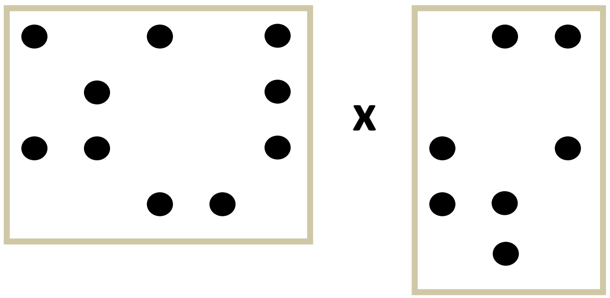
Communication-Avoiding Linear-Algebraic Primitives For Graph Analytics

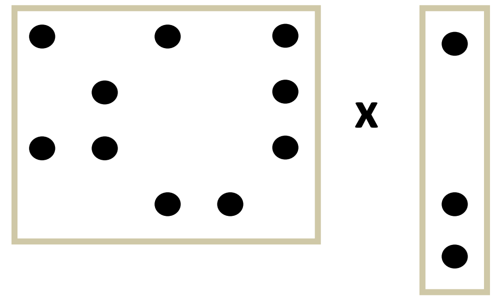
Aydın Buluç
Berkeley Lab (LBNL)
May 19, 2014
GABB'14 at IPDPS

Linear-algebraic primitives for graphs

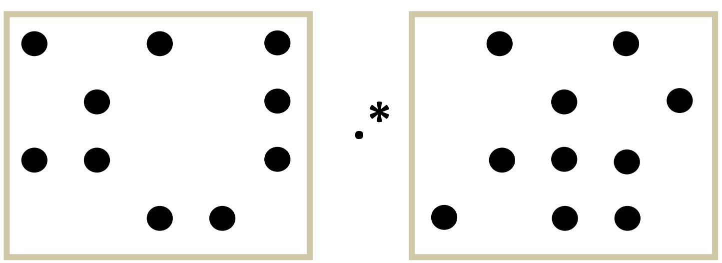
Sparse matrix-sparse matrix multiplication



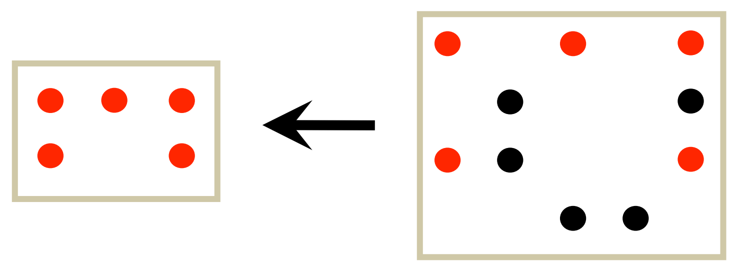
Sparse matrix-sparse vector multiplication



Element-wise operations



Sparse matrix indexing



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

Some Combinatorial BLAS functions

* Grossly simplified (parameters, semiring)

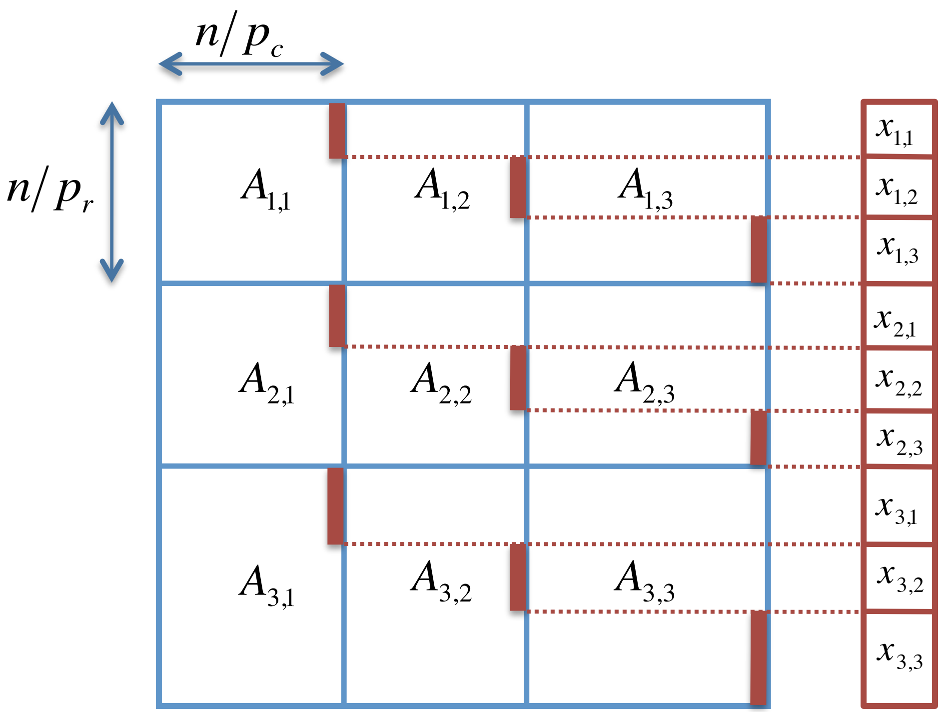
Function	Parameters	Returns	Math Notation
SpGEMM	- sparse matrices **A** and **B** - unary functors (op)	sparse matrix	**C** = op(**A**) * op(**B**)
SpM{Sp}V **(Sp: sparse)**	- sparse matrix **A** - sparse/dense vector **x**	sparse/dense vector	**y** = **A** * **x**
SpEWiseX	- sparse matrices or vectors - binary functor and predicate	in place or sparse matrix/vector	**C** = **A** .* **B**
Reduce	- sparse matrix **A** and functors	dense vector	**y** = sum(**A**, op)
SpRef	- sparse matrix **A** - index vectors **p** and **q**	sparse matrix	**B** = **A**(**p**,**q**)
SpAsgn	- sparse matrices **A** and **B** - index vectors **p** and **q**	none	**A**(**p**,**q**) = **B**
Scale	- sparse matrix **A** - dense matrix or vector **X**	none	check manual
Apply	- any matrix or vector **X** - unary functor (op)	none	op(**X**)

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

2D layout for sparse matrices & vectors



Matrix/vector distributions, interleaved on each other.

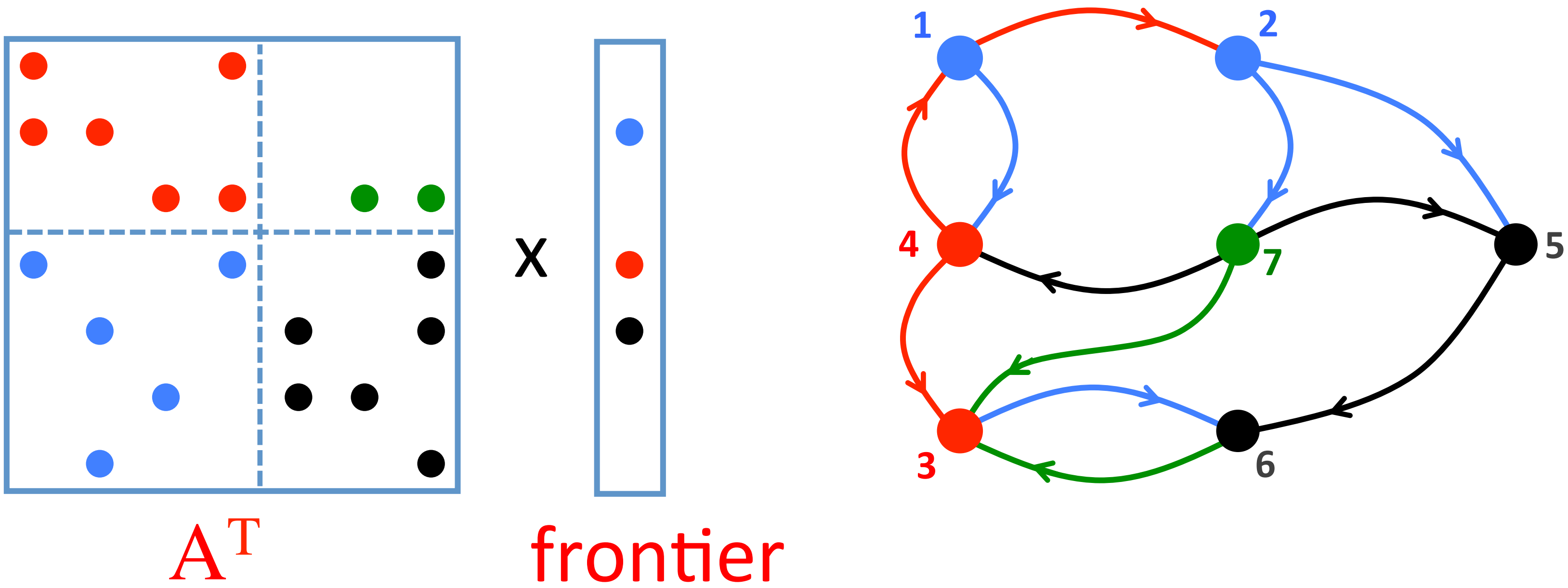
Default distribution in Combinatorial BLAS.

