OpenSPARSE: An Open Platform for Sparse Basic Linear Algebra Subprograms

Weifeng Liu, Norwegian University of Science and Technology
Guangming Tan, Institute of Computing Technology, Chinese Academy of Sciences
Wei Xue, Tsinghua University
Hao Wang, Ohio State University

Outline

- A brief history of BLAS, Sparse BLAS, CombBLAS and GraphBLAS
- Recent work on optimizing sparse kernels
- Observations on performance and usage of sparse kernels
- OpenSPARSE: objective, design and preliminary results

A brief history of BLAS, Sparse BLAS, CombBLAS and GraphBLAS

Some milestones of BLAS - 1973

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Memorandum 33-660

A Proposal for Standard Linear Algebra Subprograms

R. J. Hanson
Washington State University

F. T. Krogh and C. L. Lawson
Jet Propulsion Laboratory

(NASA-CR-136216) A PROPOSAL FOR STANDARD LINEAR ALGEBRA SUBPROGRAMS (Jet Propulsion Lab.) ~~19~~ 20 p HC $3.00 CSCL 12A
N74-13288
Unclas
G3/19 24601

$w := \sum_{i=1}^{N} x_i y_i$	SW = SDOT(N, SX, INCX, SY, INCY)	SW, SDOT, SX( ), SY( ), single precision
$w := \sum_{i=1}^{N} x_i y_i$	DW = DSDOT(N, SX, INCX, SY, INCY)	DW, DSDOT double precision SX( ), SY( ) single precision
$w := \sum_{i=1}^{N} x_i y_i$	DW = DDOT(N, DX, INCX, DY, INCY)	DW, DDOT, DX( ), DY( ) double precision
$w := \sum_{i=1}^{N} x_i y_i$	CW = CDOT(N, CX, INCX, CY, INCY, 0)	CW, CDOT, CX( ), CY( ) complex
$w := \sum_{i=1}^{N} \bar{x}_i y_i$	CW = CDOT(N, CX, INCX, CY, INCY, 1)	CW, CDOT, CX( ), CY( ) complex
y := ax+y	CALL SELVOP(N, SA, SX, INCX, SY, INCY)	SA, SX( ), SY( ) single precision.
y := ax+y	CALL DELVOP(N, DA, DX, INCX, DY, INCY)	DA, DX( ), DY( ) double precision.
y := ax+y	CALL CELVOP(N, CA, CX, INCX, CY, INCY)	CA, CX( ), CY( ) complex.

R. J. Hanson, F. T. Krogh, C. L. Lawson. 1973. A Proposal for Standard Linear Algebra Subprograms. Technical Report. NASA.

NTNU

Some milestones of BLAS - 1988

An Extended Set of FORTRAN Basic Linear Algebra Subprograms

JACK J. DONGARRA
Argonne National Laboratory
JEREMY DU CROZ and SVEN HAMMARLING
Numerical Algorithms Group, Ltd.
and
RICHARD J. HANSON
Sandia National Laboratory

This paper describes an extension to the set of Basic Linear Algebra Subprograms. The extensions are targeted at matrix-vector operations that should provide for efficient and portable implementations of algorithms for high-performance computers.

The following three types of basic operation are performed by the Level 2 BLAS:

(1) matrix-vector products of the form

$$y \leftarrow \alpha A x + \beta y, \qquad y \leftarrow \alpha A^T x + \beta y, \qquad \text{and} \qquad y \leftarrow \alpha \bar{A}^T x + \beta y,$$

where α and β are scalars, x and y are vectors, and A is a matrix; and

$$x \leftarrow Tx, \qquad x \leftarrow T^T x, \qquad \text{and} \qquad x \leftarrow \bar{T}^T x,$$

where x is a vector and T is an upper or lower triangular matrix.

(2) rank-one and rank-two updates of the form

$$A \leftarrow \alpha x y^T + A, \qquad A \leftarrow \alpha x \bar{y}^T + A, \qquad H \leftarrow \alpha x \bar{x}^T + H, \qquad \text{and}$$
$$H \leftarrow \alpha x \bar{y}^T + \bar{\alpha} y \bar{x}^T + H,$$

where H is a Hermitian matrix.

(3) solution of triangular equations of the form

$$x \leftarrow T^{-1} x, \qquad x \leftarrow T^{-T} x, \qquad \text{and} \qquad x \leftarrow \bar{T}^{-T} x,$$

where T is a nonsingular upper or lower triangular matrix.

J. J. Dongarra, J. D. Croz, S. Hammarling, R. J. Hanson. 1988. An extended set of FORTRAN basic linear algebra subprograms. ACM Trans. Math. Softw.

Some milestones of BLAS - 1990

A Set of Level 3 Basic Linear Algebra Subprograms

JACK J. DONGARRA
University of Tennessee and Oak Ridge National Laboratory
JEREMY DU CROZ and SVEN HAMMARLING
Numerical Algorithms Group, Ltd.
and
IAIN DUFF
Harwell Laboratory

This paper describes a set of Level 3 Basic Linear Algebra Subprograms (Level 3 BLAS). The Level 3 BLAS are targeted at matrix-matrix operations, with the aim of providing more efficient, but portable, implementations of algorithms on high-performance computers, especially those with hierarchical memory and parallel processing capability.

In real arithmetic, the operations proposed for the Level 3 BLAS have the following forms:

(a) Matrix-matrix products

$$C \leftarrow \alpha AB + \beta C$$
$$C \leftarrow \alpha A^T B + \beta C$$
$$C \leftarrow \alpha AB^T + \beta C$$
$$C \leftarrow \alpha A^T B^T + \beta C$$

Note that these operations are more accurately described as matrix-matrix multiply-and-add operations; they include rank-k updates of a general matrix.

(b) Rank-k and rank-$2k$ updates of a symmetric matrix:

$$C \leftarrow \alpha AA^T + \beta C$$
$$C \leftarrow \alpha A^T A + \beta C$$
$$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$$
$$C \leftarrow \alpha A^T B + \alpha B^T A + \beta C$$

(c) Multiplying a matrix by a triangular matrix:

$$B \leftarrow \alpha TB$$
$$B \leftarrow \alpha T^T B$$
$$B \leftarrow \alpha BT$$
$$B \leftarrow \alpha BT^T$$

(d) Solving triangular systems of equations with multiple right-hand sides:

$$B \leftarrow \alpha T^{-1} B$$
$$B \leftarrow \alpha T^{-T} B$$
$$B \leftarrow \alpha BT^{-1}$$
$$B \leftarrow \alpha BT^{-T}$$

J. J. Dongarra, J. D. Croz, S. Hammarling, I. S. Duff. 1990. A set of level 3 basic linear algebra subprograms. ACM Trans. Math. Softw.

Some milestones of Sparse BLAS - 1991

Sparse Extensions to the FORTRAN Basic Linear Algebra Subprograms

DAVID S. DODSON
Convex Computer Corporation
and
ROGER G. GRIMES and JOHN G. LEWIS
Boeing Computer Services

This paper describes an extension to the set of Basic Linear Algebra Subprograms. The extension is targeted at sparse vector operations, with the goal of providing efficient, but portable, implementations of algorithms for high-performance computers.

Table I. Summary of Functions and Names of the Sparse BLAS Subprograms

Function	Root of name	Prefix and suffix of name							
Dot product	_DOT-	S-I	D-I	C-UI	Z-UI	C-CI	Z-CI		
Scalar times a vector added to a vector	_AXPY-	S-I	D-I	C-I	Z-I				
Apply Givens rotation	_ROT-	S-I	D-I						
Gather y into x	_GTHR-	S-	D-	C-	Z-	S-Z	D-Z	C-Z	Z-Z
Scatter x into y	_SCTR	S-	D-	C-	Z-				

D. S. Dodson, R. G. Grimes, J. G. Lewis. 1991. Sparse extensions to the FORTRAN Basic Linear Algebra Subprograms. ACM Trans. Math. Softw.

Some milestones of Sparse BLAS - 1992/1996

A Proposal for a Sparse BLAS Toolkit

SPARKER Working Note # 2

Michael A. Heroux

Cray Research, Inc., 655 Lone Oak Dr. Eagan, MN 55121 USA.

December 1992

Abstract

This paper describes a proposal for a "toolkit" of kernel routines for some of the basic operations in (iterative) sparse numerical methods. In particular, we describe an interface for routines which perform (i) sparse matrix times dense matrix product, (ii) the solution of a sparse triangular system with multiple right-hand-sides, (iii) the right permutation of a sparse matrix and (iv) a check for the integrity of a sparse matrix representation. The interfaces for these four operations are defined for a variety of common data structures and a set of guidelines is given to define interfaces for new data structures. The primary purpose of this toolkit is to provide a set of basic routines upon which the "User Level Sparse BLAS," as described in [6], can be built.

