GraphBLAS

A Programming Specification for Graph Analysis

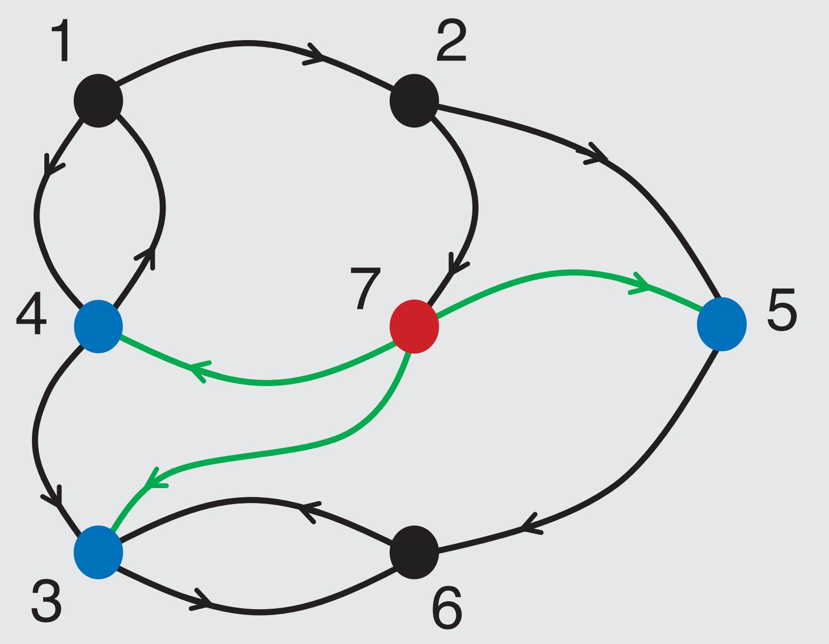
Graph algorithms are in wide use in DoD software applications, including intelligence analysis, autonomous systems, cyber intelligence and security, and logistics optimizations. However, graph algorithms are difficult and costly to implement efficiently on hardware systems. As the size of graphs and the pace at which new hardware is being developed increase, the complexity of developing high performance graph libraries becomes a prohibitive barrier to the work of analyzing the deluge of information.

Currently deep expertise is needed in graph algorithms and hardware tuning to achieve good performance on targeted hardware. It is rare to find this in individuals or even on teams within one organization.

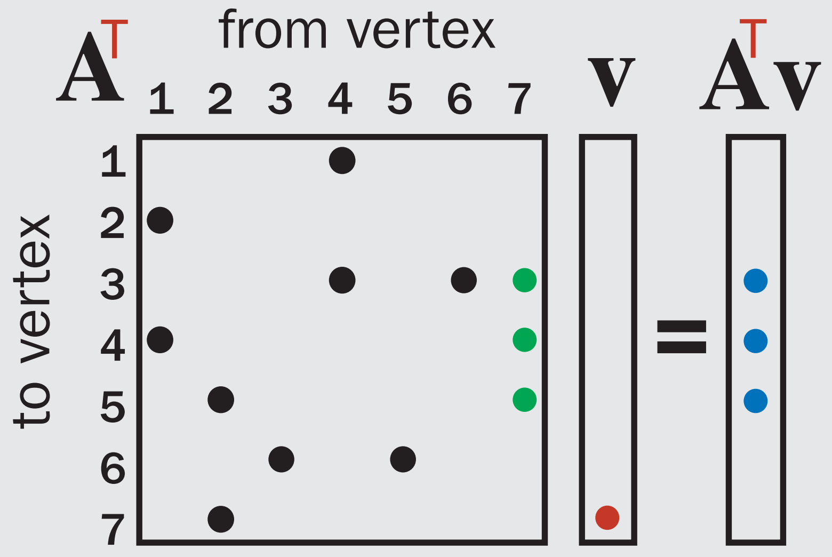
The GraphBLAS Forum – a government, academic and industry consortium – has defined a set of graph primitive objects and operations and is nearing completion of the C Application Programming Interface (API) specification that is able to separate the concerns between:

- the **graph expertise** needed to develop advanced graph analytics (writing code using the API) and
- the **hardware expertise** is needed to achieve high levels of performance (implementing efficient versions of the API for specific hardware).

For more information on the GraphBLAS Forum: http://graphblas.org



Graphs are a fundamental mathematical structure that captures the relationships (edges) between objects (vertices), as shown above. They can be represented as sparse matrices and an operation, such as matrix multiplication, is a key primitive in graph computations to find neighbors of a node as shown in both figures.

