ALADIN-HIRLAM Newsletter

No. 9, September 7th, 2017

Joint 27th ALADIN Workshop & HIRLAM All Staff Meeting 2017
03-06/04/2017, Helsinki, Finland

ALADIN Programme, c/o P. Termonia, IRM, Avenue Circulaire 3, 1180 Bruxelles, Belgium

HIRLAM-C Programme, c/o J. Onvlee, KNMI, P.O. Box 201, 3730 AE De Bilt, The Netherlands

CONTENTS

Introduction

Patricia Pottier, Frank Lantsheer

Welcome to the combined 9th edition Newsletter of the HIRLAM and ALADIN consortia.

This summer 2017 edition is supposed to be mainly dedicated to the "***27th ALADIN Workshop & HIRLAM All Staff Meeting 2017***" that took place on 3-6 April 2017 in Helsinki (Finland).

We like to thank all those that contributed to this Newsletter with their articles. However, not so many articles were received. Therefore, we have added the full list of talks and posters with a (direct) link to the pdf of the presentations.

Please be reminded that the **sessions have been recorded and can be viewed on youtube.**

Furthermore a special focus is proposed on 2 specific articles:

- AROME for nowcasting (AROME-NWC)
- Tuning the implementation of the radiation scheme ACRANEB2

The list of ALADIN/HIRLAM events planned for the second semester of 2017 and later on as far as now known is also added to the Newsletter (for actual information please check the websites).

We hope you enjoy reading the ninth ALADIN-HIRLAM Newsletter. Thanks for all authors for their contributions and hand it off first to Patricia for the Edito.

Patricia and Frank

For additional information, please visit the ALADIN and HIRLAM websites, or just ask the authors of the articles.

Edito : ALADIN-HIRLAM : a long story to make longer

Patricia Pottier

From the first contacts to the aim of forming one single consortium

After the **first discussions between the HIRLAM Management Group (Per Unden) and ALADIN (Jean-François Geleyn) at the autumn 2003** besides a HAC meeting in Toulouse, many ALADIN-HIRLAM collaboration acts occurred in 2004 : training course, granted licence for research, full code cooperation …

In October 2003: first contacts and discussions between ALADIN (Jean-François Geleyn) and HIRLAM (Per Unden), taking the opportunity of a HAC meeting in Toulouse.

On March 15-19, 2004 a training course on ALADIN-NH was organised in Toulouse for 17 ALADIN & ALADIN students whereas 6 teachers gave 27 hours of lecture.

In April 2004, HIRLAM officially requested for the license of the ALADIN code for research purposes and the HAC proposed a full code collaboration with ALADIN. The HIRLAM Council approved in June 2004.

During the 9th Assembly of ALADIN Partners (Split, 29-30 0ctober 2004), a roadmap and a time-schedule for the cooperation were proposed (see Andras Horanyi and Per Unden presentations at this Assembly). A resolution on the ALADIN-HIRLAM cooperation was unanimously adopted. The HIRLAM Council (Reading, 15 December 2004) gave a very positive answer to this resolution. Different levels of cooperation were established : a full code collaboration on common code with defined rights for usage and for products; scientific/research collaboration on a few number of topics for a beginning, to be discussed at joint HMG/CSSI meetings; management with both consortia maintaining their Assembly/Council or HAC/PAC meetings but with mutual cross-representation.

In June 2005, the HMG/CSSI first meet jointly in Bratislava and have kept meeting jointly each year since then.

Additionally, as the annual work plans were derived during **the HIRLAM ASM and the ALADIN Wk**, it was decided to organize those meetings roughly the same time of the year. In 2006, the ALADIN Wk and the HIRLAM ASM took place in parallel in Sofia with a small cross-participation and, from 2007, the Wk and the ASM have been organised simultaneously at the same venue (on even years in an ALADIN country and on odd years in an HIRLAM country), with more and more common sessions, until 2012 when it ended a fully joint Wk/ASM with all sessions in common.

On December 5, 2005, during the HIRLAM Council, a **Cooperation Agreement between the ALADIN consortium and the HIRLAM consortium** was signed with prime objective "to provide the ALADIN and the HIRLAM Members with a state-of-the-art NWP-model for Short and Very Short Range Forecasting including Nowcasting, for both Research and Development activities and Operational usage". This Agreement specified the conditions for the collaboration between the ALADIN consortium and the HIRLAM consortium and was annexed to their Memorandums of Understanding. This agreement was renewed on December 1st, 2010: the respective strengths and weaknesses of both consortia were founded to complement each other well, with ALADIN and HIRLAM having collaborated more and more closely, giving a new impulse to the European structuring of NWP activities.

In December 2012, the ALADIN General Assembly and the HIRLAM Council created a Task Force to analyse the pros and cons of a further merge. The Task Force prepared a draft merger road map that

was discussed during the **first joint HAC/PAC meeting in May 2013**. Since then, a joint HAC/PAC meeting is yearly held.

In September 2013, the **first combined ALADIN-HIRLAM Newsletter** was published after the decision by the HMG/CSSI to join forces and produce twice a year a common newsletter, as the researchers from both consortia were working closer and closer together.

In December 2014, the ALADIN General Assembly and the HIRLAM Council held their first joint meeting : the Council and the GA set, through a **joint declaration**, strong directions for the PMs to head towards with the aim of forming one single consortium by the end of the 2016-2020 MoUs and five issues to be resolved :

1. code ownership (software IPR)
2. data policy
3. global picture of annual contribution of countries to the various types of activities
4. identification of common activities and specific activities (possibility of core and optional programs);
5. branding (including suitable evolution of the name of the system).

In the HIRLAM-C MoU and the 5th ALADIN MoU, the ALADIN-HIRLAM cooperation agreement has been removed from the MoU texts. It remains as a stand alone document : this agreement was approved on December 8, 2016 by the ALADIN GA and HIRLAM Council at their 2nd joint meeting.

Convergence Roadmap : towards a single consortium in 2021

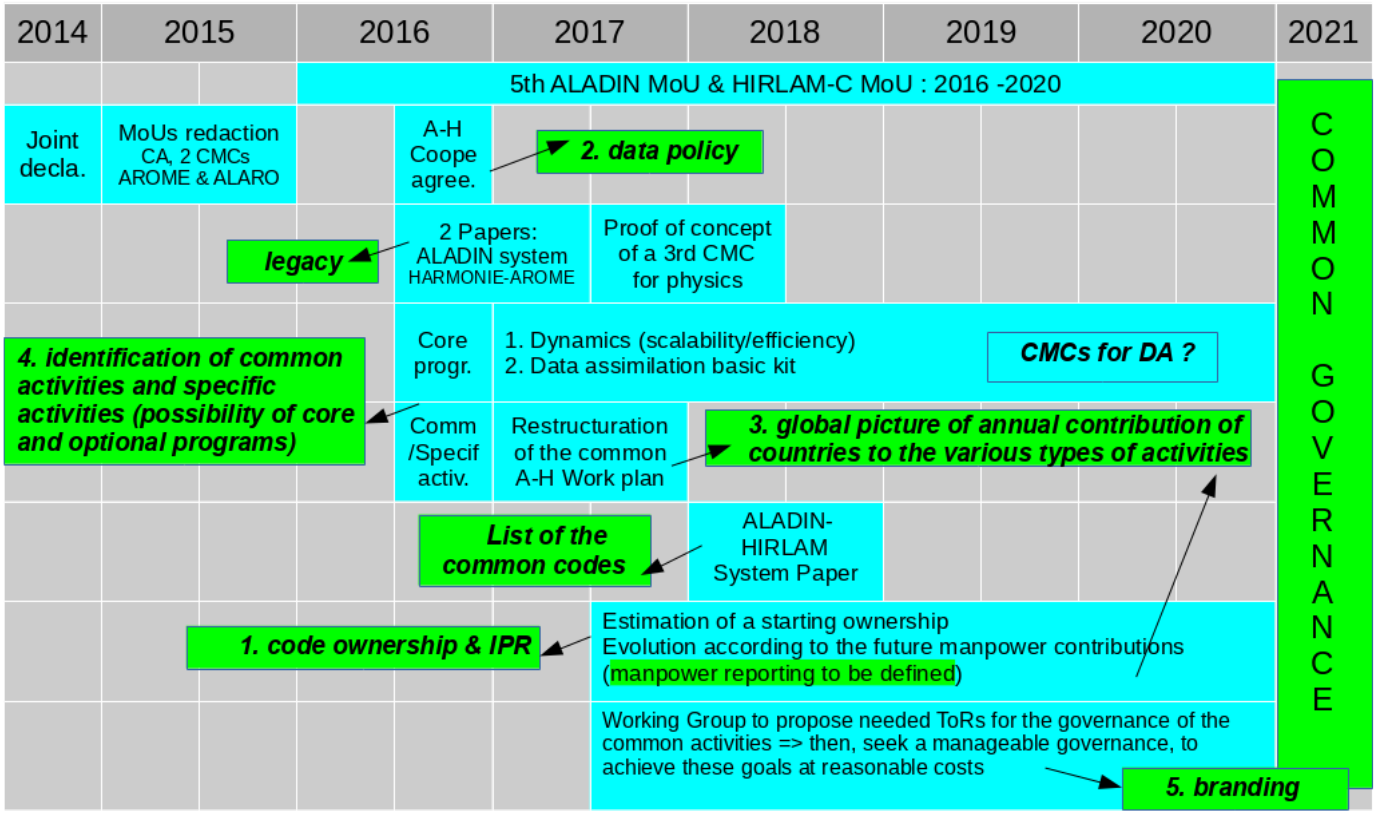


Figure 1: Sequence of steps (actions/events) and the corresponding clarified issue from the 2014 Declaration

More details and explanations on this roadmap were given and approved by the 2nd joint ALADIN GA/HIRLAM Council (Dec. 2016) and the 5th HAC/PAC meeting (May 2017).

Events announced for 2017 (and later on)

The Newsletters only give a static (twice a year) overview with upcoming meetings for the future time frame. Actual information (year rond) is available through the ALADIN / HIRLAM websites. You might find also events of interest through the LACE website.

1 ALADIN/HIRLAM meetings

- 39th EWGLAM and 24th SRNWP meetings, 2-5 October 2017, ECMWF, Reading, UK
- Regular 22nd General Assembly and 3rd joint ALADIN GA/HIRLAM Council, 21-22 November 2017, Cracow, Poland

In 2018 and after :

- Joint 28th ALADIN Workshop/HIRLAM All Staff Meeting 2018 , 16-20 April 2018, Toulouse, France
- 40th EWGLAM and 25th SRNWP meetings, 1-4 October 2018, Austria (place t.b.d.)
- 6th joint HAC/PAC meeting, October 2018, Prague (Czech Republic)
- 4th joint ALADIN GA/HIRLAM Council (location and date t.b.d.)

- 2019: Joint 29th ALADIN Wk/HIRLAM ASM 2019 will be hosted by AEMET in Madrid, probably on April 1-4, 2019.

2 ALADIN/HIRLAM Working Weeks / Working Days

- Radiation Side Meeting EMS 2017, Dublin, Ireland, 5 September 2017
- HARMONIE workshop on physics, 10-14 September 2007, Helsinki
- September 18-20, 2017, Ljubljana (Si) : joint LACE Data Assimilation Working Days & HIRLAM/ALADIN/LACE/SURFEX Surface Working Days.
- Lake workshop 2017 in Berlin: http://www.flake.igb-berlin.de/Lake17/

Furthermore the following topics through working weeks will be adressed:

- Data assimilation algorithms
- Use of Observations
- Radiation, clouds, aerosols (March 2018, at ECMWF, invited by Richard Hogan)
- Surface, focus on modelling
- Surface, focus on DA
- EPS
- HARP meeting
- System
- WG Hi-res modelling

Training (provisional):

- DA-oriented course
- Training/webinar for developers

3 Regular video meetings

Following the positive outcome of the initiative by Roger Randriamiampianina to organize regular group video meetings (via google hangouts) for Data Assimilation staff (from both ALADIN and HIRLAM), these type of group video meetings were also organized for System and Scalability by Daniel Santos Munoz and for Surface by Patrick Samuelsson.

They will be happy to help you if you plan to set up your own group video meeting.

4 About the past events

Find on-line information about the past ALADIN-HIRLAM common events such as the joint ALADIN Workshops & HIRLAM All Staff Meetings, the minutes of the HMG/CSSI meetings, the joint HAC/PAC meetings, the joint ALADIN General Assemblies and HIRLAM Councils.

27th ALADIN Wk & HIRLAM 2017 ASM

Hosted by the Finish Meteorological Institute (FMI), Finland
April 3-6, 2017 in Helsinki

Not so many of those who presented something at the last Wk&ASM have found time to write down an article around their slides/posters. Therefore, an overview is given with the recorded sessions as well as the full list of presentation (in pdf link). In case you would like to know more about a presentation, don't hesitate to contact directly its author !

The sessions can be viewed on youtube :

- Opening session: https://youtu.be/ZBiPOgxBMdI
- Plenary session 1 on DA : https://youtu.be/SiFzVg2A-kI
- Plenary session 1 on DA (cont) : https://youtu.be/j5h-HRzvCuA
- Plenary session 2 on Dynamics : https://youtu.be/d4BVLyTrQq4
- Plenary session 3 on Physics : https://youtu.be/rWaK5EWakak
- Plenary session 3 on Physics (cont) : https://youtu.be/ed87zLF0nbQ
- Plenary session 4 on Quality, User and climate aspects : https://youtu.be/GXHK5ZYWuHU
- Plenary session 4 on Quality, User and climate aspects (cont) : https://youtu.be/CaQcpcr1z2c
- Plenary session 5 on Surface : https://youtu.be/stmYiz_XZwg
- Plenary session 6 on System and Scalability issues : https://youtu.be/wpp6xXAYTPI
- Plenary session 6 on System and Scalability issues (cont) : https://youtu.be/yvTWGThDJI8
- Plenary session 7 on EPS : https://youtu.be/Y3fXeUYn1Oo
- Plenary session 7 on EPS (cont): https://youtu.be/rGHF8U5cstU
- Plenary closing session: https://youtu.be/0KR6Bj84-dI

The presentations and posters in pdf are available through the below links.

Presentations

Plenary opening session

- Recent HIRLAM highlights : Jeanette Onvlee
- ALADIN status overview : Piet Termonia
- Status of the EUMETNET C-SRNWP project : Balazs Szintai

Plenary session 1: Data Assimilation

- Progress and plans of global data assimilation at MF : Claude Fischer
- HIRLAM atmospheric data assimilation : Roger Randriamampianina
- The latest data assimilation activities in LACE countries : Mate Mile
- Appropriate Bmatrix for BlendVar : Antonin Bucanek
- OOPS developments at ECMWF : Roel Stappers
- Tests on cloud initialisation with AROME over Austria and Germany : Florian Meier
- Cloud (NWCSAF) assimilation in Harmonie : Erik Gregow
- Assimilation of ATOVS and GNSS ZTD observations in Aemet : Jana Sanchez Arriola
- Assimilation of GNSS ZTD from the NGAA processing centre : Martin Ridal
- An update on observation processing : Eoin Whelan

Plenary session 2: Dynamics

- The dynamics core program : Steven Caluwaerts
- Current Research and Developpment in dynamics at Meteo-France : Ludovic Auger
- LACE-problems in dynamics & coupling and some solutions : Petra Smolikova
- Local semi-implicit scheme : Filip Vana

Plenary session 3: Physics

- Progress and plans in the ARPEGE and AROME models physics : Francois Bouyssel
- ALARO status overview : Neva Pristov
- Cloud and convection in Alaro-1 : Luc Gerard: Article in this Newsletter: Deep convection
- Convective Precipitation in HARMONIE-AROME : Lisa Bengtsson
- Current development of the cloud microphysics within MetCoOp : Karl-Ivar Ivarsson: Article in this Newsletter: Recent development of cloud micorphysics (ICE3) within Met CoOp
- 1D and 3D evaluation of several options for PBL schemes with a focus on low cloud : Eric Bazile
- Experiences with sub-km HARMONIE-AROME : Xiaohua Yang
- Comparison of HIRLAM radiation fluxes to surface observations : Laura Rontu : Article in this Newsletter: HIRLAM and HARMONIE-AROME radiation comparisons

Plenary session 4: Quality, User and climate aspects

- HARP news : Christoph Zingerle
- Status on Verification and Quality Assurance in HIRLAM-C : Bent Sass
- Operational HARMONIE-AROME issues (clouds and precipitation) : Sander Tijm
- High-resolution operational ALADIN System for forecasting meteotsunamis : Martina Tudor: Article in this Newsletter:High-resolution operational NWP for forecasting meteotsunamis
- The Reanalyses and Data produced in UERRA : Per Unden
- Studying the local microclimate in Gent : Steven Caluwaerts
- Met Eireann 35-year regional reanalysis : Emily Gleeson

Plenary session 5: Surface

- Overview of HIRLAM surface progress and plans : Patrick Samuelsson
- Preliminary tests of the CMC ALARO-1 coupled to SURFEX-8 using CY43T2 over Belgium : Rafiq Hamdi
- FLake in Harmonie : Ekaterina Kurzeneva
- Lake Surface Water Temperature autocorrelation functions : Margarita Choulga
- Impact of the Sea Surface Temperature in operational forecast using ALADIN System : Martina Tudor
- Using wave forecast model to estimate the accuracy of surface wind fields in the Baltic Sea : Laura Tuomi

Plenary session 6: System and Scalability issues

- System status: about IFS cycles, about OOPS : Claude Fischer
- Harmonie-Arome system: Status and future : Daniel Santos Munoz
- LAM features in the ATLAS framework : Daan Degrauwe : Article in this Newsletter: The Atlas framework and LAM features therein
- Single precision IFS : Filip Vana
- Recent progress on the ACRANEB2 dwarf from the ESCAPE project , Part I : Per Berg
- Recent progress on the ACRANEB2 dwarf from the ESCAPE project, Part II : Jacob Weismann Poulsen
- ESCAPE status on ACRANEB2: Code efficiency tests : Bent Hansen Sass

Plenary session 7: EPS

- Developments on HarmonEPS and GLAMEPS : Inger-Lise Frogner
- LAM-EPS activities in LACE : Martin Bellus
- Experiences and challenges with MetCOoP EPS : Ulf Andrae
- RMI-EPS: a prototype convection-permitting EPS for Belgium : Geert Smet
- Stochastic perturbation of partial tendencies in AROME : Clemens Wastl
- Surface perturbations in HarmonEPS : Andrew Singleton
- The growth of ensemble spread in mesoscale NWP model due to LB and IC perturbations : Jure Cedilnik

Plenary closing session

- Outcomes of the meeting
- Announce of the 2018 meeting

Posters

- Abdenour Ambar : Desert dusts modeling in AROME: Contribution of physical parameterizations at convective scale
- Florian Meier : National poster Austria
- Alex Deckmyn : National poster Belgium
- Boryana Tsenova : News in numerical weather prediction in Bulgaria
- Rilka Valcheva : The Impact of Climate Change (until 2050) on the potential of renewable energy sources (wind and solar radiation) for the territory of Bulgaria
- Tomislav Kovacic : ALARO at 4 km forecast post processing using Kalman filter
- Iris Odak Plenkovic : Probabilistic Wind Speed Predictions with an Analog Ensemble
- Martina Tudor : ALADIN in Croatia
- Petra Smolikova : Numerical Weather Prediction@ Czech Hydrometerological Institute
- Kristian Pagh Nielsen : ESCAPE: Optimizing physics subroutines for multithreaded
- Erik Gregow : Cloud (NWCSAF) assimilation in Harmonie (Note: In case of no oral presentation) : Article in this Newsletter: Harmonie – MSG cloud data-assimilation experiments
- Markku Kangas : Mast Verification
- Patricia Pottier : The NWP systems at Meteo-France : Article in this Newsletter (AROME Nowcasting part of the poster)
- Viktoria Homonnai : Low cloud experiments with AROME over Hungary
- Balazs Szintai : NWP at the Hungarian Meteorological Service
- Emily Gleeson : Poster on radiation work
- Eoin Whelan : Operational poster for Ireland
- Siham Sbii : Status of numerical weather prediction in Morocco
- Yurii Batrak, Bin Cheng : Sea ice mass balance in the Arctic Ocean
- Roger Randriamampianina : Impact of Atmospheric Motion Vectors on NWP over high-latitudes and the Arctic
- Bogdan Bochenek : Polish national poster
- Maria Monteiro : ALADIN – Portuguese Technical and Scientific Activities
- Raluca Iordache : ALADIN activities in Romania
- Maria Derkova : ALADIN related activities @SHMU
- Benedikt Strajnar : ALADIN in Slovenia 2017
- Javier Calvo : AEMET operational suit: HARMONIE Regular Cycle of Reference for cycle 40
- Angeles Hernandez : Assimilation of Atmospheric Motion Vectors in AEMET

- Daniel Martin & Gema Morales : Use of Ceilometer Data for Cloud Validation
- Klaus Zimmermann : OBSMON: New Developments for UERRA and HARMONIE
- Zied Sassi : ALADIN Related Activities in Tunisia
- Guser Alper : NWP Related Activities in Turkey

Working groups, side-meetings, LTM meeting, HMG/CSSI meeting, ...

More information on the 27th Wk&ASM 2017 (agenda and participants, photos) and the side events (WG discussions, LTM meeting, HMG/CSSI meeting) are available on the dedicated page on the ALADIN website.

- WG on Recent upper-air DA issues (Roger Randriamampianina)
- Surface side meeting on Surface processes and data assimilation (Patrick Samuelsson)
- WG on clouds, radiation and aerosol, (minutes on HIRLAM wiki, for authorized access only) (Laura Rontu)
- WG on System
- Verification/HARP side meeting (Bent Hansen Sass, Christoph Zingerle)

- LTM meeting
- HMG/CSSI meeting.

The Atlas framework and LAM features therein

Daan Degrauwe and Willem Deconinck

1 Introduction

Atlas is a framework for parallel data structures, being developed at ECMWF (Wedi, 2015; Deconinck, 2016; Deconinck, 2017). Several motivations led to the decision for this development. First, ECMWF scientists are looking at finite-volume based discretizations on unstructured grids (EULAG model, Prusa (2008)), as an alternative for the structured, spectral discretization of IFS. For the sake of bringing both approaches together, a software framework is required that supports both types of grids (structured and unstructured), and both types of discretizations.

A second motivation for the development of Atlas is the scalability challenge that the NWP community is facing today. In order to prepare the code for (unknown) future hardware evolutions, the flexibility of the current code needs to be enhanced. The goal of Atlas in this context is to provide a software layer that allows to separate scientific issues (e.g. the type of discretization), from technical issues (e.g. the type of hardware). Moving to some new type of hardware then only would require porting and optimizing Atlas, instead of the entire NWP model.

The third motivation for the development of Atlas originates in the ESCAPE project (Horizon 2020, grant No 67162). The key idea of this project is to break down NWP models into fundamental algorithmic building blocks (so-called NWP dwarfs), and to improve the scalability and energy-efficiency for these dwarfs separately. Atlas provides a framework that is common between these dwarfs.

Given the close connection between the models of the ALADIN-HIRLAM system and the IFS model, it is important that our LAM community follows ECMWF's development of Atlas, and makes sure that Atlas supports LAM models (Figure 1). It goes without saying that it is preferable to do this already at an early stage in the development of Atlas. This document shows the work that has been done to introduce LAM features in the Atlas framework. The next section gives a very limited overview of the aims and design of Atlas. In section 3, LAM-specific features in Atlas are described. Section 4 provides some conclusions.

2 The Atlas framework

2.1 Generalities

Since one of the motivations for Atlas is to enhance the flexibility of dealing with parallel data structures, it was decided to develop Atlas in an object-oriented way. This greatly improves the runtime flexibility and allows for an abstraction of low-level implementation details. For this reason, Atlas is mainly developed in C++, although a complete interface to Fortran is maintained. This makes it possible to access Atlas functionality from our current models, which are mainly Fortran-based. Atlas also follows modern coding standards such as unit-testing, in order to improve the reliability of the code.

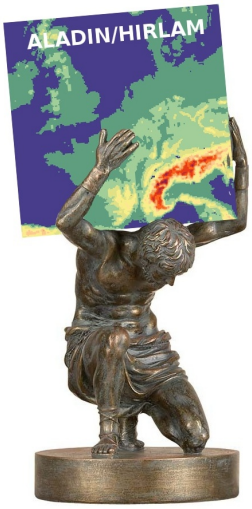


Figure 1: The ancient Greek titan Atlas supporting a LAM model.

Important to notice here is that Atlas is a support library, rather than an NWP system such as OOPS. As such, the introduction of Atlas features in the IFS/ALADIN-HIRLAM code can be done progressively, and does not require any disruptive steps.

2.2 Tasks of Atlas

Since Atlas acts as a layer between the NWP model and the hardware, its list of tasks is pretty extensive. So far, following features are considered:

- Grid and mesh generation;
- Parallelization and communication patterns;
- Data structures in parallel and heterogeneous hardware environments (e.g. GPU's);
- Interpolation algorithms (e.g. to change from one grid to another, but also for semi-Lagrangian advection schemes);
- I/O support;
- Logging with flexible level of detail.

2.3 Classes in Atlas

2.3.1 The grid class

A grid in Atlas is a merely collection of gridpoints. A hierarchy of grid types is defined in Atlas, as shown in Figure 2. It should be noted that the grids of IFS and ARPEGE (reduced Gaussian grids) and those of the ALADIN-HIRLAM models (regular grids) are supported.

2.3.2 The mesh class

A mesh object in Atlas describes the connectivity of the gridpoints (although for a spectral model, this connectivity is less relevant). The mesh object is a distributed object: every processor only treats one part of the

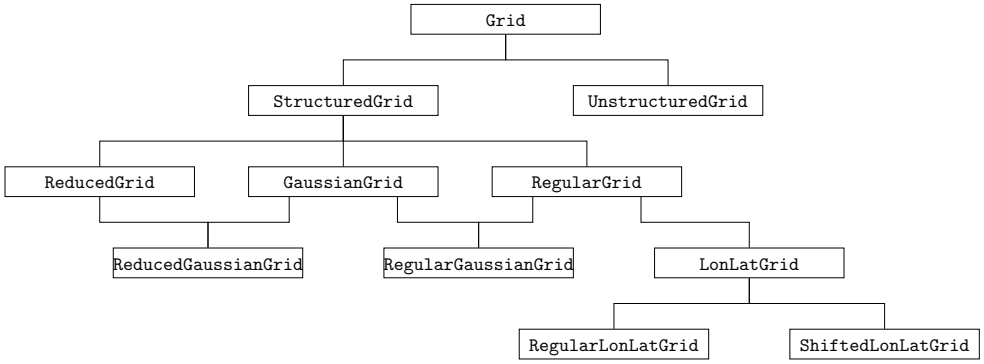


Figure 2: Grid types hierarchy.

complete mesh, as shown in figure 3. Also noteworthy is that each mesh part also includes information about a halo, so that communications with processors treating neighbouring mesh parts are readily set up.

