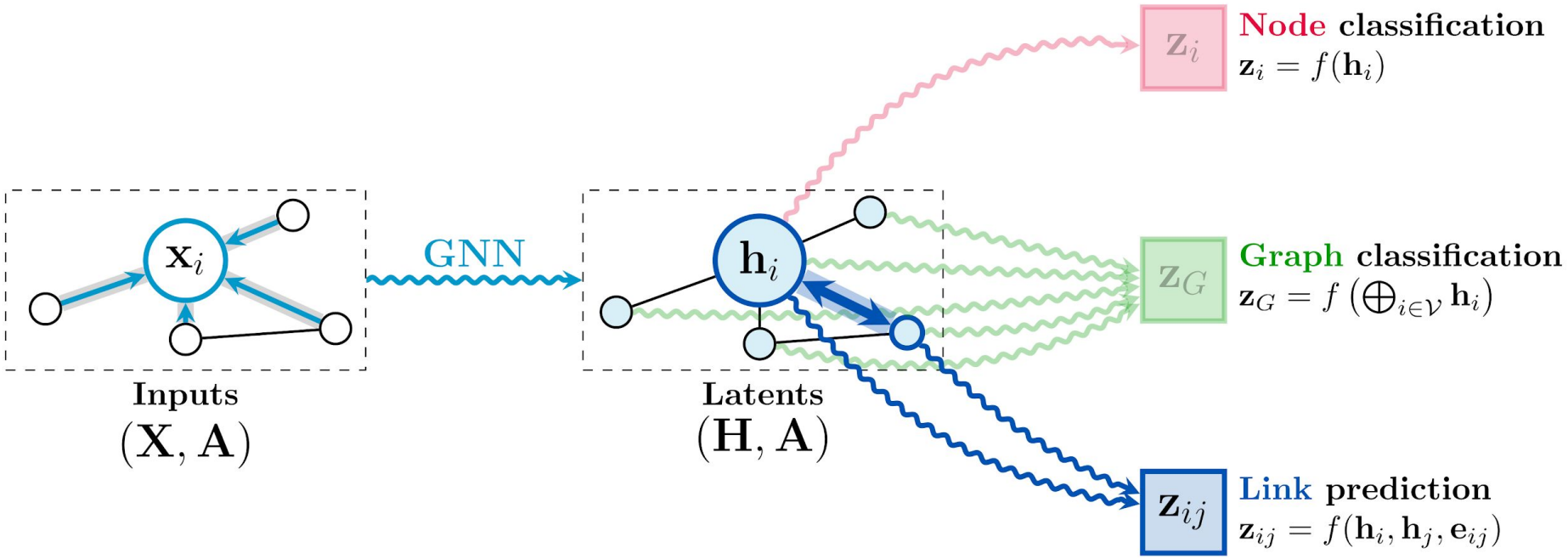


Figure source: Petar Veličković

Full-graph vs. mini-batch SGD

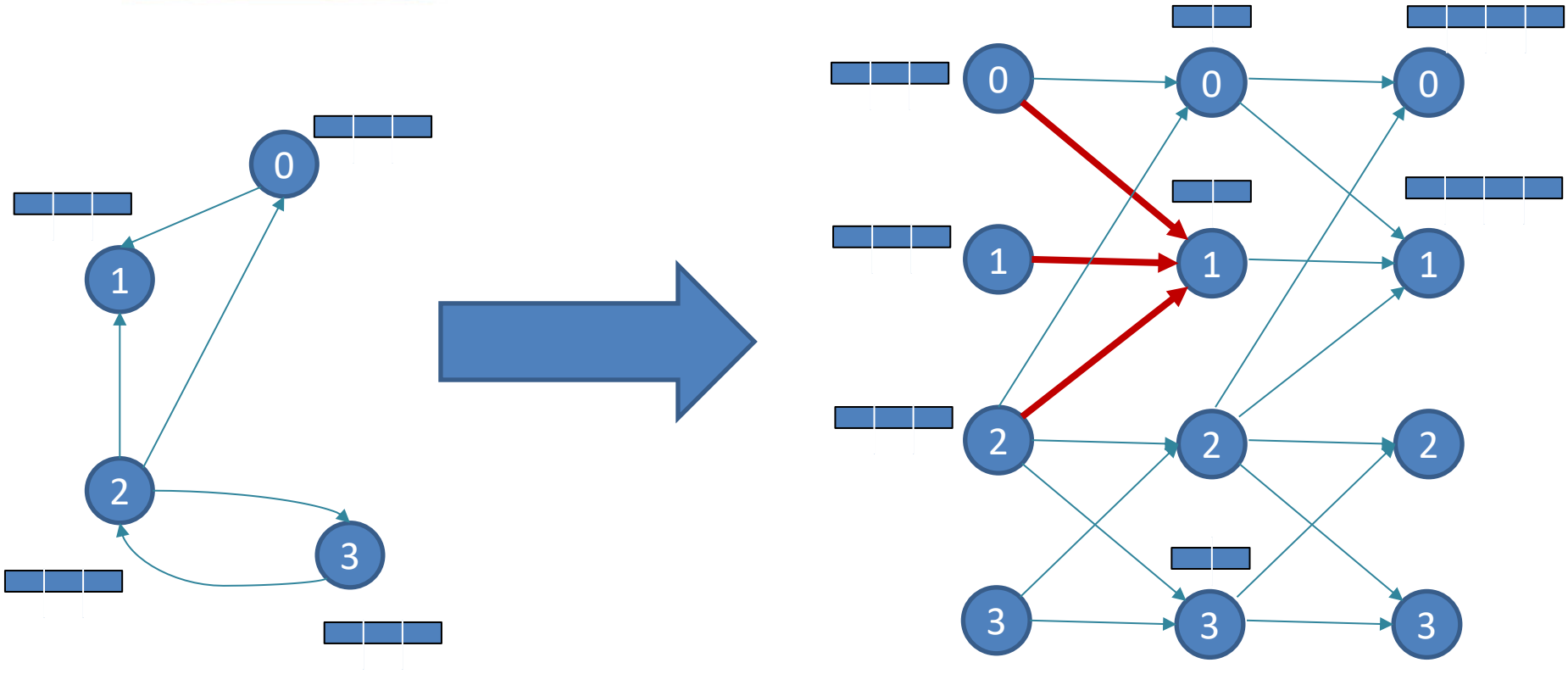
Full-graph training:

- 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

Graph convolution illustrated



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

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 = σ(Z^l)
    ...
    for l = L-1 … 1
        G^l = A * G^(l+1) * (W^(l+1))^T ⊙ σ'(Z^l)
