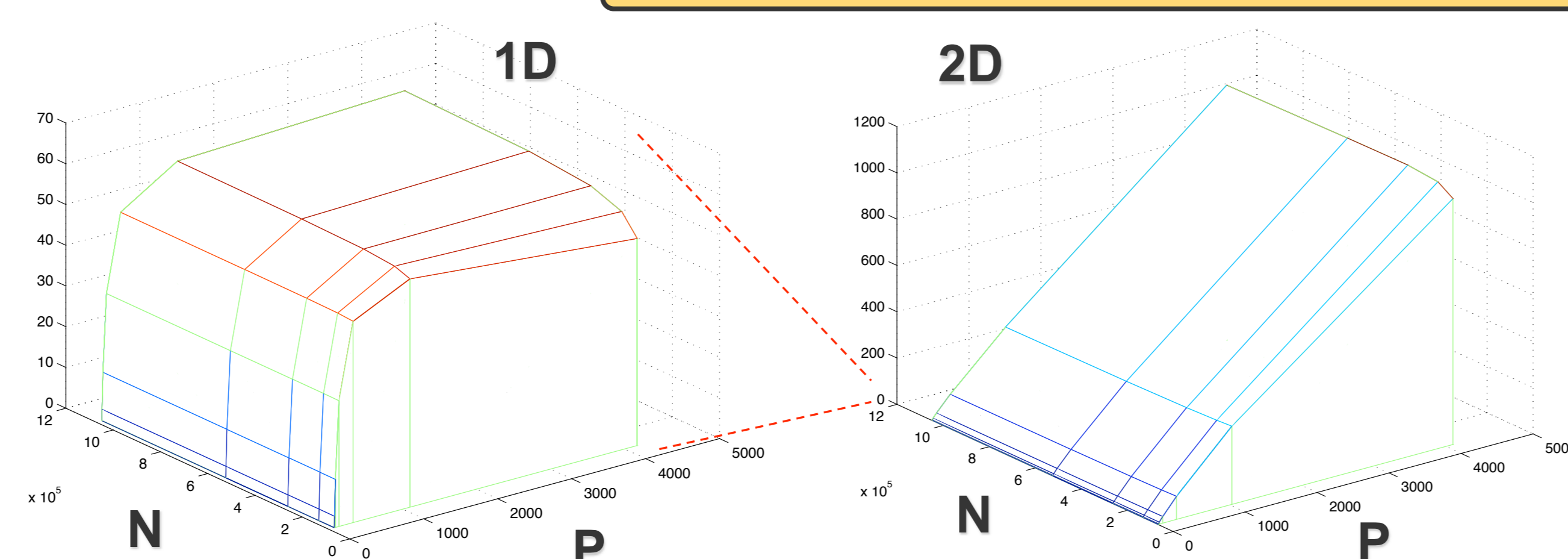
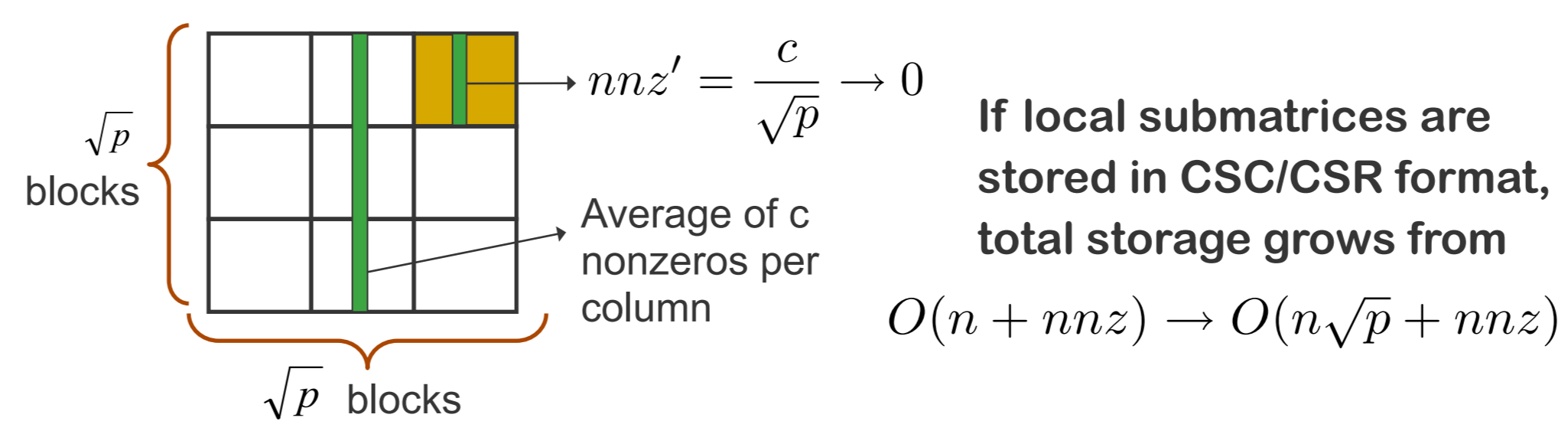


Distributed Memory Combinatorial BLAS

Parallel Data Distribution : Where does the data reside?



