Reducing Communication in Parallel Graph Computations

Aydın Buluç
Berkeley Lab

August 8, 2013
ASCR Applied Math PI Meeting
Albuquerque, NM

Acknowledgments (past & present)

- Krste Asanovic (UC Berkeley)
- Grey Ballard (UC Berkeley)
- Scott Beamer (UC Berkeley)
- Jim Demmel (UC Berkeley)
- Erika Duriakova (UC Dublin)
- Armando Fox (UC Berkeley)
- John Gilbert (UCSB)
- Laura Grigori (INRIA)
- Shoaib Kamil (MIT)
- Ben Lipshitz (UC Berkeley)
- Adam Lugowski (UCSB)
- Lenny Oliker (Berkeley Lab)
- Lenny Oliker (Berkeley Lab)
- Dave Patterson (UC Berkeley)
- Steve Reinhardt (YarcData)
- Oded Schwartz (UC Berkeley)
- Edgar Solomonik (UC Berkeley)
- Sivan Toledo (Tel Aviv Univ)
- Sam Williams (Berkeley Lab)

This work is funded by:

U.S. DEPARTMENT OF ENERGY | Office of Science

Informal Outline

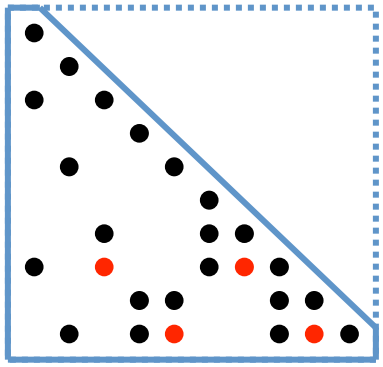
- Large-scale **graphs**
- **Communication** costs of parallel graph computations
- Communication takes **energy**
- **Linear algebraic primitives** enable scalable graph computation
- **Semiring** abstraction glues graphs to linear algebra
- **Direction-optimizing BFS** does >6X less communication
- New communication-optimal **sparse matrix-matrix multiply**
- New communication-optimal **dense all-pairs shortest-paths**

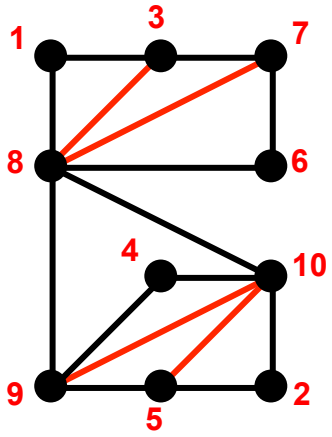
Graph abstractions in Computer Science

Compiler optimization:
Control flow graph : graph dominators
Register allocation: graph coloring

Scientific computing:
Preconditioning: support graphs, spanning trees
Sparse direct solvers: chordal graphs
Parallel computing: graph separators

Computer Networks:
Routing: shortest path algorithms
Web crawling: graph traversal
Interconnect design: Cayley graphs



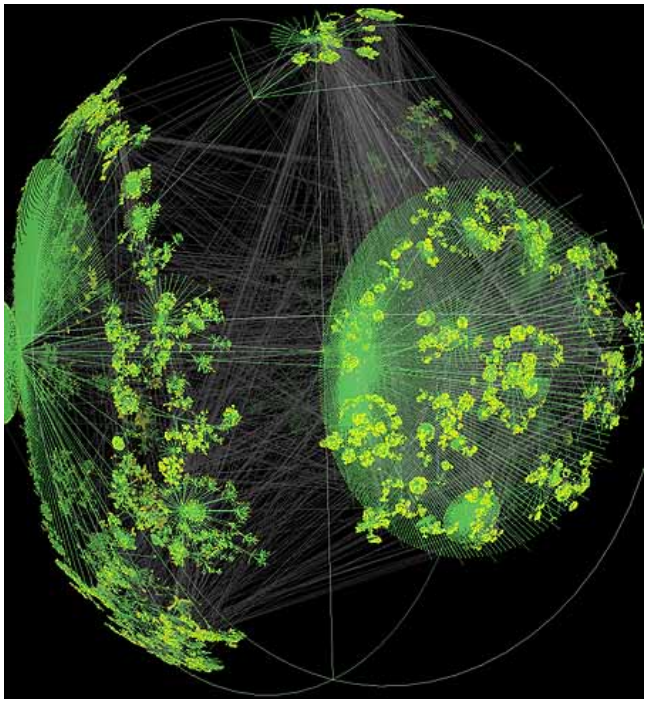


$G^+(A)$
[chordal]

Large graphs are everywhere

Internet structure

Social interactions



WWW snapshot, courtesy Y. Hyun

Scientific datasets: biological, chemical, cosmological, ecological, …

Yeast protein interaction network, courtesy H. Jeong

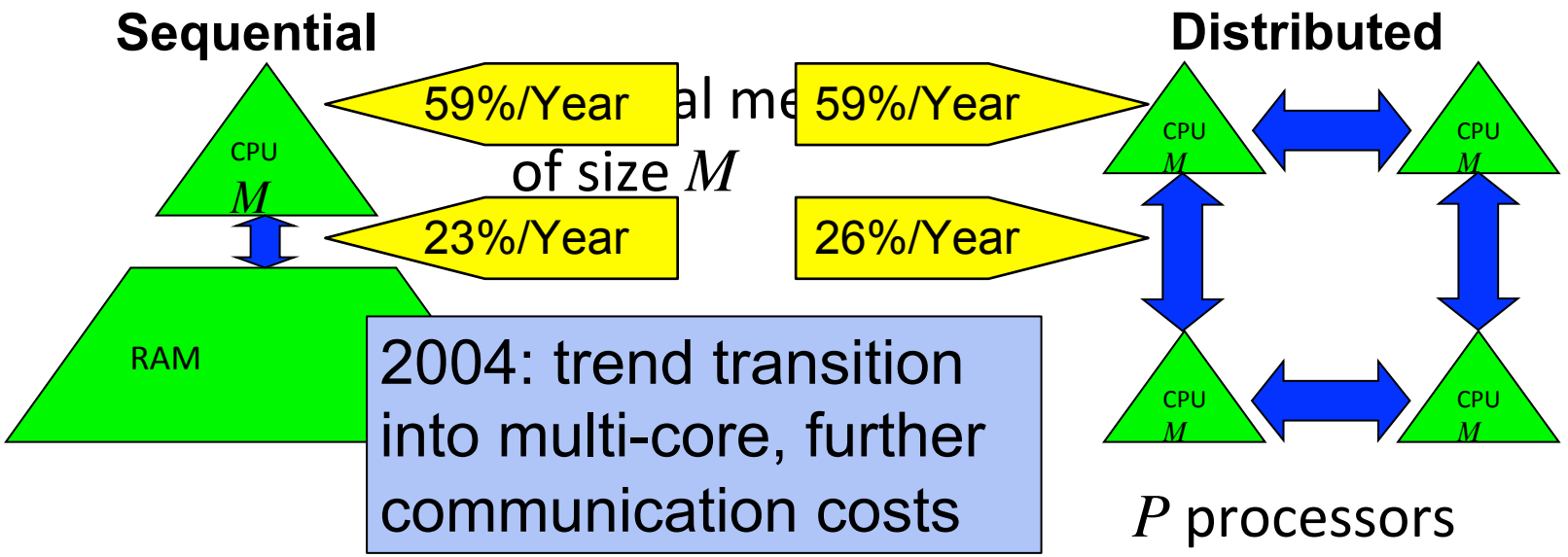
Model and Motivation

Two kinds of costs:

- Arithmetic (FLOPs)
- Communication: moving data

Develop faster algorithms: minimize communication (to lower bound if possible)

$$Running\ time = \gamma \cdot \#FLOPs + \beta \cdot \#Words + (\alpha \cdot \#Messages)$$



2004: trend transition into multi-core, further communication costs

Communication -> Energy

Communication-avoiding algorithms: **Save time**

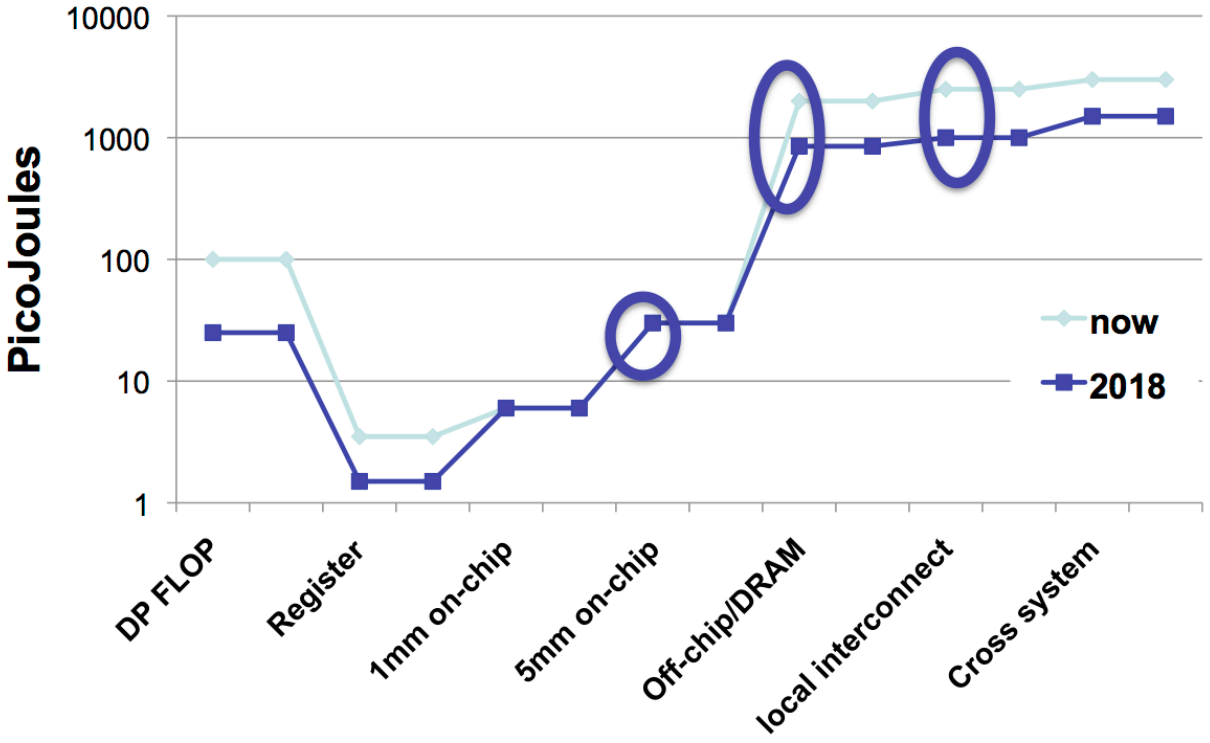
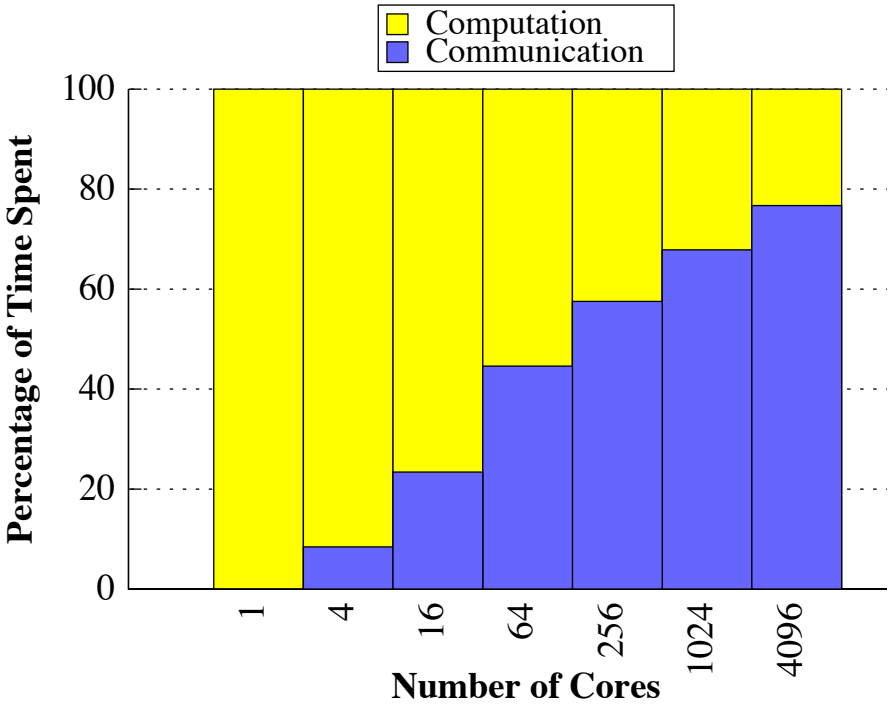


