TR-2015-02

Comparison of SPMV performance on matrices with different matrix format using CUSP, cuSPARSE and ViennaCL

Ang Li, Hammad Mazhar, Radu Serban, Dan Negrut

February 10, 2015

Contents

1 Introduction **2**

2 Simulation **2**

2.1 Environment 2

2.2 Results 2

2.2.1 Performance comparison among libraries 2

2.2.2 Performance comparison among matrix formats 3

1 Introduction

ViennaCL is a free open-source linear algebra library for computations on many-core architectures (GPUs, MIC) and multi-core CPUs. The library is written in C++ and supports CUDA, OpenCL, and OpenMP. In addition to core functionality and many other features including BLAS level 1-3 support and iterative solvers, the latest release family ViennaCL 1.6.x provides fast pipelined iterative solvers including fast sparse matrix-vector products based on CSR-adaptive, a new fully HTML-based documentation, and a new sparse matrix type. Also, a Python wrapper named PyViennaCL is available [2].

CUSP is an open source C++ library of generic parallel algorithms for sparse linear algebra and graph computations on CUDA architecture GPUs. CUSP provides a flexible, high-level interface for manipulating sparse matrices and solving sparse linear systems [5].

The NVIDIA CUDA Sparse Matrix library (`cuSPARSE`) provides a collection of basic linear algebra subroutines used for sparse matrices that delivers up to 8x faster performance than the latest MKL. The `cuSPARSE` library is designed to be called from C or C++, and the latest release includes a sparse triangular solver [1].

2 Simulation

2.1 Environment

Hardware. We ran all simulations on Intel Xeon E5-2690 V2 CPU and NVIDIA's K20x GPU. Mazhar and Negrut provided in [7] details of Intel Xeon and K20x[1].

Software. We are comparing among SPMVs in three libraries: `CUSP`, `cuSPARSE` and `Vienna CL`. For `CUSP` and `Vienna CL`, SPMV performance comparison were carried out in four matrix formats: CSR, COO, ELL and HYB. For `cuSPARSE`, we only concerned matrices in CSR format.

Out of all 131 application matrices, 127 come from University of Florida Matrix Collection [6], and the other four come from Flexible Multibody Dynamics problems in the Simulation Based Engineering Lab (SBEL) [3].

2.2 Results

2.2.1 Performance comparison among libraries

CSR matrices. Fig. 1 and 2 show the comparison of SPMV performance between `CUSP` and `cuSPARSE`. We observed that for 93 out of 131 application matrices, `cuSPARSE` outperforms `CUSP`.

Fig. 3 and 4 show the comparison of SPMV performance between `CUSP` and `Vienna CL`. We observed that for 113 out of 131 application matrices, `CUSP` outperforms `Vienna CL`.

[1]The compiler and compiler flags are different in our simulation. We used GCC 4.4.6 and Level 2 optimization.

Matrices in other formats.

Fig. 5 and 6 show the comparison of SPMV performance between `CUSP` and `Vienna CL` for ELL matrices. We observed that for around 76 out of 118 application matrices, `CUSP` outperforms `Vienna CL`. For 13 matrices, both implementations run out of memory.

Fig. 7 and 8 show the comparison of SPMV performance between `CUSP` and `Vienna CL` for HYB matrices. We observed that for around half (70) of 131 application matrices, `CUSP` outperforms `Vienna CL`.

Fig. 9 shows the comparison of SPMV performance between `CUSP` and `cuSPARSE` for HYB matrices.

2.2.2 Performance comparison among matrix formats

Besides the comparison among the implementations of SPMV for a certain matrix format in different libraries, we also intend to get the knowledge of matrices in which formats are of the highest potential of obtaining the best performance of SPMV.

We do our comparison with both `CUSP` and `Vienna CL` implementations, with the exception that there is no comparison between CSR format and COO format with `Vienna CL` implementation, in that `Vienna CL` implementation of COO SPMV is significantly slower. And we compare SPMV performance of all other matrix formats against SPMV performance of CSR format, which is used in `SPIKE::GPU`, a sparse linear system on GPU in SBEL [4].

COO vs. CSR. Fig. 10 shows the comparison of SPMV performance with `CUSP` implementation between matrices in CSR and COO formats.

ELL vs. CSR. Fig. 11 shows the comparison of SPMV performance with `CUSP` implementation between matrices in ELL and CSR formats.

Fig. 12 shows the comparison of SPMV performance with `Vienna CL` implementation between matrices in ELL and CSR formats.

HYB vs. CSR. Fig. 13 shows the comparison of SPMV performance with `CUSP` implementation between matrices in HYB and CSR formats.

Fig. 14 shows the comparison of SPMV performance with `Vienna CL` implementation between matrices in HYB and CSR formats.

References

[1] cuSPARSE. https://developer.nvidia.com/cuSPARSE.

[2] ViennaCL Library. http://viennacl.sourceforge.net.

[3] Simulation Based Engineering Lab. http://sbel.wisc.edu, 2006.

[4] SPIKE GPU: An implementation of a recursive divide-and-conquer parallel strategy for solving large systems of linear equations. http://spikegpu.sbel.org, 2013.

[5] N. Bell and M. Garland. Cusp: Generic parallel algorithms for sparse matrix and graph computations, 2012. Version 0.3.0.

[6] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[7] M. H. and N. D. Comparison of opencl performance on different platforms. Technical Report TR-2015-01, Simulation Based Engineering Lab, University of Wisconsin-Madison, 2015.

