ScalA'17: 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems
November 13, 2017

Flexible Batched Sparse Matrix-Vector Product on GPUs

Hartwig Anzt, Gary Collins, Jack Dongarra, Goran Flegar, Enrique, S. Quintana-Orti

Karlsruhe Institute of Technology

Denver, CO | hpc connects.

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

Input A, x, y Output $y = A \cdot x$

- Matrix A contains only few nonzero elements.
- Storing all entries results in large overhead (memory & computation).

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

Input A, x, y Output $y = A \cdot x$

- Matrix A contains only few nonzero elements.
- Storing all entries results in large overhead (memory & computation).
- Idea: Store only nonzero elements [nz] explicitly.

$$A = \begin{pmatrix} 5.4 & 1.1 & 0 & 0 & 0 & 0 \\ 2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\ 0 & 0 & 4.2 & 0 & 0 & 0 \\ 5.4 & 0 & 0 & 9.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{pmatrix}$$

$$\text{value} = [\ 5.4 \quad 1.1 \quad 2.2 \quad 8.3 \quad 3.7 \quad 1.3 \quad 3.8 \quad 4.2 \quad 5.4 \quad 9.2 \quad 1.1 \quad 8.1\ ]$$ Value

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

Input A, x, y Output $y = A \cdot x$

- Matrix A contains only few nonzero elements.
- Storing all entries results in large overhead (memory & computation).
- Idea: Store only nonzero elements [nz] explicitly.

Need to also store location of nonzero elements!

$$A = \begin{pmatrix} 5.4 & 1.1 & 0 & 0 & 0 & 0 \\ 2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\ 0 & 0 & 4.2 & 0 & 0 & 0 \\ 5.4 & 0 & 0 & 9.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{pmatrix}$$

Memory footprint of COO format:
nz(val) + 2*nz(int)

COO format :

$$\text{value} = [\ 5.4\ \ 1.1\ \ 2.2\ \ 8.3\ \ 3.7\ \ 1.3\ \ 3.8\ \ 4.2\ \ 5.4\ \ 9.2\ \ 1.1\ \ 8.1\ ]$$ Value

$$\text{colidx} = [\ 0\ \ 1\ \ 0\ \ 1\ \ 3\ \ 4\ \ 5\ \ 2\ \ 0\ \ 3\ \ 4\ \ 5\ ]$$ Column-index

$$\text{rowidx} = [\ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 2\ \ 3\ \ 3\ \ 4\ \ 5\ ]$$ Row-index

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

- Matrix A contains only few nonzero elements.
- Storing all entries results in large overhead (memory & computation).
- Idea: Store only nonzero elements [nz] explicitly.

Need to also store location of nonzero elements!

$$A = \begin{pmatrix} 5.4 & 1.1 & 0 & 0 & 0 & 0 \\ 2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\ 0 & 0 & 4.2 & 0 & 0 & 0 \\ 5.4 & 0 & 0 & 9.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{pmatrix}$$

Memory footprint of COO format:
nz(val) + 2*nz(int)

Memory footprint of CSR format:
nz(val) + nz(int) + (n+1) (int)

CSR format :

$$\text{value} = [\ 5.4 \quad 1.1 \quad 2.2 \quad 8.3 \quad 3.7 \quad 1.3 \quad 3.8 \quad 4.2 \quad 5.4 \quad 9.2 \quad 1.1 \quad 8.1\ ]$$ Value

$$\text{colidx} = [\ 0 \quad 1 \quad 0 \quad 1 \quad 3 \quad 4 \quad 5 \quad 2 \quad 0 \quad 3 \quad 4 \quad 5\ ]$$ Column-index

$$\text{rowptr} = [\ 0 \quad 2 \quad 7 \quad 8 \quad 10 \quad 11 \quad 12\ ]$$ Points to the first element in each row

Number of nonzero elements

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

How to parallelize this?

$$A = \begin{pmatrix} 5.4 & 1.1 & 0 & 0 & 0 & 0 \\ 2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\ 0 & 0 & 4.2 & 0 & 0 & 0 \\ 5.4 & 0 & 0 & 9.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{pmatrix}$$

$$\text{value} = [\ 5.4 \quad 1.1 \quad 2.2 \quad 8.3 \quad 3.7 \quad 1.3 \quad 3.8 \quad 4.2 \quad 5.4 \quad 9.2 \quad 1.1 \quad 8.1\ ]$$ Value

$$\text{colidx} = [\ 0 \quad 1 \quad 0 \quad 1 \quad 3 \quad 4 \quad 5 \quad 2 \quad 0 \quad 3 \quad 4 \quad 5\ ]$$ Column-index

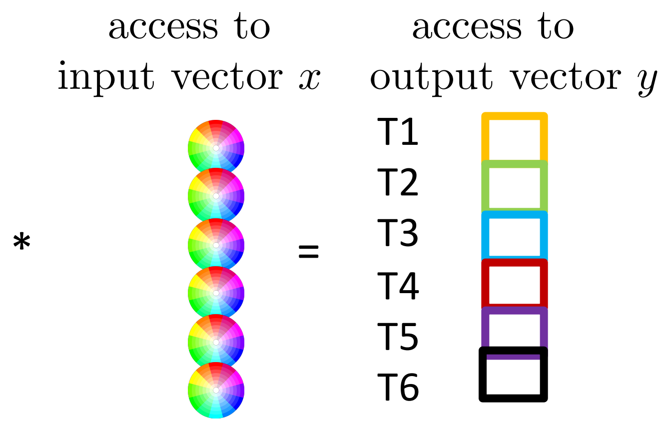
$$\text{rowidx} = [\ 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2 \quad 3 \quad 3 \quad 4 \quad 5\ ]$$ Row-index

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

How to parallelize this?

- Parallelize by rows:
  - Every “thread” handles the computation of one sum in local memory.
  - Significant workload imbalance!
  - Need branching logic, branch divergence on vector machines.

$$\begin{array}{l} T1 \\ T2 \\ T3 \\ T4 \\ T5 \\ T6 \end{array} \begin{pmatrix} 5.4 & 1.1 & 0 & 0 & 0 & 0 \\ 2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\ 0 & 0 & 4.2 & 0 & 0 & 0 \\ 5.4 & 0 & 0 & 9.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{pmatrix} *$$



$$\text{value} = [\; 5.4 \;\; 1.1 \;|\; 2.2 \;\; 8.3 \;\; 3.7 \;\; 1.3 \;\; 3.8 \;|\; 4.2 \;|\; 5.4 \;\; 9.2 \;|\; 1.1 \;|\; 8.1 \;]$$ Value

$$\text{colidx} = [\; 0 \;\; 1 \;|\; 0 \;\; 1 \;\; 3 \;\; 4 \;\; 5 \;|\; 2 \;|\; 0 \;\; 3 \;|\; 4 \;|\; 5 \;]$$ Column-index

$$\text{rowidx} = [\; 0 \;\; 0 \;|\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;|\; 2 \;|\; 3 \;\; 3 \;|\; 4 \;|\; 5 \;]$$ Row-index

