Communication-Avoiding Sparse Matrix Primitives for Parallel Machine Learning

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

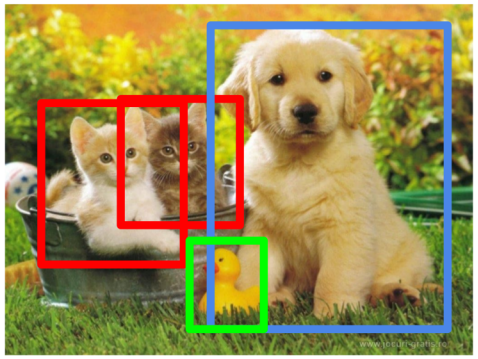
Sparse Days, Toulouse
September 28, 2018

Machine Learning for Science

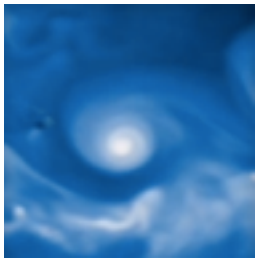
Classification

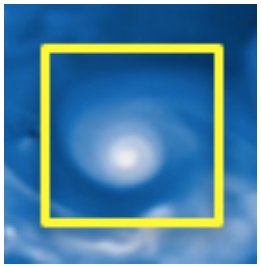
Classification + Localization

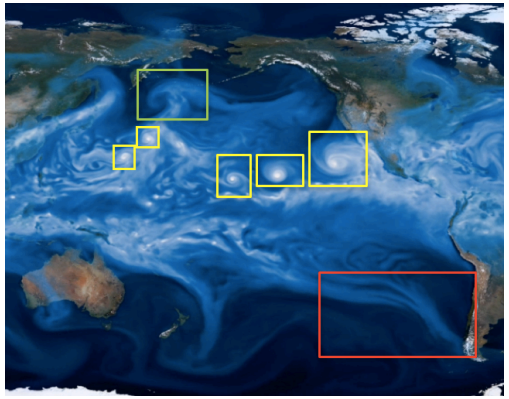
Object Detection

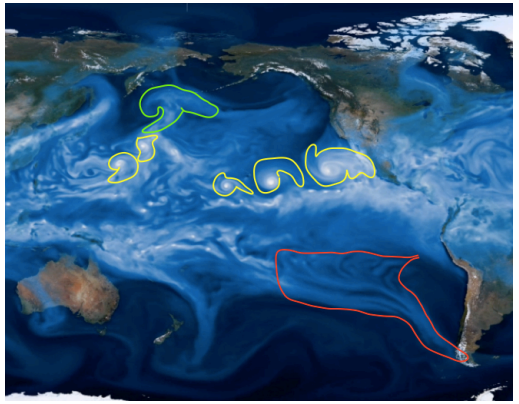


Instance Segmentation



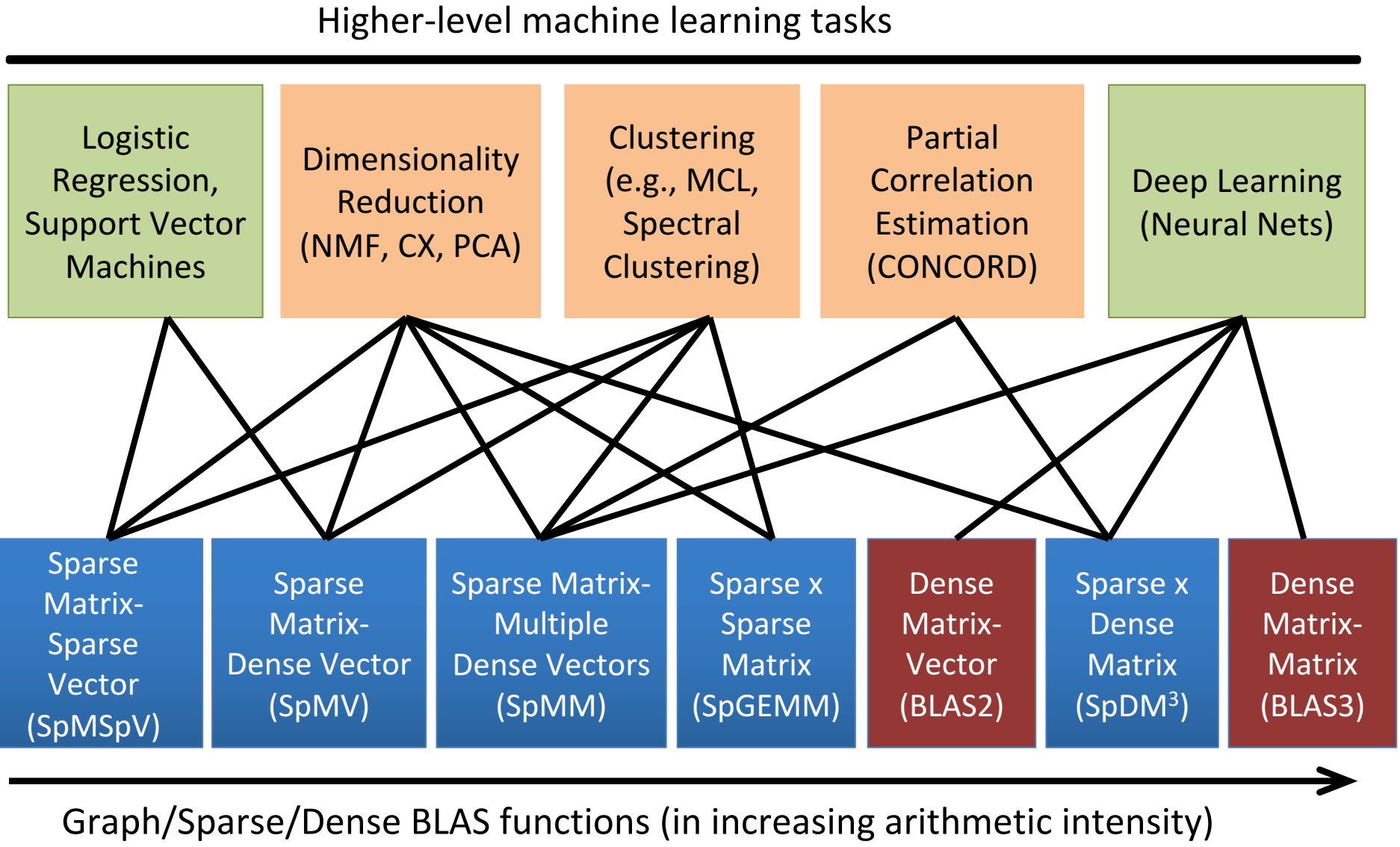






Slide source: Prabhat

Machine Learning relies a lot on Linear Algebra



Parallelism in Machine Learning

Implicit Parallelization: Keep the overall algorithm structure (the sequence of operations) intact and parallelize the individual operations.

Example: parallelizing the BLAS operations in previous figure

+ Often achieves exactly the same accuracy (e.g., model parallelism in DNN training)

- Scalability can be limited if the critical path of the algorithm is long

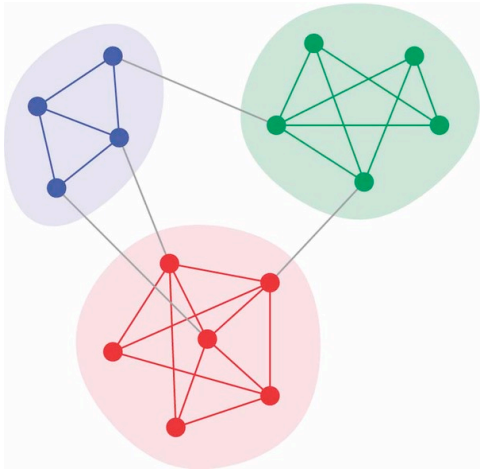
Explicit Parallelization: Modify the algorithm to extract more parallelism, such as working on individual pieces whose results can later be combined

Examples: CA-SVM and data parallelism in DNNs

+ Significantly better scalability can be achieved

- No longer the same algorithmic properties (e.g. HogWild!).

Philosophy of the Markov Cluster Algorithm (MCL)



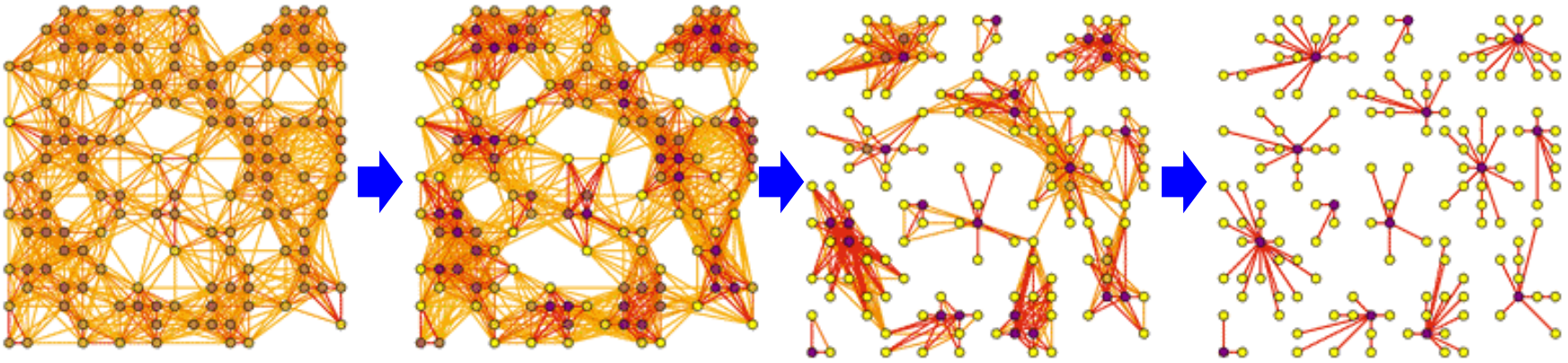
The number of **edges or higher-length paths** between two arbitrary nodes in a cluster is greater than the number of paths between nodes from different clusters

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

Markov Cluster Algorithm (MCL)

Widely popular and successful algorithm for discovering clusters in protein interaction and protein similarity networks



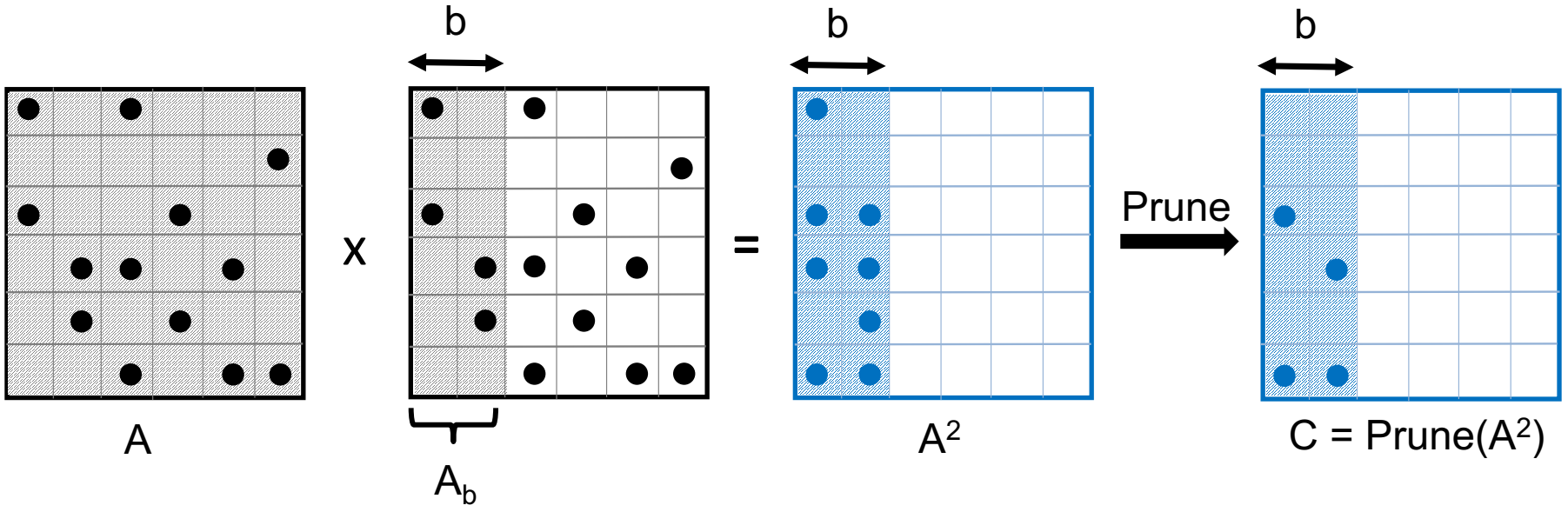
At each iteration:

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

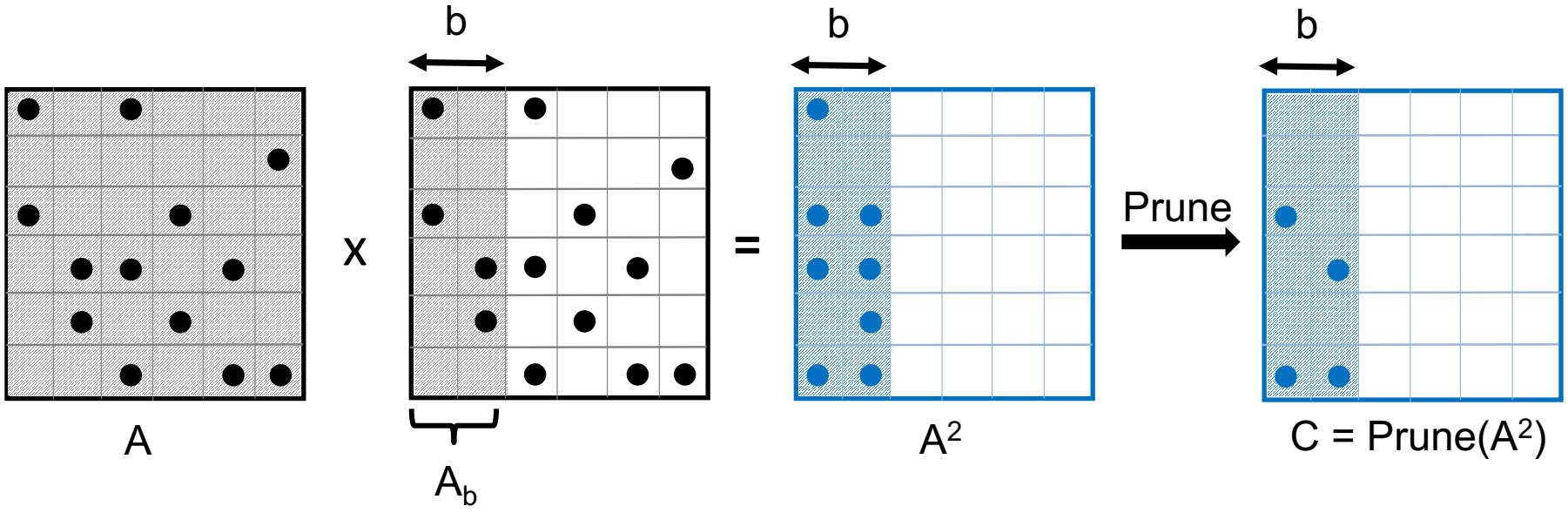
Step 2 (Inflation) : taking powers entry-wise

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

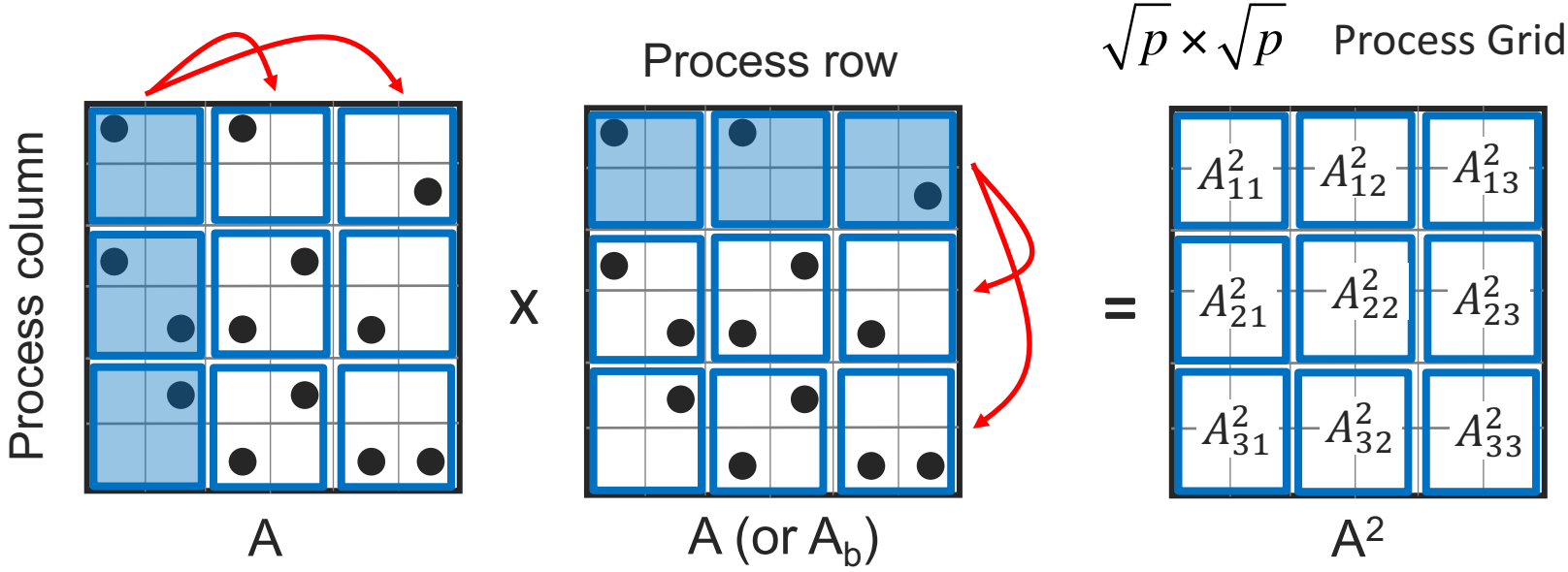
A combined expansion and pruning step



- b: number of columns in the output constructed at once
  - HipMCL selects b dynamically as permitted by the available memory
  - The algorithm works in h=N/b phases where N is the number of columns (vertices in the network) in the matrix

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*

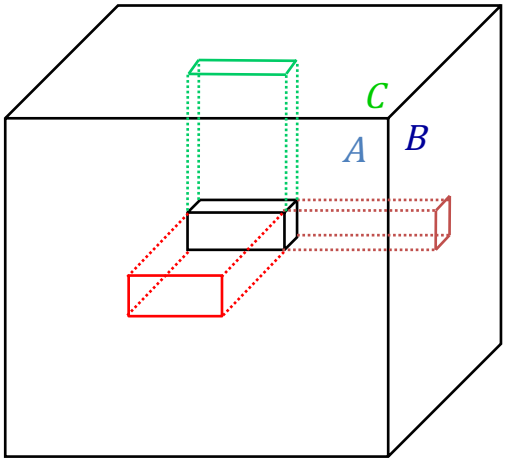
HipMCL on large networks

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

The computation cube of matrix-matrix multiplication

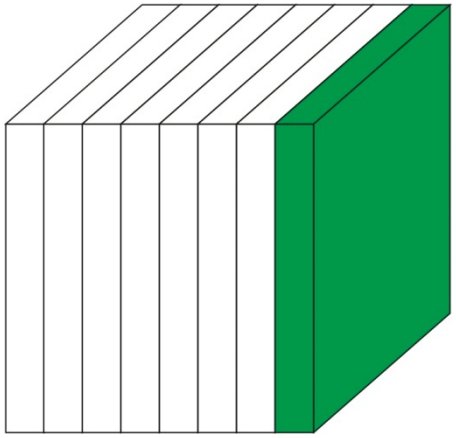
Matrix multiplication: $\forall(i,j) \in n \times n, \quad C(i,j) = \Sigma_k A(i,k)B(k,j),$



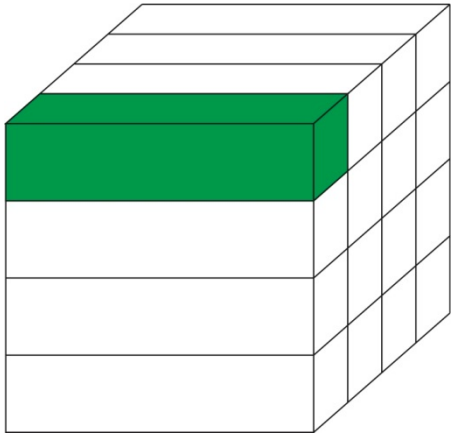
The *computation (discrete) cube*:

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

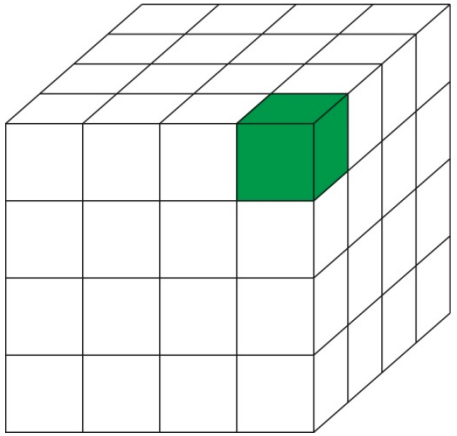
How about sparse algorithms?



