A PURE VIRTUAL APPROACH FOR MANAGING PLATFORM PORTABILITY ON HYBRID SUPERCOMPUTERS

Xavier ÁLVAREZ-FARRÉ[1], Andrey GOROBETS[2], Àdel ALSATI[1] AND F. Xavier TRIAS[1]

In the 31st International Conference on Parallel Computational Fluid Dynamics (ParCFD2019)
May 14th, 2019, Antalya (Turkey)

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

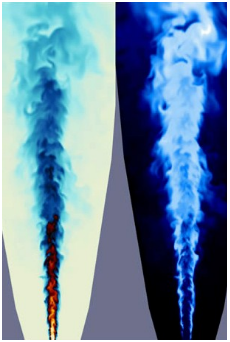
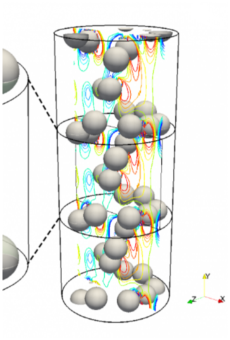
Centre Tecnològic de Transferència de Calor
UNIVERSITAT POLITÈCNICA DE CATALUNYA

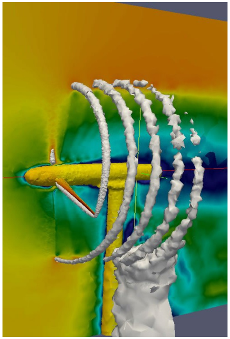
Background

Research at CTTC Laboratory

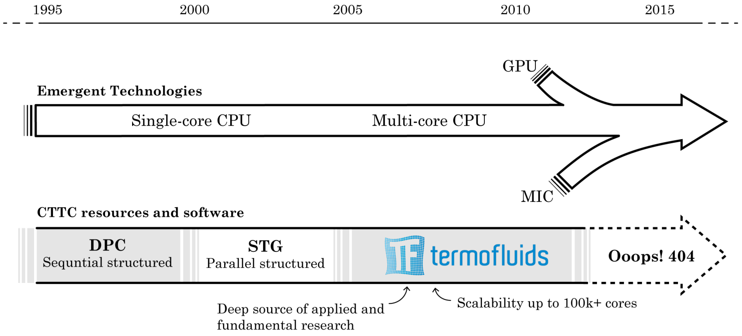
The Heat and Mass Transfer Technological Center (CTTC) in Technical University of Catalonia (UPC) has been working on CFD for more than 20 years. Our research activities are focused on two main lines:

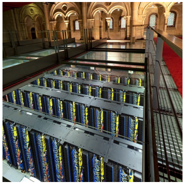
- Fundamental research on fluid dynamics and heat and mass transfer phenomena.
- Applied research on thermal and fluid dynamic optimization of thermal system and equipment.



HPC at CTTC Laboratory





Emerging architectures

Continuous enhancement

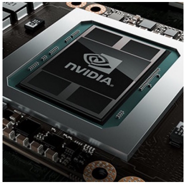
in hardware technologies enables scientific computing to advance incessantly to reach further aims. After hitting petascale speeds in 2008, several organisations and institutions began the well-known global race for exascale high-performance computing (HPC).

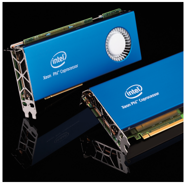
Emerging architectures

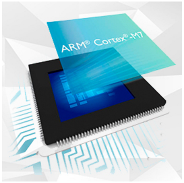
Continuous enhancement

in hardware technologies enables scientific computing to advance incessantly to reach further aims. After hitting petascale speeds in 2008, several organisations and institutions began the well-known global race for exascale high-performance computing (HPC).

Massively-parallel devices of various architectures are being incorporated into the newest 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 with stream processing (SP), which is a very restrictive parallel paradigm.







The future of scientific computing codes

¿Is it necessary to use the new hardware architectures?

- In our opinion, yes. New hardware is designed to overcome the power constraint in the context of the exascale challenge.

The future of scientific computing codes

¿Is it necessary to use the new hardware architectures?

- In our opinion, yes. New hardware is designed to overcome the power constraint 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 suitable for CPU architectures.

The future of scientific computing codes

¿Is it necessary to use the new hardware architectures?

