Increasing arithmetic intensity in algebra-based CFD

Octavi Pavon[1], Marcial Mosqueda-Otero[1], Josep Plana-Riu[1], Àdel Alsalti-Baldellou[2], **Xavier Álvarez-Farré**[3], Andrey Gorobets[4], F. Xavier Trias[1], Guillem Colomer[1], Assensi Oliva[1]

[1] Heat and Mass Transfer Technological Center, Technical University of Catalonia, Terrassa (Barcelona), Spain
[2] Department of Civil, Environmental and Architectural Engineering, University of Padova, Italy
[3] High-Performance Computing and Visualization Team, SURF, Amsterdam, The Netherlands
[4] Keldysh Institute of Applied Mathematics, Moscow, Russia

37th International Conference on Parallel Computational Fluid Dynamics (ParCFD 2026)
MS7: Modern CFD on AI-Driven HPC Systems: Challenges, Bottlenecks and Successes
November 6th, 2026, Nantes, France

SURF

CTTC UPC
Centre Tecnològic de Transferència de Calor
UNIVERSITAT POLITÈCNICA DE CATALUNYA

UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Spoiler: one operator, many vectors

If the operator has Kronecker structure ...

$$\hat{\mathbf{A}} = \mathbf{I}_K \otimes \mathbf{A}, \qquad \mathbf{A} \in \mathbb{R}^{n \times n}$$

Spoiler: one operator, many vectors

If the operator has Kronecker structure ...

$$\hat{\mathbf{A}} = \mathbf{I}_K \otimes \mathbf{A}, \qquad \mathbf{A} \in \mathbb{R}^{n \times n}$$

... then one long SpMV is one SpMM

$$\underbrace{\boldsymbol{y} = \hat{\mathbf{A}}\,\boldsymbol{x}}_{\text{SpMV, size } Kn} \iff \underbrace{\mathbf{Y} = \mathbf{A}\,\mathbf{X}}_{\text{SpMM, } K \text{ columns}}$$

$$\boldsymbol{x} = (\boldsymbol{x}_1; \ldots; \boldsymbol{x}_K), \quad \mathbf{X} = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_K \end{bmatrix}$$

- $\mathbf{A}$ is read once for all K vectors: same flops, fewer bytes.
- Column by column, $\boldsymbol{y}_k = \mathbf{A}\,\boldsymbol{x}_k$: no approximation, same result.

Spoiler: one operator, many vectors

If the operator has Kronecker structure …

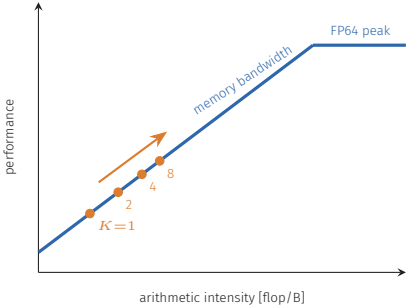
$$\hat{\mathbf{A}} = \mathbf{I}_K \otimes \mathbf{A}, \qquad \mathbf{A} \in \mathbb{R}^{n \times n}$$

… then one long SpMV is one SpMM

$$\underbrace{\boldsymbol{y} = \hat{\mathbf{A}}\,\boldsymbol{x}}_{\text{SpMV, size } Kn} \iff \underbrace{\mathbf{Y} = \mathbf{A}\,\mathbf{X}}_{\text{SpMM, } K \text{ columns}}$$

$$\boldsymbol{x} = (\boldsymbol{x}_1; \ldots; \boldsymbol{x}_K), \quad \mathbf{X} = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_K \end{bmatrix}$$

- $\mathbf{A}$ is read once for all K vectors: same flops, fewer bytes.
- Column by column, $\boldsymbol{y}_k = \mathbf{A}\,\boldsymbol{x}_k$: no approximation, same result.

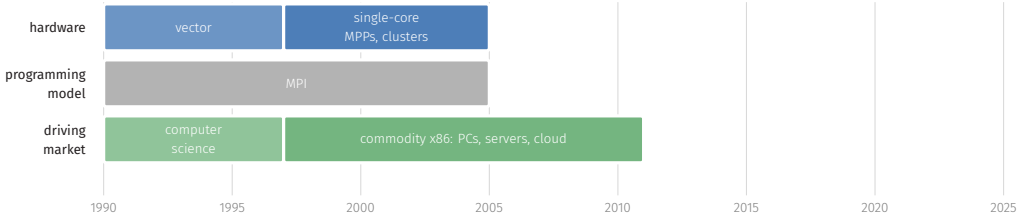


Schematic, CSR in FP64; roofs not to scale.

Modern systems, modern CFD

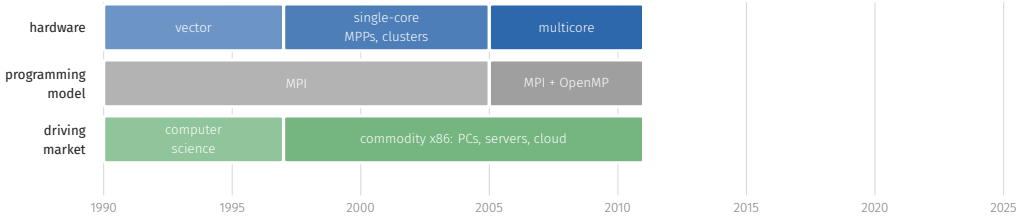
What drives HPC hardware?

When did your code last get faster without you touching it?



- Until about 2005 the CPU was enough: vector units, then clock rate, made the same code faster.

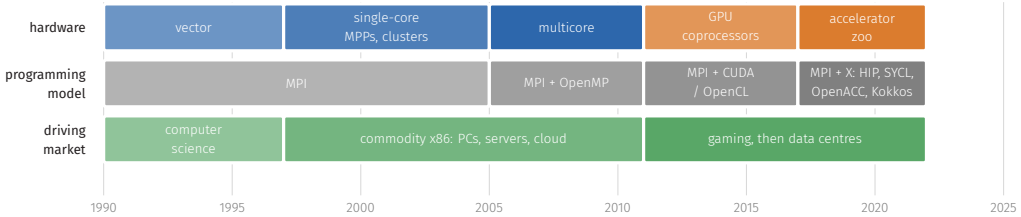
When did your code last get faster without you touching it?



- Until about 2005 the CPU was enough: vector units, then clock rate, made the same code faster.
- Then the clock stalled and cores multiplied: the first rewrite, threads on top of MPI.

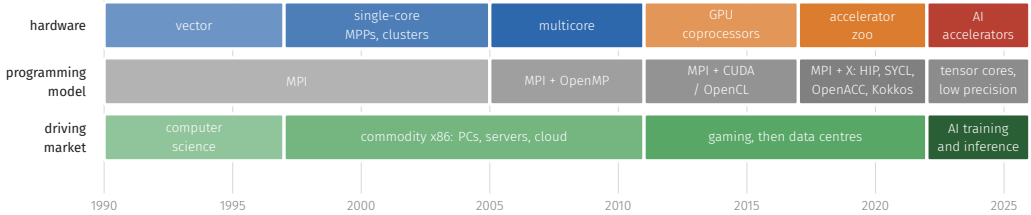
What drives HPC hardware?

When did your code last get faster without you touching it?



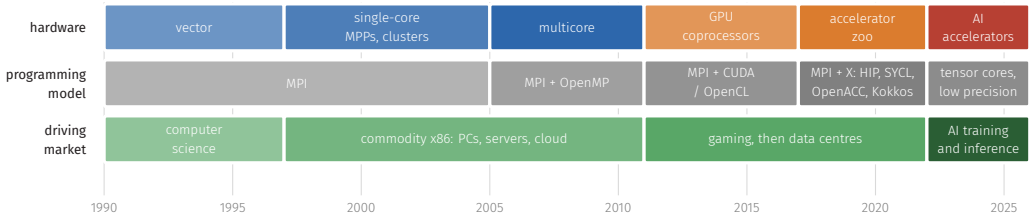
- Until about 2005 the CPU was enough: vector units, then clock rate, made the same code faster.
- Then the clock stalled and cores multiplied: the first rewrite, threads on top of MPI.
- Then accelerators from gaming, then a zoo, each with its own programming model.

When did your code last get faster without you touching it?



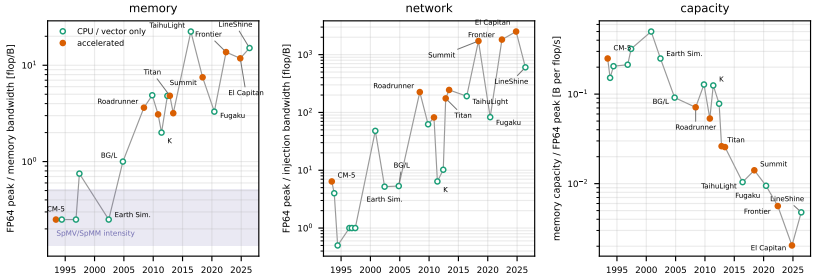
- Until about 2005 the CPU was enough: vector units, then clock rate, made the same code faster.
- Then the clock stalled and cores multiplied: the first rewrite, threads on top of MPI.
- Then accelerators from gaming, then a zoo, each with its own programming model.
- Now the market is AI: low precision grows by orders of magnitude, FP64 grows slowly or is cut.

When did your code last get faster without you touching it?



- Until about 2005 the CPU was enough: vector units, then clock rate, made the same code faster.
- Then the clock stalled and cores multiplied: the first rewrite, threads on top of MPI.
- Then accelerators from gaming, then a zoo, each with its own programming model.
- Now the market is AI: low precision grows by orders of magnitude, FP64 grows slowly or is cut.
- HPC shaped a few parts (A64FX with HBM, MI300A for El Capitan, FP64 tensor cores in A100), but it is a minority customer.

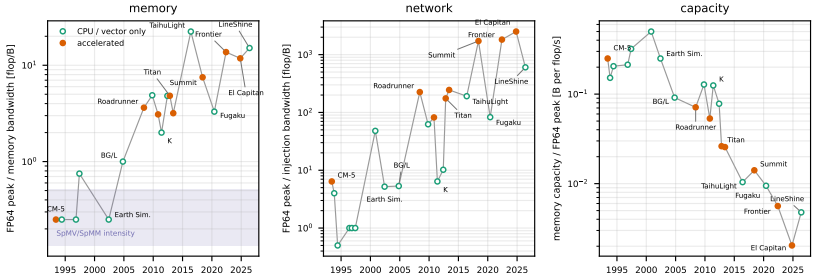
Thirty years of TOP500 number ones, per node



- Until 2002 a node delivered 1–4 bytes per flop: SpMV ran close to peak.

Per node, each #1 at its first list as #1. TOP500 lists 1993–2026 and system documentation. LineShine: HBM only, matrix-unit peak.

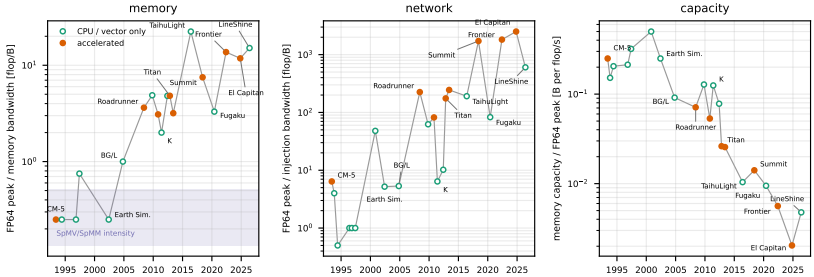
Thirty years of TOP500 number ones, per node



- Until 2002 a node delivered 1–4 bytes per flop: SpMV ran close to peak.
- Today: 7–15 flop/B (Summit to LineShine); SpMV gets 1–3 %.

Per node, each #1 at its first list as #1. TOP500 lists 1993–2026 and system documentation. LineShine: HBM only, matrix-unit peak.

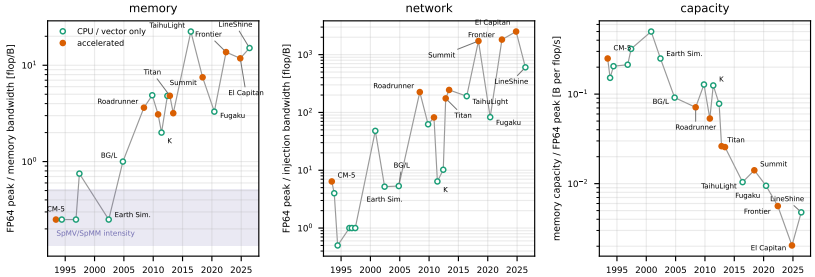
Thirty years of TOP500 number ones, per node



- Until 2002 a node delivered 1–4 bytes per flop: SpMV ran close to peak.
- Today: 7–15 flop/B (Summit to LineShine); SpMV gets 1–3 %.
- The network moved 2–3 orders of magnitude: up to 2500 flop/B (El Capitan).

Per node, each #1 at its first list as #1. TOP500 lists 1993–2026 and system documentation. LineShine: HBM only, matrix-unit peak.

Thirty years of TOP500 number ones, per node



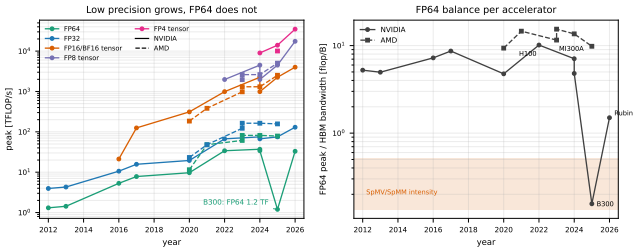
- Until 2002 a node delivered 1–4 bytes per flop: SpMV ran close to peak.
- Today: 7–15 flop/B (Summit to LineShine); SpMV gets 1–3 %.

- The network moved 2–3 orders of magnitude: up to 2500 flop/B (El Capitan).

- Capacity per flop fell about 100×.
- Bandwidth-rich CPUs (K; Fugaku, LineShine with HBM) bend the curves, not reverse them.

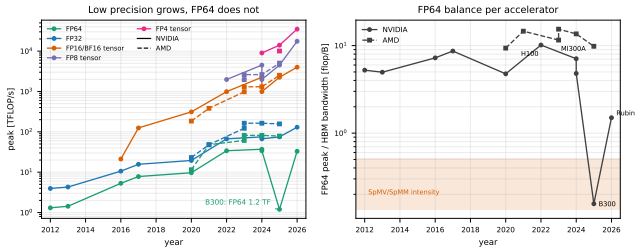
Per node, each #1 at its first list as #1. TOP500 lists 1993–2026 and system documentation. LineShine: HBM only, matrix-unit peak.

The AI era: FP64 is negotiable, memory is expensive



Vendor datasheets (tensor-core peaks without 2:4 sparsity); NVIDIA blog on FP64 emulation in cuBLAS (Oct 2025); TrendForce, Dec 2025 and Feb 2026; Counterpoint, Feb 2026.

The AI era: FP64 is negotiable, memory is expensive

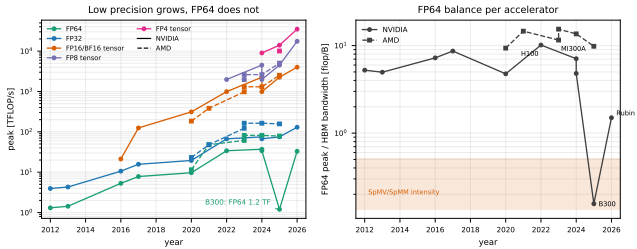


Flops

- Tensor-core FP16 is 30–120× FP64; B300 cuts FP64 to 1.2 TF/s, inside the SpMV band.
- FP64 matrix products can be emulated on integer tensor cores; sparse kernels have none, and their limit is bandwidth.

Vendor datasheets (tensor-core peaks without 2:4 sparsity); NVIDIA blog on FP64 emulation in cuBLAS (Oct 2025); TrendForce, Dec 2025 and Feb 2026; Counterpoint, Feb 2026.

The AI era: FP64 is negotiable, memory is expensive



Flops

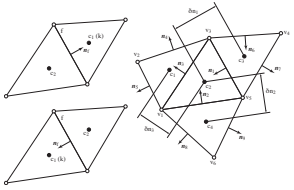
- Tensor-core FP16 is 30–120× FP64; B300 cuts FP64 to 1.2 TF/s, inside the SpMV band.
- FP64 matrix products can be emulated on integer tensor cores; sparse kernels have none, and their limit is bandwidth.

Bytes

- Server DRAM +90 % in one quarter (1Q26): a 64 GB RDIMM went from $450 to over $900.
- HBM3e costs 4–5× DDR5: AI moved DRAM capacity to HBM. Footprint is now a cost item.

Vendor datasheets (tensor-core peaks without 2:4 sparsity); NVIDIA blog on FP64 emulation in cuBLAS (Oct 2025); TrendForce, Dec 2025 and Feb 2026; Counterpoint, Feb 2026.

$$\boldsymbol{q} = -\lambda \nabla T \quad \longrightarrow \quad \boldsymbol{q}_s = -\lambda \, \mathbf{G} \, \boldsymbol{T}_c$$

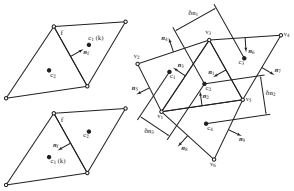


Operators are sparse matrices built once from the mesh; the solver only sees matrices and vectors.

Álvarez-Farré, Gorobets, Trias, Comput. Fluids 214 (2021) 104768

Algebra-based CFD: a time step is SpMV, axpy, dot

$$\boldsymbol{q} = -\lambda \nabla T \quad \longrightarrow \quad \boldsymbol{q}_s = -\lambda\, \mathbf{G}\, \boldsymbol{T}_c$$



Operators are sparse matrices built once from the mesh; the solver only sees matrices and vectors.

Navier–Stokes, collocated, in algebraic form

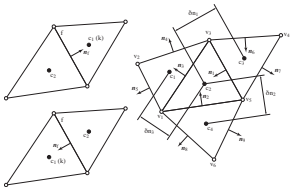
$$\mathbf{M}\, \boldsymbol{u}_s = \mathbf{0},$$

$$\boldsymbol{\Omega} \frac{d\boldsymbol{u}_c}{dt} + \mathbf{C}(\boldsymbol{u}_s)\, \boldsymbol{u}_c + \mathbf{D}\, \boldsymbol{u}_c + \boldsymbol{\Omega}\, \mathbf{G}_c\, \boldsymbol{p}_c = \mathbf{0}$$

Álvarez-Farré, Gorobets, Trias, Comput. Fluids 214 (2021) 104768

Algebra-based CFD: a time step is SpMV, axpy, dot

$$\boldsymbol{q} = -\lambda \nabla T \quad \longrightarrow \quad \boldsymbol{q}_s = -\lambda\, \mathbf{G}\, \boldsymbol{T}_c$$



Operators are sparse matrices built once from the mesh; the solver only sees matrices and vectors.

Navier–Stokes, collocated, in algebraic form

$$\mathbf{M}\,\boldsymbol{u}_s = \mathbf{0},$$

$$\boldsymbol{\Omega}\frac{d\boldsymbol{u}_c}{dt} + \mathbf{C}(\boldsymbol{u}_s)\,\boldsymbol{u}_c + \mathbf{D}\,\boldsymbol{u}_c + \boldsymbol{\Omega}\,\mathbf{G}_c\,\boldsymbol{p}_c = \mathbf{0}$$

One time step

- Apply $\mathbf{M}$, $\mathbf{G}$, $\mathbf{D}$, $\mathbf{C}(\boldsymbol{u}_s)$: SpMV
- Runge–Kutta stages, projection: axpy
- Poisson equation by PCG: SpMV, axpy, dot

SpMV dominates the time step.

Kernel set and portable implementation: F. X. Trias, 09:10.

Álvarez-Farré, Gorobets, Trias, Comput. Fluids 214 (2021) 104768

SpMV sits at the foot of the roofline

SURF

CSR SpMV

Each non-zero brings one FP64 value (8 B) and one int32 column index (4 B) for one multiply-add (2 flop):

$$I \approx \frac{2 \text{ flop}}{12 \text{ B}} = \frac{1}{6} \text{ flop/B}$$

Row pointers and the vectors only lower it: about 0.13 flop/B for a 7-point stencil.

Williams, Waterman, Patterson, Commun. ACM 52(4) (2009).

SpMV sits at the foot of the roofline

SURF

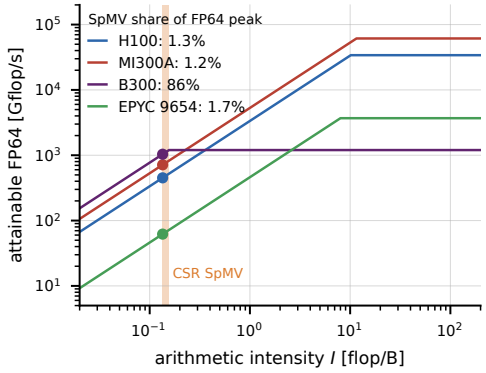
CSR SpMV

Each non-zero brings one FP64 value (8 B) and one int32 column index (4 B) for one multiply-add (2 flop):

$$I \approx \frac{2 \text{ flop}}{12 \text{ B}} = \frac{1}{6} \text{ flop/B}$$

Row pointers and the vectors only lower it: about 0.13 flop/B for a 7-point stencil.

- Bound: $I\beta/\pi \approx$ 1–3 % of FP64 peak on current HPC devices.
- B300 reaches about 85 %, only because its FP64 peak was cut: the time per SpMV is set by bandwidth alone.
- Roofline and roofplane in detail: M. Mosqueda-Otero, 08:50.



Vendor peaks; SpMV point for a 7-point stencil.

Williams, Waterman, Patterson, Commun. ACM 52(4) (2009).

Three pressures on sparse and stencil CFD

Bytes per flop

Peak flops outgrow bandwidth; SpMV runs at 1–3 % of FP64 peak on every current device.

Three pressures on sparse and stencil CFD

Bytes per flop

Peak flops outgrow bandwidth; SpMV runs at 1–3 % of FP64 peak on every current device.

Bytes per euro

Memory is the expensive part of a node; footprint is a cost item.

Three pressures on sparse and stencil CFD

Bytes per flop

Peak flops outgrow bandwidth; SpMV runs at 1–3 % of FP64 peak on every current device.

Bytes per euro

Memory is the expensive part of a node; footprint is a cost item.

Code per architecture

Every hardware era cost a rewrite; the next one will too.

Three pressures on sparse and stencil CFD

Bytes per flop

Peak flops outgrow bandwidth; SpMV runs at 1–3 % of FP64 peak on every current device.

Reuse each matrix byte for K vectors: SpMM.

Bytes per euro

Memory is the expensive part of a node; footprint is a cost item.

Code per architecture

Every hardware era cost a rewrite; the next one will too.

Three pressures on sparse and stencil CFD

Bytes per flop

Peak flops outgrow bandwidth; SpMV runs at 1–3 % of FP64 peak on every current device.

Reuse each matrix byte for K vectors: SpMM.

Bytes per euro

Memory is the expensive part of a node; footprint is a cost item.