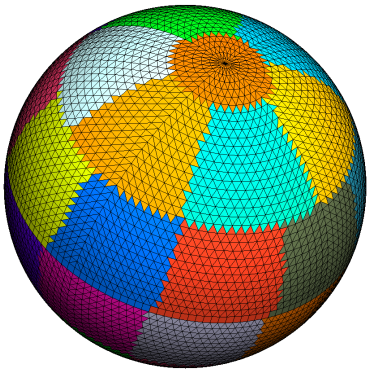


Figure 3: A mesh on an octrahedral grid with $N = 32$, partitioned into 32 parts.

2.3.3 The field and functionspace classes

A field in Atlas is a numerical representation of an atmospheric variable. However, it should be noted that this numerical representation depends on the type of discretization: in a spectral model, this is by the spectral coefficients, whereas in a finite-volume model, this can be by cell-center values, egde-center values, etc.

In order to cope with this variety of possible representations, the functionspace class is introduced in Atlas. Numerical operators such as the ∇-operator are then defined within a specific functionspace. Here, the benefit of the object-oriented design becomes clear: the developer does not have to worry about the underlying discretization of a field: he/she just can take the gradient of a field, and Atlas makes sure that the appropriate calculations are done, depending on whether the field is defined in a spectral, finite-element, or finite-volume functionspace. Moreover, Atlas also decides on the actual memory layout of the field, so that portability and efficiency on different hardware platforms are ensured.

Figure 4 shows the overall workflow and the relation between these classes.

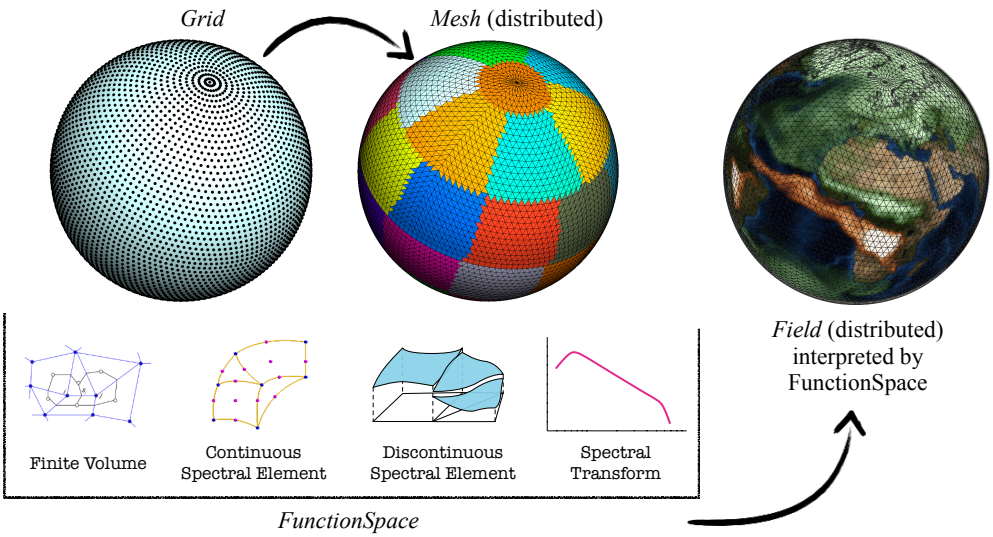


Figure 4: Workflow in Atlas.

3 LAM features in Atlas

3.1 Grids and projections

Limited-area models usually act on a (planar) projected grid. This requires the distinction between the geographic coordinates (longitude and latitude) of a gridpoint, and its grid coordinates. The conversion between both coordinate spaces is done by a projection. Several projection types have been introduced in Atlas, which satisfy the needs of the models of the ALADIN-HIRLAM system:

- (rotated) longitude-latitude
- conformal Lambert
- (rotated) Schmidt (ARPEGE stretching)
- (rotated) Mercator

3.2 Meshes and partitioners

When partitioning a global mesh, it is beneficial to assign the region around a pole to a single processor (as shown in figure 3). However, for LAM meshes, it is better to keep the number of processors per latitude band constant, leading to a checkerboard pattern. To this goal, a new mesh partitioner has been introduced in Atlas. Figure 5 shows a distributed (and projected) LAM mesh.

3.3 Boundary conditions

The boundary conditions in the models of the ALADIN-HIRLAM system are applied through a Davies-relaxation. This method acts entirely in gridpoint space, and in this sense, it does not require any specific functionality in Atlas.

However, an implicit assumption in our spectral LAM models, is that the fields are biperiodic. Since this involves the connectivity between gridpoints (one side of the LAM domain is ‘connected’ to the opposite

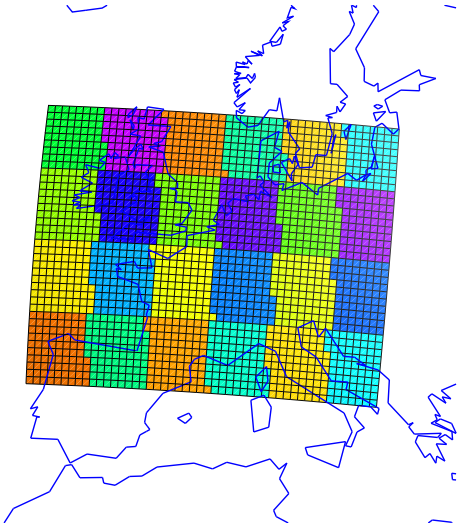
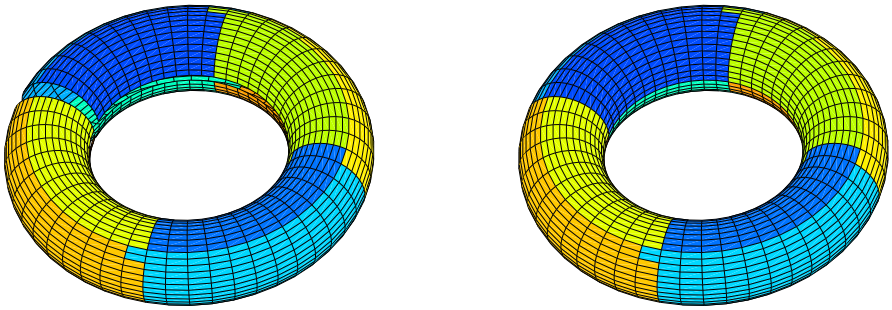


Figure 5: LAM mesh, based on a projected grid, and partitioned with the checkerboard partitioner.

side), biperiodism should be supported by Atlas. A specific mesh generator was introduced in Atlas to create biperiodic meshes, as shown in figure 6.



(a) (b)

Figure 6: Periodic boundary conditions in Atlas: (a) non-periodic LAM mesh (opposite sides not connected); (b) bi-periodic LAM mesh (opposite sides connected).

4 Conclusions

No one can predict the future developments in computer hardware. Several platforms are being developed next to traditional CPU's, e.g. GPU's, co-processors, and optical processors. Besides raw computing power, also the energy-efficiency of these platforms is becoming an important factor.

The Atlas framework is an attempt to prepare our NWP models for the future, without knowing what this future will be. By providing a layer that separates science from hardware-dependend technicalities, it ensures that scientists will be able to keep focusing on their research, while at the same time allowing to run the models efficiently on heterogeneous platforms.

This document describes the work that has been done to ensure that Atlas not only supports global atmospheric models, but also the LAM models of the ALADIN-HIRLAM system. Although it has not been decided yet

when Atlas will enter the IFS code, it is reassuring that the necessary features for LAM modeling are already present.

5 References

Deconinck W., M. Hamrud, C. Kühnlein, G. Mozdzynski, P. Smolarkiewicz, J. Szmelter and N. Wedi: Accelerating Extreme-Scale Numerical Weather Prediction. In: Wyrzykowski R., Deelman E., Dongarra J., Karczewski K., Kitowski J., Wiatr K. (eds) Parallel Processing and Applied Mathematics. Lecture Notes in Computer Science, vol 9574. Springer, Cham, 2016.

Deconinck, W., P. Bauer, M. Diamantakis, M. Hamrud, C. Kühnlein, P. Maciel, G. Mengaldo, T. Quintino, B. Raoult, P. Smolarkiewicz and N. Wedi: Atlas, a library for numerical weather prediction and climate modelling. To appear in Computer Physics Communications, 2017.

Prusa, J.M., P.K. Smolarkiewicz, and A.A. Wyszogrodzki: EULAG, a computational model for multiscale flows. Comput. Fluids., 37, 1193-1207, 2008.

Wedi N.P., P. Bauer, W. Deconinck, M. Diamantakis, M. Hamrud, C. Kuehnlein, S. Malardel, K. Mogensen, G. Mozdzynski and P.K. Smolarkiewicz: The modelling infrastructure of the Integrated Forecasting System: Recent advances and future challenges. ECMWF Technical Memorandum 760, 2015.

Cloud and convection in Alaro-1

Luc Gerard

1 Introduction

The introduction into Alaro-1 of the new complementary subgrid draught scheme (CSD, Gerard 2015) for deep convection shed fresh light on the old problem of non-unique cloud representation in the model and the limitations of the statistical cloud scheme. A first aspect is to cleanly re-separate the tunings of the condensation adjustment from the cloud diagnostic used in radiation. This is because the two work differently, while their tuning were badly entangled. The second step is to try to use directly the prognostic cloud fraction and cloud condensates into the radiation scheme instead of re-diagnosing distinct values for it.

2 Context: CSD specific features

Complementary subgrid draughts

The CSD approach is a further development of the 3MT scheme aiming at handling more consistentlty some flaws of the latter at convection-permitting resolutions. The main feature of CSD is to be scale-aware, in the sense that its signal gradually fades out when the explicit signal associated with deep convection increases. Table 1 synthetizes the main conceptual differences.

Table 1: Compared features of 3MT and CSD

	3MT	**CSD**
Triggering	none (local buoyancy diagnostic)	Explicit computation from updraught source layer, resolved-condensation based
Plume	absolute	perturbation
entrainment	tunable arbitrary profile + RH dependence via GENVSRH>0, LENTCH for downdraught effects	Organised + tunable turbulent mixing profile
mesh fraction	prognostic, constant along vertical	prognostic with vertical variation depending on its magnitude
Closure	Prognostic moisture convergence + RMULACVG scaling	Mixed CAPE and MoCon + eddy transport reduction
Fluxes	Single mass flux scaling condensation and transport	Separation of production flux and subgrid transport flux
Parametrisations interaction	Convective cloud fraction Nc protection, calculation limited to area (1-Nc) Microphysics uses an equivalent cloud fraction Neq combining Nc and Ns	Nc stricter protection against re-evaporation but allowing condensation (resolved part of convection). Different Neq formulation for microphysics
Toucans shallow cloud mass flux scheme (acscctr)	use horizontal mixing; top limited by TTE or TKE	no horizontal mixing, top limitation by buoyancy and TKE

Fig. 1 shows a typical difference in behaviour: the subgrid scheme remains more active (more condensation) in 3MT than in CSD, while the cloud scheme in 3MT remains dominated by evaporation over the domain, as it does not contribute to the convective updraughts. In other words, CSD allows a part of the convective condensation to be represented by the cloud scheme, and this part is growing with the model resolution, producing the scale-aware behaviour.

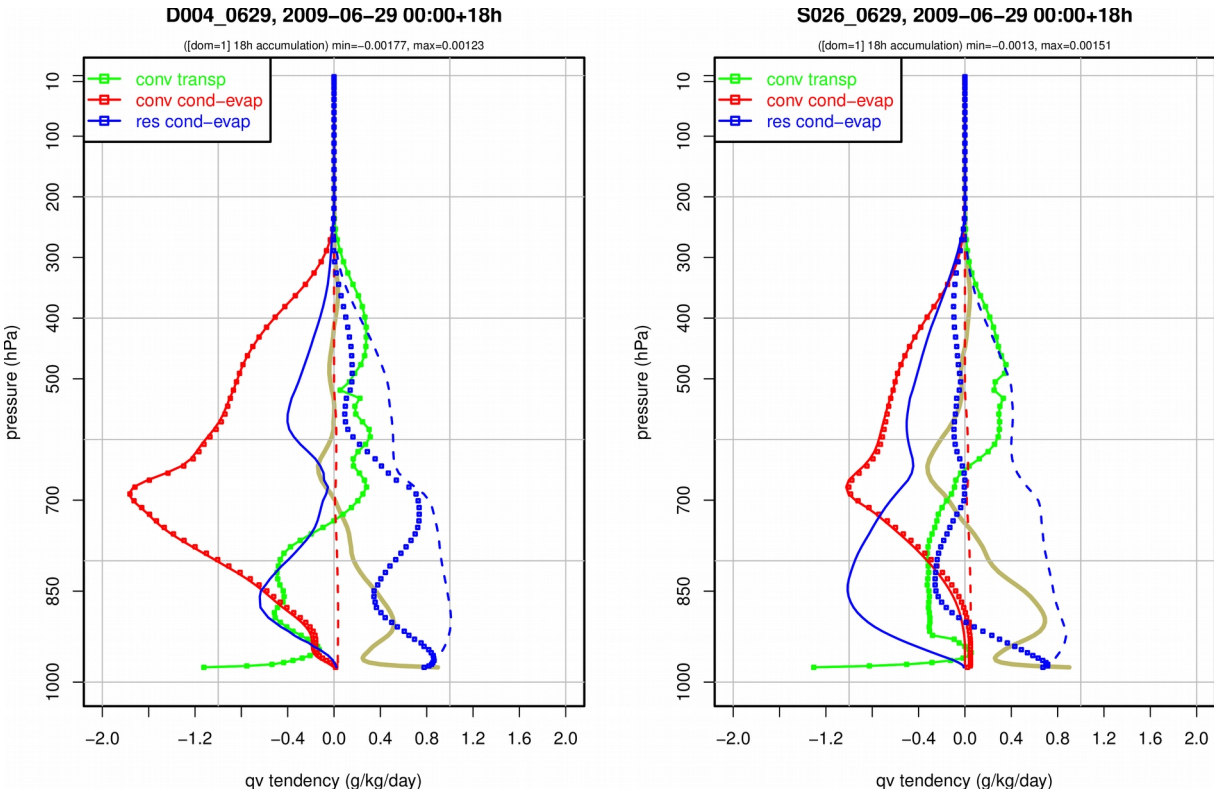


Figure 1: some DDH components of qv tendency for 3MT (letft) and CSD (right). Beige: total tendency, green subgrid transport, blue: cloud scheme condensation (solid)-evaporation (dashed) and sum (with symbols), red: convective scheme condensation-evaporation.

We can also observe on Fig. 1 that the convective cloud base is at the surface with 3MT and higher up with CSD, thanks to triggering parameterization.

3 Clouds and condensates in Alaro

Adjustment

Alaro physics introduced prognostic variables for cloud ice and droplets (and for precipitation). Condensation in the cloud scheme is based on the Xu-Randall (1996) formula combined with a geometrical hypothesis, and completed by a protection of the convective area, to prevent re-evaporation of condensates produced in the convective scheme at previous time step:

$$N = XR[q_c, q_t, q_w] \quad \text{and} \quad q_v = q_w.N + H[z, phase, \Delta x].q_w.(1-N)$$

where $H[z, phase, \Delta x]$ is a critical relative humidity profile, N the cloud fraction, qc, qv, qt, qw the cloud condensate, vapour, total and wet bulb speicific humitities. The cloud scheme delivers values of condensation-evaporation, responsible for latent heat conversion and modification of the water phases contents. The initial XR formula was actually describing an equilibrium, while here, a

substantial fraction of the condensate q_c it delivers is assumed to be precipitated in the microphysical scheme. This implies that the final q_c after precipitation will break this equilibrium. To try to compensate this inconsistency, the gain α = QXRAL_ADJ in XR function, that should be 100 in the original equilibrium formulation, had to be increased to 130 or 150 in this implementation.

Radiative cloud properties

The radiative scheme needed estimation of cloud fraction and cloud condensates well before the introduction of Alaro. To avoid moving everything at once, a separate approach was kept when introducing Alaro.

The stratiform cloud fraction is essentially diagnosed by estimating the oversaturation with respect to a relative humidity profile $q_{cs}=q_t-H'[z].q_{sat}$, where the critical relative humidity $H'[z]$ is a threshold of q_t/q_{sat} unlike $H[z,phase,\Delta x]$, and has nothing to do with the geometrical criterion of the adjustment scheme. As Alaro introduced specific convective components, the convective condensation is estimated by inverting the Xu-Randall formula: $q_{cc}=XR^{-1}[N_c,q_t,q_w]$ and the total radiative cloud is then re-built using $N=XR[q_{cc}+q_{cs},q_t,q_w]$.

This formulation posed the following problems:

1) The difference between H and H' has been subject to confusion, and their respective tunings are actually badly entangled with each other: the coefficients HUCOE, HUTIL1, HUTIL2 define the shape and amplitude of the radiative H'; the H for adjustment is derived from H', with additional dependency to phase and grid spacing through the parameters SCLESPR, SCLESPS, and a further modification of amplitude by HUCRED. However the shapes of the two functions remain very interdependent.
2) The radiative cloud diagnostic is strongly dependent on the convective cloud fraction Nc (value from previous time step) , and the CSD induced a significant shift in this concept, since the cloud scheme can directly represent a susbstantial part of the convective condensation, while the convective fraction Nc is still allowed to grow up to 1. In this case, re-estimating the convective condensation from Nc as a complement to the resolved part is inadequate.

Separating the tunings

A first step was to cure the entanglement, by separating completely the parameters of H from those of H': new parameters HUCO_ADJ, HUT1_ADJ, HUT2_ADJ replace HUCOE, HUTIL1, HUTIL2 for the adjustment. We then observed that with CSD, H should better vary little along the vertical, contrary to H' (Fig. 2). This way, the model can produce realistic results with radiative diagnostic condensates closer to the prognostic cloud condensates (Fig. 3).

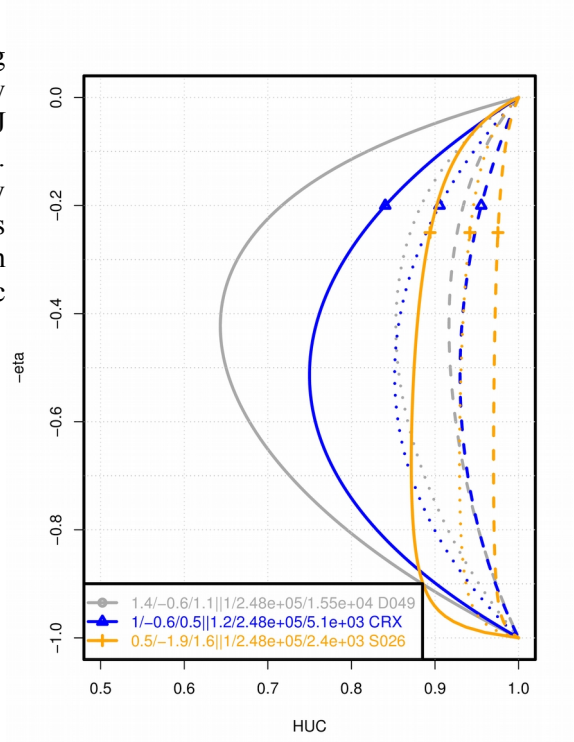


Fig. 2 vertical profile of critical relative humidities: radiative H' (blue, solid), initial H (blue, dash for ql and dot for qi) and new H (orange). In grey, old tuning of H, H' in Alaro-0.

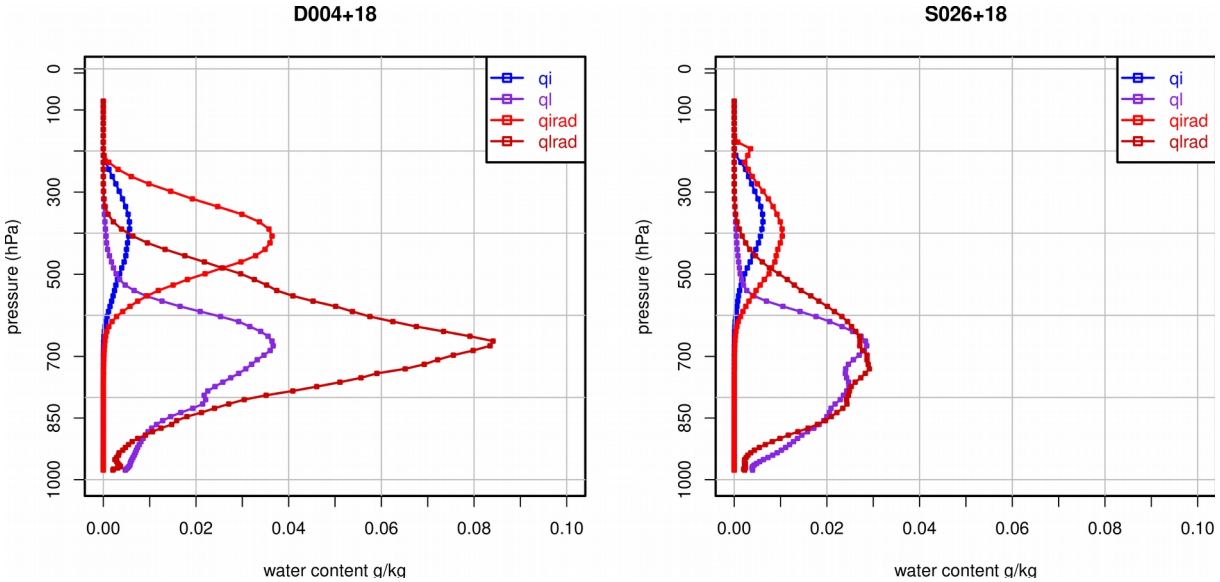


Figure 3: prognostic cloud condensates (blue qi, violet ql) vs values re-diagnosed for radiation (red qi_rad, dark red ql_rad), with initial H and 3MT (left), new H and CSD (right).

Re-unifying the clouds

The next step is to use the prognostic condensates and cloud fraction directly in the radiation scheme, instead of re-diagnosing them. This work is going on, it remains a hard job to find an adequate tuning maintaining the scores in different summer and winter situations.

4 Conclusion

The quest to model scores (verifpack) remains a difficult task in Alaro physics. DDH tools can help the tuning process, but the budgets are still incomplete and sometimes difficult to interpret. We also notice that it is risky to focus on one specific episode, that can be defavorable for the new scheme, while many other situations are improved by it. The scores are made at a given resolution, while local model implemetations use different resolutions and some specific tunings; further the multi-resolution behaviour can only be tuned by intercomparison, without re-running the full verification package.
A final remark is that Alaro physics still has big challenges to take up, and that more effort and human resources need to be put on it, in order to keep Alaro competitive in operation as well as in climate applications.

5 References

Gerard, L., 2015: Bulk mass-flux perturbation formulation for a unified approach of deep convection at high resolution, Mon. Wea. Rev., **143**, pp 4038-4063.

Xu, K.-M. and Randall, D.A., 1996: A semi-empirical cloudiness parameterization for use in climate models, J. Atm. Sci. **53**, pp 3084-3102.

Harmonie – MSG cloud data-assimilation experiments

Erik Gregow

1 Introduction

The FMI target is to ingest cloud information into Harmonie, in order to improve the cloudiness and radiation forecast at short forecast lenghts. This is done according to the method described in van der Veen (2013). There is a cooperation with SMHI who have implemented the above method into Harmonie version c38h12. This code has been shared with FMI to be used for running experiments and for which we here present the first preliminary results (from February 2017). FMI is closely cooperating with KNMI and SMHI in this development.

Experiment setup at FMI:

1. Usage of Harmonie version c38h12 with modified ingest routines and micro-physics, in order to adopt the new cloud ingestion, which has been setup at the ECMWF HPC-facilities.
2. Harmonie setup for MetCoOP-area and with 3h cycling, where main 48h forecasts are run at 00 and 12Z.
3. Running 1 full month experiments for July 2016, with following outcome:

1) One reference run (i.e. with no cloud DA), hereafter “Ref Exp” [Done]

2) One cloud DA run using S. Van der Veen methodology has been performed (i.e. using MSG-NWCSAF cloud-mask and cloud-top temperature, and cloud-base height from interpolated Synop observations). The run is ready and the results are being analysed, see first results here below. Hereafter named “MSG Exp”. [Done, December 2016]

3) Start one cloud DA run with slightly different approach. Similar to 2) the cloud-mask and cloud-top temperature are coming from MSG-NWCSAF products. But the cloud-base height is here retrieved from a static (first-guess) dataset, from SMHI developed algorithms. Hereafter named “MSG-SWE Exp”. [On-going, December 2016]

4) After analysing the results from 2) and 3), new experiments will be setup, this time for 1 week experiment period, within the July 2016 month. The setup of the experiments will be conducted in close cooperation with KNMI and SMHI, in order to improve the results seen from 1) – 3), by focusing on different modifications, solutions and to optimize our efforts in a joint research work. [Planned for Q1 2017]

5) When finding the best approach, from 4) above, FMI will run another 1 full month experiment, where the outcome is aiming to improve the results seen in 1) – 3), and described in the below result section. [Planned for Q2-Q3 2017]

2 Results

The first and preliminary results from the MSG Exp have been validated against the reference run (Ref Exp) and observations. The tool WebGraf has been used to verify against surface and upper-air observations. Through extracting gridpoints information from the Harmonie output the radiation parameters have been verified against Finnish stations. The main conclusion is that the ingestion of

clouds in MSG Exp does give a slight negative impact to the forecast skills; too much clouds being added, a negative bias in surface temperature and a slight negative bias to the upper-air parameters. Also the radiation verification indicate a negative bias, i.e. the global SW radiation is too low. Though, with the intention to develop this method (according to the planned updates described above), there is a clear potential to improve the results and to add value to the forecast skills.

From WebGraf it is seen that the total cloud-cover bias becomes larger (see Fig. 1), when using the MSG Exp as in setup 2). The effect persists out to approxmately +40h forecast step. This also affects to the 2 meter temperature, where there is a negative bias (too low temperature) for the first +18h forecast period. From figure 2 it is seen that there are too much cloud generated, i.e. the chance of having 0 Octas cloud has decreased (green line below red line) and consequently higher chances of having 8 Octas cloud (green line slightly higher than the red line)

From the verification of upper-air parameters, the temperature becomes slightly worse with the MSG Exp (mainly between levels 300-700 hPa, for both the 00 and 12Z runs). For the specific humidity the results are a little contradicting, at 00Z run the scores are little better with a positive temperature bias, but for the 12Z run the scores are worse (Fig. 3a-b).

From the histograms (Figs. 4-5) one can see that the Ref Exp mainly gives too high frequency of high-valued radation data, i.e. overpredicts the occurenece of clear skies. The MSG Exp run on the other hand, overestimates the cloudiness and reduce the radiation, which can be seen in figure 6, where the red columns are too high at the low end of radiation distribution.

3 Discussion

A discussion of the different cloud-base methods, used in experiments MSG Exp and the MSG-SWE Exp. The differences lay in how the cloud-base is estimated, both methods have their own problems. In setup MSG Exp, i.e. where the cloud-base is taken from an interpolated field of Synop observations, the main problem is in areas between Synop's, especially if there exist a border of different clouds in that area. For example, the Synop observations is placed in an area of low clouds but the nearby area (with no Synop) contains only high Cirrus. Here the low cloud-base information will be transferred to the Cirrus cloud and affect the vertical cloud extension for this area, i.e. creating a much too thick cloud. On the other hand, in setup MSG-SWE Exp the edges/areas of different clouds are mainly well captured but here, the cloud-base is determined from the NWCSAF cloud-type. As a consequence, if there is a high cloud, the NWCSAF product is not able to detect if there exist a layer of cloud beneath. Therefore, there is a clear risk that setup MSG-SWE Exp will miss the low-clouds, if there are sheltering high clouds, and the cloud-base is set too high compared to the reality. As a conclusion one can say that both cloud ingest routines have their weakness and strengths.