M. A. Heroux. 1992. A Proposal for a Sparse BLAS Toolkit. Technical Report. SPARKER Working Note 2.

A Revised Proposal for a Sparse BLAS Toolkit

SPARKER Working Note # 3

Sandra Carney* Michael A. Heroux† Guangye Li† Roldan Pozo‡
Karin A.Remington‡ Kesheng Wu*

January 1996

Abstract

This paper describes a proposal for a "toolkit" of kernel routines for some of the basic operations in (iterative) sparse numerical methods. In particular, we describe an interface for routines which perform (i) sparse matrix times dense matrix product, (ii) the solution of a sparse triangular system with multiple right-hand-sides, (iii) the right permutation of a sparse matrix and (iv) a check for the integrity of a sparse matrix representation. The interfaces for these four operations are defined for a variety of common data structures and a set of guidelines is given to define interfaces for new data structures. The primary purpose of this toolkit is to provide a set of basic routines upon which the "User Level Sparse BLAS," as described in [9], can be built. This paper is a revision of the original proposal found in [14].

S. Carney, M. A. Heroux, G. Li, K. Wu. 1996. A Revised Proposal for a Sparse BLAS Toolkit. Technical Report. SPARKER Working Note 3.

Data Structure
COO
CSC
CSR
DIA
ELL
JAD
SKY
BCO
BSC
BSR
BDI
BEL
VBR

NTNU

Some milestones of Sparse BLAS - 1997

Level 3 Basic Linear Algebra Subprograms for Sparse Matrices: A User-Level Interface

IAIN S. DUFF
Rutherford Appleton Laboratory
and
MICHELE MARRONE, GIUSEPPE RADICATI, and CARLO VITTOLI
IBM

This article proposes a set of Level 3 Basic Linear Algebra Subprograms and associated kernels for sparse matrices. A major goal is to design and develop a common framework to enable efficient, and portable, implementations of iterative algorithms for sparse matrices on high-performance computers. We have designed the routines to shield the developer of mathematical software from most of the complexities of the various data structures used for sparse matrices. We have kept the interface and suite of codes as simple as possible while at the same time including sufficient functionality to cover most of the requirements of iterative solvers and sufficient flexibility to cover most sparse matrix data structures. An important aspect of our framework is that it can be easily extended to incorporate new kernels if the need arises. We discuss the design, implementation, and use of subprograms for the multiplication of a full matrix by a sparse one and for the solution of sparse triangular systems with one or more (full) right-hand sides. We include a routine for checking the input data, generating a new sparse data structure from that input, and scaling a sparse matrix. The new data structure for the transformation can be specified by the user or can be chosen automatically by vendors to be efficient on their machines. We also include a routine for permuting the columns of a sparse matrix and one for permuting the rows of a full matrix.

We propose interfaces for the following functions:

(1) a routine for performing the product of a sparse and a full matrix,

(2) a routine for solving a sparse upper or lower triangular system of linear equations for a full matrix of right-hand sides,

(3) a routine to check the input data, to transform from one sparse format to another, and to scale a sparse matrix, and

(4) a routine to permute the columns of a sparse matrix and a routine to permute the rows of a full matrix.

I. S. Duff, M. Marrone, G. Radicati, C. Vittoli. 1997. Level 3 basic linear algebra subprograms for sparse matrices: a user-level interface. ACM Trans. Math. Softw.

Some milestones of Sparse BLAS - 2002

An Overview of the Sparse Basic Linear Algebra Subprograms: The New Standard from the BLAS Technical Forum

IAIN S. DUFF
CERFACS and Rutherford Appleton Laboratory
MICHAEL A. HEROUX
Sandia National Laboratories
and
ROLDAN POZO
National Institute of Standards and Technology

We discuss the interface design for the Sparse Basic Linear Algebra Subprograms (BLAS), the kernels in the recent standard from the BLAS Technical Forum that are concerned with unstructured sparse matrices. The motivation for such a standard is to encourage portable programming while allowing for library-specific optimizations. In particular, we show how this interface can shield one from concern over the specific storage scheme for the sparse matrix. This design makes it easy to add further functionality to the sparse BLAS in the future.

We illustrate the use of the Sparse BLAS with examples in the three supported programming languages, Fortran 95, Fortran 77, and C.

Table I. Level 1 Sparse BLAS: Sparse Vector Operations

USDOT	sparse dot product	$r \leftarrow x^T y$, $r \leftarrow x^H y$
USAXPY	sparse vector update	$y \leftarrow \alpha x + y$
USGA	sparse gather	$x \leftarrow y\|_x$
USGZ	sparse gather and zero	$x \leftarrow y\|_x; y\|_x \leftarrow 0$
USSC	sparse scatter	$y\|_x \leftarrow x$

Table II. Level 2 Sparse BLAS: Sparse Matrix-Vector Operations

USMV	sparse matrix-vector multiply	$y \leftarrow \alpha A x + y$ $y \leftarrow \alpha A^T x + y$ $y \leftarrow \alpha A^H x + y$
USSV	sparse triangular solve	$x \leftarrow \alpha A^{-1} x$ $x \leftarrow \alpha A^{-T} x$ $x \leftarrow \alpha A^{-H} x$

Table III. Level 3 Sparse BLAS: Sparse Matrix-Matrix Operations

USMM	sparse matrix-matrix multiply	$C \leftarrow \alpha AB + C$ $C \leftarrow \alpha A^T B + C$ $C \leftarrow \alpha A^H B + C$
USSM	sparse triangular solve	$B \leftarrow \alpha T^{-1} B$ $B \leftarrow \alpha T^{-T} B$ $B \leftarrow \alpha T^{-H} B$

I. S. Duff, M. A. Heroux, R. Pozo. 2002. An overview of the sparse basic linear algebra subprograms: The new standard from the BLAS technical forum. ACM Trans. Math. Softw.

Some implementations of Sparse BLAS - 1994

A Sparse Matrix Library in C++ for High Performance Architectures*

Jack Dongarra[§‡], Andrew Lumsdaine[◇], Xinhiu Niu[◇]
Roldan Pozo[‡], Karin Remington[§]

[§]Oak Ridge National Laboratory
Mathematical Sciences Section

[‡]University of Tennessee
Dept. of Computer Science

[◇]University of Notre Dame
Dept. of Computer Science & Engineering

Abstract

We describe an object oriented sparse matrix library in C++ built upon the Level 3 Sparse BLAS proposal [5] for portability and performance across a wide class of machine architectures. The C++ library includes algorithms for various iterative methods and supports the most common sparse data storage formats used in practice. Besides simplifying the subroutine interface, the object oriented design allows the same driving code to be used for various sparse matrix formats, thus addressing many of the difficulties encountered with the typical approach to sparse matrix libraries.

neering applications. Nevertheless, comprehensive libraries for sparse matrix computations have not been developed and integrated to the same degree as those for dense matrices. Several factors contribute to the difficulty of designing such a comprehensive library. Different computer architectures, as well as different applications, call for different sparse matrix data formats in order to best exploit registers, data locality, pipelining, and parallel processing. Furthermore, code involving sparse matrices tends to be very complicated, and not easily portable, because the details of the underlying data formats are invariably entangled within the application code.

The design of the library is based on the following principles:

Clarity: Implementations of numerical algorithms should resemble the mathematical algorithms on which they are based. This is in contrast to Fortran 77, which can require complicated subroutine calls, often with parameter lists that stretch over several lines.

Reuse: A particular algorithm should only need to be coded once, with identical code used for all matrix representations.

Portability: Implementations of numerical algorithms should be directly portable across machine platforms.

High Performance: The object oriented library code should perform as well as optimized data-format-specific code written in C or Fortran.

J. Dongarra, A. Lumsdaine, X. Niu, R. Pozo, K. Remington. 1994. LAPACK Working Note 74: A Sparse Matrix Library in C++ for High Performance Architectures. Technical Report.

NTNU

Some implementations of Sparse BLAS - 2000

PSBLAS: A Library for Parallel Linear Algebra Computation on Sparse Matrices

SALVATORE FILIPPONE
IBM Italia
and
MICHELE COLAJANNI
Università di Modena e Reggio Emilia

Many computationally intensive problems in engineering and science give rise to the solution of large, sparse, linear systems of equations. Fast and efficient methods for their solution are very important because these systems usually occur in the innermost loop of the computational scheme. Parallelization is often necessary to achieve an acceptable level of performance. This paper presents the design, implementation, and interface of a library of Basic Linear Algebra Subroutines for sparse matrices (PSBLAS) which is specifically tailored to distributed-memory computers. PSBLAS enables easy, efficient, and portable implementations of parallel iterative solvers for linear systems. The interface keeps in view a Single Program Multiple Data programming model on distributed-memory machines. However, the architecture of the library does not exclude an implementation in different paradigms, such as those based on the shared-memory model.

The main operations in the PSBLAS library are:

—Matrix-matrix products

$$C \leftarrow \alpha P_R A P_C B + \beta C$$

$$C \leftarrow \alpha P_R A^T P_C B + \beta C$$

—Triangular system solutions

$$C \leftarrow \alpha P_R T^{-1} P_C B + \beta C$$

$$C \leftarrow \alpha D P_R T^{-1} P_C B + \beta C$$

$$C \leftarrow \alpha P_R T^{-1} P_C D B + \beta C$$

$$C \leftarrow \alpha P_R T^{-T} P_C B + \beta C$$

$$C \leftarrow \alpha D P_R T^{-T} P_C B + \beta C$$

$$C \leftarrow \alpha P_R T^{-T} P_C D B + \beta C$$

—Matrix sums

$$Y \leftarrow \alpha X + \beta Y$$

—Scalar products

$$x^T y \quad \text{or} \quad x^H y$$

—Dense vector 1, 2 and infinity norms

$$\|x\|$$

—Sparse matrix infinity norm

$$\|A\|_\infty$$

S. Filippone, M. Colajanni. 2000. PSBLAS: a library for parallel linear algebra computation on sparse matrices. ACM Trans. Math. Softw.