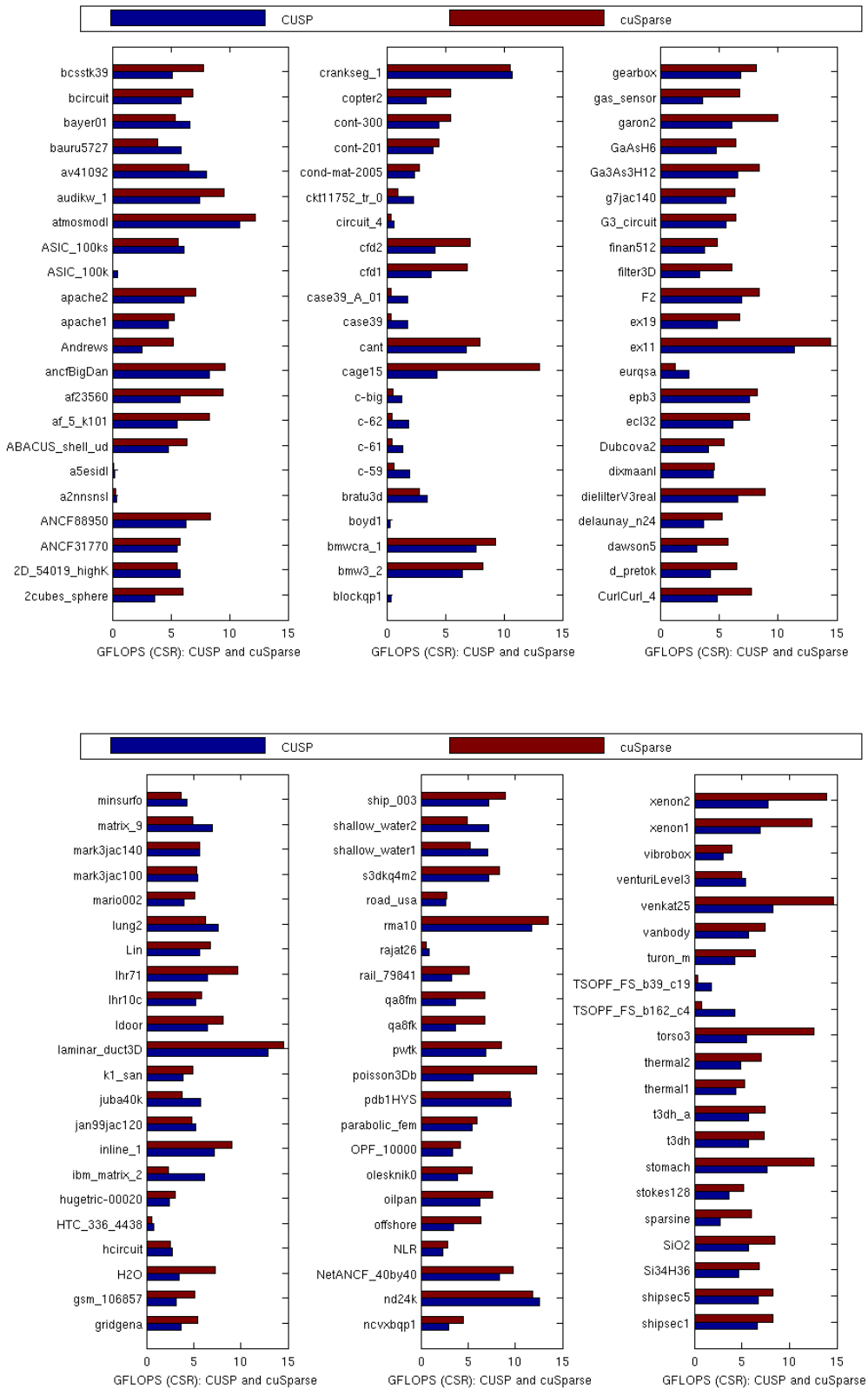

Figure 1: SPMV GFLOPS of `CUSP` and `cuSPARSE`. Matrices are in CSR format.

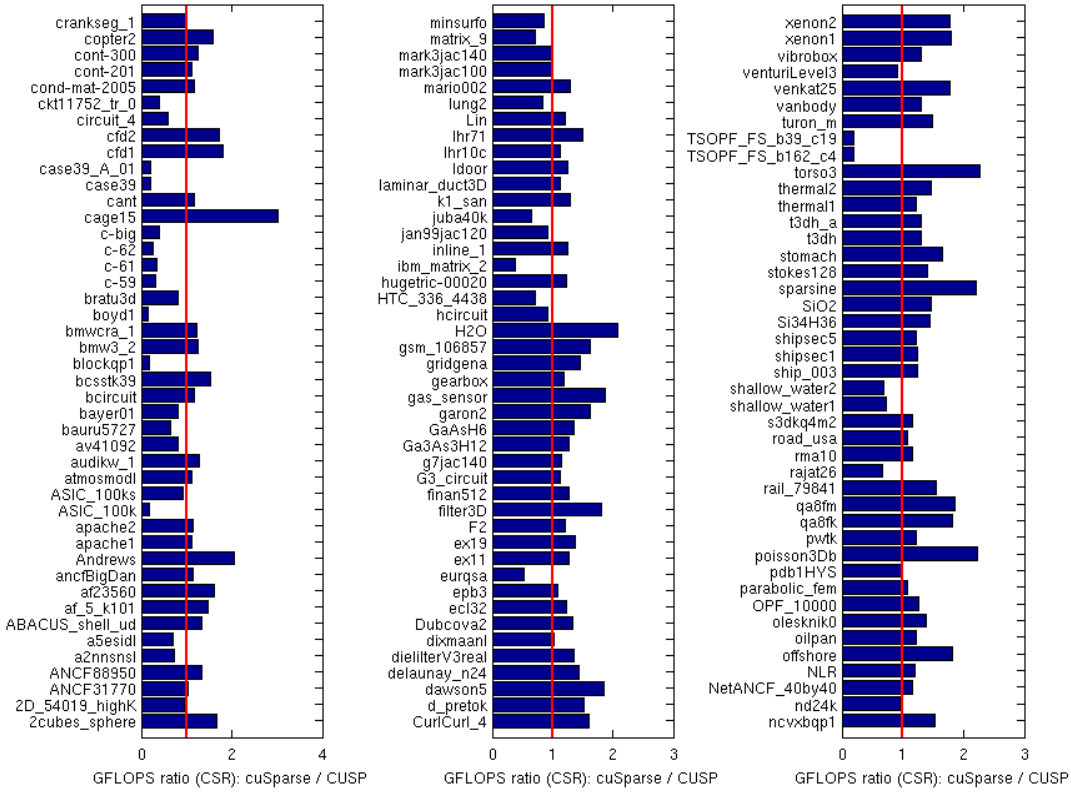

Figure 2: SPMV GFLOPS ratio of `cuSPARSE` over `CUSP`. Matrices are in CSR format.

