GraphBLAS: A Programming Specification for Graph Analysis
Scott McMillan, PhD
GraphBLAS: A Programming Specification for Graph Analysis
Scott McMillan, PhD

In collaboration with:
MIT/LL, LBNL, Intel, IBM, UC Davis, Indiana, and many others.
Summary

Graph analysis is *important* and *pervasive* in the DoD community.

**GraphBLAS Forum (a world-wide consortium of researchers)**

- Government/FFRDCs, academia, and industry
- Goal: application programming specification (API) for graph analysis
- GraphBLAS: Graph Basic Linear Algebra Subprograms

**SEI contributions**

- Member of the Graph BLAS Forum
- Member of the C API Specification committee
- Early implementation of a C++ library: GraphBLAS Template Library (open-source)
Graph Analysis is **Important and Pervasive**

---

**ISR**
- Graphs represent entities and relationships detected through multi-INT sources
- 1,000s – 1,000,000s tracks and locations
- GOAL: Identify anomalous patterns of life

**Social**
- Graphs represent relationships between individuals or documents
- 10,000s – 10,000,000s individual and interactions
- GOAL: Identify hidden social networks

**Cyber**
- Graphs represent communication patterns of computers on a network
- 1,000,000s – 1,000,000,000s network events
- GOAL: Identify cyber attacks or malicious software

---

**Common Goal:** Detection of subtle patterns in massive graphs

---

Common Operations

- Finding Neighbors
- Shortest Paths
- Clustering
- “Important” nodes
  - Influencers
  - Bottlenecks
  - Outliers

Graphs as Matrices

Graphs are represented as adjacency matrices that have irregular and sparse structure.
Graph Operations as Matrix Operations

- Matrix multiply $\rightarrow$ find neighbors (most important primitive)
- Used in breadth-first traversal, shortest paths, and many others
- Sparsity and irregularity of matrix structure is a barrier to high performance
Today’s Computing Landscape

Intel Xeon E5-2699v3
662 Gflop/s, 145 W
18 cores, 2.3 GHz
4-way/8-way AVX2

IBM POWER8
384 Gflop/s, 200 W
12 cores, 4 GHz
2-way/4-way VMX/VSX

NVIDIA Tesla P100
10.6 Tflop/s, 250 W
3584 cores, 1.48 GHz
64-way SIMT

Intel Xeon Phi
1.2 Tflop/s, 300 W
61 cores, 1.24 GHz
8-way/16-way LRBni

Qualcomm Snapdragon 810
10 Gflop/s, 2 W
4 cores, 2.5 GHz
A330 GPU, V50 DSP, NEON

Intel Atom C2750
29 Gflop/s, 20 W
8 cores, 2.4 GHz
2-way/4-way SSSE3

Dell PowerEdge R920
1.34 Tflop/s, 850 W
4x 15 cores, 2.8 GHz
4-way/8-way AVX

IBM BlueGene/Q
10 Pflop/s, 8 MW
48k x 16 cores, 1.6 GHz
4-way QPX

Separation of Concerns

Separate the complexity of graph analysis from the complexity of hardware systems:
“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.”

GraphBLAS Forum

<table>
<thead>
<tr>
<th>FFRDCs</th>
<th>Industry</th>
<th>Academia</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIT/Lincoln Labs*</td>
<td>Intel*†</td>
<td>UC Santa Barbara*, Davis †, Berkeley</td>
</tr>
<tr>
<td>Lawrence Berkeley NL*†</td>
<td>IBM†</td>
<td>Georgia Tech*</td>
</tr>
<tr>
<td>CMU/Software Engineering Institute †</td>
<td>NVIDIA</td>
<td>Karlsruhe (KIT)*</td>
</tr>
<tr>
<td>Pacific Northwest NL</td>
<td>Hauwei</td>
<td>CMU</td>
</tr>
<tr>
<td>Sandia NL</td>
<td>Reservoir Labs</td>
<td>Indiana U.</td>
</tr>
<tr>
<td></td>
<td>Galois</td>
<td>MIT</td>
</tr>
<tr>
<td></td>
<td>Mellanox</td>
<td>U. Washington</td>
</tr>
<tr>
<td></td>
<td>and others</td>
<td>and others</td>
</tr>
</tbody>
</table>

* - steering committee, † - C API subcommittee
Separation of Concerns

GOAL: write once, run everywhere (with help from hardware experts).
## GraphBLAS Primitives

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mxm, mxv, vxm</td>
<td>Perform matrix multiplication (e.g., breadth-first traversal)</td>
</tr>
<tr>
<td>eWiseAdd,</td>
<td>Element-wise addition and multiplication of matrices (e.g., graph union,</td>
</tr>
<tr>
<td>eWiseMult</td>
<td>intersection)</td>
</tr>
<tr>
<td>extract</td>
<td>Extract a sub-matrix from a larger matrix (e.g., sub-graph selection)</td>
</tr>
<tr>
<td>assign</td>
<td>Assign to a sub-matrix of a larger matrix (e.g., sub-graph assignment)</td>
</tr>
<tr>
<td>apply</td>
<td>Apply unary function to each element of matrix (e.g., edge weight modification)</td>
</tr>
<tr>
<td>reduce</td>
<td>Reduce along columns or rows of matrices (vertex degree)</td>
</tr>
<tr>
<td>transpose</td>
<td>Swaps the rows and columns of a sparse matrix (e.g., reverse directed edges)</td>
</tr>
<tr>
<td>buildMatrix</td>
<td>Build an matrix representation from row, column, value tuples</td>
</tr>
<tr>
<td>extractTuples</td>
<td>Extract the row, column, value tuples from a matrix representation</td>
</tr>
</tbody>
</table>
GraphBLAS Primitives: The Math

