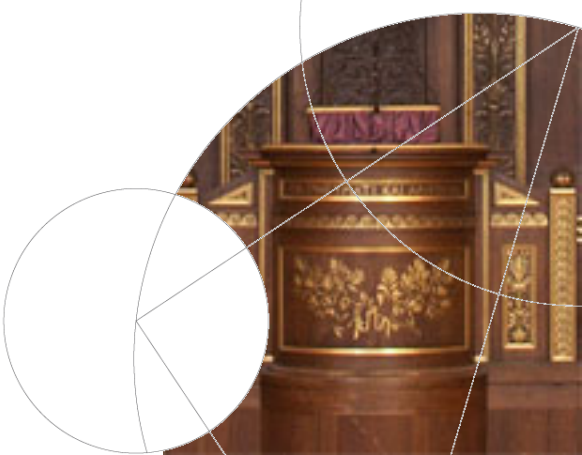
An Empirical Study of Sparse BLAS on Emerging Heterogeneous Processors

Weifeng Liu, Brian Vinter
Niels Bohr Institute,
University of Copenhagen, Denmark

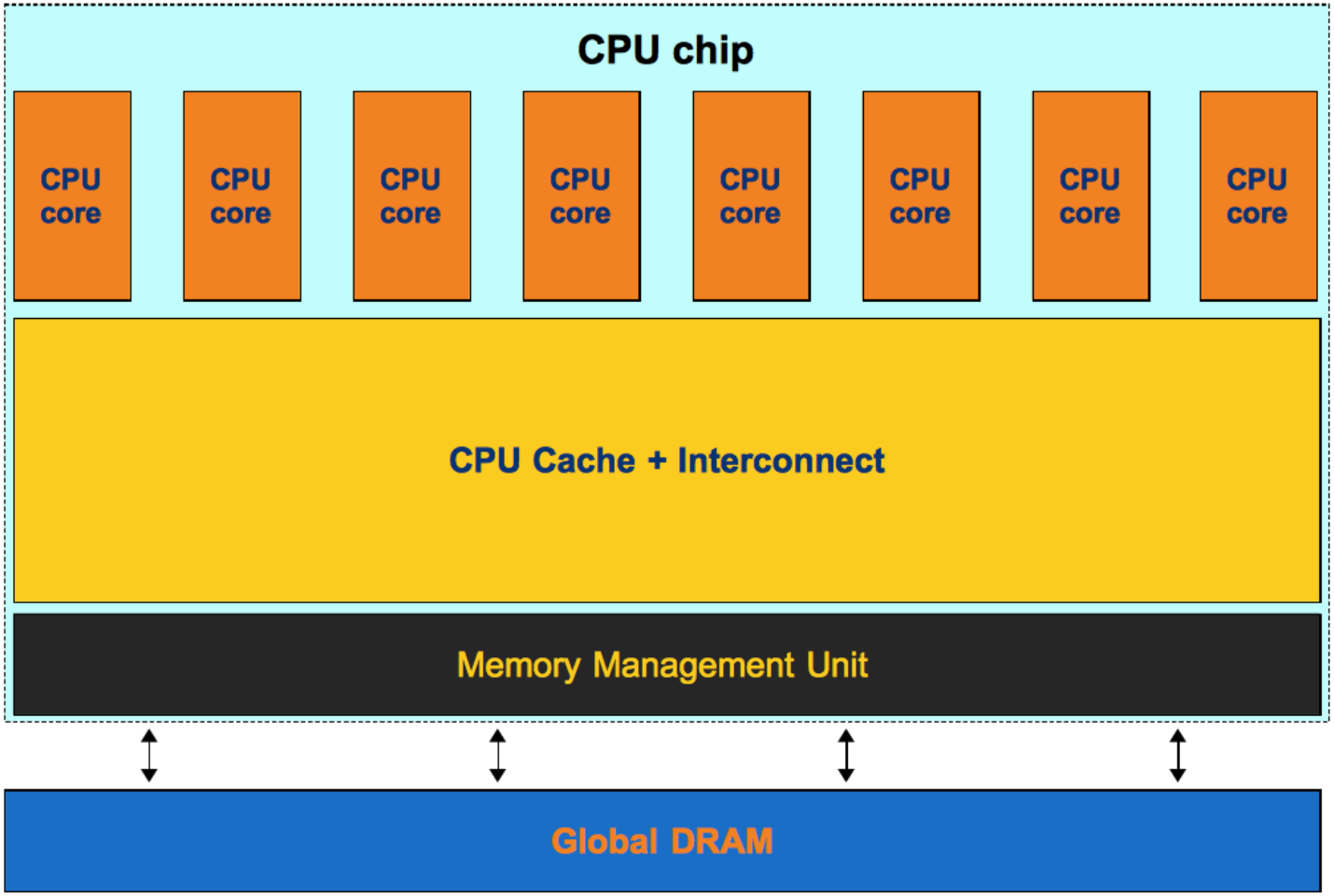


PMAA `16, Bordeaux, France, 8 July 2016

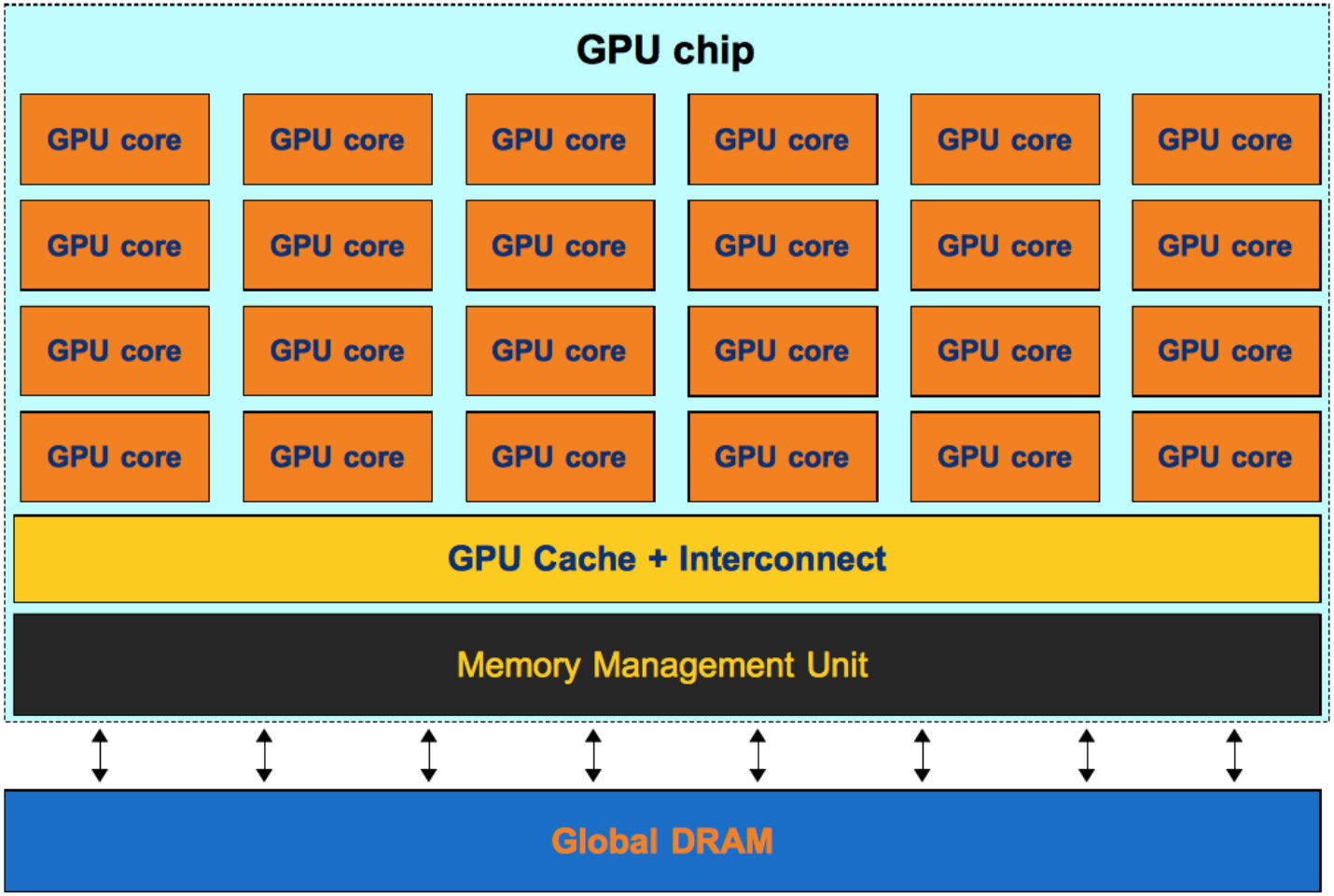
Background:

Emerging Heterogeneous Processors

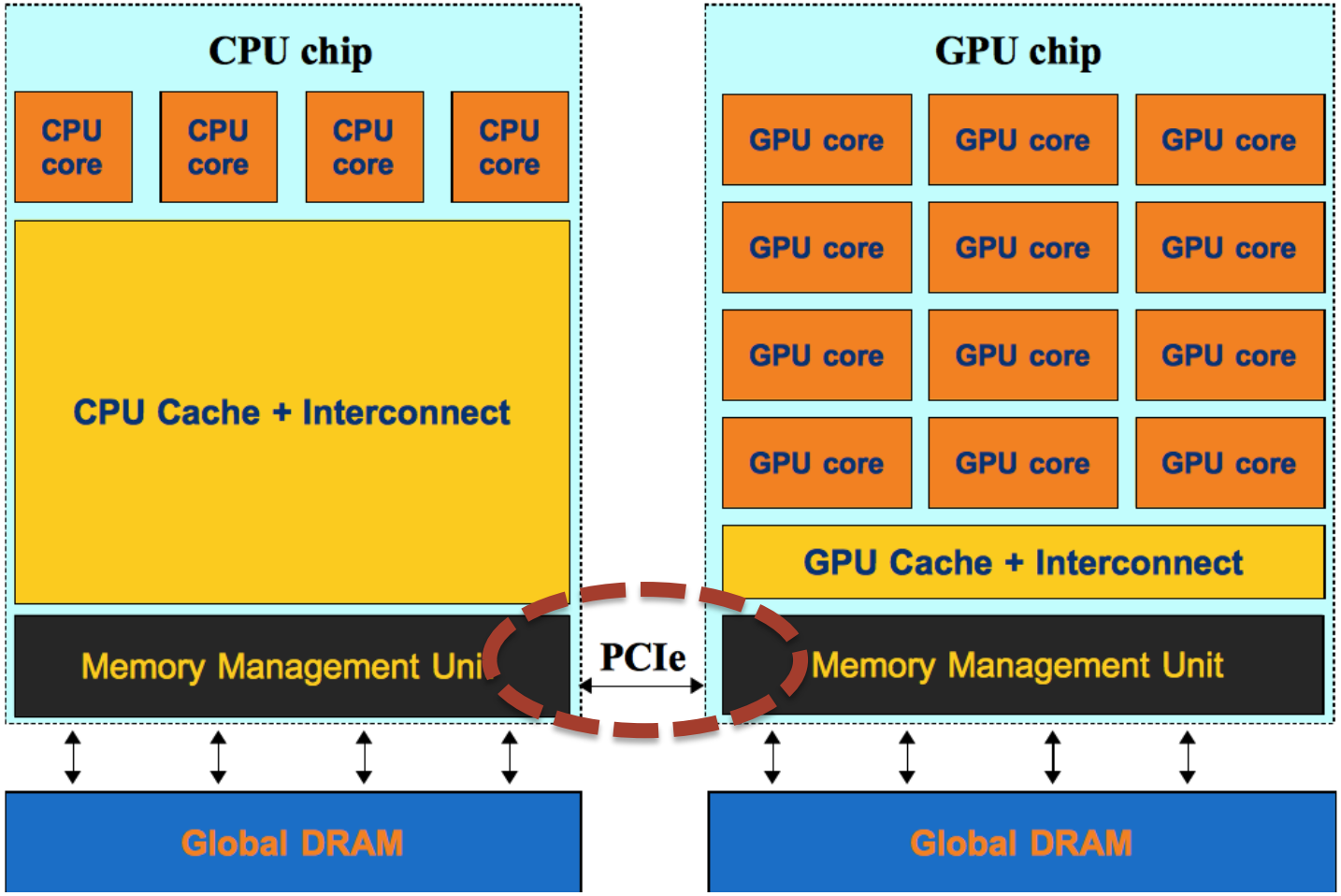
CPU Chip


CPU chip
CPU core
CPU core
CPU core
CPU core
CPU core
CPU core
CPU core
CPU core
CPU Cache + Interconnect
Memory Management Unit
Global DRAM

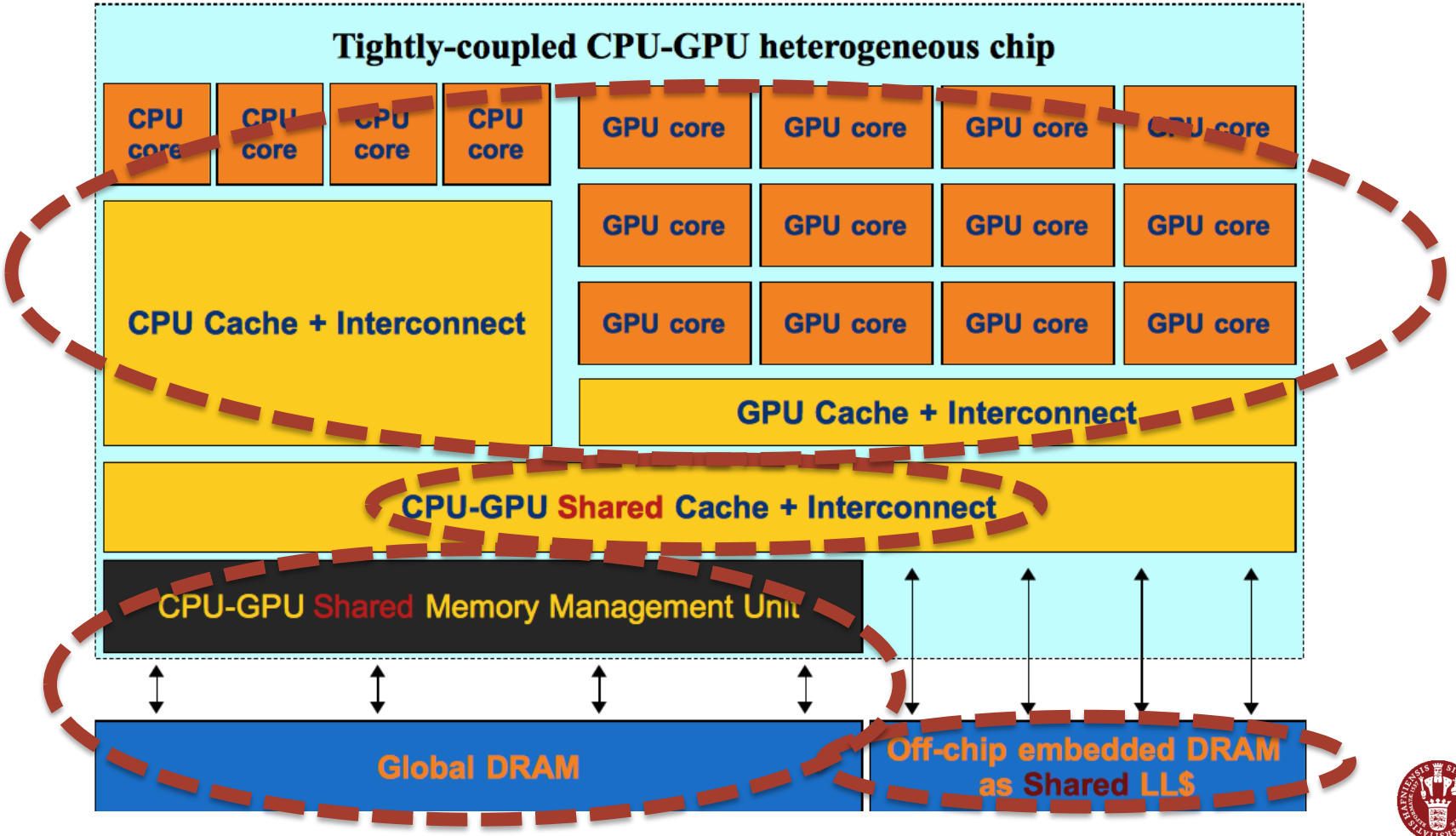
GPU Chip


GPU chip
GPU core
GPU Cache + Interconnect
Memory Management Unit
Global DRAM

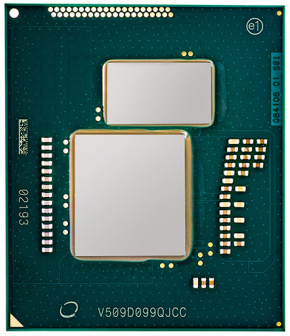
Loosely-coupled CPU-GPU Platform

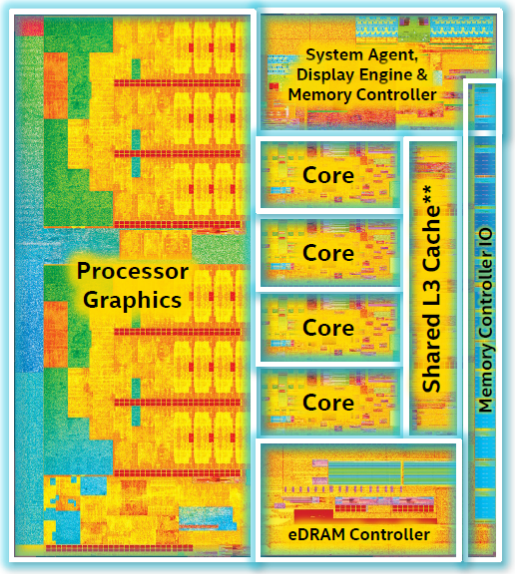


Tightly-coupled CPU-GPU Chip

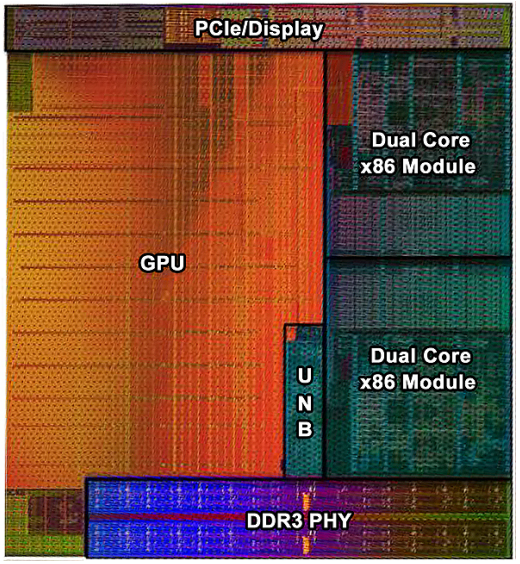


Tightly-coupled CPU-GPU Chip





Intel 5th (Broadwell) CPU-GPU chip
(quad-core, 48 EU, 128 MB eDRAM)



AMD 5th (Kaveri) CPU-GPU chip
(quad-core, 512 Radeon cores)

Three Topics in This Talk

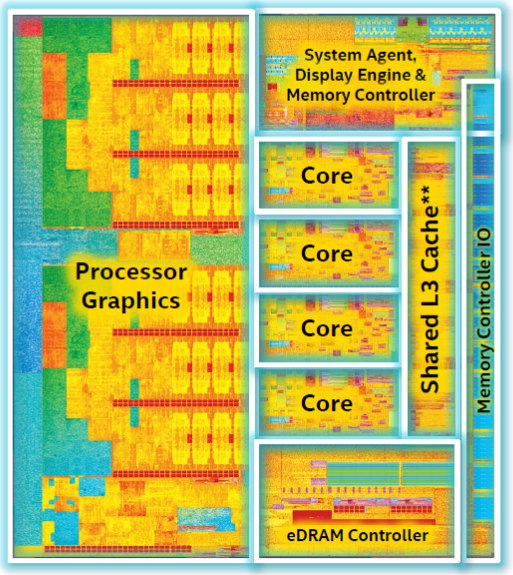
T1. Is eDRAM effective?

