clSPARSE: A Vendor-Optimized Open-Source Sparse BLAS Library

Joseph L. Greathouse[†] Kent Knox[†] Jakub Poła[‡§] Kiran Varaganti[†] Mayank Daga[†]

[†]Advanced Micro Devices, Inc. [‡]University of Wrocław [§]Vratis Ltd.

{Joseph.Greathouse, Kent.Knox, Kiran.Varaganti, Mayank.Daga}@amd.com jakub.pola@gmail.com

ABSTRACT

Sparse linear algebra is a cornerstone of modern computational science. These algorithms ignore the zero-valued entries found in many domains in order to work on much larger problems at much faster rates than dense algorithms. Nonetheless, optimizing these algorithms is not straightforward. Highly optimized algorithms for multiplying a sparse matrix by a dense vector, for instance, are the subject of a vast corpus of research and can be hundreds of times longer than naïve implementations. Optimized sparse linear algebra libraries are thus needed so that users can build applications without enormous effort.

Hardware vendors release proprietary libraries that are highly optimized for their devices, but they limit interoperability and promote vendor lock-in. Open libraries often work across multiple devices and can quickly take advantage of new innovations, but they may not reach peak performance. The goal of this work is to provide a sparse linear algebra library that offers both of these advantages.

We thus describe clSPARSE, a permissively licensed open-source sparse linear algebra library that offers state-of-the-art optimized algorithms implemented in OpenCL™. We test clSPARSE on GPUs from AMD and Nvidia and show performance benefits over both the proprietary cuSPARSE library and the open-source ViennaCL library.

CCS Concepts

•**Mathematics of computing** → **Computations on matrices; Mathematical software performance;**
•**Computing methodologies** → **Linear algebra algorithms;** *Graphics processors;*
•**Software and its engineering** → *Software libraries and repositories;*

Keywords

clSPARSE; OpenCL; Sparse Linear Algebra; GPGPU

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

IWOCL '16 April 19-21, 2016, Vienna, Austria

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4338-1/16/04.

DOI: http://dx.doi.org/10.1145/2909437.2909442

1. INTRODUCTION

Modern high-performance computing systems, including those used for physics [1] and engineering [5], rely on sparse data containing many zero values. Sparse data is best serviced by algorithms that store and rely only on the non-zero values. Such algorithms enable faster solutions by skipping many needless calculations and allow larger problems to fit into memory. Towards this end, the Basic Linear Algebra Subprograms (BLAS) specification dedicates a chapter to sparse linear algebra [3].

The description of these sparse algorithms is straightforward. While simple serial implementation of sparse matrix-dense vector multiplication (SpMV) can be described in six lines of pseudocode [6], highly optimized implementations require a great deal more effort. Langr and Tvrdík cite over fifty SpMV implementations from the recent literature, each ranging from hundreds to thousands of lines of code [8]. Su and Keutzer found that the best algorithm depended both on the input and on the target hardware [12].

Good sparse BLAS implementations must thus understand not just the data structures and algorithms but also the underlying hardware. Towards this end, hardware vendors offer optimized sparse BLAS libraries, such as Nvidia's cuSPARSE. These libraries are proprietary and offer limited or no support for hardware from other vendors. Their closed-source nature prevents even minor code or data structure modifications that would result in faster or more maintainable code.

In addition, the speed of advancement for these libraries can be limited because the community has no control over their code. Few (if any) of the algorithms described by Langr and Tvrdík made their way into closed-source libraries, and researchers do not know what optimizations or algorithms the vendors use. While proprietary, vendor-optimized libraries offer good performance for a small selection of hardware, they limit innovation and promote vendor lock-in.

Existing open-source libraries, such as ViennaCL [10], help alleviate these issues. ViennaCL offers support for devices from multiple vendors, demonstrating the benefits of open-source software in preventing vendor lock-in [11]. As an example of the benefits to open source innovation, ViennaCL added support for the new CSR-Adaptive algorithm within days of its publication [4], while additionally adding the new SELL-C-σ algorithm [7].

