Accelerating Extreme-Scale Numerical Weather Prediction

Willem Deconinck[1(✉)], Mats Hamrud[1], Christian Kühnlein[1], George Mozdzynski[1], Piotr K. Smolarkiewicz[1], Joanna Szmelter[2], and Nils P. Wedi[1]

[1] European Centre for Medium-Range Weather Forecasts, ECMWF, Reading RG2 9AX, UK
{willem.deconinck,mats.hamrud,christian.kuehnlein,george.mozdzynski, piotr.smolarkiewicz,nils.wedi}@ecmwf.int
[2] Wolfson School, Loughborough University, LE11 3TU Loughborough, UK
j.szmelter@lboro.ac.uk

Abstract. Numerical Weather Prediction (NWP) and climate simulations have been intimately connected with progress in supercomputing since the first numerical forecast was made about 65 years ago. The biggest challenge to state-of-the-art computational NWP arises today from its own software productivity shortfall. The application software at the heart of most NWP services is ill-equipped to efficiently adapt to the rapidly evolving heterogeneous hardware provided by the supercomputing industry. If this challenge is not addressed it will have dramatic negative consequences for weather and climate prediction and associated services. This article introduces Atlas, a flexible data structure framework developed at the European Centre for Medium-Range Weather Forecasts (ECMWF) to facilitate a variety of numerical discretisation schemes on heterogeneous architectures, as a necessary step towards affordable exascale high-performance simulations of weather and climate. A newly developed hybrid MPI-OpenMP finite volume module built upon Atlas serves as a first demonstration of the parallel performance that can be achieved using Atlas' initial capabilities.

Keywords: Numerical weather prediction · Climate simulation · High performance computing · Exascale

1 Introduction

Numerical Weather Prediction (NWP) and climate simulations have been intimately connected with progress in supercomputing since the first numerical forecast was made about 65 years ago. In the early days, computing power was gained through taking advantage of vector instructions on a large single-processor computer. With little modification to existing code bases, shared memory parallelization was introduced when more and more processors were added to one computer. When multi-node architectures became the norm, however, the required effort to

R. Wyrzykowski et al. (Eds.): PPAM 2015, Part II, LNCS 9574, pp. 583–593, 2016.
DOI: 10.1007/978-3-319-32152-3_54

port existing code to make use of distributed memory was significant, and often meant a rewrite or redesign of large parts of the code. Over a decade later, many NWP codes have grown in size to millions of lines making use of hybrid parallelization with distributed and shared memory on CPU architectures. As the traditional approaches to boosting CPU performance ran out of room [1], more nodes are being added to computing clusters, putting more and more strain on distributed memory parallelism, as well as ever increasing the energy bill. Hardware vendors are introducing alternatives to computing on traditional CPU architectures in order to keep increasing FLOP rates at reduced energy costs; for example offloading computations to GPU's or Intel MIC's. These alternatives are of less general purpose, and significant effort is required to adapt the existing code base to optimally use the hardware. The application software at the heart of most NWP services is ill-equipped to efficiently adapt to these rapidly evolving heterogeneous hardware. It is furthermore not yet clear which hardware, or combination of hardware will offer the best performance at minimal cost.

To address the challenge of extreme-scale, energy-efficient high-performance computing, the European Centre for Medium-Range Weather Forecasts (ECMWF) is leading the ESCAPE project, a European funded project involving regional NWP centres, Universities, HPC centres and hardware vendors, including an enterprise to explore the use of optical accelerators. The aim is to combine interdisciplinary expertise for defining and co-designing the necessary steps towards affordable, exascale high-performance simulations of weather and climate. Key project elements are to develop and isolate basic algorithmic ideas by breaking down NWP legacy codes into Weather & Climate Dwarfs in analogy to the Berkeley Dwarfs [2], and to classify these canonical building blocks with energy-aware performance metrics. The Berkeley Dwarfs identify archetypical computational and communication patterns, defining the priority areas for community research in numerical methods to enhance parallel computing and communication. The Berkeley dwarfs relevant to NWP contain algorithms responsible for: spectral methods, sparse linear algebra, dense linear algebra, unstructured meshes, structured grids, dynamic programming, and graphical models. Weather & Climate Dwarfs share these motifs but go further, by providing practical solutions to support co-design development and on which sustainable NWP services can be built. The distinct motivation to define Weather & Climate Dwarfs is to focus on maximizing computing performance with minimal energy consumption, i.e. the cost to solution, enveloped by the very strict requirement that NWP models need to run at speeds 200–300 times faster than real time. The ideas developed to facilitate extreme-scaling for the Weather & Climate Dwarfs, could then be implemented back in NWP models.

