



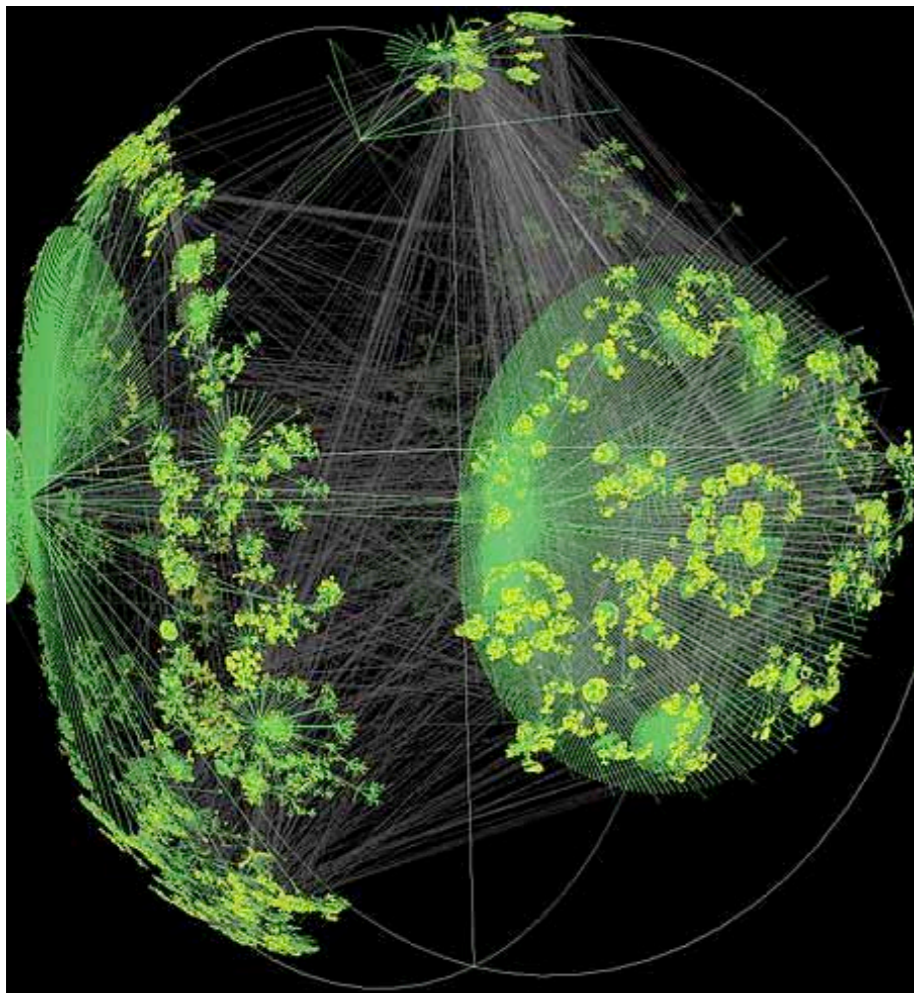
Scalable Parallel Primitives for Massive Graph Computation

Aydın Buluç

University of California, Santa Barbara

Sources of Massive Graphs

Graphs naturally arise from the internet and social interactions



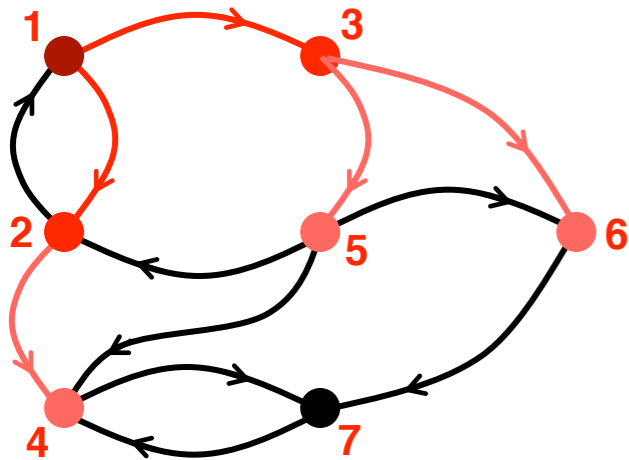
(WWW snapshot, courtesy Y. Hyun)

Many scientific (biological, chemical, cosmological, ecological, etc) datasets are modeled as graphs.



(Yeast protein interaction network, courtesy H. Jeong)

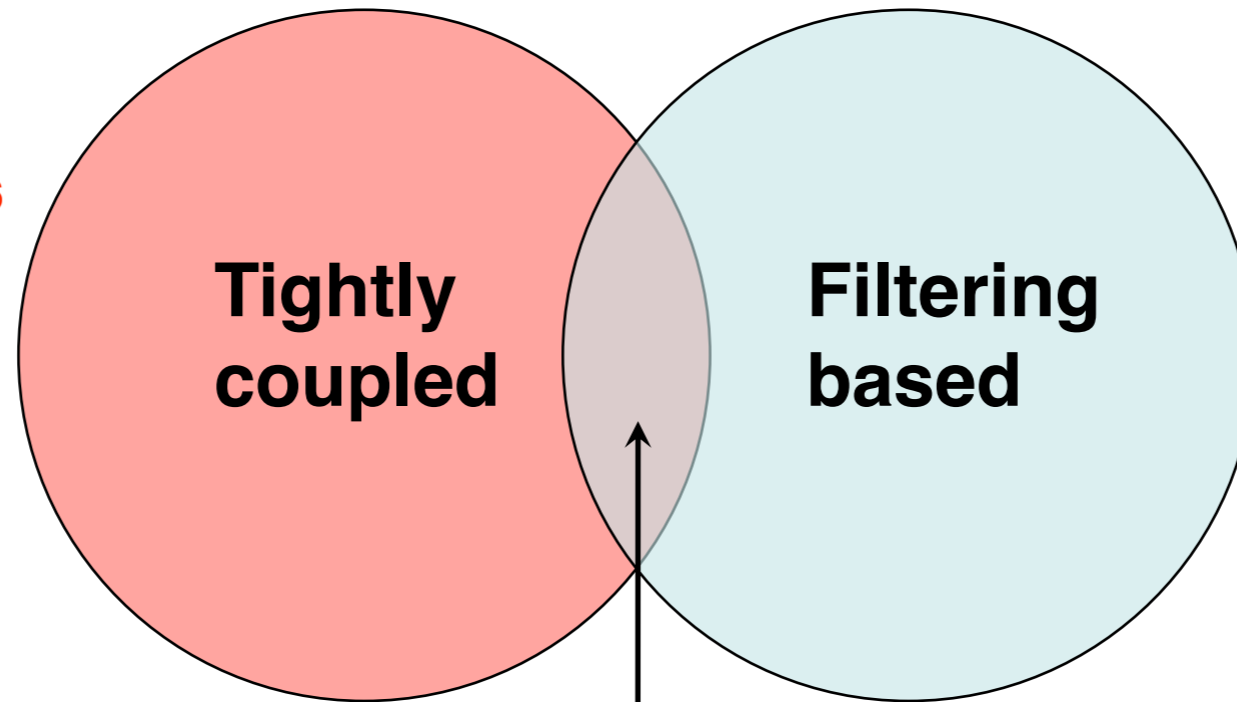
Types of Graph Computations



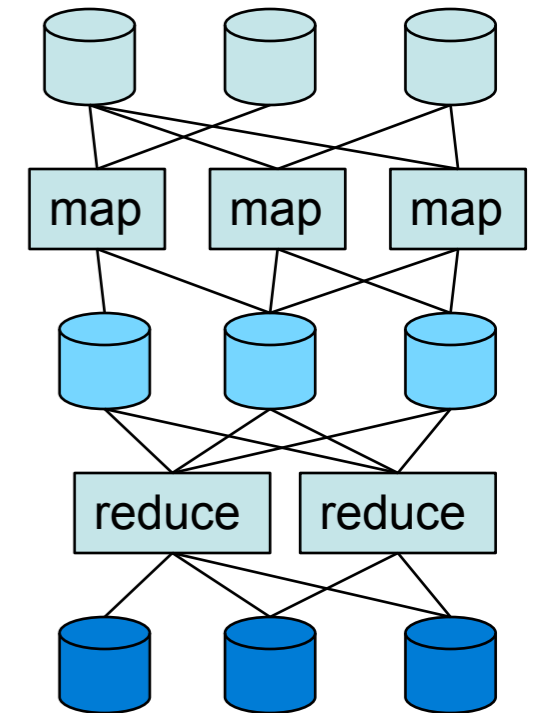
Tool: Graph Traversal

Examples:

- Centrality
- Shortest paths
- Network flows
- Strongly Connected Components



Fuzzy intersection
Examples: Clustering,
Algebraic Multigrid



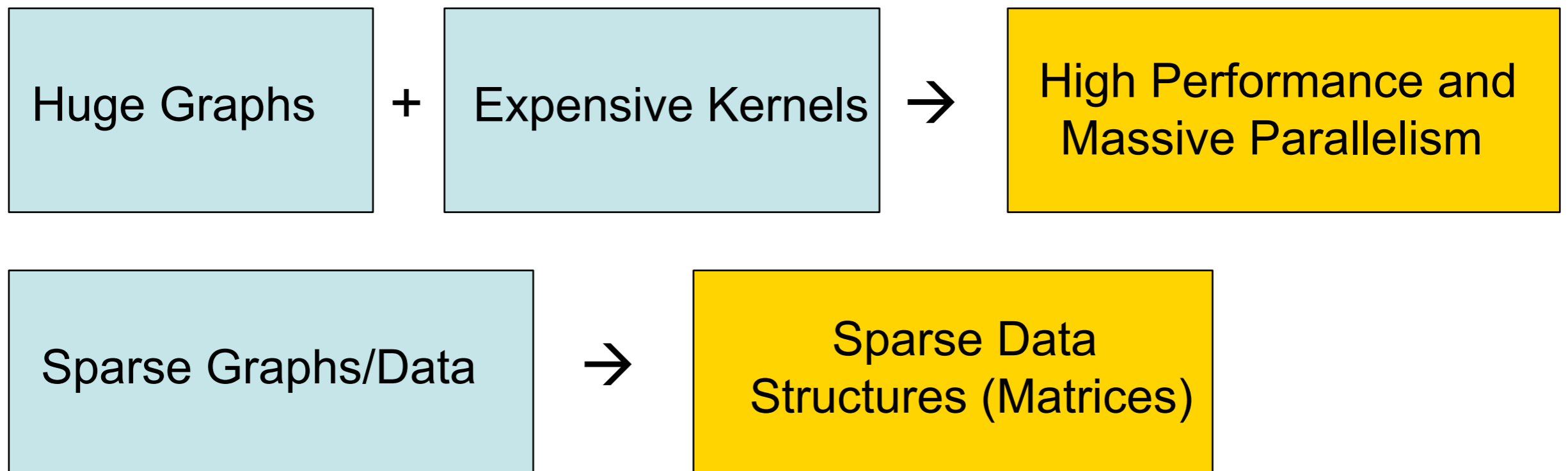
Tool: Map/Reduce

Examples:

- Loop and multi edge removal
- Triangle/Rectangle enumeration

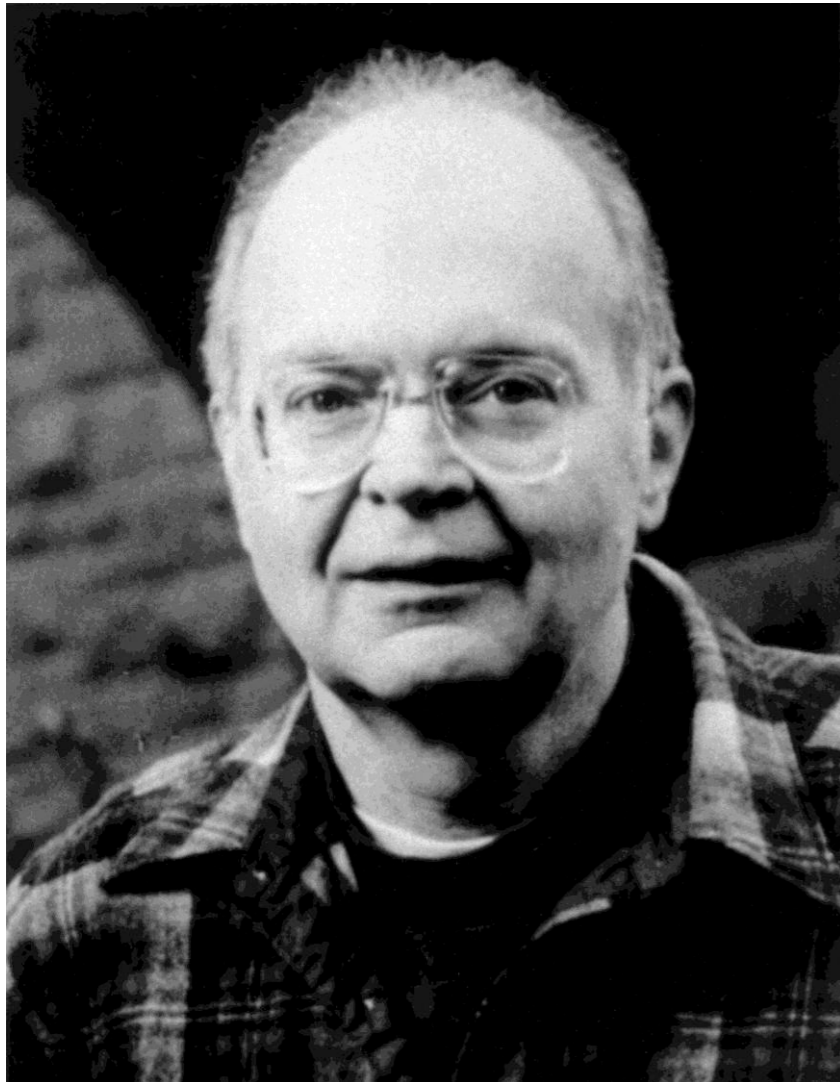
Tightly Coupled Computations on Sparse Graphs

- Many graph mining algorithms are computationally intensive. (e.g. graph clustering, centrality)
- Some computations are inherently latency-bound. (e.g. finding shortest paths)
- Interesting graphs are sparse, typically $|\text{edges}| = O(|\text{vertices}|)$



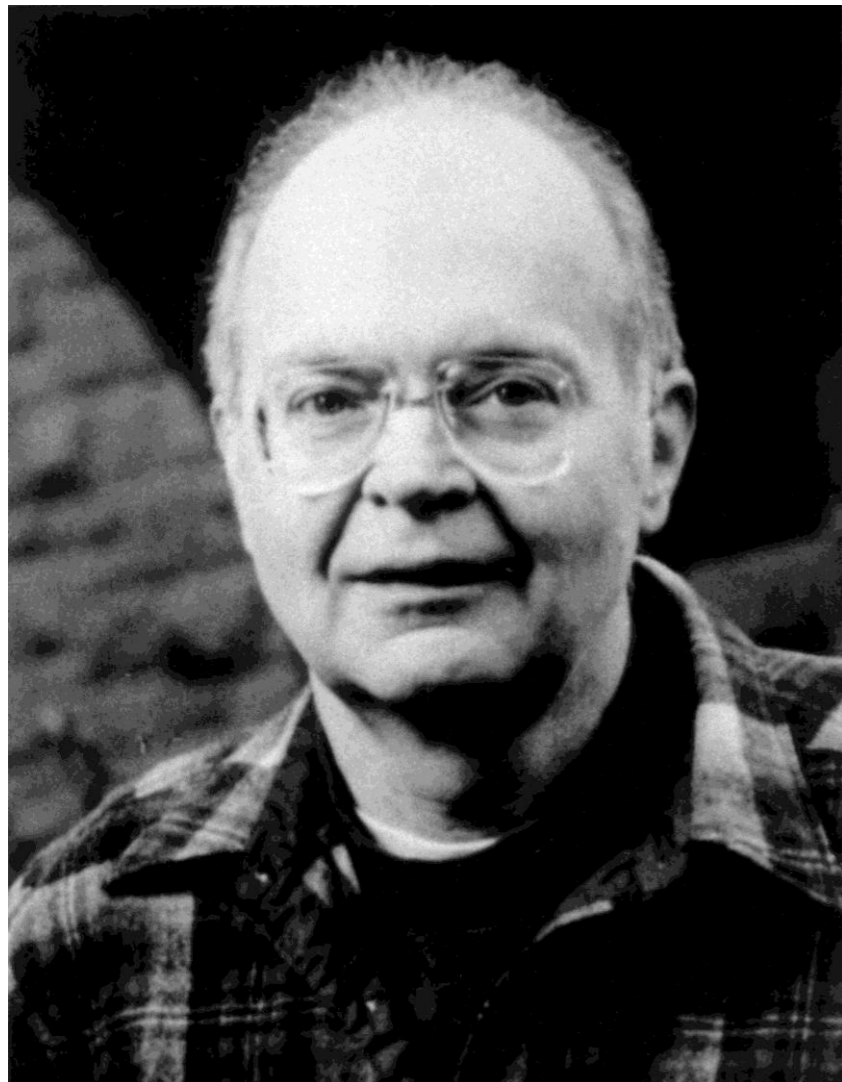
Software for Graph Computation

“...my main conclusion after spending ten years of my life on the TeX project is that software is hard. It's harder than anything else I've ever had to do”



Software for Graph Computation

“...my main conclusion after spending ten years of my life on the TeX project is that software is hard. It's harder than anything else I've ever had to do”

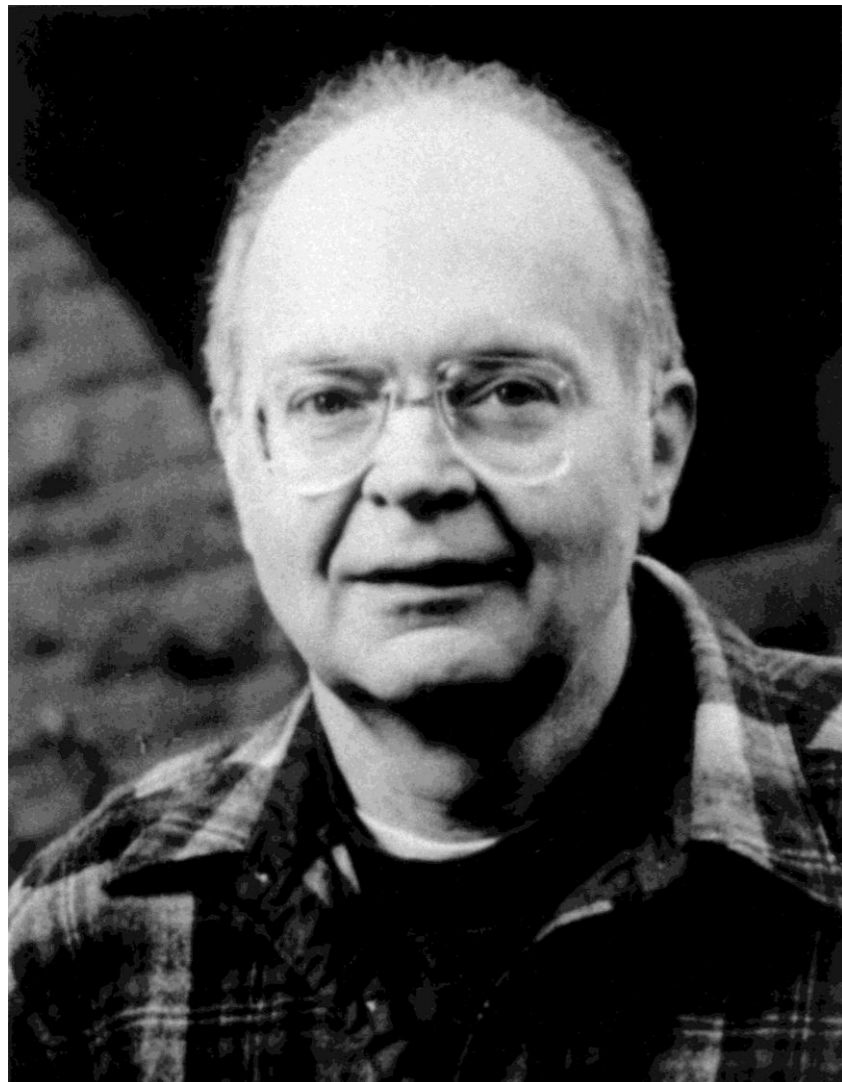


Dealing with software is hard !



Software for Graph Computation

“...my main conclusion after spending ten years of my life on the TeX project is that software is hard. It's harder than anything else I've ever had to do”



Dealing with software is hard !

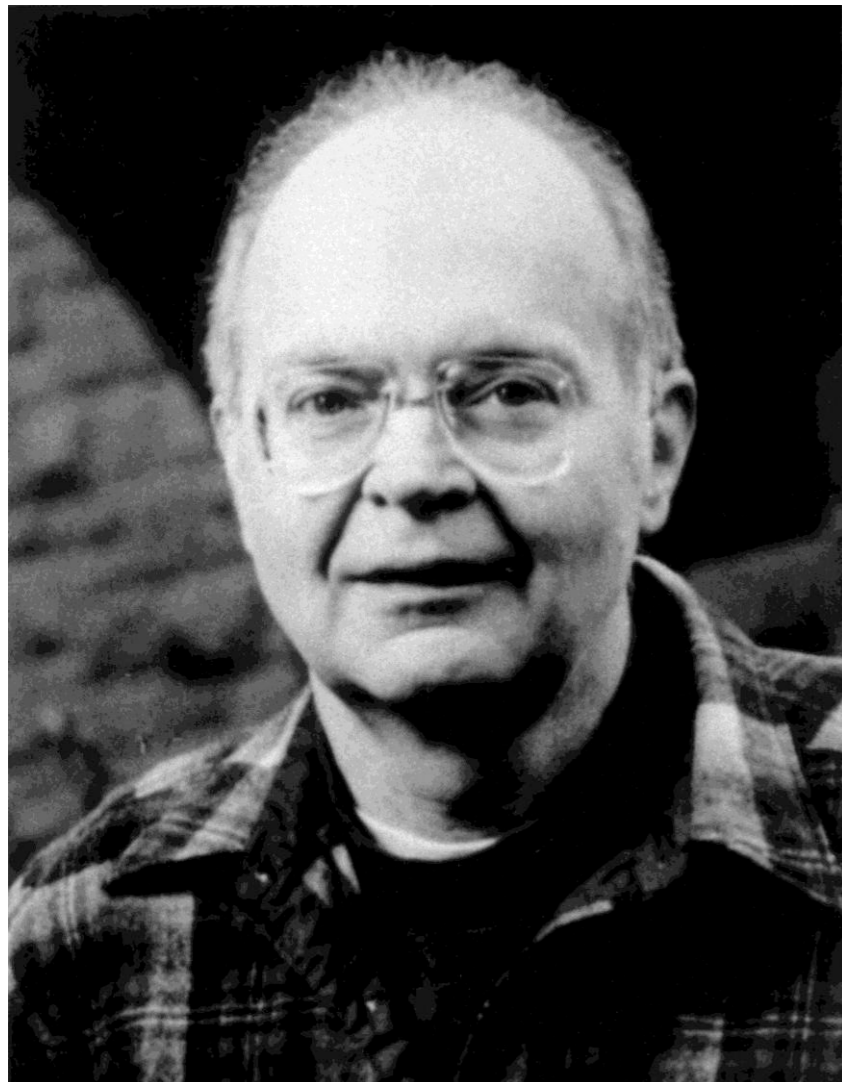


High performance computing (HPC) software is harder !



Software for Graph Computation

“...my main conclusion after spending ten years of my life on the TeX project is that software is hard. It's harder than anything else I've ever had to do”



Dealing with software is hard !



High performance computing (HPC) software is harder !



Deal with parallel HPC software?