1D algorithms

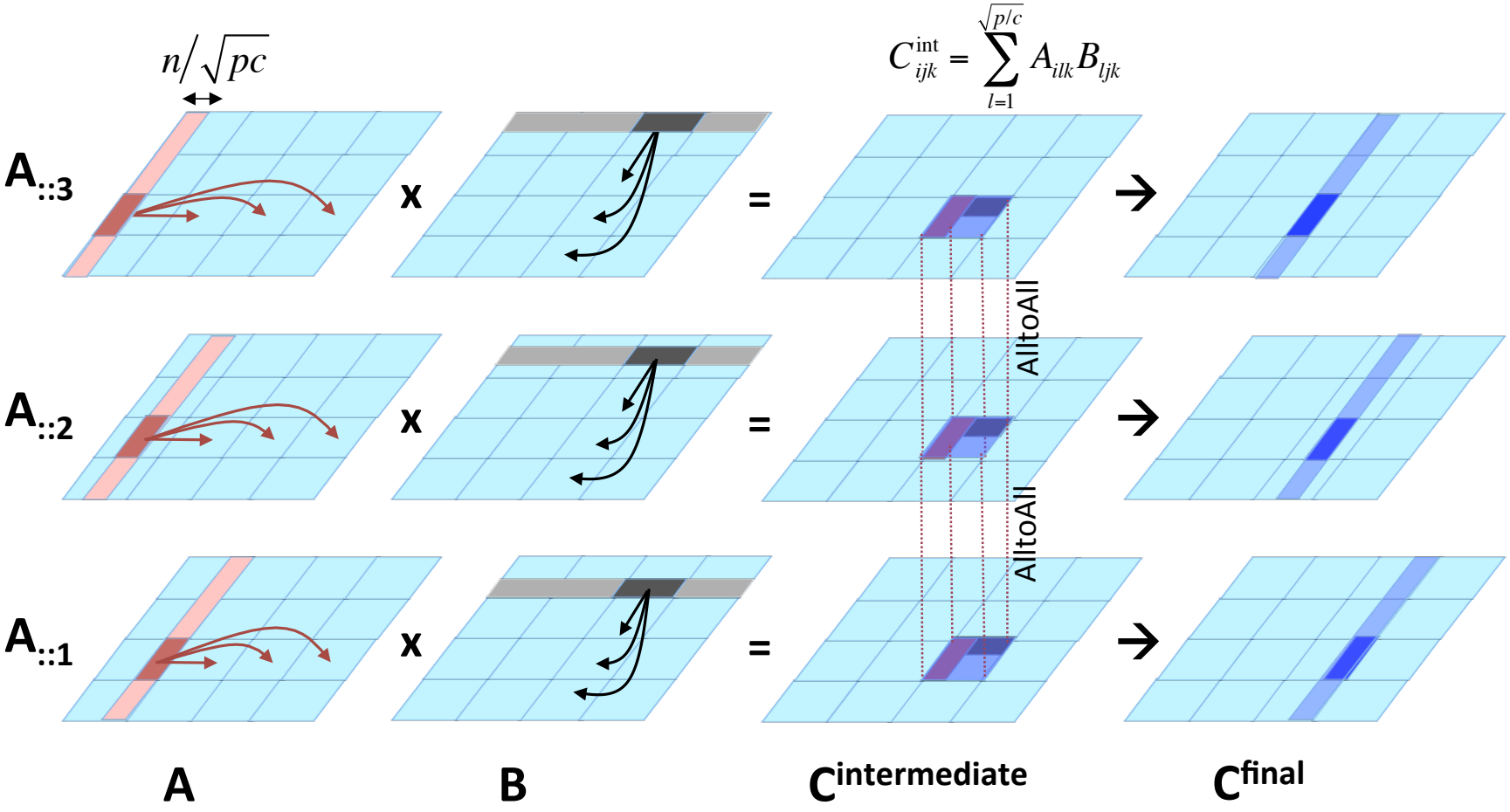


2D algorithms



3D algorithms

3D parallel SpGEMM in a nutshell

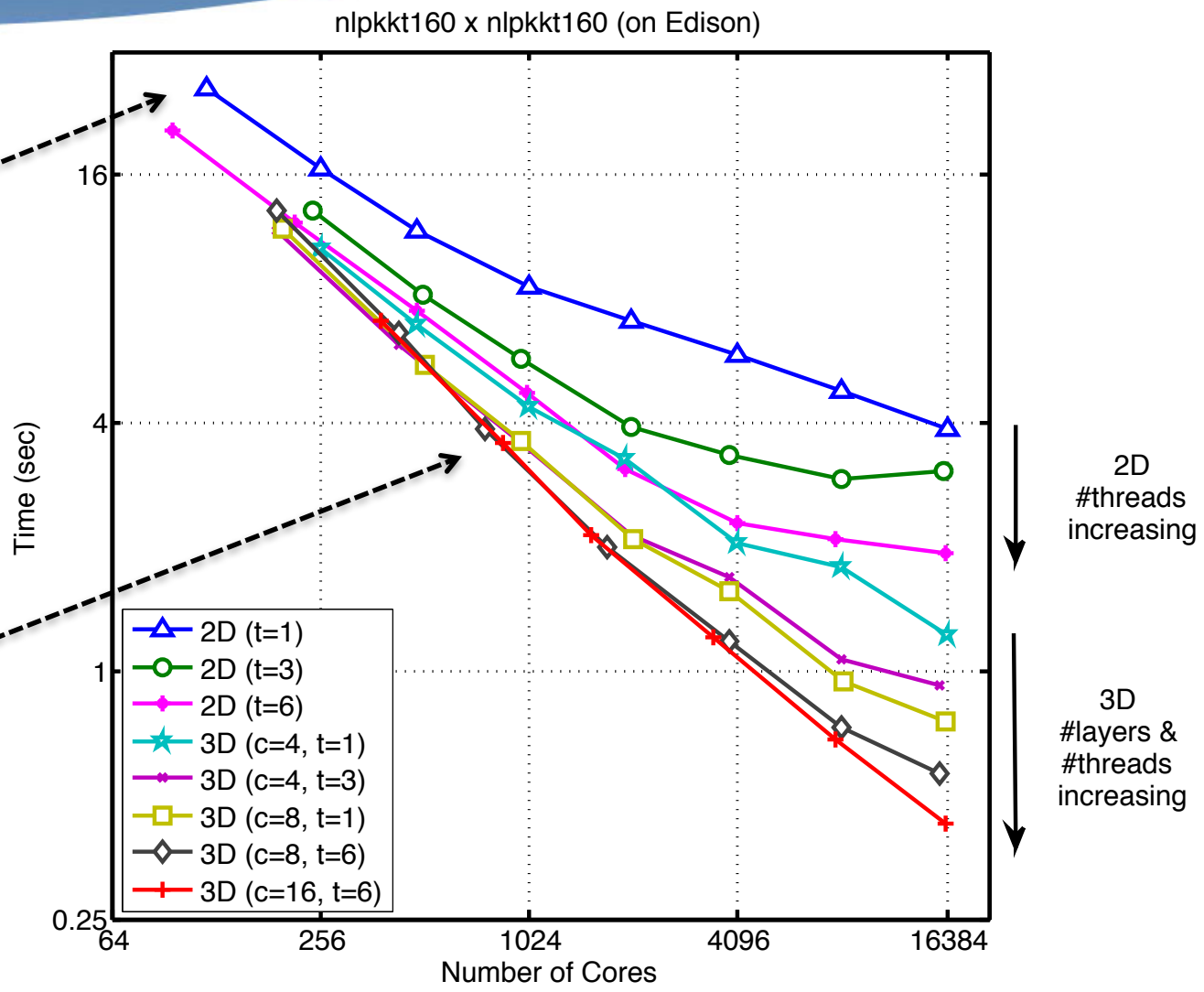


Azad, A., Ballard, G., Buluc, A., Demmel, J., Grigori, L., Schwartz, O., Toledo, S. and Williams, S., 2016. Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. *SIAM Journal on Scientific Computing*, *38*(6), pp.C624-C651.

3D SpGEMM performance

2D (non-threaded) is the previous state-of-the art

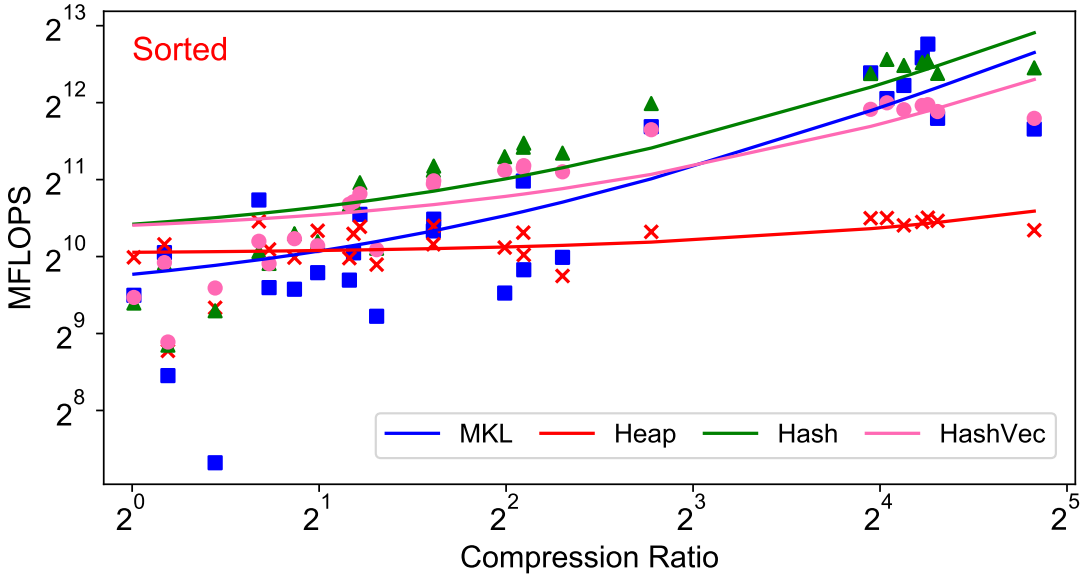
3D (threaded) – first presented here – beats it by 8X at large concurrencies

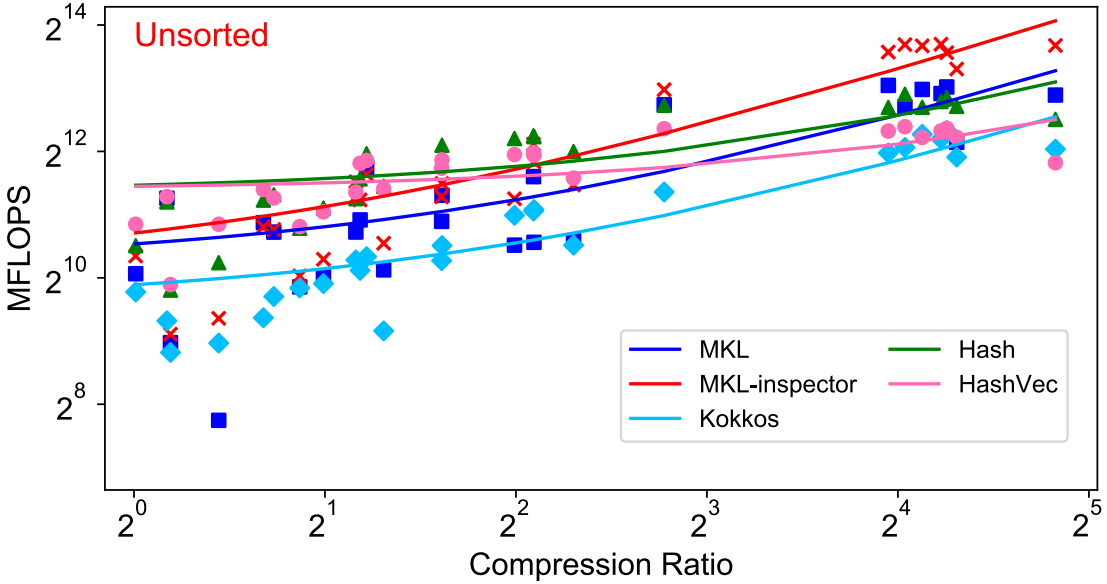


Strong scaling of different variants of 2D and 3D algorithms when squaring of nlpkkt160 matrix on Edison.

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





- We will integrate hash algorithms to CombBLAS and HipMCL

Yusuke Nagasaka, Satoshi Matsuoka, Ariful Azad, and Aydin Buluc. High-performance sparse matrix-matrix products on intel KNL and multicore architectures. In ICPPW, 2018.