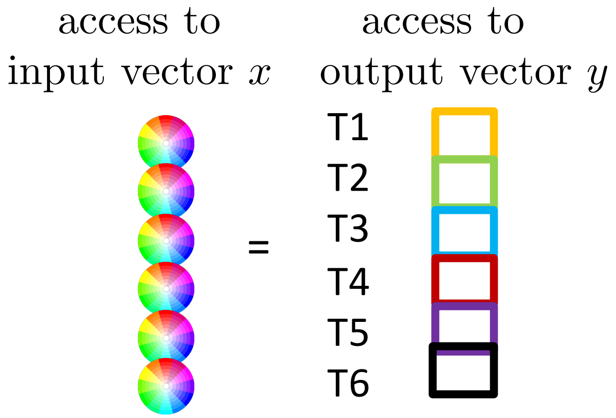
A never ending story: The sparse matrix vector Product (SpMV) on Manycore

How to parallelize this?

- Parallelize by rows:
  - Every “thread” handles the computation of one sum in local memory.
  - Balanced workload.
  - Can result in significant overhead for unbalanced problems.

ELL format :

$$\begin{array}{c} T1 \\ T2 \\ T3 \\ T4 \\ T5 \\ T6 \end{array} \begin{pmatrix} 5.4 & 1.1 & 0 & 0 & 0 & 0 \\ 2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\ 0 & 0 & 4.2 & 0 & 0 & 0 \\ 5.4 & 0 & 0 & 9.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{pmatrix} *$$



Values and column-index padded for uniform “row-length”

$$\text{value} = [\ 5.4\ 1.1\ 0.0\ 0.0\ 0.0\ 2.2\ 8.3\ 3.7\ 1.3\ 3.8\ 4.2\ 0.0\ 0.0\ 0.0\ 0.0\ 5.4\ 9.2\ 0.0\ 0.0\ 0.0\ 1.1\ 0.0\ 0.0\ 0.0\ 0.0\ 8.1\ 0.0\ 0.0\ 0.0\ ]$$

$$\text{colidx} = [\ 0\ \ 1\ -\ -\ -\ 0\ \ 1\ \ 3\ \ 4\ \ 5\ \ 0\ -\ -\ -\ -\ 0\ \ 3\ -\ -\ -\ 4\ -\ -\ -\ -\ 5\ -\ -\ -\ -\ ]$$

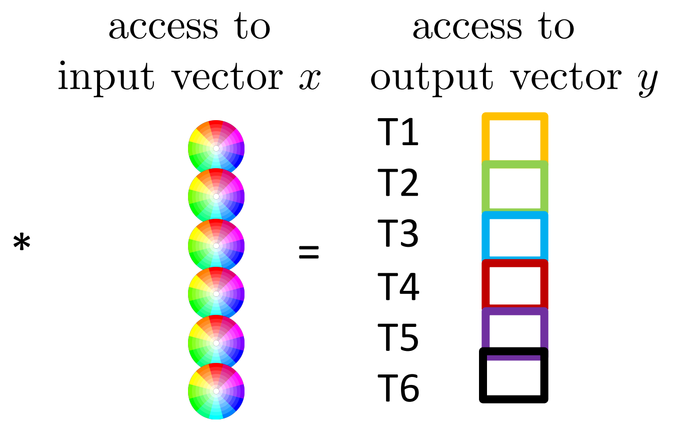
Number of nonzero elements

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

How to parallelize this?

- Parallelize by rows:
  - Every “thread” handles the computation of one sum in local memory.
  - Significant workload imbalance!
  - “Ordered” access to input vector x.

$$
\begin{array}{l}
\text{T1} \\ \text{T2} \\ \text{T3} \\ \text{T4} \\ \text{T5} \\ \text{T6}
\end{array}
\begin{pmatrix}
5.4 & 1.1 & 0 & 0 & 0 & 0 \\
2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\
0 & 0 & 4.2 & 0 & 0 & 0 \\
5.4 & 0 & 0 & 9.2 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.1 & 0 \\
0 & 0 & 0 & 0 & 0 & 8.1
\end{pmatrix}
\; * \; x \; = \; y
$$



$$
\begin{aligned}
\text{value} &= [\; 5.4 \;\; 1.1 \;|\; 2.2 \;\; 8.3 \;\; 3.7 \;\; 1.3 \;\; 3.8 \;|\; 4.2 \;|\; 5.4 \;\; 9.2 \;|\; 1.1 \;|\; 8.1 \;] \\
\text{colidx} &= [\; 0 \;\; 1 \;|\; 0 \;\; 1 \;\; 3 \;\; 4 \;\; 5 \;|\; 2 \;|\; 0 \;\; 3 \;|\; 4 \;|\; 5 \;] \\
\text{rowidx} &= [\; 0 \;\; 0 \;|\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;|\; 2 \;|\; 3 \;\; 3 \;|\; 4 \;|\; 5 \;]
\end{aligned}
$$

Value

Column-index

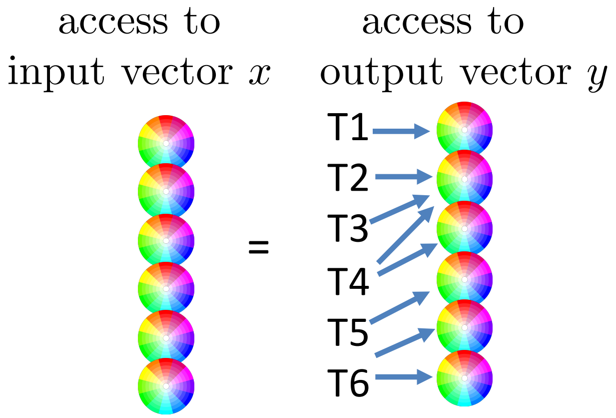
Row-index

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

How to parallelize this?