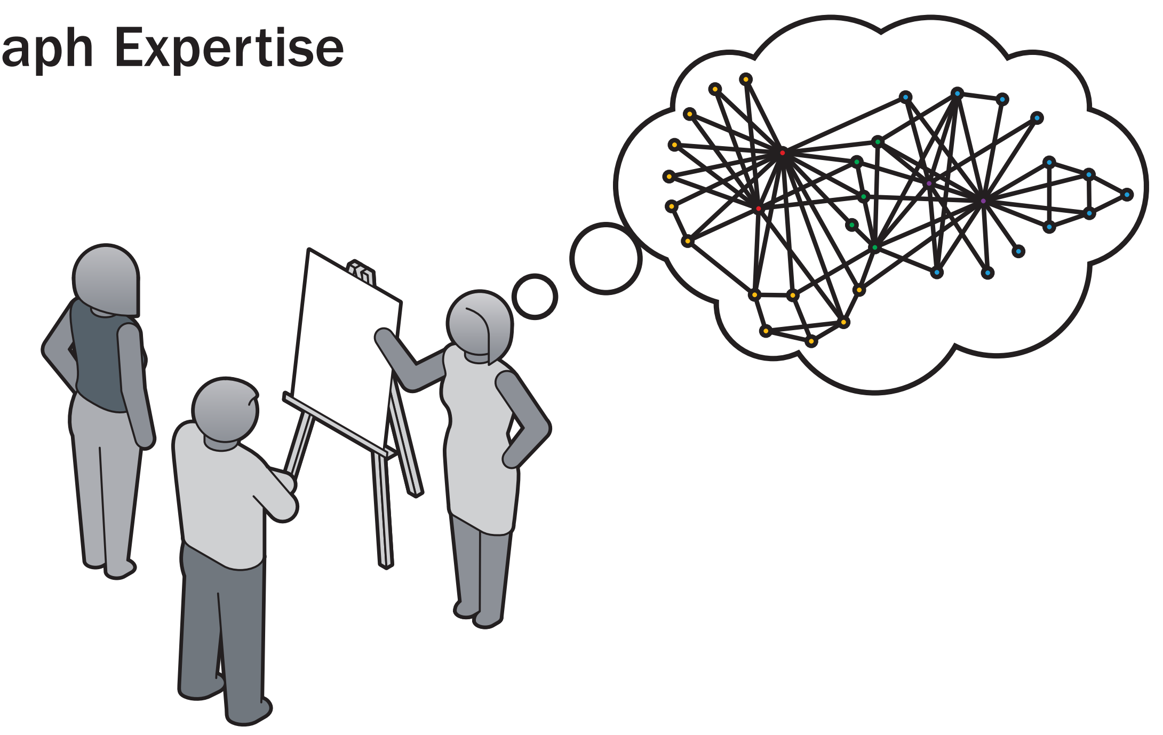


Operation	Description	Mathematical Description
mxm, mxv, vxm	Perform matrix multiplication (e.g., breadth-first traversal, shortest paths)	$\mathbf{C}(\neg\mathbf{M})\ \oplus\!=\mathbf{A}^{T}\ \oplus.\otimes\ \mathbf{B}^{T}$ $\mathbf{c}(\neg\mathbf{m})\ \oplus\!=\mathbf{A}^{T}\ \oplus.\otimes\ \mathbf{b}$
eWiseAdd, eWiseMult	Element-wise addition and multiplication of matrices (e.g., graph union, intersection)	$\mathbf{C}(\neg\mathbf{M})\ \oplus\!=\mathbf{A}^{T}\ \oplus\ \mathbf{B}^{T}$ $\mathbf{C}(\neg\mathbf{M})\ \oplus\!=\mathbf{A}^{T}\ \otimes\ \mathbf{B}^{T}$
extract	Extract a sub-matrix from a larger matrix (e.g., sub-graph selection)	$\mathbf{C}(\neg\mathbf{M})\ \oplus\!=\mathbf{A}^{T}(\mathbf{i},\mathbf{j})$
assign	Assign to a sub-matrix of a larger matrix (e.g., sub-graph assignment)	$\mathbf{C}(\neg\mathbf{M})\ (\mathbf{i},\mathbf{j})\ \oplus\!=\mathbf{A}^{T}$
apply	Apply unary function to each element of matrix (e.g., edge weight modification)	$\mathbf{C}(\neg\mathbf{M})\ \oplus\!=f(\mathbf{A}^{T})$
reduce	Reduce along columns or rows of matrices (vertex degree)	$\mathbf{c}(\neg\mathbf{m})\ \oplus\!=\oplus_{j}\ \mathbf{A}^{T}(:,\mathbf{j})$
transpose	Swaps the rows and columns of a sparse matrix (e.g., reverse directed edges)	$\mathbf{C}(\neg\mathbf{M})\ \oplus\!=\mathbf{A}^{T}$
buildMatrix	Build an matrix representation from row, column, value tuples	$\mathbf{C}(\neg\mathbf{M})\ \oplus\!=\mathbb{S}^{mxn}(\mathbf{i},\mathbf{j},\mathbf{v},\oplus)$
extractTuples	Extract the row, column, value tuples from a matrix representation	$(\mathbf{i},\mathbf{j},\mathbf{v},)=\mathbf{A}(\neg\mathbf{M})$

