NUMA–aware strategies for the efficient execution of CFD simulations on CPU supercomputers

---

Xavier Álvarez Farré[1], Andrey Gorobets[2], F. Xavier Trias[1] and Assensi Oliva[1]

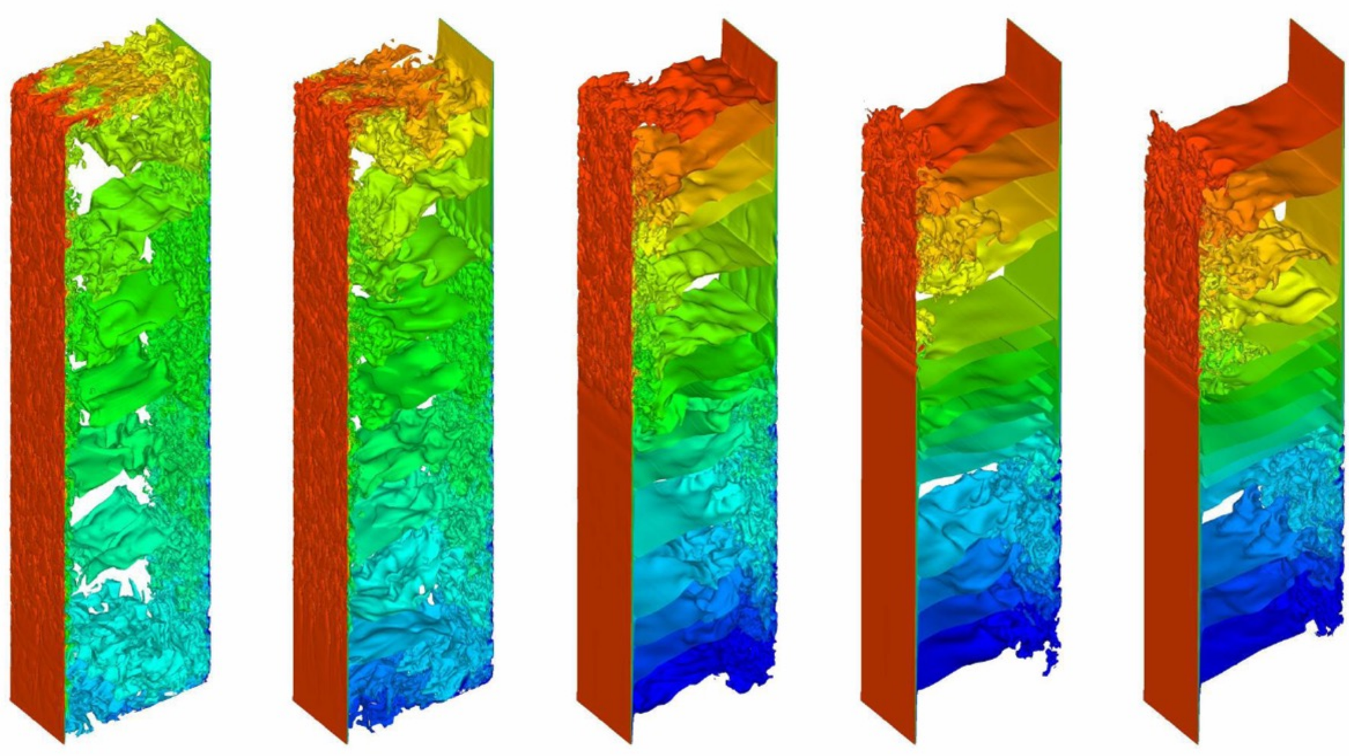
In the 32nd International Conference on Parallel Computational Fluid Dynamics, ParCFD'2020,
May 17—19, virtual conference hosted in Nice, France

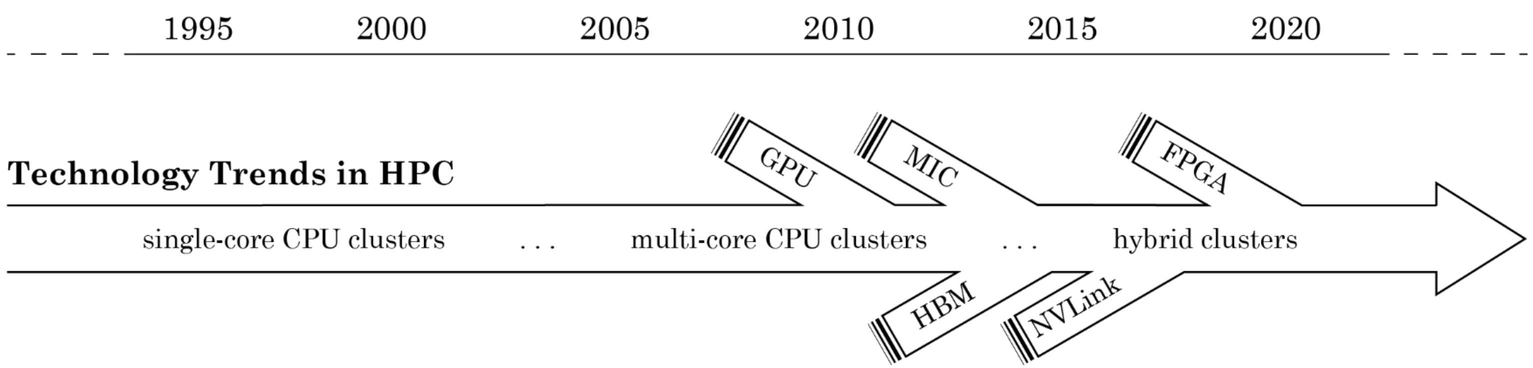
[1] Heat and Mass Transfer Technological Center, **Technical University of Catalonia**
[2] Keldysh Institute of Applied Mathematics, **Russian Academy of Science**

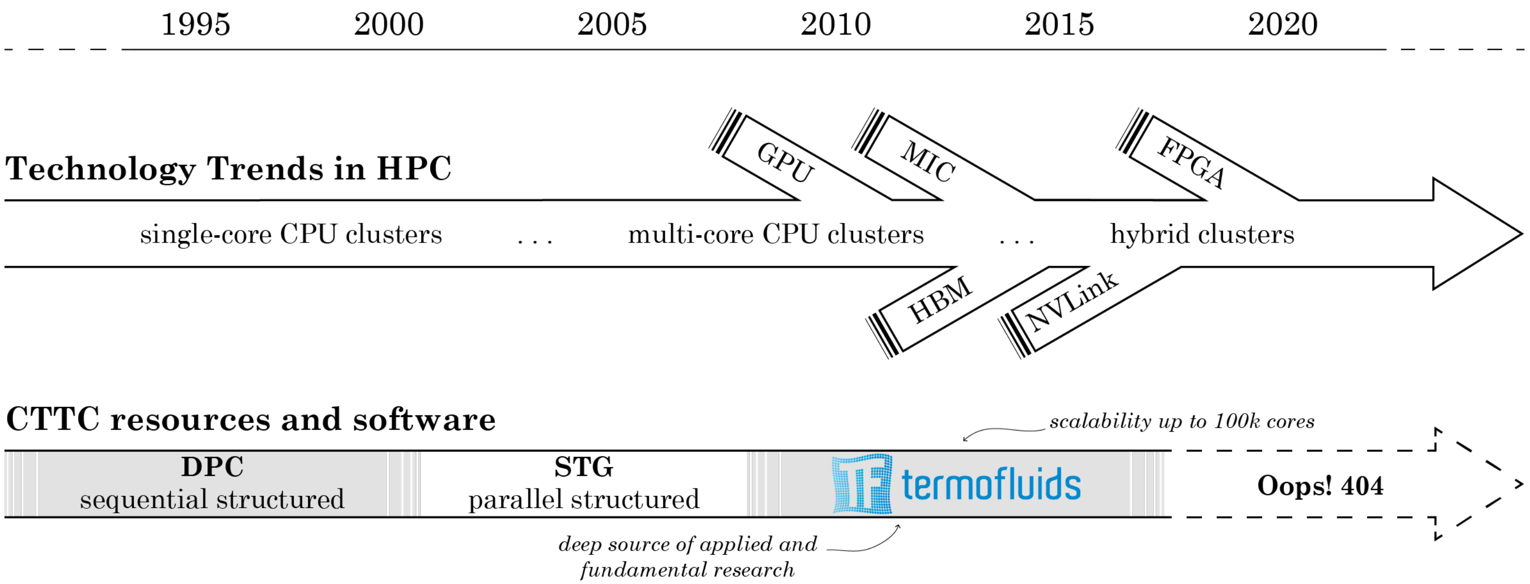
An algebraic approach for CFD simulations

BACKGROUND

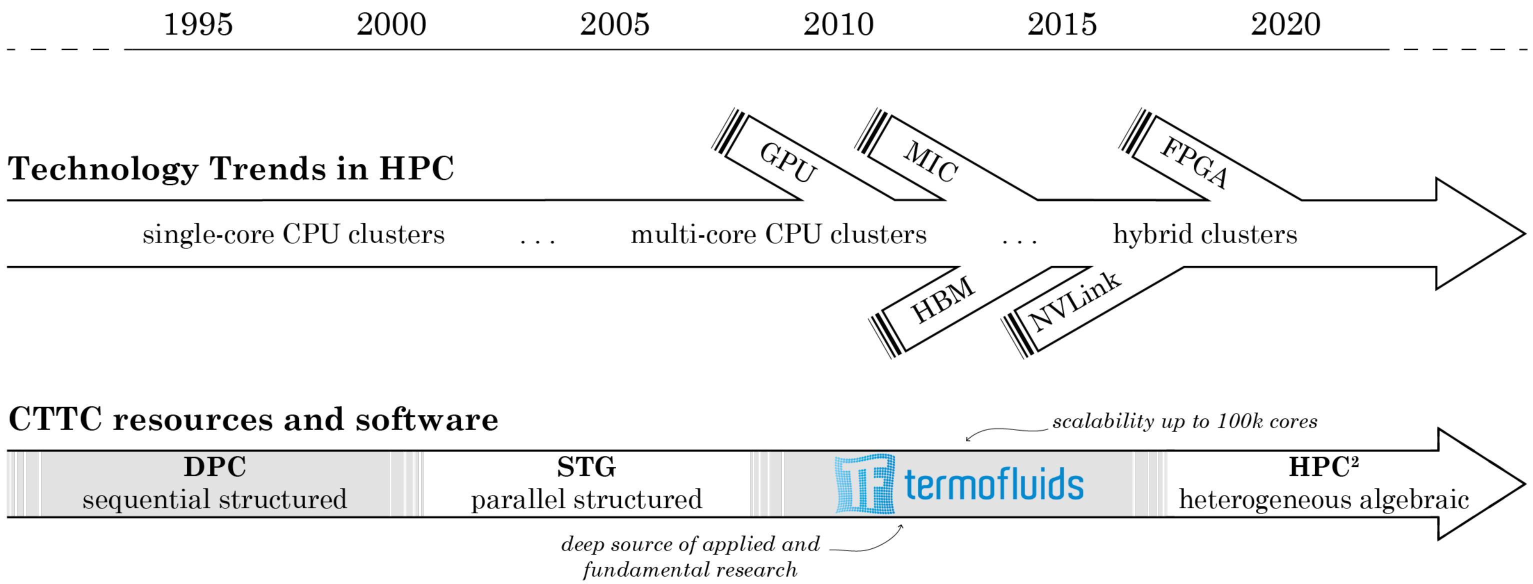
The Heat and Mass Transfer Technological Center
is a research group of the Technical University of Catalonia highly concerned about the environmental sustainability. Specifically, researchers at the CTTC have been enrolled in both fundamental and applied research, studying several phenomena: natural and forced convection, multi-phase flow, aerodynamics, among many others.




1995
2000
2005
2010
2015
2020
Technology Trends in HPC
GPU
MIC
FPGA
single-core CPU clusters
. . .
multi-core CPU clusters
. . .
hybrid clusters
HBM
NVLink



Since the beginning,
researchers of CTTC have developed and adapted CFD codes for the state-of-the art computer resources.



Since the beginning,

researchers of CTTC have developed and adapted CFD codes for the state-of-the art computer resources.

Currently,

a fully-portable, algebra-based framework for heterogeneous computing is being developed. Namely, the traditional stencil data structures and sweeps are replaced by algebraic data structures and kernels, and the discrete operators and mesh functions are then stored as sparse matrices and vectors, respectively.

In an algebraic approach,
the traditional stencil data structures and sweeps are replaced by algebraic data structures and kernels[1].

$$\nabla \cdot u = 0, \qquad u + (u \cdot \nabla)u - \frac{1}{Re}\Delta u + \nabla p = 0$$

$$\mathbf{M}\mathbf{u_s} = \mathbf{0_c}, \qquad \mathbf{\Omega} d_t \mathbf{u_c} + \mathbf{M}\mathbf{U_s}\mathbf{u_c} - \mathbf{D}\mathbf{u_c} - \mathbf{M}^{\mathbf{T}}\mathbf{p_c} = \mathbf{0_c}$$

[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] Álvarez-Farré et al., HPC²–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD, Computers and Fluids, 173, 285–292, 2018.

In an algebraic approach,
the traditional stencil data structures and sweeps are replaced by algebraic data structures and kernels[1].

$$\nabla \cdot u = 0, \qquad u + (u \cdot \nabla)u - \frac{1}{Re}\Delta u + \nabla p = 0$$

$$\mathbf{M}\mathbf{u_s} = \mathbf{0_c}, \qquad \boldsymbol{\Omega} d_t \mathbf{u_c} + \mathbf{M}\mathbf{U_s}\mathbf{u_c} - \mathbf{D}\mathbf{u_c} - \mathbf{M}^{\mathbf{T}}\mathbf{p_c} = \mathbf{0_c}$$

- The discrete operators and mesh functions are stored as sparse matrices and vectors, respectively (i.e., the numerical method results fully integrated into the data).

[1] F. Xavier Trias et al. “Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids”, JCP 258 (2014), 246–267.
[2] Álvarez-Farré et al., HPC²–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD, Computers and Fluids, 173, 285–292, 2018.

In an algebraic approach,

the traditional stencil data structures and sweeps are replaced by algebraic data structures and kernels[1].

$$\nabla \cdot u = 0, \qquad u + (u \cdot \nabla)u - \frac{1}{Re}\Delta u + \nabla p = 0$$

$$\mathbf{M}\mathbf{u_s} = \mathbf{0_c}, \qquad \mathbf{\Omega} d_t \mathbf{u_c} + \mathbf{M}\mathbf{U_s}\mathbf{u_c} - \mathbf{D}\mathbf{u_c} - \mathbf{M}^{\mathbf{T}}\mathbf{p_c} = \mathbf{0_c}$$

SpMV

axpy

- The discrete operators and mesh functions are stored as sparse matrices and vectors, respectively (i.e., the numerical method results fully integrated into the data).
- The algorithm for the DNS and LES of incompressible turbulent flows relies on a set of only three algebraic operations: **SpMV**, **axpy** and **dot**[2].

[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] Álvarez-Farré et al., HPC²–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD, Computers and Fluids, 173, 285–292, 2018.

Stencil-based method

Algebra-based method

Stencil-based method

Given a mesh,
specific stencil loops or kernels are designed to compute quantities such as Gradient or Divergence.

Algebra-based method

Given a mesh,
specific sparse matrices are built to represent discrete operators such as Gradient or Divergence.

Stencil-based method

Given a mesh,
specific stencil loops or kernels are designed to compute quantities such as Gradient or Divergence.

Multiple computing kernels
are required to implement different numerical methods.

Algebra-based method

Given a mesh,
specific sparse matrices are built to represent discrete operators such as Gradient or Divergence.

Unique computing kernels
are required to implement different numerical methods.

Stencil-based method

Given a mesh,
specific stencil loops or kernels are designed to compute quantities such as Gradient or Divergence.

Multiple computing kernels
are required to implement different numerical methods.

Complex kernels
minimize intermediate calculations and help to increase the arithmetic intensity.

Algebra-based method

Given a mesh,
specific sparse matrices are built to represent discrete operators such as Gradient or Divergence.