- Parallelize by elements:
  - Balanced workload.
  - Partial sums need synchronization: Write conflicts!

$$\begin{pmatrix} 5.4 & 1.1 & 0 & 0 & 0 & 0 \\ 2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\ 0 & 0 & 4.2 & 0 & 0 & 0 \\ 5.4 & 0 & 0 & 9.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{pmatrix} * x = y$$



$$\text{value} = [\ 5.4\ \ 1.1\ \ 2.2\ \ 8.3\ \ 3.7\ \ 1.3\ \ 3.8\ \ 4.2\ \ 5.4\ \ 9.2\ \ 1.1\ \ 8.1\ ]$$

$$\text{colidx} = [\ 0\ \ 1\ \ 0\ \ 1\ \ 3\ \ 4\ \ 5\ \ 2\ \ 0\ \ 3\ \ 4\ \ 5\ ]$$

$$\text{rowidx} = [\ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 2\ \ 3\ \ 3\ \ 4\ \ 5\ ]$$

Value

Column-index

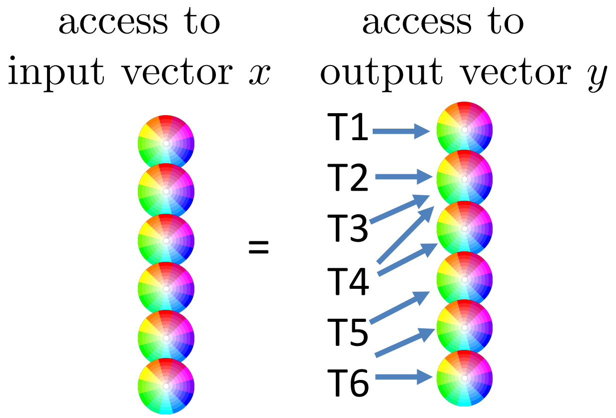
Row-index

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

How to parallelize this?

- Parallelize by elements:
  - Balanced workload.
  - Partial sums need synchronization: Write conflicts!

$$\begin{pmatrix} 5.4 & 1.1 & 0 & 0 & 0 & 0 \\ 2.2 & 8.3 & 0 & 3.7 & 1.3 & 3.8 \\ 0 & 0 & 4.2 & 0 & 0 & 0 \\ 5.4 & 0 & 0 & 9.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.1 \end{pmatrix} * \quad = $$



$$\begin{aligned} \text{value} &= [\; 5.4 \;\; 1.1 \;\; 2.2 \;\; 8.3 \;\; 3.7 \;\; 1.3 \;\; 3.8 \;\; 4.2 \;\; 5.4 \;\; 9.2 \;\; 1.1 \;\; 8.1 \;] \\ \text{colidx} &= [\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 3 \;\; 4 \;\; 5 \;\; 2 \;\; 0 \;\; 3 \;\; 4 \;\; 5 \;] \\ \text{rowidx} &= [\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 2 \;\; 3 \;\; 3 \;\; 4 \;\; 5 \;] \end{aligned}$$

Value

Column-index

Row-index

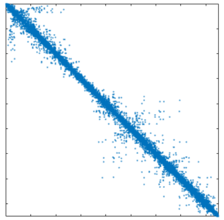
G. Flegar et al.: **Overcoming Load Imbalance for Irregular Sparse Matrices**, IA^3, 2017.

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

"Different kernels optimal for different problem classes"

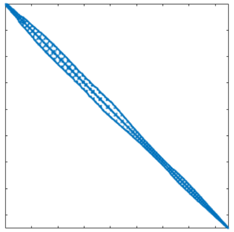
CSR

- small memory footprint
- Needs some logic for row-parallel processing



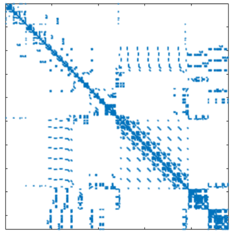
ELL

- zero-padding allows for efficient SIMD execution
- Efficient for balanced matrices



COO

- can compensate workload imbalance for irregular patterns

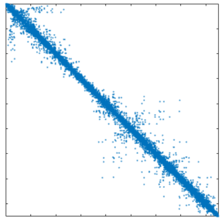


…

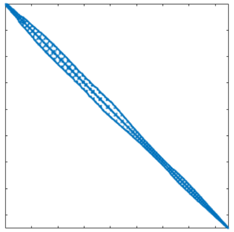
A never ending story: The sparse matrix vector Product (SpMV) on Manycore

"Different kernels optimal for different problem classes"

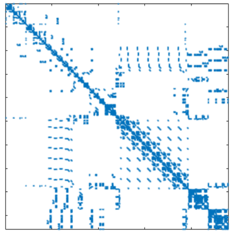
CSR
- small memory footprint
- Needs some logic for row-parallel processing



ELL
- zero-padding allows for efficient SIMD execution
- Efficient for balanced matrices



COO
- can compensate workload imbalance for irregular patterns



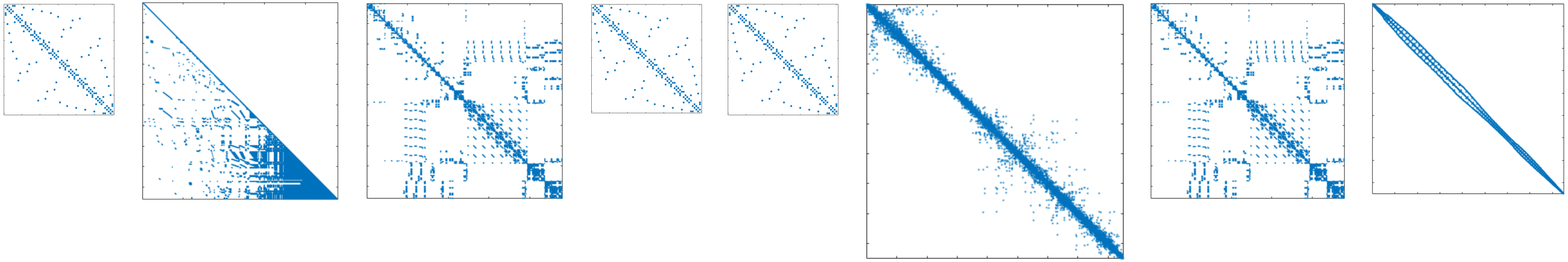
…

For a single problem, we can usually find an optimal kernel, BUT…

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

- *What if we process many different matrices at a time? (Assume they are all small…)*

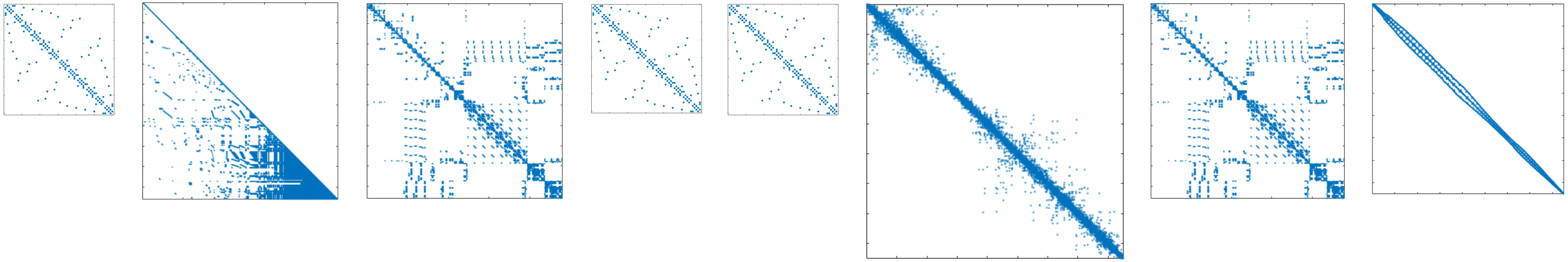
…



…

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

- *What if we process many different matrices at a time? (Assume they are all small…)*

