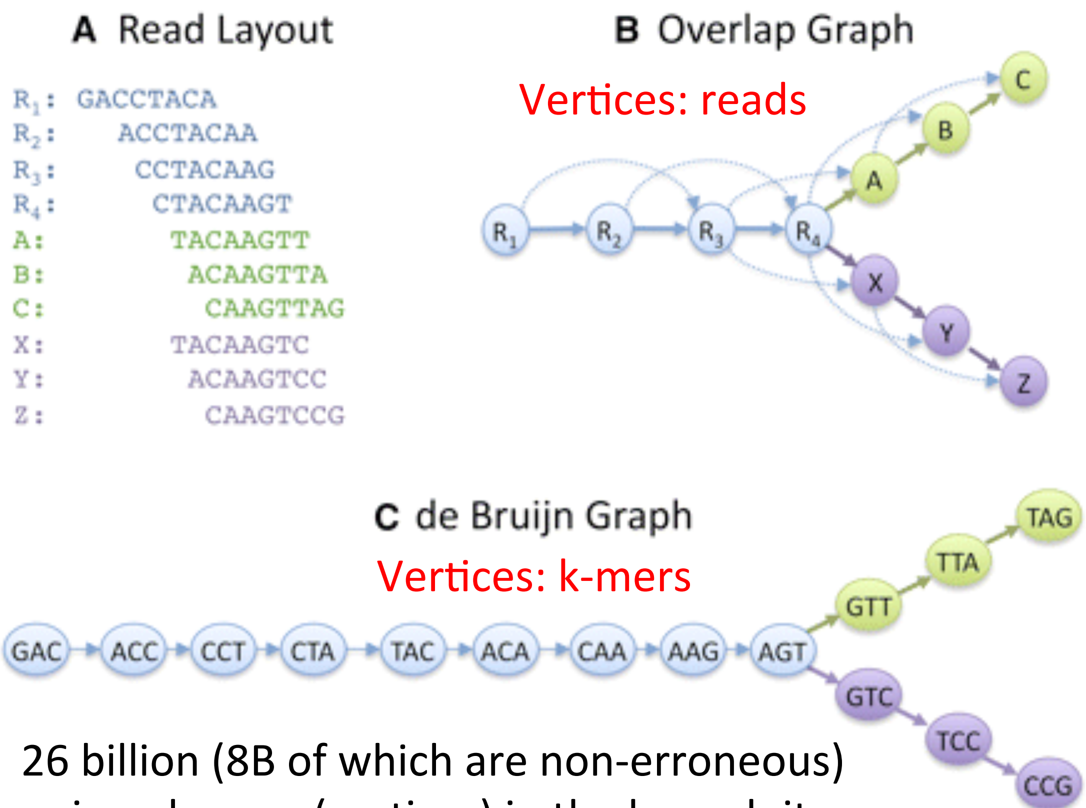
# The Graph BLAS effort and its implications for Exascale

David Bader (GA Tech), **Aydın Buluç (LBNL)**, John Gilbert (UCSB), Joseph Gonzalez (UCB), Jeremy Kepner (MIT LL), Tim Mattson (Intel)

SIAM Workshop on Exascale Applied Mathematics Challenges and Opportunities (EX14)
July 6, 2014

# Graphs matter in Applied Math

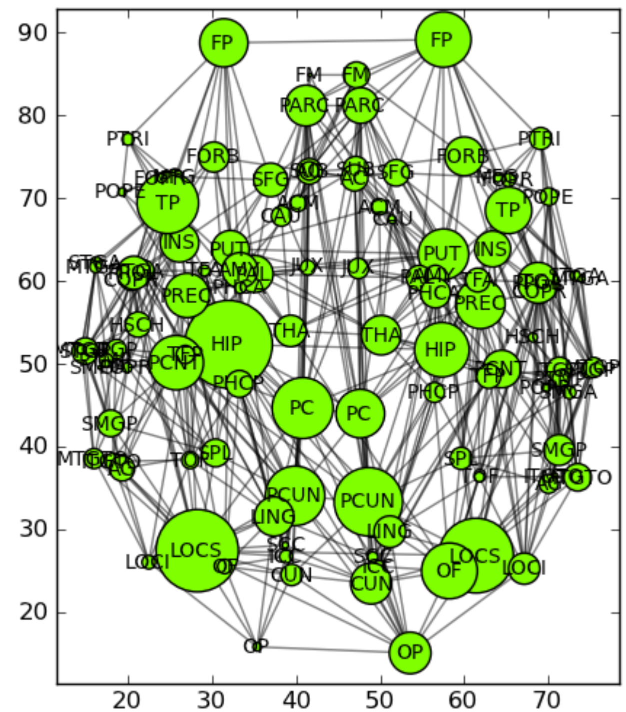
## Whole genome assembly




26 billion (8B of which are non-erroneous) unique k-mers (vertices) in the hexaploit wheat genome W7984 for k=51

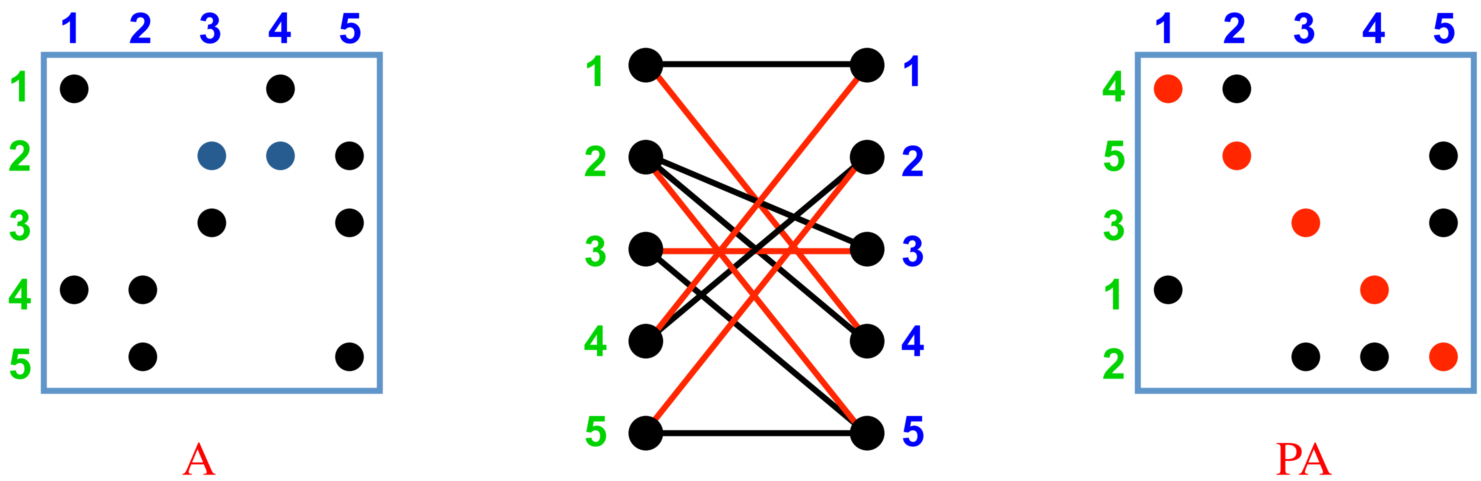
Schatz et al. (2010) Perspective: Assembly of Large Genomes w/2nd-Gen Seq. Genome Res. (figure reference)