Image courtesy of John Shalf (LBNL)

Communication-avoiding algorithms: **Save energy**

Communication crucial for graphs

- Often no surface to volume ratio.
- Very little data reuse in existing algorithmic formulations *
- Already heavily communication bound



2D sparse matrix-matrix multiply emulating:

- Graph contraction
- AMG restriction operations

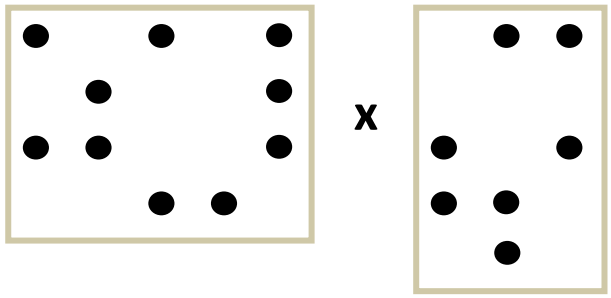
Scale 23 R-MAT (scale-free graph) **times** order 4 restriction operator

Cray XT4, Franklin, NERSC

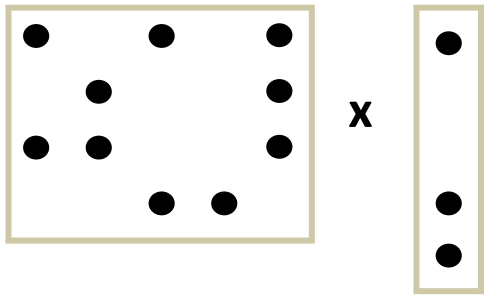
B., Gilbert. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. *SIAM Journal of Scientific Computing*, 2012

Linear-algebraic primitives

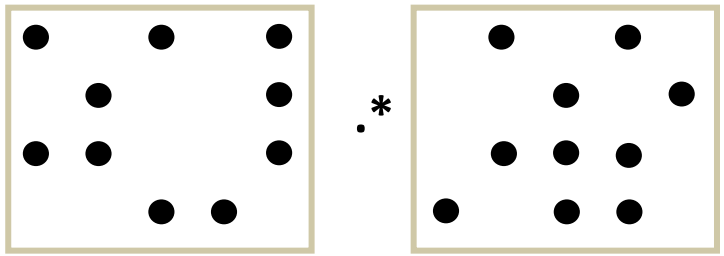
Sparse matrix-sparse matrix multiplication (Sparse GEMM)



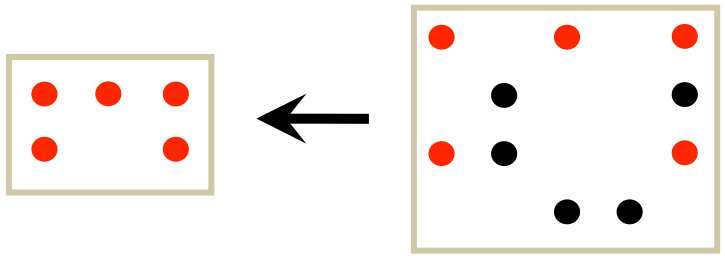
Sparse matrix-sparse vector multiplication (SpMV)



Element-wise operations



Sparse matrix indexing



The Combinatorial BLAS implements these, and more, on arbitrary semirings, e.g. (×, +), (and, or), (+, min)

The case for sparse matrices

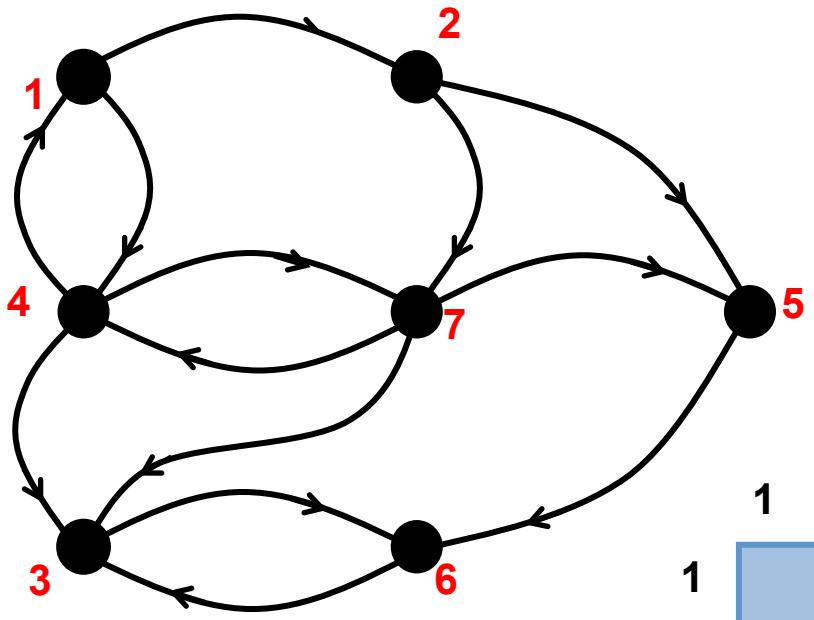
Many irregular applications contain coarse-grained parallelism that can be exploited by abstractions at the proper level.

Traditional graph computations
Data driven, unpredictable communication.
Irregular and unstructured, poor locality of reference
Fine grained data accesses, dominated by latency

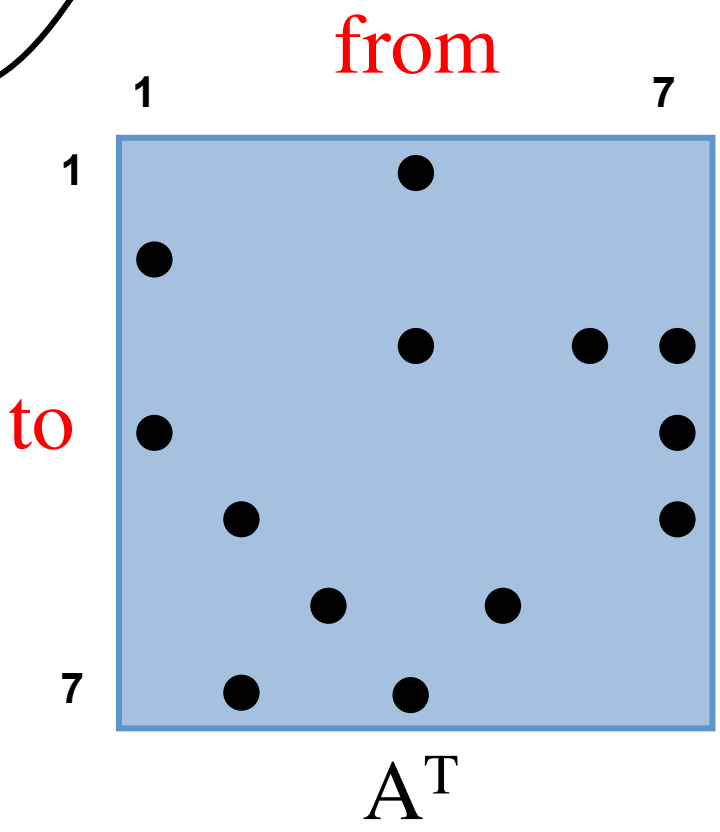
The case for sparse matrices

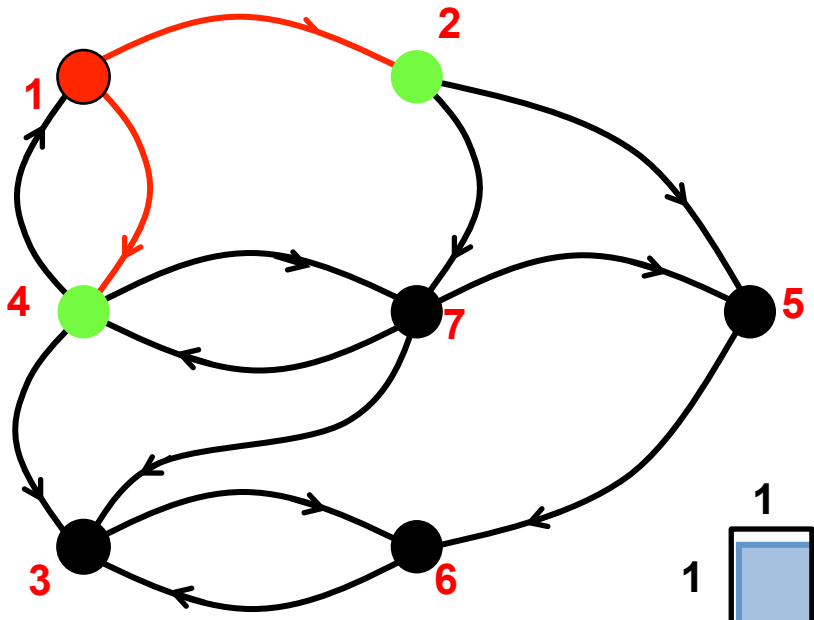
Many irregular applications contain coarse-grained parallelism that can be exploited by abstractions at the proper level.

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 exploit memory hierarchy
Fine grained data accesses, dominated by latency	Coarse grained parallelism, bandwidth limited



Breadth-first search in the language of linear algebra

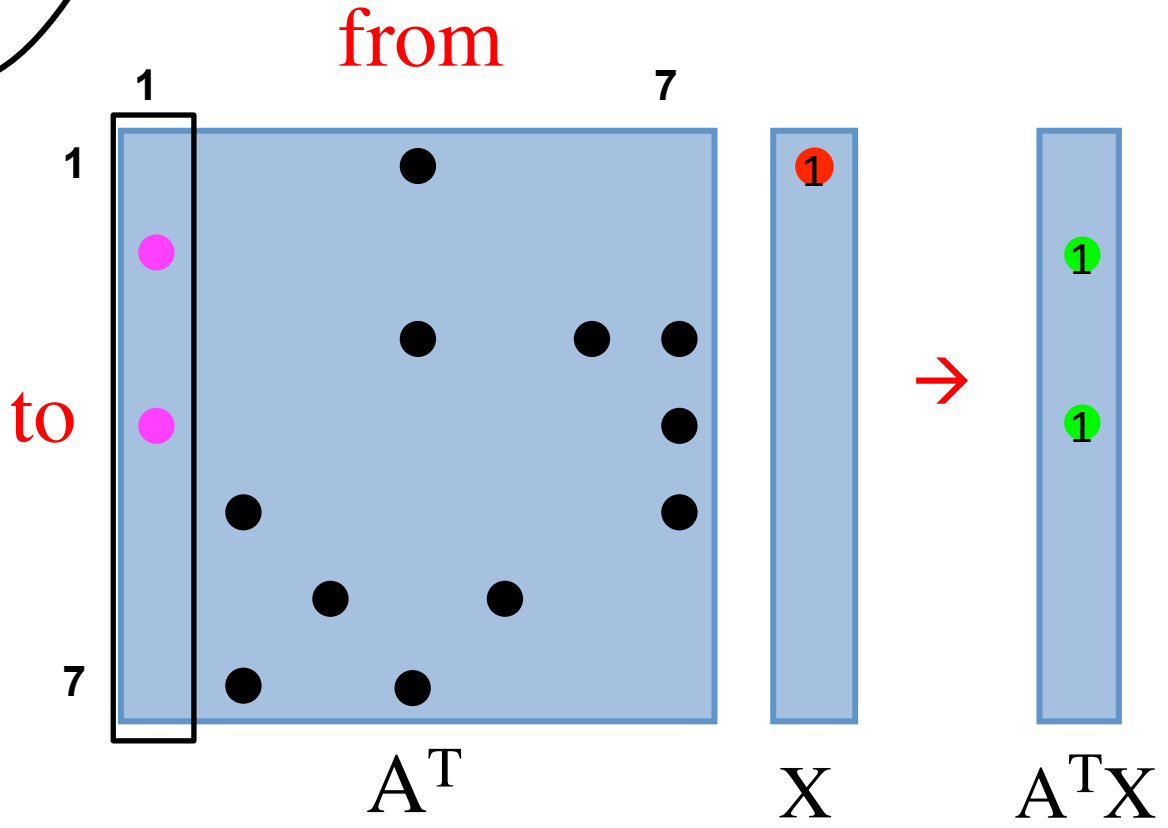




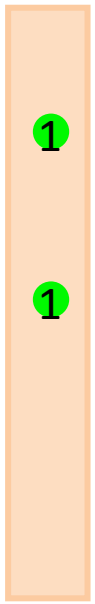
Particular semiring operations:

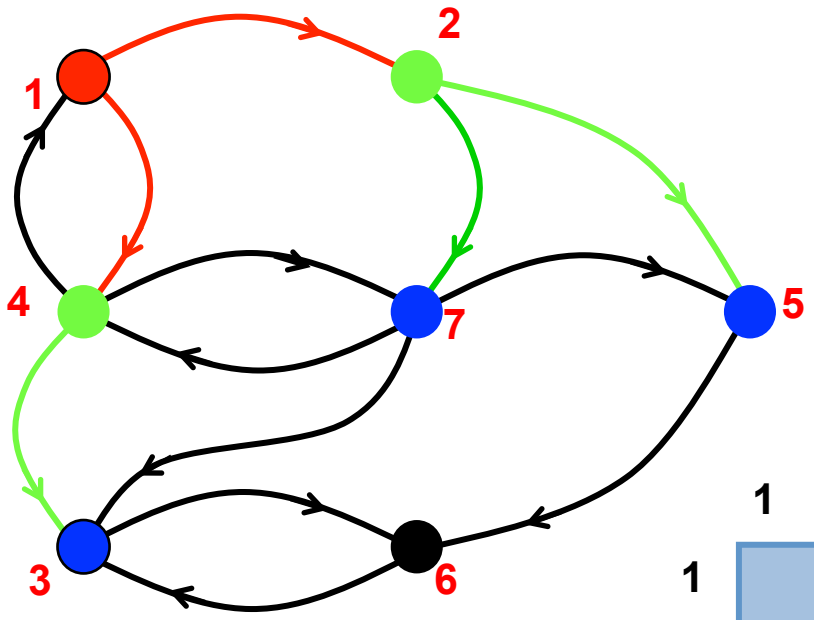
Multiply: select

Add: minimum



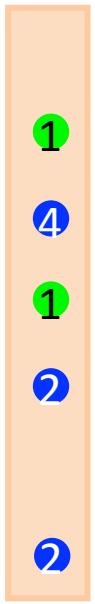
parents:

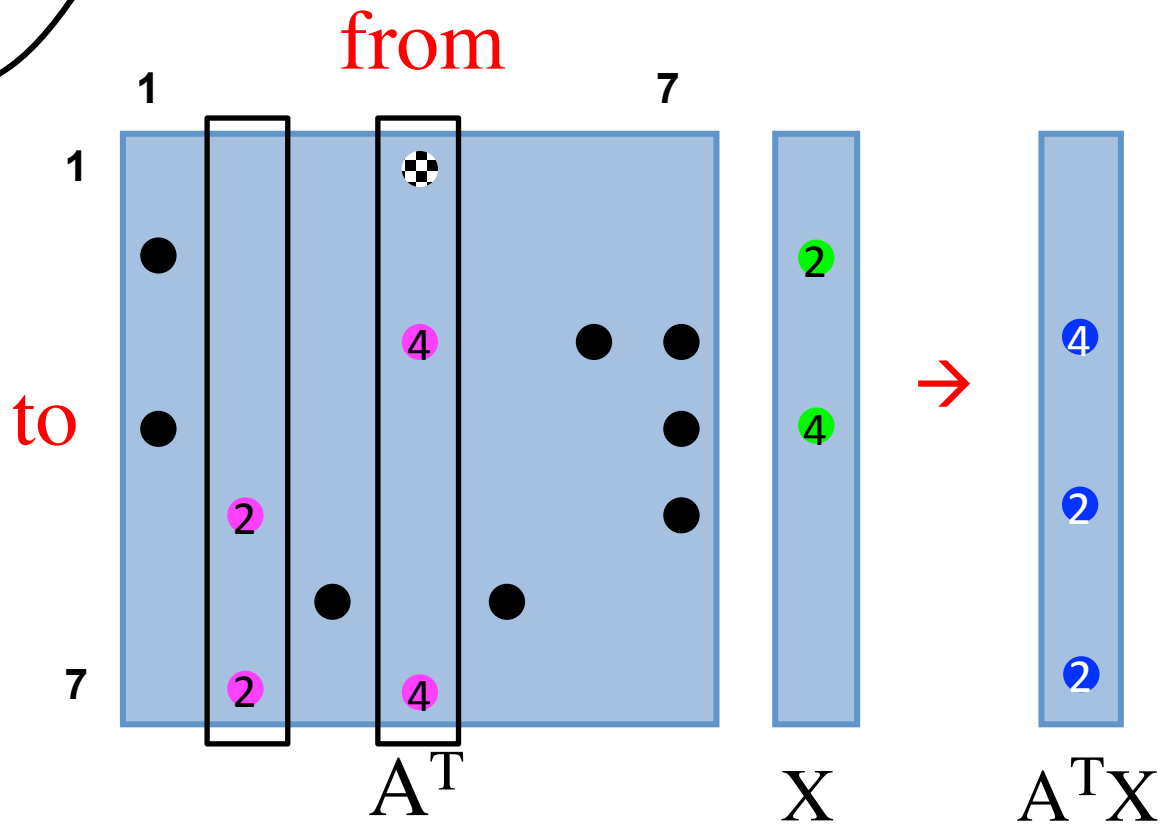


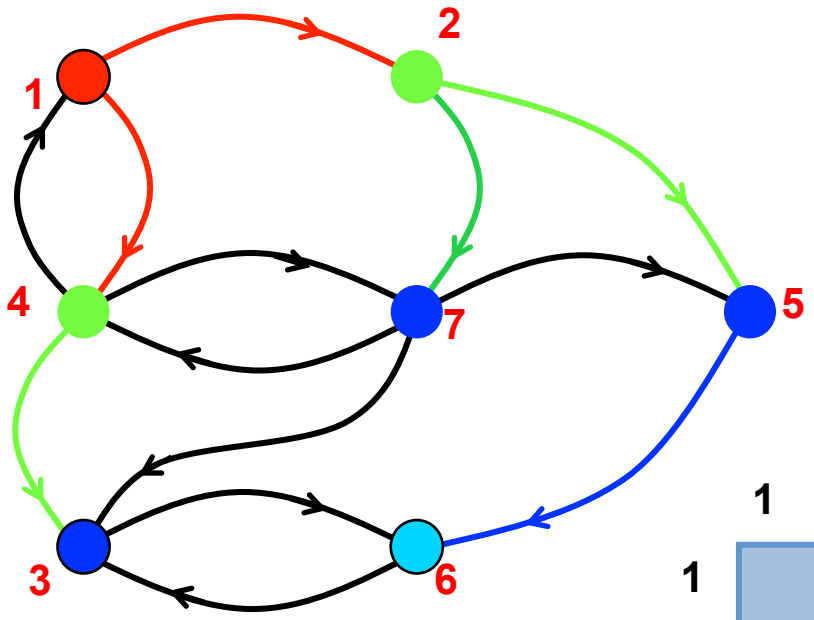


Multiple traverses outgoing edges
Add chooses among incoming edges

parents:

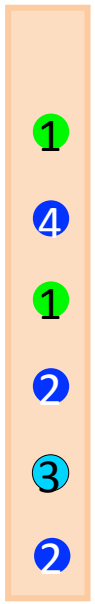


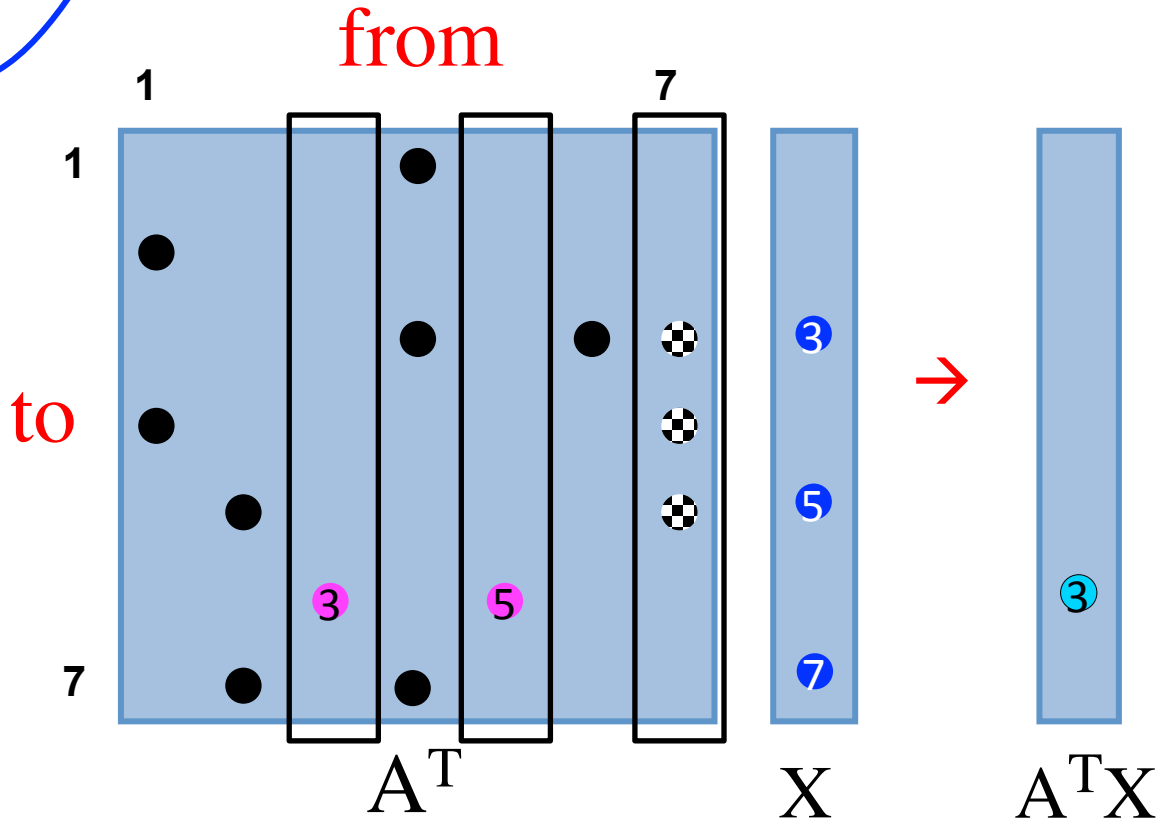


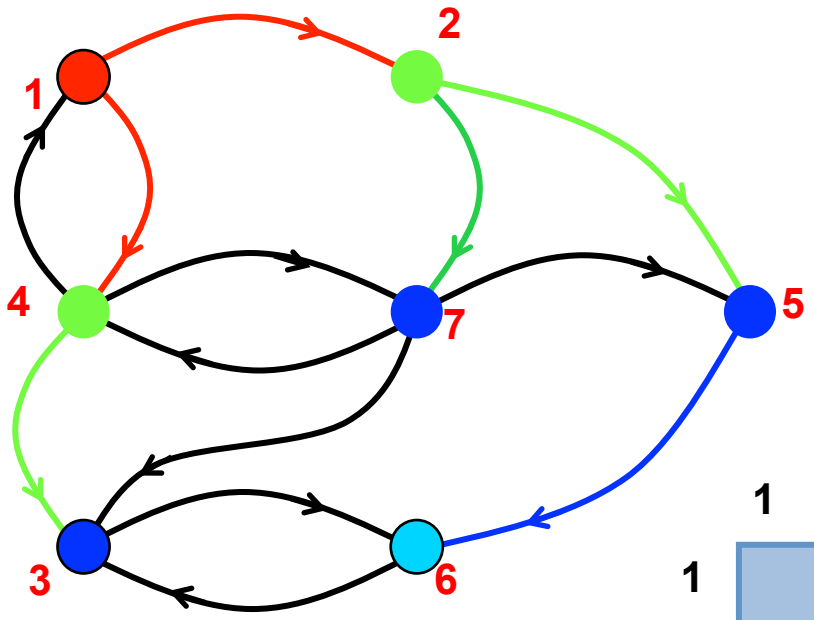


Select vertex with
minimum label as parent

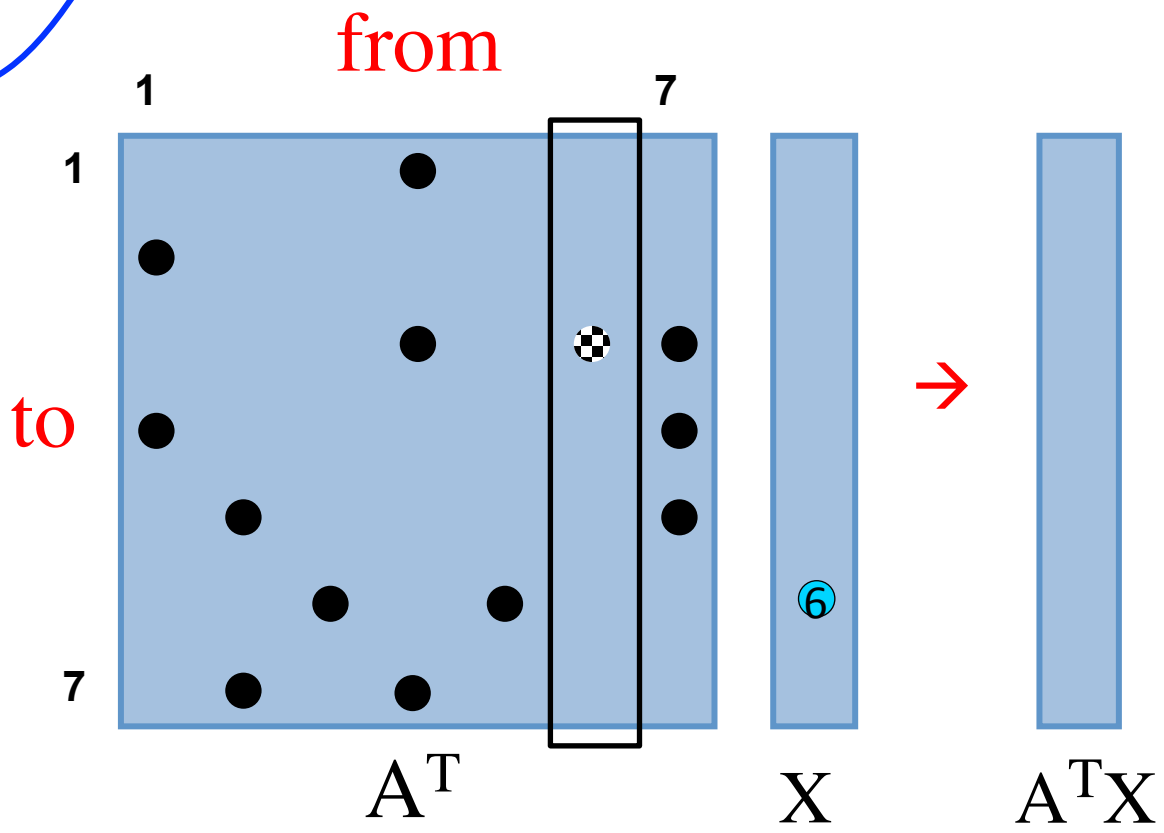
parents:





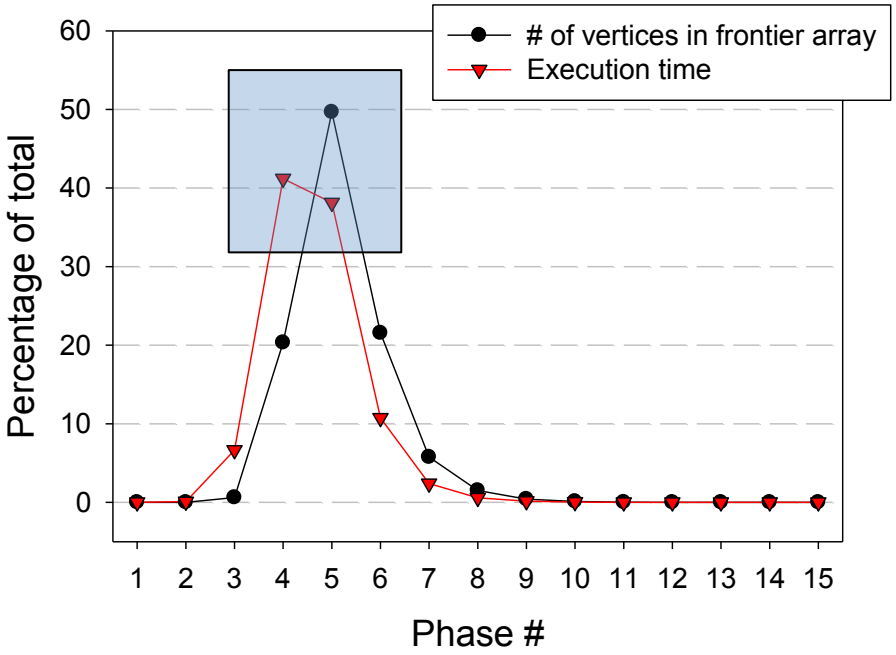


Result: Deterministic breadth-first search



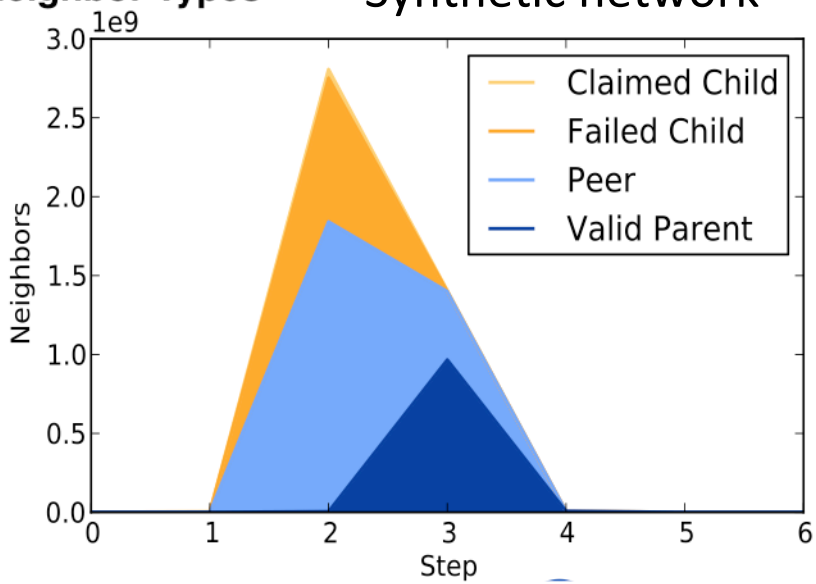
Performance observations of level-synchronous BFS

Youtube social network

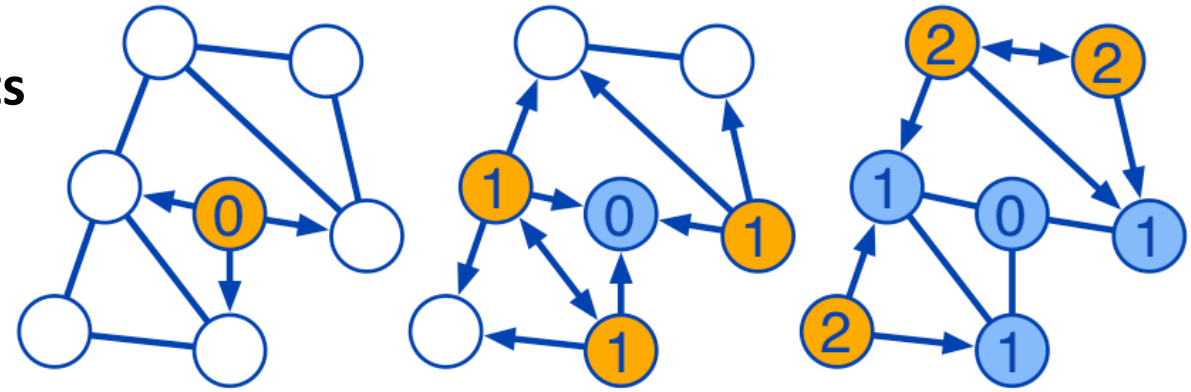


Neighbor Types

Synthetic network



When the frontier is at its peak, almost all edge examinations "fail" to claim a child

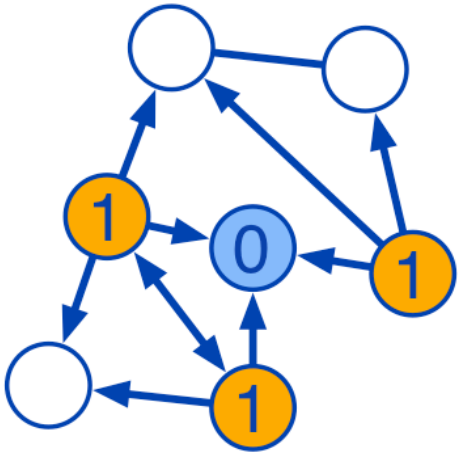


Bottom-up BFS algorithm

[Single-node: Beamer et al. 2012]

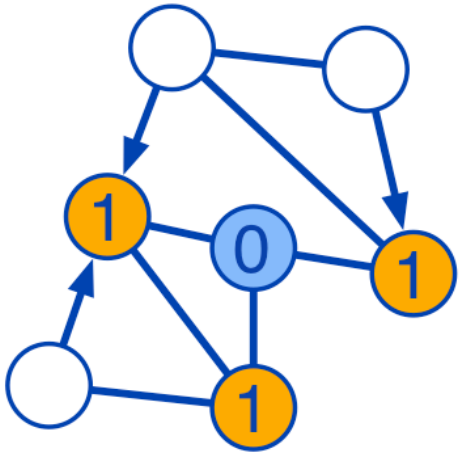
Classical (top-down) algorithm is optimal in worst case, but pessimistic for low-diameter graphs (previous slide).

