Geosci. Model Dev., 5, 111–128, 2012
www.geosci-model-dev.net/5/111/2012/
doi:10.5194/gmd-5-111-2012

Geoscientific Model Development

The 1-way on-line coupled atmospheric chemistry model system MECO(n) – Part 2: On-line coupling with the Multi-Model-Driver (MMD)

A. Kerkweg[1] **and P. Jöckel**[2]

[1]Institute for Atmospheric Physics, University of Mainz, 55099 Mainz, Germany
[2]Deutsches Zentrum für Luft- und Raumfahrt (DLR), Institut für Physik der Atmosphäre, 82234 Oberpfaffenhofen, Germany

Correspondence to: A. Kerkweg (kerkweg@uni-mainz.de)

Received: 23 May 2011 – Published in Geosci. Model Dev. Discuss.: 21 June 2011
Revised: 10 January 2012 – Accepted: 11 January 2012 – Published: 19 January 2012

Abstract. A new, highly flexible model system for the seamless dynamical down-scaling of meteorological and chemical processes from the global to the meso-γ scale is presented. A global model and a cascade of an arbitrary number of limited-area model instances run concurrently in the same parallel environment, in which the coarser grained instances provide the boundary data for the finer grained instances. Thus, disk-space intensive and time consuming intermediate and pre-processing steps are entirely avoided and the time interpolation errors of common off-line nesting approaches are minimised. More specifically, the regional model COSMO of the German Weather Service (DWD) is nested on-line into the atmospheric general circulation model ECHAM5 within the Modular Earth Submodel System (MESSy) framework. ECHAM5 and COSMO have previously been equipped with the MESSy infrastructure, implying that the same process formulations (MESSy submodels) are available for both models. This guarantees the highest degree of achievable consistency, between both, the meteorological and chemical conditions at the domain boundaries of the nested limited-area model, and between the process formulations on all scales.

The on-line nesting of the different models is established by a client-server approach with the newly developed Multi-Model-Driver (MMD), an additional component of the MESSy infrastructure. With MMD an arbitrary number of model instances can be run concurrently within the same message passing interface (MPI) environment, the respective coarser model (either global or regional) is the server for the nested finer (regional) client model, i.e. it provides the data required to calculate the initial and boundary fields to the client model. On-line nesting means that the coupled (client-server) models exchange their data via the computer memory, in contrast to the data exchange via files on disk in common off-line nesting approaches. MMD consists of a library (Fortran95 and some parts in C) which is based on the MPI standard and two new MESSy submodels, MMDSERV and MMDCLNT (both Fortran95) for the server and client models, respectively.

MMDCLNT contains a further sub-submodel, INT2COSMO, for the interpolation of the coarse grid data provided by the server models (either ECHAM5/MESSy or COSMO/MESSy) to the grid of the respective client model (COSMO/MESSy). INT2COSMO is based on the off-line pre-processing tool INT2LM provided by the DWD.

The new achievements allow the setup of model cascades for zooming (down-scaling) from the global scale to the lower edge of the meso-γ scale ($\approx$1 km) with a very high degree of consistency between the different models and between the chemical and meteorological boundary conditions.

1 Introduction

The quality of the results of a regional (or limited-area) atmospheric model are highly influenced by the conditions prescribed at the model domain boundaries.

For the meteorological/dynamical state of limited-area models, these boundary conditions are usually prescribed from analysis, reanalysis or forecast data from global or regional numerical weather prediction models or from global climate models, the so-called *driving models*[1]. Technically,

[1]Appendix B contains a glossary explaining some terms repeatedly used here. The terms from the glossary are written in italics throughout the article.

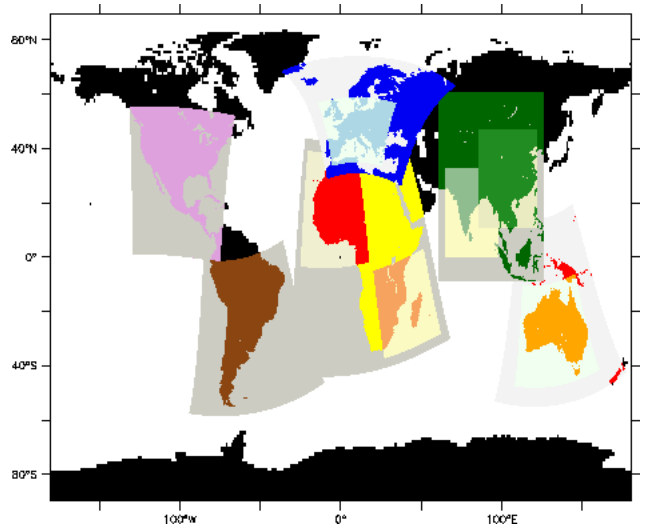

Fig. 1. Example for a MECO(n) model cascade setup using 13 instances (1 ECHAM/MESSy, $n = 12$ COSMO/MESSy instances).

the boundary data is read from specifically pre-processed data files stored on disk with a typical time resolution of 1–6 h, depending on the *driving model*.

Including processes for atmospheric chemistry in limited-area models forms a particular challenge in this respect, mainly because the amount of prognostic variables increases drastically. First, the chemical constituents need to be included, and second a higher update frequency for the chemical boundary conditions is desirable in order to be able to resolve the diurnal changes. In addition, it is favourable to describe the chemical processes in the limited-area model as consistent as possible with those in the *driving model*.

These requirements render the off-line coupling, where data are exchanged via the disk system, unfeasible. We therefore propose an on-line coupling approach, where the *driving model* and the limited-area model run concurrently in the same MPI environment and directly exchange the fields required to calculate the initial and boundary conditions via the computer memory. Figure 1 gives an example for such a model setup, showing 13 model instances running concurrently: one global instance (ECHAM5/MESSy) drives six COSMO/MESSy instances directly. Two of these COSMO/MESSy instances (covering Europe and Australia) drive again one further smaller scale COSMO/MESSy instance each, two more COSMO/MESSy instances (covering Asia and Africa) drive two further COSMO/MESSy instances each. All these instances are running simultaneously using the newly developed system. To abbreviate the naming we call this system MECO(n), "**M**ESSy-fied **E**CHAM and **CO**SMO nested **n** times".

