SPPEXA Annual Meeting 2016,
January 25th, 2016, Garching, Germany

Feeding of the Thousands – Leveraging the GPU's Computing Power for Sparse Linear Algebra

Hartwig Anzt

INNOVATIVE COMPUTING LABORATORY

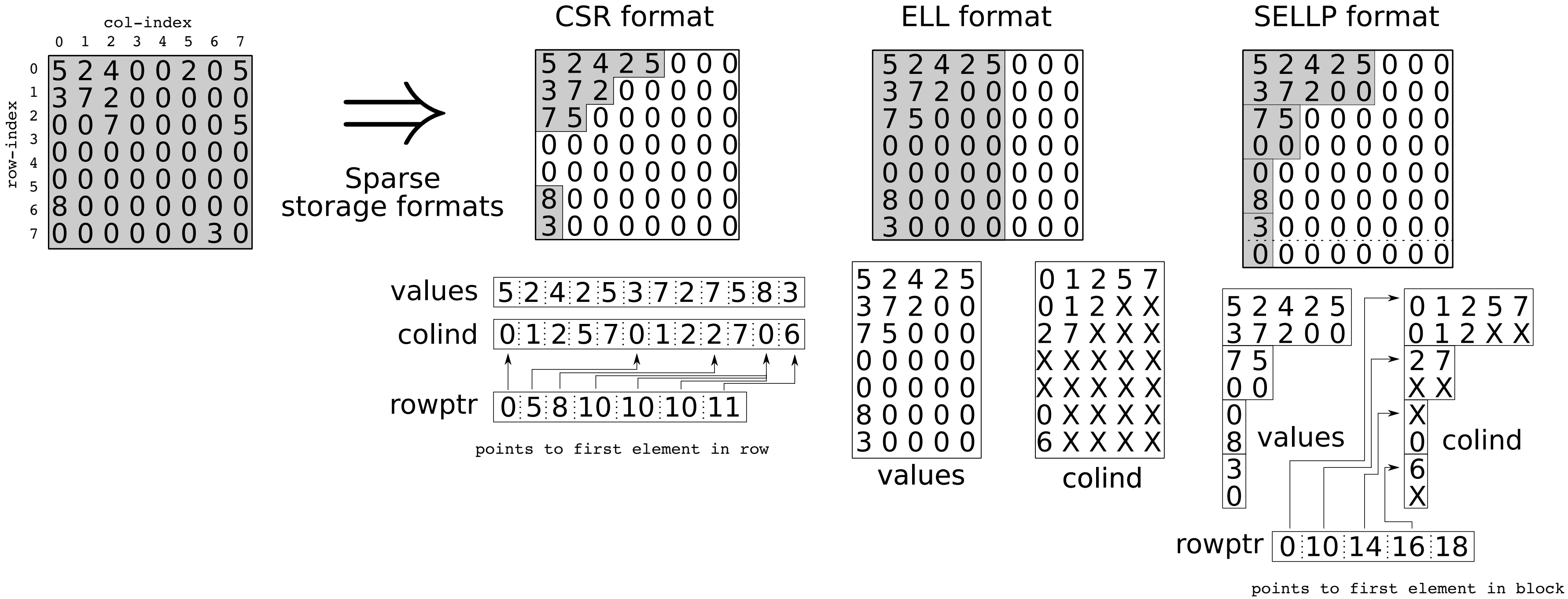
THE UNIVERSITY OF TENNESSEE KNOXVILLE

Sparse Linear Algebra on GPUs

- Inherently parallel operations
  - *axpy, copy, gemv…*
  - *usually memory bound*
  - *Kernel fusion*
- Sparse matrix vector product
  - *often computationally most expensive part*
  - *variety of storage formats, kernels…*
  - *component-thread mapping can result in imbalance*
  - *malicious memory access*
- Bottlenecks
  - *sequential operations, unstructured, random memory access*
  - *incomplete factorizations (ILU/IC)*
  - *sparse triangular solves*

Sparse Matrix Vector Product (SpMV)

