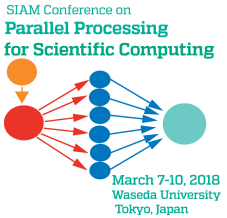


When Sparse Matrices Met Heterogeneous Processors: Opportunities and Challenges

Weifeng Liu

Norwegian University of Science and Technology, Norway

NTNU
Norwegian University of
Science and Technology

Norwegian University of Science and Technology

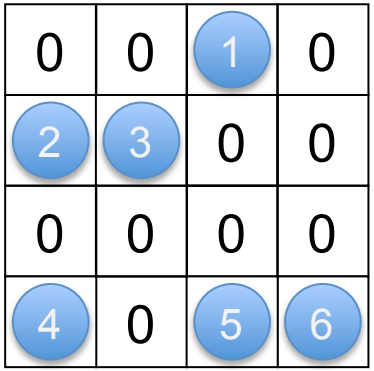
March 10th, 2018, Tokyo, Japan

Background

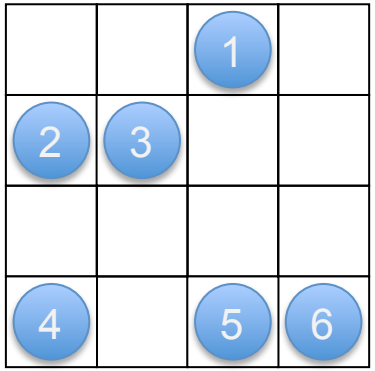
Sparse Matrix
&
Heterogeneous Processor
(Tightly-coupled CPU-GPU Chip)

Sparse matrix

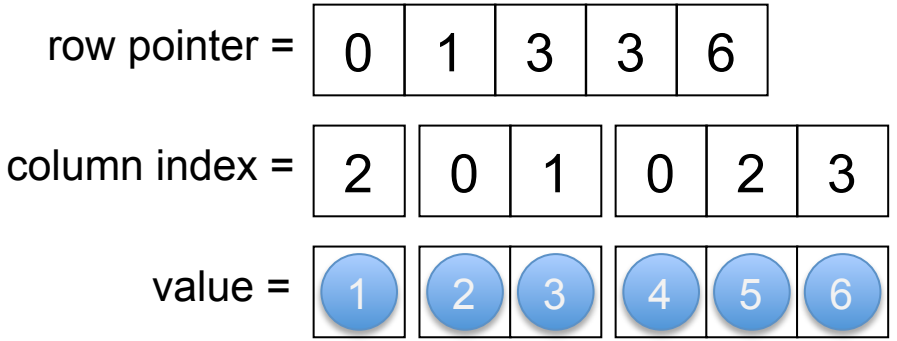
- If the majority of entries in a matrix are zeros, we can store the matrix by using a sparse representation and only records nonzero entries.
- The Compressed Sparse Row (CSR) format is the most widely used.