Scalable with increasing number of processes

- 2D matrix layout wins over 1D with large core counts and with limited bandwidth/compute
- 2D vector layout sometimes important for load balance

B., Gilbert. The Combinatorial BLAS: Design, implementation, and applications. *The International Journal of High Performance Computing Applications*, 2011.

2D parallel BFS algorithm

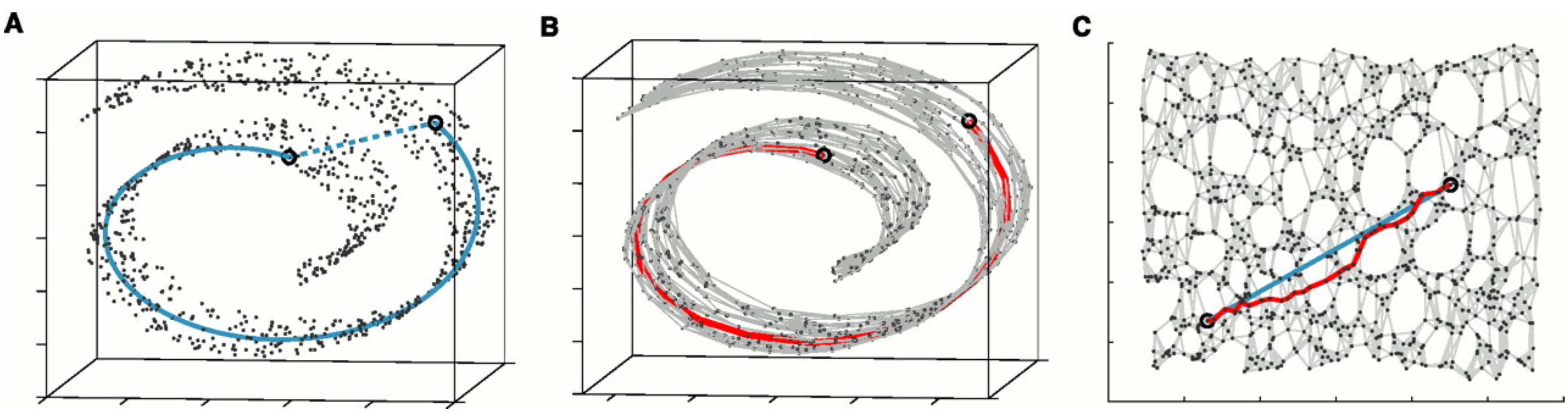


ALGORITHM:

1. Gather vertices in *processor column* **[communication]**
2. Find owners of the current frontier's adjacency [computation]
3. Exchange adjacencies in *processor row* **[communication]**
4. Update distances/parents for unvisited vertices. [computation]

Outline

- Communication-avoiding graph/sparse-matrix algorithms
- Why do they matter in science?
  - Isomap (non-linear dimensionality reduction) needs APSP
  - Parallel betweenness centrality needs sparse matrix-matrix product
- What kind of infrastructure/software support they need?



Tenenbaum, Joshua B., Vin De Silva, and John C. Langford. "A global geometric framework for nonlinear dimensionality reduction." Science 290.5500 (2000): 2319-2323.

Communication → Energy

Communication-avoiding algorithms: **Save time**

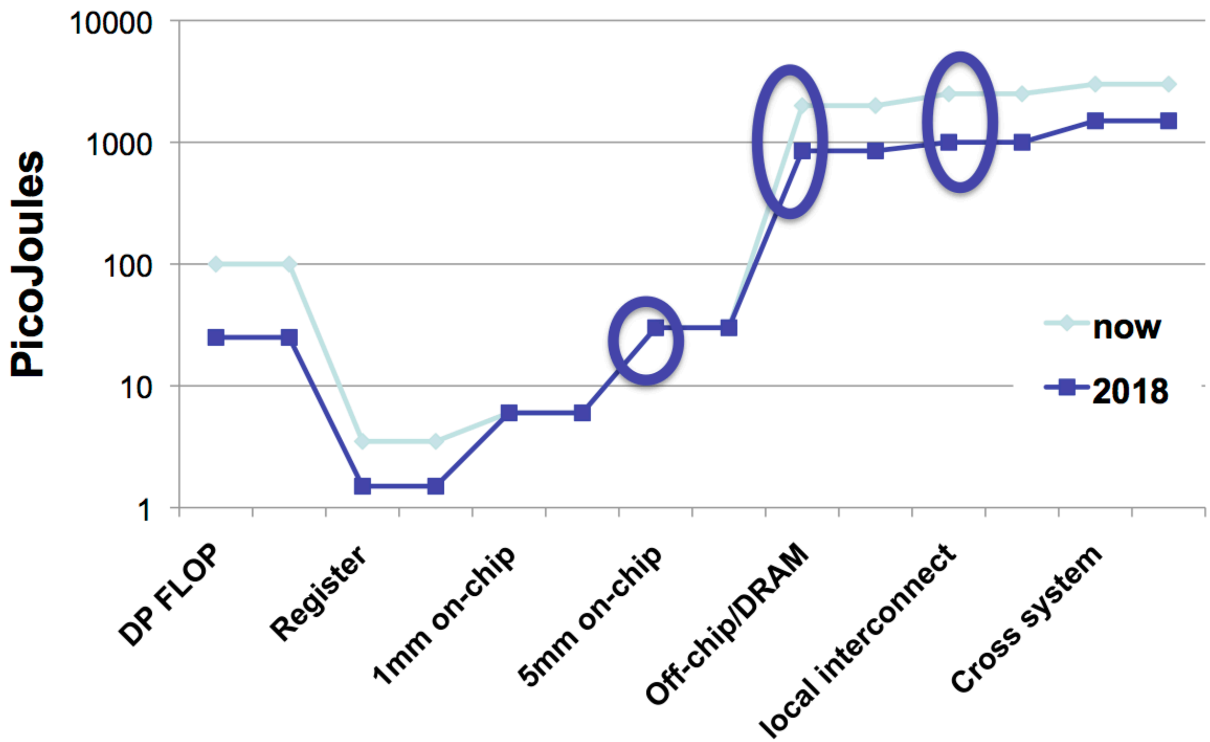


Image courtesy of John Shalf (LBNL)

Communication-avoiding algorithms: **Save energy**

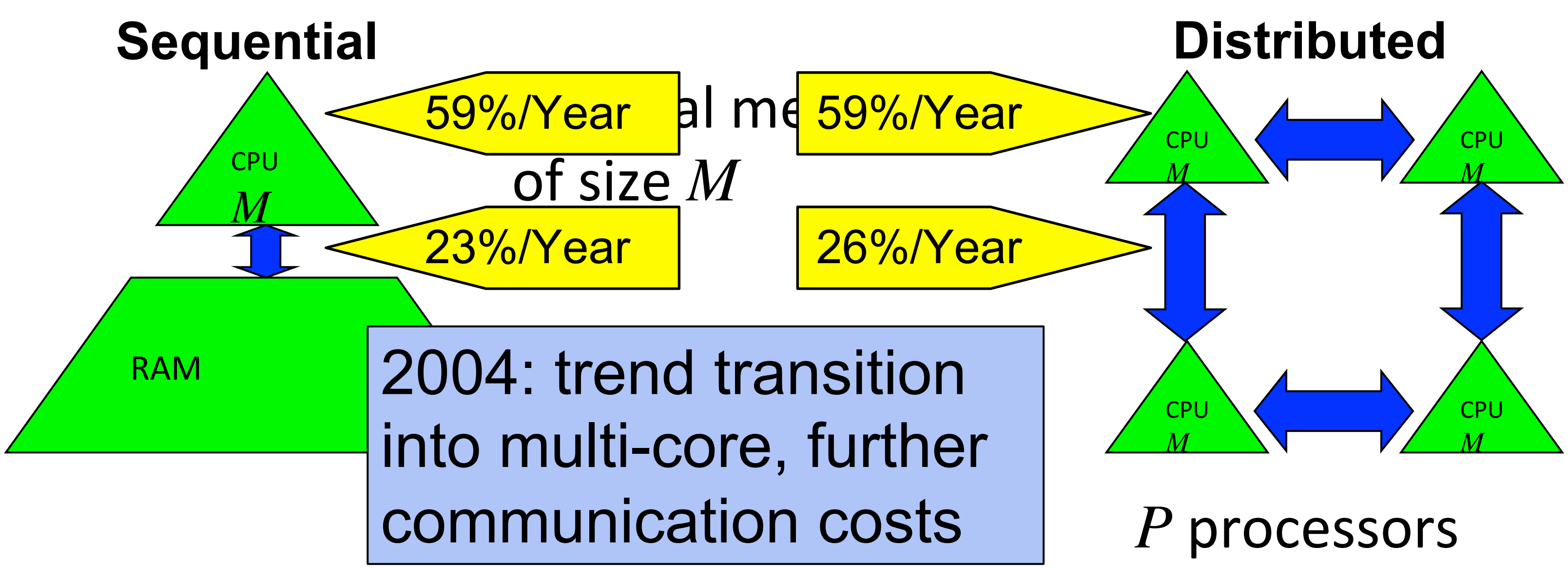
Communication-avoidance motivation

Two kinds of costs:

- Arithmetic (FLOPs)
- Communication: moving data

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

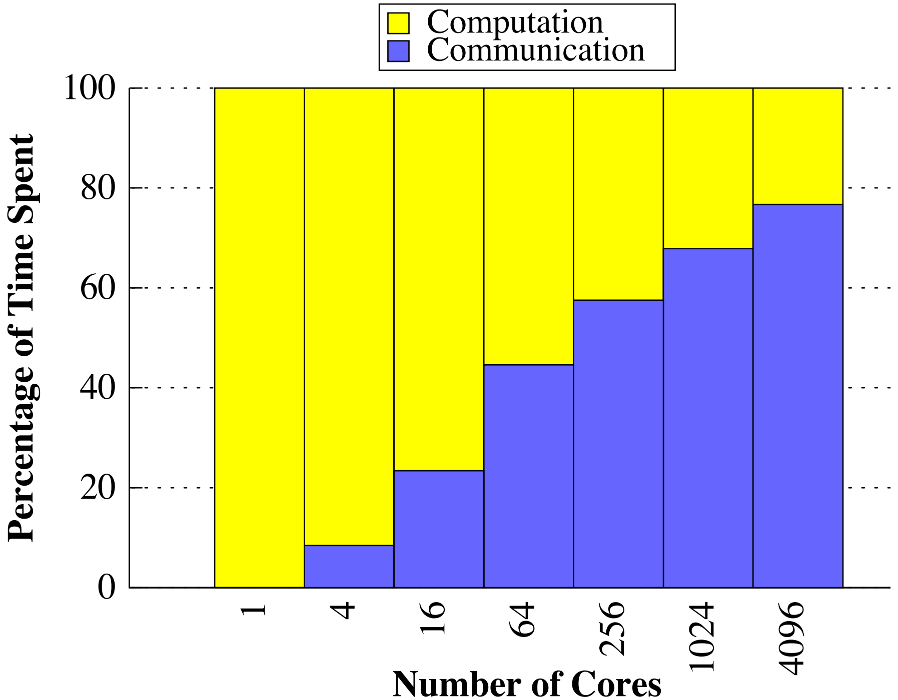
$$Running\ time\ = \gamma \cdot \#FLOPs + \beta \cdot \#Words \quad + (\alpha \cdot \#Messages)$$



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

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

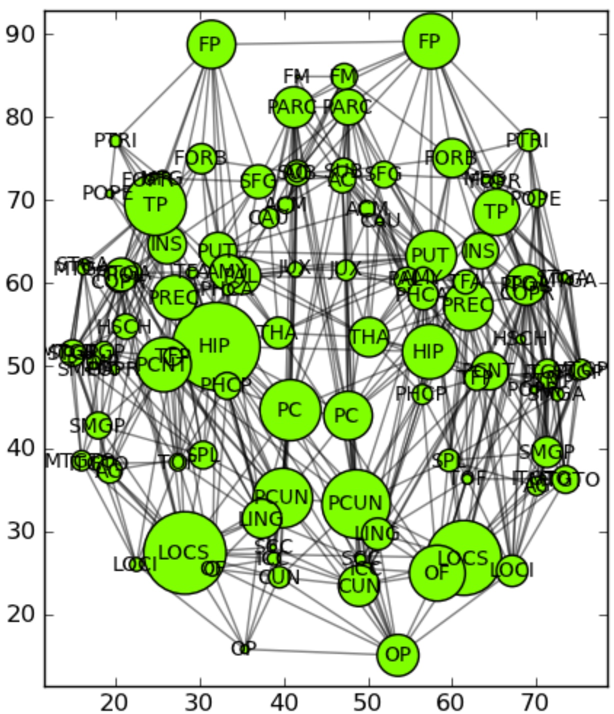
Scale 23 R-MAT (scale-free graph) **times** order 4 restriction operator

Cray XT4, Franklin, NERSC

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

Graph Analysis for the Brain

- Connective abnormalities in schizophrenia [van den Heuvel et al.]
  - Problem: understand disease from anatomical brain imaging
  - Tools: **betweenness centrality (BC)**, shortest path length
  - Results: global statistics on connection graph correlate w/ diagnosis



BC measures "influence"

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

Among all the shortest paths, what fraction passes through v?

$$Cost = O(mn)$$

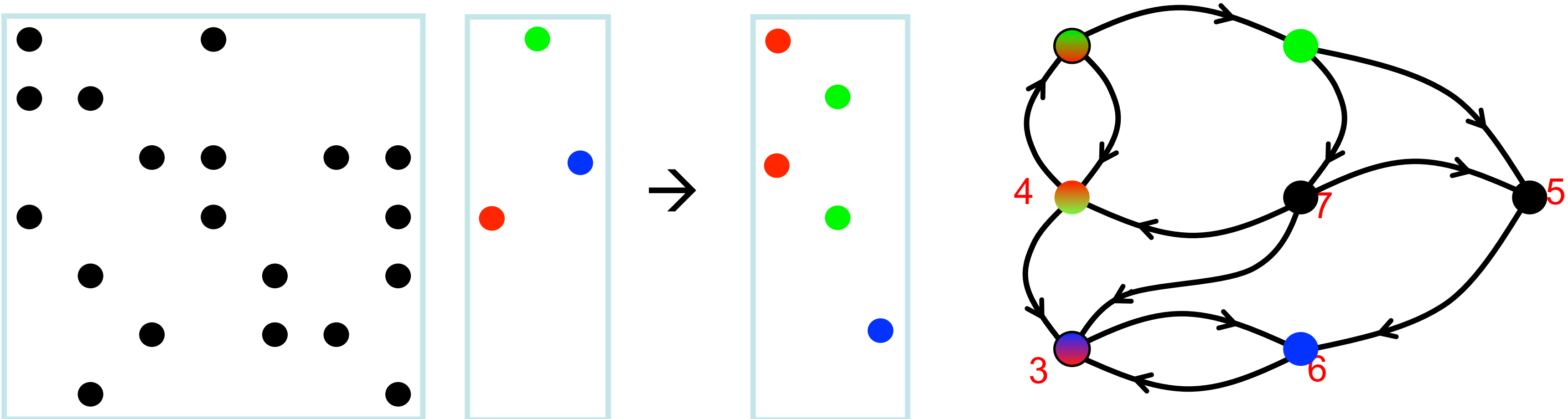
-unweighted-

Computationally intensive !

Input type	**Resolution**	**Time**
ROIs	256	seconds
Voxels	60,000	hours
Neurons	1M+	years???

Distributed memory BC algorithm

Work-efficient parallel breadth-first search via **parallel sparse matrix-matrix multiplication over semirings**



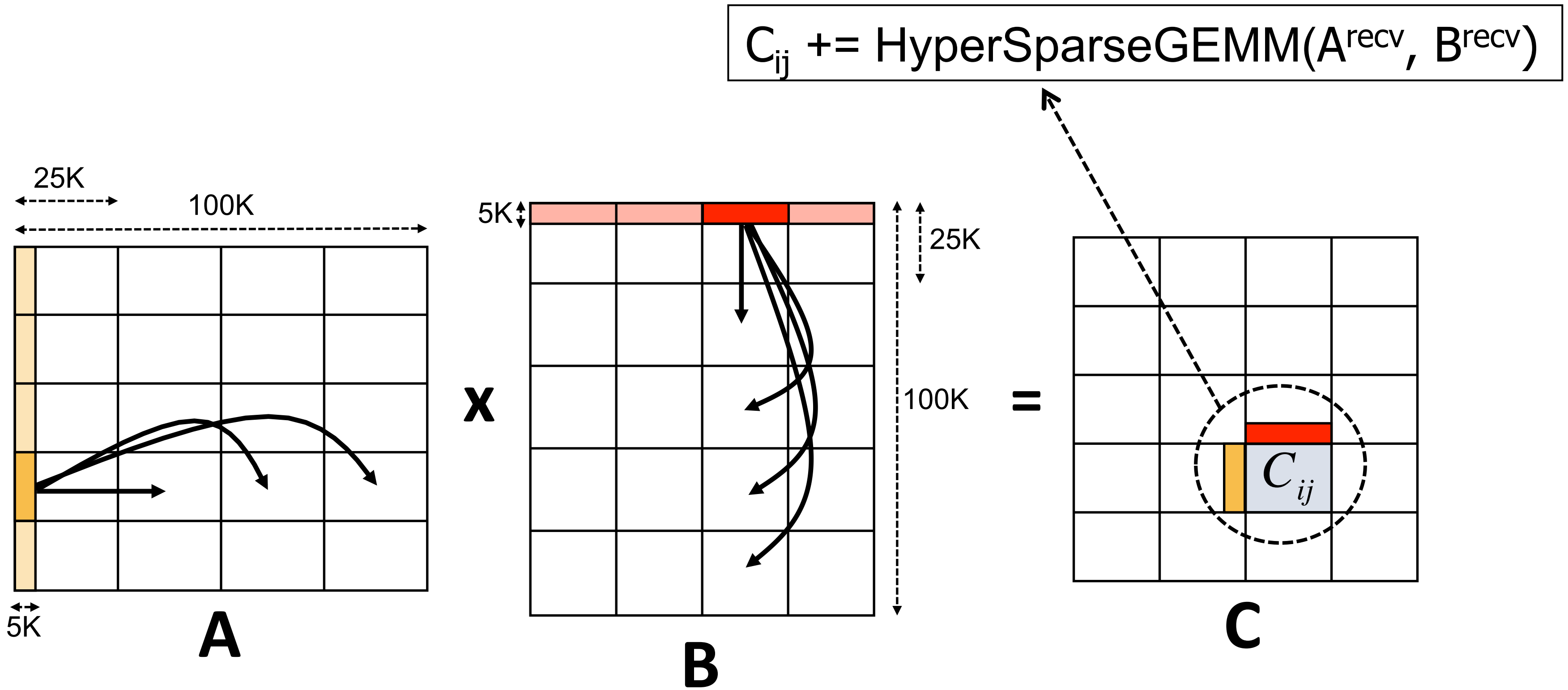
A^T $\quad$ B $\quad$ $(A^T X) .* \neg B$

