HIGH PERFORMANCE UNSTRUCTURED SPMM COMPUTATION USING TENSOR CORES

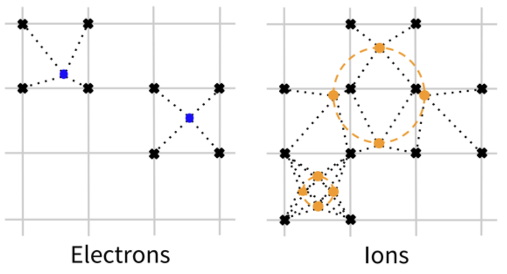
Patrik Okanovic[1], Grzegorz Kwasniewski[1], Paolo Sylos Labini[2], Maciej Besta[1], Flavio Vella[3], Torsten Hoefler[1]

[1] ETH Zurich

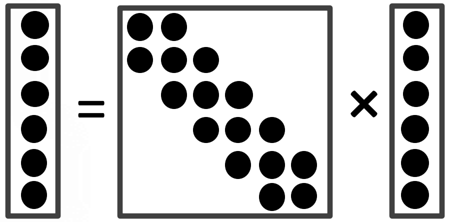
[2] Free University of Bozen-Bolzano

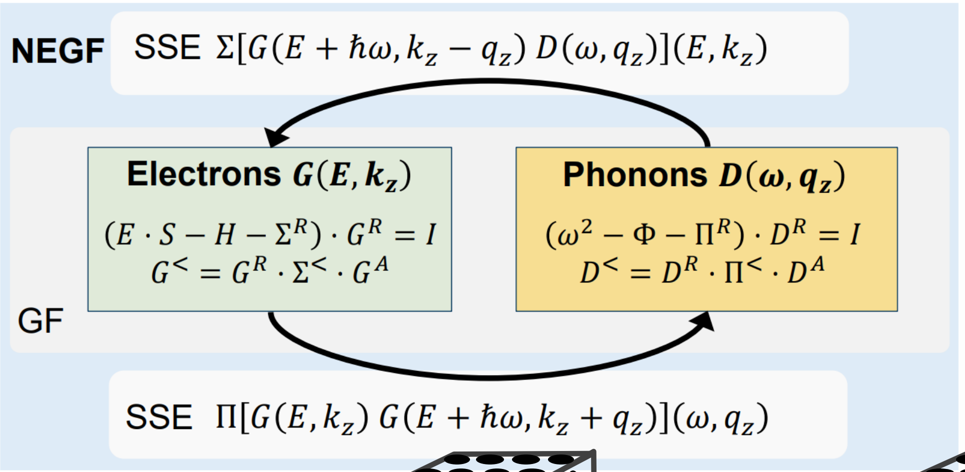
[3] University of Trento

SC24 Atlanta, GA | hpc creates.

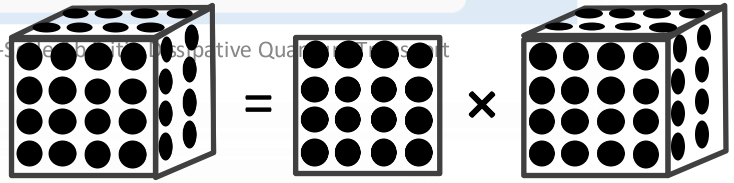


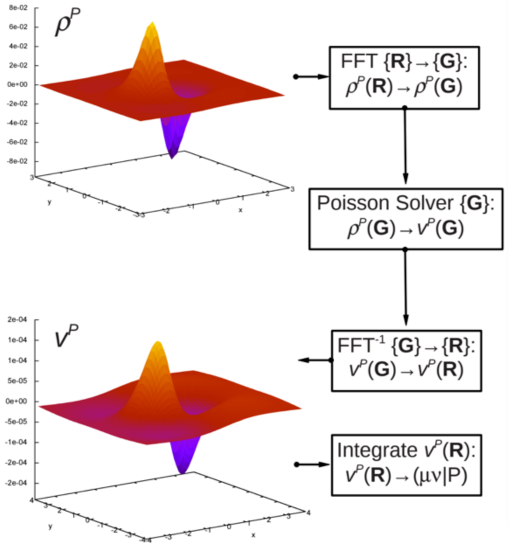
Extreme Scale Plasma Turbulence Simulations on Top Supercomputers Worldwide, Tang et al.



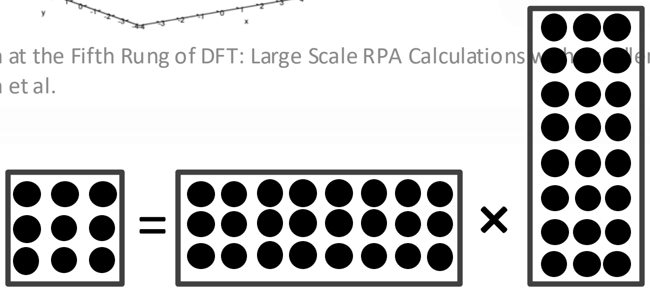


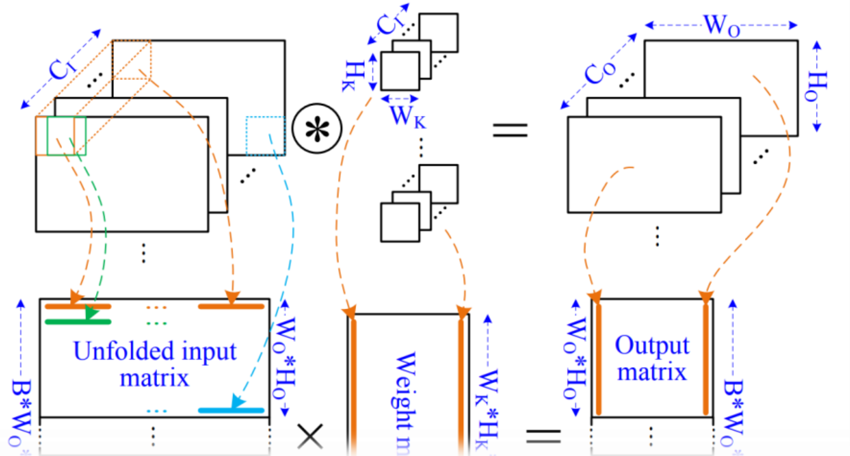
A Data-Centric Approach to Extreme-S… …ative Qua… …t Simulations, Ziogas et al.

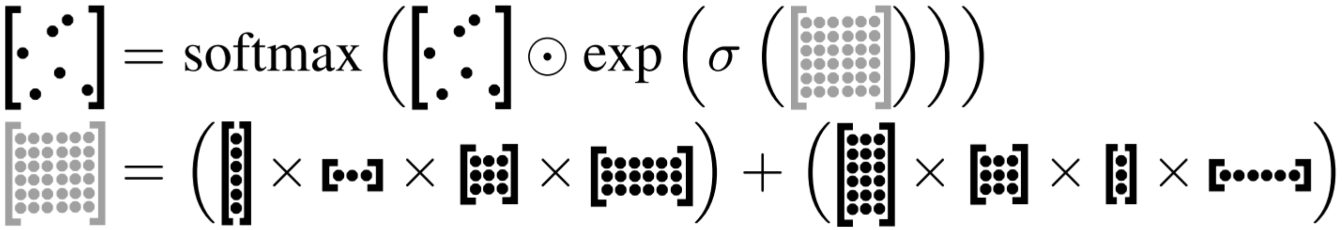




Enabling Simulation at the Fifth Rung of DFT: Large Scale RPA Calculations w… …ent Time to Solution, Del Ben et al.



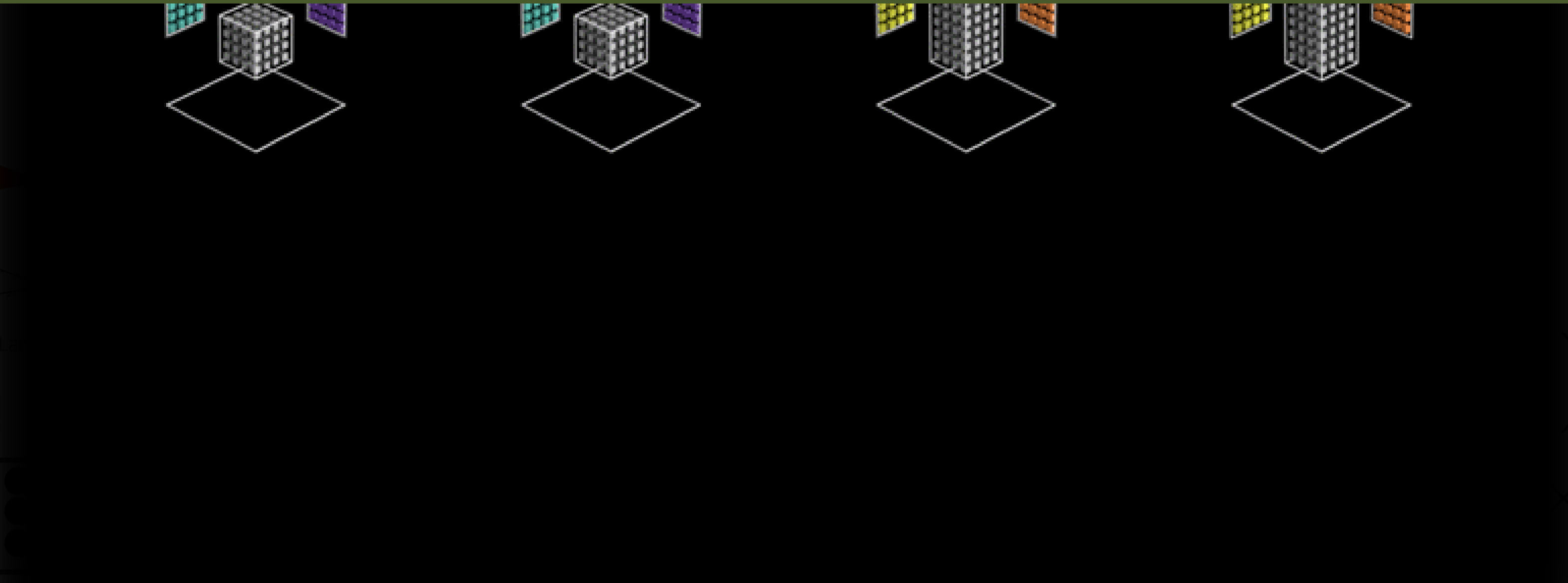


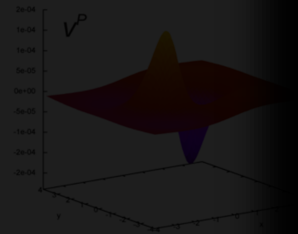


NEGF SSE $\Sigma[G(E+\hbar\omega, k_z - q_z)\, D(\omega, q_z)](E, k_z)$

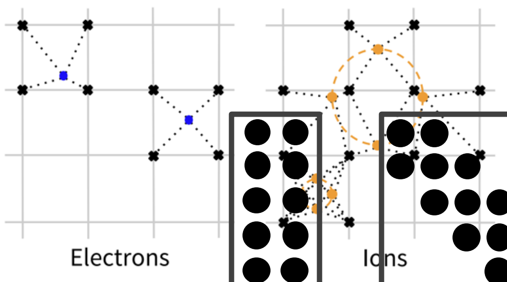
Ideal for hardware!

...All done?...

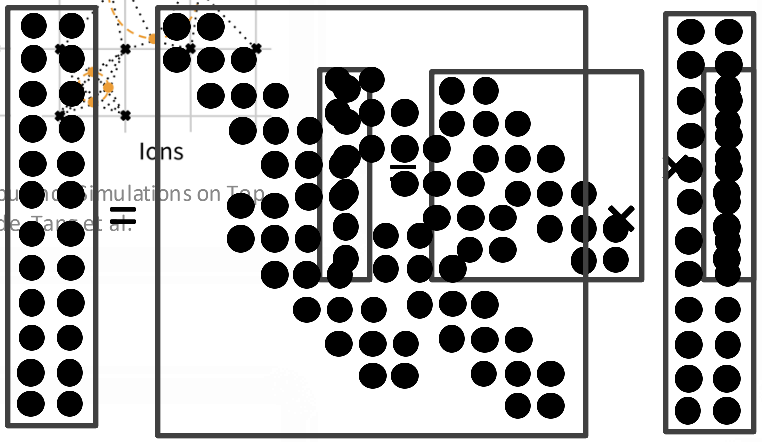


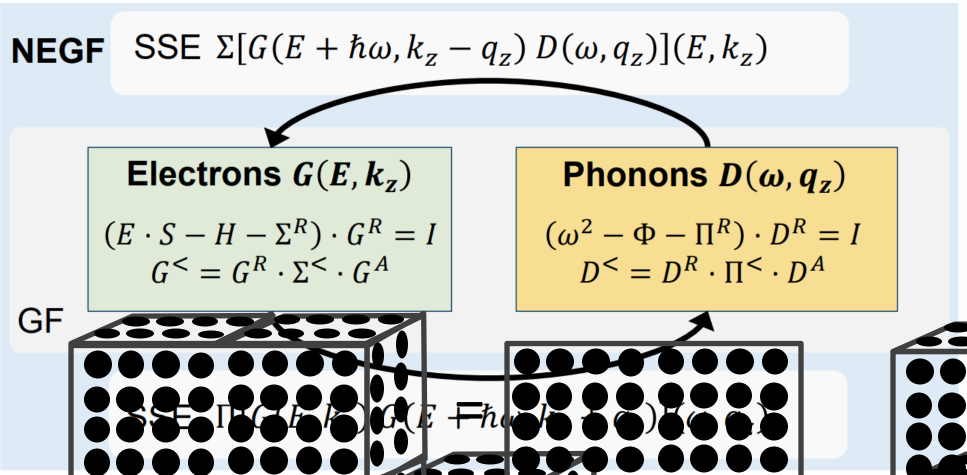


Enabling Simulation at the Fifth Rung of DFT: ...
to Solution, Del Ben et al.

