Validating the roofplane performance model for parallel CFD across HPC architectures

Strong scaling of HPL, HPCG and TFA+HPC^2 on MareNostrum 5 GPP and JFF2

M. Mosqueda-Otero[1], X. Álvarez-Farré[2], À. Alsalti-Baldellou[1,3], A. Gorobets[4], F. X. Trias[1], A. Oliva[1]

[1] Heat and Mass Transfer Technological Center (CTTC), Universitat Politècnica de Catalunya, Spain
[2] High-Performance Computing Team, SURF, The Netherlands
[3] Department of Civil, Environmental and Architectural Engineering, University of Padova, Italy
[4] Keldysh Institute of Applied Mathematics, Russia

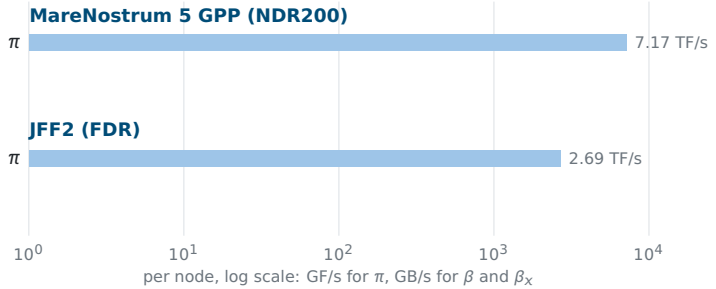
ParCFD 2026, 37th International Conference on Parallel Computational Fluid Dynamics
Nantes, France, 3–6 November 2026

Centre Tecnològic de Transferència de Calor
Universitat Politècnica de Catalunya

1. Introduction

Bandwidths lag behind the peak

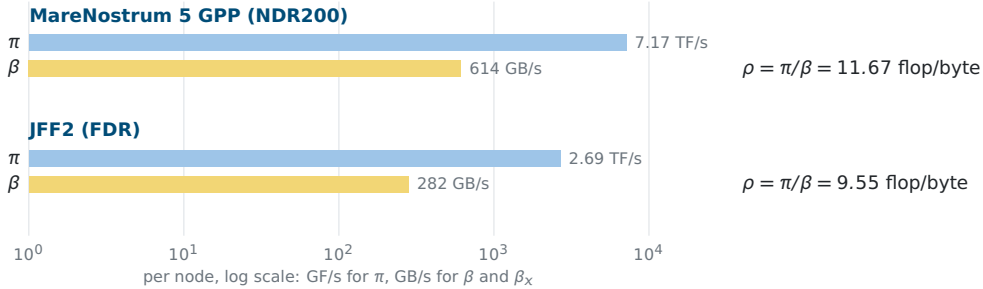
Peak throughput has outgrown memory and network bandwidth for decades[1]:



[1] McCalpin (2016): *"Memory Bandwidth and System Balance in HPC Systems"*

Bandwidths lag behind the peak

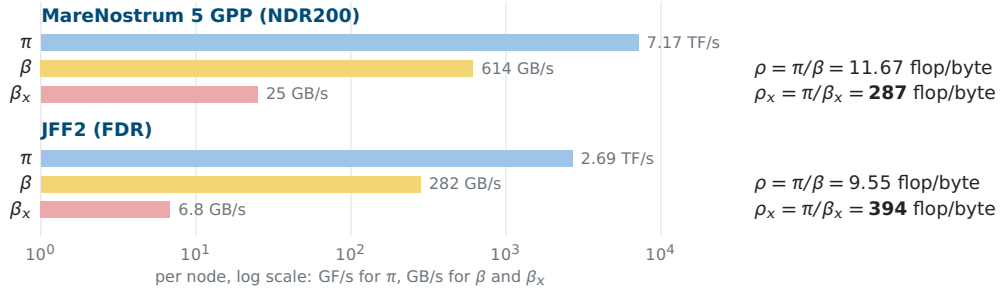
Peak throughput has outgrown memory and network bandwidth for decades[1]:



[1] McCalpin (2016): *"Memory Bandwidth and System Balance in HPC Systems"*

Bandwidths lag behind the peak

Peak throughput has outgrown memory and network bandwidth for decades[1]:



The balance of a node

To compute at its peak, a node of MN5 needs 11.67 flop per byte read from memory and **287 flop per byte received from the network**.

[1] McCalpin (2016): *"Memory Bandwidth and System Balance in HPC Systems"*

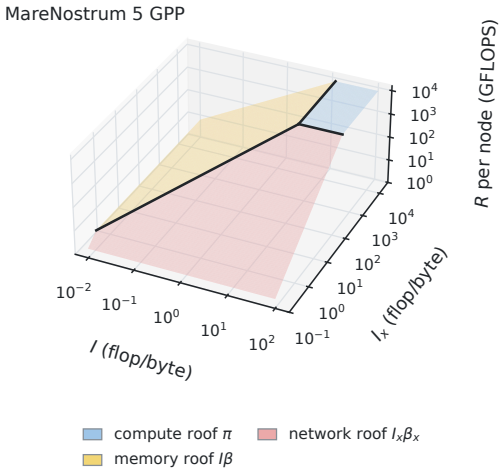
From the roofline to the roofplane

The roofline bounds a device by its peak and its memory bandwidth[2]; the roofplane adds the network as a third roof[3]:

$$R^{RP} = \min(\pi,\ I\beta,\ I_x\beta_x)$$

$$I = \frac{W}{Q}, \qquad I_x = \frac{W}{Q_x}$$

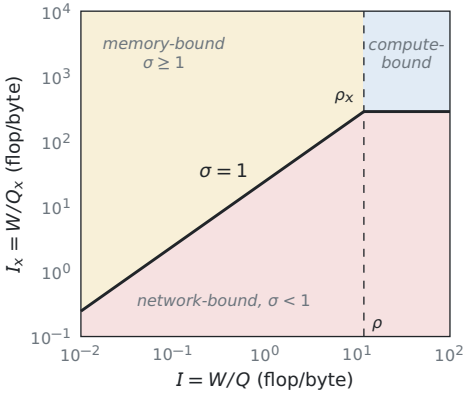
- W: floating-point work
- Q: bytes moved from main memory
- Q_x: bytes received from other nodes, one direction of the full-duplex NIC; per node, the mean over the nodes of the job