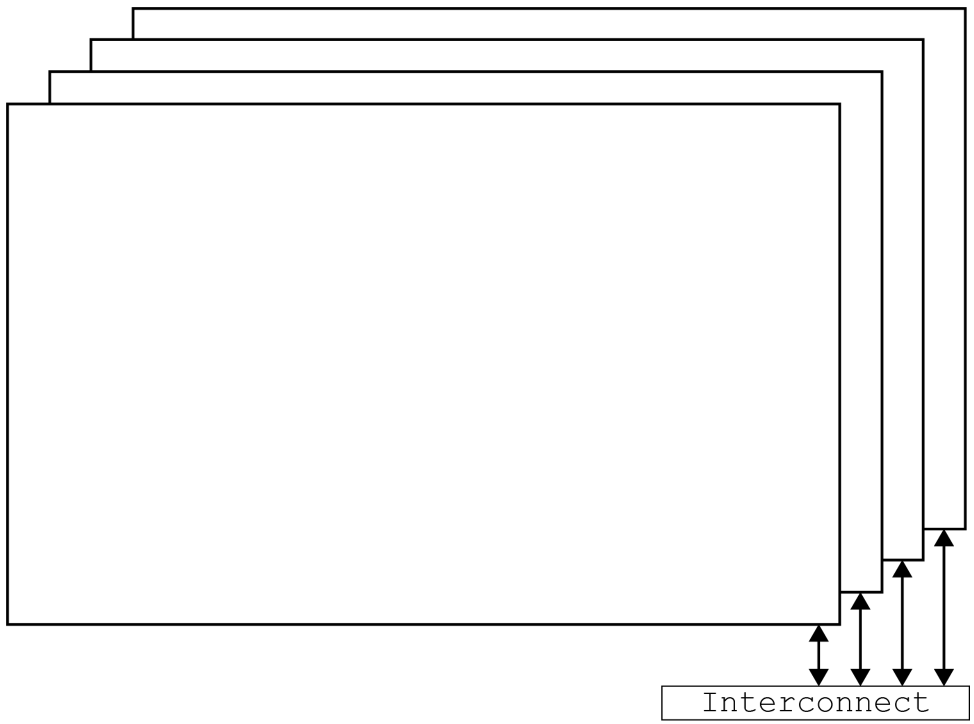
Unique computing kernels
are required to implement different numerical methods.

Simple kernels
are reusable and exist in many optimized libraries. Thus, an algebra-based framework is naturally portable.

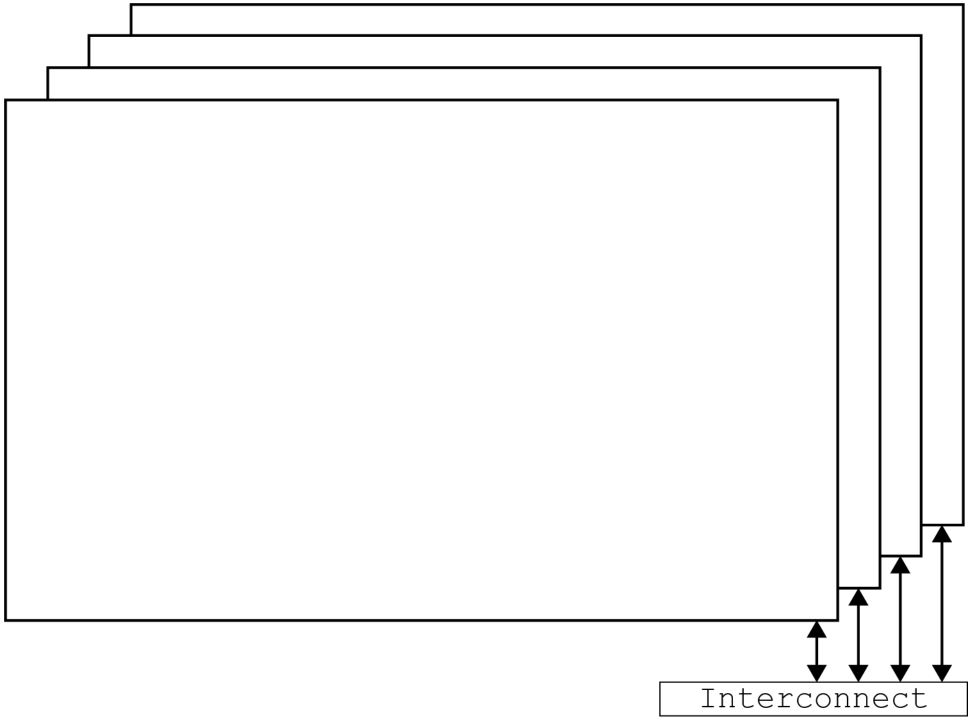
Hierarchical implementation of the HPC² framework

IMPLEMENTATION

High-performance computing systems



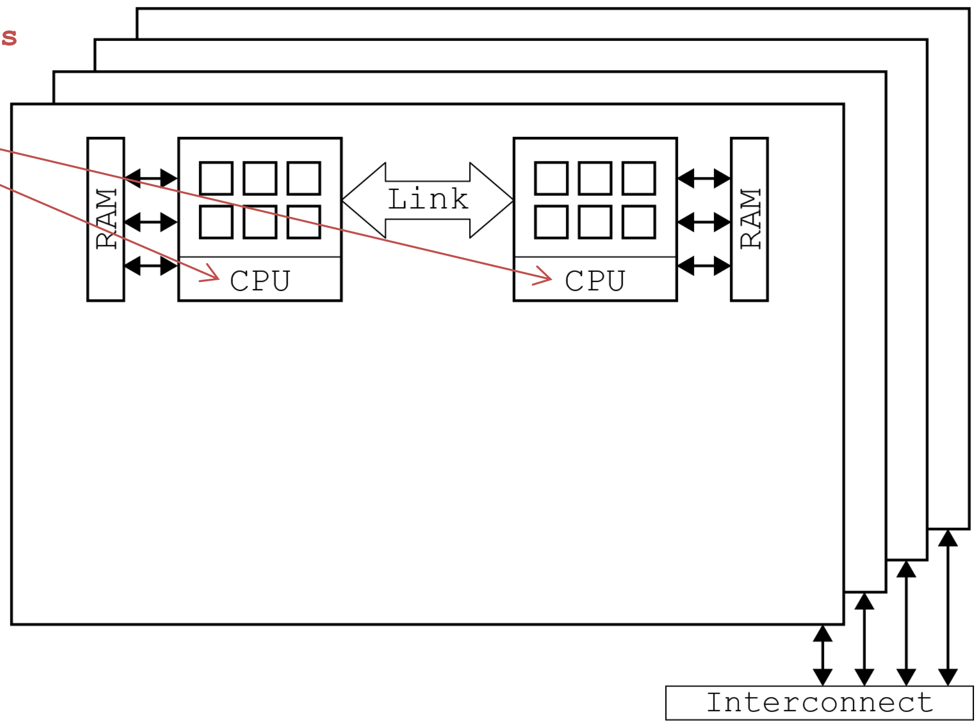
High-performance computing systems



multiple nodes interconnected via
high-bandwidth network
we use MPI at this level

High-performance computing systems

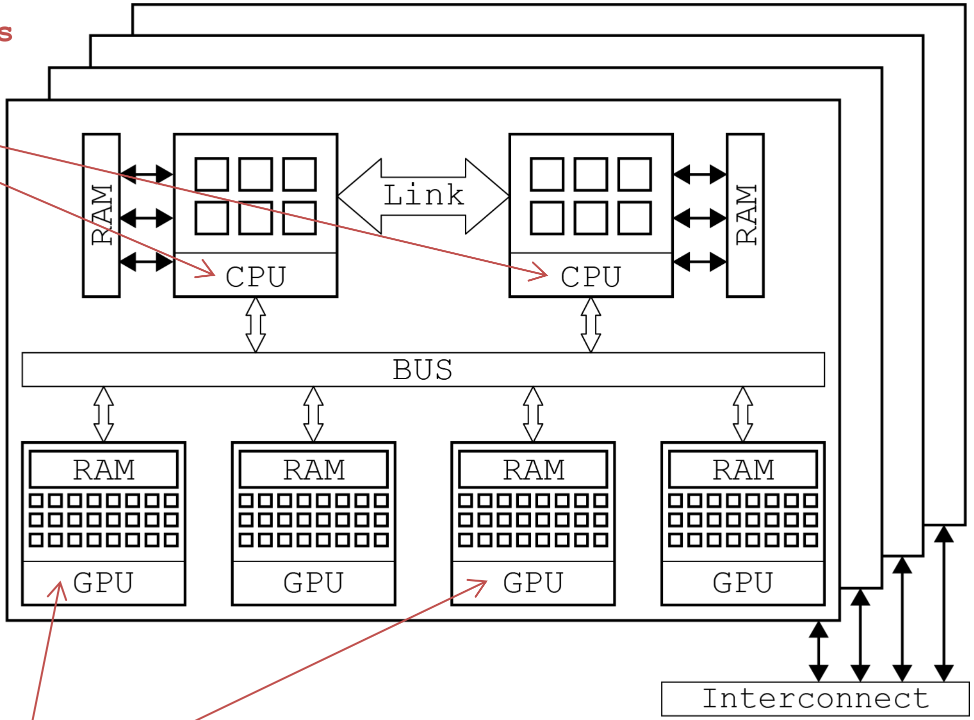
multiple multi-core CPU per node
we use OpenMP at this level



multiple nodes interconnected via high-bandwidth network
we use MPI at this level

High-performance computing systems

multiple multi-core CPU per node
we use OpenMP at this level



multiple accelerators per node
we use OpenCL/CUDA at this level

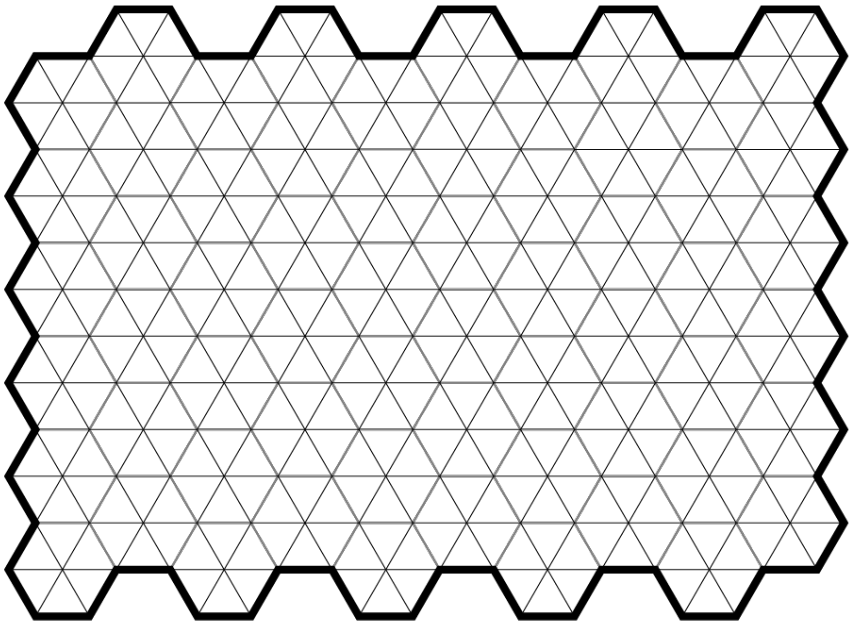
multiple nodes interconnected via high-bandwidth network
we use MPI at this level

Multilevel workload distribution consists of dividing the computational domain (mesh) into subsets recursively to distribute it among the hardware of a computing system.

First-level

Second-level

Third-level

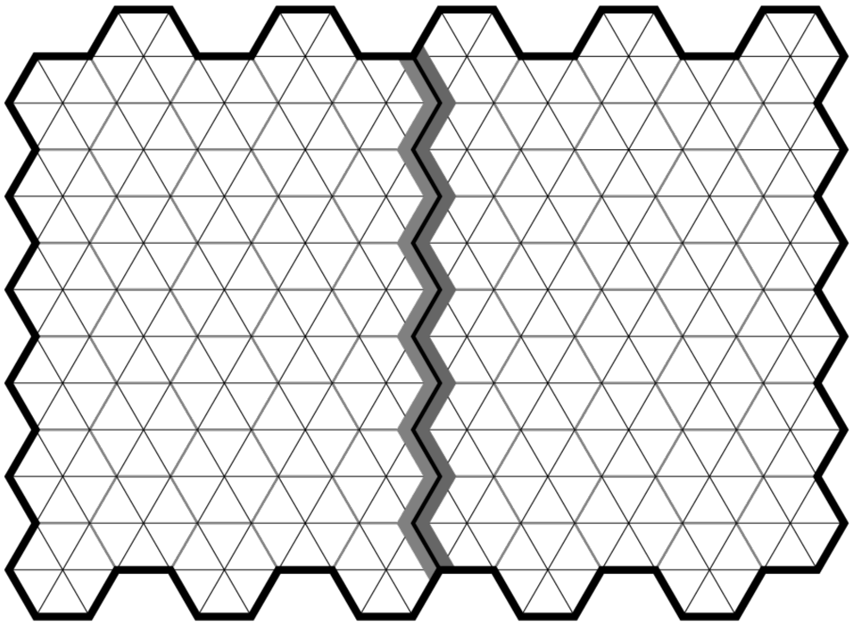


Multilevel workload distribution consists of dividing the computational domain (mesh) into subsets recursively to distribute it among the hardware of a computing system.

First-level decomposition divides the workload among the computing nodes, that is, the MPI processes.

Second-level

Third-level

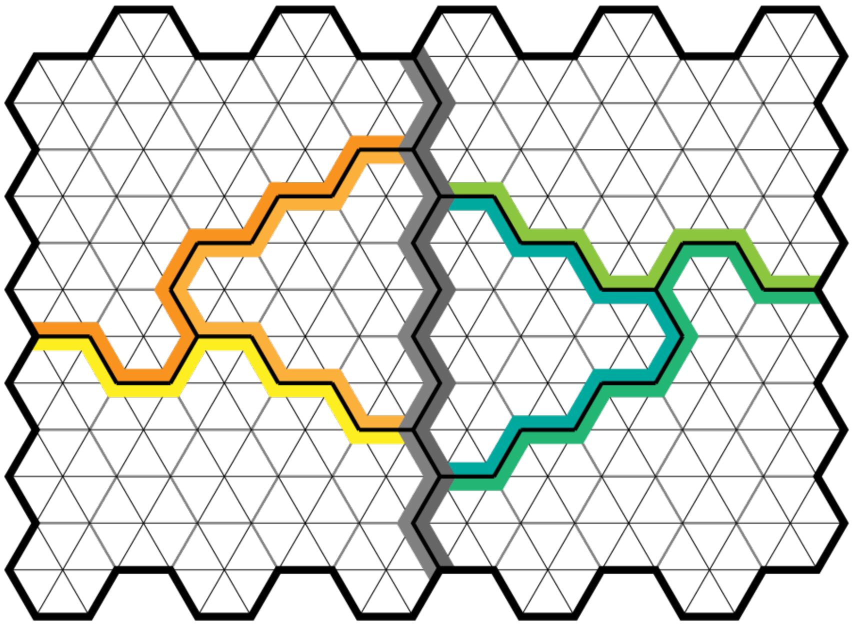


Multilevel workload distribution consists of dividing the computational domain (mesh) into subsets recursively to distribute it among the hardware of a computing system.

First-level decomposition divides the workload among the computing nodes, that is, the MPI processes.

Second-level decomposition divides the first-level partitions to share each MPI's workload among its available hardware, that is, the host and co-processors.

Third-level

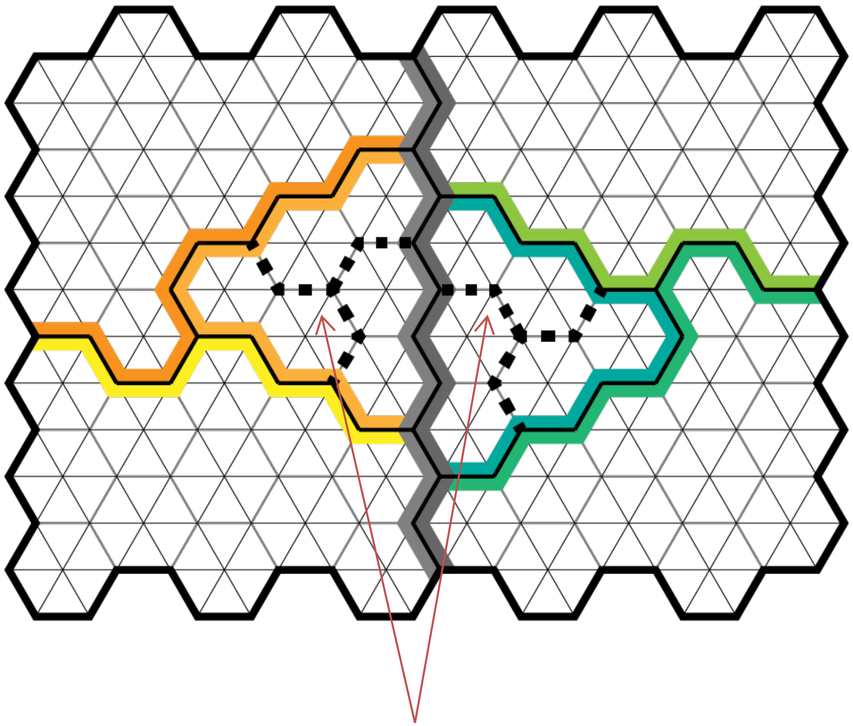


Multilevel workload distribution consists of dividing the computational domain (mesh) into subsets recursively to distribute it among the hardware of a computing system.