        dH/dW = (H^(l-1))^T * A * G^l
```

$A \in n\ x\ n$

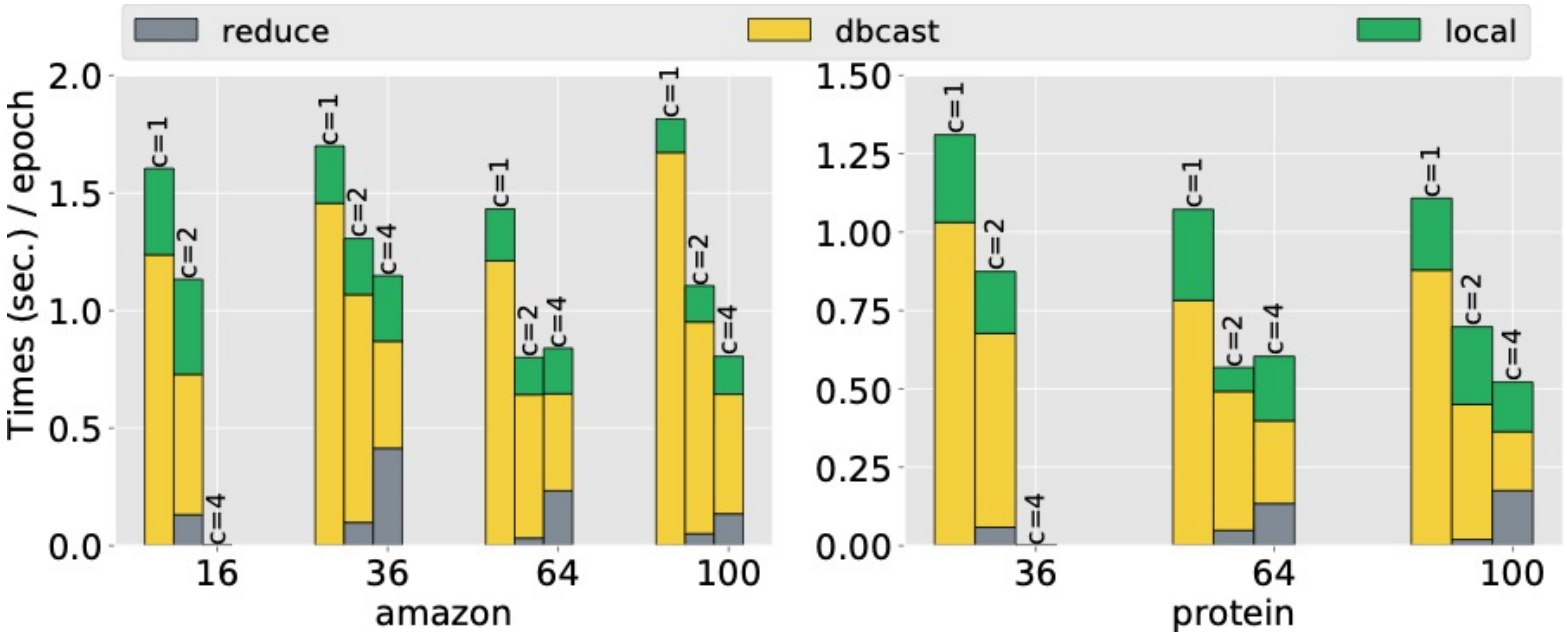
$H^l \in n\ x\ f^l$

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

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

- 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

Pattern: Sparse matrix times tall-skinny dense matrix (SpMM)

Feature aggregation from neighbors:

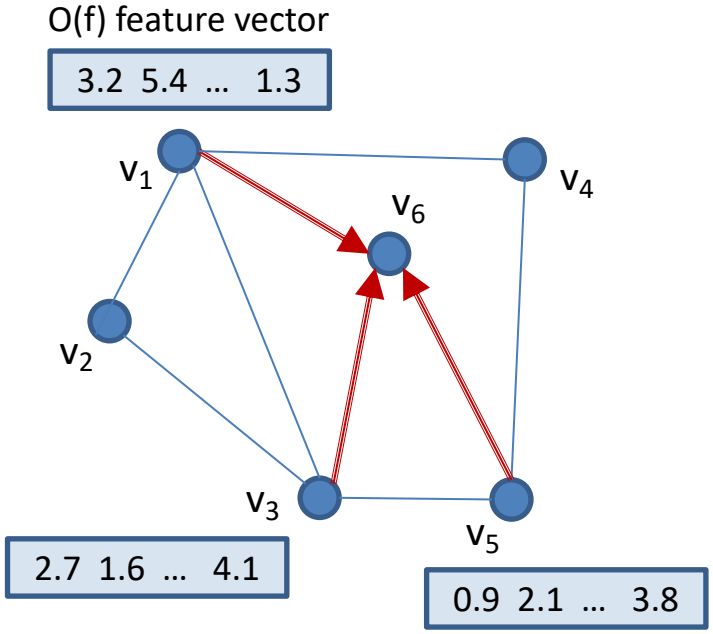