<table>
<thead>
<tr>
<th>Operation</th>
<th>Mathematical Description</th>
<th>Outputs</th>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>mxm</td>
<td>$C(\neg M) \oplus = A^T \odot \otimes B^T$</td>
<td>C</td>
<td>$\neg, M, \oplus, A, T, \odot \otimes, B, T$</td>
</tr>
<tr>
<td>mxv (vxm)</td>
<td>$c(\neg m) \oplus = A^T \odot \otimes b$</td>
<td>c</td>
<td>$\neg, m, \oplus, A, T, \odot \otimes, b$</td>
</tr>
<tr>
<td>eWiseMult</td>
<td>$C(\neg M) \oplus = A^T \otimes B^T$</td>
<td>C</td>
<td>$\neg, M, \oplus, A, T, \otimes, B, T$</td>
</tr>
<tr>
<td>eWiseAdd</td>
<td>$C(\neg M) \oplus = A^T \otimes B^T$</td>
<td>C</td>
<td>$\neg, M, \oplus, A, T, \otimes, B, T$</td>
</tr>
<tr>
<td>reduce (row)</td>
<td>$c(\neg m) \oplus = \oplus_j A^T(:,j)$</td>
<td>c</td>
<td>$\neg, m, \oplus, A, T, \otimes$</td>
</tr>
<tr>
<td>apply</td>
<td>$C(\neg M) \oplus = f(A^T)$</td>
<td>C</td>
<td>$\neg, M, \oplus, A, T, f$</td>
</tr>
<tr>
<td>transpose</td>
<td>$C(\neg M) \oplus = A^T$</td>
<td>C</td>
<td>$\neg, M, \oplus, A (T)$</td>
</tr>
<tr>
<td>extract</td>
<td>$C(\neg M) \oplus = A^T(i,j)$</td>
<td>C</td>
<td>$\neg, M, \oplus, A, T, i, j$</td>
</tr>
<tr>
<td>assign</td>
<td>$C(\neg M) (i,j) \oplus = A^T$</td>
<td>C</td>
<td>$\neg, M, \oplus, A, T, i, j$</td>
</tr>
<tr>
<td>buildMatrix</td>
<td>$C(\neg M) \oplus = \mathbb{S}^{mxn}(i,j,v,\oplus)$</td>
<td>C</td>
<td>$\neg, M, \oplus, \oplus, m, n, i, j, v$</td>
</tr>
<tr>
<td>extractTuples</td>
<td>$(i,j,v) = A(\neg M)$</td>
<td>i,j,v</td>
<td>$\neg, M, A$</td>
</tr>
</tbody>
</table>

Notation: $i,j$ – index arrays, $v$ – scalar array, $m$ – 1D mask, other bold-lower – vector (column), $M$ – 2D mask, other bold-caps – matrix, $T$ – transpose, $\neg$ – structural complement, $\oplus$ monoid/binary function, $\odot \otimes$ semiring, blue – optional parameters, red – optional modifiers
GraphBLAS Primitives: The Code

\[ C(\neg M) \bigoplus = A^T \bigodot \cdot B^T \]

```c
GrB_Info GrB_mxm(GrB_Matrix *C, // output
    const GrB_Matrix Mask,
    const GrB_BinaryFunction accum,
    const GrB_Semiring op,
    const GrB_Matrix A, // input matrix
    const GrB_Matrix B // input matrix
[, const Descriptor desc]);
```

Common Elements

- Matrices (C, Mask, A, and B) are opaque data structures defined by the library implementers.
- Destination (C) is first.
- Output mask (M) specifies which output elements can assigned (optional).
- Accumulation function (accum) allows operation to combine with existing values (optional).
- Descriptor can specify transpose of input matrices or logical complement of the output mask.
The GraphBLAS Forum

• Meeting since 2014
• Teleconferences once per month (C API group more often)
• Forum meetings open to the public:
  - May: IEEE IPDPS (Symposium) – Graph Algorithms Building Blocks Workshop
  - Sept.: IEEE HPEC (Conference) – GraphBLAS BoF
  - Nov.: IEEE/ACM Supercomputing (Conference) – GraphBLAS Working Group
• Status and next steps:
  - Primitives established
  - C Application Programming Interface (API) – First draft review in August
  - Targeted release: November 2016 at Supercomputing
• For more information: [http://graphblas.org](http://graphblas.org)
Current and Future Work

• Continue work on C++ API from the open-source GraphBLAS Template Library (GBTL) [collaboration with Indiana/PNNL]

• Line Project (FY17-18) on automated code generation for high performance graph libraries [collaboration with CMU ECE department]
  - Starting with the GraphBLAS primitives
  - Targeting COTS hardware (CPUs and GPUs)

• DARPA’s HIVE (Hierarchical Identify Verify Exploit) BAA released in August
  - References the work of the GraphBLAS Forum
  - Goal: develop a special-purpose graph processing chip
Contact Information

Presenter / Point of Contact
Scott McMillan, PhD
Senior Member of Technical Staff
Telephone:  +1 412.268.5156
Email:  smcmillan@sei.cmu.edu