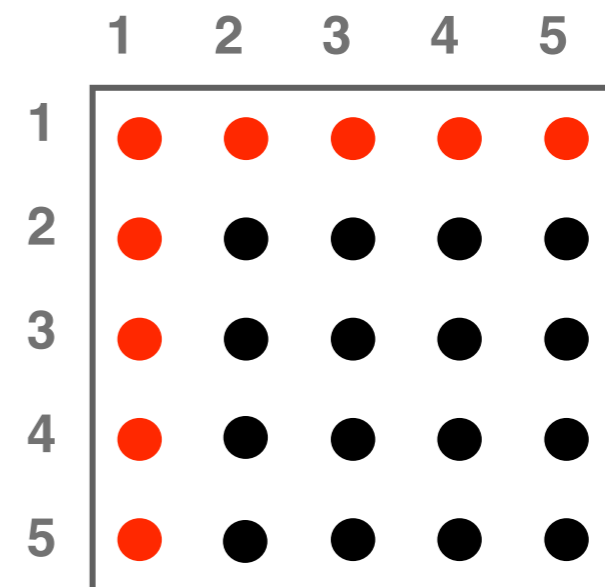
Outline

- **The Case for Primitives**
- The Case for Sparse Matrices
- Parallel Sparse Matrix-Matrix Multiplication
- Software Design of the Combinatorial BLAS
- An Application in Social Network Analysis
- Other Work
- Future Directions

All-Pairs Shortest Paths

- Input: Directed graph with “costs” on edges
- Find least-cost paths between all reachable vertex pairs
- Classical algorithm: Floyd-Warshall

```
for k=1:n // the induction sequence
  for i = 1:n
    for j = 1:n
      if( $w(i \rightarrow k) + w(k \rightarrow j) < w(i \rightarrow j)$ )
         $w(i \rightarrow j) := w(i \rightarrow k) + w(k \rightarrow j)$ 
```



k = 1 case

- Case study of implementation on multicore architecture:
 - graphics processing unit (GPU)

GPU characteristics

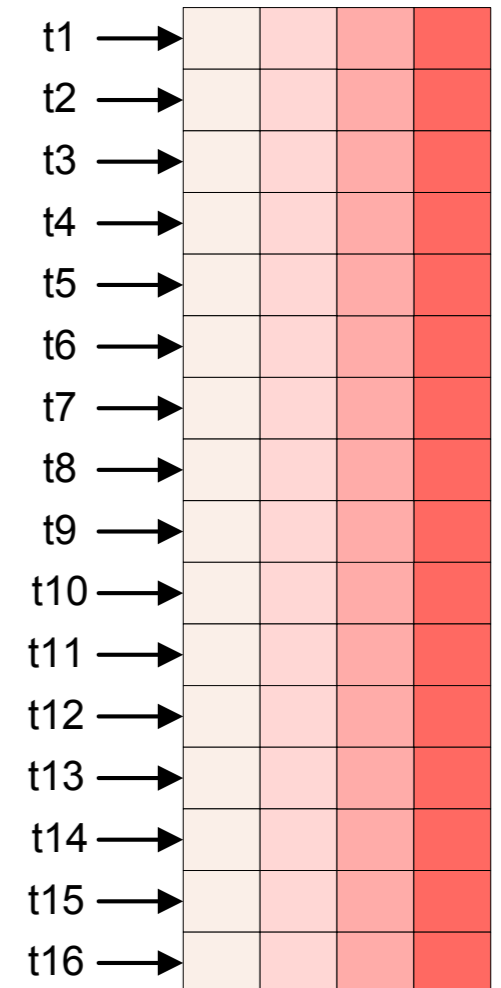


Powerful: two Nvidia 8800s > 1 TFLOPS

Inexpensive: \$500 each

But:

- Difficult programming model:
 - One instruction stream drives 8 arithmetic units
- Performance is counterintuitive and fragile:
 - Memory access pattern has subtle effects on cost
- Extremely easy to underutilize the device:
 - Doing it wrong easily costs 100x in time

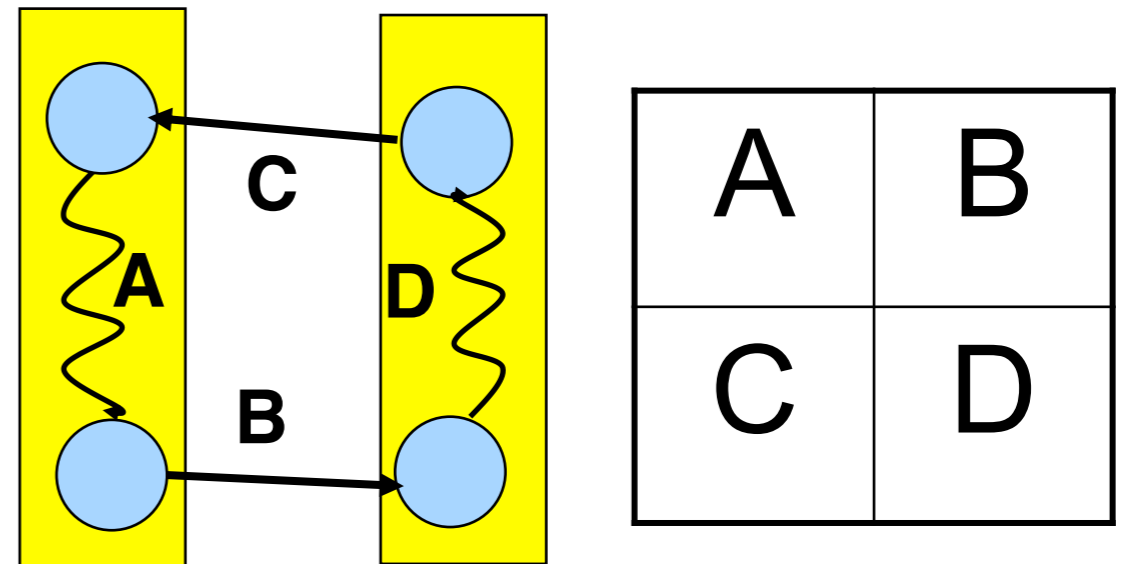


Recursive All-Pairs Shortest Paths

Based on R-Kleene algorithm

Well suited for GPU architecture:

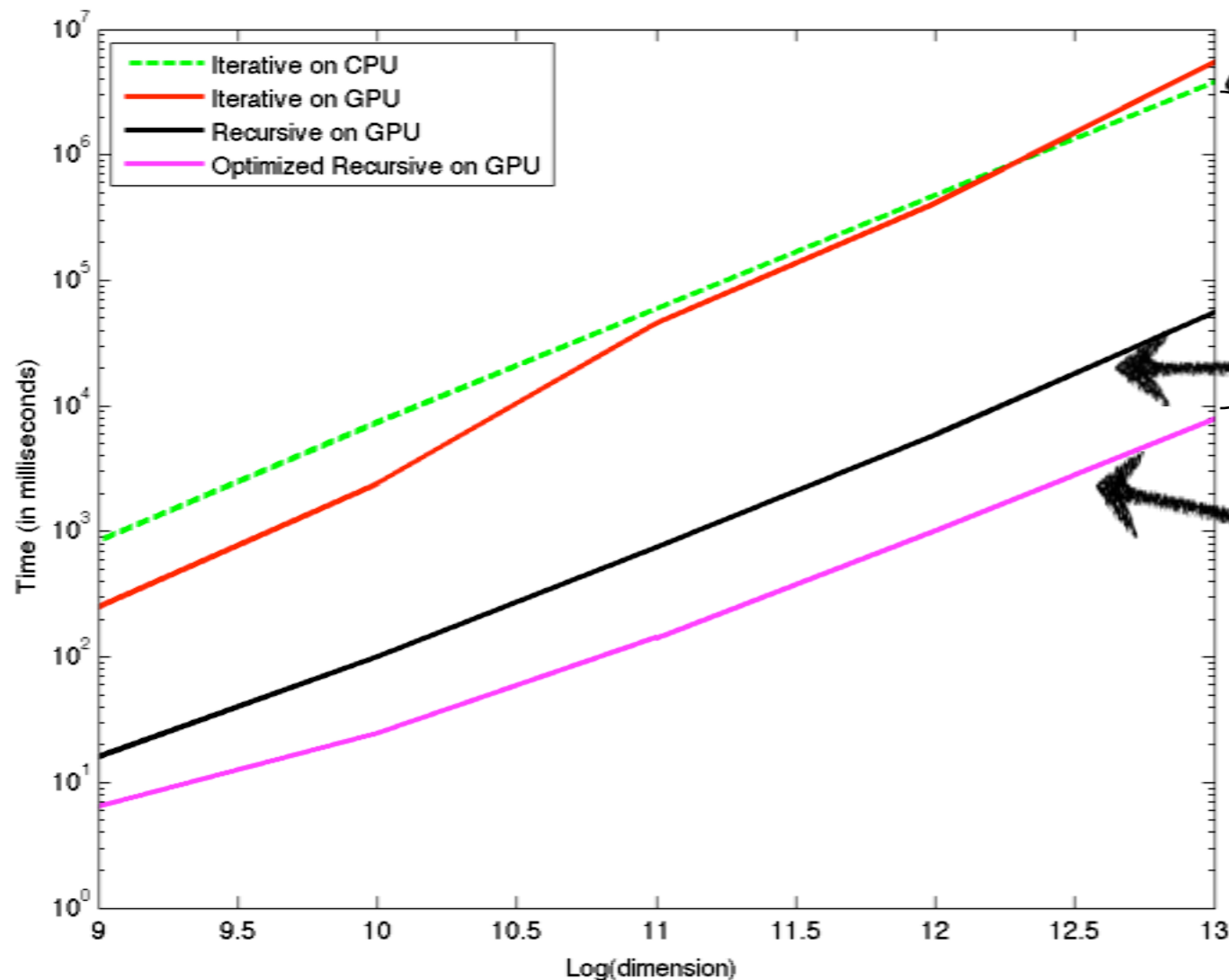
- Fast matrix-multiply kernel
- In-place computation => low memory bandwidth
- Few, large MatMul calls => low GPU dispatch overhead
- Recursion stack on host CPU, not on multicore GPU
- Careful tuning of GPU code



+ is "min", × is "add"

```
A = A*;  
B = AB; C = CA;  
D = D + CB;  
D = D*;  
B = BD; C = DC;  
A = A + BC;
```

APSP: Experiments and observations



Lifting Floyd-Warshall to GPU

480x

Unorthodox R-Kleene algorithm

The right primitive !

Conclusions:

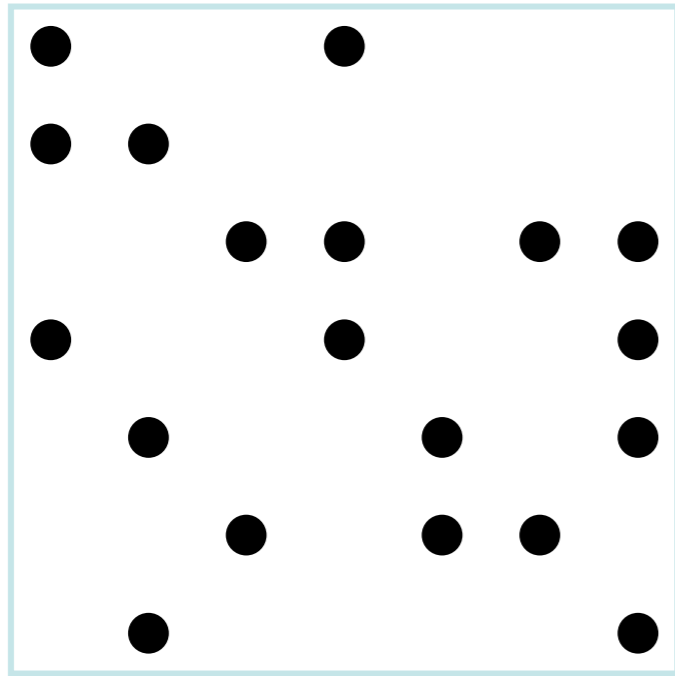
High performance is achievable but not simple