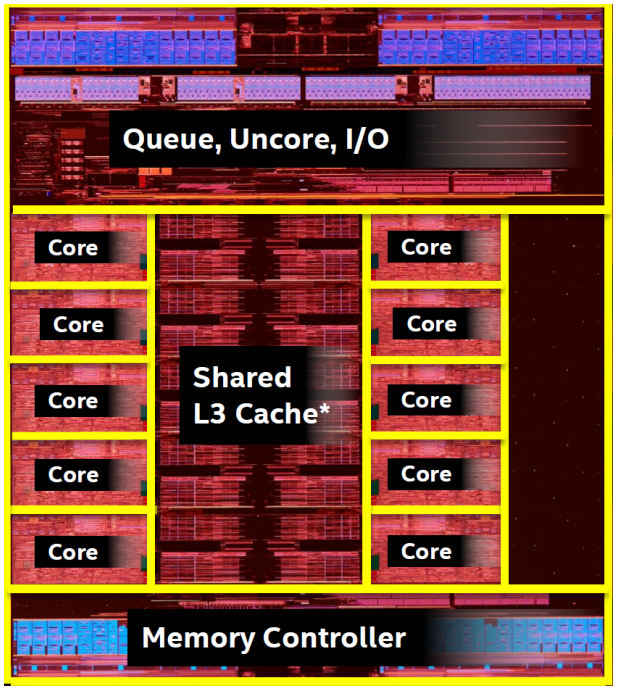
T2. Do Small iGPUs need scalable methods?

T3. Any opportunities for new algorithms?

T1. Effectiveness of eDRAM

Testbed





Intel Core i7-5775c (Broadwell)
CPU: 4 cores @ 3.3-3.7 GHz, 211.2 Gflop/s
GPU: 48 EU @ 1.15 GHz
LLC: 6 MB
eDRAM: 128 MB
DRAM: dual-ch. DDR3-2133, 34.1 GB/s

Intel Xeon E5-2630 v4 (Broadwell)
CPU: 10 cores @ 2.2-3.1 GHz, 352 Gflop/s
GPU: --
LLC: 25 MB
eDRAM: --
DRAM: quad-ch. DDR4-2133, 68.3 GB/s

Three Groups of Benchmark Suites

G1. Sparse BLAS	**G2. Sparse Direct Solvers**	**G3. Benchmark**
SpTRANS	Cholesky	HPCG
SpMV	LU	(High
SpTRSV	QR	Performance
SpGEMM		Conjugate
		Gradients)

255 Sparse Matrices Benchmarked

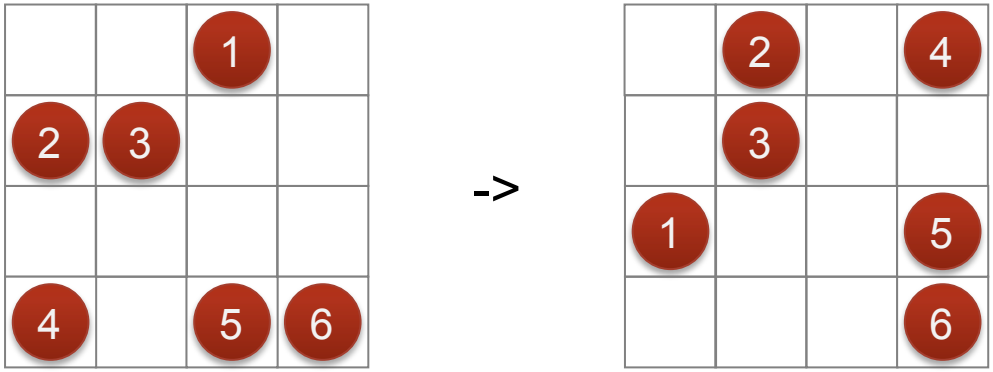
Andrianov (1-7)
Bodendiek (8-11)
Boeing (12-15)
Bourchtein (16-19)
Chen (20-30)
Dense (31)
DIMACS10 (32-89)
DNVS (90-102)
Dziekonski (103-107)
Freescale (108-113)
GHS_indef (114-118)
GHS_psdef (119-133)
Gleich (134-139)
Janna (140-156)
LAW (157-163)
ND (164-167)
Oberwolfach (168-174)
PARSEC (175-187)
Schenk_AFE (188-203)
Schenk_ISEI (204-219)
SNAP (220-232)
TSOPF (233-245)
vanHeukelum (246-249)
Williams (250-255)

Matrix selected from the University of Florida Sparse Matrix Collection:
square, 2M <= #nnz <= 200M, groups having no less than 4 matrices.

Sparse Matrix Transposition

- Example: *B* = sptrans(*A*)

  Transpose a sparse matrix *A* (in CSR) to a sparse matrix *B* (in CSR). Fotmat conversion CSR<->CSC shares the same operation.

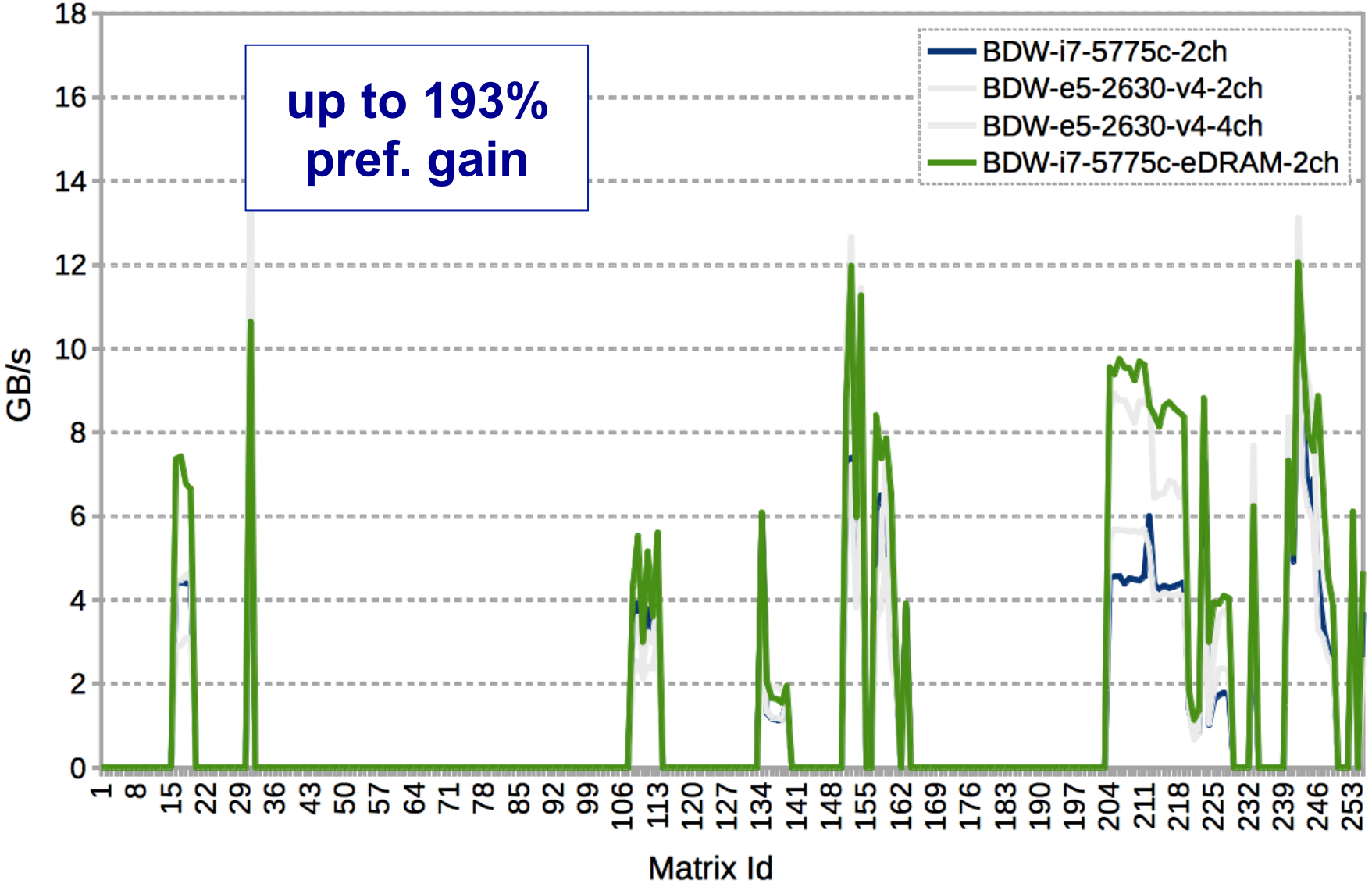


A
(4x4)
nnzA = 6

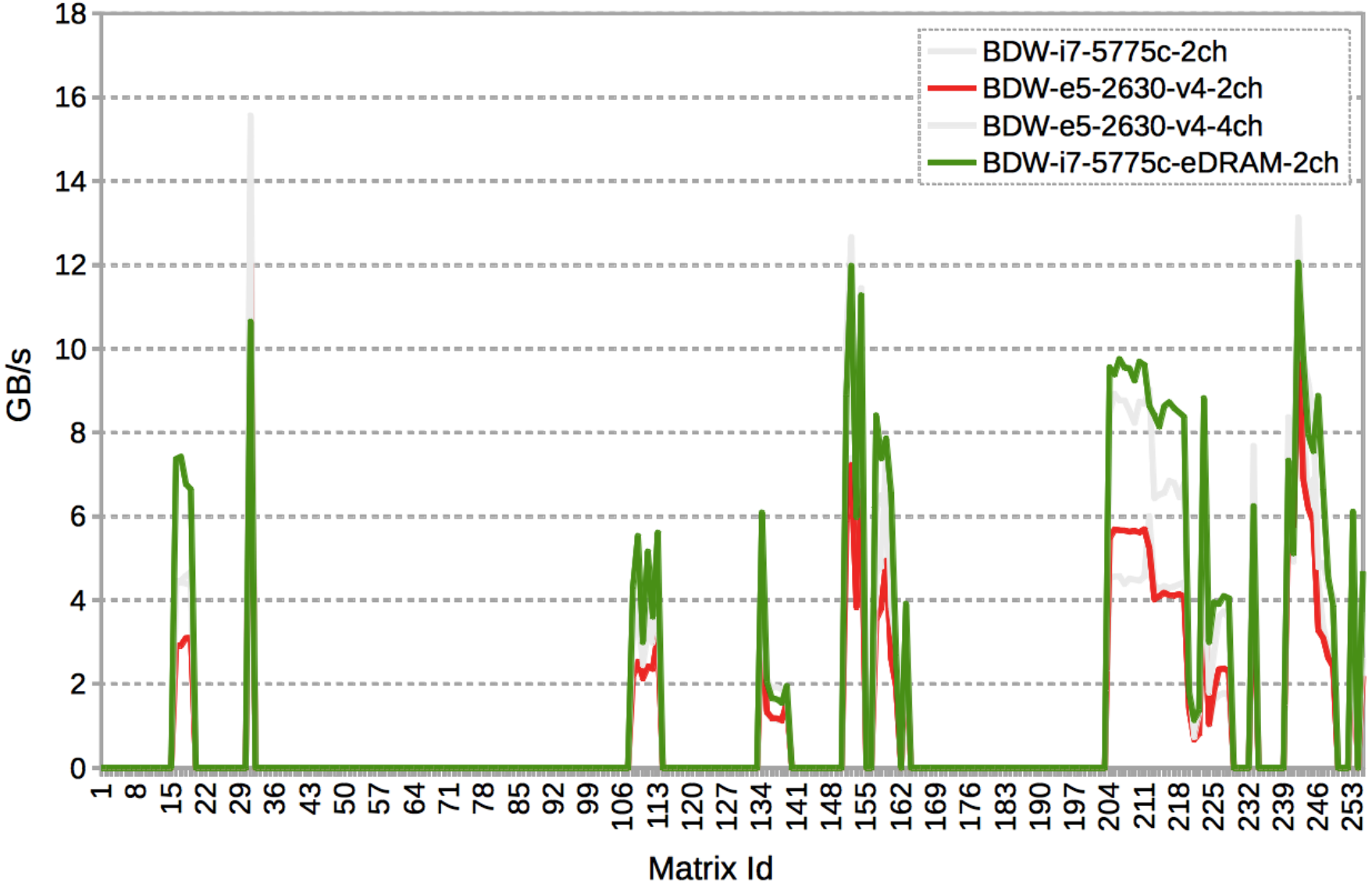
B
(4x4)
nnzA = 6

H. Wang, W. Liu, K. Hou, and W. Feng. **Parallel Transposition of Sparse Data Structures**. *ICS '16*.

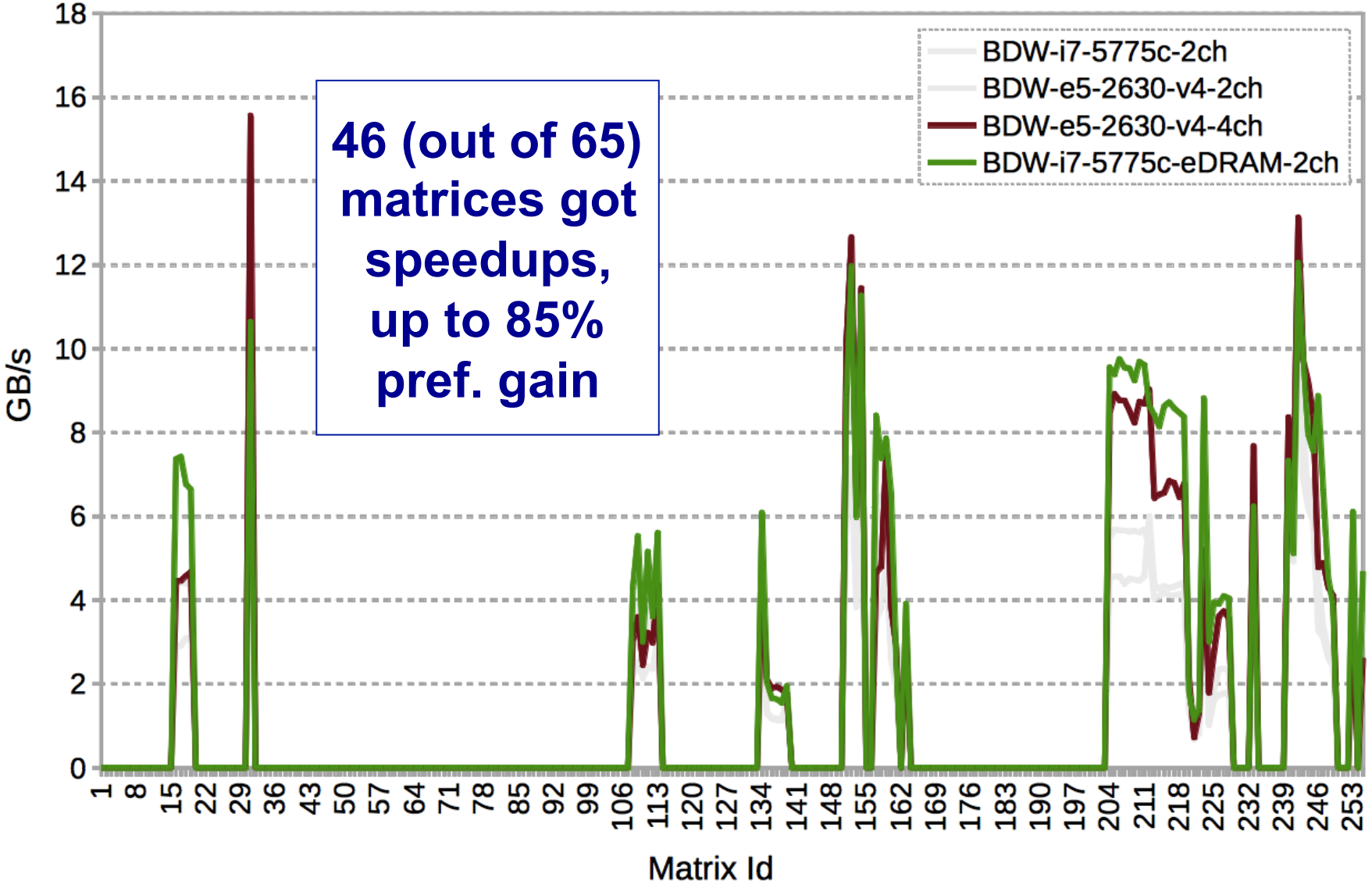
CSR-based SpTRANS (MKL)



CSR-based SpTRANS (MKL)



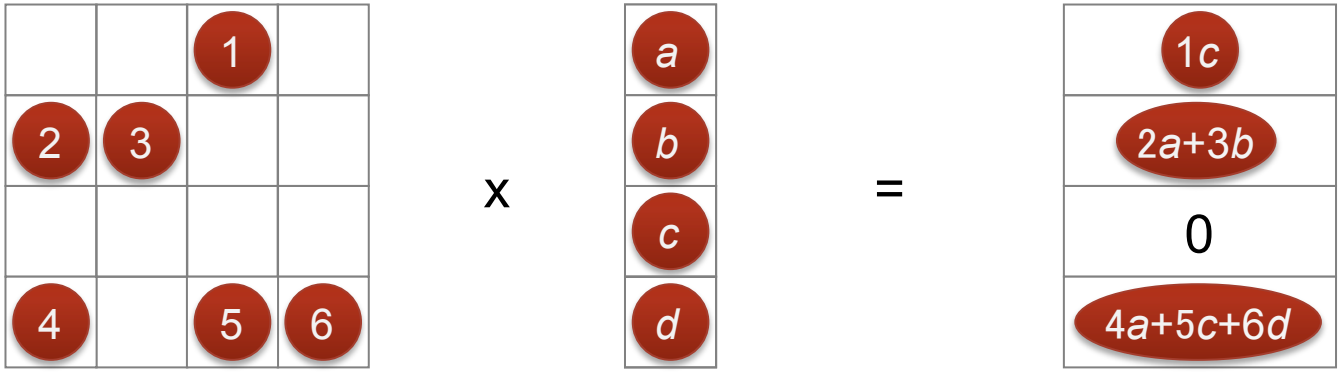
CSR-based SpTRANS (MKL)



Sparse Matrix-Vector Multiplication

- Example: $y = Ax$

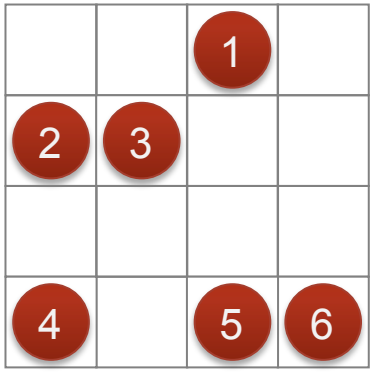
Multiply a sparse matrix A by a dense vector x, obtain a dense vector y.