Used in Graph neural networks, graph embedding, etc.

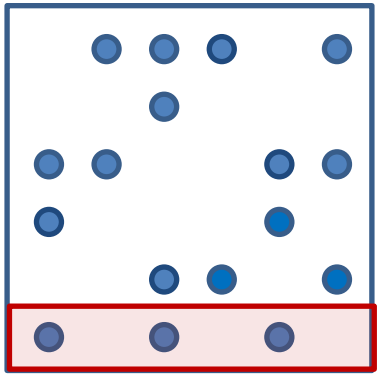
GrB_mxm(W, GrB_NULL, <semiring>, A, H, <desc>)

A: sparse adjacency matrix, n-by-n

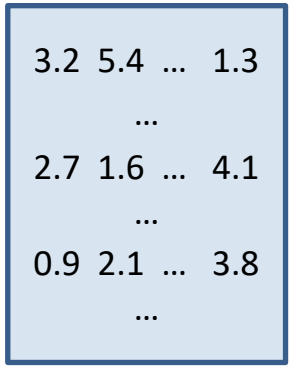
H: input dense matrix, n-by-f where f << n is the feature dimension

W: output dense matrix, new features



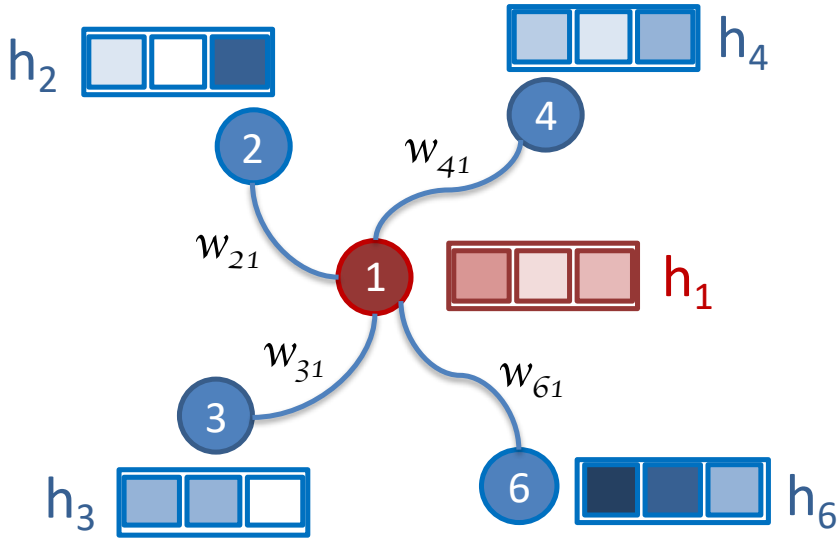


A^T



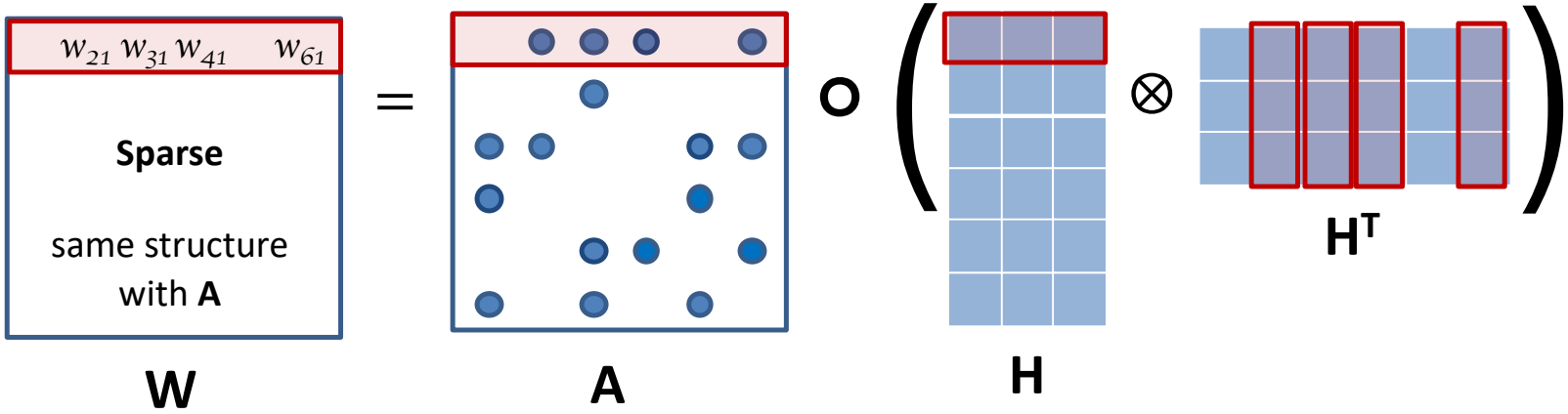