Figures

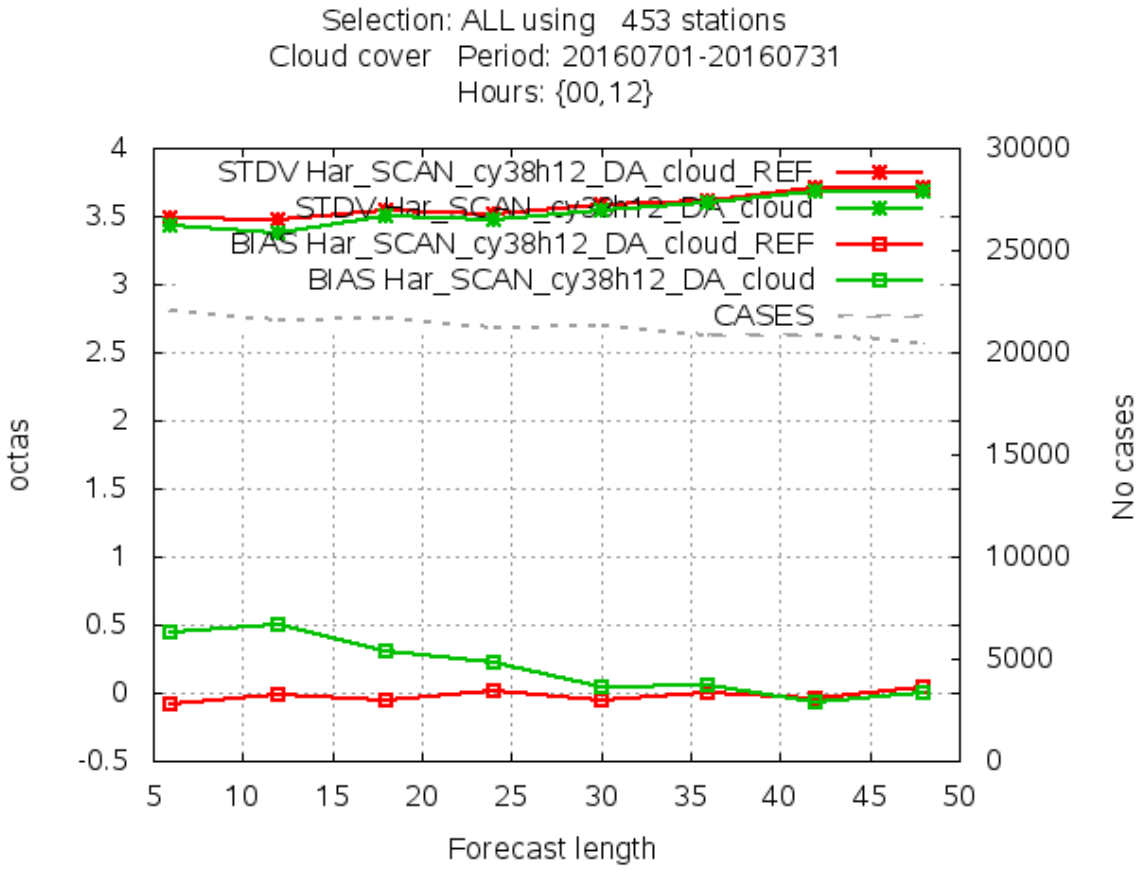


Figure 1. WebGraf: Verification of Ref Exp (red color), and MSG Exp (green color).

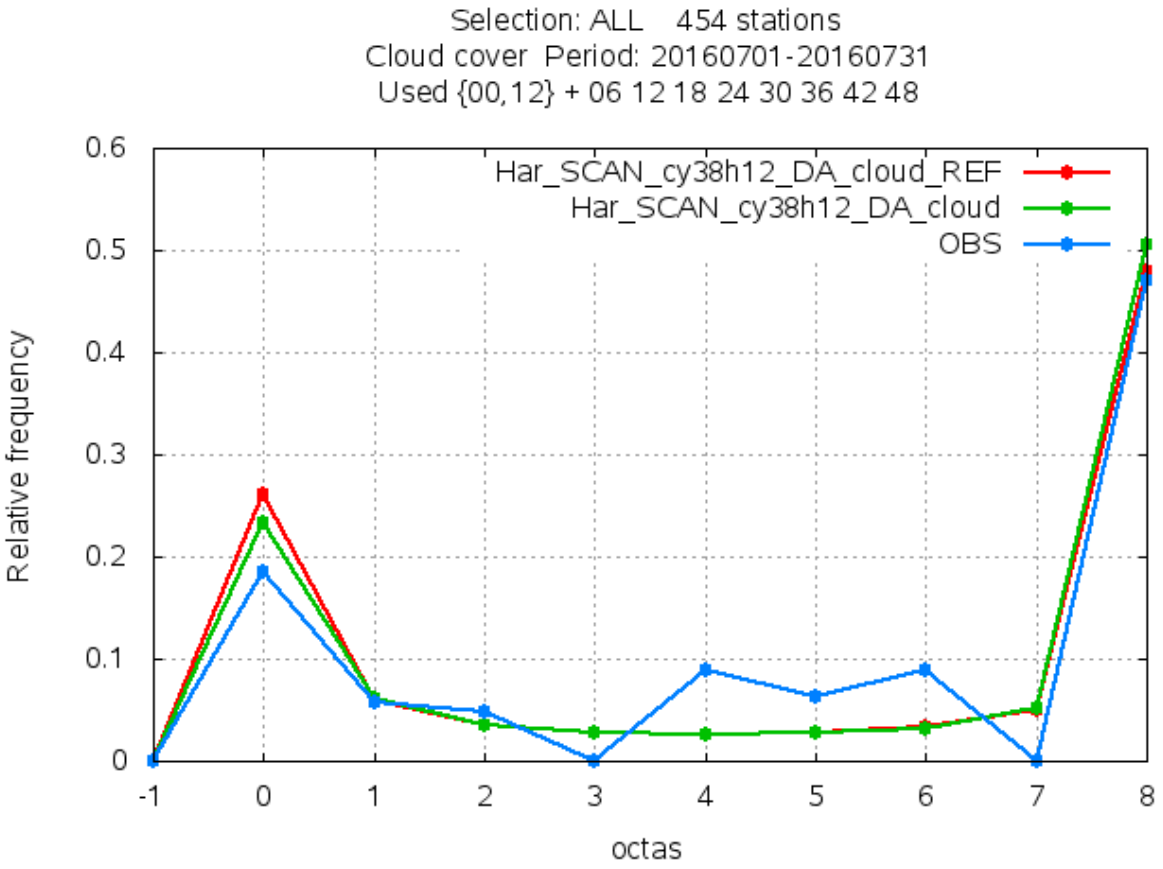


Figure 2. Total cloud cover frequency distribution. Verification of Ref Exp (red color) and MSG Exp (green color).

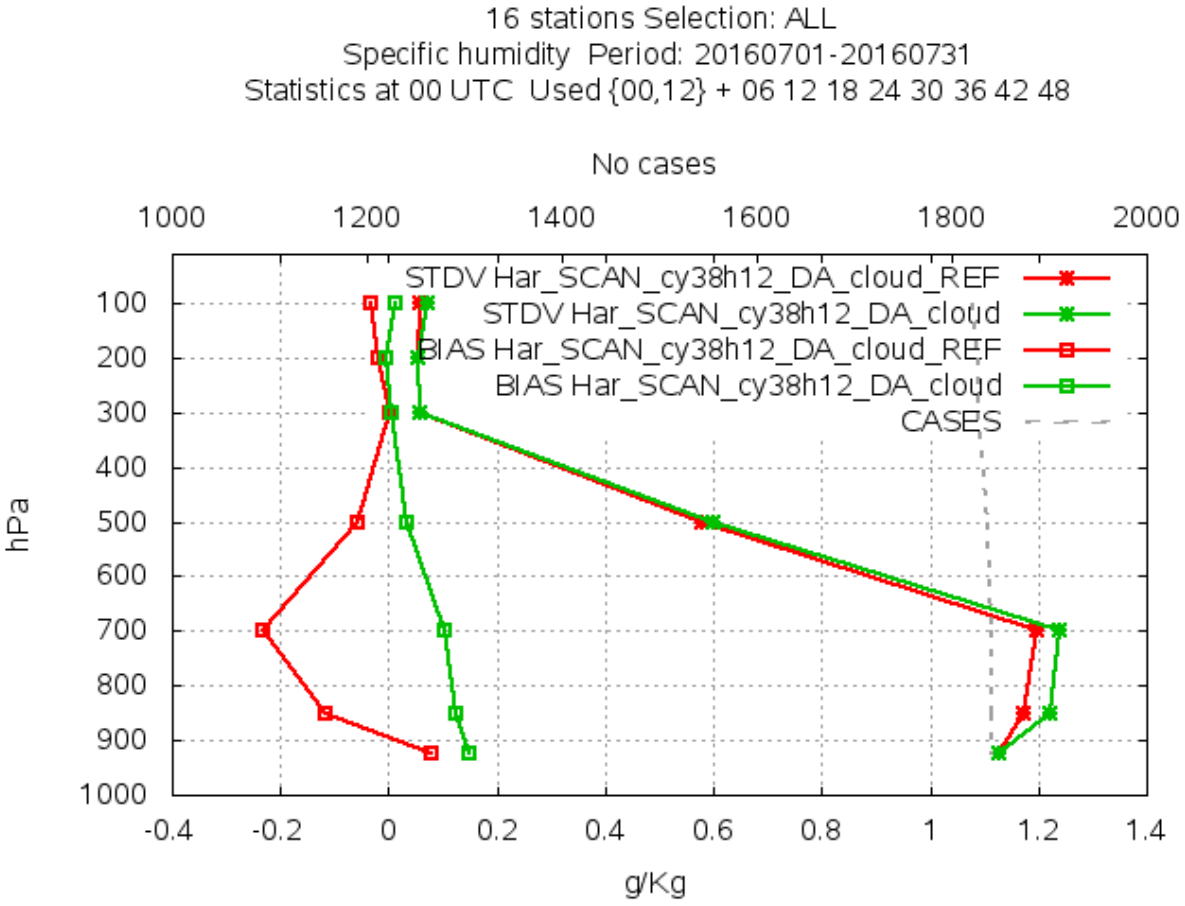


Figure 3a. WebGraf verification of upper-air specific humidity for 00Z run, for Ref Exp (in red) and MSG Exp (in green).

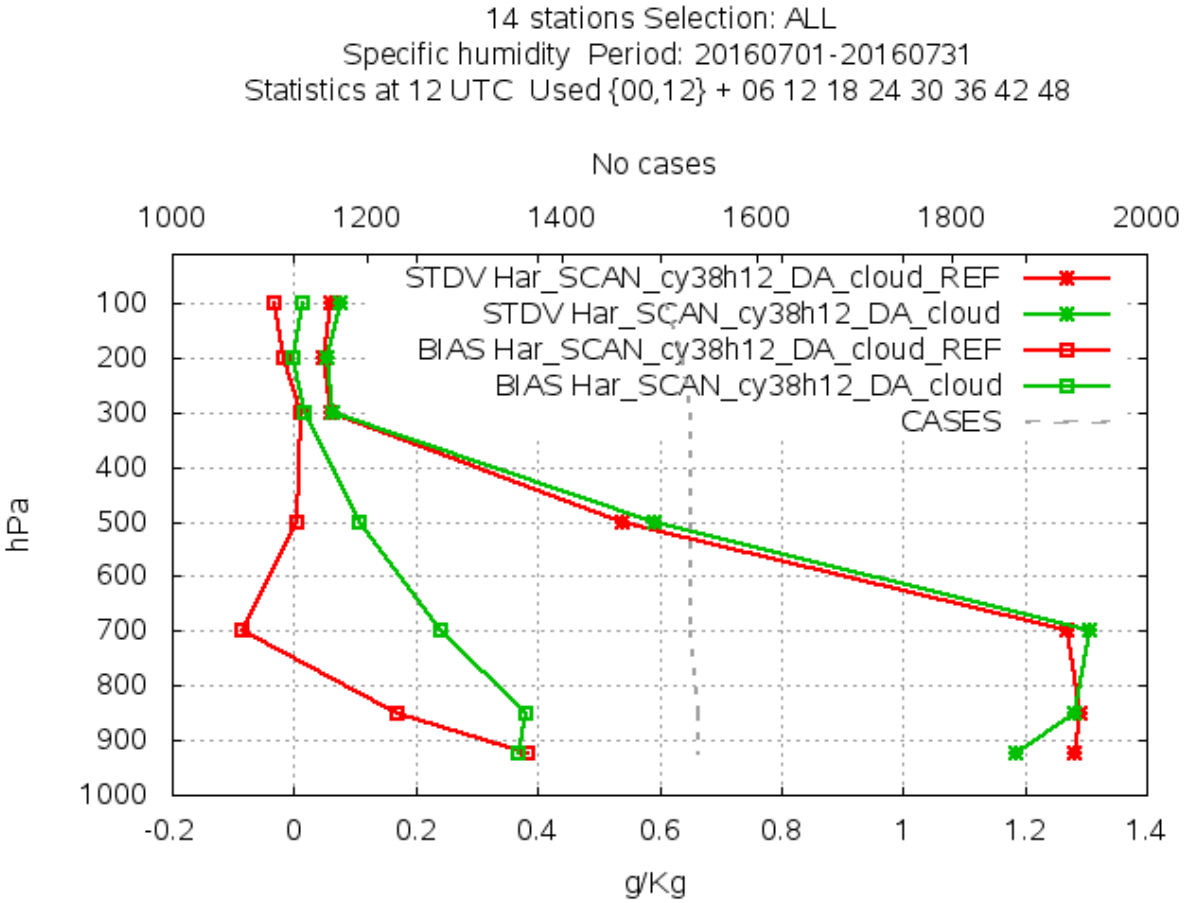


Figure 3b. WebGraf verification of upper-air specific humidity for 12Z run, where Ref Exp is in red color and MSG Exp green color.

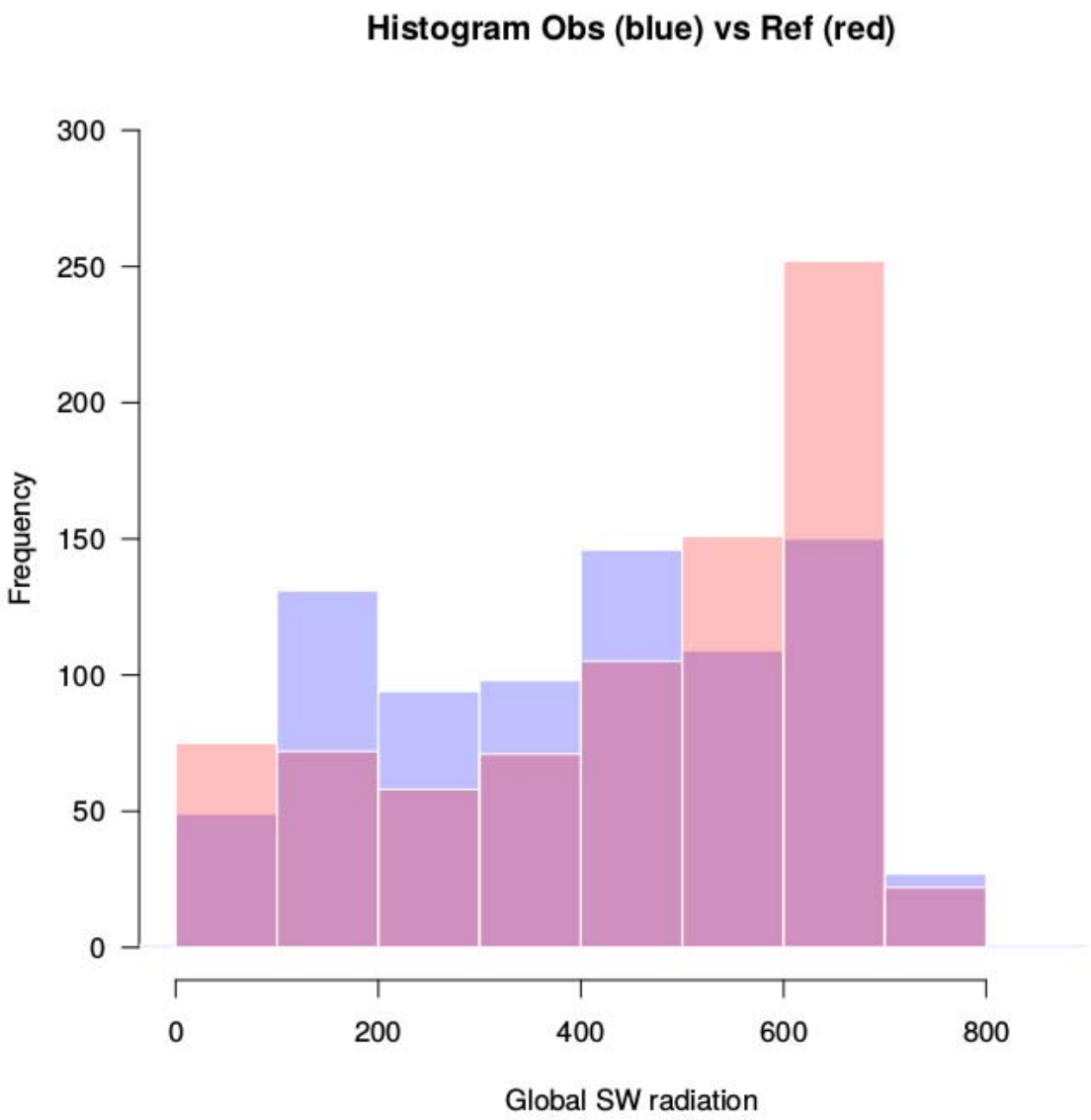


Figure 4. Histogram of Global SW radiation, Ref Exp (red color) vs Observations (blue color). Note: the dark-red color indicate overlapped areas.

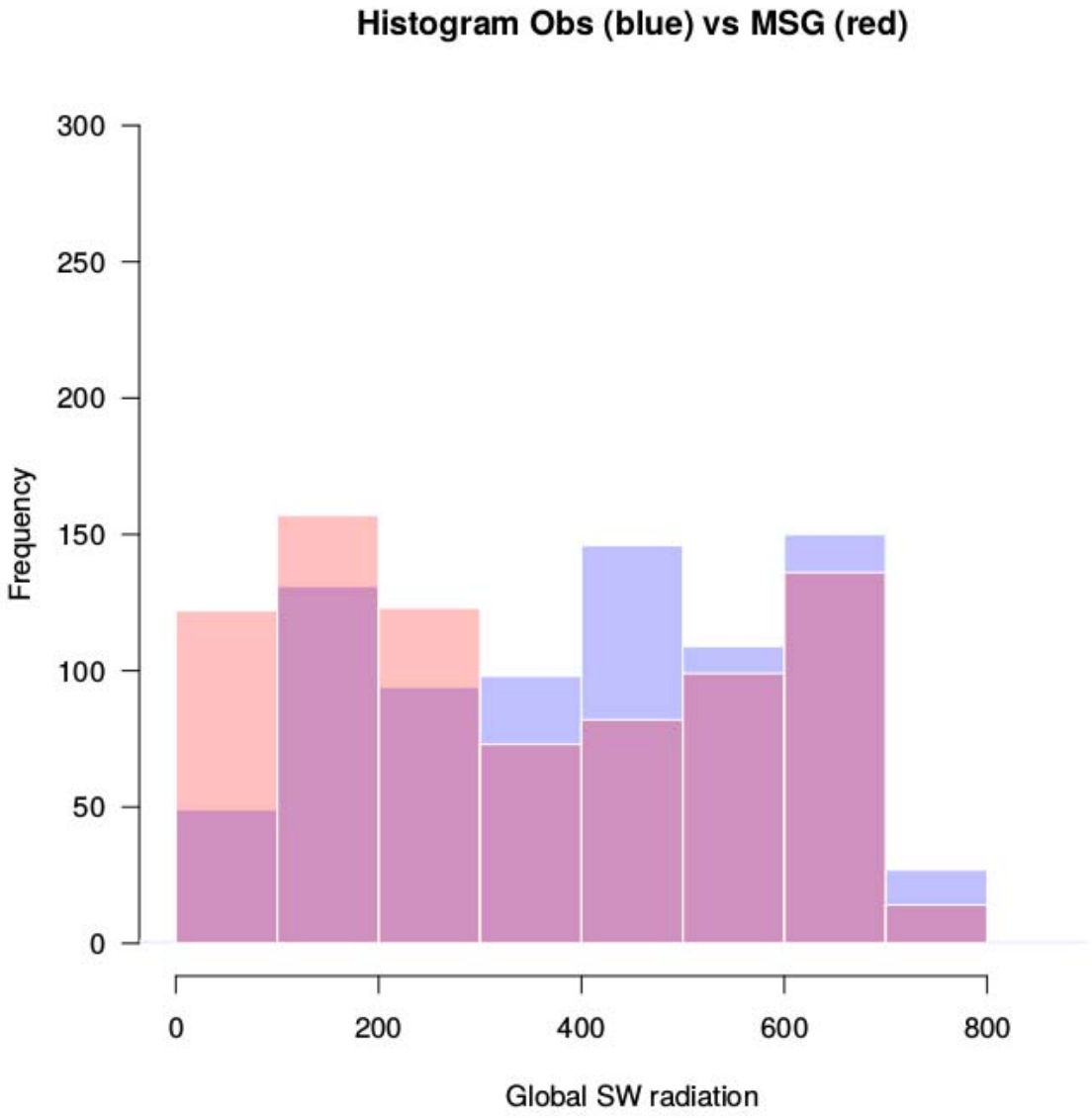


Figure 5. Histogram of Global SW radiation, MSG Exp (red color) vs Observations (blue color). Note: the dark-red color indicate overlapped areas.

4 References

van der Veen, S. H., 2013: Improving NWP model cloud forecasts using Meteosat Second-Generation imagery. Mon. Wea. Rev., 141, 1545–1557, doi:10.1175/MWR-D-12-00021.1.

Recent development of cloud microphysics (ICE3) within MetCoOp

Karl-Ivar Ivarsson, Swedish meteorological and hydrological institute

1 Introduction

Harmonie with AROME physics is used operationally in MetCoOp, which is the name of the collaboration in operational NWP-production between Norway,Sweden and Finland. Currently, an ensemble system with ten AROME members is used. Only the lateral boundaries are perturbed, using the SLAF technique. But there are plans to include other perturbations such as the cloud microphysics. This paper gives a short summary the MetCoOp work with cloud microphysics.

In section 2 a way of improving the forecasts of supercooled rain is presented and in section 3 the ongoing work of improving the microphysics and to get a better consistency between microphysics and radiation are described. Finally, in section 4, new tuning parameters, introduced in cycle 40h1.2 are presented. Some of them may be used for perturbing cloud microphysics.

2 Improvement of forecasts of super-cooled rain

The problem

Supercooled rain (rain drops with a temperature below freezing point) has generally been forecast less frequent than what is observed. Since most supercooled rain is formed when rain enters a layer of air with temperatures below freezing, the model will not be able to predict the event of supercooled rain if the warm layer layer of air above is not predicted (then it will be snow instead) or if the cold layer near the ground is not predicted. Those errors can not be treated by modifying the cloud microphysics. But one important reason for lack of forecasts of supercooled rain is that the rain refreezes too quickly.

Solution

The refreezing is reduced by the following changes:

- No 'raindrop accretion on the small size aggregates' which turns raindrops into snowflakes (5.2.4 in rain_ice.F90).
- Activate the process of 'raindrop accretion-conversion of the large sized aggregates into graupels' only if the mixing ratio of snow exceeds 1.0 E-5 (5.2.6 in rain_ice.F90). Using thresholds like this is proposed by Rutledge and and Hobbs (1984) and Thompson et al (2004).
- Activate 'rain contact freezing' , (6.1 in rain_ice.F90) only if the ice-number concentration is above 1.0E-8 m-3
- Activate the 'wet and dry' cases (6.3 in rain_ice.F90) only when the mixing ratio of graupel exceeds 3.0E-7.

A case of improved supercooled rain is seen in figure 1.

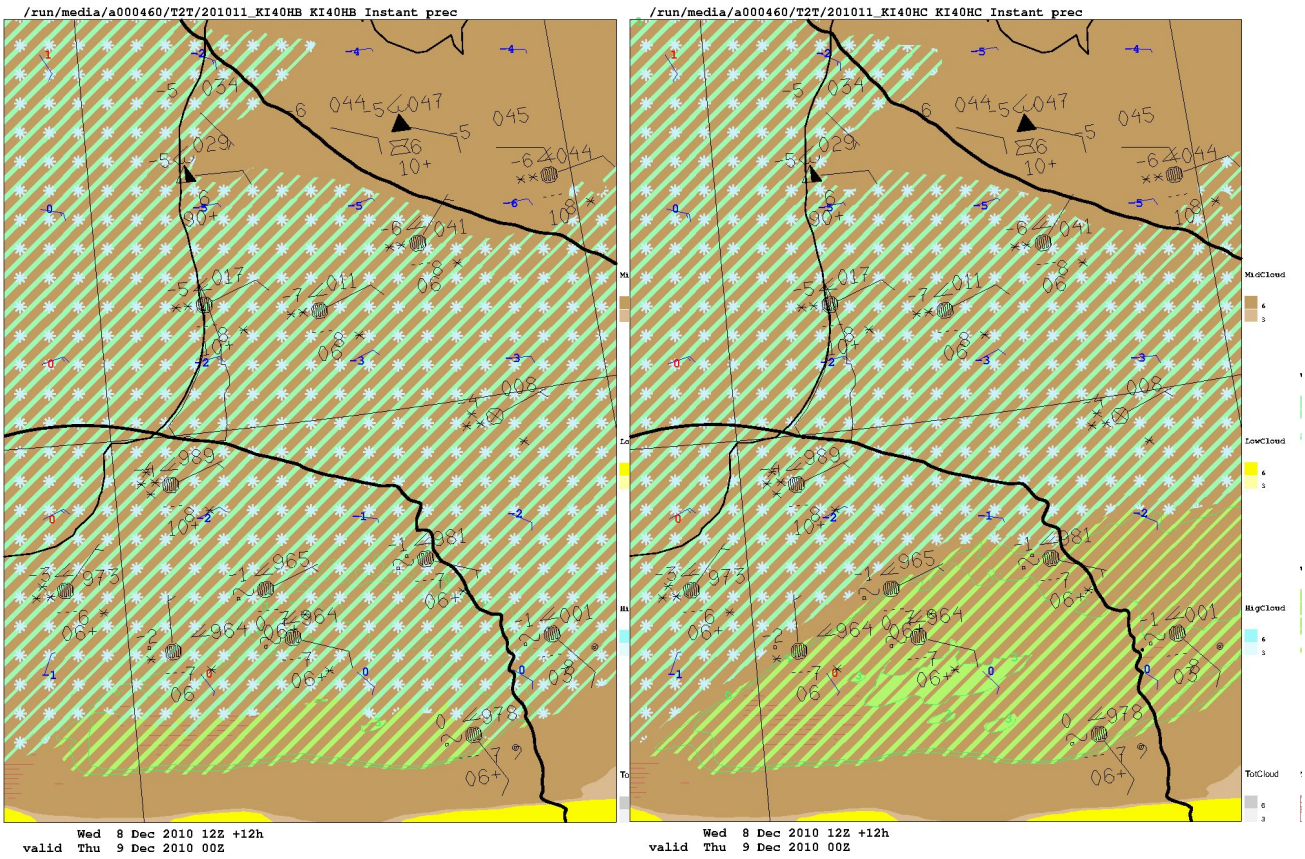


Figure 1: Forecast of rain (leaf green lines) and snow (bright green lines with stars) Observations in black. A '~' indicates observed super-cooled rain. To the left., the forecast without the modification and with the modification to the right. The plotted forecast area is south-east of the Baltic sea and the time is 2010-12-09 00 UTC. Forecast lenght is 12 hours.