First-level decomposition divides the workload among the computing nodes, that is, the MPI processes.

Second-level decomposition divides the first-level partitions to share each MPI's workload among its available hardware, that is, the host and co-processors.

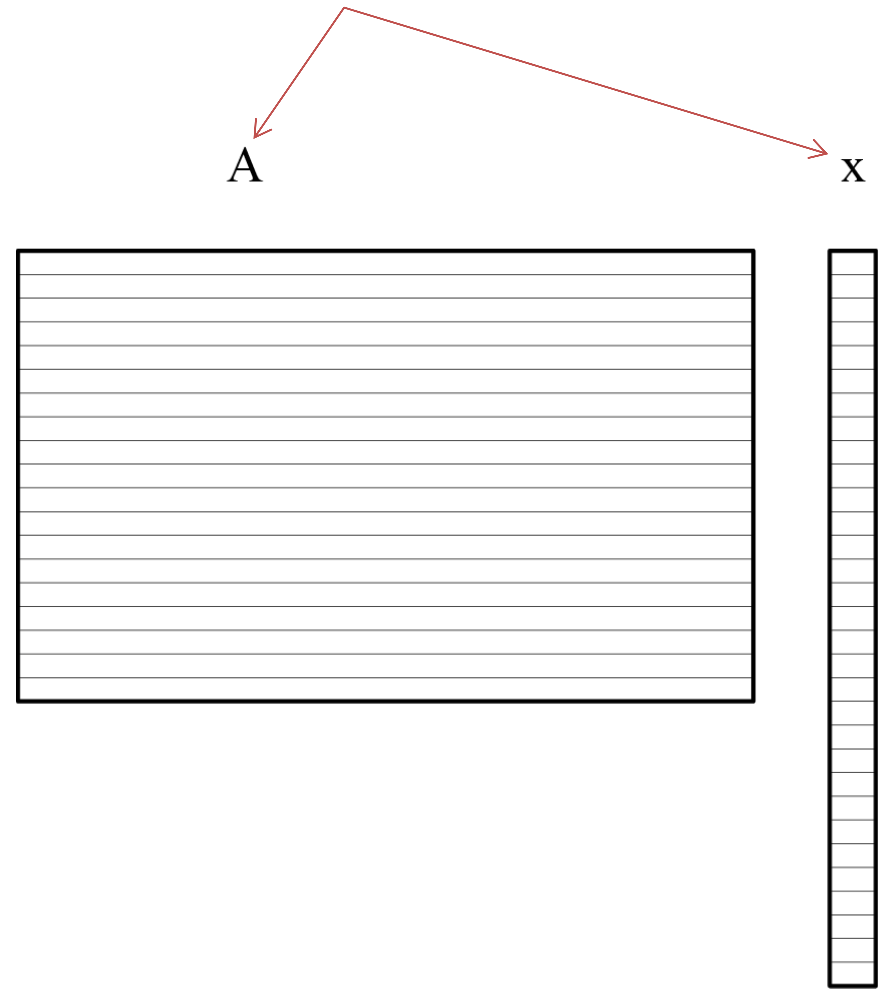
Third-level decomposition divides the second-level partitions to distribute the workload of a device whose shared-memory space introduces a significant NUMA factor, that is, multiple NUMA nodes in a manycore CPU.



can be done with implicit dynamic scheduling or explicit static limits

Sparse matrices and vectors

blocks represent the second-level partition of sparse matrix and vector instances assigned to each device.

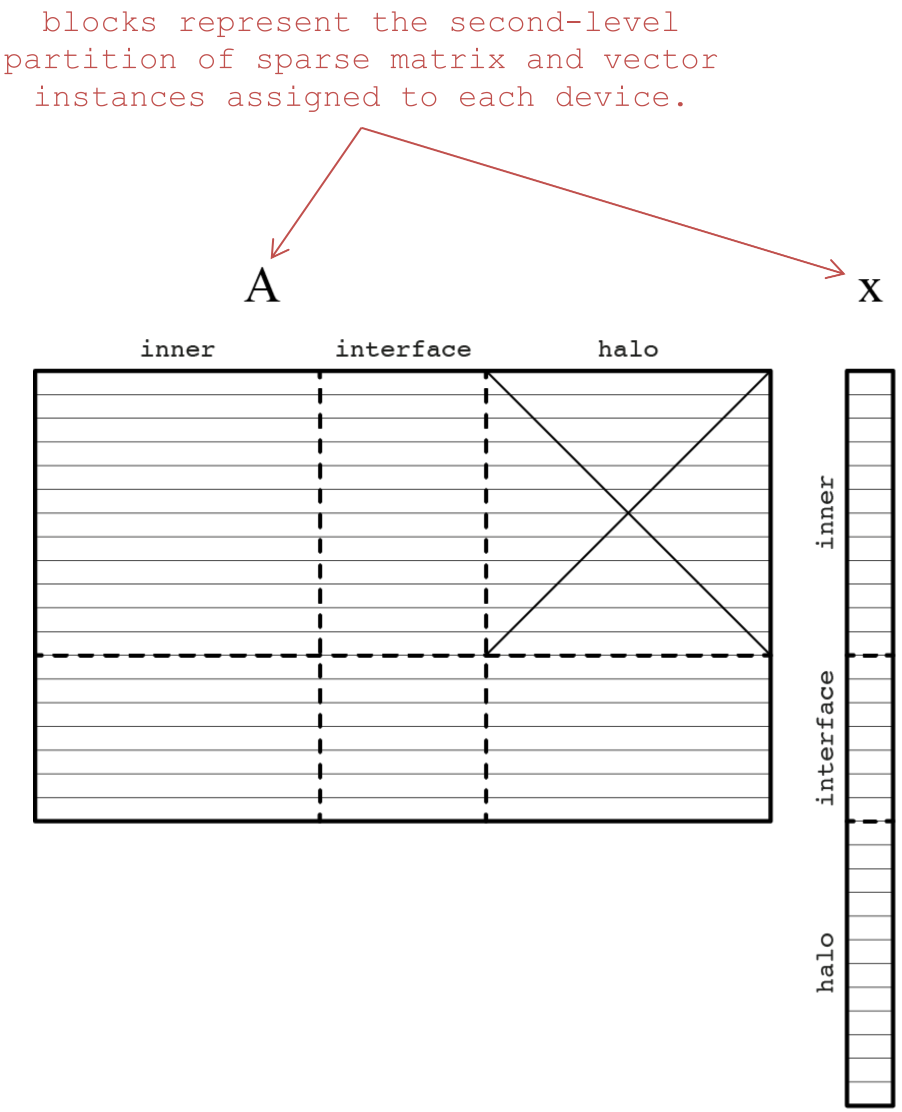


Sparse matrices and vectors represent discrete operators and mesh functions in our algebra-based framework, respectively. Thus, sparse matrix rows and vector elements are distributed among subdomains, and classified into three subsets:

Inner

Interface

Halo



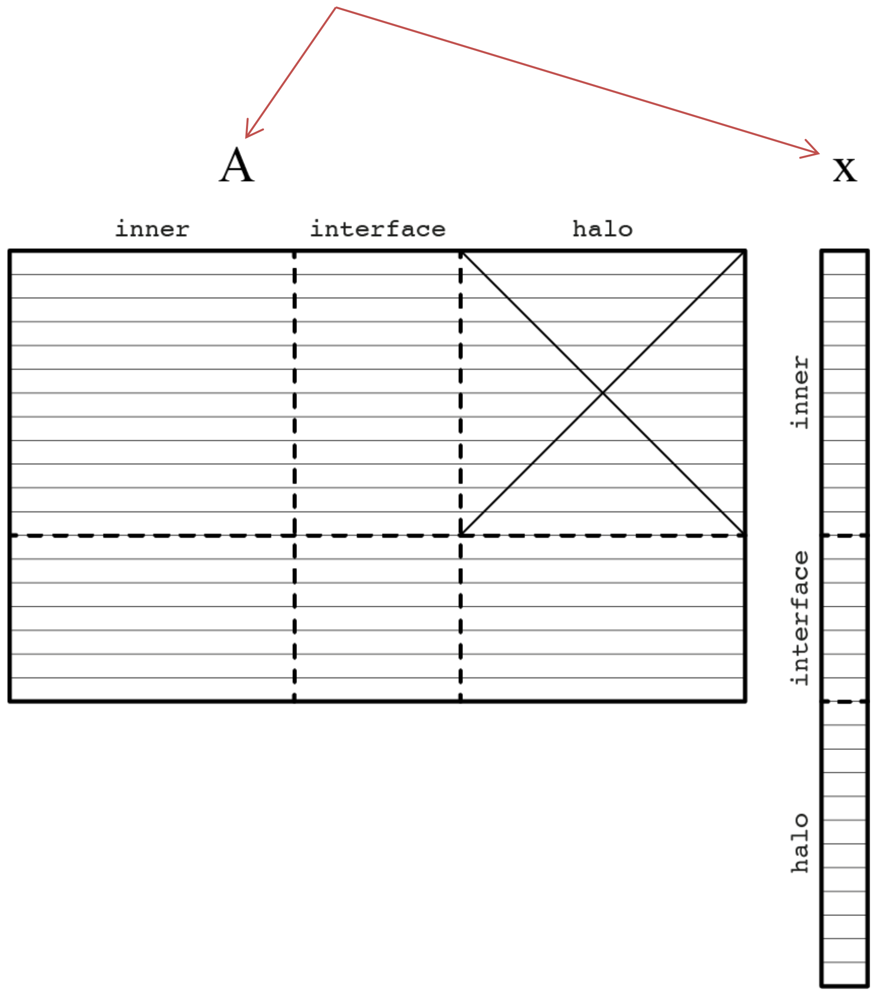
Sparse matrices and vectors represent discrete operators and mesh functions in our algebra-based framework, respectively. Thus, sparse matrix rows and vector elements are distributed among subdomains, and classified into three subsets:

Inner
subset is composed of local elements which are coupled with local elements only.

Interface

Halo

blocks represent the second-level partition of sparse matrix and vector instances assigned to each device.



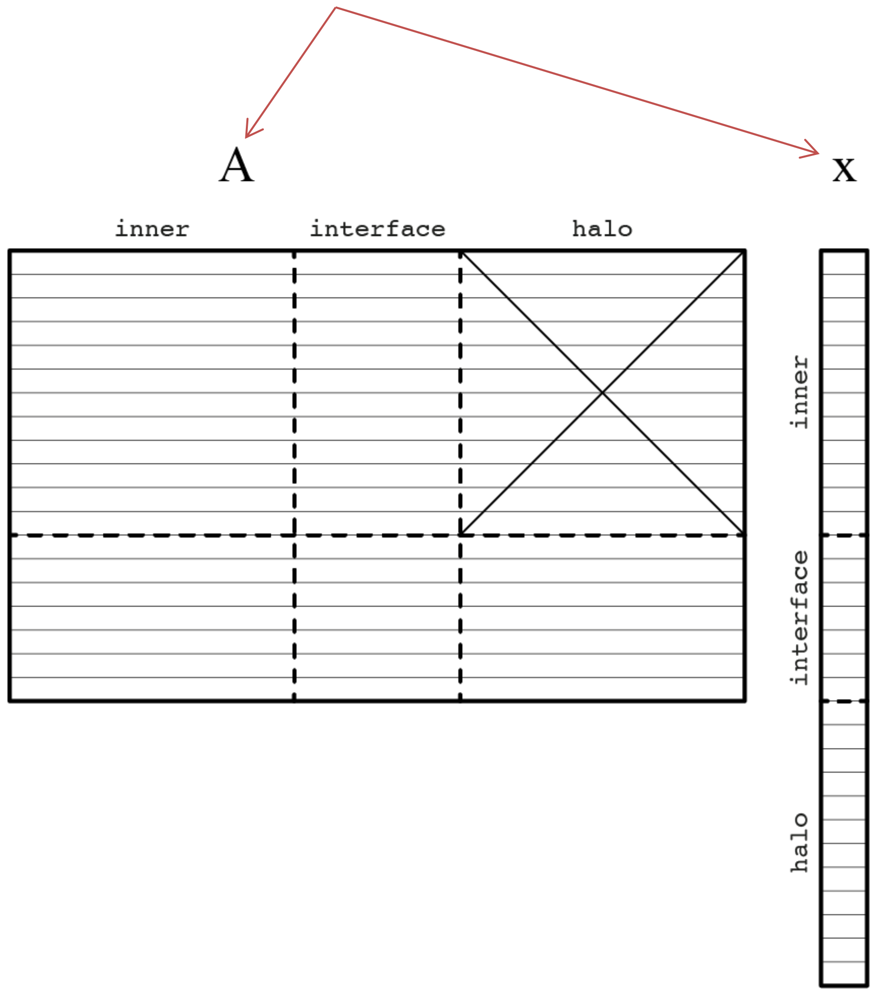
Sparse matrices and vectors represent discrete operators and mesh functions in our algebra-based framework, respectively. Thus, sparse matrix rows and vector elements are distributed among subdomains, and classified into three subsets:

Inner subset is composed of local elements which are coupled with local elements only.

Interface subset is composed of local elements which are coupled with an element from other subdomain.

Halo

blocks represent the second-level partition of sparse matrix and vector instances assigned to each device.



Sparse matrices and vectors

represent discrete operators and mesh functions in our algebra-based framework, respectively. Thus, sparse matrix rows and vector elements are distributed among subdomains, and classified into three subsets:

Inner

subset is composed of local elements which are coupled with local elements only.