Some implementations of Sparse BLAS - 2002

Algorithm 818: A Reference Model Implementation of the Sparse BLAS in Fortran 95

IAIN S. DUFF
CERFACS, France and Atlas Centre, RAL, England
and
CHRISTOF VÖMEL
CERFACS, France

The Basic Linear Algebra Subprograms for sparse matrices (Sparse BLAS) as defined by the BLAS Technical Forum are a set of routines providing basic operations for sparse matrices and vectors. A principal goal of the Sparse BLAS standard is to aid in the development of iterative solvers for large sparse linear systems by specifying on the one hand interfaces for a high-level description of vector and matrix operations for the algorithm developer and on the other hand leaving enough freedom for vendors to provide the most efficient implementation of the underlying algorithms for their specific architectures.

The Sparse BLAS standard defines interfaces and bindings for the three target languages: C, Fortran 77 and Fortran 95. We describe here our Fortran 95 implementation intended as a reference model for the Sparse BLAS. We identify the underlying complex issues of the representation and the handling of sparse matrices and give suggestions to other implementors of how to address them.

Table I. Sparse BLAS: Operations for the Handling of Sparse Matrices

USCR_BEGIN	begin point-entry construction
USCR_BLOCK_BEGIN	begin block-entry construction
USCR_VARIABLE_BLOCK_BEGIN	begin variable block-entry construction
USCR_INSERT_ENTRY	add point-entry
USCR_INSERT_ENTRIES	add list of point-entries
USCR_INSERT_COL	add a compressed column
USCR_INSERT_ROW	add a compressed row
USCR_INSERT_CLIQUE	add a dense matrix clique
USCR_INSERT_BLOCK	add a block entry
USCR_END	end construction
USSP	set matrix property
USGP	get/test for matrix property
USDS	release matrix handle

I. S. Duff, C. Vömel. 2002. Algorithm 818: A reference model implementation of the sparse BLAS in fortran 95. ACM Trans. Math. Softw.

Some implementations of Sparse BLAS - 2003

Object-Oriented Techniques for Sparse Matrix Computations in Fortran 2003

SALVATORE FILIPPONE, University of Rome "Tor Vergata"
ALFREDO BUTTARI, CNRS-IRIT Toulouse

The efficiency of a sparse linear algebra operation heavily relies on the ability of the sparse matrix storage format to exploit the computing power of the underlying hardware. Since no format is universally better than the others across all possible kinds of operations and computers, sparse linear algebra software packages should provide facilities to easily implement and integrate new storage formats within a sparse linear algebra application without the need to modify it; it should also allow to dynamically change a storage format at run-time depending on the specific operations to be performed. Aiming at these important features, we present an Object Oriented design model for a sparse linear algebra package which relies on Design Patterns. We show that an implementation of our model can be efficiently achieved through some of the unique features of the Fortran 2003 language. Experimental results show that the proposed software infrastructure improves the modularity and ease of use of the code at no performance loss.

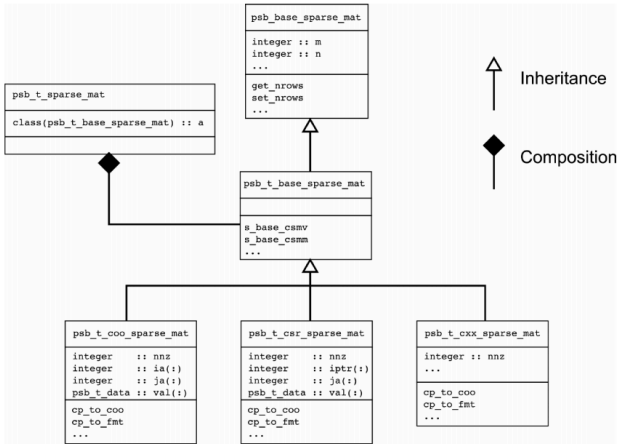


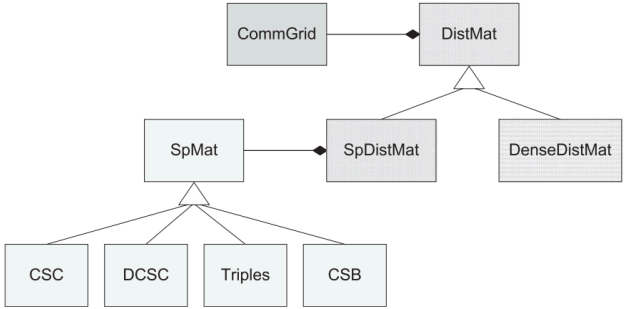
Fig. 1. UML class diagram.

S. Filippone, A. Buttari. 2012. Object-Oriented Techniques for Sparse Matrix Computations in Fortran 2003. ACM Trans. Math. Softw.

Combinatorial BLAS - 2011

The Combinatorial BLAS: design, implementation, and applications

Aydın Buluç[1] and John R Gilbert[2]



Abstract
This paper presents a scalable high-performance software library to be used for graph analysis and data mining. Large combinatorial graphs appear in many applications of high-performance computing, including computational biology, informatics, analytics, web search, dynamical systems, and sparse matrix methods. Graph computations are difficult to parallelize using traditional approaches due to their irregular nature and low operational intensity. Many graph computations, however, contain sufficient coarse-grained parallelism for thousands of processors, which can be uncovered by using the right primitives. We describe the parallel Combinatorial BLAS, which consists of a small but powerful set of linear algebra primitives specifically targeting graph and data mining applications. We provide an extensible library interface and some guiding principles for future development. The library is evaluated using two important graph algorithms, in terms of both performance and ease-of-use. The scalability and raw performance of the example applications, using the Combinatorial BLAS, are unprecedented on distributed memory clusters.

Function	Applies to
SpGEMM	Sparse Matrix (as friend)
SpMV	Sparse Matrix (as friend)
SpEWiseX	Sparse Matrices (as friend)
Reduce	Any Matrix (as method)
SpRef	Sparse Matrix (as method)
SpAsgn	Sparse Matrix (as method)
Scale	Any Matrix (as method)
Scale	Any Vector (as method)
Apply	Any Object (as method)

A. Buluç, J. R. Gilbert. 2011. The Combinatorial BLAS: design, implementation, and applications. Int. J. High Perform. Comput. Appl.

GraphBLAS - 2017

Design of the GraphBLAS API for C

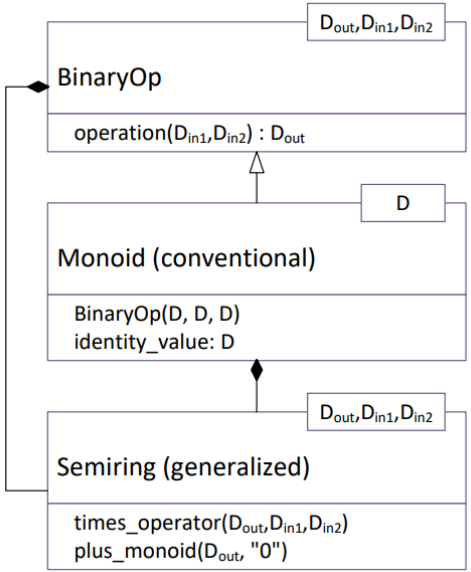
Aydın Buluç[†], Tim Mattson[‡], Scott McMillan[§], José Moreira[¶], Carl Yang[*,†]

[†]*Computational Research Division, Lawrence Berkeley National Laboratory*
[‡]*Intel Corporation* [§]*Software Engineering Institute, Carnegie Mellon University* [¶]*IBM Corporation*
[*]*Electrical and Computer Engineering Department, University of California, Davis, USA*

***Abstract*—The purpose of the GraphBLAS Forum is to standardize linear-algebraic building blocks for graph computations. An important part of this standardization effort is to translate the mathematical specification into an actual Application Programming Interface (API) that (i) is faithful to the mathematics and (ii) enables efficient implementations on modern hardware. This paper documents the approach taken by the C language specification subcommittee and presents the main concepts, constructs, and objects within the GraphBLAS API. Use of the API is illustrated by showing an implementation of the betweenness centrality algorithm.**

C++. Since this is the first specification of GraphBLAS in any language, our burden has largely been to define the programming concepts for the first time. We believe our design will largely carry over to future specs of the GraphBLAS in other languages.

This paper summarizes the GraphBLAS C API and the motivation behind our decisions. We begin by summarizing the mathematical ideas behind the GraphBLAS and how those ideas influenced our notation. We then explain data structures, algebraic objects, and objects that control the semantics of the



A. Buluç, T. Mattson, S. McMillan, J. Moreira, C. Yang. Design of the GraphBLAS API for C. 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW).

SuiteSparse:GraphBLAS - 2018

Algorithm 9xx: SuiteSparse:GraphBLAS: graph algorithms in the language of sparse linear algebra

TIMOTHY A. DAVIS, Texas A&M University

SuiteSparse:GraphBLAS is a full implementation of the GraphBLAS standard, which defines a set of sparse matrix operations on an extended algebra of semirings using an almost unlimited variety of operators and types. When applied to sparse adjacency matrices, these algebraic operations are equivalent to computations on graphs. GraphBLAS provides a powerful and expressive framework for creating graph algorithms based on the elegant mathematics of sparse matrix operations on a semiring. An overview of the GraphBLAS specification is given, followed by a description of the key features of its implementation in the SuiteSparse:GraphBLAS package.

