Berkeley
UNIVERSITY OF CALIFORNIA

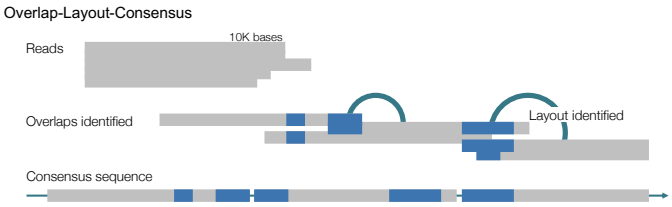
Computational Journeys in a Sparse Universe

Aydın Buluç
Applied Math & Computational Research Division, LBNL
EECS Department, UC Berkeley

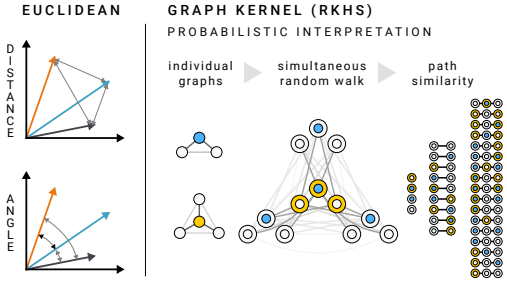
Georgia Tech.
CSE 6230: HPC Tools and Applications
March 28, 2024

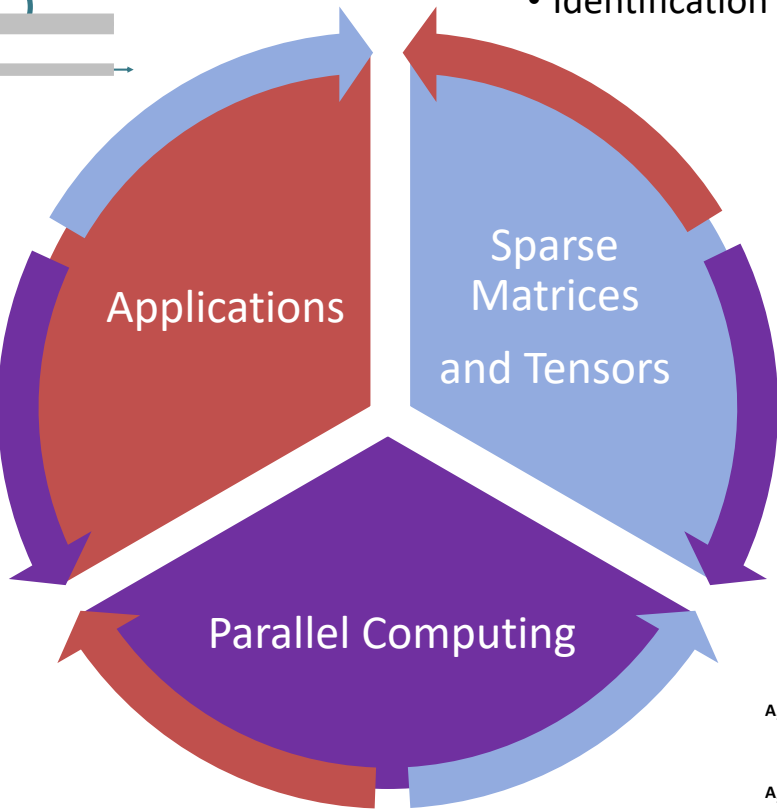
PASSION Lab Research Agenda

http://passion.lbl.gov

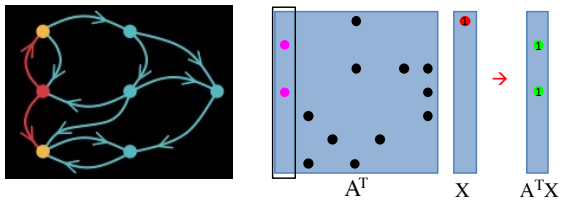


- Genomics
- Graph analysis
- Proteomics
- Graph learning



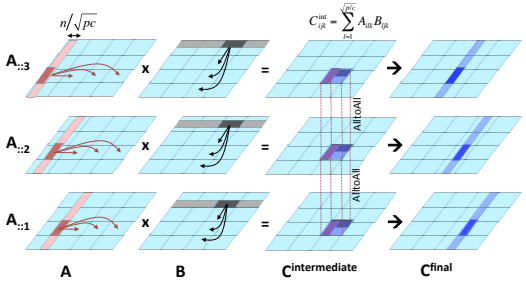


- New sparse data structures and algorithms
- Identification of computational primitives



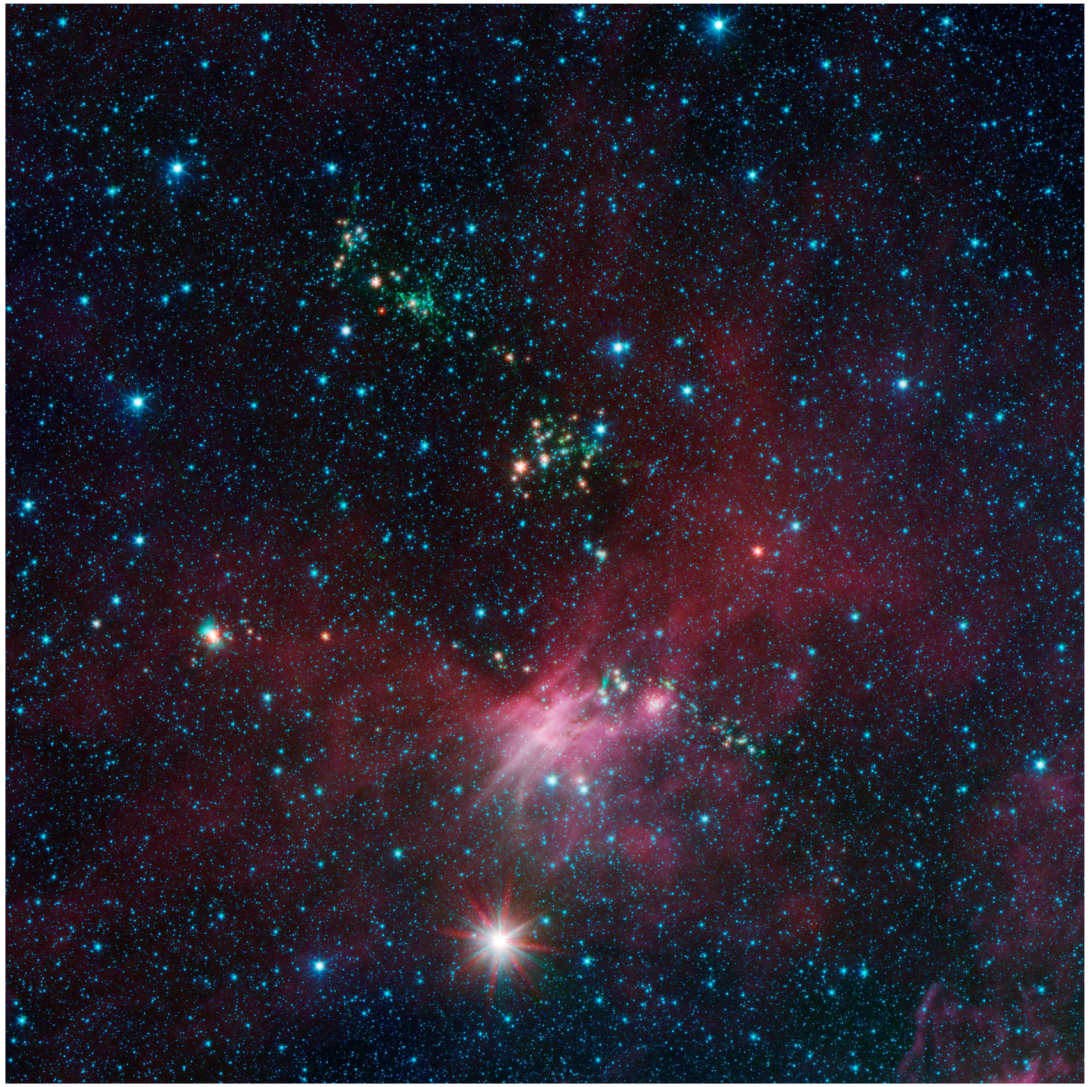
GraphBLAS: graphs in the language of linear algebra
http://graphblas.org

Communication-avoiding algorithms for sparse matrices



- Parallel data structures
- Parallel programming
- Communication bounds

It is a Sparse Universe we live in



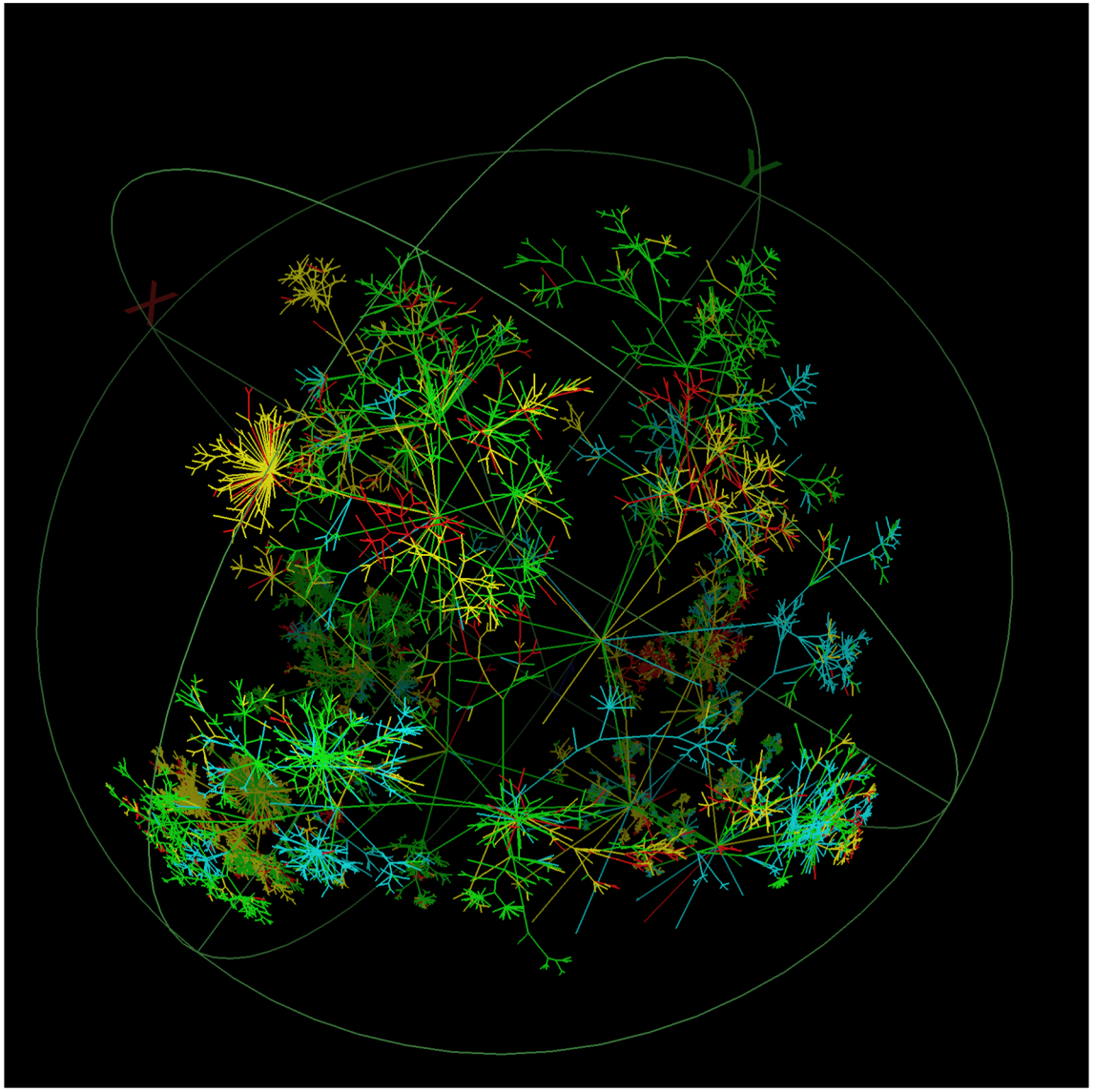
Stars Shoot Jets in Cosmic Playground

https://www.flickr.com/photos/nasablueshift/9027742916

With Sparse Interactions

Round-Trip Time Internet Measurements from CAIDA's Macroscopic Internet Topology Monitor

https://www.caida.org/catalog/software/walrus/rtt/

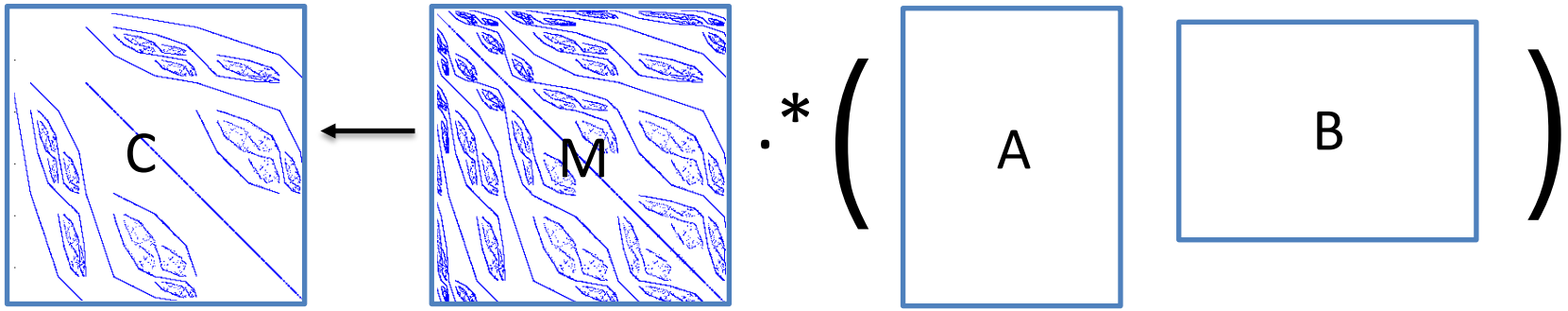


Sparsity enables Scalability

- **Curse of dimensionality:** As the dimensions get larger so does the sparsity (if defined as "the ratio of potential interactions to non-negligible interactions")
  - Higher-dimensional networks and tensor are even sparser by that definition
- **Scalability of models:**
  - Not every cell can directly interact with every other cell in a meaningfully impactful way.
  - Sparsity is a precondition for compressed sensing in signal processing
  - Power grid models, traffic models, molecule models, are all sparse by construction
  - Most machine learning models (CNNs, GNNs) are sparse
- **Scalability of solutions:** one can't solve any system with $O(N^k)$ for $k \geqq 2$ for really large N
  - Many of the "algorithms of the century" are based on sparsity assumptions (e.g., FMM)
  - Fast numerical optimization methods aggressively exploit sparsity

Sparse matrix-matrix multiplication

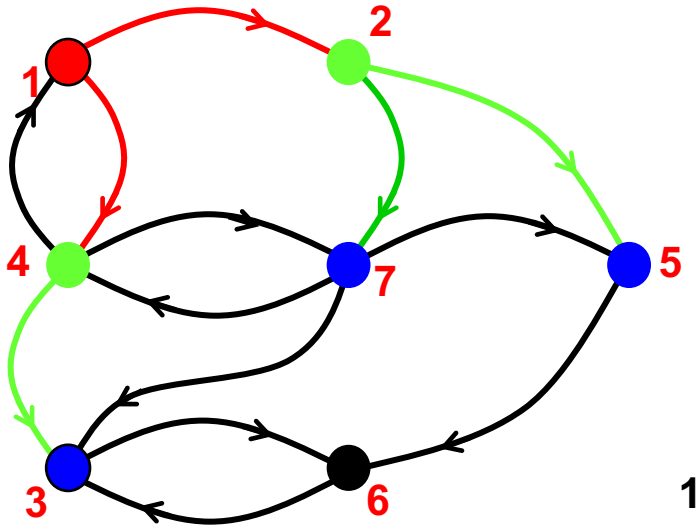
$$C(\neg M) \oplus = A^T \oplus . \otimes B^T$$



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

Pattern 1: Sparse matrix times sparse vector (SpMSpV)



Single-source traversal:

BFS, connected components, matching, ordering, etc.