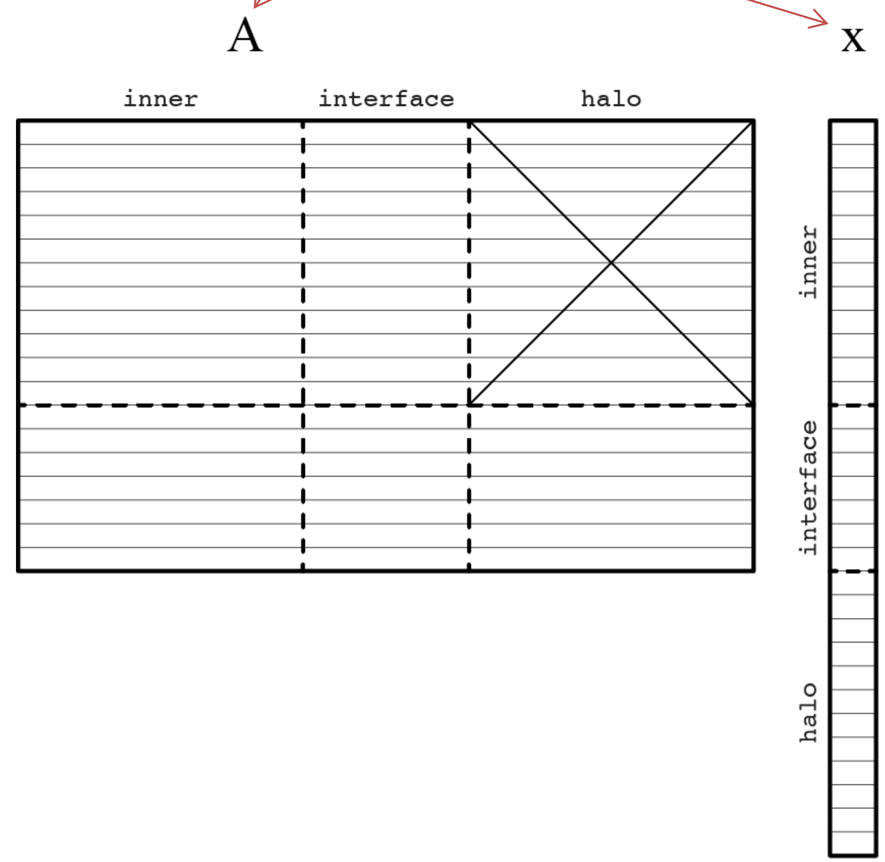
Interface

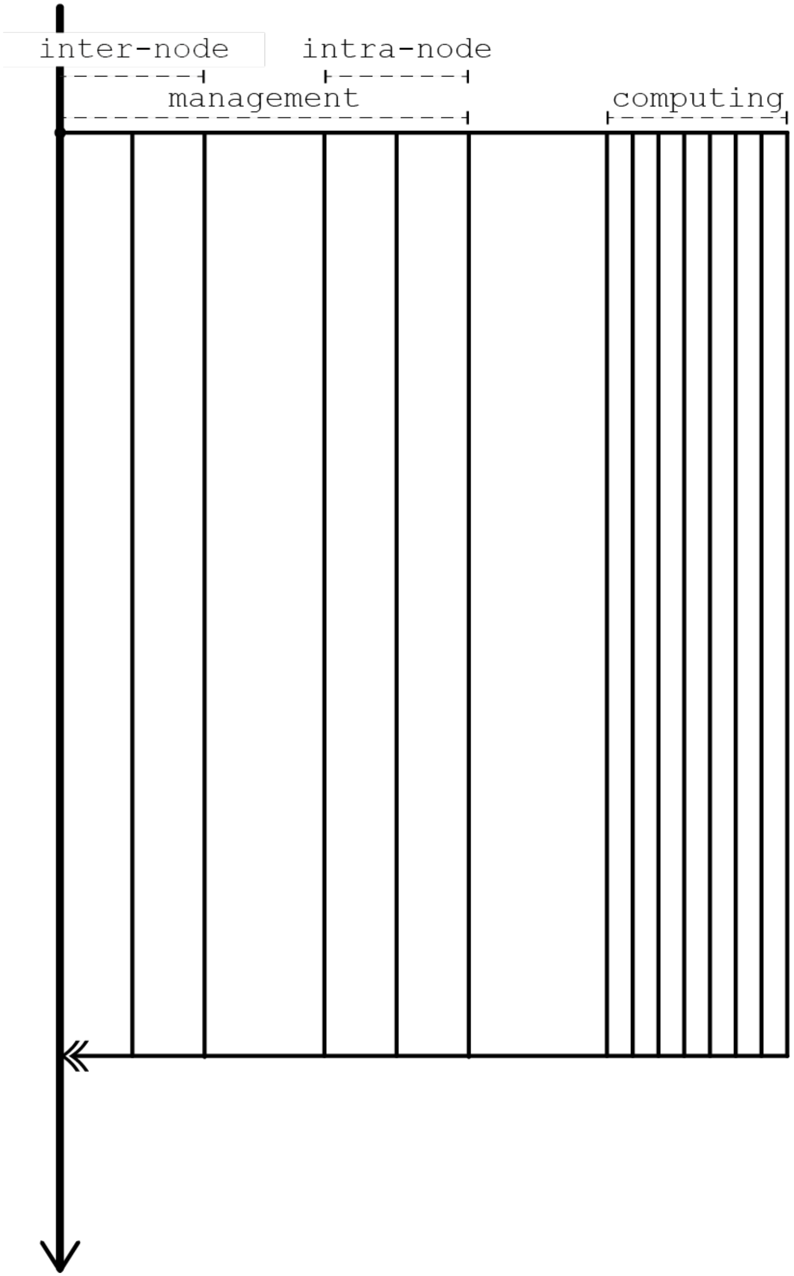
subset is composed of local elements which are coupled with an element from other subdomain.

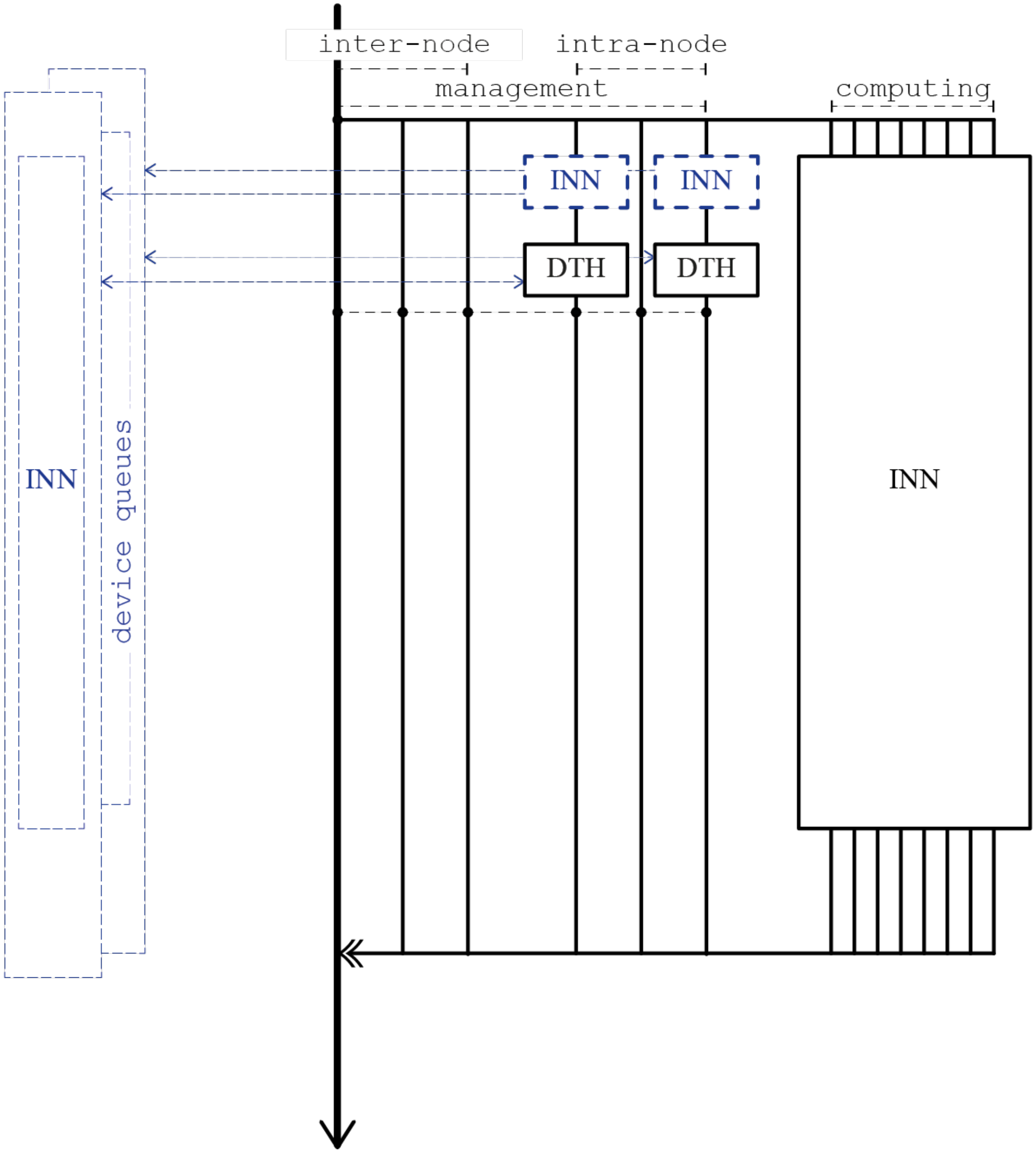
Halo

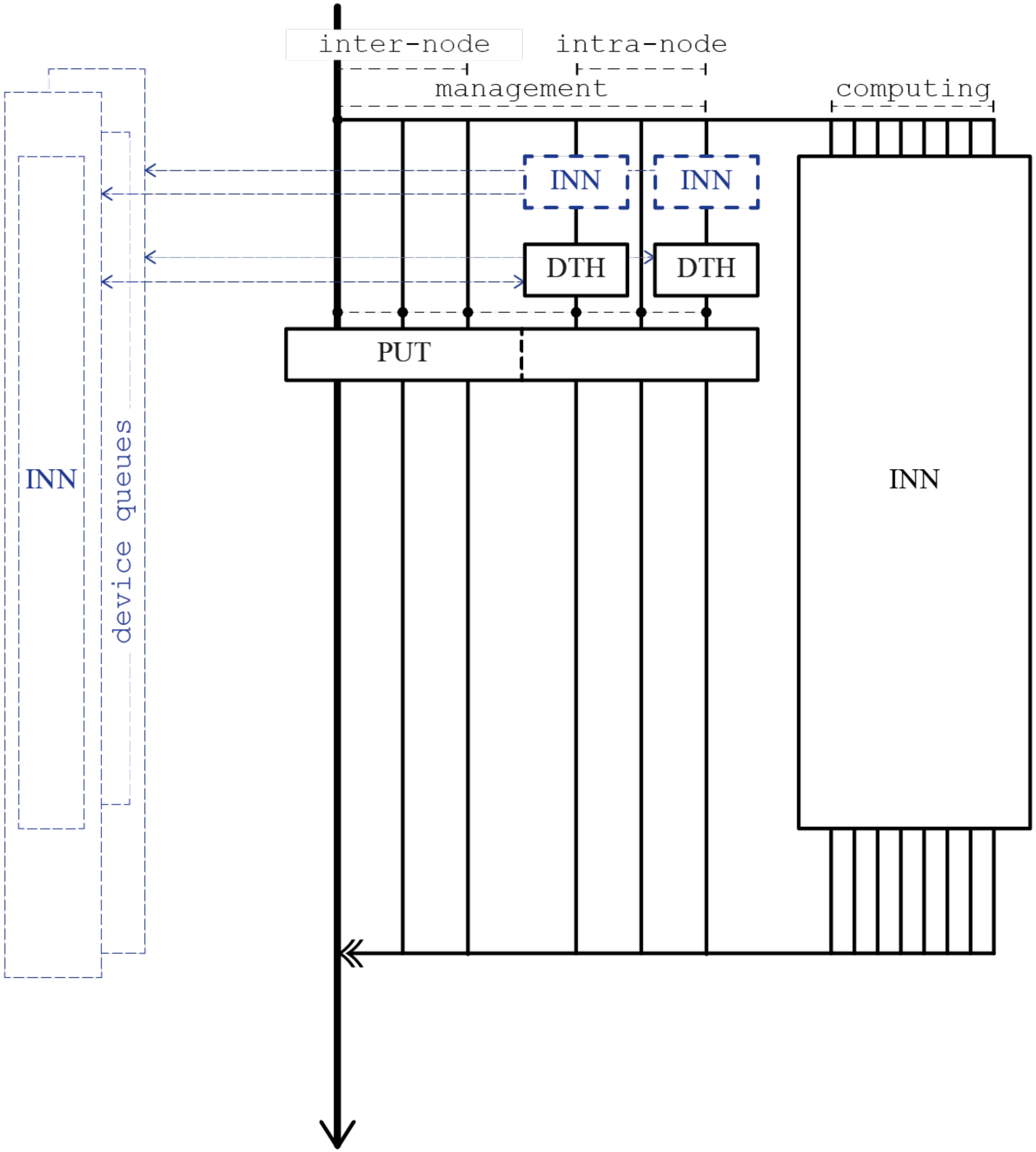
subset is composed of elements from other subdomains which are coupled with local elements.

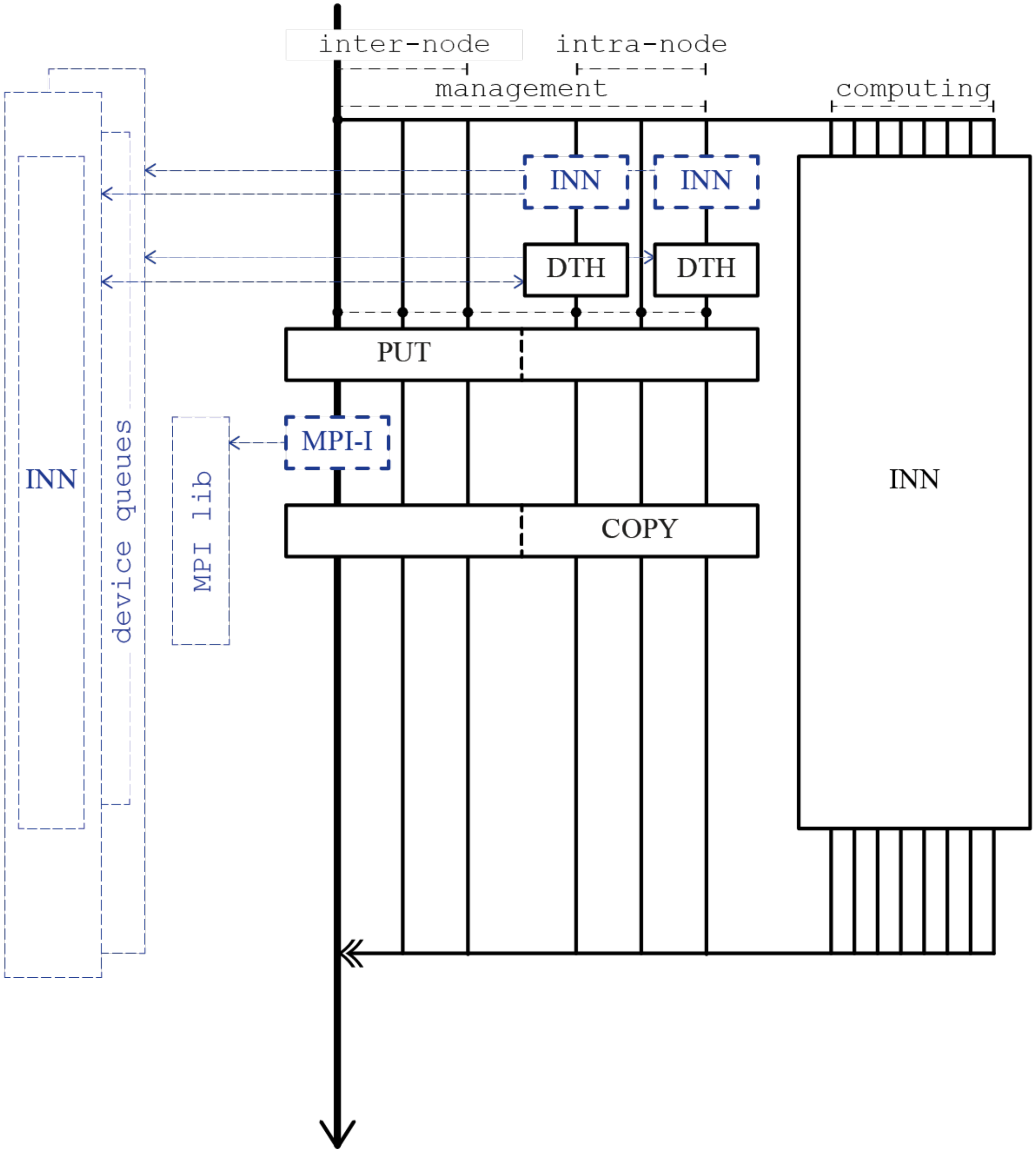
blocks represent the second-level partition of sparse matrix and vector instances assigned to each device.

