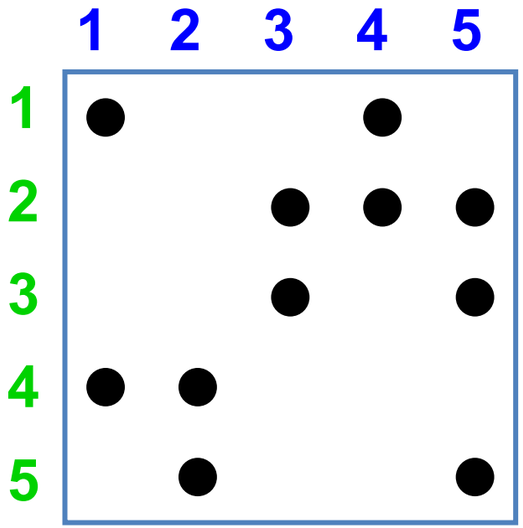
Communication-Avoiding Sparse Matrix Algorithms for Large Graph and Machine Learning Problems

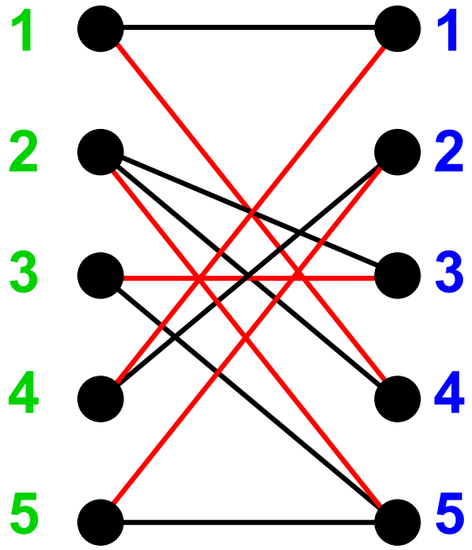
Aydın Buluç
Computational Research Division, LBNL
EECS Department, UC Berkeley

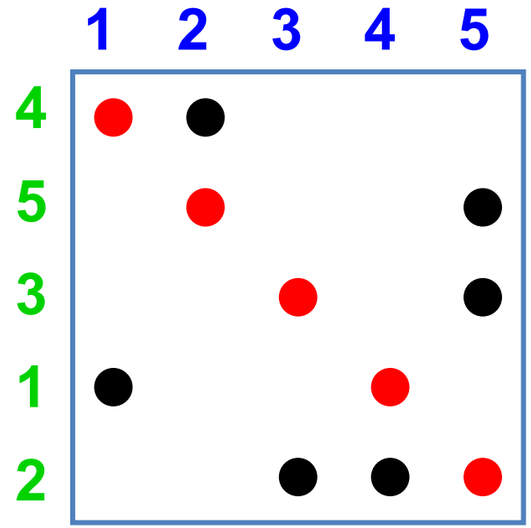
February 28, 2019
SIAM CSE, Spokane, WA

Large Graphs in Traditional Scientific Computing



A





PA

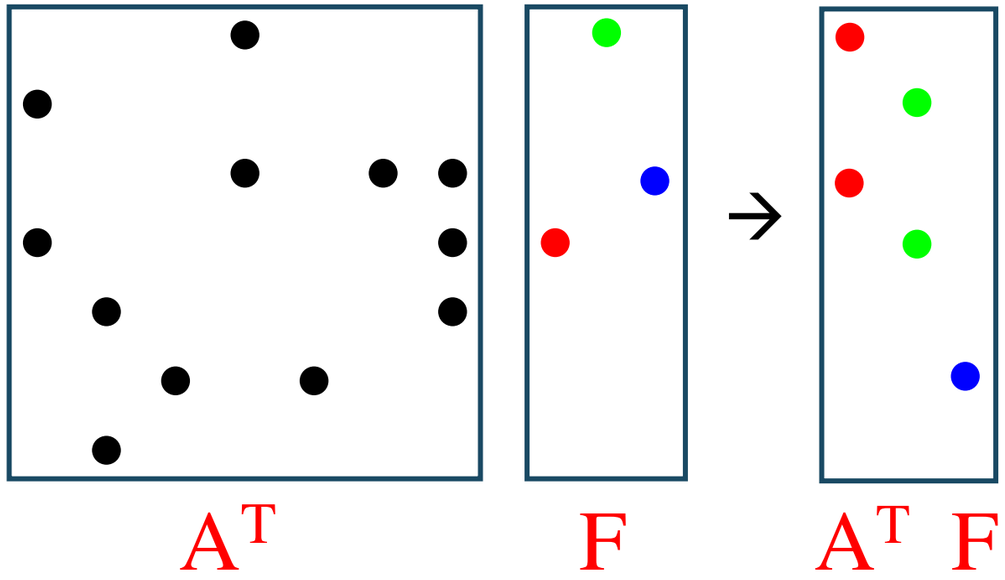
Matching in bipartite graphs: Permuting to heavy diagonal or block triangular form

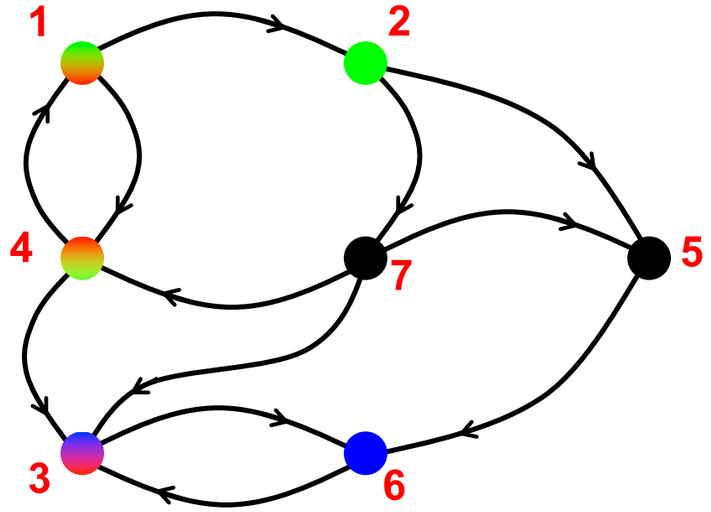
What are sparse matrices?

