

An Empirical Study of Sparse BLAS on Emerging Heterogeneous Processors

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

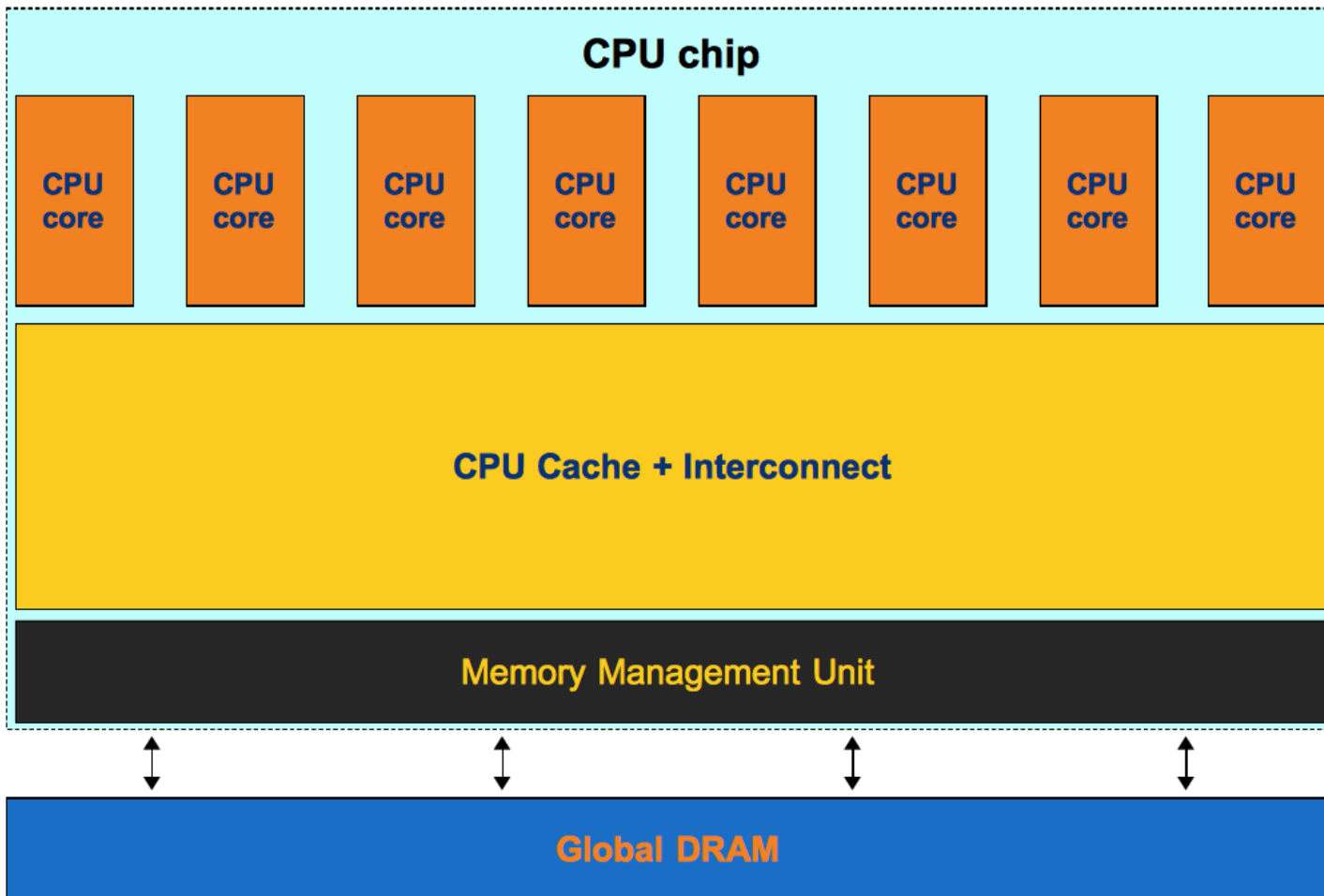
PMAA '16, Bordeaux, France, 8 July 2016

Background:

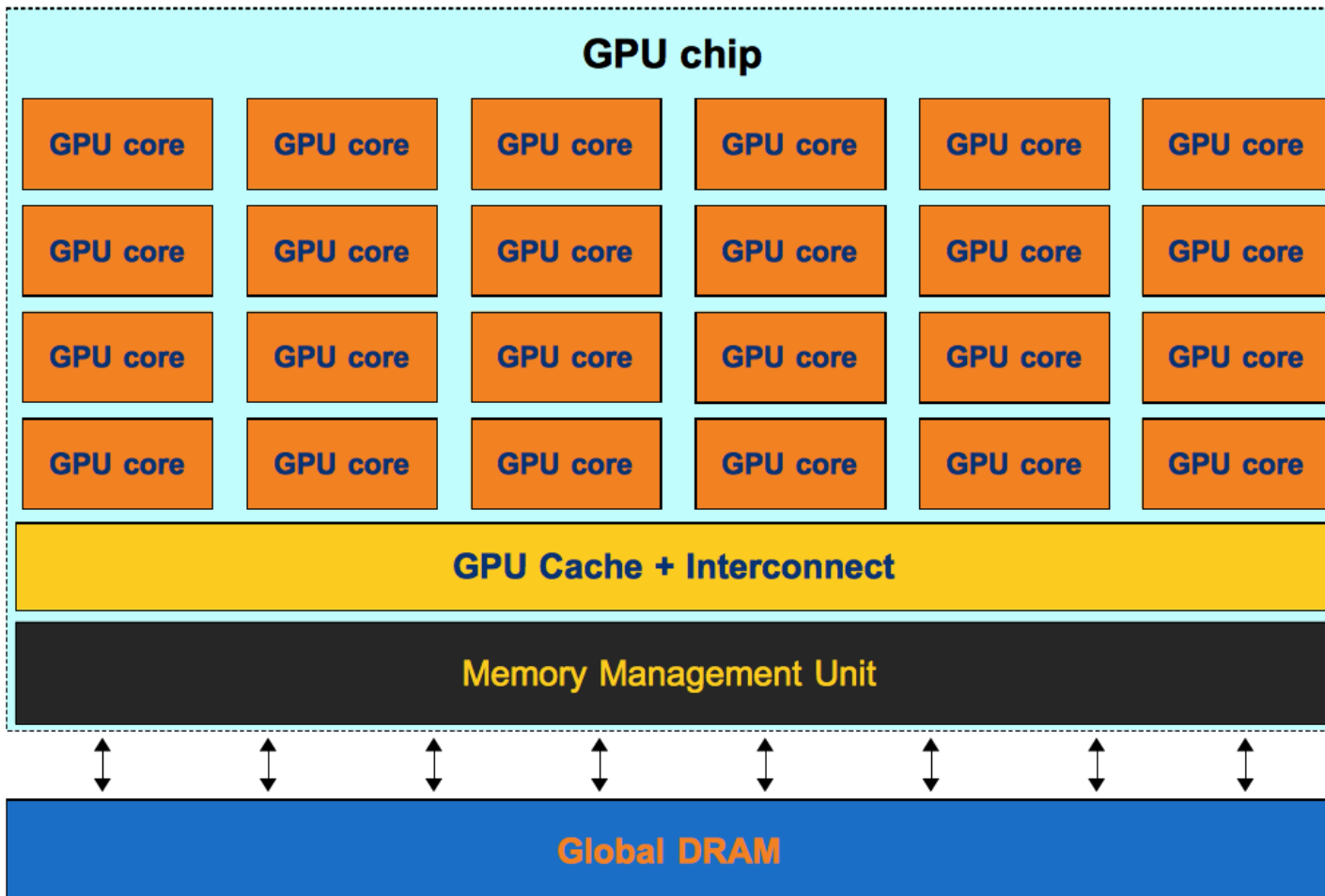
Emerging Heterogeneous Processors



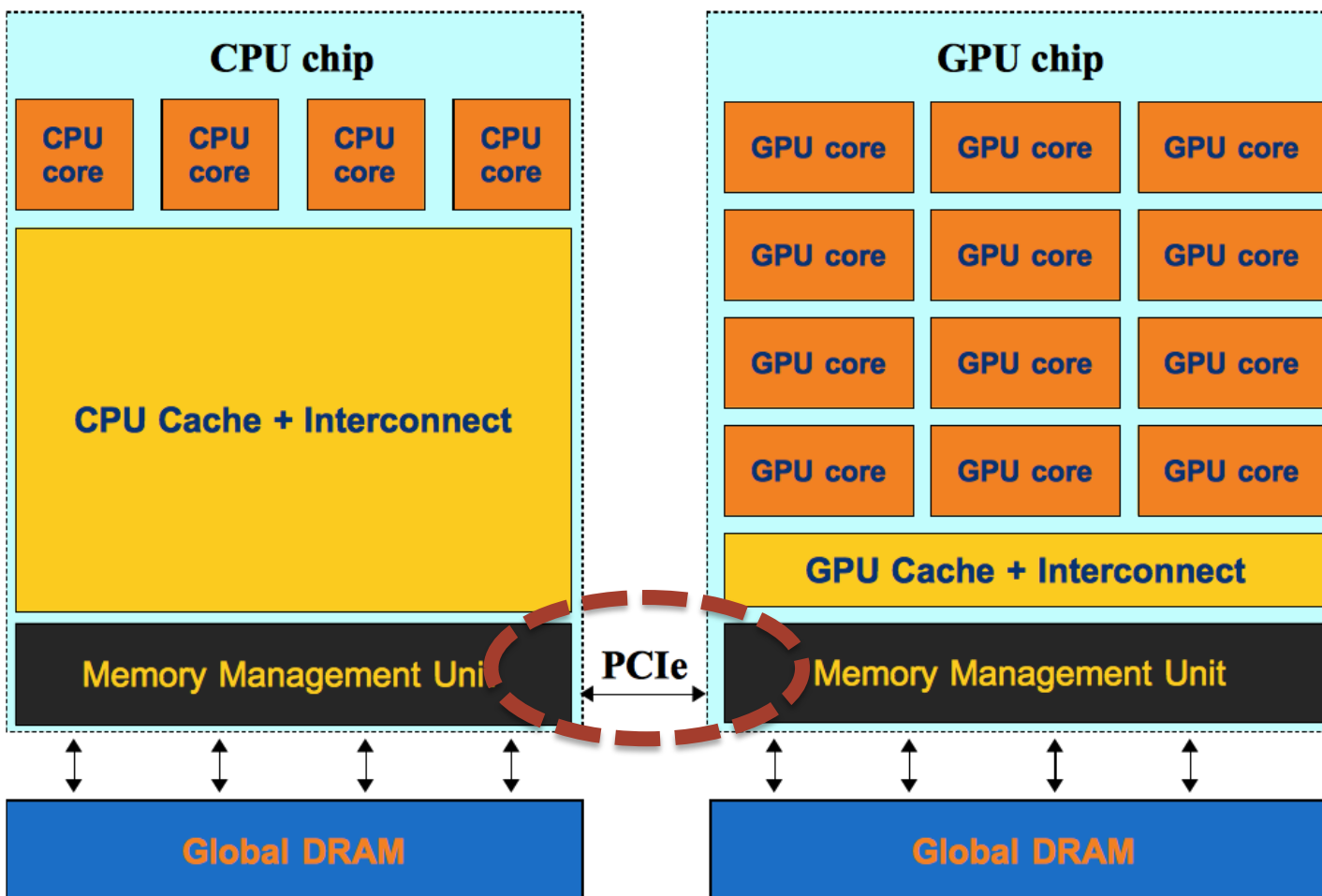
CPU Chip



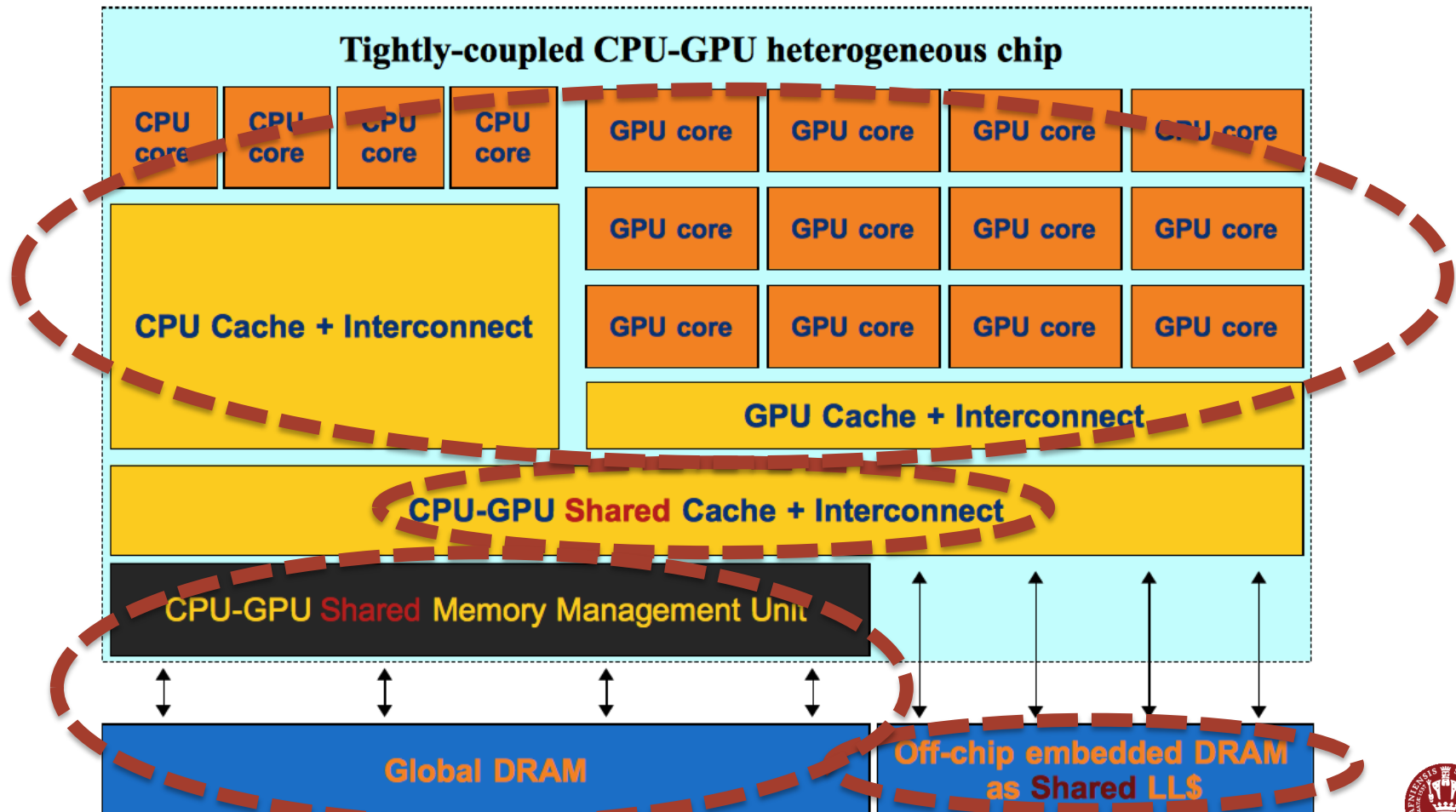
GPU Chip



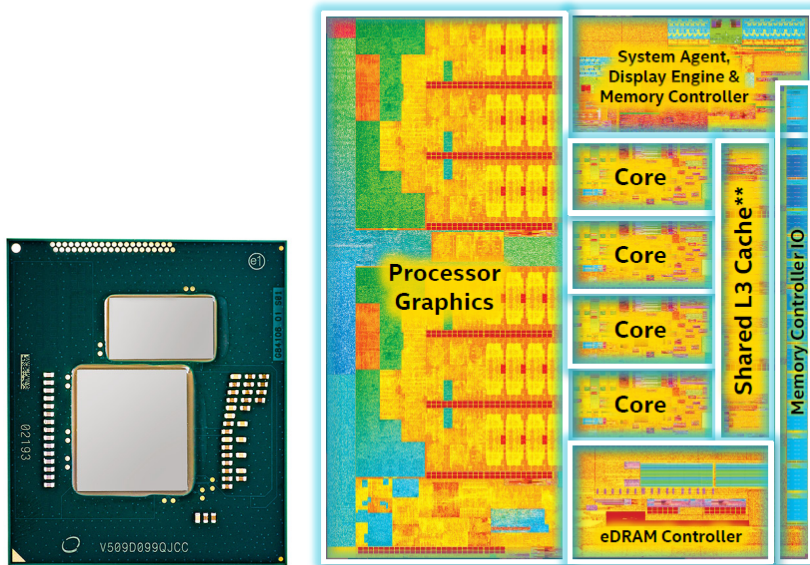
Loosely-coupled CPU-GPU Platform



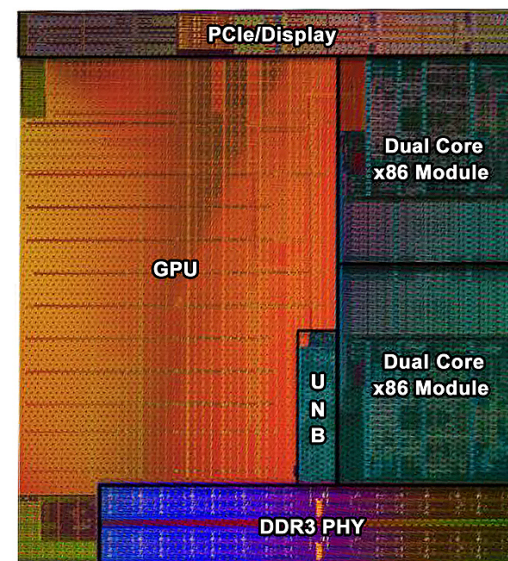
Tightly-coupled CPU-GPU Chip



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?