Carefully chosen and optimized **primitives** will be key

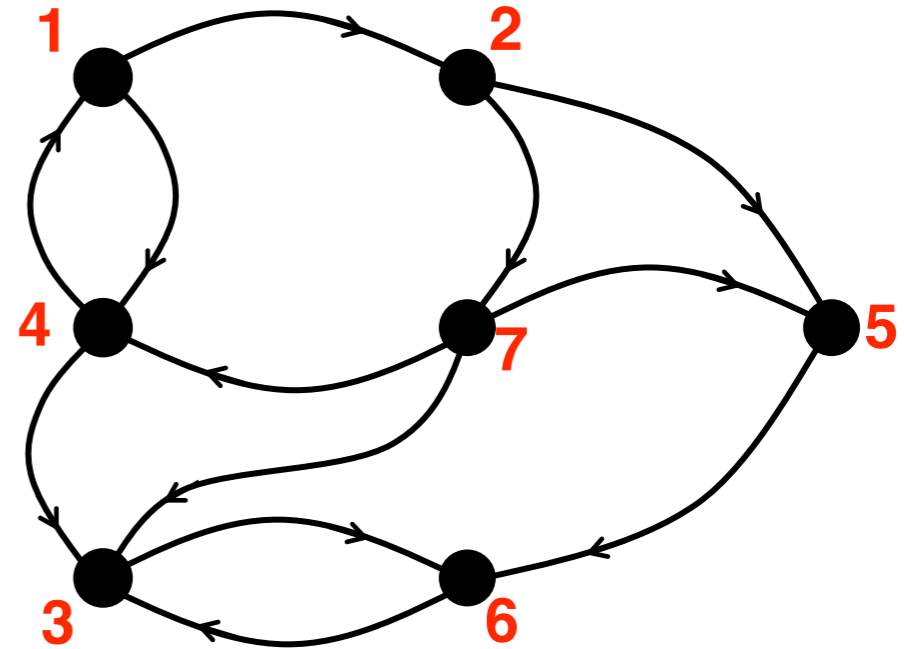
Outline

- The Case for Primitives
- **The Case for Sparse Matrices**
- Parallel Sparse Matrix-Matrix Multiplication
- Software Design of the Combinatorial BLAS
- An Application in Social Network Analysis
- Other Work
- Future Directions

Sparse Adjacency Matrix and Graph



A^T



- Every graph is a sparse matrix and vice-versa
- Adjacency matrix: sparse array w/ nonzeros for graph edges
- Storage-efficient implementation from sparse data structures

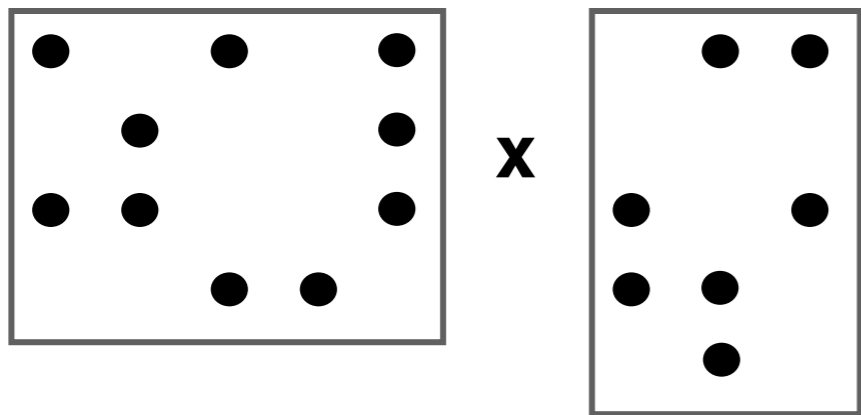
The Case for Sparse Matrices

- Many irregular applications contain sufficient coarse-grained parallelism that can ONLY be exploited using abstractions at proper level.

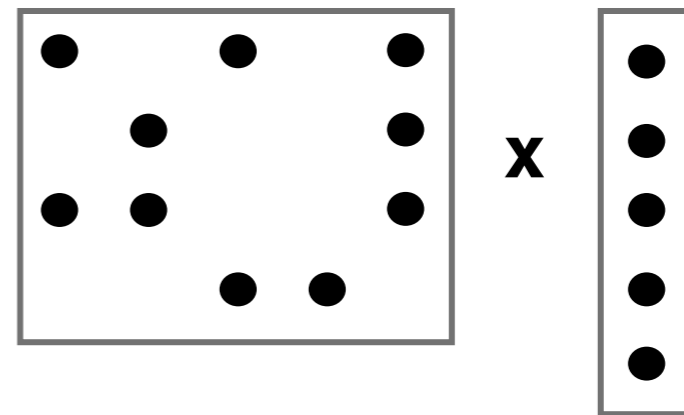
Traditional graph computations	Graphs in the language of linear algebra
Data driven. Unpredictable communication.	Fixed communication patterns.
Irregular and unstructured. Poor locality of reference	Operations on matrix blocks. Exploits memory hierarchy
Fine grained data accesses. Dominated by latency	Coarse grained parallelism. Bandwidth limited

Linear Algebraic Primitives

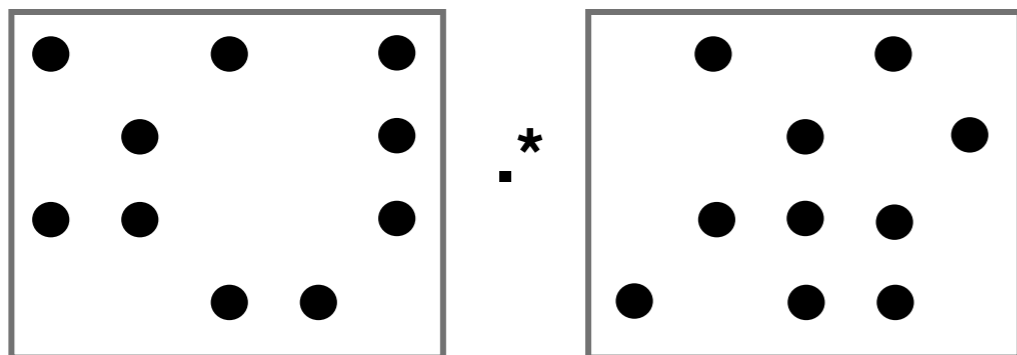
Sparse matrix-matrix
Multiplication (SpGEMM)



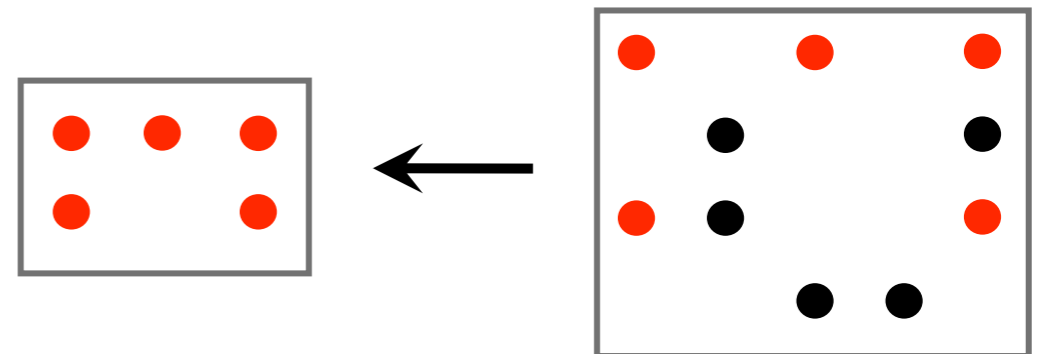
Sparse matrix-vector
multiplication



Element-wise operations



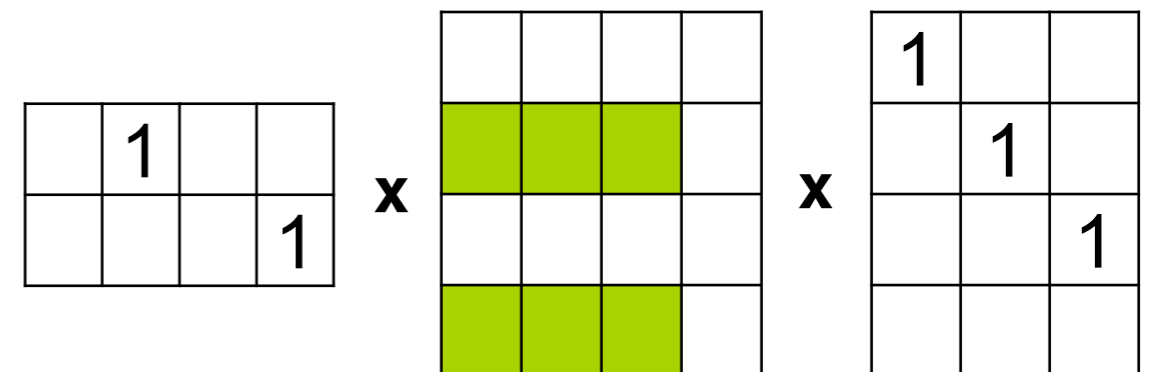
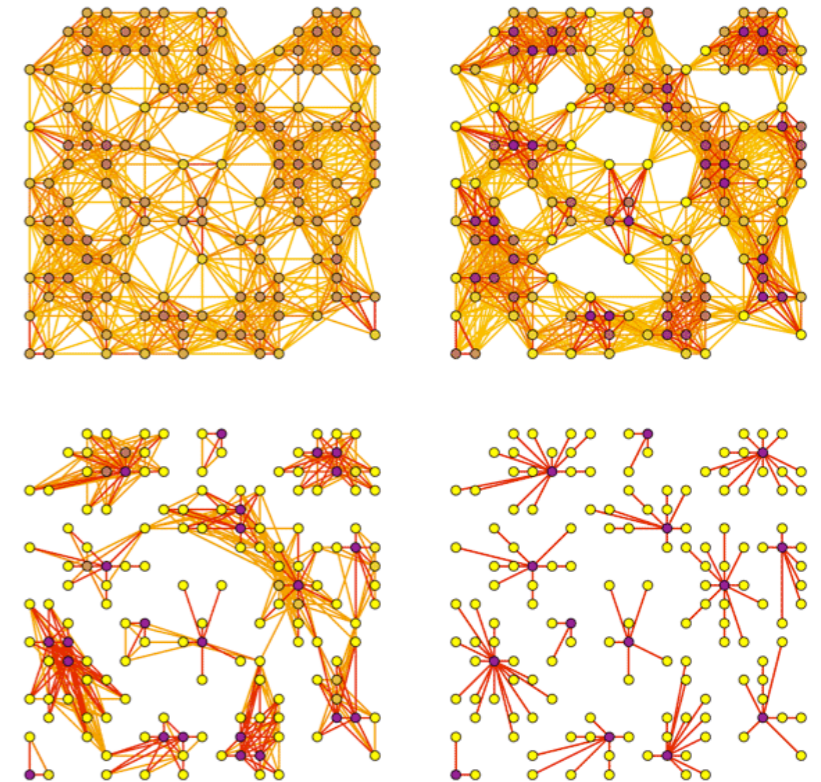
Sparse Matrix Indexing



Matrices on semirings, e.g. $(\cdot, +)$, (and, or), $(+, \min)$

Applications of Sparse GEMM

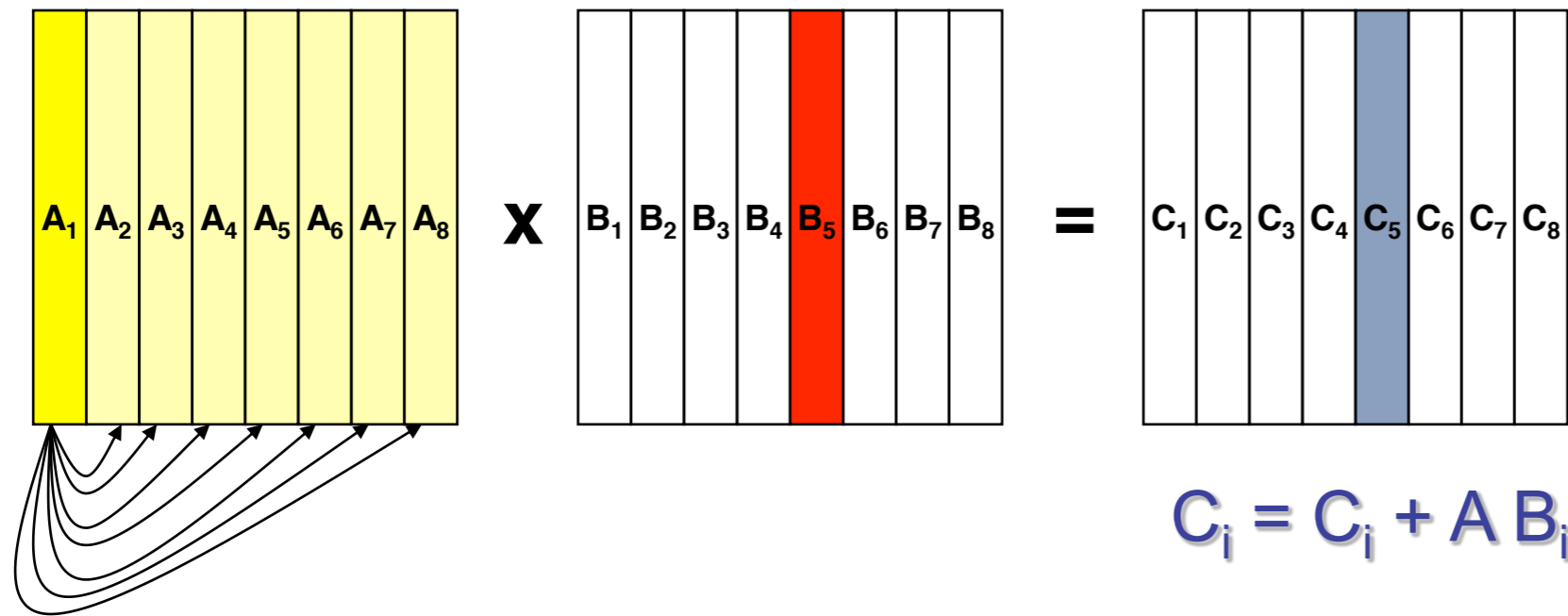
- Graph clustering (Markov, peer pressure)
- Subgraph / submatrix indexing
- Shortest path calculations
- Betweenness centrality
- Graph contraction
- Cycle detection
- Multigrid interpolation & restriction
- Colored intersection searching
- Applying constraints in finite element computations
- Context-free parsing ...



