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

Ozan Karsavuran[1]

Kadir Akbudak (speaker)[2]
sites.google.com/site/kadircs
kadir.cs@gmail.com

Cevdet Aykanat[1]

[1]Bilkent University, Turkey

[2]KAUST, KSA

SIAM Workshop on Combinatorial Scientific Computing (CSC), Albuquerque, NM, USA, October 10-12, 2016

O. Karsavuran, **K. Akbudak**, and C. Aykanat, *Locality-Aware Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication on Many-Core Architectures*, **IEEE Transactions on Parallel and Distributed Systems (TPDS)**, vol. 27(6), pp. 1713-1726, 2016, available at ieeexplore.ieee.org/document/7152923/

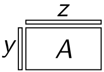
1. Introduction: $y = AA^T x$
2. Open problems & Related work
3. Parallel $\mathrm{SpAA^T}$ based on 1D partitioning of A and A^T matrices
   - Quality criteria for efficient parallelization of $\mathrm{SpAA^T}$
   - Proposed $\mathrm{SpAA^T}$ algorithms
   - Experiments
4. References

Thread-level parallelization of $y = AA^T x$ ($\mathrm{SpAA^T}$)

- $y = AA^T x$ is computed as two Sparse Matrix-Vector Multiplies (SpMV)
  - $z = A^T x$ and then

Sparse Matrix-Transpose–Vector Multiply ($\mathrm{Sp}A^T$)

  - $y = Az$



Sparse Matrix-Vector Multiply ($\mathrm{Sp}A$)

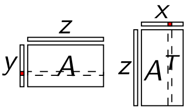
- Thread-level parallelization of repeated and consecutive $\mathrm{Sp}A$ and $\mathrm{Sp}A^T$ that involve the same sparse matrix A
- Examples:
  - Linear Programming (LP) problems via interior point methods
  - nonsymmetric systems via
    - Bi-CG, CGNE, Lanczos Bi-ortagonalization
  - least squares problem via LSQR
  - linear feasibility problem via Surrogate Constraints method
  - Krylov-based balancing algorithms used as preconditioners for sparse eigensolvers
  - web page ranking via HITS algorithm

Open problems

- Utilize the opportunity of reusing A-matrix nonzeros?
- Obtain close performance for both $z = A^T x$ and $y = Az$ at the same time?
  - Single storage of A for both $z = A^T x$ and $y = Az$
  - Storage of A^T for $z = A^T x$ and a separate storage of A for $y = Az$

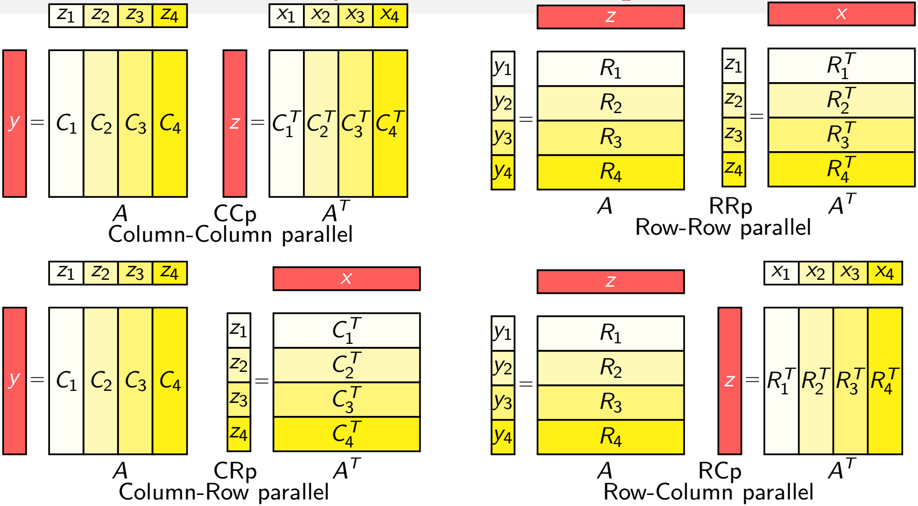
Related work

- Optimized Sparse Kernel Interface (OSKI), Berkeley
  - Serial
  - Each row/column is reused.



- Compressed Sparse Blocks (CSB) by Buluc et. al. [10]
  - Parallel
  - Same data structure for both $\mathrm{Sp}A$ and $\mathrm{Sp}A^T$ operations without any performance degradation
  - Two phase, i.e., $\mathrm{Sp}A$ and $\mathrm{Sp}A^T$ are not performed simultaneously

Thread-level baseline parallelization of $\mathrm{SpAA^T}$



YELLOW scale tone: exclusive accesses by a single thread

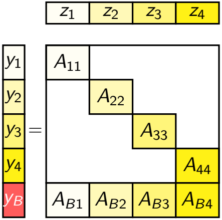
RED color: concurrent accesses by multiple threads.

