The ESCAPE project: Energy-efficient Scalable Algorithms for Weather Prediction at Exascale

Andreas Müller[1], Willem Deconinck[1], Christian Kühnlein[1], Gianmarco Mengaldo[1], Michael Lange[1], Nils Wedi[1], Peter Bauer[1], Piotr K. Smolarkiewicz[1], Michail Diamantakis[1], Sarah-Jane Lock[1], Mats Hamrud[1], Sami Saarinen[1], George Mozdzynski[1], Daniel Thiemert[1], Michael Glinton[2], Pierre Bénard[2], Fabrice Voitus[2], Charles Colavolpe[2], Philippe Marguinaud[2], Yongjun Zheng[2], Joris Van Bever[3], Daan Degrauwe[3], Geert Smet[3], Piet Termonia[3,4], Kristian P. Nielsen[5], Bent H. Sass[5], Jacob W. Poulsen[5], Per Berg[5], Carlos Osuna[6], Oliver Fuhrer[6], Valentin Clement[7], Michael Baldauf[8], Mike Gillard[9], Joanna Szmelter[9], Enda O'Brien[10], Alastair McKinstry[10], Oisín Robinson[10], Parijat Shukla[10], Michael Lysaght[10], Michał Kulczewski[11], Milosz Ciznicki[11], Wojciech Piątek[11], Sebastian Ciesielski[11], Marek Błażewicz[11], Krzysztof Kurowski[11], Marcin Procyk[11], Pawel Spychala[11], Bartosz Bosak[11], Zbigniew Piotrowski[12], Andrzej Wyszogrodzki[12], Erwan Raffin[13], Cyril Mazauric[13], David Guibert[13], Louis Douriez[13], Xavier Vigouroux[13], Alan Gray[14], Peter Messmer[14], Alexander J. Macfaden[15], and Nick New[15]

[1]European Centre for Medium-Range Weather Forecasts (ECMWF), Reading, UK
[2]Centre National de Recherches Météorologiques, Météo-France, Toulouse, France
[3]Royal Meteorological Institute (RMI), Brussels, Belgium
[4]Department of Physics and Astronomy, Ghent University, Ghent, Belgium
[5]The Danish Meteorological Institute (DMI), Copenhagen, Denmark
[6]Federal Office of Meteorology and Climatology MeteoSwiss, Zurich, Switzerland
[7]Center for Climate System Modeling, Zurich, Switzerland
[8]Deutscher Wetterdienst (DWD), Offenbach, Germany
[9]Loughborough University, Leicestershire, UK
[10]Irish Centre for High-End Computing (ICHEC), National University of Ireland, Galway, Ireland
[11]Poznan Supercomputing and Networking Center (PSNC), Poznań, Poland
[12]Institute of Meteorology and Water Management - National Research institute (IMGW-PIB), Warsaw, Poland
[13]ATOS, Bezons, France
[14]NVIDIA Switzerland, Zurich, Switzerland
[15]Optalysys Ltd., Glasshoughton, UK

Correspondence: Andreas Müller (andreas.mueller@ecmwf.int)

Abstract. In the simulation of complex multi-scale flows arising in weather and climate modelling, one of the biggest challenges is to satisfy strict service requirements in terms of time-to-solution and energy-to-solution, without compromising the accuracy and stability of the calculation. These competing factors require the development of robust algorithms that can optimally exploit the targeted underlying hardware and efficiently deliver the extreme computational efforts typically required in operational forecast production. These algorithms should: (i) minimise the energy footprint along with the time required to produce a solution; (ii) maintain the scientifically required level of accuracy; and (iii) be numerically stable, and resilient in case of hardware or software failure.

The European Centre for Medium Range Weather Forecasts (ECMWF) is leading a project called ESCAPE (Energy-efficient SCalable Algorithms for weather Prediction on Exascale supercomputers) which is funded by Horizon 2020 (H2020) under initiative Future and Emerging Technologies in High Performance Computing (FET-HPC). The goal of the ESCAPE project is to develop a sustainable strategy to evolve weather and climate prediction models to next-generation computing technologies. The project partners incorporate the expertise of leading European regional forecasting consortia, university research, experienced high-performance computing centres and hardware vendors.

This paper presents an overview of this ESCAPE strategy: (i) identify domain-specific key algorithmic motifs in weather prediction and climate models (Weather & Climate dwarfs), (ii) categorise them in terms of computational and communication patterns, while (iii) adapting them to different hardware architectures with alternative programming models, (iv) analyse the challenges in optimising and (v) finding alternative algorithms for the same scheme. The participating weather prediction models are: IFS (Integrated Forecasting System); ALARO, a combination of AROME (Application de la Recherche à l'Opérationnel à Meso-Echelle) and ALADIN (Aire Limitée Adaptation Dynamique Développement International); and COSMO-EULAG, a combination of COSMO (Consortium for Small-scale Modeling) and EULAG (Eulerian/semi-Lagrangian fluid solver). For many of the Weather & Climate dwarfs ESCAPE provides prototype implementations on different hardware architectures (mainly Intel Skylake CPUs, NVIDIA GPUs, Intel Xeon Phi) with different programming models. The spectral transform dwarf represents a detailed example of the co-design cycle of an ESCAPE dwarf.

1 Introduction

Numerical weather and climate prediction capabilities represent substantial socio-economic value in nearly all sectors of human society, namely for the mitigation of the impact of extremes, in food production, renewable energy and water management, infrastructure planning, and for finance and insurance where weather sensitive goods and services are traded. Despite significant progress achieved over past decades (Bauer et al., 2015) there are substantial shortcomings in our ability to predict, for example, weather extremes with sufficient lead time and the impact of climate change at regional or national level. Extreme weather events caused over 500 thousand casualties and over 2 trillion $US economic damages in the past 20 years (Wallemacq et al., 2018). Another example is the prediction of climate change mitigation and adaptation targets. Failure to meet these targets is ranked among the leading threats to global society (World Economic Forum, 2019).

One of the key sources of model error is limited spatial (and temporal) resolution (Palmer, 2014) which implies that key physical processes that drive global circulation, like deep convection in the tropics and mesoscale eddies in the ocean, are only crudely represented in models by so-called parameterisations. In addition, deficiencies in the representation of process interactions between atmosphere and ocean/sea-ice, as well as atmosphere and land, including a time-varying biosphere, are highly relevant strategic targets to improve the representation of the internal variability of the Earth system in models (Katzav and Parker, 2015). However, better spatial resolution and enhanced model complexity translate immediately into significant computational challenges (Schulthess et al., 2019), whereby spatial resolution is the most demanding because a doubling of resolution roughly translates into eight times more computations (doubling the horizontal resolution in both directions and

a corresponding decrease in the time step). The critical resolution threshold at which global weather & climate models may eventually be able to overcome the bulk of the limiting deficiencies is unclear; however, O(1km) emerges as a defendable intermediate target (Shukla et al., 2010; Neumann et al., 2019).

The ESCAPE project has been motivated by the need for running Earth-system models at much higher resolution and complexity than presently available, but within the same time-critical path of daily production for weather forecasts and with the same production throughput needed for decadal/centennial climate projections as used today. This translates to computing one simulated year per wallclock day. Energy efficiency is a key requirement as power envelopes for HPC systems cannot be scaled up at the same rate as the computing demand. Obviously, this presents a substantial challenge to all aspects of computing, namely the choice and implementation of numerical methods and algorithms, and the programming models to map memory access and communication patterns onto specific hardware (Wehner et al., 2011).

The ECMWF specific strategic target is to provide skilful predictions of high-impact weather up to two weeks ahead by 2025 based on a global 5 km-scale Earth-system model ensemble, complemented by the corresponding, high-resolution data assimilation system for creating the initial conditions. This system needs sufficient model complexity in the atmosphere (physics and chemistry), oceans, sea-ice and land to represent all processes acting on such scales, and a sufficiently large ensemble size so that complex probability distributions of extreme forecast features can be sampled well enough. In terms of computational demand this is comparable to the above 1-km resolution target for a single forecast.

The supercomputers used for numerical weather prediction (NWP) have changed dramatically over the past decades, and ECMWF's Integrated Forecasting System (IFS; Wedi et al., 2015, and references therein) has exhibited rather different levels of sustained performance ranging from 40-50% on the parallel vector-processor Cray-XMP, YMP and C90 architectures between 1984 and 1996, to 30% on Fujitsu's similar-type VPP-700 and 5000 systems between 1996 and 2002, down to 5-10% on the scalar multi-core IBM P5-7 and Cray XC30 and 40 architectures operated between 2002 and today. Despite sustained performance declining, overall FLOP performance increased exponentially thanks to Moore's law, Dennard scaling and processor pricing (e.g. Flamm, 2018) so that significant upgrades of model resolution, complexity and ensemble size remained affordable in terms of capital cost and power supply. As this 'natural' technological evolution is slowing down, new concepts for designing algorithms and for mapping the associated computational patterns onto the available architectures - from processor to system level - are needed (Lawrence et al., 2018). Many of the algorithms used in NWP were designed well before the multi-core era started, but even though they contain highly-tuned shared and distributed memory parallelisation they only achieve such limited sustained performance because of poor arithmetic density (ratio of data communication / computations) in some of the highly varying kernels, sequential tasking due to strong scientific dependencies, and rather complex algorithms required to solve a multi-scale and multi-phase fluid dynamics problem on a rotating sphere with sufficient accuracy.

If the envisioned increase in model fidelity is constrained by only marginally growing power envelopes and decelerating general purpose processor speed, then performance issues need to be addressed at the root, and a more radical redesign of the basic algorithms and codes used for weather prediction needs to be considered. This is why ESCAPE investigates both HPC adaptation and alternative numerical formulations in scientifically and computationally well defined model components,

the Weather & Climate dwarfs, followed by a detailed analysis of their respective computational bottlenecks, and subsequent hardware and algorithm dependent optimisations.

The Weather & Climate dwarf idea is introduced in Section 2. In Section 3 we illustrate the usefuleness of the dwarf concept with the example of the spectral transform dwarf. The paper ends with conclusions and outlook in Section 4.