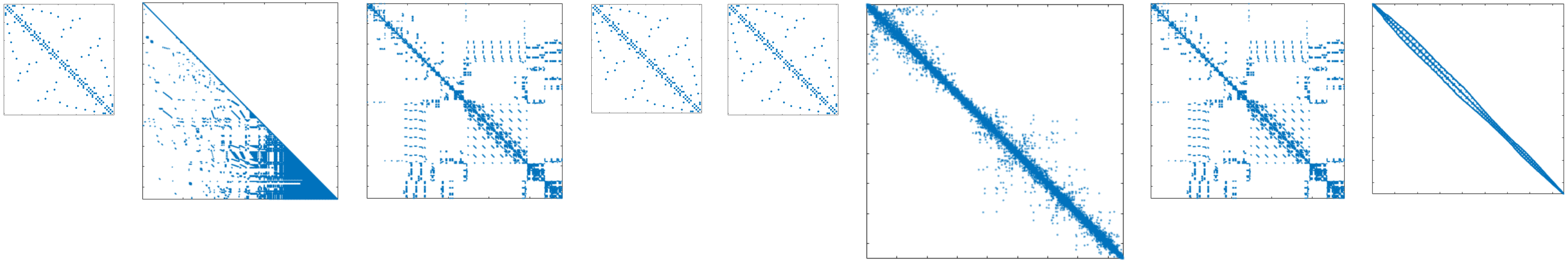
…  …

- Design a **batched SpMV kernel**.
  - *Process a large number of data-independent problems in parallel.*

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

- *What if we process many different matrices at a time? (Assume they are all small…)*

…



…

- Design a **batched SpMV kernel**.
  - *Process a large number of data-independent problems in parallel.*

Batched, Reproducible, and Reduced Precision BLAS

SESSION LEADER: Piotr Luszczek

ask a question · give feedback

ADDITIONAL SESSION LEADERS: Jack Dongarra, Cris Cecka, Timothy Costa, Sivasankaran Rajamanickam, Azzam Haidar, Mawussi Zounon

EVENT TYPE: Birds of a Feather

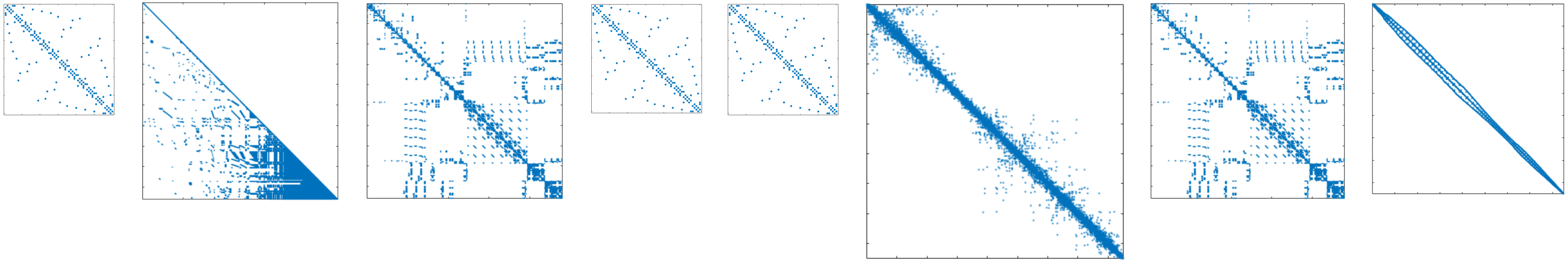
EVENT TAGS: Ex TP

TIME: Tuesday, November 14th, 12:15pm - 1:15pm

LOCATION: 402-403-404

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

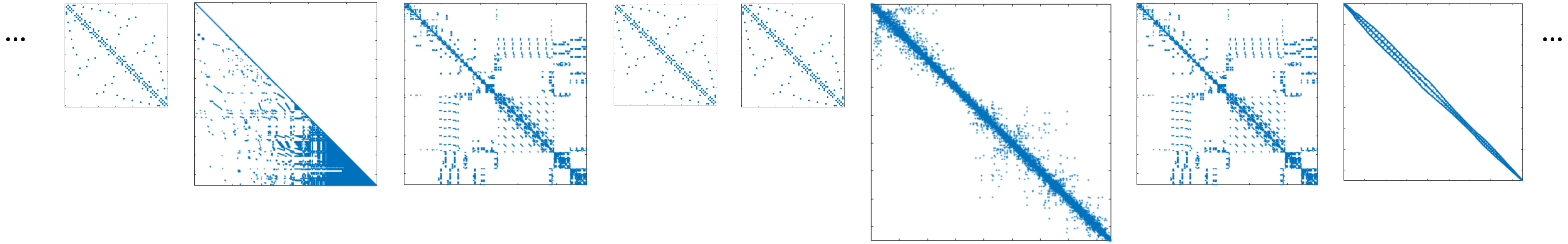
- *What if we process many different matrices at a time? (Assume they are all small…)*

…  …

- Design a **batched SpMV kernel**.
  - *Process a large number of data-independent problems in parallel.*
  - *Are the problems*
    - Same Size?
    - Same number of nonzeros overall?
    - Same number of nonzeros in every row?
    - Same sparsity pattern?

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

- *What if we process many different matrices at a time? (Assume they are all small…)*



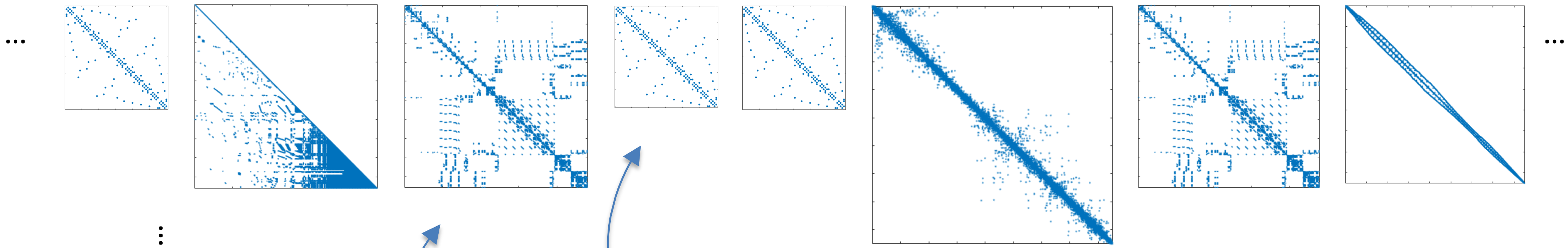
- Design a **batched SpMV kernel**.
    - *Process a large number of data-independent problems in parallel.*
    - *Are the problems*
        - Same Size?
        - Same number of nonzeros overall?
        - Same number of nonzeros in every row?
        - Same sparsity pattern?