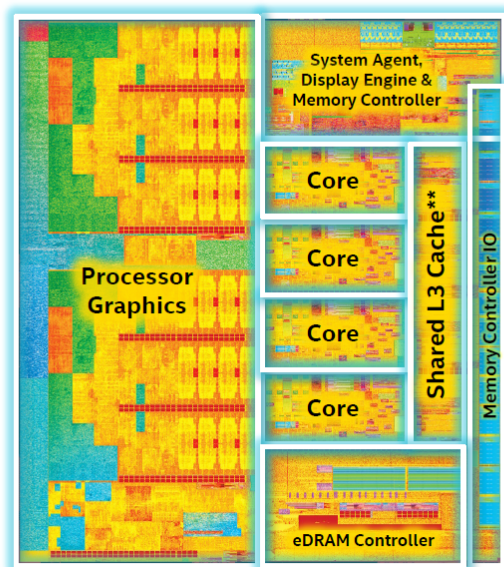
T3. Any opportunities for new algorithms?



T1. Effectiveness of eDRAM



Testbed



Intel Core i7-5775c (Broadwell)

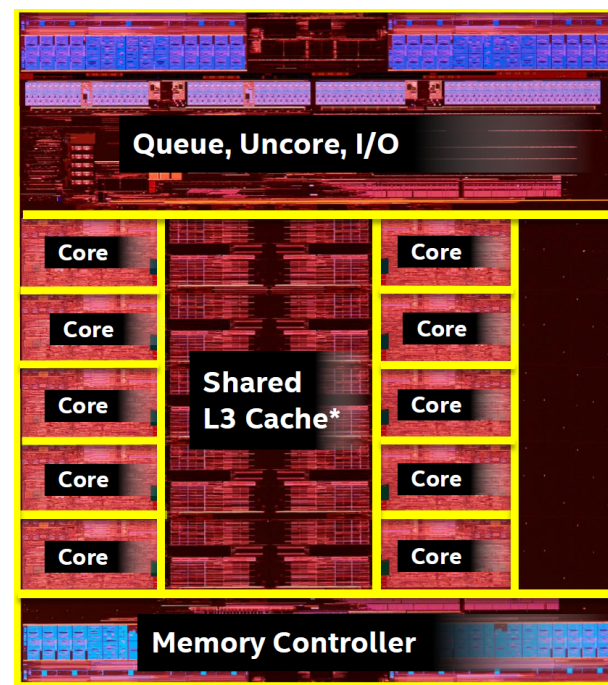
CPU: 4 cores @ 3.3-3.7 GHz, 211.2 Gflop/s

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

SpTRANS
SpMV
SpTRSV
SpGEMM

G2. Sparse Direct Solvers

Cholesky
LU
QR

G3. Benchmark

HPCG
(High
Performance
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)

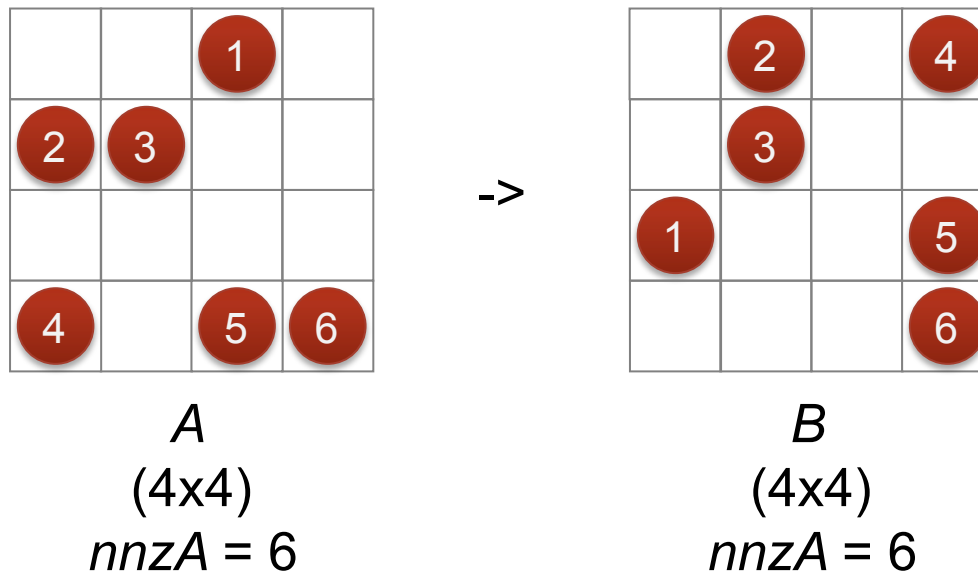
Matrix selected from the University of Florida Sparse Matrix Collection:
square, $2M \leq \#nnz \leq 200M$, groups having no less than 4 matrices.



Sparse Matrix Transposition

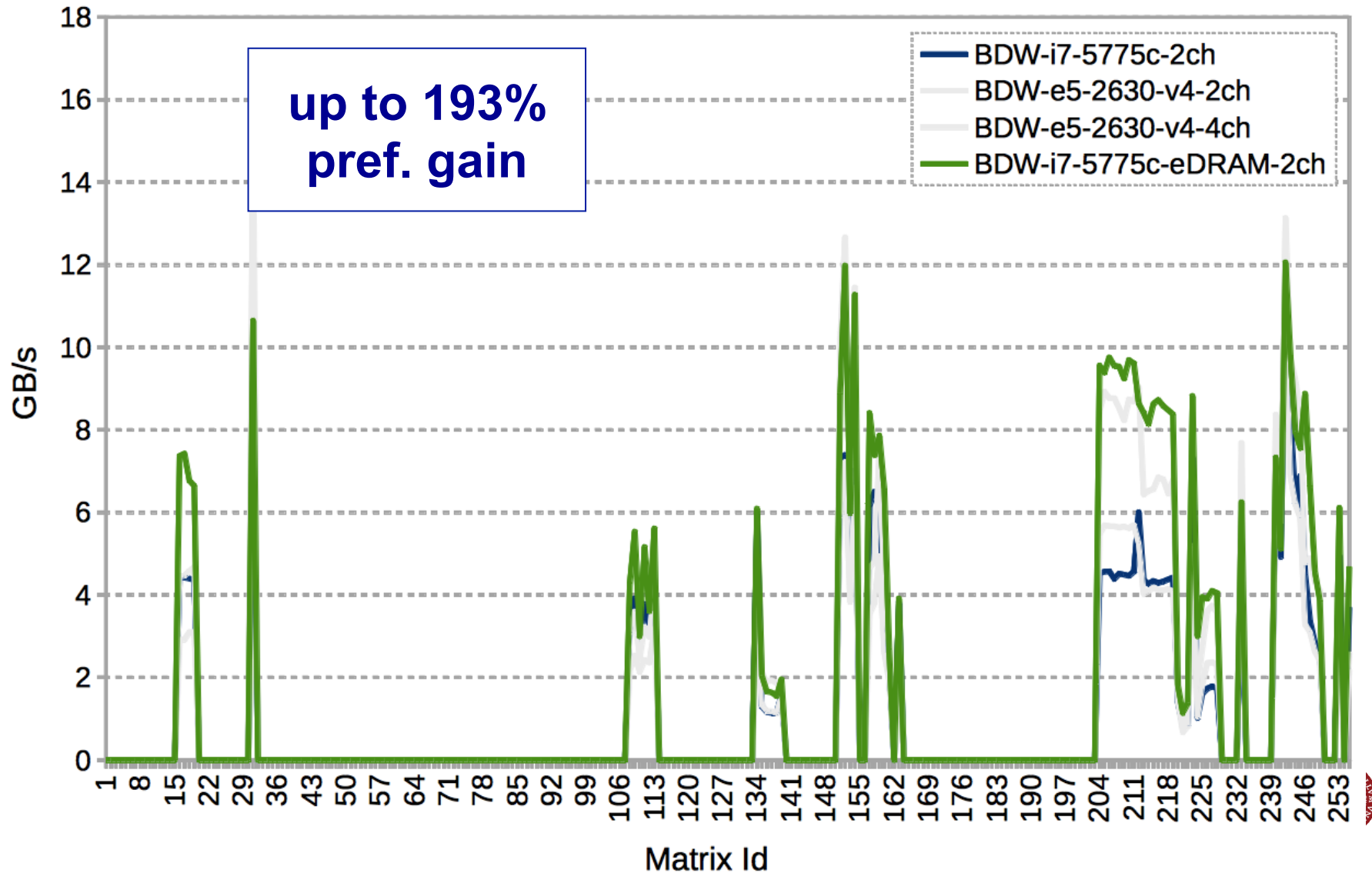
- Example: $B = \text{sptrans}(A)$

Transpose a sparse matrix A (in CSR) to a sparse matrix B (in CSR). Format conversion CSR \leftrightarrow CSC shares the same operation.

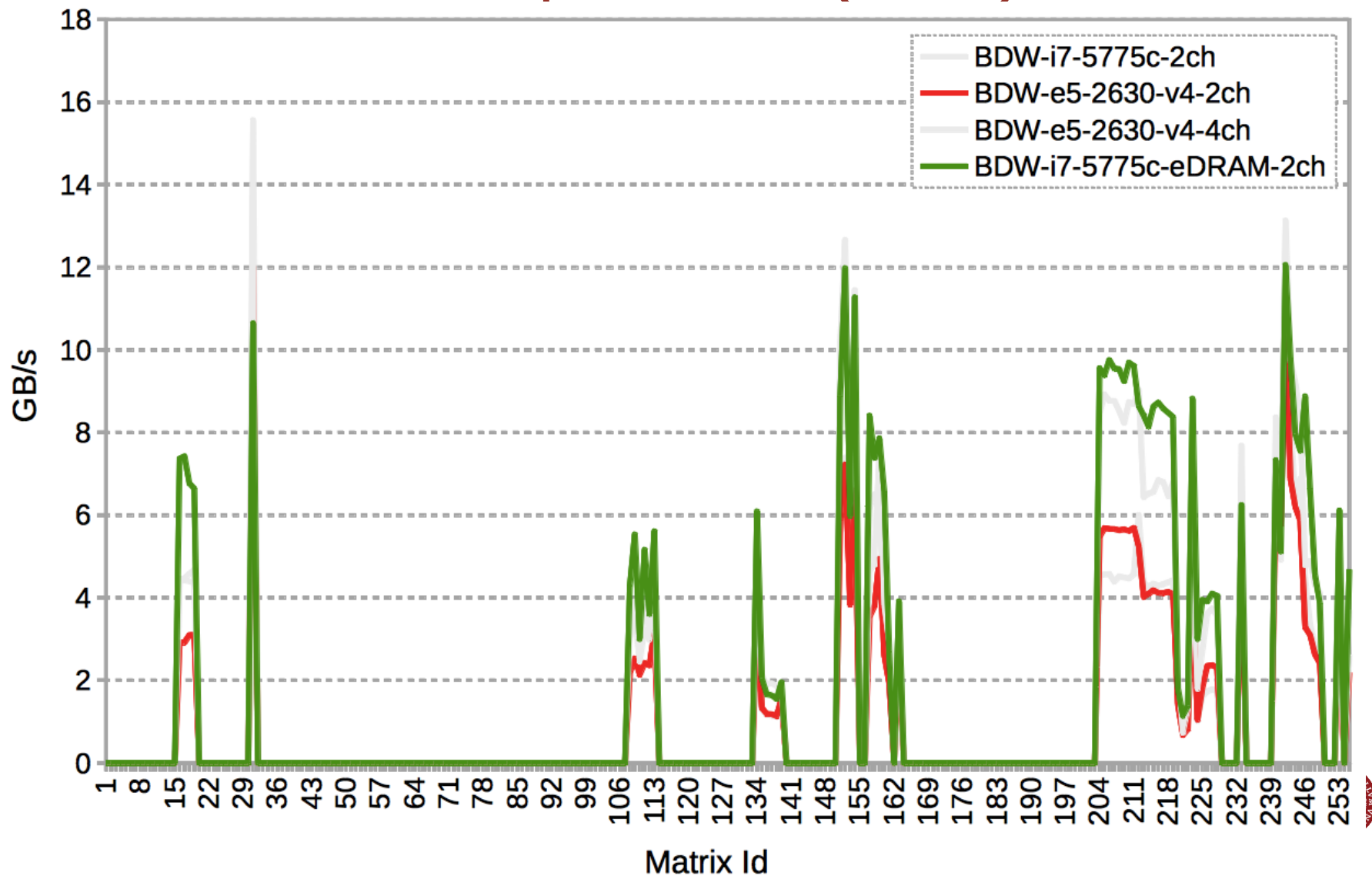


H. Wang, W. Liu, K. Hou, and W. Feng. **Parallel Transposition of Sparse Data Structures**. *ICS '16*.

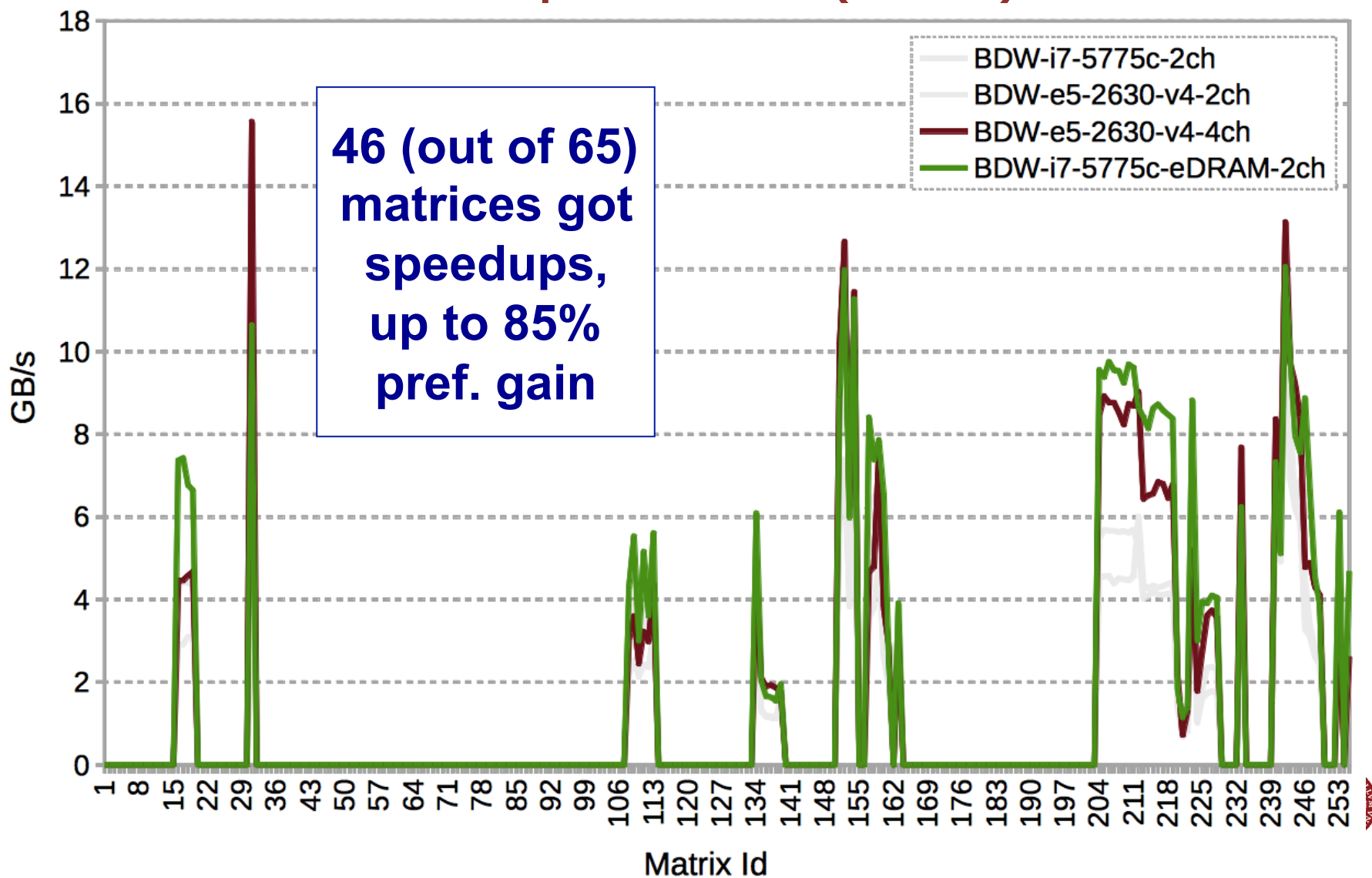
CSR-based SpTRANS (MKL)



CSR-based SpTRANS (MKL)



CSR-based SpTRANS (MKL)



Sparse Matrix-Vector Multiplication

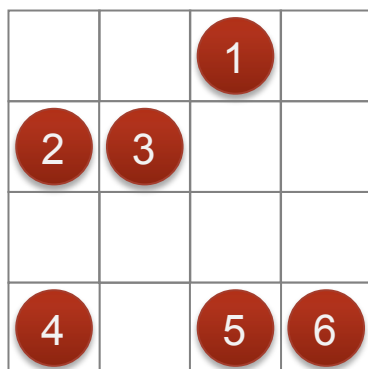
- Example: $y = Ax$

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

<table border="1" style="border-collapse: collapse; width: 100px; height: 100px;"> <tr><td></td><td></td><td style="text-align: center;">1</td><td></td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">3</td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td style="text-align: center;">4</td><td></td><td style="text-align: center;">5</td><td style="text-align: center;">6</td></tr> </table>			1		2	3							4		5	6	×	<table style="border-collapse: collapse; width: 100px; height: 100px;"> <tr><td style="text-align: center;">a</td></tr> <tr><td style="text-align: center;">b</td></tr> <tr><td style="text-align: center;">c</td></tr> <tr><td style="text-align: center;">d</td></tr> </table>	a	b	c	d	=	<table border="1" style="border-collapse: collapse; width: 100px; height: 100px;"> <tr><td style="text-align: center;">1c</td></tr> <tr><td style="text-align: center;">2a+3b</td></tr> <tr><td style="text-align: center;">0</td></tr> <tr><td style="text-align: center;">4a+5c+6d</td></tr> </table>	1c	2a+3b	0	4a+5c+6d
		1																										
2	3																											
4		5	6																									
a																												
b																												
c																												
d																												
1c																												
2a+3b																												
0																												
4a+5c+6d																												
A (4×4) $nnzA = 6$		x (4×1) <i>dense</i>		y (4×1) <i>dense</i>																								