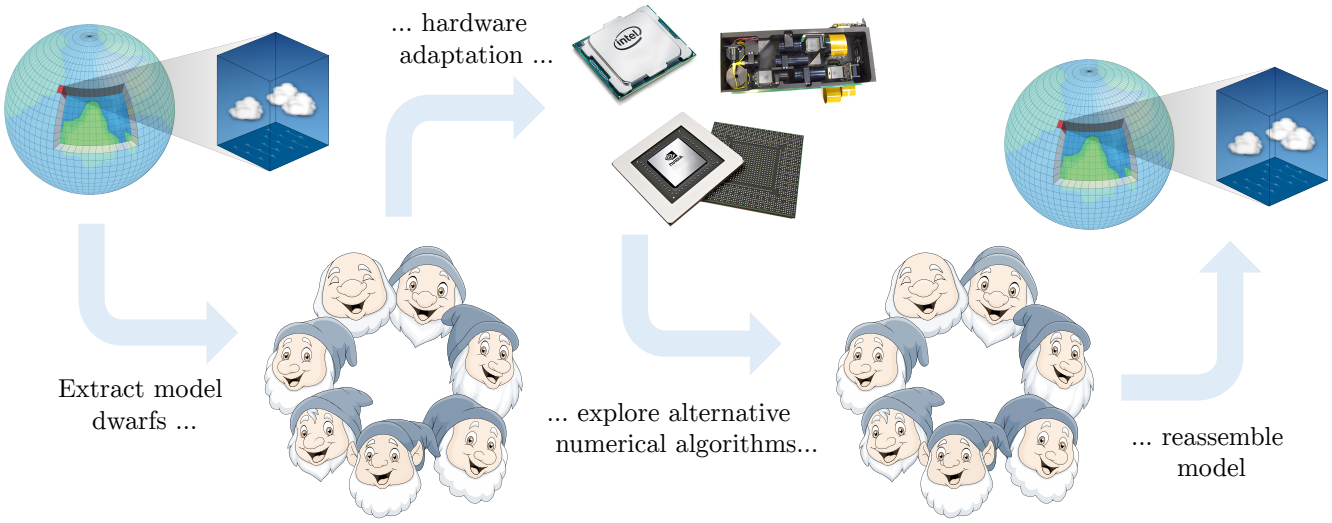
2 Weather & Climate dwarfs

2.1 Motivation

In 2004 Phillip Colella introduced the seven dwarfs of algorithms for high-end simulation in the physical sciences (Colella, 2004). These were later extended to the 13 Berkeley dwarfs (Asanović et al., 2006, 2009) which are meant to represent characteristic patterns of computation and communication. These dwarfs were created to cover the characteristic computational properties of a broad range of scientific applications. These dwarfs are the basis for the OpenDwarfs benchmark suite (Feng et al., 2012; Krommydas et al., 2015; Johnston and Milthorpe, 2018) and were also applied to benchmark cloud computing in Phillips et al. (2011). In a similar fashion Kaltofen (2011) introduced the seven dwarfs of symbolic computation.

Following this idea, we categorise key algorithmic motives specific to weather prediction and climate models and identify their specific computational and communication patterns, which in return are crucial for the entire model performance. The dwarfs thus represent domain specific mini-applications (Messer et al., 2016) which include direct input from the domain scientist, documentation, timers for profiling purposes as well as error estimates for verification purposes. In this way the dwarfs facilitate communication of weather domain specific knowledge and algorithmic motifs with specialists in other domains. Different implementations of the dwarfs can be used as a first step towards optimising and adapting weather prediction and climate models to new hardware architectures and to benchmark current and future supercomputers with these simple but relevant applications. Identifying these key algorithms also allows better collaboration between operational weather prediction centres, hardware vendors and academia because they are much simpler to understand than the full models. The concept of dwarfs is different from the existing separation of weather and climate models into different model components, such as atmosphere, land-surface and ocean for which separate dynamical core and physical parameterisation packages already exist. Instead, dwarfs define a runnable and more manageable sub-component in a hierarchy of model complexity for specific targets such as adaptation to GPUs, exploring alternative programming models, and developing performance portable domain specific languages. But dwarfs can also be used by domain scientists for developing alternative algorithms.

The fundamental starting point of the ESCAPE project is to identify the dwarfs in the participating weather and climate models (Figure 1) and to adapt them to different hardware architectures. The knowledge gained in this adaptation is used to research alternative numerical algorithms which are better suited for those new architectures, and experiment with alternative programming models, towards improving the overall energy-efficiency of weather prediction applications.



Earth illustration: used under license from GraphicsRF/Shutterstock.com.
Dwarf illustrations: used under license from Teguh Mujiono/Shutterstock.com

Figure 1. Illustration of the main idea behind the ESCAPE project. The entire model is broken down into smaller building blocks called dwarfs. These are adapted to different hardware architectures. Based on the feedback from hardware vendors and high performance computing centres alternative numerical algorithms are explored. These improvements are eventually built into the operational model.

2.2 List of dwarfs created in ESCAPE

Table 1 gives an overview of the dwarfs defined in the ESCAPE project. These dwarfs have been chosen because they represent key algorithms that are representative of any weather and climate model, or because their computational and communication patterns represent one of the most runtime consuming parts of weather forecasting systems (Figure 2). For many of the dwarfs we created so called prototypes. Each prototype implementation for a specific hardware (by default CPU) comes with documentation (Mengaldo, 2016; Müller et al., 2017), error measures which allow us to quickly see if optimisations affect the results and timers to see the speedup obtained through optimisations. Table 1 also lists the models in which we identified the dwarf and from which the prototype implementations originated. The models include the IFS with the spectral transform method (IFS-ST), the Finite-Volume Module of the IFS (IFS-FVM), ALARO/AROME and COSMO-EULAG. In order to explore alternative discretizations in ESCAPE, we created a new global shallow water model called GRASS (Global Reduced A-grid Spherical-coordinate System). Table 1 further shows which dwarfs have prototypes that are based on the data structure framework Atlas. We use Atlas to ease adaptation to future architectures and avoid code duplication across prototypes. Atlas handles both the mesh generation and parallel communication aspects (Deconinck et al., 2017). In the ESCAPE project, Atlas has been improved and extended by adding support for limited area grids and adding support for accelerators through GridTools (Deconinck, 2017a, b). Table 1 also shows the programming model adopted for each of the dwarf prototypes. In particular, we

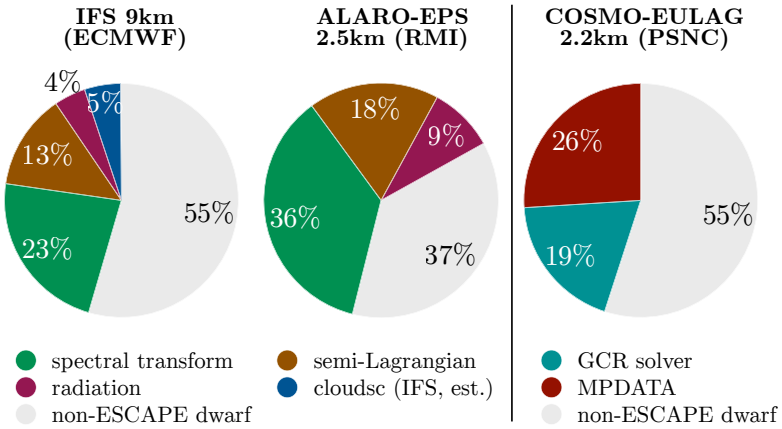


Figure 2. Portion of the forecast runtime spent in ESCAPE dwarfs for the three different models IFS (left), ALARO ensemble prediction system (EPS) (middle) and COSMO-EULAG (right). The measurements for IFS were taken during an operational run on 352 nodes (1408 MPI processes, 18 OpenMP threads per process). The limited area models ALARO-EPS and COSMO-EULAG used each 576 MPI processes for the simulations shown here. IFS and ALARO-EPS are both based on the spectral transform method with latter employing 2D FFTs. For this reason the two left pie charts share one legend. COSMO-EULAG uses different methods and the pie chart has its own legend. The vertical line separates the two different legends.

used MPI for distributed-memory parallelism, OpenMP and OpenACC for shared-memory parallelism, and DSL, that stands for domain-specific language and uses GridTools.

In addition to identifying computational and communication patterns in existing models we also performed research on new algorithms and approaches which have the potential to reduce runtime and energy consumption significantly in the future. We developed a multigrid preconditioner for the Krylov-subspace solver employed in the semi-implicit time integration in IFS-FVM (Müller et al., 2017) to reduce the number of iterations in iterative solves, a HEVI time-integration scheme with significantly improved stability (Colavolpe et al., 2017) to avoid some of the global communication patterns, and connected to this, explored alternative finite difference methods on the sphere (Bénard and Glinton, 2019; Glinton and Bénard, 2019). Finally, we explored FFTs and spherical harmonics on optical (co-)processors (Macfaden et al., 2017) to potentially scale these transforms towards higher resolutions at a fixed energy cost. An overview of the work with the optical processor can be found in section 3.5.

As an example of the cycle involved in identifying, dwarfing, testing, adapting, optimising, and considering alternative solution procedures for a specific algorithmic motif in the ESCAPE project, we shall illustrate in the next section the work on the spectral transform dwarf in detail.

Table 1. Overview of dwarfs identified in the ESCAPE project, short description of the computational characteristics of each dwarf and the models in which we identified the dwarf. Standalone applications including documentation, profiling and verification capabilities called prototypes have been released for many of the dwarfs. The column Atlas shows which of the dwarfs have prototypes that are based on the data structure framework Atlas (Deconinck et al., 2017). Dwarfs for which prototypes with MPI and OpenMP/OpenACC is available can be used with both programming models together as hybrid parallelisation. LAITRI is used in the full model in each MPI process individually and therefore does not include MPI parallelisation inside the dwarf. Further explanations can be found in the references given in the last column.

	computational characteristics	IFS-ST	IFS-FVM	ALARO/AROME	COSMO-EULAG	GRASS	prototype released	Atlas	MPI	OpenMP	OpenACC	DSL (Section 3.7)	Optalysys (Section 3.5)	references
Dynamics dwarfs:														
• time-stepping:														
spectral transform:														
– spherical harmonics	matrix multiplication, global transposition	✓					✓	✓	✓	✓	✓		✓	Mengaldo (2016)
– biFFT	1D and 2D FFT algorithm, global transposition			✓					✓	✓	✓		✓	Mengaldo (2016)
elliptic solver - GCR	iteration, global reduction		✓		✓		✓	✓	✓	✓				Mengaldo (2016)
elliptic solver - multigrid	interpolation for remapping		✓				✓	✓	✓	✓				Müller et al. (2017)
HEVI	nearest neighbour column communication					✓[1]			✓	✓				Mengaldo et al. (2018); Colavolpe et al. (2017)
• advection:														
semi-Lagrangian	large halo communication	✓		✓			✓	✓	✓	✓				Mengaldo (2016)
– LAITRI (3d interpolation)	efficient data access	✓		✓			✓			✓	✓			Robinson et al. (2016)
MPDATA	small halo communication, indirect addressing[2]		✓		✓		✓	✓	✓	✓	✓	✓		Mazauric et al. (2017b); Osuna (2018)
• gradient computation:														
high order finite difference	large halo communication					✓			✓	✓				Bénard and Glinton (2019); Glinton and Bénard (2019)
finite volume	small halo communication, indirect addressing[2]		✓		✓			✓	✓	✓				Kühnlein et al. (2019)
Physics dwarfs:														
• radiation scheme:														
ACRANEB2	many transcendental functions			✓			✓		✓	✓	✓			Müller et al. (2017); Poulsen and Berg (2017)
• cloud microphysics:														
CLOUDSC	single (vertical) column, many if-statements, many local variables	✓	✓				✓		✓	✓	✓			Xiao et al. (2017); Clement et al. (2018)

[1] The GRASS model is currently only solving the shallow-water equations. Research on HEVI time integration methods was performed to explore future options.

[2] when using unstructured meshes

3 Dwarf example: spectral transform

We start with a short description of the (domain-specific) spectral transform and continue with a subsection discussing computational challenges with a particular focus on the data structures and data access patterns used in IFS. After that we present the work that has been done on adapting and optimising the dwarf for GPUs, CPUs and optical processors. We finish with a comparison between the results obtained for the different architectures and a discussion on the sustainability of the chosen techniques.

3.1 Background

Each time-step of IFS is split between computations in physical space (i.e. a grid point representation) and computations with respect to spectral space (i.e. spectral coefficients at different wavenumbers). Semi-Lagrangian advection, physical parameterisations and products of terms are computed most efficiently in grid point space while horizontal gradients, semi-implicit calculations and horizontal diffusion are computed more efficiently in spectral space. The transform between these two spaces is performed on the sphere with spherical harmonics, that is computing these results along longitudes in a Fast Fourier transform (FFT) and a Legendre transform (LT) along latitudes. Limited area models replace the Legendre transform with another FFT which leads to the name biFFT.