A
(4x4)
dense

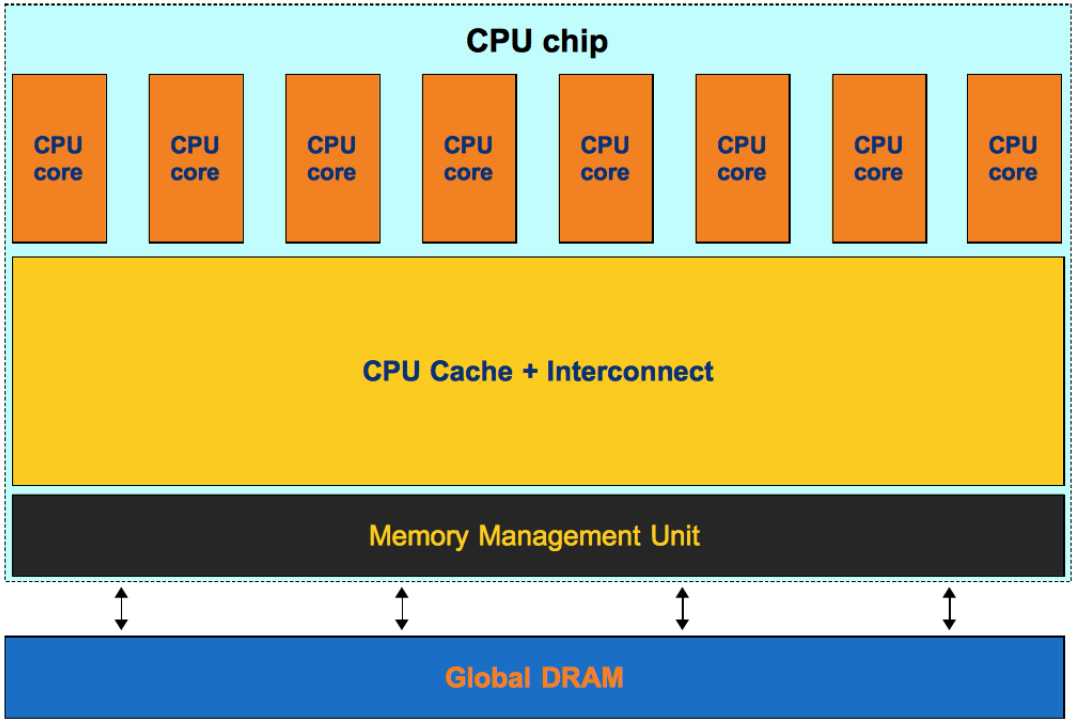


A
(4x4)
sparse, 6 nonzeros

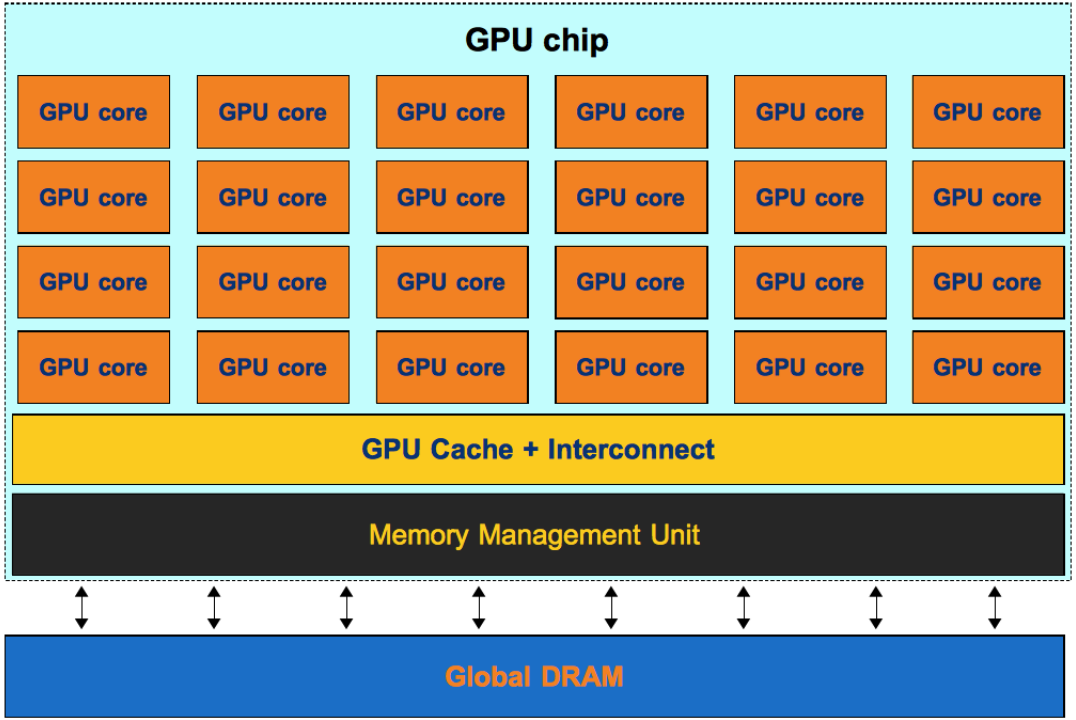


A (in CSR)
(4x4)
6 nonzeros

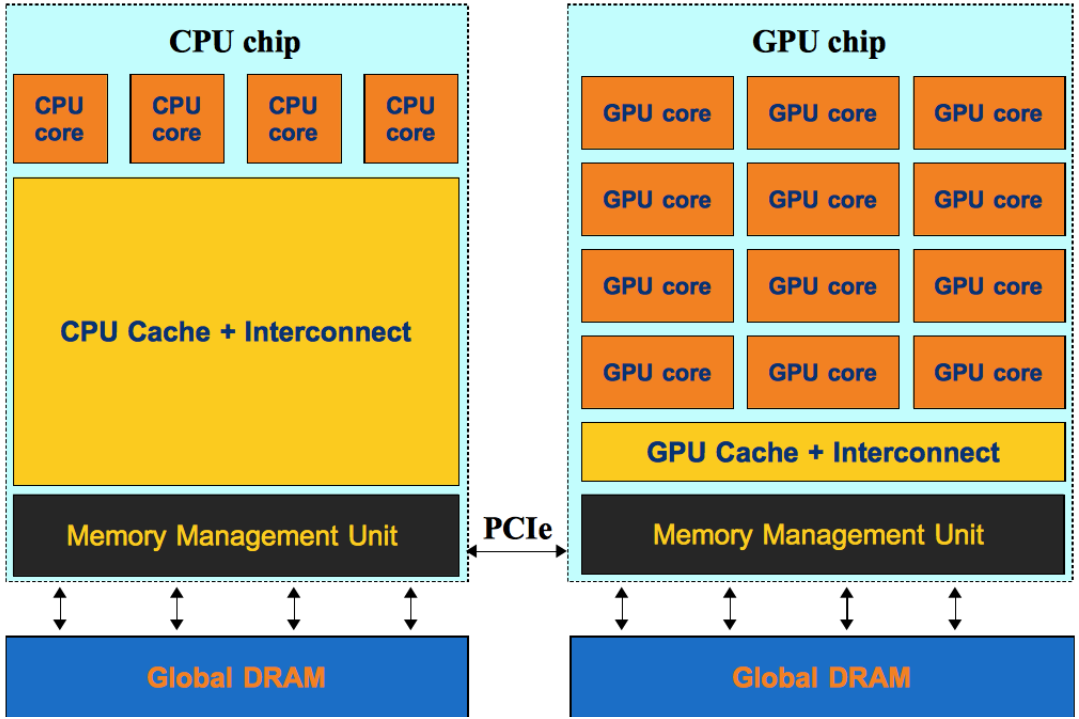
CPU chip


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

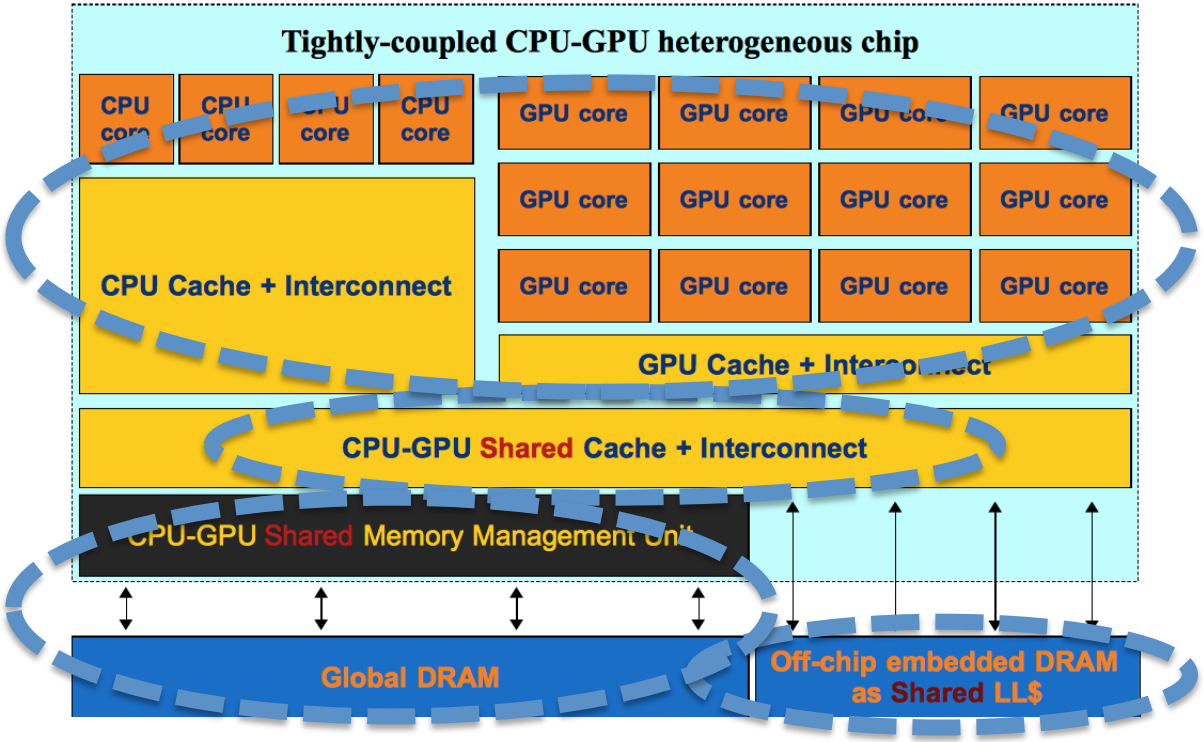
GPU chip


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

Loosely-coupled CPU-GPU platform



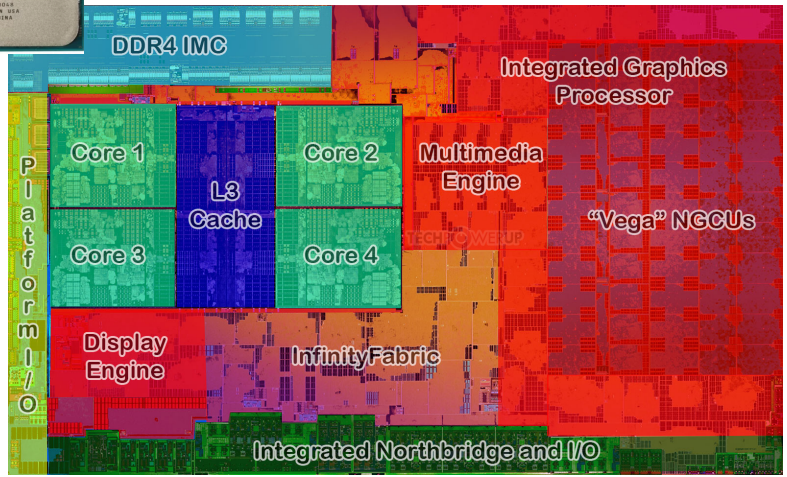
Tightly-coupled CPU-GPU chip



Experimental Setup

One APU (AMD R5-2400G)
&
Four Sparse Matrix Kernels (OMP/OCL)
&
956 Sparse Matrices (from SuiteSparse)

Testbed



Ryzen 5 2400G, Zen+Vega, quad-core CPU, 4MB L3 cache, 704 GCN-core GPU), 2-ch 16GB DDR4-2933, 46.9 GB/s B/W. MS Win10 :O

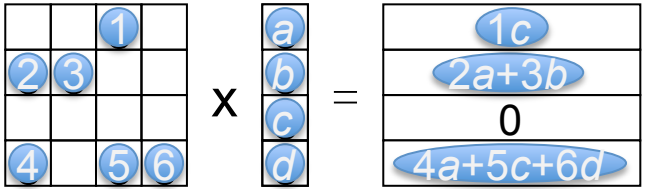
nVidia GeForce Titan X, Pascal GP102, 3584 CUDA cores, 3 MB L2 cache, 12GB GDDR5X, 480 GB/s B/W.

(only used for some perf. comparison)

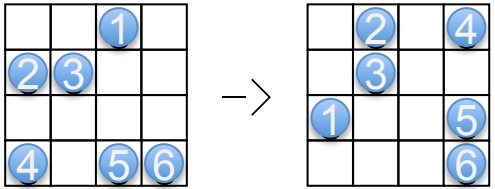
NTNU

Four sparse kernels

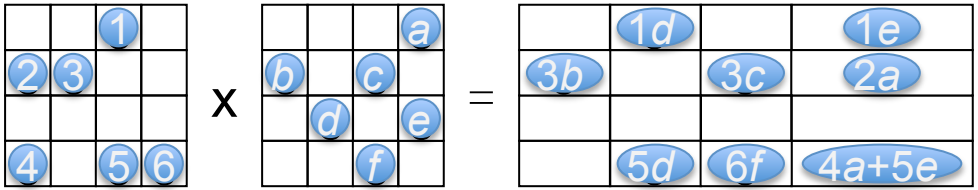
- Sparse matrix-vector Multiplication (SpMV)



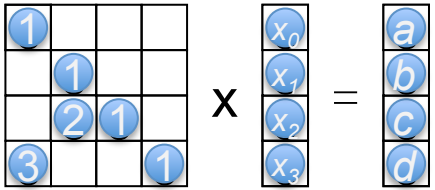
- Sparse transposition (SpTRANS)



- Sparse matrix-matrix Multiplication (SpGEMM)

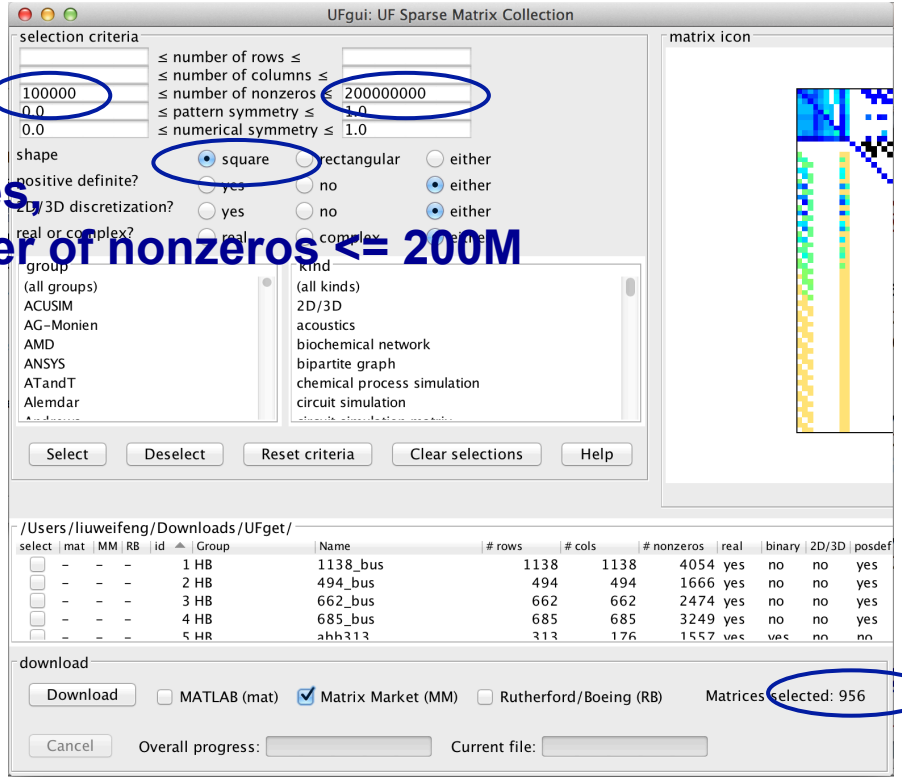


- Sparse triangular solve (SpTRSV)



956 matrices from SuiteSparse Collection

square matrices,
100K <= number of nonzeros <= 200M



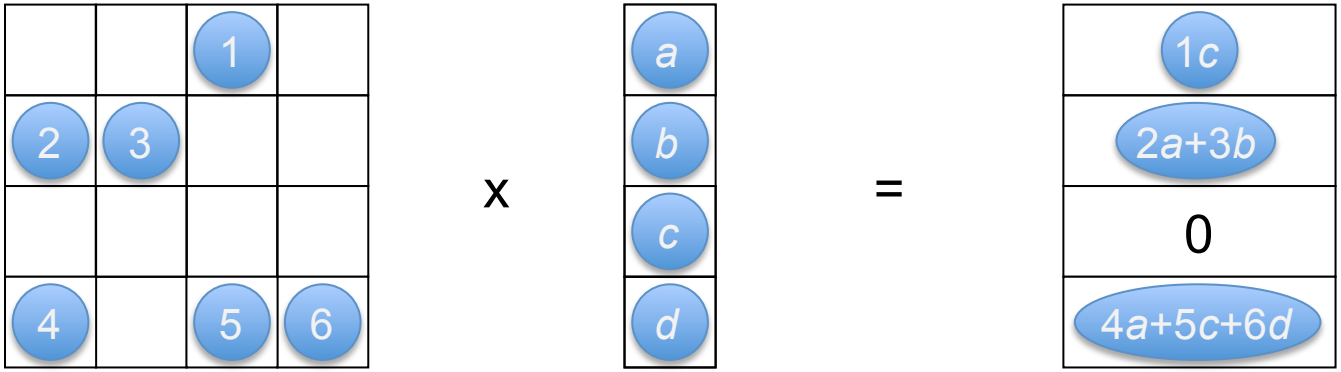
956 matrices from in total 2757 matrices

NTNU

Kernel 1. Sparse Matrix-Vector Multiplication (SpMV)

SpMV

- Multiply a sparse matrix *A* by a dense vector *x*, and obtain a resulting dense vector *y*.