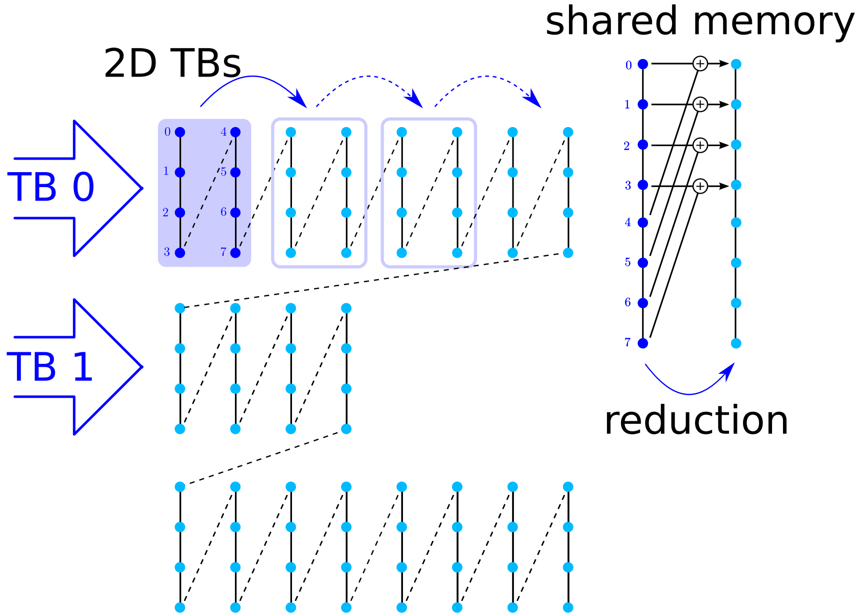
- Sliced Ellpack (SELL) format as trade-off between CSR and Ellpack
- Sorting can improve load-balancing



Kreutzer et al.: ***A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units***, *SISC 36(5), 2014.*

Sparse Matrix Vector Product (SpMV)

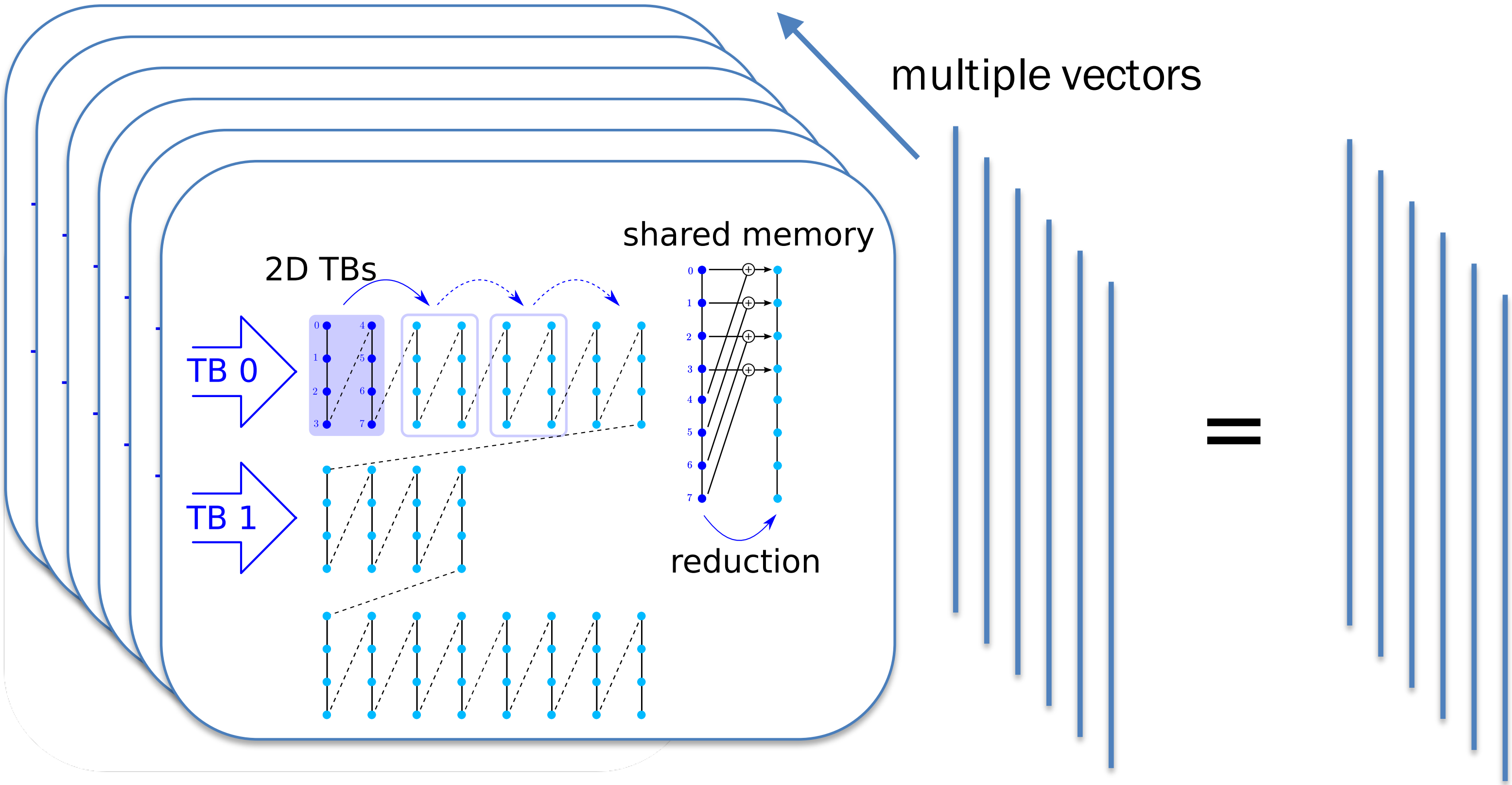
- Assign multiple threads to each row
- 2-dimensional thread blocks