## Graph Theoretical analysis of Brain Connectivity

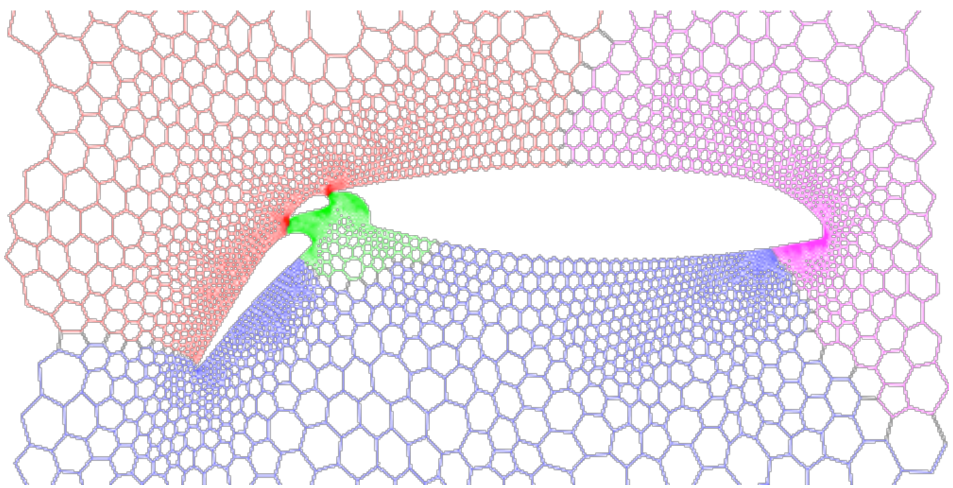



Potentially millions of neurons and billions of edges with developing technologies

# Graphs matter in Applied Math




Matching in bipartite graphs: Permuting to heavy diagonal or block triangular form

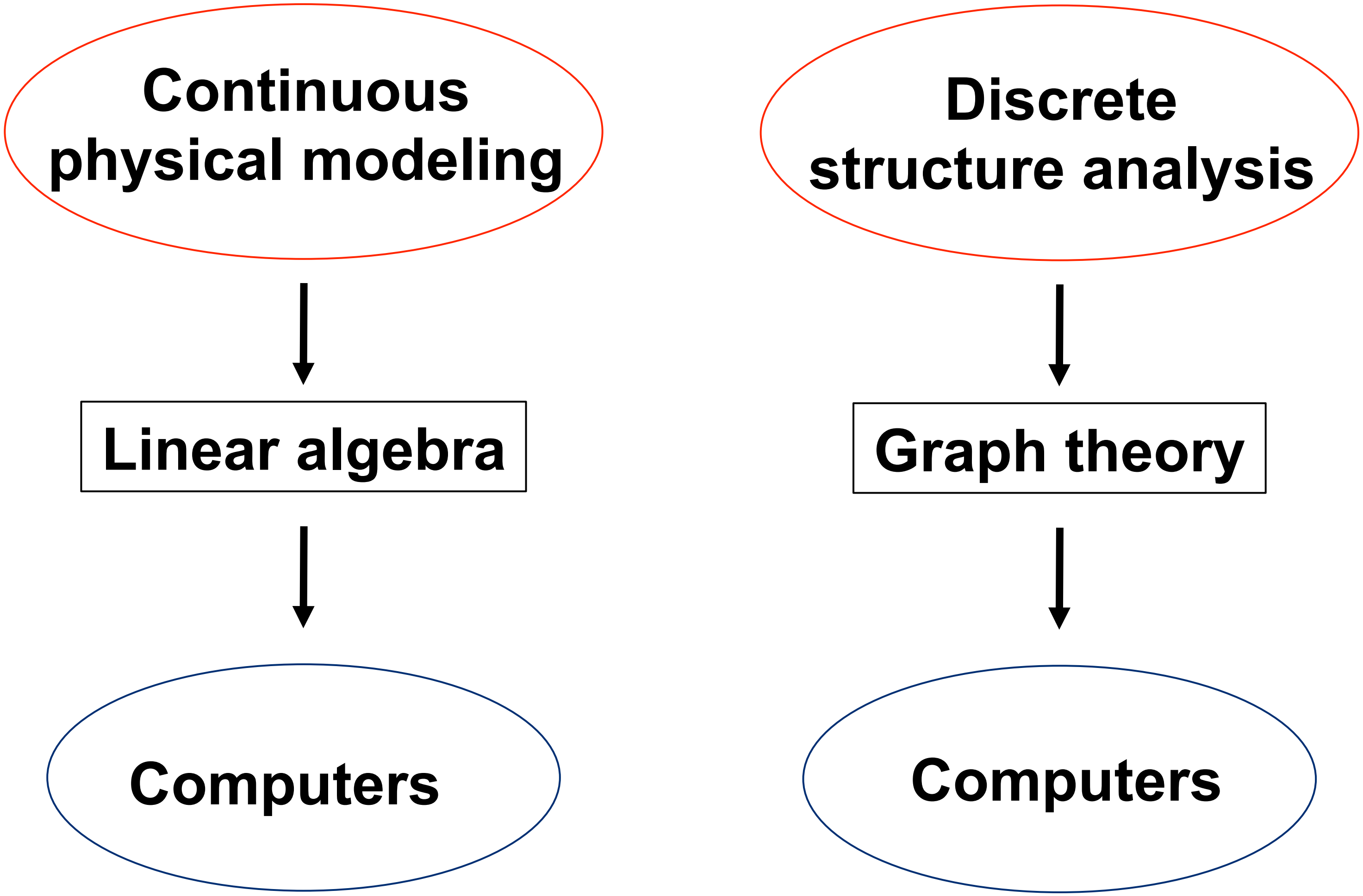


**Graph partitioning:** *Dynamic load balancing* in parallel simulations

Picture (left) credit: Sanders and Schulz

**Problem size:** as big as the sparse linear system to be solved or the simulation to be performed

# Graphs as middleware

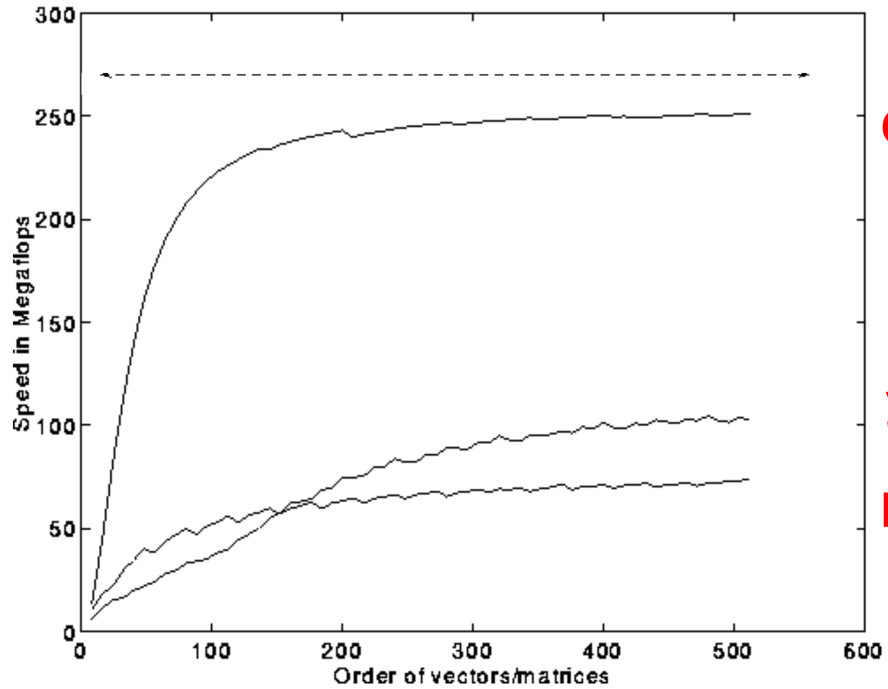


# Graphs as middleware

By analogy to numerical scientific computing. . .

