Performance of Sparse Matrix-Multiple Vectors Multiplication on Multicore and GPUs

Walid Abu-Sufah, University of Illinois at Urbana-Champaign; Khalid Ahmad , University of Jordan

Introduction

We implemented a heuristics-based auto-tuning framework for sparse matrix-vector multiplication (SpMV) on GPUs. For a given sparse matrix, our auto-tuner delivers the highest performing SpMV kernel which combines the use of the most efficient storage format and tuned parameters of the corresponding CUDA code targeting the underlying GPU architecture.

Different storage schemes/kernels perform best for matrices with different sparsity patterns. So far, our auto-tuner considers the storage formats: Diagonal (DIA), ELLPACK (ELL), CSR (vector), Coordinate (COO), Hybrid (HYB), Blocked ELLPACK, and our Blocked Transpose Jagged Diagonal Storage format (BTJAD). Other formats are currently being considered including SELL-C-σ and SEL-P.

When nonzero values are restricted to a small number of diagonals, DIA-SpMV is the best performing kernel. For matrices with uniform row lengths, ELL-SpMV is the best performing kernel. Currently, we are integrating sparse matrix-multiple vector multiplication (SpMM) kernels in our auto-tuner. So far we have implemented DIA-SpMM for structured matrices and ELL-SpMM for uniform row length matrices.

The Platforms and Matrices

We executed kernels on nodes of two HPC clusters: (1) one node of NVIDIA benchmarking PSG cluster using a Kepler K40m GPU with a dual socket 10-core Intel Ivy Bridge E5-2690 v2 @ 3.00GHz CPU, and (2) one node of the Cyprus Institute Cy-Tera cluster using a Fermi M2070 GPU with a dual socket Intel Westmere 6-core X5650 @ 2.66GHz CPU.

We used 5 structured matrices that represent common stencil operations on regular grids (Table 1). They are formed by applying a Laplace stencil operator to every point in an N-dimensional space. This results in nonzero elements that are restricted to a small number of diagonals, where the number of diagonals is the number of points in the stencil. From the University of Florida collection we used another 28 structured matrices (Table 2) and 29 matrices with uniform row lengths (Table 3). Performance is evaluated in terms of GFLOP/s, which is computed by dividing the number of arithmetic operations by the average running time of 1000 runs. Our time measurements do not include the time required to transfer data between the host and the GPU. Our experiments test our SpMM implementations for multiplying a sparse matrix by up to 512 vectors.

TABLE I
LAPLACIAN OPERATORS DISCRETIZED AS K-POINT FINITE DIFFERENCE STENCILS ON REGULAR GRIDS

Matrix	Grid	Diagonals	Nonzero Elements
Laplace 3pt	1,000,000	3	2,999,997
Laplace 5pt	$(1,000)^2$	5	4,994,001
Laplace 7pt	$(100)^3$	7	6,626,349
Laplace 9pt	$(1,000)^2$	9	8,986,005
Laplace 27pt	$(100)^3$	27	26,178,647

TABLE II
28 UNIVERSITY OF FLORIDA STRUCTURED MATRICES

Matrix	Nonzeros	Diagonals
Maximum	10,319,760	99
Average	1,537,409	16.6
Minimum	19,996	5

TABLE III
29 UNIVERSITY OF FLORIDA UNIFORM ROW LENGTH MATRICES

Matrix	Nonzeros	Average nonzeros per row	Maximum nonzeros per row	Standard deviation
Maximum	7,791,168	72.1	81	19.1
Average	1,339,107	10.3	10. 9	1.1
Minimum	20,360	1	1	0

In this poster we show K40m and Intel CPUs results. The improvements achieved by our kernels executing on the Fermi M2070 are close to the improvements achieved on the K40m.

Our kernels have superior performance because; (i) we use the most efficient storage schemes for the test matrices, (ii) we multiply each matrix by a block of vectors instead of one, and (iii) our code makes the most of GPU registers to exploit data reuse in SpMM. The design skeletons of our kernels are described elsewhere.