Kreutzer et al.: ***A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units***, *SISC 36(5), 2014.*

Sparse Matrix Vector Product with multiple Vectors (SpMM)

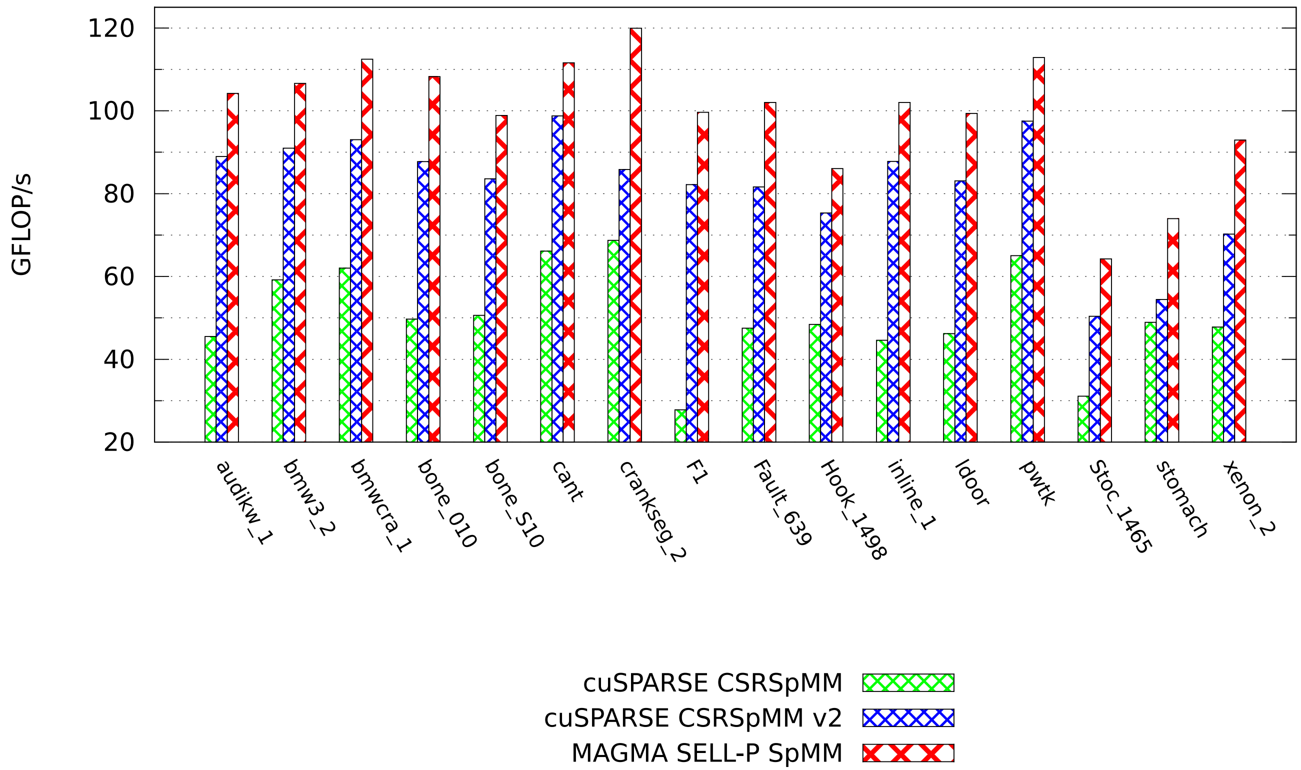
- 3-dimensional thread blocks for processing multiple vectors simultaneously



*Anzt et al.: **Energy efficiency and performance frontiers for sparse computations on GPU supercomputers**, PMAM 2015.*