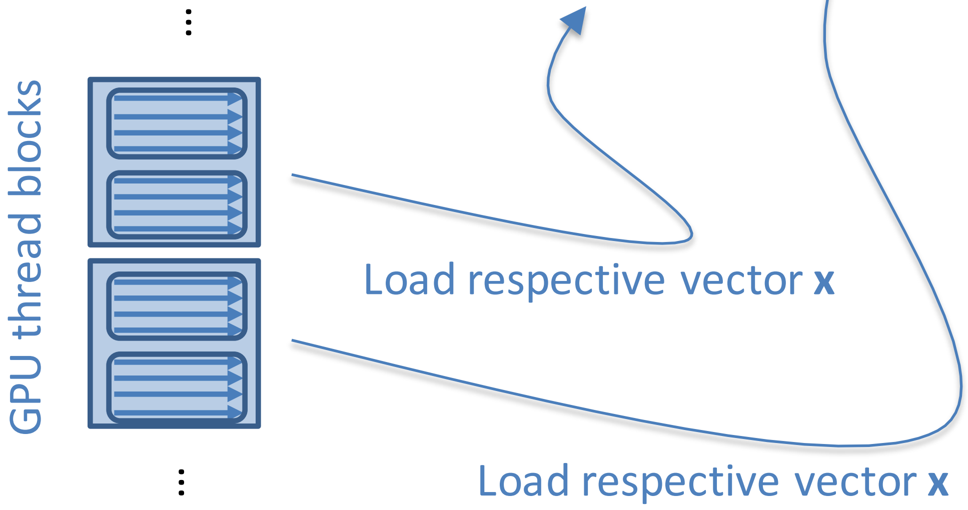
Let's be as flexible as possible!

- Flexible means "everything can be different"
- Every Thread block handles one system
- Memory pointers to distinct systems
- Load input vector **x** into shared memory
- Kernel for all matrices in CSR, COO, ELL

A never ending story: The sparse matrix vector Product (SpMV) on Manycore

- *What if we process many different matrices at a time? (Assume they are all small…)*





All matrices stored the same format.

Let's be as flexible as possible!

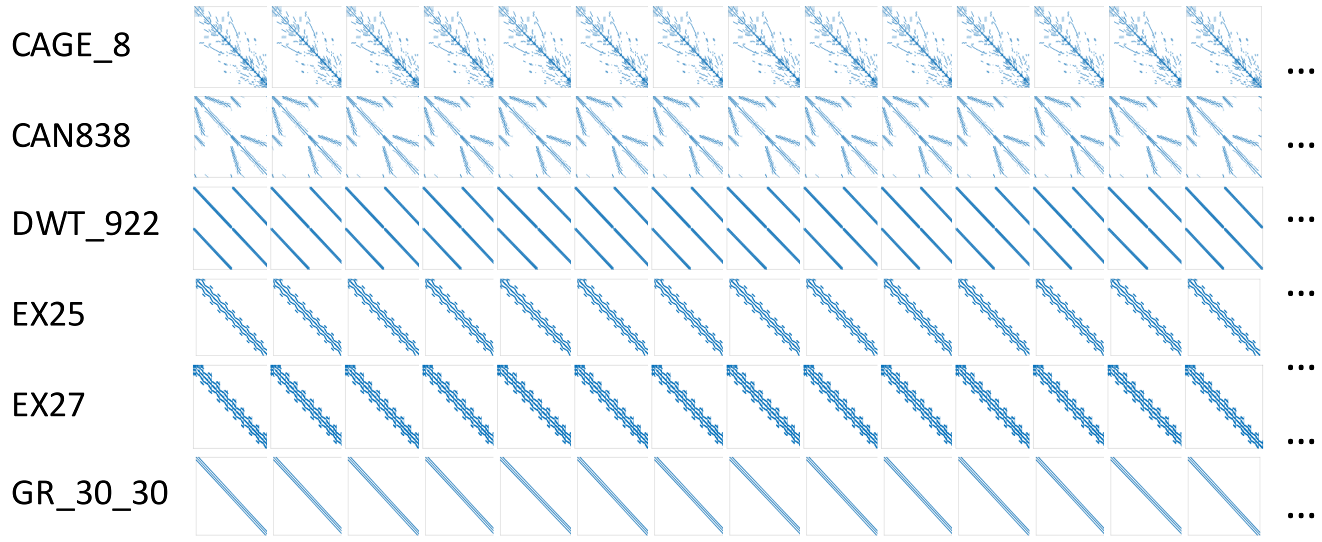
- Flexible means "everything can be different"
- Every Thread block handles one system
- Memory pointers to distinct systems
- Load input vector **x** into shared memory
- Kernel for all matrices in CSR, COO, ELL

Flexible batched SpMV

NVIDIA P100 GPU
56 SMX, 5.3 TF DP
16 GB @ 768GB/s

First experiment:

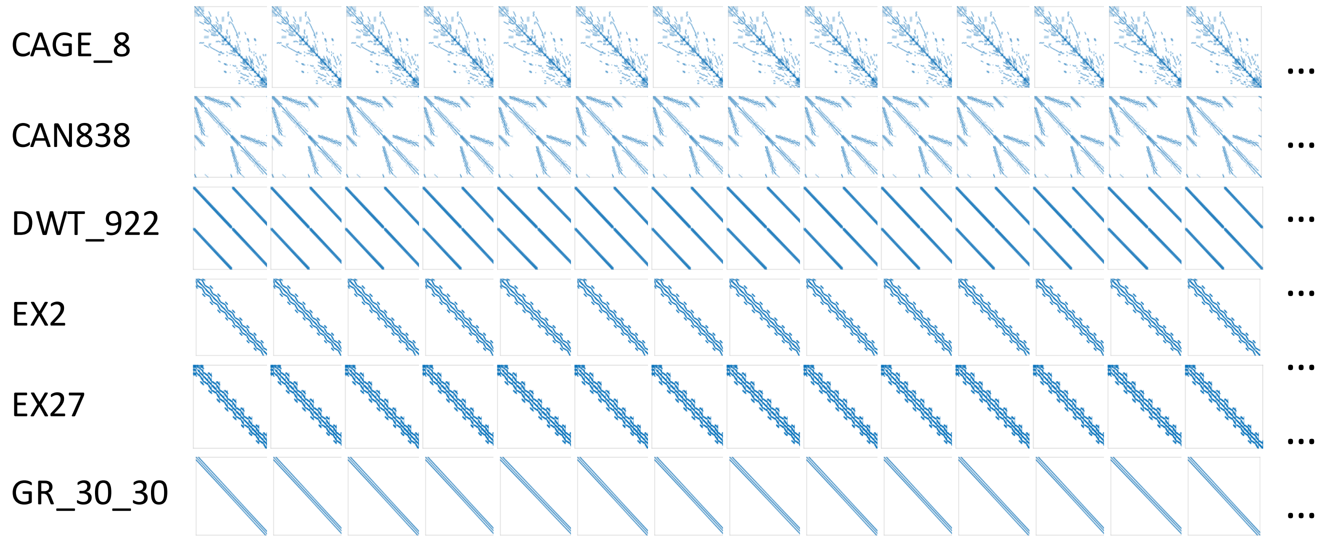
- Use different batched SpMV kernels (COO, CSR, ELL …)
- A batch consisting of the same matrices (*homogeneous batch*)