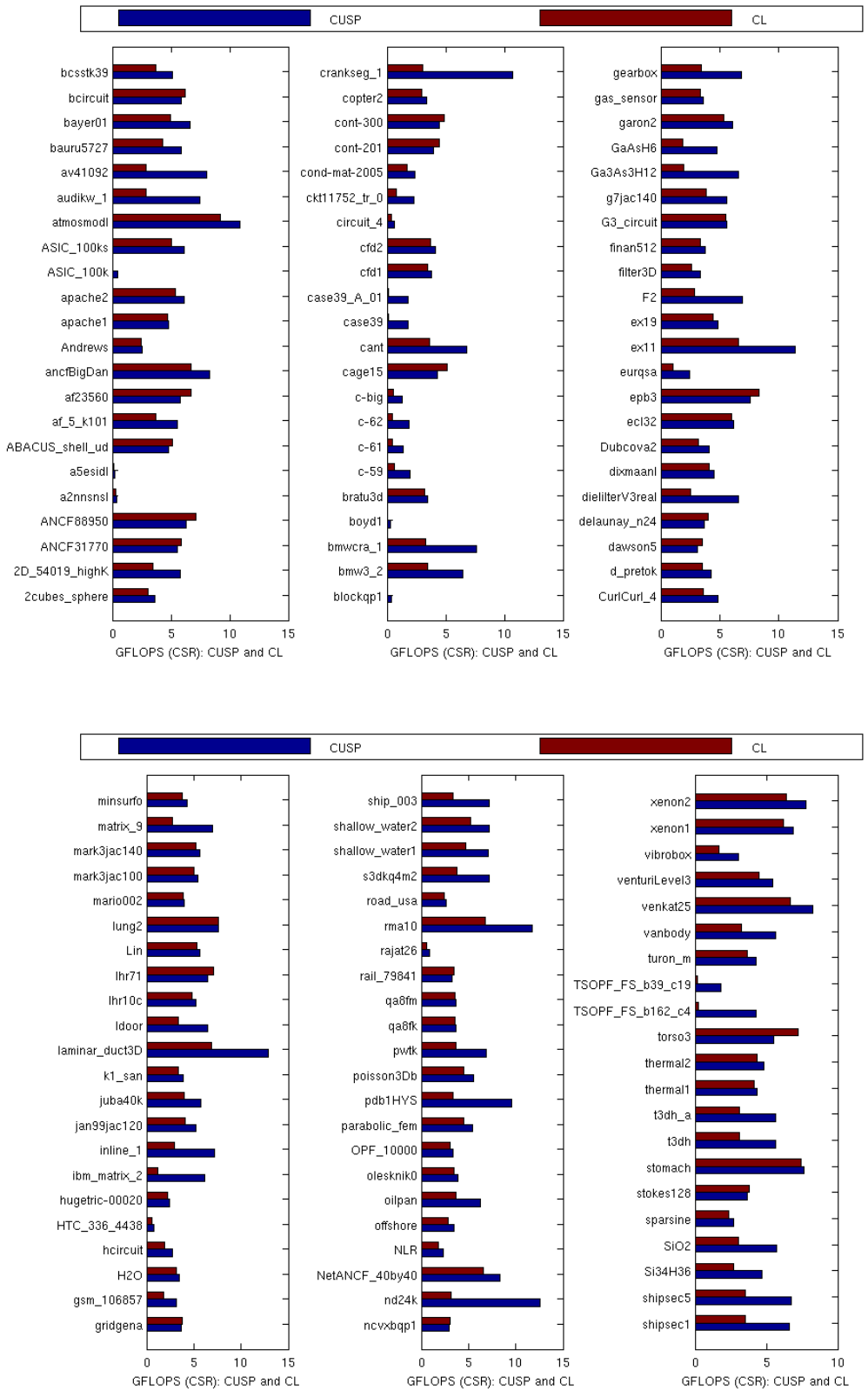

Figure 3: SPMV GFLOPS of `CUSP` and `Vienna CL`. Matrices are in CSR format.

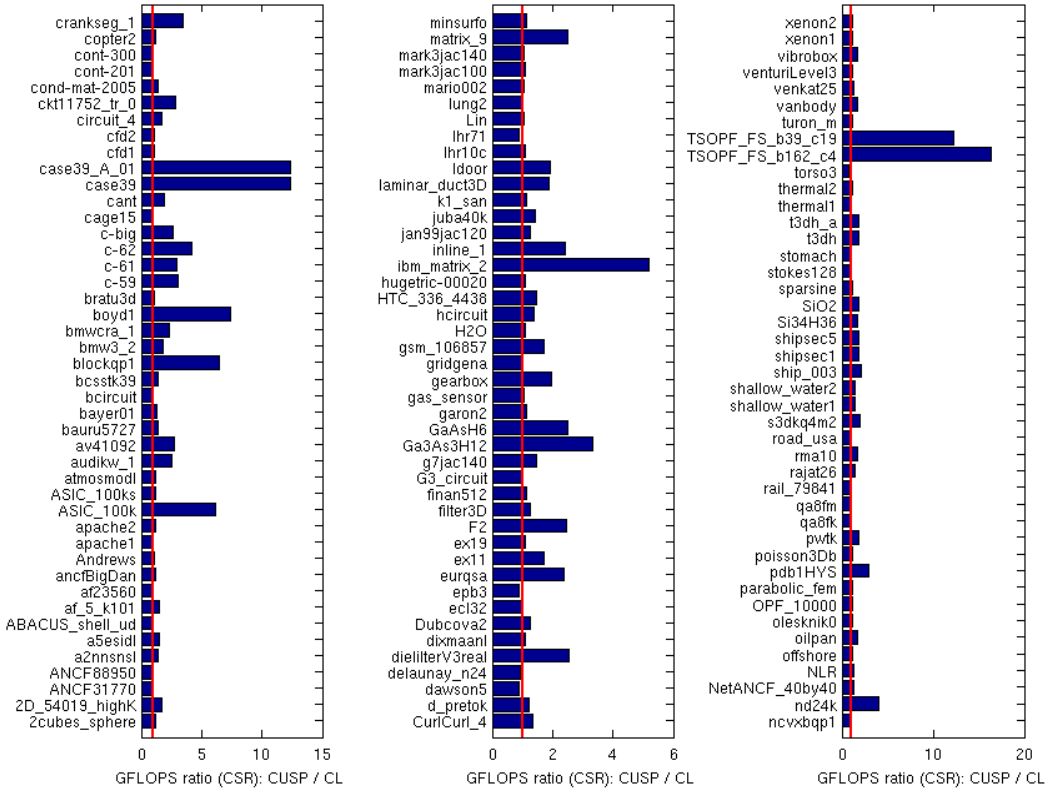

Figure 4: SPMV GFLOPS ratio of `CUSP` over `Vienna CL`. Matrices are in CSR format.