"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

Graphs in the language of sparse matrices

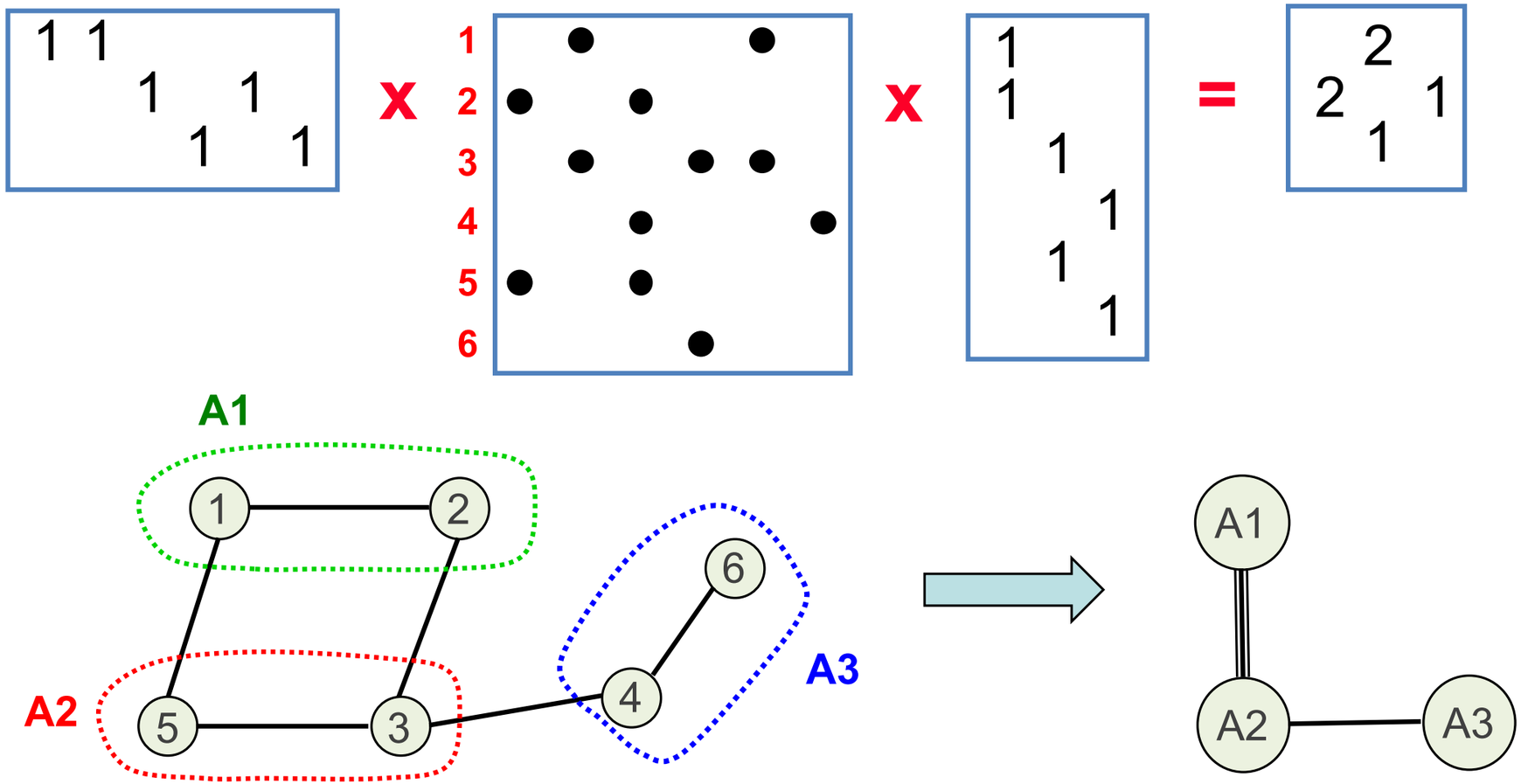




- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism:
  - searches, vertices, edges

This multi-source breadth-first traversal leads to a highly-parallel implementation for betweenness centrality (measure of influence in graphs, based on shortest paths)

Graph coarsening via sparse matrix products



Buluç and Gilbert. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. SISC, *2012*.

The case for sparse matrices

Many irregular applications contain coarse-grained parallelism that can be exploited by abstractions at the proper level.

More on http://graphblas.org

Traditional graph computations	Graphs in the language of linear algebra
Data driven, unpredictable communication.	Fixed communication patterns
Irregular and unstructured, poor locality of reference	Operations on matrix blocks exploit memory hierarchy
Fine grained data accesses, dominated by latency	Coarse grained parallelism, bandwidth limited

"Our mission is to build up a linear algebra sense to the extent that vector-level thinking becomes as natural as scalar-level thinking."

- Charles Van Loan

GraphBLAS Status: C API 1.2 released and in use

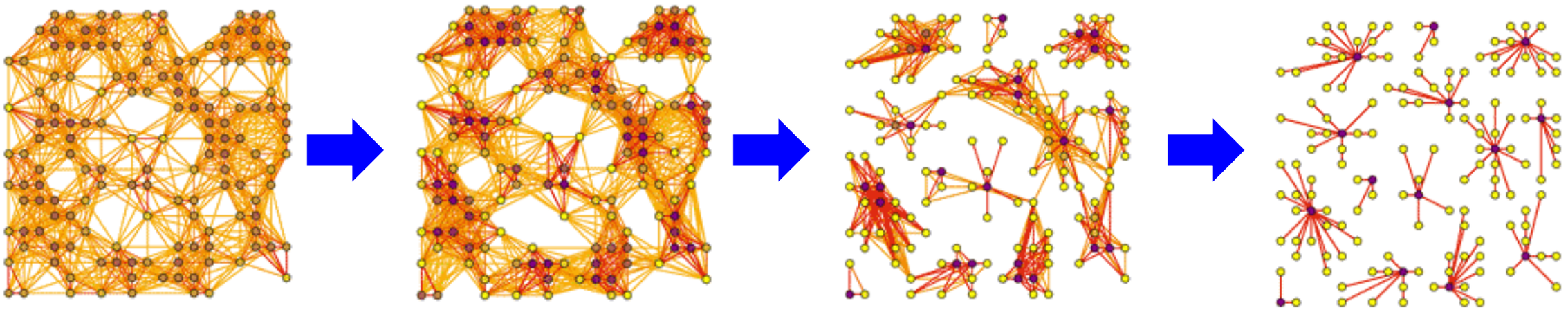
- Implementations of the GraphBLAS C specification:
  - SuiteSparse http://faculty.cse.tamu.edu/davis/suitesparse.html
  - IBM https://github.com/IBM/ibmgraphblas
  - Test suite for validating an implementation of the C-spec from SEI/CMU (… to be released "soon")

- Systems using the GraphBLAS
  - RedisGraph v1.0 preview release:
    - RedisGraph is a graph database architecture implemented as a Redis Module, using GraphBLAS sparse matrices for internal data representation and linear algebra for query execution.
    - https://redislabs.com/blog/release-redisgraph-v1-0-preview/
  - Lincoln Labs GraphProcessor designed around the GraphBLAS.