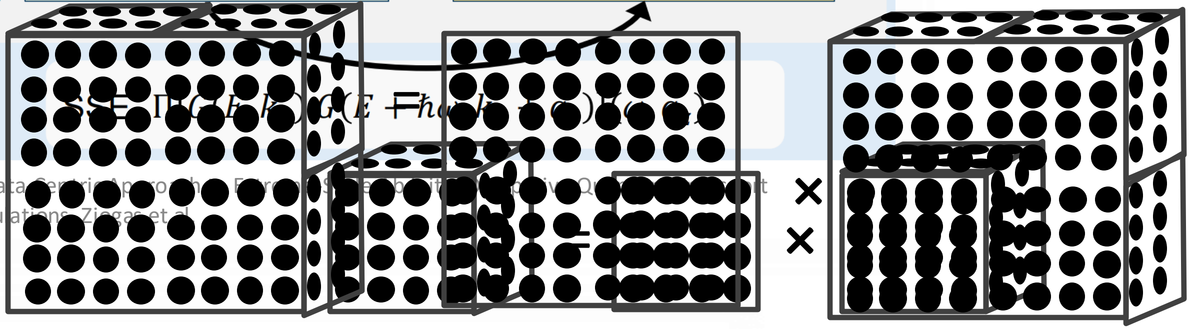


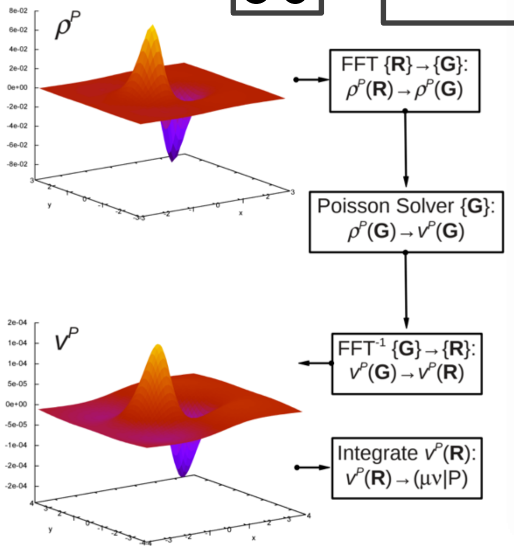
Extreme Scale Plasma Turbulence Simulations on Top Supercomputers Worldwide, Tang et al.



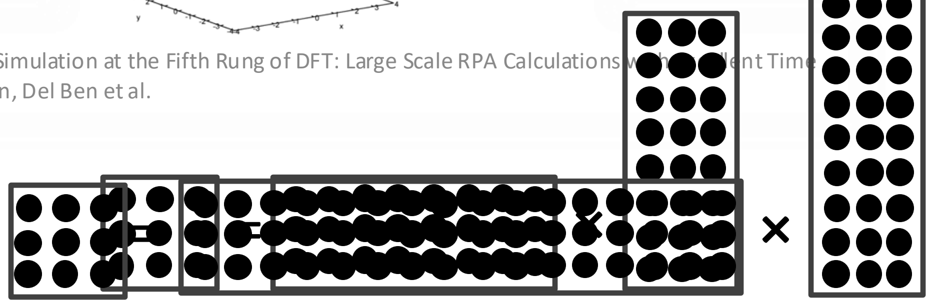


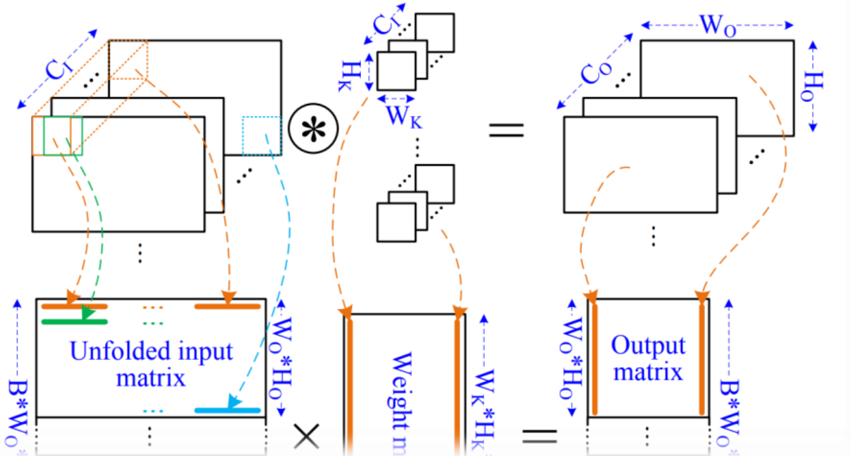
A Data-Centric Approach to Extreme-Scale Ab initio Dissipative Quantum Transport Simulations, Ziogas et al.

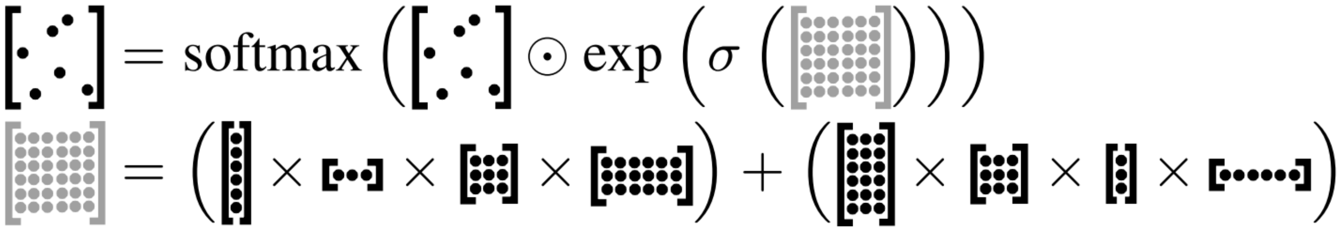




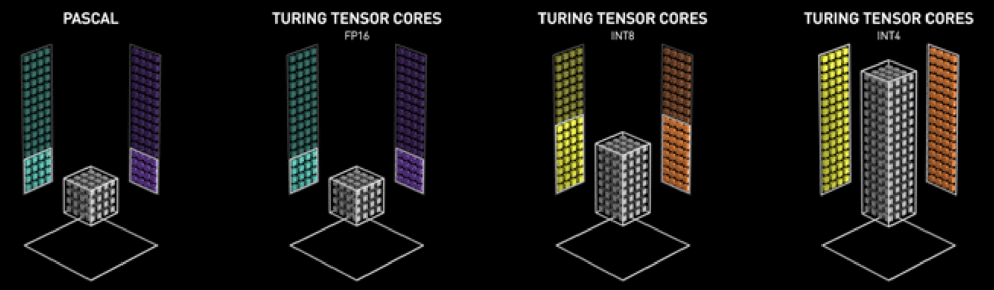
Enabling Simulation at the Fifth Rung of DFT: Large Scale RPA Calculations with Excellent Time to Solution, Del Ben et al.



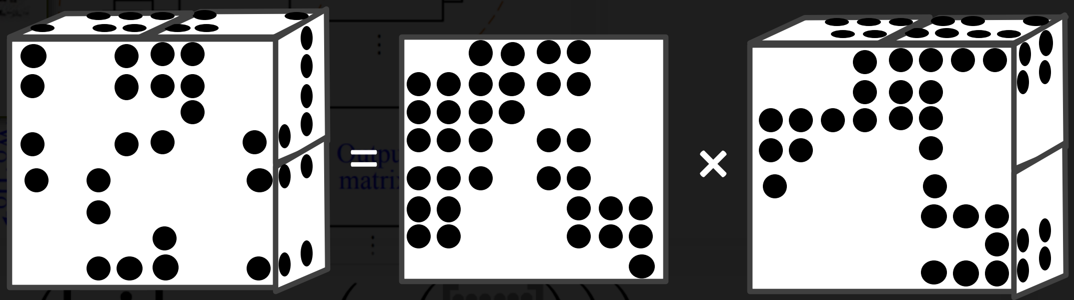




Hardware requirement:
Dense matrices



Problems to solve:
Large, sparse data structures



…but… we do have solutions… right?

Libraries exist!
- *cuSPARSE*
- *hipSPARSE*

Hardware exists!
- *Sparse Tensor Cores*

Dedicated solutions exist!
- *DASP [2]*
- *Magicube [3]*
- *VENOM [4]*
- *cuSPARSElLt [5]*
- …

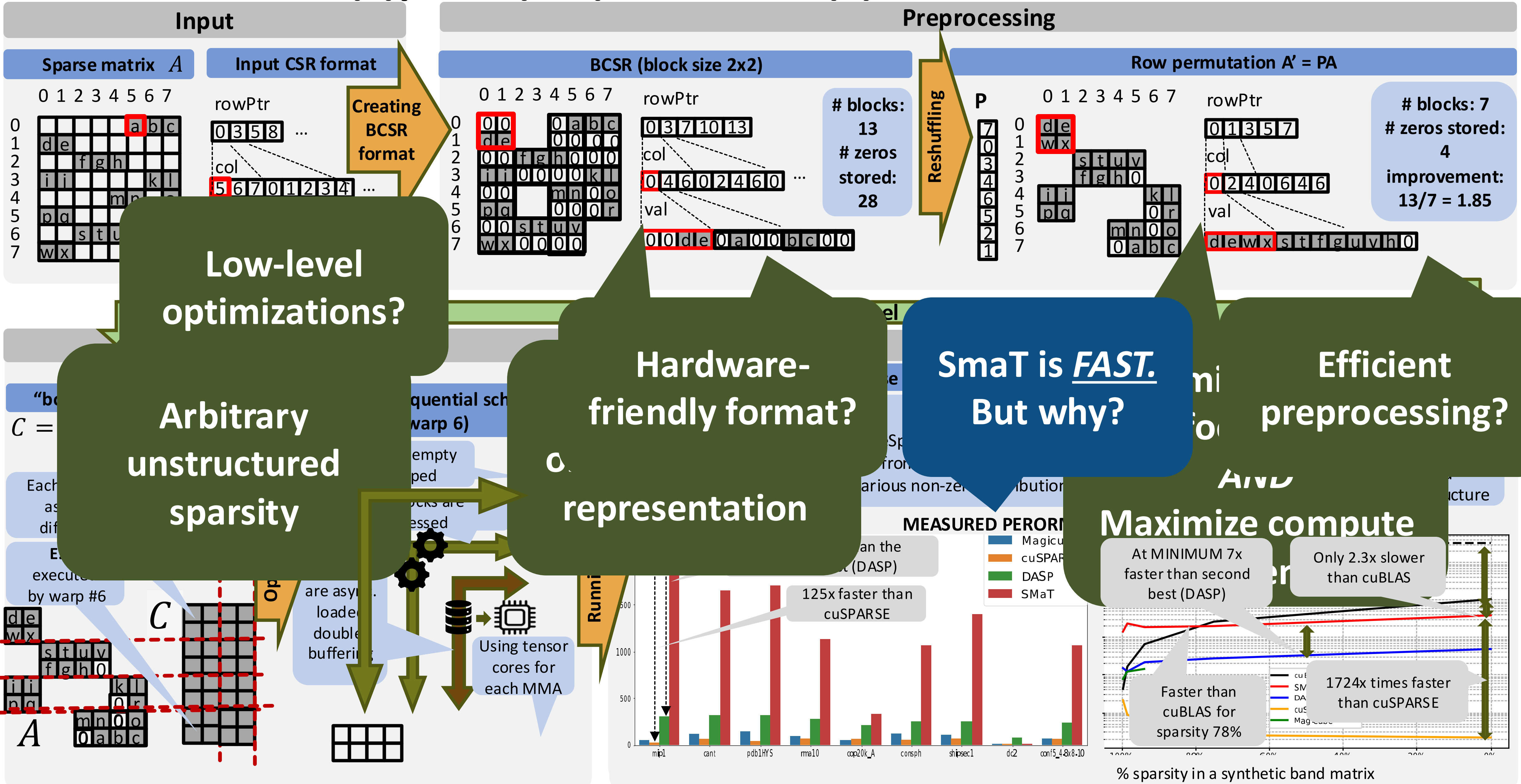
cuSPARSE [1]:
Up to 73,350
(73 thousand!)
times slower than the achievable peak!