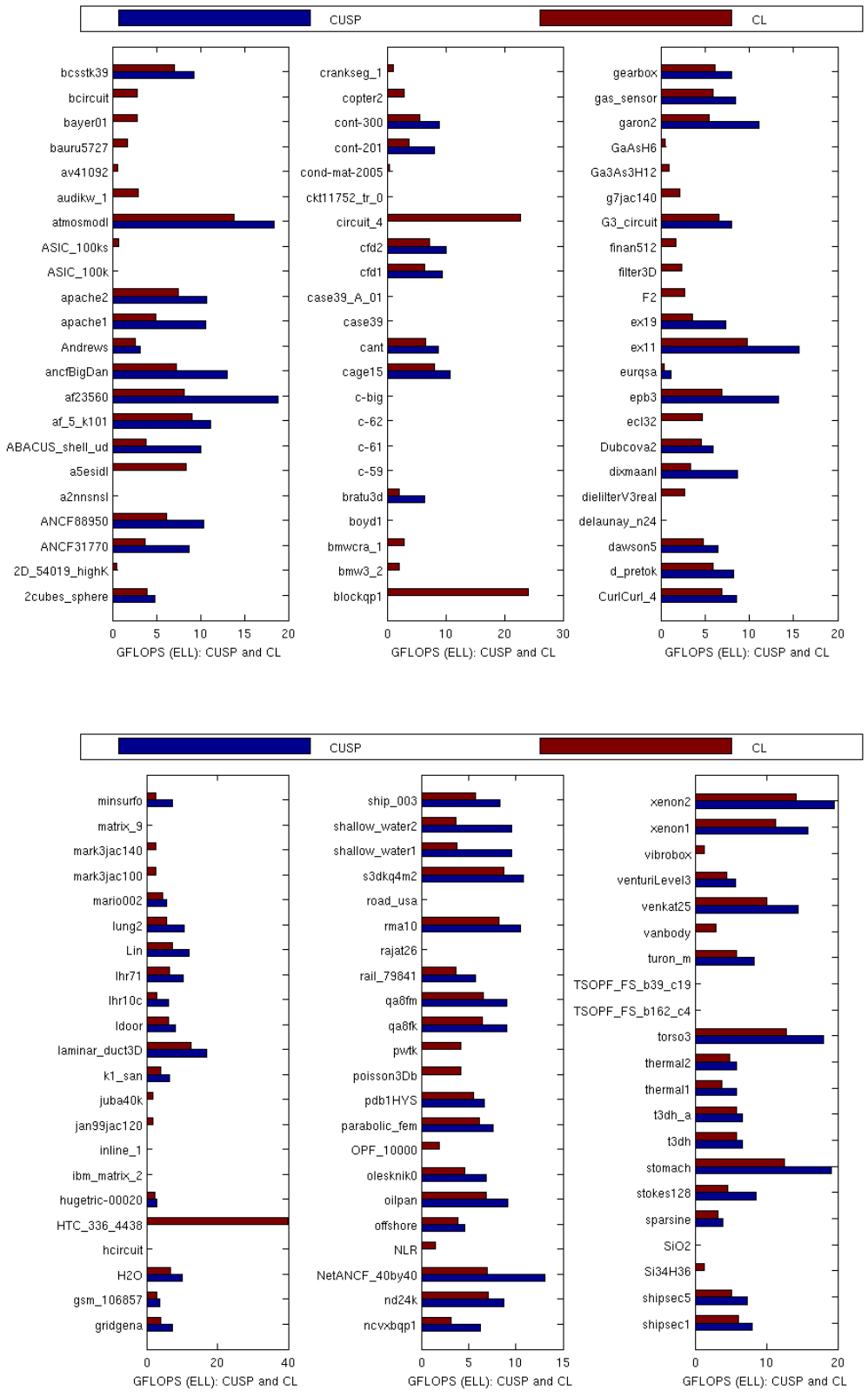

Figure 5: SPMV GFLOPS of `CUSP` and `Vienna CL`. Matrices are in ELL format.

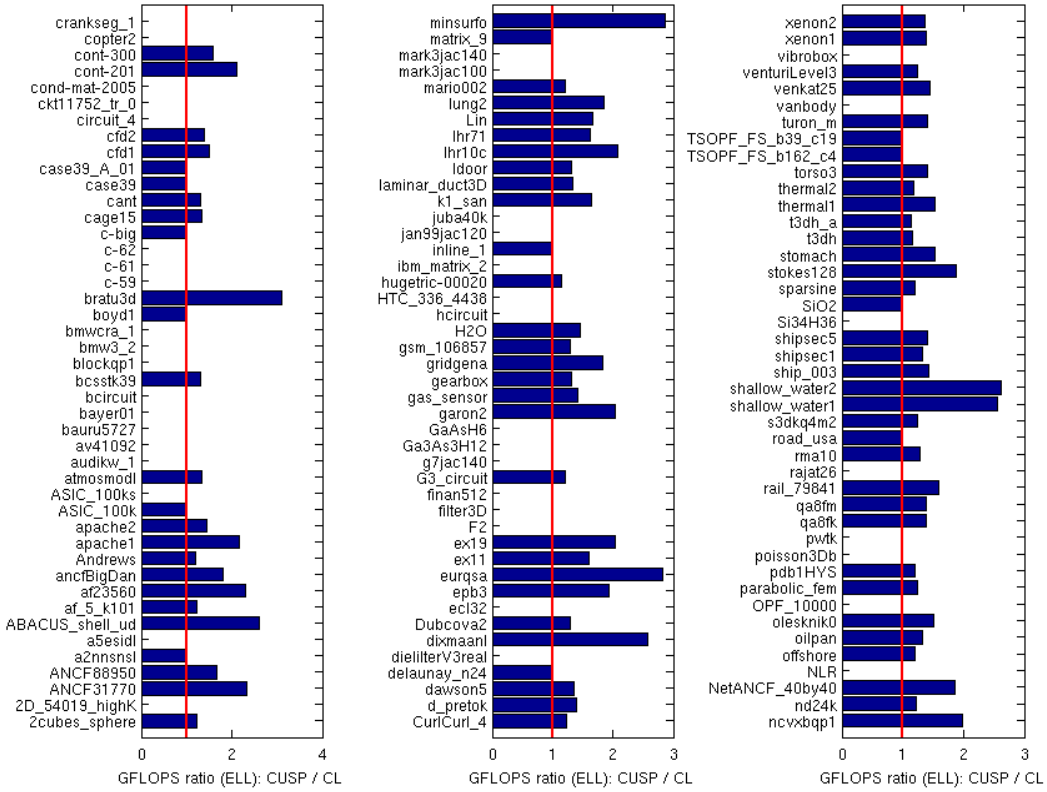

Figure 6: SPMV GFLOPS ratio of `CUSP` over `Vienna CL`. Matrices are in ELL format.