The outline of this document is as follows: first, we describe the applied model components (Sect. 2) and provide details about the implementation of the on-line coupling (Sect. 3). In Sect. 4 we classify our coupling approach in comparison to other coupling approaches used in Earth system modeling. Section 5 explains the few adjustments of namelist and run-script entries, which are required to run a model cascade. The Multi-Model-Driver (MMD) library, performing the data exchange between the concurrently running models, is described briefly in Sect. 6. The "MMD library manual", which is part of the Supplement, comprises a detailed description of the library routines. Sections 7 and 8 sketch the work flow of the server submodel MMDSERV and the client submodel MMDCLNT, organizing the coupling on the server and the client side, respectively. Section 9 explains some technical details about the implementation of the on-line coupling into COSMO/MESSy and of the stand-alone program INT2LM as MESSy sub-submodel INT2COSMO[2]. Some remarks on how to achieve the optimum run-time performance efficiency with MECO(n) are provided in Sect. 10, and in Sect. 11 we close with a summary and an outlook. Example applications are presented in the companion articles (Kerkweg and Jöckel, 2012; Hofmann et al., 2012).

2 Applied model components

This new approach is based on the Modular Earth Submodel System (MESSy, Jöckel et al., 2005, 2010). MESSy provides a variety of process parameterisations coded as independent submodels, e.g. for gas-phase chemistry (MECCA, Sander et al., 2005), for scavenging of trace gases (SCAV, Tost et al., 2006), convective tracer transport (CVTRANS, Tost et al., 2010), etc. Furthermore, MESSy provides the interface to couple these submodels to a basemodel via a highly flexible data management facility.

MESSy has been connected to the global climate model ECHAM5 (Roeckner et al., 1996) extending it to the atmospheric chemistry general circulation model (AC-GCM) ECHAM5/MESSy (Jöckel et al., 2006, 2010). Furthermore, MESSy was connected to the non-hydrostatic limited-area weather prediction model of the Consortium for Small-Scale Modelling (COSMO, previously called "Lokal-Model" (LM), Steppeler et al., 2003; Doms and Schättler, 1999) resulting in the regional atmospheric chemistry model COSMO/MESSy (Kerkweg and Jöckel, 2012). Therefore, all processes included in MESSy and used in both models are described consistently on the global and the regional scale, if ECHAM5/MESSy is used as *driving model* for COSMO/MESSy.

During the last years, the numerical weather prediction model COSMO was further developed to fulfil the requirements for a regional climate model by the Climate Limited-area Modelling (CLM)-community (see Rockel et al., 2008). These developments also include the expansion of the stand-alone program INT2LM, which is provided by the German

[2]The "MMD user manual", which is also part of the Supplement, provides detailed information about this.

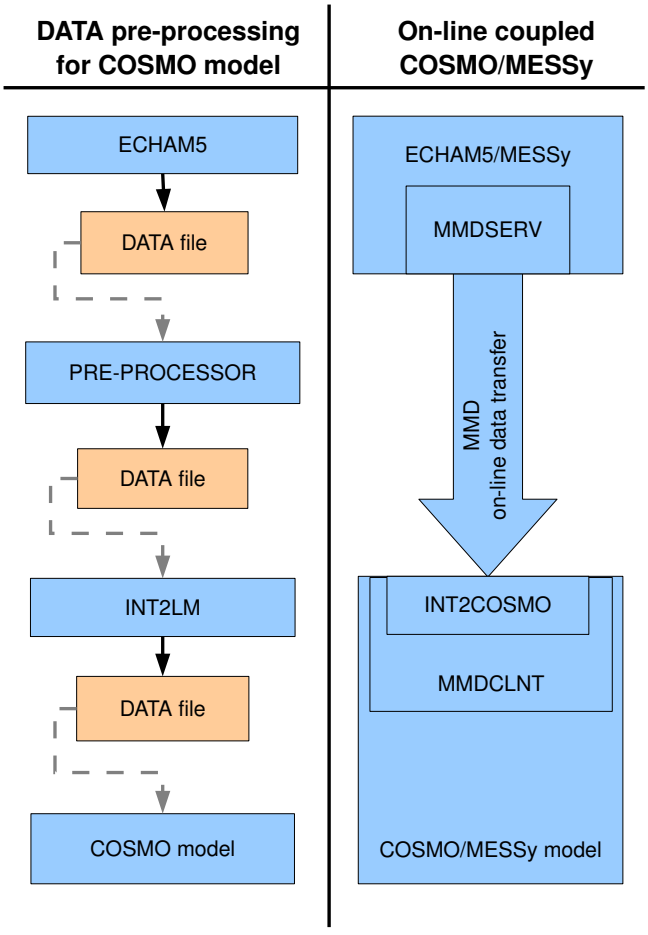

Fig. 2. Comparison of the data (pre-)processing procedure providing boundary data for one time step to one instance of the stand-alone COSMO model (left) and of the on-line coupled COSMO/MESSy model using ECHAM5/MESSy as *driving model* (right). The stand-alone COSMO model requires four independent sequential tasks. First, the *driving model* ECHAM5 is run. Second, the output files of ECHAM5 are used as input to a pre-processing tool converting the ECHAM5 data to a format readable by INT2LM. Third, the pre-processor INT2LM calculates and outputs the initial and boundary files for the COSMO model, which, fourth, is run to perform the intended simulation. In the on-line coupled model system the input data required for the COSMO model is exchanged on-line via the MMD library and interpolated on-line in the MESSy sub-submodel INT2COSMO. Thus no intermediate manual data processing is required.