Sparsity requirements:
2:4
EVERY *four elements*
EXACTLY *two nonzeros*

Lacking performance:
At least 200x slower than the peak
Narrow applicability:
Only machine learning, "dense" sparse matrices (up to 80-90% sparsity)

[1] cuSPARSE v12.0 vs CUBLAS c12.0 on dense matrices, NVIDIA A100

SMaT: (S)parse (Ma)trix Matrix (T)ensor Core-accelerate



Performance Model

$$T_{tot} = T_e \cdot n_e + T_{init}$$

Total runtime of the kernel

Compute time per the BCSR block

Number of dense blocks in the BCSR format

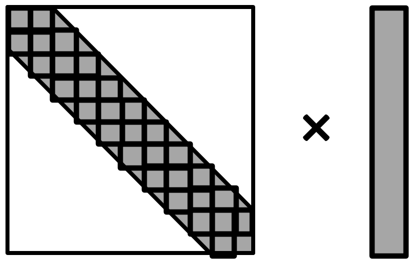
Any startup, initialization, cache warm-up

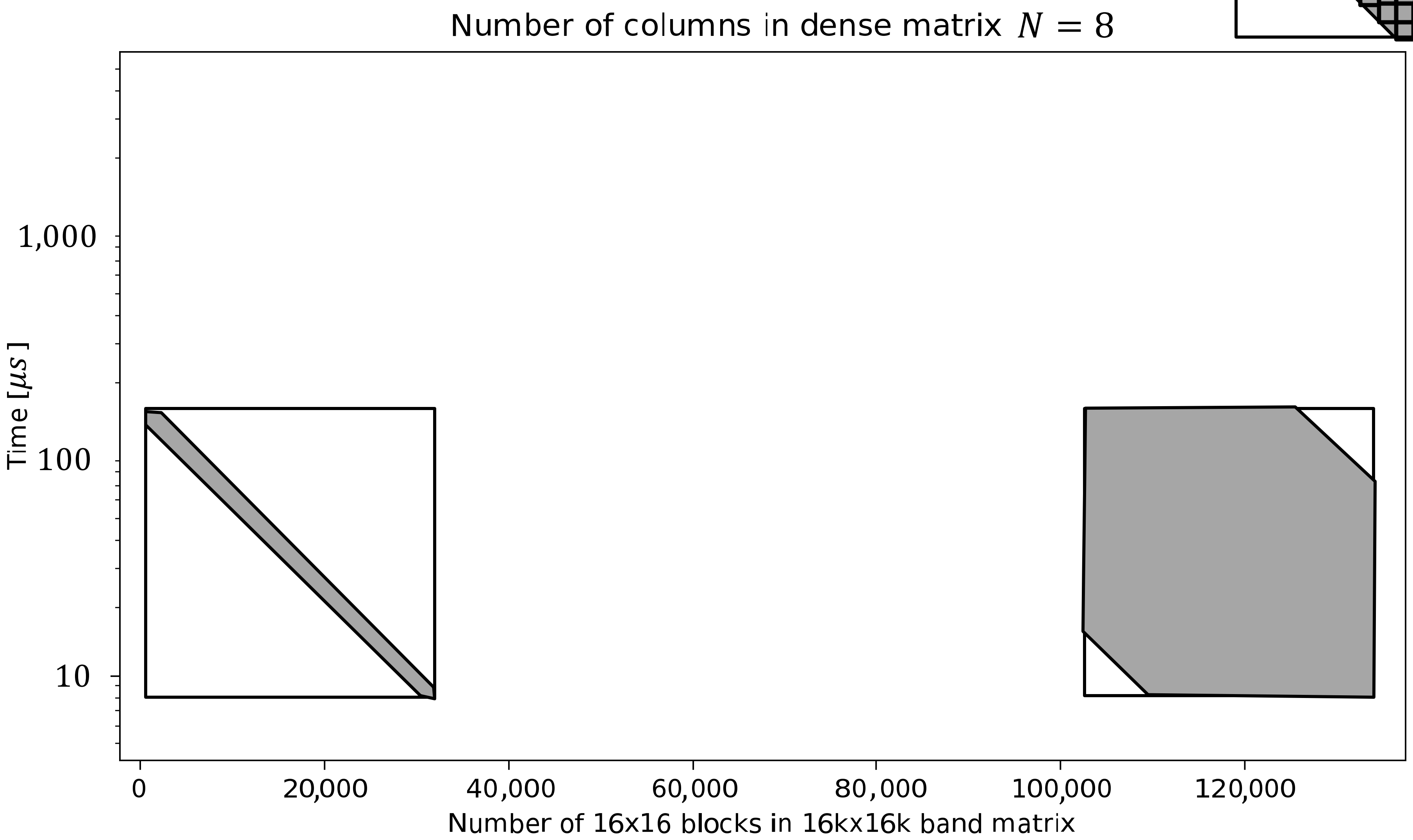
Implementation optimizations: Reducing the time per block