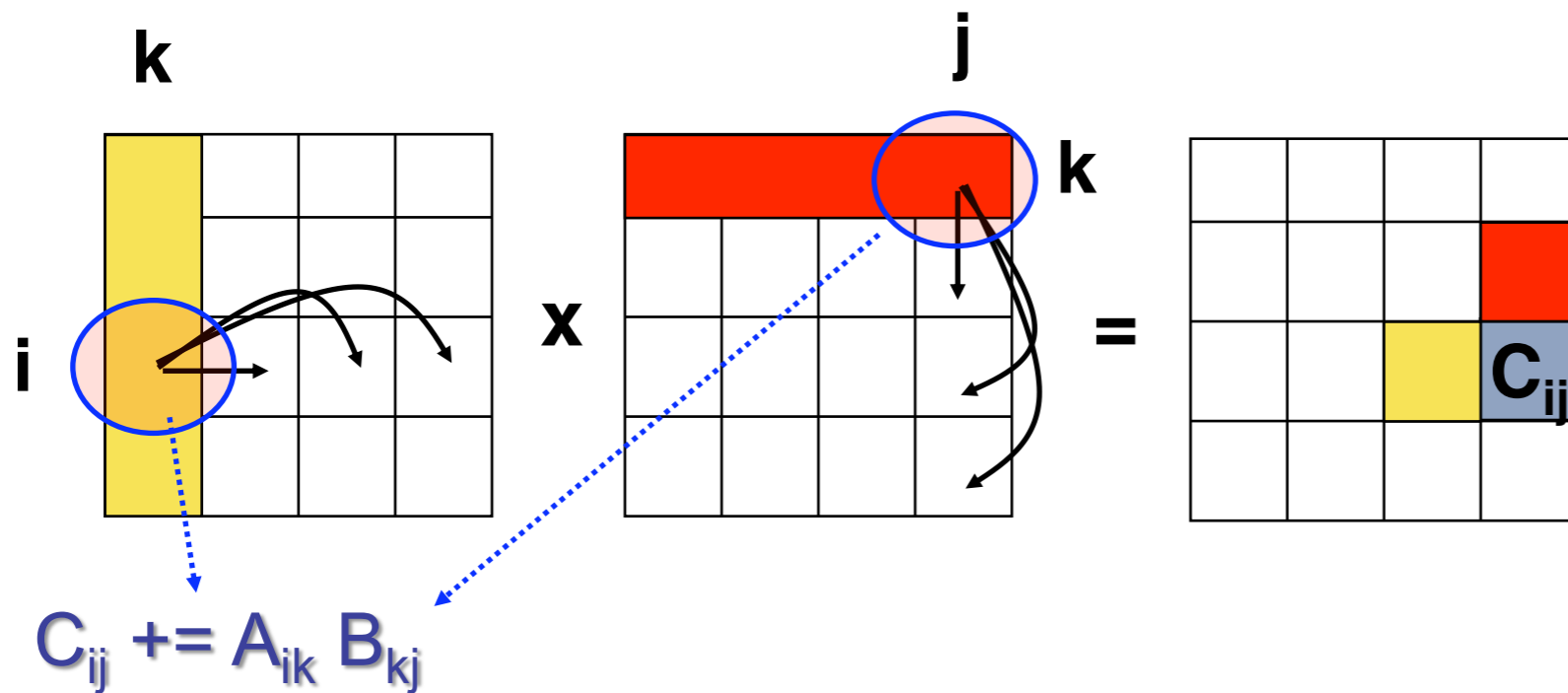
Outline

- The Case for Primitives
- The Case for Sparse Matrices
- **Parallel Sparse Matrix-Matrix Multiplication**
- Software Design of the Combinatorial BLAS
- An Application in Social Network Analysis
- Other Work
- Future Directions

Two Versions of Sparse GEMM

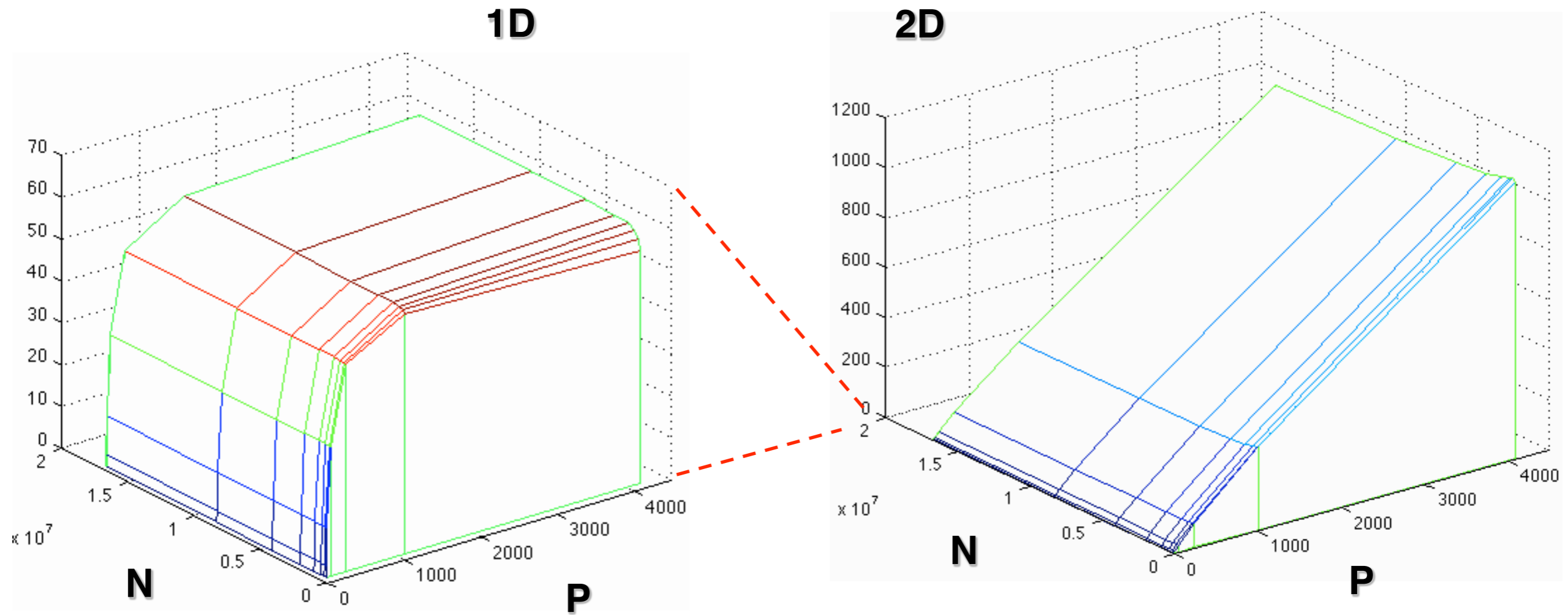


1D block-column distribution



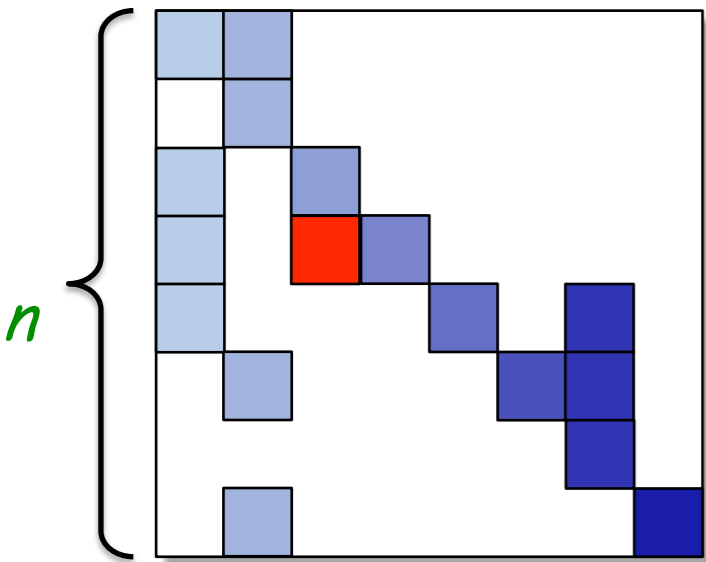
Checkerboard (2D block) distribution

Projected performances of Sparse 1D & 2D



In practice, 2D algorithms have the potential to scale, if implemented correctly. Overlapping communication, and maintaining load balance are crucial.

Compressed Sparse Columns (CSC): A Standard Layout



$n \times n$ matrix with
 nnz nonzeros

Column pointers

0
4
8
10
11
12
13
16
17

rowind

0	2	3	4	0	1	5	7	2	3	3	4	5	4	5	6	7
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

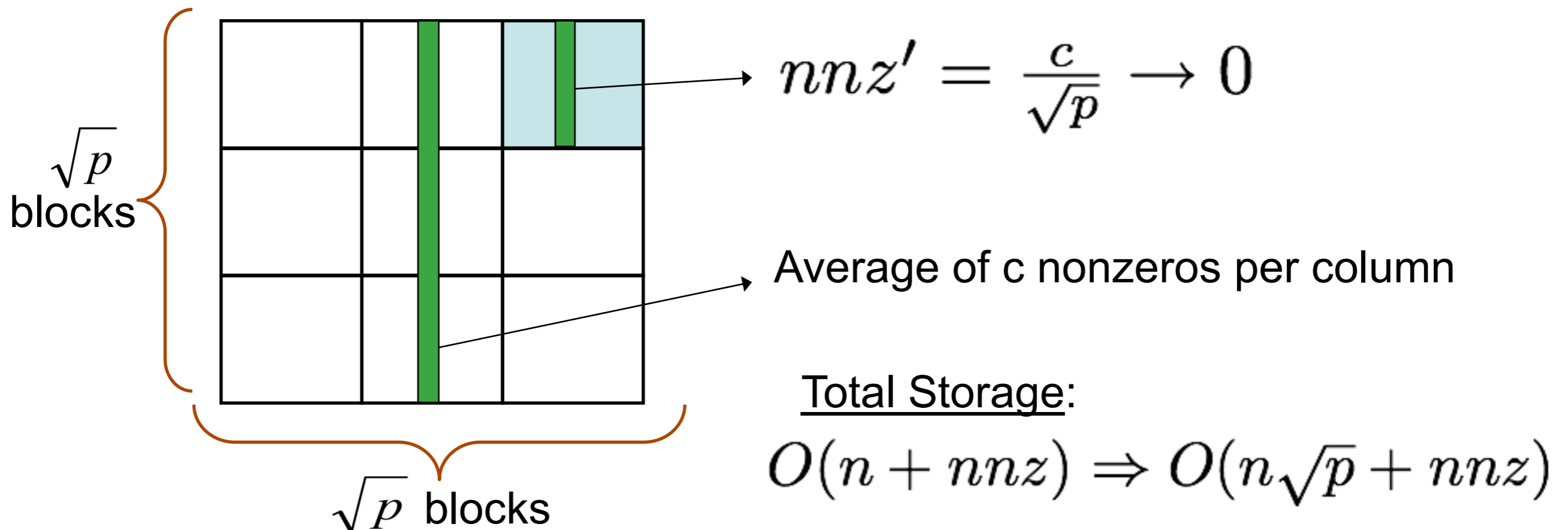
data

□	□	□	□	□	□	□	□	□	■	□	□	□	□	□	□	□
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- Stores entries in column-major order
- Dense collection of “*sparse columns*”
- Uses $O(n + nnz)$ storage.