Four baseline $\mathrm{SpAA^T}$ algorithms for computing $y = A\,z$ after $z = A^T x$ by four threads.

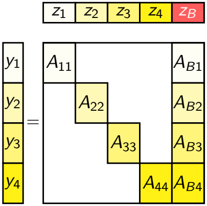
Contributions

- Identify five quality criteria (QC), which have impact on performance of parallel $\mathrm{SpAA^T}$
- Singly-bordered block-diagonal (SB) form based methods: *sb*CRp and *sb*RCp

Matrix A partitioned in to four and the submatrices are processed by four threads.



For *sb*CRp (SB-based Column-Row parallel algorithm), we permute matrix A into a rowwise SB form, which induces a columnwise SB form of matrix A^T

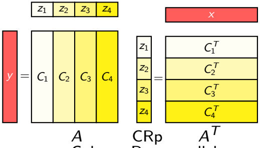


For *sb*RCp (SB-based Row-Column parallel algorithm), we permute matrix A into a columnwise SB form, which induces a rowwise SB form of matrix A^T

- Achieve (a) (z-vector reuse) and (b) (A-matrix reuse).
- Objectives of minimizing the size of the row/column border in the SB form of $A \approx$ achieve QC (c), (d), and (e) in *sb*CRp/*sb*RCp.

Quality criteria for efficient parallelization of $\mathrm{SpAA}^{\mathrm{T}}$

	Quality Criteria	RRp	CRp	RCp	*sb*CRp	*sb*RCp
(a)	Reusing z-vector entries generated in $z = A^T x$ and then read in $y = A\,z$	×	✓	×	✓	✓✗[1]
(b)	Reusing matrix nonzeros (together with their indices) in $z = A^T x$ and $y = A\,z$	×	✓	×	✓	✓✗[2]
(c)	Exploiting temporal locality in reading input vector entries in row-parallel SpMVs	×[3]	×[3]	×[3]	✓	✓
(d)	Exploiting temporal locality in updating output vector entries in column-parallel SpMVs	—	×[3]	×[3]	✓	✓
(e)	Minimizing the number of concurrent writes performed by different threads in column-parallel SpMVs	✓	×	×	✓	✓



Column-Row parallel

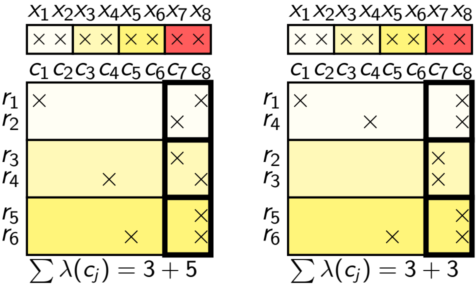
✓: satisfied
—: not applicable
×: not satisfied

✓✗[1]: satisfied except z_B border subvectors
✓✗[2]: satisfied except A_{kB} border submatrices
×[3]: may be satisfied through row/column reordering

- Maintaining balance on the number of nonzeros at each slice
  - Reducing parallel time under arbitrary task scheduling
- Reducing border size

Reducing # of cache misses due to loss of temporal locality

$$\lambda(c_j) = |\{A_k : c_j \text{ has at least one nonzero at } A_k, \forall k \in 1, \ldots, K\}|$$

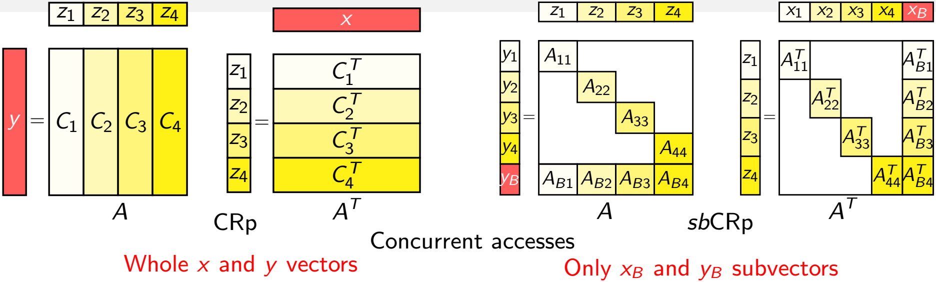


Matrix A partitioned in to three and the submatrices are processed by three threads.