Encapsulates <u>three level of parallelism:</u>

1. columns(B): multiple BFS searches in parallel
2. columns(A^T)+rows(B): parallel over frontier vertices in each BFS
3. rows(A^T): parallel over incident edges of each frontier vertex

A. Buluç, J.R. Gilbert. The Combinatorial BLAS: Design, implementation, and applications. IJHPCA, 2011.

Parallel sparse matrix-matrix multiplication algorithms

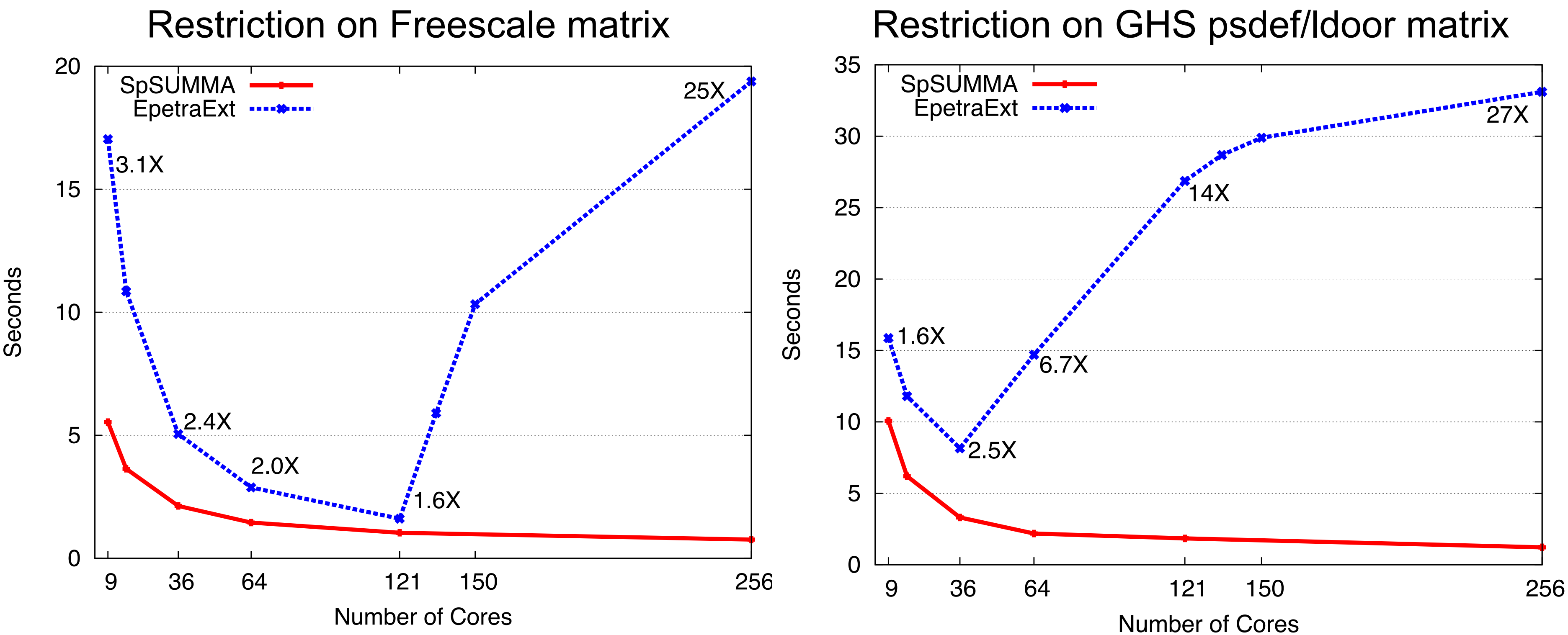


2D algorithm: Sparse SUMMA (based on dense SUMMA)

General implementation that handles rectangular matrices

B., Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. In *ICPP'08*

1D vs. 2D scaling for sparse matrix-matrix multiplication



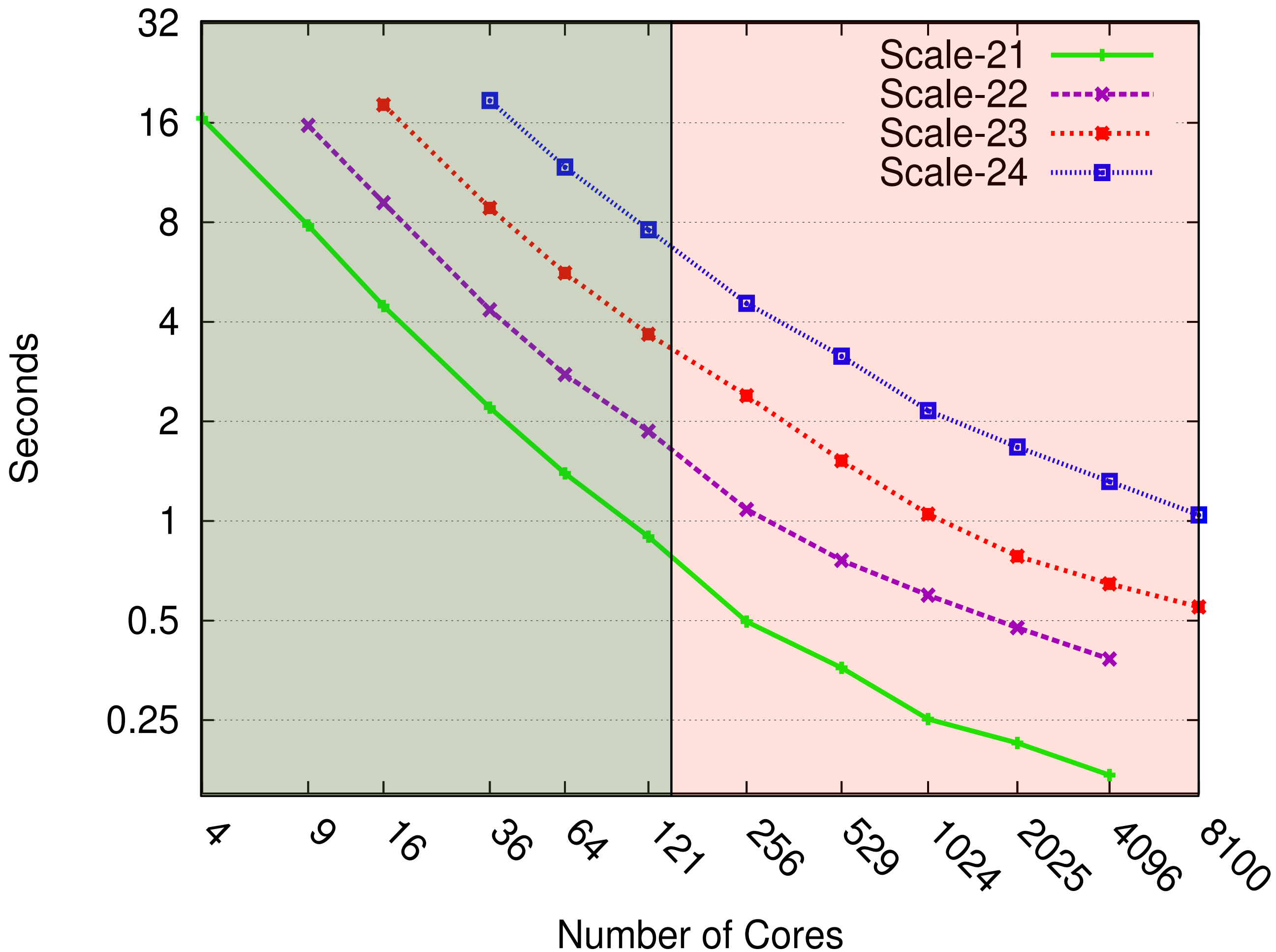
SpSUMMA = 2-D data layout (Combinatorial BLAS)
EpetraExt = 1-D data layout (Trilinos)