Preprocessing: Reducing the number of blocks

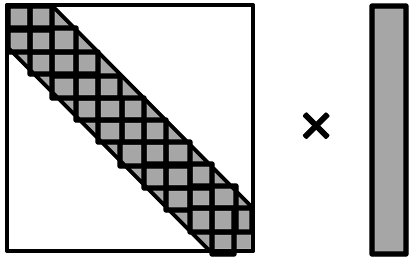
Performance Model

$$T_{tot} = T_e \cdot n_e + T_{init}$$

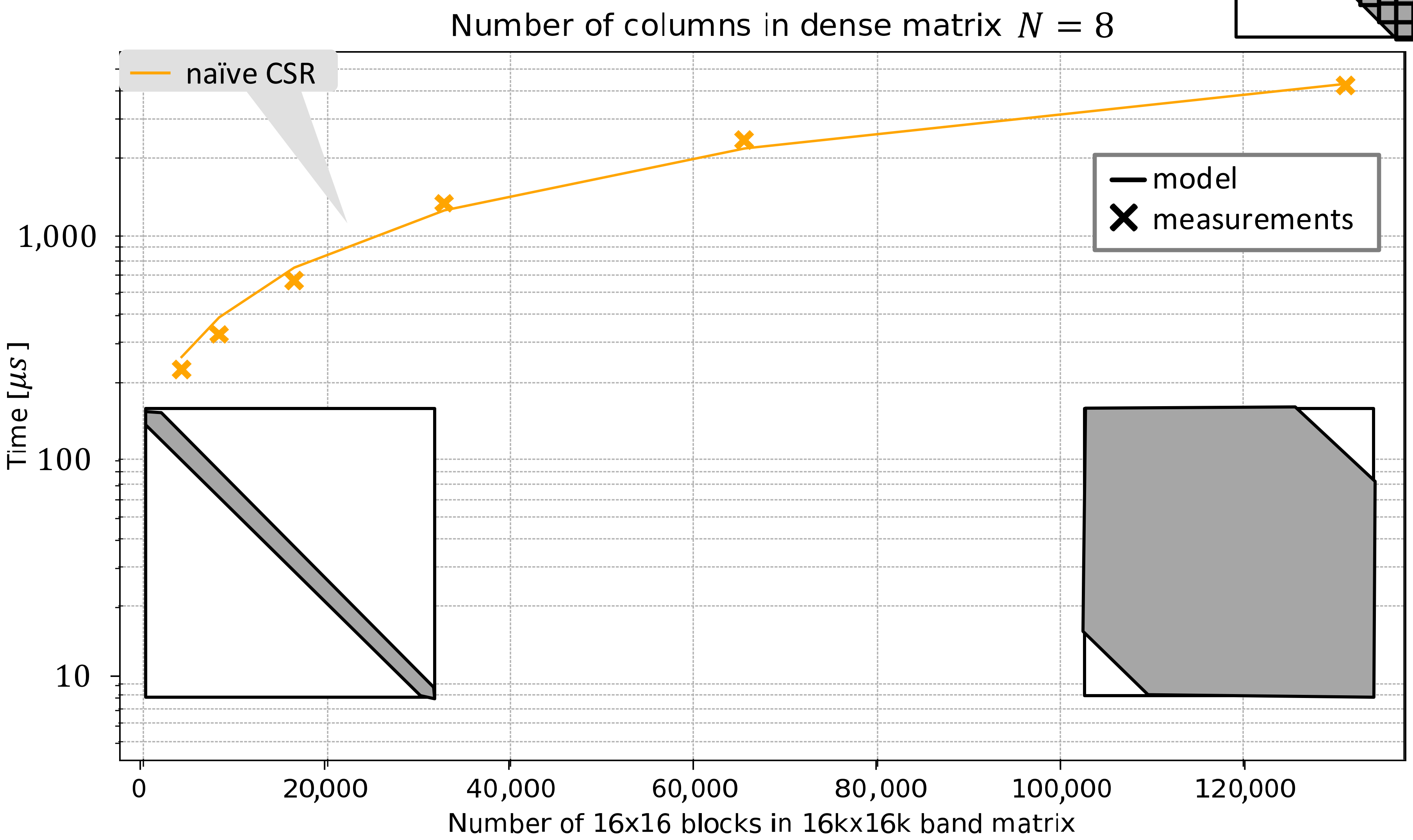




Performance Model

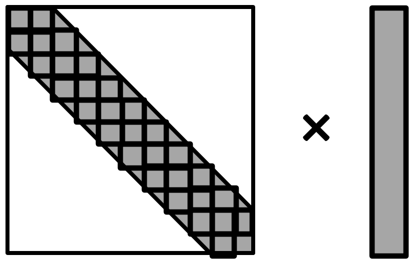


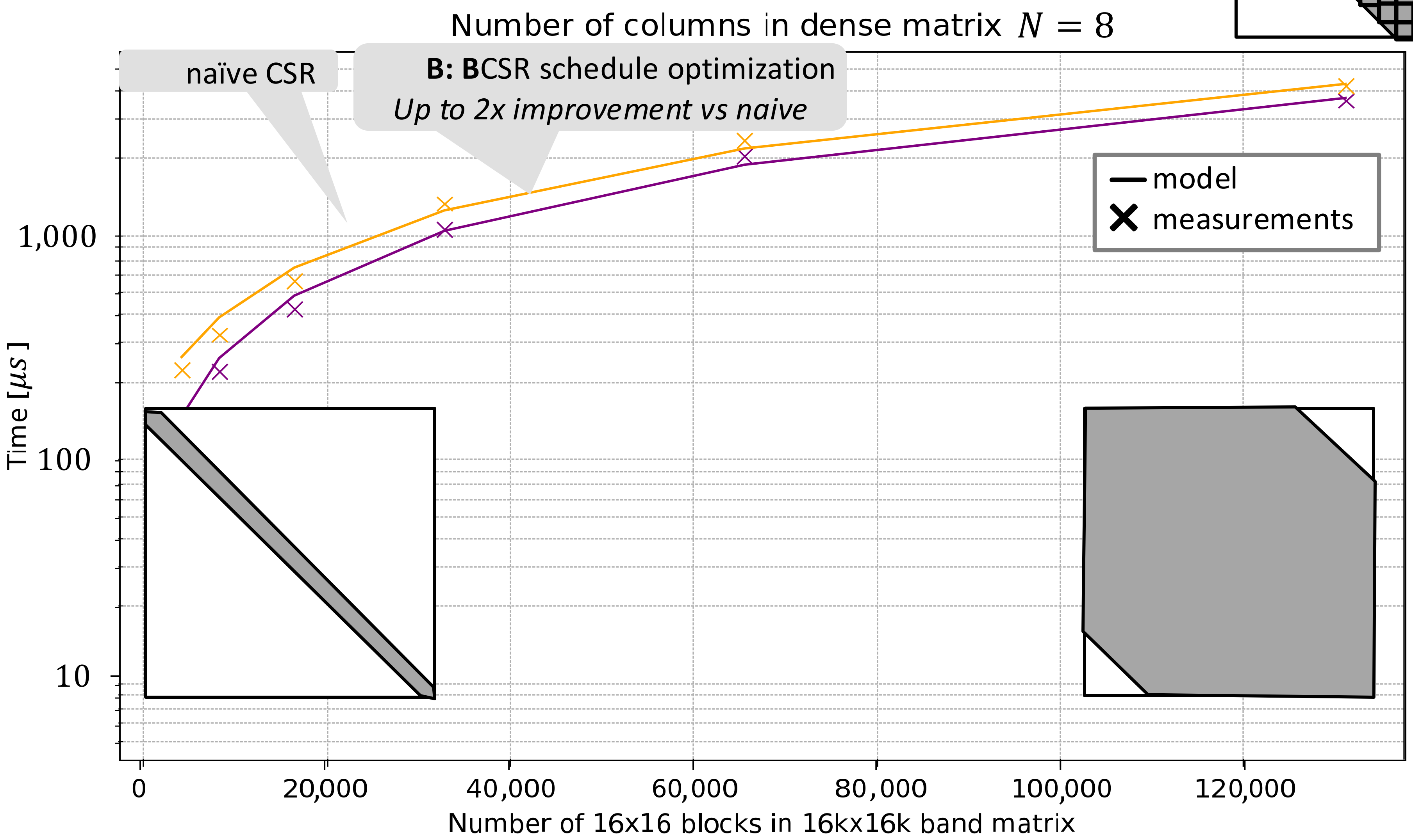
$$T_{tot} = T_e \cdot n_e + T_{init}$$



Performance Model

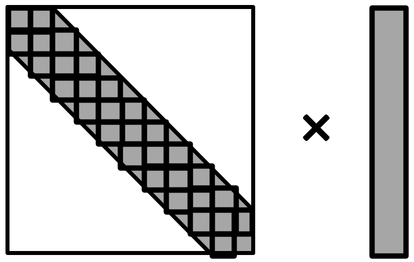
$$T_{tot} = T_e \cdot n_e + T_{init}$$

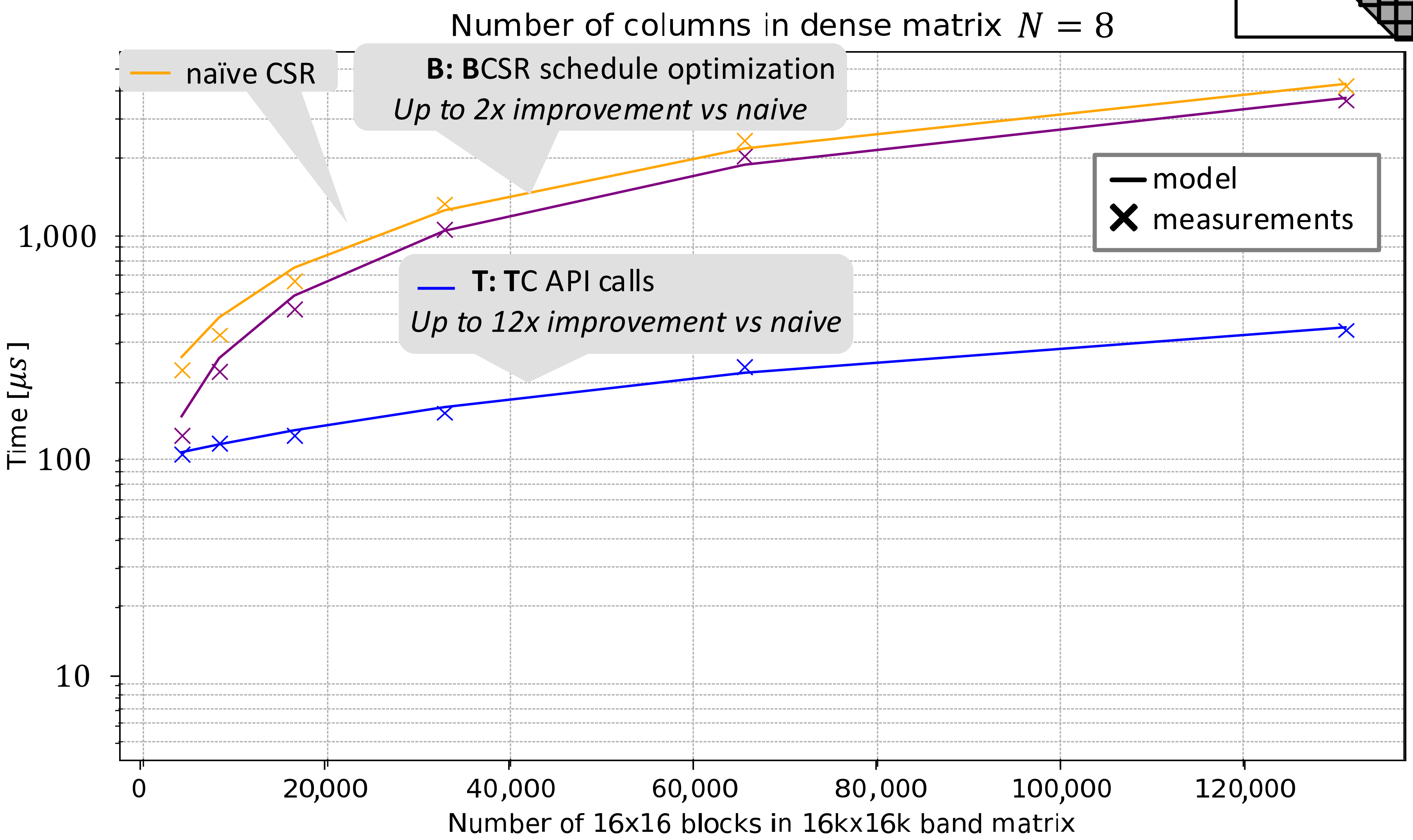




Performance Model

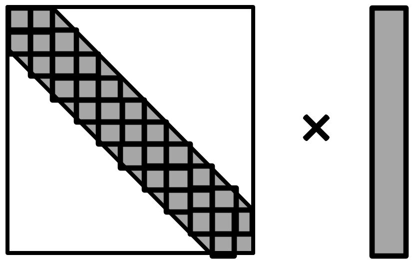
$$T_{tot} = T_e \cdot n_e + T_{init}$$

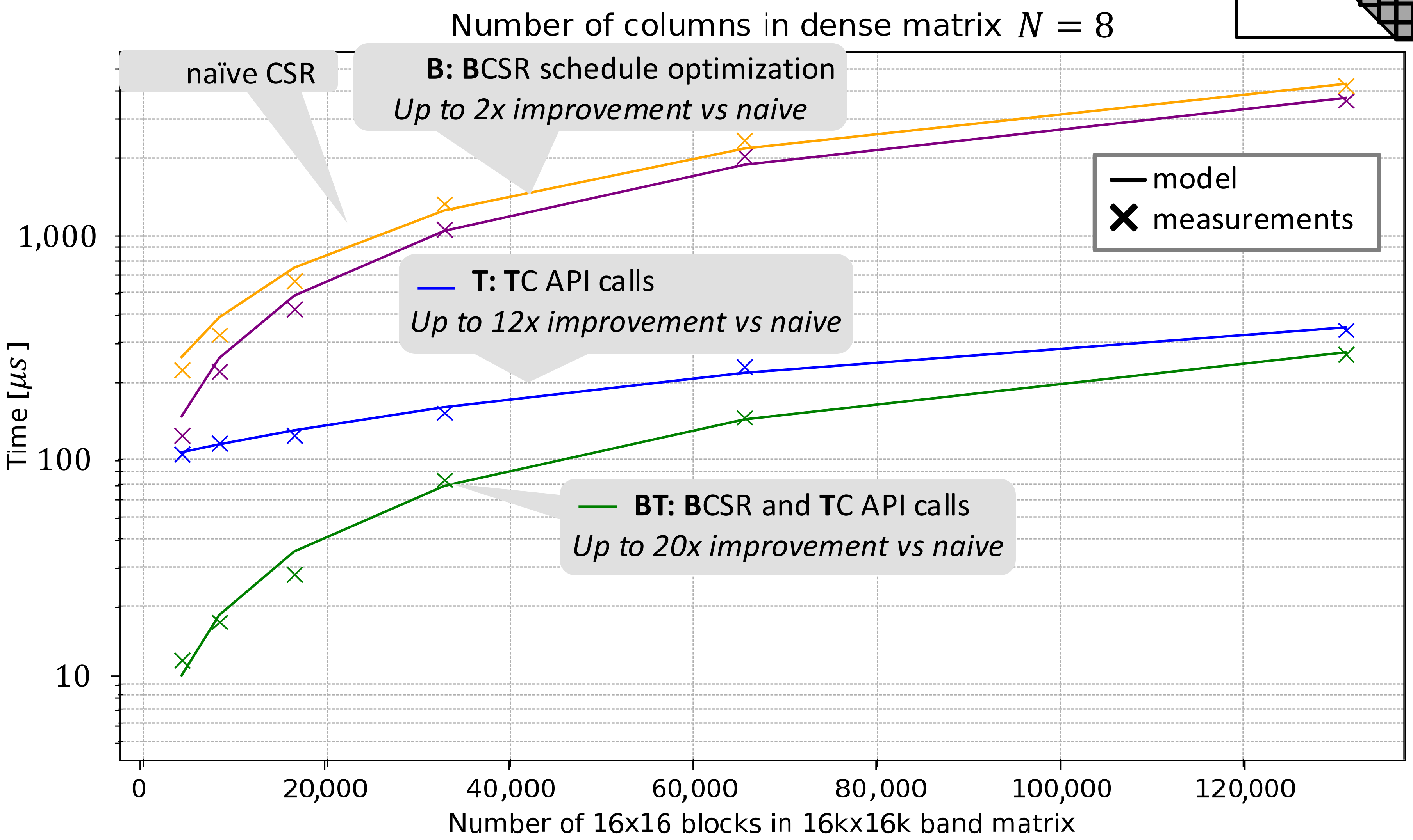




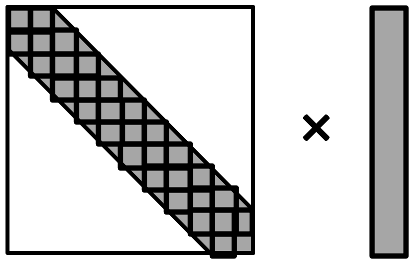
Performance Model

$$T_{tot} = T_e \cdot n_e + T_{init}$$

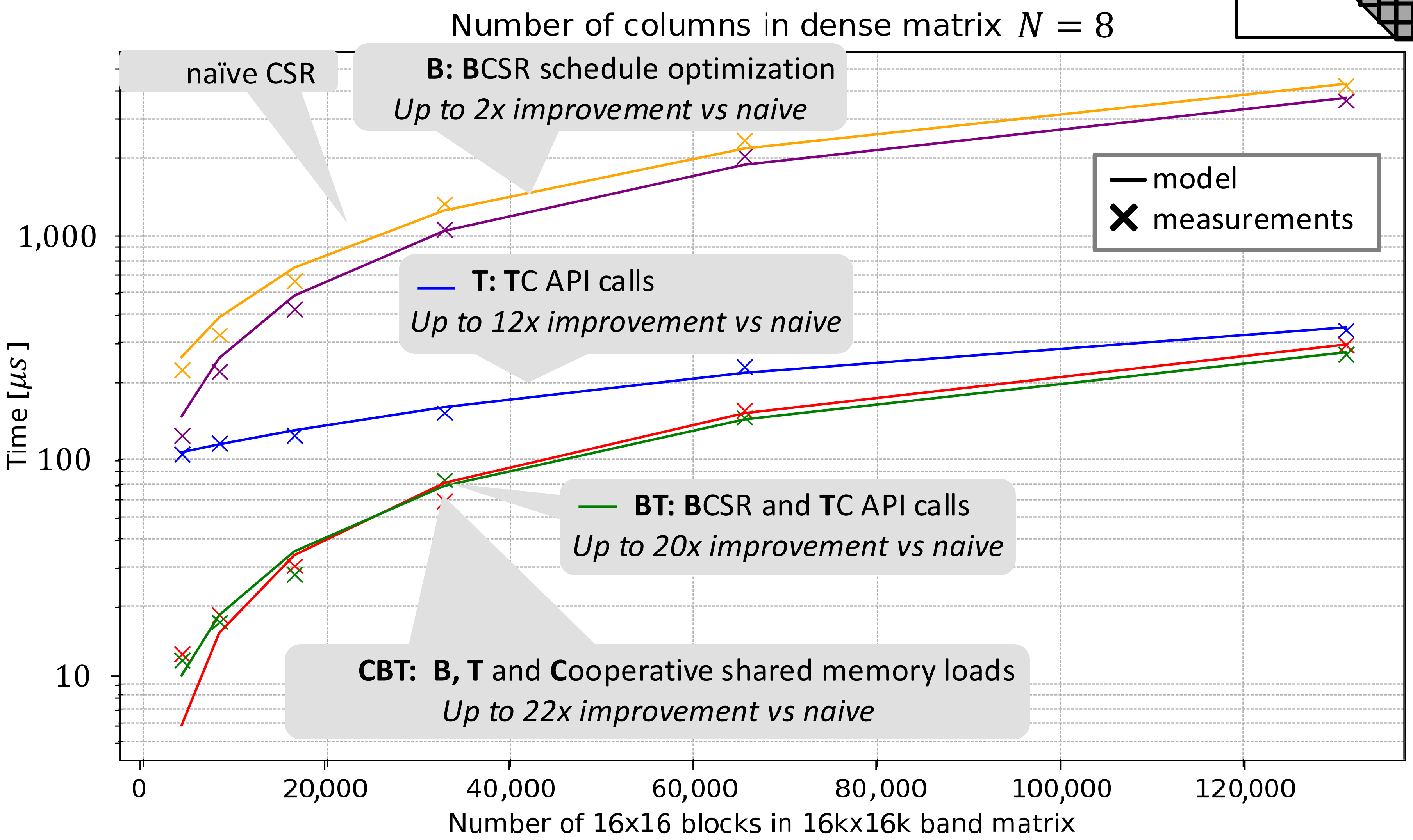




Performance Model

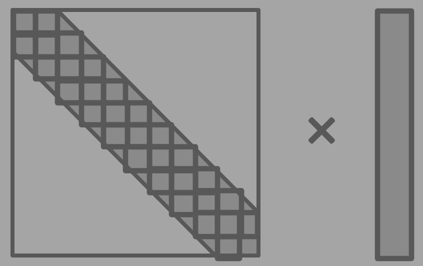


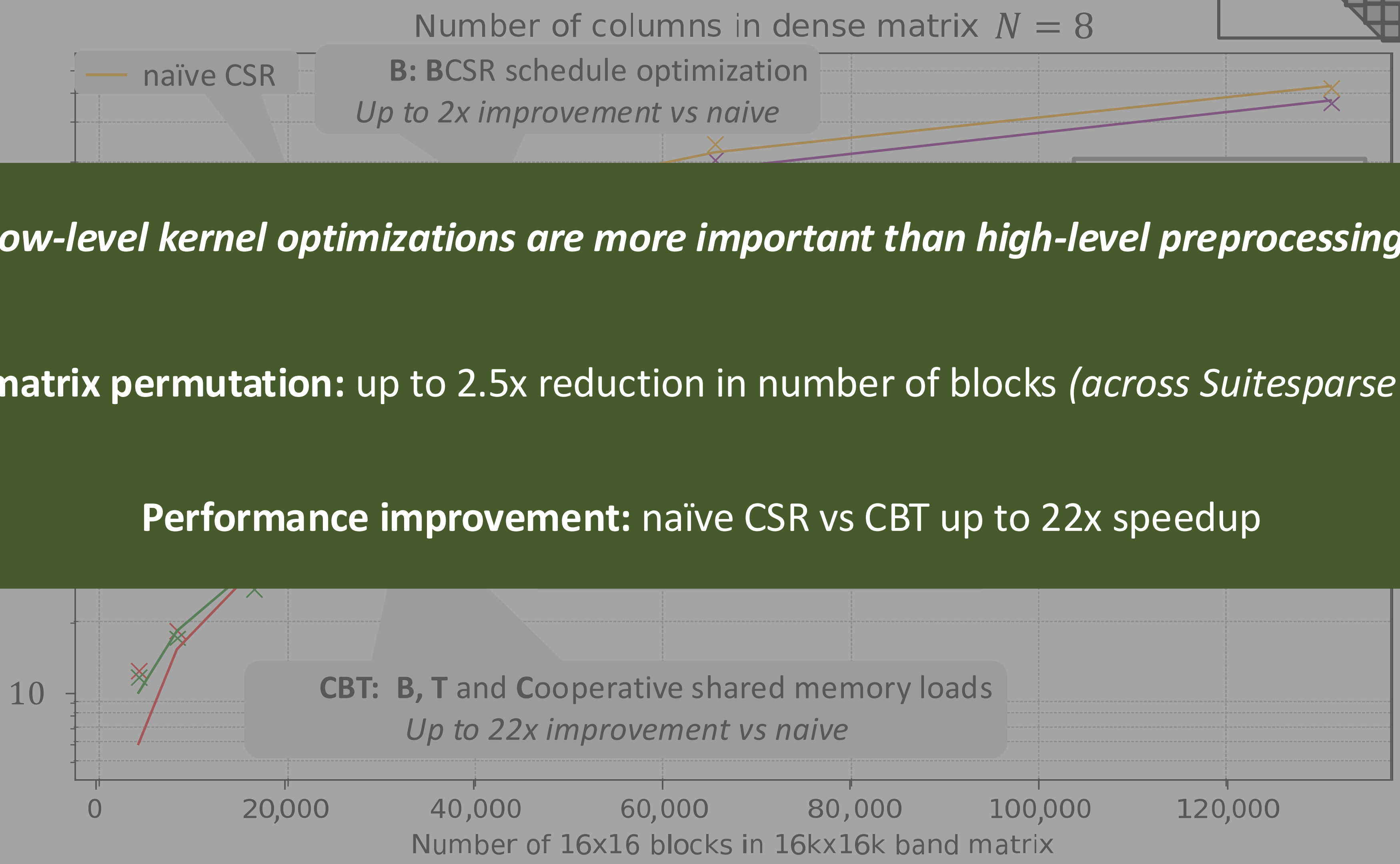
$$T_{tot} = T_e \cdot n_e + T_{init}$$



Performance Model

$$T_{tot} = T_e \cdot n_e + T_{init}$$





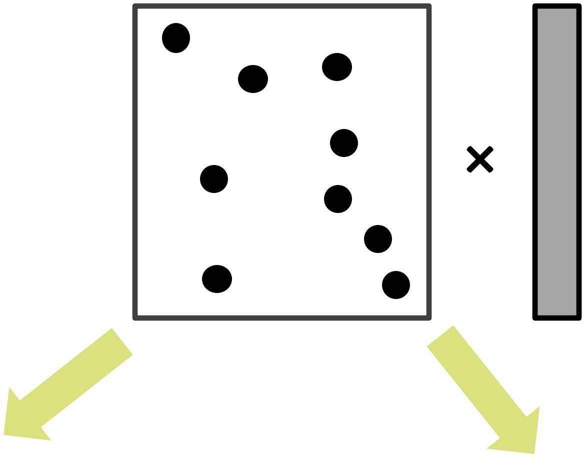
Low-level kernel optimizations are more important than high-level preprocessing!

Optimal matrix permutation: up to 2.5x reduction in number of blocks *(across Suitesparse matrices)*

Performance improvement: naïve CSR vs CBT up to 22x speedup