Coalescence of super-cooled cloud droplets

This process may also result in supercooled rain, or more commonly in supercooled drizzle. Here, no layer with temperatures above freezing is needed. This process can only be described properly with a detailed knowledge of the ice-number concentration, but currently this concentration is only parameterized. However, there are some events when this process seems to be the most likely reason for the observed supercooled rain and also being better predicted with the modifications discussed here. But more studies on this issue are needed.

3 Ongoing work with cloud microphysics and its coupling to radiation

Consistency between microphysics and radiation

There are some different assumptions about the physical properties between the radiation schemes and the microphysics scheme. Two examples are:

- The prescribed cloud nucleus concentration (CCN) is different between the radiation scheme (900 cm-3 over land and 50 cm-3 over sea) and the cloud microphysics (300 cm-3 over land, 100 cm-3 over sea and 500 cm-3 for town)
- The cloud cover calculation in microphysics is based both on the content of water species and the subgrid-scale fractions of those species. What a radiation scheme needs is basically the subgrid-scale fractions for water species, mixing ratio, the number concentration for each water spicy and a optical depth calculation based on mixing ratio and the size distribution. Currently the microphysics only provides the cloud cover, which is used as a subgrid-scale fraction for all water species used in the radiation scheme. The optical depth is then calculated in the radiations scheme and is generally based of some other number concentration and size distribution than in microphysics.

Some very preliminary test have been done by reducing those differences, but a lot of work remains.

Remove unnecessary differences between OCND2 = false/true for cloud microphysics

The main differences between using OCND2 false/true are:

- Only the amount of cloud liquid is calculated from the statistical cloud scheme with OCND2 instead of both water and ice.
- Using OCND2, the cloud cover is based on a different subgrid-scale fractions of water and ice and the mixing ratio of cloud condensate and solid precipitation, instead of one single cloudcover based on the statistical cloud scheme only. With OCND2, the subgrid-scale fraction of liquid is determined by the statistical cloud scheme.
- Large ice crystals are converted to snow with OCND2.
- The deposition/evaporation rates of snow and graupel are reduced by a factors based on experimentation with OCND2.
- The Bergeron-Findeisen process (convert supercooled liquid to ice) is only used with OCND2=false. With OCND2= true it is replaced by a deposition/evaporation of ice. Currently this is based on a mean ice crystal size instead of a size distribution.

The two last bullets must not necessary be that different, especially if one wants to increase the consistency between radiation and microphysics.
Instead of using a reduction of deposition/evaporation rates of snow and graupel, a different size distribution of snow could be used. This is currently investigated at Meteo-France. Tests with alternative size distributions have also by preformed within MeCoOp.
Instead of using a mean ice crystal size for the deposition/evaporation of ice, one may use the same distribution as for OCND2=false. But in order to avoid too fast deposition/evaporation rate for temperatures just below freezing point, it is necessary to ignore the ventilation factor. The large effect of the ventilation factor seems to be less physical due to the slow fall speed of pristine ice crystals.
The new deposition/evaporation of ice with OCND2 is still experimental, but can be tested by setting the flag LMODICEDEP to true. (cycle 40h1.2, see next section)

New tuning parameters available for cycle 40h1.2

The new tuning parameters are all in the list NAMPARAR. They are explained below:

- LKOGAN: True: Use Kogan autoconversion instead of Kessler. In Cy40h1.1 and earlier, Kogan autoconversion was default for OCND2 and Kessler for **not** OCND2. Now this switch can be used independently of the OCND2 switch.
- LMODICEDEP: Use new experimental deposition/evaporation of ice for OCND2.

- RFRMIN: The default values are RFRMIN(0:6) = 0. ,RFRMIN(7-9,11), = 1. and RFRMIN(10) = 10. With the default settings, nothing is altered .
- Supercooled rain modifications mentioned in section 2: Use RFRMIN(1) = 1.0E-5, RFRMIN(2) = 1.0E-8, RFRMIN(3) = 3.0E-7, RFRMIN(4) = 3.0E-7, and RFRMIN(7) = 0.
- Reduce graupel: Use RFRMIN(5) = 1.0E-7 and RFRMIN(6) = 0.15. This corresponds to LGRSN = true in cycle 40h.1.1.
- Decrease the melting speed of graupel: Use e.g. RFRMIN(8)=0.5. A slower melting rate is proposed by Norwegian duty forecasters and by Sander Tijm.
- Increase/decrease the ice-nucleus concentration: E.g. set RFRMIN(9)=10 (increase) or 0.1 (decrease). Only active for cloud temperatures below freezing. Higher values leads to more snow and less clouds. Low values have the opposite effect.
- Increase/decrease the Kogan auto-conversion of liquid into rain: E.g. set RFRMIN(10) to 100 (increase) or 1 (decrease). Only active if LKOGAN = true and for cloud temperatures above freezing. High values leads to somewhat more rain, perhaps also somewhat less fog. Low values leads to somewhat less rain and occasionally also more fog.
- Possible subgrid-scale calculation of Kogan autoconversion. Set RFRMIN(11) to e.g. 0.01. Some risk of unexpected or unwanted behaviour for values at/or very near zero. Values above one are not recommended. Only active if LKOGAN=true and for cloud temperatures above freezing. Low values increases precipitation if the grid-box cloud cover is less than unity.

4 References

Rutledge, S. A. and P.V. Hobbs 1984 : The mesoscale and microscale structure and organization of clouds and precipitation in midlatitude cyclones. XII : A diagnostic modelling study of precipitation development in narrow cold-frontal rainbands. *J. Atmos. Sci.,* **41** 2970-2972.

Thompson, G., R.M. Rasmussen, and K. Manning, 2004: Explicit forecasts of winter precipitation using an improved microphysics scheme. Part I: description and sensitivity analysis. *Mon. Wea. Rew.*., **132** , 519-542

HIRLAM and HARMONIE-AROME radiation comparisons

Laura Rontu, Kristian Pagh Nielsen, Emily Gleeson

1 Introduction

The advantage of broadband, over spectral, radiation schemes is that they can be called more frequently within a NWP model, without compromising on computational efficiency. In mesoscale models fast interactions between clouds and radiation and the surface and radiation can be of greater importance than accounting for the spectral details of clear-sky radiation; thus calling the routines more frequently can be of greater benefit than the deterioration due to loss of spectral details. Fast but physically based radiation parametrizations are expected to be valuable for high-resolution ensemble forecasting, because as well as the speed of their execution, they may provide realistic physical perturbations.

We summarize validation of FMI operational HIRLAM NWP model results against Sodankylä and Jokioinen surface radiation measurements during 2006-2016. With this validation we wanted to learn

- How well does the HIRLAM radiation scheme HLRADIA, available for testing also in HARMONIE-AROME, behave in an operational NWP system?
- How to use radiation observations for NWP validation - can we evaluate model performance and detect changes?
- How to treat uncertainties in validation due to uncertainties of both the models and the observations?

We also show a HARMONIE example of how the radiation call frequency may influence results in a convective case.

2 Results

2.1 Ten-year validation of surface radiation fluxes at two stations

Finnish Meteorological Institute HIRLAM operational +3h and +6h forecasts were validated against Jokioinen (WMO station 02963, latitude 60.814°N, longitude 23.498°E, elevation 103 m.a.s.l.) and Sodankylä (02836, 67.362°N, 26.638°E, 179 m.a.s.l.) 3-hourly radiation observations, averaged from hourly means, between the 1st of April, 2006 and the 31st of March, 2016. During this period, practically unmodified version of the HLRADIA broadband radiation scheme was applied as documented by Rontu et al. (2017). The FMI operational HIRLAM model updates included introduction of the "newsnow" surface parametrizations and some minor changes, see Table 1 in Eerola (2013).

During 2006-2016 HIRLAM performed generally well compared to surface radiation measurements at two Finnish meteorological stations. Small systematic underestimation the SW absorption and overestimation of the LW absorption by the atmosphere was found. The SWDS bias of max. 20 Wm^{-2} may be due to the assumed

inhomogeneity correction of 20% of the cloud condensate content. The LWDS bias of max. 5 Wm^{-2} could be avoided by modifying an extra correction term of ca. +15 due to an assumed effect of "other greenhouse gases".

Radiation fluxes showed large variability due to cloud variations. Classifying the model and observation data according to cloudiness contains large uncertainties, especially for solar radiation when the solar elevation is low.

The reflected SW radiation (and consequently, albedo) shown by the model grid-average values and observed locally are not comparable due to the representativity error, especially over the snow-covered terrain. Upwelling LW flux, from which the grid-average surface temperature T_{surf} can be derived, suffers less of this problem, thus LWUP observations might be used, to some extent, to measure the performance of model's T_{surf}.

2.2 Frequency of the radiation call

The sensitivity of the HARMONIE-AROME results to the calling frequency of the IFS radiation scheme is illustrated by the results of experiments that were run for a domain over China around Shanghai for a convective case at the 30th of July 2010. Figure 1 shows the difference in 1-hour global SW flux and downwelling LW flux when the IFS radiation scheme was called every time-step (EXP01) compared to the call every 15th time-step (EXP15), which is the default. Note that negative differences EXP01-EXP15 indicate a positive bias of the default intermittent experiment with respect to the experiment with full time-resolution and vice versa. Differences in cloud cover and precipitation were also found (not shown).

A day with few clouds was chosen for this example, which is taken from experiments done during the MarcoPolo FP7 project (Nielsen et al., 2017). In the left-hand panel of Figure 1 the differences in the SW irradiances are shown. Overall these are $<<$ 1 Wm^{-2} and in average -0.2 Wm^{-2}. The larger differences are due to shifts in the cloud positions, which illustrates the high sensitivity of clouds to changes in the physics computations. In the right-hand panel of Figure 1 the differences in the LW irradiances are shown. Here significant overall differences are seen over land in the clear-sky parts of the domain. These differences go up to 13.2 Wm^{-2}. The explanation for this negative bias is that the integrated water vapour over land is increasing during the morning hours. Thus, more heat is gradually trapped. The intermittent setup fails to account for this effect.

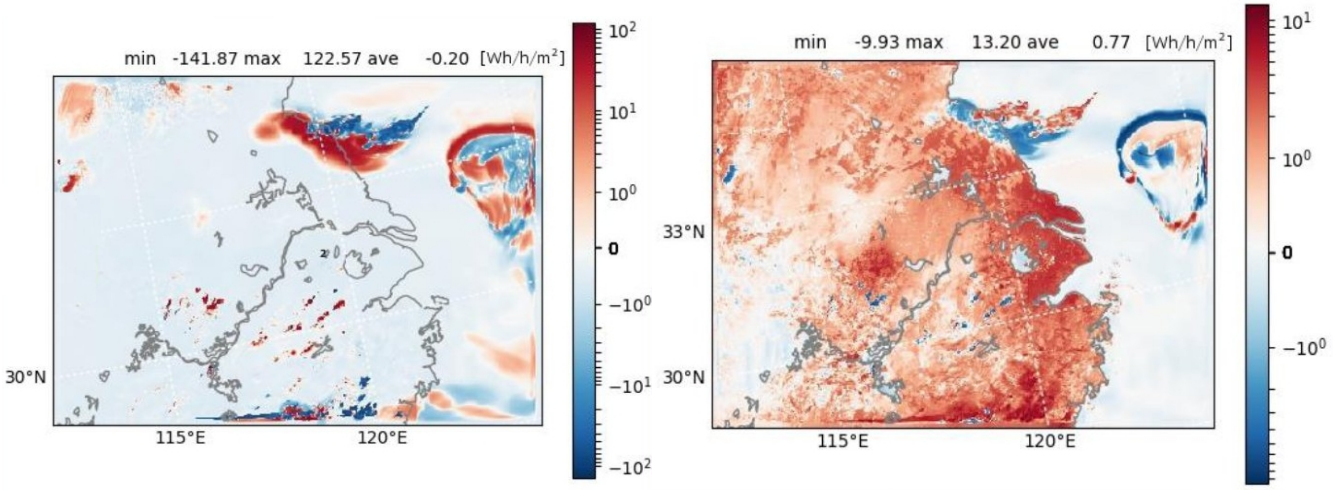


Figure 1: Difference EXP01-EXP15 in average irradiances: SW (left), LW (right), unit Wm^{-2}. The domain shown covers the Yangtze River Delta at the coast of China. The time interval is 1 hour from 0 to 1 UTC (8-9 AM local time) on the 30th of July 2010.

This example shows that the HARMONIE-AROME results can be quite sensitive to the calling frequency of radiation parametrizations. Further, more systematic studies are needed to understand the significance of

such differences for weather forecasts and to validate the results against observed radiation fluxes and standard meteorological observations.

Acknowledgements

We acknowledge the funding from the EU FP7 MarcoPolo project for the Yangtze River Delta simultation results shown here. We also acknowledge our overall funding from the HIRLAM-C Programme.

References

Eerola, K., 2013. Twenty-one years of verification from the HIRLAM NWP system. *Wea. Forecasting* **28**, 270–285. DOI:10.1175/WAF-D-12-00068.1

Nielsen, K. P., A. K. Georgoulias, K. Kourtidis and S. Stathopoulos, 2017. Relationship between air pollution and meteorology, Deliverable D 5.4 from the EU FP7 project: "Monitoring and Assessment of Regional air quality in China using space Observations (MarcoPolo)," accepted for publication, 2017. Available as https://www.researchgate.net/profile/Kristian_Nielsen/project/2014-2017-FP7-MarcoPolo-Monitoring-and-Assessment-of-Regional-air-quality-in-China-using-space-Observations-Project-Of-Long-term-sino-european-co-Operation/attachment/593feb4c1042bfac89199719/AS:504783733616640@1497361228470/download/D5.4_Relationship_between_air_pollution_and_meteorology.pdf?context=projectUpdatesLog

Rontu L., E. Gleeson, P. Räisänen, K. P. Nielsen, H. Savijärvi, and B. H. Sass, 2017. The HIRLAM fast radiation scheme for mesoscale numerical weather prediction models. Adv. Sci. Res., 1, 1-21. https://doi.org/10.5194/asr-1-1-2017.

High-resolution operational NWP for forecasting meteotsunamis

M. Tudor, J. Šepić (IOF), I. Vilibić (IOF), I. Janeković (IRB), S. Ivatek-Šahdan, A. Stanešić

1 Introduction

Meteorological tsunamis [Monserrat et al.(2006), Vilibić et al.(2016)] are long-ocean waves generated by intense small-scale air pressure disturbances. The waves can be several metres high and cause substantial damage to coastal towns. The main objective of the MESSI project (Meteotsunamis, destructive long ocean waves in the tsunami frequency band: from observations and simulations towards a warning system) is to build a reliable prototype of a meteotsunami warning system.

A meteotsunami or meteorological tsunami is a tsunami-like wave of meteorological origin.

- 10% of tsunamis worldwide have unknown origin [Vilibić and Šepić(2017)]
- 3% already assigned to meteorological conditions such as: atmospheric gravity waves, pressure jumps, frontal passages, squalls ... [Šepić et al.(2016)]
- meteotsunamis hit coastlines around the world and have many local names: rissaga (Catalan), ressaca (Portuguese), milghuba (Maltese), marrobbio (Italian), abiki (Japanese), šćiga (Croatian)
- previous known events: the highest meteotsunami recorded so far was in Vela Luka (1978, 6m) in Croatia which caused considerable damage but no people died, while there were other events that did cause human cost, such as in Chichago (1954,3m), Nagasaki (1979,5m), Ciutadella (2006,4m), Daytona Beach (1992,3.5m) and there were events recorded in Australia, New Zealand, UK, France, and even Finland!
- High waves destroy coastlines, strong currents endanger marine traffic in passages and channels that lead to harbours, while sea trafic is also endangered due to reduced sea depth during low tide.
- These events are dangeorus, especially in areas where the tide amplitude is low. In Adriatic the tide amplitude is 0.5m so consequently the towns lay very low above the medium sea level (e.g. Vela Luka, Stari Grad, Mali Lošinj). Here are several links to videos:
- https://www.youtube.com/watch?v=y-QlJO0ChwA
- https://www.youtube.com/watch?v=lzA5DTk_vIg

Meteorological tsunamis result from several resonance mechanisms that amplify the wave at the ocean surface for several orders of magnitude. But the wave has to be generated at the sea surface by the propagating pressure disturbance. Atmospheric gravity waves will propagate in a stable layer (Richardson number $Ri < 0.25$) from surface up to about 500hPa if the layer above is unstable ($Ri > 0.25$). At the level about 500hPa there is a jet that can exceed 40m/s. Strong gradients there support the generation of atmospherc gravity waves. These waves would propagate upward if there was no unstable layer at 500 hPa and the wave energy would simply propagate vertically. If the unstable layer contains a a critical (steering) level in which the wind speed equals the propagation speed of ducted waves, then the waves become trapped in a stable atmospheric layer adjacent to the ground. If there was no critical level, the waves would be absorbed.

When an air pressure disturbance of several hPa in amplitude propagates above the sea surface at the speed of long ocean waves $c = \sqrt{gH}$ where g is gravity acceleration and H is ocean depth, the long ocean wave

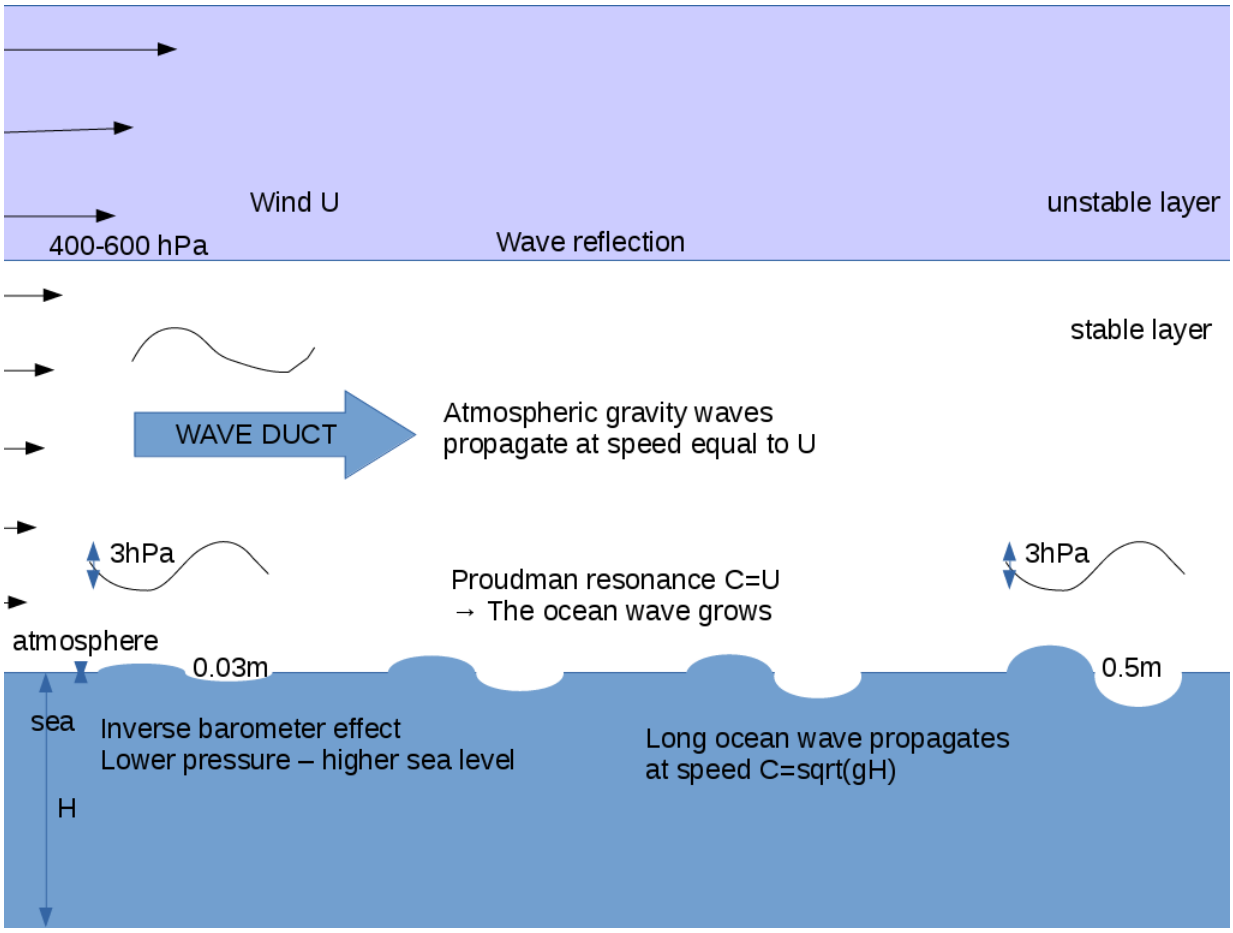


Figure 1: Illustration of the meteorological part in the meteotsunami generation processes. Atmospheric gravity waves are trapped below the unstable layer and propagate with speed U (equal to wind speed of unstable layer) as a duct wave. The wave in surface pressure generates a wave at the ocean surface due to inverse barometer effect (lower pressure - higher sea surface, higher pressure - lower sea surface). As the atmospheric wave moves (with the speed U), so does the wave at the ocean surface (with a speed of long ocean waves $C = \sqrt{gH}$). If $c = U$ the wave grows due to Proudman resonance.

amplifies due to Proudman resonance. Therefore, the meteorological model should predict the intensity, speed and direction of a fast and intensive pressure disturbance. The wave height at the open sea is on the order of centimeters but grows due to Proudman resonance (Figure 1). The ocean wave later amplifies due to shoaling when the wave slows down but the amplitude increases as the sea depth decreases (c in the above formula decreases with H). This increases the wave height up to one meter. Finaly, some harbours are particulatly vurnerable and amplify the wave to several meters. Incoming ocean waves can be amplified more than 100 times before hitting the coast as a destructive meteotsunami.

2 Meteorological conditions and model technicalities

Synoptic setting:

- Inflow of warm air from Africa 850 hPa
- SW jet > 20 m/s at 500 hPa

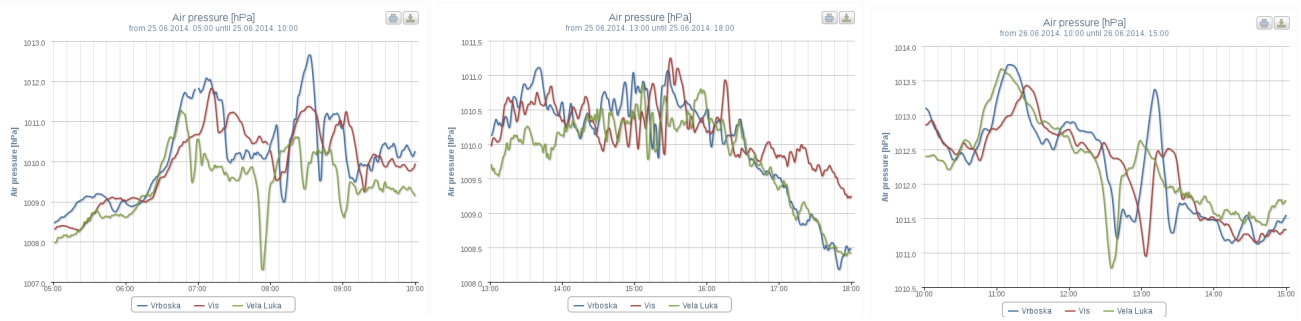


Figure 2: Air pressure measured on stations Vrboska (blue, Hvar island), Vis (red) and Vela Luka (green) with one second data interval during a widespread meteotsunami event on 25-26 June 2014, maintained by IOF .

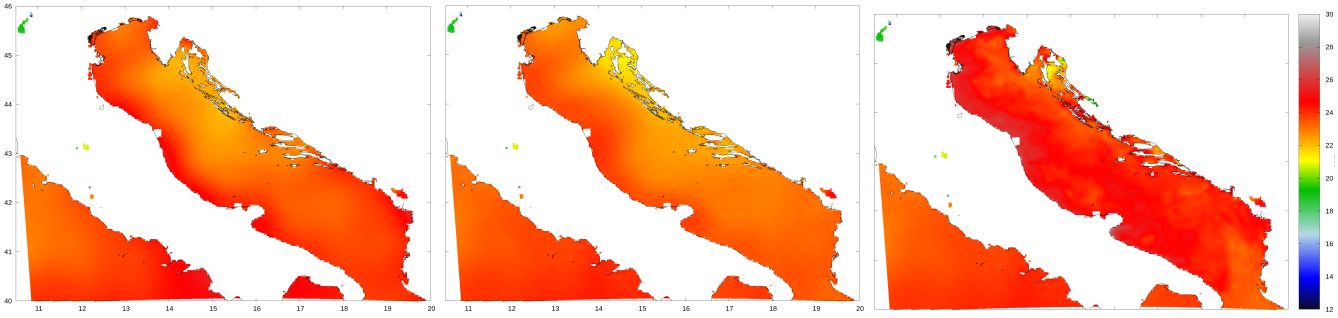


Figure 3: The SST in the operational forecast (left), when using SST from OSTIA (middle) and ROMS (right). SST influences the stability of the lower portion of the troposphere and the possibility of generating and propagating pressure disturbances.

- Unstable layer (Ri<0.25) 400-600 hPa
- High resolution: Forecasting a pressure change of more than 1hPa/1min
- Model output every minute is needed in order to detect the rapidly moving pressure wave (Figure 2).

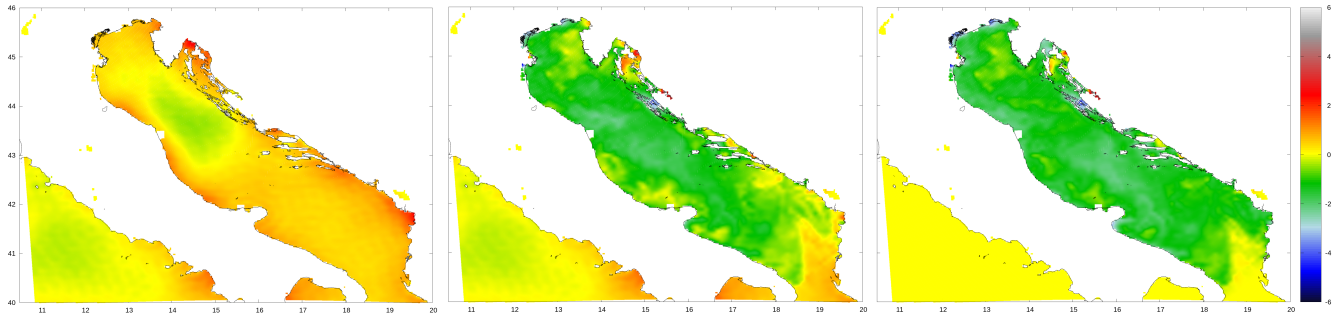


Figure 4: The differences between SSTs: in the operational forecast minus OSTIA (left), OPER-ROMS (middle) and OSTIA-ROMS (right). In this case, ROMS is warmest above open sea and consequently lower layers of troposphere are less stable.