In 2D, submatrices are hypersparse ($nmz \ll n$)



- A data structure or algorithm that depends on the matrix dimension n (such as CSR/ CSC) is asymptotically too wasteful for submatrices.
- CSC and CSR are vertex based data structures, isomorphic to adjacency list representation, while the 2D distribution is based on edges.
- Our new data structure (**DCSC**) and sequential kernels solve this problem. Result: **work-efficient parallel algorithm**.
- Scalable sequential kernel uses novel ideas such as outer-product matrix multiplication with heap-assisted multi-way merging

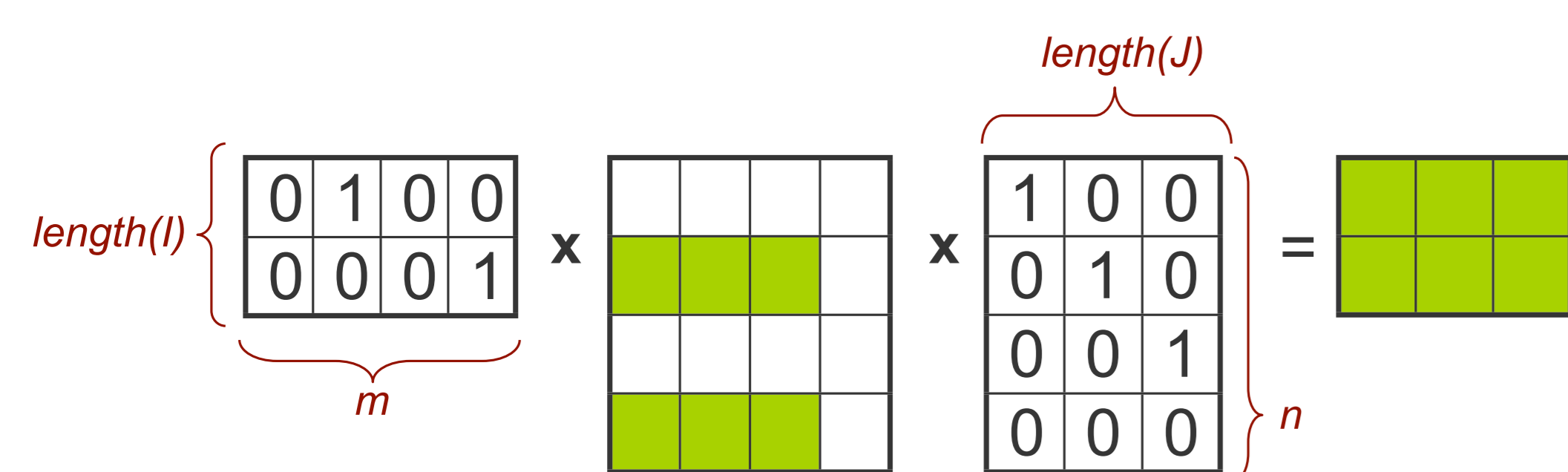
Estimated Speedup (Ideal) of 1D and 2D SpGEMM algorithms

- 2D algorithms have the **potential** to scale, if implemented correctly. Sparse matrix additions, overlapping communication, and maintaining load balance are crucial.
- Actual break-even point between 1D and 2D algorithms is around 50 processors. Performance of 1D algorithms flattens out around 40 processors.