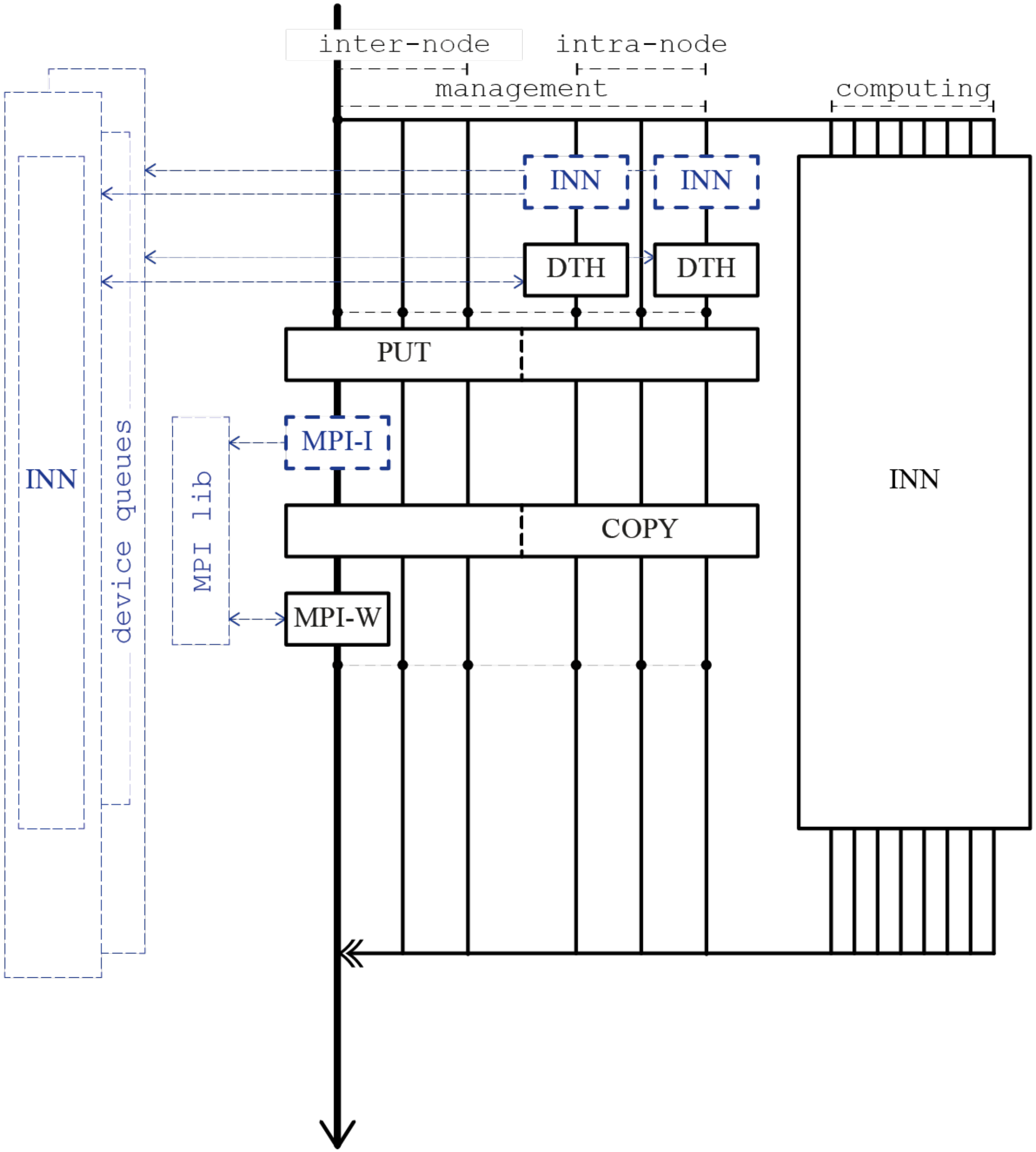


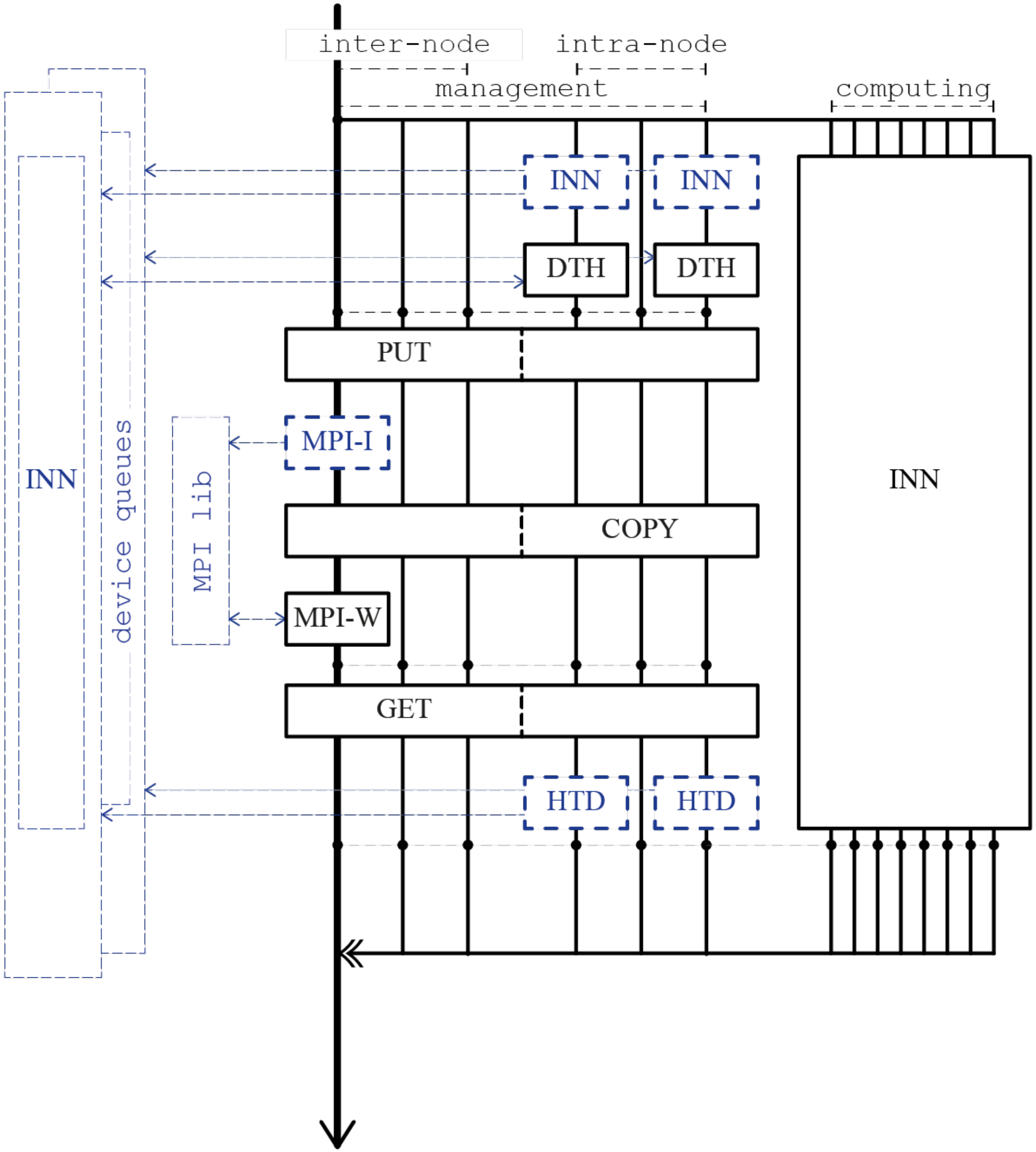

inter-node
intra-node
management
computing

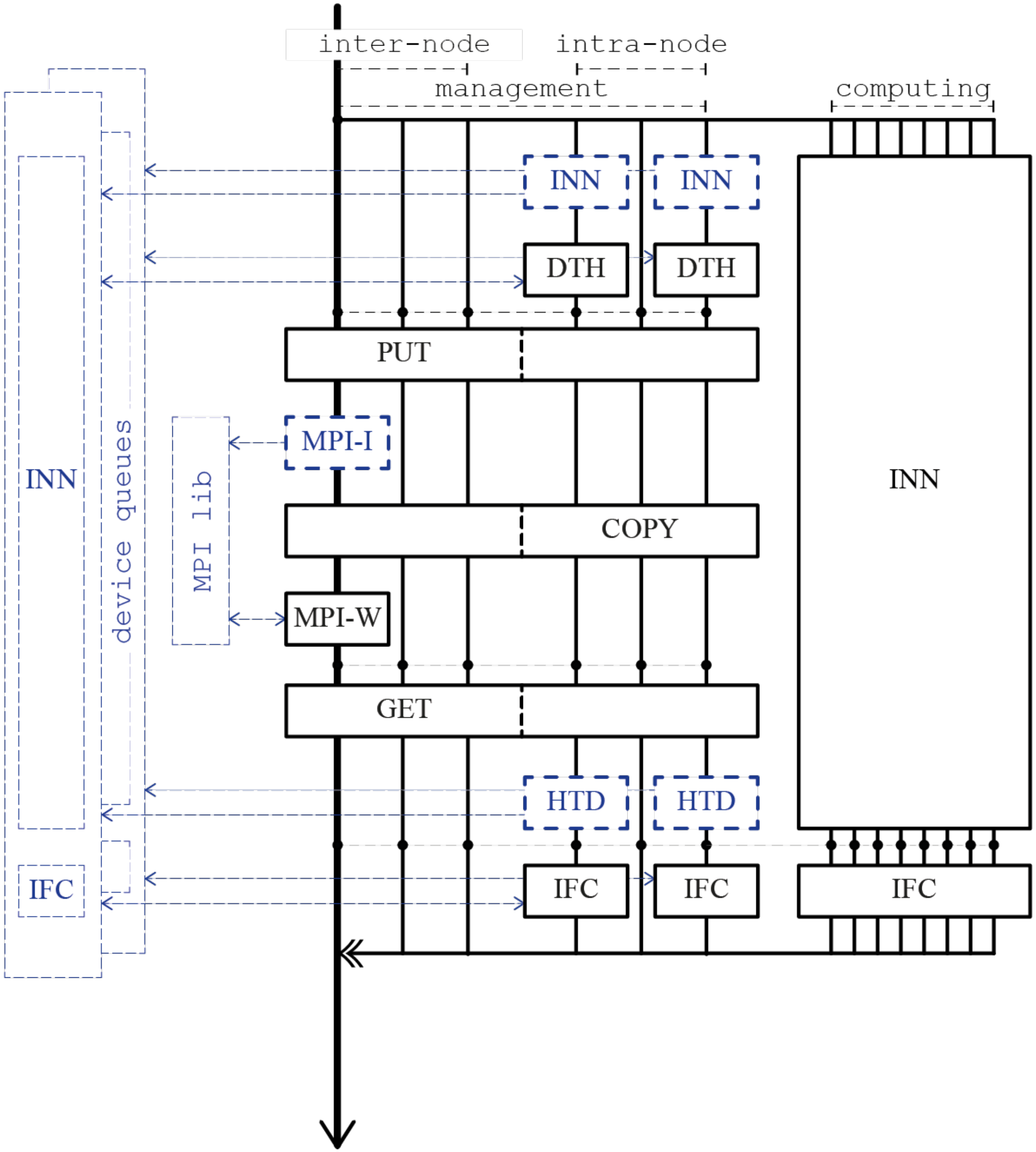

inter-node
intra-node
management
computing
INN
INN
DTH
DTH
INN
device queues
INN


inter-node
intra-node
management
computing
INN
INN
DTH
DTH
PUT
INN
device queues
INN


inter-node
intra-node
management
computing
INN
INN
DTH
DTH
PUT
MPI-I
COPY
INN
INN
device queues
MPI lib

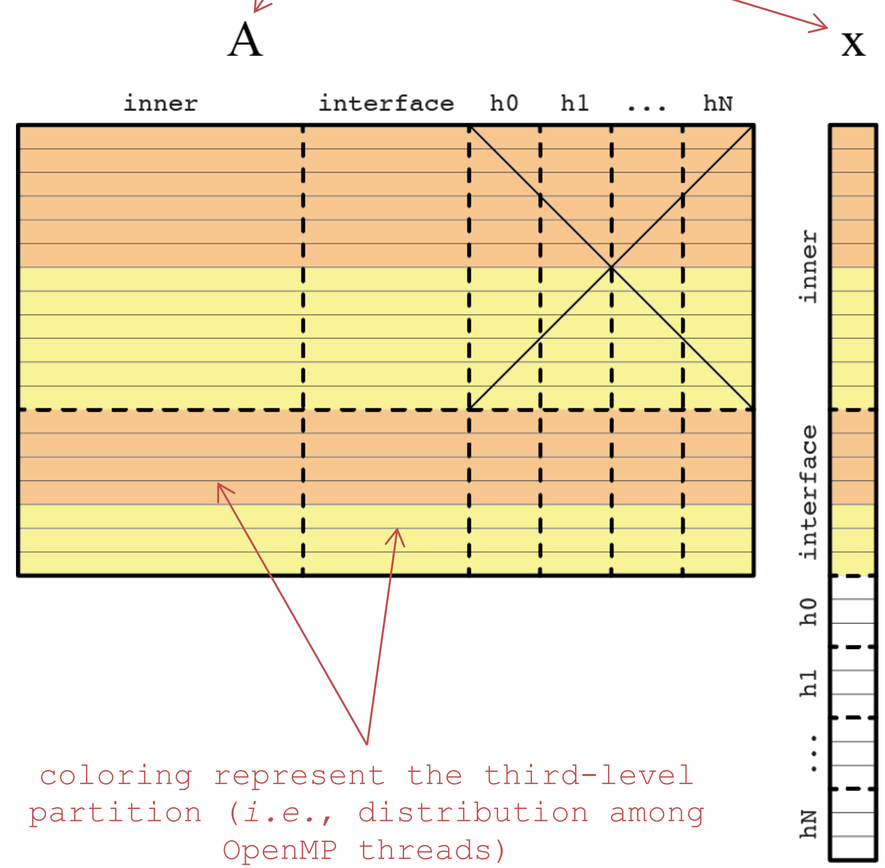

inter-node
intra-node
management
computing
INN
INN
DTH
DTH
PUT
MPI-I
COPY
MPI-W
INN
device queues
MPI lib
INN


inter-node
intra-node
management
computing
INN
INN
DTH
DTH
PUT
MPI-I
COPY
MPI-W
GET
HTD
HTD
INN
INN
device queues
MPI lib


inter-node
intra-node
management
computing
INN
INN
DTH
DTH
PUT
MPI-I
COPY
MPI-W
GET
HTD
HTD
IFC
IFC
IFC
INN
INN
IFC
device queues
MPI lib

Third-level partition
consists of sharing the device's workload in a shared-memory space. This can be implicitly achieved with the OpenMP scheduler.

blocks represent the second-level partition of sparse matrix and vector instances assigned to each device.

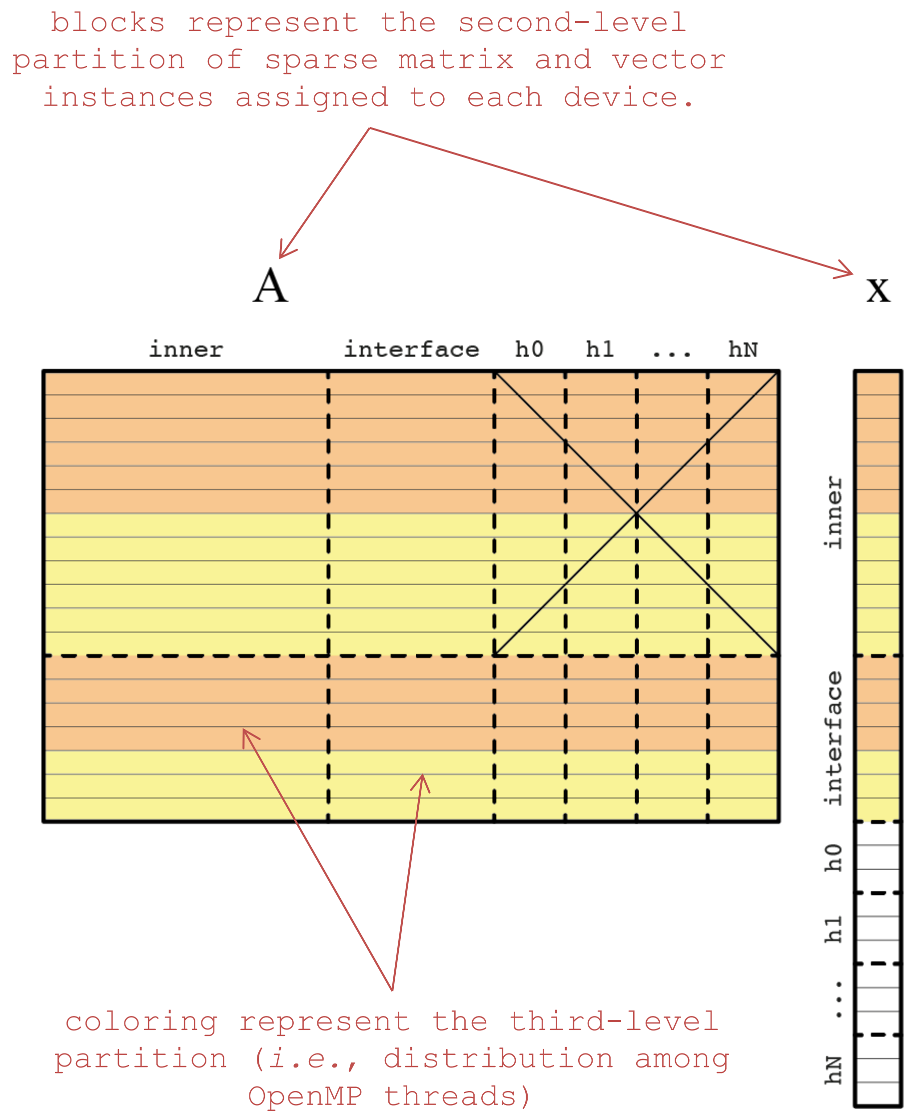


coloring represent the third-level partition (*i.e.*, distribution among OpenMP threads)

Third-level partition

consists of sharing the device's workload in a shared-memory space. This can be implicitly achieved with the OpenMP scheduler.

However, in NUMA configurations, a thread accessing to the memory allocated by another memory controller goes through a much slower bus.