- C++ bindings to the GraphBLAS
  - GBTL from SEI/CMU: https://github.com/cmu-sei/gbtl
  - GraphBLAST (GPUs): https://github.com/gunrock/graphblast

Protein Clustering with Markov Clustering (MCL)

MCL is a widely popular and successful algorithm for discovering clusters in protein interaction and protein similarity networks. It is robust against perturbations



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

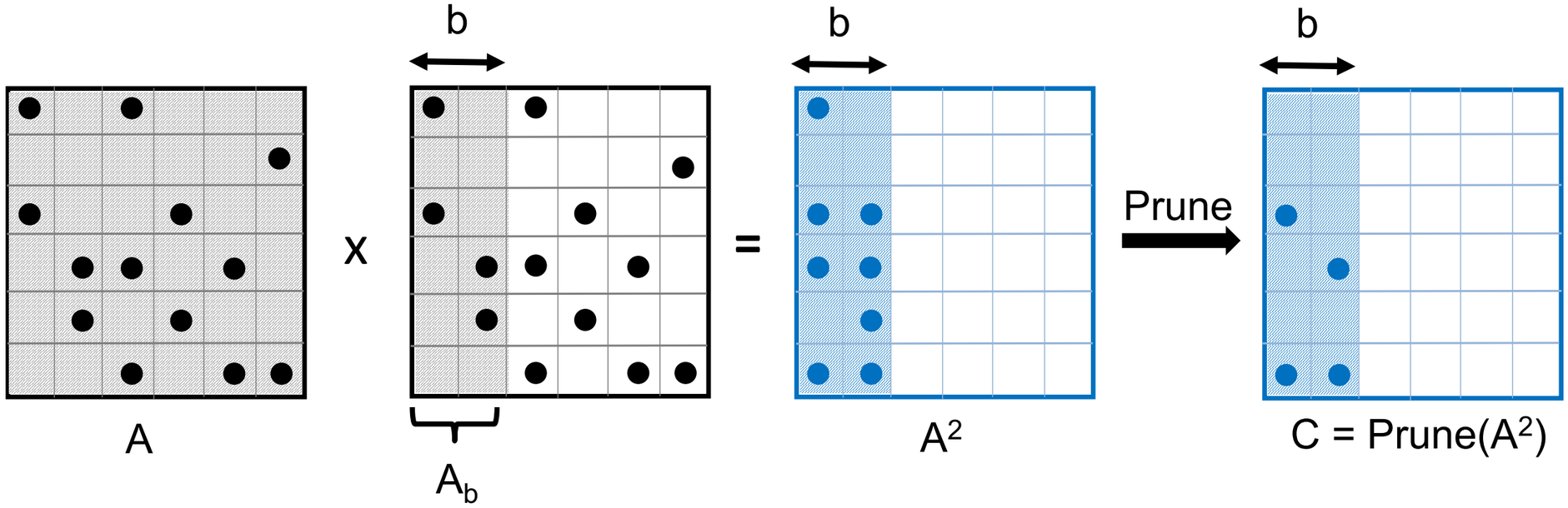
Step 2 (Inflation) : taking powers entry-wise

After Convergence:

- Find connected components of the final graph (matrix)
- Return them as clusters

HipMCL: Combined Expand, Inflate, Prune Cycle

- HipMCL is a high-performance distributed memory parallel implementation of MCL
- HipMCL relies on state of the art parallel sparse matrix operations of **Combinatorial BLAS**



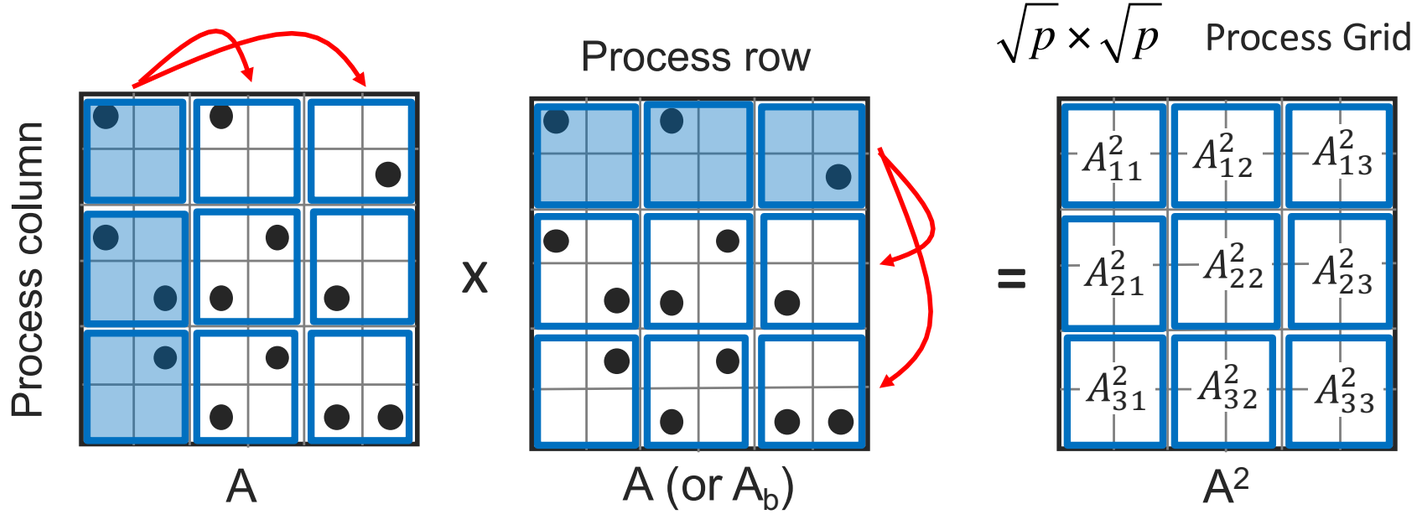
b: number of columns constructed at once, dynamically selected based on memory

- Smaller b: less parallelism, memory efficient (b=1 ~ SpMSpV used in MCL)
- Larger b: more parallelism, memory intensive (also more communication-efficient)

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

Parallel Sparse Matrix Products in HipMCL

- 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.