Store the operator once for many fields; symmetries and repeated geometry shrink mesh and fields.

Code per architecture

Every hardware era cost a rewrite; the next one will too.

Three pressures on sparse and stencil CFD

SURF

Bytes per flop

Peak flops outgrow bandwidth; SpMV runs at 1–3 % of FP64 peak on every current device.

Reuse each matrix byte for K vectors: SpMM.

Bytes per euro

Memory is the expensive part of a node; footprint is a cost item.

Store the operator once for many fields; symmetries and repeated geometry shrink mesh and fields.

Code per architecture

Every hardware era cost a rewrite; the next one will too.

A minimal set of algebraic kernels keeps the method portable (F.X. Trias, 09:10).

Three pressures on sparse and stencil CFD

SURF

Bytes per flop

Peak flops outgrow bandwidth; SpMV runs at 1–3 % of FP64 peak on every current device.

Reuse each matrix byte for K vectors: SpMM.

Bytes per euro

Memory is the expensive part of a node; footprint is a cost item.

Store the operator once for many fields; symmetries and repeated geometry shrink mesh and fields.

Code per architecture

Every hardware era cost a rewrite; the next one will too.

A minimal set of algebraic kernels keeps the method portable (F.X. Trias, 09:10).

This talk: the same CFD formulation, more reuse per byte.

Four ways to raise intensity or cut traffic

Family	Examples	Saves	Costs	Keeps the algebra?
Fewer bytes per entry	mixed precision, index compression, SELL-C-σ, blocked formats	matrix bytes per non-zero (12 B $\rightarrow$ 4–8 B)	accuracy analysis, padding, format conversion	yes
No stored matrix	matrix-free and stencil kernels, Kronecker sums (CARTAN)	(almost) all operator bytes	one kernel per operator; structured grids	partly
Change the method	high-order DG, spectral elements, lattice Boltzmann	dense local work per DoF; local stencils, fewer stored operators	new discretisation, meshing and boundary treatment, re-validation, rewrite of the code and its tools	no: a new method
More vectors per matrix	multiple RHS, block Krylov, ensembles, batched solves	matrix bytes, divided by K	K vectors in memory; K problems at once	yes

Four ways to raise intensity or cut traffic

Family	Examples	Saves	Costs	Keeps the algebra?
Fewer bytes per entry	mixed precision, index compression, SELL-C-σ, blocked formats	matrix bytes per non-zero (12 B $\rightarrow$ 4–8 B)	accuracy analysis, padding, format conversion	yes
No stored matrix	matrix-free and stencil kernels, Kronecker sums (CARTAN)	(almost) all operator bytes	one kernel per operator; structured grids	partly
Change the method	high-order DG, spectral elements, lattice Boltzmann	dense local work per DoF; local stencils, fewer stored operators	new discretisation, meshing and boundary treatment, re-validation, rewrite of the code and its tools	no: a new method
More vectors per matrix	multiple RHS, block Krylov, ensembles, batched solves	matrix bytes, divided by K	K vectors in memory; K problems at once	yes

The last row is where this talk goes; the others compose with it.

One pattern: Kronecker products

Kronecker identity

For $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{W} \in \mathbb{R}^{K \times K}$ and $\mathbf{X} = [\boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_K] \in \mathbb{R}^{n \times K}$:

$$(\mathbf{W} \otimes \mathbf{A}) \operatorname{vec}(\mathbf{X}) = \operatorname{vec}\big(\mathbf{A}\,\mathbf{X}\,\mathbf{W}^T\big).$$

Kronecker identity

For $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{W} \in \mathbb{R}^{K \times K}$ and $\mathbf{X} = [\boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_K] \in \mathbb{R}^{n \times K}$:

$$(\mathbf{W} \otimes \mathbf{A}) \operatorname{vec}(\mathbf{X}) = \operatorname{vec}\big(\mathbf{A}\,\mathbf{X}\,\mathbf{W}^T\big).$$

Diagonal factor: the SpMV becomes an SpMM

With $\mathbf{W} = \operatorname{diag}(w_1, \ldots, w_K)$ and $\hat{\mathbf{A}} = \mathbf{W} \otimes \mathbf{A} \in \mathbb{R}^{Kn \times Kn}$:

$$\underbrace{\boldsymbol{y} = \hat{\mathbf{A}}\,\boldsymbol{x}}_{\text{SpMV, } \operatorname{nnz}(\hat{\mathbf{A}}) = K \operatorname{nnz}(\mathbf{A})} \quad \Longleftrightarrow \quad \underbrace{[\boldsymbol{y}_1 \ \cdots \ \boldsymbol{y}_K] = \mathbf{A}\,[w_1\boldsymbol{x}_1 \ \cdots \ w_K\boldsymbol{x}_K]}_{\text{SpMM, } \mathbf{A} \text{ read once}}$$

Sparse matrix times dense matrix

$$\mathbf{Y} = \mathbf{A}\,\mathbf{X}, \qquad \mathbf{A} \in \mathbb{R}^{n\times n} \text{ sparse, CSR}, \quad \mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n\times K} \text{ dense.}$$

FP64 values (8 B), int32 column indices and row pointers (4 B), each entry of $\mathbf{X}$ loaded once.

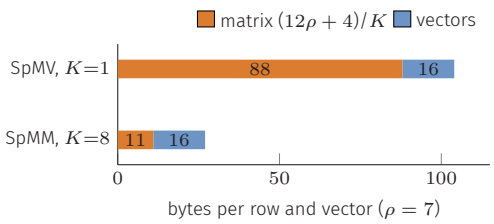
	SpMV (K calls)	SpMM (one call)
Work	$2\,\mathrm{nnz}(\mathbf{A}) \cdot K$	$2\,\mathrm{nnz}(\mathbf{A}) \cdot K$
Loads	$\big[12\,\mathrm{nnz}(\mathbf{A}) + 4(n+1) + 8n\big] \cdot K$	$12\,\mathrm{nnz}(\mathbf{A}) + 4(n+1) + 8n\,K$
Stores	$8n \cdot K$	$8n\,K$

Sparse matrix times dense matrix

$$\mathbf{Y} = \mathbf{A}\,\mathbf{X}, \qquad \mathbf{A} \in \mathbb{R}^{n\times n} \text{ sparse, CSR}, \quad \mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n\times K} \text{ dense.}$$

FP64 values (8 B), int32 column indices and row pointers (4 B), each entry of $\mathbf{X}$ loaded once.

	SpMV (K calls)	SpMM (one call)
Work	$2\,\mathrm{nnz}(\mathbf{A}) \cdot K$	$2\,\mathrm{nnz}(\mathbf{A}) \cdot K$
Loads	$\big[12\,\mathrm{nnz}(\mathbf{A}) + 4(n+1) + 8n\big] \cdot K$	$12\,\mathrm{nnz}(\mathbf{A}) + 4(n+1) + 8n\,K$
Stores	$8n \cdot K$	$8n\,K$

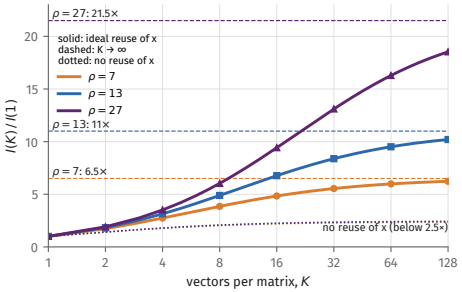


- The matrix stream is paid once: $88 \to 11$ B/row at $K = 8$.
- The vectors, 16 B/row each, are not amortised: they become the floor.
- From there, only lower precision of $\boldsymbol{x}, \boldsymbol{y}$ or fewer passes over them help.

Arithmetic intensity grows with K, up to a ceiling

Per row of $\mathbf{Y} = \mathbf{AX}$ (ρ nonzeros, $n \gg 1$):

$$I(K) \approx \frac{2\rho K}{12\rho + 4 + 16K} \quad \frac{\text{flop}}{\text{B}}$$



CSR, FP64 values, int32 indices; $I(1) = 0.135, 0.148, 0.157$ flop/B for $\rho = 7, 13, 27$.

Same model as Alsalti-Baldellou et al., JCP 486 (2023) eqs. 27–28; Comput. Fluids 275 (2024) eq. 8; Plana-Riu et al., CPC 326 (2026) eq. 2, up to $O(1/n)$ terms

Arithmetic intensity grows with K, up to a ceiling

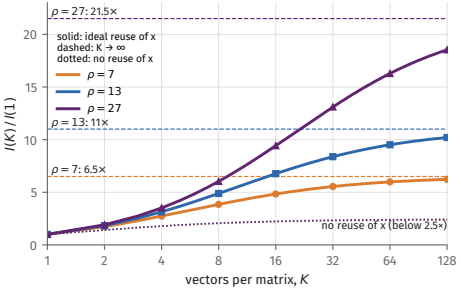
Per row of $\mathbf{Y} = \mathbf{AX}$ (ρ nonzeros, $n \gg 1$):

$$I(K) \approx \frac{2\rho\, K}{12\rho + 4 + 16K} \quad \frac{\text{flop}}{\text{B}}$$

Ceiling, ideal reuse of $\boldsymbol{x}$

$$\frac{I(K)}{I(1)} \xrightarrow{K\to\infty} \frac{12\rho + 20}{16}$$

Without reuse of $\boldsymbol{x}$ (dotted in the figure) the gain stays below 2.5.



CSR, FP64 values, int32 indices; $I(1) = 0.135, 0.148, 0.157$ flop/B for $\rho = 7, 13, 27$.

Same model as Alsalti-Baldellou et al., JCP 486 (2023) eqs. 27–28; Comput. Fluids 275 (2024) eq. 8; Plana-Riu et al., CPC 326 (2026) eq. 2, up to $O(1/n)$ terms

Arithmetic intensity grows with K, up to a ceiling

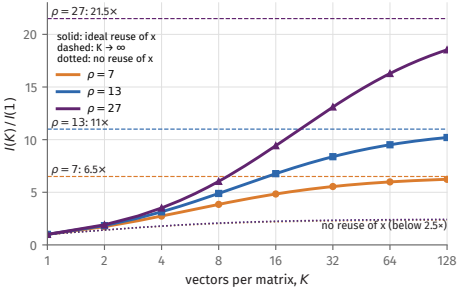
Per row of $\mathbf{Y} = \mathbf{AX}$ (ρ nonzeros, $n \gg 1$):

$$I(K) \approx \frac{2\rho\,K}{12\rho + 4 + 16K} \quad \frac{\text{flop}}{\text{B}}$$

Ceiling, ideal reuse of $\boldsymbol{x}$

$$\frac{I(K)}{I(1)} \xrightarrow{K\to\infty} \frac{12\rho + 20}{16}$$

Without reuse of $\boldsymbol{x}$ (dotted in the figure) the gain stays below 2.5.



CSR, FP64 values, int32 indices; $I(1) = 0.135, 0.148, 0.157$ flop/B for $\rho = 7, 13, 27$.

K	2	3	4	8	16	∞
$I(K)/I(1)$ for $\rho = 7$	1.73	2.29	2.74	3.85	4.84	6.5

Same model as Alsalti-Baldellou et al., JCP 486 (2023) eqs. 27–28; Comput. Fluids 275 (2024) eq. 8; Plana-Riu et al., CPC 326 (2026) eq. 2, up to $O(1/n)$ terms

Where does K come from?

Source of K	K	Kronecker form	$\mathbf{W}$
Components and fields	3, scalars	$\mathbf{I}_3 \otimes \mathbf{D}_c,\ \mathbf{I}_K \otimes \mathbf{M}$	identity

Where does K come from?

Source of K	K	Kronecker form	$\mathbf{W}$
Components and fields	3, scalars	$\mathbf{I}_3 \otimes \mathbf{D}_c,\ \mathbf{I}_K \otimes \mathbf{M}$	identity
Ensembles, sweeps (parallel in time)	states	$\mathbf{I}_K \otimes \mathbf{A},\ \mathrm{diag}(\boldsymbol{\nu}) \otimes \mathbf{L}$	identity, diagonal

Where does K come from?

Source of K	K	Kronecker form	$\mathbf{W}$
Components and fields	3, scalars	$\mathbf{I}_3 \otimes \mathbf{D}_c,\ \mathbf{I}_K \otimes \mathbf{M}$	identity
Ensembles, sweeps (parallel in time)	states	$\mathbf{I}_K \otimes \mathbf{A},\ \mathrm{diag}(\boldsymbol{\nu}) \otimes \mathbf{L}$	identity, diagonal
Spatial symmetries	2^s	$\mathbf{I}_{2^s} \otimes \mathbf{A}_{\mathrm{inn}} + \mathbf{A}_{\mathrm{out}}$	signed permutations

Where does K come from?

Source of K	K	Kronecker form	$\mathbf{W}$
Components and fields	3, scalars	$\mathbf{I}_3 \otimes \mathbf{D}_c,\ \mathbf{I}_K \otimes \mathbf{M}$	identity
Ensembles, sweeps (parallel in time)	states	$\mathbf{I}_K \otimes \mathbf{A},\ \mathrm{diag}(\boldsymbol{\nu}) \otimes \mathbf{L}$	identity, diagonal
Spatial symmetries	2^s	$\mathbf{I}_{2^s} \otimes \mathbf{A}_{\mathrm{inn}} + \mathbf{A}_{\mathrm{out}}$	signed permutations
Repeated geometry	n_b blocks	$\mathbf{I}_{n_b} \otimes \mathbf{A}_{\mathcal{I}}$ on inner unknowns, plus the interface	identity

Where does K come from?

Source of K	K	Kronecker form	$\mathbf{W}$
Components and fields	3, scalars	$\mathbf{I}_3 \otimes \mathbf{D}_c,\ \mathbf{I}_K \otimes \mathbf{M}$	identity
Ensembles, sweeps (parallel in time)	states	$\mathbf{I}_K \otimes \mathbf{A},\ \mathbf{diag}(\boldsymbol{\nu}) \otimes \mathbf{L}$	identity, diagonal
Spatial symmetries	2^s	$\mathbf{I}_{2^s} \otimes \mathbf{A}_{\mathrm{inn}} + \mathbf{A}_{\mathrm{out}}$	signed permutations
Repeated geometry	n_b blocks	$\mathbf{I}_{n_b} \otimes \mathbf{A}_{\mathcal{I}}$ on inner unknowns, plus the interface	identity

In every case a shared sparse operator meets K vectors: one SpMM kernel serves them all, and the sources can be combined (symmetry $\times$ components $\times$ runs multiplies K).

Rows 2–4: CPC 2026; JCP 2023 and C&F 2024; SISC 2024.

Benchmarks on Snellius

Measuring it: counters, not models

Benchmark

- hpc2lib, $\mathbf{I}_K \otimes \mathbf{L}$: 7-point Laplacian, stretched 3-D mesh, CSR, FP64, int32 indices.
- $K = 1 \ldots 64$ vectors; 24–110 M rows, far beyond L2/L3.
- Snellius (SURF): 2×EPYC 9654 node (192 cores, 96 ranks × 2 threads), and one H100 SXM5 94 GB.

What is measured

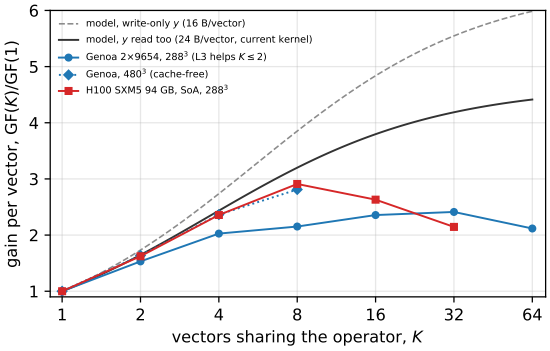
- DRAM bytes: LIKWID (Data Fabric counters, all cores) and Nsight Compute (`dram__bytes`).
- Flops: FP64 instruction counts; kernel time from hpc2lib's own timers.
- Roofs: measured streaming bandwidth on the same node — 736 GB/s (vendor 922), 2.11 TB/s (nominal 2.40).

Model the measurements are compared with

Per row: $(2\rho + 2)K$ flop, $12\rho + 4$ matrix bytes $+ 24K$ vector bytes ($\boldsymbol{x}$ read, $\boldsymbol{y}$ read and written; hpc2lib computes $\boldsymbol{y} \leftarrow a\,\mathbf{A}\boldsymbol{x} + b\,\boldsymbol{y}$ and reads $\boldsymbol{y}$ even for $b = 0$).

Runs of 3–4 Nov 2026; scripts and raw counter output in the talk repository

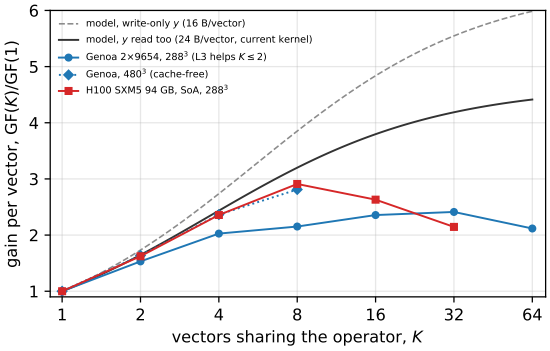
Gain per vector: the model holds up to $K \approx 4$



- Up to $K = 4$ both machines follow the traffic model: $1.6\times$ at $K = 2$, $2.4\times$ at $K = 4$.

Snellius, hpc2lib; Genoa: chunked SpMM kernel, LIKWID; H100: hpc2lib c9ae0d3, Nsight Compute. $\rho = 7$

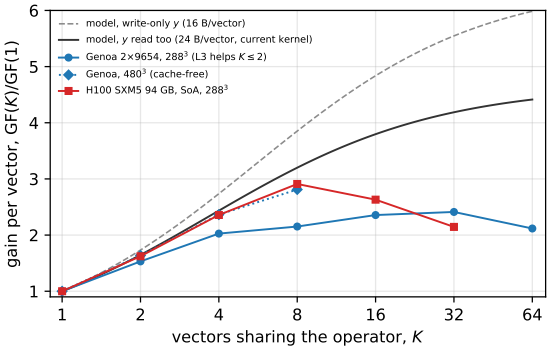
Gain per vector: the model holds up to $K \approx 4$



- Up to $K = 4$ both machines follow the traffic model: $1.6\times$ at $K = 2$, $2.4\times$ at $K = 4$.
- Then the gain saturates and turns back: best $2.9\times$ at $K = 8$ on the H100, $2.4\times$ at $K = 32$ on Genoa.

Snellius, hpc2lib; Genoa: chunked SpMM kernel, LIKWID; H100: hpc2lib c9ae0d3, Nsight Compute. $\rho = 7$

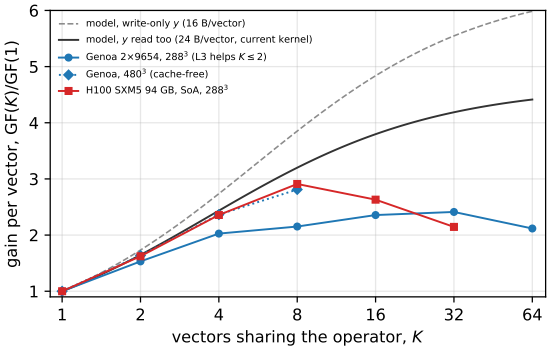
Gain per vector: the model holds up to $K \approx 4$



- Up to $K = 4$ both machines follow the traffic model: $1.6\times$ at $K = 2$, $2.4\times$ at $K = 4$.
- Then the gain saturates and turns back: best $2.9\times$ at $K = 8$ on the H100, $2.4\times$ at $K = 32$ on Genoa.
- Both stay on their bandwidth roof at every K — the loss is extra bytes, not idle bandwidth.

Snellius, hpc2lib; Genoa: chunked SpMM kernel, LIKWID; H100: hpc2lib c9ae0d3, Nsight Compute. $\rho = 7$

Gain per vector: the model holds up to $K \approx 4$

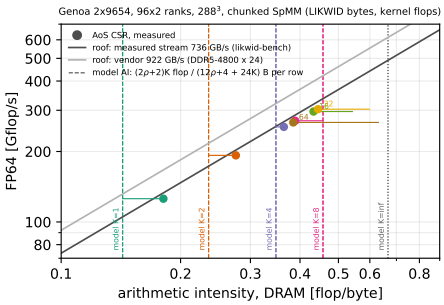


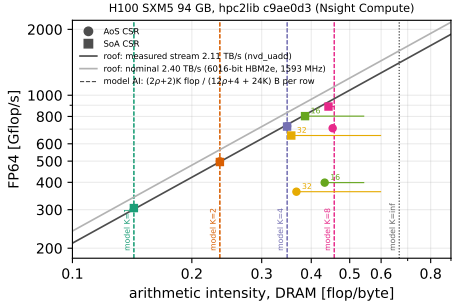
- Up to $K = 4$ both machines follow the traffic model: $1.6\times$ at $K = 2$, $2.4\times$ at $K = 4$.
- Then the gain saturates and turns back: best $2.9\times$ at $K = 8$ on the H100, $2.4\times$ at $K = 32$ on Genoa.
- Both stay on their bandwidth roof at every K — the loss is extra bytes, not idle bandwidth.

SpMM pays off, but the useful K is set by the cache: once the $\boldsymbol{x}$ values of neighbouring rows no longer fit, they are read again.

Snellius, hpc2lib; Genoa: chunked SpMM kernel, LIKWID; H100: hpc2lib c9ae0d3, Nsight Compute. $\rho = 7$

On the roof: measured bytes against estimated intensity





- Dashed: estimated intensity per K; dot: measured. Horizontal gap = bytes the model does not predict.
- H100: estimate and measurement coincide up to $K = 8$ (flops and bytes within 3 %); 93–101 % of the measured roof.
- Genoa: L3 reuse *lowers* traffic at $K \leq 2$; from $K = 8$ traffic exceeds the model; 93–95 % of the roof throughout.

Roofs: measured streaming bandwidth (likwid-bench on Genoa, hpc2lib streaming kernel on the H100)

Parallel in time: ensembles on one mesh

Run K simulations of the same case together

- Ensembles: K decorrelated realisations average K times faster per time unit.
- Parameter sweeps: K angles of attack, inflow or material values.

Plana-Riu et al., CPC 326 (2026) 110232, Secs. 2–3, Fig. 3

K flow states on one mesh

Run K simulations of the same case together

- Ensembles: K decorrelated realisations average K times faster per time unit.
- Parameter sweeps: K angles of attack, inflow or material values.

Every operator becomes $\mathbf{I}_K \otimes \mathbf{A}$

$$\mathbf{I}_K \otimes \mathbf{M}, \quad \mathbf{I}_K \otimes \mathbf{G}, \quad \mathbf{I}_K \otimes \mathbf{D}, \quad \mathbf{I}_K \otimes \Gamma$$

Convection differs per state only through a diagonal:

$$\hat{\mathbf{C}}(\boldsymbol{u}_s)\,\boldsymbol{u}_c = \underbrace{\hat{\mathbf{M}}}_{\text{SpMM}} \underbrace{\hat{\mathbf{U}}_s}_{\text{axty}} \underbrace{\hat{\Gamma}\,\boldsymbol{u}_c}_{\text{SpMM}}$$

Plana-Riu et al., CPC 326 (2026) 110232, Secs. 2–3, Fig. 3

K flow states on one mesh

Run K simulations of the same case together

- Ensembles: K decorrelated realisations average K times faster per time unit.
- Parameter sweeps: K angles of attack, inflow or material values.