H

Graph attention: making edge weights learnable



SDDMM: Sampled dense-dense matrix multiplication

GrB_mxm(W, A, H, H, …);



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)

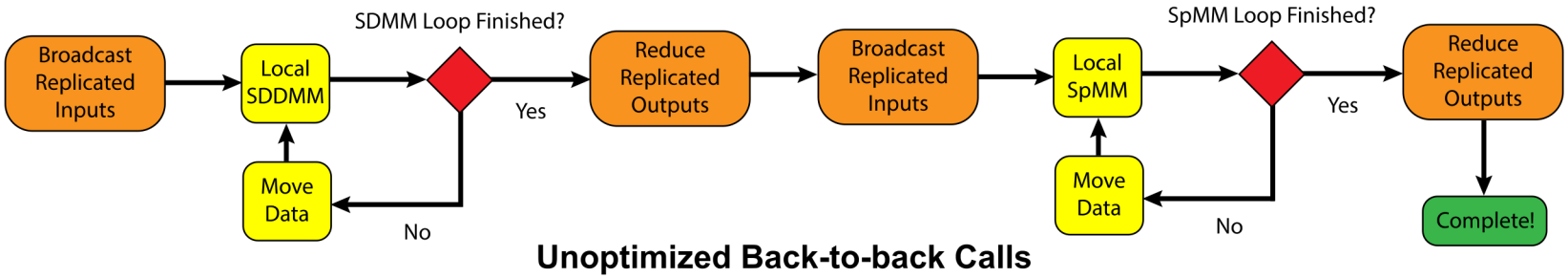
- Co-locate S_{ij}, A_{ik}, B_{jk} on a single processor
- Perform the update $A_{ik} \mathrel{+}= S_{ij} B_{jk}$

Transform this algorithm as follows:

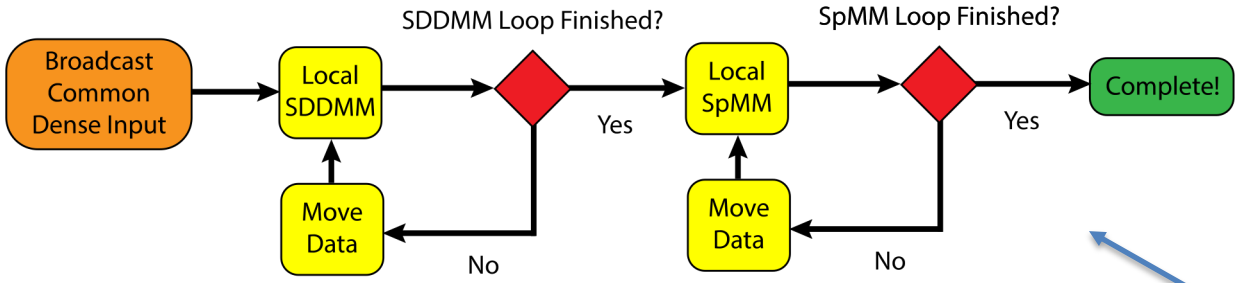