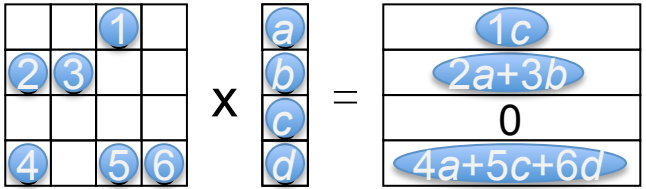
A
(4x4)
sparse, 6 nonzeros

x
(4x1)
dense

y
(4x1)
dense

Question, Experiment, Opportunity and Challenge

- Sparse matrix-vector Multiplication (SpMV)



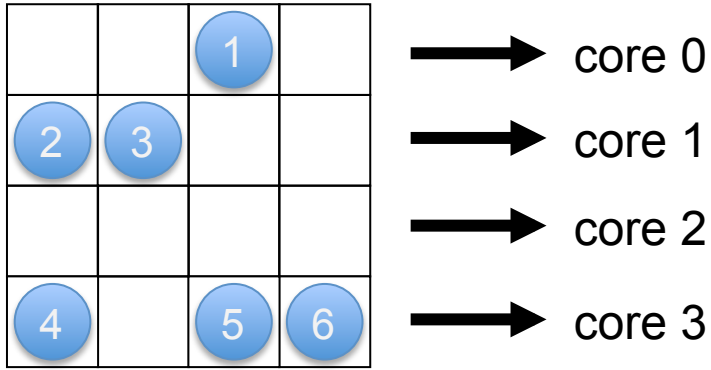
Question 1: Who (CPU side or GPU side) gives the **best** performance?

Experiment 1: Test four SpMV methods (CSR-omp, CSR5-omp, CSR-adaptive-opencl, CSR5-opencl) on CPU side and GPU side.

Opportunity 1 and Challenge 1

Parallel SpMV (using the CSR format)

- Assign a row block to one core of a many-core processor.

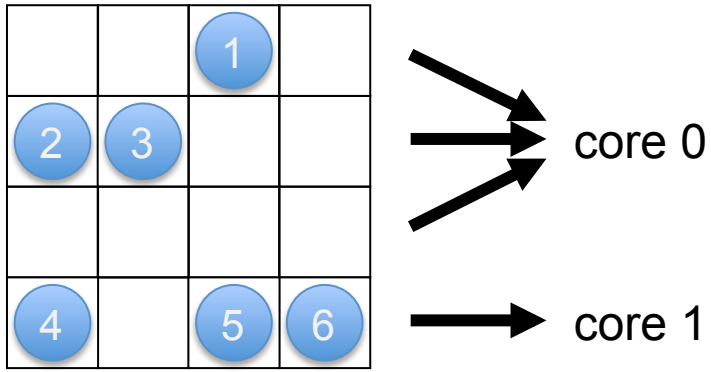


Probably imbalanced computation, degraded performance on many-core processors

Nathan Bell, Michael Garland. **Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors**. *SC09*.

Parallel SpMV (using CSR-Adaptive)

- Adaptively assign row blocks to cores for better load balancing.

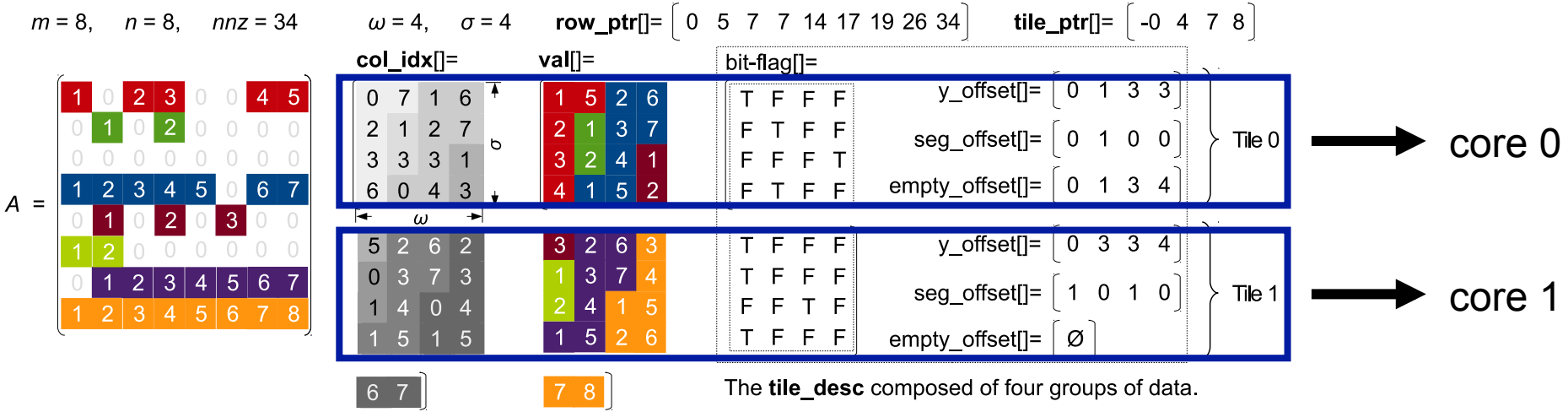


But when several rows are much longer (power-law distribution), possible imbalance still exists.

Joseph L. Greathouse, Mayank Daga. **Efficient sparse matrix-vector multiplication on GPUs using the CSR storage format**. *SC14*.

Parallel SpMV (using CSR5) (our work)

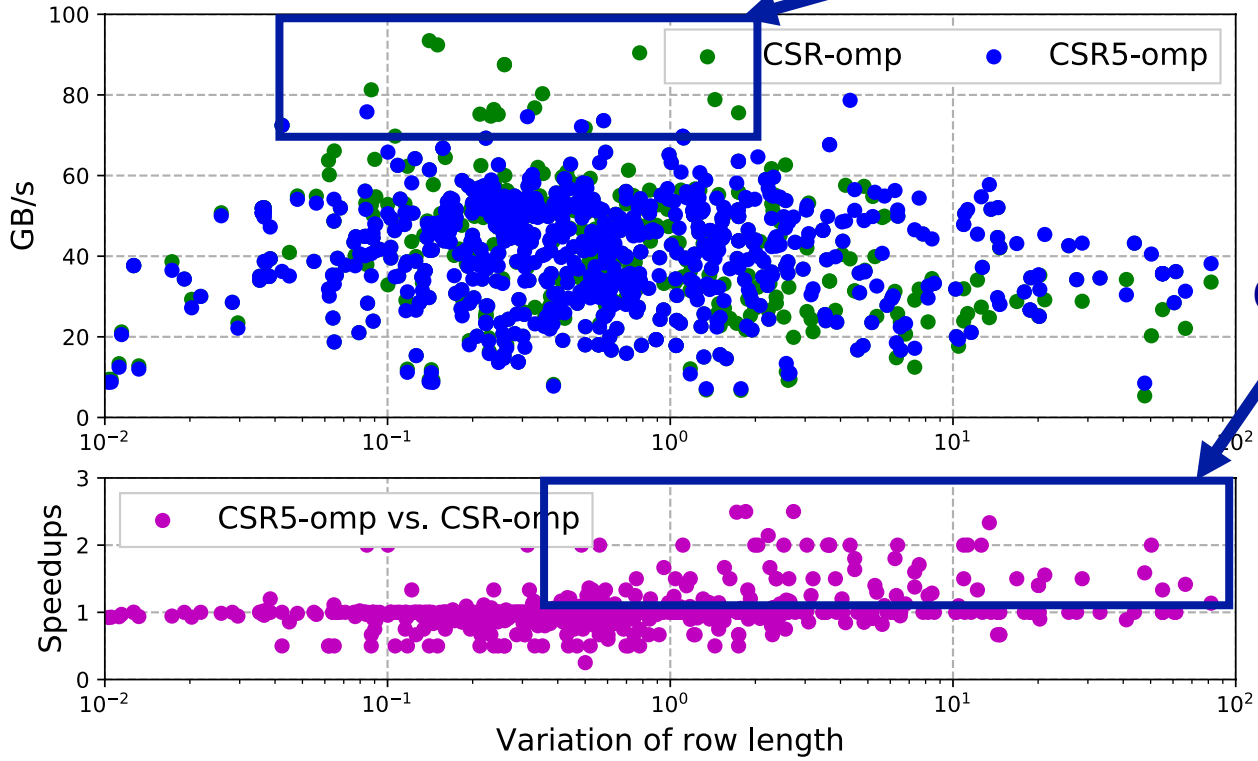
- Organize nonzeros in Tiles of identical size. The design objectives include load balancing, SIMD-friendly, low preprocessing cost and reduced storage space.



Weifeng Liu, Brian Vinter. **CSR5: An Efficient Storage Format for Cross-Platform Sparse Matrix-Vector Multiplication**. *ICS15*.

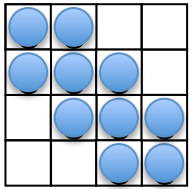
SpMV on CPU side

best `top' performance: CSR-omp wins

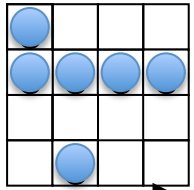


best `load-balanced' performance: CSR5-omp wins

uniform dist.