[2] Williams et al. (2009): *"Roofline: An Insightful Visual Performance Model for Multicore Architectures"*

[3] Alvarez-Farré et al. (2025): *"Roofplane: A Visual Performance Model for High-Performance Distributed Systems"*

The parallel potential σ



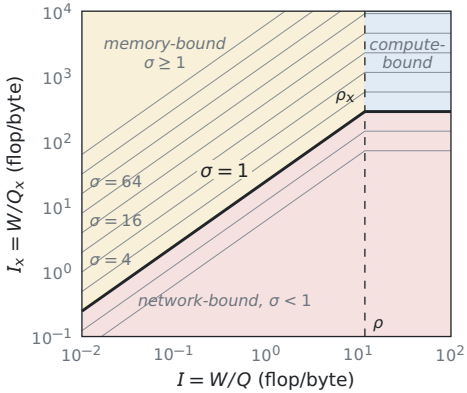
$$\sigma = \max\left(1, \frac{\rho}{I}\right)\frac{I_x}{\rho_x}, \qquad \rho = \frac{\pi}{\beta}, \quad \rho_x = \frac{\pi}{\beta_x}$$

- $\sigma \geq 1$: the network can follow the code
- $\sigma < 1$: network-bound, whatever the nodes added

Hypotheses[4]: (1)–(3) independent operations, memory requests and large non-blocking messages, at π, β and β_x; and

[4] Álvarez-Farré et al. (2025): *"Roofplane: A Visual Performance Model for High-Performance Distributed Systems"*

The parallel potential σ



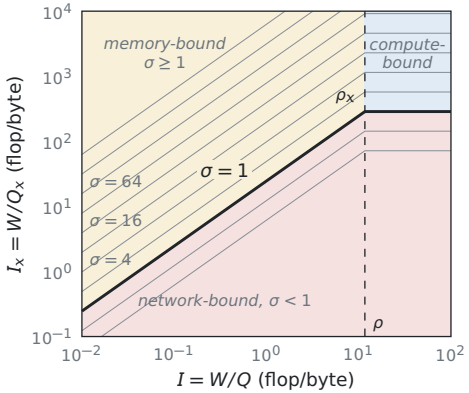
$$\sigma = \max\left(1, \frac{\rho}{I}\right)\frac{I_x}{\rho_x}, \qquad \rho = \frac{\pi}{\beta}, \quad \rho_x = \frac{\pi}{\beta_x}$$

- $\sigma \geq 1$: the network can follow the code
- $\sigma < 1$: network-bound, whatever the nodes added

Hypotheses[4]: (1)–(3) independent operations, memory requests and large non-blocking messages, at π, β and β_x; and

[4] Álvarez-Farré et al. (2025): *"Roofplane: A Visual Performance Model for High-Performance Distributed Systems"*

The parallel potential σ



$$\sigma = \max\left(1, \frac{\rho}{I}\right)\frac{I_x}{\rho_x}, \qquad \rho = \frac{\pi}{\beta}, \quad \rho_x = \frac{\pi}{\beta_x}$$

- $\sigma \geq 1$: the network can follow the code
- $\sigma < 1$: network-bound, whatever the nodes added

Hypotheses[4]: (1)–(3) independent operations, memory requests and large non-blocking messages, at π, β and β_x; and

⚠ (4) Overlap

The exchanges run behind the bulk computation.

[4] Álvarez-Farré et al. (2025): *"Roofplane: A Visual Performance Model for High-Performance Distributed Systems"*

Research questions

RQ1

Does σ predict the strong-scaling efficiency of codes whose behaviour is known, HPL and HPCG?

Research questions

RQ1

Does σ predict the strong-scaling efficiency of codes whose behaviour is known, HPL and HPCG?

RQ2

Where does a CFD Poisson solver, TFA+HPC^2, sit against HPCG at the same unknowns per rank, and why?

Research questions

RQ1

Does σ predict the strong-scaling efficiency of codes whose behaviour is known, HPL and HPCG?

RQ2

Where does a CFD Poisson solver, TFA+HPC^2, sit against HPCG at the same unknowns per rank, and why?

RQ3

Can W, Q and Q_x be measured consistently across codes, machines and MPI libraries?

2. Setup

Workloads and machines

Three workloads that span the three roofs, HPL[5], HPCG[6] and the pressure CG of the CFD code[7] TFA+HPC^2; MPI-only, full nodes, strong scaling from 2 nodes:

	MN5 GPP 2 × SPR 8480+ NDR200	**JFF2** 2 × Gold 6230 FDR	**Deucalion** A64FX HBM2	**Snellius** 2 × Genoa 9654 NDR
TFA+HPC^2	525 k cells per rank 2–128 nodes	525 k cells per rank 2–16 nodes	in progress	in progress
HPCG	525 k and 2.1 M unknowns per rank 2–128 nodes	525 k and 2.1 M unknowns per rank 2–16 nodes	in progress	in progress
HPL	$N = 212\,480$ 2–128 nodes	$N = 129\,280$ 2–16 nodes	in progress	in progress

[5] Dongarra et al. (2003): *"The LINPACK Benchmark: past, present and future"*
[6] Dongarra et al. (2016): *"High-Performance Conjugate-Gradient Benchmark: A New Metric for Ranking High-Performance Computing Systems"*
[7] Álvarez-Farré et al. (2018): *"HPC^2 – A fully portable algebra-based framework for heterogeneous computing. Application to CFD"*

Roofs: nominal and measured

		π (TF/s)	β (GB/s)	β_x (GB/s)	ρ	ρ_x
MN5 GPP	nominal	7.17	614.4	25.0	11.67	287
	measured	4.74 (66 %)	399.1 (65 %)	23.65 (95 %)		
JFF2	nominal	2.69	281.6	6.82	9.55	394
	measured	1.49 (55 %)	162.1 (58 %)	4.65 (68 %)		

Measured with HPL (π), STREAM Triad[8] (β) and IMB Biband/2 (β_x, one direction with both directions loaded), per node. Two balances of the network: NDR200 against FDR.