Sparse Inverse Covariance Matrix Estimation

- **Precision matrix = Inverse covariance matrix**
- **Goal**: Estimating graphical model structure
- "The zeros of a precision matrix correspond to zero partial correlation, a necessary and sufficient condition for conditional independence (Lauritzen, 1996)"
- Sparsity often enforced by regularization
- One algorithm (HP-CONCORD)'s objective function:

$$\underset{\Omega \in \mathbf{R}^{p \times p}}{\text{minimize}} \; -\log\det(\Omega_D^2) + \mathbf{tr}(\Omega S \Omega) + \lambda_1 \|\Omega_X\|_1 + \frac{\lambda_2}{2}\|\Omega\|_F^2,$$

- Ω is the sparse inverse covariance matrix we are trying to estimate

Why care? Finding Direct Associations

Partial Correlation (a.k.a. sparse inverse covariance estimation): direct association without confounders

- Gene Regulatory Network (GRN) estimation
- Joint modeling of SNPs and GRN
- Linkage Disequilibrium (LD) estimation
- Canonical Correlation Analysis (CCA)
- Genome-wide association studies (GWAS)

Data-driven hypothesis generation!

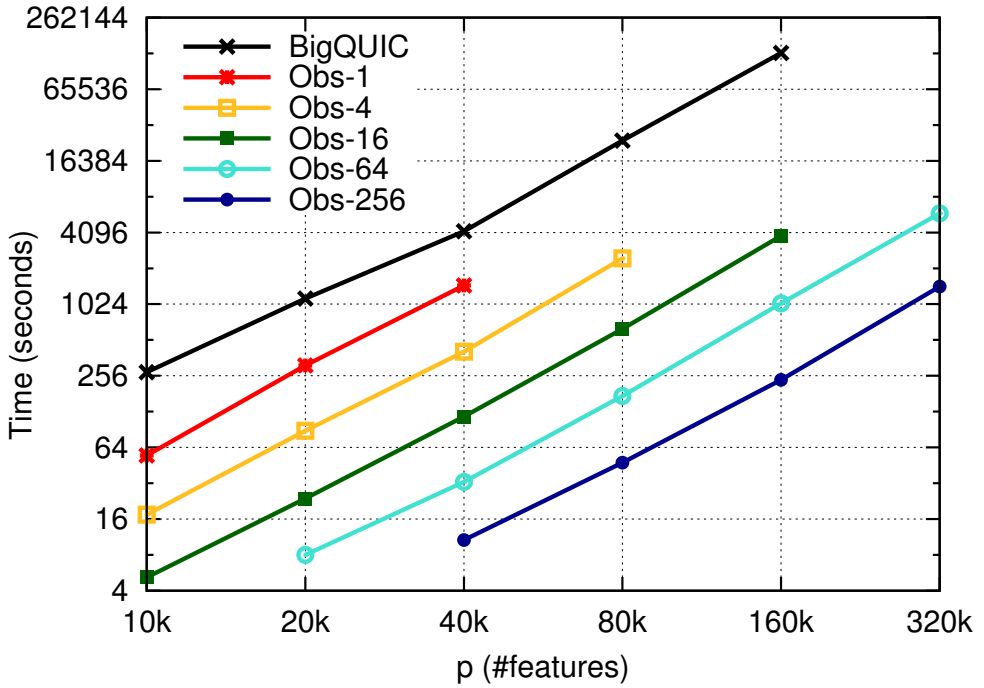
Fig. 1. Conditionally on the height of snow, the number of snowmen is independent of the intensity of traffic jams. This is represented by a two edges graph.

- Computationally challenging;
- **HP-CONCORD** on distributed memory increases scalability

HP-CONCORD Advantages

- HP-CONCORD makes fewer assumptions about the data (in particular, no Gaussianity is assumed) compared to competitors
- Thanks to **communication-avoiding matrix algorithms**, it reaches unprecedented scales

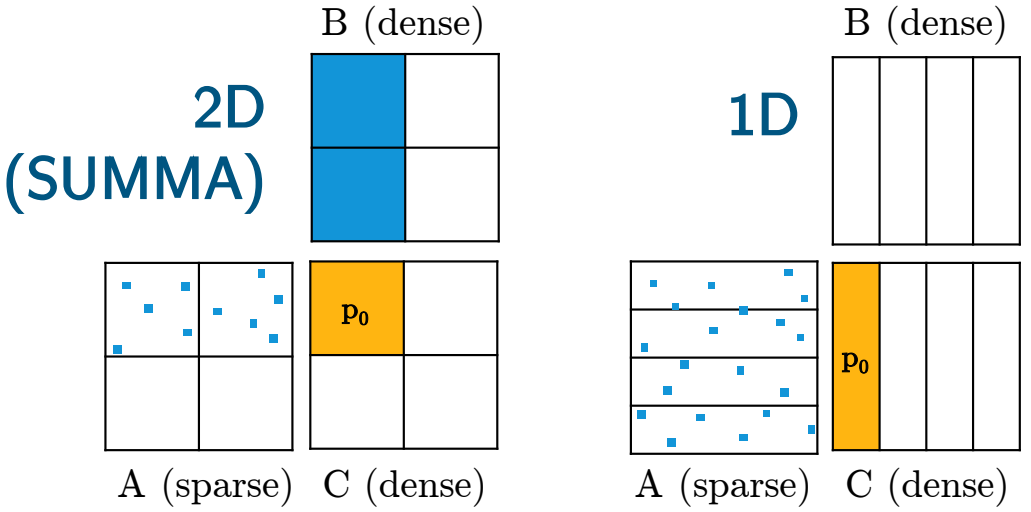
  - BigQUIC: previous state-of-the-art
  - Obs-K are our HP-CONCORD algorithm (K: number of nodes)
  - Experiment is trying to recover a random graph structure.

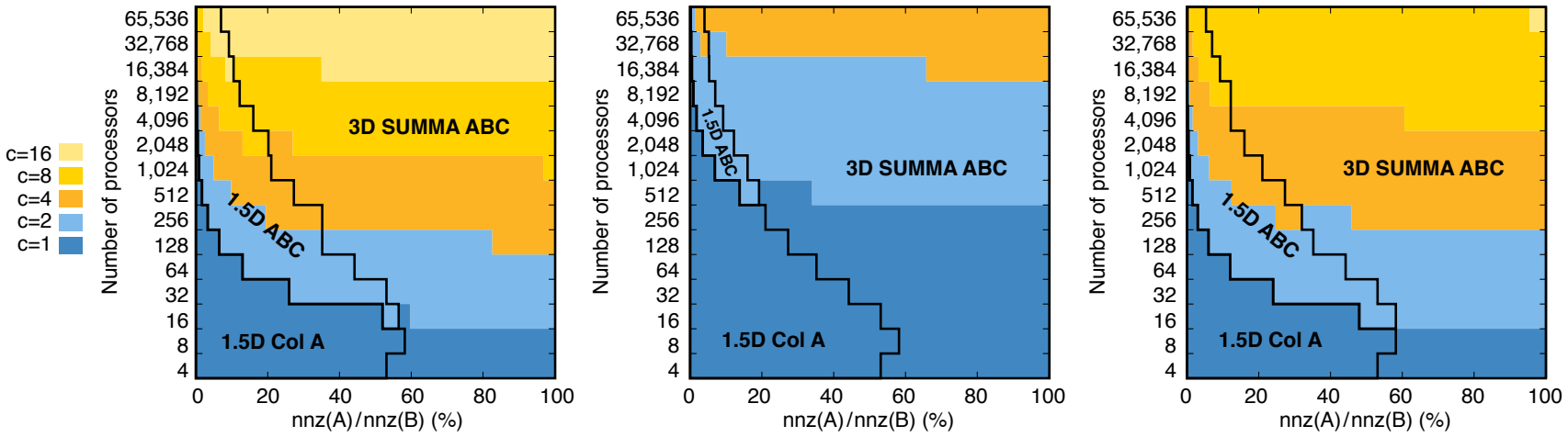


P. Koanantakool, A. Ali, A. Azad, A. Buluç, D. Morozov, S. Oh, L.Oliker, and K. Yelick "Communication-Avoiding Optimization Methods for Massive-Scale Graphical Model Structure Learning". In: International Conference on Artificial Intelligence and Statistics (AISTATS). 2018.

SpDM^3: Sparse x Dense Matrix: The workhorse of HP-CONCORD

- The best algorithm when multiplying two matrices with unequal nonzero counts?
- Depends on the concurrency!

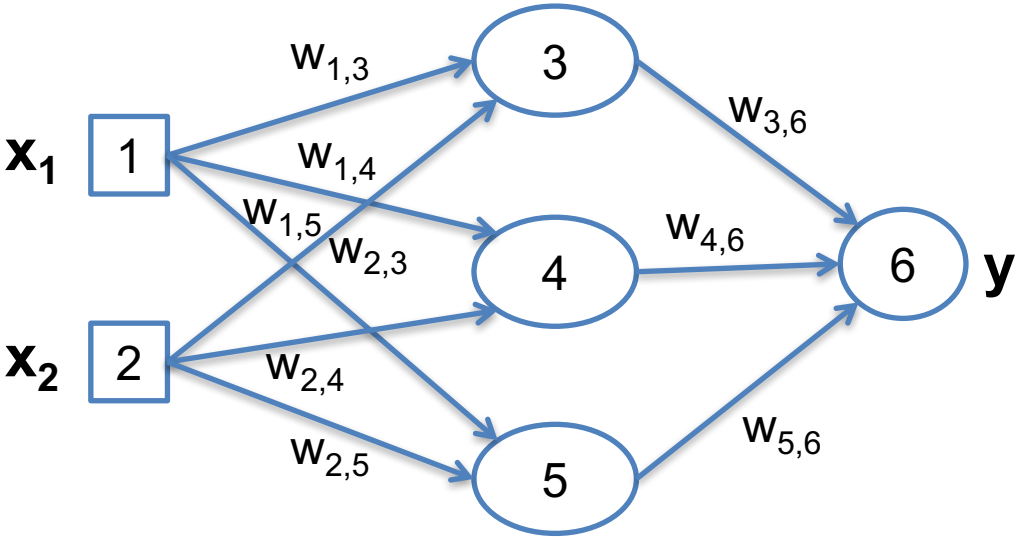




Koanantakool, Penporn, et al. "Communication-avoiding parallel sparse-dense matrix-matrix multiplication." IPDPS, 2016

Training Neural Networks

- Training is to **adjust the weights (W)** in the connections of the neural network, in order to change the function it represents.