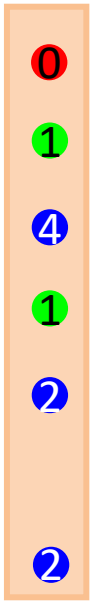
GrB_mxv(y, p, <semiring>, A, x, <desc>)

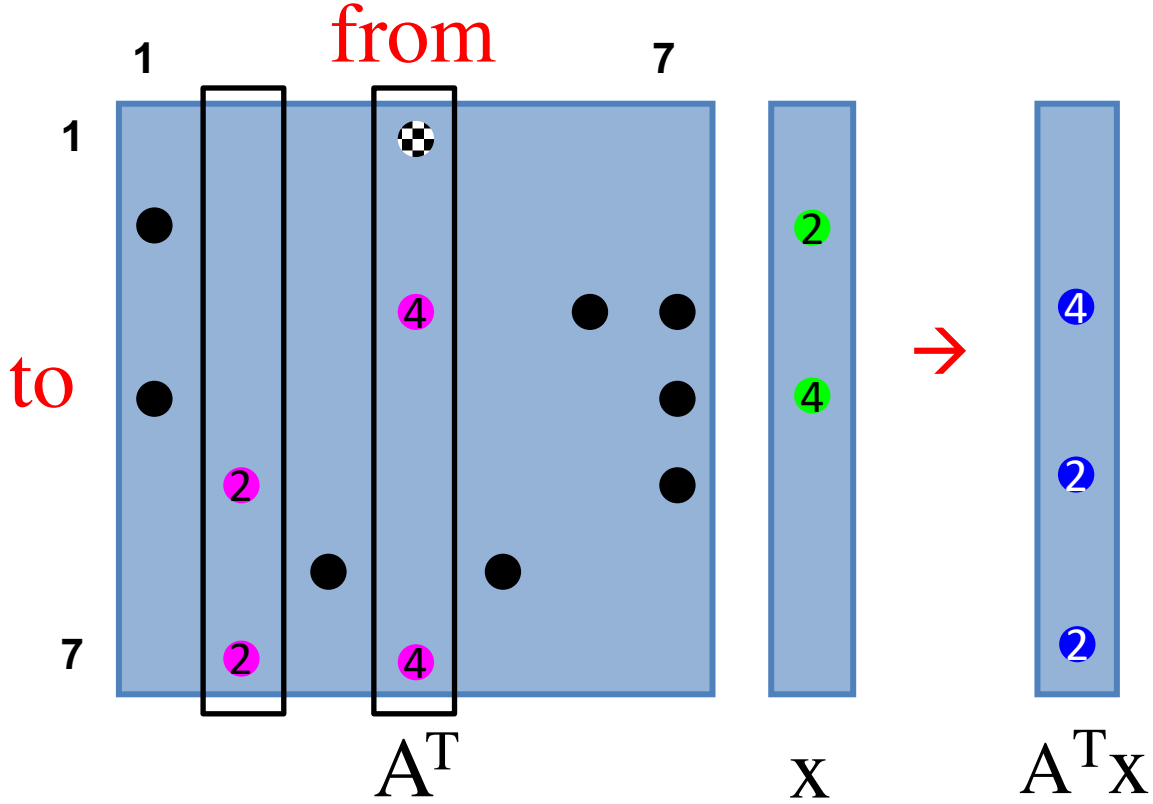
A: sparse adjacency matrix

x: sparse input vector (previous frontier)

p: mask (already discovered vertices)

parents (p):





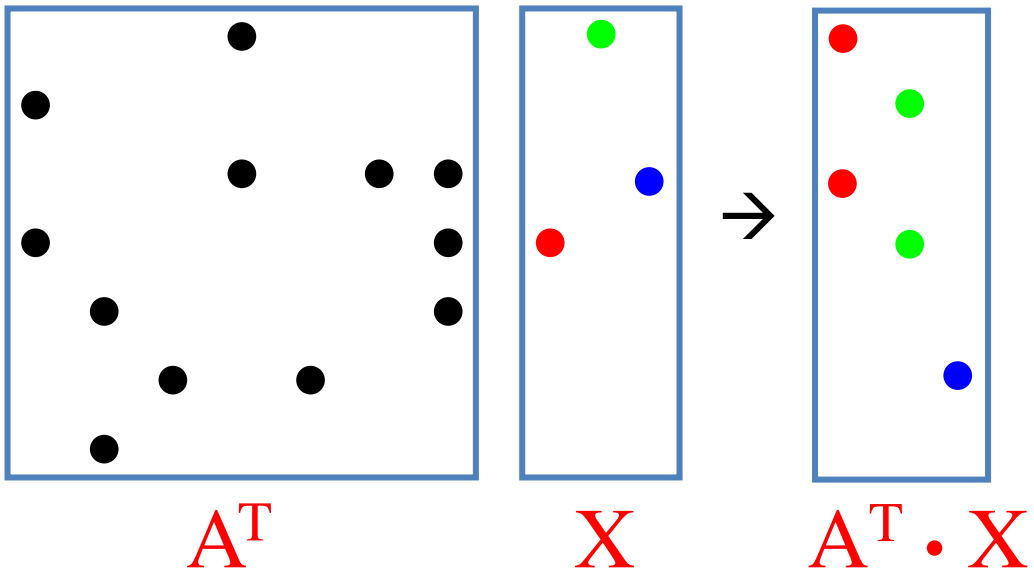
Pattern 2: Sparse matrix times sparse matrix (SpGEMM)

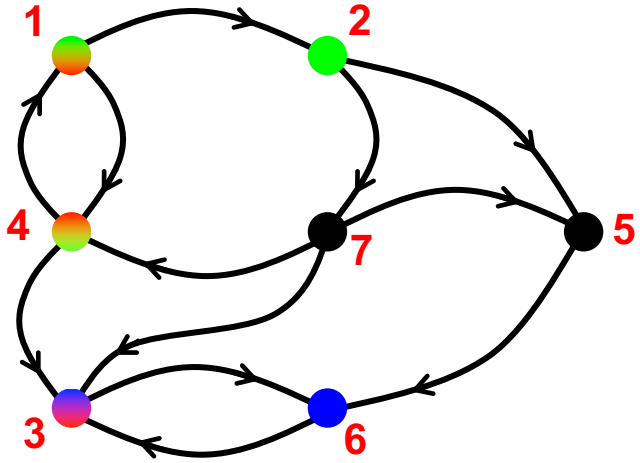
Multi-source traversal:

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

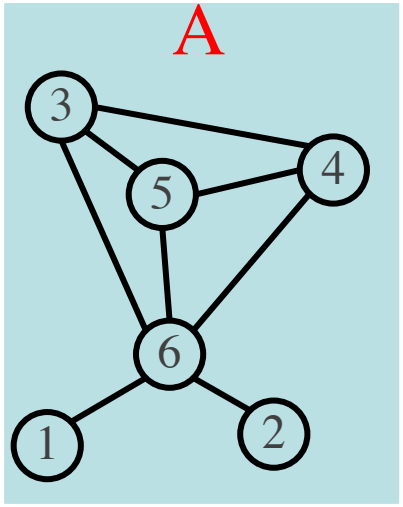




Pattern 2: Sparse matrix times sparse matrix (SpGEMM)

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>)

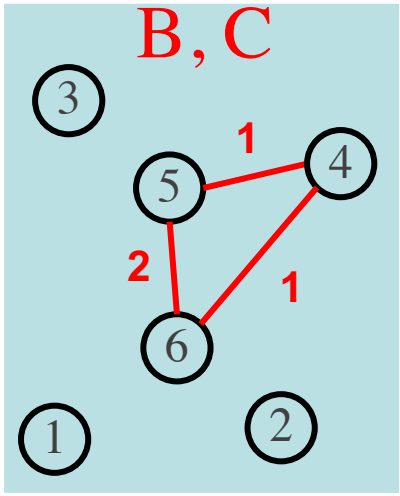


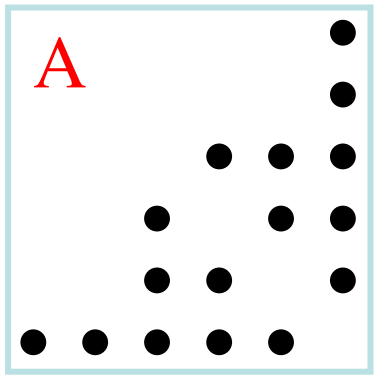
$A = L + U$ (hi->lo + lo->hi)

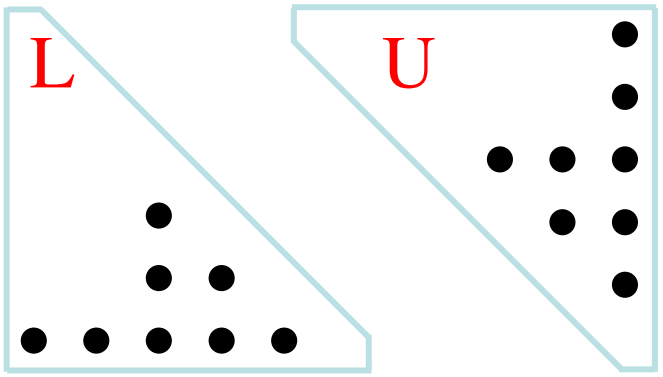
$L \times U = B$ (wedge, low hinge)

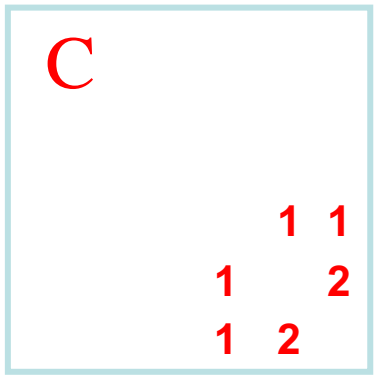
$A \wedge B = C$ (closed wedge)

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

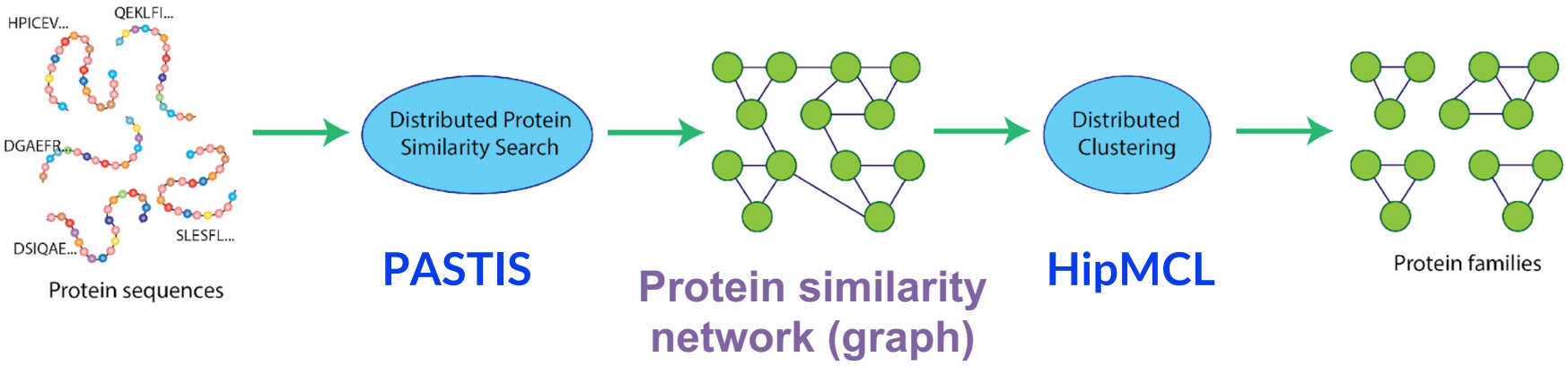




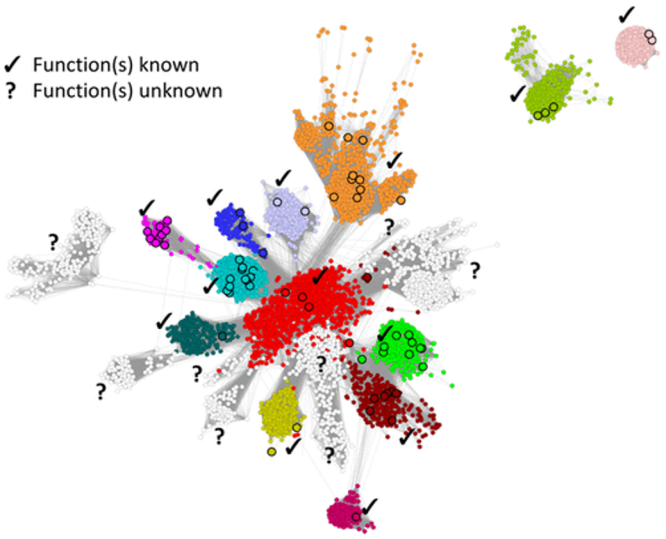




Protein Family Identification



- 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
- Often, only sequences (and not structure) of the proteins are available, so we infer homology via sequence similarity



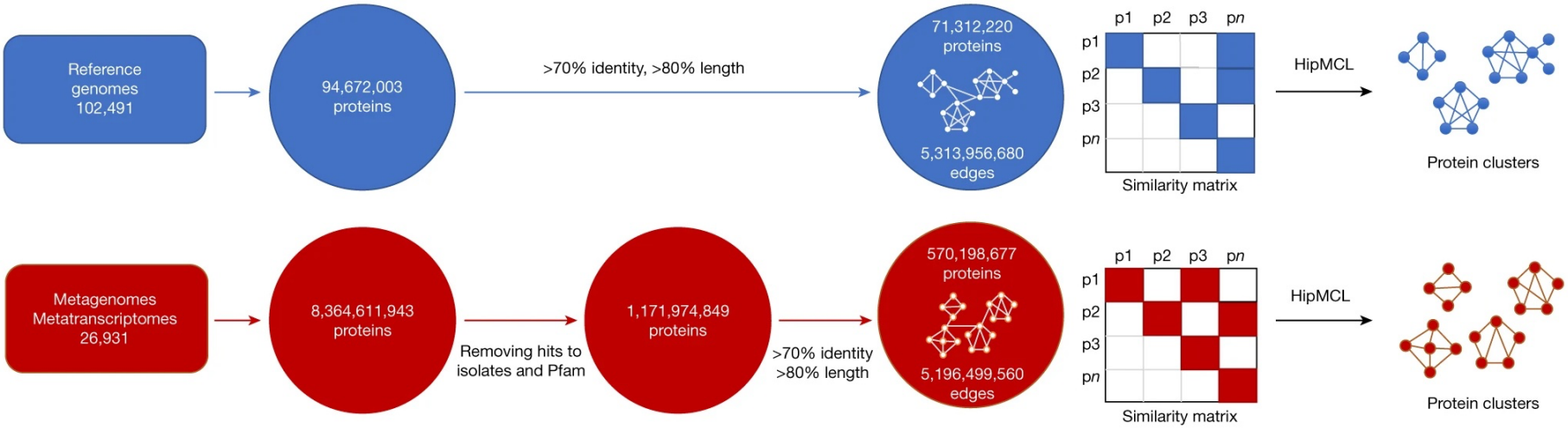
Novel Protein Families in Microbial Dark Matter

Microbial dark matter: novel proteins after removing matches to a database of over 100,000 genomes (including Archaeal, Bacteria, Viral and Eukaryotic)

Unraveling the functional dark matter through global metagenomics

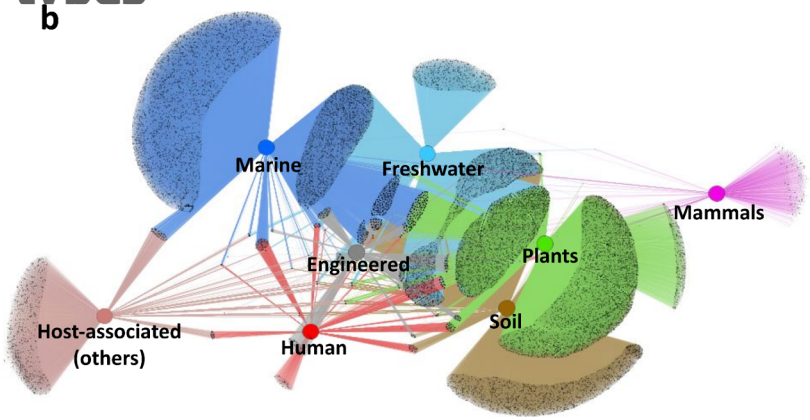
Georgios A. Pavlopoulos ✉, Fotis A. Baltoumas, Sirui Liu, Oguz Selvitopi, Antonio Pedro Camargo, Stephen Nayfach, Ariful Azad, Simon Roux, Lee Call, Natalia N. Ivanova, I. Min Chen, David Paez-Espino, Evangelos Karatzas, Novel Metagenome Protein Families Consortium, Ioannis Iliopoulos, Konstantinos Konstantinidis, James M. Tiedje, Jennifer Pett-Ridge, David Baker, Axel Visel, Christos A. Ouzounis, Sergey Ovchinnikov, Aydin Buluç & Nikos C. Kyrpides ✉

Nature **622**, 594–602 (2023) | Cite this article



Diversity of Novel Protein Families

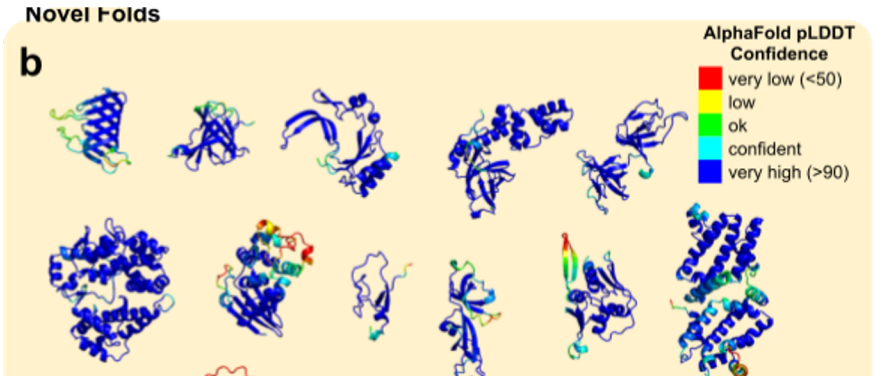
Novel protein clusters are distributed across 8 ecosystem types



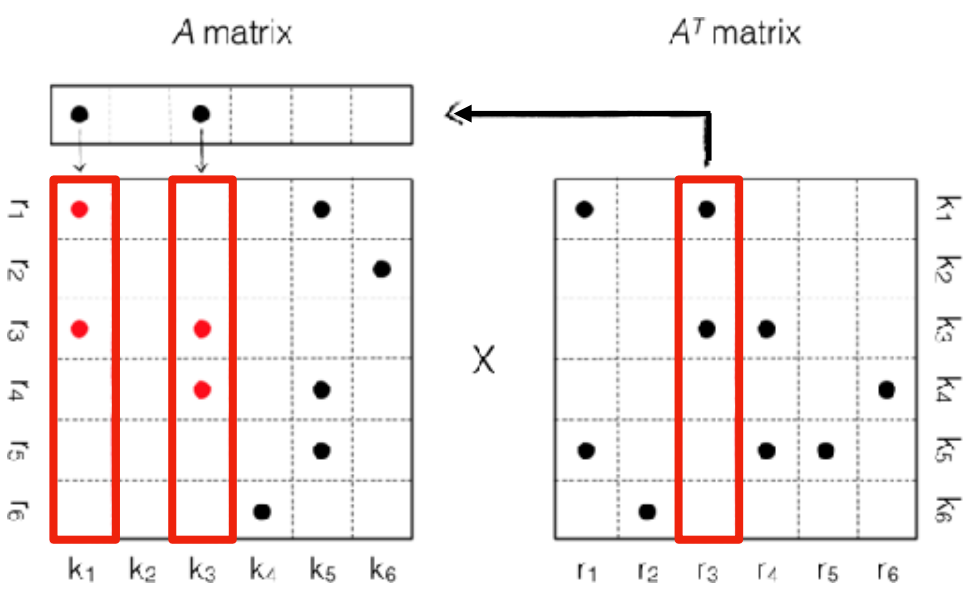
Network representation of protein clusters (gray peripheral) and their associated ecosystems (colored central)