In spectral transform methods such as the one used in IFS (Wedi et al., 2013), the specific form of the semi-implicit system facilitating large time-steps (and thus time-to-solution efficiency) is derived from subtracting a system of equations, linearised around a horizontally homogeneous reference state. The solution of this linear system is greatly accelerated by the separation of the horizontal and the vertical part, which matches the large anisotropy of horizontal to vertical grid dimensions prevalent in atmospheric and oceanic models. In spectral transform methods one uses the special property of the horizontal Laplacian operator in spectral space on the sphere

$$\nabla^2 \psi_n^m = -\frac{n(n+1)}{a^2} \psi_n^m, \tag{1}$$

where ψ symbolises a prognostic variable, a is the Earth radius, and (n, m) are the total and zonal wavenumbers of the spectral discretisation (Wedi et al., 2013). This conveniently transforms the 3D Helmholtz problem into an array (for each zonal wavenumber) of 2D matrix operator inversions with the dimension of the vertical levels square, or in the case of treating the Coriolis term implicitly, vertical levels times the maximum truncation, resulting in a very cheap direct solve.

In this paper we focus on the computational aspects and especially on the data layout. We illustrate here the inverse spectral transform on the sphere which goes from spectral space to grid point space. The direct transform adds one numerical integration. Otherwise the only change between inverse and direct transform is that the direct transform starts with the Fourier transformation and applies the Legendre transformation afterwards.

The inverse spectral transform begins with the spectral data $\mathbf{D}(f, \mathrm{i}, n, m)$ which is a function of field index f (for the variable surface pressure at a single level and for wind vorticity, wind divergence and temperature at each height level), real and imaginary part i and wave numbers (zonal wave number $m = 0, \ldots, N_T$ and total wave number $n = 0, \ldots, N_T - m$ where N is

the spectral truncation). Please note that we deviate here from the usual notation where total wavenumber goes from m to N_T because this simplifies the separation between even and odd n. We use here column-major order like in FORTRAN, i.e. the field index f is the fastest moving index and the zonal wave number m is the slowest moving index. Typical dimensions can be seen in the operational high resolution (9km) forecast run at ECMWF: the number of fields is in this case 412 for the direct transform and 688 for the inverse transform and the number of zonal wave numbers is given by the truncation $N_T = 1279$. The number of latitudes is $2N_T + 2 = 2560$ and the number of longitudes increases linearly from 20 next to the poles to $4N_T + 20 = 5136$ next to the equator.

We take advantage of the symmetry of the Legendre polynomials for even n and anti-symmetry for odd n. The coefficients of the Legendre polynomials are pre-computed and stored in $\mathbf{P}_{e,m}(n,\phi)$ for even n and $\mathbf{P}_{o,m}(n,\phi)$ for odd n, where ϕ stands for the latitudes of our Gaussian mesh. Only latitudes on the northern hemisphere are computed. Latitudes on the southern hemisphere are reconstructed from the northern latitudes as we will show later. In the same way we split the spectral data for each m into even part $\mathbf{D}_{e,m}(f,\mathrm{i},n)$ and odd part $\mathbf{D}_{o,m}(f,\mathrm{i},n)$. We write variables over which we can parallelise our computations as indices. The inverse Legendre transform is performed by computing the following matrix multiplications using BLAS:

$$\begin{aligned}\mathbf{S}_m(f,\mathrm{i},\phi) &= \sum_n \mathbf{D}_{e,m}(f,\mathrm{i},n)\cdot\mathbf{P}_{e,m}(n,\phi),\\ \mathbf{A}_m(f,\mathrm{i},\phi) &= \sum_n \mathbf{D}_{o,m}(f,\mathrm{i},n)\cdot\mathbf{P}_{o,m}(n,\phi).\end{aligned} \tag{2}$$

The resulting array for the symmetric and antisymmetric parts are now combined into the Fourier coefficients on the northern and southern hemisphere:

$$\begin{aligned}\phi > 0 &: \mathbf{F}(\mathrm{i},m,\phi,f) = \mathbf{S}_m(f,\mathrm{i},\phi) + \mathbf{A}_m(f,\mathrm{i},\phi),\\ \phi < 0 &: \mathbf{F}(\mathrm{i},m,\phi,f) = \mathbf{S}_m(f,\mathrm{i},\phi) - \mathbf{A}_m(f,\mathrm{i},\phi).\end{aligned}$$

These Fourier coefficients are finally used to compute the fields in grid point space at each longitude λ via FFT:

$$\mathbf{G}_{\phi,f}(\lambda) = \mathrm{FFT}(\mathbf{F}_{\phi,f}(\mathrm{i},m)). \tag{3}$$

3.2 Computational challenges

The computations in grid point space and spectral space require all the fields f to be on the same computational node. The summation over the total wavenumber n in the Legendre transform (2) makes it most efficient to have all total wavenumbers on the same node and the Fourier transform (3) over (i,m) makes it most efficient to have all of the zonal wavenumbers m with real and imaginary part on the same node. This is only possible if the data is transposed before and after the spectral transform as well as in between Legendre and Fourier transform. These transpositions produce substantial communication which increases the contribution of the spectral transform to the overall runtime for future resolutions (Figure 3).

Simplified simulations of the MPI communications performed in ESCAPE show that the strong scalability of the communication time for the spectral transform transpositions is better than for halo communication required by semi-Lagrangian advection and global norm computation commonly used in semi-implicit methods (Zheng and Marguinaud, 2018).

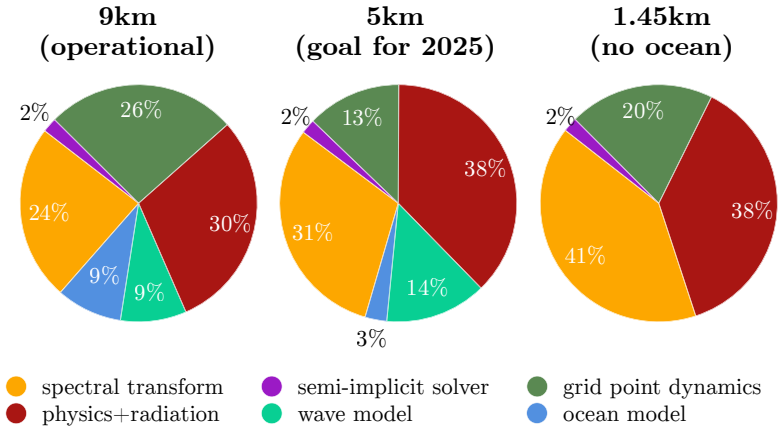


Figure 3. Cost profiles of the significant components of the IFS NWP model in percent of CPU time, at the operational 9km horizontal resolution with 137 vertical levels (left), the anticipated future horizontal resolution of 5km with 137 vertical levels (middle) and an experimental resolution of 1.45km with 62 vertical levels (right). The grid point dynamics represents the advection and gridpoint computations related to the dynamical core, semi-implicit solver represents the computations and communications internal to spectral space, spectral transform relates to the communications and computations in the transpositions from gridpoint to spectral and reverse as well as the FFT and DGEMM computations (see also spectral transform schematic below), physics+radiation relates to the cost of physical parametrisations including radiation, and finally accounting for the additional components of the wave and the ocean model. The simulation at 1.45km is without ocean and waves. All of these profiles have been obtained through measurements on the Cray XC40 supercomputer of ECMWF.

These results indicate that halo communication will become almost as costly as the transpositions in the spectral transform method if we use a very large number of MPI processes. An alternative which avoids transpositions and halo communication is given by the spectral element method shown in Müller et al. (2018) with explicit time integration in the horizontal direction. This leads to a very small amount of data that is communicated in each time-step because this method only communicates the values that are located along the interface between different processor domains. This method, however, requires much smaller time-steps which leads overall to an even larger communication volume (Figure 4). Figure 4 is based on the model comparison presented in Michalakes et al. (2015) and does not include all of the optimisations for the spectral element method presented in Müller et al. (2018). The spectral transform results are based on the operational version of IFS and do not contain the optimisations presented in this paper. Both models have significant potential for optimisation and it is not obvious which method will have the lowest communication volume when fully optimised. The only true solution to avoid waiting time during communication is to overlap different parts of the model such that useful computation can be done while the data is communicated (Mozdzynski et al., 2015; Dziekan et al., 2019).

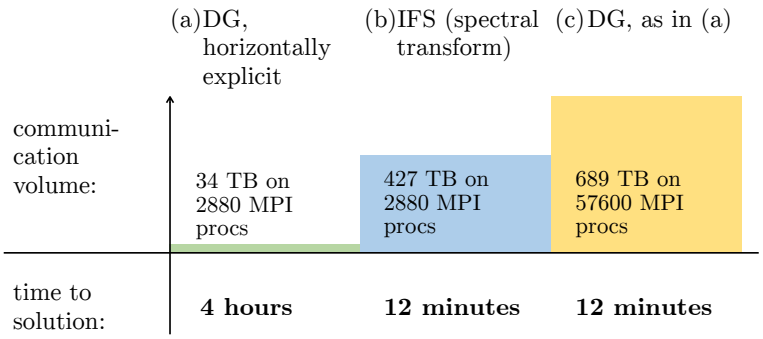


Figure 4. Overall communication volume comparing spectral element (SEM) from Müller et al. (2018) and the global spectral transform methods. The SEM requires a substantially lower amount of communication at the same number of cores, but due to the smaller timestep requires a much higher frequency of repeated communications for the given 2-day simulation. Increasing the number of MPI processes to achieve the same time to solution results in a larger amount of communication for the SEM. Here we assume SEM $\Delta t = 4s$; IFS $\Delta t = 240s$; communication volume is calculated for a 48 hour forecast SEM as 290.4kBytes per MPI task and Δt; IFS as 216mBytes per MPI task and Δt; SEM time-to-solution = 20 x IFS based on the performance results in Michalakes et al. (2015).

3.3 GPU optimisation

For the GPU version, we restructure the prototype to: allow the grid-based parallelism to be fully exposed to the GPU in a flexible manner; ensure that memory coalescing is achieved; and optimise data management. We will now describe each of these in some more detail.