Top-Down



for all *v* in frontier
attempt to parent ***all*** neighbors(*v*)

Bottom-Up

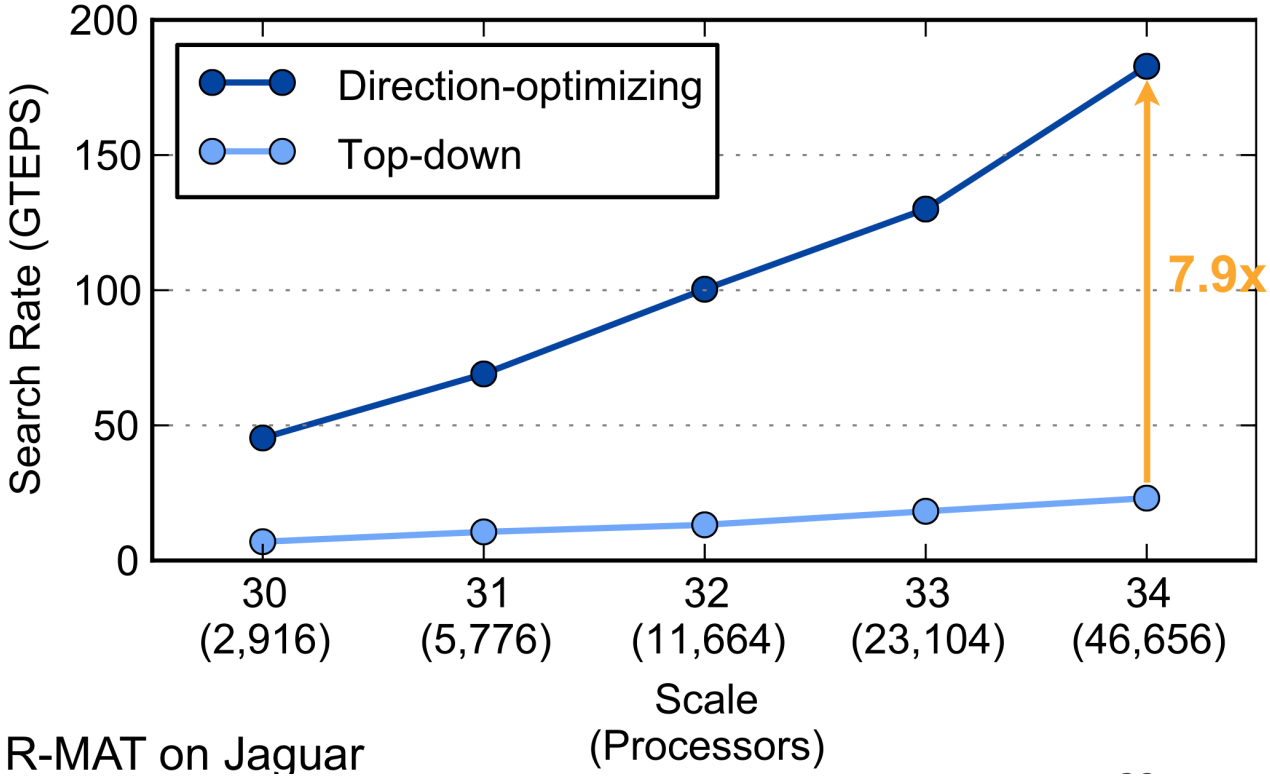


for all *v* in unvisited
find ***any*** parent
(neighbor(*v*) in frontier)

Direction-optimizing approach:

- Switch from top-down to bottom-up search
- When the majority of the vertices are discovered.

Direction-optimizing BFS on distributed memory

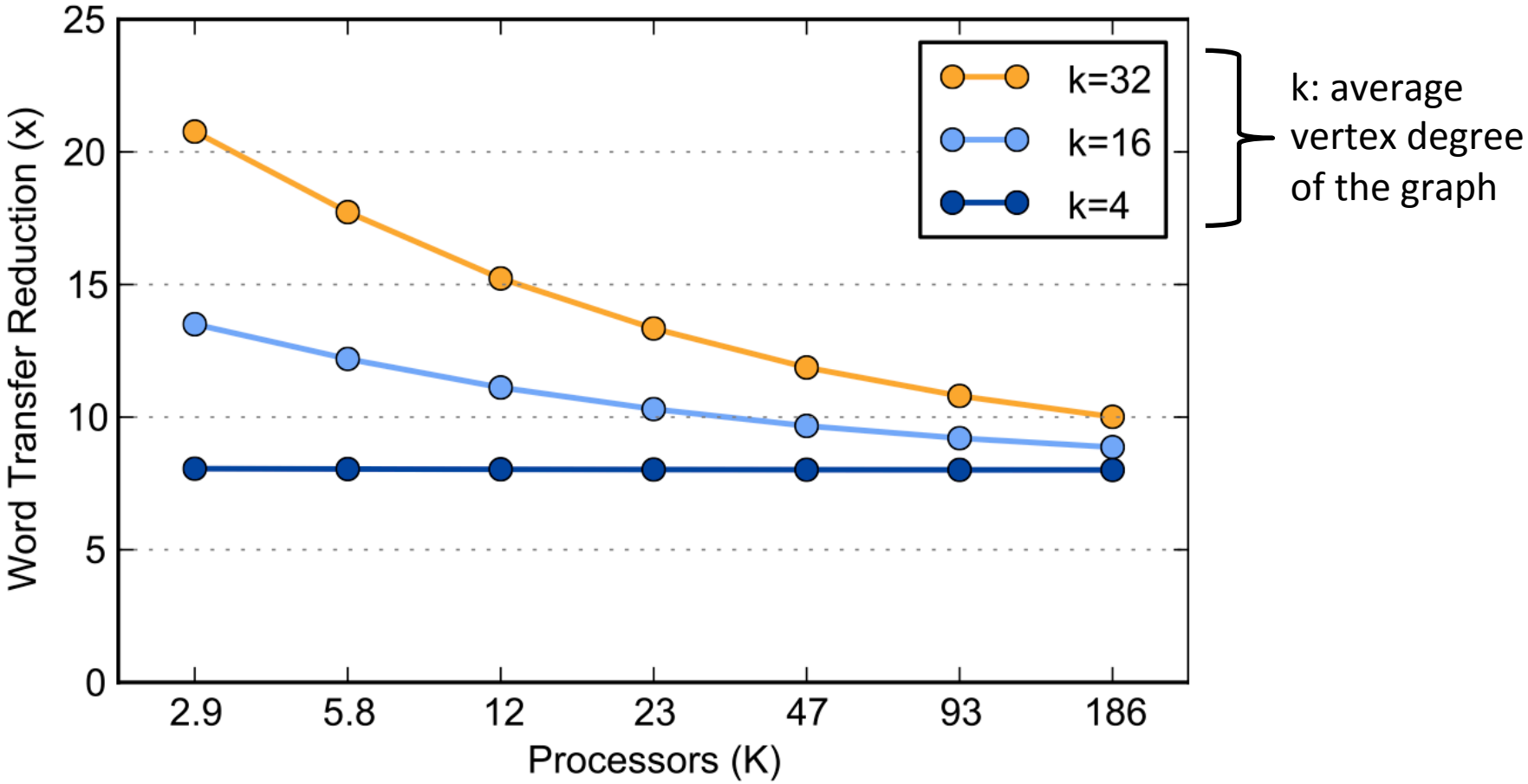


- Kronecker (Graph500): 16 billion vertices and 256 billion edges.
- Implemented on top of **Combinatorial BLAS** (i.e. uses 2D decomposition)

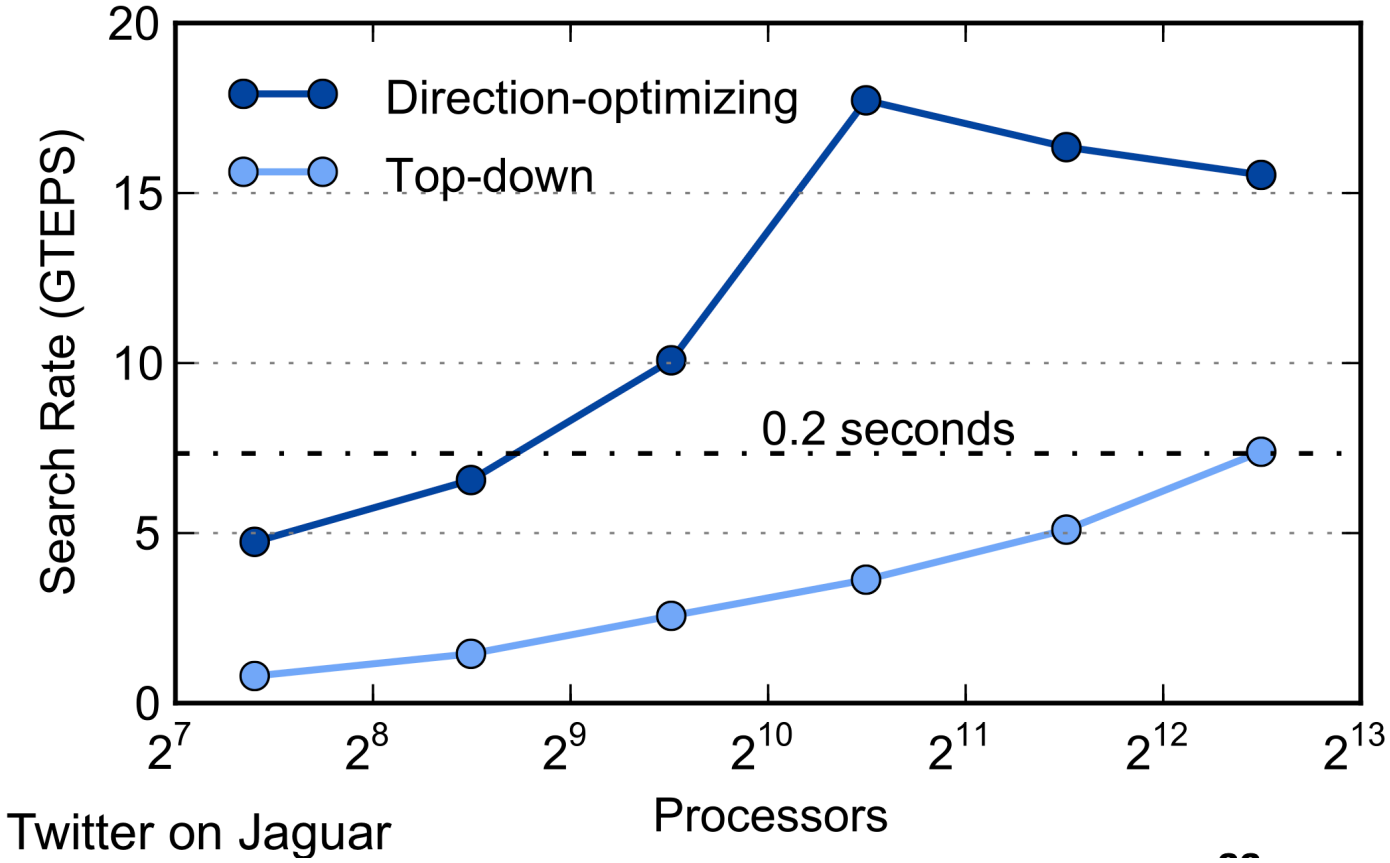
Beamer, **B.**, Asanović, and Patterson, "Distributed Memory Breadth-First Search Revisited: Enabling Bottom-Up Search", *MTAAP'13, in conjunction with IPDPS*

Direction-optimizing BFS reduces communication

Communication volume reduction based on analytical model derived in the paper, assuming 4 bottom-up steps (typical for power-law graphs)



Direction-optimizing BFS saves energy



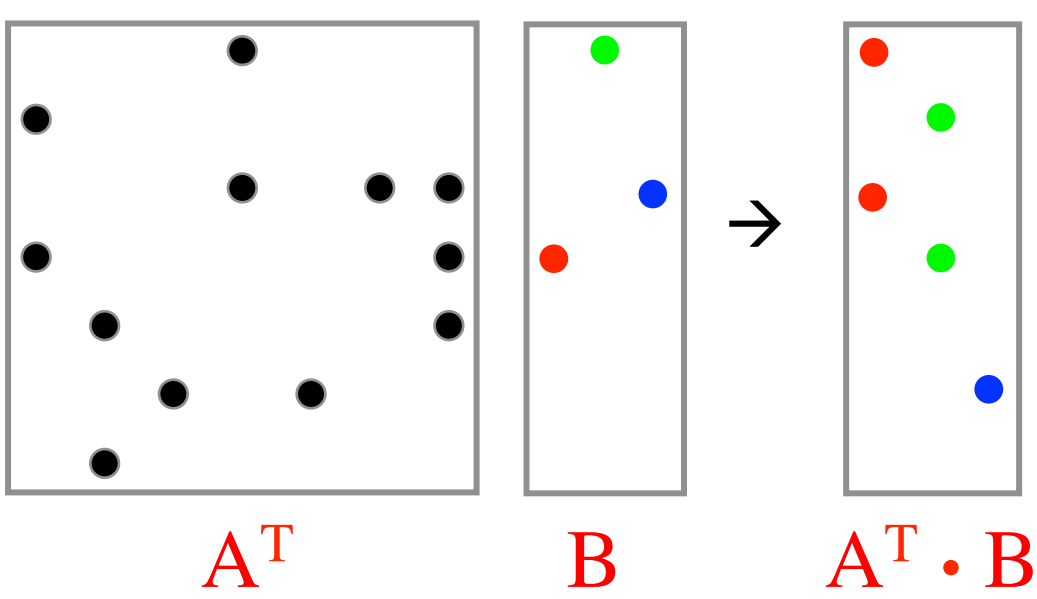
For a fixed-sized real input, direction-optimizing algorithm completes at the same time using **1/16th of processors (and energy)**

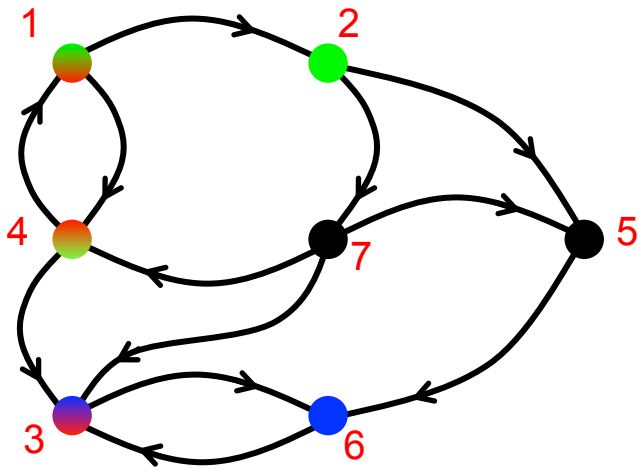
Betweenness Centrality using Sparse GEMM

Betweenness Centrality is a measure of influence.