Flexible batched SpMV

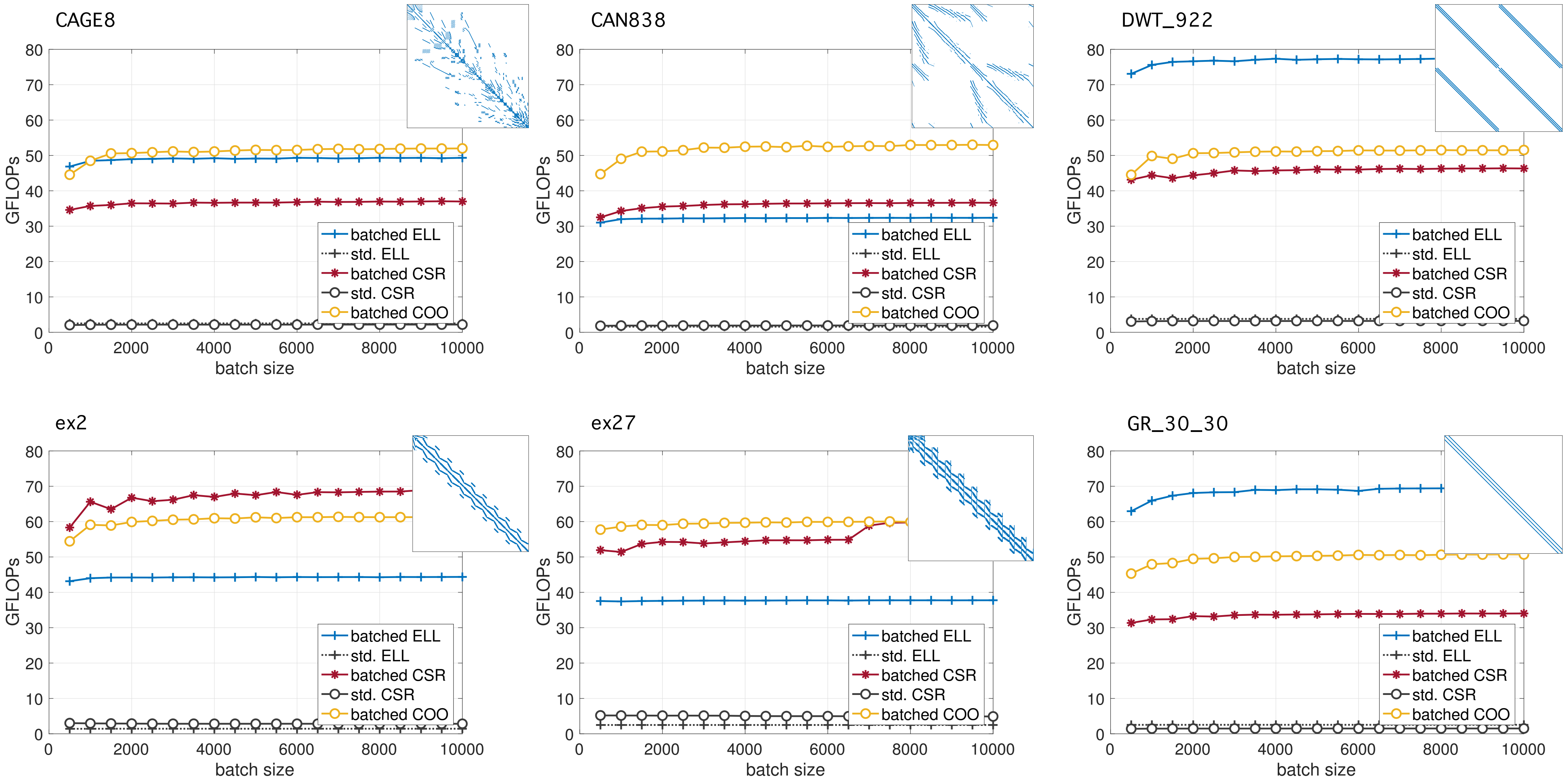
NVIDIA P100 GPU
56 SMX, 5.3 TF DP
16 GB @ 768GB/s

First experiment:
- Use different batched SpMV kernels (COO, CSR, ELL …)
- A batch consisting of the same matrices (*homogeneous batch*)