Weather Service (DWD) for the pre-processing of the initial and boundary data for the COSMO model: the original INT2LM as provided by DWD can process data from three different *driving models*:

– the global DWD grid point model on an icosahedral grid (GME),

– the global spectral model IFS of ECMWF and

– the regional COSMO model, as the COSMO model can be nested (so far off-line) into itself.

In addition to the standard *driving models* supported by the DWD, the INT2LM was further developed by the CLM-Community to also interpolate data of climate models (e.g. ECHAM or REMO) and other weather prediction models. In case of ECHAM an additional pre-processing procedure is required to transform the standard output data of the climate model into a uniform format which can be handled by INT2LM.

The left-hand side of Fig. 2 depicts a common pre-processing procedure for producing the initial and boundary data with the stand-alone INT2LM for one time step of the COSMO model: first, the global model (here ECHAM5) is run. Afterwards, the output of ECHAM5 needs to be pre-processed to be readable for INT2LM. Subsequently, INT2LM is run and initial and boundary data for the COSMO model are produced. Finally, the simulation with the COSMO model is performed.

If this pre-processing procedure is not only performed for the dynamical part of the model, but also for the chemical part, the pre-processing time and the required data storage increase enormously, as chemical setups require boundary data for most chemical tracers taken into account. State-of-the-art atmospheric chemistry mechanisms typically consist of about 50 to a few hundred tracers. Furthermore, to capture the diurnal variations, the coupling frequency is higher for atmospheric chemistry simulations. To avoid the increasing effort for the pre-processing and the additional data, we implemented the on-line coupling of ECHAM5/MESSy to COSMO/MESSy and of COSMO/MESSy to COSMO/MESSy into MESSy. This is sketched on the right-hand side of Fig. 2. Currently, the on-line coupled model system is based on the ECHAM model version 5.3.02, the COSMO model version `cosmo_4.8_clm12` and MESSy version 2.41.

3 Coupling procedure

To carry out the on-line coupling, we extended MESSy by the Multi-Model-Driver (MMD) library and two MESSy sub-models (MMDSERV and MMDCLNT). Following a client-server approach, the server (*driving model*) provides the data to the client model, which subsequently calculates its own initial and boundary data. The MMD library manages the data exchange between the individual (parallel decomposed) tasks of the different model executables very efficiently, as the field exchange during the time integration is implemented as point-to-point, single-sided, non-blocking MPI communication. Figure 3 sketches the role of the MMD library, which is based on the message passing interface (MPI) library.

The right-hand side of Fig. 2 depicts the data processing procedure for the on-line coupled models ECHAM5/MESSy and COSMO/MESSy. The MESSy submodel MMDSERV

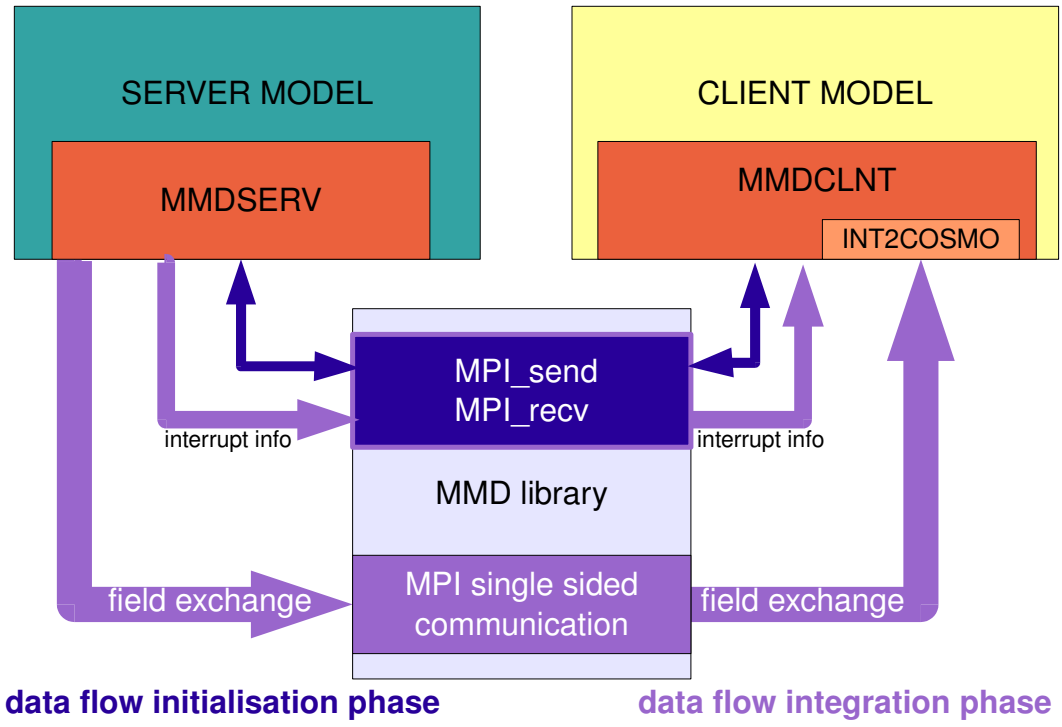

Fig. 3. Data exchange between the different model components: during the initialisation phase information is exchanged via MPI direct communication between the server and the client model (dark blue arrows). During the integration phase the server provides the *in-fields* to the client via MPI point-to-point single sided non-blocking communication (violet). The additional information, if the server is interrupted after the current time step (e.g. for restarts), is exchanged each server time step using MPI direct communication.

manages the data exchange for the server model. MMDCLNT not only carries out the data exchange for the client, it also performs the interpolation of the data provided by the server to produce the initial and boundary data required by the COSMO model. The latter is accomplished by the implementation of the stand-alone pre-processing program INT2LM as MMDCLNT submodel INT2COSMO. Thus, the INT2LM routines are also used for the calculation of the initial and boundary data in our on-line coupling approach. Furthermore, MMDCLNT provides the framework to use the INT2LM interpolation routines to interpolate *additional fields*, e.g. the tracers for chemistry calculations.