- In our opinion, yes. New hardware is designed to overcome the power constraint 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 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 huge amount of hardware architectures and it is difficult to determine which are going to prevail. The scientific computing software should be designed somehow so that computing kernels can operate independently.

The algebra-based approach

Stencil-based

Traditionally, the stencil-based implementations are used by the scientific computing community. These implementations arise straightforward from the formulation of the numerical method. However, they require **specific stencil sweeps and data structures** for each numerical method.

Algebra-based

Algebra-based implementations only rely on a reduced number of **universal algebraic kernels and data structures,** allowing the use of standard optimised libraries and, therefore, providing portability. As a counterpart, the formulation of the numerical method becomes more complex and could even lead to an increase in the number of operations.

The algebra-based approach

Following an algebra-based approach, we replace the traditional stencil data structures and sweeps by algebraic data structures and kernels[1]. For instance, the algebra-based, finite-volume discretisation of NS and continuity equations on an arbitrary collocated mesh can be written as[2]

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

$$\mathbf{M}\mathbf{u}_s = \mathbf{0}_c, \qquad \mathbf{\Omega} d_t \mathbf{u}_c + \mathbf{M}\mathbf{U}_s \mathbf{u}_c + \mathbf{D}\mathbf{u}_c - \mathbf{M}^{\top} p_c = \mathbf{0}_c.$$

[1] Álvarez-Farré et al., HPC^2–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD, **Computers and Fluids**, 173, 285–292, 2018.

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

The algebra-based approach

Following an algebra-based approach, we replace the traditional stencil data structures and sweeps by algebraic data structures and kernels[1]. For instance, the algebra-based, finite-volume discretisation of NS and continuity equations on an arbitrary collocated mesh can be written as[2]

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

$$\mathbf{M}\mathbf{u}_s = \mathbf{0}_c, \qquad \mathbf{\Omega} d_t \mathbf{u}_c + \mathbf{M}\mathbf{U}_s \mathbf{u}_c + \mathbf{D}\mathbf{u}_c - \mathbf{M}^{\top} p_c = \mathbf{0}_c.$$

1. The discrete variables are stored in vectors and the discrete operators in sparse matrices.

[1] Álvarez-Farré et al., HPC^2–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD, **Computers and Fluids**, 173, 285–292, 2018.

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

The algebra-based approach

Following an algebra-based approach, we replace the traditional stencil data structures and sweeps by algebraic data structures and kernels[1]. For instance, the algebra-based, finite-volume discretisation of NS and continuity equations on an arbitrary collocated mesh can be written as[2]

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

$$\mathbf{M}\mathbf{u}_s = \mathbf{0}_c, \qquad \mathbf{\Omega} d_t \mathbf{u}_c + \mathbf{M}\mathbf{U}_s \mathbf{u}_c + \mathbf{D}\mathbf{u}_c - \mathbf{M}^\top p_c = \mathbf{0}_c.$$

1. The discrete variables are stored in vectors and the discrete operators in sparse matrices.
2. The numerical method results fully integrated into the data structures somehow so that computing kernels can operate independently.

[1] Álvarez-Farré et al., HPC^2–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD, **Computers and Fluids**, 173, 285–292, 2018.

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

The algebra-based approach

Following an algebra-based approach, we replace the traditional stencil data structures and sweeps by algebraic data structures and kernels[1]. For instance, the algebra-based, finite-volume discretisation of NS and continuity equations on an arbitrary collocated mesh can be written as[2]

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

$$\mathbf{M}\mathbf{u}_s = \mathbf{0}_c, \qquad \mathbf{\Omega} d_t \mathbf{u}_c + \mathbf{M}\mathbf{U}_s \mathbf{u}_c + \mathbf{D}\mathbf{u}_c - \mathbf{M}^{\mathsf{T}} p_c = \mathbf{0}_c.$$

1. The discrete variables are stored in vectors and the discrete operators in sparse matrices.
2. The numerical method results fully integrated into the data structures somehow so that computing kernels can operate independently.
3. The discrete operators can be built directly from the inherent incidence matrices that define the mesh mimicking the properties of the continuum operators.

[1] Álvarez-Farré et al., HPC^2–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD, **Computers and Fluids**, 173, 285–292, 2018.