Distribution of protein structures

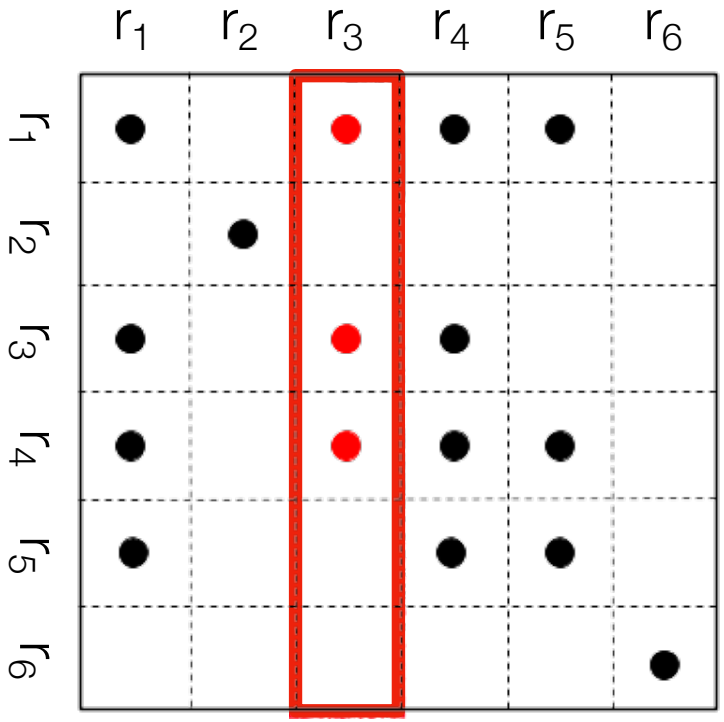
- 4,361 unique structures were predicted using AlphaFold
- 3,808 structures has hits in SCOPe.
- 345 has hits in Protein Data Bank (PDB).
- After further filtering, 162 structures are considered novel folds



Finding candidate sequence pairs



Sequence-by-sequence overlap matrix: AA^T



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

Use any fast SpGEMM (sparse matrix times sparse matrix multiplication) algorithm and implementation, needs to run on ***arbitrary semirings for position tracking***

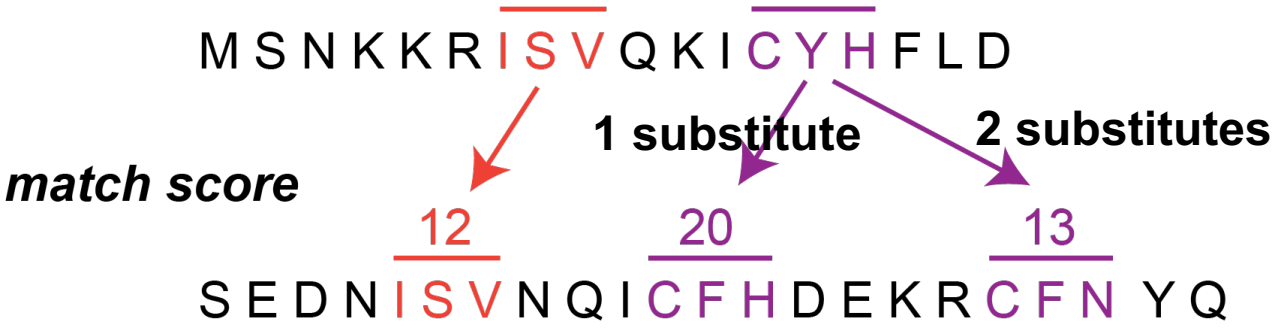
Distributed-memory many-to-many protein alignment

- For proteins we need to relax the exact match restriction of AA^T
- 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



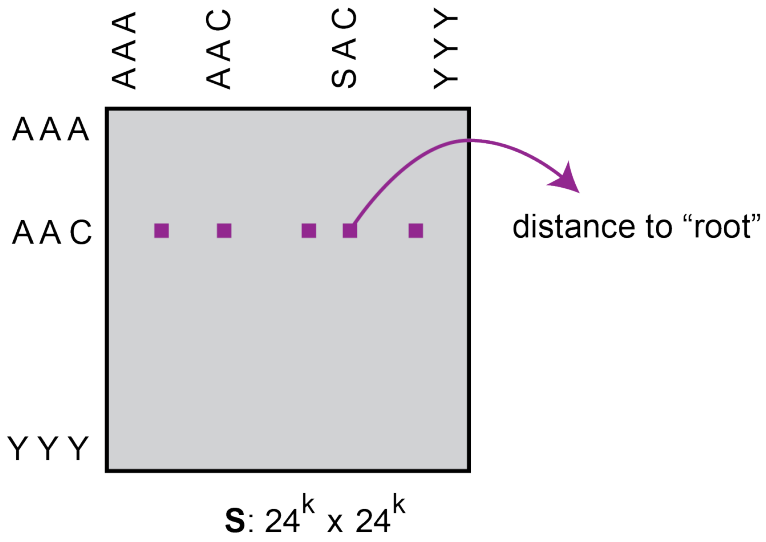
SpGEMM for many-to-many protein alignment

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 as 2022 Gordon Bell Finalist

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

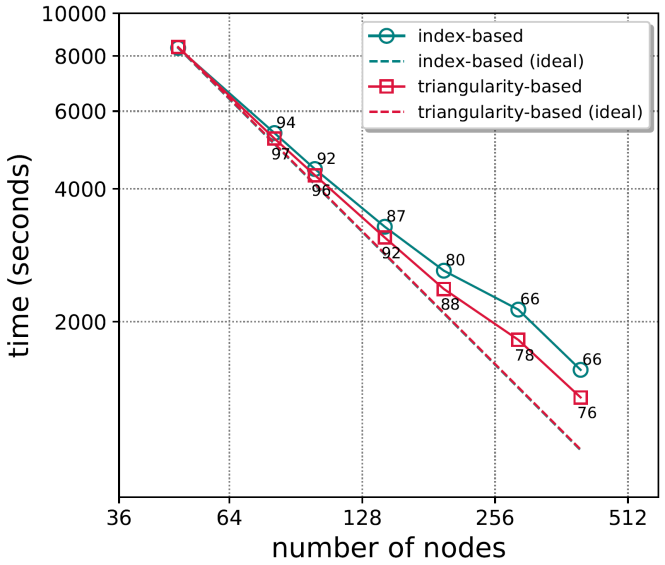
Abstract-- … We unleash the power of over 20,000 GPUs on the Summit system to perform all-vs-all protein similarity search on one of the largest publicly available datasets with 405 million proteins, in less than 3.5 hours, cutting the time-to-solution for many use cases from weeks. The variability of protein sequence lengths, as well as the sparsity of the space of pairwise comparisons, make this a challenging problem in distributed memory …

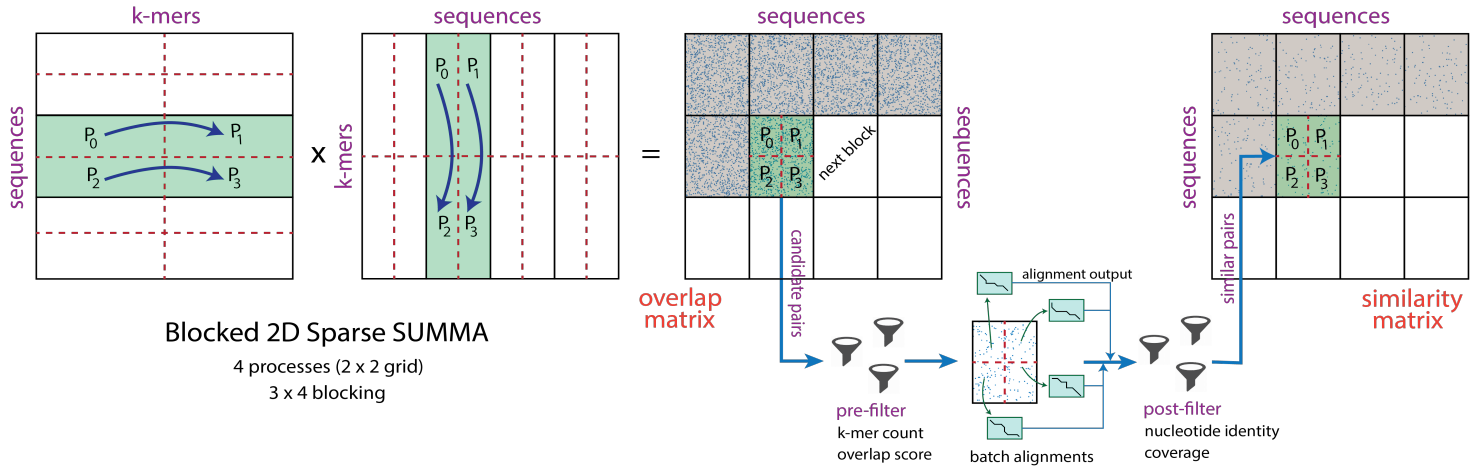
Extreme-scale many-against-many protein similarity search

Advances: memory-consumption optimizations, new parallel algorithms taking advantage of the symmetry in the sequence similarity matrix, GPU acceleration, the ability to address load imbalance issues

Result: many-against-many protein search on 405 million proteins with PASTIS on 3364 compute nodes of ORNL Summit in 3.4 hours, sustaining a rate of 691 million alignments/sec and attaining ~176 TCUPs (Tera Cell Updates/sec).

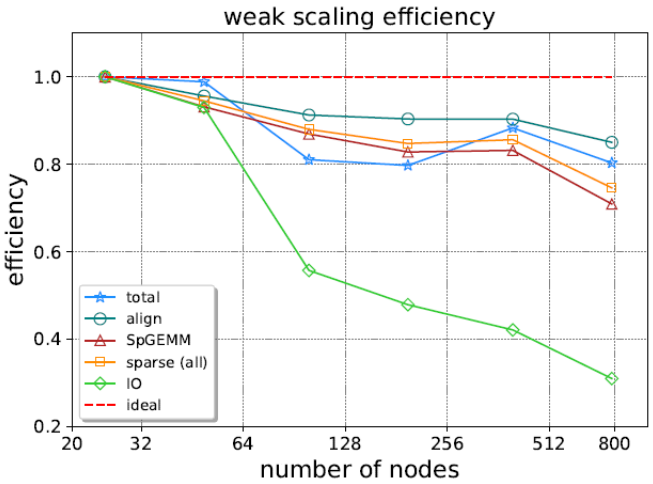
The output protein sequence similarity graph is 27 TB.

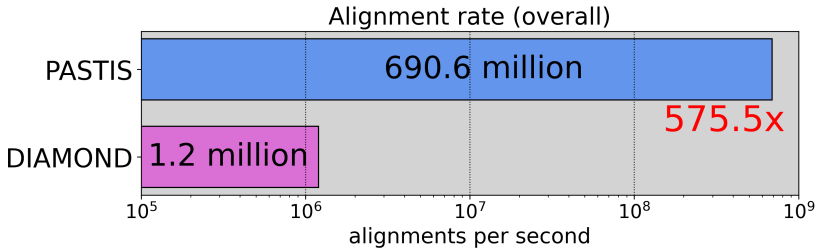


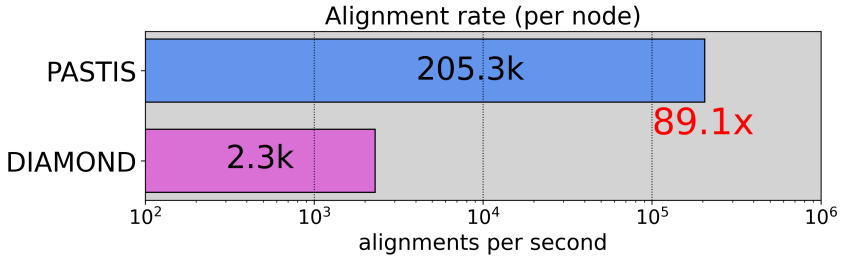


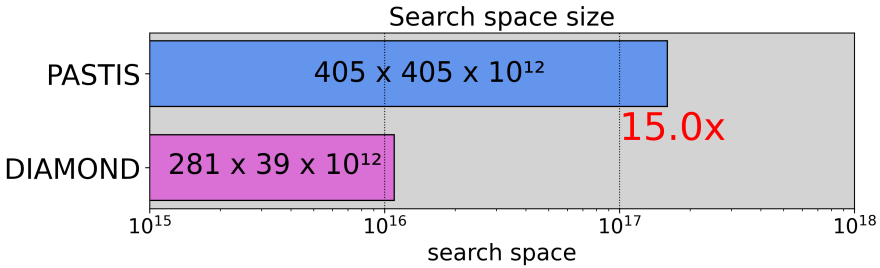
Similarity search at scale: Advancements

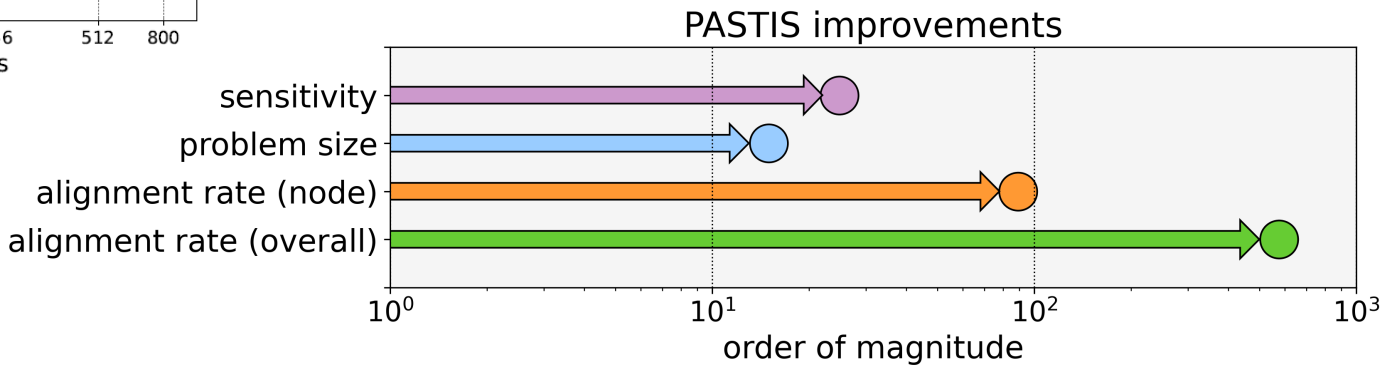
- Discovered candidates: **96T**
- Performed alignments: **8.6T**
- Similar pairs: **1.1T**
- Runtime: **3.44 hours**





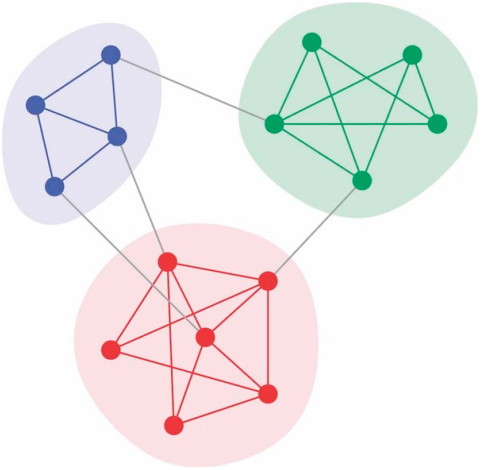






The Markov Cluster Algorithm (MCL)

Widely popular and successful algorithm for discovering clusters (e.g. protein families) in protein interaction and protein sequence similarity networks

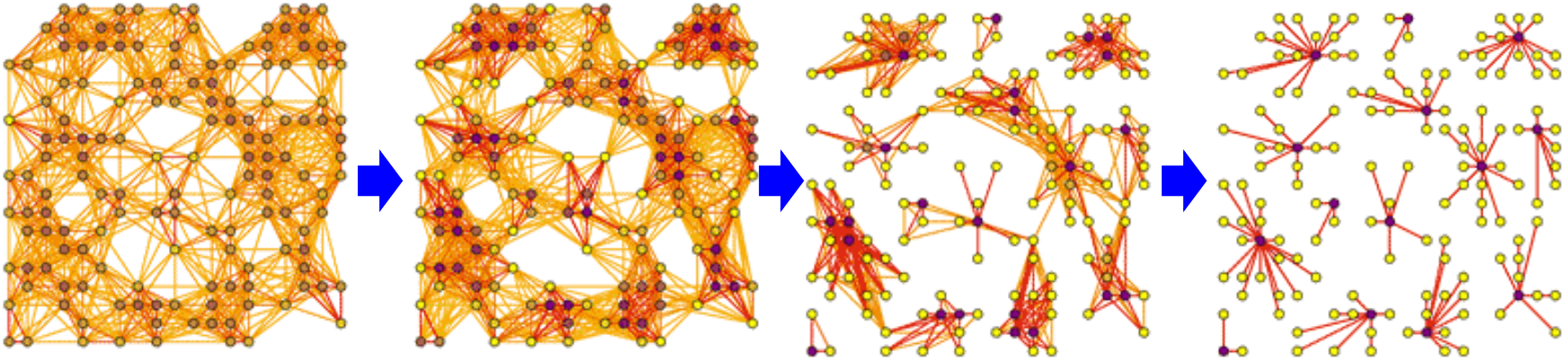


The number of **edges or higher-length paths** between two arbitrary nodes in a cluster is greater than the number of paths between nodes from different clusters