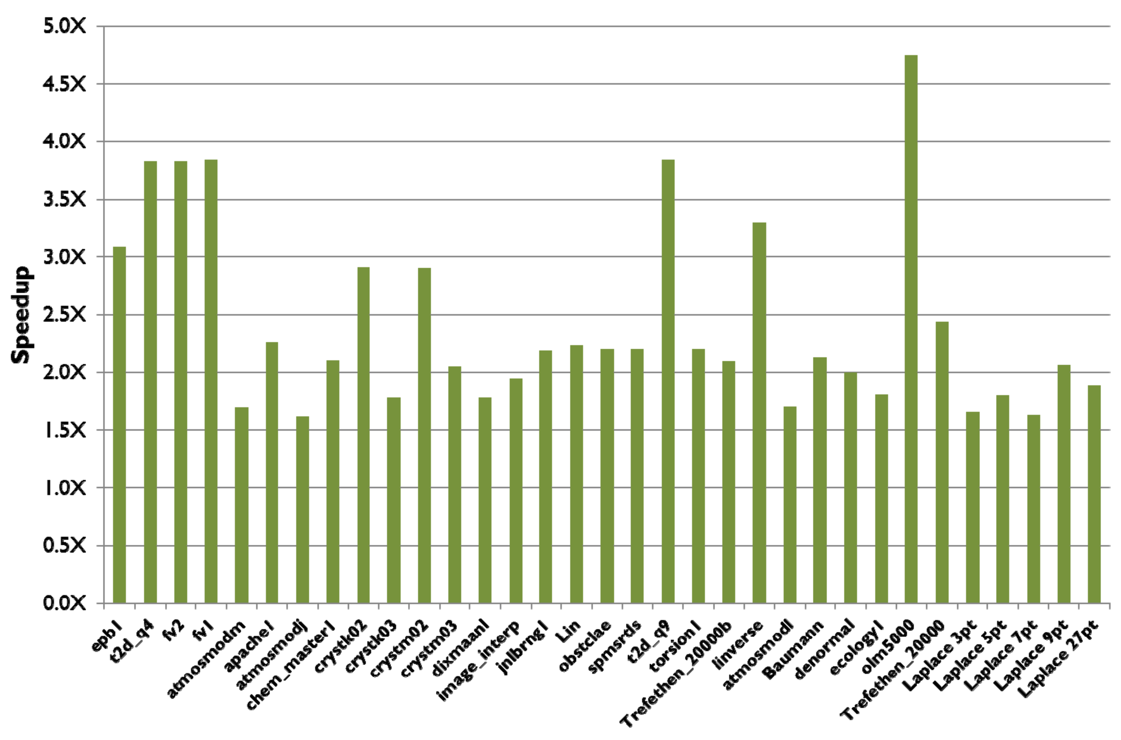
Acknowledgement

We thank Ahmed Sameh of Purdue University for his comments. We thank NVIDIA Corporation for providing the PSG benchmarking cluster. This work was supported in part by the Scientific Research Support Fund of Jordan grant number ICT/2/02/2012 and the LinkSCEEM-2 project, funded by the European Commission under the 7th Framework Programme through Capacities Research Infrastructure, INFRA-2010-1.2.3 Virtual Research Communities, Combination of Collaborative Project and Coordination and Support Actions (CP-CSA) under grant agreement no RI-2616000. This work also used the Stampede cluster of the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