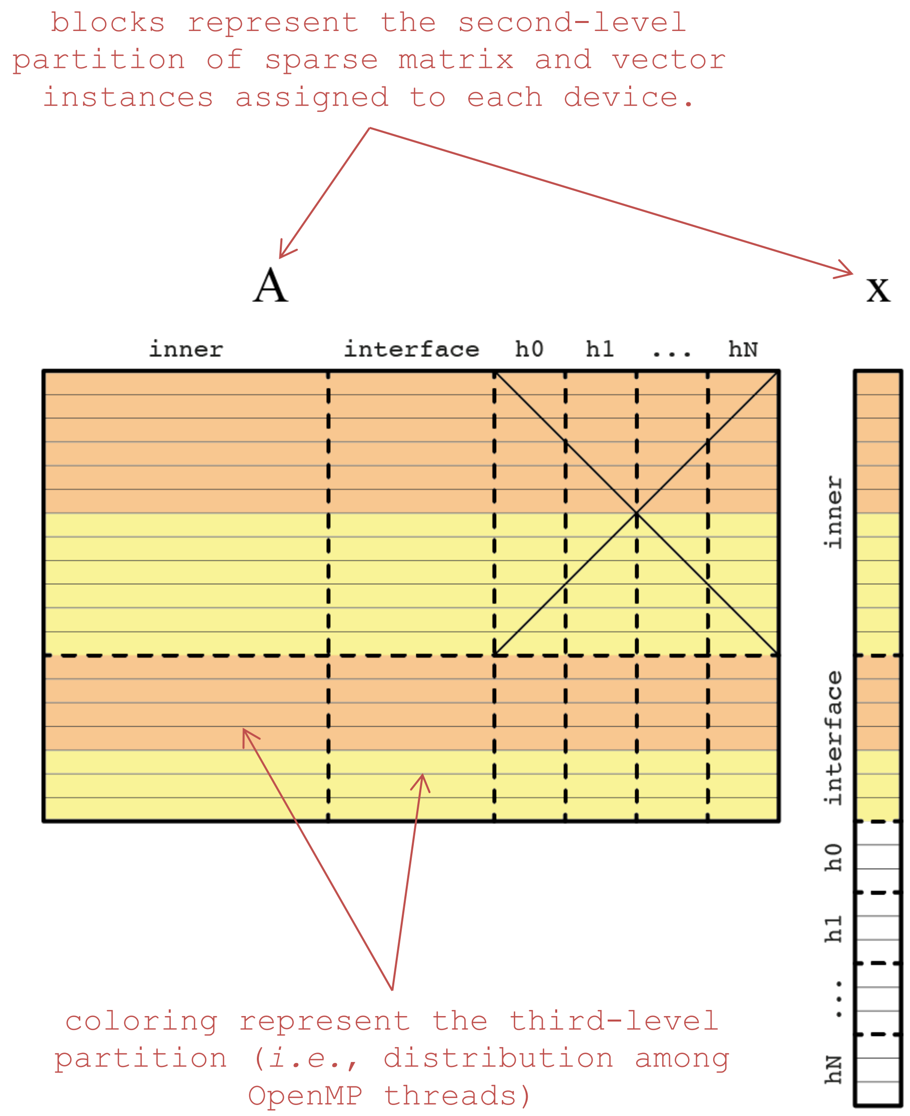


Third-level partition

consists of sharing the device's workload in a shared-memory space. This can be implicitly achieved with the OpenMP scheduler.

However, in NUMA configurations, a thread accessing to the memory allocated by another memory controller goes through a much slower bus.

Since SpMV is executed separately for inner and interface, our third-level partitioning distributes the inner and interface subsets among threads separately too. In other words, each OpenMP thread is assigned a chunk of inner and a chunk of interface.



Third-level partition

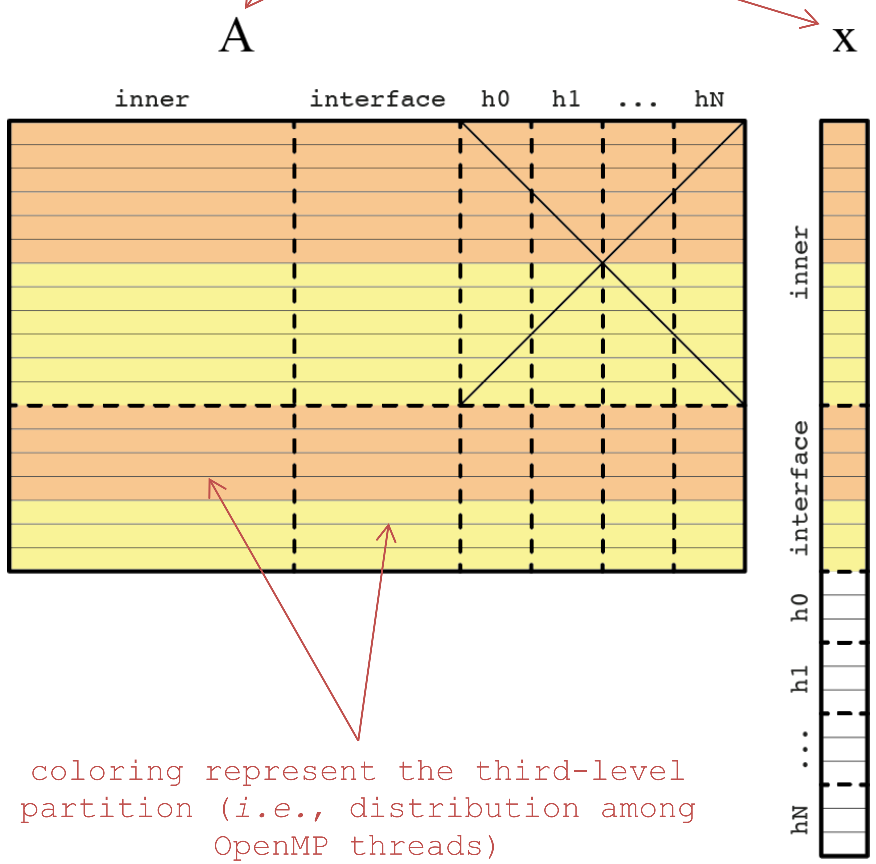
consists of sharing the device's workload in a shared-memory space. This can be implicitly achieved with the OpenMP scheduler.

However, in NUMA configurations, a thread accessing to the memory allocated by another memory controller goes through a much slower bus.

Since SpMV is executed separately for inner and interface, our third-level partitioning distributes the inner and interface subsets among threads separately too. In other words, each OpenMP thread is assigned a chunk of inner and a chunk of interface.

Thread migration must be avoided to ensure an efficient memory access!

blocks represent the second-level partition of sparse matrix and vector instances assigned to each device.



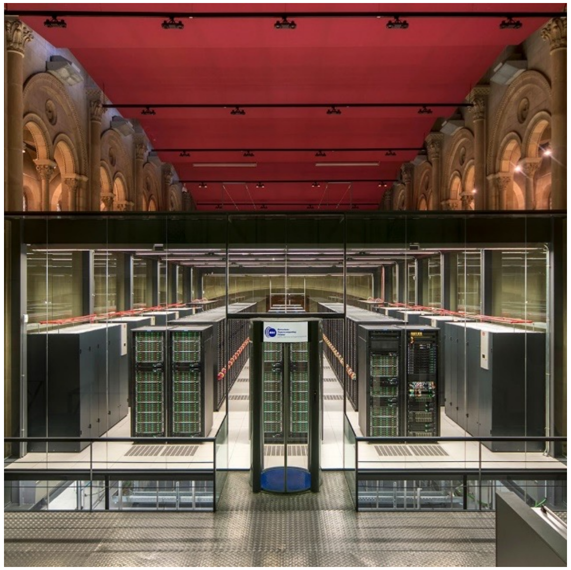
coloring represent the third-level partition (*i.e.*, distribution among OpenMP threads)

Execution of SpMV on different modern supercomputers

PERFORMANCE STUDY

Testing setup

MareNostrum 4



rank #42
3456 nodes with:
  2× Intel Xeon 8160
  1× Intel Omni-Path

TSUBAME3.0

rank #31
540 nodes with:
  2× Intel Xeon E5-2680 v4
  4× NVIDIA Tesla P100
  4× Intel Omni-Path

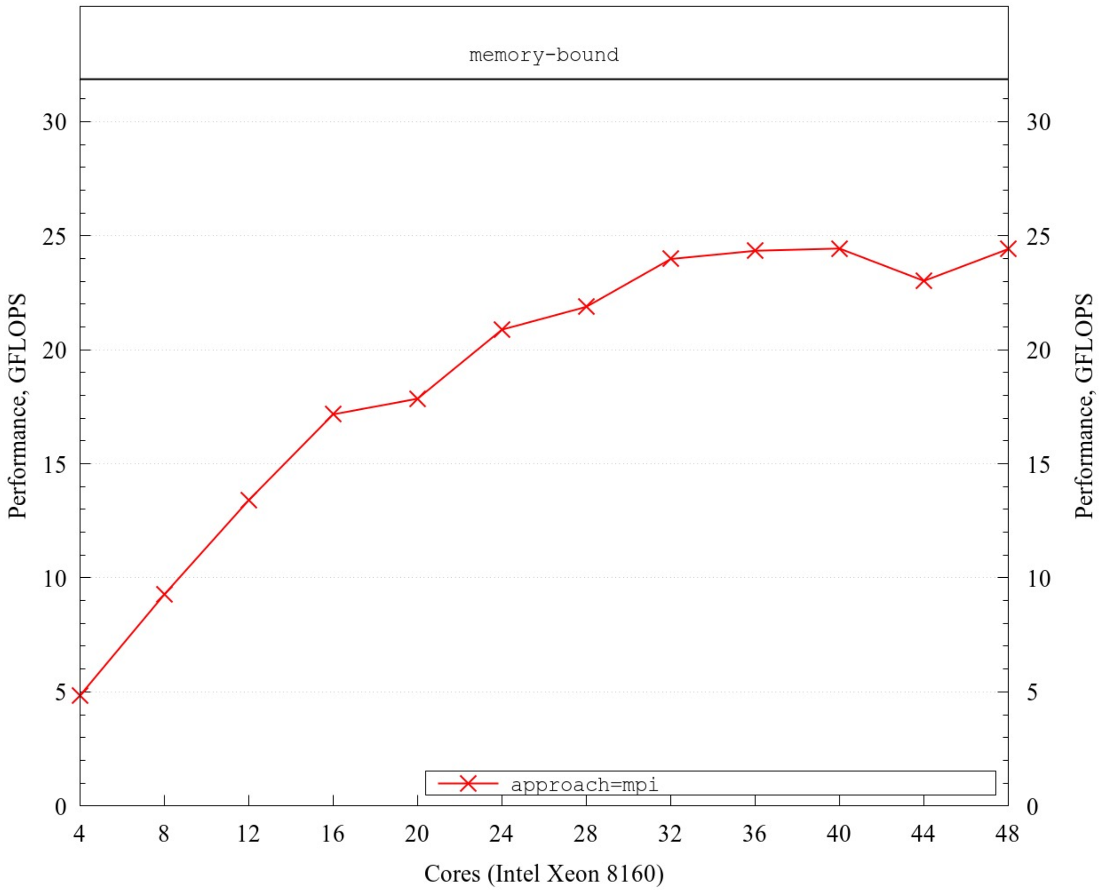
Test case 1:

Single-node performance of SpMV kernel vs number of cores on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells. The sparse matrix storage format used is ELLPACK[2].

[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 1:

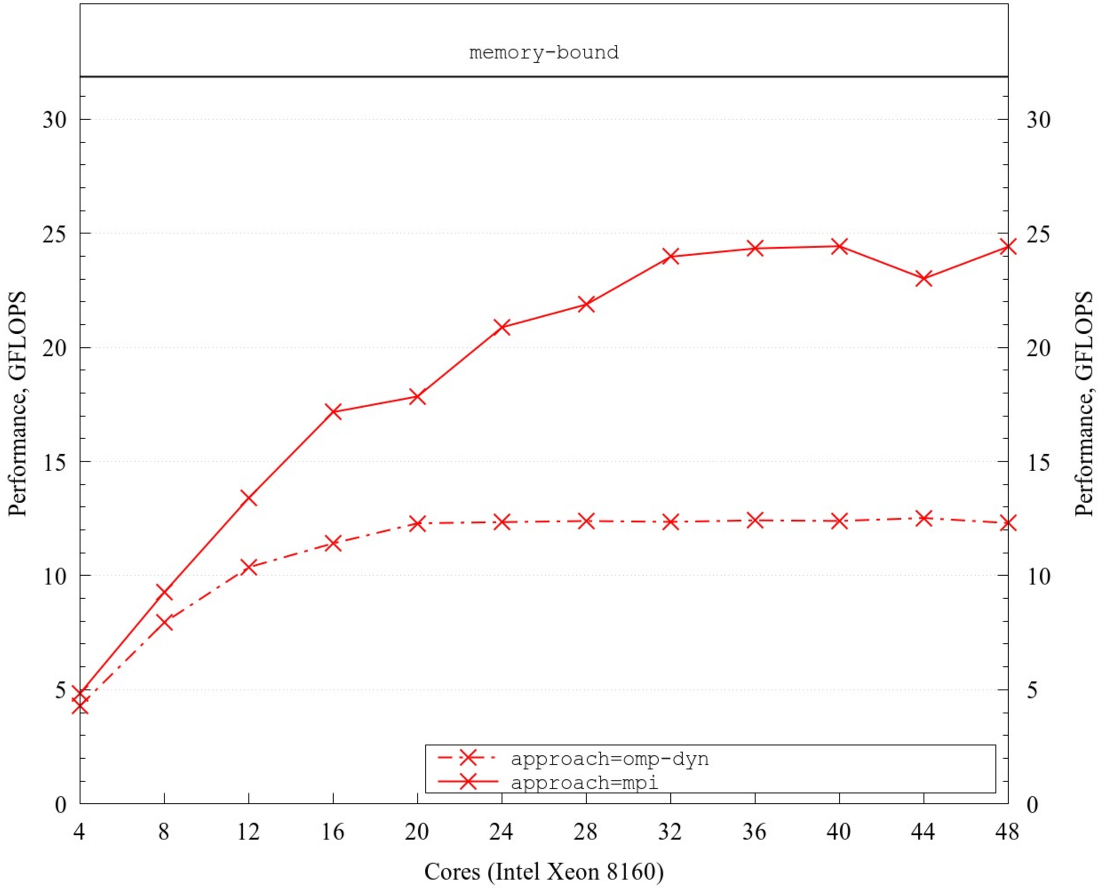
Single-node performance of SpMV kernel vs number of cores on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells. The sparse matrix storage format used is ELLPACK[2].



[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 1:

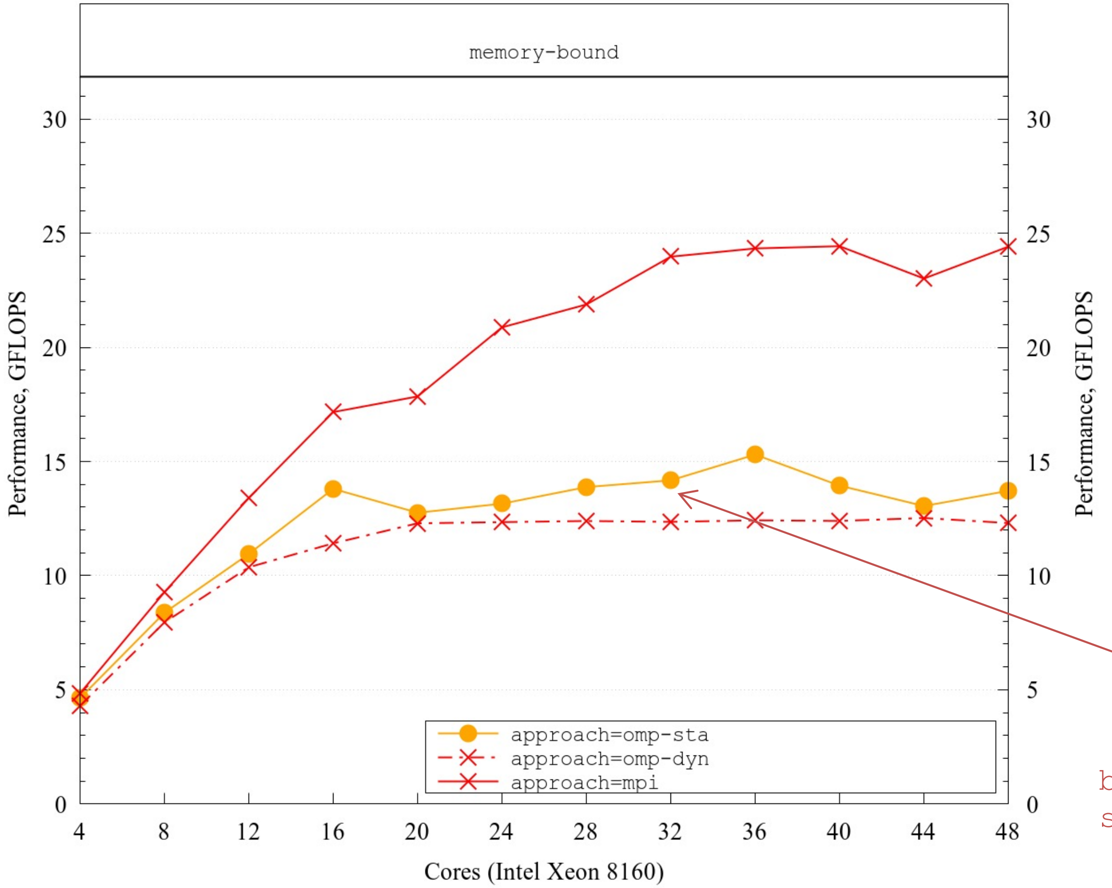
Single-node performance of SpMV kernel vs number of cores on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells. The sparse matrix storage format used is ELLPACK[2].