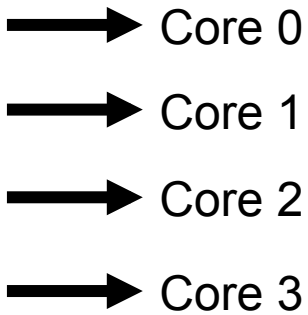


A	x	y
(4x4)	(4x1)	(4x1)
nnzA = 6	*dense*	*dense*

CSR-based SpMV

- assign a row block (multiple rows) to one core.



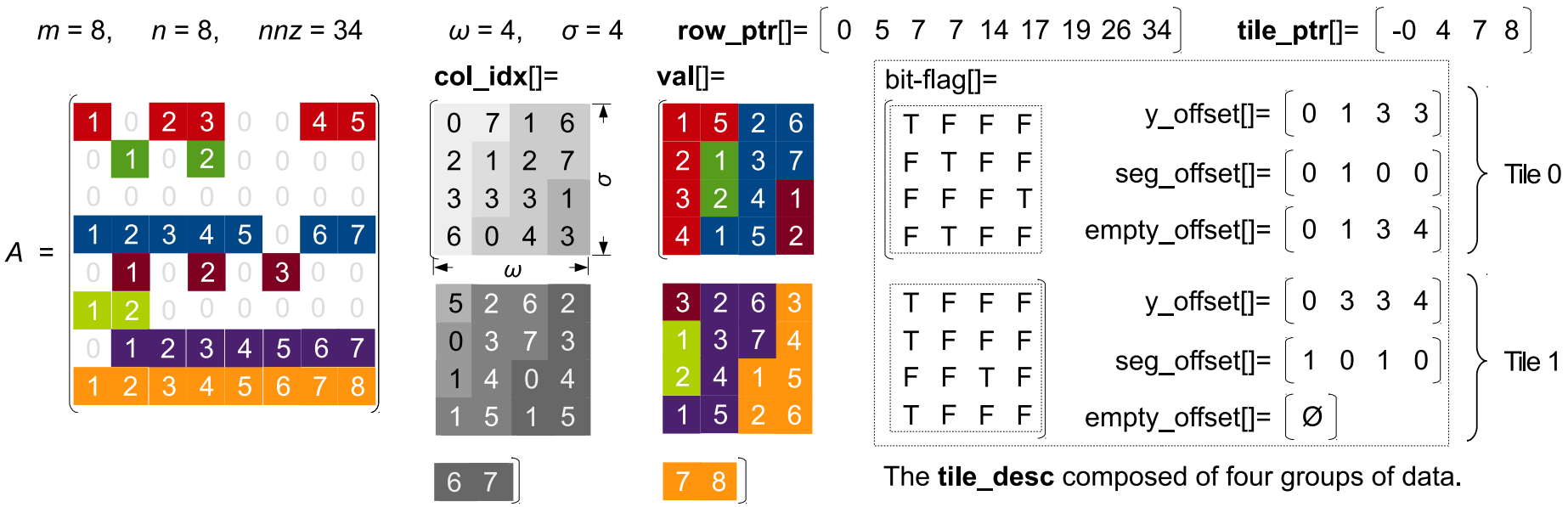


Possibly imbalanced computation

N. Bell and M. Garland. **Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors**. *SC '09*.

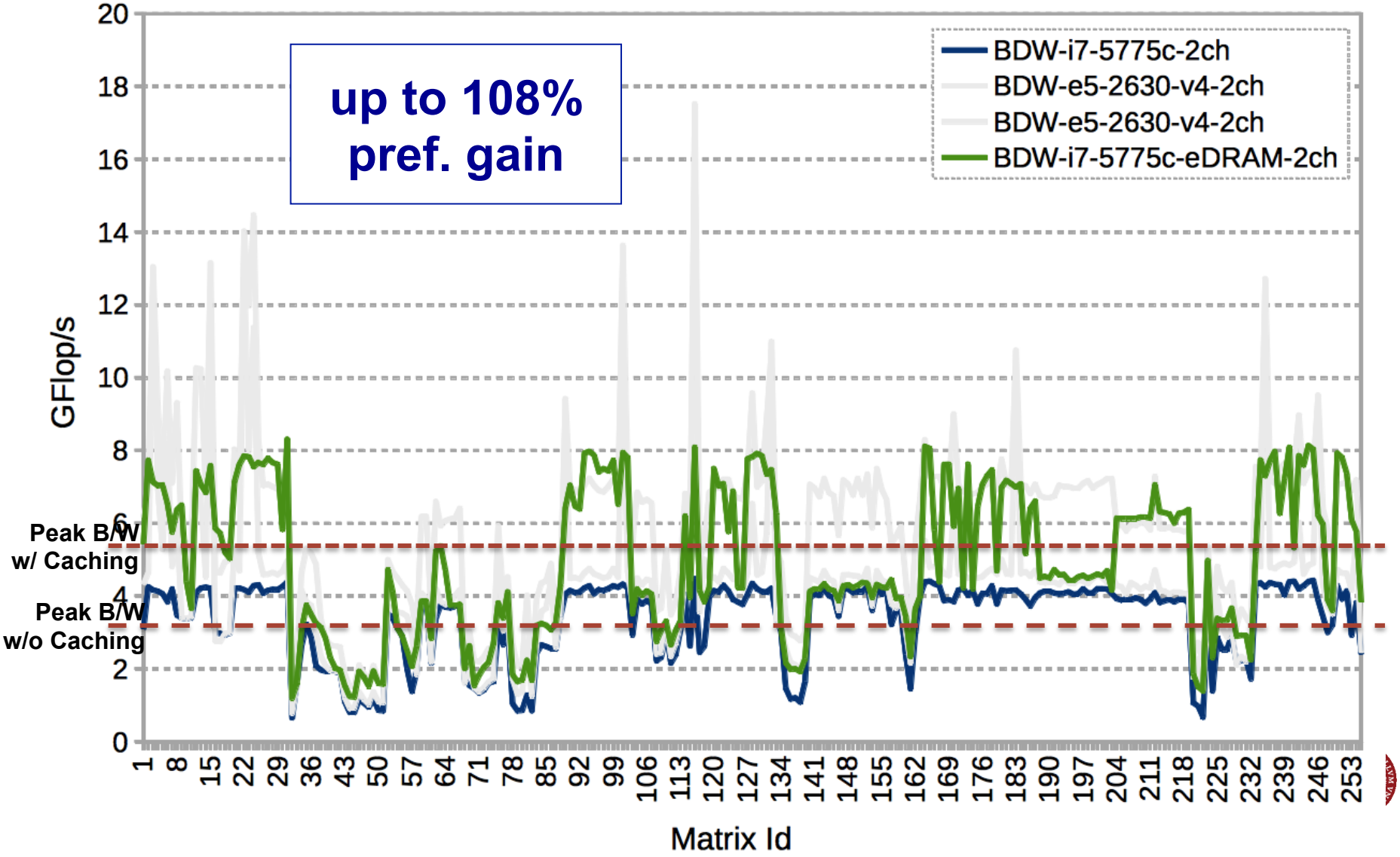
Storage format – CSR5

- Save CSR data in (transposed) tiles of the same size.

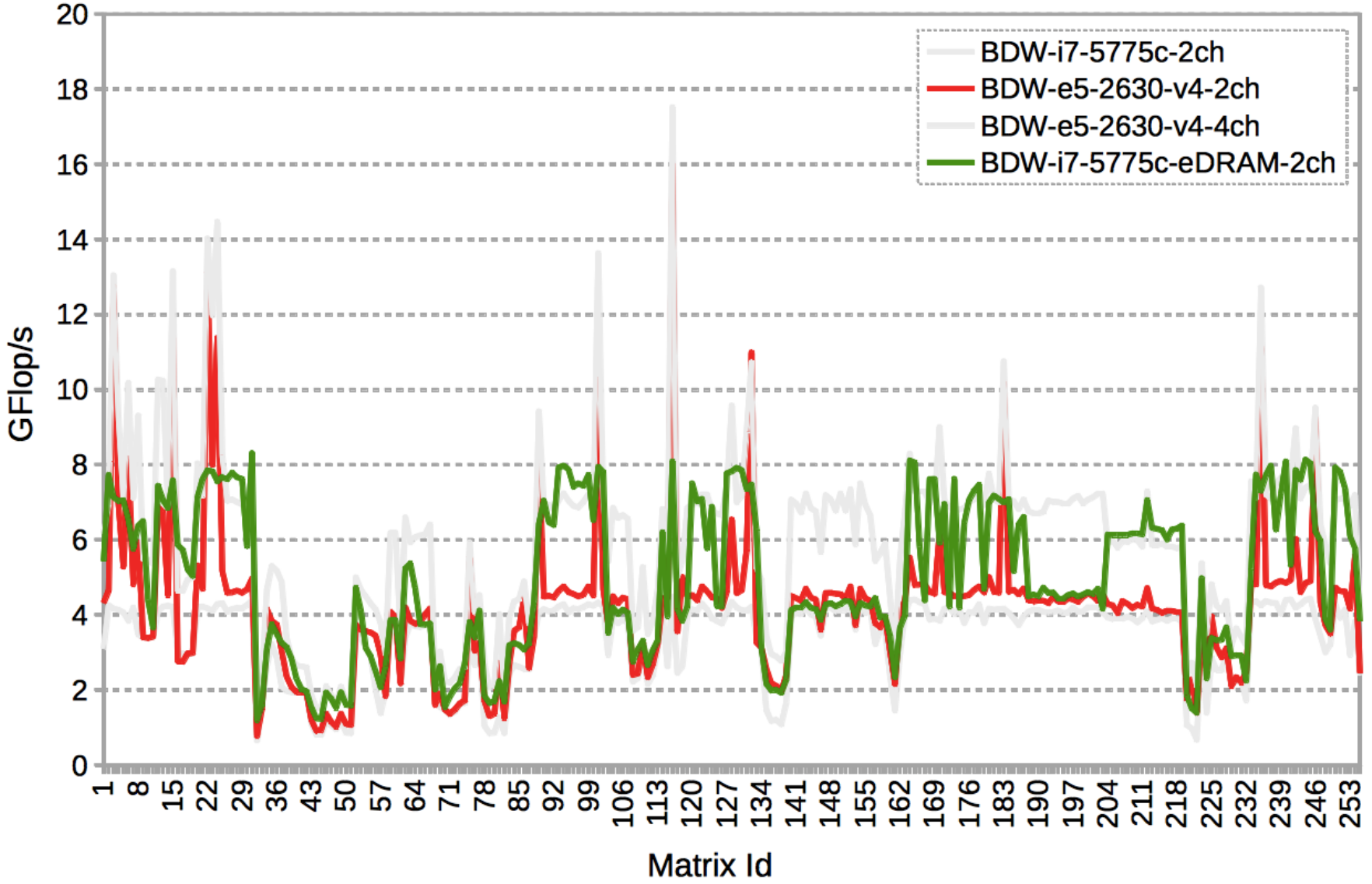


W. Liu and B. Vinter. **CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication**. *ICS '15*.

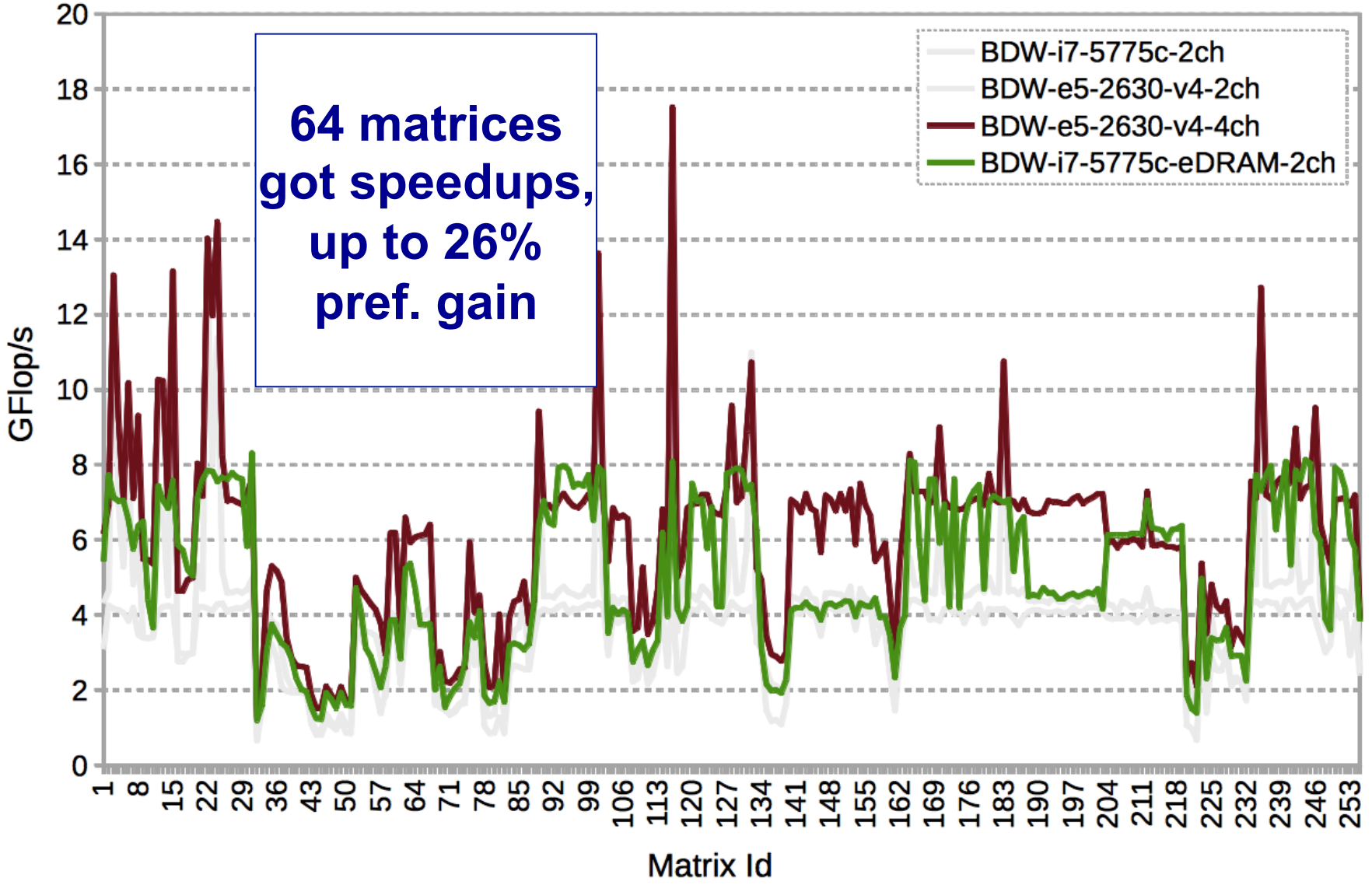
CSR5-based SpMV (bhSPARSE)



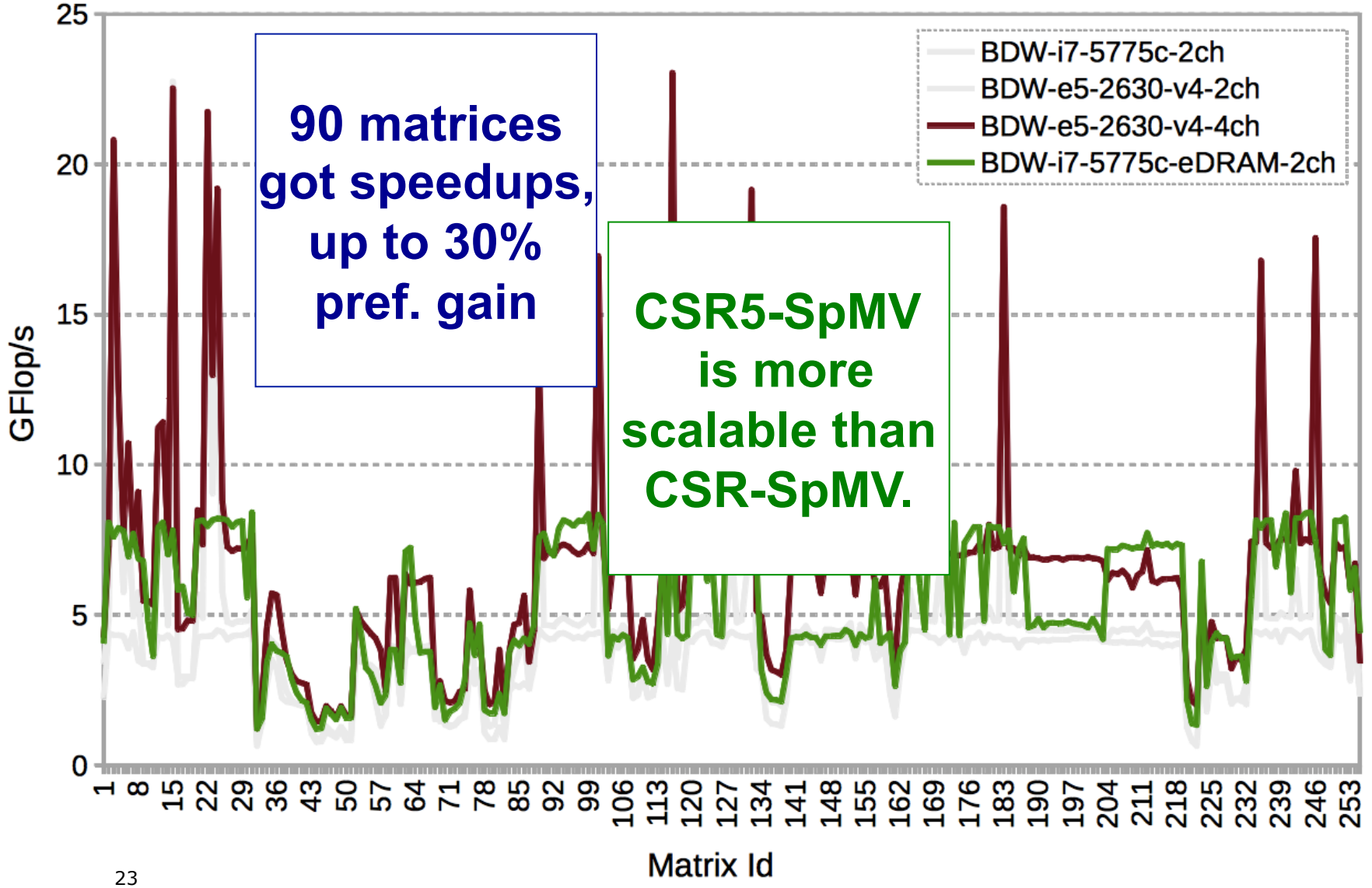
CSR5-based SpMV (bhSPARSE)