CSR-based SpMV

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



→ Core 0

→ Core 1

→ Core 2

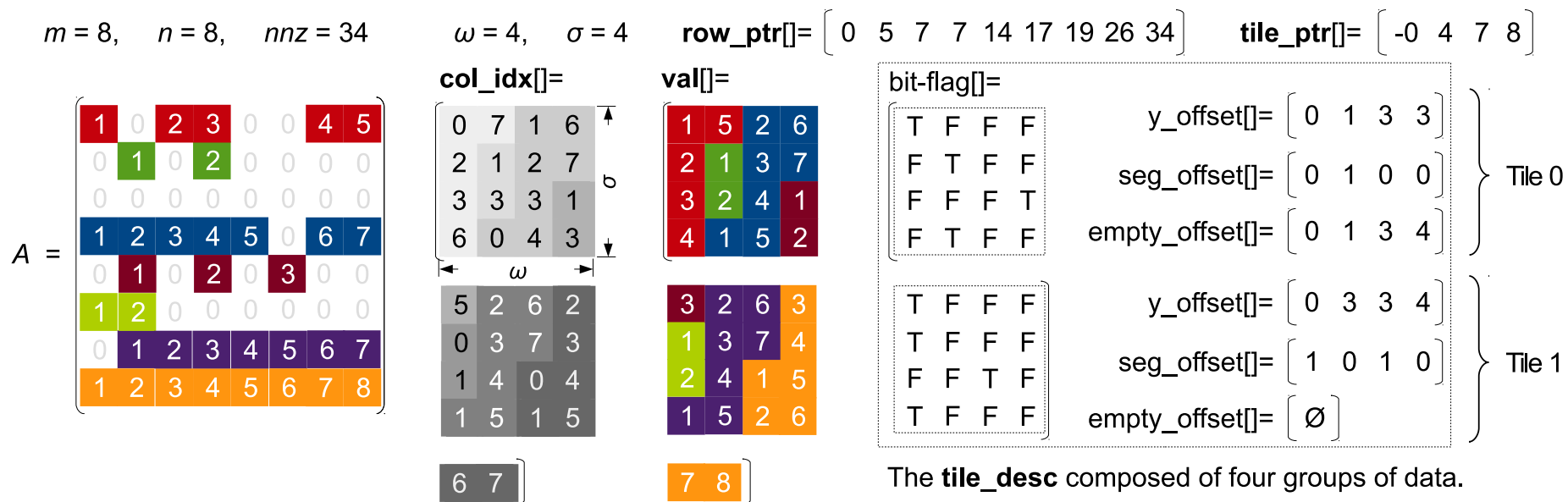
→ Core 3

**Possibly
imbalanced
computation**

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

Storage format – CSR5

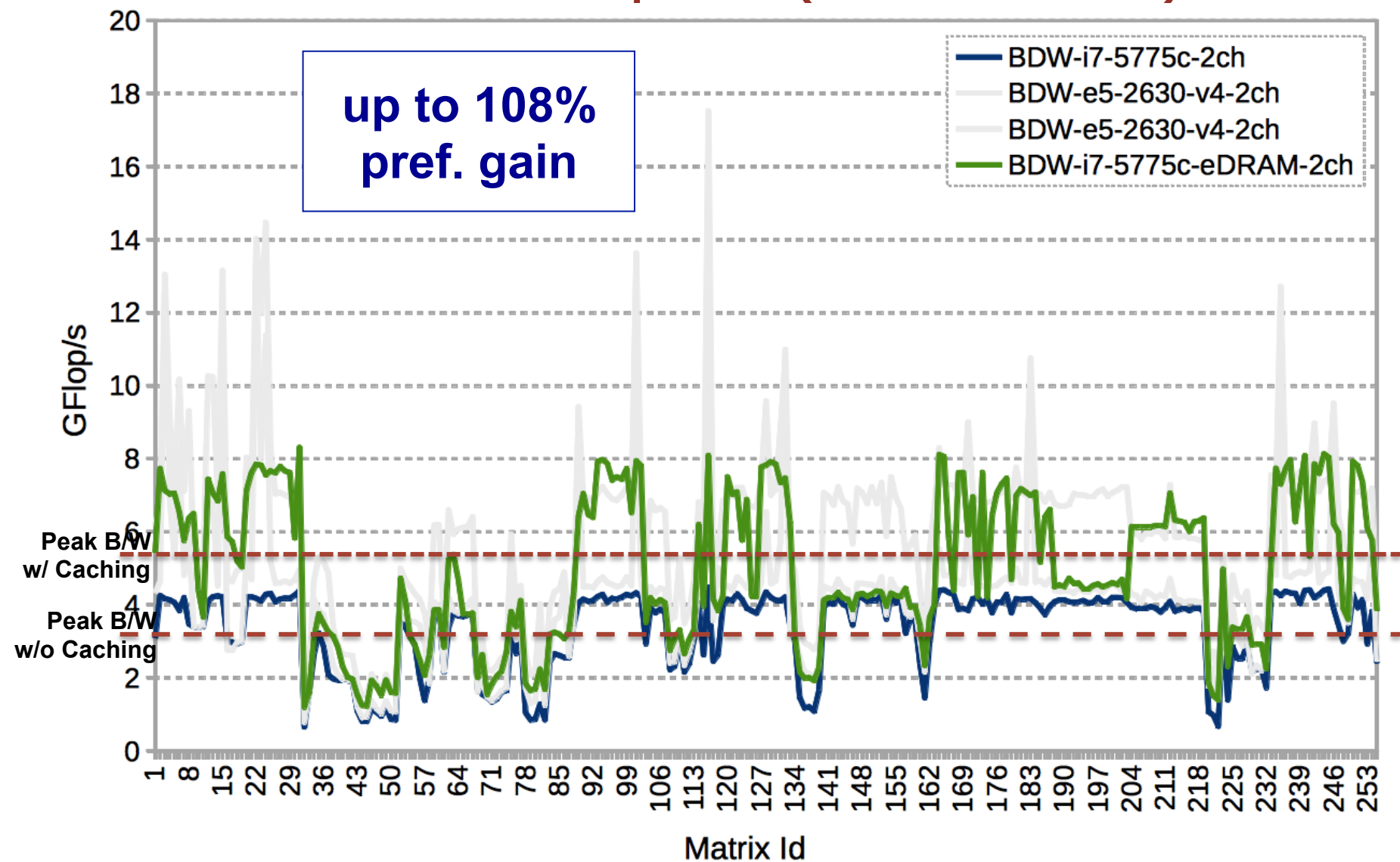
- Save CSR data in (transposed) tiles of the same size.



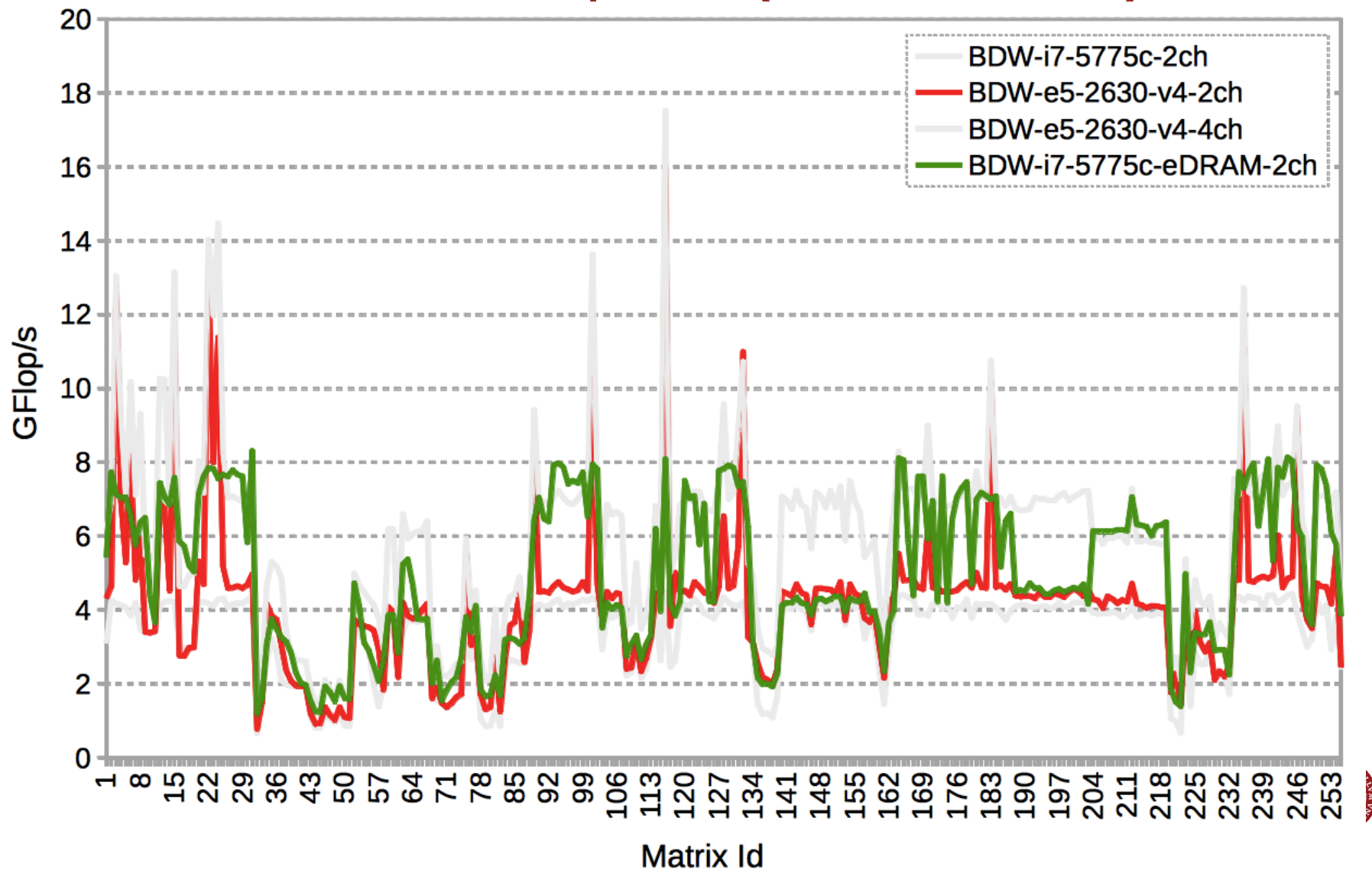
W. Liu and B. Vinter. **CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication.** *ICS '15.*



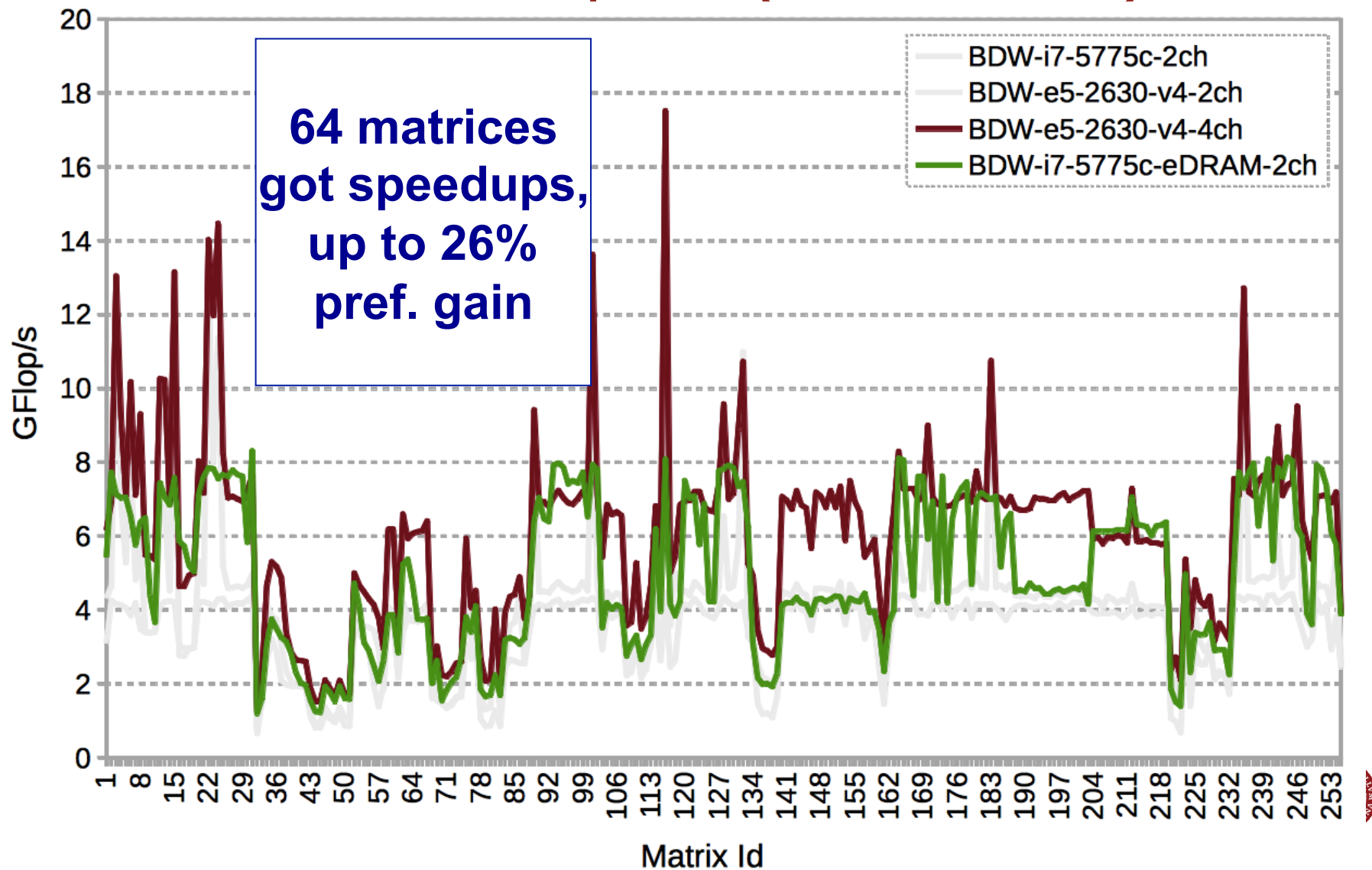
CSR5-based SpMV (bhSPARSE)



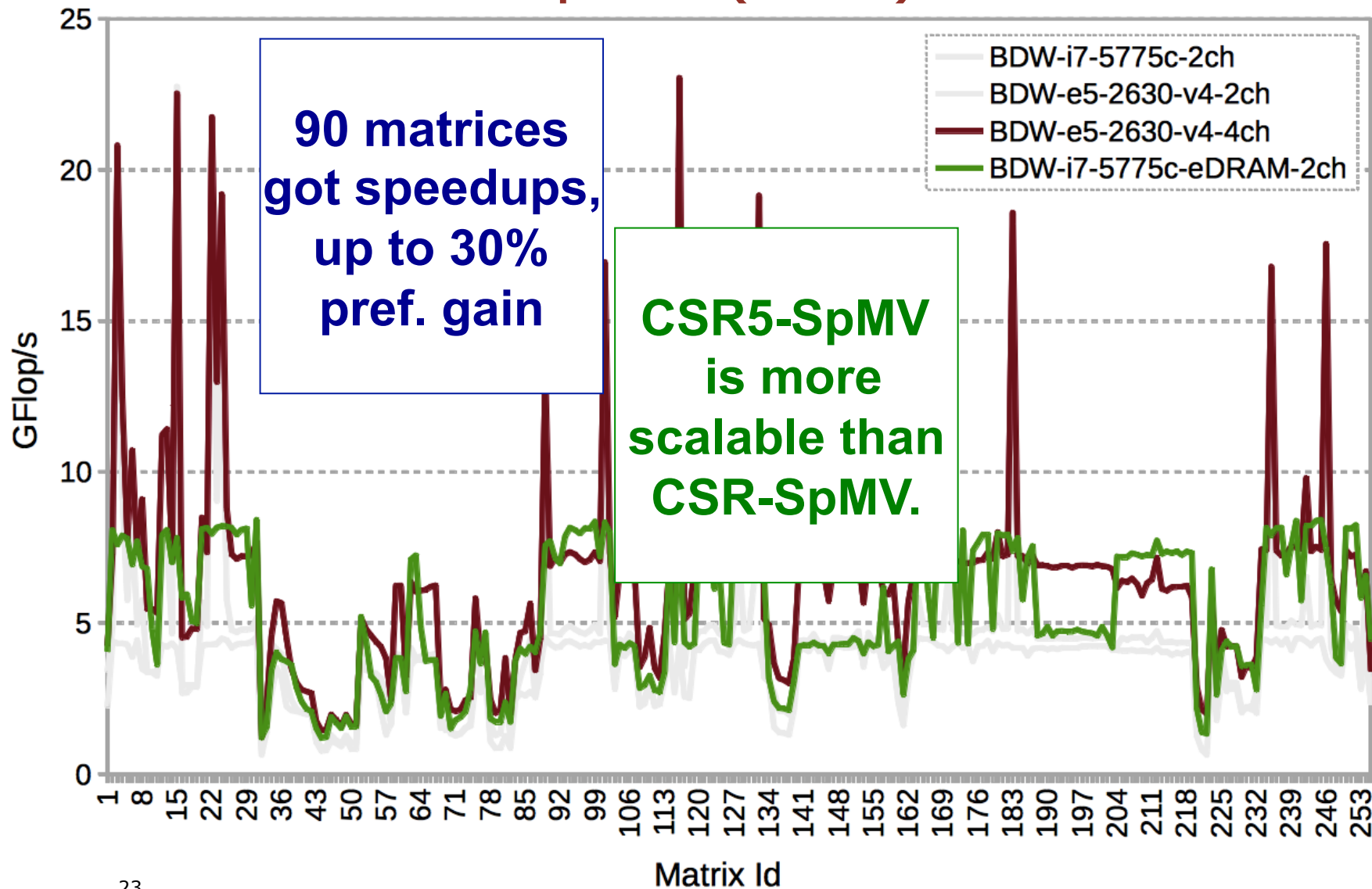
CSR5-based SpMV (bhSPARSE)