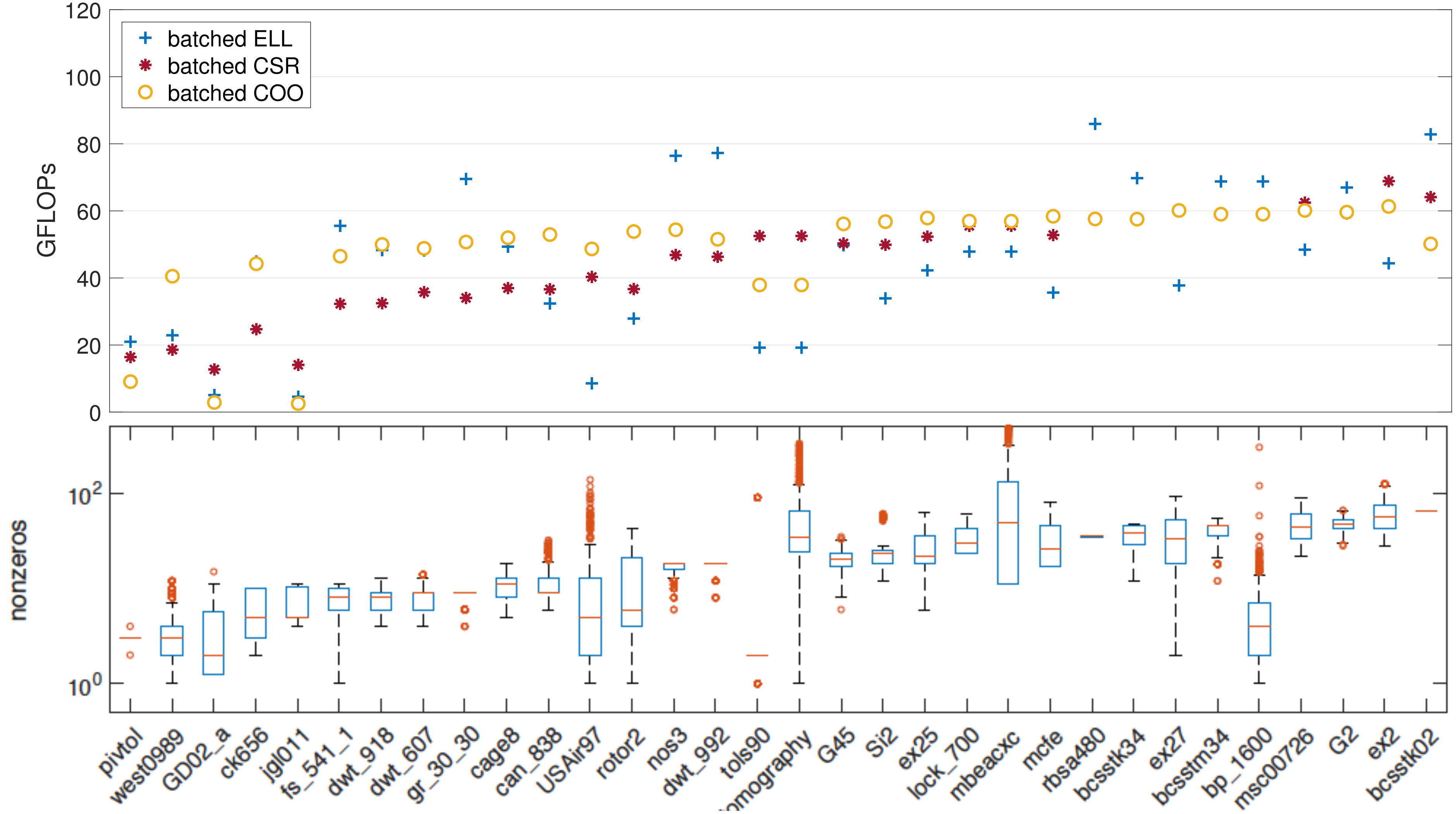


Disclaimer: This is an artificial problem setting!
In a real-world scenario, a homogeneous batched SpMV would be handled as SpMM.

Flexible batched SpMV



Flexible batched SpMV



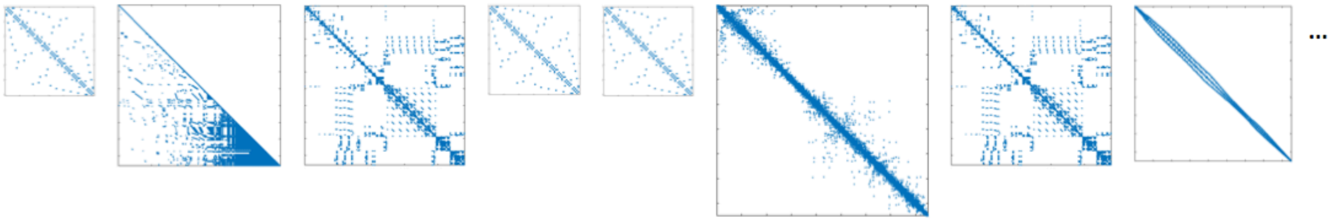
Flexible batched SpMV

Second experiment:

- Use different batched SpMV kernels (COO, CSR, ELL …)
- A batch consisting of different matrices (*in-homogeneous batch*)

  1. “somewhat similar” (similar size, nonzero count)