Node Level Considerations

Submatrices are “*hypersparse*” (i.e. $nnz \ll n$)



- A data structure or algorithm that depends on the matrix dimension n (e.g. CSR or CSC) is asymptotically too wasteful for submatrices

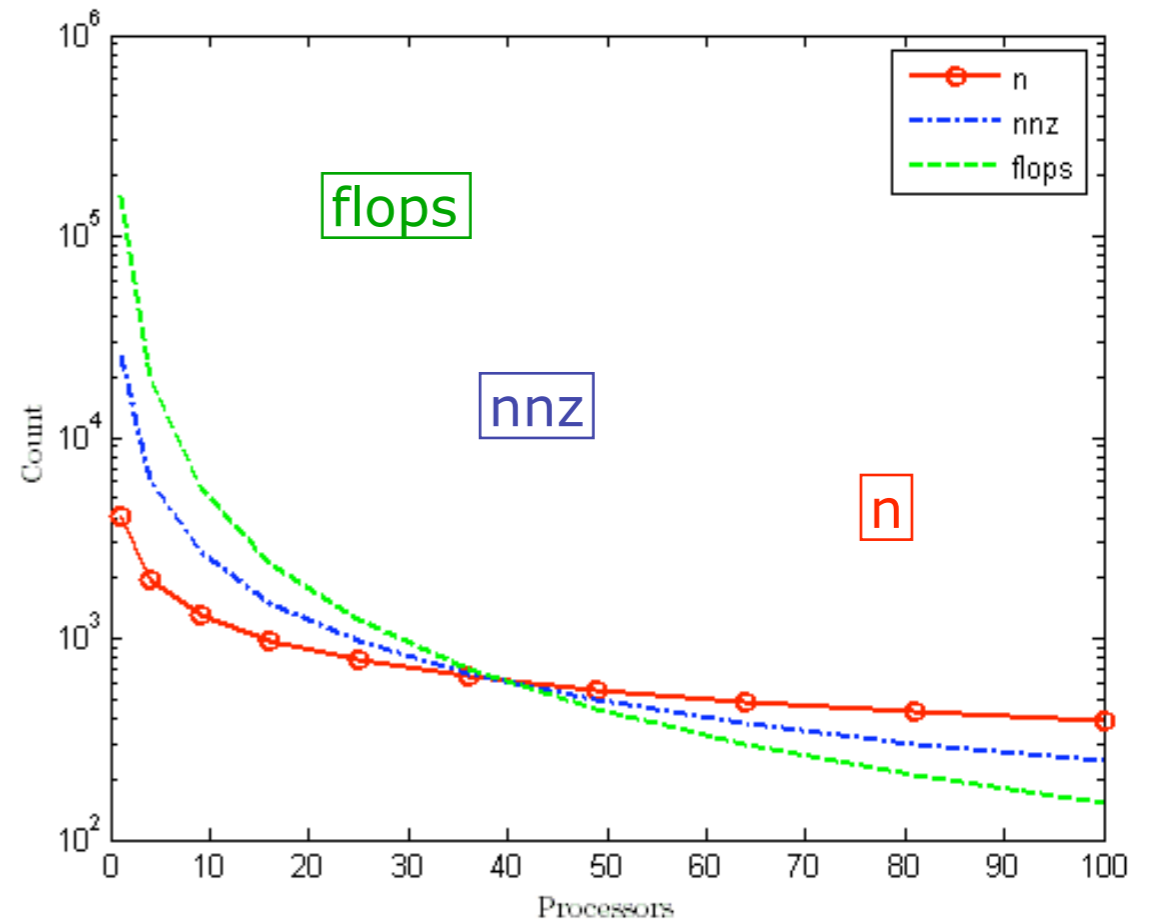
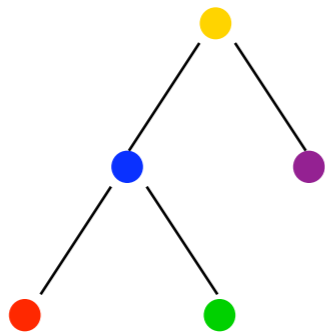
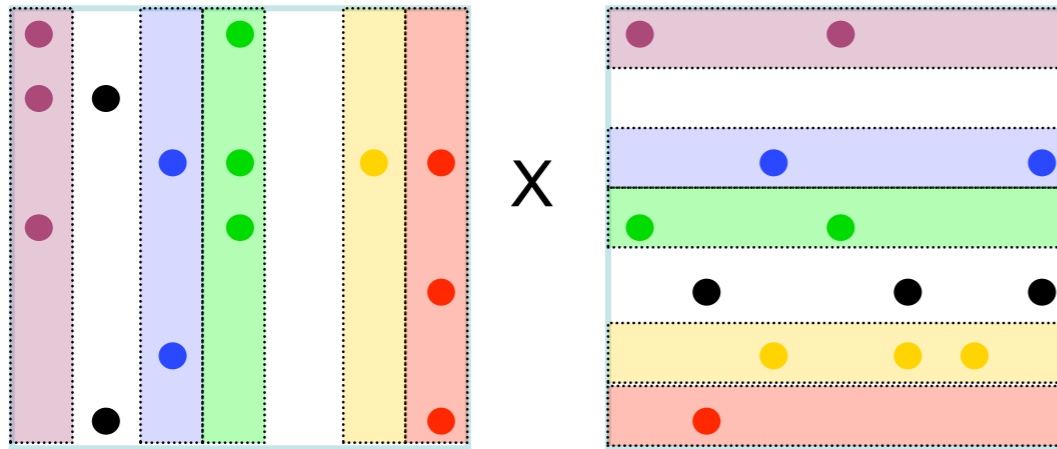
Sequential Hypersparse Kernel

Standard algorithm's complexity:

$$\Theta(\text{flops} + \text{nnz}(B) + n + m)$$

New hypersparse kernel:

$$\Theta(\text{flops} \cdot \lg ni + \text{nzc}(A) + \text{nzc}(B))$$

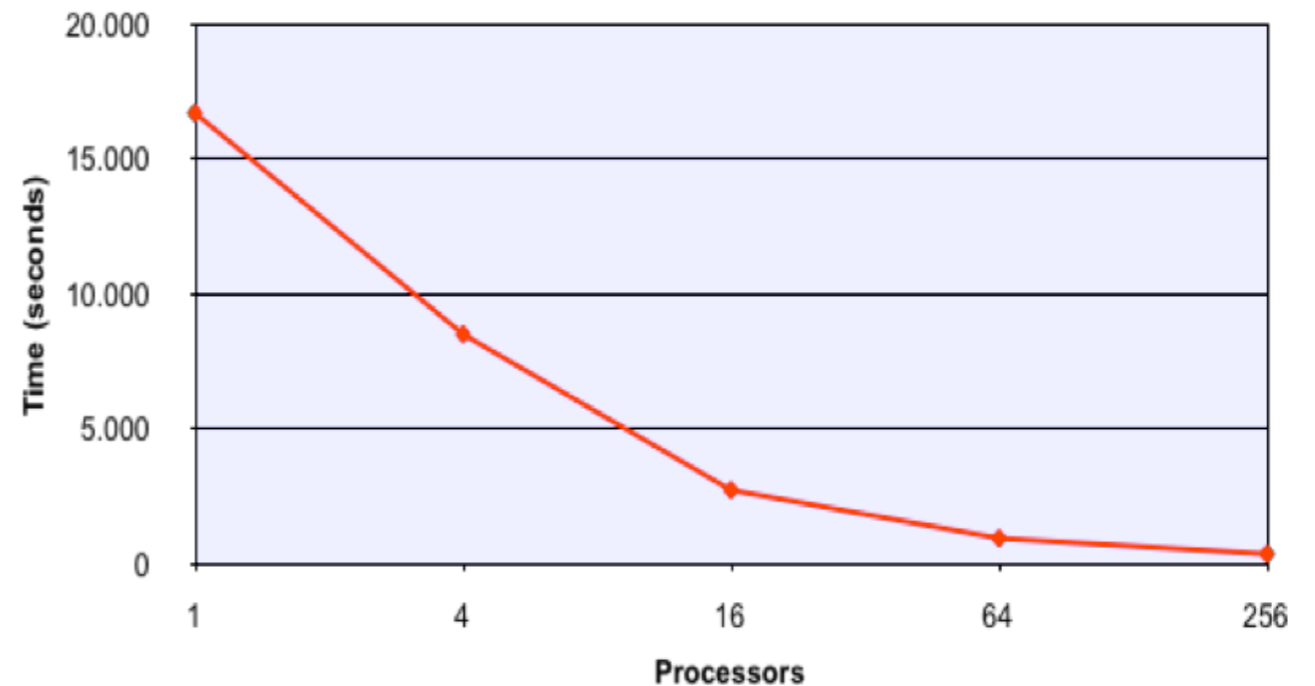


- Strictly $O(\text{nnz})$ data structure
- Outer-product formulation
- Work-efficient

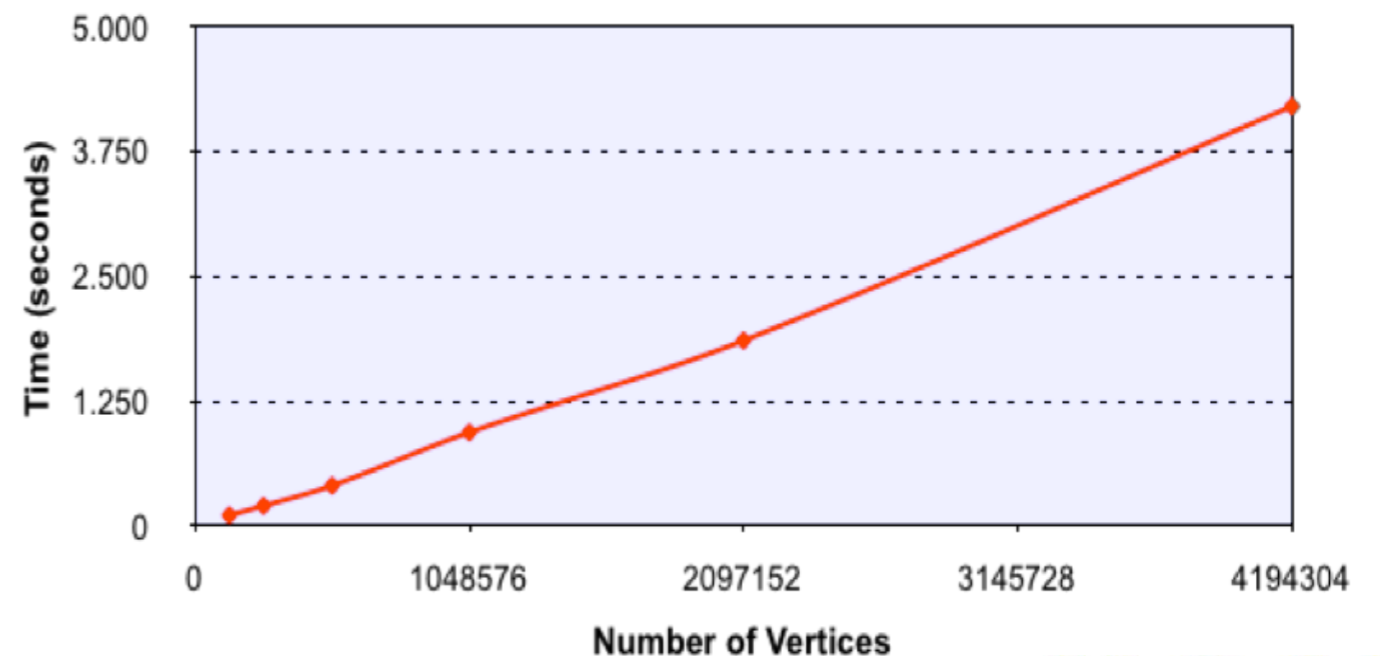
Scaling Results for SpGEMM

- RMat X RMat product (graphs with high variance on degrees)
- Random permutations useful for the overall computation.
- Bulk synchronous algorithms may still suffer due to imbalance **within the stages.**
- Asynchronous algorithm to avoid the **curse of synchronicity**
- One sided communication via RDMA (using MPI-2)
- Results obtained on TACC/Lonestar for graphs with average degree 8

Parallel PSpGEMM Scalability, Rmat-Scale20



PSpGEMM Scalability with Increasing Problem Size 64 Processors



Outline

- The Case for Primitives
- The Case for Sparse Matrices
- Parallel Sparse Matrix-Matrix Multiplication
- **Software Design of the Combinatorial BLAS**
- An Application in Social Network Analysis
- Other Work
- Future Directions

Software design of the Combinatorial BLAS

Generality, of the numeric type of matrix elements, algebraic operation performed, and the library interface.