The grid is a two-dimensional sphere, with a third altitude dimension represented by multiple fields at each point on the sphere. The updates are inherently parallel across this grid, so all this parallelism should be exposed to the GPU to get maximal performance. However, the original implementation had a sequential loop over one of the spherical dimensions (at a high level in the call tree of the application). We re-structured the code such that, for each operation, the loops over the three dimensions became tightly nested, and when mapping these to the GPU via OpenACC directives we used the "collapse" clause to instruct the compiler to collapse these to a single loop, such that it can map all inherent parallelism to hardware in an efficient manner. We re-structured the code as follows:

Similarly, for library calls it is important to maximise exposure of parallelism through batching computations using provided interfaces. On the GPU we perform all of matrix multiplications in the Legendre transform (2) with a single batched call of the cuBlasDgemm library. The different matrices in (2) have different sizes because the total wavenumber goes from 0 to NT. To use the fully batched matrix multiplication we pad each matrix with zeroes up to the largest size, since the library currently does not support differing sizes within a batch. This step increases the overall number of floating point operations by almost a factor 10 but still improves the overall performance (Figure 5). We perform the FFT in equation (3) with the cuFFT library,

Code example 1

```
$!ACC parallel loop collapse(3)
loop over 1st dimension
  loop over 2nd dimension
    loop over fields
      ...
      Operation
```

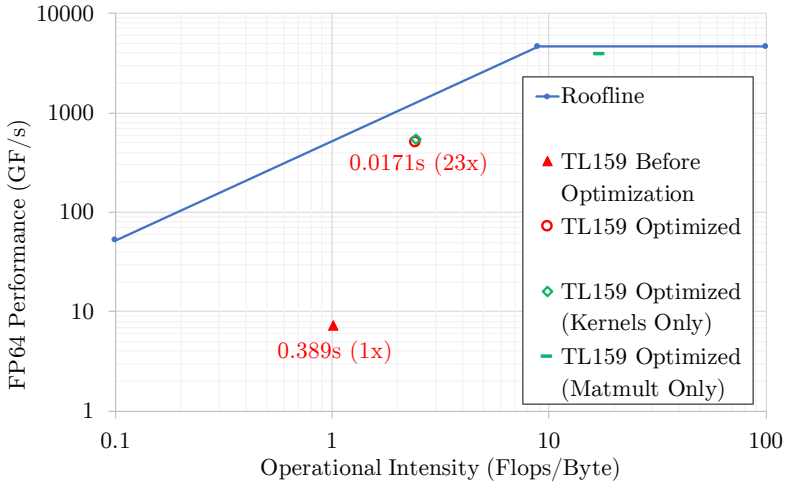


Figure 5. Roofline plot for the spectral transform dwarf at 125km resolution ($N_T = 159$) on the NVIDIA Tesla P100 GPU. The full time-step of the original prototype is represented by the solid red triangle. The corresponding time-step for the optimised prototype is represented by the red circle. Also included are partial results for kernels only (open green diamond) and matrix multiplication only (green dash). Each point is positioned in the plot according to its operational intensity: points under the sloping region of the roofline are limited by available memory bandwidth, and points under the horizontal region are limited by peak computational performance.

where we batch over the vertical dimension but multiple calls are still needed over the spherical dimension (noting FFTs cannot be padded in a similar way to matrix multiplications). Therefore the implementation remains suboptimal here: we are still not fully exposing parallelism and there would be scope for further improvements if a FFT batching interface supporting differing sizes were to become available.

We restructured array layouts to ensure that multiple threads on the GPU can cooperate to load chunks of data from memory in a "coalesced" manner. This would allow a high percentage of available memory throughput. This is achieved when the fastest moving index in the multidimensional array corresponds to the OpenACC loop index occurring at the innermost level in the collapsed loop nest as follows:

Code example 2

```
$!ACC parallel loop collapse(3)
do k=1, ...
  do j=1, ...
    do i=1, ...
      ...
      array(i,j,k)=...
```

In FORTRAN, arrays are structured such that elements accessed by consecutive innermost indices (i in this example) are consecutive in memory. Since i is used as the innermost loop index in this collapsed loop nest, then memory coalescing is achieved.

Sometimes matrix transposes are necessary, but where possible these were pushed into the DGEMM library calls, which have much higher-performing implementations of transposed data accesses. There remain transpose patterns within kernels involved in transposing grid point data from column structure to latitudinal (and inverse) operations, which naturally involve transposes and are thus harder to fix through restructuring. However, we optimised these using the "tile" OpenACC clause, which instructs the compiler to stage the operation through multiple relatively small tiles which can perform the transpose operations within fast on-chip memory spaces, such that the accesses to global memory are much more regular.

Data allocation on the GPU is expensive, as is data movement between the CPU and GPU. We structured the code such that the fields stay resident on the GPU for the whole timestep loop: all allocations/frees have been moved outside the timestep loop with re-use of temporary arrays, and thus all data transfer has been minimized.

The restructured algorithm achieves an overall speedup factor of 23x compared to the initial version which also used cuBlas and cuFFT but followed the CPU version more closely. Matrix multiplication performance is higher than the overall performance (in flops) and the operational intensity is increased into the compute-bound regime. Note that matrix multiplication is associated with $O(N^3)$ computational complexity for $O(N^2)$ memory accesses. The extra padding operations lead to larger N and therefore also to increased operational intensity. More details about the single GPU optimisations can be found in Mazauric et al. (2017b).

Beyond a single GPU, with multiple interconnected GPUs we see a large benefit by using the modern NVLink interconnect and the recently announced NVSwitch due to the high importance of communication for the transpositions described in section 3.2. Each GPU features multiple ports of high-bandwidth NVlink connections, each providing 50 GB/s of bi-directional bandwidth when using the Volta GPUs. For full bandwidth connectivity when using more than 4 GPUs we use the NVSwitch interconnect on the DGX-2 server. The DGX-2 server has 16 Volta V100 GPUs: each with six 50 GB/s NVLink connections into the switch with routing to any of the other GPUs in the system. This allows 300 GB/s communications between any pair of GPUs in the system, or equivalently 2.4 TB/s total throughput.

When running a single application across multiple GPUs, it is necessary to transfer data between the distinct memory spaces. Traditionally, such transfers needed to be realised via host memory and required the participation of the host CPU. Not only did

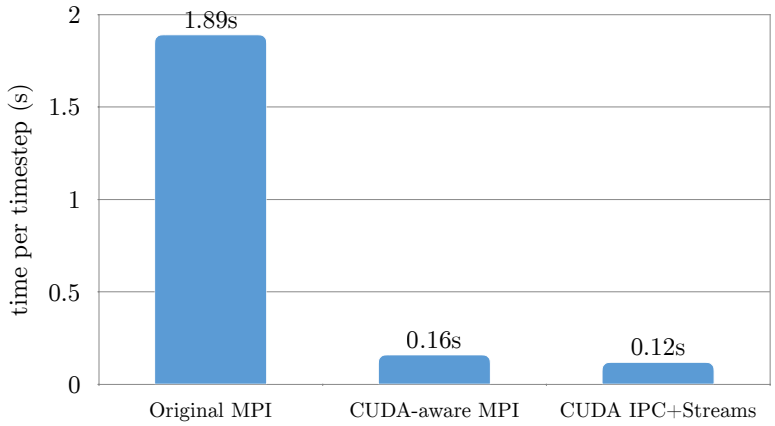


Figure 6. Computational performance of the spectral transform dwarf at 18km resolution ($N_T = 639$) on 4 NVIDIA V100 GPUs of the DGX1 with the original MPI implementation (left), CUDA-aware MPI communication (middle) and NVLink optimised communication (right). This resolution is currently used operationally for the members of the ensemble forecast at ECMWF.

this introduce additional latency, but also limited the overall bandwidth to the bandwidth offered by the PCIe bus connecting CPU and GPUs. However, modern MPI implementations are CUDA-aware. This means that pointers to GPU memory can be passed directly into the MPI calls, avoiding unnecessary transfers (both in the application and in the underlying MPI implementation). This is particularly useful when using a server that features high-bandwidth NVLink connections between GPUs, in which case CUDA-aware MPI will use these links automatically. Moving our dwarf to CUDA-aware MPI gave us a speedup of 12x (Figure 6). However, even with this optimisation the all-to-all operations remained inefficient because communication between different GPUs was not exchanged concurrently. Perfect overlap was achieved by implementing an optimised version of the all-to-all communication phase directly in CUDA using the Inter Process Communication (IPC) API. Using memory handles, rather than pointers, CUDA IPC allows to share memory spaces between multiple processes, thus allowing one GPU to directly access memory on another GPU. This allowed another speedup of about 30% (Figure 6).

In Figure 7 we demonstrate how the use of DGX-2 with NVSwitch allows significantly better scaling than the use of DGX-1 for a spectral transform at 18km resolution ($N_T = 639$). Note that we tune the number of MPI tasks in use: we use the NVIDIA Multi Process Service to allow oversubscription of GPUs such that, e.g. the 8 GPU result on DGX-2 uses 16 MPI tasks across the 8 GPUs (i.e. 2 operating per GPU). This is because such oversubscription can sometimes be beneficial to spread out any load imbalance resulting from the spherical grid decomposition (see below) and hide latencies. We chose the best performing number of MPI tasks per GPU in each case.

As we increase the number of GPUs, the scaling on DGX-1V is limited: This is because we no longer retain full connectivity and some messages must go through the lower-bandwidth PCIe and QPI links and/or Infiniband when scaling across multiple servers. But on DGX-2 with NVSwitch, all 16 GPUs have full connectivity: that is we have maximum peak bandwidth of 300 GB/s between each pair of GPUs in use. The performance is seen to scale well out to the full 16 GPUs on DGX-2, where the

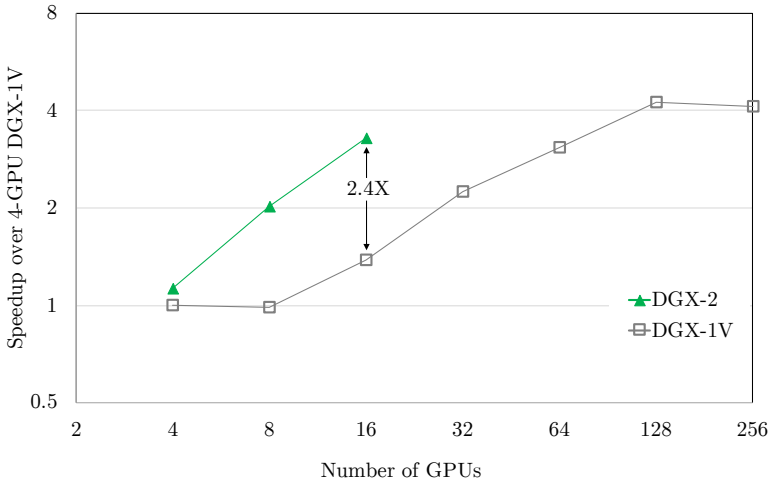


Figure 7. Computational performance of the spectral transform dwarf at 18km resolution ($N_T = 639$) on up to 16 GPUs on one DGX-2 and up to 32 DGX-1 servers connected with Infiniband. The DGX-1V uses MPI for ≥ 8 GPUs (due to lack of AlltoAll links), all others use CUDA IPC. DGX-2 results use pre-production hardware. The points were connected with lines for the purpose of improving readability.

difference with the 16 GPU (2-server) DGX-1V result is 2.4x. It can also be seen that the speedup going from 4 to 16 GPUs on DGX-2 is 3.2x, whereas the ideal speedup would be 4x. However, initial investigations reveal that this deviation from ideal scaling is not primarily due to communication overhead but instead to load imbalance between the MPI tasks from the spherical grid decomposition that is chosen by the application in each case, which would indicate that better scaling would be observed with a more balanced decomposition. More details about the multi-node optimisation of the spectral transform dwarf can be found in Douriez et al. (2018).

First results on the supercomputer Summit of the Oak Ridge National Laboratory are shown in Figure 8. At this large scale we observe a speedup of about 2x when comparing the optimised GPU version with the initial CPU version of the spectral transform dwarf. These simulations were run at 2.5km resolution ($N_T = 3999$) and use 240 fields. In operational application the number of fields should be larger (see Section 3.1). We still need to optimise the memory consumption and the initialisation to test our prototype for more realistic numbers. Also there is still room for improving the setup of the environment on Summit, and exploring alternative programming models. The important message is that the GPU optimised dwarf from ESCAPE was capable of running on a huge number of GPUs on Summit and first results suggest that we achieve a significant speedup. We will do more analysis of these simulations in the future.