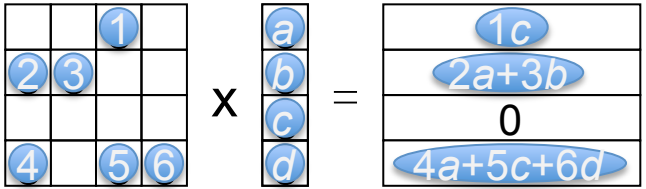
object
GrB_Type
GrB_UnaryOp
GrB_BinaryOp
GxB_SelectOp
GrB_Monoid
GrB_Semiring
GrB_Matrix
GrB_Vector
GrB_Descriptor

T. Davis. Algorithm 9xx: SuiteSparse:GraphBLAS: graph algorithms in the language of sparse linear algebra. ACM Trans. Math. Softw. Under review.

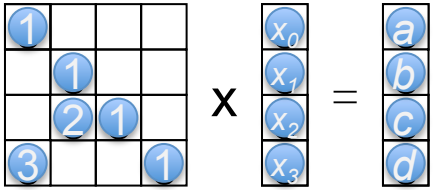
Recent work on optimizing sparse kernels

Sparse kernels received much attention

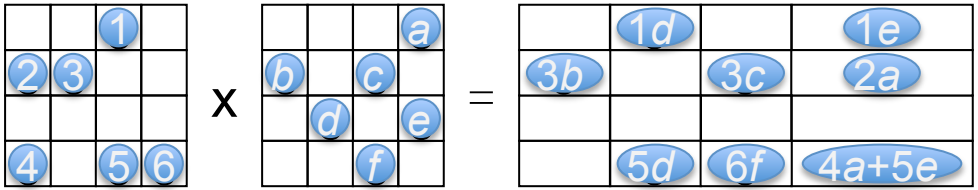
- Sparse matrix-vector Multiplication (SpMV)



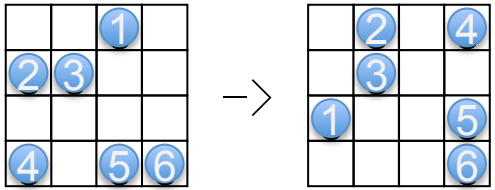
- Sparse triangular solve (SpTRSV)



- Sparse matrix-matrix Multiplication (SpGEMM)



- Sparse transposition (SpTRANS)



NTNU

Some recent sparse kernels – 2014

- [SpMV] J. L. Greathouse, M. Daga. Efficient Sparse Matrix-Vector Multiplication on GPUs using the CSR Storage Format. SC ’14.
- [SpMV] A. Ashari, N. Sedaghati, J. Eisenlohr, S. Parthasarathy, P. Sadayappan. Fast Sparse Matrix-Vector Multiplication on GPUs for Graph Applications. SC ’14.
- [SpMV] A. Ashari, N. Sedaghati, J. Eisenlohr, P. Sadayappan. An Efficient Two-Dimensional Blocking Strategy for Sparse Matrix-vector Multiplication on GPUs. ICS ’14.
- [SpMV] S. Yan, C. Li, Y. Zhang, H. Zhou. yaSpMV: Yet Another SpMV Framework on GPUs. PPoPP ’14.
- [SpMV] M. Kreutzer, G. Hager, G. Wellein, H. Fehske, A. Bishop. A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units. SISC.
- [SpGEMM] W. Liu, B. Vinter. An efficient GPU general sparse matrix-matrix multiplication for irregular data. IPDPS ’14.
- [SpTRSV] J. Park, M. Smelyanskiy, N. Sundaram, P. Dubey. Sparsifying Synchronization for High-Performance Shared-Memory Sparse Triangular Solver. ISC ’14.

Some recent sparse kernels - 2015

- [SpMV] W. Liu, B. Vinter. CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication. ICS ’15.
- [SpMV] N. Sedaghati, T. Mu, L. N. Pouchet, et al. Automatic selection of sparse matrix representation on GPUs. ICS ’15.
- [SpMV] M. Daga, J. L. Greathouse. Structural agnostic SpMV: Adapting CSR-adaptive for irregular matrices. HiPC ’15.
- [SpMV, SpGEMM] S. Dalton, S. Baxter, D. Merrill, L. Olson. Optimizing Sparse Matrix Operations on GPUs Using Merge Path. IPDPS ’15.
- [SpGEMM] F. Gremse, A. Hofter, L. O. Schwen, F. Kiessling, U. Naumann. GPU-accelerated sparse matrix-matrix multiplication by iterative row merging. SISC.
- [SpGEMM] M. M. A. Patwary, N. R. Satish, N. Sundaram, J. Park. Parallel efficient sparse matrix-matrix multiplication on multicore platforms. ISC ’15.
- [SpGEMM] S. Dalton, L. Olson, N. Bell. Optimizing Sparse Matrix-Matrix Multiplication for the GPU. TOMS.
- [SpTRSV] H. Kabir, J.D. Booth, G. Aupy, A. Benoit, Y. Robert, P. Raghavan. STSk: A Multilevel Sparse Triangular Solution Scheme for NUMA Multicores. SC ’15.

Some recent sparse kernels - 2016

- [SpMV] Y. Zhang, S. Li, S. Yan, H. Zhou. A cross-platform SpMV framework on many-core architectures. TACO.
- [SpMV] D. Merrill, M. Garland. Merge-based parallel sparse matrix-vector multiplication. SC ’16.
- [SpGEMM] A. Azad, G. Ballard, A. Buluc, J. Demmel, L. Grigori. Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. SISC.
- [SpGEMM] P. N. Q. Anh, R. Fan, Y. Wen. Balanced hashing and efficient gpu sparse general matrix-matrix multiplication. ICS ’16.
- [SpTRSV] W. Liu, A. Li, J. D. Hogg, I. S. Duff, B. Vinter. A Synchronization-Free Algorithm for Parallel Sparse Triangular Solves. Euro-Par ’16.
- [SpTRSV] A. M. Bradley. A Hybrid Multithreaded Direct Sparse Triangular Solver. CSC ’16.
- [SpTRANS] H. Wang, W. Liu, K. Hou, W. Feng. Parallel Transposition of Sparse Data Structures. ICS ’16.

Some recent sparse kernels - 2017

- [SpMV] M. Steinberger, R. Zayer, H. P. Seidel. Globally homogeneous, locally adaptive sparse matrix-vector multiplication on the GPU. ICS ’17.
- [SpMV] A. Elafrou, G. Goumas, N. Koziris. Performance Analysis and Optimization of Sparse Matrix-Vector Multiplication on Modern Multi-and Many-Core Processors. ICPP ’17.
- [SpMV] J. P. Ecker, R. Berrendorf, F. Mannuss. New Efficient General Sparse Matrix Formats for Parallel SpMV Operations. Euro-Par ’17.
- [SpMV] G. Flegar, E. S. Quintana-Ortí. Balanced CSR Sparse Matrix-Vector Product on Graphics Processors. Euro-Par ’17.
- [SpMSpV] A. Azad, A. Buluç. A work-efficient parallel sparse matrix-sparse vector multiplication algorithm. IPDPS ’17.
- [SpGEMM] K. Akbudak, C. Aykanat. Exploiting locality in sparse matrix-matrix multiplication on many-core architectures. TPDS.
- [SpGEMM] Y. Nagasaka, A. Nukada, S. Matsuoka. High-performance and Memory-saving Sparse General Matrix-Matrix Multiplication for NVIDIA Pascal GPU. ICPP ’17.
- [SpGEMM] R. Kunchum, A. Chaudhry, A. Sukumaran-Rajam, Q. Niu, I. Nisa, P. Sadayappan. On improving performance of sparse matrix-matrix multiplication on GPUs. ICS ’17.

Some recent sparse kernels - 2018

- [SpMV] Y. Zhao, W. Zhou, X. Shen, G. Yiu. Overhead-Conscious Format Selection for SpMV-Based Applications. IPDPS ’18.
- [SpMV] C. Liu, B. Xie, X. Liu, W. Xue, H. Yang, X. Liu. Towards Efficient SpMV on Sunway Manycore Architectures. ICS ’18.
- [SpMV] B. Xie, J. Zhan, X. Liu, W. Gao, Z. Jia, X. He. CVR: efficient vectorization of SpMV on x86 processors. CGO ’18.
- [SpMV] A. Elafrou, V. Karakasis, T. Gkountouvas. SparseX: A Library for High-Performance Sparse Matrix-Vector Multiplication on Multicore Platforms. TOMS.
- [SpMV] Q. Sun, C. Zhang, C. Wu, J. Zhang, L. Li. Bandwidth Reduced Parallel SpMV on the SW26010 Many-Core Platform. ICPP ’18.
- [SpMV] G. Tan, J. Liu, J. Li. Design and Implementation of Adaptive SpMV Library for Multicore and Many-Core Architecture. TOMS.
- [SpMM] C. Yang, A Buluç, J. D. Owens. Design Principles for Sparse Matrix Multiplication on the GPU. Euro-Par ’18.
- [SpMM] C. Hong, A. Sukumaran-Rajam. Efficient sparse-matrix multi-vector product on GPUs. HPDC ’18.