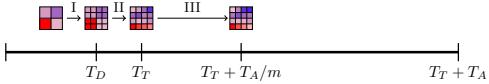
Every operator becomes $\mathbf{I}_K \otimes \mathbf{A}$

$$\mathbf{I}_K \otimes \mathbf{M}, \quad \mathbf{I}_K \otimes \mathbf{G}, \quad \mathbf{I}_K \otimes \mathbf{D}, \quad \mathbf{I}_K \otimes \Gamma$$

Convection differs per state only through a diagonal:

$$\hat{\mathbf{C}}(\boldsymbol{u}_s)\,\boldsymbol{u}_c = \underbrace{\hat{\mathbf{M}}}_{\text{SpMM}}\,\underbrace{\hat{\mathbf{U}}_s}_{\text{axty}}\,\underbrace{\hat{\Gamma}\,\boldsymbol{u}_c}_{\text{SpMM}}$$

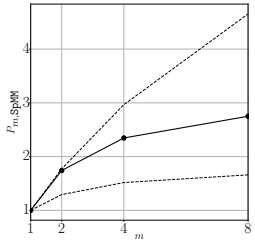
Ensembles: start on a coarse mesh



I: one run on a coarse mesh; II: map it to K perturbed states and develop them; III: average each over T_A/K.

Plana-Riu et al., CPC 326 (2026) 110232, Secs. 2–3, Fig. 3

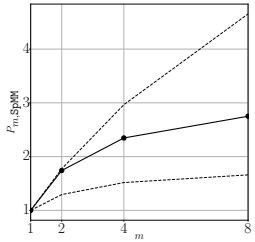
Channel $Re_\tau = 180$: kernel



SpMM between its bounds,
$\approx 2.7\times$ at $K = 8$; up to 3.0, 3.3,
$4.1\times$ for 7, 13, 27 points.

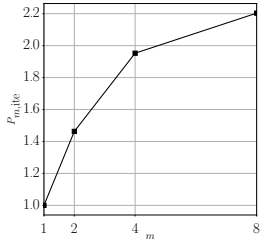
Plana-Riu et al., CPC 326 (2026) 110232, Figs. 8, 11, 14, 15; MareNostrum 5 GPP; $m = K$ in the plots

Channel $Re_\tau = 180$: kernel



SpMM between its bounds, $\approx 2.7\times$ at $K = 8$; up to 3.0, 3.3, $4.1\times$ for 7, 13, 27 points.

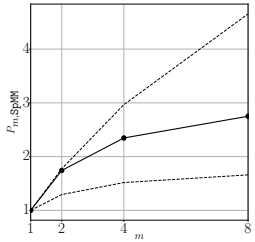
Channel: whole time step



Time step $\approx 2.2\times$ at $K = 8$. Whole simulation $1.54\times$; without the coarse start the transient costs it all ($0.94\times$).

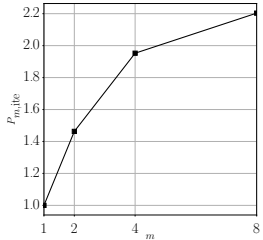
Plana-Riu et al., CPC 326 (2026) 110232, Figs. 8, 11, 14, 15; MareNostrum 5 GPP; $m = K$ in the plots

Channel $Re_\tau = 180$: kernel



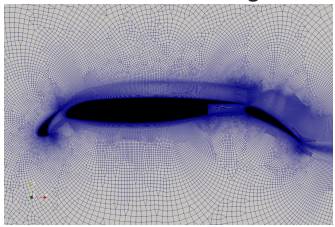
SpMM between its bounds, $\approx 2.7\times$ at $K = 8$; up to 3.0, 3.3, $4.1\times$ for 7, 13, 27 points.

Channel: whole time step



Time step $\approx 2.2\times$ at $K = 8$. Whole simulation $1.54\times$; without the coarse start the transient costs it all ($0.94\times$).

30P30N airfoil: K angles

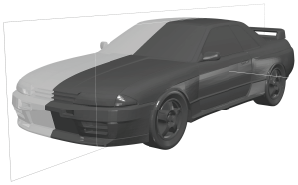


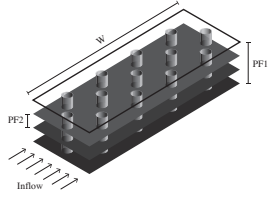
Unstructured, 14M cells. No transient to repeat: iteration and simulation speed-up up to $1.8\times$.

Plana-Riu et al., CPC 326 (2026) 110232, Figs. 8, 11, 14, 15; MareNostrum 5 GPP; $m = K$ in the plots

Spatial symmetries

2^s mirror images of one subdomain





One plane (car), two planes (heat exchanger)

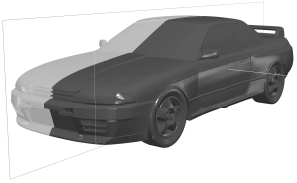
Mirror the ordering, split the operator

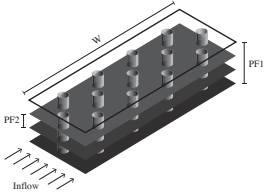
Number the 2^s subdomains one after another, each with the base subdomain's ordering mirrored:

$$\mathbf{A} = \underbrace{\mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}}}_{\text{SpMM, } K=2^s} + \underbrace{\mathbf{A}_{\text{out}}}_{\text{thin coupling}}$$

Alsalti-Baldellou et al., JCP 486 (2023) 112133; Comput. Fluids 275 (2024) 106247

2^s mirror images of one subdomain





One plane (car), two planes (heat exchanger)

Mirror the ordering, split the operator

Number the 2^s subdomains one after another, each with the base subdomain's ordering mirrored:

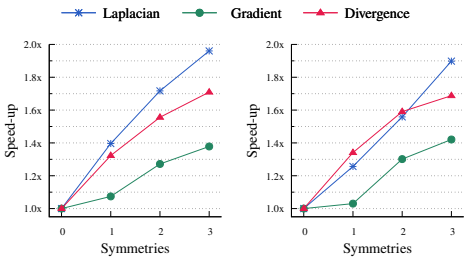
$$\mathbf{A} = \underbrace{\mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}}}_{\text{SpMM, } K=2^s} + \underbrace{\mathbf{A}_{\text{out}}}_{\text{thin coupling}}$$

- Only the base mesh and $\mathbf{A}_{\text{inn}}$ are stored: footprint $/\,2^s$.
- Boundary conditions need not be symmetric: they go into $\mathbf{A}_{\text{out}}$.
- For Poisson, a change of basis decouples 2^s smaller systems.

Alsalti-Baldellou et al., JCP 486 (2023) 112133; Comput. Fluids 275 (2024) 106247

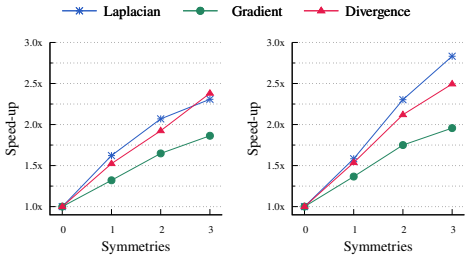
Results: symmetry-aware SpMV

CPU: 2× Xeon 8160



Laplacian up to 2.0×

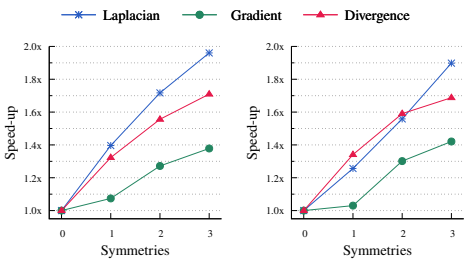
GPU: RTX A5000



Laplacian up to 3.3×

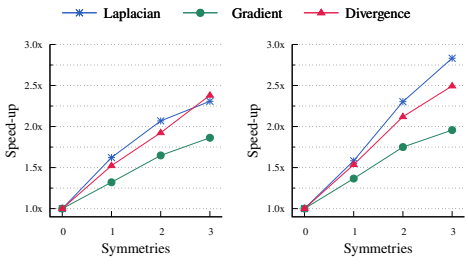
Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Figs. 8–9; JCP 486 (2023), Sec. 4.1

CPU: 2× Xeon 8160



Laplacian up to 2.0×

GPU: RTX A5000



Laplacian up to 3.3×

- Cube (15.5M cells) and finned tube (17.7M), base mesh fixed: no cache bonus. $\mathbf{L}$, $\mathbf{M}$, $\mathbf{G}$ rank by intensity.
- Matrix footprint up to $8\times$ smaller at $s = 3$.
- Poisson up to $4.3\times$ faster, from fewer iterations after the change of basis.

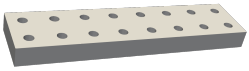
Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Figs. 8–9; JCP 486 (2023), Sec. 4.1

Repeated geometry

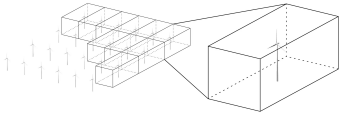
Repeated blocks: mesh one, solve all



DrivAer: reflection, $n_b = 2$



Heat exchanger: $n_b \leq 8$



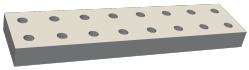
Wind farm: translation, $n_b \leq 24$

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

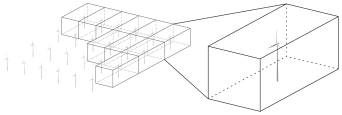
Repeated blocks: mesh one, solve all



DrivAer: reflection, $n_b = 2$



Heat exchanger: $n_b \leq 8$



Wind farm: translation, $n_b \leq 24$

Inner unknowns first, same order per block

$$\mathbf{A} = \begin{pmatrix} \mathbf{I}_{n_b} \otimes \mathbf{A}_{\mathcal{I}} & \mathbf{I}_{n_b} \otimes \mathbf{B} \\ \mathbf{I}_{n_b} \otimes \mathbf{B}^T & \mathbf{A}_{\Gamma\Gamma} \end{pmatrix}$$

- One block is stored; the interface $\mathbf{A}_{\Gamma\Gamma}$ can be anything.
- AMGR: a multigrid reduction whose smoother on the inner unknowns is $\mathbf{I}_{n_b} \otimes \mathbf{G}$, one SpMM.
- Reflections give $n_b \leq 8$; translations are unbounded.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

SURF

PCG + AMGR vs. PCG + AMG (Poisson solve)

case	unknowns	n_b	iterations	speed-up
DrivAer	106.4 M	2	$26 \to 26$	$1.43\times$
heat exchanger	18.4 M	2	$20 \to 19$	$1.38\times$
		4	$20 \to 19$	$1.50\times$
		8	$20 \to 18$	$1.68\times$
wind farm	52 M	2	$67 \to 49$	$1.65\times$
		4	$67 \to 50$	$1.64\times$
		24	$67 \to 47$	$1.51\times$

JFF cluster (CPU), Chronos.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

PCG + AMGR vs. PCG + AMG (Poisson solve)

case	unknowns	n_b	iterations	speed-up
DrivAer	106.4 M	2	$26 \to 26$	$1.43\times$
heat exchanger	18.4 M	2	$20 \to 19$	$1.38\times$
		4	$20 \to 19$	$1.50\times$
		8	$20 \to 18$	$1.68\times$
wind farm	52 M	2	$67 \to 49$	$1.65\times$
		4	$67 \to 50$	$1.64\times$
		24	$67 \to 47$	$1.51\times$

JFF cluster (CPU), Chronos.

Where the gain erodes

- More blocks $\Rightarrow$ more interface $\Rightarrow$ larger coarse operator: the wind farm peaks at $n_b = 2$.
- CFD Poisson matrices want a very light smoother (FSAI with ≈ 4 nnz/row): little for SpMM to accelerate, memory similar to AMG.
- Denser operators (high order, elasticity) should benefit more; GPU is future work.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

Conclusions

Conclusions

1. The bandwidth wall is here to stay. Flops grow faster than bytes; SpMV in CSR sits at ≈ 0.13 flop/B on any current device, CPU or GPU.

Conclusions

1. The bandwidth wall is here to stay. Flops grow faster than bytes; SpMV in CSR sits at ≈ 0.13 flop/B on any current device, CPU or GPU.
2. K comes from structure the CFD already has. Velocity components, transported scalars, parameters, ensembles, reflection symmetries, repeated blocks: whenever one sparse operator meets K vectors, $\mathbf{I}_K \otimes \mathbf{A}$ turns K SpMVs into one SpMM.

Conclusions

1. The bandwidth wall is here to stay. Flops grow faster than bytes; SpMV in CSR sits at ≈ 0.13 flop/B on any current device, CPU or GPU.
2. K comes from structure the CFD already has. Velocity components, transported scalars, parameters, ensembles, reflection symmetries, repeated blocks: whenever one sparse operator meets K vectors, $\mathbf{I}_K \otimes \mathbf{A}$ turns K SpMVs into one SpMM.
3. Same formulation, more reuse. No change to the discretisation or its accuracy; the matrix is read once per K vectors. It composes with lower-precision storage and matrix-free operators.

Conclusions

1. The bandwidth wall is here to stay. Flops grow faster than bytes; SpMV in CSR sits at ≈ 0.13 flop/B on any current device, CPU or GPU.
2. K comes from structure the CFD already has. Velocity components, transported scalars, parameters, ensembles, reflection symmetries, repeated blocks: whenever one sparse operator meets K vectors, $\mathbf{I}_K \otimes \mathbf{A}$ turns K SpMVs into one SpMM.
3. Same formulation, more reuse. No change to the discretisation or its accuracy; the matrix is read once per K vectors. It composes with lower-precision storage and matrix-free operators.

Measured so far: kernels 2–$4\times$, whole simulations or Poisson solves 1.3–$1.8\times$, memory footprint up to $8\times$ smaller with symmetries.

Thank you

Questions?

Acknowledgements

O.P., M.M-O., J.P-R., F.X.T., G.C. and A.O. are supported by the SIMEX project (PID2022-142174OB-I00) of the Ministerio de Ciencia e Innovación and the European Union NextGenerationEU, MCIN/AEI/10.13039/501100011033.

Backup

Backup: SpMV and SpMM traffic in full

Counts for $\mathbf{Y} = \mathbf{A}\mathbf{X}$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{X} \in \mathbb{R}^{n \times K}$
CSR, FP64 values, int32 indices, one FMA per non-zero:

$$W(K) = 2\,\mathrm{nnz}(\mathbf{A})\,K$$

$$Q(K) = \underbrace{8\,\mathrm{nnz}(\mathbf{A})}_{\text{values}} + \underbrace{4\,\mathrm{nnz}(\mathbf{A})}_{\text{columns}} + \underbrace{4(m+1)}_{\text{row ptr}} + \underbrace{(8n + 8m)K}_{\boldsymbol{x}\text{ read, }\boldsymbol{y}\text{ written}}$$

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133, eqs. 27–28; Comput. Fluids 275 (2024) 106247, eq. 8; J. Plana-Riu et al., Comput. Phys. Commun. 326 (2026) 110232, eqs. 2–3

Backup: SpMV and SpMM traffic in full

Counts for $\mathbf{Y} = \mathbf{A}\mathbf{X}$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{X} \in \mathbb{R}^{n \times K}$
CSR, FP64 values, int32 indices, one FMA per non-zero:

$$W(K) = 2\,\mathrm{nnz}(\mathbf{A})\,K$$

$$Q(K) = \underbrace{8\,\mathrm{nnz}(\mathbf{A})}_{\text{values}} + \underbrace{4\,\mathrm{nnz}(\mathbf{A})}_{\text{columns}} + \underbrace{4(m+1)}_{\text{row ptr}} + \underbrace{(8n+8m)K}_{\boldsymbol{x}\text{ read, }\boldsymbol{y}\text{ written}}$$

Square, $\rho = \mathrm{nnz}/n$ per row, $n \gg 1$:

$$I(K) \approx \frac{2\rho K}{12\rho + 4 + 16K}, \qquad \frac{I(K)}{I(1)} \xrightarrow{K\to\infty} \frac{12\rho+20}{16}$$

hpc2lib also reads $\boldsymbol{y}$ ($b\,\boldsymbol{y}$ even for $b = 0$): $16K \to 24K$; flops $(2\rho+2)K$, as measured.

No reuse of $\boldsymbol{x}$ ($8nK \to 8\,\mathrm{nnz}\,K$):

$$I_{\mathrm{lb}}(K) \approx \frac{2\rho K}{12\rho + 4 + 8(\rho+1)K}, \qquad \frac{I_{\mathrm{lb}}(K)}{I_{\mathrm{lb}}(1)} \to \frac{20\rho+12}{8\rho+8}$$

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133, eqs. 27–28; Comput. Fluids 275 (2024) 106247, eq. 8; J. Plana-Riu et al., Comput. Phys. Commun. 326 (2026) 110232, eqs. 2–3

Backup: SpMV and SpMM traffic in full

Counts for $\mathbf{Y} = \mathbf{A}\mathbf{X}$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{X} \in \mathbb{R}^{n\times K}$
CSR, FP64 values, int32 indices, one FMA per non-zero:

$$W(K) = 2\,\mathrm{nnz}(\mathbf{A})\,K$$

$$Q(K) = \underbrace{8\,\mathrm{nnz}(\mathbf{A})}_{\text{values}} + \underbrace{4\,\mathrm{nnz}(\mathbf{A})}_{\text{columns}} + \underbrace{4(m+1)}_{\text{row ptr}} + \underbrace{(8n+8m)K}_{\boldsymbol{x}\text{ read, }\boldsymbol{y}\text{ written}}$$

Square, $\rho = \mathrm{nnz}/n$ per row, $n \gg 1$:

$$I(K) \approx \frac{2\rho K}{12\rho + 4 + 16K}, \qquad \frac{I(K)}{I(1)} \xrightarrow{K\to\infty} \frac{12\rho+20}{16}$$

hpc2lib also reads $\boldsymbol{y}$ ($b\,\boldsymbol{y}$ even for $b = 0$): $16K \to 24K$; flops $(2\rho+2)K$, as measured.

No reuse of $\boldsymbol{x}$ ($8nK \to 8\,\mathrm{nnz}\,K$):

$$I_{\mathrm{lb}}(K) \approx \frac{2\rho K}{12\rho + 4 + 8(\rho+1)K}, \qquad \frac{I_{\mathrm{lb}}(K)}{I_{\mathrm{lb}}(1)} \to \frac{20\rho+12}{8\rho+8}$$

Gain $I(K)/I(1)$

ρ	$I(1)$	$K=2$	4	8	16	∞
7	0.135	1.73	2.74	3.85	4.84	6.5
13	0.148	1.83	3.14	4.89	6.77	11
27	0.157	1.91	3.51	6.04	9.42	21.5

Lower bound at $K = 8$: ≈ 2.0–$2.1\times$ for all ρ.

Differences between our papers

- JCP and CPC count "+1" flop instead of $+m$ (C&F corrects it); the limit is unchanged.
- JCP and CPC add $+8$ bytes per row for the accumulator, which lives in a register.
- With $\boldsymbol{y} \mathrel{+}= \mathbf{A}\boldsymbol{x}$, $\boldsymbol{y}$ is read too: $16mK$ instead of $8mK$.

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133, eqs. 27–28; Comput. Fluids 275 (2024) 106247, eq. 8; J. Plana-Riu et al., Comput. Phys. Commun. 326 (2026) 110232, eqs. 2–3

Backup: machine-balance data

device	year	FP64 [TF/s]	FP64 matrix [TF/s]	BW [TB/s]	balance [flop/B]	matrix [flop/B]	status
NVIDIA V100 SXM2	2017	7.8	–	0.90	8.7	–	verify
NVIDIA A100 SXM 80 GB	2020	9.7	19.5	2.04	4.8	9.6	verify
AMD MI250X	2021	47.9	95.7	3.28	14.6	29.2	verify
NVIDIA H100 SXM	2022	34.0	67.0	3.35	10.1	20.0	verify
AMD EPYC 9654 (1 socket)	2022	3.69	–	0.461	8.0	–	verify
AMD MI300A	2023	61.3	122.6	5.3	11.6	23.1	checked
NVIDIA B200	2024	37.0	37.0	7.7	4.8	4.8	verify
AMD MI355X	2025	78.6	78.6	8.0	9.8	9.8	checked
NVIDIA B300	2025	1.2	–	8.0	0.15	–	verify

Sources

- Vendor datasheets (V100, A100, MI250X, H100).
- AMD MI300A data sheet and MI355X brochure (checked 2026-11-03).
- B200, B300: secondary (glennklockwood.com); B300 FP64 cut to $\approx 1{:}64$ of FP32.
- EPYC 9654: $96 \times 2.4\,\text{GHz} \times 16$ flop/cycle; $12\times$DDR5-4800.

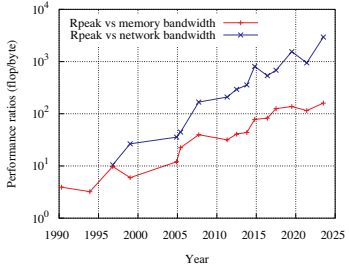
Rows marked “verify” come from memory or secondary sources. [TODO: check them against the vendor datasheets before the talk]

SpMV in CSR: ≈ 0.15 flop/B. Only the FP64-reduced B300 is balanced for it.

data/machine-balance.csv

Flops outgrow bytes

Three decades of balance (1990–2023)

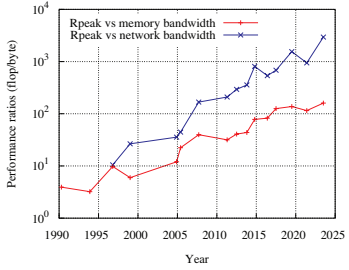


Peak / memory and peak / injection bandwidth, flop per 64-bit word ($\div 8$ for flop/B).

π/β: $+14.2\%$/yr; π/β_{inj}: $+22.3\%$/yr [TODO: these are 3.8×/7.5× per decade; Roofplane paper says 4.5×/9× (McCalpin SC16): keep one, cite it]

Flops outgrow bytes

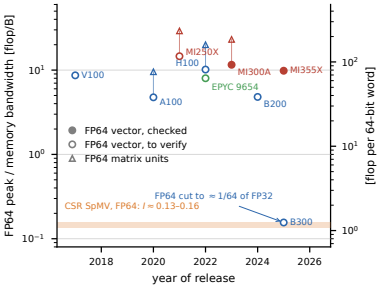
Three decades of balance (1990–2023)



Peak / memory and peak / injection bandwidth, flop per 64-bit word ($\div 8$ for flop/B).