Combinatorial BLAS: Linear Algebraic Primitives for Graphs

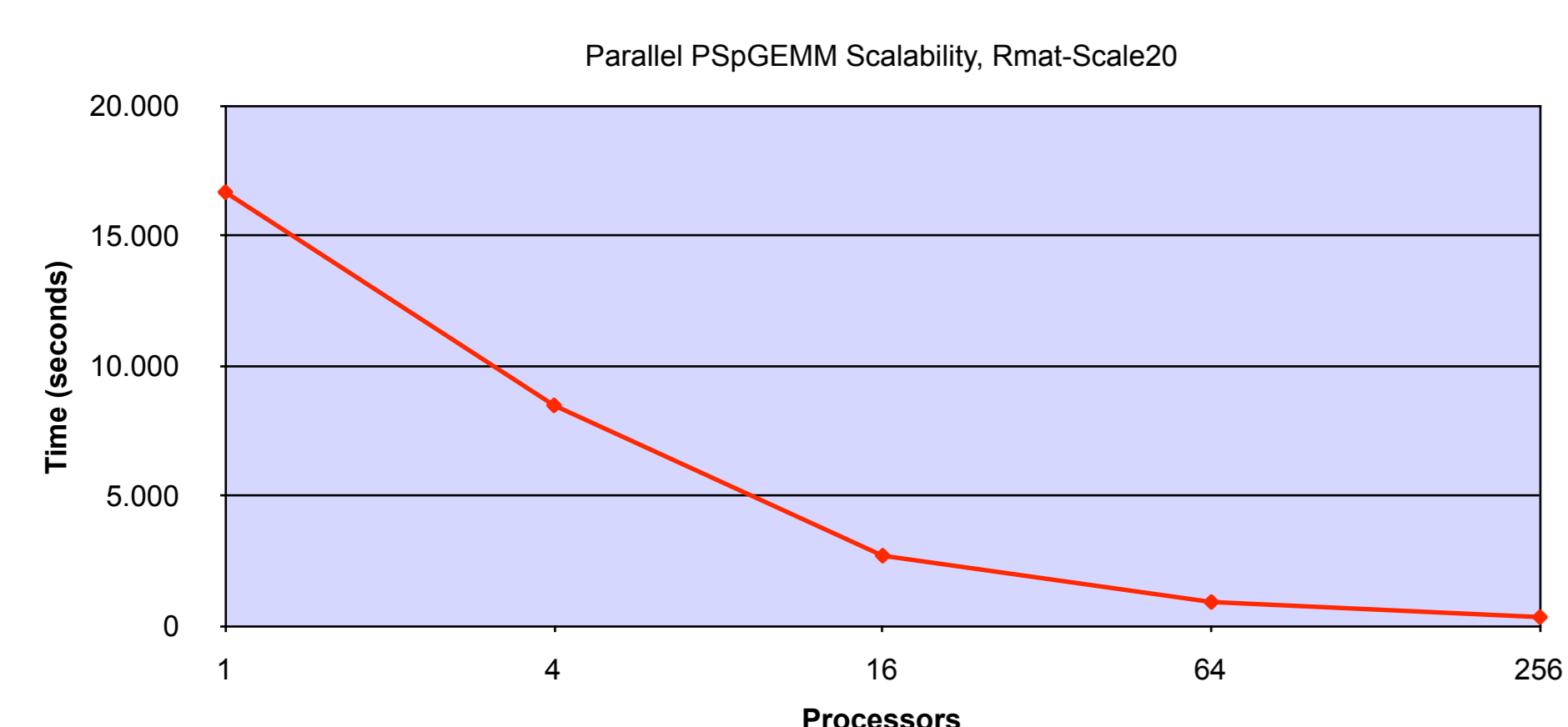
- Sparse matrix-matrix multiplication (SpGEMM)
Most general and challenging parallel primitive.
- Sparse matrix-vector multiplication (SpMV)
- Sparse matrix-transpose-vector multiplication (SpMVT)
Equivalently, multiplication from the left $y' \leftarrow x' A$
- Addition and other point-wise operations (SpAdd)
Included in SpGEMM
- Indexing and assignment (SpRef, SpAsgn)
 $A(I,J)$ where I and J are arrays of indices
Reduces to SpGEMM