The Integrated Forecasting System (IFS) is one of such NWP models developed at the ECMWF. It relies on spectral transforms and corresponding computations in spectral space to evaluate horizontal derivatives. Spectral transforms on the sphere involve discrete spherical-harmonics transformations between physical (grid-point) space and spectral (spherical-harmonics) space. Extreme scaling performance of spectral transforms is crucial for its applicability in IFS

on future hardware. Spectral transforms are identified as one of the Weather & Climate Dwarfs. Spectral transforms require data-rich global communication, and it is ultimately the communication overhead and not the computational burden that will limit the applicability of the spectral transform method at extreme scale. This is illustrated in Fig. 1 for 5 km and 2.5 km IFS simulations, where the communication cost amounts to 75 % of the transform cost on TITAN, the number 2 HPC system in the top 500 super computing list as in May 2015. To put these resolutions in perspective with the ECMWF operational requirements of solving 240 forecast days per day (or 10 days in 1 h), the 2.5 km resolution simulation is estimated to require 270,000 cores of a Cray XC-30 HPC, whereas the 5 km resolution simulation requires 25,000 cores to achieve this goal.

The cost of computing derivatives with the spectral transform method with the current state-of-the-art spectral transforms is still relatively high and hence the use of derivatives has been minimized carefully [3]. As part of the ESCAPE project, the spectral transform algorithms will be heavily scrutinized and improved upon by e.g. overlapping communications with computations. Hardware vendors will work together with scientists, and explore the use of bespoke accelerators based on optical computations.

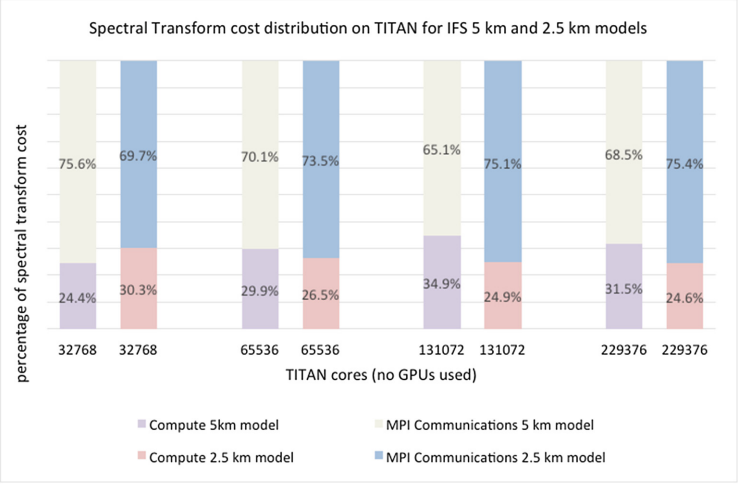


Fig. 1. Spectral transform cost distribution on TITAN for IFS 5 km and 2.5 km models. The total spectral transform cost on TITAN was in the range 29 to 51 % of the total wall time.

Notwithstanding the outcome of further applicability of the spectral transform method, alternative grid-point discretisation methods are desired to compute derivatives locally with compact grid-point stencils. Computations using compact grid-point stencils require only local — nearest neighbor — communication, which inherently scales much better and may better represent

the locality of a physical process. Such discretisation methods are typically based on meshes composed of elements such as triangles, quadrilaterals, or lines. Examples include the finite volume method (FV), the finite element method (FE), or higher-order (>2) methods such as the Discontinuous Galerkin method (DG) and the Spectral Difference method (SD).