- HipMCL 1.0 uses a 2D algorithm for computing sparse matrix products (SpGEMM)
- It uses a memory scalable multithreaded heap-based implementation for in-node SpGEMM

HipMCL Impact and Future Work

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

Impact:

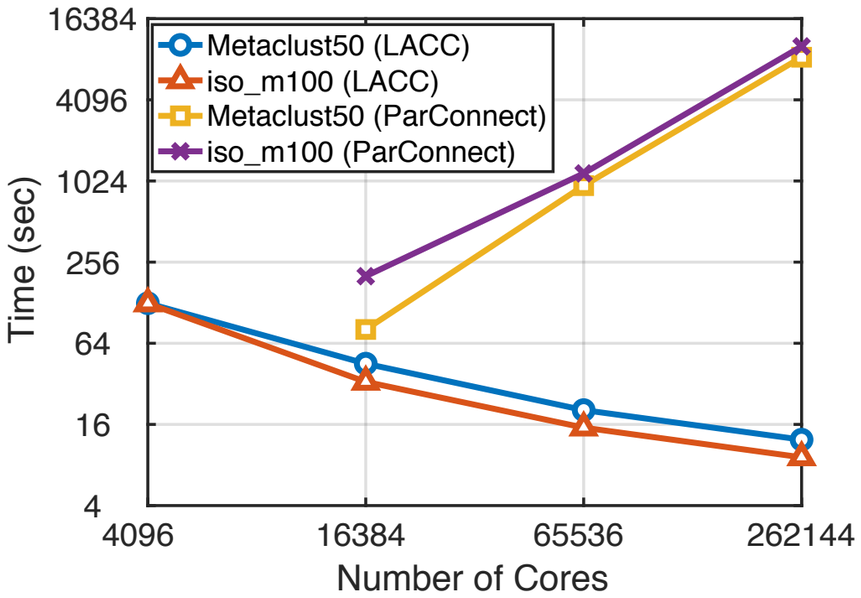
- MCL can not cluster these networks
- Several novel protein families from metagenomic data are also discovered using HipMCL

Future/ongoing work:

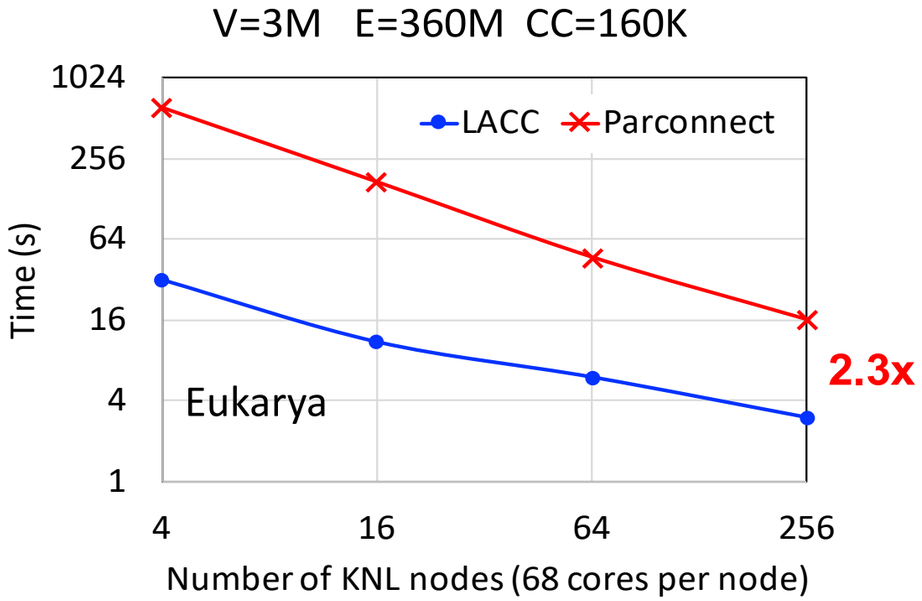
- Generating protein similarity networks directly in distributed memory
- GPU support and integration of 3D SpGEMM to HipMCL.

LACC: Linear Algebraic Connected Components

Parallel connected components for cluster identification (after MCL iterations converge): Awerbuch-Shiloach algorithm using SpMSpV and a few other GraphBLAS operations



- Exploiting sparsity was the key to performance and scaling
- Implemented using Combinatorial BLAS

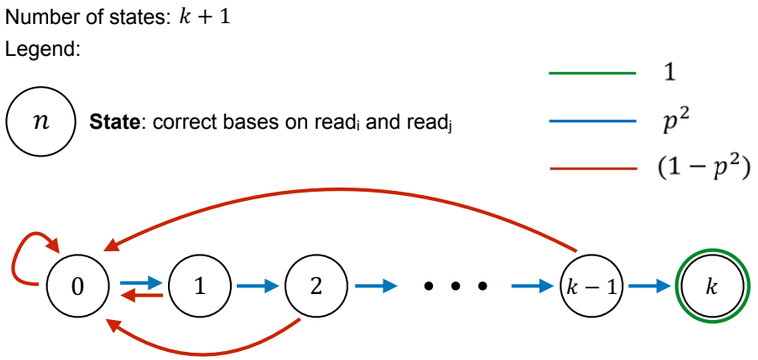


Impact: More than 2x faster connected component identification across different scales. Orders of magnitude faster at large concurrencies, enabling continued scaling of HipMCL by removing this potential Amdahl's bottleneck.

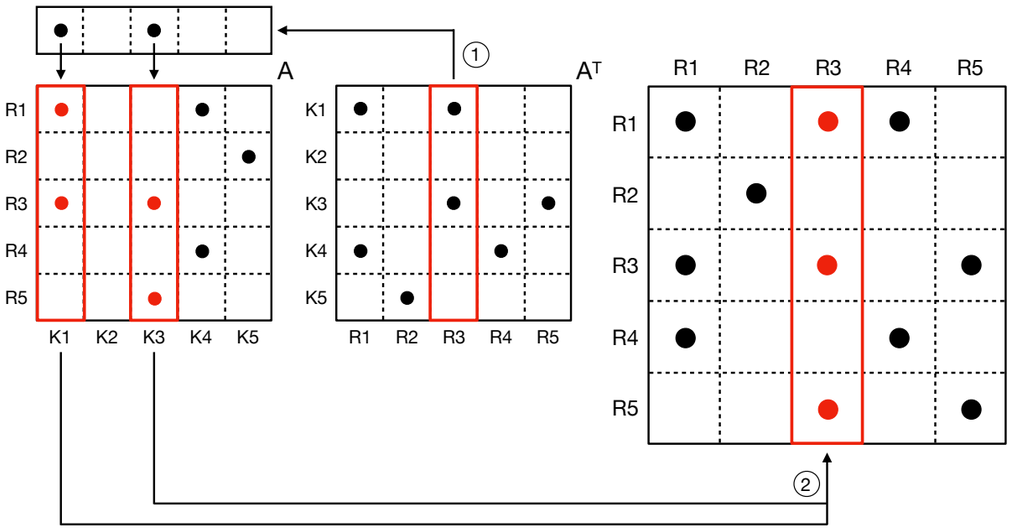
Azad, Buluç. LACC: a linear-algebraic algorithm for finding connected components in distributed memory (*IPDPS 2019*)

BELLA: Berkeley Long-read to Long-read Aligner

- **Long reads** from PacBio and Oxford Nanopore have the potential to revolutionize de-novo assembly
- **Overlap-Consensus-Layout** paradigm is more suitable than de Bruijn graph paradigm.
- **Overlapping** is the most computationally expensive step.



