COSMO-EULAG on accelerators (CEL-ACCEL)

Zbigniew Piotrowski, Marcin Polkowski, Adam Ryczkowski

Institute of Meteorology and Water Management - National Research Institute

September 11th, 2017

New priority project CEL-ACCEL: foundation and tasks

- New priority project: why not extending CELO ?
- Main goals of CEL-ACCEL.

PROPOZE project - funding and context

The **Numerical Weather Prediction for Sustainable Europe** project is financed by the European fund within the **Smart Growth Operational Programme 2014–2020**, Measure 4.4. "Boosting human potential in R&D sector", through the **First Team** 1/2016 award of **Foundation for Polish Science** with budget close to 500 000 EUR for three years. The project aims at accelerating the operationalization of the EULAG dynamical core of COSMO model, complementary to the CELO priority project of COSMO Consortium.

Aim of this project is not to replace but to support and extend **CELO** efforts.

Why bother when there is an ICON ?

1. EULAG employs unique numerical strategy, connecting second-order diffusive numerics with ILES capability (implicit realization of the subgridscale diffusion via diffusive error of advection) and implicit Krylov solver. It is therefore worth to investigate its capabilities for the NWP.
2. The latter is being done in concert with the efforts at ECMWF, where independent but fully corresponding approach, the Finite Volume Module for IFS, is being advanced.
3. The project will also directly empower the standalone EULAG model for multiscale research

Work package 1 - Accelerator port of COSMO-EULAG

- Reformulation of EULAG diffusion package in Fortran to facilitate adaptation to accelerators
- GridTools port of COSMO-EULAG, including serialization against Fortran results with Serialbox
- Directives-based port of COSMO-EULAG

Work package 2 - Testing COSMO-EULAG on accelerators

- Testing of COSMO-EULAG ensembles on accelerator-based supercomputer
- Evaluation of the forecasting scores with particular attention to the boundary layer.

CEL-ACCEL project

Task	Contributing scientist(s)	FTE- years	Start	Deliverables	Date of delivery	Preceding tasks
L	Zbigniew Piotrowski (IMGW)	0.125 FTE	03.2017	Reports	09.2019	none
1	Adam Ryczkowski (IMGW Zbigniew Piotrowski (IMGW) Marcin Polkowski (IMGW) N.N (IMGW) Hannes Vogt, Carlos Osuna (MeteoSwiss)	1.8 FTE	03.2017	Rewritten diffusion operator for COSMO-EULAG Gridtools port of all COSMO-EULAG stencil components COSMO-EULAG dynamical core in C++/Gridtools Directive-based port of COSMO-EULAG	09.2017 03.2018 09.2018 09.2019	none
2	Zbigniew Piotrowski (IMGW) N.N N.N	0.8 FTE	10.2018	Report of stability, performance and forecasting scores of COSMO-EULAG	09.2019	1
		0.7 /CY 2017 0.9 /CY 2018 1.3 /CY 2019 Total: 2.9 FTE				