$C_B(v)$: Among all the shortest paths, what fraction passes through v?

- Parallel breadth-first search is implemented with sparse GEMM
- Work-efficient, highly-parallel implementation of Brandes' algorithm





Sparse Matrix-Matrix Multiplication

Why sparse matrix-matrix multiplication?

- Used for algebraic multigrid, graph clustering, betweenness centrality, graph contraction, subgraph extraction, cycle detection, quantum chemistry,…

How do dense and sparse GEMM compare?

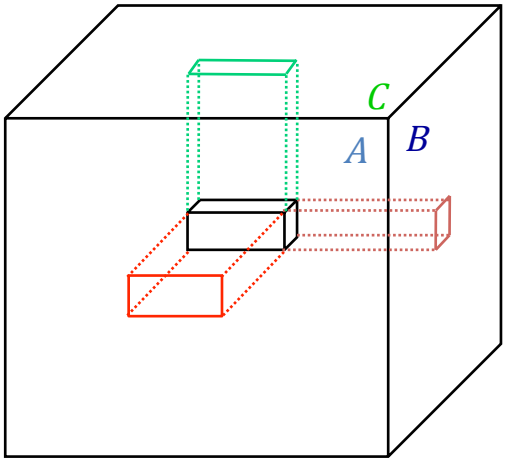
Dense:	**Sparse:**
Lower bounds match algorithms.	Significant gap
Allows extensive data reuse	Inherent poor reuse?

What do we obtain here?

- Improved (higher) lower bound
- New optimal algorithms
- **Only for random matrices**
  - Erdős-Rényi(n,d) graphs aka $G(n, p=d/n)$

The computation cube and sparsity-independent algorithms

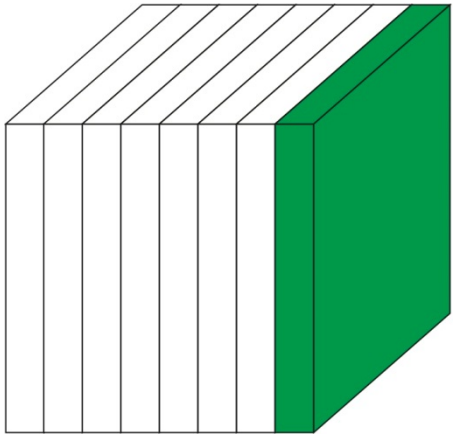
Matrix multiplication: $\forall(i,j) \in n \times n, \quad C(i,j) = \Sigma_k A(i,k)B(k,j),$



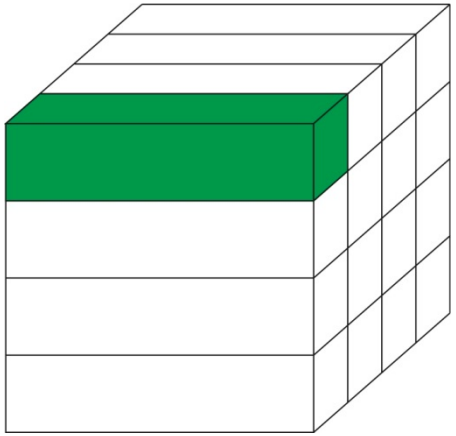
The *computation (discrete) cube*:

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

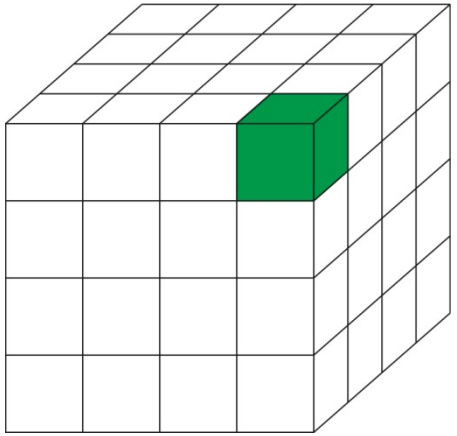
How about sparse algorithms?



1D algorithms



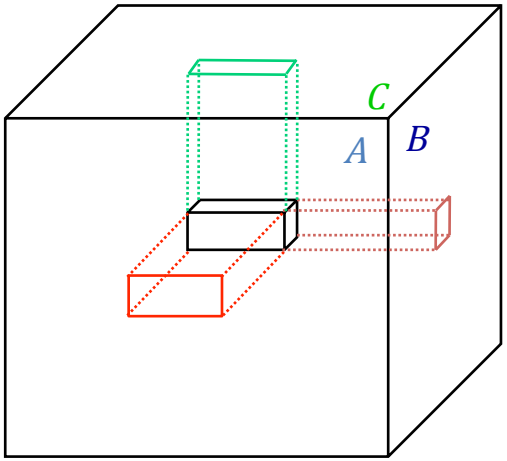
2D algorithms



3D algorithms

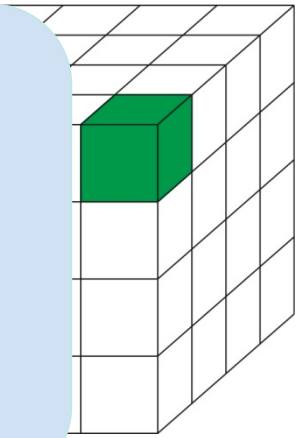
The computation cube and sparsity-independent algorithms

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



Sparsity independent algorithms:
assigning grid-points to processors is independent of sparsity structure.
- In particular: if C_{ij} is non-zero, who holds it?
- all standard algorithms are sparsity independent

Assumptions:
- Sparsity independent algorithms
- input (and output) are sparse:
- The algorithm is load balanced



1D algorithms 2D algorithms 3D algorithms

Algorithms attaining lower bounds

Previous Sparse Classical:

$$\Omega\left(\frac{\#FLOPs}{\left(\sqrt{M}\right)^3}\cdot\frac{M}{P}\right) = \Omega\left(\frac{d^2n}{P\sqrt{M}}\right)$$

✗ No algorithm attain this bound!

[Ballard, et al. SIMAX'11]

New Lower bound for Erdős-Rényi(n,d) :

$$\Omega\left(\min\left\{\frac{dn}{\sqrt{P}},\frac{d^2n}{P}\right\}\right)$$ ✓

[here] Expected
(Under some technical assumptions)

No previous algorithm attain these.

Two new algorithms achieving the bounds
(Up to a logarithmic factor)

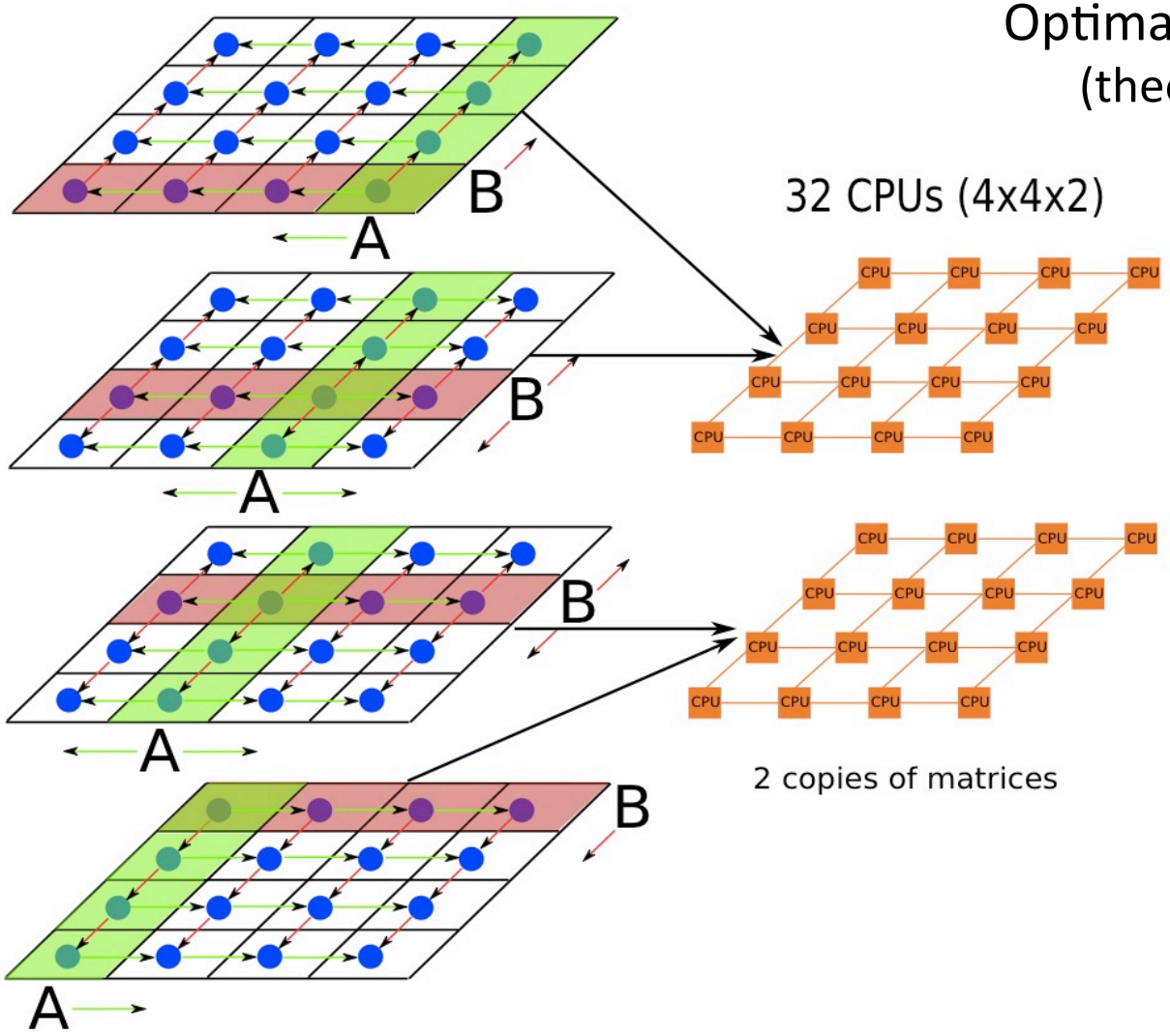
i. Recursive 3D, based on
[Ballard, et al. SPAA'12]

ii. Iterative 3D, based on
[Solomonik & Demmel EuroPar'11]

Ballard, **B.**, Demmel, Grigori, Lipshitz, Schwartz, and Toledo. Communication optimal parallel multiplication of sparse random matrices. In SPAA 2013.

