ParCFD'2019
31^{st} International Conference on Parallel Computational Fluid Dynamics
May-14-17 2019, Antalya TURKEY

A PURE VIRTUAL APPROACH FOR MANAGING PLATFORM PORTABILITY ON HYBRID SUPERCOMPUTERS

Xavier ÁLVAREZ-FARRÉ[*], Andrey GOROBETS[†], Àdel ALSALTI-BALDELLOU[*] AND F. Xavier TRIAS[*]

[*]Technical University of Catalonia
Heat and Mass Transfer Technological Center
C/ Colom 11, 08222 Terrassa (Barcelona), Spain
e-mail: xavier@cttc.upc.edu, web page: https://www.upc.edu/en

[†]Keldysh Institute of Applied Mathematics
Miusskaya Sq. 4, 125047 Moscow, Russia

Key words: Platform portability, Heterogeneous CFD computing, Hybrid supercomputers, MPI + OpenMP + OpenCL + CUDA, CPU + GPU

Abstract. Commonly, the development of scientific computing software follows a stencil-based approach: the numerical methods and algorithms are introduced into computing systems through stencil data structures and sweeps (*i.e.* traversing element by element the mesh and performing the required computations). Making an effort to remove the dependencies of the kernels on the numerical method may greatly facilitate the portability. Thus, the numerical method must be fully integrated into the data structures somehow so that computing kernels can operate independently. In this work, we propose a pure virtual approach for managing the architecture-specific data structure and computing kernel implementations while providing the users with a unique interface.

1 INTRODUCTION

Continuous enhancement in hardware technologies enables scientific computing advancing 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). Thenceforth, hardware developers are facing two significant challenges. Firstly, the energy efficiency of the exascale systems must be increased by two orders of magnitude. Secondly, there is an increasing demand for higher memory bandwidth. The common FLOP-oriented architectures (*i.e.* very high and growing FLOPS to memory bandwidth ratios) are not efficiently dealing with most of the algorithms in scientific computing, they barely reach 3% of their peak as shown by the HPCG Benchmark [1]. Therefore, massively-parallel devices of various architectures are being incorporated into the newest supercomputers, leading to an increasing hybridisation of HPC systems.

In this context of accelerated innovation, the software portability and efficiency become of crucial importance. The computing operations that form the algorithm, the so-called kernels, must be compatible with distributed- and shared-memory MIMD parallelism and, more importantly, with stream processing, which is a more restrictive parallel paradigm [2]. Consequently, the fewer the kernels of an application, the easier it is to provide portability.

Commonly, the development of scientific computing software follows a stencil-based approach: the numerical methods and algorithms are introduced into computing systems through stencil data structures and sweeps (*i.e.* traversing element by element the mesh and performing the required computations). In other words, both data structures and computing kernels integrate the numerical method and algorithm. Despite being a very intuitive and versatile approach [3, 4], it hampers the design of applications composed of a reduced number of computing kernels and introduces an enormous complexity when porting codes.

Making an effort to remove the dependencies of the kernels on the numerical method may greatly facilitate the portability. Thus, the numerical method must be fully integrated into the data structures somehow so that computing kernels can operate independently. For instance, the platform portability is achieved in [5] by casting all the operations as matrix multiplications and point-wise operations. Following an algebra-based approach, we replace the traditional stencil data structures and sweeps by algebraic data structures and kernels. The discrete variables are stored in vectors and the discrete operators in sparse matrices. Thus, having the numerical method integrated only on the data, the computing kernels become independent. For instance, the gradient of a scalar field could be computed using several different numerical methods but a single kernel: the sparse matrix-vector multiplication.

Even having a modular code or framework without dependencies on the numerical method, the management of different implementation types is still challenging. In our previous work [6], we presented the HPC^2 (Heterogeneous Portable Code for HPC). It is a fully-portable, algebra-based framework capable of heterogeneous computing with many potential applications in the fields of computational physics and mathematics. As a result, the algorithm of the time-integration phase of Direct Numerical Simulations (DNS) of incompressible turbulent flows relies on simple algebraic kernels such as the matrix-vector multiplication and the linear combination of vectors. Moreover, 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.

In this work, we propose a pure virtual approach for managing the architecture-specific data structure and kernel implementations while providing the users with a unique interface.

2 A PURE VIRTUAL APPROACH FOR MANAGING PLATFORM PORTABILITY

Let us assume we have a framework that provides us with a mesh and discrete fields and differential operators (without going into detail about the numerical method used to discretise). Then consider, for instance, the evaluation of the heat flux as

$$\mathbf{q} = -k\nabla T. \tag{1}$$

Assuming the discrete gradient operator and temperature field are given and, therefore, the size of both the input and output vector spaces are known, we want to be able to write the Equation 1 as

```
q = - k*G*T;
```

where `T` is the discrete temperature field represented as an element of the vector space the gradient maps from, `G` is the discrete gradient operator, `k` is the thermal conductivity and `q` is the discrete heat flux represented as an element of the vector space the gradient maps to. Besides, we want to run the model on, say, both a laptop and a hybrid supercomputer without changing any line of the code. Thus, the variables and operators must rely on data structures and kernels that are appropriately managed in a lower-level code block.