Atlas, a flexible parallel data structure framework that can handle a number of different structured grids and unstructured meshes, is being developed at the ECMWF, both to support research on alternative grid-point discretisation methods, as well as serving as the basis for leveraging the anticipated integration efforts resulting from the ESCAPE project. This Atlas framework will be introduced in detail in Sect. 2.

2 Atlas – A Flexible, Scalable, and Sustainable Model Infrastructure

2.1 Motivation

In order for NWP applications to optimally exploit future computer hardware emerging over the next 20–30 years, a flexible and dynamic data structure framework, named *Atlas* is being developed to serve as a foundation for a wide variety of applications, ranging from the use within European NWP models and for the development of alternative dynamical core modules [4], to development of applications responsible for pre- and post-processing of exponentially growing output data. It is imperative for Atlas to remain flexible and maintainable since the development of NWP models typically takes a decade, and NWP models typically last much longer than that.

The *Atlas* framework provides parallel distributed, flexible, object-oriented data structures for both structured grids and unstructured meshes on the sphere. It separates concerns of mathematical model formulation and numerical solutions from the cumbersome management of unstructured meshes, distributed memory parallelism and input/output of data. It is recognized that handling flexible object-oriented structures and carefully controlled memory-management, as would be required with the expected deepening of memory hierarchies in future hardware, is not easily achieved with the Fortran language, which is currently widely used in most NWP models. Hence, the language of choice for *Atlas* is C++, a highly performant language providing excellent object-orientation support, and building upon C's memory management proficiency. A Fortran2003 interface exports all of *Atlas*' functionality to Fortran applications, hence seamlessly introducing modular object-oriented concepts to legacy NWP models. Moreover, many other C++ based applications can directly benefit from *Atlas*.

2.2 Design

The application (e.g. the NWP model) starts by instructing *Atlas* to construct a *Grid* object, which provides a description of all longitude-latitude nodes in

the model domain. The *Grid* object may not require significant memory in case it describes a structured grid in which simple formulas provide the longitude and latitude of every node on the sphere. In contrast, a *Mesh* object makes no assumption on any structure of the grid, and actually stores the horizontal coordinates of every node. Optionally, elements such as triangles, quadrilaterals and lines can be created by providing element-to-node connectivity tables. As the coordinates and the connectivity tables can have a significant memory footprint for large meshes, the *Mesh* is a distributed object, meaning that the mesh is subdivided in partitions which reside in memory on different parallel processes. With the *Grid* and the provision of a distribution scheme, *Atlas* generates a distributed *Mesh*. Specialized *FunctionSpace* objects can be created on demand, using the *Mesh*. A *FunctionSpace* describes in which manner *Fields*, objects that hold the memory of a full scalar, vector or tensor field, are discretized on the mesh. A straightforward *FunctionSpace* is the one where fields are discretized in the nodes of the grid. Other *FunctionSpace* objects could describe fields discretized in cell-centres of triangles and quadrilaterals, or in edge-centres of these elements. Yet another type of *FunctionSpace* can describe spectral fields in terms of spherical harmonics. *Fields* are stored contiguously in memory as a one-dimensional array and can be mapped to an arbitrary indexing mechanism to cater for the specific memory layout of NWP models, or a different memory layout that proves beneficial on emerging computer hardware. *Fields* are addressed by user-defined names and can be associated to *Metadata* objects, which store simple information like the units of the field or a time stamp. It is this flexibility and object-oriented design that leads to more maintainable and sustainable future-proof code. Figure 2 illustrates the object-oriented design and the relevant classes.