So far, we presented only one on-line coupled client-server pair. However, the MMD library provides the possibility to run an arbitrary number of models concurrently in the same MPI environment, which is only limited by the hardware capabilities. Thus a simulation setup is possible, in which one global model (i.e. ECHAM5/MESSy) is server for a number of COSMO/MESSy models. Each of the COSMO/MESSy models can again be server for a number of smaller scale COSMO/MESSy models and so forth (cf. Fig. 1). Thus, an entire cascade of on-line coupled models can be run concurrently. Figure 4 illustrates an example layout for an on-line coupled MESSy model cascade. Further examples for a ECHAM5/MESSy → COSMO/MESSy → COSMO/MESSy coupling are presented in the accompanying articles (Kerkweg and Jöckel, 2012; Hofmann et al., 2012).

4 Discussion on the chosen coupling approach

In the Earth system modelling community the terms *coupling*[3], *coupler* and *coupled* are widely used to describe a variety of aspects of connections within an Earth System Model (ESM), though rarely precised. In general, these terms are used to express some way of how different *components* influence each other. Appendix A provides a detailed classification of coupling approaches.

The choice of the most suited *coupling* method depends on many aspects, not least on the desired application, but in summary a proper choice usually provides a compromise to minimise the computational and communication overheads, the implementation effort (which depends mostly on the code structure of the involved legacy model(s)), and to maximise the desired degree of flexibility and sustainability. From the classification of *operators* (also in Appendix A),

[3]Note that the terms written in italics in this section do not refer to the Glossary in Appendix B, but to the classification introduced in Appendix A.

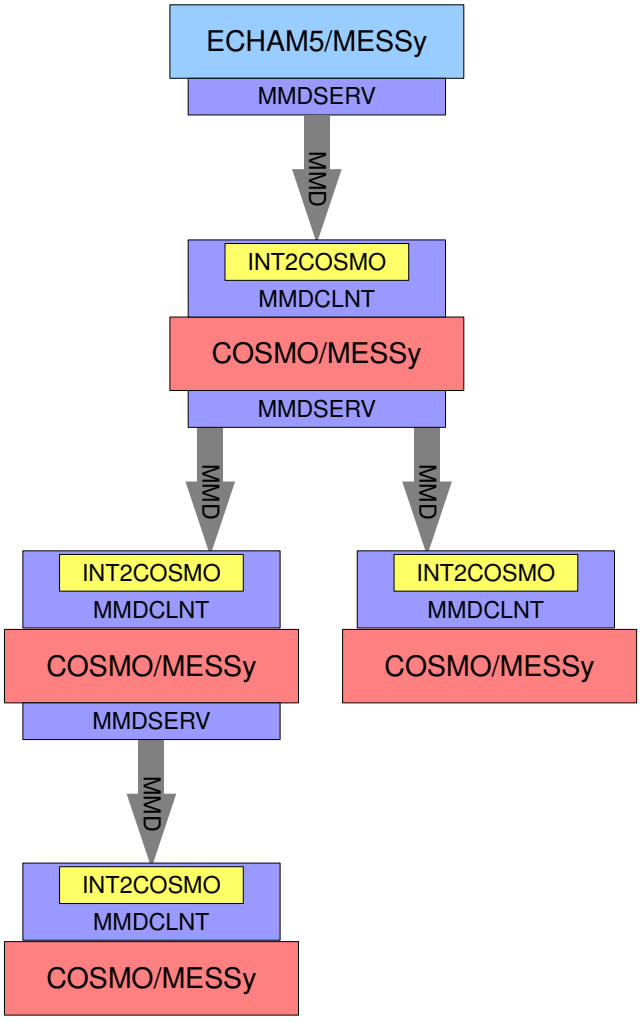

Fig. 4. Illustration of the data flow in an exemplary MECO(4) setup with ECHAM5/MESSy as *master Server* and four COSMO/MESSy clients.

it is immediately clear that the *internal coupling* is the most widely used approach, since it is inherent to all Earth system models. Jöckel et al. (2005) formalised this approach as *process coupling* by defining submodel implementations for process formulations (and diagnostics) and by defining an interface structure, which allows the strict separation of process implementations from shared model infrastructure (such as memory management, input/output, meta-data handling, call sequence etc.).

Examples for *internal process coupling* are the implementation of atmospheric chemistry and/or the treatment of aerosols into an atmosphere model, as e.g. in COSMO-ART (Vogel et al., 2009), or the regular submodels of the Modular Earth Submodel System (MESSy, Jöckel et al., 2005), which have been coupled to ECHAM5 (Jöckel et al., 2010) and COSMO (Kerkweg and Jöckel, 2012) using the MESSy infrastructure.

For coupling operators on the *domain* level, examples for both, *internal coupling* and *indirect external on-line coupling*, exist. The Earth System Modeling Framework (ESMF, Collins et al., 2005) and the Community Climate System Model version 4 (CCSM4, Gent et al., 2011) use *internal coupling* to allow for feedback between different *domain* models. Pozzer et al. (2011) coupled the ocean model MPIOM (Jungclaus et al., 2006) to ECHAM/MESSy, and compared results and run-time performance to the *indirectly, externally on-line coupled* ECHAM5/MPIOM system (Jungclaus et al., 2006), which uses the OASIS3 *coupler* (Valcke et al., 2006). Other examples for external couplers are CPL6 (Craig et al., 2005) as employed in the Community Climate System Model 3 (Collins et al., 2006) and OASIS4 (Redler et al., 2010).