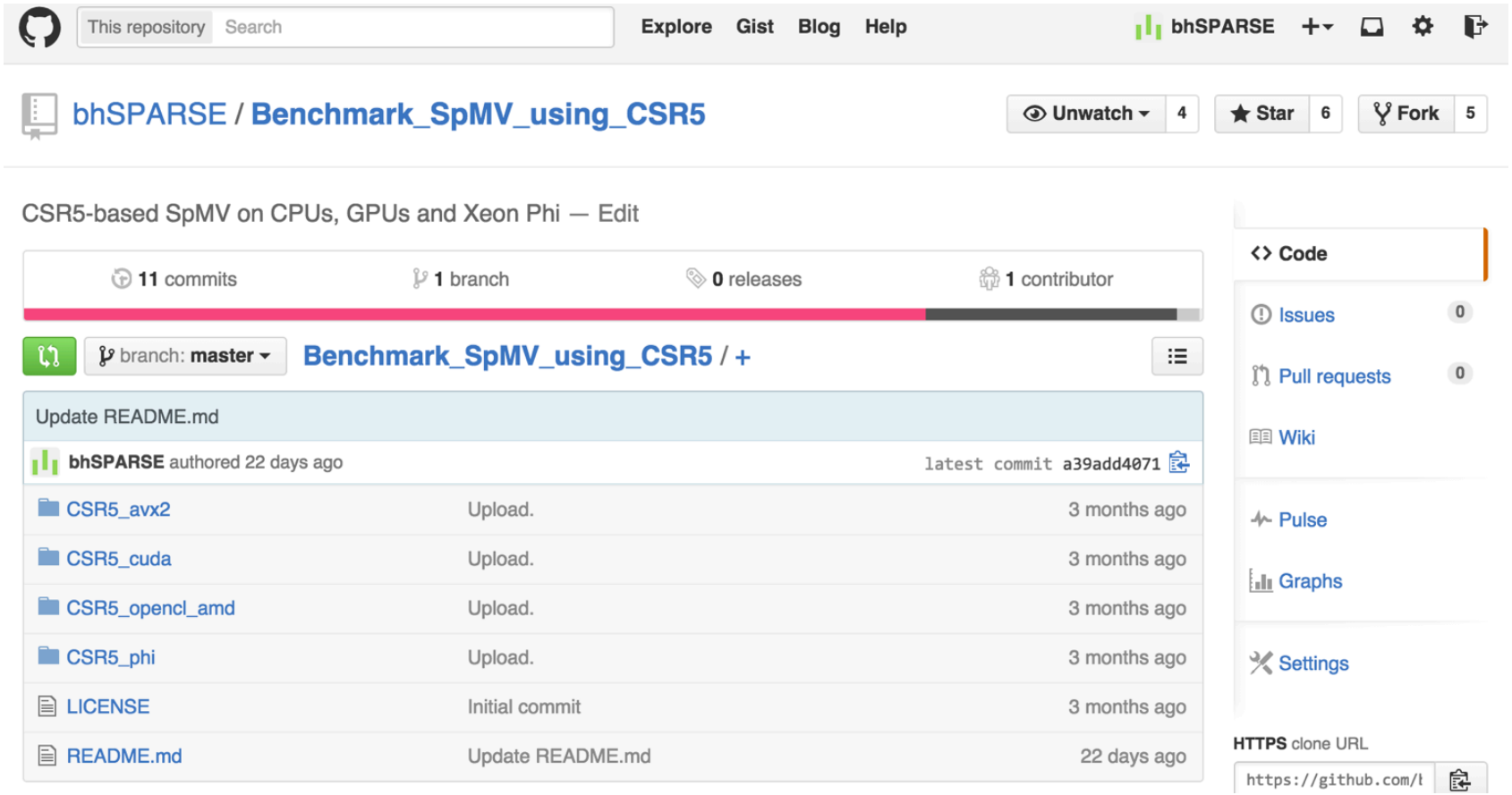
CSR5-based SpMV (bhSPARSE)



CSR-based SpMV (MKL)



Source Code of CSR5 at Github

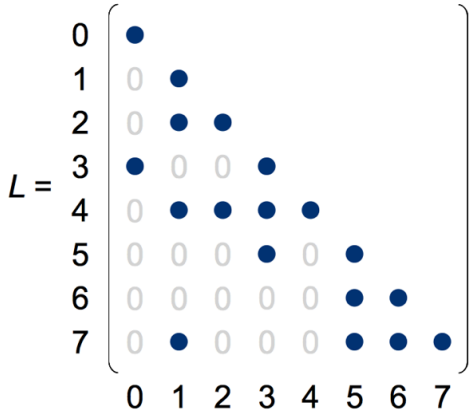


https://github.com/bhSPARSE/Benchmark_SpMV_using_CSR5

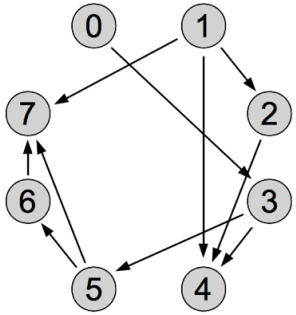
Sparse Triangular Solve

- Example: $Lx = b$

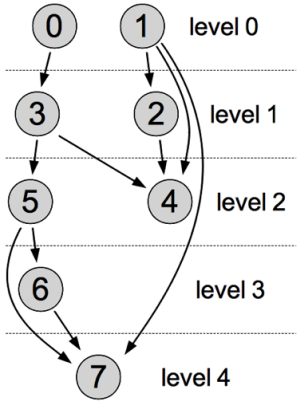
Compute a dense solution vector x from the sparse linear system, where L is a square lower triangular sparse matrix, and b is a dense vector.



L's matrix form



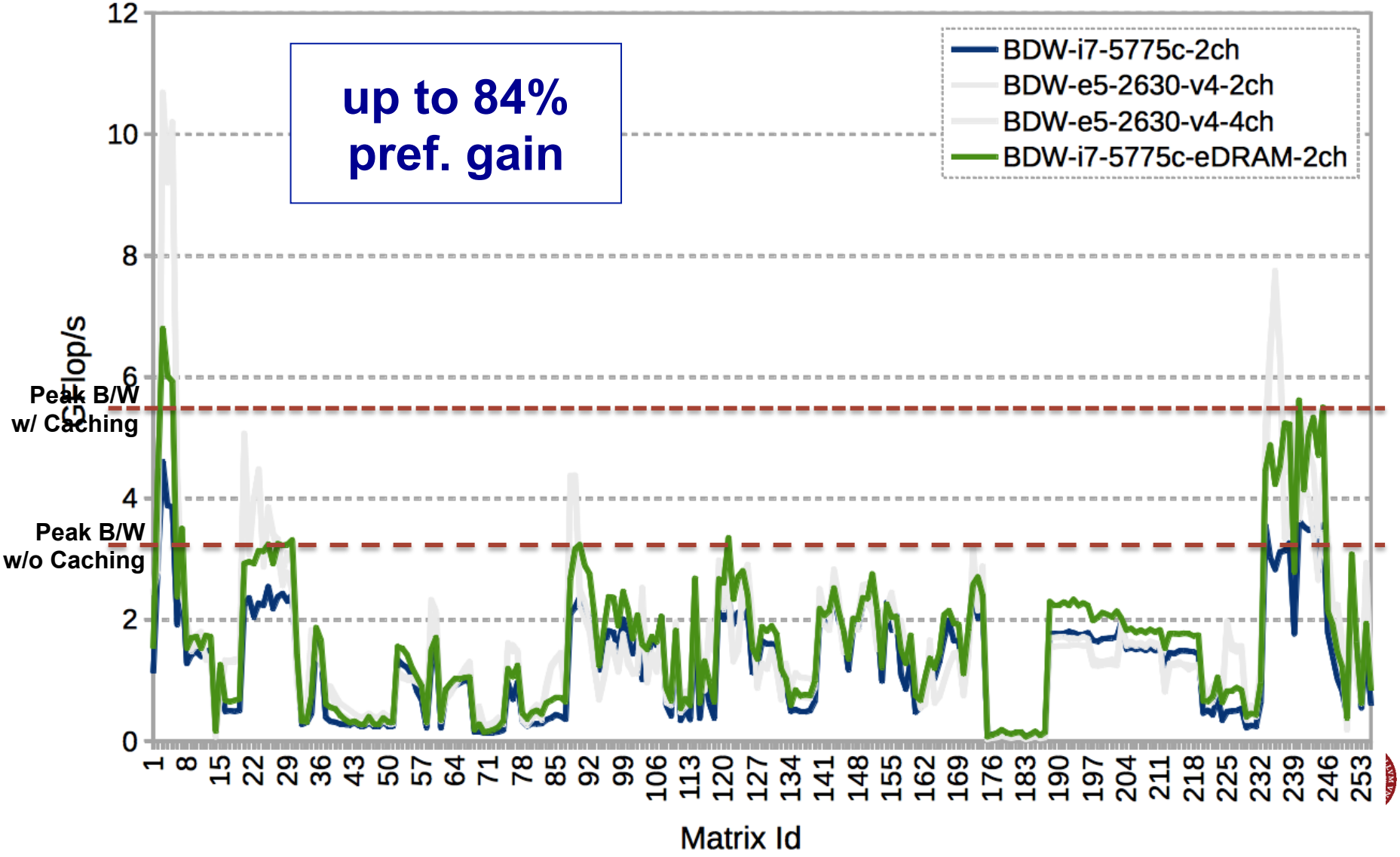
L's graph form



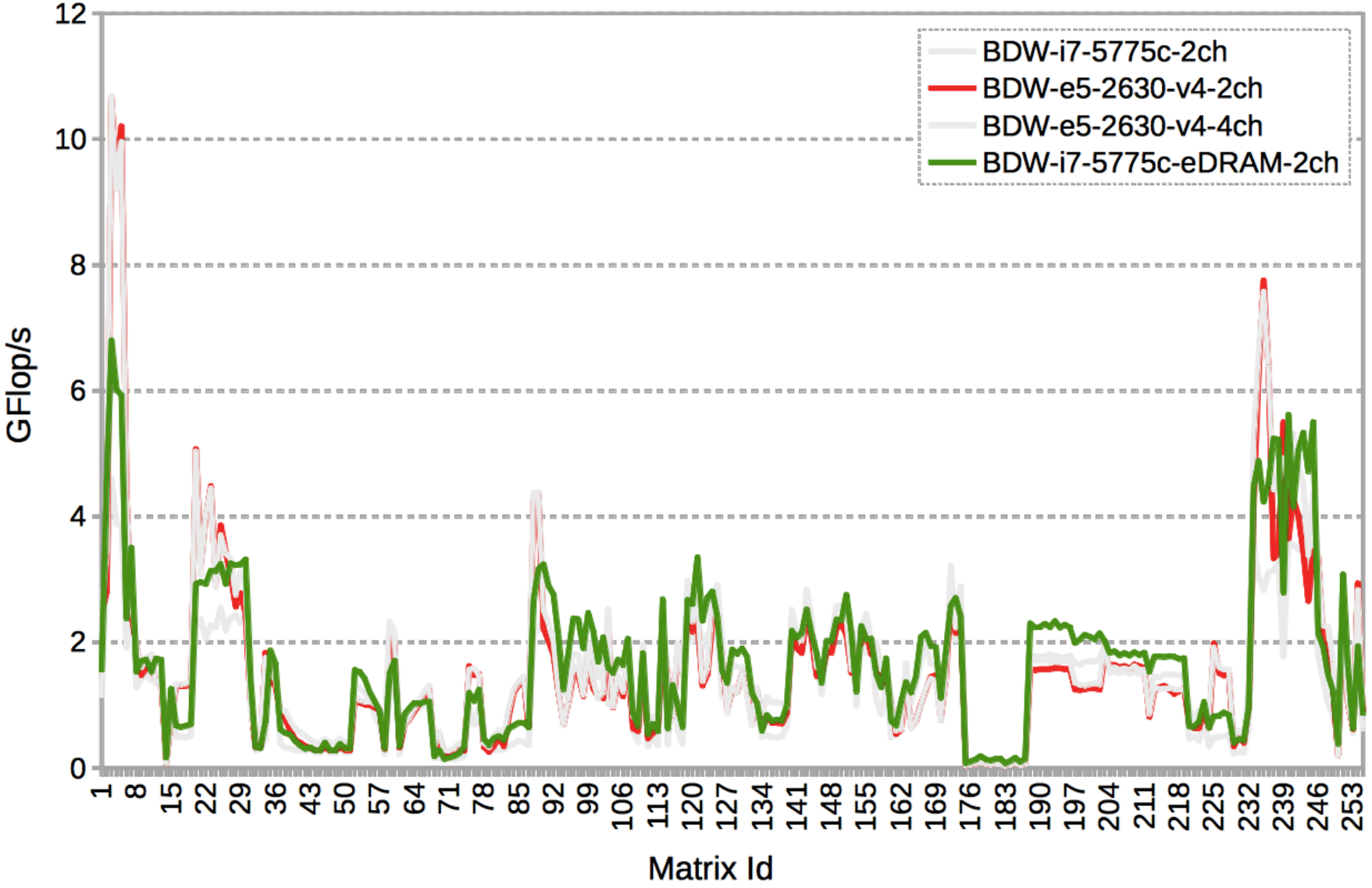
parallelizable Level-sets

W. Liu, A. Li, J. Hogg, I. Duff, and B. Vinter. **A Synchronization-Free Algorithm for Parallel Sparse Triangular Solves**. *Euro-Par '16*.

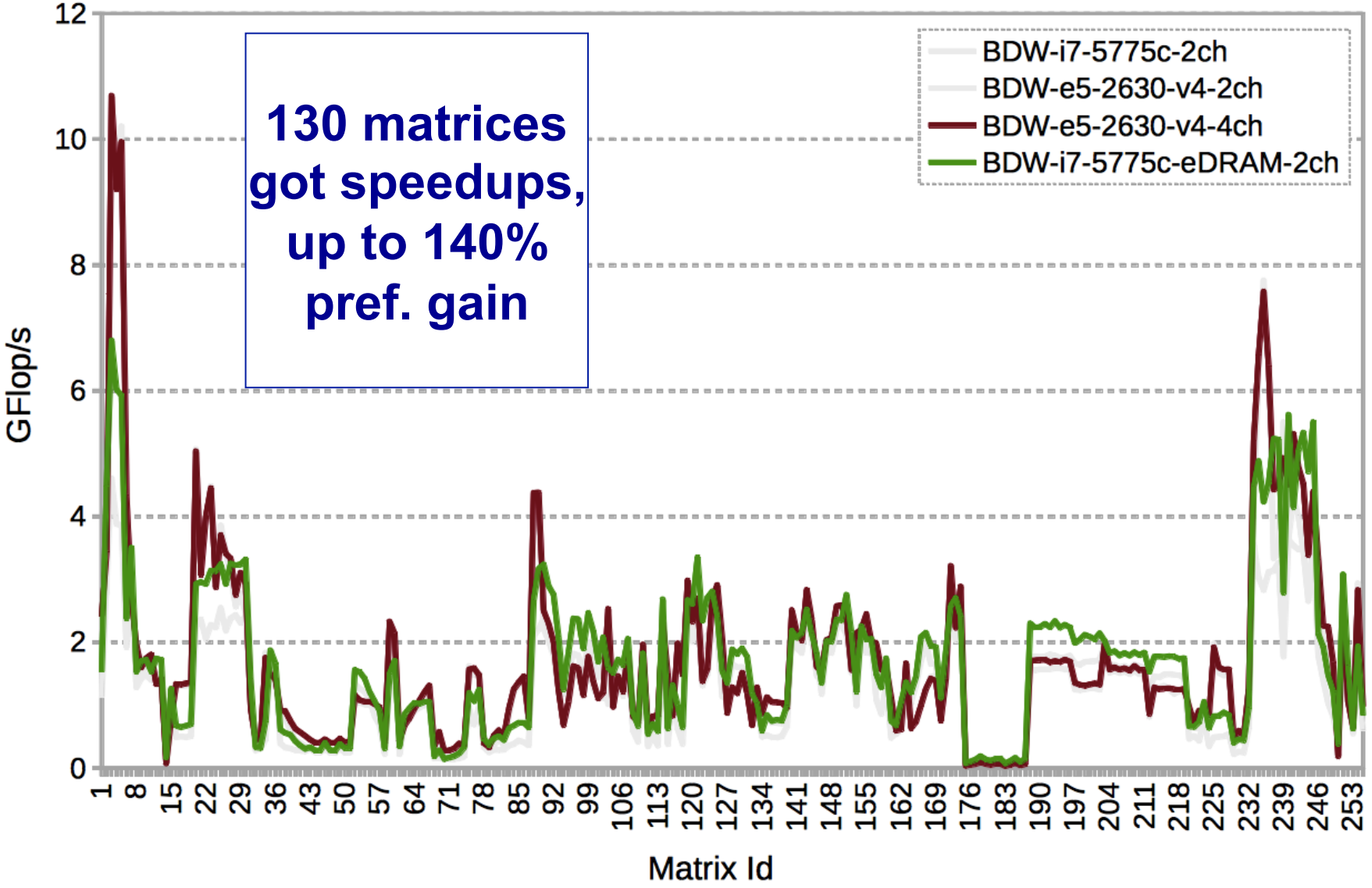
CSR-based SpTRSV (MKL)



CSR-based SpTRSV (MKL)



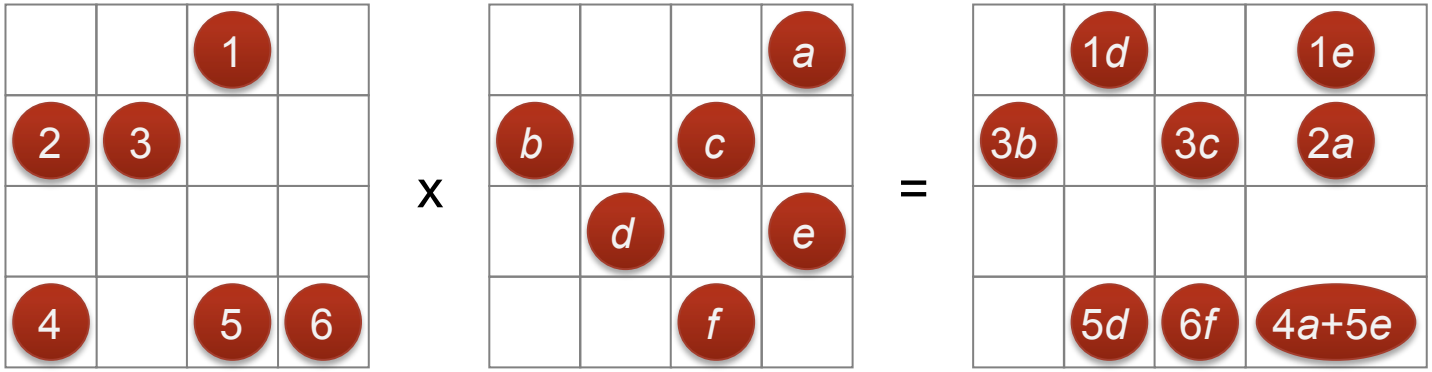
CSR-based SpTRSV (MKL)



Sparse Matrix-Matrix Multiplication

- Example: $C = AB$

Multiply a sparse matrix A by a sparse matrix B, obtain a sparse matrix C.