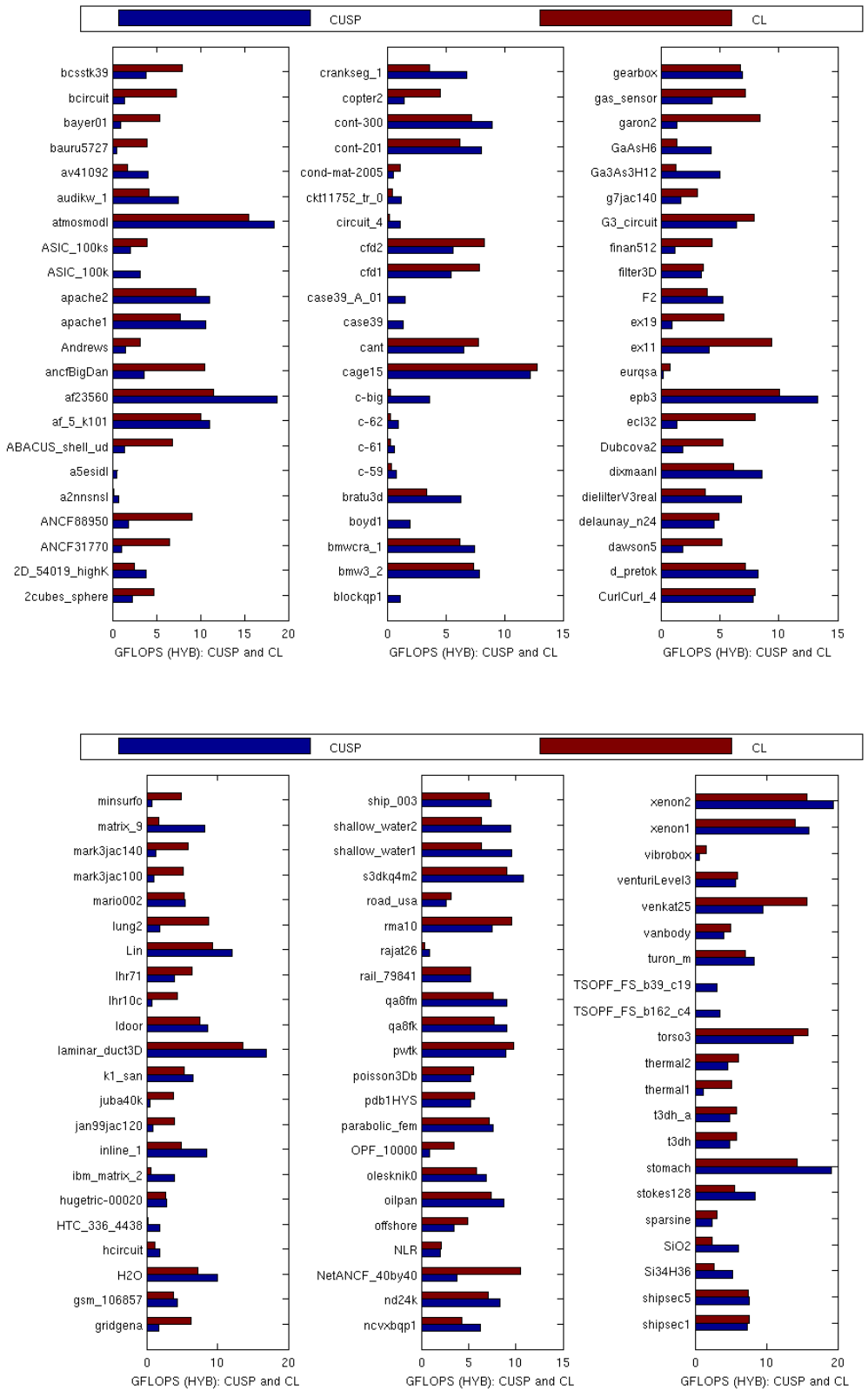

Figure 7: SPMV GFLOPS of `CUSP` and `Vienna CL`. Matrices are in HYB format.

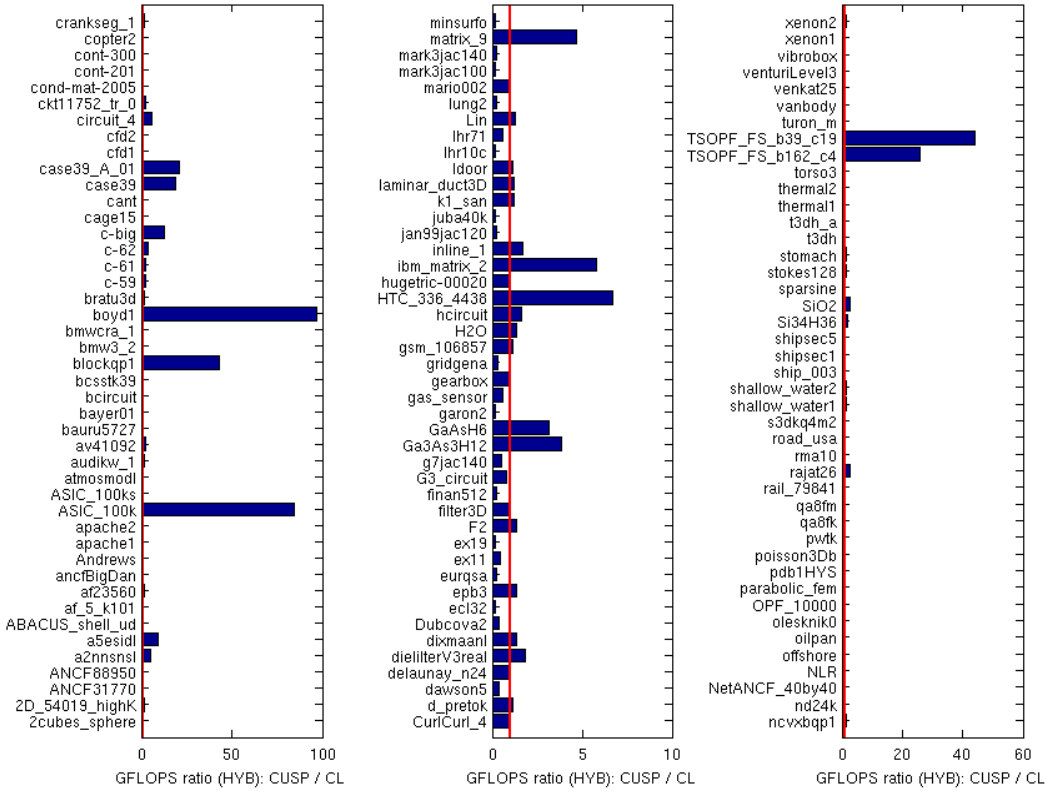

Figure 8: SPMV GFLOPS ratio of `CUSP` over `Vienna CL`. Matrices are in HYB format.