Organizing COSMO-EULAG C++ around dwarfs

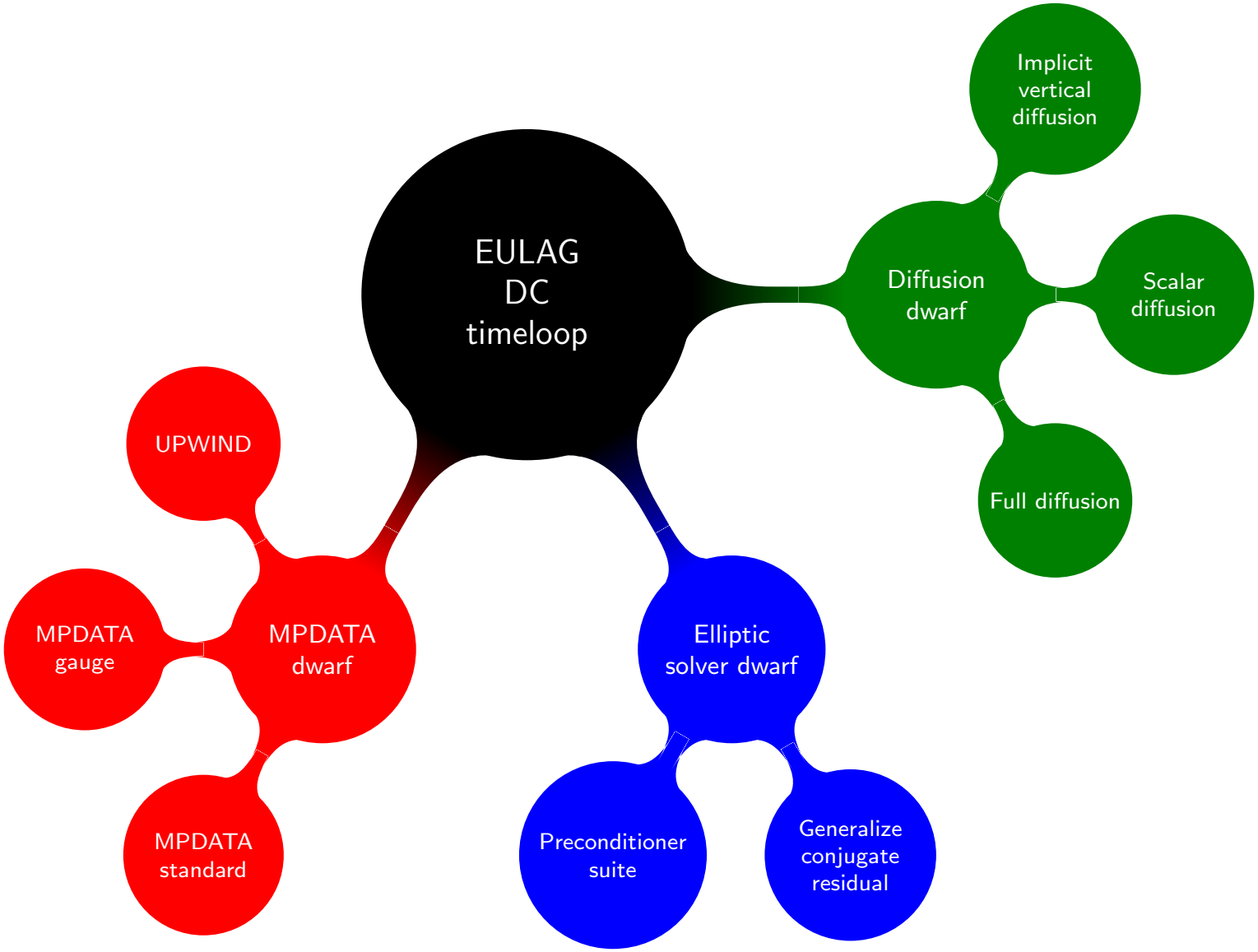
What are dwarfs ?

Dwarfs are mini-applications capable of performing a specific task. In principle dwarf should be able to live independently, e.g. without the heavy full codebase of COSMO. As proposed by the ESCAPE H2020 project, dwarf should take part in and be subject of co-design involving algorithm development and computationally effective and energy efficient implementations on existing architectures. Finally, the dwarf should be easily reimplemented into the NWP framework.

Motivation

To be able to develop EULAG DC ports without the necessary burden of COSMO framework and licensing issues for the third party partners, yet maintaining the close link to mainline EULAG DC code.

Envisaged COSMO-EULAG structure



Propozed organization of the official COSMO-EULAG dynamical core release

- Implementation of COSMO-EULAG into the official COSMO trunk is approaching, main scientific challenges of CELO seem to be solved at this stage.
- Strategic decisions are to be made with respect to the paradigms of official COSMO-EULAG release
- Next, slides on the propozed organization of the COSMO-EULAG follow towards scientific maintainability while facilitating parallel development of the C++ EULAG dycore.

Tenets of EULAG dwarfs

- Independent mini-application, able to perform experiments that were published previously or at least verifiable against analytical solution.

Tenets of EULAG dwarfs

- Independent mini-application, able to perform experiments that were published previously or at least verifiable against analytical solution.
- Minimal set of auxiliary code to limit the complexity.

Tenets of EULAG dwarfs

- Independent mini-application, able to perform experiments that were published previously or at least verifiable against analytical solution.
- Minimal set of auxiliary code to limit the complexity.
- Capable of parallel runs with 3D MPI decomposition, aiming at (future) MPI+MPI/MPI+OpenACC rather than MPI+OpenMP.