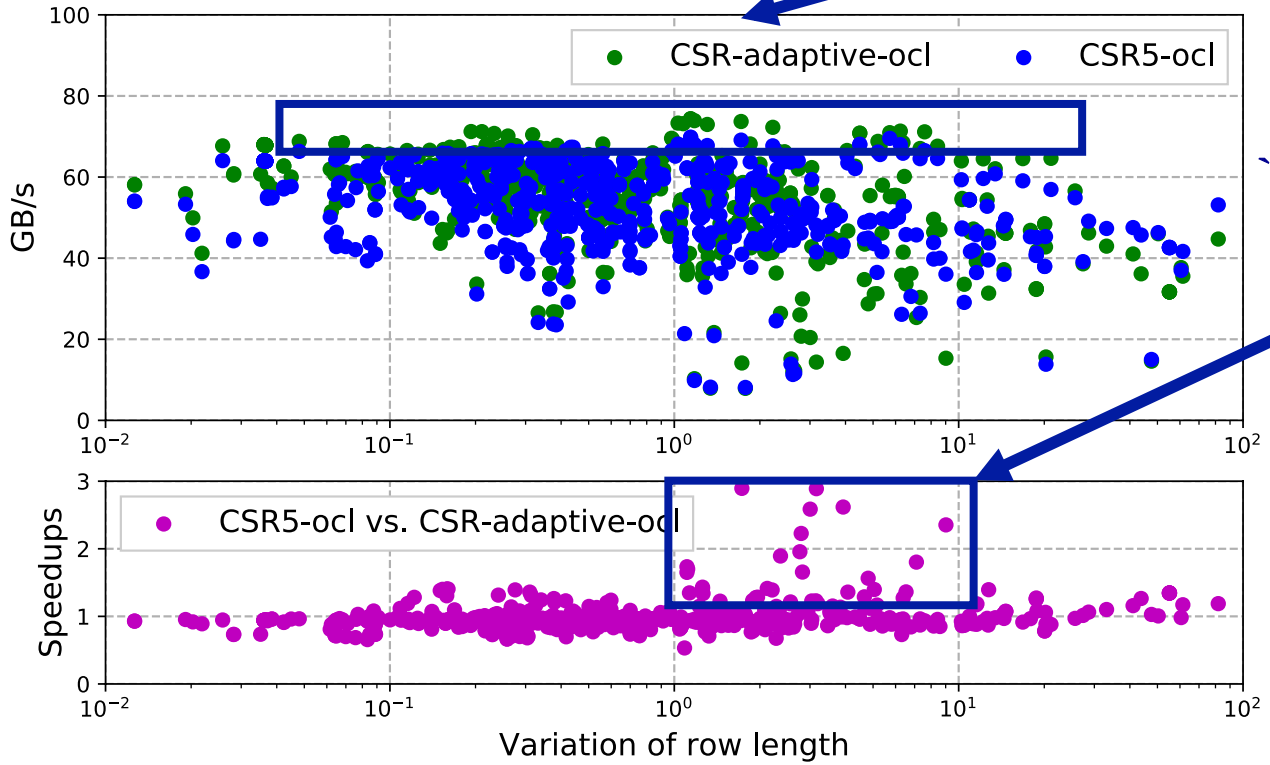
power-law dist.



NTNU

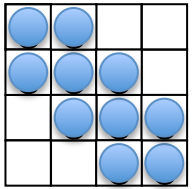
SpMV on GPU side

best `top' performance:
CSR-adaptive-ocl wins a bit

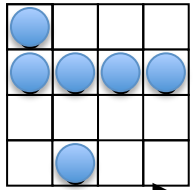


best
`load-balanced'
performance:
CSR5-ocl wins

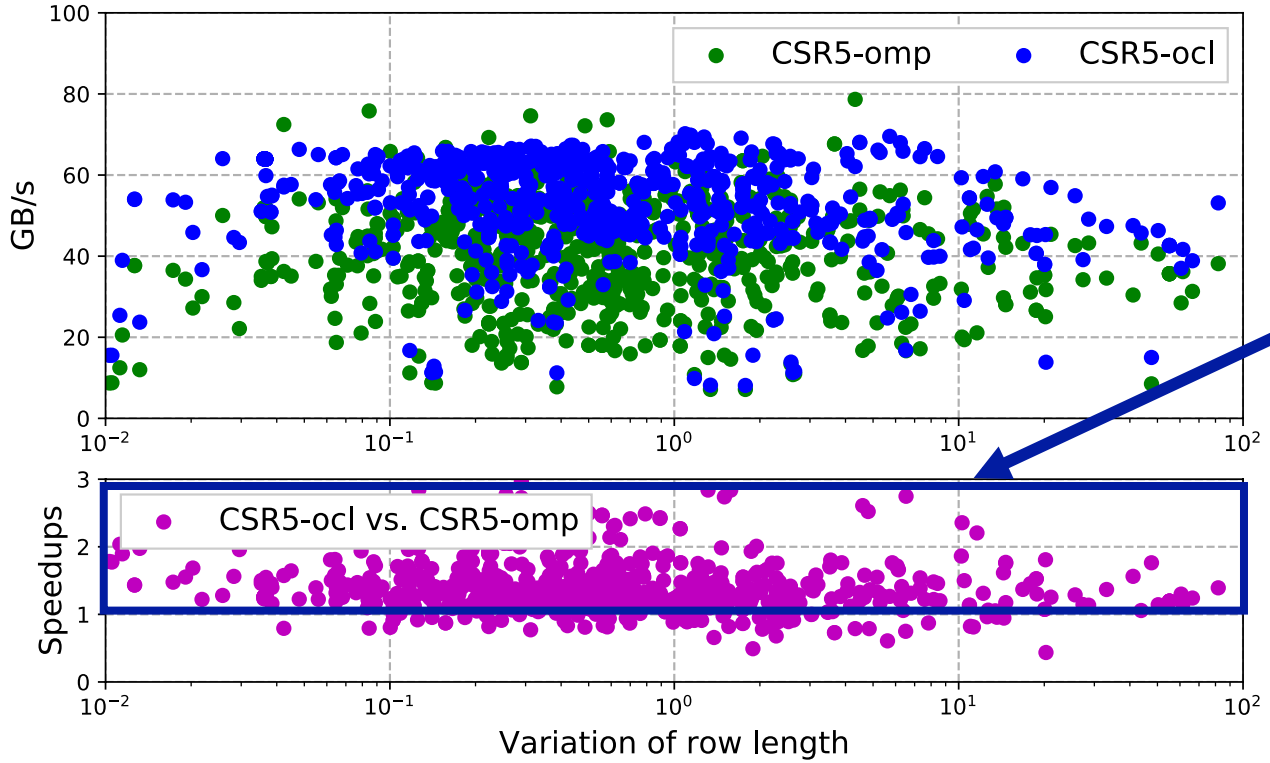
uniform
dist.



power-law
dist.



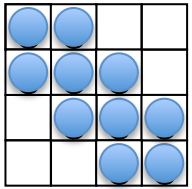
CSR5-SpMV (GPU vs. CPU)

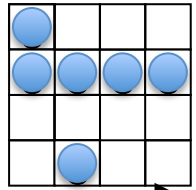


best `overall' performance: CSR5-ocl wins

uniform dist.

power-law dist.



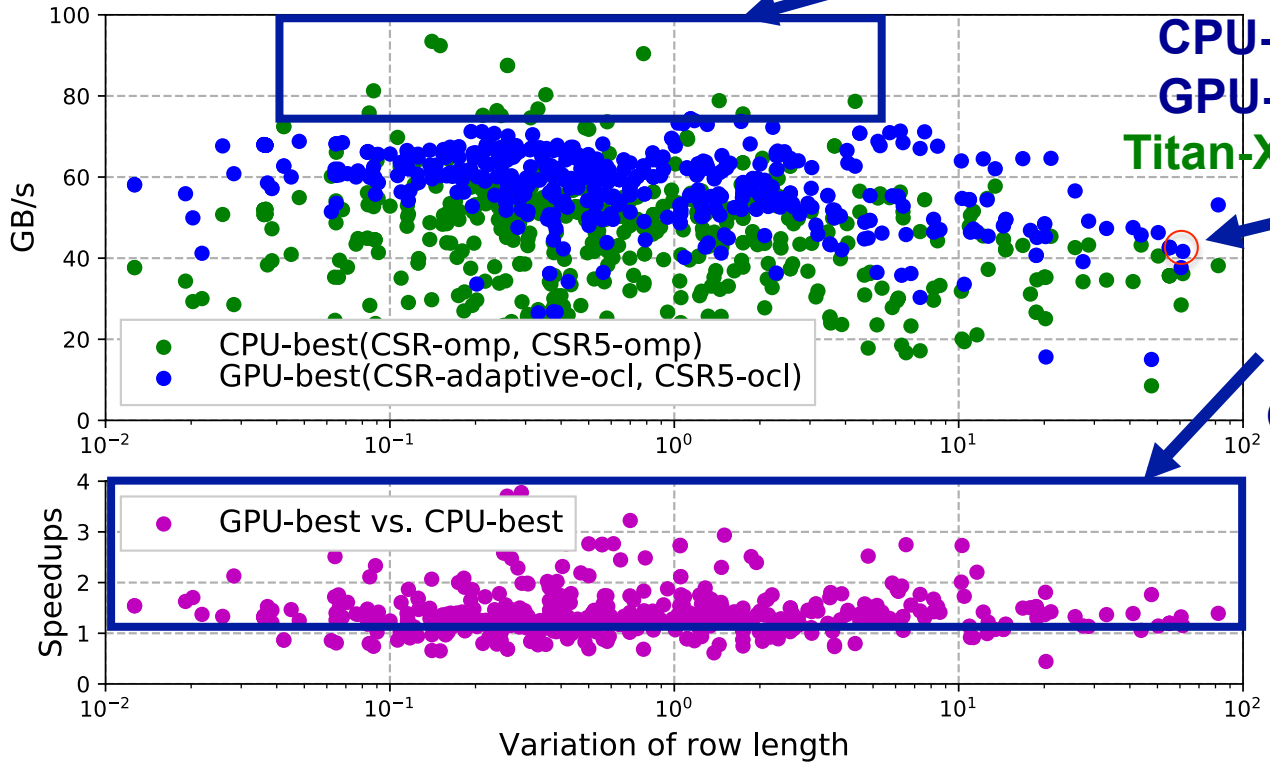


Best SpMV (GPU vs. CPU)

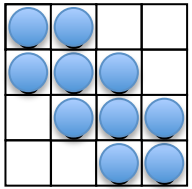
best `top' performance:
CPU-best wins

matrix `dc1' :
CPU-best: 35 GB/s
GPU-best: 42 GB/s
Titan-X CSR: 1.1 GB/s

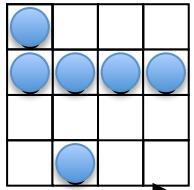
best `overall' performance:
GPU-best wins



uniform dist.



power-law dist.



