Exploiting Matrix Reuse and Data Locality in Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication on Many-Core Architectures

M. Ozan Karsavuran (1) Kadir Akbudak (2) Cevdet Aykanat (1)

(1) Bilkent University (2) KAUST

Abstract

We focus on the efficient thread-level parallelization of sparse matrix-vector and matrix-transpose-vector multiplication (SpMM$^{\mathrm{T}}$V) operations of the form $y = AA^T x$, which is performed as two successive SpMV operations $z \leftarrow A^T x$ and $y \leftarrow A\ z$. There are several iterative methods that involve repeated and consecutive SpMV and SpM$^{\mathrm{T}}$V operations for the same sparse matrix A.

The SpMV operation is one of the latency bound kernels [1, 2, 3] due to its irregular accesses and low flop-to-byte ratios (amount of arithmetic operations per word retrieved from RAM). Decreasing the amount of accesses to RAM via exploiting locality is expected to improve performance of the SpMM$^{\mathrm{T}}$V operation especially on cache-coherent processors involving large number of cores, such as Intel Xeon Phi.

Based on one-dimensional (1D) rowwise and columnwise partitioning of A and A^T matrices, four parallel SpMM$^{\mathrm{T}}$V approaches are viable: column-column parallel (CCp), row-row parallel (RRp), column-row parallel (CRp), and row-column parallel (RCp). Here, CCp is based on column-parallel SpMV in both $y \leftarrow A\ z$ and $z \leftarrow A^T x$, where RRp is based on row-parallel SpMV in both $y \leftarrow A\ z$ and $z \leftarrow A^T x$. CRp is based on column-parallel SpMV in $y \leftarrow A\ z$ and row-parallel SpMV in $z \leftarrow A^T x$, where RCp is based on row-parallel SpMV in $y \leftarrow A\ z$ and column-parallel SpMV in $z \leftarrow A^T x$. Fig. 1 shows the execution of these four parallel SpMM$^{\mathrm{T}}$V algorithms by a four-thread system. A gray scale tone shows exclusive access by a single thread and black color shows concurrent accesses by multiple threads.

We identify the following five quality criteria which have impact on the performance of the above mentioned thread-level parallelization schemes:

(a) Reusing z-vector entries, which are written in $z \leftarrow A^T x$ and then loaded in $y \leftarrow A\ z$.

(b) Reusing matrix nonzeros (together with their index structures) among consecutive $z \leftarrow A^T x$ and $y \leftarrow A\ z$ operations.

(c) Exploiting temporal locality in loading input vector entries in row-parallel SpMV operations.

(d) Exploiting temporal locality in updating output vector entries in column-parallel SpMV operations.

(e) Minimizing number of concurrent writes by different threads in column-parallel SpMV operations.

Table 1: Quality criteria coverage [4].

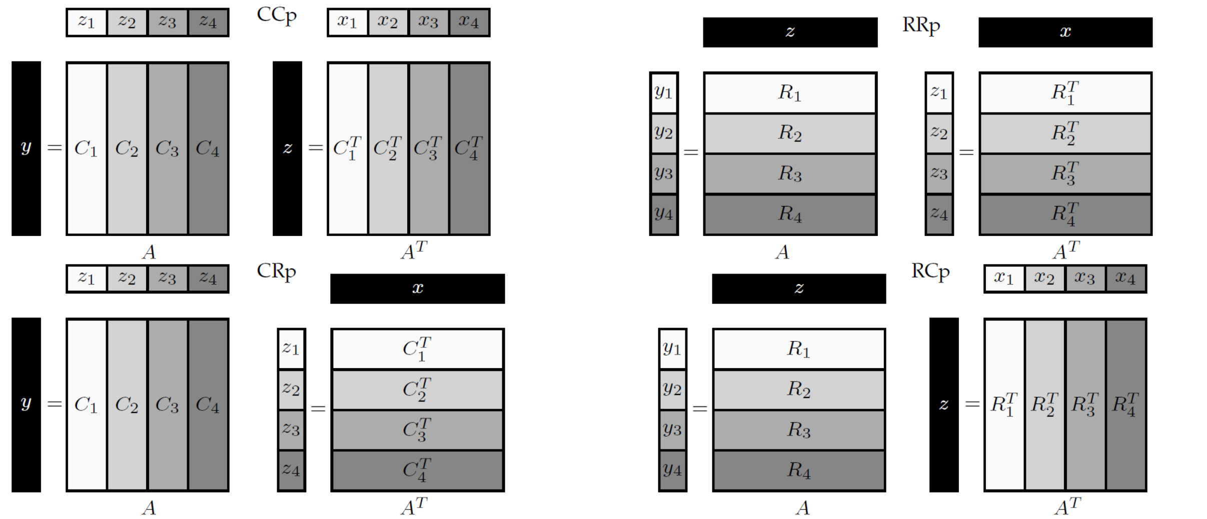
Quality Criteria	RRp	CRp	RCp	*sb*CRp	*sb*RCp
(a) z-vector reuse	×	✓	×	✓	$\checkmark\!\!\times^{1}$
(b) A-matrix nonzero reuse	×	✓	×	✓	$\checkmark\!\!\times^{2}$
(c) temporal locality in row-parallel SpMV	$\times^3$	$\times^3$	$\times^3$	✓	✓
(d) temporal locality in col-parallel SpMV	−	$\times^3$	$\times^3$	✓	✓
(e) minimizing concurrent writes	✓	×	×	✓	✓