Tenets of EULAG dwarfs

- Independent mini-application, able to perform experiments that were published previously or at least verifiable against analytical solution.
- Minimal set of auxiliary code to limit the complexity.
- Capable of parallel runs with 3D MPI decomposition, aiming at (future) MPI+MPI/MPI+OpenACC rather than MPI+OpenMP.
- Written in modern Fortran, capable of both fully static or fully dynamic memory allocation.

Tenets of EULAG dwarfs

- Independent mini-application, able to perform experiments that were published previously or at least verifiable against analytical solution.
- Minimal set of auxiliary code to limit the complexity.
- Capable of parallel runs with 3D MPI decomposition, aiming at (future) MPI+MPI/MPI+OpenACC rather than MPI+OpenMP.
- Written in modern Fortran, capable of both fully static or fully dynamic memory allocation.
- Using centralized data storage, i.e. no local memory allocation of arrays.

Tenets of EULAG dwarfs

- Independent mini-application, able to perform experiments that were published previously or at least verifiable against analytical solution.
- Minimal set of auxiliary code to limit the complexity.
- Capable of parallel runs with 3D MPI decomposition, aiming at (future) MPI+MPI/MPI+OpenACC rather than MPI+OpenMP.
- Written in modern Fortran, capable of both fully static or fully dynamic memory allocation.
- Using centralized data storage, i.e. no local memory allocation of arrays.
- Easy to glue together within NWP or research solver, despite being independent of each other.

Tenets of EULAG dwarfs

- Independent mini-application, able to perform experiments that were published previously or at least verifiable against analytical solution.
- Minimal set of auxiliary code to limit the complexity.
- Capable of parallel runs with 3D MPI decomposition, aiming at (future) MPI+MPI/MPI+OpenACC rather than MPI+OpenMP.
- Written in modern Fortran, capable of both fully static or fully dynamic memory allocation.
- Using centralized data storage, i.e. no local memory allocation of arrays.
- Easy to glue together within NWP or research solver, despite being independent of each other.
- Exposing stencil computations such the stencil is the same everywhere in the computational domain, wherever possible.

Tenets of EULAG dwarfs

- Independent mini-application, able to perform experiments that were published previously or at least verifiable against analytical solution.
- Minimal set of auxiliary code to limit the complexity.
- Capable of parallel runs with 3D MPI decomposition, aiming at (future) MPI+MPI/MPI+OpenACC rather than MPI+OpenMP.
- Written in modern Fortran, capable of both fully static or fully dynamic memory allocation.
- Using centralized data storage, i.e. no local memory allocation of arrays.
- Easy to glue together within NWP or research solver, despite being independent of each other.
- Exposing stencil computations such the stencil is the same everywhere in the computational domain, wherever possible.
- Using halos to compute appropriate boundary conditions (e.g. no flux divergence, zero flux, prescribed flux) by specialized subroutines.

Status of the COSMO-EULAG dynamics' dwarf implementations

Dwarf	MPDATA	Elliptic solver	Diffusion
Dwarf form	Complete	Partial (GCR, no preconditioners)	3D scalar diffusion
Memory optimization	Complete	Advanced	Complete
Forks on accelerators	C++ CUDA	C++ CUDA	None
GRIDTOOLS C++ port	Proof of concept	None	In progress

- *Complete* - development in stable and mature version, verifiable against or analytic results, polishing and bugfixes only.
- *Advanced* - development substantially mature, although with some prospects for important improvement.
- *Forks* - independent ports with no immediate connection to dwarfs, although with the added value of the algorithm analysis

Gridtools port of the MPDATA gague and UPWIND schemes has just been finished (A. Ryczkowski, PROPOZE project efforts). The development revealed the necessity of extension (or evaluation of the existing solutions) to GridTools to accommodate new types of boundary conditions.

MPDATA and diffusion dwarf: facts

- Since regional NWP targets at limited number of cores, we focus on single core performance.