CSR5-based SpMV (bhSPARSE)



CSR-based SpMV (MKL)



Source Code of CSR5 at Github

This repository Search Explore Gist Blog Help bhSPARSE + -

bhSPARSE / Benchmark_SpMV_using_CSR5 Unwatch 4 Star 6 Fork 5

CSR5-based SpMV on CPUs, GPUs and Xeon Phi — Edit

11 commits 1 branch 0 releases 1 contributor

branch: master Benchmark_SpMV_using_CSR5 / +

Update README.md

bhSPARSE authored 22 days ago latest commit a39add4071

CSR5_avx2	Upload.	3 months ago
CSR5_cuda	Upload.	3 months ago
CSR5_opencv_amd	Upload.	3 months ago
CSR5_phi	Upload.	3 months ago
LICENSE	Initial commit	3 months ago
README.md	Update README.md	22 days ago

Code

Issues 0

Pull requests 0

Wiki

Pulse

Graphs

Settings

HTTPS clone URL

https://github.com/t

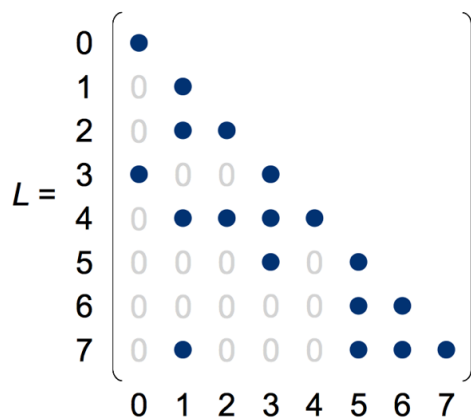
https://github.com/bhSPARSE/Benchmark_SpMV_using_CSR5



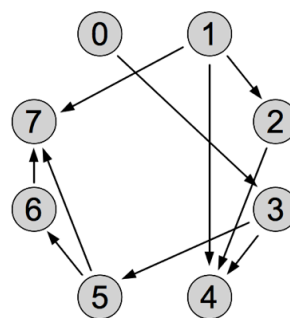
Sparse Triangular Solve

- Example: $Lx = b$

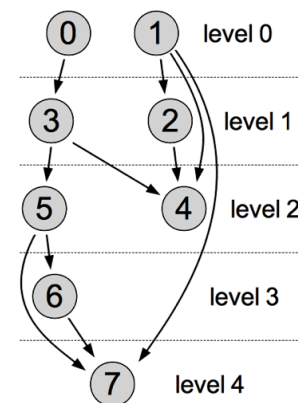
Compute a dense solution vector x from the sparse linear system, where L is a square lower triangular sparse matrix, and b is a dense vector.



L 's matrix form



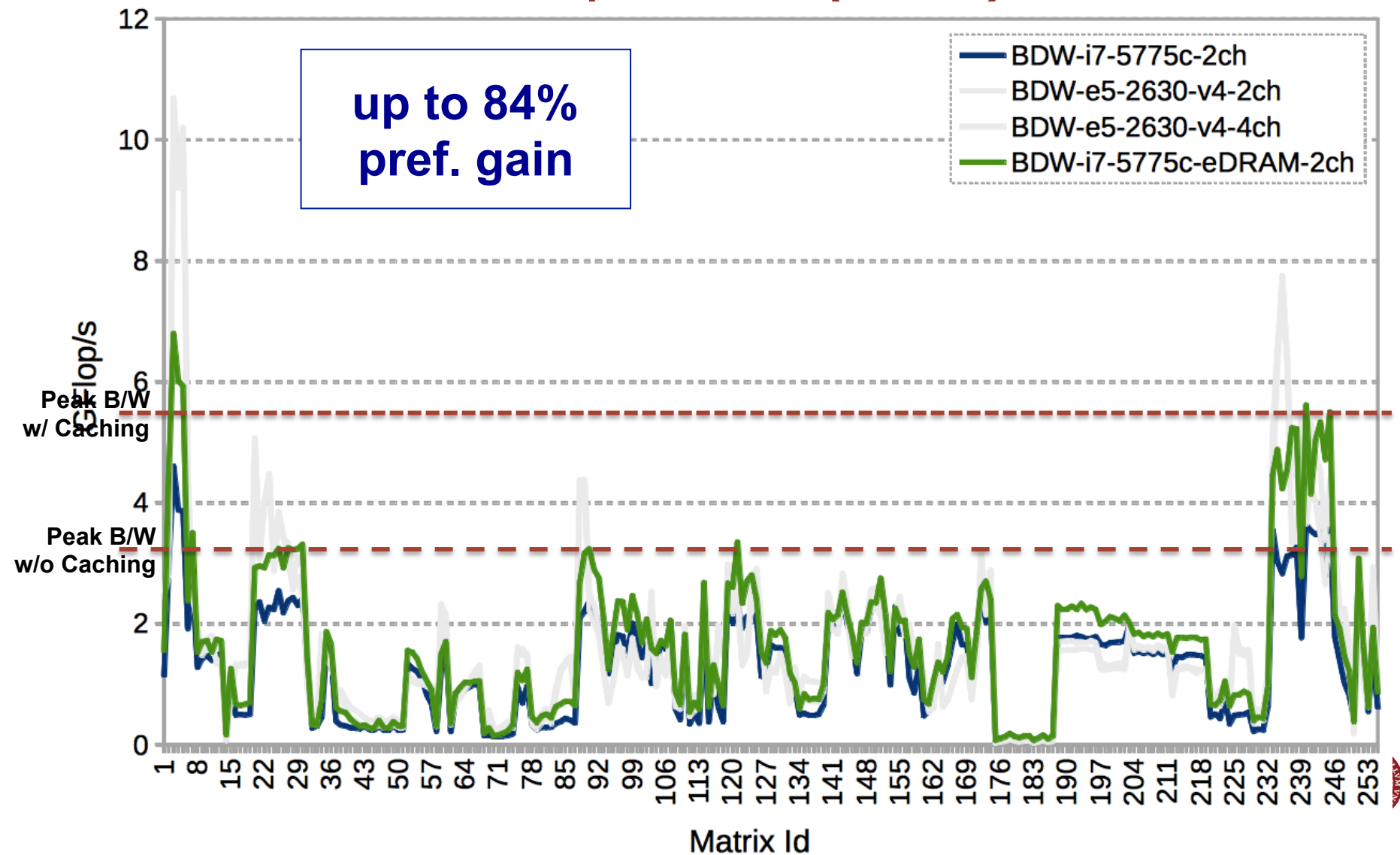
L 's graph form



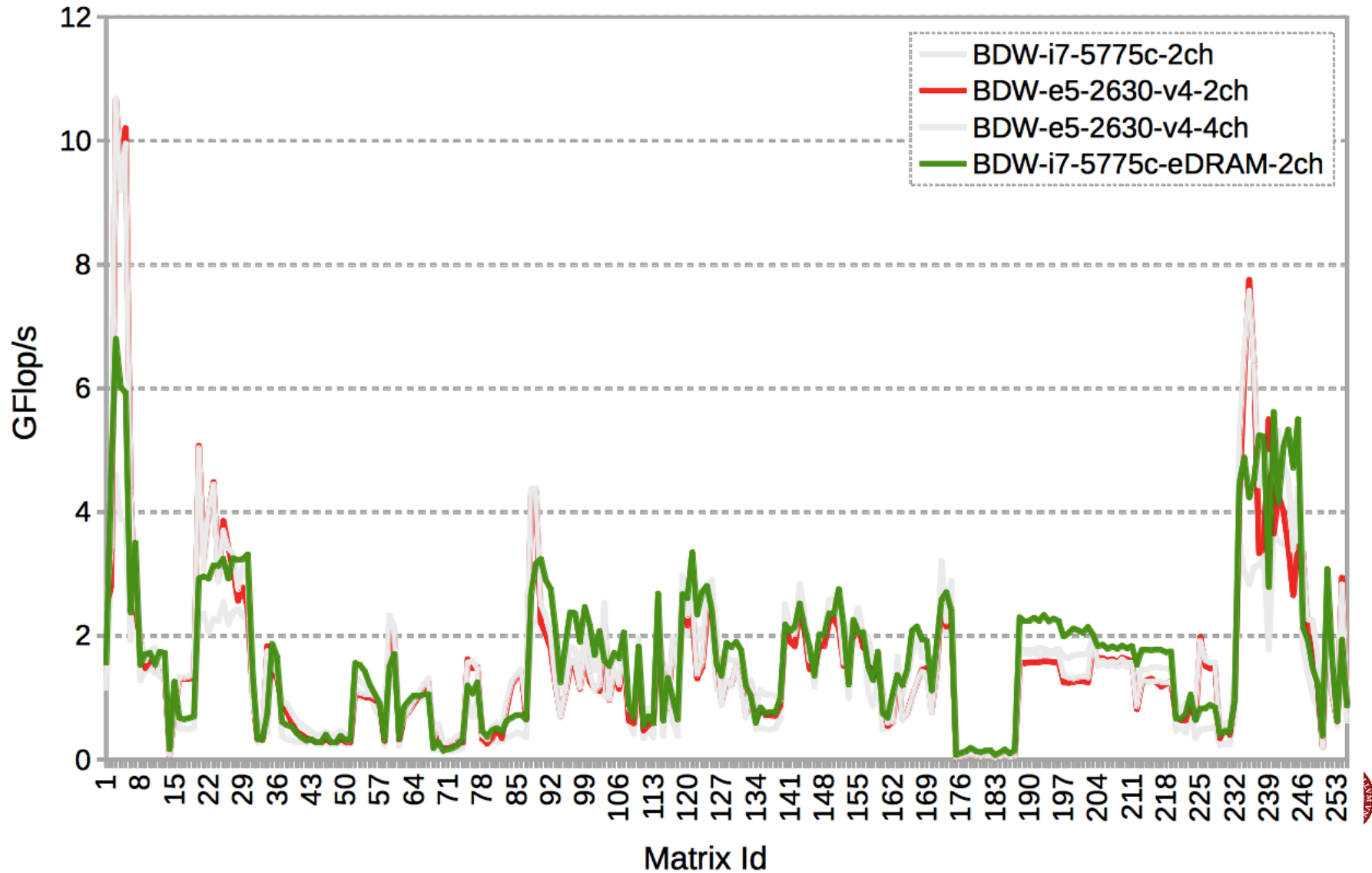
parallelizable
Level-sets

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

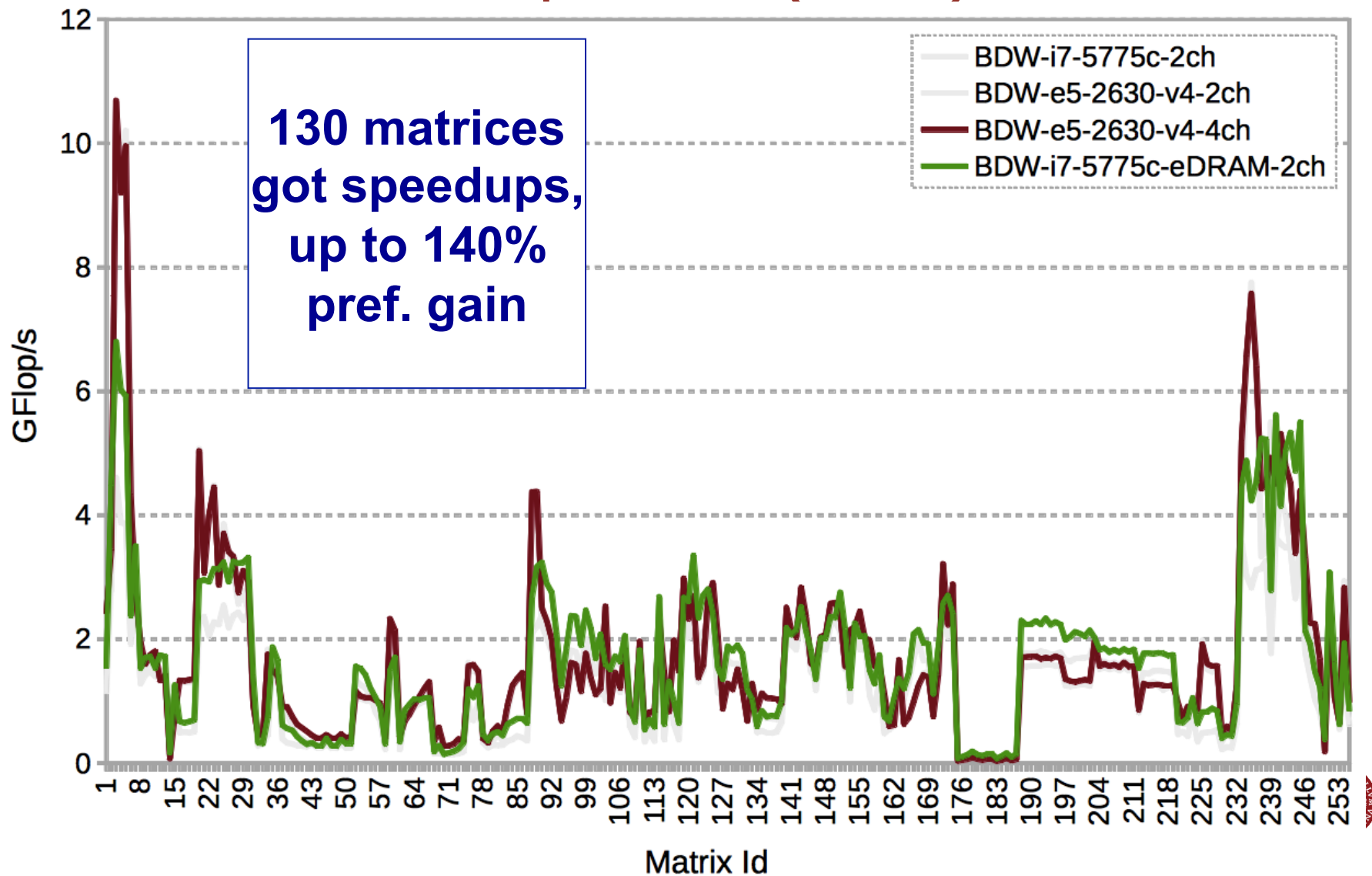
CSR-based SpTRSV (MKL)



CSR-based SpTRSV (MKL)



CSR-based SpTRSV (MKL)