Only parameters are weights for simplicity (i.e. ignore bias parameters)

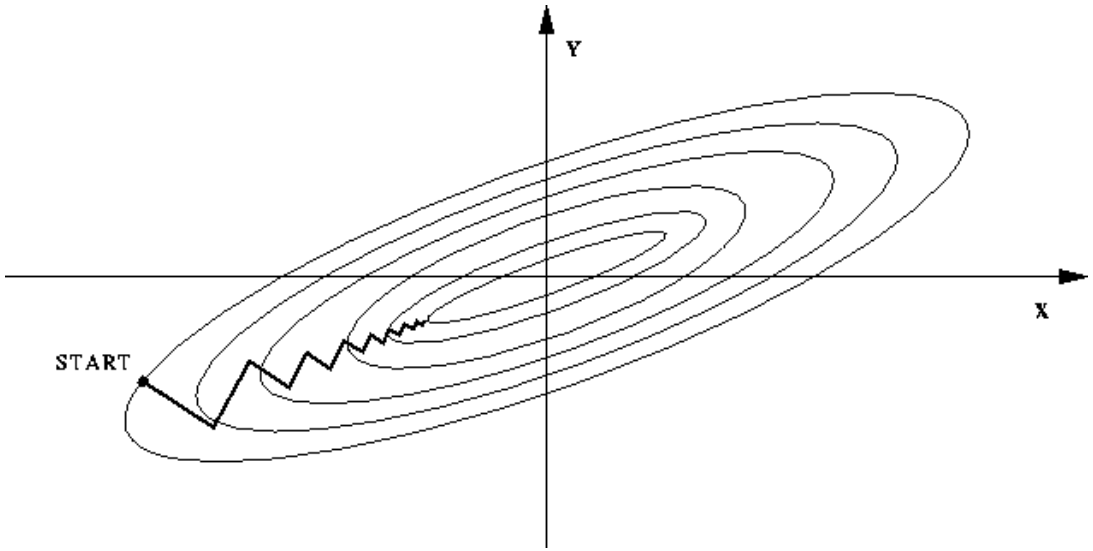
W: the matrix of weights

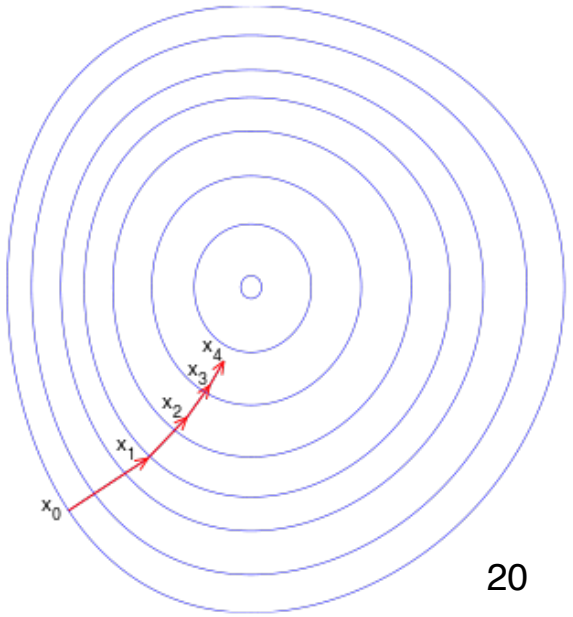
A "shallow" neural network with only one hidden layer (nodes 3,4,5), two inputs and one output.

Gradient Descent

$$W^{t+1} \leftarrow W^{t} - \alpha \cdot \nabla_{W} f(W^{t}, x)$$

- Also called the steepest descent algorithm
- In order to minimize a function, move towards the opposite direction of the gradient at a rate of α.
- α is the step size (also called the learning rate)
- Used as the ***optimization backend*** of many other machine learning methods (example: NMF)





Stochastic Gradient Descent (SGD)

$$\text{Assume } f(W^t, x) = \frac{1}{n}\sum_{i=1}^{n} f_i(W^t, x)$$

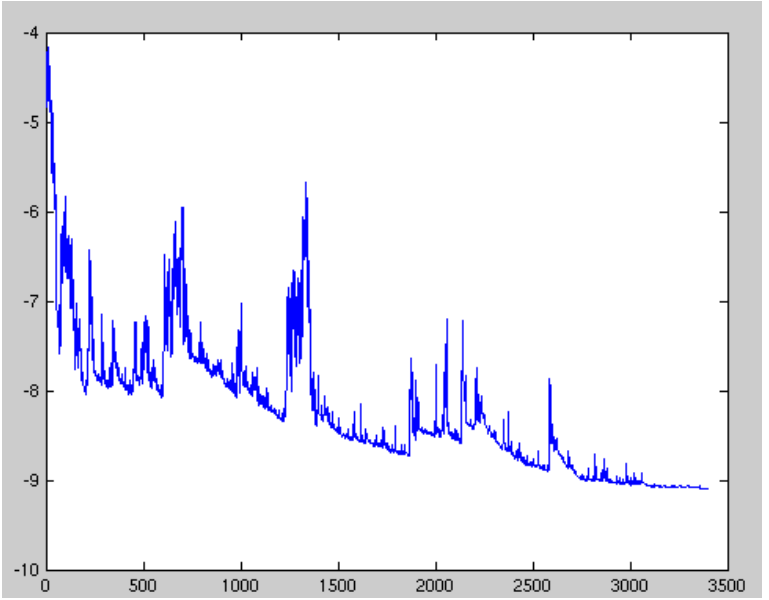
$$W^{t+1} \leftarrow W^t - \alpha \cdot \nabla_W f_i(W^t, x)$$

Pure SGD: compute gradient using 1 sample

$$W^{t+1} \leftarrow W^t - \alpha \cdot \frac{1}{b}\sum_{i=k+1}^{k+b} \nabla_W f_i(W^t, x)$$

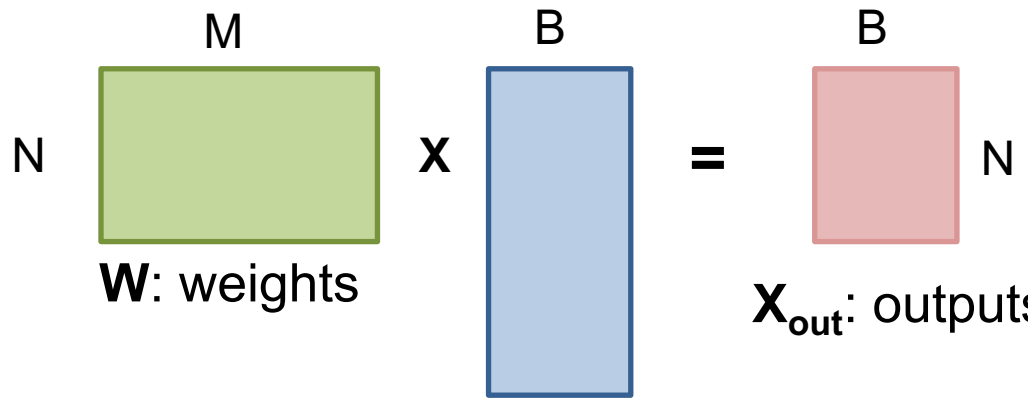
Mini-batch: compute gradient using b samples

f is not going down for every iteration



- Actually the name is a misnomer, *this is not a "descent" method*
- But we will stick to it anyway to avoid confusion.
- Performance and parallelism requires batch training
- Larger batch sizes hurt convergence as they get trapped easily
- SGD escapes sharp local minima due to its "noisy" gradients

SGD training of NNs as matrix operations

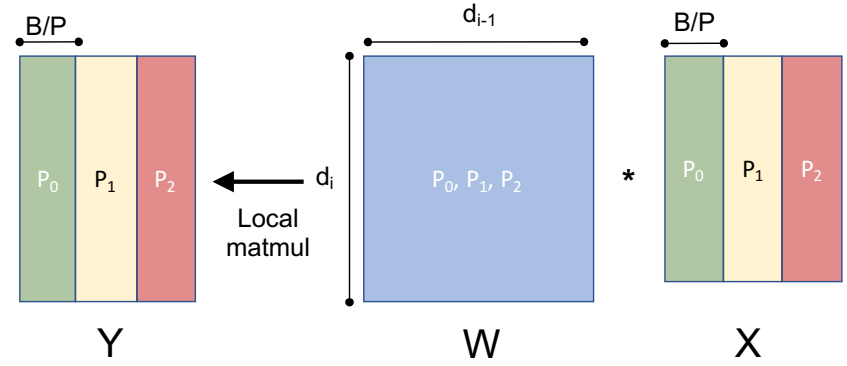


N = the number of outputs
M = the number of inputs
B = the size of the minibatch

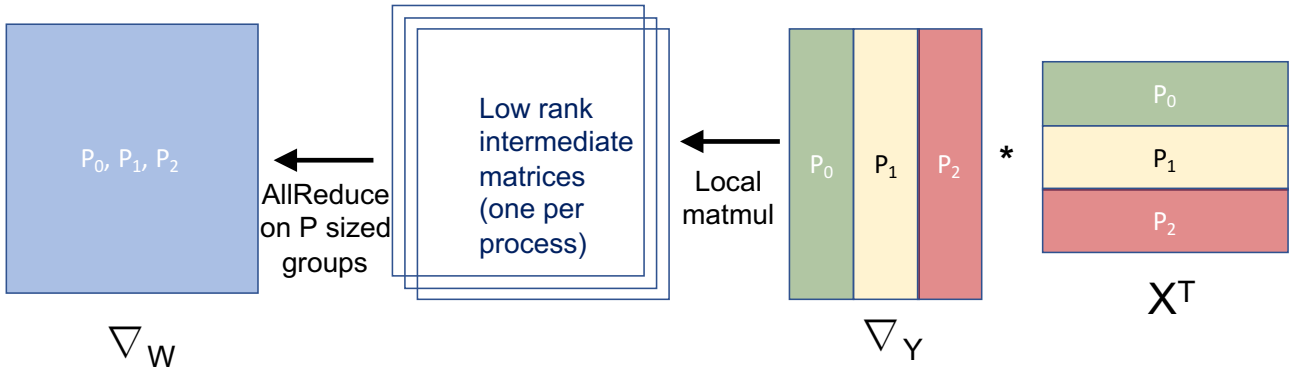
The impact to parallelism:

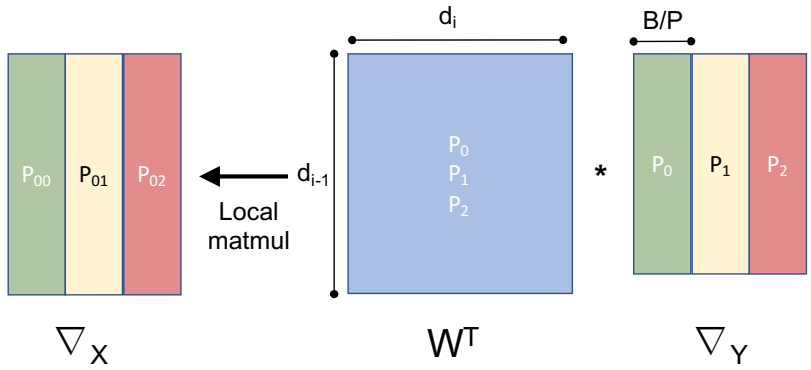
- **W** is replicated to processor, so it doesn't change
- $\mathbf{X_{in}}$ and $\mathbf{X_{out}}$ gets skinnier if we only use data parallelism, i.e. distributing **b=B/p** mini-batches per processor
- GEMM performance suffers as *matrix dimensions get smaller* and *more skewed*
- **Result:** Data parallelism can hurt single-node performance

Data Parallel SGD training of NNs as matrix operations



1. Which matrices are replicated?
2. Where is the communication?
3. Which steps can be overlapped?



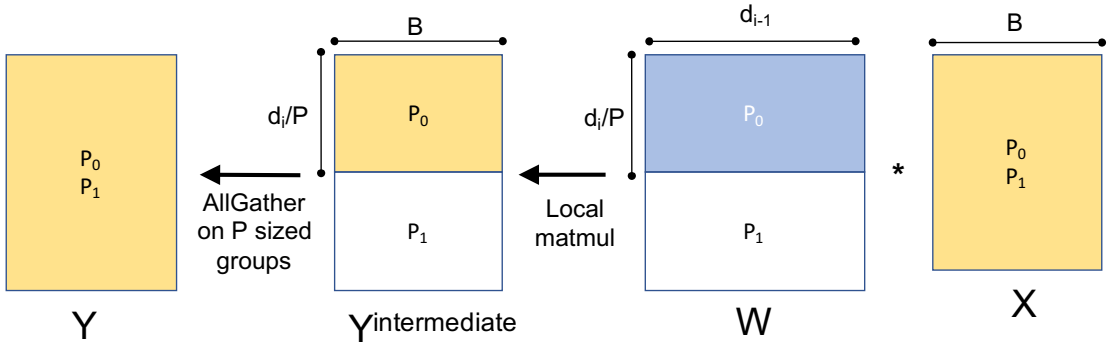


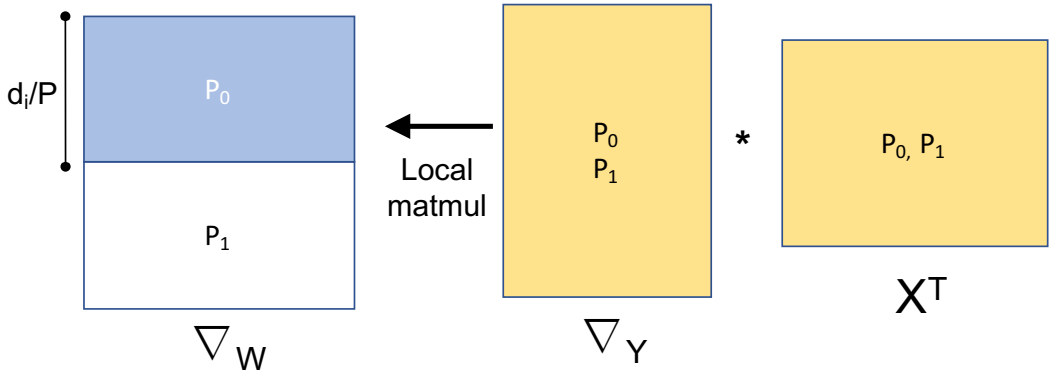
$\nabla_Y = \partial L / \partial Y$ = how did the loss function change as output activations changed?

