On the benefits and applications of sparse matrix-matrix product on various parallel architectures

Xavier Álvarez-Farré[1], Àdel Alsalti-Baldellou[1,2], Andrey Gorobets[3], Assensi Oliva[1], F. Xavier Trias[1]

In the 33rd International Conference on Parallel Computational Fluid Dynamics, May 25th–27th, 2022, Alba, Italy

[1]Heat and Mass Transfer Technological Center, **Technical University of Catalonia** (BarcelonaTech)
[2]**TermoFluids S.L.**
[3]Who cares? e-mail: andrey.gorobets@gmail.com

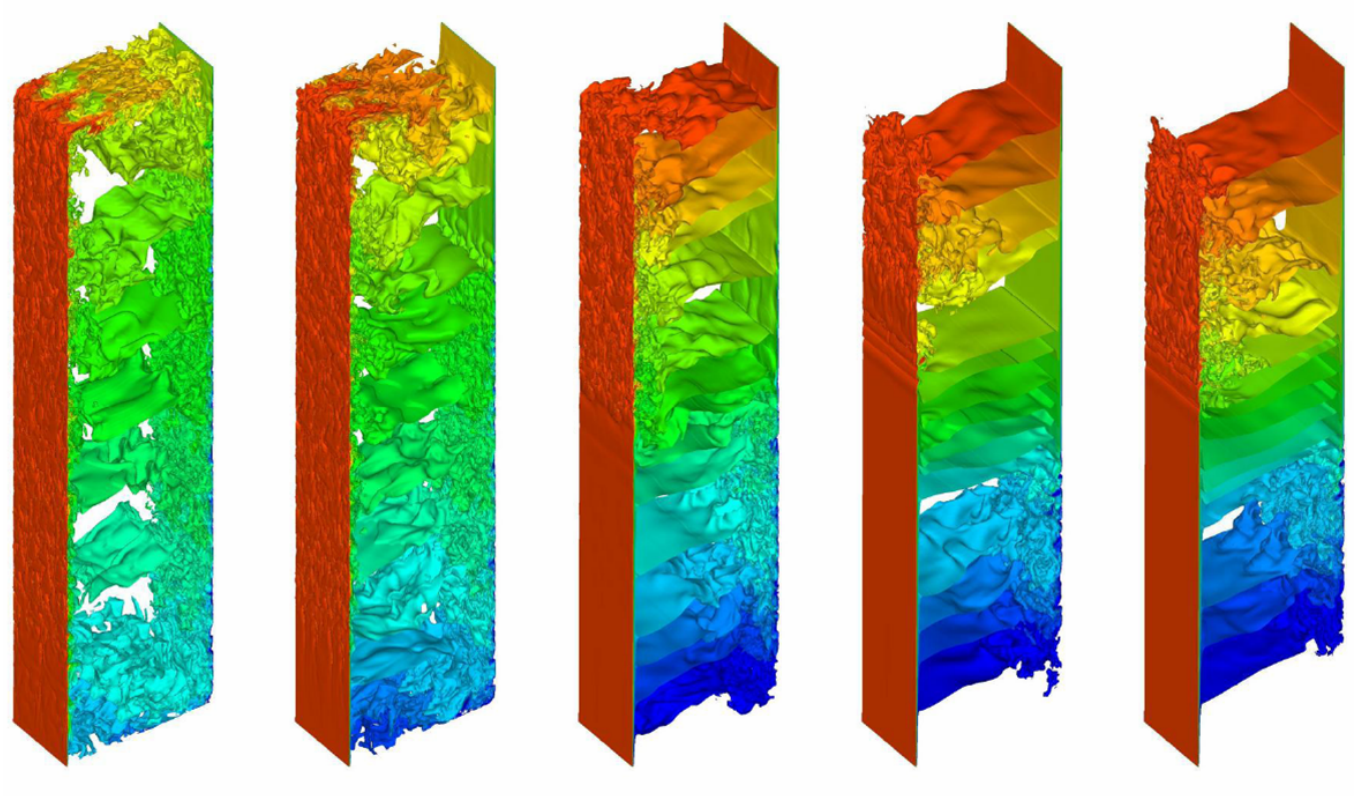
Centre Tecnològic de Transferència de Calor
UNIVERSITAT POLITÈCNICA DE CATALUNYA

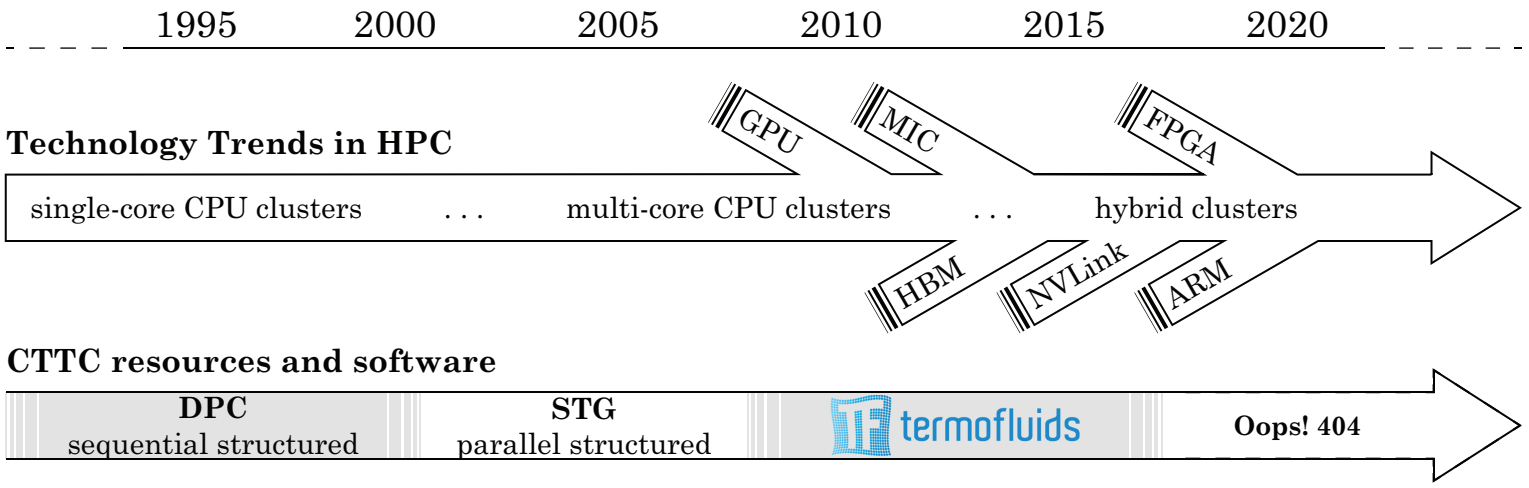
Parallel Computational Fluid Dynamics

Background

Research at CTTC Laboratory

The **Heat and Mass Transfer Technological Center** (CTTC) 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.

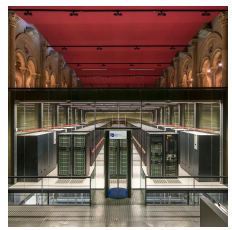


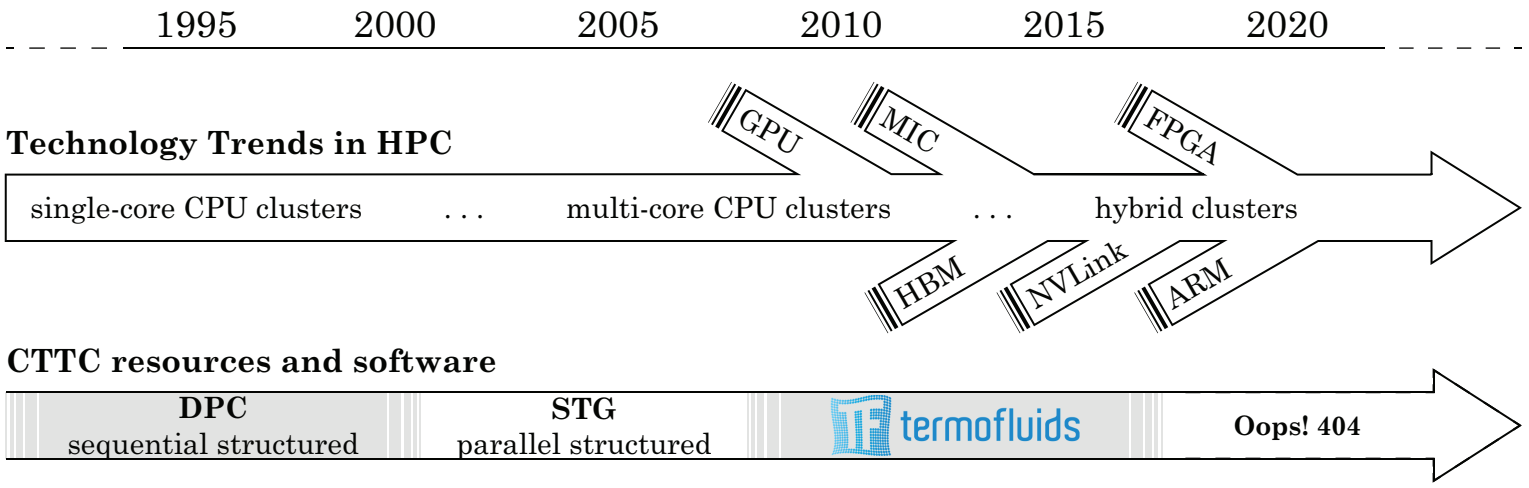


The evolution in hardware technologies

enables scientific computing to advance incessantly and reach further aims. Nowadays, the use of HPC systems is rather common on the solution of both industrial and academic scale problems.



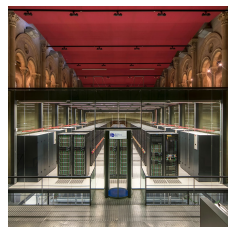



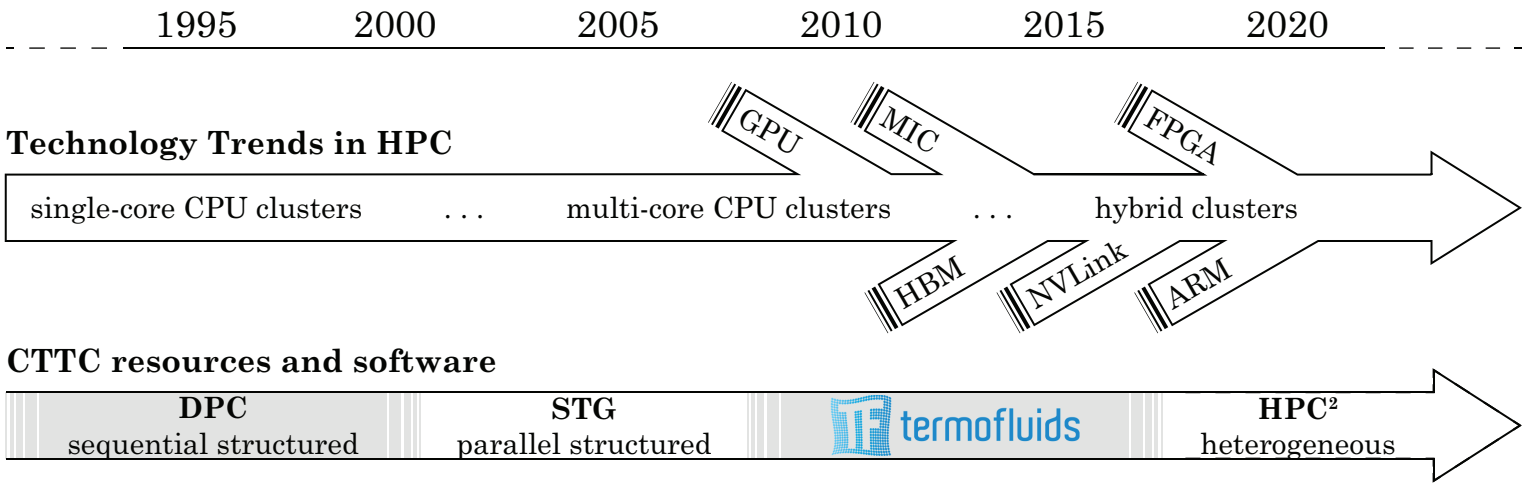



Since the beginning,

researchers of CTTC is devoted to develop and adapt CFD codes for the state-of-the art computer resources, from sequential structured to parallel unstructured applications.