NTNU

SpMV (Opportunity and Challenge)

Opportunities:

- Load balancing is still a problem on such `small' processors
- Load balanced methods on APU can largely outperforms naive code on GPU
- Load balanced methods can be slower in many cases
- CPU achieves higher `top' performance
- GPU achieves higher `overall' performance

Challenges:

- Understanding correlations between performance on real hardware and sparsity structures of sparse matrices
- Utilizing both CPU side and GPU side more efficiently

Kernel 2. Sparse Matrix-Matrix Multiplication (SpGEMM)

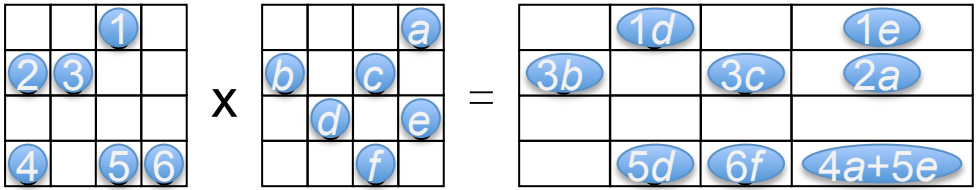
Question, Experiment, Opportunity and Challenge

Question 2: Is the unified memory (aka, shared virtual memory) helpful?

Experiment 2: SpGEMM running on GPU side but using the unified memory.

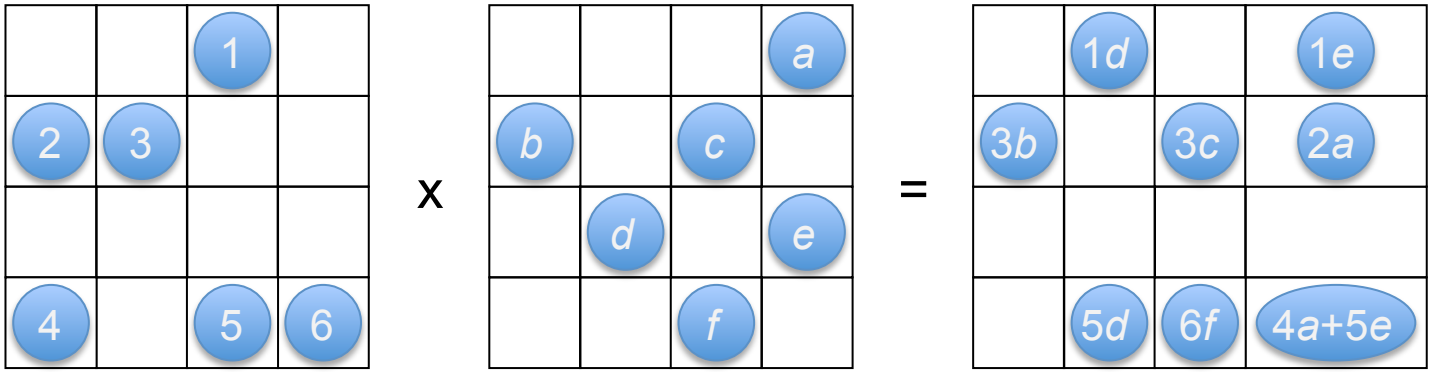
Opportunity 2 and Challenge 2

- Sparse matrix-matrix Multiplication (SpGEMM)



SpGEMM

- Multiply a sparse matrix *A* by a sparse matrix *B*, and obtain a resulting sparse matrix *C*.



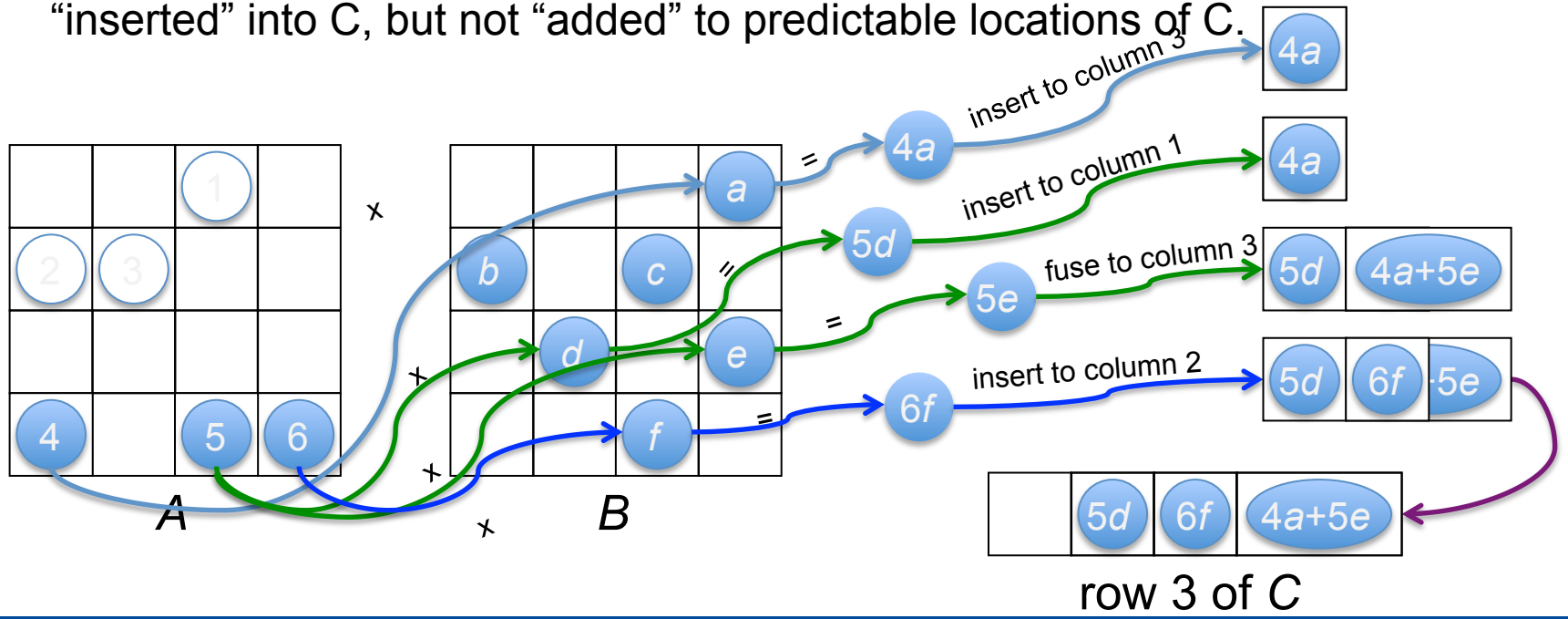
A (4x4) sparse, 6 nonzeros

B (4x4) sparse, 6 nonzeros

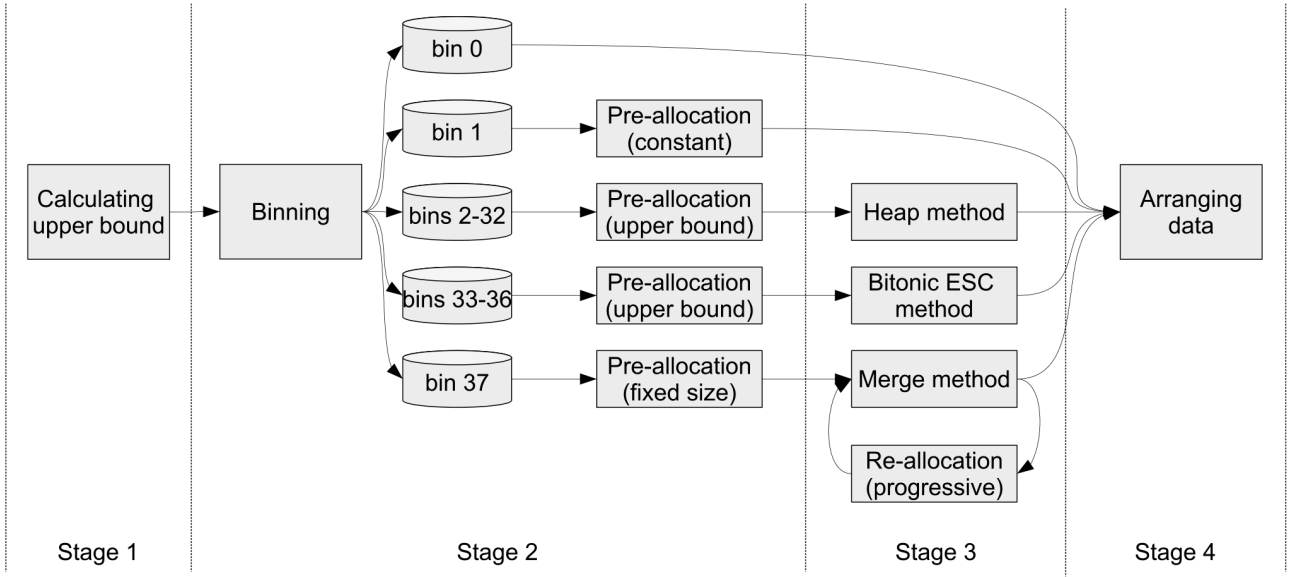
C (4x4) sparse, 8 nonzeros

SpGEMM - parallel insertion

- Because of the compressed sparse format, the result nonzeros are “inserted” into C, but not “added” to predictable locations of C.