$\nabla_X = \partial L / \partial X$

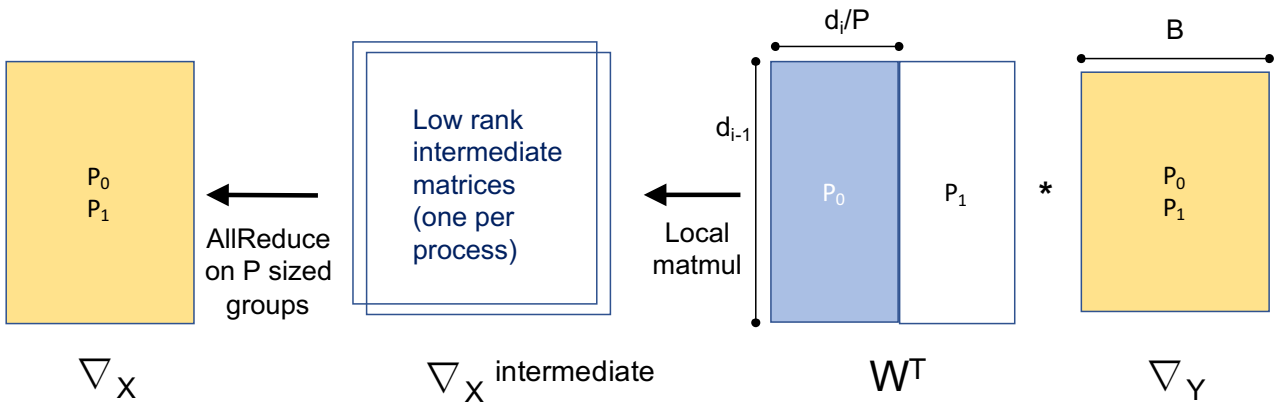
$\nabla_W = \partial L / \partial W$

Model Parallel SGD training of NNs as matrix operations





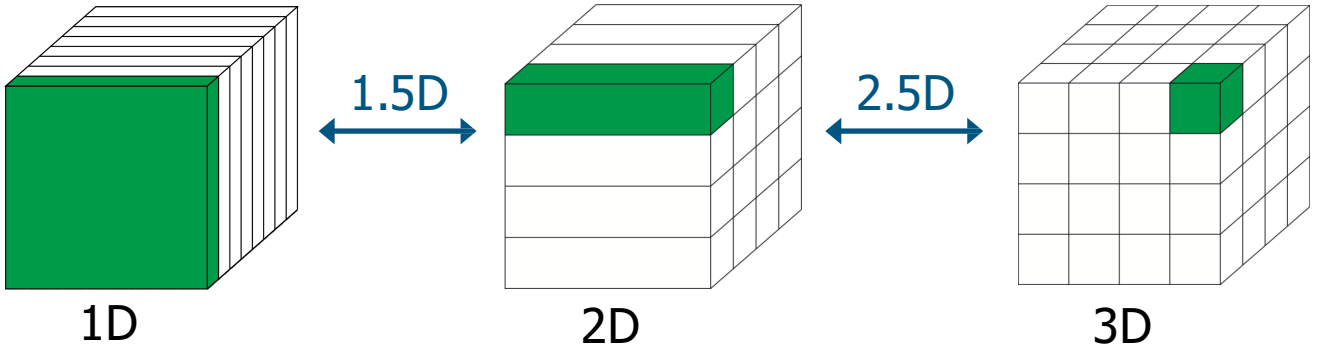
1. Which matrices are replicated?
2. Where is the communication?
3. How can matrix algebra capture both model and data parallelism?



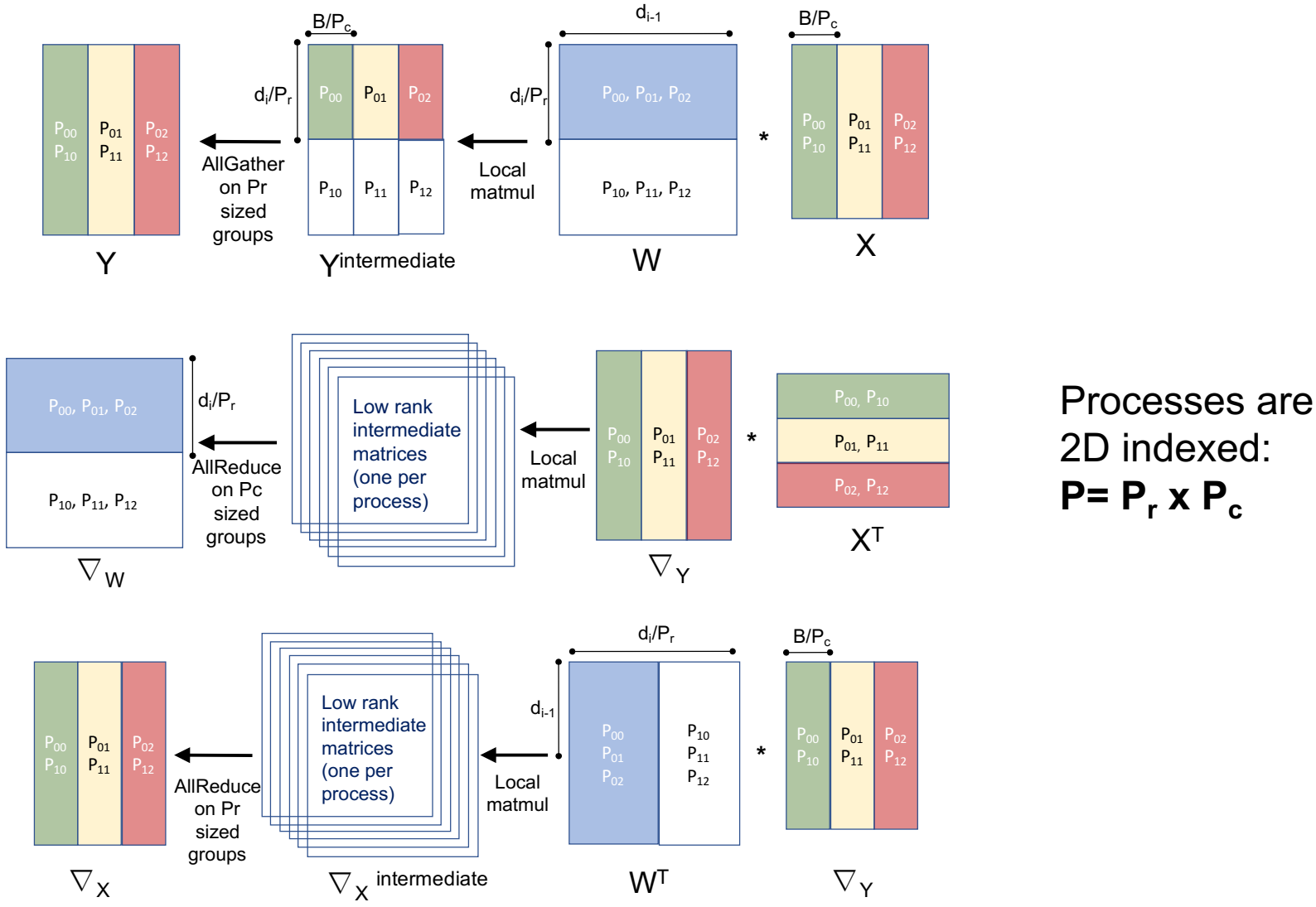
Combinations of various parallelism opportunities

- 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



Data & Model Parallel SGD training of NNs as matrix operations

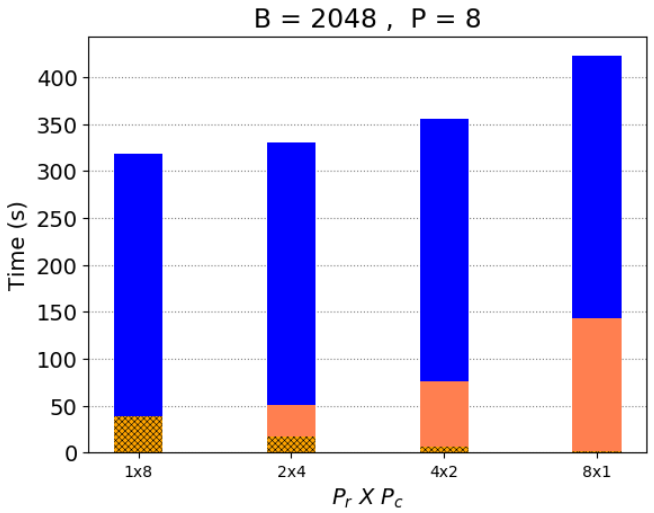


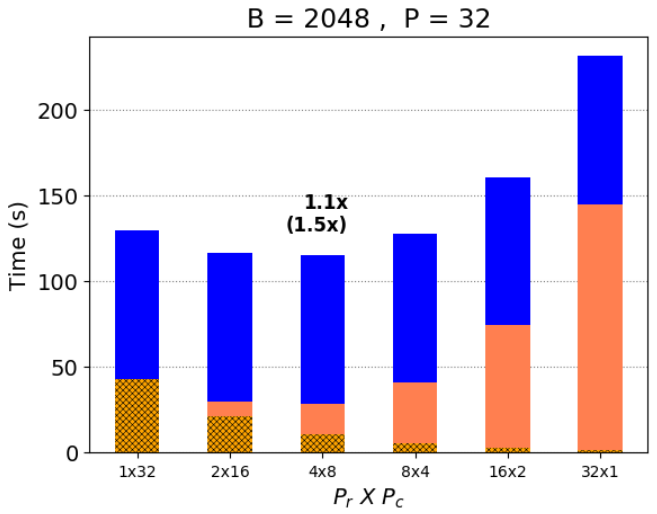
Processes are 2D indexed:
$P = P_r \times P_c$