Nonetheless, as a library that must sometimes trade performance for portability concerns, ViennaCL can sometimes leave performance on the ground [2]. There are potential performance benefits to vendor-optimized libraries, includ-

ing the ability to preemptively add optimizations for hardware that is not yet released.

In summary, existing sparse BLAS libraries offer either limited closed-source support with high performance or broad open source support that is not necessarily completely optimized. This work sets out to solve these problems. We describe and test clSPARSE, a vendor-optimized, permissively licensed, open-source sparse BLAS library built using OpenCL™. clSPARSE was built in a collaboration between Advanced Micro Devices, Inc. and Vratis Ltd., and it is available at:
http://github.com/clMathLibraries/clSPARSE/

2. CLSPARSE

As part of AMD's GPUOpen initiative to bring strong open-source library support to GPUs, we recently released clSPARSE, an OpenCL™ sparse BLAS library optimized for GPUs. Like the clBLAS, clFFT, and clRNG libraries, clSPARSE is permissively licensed, works on GPUs from multiple vendors, and includes numerous performance optimizations, both from engineers within AMD and elsewhere. clSPARSE is built to work on both Linux® and Microsoft Windows® operating systems.

clSPARSE started as a collection of OpenCL routines developed by AMD and Vratis Ltd. to perform format conversions, level 1 vector operations, general SpMV, and solvers such as conjugate-gradient and biconjugate gradient. Further development led to the integration of AMD's CSR-Adaptive algorithm for SpMV [4, 2], a CSR-Adaptive-based sparse matrix-dense matrix multiplication algorithm, and a high-performance sparse matrix-sparse matrix multiplication developed by the University of Copenhagen [9].

clSPARSE has numerous benefits over proprietary libraries such as Nvidia's cuSPARSE. First and foremost, it works across devices from multiple vendors. As we will demonstrate in Section 3, clSPARSE achieves high performance on GPUs from both AMD and Nvidia, and the algorithms are portable to other OpenCL devices. In addition, the algorithms in clSPARSE, due to their open-source nature, are visible to users and easy to change. Even if users do not directly link against clSPARSE, these implementations can be directly used and modified by application developers. Such problem-specific kernel modifications has proven beneficial for techniques such as kernel fusion [11].

Compared to other open-source libraries, such as ViennaCL, clSPARSE offers significant performance benefits. For example, while ViennaCL uses an early version of the CSR-Adaptive algorithm for SpMV, the newer version used in clSPARSE performs better across a wider range of matrices [2]. We demonstrate this in Section 3.

If performance benefits alone do not outweigh application porting costs, the algorithms in clSPARSE could instead be directly brought into libraries like ViennaCL due to its permissive open-source licensing. Other libraries, such as ViennaCL and SciPy, are more general and include other algorithms besides those for sparse linear algebra. We feel that clSPARSE is a more focused effort dedicated to high-performance sparse linear algebra. It could easily serve as the basis for sparse BLAS in these libraries. In addition, unlike ViennaCL, clSPARSE includes a C interface to ease linking with complex C and FORTRAN software.

Table 1: Test Platforms

GPU	AMD Radeon™ Fury X	Nvidia GeForce GTX TITAN X
CPU	Intel Core i5-4690K	Intel Core i7-5960X
DRAM	16 GB Dual-channel DDR3-2133	64GB Quad-channel DDR4-2133
OS	Ubuntu 14.04.4 LTS	
Driver	fglrx 15.302	352.63
SDK	AMD APP SDK 3.0	CUDA 7.5
Library	ViennaCL v1.7.1	cuSPARSE v7.5
	clSPARSE v0.11	

3. PERFORMANCE COMPARISON

This section shows performance comparisons of our library against both a popular proprietary library, cuSPARSE, and a popular open-source library, ViennaCL. Our test platforms are described in Table 1. We test ViennaCL and clSPARSE on an AMD Radeon™ Fury X GPU, currently AMD's top-of-the-line consumer device. To show that clSPARSE works well on GPUs from multiple vendors, we also test an Nvidia GeForce GTX TITAN X, currently Nvidia's top-of-the-line consumer device. Due to space constraints, we only test this GPU on clSPARSE and cuSPARSE.