Evaluation

State-of-the-art benchmark:
Large, widely-used repository of sparse matrices from real-world applications

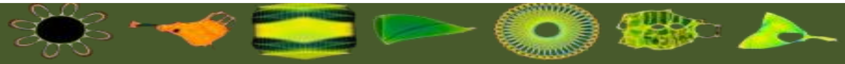


Motivation:

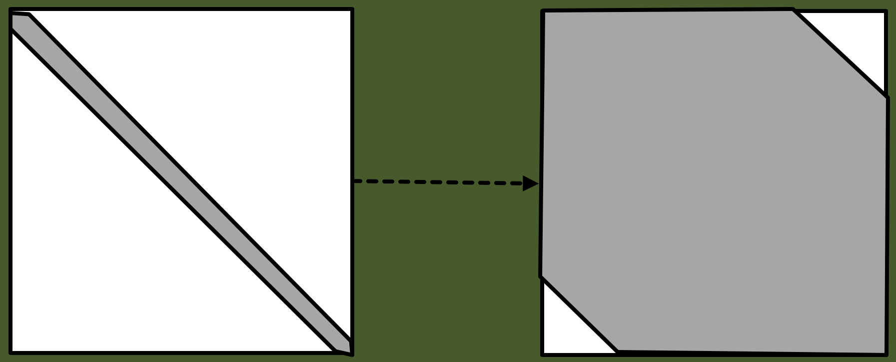
1. Remove nondeterminism and isolate performance-critical aspects:
   - Asynchronous, pipelined loads
   - Compute efficiency (tensor cores)
   - Stress test on "dense" sparse matrices
2. Unstructured sparsity kernels on structured sparsity matrices: ***e.g., HPCG***

SuiteSparse Collection

Domain	Name	Size	*nnz*	Sparsity
optimization	mip1	66K×66K	10.4M	99.76%
quantum chem.	conf5_4-8x8	49K×49K	1.9M	99.92%
2D/3D mesh	cant	62K×62K	4M	99.89%
weighted graph	pdb1HYS	36K×36K	4.3M	99.67%
fluid dynamics	rma10	46.8K×46.8K	2.3M	99.89%
2D/3D mesh	cop20k_A	121K×121K	2.6M	99.98%
2D/3D mesh	consph	83K×83K	6M	99.91%
structural	shipsec1	140K×140K	7.8M	99.96%
circuit simulation	dc2	116K×116K	766K	99.99%



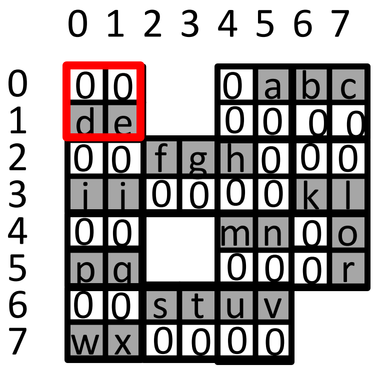
Synthetic Band Matrices



Preprocessing reordering

BCSR (b

A' = PA



4 6

t f g u v h 0

blocks: 7
zeros stored: 4
improvement: 13/7 = 1.85

Many blocks are almost

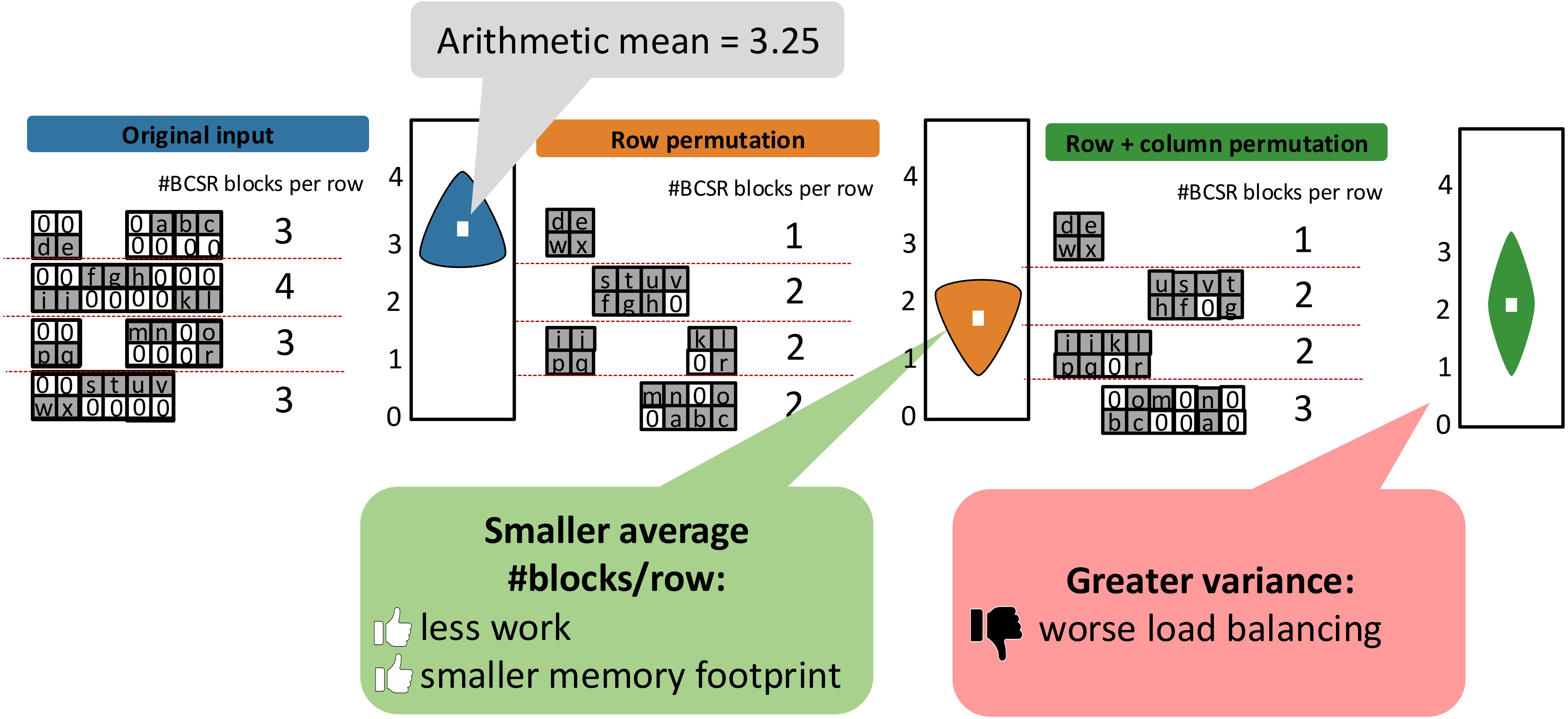
pooling-based bit masking
clustering based on
rd distance