  2. completely different

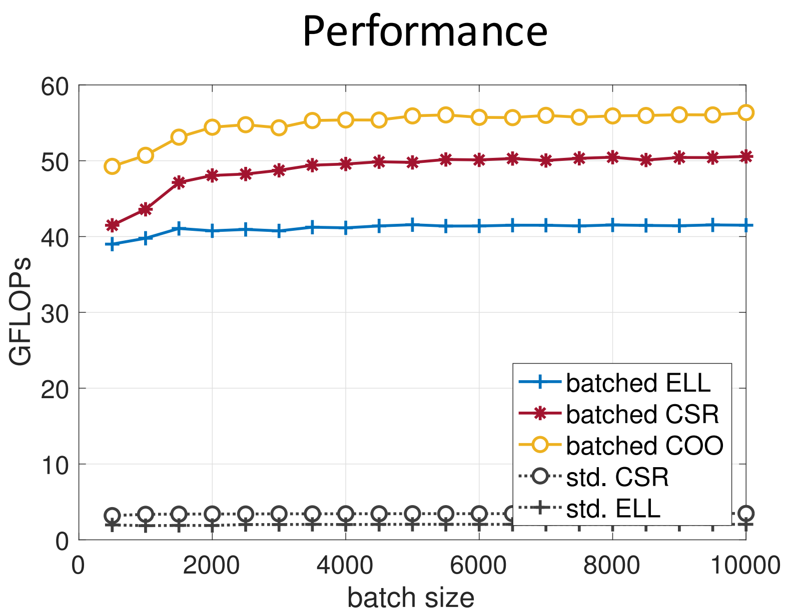


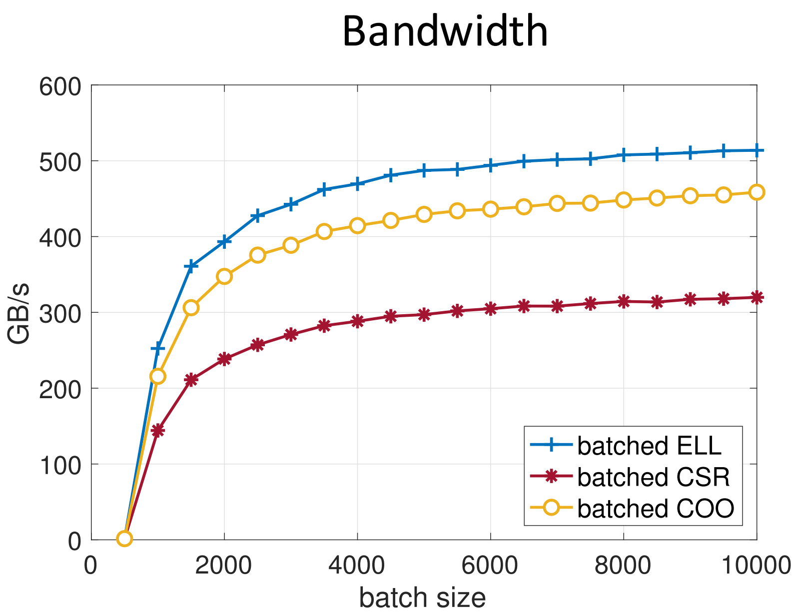
Flexible batched SpMV

Batch of random
"similar-sized" problems

$$n \in [900, 1000]$$

$$nz \in [3000, 40000]$$





Flexible batched SpMV

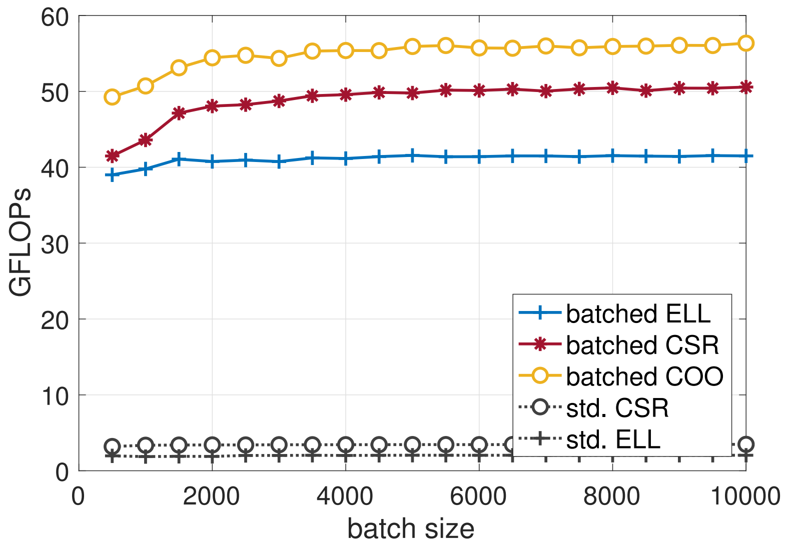
Performance

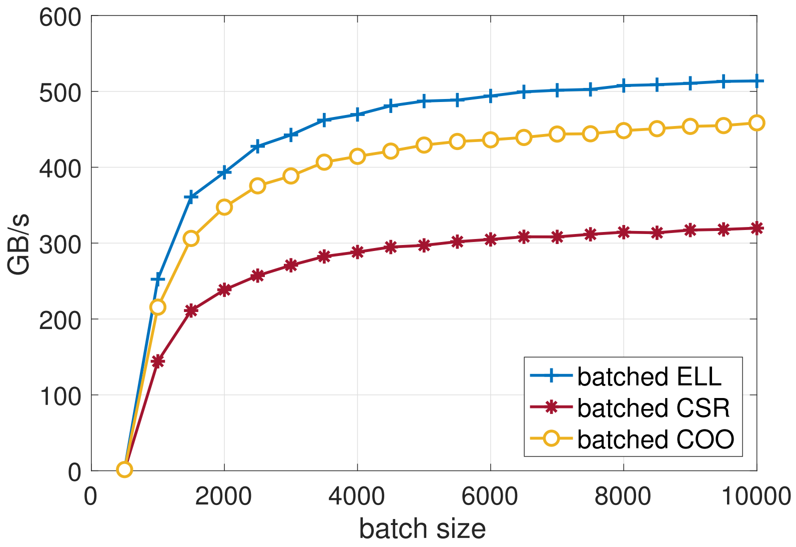
Bandwidth

Batch of random
“similar-sized” problems

$$n \in [900, 1000]$$

$$nz \in [3000, 40000]$$

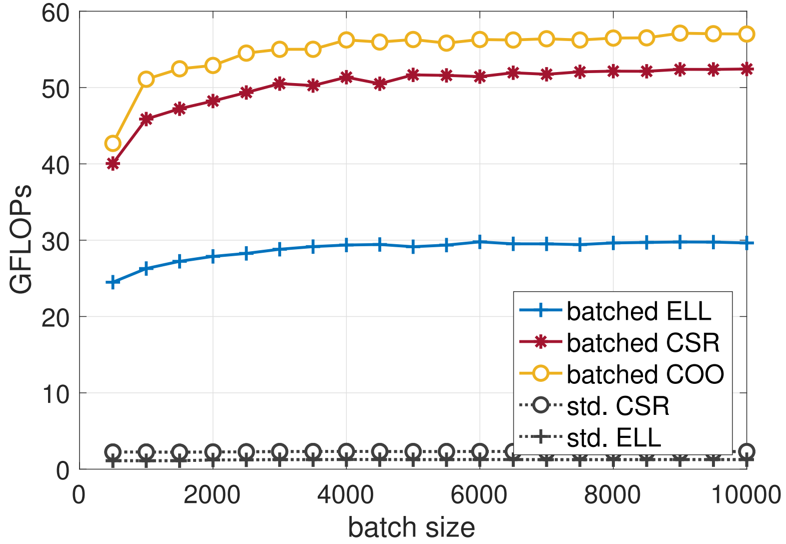


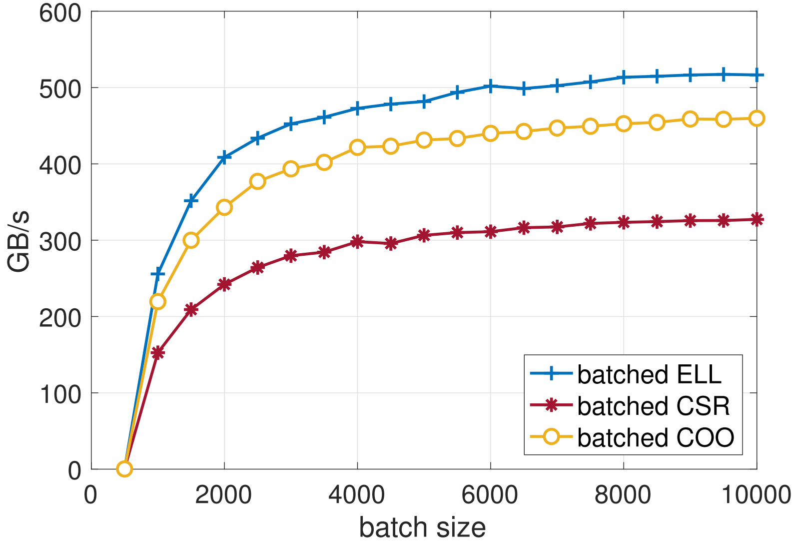


Batch of random
”any-sized” problems.

$$n \in [10, 1000]$$

$$nz \in [100, 40000]$$





ScalA'17: 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems
November 13, 2017

Flexible batched SpMV on GPUs

- Large number of small SpMV simultaneously
- Matrices can be different in size, nnz, pattern
- COO format most suitable for inhomogeneous batches

This work is in Collaboration with:

Karlsruhe Institute of Technology

EXASCALE COMPUTING PROJECT

RESEARCH SPONSORED BY
The Exascale Computing Project
A Collaborative effort of the U.S. Department of Energy Office of Science And the National Nuclear Security Administration
(17-SC-20-SC)

U.S. DEPARTMENT OF ENERGY | Office of Science

National Nuclear Security Administration