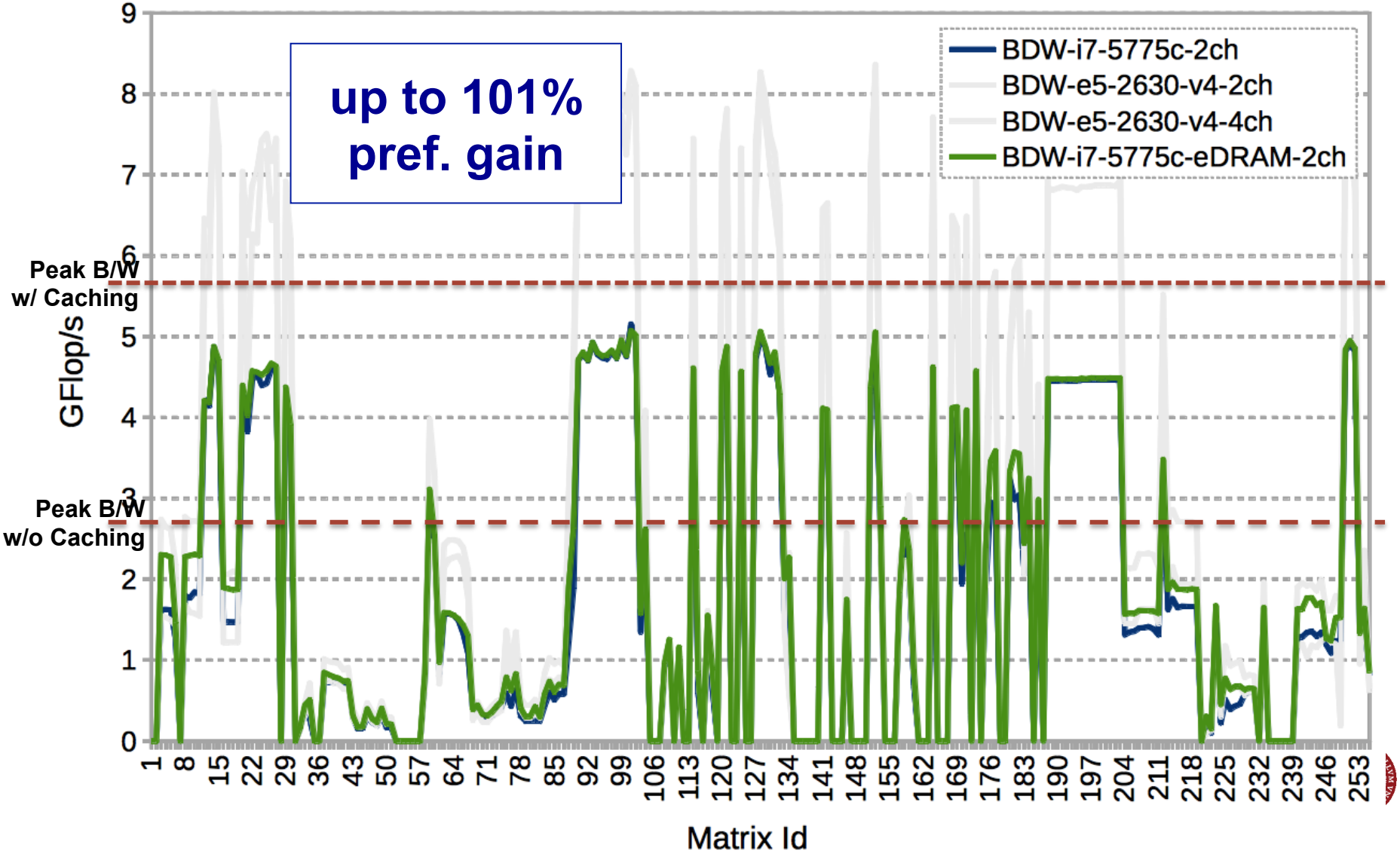


A
(4x4)
$nnzA = 6$

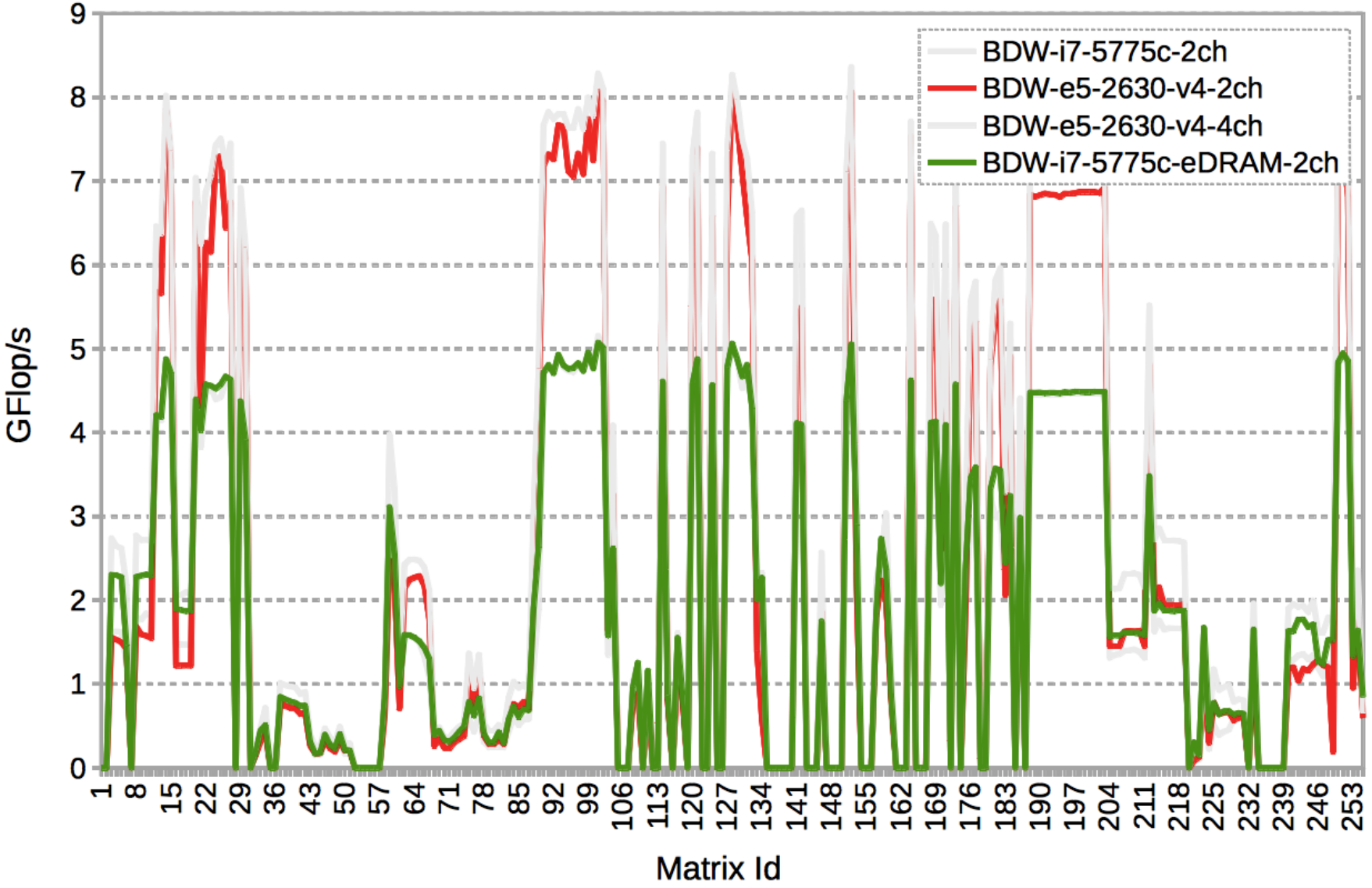
B
(4x4)
$nnzB = 6$

C
(4x4)
$nnzC = 8$

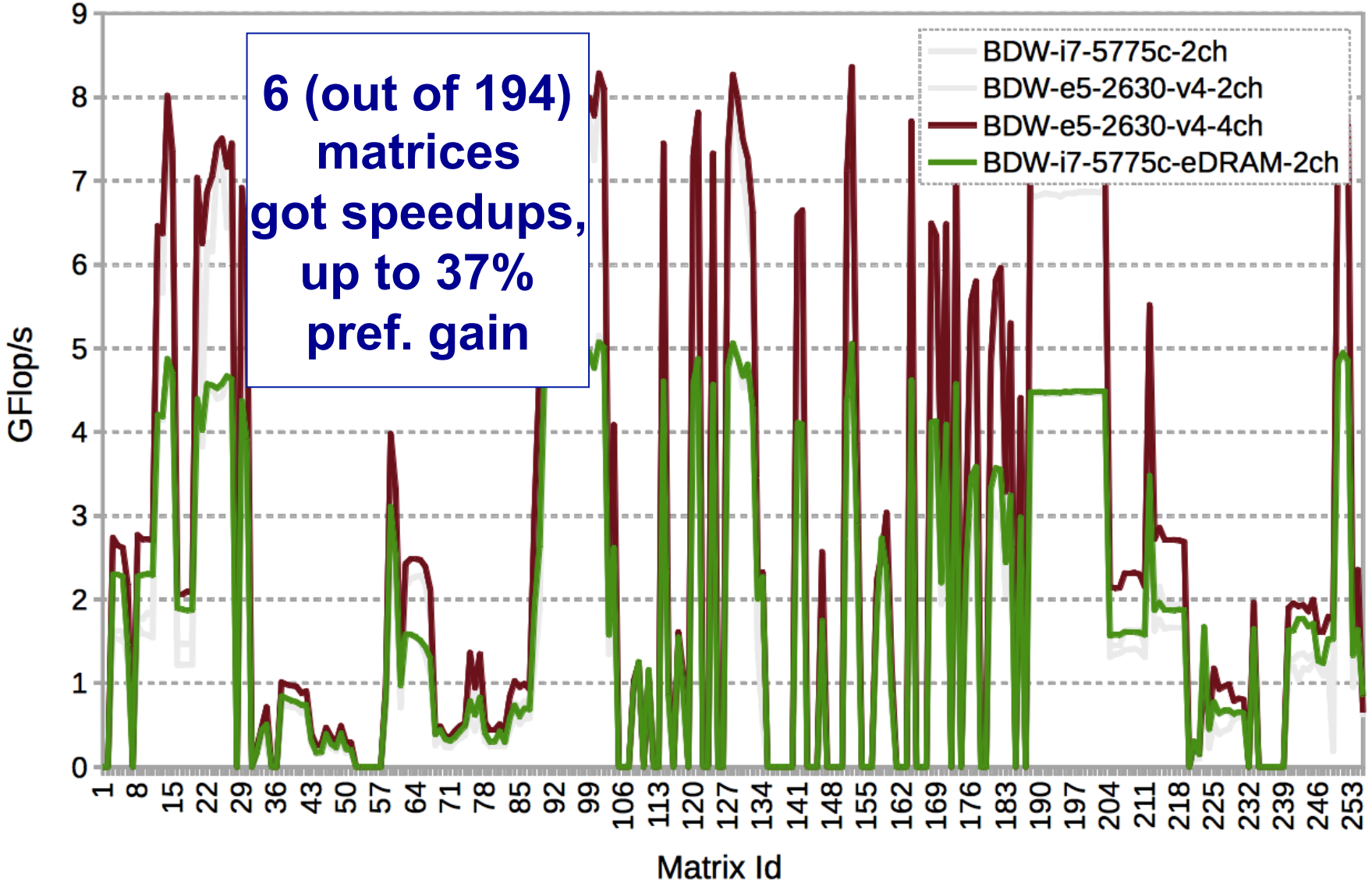
CSR-based SpGEMM (MKL)



CSR-based SpGEMM (MKL)

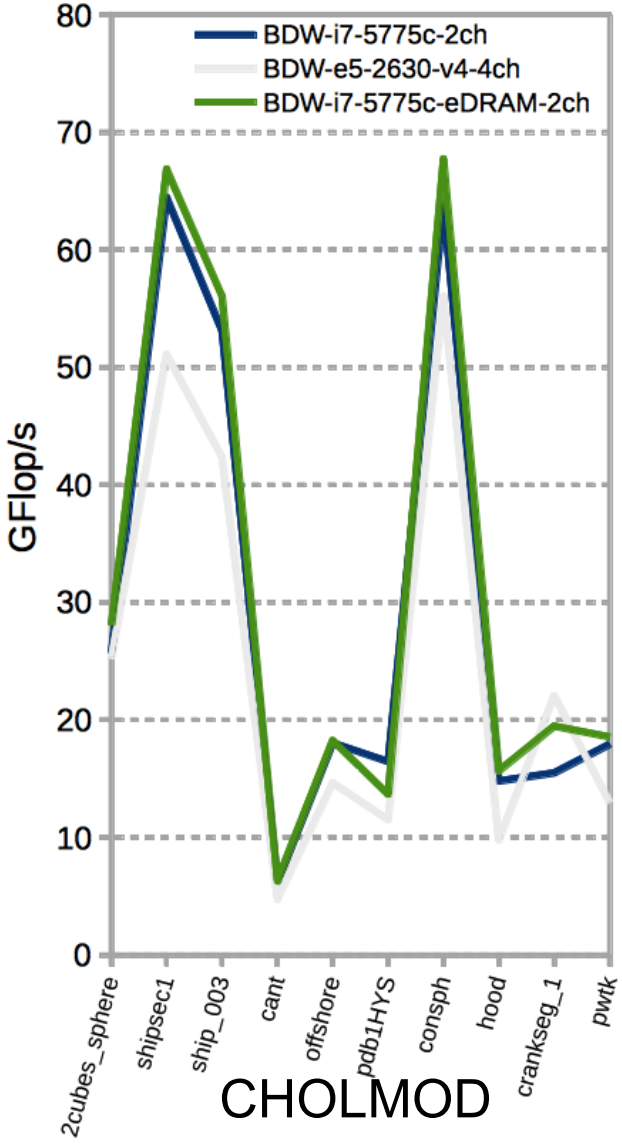


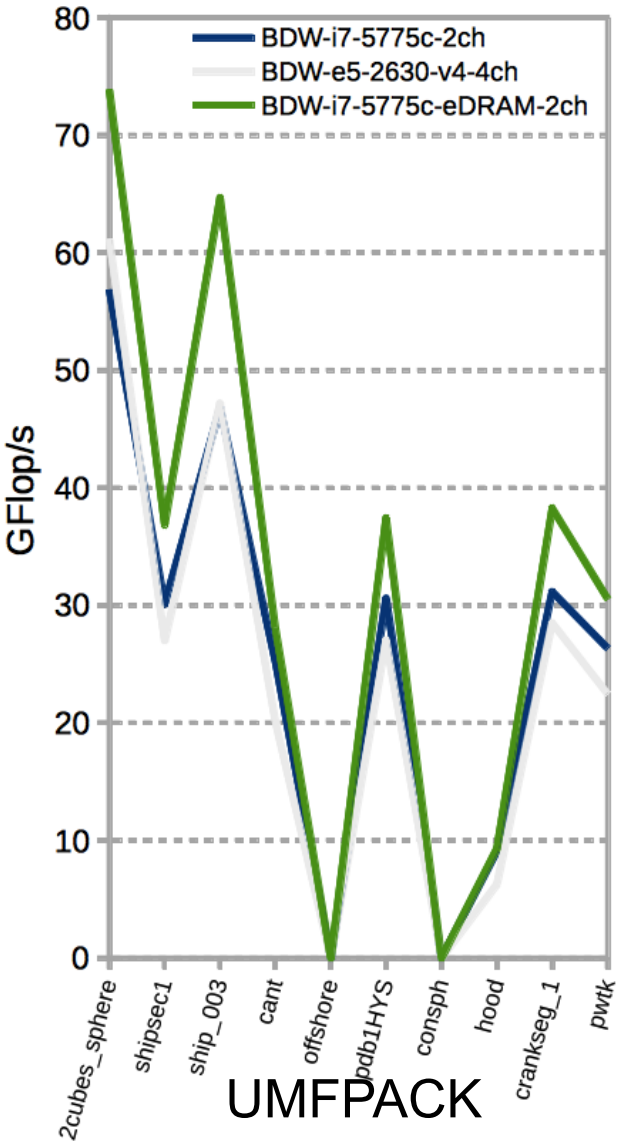
CSR-based SpGEMM (MKL)

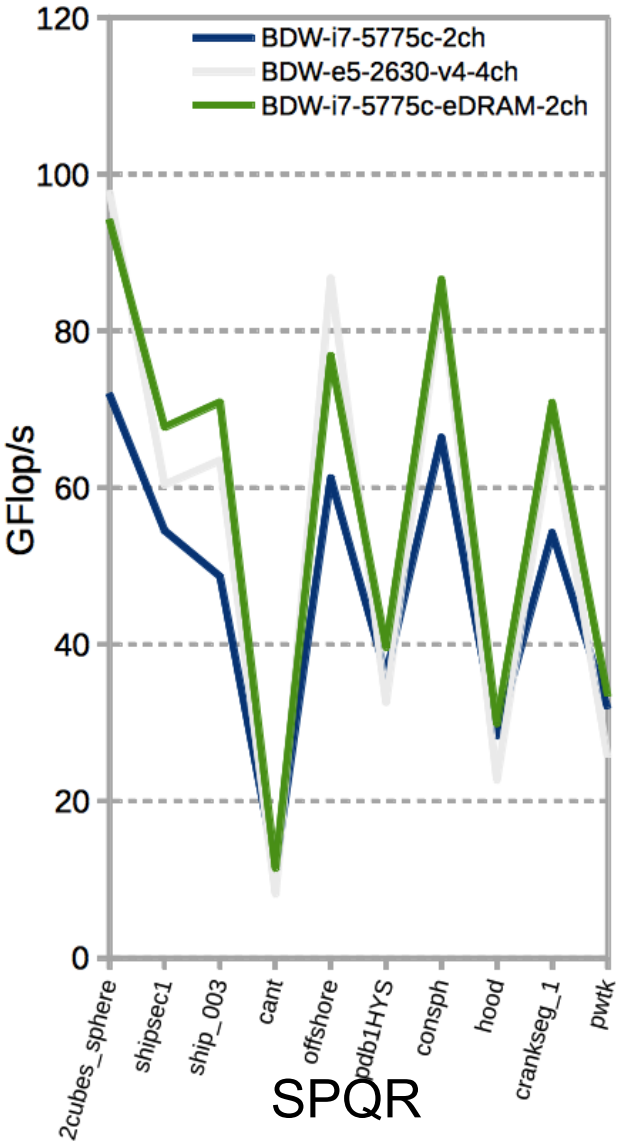


Sparse Direct Solvers

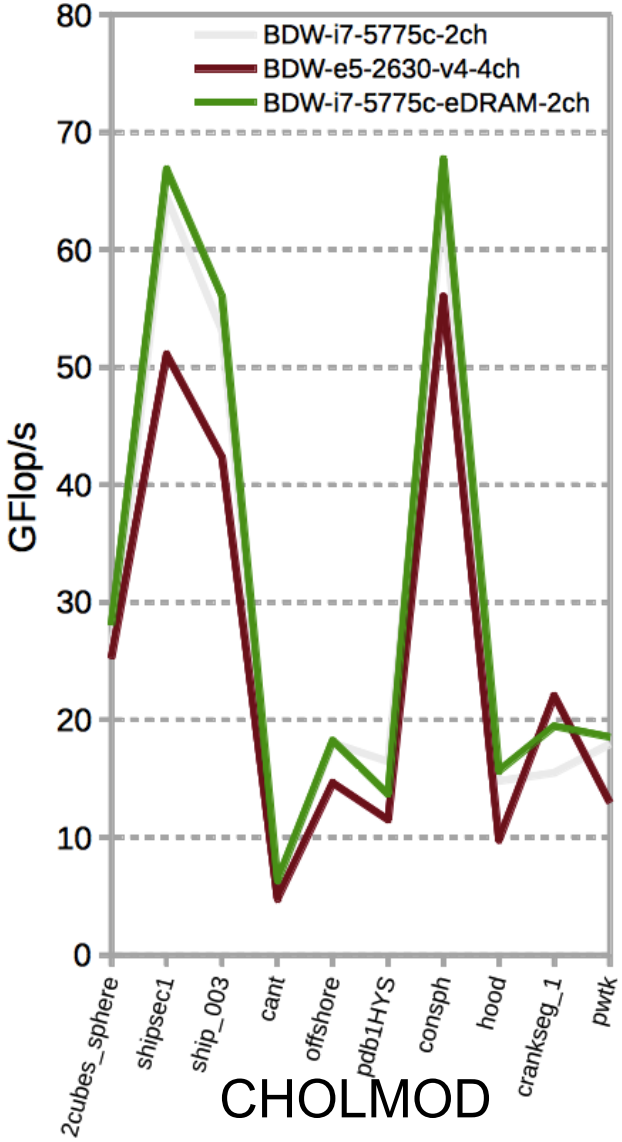
Cholesky, LU, & QR (SuiteSparse)