Reducing # of concurrent writes

$$\lambda(r_i) = |\{A_k : r_i \text{ has at least one nonzero at } A_k, \forall k \in 1, \ldots, K\}|$$

Merits of Singly-Bordered Block Diagonal (SB) Form on CRp



- Exploits temporal locality in reading x-vector entries in row parallel $z = A^T x$
- Exploits temporal locality in updating y-vector entries in column-parallel $y = A\,z$

Minimizing border size in the SB form → Minimizing number of concurrent writes by different threads in column-parallel $y = A\,z$

Require: A_{kk} and A_{Bk} matrices; x, y, and z vectors

for $k \leftarrow 1$ **to** K **in parallel do**
- $z_k \leftarrow A_{kk}^T\, x_k$
- $z_k \leftarrow z_k + A_{Bk}^T\, x_B$

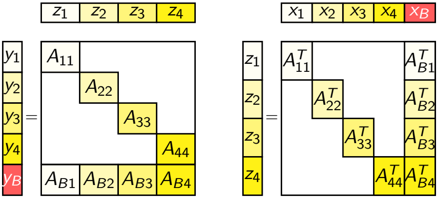
$\Big\}\ z_k \leftarrow C_k^T\, x$

- $y_k \leftarrow A_{kk}\, z_k$
- $y_B \leftarrow y_B + A_{Bk}\, z_k$ ▷ Concurrent writes

$\Big\}\ y \leftarrow C_k\, z_k$

end for

Singly-bordered block-diagonal (SB) form



A *sb*CRp A^T

SB-based Column-Row parallel

Require: A_{kk} and A_{kB} matrices; x, y, and z vectors

for $k \leftarrow 1$ **to** K **in parallel do**
- $z_k \leftarrow A_{kk}^T\, x_k$
- $z_B \leftarrow z_B + A_{kB}^T\, x_k$ ▷ Concurrent writes

$\Big\}\ z \leftarrow R_k^T\, x_k$

- $y_k \leftarrow A_{kk}\, z_k$

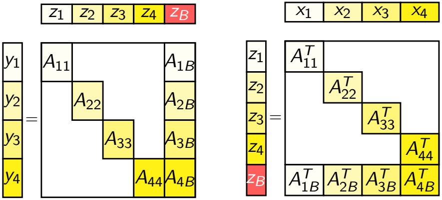
end for

for $k \leftarrow 1$ **to** K **in parallel do**
- $y_k \leftarrow y_k + A_{kB}\, z_B$

end for

$\Big\}\ y_k \leftarrow R_k\, z$

Singly-bordered block-diagonal (SB) form



A *sb*RCp A^T

SB-based Row-Column parallel

Iterative Methods		CRp	*sb*CRp	*sb*RCp
Directly applicable				
LP [1, 2]	$z \leftarrow A^T x$ $y \leftarrow Az$	✓	✓	✓
Directly (no dependency since inner product can be delayed)				
CGNE [3]	$z \leftarrow q - Ax$ $\beta \leftarrow (z, z)/(q, q)$ $y \leftarrow A^T z$	✓	✓	✓
Directly (linear vector operations without synchronization)				
LSQR [4]	$z \leftarrow Ax$ $w \leftarrow f(z)$ $y \leftarrow A^T w$	✓	✓	✓
Surrogate Constraints [5, 6]	$z \leftarrow Ax$ $w \leftarrow f(z)$ $y \leftarrow A^T w$	✓	✓	✓
Independent SpMVs (the two for loops of *sb*RCp can be fused.)				
BiCG [3]	$z \leftarrow Ax$ $y \leftarrow A^T w$	✓	✓	✓
Lanczos Bi-orthogonalization [3]	$z \leftarrow Ax$ $y \leftarrow A^T w$	✓	✓	✓
HITS [7, 8]	$z \leftarrow Ax$ $y \leftarrow A^T w$	✓	✓	✓
Krylov-based Balancing [9]	$z \leftarrow Ax$ $y \leftarrow A^T x$	✓	✓	✓
Not applicable due to inner product and inter-dependency				
CGNR [3]	$z \leftarrow Ax$ $\alpha \leftarrow \|\|y\|\|_2^2/\|\|z\|\|_2^2$ $y \leftarrow A^T\ \alpha w$	×	×	×

Performance Results on Intel Xeon Phi

- Average results of 28 sparse matrices from UFL
  - Up to 20M nonzeros, 3.5M rows/cols
- Baseline methods
  - RRp, CRp, RCp (OpenMP)
  - RRp with vendor-provided MKL
  - Reverse Cuthill-McKee for QC (c) and (d)
- Proposed methods
  - *sb*CRp, *sb*RCp (OpenMP, PaToH-3runs)
- Highly-tuned SpMV libs can be integrated.
- Normalized wrt RRp with original ordering

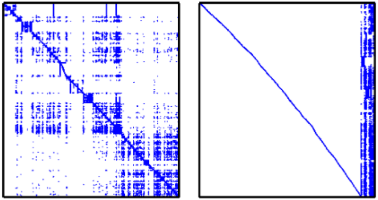
Normalized parallel $\mathrm{SpAA^T}$ times