Sparse Matrix Vector Product with multiple Vectors (SpMM)

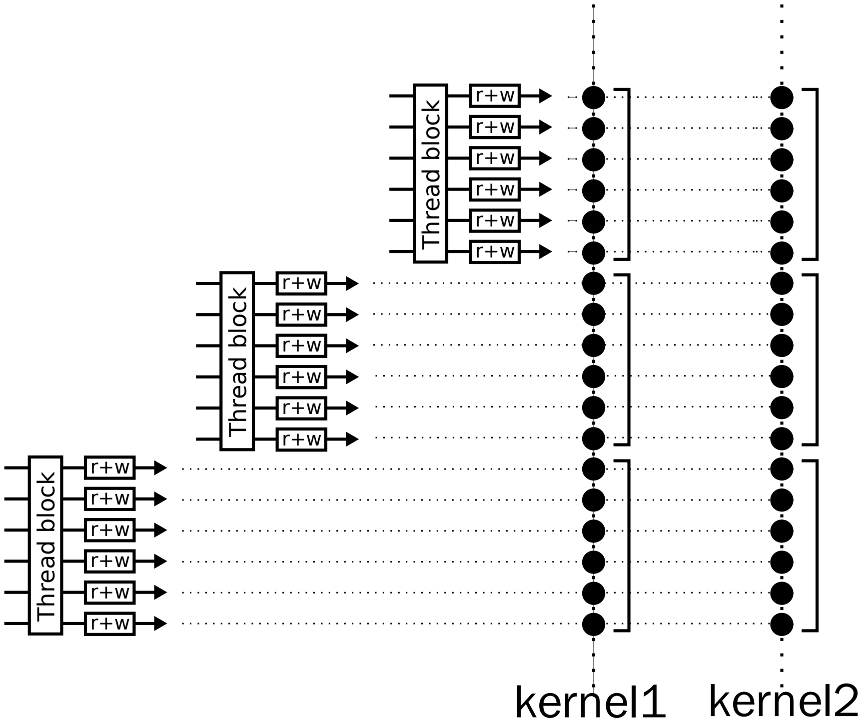
- 3-dimensional thread blocks for processing multiple vectors simultaneously
- Performance on NVIDIA K40, 64 vectors, DP:



Anzt et al.: **Energy efficiency and performance frontiers for sparse computations on GPU supercomputers,** PMAM 2015.

Kernel Fusion in Sparse Iterative Algorithms

- Memory bandwidth in many cases the performance bottleneck.
- Sequence of consecutive vector updates (BLAS 1) benefits from enhanced data locality.
- Design of algorithm-specific kernels.



Kernel Fusion in Sparse Iterative Algorithms

- Memory bandwidth in many cases the performance bottleneck
- Sequence of consecutive vector updates (BLAS 1) benefits from enhanced data locality

```
while( ( k < maxiter ) && ( res > epsilon ) ){
  Scalar_SpMV <<<Gs,Bs>>> ( n, rowA, colA, valA, d, z );
  tmp = cublasSdot ( n, d, 1, z, 1 );
  rho = beta / tmp;
  gamma = beta;
  cublasSaxpy (n,  rho, d, 1, x, 1 );
  cublasSaxpy (n, -rho, z, 1, r, 1 );
  tmp = cublasSdot ( n, r, 1, r, 1 );
  beta = tmp;
  alpha = beta / gamma;
  cublasSscal (n, alpha, d, 1 );
  cublasSaxpy (n, one, r, 1, d, 1 );
  res = sqrt( beta );
  k++;
} // end-while
```

```
while( ( k < maxiter ) && ( res > epsilon ) ){
  scalar_fusion_1 <<<Gs, Bs, Ms>>> ( n, rowA, colA, valA,
                                     d, z, beta, rho, gamma, vtmp );
  fusion_2 ( Gs, Bs, Ms, n, beta, rho, vtmp );

  fusion_3 <<<Gs, Bs, Ms>>> ( n, rho, d, x, z, r, vtmp );
  fusion_4 ( Gs, Bs, Ms, n, vtmp, vtmp2 );

  fusion_5 <<<Gs, Bs>>> ( n, beta, gamma, alpha,
                                              d, r, vtmp );
  cudaMemcopy( &res, beta, sizeof(float),
                                 cudaMemcpyDeviceToHost );
  res = sqrt( beta );
  k ++;
} // end-while
```

*Aliaga et al.: **Reformulated Conjugate Gradient for the Energy-Aware Solution of Linear Systems on GPUs**, Parallel Processing (ICPP), 2013.*