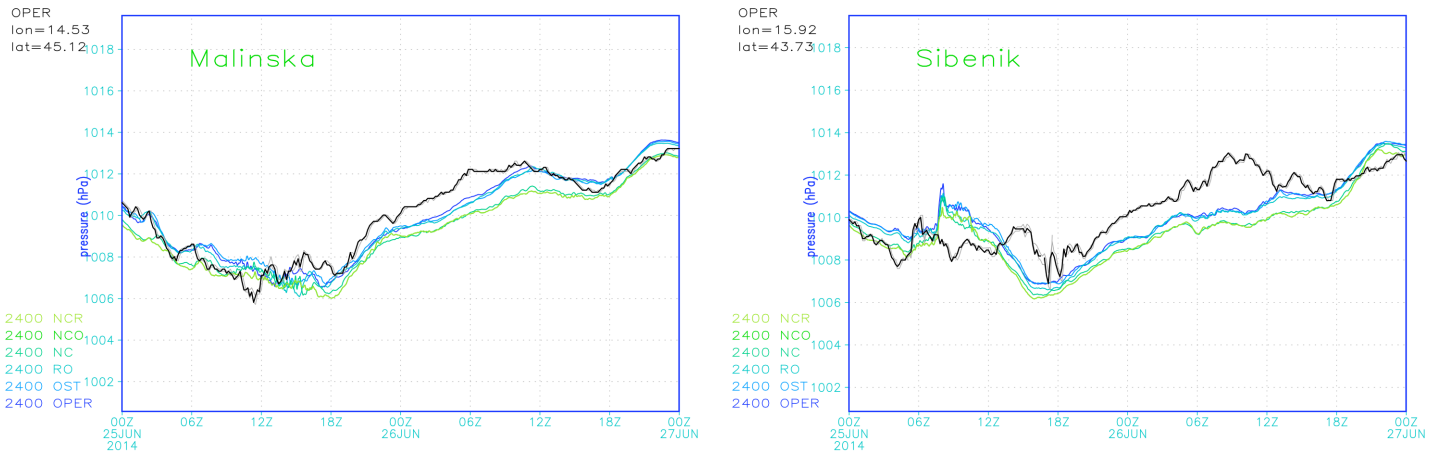


Figure 5: Figures show measured pressure (black line, 10 min interval) and output every time step (1 min) from operational run and experiments using SST from OSTIA and ROMS, new surface representation alone and in combination with ROMS. Both SST and roughness of the surrounding land surfaces influence the development, intensity and location of high frequency oscillations in pressure. OPER old topography and z0 IFS SST, OST using OSTIA SST, RO using ROMS SST, NC new topography and z0, NCO new topo + OSTIA SST, NCR new topo + ROMS SST.

- The atmospheric model should predict the pressure disturbance moving in the right direction (for Adriatic, the direction of SW jet at cca 500hPa) and at the right speed (speed of SW jet) and at the right position in space and time (if this is to be useful further).

Atmospheric numerical weather prediction models represent one of the main components of any meteotsunami warning system. The non-hydrostatic 2km resolution ALADIN forecast is running operationally in Meteorological and Hydrological Service of Croatia since July 2011. The suite predicts propagating small-scale pressure disturbances capable to excite meteotsunamis. However, the comparison of forecast pressure evolution to the measured data shows that the intensity of the observed pressure disturbances is simulated fairly by the model, but at a slightly different position and time, and propagate with slightly different speed and direction. Meteotsunamis are known to be highly sensitive to these parameters.

One-minute model time-step is used for reproducing the disturbances. This allows for an accurate estimate of the error in the position, shape, variability in space and time, speed and direction of the model disturbances with respect to those known to have generated meteotsunamis. We have further tried to improve the operational forecast by using of more realistic SST, e.g. coming from the ROMS ocean model [Janeković et al.(2014)], and more realistic physiography of the terrain surrounding the Adriatic sea.

3 Experiments with different SSTs and z0 for one particular meteotsunami event

The recent meteotsunamis are investigated using available atmospheric data and meso-scale atmospheric model ALADIN with ALARO physics package. ALADIN model is used for reproduction of travelling air pressure disturbances during the Adriatic meteotsunami event.

Here we analyse a widespread meteotsunami event on 25-26 June 2014 [Šepić et al.(2016), Šepić et al.(2016)]. The sea surface temperature (SST) used in the model forecast arrives from the global model that is used for

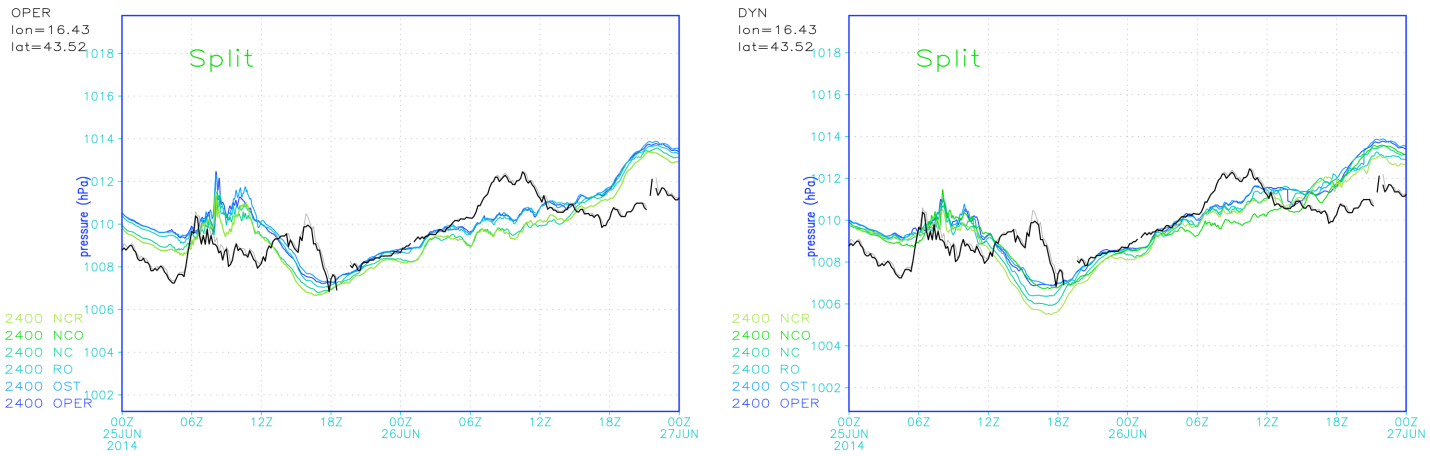


Figure 6: As Figure 5 but for Split (left) and using different dynamics set-up (right). OPER old topography and z0 IFS SST, OST using OSTIA SST, RO using ROMS SST, NC new topography and z0, NCO new topo + OSTIA SST, NCR new topo + ROMS SST.

lateral boundary conditions. It has been shown that model SST can be quite far from real values over the Adriatic, especially over the coastal areas, such as in the WAC and Kvarner bay (Figure 3. The use of more realistic SST, from OSTIA analysis and the ROMS ocean model influences the intensity and propagation of the pressure disturbance. Recently, it has been shown that the physiography fields used by the model are of too low resolution and contain errors in the Adriatic area. More realistic physiography of the terrain surrounding the Adriatic sea affects the triggering of the disturbance (Figure 5). Different dynamics set-up does affect the amplitude of the pressure wave (Figure 6) as well as the evolution of larger scale pressure features.

4 Conclusions

If the large scale setting (synoptic scale) is forecasted by the large scale model, then the small scale LAM generates atmospheric gravity waves. The wave trapping of the propagating wave depend on the existence of unstable layer at about 500 hPa and a stable layer below and a critical level in the unstable layer. The local manifestation of the pressure wave is sensitive to many factors. SST influences the stability of the lower portion of the troposphere and the possibility of generating and propagating pressure disturbances. In this case, ROMS is warmest above open sea and consequently lower layers of troposphere are less stable.

5 References

Janeković, I., Mihanović, H., Vilibić, I., and Tudor, M.: Extreme cooling and dense water formation estimates in open and coastal regions of the Adriatic Sea during the winter of 2012, *J. Geophys. Res. Oceans*, 119, 3200–3218, doi:10.1002/2014JC009865, 2014.

Monserrat, S., Vilibić, I. Rabinovich, A. B. Meteotsunamis: atmospherically induced destructive ocean waves in the tsunami frequency band. *Nat. Hazards Earth Syst. Sci.* 6, 1035âĂŞ1051, 2006.

Šepić, J., Vilibić, I., Monserrat, S., 2016. Quantifying the probability of meteotsunami occurrence from synoptic atmospheric patterns. *Geophysical Research Letters*, doi: 10.1002/2016GL070754

Šepić, J., Međugorac, I., Janeković, I., Dunić, N., Vilibić, I., 2016. Multi-meteotsunami event in the Adriatic Sea generated by atmospheric disturbances of 25-26 June 2014. *Pure and Applied Geophysics*, doi: 10.1007/s00024-016-1249-4

Šepić, J., , I. Vilibić, A.B. Rabinovich and S. Monserrat, 2016: Widespread tsunami-like waves of 23-27 June in the Mediterranean and Black Seas generated by high-altitude atmospheric forcing. *Sci. Rep.* | DOI: 10.1038/srep11682

Vilibić, I., Šepić, J., 2017. Global mapping of nonseismic sea level oscillations at tsunami timescales. *Scientific Reports*, 40818, doi:10.1038/srep40818

Vilibić, I., Šepić, J., Rabinovich, A. B., Monserrat, S., 2016. Modern Approaches in Meteotsunami Research and Early Warning. *Frontiers in Marine Sciences*, http://dx.doi.org/10.3389/fmars.2016.00057

AROME-NWC : AROME-France for Nowcasting

Nicolas Merlet, Philippe Cau, Céline Jauffret (MF, Nowcasting Department)

1 Introduction

Most traditional nowcasting tools are based on extrapolation techniques of observations, using radar or satellite. Those techniques are currently used to forecast smallest scales phenomenon's location for a short range into the future until it become unpredictable. Extrapolation techniques can't create non-observed systems nor change their intensity or motion and have difficulties to take into account the orography. Therefore their predictions are less reliable when exceeding 1 hour.
NWP models now start entering the nowcasting domain. Indeed, new non-hydrostatic and high resolution models are now able to simulate short time and spatial scales. Recent works on spin-up and data assimilation make them relevant for nowcasting issues. Moreover, the increasing capacity of the computing centres allows a real-time operation of these models.

AROME-NWC nowcasting NWP model has been in operation since March 2016. It has been designed for the forecasters issues and also as way of improving existing nowcasting products.
This document is an overview of AROME-NWC after one year in operation. It also gives the main lines of tested evolutions and planed uses of this new model.

2 An overview of AROME-NWC 2017 :

AROME-NWC: "AROME for nowcasting"

AROME-NWC is built around a configuration of the existing mesoscale-scale and limited area model AROME-FR. Both models share the same characteristics such as domain, physics and dynamics, 3DVar data assimilation system, spatial scale (1.3km), ARPEGE coupling model…

Focus on AROME-NWC (cy41t1) :
24 runs (hourly atmospheric analysis)
No cycles : guess from AROME
Surface initialisation: from a SURFEX AROME-FR forecast
Maximum forecast range: +6h
output : every 15min
assimilation window: [-10min; +10min[; cut-off :10min (versus 1h30 for AROME-France)
output availability: from H+30min for the 6h forecast

However, nowcasting issues drive new contradictory challenges : in one hand, forecast fields must be frequently refreshed, to take into account the latest observations, on the other hand the outputs must be delivered as soon as possible (within 30 minutes after the latest observations).
These constraints lead to a compromise between the observation update and computational time. Thus, the observation time window of AROME-NWC is narrower, hence assimilates fewer observations, than AROME-FR's one.

AROME NWC is mainly designed for surface condition forecasting (rainfall, snow, fog, gusts, humidity and cloudiness). Its main characteristics are

1. Dense forecast of several parameters (wind, temperature, humidity, but also reflectivity, precipitation, kind of hydro-meteors..)
2. High frequency of forecast (hourly refreshed)
3. High spatial and temporal resolution: 1.3 km mesh and for a given forecast, forecast fields are produced every 15 minutes
4. Maximum forecast range of 6 hours

These forecasts are available within 30 minutes after the latest observations.

A non cycled model

AROME-NWC is performed every hour. Its 3D-var assimilation system starts with an analysis from the last available AROME-FR forecast valid at analysis time (*the guess*) and observations from 10min before to 10min after analysis time. As AROME-FR is not hourly refreshed, AROME-NWC runs are based on forecast of different ages (see fig.1).

This configuration is based on technical and efficiency reasons. Indeed, due to its narrow assimilation window, AROME-NWC 2017 mainly assimilates surface informations from ground or radar network. So biases or imbalances might develop especially in upper-level fields when run in cycled-mode. Moreover, it's easier to run an AROME-NWC hourly even if the previous forecast did not correctly run.

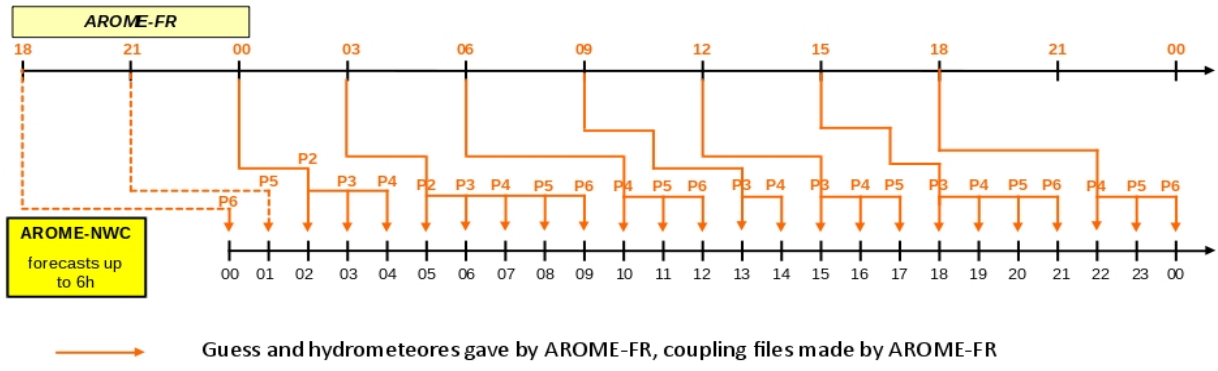


fig. 1: Dependence of AROME-NWC to AROME-FR

An assessment of AROME-Nowcasting's forecasts vs AROME-FR forecasts available at the same time confirms the contribution of an adjustment every hour with the last available observations up to 2-3 hours range although its assimilates less observations (*Auger et al. 2015)*). More recent scores of the current operational versions of these two models show similar conclusions .

Which output fields for AROME-NWC?

With an hourly production, to keep all the fields would represent too much data to store.

The following analysed and predicted data are available:

- hourly 10m wind and wind gusts
- 2m temperature and humidity
- Θ'_w at 950 and 850 hPa
- mean sea level pressure
- low cloudiness
- maximum radar reflectivity
- 15 min cumulated rainfall (rain, snow, graupel)

Other fields are produced only for the calculation of some diagnostics like convection, fog, winter weather surface phenomena and are not stored.

Finally, 838 new AROME-NWC fields are produced each hour. This very high rate of production makes a systematic use of the outputs difficult. Therefore a dashboard helps the forecasters: for a selection of parameters, it shows different colours corresponding to different levels of warning and helps to look at the forecasts only when useful. For a given date, several forecasts started from different initial dates are available. Then the forecaster has the possibility to look at different solutions

given by the model for this given date, which can be seen as a "poor man ensemble forecast". This site was tailored to meet forecasters' needs and expectations during the 2015 experiments.

3 What results after one year?

As for other operational models, AROME-NWC's skill and accuracy are analyzed through scores and case studies .

2016 scores (March-December) (source: MF internal Monitoring Report 118-120)

Forecast fields such as precipitation, mean wind speed, 10m gust, 2m temperature are monitored. The idea is to check the added-value of AROME-NWC compared to AROME-FR which gave the background for the assimilation. We are thus in an operational context when comparing AROME-NWC to the available AROME-FR at the same time and thus based on a less recent analysis.

Precipitations: better scores than AROME-FR

For the hourly cumulative rainfall/precipitation (over 5mm/h), AROME-NWC reduces the overestimations made by AROME-FR in each trimester except in summer 2016. During this period, the AROME-NWC's morning runs tend to underestimate the precipitation while AROME-NWC's afternoon runs correctly rectify AROME-Fr's biases.

Since March, the false alarm rates are lower and the detection rates are slightly better with AROME-NWC. The false alarm rate is quite important in the morning while detection rates are low, leading to negative Heidke skill scores (HSS) but, nevertheless, better than those of AROME-FR. During the afternoon, there are fewer false alarm rates and detection rates are better, HSS are positive and again, better than those of AROME-FR.

Wind: better scores specially during the first hours

For 10 m gust over 60km/h, AROME-NWC performs much better than AROME-FR during the first two hours, reducing under-estimations but still not enough. For these forecasts, AROME-NWC improves detection rates..For other forecast ranges, both models perform equally.

As for precipitations, HSS for AROME-NWC gust wind are better than those for AROME-FR.

During its 9 first months of operation, the balance is positive for AROME-NWC forecasts when compared to AROME-FR forecasts that are available at the same time in an operational context.

Note : *For precipitations and gust winds, HSS are monitored against persistence initialized with available observations at the start of each AROME-NWC run. Persistence is a really good challenger in nowcasting because of its good scores for the first hours forecasts.*

A word about scores:
the success rate is the total number of good forecasts in relation to the total number of events.
HSS is equal to: success rate – reference success rate/1- reference success rate
HSS characterizes all good forecasts against a reference forecast (here the persistence).
It varies from -∞ to 1, where 1 is the perfect score and 0 is the reference forecast score

Subjective evaluations

Subjective evaluations are carried out on selected meteorological events. The following elements are based on the 2015-2016 experiment and on the control of some recurrent behaviours of models.

2015 test bed

The forecasting department carried out some tests in quasi-operational conditions during the 2015 autumn. Those tests involved more than 40 forecasters from the national and regional forecasting departments. 10 events were thus replayed and studied on different meteorological issues: convection, frontal rain, synoptic wind, fog maritime entry, gust and snow.

This session concluded that AROME-NWC can be useful for forecast ranges under 2 h (by giving details about the first stages of an event, for example). Nevertheless, it also brought to light some

defaults such as the variability of successive AROME-NWC runs and its tendency towards AROME-FR forecasts after 2 hours forecast.

Overview on the recurrent behaviour of operational AROME-NWC

Since it became operational, AROME-NWC is regularly monitored by the forecasters; they aim at pinpointing the erroneous and reproducible behaviours of the model. The latest published report deals with a dozen identified troublesome events over the period April-June 2016.
Most feedback are about difficulties encountered by AROME-NWC to readjust to observations (late or no readjustment). For extreme cases, AROME-NWC fails to improve AROME-FR or even worsen AROME-FR forecast.

4 Planned changes

AROME-NWC is a recent model which needs to be improved and calibrated. The first feedbacks show:

- Specific behaviours of AROME-FR transmitted to AROME-NWC
- Apart from the first few hours, AROME-NWC is found to be too close to AROME-FR
- The variability between AROME-NWC runs complicates its use
- In some cases, difficulties encountered by AROME-NWC to readjust to observations

In addition to AROME-FR improvements that will benefit AROME-NWC, other avenues are being explored.

Less weight given to observations?

At the present time, observations are given more importance in the assimilation of AROME-NWC than in that of AROME-FR. Hence, AROME-NWC analyses are closer to observations than those of AROME-FR. To a numerical model, an analysis close to observations is no warranty of good predictions. As a matter of fact, during the assimilation, the closest post-processing is to observations, the furthest it may depart from its equilibrium which affects the first predictions of the model. To reduce the nudging coefficient leads to a reduction of the imbalance in the model fields due to the analysis stage, and hence, leads to better forecasts. This modification is currently in the process of validation to be implemented in the next e-suite.

Extension to the past of the observation extraction window

The current AROME-NWC assimilation window takes into account almost only radar and surface observations (essentially in precipitating areas covered by the French network). To increase the number of observed data over the domain and in all weather, it was suggested that the current window should be set to [-20mn;+10mn[so that SEVERI satellite observations, valid at H-15min (those at H being not available with a 10min cut-off) could be assimilated. However, this brought no improvement and will not be implemented in the next e-suite.

Use of a forecast initialised by IAU (Incremental Analysis Update)

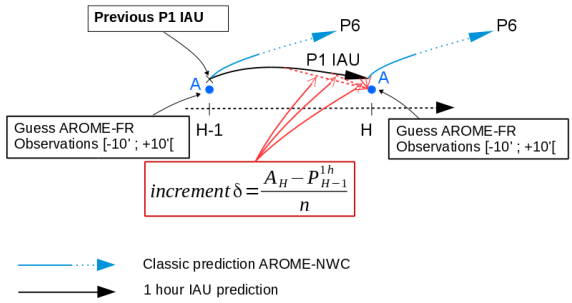
For AROME-NWC as for AROME-FR, hydro-meteor fields cannot be modified during the analysis phase as it takes some time to adjust to the new analysed fields (wind, temp, specific humidity and surface pressure). Therefore, reflectivity fields for the 15 or 30 minutes AROME-NWC forecasts may not be representative of the new state of the model.
The use of IAU, already implemented for AROME-FR, can shorten the adjustment time. IAU shall be used in a slightly different way in AROME-NWC (see boxed text “IAU at a glance”)

Towards a cycled IAU initialised forecast

AROME-NWC is not run in cycle and does not use its own predictions for future predictions for lack of assimilated observations (too short cut-off). Consequently, the AROME-NWC run does not directly benefit from the previous AROME-NWC run modifications.
Cycling IAU in AROME-NWC aims at mixing information from two consecutive cycles in an innovative way, so that a maximum observations can gain control over analyses. Cycling IAU should also reduce the inter-run variability. The first results are promising.

IAU at a glance

In AROME-FR, IAU is used to shift the first 1h prediction closer to the following run prediction by the addition by fraction of the (δ) analysis increment at each time-step.
For AROME-NWC, the idea is to take as starting point of the prediction, the result of a 1h prediction of the previous run, modified by IAU to get closer to the analysis of the run of the hour H.



5 Data fusion extrapolation / numerical predictions

At nowcasting scale, AROME-NWC data are added to the conventional ones retrieved from the extrapolation of observations.
Thus, the extrapolation of observations and observed situations get closer after the first hours of forecast. AROME-NWC fields can remedy known defects of the extrapolation of observations (relief areas, no occurrence or disappearance of cells…).
Blending these very different data is an important line of work for the nowcasting department. The aim is to take the best of each method to have the most relevant information without break during the]0-3h] forecasts.
Several approaches have been investigated. The first method, based on predicted and extrapolated matching cells, was in the end discarded as too complex. The method, developed since June 2016, rests on a so-called “sequential aggregation of predictors” method. This method aims to blend two predictors (in our case extrapolation and AROME-NWC) so as to get a compound close or better than the best of any of them, the end product being a weighted sum of numerical prediction fields and of extrapolations. The weights given to each predictor are adjusted in real time according to their recent behaviour in regard to observation.
A first merger version between extrapolation and numerical prediction of rainfall (time step 5’) is produced since December 2016. It will be tested and improved during 2017. A fusion of reflectivities will also be implemented in 2017.

6 References

Auger, L., Dupont, O., Hagelin, S., Brousseau, P. and Brovelli, P. (2015), AROME–NWC: a new nowcasting tool based on an operational mesoscale forecasting system. Q.J.R. Meteorol. Soc., 141: 1603–1611. doi:10.1002/qj.2463

Tuning the implementation of the radiation scheme ACRANEB2

Jacob Weismann Poulsen, Per Berg
IT Department, DMI, Copenhagen, Denmark
Email: jwp@dmi.dk, per@dmi.dk

This paper is available in the original and more compact IEEE format at:
http://www.dmi.dk/fileadmin/user_upload/Rapporter/TR/2017/SR17-22.pdf

Abstract

It is not trivial to write code that leads to efficient performance on modern hardware and it becomes even more involved if the performance has to be *portable* and *competitive* across different architectures. This paper describes the work that was done to improve the performance of the radiation dwarf pertaining to the ESCAPE[1] project embracing the well-known IFS and ALADIN-HIRLAM numerical weather prediction models. The overall idea is to demonstrate that the *implementation* of the radiation scheme known as ACRANEB2 can indeed be refactored so that it runs with competitive performance on modern throughput architectures such as the 2nd generation Intel® Xeon Phi™ processors (codenamed Knights Landing™) and accelerator architectures from NVIDIA. We show that the refactored codes run significantly faster on KNL and on NVIDIA P100 than they run on the strongest dual-socket Intel® Xeon system[2] available on the market today. In addition, the refactored code also runs significantly faster than the original code on all the dual-socket Intel Xeon systems used during this study. The parallelism itself is expressed using directive based approaches, OpenMP and OpenACC, respectively. We show that competitive performance is obtained by completely different code bases and hence that performance on a given target architecture comes from the source code within the scope of the directives rather than from the directives themselves. The performance results are presented as *time-to-solution* and *energy-to-solution* and to be fair focus is on comparing performance across hardware released in 2016. The results of the refactored implementations are also related to Moores law and cross-compared with the evolution of the de-facto standard processor benchmarks HPL and Stream. Finally, we show how one have to use phenomenological modeling in order to apply the roofline model in cases like this where transcendental functions are heavily used.

Keywords: Performance, SIMD, OpenMP, OpenACC, GPU, HPC, Exascale, NWP, ESCAPE, Xeon Phi, KNL, NVIDIA P100, roofline, Energy efficient computing.

1 Introduction

Radiation physics is one of the most time-consuming physics components in NWP today, cf. figure 1. It is an interesting component of the physics due to its intensity in floating point operations which is unusual in NWP models which tend to be bound primarily by memory bandwidth. There is a strong desire from a physics perspective to run radiation physics at every timestep of the model instead of only intermittently[3] as dictated by the computational demands of the current operational production, and this desire increases with increasing resolution but the cost of doing so is simply too high today. This is the overall motivation for choosing the radiation as the physics component in the ESCAPE project. There are several schemes available

[1] http://www.hpc-escape.eu

[2] Intel Xeon E5-2699v4.