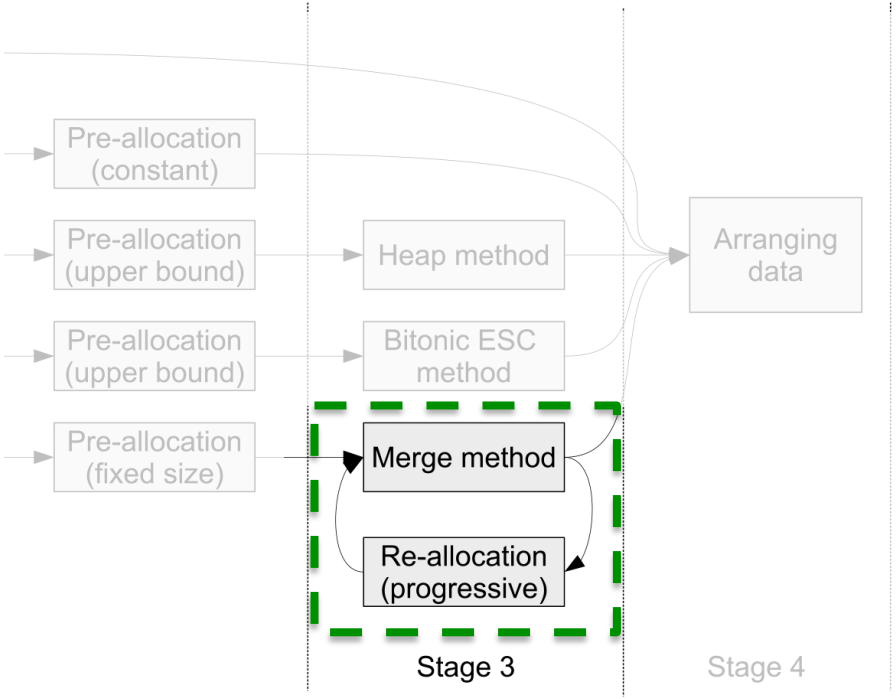
An SpGEMM framework (our work)

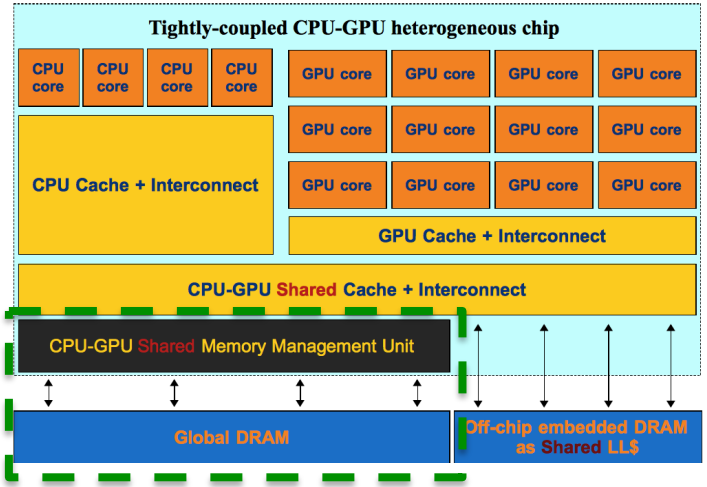


Weifeng Liu, Brian Vinter. **A Framework for General Sparse Matrix-Matrix Multiplication on GPUs and Heterogeneous Processors**. *JPDC. 2015.* (extended from *IPDPS14*)

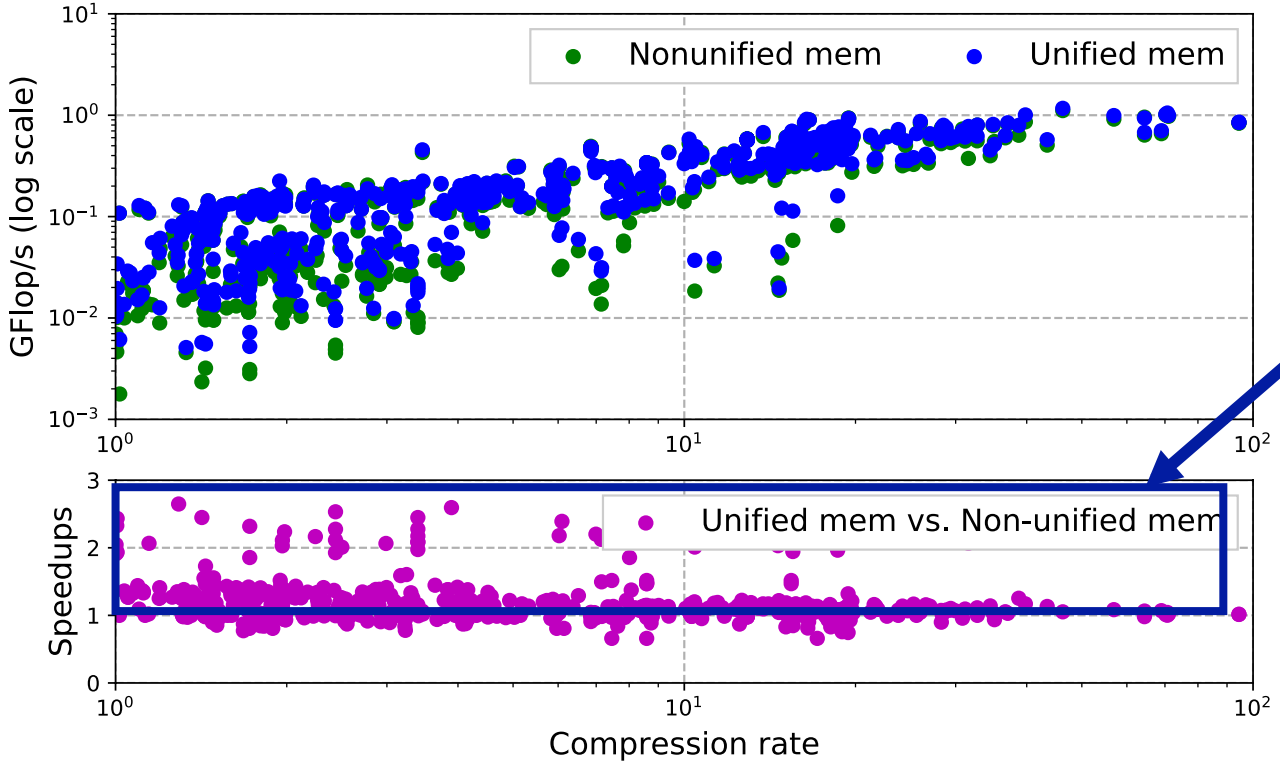
Weifeng Liu, Brian Vinter. **An Efficient GPU General Sparse Matrix-Matrix Multiplication for Irregular Data**. *IPDPS14*.

An SpGEMM framework (our work)





SpGEMM on GPU (Unified vs. Non-unified mem)



Obvious performance gain

more "insertion"

more "fusion"

SpGEMM (Opportunity and Challenge)

Opportunity:

- For GPU-friendly irregular problems with output of unknown size, using unified memory can bring higher performance.

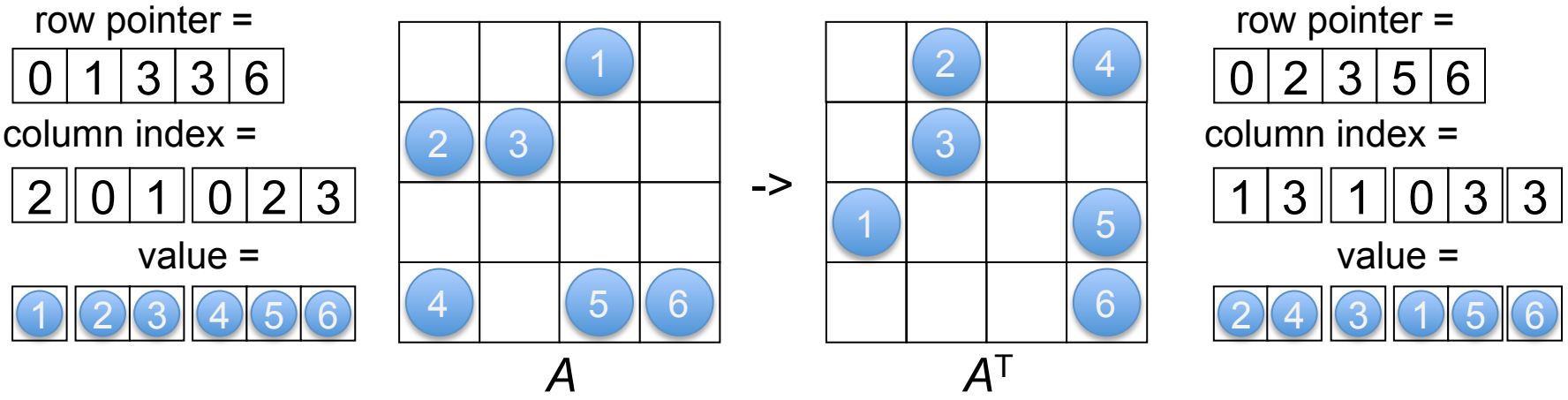
Challenge:

- Various memory configurations and deeper memory hierarchy need more careful tuning.

Kernel 3. Sparse Transposition (SpTRANS)

SpTRANS

- Transpose a sparse matrix *A* (in CSR) to a sparse matrix *B* (i.e., A^T, in CSR). This is equivalent to a conversion from CSR to CSC, or vice versa.



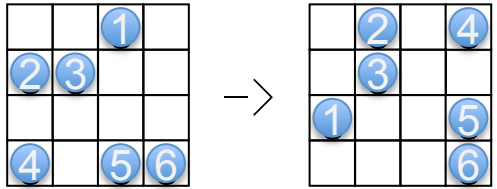
Question, Experiment, Opportunity and Challenge

Question 3: Can atomic operations work well?

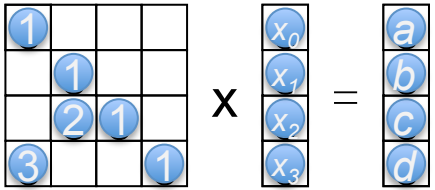
Experiment 3: SpTRANS and SpTRSV on GPU side.