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

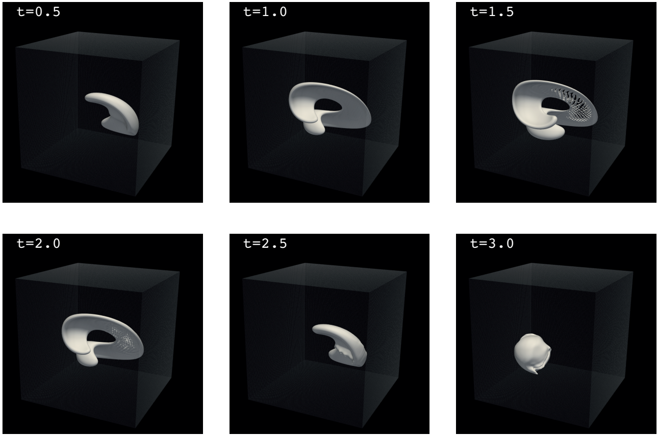
The HPC^2 framework

The HPC^2 framework

The HPC^2 (Heterogeneous Portable Code for HPC) is a fully-portable, algebra-based framework with many potential applications in the fields of computational physics and mathematics. Its algebraic approach combined with a multilevel MPI + OpenMP + OpenCL + CUDA parallelisation naturally provides modularity and portability.

The HPC^2 framework

The HPC^2 (Heterogeneous Portable Code for HPC) is a fully-portable, algebra-based framework with many potential applications in the fields of computational physics and mathematics. Its algebraic approach combined with a multilevel MPI + OpenMP + OpenCL + CUDA parallelisation naturally provides modularity and portability.



The HPC^2 framework

The HPC^2 (Heterogeneous Portable Code for HPC) is a fully-portable, algebra-based framework with many potential applications in the fields of computational physics and mathematics. Its algebraic approach combined with a multilevel MPI + OpenMP + OpenCL + CUDA parallelisation naturally provides modularity and portability.

Using our algebra-based framework, a numerical algorithm relies on a few computing kernels, grouped in the following three types of algebraic operations:

- `spmv`: sparse matrix-vector operations.
- `nary`: n-ary vector operations require n input vectors and return an output vector.
- `sred`: reduction operations require n input vectors and return a scalar value.

Besides, having a reduced set of simple kernels, the arithmetic intensity of an algorithm[3] can be estimated easily and thus its relative performance.

Kernel	Work	Read	Write	AI
spmv	13N	64N	8N	13/72
axpy	N	8N	8N	1/16
sdot	2N	16N	8	1/8

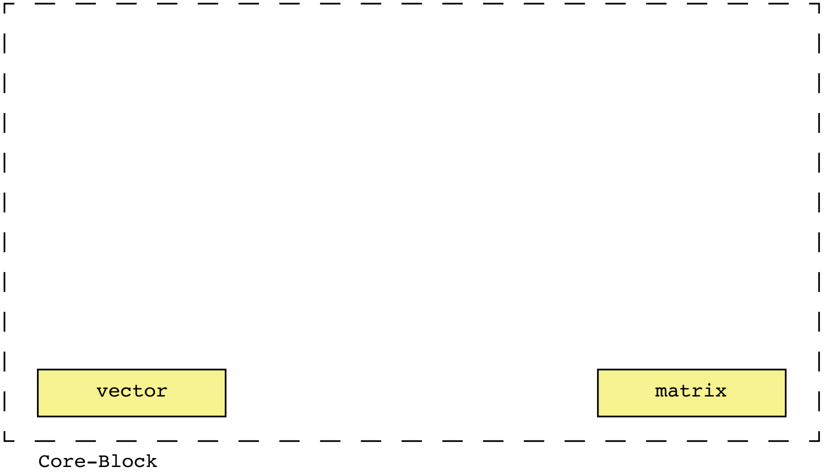
[3] Álvarez-Farré et al., HPC^2–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD, **Computers and Fluids**, 173, 285–292, 2018.

The HPC^2 pure-virtual core

The HPC^2 is organised as follows: `vector`, `matrix` and `unit` are pure virtual classes; derived classes may be developed for OpenMP, OpenCL, CUDA, etc. The handlers `VECTOR`, `MATRIX` and `NODE` may contain several instances of virtual objects.