Read overlapping using shared k-mers is computing a **sparse matrix product**, for which we know good algorithms and implementations

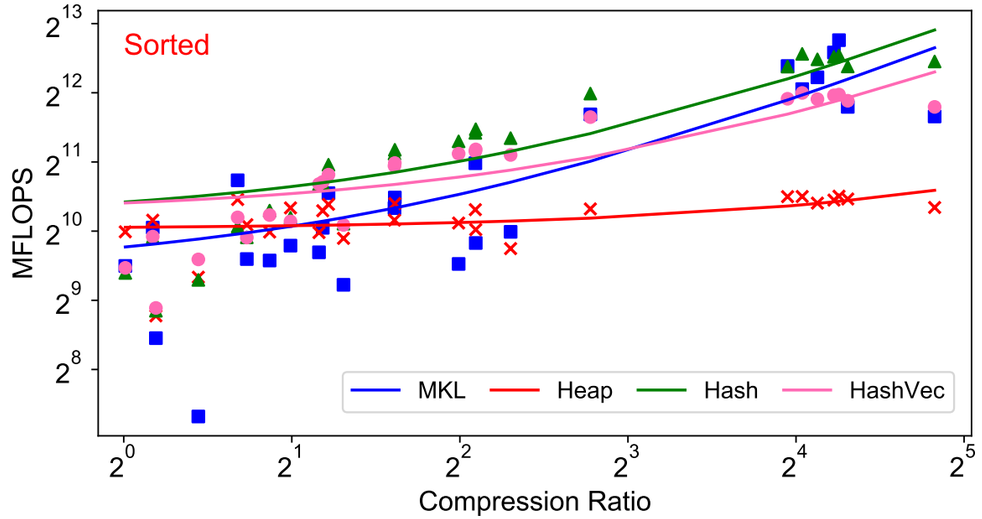


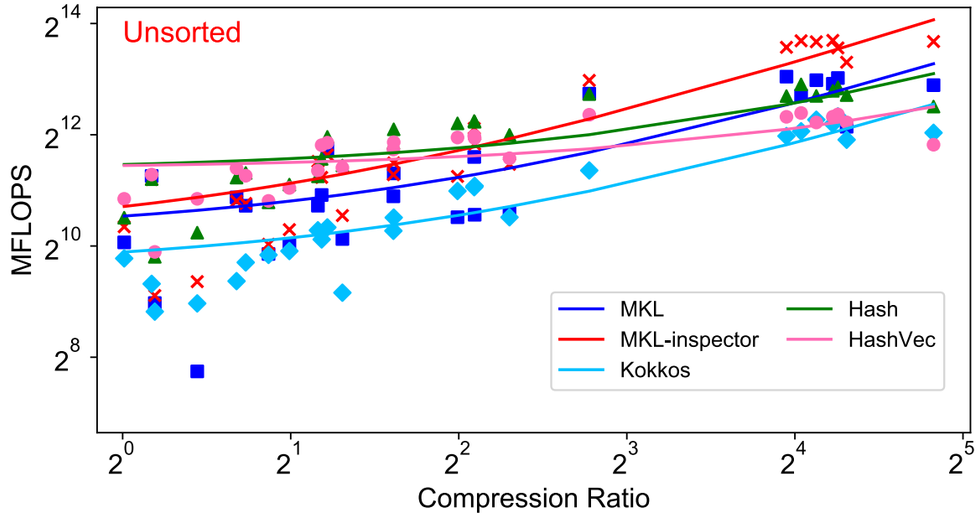
BELLA addresses:

- 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, Ellis, Rokhsar, Yelick, Buluç. *BELLA: Berkeley efficient long-read to long-read aligner and overlapper*. (bioRxiv, 2018)

New Shared-Memory SpGEMM Kernels





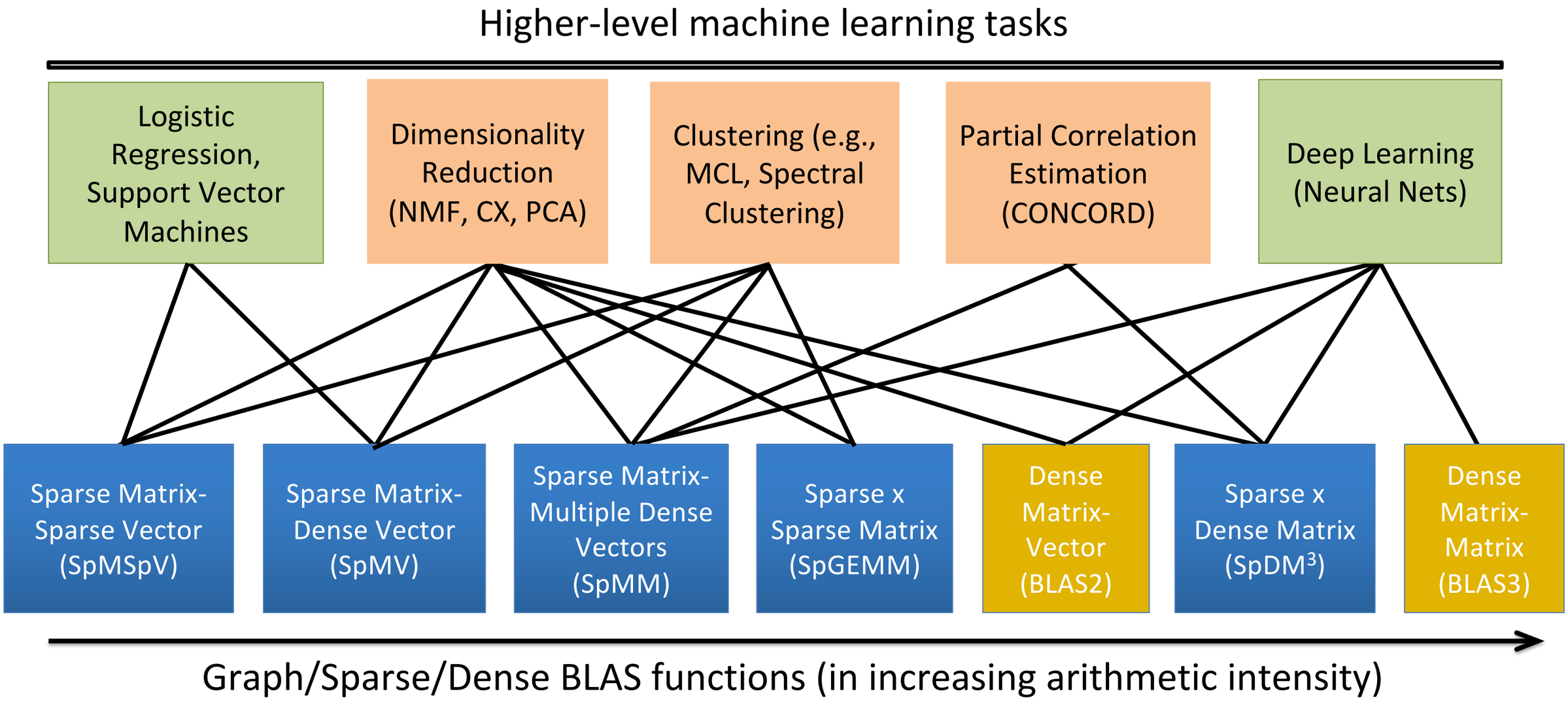
- Compression ratio (CR): flops/nnz(C)
- Combinatorial BLAS and HipMCL uses heap
- Stable performance but significant gap in high CR
- HipMCL inputs have high CR
- New hash implementation gives significant boost