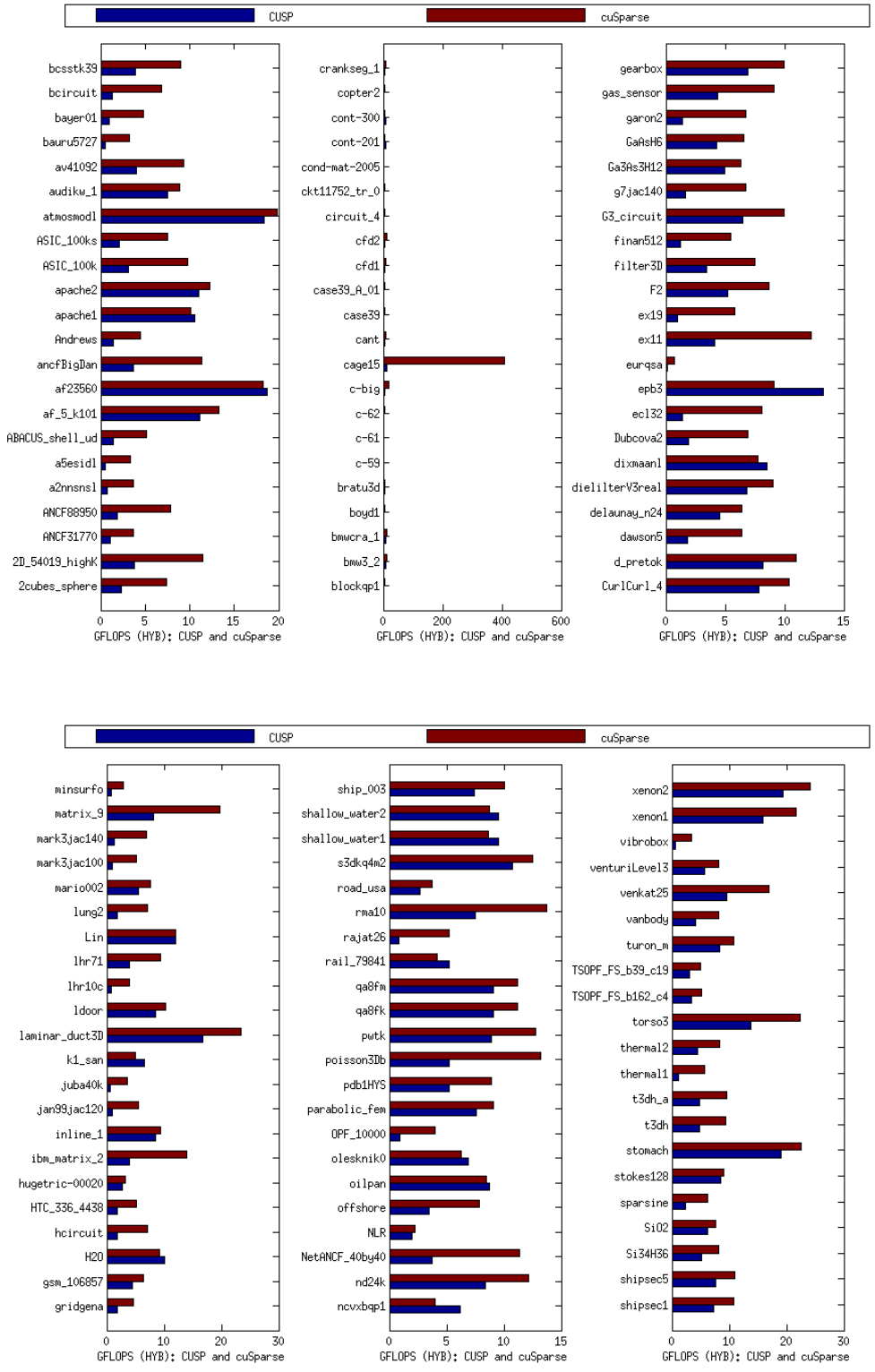

Figure 9: SPMV GFLOPS of `CUSP` and `cuSPARSE`. Matrices are in HYB format.

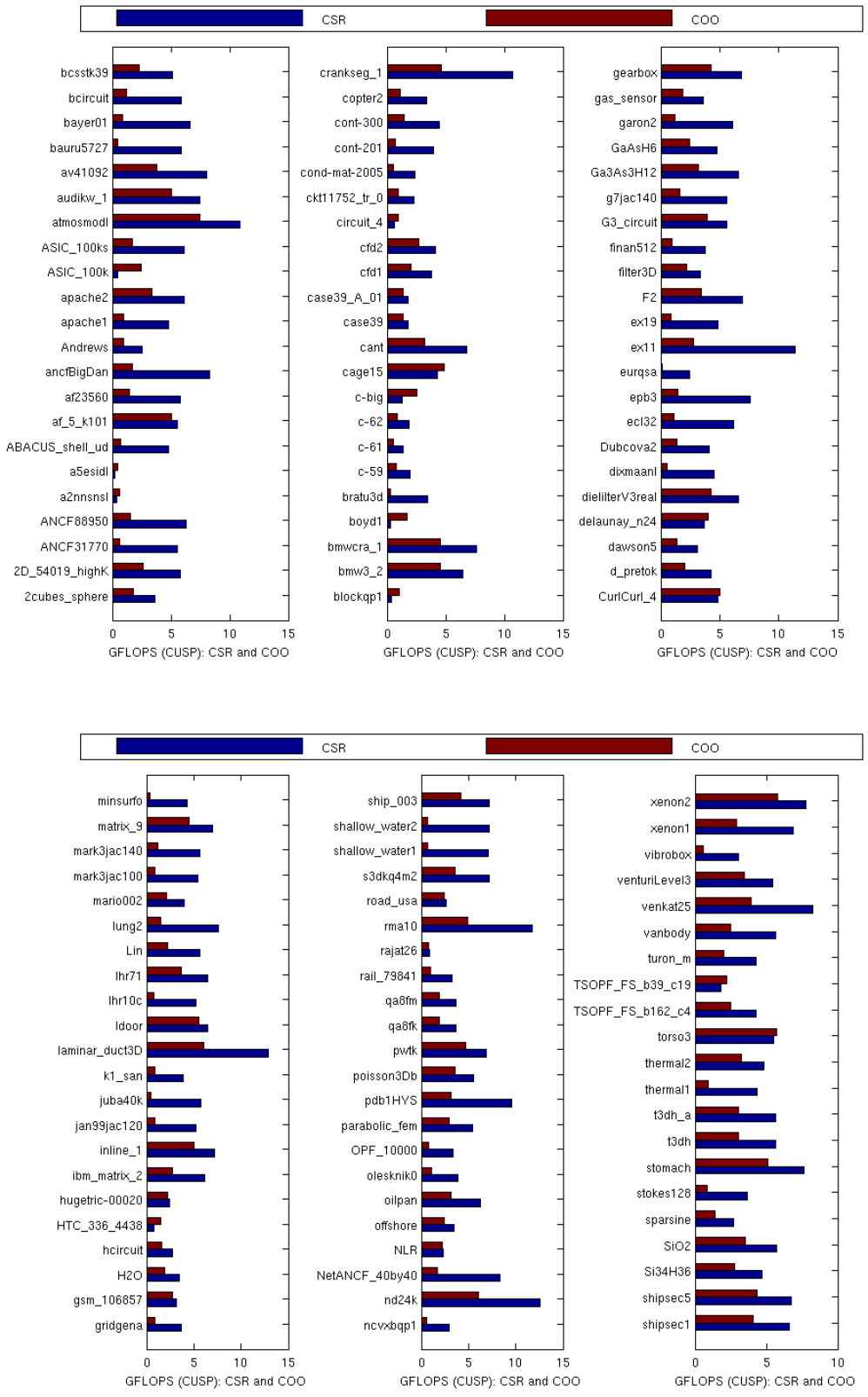

Figure 10: SPMV GFLOPS of matrices in CSR and COO formats using `CUSP` implementation.