What should the Combinatorial BLAS look like?

**Basic Linear Algebra Subroutines (BLAS): Ops/Sec vs. Matrix Size**




$C = A*B$

$y = A*x$

$\mu = x^T y$

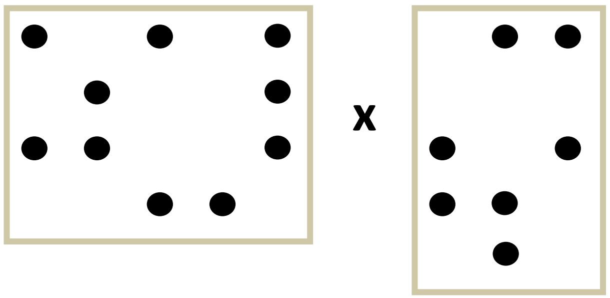
# The case for sparse matrices

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

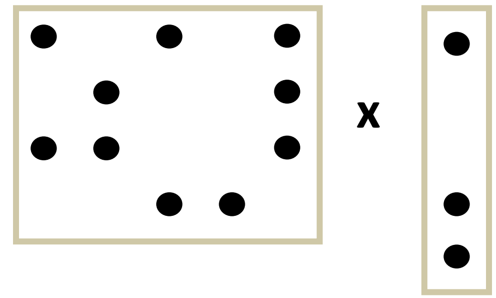
| Traditional graph computations | Graphs in the language of linear algebra |
|---|---|
| Data driven, unpredictable communication. | Fixed communication patterns |
| Irregular and unstructured, poor locality of reference | Operations on matrix blocks exploit memory hierarchy |
| Fine grained data accesses, dominated by latency | Coarse grained parallelism, bandwidth limited |

# Linear-algebraic primitives for graphs

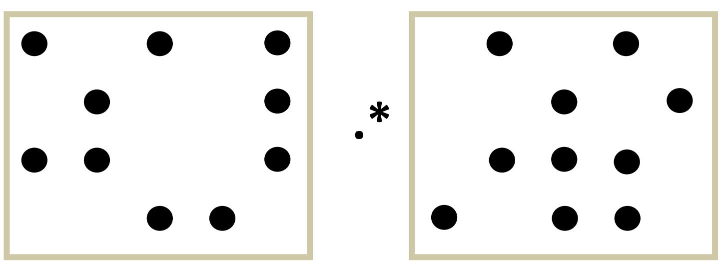
Sparse matrix-sparse matrix multiplication



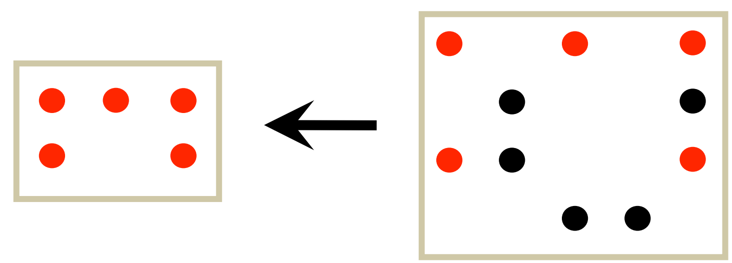

Sparse matrix-sparse vector multiplication




Element-wise operations




Sparse matrix indexing



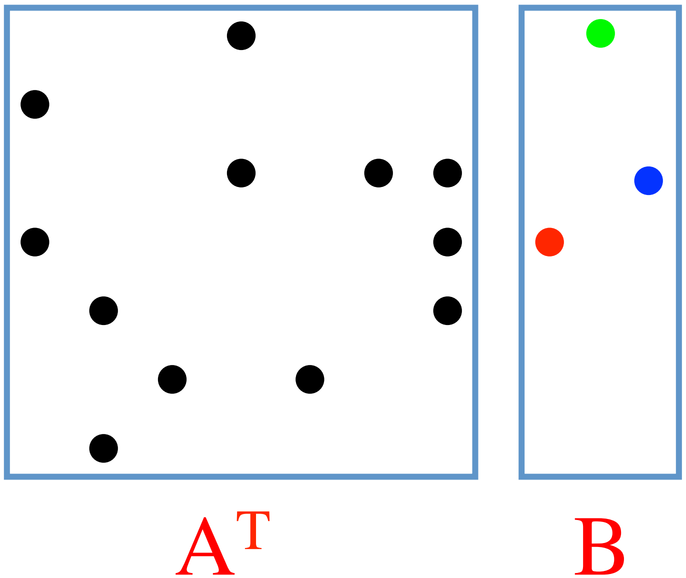
**The Combinatorial BLAS implements these, and more, on arbitrary semirings, e.g. $(\times, +)$, (and, or), $(+, \min)$**

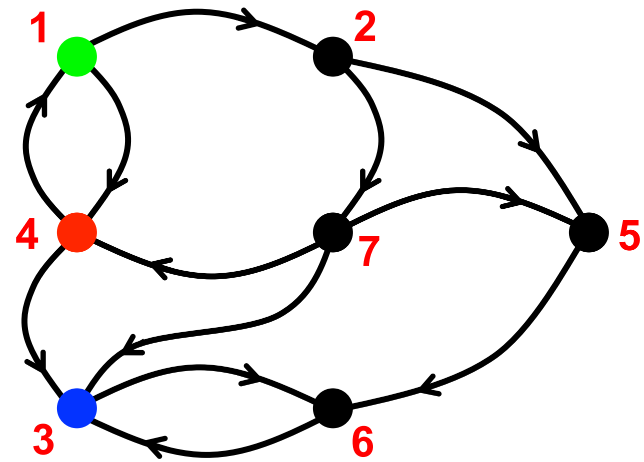
# Examples of semirings in graph algorithms

| | |
|---|---|
| Real field: **(R, +, x)** | Classical numerical linear algebra |
| Boolean algebra: **({0 1}, \|, &)** | Graph traversal |
| Tropical semiring: **(R U {∞}, min, +)** | Shortest paths |
| **(S, select, select)** | Select subgraph, or contract nodes to form quotient graph |
| (edge/vertex attributes, vertex data aggregation, edge data processing) | Schema for user-specified computation at vertices and edges |
| **(R, max, +)** | Graph matching &network alignment |
| **(R, min, times)** | Maximal independent set |

- **Shortened semiring notation: (Set, Add, Multiply)**. Both identities omitted.
- **Add:** Traverses edges, **Multiply:** Combines edges/paths at a vertex
- Neither add nor multiply needs to have an inverse.
- Both **add** and **multiply** are **associative**, **multiply distributes** over **add**