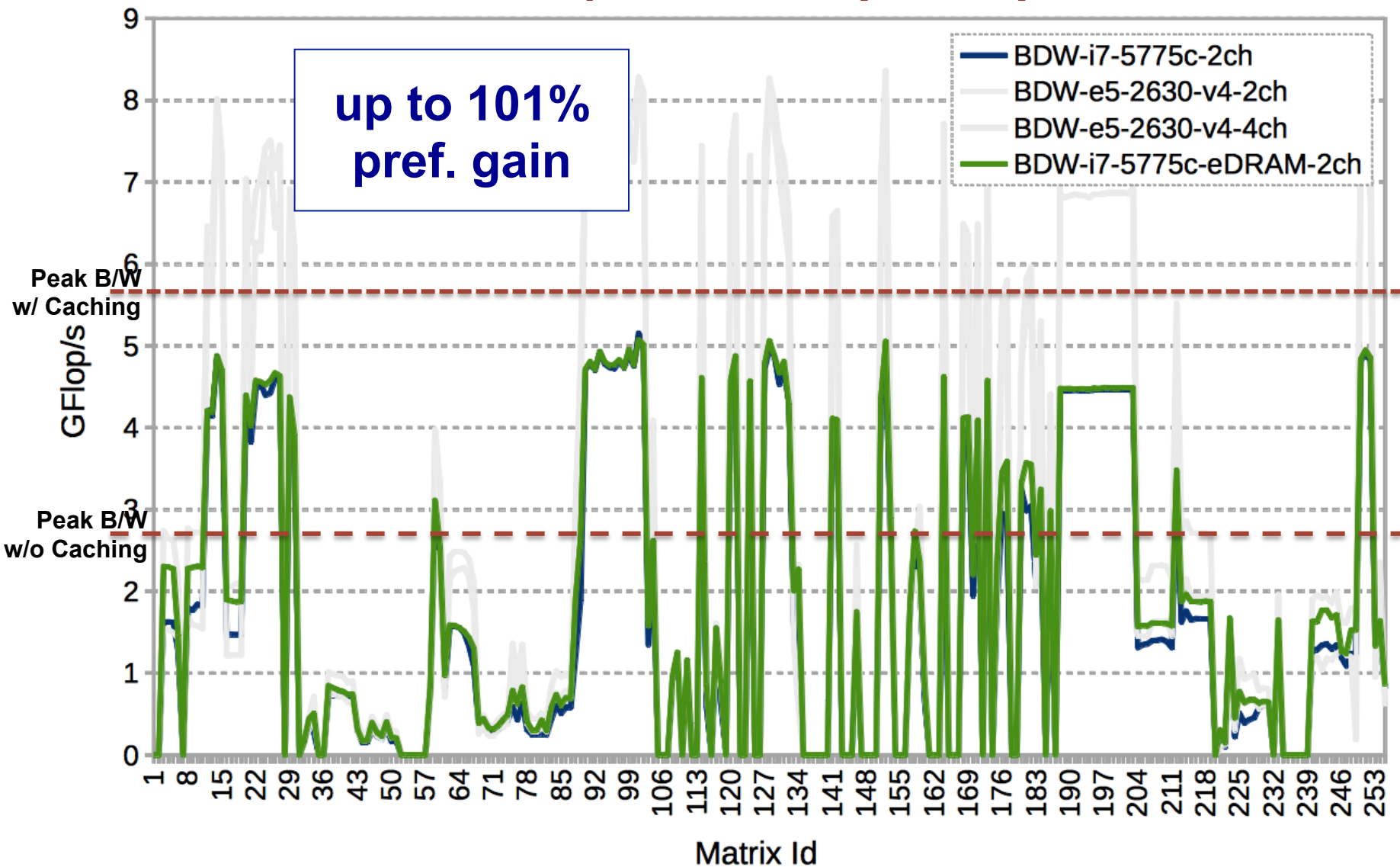
Sparse Matrix-Matrix Multiplication

- Example: $C = AB$

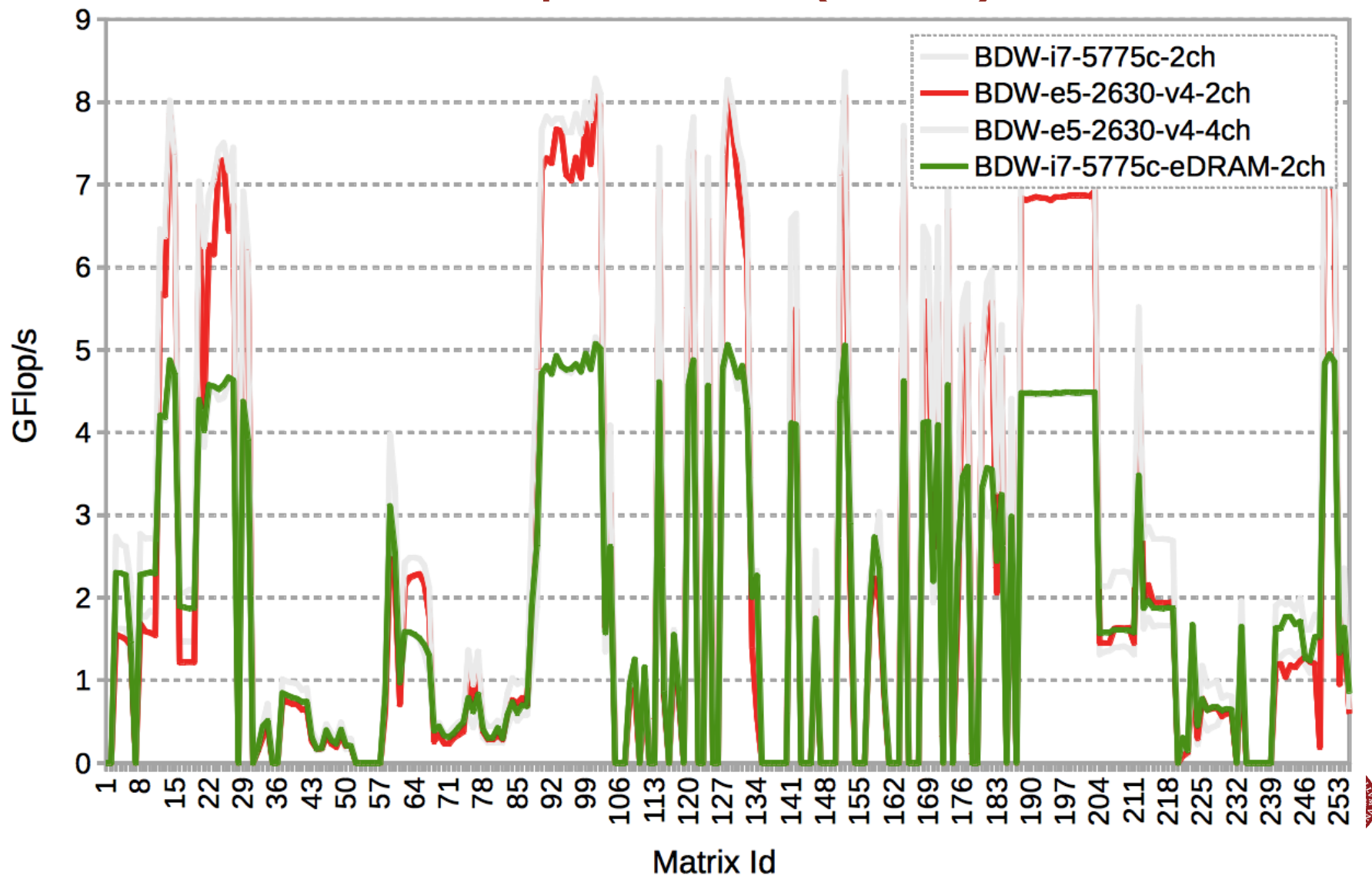
Multiply a sparse matrix A by a sparse matrix B , obtain a sparse matrix C .

<table border="1" style="border-collapse: collapse; text-align: center; width: 100px; height: 100px;"> <tr><td></td><td></td><td>1</td><td></td></tr> <tr><td>2</td><td>3</td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td></td><td>5</td><td>6</td></tr> </table>			1		2	3							4		5	6	x	<table border="1" style="border-collapse: collapse; text-align: center; width: 100px; height: 100px;"> <tr><td></td><td></td><td></td><td>a</td></tr> <tr><td>b</td><td></td><td>c</td><td></td></tr> <tr><td></td><td>d</td><td></td><td>e</td></tr> <tr><td></td><td></td><td>f</td><td></td></tr> </table>				a	b		c			d		e			f		=	<table border="1" style="border-collapse: collapse; text-align: center; width: 100px; height: 100px;"> <tr><td></td><td>1d</td><td></td><td>1e</td></tr> <tr><td>3b</td><td></td><td>3c</td><td>2a</td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td>5d</td><td>6f</td><td>4a+5e</td></tr> </table>		1d		1e	3b		3c	2a						5d	6f	4a+5e
		1																																																		
2	3																																																			
4		5	6																																																	
			a																																																	
b		c																																																		
	d		e																																																	
		f																																																		
	1d		1e																																																	
3b		3c	2a																																																	
	5d	6f	4a+5e																																																	
A (4x4) $nnzA = 6$		B (4x4) $nnzB = 6$		C (4x4) $nnzC = 8$																																																

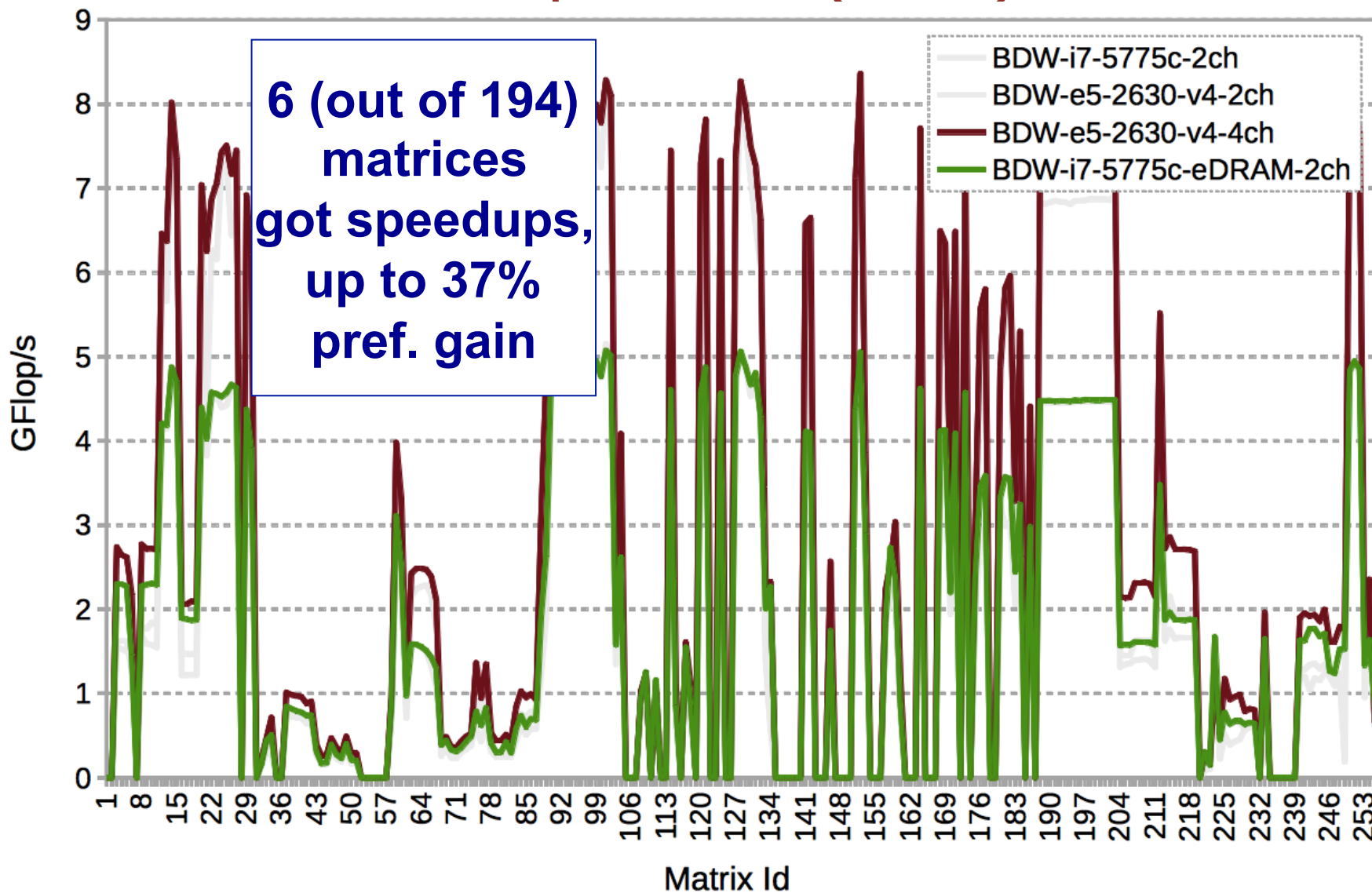
CSR-based SpGEMM (MKL)



CSR-based SpGEMM (MKL)



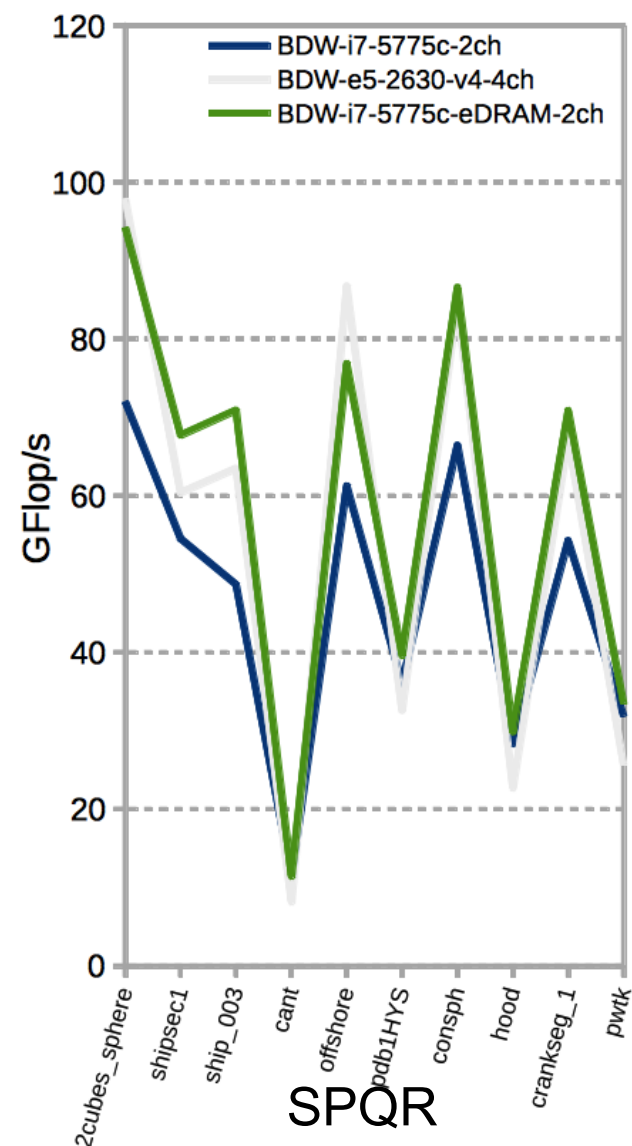
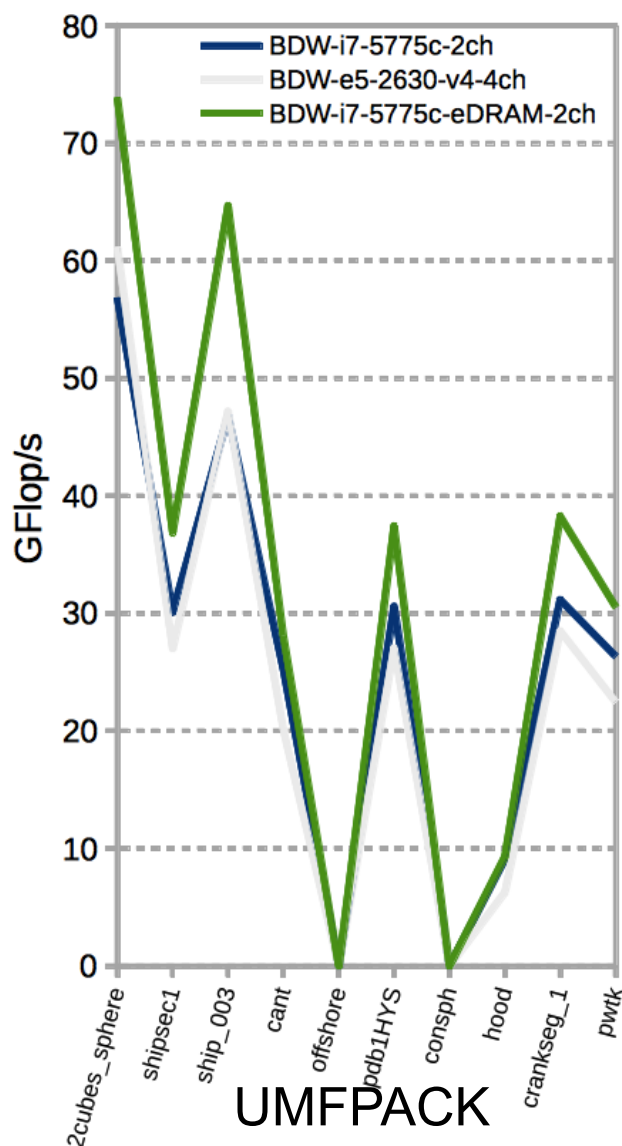
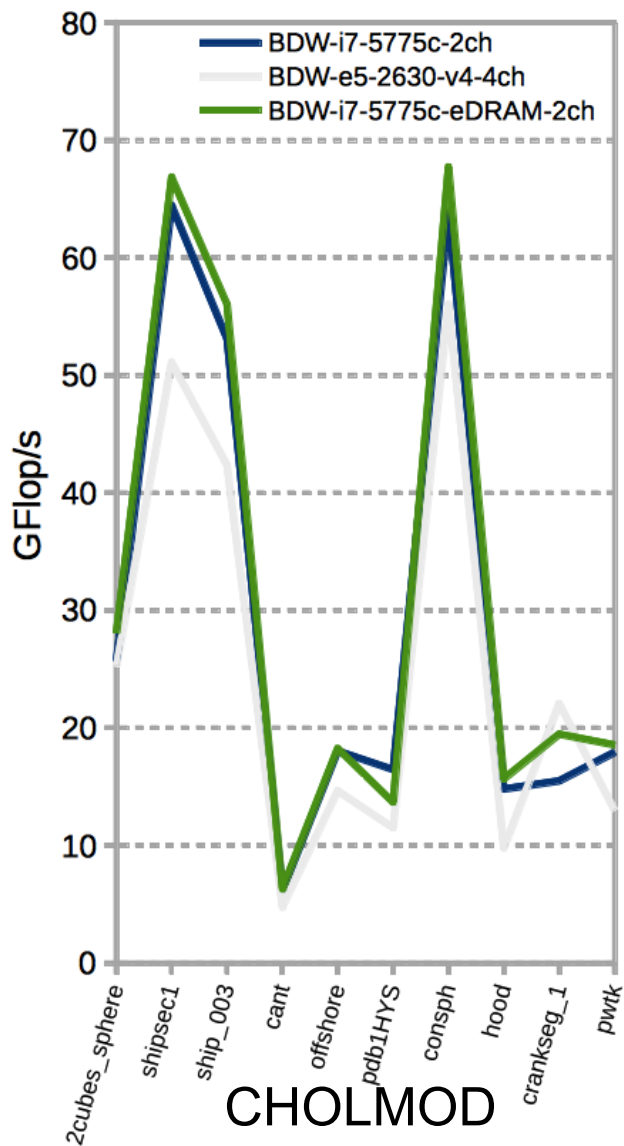
CSR-based SpGEMM (MKL)