In practice, 2D algorithms have <u>the potential</u> to scale, but not linearly

$$T_{comm}(2\mathrm{D}) = \alpha p\sqrt{p} + \beta c n\sqrt{p} \qquad T_{comp}(optimal) = c^2 n$$

2D sparse matrix multiplication

Almost linear scaling until bandwidth costs starts to dominate



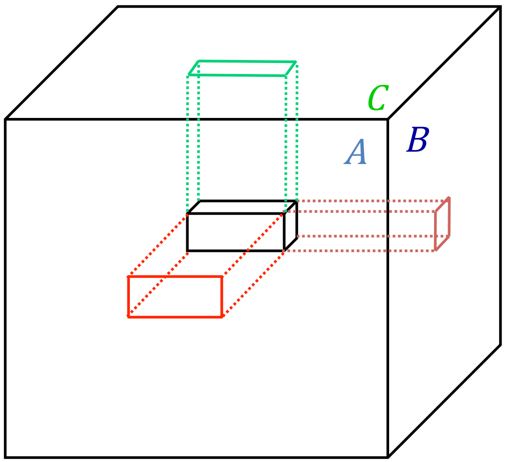
Scaling proportional to $\sqrt{p}$ afterwards

NERSC/Franklin
Cray XT4

R-MAT, edgefactor: 8
a=0.6, b=c=d=0.4/3

The computation cube and sparsity-independent algorithms

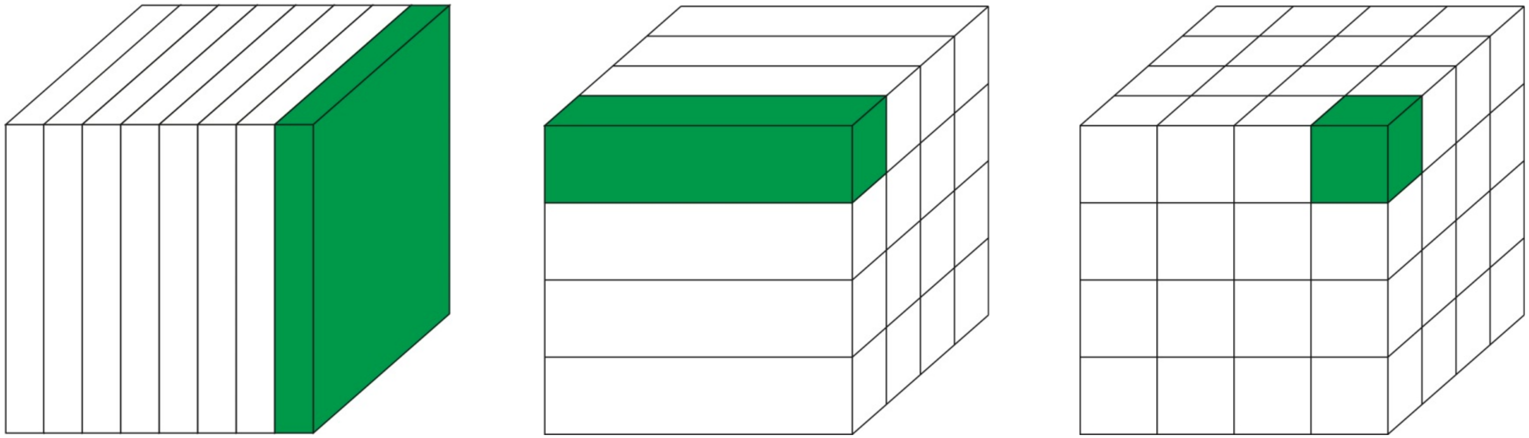
Matrix multiplication: $\forall (i,j) \in n \times n, \quad C(i,j) = \Sigma_k A(i,k)B(k,j),$



The *computation (discrete) cube*:

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

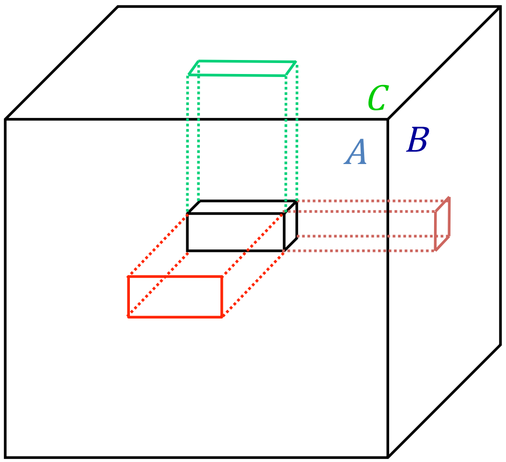
How about sparse algorithms?



1D algorithms 2D algorithms 3D algorithms

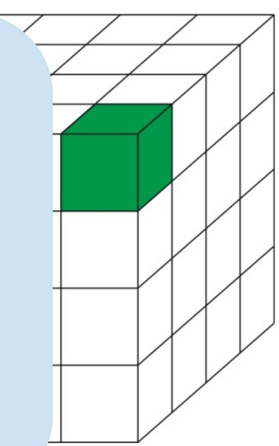
The computation cube and sparsity-independent algorithms

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:

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

No algorithm attain this bound!

[Ballard, et al. SIMAX'11]

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

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

[here] Expected
(Under some technical assumptions)

No previous algorithm attain these.

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

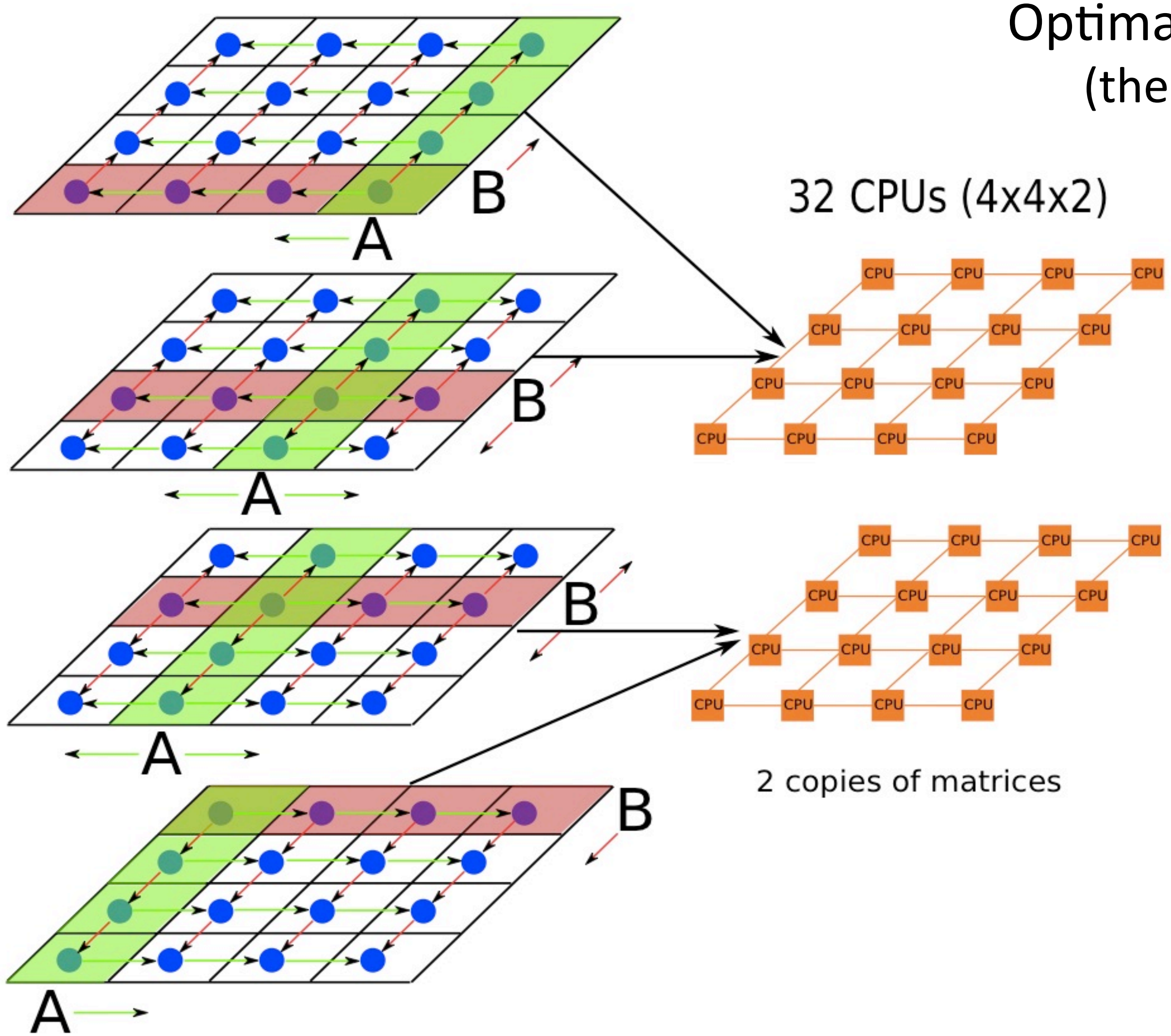
i. Recursive 3D, based on
[Ballard, et al. SPAA'12]