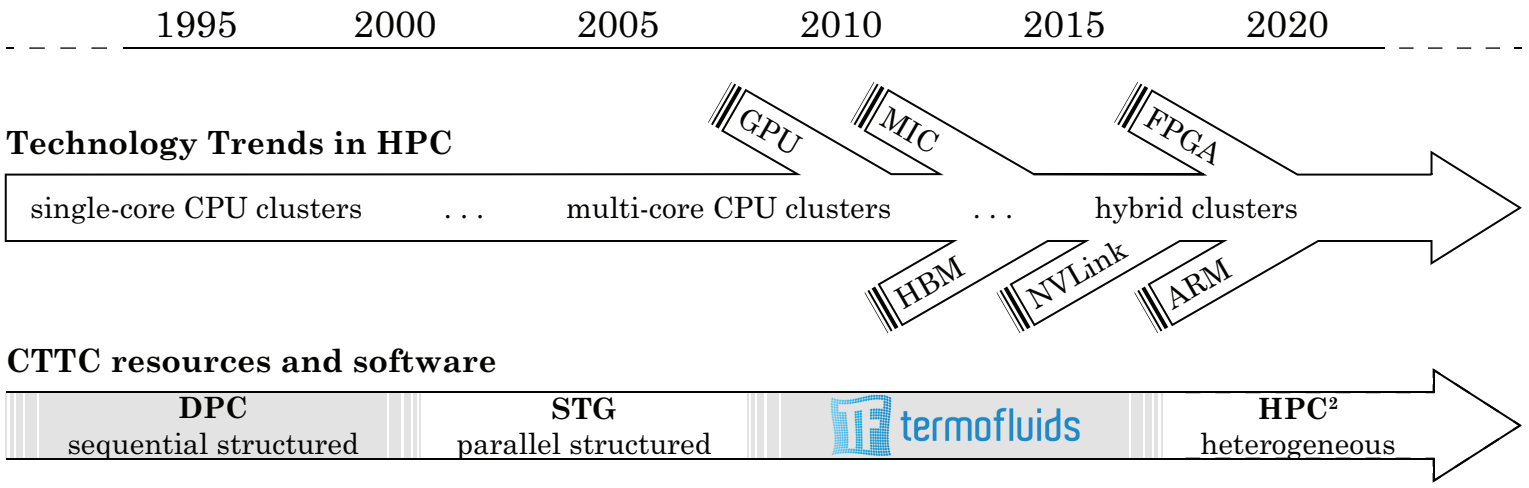


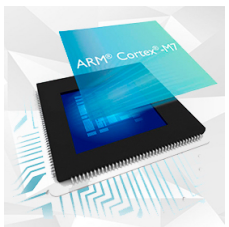
Massively-parallel devices

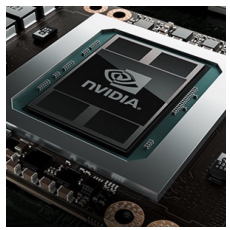
of various architectures are incorporated into modern supercomputers, causing the hybridisation of HPC systems and making the design of computing applications a rather complex problem: the kernels conforming the algorithms must be compatible with distributed- and shared-memory SIMD and MIMD parallelism, and stream processing.



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.



Is it necessary to use the new hardware architectures?

- In our opinion, yes. New hardware is designed to increase energy efficiency, an imperative to overcome the power constraints in the context of the exascale challenge.

Is it necessary to use the new hardware architectures?

- In our opinion, yes. New hardware is designed to increase energy efficiency, an imperative to overcome the power constraints in the context of the exascale challenge.

Do the traditional implementation models facilitate code portability?

- In our opinion, no. Legacy codes were not designed portable simply because it was not necessary before; these codes usually contain a large number of complex kernels and data structures mostly suitable for CPU architectures.

Is it necessary to use the new hardware architectures?

- In our opinion, yes. New hardware is designed to increase energy efficiency, an imperative to overcome the power constraints in the context of the exascale challenge.

Do the traditional implementation models facilitate code portability?

- In our opinion, no. Legacy codes were not designed portable simply because it was not necessary before; these codes usually contain a large number of complex kernels and data structures mostly suitable for CPU architectures.

Do we need to change the way we look at scientific computing in general?

- In our opinion, yes. There is a large variety of hardware architectures and it is difficult to determine which are going to prevail. Therefore, sustainability and portability should become the center of scientific computing software design.

The algebraic approach

Stencil

Traditionally, the development of scientific computing software is based on calculations in iterative stencil loops over a discretized geometry—the mesh. Despite being intuitive and versatile, the interdependency between algorithms and their computational implementations in stencil applications usually introduces an inevitable complexity when it comes to portability and sustainability.

Algebraic

By casting discrete operators and mesh functions into sparse matrices and vectors, it has been shown that all the calculations in a typical CFD algorithm for the DNS and LES of incompressible turbulent flows boil down to a minimalist set of algebraic subroutines.

The idea is to use the stencils just for building data and leave the calculations to an algebraic framework; thus, legacy codes may be maintained indefinitely as preprocessing tools, and the calculation engines become easy to port and optimize.

Continuous, dimensionless Navier–Stokes equations read:

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

[1] Trias et al., Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids, J.Comp.Phys., 258, 246-267, 2014.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh[1]:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

[1] Trias et al., Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids, J.Comp.Phys., 258, 246-267, 2014.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh[1]:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

where:

$$\text{cells} \in \mathbb{R}^n \quad \text{faces} \in \mathbb{R}^m$$

[1] Trias et al., Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids, J.Comp.Phys., 258, 246-267, 2014.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh[1]:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

where:

$$\text{cells} \in \mathbb{R}^n \quad \text{faces} \in \mathbb{R}^m$$

$$\boldsymbol{p}_c \in \mathbb{R}^n \quad \boldsymbol{u}_c \in \mathbb{R}^{3n} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3)^T \quad \boldsymbol{u}_s \in \mathbb{R}^m$$

[1] Trias et al., Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids, J.Comp.Phys., 258, 246-267, 2014.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh[1]:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

where:

$$\text{cells} \in \mathbb{R}^n \quad \text{faces} \in \mathbb{R}^m$$

$$\boldsymbol{p}_c \in \mathbb{R}^n \quad \boldsymbol{u}_c \in \mathbb{R}^{3n} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3)^T \quad \boldsymbol{u}_s \in \mathbb{R}^m$$

$$\mathsf{G}_c \in \mathbb{R}^{3n \times n} \quad \mathsf{M} \in \mathbb{R}^{n \times m}$$

[1] Trias et al., Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids, J.Comp.Phys., 258, 246-267, 2014.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh[1]:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

where:

$$\text{cells} \in \mathbb{R}^n \quad \text{faces} \in \mathbb{R}^m$$

$$\boldsymbol{p}_c \in \mathbb{R}^n \quad \boldsymbol{u}_c \in \mathbb{R}^{3n} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3)^T \quad \boldsymbol{u}_s \in \mathbb{R}^m$$

$$\mathsf{G}_c \in \mathbb{R}^{3n \times n} \quad \mathsf{M} \in \mathbb{R}^{n \times m}$$

$$\Omega_c^{3d} \in \mathbb{R}^{3n \times 3n} = \mathsf{I}_3 \otimes \Omega_c \quad \mathsf{C}_c^{3d}(\boldsymbol{u}_s) \in \mathbb{R}^{3n \times 3n} = \mathsf{I}_3 \otimes \mathsf{C}_c(\boldsymbol{u}_s) \quad \mathsf{D}_c^{3d} \in \mathbb{R}^{3n \times 3n} = \mathsf{I}_3 \otimes \mathsf{D}_c$$

[1] Trias et al., Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids, J.Comp.Phys., 258, 246-267, 2014.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

- Discrete mesh functions and operators stored in vectors and sparse matrices.

[2] Álvarez-Farré et al., A hierarchical parallel implementation for heterogeneous computing. Application to algebra-based CFD simulations on hybrid supercomputers, Computers & Fluids, 214, 104768, 2021.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

- Discrete mesh functions and operators stored in vectors and sparse matrices.
- Numerical method is fully integrated into data structures and BLAS can be used[2]:

SpMV axpy dot

[2] Álvarez-Farré et al., A hierarchical parallel implementation for heterogeneous computing. Application to algebra-based CFD simulations on hybrid supercomputers, Computers & Fluids, 214, 104768, 2021.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

- Discrete mesh functions and operators stored in vectors and sparse matrices.
- Numerical method is fully integrated into data structures and BLAS can be used[2]:

  SpMV axpy dot
- Computational implementation is independent of spatial discretization.

[2] Álvarez-Farré et al., A hierarchical parallel implementation for heterogeneous computing. Application to algebra-based CFD simulations on hybrid supercomputers, Computers & Fluids, 214, 104768, 2021.

Continuous, dimensionless Navier–Stokes equations read:

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

Finite-volume, algebra-based discretization on arbitrary collocated mesh:

$$\mathsf{M}\boldsymbol{u}_s = \mathbf{0}_c, \qquad \Omega_c^{3d} d_t \boldsymbol{u}_c + \mathsf{C}_c^{3d}(\boldsymbol{u}_s)\boldsymbol{u}_c + \mathsf{D}_c^{3d}\boldsymbol{u}_c - \Omega_c^{3d}\mathsf{G}_c\boldsymbol{p}_c = \mathbf{0}_c,$$

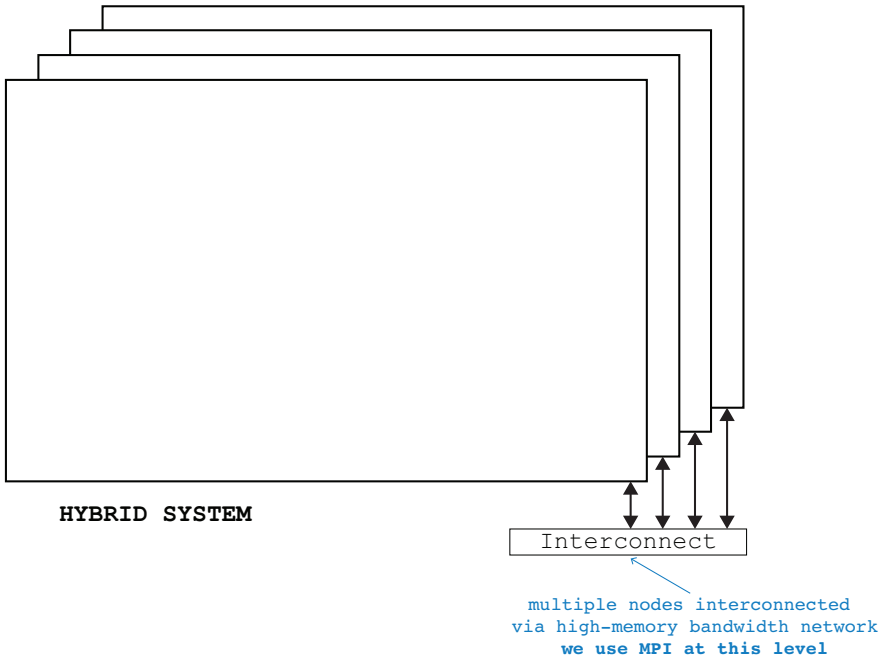
- Discrete mesh functions and operators stored in vectors and sparse matrices.
- Numerical method is fully integrated into data structures and BLAS can be used[2]:

  SpMV axpy dot
- Computational implementation is independent of spatial discretization.
- Discrete operators can be built to mimick the properties of the continuum operators.