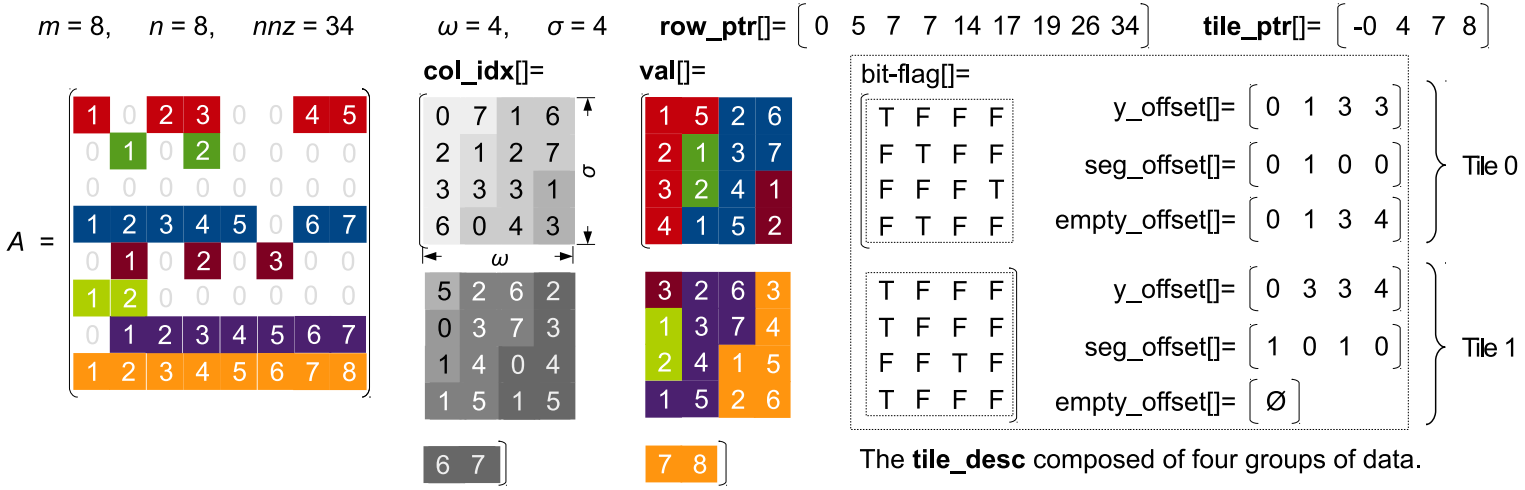
Some recent sparse kernels - 2018 (cont.)

- [SpGEMM] M. Deveci, C. Trott, S. Rajamanickam. Multi-threaded Sparse Matrix-Matrix Multiplication for Many-Core and GPU Architectures. PARCO.
- [SpGEMM] J. Liu, X. He, W. Liu, G. Tan. Register-Aware Optimizations for Parallel Sparse Matrix-Matrix Multiplication. IJPP.
- [SpGEMM] F. Gremse, K. Küpper, U. Naumann. Memory-Efficient Sparse Matrix-Matrix Multiplication by Row Merging on Many-Core Architectures. SISC.
- [SpGEMM] Y. Nagasaka, S. Matsuoka, A. Azad, A. Buluç. High-performance sparse matrix-matrix products on Intel KNL and multicore architectures. ICPPW ’18.
- [SpTRSV] X. Wang, W. Liu, W. Xue, L. Wu. swSpTRSV: a fast sparse triangular solve with sparse level tile layout on sunway architectures. PPoPP ’18.
- [SpTRSV] E. Dufrechou, P. Ezzatti. A New GPU Algorithm to Compute a Level Set-Based Analysis for the Parallel Solution of Sparse Triangular Systems. IPDPS ’18.
- [SpTRSV] X. Wang, P. Xu, W. Xue, Y. Ao, C. Yang, H. Fu. A Fast Sparse Triangular Solver for Structured-grid Problems on Sunway Many-core Processor SW26010. ICPP ’18.

Some observations
1. Diverse performance

CSR5-based SpMV (our work)

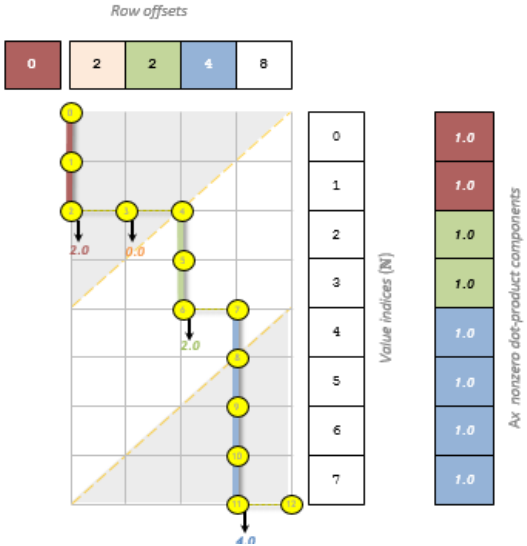
- Organize nonzeros in Tiles of identical size. The design objectives include load balancing, SIMD-friendly, low preprocessing cost and reduced storage space.



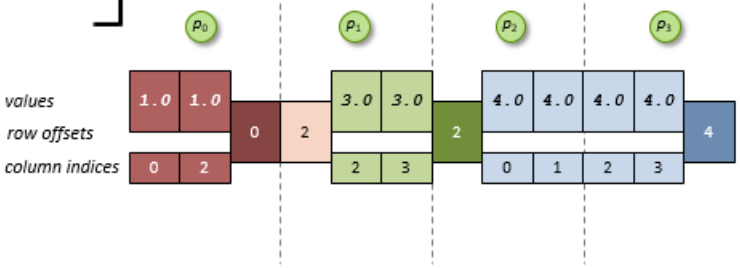
W. Liu, B. Vinter. **CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication**. *ICS15*.

Merge-based SpMV

- Both nonzeros and output vector are assigned to CTAs/processes in a balanced way.



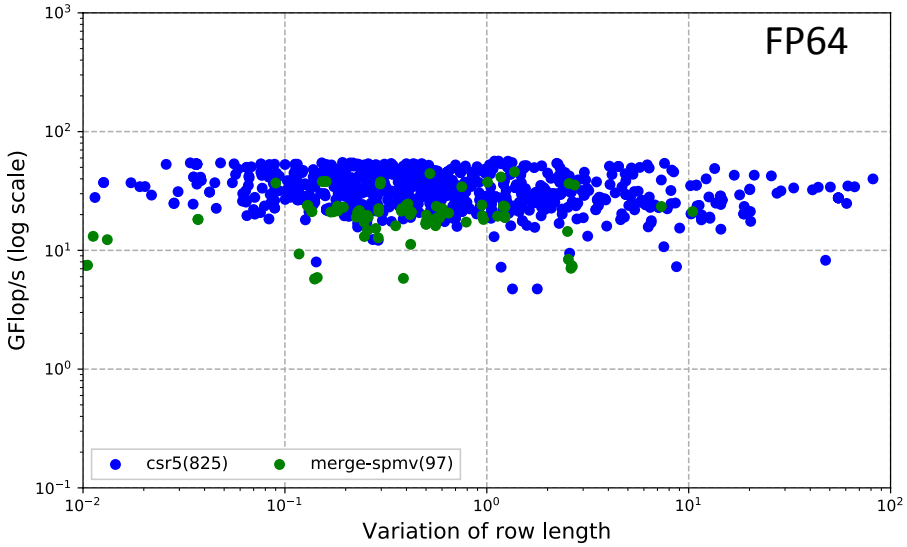
$$\begin{bmatrix} 1.0 & -- & 1.0 & -- \\ -- & -- & -- & -- \\ -- & -- & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$$