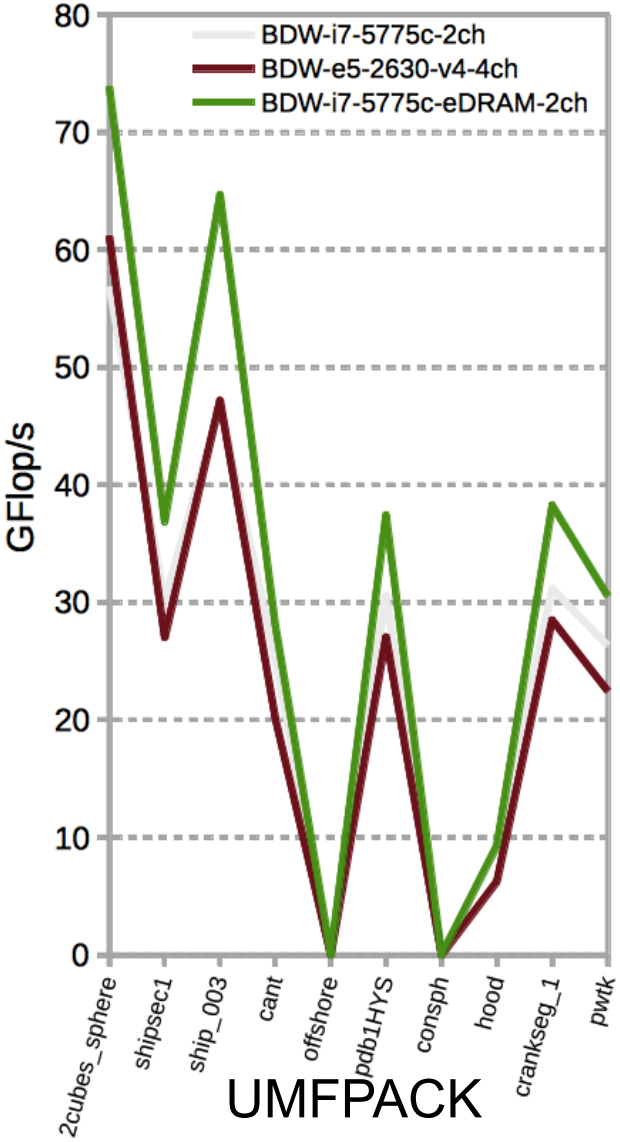


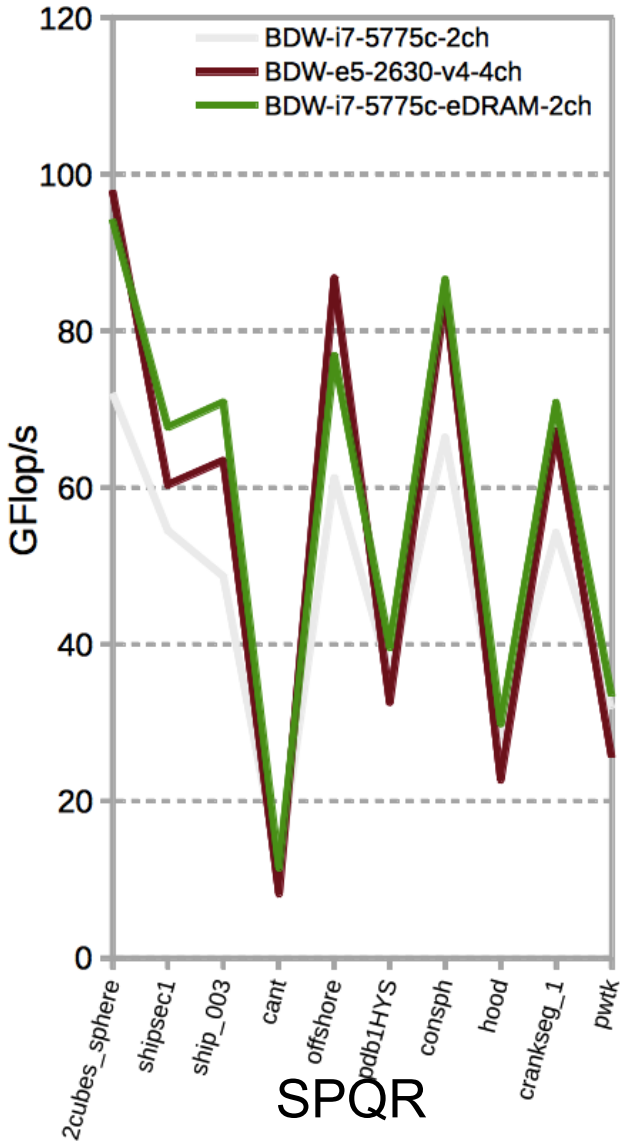




Cholesky, LU, & QR (SuiteSparse)

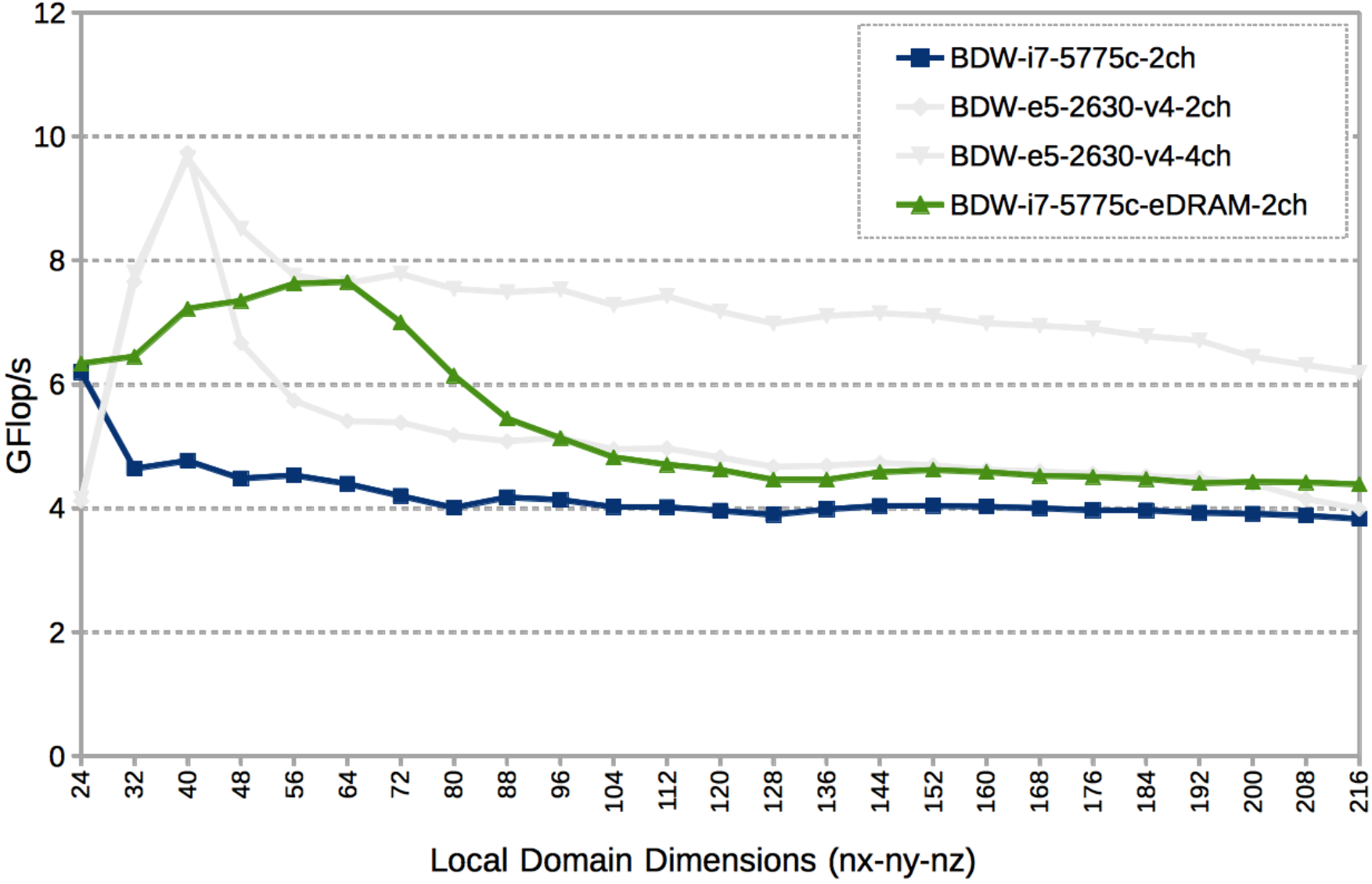




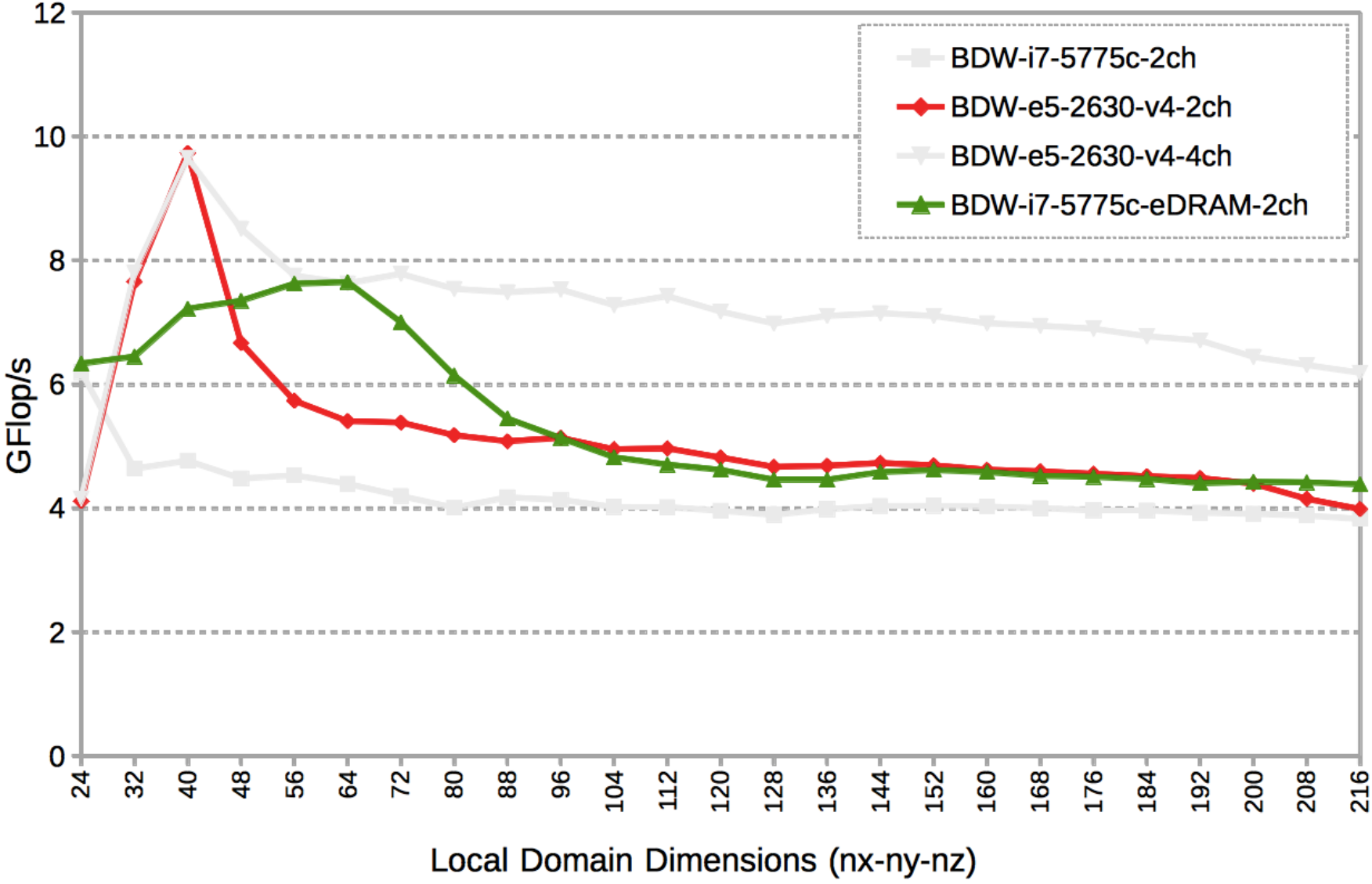


HPCG Benchmark

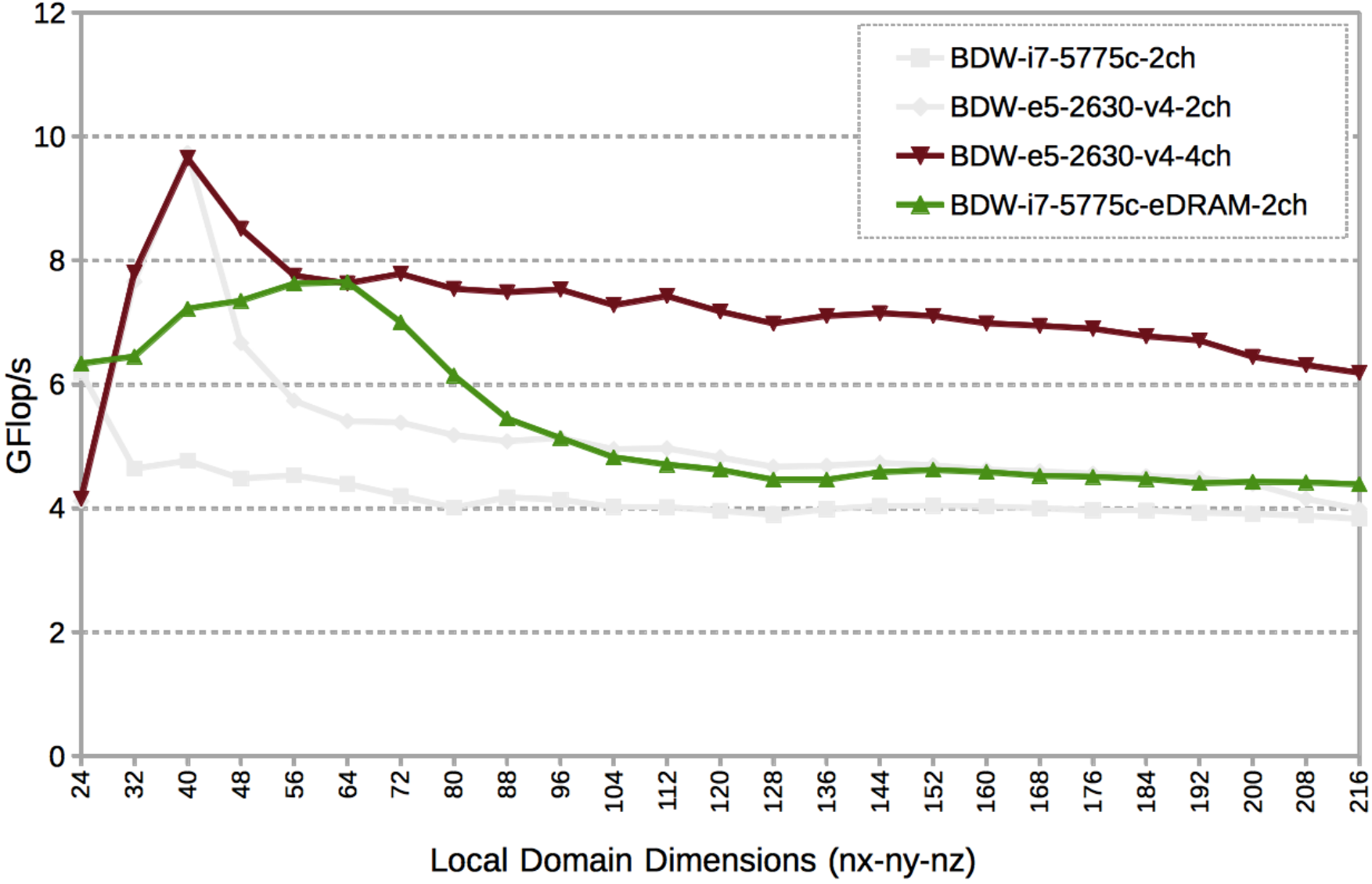
HPCG Benchmark



HPCG Benchmark



HPCG Benchmark

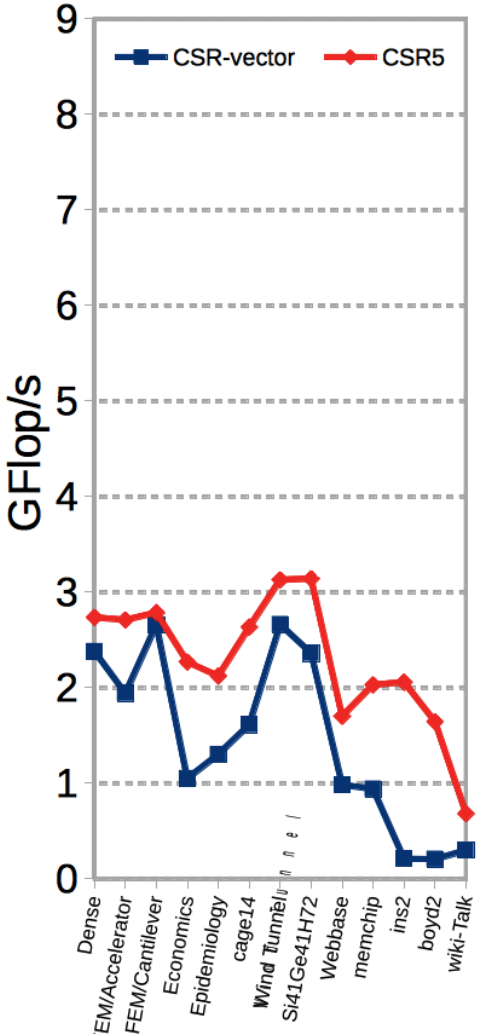


T2. Scalability on Integrated GPU

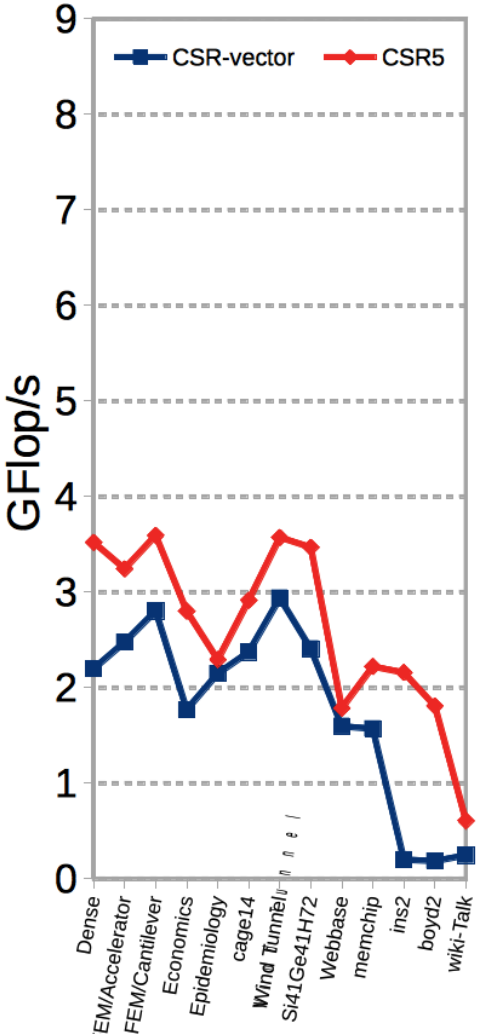
Testbed

	i3-5010u	i5-5250u	i7-5775c	i7-5775c
Family	Broadwell	Broadwell	Broadwell	Broadwell
CPU	2c @ 2.1 GHz	2c @ 2.7 GHz	4c @ 3.7 GHz	4c @ 3.7 GHz
GPU	**24eu** @ 0.9 GHz	**48eu** @ 0.95 GHz	**48eu** @ 1.15 GHz	**48eu** @ 1.15 GHz
LLC	3 MB	3 MB	6 MB	6 MB
eDRAM	--	--	--	**128 MB**
DRAM	2ch **DDR3-1600**	2ch **DDR3-1600**	2ch **DDR3-2133**	2ch **DDR3-2133**
Bandwidth	25.6 GB/s	25.6 GB/s	34.1 GB/s	34.1 GB/s
TDP	15w	15w	65w	65w