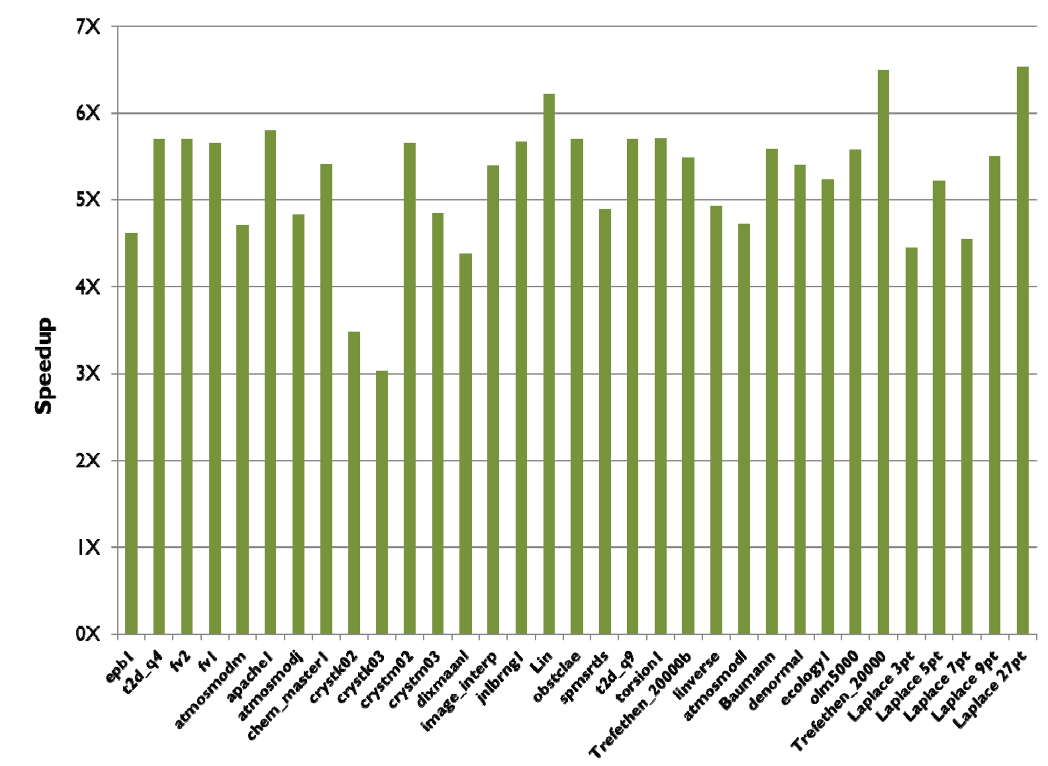
K40m GPU Results

Structured Matrices

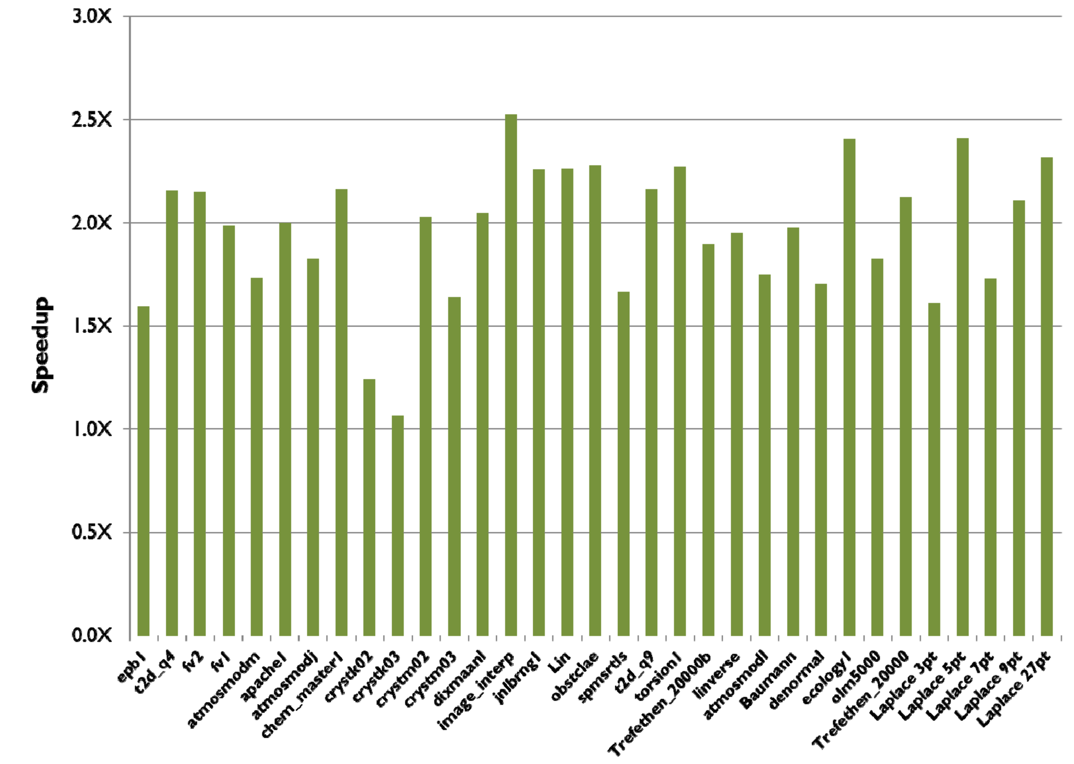
Our DIA SpMM: 2.4x Faster than CUSP DIA-SpMV