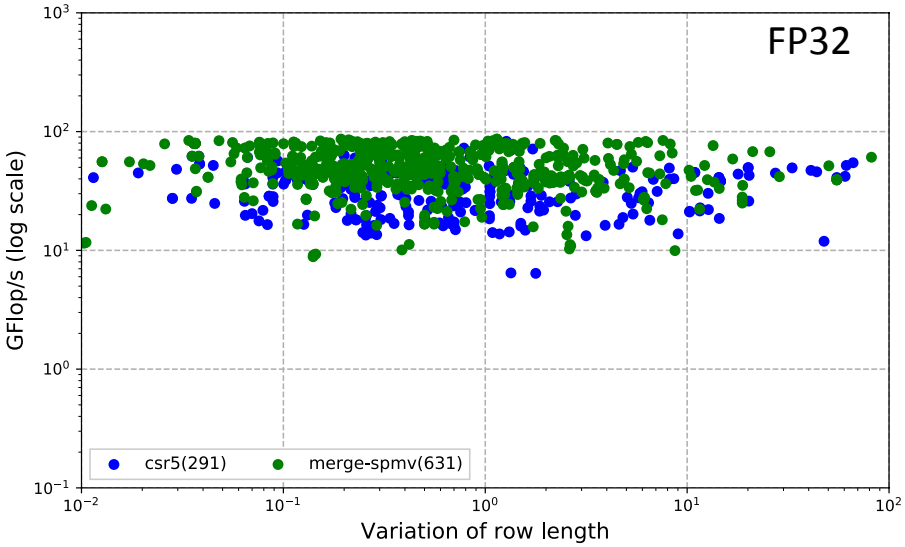


D. Merrill, M. Garland. **Merge-based Parallel Sparse Matrix-Vector Multiplication**. *SC16.*

Diverse performance - SpMV

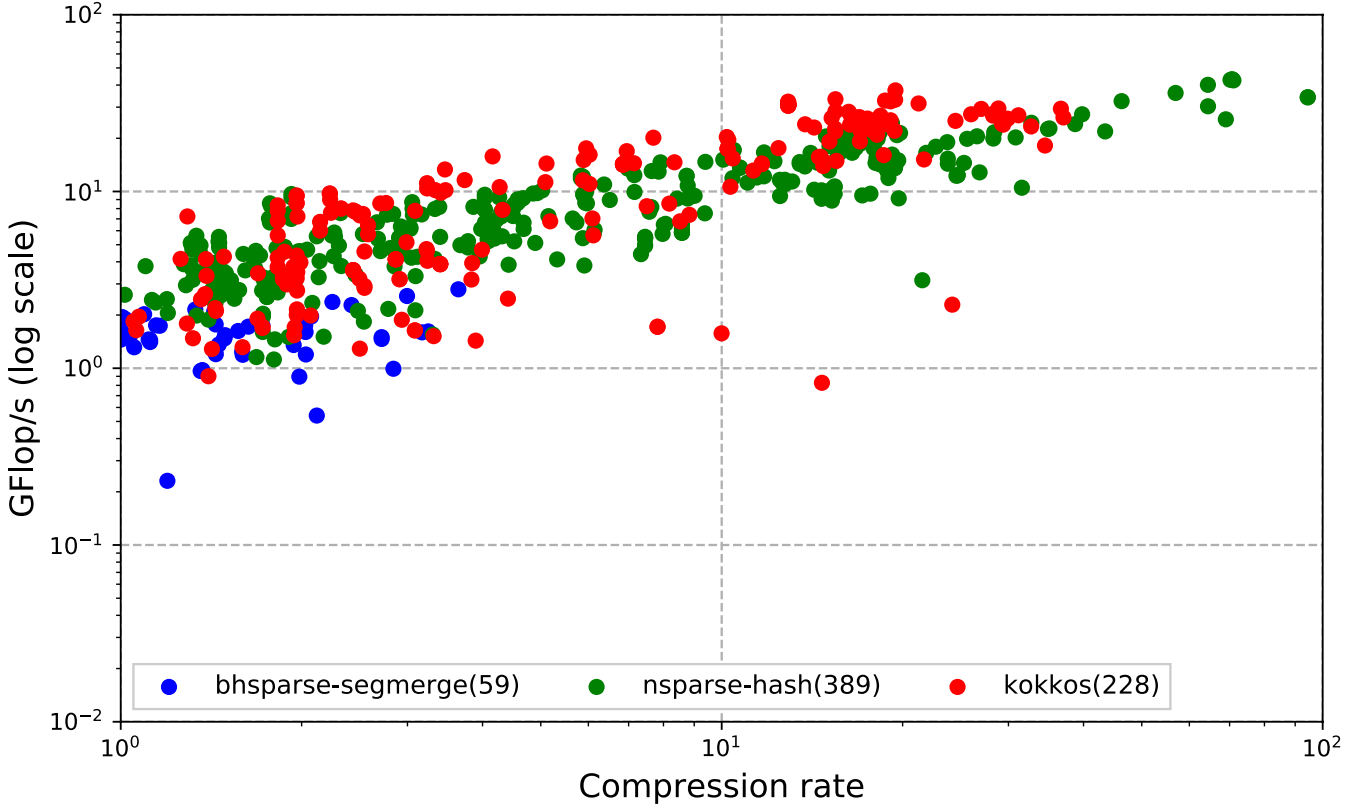
- CSR5 outperforms merge-spmv in double precision, but merge-spmv outperforms CSR5 in single precision.





Running 956 matrices on an NVIDIA Titan X Pascal.

Diverse performance - SpGEMM

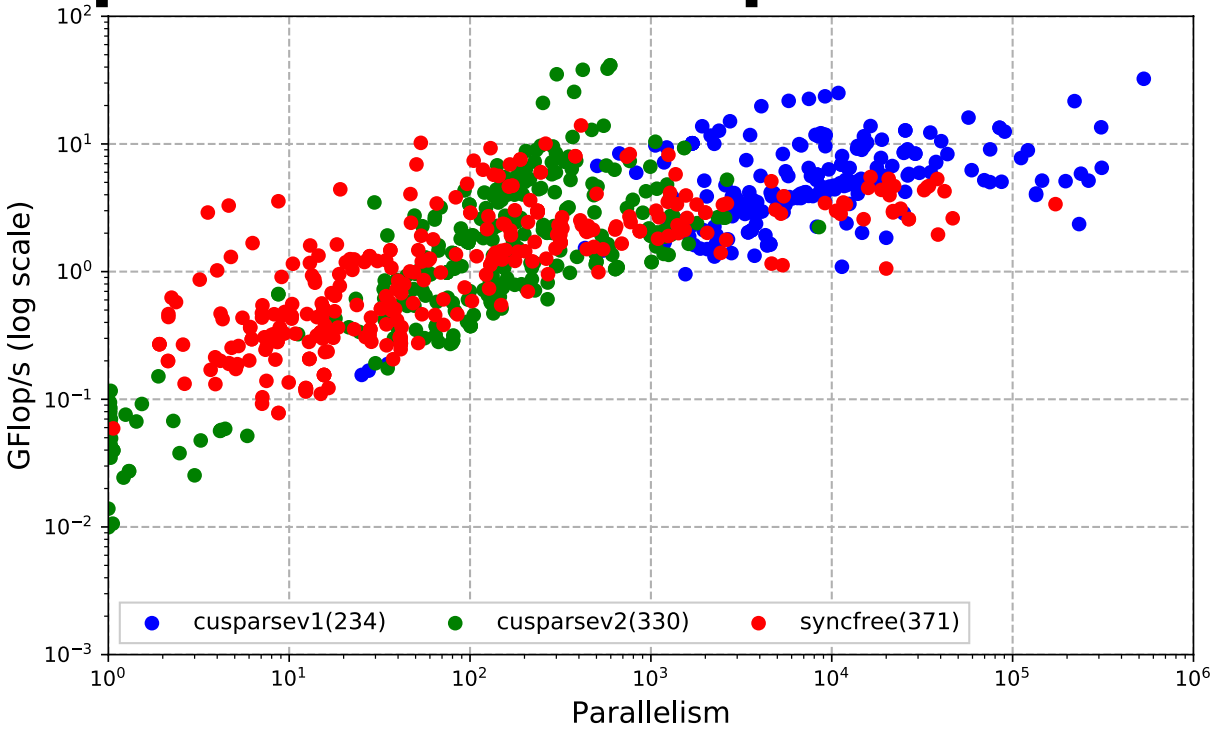


W. Liu, B. Vinter. **A Framework for General Sparse Matrix-Matrix Multiplication on GPUs and Heterogeneous Processors**. *JPDC. 2015.*

Y. Nagasaka, A. Nukada, S. Matsuoka. **High-performance and Memory-saving Sparse General Matrix-Matrix Multiplication for NVIDIA Pascal GPU**. *ICPP17*.

M. Deveci, C. Trott, S. Rajamanickam. **Multi-threaded Sparse Matrix-Matrix Multiplication for Many-Core and GPU Architectures**. *PARCO. 2018*.

Diverse performance - SpTRSV



W. Liu, A. Li, J. D. Hogg, I. S. Duff, B. Vinter. **Fast Synchronization-Free Algorithms for Parallel Sparse Triangular Solves with Multiple Right-Hand Sides**. *CCPE. 2017.*

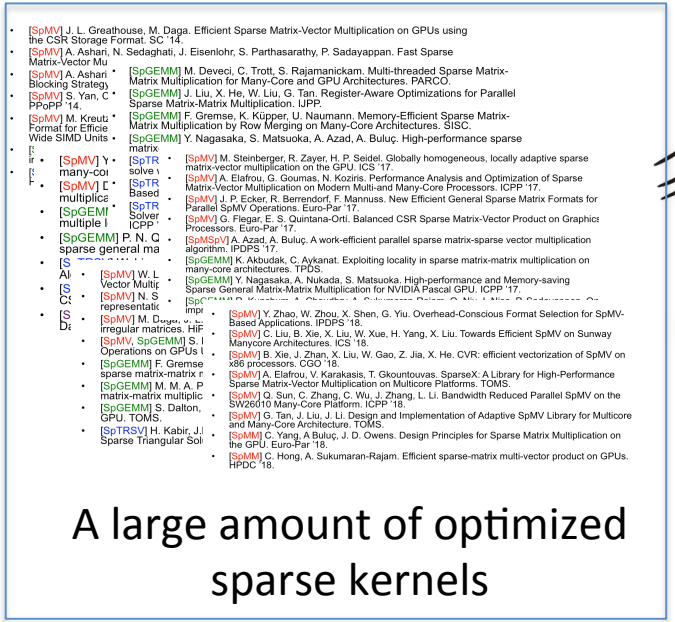
Some observations
2. Libraries get benefits from very limited kernels