The inputs used in our tests are shown in Tables 2 and 3. Some of the matrices include explicit zero values from the input files. We measure runtime by taking a timestamp before calling the library's API, waiting for the work to finish, and then take another timestamp. We execute each kernel 1000 times on each input and take an average of the runtime. GFLOPs are calculated using the widely accepted metric of estimating 2 FLOPs for every output value.

Figure 1 shows the performance benefit of the clSPARSE SpMV compared to the cuSPARSE `csrmv()` algorithm. This data shows that clSPARSE on the AMD GPU yields an average performance increase of 5.4× over cuSPARSE running on the Nvidia GPU. While this demonstrates the benefits of AMD-optimized software on AMD hardware, it's also interesting to note that clSPARSE results in a performance increase of 4.5× on the Nvidia GPU when compared against the proprietary Nvidia library. This shows that clSPARSE is beneficial across vendors.

Figure 2 illustrates the performance of the SpMV algorithm in clSPARSE versus the ViennaCL SpMV algorithm on the AMD GPU. It is worth noting that ViennaCL uses an earlier version of the CSR-Adaptive algorithm used in clSPARSE. However, clSPARSE includes both new algorithmic changes [2] and numerous minor performance tuning changes. Through these changes, clSPARSE is able to achieve a 2.5× higher performance than ViennaCL.

Figure 3 and 4 show similar comparisons between libraries and cards, but using sparse matrix-sparse matrix multiplication (SpMSpM) algorithms. These results show that, while cuSPARSE yields better SpMSpM performance for some input matrices (like Protein), the average performance between clSPARSE and cuSPARSE is comparable because of inputs like amazon0312. In addition, we see that clSPARSE is 27% faster than the open-source ViennaCL.

These results demonstrate that that the performance of clSPARSE compares favorably to proprietary libraries while maintaining the benefits open-source libraries. Contributions to clSPARSE from the community are welcome, as are feedback, feature requests, and bug reports.

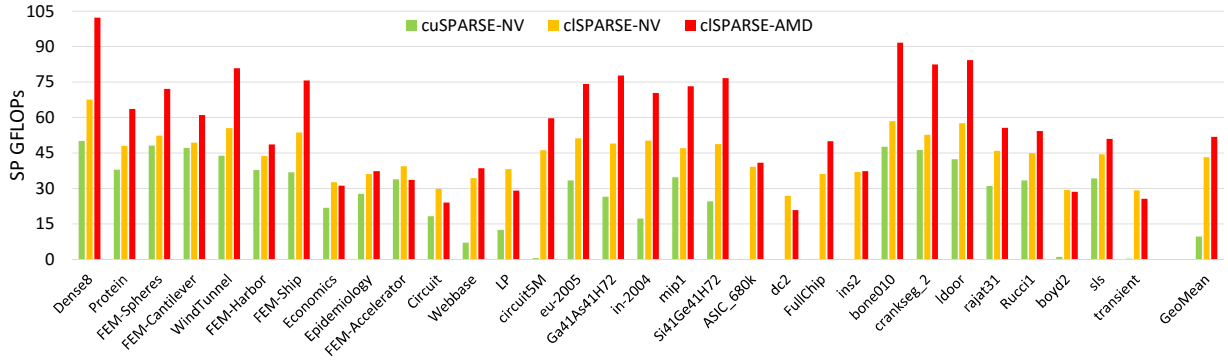

Figure 1: Performance comparison of single-precision SpMV on an Nvidia GeForce GTX TITAN X GPU (using both cuSPARSE and clSPARSE) and an AMD Radeon™ Fury X GPU (using clSPARSE).