# Multiple-source breadth-first search

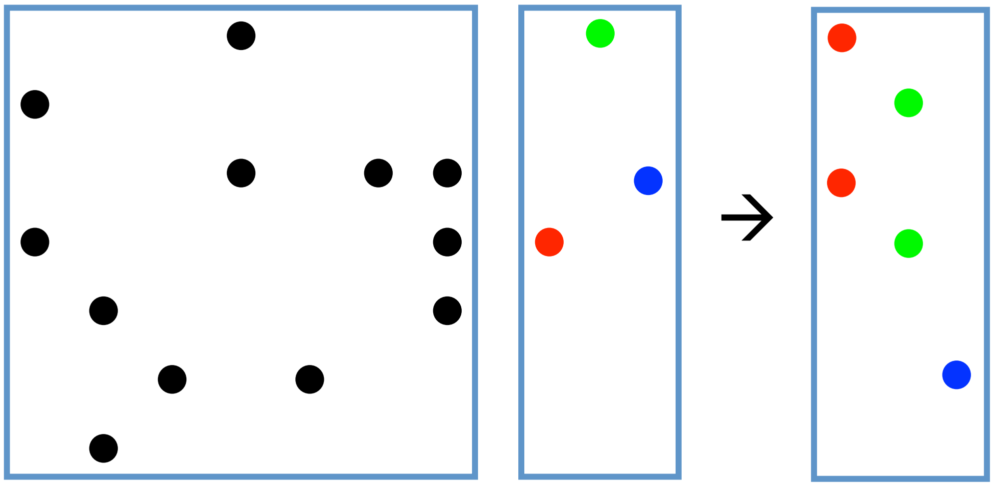


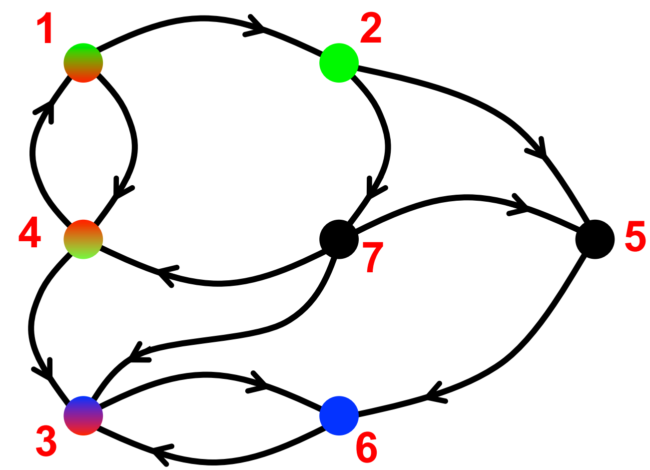

- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Highly-parallel implementation for Betweenness Centrality*

*: A measure of influence in graphs, based on shortest paths

# Multiple-source breadth-first search



$A^T$ $\quad$ $B$ $\quad$ $A^T \cdot B$




- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Highly-parallel implementation for Betweenness Centrality*

*: A measure of influence in graphs, based on shortest paths

# Graph algorithm comparison (LA: linear algebra)

Slide inspiration: Jeremy Kepner (MIT LL)

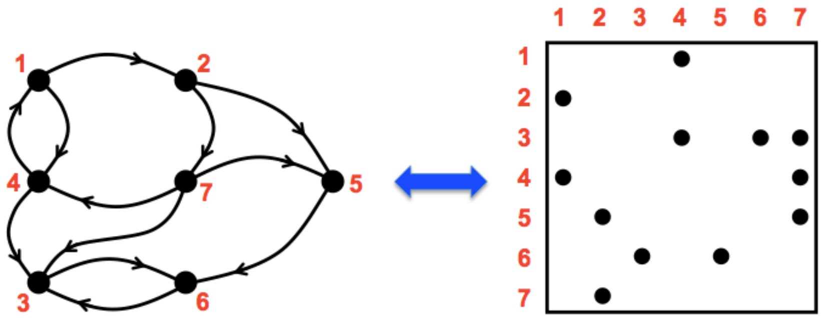
| Algorithm (Problem) | Canonical Complexity | LA-Based Complexity | Critical Path (for LA) |
|---|---|---|---|
| Breadth-first search | $\Theta(m)$ | $\Theta(m)$ | $\Theta(diameter)$ |
| Betweenness Centrality (unweighted) | $\Theta(mn)$ | $\Theta(mn)$ | $\Theta(diameter)$ |
| All-pairs shortest-paths (dense) | $\Theta(n^3)$ | $\Theta(n^3)$ | $\Theta(n)$ |
| Prim (MST) | $\Theta(m+n \log n)$ | $\Theta(n^2)$ | $\Theta(n)$ |
| Borůvka (MST) | $\Theta(m \log n)$ | $\Theta(m \log n)$ | $\Theta(\log^2 n)$ |
| Edmonds-Karp (Max Flow) | $\Theta(m^2 n)$ | $\Theta(m^2 n)$ | $\Theta(mn)$ |
| Greedy MIS (MIS) | $\Theta(m+n \log n)$ | $\Theta(mn+n^2)$ | $\Theta(n)$ |
| Luby (MIS) | $\Theta(m+n \log n)$ | $\Theta(m \log n)$ | $\Theta(\log n)$ |

Majority of selected algorithms can be represented with array-based constructs with equivalent complexity.

($n = |V|$ and $m = |E|$)

# Combinatorial BLAS

http://gauss.cs.ucsb.edu/~aydin/CombBLAS




An extensible distributed-memory library offering a small but powerful set of linear algebraic operations specifically targeting graph analytics.

- Aimed at graph algorithm designers/programmers who are not expert in mapping algorithms to parallel hardware.
- Flexible templated C++ interface; 2D data decomposition
- Scalable performance from laptop to 100,000-processor HPC.
- Open source software (v1.4.0 released January, 2014)

# Matrix times Matrix over semiring

**Inputs**

matrix $\mathbf{A}$: $\mathbb{S}^{MxN}$ (sparse or dense)

matrix $\mathbf{B}$: $\mathbb{S}^{NxL}$ (sparse or dense)

**Optional Inputs**

matrix $\mathbf{C}$: $\mathbb{S}^{MxL}$ (sparse or dense)

scalar "add" function $\oplus$

scalar "multiply" function $\otimes$

transpose flags for $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$

**Outputs**

matrix $\mathbf{C}$: $\mathbb{S}^{MxL}$ (sparse or dense)

**Implements** $\mathbf{C} \oplus= \mathbf{A} \oplus.\otimes \mathbf{B}$

**for** j = 1 : N

$\mathbf{C}(i,k) = \mathbf{C}(i,k) \oplus (\mathbf{A}(i,j) \otimes \mathbf{B}(j,k))$

If input $\mathbf{C}$ is omitted, implements

$\mathbf{C} = \mathbf{A} \oplus.\otimes \mathbf{B}$

Transpose flags specify operation on $\mathbf{A}^T$, $\mathbf{B}^T$, and/or $\mathbf{C}^T$ instead

**Notes**

$\mathbb{S}$ is the set of scalars, user-specified

$\mathbb{S}$ defaults to IEEE double float

$\oplus$ defaults to floating-point +

$\otimes$ defaults to floating-point *

**Specific cases and function names:**

SpGEMM: sparse matrix times sparse matrix

SpMSpV: sparse matrix times sparse vector

SpMV: Sparse matrix times dense vector

SpMM: Sparse matrix times dense matrix

# Can we standardize a “Graph BLAS”?

**No**, it’s not reasonable to define a universal set of building blocks.

Huge diversity in matching graph algorithms to hardware platforms.
No consensus on data structures or linguistic primitives.
Lots of graph algorithms remain to be discovered.
Early standardization can inhibit innovation.

