Sparse Matrices for High-Performance Graph Analytics

John R. Gilbert
University of California, Santa Barbara

Oak Ridge National Laboratory
October 3, 2014

Support: Intel, Microsoft, DOE Office of Science, NSF
Thanks …

Lucas Bang (UCSB), Jon Berry (Sandia), Eric Boman (Sandia), Aydin Buluc (LBL), John Conroy (CCS), Kevin Deweese (UCSB), Erika Duriakova (Dublin), Armando Fox (UCB), Shoaib Kamil (MIT), Jeremy Kepner (MIT), Tristan Konolige (UCSB), Adam Lugowski (UCSB), Tim Mattson (Intel), Brad McRae (TNC), Dave Mizell (YarcData), Lenny Oliker (LBL), Carey Priebe (JHU), Steve Reinhardt (YarcData), Lijie Ren (Google), Eric Robinson (Lincoln), Viral Shah (UIDAI), Veronika Strnadova (UCSB), Yun Teng (UCSB), Joshua Vogelstein (Duke), Drew Waranis (UCSB), Sam Williams (LBL)
Outline

- Motivation: Graph applications
- Mathematics: Sparse matrices for graph algorithms
- Software: CombBLAS, QuadMat, KDT
- Standards: Graph BLAS Forum
Computational models of the physical world

Cortical bone

Trabecular bone
Large graphs are everywhere...

- Internet structure
- Social interactions
- Scientific datasets: biological, chemical, cosmological, ecological, ...

WWW snapshot, courtesy Y. Hyun

Yeast protein interaction network, courtesy H. Jeong
Social network analysis (1993)

Co-author graph from 1993 Householder symposium
Social network analysis (2014)

Facebook graph:
> 1,000,000,000 vertices
Large-scale genomic mapping and sequencing
[Strnadova, Buluc, Chapman, G, Gonzalez, Jegelska, Rokhsar, Oliker 2014]

- Problem: scale to millions of markers times thousands of individuals, with “unknown” rates > 50%
- Tools used or desired: spanning trees, approximate TSP, incremental connected components, spectral and custom clustering, k-nearest neighbors
- Results: using more data gives better genomic maps
Alignment and matching of brain scans
[Conroy, G, Kratzer, Lyzinski, Priebe, Vogelstein 2014]

- Problem: match functional regions across individuals
- Tools: Laplacian eigenvectors, geometric spectral partitioning, clustering, and more. . .
Landscape connectivity modeling

[McRae et al.]

- Habitat quality, gene flow, corridor identification, conservation planning
- Targeting larger problems: Yellowstone-to-Yukon corridor
- Tools: Graph contraction, connected components, Laplacian linear systems

Figures courtesy of Brad McRae, NCEAS
Combinatorial acceleration of Laplacian solvers
[Boman, Deweese, G 2014]

\[(B^{-1/2} A B^{-1/2}) (B^{1/2} x) = B^{-1/2} b\]

- Problem: approximate target graph by sparse subgraph
- \( Ax = b \) in nearly linear time in theory [ST08, KMP10, KOSZ13]
- Tools: spanning trees, subgraph extraction and contraction, breadth-first search, shortest paths, . . .
The middleware challenge for graph analysis

Continuous physical modeling

Linear algebra

Computers

Discrete structure analysis

Graph theory

Computers
The middleware challenge for graph analysis

• By analogy to numerical scientific computing...

• What should the combinatorial BLAS look like?

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

$c = a \times b$
$y = a \times x$
$\mu = x^T \times y$
Sparse array primitives for graph manipulation

Sparse matrix-matrix multiplication (SpGEMM)

Element-wise operations

Sparse matrix-dense vector multiplication

Sparse matrix indexing

Matrices over various semirings: \((+ \cdot x), (\min \cdot +), (\text{or} \cdot \text{and}), \ldots\)
Multiple-source breadth-first search

\[ A^T \quad X \]
Multiple-source breadth-first search

- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
Examples of semirings in graph algorithms

<table>
<thead>
<tr>
<th>(edge/vertex attributes, vertex data aggregation, edge data processing)</th>
<th>Schema for user-specified computation at vertices and edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real field: ((\mathbb{R}, +, \ast))</td>
<td>Numerical linear algebra</td>
</tr>
<tr>
<td>Boolean algebra: (({0, 1},</td>
<td>, &amp;))</td>
</tr>
<tr>
<td>Tropical semiring: ((\mathbb{R} \cup {\infty}, \text{min}, +))</td>
<td>Shortest paths</td>
</tr>
<tr>
<td>((\mathbb{S}, \text{select}, \text{select}))</td>
<td>Select subgraph, or contract nodes to form quotient graph</td>
</tr>
</tbody>
</table>
Graph contraction via sparse triple product