Random walks on the graph will frequently remains within a cluster

The algorithm **computes the probability** of random walks through the graph and **removes lower probability terms** to form clusters

The Markov Cluster Algorithm (MCL)



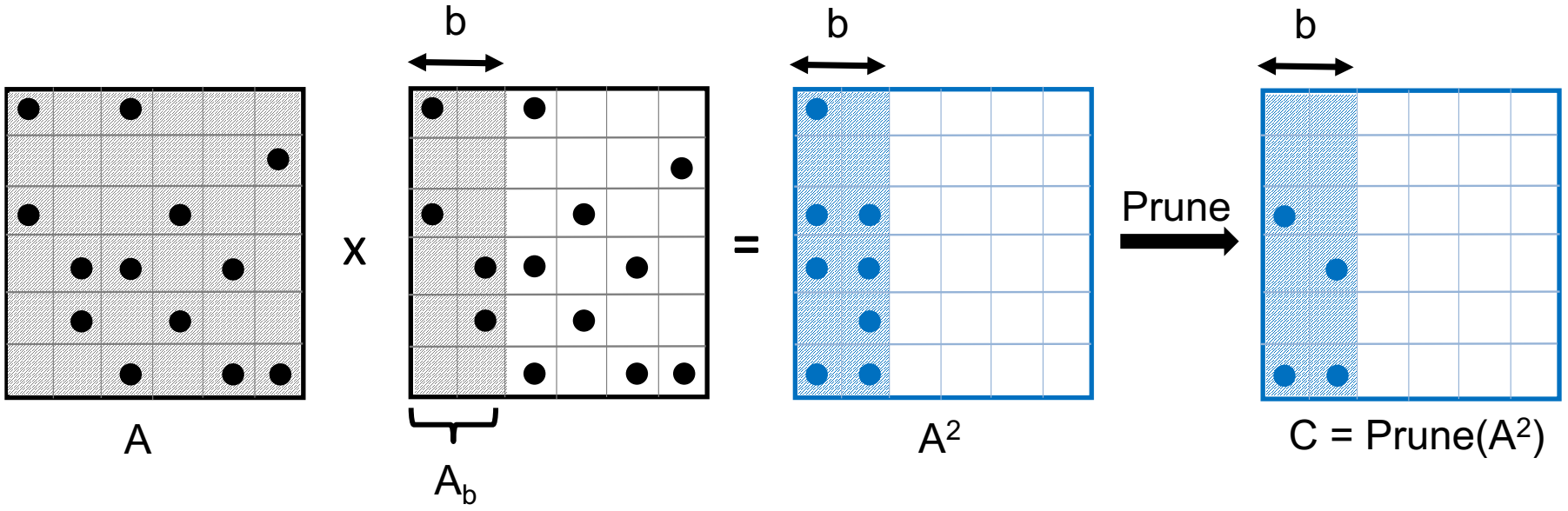
At each iteration:

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

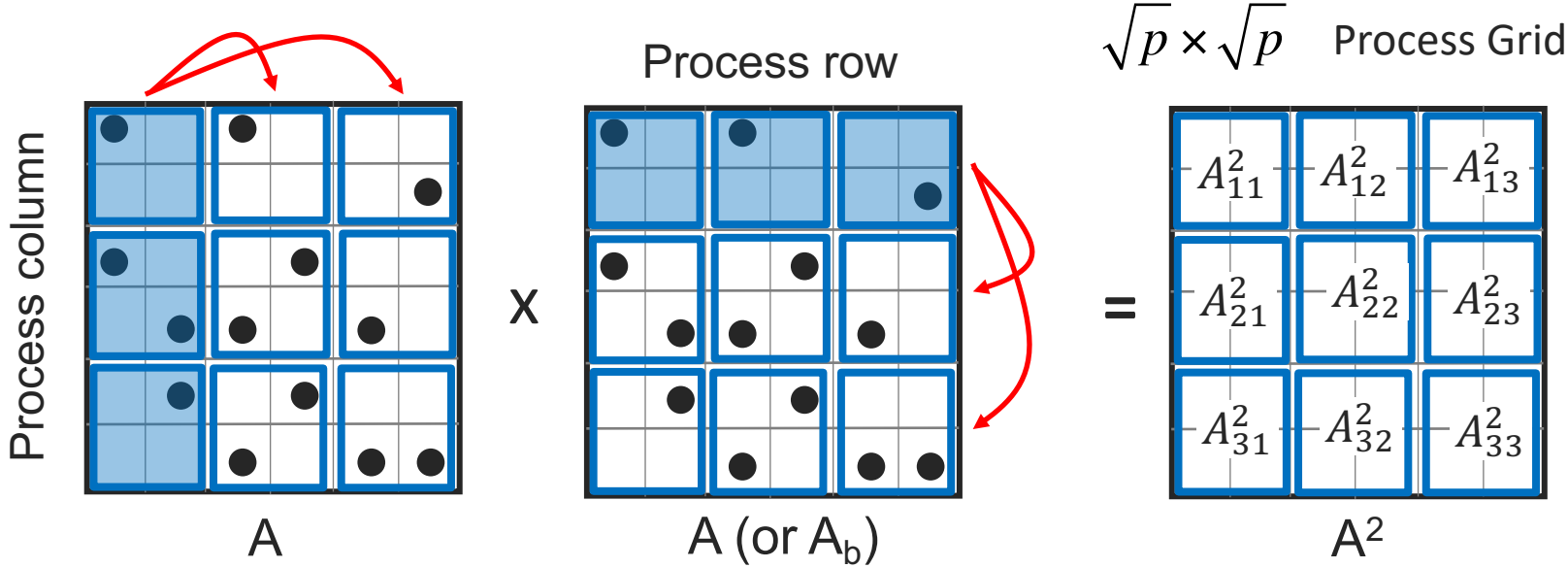
A combined expansion and pruning step



- b: number of columns in the output constructed at once
  - Smaller b: less parallelism, memory efficient (b=1 is equivalent to sparse matrix-sparse vector multiplication used in MCL)
  - Larger b: more parallelism, memory intensive

HipMCL: High-performance MCL

- MCL process is both **computationally expensive** and **memory hungry**, limiting the sizes of networks that can be clustered
- HipMCL overcomes such limitation via **sparse parallel algorithms**.
- **Up to 1000X times faster** than original MCL with same accuracy.



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*

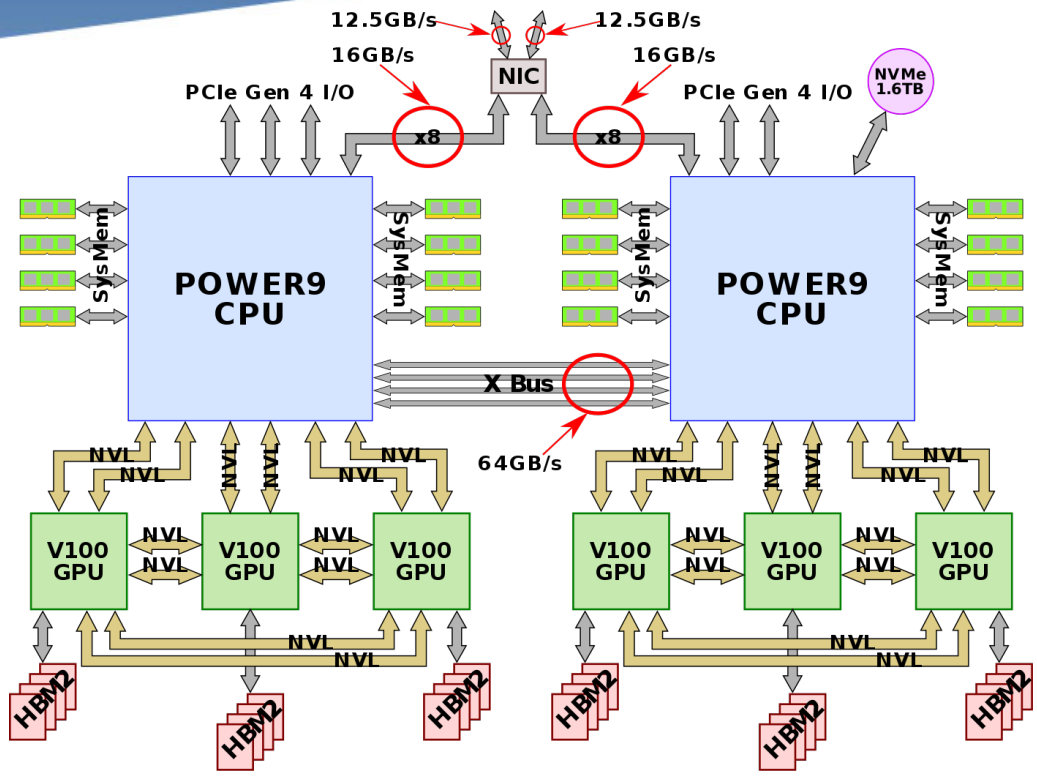
HipMCL on large networks

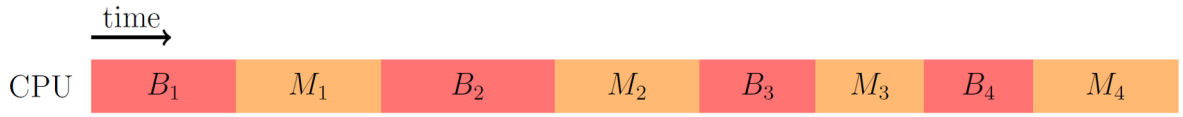
Data	Proteins	Edges	#Clusters	HipMCL time	platform
Isolate-1	47M	7 B	1.6M	1 hr	1024 nodes Edison
Isolate-2	69M	12 B	3.4M	1.66 hr	1024 nodes Edison
Isolate-3	70M	68 B	2.9M	2.41 hr	2048 nodes Cori KNL
MetaClust50	282M	37B	41.5M	3.23 hr	2048 nodes Cori KNL

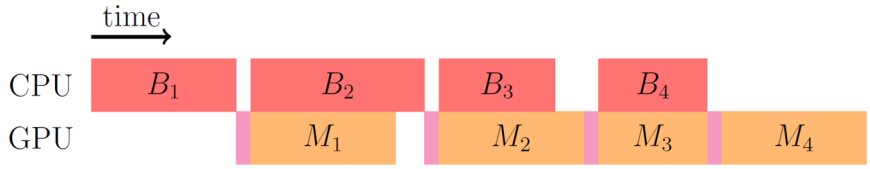
MCL can not cluster these networks

HipMCL on Supercomputers with accelerators

- Recent top supercomputers are all accelerated (e.g. with GPUs)
- This is what a ORNL Summit node looks like
- There are 4608 such nodes in the system
- Challenges: (1) Utilizing all GPUs, (2) hiding the communication



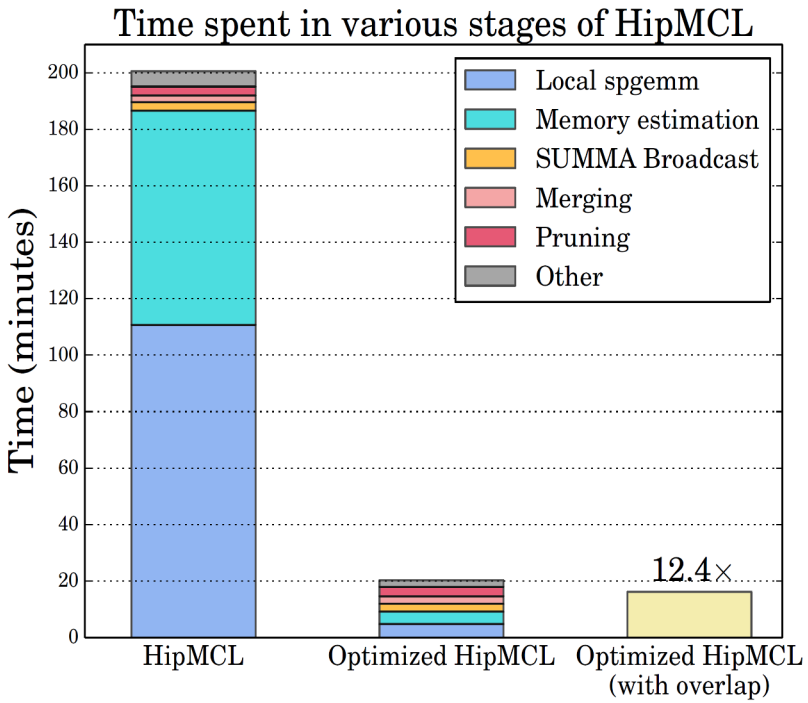


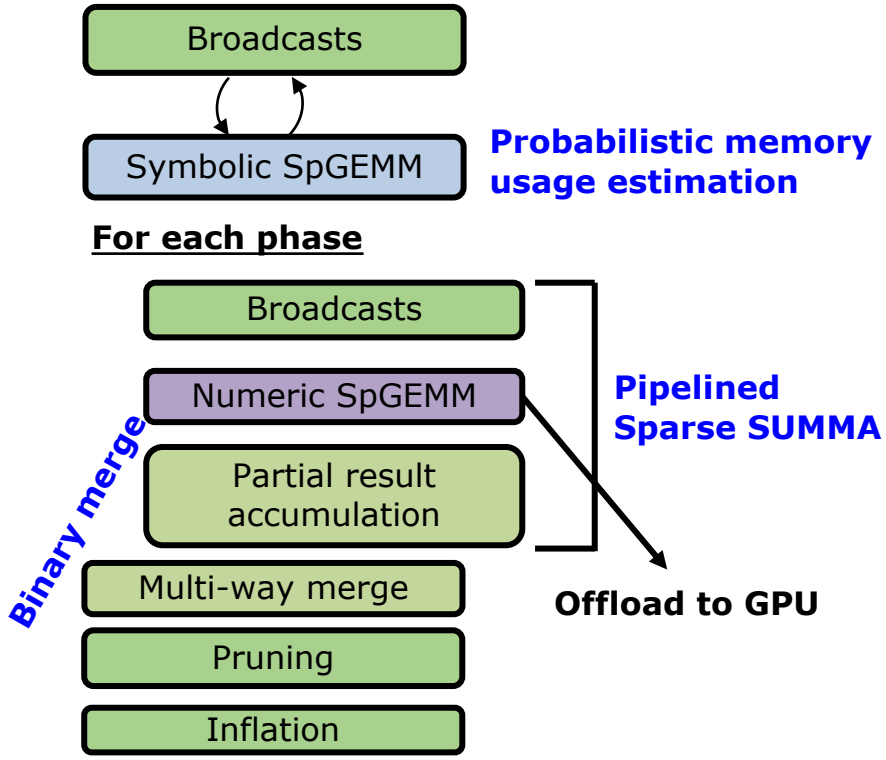


Pipelined Sparse SUMMA

Joint CPU-GPU distributed memory expansion of MCL algorithm

HipMCL on Supercomputers with accelerators





Other changes to HipMCL for the CPU-GPU workflow:

- *Randomized memory estimation algorithm* avoids symbolic phase
- New *eager binary merging* reduces memory footprint
- Integration of a much faster hash-based CPU SpGEMM algorithm

O. Selvitopi, M.T. Hussain, A. Azad, and A. Buluç. Optimizing high performance Markov clustering for pre-exascale architectures. IPDPS, 2020

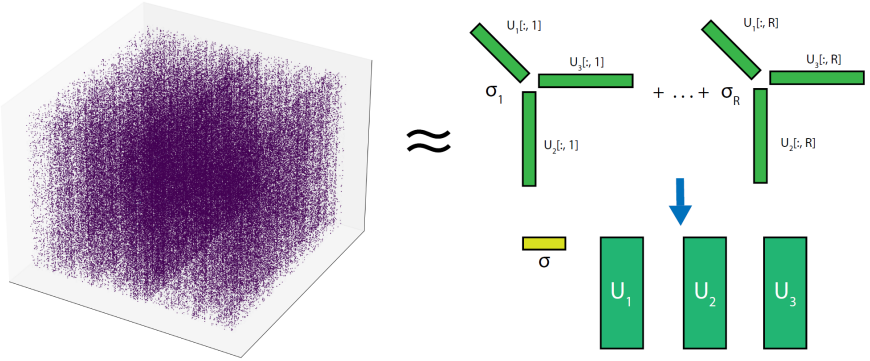
Fast Exact Leverage Score Sampling from Khatri-Rao Products

The Khatri-Rao product (KRP, denoted $\odot$) is the column-wise Kronecker product of two matrices: $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \odot \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} aw & bx \\ cw & dx \\ ay & bz \\ cy & dz \end{bmatrix}$

We want to efficiently solve an overdetermined linear least-squares problem

$\min_X \|AX - B\|_F$ where $A = U_1 \odot \cdots \odot U_N$ with $U_j \in \mathbb{R}^{I_j \times R}$.

This structured least-squares problem is the computational bottleneck in alternating least-squares Candecomp / PARAFAC (CP) Decomposition.



We focus on large **sparse tensors** (mode sizes in the millions) and moderate decomposition rank $R \approx 10^2$. Assume $I_j = I$ for all j and $I \geq R$.