[2] Álvarez-Farré et al., A hierarchical parallel implementation for heterogeneous computing. Application to algebra-based CFD simulations on hybrid supercomputers, Computers & Fluids, 214, 104768, 2021.

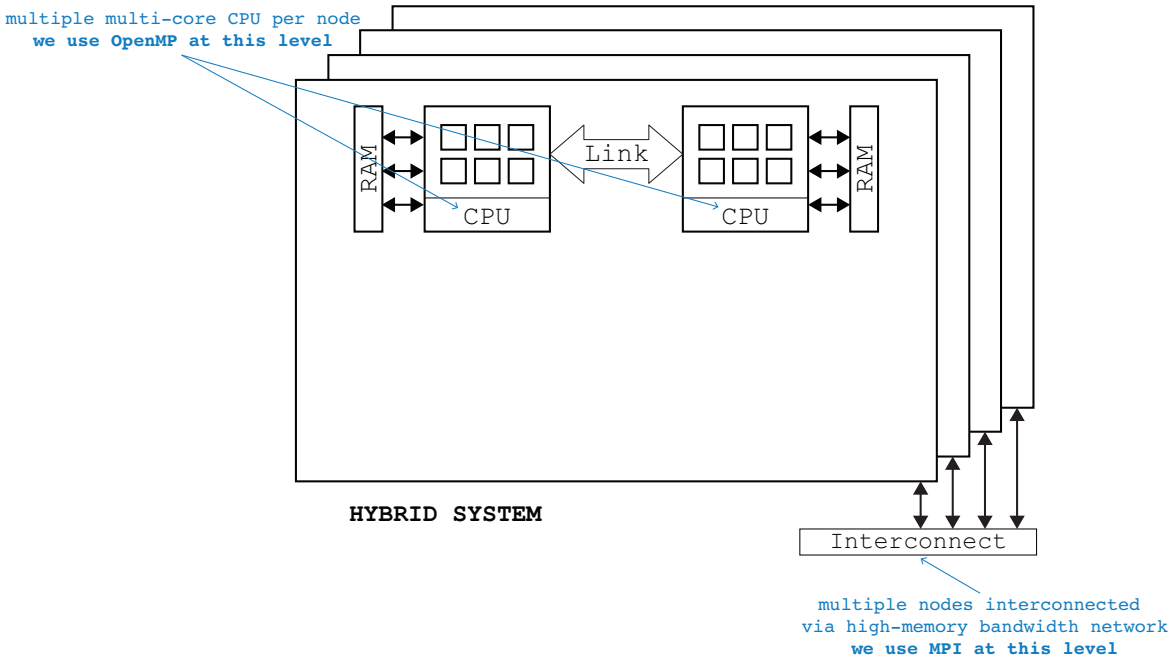
A hierarchical parallel implementation

Modern HPC systems consist of multiple hybrid computing nodes interconnected via a communication infrastructure. The nodes are composed of many hardware devices of different architectures, such as central processing unit (CPU) or graphics processing unit (GPU), among others.



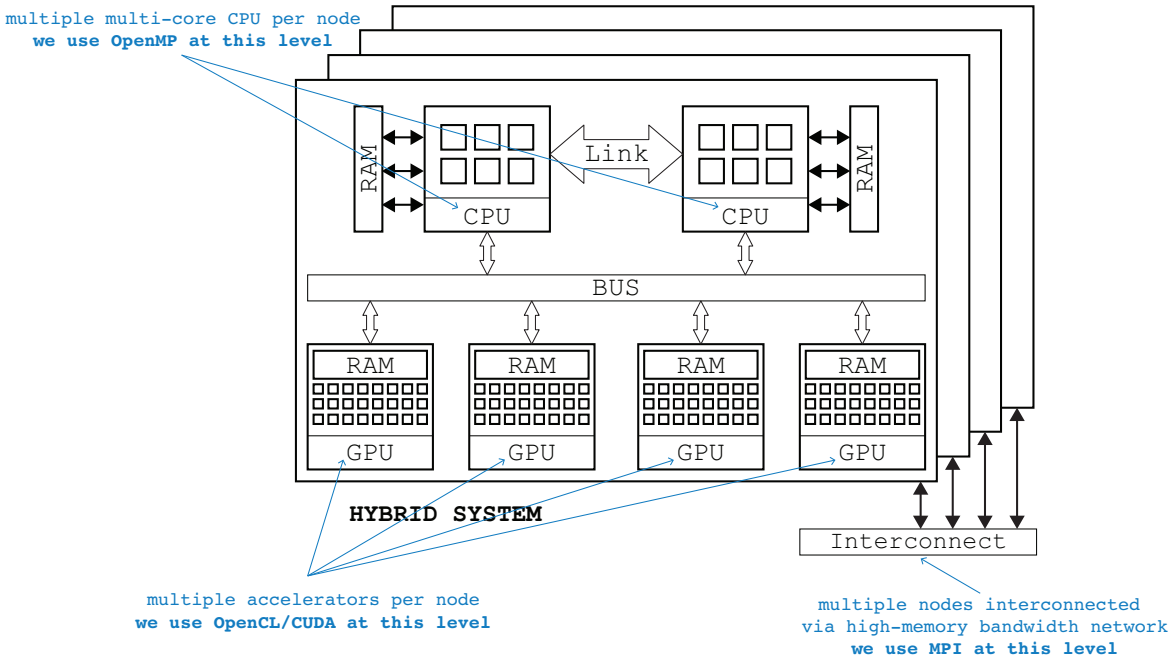
Hybrid HPC systems

Modern HPC systems consist of multiple hybrid computing nodes interconnected via a communication infrastructure. The nodes are composed of many hardware devices of different architectures, such as central processing unit (CPU) or graphics processing unit (GPU), among others.

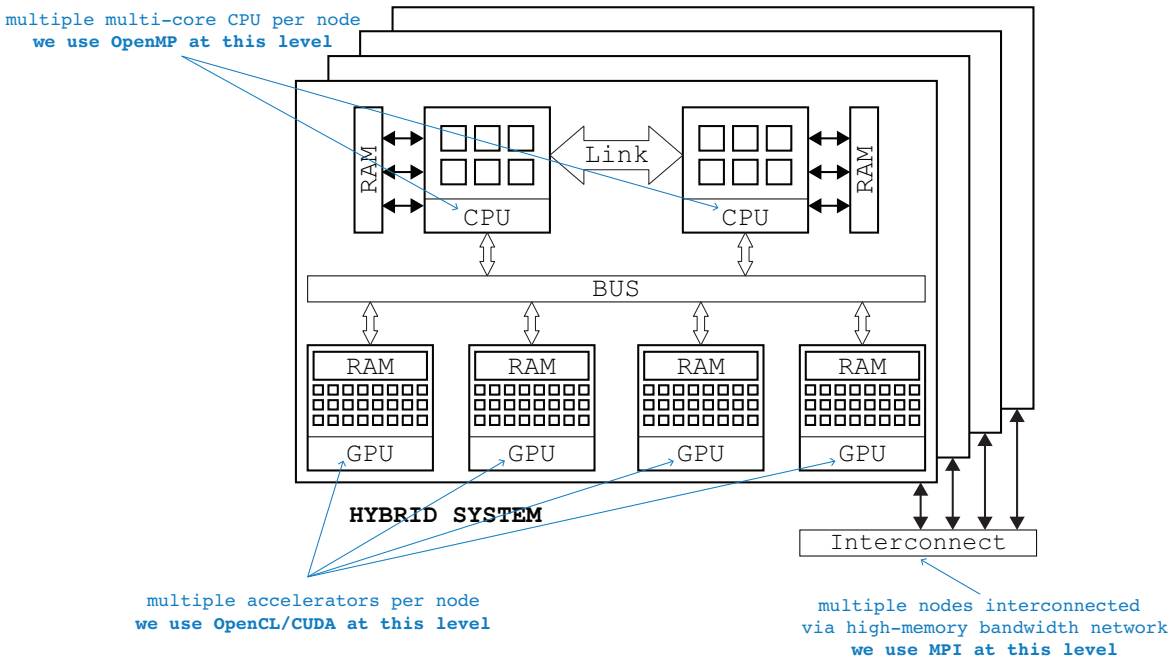


Hybrid HPC systems

Modern HPC systems consist of multiple hybrid computing nodes interconnected via a communication infrastructure. The nodes are composed of many hardware devices of different architectures, such as central processing unit (CPU) or graphics processing unit (GPU), among others.



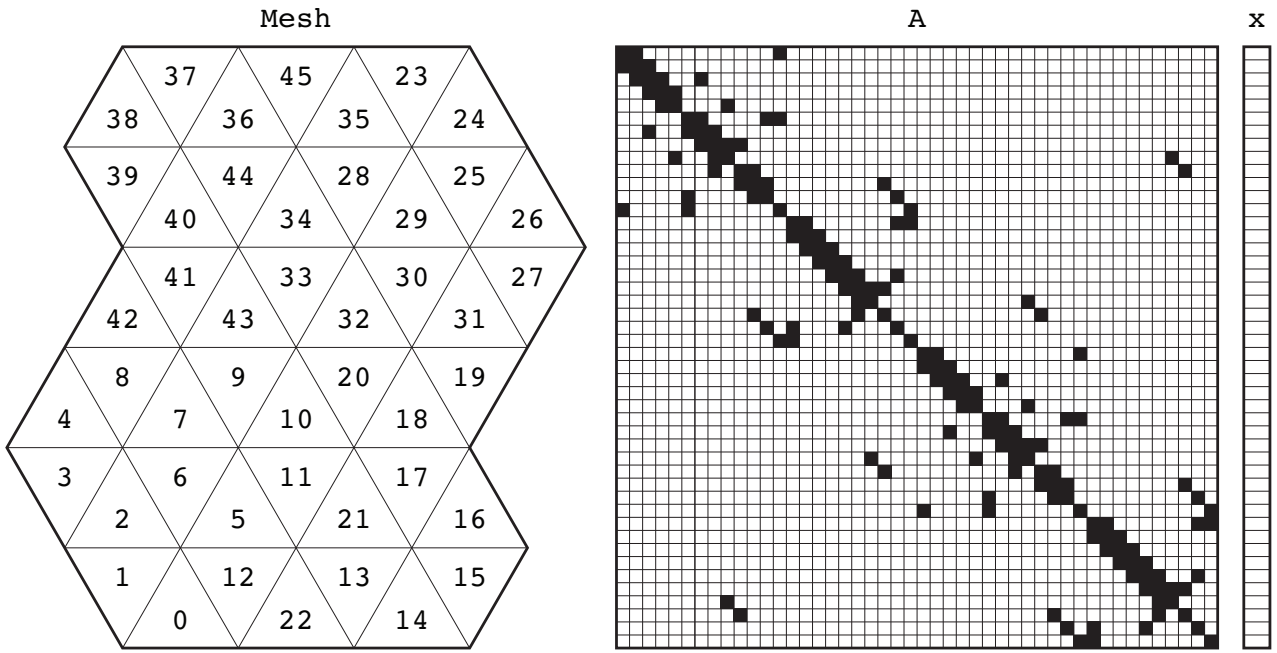
Modern HPC systems consist of multiple hybrid computing nodes interconnected via a communication infrastructure. The nodes are composed of many hardware devices of different architectures, such as central processing unit (CPU) or graphics processing unit (GPU), among others.



The algorithms must be compatible with distributed- and shared-memory multiple instruction, multiple data (DMMIMD and SMMIMD, respectively) parallelism, and more importantly, with stream processing (SP).