✓: satisfied $\checkmark\!\!\times^{1}$: satisfied except z_B border subvectors
−: not applicable $\checkmark\!\!\times^{2}$: satisfied except A_{kB} border submatrices
×: not satisfied $\times^3$: may be satisfied through row/column reordering

The first four criteria consider data reuse. The criterion (b) for achieving A-matrix nonzero reuse in 1D parallelization incurs concurrent writes. Hence, the last criterion is introduced to refer to the trade-off between the concurrent writes and A-matrix nonzero reuse. Here, we only consider CRp and RCp since CCp and RRp are not amenable to achieve criteria (a) and (b)

In order to satisfy all five quality criteria at the same time for CRp and RCp, we propose permuting A and A^T matrices into dual singly-bordered block-diagonal (SB) forms [4]. For CRp, we permute matrix A into a rowwise SB form, which induces a columnwise SB form of matrix A^T. For RCp, we permute matrix A into a columnwise SB form, which induces a rowwise SB form of matrix A^T. Fig. 2, shows the proposed SB-based parallel SpMM$^{\mathrm{T}}$V algorithms which are referred to here as sbCRp and sbRCp. We show that these two dual SB forms enable to achieve the quality criteria (a) and (b) in CRp and RCp. We also show that the objectives of minimizing the size of the row border and column border in the SB form of A correspond to achieve the quality criteria (c), (d), and (e) in CRp and RCp, respectively. Table 1 shows quality criteria coverage of the algorithms.

The validity of the algorithms are evaluated on a single Xeon Phi processor for 28 sparse matrices arising from a wide range of applications. Fig. 3 shows the performance profile that compares the proposed SB-based algorithms against two baseline algorithms in terms of running time. In spite of the advantages of sbCRp over sbRCp as shown in Table 1, sbCRp may perform worse than sbRCp for some matrices. So in Fig 3, CRp/RCp-SB refers to considering the minimum running time of sbCRp and sbRCp for each matrix.


CCp
z_1 z_2 z_3 z_4
x_1 x_2 x_3 x_4
y = C_1 C_2 C_3 C_4
z = C_1^T C_2^T C_3^T C_4^T
A
A^T
RRp
z
x
y_1 y_2 y_3 y_4 = R_1 R_2 R_3 R_4
z_1 z_2 z_3 z_4 = R_1^T R_2^T R_3^T R_4^T
A
A^T
CRp
z_1 z_2 z_3 z_4
x
y = C_1 C_2 C_3 C_4
z_1 z_2 z_3 z_4 = C_1^T C_2^T C_3^T C_4^T
A
A^T
RCp
z
x_1 x_2 x_3 x_4
y_1 y_2 y_3 y_4 = R_1 R_2 R_3 R_4
z = R_1^T R_2^T R_3^T R_4^T
A
A^T

A gray scale tone denotes exclusive accesses by a single thread, whereas black color denotes concurrent accesses by multiple threads.

Figure 1: Four baseline SpMM$^{\mathrm{T}}$V algorithms for computing $y \leftarrow A\ z$ after $z \leftarrow A^T x$ by four threads [4].