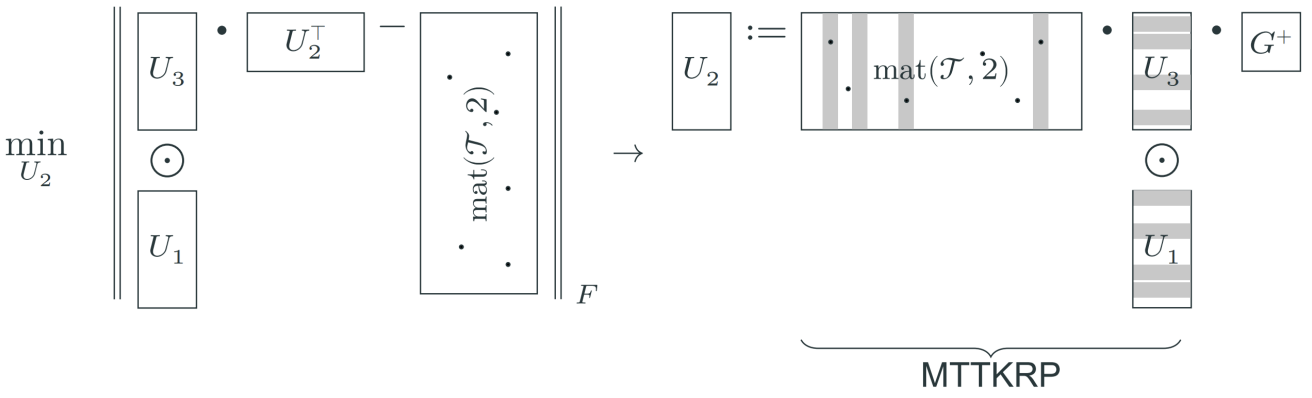
Randomized Linear Least Squares

- Apply sketching operator S to A and B, solve reduced problem $\min_{\tilde{X}} \left\| SA\tilde{X} - SB \right\|_F$

- Want an (ε, δ) guarantee on solution quality: with high probability $(1-\delta)$, $\left\| A\tilde{X} - B \right\|_F \leq (1+\varepsilon) \min_X \|AX - B\|_F$
- Restrict S to be a *sampling matrix*: selects and reweights rows from A and B.
- Crux: ***sample Khatri-Rao product without forming the product***

The grey rows are sampled:

$$\min_{U_j} \left\| \left[\bigodot_{k \neq j} U_k \right] \cdot U_j^\top - \text{mat}(\mathcal{T}, j)^\top \right\|_F$$



Leverage Score Sampling

We will sample rows i.i.d. from A according to the *leverage score distribution* on its rows. Leverage score ℓ_i of row i is $\ell_i = A[i,:](A^\top A)^+ A[i,:]^\top$

Theorem (Leverage Score Sampling Guarantees)

Suppose $S \in \mathbb{R}^{J \times I}$ is a leverage-score sampling matrix for $A \in \mathbb{R}^{I \times R}$, and define

$$\tilde{X} \coloneqq \operatorname{argmin}_{\tilde{X}} \left\| SA\tilde{X} - SB \right\|_F$$

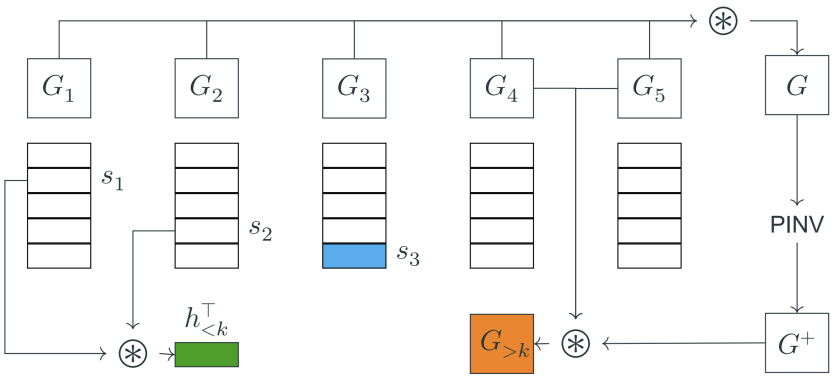
If $J \geq \Omega(R \max(\log(R/\delta), 1/(\varepsilon\delta)))$, then with probability at least $1 - \delta$,

$$\left\| A\tilde{X} - B \right\|_F \leq (1 + \varepsilon) \min_X \|AX - B\|_F$$

For $I = 10^7, N = 3$, matrix A has 10^{21} rows. Far too expensive to compute all leverage scores – can't even index rows with 64-bit integers.

Instead: draw a row from each of $U_1, \ldots, U_N$, return their Hadamard product.

Conditional sampling and key primitive



Let $\hat{s}_j$ be the random variable for the index drawn from U_j. Assume $(\hat{s}_1, \ldots, \hat{s}_N)$ jointly follows the leverage score distribution on A.

Theorem

$$p(\hat{s}_k = s_k | \hat{s}_{<k} = s_{<k}) \propto \langle h_{<k} h_{<k}^\top, U_k[s_k, :]^\top U_k[s_k, :], G_{>k} \rangle$$

Reduce to following problem: Given $U \in \mathbb{R}^{I \times R}$, design a data structure so for any query vector $h \in \mathbb{R}^R$, you can efficiently draw a sample according to probabilities

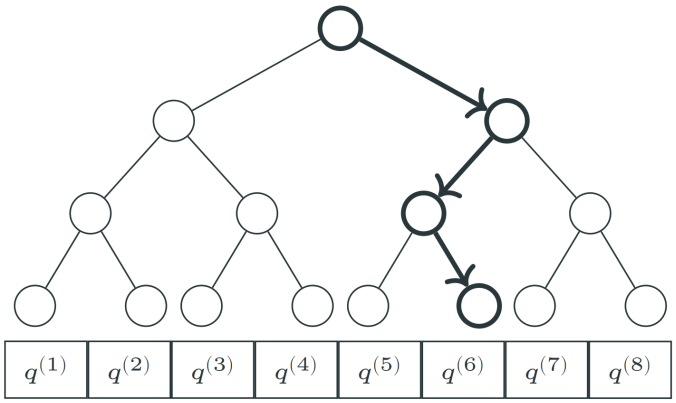
$$q = (U \cdot h)^2$$

We give a data structure based on a binary-tree caching scheme with

$O(IR^2)$ construction time

$O(IR)$ storage space

$O\left(R^2 \log (I/R)\right)$ time per query



Contributions of this work

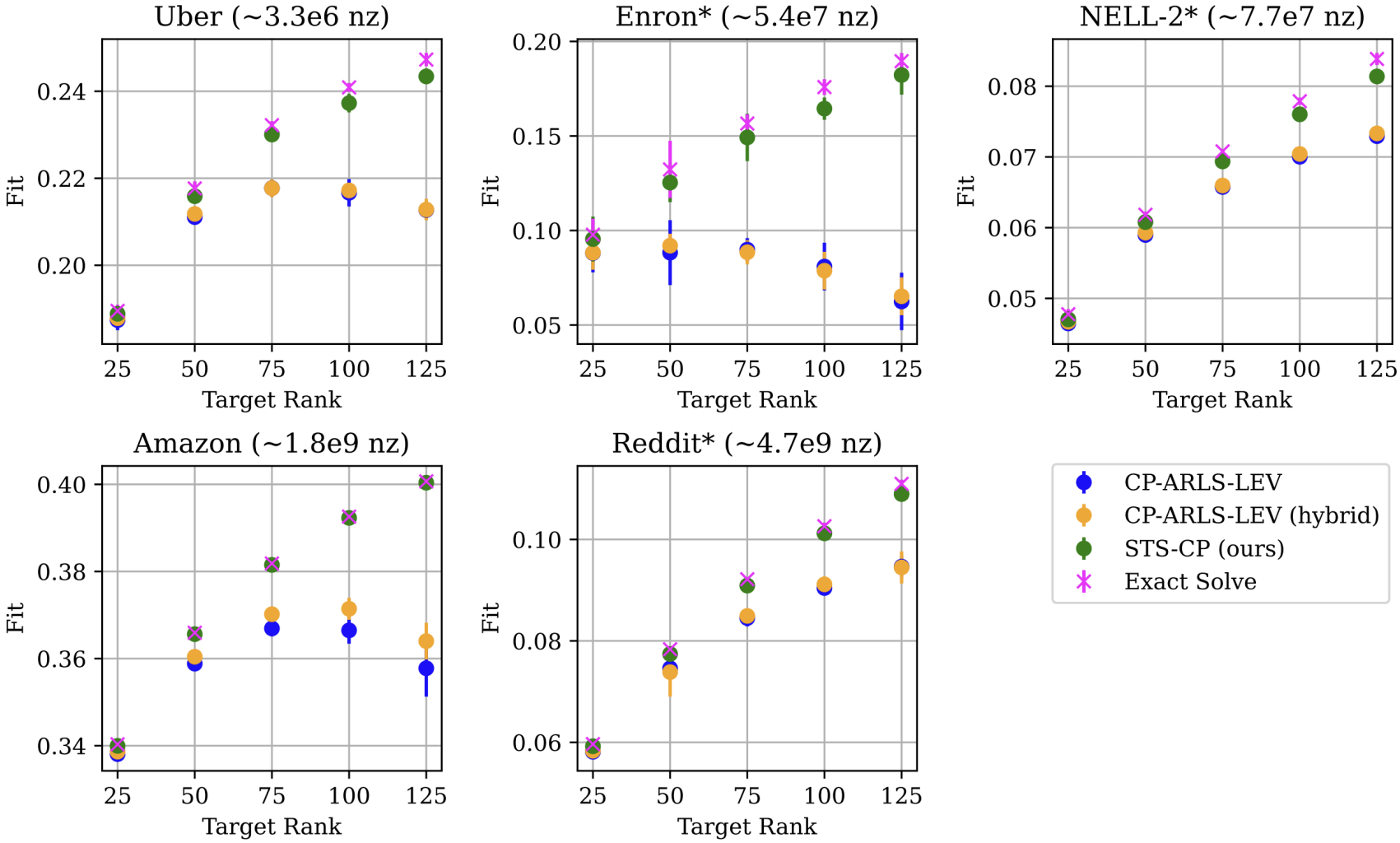
Method	$\tilde{O}$ Complexity per ALS Round
CP-ALS	$N(N+I)I^{N-1}R$
CP-ARLS-LEV (2022)	$N(R+I)R^N/(\varepsilon\delta)$
TNS-CP (2022)	$N^3IR^3/(\varepsilon\delta)$
GTNE (2022)	$N^2(N^{1.5}R^{3.5}/\varepsilon^3+IR^2)/\varepsilon^2$
STS-CP (ours)	$N(NR^3\log I+IR^2)/(\varepsilon\delta)$

- We build a data structure requiring runtime **logarithmic** in the height of the Khatri-Rao product and quadratic in R to sample from the *leverage score distribution* of A.

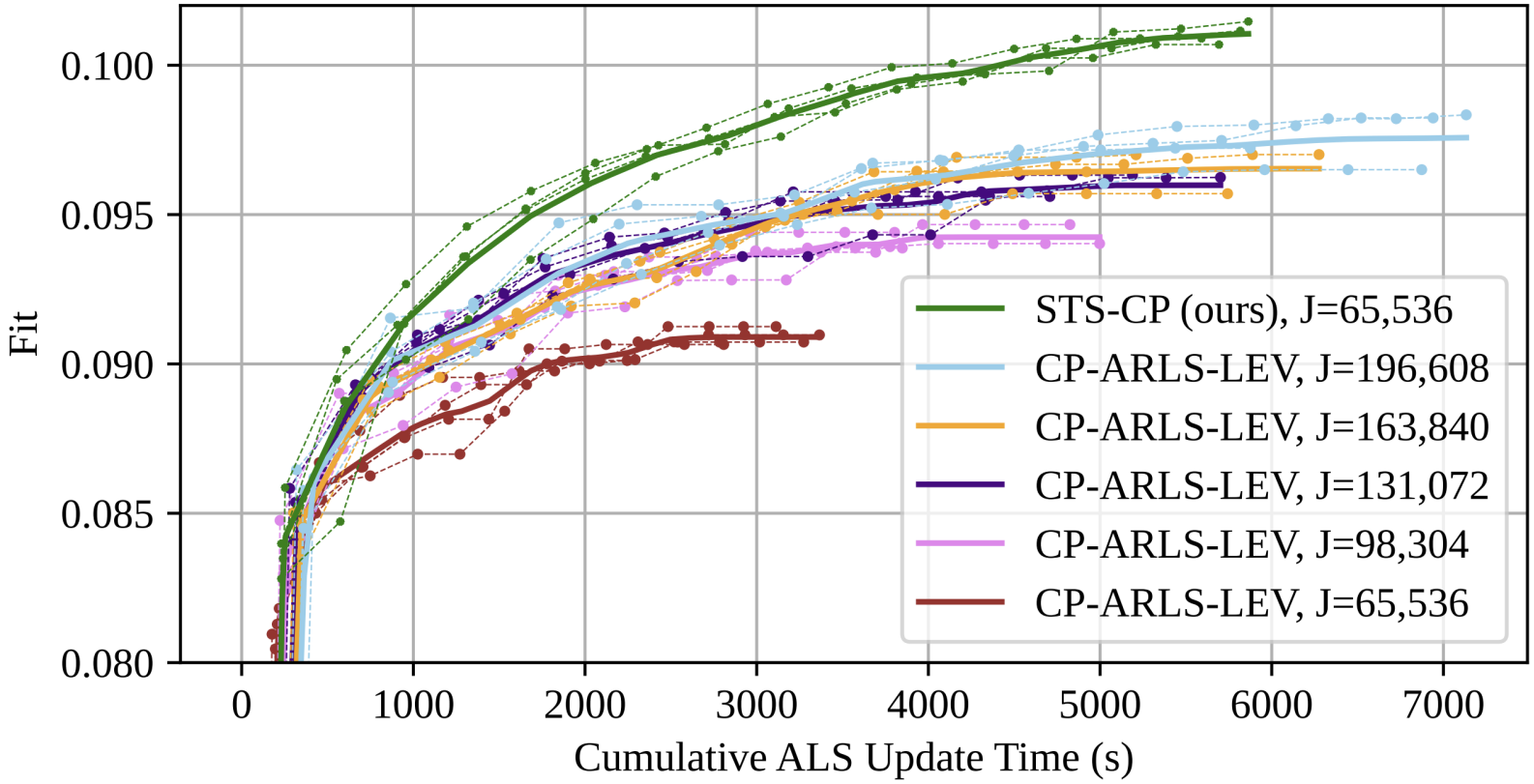
- **STS-CP** algorithm: lower asymptotic runtime for randomized CP decomposition than SOTA methods. Practical for sparse tensors w/ **billions of nonzeros.**

Fast Exact Leverage Score Sampling from Khatri-Rao Products with Applications to Tensor Decomposition. V Bharadwaj, OA Malik, R Murray, L Grigori, A Buluç, J Demmel. NeurIPS 2023

Accuracy Comparison for Fixed Sample Count

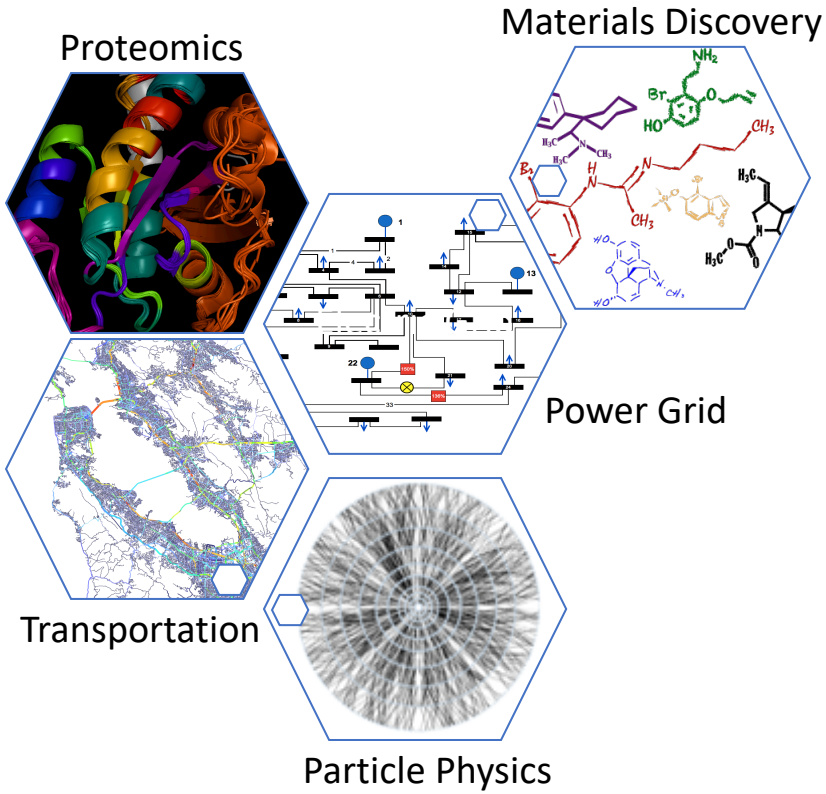


STS-CP Makes Faster Progress Towards Solution



Fit vs. ALS Update Time, Reddit Tensor, $R = 100$.

Graph Neural Networks (GNNs)

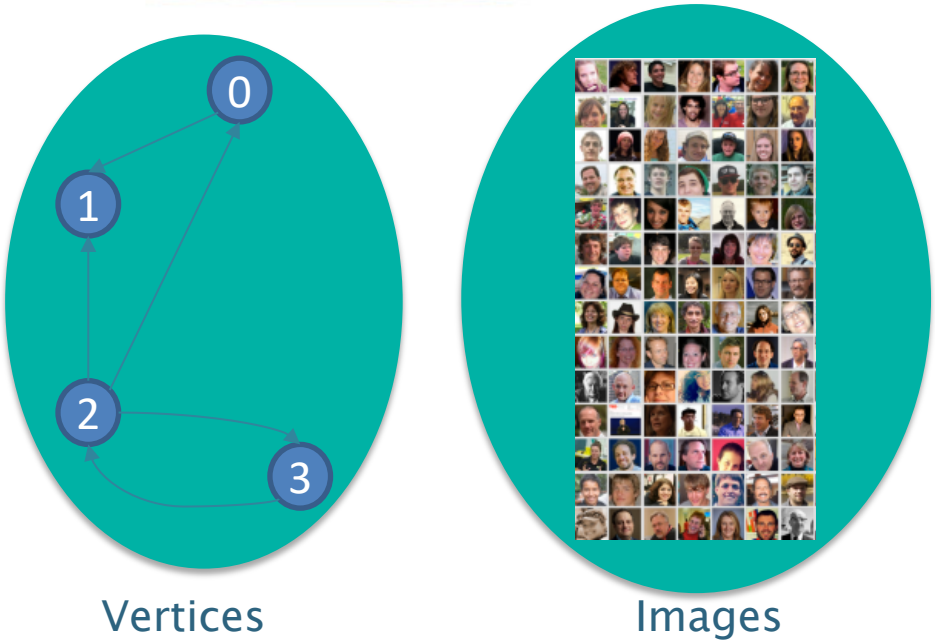


GNNs are finding success in many challenging scientific problems that involve interconnected data.