RRp		MKL		Best of CRp/RCp		
org	RCM	org	RCM	org	RCM	SB
1.00	0.76	1.42	1.16	1.16	0.96	0.58

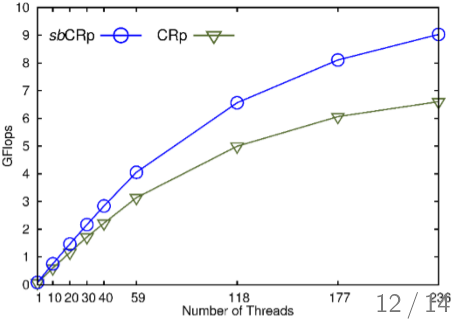
*Smaller the better
**Best of 1, 2, 3, and 4 threads per core

web-BerkStan

sbCRp

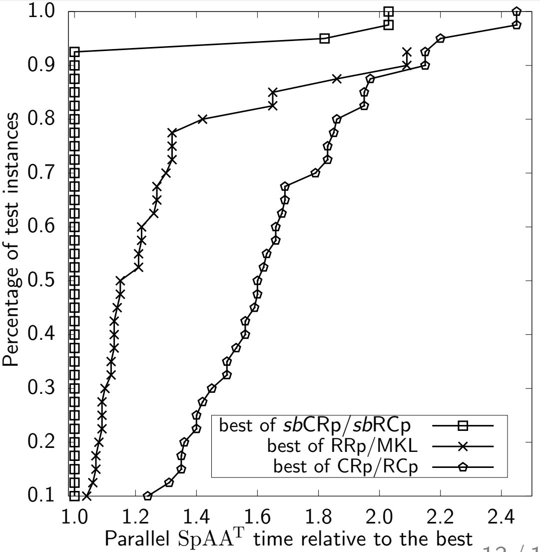


Original | Columnwise SB form



Performance Profiles

- Proposed methods: *sb*CRp, *sb*RCp
- Double storage of A:
  - RRp, MKL
  - Original order, RCM ordering
- Single storage of A:
  - CRp, RCp
  - Original order, RCM ordering



Performance Results on Xeon

- Two E5-2643 processors @3.30GHz
- 8 cores in total
- 16 threads with HyperThreading

Normalized parallel $\mathrm{SpAA^T}$ times

Matrix	RRp org	RRp RCM	MKL org	MKL RCM	Best of CRp/RCp org	Best of CRp/RCp RCM	Best of CRp/RCp SB
degme	1.00	1.21	1.22	1.11	0.69	0.97	0.58
LargeRegFile	1.00	1.02	1.53	1.38	0.75	1.12	0.45
Stanford	1.00	0.48	0.77	0.57	3.09	0.40	0.31
web-BerkStan	1.00	0.93	1.29	1.82	1.70	1.88	0.91

*Smaller the better

Preprocessing overhead in terms of number of $\mathrm{SpAA^T}$ operations using RRp

Matrix	*sb*CRp/*sb*RCp
degme	136
LargeRegFile	143
Stanford	2
web-BerkStan	12

References:

[1] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Proc. 16th annual ACM symposium on Theory of computing*, pp. 302–311, 1984.

[2] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 575–601, 1992.

[3] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[4] C. C. Paige and M. A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least squares," *ACM Transactions on Mathematical Software (TOMS)*, vol. 8, no. 1, pp. 43–71, 1982.

[5] K. Yang and K. G. Murty, "New iterative methods for linear inequalities," *Journal of Optimization Theory and Applications*, vol. 72, no. 1, pp. 163–185, 1992.

[6] B. Uçar, C. Aykanat, M. Ç. Pınar, and T. Malas, "Parallel image restoration using surrogate constraint methods," *Journal of Parallel and Distributed Computing*, vol. 67, no. 2, pp. 186–204, 2007.

[7] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[8] X. Yang, S. Parthasarathy, and P. Sadayappan, "Fast sparse matrix-vector multiplication on GPUs: Implications for graph mining," *Proc. VLDB Endow.*, vol. 4, no. 4, pp. 231–242, Jan. 2011.

[9] T.-Y. Chen and J. W. Demmel, "Balancing sparse matrices for computing eigenvalues," *Linear Algebra and its Applications*, vol. 309, no. 13, pp. 261 – 287, 2000.

[10] A. Buluç, J. T. Fineman, M. Frigo, J. R. Gilbert, and C. E. Leiserson, "Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks," *Proc. 21st symposium on Parallelism in Algorithms and Architectures*, pp. 233–244, 2009.

[11] K. Akbudak, E. Kayaaslan, and C. Aykanat, "Hypergraph partitioning based models and methods for exploiting cache locality in sparse matrix-vector multiplication," *SIAM Journal on Scientific Computing*, vol. 35, no. 3, pp. C237–C262, 2013.