2.3 Massively Parallel Distribution

The MPI parallelisation relies on the distribution of the computational mesh with its unstructured horizontal index. The *Atlas* framework is responsible for generating a distributed mesh, and provides communication patterns using MPI to exchange information between the different partitions of the mesh. To minimise the cost of sending and receiving data, the distribution of the mesh is based on an equal regions domain decomposition algorithm optimal for a quasi-uniform node distribution on surface of the sphere [5,6]. The equal regions domain decomposition divides the sphere into bands oriented in zonal direction, and subdivides each band in a number of regions so that globally each region has the same number of nodes. Notably, the bands covering the poles are not subdivided, forming two polar caps. Figure 3 shows the partitioning of a spherical mesh with ∼6.6 million nodes, quasi-uniformly distributed with grid spacing ∼9 km, in 1600 partitions, the anticipated number of MPI tasks for this model resolution. The vertical direction is structured and not parallelized.

Because global communication across the entire supercomputing cluster are foreseen to become prohibitively expensive (see Fig. 1), numerical algorithms may be required to limit communication to – even physically located – nearest

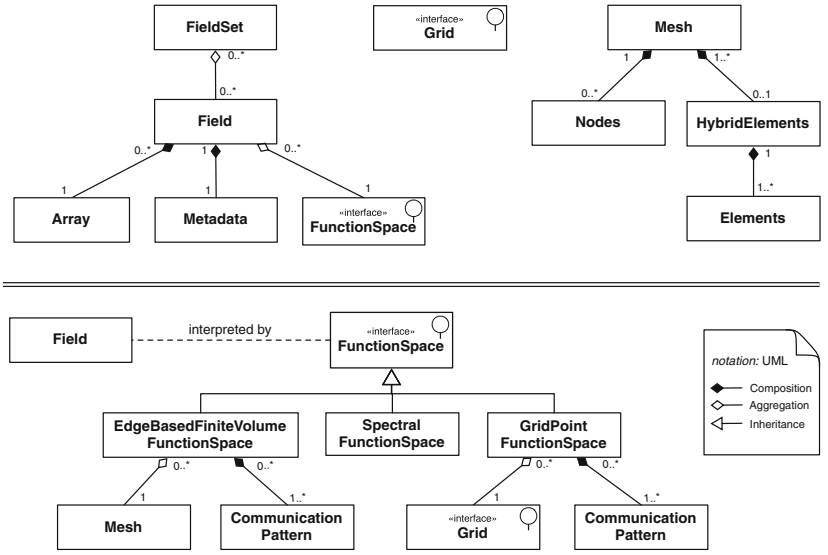


Fig. 2. Atlas data structure design.

Fig. 3. Equal regions domain decomposition (1600 partitions) for a fine mesh with ∼6.6 million nodes (∼9 km grid spacing).

neighbors. Such communication typically happens through thin halos (of 1 element wide) surrounding every mesh partition, hence creating an overlap with neighboring partitions. *Atlas* provides routines that expand the mesh partitions with these halos and provides communication patterns to update field values in the halos with field values from neighboring partitions. Figure 4 shows the equal regions domain decomposition into 32 partitions for a coarse mesh with ∼3500 nodes, quasi uniformly distributed with grid spacing ∼400 km, projected on a

longitude-latitude domain. Figure 4 also illustrates the creation of internal and periodic halos. With periodic halos the periodicity in a longitude-latitude domain is treated exactly like any other internal boundary between different partitions. Notably, the partitions involving the poles are periodic with themselves.