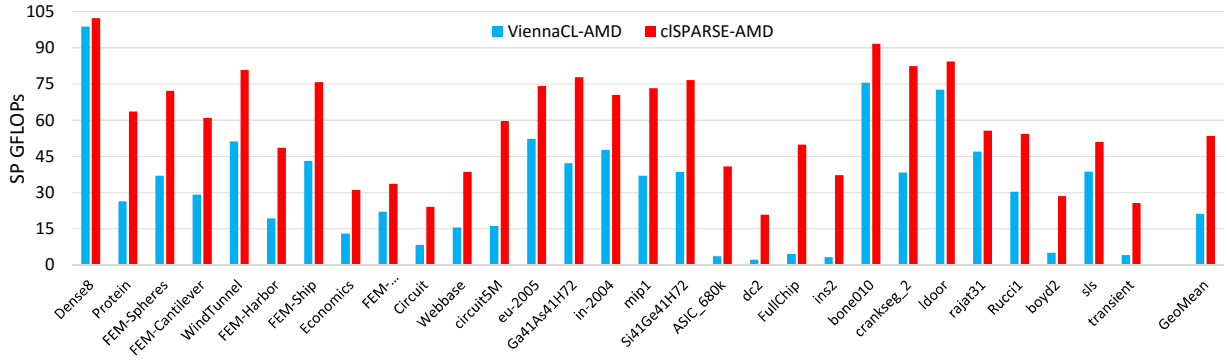

Figure 2: Performance comparison of single-precision SpMV between the open source ViennaCL library and clSPARSE on an AMD Radeon™ Fury X GPU.

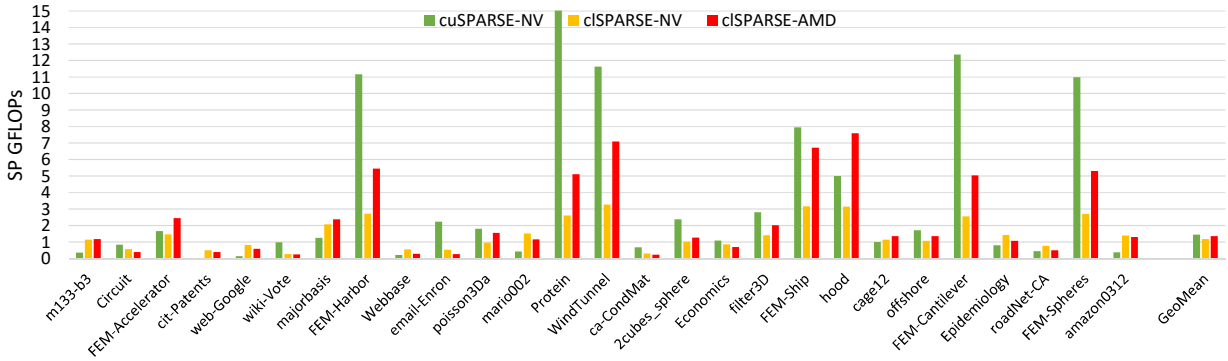

Figure 3: Performance comparison of single-precision SpMSpM on an Nvidia GeForce GTX TITAN X GPU (using both cuSPARSE and clSPARSE) and an AMD Radeon™ Fury X GPU (using clSPARSE).