Matrices on semirings, e.g. $(\times, +)$, (and, or) , $(+, \text{min})$

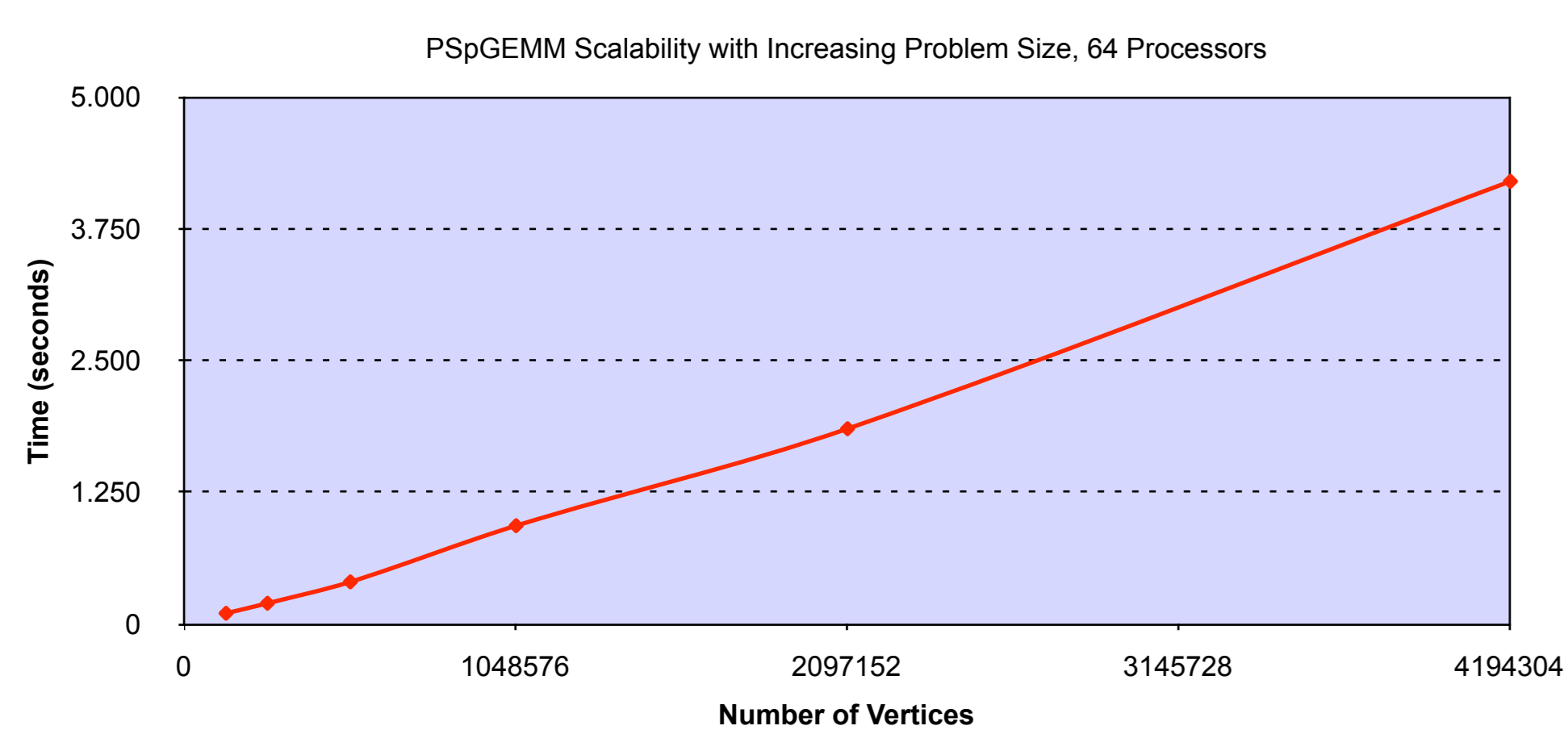


Sparse matrix indexing (SpRef) using mixed-mode sparse matrix-matrix multiplication (SpGEMM). On an m -by- n matrix A , the SpRef operation $A(I,J)$ extracts a submatrix of size $\text{length}(I) \times \text{length}(J)$, where I is a vector of row indices and J is a vector of column indices. The example shows the operation for $A([1,3],[0,1,2])$. It performs two SpGEMM operations between a boolean matrix and a floating point matrix.

SpGEMM Scalability on Lonestar (TACC)