Dataset	Target	Heap	Hash	Hybrid
Viruses	SpGEMM	20.38	5.14	5.13
	Total	62.59	47.87	47.77
Archaea	SpGEMM	7700.50	717.07	716.42
	Total	11535.00	4550.22	4539.07
Eukarya	SpGEMM	18448.10	1941.93	1964.34
	Total	26717.00	10241.60	10284.80

Impact of integrating hash implementation to HipMCL (single node KNL results)

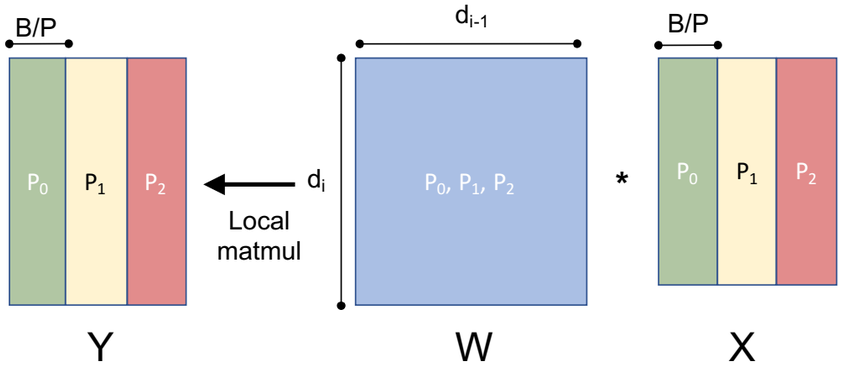
Nagasaka, Matsuoka, Azad, Buluc. High-performance sparse matrix-matrix products on intel KNL and multicore architectures. ICPPW'18

Role of Matrix Operations in Machine Learning



Neural Network Training as Matrix Operations

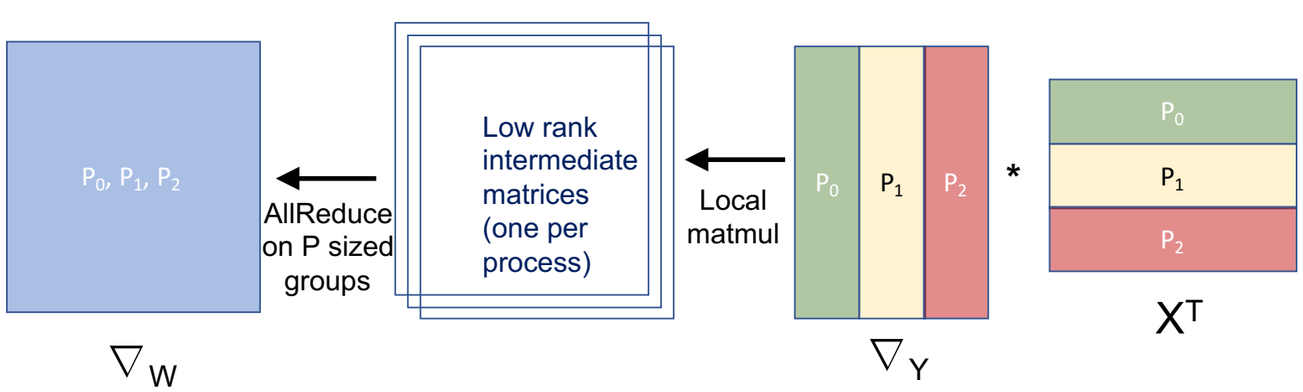
- Batch parallel training of neural networks is one-dimensional column-wise parallel matrix multiplication with full replication of the weights matrix (W)

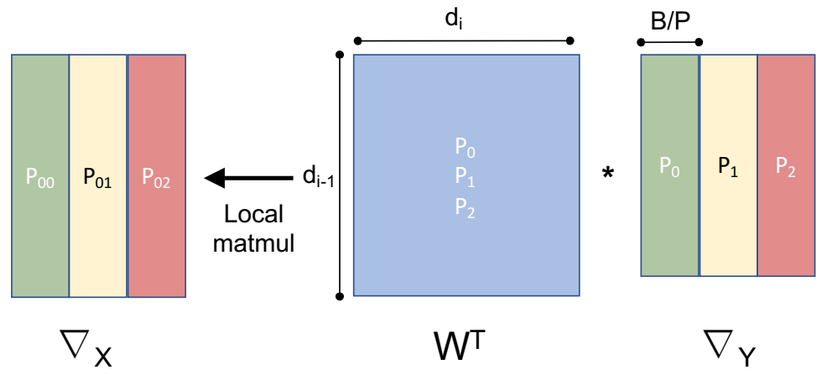


- Batch parallelism is preferred when W is small (e.g. CNNs)
- Hard limit to parallelism P<B where B is the batch size
- In practice, P< k*B where k~32 at least for performance