The roofs are nominal

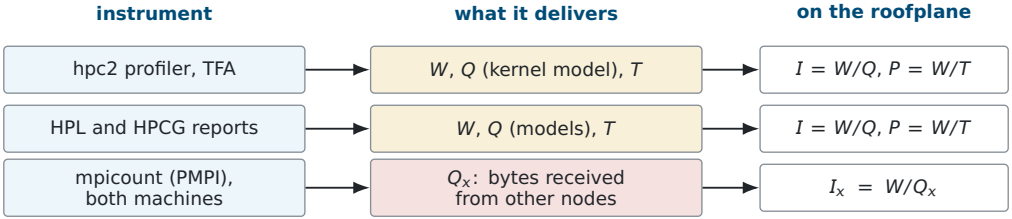
The measured rates give the efficiency of each roof, not the roof.

[8] McCalpin (1995): *"Memory Bandwidth and Machine Balance in Current High Performance Computers"*

3. Measurement

From the runs to the roofplane

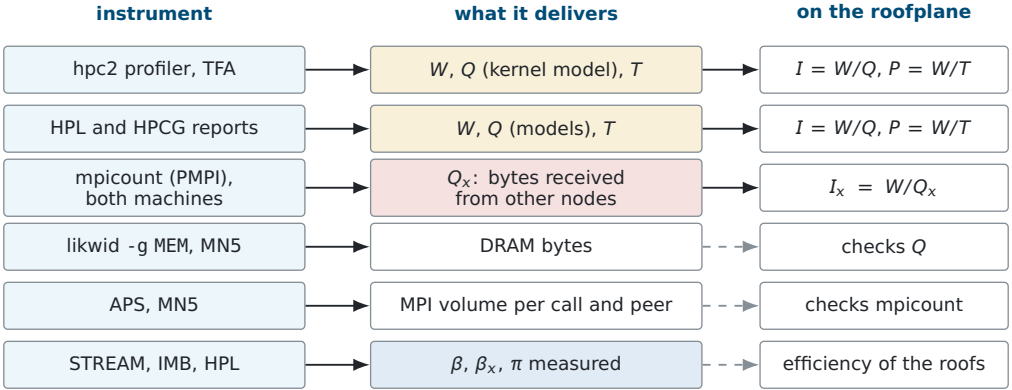
Every point holds W, Q and Q_x of one solver window over its time T; one PMPI counter gives Q_x on any MPI library, and likwid[9] the DRAM bytes of MN5:



[9] Treibig et al. (2010): *"LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments"*

From the runs to the roofplane

Every point holds W, Q and Q_x of one solver window over its time T; one PMPI counter gives Q_x on any MPI library, and likwid[9] the DRAM bytes of MN5:



[9] Treibig et al. (2010): *"LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments"*

Models of W, Q and Q_X

HPL

W $\frac{2}{3}N^3 + \frac{3}{2}N^2$ (HPL count)[10]

Q $16N^3/(3N_B)$ (right-looking LU)[11], so $I \leq N_B/8$

Q_X byte model of its communication operations[10],[12]

✔ Q_X: 1.004–1.020 of the model

[10] Dongarra et al. (2003): *"The LINPACK Benchmark: past, present and future"*

[11] Toledo (1997): *"Locality of reference in LU decomposition with partial pivoting"*

[12] Petitet et al. (2018): *HPL – A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*

Models of W, Q and Q_X

HPL

W $\frac{2}{3}N^3 + \frac{3}{2}N^2$ (HPL count)[10]

Q $16N^3/(3N_B)$ (right-looking LU)[11], so $I \leq N_B/8$

Q_X byte model of its communication operations[10],[12]

✔ Q_X: 1.004–1.020 of the model

HPCG

W flop model of the report[13]

Q byte model of the report

Q_X halo model, from partitions and HPCG 3.1 source

per CG set; Q_X by difference of two run times

✔ Q_X: 1.000000 of the model at every size

10 Dongarra et al. (2003): *"The LINPACK Benchmark: past, present and future"*

11 Toledo (1997): *"Locality of reference in LU decomposition with partial pivoting"*

12 Petitet et al. (2018): *HPL – A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*

13 Dongarra et al. (2016): *"High-Performance Conjugate-Gradient Benchmark: A New Metric for Ranking High-Performance Computing Systems"*

Models of W, Q and Q_X

HPL

W $\frac{2}{3}N^3 + \frac{3}{2}N^2$ (HPL count)[10]

Q $16N^3/(3N_B)$ (right-looking LU)[11], so $I \leq N_B/8$

Q_X byte model of its communication operations[10].,[12]

✔ Q_X: 1.004–1.020 of the model

HPCG

W flop model of the report[13]

Q byte model of the report

Q_X halo model, from partitions and HPCG 3.1 source

per CG set; Q_X by difference of two run times

✔ Q_X: 1.000000 of the model at every size

TFA+HPC^2

W $\sum_{k \in K} \text{GFLOPS}_k \, T_k$

Q $\sum_{k \in K} \text{mem}_k \, T_k$

Q_X $\frac{1}{2} \sum_{k \in K} \text{inter}_k \, T_k$

$K = \{\text{SpMV, axpy, axty, dot}\}$, hpc2 kernel model; $T = \sum_K T_k$, $P = W/T$, $I = 35/256 = 0.137$

✔ Q_X: equal to the halo map of hpc2, byte by byte

Q_X counted with mpicount (PMPI) on both machines, each against an independent model.