A1

A2

A3

Contract

A1

A2

A3

1 2 3 4 5 6

1 1 1 1 1 1

1 2 3 4 5 6

X

1 1 1 1

1 1 1 1

X

=
Subgraph extraction via sparse triple product

Extract

\[
\begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{pmatrix}
\times
\begin{pmatrix}
\bullet & & \\
& \bullet & \\
& & \bullet \\
\end{pmatrix}
\times
\begin{pmatrix}
1 & 1 \\
1 & 1 \\
\end{pmatrix}
= 
\begin{pmatrix}
\bullet & \bullet & \\
& \bullet & \\
& & \bullet \\
\end{pmatrix}
\]
Counting triangles (clustering coefficient)

A

3 5 6 4

1 2

Clustering coefficient:

• Pr (wedge i-j-k makes a triangle with edge i-k)
• $3 \times \frac{\# \text{triangles}}{\# \text{wedges}}$
• $3 \times \frac{4}{19} = 0.63$ in example
• may want to compute for each vertex j
Counting triangles (clustering coefficient)

- Count triangles by lowest-degree vertex.

- Enumerate “low-hinged” wedges.

- Keep wedges that close.

Clustering coefficient:
- \( \Pr (\text{wedge } i-j-k \text{ makes a triangle with edge } i-k) \)
- \( 3 \times \frac{\# \text{ triangles}}{\# \text{ wedges}} \)
- \( 3 \times \frac{4}{19} = 0.63 \) in example
- may want to compute for each vertex \( j \)

Cohen’s algorithm to count triangles:
Counting triangles (clustering coefficient)

\[ A = L + U \quad \text{(hi->lo + lo->hi)} \]
\[ L \times U = B \quad \text{(wedge, low hinge)} \]
\[ A \land B = C \quad \text{(closed wedge)} \]
\[ \text{sum}(C)/2 = 4 \text{ triangles} \]
A few other graph algorithms we’ve implemented in linear algebraic style

• Maximal independent set (KDT/SEJITS) [BDFGKLOW 2013]

• Peer-pressure clustering (SPARQL) [DGLMR 2013]

• Time-dependent shortest paths (CombBLAS) [Ren 2012]

• Gaussian belief propagation (KDT) [LABGRTW 2011]

• Markoff clustering (CombBLAS, KDT) [BG 2011, LABGRTW 2011]

• Betweenness centrality (CombBLAS) [BG 2011]

• Hybrid BFS/bully connected components (CombBLAS) [Konolige, in progress]

• Geometric mesh partitioning (Matlab 😊) [GMT 1998]
Graph algorithms in the language of linear algebra

- Kepner et al. study [2006]: fundamental graph algorithms including min spanning tree, shortest paths, independent set, max flow, clustering, …
- SSCA#2 / centrality [2008]
- Basic breadth-first search / Graph500 [2010]
- Beamer et al. [2013] direction-optimizing breadth-first search, implemented in CombBLAS
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.
- Scalable performance from laptop to 100,000-processor HPC.
- Open source software.
- Version 1.4.0 released January 2014.
Combinatorial BLAS in distributed memory

- **DenseDistMat**
- **SpDistMat**
- **SpMat**
- **SpDistVec**
- **DenseDistVec**
- **CommGrid**

- **DistMat**
- **FullyDistVec**

- **Enforces interface only**

- **Polymorphism**

**Combinatorial BLAS functions and operators**
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
Satish et al., *Navigating the Maze of Graph Analytics Frameworks Using Massive Graph Datasets*, SIGMOD 2014.

Combinatorial BLAS was fastest of all tested graph processing frameworks on 3 out of 4 benchmarks.
Combinatorial BLAS “users” (Sep 2014)

- IBM (T.J.Watson, Zurich, & Tokyo)
- Intel
- Cray
- Microsoft
- Stanford
- UC Berkeley
- Carnegie-Mellon
- Georgia Tech
- Ohio State
- U Texas Austin
- Columbia
- U Minnesota
- NC State
- UC San Diego
- Sandia Labs
- SEI
- Paradigm4
- IHPC (Singapore)
- King Fahd U (Saudi Arabia)
- Tokyo Inst of Technology
- Chinese Academy of Sciences
- U Ghent (Belgium)
- Bilkent U (Turkey)
- U Canterbury (New Zealand)
- Purdue
- Indiana U
- UC Merced
- Mississippi State
QuadMat shared-memory data structure

[Lugowski, G]

Blocks store enough matrix elements for meaningful computation; denser parts of matrix have more blocks.

subdivide by dimension on power of 2 indices
- **Problem**: Natural recursive matrix multiplication is inefficient due to deep tree of sparse matrix additions.