- 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.

Full-graph vs. mini-batch SGD



Full-graph training:

- Train on **entire** training set
- Slower convergence per epoch
- Faster training per epoch
- More memory hungry

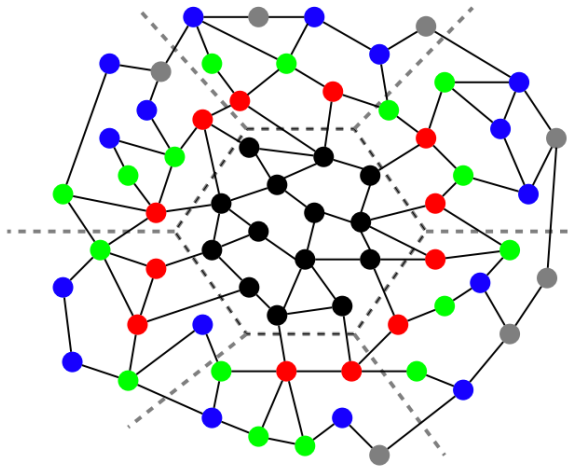
Mini-batch SGD:

- 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

No dependencies

sample



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?
- This talk will cover both full-graph training and sampling-based training
- CAGNET (Communication-Avoiding Graph Neural nETworks): https://github.com/PASSIONLab/CAGNET/

Graph convolution illustrated

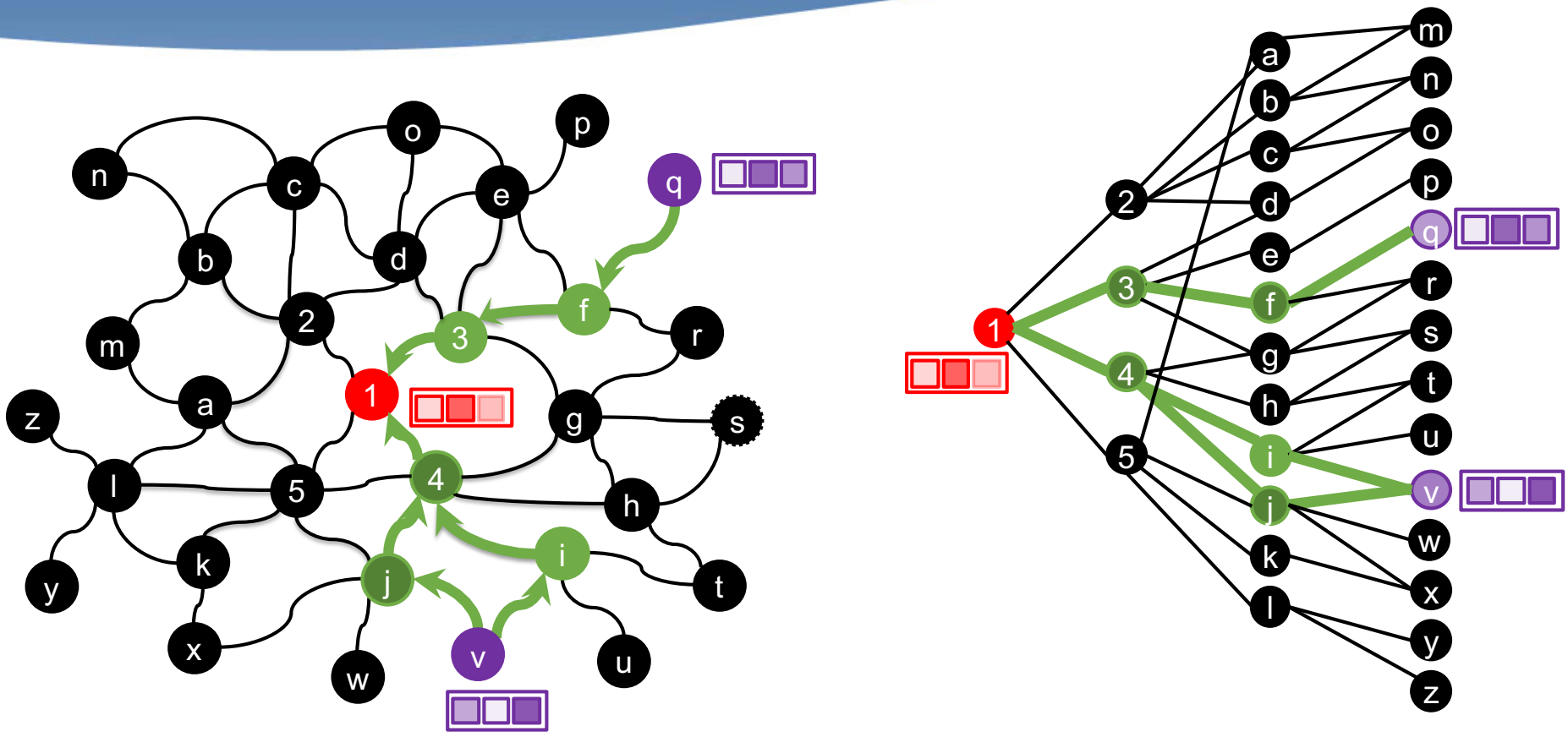


Illustration of the information flow in a Graph Neural Network (GNN). On the left is the graph in its natural form. The features (the shaded boxes) of vertices v and q are aggregated at vertex 1 through intermediate (green) vertices and edges. Features of other nodes are not shown but are also propagated. During training, the error is backpropagated in the opposite direction in the neural network, where each layer of the neural network propagates one hop of information.

Pattern 3: Sparse matrix times tall-skinny dense matrix (SpMM)

Feature aggregation from neighbors:

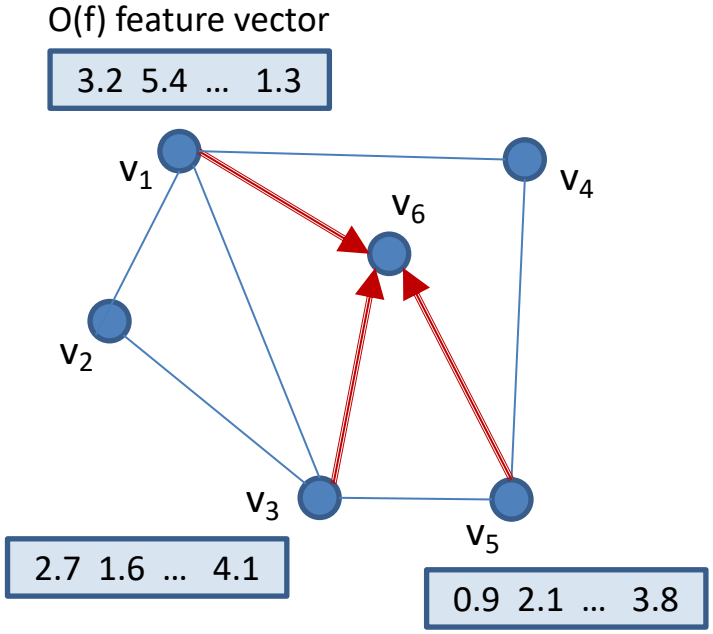
Used in Graph neural networks, graph embedding, etc.

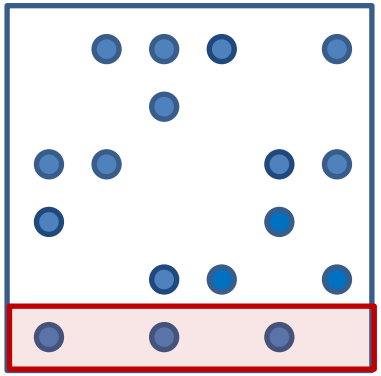
GrB_mxm(W, GrB_NULL, <semiring>, A, H, <desc>)

A: sparse adjacency matrix, n-by-n

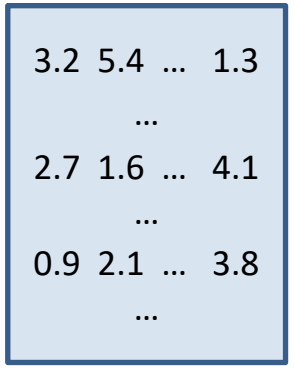
H: input dense matrix, n-by-f where f << n is the feature dimension