π/β: $+14.2\%$/yr; $\pi/\beta_{\mathfrak{A}}$: $+22.3\%$/yr [TODO: these are 3.8× /7.5× per decade; Roofplane paper says 4.5× /9× (McCalpin SC16): keep one, cite it]

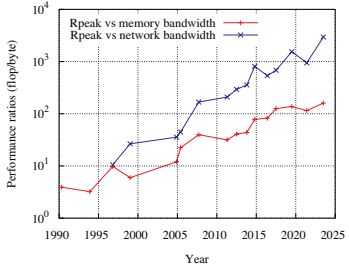
Accelerators since 2017



FP64 peak / memory bandwidth per device. Vendor peaks; to verify (hollow markers).

Flops outgrow bytes

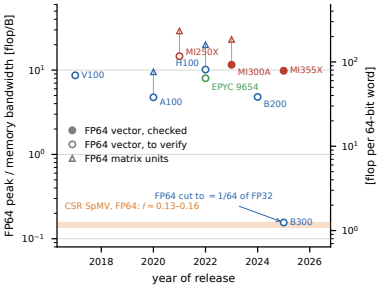
Three decades of balance (1990–2023)



Peak / memory and peak / injection bandwidth, flop per 64-bit word ($\div 8$ for flop/B).

π/β: $+14.2\%$/yr; π/β_{inj}: $+22.3\%$/yr [TODO: these are 3.8× /7.5× per decade; Roofplane paper says 4.5× /9× (McCalpin SC16): keep one, cite it]

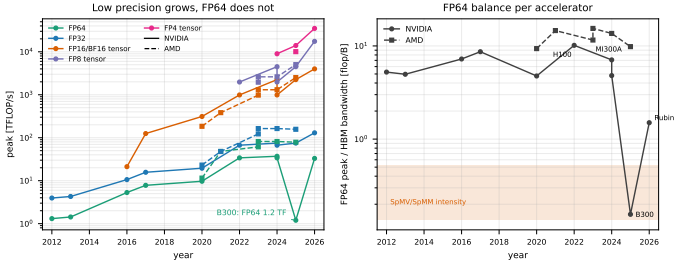
Accelerators since 2017



FP64 peak / memory bandwidth per device. Vendor peaks; to verify (hollow markers).

HPC parts: FP64 balance stays at ≈ 5–15 flop/B (MI300A ≈ 12, MI355X ≈ 10). AI parts cut FP64 (B300 ≈ 0.15 flop/B) and put the growth in low precision.
Either way the bandwidth wall stays: CSR SpMV needs only ≈ 0.15 flop/B and is limited by β alone.
[TODO: storyline says FP64 balance "keeps growing"; the 2017–2025 CSV shows a plateau (NVIDIA 5–10, AMD 15⟶10)]

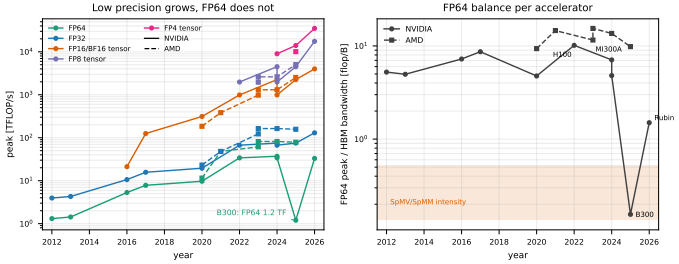
The AI era: low precision explodes, FP64 is negotiable



- Tensor-core FP16 is 30–120× FP64. B300 cuts FP64 to 1.2 TF/s; Rubin returns to 33, below B200's 37.

Vendor datasheets; tensor-core peaks without 2:4 structured sparsity. NVIDIA developer blog on FP64 emulation in cuBLAS, CUDA 13.0 U2 (Oct 2025); The Register, 18 Jan 2026.

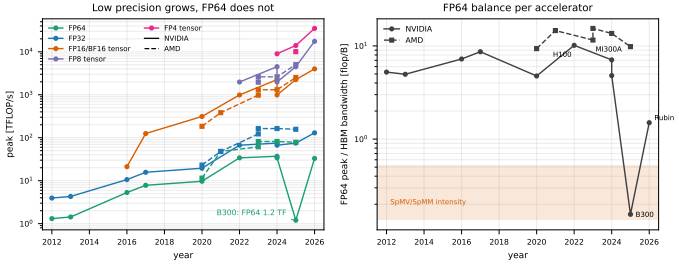
The AI era: low precision explodes, FP64 is negotiable



- Tensor-core FP16 is 30–120× FP64. B300 cuts FP64 to 1.2 TF/s; Rubin returns to 33, below B200's 37.
- B300: 0.16 flop/B, inside the SpMV band. SpMV just fits; SpMM with $K \geq 2$ is compute-bound.

Vendor datasheets; tensor-core peaks without 2:4 structured sparsity. NVIDIA developer blog on FP64 emulation in cuBLAS, CUDA 13.0 U2 (Oct 2025); The Register, 18 Jan 2026.

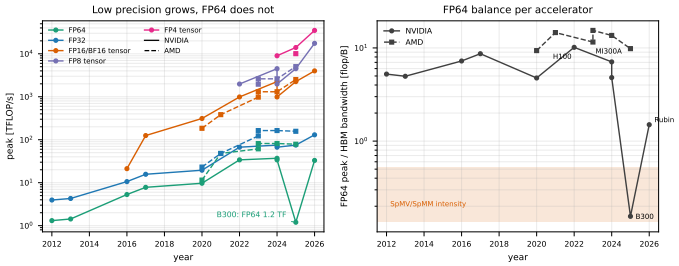
The AI era: low precision explodes, FP64 is negotiable



- Tensor-core FP16 is 30–120× FP64. B300 cuts FP64 to 1.2 TF/s; Rubin returns to 33, below B200's 37.
- B300: 0.16 flop/B, inside the SpMV band. SpMV just fits; SpMM with $K \geq 2$ is compute-bound.
- FP64 *matrix products* can be emulated on integer tensor cores (Ozaki scheme, in cuBLAS); Rubin's 200 TF/s DGEMM is emulated.

Vendor datasheets; tensor-core peaks without 2:4 structured sparsity. NVIDIA developer blog on FP64 emulation in cuBLAS, CUDA 13.0 U2 (Oct 2025); The Register, 18 Jan 2026.

The AI era: low precision explodes, FP64 is negotiable



- Tensor-core FP16 is 30–120× FP64. B300 cuts FP64 to 1.2 TF/s; Rubin returns to 33, below B200's 37.
- B300: 0.16 flop/B, inside the SpMV band. SpMV just fits; SpMM with $K \geq 2$ is compute-bound.
- FP64 *matrix products* can be emulated on integer tensor cores (Ozaki scheme, in cuBLAS); Rubin's 200 TF/s DGEMM is emulated.
- Sparse and stencil kernels have no matrix products to emulate; their limit is memory bandwidth, which no arithmetic trick raises.

Vendor datasheets; tensor-core peaks without 2:4 structured sparsity. NVIDIA developer blog on FP64 emulation in cuBLAS, CUDA 13.0 U2 (Oct 2025); The Register, 18 Jan 2026.

Memory: not scarcer per flop, but much more expensive

Capacity per core and per flop

node	year	GB/core	GB per FP64 TF/s
SuperMUC (Sandy Bridge)	2012	2.0	≈ 93
SuperMUC-NG (Skylake)	2018	2.0	≈ 23
Snellius Rome (7H12)	2021	2.0	≈ 48
Snellius Genoa (9654)	2023	2.0	≈ 52
H100 SXM (HBM)	2022	–	2.4
Rubin (HBM4)	2026	–	8.7

- CPU nodes keep ≈ 2 GB per core; per flop it fell 2–4× since 2012.
- GPUs stay at 2–9 GB per FP64 TF/s, but per FP16 TF/s it fell 3× since V100.

Node and accelerator datasheets; TrendForce press releases 18 Dec 2025, 2 Feb 2026; Counterpoint Memory Price Tracker, 5 Feb 2026.

Memory: not scarcer per flop, but much more expensive

Capacity per core and per flop

node	year	GB/core	GB per FP64 TF/s
SuperMUC (Sandy Bridge)	2012	2.0	≈ 93
SuperMUC-NG (Skylake)	2018	2.0	≈ 23
Snellius Rome (7H12)	2021	2.0	≈ 48
Snellius Genoa (9654)	2023	2.0	≈ 52
H100 SXM (HBM)	2022	–	2.4
Rubin (HBM4)	2026	–	8.7

- CPU nodes keep ≈ 2 GB per core; per flop it fell 2–4× since 2012.
- GPUs stay at 2–9 GB per FP64 TF/s, but per FP16 TF/s it fell 3× since V100.

Price of a byte

- Server DRAM contract prices ≈ +90 % in one quarter (1Q26), the largest rise on record.
- 64 GB RDIMM: \$450 (4Q25) → > \$900 (1Q26).
- HBM3e at 4–5× the price of DDR5 (Dec 2025).
- Cause: AI demand moved DRAM wafer capacity to HBM and server DDR5.

Node and accelerator datasheets; TrendForce press releases 18 Dec 2025, 2 Feb 2026; Counterpoint Memory Price Tracker, 5 Feb 2026.

Memory: not scarcer per flop, but much more expensive

Capacity per core and per flop

node	year	GB/core	GB per FP64 TF/s
SuperMUC (Sandy Bridge)	2012	2.0	≈ 93
SuperMUC-NG (Skylake)	2018	2.0	≈ 23
Snellius Rome (7H12)	2021	2.0	≈ 48
Snellius Genoa (9654)	2023	2.0	≈ 52
H100 SXM (HBM)	2022	–	2.4
Rubin (HBM4)	2026	–	8.7

- CPU nodes keep ≈ 2 GB per core; per flop it fell 2–4× since 2012.
- GPUs stay at 2–9 GB per FP64 TF/s, but per FP16 TF/s it fell 3× since V100.

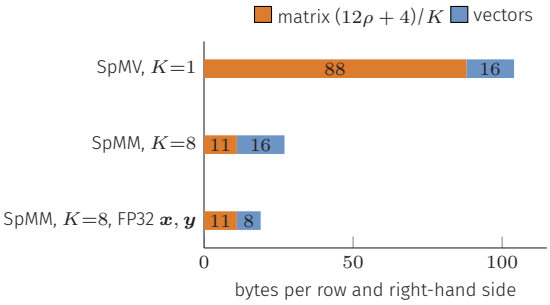
Price of a byte

- Server DRAM contract prices ≈ +90 % in one quarter (1Q26), the largest rise on record.
- 64 GB RDIMM: \$450 (4Q25) → > \$900 (1Q26).
- HBM3e at 4–5× the price of DDR5 (Dec 2025).
- Cause: AI demand moved DRAM wafer capacity to HBM and server DDR5.

The memory footprint of a simulation is now a cost item: fewer bytes per DoF is worth as much as fewer bytes per flop.

Node and accelerator datasheets; TrendForce press releases 18 Dec 2025, 2 Feb 2026; Counterpoint Memory Price Tracker, 5 Feb 2026.

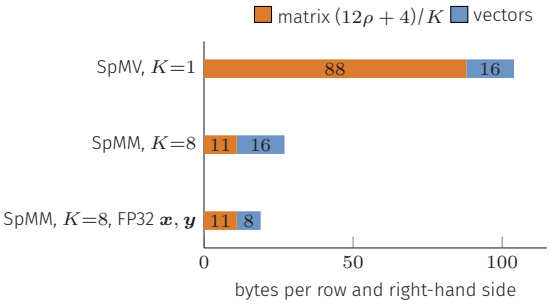
The levers compose



Model for a 7-point stencil, CSR with FP64 values and int32 indices.

- SpMM amortises the matrix bytes: $88 \to 11$ B/row at $K = 8$.
- Then the vectors are the floor: 16 B/row per right-hand side.

The levers compose



Model for a 7-point stencil, CSR with FP64 values and int32 indices.

- SpMM amortises the matrix bytes: $88 \to 11$ B/row at $K = 8$.
- Then the vectors are the floor: 16 B/row per right-hand side.
- The floor is where precision (vector storage) and fusion (fewer vector passes) come back.

Where the extra bytes come from, and what to do

DRAM bytes per row, measured / model (CSR, $\rho = 7$)

K	model	Genoa	H100
1	112	88	112
4	184	176	184
8	280	329	289
16	472	592	662
32	856	1153	1434
64	1624	2662	–

Genoa 288^3, 96×2 ranks; H100 288^3, SoA. $K = 64$ does not launch on the GPU (shared memory); $K = 128$ exceeds hpc2lib's 64 components.

1. $\boldsymbol{x}$ reuse across mesh planes: the $\pm z$ neighbours of a row lie one plane away; once three planes of K-wide $\boldsymbol{x}$ exceed the cache, they are read again.

Interpretation of the counter data from three rank-block shapes on Genoa (cubic, slab, two ranks); not yet confirmed with per-region counters

Where the extra bytes come from, and what to do

DRAM bytes per row, measured / model (CSR, $\rho = 7$)

K	model	Genoa	H100
1	112	88	112
4	184	176	184
8	280	329	289
16	472	592	662
32	856	1153	1434
64	1624	2662	–

Genoa 288^3, 96×2 ranks; H100 288^3, SoA. $K = 64$ does not launch on the GPU (shared memory); $K = 128$ exceeds hpc2lib's 64 components.

1. $\boldsymbol{x}$ reuse across mesh planes: the $\pm z$ neighbours of a row lie one plane away; once three planes of K-wide $\boldsymbol{x}$ exceed the cache, they are read again.
2. Halo update inside the product grows with the surface per rank and with K.

Interpretation of the counter data from three rank-block shapes on Genoa (cubic, slab, two ranks); not yet confirmed with per-region counters

Where the extra bytes come from, and what to do

DRAM bytes per row, measured / model (CSR, $\rho = 7$)

K	model	Genoa	H100
1	112	88	112
4	184	176	184
8	280	329	289
16	472	592	662
32	856	1153	1434
64	1624	2662	–

Genoa 288^3, 96×2 ranks; H100 288^3, SoA. $K = 64$ does not launch on the GPU (shared memory); $K = 128$ exceeds hpc2lib's 64 components.

1. $\boldsymbol{x}$ reuse across mesh planes: the $\pm z$ neighbours of a row lie one plane away; once three planes of K-wide $\boldsymbol{x}$ exceed the cache, they are read again.
2. Halo update inside the product grows with the surface per rank and with K.
3. $\boldsymbol{y}$ is read although $b = 0$: 8 of the 24 vector bytes are avoidable.

Interpretation of the counter data from three rank-block shapes on Genoa (cubic, slab, two ranks); not yet confirmed with per-region counters

Where the extra bytes come from, and what to do

DRAM bytes per row, measured / model (CSR, $\rho = 7$)

K	model	Genoa	H100
1	112	88	112
4	184	176	184
8	280	329	289
16	472	592	662
32	856	1153	1434
64	1624	2662	–

Genoa 288^3, 96×2 ranks; H100 288^3, SoA. $K = 64$ does not launch on the GPU (shared memory); $K = 128$ exceeds hpc2lib's 64 components.

1. $\boldsymbol{x}$ reuse across mesh planes: the $\pm z$ neighbours of a row lie one plane away; once three planes of K-wide $\boldsymbol{x}$ exceed the cache, they are read again.
2. Halo update inside the product grows with the surface per rank and with K.
3. $\boldsymbol{y}$ is read although $b = 0$: 8 of the 24 vector bytes are avoidable.

Next steps in the kernel

Write-only $\boldsymbol{y}$ for $b = 0$; tiled or blocked row order so $\boldsymbol{x}$ planes fit in cache; K in register chunks (done for the CPU, $2.4\times$ at $K = 32$).

Interpretation of the counter data from three rank-block shapes on Genoa (cubic, slab, two ranks); not yet confirmed with per-region counters

Summary: where K comes from and what it buys

source of K	K	AI gain bound*	measured kernel speed-up	measured end-to-end gain	hardware	reference
components, fields, ensembles, parameters	3; 2–16	$\leq 6.5\times$ ($\rho = 7$, $K \to \infty$)	SpMM up to $3.0\times$ (7-pt), $3.3\times$ (13-pt), $4.1\times$ (27-pt)	channel $Re_\tau = 180$: $1.35/1.54/1.54\times$ ($m = 2/4/8$, coarse-mesh start); airfoil 30P30N multi-angle $\approx 1.8\times$	MareNostrum 5 GPP (CPU)	CPC 326 (2026)
spatial symmetries	$2^s \leq 8$	$3.85\times$ ($K = 8$, $\rho = 7$)	$\mathbf{L}$: $2.0\times$ CPU, $3.3\times$ GPU (fair, no cache bonus); footprint up to $8\times$ smaller	Poisson: up to $4.3\times$ (Krylov, fewer iterations); AMG up to $1.3\times$	MN4 Xeon 8160; RTX A5000; JFF	C&F 275 (2024); JCP 486 (2023)
repeated geometry	$n_b \leq 24$	$\leq 4.25\times$ (smoother $\rho \approx 4$)	not reported separately	Poisson (AMGR vs. AMG): 1.4–$1.7\times$	JFF (CPU)	SISC 46(6) (2024)
same operator, K vectors (this work)	1–64	$3.2\times$ ($K = 8$, $\boldsymbol{y}$ read)	per vector: $2.9\times$ ($K = 8$) H100; $2.4\times$ ($K = 32$) Genoa; on the bandwidth roof	– (kernel only)	Snellius H100, 2 × EPYC 9654	next slides

* Model per row, CSR, FP64 values, int32 indices, ideal reuse of $\boldsymbol{\alpha}$: $I(K)/I(1)$ with $I(K) \approx 2\rho K/(12\rho + 4 + 16K)$, limit $(12\rho + 20)/16$.

Summary: where K comes from and what it buys

source of K	K	AI gain bound*	measured kernel speed-up	measured end-to-end gain	hardware	reference
components, fields, ensembles, parameters	3; 2–16	$\leq 6.5\times$ ($\rho = 7$, $K \to \infty$)	SpMM up to $3.0\times$ (7-pt), $3.3\times$ (13-pt), $4.1\times$ (27-pt)	channel $Re_\tau = 180$: $1.35/1.54/1.54\times$ ($m = 2/4/8$, coarse-mesh start); airfoil 30P30N multi-angle $\approx 1.8\times$	MareNostrum 5 GPP (CPU)	CPC 326 (2026)
spatial symmetries	$2^s \leq 8$	$3.85\times$ ($K = 8$, $\rho = 7$)	$\mathbf{L}$: $2.0\times$ CPU, $3.3\times$ GPU (fair, no cache bonus); footprint up to $8\times$ smaller	Poisson: up to $4.3\times$ (Krylov, fewer iterations); AMG up to $1.3\times$	MN4 Xeon 8160; RTX A5000; JFF	C&F 275 (2024); JCP 486 (2023)
repeated geometry	$n_b \leq 24$	$\leq 4.25\times$ (smoother $\rho \approx 4$)	not reported separately	Poisson (AMGR vs. AMG): 1.4–$1.7\times$	JFF (CPU)	SISC 46(6) (2024)
same operator, K vectors (this work)	1–64	$3.2\times$ ($K = 8$, $\boldsymbol{y}$ read)	per vector: $2.9\times$ ($K = 8$) H100; $2.4\times$ ($K = 32$) Genoa; on the bandwidth roof	– (kernel only)	Snellius H100, $2\times$ EPYC 9654	next slides

* Model per row, CSR, FP64 values, int32 indices, ideal reuse of $\boldsymbol{x}$: $I(K)/I(1)$ with $I(K) \approx 2\rho K/(12\rho + 4 + 16K)$, limit $(12\rho + 20)/16$.

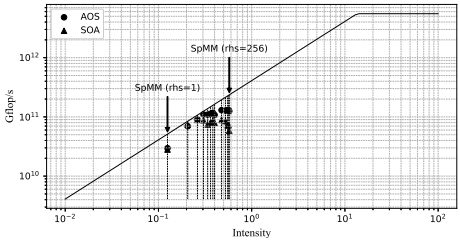
Reading the table

- Kernels reach roughly 45–85% of their model bound; end to end, 1.3–$1.8\times$ is typical.

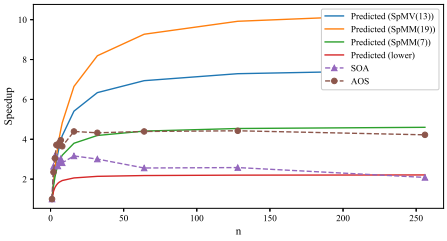
Not comparable as-is

Machines, operators and baselines differ; the JCP Poisson gain comes from fewer iterations, not SpMM. [TODO: check CPC and SISC values against the published versions]

Measured: SpMM on a GPU node



Roofline, SpMM with 1 to 256 right-hand sides, AoS and SoA layouts



Speed-up against K SpMVs, with the model bounds

- The interleaved (AoS) kernel follows the ideal-reuse prediction and saturates; the strided (SoA) kernel falls back towards the no-reuse bound.
- The layout decides which of the two bounds the kernel sees.

[TODO: state matrix (size, stencil), kernel and GPU of these Snellius gcn runs; source data and scripts of both plots not found]

Velocity components share one operator

Collocated, symmetry-preserving discretisation, $\boldsymbol{u}_c = (\boldsymbol{u}_1; \boldsymbol{u}_2; \boldsymbol{u}_3) \in \mathbb{R}^{3n}$

$$\Omega = \mathbf{I}_3 \otimes \Omega_c, \qquad \mathbf{D} = \mathbf{I}_3 \otimes \mathbf{D}_c, \qquad \Gamma = \mathbf{I}_3 \otimes \Gamma_{c \to s}$$

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Sec. 2; Plana-Riu et al., CPC 326 (2026), eq. 10

Velocity components share one operator

Collocated, symmetry-preserving discretisation, $\boldsymbol{u}_c = (\boldsymbol{u}_1; \boldsymbol{u}_2; \boldsymbol{u}_3) \in \mathbb{R}^{3n}$

$$\Omega = \mathbf{I}_3 \otimes \Omega_c, \qquad \mathbf{D} = \mathbf{I}_3 \otimes \mathbf{D}_c, \qquad \Gamma = \mathbf{I}_3 \otimes \Gamma_{c\to s}$$

Diffusion of the three components: $K = 3$

$$\mathbf{D}\,\boldsymbol{u}_c = -\left(\nu\,\mathbf{I}_3 \otimes \mathbf{L}\right)\boldsymbol{u}_c = -\begin{pmatrix} \nu\mathbf{L} & & \\ & \nu\mathbf{L} & \\ & & \nu\mathbf{L} \end{pmatrix}\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \end{pmatrix} \quad \Longleftrightarrow \quad [\,\cdot\,] = -\nu\,\mathbf{L}\,[\boldsymbol{u}_1\ \boldsymbol{u}_2\ \boldsymbol{u}_3]$$

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Sec. 2; Plana-Riu et al., CPC 326 (2026), eq. 10

Velocity components share one operator

Collocated, symmetry-preserving discretisation, $\boldsymbol{u}_c = (\boldsymbol{u}_1; \boldsymbol{u}_2; \boldsymbol{u}_3) \in \mathbb{R}^{3n}$

$$\Omega = \mathbf{I}_3 \otimes \Omega_c, \qquad \mathbf{D} = \mathbf{I}_3 \otimes \mathbf{D}_c, \qquad \Gamma = \mathbf{I}_3 \otimes \Gamma_{c\to s}$$

Diffusion of the three components: $K = 3$

$$\mathbf{D}\,\boldsymbol{u}_c = -\left(\nu\,\mathbf{I}_3 \otimes \mathbf{L}\right)\boldsymbol{u}_c = -\begin{pmatrix} \nu\mathbf{L} & & \\ & \nu\mathbf{L} & \\ & & \nu\mathbf{L} \end{pmatrix}\begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \end{pmatrix} \quad\Longleftrightarrow\quad [\,\cdot\,] = -\nu\,\mathbf{L}\,[\boldsymbol{u}_1\ \boldsymbol{u}_2\ \boldsymbol{u}_3]$$

- In our code all products reduce to three sparsity patterns: cells→cells ($\mathbf{L}$), cells→faces ($\mathbf{G}$, $\Gamma_{c\to s}$), faces→cells ($\mathbf{M}$).
- The block-diagonal structure is free: no reordering, no change of discretisation.