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

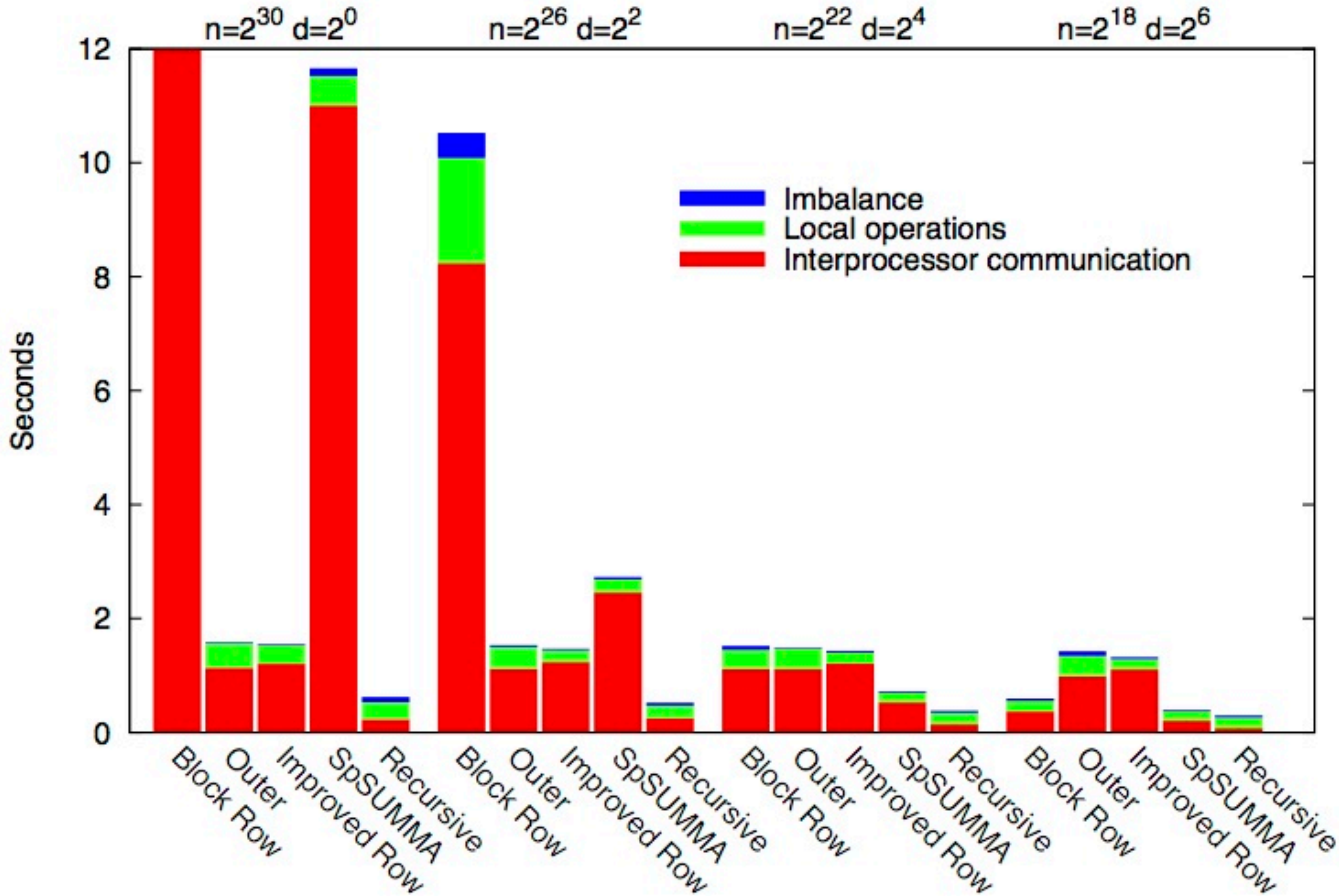
[Dense case: Solomonik and Demmel, 2011]



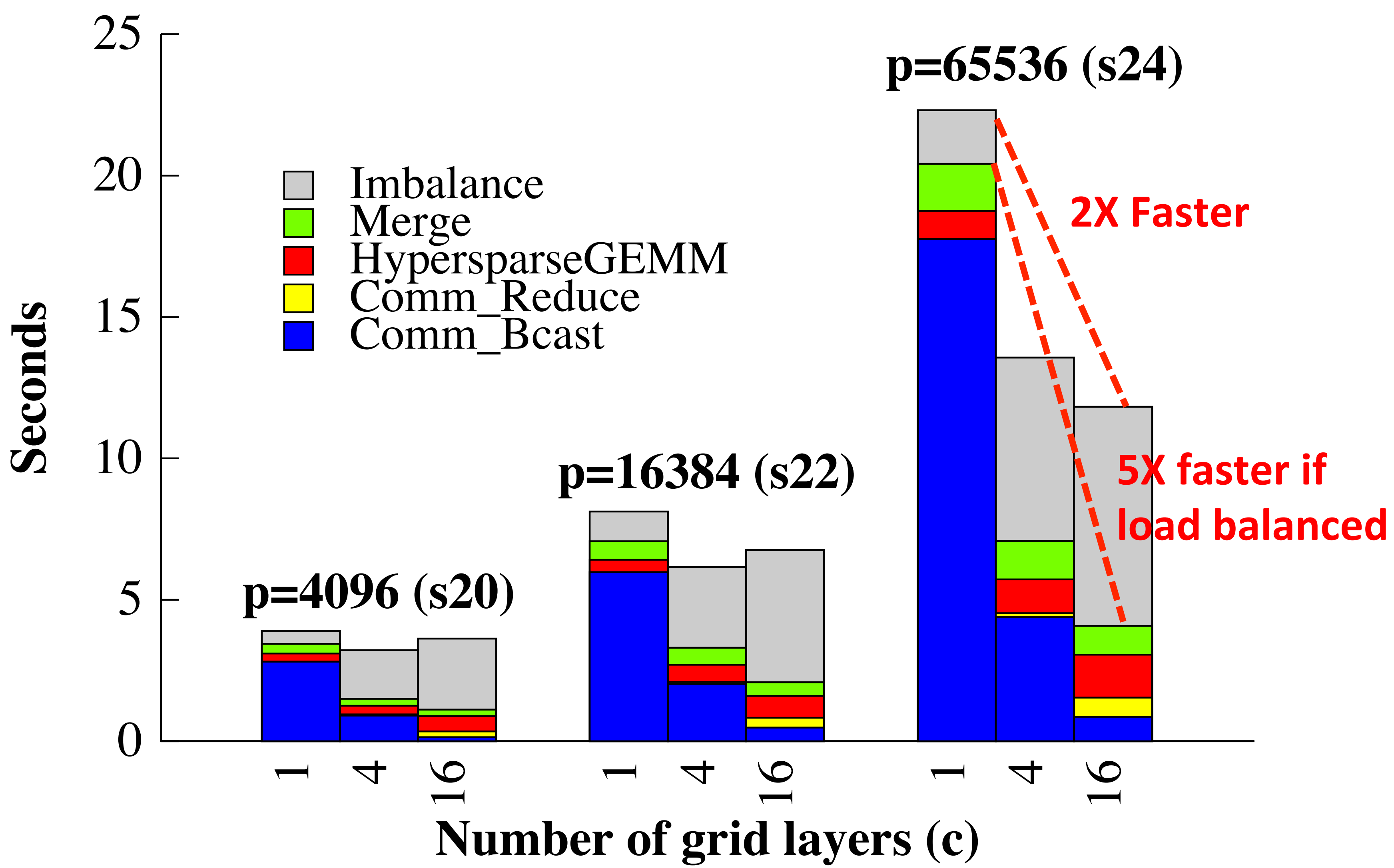
Optimal replicas: $c = \Theta\left(\frac{p}{d^2}\right)$
(theoretically)

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)