W: output dense matrix, new features





A^T



H

GCN 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
```

$A \in n\ x\ n$

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

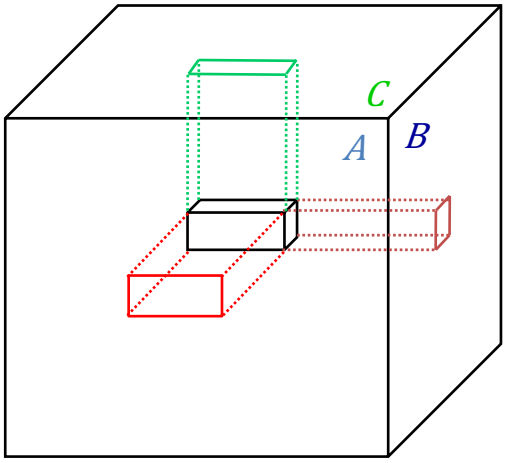
$G^l \in n\ x\ f^l$

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

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

The computation cube of matrix-matrix multiplication

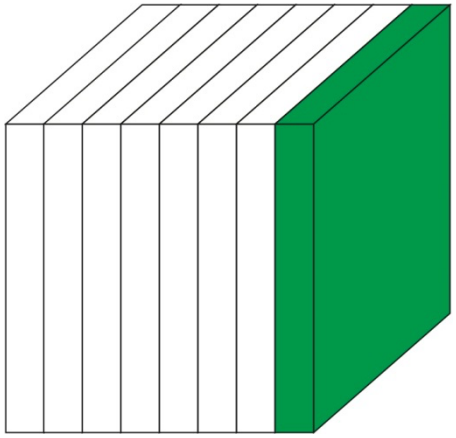
Matrix multiplication: $\forall(i,j) \in n \times n, \quad C(i,j) = \Sigma_k A(i,k)B(k,j),$



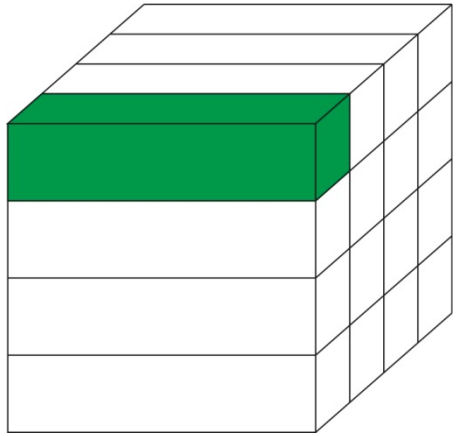
The *computation (discrete) cube*:

- A face for each (input/output) matrix
- A grid point for each multiplication

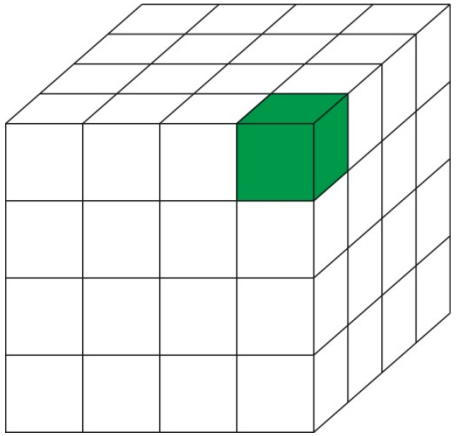
How about sparse algorithms?



1D algorithms

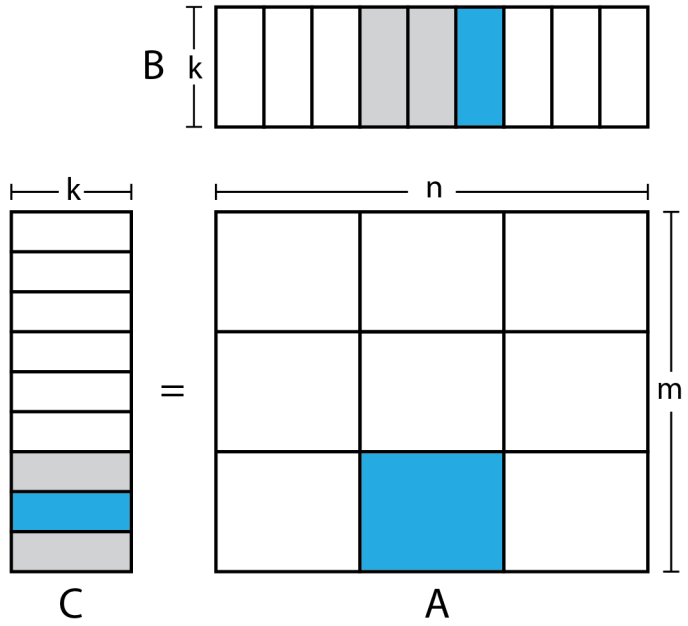


2D algorithms



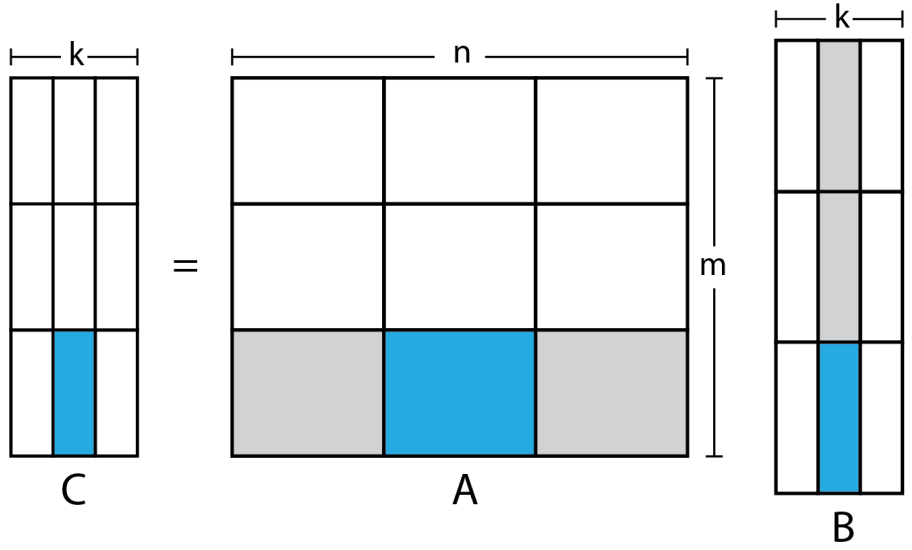
3D algorithms

Distributed SpMM algorithms



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

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



- 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

Distributed SpMM algorithms

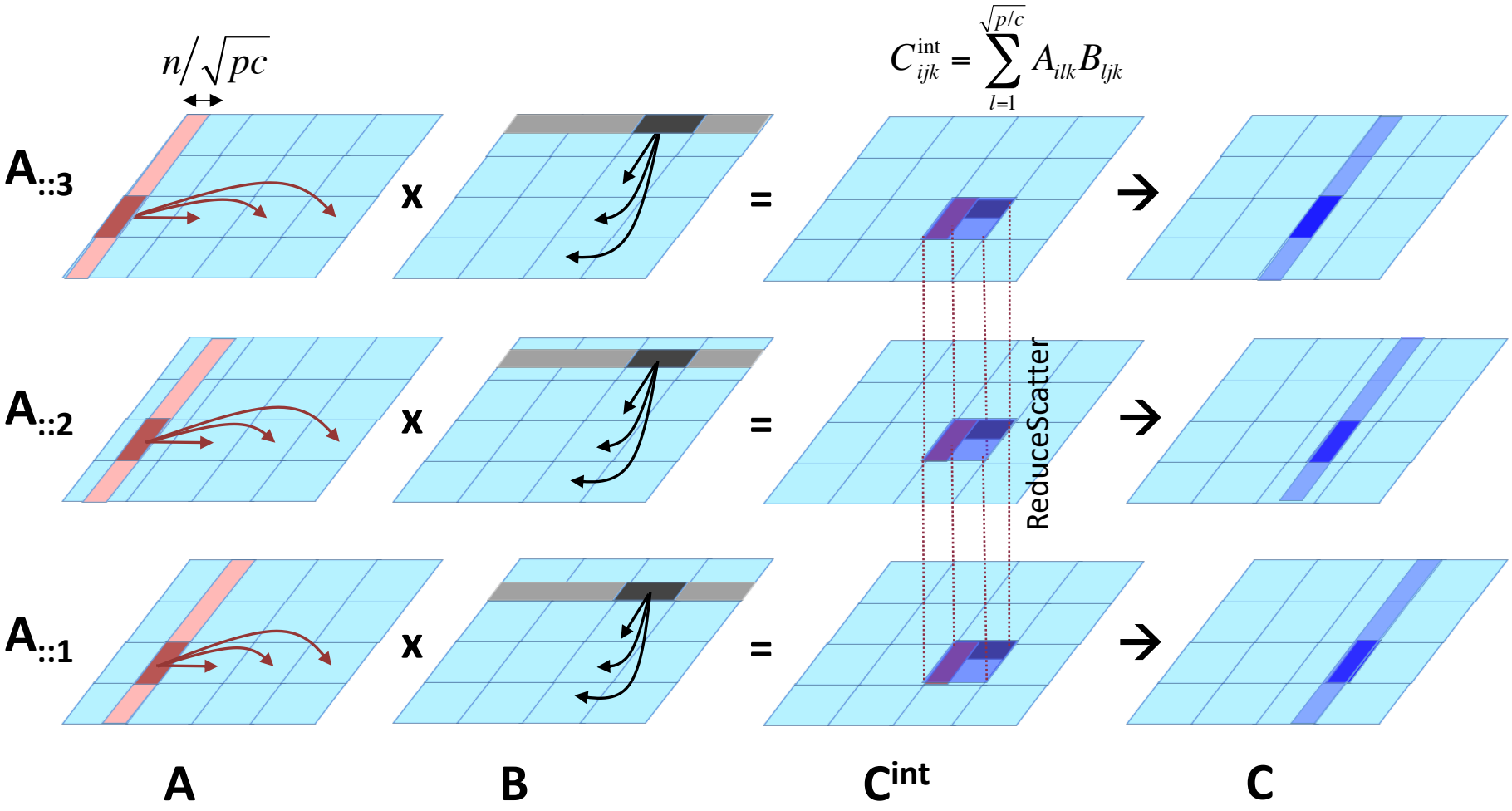


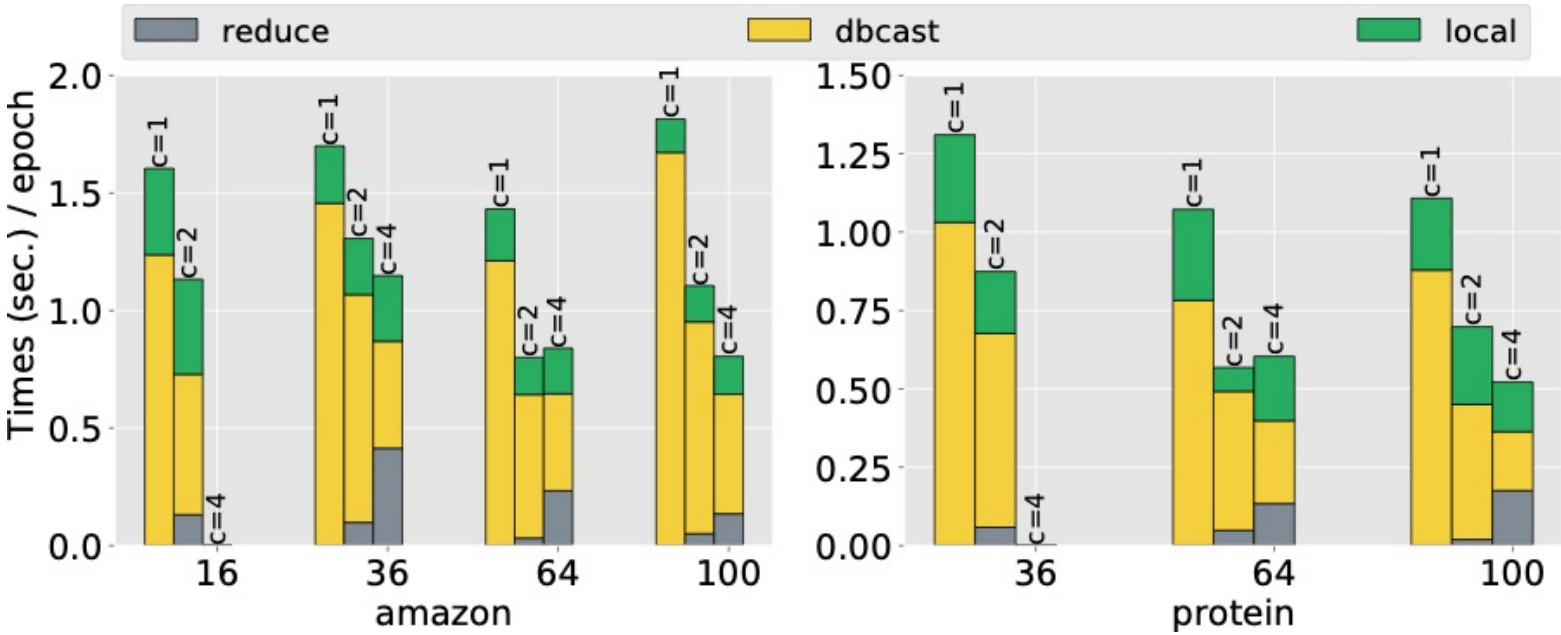
Illustration of the 3D algorithm on a $\sqrt{p} \times \sqrt{p} \times c$ grid

Communication analysis

CAGNET Cost Analyses (per process)			
Algorithm	Latency	Bandwidth	Memory
1D	$\lg P + 2P$	$2nf + f^2$	$\frac{nnz(\mathbf{A})+nfL}{P}$
1.5D	$2\frac{P}{c^2}\lg\frac{P}{c^2}$	$\frac{2nf}{c} + \frac{2nfc}{P}$	$\frac{nnz(\mathbf{A})+nfL}{P} + \frac{nfc}{P}$
2D	$5\sqrt{P} + 3\lg P$	$\frac{8nf}{\sqrt{P}} + \frac{2nnz(\mathbf{A})}{\sqrt{P}}$	$\frac{nnz(\mathbf{A})+nfL}{P}$
3D	$4P^{1/3}$	$\frac{2nnz(\mathbf{A})}{P^{2/3}} + \frac{12nf}{P^{2/3}}$	$\frac{nnz(\mathbf{A})+nfL}{P} + \frac{nfc}{P}$

Symbols and Notations	
Symbol	Description
$\mathbf{A}$	Modified adjacency matrix of graph $(n \times n)$
$\mathbf{H}^l$	Embedding matrix in layer l $(n \times f)$
$\mathbf{W}^l$	Weight matrix in layer l $(f \times f)$
$\mathbf{Y}^l$	Matrix form of $\frac{\partial \mathcal{L}}{\partial W^l_{ij}}$ $(f \times f)$
$\mathbf{Z}^l$	Input matrix to activation function $(n \times f)$
$\mathbf{G}^l$	Matrix form of $\frac{\partial \mathcal{L}}{\partial Z^l_{ij}}$ $(n \times f)$
σ	Activation function
f	Length of feature vector per vertex
f_u	Feature vector for vertex u
L	Total layers in GNN
P	Total number of processes
α	Latency
β	Reciprocal bandwidth

1.5D algorithm results for full-graph GCN 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)
- These results are from Summit at ORNL

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

Sketching Sparse Data with SpMM

Large sparse data matrix ***A*** is reduced to a smaller matrix via sketching in order to accelerate downstream computation, linear regression, low-rank approximation, full-rank matrix decomposition, trace estimation, graph sparsification, and more.

Assume ***A*** is an m-by-n tall skinny **sparse matrix** representing the data with m >> n. We want to apply a d-by-m sketching matrix ***S*** that is dense. The entries of ***S*** can be random Gaussian, uniform over (−1,1), or simply ± 1.

Want: fast kernel for computing the dense-sparse matrix matrix product (SpMM):

$$\widehat{\boldsymbol{A}} = \boldsymbol{S}\,\boldsymbol{A}$$

One can naïvely generate the dense matrix and use an optimized SpMM kernel for this operation. In this case, S might not even fit in memory.

Central Challenge: How to use on-demand random number generation to convert a portion of memory movement cost into computation cost?

Tianyu Liang, Riley Murray, Aydın Buluç, and James Demmel. Fast multiplication of random dense matrices with fixed sparse matrices. In International Parallel and Distributed Processing Symposium (IPDPS). IEEE, 2024.

Classes of GNN Samplers

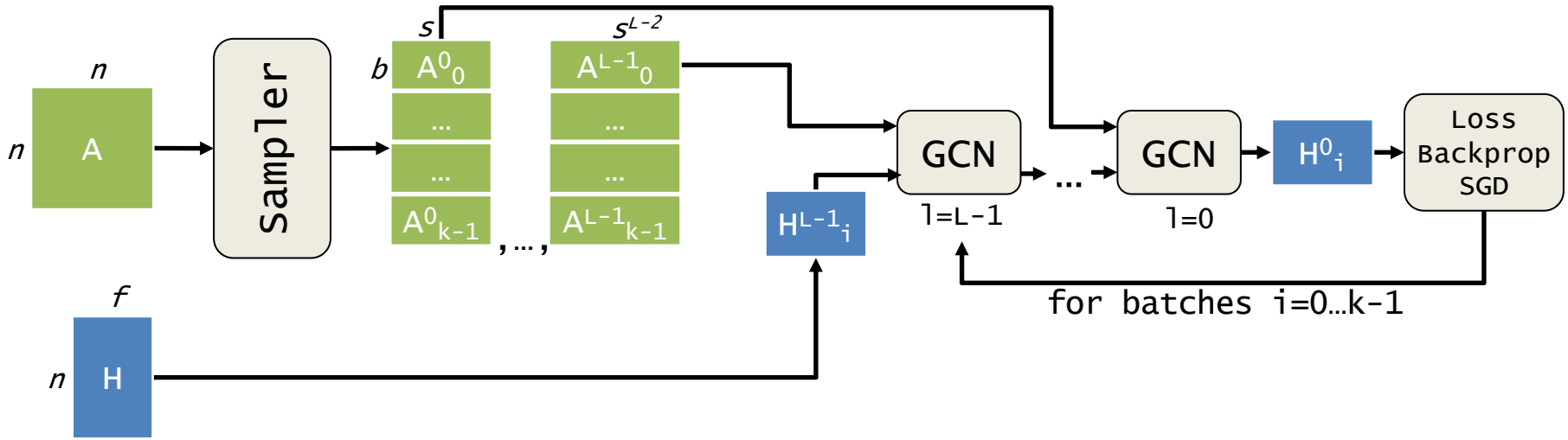
Node-wise: Sample *k* neighbors per vertex in batch
- » GraphSAGE, PinSAGE
- » Distributed CPU and Single-node GPU implementations exist

Layer-wise: Sample *k* neighbors in aggregated neighborhood of batch
- » FastGCN, LADIES
- » Single-node GPU implementation exists

Graph-wise: Sample *k* vertices in graph, use induced subgraph as batch
- » No current distributed CPU or GPU implementation exists

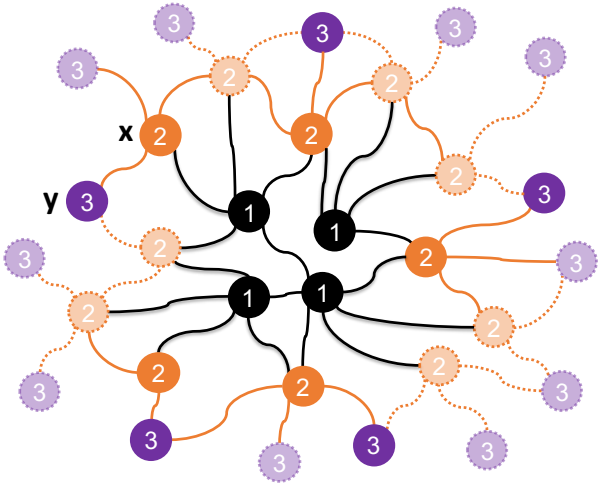
My talk from now on will focus on **distributed GPU implementations** of node-wise and layer-wise sampling, by using communication-avoiding sparse-matrix multiplication

Full sampling-based training pipeline

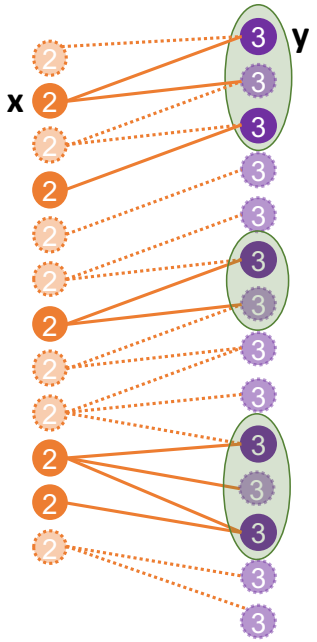


b: number of vertices in each batch
s: number of vertices to be sampled per vertex (GraphSAGE) or layer (LADIES)
f: number of features
n: number of total vertices
k: number of batches to be sampled at once

Graph Sampling in GNN training



Black nodes are the mini-batch of source vertices chosen for that training step.



Layer-dependent sampled bipartite graph for the 2nd layer of GCN.

Vertices in adj(S_2) are highlighted with green halos around them on the right side of the bipartite graph.

Procedure A_s = SAMPLE(A, Q):