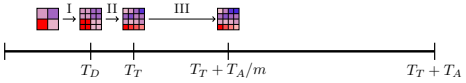
Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Sec. 2; Plana-Riu et al., CPC 326 (2026), eq. 10

Caveat: ensembles pay the transient K times

Channel, without the coarse-mesh start

K	2	4	8
coarse start	$1.35\times$	$1.54\times$	$1.54\times$
no coarse start (model)	$1.23\times$	$1.21\times$	$0.94\times$

- Most of the end-to-end gain at large K comes from the coarse start, not from SpMM alone.
- Needs decorrelated realisations and T_A/K above one time unit; T_D trades accuracy against speed.



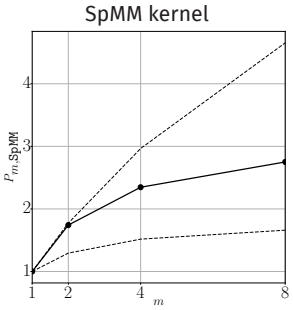
I: mapping (coarse), II: developing, III: averaging over m states. CPC 2026, Fig. 3.

[TODO: verify against published CPC 326 (2026) 110232]

Parameter sweeps get the kernel gain without this penalty; ensembles trade it against the transient.

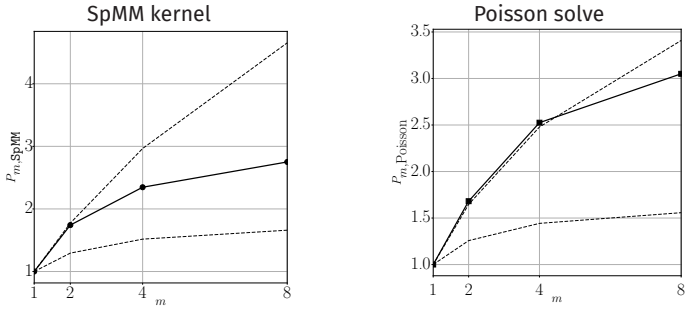
Plana-Riu et al., CPC 326 (2026), Sec. 4.1

From kernel to solver to time step (channel flow, 7 points)



J. Plana-Riu et al., CPC 326 (2026) 110232; $m = K$; figures from J. Plana-Riu's thesis.

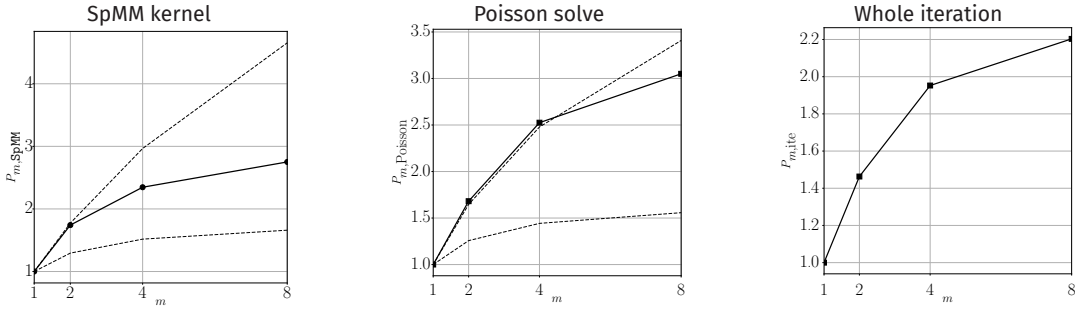
From kernel to solver to time step (channel flow, 7 points)



- The Poisson solve (CG with AMG) follows its own bounds: SpMM dominates its cost.

J. Plana-Riu et al., CPC 326 (2026) 110232; $m = K$; figures from J. Plana-Riu's thesis.

From kernel to solver to time step (channel flow, 7 points)

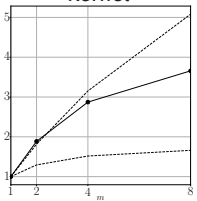


- The Poisson solve (CG with AMG) follows its own bounds: SpMM dominates its cost.
- The time step keeps $\approx 2.2\times$ at $K = 8$: the rest of the step (axpy, dot, non-linear terms) dilutes the kernel gain.

J. Plana-Riu et al., CPC 326 (2026) 110232; $m = K$; figures from J. Plana-Riu's thesis.

Results on MareNostrum 5 (CPU)

Kernel



27 points; 7 and 13 on the previous frames

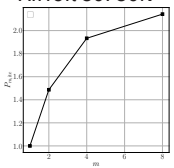
- SpMM up to $\approx$ $3.0\times$, $3.3\times$, $4.1\times$ for 7, 13, 27 points.
- Ideal-reuse bound at $K = 16$: 4.8, 6.8, $9.4\times$: denser stencils fall further below it.

Channel $Re_\tau = 180$

K	2	4	8
simulation	$1.35\times$	$1.54\times$	$1.54\times$

- 160^3, coarse start on 80^3, 64 cores; $\beta = 10$, $\gamma = 15$.
- Ensemble statistics agree for $K = 1, 2, 4, 8$.
- Model within ≈8–10% of the measurement.

Airfoil 30P30N



Whole iteration, $m = K$. CPC 2026, Fig. 15.

- Several angles of attack; unstructured, 14M cells, 112 cores.
- Iteration speed-up $\approx$ $1.8\times$, which is the simulation speed-up: no transient penalty for a parameter sweep.

[TODO: verify against published CPC 326 (2026) 110232; numbers taken from the August 2025 submission]

Plana-Riu et al., CPC 326 (2026) 110232, Figs. 8, 15, Tables 5–6

Fields, parameters and ensembles

K runs on one mesh

- several transported scalars (temperature, species);
- parameter sweeps (angles of attack, inflow conditions);
- ensembles of decorrelated realisations.

$$\mathbf{I}_K \otimes \mathbf{M}, \qquad \mathbf{I}_K \otimes \mathbf{G}_c, \qquad \mathbf{I}_K \otimes \Gamma_{c \to s}$$

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), eq. (kron); Imamura et al., Int. J. CFD 30 (2016); Plana-Riu et al., CPC 326 (2026)

Fields, parameters and ensembles

K runs on one mesh

- several transported scalars (temperature, species);
- parameter sweeps (angles of attack, inflow conditions);
- ensembles of decorrelated realisations.

$$\mathbf{I}_K \otimes \mathbf{M}, \qquad \mathbf{I}_K \otimes \mathbf{G}_c, \qquad \mathbf{I}_K \otimes \Gamma_{c\to s}$$

Different coefficients: diagonal $\mathbf{W}$

$$\mathrm{diag}(\boldsymbol{\nu}) \otimes \mathbf{L}, \qquad \boldsymbol{\nu} = (\nu_1, \ldots, \nu_K)$$

Combined with the components:

$$\mathrm{diag}(\boldsymbol{\nu}) \otimes \mathbf{I}_3 \otimes \mathbf{L} \quad \Rightarrow \quad 3K \text{ vectors}$$

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), eq. (kron); Imamura et al., Int. J. CFD 30 (2016); Plana-Riu et al., CPC 326 (2026)

Fields, parameters and ensembles

K runs on one mesh

- several transported scalars (temperature, species);
- parameter sweeps (angles of attack, inflow conditions);
- ensembles of decorrelated realisations.

$$\mathbf{I}_K \otimes \mathbf{M}, \qquad \mathbf{I}_K \otimes \mathbf{G}_c, \qquad \mathbf{I}_K \otimes \Gamma_{c\to s}$$

Different coefficients: diagonal $\mathbf{W}$

$$\operatorname{diag}(\boldsymbol{\nu}) \otimes \mathbf{L}, \qquad \boldsymbol{\nu} = (\nu_1, \ldots, \nu_K)$$

Combined with the components:

$$\operatorname{diag}(\boldsymbol{\nu}) \otimes \mathbf{I}_3 \otimes \mathbf{L} \quad \Rightarrow \quad 3K \text{ vectors}$$

$$\begin{pmatrix} \mathbf{D}_1 & \\ & \mathbf{D}_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_{c,1} \\ \boldsymbol{u}_{c,2} \end{pmatrix} = - \begin{pmatrix} \nu_1\mathbf{L} & & & & & \\ & \nu_1\mathbf{L} & & & & \\ & & \nu_1\mathbf{L} & & & \\ & & & \nu_2\mathbf{L} & & \\ & & & & \nu_2\mathbf{L} & \\ & & & & & \nu_2\mathbf{L} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_{c,1} \\ \boldsymbol{u}_{c,2} \end{pmatrix} = -\left(\operatorname{diag}(\nu_1, \nu_2) \otimes \mathbf{I}_3 \otimes \mathbf{L}\right) \boldsymbol{u}_c$$

Operators are stored once; the fields grow with K. Unlike symmetries, the mesh is not reduced.

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), eq. (kron); Imamura et al., Int. J. CFD 30 (2016); Plana-Riu et al., CPC 326 (2026)

Navier–Stokes for K realisations

Semi-discrete system, all realisations stacked

$$\hat{\mathbf{M}}\,\boldsymbol{u}_s = \mathbf{0}, \qquad \hat{\Omega}\,\frac{d\boldsymbol{u}_c}{dt} + \hat{\mathbf{M}}\,\hat{\mathbf{U}}_s\,\hat{\Gamma}\,\boldsymbol{u}_c + \hat{\mathbf{D}}\,\boldsymbol{u}_c + \hat{\Omega}\,\hat{\mathbf{G}}_c\,\boldsymbol{p}_c = \mathbf{0}$$

$$\hat{\mathbf{M}} = \mathbf{I}_K \otimes \mathbf{M}, \quad \hat{\Omega} = \mathbf{I}_K \otimes \Omega, \quad \hat{\mathbf{D}} = \mathbf{I}_K \otimes \mathbf{D}, \quad \hat{\mathbf{G}}_c = \mathbf{I}_K \otimes \mathbf{G}_c, \quad \hat{\Gamma} = \mathbf{I}_K \otimes \Gamma_{c\to s}, \quad \hat{\mathbf{U}}_s = \mathrm{diag}(\boldsymbol{u}_{s,1}, \ldots, \boldsymbol{u}_{s,K})$$

Plana-Riu et al., CPC 326 (2026), eqs. 10a–b, in this talk's notation

Navier–Stokes for K realisations

Semi-discrete system, all realisations stacked

$$\hat{\mathbf{M}}\,\boldsymbol{u}_s = \mathbf{0}, \qquad \hat{\Omega}\,\frac{d\boldsymbol{u}_c}{dt} + \hat{\mathbf{M}}\,\hat{\mathbf{U}}_s\,\hat{\Gamma}\,\boldsymbol{u}_c + \hat{\mathbf{D}}\,\boldsymbol{u}_c + \hat{\Omega}\,\hat{\mathbf{G}}_c\,\boldsymbol{p}_c = \mathbf{0}$$

$$\hat{\mathbf{M}} = \mathbf{I}_K \otimes \mathbf{M}, \quad \hat{\Omega} = \mathbf{I}_K \otimes \Omega, \quad \hat{\mathbf{D}} = \mathbf{I}_K \otimes \mathbf{D}, \quad \hat{\mathbf{G}}_c = \mathbf{I}_K \otimes \mathbf{G}_c, \quad \hat{\Gamma} = \mathbf{I}_K \otimes \Gamma_{c\to s}, \quad \hat{\mathbf{U}}_s = \operatorname{diag}(\boldsymbol{u}_{s,1}, \ldots, \boldsymbol{u}_{s,K})$$

Convection is not a Kronecker product

$\hat{\mathbf{C}}(\boldsymbol{u}_s) = \hat{\mathbf{M}}\,\hat{\mathbf{U}}_s\,\hat{\Gamma}$ differs per realisation, but it factors:

$$\hat{\mathbf{C}}(\boldsymbol{u}_s)\,\boldsymbol{u}_c = \underbrace{\hat{\mathbf{M}}}_{\text{SpMM}}\Big(\underbrace{\hat{\mathbf{U}}_s}_{\text{axty}}\big(\underbrace{\hat{\Gamma}\,\boldsymbol{u}_c}_{\text{SpMM}}\big)\Big)$$

Only the diagonal depends on the realisation: an element-wise product.

Plana-Riu et al., CPC 326 (2026), eqs. 10a–b, in this talk's notation

Navier–Stokes for K realisations

Semi-discrete system, all realisations stacked

$$\hat{\mathbf{M}}\,\boldsymbol{u}_s = \mathbf{0}, \qquad \hat{\Omega}\,\frac{d\boldsymbol{u}_c}{dt} + \hat{\mathbf{M}}\,\hat{\mathbf{U}}_s\,\hat{\Gamma}\,\boldsymbol{u}_c + \hat{\mathbf{D}}\,\boldsymbol{u}_c + \hat{\Omega}\,\hat{\mathbf{G}}_c\,\boldsymbol{p}_c = \mathbf{0}$$

$$\hat{\mathbf{M}} = \mathbf{I}_K \otimes \mathbf{M}, \quad \hat{\Omega} = \mathbf{I}_K \otimes \Omega, \quad \hat{\mathbf{D}} = \mathbf{I}_K \otimes \mathbf{D}, \quad \hat{\mathbf{G}}_c = \mathbf{I}_K \otimes \mathbf{G}_c, \quad \hat{\Gamma} = \mathbf{I}_K \otimes \Gamma_{c\to s}, \quad \hat{\mathbf{U}}_s = \mathrm{diag}(\boldsymbol{u}_{s,1}, \ldots, \boldsymbol{u}_{s,K})$$

Convection is not a Kronecker product

$\hat{\mathbf{C}}(\boldsymbol{u}_s) = \hat{\mathbf{M}}\,\hat{\mathbf{U}}_s\,\hat{\Gamma}$ differs per realisation, but it factors:

$$\hat{\mathbf{C}}(\boldsymbol{u}_s)\,\boldsymbol{u}_c = \underbrace{\hat{\mathbf{M}}}_{\text{SpMM}}\Big(\underbrace{\hat{\mathbf{U}}_s}_{\text{axty}}\big(\underbrace{\hat{\Gamma}\,\boldsymbol{u}_c}_{\text{SpMM}}\big)\Big)$$

Only the diagonal depends on the realisation: an element-wise product.

- Every sparse product of the step is an SpMM; Poisson sees K right-hand sides. Heun RK3 with the adaptive eigenvalue-bound Δt (AlgEigCD) [TODO: how one Δt is chosen across the K runs].

Plana-Riu et al., CPC 326 (2026), eqs. 10a–b, in this talk's notation

From kernel speed-up to simulation speed-up

Level	Model	Meaning
Kernel	$P_{K,\mathrm{SpMM}} \in \left[P_K^{lb},\ P_K^{ub}\right]$	traffic bounds of the previous section

Plana-Riu et al., CPC 326 (2026), eqs. 3, 7, 12, 21, Tables 2–3; eq. 7 rewritten in Amdahl form

From kernel speed-up to simulation speed-up

Level	Model	Meaning
Kernel	$P_{K,\mathrm{SpMM}} \in [P_K^{lb},\ P_K^{ub}]$	traffic bounds of the previous section
Poisson	$P_{K,\mathrm{Poisson}} = \left(\chi / P_{K,\mathrm{SpMM}} + 1 - \chi\right)^{-1}$	Amdahl; χ: SpMM share of the solve

Plana-Riu et al., CPC 326 (2026), eqs. 3, 7, 12, 21, Tables 2–3; eq. 7 rewritten in Amdahl form

From kernel speed-up to simulation speed-up

Level	Model	Meaning
Kernel	$P_{K,\mathrm{SpMM}} \in [P_K^{lb},\ P_K^{ub}]$	traffic bounds of the previous section
Poisson	$P_{K,\mathrm{Poisson}} = \left(\chi / P_{K,\mathrm{SpMM}} + 1 - \chi\right)^{-1}$	Amdahl; χ: SpMM share of the solve
Iteration	$P_{K,\mathrm{ite}} = \left(\Theta / P_{K,\mathrm{SpMM}} + 1 - \Theta\right)^{-1}$	Θ: share of sparse products in a step

Plana-Riu et al., CPC 326 (2026), eqs. 3, 7, 12, 21, Tables 2–3; eq. 7 rewritten in Amdahl form

From kernel speed-up to simulation speed-up

Level	Model	Meaning
Kernel	$P_{K,\mathrm{SpMM}} \in [P_K^{lb},\ P_K^{ub}]$	traffic bounds of the previous section
Poisson	$P_{K,\mathrm{Poisson}} = \left(\chi/P_{K,\mathrm{SpMM}} + 1 - \chi\right)^{-1}$	Amdahl; χ: SpMM share of the solve
Iteration	$P_{K,\mathrm{ite}} = \left(\Theta/P_{K,\mathrm{SpMM}} + 1 - \Theta\right)^{-1}$	Θ: share of sparse products in a step
Simulation	$P_K = \dfrac{1+\tilde{\beta}}{\Phi K + \tilde{\beta}}\ P_{K,\mathrm{ite}}$	every realisation repeats the transient

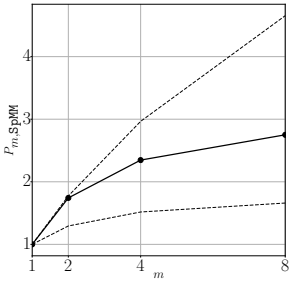
- $\beta = T_A/T_T$: averaging over transient time; each run needs $T_T + T_A/K$.
- Coarse-mesh start: develop on a coarse mesh up to T_D, then refine; $\tilde{\beta} = \dfrac{\Pi(1+\gamma)}{\gamma+\Pi}\,\beta$, $\Pi \approx r_f^4$.
- [TODO: define Φ, γ and T_{DT} as in CPC eq. 21]

A priori ($\Theta = 0.65$, $\Pi = 16$, $\gamma = 5$)

$\rho = 7$: $1.14\times$ ($K = 2$, $\beta = 1$) to $1.78\times$ ($K = 16$, $\beta = 40$).
$\rho = 27$: up to $2.14\times$.
Large K with small β: slow-down (e.g. $0.52\times$).

Plana-Riu et al., CPC 326 (2026), eqs. 3, 7, 12, 21, Tables 2–3; eq. 7 rewritten in Amdahl form

Channel flow $Re_\tau = 180$: SpMM speed-up against its bounds

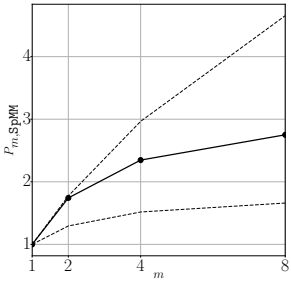


7 nonzeros per row

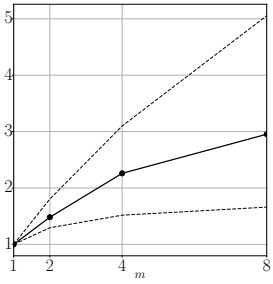
- 160^3 cells, 64 cores of one MareNostrum 5-GPP node, MPI+OpenMP; $m = K$ flow states advanced together (ensemble averaging), $K = 1, 2, 4, 8$.

J. Plana-Riu et al., CPC 326 (2026) 110232; figures from J. Plana-Riu's thesis.

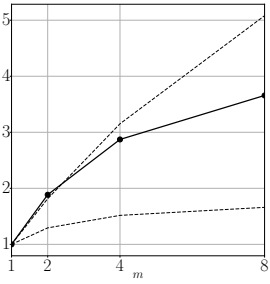
Channel flow $Re_\tau = 180$: SpMM speed-up against its bounds



7 nonzeros per row



13 nonzeros per row

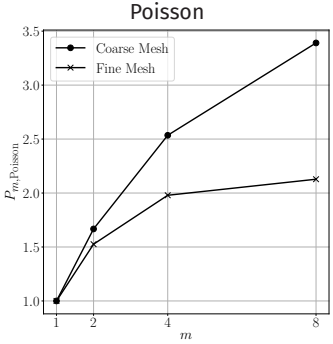


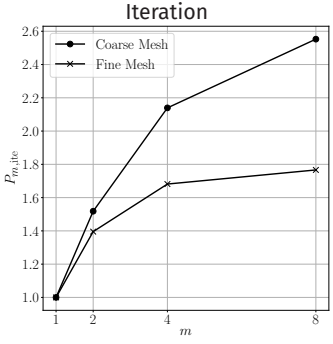
27 nonzeros per row

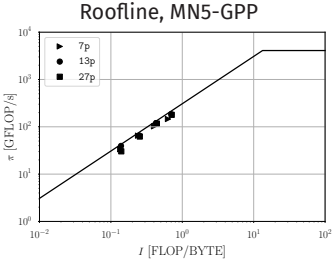
- 160^3 cells, 64 cores of one MareNostrum 5-GPP node, MPI+OpenMP; $m = K$ flow states advanced together (ensemble averaging), $K = 1, 2, 4, 8$.
- Measured SpMM speed-up (solid) between the upper and lower traffic bounds (dashed); the denser the stencil, the larger the gain and the larger the distance to the upper bound.

J. Plana-Riu et al., CPC 326 (2026) 110232; figures from J. Plana-Riu's thesis.

Backup: channel flow, coarse vs fine mesh and roofline







- The coarse mesh gains more than the fine one; the fine mesh is the production case. [TODO: confirm mesh sizes and what "coarse" means with Josep]
- All three stencils stay on the bandwidth roof as K grows.

J. Plana-Riu et al., CPC 326 (2026) 110232; figures from J. Plana-Riu's thesis.

“Parallel in time”: two different things

(a) Ensemble averaging

What our CPC paper calls “parallel in time”.

- K decorrelated realisations advance together:

$$\hat{\mathbf{A}} = \mathbf{I}_K \otimes \mathbf{A}, \qquad \mathbf{C}(\boldsymbol{u}) = \mathbf{M}\,\mathbf{U}_s\,\Gamma$$

  convection: SpMM ($\mathbf{M}$, Γ) + axty ($\mathbf{U}_s$).

- Each realisation averages only T_A/K, but repeats the transient T_T. With $\beta = T_A/T_T$ (no coarse-mesh start): $P_K = \frac{1+\beta}{K+\beta}\,P_{K,\text{step}}$.