The table above lists all of the primitive operations supported by the GraphBLAS API along with their mathematical description. These mathematical “requirements” are being captured in a C API Specification as shown for matrix multiplication (mxm) below.

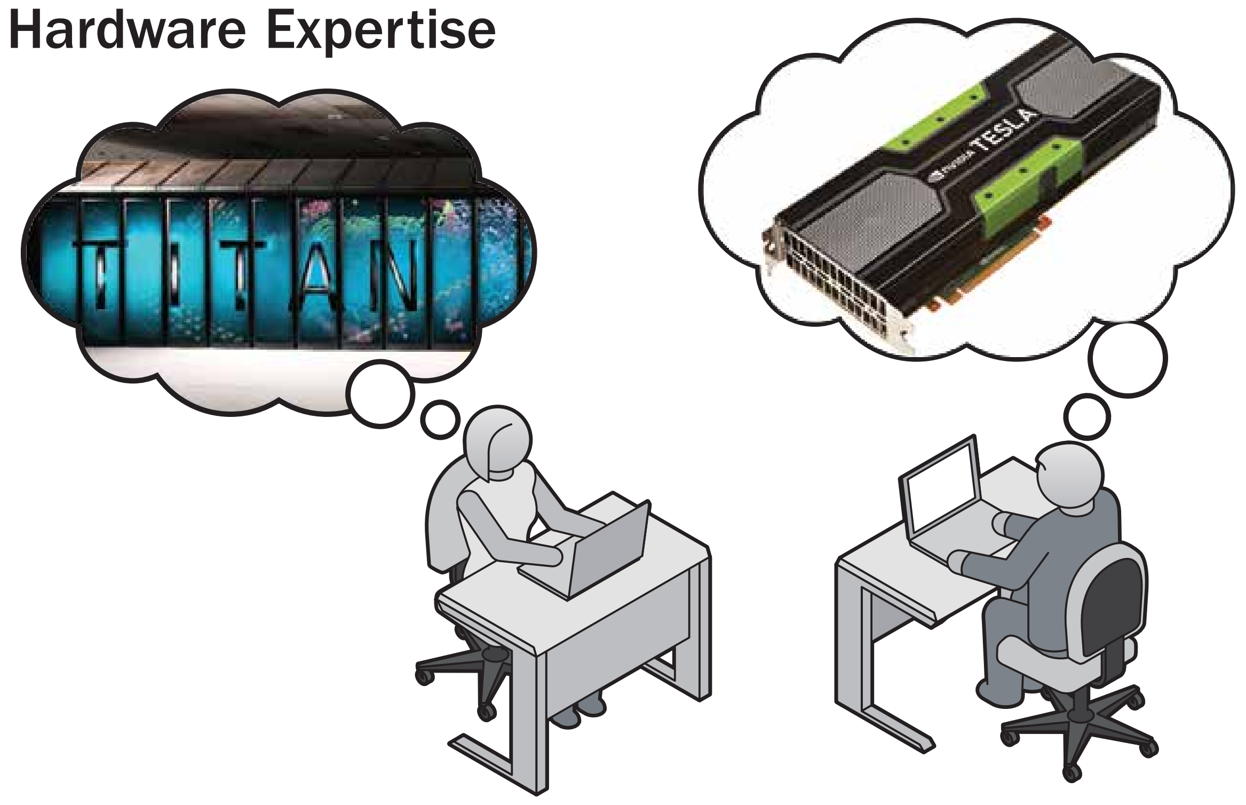
```
GrB_Info GrB_mxm(GrB_Matrix                *C,
                 const GrB_Matrix           Mask,
                 const GrB_BinaryFunction   accum,
                 const GrB_Semiring         op,
                 const GrB_Matrix           A,
                 const GrB_Matrix           B,
      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

Graph Expertise



Separation of Concerns:
GraphBLAS Application Programming Interface (API)

Hardware Expertise



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

Photos: https://www.flickr.com/people/gbpublic/

Contact: Scott McMillan | smcmillan@sei.cmu.edu