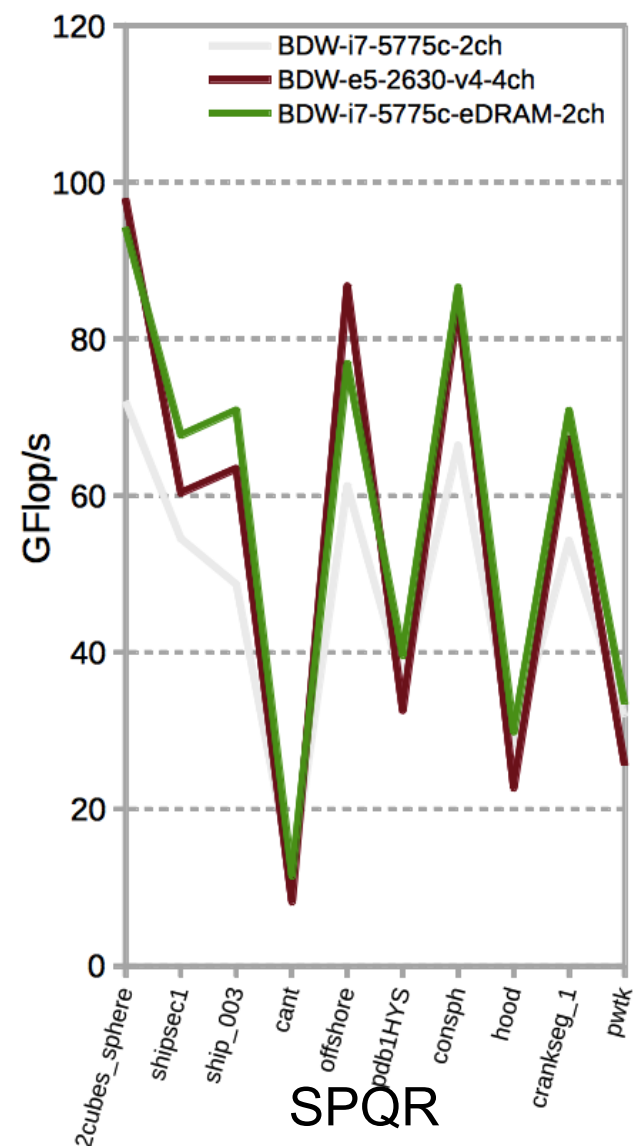
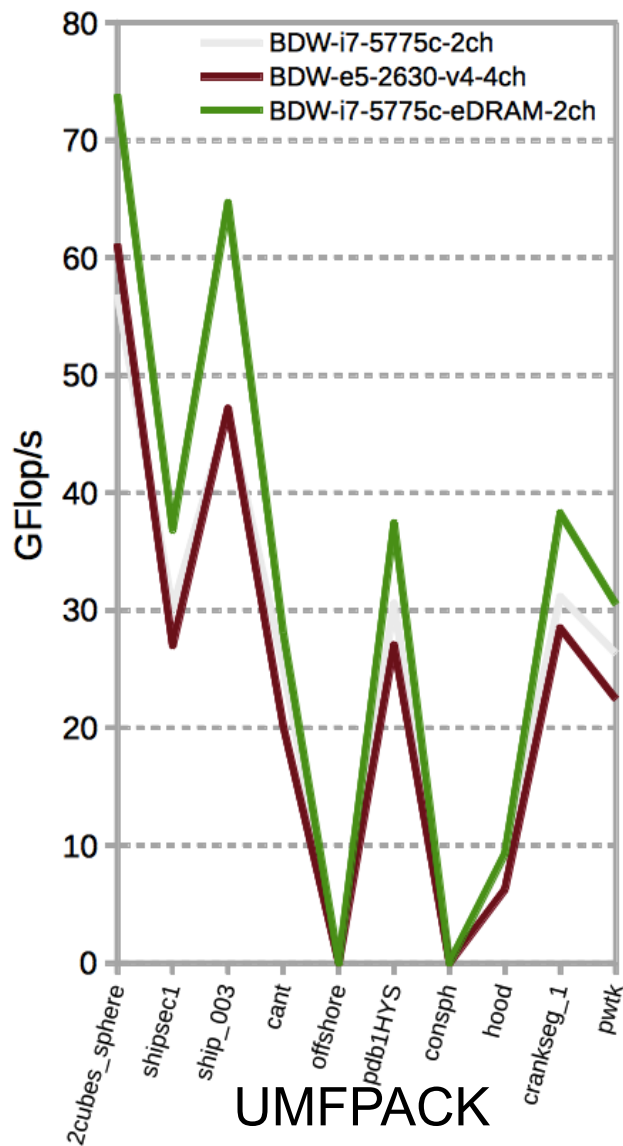
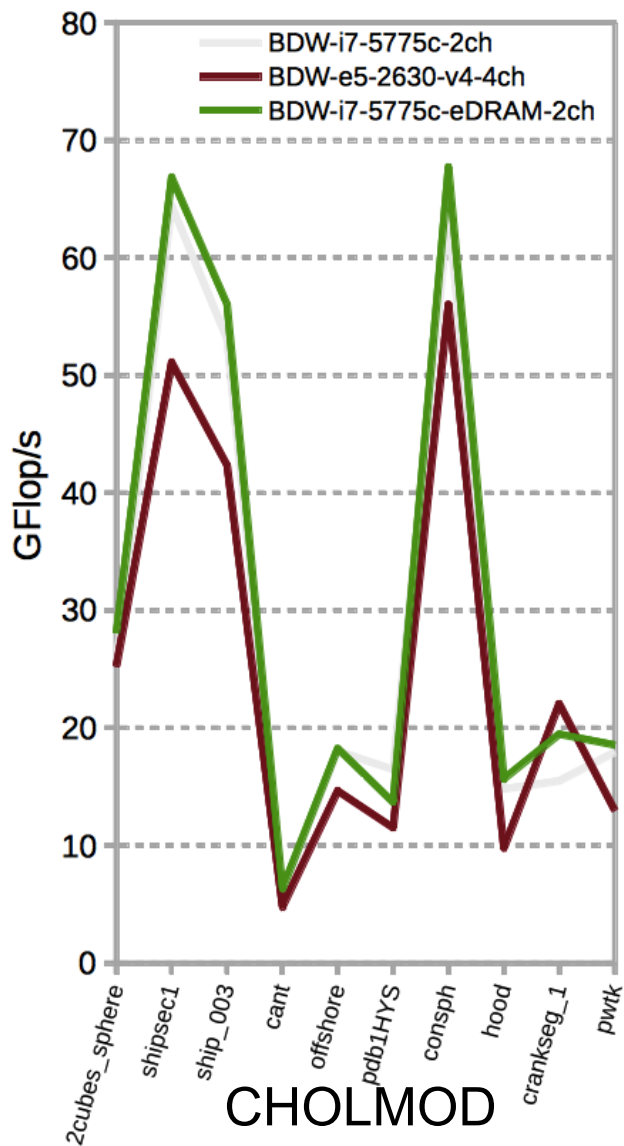
Sparse Direct Solvers



Cholesky, LU, & QR (SuiteSparse)



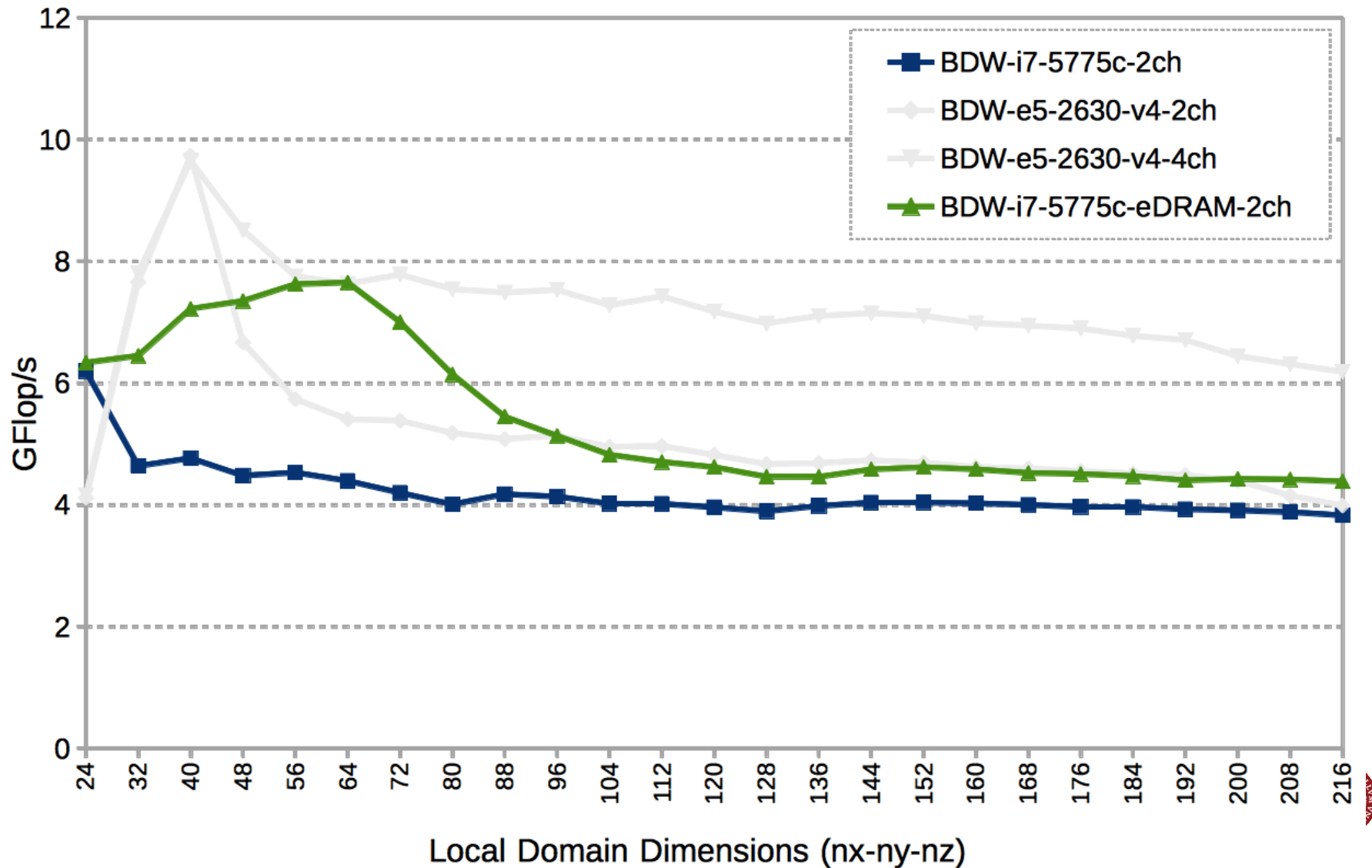
Cholesky, LU, & QR (SuiteSparse)