Libraries get benefits from very limited kernels

- [MAGMA-SpMV] W. Liu, B. Vinter. CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication. ICS ’15.
- [MAGMA-SpTRSV] W. Liu, A. Li, J. D. Hogg, I. S. Duff, B. Vinter. A Synchronization-Free Algorithm for Parallel Sparse Triangular Solves. Euro-Par ’16.
- [Trilinos-SpGEMM] M. Deveci, C. Trott, S. Rajamanickam. Multi-threaded Sparse Matrix-Matrix Multiplication for Many-Core and GPU Architectures. PARCO. 2018.
- [Trilinos-SpTRSV] A. M. Bradley. A Hybrid Multithreaded Direct Sparse Triangular Solver. CSC ’16.
- [CombBLAS-SpMSpV] A. Azad, A. Buluç. A work-efficient parallel sparse matrix-sparse vector multiplication algorithm. IPDPS ’17.
- [CombBLAS-SpGEMM] A. Azad, G. Ballard, A. Buluc, J. Demmel, L. Grigori. Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. SISC. 2016.
- [clSPARSE-SpGEMM] W. Liu, B. Vinter. An efficient GPU general sparse matrix-matrix multiplication for irregular data. IPDPS ’14.
- [GHOST-SpMV] M. Kreutzer, G. Hager, G. Wellein, H. Fehske, A. Bishop. A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units. SISC.
- [ViennaCL-SpGEMM] F. Gremse, A. Hofter, L. O. Schwen, F. Kiessling, U. Naumann. GPU-accelerated sparse matrix-matrix multiplication by iterative row merging. SISC. 2015.
- [cuSPARSE-SpMV] D. Merrill, M. Garland. Merge-based parallel sparse matrix-vector multiplication. SC ’16.

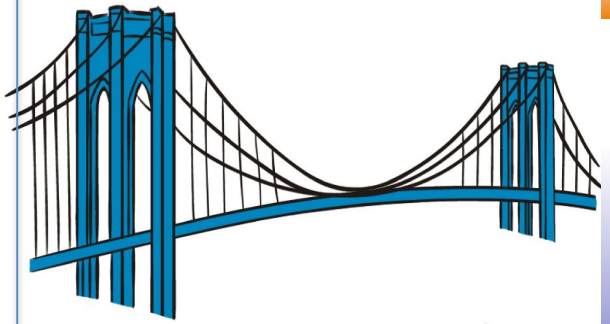
OpenSPARSE: An open platform for Sparse BLAS - objective, design and preliminary results

OpenSPARSE: Objective



- [SpMV] J. L. Greathouse, M. Daga. Efficient Sparse Matrix-Vector Multiplication on GPUs using the CSR Storage Format. SC '14.
- [SpMV] A. Ashari, N. Sedaghati, J. Eisenlohr, S. Parthasarathy, P. Sadayappan. Fast Sparse Matrix-Vector Mu
- [SpMV] A. Ashari Blocking Strateg
- [SpMV] S. Yan, C PPoPP '14.
- [SpMV] M. Kreutz Format for Efficie Wide SIMD Units
- [SpMV] Y many-co
- [SpMV] C multiplica
- [SpGEM multiple I
- [SpGEMM] P. N. Q sparse general ma
- [SpMV] W. L Vector Multi
- [SpMV] N. S representatic
- [SpMV] M. D irregular matrices. HIP
- [SpMV, SpGEMM] S. I Operations on GPUs I
- [SpGEMM] F. Gremse sparse matrix-matrix r
- [SpGEMM] M. M. A. P matrix-matrix multiplic
- [SpGEMM] S. Dalton, GPU. TOMS.
- [SpTRSV] H. Kabir, J. Sparse Triangular Sol
- [SpGEMM] M. Deveci, C. Trott, S. Rajamanickam. Multi-threaded Sparse Matrix-Matrix Multiplication for Many-Core and GPU Architectures. PARCO.
- [SpGEMM] J. Liu, X. He, W. Liu, G. Tan. Register-Aware Optimizations for Parallel Sparse Matrix-Matrix Multiplication. IJPP.
- [SpGEMM] F. Gremse, K. Küpper, U. Naumann. Memory-Efficient Sparse Matrix-Matrix Multiplication by Row Merging on Many-Core Architectures. SISC.
- [SpGEMM] Y. Nagasaka, S. Matsuoka, A. Azad, A. Buluç. High-performance sparse matrix
- [SpTR solve
- [SpTR Based
- [SpTR Solver ICPP
- [SpMV] M. Steinberger, R. Zayer, H. P. Seidel. Globally homogeneous, locally adaptive sparse matrix-vector multiplication on the GPU. ICS '17.
- [SpMV] A. Elafrou, G. Goumas, N. Koziris. Performance Analysis and Optimization of Sparse Matrix-Vector Multiplication on Modern Multi-and Many-Core Processors. ICPP '17.
- [SpMV] J. P. Ecker, R. Berrendorf, F. Mannuss. New Efficient General Sparse Matrix Formats for Parallel SpMV Operations. Euro-Par '17.
- [SpMV] G. Flegar, E. S. Quintana-Ortí. Balanced CSR Sparse Matrix-Vector Product on Graphics Processors. Euro-Par '17.
- [SpMSpV] A. Azad, A. Buluç. A work-efficient parallel sparse matrix-sparse vector multiplication algorithm. IPDPS '17.
- [SpGEMM] K. Akbudak, C. Aykanat. Exploiting locality in sparse matrix-matrix multiplication on many-core architectures. TPDS.
- [SpGEMM] Y. Nagasaka, A. Nukada, S. Matsuoka. High-performance and Memory-saving Sparse General Matrix-Matrix Multiplication for NVIDIA Pascal GPU. ICPP '17.
- [SpMV] Y. Zhao, W. Zhou, X. Shen, G. Yiu. Overhead-Conscious Format Selection for SpMV-Based Applications. IPDPS '18.
- [SpMV] C. Liu, B. Xie, X. Liu, W. Xue, H. Yang, X. Liu. Towards Efficient SpMV on Sunway Manycore Architectures. ICS '18.
- [SpMV] B. Xie, J. Zhan, X. Liu, W. Gao, Z. Jia, X. He. CVR: efficient vectorization of SpMV on x86 processors. CGO '18.
- [SpMV] A. Elafrou, V. Karakasis, T. Gkountouvas. SparseX: A Library for High-Performance Sparse Matrix-Vector Multiplication on Multicore Platforms. TOMS.
- [SpMV] Q. Sun, C. Zhang, C. Wu, J. Zhang, L. Li. Bandwidth Reduced Parallel SpMV on the SW26010 Many-Core Platform. ICPP '18.
- [SpMV] G. Tan, J. Liu, J. Li. Design and Implementation of Adaptive SpMV Library for Multicore and Many-Core Architecture. TOMS.
- [SpMM] C. Yang, A Buluç, J. D. Owens. Design Principles for Sparse Matrix Multiplication on the GPU. Euro-Par '18.
- [SpMM] C. Hong, A. Sukumaran-Rajam. Efficient sparse-matrix multi-vector product on GPUs. HPDC '18.

A large amount of optimized sparse kernels



OpenSPARSE: To build an open platform that bridges the gap between optimized sparse kernels and mathematical libraries.

Real-world applications

Mathematical libraries: MAGMA, Trilinos, CombBLAS, GraphBLAS, clSPARSE, GHOST, ViennaCL, ……

NTNU

OpenSPARSE: Design

- Language: C11
- Environments: OpenMP, CUDA, OpenCL, etc.
- Kernels: defined in Sparse BLAS with sparse/dense inputs/outputs.
- Basic matrix formats: DIA, COO, ELL, CSR, CSC, etc.
- Data types: BOOL, INT8/16/32/64, FP16/32/64, COMPLEX16/32/64, etc.
- Operators: multiplication/addition and other semirings in GraphBLAS.
- Code generator: Python scripts

OpenSPARSE: Matrix data structure

```
struct OSP_Matrix_struct
{
    bool initialized;
    int64_t nrows;
    int64_t ncols;
    int64_t nvals;
    OSP_Type ntype;
    OSP_Type itype;
    OSP_Type vtype;
    OSP_Format format;
    OSP_Matrix_DEN den;
    OSP_Matrix_COO coo;
    OSP_Matrix_DIA dia;
    OSP_Matrix_ELL ell;
    OSP_Matrix_CSR csr;
    OSP_Matrix_CSC csc;
    OSP_Matrix_CSR5 csr5;
};
```

```
struct OSP_Matrix_CSR_struct
{
    void *row_ptr;
    void *col_idx;
    void *val;
    bool row_ptr_shallow;
    bool col_idx_shallow;
    bool val_shallow;
};
```

```
OSP_Info OSP_Matrix_build_CSR_i64i16f32
(
    OSP_Matrix *matrix,
    int64_t *row_ptr,
    int16_t *col_idx,
    float *val,
    OSP_Index nvals,
    bool shallow
);
OSP_Info OSP_Matrix_build_CSR_i16i16f32(...);
OSP_Info OSP_Matrix_build_CSR_i32i32f32(...);
OSP_Info OSP_Matrix_build_CSR_i64i64f64(...);
OSP_Info OSP_Matrix_build_CSR_i32i16f16(...);
```