**Yes**, it *is* reasonable to define a common set of building blocks…
… for graphs as linear algebra.

Representing graphs in the language of linear algebra is a mature field.
Algorithms, high level interfaces, and implementations vary.
But the core primitives are well established.

# The Graph BLAS effort

## Standards for Graph Algorithm Primitives

Tim Mattson (Intel Corporation), David Bader (Georgia Institute of Technology), Jon Berry (Sandia National Laboratory), Aydin Buluc (Lawrence Berkeley National Laboratory), Jack Dongarra (University of Tennessee), Christos Faloutsos (Carnegie Melon University), John Feo (Pacific Northwest National Laboratory), John Gilbert (University of California at Santa Barbara), Joseph Gonzalez (University of California at Berkeley), Bruce Hendrickson (Sandia National Laboratory), Jeremy Kepner (Massachusetts Institute of Technology), Charles Leiserson (Massachusetts Institute of Technology), Andrew Lumsdaine (Indiana University), David Padua (University of Illinois at Urbana-Champaign), Stephen Poole (Oak Ridge National Laboratory), Steve Reinhardt (Cray Corporation), Mike Stonebraker (Massachusetts Institute of Technology), Steve Wallach (Convey Corporation), Andrew Yoo (Lawrence Livermore National Laboratory)

*Abstract*-- It is our view that the state of the art in constructing a large collection of graph algorithms in terms of linear algebraic operations is mature enough to support the emergence of a standard set of primitive building blocks. This paper is a position paper defining the problem and announcing our intention to launch an open effort to define this standard.

- The Graph BLAS Forum: http://istc-bigdata.org/GraphBlas/
- Graph Algorithms Building Blocks (GABB workshop at IPDPS'14): http://www.graphanalysis.org/workshop2014.html

# Challenges at Exascale

*"New algorithms need to be developed that identify and leverage more concurrency and that reduce synchronization and communication"* - ASCR Applied Mathematics Research for Exascale Computing Report

***High-performance** requirement is the invariant for {any}scale*

**Challenges specific to Exascale and beyond:**

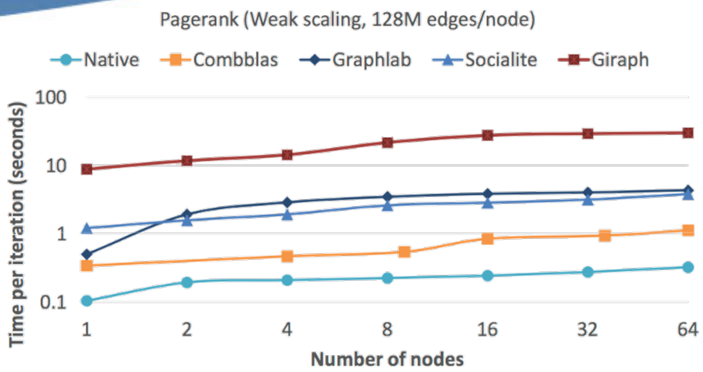
- Power/Energy
- Data Locality
- Extreme Concurrency/Parallelism
- Resilience/ Fault tolerance

# Performance of Linear Algebraic Graph Algorithms

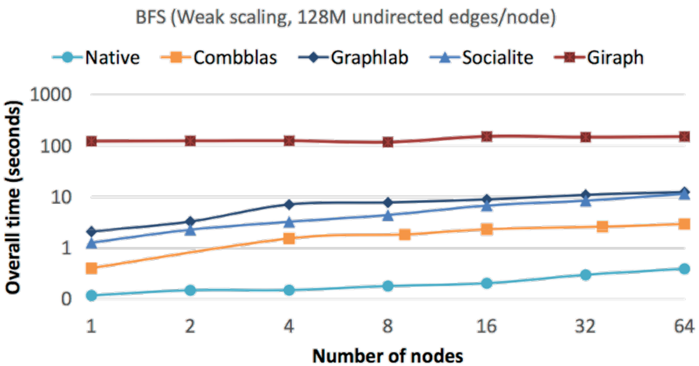
*Combinatorial BLAS fastest among all tested graph processing frameworks on 3 out of 4 benchmarks in an independent study by Intel.*

*The linear algebra abstraction enables high performance, within 4X of native performance for PageRank and Collaborative filtering.*

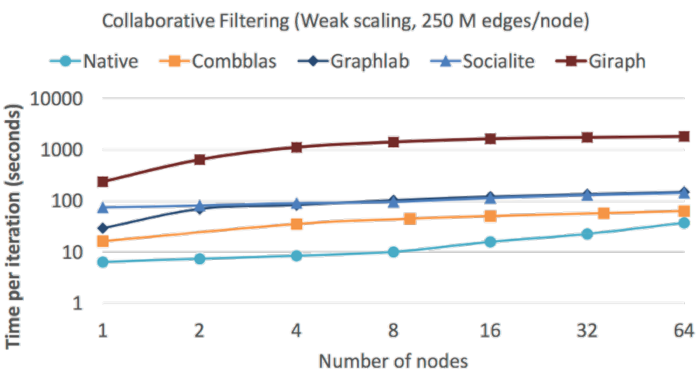
Satish, Nadathur, et al. "Navigating the Maze of Graph Analytics Frameworks using Massive Graph Datasets", in SIGMOD'14