Multilevel domain decomposition

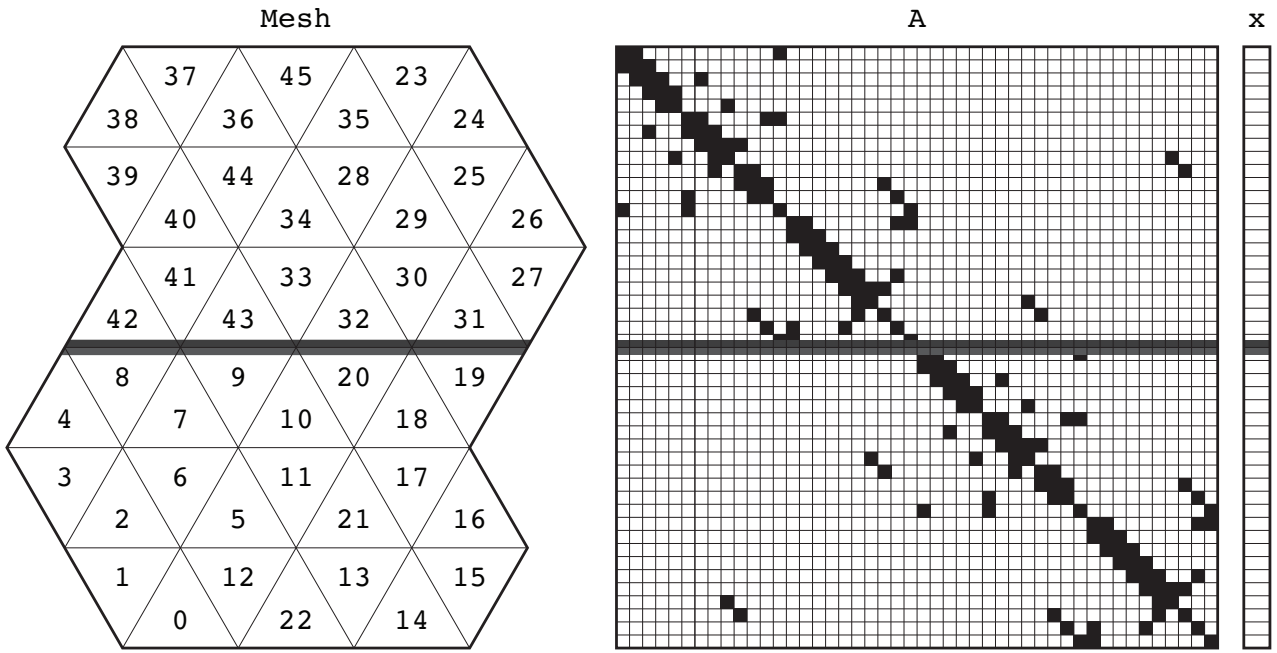
Multilevel workload distribution consists of dividing the computational domain (mesh) into subsets recursively to distribute it among the hardware of a computing system.



Multilevel domain decomposition

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

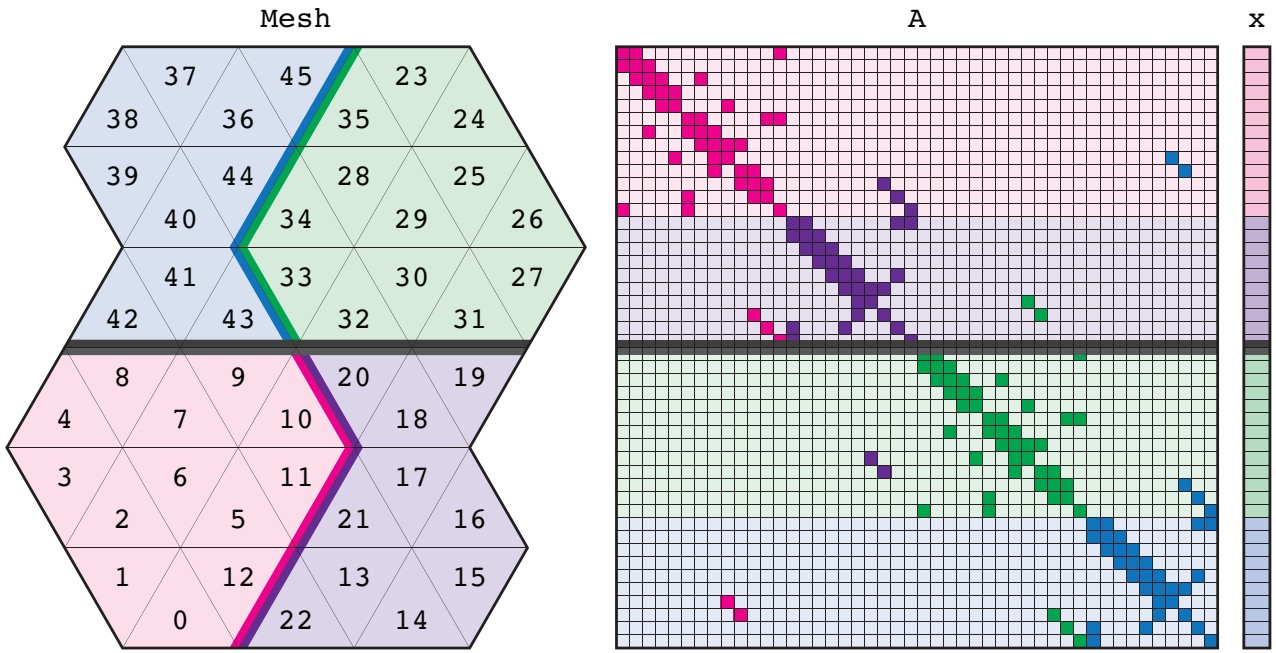
- First-level among computing nodes, *i.e.*, MPI processes.



Multilevel domain decomposition

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

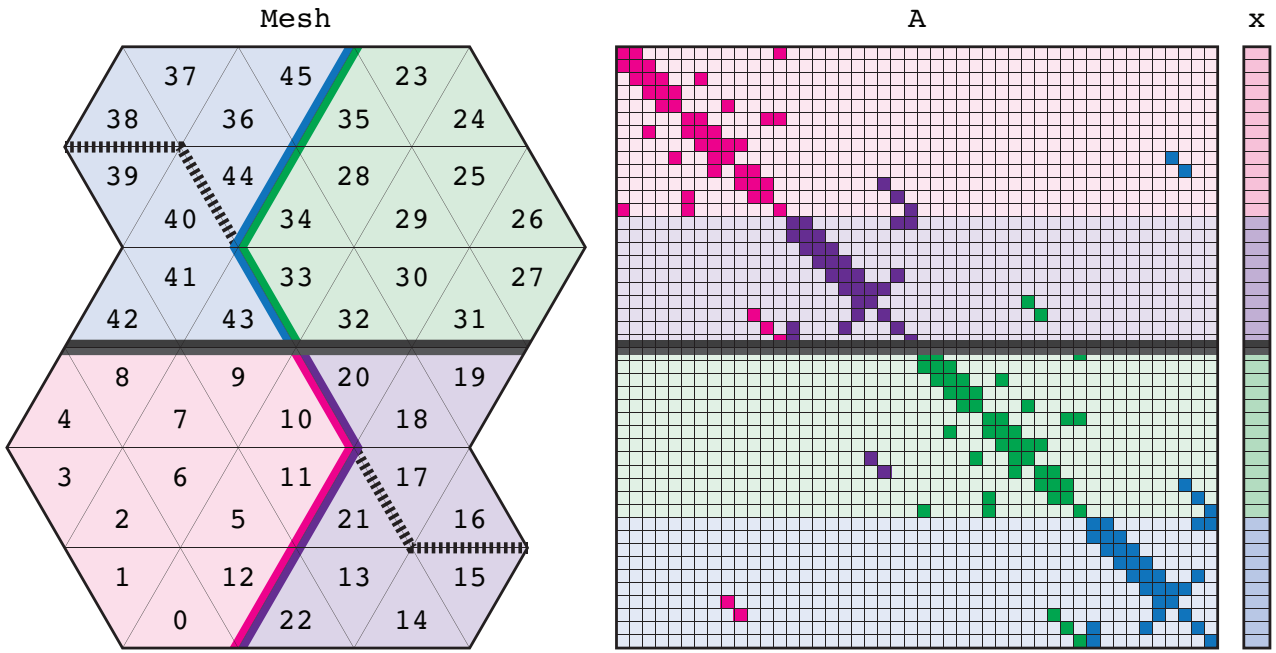
- First-level among computing nodes, *i.e.*, MPI processes.
- Second-level among computing units, *i.e.*, host and accelerators.



Multilevel domain decomposition

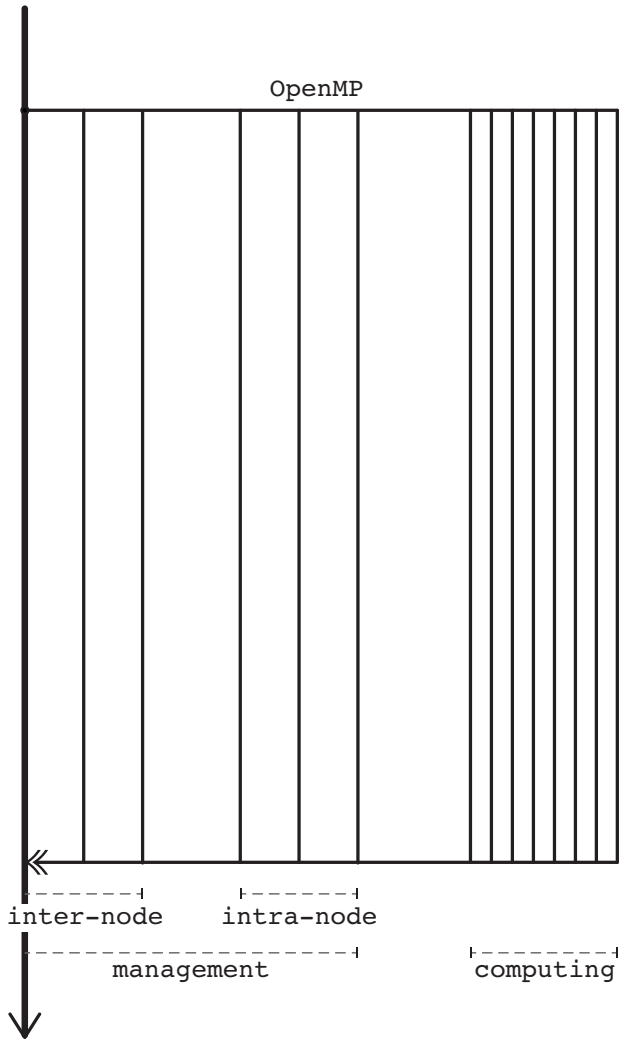
Multilevel workload distribution consists of dividing the computational domain (mesh) into subsets recursively to distribute it among the hardware of a computing system.

- First-level among computing nodes, *i.e.*, MPI processes.
- Second-level among computing units, *i.e.*, host and accelerators.
- Third-level among threads in NUMA shared-memory spaces.