Opportunity 3 and Challenge 3

- Sparse transposition (SpTRANS)

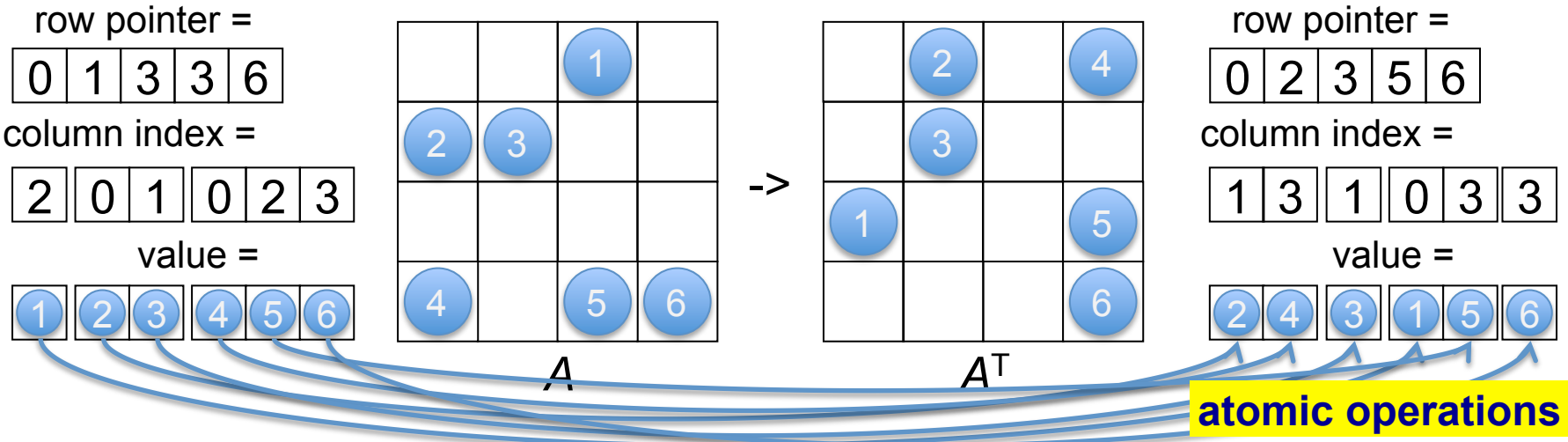


- Sparse triangular solve (SpTRSV)



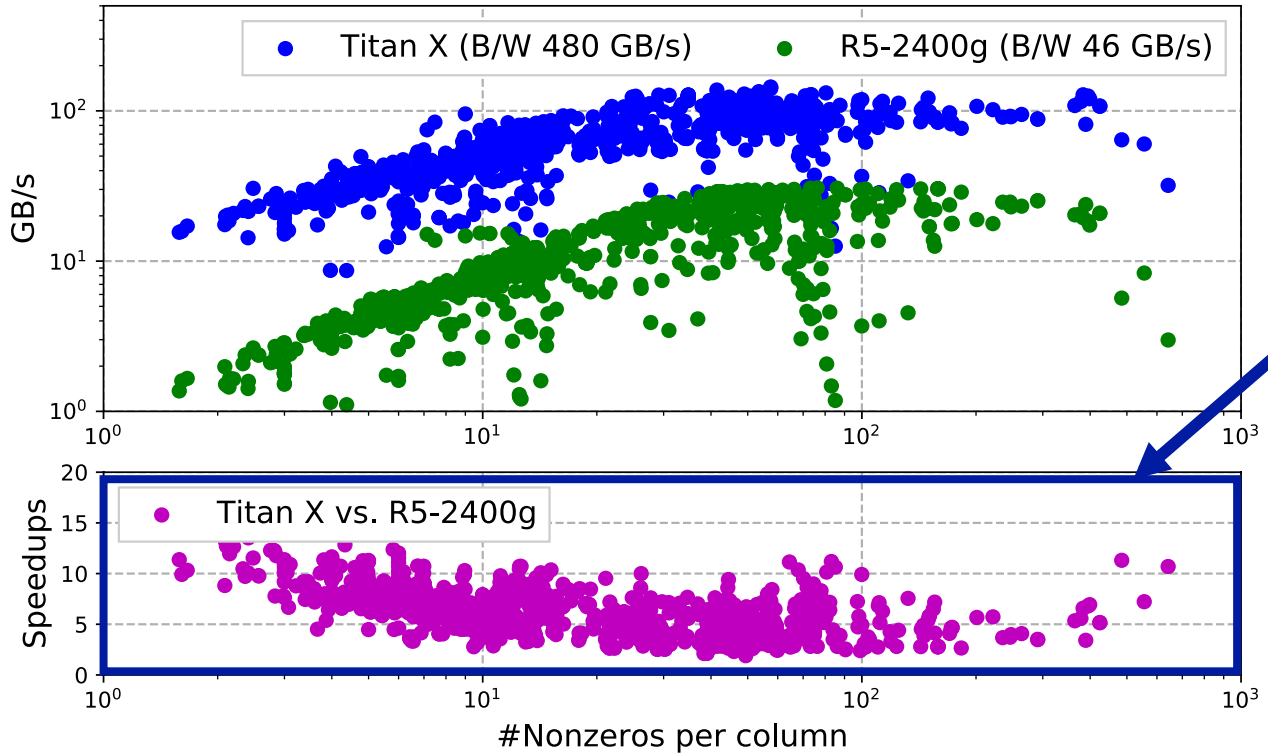
SpTRANS

- Transpose a sparse matrix *A* (in CSR) to a sparse matrix *B* (i.e., A^T, in CSR). This is equivalent to a conversion from CSR to CSC, or vice versa.



Hao Wang, Weifeng Liu, Kaixi Hou, Wu-chun Feng. **Parallel Transposition of Sparse Data Structures**. *ICS16*.

Performance on GPU (dGPU vs. iGPU)

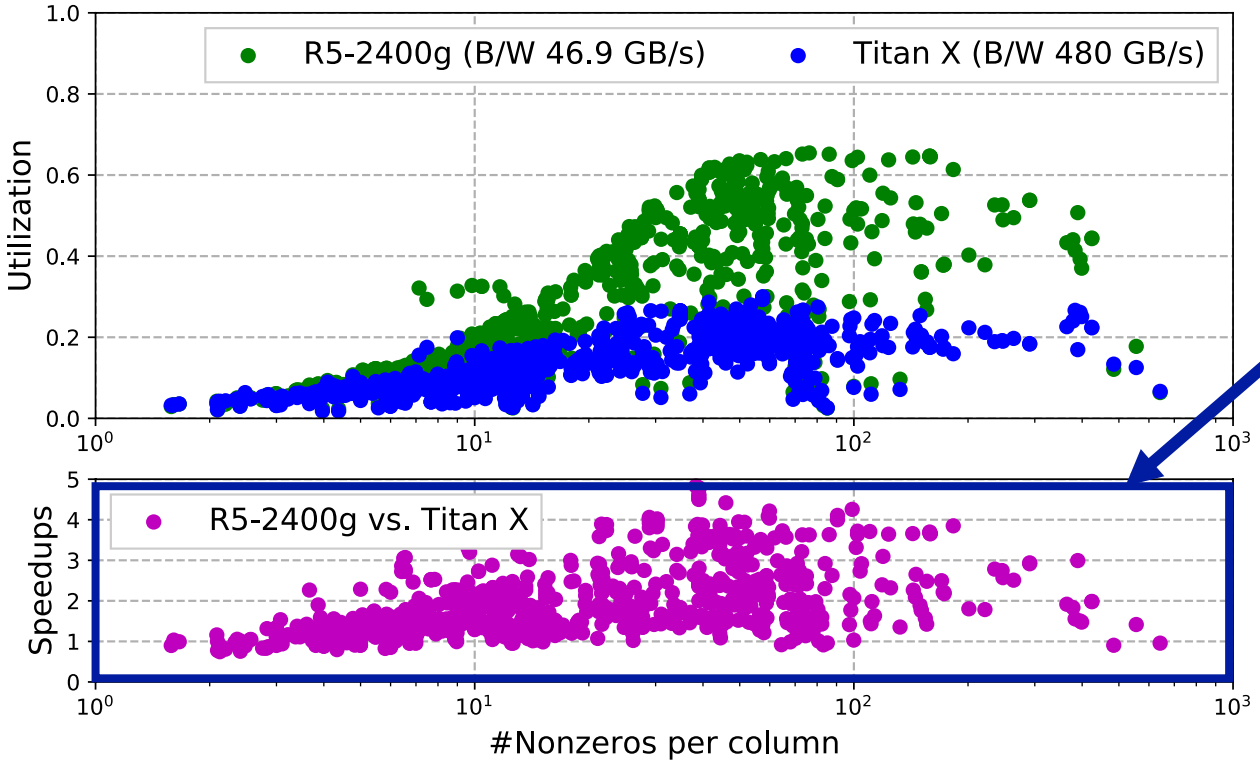


dGPU has much higher performance

more "sparse"

more "dense"

B/W utilization on GPU (iGPU vs. dGPU)



iGPU has much higher bandwidth utilization

more "sparse"

more "dense"

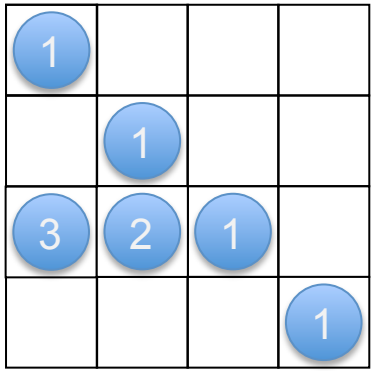
NTNU

Kernel 4. Sparse Triangular Solve

(SpTRSV)

SpTRSV

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



L
(4x4)
nnzL = 6
known

x

x
(4x1)
dense
unknown

=

b
(4x1)
dense
known

system:

$1*x_0 = a$
$1*x_1 = b$
$3*x_0+2*x_1+1*x_2 = c$
$1*x_3 = d$

solution:

$x_0 = a$
$x_1 = b$
$x_2 = c - 3a - 2b$
$x_3 = d$

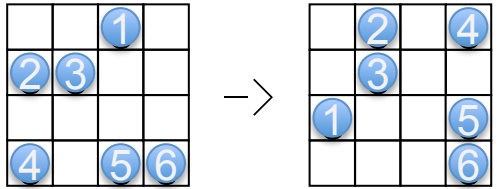
Question, Experiment, Opportunity and Challenge

Question 3: Can atomic operations work well?