Our DIA-SpMM: 5.2x Faster than cuSPARSE CSR-SpMV

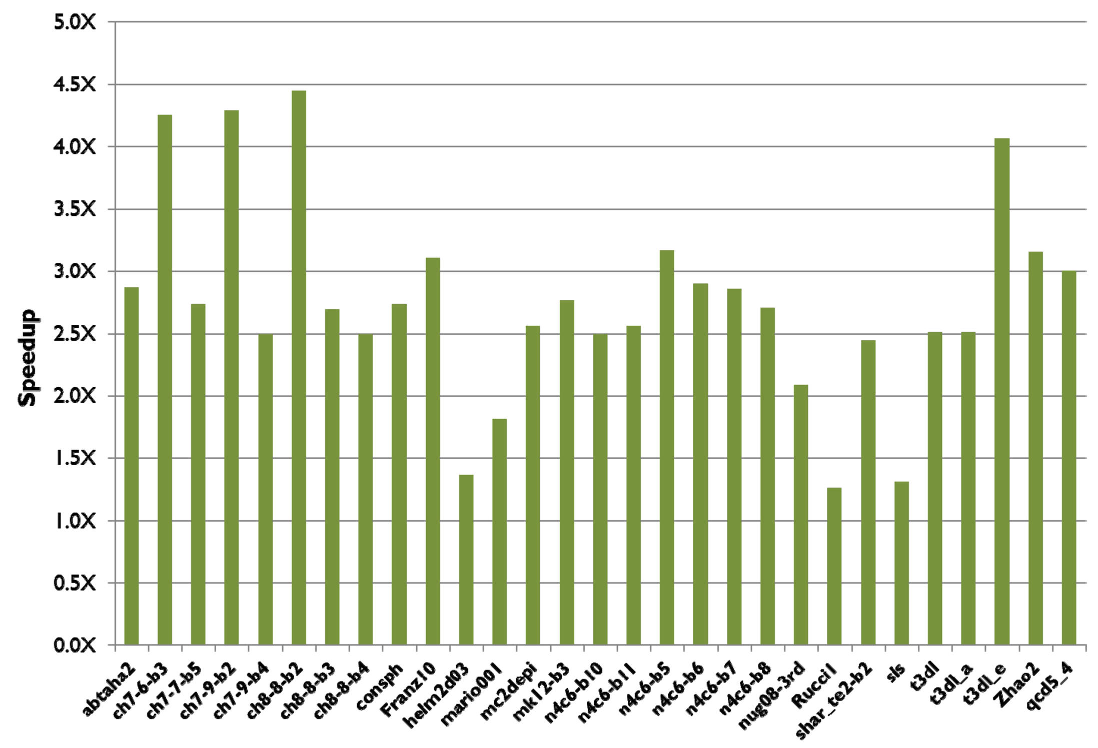


Our DIA SpMM: 2x Faster than cuSPARSE CSR-SpMM