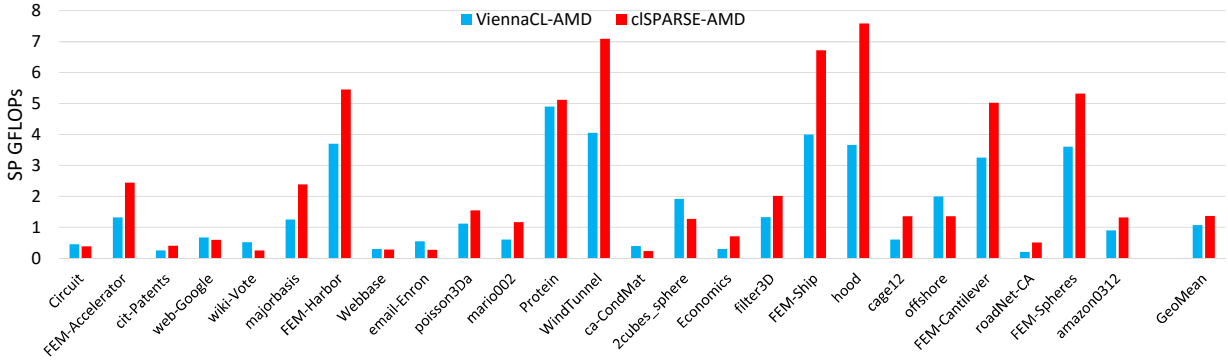

Figure 4: Performance comparison of single-precision SpMSpM between the open source ViennaCL library and clSPARSE on an AMD Radeon™ Fury X GPU.

Table 2: Matrices used for the SpMV evaluation

Name	Size	NNZ	NNZ /row	Max
Dense8	8K * 8K	8 M	8,000	8,000
Protein	36K * 36K	4.3 M	119	204
FEM/Spheres	83K * 83K	6.0 M	72	81
FEM/Cantilever	62K * 62K	4.0 M	64	78
Wind Tunnel	218K * 218K	11.6 M	53	180
FEM/Harbor	47K * 47K	2.4 M	51	145
FEM/Ship	141K * 141K	7.8 M	55	102
Economics	207K * 207K	1.3 M	6	44
Epidemiology	526K * 526K	2.1 M	4	4
FEM/Accelerator	121K * 121K	2.6 M	22	81
Circuit	171K * 171K	0.96 M	6	353
Webbase	1,000K * 1,000K	3.1 M	3	4.7 K
LP	4K * 1,097K	11.3 M	2,634	56 K
circuit5M	5,558K * 5,558K	59.5 M	11	1.29 M
eu-2005	863K * 863K	19.2 M	22	4.2 K
Ga41As41H72	268K * 268K	18.4 M	69	702
in-2004	1,383K * 1,383K	16.9 M	12	7.8 K
mip1	66K * 66K	10.4 M	156	66 K
Si41Ge41H72	186K * 186K	15.0 M	81	662
ASIC_680k	683K * 683K	3.9 M	6	395 K
dc2	117K * 117K	0.77 M	7	114 K
FullChip	2,897K * 2,897K	26.6 M	9	2.3 M
ins2	309K * 309K	2.8 M	9	309 K
bone010	986K * 986K	47.9 M	48	81
crankseg_2	121K * 121K	2.6 M	22	3.4 K
ldoor	952K * 952K	42.4 M	45	77
rajat31	4,690K * 4,690K	20.3 M	4	1.3 K
Rucci1	1,978K * 109K	7.8 M	4	5
boyd2	466K * 466K	1.5 M	3	93 K
sls	1,748K * 63K	6.8 M	4	5
transient	179K * 179K	0.96 M	5	60 K

Table 3: Matrices used for the SpMSpM evaluation

Name	Size	NNZ
m133-b3	200K * 200K	800,800
Circuit	171K * 171K	958,936
FEM/Accelerator	121K * 121K	2,624,331
cit-Patents	3774K * 3774K	16,518,948
web-Google	916K * 916K	5,105,039
wiki-Vote	8,297 * 8,297	103,689
majorbasis	160K * 160K	1,750,416
FEM/Harbor	47K * 47K	2,374,001
Webbase	1,000K * 1,000K	3,105,536
email-Enron	37K * 37K	367,662
poisson3Da	13.5K * 13.5K	352,762
mario002	390K * 390K	2,101,242
Protein	36K * 36K	4,344,765
WindTunnel	218K * 218K	11,634,424
ca-CondMat	23K * 23K	186,936
2cubes_sphere	101K * 101K	1,647,264
Economics	207K * 207K	1,273,389
filter3D	106K * 106K	2,707,179
FEM/Ship	141K * 141K	7,813,404
hood	221K * 221K	10,768,436
cage12	130K * 130K	2,032,536
offshore	260K * 260K	4,242,673
FEM/Cantilever	62K * 62K	4,007,383
Epidemiology	526K * 526K	2,100,225
roadNet-CA	1,971K * 1,971K	5,533,214
FEM/Spheres	83K * 83K	6,010,480
amazon0312	401K * 401K	3,200,440