DISCLAIMER:
The preprocessing time is **not** included in the performance measurements

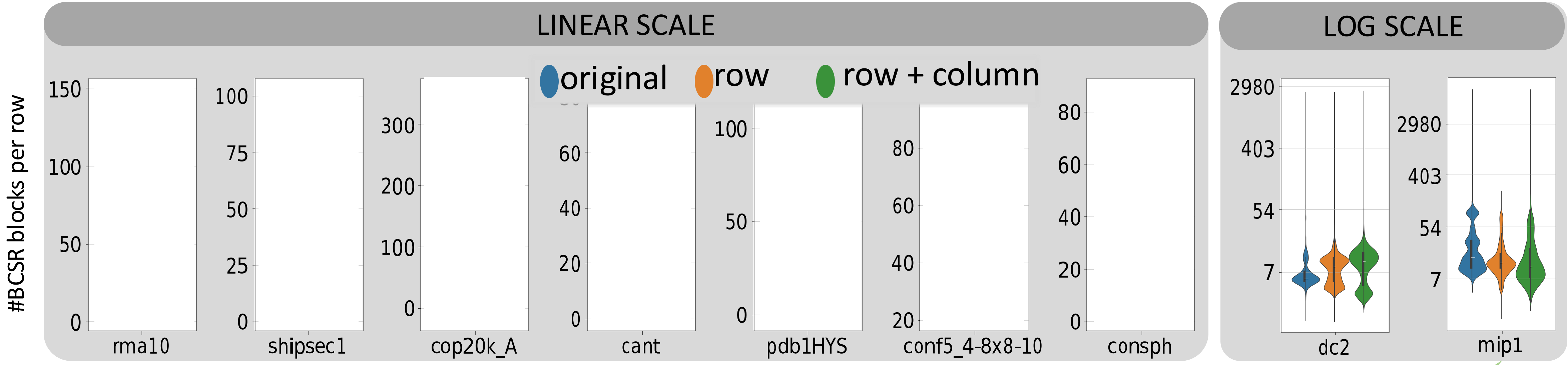
- Creating BCSR format:
  - Less that 1% of the runtime
  - Fair comparison: related works also measure only compute time

[6] P. S. Labini, M. Bernaschi, W. Nutt, F. Silvestri, and F. Vella, "Blocking sparse matrices to leverage dense-specific multiplication," in 2022 IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms (IA3), 2022, pp. 19–24.

Preprocessing reordering



Preprocessing reordering



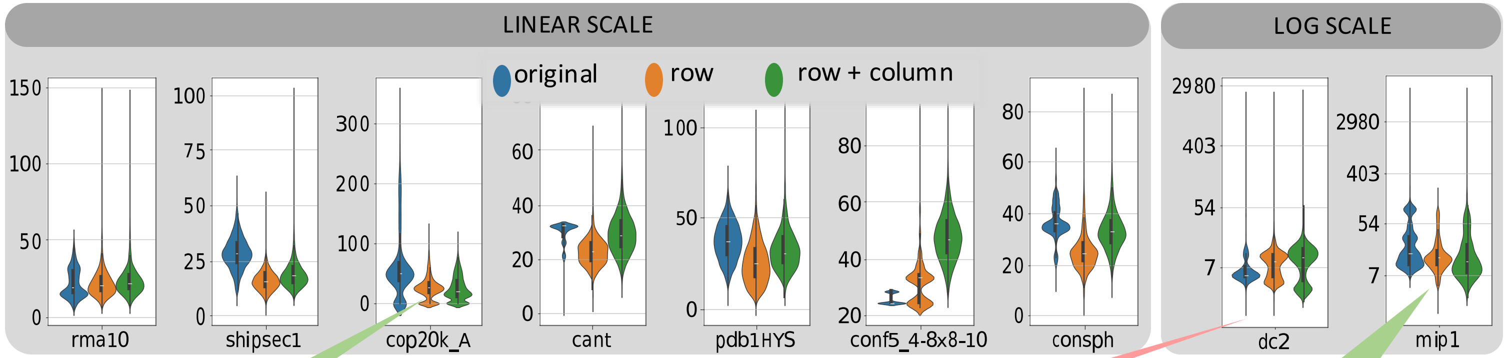
Distribution of the #BCSR blocks per row

2.5x fewer blocks
3x smaller standard deviation

99.994% sparsity
Coefficient of variation: 10.9

1.8x fewer blocks
8.4x smaller standard deviation

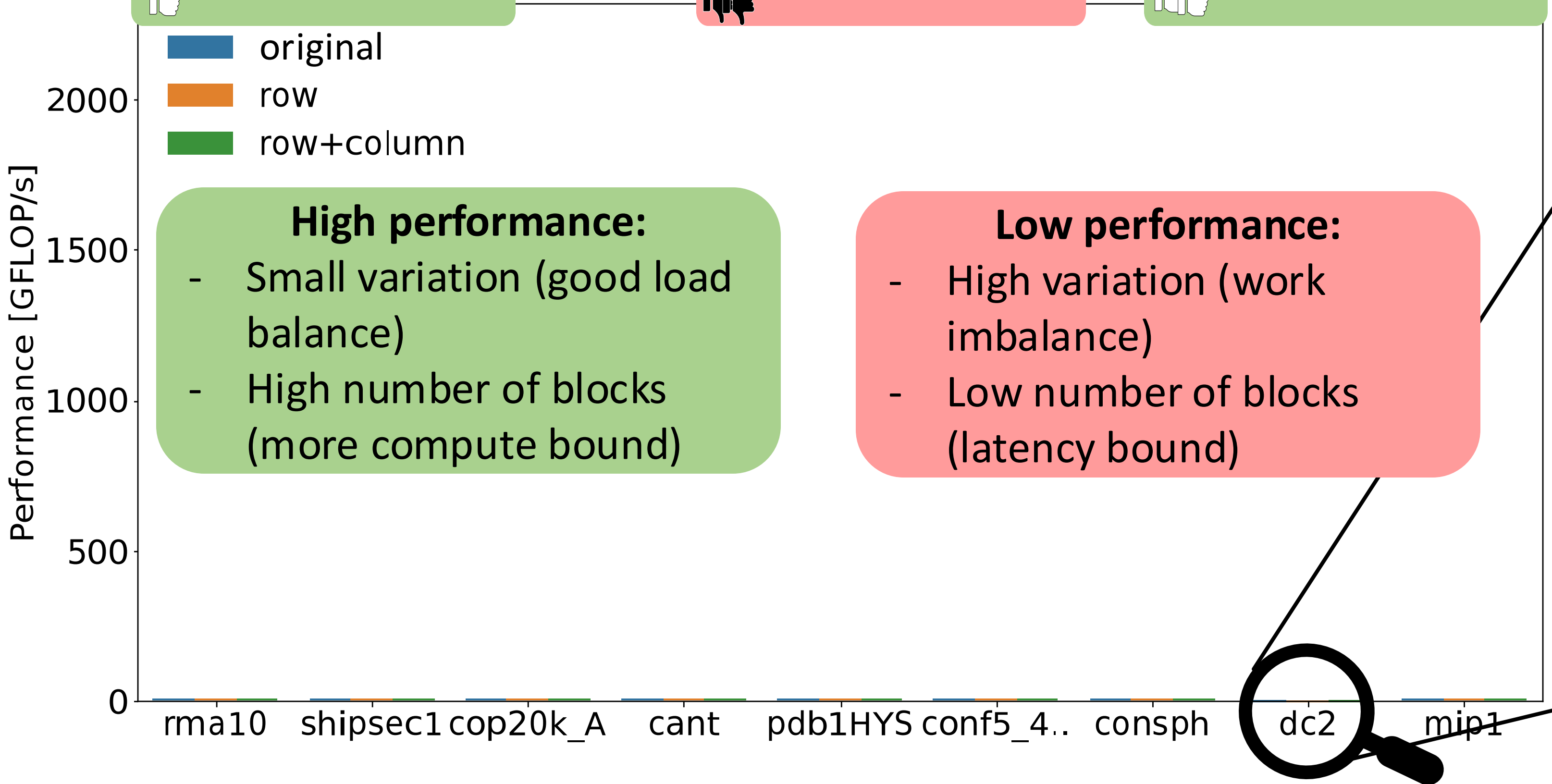
Preprocessing reordering



2.5x fewer blocks
3x smaller standard deviation

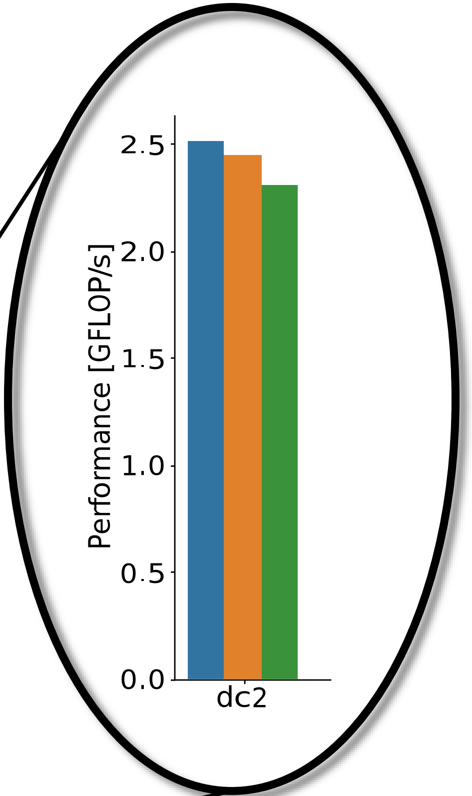
99.994% sparsity
Coefficient of variation: 10.9

1.8x fewer blocks
8.4x smaller standard deviation

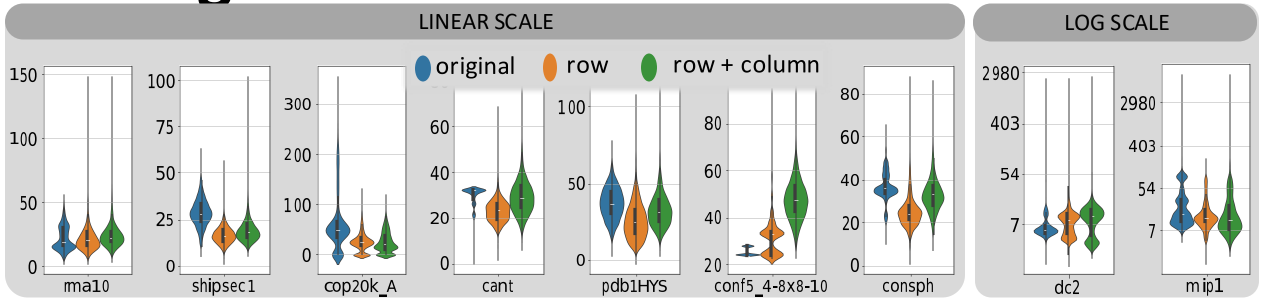


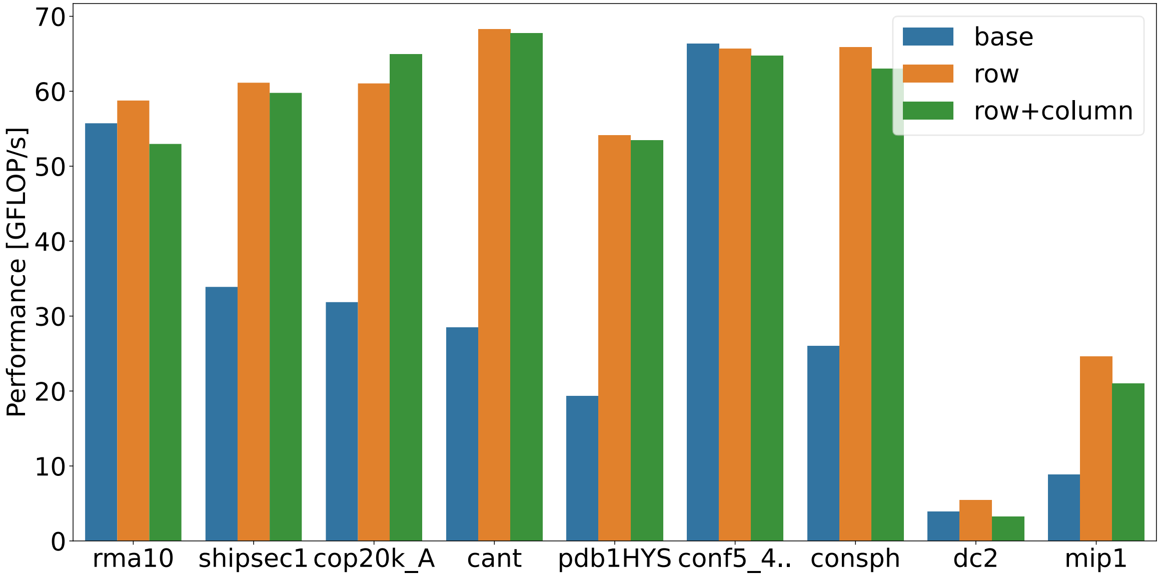
High performance:
- Small variation (good load balance)
- High number of blocks (more compute bound)