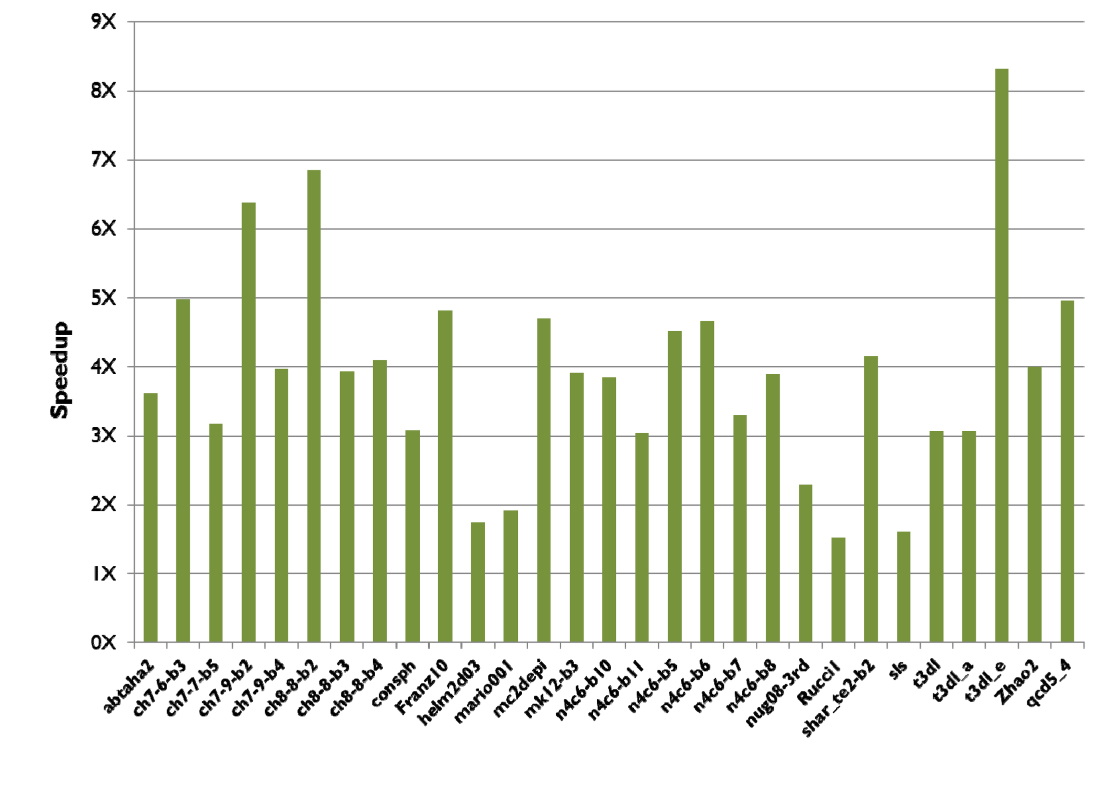


Uniform Row Length Matrices

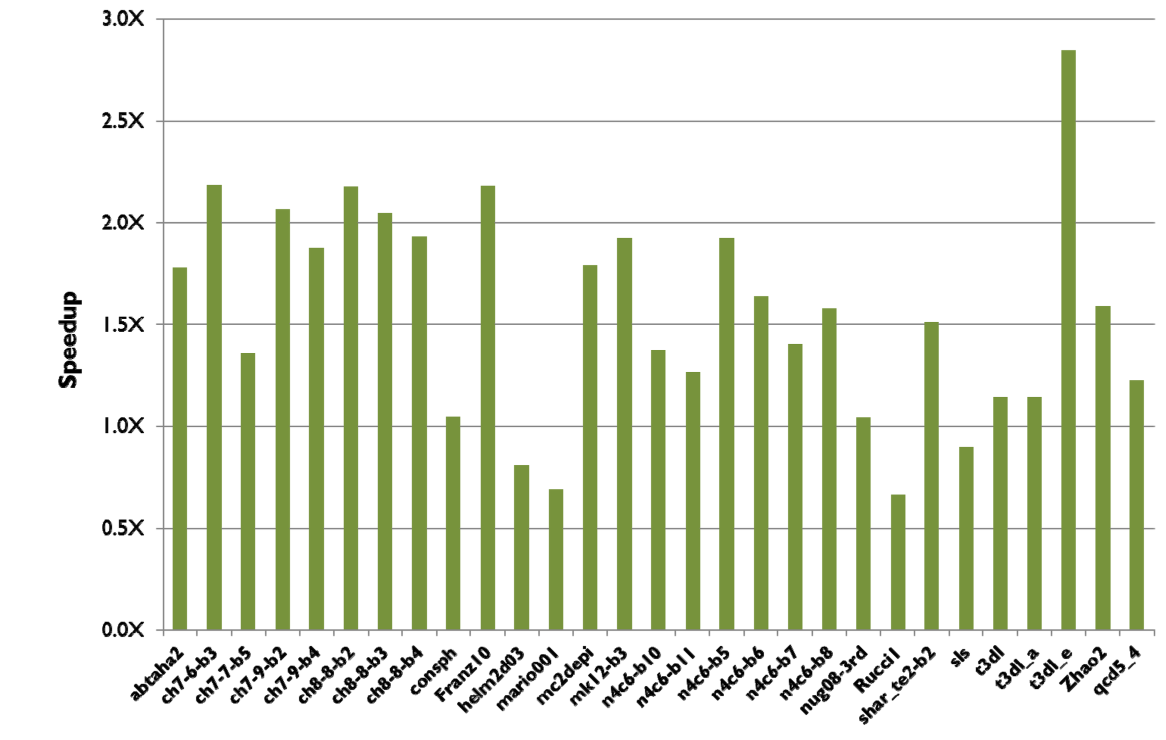
Our ELL-SpMM: 2.8x Faster than CUSP ELL-SpMV