AMD, the AMD Arrow logo, AMD Radeon, and combinations thereof are trademarks of Advanced Micro Devices, Inc. OpenCL is a trademark of Apple, Inc. used by permission by Khronos. Windows is a registered trademark of Microsoft Corporation. Linux is a registered trademark of Linus Torvalds. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

4. REFERENCES

[1] H. M. Aktulga, A. Buluç, S. Williams, and C. Yang. Optimizing Sparse Matrix-Multiple Vectors Multiplication for Nuclear Configuration Interaction Calculations. In *Proc. of the Int'l Parallel and Distributed Processing Symposium (IPDPS)*, 2014.

[2] M. Daga and J. L. Greathouse. Structural Agnostic SpMV: Adapting CSR-Adaptive for Irregular Matrices. In *Proc. of the Int'l Conf. on High Performance Computing (HiPC)*, 2015.

[3] I. S. Duff, M. A. Heroux, and R. Pozo. An Overview of the Sparse Basic Linear Algebra Subprograms: The New Standard from the BLAS Technical Forum. *Trans. on Mathematical Software*, 28(2):239–267, 2002.

[4] J. L. Greathouse and M. Daga. Efficient Sparse Matrix-Vector Multiplication on GPUs Using the CSR Storage Format. In *Proc. of the Int'l Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.

[5] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith. Towards Realistic Performance Bounds for Implicit CFD Codes. In *Proc. of the Int'l Parallel Computational Fluid Dynamics Conf. (PARCFD)*, 1999.

[6] A. Kaiser, S. Williams, K. Madduri, K. Ibrahim, D. H. Bailey, J. W. Demmel, and E. Strohmaier. TORCH Computational Reference Kernels: A Testbed for Computer Science Research. Technical Report LBNL-4172E, Lawrence Berkeley National Laboratory, 2010.

[7] M. Kreutzer, G. Hager, G. Wellein, H. Fehske, and A. R. Bishop. A Unified Sparse Matrix Data Format for Modern Processors with Wide SIMD Units. *SIAM Journal on Scientific Computing*, 36(5):C401–C423, 2014.

[8] D. Langr and P. Tvrdík. Evaluation Criteria for Sparse Matrix Storage Formats. *IEEE Trans. on Parallel and Distributed Systems*, 27(2):428–440, Feb. 2016.

[9] W. Liu and B. Vinter. An Efficient GPU General Sparse Matrix-Matrix Multiplication for Irregular Data. In *Proc. of the Int'l Parallel and Distributed Processing Symp. (IPDPS)*, 2014.

[10] K. Rupp, F. Rudolf, and J. Weinbub. ViennaCL - A High Level Linear Algebra Library for GPUs and Multi-Core CPUs. In *Int'l Workshop on GPUs and Scientific Applications (GPUScA)*, 2010.

[11] K. Rupp, J. Weinbub, A. Jüngel, and T. Grasser. Pipelined Iterative Solvers with Kernel Fusion for Graphics Processing Units. *CoRR*, abs/1410.4054, 2014.

[12] B.-Y. Su and K. Keutzer. clSpMV: A Cross-Platform OpenCL SpMV Framework on GPUs. In *Proc. of the Int'l Conf. on Supercomputing (ICS)*, 2012.