$\mathbf{P} \leftarrow \mathbf{QA}$ ← Compute probabilities via SpGEMM

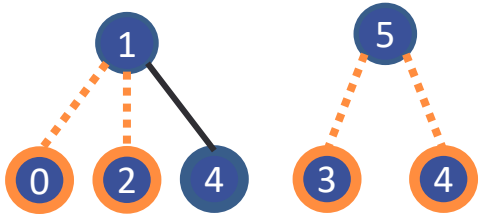
$\hat{\mathbf{Q}} \leftarrow \text{SAMPLE}(\mathbf{P})$ ← Independent rejection sampling per row

$\hat{\mathbf{A}} \leftarrow \text{rowsel}(\mathbf{Q})\mathbf{A}$ ← Row extraction

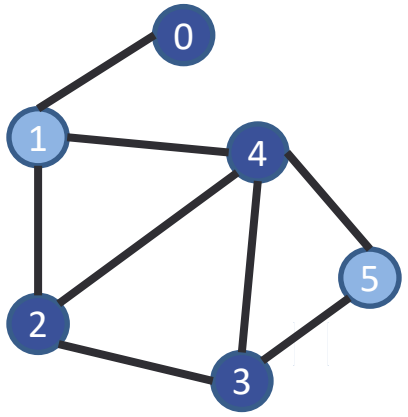
$\mathbf{A_S} \leftarrow \hat{\mathbf{A}}\ \text{colsel}(\hat{\mathbf{Q}})$ ← Column extraction

} **SpGEMM**

GraphSAGE sampler

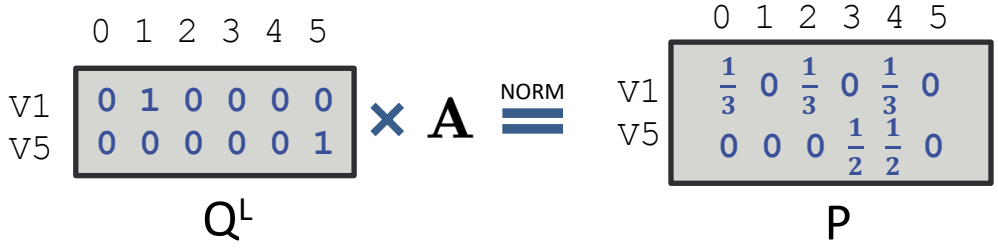


Mini-Batch



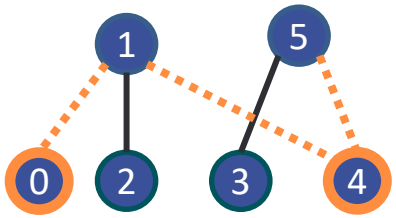
Graph G

- GraphSAGE samples *s* neighbors per vertex in batch u.a.r.
- We compute **each vertex's probability distribution** with SpGEMM

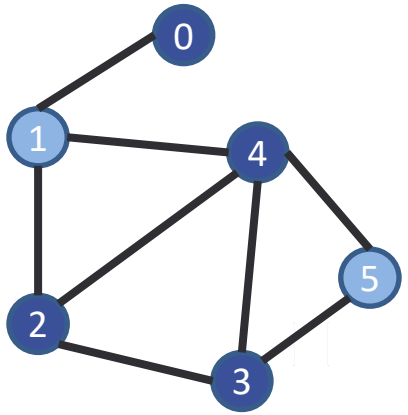


SpGEMM with A (adjacency matrix of G)

LADIES sampler



Mini-Batch

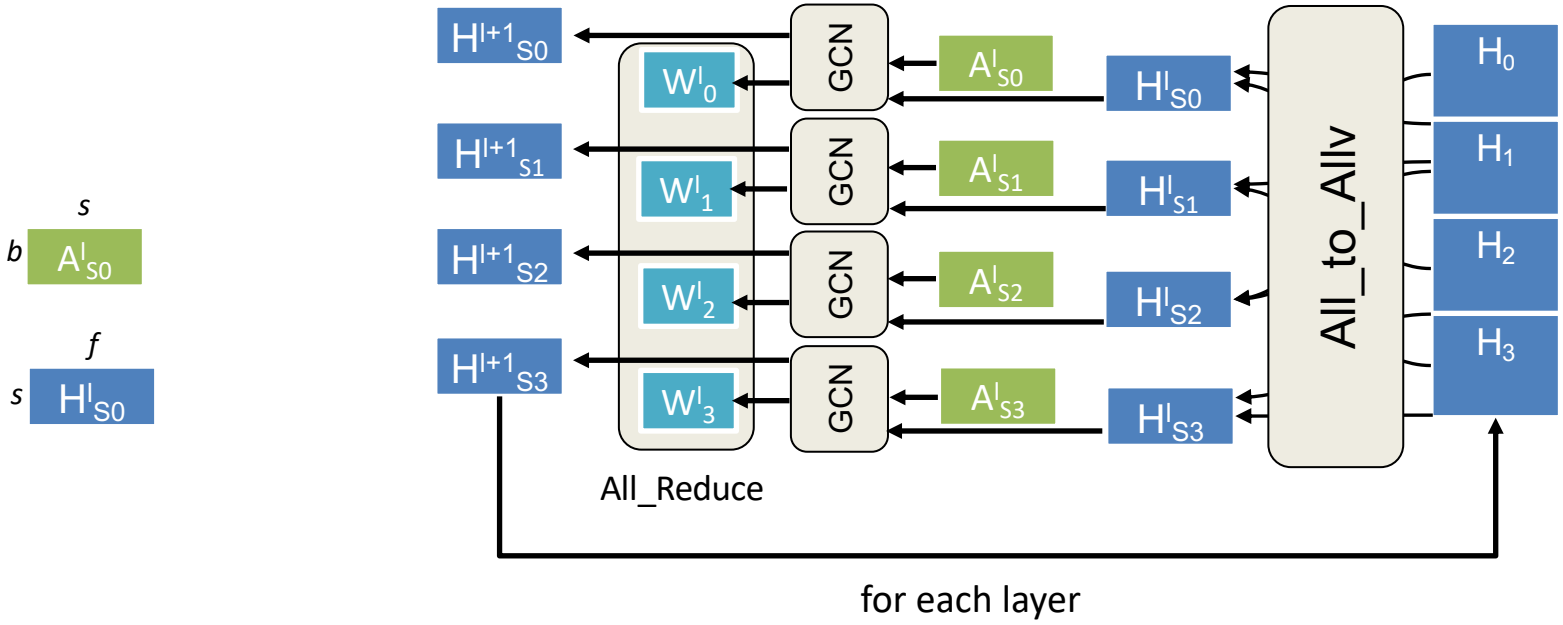


Graph G

- LADIES samples *s* neighbors in **aggregated neighborhood of batch**
- We compute **each batch's probability distribution** with SpMSpV
- We compute distributions of **k batches at once**, via SpGEMM

SpMSpV with A (adjacency matrix of G)

Full sampling-based training pipeline (w/ parallelism)



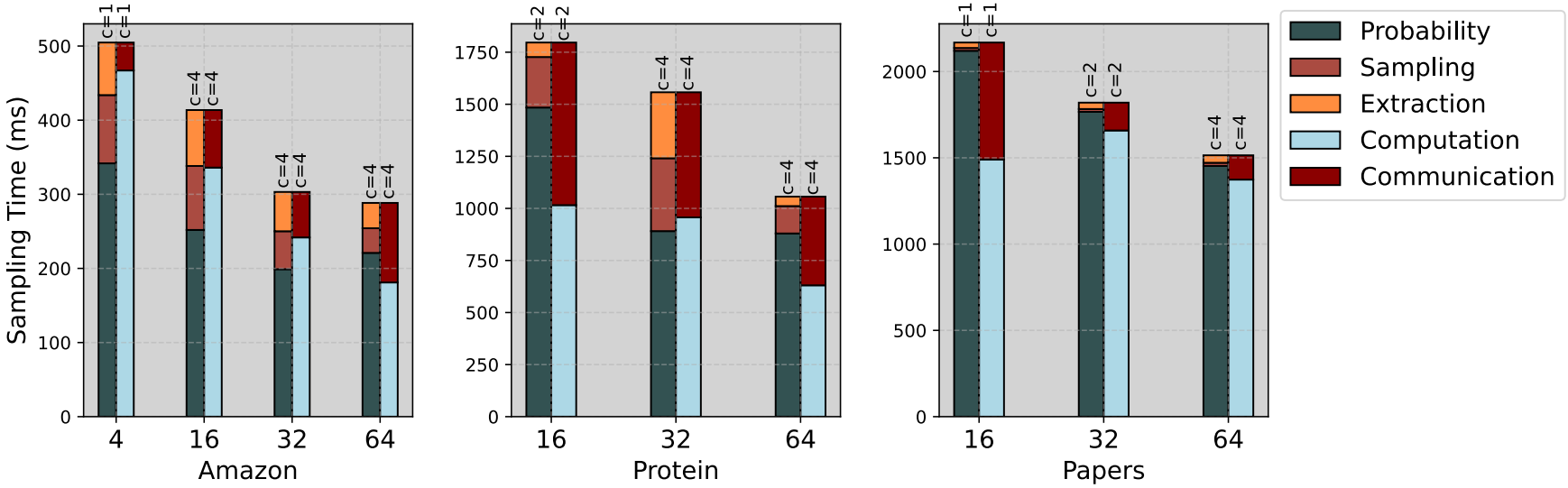
b: number of vertices in each batch
s: number of vertices to be sampled per vertex (GraphSAGE) or per layer (LADIES)
f: number of features
n: number of total vertices
k: number of batches to be sampled at once

Implementation details

- PyTorch 1.13 with NCCL 2.9 backend
  - » Kipf-Welling GCN model (2-layers, 16 hidden activations)
- System:
  - » Perlmutter at NERSC/LBL
  - » 4 NVIDIA A100s per node
- Sampling Baseline:
  - » Quiver (v0.1.1)[1]
  - » Single-node, multi-GPU GNN library on top of PyTorch Geometric[2]
  - » Supports GraphSAGE sampling
- Datasets:

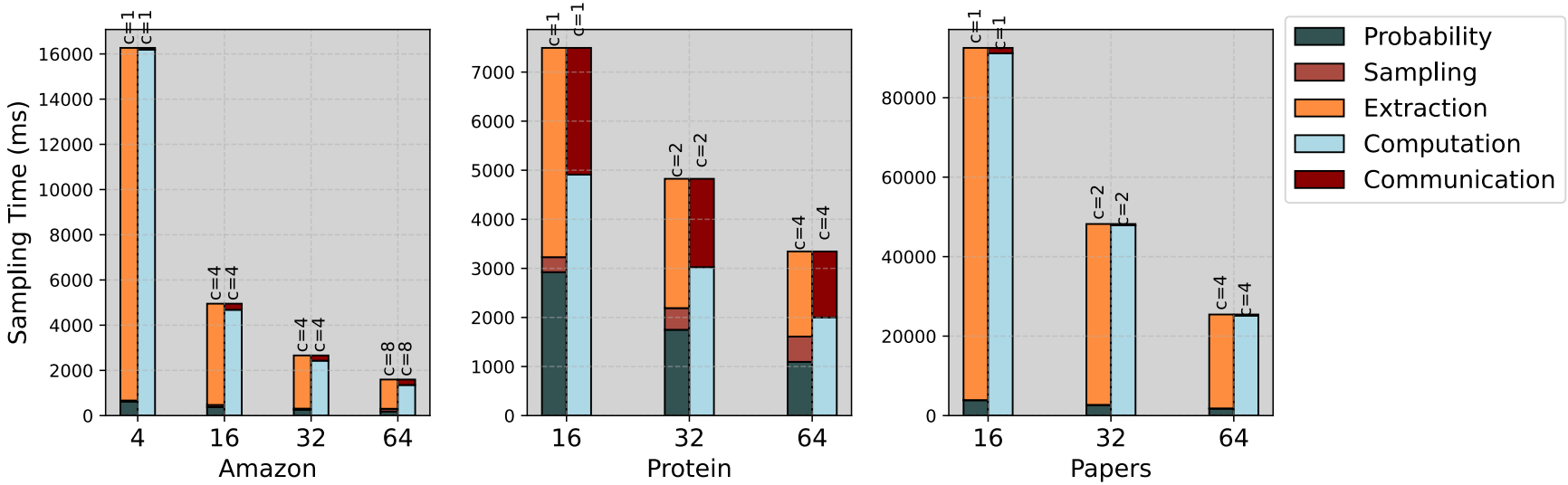
Name	Vertices	Edges	Features	Labels
Amazon	14M	231M	300	24
Protein	8M	2B	128	256
Papers	111M	1.6B	128	172

Distributed GraphSAGE sampling with 1.5D SpGEMM



- Batch size = 256, Sample number = 10, Minibatch Count = n / 256
- Speedup over Quiver for same GPU count (40X for Amazon, 3X for Protein)
  - » Quiver iterates over minibatches to sample
  - » We can sample a bulk set of minibatches with a larger **Q** matrix
- For appropriate replication factor, scales across process count

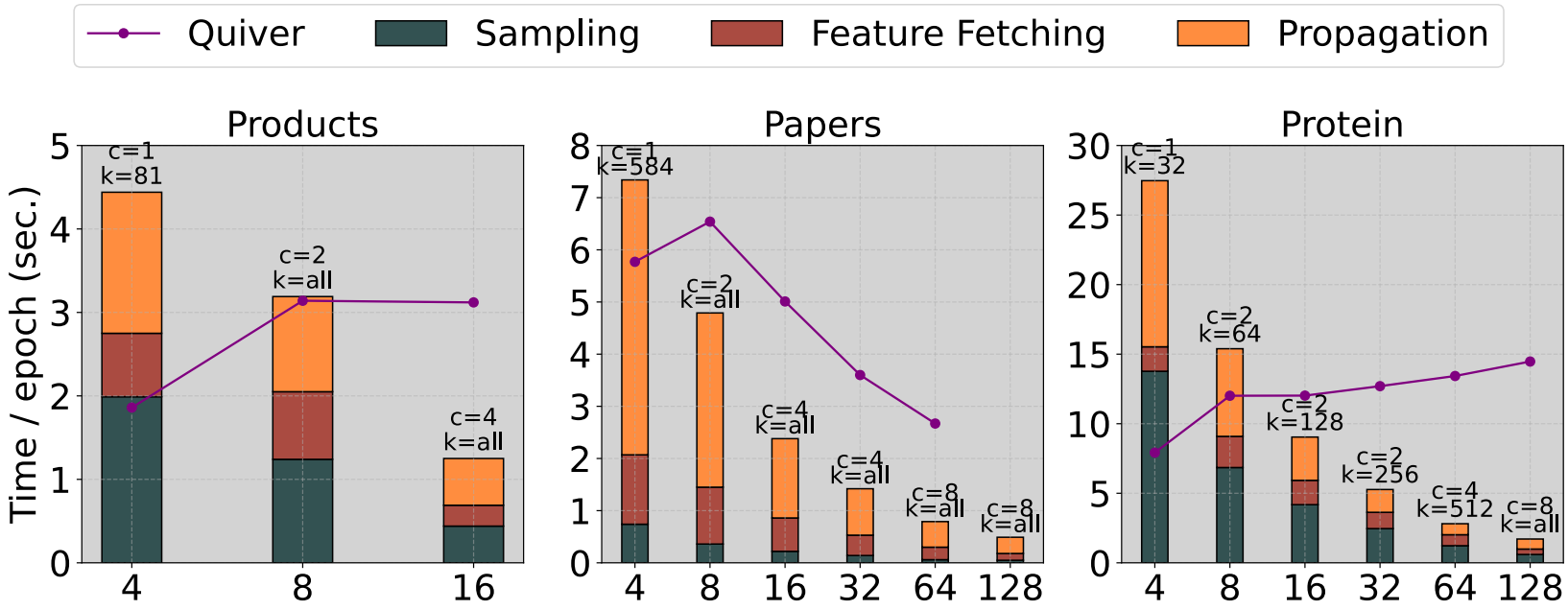
Distributed LADIES sampling with 1.5D SpGEMM



- Batch size = 256, Sample number = 10, Minibatch Count = n / 256
- Like GraphSAGE, we can sample a bulk set of minibatches with a larger matrix
- For appropriate replication factor, scales linearly across process count

Alok Tripathy, Katherine Yelick, Aydın Buluç. Distributed Matrix-Based Sampling for Graph Neural Network Training. MLSys 2024 (to appear)

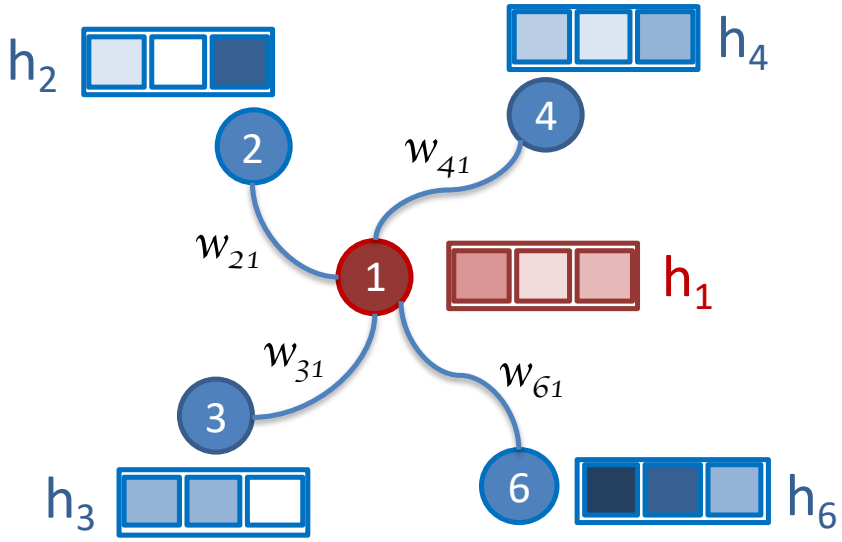
What if we replicate the graph topology?



- Performance of our graph replication-based algorithm.
- We show speedups over Quiver on large GPU counts on each dataset.
- Quiver's preprocessing step ran out of memory on Papers with 128 GPUs, so we do not include a Quiver datapoint there.

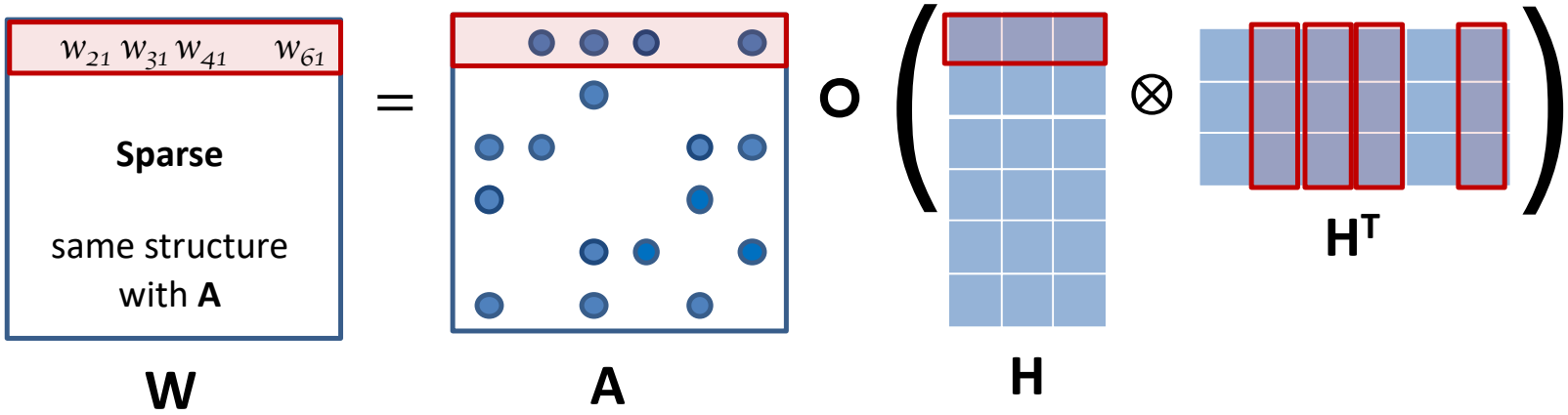
Alok Tripathy, Katherine Yelick, Aydın Buluç. Distributed Matrix-Based Sampling for Graph Neural Network Training. MLSys 2024 (to appear)

Pattern 4: Sampled dense-dense matrix Multiplication (SDDMM)



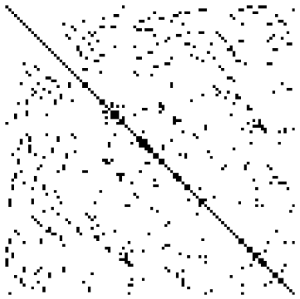
Graph attention: making edge weights learnable

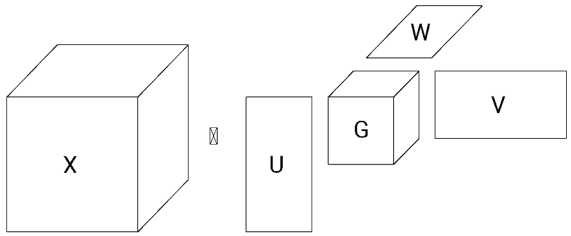
GrB_mxm(W, A, H, H, …);

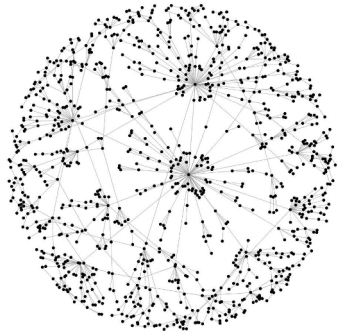


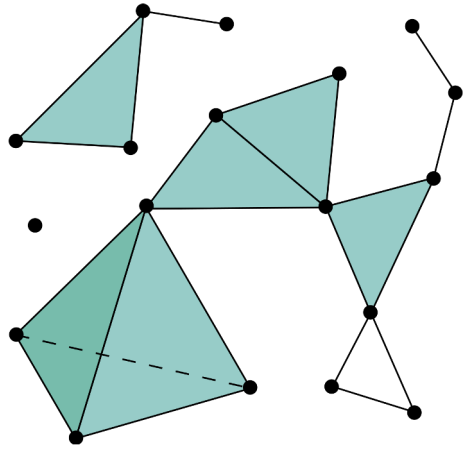
Sparsitute MMICCs center

- **Sparsitute**: A mathematical Institute for Sparse Computations in Science and Engineering is a new Mathematical Multifaceted Integrated Capability Center (MMICC) funded by the US Department of Energy.
- Sparsitute brings leading researchers across the nation working on various aspects of sparsity together to accelerate progress and impact.
- Its research agenda aims to advance the state-of-the-art in sparse computations both as a unified topic and within three broad pillars: **sparse and structured matrix computations, sparse tensor computations,** and **sparse network computations**, as well as their interconnections.









http://sparsitute.lbl.gov

Conclusions

- **Sparsity** is a common problem in computational science and machine learning and is of interest to many challenges in high-energy physics, nuclear and plasma physics, power grid analysis, biology, traffic modeling and quantum chemistry.
- The overarching strategic goal of our research is to provide an ***integrated treatment of sparsity across different applied math research areas***, to ***increase the profile of research in sparse computations***, and to make a ***long-lasting impact on science applications***
- Extreme parallelism and extremes-scale 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 and tensors** will be the key to overcome these adverse technological trends

Key acknowledgments

Ariful Azad

Vivek Bharadwaj

James Demmel

Saliya Ekanayake

Giulia Guidi

Nikos Kyrpides

Georgios Pavlopoulos

Oguz Selvitopi

Alok Tripathy

Katherine Yelick