Our ELL-SpMM: 3.9x faster than cuSPARSE CSR-SpMV



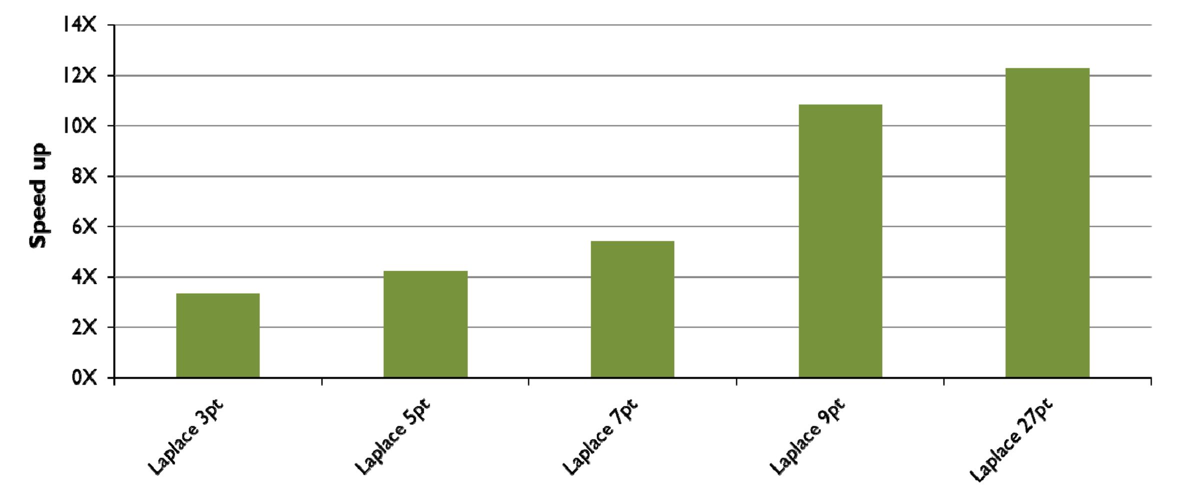
Our ELL SpMM: 1.6x faster than cuSPARSE CSR-SpMM