10 Dongarra et al. (2003): *"The LINPACK Benchmark: past, present and future"*

11 Toledo (1997): *"Locality of reference in LU decomposition with partial pivoting"*

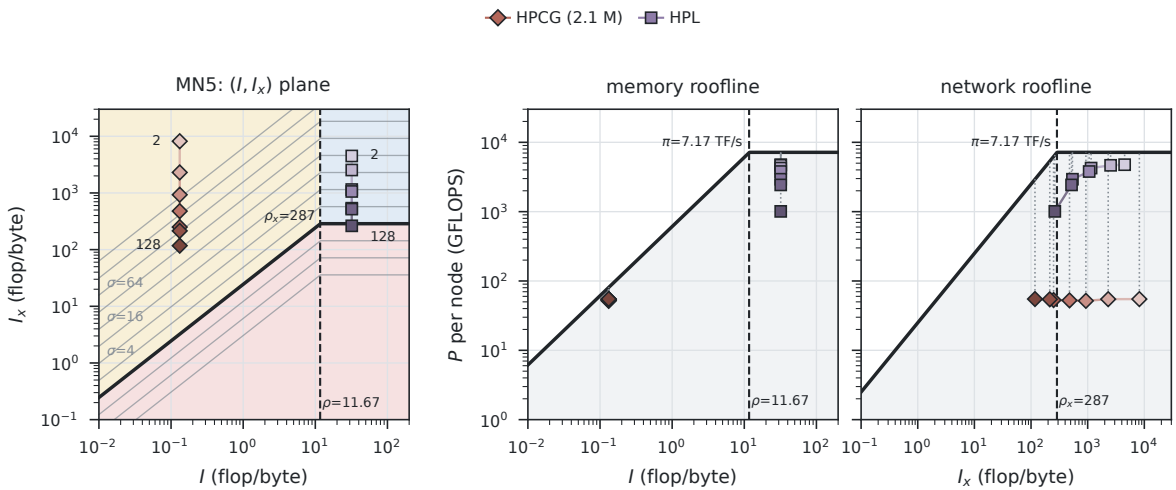
12 Petitet et al. (2018): *HPL – A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*

13 Dongarra et al. (2016): *"High-Performance Conjugate-Gradient Benchmark: A New Metric for Ranking High-Performance Computing Systems"*

4. Benchmarks

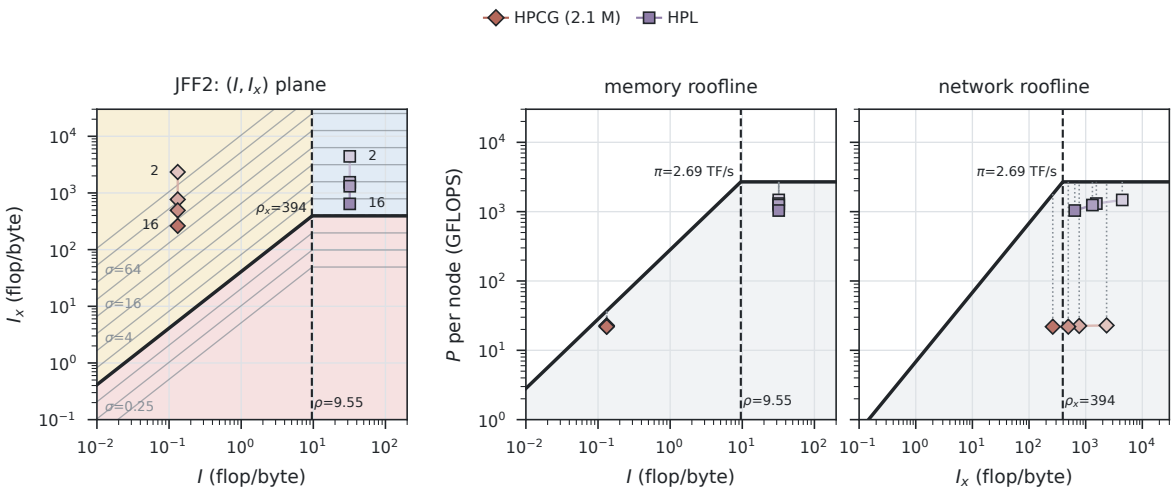
The benchmarks on the roofplane

HPCG at 2.1 M unknowns per rank on 2 nodes, HPL with $N = 212\,480$ on MN5 and 129 280 on JFF2



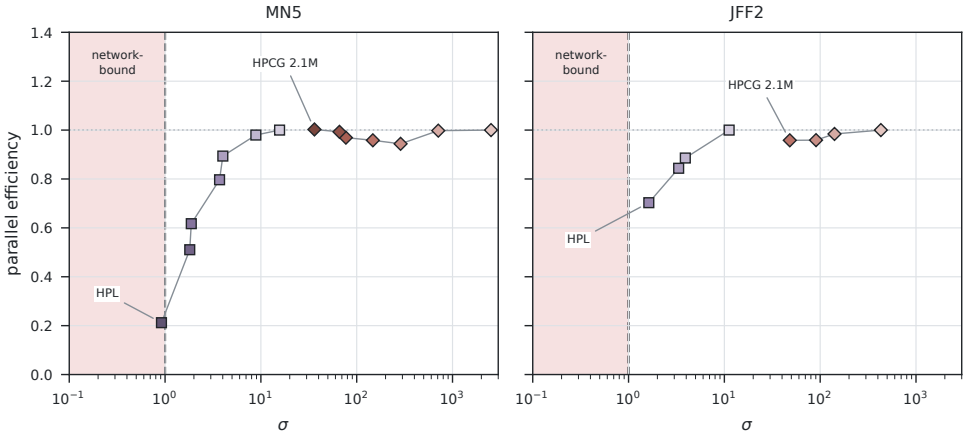
The benchmarks on the roofplane

HPCG at 2.1 M unknowns per rank on 2 nodes, HPL with $N = 212\,480$ on MN5 and 129 280 on JFF2



σ against the measured efficiency

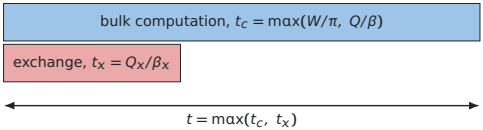
HPCG at 2.1 M unknowns per rank, HPL



HPCG keeps $\sigma \geq 36$ and 94–100 %. HPL falls to 62 % and 51 % at 32 and 64 nodes with $\sigma = 1.88$ and 1.81, and crosses $\sigma = 1$ only at 128 nodes (21 %).