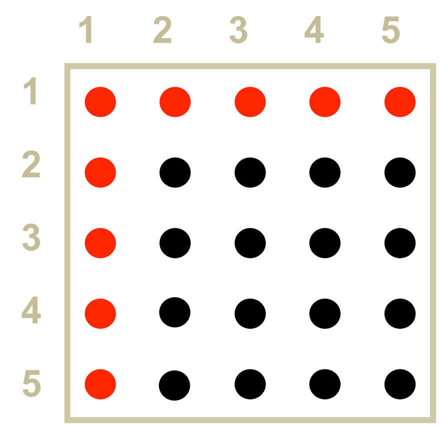
Iterative 2D-3D performance results (R-MAT graphs)



All-pairs shortest-paths problem

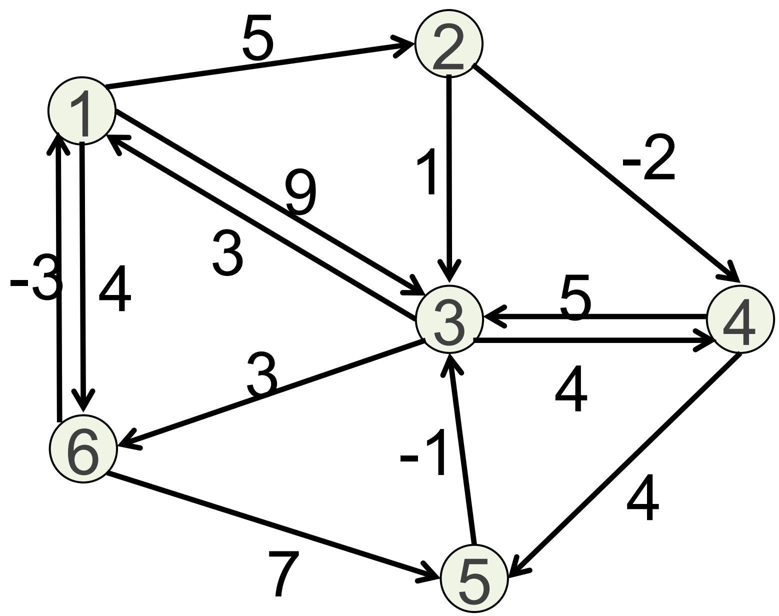
- <u>Input:</u> Directed graph with "costs" on edges
- Find least-cost paths between all reachable vertex pairs
- Classical algorithm: Floyd-Warshall

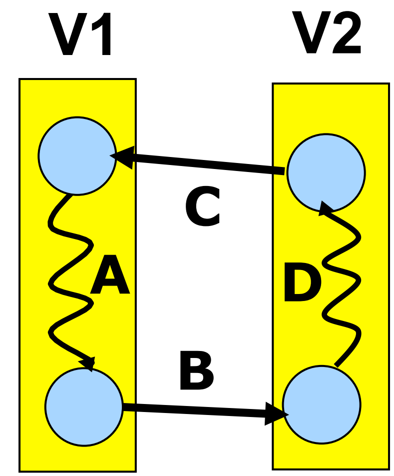
```
for k=1:n     // the induction sequence
    for i = 1:n
        for j = 1:n
            if(w(i→k)+w(k→j) < w(i→j))
                w(i→j):= w(i→k) + w(k→j)
```



k = 1 case

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





A	B
C	D

+ is “min”, × is “add”

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

A = A*; % recursive call

B = AB; C = CA;

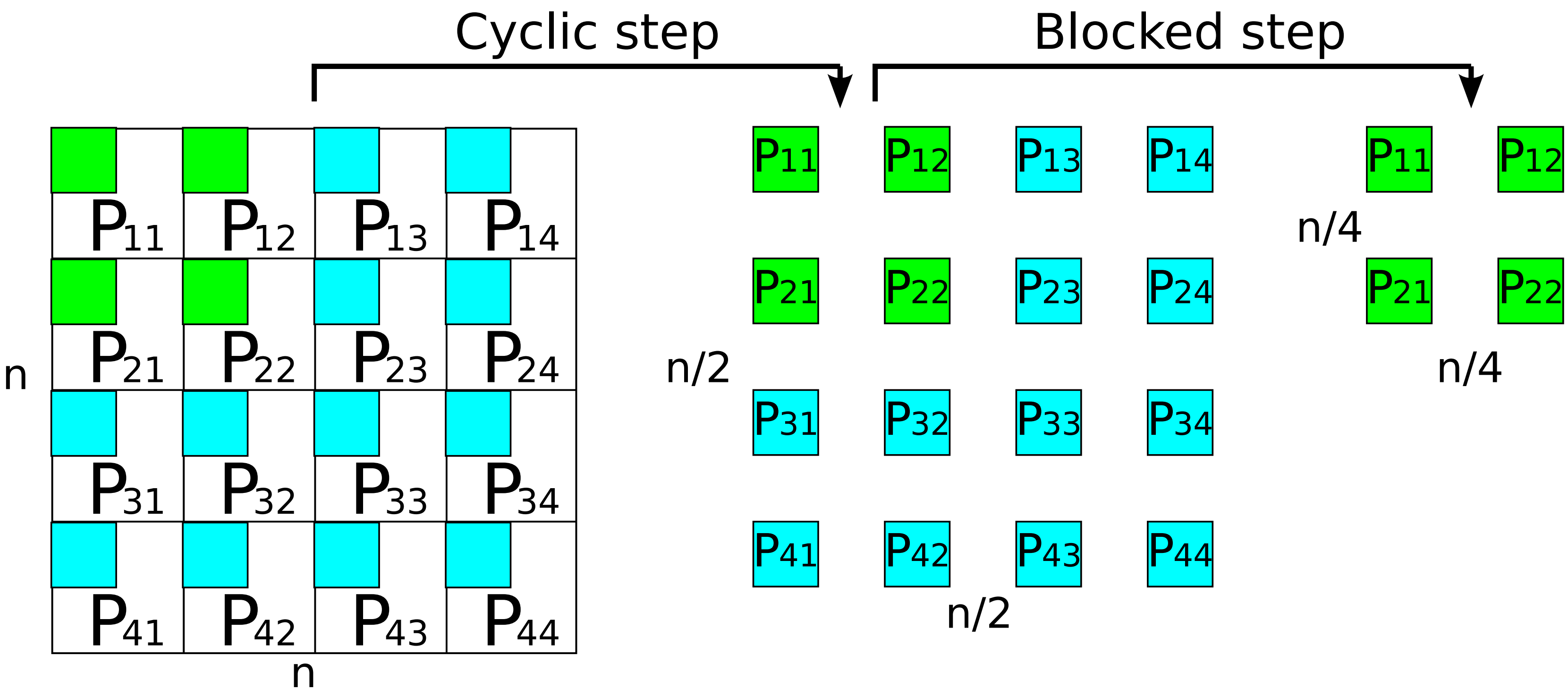
D = D + CB;

D = D*; % recursive call

B = BD; C = DC;

A = A + BC;

Communication-avoiding APSP on distributed memory



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

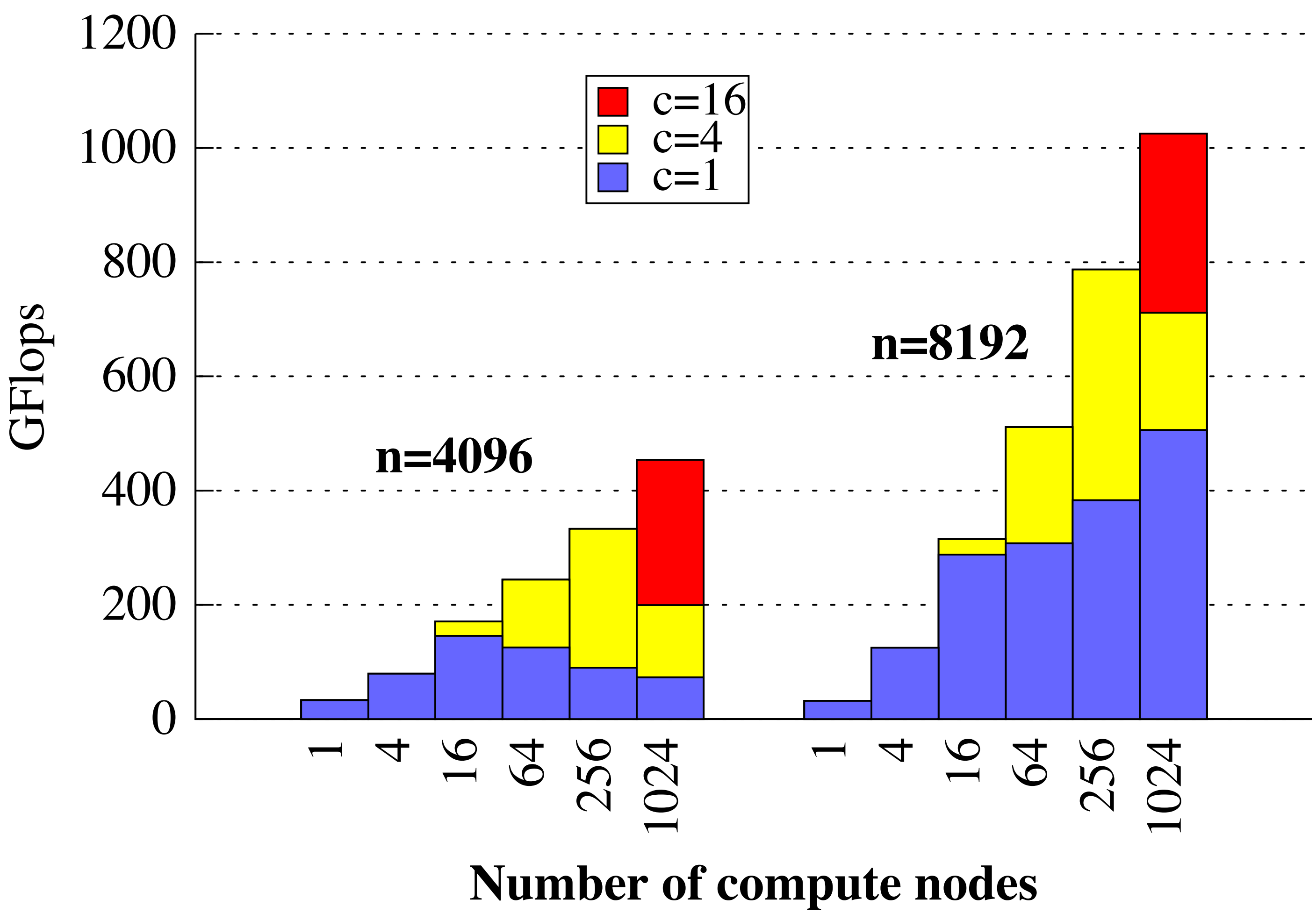
Latency: $S_{\text{bc-2.5D}}(p) = O\left(\sqrt{cp}\,\log^2(p)\right)$