Low performance:
- High variation (work imbalance)
- Low number of blocks (latency bound)



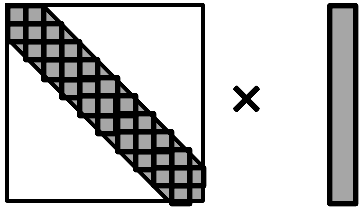
Preprocessing reordering



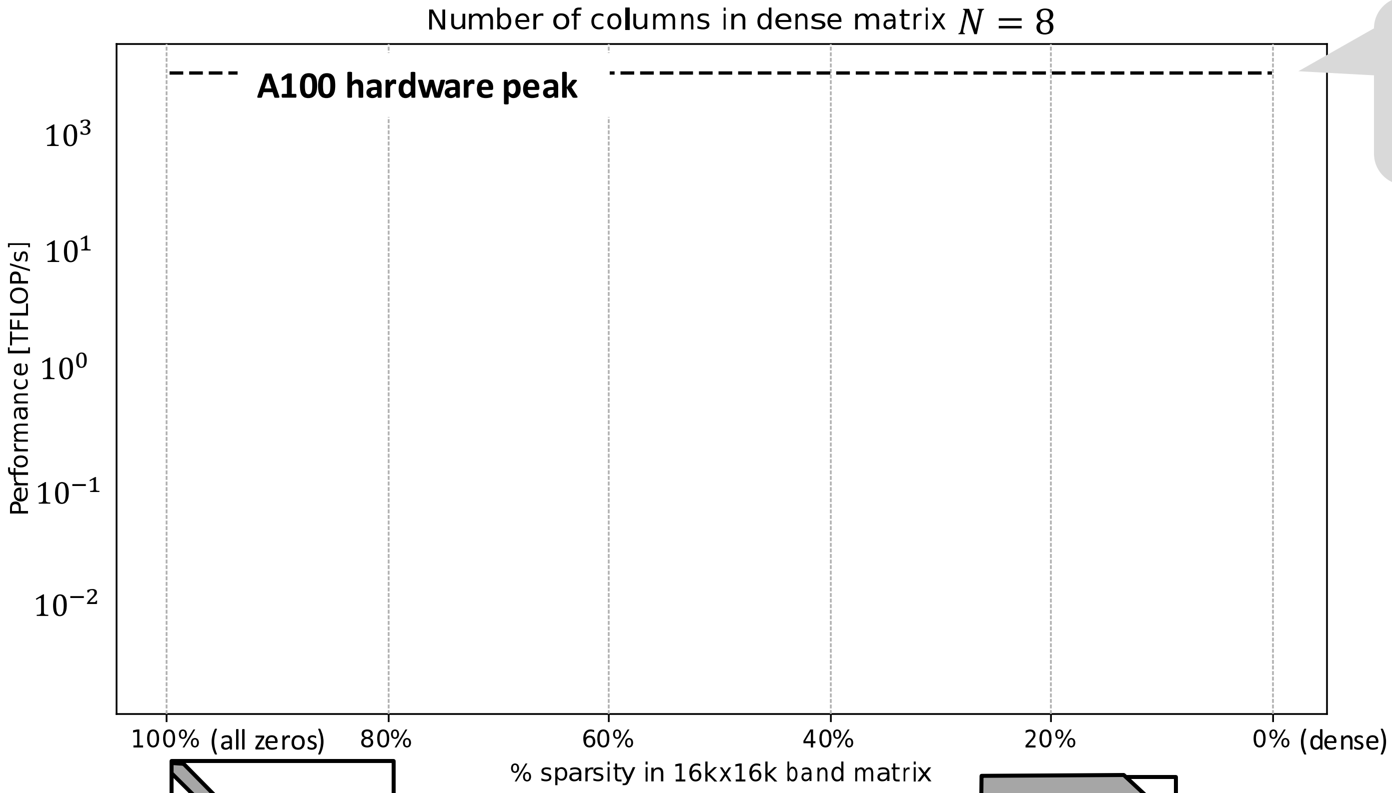


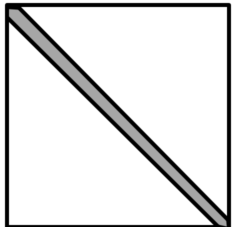
Synthetic Matrices

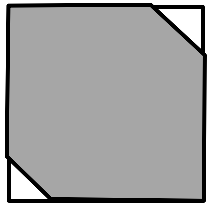
16,384 x 16,384



Measure the dependence on the number of blocks and isolate the randomness of the sparse matrix structure

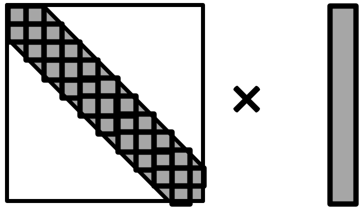






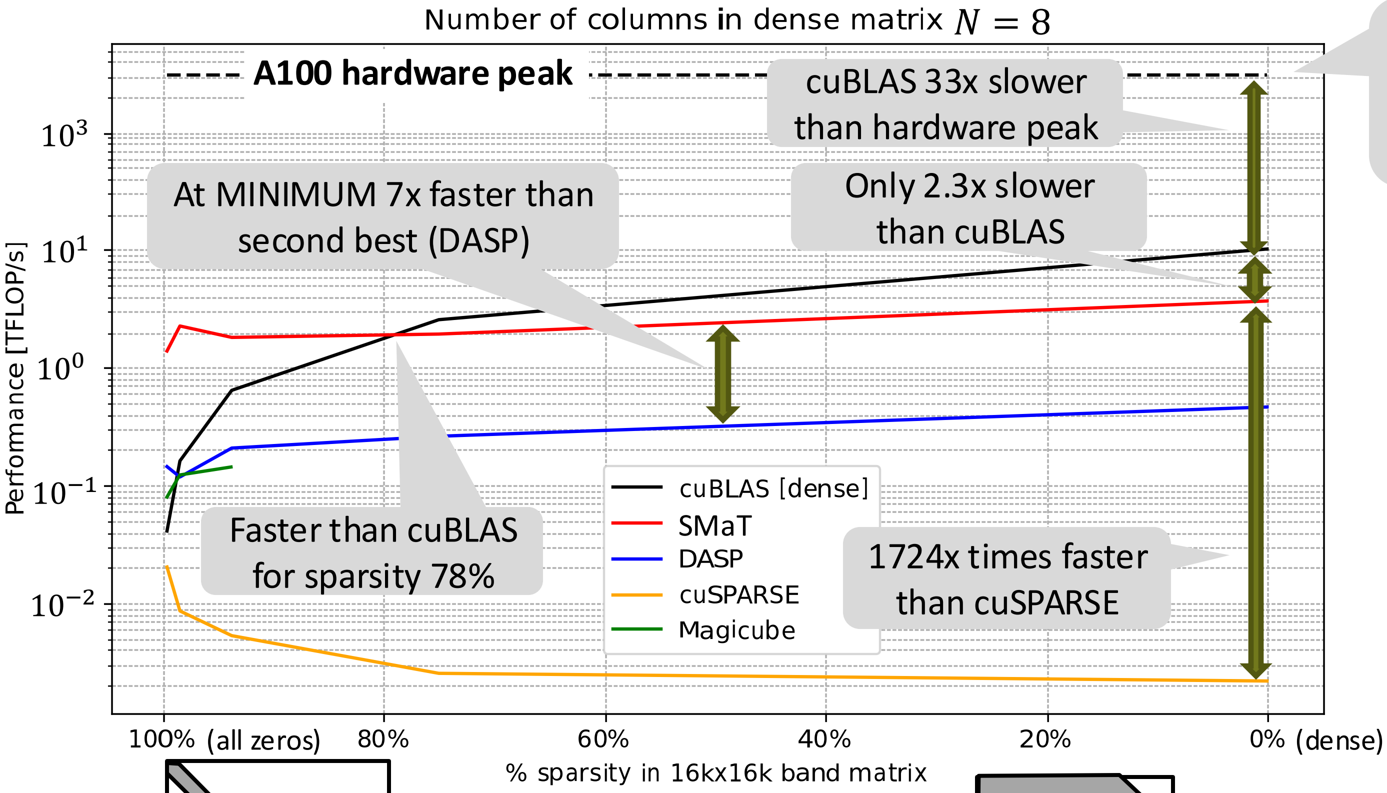
At what sparsity threshold a sparse library can outperform a highly-optimized dense library?

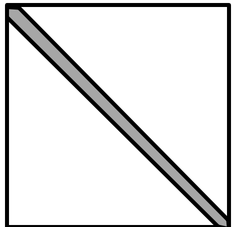
Synthetic Matrices

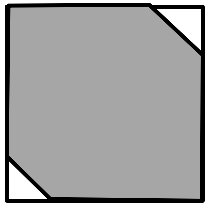


16,384 x 16,384

Measure the dependence on the number of blocks and isolate the randomness of the sparse matrix structure



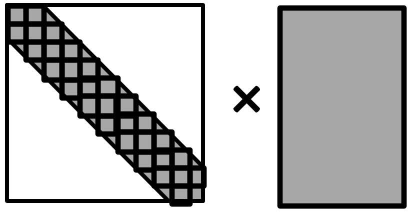




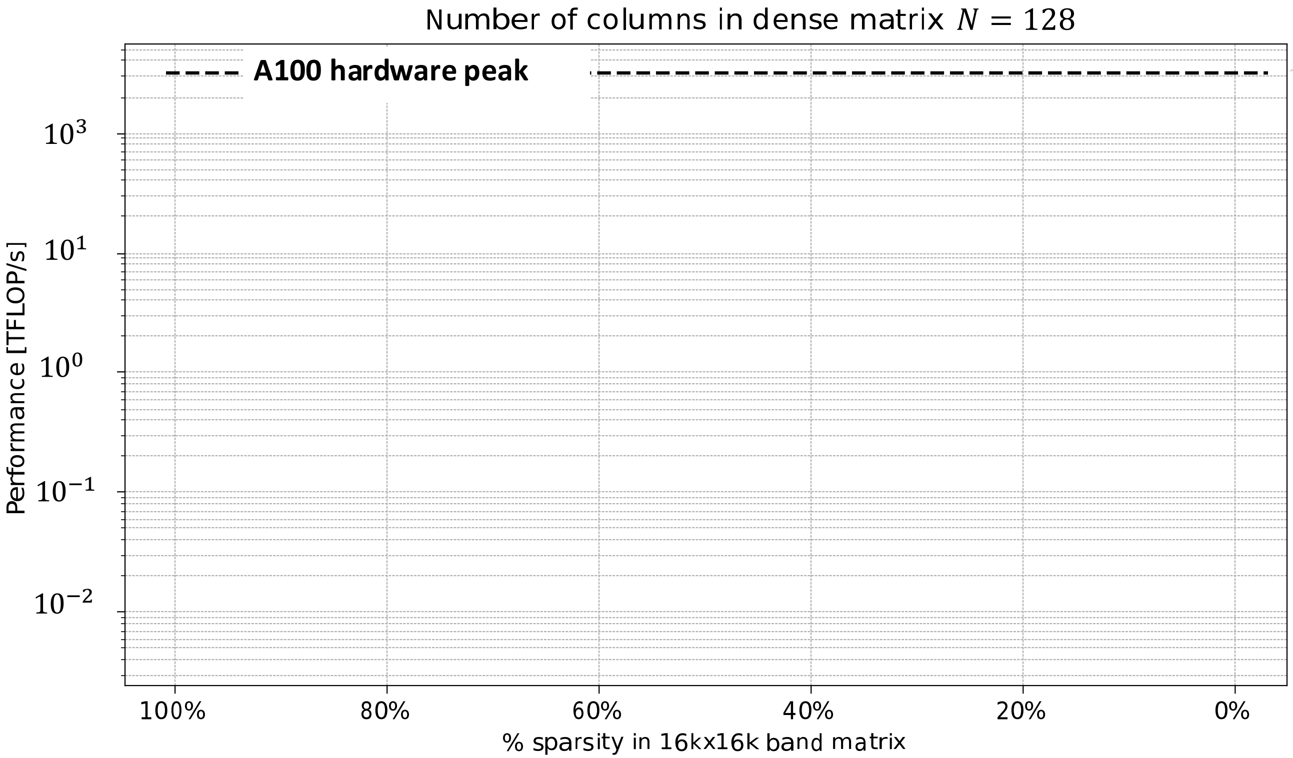
At what sparsity threshold a sparse library can outperform a highly-optimized dense library?