Kernel Fusion in Sparse Iterative Algorithms

- **Which operations can be merged into a single kernel?**
  - kernels compatible in terms of component-thread mapping
  - example classification for Jacobi-CG:

Operation		Input vector(s)		Output
		x	y/M^{-1}	y/α
AXPY	$y = \alpha x + y$	mapped	mapped	mapped
COPY	$y = x$	mapped	mapped	mapped
DOT	$\alpha = \langle x, y \rangle$	mapped	mapped	unmapped
Jacobi-Prec	$y = M^{-1} x$	mapped	mapped	mapped
SpMV CSR	$y = Ax$	unmapped	-	mapped
SpMV ELL	$y = Ax$	unmapped	-	mapped
SpMV SELL-P	$y = Ax$	unmapped	-	unmapped

*Aliaga et al.: **Systematic Fusion of CUDA Kernels for Iterative Sparse Linear System Solvers**, Euro-Par 2015, LLNCS 9233, 2015.*

Kernel Fusion in Sparse Iterative Algorithms

- Which operations can be merged into a single kernel?
- Which kernels do we want to merge?
  - performance vs. flexibility…

Operation		Input vector(s)		Output
		x	y/M^{-1}	y/α
AXPY	$y = \alpha x + y$	mapped	mapped	mapped
COPY	$y = x$	mapped	mapped	mapped
DOT	$\alpha = \langle x, y \rangle$	mapped	mapped	unmapped
Jacobi-Prec	$y = M^{-1} x$	mapped	mapped	mapped
SpMV CSR	$y = Ax$	unmapped	-	mapped
SpMV ELL	$y = Ax$	unmapped	-	mapped
SpMV SELL-P	$y = Ax$	unmapped	-	unmapped

Code Jacobi-preconditioned J-CG, J-BiCGSTAB, J-IDR, J-GMRES…? How about ILU/IC?

Kernel Fusion in Sparse Iterative Algorithms

- Which operations can be merged into a single kernel?
- Which kernels do we want to merge?
  - performance vs. flexibility…

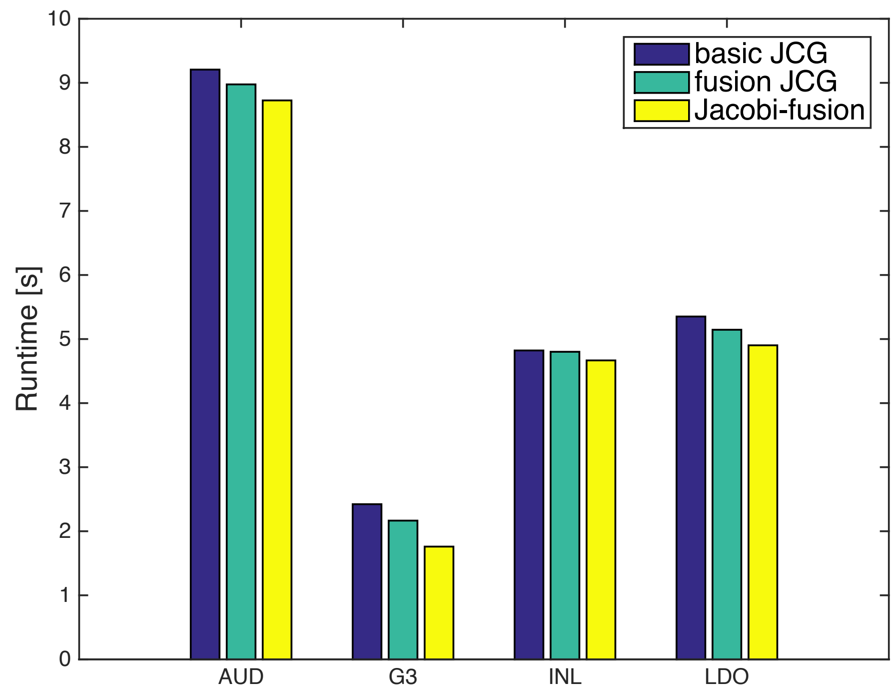
Operation		Input vector(s)		Output
		x	y/M^{-1}	y/α
AXPY	$y = \alpha x + y$	mapped	mapped	mapped
COPY	$y = x$	mapped	mapped	mapped
DOT	$\alpha = \langle x, y \rangle$	mapped	mapped	unmapped
Jacobi-Prec	$y = M^{-1}x$	mapped	mapped	mapped
SpMV CSR	$y = Ax$	unmapped	-	mapped
SpMV ELL	$y = Ax$	unmapped	-	mapped
SpMV SELL-P	$y = Ax$	unmapped	-	unmapped

One stand-alone code for each SpMV kernel?