3.4 CPU optimisation

The spectral transform dwarf is based on the operational implementation used in IFS that has been continuously optimised over multiple decades. According to profiling results, it clearly appeared that the main computational intensive kernels are the FFT and matrix multiplication executed by a dedicated highly tuned library (as Intel Mathematics Kernel Library, called MKL).

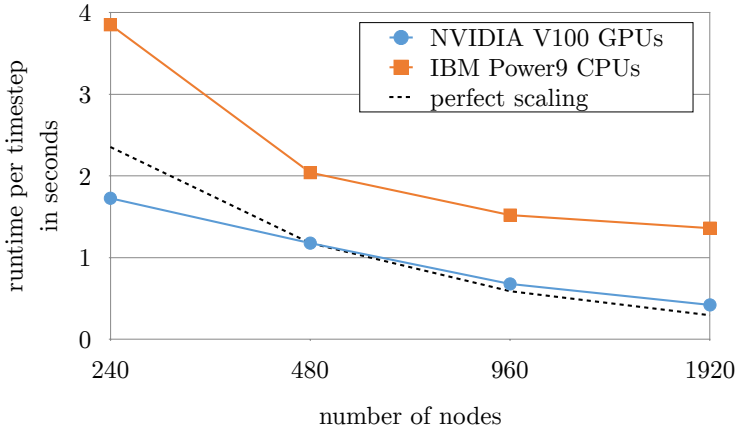


Figure 8. Strong scaling comparison between GPU optimised version and initial CPU version of the spectral transform dwarf on Summit at the Oak Ridge National Laboratory. All of these simulations are using 2.5km resolution ($N_T = 3999$) with 240 fields. The GPU version uses 6 V100 GPUs per node which leads to a maximum number of 11520 GPUs. The perfect scaling was chosen such that it goes through the GPU result at 480 nodes to illustrate the scaling efficiency of the GPU version. The points were connected with lines for the purpose of improving the readability of the plot.

In support of this work we looked into different data scope analysis tools. A comparison of the different tools is available in Mazauric et al. (2017a). The first optimisation strategy concentrated the effort on non-intrusive optimisations which have the advantage of being portable and maintainable. Among these optimisations, the use of extensions to the x86 instruction set architecture (ISA) as SSE, AVX, AVX2, AVX-512 is notable, because it indicates how much of the source code can be vectorised by the compiler. When the compiler failed at vectorising some loops or loop nests, a deeper investigation of how to use compiler directives followed. As the different instruction sets are not supported by all processors, the study proposed an intra node scalability comparison study among several available systems (at the time of benchmarking).

System tuning using Turbo frequency (TUR), Transparent Huge Page (THP), memory allocator (MAP) can be done without modifying the source code. This exposes both performance gains and interesting information on dwarf behaviour. Indeed, on ATOS BullSequana x400 supercomputer with Intel® Xeon Broadwell E5-2690v4 processors, enabling turbo offers a gain equal to 11%, enabling THP gives 22%, MAP 27%, and finally the best performance (35% of performance gain) is achieved by the combination of MAP and TUR. This shows that memory management is a key point. More details about the single-node CPU optimisation can be found in Mazauric et al. (2017b).

Multi-node optimisation for CPUs focused on improving the MPI communication. The largest potential for optimisations was found to be in the preparation phase of point-to-point communications. During the preparation phase the sender side gathers the local data into a contiguous buffer (Pack operation) and hands it off to the MPI library. On the receiver side, data is then scattered from a contiguous user buffer to its correct location (Unpack operation). Pack and Unpack are nearly inevitable with scattered data because Remote Direct Memory Access (RDMA) with no gather-scatter operations are known to

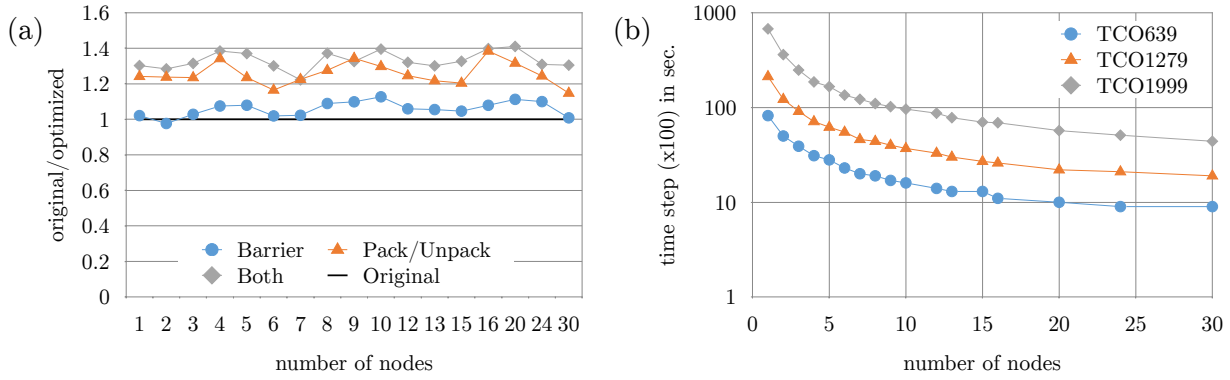


Figure 9. Performance measurements on up to 30 Intel® Xeon Skylake 6148 nodes. Subpanel (a) shows the speed-up at 18km resolution ($N_T = 639$) with regards to different optimisations of the communication preparation phase. Subpanel (b) shows the computational performance for three different resolutions of 18km ($N_T = 639$), 9km ($N_T = 1279$) and 5km ($N_T = 1999$) representative of current and future operational requirements. Note the use of log-scale. The points were connected with lines for the purpose of improving readability. All of these results have been obtained through measurements on the ATOS BullSequana x400 supercomputer of ATOS internal HPC resources.

be often less effective, notably due to the memory pinning latency (Wu et al., 2004). It also means that sender and receiver must exchange their memory layout as they may differ. The performance improvement came from reordering the loops for both pack and unpack following the memory layout of the scattered buffer. This optimisation decreased the number of tests (i.e. copy or not copy) and avoids scanning memory multiple times which was unnecessary. We reduced this with a global performance gain on the whole dwarf of about 20% (Figure 9a). A few percent of improvement came also from disabling MPI barriers which existed in the code to profile the MPI communications at low level. These barriers produced imbalance, the necessary synchronisations created contention and overall created a bias in the communications profile. The computational performance up to 30 ATOS BullSequana x400 nodes equipped with Intel® Xeon Skylake 6148 processors is shown in Figure 9b. This work has been performed on ATOS internal HPC resources. This optimisation can be immediately applied to the operational model IFS due to its non-intrusiveness. More details about the multi-node optimisation on CPUs can be found in Douriez et al. (2018).

The speedup of the GPU optimisations came from using highly optimised GPU libraries for batched DGEMM and FFT and avoiding the transposition of temporary arrays which was not necessary. On the CPU we can apply these improvements in the handling of temporary arrays as well. We used these optimisations in a newly developed serial spectral transform inside Atlas which is used operationally at ECMWF for post-processing purposes since the beginning of 2019. Post-processing is run in serial mode due to the large number of concurrent post-processing jobs. Compared to the previous operational serial transform used for post-processing we find a speedup of about 3x (Figure 10).

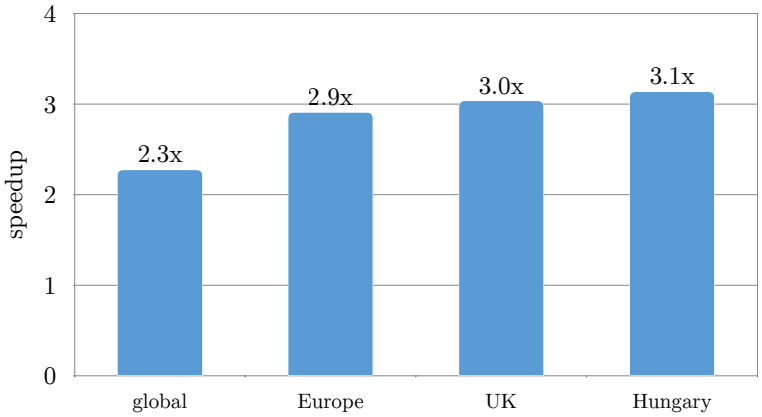


Figure 10. Speedup of the spectral transform by porting optimisations introduced in the GPU version back to the CPUs. The base line for this comparison is the current operational post-processing library used at ECMWF. This version of the spectral transform allows the computation on limited area domains. The speedup is given here for the global transform and three examples of limited domains (Europe, UK and Hungary).

.

3.5 Optical processors

Optalysys have been investigating an optical implementation of the spectral transform dwarf (biFFT for limited area models as well as spherical harmonics for global models). The fundamental idea behind optical processors is to encode information into a laser beam by adjusting the magnitude and phase in each point of the beam. This information becomes the Fourier transform of the initial information in the focal plane of a lens. The information can be encoded into the optical beam by using spatial light modulators (SLMs) as illustrated in Figure 11. The result of the Fourier transform can be recorded by placing a camera in the focal plane of a lens. A photo of an early prototype is shown in Figure 12.

SLMs are optical devices with an array of pixels which can modulate an optical field as it propagates through (or is reflected by) the device. These pixels can modulate the phase (essentially applying a variable optical delay) or polarisation state of the light. Often they modulate a combination of the two. When combined with polarisers, this polarisation modulation can be converted into an amplitude modulation. Hence the modulation capability of a given SLM as a function of 'grey level' can be expressed by a complex vector, which describes an operating curve on the complex plane. Each pixel of the SLM is generally a 1-parameter device; arbitrary complex modulation is not offered by the SLM, only some sub-set. This is one of the key issues with regards to exploiting the optical Fourier transform.

Sensor arrays - essentially common camera sensors - are used to digitise the optical field. They are in general sensitive to the intensity of the light, which is the magnitude of the amplitude squared. This poses a difficulty to sensitively measuring the amplitude. Moreover, they are not sensitive to optical phase. We overcame this with a method that allowed the optical phase to

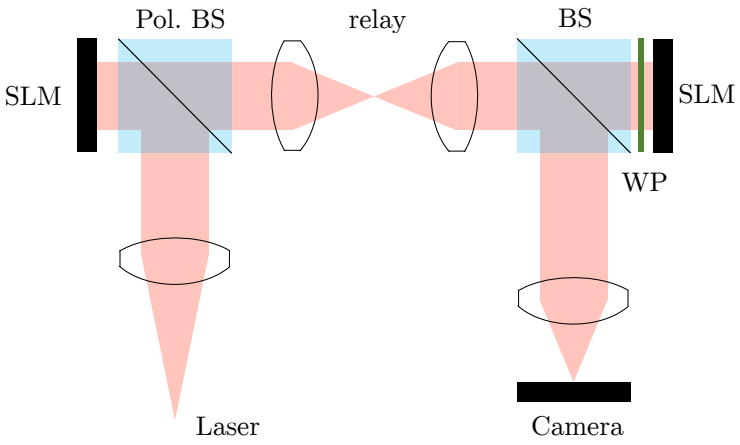


Figure 11. Illustration of the fundamental idea behind the optical processor. The laser beam is emitted on the bottom left. Two spatial light modulators (SLM) are used together to input the complex function. The system uses beamsplitters (BS) and an optical relay to image one reflective SLM onto another, followed by a lens assembly which approximates an ideal thin lens and renders the optical Fourier transform on a camera sensor. The half-waveplate (WP) before the second SLM is used to rotate linearly polarised light onto the axis of SLM action (the direction in which the refractive index switches), thus causing it to act as a phase modulator.