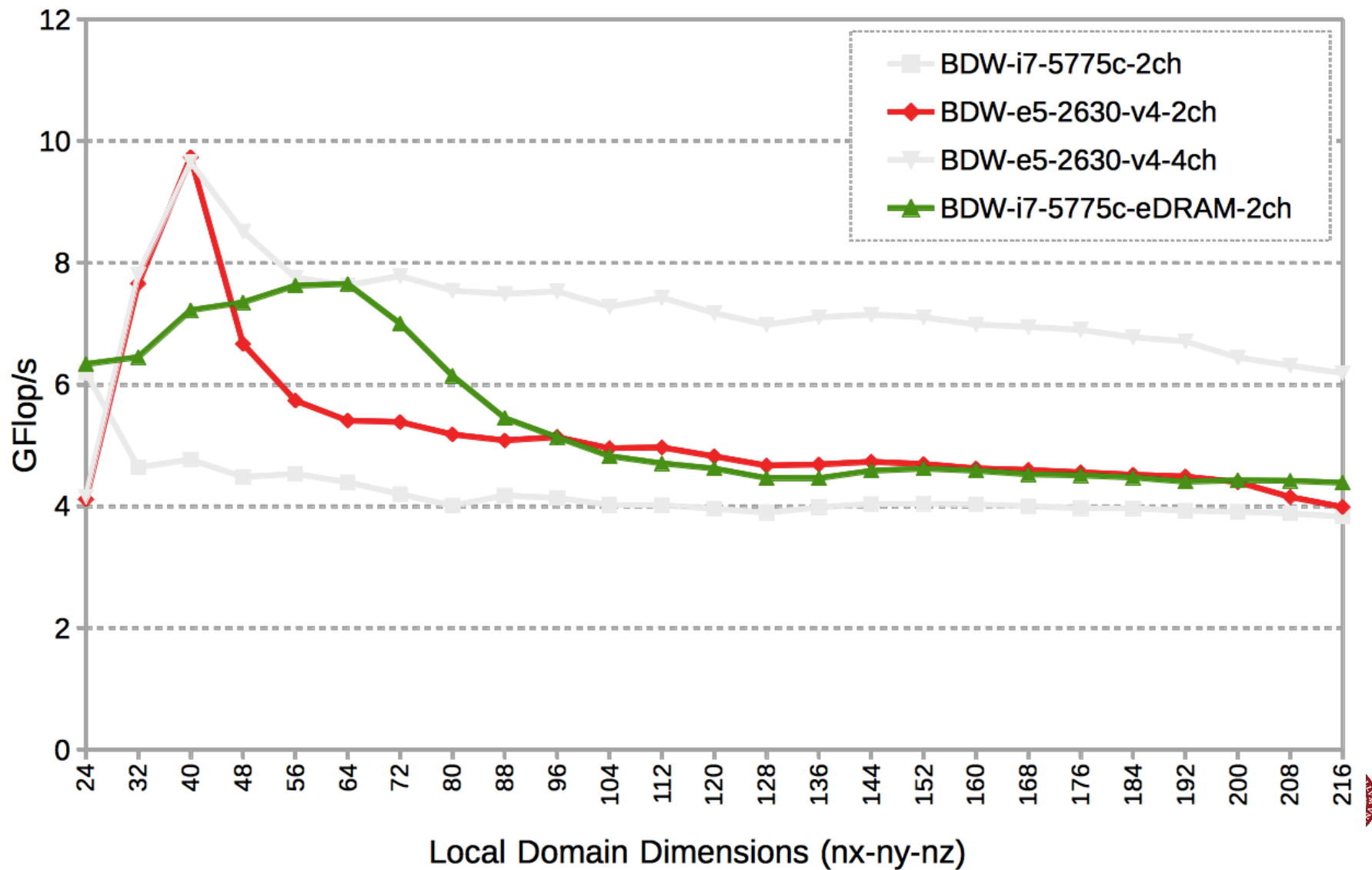
HPCG Benchmark



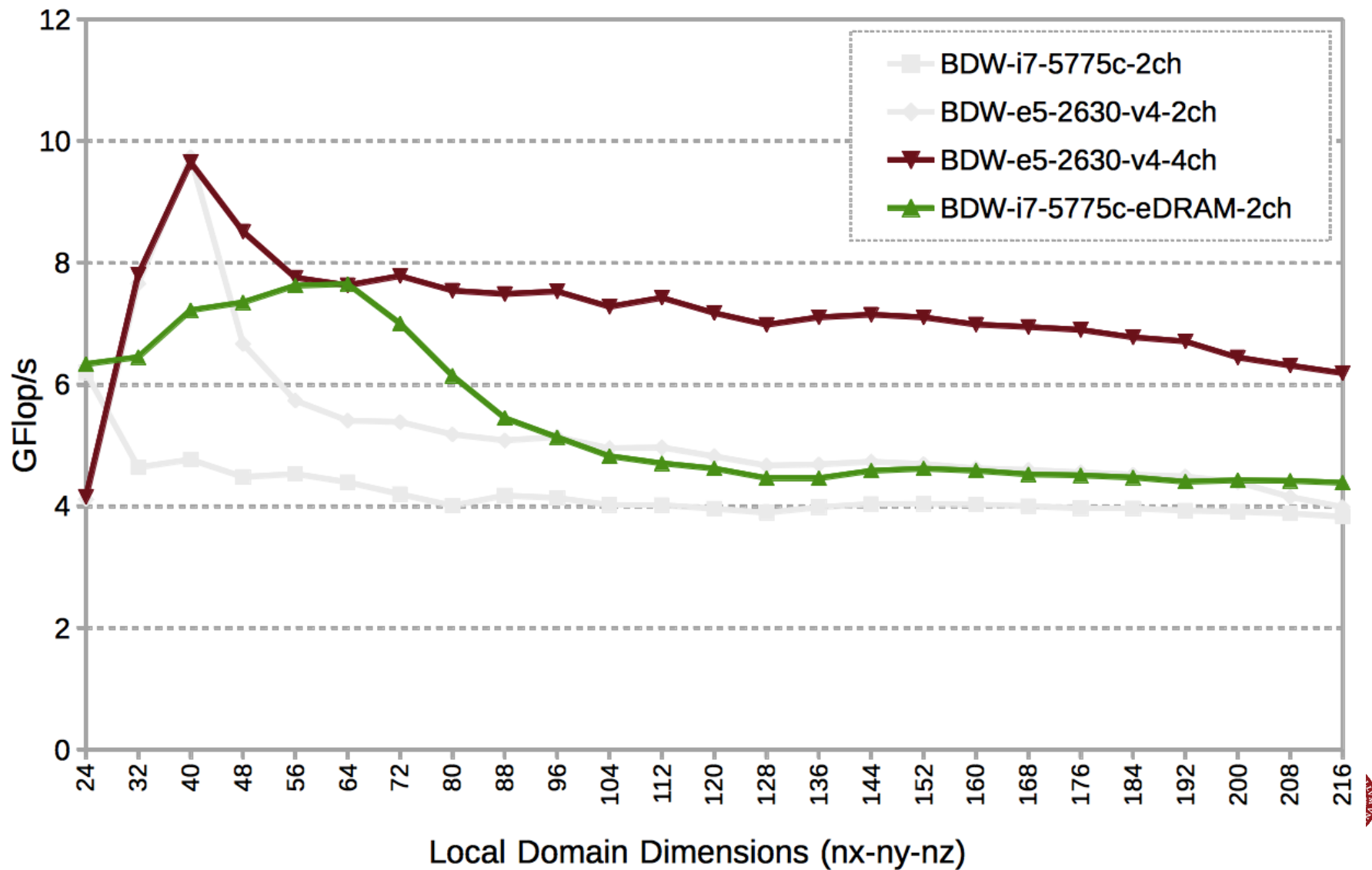
HPCG Benchmark



HPCG Benchmark



HPCG Benchmark



T2. Scalability on Integrated GPU

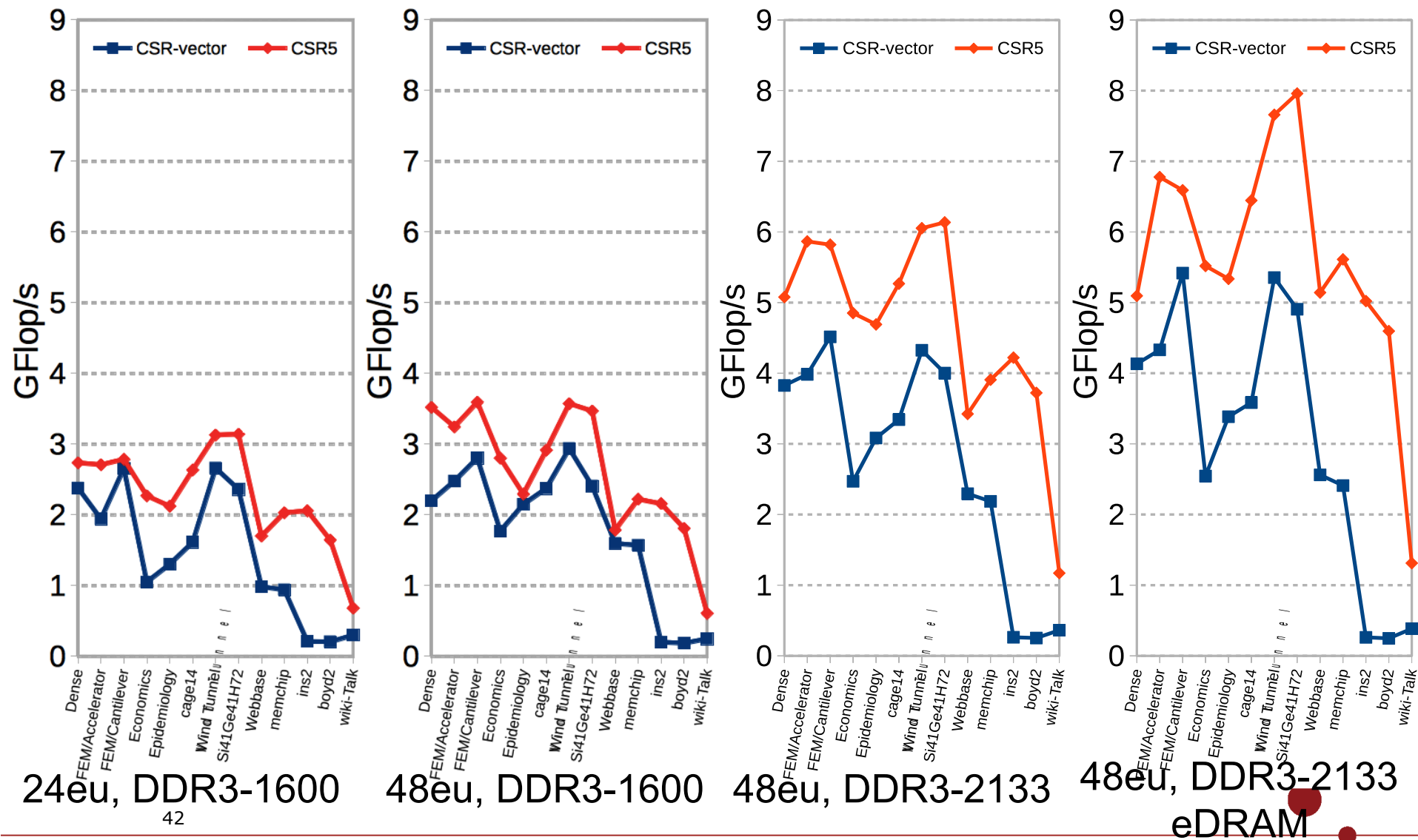


Testbed

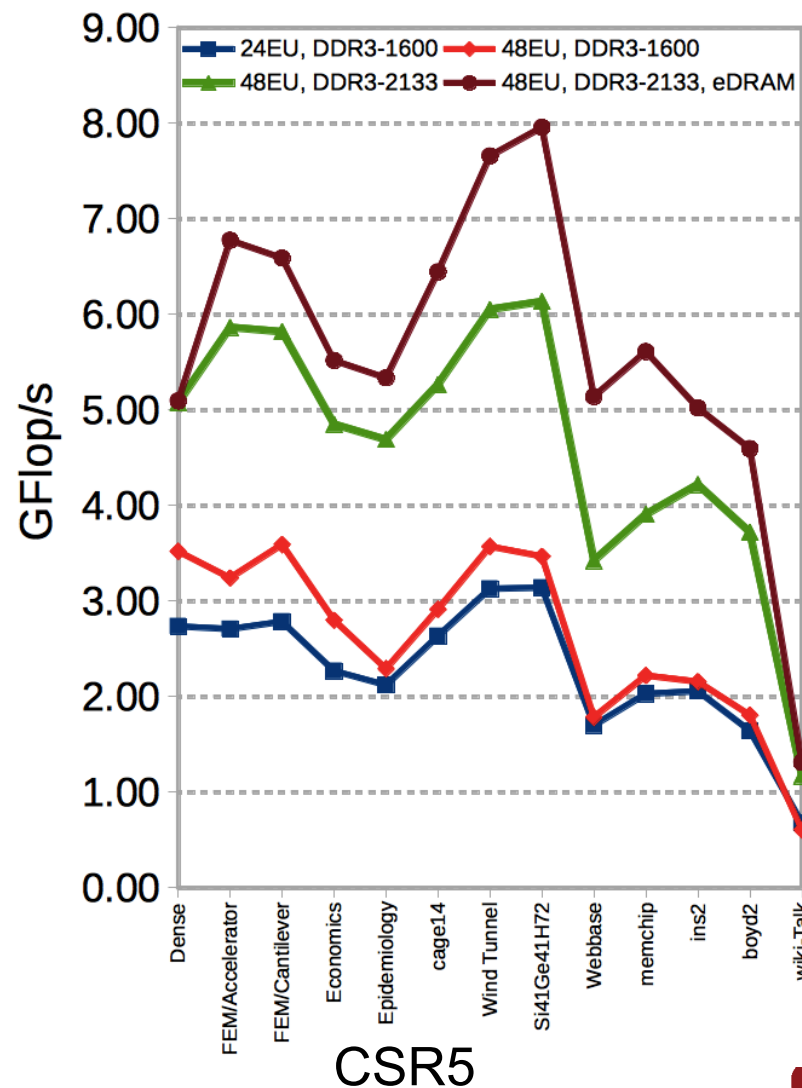
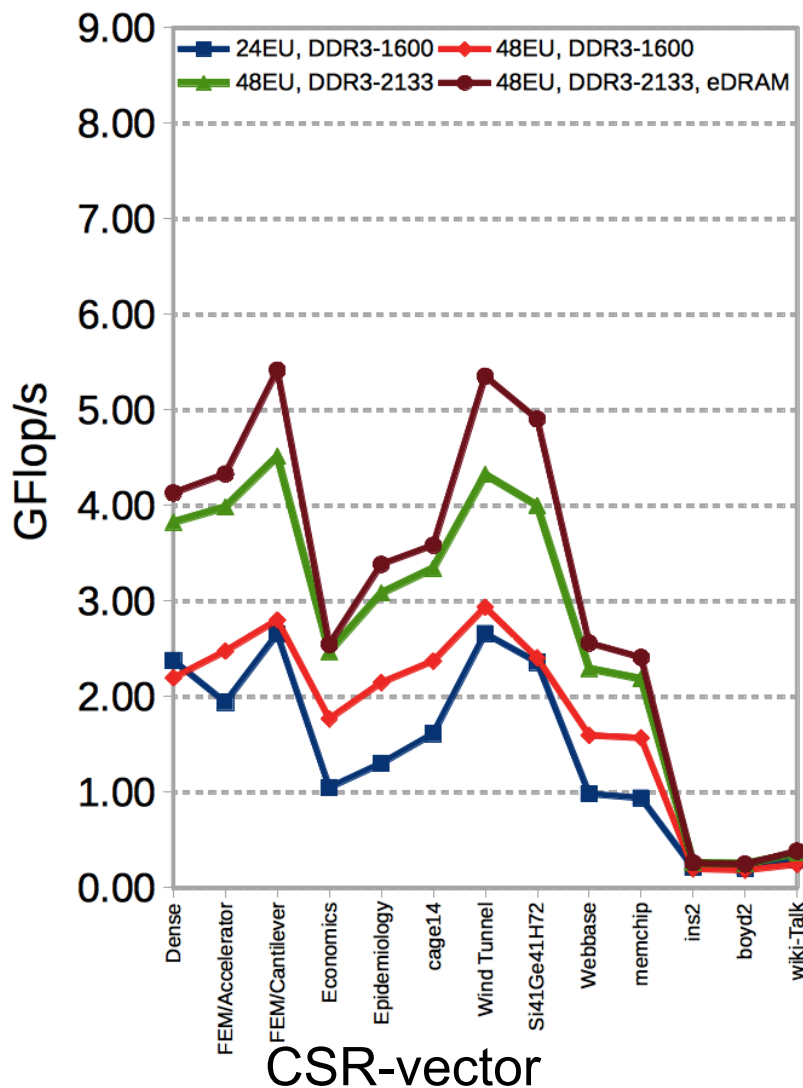
	i3-5010u	i5-5250u	i7-5775c	i7-5775c
Family	Broadwell	Broadwell	Broadwell	Broadwell
CPU	2c @ 2.1 GHz	2c @ 2.7 GHz	4c @ 3.7 GHz	4c @ 3.7 GHz
GPU	24eu @ 0.9 GHz	48eu @ 0.95 GHz	48eu @ 1.15 GHz	48eu @ 1.15 GHz
LLC	3 MB	3 MB	6 MB	6 MB
eDRAM	--	--	--	128 MB
DRAM	2ch DDR3-1600	2ch DDR3-1600	2ch DDR3-2133	2ch 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)



Scalability (algorithm-wise)



T3. New Opportunities

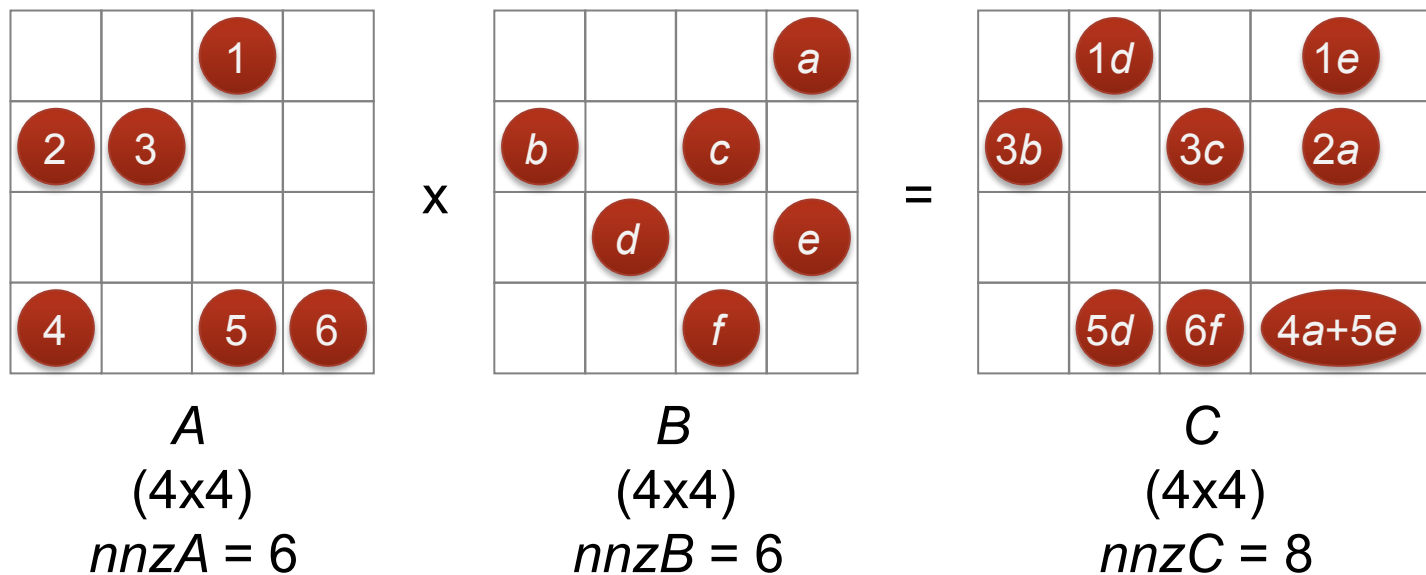
Case 1. Memory Re-allocation (SpGEMM)