[3] In current HARMONIE-AROME configuration the radiation physics is updated only every 12^{th} timestep corresponding to 15 minutes intervals in the model.

for radiation today and the scheme chosen for this study is currently used in production setups in the ALADIN[4] community and one that is planned for near-future setups locally where HARMONIE-AROME is used, cf. [1]. The baseline version of this dwarf consists of the upstream ACRANEB2 code extracted from the full IFS code base as a stand-alone application but with loop and index ordering interchanged compared to the upstream implementation. The SLOC of the baseline dwarf is around 6.000[5]. The original ACRANEB2 scheme is described in [9] and [6]. Moreover, the upstream data structures and loop nesting is described in the IFS documentation, cf. [5].

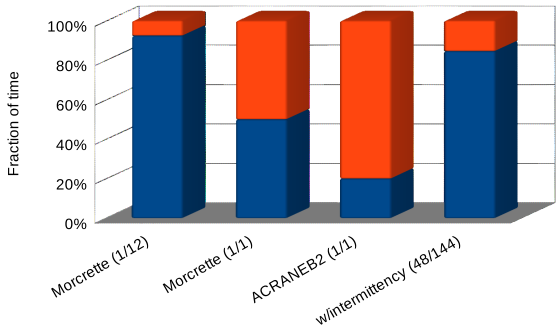


Figure 1: Fractional split of compute time spend in radiation (red) and in all remaining parts (blue) at a node count corresponding to a real production run and with settings corresponding to a real production run. The red area will increase as the number of nodes decreases and decrease as the number of nodes increases. The first bar represents the current state where the Morcrette radiation scheme (see [4]) is only called every 12^{th} timestep due to the computation resources required for each call. The second bar represents the split if the Morcrette radiation scheme was called at every timestep and clearly shows why this is not feasible. The third bar represents the split when the full ACRANEB2 scheme is run at every timestep and finally, the fourth bar represents the split in the ACRANEB2 scheme using the newly developed algorithm with intermittency that allows for a few expensive timesteps and several less expensive timesteps. The total time spend running the new ACRANEB2 with intermittency is more attractive than running the current operational algorithm at every timestep but still more expensive than can be afforded in production runs.

The baseline implementation covers multiple radiation options of different complexity, and further, the original ACRANEB2 algorithm has support for selective intermittency too. This means, that the upstream code includes segments that are more or less frequently visited during a forecast simulation and describes more or less advanced physics, cf. the fourth bar in figure 1. For the present study, however, we have chosen to dive into the implementation of the most computationally expensive part, i.e. we choose the most challenging path through the call tree as seen both from a radiation science and a computer science perspective. This corresponds to the third bar in figure 1.

Throughout this paper we use a 400x400x80 test case, i.e. with 400 points in both horizontal directions and 80 layers in the vertical. This corresponds to the largest test case we could run with the baseline code on a single 64 GB node. The target problem size for real operational, regional models now and in near future has up to about 3 times as many points in each horizontal direction and 65-80 layers; for example, the largest operational setup locally at our institute is 1200x1080x65 at present. The refactored code can easily run such cases (not shown in this paper) on a regular 64 GB node. Moreover, the problem as layed out in our approach is embarrassingly parallel hence scaling to more nodes is trivial and will not be considered in this paper.

The paper is organized as follows: First, we define our perception of performance and explain the performance improvement process in general terms in section 2. In section 3 we describe our initial refactoring and with a detailed presentation of the performance model used during the study placed in appendix 8.1. In section 4 we present the basic data structures and the parallelization of the code. Section 5 reveals the performance results

[4]http://www.umr-cnrm.fr/aladin/spip.php?article304
[5]as generated using David A. Wheeler's 'SLOCCount'

obtained on the different target architectures by codes specifically crafted towards performance on each target, and we also describe some further refactorization steps needed to bring the GPU on a similar performance level as the Xeon Phi. Finally, based on our work, we draw some conclusions in section 6 and suggest direction for future model development in section 7. Build and run specifications used throughout is placed in appendix 8.2 together with some system reference numbers.

2 Performance

It seems reasonable to define what we mean by *performance* and to specify how we can measure it. In this context, *performance* is *time-to-solution* $T2S$, i.e. the seconds it takes to complete a given task. That is, `INPUT` is fixed and `OUTPUT` is fixed by the algorithm itself and the freedom comes solely with the implementation of the algorithm in the source languages as well as in the target ISA and its runtime environment. Thus, we can not allow that results describing the physics are changed as a consequence of our changes in the implementation. Moreover, we will require that all results, both with respect to the physics and more technical measures like time-to-solution and energy-to-solution, are reproducible. Needless to say, we actually rate *correctness* and *reproducibility* higher than performance gain, and we strive towards securing these properties at all times. Of course, results might change numerically due to e.g. choices of different math libraries, use of SIMD reduction instead of scalar reduction, etc., but we always verify that we obtain identical results from one code release to the next, also across platforms, by performing "safe math" experiments. We also verify measurements like timings by repeating the experiment many times.

Thus, *performance tuning* is a process where we tweak the implementation and its build and run environment in ways that allows us to benefit most from the silicon provided by a given architecture vendor, keeping the results fixed. We illustrate this in figure 2 where we seek an implementation I within one of the two circles in the subset of the left hand side that with a given set of build (b) and run-time (r) environment will attain $\inf_{I,b,r}\{T2S(r(b(I)))\}$ for target 1 and target 2, respectively. The figure also hints that the idea of *portable performance is a contradiction in terms* and we will elaborate further on this in section 5. The real challenge, however, is that the infimum is not known beforehand and the tuning process will consequently attempt to take steps that will make $T2S$ decrease until one runs out of ideas or there is no more time to improve it further. Performance modelling is very useful in setting reasonable expectations and guiding this process, cf. appendix 8.1.

The target architectures that we aim at in this study have many similarities from an abstract point of view (see e.g. [8]) and this allows for a portable strategy towards optimization of the implementation. However, they are *not* identical and the devil is in the details. Eventually, one path will improve performance further on one architecture but will impair the performance on another. This is an important fact that requires special attention when one tries to compare performance across different platforms. The fact that the strategy towards optimization has many similarities makes it very tempting to approach the refactorization task using a classical computer science approach with abstraction layers etc. We tend to believe that this is a wrong approach for legacy codes like the one considered here since the restructuring required is simply too involved and there are no easy routes but analyzing the entire implementation line by line if the goal is to seriously improve the performance. We tend to think of the continued improvement process as depicted in figure 3. Imagine that you first shrink the algorithm representation to a minimal amount of memory transfers. Then the refactorization will attempt to organize all the loops such that parallel exposure is maximized while keeping the temporary memory overhead in storage and in transfers due to the implementation as low as possible. Eventually, trading additional memory transfers required for further splitting of the loops will not out-weight the benefits of added parallelism and the process will stop. This turning point differs from one target architecture to the next. Hence, the tuning process is much like the famous banana problem.

Worldwide, NWP codes are being refactored towards performance on the modern throughput architectures, cf. [7]. Since different versions of the source codes optimized for different target architectures are needed

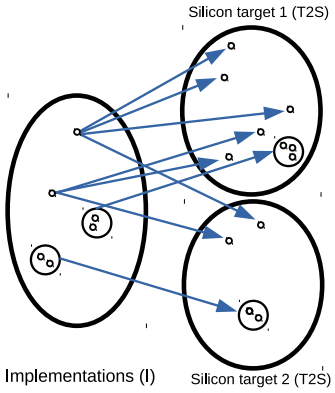


Figure 2: Implementation choices. The left hand side illustrates the set of all possible implementations of a given algorithm with total freedom in the choice of programming languages, parallelization models, etc. The two subsets on the right hand side illustrate the generated codes for two specific targets as a result of the implementation itself (I), the build and link instructions (b) and the run-time environment (r). The aim is to reach infimum; circles show that this is not attained by a unique combination.

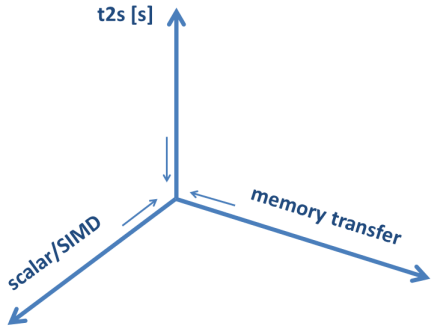


Figure 3: The tuning process. The aim is reducing $T2S$ as much as possible. Extra memory transfers need be traded for more SIMD vectorization. Splitting into more sub-loops implies increased temporary storage to provide an interface between these which again implies extra memory transfers that could have been handled in registers or at least in short latency cache parts.

and when even different generations of hardware are considered, a fair comparison of performance results is a challenge and can often be misleading. We will keep this issue in mind when presenting performance results in section 5.

3 Refactorization Steps

The initial optimization work aimed at ensuring a proper threading of the code. We used our usual SPMD approach to complete this, cf. [10]. This required a transition to Fortran-90 assumed-shape interfaces and that the stack memory usage was trimmed considerably. The primary elements to the refactorization process were ensuring contiguous data; reduce overall stack pressure by turning local temporary 2D/3D variables into 1D/2D variables and even in a few cases into scalars by aligning computations properly such that temporary storage

could be reduced or even omitted completely; the largest stack arrays was moved to the heap; proper NUMA-initialization of the heap arrays; collapsing loops over the outermost horizontal index; assuring no side effects in local functions (`pure` in Fortran); constant variables declared as constants (`parameter` in Fortran).

Further refactorization consisted of reducing the memory overhead and of pushing all branching out of the loops such that choices between different physical conditions are made from a top level of the dwarf. From the emerging more bare implementation we began to shuffle computations around to maximize the parallel exposure within each column, guided by recognition of commonly occurring computational patterns, cf. also appendix 8.1. That is, we organized all the vertical loops into sub-loops that had no dependencies and those that did. Sub-loops with dependencies are those that do conventional operations such as prefix-sums and reductions. All sub-loops without dependencies was SIMD vectorized[6] and SIMD tuned, and the ones with dependencies were vectorized if it seemed beneficial, e.g. using OpenMP SIMD reductions. It should be mentioned that prefix-sum operations can indeed be parallelized but the parallel algorithms for prefix-sums do not work well for the trip-counts of relevance to our applications and they were left as minimized non-SIMD parallelized vertical loops. Moreover, despite the fact that parallel prefix-sum operations are easily expressed using threads there are currently no directives in the OpenMP specification that allow for such expressions so one would have to express them explicitly. The result of these efforts are summarized in figure 4 and figure 5.

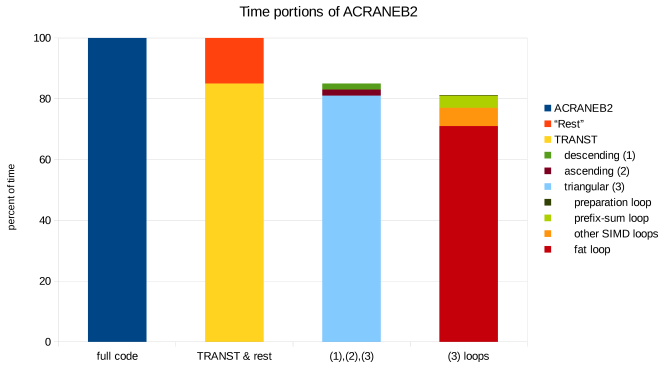


Figure 4: Classification of the main loops resulting from our initial tuning analysis. The bars indicates the portion of time that is spend in the respective code fragments on Xeon. See text for explanation.

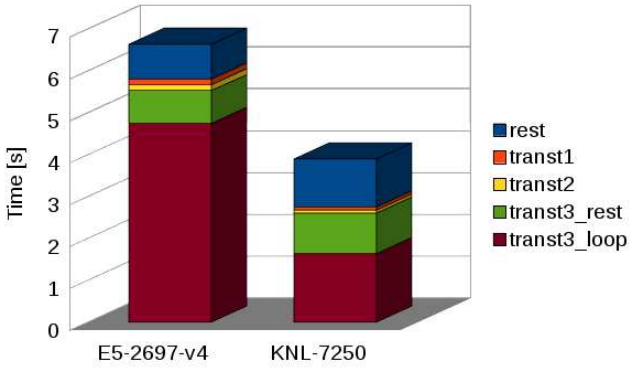


Figure 5: Time-to-solution when running the complete ACRANEB2 dwarf on a dual-socket Xeon (72 threads; left) versus single-socket Xeon Phi (272 threads; right). There is almost ∼2x difference in the overall performance but there are individual performance differences seen in individual sub-components.

Figure 4 shows the classification of sub-components that resulted from our refactorization efforts. The time spend in the full ACRANEB2 code (dark blue bar) is divided between the thermal radiation scheme, called

[6]For complicated loops, this does not happen automatically and one needs to tweak the code to allow the compiler to translate it into efficient SIMD instructions.

`transt` (yellow), and all the rest of the radiation physics (red). The `transt` is clearly the most time-consuming part of the full ACRANEB2 code. Diving into the `transt` component in the third bar, we separate that into three parts; a descending part (1), an ascending part (2), and a triangular part (3). The triangular part, which we shall denote by `transt3` in the following, is clearly the most expensive component, corresponding to $\sim$80% of the total ACRANEB2 compute time on a dual-socket Xeon E5-26xxv4. The fourth bar shows our final classification of the triangular part into a tiny preparation loop, a relatively expensive prefix-sum loop, a huge loop with high arithmetic intensity and referred to as the fat loop from now on, and a collection of smaller SIMD loops and non-SIMD loops.

Inside the fat loop, a large number of simple mathematical operations and transcendental functions as shown in table 1 are executed and 33 memory transfers of double precision data are performed. Since it is a triangular double nested loop the total trip count is $((81 * 80)/2 = 3240$ in our 80 layers test case for each horizontal point so the total FLOP-count becomes very large for this part of the code.

Table 1: Count of operations/functions inside the fat loop.

operation/ function	**count**
max	24
add	454
mul	308
div	48
sqrt	18
exp	14
log	8
pow	22

The cost of splitting the `transt` component into these sub-components is that the preparation loop must be repeated in each of the three parts (1), (2) and (3). However, this preparation loop was straightforward to SIMD vectorize and as a result time spend here was brought down to an insignificant contribution in the full context as indicated by the very thin slice which can hardly be seen on top of the fourth bar in figure 4. This is indeed well spent since it serves for preparing coefficient arrays for the more involved loops that follow, which can then concentrate on doing the work they are meant to do.

Figure 5 shows that our general refactorization efforts already look very promising on KNL, i.e. it was sufficient to SPMD thread parallelize the computations over the columns and SIMD vectorize over the vertical layers within each column in order to obtain competitive performance when running the complete dwarf on KNL. The actual time portions of the individual sub-components are slightly different on Xeon Phi than on Xeon: As expected, the non-SIMD vectorizable parts become relatively more expensive on Xeon Phi supporting AVX-512 than on Xeon supporting AVX2, and this appears as the blue `rest` part that has not been included in our major refactoring at all and in the green `transt3_rest` part that has been refactored but still contains prefix-sum and reduction patterns.

The initial refactorization efforts allowed us to simplify the dwarf and confine our focus to the `transt` kernel and the `transt3` kernel with a `SLOC` around 1600 and 700, respectively. These kernels have been ported and tuned to the target throughput architectures (Intel Xeon Phi and NVIDIA GPUs) of the ESCAPE project and have been evaluated against various SKUs from the Intel Xeon E5-26xxv4 series of CPUs released in 2016.

One of the most important steps during the continued refactorization process was to reorganize the loops such that the fat loop got a constant trip count. That is, in our 80 layers test case, for the triangular double nested loop with 80 iterations of varying trip count from 1 to 80, we paired short and long loop lengths and thereby obtained a constant trip count of 81 in the inner most loop but only half as many iterations.

4 Data structures and parallelization

Figure 6 shows the data-structure layout from the upstream code as documented in the IFS documentation, cf. [5], whereas figure 7 shows the new data-structure layout that we have mainly focused on in this paper. The upstream IFS thread parallelization is as shown in figure 8 done over the horizontal with a block granularity of tunable size `nproma`. Our new thread parallelization, shown in figure 9, is done over the horizontal too but with the fine granularity of a single horizontal point. It is important to stress that `nproma`=1 is *not* the same as a granularity of a single horizontal point. Both thread parallelization approaches are done using outlined constructs in order to minimize synchronizations costs and thus allowing the threads maximum freedom for parallel work.

```
subroutine foo_orig(...,jup,jlow,klon,klev,...)
  ! arguments a*
  real(kind=jprb), intent(in)    :: a1(klon)        ! size klon
  real(kind=jprb), intent(in)    :: a2(klon,klev)   ! size klon*klev
  real(kind=jprb), intent(inout) :: a3(klon,0:klev) ! size klon*(klev+1)
  ...
  ! local variables l*
  real(kind=jprb)                :: l1(klon)        ! size klon
  real(kind=jprb)                :: l2(klon,klev)   ! size klon*klev
  ...
  ! typical loop nest
  do jlev=0,klev     ! vertical loop with loop-carried dependencies
    do jlon=jlow,jup ! no loop-carried dependencies
      ...
      a3(jlon,jlev)= ...
      ...
    enddo
  enddo
  ...
end subroutine foo_orig
```

Figure 6: Fragment of the original ACRANEB2 code using Fortran-77 fixed-size dummy argument declarations implying that the actual arguments must be contiguous in memory. If the actual argument is not or might not be contiguous, the semantics of the language will force the compiler to copy the actual argument array to a contiguous temporary array and back upon return. The innermost `jlon`*-loop will be SIMD vectorizable by definition and this holds for all physics subroutines whereas the outermost* `jlev`*-loop often will suffer from loop carried dependencies. Note the artificial memory overhead for all stack variables* `l1,l2,...` *at this point in the call-tree and beyond imposed by this way of implementing the loop nests within the physics.*

The refactorization can be summarized as

- a significant reduction in the thread-local stack pressure
- a more fine grained thread parallel decomposition unit
- full exposure of yet another dimension of parallelism in the algorithm itself

The first item allows us to run far more threads simultaneously without hitting stack limits; this is a *necessary* condition that must be met if one wishes to scale the runs to many threads. The second item allows a better load balancing between the threads and the importance of this again increases at scale. The third item which carefully exposes the vertical parts that have no dependencies and hence can run in parallel from those that have dependencies is another *necessary* condition for running on highly parallel architectures, i.e. all parallelism inherited in the scheme must be explicitly exposed to the compiler. Finally, the size of the sub-chunks with dependencies have been minimized to allow for as much parallelism as possible.

At this point it seems reasonable to consider if we could benefit from reintroducing the blocked `jlon`-approach allowing the non-SIMD innermost sub-loops to SIMD vectorize by re-interchanging the loops. Figure 10 is an attempt to integrate our improvements with the upstream data structures assuming that the complete interchange of array indices is too time-consuming to do for the whole physics code base at once and that one therefore in

```
subroutine foo_new(...,jup,jlow,klon,klev,...)
  ! arguments a*
  real(kind=jprb), intent(in)    :: a1(:)    ! size klon
  real(kind=jprb), intent(in)    :: a2(:,:)  ! size klev*klon
  real(kind=jprb), intent(inout) :: a3(0:,:) ! size (klev+1)*klon
  ...
  contiguous :: a1,a2,a3
  ...
  ! local variables l*
  real(kind=jprb)                :: l1       ! size 1
  real(kind=jprb)                :: l2(klev) ! size klev
  ...
  ! typical loop nest
!$acc parallel                                                                &
!$acc present(...)
!$acc loop gang private(...)
  do jlon=jlow,jup  ! no loop carried dependencies
!$acc loop vector
    do jlev=0,klev  ! vertical sub-loop with no loop carried dependencies
      ...
      a3(jlev,jlon)= ...
      ...
    enddo
!$acc loop seq
    do jlev=0,klev  ! vertical sub-loop with loop carried dependencies
      ...
      a3(jlev,jlon)= ...
      ...
    enddo
  enddo
  ...
end subroutine foo_new
```

Figure 7: Fragment of our new ACRANEB2 code using Fortran-90 assumed-shape dummy argument declarations and with interchanged loop ordering. The bulk of the computations in the innermost `jlev`*-loop are SIMD vectorizable but some are not as shown here. There are no column dependencies so the outermost* `jlon`*-loop is thread parallelizable by definition and each thread will handle its own contiguous chunk of the global loop over all columns. Note that the artificial stack overhead caused by the original loop nest ordering is completely gone now.*

practise must do the refactoring component by component. The cost of this integrated approach compared to our proposal in figure 7 is a more bulky thread-local stack frame. Needless to mention this transition will come at a cost of higher thread stack pressure so it would only work well up to a certain size. With fewer threads this may not be an issue but as the number of threads increases so does issues related to this overhead, making it a competitive candidate on multi-core architectures with relatively few threads per node but less attractive on modern many-thread architectures. Thus, will we benefit from trading the overhead introduced with the added stack pressure with that of faster computations in the small loops that cannot be SIMD vectorized with the new data layout? This is an open question that we will address in section 5.

5 Performance results

We confine ourselves to present the *performance* attained on the reduced `transt3` kernel. We have verified (not shown here) that the results and the timings for the `transt3` component are the same if we perform measurements on this reduced `transt3` kernel or on the more involved `transt` kernel or on the full `acraneb2` dwarf, so that there is no need to complicate things more than necessary. Table 2 lists the architectures and SKUs used in this study, and throughout this paper we shall use the abbreviations shown in the first row of the table.

5.1 Time-to-solution results

Figure 11 summarizes the best single node, core and thread performance we attained on different Xeon and Xeon Phi systems. It is seen that the **weakest** KNL significantly outperforms the **strongest** dual-socket BDW

```
program bar_old
  ...
!$omp parallel do schedule(dynamic,1) private (jkglo,ibl)
  do jkglo=1,kgpcomp,nproma
    ibl=(jkglo-1)/nproma+1
    call foo_old(a1(1,1,ibl), ....)  ! F77-style
  enddo
  ...
end program

subroutine foo_old(a1,jlow,jup,klev,nproma...)
  real(kind=jprb),intent(inout)  :: a1(nproma,0:klev)
  ...
  real(kind=jprb)                :: l1(nproma)        ! size nproma
  real(kind=jprb)                :: l2(nproma,klev)   ! size nproma*klev
  ...
  do jlev=0,klev     ! vertical loop with loop carried dependencies
    do jlon=jlow,jup ! innermost loop without loop carried dependencies
      ...
    enddo
  enddo
  ...
```

Figure 8: Fragment of how threading is implemented in upstream IFS and HIRLAM-ALADIN codes. Unfortunately, the dwarf that we received for this study did not have surrounding OpenMP loop and comparisons with the original code beyond one core should therefore be treated with care.

at the node level with our refactored code. The fact that KNL at all comes close to BDW even at the core level is due to the strong SIMD parallelism that has been achieved as part of the refactorization of the implementation. The boxes in the same figure show the result of 4 years of Moores law by cross-comparing a dual-socket SNB in the high end of the SKUs released in 2012 with that of a dual-socket BDW and single socket KNL which both emerged in 2016. Note that there is a remarkable improvement as a result of Moores law even at the core and thread level for our refactored code.

Table 2: List of architectures

	SNB	**BDW**	**KNL**	**P100**
μ-arch	SandyBridge	Broadwell	KnightsLanding	Pascal
Released	2012	2016	2016	2016
SKUs	E5-2680v1	E5-2697v4	7210	P100
		E5-2699v4	7250	

Figure 13 shows time-to-solution for two different refactored codes on three platforms, BDW, KNL and P100. The code refactored for the GPU target is not really suited for the Xeon/Xeon Phi target, a property we shall come back to when discussing portable performance in section 5.2.

To answer the question posed at the end of section 4 we summarize in figure 12 the result from using the refactored code but retaining the original data-structures and original loop structures and the corresponding tuneable parameter `nproma`, cf. figure 10. All timings from this blended approach are consistently higher than the timings we can attain with our new codes, i.e. those shown in figure 13. The performance loss that comes from the original data organization is significant on KNL. The performance loss is consistent but less significant on the more traditional BDW for all values of `nproma`[7]. This experiment suggests that the traditional data structures and corresponding loop structures in atmospheric models is up for a reconsideration when one targets KNL and even BDW to a lesser extent, though. It is interesting to note that while the conclusion is clear for KNL, the conclusion for the P100 is less clear. Figure 12 reveals a sweet spot for `nproma=32` on P100. It is still 15% slower than the version with the new data-structures but the fact that none of the GPU alternatives so far have shown competitive absolute performance makes the `nproma`-version of the code another good candidate for tuning for the P100. Actually, when we got stuck in attempting to improve the performance

[7] Note that we had to increase `OMP_STACKSIZE` in order to run with the larger `nproma` values

```
program bar_new
  ...
!$OMP PARALLEL DEFAULT(shared)
  call acraneb2_numainit(...)
!$OMP END PARALLEL
  ...
!$OMP PARALLEL DEFAULT(shared)
  call acraneb2( ....)
!$OMP END PARALLEL
  ...
end program

subroutine acraneb2(...)
  ...
  call domp_get_domain(...,jlow,jup) ! get thread bounds
  ...
  do jlon=jlow,jup    ! chunk of horizontal loop handled by this thread
    do jlev=0,klev    ! innermost vertical loop
      ...
    enddo
  enddo
  ...
  call acraneb_subr(jlow,jup,...)  ! thread local calls to subroutines
  ...
end subroutine acraneb2
```

Figure 9: Fragment of how threading is implemented in our new code. Note that it is designed such that the load can be balanced among the threads based on the local properties of ACRANEB2 (or other properties that one might wish to expose to the implementation) through the `domp_get_domain` *call. A straightforward balancing would simply distribute the* `jlon`*-iterations evenly among the threads but for some physics component this could give rise to ill-balanced load. Thus, the design allows for flexible hooks to balance the load.*

on P100 further, we turned our attention to this `nproma`-candidate again and the best GPU result shown in figure 15 in section 5.4 stems from further GPU tuning of this implementation.

5.2 Portable performance

The source code used for all the targets is Fortran. The baseline code was written in Fortran and the authors have no reason to believe that code generation could be improved by switching entirely to or by combining it with source code written in another programming language. The parallelization, on the other hand, is expressed using the OpenMP programming model when targeting Xeon and Xeon Phi and using the OpenACC programming model when targeting NVIDIA GPUs. According to our experience the GPU does not like to treat larger chunks at the same time since this will lead to data spill, i.e. data that cannot reside in registers will get evicted to global memory and if the corresponding latencies can not be hidden the processor will simply idle. So, for performance on the GPU we need to confine the loops to treat smaller fractions one by one. Moreover, if shared memory is used too then this will also limit the number of thread-blocks that can run concurrently on the device and again be a performance obstacle. All in all this leads to a poor utilization of the available bandwidth, and the GPU will be mostly waiting for data and overall performance will suffer. Thus, the GPU tends to prefer more loop splitting (assuming that the latencies from the additional memory transfers resulting from this can be hidden behind real work) whereas with KNL one would stop the splitting once all SIMD potential is exposed and the caching system is well utilized. Therefore, *competitive performance* can not be portable across very different architectures such as the GPU and the Xeon Phi. The traditional Xeon line, on the other hand, seems to be less sensitive to the number of loop splits compared to Xeon Phi.