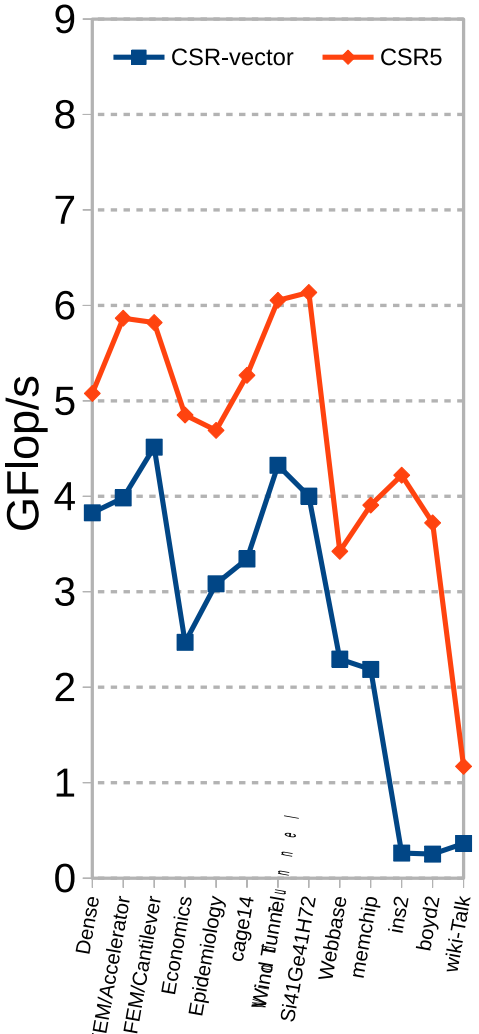
Scalability (device-wise)



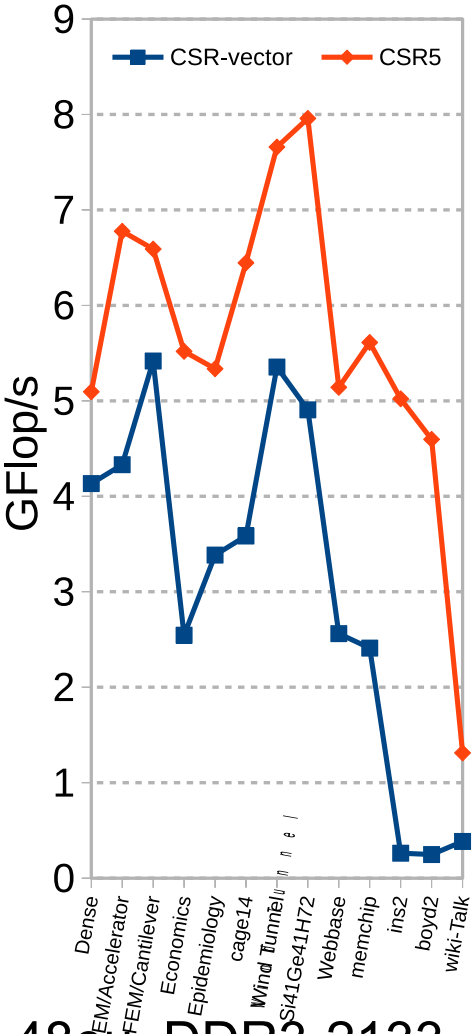
24eu, DDR3-1600



48eu, DDR3-1600

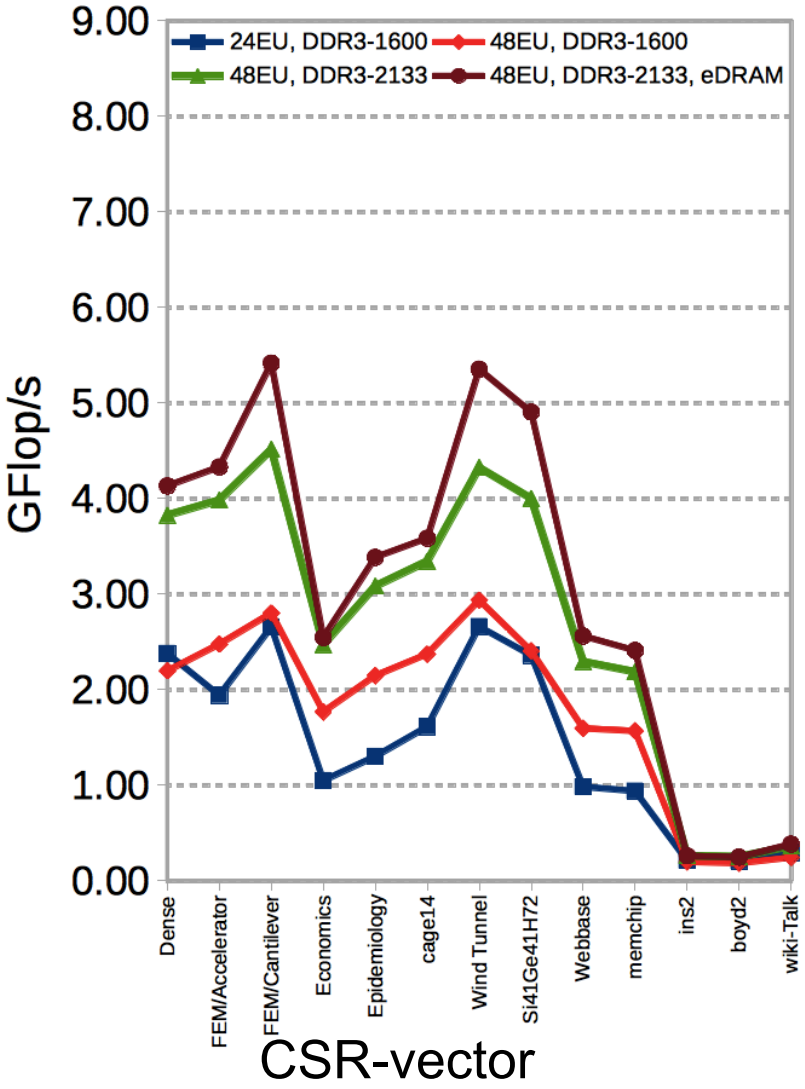


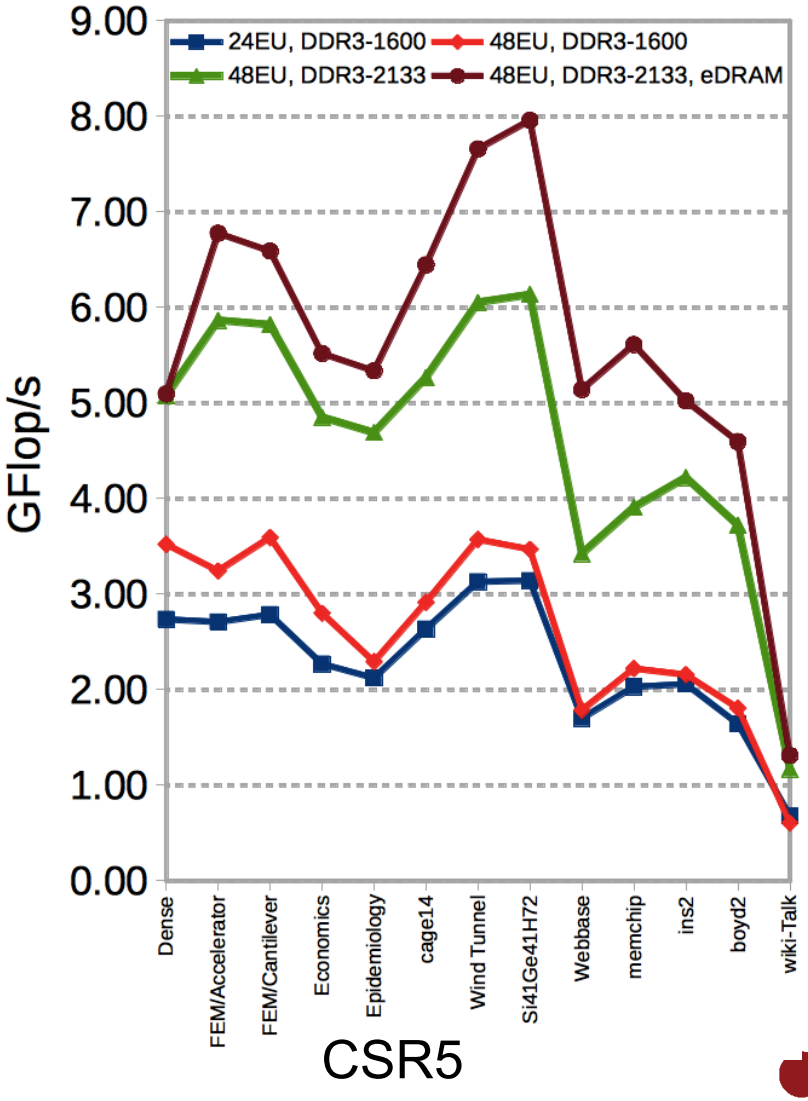
48eu, DDR3-2133



48eu, DDR3-2133 eDRAM

Scalability (algorithm-wise)





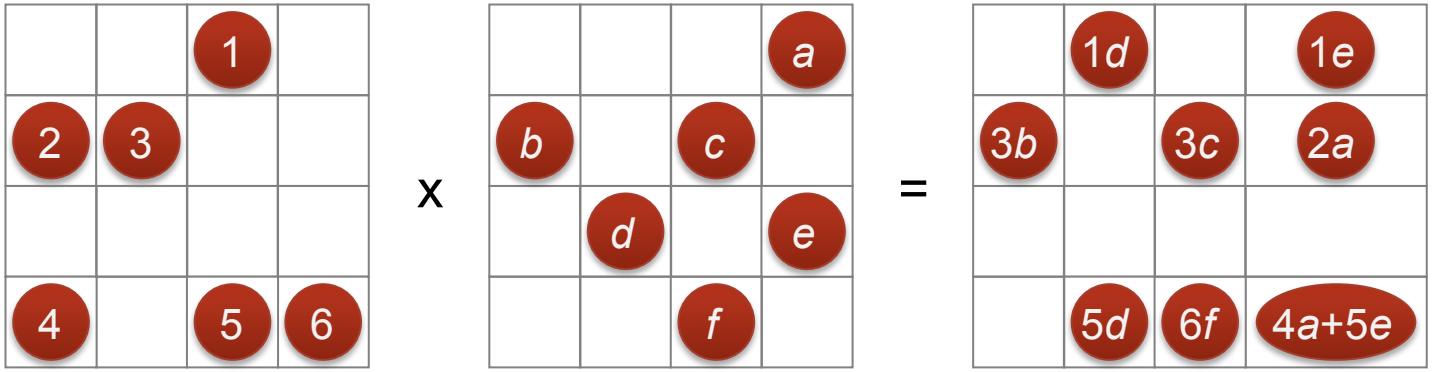
T3. New Opportunities

Case 1. Memory Re-allocation (SpGEMM)

Sparse Matrix-Matrix Multiplication

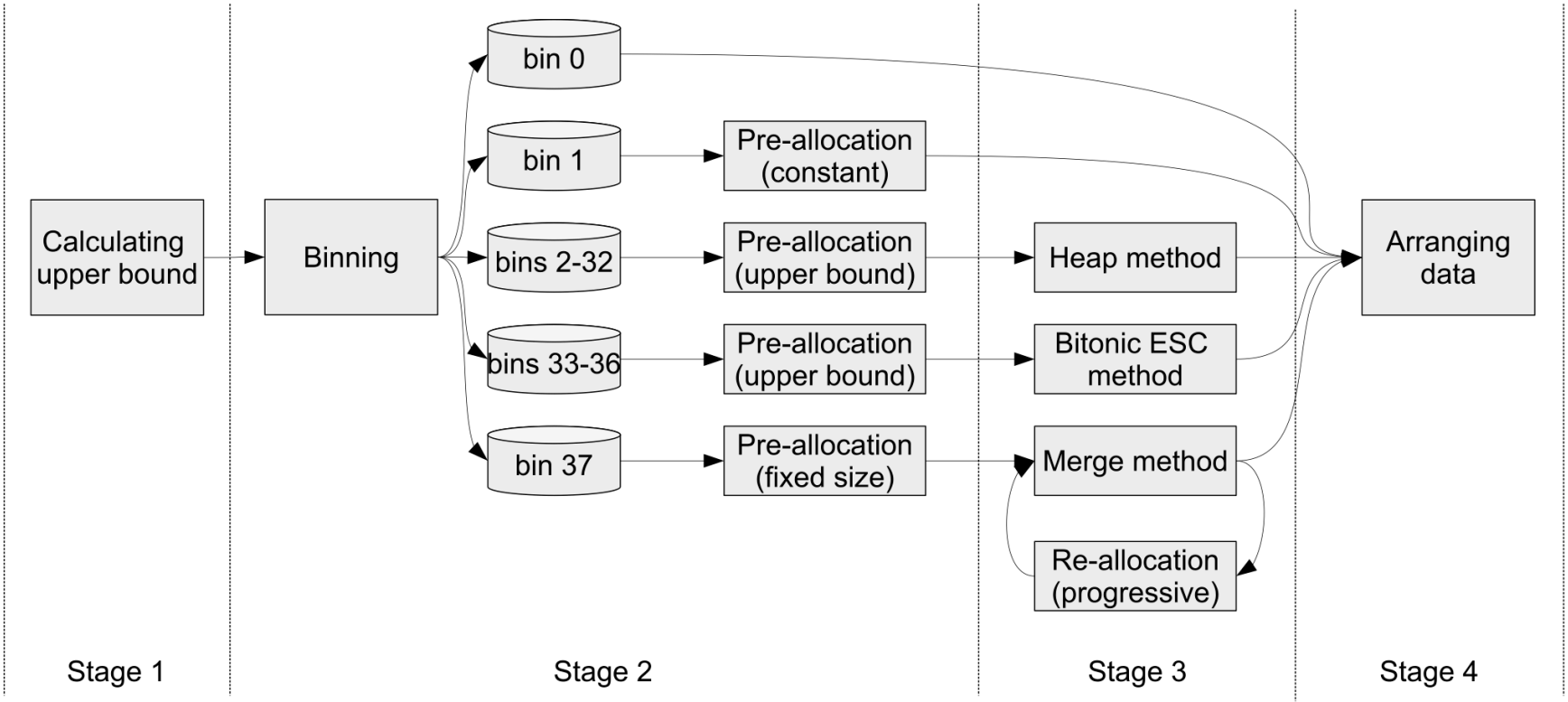
- Example: $C = AB$

Multiply a sparse matrix A by a sparse matrix B, obtain a sparse matrix C.