1. Change the input sparse matrix S to an output that is initialized to 0.
2. Change A from an output to an input.
3. Have each processor execute the local update: $S_{ij} \mathrel{+}= A_{ik} B_{jk}$

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

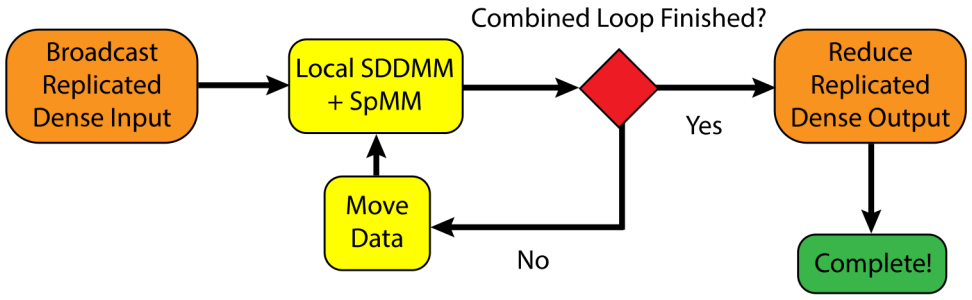
Communication Eliding Strategies for FusedMM: SDDMM+SpMM



Unoptimized Back-to-back Calls



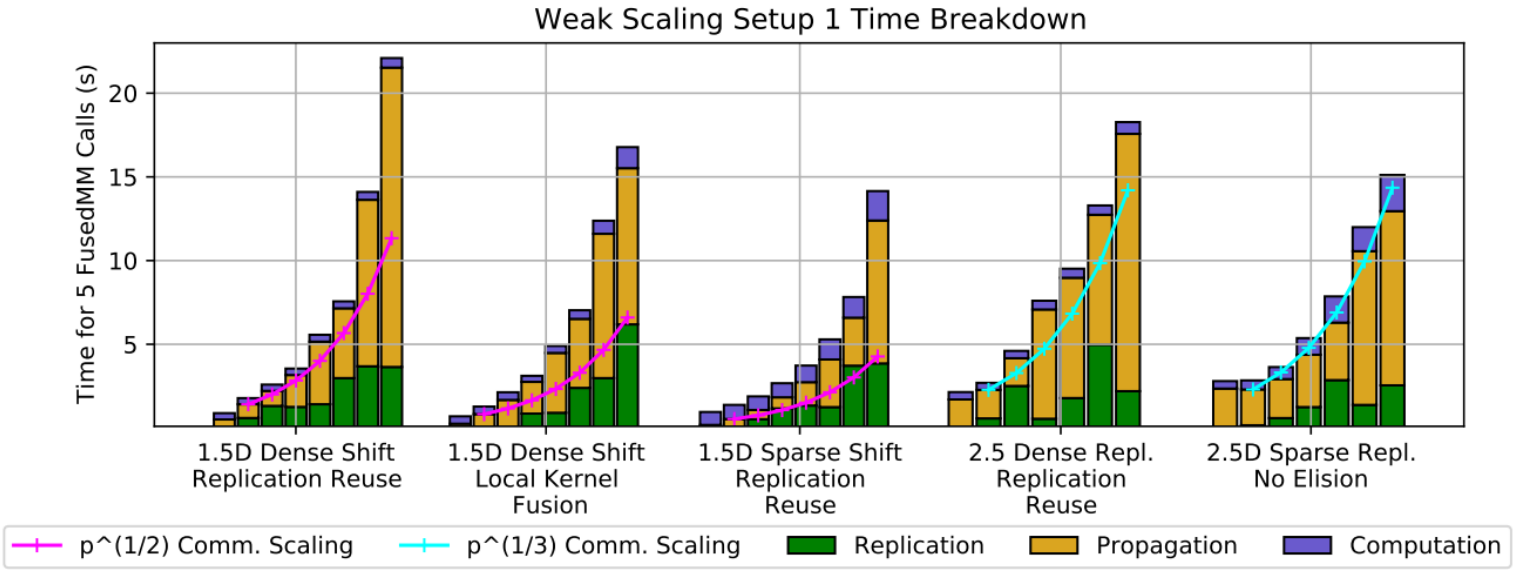
Replication Reuse