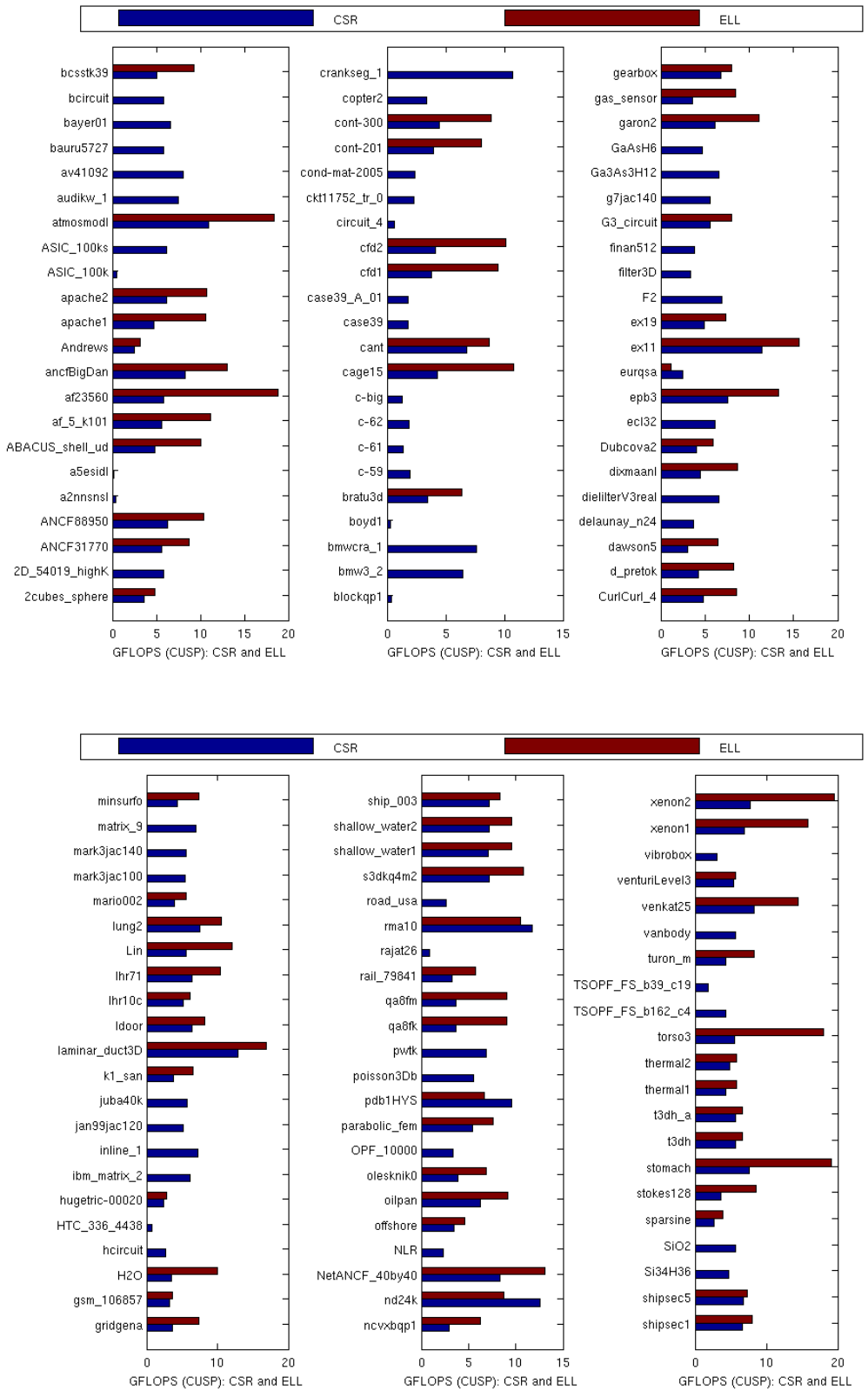

Figure 11: SPMV GFLOPS of matrices in CSR and ELL formats using `CUSP` implementation.

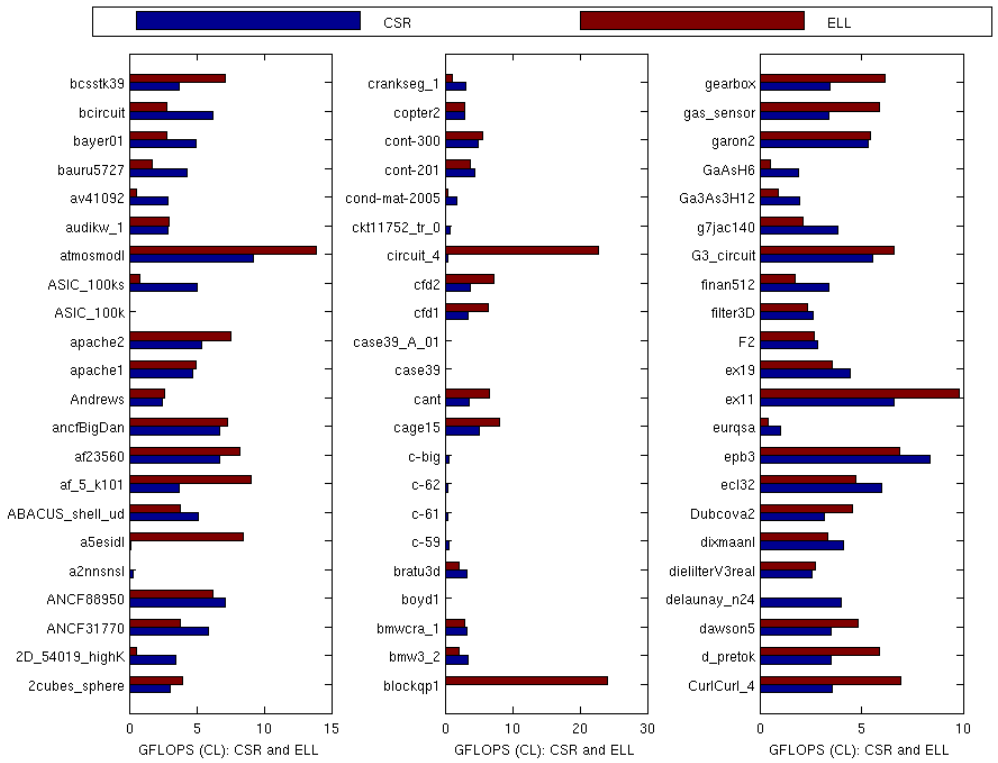

Figure 12: SPMV GFLOPS of matrices in CSR and ELL formats using `Vienna CL` implementation.