OpenSPARSE: An SpMV function

$$y = \alpha A x + \beta y$$

```
OSP_Info OSP_dVspMdVdV_f32_f32_i64i16f32_f64_f64_f32
(
        OSP_Vector yout,
    const OSP_Scalar alpha,
    const OSP_Matrix *matrix,
    const OSP_Vector x,
    const OSP_Scalar beta,
    const OSP_Vector yin
)
{
    OSP_Matrix A = *matrix;

    float *restrict youtval = yout->den->val;  // osp_zout_vtype
    const int64_t *restrict Arowptr = A->csr->row_ptr; // osp_x_ntype
    const int16_t *restrict Acolidx = A->csr->col_idx; // osp_x_itype
    const float *restrict Aval = A->csr->val;  // osp_x_vtype
    const double *restrict xval = x->den->val;  // osp_y_vtype
    const float *restrict yinval = yin->den->val;  // osp_zin_vtype
    const float *restrict alphaval = alpha->val;  // osp_a_vtype
    const double *restrict betaval = beta->val;  // osp_b_vtype
                                ...
```

```
...

    for (OSP_Index i = 0; i < A->nrows; i++)
    {
        for (int64_t j = Arowptr[i]; j < Arowptr[i+1]; j++)
        {
            float res = 0;
            int16_t col = Acolidx[j];
            float v1 = Aval[j];
            double v2 = xval[col];
            res = v1 * v2;
            res = alphaval * res;
            float v3 = betaval * yinval[i];
            res += v3;
            youtval[i] = res;
        }
    }
}
```

OpenSPARSE: A complete SpMV program

```c
#include "OpenSPARSE.h"

int main (int argc, char **argv)
{
    // create a csr matrix of size 6x8 including 9 nonzeros
    // /- - 1 - 3 - - -\
    // |- 2 - - - - - -|
    // |- - - - - 6 - -|
    // |- - - - - - - -|
    // |- 5 - - 4 - - 8|
    // \- - - 7 - - 9 -/
    OSP_Index Anrows = 6;
    OSP_Index Ancols = 8;
    OSP_Index Anvals = 9;
    int64_t ARowPtr[7] = {0, 2, 3, 4, 4, 7, 9};
    int16_t AColIdx[9] = {2, 4, 1, 5, 1, 4, 7, 3, 6};
    float AVal[9] = {1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0};

    // create dense vectors x of size 8 and y of size 6
    double xVal[8] = {1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0};
    float yVal[6] = {1.0, 1.0, 1.0, 1.0, 1.0, 1.0};

    // create two scalars: alpha = 1.0 and beta = 2.0
    float alphaVal = 1.0;
    double betaVal = 2.0;
```

...

...

```c
    // init OpenSPARSE
    OSP_init();

    // build two scalars, two vectors and a matrix in OpenSPARSE
    OSP_Matrix A = NULL;
    OSP_Matrix_new(&A, OSP_TYPE_INT64, OSP_TYPE_INT16, OSP_TYPE_FP32,
                   Anrows, Ancols);

    bool shallow = true;
    OSP_Matrix_build_CSR_i64i16f32(&A, ARowPtr, AColIdx, AVal, Anvals, shallow);

    OSP_Vector x = NULL, y = NULL;
    OSP_Vector_new(&x, NULL, OSP_TYPE_FP64, Ancols);
    OSP_Vector_build_DEN_f64(&x, xVal, shallow);

    OSP_Vector_new(&y, NULL, OSP_TYPE_FP32, Anrows);
    OSP_Vector_build_DEN_f32(&y, yVal, shallow);

    OSP_Scalar alpha = NULL, beta = NULL;
    OSP_Scalar_new(&alpha, OSP_TYPE_FP32);
    OSP_Scalar_build_f32(&alpha, alphaVal);

    OSP_Scalar_new(&beta, OSP_TYPE_FP64);
    OSP_Scalar_build_f64(&beta, betaVal);

    // call spmv function
    OSP_dVspMdVdV_f32_f32_i64i16f32_f64_f64_f32(y, alpha, &A, x, beta, y);
```

```c
    // free matrix, vectors and scalars
    OSP_Matrix_free(&A);
    OSP_Vector_free(&x);
    OSP_Vector_free(&y);
    OSP_Scalar_free(&alpha);
    OSP_Scalar_free(&beta);

    // finalize OpenSPARSE
    OSP_finalize();
}
```

OpenSPARSE: Add a new format

```
struct OSP_Matrix_struct
{
    bool initialized;
    int64_t nrows;
    int64_t ncols;
    int64_t nvals;
    OSP_Type ntype;
    OSP_Type itype;
    OSP_Type vtype;
    OSP_Format format;
    OSP_Matrix_DEN den;
    OSP_Matrix_COO coo;
    OSP_Matrix_DIA dia;
    OSP_Matrix_ELL ell;
    OSP_Matrix_CSR csr;
    OSP_Matrix_CSC csc;
    OSP_Matrix_CSR5 csr5;
};
```

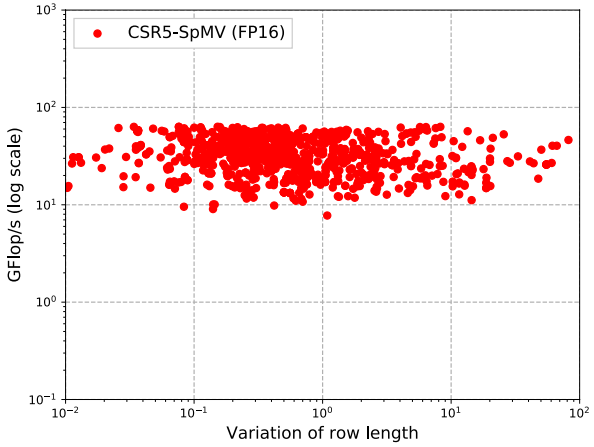
```
struct OSP_Matrix_CSR5_struct
{
    void *row_ptr;
    void *col_idx;
    void *val;
    OSP_Index omega;
    OSP_Index sigma;
    OSP_Index bit_y_off;
    OSP_Index bit_scansum_off;
    OSP_Index tail_partition_start;
    OSP_Index ntiles;
    OSP_Index p;
    void *tile_ptr;
    void *tile_desc;
    OSP_Index noffs;
    void *tile_desc_off_ptr;
    void *tile_desc_off;
    void *calibrator;
    bool row_ptr_shallow;
    bool col_idx_shallow;
    bool val_shallow;
    bool tile_ptr_shallow;
    bool tile_desc_shallow;
    bool tile_desc_off_ptr_shallow;
    bool tile_desc_off_shallow;
    bool calibrator_shallow;
};
```

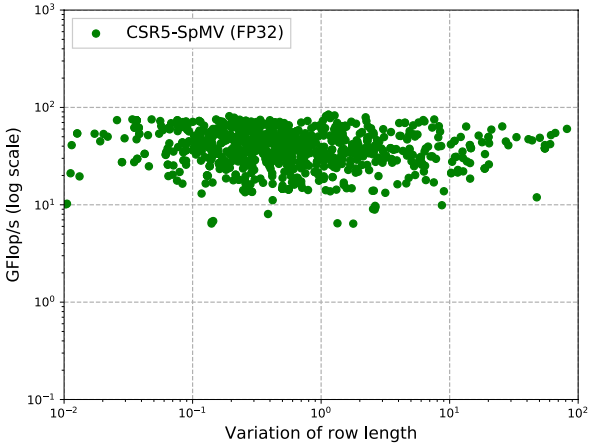
```
OSP_Info OSP_Matrix_build_CSR5_i64i16f32
(
    OSP_Matrix *matrix,
    int64_t *row_ptr,
    int16_t *col_idx,
    float *val,
    OSP_Index nvals,
    OSP_Index omega,
    OSP_Index sigma,
    bool shallow
);
```

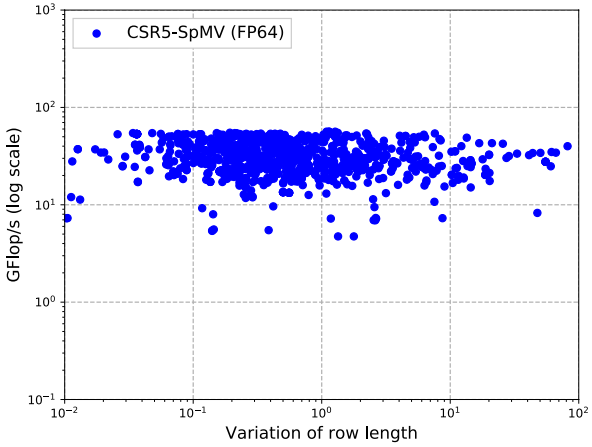
```
OSP_Info OSP_dVspMdVdV_f32_f32_i64i16f32_f64_f64_f32
(
        OSP_Vector yout,
    const OSP_Scalar alpha,
    const OSP_Matrix *matrix,
    const OSP_Vector x,
    const OSP_Scalar beta,
    const OSP_Vector yin
)
```

OpenSPARSE: Preliminary performance

- CSR5-SpMV performance in OpenSPARSE







Running 956 matrices on an NVIDIA Titan X Pascal.

We welcome your cooperation!

T k u !

0	4	8	9

A y Q s n s ?

0	2	4	7	11	12	13