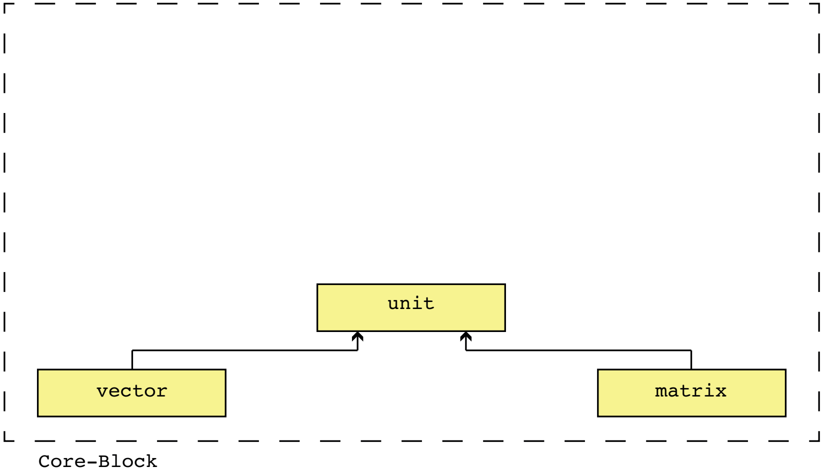
The HPC^2 pure-virtual core

The HPC^2 is organised as follows: `vector`, `matrix` and `unit` are pure virtual classes; derived classes may be developed for OpenMP, OpenCL, CUDA, etc. The handlers `VECTOR`, `MATRIX` and `NODE` may contain several instances of virtual objects.



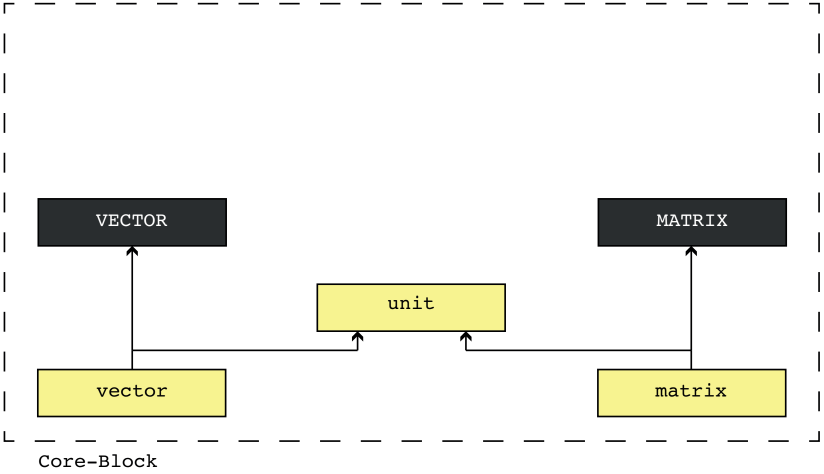
The HPC^2 pure-virtual core

The HPC^2 is organised as follows: `vector`, `matrix` and `unit` are pure virtual classes; derived classes may be developed for OpenMP, OpenCL, CUDA, etc. The handlers `VECTOR`, `MATRIX` and `NODE` may contain several instances of virtual objects.



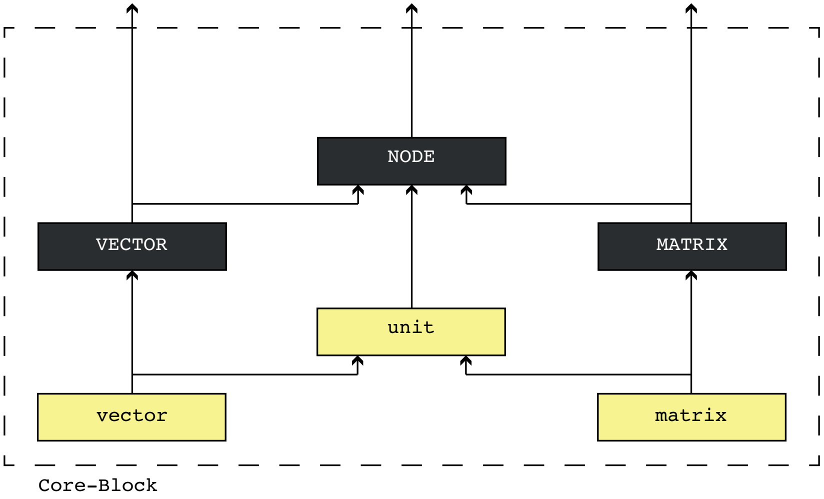
The HPC^2 pure-virtual core

The HPC^2 is organised as follows: `vector`, `matrix` and `unit` are pure virtual classes; derived classes may be developed for OpenMP, OpenCL, CUDA, etc. The handlers `VECTOR`, `MATRIX` and `NODE` may contain several instances of virtual objects.