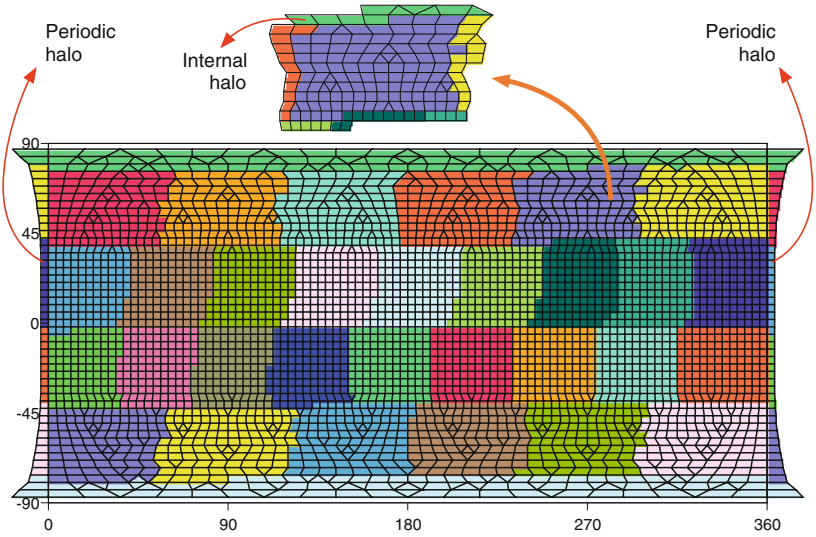


Fig. 4. Equal regions domain decomposition (32 partitions) for a coarse mesh with ∼3500 nodes (∼400 km grid spacing), projected on a longitude-latitude domain. Each partition is surrounded by a thin halo. Periodicity is provided through periodic halos.

3 Application in a Hybrid Finite Volume Module

As mentioned in Sect. 1, the NWP and climate models using the spectral transform method require global communication of large amounts of data. Realising the performance limitations of spectral transform methods at scale, a three-dimensional hybrid finite difference - finite volume module is developed [4], which only relies on nearest neighbor communication patterns as illustrated in Fig. 4. In the horizontal direction an unstructured median-dual edge-based finite volume method is used [7], while the vertical direction is discretized with a structured finite-difference method preferred with the vertically shallow nature of the atmosphere compared to the radius of the earth. Although capable to use arbitrary unstructured meshes in horizontal direction, bespoke meshes (such as in Fig. 4) based on reduced Gaussian grids [8,9] are generated, i.e. the location of its nodes coincides with those of the reduced Gaussian grid, which still facilitates the use of the spectral transform method. The generated mesh defines control volumes for a finite volume method while sharing the same data points as used in

the spectral transform method. This approach allows a hybrid and evolutionary avenue to reach extreme scale, in which the finite volume module can initially be coupled to add extra functionality to the spectral transform model IFS, and could replace the spectral method when it appears to be advantageous in terms of performance, energy-efficiency, and forecast skill.

3.1 Memory Layout for Hybrid Parallelization

Additional performance can be gained using shared-memory parallelization techniques such as OpenMP, on multi-core architectures in which cores are distributed over nodes sharing the same memory. With recent hardware developments such as the Intel MIC's, the number of cores in one processor socket is increasing. The use of shared memory parallelisation when possible is often preferred to reduce the load with the MPI communications. The memory layout of fields plays an important role in optimal performance, especially with shared memory parallelization techniques.

In memory, a field is stored as a one-dimensional array that can be reinterpreted as a multi-dimensional array. The memory layout of a full 3D field can hence be reinterpreted with the horizontal unstructured index as the slowest moving index, followed by the structured vertical index, and the fastest index being the number of variables a field contains (e.g. scalar = 1, vector = 3, tensor = 9). For a scalar field, the "variable" index can be ignored. The memory layout for a three-dimensional scalar field is sketched in Fig. 5.

The advantage of this memory layout is twofold. First, it makes the vertical columns contiguous in memory, so that a halo-exchange involves contiguous chunks of memory and makes the packing and unpacking of send/receive buffers more efficient. Second, it favours the outer loop to be over the columns and