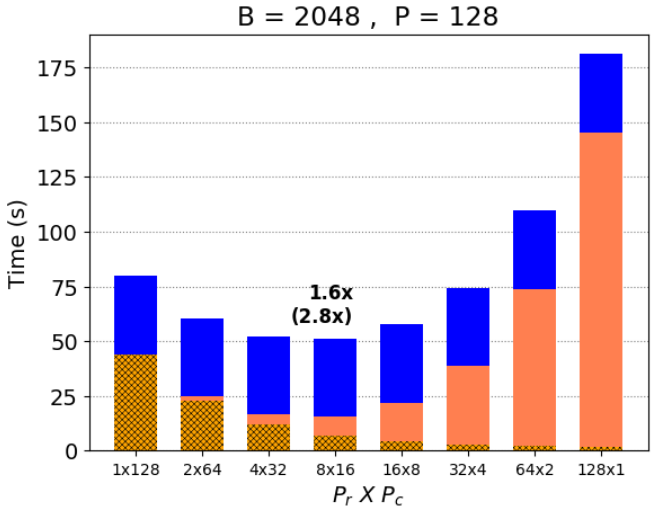
A. Gholami, A. Azad, P. Jin, K. Keutzer, A. Buluç. Integrated Model, Batch, and Domain Parallelism in Training Neural Networks. ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'18)

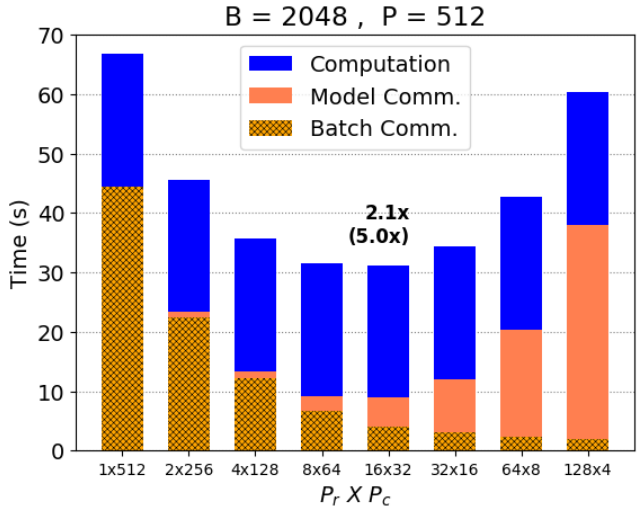
Integrated Batch + Model Scaling

- **For large processes integrated could provide up to 2x speedup**









Conclusions

- 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.
- Check out http://graphblas.org, HipMCL, and HP-CONCORD
- **We are hiring! (talk to me afterwards)**

Acknowledgments

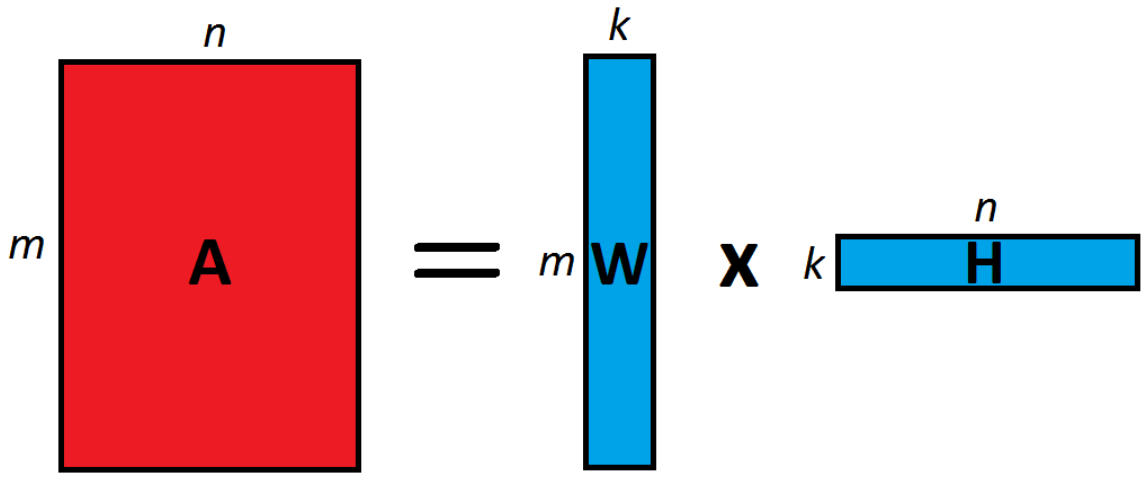
Alnur Ali, Ariful Azad, David Bader, Grey Ballard, James Demmel, Amir Gholami, John Gilbert, Laura Grigori, Peter Jin, Jeremy Kepner, Kurt Keutzer, Penporn Koanantakool, Nikos Kyrpides, Tim Mattson, Scott McMillan, Dmitriy Morozov, Jose Moreira, Sang-Yun Oh, Lenny Oliker, John Owens, Christos Ouzounis, Georgios Pavlopoulos, Prabhat, Oded Schwartz, Sivan Toledo, Sam Williams, Carl Yang, Kathy Yelick.

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

Extra Slides

Non-negative matrix factorization (NMF)

$$\min_{W \geq 0, H \geq 0} f(W,H) = \frac{1}{2}\|A - WH\|_F^2$$



- Dimensionality reduction with non-negativity constraints
- The name “factorization” is a misnomer; NMF is just a low-rank approximation as exact factorization is NP-hard
- NMF is a family of methods, not just one algorithm

The Alternating Updates Framework

Initialize H

Repeat until convergence:

1. For fixed H, solve $\min_{W \geq 0,} \left\| H^T W^T - A^T \right\|_F^2$
2. For fixed W, solve $\min_{H \geq 0,} \left\| WH - A \right\|_F^2$

Lots of algorithms fall into this framework.

- Multiplicative update (MU)
- Alternating least squares (ALS)
- Alternating non-negative least squares (ANLS)

J. Kim and H. Park. "Fast nonnegative matrix factorization: An active-set-like method and comparisons." SIAM Journal on Scientific Computing, 2011

Caveat emptor: This is not the only method for finding an NMF

Gemulla, Rainer, et al. "Large-scale matrix factorization with distributed stochastic gradient descent." KDD, 2011

The Alternating Updates Framework

Main computation is large-scale matrix multiplications:

1. $\mathbf{H}\mathbf{H}^T$ and $\mathbf{A}\mathbf{H}^T$ for updating **W**, given a fixed **H**
2. $\mathbf{W}^T\mathbf{W}$ and $\mathbf{W}^T\mathbf{A}$ for updating **H**, given a fixed **W**

- In general **W** and **H** are dense, but short-fat or tall-skinny
- **A** is often sparse but can be dense depending on application
- For increased interpretability, **H** or **W** can also be sparse

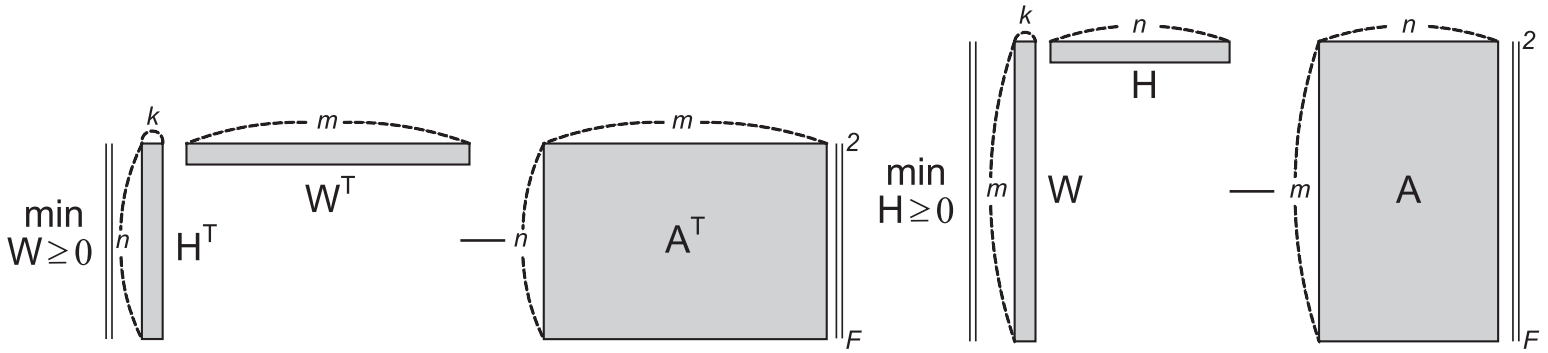


Figure: Kim and Park

The Alternating Updates Framework

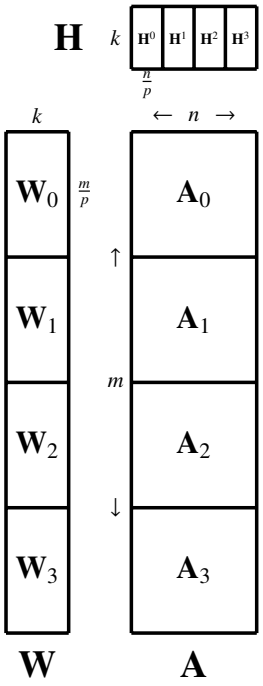
Main computation is large-scale matrix multiplications

Choose the best distribution and algorithm depending on:

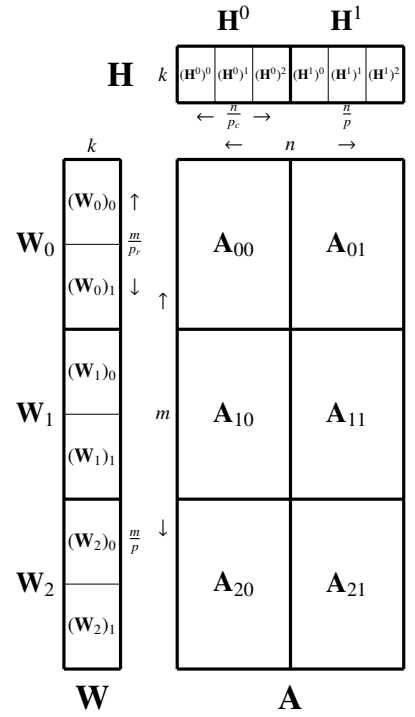
1- the relative sizes of the dimensions of the matrices

2- the number of processors

This work: Never communicate A, because it is asymptotically larger than H and W



(a) 1D Distribution with $p = p_r = 4$ and $p_c = 1$.

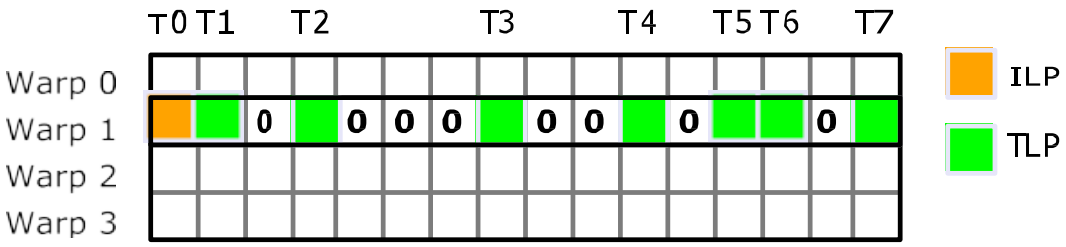


(b) 2D Distribution with $p_r = 3$ and $p_c = 2$.