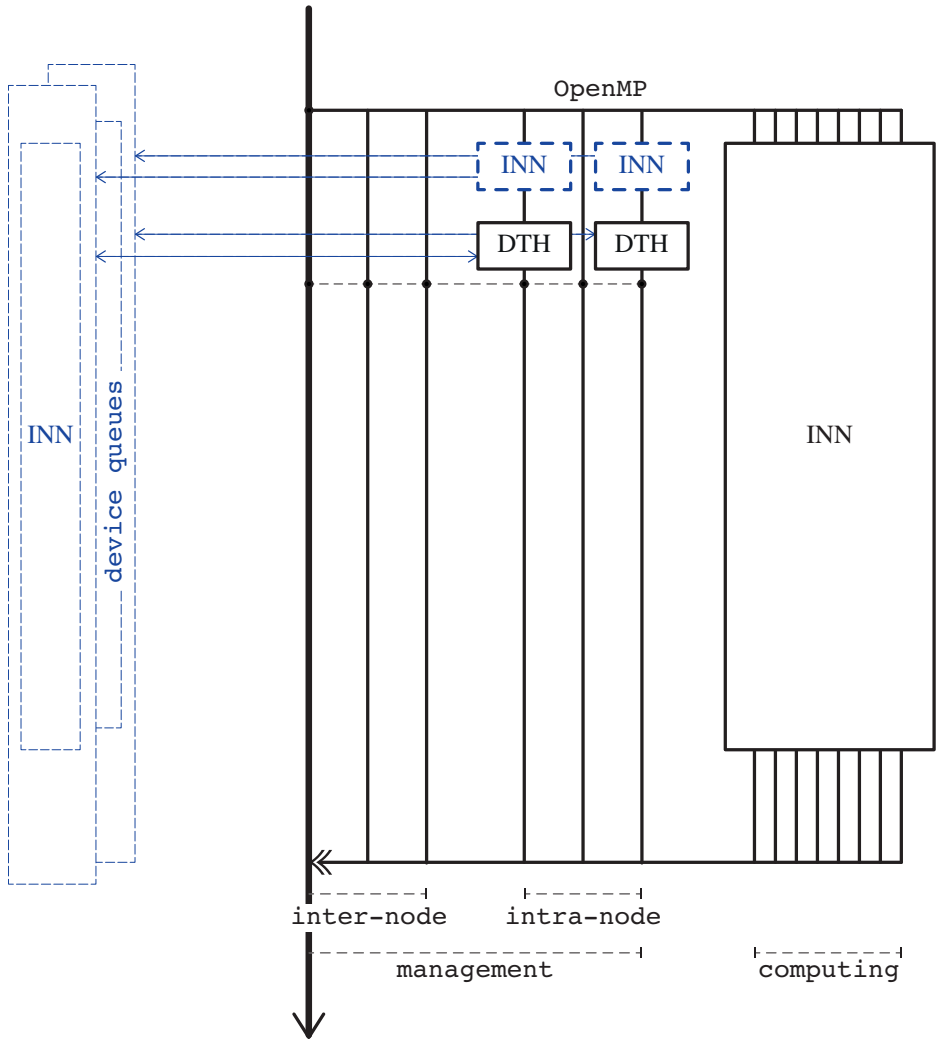
Flat multithreaded execution strategies

To minimise the overhead of the communications, efficient multithreaded execution strategies are required. Roughly, the idea is to overlap the communications with the computations.



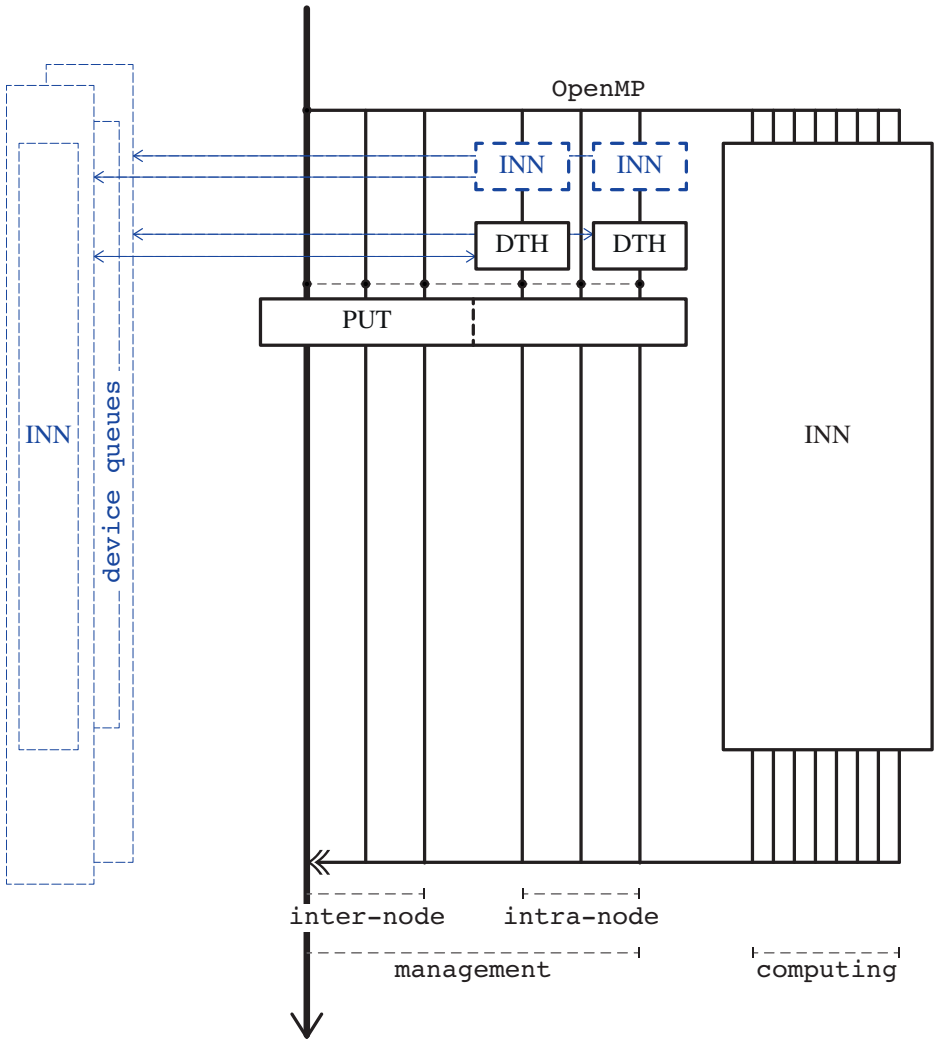
Flat multithreaded execution strategies

To minimise the overhead of the communications, efficient multithreaded execution strategies are required. Roughly, the idea is to overlap the communications with the computations.

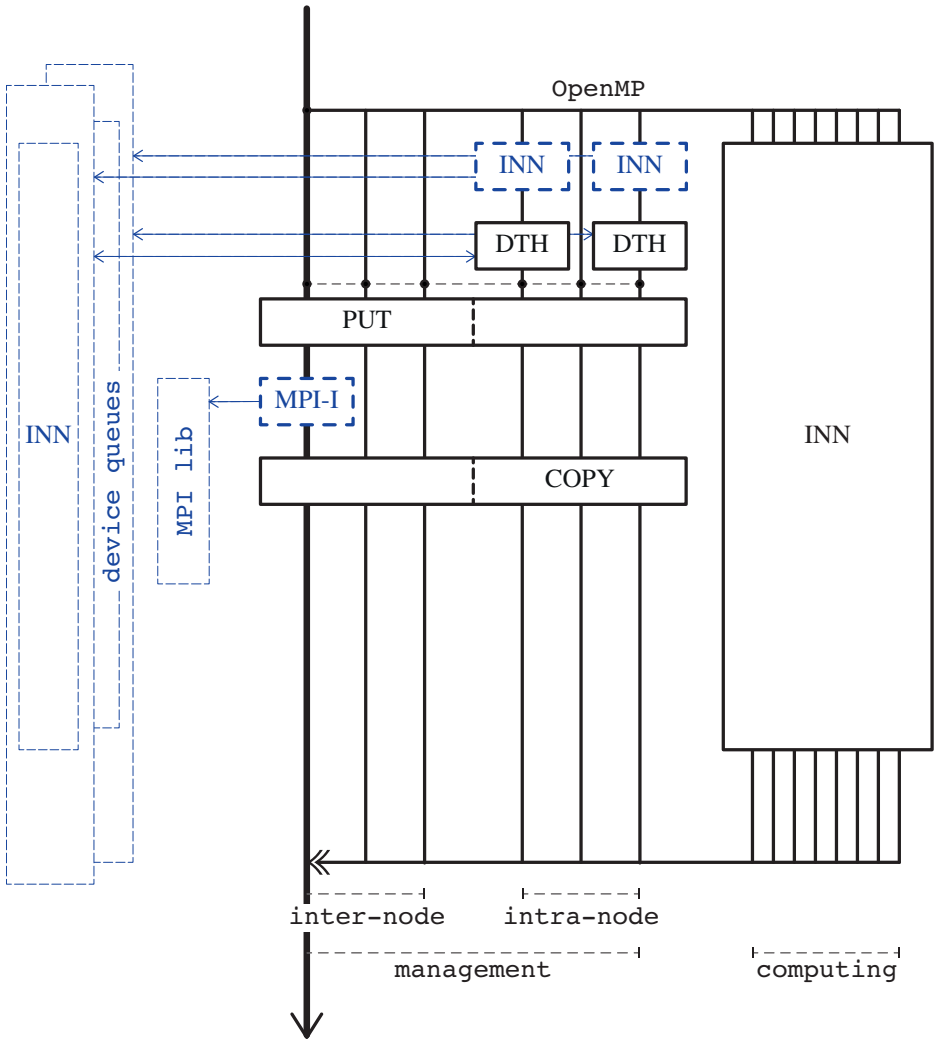


Flat multithreaded execution strategies

To minimise the overhead of the communications, efficient multithreaded execution strategies are required. Roughly, the idea is to overlap the communications with the computations.

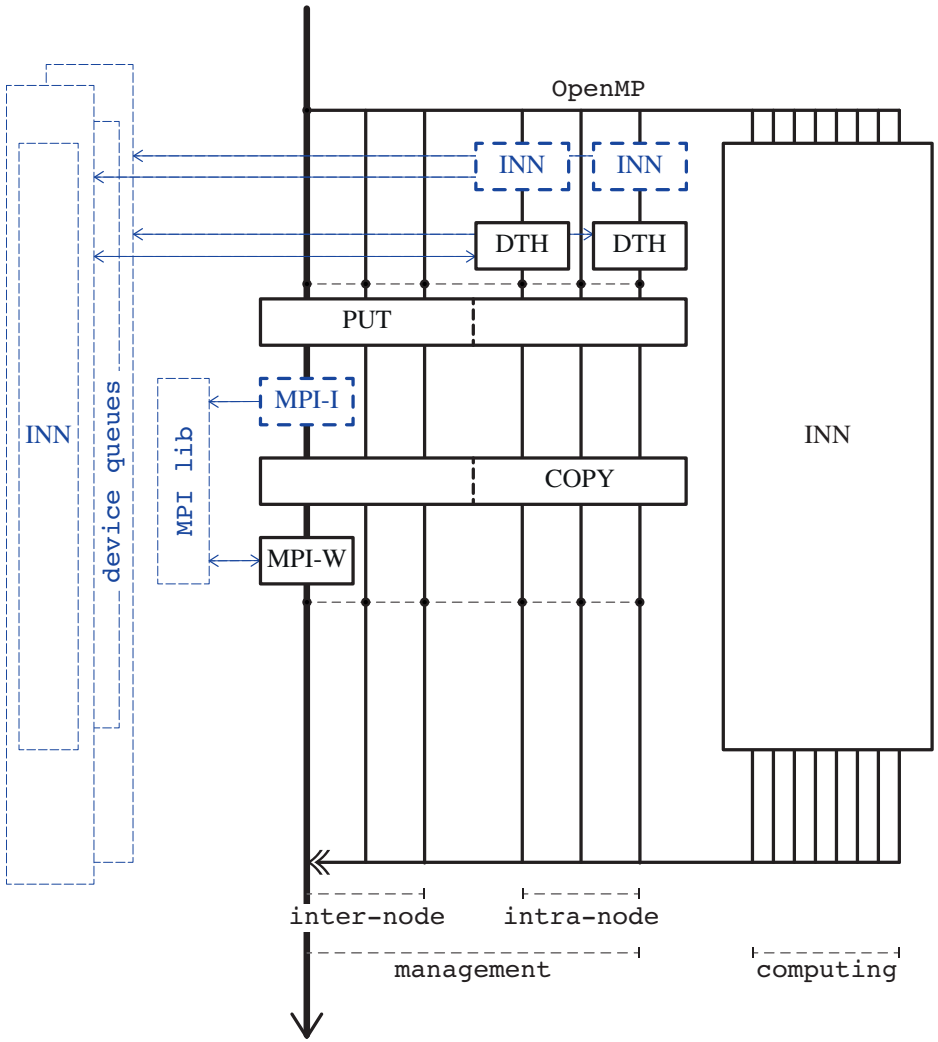


To minimise the overhead of the communications, efficient multithreaded execution strategies are required. Roughly, the idea is to overlap the communications with the computations.