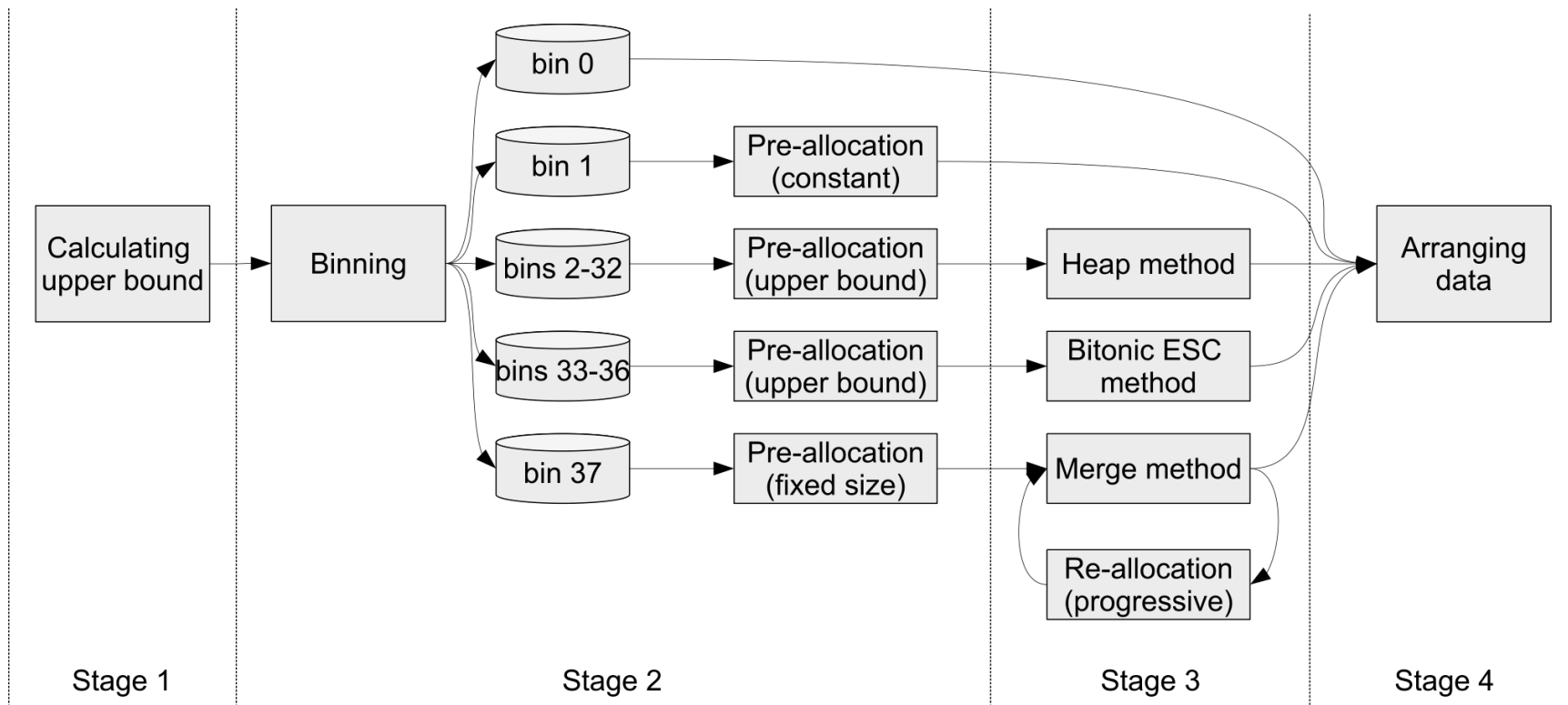
Sparse Matrix-Matrix Multiplication

- Example: $C = AB$

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

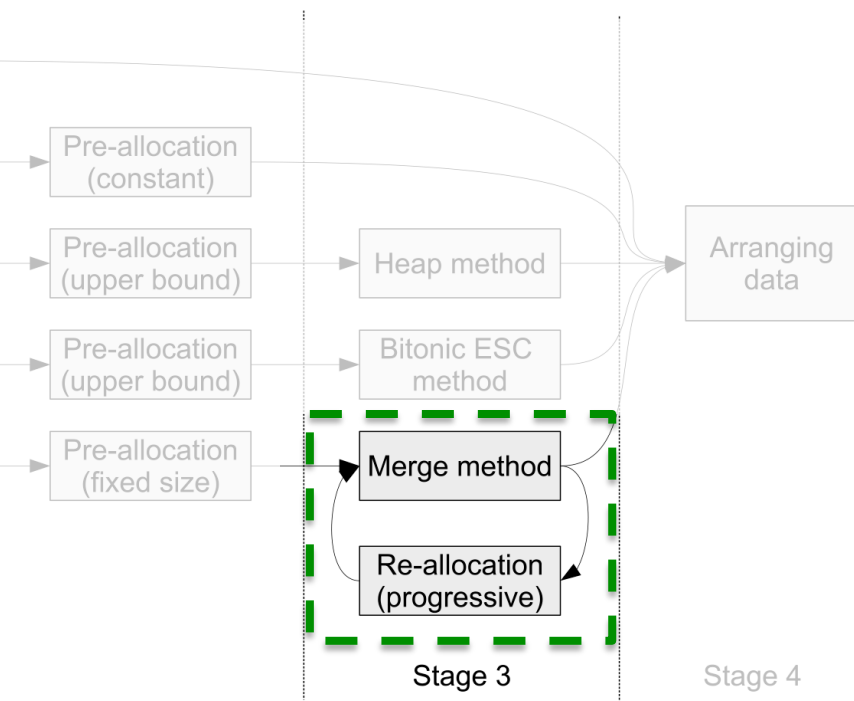


SpGEMM Framework

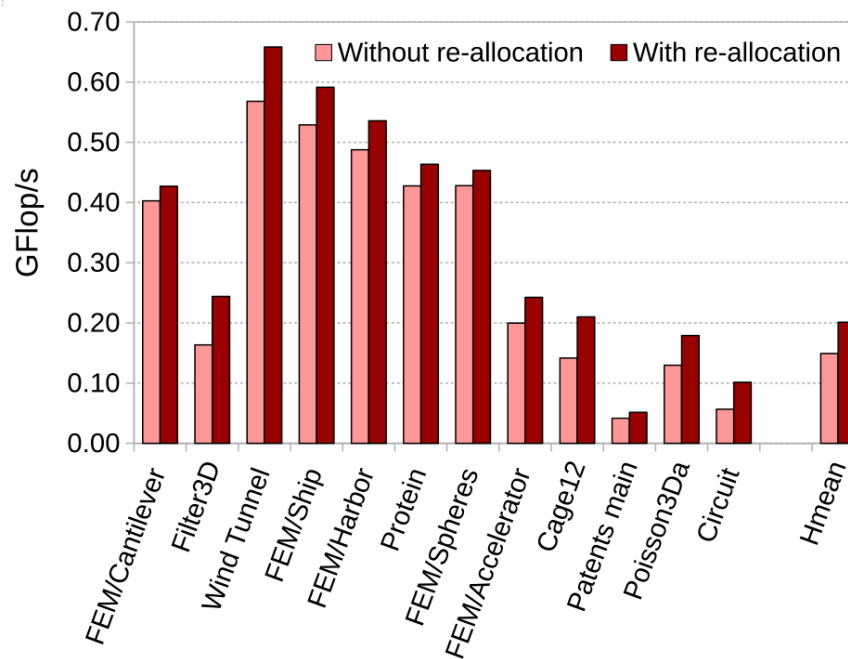
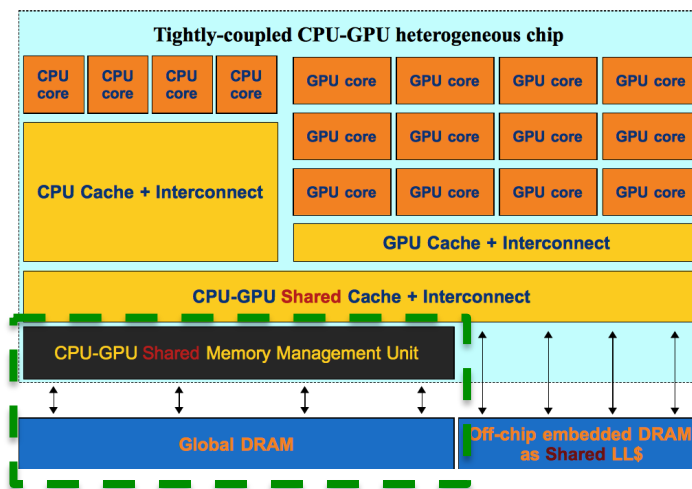


W. Liu and B. Vinter. **A Framework for General Sparse Matrix-
Matrix Multiplication on GPUs and Heterogeneous Processors.**
Journal of Parallel and Distributed Computing. 2015. (extended from
IPDPS '14)

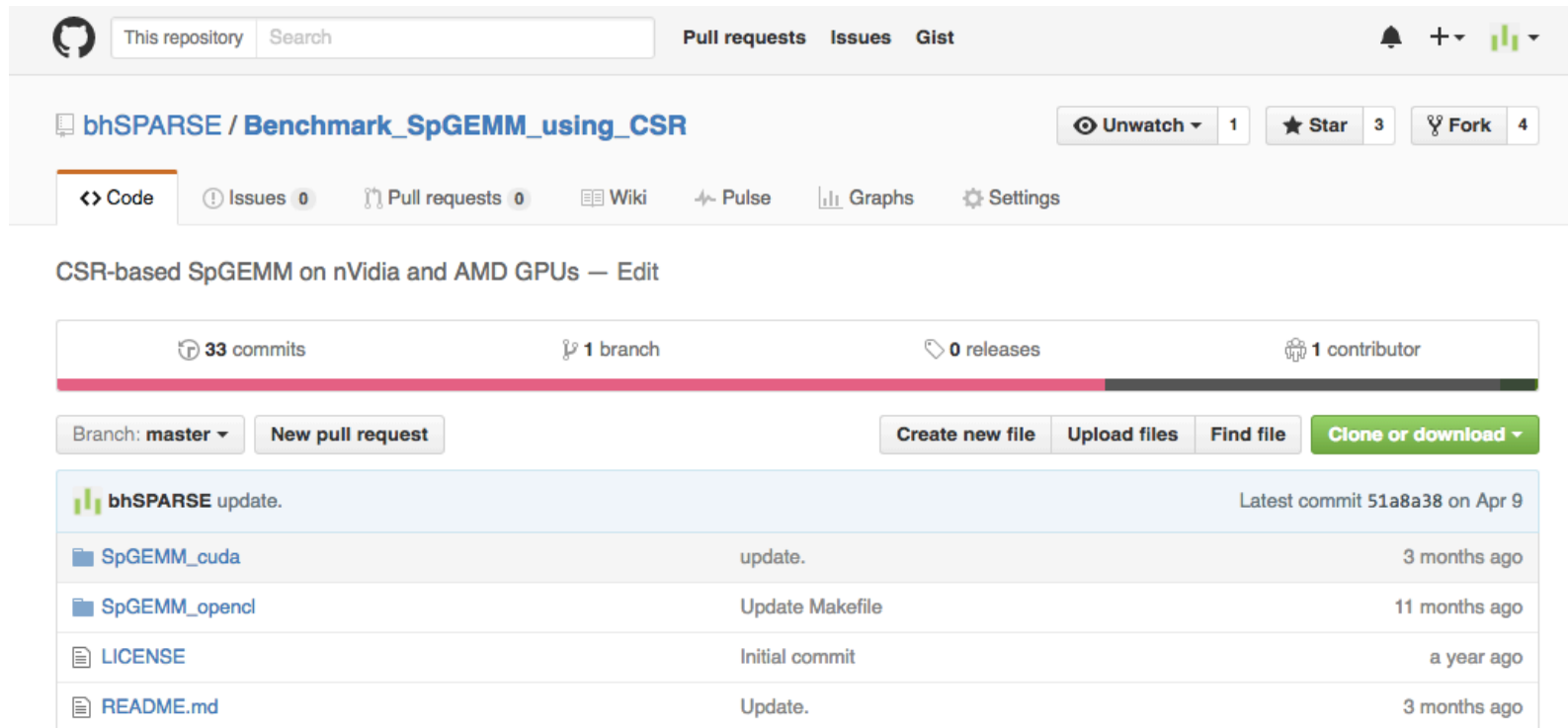
Memory Re-allocation on Hetero-chips



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



Source Code at Github



CSR-based SpGEMM on nVidia and AMD GPUs — Edit

33 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

File/Folder	Commit Message	Time Ago
SpGEMM_cuda	update.	3 months ago
SpGEMM_openc1	Update Makefile	11 months ago
LICENSE	Initial commit	a year ago
README.md	Update.	3 months ago

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 .

		1	
2	3		
4		5	6

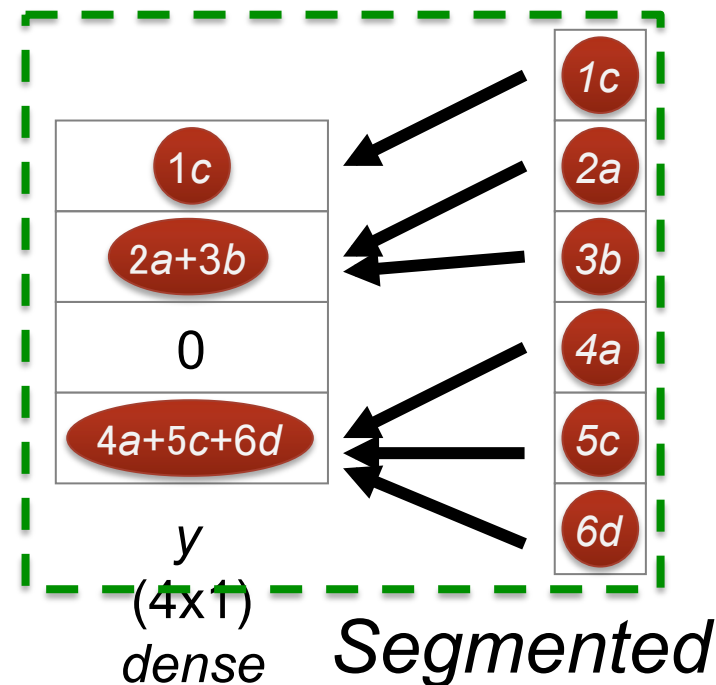
A
(4x4)
 $nnzA = 6$

\times

a
b
c
d

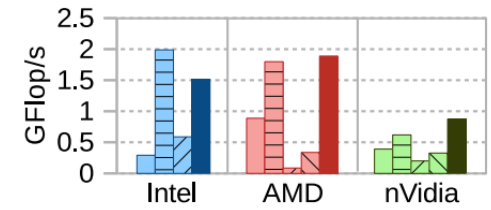
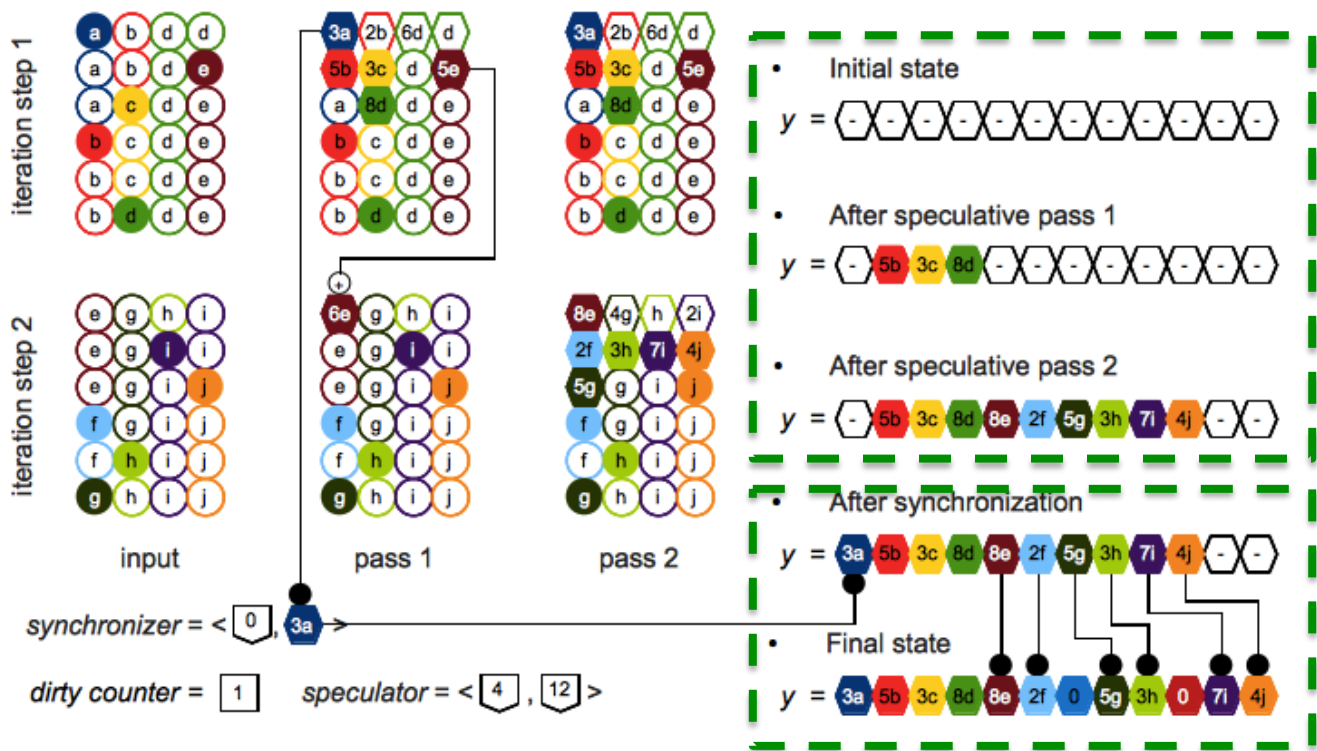
x
(4x1)
dense

=

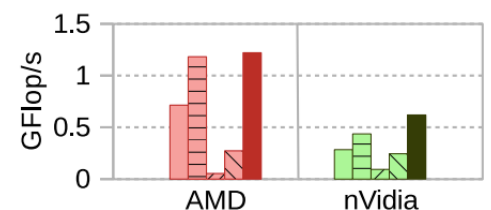


SpMV using Speculative Seg. Sum

row pointer = $\langle 0, 3, 8, 11, 19, 27, 29, 34, 37, 37, 44, 48 \rangle$ boundary = $\langle 0, 24, 48 \rangle$ tile offset = $\langle 0, 4, 12 \rangle$



single precision



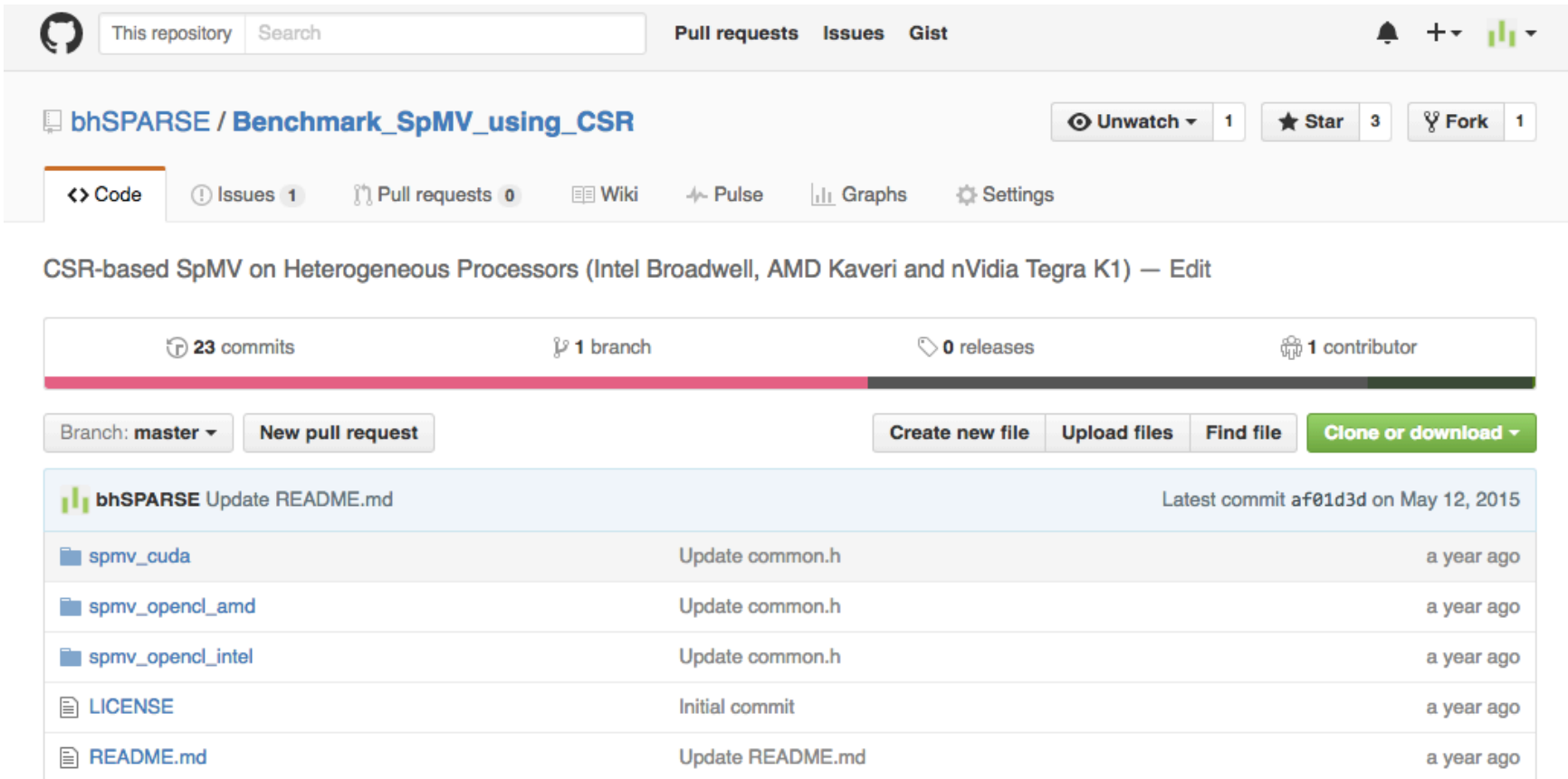
double precision

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



The screenshot shows the GitHub interface for the repository `bhSPARSE / Benchmark_SpMV_using_CSR`. At the top, there is a search bar and navigation links for Pull requests, Issues, and Gist. The repository name is displayed in blue, with statistics for Unwatch (1), Star (3), and Fork (1). Below this, there are tabs for Code, Issues (1), Pull requests (0), Wiki, Pulse, Graphs, and Settings. The main heading is "CSR-based SpMV on Heterogeneous Processors (Intel Broadwell, AMD Kaveri and nVidia Tegra K1) — Edit". A summary bar shows 23 commits, 1 branch, 0 releases, and 1 contributor. Below this are buttons for "Branch: master", "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". The commit history table is as follows:

Commit	Message	Time
 <code>bhSPARSE</code> Update README.md	Update README.md	Latest commit af01d3d on May 12, 2015
 <code>spmv_cuda</code>	Update common.h	a year ago
 <code>spmv_opencl_amd</code>	Update common.h	a year ago
 <code>spmv_opencl_intel</code>	Update common.h	a year ago
 LICENSE	Initial commit	a year ago
 README.md	Update README.md	a year ago

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
---	---	---	---	----	----	----