Top: Forward propagation, Bottom: Backpropagation

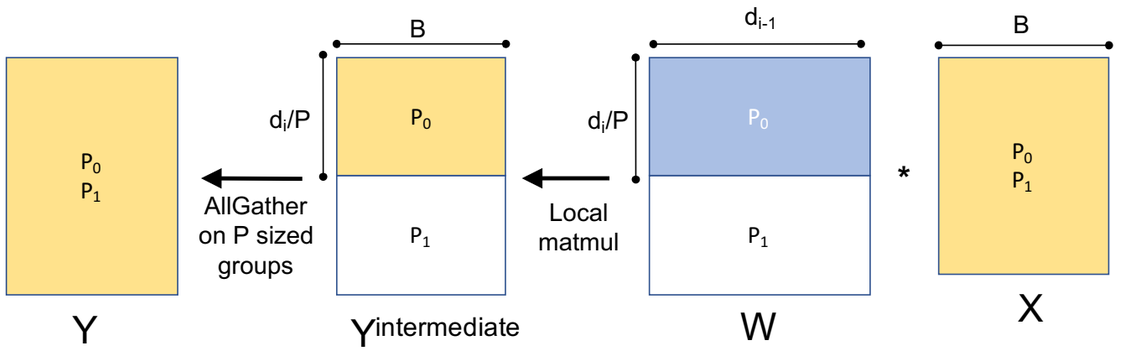
$\nabla_Y = \partial L / \partial Y$ = how did the loss function change as output activations changed?
$\nabla_X = \partial L / \partial X$, $\nabla_W = \partial L / \partial W$



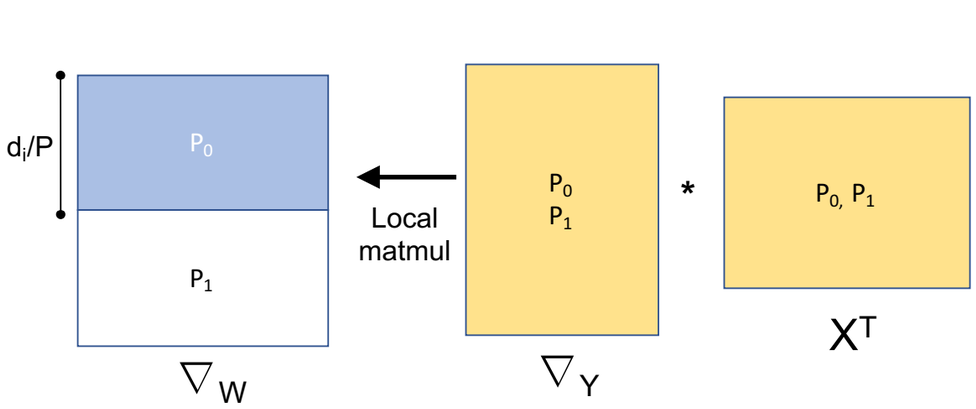


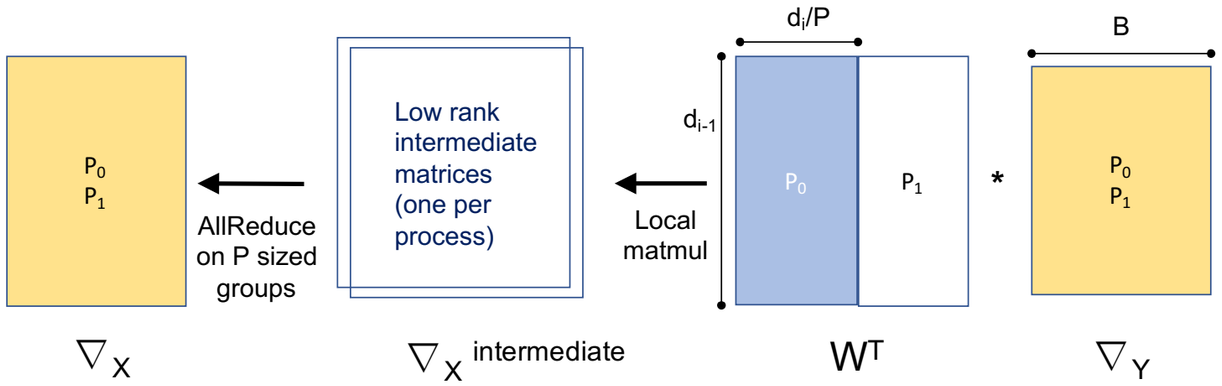
Neural Network Training as Matrix Operations

- Model parallel training of neural networks is one-dimensional row-wise parallel matrix multiplication with full replication of the data/activation matrices



- Top: Forward propagation
- Bottom: Backpropagation
- Better than batch parallel when W is large.
- Bad fit for high-latency networks (especially in its pure form), hence rarely used in industry