Kernel Fusion in Sparse Iterative Algorithms

- Which operations can be merged into a single kernel?
- Which kernels do we want to merge?
  - performance vs. flexibility…

Matrix	Size	Nonzeros
AUD	943,695	77,651,847
G3	1,585,478	7,660,826
INL	503,712	36,816,342
LDO	952,203	46,522,475

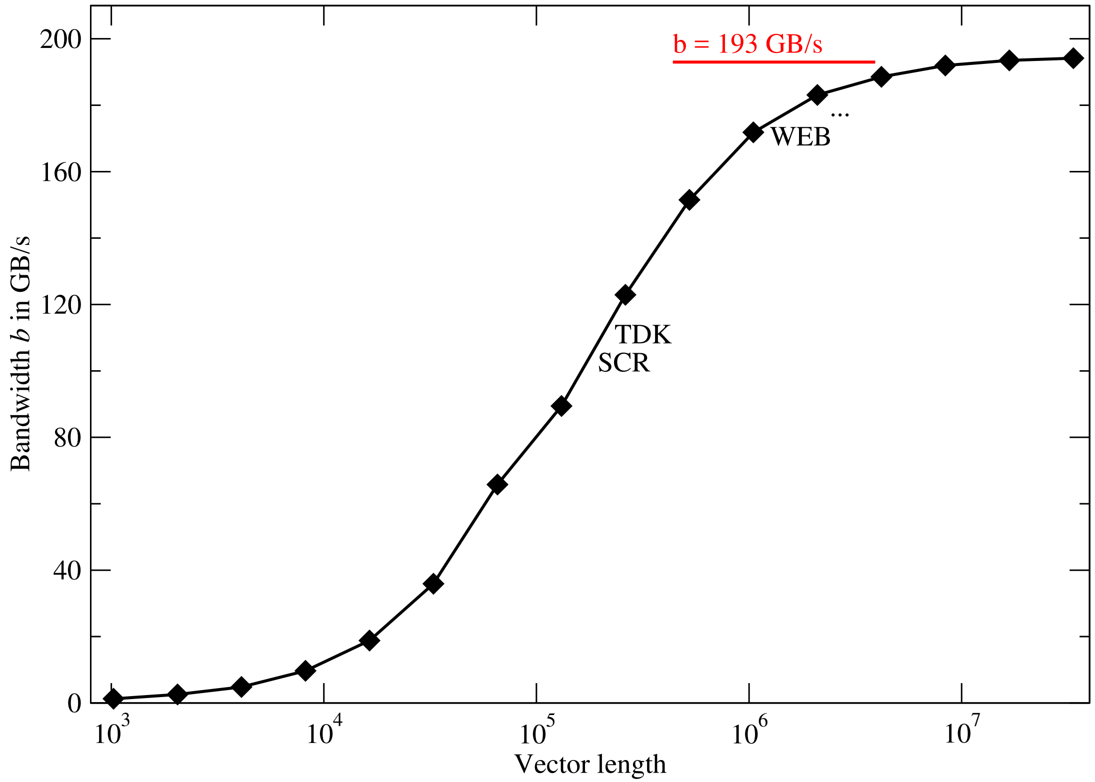


- Benefits from fusion Jacobi-preconditioner for very large and sparse matrices.
- Smaller benefits for more complex algorithms (BiCGSTAB, CGS, QMR, IDR…)

Kernel Fusion in Sparse Iterative Algorithms

- How close can kernel fusion bring us to the
  theoretical performance bound induced by memory bandwidth?
- Cooperation with *Moritz Kreutzer, Eduardo Ponce*.
- NVIDIA K40, theoretical bandwidth: 288 GB/s…