(a) Krasnopolsky, CPC 229 (2018) 8–19; Makarashvili et al., CPC 219 (2017) 236–245; Plana-Riu et al., CPC 326 (2026) 110232. (b) Gander, Vandewalle, SISC 29(2) (2007) 556–578; Falgout et al., SISC 36(6) (2014) C635–C661; McDonald, Pestana, Wathen, SISC 40(2) (2018) A1012–A1033

“Parallel in time”: two different things

(a) Ensemble averaging

What our CPC paper calls “parallel in time”.

- K decorrelated realisations advance together:

$$\hat{\mathbf{A}} = \mathbf{I}_K \otimes \mathbf{A}, \qquad \mathbf{C}(\boldsymbol{u}) = \mathbf{M}\,\mathbf{U}_s\,\Gamma$$

  convection: SpMM ($\mathbf{M}$, Γ) + axty ($\mathbf{U}_s$).

- Each realisation averages only T_A/K, but repeats the transient T_T. With $\beta = T_A/T_T$ (no coarse-mesh start): $P_K = \frac{1+\beta}{K+\beta}\,P_{K,\text{step}}$.

(b) Time-parallel methods

Parareal, MGRIT, ParaDiag. Implicit Euler, T steps, all at once (Ω: cell volumes):

$$\left(\mathbf{I}_T \otimes (\Omega + \Delta t\,\mathbf{A}) \;-\; \mathbf{S}_T \otimes \Omega\right) \boldsymbol{u} = \boldsymbol{b}$$

Its product: one SpMM of $\mathbf{A}$ over the T time levels.

(a) Krasnopolsky, CPC 229 (2018) 8–19; Makarashvili et al., CPC 219 (2017) 236–245; Plana-Riu et al., CPC 326 (2026) 110232. (b) Gander, Vandewalle, SISC 29(2) (2007) 556–578; Falgout et al., SISC 36(6) (2014) C635–C661; McDonald, Pestana, Wathen, SISC 40(2) (2018) A1012–A1033

“Parallel in time”: two different things

(a) Ensemble averaging

What our CPC paper calls “parallel in time”.

- K decorrelated realisations advance together:

$$\hat{\mathbf{A}} = \mathbf{I}_K \otimes \mathbf{A}, \qquad \mathbf{C}(\boldsymbol{u}) = \mathbf{M}\,\mathbf{U}_s\,\Gamma$$

  convection: SpMM ($\mathbf{M}$, Γ) + axty ($\mathbf{U}_s$).

- Each realisation averages only T_A/K, but repeats the transient T_T. With $\beta = T_A/T_T$ (no coarse-mesh start): $P_K = \frac{1+\beta}{K+\beta}\,P_{K,\text{step}}$.

(b) Time-parallel methods

Parareal, MGRIT, ParaDiag. Implicit Euler, T steps, all at once (Ω: cell volumes):

$$\left(\mathbf{I}_T \otimes (\Omega + \Delta t\,\mathbf{A}) \;-\; \mathbf{S}_T \otimes \Omega\right) \boldsymbol{u} = \boldsymbol{b}$$

Its product: one SpMM of $\mathbf{A}$ over the T time levels.
ParaDiag: $\mathcal{P}_\alpha$ = same with $\mathbf{S}_T \to \alpha$-circulant $\mathbf{V}\mathbf{D}\mathbf{V}^{-1}$:

$$(\mathbf{V}^{-1} \otimes \mathbf{I})\,\mathcal{P}_\alpha\,(\mathbf{V} \otimes \mathbf{I}) = (\mathbf{I}_T - \mathbf{D}) \otimes \Omega + \Delta t\,\mathbf{I}_T \otimes \mathbf{A}$$

T shifted systems on $\mathbf{A}$'s pattern: $\sum_j \mathbf{W}_j \otimes \mathbf{A}_j$, complex diagonal $\mathbf{W}_j$.

(a) Krasnopolsky, CPC 229 (2018) 8–19; Makarashvili et al., CPC 219 (2017) 236–245; Plana-Riu et al., CPC 326 (2026) 110232. (b) Gander, Vandewalle, SISC 29(2) (2007) 556–578; Falgout et al., SISC 36(6) (2014) C635–C661; McDonald, Pestana, Wathen, SISC 40(2) (2018) A1012–A1033

“Parallel in time”: two different things

(a) Ensemble averaging

What our CPC paper calls “parallel in time”.

- K decorrelated realisations advance together:

$$\hat{\mathbf{A}} = \mathbf{I}_K \otimes \mathbf{A}, \qquad \mathbf{C}(\boldsymbol{u}) = \mathbf{M}\,\mathbf{U}_s\,\Gamma$$

  convection: SpMM ($\mathbf{M}$, Γ) + axty ($\mathbf{U}_s$).

- Each realisation averages only T_A/K, but repeats the transient T_T. With $\beta = T_A/T_T$ (no coarse-mesh start): $P_K = \frac{1+\beta}{K+\beta}\,P_{K,\text{step}}$.

(b) Time-parallel methods

Parareal, MGRIT, ParaDiag. Implicit Euler, T steps, all at once (Ω: cell volumes):

$$\left(\mathbf{I}_T \otimes (\Omega + \Delta t\,\mathbf{A}) \;-\; \mathbf{S}_T \otimes \Omega\right) \boldsymbol{u} = \boldsymbol{b}$$

Its product: one SpMM of $\mathbf{A}$ over the T time levels.
ParaDiag: $\mathcal{P}_\alpha$ = same with $\mathbf{S}_T \to \alpha$-circulant $\mathbf{V}\mathbf{D}\mathbf{V}^{-1}$:

$$(\mathbf{V}^{-1} \otimes \mathbf{I})\,\mathcal{P}_\alpha\,(\mathbf{V} \otimes \mathbf{I}) = (\mathbf{I}_T - \mathbf{D}) \otimes \Omega + \Delta t\,\mathbf{I}_T \otimes \mathbf{A}$$

T shifted systems on $\mathbf{A}$’s pattern: $\sum_j \mathbf{W}_j \otimes \mathbf{A}_j$, complex diagonal $\mathbf{W}_j$.

(b) is outlook: none of our papers covers it yet (details and open points in backup).

(a) Krasnopolsky, CPC 229 (2018) 8–19; Makarashvili et al., CPC 219 (2017) 236–245; Plana-Riu et al., CPC 326 (2026) 110232. (b) Gander, Vandewalle, SISC 29(2) (2007) 556–578; Falgout et al., SISC 36(6) (2014) C635–C661; McDonald, Pestana, Wathen, SISC 40(2) (2018) A1012–A1033

Backup: all-at-once time stepping in Kronecker form

Outlook, not yet in our papers: standard time-parallel material in this talk's notation.

Implicit Euler, T steps

$$(\Omega + \Delta t\, \mathbf{A})\, \boldsymbol{u}^n - \Omega\, \boldsymbol{u}^{n-1} = \Delta t\, \boldsymbol{f}^n$$

$$\big(\mathbf{I}_T \otimes (\Omega + \Delta t\, \mathbf{A}) - \mathbf{S}_T \otimes \Omega\big)\, \boldsymbol{u} = \boldsymbol{b},$$

$\mathbf{S}_T$: ones on the subdiagonal.

M.J. Gander, S. Vandewalle, SISC 29(2) (2007) 556–578; R.D. Falgout et al., SISC 36(6) (2014) C635–C661; E. McDonald, J. Pestana, A. Wathen, SISC 40(2) (2018) A1012–A1033

Backup: all-at-once time stepping in Kronecker form

Outlook, not yet in our papers: standard time-parallel material in this talk's notation.

Implicit Euler, T steps

$$(\Omega + \Delta t\, \mathbf{A})\, \boldsymbol{u}^n - \Omega\, \boldsymbol{u}^{n-1} = \Delta t\, \boldsymbol{f}^n$$

$$\big(\mathbf{I}_T \otimes (\Omega + \Delta t\, \mathbf{A}) - \mathbf{S}_T \otimes \Omega\big)\, \boldsymbol{u} = \boldsymbol{b},$$

$\mathbf{S}_T$: ones on the subdiagonal.
Its product

$$\mathbf{Y} = (\Omega + \Delta t\, \mathbf{A})\, [\boldsymbol{u}^1 \cdots \boldsymbol{u}^T] - \Omega\, [\mathbf{0}\ \boldsymbol{u}^1 \cdots \boldsymbol{u}^{T-1}]$$

one SpMM with $K = T$, plus diagonal scalings.

M.J. Gander, S. Vandewalle, SISC 29(2) (2007) 556–578; R.D. Falgout et al., SISC 36(6) (2014) C635–C661; E. McDonald, J. Pestana, A. Wathen, SISC 40(2) (2018) A1012–A1033

Backup: all-at-once time stepping in Kronecker form

Outlook, not yet in our papers: standard time-parallel material in this talk's notation.

Implicit Euler, T steps

$$(\Omega + \Delta t\, \mathbf{A})\, \boldsymbol{u}^n - \Omega\, \boldsymbol{u}^{n-1} = \Delta t\, \boldsymbol{f}^n$$

$$\big(\mathbf{I}_T \otimes (\Omega + \Delta t\, \mathbf{A}) - \mathbf{S}_T \otimes \Omega\big)\, \boldsymbol{u} = \boldsymbol{b},$$

$\mathbf{S}_T$: ones on the subdiagonal.

Its product

$$\mathbf{Y} = (\Omega + \Delta t\, \mathbf{A})\, [\boldsymbol{u}^1 \cdots \boldsymbol{u}^T] - \Omega\, [\mathbf{0}\ \boldsymbol{u}^1 \cdots \boldsymbol{u}^{T-1}]$$

one SpMM with $K = T$, plus diagonal scalings.

ParaDiag-type preconditioner

$\mathbf{B}_\alpha = \mathbf{S}_T + \alpha\, \boldsymbol{e}_1 \boldsymbol{e}_T^T = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}$ (scaled DFT), and

$$\mathcal{P}_\alpha = (\mathbf{V} \otimes \mathbf{I})\big[(\mathbf{I}_T - \mathbf{D}) \otimes \Omega + \Delta t\, \mathbf{I}_T \otimes \mathbf{A}\big](\mathbf{V}^{-1} \otimes \mathbf{I})$$

T shifted complex systems $\big((1-d_k)\Omega + \Delta t\mathbf{A}\big)\boldsymbol{u}_k = \boldsymbol{r}_k$; a Krylov solve on all of them applies $\mathbf{A}$ as one SpMM.

M.J. Gander, S. Vandewalle, SISC 29(2) (2007) 556–578; R.D. Falgout et al., SISC 36(6) (2014) C635–C661; E. McDonald, J. Pestana, A. Wathen, SISC 40(2) (2018) A1012–A1033

Backup: all-at-once time stepping in Kronecker form

Outlook, not yet in our papers: standard time-parallel material in this talk's notation.

Implicit Euler, T steps

$$(\Omega + \Delta t\,\mathbf{A})\,\boldsymbol{u}^n - \Omega\,\boldsymbol{u}^{n-1} = \Delta t\,\boldsymbol{f}^n$$

$$\big(\mathbf{I}_T \otimes (\Omega + \Delta t\,\mathbf{A}) - \mathbf{S}_T \otimes \Omega\big)\,\boldsymbol{u} = \boldsymbol{b},$$

$\mathbf{S}_T$: ones on the subdiagonal.

Its product

$$\mathbf{Y} = (\Omega + \Delta t\,\mathbf{A})\,[\boldsymbol{u}^1 \cdots \boldsymbol{u}^T] - \Omega\,[\mathbf{0}\ \boldsymbol{u}^1 \cdots \boldsymbol{u}^{T-1}]$$

one SpMM with $K = T$, plus diagonal scalings.

ParaDiag-type preconditioner

$\mathbf{B}_\alpha = \mathbf{S}_T + \alpha\,\boldsymbol{e}_1\boldsymbol{e}_T^T = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}$ (scaled DFT), and

$$\mathcal{P}_\alpha = (\mathbf{V}\otimes\mathbf{I})\big[(\mathbf{I}_T - \mathbf{D})\otimes\Omega + \Delta t\,\mathbf{I}_T\otimes\mathbf{A}\big](\mathbf{V}^{-1}\otimes\mathbf{I})$$

T shifted complex systems $\big((1-d_k)\Omega + \Delta t\mathbf{A}\big)\boldsymbol{u}_k = \boldsymbol{r}_k$; a Krylov solve on all of them applies $\mathbf{A}$ as one SpMM.

Open points for NS

- Constant Δt and a linear operator: $\mathbf{C}(\boldsymbol{u})$ changes per step (per level: $\mathbf{M}\,\mathbf{U}_s\,\Gamma$, as in CPC).
- Convergence of time-parallel methods for advection-dominated flows.
- Memory: T copies of every field.

M.J. Gander, S. Vandewalle, SISC 29(2) (2007) 556–578; R.D. Falgout et al., SISC 36(6) (2014) C635–C661; E. McDonald, J. Pestana, A. Wathen, SISC 40(2) (2018) A1012–A1033

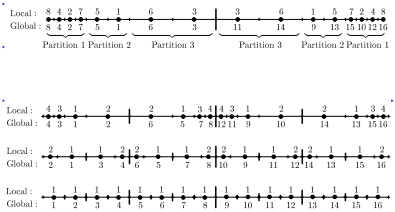
Symmetric domains and mirrored ordering





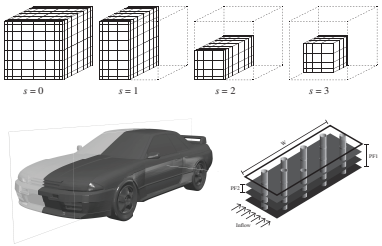
$s = 0, \ldots, 3$ reflection symmetries: only $1/2^s$ of the mesh is built

- Number the 2^s subdomains one after another; give each the base subdomain's local ordering, mirrored across the symmetry planes.



Alsalti-Baldellou et al., JCP 486 (2023) Sec. 2; Comput. Fluids 275 (2024) Sec. 3

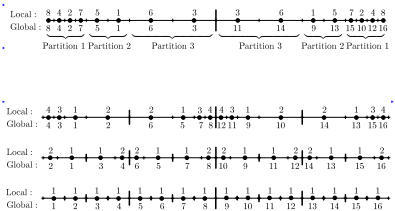
Symmetric domains and mirrored ordering



$s = 0, \ldots, 3$ reflection symmetries: only $1/2^s$ of the mesh is built

$\mathbf{A}_1$: couplings of the base subdomain with itself; $\mathbf{A}_{j>1}$: with its mirror images.

- Number the 2^s subdomains one after another; give each the base subdomain's local ordering, mirrored across the symmetry planes.



Any operator that depends only on the geometry then has repeated blocks:

$$s = 1 : \mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_2 & \mathbf{A}_1 \end{pmatrix}, \qquad s = 2 : \mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_3 & \mathbf{A}_4 \\ \mathbf{A}_2 & \mathbf{A}_1 & \mathbf{A}_4 & \mathbf{A}_3 \\ \mathbf{A}_3 & \mathbf{A}_4 & \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_4 & \mathbf{A}_3 & \mathbf{A}_2 & \mathbf{A}_1 \end{pmatrix}$$

Alsalti-Baldellou et al., JCP 486 (2023) Sec. 2; Comput. Fluids 275 (2024) Sec. 3

The physical basis: one SpMM plus a thin coupling

Splitting for any s

$$\mathbf{A} = \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}} + \mathbf{A}_{\text{out}}, \qquad \mathbf{A}_{\text{inn}} := \mathbf{A}_1 \in \mathbb{R}^{n/2^s \times n/2^s}, \qquad \mathbf{A}_{\text{out}} := \mathbf{A} - \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}}$$

- $\mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}}$: one SpMM with $K = 2^s$ vectors; only the base mesh is stored.
- $\mathbf{A}_{\text{out}}$: couplings across the symmetry planes; for compact stencils it is very sparse.

Alsalti-Baldellou et al., Comput. Fluids 275 (2024) eq. (split_s); JCP 486 (2023) eq. 24

The physical basis: one SpMM plus a thin coupling

Splitting for any s

$$\mathbf{A} = \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}} + \mathbf{A}_{\text{out}}, \qquad \mathbf{A}_{\text{inn}} := \mathbf{A}_1 \in \mathbb{R}^{n/2^s \times n/2^s}, \qquad \mathbf{A}_{\text{out}} := \mathbf{A} - \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}}$$

- $\mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}}$: one SpMM with $K = 2^s$ vectors; only the base mesh is stored.
- $\mathbf{A}_{\text{out}}$: couplings across the symmetry planes; for compact stencils it is very sparse.
- Asymmetric boundary conditions are allowed: their entries go into $\mathbf{A}_{\text{out}}$. Only the geometry has to be symmetric.
- Operators acting identically on every subdomain (e.g. $\mathbf{M}$) are exactly $\mathbf{I}_{2^s} \otimes \mathbf{A}$.

Alsalti-Baldellou et al., Comput. Fluids 275 (2024) eq. (split_s); JCP 486 (2023) eq. 24

The physical basis: one SpMM plus a thin coupling

Splitting for any s

$$\mathbf{A} = \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}} + \mathbf{A}_{\text{out}}, \qquad \mathbf{A}_{\text{inn}} := \mathbf{A}_1 \in \mathbb{R}^{n/2^s \times n/2^s}, \qquad \mathbf{A}_{\text{out}} := \mathbf{A} - \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}}$$

- $\mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}}$: one SpMM with $K = 2^s$ vectors; only the base mesh is stored.
- $\mathbf{A}_{\text{out}}$: couplings across the symmetry planes; for compact stencils it is very sparse.
- Asymmetric boundary conditions are allowed: their entries go into $\mathbf{A}_{\text{out}}$. Only the geometry has to be symmetric.
- Operators acting identically on every subdomain (e.g. $\mathbf{M}$) are exactly $\mathbf{I}_{2^s} \otimes \mathbf{A}$.

A group-algebra view

With $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ and $\vec{\pi} \in \{0,1\}^s$:

$$\mathbf{A} = \sum_{\vec{\pi}} \left(X^{\pi_1} \otimes \cdots \otimes X^{\pi_s} \right) \otimes \mathbf{A}_{\vec{\pi}}$$

a sum of Kronecker products whose small factors are the permutation matrices of $(\mathbb{Z}_2)^s$. Equivalent to JCP eqs. 12–14; checked numerically on a $2 \times 2 \times 4$ Laplacian.

In general, $\hat{\mathbf{A}} = \sum_j \mathbf{W}_j \otimes \mathbf{A}_j$ gives $\mathbf{Y} = \sum_j \mathbf{A}_j \, (\mathbf{X}\, \mathbf{W}_j^T)$: one SpMM per term plus a small dense product.

Alsalti-Baldellou et al., Comput. Fluids 275 (2024) eq. (split_s); JCP 486 (2023) eq. 24

Change of basis: 2^s decoupled Poisson systems

Walsh–Hadamard change of basis (JCP eqs. 15–16)

$$\mathbf{P}_s = \frac{1}{\sqrt{2^s}} \left(\bigotimes_{i=1}^{s} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right) \otimes \mathbf{I}_{n/2^s}, \qquad \mathbf{P}_s = \mathbf{P}_s^T = \mathbf{P}_s^{-1}$$

Alsalti-Baldellou et al., JCP 486 (2023), Props. 1–5, Thm. 1, Cor. 1, Alg. 1

Change of basis: 2^s decoupled Poisson systems

Walsh–Hadamard change of basis (JCP eqs. 15–16)

$$\mathbf{P}_s = \frac{1}{\sqrt{2^s}} \Big(\bigotimes_{i=1}^{s} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \Big) \otimes \mathbf{I}_{n/2^s}, \qquad \mathbf{P}_s = \mathbf{P}_s^T = \mathbf{P}_s^{-1}$$

Theorem 1 (JCP eq. 17)

$$\mathbf{P}_s \mathbf{L} \mathbf{P}_s^{-1} = \sum_{\vec{\pi} \in \{0,1\}^s} \mathbf{D}_{\vec{\pi}} \otimes \mathbf{L}_{\vec{\pi}}, \qquad \mathbf{D}_{\vec{\pi}} = Z^{\pi_1} \otimes \cdots \otimes Z^{\pi_s}, \;\; Z = \mathrm{diag}(1, -1)$$

All small factors are diagonal: $\mathbf{P}_s \mathbf{L} \mathbf{P}_s^{-1} = \mathrm{blkdiag}(\hat{\mathbf{L}}_1, \ldots, \hat{\mathbf{L}}_{2^s})$.

Alsalti-Baldellou et al., JCP 486 (2023), Props. 1–5, Thm. 1, Cor. 1, Alg. 1

Change of basis: 2^s decoupled Poisson systems

Walsh–Hadamard change of basis (JCP eqs. 15–16)

$$\mathbf{P}_s = \frac{1}{\sqrt{2^s}} \Big(\bigotimes_{i=1}^{s} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \Big) \otimes \mathbf{I}_{n/2^s}, \qquad \mathbf{P}_s = \mathbf{P}_s^T = \mathbf{P}_s^{-1}$$

Theorem 1 (JCP eq. 17)

$$\mathbf{P}_s \mathbf{L} \mathbf{P}_s^{-1} = \sum_{\vec{\pi} \in \{0,1\}^s} \mathbf{D}_{\vec{\pi}} \otimes \mathbf{L}_{\vec{\pi}}, \qquad \mathbf{D}_{\vec{\pi}} = Z^{\pi_1} \otimes \cdots \otimes Z^{\pi_s}, \;\; Z = \mathrm{diag}(1, -1)$$

All small factors are diagonal: $\mathbf{P}_s \mathbf{L} \mathbf{P}_s^{-1} = \mathrm{blkdiag}(\hat{\mathbf{L}}_1, \ldots, \hat{\mathbf{L}}_{2^s})$.

Solve

$\hat{\boldsymbol{b}} = \mathbf{P}_s \boldsymbol{b}$; $\hat{\mathbf{L}}_j \hat{\boldsymbol{x}}_j = \hat{\boldsymbol{b}}_j$, $j = 1, \ldots, 2^s$; $\boldsymbol{x} = \mathbf{P}_s \hat{\boldsymbol{x}}$.

- Spectrum, symmetry and (semi-)definiteness preserved.
- Only $\hat{\mathbf{L}}_1 = \sum_{\vec{\pi}} \mathbf{L}_{\vec{\pi}}$ keeps the constant kernel; the others are SPD.