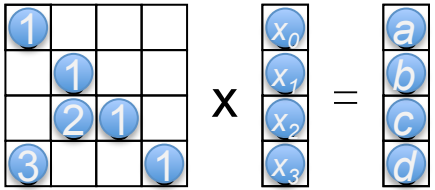
Experiment 3: SpTRANS and SpTRSV on GPU side.

Opportunity 3 and Challenge 3

- Sparse transposition (SpTRANS)

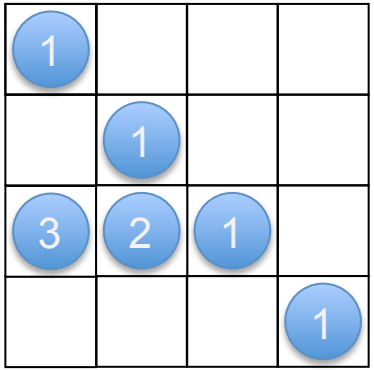


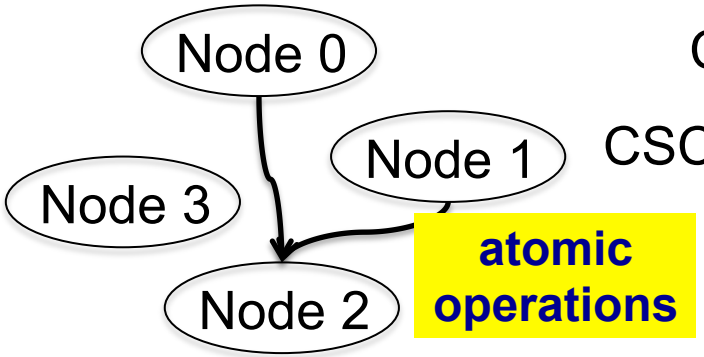
- Sparse triangular solve (SpTRSV)



Sync-free method for SpTRSV (our work)

- All cores (one for each column as a node) are activiated: some of them compute solution, the others busy-wait for spin-locks unlocked to start.





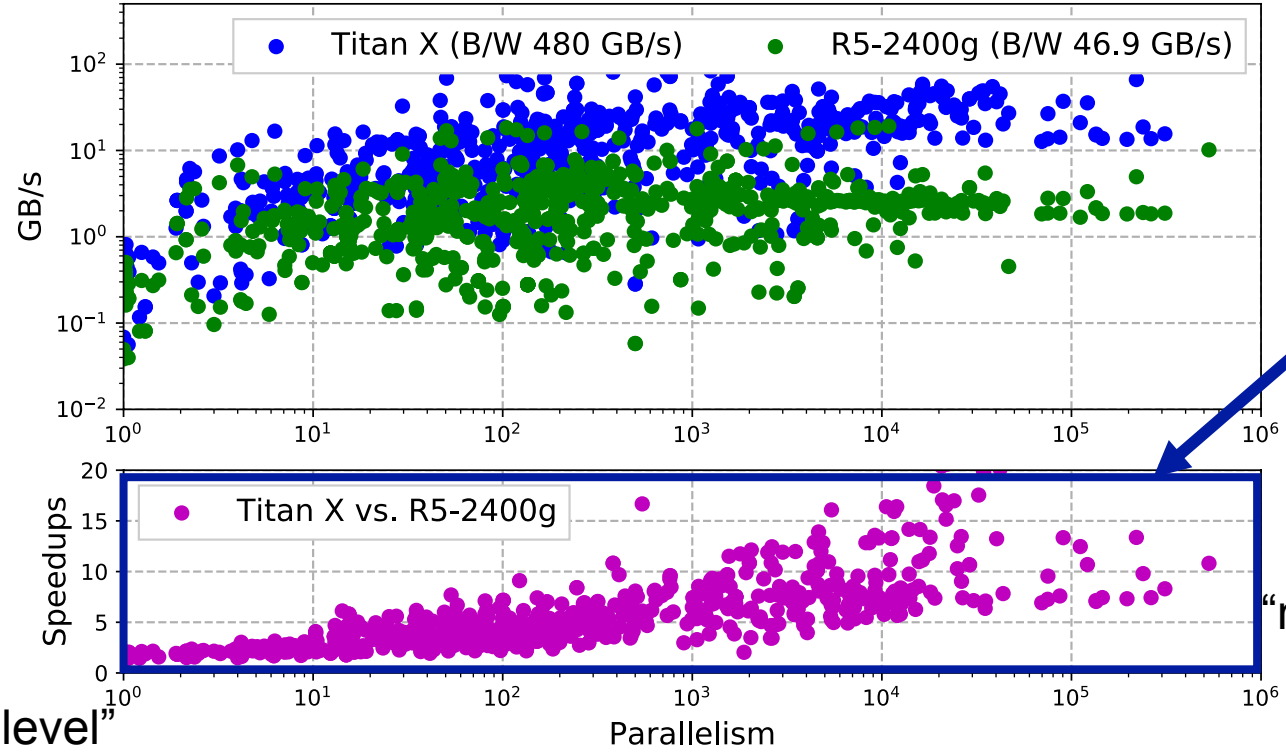
CSR row_ptr:	0	1	2	5	6
CSC column_ptr:	0	2	4	5	6

In-degree:	1	1	3	1
Out-degree:	2	2	1	1

Weifeng Liu, Ang Li, Jonathan Hogg, Iain S. Duff, Brian Vinter. **Fast Synchronization-Free Algorithms for Parallel Sparse Triangular Solves with Multiple Right-Hand Sides**. *CCPE. 2017.* (extended from *Europar16*)

Weifeng Liu, Ang Li, Jonathan Hogg, Iain S. Duff, Brian Vinter. **A Synchronization-Free Algorithm for Parallel Sparse Triangular Solves**. *Europar16*.

Performance on GPU (dGPU vs. iGPU)



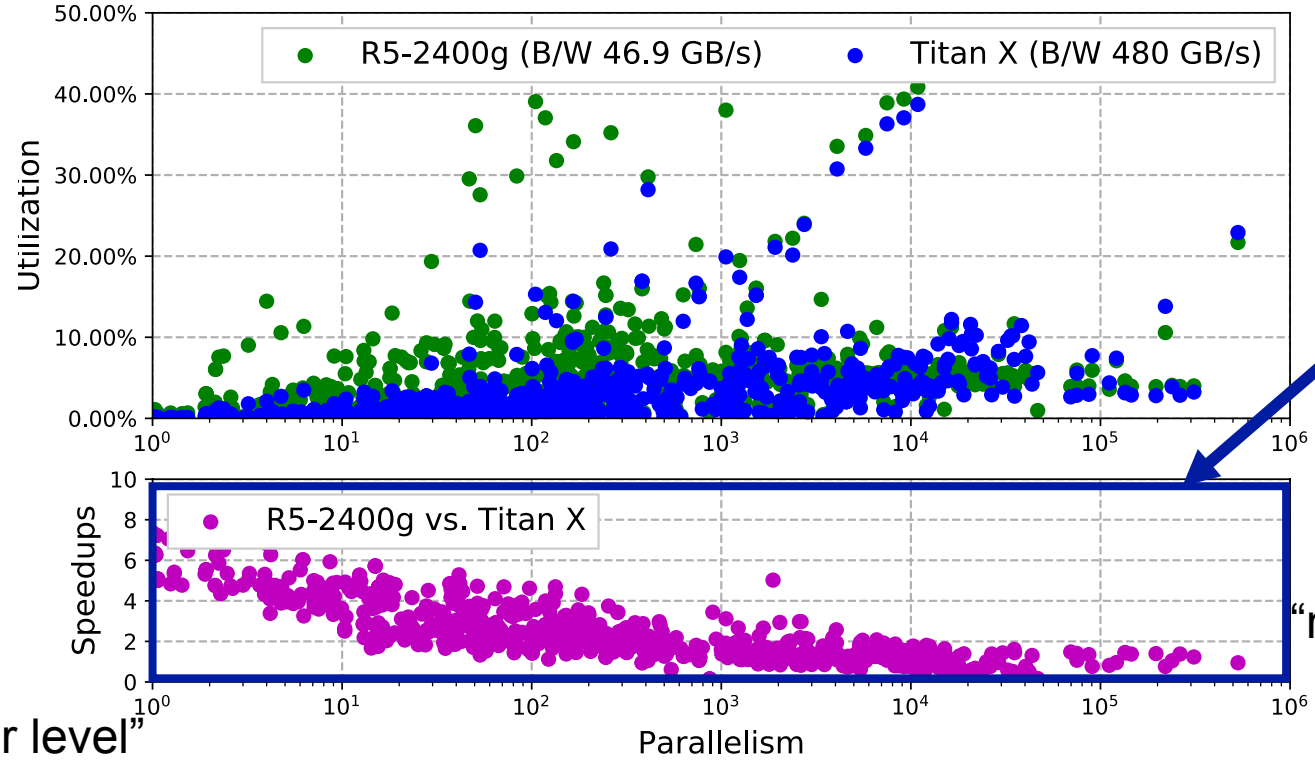
dGPU has much higher performance

more "levels" less "nodes per level"

more "nodes per level" less "levels"

NTNU

B/W utilization on GPU (iGPU vs. dGPU)



iGPU has much higher bandwidth utilization

more "levels" less "nodes per level"

more "nodes per level" less "levels"

NTNU

SpTRANS and SpTRSV (Opportunity and Challenge)

Opportunity:

- In terms of utilization, using more atomic operations looks more promising on APUs, though overall performance is not competitive over dGPU yet.

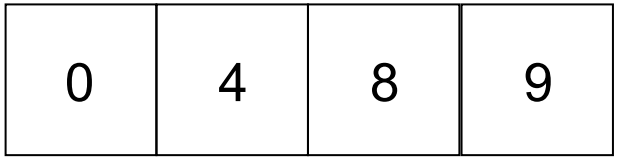
Challenge:

- Possibly converting/eliminating more (global) synchronizations to atomic operations for overall higher performamce.

Conclusion

- The emergement of heterogeneous processors bring larger algorithm design and tuning space.
- Based on a large set of experimental results, some opportunities are observed.
- New exciting challenges are coming.

T k u !



A y Q s n s ?