Matrix	Size
SCR	170,998
TDK	204,316
WEB	1,000,005
DIE	1,157,456



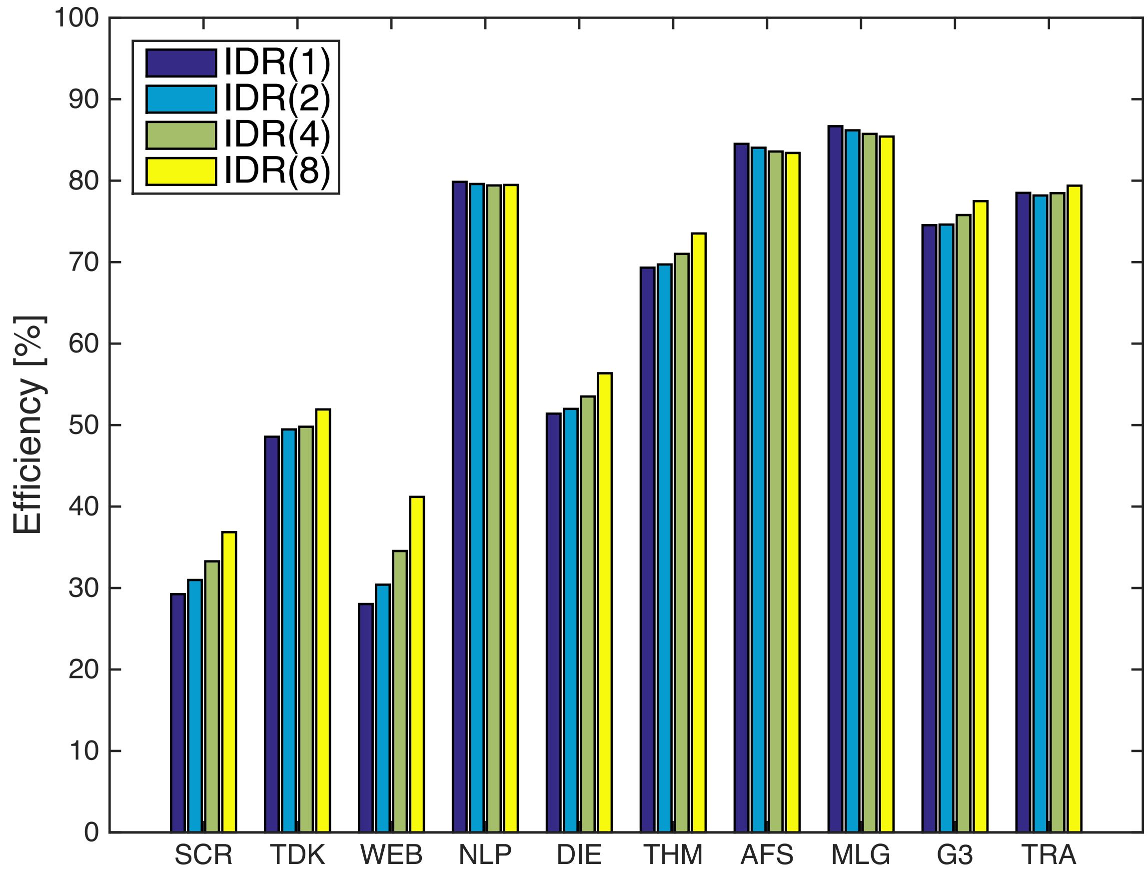
Kernel Fusion in Sparse Iterative Algorithms

- **Efficiency compared to roofline performance model:** $P = \min(P^{\text{peak}}; Ib)$ Gflop/s
  P theoretical compute peak, *I* intensity, *b* bandwidth

IDR(s) general Krylov solver

Moritz Kreutzer, Eduardo Ponce

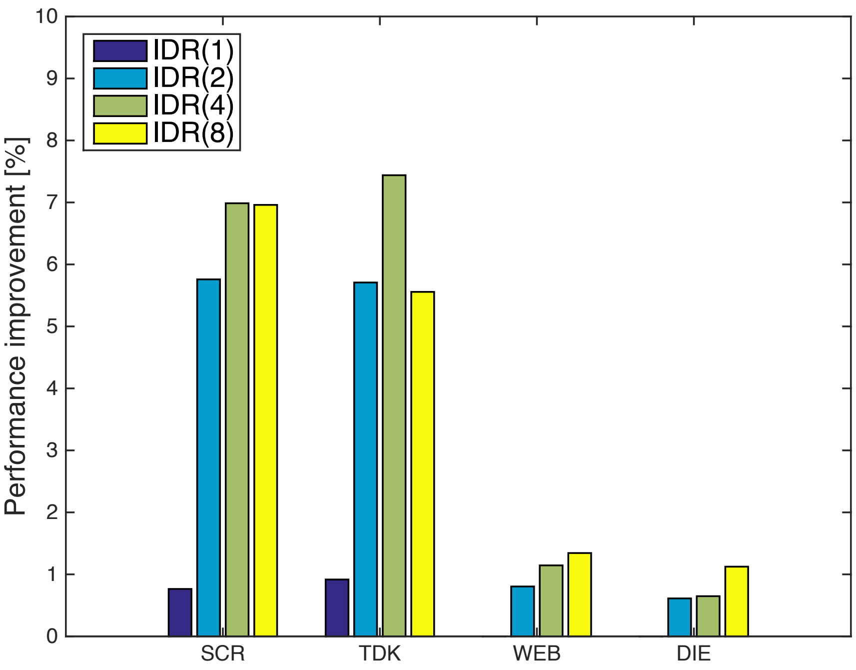
Matrix	Size	Nonzeros
SCR	170,998	958,936
TDK	204,316	2,846,228
WEB	1,000,005	3,105,536
NLP	1,062,400	28,704,672
DIE	1,157,456	48,538,952
THM	1,228,045	8,580,313
AFS	1,508,065	52,672,325
MLG	1,504,002	110,879,972
G3	1,585,478	7,660,826
TRA	1,602,111	23,500,731