[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 1:

Single-node performance of SpMV kernel vs number of cores on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells. The sparse matrix storage format used is ELLPACK[2].

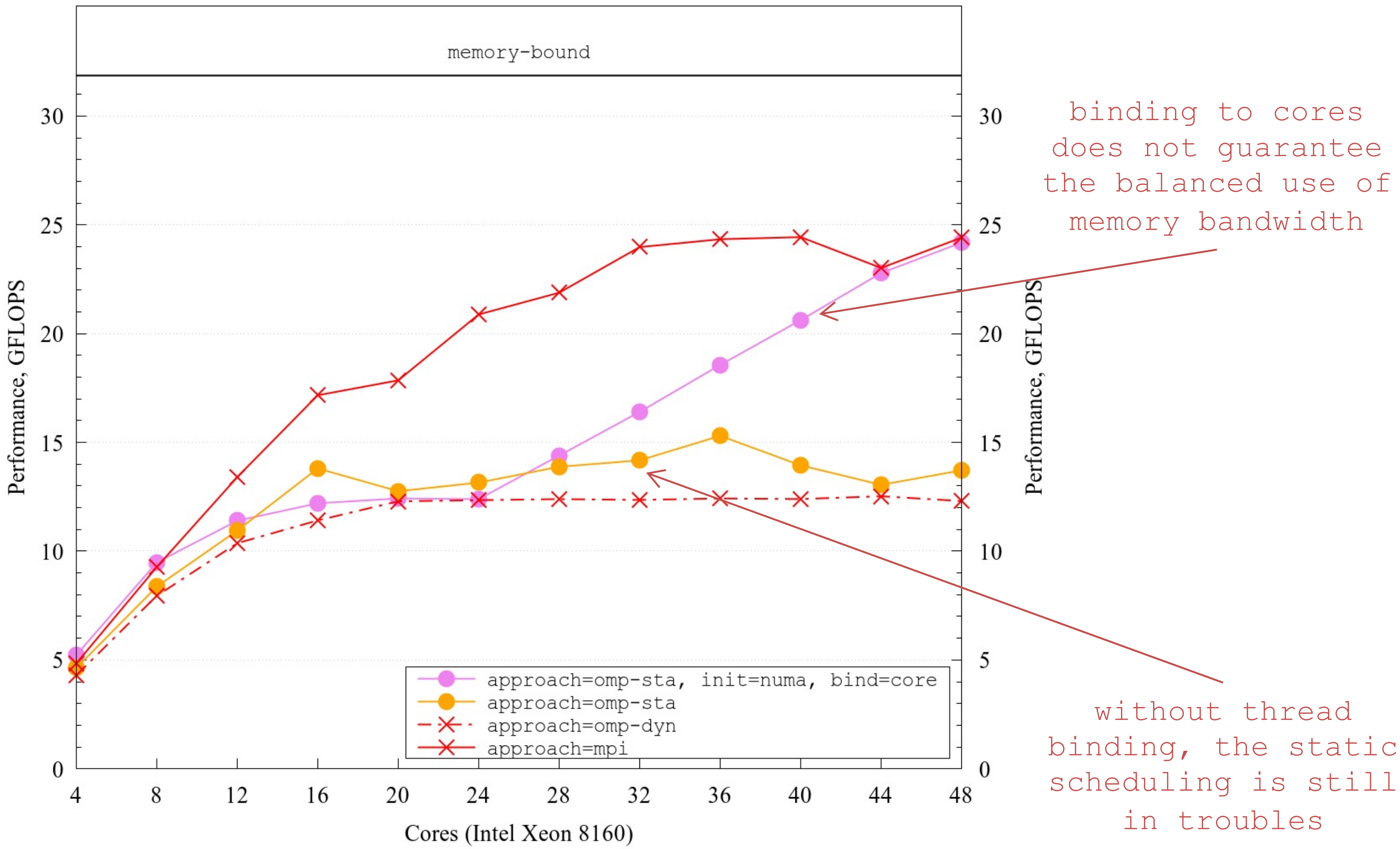


without thread binding, the static scheduling is still in troubles

[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 1:

Single-node performance of SpMV kernel vs number of cores on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells. The sparse matrix storage format used is ELLPACK[2].

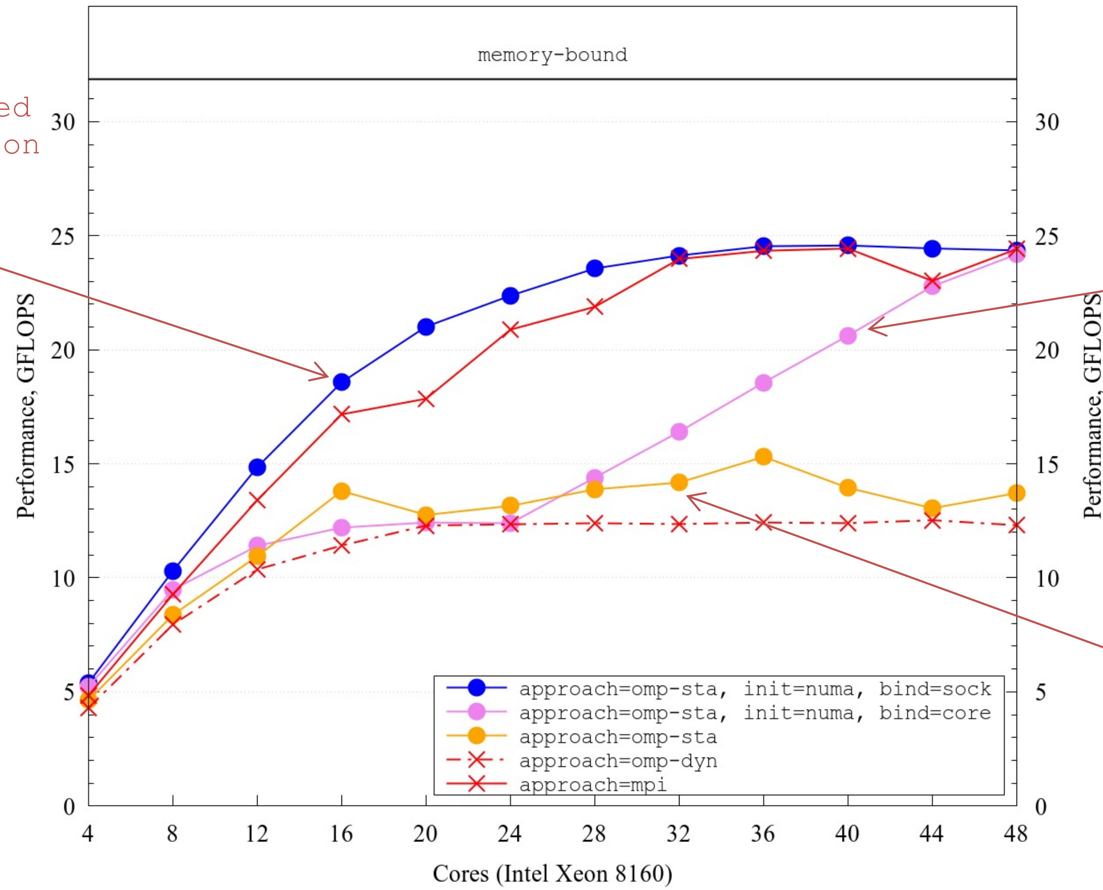


[1] F. Xavier Trias et al. “Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids”, JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. “Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers“, IJCFD 31 (2017), 396–411.

Test case 1:

Single-node performance of SpMV kernel vs number of cores on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells. The sparse matrix storage format used is ELLPACK[2].

thread binding to sockets and threaded memory initialization is our choice



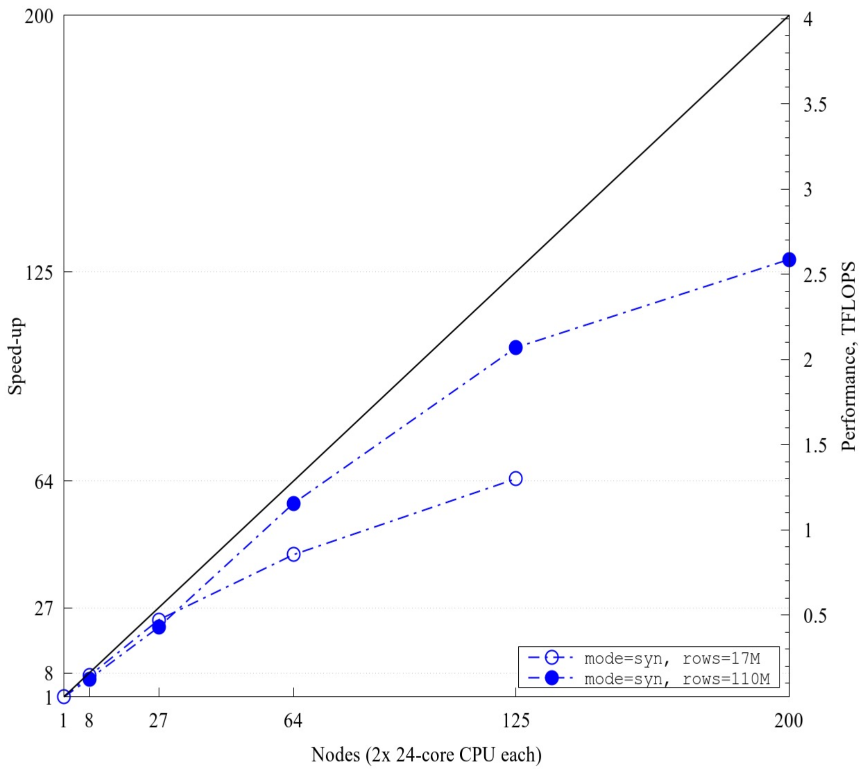
binding to cores does not guarantee the balanced use of memory bandwidth

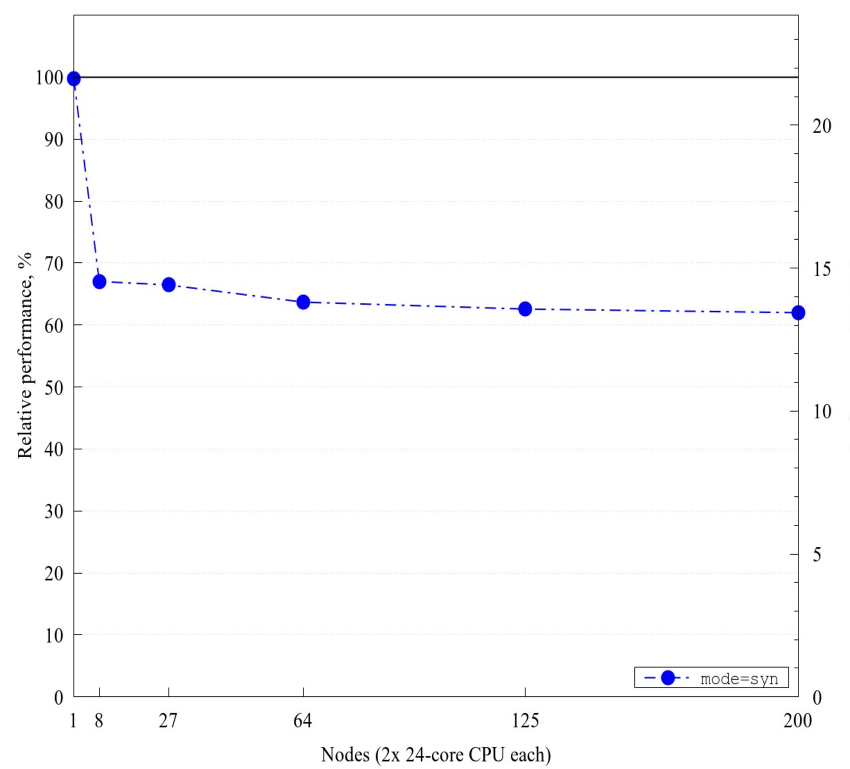
without thread binding, the static scheduling is still in troubles

[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 2:

Multi-node strong (left) and weak (right) scaling of SpMV kernel on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells (results for 110 million cells are also reported in strong scaling). The sparse matrix storage format used is ELLPACK[2].

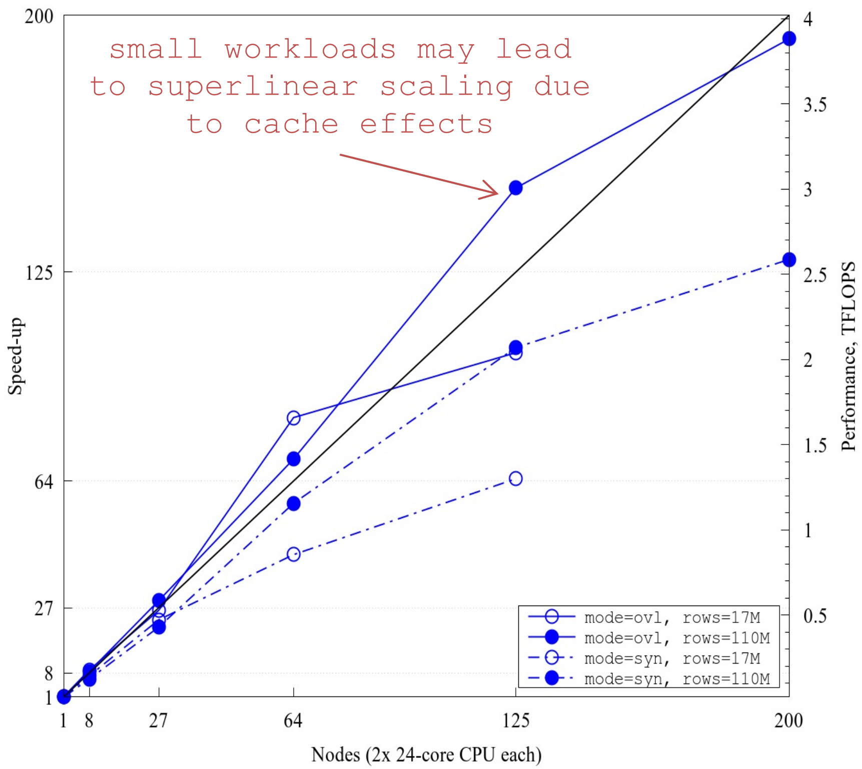


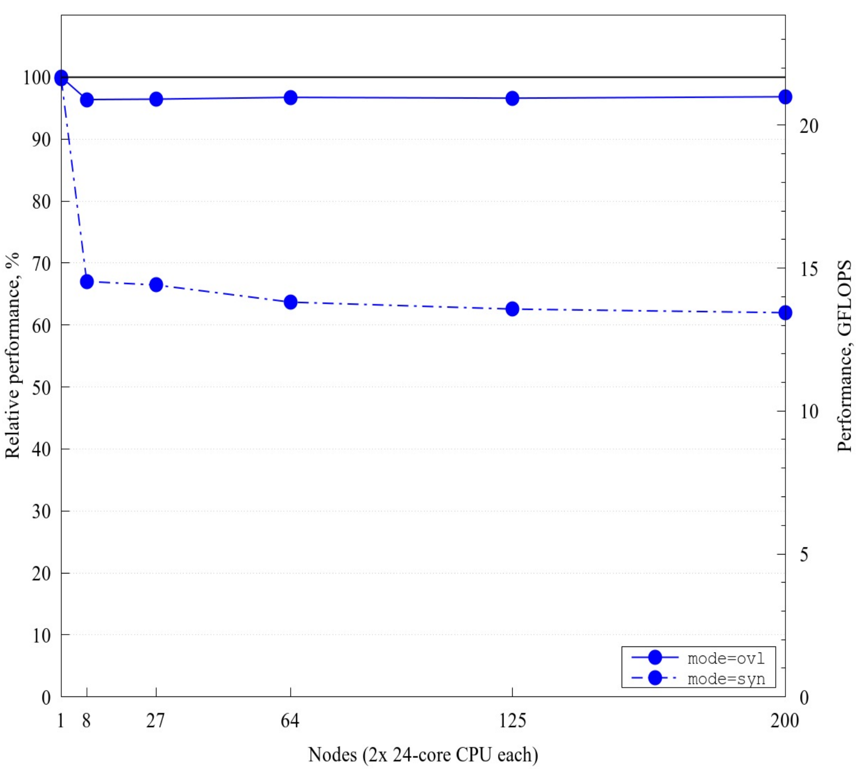


[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 2:

Multi-node strong (left) and weak (right) scaling of SpMV kernel on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells (results for 110 million cells are also reported in strong scaling). The sparse matrix storage format used is ELLPACK[2].