We propose the solution depicted in Figure 1. The three lowest-level objects, `vVector`, `vMatrix` and `vUnit`, are pure virtual classes (*i.e.* they can not be instantiated but are used as base classes); derived classes are created for OpenMP, OpenCL and CUDA implementations. The derived `vVector` and `vMatrix` classes contain the necessary elements to store all the data of a vector and a sparse matrix. The derived `vUnit` classes contain methods to manage the memory of the derived `vVector` and `vMatrix`, and also kernels to operate them.

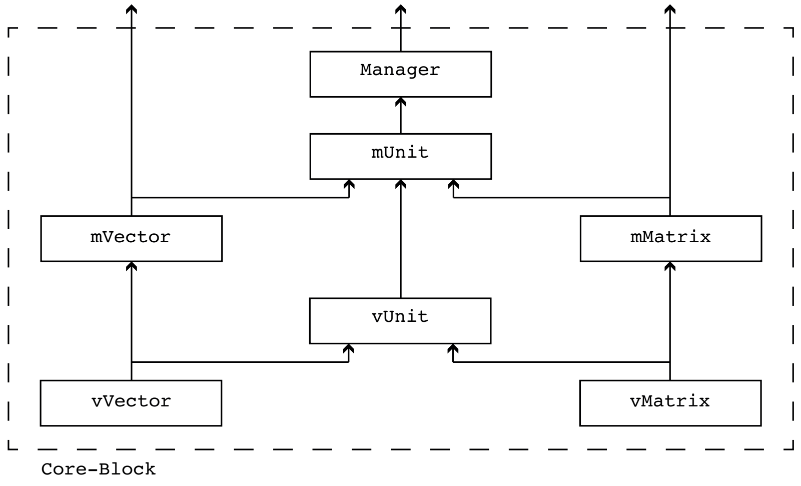

Figure 1: Diagram of classes in our pure virtual approach for managing platform portability.

To deal with heterogeneous computing using a multilevel implementation approach [2, 7] (*i.e.* the local partition of the computational domain within an MPI process is further distributed among its available computing hardware, namely *host* and *devices*), we need a handler that allows working with groups of different derived classes, one for each computing device. The three following objects, `mVector`, `mMatrix` and `mUnit` are the handlers that contain a set of pointers to `vVector`, `vMatrix` and `vUnit` base classes respectively. Finally, the `Manager` is the object in charge of managing the handlers of virtual classes. Given a multilevel partition, the `Manager` can build and operate objects of type `mVector` and `mMatrix` using `mUnit`.

The `Manager` object is unique for every MPI process and is the only interface a user needs to use the data objects and kernels. Consequently, if the multilevel parallelisation is not requested, the handlers are built with only one derived class. Currently, the `Manager` is set up at construction by command-line parameters (further work should allow the `Manager` to have an automatic set up depending on the available hardware and its characteristics). Therefore, considering that we have a higher-level code block that provides with the required `INPUT` data for simplicity (*i.e.* the mesh properties, the discrete gradient operator and the discrete temperature field), the following lines of code may be run on, say, both a laptop and a hybrid supercomputer:

```
Manager M;

double  k = INPUT_K;
mVector q = M.CreateVector(INPUT_FACES);
mVector T = M.CreateVector(INPUT_CELLS, INPUT_T);
mMatrix G = M.CreateMatrix(INPUT_FACES, INPUT_CELLS, INPUT_G);

q = - k*G*T;
```

where `INPUT_K` is the value of thermal conductivity, `INPUT_CELLS` and `INPUT_FACES` are the sizes of the input and output vector spaces of the operator `G`, and `INPUT_T` and `INPUT_G` are the values of the temperature field and gradient operator respectively.

In the conference, we are going to present the pure virtual approach in detail and performance analysis on different computing systems. Additionally, we are going to show its application to CFD simulations.

REFERENCES

[1] Jack Dongarra et al. The International Exascale Software Project roadmap. *Int. J. High Perform. Comput. Appl.*, 25(1):3–60, feb 2011.

[2] Andrey Gorobets, S. Soukov, and P. Bogdanov. Multilevel parallelization for simulating compressible turbulent flows on most kinds of hybrid supercomputers. *Comput. Fluids*, 2018.

[3] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Comput. Phys.*, 12(6):620, 1998.

[4] Frederic Archambeau, Namane Méchitoua, and Marc Sakiz. Code_Saturne : a Finite Volume Code for the Computation of Turbulent Incompressible Flows. *Ind. Appl. Int. J. Finite Vol.*, 1, 2004.

[5] F. D. Witherden, B. C. Vermeire, and Peter E. Vincent. Heterogeneous computing on mixed unstructured grids with PyFR. *Comput. Fluids*, 120:173–186, oct 2015.

[6] Xavier Álvarez, Andrey Gorobets, F. Xavier Trias, Ricard Borrell, and Guillermo Oyarzun. HPC^2–A fully-portable, algebra-based framework for heterogeneous computing. Application to CFD. *Comput. Fluids*, 173:285–292, 2018.

[7] Xavier Álvarez, Andrey Gorobets, and F. Xavier Trias. Strategies for the heterogeneous execution of large-scale simulations on hybrid supercomputers. In *7th Eur. Conf. Comput. Fluid Dyn.*, 2018.