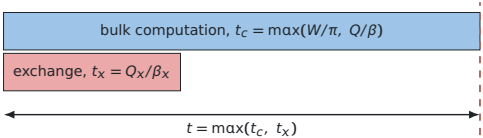
Why the efficiency falls before $\sigma = 1$

Roofplane, hypothesis 4: the exchange is hidden

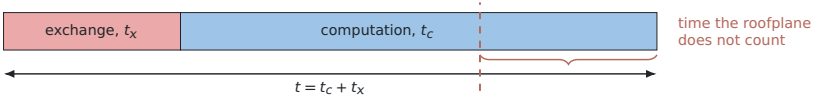


Why the efficiency falls before $\sigma = 1$

Roofplane, hypothesis 4: the exchange is hidden



Our runs (MPI-only, HPL with DEPTH 0): one after the other



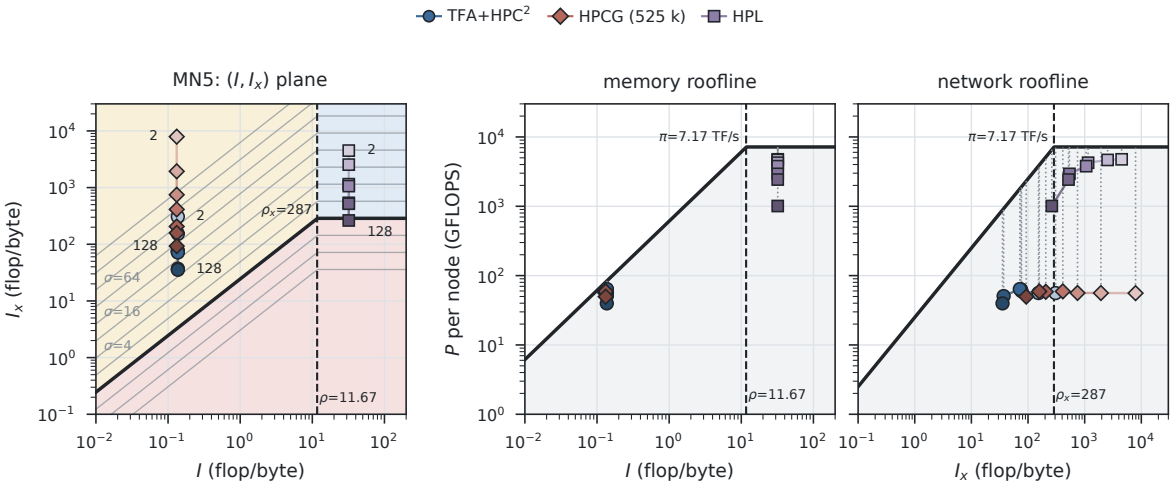
⚠ No series overlaps

Every run is MPI-only, and HPL runs with DEPTH 0, without look-ahead: the exchange adds to the computation. σ is an upper bound that only an overlapped run can reach.

5. CFD

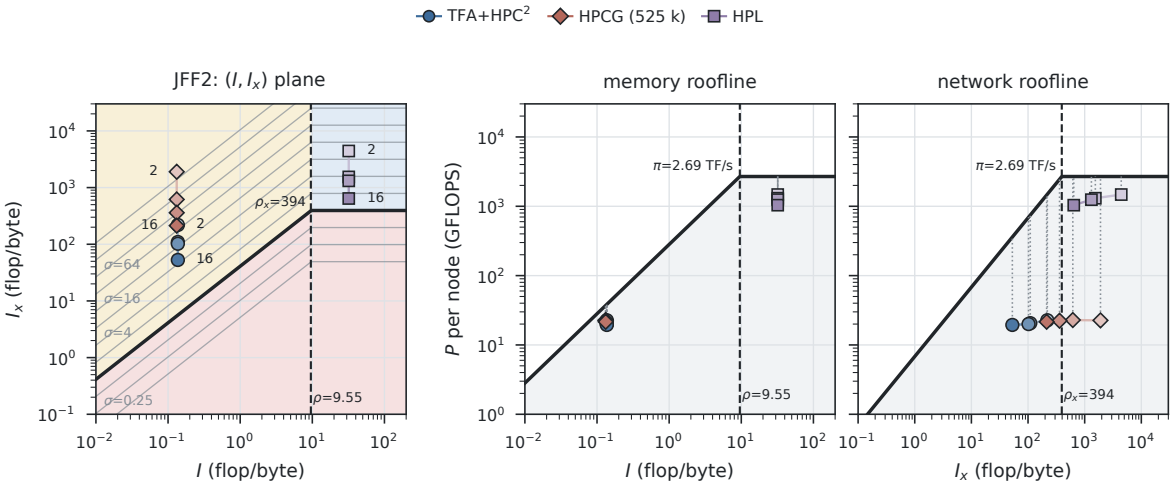
TFA+HPC^2 against HPCG at the same load

525 k unknowns per rank on 2 nodes, the same at every node count; HPL as reference



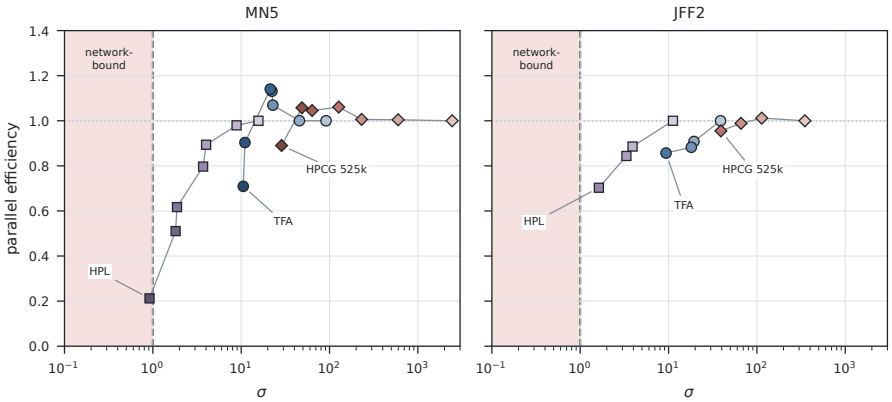
TFA+HPC^2 against HPCG at the same load

525 k unknowns per rank on 2 nodes, the same at every node count; HPL as reference



σ against the efficiency

TFA+HPC^2 and HPCG at 525 k unknowns per rank, HPL



Neither solver is network-bound: TFA+HPC^2 keeps $\sigma \geq 9.4$, HPCG $\sigma \geq 28.6$. TFA+HPC^2 loses efficiency from 64 nodes on MN5 (90 %, 71 %) mostly in the Allreduce of its dot products: latency, which σ, a ratio of bytes, does not see.