With MMD, we here provide a method which, in our classification (Appendix A), provides a *direct external on-line coupling* method, i.e. avoiding the need of an additional external *coupler*. The client – server approach of MMD comprises two parts:

- The Multi-Model-Driver library provides a high-level API (application programming interface) for MPI based data exchange. The data transfer is implemented as single-sided, point-to-point communication, meaning that the data is sent from each server-task directly to the depending client tasks. The MMD library does not contain any rediscretisation facility. It is basemodel independent and straightforwardly applicable for the coupling of other models.

- The server and client submodels MMDSERV and MMDCLNT, respectively, are basemodel dependent and need to be adapted to newly coupled models. In particular the client model, which includes the model specific interpolation routines (here in form of INT2COSMO) requires this adaption. Nevertheless, the basic structure of MMDCLNT and MMDSERV are applicable to other models as well.

The client – server approach and the MMD library have some advantages in comparison to an *external coupler*: The direct point-to-point data exchange is highly efficient, because it does not require collective operations and minimises the memory consumption. This is specifically of importance, since our target application, an on-line nested global – regional – local atmospheric chemistry model, requires large amounts of data to be exchanged. Furthermore, our application requires a tailor-made, client-specific data transformation from the server to the client grid. The required interpolation routines existed (INT2LM), already parallelised (distributed memory) and could be easily recycled as INT2COSMO. This implies that the client, which "knows best" its own grid-structure, performs its own grid-transformations. The gain is a high consistency with the existing off-line nesting approach and avoids the (presumably

computationally more expensive) generalised transformation routines of a universal *coupler*.

5 Running the on-line coupled model system

Following the MESSy philosophy, setting up an on-line nested model cascade is as user-friendly as possible. In addition to the usual setup of each model instance (which remains the same as for a single instance setup), the user needs to edit only the run-script and one namelist file per client-server pair:

- In the run-script `xmessy_mmd` the model layout of the on-line coupled model cascade is defined. The user determines the number of model instances and the dependencies of the models, i.e. the client-server pairs. From this, the run-script generates the MMD namelist file `mmd_layout.nml` (see Sect. 6). Additionally, the file names and directories for the external data, required by INT2COSMO, have to be specified. A detailed explanation of the setup specific parts of the run-script are provided in the "MMD user manual" in the Supplement.

- The MMDCLNT namelist file `mmdclnt.nml` contains all information required for the data exchange, i.e. the time interval for the data exchange and the fields that are to be provided by the server to the client. Section 8.1 explains the meaning of the individual namelist entries.

6 The Multi-Model-Driver (MMD) library

The Multi-Model-Driver (MMD) library manages the data exchange between the different tasks of one ECHAM5/MESSy and/or an arbitrary number of COSMO/MESSy instances as illustrated in Fig. 4. The configuration of the client-server system is defined in the file `MMD_layout.nml` (which is written automatically by the run-script). This namelist contains the information about the number of models within one cascade, the number of MPI tasks assigned to each model and the definition of the server of the respective model (for further details see the "MMD library manual" in the Supplement).

The library contains a high-level API for the data exchange between the different models. Figure 3 illustrates the functional principle of the MMD library. During the initialisation phase, the exchange of information required by the server from the client model and vice versa, is accomplished by utilising the MPI routines `MPI_send` and `MPI_recv`. During the integration phase, data is exchanged only in one direction, i.e. from the server to the client. Point-to-point, single-sided, non-blocking communication is applied to exchange the required data. For longer simulations, a model interruption and *restart* is required to partition a simulation into subparts, fitting into the time limitation of a job scheduler on a super-computer. Therefore, one additional communication step occurs during the integration phase: for the synchronisation of the models w.r.t. such upcoming interrupts, the server has to send the information whether the simulation is interrupted after the current time step. This data exchange is implemented as direct MPI communication using `MPI_send` and `MPI_recv`.

As the routine `MPI_alloc_mem`, used to allocate the memory (buffer) required for the data exchange, can only be used in C (and not in Fortran95), some parts of the MMD library are written in C, however most parts are written in Fortran95 for consistency with the POINTER arithmetic used for the MESSy memory management (see Jöckel et al., 2010).

The MMD library routines and their usage are described in detail in the "MMD library manual" (see Supplement).

7 The server submodel MMDSERV

The server has to fulfil two tasks:

- it determines the date/time setting of the client models and

- it provides the data fields requested by the client.

In contrast to the client, which is associated to exactly one server model, a model can be server for an arbitrary number of clients. The number of clients of one server model is determined in the MMD namelist file (`MMD_layout.nml`, see Sect. 6). The right-hand side of Fig. 5 shows a simplified work flow for the MMDSERV submodel.

7.1 The initialisation phase

The server model receives information directly from the MMD library (read in from the MMD namelist file `MMD_layout.nml` about the overall simulation setup), and from its clients (i.e. client specific requirements). First, a server needs to know which models in the overall MMD setup are its clients. The information is acquired during the initialisation of the server specific MMD setup. In the MMDSERV subroutine `mmdserv_initialize`, the number of clients of this specific server is inquired and dependent variables are allocated accordingly.

The coupling to the client models is prepared within the MMDSERV subroutine `mmdserv_init_coupling`. For each client model the server passes the following procedure:

1. The server receives and stores the time interval (in seconds), in which data is requested by the client model and initialises a *coupling event*.

2. The server imposes its date and time settings (not the time step length!) on the client. Additionally, it sends its own time step length, to determine the time interval for the exchange of information about an upcoming model interruption (see Sect. 8).