Alsalti-Baldellou et al., JCP 486 (2023), Props. 1–5, Thm. 1, Cor. 1, Alg. 1

Change of basis: 2^s decoupled Poisson systems

Walsh–Hadamard change of basis (JCP eqs. 15–16)

$$\mathbf{P}_s = \frac{1}{\sqrt{2^s}} \Big(\bigotimes_{i=1}^{s} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \Big) \otimes \mathbf{I}_{n/2^s}, \qquad \mathbf{P}_s = \mathbf{P}_s^T = \mathbf{P}_s^{-1}$$

Theorem 1 (JCP eq. 17)

$$\mathbf{P}_s \mathbf{L} \mathbf{P}_s^{-1} = \sum_{\vec{\pi} \in \{0,1\}^s} \mathbf{D}_{\vec{\pi}} \otimes \mathbf{L}_{\vec{\pi}}, \qquad \mathbf{D}_{\vec{\pi}} = Z^{\pi_1} \otimes \cdots \otimes Z^{\pi_s}, \;\; Z = \mathrm{diag}(1, -1)$$

All small factors are diagonal: $\mathbf{P}_s \mathbf{L} \mathbf{P}_s^{-1} = \mathrm{blkdiag}(\hat{\mathbf{L}}_1, \ldots, \hat{\mathbf{L}}_{2^s})$.

Solve

$\hat{\boldsymbol{b}} = \mathbf{P}_s \boldsymbol{b}$; $\hat{\mathbf{L}}_j \hat{\boldsymbol{x}}_j = \hat{\boldsymbol{b}}_j$, $j = 1, \ldots, 2^s$; $\boldsymbol{x} = \mathbf{P}_s \hat{\boldsymbol{x}}$.

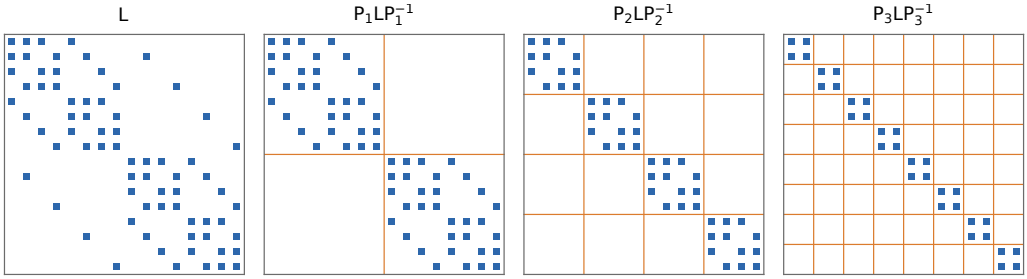
- Spectrum, symmetry and (semi-)definiteness preserved.
- Only $\hat{\mathbf{L}}_1 = \sum_{\vec{\pi}} \mathbf{L}_{\vec{\pi}}$ keeps the constant kernel; the others are SPD.

Why it works

$\mathbf{P}_s$ is the Fourier transform on $(\mathbb{Z}_2)^s$: it diagonalises every $X^{\pi_1} \otimes \cdots \otimes X^{\pi_s}$ at once, since $\frac{1}{2} H X H = Z$. Hence the block diagonalisation.

Alsalti-Baldellou et al., JCP 486 (2023), Props. 1–5, Thm. 1, Cor. 1, Alg. 1

Block diagonalisation on a $2 \times 2 \times 4$ mesh



7-point Laplacian, Neumann boundaries, mirrored ordering, $s = 3$; orange lines: the 2^k subsystems after k transforms.

- Each $\mathbf{P}_k$ halves the coupled blocks; $k = s$ leaves 2^s decoupled $\hat{\mathbf{L}}_j$ on the base mesh.
- Checked numerically: $\mathbf{P}_k$ orthogonal, eigenvalues preserved, Theorem 1 holds, only $\hat{\mathbf{L}}_1$ has the constant kernel.

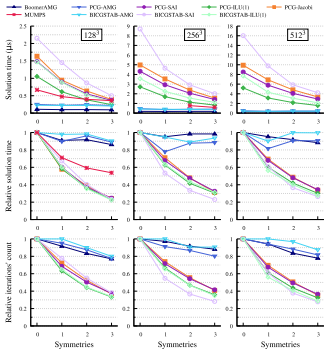
Reproduces JCP 486 (2023) Fig. 4; script `scripts/sym-blockdiag.py`

Poisson: fewer iterations and less memory

Rayleigh–Bénard Poisson system, $Ra = 10^9$

JFF cluster (2× Xeon 6230 per node), 128^3–512^3, $s \leq 3$, hypre and MUMPS; subsystems solved with separate library calls.

- PCG/BiCGSTAB with SAI or ILU: $\approx$ 30%, 50%, 70% fewer iterations for $s = 1, 2, 3$; up to 4.3× faster (Sec. 4.1; the conclusions say 3.6×).
- AMG: up to 23% fewer iterations, up to 1.3× faster.
- MUMPS: memory, setup, solve down by up to 48%, 58%, 46% (Sec. 4); exploiting symmetries made 256^3 feasible.
- The transforms cost $\leq 0.2\%$ of the solve.



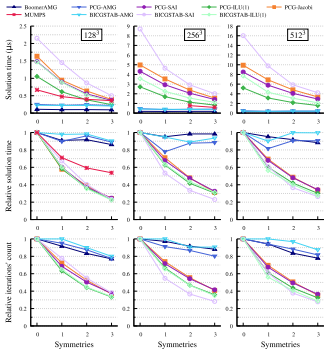
Alsalti-Baldellou et al., JCP 486 (2023), Sec. 4, Figs. 9–11

Poisson: fewer iterations and less memory

Rayleigh–Bénard Poisson system, $Ra = 10^9$

JFF cluster (2× Xeon 6230 per node), 128^3–512^3, $s \leq 3$, hypre and MUMPS; subsystems solved with separate library calls.

- PCG/BiCGSTAB with SAI or ILU: $\approx$ 30%, 50%, 70% fewer iterations for $s = 1, 2, 3$; up to 4.3× faster (Sec. 4.1; the conclusions say 3.6×).
- AMG: up to 23% fewer iterations, up to 1.3× faster.
- MUMPS: memory, setup, solve down by up to 48%, 58%, 46% (Sec. 4); exploiting symmetries made 256^3 feasible.
- The transforms cost $\leq 0.2\%$ of the solve.



Limitations

Condition number barely improves; we argue faster convergence qualitatively. SpMM products ran up to 2.5× faster; we projected the whole-solver gains from SpMM, not measured them.

Alsalti-Baldellou et al., JCP 486 (2023), Sec. 4, Figs. 9–11

All operators in the physical basis: SymSpMV

SymSpMV: fused $\mathrm{SpMM}(\mathbf{A}_{\mathrm{inn}}, \mathbf{X}) + \mathrm{SpMV}(\mathbf{A}_{\mathrm{out}}, \boldsymbol{x})$, no intermediate result stored

Implementation	Bytes moved (square $\mathbf{A}$, FP64/int32)
SpMV	$12\,\mathrm{nnz}(\mathbf{I} \otimes \mathbf{A}_{\mathrm{inn}} + \mathbf{A}_{\mathrm{out}}) + 4(n+1) + 16n$
SpMM + SpMV	$\left(12\,\mathrm{nnz}(\mathbf{A}_{\mathrm{inn}}) + 4(n/2^s + 1) + 16n\right) + \left(12\,\mathrm{nnz}(\mathbf{A}_{\mathrm{out}}) + 4(n+1) + 16n\right)$
SymSpMV	$12\,\mathrm{nnz}(\mathbf{A}_{\mathrm{inn}} + \mathbf{A}_{\mathrm{out}}) + 4(n+1) + 4(n/2^s + 1) + 16n$

Saving of SymSpMV against SpMV: $12(2^s - 1)\,\mathrm{nnz}(\mathbf{A}_{\mathrm{inn}}) - 4(n/2^s + 1)$ bytes.

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Alg. 2, Table 1, Sec. 3.2, Fig. 7

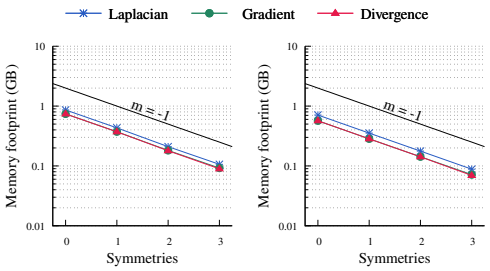
All operators in the physical basis: SymSpMV

SymSpMV: fused $\mathrm{SpMM}(\mathbf{A}_{\mathrm{inn}}, \mathbf{X}) + \mathrm{SpMV}(\mathbf{A}_{\mathrm{out}}, \boldsymbol{x})$, no intermediate result stored

Implementation	Bytes moved (square $\mathbf{A}$, FP64/int32)
SpMV	$12\,\mathrm{nnz}(\mathbf{I} \otimes \mathbf{A}_{\mathrm{inn}} + \mathbf{A}_{\mathrm{out}}) + 4(n+1) + 16n$
SpMM + SpMV	$\left(12\,\mathrm{nnz}(\mathbf{A}_{\mathrm{inn}}) + 4(n/2^s + 1) + 16n\right) + \left(12\,\mathrm{nnz}(\mathbf{A}_{\mathrm{out}}) + 4(n+1) + 16n\right)$
SymSpMV	$12\,\mathrm{nnz}(\mathbf{A}_{\mathrm{inn}} + \mathbf{A}_{\mathrm{out}}) + 4(n+1) + 4(n/2^s + 1) + 16n$

Saving of SymSpMV against SpMV: $12(2^s - 1)\,\mathrm{nnz}(\mathbf{A}_{\mathrm{inn}}) - 4(n/2^s + 1)$ bytes.

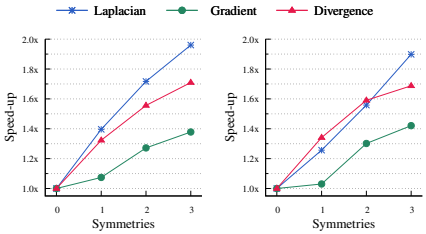
- Partition the base mesh only, replicate to the images: one MPI message per neighbour, as before.
- GPU: one work-item per output entry; mirror images in the same warp share the coefficient, 2–3× faster than a loop over K per thread ($K = 8$).
- Footprint of the matrices up to 8× smaller ($s = 3$); not exactly 2^s because $\mathbf{A}_{\mathrm{out}}$ is stored explicitly.



Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Alg. 2, Table 1, Sec. 3.2, Fig. 7

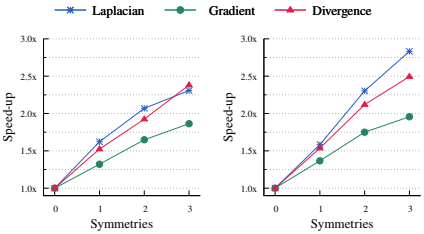
Fair speed-ups: fixed base mesh, larger problem

CPU: MareNostrum 4 node, 2× Xeon 8160



up to $2.0\times$ ($\mathbf{L}$), $1.7\times$ ($\mathbf{M}$), $1.4\times$ ($\mathbf{G}$)

GPU: NVIDIA RTX A5000



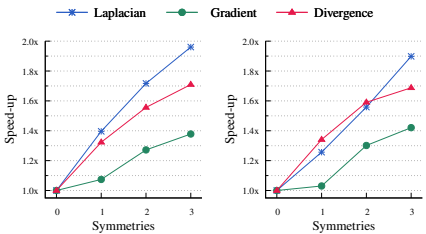
up to $3.3\times$ ($\mathbf{L}$), $2.8\times$ ($\mathbf{M}$), $2.2\times$ ($\mathbf{G}$)

- Cube (15.5M cells) and finned-tube exchanger (17.7M cells); SymSpMV against SpMV; ranking follows intensity.

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Figs. 8–9, Sec. 4

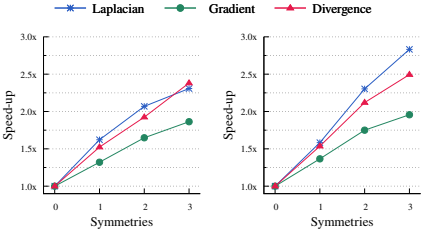
Fair speed-ups: fixed base mesh, larger problem

CPU: MareNostrum 4 node, 2× Xeon 8160



up to $2.0\times$ ($\mathbf{L}$), $1.7\times$ ($\mathbf{M}$), $1.4\times$ ($\mathbf{G}$)

GPU: NVIDIA RTX A5000



up to $3.3\times$ ($\mathbf{L}$), $2.8\times$ ($\mathbf{M}$), $2.2\times$ ($\mathbf{G}$)

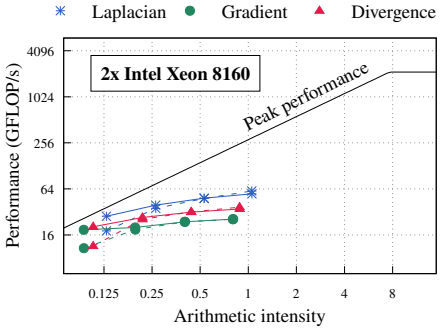
- Cube (15.5M cells) and finned-tube exchanger (17.7M cells); SymSpMV against SpMV; ranking follows intensity.

The $5.0\times$ headline is a cache effect

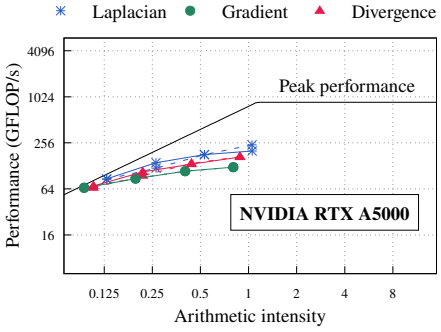
It comes from halving the base mesh at fixed problem size: $\mathbf{L}_{\text{inn}}$ then fits in the 33 MB L3 (already $2.9\times$ at $s = 1$). We do not expect it in target applications; $2.0\times$ (CPU) and $3.3\times$ (GPU) are the fair numbers.

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Figs. 8–9, Sec. 4

Still memory-bound, but further right



CPU, 2× Xeon 8160, 128 GB/s

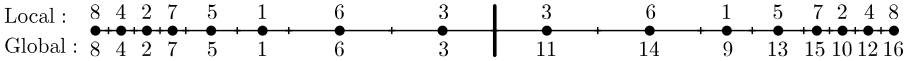


GPU, RTX A5000, 768 GB/s

- Every product stays memory-bound; intensity rises with s as the traffic model predicts.
- On the A5000, $s = 3$ approaches the ridge point, because the FP64 peak of that GPU is low.
- SymSpMV is still far from its theoretical peak.

Alsalti-Baldellou et al., Comput. Fluids 275 (2024), Fig. 10

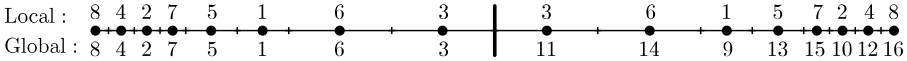
Backup: mirrored ordering in 1D



- One reflection, $s = 1$: number the left half, then give the right half the mirrored local order.

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133; Comput. Fluids 275 (2024) 106247

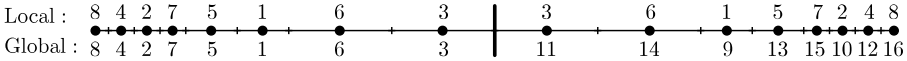
Backup: mirrored ordering in 1D



- One reflection, $s = 1$: number the left half, then give the right half the mirrored local order.
- Each partition of the base half is replicated to its mirror image: MPI halos of all images go in one message.

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133; Comput. Fluids 275 (2024) 106247

Backup: mirrored ordering in 1D

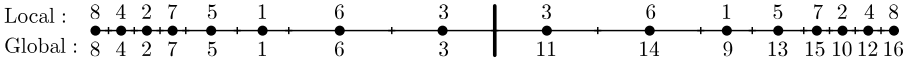


- One reflection, $s = 1$: number the left half, then give the right half the mirrored local order.
- Each partition of the base half is replicated to its mirror image: MPI halos of all images go in one message.
- Any geometry-only operator then reads

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_2 & \mathbf{A}_1 \end{pmatrix} = \mathbf{I}_2 \otimes \mathbf{A}_1 + \mathbf{A}_{\text{out}}$$

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133; Comput. Fluids 275 (2024) 106247

Backup: mirrored ordering in 1D

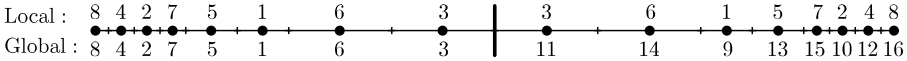


- One reflection, $s = 1$: number the left half, then give the right half the mirrored local order.
- Each partition of the base half is replicated to its mirror image: MPI halos of all images go in one message.
- Any geometry-only operator then reads

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_2 & \mathbf{A}_1 \end{pmatrix} = \mathbf{I}_2 \otimes \mathbf{A}_1 + \mathbf{A}_{\text{out}}$$

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133; Comput. Fluids 275 (2024) 106247

Backup: mirrored ordering in 1D



- One reflection, $s = 1$: number the left half, then give the right half the mirrored local order.
- Each partition of the base half is replicated to its mirror image: MPI halos of all images go in one message.
- Any geometry-only operator then reads

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{A}_2 & \mathbf{A}_1 \end{pmatrix} = \mathbf{I}_2 \otimes \mathbf{A}_1 + \mathbf{A}_{\text{out}}$$

Change of basis

$$\mathbf{P} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{pmatrix},$$

$$\mathbf{P}\mathbf{A}\mathbf{P}^{-1} = \mathrm{blkdiag}(\mathbf{A}_1 + \mathbf{A}_2,\ \mathbf{A}_1 - \mathbf{A}_2)$$

Asymmetric boundary conditions change only a few rows: those entries go into $\mathbf{A}_{\text{out}}$.

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133; Comput. Fluids 275 (2024) 106247

Repeated geometry: the inner–interface ordering

Classify unknowns

- inner: coupled only to its own block;
- interface: coupled to another block.

Give every block the same local ordering of its inner unknowns; put the interface last.

À. Alsalti-Baldellou, C. Janna, X. Álvarez-Farré, F.X. Trias, SIAM J. Sci. Comput. 46(6) (2024) B860–B883

Repeated geometry: the inner–interface ordering

Classify unknowns

- inner: coupled only to its own block;
- interface: coupled to another block.

Give every block the same local ordering of its inner unknowns; put the interface last.

$$\mathbf{A} = \begin{pmatrix} \mathbf{I}_{n_b} \otimes \mathbf{A}_{\mathcal{I}} & \mathbf{I}_{n_b} \otimes \mathbf{B} \\ \mathbf{I}_{n_b} \otimes \mathbf{B}^T & \mathbf{A}_{\Gamma\Gamma} \end{pmatrix}$$

- $\mathbf{A}_{\mathcal{I}}$, $\mathbf{B}$: one block's inner and inner–interface couplings, stored once;
- $\mathbf{A}_{\Gamma\Gamma}$: interface block, any structure and ordering;
- boundary conditions need not repeat (SISC: AMGR imposes no condition on them).

À. Alsalti-Baldellou, C. Janna, X. Álvarez-Farré, F.X. Trias, SIAM J. Sci. Comput. 46(6) (2024) B860–B883

Repeated geometry: the inner–interface ordering

Classify unknowns

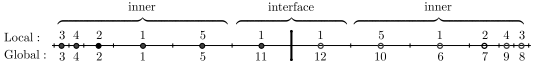
- inner: coupled only to its own block;
- interface: coupled to another block.

Give every block the same local ordering of its inner unknowns; put the interface last.

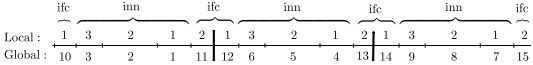
$$\mathbf{A} = \begin{pmatrix} \mathbf{I}_{n_b} \otimes \mathbf{A}_{\mathcal{I}} & \mathbf{I}_{n_b} \otimes \mathbf{B} \\ \mathbf{I}_{n_b} \otimes \mathbf{B}^T & \mathbf{A}_{\Gamma\Gamma} \end{pmatrix}$$

- $\mathbf{A}_{\mathcal{I}}$, $\mathbf{B}$: one block's inner and inner–interface couplings, stored once;
- $\mathbf{A}_{\Gamma\Gamma}$: interface block, any structure and ordering;
- boundary conditions need not repeat (SISC: AMGR imposes no condition on them).

Reflection ($n_b = 2$)



Translation ($n_b = 3$)



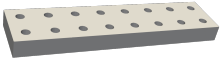
Every product by the inner rows is $\mathbf{I}_{n_b} \otimes \mathbf{A}_{\mathcal{I}}$: an SpMM with $K = n_b$.

À. Alsalti-Baldellou, C. Janna, X. Álvarez-Farré, F.X. Trias, SIAM J. Sci. Comput. 46(6) (2024) B860–B883

Reflection, translation, rotation



DrivAer: one reflection, $n_b = 2$, 106.4 M unknowns



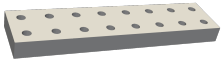
Finned-tube heat exchanger: three reflections, $n_b \leq 8$, 18.4 M unknowns

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

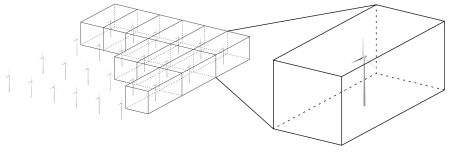
Reflection, translation, rotation



DrivAer: one reflection, $n_b = 2$, 106.4 M unknowns



Finned-tube heat exchanger: three reflections, $n_b \leq 8$, 18.4 M unknowns



6×4 wind farm: translation, $n_b \leq 24$, 52 M unknowns

Reflections give $n_b = 2^s \leq 8$. Translations and rotations are not bounded: in principle one turbine meshed, the whole farm solved.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

AMGR: put the SpMM where the blocks repeat

Multigrid reduction

$$\mathcal{P} = \begin{pmatrix} \mathbf{Z} \\ \mathbf{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c}, \qquad \mathbf{A}_c = \mathcal{P}^T \mathbf{A}\, \mathcal{P}$$

- Coarse nodes: all interface nodes plus an aggressive selection of inner nodes (independent set on T^k, $k \leq 2$ for ext+i interpolation).