6. Conclusions

Conclusions

- **RQ1.** σ predicts the direction: HPCG keeps $\sigma \geq 36$ and 94–100 %; HPL loses efficiency as σ approaches 1, earlier than the model, since no run overlaps its exchanges.

Conclusions

- **RQ1.** σ predicts the direction: HPCG keeps $\sigma \geq 36$ and 94–100 %; HPL loses efficiency as σ approaches 1, earlier than the model, since no run overlaps its exchanges.
- **RQ2.** TFA+HPC^2 and HPCG are memory-bound Poisson solvers far from the network roof on both machines; the losses of TFA+HPC^2 come from latency and the caches, which a model of bytes does not capture.

Conclusions

- **RQ1.** σ predicts the direction: HPCG keeps $\sigma \geq 36$ and 94–100 %; HPL loses efficiency as σ approaches 1, earlier than the model, since no run overlaps its exchanges.
- **RQ2.** TFA+HPC^2 and HPCG are memory-bound Poisson solvers far from the network roof on both machines; the losses of TFA+HPC^2 come from latency and the caches, which a model of bytes does not capture.
- **RQ3.** One PMPI counter delivers Q_x exactly for every code and MPI library; likwid checks Q on MN5.

Conclusions

- **RQ1.** σ predicts the direction: HPCG keeps $\sigma \geq 36$ and 94–100 %; HPL loses efficiency as σ approaches 1, earlier than the model, since no run overlaps its exchanges.
- **RQ2.** TFA+HPC^2 and HPCG are memory-bound Poisson solvers far from the network roof on both machines; the losses of TFA+HPC^2 come from latency and the caches, which a model of bytes does not capture.
- **RQ3.** One PMPI counter delivers Q_x exactly for every code and MPI library; likwid checks Q on MN5.

Take-away

The roofplane bounds the scaling; the overlap of the exchanges decides how close a code gets to the bound.

Future work

- Overlapped runs: HPL with DEPTH 1, and TFA+HPC^2 with MPI+OpenMP and a communication thread.
- The same three workloads on A64FX (Deucalion) and Genoa (Snellius).
- A quantitative relation between σ and the efficiency: latency and the system overhead χ of the roofplane.

Thank you

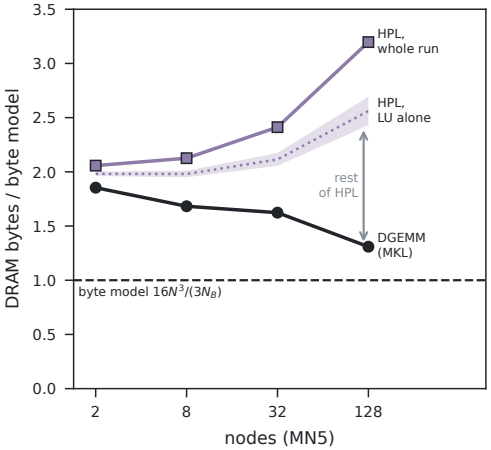
M. Mosqueda-Otero, marcial.francisco.mosqueda@upc.edu

M. Mosqueda-Otero and F. X. Trias are supported by the Ministerio de Ciencia, Innovación y Universidades, Spain, SIMEX project (PID2022-142174OB-I00). M. Mosqueda-Otero is also supported by the Catalan Agency for Management of University and Research Grants (2024 FI-1 00684). Computing time on MareNostrum 5 was granted by BSC, and on the JFF2 cluster by the Heat and Mass Transfer Technological Center (CTTC), UPC.

7. Backup

HPL moves twice its byte model

- likwid sees 2.06 times $16N^3/(3N_B)$ at 2 nodes and 3.20 at 128
- The DGEMM of MKL alone, with the local shapes of HPL, moves 1.85 times its share at 2 nodes and 1.31 at 128: it re-reads the packed panel of U per block of rows[14]
- The factor 2 is the library; its growth with the nodes, the rest of HPL

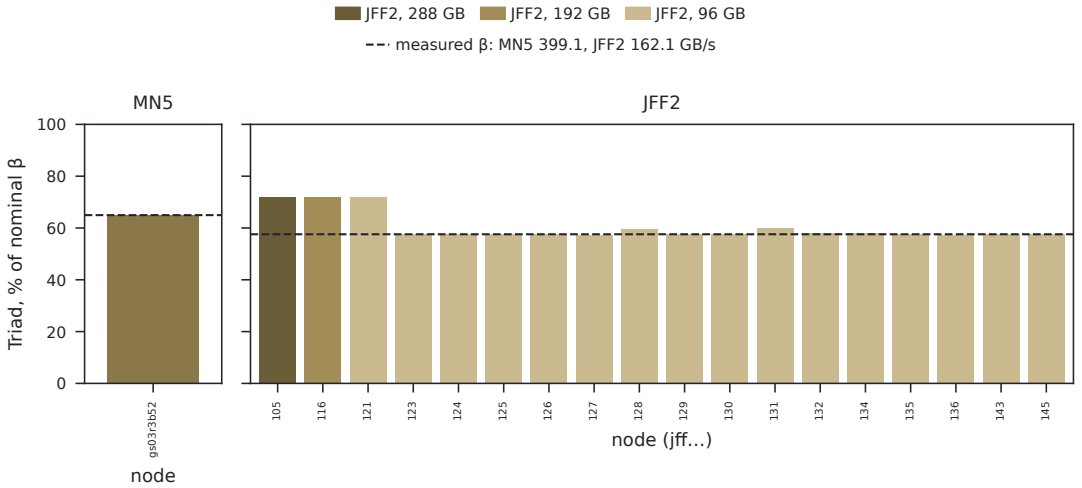


[14] Goto et al. (2008): *"Anatomy of high-performance matrix multiplication"*

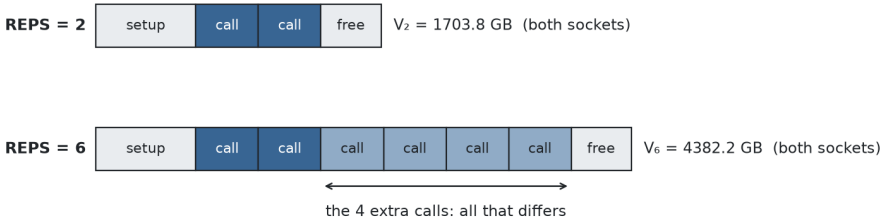
HPCG: Q_x and the placement of the ranks

- Block placement, as TFA+HPC^2 and HPL: each node holds 112 consecutive ranks in the x-fastest order of HPCG, slabs and stepped planes rather than boxes.
- From 16 nodes on, Q_x is 1.5–2.2 times what a box per node would give; at 128 nodes 41.6 % of the halo crosses the network.
- The model of Q_x follows the grid and the placement exactly: ratio 1.000000 at every size, both machines.
- At the same load the network share of the halo still differs: 1.9 % for HPCG at 2 nodes against 19 % for TFA+HPC^2, a property of how each code is partitioned.