3D Iterative Sparse GEMM

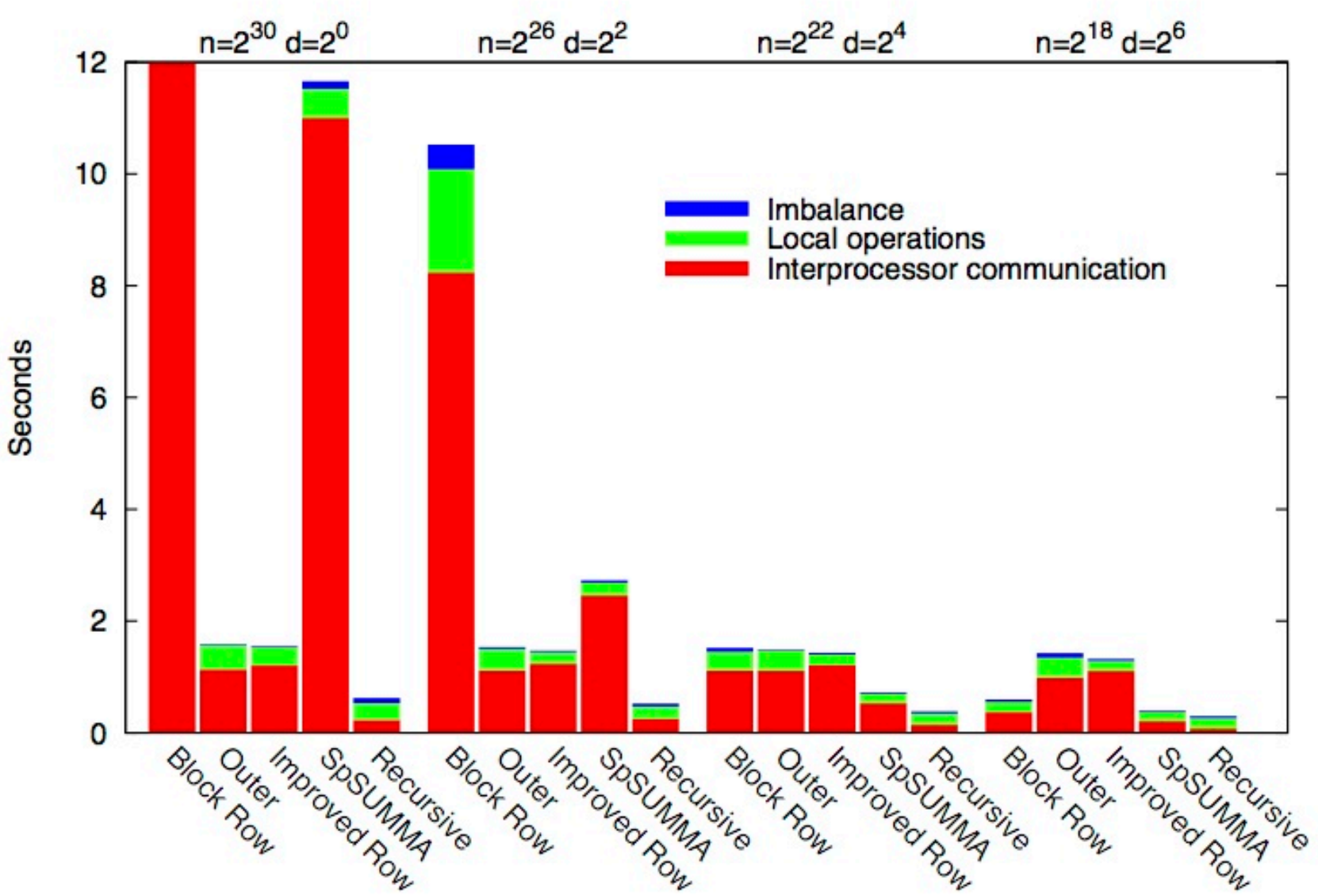
[Dense case: Solomonik and Demmel, 2011]



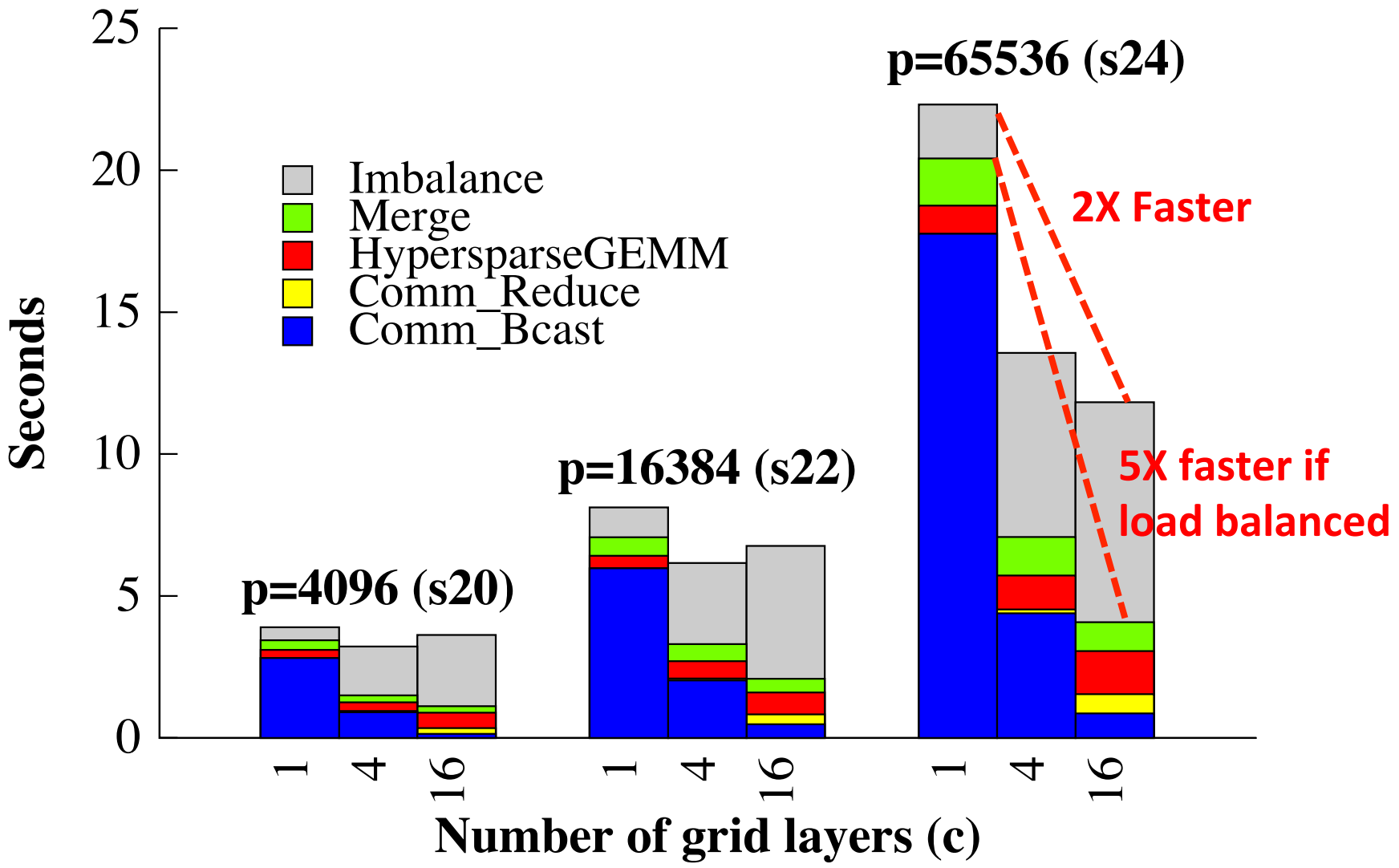
Optimal replicas: $c = \Theta\left(\frac{p}{d^2}\right)$
(theoretically)

3D Algorithms:
Using extra memory (c replicas) reduces communication volume by a factor of $c^{1/2}$ compared to 2D

Rerformance results (Erdős-Rényi graphs)



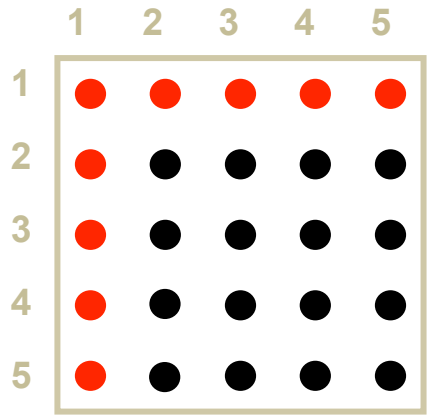
Iterative 2D-3D performance results (R-MAT graphs)



All-pairs shortest-paths problem

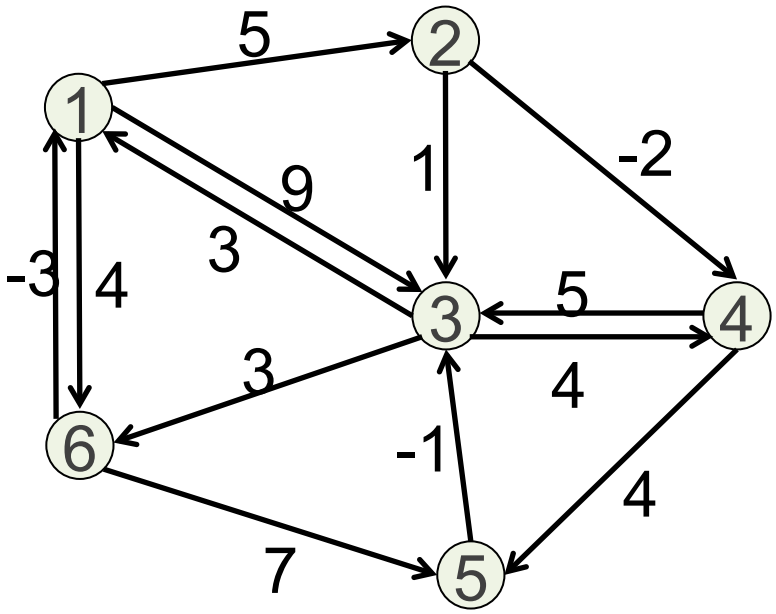
- Input: Directed graph with "costs" on edges
- Find least-cost paths between all reachable vertex pairs
- Classical algorithm: Floyd-Warshall

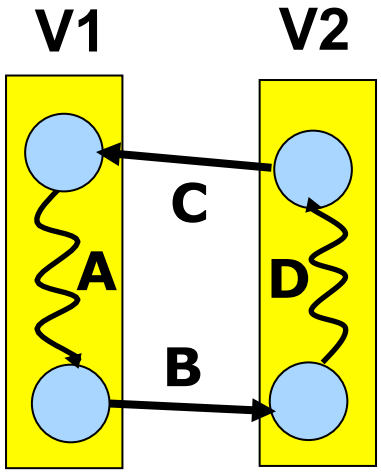
```
for k=1:n     // the induction sequence
    for i = 1:n
        for j = 1:n
            if(w(i→k)+w(k→j) < w(i→j))
                w(i→j):= w(i→k) + w(k→j)
```



k = 1 case

- It turns out a previously overlooked **recursive version** is more parallelizable than the triple nested loop



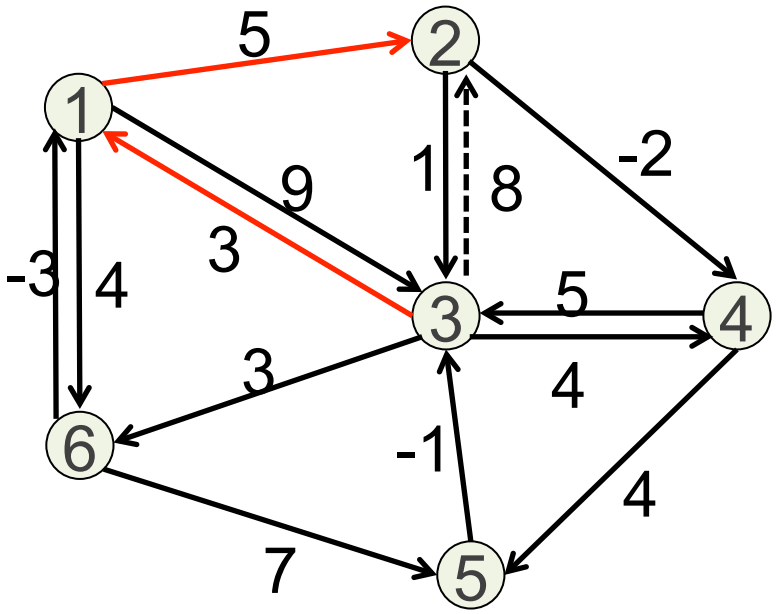


A	B
C	D

$$\begin{bmatrix} 0 & 5 & 9 & \infty & \infty & 4 \\ \infty & 0 & 1 & -2 & \infty & \infty \\ 3 & \infty & 0 & 4 & \infty & 3 \\ \infty & \infty & 5 & 0 & 4 & \infty \\ \infty & \infty & -1 & \infty & 0 & \infty \\ -3 & \infty & \infty & \infty & 7 & 0 \end{bmatrix}$$