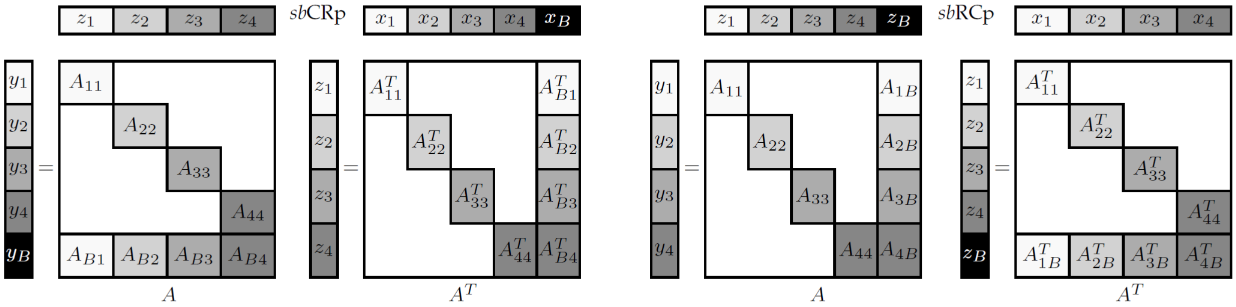

sbCRp
z_1 z_2 z_3 z_4
x_1 x_2 x_3 x_4 x_B
y_1 y_2 y_3 y_4 y_B =
A_{11} A_{22} A_{33} A_{44}
A_{B1} A_{B2} A_{B3} A_{B4}
z_1 z_2 z_3 z_4 =
A_{11}^T A_{22}^T A_{33}^T A_{44}^T
A_{B1}^T A_{B2}^T A_{B3}^T A_{B4}^T
A
A^T
sbRCp
z_1 z_2 z_3 z_4 z_B
x_1 x_2 x_3 x_4
y_1 y_2 y_3 y_4 =
A_{11} A_{22} A_{33} A_{44}
A_{1B} A_{2B} A_{3B} A_{4B}
z_1 z_2 z_3 z_4 z_B =
A_{11}^T A_{22}^T A_{33}^T A_{44}^T
A_{1B}^T A_{2B}^T A_{3B}^T A_{4B}^T
A
A^T

Figure 2: Proposed SB-based SpMM$^{\mathrm{T}}$V algorithms for computing $y \leftarrow A\ z$ after $z \leftarrow A^T x$ by four threads [4].

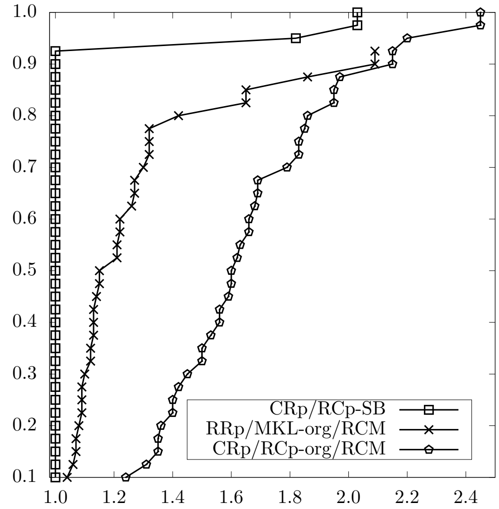


Figure 3: Performance profile curves.

We adopt a similar "best-of" approach for both baseline algorithms. RRp/MKL-org/RCM refers to considering the minimum running time of RRp and MKL for original and RCM ordering. CRp/RCp-org/RCM refers to considering the minimum running time of CRp and RCp for original and RCM ordering. As seen in Fig 3, the proposed SB-based algorithms performs significantly better than the baseline algorithms. As also seen in the figure, the proposed SB-based algorithms achieve the best performance in 92% of the SpMM$^{\mathrm{T}}$V instances. On the average, the proposed methods runs 28% faster than the best baseline algorithm.

References

[1] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Com.ACM*, pp. 65–76, 2009.

[2] E. Saule, K. Kaya, and U. V. Catalyürek, "Performance evaluation of sparse matrix multiplication kernels on Intel Xeon Phi," in *Parallel Processing and Applied Mathematics*, LNCS, pp.559–570, 2014.

[3] K. Akbudak, E. Kayaaslan, and C. Aykanat, "Hypergraph partitioning based models and methods for exploiting cache locality in sparse matrix-vector multiplication," *SIAM SISC*, pp. C237–C262, 2013.

[4] M. O. Karsavuran, K. Akbudak, and C. Aykanat, "Locality-aware parallel sparse matrix-vector and matrix-transpose-vector multiplication on many-core processors," *IEEE TPDS*, 2016.