- **Solution**: Rearrange into block inner product *pair lists*.

- A single matrix element can participate in pair lists with different block sizes.

- Symbolic phase followed by computational phase

- Multithreaded implementation in Intel TBB
QuadMat compared to Csparse & CombBLAS
Knowledge Discovery Toolbox
http://kdt.sourceforge.net/

- Aimed at domain experts who know their problem well but don’t know how to program a supercomputer
- Easy-to-use Python interface
- Runs on a laptop as well as a cluster with 10,000 processors
- Open source software (New BSD license)

A general graph library with operations based on linear algebraic primitives
Attributed semantic graphs and filters

Example:
- Vertex types: Person, Phone, Camera, Gene, Pathway
- Edge types: PhoneCall, TextMessage, CoLocation, SequenceSimilarity
- Edge attributes: Time, Duration
- Calculate centrality just for emails among engineers sent between given start and end times

```python
def onlyEngineers (self):
    return self.position == Engineer

def timedEmail (self, sTime, eTime):
    return ((self.type == email) and
            (self.Time > sTime) and
            (self.Time < eTime))

G.addVFilter(onlyEngineers)
G.addEFilter(timedEmail(start, end))

# rank via centrality based on recent email transactions among engineers
bc = G.rank('approxBC')
```
SEJITS for filter/semiring acceleration

- **Standard KDT**
  - KDT Algorithm
  - Filter (Py)
  - Semiring (Py)

- **KDT+SEJITS**
  - KDT Algorithm
  - Filter (C++)
  - Semiring (C++)

**Embedded DSL: Python for the whole application**
- Introspect, translate Python to equivalent C++ code
- Call compiled/optimized C++ instead of Python
Filtered BFS with SEJITS

Time (in seconds) for a single BFS iteration on scale 25 RMAT (33M vertices, 500M edges) with 10% of elements passing filter. Machine is NERSC’s Hopper.
The Basic Linear Algebra Subroutines (BLAS) had a revolutionary impact on computational linear algebra.

<table>
<thead>
<tr>
<th>BLAS 1</th>
<th>vector ops</th>
<th>Lawson, Hanson, Kincaid, Krogh, 1979</th>
<th>LINPACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAS 2</td>
<td>matrix-vector ops</td>
<td>Dongarra, Du Croz, Hammarling, Hanson, 1988</td>
<td>LINPACK on vector machines</td>
</tr>
<tr>
<td>BLAS 3</td>
<td>matrix-matrix ops</td>
<td>Dongarra, Du Croz, Hammarling, Hanson, 1990</td>
<td>LAPACK on cache based machines</td>
</tr>
</tbody>
</table>

- Experts in mapping algorithms to hardware tune BLAS for specific platforms.
- Experts in numerical linear algebra build software on top of the BLAS to get high performance “for free.”

Today every computer, phone, etc. comes with `/usr/lib/libblas`
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.
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.

Graph BLAS Forum: http://istc-bigdata.org/GraphBlas/
## Matrix times matrix over semiring

### Inputs
- **matrix** $A$: $S^{M \times N}$ (sparse or dense)
- **matrix** $B$: $S^{N \times L}$ (sparse or dense)

### Optional Inputs
- **matrix** $C$: $S^{M \times L}$ (sparse or dense)
- scalar “add” function $\oplus$
- scalar “multiply” function $\otimes$
- transpose flags for $A$, $B$, $C$

### Outputs
- **matrix** $C$: $S^{M \times L}$ (sparse or dense)

### Implements

$$C \oplus = A \oplus . \otimes B$$

for $j = 1 : N$

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

If input $C$ is omitted, implements

$$C = A \oplus . \otimes B$$

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

### 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

### Notes
- $S$ is the set of scalars, user-specified
- $S$ defaults to IEEE double float
- $\oplus$ defaults to floating-point $+$
- $\otimes$ defaults to floating-point $*$
Conclusion

• Matrix computation is beginning to repay a 50-year debt to graph algorithms.

• Graphs in the language of linear algebra are sufficiently mature to support a standard set of BLAS.

• It helps to look at things from two directions.