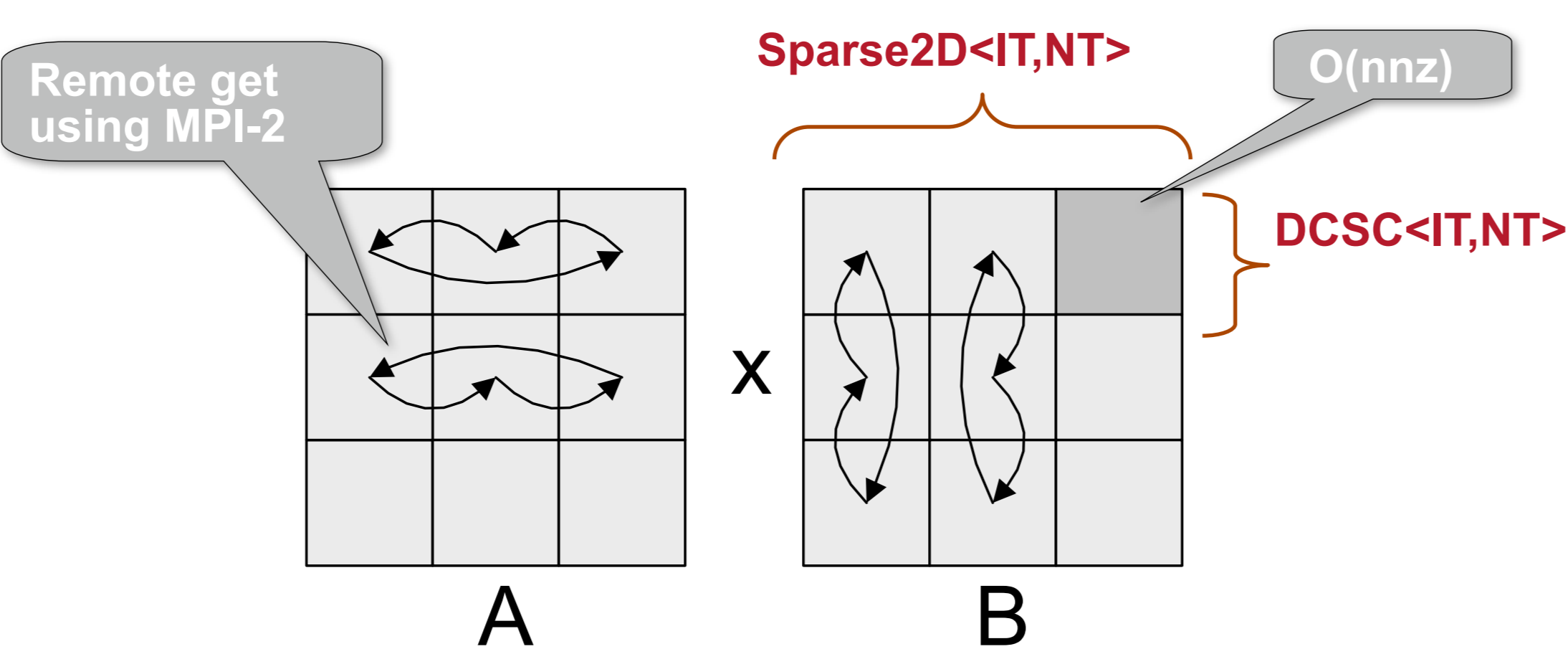
Parallel scalability of PSpGEMM with increasing number of processors. The product of two RMAT (Recursive MATrix generator) matrices is computed in parallel. RMAT is a synthetic graph generator that reproduces the power-law degrees of real-world graphs such as the internet graph, citations graph, etc.



Scalability of Parallel SpGEMM implementation with increasing problem size. Experiment was conducted with RMAT matrices of scale from 17 to 22. The sparsity is kept constant at $nmz = 8n$ (i.e., average vertex degree in the corresponding graph is 8).

Design Principles for Combinatorial BLAS

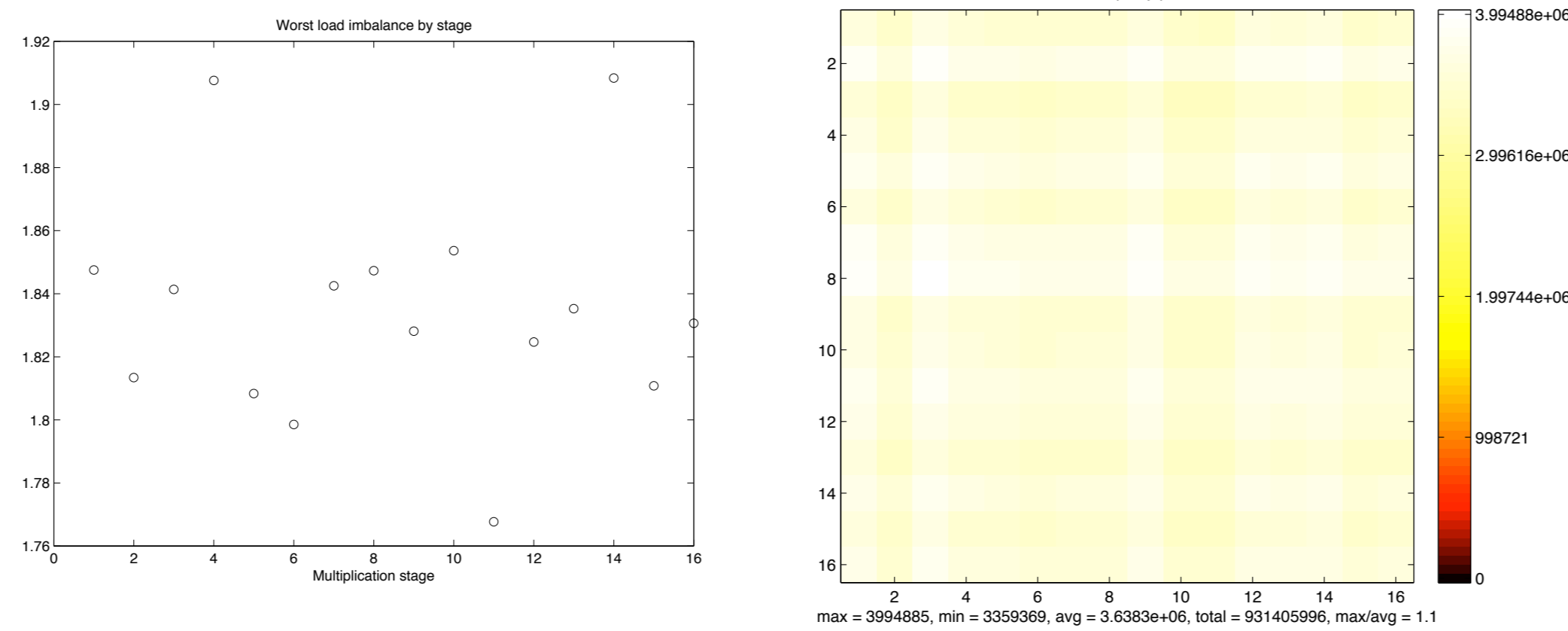
Parallelism is decoupled from the sequential kernels. Any sequential data structure that implements minimal set of functions can be used by the parallel sparse matrix class. Operations between any type of matrices (mixed-precision) and over any semiring are natively supported through type-traits and function objects.