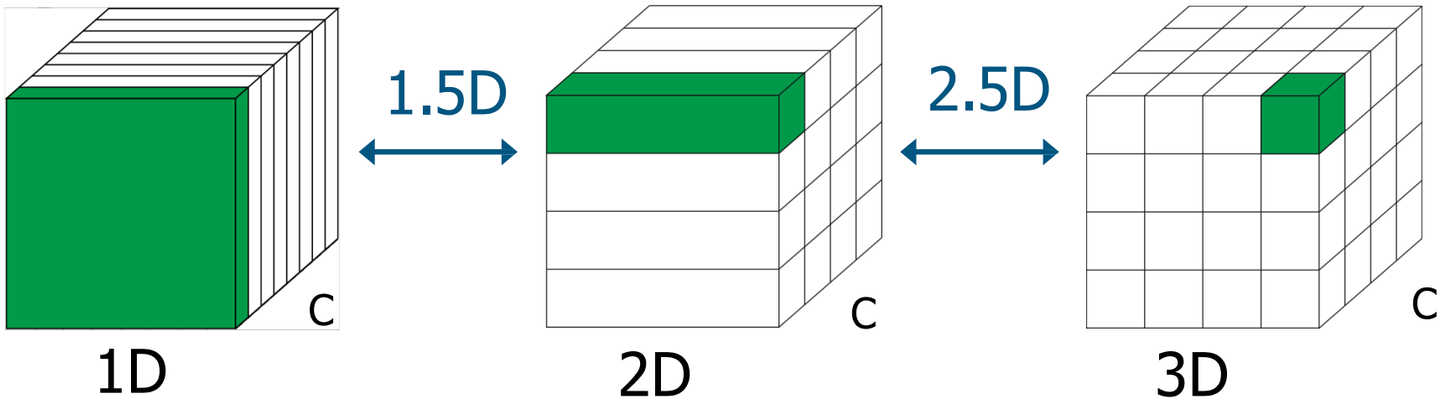




Combinations of DNN parallelism opportunities

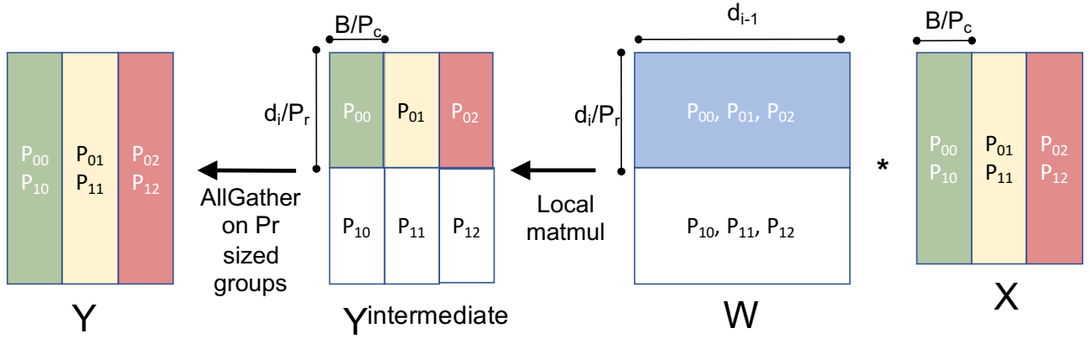
- There are various different ways to combine DNN training parallelism opportunities.
  - It helps to think in terms of matrices again.
- We will exploit communication-avoiding matrix algorithms, which trade off some storage (judicious replication) at the expense of reduced communication.
  - Deep Learning community is already OK with data or model replication in many cases

A succinct classification of parallel matrix multiplication algorithms

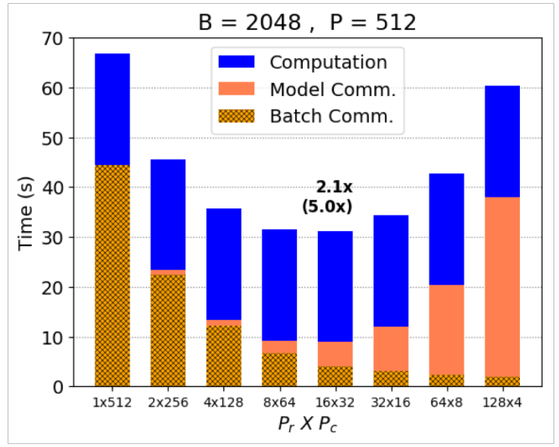


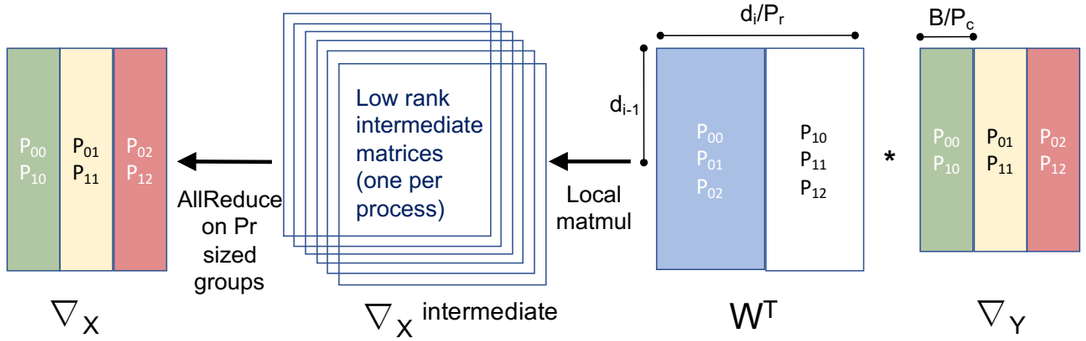
Integrated Model, Batch & Domain Parallelism in Training DNNs

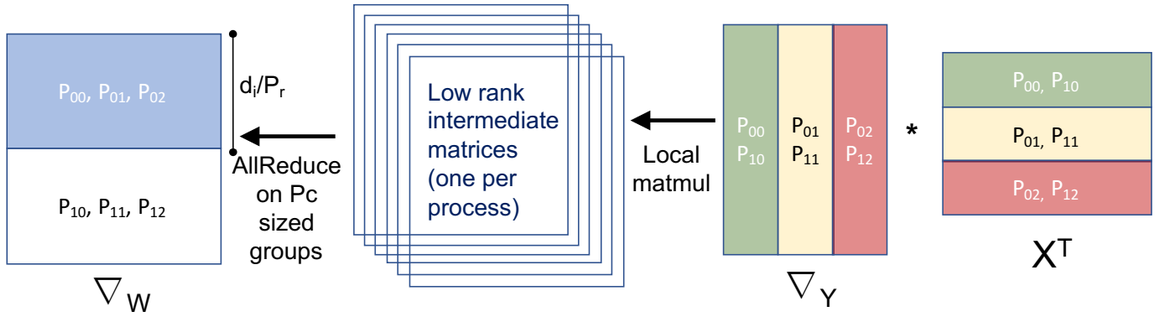
- Combining data (batch + domain) and model parallelism optimally is done using communication avoiding 1.5D algorithms due to skew in matrix sizes



Leads to optimal performance in many networks. Shown right is AlexNet







Gholami, Azad, Jin, Keutzer, Buluç. *Integrated model, batch, and domain parallelism in training neural networks* (SPAA,2018)

Conclusions

- Both graph algorithms and machine learning have growing importance in scientific applications
- Not everything is [sparse] linear algebra, but a lot of things are. Transfer of techniques and knowledge is easier when your scientific base is not domain specific
- Communication-avoiding [sparse] linear algebra algorithms provide unprecedented scaling for problems outside traditional scientific computing, such as computational biology, graph analysis, and machine learning. GraphBLAS provides a foundation for this.
- Check out http://graphblas.org, HipMCL, and Combinatorial BLAS

Acknowledgments

- **Collaborators on presented work:** Ariful Azad, Saliya Ekanayake, Amir Gholami, John Gilbert, Giulia Guidi, Peter Jin, Jeremy Kepner, Kurt Keutzer, Nikos Kyrpides, Tim Mattson, Scott McMillan, Jose Moreira, Christos Ouzounis, Dan Rokhsar, Oguz Selvitopi, Carl Yang, Kathy Yelick

- My lab website is http://passion.lbl.gov where you can find a longer version of this talk and its associated video
- We always hire good people (PhD students, postdocs, research scientists)

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