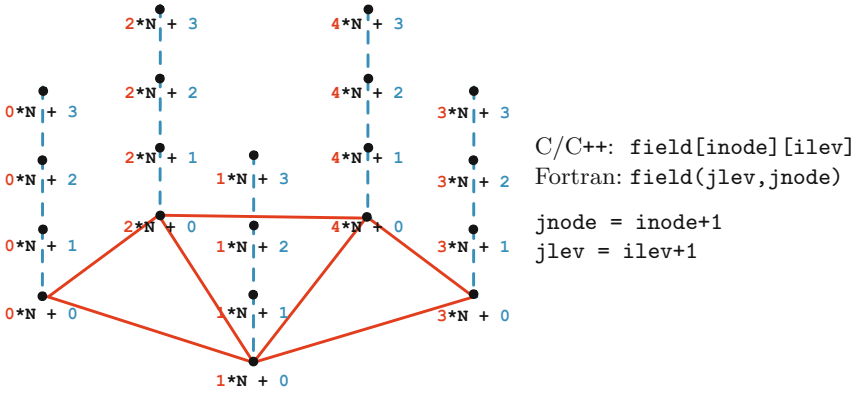
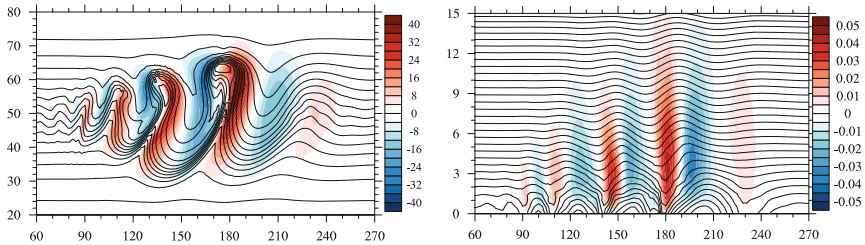


Fig. 5. Memory layout of a scalar field in 3D domain: solid lines show the underlying unstructured mesh; dashed lines mark the structured vertical columns with N denoting the number of vertical levels; and dots, numbered with memory offsets from the first index of the 1D array, represent the field values.

the inner loop to be over the levels within each column, with corresponding indices jnode and jlev, Fig. 5. Due to the unstructured nature of the horizontal jnode index, indirect addressing is required to access neighboring column data. The horizontal index for the outer loop then reduces the cost of this lookup by reusing the node specific computations for the entire column in the inner loop, giving the compiler the opportunity to optimise the inner loop in the vertical direction further with vector instructions, provided that computations for each vertical level are independent of each other. By using shared-memory parallelization with OpenMP over the outer horizontal index, further need for distributing the mesh is avoided.

3.2 Three-Dimensional Compressible Non-hydrostatic Dynamics

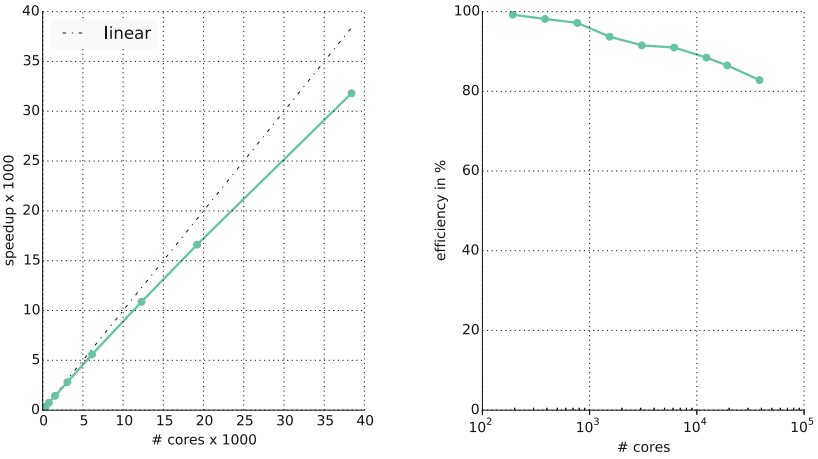
To illustrate the current status of the hybrid finite volume module, a baroclinic instability test case [10] is simulated to day 8 with results shown in Fig. 6. The simulation is three-dimensional and uses compressible non-hydrostatic dynamics. For a full description of the methodology, the reader is referred to [4].



a) Horizontal slice b) Vertical slice

Fig. 6. Simulation result using hybrid finite volume module for baroclinic instability test case [10] after 8 days. (a) Horizontal slice at surface showing meridional velocity (m/s). (b) Vertical slice at latitude ∼53° N (X-axis in degrees, Y-axis in km) with shading showing vertical velocity (m/s) and solid contour lines showing isentropes (5 K interval).