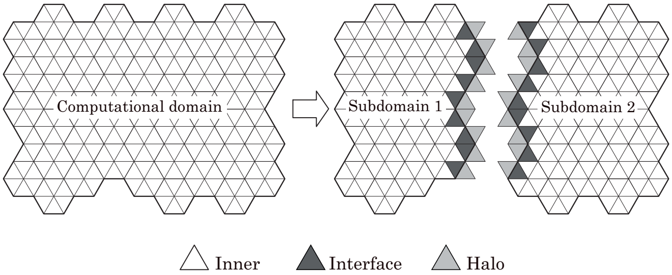
The HPC^2 pure-virtual core

The HPC^2 is organised as follows: `vector`, `matrix` and `unit` are pure virtual classes; derived classes may be developed for OpenMP, OpenCL, CUDA, etc. The handlers `VECTOR`, `MATRIX` and `NODE` may contain several instances of virtual objects.



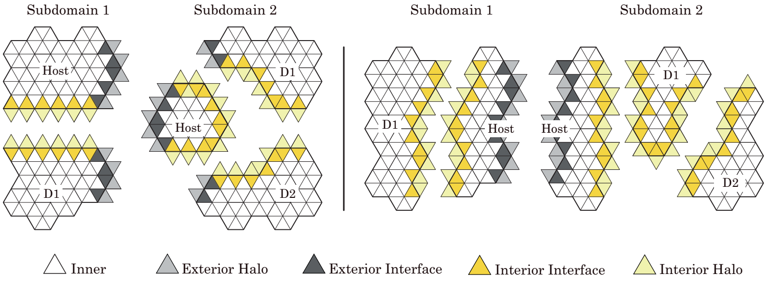
First-level decomposition, distributed-memory parallelisation

The first-level domain decomposition distributes the workload among the computing nodes. Subdomain elements are classified into *inner* and *interface* categories. Consequently, the adjacent elements from other subdomains form a *halo*.



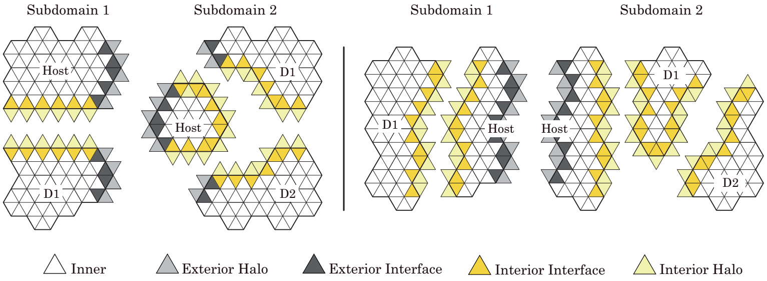
Second-level decomposition, intra-node parallelisation

The second-level domain decomposition distributes the workload of each MPI process among its computing hardware, namely *host* and *devices*. The interface and halo elements are further classified as *external* and *internal*.



Second-level decomposition, intra-node parallelisation

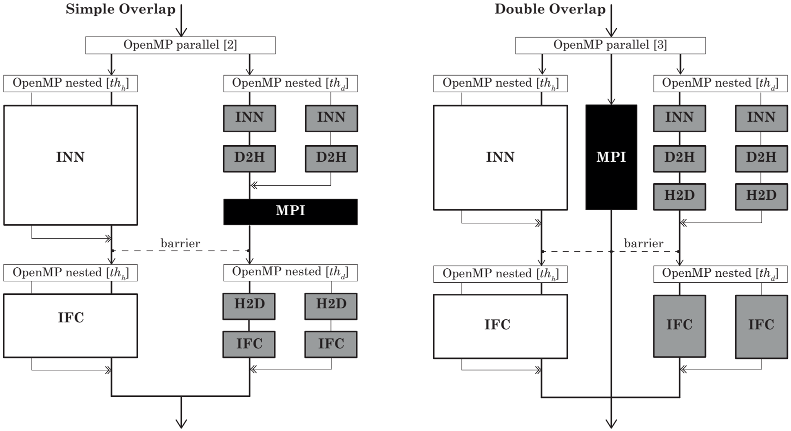
The second-level domain decomposition distributes the workload of each MPI process among its computing hardware, namely *host* and *devices*. The interface and halo elements are further classified as *external* and *internal*.