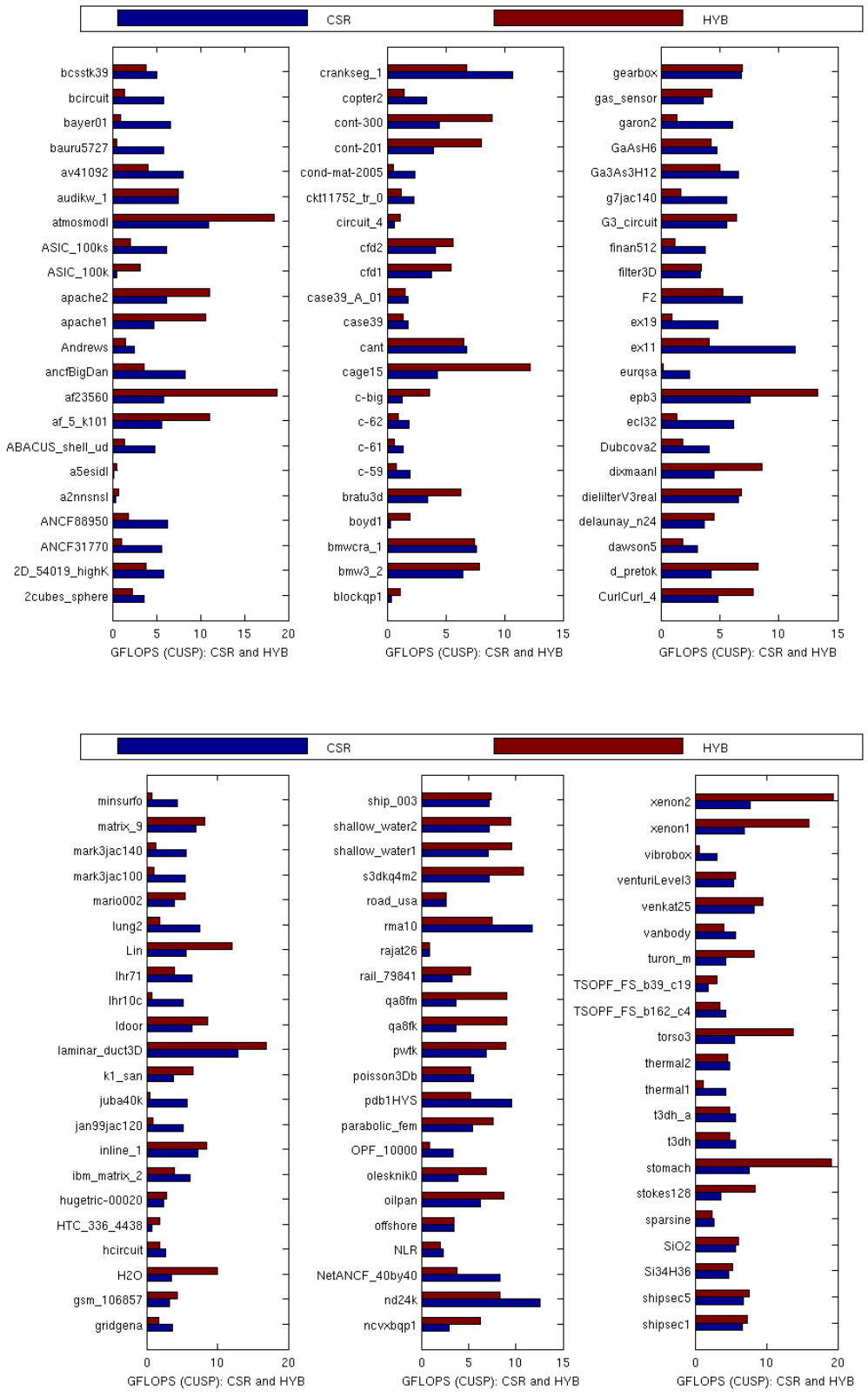

Figure 13: SPMV GFLOPS of matrices in HYB and CSR formats using `CUSP` implementation.

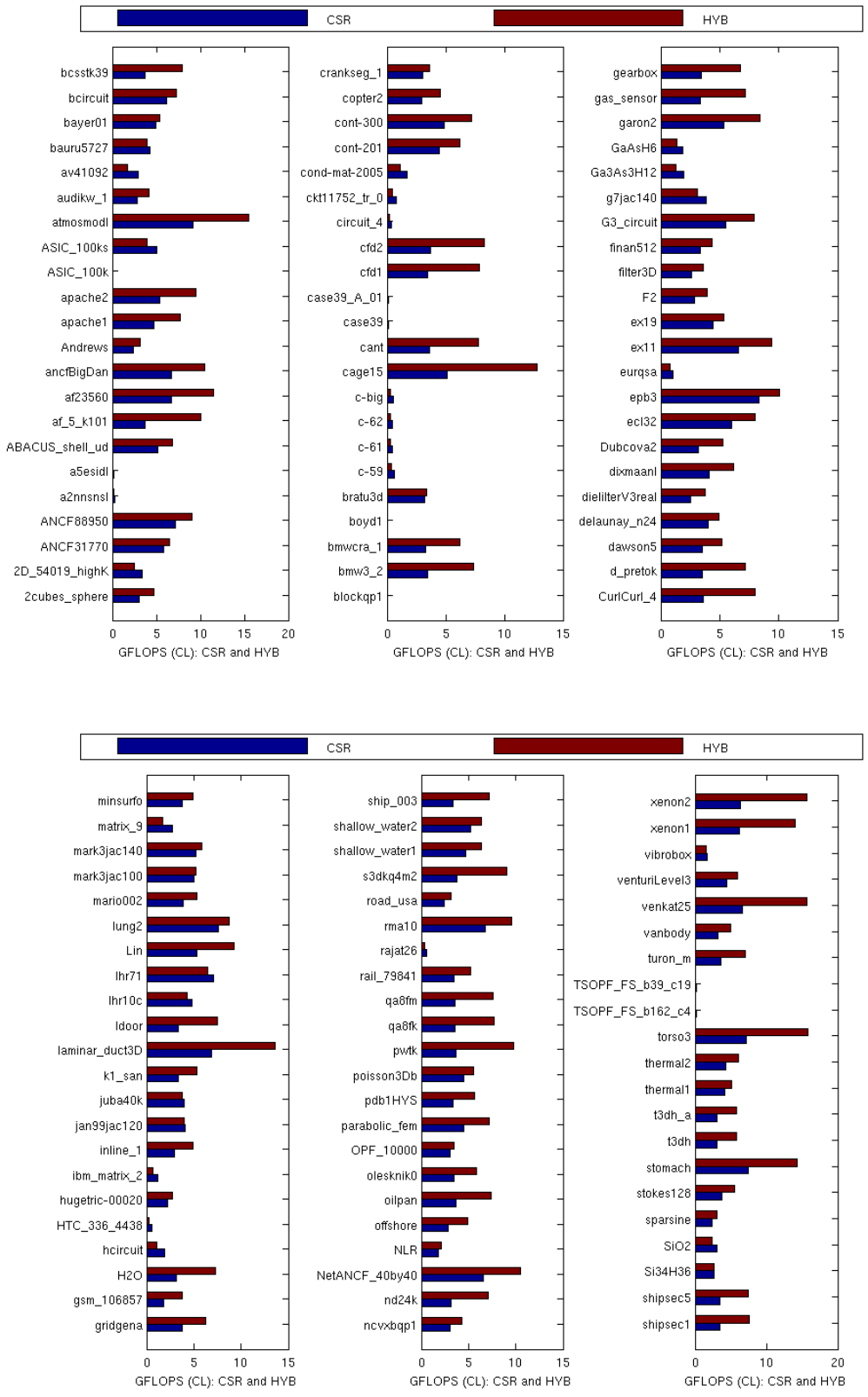

Figure 14: SPMV GFLOPS of matrices in HYB and CSR formats using `Vienna CL` implementation.