In order to treat all targets equal we have decided not to focus on the code resulting from refactoring for the GPU nor for the Xeon Phi solely since as revealed in figure 13 this could have led to too simple conclusions, especially in the case where the refactoring was done for the GPU. The figure also demonstrates that what one could refer to as *portable performance* can be quite far from *competitive performance* so in weighting the importance of portability versus performance one may sometimes have to choose between a portable layout of the loops resulting in *portable performance* and a less performance portable layout of the loops resulting in *competitive performance* on the primary target platform. However, on the GPU, with a gap of $\sim$12% the

```
subroutine foo_nproma(...,jup,jlow,klon,klev,...)
  ! arguments a*
  real(kind=jprb), intent(in)    :: a1(:)           ! size klon
  real(kind=jprb), intent(in)    :: a2(:,:)         ! size klon*klev
  real(kind=jprb), intent(inout) :: a3(:,0:)        ! size klon*(klev+1)
  ...
  contiguous :: a1,a2,a3
  ...
  ! local variables l*
  real(kind=jprb)                :: l1(nproma)      ! size nproma
  real(kind=jprb)                :: l2(nproma,klev) ! size nproma*klev
  ...
  do j=jlow,jup,nproma  ! no loop carried dependencies
    iup = min(nproma,jup+1-j)
    do jlev=0,klev      ! vertical sub-loop with no loop carried dependencies
      do i=1,iup        ! no loop carried dependencies
        jlon = j + i - 1
        l2(i,jlev)= ...
        ...
      enddo
    enddo
    do jlev=0,klev      ! vertical sub-loop loop carried dependencies
      do i=1,iup        ! no loop carried dependencies
        jlon = j + i - 1
        a3(jlon,jlev)= ... l2(i,jlev)
        ...
      enddo
    enddo
  enddo
  ...
end subroutine foo_nproma
```

Figure 10: Fragment of the new ACRANEB2 code using Fortran-90 assumed-shape dummy argument declarations and with all the sub-loop rewrites from `foo_new()` *but with the original* `nproma`*-blocked loop nest ordering inlined into the subroutine itself. All innermost loops are now SIMD vectorizable and the bulk of the outermost loops are SIMD vectorizable too. The block size* `nproma` *is a tunable parameter that one can used to tune the size of the individual stack-frames for performance.*

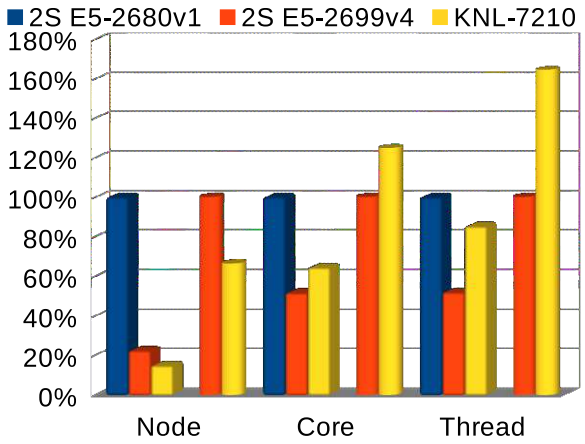


Figure 11: Node, core and thread performance for `transt3` *on different Xeon and Xeon Phis for the refactored code. The cylinders cross-compare the performance of the* **strongest** *dual-socket BDW SKU aka E5-2699v4 (red) with that of the* **weakest** *KNL SKU aka KNL-7210 (yellow). The boxes cross-compare a dual-socket SNB (blue) with that of BDW (red) and KNL (yellow).*

performance of the Xeon targeted code is not too far from that of the GPU targeted code in our case which could guide the choice if one had to stick to a single code version due to e.g. maintenance costs. This would imply that the GPU target become less interesting since the GPU performance is by no means competitive with this source code. In this context it should be stressed that both code versions could be improved further for their respective targets, thus certainly enlarging the gap; this is shown later in section 5.4 for the GPU target.

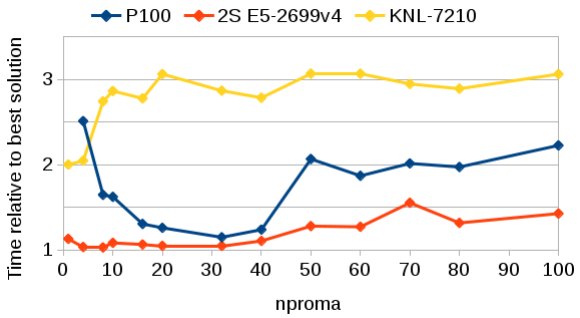


Figure 12: The `nproma` *experiment with* time-to-solution *relative to the* time-to-solution *obtained for each of the three platforms with our refactored data organization for varying values of* `nproma`*. Again, timings are for* `transt3`*. The GPU timings do* not *include PCI communication.*

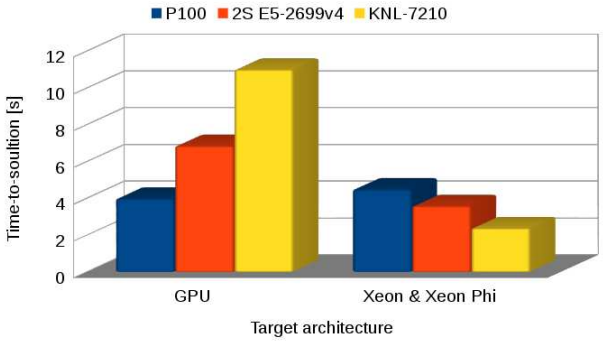


Figure 13: Time-to-solution for two different refactored codes that both retain the same interface on three platforms. The left-hand side shows the performance attained on the three platforms when the code was refactored for the NVIDIA GPU target whereas the right-hand side shows the performance attained on the three platforms when the code was refactored for the Xeon Phi target. Timings are again for `transt3`*. Note that section 5.4 investigates a faster version on the GPU where we allowed the interface to change too. The GPU timings do* not *include PCI communication.*

5.3 Absolute performance

If we only present *time-to-solution* in a relative context as we did in the previous sections then we may cheat ourselves by a poor baseline for performance. Thus, we now turn our attention to absolute performance measures to put the results into a proper context. We used the Intel SDE tool[8] and instrumented the code with an SDE portion surrounding the fat loop within the `transt3` kernel.

Figure 14 shows absolute performance on KNL-7210 for the fat loop. The fat loop sustains approximately 800 GFLOP/s DP and 900 GFLOP/s DP on KNL-7210 and KNL-7250, respectively. Being an absolute measure, we can cross-compare it with other published numbers. For instance, [11] shows that the fastest kernel out of 8 kernels in the NERSC/Trinity benchmark sustains 506 GFLOP/s. In appendix 8.1 we will treat the question if sustaining 41%-46% of achievable peak (HPL performance) constitutes a roof or if there is opportunities for improvements. The good absolute performance on KNL translates to BDW too, not directly one-to-one but in the sense that improvements from refactoring for KNL also yields improved absolute performance on BDW. This is the case for NVIDIA P100 too, i.e. efforts on improving for KNL also improved the performance on P100 but as shown in figure 13 this did not lead to competitive performance on P100 nor did the further tunings efforts on this version of the code. A profile on P100 confirmed (not shown here) that the GPU utilization is

[8] https://software.intel.com/en-us/articles/calculating-flop-using-intel-software-development-emulator-intel-sde. It is our experience that this tool is the most reliable tool to measure the FLOP-counts.

limited by register usage and each SM is limited to execute only 4 blocks simultaneously. Thus, in theory there is indeed room for improvements on P100 if we can manage to split the computations further and at the same time be able to hide the memory latencies resulting from extra memory transfers required to bind the smaller chunks together. For this code, however, we were not able to improve it in practice despite the theoretical potential. The totally different `nproma`-candidate was much easier to improve for the GPU target as revealed in section 5.4.

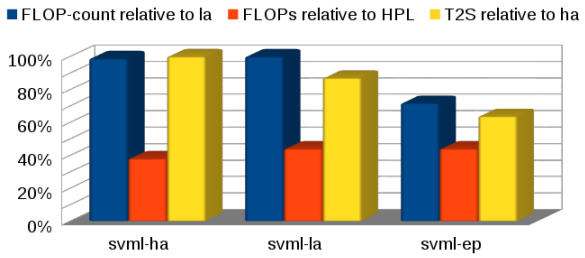


Figure 14: Absolute performance on KNL 7210 using different modes for the MKL vector math library revealing that GFLOP/s is a poor performance measure of performance for the fat loop in this particular kernel. Moreover, we sustain 41%-46% of HPL performance for two of the 3 modes of the MKL vector math library.

The algorithm used in this chunk is *compute minimal* in the sense that all computations are necessary and sufficient for defining the output. The algorithm delivers results in two output arrays, O_1 and O_2, and is consequently not considered to be *memory output minimal*. On Xeon and Xeon Phi there is sufficient cache memory available to benefit from computing O_1 and O_2 in one go. On the GPU, on the other hand, the fastest version shown in figure 15 consists of two independent *memory output minimal* chunks, one computing O_1 and another computing O_2. As revealed above this split is not sufficient so further splitting is needed and this will - by definition - introduce additional overhead that has to be compensated for, either by completely hiding this overhead or by exceeding the sustained KNL performance in order to become competitive with the KNL performance.

5.4 Best performance

As hinted in previous subsections we needed to tune the GPU code variant further to utilize the GPU potential better and achieve competitive performance. Thus, we departed from the `nproma`-candidate and introduced more loop splitting to overcome the obstacles revealed above to create our best performing code for the GPU target. We gained a further $\sim$1.9 times speedup such that instead of the 4.0 s for the GPU to the left in figure 13 we achieved 2.3 s which is faster than our best timing on the smallest KNL to the right in figure 13. It should be stressed, that when running with this more dedicated GPU code version on Xeon and Xeon Phi the performance suffered seriously on the Xeons[9] to a degree much worse than apparent from left part of figure 13 due to severe cache pollution.

The best node performance that we attained on different architectures released in 2016 is summarized in figure 15. This is a direct head-to-head comparison of our implementations on architectures that one could purchase at the same time. Note that in order to obtain a performance on the GPU that is competitive with the performance on Xeon and Xeon Phi (and vice versa) we need to handle different refactored code versions, but when doing so, performance become almost identical on the largest KNL and on the largest GPU that were available for purchase at the same time.

If one further as an experiment relaxes a little bit on the restriction not to modify the mathematical functions that come with the algorithm developed by renowned radiation physicists, one can replace the power function

[9]Timings increased to more than 1 minute on the BDW and 5 minutes on the KNL, cf. left part of figure 18.

`x**y` with the mathematically equivalent but numerically different expression `exp(y*log(x))` (in Fortran, that is). The replacement forces a more straightforward implementation which avoids too high local memory usage by the compiler. The result is a further ∼1.25 times speedup on P100 which we show as alternative (b) in figure 15. A similar gain can not be achieved on Xeon or Xeon Phi where the compiler and the performance math library (svml-ep) already are doing a similar job. It should be mentioned that in the testcase used here we obtained the same results with the two formulations on P100, but this will generally, of course, not be the case, maybe not even for the range of values that occur in radiation physics, so one should be careful not to draw conclusions too soon; it is out of the scope of the present paper to question the mathematical formulas used in the radiation code.

Note, there is a ∼3x between the fastest and the slowest timings on figure 15 if one allows for both transposed data structures and thereby changed interface as well as changed mathematical formulation. It should, however, be stressed that the performance attained is also a function of the algorithm at hand and not just a function of the hardware capabilities. A given algorithm may map better to some architectures than to others and this does not imply that some architectures are better than others. Thus, this figure does *not* imply that best possible performance of *any* algorithm is always almost the same on KNL and GPU. It only shows the status of our work on the various refactorizations of the implementation of the ACRANEB2 algorithm.

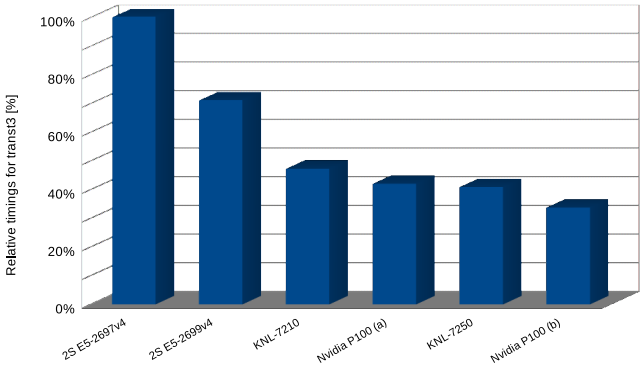


Figure 15: Relative time-to-solution for `transt3` *from the best performing code versions on the respective architectures, i.e. this is not portable performance but a result of cross-comparing different versions of the source code, each one explicitly crafted to target an individual architecture. For P100, (a) and (b) are without and with algebraic rewrite of the power function, respectively. The GPU timing does* not *include PCI communication.*

5.5 Energy results

We will now treat performance using the measure *energy-to-solution*. We ran this test on E5-2697v4 and KNL-7250 without turbo mode using 72 and 272 threads, respectively and we ran 500 iterations of the fat loop in order to get sufficient samples for the power measurements[10]. The power was measured using the method described in [3] using the `ISCoL` tool. Table 3 presents the entire system characteristics, including measured time and power consumption for the fat loop.

The normalized node performance relative to the dual-socket E5-2697v4 is summarized in figure 16 for our refactored code and shows that *time-to-solution* is improved by 2.4x by choosing the KNL over a dual-socket BDW but *energy-to-solution* is improved even more by 2.9x so KNL is indeed delivering more performance per Watt. Thus, our refactored code is more efficient on KNL compared to on BDW than what would be suggested from the HPL performance in table 3 (HPL ratio 1.57x and EER 2.32x, respectively) both with respect to time and energy.

[10]This is system power measurements for the entire node, i.e. including both CPU and memory system. Energy is power times time.

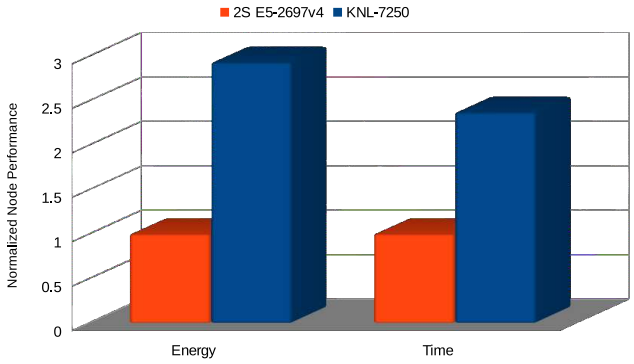


Figure 16: Node performance improvement normalized to BDW. Note that the ratio exceeds the ratio obtained by HPL both in time and energy.

Table 3: Comparison of `transt3` *performance to system characteristics. Percentages are relative to BDW. HPL EER is the HPL energy efficiency ratio, i.e. the HPL performance per Watt, relative to BDW. The two last rows are measurements on the* `fat loop`.

	BDW	**KNL**
SKU	E5-2697v4	7250
HPL [GFLOP/s]	1236	1939
HPL [GFLOP/s/W]	2.26	5.24
HPL ratio [%]	100	157
HPL time [%]	100	64
HPL EER [%]	100	232
`loop` power [W], 500 iterations	4.59	3.71
`loop` time [s], 1 iteration	3.377	1.428

5.6 Scaling

The 400x400 setup exceeds Amdahl-99.95% strong scaling and it also weak scales perfectly from 400x400 to 1500x1500 on KNL-7210 (not shown here). Thus, up-scaling the timing so that we account for 100% and not just the 80% accounted for by `transt3`, we reach a first crude estimate of the number of nodes needed for running a setup of size 1200x1080x80 which in the horizontal corresponds to the largest setup that we run in production today and in the vertical exceeds the largest setup by 15 layers. This means that 5 to 10 KNL-7210 nodes would be sufficient to run the full ACRANEB2 on this large setup in 0.5 to 1 seconds.

6 Conclusion

Our results suggest that investments in software development and performance maintenance[11] certainly pays off and refactoring of legacy code may have a significant impact on performance on modern hardware. There are multiple arguments as summarized in the following.

First, we may draw the attention to the challenge we started off with, namely that the radiation scheme is a bottleneck in today's operational NWP production, cf. figure 1. There is a vast potential for improving

[11] We consider a continued effort in refactoring the code to adjust to trends in hardware evolution as a MUST for the daily maintenance of the code. The surroundings are moving, new conditions are being prescribed and one will have to follow in order not to contribute to the technical debt of the project.

the current implementation as revealed in this paper. Our completely refactored implementation of the most expensive algorithm outperforms the effects of improving it at the algorithmic level, i.e. by adding support for intermittency. This software re-factoring immediately pays off since it allows for doing much more physics under the fixed constrains on time-to-solution and on hardware investment as well as on the energy budget.

Secondly, the importance of our software refactoring becomes even more important on the newer architectures as shown in figure 17. The baseline code was clearly not suited for the modern throughput architectures. To be able to run the baseline code at all on a NVIDIA GPU, we had to do a significant amount of non-trivial code preparation just to ensure the semantics ended up being correctly understood by the compiler. The correctness of this work was verified with the Cray compiler on an older NVIDIA K20x. Further, with this modified baseline code we had to use smaller testcases on the GPUs due to lag of sufficient memory space and up-scale the timings to the 400x400x80 reference. The performance of this GPU-ported baseline code is better when instead the PGI compiler is used with similar performance on K20x (not shown) as on P100, but unfortunately the initial results were also slightly off so the initial preparation steps were apparently not sufficient to obtain portable OpenACC behaviour. On Intel Xeon and Xeon Phi the baseline code ran correctly out of the box. The completely refactored codes gave correct results on all the tested hardware and with all compilers tested across all incarnations (testcase size, thread count, etc) and this includes the OpenACC ports to the GPUs too.

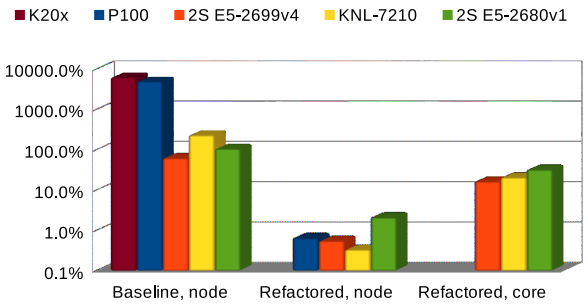


Figure 17: Time-to-solution relative to the baseline implementation on a SNB node for `transt` *in the full* `acraneb2` *dwarf. Note, the vertical axis is logarithmic in order to embrace the range of performance results. The baseline code performance on single nodes of different architectures is shown to the left. Bars in the middle show the single node performance of the refactored codes, and the right bars show the single core performance of the refactored codes. Using the Cray compiler on NVIDIA GPU K20x (brown) and the PGI compiler on P100 (blue), and the Intel compiler on Intel BDW (red), KNL (yellow) and SNB (green). Single-core performance is not sensible for the GPU.*

Figure 17 shows that the two 2016 technologies Intel KNL and NVIDIA P100 perform much worse than the 2012 technology (SNB) when we run the baseline code, and the gap is significant with KNL-7210 being $\sim$2 times slower and P100 being $\sim$4500 times slower than a dual-socket SNB from 2012. However, running the refactored code on all platforms reveals a very different picture. Now P100 and KNL-7210 beat SNB by more than a factor of 6. For single core, the baseline code on the 2016 Xeon technology (BDW) beats the 2012 Xeon technology (SNB) by a factor of 1.7, but with our refactored version the factor is more than doubled to 3.8.

Figure 18 is showcasing the difference between portable performance and competitive performance. In this figure code bases X and G (which are also shown earlier in figure 13) are pretty much the same code except for the splitting in G, while code base GNM is essentially a complete re-write with modified data-structures, interface, loop order, reformulation of power function, and on top of that a more involved splitting. Note, we had to use a log-axis to cover the range of timings.

The obtained gains in performance should be seen in the perspective of how much one can expect from the hardware evolution, and to this end we compare node performance of the `transt3` kernel with HPL and STREAM TRIAD node performance relative to SNB in table 4. For BDW vs SNB the ratios are $\sim$4.2 and $\sim$1.6, respectively, and thus with a factor of $\sim$4.5 our refactored code performs slightly better than expected

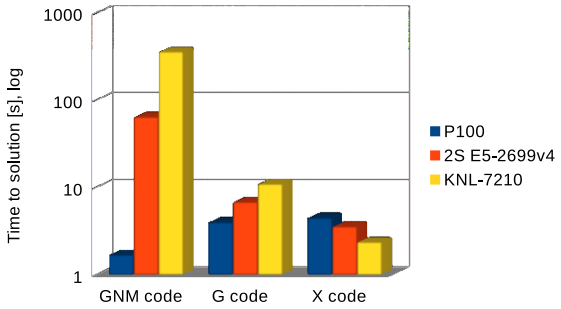


Figure 18: Time-to-solution for the three different code bases on three different architectures. X is the Xeon target code. G is the GPU target code using the data structures as X, but with split into seven chunks. GNM is the GPU target with transposed data structures as compared to X and G, reformulated power function and even more splits (into 12 chunks).

from the hardware evolution alone. For the smaller KNL-7210 the ratios are ∼5.6 and ∼5.6, and for the larger KNL-7250 the ratios are ∼5.7 and ∼6.2, and our refactored code with ∼6.7 and ∼7.8, respectively, performs significantly out of these ranges which we attribute to the fact that KNL has some of the transcendental functions implemented in hardware[12] and this part of the ISA is not exercised by HPL. Thus, good SIMD vectorization in the code therefore becomes even more important. For P100, the performance improvement is better for our refactored code than for STREAM TRIAD, and including the rewrite of the power function the improvement factor is getting quite close to the high HPL improvement factor, thus utilizing a major portion of the potential performance boost from the SNB to P100 evolution. Note, such improvements as demonstrated in table 4 can not be obtained with the baseline code for any of the architechtures, only with the refactored code. Even on the single core the refactorization pays off compared to the baseline code on a full node; this holds for the older SNB hardware too but even more on the newer BDW and KNL. The improvements of the refactoring on newer hardware compared to the older SNB is more than accounted for by increased thread-count times clock-frequency, which demonstrates the importance of proper utilization of SIMD vectorization[13]. Thus, it is evident that our months on refactoring of the code has orders of magnitude higher impact on the performance for this code than 4 years of hardware evolution. It is important to stress that this does *not* prove lack of progress in evolution of hardware but rather it emphasizes the issue with legacy code.

The improvement in time-to-solution due to our refactoring for the entire `transt` code and not only for `transt3` is summarized in table 5 and figure 17. It is interesting but not surprising to observe that the refactoring has a more significant impact on newer hardware than on older hardware. It is important to stress that the improvements at the node level are somewhat incomplete in the sense that the dwarf that we received was single threaded. It is also important to stress that we did not have time to merge the fastest implementation of `transt3` on P100[14] into the `transt` code; completing this step will bring the refactored node performance for the GPU to be fastest of all the architectures considered in this paper, and in the last row (italicized) in table 5 we have estimated the corresponding improvement factor for the GPU target.

Based on our experience from working with operational met-ocean models, we believe that the radiation dwarf considered in the present paper serves as a typical example with respect to refactoring potential for NWP components, so for entire models we will expect that speed-up in orders of magnitudes can indeed be achieved on modern hardware by a deep refactoring of the entire code. It will, however, take a huge and continued effort to deal with the technical debts inherent in many of these models currently as well as to prevent it from growing further as the hardware trends evolve.

We have also shown that the process of tuning code for different architectures is the same but also that it

[12] ISA improvements in SQRT, DIV and AVX-512ER

[13] SNB has AVX with 4 SIMD lanes, BDW has AVX2 with 4 SIMD lanes but also FMA, KNL has AVX-512 with 8 SIMD lanes and FMA.

[14] i.e. from the best performing code version shown in figures 15 and 18.

Table 4: Node performance improvement factors relative to SNB. For P100, (a) and (b) are without and with algebraic rewrite of the power function, respectively.

Architecture	HPL	Stream Triad	`transt3`
E5-2680v1	1.0	1.0	1.0
E5-2699v4	4.2	1.6	4.5
KNL-7210	5.6	5.6	6.7
KNL-7250	5.7	6.2	7.8
NVIDIA-P100 (a)	11.4	6.9	7.6
NVIDIA-P100 (b)	11.4	6.9	9.5

Table 5: Refactorization improvement factor on a single node and on a single core for different architectures.

Architecture	**Core**	**Node**
E5-2680v1	3.3	50
E5-2699v4	3.7	110
KNL-7210	11.0	667
NVIDIA P100	N/A	7302
NVIDIA P100	N/A	*17000*

will diverge eventually and one will end up with completely different code bases in the end. To quantify the differences in the two incomplete attempts of today (one for the GPU target and one for the Xeon target), the relative `SLOC` difference is 50% and the size of the `diff` between the two source files exceeds the size of each of the files. The local variables in the two implementations have different dimensions and the input/output used in one implementation are transposed in the other implementation so even the interfaces differ. In practice, one would consequently have to maintain two code bases despite the fact that we have confined ourselves to the use of directive based approaches. Moreover, we have seen that the latency tuned architectures are less sensitive to where we stop the splitting process and also less sensitive to the choice of loop nest ordering. The highly parallel throughput tuned architectures, on the other hand, are very sensitive to this. Thus, from this particular study we can conclude that *portable performance* is quite far from *competitive performance* and we need to be very cautious when cross-comparing performance obtained on KNL vs GPU. One could have chosen to stop refactoring at the simplest code X in figure 18, claiming that one code base is sufficient, sacrificing competive performance on the GPU for increased portability and maintenance costs. There is already some orders of magnitudes gain in performance on both KNL and P100 using the X target code compared to using the legacy code, cf. figure17, so it might be tempting to stop the refactoring process here. But if performance really matters we would have to discriminate the refactoring. We can certainly *not* expect that we can just decorate the very same code base both with OpenMP and OpenACC directives and then get code generated that will run efficiently on both targets. Numerous attempts on a pure OpenMP and OpenACC directive approach using the very same code base have been made by the present authors and their collaborators, and failed.

Finally, we have seen that refactoring of legacy code *is* indeed required for getting performance out of investment in newer hardware, and with refactored code we can improve *time-to-solution* by choosing one of the new highly-parallel and throughput tuned architectures but we have also seen that on top of this we gain even more performance improvement if the metric is energy. Thus, it seems obvious to us that we have to prepare our entire workload such that it will be able to embrace the future technologies. There is a vast potential in legacy codes that will be revealed when we start to invest in refactorization and we say *when* and not *if* because the latter - to the best of our knowledge - is not a sustainable option.

7 Future directions

Our refactorization plan follows a pattern that is directly applicable to all the other physics components in IFS and ALADIN-HIRLAM systems too, thus accounting for about half of the total runtime in today's operational NWP models. As we have shown earlier in a previous study, cf. [2], it is indeed possible to refactor the more involved dynamics too and thereby the entire model which would require a new in-depth analysis of the current implementation of dynamics.

It is an open question if it would pay off combining the improved algorithm using intermittency with our improved implementation of the expensive step and reap the harvest from both improvements simultaneously. This will for sure increase maintenance costs and there might not be gain in physics results so the gain in time-to-solution should be significant to justify such an approach.

Our present study also revealed a significant use of transcendental functions and an obvious question would be if one could relax on the mathematical physics formulation by substituting the use of these functions with purpose build Padé polynomials instead. Finally, as the resolution scale becomes even finer, it also seems relevant to investigate multi-grid strategies and our colleagues have actually pursued this idea further.