Second-level decomposition reduces the volume of the expensive device-host-MPI-host-device communications several times!

Multithreaded overlap strategies

The strategies for an efficient heterogeneous execution of large-scale simulations on hybrid supercomputers that are part of the HPC^2 core are shown in the flowchart below:



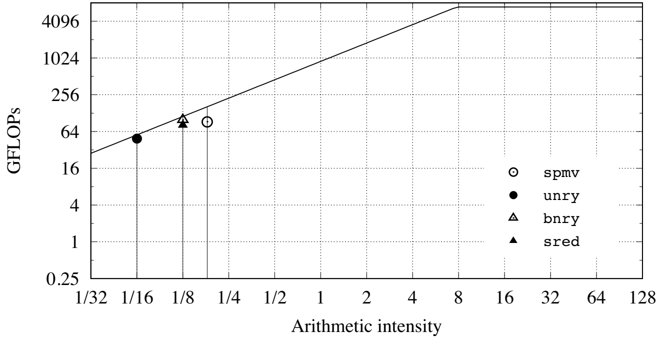
Performance analysis

Single-node, roofline study

The roofline model is an intuitive performance analysis model used to estimate the maximum performance of a given computing kernel depending on the actual hardware specifications: the memory bandwidth, π, and the peak performance, β:

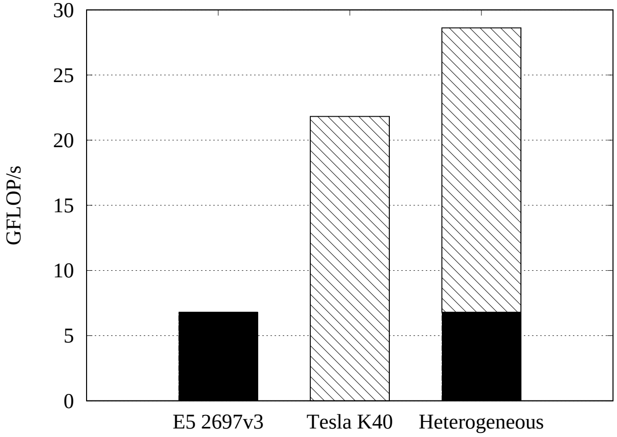
$$R_k = min(\pi, \beta \cdot AI_k).$$

Nvidia V100



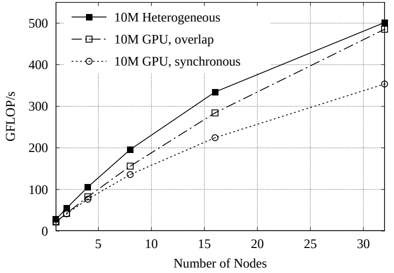
Single-node, heterogeneous performance study

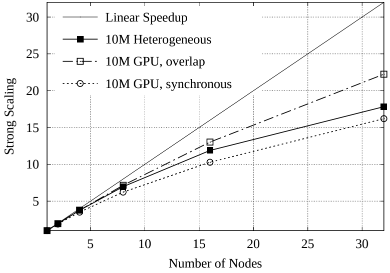
The heterogeneous performance study shows an increase of 32% compared to GPU-only mode, which corresponds to a 98% of heterogeneous efficiency if compared to the sum of the performance of the CPU-only and the GPU-only modes.



Multi-node, heterogeneous performance study

The strong scalability study (ran on nodes equipped with an Intel E5-2697v3 and a NVIDIA Tesla K40M) shows that the simple overlap strategy improves the performance by hiding the communications. The scalability decays faster in the heterogeneous mode because the computational load per GPU is smaller and the CPU is loaded with computations which may interfere with MPI library routines, reducing the overlapping operational range.