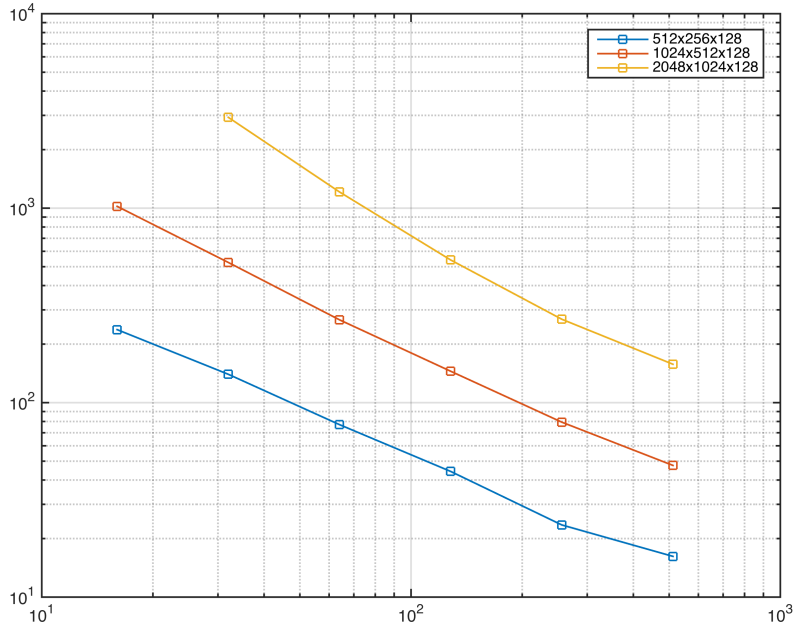
Advection and diffusion dwarfs: per-core performance improvement

Algorithm	1st order UPWIND	MPDATA standard	MPDATA gauge	3D diffusion
32x32x96 grid	1.58	1.40	2.33	2.82
16x16x128 grid	1.36	1.27	2.01	2.12

- Two times speedup is roughly consistent with earlier experiences with structured and unstructured elliptic solver and also with earlier code refactoring and C++ implementation performed on the operational Runge-Kutta dynamical core of COSMO.
- Size of the code is one-fourth of the legacy version.
- MPDATA and other EULAG dwarfs enjoy compact stencils and demand only single halo line in each direction.

MPDATA and diffusion dwarf: facts

Scalability testing of EULAG dwarfs is facilitated by its dynamical(complementary to static) memory allocation capability driven by the command line parameters. This makes it feasible for automation of the scalability testing via python driver generating submission scripts. Sample testing of the diffusion dwarf (M. Polkowski, PROPOZE project efforts) is given below. Horizontal: number of cores, vertical: time to solution. Independence of the solution on the MPI processor distribution is tested with the help of Parallel Netcdf output.



Dwarf reassembly and outlook.

- MPDATA and diffusion dwarfs were reassembled into COSMO-EULAG; direct implementation and library implementation were attempted.
- Going beyond standard Fortran formulation is needed, as the potential of code optimization while maintaining code readability (and portability !) is limited. Further performance improvement of EULAG dwarfs is expected with the use GridTools DSL. However, as the employment of DSL is a complex software engineering task, a pre-refactored Fortran (or C++) code seems to be a key.

Technical foundations of the development of COSMO-EULAG dwarf-based structure

The following software engineering tasks were undertaken:

- Launching and configuring the needed internal infrastructure services
- Creating all the git repositories for our code
- Designing the build process
- Designing drivers for our test cases both in Fortran and C++
- Designing the testing framework and CI
- Implementing the dwarfs using the Gridtools library

The following tools were used:

- Gitlab server at IMGW
- Modular cmake building environment for Fortran and C++ codes; including generation of the Eclipse projects
- Attempts to minimize efforts related to C++ GridTools complexity as for autumn 2016.
- Designing drivers for analytical test cases both in Fortran and C++ for the diffusion and advection miniapps
- Designing the testing framework and CI
- Implementing the dwarfs using the Gridtools library

Things to consider

- Achieving convergence with the dual C++/Fortran dynamical core of COSMO
- Minimizing the future effort for Uli
- Optimizing the exchange of know-how in fast-developing GridTools environment (DSL, testing and CI frameworks
- Do you like the idea of NWP dwarfs in COSMO ?