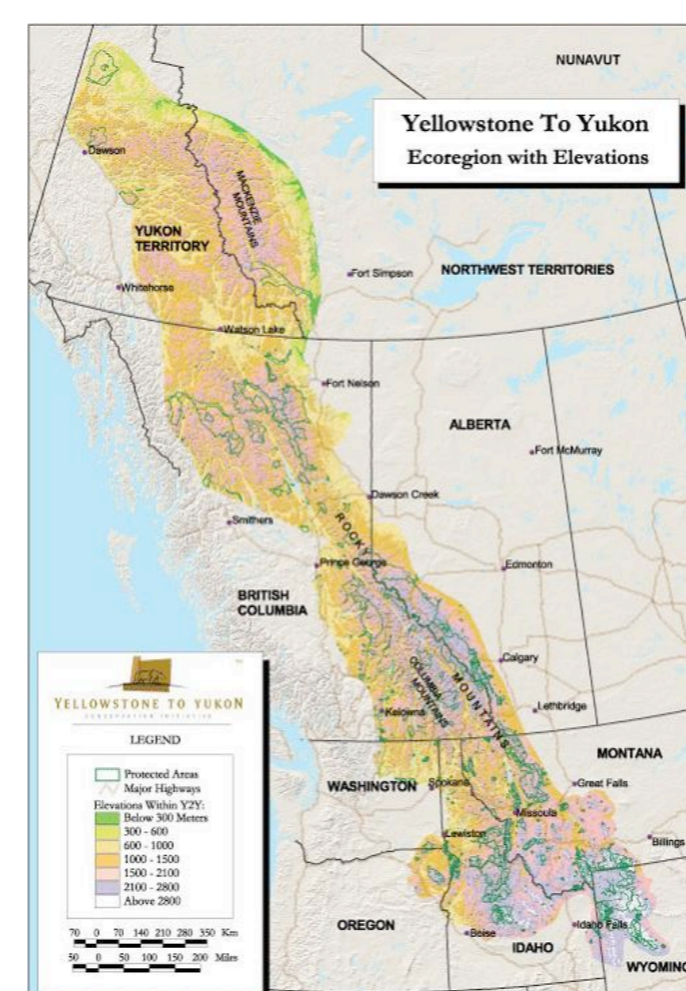
Distributed Memory SpGEMM

Load balance is achieved through symmetric random permutations, and asynchronous computation without the notion of stages. Overlapping communication with computation is crucial (achieved through one sided operations and RDMA support), because both communication and computation costs increase at the same rate as the problem size increases (unlike dense GEMM). For portability, we use MPI-2's passive target one-sided communication support. The algorithm avoids hot spots, i.e. with very high probability, a block is accessed at most by a single remote get operation at any given time

Significant load imbalance remains within synchronous stages, even after random permutation. An asynchronous algorithm does not suffer because the overall load on each processor is even when there are no stages (Test on RMAT matrices of scale 19)