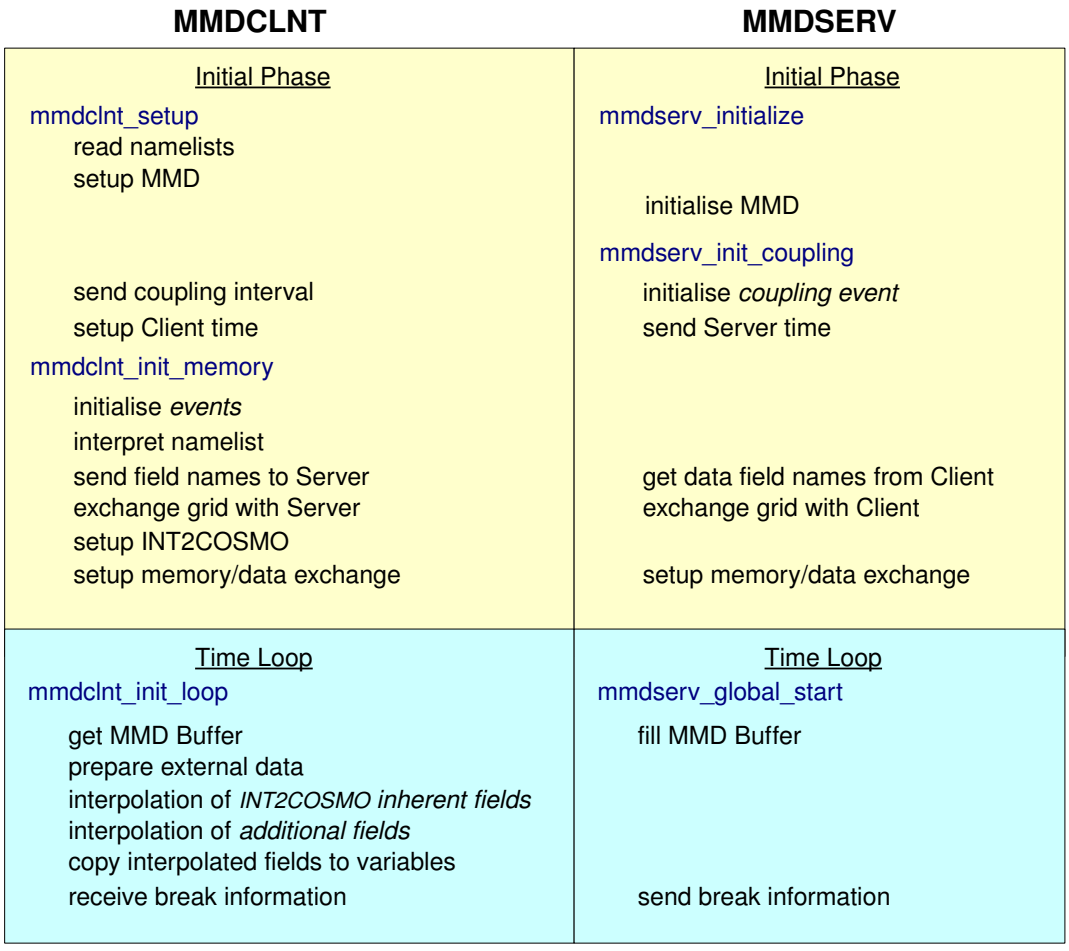

Fig. 5. Work flow of the MESSy submodels MMDCLNT and MMDSERV. The order of the subroutines corresponds to the calling sequence. Subroutines in the same row exchange information/data with each other.

3. The server side of the MMD library receives and stores the names of the fields requested from the client model.

4. The server receives the geographical coordinates of the client model grid points. Based on those, the server determines the server grid segment required by the client for the interpolation. It has to cover the entire client domain plus an additional frame. The corresponding server grid information is sent back to the client.

5. The server determines an index list used by the MMD library for the data exchange. To minimise the message passing traffic between the individual process entities (PEs[4]), each server PE provides exactly those data points required by the respective individual client PE. The calculation of the index list associating the individual grid points of the parallel decomposed server grid with the individual grid point in the *in-coming* coarse (also parallel decomposed) client *grid*[5] is explained in detail in the "MMD user manual" in the Supplement.

6. As last step of the initialisation, the server associates the POINTERs to the fields requested for the data exchange. First, the server receives the names of the fields from the MMD library (see step 3 above). If required, the server further acquires the *representation* (i.e. the geometric layout) of the fields and sends it to the client. Finally, the POINTERs are handed to the MMD library to access the requested data during the time integration phase.

7.2 Data exchange during the time integration phase

The MMDSERV submodel has to be called at the very beginning of the time loop to invoke the data exchange, i.e. it is called in `messy_global_start` (see the right lower, cyan part of Fig. 5). MMDSERV tests individually for each

[4]Here, PE is equivalent to an MPI task.

[5]This is the parallel decomposed grid on which the data of the server (*driving model*) is defined in INT2COSMO.

client, if the data exchange is desired within the current time step. In this case, new data is provided to the client by the MMD library subroutine `MMD_S_FillBuffer`. In this routine the MMD library uses the index list calculated during the initialisation phase (step 5 above) and the POINTERs to the data fields (step 6) to copy the required data into the memory buffer to be accessed by the respective client PE.

In addition, to synchronise the interrupt of all models of a model cascade, a server sends each time step the information to the client, whether the simulation is going to be interrupted after the current time step. Such an interruption, followed by a *restart*, is indispensable for longer simulations to fit the simulation into the time limitation dictated by the scheduling system.

8 The client submodel MMDCLNT

The MESSy submodel MMDCLNT manages the data acquisition, the data exchange with the server model, the successive interpolation and the supply of the interpolated data to the client model. MMDCLNT distinguishes three different destination types of data fields provided to the client model:

a. *initial fields*,

b. *boundary fields* and

c. *input fields*.