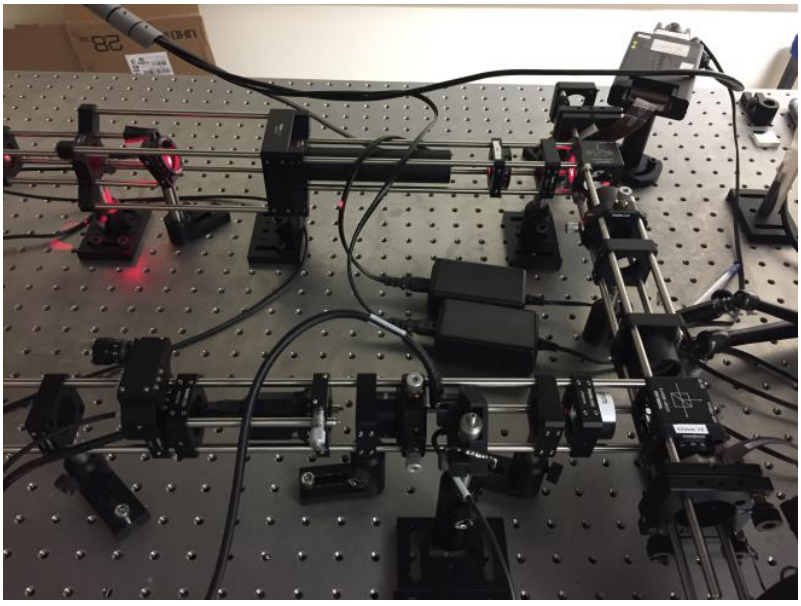


Figure 12. Photo of the first prototype of the optical processor. The final product is built into an enclosure of similar size like a GPU.

be derived from the intensity-only measurements. This was done by introducing known terms into the functions and measuring the resulting change in the output.

Each pixel of the SLM is addressable with an 8-bit value (256 levels). The SLM is not capable of independently modulating the magnitude and phase of the optical field. In the Optalysys processing system, the SLMs are configured to modulate both the amplitude and phase in a coupled manner, such that optimal correlation performance is achieved. The optical Fourier transform and all of the functions are inherently two dimensional. The propagating light beam can be thought of as a 2D function propagating and transforming along a third direction. The system is most naturally applied to 2D datasets, and many problems can be mapped to an appropriate representation.

A critical aspect to realizing the potential of optical processing systems is the interface to a traditional computing platform. Bridging this gap has been a significant undertaking for Optalysys, and has resulted in the development of a custom PCIe drive board. This board interfaces to a host machine over PCIe and has direct memory access (DMA) to the system memory (RAM). It provides an interface to 4 SLMs and 2 cameras. The cameras are 4K (4096x3072). Initially, they operate at 100 Hz, but a future firmware upgrade will unlock 300 Hz operation, and 600 Hz half-frame operation, dramatically increasing the potential data throughput.

There are currently two options for the SLMs. One option is using high speed binary SLMs which operate at 2.4kHz. This offers correlation at binary precision. The second option is greyscale SLMs which operate at 120 Hz. This is currently the only option to reach more than binary precision. The performance of the entire processor is determined by the part with the lowest frequency. The main bottleneck with multiple bit precision is the operating frequency of the greyscale SLM. There is currently no easy solution to increase the frequency of greyscale SLMs.

Optical processing is more appropriately applied to cases where high-throughput relatively-complex operations are the priority, with less of an emphasis on numerical precision. The inherent ability of optical correlators to rapidly process convolutions naturally leads to the formation of convolution neural nets and machine learning technologies. More details about the Optalysys optical processor have been published in Macfaden et al. (2017) and Mazauric et al. (2017b).

3.6 Comparison between processors in terms of runtime and energy consumption

We will loose a lot of the speedup achieved by running the spectral transform on accelerators if the CPUs are idle during the computation on the accelerators. Also we need to take the cost of data transfer between CPU and accelerator into account which has not been included in the speedup numbers in this section. To take full advantage of the NVLink and NVSwitch we would need to run the entire simulation on a single node which requires at the currently operational resolutions more work on optimising the memory footprint of the model.

For the CPUs and GPUs used in this paper the overall cost is dominated by the cost of the hardware and therefore by the number of sockets/devices required to reach the desired runtime. In addition to the number of devices we also compare the energy consumption. The large number of zero operations caused in the optimised GPU version by the padding of the matrices in the Legendre transform makes it impossible to do a fair comparison between CPU and GPU by comparing metrics based on floating point operations including comparing roofline plots.

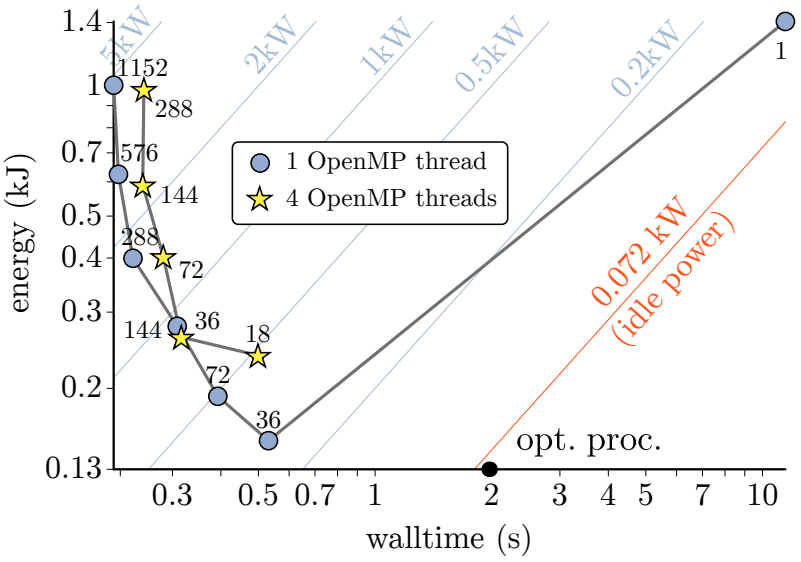


Figure 13. Log-log plot of the energy consumption vs. wall-clock time for the BiFFT dwarf and corresponding to the combination of one direct and one inverse transformation for 525 fields. Each data point is the result of averaging the outcome of two separate runs. Grey lines connect runs with the same number of OpenMP threads (1 resp. 4). Added are lines of constant power (light blue lines), including the power delivered by a node in the idle state (orange line). Indices next to each data point denote the number of MPI tasks. The black dot represents the estimate of the Optalysys optical processor when using a greyscale SLM. The performance of the optical processor at binary precision is much higher (not shown).

In the full operational model the 18km resolution ensemble member ($N_T = 639$) using 30 nodes on the Cray XC40 takes about 1.4s per time-step and the spectral transform component is about 15 percent (0.21s). Measurements with the dwarf on the Cray XC40 at 18km resolution ($N_T = 639$) resulted in 4.35s per timestep on a single node (4 MPI tasks, 18 threads per task), and 1.77s on 2 nodes. The energy consumption was measured at around 0.3Wh on the Cray XC40, which compares to 0.026 Wh measured on 4 V100 GPUs on a DGX1 which take 0.12s per time-step. The energy measurement on the XC40 is based on proprietary power management counters. The measurement on the V100 GPUs uses the nvidia-smi monitoring tool.

Tests in ESCAPE on the latest generation of Intel Skylake CPUs have shown 0.12s per timestep using 13 Intel® Xeon Skylake 6148 nodes (connected via a fat-tree EDR network) as shown in Figure 9. This parallel CPU version has not seen the more radical changes which have been used in redesigning the algorithm for the parallel GPU and serial CPU version. There might still be potential for more substantial optimisations in the parallel CPU version which we will explore in future research.

A comparison between CPUs and optical processor with greyscale SLM is shown in Figure 13. The energy consumption of the optical processor is much lower than for the CPU. The runtime of the optical processor is larger due to the relatively slow performance of the grayscale SLM which is currently necessary to reach the precision necessary for NWP applications. More details about the comparison between CPU and optical processor can be found in Van Bever et al. (2018).

3.7 Sustainability of code optimisation techniques

The optimisation results presented in this section are just an example for one of the dwarfs. Similar improvements have been achieved for the advection scheme MPDATA (Douriez et al., 2018) and the radiation scheme ACRANEB2 (Poulsen and Berg, 2017). These optimisations are hardware specific and will be difficult to maintain on upcoming new architectures. As a strategy towards performance portability and sustainability we have worked in the ESCAPE project on domain specific languages through the GridTools framework. The main goal of the GridTools library is to provide a solution for weather and climate models to run one code base on many different architectures (portability) and achieve good performance (performance portability). However, the main operational product of GridTools so far focused on solutions for lat-lon grid models like COSMO. The work developed in the ESCAPE project aimed at extending the DSL support for irregular grids and the efficient generation of backends for multiple architectures.

These DSL developments have been used to implement a portable version of the MPDATA dwarf. The DSL version hides such details as the nested loops and the OpenACC directives used to specify properties of the GPU kernel and data layouts of the FORTRAN arrays. Furthermore, the DSL allows to compose several of these operators together, which is used by the library to apply advanced performance optimisations like loop fusion or software managed caches. Comparing the FORTRAN OpenACC kernel with the DSL version gives us a speedup of 2.1x for the DSL version. This speedup could also be achieved by hand-tuned optimisation. The DSL prevents the repeated manual effort of tuning the code for multiple architectures. At the same time the DSL allows to perform optimisations which would otherwise make the code unreadable. More details about this work including code examples on how to use the new backend to GridTools can be found in Osuna (2018). More work on sustainability through domain specific languages will follow through the ESCAPE-2 project which started in October 2018.

In addition to GridTools we also worked on using domain specific languages via the CLAW DSL (Clement et al., 2018) for the cloud microphysics dwarf. In particular the use of the Single Column Abstraction (SCA), where physical parameterisations are defined solely in terms of a single horizontal column, enables domain scientists to define physics equations purely in terms of vertical dependencies without needing to account for parallelisation issues. The CLAW DSL then inserts loops over the data-parallel horizontal dimension specific to the hardware architecture and programming model (OpenMP, OpenACC) via source-to-source translation, allowing multiple architectures to be targeted from a single source code. A GPU implementation of the CLOUDSC dwarf generated by automated source translation tools has been used to generate similar performance results to the ones presented by Xiao et al. (2017) on K80 GPUs via the OpenACC backend of the CLAW DSL.

4 Conclusions and outlook

The ESCAPE project has introduced the concept of Weather & Climate dwarfs as fundamental domain-specific building blocks into the weather and climate community. Their categorisation of computational and communication patterns has been extremely useful in further breaking down the complexity of weather prediction and climate models, and advancing their adaptation to future hardware. Prototype implementations of these dwarfs have been used to work on optimising them and using them for the purpose of benchmarking new computers. These included measures for verifying their scientific correctness, documentation

and input from domain scientists. Our dwarfs are very well suited for optimisation and benchmarking, they are small enough to be fairly easy to understand, and at the same time represent a significant part of the performance of the whole weather prediction model.

The paper gives an overview of the dwarfs that were identified in the ESCAPE project, while illustrating with the spectral transform dwarf a detailed example of the optimisation cycle within ESCAPE. In ESCAPE-2 we further identify dwarfs in other Earth System components such as sea-ice and ocean models, that are crucial for the performance of coupled applications.