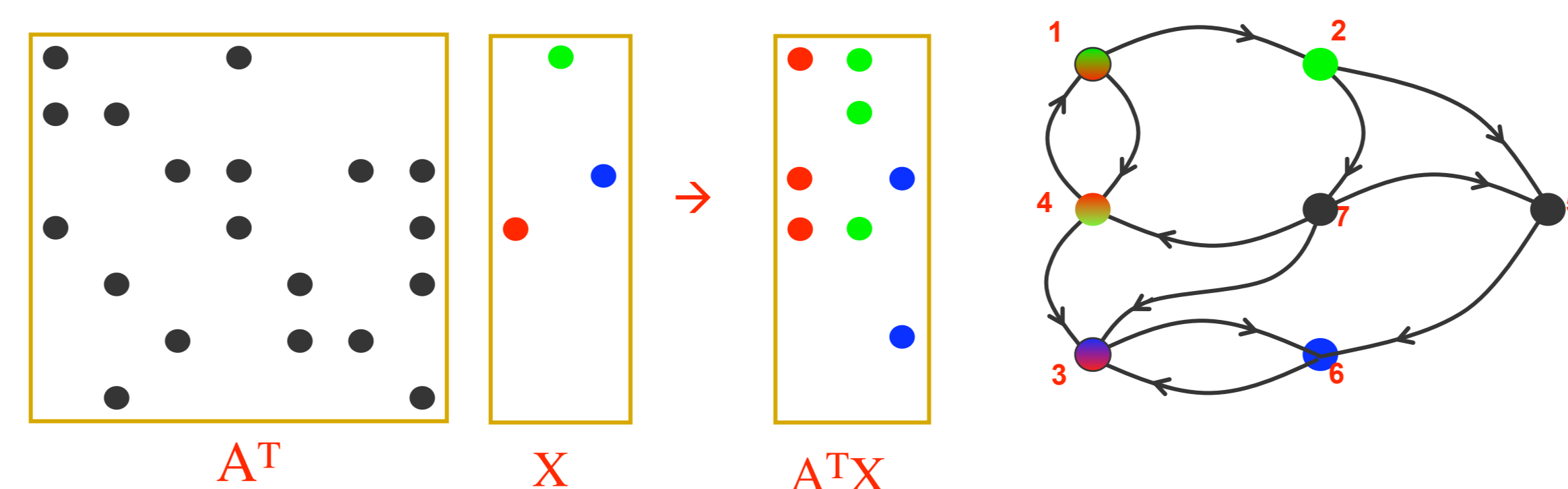
SpGEMM Applications



Landscape connectivity modeling using KDT, a toolbox for graph analysis and pattern discovery by G. Reinhardt, Shah, uses SpGEMM in building the connectivity graph. [Image courtesy of Brad McRae, NCEAS]



- Graph Clustering**
Markov Cluster Algorithm (MCL)
Clustering by peer pressure
- Betweenness Centrality**
Uses breadth-first search from multiple source vertices
- Subgraph Extraction**
Uses sparse matrix indexing
- Graph Contraction**
- Shortest Path Computations**
- Multigrid Interpolation & Restriction**
- Context-free Parsing**
- Cycle Detection**
- Interior point methods ($A^T A$)**
- Colored intersection searching**
- Applying Constraints in Finite Element Computations**



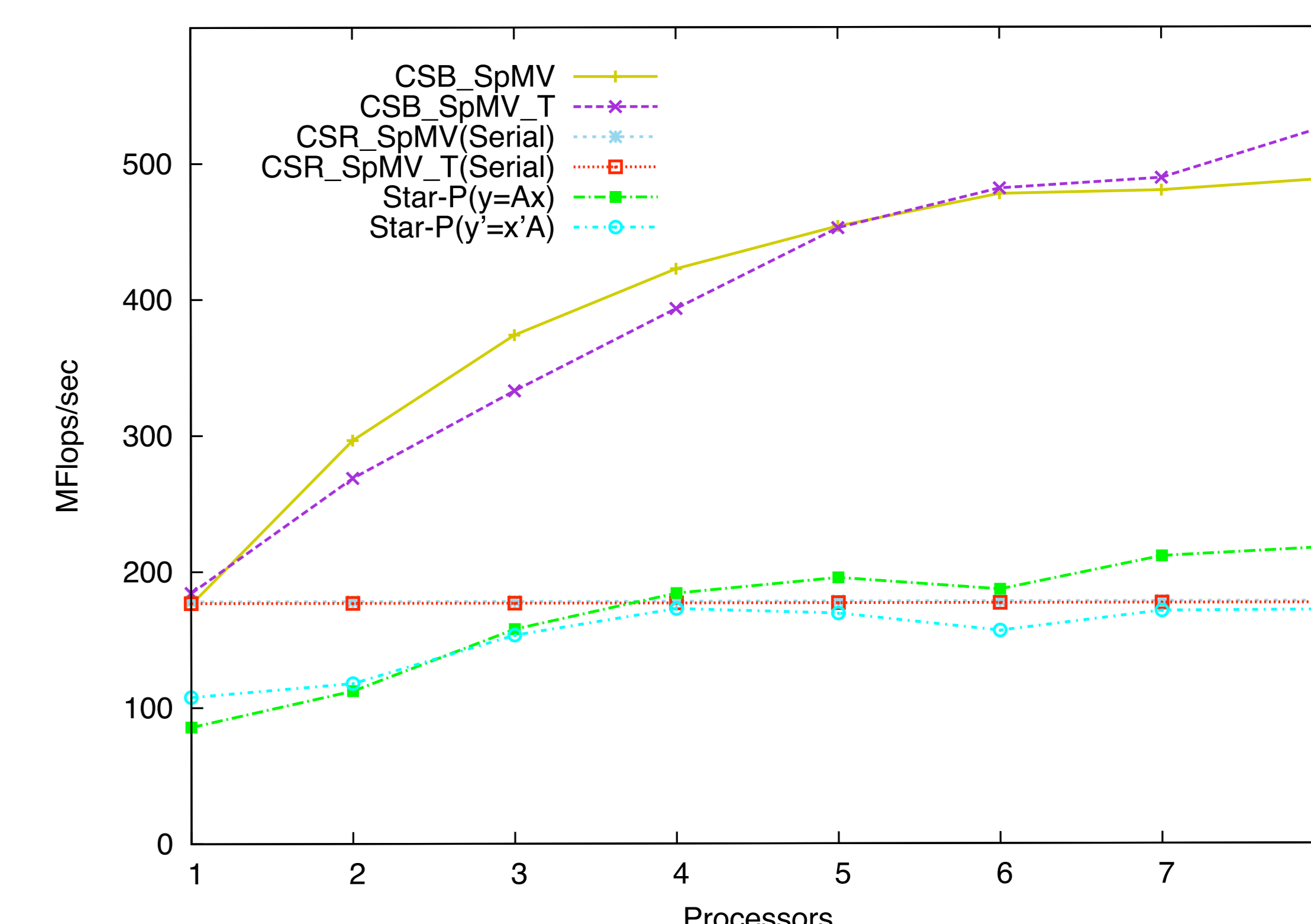
Space efficiency achieved through the sparse array representation. Work efficiency achieved using sparse matrix-matrix multiplication (SpGEMM). Load balance depends on SpGEMM implementation. It can also be performed by sparse matrix-vector multiplication with multiple right-hand-side vectors where A is streamed just once for multiple dense vectors, albeit paying less attention to sparsity.