Synthetic Matrices

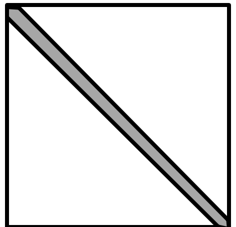
16,384 x 16,384

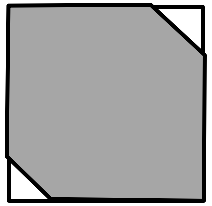


Measure the dependence on the number of blocks and isolate the randomness of the sparse matrix structure



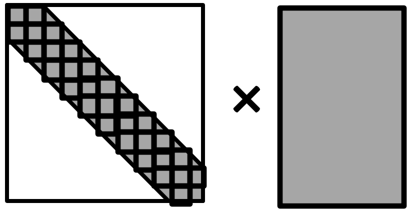
At what sparsity threshold a sparse library can outperform a highly-optimized dense library?



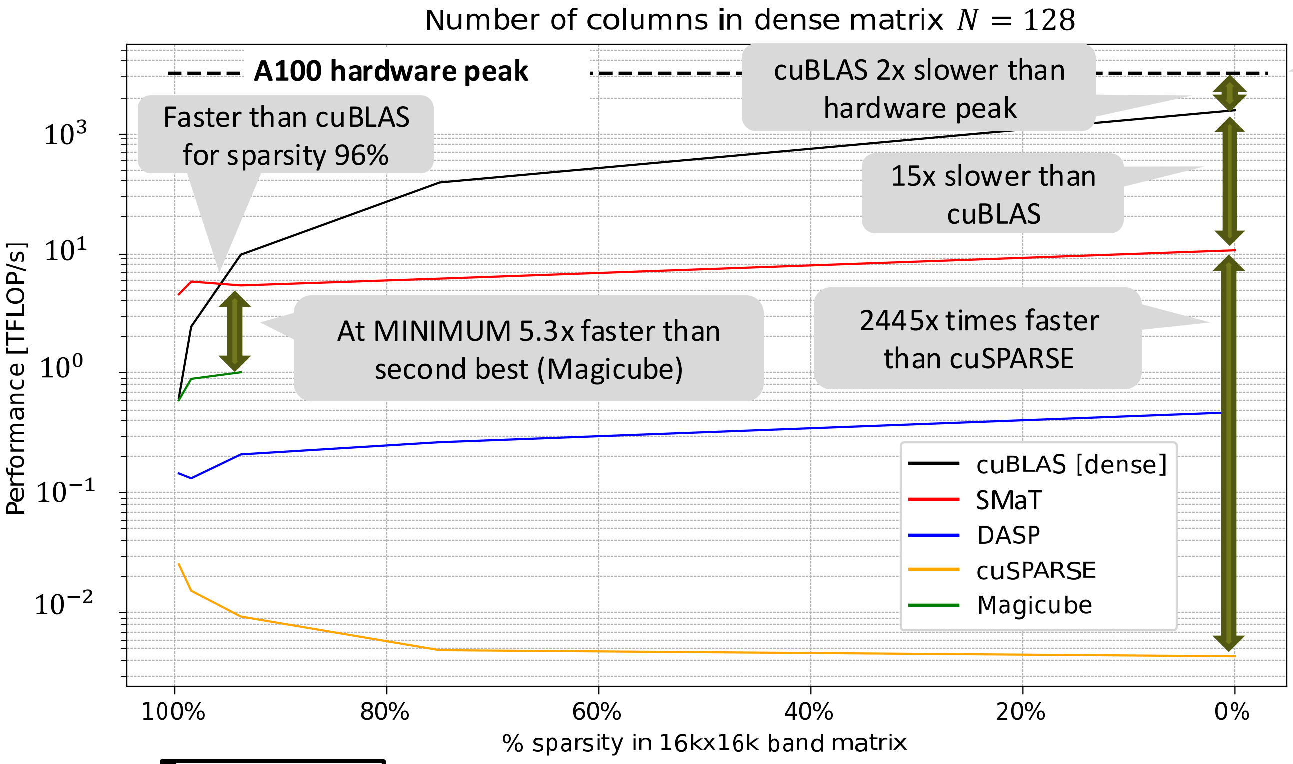


Synthetic Matrices

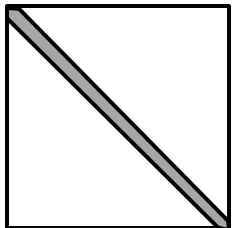
16,384 x 16,384

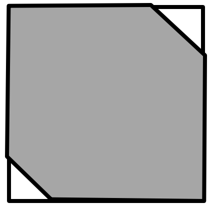


Measure the dependence on the number of blocks and isolate the randomness of the sparse matrix structure



At what sparsity threshold a sparse library can outperform a highly-optimized dense library?



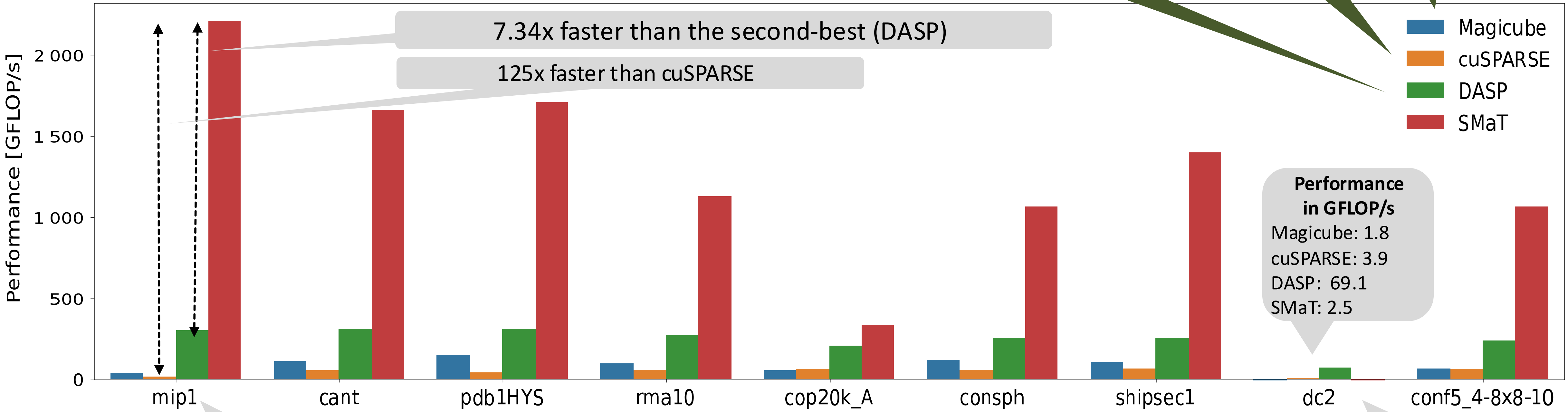


SuiteSparse performance comparison

2.60× faster (up to **7.34×**)

16.32× faster (up to **125.48×**)

10.78× faster (up to **51.23×**)

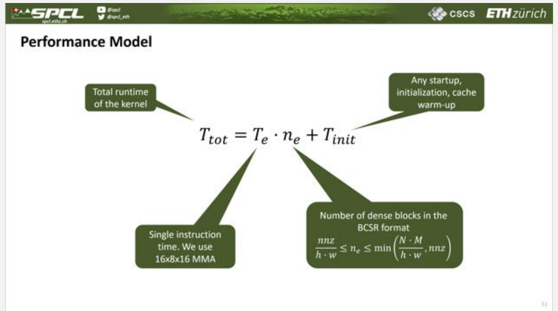


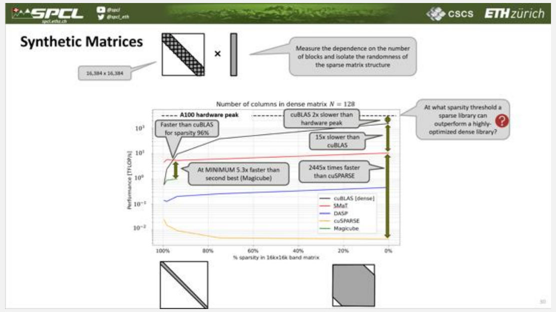
Importance of good preprocessing

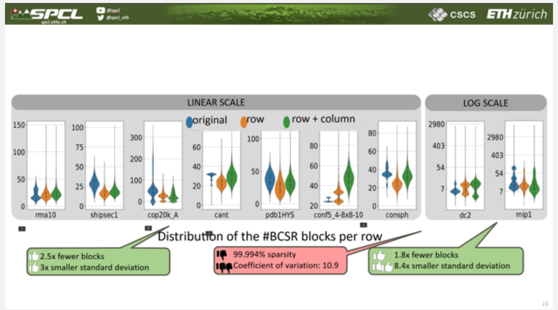
Ill-suited for SMaT's execution model with static 2D parallel schedule

SMaT is better on average **7.71×** (geometric mean) than the respective baselines

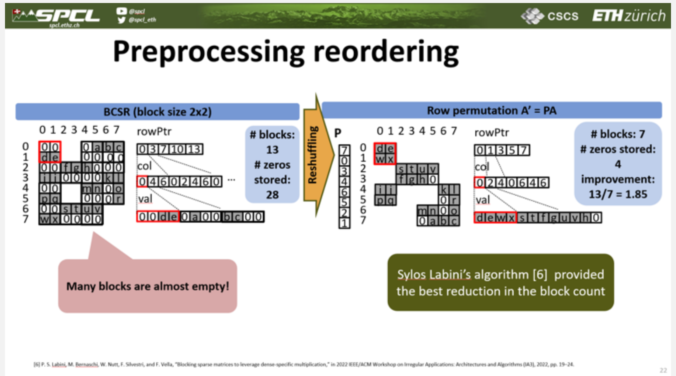
UNDERSTANDING performance

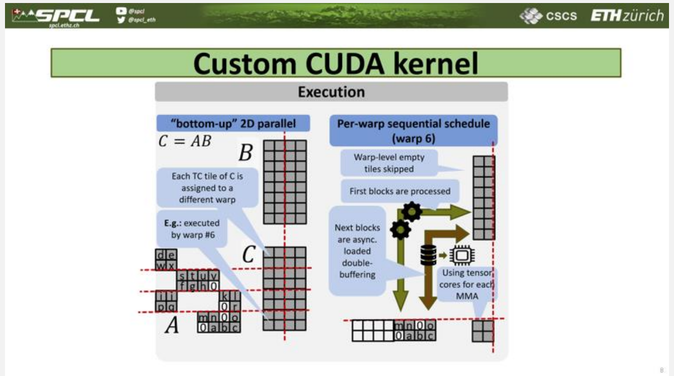




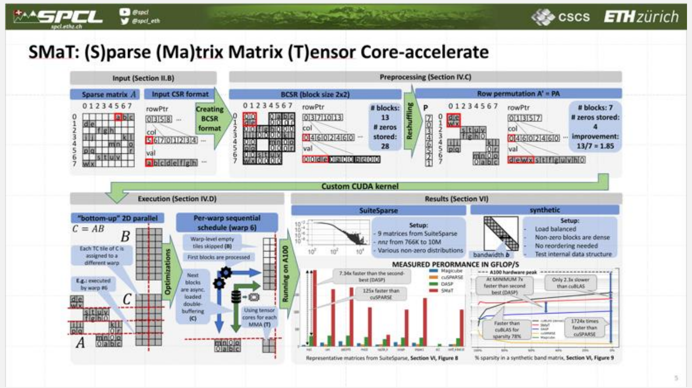


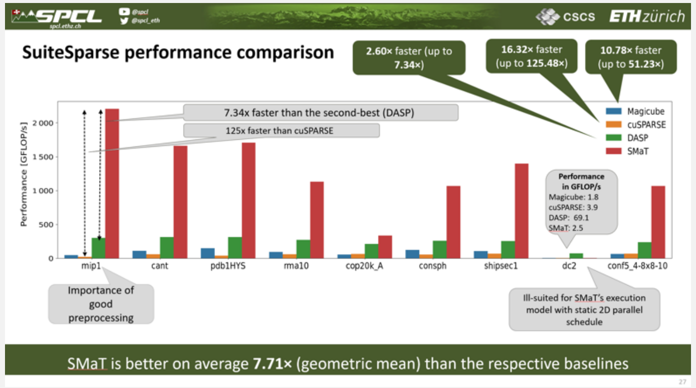
OPTIMIZING performance





DELIVERING performance





More of SPCL's research:

youtube.com/@spcl — 210+ Talks

twitter.com/spcl_eth — 1.6K+ Followers

github.com/spcl — 5.6K+ Stars

… or spcl.ethz.ch