STREAM on every node



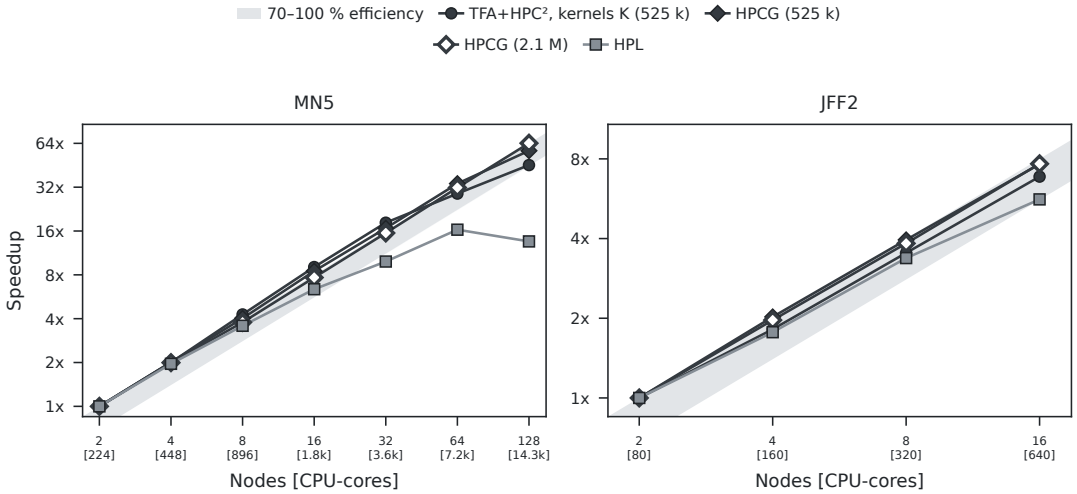
DRAM bytes by difference



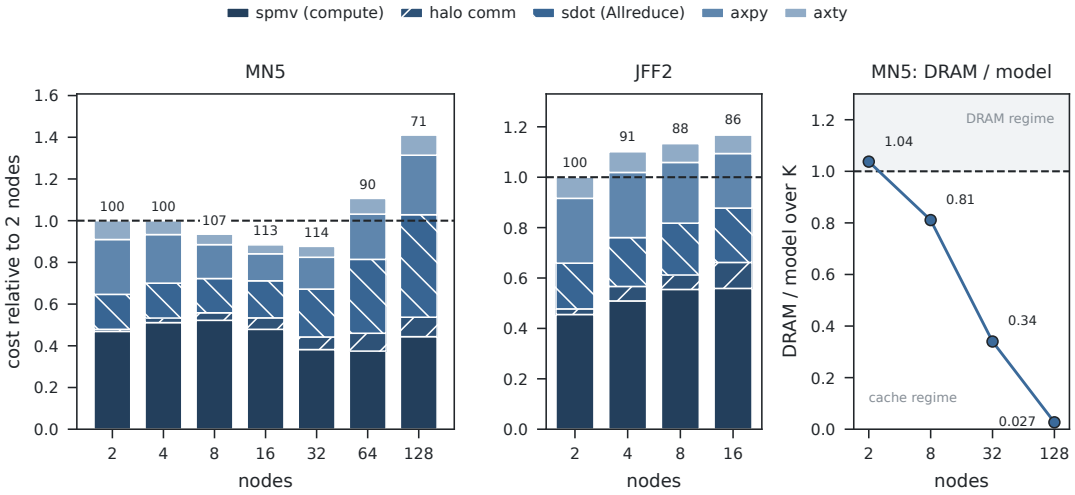
$$\text{DRAM per call} = \frac{V_6 - V_2}{(6-2)\cdot 112} = \frac{2678.4\ \text{GB}}{448} = 5.98\ \text{GB}$$

likwid `-g MEM` wraps one rank per socket; two runs that differ only in the work (2 and 6 time steps for TFA+HPC^2, rt 60 and 300 s for HPCG) cancel the setup, the input and the output.

Speed-up of every series



TFA+HPC^2: cost per part of K



Why Q_x counts one direction

IMB, two full nodes	MN5 GPP	JFF2
Uniband (GB/s)	23.7	5.54
Biband (GB/s)	47.3	9.3
Biband / Uniband	**2.00**	**1.68**
Biband / 2 (GB/s)	23.65	4.65

- The NIC carries both directions at once: the nominal β_x is a one-way rate.
- The codes keep both directions in flight: hpc2 posts every `MPI_Irecv` and `MPI_Isend` before its `MPI_Waitall`; HPCG posts its `MPI_Irecv` before its `MPI_Send`.
- The roofplane takes the halo a process receives through the injection bandwidth[15].
- With both directions loaded, one direction gets Biband/2: the efficiency of β_x.

⚠ Sent + received against the one-way β_x

It would count the demand twice and halve every I_x and σ.