K40m GPU & Intel CPUs Results

Structured Laplace Matrices

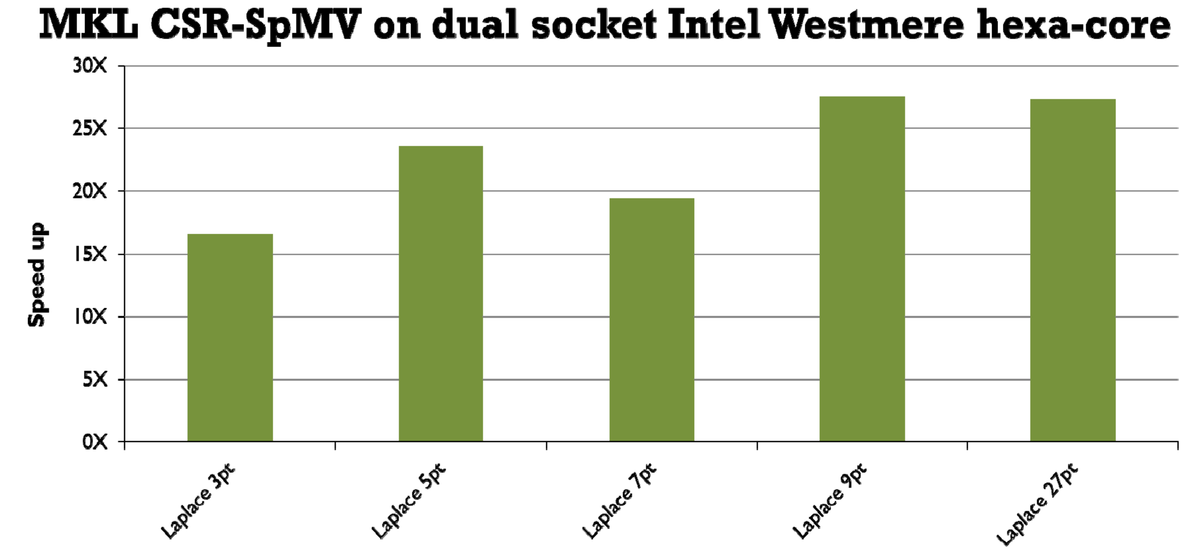
Our DIA-SpMM: 7.2x faster than the best performing Intel MKL CSR-SpMV on dual socket Intel Ivy Bridge 10-core



Matrix	Number of threads producing the best MKL kernel performance
Laplace 3pt	19
Laplace 5pt	19
Laplace 7pt	15
Laplace 9pt	15
Laplace 27pt	9

- MKL 11.0 on dual socket Intel Ivy Bridge 10-core E5-2690 v2 @ 3.00GHz CPU
- Structured grid sparse matrices; applying a Laplace stencil operator to every point in an N-dimensional space; https://sites.google.com/site/sparseautotuner/matrices/laplace