The stand-alone INT2LM handles *initial* and *boundary fields*. *Initial fields* are only required for the very first step of the simulation and used to initialise the client variables (e.g. temperature `T`, pressure deviation `pp`, water vapour `qv`, etc.). During the simulation the COSMO model is only supplied with *boundary fields*. They are copied to the boundary variables within the COSMO model (e.g. `t_bd`, `pp_bd` or `qv_bd`). MMDCLNT and INT2COSMO have been expanded to interpolate *additional fields*, i.e. fields that are not required by the stand-alone COSMO model, but by the MESSy submodels (e.g. tracers, emission flux fields etc.). The transfer and interpolation of *additional fields* up to rank 4 is possible (see Sect. 8.3.1). The stand-alone INT2LM processes only *boundary fields* for the COSMO model integration phase. But, for COSMO/MESSy it is desired to exchange also fields required for the entire model domain (in comparison to those prescribed at the lateral domain boundaries), e.g. emission flux fields, ozone climatologies etc. For the processing of these fields the third data destination type has been added, the so-called *input fields*. The *input fields* are interpolated in the same way as the *initial* and *boundary fields* and afterwards transferred to the respective *target* variables. This is performed in each coupling time step (see Sect. 8.3.2).

The number of fields to be provided by the server to the client and their destination type is flexible as the list of *exchanged* fields is determined by the MMDCLNT namelists in the namelist file `mmdclnt.nml` (see Sect. 8.1).

8.1 The MMDCLNT namelist

The namelists of the submodel MMDCLNT are a vital part of the entire coupling procedure. For each coupled instance, it consists of two parts:

– The `&CPL`-namelist determines the time interval for the data exchange from the server to the client.

– The server dependent CPL-namelists `&CPL_ECHAM` and `&CPL_COSMO` contain lists with the information about the fields, which need to be exchanged. These lists include the names of the *channel objects* in the memory management interface, information about the interpolation method and the destination of the field in the client model. Both namelists (`&CPL_ECHAM` and `&CPL_COSMO`) consist of two blocks: *mandatory fields* and *optional fields*. Mandatory are those fields, which are absolutely required for the COSMO basemodel[6] and/or are needed for the interpolation procedure itself. The variables required by the COSMO model depend on the COSMO model setup, thus the list of *mandatory fields* varies between different setups. *Optional fields* are mostly *additional fields*, i.e. fields not taken into account in INT2COSMO, but required by MESSy submodels. For *additional fields* the interpolation methods must be specified in the namelist. The horizontal interpolation method can be either "Q" (quadratic), "L" (linear) or "M" (match interpolation). For further information about the interpolation methods we refer to the INT2LM documentation[7]. Additional namelist parameters switch whether the field is interpolated also vertically, and whether monotonicity and/or positive definiteness are required for the interpolation result.

  Last but not least, the namelist entries determine the destination of the interpolated fields. We distinguish three destination types (see above): *initial*, *boundary* and/or *input* fields. *Mandatory fields* can be *initial* and *boundary fields*. For the *optional fields* the choice of *initial* and/or *boundary* and of *input* destination is exclusive, as *input* already implies *initial* and the provision of *boundary* data is meaningless, since the field is overwritten each coupling time step.

[6] There are also *optional fields* for the COSMO basemodel. For instance, as not all *driving models* provide the ice water content, this is not absolutely required as *in-field* for INT2LM. If the ice water is not available, INT2LM deduces the ice water content from the specific humidity and the temperature.

[7] http://www.cosmo-model.org/content/model/documentation/core/cosmoInt2lm.pdf

Examples and a detailed description of the namelists contained in the namelist file `mmdclnt.nml` are provided in the "MMD user manual" (see Supplement).

In addition to the coupling of standard 2-D and 3-D data fields, the coupling of 4-D data fields is implemented. They are treated exactly in the same way.

8.2 Initialisation phase

For MMDCLNT the initialisation phase is split into two subroutines. This is required, as the server model determines the timing (i.e. the start and the stop date and time) of the client model[8]. Since each server defines the timing of its clients, the coarsest model determines the timing of all coupled models. This coarsest model is hereafter called the *master server*.

8.2.1 mmdclnt_setup

The timing information is already required at the beginning of the initialisation phase of the basemodel. Thus, this data exchange proceeds at the first entry point of MESSy in COSMO (i.e. `messy_setup`). Figure 5 (left) sketches the procedure.

- Read namelist: first, the two namelists `&CPL` and (dependent on the server model) `&CPL_ECHAM` or `&CPL_COSMO` are read (see Sect. 8.1).
- Setup MMD: second, MMDCLNT and the client side of the MMD library are set up by defining client specific variables, the MPI-communicators required for data exchange and MMD internal initialisations. After initialising MMD, data can be exchanged between the server and the client.
- Setup date/time: third, the client and the server are synchronised. To achieve this, the client sends the time interval (in seconds) for the data exchange (from the server to the client) to the server. Next, the client receives the date settings from the server[9]. Based on these dates, the COSMO model and TIMER submodel time variables are redefined. Additionally, the client receives the time step length of the server. It is used to ensure the synchronised interrupt of the entire model cascade. Otherwise, one of the models would end up in a dead-lock during MPI communication. An interruption (and *restart*) of the model cascade is desirable for simulations exceeding the available time limits as defined by the scheduler of a super-computer. Thus the client model defines an *event*, which is triggered each server time step to exchange the information, whether or not the server model will be interrupted after the current time step, the so-called *break event*.

[8]The term "timing" does not include the model time step length!

[9]For the definition of the dates we refer to the "TIMER user manual", which is part of the Supplement of Jöckel et al. (2010).

8.2.2 mmdclnt_init_memory

Depending on the data destination (*initial*, *boundary* or *input*) of the *coupling fields*, memory needs to be allocated during the initialisation phase. Therefore, the second part of the initialisation is performed in `mmdclnt_init_memory`. Since the coupling procedure requires the presence of all other *channel objects* required for the coupling, `mmdclnt_init_memory` is called last within the entry point `messy_init_memory`. The lower left part of the yellow box in Fig. 5 illustrates the work flow in `mmdclnt_init_memory`.