(a) PageRank



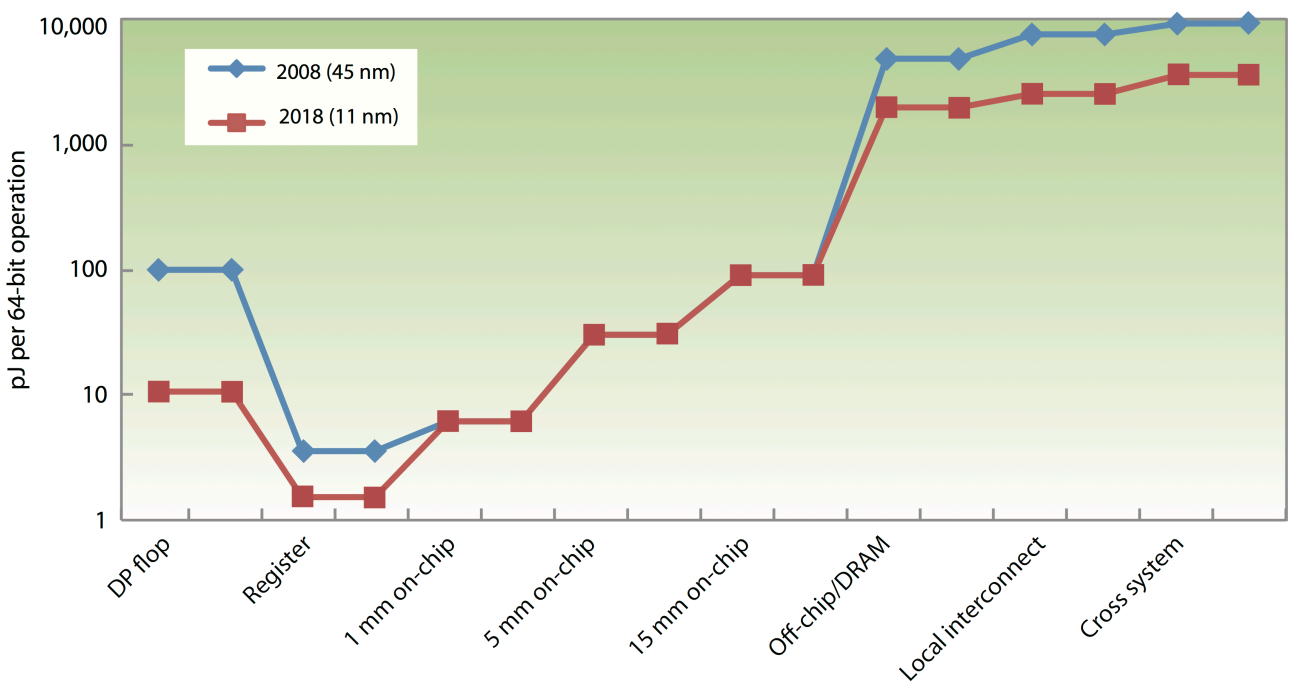

(b) Breadth-First Search




(c) Collaborative Filtering

# Energy and data locality challenges

"Data movement is overtaking computation as the most dominant cost of a system both in terms of dollars and in terms of energy consumption. Consequently, we should be more explicit about reasoning about data movement."




Data movement (communication) costs **energy**

Image courtesy of Kogge & Shalf

Exascale Computing Trends: Adjusting to the "New Normal" for Computer Architecture
Kogge, Peter and Shalf, John, Computing in Science & Engineering, 15, 16-26 (2013)

# Communication-avoidance motivation

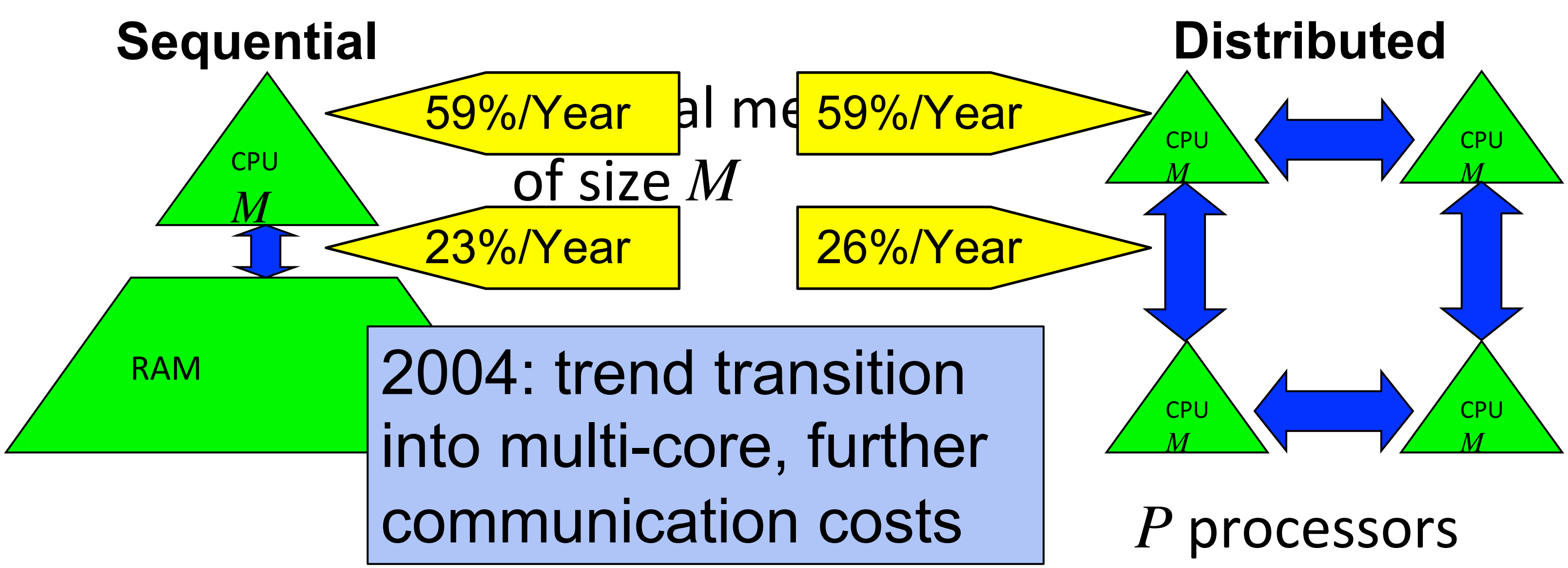
Two kinds of costs:

Arithmetic (FLOPs)

Communication: moving data

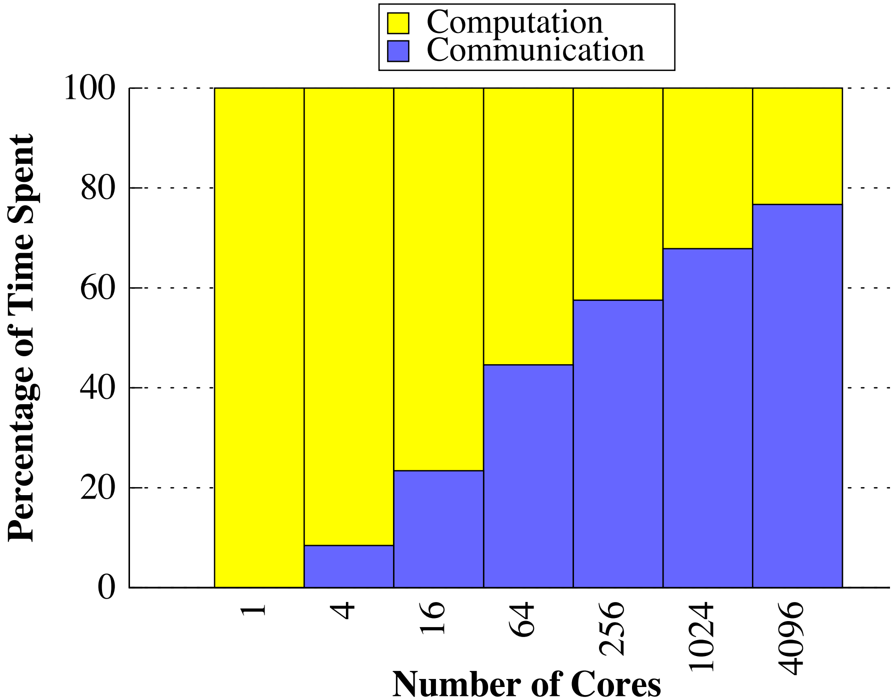
Develop faster algorithms: minimize communication (to lower bound if possible)

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



# 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

# Reduced Communication Graph Algorithms

**Communication avoiding approaches in linear algebra:**

[A] Exploiting extra available memory (2.5D algorithms)

- typically applies to matrix-matrix operations

[B] Communicating once every k steps (k-step Krylov methods)

- typically applies to iterative sparse methods

**Good news:** We successfully generalized A to sparse matrix-matrix multiplication (graph contraction, multi-source BFS, clustering, etc.) and all pairs shortest paths (Isomap).