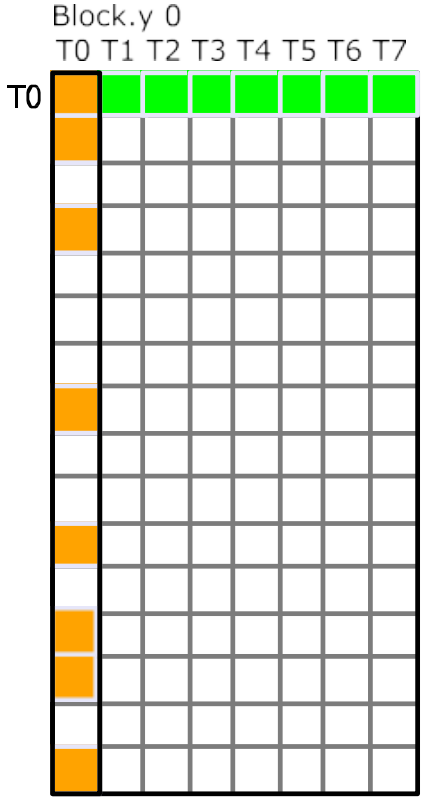
Kannan, Ballard, Park. “MPI-FAUN: An MPI-Based Framework for Alternating-Updating Nonnegative Matrix Factorization”. 2016.

Sparse Matrix – Multi Vector Multiplication (SpMM) on the GPU

- Thinking about TLP and ILP is the right way to think about sparse matrix multiplication
- Isolate *memory reads* from *compute* from *memory writes*, because this allows ILP to materialize



(a) Sparse matrix A

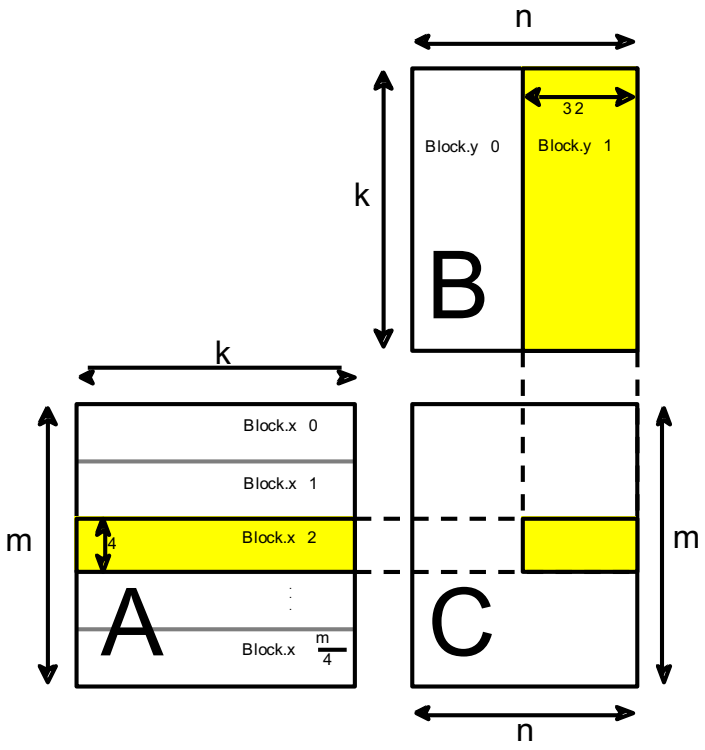


(b) Dense matrix B

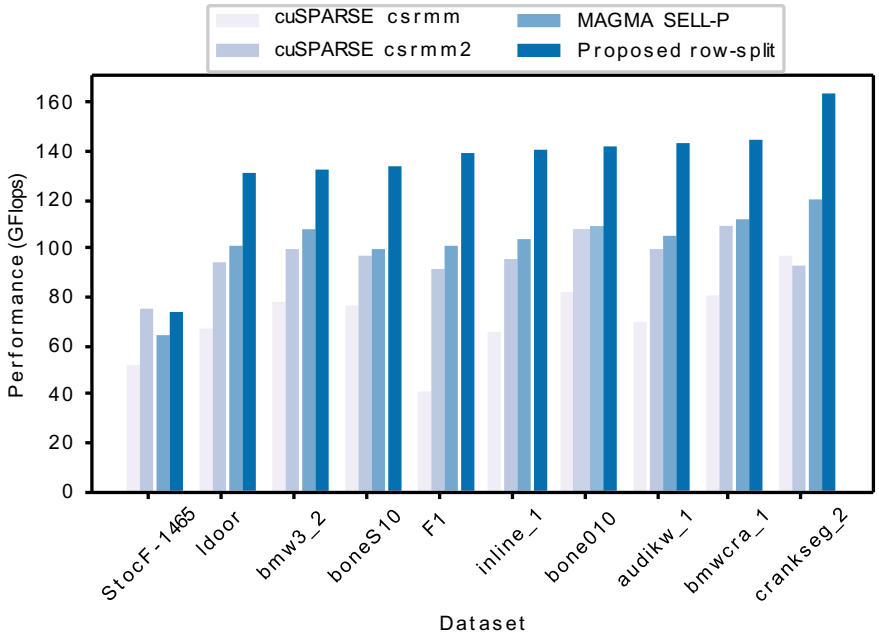
1. T0 broadcasts 0 to T1, T2, ..., T7
2. T0, T1, T2, ..., T7 do coalesced memory access for row 0

Carl Yang, Aydin Buluç, and John D Owens. Design principles for sparse matrix multiplication on the GPU. In Euro-Par, 2018. Distinguished Paper and Best Artifact Awards

Sparse Matrix – Multi Vector Multiplication (SpMM) on the GPU



A new data access pattern



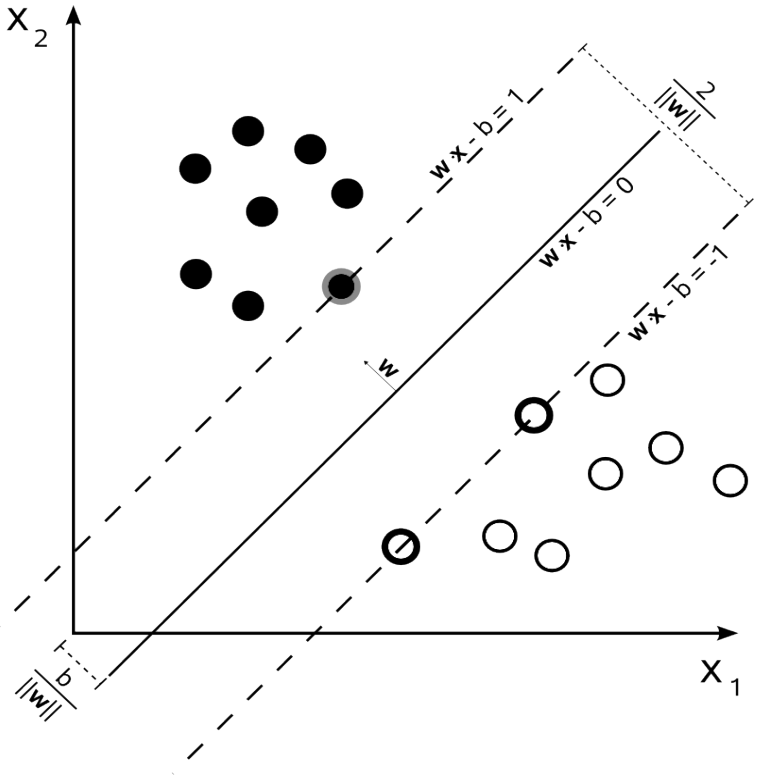
The paper also proposes a merge-based algorithm for sparser inputs

Carl Yang, Aydin Buluç, and John D Owens. Design principles for sparse matrix multiplication on the GPU. In Euro-Par, 2018. Distinguished Paper and Best Artifact Awards

Support Vector Machine

$$\max_{w,b} \quad \frac{1}{\|w\|}$$

$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$



- Only the classification constraints on the support vectors are active
- Naively, leads to a giant quadratic constrained optimization (QP) problem
- Special algorithms, such as Sequential Minimal Optimization (SMO), decompose this giant QP to smaller (in fact ***minimal***) QP sub-problems.

Major Bottleneck of Kernel SVM

- Input dataset: n-by-d matrix (n >> d)
  - X1, X2, …, Xn. Xi is a vector has d features
- Generate a n-by-n Kernel matrix at runtime
  - K[i][j] = exp(-r||Xi - Xj||^2), r is positive number
- O(n^3) operations and O(n^2) memory are huge!
  - a small input generates a large Kernel matrix
  - 357MB input (52K-by-90) = 2000GB Kernel matrix
- Solution: SMO (sequential minimal optimization)
  - using iterative method, avoiding Kernel matrix
  - using ***two rows of Kernel matrix*** each iteration
  - key kernel for sparse inputs: ***sparse matrix times sparse vector*** (as in the BFS case from graph lecture)

Sequential Minimal Optimization

$$\max_{\alpha} \; W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \boxed{\kappa(\mathbf{x}_i^T \mathbf{x}_j)}$$

The kernel function

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\boxed{\sum_{i=1}^{m} \alpha_i y_i = 0.}$$

The equality constraint

- The smallest possible optimization problem involves "two" Lagrange multipliers at a time
- Because just changing on multiplier would violate the equality constraint

Platt, John. "Fast training of support vector machines using sequential minimal optimization." Advances in kernel methods. 1999.

Sequential Minimal Optimization

$$\max_{\alpha} \; W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \boxed{\kappa(\mathbf{x}_i^T \mathbf{x}_j)}$$

The kernel function

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\boxed{\sum_{i=1}^{m} \alpha_i y_i = 0.}$$

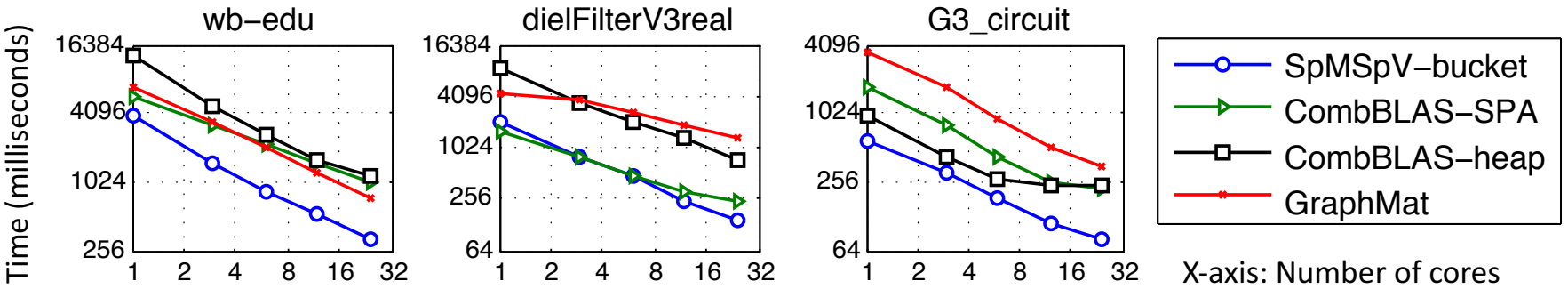
The equality constraint

Repeat until convergence:

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).
2. Re-optimize W(α) with respect to α_i and α_j, while holding all the other α_k 's fixed.

A work-efficient parallel algorithm for sparse matrix-sparse vector multiplication (SpMSpV)

- **Goal:** A scalable SpMSpV algorithm without doing more work on higher concurrency
- **Application:** Breadth-first search, graph matching, support vector machines, etc.
- **Algorithmic innovation:**
  - Attains work-efficiency by arranging necessary columns of the matrix into buckets where each bucket is processed by a single thread
  - Avoids synchronization by row-wise partitioning of the matrix on the fly
- **Performance:**
  - First ever work-efficient algorithm for SpMSpV that attains up to 15x speedup on a 24-core Intel Ivy Bridge processor and up to 49x speedup on a 64-core KNL processor
  - Up to an order of magnitude faster than its competitors, especially for sparser vector



A.Azad, A. Buluç. A work-efficient parallel sparse matrix-sparse vector multiplication algorithm. IPDPS'17