Without the language abstraction penalty: C++ Templates

```
template <class IT, class NT, class DER>  
class SpMat;
```

- Achieve mixed precision arithmetic: Type traits
 - Enforcing interface and strong type checking: CRTP
 - General semiring operation: Function Objects
-
- Abstraction penalty is not just a programming language issue.
 - In particular, view matrices as indexed data structures and stay away from single element access (Interface should discourage)

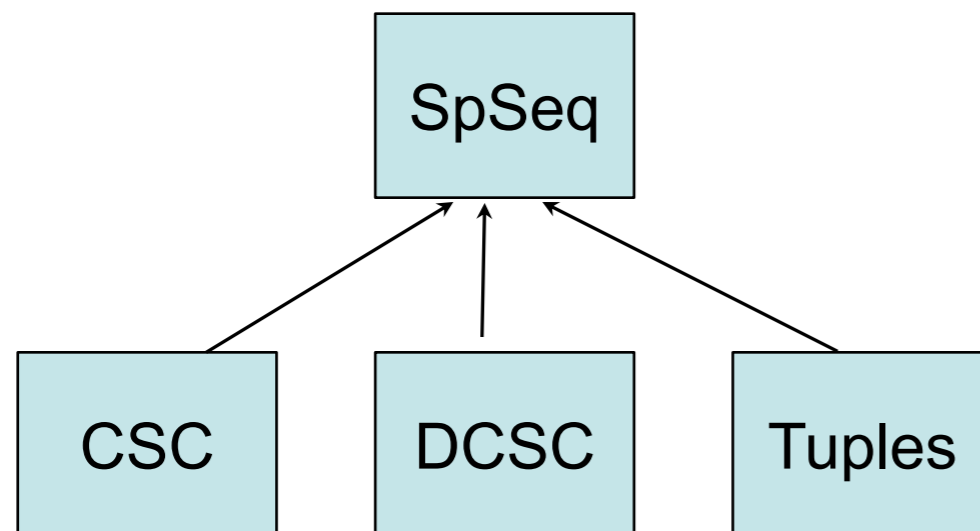
Software design of the Combinatorial BLAS

Extendability, of the library while maintaining compatibility and seamless upgrades.

➔ Decouple parallel logic from the sequential part.

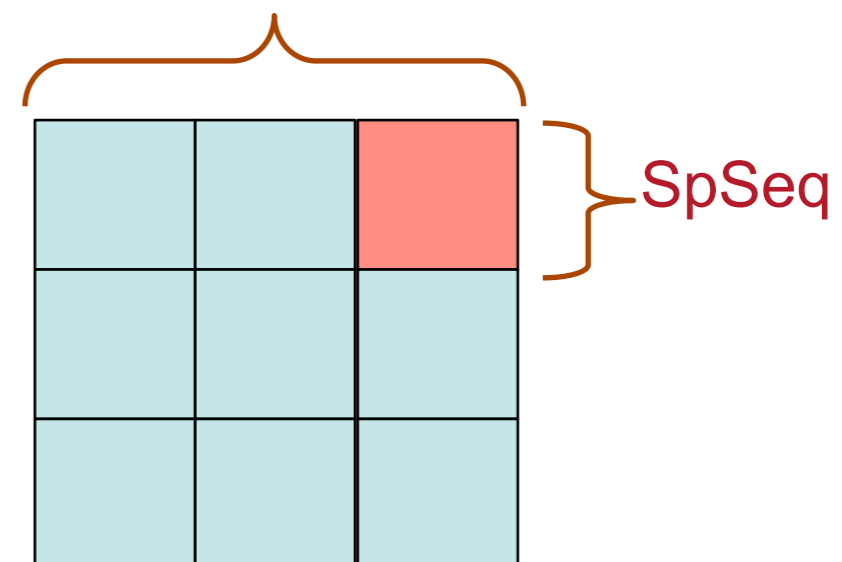
Commonalities:

- Support the sequential API
- Composed of a number of arrays



Any parallel logic:
asynchronous, bulk synchronous, etc

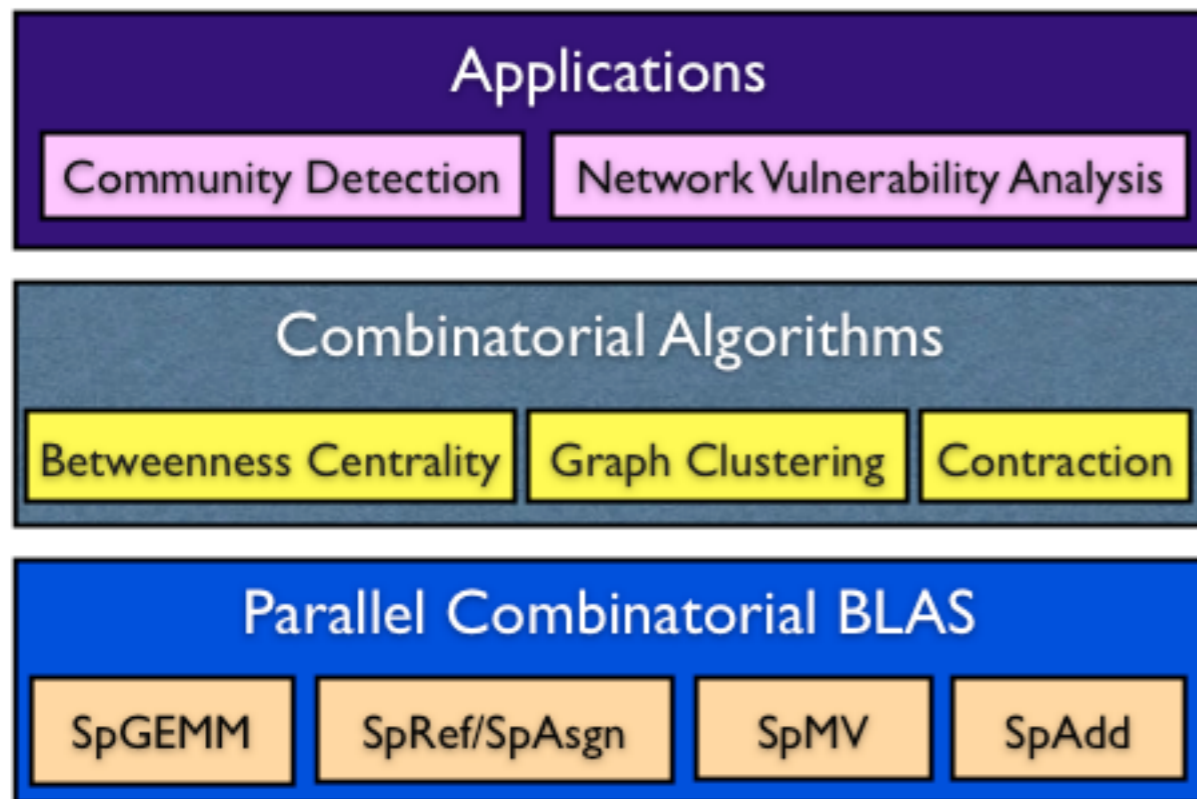
$\text{SpPar}\langle \text{Comm}, \text{SpSeq} \rangle$



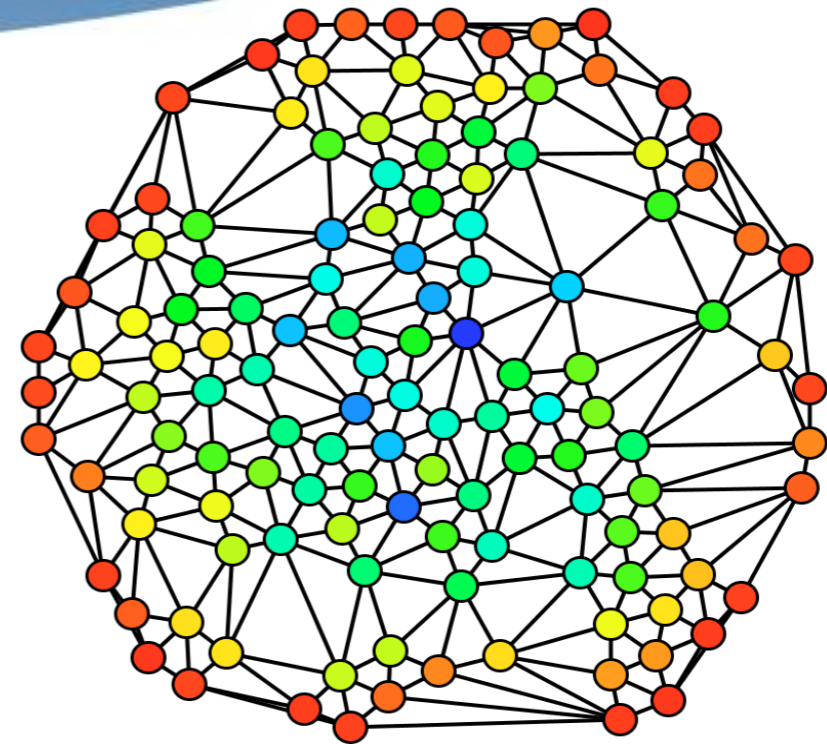
Outline

- The Case for Primitives
- The Case for Sparse Matrices
- Parallel Sparse Matrix-Matrix Multiplication
- Software Design of the Combinatorial BLAS
- **An Application in Social Network Analysis**
- Other Work
- Future Directions

Social Network Analysis



A typical software stack for an application enabled with the Combinatorial BLAS



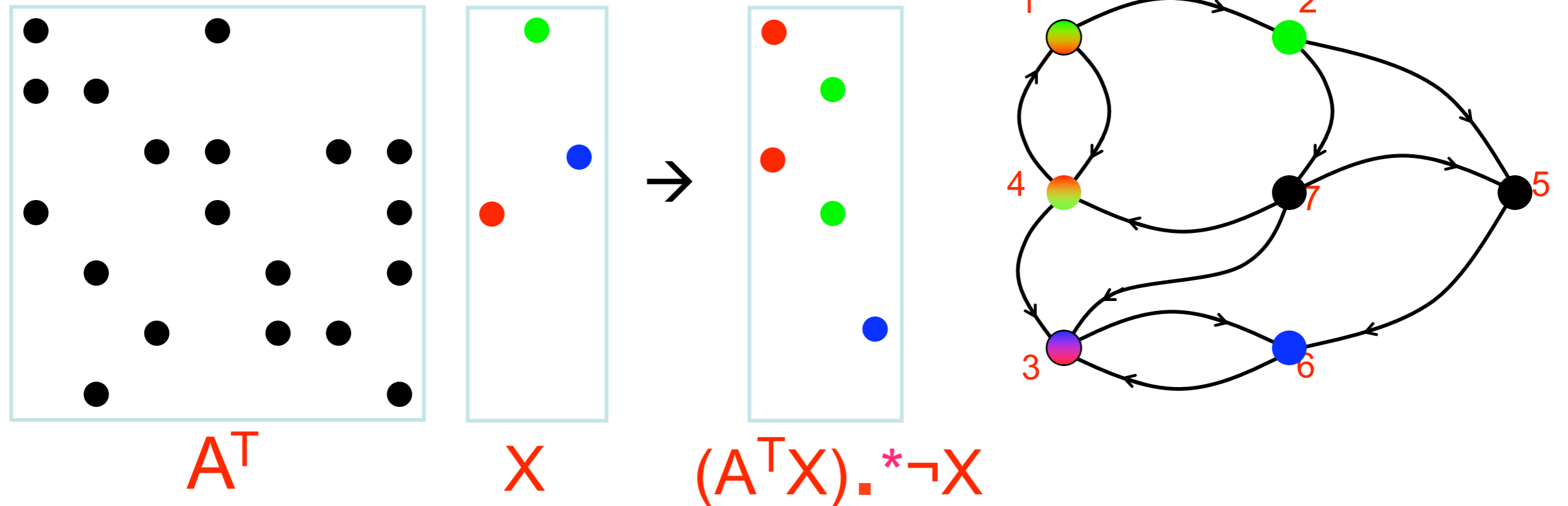
Betweenness Centrality (BC)