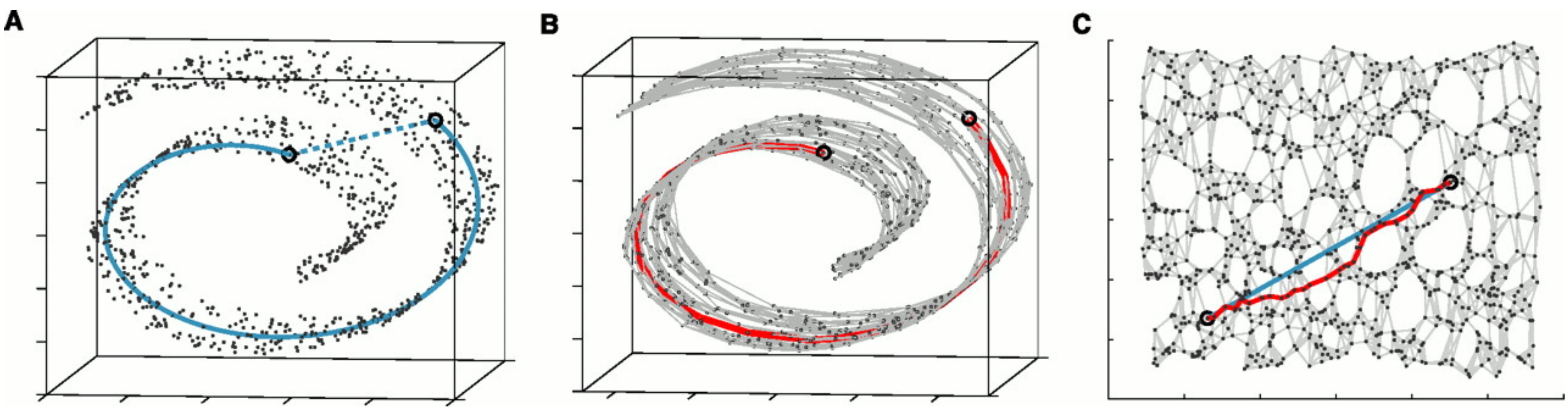
**Unknown:** if B can be applied to iterative graph algorithms.

# Manifold Learning

**Isomap (Nonlinear dimensionality reduction):** Preserves the intrinsic geometry of the data by using the geodesic distances on manifold between all pairs of points

**Tools used or desired:**

- K-nearest neighbors
- *All pairs shortest paths (APSP)*
- Top-k eigenvalues

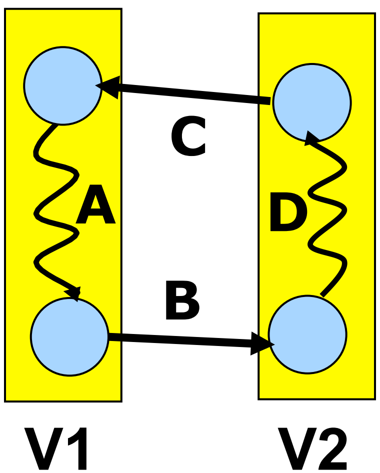



Tenenbaum, Joshua B., Vin De Silva, and John C. Langford. "A global geometric framework for nonlinear dimensionality reduction." Science 290.5500 (2000): 2319-2323.

# All pairs shortest paths at Scale

**R-Kleene:** A recursive APSP algorithm that is rich in semiring matrix-matrix multiplications

**+** is "min", **×** is "add"




| | |
|---|---|
| A | B |
| C | D |

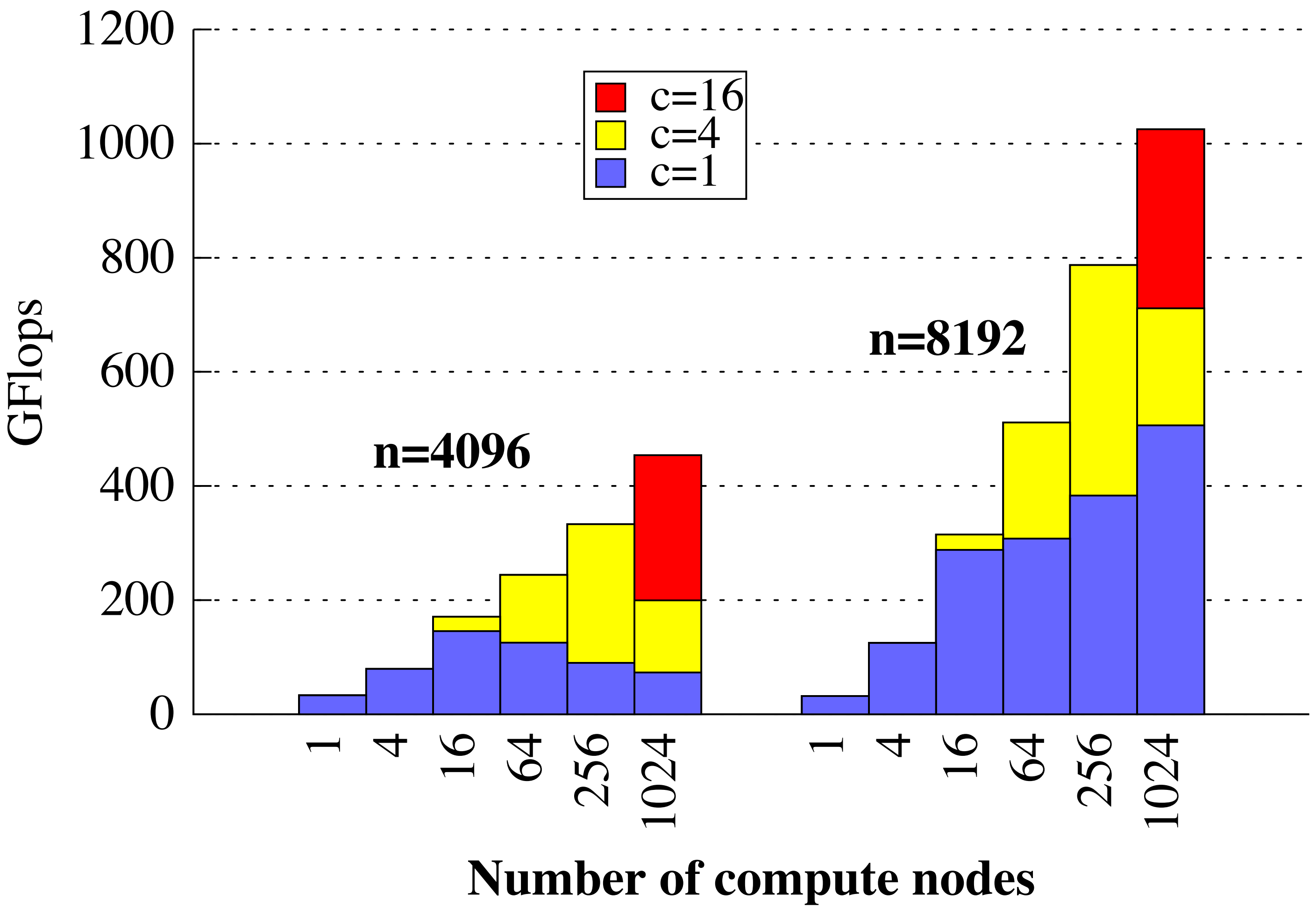
```
A = A*;      % recursive call
B = AB;  C = CA;
D = D + CB;
D = D*;      % recursive call
B = BD;  C = DC;
A = A + BC;
```