+ is "min", **×** is "add"

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



$$\begin{bmatrix} \infty \\ 3 \end{bmatrix} \quad \begin{bmatrix} 5 & 9 \end{bmatrix} = \begin{bmatrix} \infty & \infty \\ 8 & 12 \end{bmatrix}$$

C **B**

The cost of 3-1-2 path

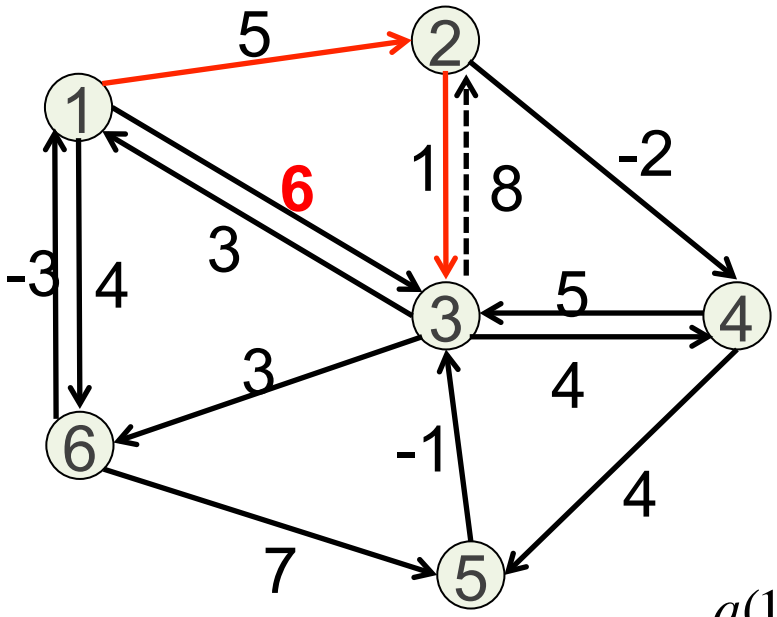
$$a(3,2) = a(3,1) + a(1,2) \xrightarrow{then} \Pi(3,2) = \Pi(1,2)$$

$$\left[\begin{array}{c|cc|ccc} 0 & 5 & 9 & \infty & \infty & 4 \\ \hline \infty & 0 & 1 & -2 & \infty & \infty \\ 3 & \mathbf{8} & 0 & 4 & \infty & 3 \\ \hline \infty & \infty & 5 & 0 & 4 & \infty \\ \infty & \infty & -1 & \infty & 0 & \infty \\ -3 & \infty & \infty & \infty & 7 & 0 \end{array}\right]$$

Distances

$$\Pi = \left[\begin{array}{c|cc|ccc} 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & \mathbf{1} & 3 & 3 & 3 & 3 \\ \hline 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{array}\right]$$

Parents



D = D*: no change

$$\begin{bmatrix} 5 & 9 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 8 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 6 \end{bmatrix}$$

B **D**

Path: 1-2-3

$$a(1,3) = a(1,2) + a(2,3) \xrightarrow{then} \Pi(1,3) = \Pi(2,3)$$

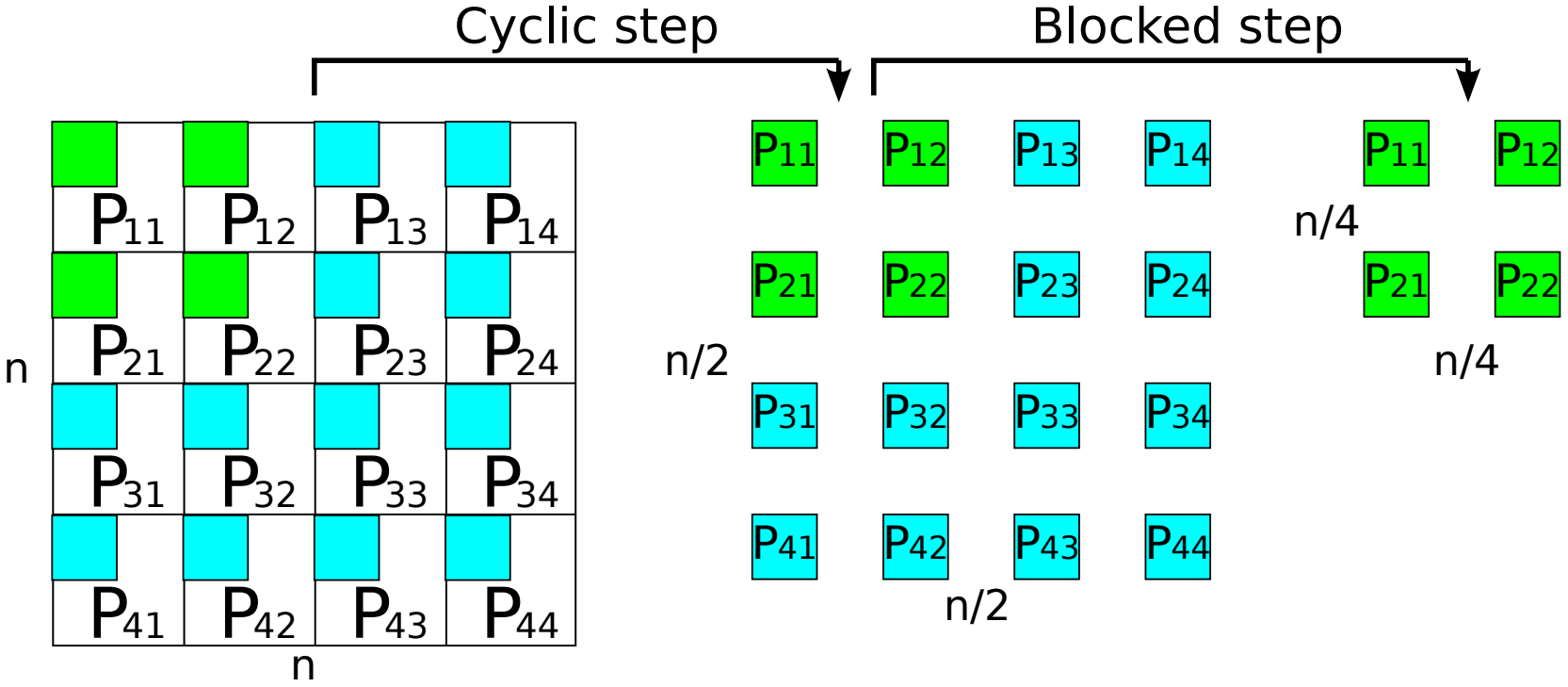
$$\begin{bmatrix} 0 & 5 & \mathbf{6} & \infty & \infty & 4 \\ \infty & 0 & 1 & -2 & \infty & \infty \\ 3 & 8 & 0 & 4 & \infty & 3 \\ \infty & \infty & 5 & 0 & 4 & \infty \\ \infty & \infty & -1 & \infty & 0 & \infty \\ -3 & \infty & \infty & \infty & 7 & 0 \end{bmatrix}$$

Distances

$$\Pi = \begin{bmatrix} 1 & 1 & \mathbf{2} & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 1 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Parents

Communication-avoiding APSP on distributed memory



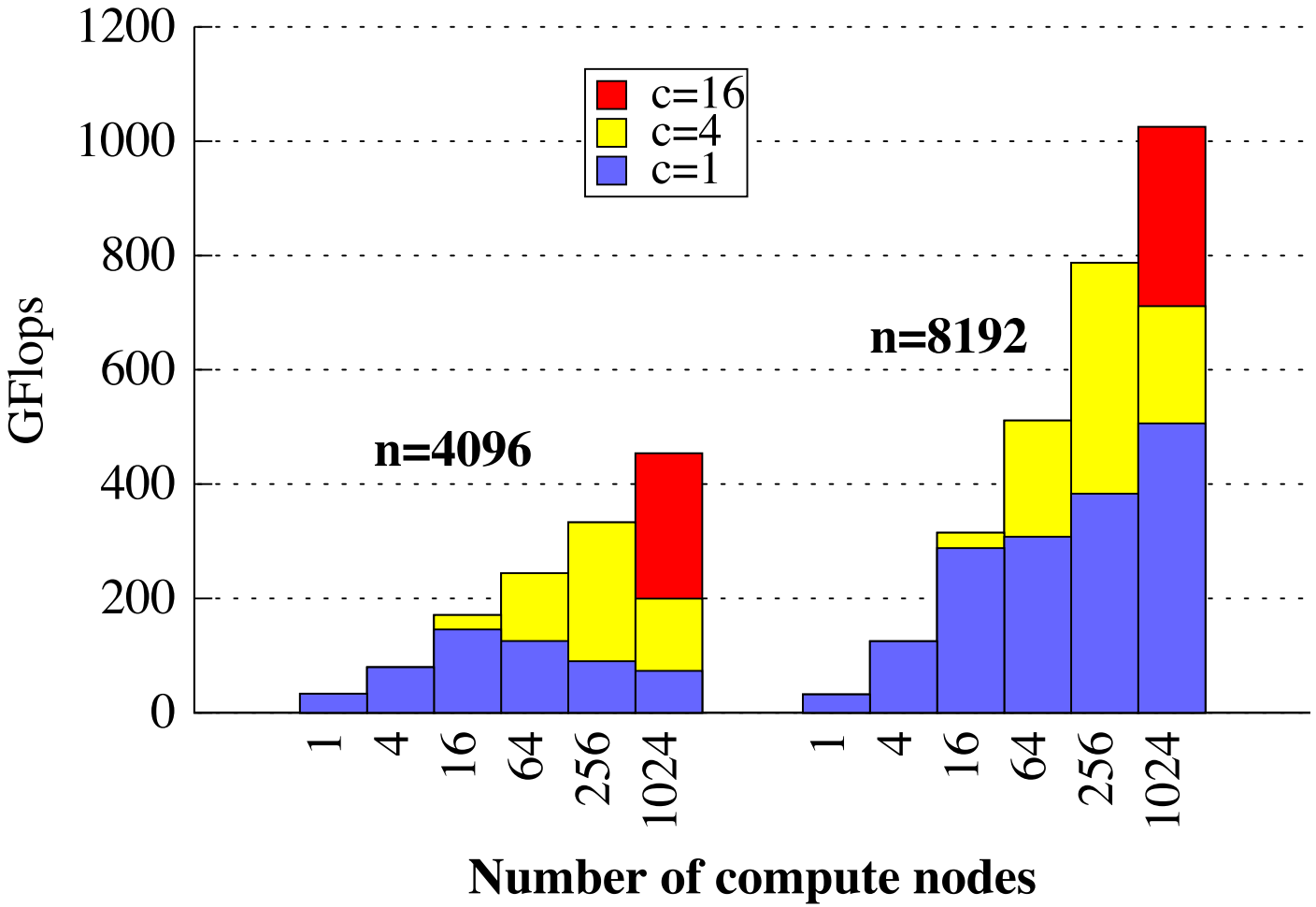
Bandwidth: $W_{\text{bc-2.5D}}(n,p) = O(n^2/\sqrt{cp})$

Latency: $S_{\text{bc-2.5D}}(p) = O\left(\sqrt{cp}\log^2(p)\right)$

c: number of replicas

Optimal for any memory size !

Communication-avoiding APSP on distributed memory



Solomonik, **B.**, and J. Demmel. “Minimizing communication in all-pairs shortest paths”, Proceedings of the IPDPS. 2013.

Conclusions