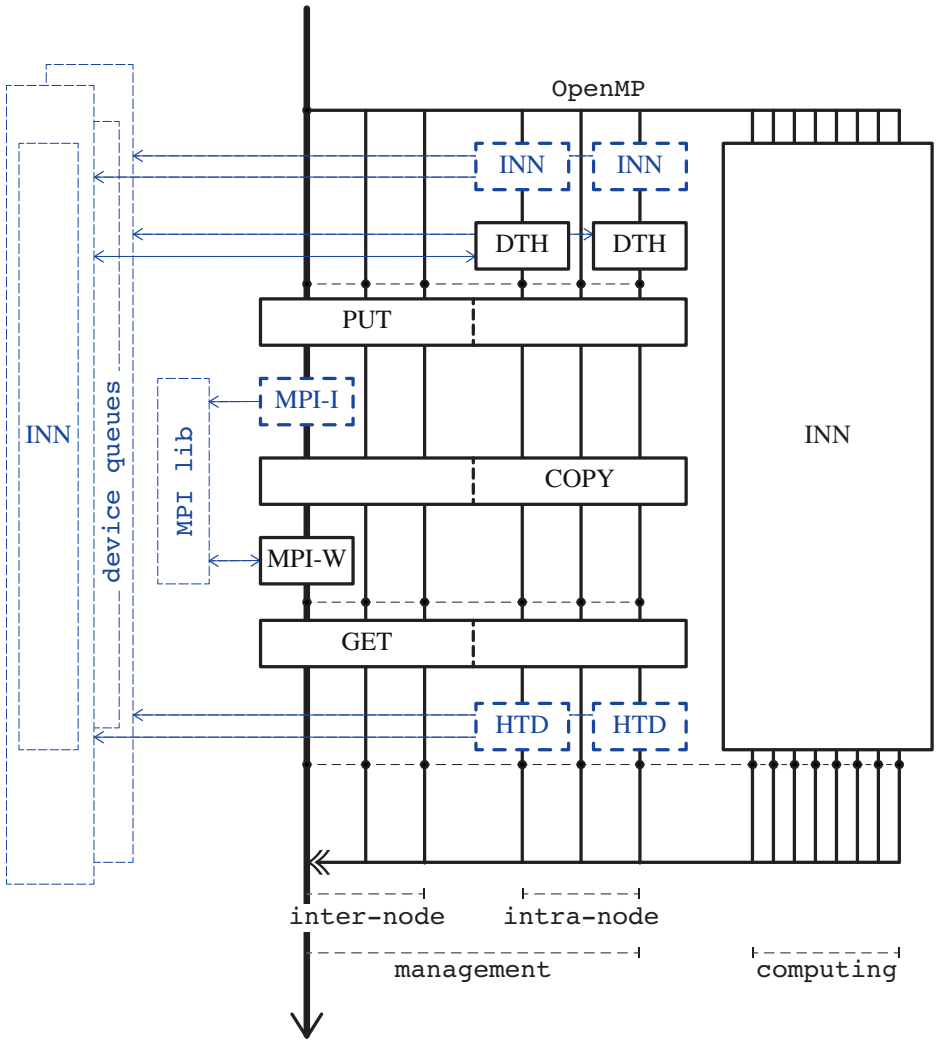
Flat multithreaded execution strategies

To minimise the overhead of the communications, efficient multithreaded execution strategies are required. Roughly, the idea is to overlap the communications with the computations.



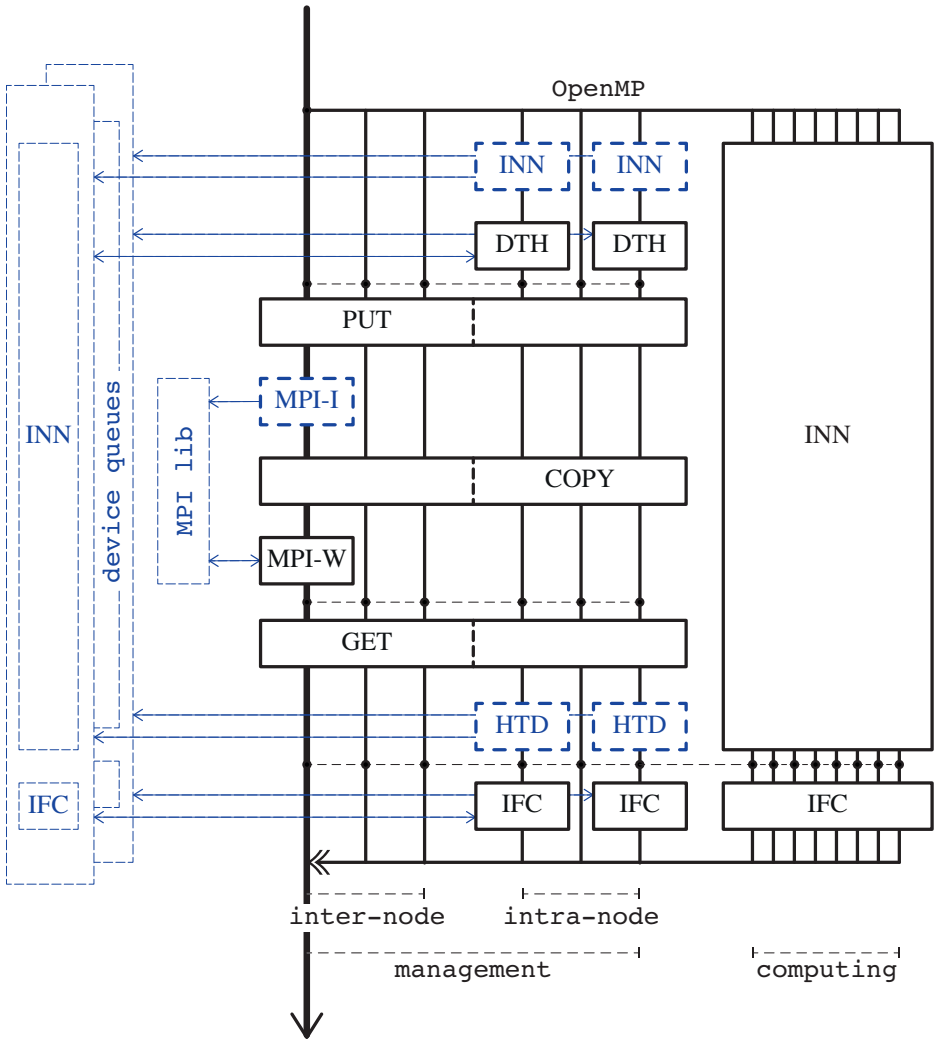
Flat multithreaded execution strategies

To minimise the overhead of the communications, efficient multithreaded execution strategies are required. Roughly, the idea is to overlap the communications with the computations.



Flat multithreaded execution strategies

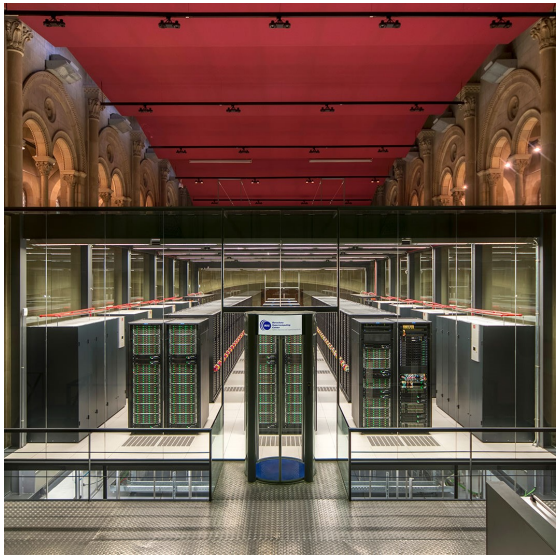
To minimise the overhead of the communications, efficient multithreaded execution strategies are required. Roughly, the idea is to overlap the communications with the computations.



Performance study

HPC systems

MareNostrum 4



#rank42
3456 nodes with:

- 2× Intel Xeon 8160
- 1× Intel Omni-Path

Lomonosov-2

#rank156
1696 nodes with:

- 1× Intel Xeon E5-2697 v3
- 1× NVIDIA Tesla K40M
- 1× InfiniBand FDR

TSUBAME3.0

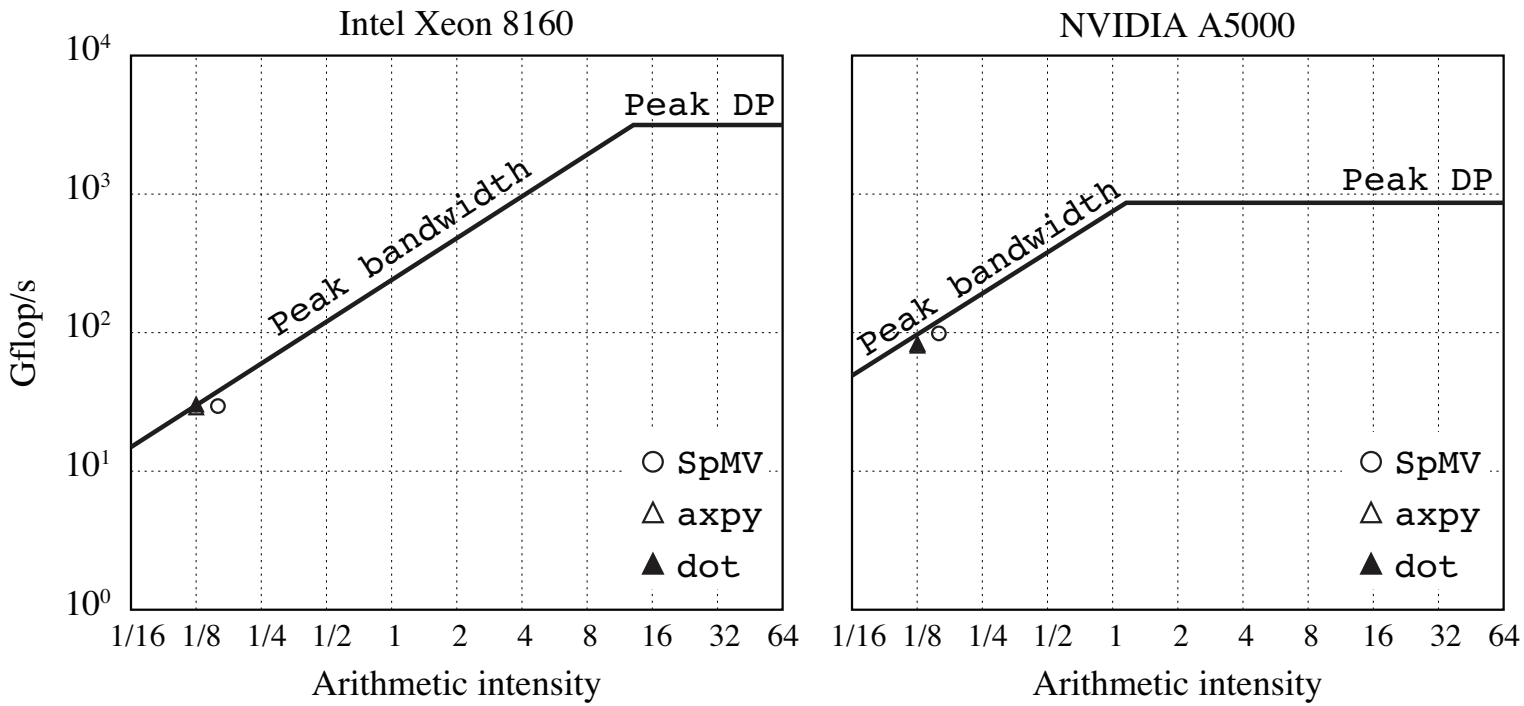
#rank31
540 nodes with:

- 2× Intel Xeon E5-2680 v4
- 4× NVIDIA Tesla P100
- 4× Intel Omni-Path

Other systems: JFF third- and fourth-generation, MareNostrum 3, MinoTauro, K60, Titan, Mira, Cori, Marconi100...