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

AMGR: put the SpMM where the blocks repeat

Multigrid reduction

$$\mathcal{P} = \begin{pmatrix} \mathbf{Z} \\ \mathbf{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c}, \qquad \mathbf{A}_c = \mathcal{P}^T \mathbf{A}\, \mathcal{P}$$

- Coarse nodes: all interface nodes plus an aggressive selection of inner nodes (independent set on T^k, $k \leq 2$ for ext+i interpolation).
- Top-level smoother: FSAI of the inner block,

$$\mathbf{G}_{\mathcal{I}} = \mathbf{I}_{n_b} \otimes \mathbf{G}, \qquad \mathbf{G}^T \mathbf{G} \approx \mathbf{A}_{\mathcal{I}}^{-1},$$

applied as an SpMM with $K = n_b$.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

AMGR: put the SpMM where the blocks repeat

Multigrid reduction

$$\mathcal{P} = \begin{pmatrix} \mathbf{Z} \\ \mathbf{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c}, \qquad \mathbf{A}_c = \mathcal{P}^T \mathbf{A}\, \mathcal{P}$$

- Coarse nodes: all interface nodes plus an aggressive selection of inner nodes (independent set on T^k, $k \leq 2$ for ext+i interpolation).
- Top-level smoother: FSAI of the inner block,

$$\mathbf{G}_{\mathcal{I}} = \mathbf{I}_{n_b} \otimes \mathbf{G}, \qquad \mathbf{G}^T \mathbf{G} \approx \mathbf{A}_{\mathcal{I}}^{-1},$$

  applied as an SpMM with $K = n_b$.
- Setup and memory of the smoother are n_b times smaller: computed on one block only.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

AMGR: put the SpMM where the blocks repeat

Multigrid reduction

$$\mathcal{P} = \begin{pmatrix} \mathbf{Z} \\ \mathbf{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c}, \qquad \mathbf{A}_c = \mathcal{P}^T \mathbf{A}\, \mathcal{P}$$

- Coarse nodes: all interface nodes plus an aggressive selection of inner nodes (independent set on T^k, $k \leq 2$ for ext+i interpolation).
- Top-level smoother: FSAI of the inner block,

$$\mathbf{G}_{\mathcal{I}} = \mathbf{I}_{n_b} \otimes \mathbf{G}, \qquad \mathbf{G}^T \mathbf{G} \approx \mathbf{A}_{\mathcal{I}}^{-1},$$

  applied as an SpMM with $K = n_b$.
- Setup and memory of the smoother are n_b times smaller: computed on one block only.
- $\mathbf{A}_c$ is solved with standard AMG.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

AMGR: put the SpMM where the blocks repeat

Multigrid reduction

$$\mathcal{P} = \begin{pmatrix} \mathbf{Z} \\ \mathbf{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c}, \qquad \mathbf{A}_c = \mathcal{P}^T \mathbf{A}\, \mathcal{P}$$

- Coarse nodes: all interface nodes plus an aggressive selection of inner nodes (independent set on T^k, $k \leq 2$ for ext+i interpolation).
- Top-level smoother: FSAI of the inner block,

$$\mathbf{G}_{\mathcal{I}} = \mathbf{I}_{n_b} \otimes \mathbf{G}, \qquad \mathbf{G}^T \mathbf{G} \approx \mathbf{A}_{\mathcal{I}}^{-1},$$

  applied as an SpMM with $K = n_b$.
- Setup and memory of the smoother are n_b times smaller: computed on one block only.
- $\mathbf{A}_c$ is solved with standard AMG.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

AMGR: put the SpMM where the blocks repeat

Multigrid reduction

$$\mathcal{P} = \begin{pmatrix} \mathbf{Z} \\ \mathbf{I}_{n_c} \end{pmatrix} \in \mathbb{R}^{n \times n_c}, \qquad \mathbf{A}_c = \mathcal{P}^T \mathbf{A}\, \mathcal{P}$$

- Coarse nodes: all interface nodes plus an aggressive selection of inner nodes (independent set on T^k, $k \leq 2$ for ext+i interpolation).
- Top-level smoother: FSAI of the inner block,

$$\mathbf{G}_{\mathcal{I}} = \mathbf{I}_{n_b} \otimes \mathbf{G}, \qquad \mathbf{G}^T \mathbf{G} \approx \mathbf{A}_{\mathcal{I}}^{-1},$$

  applied as an SpMM with $K = n_b$.
- Setup and memory of the smoother are n_b times smaller: computed on one block only.
- $\mathbf{A}_c$ is solved with standard AMG.

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883

What it does not change

- AMG's convergence is kept (model problem 64^3: 9 vs. 6 iterations).
- Weak and strong scaling: no significant degradation on the unit cube.
- SpMM is used only on the inner unknowns, so the interface may have any ordering.

Why not low-rank corrections?

Reusing the FSAI of $\mathbf{A}_{\mathcal{I}}$ with a low-rank correction works for FSAI, but not for AMG: LRCAMG needs 15–16 iterations vs. 6 for AMG on the model problem.

Outlook: periodic arrays are block-circulant

Interpretation, not in the papers

Not yet shown or tested in our SISC, JCP or C&F papers.

A periodic array of n_b identical blocks

Ordered block after block, with $\mathbf{S}$ the $n_b \times n_b$ cyclic shift:

$$\mathbf{A} = \sum_r \mathbf{S}^r \otimes \mathbf{A}_r \quad \left(\text{e.g. } \mathbf{I} \otimes \mathbf{A}_0 + \mathbf{S} \otimes \mathbf{A}_1 + \mathbf{S}^T \otimes \mathbf{A}_1^T\right)$$

Outlook: periodic arrays are block-circulant

Interpretation, not in the papers

Not yet shown or tested in our SISC, JCP or C&F papers.

A periodic array of n_b identical blocks

Ordered block after block, with $\mathbf{S}$ the $n_b \times n_b$ cyclic shift:

$$\mathbf{A} = \sum_r \mathbf{S}^r \otimes \mathbf{A}_r \quad \left(\text{e.g. } \mathbf{I} \otimes \mathbf{A}_0 + \mathbf{S} \otimes \mathbf{A}_1 + \mathbf{S}^T \otimes \mathbf{A}_1^T\right)$$

$\mathbf{S} = \mathbf{F}_{n_b}^* \Lambda \mathbf{F}_{n_b}$, $\lambda_k = \mathrm{e}^{2\pi \mathrm{i} k/n_b}$, so

$$(\mathbf{F}_{n_b} \otimes \mathbf{I})\, \mathbf{A}\, (\mathbf{F}_{n_b} \otimes \mathbf{I})^* = \mathrm{blkdiag}_k \left(\sum_r \lambda_k^r \mathbf{A}_r \right) :$$

n_b decoupled systems on one block's pattern.

Outlook: periodic arrays are block-circulant

Interpretation, not in the papers

Not yet shown or tested in our SISC, JCP or C&F papers.

A periodic array of n_b identical blocks

Ordered block after block, with $\mathbf{S}$ the $n_b \times n_b$ cyclic shift:

$$\mathbf{A} = \sum_r \mathbf{S}^r \otimes \mathbf{A}_r \quad \left(\text{e.g. } \mathbf{I} \otimes \mathbf{A}_0 + \mathbf{S} \otimes \mathbf{A}_1 + \mathbf{S}^T \otimes \mathbf{A}_1^T\right)$$

$\mathbf{S} = \mathbf{F}_{n_b}^* \Lambda \mathbf{F}_{n_b}$, $\lambda_k = \mathrm{e}^{2\pi \mathrm{i} k/n_b}$, so

$$(\mathbf{F}_{n_b} \otimes \mathbf{I})\, \mathbf{A}\, (\mathbf{F}_{n_b} \otimes \mathbf{I})^* = \mathrm{blkdiag}_k \Big(\sum_r \lambda_k^r \mathbf{A}_r \Big) :$$

n_b decoupled systems on one block's pattern.

Why it fits the talk

- Reflections are the same statement on $(\mathbb{Z}_2)^s$: the Walsh–Hadamard $\mathbf{P}_s$ of JCP plays the role of $\mathbf{F}$.
- Every product by $\mathbf{A}_r$ is an SpMM over n_b (complex) vectors.

Costs and conditions

- needs periodicity across the array (a real wind farm has inflow and outflow);
- complex arithmetic: twice the vector bytes;
- SISC treats the non-periodic case with MGR instead.

Backup: AMGR tables (first coarse level)

Ratio and nnz/row: first coarse level; time-to-solution speed-up of the PCG solve.

DrivAer, 106.4 M, 5 JFF nodes

	n_b	ratio	nnz/row	its	speed-up
AMG	1	0.36	14.7	26	1.00
AMGR	2	0.14	37.4	26	1.43

Heat exchanger, 18.4 M, 2 nodes

	n_b	ratio	nnz/row	its	speed-up
AMG	1	0.36	14.5	20	1.00
AMGR	2	0.14	37.5	19	1.38
AMGR	4	0.15	37.4	19	1.50
AMGR	8	0.15	37.6	18	1.68

Wind farm 6×4, 52 M, 1 node

	n_b	ratio	nnz/row	its	speed-up
AMG	1	0.41	13.5	67	1.00
AMGR	2	0.23	21.3	49	1.65
AMGR	4	0.24	20.4	50	1.64
AMGR	24	0.26	19.9	47	1.51

On the wind farm the coarsening ratio grows with n_b (more interface), and the coarse AMG solve takes back the smoother's SpMM gain.

[TODO: values from the review preprint (~/Git/symMGR); AMG wind-farm iterations (67) marked unsure there; verify against the published SISC version]

À. Alsalti-Baldellou et al., SIAM J. Sci. Comput. 46(6) (2024) B860–B883, Tables on DrivAer, heat exchanger, wind farm

A real DNS/LES code needs more than a fast kernel

What has to fit the operator framework

- Boundary conditions: local, often irregular.

A real DNS/LES code needs more than a fast kernel

What has to fit the operator framework

- Boundary conditions: local, often irregular.
- Nonlinear convection: $\mathbf{C}(\boldsymbol{u})$ changes every step.

A real DNS/LES code needs more than a fast kernel

What has to fit the operator framework

- Boundary conditions: local, often irregular.
- Nonlinear convection: $\mathbf{C}(\boldsymbol{u})$ changes every step.
- Time-step control: a CFL estimate every step.

A real DNS/LES code needs more than a fast kernel

What has to fit the operator framework

- Boundary conditions: local, often irregular.
- Nonlinear convection: $\mathbf{C}(\boldsymbol{u})$ changes every step.
- Time-step control: a CFL estimate every step.
- Poisson solver: the dominant cost of a projection method.

A real DNS/LES code needs more than a fast kernel

What has to fit the operator framework

- Boundary conditions: local, often irregular.
- Nonlinear convection: $\mathbf{C}(\boldsymbol{u})$ changes every step.
- Time-step control: a CFL estimate every step.
- Poisson solver: the dominant cost of a projection method.
- Portability: CPU and GPU from one code.

A real DNS/LES code needs more than a fast kernel

What has to fit the operator framework

- Boundary conditions: local, often irregular.
- Nonlinear convection: $\mathbf{C}(\boldsymbol{u})$ changes every step.
- Time-step control: a CFL estimate every step.
- Poisson solver: the dominant cost of a projection method.
- Portability: CPU and GPU from one code.

A real DNS/LES code needs more than a fast kernel

What has to fit the operator framework

- Boundary conditions: local, often irregular.
- Nonlinear convection: $\mathbf{C}(\boldsymbol{u})$ changes every step.
- Time-step control: a CFL estimate every step.
- Poisson solver: the dominant cost of a projection method.
- Portability: CPU and GPU from one code.

The rule we keep

Every ingredient is written as sparse operators and the same few kernels. Then each one inherits the SpMM rewrite when $K > 1$.

A real DNS/LES code needs more than a fast kernel

What has to fit the operator framework

- Boundary conditions: local, often irregular.
- Nonlinear convection: $\mathbf{C}(\boldsymbol{u})$ changes every step.
- Time-step control: a CFL estimate every step.
- Poisson solver: the dominant cost of a projection method.
- Portability: CPU and GPU from one code.

The rule we keep

Every ingredient is written as sparse operators and the same few kernels. Then each one inherits the SpMM rewrite when $K > 1$.

Companion talk

F.X. Trias, 09:10 in this session:
“Structure-preserving multiphysics simulations using portable algebraic kernels”.

Boundary conditions are sparse operators too

Affine operators

$$\mathbf{L}\,\boldsymbol{u} \;\longrightarrow\; \mathbf{L}\,\boldsymbol{u} + \mathbf{L}_b\,\boldsymbol{u}_b$$

- $\mathbf{L}_b$: sparse map from boundary data to the adjacent cells; same SpMV/SpMM kernel, no special code path.
- K fields or realisations with the same BC types but different data: $\mathbf{I}_K \otimes \mathbf{L}_b$, again an SpMM.
- Different BC types per realisation give different operators: no Kronecker structure.

À. Alsalti-Baldellou et al., Comput. Fluids 275 (2024) 106247; SIAM J. Sci. Comput. 46(6) (2024) B860–B883

Boundary conditions are sparse operators too

Affine operators

$$\mathbf{L}\,\boldsymbol{u} \;\longrightarrow\; \mathbf{L}\,\boldsymbol{u} + \mathbf{L}_b\,\boldsymbol{u}_b$$

- $\mathbf{L}_b$: sparse map from boundary data to the adjacent cells; same SpMV/SpMM kernel, no special code path.
- K fields or realisations with the same BC types but different data: $\mathbf{I}_K \otimes \mathbf{L}_b$, again an SpMM.
- Different BC types per realisation give different operators: no Kronecker structure.

With symmetries and repeated blocks

$$\mathbf{A} = \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}} + \mathbf{A}_{\text{out}}$$

- Only the geometry must be symmetric.
- Asymmetric boundary conditions go into $\mathbf{A}_{\text{out}}$.
- Inner–interface ordering: BCs need not repeat at all.

À. Alsalti-Baldellou et al., Comput. Fluids 275 (2024) 106247; SIAM J. Sci. Comput. 46(6) (2024) B860–B883

CFL time step from algebraic eigenvalue bounds

The question

Explicit convection–diffusion is stable if the eigenvalues of

$$\Delta t\, \Omega^{-1}\big(\mathbf{C}(\boldsymbol{u}) + \mathbf{D}\big)$$

lie in the stability region of the scheme.

F.X. Trias et al., “An efficient eigenvalue bounding method: CFL condition revisited”, Comput. Phys. Commun. 305 (2024) 109351

CFL time step from algebraic eigenvalue bounds

The question

Explicit convection–diffusion is stable if the eigenvalues of

$$\Delta t\, \Omega^{-1}\big(\mathbf{C}(\boldsymbol{u}) + \mathbf{D}\big)$$

lie in the stability region of the scheme.

The algebraic answer

- Bound the spectrum from the matrices themselves, not from a geometric CFL on cell sizes.
- Works on any mesh the operators are built on, unstructured included.
- Evaluated with the same kernels as the time step.

[TODO: add the bound of Trias et al. (2024) itself (formula) after checking the paper]

F.X. Trias et al., "An efficient eigenvalue bounding method: CFL condition revisited", Comput. Phys. Commun. 305 (2024) 109351

CFL time step from algebraic eigenvalue bounds

The question

Explicit convection–diffusion is stable if the eigenvalues of

$$\Delta t\, \Omega^{-1}\big(\mathbf{C}(\boldsymbol{u}) + \mathbf{D}\big)$$

lie in the stability region of the scheme.

The algebraic answer

- Bound the spectrum from the matrices themselves, not from a geometric CFL on cell sizes.
- Works on any mesh the operators are built on, unstructured included.
- Evaluated with the same kernels as the time step.

[TODO: add the bound of Trias et al. (2024) itself (formula) after checking the paper]

In the ensemble runs

The CPC ensemble simulations use this method (AlgEigCD) for adaptive Δt.

For K runs

With K realisations, $\mathbf{C}(\boldsymbol{u}_k)$ differs per realisation; one shared Δt means the minimum of K bounds, computed in one pass. [TODO: confirm how the CPC runs choose Δt across realisations]

F.X. Trias et al., "An efficient eigenvalue bounding method: CFL condition revisited", Comput. Phys. Commun. 305 (2024) 109351

Poisson solver: where structure pays twice

Symmetries

- Change of basis $\mathbf{P}_s$: 2^s decoupled systems.
- Fewer iterations: PCG/BiCGSTAB with SAI/ILU up to $4.3\times$ faster, AMG up to $1.3\times$ (separate solver calls, no SpMM).
- MUMPS becomes feasible at 256^3.

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133; SIAM J. Sci. Comput. 46(6) (2024) B860–B883; J. Plana-Riu et al., Comput. Phys. Commun. 326 (2026) 110232

Poisson solver: where structure pays twice

Symmetries

- Change of basis $\mathbf{P}_s$: 2^s decoupled systems.
- Fewer iterations: PCG/BiCGSTAB with SAI/ILU up to $4.3\times$ faster, AMG up to $1.3\times$ (separate solver calls, no SpMM).
- MUMPS becomes feasible at 256^3.

Repeated geometry

- AMGR: SpMM smoother on the inner blocks, 1.4–$1.7\times$ vs. AMG.

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133; SIAM J. Sci. Comput. 46(6) (2024) B860–B883; J. Plana-Riu et al., Comput. Phys. Commun. 326 (2026) 110232

Poisson solver: where structure pays twice

Symmetries

- Change of basis $\mathbf{P}_s$: 2^s decoupled systems.
- Fewer iterations: PCG/BiCGSTAB with SAI/ILU up to $4.3\times$ faster, AMG up to $1.3\times$ (separate solver calls, no SpMM).
- MUMPS becomes feasible at 256^3.

Repeated geometry

- AMGR: SpMM smoother on the inner blocks, 1.4–$1.7\times$ vs. AMG.

Many solutions

- K right-hand sides through one AMG hierarchy: every level is an SpMM.
- Coarse levels are denser: 6.9 nnz/row on the finest level, 102–181 on the coarsest (CPC App. A), so the intensity bound rises with depth.

Caveat

The JCP gains come from fewer iterations; we projected the SpMM gain on top, not yet measured end to end.

À. Alsalti-Baldellou et al., J. Comput. Phys. 486 (2023) 112133; SIAM J. Sci. Comput. 46(6) (2024) B860–B883; J. Plana-Riu et al., Comput. Phys. Commun. 326 (2026) 110232

Portability: one minimal kernel set

Kernels

- SpMV $\rightarrow$ SpMM, axpy, axty, dot: the whole time step.
- Hierarchical MPI + OpenMP/CUDA/OpenCL implementation.
- HPC^2 (hpc2lib): CPU, CUDA and OpenCL SpMM kernels in CSR and ELL.

X. Álvarez-Farré, A. Gorobets, F.X. Trias, Comput. Fluids 214 (2021) 104768; À. Alsalti-Baldellou et al., Comput. Fluids 275 (2024) 106247

Portability: one minimal kernel set

Kernels

- SpMV → SpMM, axpy, axty, dot: the whole time step.
- Hierarchical MPI + OpenMP/CUDA/OpenCL implementation.
- HPC^2 (hpc2lib): CPU, CUDA and OpenCL SpMM kernels in CSR and ELL.

Layout

The algebra is written subdomain-major, $\mathbf{I}_K \otimes \mathbf{A}$. The kernel stores the K vectors interleaved, $x[Kj + k]$, i.e. $\mathbf{A} \otimes \mathbf{I}_K$ after a perfect shuffle.

X. Álvarez-Farré, A. Gorobets, F.X. Trias, Comput. Fluids 214 (2021) 104768; À. Alsalti-Baldellou et al., Comput. Fluids 275 (2024) 106247

Portability: one minimal kernel set

Kernels

- SpMV → SpMM, axpy, axty, dot: the whole time step.
- Hierarchical MPI + OpenMP/CUDA/OpenCL implementation.
- HPC^2 (hpc2lib): CPU, CUDA and OpenCL SpMM kernels in CSR and ELL.

Layout

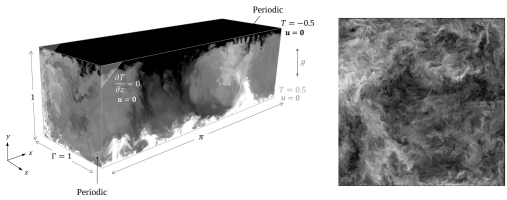
The algebra is written subdomain-major, $\mathbf{I}_K \otimes \mathbf{A}$. The kernel stores the K vectors interleaved, $x[Kj + k]$, i.e. $\mathbf{A} \otimes \mathbf{I}_K$ after a perfect shuffle.

GPU mapping

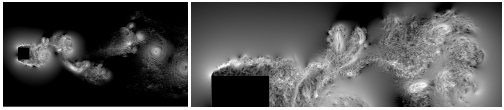
- One work-item per (row, vector): K neighbouring lanes share each coefficient.
- Coalesced accesses to $\boldsymbol{x}$ and $\boldsymbol{y}$.
- 2–$3\times$ faster than looping over K inside the thread, for $K = 8$.

X. Álvarez-Farré, A. Gorobets, F.X. Trias, Comput. Fluids 214 (2021) 104768; À. Alsalti-Baldellou et al., Comput. Fluids 275 (2024) 106247

Representative DNS with symmetry-preserving discretisations



$Ra = 10^{11}$: 5.7 billion grid points, 8192 CPU cores, MareNostrum 4



2.6 billion grid points, 3136 CPU cores, MareNostrum 5-GPP

These are the production targets: operators shared by velocity components, scalars and ensembles, on meshes where symmetries or repeated blocks exist.

F. Dabbagh et al., Phys. Rev. Fluids 5 (2020) 024603; F.X. Trias, À. Alsalti-Baldellou, A. Oliva, Phys. Fluids 38(4) (2026) 045157

Outlook

Combine the sources of K

$$\mathbf{I}_{K_{\text{ens}}} \otimes \mathbf{I}_3 \otimes \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}} \ (+ \text{ outer couplings})$$

Symmetries $\times$ components $\times$ ensembles multiply: $K = 8 \cdot 3 \cdot 4 = 96$ is not exotic. The vector traffic ($16K$ B/row) then dominates.

Outlook

Combine the sources of K

$$\mathbf{I}_{K_{\text{ens}}} \otimes \mathbf{I}_3 \otimes \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}} \ (+ \text{ outer couplings})$$

Symmetries $\times$ components $\times$ ensembles multiply: $K = 8 \cdot 3 \cdot 4 = 96$ is not exotic. The vector traffic ($16K$ B/row) then dominates.

SpMM inside the solver

- Multiple-RHS AMG: coarse levels at 102–181 nnz/row (CPC App. A), where SpMM gains most.
- Block Krylov for the ensemble Poisson solves (cf. X. Allka, 08:30).

Outlook

Combine the sources of K

$$\mathbf{I}_{K_{\text{ens}}} \otimes \mathbf{I}_3 \otimes \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}} \ (+ \text{ outer couplings})$$

Symmetries $\times$ components $\times$ ensembles multiply: $K = 8 \cdot 3 \cdot 4 = 96$ is not exotic. The vector traffic ($16K$ B/row) then dominates.

SpMM inside the solver

- Multiple-RHS AMG: coarse levels at 102–181 nnz/row (CPC App. A), where SpMM gains most.
- Block Krylov for the ensemble Poisson solves (cf. X. Allka, 08:30).

Hardware

- GPU runs of the CPC ensembles (so far CPU only).
- AMGR on GPU (future work in SISC).

Outlook

Combine the sources of K

$$\mathbf{I}_{K_{\text{ens}}} \otimes \mathbf{I}_3 \otimes \mathbf{I}_{2^s} \otimes \mathbf{A}_{\text{inn}} \ (+ \text{ outer couplings})$$

Symmetries $\times$ components $\times$ ensembles multiply: $K = 8 \cdot 3 \cdot 4 = 96$ is not exotic. The vector traffic ($16K$ B/row) then dominates.

SpMM inside the solver

- Multiple-RHS AMG: coarse levels at 102–181 nnz/row (CPC App. A), where SpMM gains most.
- Block Krylov for the ensemble Poisson solves (cf. X. Allka, 08:30).

Hardware

- GPU runs of the CPC ensembles (so far CPU only).
- AMGR on GPU (future work in SISC).

Time

- Time-parallel all-at-once formulations give $K = T$ time levels on the same operator; Para-Diag keeps the Kronecker form with complex diagonal factors.
- Not yet explored.