Conclusions

Design of modular frameworks

Making an effort to

design modular frameworks composed of a reduced number of independent and well-defined code blocks is worth it:

Design of modular frameworks

Making an effort to

design modular frameworks composed of a reduced number of independent and well-defined code blocks is worth it:

- On the one hand, this allows researchers to develop different code blocks avoiding interferences, reduces the generation and propagation of errors and facilitates debugging. The progress of a modular code may be accumulative.

Design of modular frameworks

Making an effort to

design modular frameworks composed of a reduced number of independent and well-defined code blocks is worth it:

- On the one hand, this allows researchers to develop different code blocks avoiding interferences, reduces the generation and propagation of errors and facilitates debugging. The progress of a modular code may be accumulative.
- On the other hand, modular applications are user-friendly and more comfortable for porting to new architectures (the fewer the kernels of an application, the easier it is to provide portability). Furthermore, if the majority of kernels represent linear algebra operations, then standard optimised libraries (*e.g.* ATLAS, clBLAST) or specific in-house implementations can be used and easily switched.

The algebra-based approach

The algebra-based implementation introduces an interesting discussion about its advantages and disadvantages.

The algebra-based approach

The algebra-based implementation introduces an interesting discussion about its advantages and disadvantages.

PROS

- It is a flexible, elegant approach in which algorithms can be represented as a combination of a few algebraic operations.

The algebra-based approach

The algebra-based implementation introduces an interesting discussion about its advantages and disadvantages.

PROS

- It is a flexible, elegant approach in which algorithms can be represented as a combination of a few algebraic operations.
- It allows for a flawless discrete mimicking of the continuum operators. In particular, it allows for the exact conservation of important secondary properties,

The algebra-based approach

The algebra-based implementation introduces an interesting discussion about its advantages and disadvantages.

PROS

- It is a flexible, elegant approach in which algorithms can be represented as a combination of a few algebraic operations.
- It allows for a flawless discrete mimicking of the continuum operators. In particular, it allows for the exact conservation of important secondary properties,
- It relies on a reduced set of computing kernels and guarantees that the computational implementation is completely independent of the numerical method and the mathematical model. Therefore it naturally provides portability.

The algebra-based approach

The algebra-based implementation introduces an interesting discussion about its advantages and disadvantages.

PROS

- It is a flexible, elegant approach in which algorithms can be represented as a combination of a few algebraic operations.
- It allows for a flawless discrete mimicking of the continuum operators. In particular, it allows for the exact conservation of important secondary properties,
- It relies on a reduced set of computing kernels and guarantees that the computational implementation is completely independent of the numerical method and the mathematical model. Therefore it naturally provides portability.

CONS

- The use of several kernels may result in many intermediate in/out data movements; this reduces the arithmetic intensity and may be a huge disadvantage for memory bounded applications.

The algebra-based approach

The algebra-based implementation introduces an interesting discussion about its advantages and disadvantages.

PROS

- It is a flexible, elegant approach in which algorithms can be represented as a combination of a few algebraic operations.
- It allows for a flawless discrete mimicking of the continuum operators. In particular, it allows for the exact conservation of important secondary properties,
- It relies on a reduced set of computing kernels and guarantees that the computational implementation is completely independent of the numerical method and the mathematical model. Therefore it naturally provides portability.

CONS

- The use of several kernels may result in many intermediate in/out data movements; this reduces the arithmetic intensity and may be a huge disadvantage for memory bounded applications.
- It also requires a larger use of data since the numerical method is completely integrated into data structures.

The algebra-based approach

The algebra-based implementation introduces an interesting discussion about its advantages and disadvantages.

PROS

- It is a flexible, elegant approach in which algorithms can be represented as a combination of a few algebraic operations.
- It allows for a flawless discrete mimicking of the continuum operators. In particular, it allows for the exact conservation of important secondary properties,
- It relies on a reduced set of computing kernels and guarantees that the computational implementation is completely independent of the numerical method and the mathematical model. Therefore it naturally provides portability.

CONS

- The use of several kernels may result in many intermediate in/out data movements; this reduces the arithmetic intensity and may be a huge disadvantage for memory bounded applications.
- It also requires a larger use of data since the numerical method is completely integrated into data structures.
- The implementation of complex boundary conditions is challenging. Are the boundary conditions an operator property or variable one?

Thank you for your attention