Kernel Overlap in Sparse Iterative Algorithms

- **Concurrent kernel execution** to exploit unused GPU compute resources.
- Rare in sparse linear algebra (most algorithms compose of memory-bound operations).
- Small benefits if datasets too small to saturate memory bandwidth.

Matrix	Size
SCR	170,998
TDK	204,316
WEB	1,000,005
DIE	1,157,456



Bottlenecks

- Operations not parallelizable to GPU thread concurrency:
  - *sequential operations*
  - *unstructured, random memory access*
  - *incomplete factorizations (ILU/IC)*
  - *sparse triangular solves*

Bottlenecks

- Operations not parallelizable to GPU thread concurrency:
  - *sequential operations*
  - *unstructured, random memory access*
  - *incomplete factorizations (ILU/IC)*
  - *sparse triangular solves*

- Don't even try – rethink the problem!

  - A different algorithm may take you to the same goal.
  - Choose algorithms with fine-grained parallelism, avoid synchronizations.
  - Sparse iterative solvers provide approximations -- relax the bit-wise reproducibility criterion.

Most popular Example: Iterative ILU (Chow et al.)

*Chow et al., **Fine-grained Parallel Incomplete LU Factorization**, SIAM Journal on Scientific Computing, 37, pp. C169-C193 (2015).*

Bottlenecks

- **Example: sparse triangular solves in ILU preconditioning:**
  - inherently sequential
  - parallelism from level-scheduling/multi-color ordering
  - unable to exploit fine-grained parallelism of GPUs

Bottlenecks

- **Example: sparse triangular solves in ILU preconditioning:**
  - inherently sequential
  - parallelism from level-scheduling/multi-color ordering
  - unable to exploit fine-grained parallelism of GPUs
- **Take an unconventional approach:**

  Approximate sparse triangular solves

  - Replace forward/backward substitutions with **iterative method**.
  - **Low solution accuracy required** as LU ≈ A typically only a rough approximation.
  - Better **scalability** of iterative methods.
  - **Jacobi converges** as spectral radius of iteration matrix smaller 1:

$$x^{k+1} = D^{-1}b + Mx^k$$

$$M_L = D_L^{-1}(D_L - L) = I - L$$
$$M_U = D_U^{-1}(D_U - U) = I - D_U^{-1}U$$

  - **Performance depends on SpMV.**

Bottlenecks

- **Example: sparse triangular solves in ILU preconditioning:**
  - inherently sequential
  - parallelism from level-scheduling/multi-color ordering
  - unable to exploit fine-grained parallelism of GPUs
- **Take an unconventional approach:**
  Approximate sparse triangular solves

Matrix	Exact IC		10 Jacobi sweeps	
Top-level PCG:	Iterations	Runtime	Iterations	Runtime
Laplace 3D 27pt	58	1.83	63	1.14
parabolic_fem	645	37.24	721	7.29
thermal2	1771	305.58	2457	67.41
G3_circuit	1625	45.60	1647	35.43

Anzt et al., ***Iterative Sparse Triangular Solves for Preconditioning****,* *Euro-Par 2015.*

Summary

- SpMV optimization central challenge in iterative sparse linear algebra.
- Algorithms memory bound, often benefit from kernel fusion.
- Trade-off between performance and flexibility:
  - SpMV /Jacobi as building block enhances modularity.
- Kernel fusion can bring performance close to **theoretical bound.**
- Concurrent kernel execution only beneficial for small problems.
- We need unconventional approaches for bottleneck-operations.
  - Iterative ILU generation (Chow et al.)
  - Iterative sparse triangular solves for ILU/IC.

This research is based on a cooperation with Enrique Quintana-Ortí from the University Jaume I, Edmond Chow from Georgia Tech, and Moritz Kreutzer from the University of Erlangen.

U.S. DEPARTMENT OF ENERGY

ICL THE UNIVERSITY OF TENNESSEE KNOXVILLE