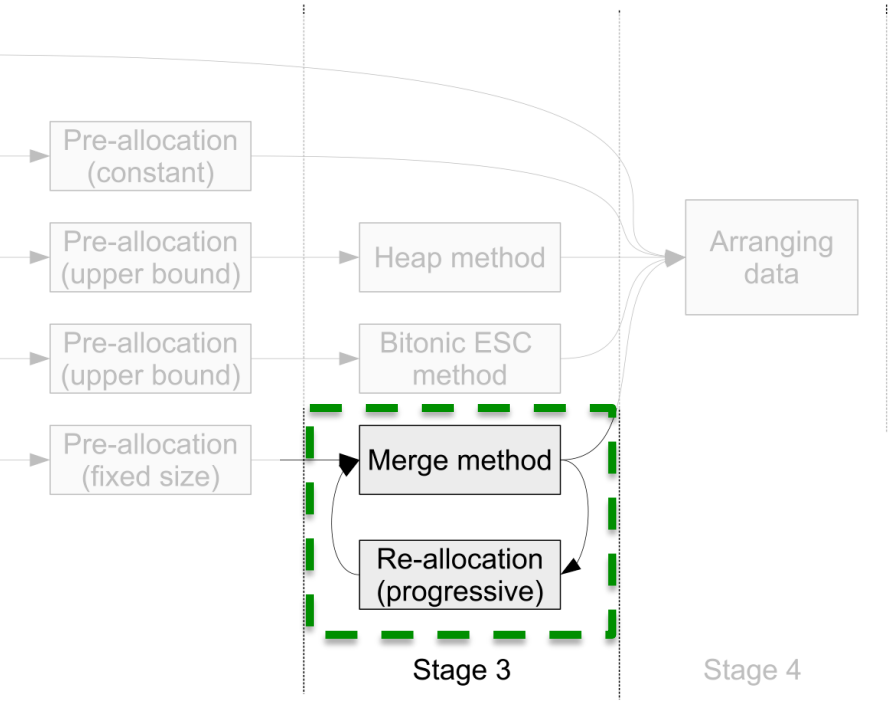
A	B	C
(4x4)	(4x4)	(4x4)
$nnzA = 6$	$nnzB = 6$	$nnzC = 8$

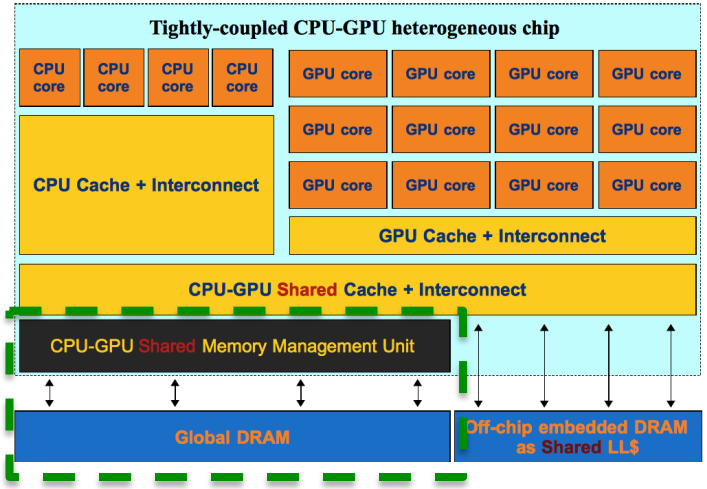
SpGEMM Framework

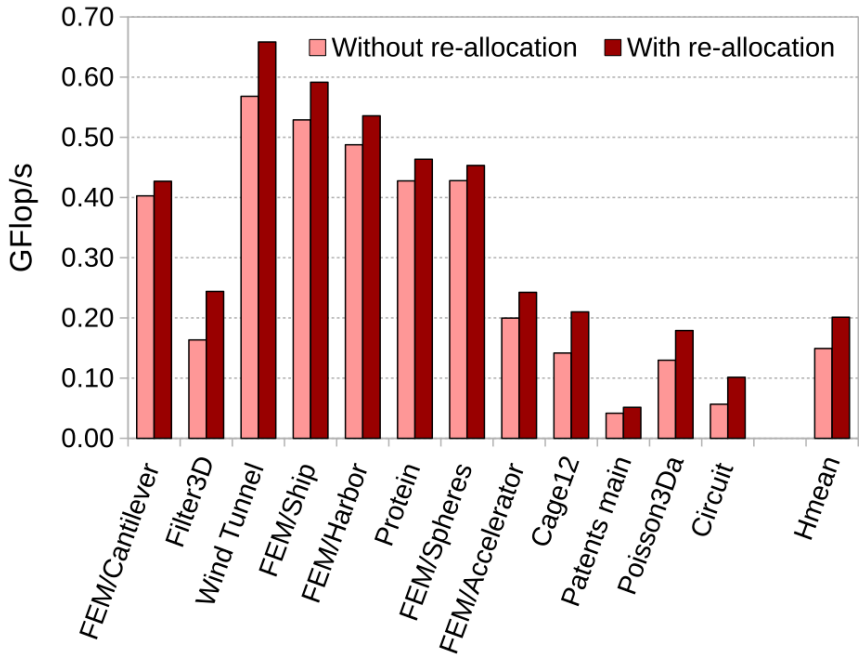


W. Liu and B. Vinter. **A Framework for General Sparse Matrix -Matrix Multiplication on GPUs and Heterogeneous Processors**. *Journal of Parallel and Distributed Computing. 2015.* (extended from IPDPS *'14*)

Memory Re-allocation on Hetero-chips

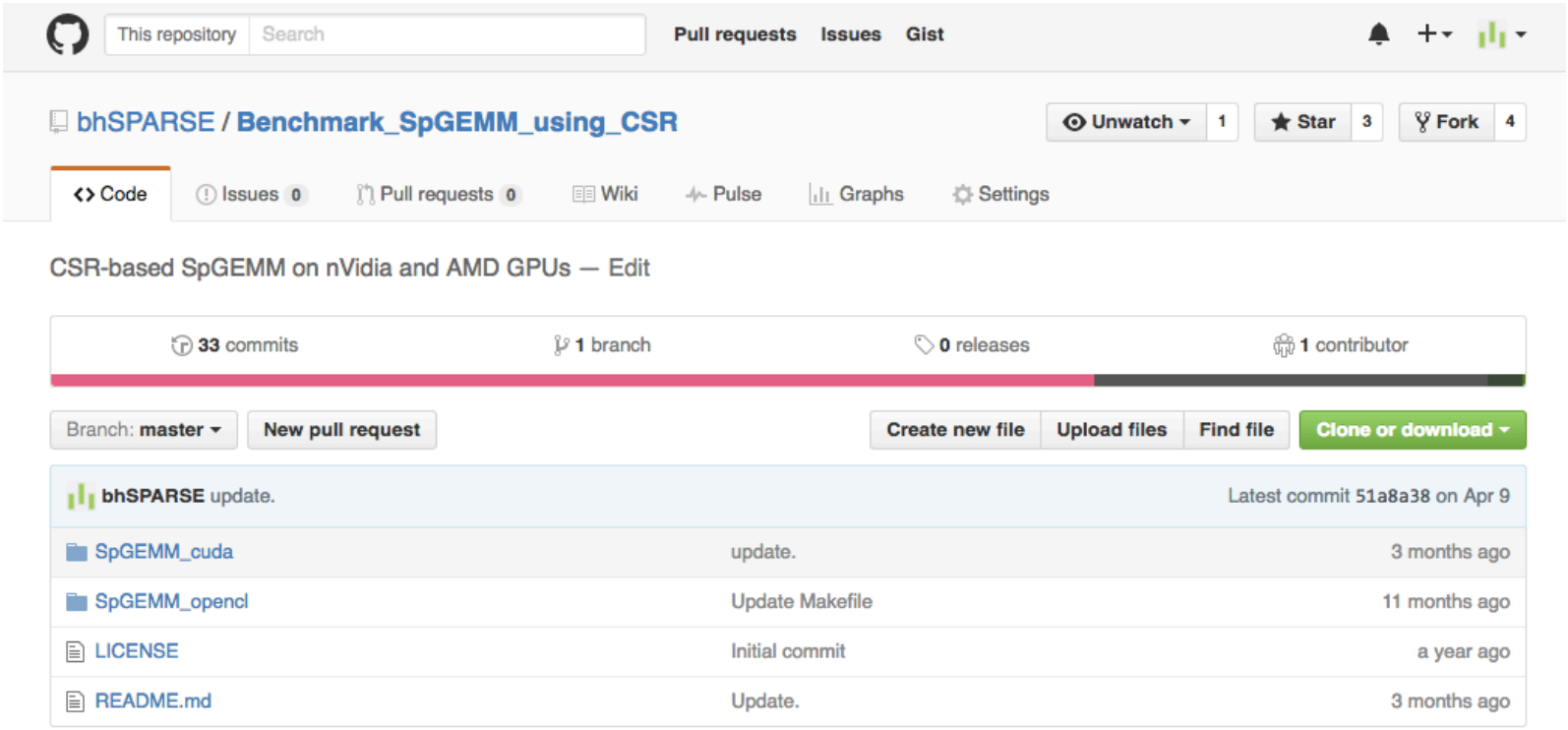






On an AMD A10-7850k APU, obtain on average 1.2x and up to 1.8x speedups over pure GPU.

Source Code at Github



https://github.com/bhSPARSE/Benchmark_SpGEMM_using_CSR

- **Also available in AMD's clSPARSE Library**
  https://github.com/clMathLibraries/clSPARSE

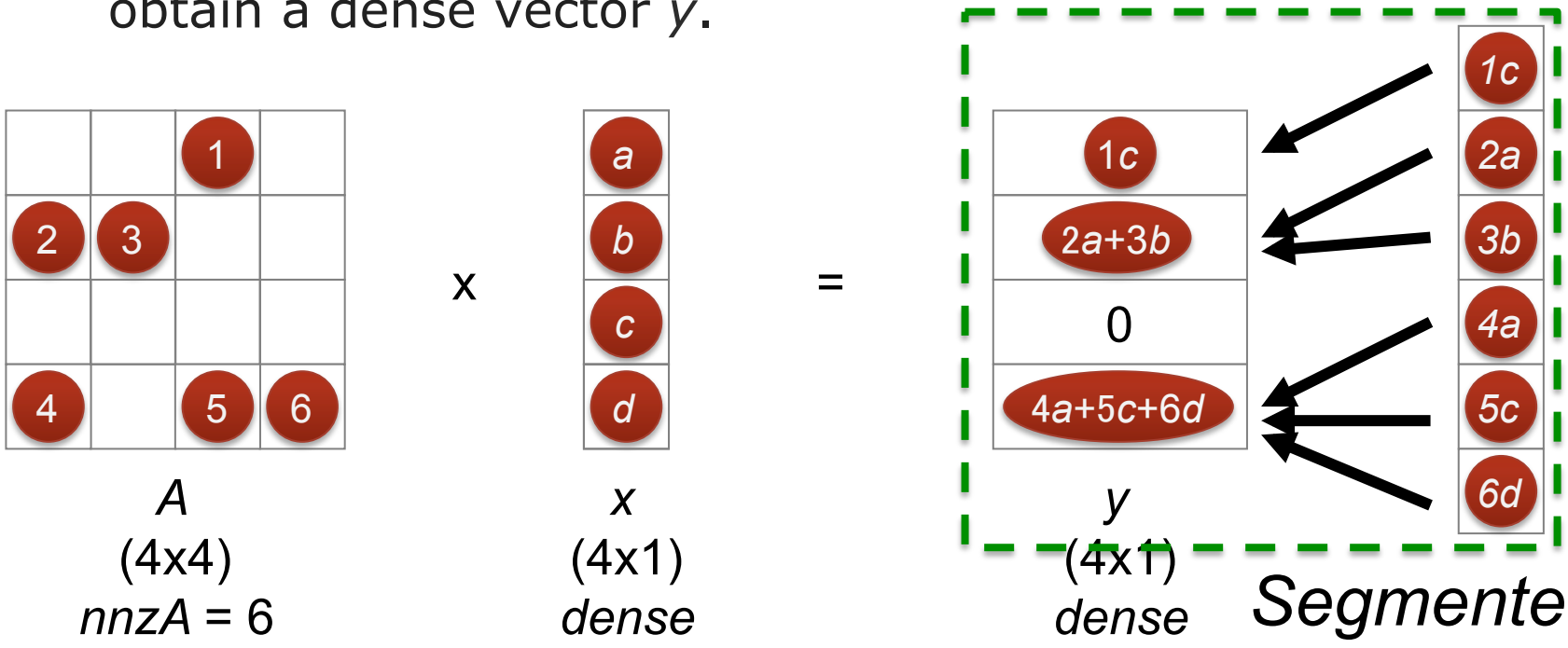
T3. New Opportunities

Case 2. Speculative Execution (SpMV)

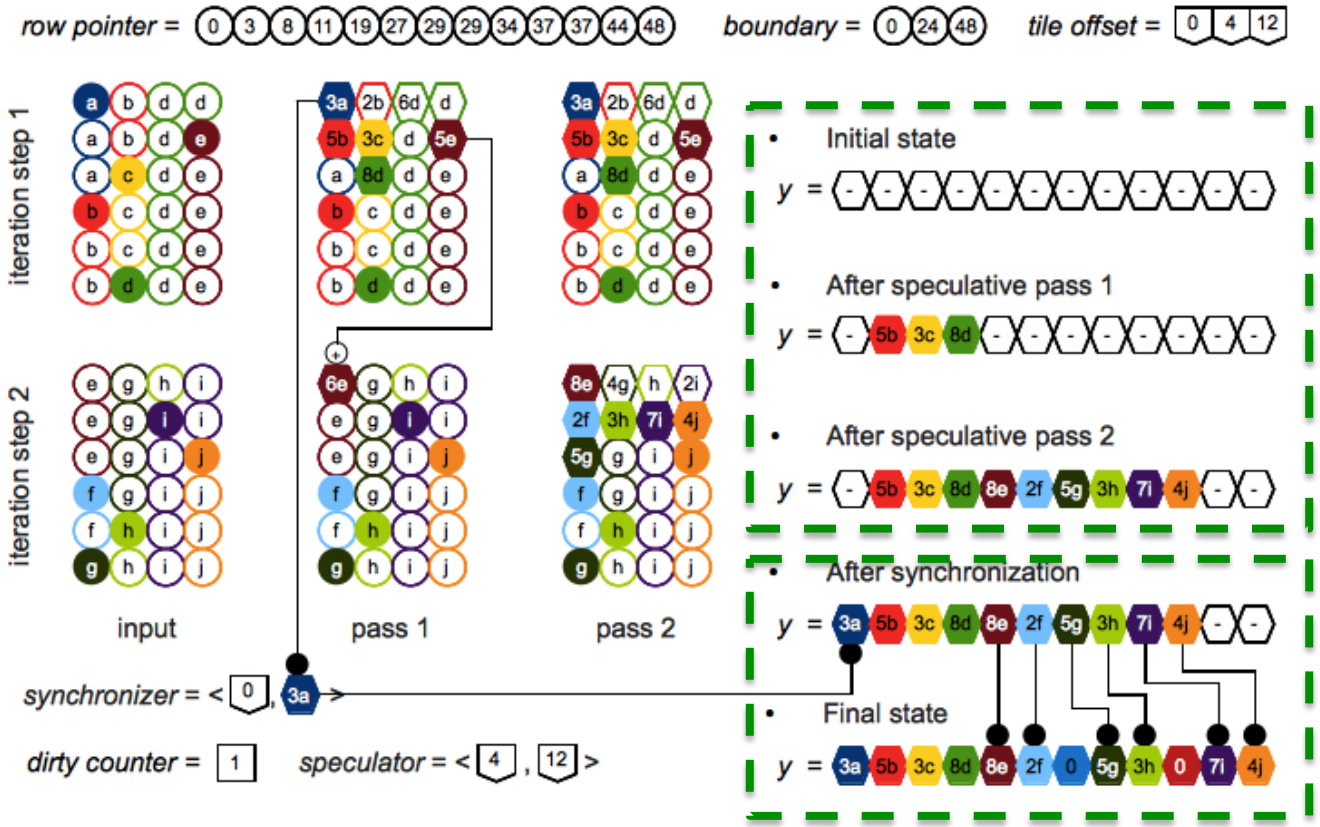
Sparse Matrix-Vector Multiplication

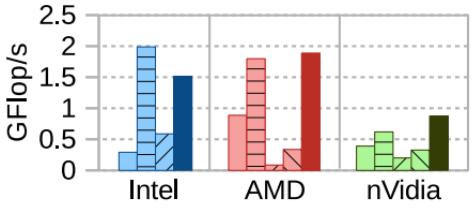
- Example: $y = Ax$

Multiply a sparse matrix A by a dense vector x, obtain a dense vector y.

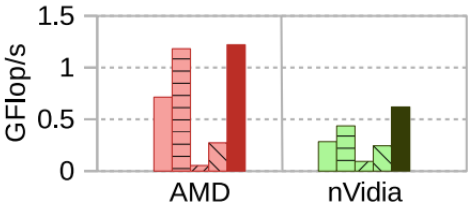


SpMV using Speculative Seg. Sum





single precision

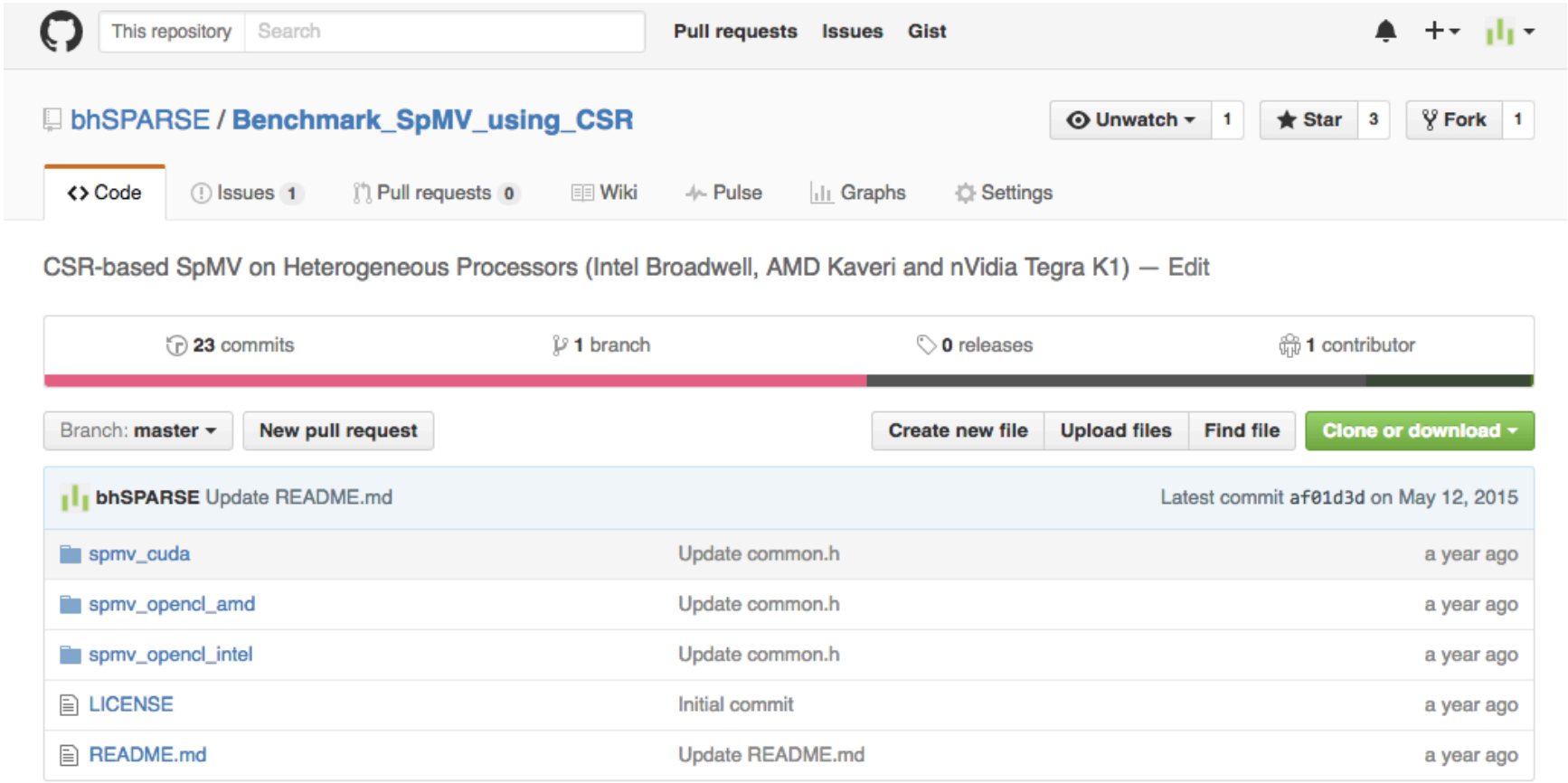


double precision

Intel i3-5010u
AMD A10-7850k
nVidia Tegra K1

W. Liu and B. Vinter. **Speculative Segmented Sum for Sparse Matrix-Vector Multiplication on Heterogeneous Processors**. *Parallel Computing. 2015.*

Source Code at Github



https://github.com/bhSPARSE/Benchmark_SpMV_using_CSR

Conclusion

- Heterogeneous processors with eDRAM can offer promising performance: much faster than processors without eDRAM, and can be faster than high-end server processors.
- Even for `small' integrated GPUs, scalable algorithms are still needed. We will see more GPUs cores in upcoming heterogeneous processors.
- Heterogeneous processors need newly designed algorithms to utilize deeper cache levels and shared units.

T k u !

0	4	8	9

A y Q s n s ?

0	2	4	7	11	12	13