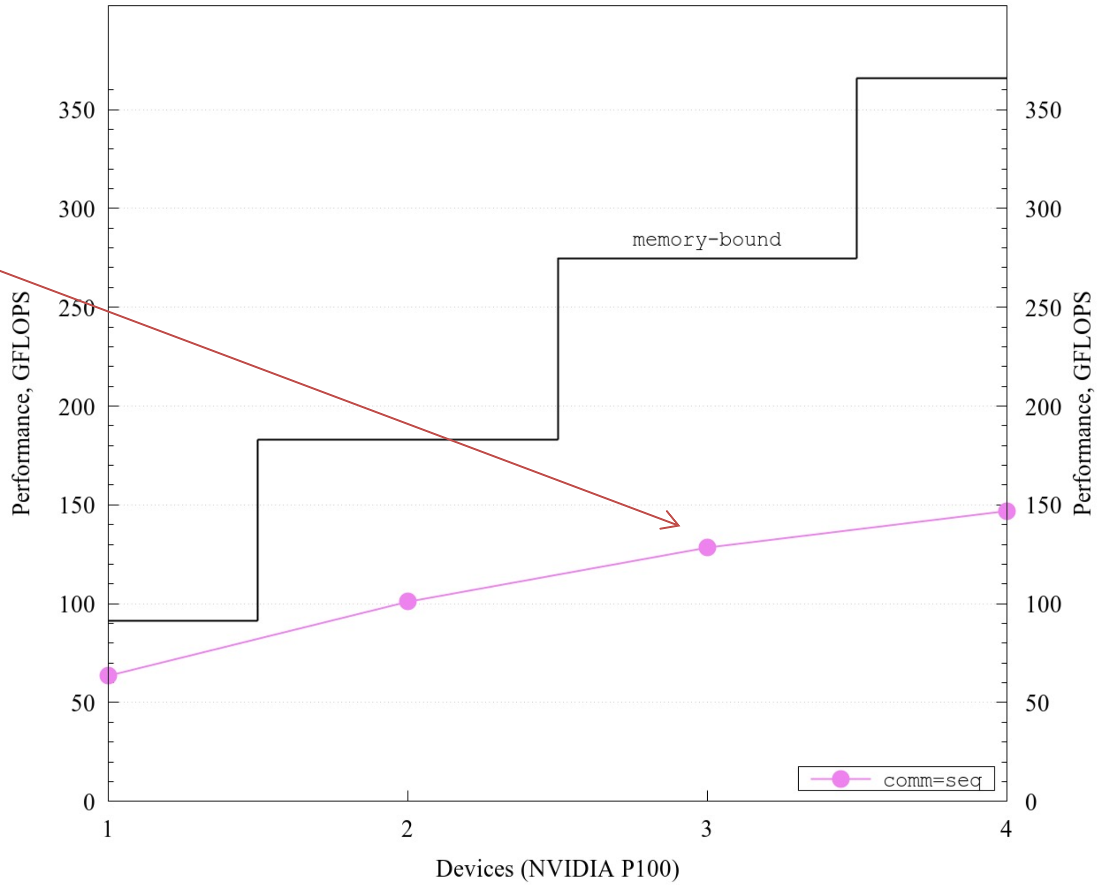


[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 4:

Single-node performance of SpMV kernel vs number of cores on TSUBAME3.0 for both sequential and parallel management of communications. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells. The sparse matrix storage format used is block-transposed ELLPACK[2].

managing data exchanges with a single thread is not enough

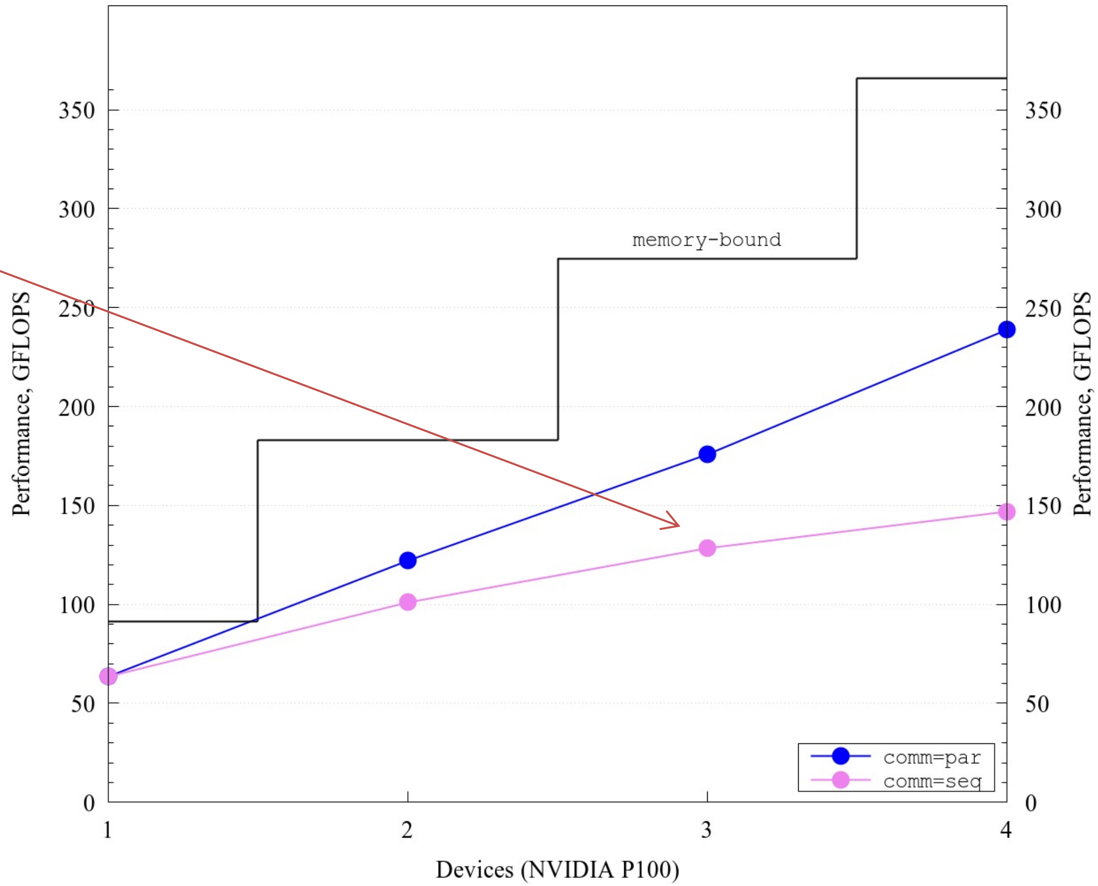


[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 4:

Single-node performance of SpMV kernel vs number of cores on TSUBAME3.0 for both sequential and parallel management of communications. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells. The sparse matrix storage format used is block-transposed ELLPACK[2].

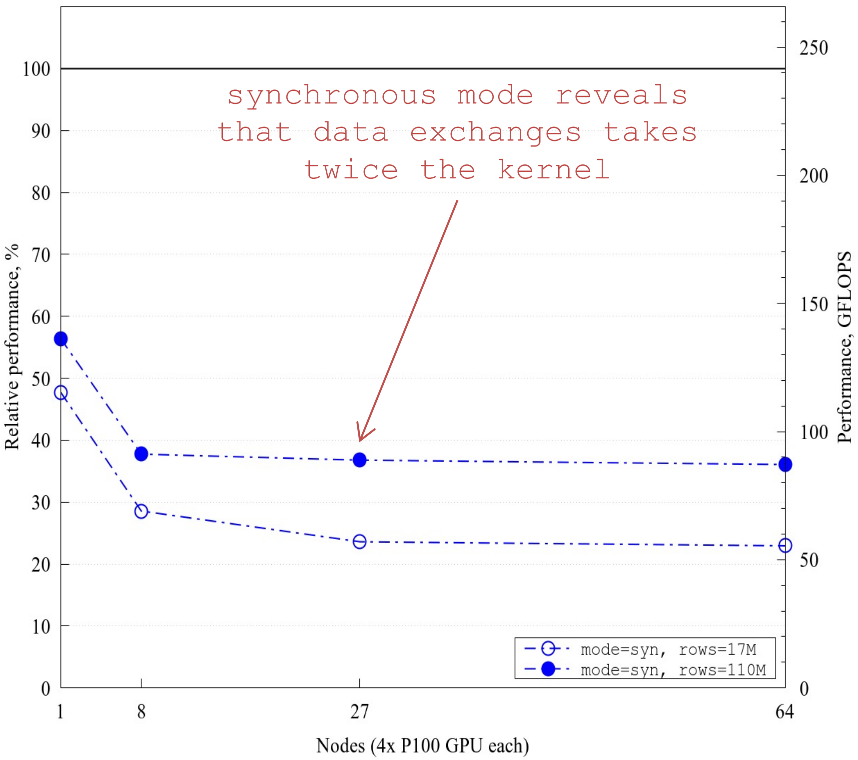
managing data exchanges with a single thread is not enough



[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 5:

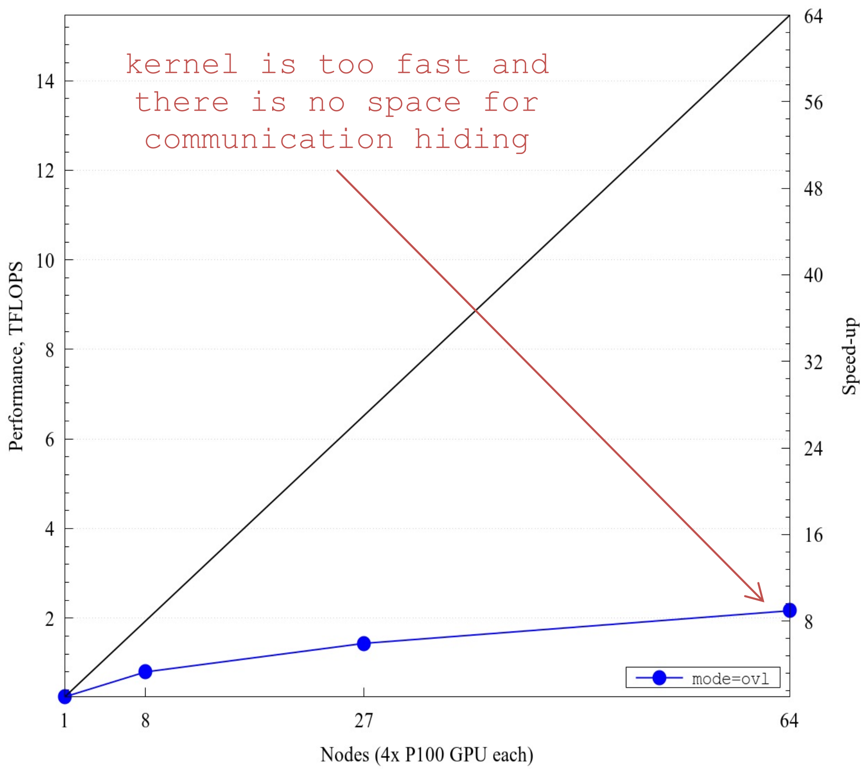
Multi-node strong (left) and weak (right) scaling of SpMV kernel on TSUBAME3.0. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 110 million cells (results for 17 million cells are also reported in weak scaling). The sparse matrix storage format used is the block-transposed ELLPACK[2].

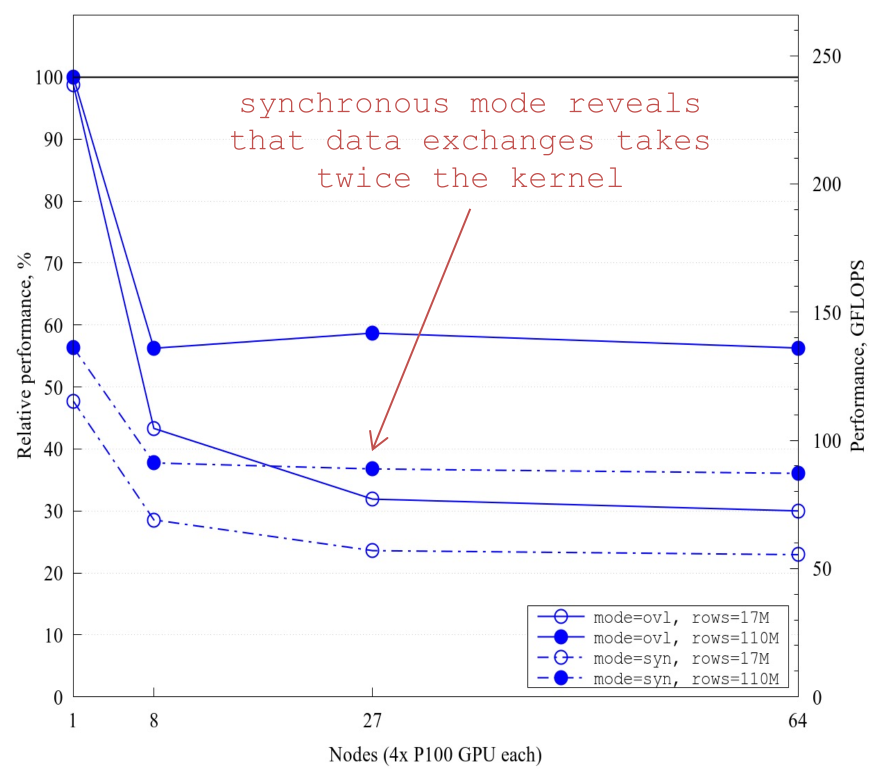


[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

Test case 5:

Multi-node strong (left) and weak (right) scaling of SpMV kernel on TSUBAME3.0. The sparse matrix used arise from the symmetry-preserving discretization[1] of the Laplacian operator on unstructured hex-dominant mesh of 110 million cells (results for 17 million cells are also reported in weak scaling). The sparse matrix storage format used is the block-transposed ELLPACK[2].





[1] F. Xavier Trias et al. "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids", JCP 258 (2014), 246–267.
[2] G. Oyarzun et al. "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers", IJCFD 31 (2017), 396–411.

CONCLUSIONS

Conclusions

i. An **algebra-based** approach has been presented as a naturally portable strategy for implementing numerical simulation codes.

Future work

Conclusions

i. An **algebra-based** approach has been presented as a naturally portable strategy for implementing numerical simulation codes.

ii. A **hierarchical parallel** implementation of the SpMV has been detailed, and its performance evaluated on various HPC systems.

Future work

Conclusions

i. An **algebra-based** approach has been presented as a naturally portable strategy for implementing numerical simulation codes.

ii. A **hierarchical parallel** implementation of the SpMV has been detailed, and its performance evaluated on various HPC systems.

iii. A multithreaded, **NUMA-aware** strategy has been described, and its efficiency on multi-CPU architectures has been proven.

Future work

Conclusions

i. An **algebra-based** approach has been presented as a naturally portable strategy for implementing numerical simulation codes.

ii. A **hierarchical parallel** implementation of the SpMV has been detailed, and its performance evaluated on various HPC systems.

iii. A multithreaded, **NUMA-aware** strategy has been described, and its efficiency on multi-CPU architectures has been proven.

iv. HPC systems with extremely high ratios of memory bandwidth to network bandwidth are harmful for computationaly light, memory bound kernels such as SpMV; the calculations become too fast to hide the communications with naive schemes.

Future work

Conclusions

i. An **algebra-based** approach has been presented as a naturally portable strategy for implementing numerical simulation codes.

ii. A **hierarchical parallel** implementation of the SpMV has been detailed, and its performance evaluated on various HPC systems.

iii. A multithreaded, **NUMA-aware** strategy has been described, and its efficiency on multi-CPU architectures has been proven.

iv. HPC systems with extremely high ratios of memory bandwidth to network bandwidth are harmful for computationaly light, memory bound kernels such as SpMV; the calculations become too fast to hide the communications with naive schemes.

Future work

- To design a new update mechanism to accelerate the data exchanges, for instance, taking into account NUMA factor in inter- and intra-node exchanges.

Thank you for attending