Local Kernel Fusion

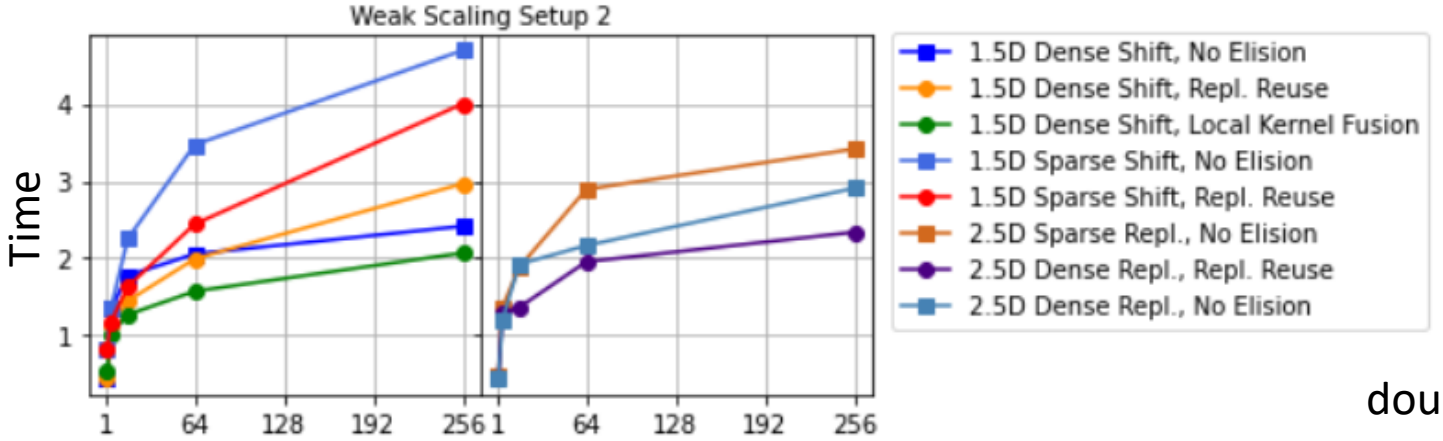
Mutually exclusive optimizations

Distributed FusedMM performance



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

remains constant



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

doubles at each process count quadrupling

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, Gabriel Raulet, 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