ESCAPE: Accelerating extreme-scale Numerical Weather Prediction

Willem Deconinck[1], Andreas Müller, Gianmarco Mengaldo, Michail Diamantakis, Nils Wedi, Peter Bauer

European Centre for Medium Weather Forecasts, Reading, UK

Carlos Osuna, Oliver Fuhrer

MeteoSwiss, Zürich, Switzerland

Introduction

- Numerical Weather Prediction (NWP) and Climate models contain decades of algorithmic developments for conventional CPU hardware architectures
- Paradigm shift towards more parallel and energy efficient many-core hardware architectures due to breakdown of Dennard scaling
- Large impact on programming models expected in the near future
- Rethink of design choices for future software frameworks:
  - Scalability
  - Energy efficiency
  - Flexibility in algorithmic choices
  - Maintainability

ESCAPE (EU Horizon 2020)

Energy-Efficient **Sc**alable **A**lgorithms for Weather **P**rediction at **E**xascale

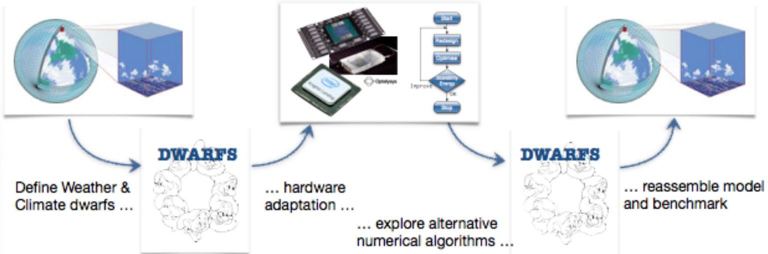
- Combine scientific and computer-science expertise
- Define and co-design necessary steps towards affordable exascale HPC simulations of weather and climate

Figure 1: ESCAPE Partners and expertise throughout Europe:
- *NWP centres*
- *HPC centres*
- *Universities*
- *Hardware vendors*

Weather and Climate Dwarfs

Weather and Climate Dwarfs are self-contained algorithms representing key functional blocks of a NWP & Climate model.
They must be verifiable and possible to integrate in back in the model.

Figure 2: The aim of ESCAPE is to (1) define and create a number of Weather and Climate Dwarfs, (2) optimise them, (3) adapt them to novel hardware technologies, and (4) measure and benchmark them both for performance as energy efficiency.



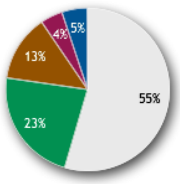


Figure 3: Coverage of ESCAPE Weather and Climate Dwarfs in ECMWF's operational Integrated Forecasting System (IFS). ESCAPE Dwarfs not covered by IFS: MPDATA advection; GCR(k) elliptic solver; BiFFT spectral transform.

- Dynamics — SH spectral transform
- Physics: radiation
- Non-ESCAPE dwarf
- Dynamics — semi-Lagrangian advection
- Physics: cloud microphysics

Atlas, a library for NWP and Climate models

ESCAPE dwarfs rely on *Atlas*, an object-oriented library for flexible parallel data structures for structured grids and unstructured meshes for both global and limited area models.

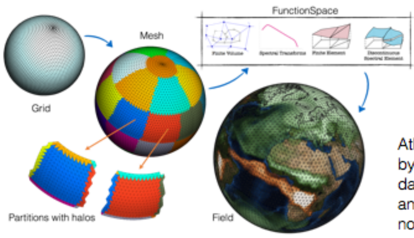
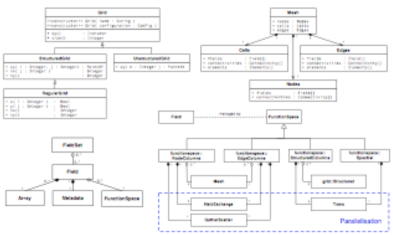


Figure 4: Schematic representation of Atlas capabilities: A global distributed mesh with halos is generated from a global grid. FunctionSpaces describe a Field's discretisation, and are responsible for its parallelisation.

Figure 5: Atlas components showing support for both structured and unstructured grids, and hybrid unstructured meshes. Meshes hold connectivities between cells, edges and nodes.



Atlas allows to reduce model development time by providing parallelisation and management of data structures for hybrid unstructured meshes, and mathematical operators. ESCAPE explores novel numerical methods using Atlas, facilitating reassembly in existing models.

```
Grid grid("O1280");          // Create O1280 octahedral Gaussian grid
FunctionSpace gp = StructuredColumns( grid, Levels(137) );
FunctionSpace sp = Spectral( 1279 );  // 1279 spectral wavenumbers
Field gpfield = gp.createField<double>();       // gridpoint field
Field spfield = sp.createField<double>();       //  spectral field
Trans trans( grid, sp );               // spectral transform operator
trans.invtrans(spfield,gpfield);       // Inverse spectral transform
```

Figure 6: Atlas code example (C++) for computing spherical harmonics spectral transforms

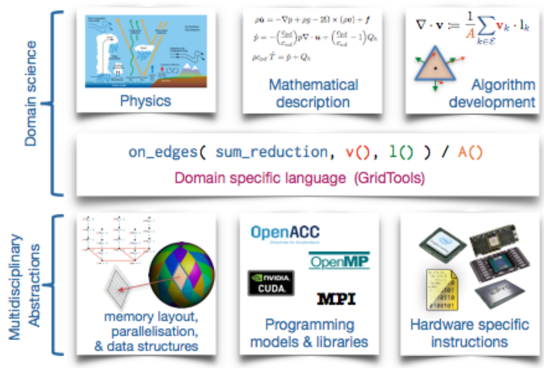
```
grid = atlas_Grid("O1280") ! Create O1280 octahedral Gaussian grid
meshgenerator = atlas_MeshGenerator("structured")
mesh = meshgenerator%generate(grid)   ! Generate mesh from grid
method = atlas_fvm_Method(mesh)       ! Setup finite volume method
nabla  = atlas_Nabla(method)          ! Create FVM nabla operator
call nabla%gradient(scalarfield, gradientfield) ! Compute gradient
```

Figure 7: Atlas code example (Fortran) for computing gradients using a finite volume method

Where are we heading?

- Variety of hardware → variety of algorithm implementations
- Single source code for maintainability is crucial
- Separation of concerns
  - Readable science code !
  - Abstract hardware specific details
  - Abstract parallelisation, memory, data structure details
  - Abstract computational loops and programming models
- Domain specific languages provide a way forward: **GridTools**

Figure 8: Separation of concerns through the GridTools domain specific language. One cannot be an expert in everything.



MeteoSwiss · RMI · NVIDIA · METEO FRANCE · Deutscher Wetterdienst · ICHEC · Optalysys · Bull · PSNC · Loughborough University · DMI

Funded by the European Union

www.hpc-escape.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671627.

Co-ordinated by ECMWF

[1] Email: willem.deconinck@ecmwf.int