The hybrid finite volume module is expected to perform well at extreme-scale due to local nearest neighbor communication patterns. Preliminary strong scaling results are reported in Fig. 7. These preliminary strong scaling results show acceptable promise with a parallel efficiency of 82 % at 40,000 CPU cores on ECMWF's Cray XC-30 HPC. It is expected that further improvement is possible by overlapping communication and computation, and by more carefully grouping communication buffers together.



a) Speedup b) Efficiency

Fig. 7. Preliminary strong scaling results for the hybrid finite-volume module on ECMWF's Cray XC-30 HPC. The results are created using a setup for solving hydrostatic equations, on a mesh with ∼5.5 million nodes and 137 vertical levels.

4 Remarks

With a variety of emerging computer architectures to tackle the challenge of extreme scale computing, a drastic change in the NWP and climate software stack is needed. A new model and datastructure framework called *Atlas* is developed for the NWP and climate community, forming the foundation for new research to alternative dynamical cores on one hand, and for an evolutionary modernisation of legacy NWP and climate codes on the other hand. During the ESCAPE project, this framework will be further developed and serve as a common infrastructure for leveraging the anticipated integration effort of the developments by the various partners involved. At the ECMWF, *Atlas* is already being succesfully used in a hybrid finite volume module. This finite volume module already shows promising scaling results, relying on *Atlas* for distributed memory parallelization and data management.

Acknowledgments. This work was supported in part by funding received from the European Research Council under the European Union's Seventh Framework Programme (FP7/2012/ERC Grant agreement no. 320375), and in part by the CRESTA project that has received funding from the European Community's Seventh Framework Programme (ICT-2011.9.13) under Grant Agreement no. 287703.

References

1. Sutter, H.: The free lunch is over: a fundamental turn toward concurrency in software. Dr. Dobb's J. **30**, 202–210 (2005)
2. Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W., Yelick, K.A.: The Landscape of Parallel Computing Research: A View from Berkeley. Technical Report, UC Berkeley (2006)
3. Wedi, N.P., Hamrud, M., Mozdzynski, G.: A fast spherical harmonics transform for global NWP and climate models. Mon. Weather Rev. **141**, 3450–3461 (2013)
4. Smolarkiewicz, P.K., Deconinck, W., Hamrud, M., Kühnlein, C., Mozdzynski, G., Szmelter, J., Wedi, N.P.: A hybrid all-scale finite-volume module for stratified flows on a rotating sphere. J. Comput. Phys. (2016, submitted)
5. Leopardi, P.: A partitioning of the unit sphere of equal area and small diameter. Electron. Trans. Numer. Anal. **25**, 309–327 (2006)
6. Mozdzynski, G.: A new partitioning approach for ECMWF's integrated forecasting system (IFS). In: Proceedings of the Twelfth ECMWF Workshop: Use of High Performance Computing in Meteorology, 30 October - 3 November 2006, Reading, UK, pp. 148–166, pp. 259–273. World Scientific (2007)
7. Szmelter, J., Smolarkiewicz, P.K.: An unstructured mesh discretisation in geospherical framework. J. Comput. Phys. **229**, 4980–4995 (2010)
8. Hortal, M., Simmons, A.J.: Use of reduced Gaussian grids in spectral models. Mon. Weather Rev. **119**, 1057–1074 (1991)
9. Wedi, N.P.: Increasing horizontal resolution in numerical weather prediction and climate simulations: illusion or panacea? Phil. Trans. R Soc. A **372**, 20130289 (2013). doi:10.1098/rsta.2013.0289
10. Jablonowski, C., Williamson, D.L.: A baroclinic instability test case for atmospheric model dynamical cores. Q. J. Roy. Meteorol. Soc. **132**, 2943–2975 (2006)