$C_B(v)$: Among all the shortest paths, what fraction of them pass through the node of interest?

$$C_B(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Brandes' algorithm

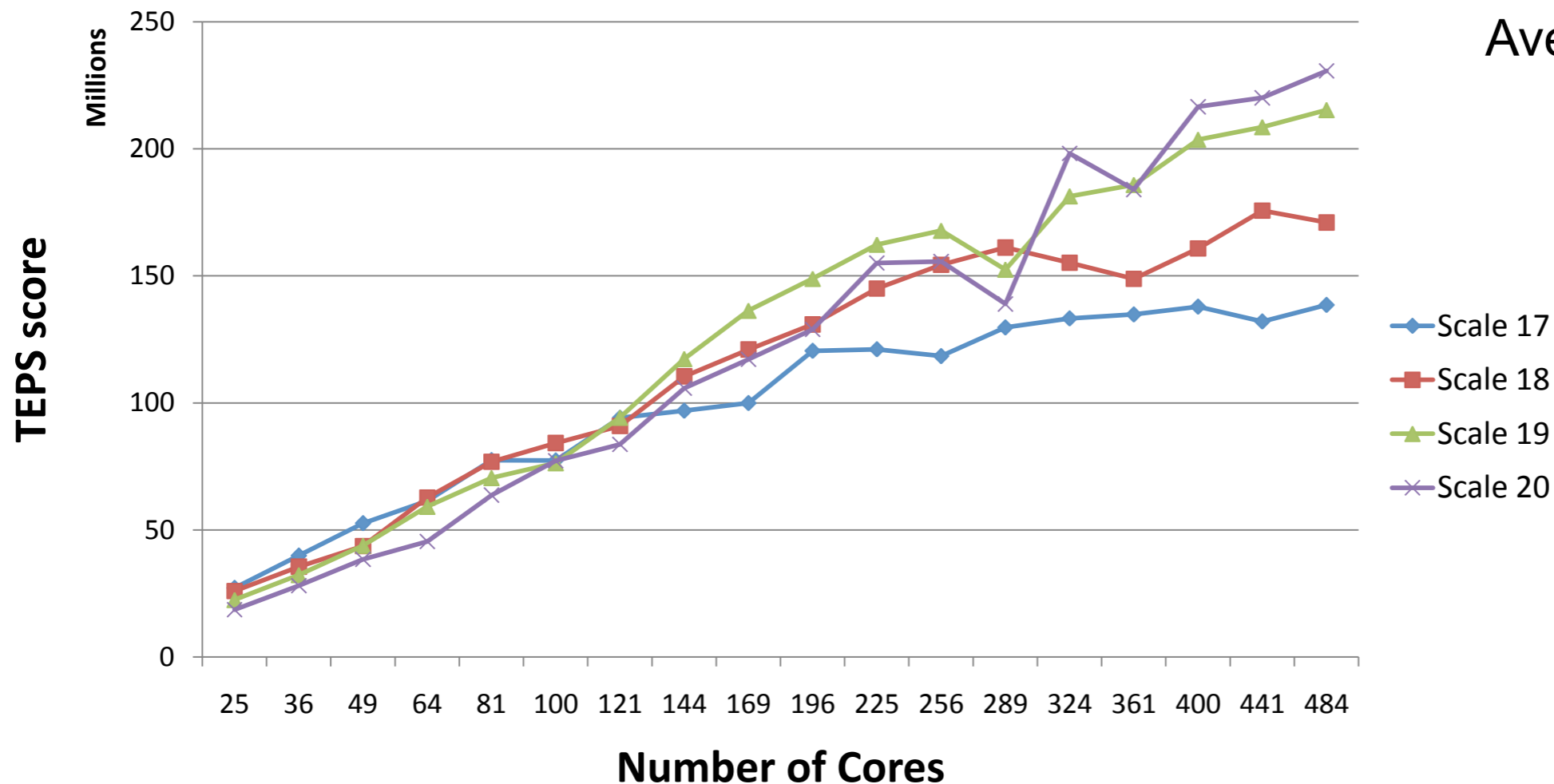
Betweenness Centrality using Sparse GEMM



- Parallel breadth-first search is implemented with sparse matrix-matrix multiplication
- Work efficient algorithm for BC

BC Performance on Distributed-memory

BC performance



Input: RMAT scale N
 2^N vertices
Average degree 8

- TEPS: Traversed Edges Per Second
- Batch of 512 vertices at each iteration
- Code only a few lines longer than Matlab version

Pure MPI-1 version.
No reliance on any particular hardware.

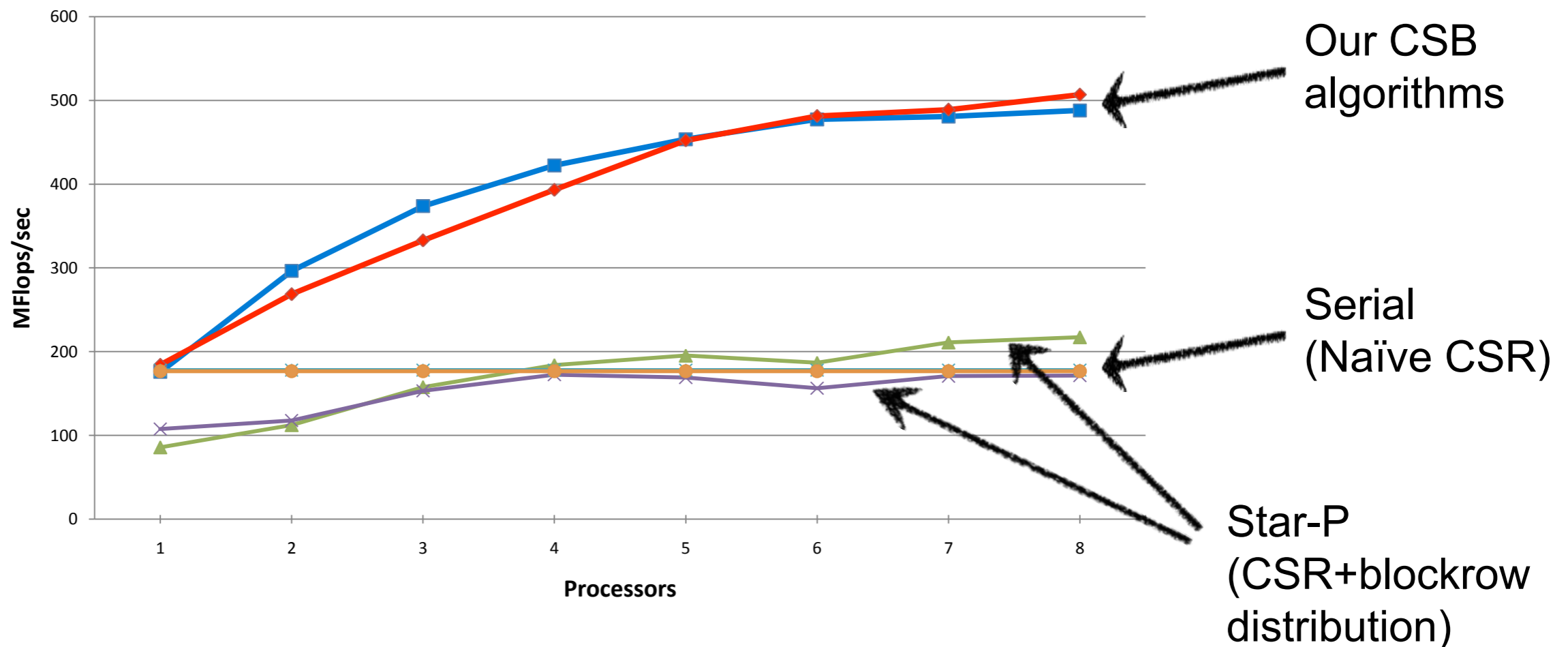
Outline

- The Case for Primitives
- The Case for Sparse Matrices
- Parallel Sparse Matrix-Matrix Multiplication
- Software Design of the Combinatorial BLAS
- An Application in Social Network Analysis
- **Other Work**
- Future Directions

SpMV on Multicore

Our parallel algorithms for $y \leftarrow Ax$ and $y' \leftarrow A^T x'$ using the new **compressed sparse blocks (CSB)** layout have

- $\Theta(\sqrt{n} \lg n)$ span, and $\Theta(nnz)$ work,
- yielding $\Theta(nnz / \sqrt{n} \lg n)$ parallelism.



Outline

- The Case for Primitives
- The Case for Sparse Matrices
- Parallel Sparse Matrix-Matrix Multiplication
- Software Design of the Combinatorial BLAS
- An Application in Social Network Analysis
- Other Work
- **Future Directions**

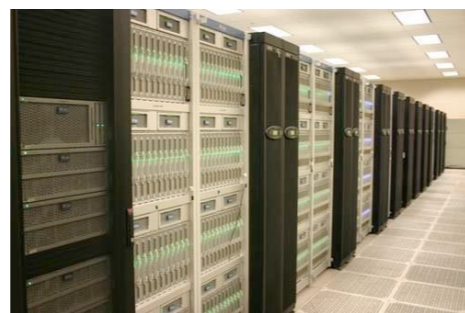
Future Directions

- ▶ Novel scalable algorithms
- ▶ Static graphs are just the beginning.
Dynamic graphs, Hypergraphs, Tensors
- ▶ Architectures (mainly nodes) are evolving
 - Heterogeneous multicores
 - Homogenous multicores with more cores per node

Hierarchical
parallelism



TACC Lonestar (2006)
4 cores / node



TACC Ranger (2008)
16 cores / node



SDSC Triton (2009)
32 cores / node



XYZ Resource (2020)

New Architectural Trends



LANL / IBM Roadrunner

Cray
XMT

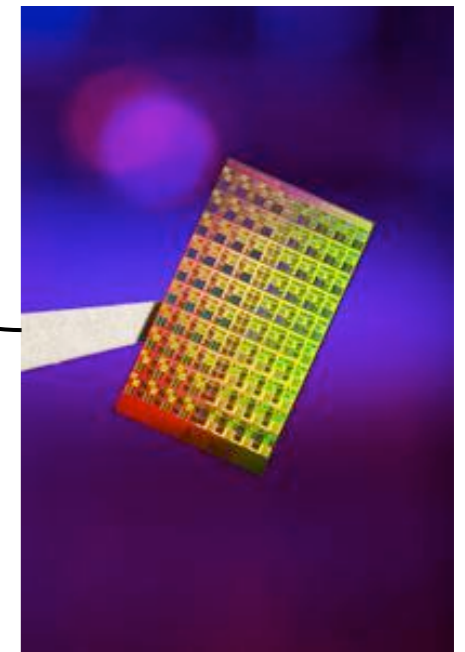


A unified
architecture ?



NVIDIA Tesla

Intel
80-core chip



Remarks

- Graph computations are pervasive in sciences and will become more so.
- High performance software libraries improve productivity.
- Carefully chosen and implemented primitive operations are key to performance.
- Linear algebraic primitives:
 - General enough to be widely useful
 - Compact enough to be implemented in a reasonable time.

Related Publications

- **Hypersparsity in 2D decomposition, sequential kernel.**
B., Gilbert, "On the Representation and Multiplication of Hypersparse Matrices", IPDPS'08
- **Parallel analysis of sparse GEMM, synchronous implementation**
B., Gilbert, "Challenges and Advances in Parallel Sparse Matrix-Matrix Multiplication, ICPP'08
- **The case for primitives, APSP on the GPU**
B., Gilbert, Budak, "Solving Path Problems on the GPU", Parallel Computing, 2009
- **SpMV on Multicores**
B., Fineman, Frigo, Gilbert, Leiserson, "Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication using Compressed Sparse Blocks", SPAA'09
- **Betweenness centrality results**
B., Gilbert, "Parallel Sparse Matrix-Matrix Multiplication and Large Scale Applications"
- **Software design of the library**
B., Gilbert, "Parallel Combinatorial BLAS: Interface and Reference Implementation"

Acknowledgments...

David Bader, Erik Boman, Ceren Budak, Alan Edelman, Jeremy Fineman, Matteo Frigo, Bruce Hendrickson, Jeremy Kepner, Charles Leiserson, Kamesh Madduri, Steve Reinhardt, Eric Robinson, Viral Shah, Sivan Toledo