Multicore Combinatorial BLAS

A symmetric sparse matrix data structure: CSB

Compressed Sparse Blocks (CSB) for representing sparse matrices requires only $n+nmz$ space for indices like CSR/CSR, but it is more symmetric than both. Because it does not favor rows over columns or vice versa, CSB admits a parallel algorithm which is efficient for computing either Ax or $A^T x$, as well as for computing Ax when A is symmetric and only half the matrix needs to be stored.

Scalable algorithms for SpMV (Ax) and SpMVT ($A^T x$) with $\theta(nmz)$ work and $\theta(\sqrt{n} \log n)$ span, yield ample parallelism. No parallelism overheads, runs comparably fast to CSR/CSC on one processor and scales up linearly until memory-bandwidth limitations are encountered. It remains to be seen whether CSB can be used effectively to parallelize sparse matrix-matrix multiplication on multicore and shared-memory architectures.



Average performance of Ax and $A^T x$ operations on a benchmark suite of 13 different (mostly irregular) sparse matrices. CSB_SpMV and CSB_SpMVT use compressed sparse blocks to perform Ax and $A^T x$, respectively. CSR_SpMV and CSR_SpMVT use OSKI and compressed sparse rows without any matrix-specific optimizations. Star-P ($y=Ax$) and Star-P ($y'=x'A$) use Star-P, a parallel code based on CSR. Experiments were run on a ccNUMA architecture powered by AMD Opteron 8214 (Santa Rosa) processors

FINANCIAL SUPPORT:

Parts of this work was supported by MIT Lincoln Laboratory under contract number 7000012980, the Department of Energy under award number DE-FG02-04ER25632, the National Science Foundation under award number 0709385 and through TeraGrid resources provided by Texas Advanced Computing Center.

REFERENCES:

- Aydın Buluç and John R. Gilbert, "On the Representation and Multiplication of Hypersparse Matrices", *The 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008)*, Miami, FL, April 14-18, 2008
- Aydın Buluç and John R. Gilbert, "Challenges and Advances in Parallel Sparse Matrix-Matrix Multiplication", *The 37th International Conference on Parallel Processing (ICPP 2008)*, Portland, Oregon, USA, 2008.
- Aydın Buluç, Jeremy T. Fineman, Matteo Frigo, John R. Gilbert, Charles E. Leiserson, "Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication using Compressed Sparse Blocks", *The 21st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2009)*, Calgary, Canada, 2009
- Aydın Buluç, John R. Gilbert, and Ceren Budak, "Gaussian Elimination Based Algorithms on the GPU", *Under review for the Special Issue of Parallel Computing on Parallel Matrix Algorithms and Applications (available as Technical Report UCSB/CS-2008-15)*.