Test case

Single-node performance of SpMV, axpy and dot kernels shown in roofline model for two different architectures. The sparse matrix used arises from the symmetry-preserving discretization of the Laplacian operator on hex-dominant mesh of 17 million cells. The sparse matrix storage format used is ELLPACK.



The SpMV is an essential operation in scientific computing, and therefore, it receives a great deal of attention. Given $\vec{x} \in \mathbb{R}^n$, $\vec{y} \in \mathbb{R}^m$, and $\mathsf{A} \in \mathbb{R}^{m \times n}$:

$$\vec{y} \leftarrow \mathsf{A}\vec{x} : AI_{spmv} = \frac{2nnz(\mathsf{A})}{12nnz(\mathsf{A}) + 4m + 8n + 8m} \approx 0.13.$$

Algorithm 1 SpMV implementation using the standard CSR matrix format.

Require: A, $\mathbf{x}$
Ensure: $\mathbf{y}$
1: **for** $i \leftarrow 1$ to m **do**
2: **for** $j \leftarrow \mathsf{A}.rptr[i]$ to $\mathsf{A}.rptr[i+1]$ **do**
3: $y[i] \leftarrow y[i] + \mathsf{A}.coef[j] \cdot \mathbf{x}[\mathsf{A}.cidx[j]]$

The SpMM represents the product of a sparse matrix by a dense matrix. It is very beneficial in terms of achievable performance to implement a specific SpMM that takes advantage of the reuse of the matrix coefficients. Given $\vec{x} \in \mathbb{R}^{kn}$, $\vec{y} \in \mathbb{R}^{km}$, and $\mathsf{A} \in \mathbb{R}^{m \times n}$:

$$\begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{pmatrix} \leftarrow \begin{pmatrix} \mathsf{A} & & 0 \\ & \ddots & \\ 0 & & \mathsf{A} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} : AI_{spmm} = \frac{2knnz(\mathsf{A})}{12nnz(\mathsf{A}) + 4m + 8kn + 8km}$$

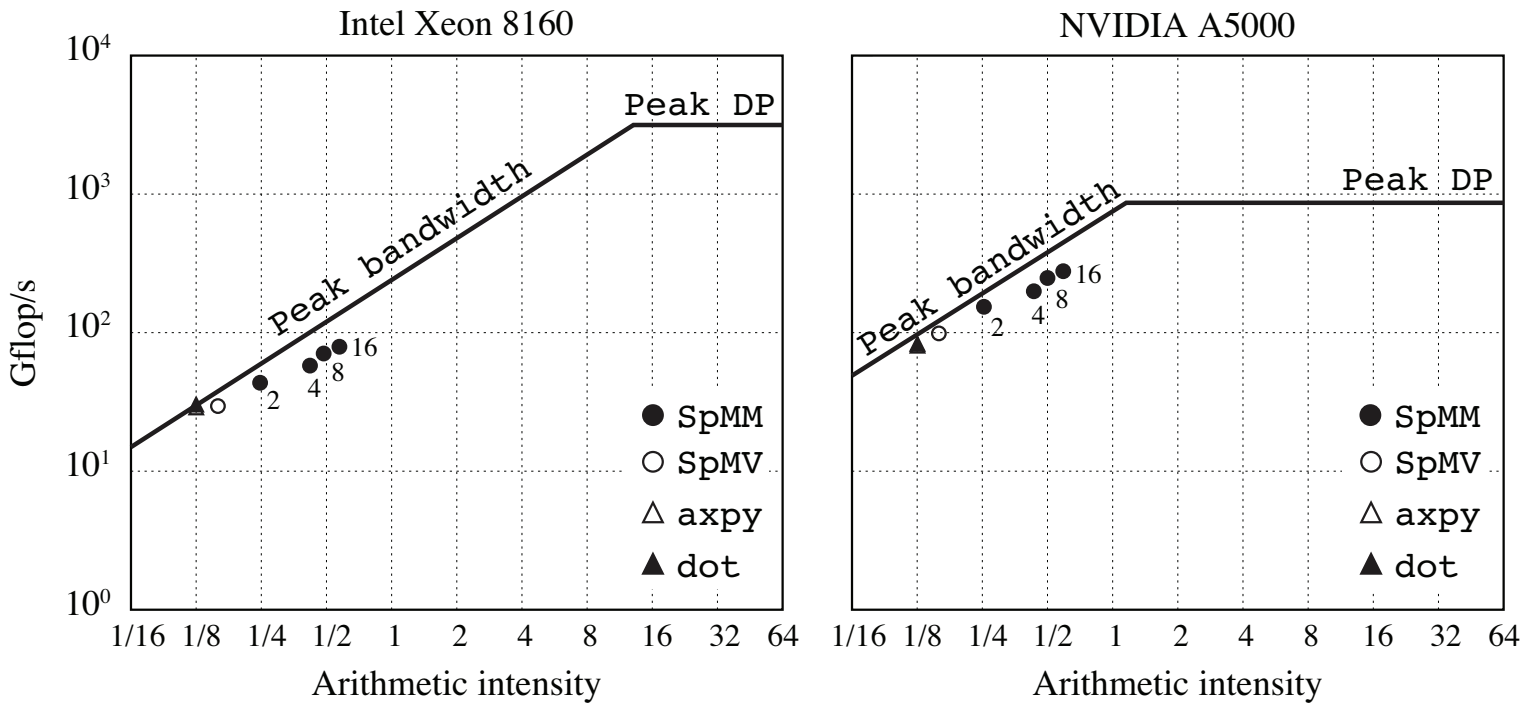
Algorithm 2 SpMM implementation using the standard CSR matrix format.

Require: A, $\mathbf{x}$
Ensure: $\mathbf{y}$

```
1: for i ← 1 to m do
2:     for j ← A.rptr[i] to A.rptr[i + 1] do
3:         for k ← 1 to K do
4:             y[i][k] ← y[i][k] + A.coef[j] · x[A.cidx[j]][k]
```

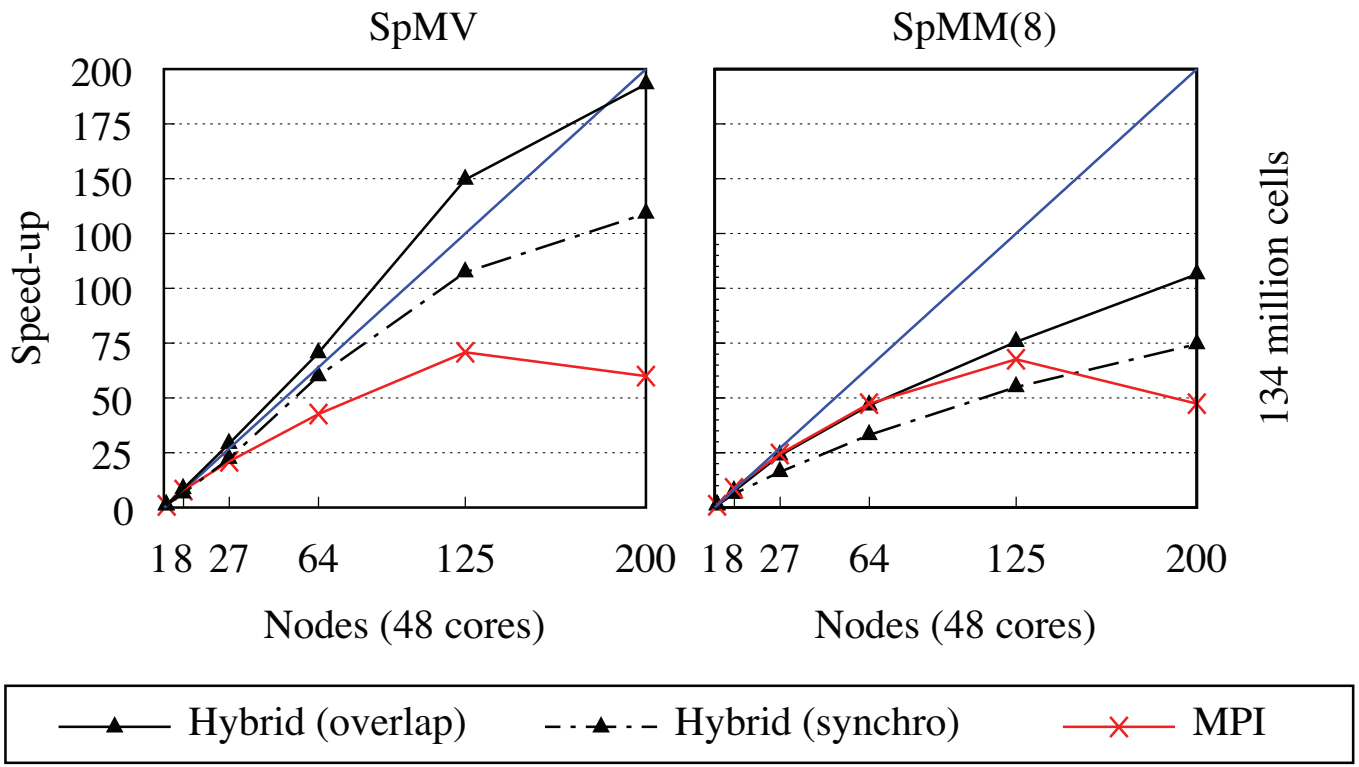
Test case

Single-node performance of SpMM, SpMV, axpy and dot kernels shown in roofline model for two different architectures. The sparse matrix used arises from the symmetry-preserving discretization of the Laplacian operator on hex-dominant mesh of 17 million cells. The sparse matrix storage format used is ELLPACK.