To avoid code duplication we used the data structure and mesh handling framework provided by Atlas. Atlas has been extended in ESCAPE to support limited area grids and DSL with GridTools. The participating models and most prototype implementations are based on FORTRAN and all our optimisations can be incorporated into FORTRAN code including the use of CUDA functions on GPUs by calling C functions from FORTRAN. We have started to incorporate the obtained code optimisations into operations which gives us a significant speedup in the spectral transform used for postprocessing. Besides optimisations of the existing code, improved algorithms have been developed which are specifically targeted at improving performance on large scale systems, which include a multigrid preconditioner for the elliptic solver to reduce iteration counts in iterative solvers, a HEVI time-integration scheme with significantly improved stability and alternative finite difference methods on the sphere in the context of reducing global communications across large processor counts, and alternative solution procedures for spectral transforms at fixed energy cost, with FFTs and spherical harmonics realised on optical processors.

Code optimisations performed in the ESCAPE project targeted Intel CPUs, Intel Xeon Phi processors, NVIDIA GPUs and Optalysys optical processors. In NWP applications, the bottleneck in terms of performance is usually the memory bandwidth between processor and main memory. Having fast interconnect like NVLink and NVSwitch can provide a massive speedup. Having all of the processing units used by the simulation connected with such a fast interconnect is still a challenge. Using accelerators only for a small part of the code destroys a lot of the benefit in terms of overall cost if the CPUs are idle while the accelerators perform their computations. We either need to move a large part of the code to the accelerator (like in Fuhrer et al., 2018; Schalkwijk et al., 2015) or we need to overlap computations on the CPU with computations on the accelerator (like in Dziekan et al., 2019). Applying this to the participating models still requires more work.

Most of the work done in the ESCAPE project in terms of code optimisation focused on hardware specific optimisations. This makes the optimisations difficult to maintain on upcoming new hardware architectures. As an approach towards performance portability we implemented a DSL prototype for advection with MPDATA by using the GridTools framework, and a prototype of the cloud microphysics dwarf by using the CLAW DSL. DSLs are a very promising tool to enable good performance on multiple architectures while maintaining a single code base. However, designing a domain specific language that is user friendly and at the same time close to hand tuned performance on each architecture is challenging. This work will continue in the ESCAPE-2 project which started in October 2018.

Comparing different methods requires to include all costs. Spectral transform with semi-implicit, semi-Lagrangian time-integration has always been considered as being poorly suited for large supercomputers due to the high volume of communication. Our work indicates that thanks to much larger time-steps the overall communication cost is not necessarily worse

than for other methods. Again overlapping different parts of the model such that useful computation can be done while data is communicated is a way forward and needs to be high priority in future research.

When moving towards very high resolution global simulations of O(1km) or less and considering exascale computations on a variety of emerging HPC architectures, there is a continued interest and need in pursuing fundamentally different algorithmic approaches that simply do not communicate beyond a certain halo size while retaining all other favourable properties, such as removing time-step size restrictions as in the semi-Lagrangian advection with semi-implicit time-stepping (SISL), and such approaches are further investigated in ESCAPE-2.

Code availability. The data structure framework Atlas is available at https://github.com/ecmwf/atlas under an Apache License 2.0. The GridTools framework used as a domain specific language approach for MPDATA is available at https://github.com/GridTools/gridtools under a BSD-3-Clause license. The CLAW DSL used for the cloud microphysics dwarf is available at https://github.com/claw-project/claw-compiler under a BSD-2-Clause license. Model codes developed at ECMWF are the intellectual property of ECMWF and its member states, and therefore the IFS code and the IFS-FVM code are not publicly available. Access to a reduced version of the IFS code may be obtained from ECMWF under an OpenIFS license (see http://www.ecmwf.int/en/research/projects/openifs for further information, last access: 28 May 2019). The ALARO and ALADIN codes, along with all their related intellectual property rights, are owned by the members of the ALADIN consortium and are shared with the members of the HIRLAM consortium in the frame of a cooperation agreement. This agreement allows each member of either consortium to license the shared ALADIN-HIRLAM codes to academic institutions of their home country for noncommercial research. Access to the codes of the ALADIN System can be obtained by contacting one of the member institutes or by submitting a request in the contact link below the page of the ALADIN website (http://www.umr-cnrm.fr/aladin/, last access: 28 May 2019) and the access will be subject to signing a standardised ALADIN-HIRLAM license agreement. The COSMO-EULAG model, along with all of its related intellectual property rights, is owned by the members of the COSMO consortium under the cooperation agreement. This agreement allows each member of the COSMO consortium to license the COSMO-EULAG code (http://www.cosmo-model.org/content/consortium/licencing.htm, last access: 28 May 2019) without fee to academic institutions of their home country for a noncommercial research. The code of the GRASS model is intellectual property of Météo-France and is not publicly available. The code of the ESCAPE dwarfs is intellectual property of ECMWF. A license for educational and non-commercial research can be obtained from ECMWF (see http://www.hpc-escape.eu for contact details, last access: 28 May 2019). For the GPU optimisation of the spectral transform dwarf we used the PGI compiler version 17.10, CUDA 9.0.176 and OpenMPI 2.1.3. For the GPU optimisation of the MPDATA dwarf we used the PGI compiler version 17.10, CUDA 9.2.88 and OpenMPI 2.1.3. CUDA includes the compiler nvcc used to compile the library function wrappers, the libraries themselves and the profiling tool nvprof which we used for profiling on the GPUs. For the CPU optimisation we used Intel compilers and libraries version 2018.1.163. This includes the compilers icc and ifort and the libraries mkl and mpi. The work on the optical processor Optalysys used the MATLAB software version 2017b.

Author contributions. Andreas Müller performed energy measurements for the spectral transform dwarf, implemented the optimised spectral transform for post-processing, worked on the implementation of the MPDATA dwarf, supported maintenance for all dwarf prototypes and assembled the paper. Willem Deconinck is the main developer of the data structure framework Atlas and contributed the work on improving

and extending Atlas. He also was involved in creating, supporting and maintaining all of the dwarf prototypes. Christian Kühnlein and Piotr K. Smolarkiewicz are the main developers of IFS-FVM and contributed to the corresponding elliptic solver and MPDATA dwarfs. Gianmarco Mengaldo was involved in the creation of the dwarfs and provided improvements to early drafts of the manuscript. Michael Lange and Valentin Clement provided DSL work for the cloud microphysics dwarf. Nils Wedi and Peter Bauer supervised the entire ESCAPE project as coordinators. They were also involved in writing and improving the manuscript. Nils Wedi performed the measurements for the cost profiles in Figure 3. Michail Diamantakis implemented the semi-Lagrangian dwarf prototype. Sarah-Jane Lock supported the work on HEVI time integration methods. Mats Hamrud, Sami Saarinen and George Mozdzynski were involved in creating initial versions of the dwarf prototypes. Sami Saarinen also contributed the code for energy measurements on the Cray XC40 supercomputer. Daniel Thiemert provided project management for the ESCAPE project. Michael Glinton and Pierre Bénard implemented the GRASS model. Fabrice Voitus and Charles Colavolpe worked on HEVI time-stepping methods. Philippe Marguinaud and Yongjun Zheng performed simulations of communication costs. Joris Van Bever, Daan Degrauwe, Geert Smet and Piet Termonia provided energy and runtime measurements for ALARO, analysed the components of the RMI ensemble prediction system and implemented limited area support in Atlas. Kristian P. Nielsen, Bent H. Sass, Jacob W. Poulsen and Per Berg provided work on the radiation dwarf. Carlos Osuna and Oliver Fuhrer provided work on GridTools and were involved in adding GPU support to Atlas. Michael Baldauf contributed work on HEVI DG. Mike Gillard and Joanna Szmelter worked on improving the preconditioning for the elliptic solver. Enda O'Brien, Alastair McKinstry, Oisín Robinson, Parijat Shukla and Michael Lysaght provided the LAITRI dwarf and created initial CPU optimisations and OpenACC ports for all of the dwarf prototypes. Michał Kulczewski, Milosz Ciznicki, Wojciech Piątek, Sebastian Ciesielski, Marek Błażewicz, Krzysztof Kurowski, Marcin Procyk, Pawel Spychala and Bartosz Bosak worked on performance modelling using the roofline model which allowed a better understanding of the experimental measurements. Zbigniew Piotrowski and Andrzej Wyszogrodzki compared different time-integration methods, supported optimisation of IFS-FVM and provided energy measurements of COSMO-EULAG. Erwan Raffin, Cyril Mazauric, David Guibert, Louis Douriez and Xavier Vigouroux provided the CPU optimisations for the spectral transform and MPDATA dwarf prototypes. Alan Gray and Peter Messmer contributed the GPU optimisations for the spectral transform and MPDATA dwarf prototypes. Alexander J. Macfaden and Nick New contributed the work on using optical processors for the spectral transform.

Acknowledgements. The ESCAPE project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671627, see also http://www.hpc-escape.eu/. This work has been also supported by HPC resources provided by Poznan Supercomputing and Networking Center and corresponding research activities under the MAESTRO grant number DEC-2013/08/A/ST6/00296 in National Science Centre (NCN). An award of computer time was provided by the INCITE program. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE office of Science User Facility supported under contract DE-AC05-00OR22725.

References

Asanović, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., and Yelick, K. A.: The landscape of parallel computing research: a view from Berkeley, Tech. Rep. UCB/EECS-2006-183, EECS Department, University of California, Berkeley, http://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html, 2006.

Asanović, K., Wawrzynek, J., Wessel, D., Yelick, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., and Sen, K.: A view of the parallel computing landscape, Communications of the ACM, 52, 56, https://doi.org/10.1145/1562764.1562783, 2009.

Bauer, P., Thorpe, A., and Brunet, G.: The quiet revolution of numerical weather prediction, Nature, 525, 47–55, https://doi.org/10.1038/nature14956, 2015.

Bénard, P. and Glinton, M.: Circumventing the pole problem of reduced lat-lon grids with local schemes. Part I: analysis and model formulation, Q.J.R. Meteorol. Soc., https://doi.org/10.1002/qj.3509, 2019.

Clement, V., Ferrachat, S., Fuhrer, O., Lapillonne, X., Osuna, C. E., Pincus, R., Rood, J., and Sawyer, W.: The CLAW DSL, in: Proceedings of the Platform for Advanced Scientific Computing Conference on - PASC '18, ACM Press, https://doi.org/10.1145/3218176.3218226, 2018.

Colavolpe, C., Voitus, F., and Bénard, P.: RK-IMEX HEVI schemes for fully compressible atmospheric models with advection: analyses and numerical testing, Q.J.R. Meteorol. Soc., 143, 1336–1350, https://doi.org/10.1002/qj.3008, 2017.

Colella, P.: Defining software requirements for scientific computing, DARPA HPCS Presentation, 2004.

Deconinck, W.: Development of the flexible data structure framework Atlas, Tech. rep., ESCAPE, https://goo.gl/11RDb6, 2017a.

Deconinck, W.: Public release of Atlas library version under an open source license which is accelerator-enabled and has improved interoperability features, Tech. rep., ESCAPE, https://goo.gl/dXaPoL, 2017b.

Deconinck, W., Bauer, P., Diamantakis, M., Hamrud, M., Kühnlein, C., Maciel, P., Mengaldo, G., Quintino, T., Raoult, B., Smolarkiewicz, P. K., and Wedi, N. P.: Atlas - A library for numerical weather prediction and climate modelling, Comp. Phys. Communications, 220, 188–204, https://doi.org/10.1016/j.cpc.2017.07.006, 2017.