Acknowledgment

The authors would like to thank Ján Mašek, ONPP/CHMI, Kristian Pagh Nielsen and Bent Hansen Sass, DMI for not just providing us with the wrapped dwarf code, the corresponding test cases and input for figure 1 but also for countless discussions on radiation physics. Moreover, we wish to express our gratitude to Karthik Raman, Ruchira Sasanka and Michael Greenfield, Intel, Peter Messmer and Stan Posey, NVIDIA and John Levesque, Cray for their great support of this study. Special thanks go to Alan Gray, NVIDIA for analysis of the register usage in the `transt3` kernel and for pointing out the advantage of reformulating the power function for the P100. Thanks to numerous people in the HPC and NWP communities for commenting constructively on various early drafts of the manuscript. Thanks to Cray for allowing us to use the Marketing Partner Network system `swan`, to Intel for allowing us to use the `Endeavour` cluster and to NVIDIA for allowing us to use their `PSG` cluster to complete this study. The ESCAPE project has received funding from the European UnionâĂŹs Horizon 2020 research and innovation programme under grant agreement No 671627.

8 Appendices

8.1 Roofline analysis

Roofline analysis is centered around a definition of *operational intensity* and a sometimes naive throughput assumption and it often serves as a valuable tool in guiding code optimization work. For any given implementation I, one may calculate the operational intensity $J(I) = W(I)/Q(I)$ defined as the ratio between the work W to the memory traffic Q. A common metric for work is FLOP-count and a common metric for memory traffic is number of bytes being moved in which case intensity will be the arithmetic intensity denoted AI and measured in FLOP/byte. The naive roofline model uses achievable peak bandwidth B_{max} sustained by the stream triad benchmark and achievable peak performance P_{max} sustained by HPL to limit the performance of I by $P(I) = min(P_{max}, J(I) \times B_{max})$. Measuring sustained performance of $S(I)$ and cross-comparing this with the computed $P(I)$ may sometimes reveal room for improvements for the implementation I at the platform given implicitly by (B_{max}, P_{max}). In cases dealing with fat loops, the I1 instruction cache may be too small to hold the loop and if that happens, then P_{max} will be too optimistic. Moreover, successive iterations of

the loop can only overlap by a small relative amount if I is fat and the throughput assumption will not hold true. Instead, the true in-core execution is dictated by the critical path execution time and hence P_{max} will again be too optimistic.

Roofline analysis is useful and reasonably accurate when the implementation mainly contains simple operations that translate directly to hardware instructions, e.g. `ADD` and `MUL`. It is is less suited for comparing different implementations of different algorithms nor is it simple to use in cases where the implementation contains many complicated operations.

For this particular kernel, we have already revealed that the main loop is rather packed with complicated transcendental functions, cf. table 1. Nevertheless we will attempt to construct a performance model for the fat loop and use it in the context of roofline analysis.

8.1.1 Roofline analysis to guide code refactoring

In section 3 we described the initial refactoring in kind of a hand-waving way, but one could also describe the process using a more formal roofline-based argumentation. For example, cf. upper part of figure 19, analysis of legacy codes will often reveal an inner loop with a contents which is recognized as a mixture of a non-SIMD patterns and some SIMD patterns. In this particular case, the prefix-sum will prevent SIMD vectorization thus spoiling performance of the entire loop. If n_f and n_b denotes the number of FLOP and BYTEs, respectively, referenced in the function `foo`, the arithmetic intensity of the mixed loop will be $(n_f + 1)/(n_b + 3 * 8)$, assuming 8 byte reals. In the refactored code, cf. lower part of figure 19, the loop has been split into an explicit prefix-sum loop and a SIMD vector loop for the remaining part. The prefix-sum loop has a very low AI of only $1/(3 * 8) \approx 0.04$ but will be able to run at full memory bandwidth, or possibly even directly out of the cache. The AI of the SIMD loop is $n_f/(n_b + 3 * 8)$ which is slightly lower than the AI of the mixed loop, but this is insignificant for performance when n_f is relatively high, i.e. especially when the SIMD loop is fat. What is important here is that this loop will now SIMD vectorize and we can sustain much better utilization of the hardware for the refactored code, even when some portions are inherently non-SIMD friendly.

```
!-  mixed-loop code with a hidden prefix-sum pattern -------
sum = 0.0_jprb
do i=1,n
  sum  = sum + z(i)
  a(i) = foo( sum, ... )
enddo


!-  refactored code w/loop split ---------------------------
!
! explicit prefix-sum:
zsum(0) = 0.0_jprb
do i=1,n
  zsum(i) = zsum(i-1) + z(i)
enddo
!
! SIMD vector loop:
do i=1,n
  a(i) = foo( zsum(i), ... )
enddo
```

Figure 19: Sketch of pattern identification and the following loop splitting in a typical refactorization process. The loop-carried dependency in the first loop will prevent SIMD vectorization. It is assumed that the function `foo` *has no dependencies or side-effects (*`pure function` *in Fortran) and that it can be inlined.*

8.1.2 Establishing a performance model

First, we notice that the fat loop contains a vast number of long-latency operations (`DIV` and `SQRT`), and of transcendental functions (`POW`, `EXP`, `LOG`) with corresponding FLOP-counts being highly implementation-, context- and argument-dependent. So, even if we could translate each operation or function into an equivalent

FLOP-count on a given platform we must be aware that the issues like pipelining of instructions, latency, dependencies and argument range may obscure the performance model, making it more crude and maybe even less useful in practice.

Then, we created series of small stand-alone kernels, one for each operation that is considered. We used the Craypat tool with both the Intel compiler and with the Cray compiler on BDW and KNL to estimate the FLOP-count for each of these kernels on each platform and thereby we are able to translate the results into a representative FLOP-count for each operation. The consistency of this approach was then tested by repeating the experiment using the Intel SDE tool with the Intel compiler on BDW and KNL. The results of these experiments are shown in tables 6 - 7. We show FLOP-counts for the simple operations and transcendental functions that appear in the fat loop. In table 6 the results are from using the Intel compiler on BDW (upper part) and KNL (lower part) and both the Craypat tool (left) and the SDE tool (right). We have here considered the four MKL variants, i.e. the serial libm and the three vector modes svml-ha, svml-la and svml-ep[15]. In table 7 the results are from using the Cray compiler on BDW (left part) and KNL (right part) and the Craypat tool. Also, different math translations are considered through different choices of compiler flag, `-O0` and `-O2`, respectively.

Table 6: FLOP-count experiment using the Intel compiler.

	BDW, Craypat tool				**BDW, SDE tool**			
	libm	**ha**	**la**	**ep**	**libm**	**ha**	**la**	**ep**
max	1	1	1	1	1	1	1	1
add	1	1	1	1	1	1	1	1
mul	1	1	1	1	1	1	1	1
div	1	1	1	1	1	1	1	1
sqrt	1	1	1	1	1	1	1	1
exp	19	20	14	10	19	21	14	10
log	25	21	16	12	24	23	19	15
pow	55	64	47	21	55	64	49	22
	KNL, Craypat tool				**KNL, SDE tool**			
	libm	**ha**	**la**	**ep**	**libm**	**ha**	**la**	**ep**
max	2	2	2	2	1	1	1	1
add	1	1	1	1	1	1	1	1
mul	1	1	1	1	1	1	1	1
div	2	16	8	8	1	1	6	6
sqrt	2	15	15	15	1	14	14	14
exp	88	17	16	11	19	23	22	13
log	66	29	22	21	28	34	28	19
pow	201	77	72	41	55	78	72	40

As expected the obtained FLOP-count for the transcendental functions varies with choice of math library. With the Intel compiler, the results obtained with Craypat are very consistent with the results obtained from SDE on BDW but not on KNL; also note here that except for the simplest operations the vector modes of MKL have relatively high FLOP-counts on KNL even for the fast low-accuracy (la) and extended-performance (ep) modes. Moreover, the results with the Cray compiler are consistent with the results with the Intel compiler on BDW (both using Craypat), but on KNL the results differ quite a lot. Finally, it must be stated (not shown) that the FLOP-count obtained in this way is of course heavily dependent on the actual values of the arguments to the functions, and we have here limited ourselves to show only results obtained with some "representative" argument values.

[15] https://software.intel.com/sites/products/documentation/doclib/mkl/vm/vmdata.htm

Table 7: FLOP-count experiment using the Cray compiler.

	BDW, Craypat tool		KNL, Craypat tool	
	-O0	**-O2**	**-O0**	**-O2**
max	1	1	4	2
add	1	1	1	1
mul	1	1	1	1
div	1	1	2	16
sqrt	1	1	19	16
exp	18	18	130	16
log	19	19	241	29
pow	190	95	1769	195

Note that on KNL, the `DIV` operation is converted to "`MUL` 1/x" with mode la and ep, and this explains the jump from 1 FLOP to 6 FLOP in the SDE Intel runs. It is expected that something similar but not quite the same happens with the Craypat tool using the Intel compiler (8 FLOP for mode la and ep).

This exercise so forth just demonstrates that it will be necessary to operate with a different set of FLOP-count numbers for functions from different math libraries and that care should be taken before relying too much on these FLOP-count numbers as a basis for code optimization like e.g. in roofline analysis.

Assuming that partial FLOP-count numbers have been collected for each considered operation and function, then it is simply a matter of using these to build a performance model for a loop: Add up the operations weighted by their respective occurrence count. Divide this total FLOP-count by the number memory transfers that you have in the loop to obtain the AI. In our case, as shown in tables 8 - 9, we obtain AI values from $\sim$6 with svml-ep from Intel MKL on BDW to a staggering $\sim$170 using low optimization with the Cray compiler on KNL. A typical application would use the more safe math for precision and accuracy studies during testing and development but jump to faster but lower precision (e.g. svml-la reached through the compiler flag `-fimf-precision=medium` for performance runs, and in these cases the AI from our performance model is $\sim$8-10 for the loop. Arithmetic intensities of this order is very high compared to what is usually seen for loops in NWP models, thus deserving its fat loop label.

Table 8: Arithmetic intensity (AI) and FLOP-count for the fat loop using the Intel compiler. PM is GFLOP/S in the loop from our performance model, TM is GFLOP/S measured by the tool, DEV is deviation between PM and TM in %.

	BDW, SDE tool				BDW, Craypat tool			
	libm	**ha**	**la**	**ep**	**libm**	**ha**	**la**	**ep**
AI	9.7	10.4	8.7	6.2	9.7	10.3	8.4	5.9
PM	41.3	44.6	37.4	26.3	41.5	44.0	36.0	25.3
TM	37.9	43.9	36.2	24.4	38.8	44.0	35.3	23.8
DEV	-8.1	-1.4	-3.1	-7.2	-6.6	0.0	-1.9	-5.9

	KNL, SDE tool				KNL, Craypat tool			
	libm	**ha**	**la**	**ep**	**libm**	**ha**	**la**	**ep**
AI	9.9	13.1	13.2	9.8	26.5	15.5	13.2	10.2
PM	42.2	56.1	56.3	41.7	113.3	66.2	56.4	43.6
TM	38.9	54.1	54.6	40.4	107.9	65.1	55.6	41.7
DEV	-8.0	-3.6	-3.0	-3.1	-4.7	-1.6	-1.5	-4.5

Table 9: Arithmetic intensity and FLOP-count for the fat loop using the Cray compiler. PM is GFLOP/S in the loop from our performance model, TM is GFLOP/S measured by the tool, DEV is deviation between PM and TM in %.

	BDW, Craypat tool		**KNL, Craypat tool**	
	-O0	**-O2**	**-O0**	**-O2**
AI	20.6	12.7	170.5	25.3
PM	88.2	54.3	729.1	108.4
TM	89.8	51.6	729.1	121.2
DEV	1.9	-5.1	0.0	11.8

8.1.3 Tool vs model

The applied tools, i.e. Craypat and SDE, can of course be used to directly measure the FLOP-count for the loop in question. One might even be so lucky that tools can be applied to profile performance of smaller fragments in a real context and obtain reliable results. But, honestly, it is our experience that this is not always the case and we also encourage to treat such measurements with great care.

The performance model described in the previous subsection is based on measuring the individual FLOP-count for each function, and this may come handy during development when one e.g. tries to implement a new code piece by piece or tries to optimize legacy code, while keeping focus on performance. We do, however, need to verify that this approach is good enough for the specific purpose at hand. We expect that our performance model is crude, but if we should be able to use it in a larger context, our model must still be sufficiently reliable under the conditions described (i.e. system by system, library by library, argument by argument, ...).

We have compared our performance model with results from the tools. Tables 8 - 9 show the total FLOP-count in GFLOP/s for the loop in a semi-real context (the test case was tuned to 500 iterations of a 1x10x80 grid configuration in order to satisfy the tools). The overall picture is that our performance model in this case explains pretty much all the FLOP measured by the tools, and most of the runs using the more optimized libraries are within $\sim$5% (again disregarding the model result from using Cray compiler on KNL which is $\sim$12% off in the fast version).

8.1.4 Roofline analysis for the fat loop

In figure 20 we show some selected results from roofline analysis of the loop in full context using the performance model. The model grid size is 400x400x80 which we were not able to profile in a reliable way with the Craypat tool and therefore we had to stick to our performance model. Since the test case is supposed to mimic a realistic situation we have used the performance library MKL svml-la, but we have for comparison also showed one result using the serial MKL libm. Actually, this serial result is placed higher in the roofline diagram and therefore has a better FLOP/s performance than the results using the vectorized library on the same node and on a smaller BDW node, but does this then mean that the performance number that really matters, i.e. the time-to-solution, also is better?

No, obviously not. In figure 21 we compare the time-to-solution from using the vector MKL svml-la on all four SKUs with that of using the serial MKL libm on the largest BDW. From this figure it is clear that MKL libm on the 88 thread BDW is slower than the rest, a conclusion that can not be drawn clearly from figure 20. This demonstrates that roofline on its own can be of limited use as a performance-measuring tool in these more involved contexts. However, the 272 threads KNL-7250 is best performing according to both roofline and time-to-solution. On the smaller KNL our implementation sustains $\sim$41% of the HPL performance both with svml-la as shown in figure 20 and with svml-ep as shown in figure 14. On the larger KNL it reaches $\sim$46%. The AI is 13.2 on the KNLs. On the two BDWs, however, AI is 8.7 which is on the left hand side of the knee

of the roofline and thus STREAM TRIAD is the proper measure here; our implementation sustains ∼26% and ∼30% of the STREAM TRIAD performance on the smaller and larger BDW, respectively.

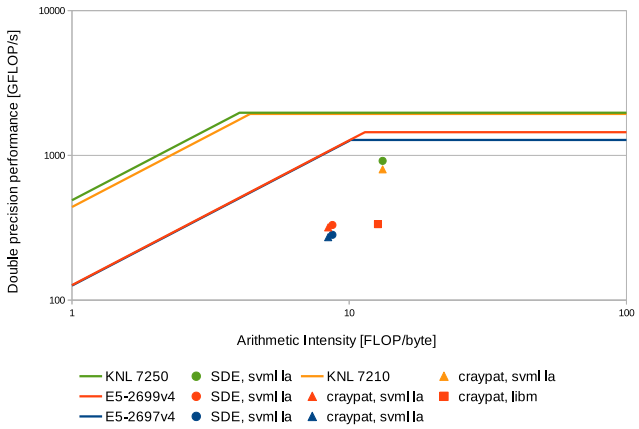


Figure 20: Roofline diagram showing some selected results. Green color is used for KNL-7250, orange is for KNL-7210, red is for E5-2699v4 and blue is for E5-2697v4. Horizontal lines are from the HPL benchmark while the sloping lines are from the STREAM TRIAD benchmark, cf. table 10. Markers are results obtained from the performance model using the Intel compiler and the maximum number threads on each SKU, i.e. 272 on the KNL-7250, 256 on the KNL-7210, 88 on the larger BDW and 72 on the smaller BDW. Results from the performance library MKL svml-la are shown as circles using SDE and as triangles using Craypat. The square marker indicates the result from using Craypat and the default MKL libm library.

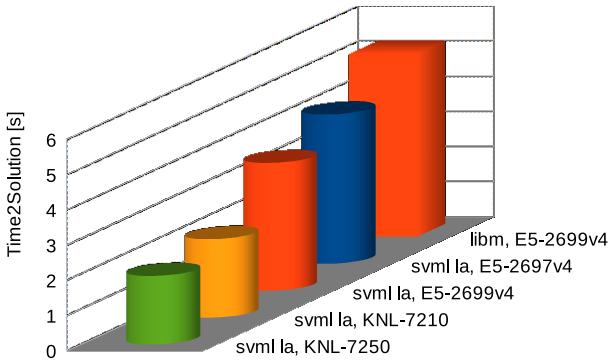


Figure 21: Time to solution for four selected cases using the Intel compiler: The circular bars are from using the performance library MKL svml-la on the four platforms, while the square bar is from using the default library MKL libm on the large BDW. Color of the bars correspond to the color of the markers in the roofline diagram in figure 20.

8.2 Build and Run specifications

Figure 22 summarizes the build instructions and figure 23 summarizes run instructions in `BASH`-syntax for Xeon/Xeon Phi and NVIDIA, respectively. The `ifort` compiler versions used was `17.0.1.132 Build 20161005` whereas the `pgi` compiler version used was `17.4-0` and the cce compiler was version `8.5.8`. We used `turbo` mode on all SKUs but E5-2697v4. For the KNL systems, this benchmark is so highly flop bound that it is neutral to whether it runs out of DDR or MCDRAM and also neutral to whether we run in flat or cache mode. As for KNL kernel configurations, we found that `CONFIG_HZ_250=y` gave slightly better timings than `CONFIG_HZ_1000=y`.

```
tar -zxvf dwarf-transt3_v<version>.tar.gz
cd dwarf-transt3_v<version>

# bdw/knl
BDW_TARGETF="-xCORE-AVX2"; KNL_TARGETF="-xMIC-AVX512"
TARGETF=<your_choice>
Fep="-O2 $TARGETF -ipo -fimf-precision=low -fp-model fast=2"
Fla="-O2 $TARGETF -ipo -fimf-precision=medium"
Fha="-O2 $TARGETF -ipo -fimf-precision=high"
FCFLAGS=$Fep FC=ifort ./configure --enable-openmp --host=x86_64-linux-gnu
make

# p100
TAF_DEFAULT="-ta=nvidia"; TAF_MAX80REGS="-ta=nvidia,maxregcount:80"
TAF=<your_choice>
F="-mp $TAF -acc -fast -Minline=levels:3 -Mcuda=cuda8.0 -Mcuda=fastmath"
FCFLAGS=$F FC=pgf90 ./configure --enable-openmp --enable-openacc && make
```

Figure 22: Build instructions for reproducing the builds used in this paper.

```
tar -zxvf dwarf-transt3_testcase.tar.gz
cd dwarf-transt3_testcase

#bdw/knl, cray system
export OMP_NUM_THREADS=<threads>; export KMP_AFFINITY="disabled,verbose"
aprun -n1 -N1 -d<threads> -j2 -cc depth dwarf # bdw
aprun -n1 -N1 -d<threads> -j4 -cc depth dwarf # knl

#bdw/knl, non-cray system
export OMP_NUM_THREADS=<threads>; export KMP_AFFINITY="compact,verbose"
dwarf

#p100
export OMP_NUM_THREADS=1
srun dwarf
```

Figure 23: Run instructions.

Finally, table 10 summarizes the HPL and Stream Triad numbers used for roofline analysis and for evaluations of the absolute performance sustained. The numbers for Intel hardware were received from private correspondence with Intel while the NVIDIA P100 numbers were obtained from a Dell published study[16] and from private correspondence with NVIDIA.

The authors are strong supporters of Nature's theme on transparent and reproducible science and code sharing[17] and welcome anyone to contact us if they are interested in the implementations mentioned in this paper.

Table 10: HPL and Stream Triad performance reference numbers.

Architecture	**HPL** [GFLOP/s]	**Stream Triad** [Gbyte/s]	**TDP** (W)
E5-2680v1	343	79	260
E5-2697v4	1278	126	290
E5-2699v4	1446	127	290
KNL-7210	1933	440	215
KNL-7250	1971	490	215
NVIDIA-P100	3900	540	300

[16] http://en.community.dell.com/techcenter/high-performance-computing/b/general_hpc/archive/2017/03/14/application-performance-on-p100-pcie-gpus

[17] https://www.nature.com/polopoly_fs/1.16232!/menu/main/topColumns/topLeftColumn/pdf/514536a.pdf

References

[1] Lisa Bengtsson, Ulf Andrae, Trygve Aspelien, Yurii Batrak, Javier Calvo, Wim de Rooy, Emily Gleeson, Bent Hansen Sass, Mariken Homleid, Mariano Hortal, Karl-Ivar Ivarsson, Geert Lenderink, Sami Niemelä, Kristian Pagh Nielsen, Jeanette Onvlee, Laura Rontu, Patrick Samuelsson, Daniel Santos Muñoz, Alvaro Subias, Sander Tijm, Velle Toll, Xiaohua Yang, and Morten Ødegaard Køltzow. The HARMONIE–AROME Model Configuration in the ALADIN–HIRLAM NWP System. *Monthly Weather Review*, 145(5):1919–1935, 2017.

[2] Per Berg, Karthik Raman, and Jacob Weismann Poulsen. Complete HBM model runs on Intel Xeon processors and Intel Xeon Phi processors - part I. Technical report, DMI, Copenhagen, 2016.

[3] W. Michael Brown, Andrey Semin, Michael Hebenstreit, Sergey Khvostov, Karthik Raman, and Steven J. Plimpton. Increasing molecular dynamics simulation rates with an 8-fold increase in electrical power efficiency. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, pages 8:1–8:14, Piscataway, NJ, USA, 2016. IEEE Press.

[4] ECMWF. *Part IV: Physical Processes*. IFS Documentation. ECMWF, 2013.

[5] ECMWF. *Part VI: Technical and Computational Procedures*. IFS Documentation. ECMWF, 2016.

[6] J.-F. Geleyn, J. Mašek, R. Brožková, P. Kuma, D. Degrauwe, G. Hello, and N. Pristov. Single interval longwave radiation scheme based on the net exchanged rate decomposition with bracketing. *Quarterly Journal of the Royal Meteorological Society*, 143(704):1313–1335, 2017.

[7] Mark Govett, Jim Rosinski, Jacques Middlecoff, Tom Henderson, Jin Lee, Alexander MacDonald, Ning Wang, Paul Madden, Julie Schramm, and Antonio Duarte. Parallelization and Performance of the NIM Weather Model on CPU, GPU and MIC Processors. *Accepted for Bulletin of the American Meteorological Society*, 2017.

[8] Brent Leback, Douglas Miles, and Michael Wolfe. Tesla vs. Xeon Phi vs. Radeon - A Compiler Writer's Perspective. In *CUG Conference Proceedings*, CUG 2013, Napa Valley, California, USA, 2013.

[9] J. Mašek, J.-F. Geleyn, R. Brožková, O. Giot, H.O. Achom, and P. Kuma. Single interval shortwave radiation scheme with parameterized optical saturation and spectral overlaps. *Quarterly Journal of the Royal Meteorological Society*, 142(694):304–326, 2016.

[10] Jacob Weismann Poulsen, Per Berg, and Karthik Raman. Chapter 3 - Better Concurrency and SIMD on HBM. In James Reinders and Jim Jeffers, editors, *High Performance Parallelism Pearls: Multicore and Many-core Programming Approaches*, volume 1, pages 43 – 67. Morgan Kaufmann, Boston, MA, USA, 2015.

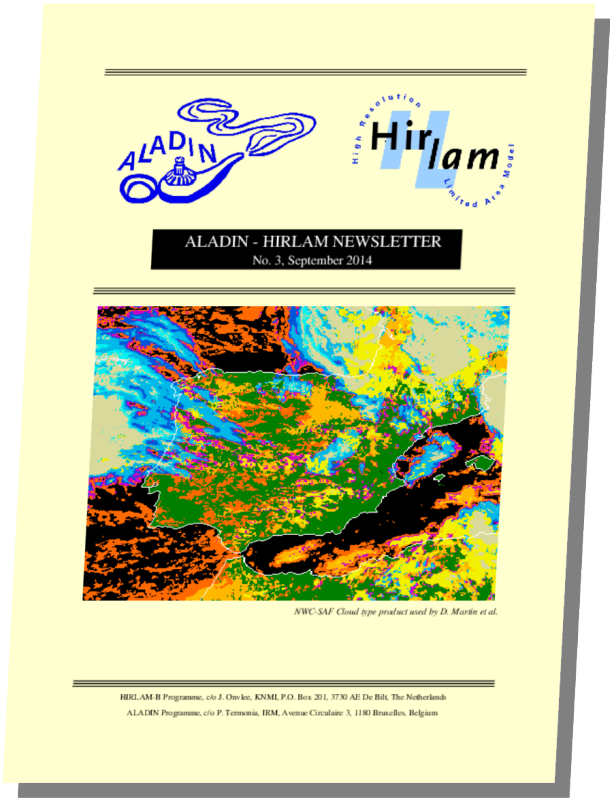
[11] Antonio C. Valles, Chuck Yount, and Sundaram Chinthamani. Chapter 25 - Trinity Workloads. In James Reinders, Jim Jeffers, and Avinash Sodani, editors, *Intel Xeon Phi Processor High Performance Programming Knights Landing Edition*, pages 549–579. Morgan Kaufmann, Boston, MA, USA, 2016.

ALADIN-HIRLAM Newsletters : previous issues

No. 1, Sept 2013

No. 2, April 2014

No. 3, Sept 2014



No. 4, Feb 2015

ALADIN-HIRLAM Newsletters : previous issues

No. 5, Aug 2015


ALADIN-HIRLAM Newsletter
No 5, August 2015

Joint 25th ALADIN Workshop & HIRLAM All Staff Meeting 2015
13-16/04/2015, Elsinore, Denmark

ALADIN Programme, c/o P. Termonia, IRM, Avenue Circulaire 3, 1180 Bruxelles, Belgium
HIRLAM-B Programme, c/o J. Onvlee, KNMI, P.O. Box 201, 3730 AE De Bilt, The Netherlands

No. 6, Fev 2016


ALADIN-HIRLAM Newsletter
No 6, February 2016

Around ALADIN&HIRLAM in 130 pages

ALADIN Programme, c/o P. Termonia, IRM, Avenue Circulaire 3, 1180 Bruxelles, Belgium
HIRLAM-B Programme, c/o J. Onvlee, KNMI, P.O. Box 201, 3730 AE De Bilt, The Netherlands

No 7. Sep 2016


ALADIN-HIRLAM Newsletter
No. 7, September 1st, 2016

Joint 26th ALADIN Workshop & HIRLAM All Staff Meeting 2016
04-07/04/2016, Lisbon, Portugal

ALADIN Programme, c/o P. Termonia, IRM, Avenue Circulaire 3, 1180 Bruxelles, Belgium
HIRLAM-C Programme, c/o J. Onvlee, KNMI, P.O. Box 201, 3730 AE De Bilt, The Netherlands

No 8. Jan. 2017


ALADIN-HIRLAM Newsletter
No 8, January 2017

Grand Tour of ALADIN & HIRLAM

ALADIN Programme, c/o P. Termonia, IRM, Avenue Circulaire 3, 1180 Bruxelles, Belgium
HIRLAM-B Programme, c/o J. Onvlee, KNMI, P.O. Box 201, 3730 AE De Bilt, The Netherlands