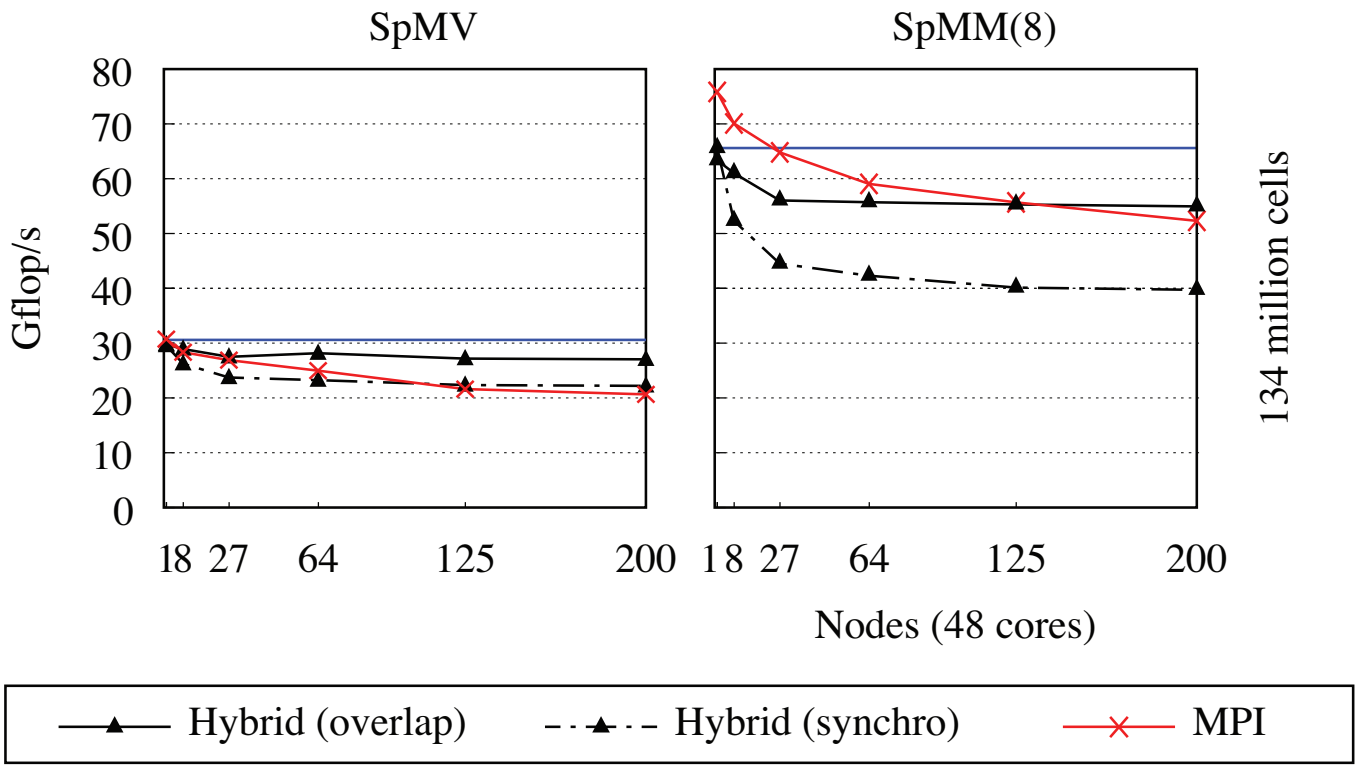
Test case

Multi-node strong scaling of SpMV and SpMM kernels on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells (also 110 million in strong scaling). The sparse matrix storage format used is ELLPACK.



Test case

Multi-node weak scaling of SpMV and SpMM kernels on MareNostrum 4. The sparse matrix used arise from the symmetry-preserving discretization of the Laplacian operator on unstructured hex-dominant mesh of 17 million cells (also 110 million in strong scaling). The sparse matrix storage format used is ELLPACK.



First of all, remark that the SpMV is the most time-consuming kernel in a typical CFD simulation deployed in our framework, nearly 90%. Therefore, any SpMV optimization has a huge impact in performance.

- Multiple components of velocity in collocated formulation. Directly $k = 3$ for 3D simulations.

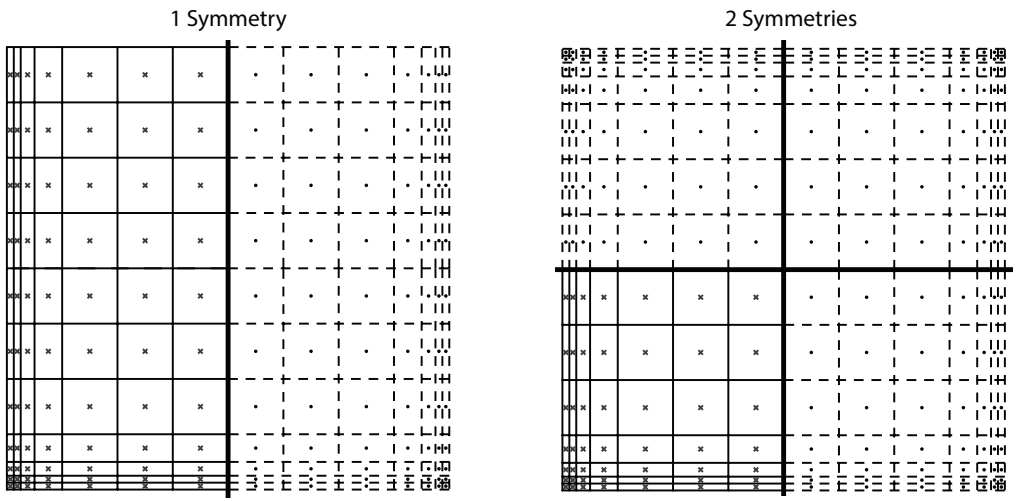
First of all, remark that the SpMV is the most time-consuming kernel in a typical CFD simulation deployed in our framework, nearly 90%. Therefore, any SpMV optimization has a huge impact in performance.

- Multiple components of velocity in collocated formulation. Directly $k = 3$ for 3D simulations.
- Multiple transport equations (*e.g.*, temperature, chemicals). Considering only temperature (Algorithm 1), increases to $k = 4$.

Applications of SpMM

First of all, remark that the SpMV is the most time-consuming kernel in a typical CFD simulation deployed in our framework, nearly 90%. Therefore, any SpMV optimization has a huge impact in performance.

- Multiple components of velocity in collocated formulation. Directly $k = 3$ for 3D simulations.
- Multiple transport equations (*e.g.*, temperature, chemicals). Considering only temperature (Algorithm 1), increases to $k = 4$.
- Simulations on a mesh with p symmetries. Increases k by a factor of p^2, and also reduces memory footprint of discrete operators.



UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Strategies to increase the arithmetic intensity of the linear solvers

Àdel Alsalti-Baldellou[1,2] Xavier Álvarez-Farré[1] Andrey Gorobets
Assensi Oliva[1] F. Xavier Trias[1]

[1]Heat and Mass Transfer Technological Center (CTTC), at UPC
Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain
adel.alsalti@upc.com

[2]Termo Fluids SL, Carrer de Magí Colet 8, 08204 Sabadell (Barcelona), Spain

33rd Parallel CFD International Conference
25-27 May 2022 – Alba (Italy)

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Strategies to increase the arithmetic intensity of the linear solvers

Àdel Alsalti-Baldellou[1,2] Xavier Álvarez-Farré[1] Andrey Gorobets
Assensi Oliva[1] F. Xavier Trias[1]

[1]Heat and Mass Transfer Technological Center (CTTC), at UPC
Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain
adel.alsalti@upc.com

[2]Termo Fluids SL, Carrer de Magí Colet 8, 08204 Sabadell (Barcelona), Spain

33rd Parallel CFD International Conference
25-27 May 2022 – Alba (Italy)

The direct benefits of this approach are:

- Reduces the number of iterations and its computational cost (increased AI).

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Strategies to increase the arithmetic intensity of the linear solvers

Àdel Alsalti-Baldellou[1,2] Xavier Álvarez-Farré[1] Andrey Gorobets
Assensi Oliva[1] F. Xavier Trias[1]

[1]Heat and Mass Transfer Technological Center (CTTC), at UPC
Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain
adel.alsalti@upc.com

[2]Termo Fluids SL, Carrer de Magí Colet 8, 08204 Sabadell (Barcelona), Spain

33rd Parallel CFD International Conference
25-27 May 2022 – Alba (Italy)

The direct benefits of this approach are:

- Reduces the number of iterations and its computational cost (increased AI).
- Reduces substantially the memory footprint of the matrices.

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Strategies to increase the arithmetic intensity of the linear solvers

Àdel Alsalti-Baldellou[1,2] Xavier Álvarez-Farré[1] Andrey Gorobets
Assensi Oliva[1] F. Xavier Trias[1]

[1]Heat and Mass Transfer Technological Center (CTTC), at UPC
Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain
adel.alsalti@upc.com

[2]Termo Fluids SL, Carrer de Magí Colet 8, 08204 Sabadell (Barcelona), Spain

33rd Parallel CFD International Conference
25-27 May 2022 – Alba (Italy)

The direct benefits of this approach are:

- Reduces the number of iterations and its computational cost (increased AI).
- Reduces substantially the memory footprint of the matrices.
- Reduces substantially the cost of building complex preconditioners.

Conclusions and Future Work

Conclusions

Future Work

Conclusions

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

Future Work

Conclusions

- An algebra-based framework has been presented as a naturally portable strategy for implementing numerical simulation codes.
- The hierarchical parallel implementation of our framework has been detailed, and its performance evaluated on various HPC system.

Future Work

Conclusions

- An algebra-based framework has been presented as a naturally portable strategy for implementing numerical simulation codes.
- The hierarchical parallel implementation of our framework has been detailed, and its performance evaluated on various HPC system.
- The SpMM kernel has proved to increase arithmetic intensity of algebraic implementations; however, it becomes computationally light and hence difficult to scale.

Future Work

Conclusions

- An algebra-based framework has been presented as a naturally portable strategy for implementing numerical simulation codes.
- The hierarchical parallel implementation of our framework has been detailed, and its performance evaluated on various HPC system.
- The SpMM kernel has proved to increase arithmetic intensity of algebraic implementations; however, it becomes computationally light and hence difficult to scale.
- The application of SpMM within algebraic frameworks is demonstrated to be versatile and powerful. Particularly, in the presence of mesh symmetries the benefits are threefold: reduces number of iterations, computational cost and memory footprint.

Future Work

Conclusions

- An algebra-based framework has been presented as a naturally portable strategy for implementing numerical simulation codes.
- The hierarchical parallel implementation of our framework has been detailed, and its performance evaluated on various HPC system.
- The SpMM kernel has proved to increase arithmetic intensity of algebraic implementations; however, it becomes computationally light and hence difficult to scale.
- The application of SpMM within algebraic frameworks is demonstrated to be versatile and powerful. Particularly, in the presence of mesh symmetries the benefits are threefold: reduces number of iterations, computational cost and memory footprint.

Future Work

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

Conclusions

- An algebra-based framework has been presented as a naturally portable strategy for implementing numerical simulation codes.
- The hierarchical parallel implementation of our framework has been detailed, and its performance evaluated on various HPC system.
- The SpMM kernel has proved to increase arithmetic intensity of algebraic implementations; however, it becomes computationally light and hence difficult to scale.
- The application of SpMM within algebraic frameworks is demonstrated to be versatile and powerful. Particularly, in the presence of mesh symmetries the benefits are threefold: reduces number of iterations, computational cost and memory footprint.

Future Work

- To design a new update mechanism to accelerate the data exchanges, for instance, taking into account NUIOA factor in inter- and intra-node exchanges.
- Applying our framework to multiple parameters simulations. Considering n different simulations, increases k by a factor of n, allowing for running multiple simulations faster while maintaining the memory footprint of discrete operators constant.

Conclusions

- An algebra-based framework has been presented as a naturally portable strategy for implementing numerical simulation codes.
- The hierarchical parallel implementation of our framework has been detailed, and its performance evaluated on various HPC system.
- The SpMM kernel has proved to increase arithmetic intensity of algebraic implementations; however, it becomes computationally light and hence difficult to scale.
- The application of SpMM within algebraic frameworks is demonstrated to be versatile and powerful. Particularly, in the presence of mesh symmetries the benefits are threefold: reduces number of iterations, computational cost and memory footprint.

Future Work

- To design a new update mechanism to accelerate the data exchanges, for instance, taking into account NUIOA factor in inter- and intra-node exchanges.
- Applying our framework to multiple parameters simulations. Considering n different simulations, increases k by a factor of n, allowing for running multiple simulations faster while maintaining the memory footprint of discrete operators constant.
- Applying our framework to parallel-in-time simulations. Considering t decompositions in time, increases k by a factor of t, allowing for solving multiple time-intervals faster while maintaining the memory footprint of discrete operators constant.

Thank you for your attention