Using the "right" recursion and 2.5D communication avoiding matrix multiplication (with c replicas):

$$\text{Bandwidth}=O\left(n^2/\sqrt{cp}\right) \qquad \text{Latency}=O\left(\sqrt{cp}\,\log^2(p)\right)$$

# Communication-avoiding APSP on distributed memory

**65K vertex dense problem solved in about two minutes**




**c**: replication in 2.5D semiring matrix multiply

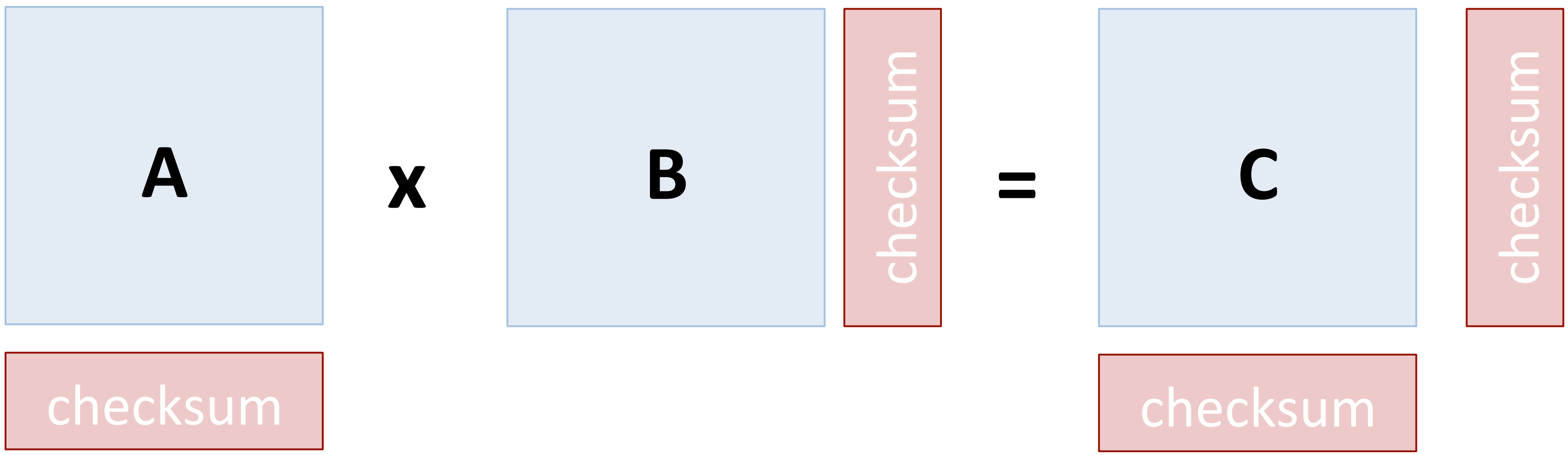
Jaguar (Titan without GPUs) at ORNL.

Nodes have 24 cores each

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

# Fault Tolerance of Linear Algebraic Graph Algorithms

Literature exists on fault tolerant linear algebra operations.
**Overhead:** O(dN) to detect/correct O(d) errors on N-by-N matrices

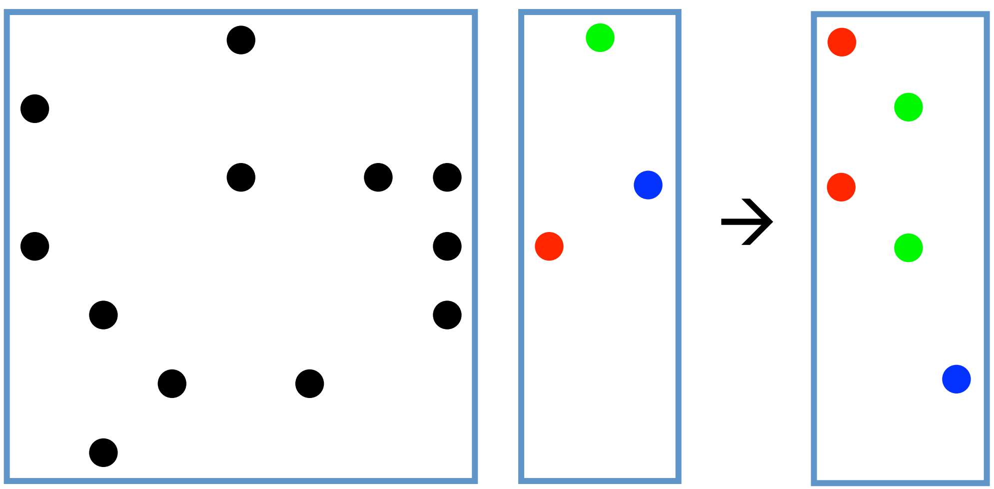



**Good news:** Overhead can often be tolerated in sparse matrix-matrix operations for graph algorithms.

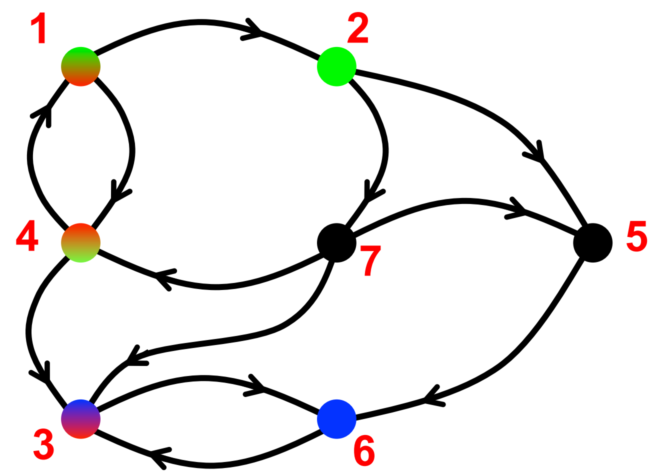
**Unknown:** Techniques are for fields/rings, how do they apply to semiring algebra?

# Extreme Concurrency/Parallelism

Linear algebra is the right abstraction for exploiting multiple levels of parallelism available in many graph algorithms



$A^T$ $\quad$ $B$ $\quad$ $A^T \cdot B$




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

**B.** and J.R. Gilbert. The Combinatorial BLAS: Design, implementation, and applications. IJHPCA, 2011.

# Conclusions

- We believe the state of the art for "Graphs in the language of Linear algebra" is mature enough to define a common set of building blocks
- Linear algebra did not solve its exascale challenges yet, but it is not clueless either.
- All other graph abstractions (think like a vertex, gather-apply-scatter, visitors) are clueless/ignorant about addressing exascale challenges.
- If graph algorithms ever scales to exascale, it will most likely be in the language of linear algebra.
- Come join our next event at HPEC'14

# Acknowledgments


- Ariful Azad (LBL)
- 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)
- Sang-Yuh Oh (LBL)
- Lenny Oliker (Berkeley Lab)
- Steve Reinhardt (Cray)
- Dan Rokhsar (JGI/UCB)
- Oded Schwartz (UCB)
- Harsha Simhadri (LBL)
- Edgar Solomonik (UCB)
- Veronika Strnadova (UCSB)
- Sivan Toledo (Tel Aviv Univ)
- Dani Ushizima (Berkeley Lab)
- Kathy Yelick (Berkeley Lab/UCB)

**My work is funded by:**

U.S. DEPARTMENT OF ENERGY | Office of Science