[15] Álvarez-Farré et al. (2025): *"Roofplane: A Visual Performance Model for High-Performance Distributed Systems"*

Sent against received, node by node

		nodes	worst node: max(sent, received) / received
HPL	MN5 GPP	2–128	1.0012
	JFF2	2–16	1.0017
HPCG	MN5 GPP	2–128	1.0000
	JFF2	2–16	1.0000
TFA+HPC^2	MN5 GPP, JFF2	2	1.0000

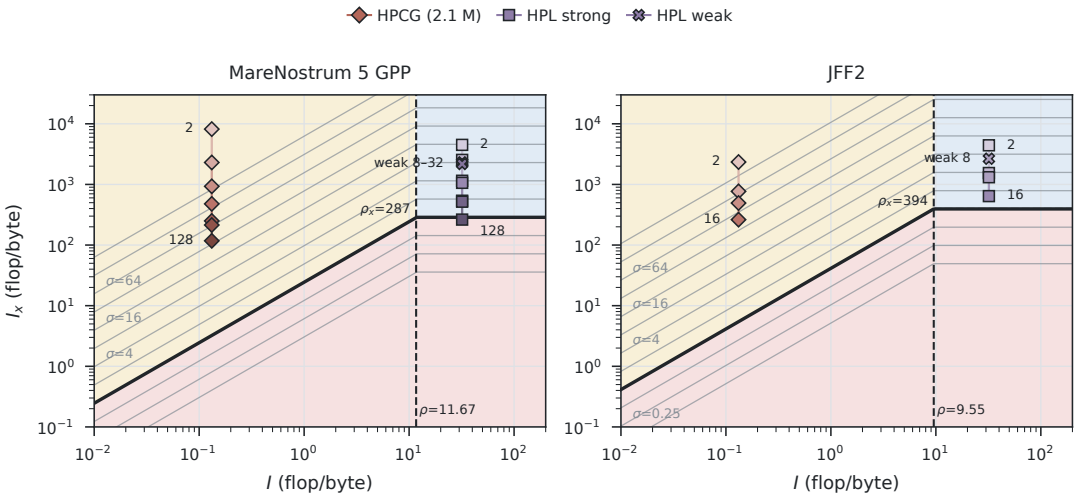
mpicount, inter-node point-to-point bytes of every rank, grouped by node. TFA: the pair of 2 and 6 time steps; its halo map gives inter_out = inter_in on every node of every run.

Every node receives what it sends

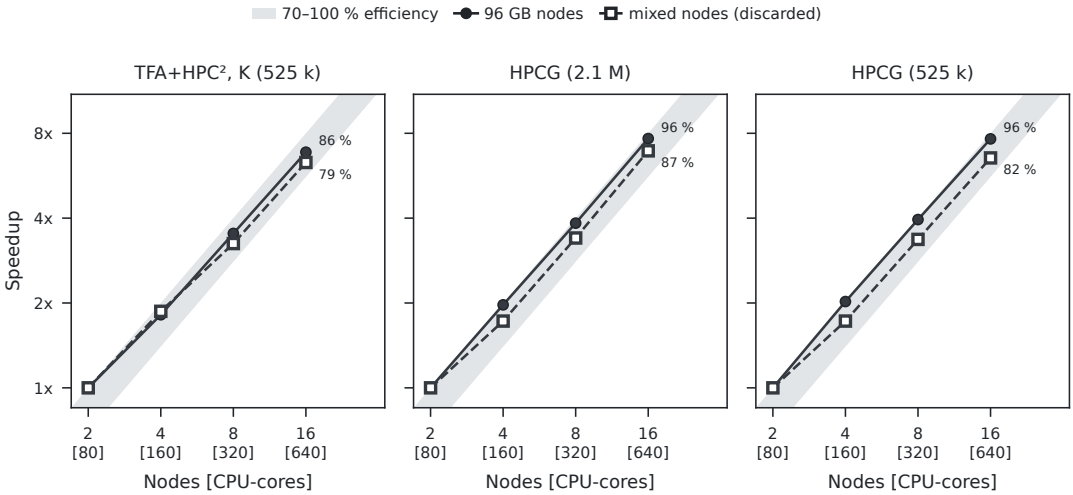
Q_x of one direction holds node by node, not only over the job: in HPL the process row that spreads U rotates with every panel, and the halos of HPCG and TFA+HPC^2 are symmetric.

HPL in weak scaling

The usual way to run HPL: N grows with $\sqrt{\text{nodes}}$, 70 % of the node memory



JFF2: the slowest node sets the pace



References

Álvarez-Farré, Xavier et al. (2025). “Roofplane: A Visual Performance Model for High-Performance Distributed Systems”. Manuscript in preparation.

Álvarez-Farré, Xavier et al. (2018). “HPC^2 – A fully portable algebra-based framework for heterogeneous computing. Application to CFD”. In: *Computers & Fluids* 173, pp. 285–292.

Dongarra, Jack, Michael A. Heroux, and Piotr Luszczek (2016). “High-Performance Conjugate-Gradient Benchmark: A New Metric for Ranking High-Performance Computing Systems”. In: *The International Journal of High Performance Computing Applications* 30.1, pp. 3–10.

Dongarra, Jack J., Piotr Luszczek, and Antoine Petitet (2003). “The LINPACK Benchmark: past, present and future”. In: *Concurrency and Computation: Practice and Experience* 15.9, pp. 803–820.

Goto, Kazushige and Robert A. van de Geijn (2008). “Anatomy of high-performance matrix multiplication”. In: *ACM Transactions on Mathematical Software* 34.3, 12:1–12:25.

McCalpin, John D. (Dec. 1995). “Memory Bandwidth and Machine Balance in Current High Performance Computers”. In: *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25.

— (2016). “Memory Bandwidth and System Balance in HPC Systems”. In: *Invited talk, SC16: The International Conference for High Performance Computing, Networking, Storage and Analysis*.

Petitet, A. et al. (2018). *HPL – A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers*. `https://www.netlib.org/benchmark/hpl/`.

Toledo, Sivan (1997). “Locality of reference in LU decomposition with partial pivoting”. In: *SIAM Journal on Matrix Analysis and Applications* 18.4, pp. 1065–1081.

Treibig, Jan, Georg Hager, and Gerhard Wellein (2010). “LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments”. In: *39th International Conference on Parallel Processing Workshops (ICPPW)*, pp. 207–216.

Williams, Samuel, Andrew Waterman, and David Patterson (2009). “Roofline: An Insightful Visual Performance Model for Multicore Architectures”. In: *Communications of the ACM* 52.4, pp. 65–76.