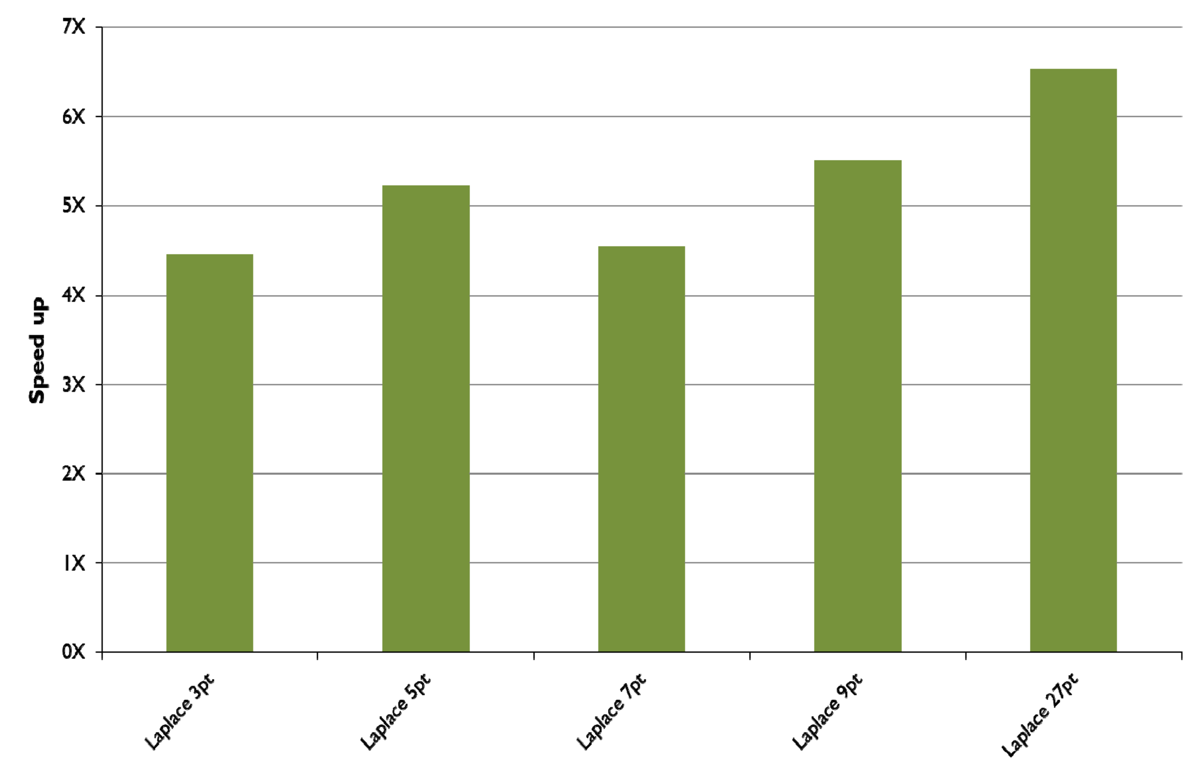
Our DIA-SpMM: 22.8x faster than the best performing Intel MKL CSR-SpMV on dual socket Intel Westmere hexa-core



Matrix	Number of threads producing the best MKL kernel performance
Laplace 3pt	11
Laplace 5pt	8
Laplace 7pt	11
Laplace 9pt	11
Laplace 27pt	9

- MKL 11.0.1.117 on two Intel Westmere hexa-core X5650 @ 2.66GHz
- Structured grid sparse matrices; applying a Laplace stencil operator to every point in an N-dimensional space: https://sites.google.com/site/sparseautotuner/matrices/laplace

Our DIA-SpMM: 5.25x faster than NVIDIA cuSPARSE CSR-SpMV



- Results of single precision routines
- Our DIA-SpMM, ELL-SpMM, NVIDIA cuSPARSE CSR-SpMM, CUSP DIA-SpMV, and ELL-SpMV on NVIDIA K40m GPU, ECC ON, input and output data on device
- Structured matrices obtained from: https://sites.google.com/site/sparseautotuner/matrices/laplace http://www.cise.ufl.edu/research/sparse/matrices/

Performance may vary based on OS version and motherboard configuration