Douriez, L., Gray, A., Guibert, D., Messmer, P., and Raffin, E.: Performance report and optimized implementations of Weather & Climate dwarfs on multi-node systems, Tech. rep., ESCAPE, http://tiny.cc/fzx36y, 2018.

Dziekan, P., Waruszewski, M., and Pawlowska, H.: University of Warsaw Lagrangian Cloud Model (UWLCM) 1.0: a modern Large-Eddy Simulation tool for warm cloud modeling with Lagrangian microphysics, Geoscientific Model Development Discussions, pp. 1–26, https://doi.org/10.5194/gmd-2018-281, 2019.

Feng, W.-C., Lin, H., Scogland, T., and Zhang, J.: OpenCL and the 13 dwarfs, in: Proceedings of the third joint WOSP/SIPEW international conference on Performance Engineering - ICPE '12, ACM Press, https://doi.org/10.1145/2188286.2188341, 2012.

Flamm, K.: Measuring Moore's Law: Evidence from Price, Cost, and Quality Indexes, Tech. rep., https://doi.org/10.3386/w24553, 2018.

Fuhrer, O., Chadha, T., Hoefler, T., Kwasniewski, G., Lapillonne, X., Leutwyler, D., Lüthi, D., Osuna, C., Schär, C., Schulthess, T. C., and Vogt, H.: Near-global climate simulation at 1km resolution: establishing a performance baseline on 4888 GPUs with COSMO 5.0, Geosci. Model Dev., 11, 1665–1681, https://doi.org/10.5194/gmd-11-1665-2018, 2018.

Glinton, M. R. and Bénard, P.: Circumventing the pole problem of reduced lat-lon grids with local schemes. Part II: validation experiments, Q.J.R. Meteorol. Soc., https://doi.org/10.1002/qj.3495, 2019.

Johnston, B. and Milthorpe, J.: Dwarfs on Accelerators, in: Proceedings of the 47th International Conference on Parallel Processing Companion - ICPP '18, ACM Press, https://doi.org/10.1145/3229710.3229729, 2018.

Kaltofen, E. L.: The seven dwarfs of symbolic computation, in: Texts & Monographs in Symbolic Computation, pp. 95–104, Springer Vienna, https://doi.org/10.1007/978-3-7091-0794-2_5, 2011.

Katzav, J. and Parker, W. S.: The future of climate modeling, Climatic Change, 132, 475–487, https://doi.org/10.1007/s10584-015-1435-x, 2015.

Krommydas, K., chun Feng, W., Antonopoulos, C. D., and Bellas, N.: OpenDwarfs: Characterization of Dwarf-Based Benchmarks on Fixed and Reconfigurable Architectures, Journal of Signal Processing Systems, 85, 373–392, https://doi.org/10.1007/s11265-015-1051-z, 2015.

Kühnlein, C., Deconinck, W., Klein, R., Malardel, S., Piotrowski, Z. P., Smolarkiewicz, P. K., Szmelter, J., and Wedi, N. P.: FVM 1.0: a nonhydrostatic finite-volume dynamical core for the IFS, Geoscientific Model Development, 12, 651–676, https://doi.org/10.5194/gmd-12-651-2019, 2019.

Lawrence, B. N., Rezny, M., Budich, R., Bauer, P., Behrens, J., Carter, M., Deconinck, W., Ford, R., Maynard, C., Mullerworth, S., Osuna, C., Porter, A., Serradell, K., Valcke, S., Wedi, N., and Wilson, S.: Crossing the chasm: how to develop weather and climate models for next generation computers?, Geosci. Model Dev., 11, 1799–1821, https://doi.org/10.5194/gmd-11-1799-2018, 2018.

Macfaden, A. J., Gordon, G. S. D., and Wilkinson, T. D.: An optical Fourier transform coprocessor with direct phase determination, Scientific Reports, 7, https://doi.org/10.1038/s41598-017-13733-1, 2017.

Mazauric, C., Raffin, E., and Guibert, D.: Report on recommendations and specifications for data scope analysis tools, Tech. rep., ESCAPE, https://goo.gl/61DZxq, 2017a.

Mazauric, C., Raffin, E., Vigouroux, X., Guibert, D., Macfaden, A., Poulsen, J., Berg, P., Gray, A., and Messmer, P.: Performance report and optimized implementation of weather & climate dwarfs on GPU, MIC and Optalysys optical processor, Tech. rep., ESCAPE, https://goo.gl/J8zd1i, 2017b.

Mengaldo, G.: Batch 1: definition of several weather & climate dwarfs based on established algorithms and motifs relevant to NWP, provision of prototype implementations and dissemination to other WPs, Tech. rep., ESCAPE, http://goo.gl/s65ojl, 2016.

Mengaldo, G., Wyszogrodski, A., Diamantakis, M., Lock, S.-J., Giraldo, F., and Wedi, N. P.: Current and emerging time-integration strategies in global numerical weather and climate prediction, Arch. Computat. Methods Eng., pp. 1–22, https://doi.org/10.1007/s11831-018-9261-8, 2018.

Messer, O. E. B., D'Azevedo, E., Hill, J., Joubert, W., Berrill, M., and Zimmer, C.: MiniApps derived from production HPC applications using multiple programing models, Int. J. High Performance Computing Applications, 32, 582–593, https://doi.org/10.1177/1094342016668241, 2016.

Michalakes, J., Govett, M., Benson, R., Black, T., Juang, H., Reinecke, A., and Skamarock, B.: AVEC Report: NGGPS Level-1 Benchmarks and software evaluation, Tech. Rep. TN-484, NCAR, Boulder, US, https://www.weather.gov/media/sti/nggps/AVECLevel1BenchmarkingReport0820150602.pdf, 2015.

Mozdzynski, G., Hamrud, M., and Wedi, N. P.: A Partitioned Global Address Space implementation of the European Centre for Medium Range Weather Forecasts Integrated Forecasting System, Int. J. High Perform. Comput. Appl., 29, 261–273, https://doi.org/10.1177/1094342015576773, 2015.

Müller, A., Gillard, M., Nielsen, K. P., and Piotrowski, Z.: Batch 2: definition of novel weather & climate dwarfs, provision of prototype implementations and dissemination to other WPs, Tech. rep., ESCAPE, https://goo.gl/zoCtck, 2017.

Müller, A., Kopera, M. A., Marras, S., Wilcox, L. C., Isaac, T., and Giraldo, F. X.: Strong scaling for numerical weather prediction at petascale with the atmospheric model NUMA, Int. J. High Perform. Comput. Appl., https://doi.org/10.1177/1094342018763966, 2018.

Neumann, P., Düben, P., Adamidis, P., Bauer, P., Brück, M., Kornblueh, L., Klocke, D., Stevens, B., Wedi, N., and Biercamp, J.: Assessing the scales in numerical weather and climate predictions: will exascale be the rescue?, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 377, 20180 148, https://doi.org/10.1098/rsta.2018.0148, 2019.

Osuna, C.: Report on the performance portability demonstrated for the relevant weather & climate dwarfs, Tech. rep., ESCAPE, https://goo.gl/tbEiAu, 2018.

Palmer, T.: Climate forecasting: Build high-resolution global climate models, Nature, 515, 338–339, https://doi.org/10.1038/515338a, 2014.

Phillips, S. C., Engen, V., and Papay, J.: Snow White Clouds and the Seven Dwarfs, in: 2011 IEEE Third International Conference on Cloud Computing Technology and Science, IEEE, https://doi.org/10.1109/cloudcom.2011.114, 2011.

Poulsen, J. W. and Berg, P.: Tuning the implementation of the radiation scheme ACRANEB2, Tech. rep., DMI report 17-22, http://www.dmi.dk/fileadmin/user_upload/Rapporter/TR/2017/SR17-22.pdf, 2017.

Robinson, O., McKinstry, A., and Lysaght, M.: Optimization of IFS subroutine LAITRI on Intel Knights Landing, Tech. rep., Prace White Papers - Evaluations on Intel MIC, 2016.

Schalkwijk, J., Jonker, H. J. J., Siebesma, A. P., and Meijgaard, E. V.: Weather Forecasting Using GPU-Based Large-Eddy Simulations, Bulletin of the American Meteorological Society, 96, 715–723, https://doi.org/10.1175/bams-d-14-00114.1, 2015.

Schulthess, T. C., Bauer, P., Wedi, N., Fuhrer, O., Hoefler, T., and Schär, C.: Reflecting on the Goal and Baseline for Exascale Computing: A Roadmap Based on Weather and Climate Simulations, Computing in Science & Engineering, 21, 30–41, https://doi.org/10.1109/mcse.2018.2888788, 2019.

Shukla, J., Palmer, T. N., Hagedorn, R., Hoskins, B., Kinter, J., Marotzke, J., Miller, M., and Slingo, J.: Toward a New Generation of World Climate Research and Computing Facilities, Bulletin of the American Meteorological Society, 91, 1407–1412, https://doi.org/10.1175/2010bams2900.1, 2010.

Van Bever, J., McFaden, A., Piotrowski, Z., and Degrauwe, D.: Report on energy-efficiency evaluation of several NWP model configurations, Tech. rep., ESCAPE, https://goo.gl/E8mHha, 2018.

Wallemacq, P., UNISDR, and CRED: Economic Losses, Poverty and Disasters 1998-2017, https://doi.org/10.13140/rg.2.2.35610.08643, 2018.

Wedi, N. P., Hamrud, M., and Mozdzynski, G.: A Fast Spherical Harmonics Transform for Global NWP and Climate Models, Mon. Weather Rev., 141, 3450–3461, https://doi.org/10.1175/mwr-d-13-00016.1, 2013.

Wedi, N. P., Bauer, P., Deconinck, W., Diamantakis, M., Hamrud, M., Kühnlein, C., Malardel, S., Mogensen, K., Mozdzynski, G., and Smolarkiewicz, P.: The modelling infrastructure of the Integrated Forecasting System: recent advances and future challenges, Tech. Rep. 760, Eur. Cent. For Medium-Range Weather Forecasts, Reading, UK, 2015.

Wehner, M. F., Oliker, L., Shalf, J., Donofrio, D., Drummond, L. A., Heikes, R., Kamil, S., Kono, C., Miller, N., Miura, H., Mohiyuddin, M., Randall, D., and Yang, W.-S.: Hardware/software co-design of global cloud system resolving models, Journal of Advances in Modeling Earth Systems, 3, https://doi.org/10.1029/2011ms000073, 2011.

World Economic Forum: The 2019 Global Risks Report, https://www.weforum.org/reports/the-global-risks-report-2019, 2019.

Wu, J., Wyckoff, P., and Panda, D.: High performance implementation of MPI derived datatype communication over InfiniBand, in: 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings., IEEE, https://doi.org/10.1109/ipdps.2004.1302917, 2004.

Xiao, H., Diamantakis, M., and Saarinen, S.: An OpenACC GPU adaptation of the IFS cloud microphysics scheme, ECMWF Tech. Memo. No. 805, 2017.

Zheng, Y. and Marguinaud, P.: Simulation of the Performance and Scalability of MPI Communications of Atmospheric Models running on Exascale Supercomputers, Geosci. Model Dev., pp. 1–34, https://doi.org/10.5194/gmd-2017-301, 2018.