c: number of replicas

Optimal for any memory size !

Communication-avoiding APSP on distributed memory

65K vertex dense problem solved in about two minutes



Solomonik, **B.**, and J. Demmel. "Minimizing communication in all-pairs shortest paths", IPDPS. 2013.

Software for communication-avoiding graph kernels and beyond

- Combinatorial BLAS used a static 2D block data distribution
  - Significantly better than 1D in almost all cases
  - Most ***communication-avoiding algorithms are 3D***
  - Infrastructure for ***recursion in distributed-memory*** is needed
  - Support for ***automatic switching between dense/sparse formats***
- Combinatorial BLAS used text I/O and non-portable binary
  - Move onto future proof self-describing format such as HDF5
  - Build a community file format on top of HDF5
- Combinatorial BLAS accepts **graphs** as input
  - Often the graph needs to be constructed from data ($G=A^{T}T$)
- Combinatorial BLAS API developed organically
  - Need to contribute to a standard interface

Conclusions

- Parallel graph libraries are crucial for analyzing big data.
- In addition to being high performance and scalable, the library should be flexible and simple to use.
- Linear-algebraic primitives should provide the core library.
- Avoiding communication improves performance and energy.

- KDT + Combinatorial BLAS was an excellent proof of concept.
- A more elegant, simpler, complete system to emerge that:
  - Is developed ***concurrently with the Graph BLAS standard***
  - Implements ***communication-avoiding algorithms***
  - Interoperates seamlessly with non-linear algebraic pieces such as the native graph systems like GraphLab and graph databases.

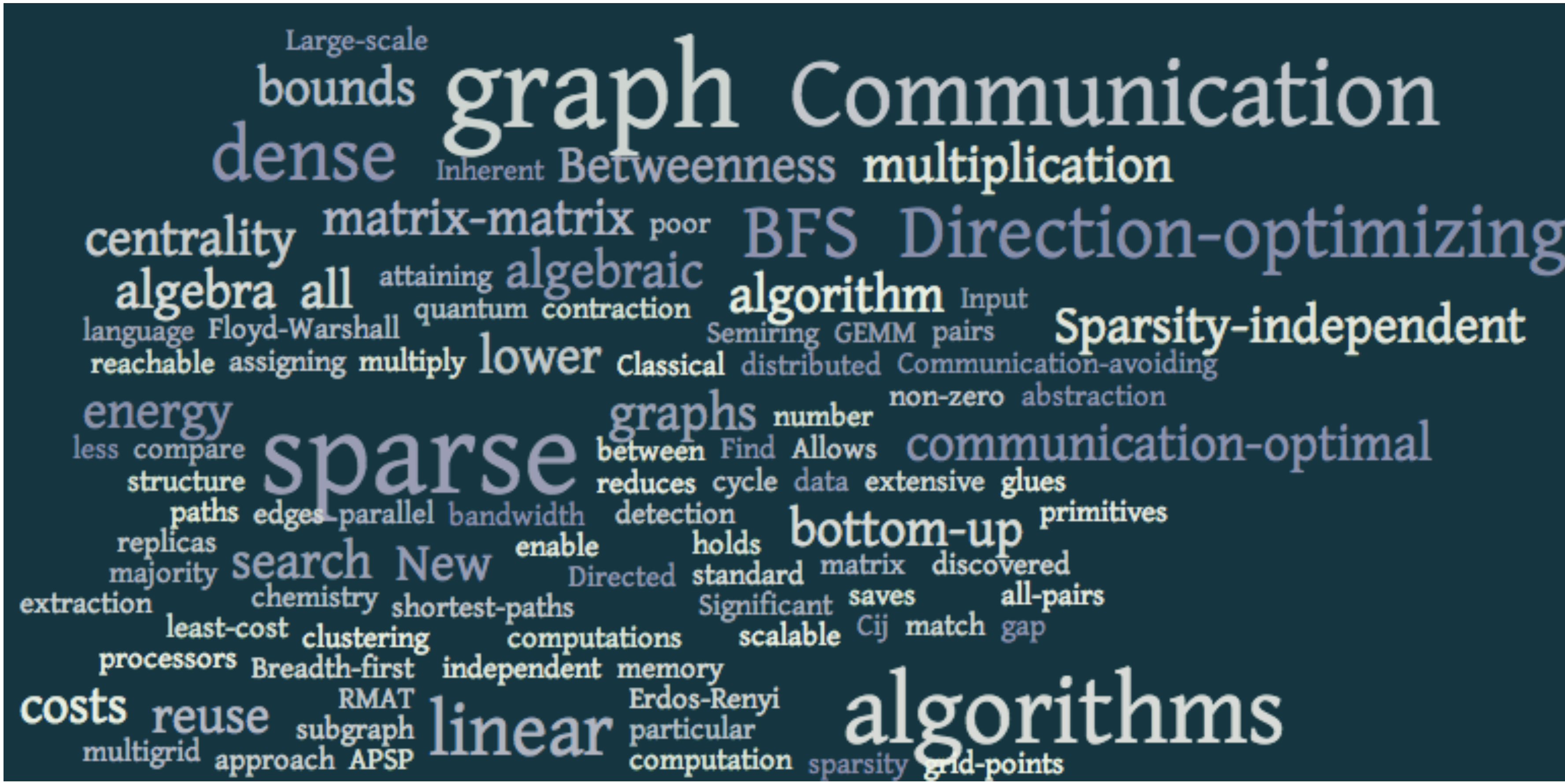
Acknowledgments

- Grey Ballard (UCB)
- Jarrod Chapman (JGI)
- Jim Demmel (UC Berkeley)
- John Gilbert (UCSB)
- Evangelos Georganas (UCB)
- Laura Grigori (INRIA)
- Ben Lipshitz (UCB)
- Adam Lugowski (UCSB)
- Steve Reinhardt (Cray)
- Dan Rokhsar (JGI/UCB)
- Lenny Oliker (Berkeley Lab)
- Oded Schwartz (UCB)
- Edgar Solomonik (UCB)
- Veronika Strnadova (UCSB)
- Sivan Toledo (Tel Aviv Univ)
- Dani Ushizima (Berkeley Lab)
- Kathy Yelick (Berkeley Lab/UCB)

This work is funded by:

U.S. DEPARTMENT OF ENERGY | Office of Science

Questions?


Large-scale
bounds graph Communication
dense Inherent Betweenness multiplication
centrality matrix-matrix poor BFS Direction-optimizing
algebra all attaining algebraic algorithm Input
quantum contraction
language Floyd-Warshall Semiring GEMM pairs Sparsity-independent
reachable assigning multiply lower Classical distributed Communication-avoiding
non-zero abstraction
energy graphs number
less compare sparse between Find Allows communication-optimal
structure reduces cycle data extensive glues
paths edges parallel bandwidth detection primitives
replicas enable holds bottom-up
majority search New Directed standard matrix discovered
extraction chemistry shortest-paths Significant saves all-pairs
least-cost clustering computations scalable Cij match gap
processors Breadth-first independent memory
RMAT Erdos-Renyi
costs reuse subgraph linear particular algorithms
multigrid approach APSP computation sparsity grid-points