- Initialise events: at the beginning of `mmdclnt_init_memory` the *coupling event* and the *break event* are initialised, as now the TIMER is fully set up and the *event* manager is available.
- Interpret namelist: because wildcards can be used in the namelists for the client *channel object* names, these namelist entries have to be translated into individual *exchange fields* in the subroutine `interpret_namelist`. Furthermore, the namelist entries are compared to the COSMO model variables `yvarini` and `yvarbd`, to ensure that the COSMO fields required by the basemodel setup are provided by MMDCLNT.
- Send field names to server: information from the namelist, required by MMD and the server, i.e. the *channel* and *channel object* names of each data field in the client and the server model (and their *representation*), are stored in an MMD library internal list. Those parts of the list required by the server are sent to the MMD library part accessed by the server.
- Exchange grid with server: afterwards, information about the grids are exchanged between client and server. First, the client sends two 2-D-fields containing the geographical longitudes and latitudes of the client grid. From this information and the definition of the server grid, the server calculates the required dimensions of the server grid section, which is transferred to the client. INT2COSMO needs a segment of the coarse grid, which covers the complete COSMO model grid plus some additional space required for the interpolation to the finer grid.

  The server sends back the complete definition of the *incoming grid*, which, in the stand-alone INT2LM, is defined in the `&GRID_IN` namelist.
- Setup INT2COSMO: with this information INT2COSMO in MMDCLNT is set up by calling the routine `setup_int2lm` (as it is used in the stand-alone INT2LM). The subroutine `setup_int2lm` is processed in nearly the same way as in the stand-alone INT2LM. However, those subroutines called in

`setup_int2lm` dealing with the decomposition of the model domain and the parallel environment are skipped or replaced. Additionally, the routines reading the coarse grid data are omitted[10].

– Setup memory/data exchange: the next step is the calculation of an index list directly mapping each grid point of the parallel decomposed *in-coming grid* of the client to the respective grid point in the parallel decomposed domain of the server. The calculation is performed by the server. It requires the geographical coordinates of the client grid separately for each client PE as input (see also Sect. 7). The index list provides the basis for the efficient data transfer by the MMD library. It enables point-to-point data exchange from one server PE to one client PE, thus avoiding the gathering of the server fields and the scattering of the client *in-fields* before and after the exchange, respectively. One of the most important features of this implementation of the coupling between two models is its flexibility combined with the possibility to use the INT2LM interpolation routines as they are[11]. For all *exchange fields* listed in the namelist file of MMDCLNT, the data is processed automatically. For this a Fortran-95 structure was defined, containing pointers to all fields (*input*, *intermediate* and *target*) used in MMDCLNT-INT2COSMO.

  A detailed description of the memory allocation procedure is available in the "MMD user manual" (see Supplement).

  In the last step of the initialisation phase, the POINTERs associated to the *in-fields* are passed to the MMD library. Together with additional information about the dimensions of the fields and the index list determined before, within the MMD library the size of the exchange buffer is calculated and the buffer is allocated subsequently.

8.3 The integration phase

The data exchange with the server takes place periodically within the time loop. As it provides the new *input* and/or *boundary fields*, this happens as early as possible within the time loop, i.e. in `messy_init_loop`[12] for MMDCLNT. In contrast to MMDCLNT, some fields need to be set in the server model before providing the data to the client. Thus for MMDSERV, the exchange is called at the end of `messy_global_start`. This is also important to avoid an MPI communication dead-lock: due to client and server dependencies `mmdclnt_init_loop` must always be called before `mmdserv_global_start` within the same basemodel.

First, MMDCLNT checks, if data exchange is requested in the current time step. If this is the case, the client acquires the data from the server by calling the MMD library subroutine `MMD_C_GetBuffer`. After this call, new data is assigned to all *in-fields*. Subsequently, the interpolation takes place (Sect. 8.3.1), after which the interpolated fields are copied (Sect. 8.3.2) to the target variables.

At the end of the subroutine `mmdclnt_init_loop`, when the current model time step coincides with a server time step, the information whether the simulation is interrupted after the current server time step is received. This information exchange is independent of the coupling interval and required to avoid an MPI communication dead-lock, caused by an interruption in the server model without informing the client models beforehand.

8.3.1 Interpolation via INT2COSMO

The interpolation applied in MMDCLNT(-INT2COSMO) is based on the stand-alone program INT2LM as provided by the German Weather Service (DWD) for the interpolation of coarse grid model data to initial and boundary data required by the COSMO model[7]. For the on-line coupling of the COSMO model to a coarse grid model (ECHAM5 or COSMO) as described here, it is necessary to perform the interpolation of the coarse grid data to the smaller scale COSMO model during the integration phase i.e. integrated into the basemodel itself. Therefore, INT2LM is implemented as sub-submodel INT2COSMO into the MESSy submodel MMDCLNT. The interpolation in MMDCLNT follows the order of the stand-alone INT2LM program.

First, the external data are prepared[13].

The subroutine `external_data` in INT2COSMO comprises three sections:

[10] All modifications and extensions in the INT2LM and the COSMO model code, which became necessary in the scope of the implementation of the on-line coupling are documented in the "MMD user manual" in the Supplement. The changes in the code are all enclosed in pre-processor directives.

[11] Thus, always the latest version of INT2LM can be used within COSMO/MESSy. For instance, a newly introduced interpolation technique in INT2LM is directly available for the on-line coupling.

[12] There is one exception: at the start of a simulation the data is exchanged in `mmdclnt_init_memory`, because the *initial fields* are required already in the initialisation phase of a model simulation. The call in `messy_init_loop` is therefore skipped in the very first time step (`lstart=.TRUE.`).

[13] The term external data refers to all data provided to the model from extern. On the one hand these are the – more or less– constant fields, the so-called external parameters required for the COSMO model and the *driving model* grid, and on the other hand data fields provided by the *driving model*.

a. input of the external parameters needed by the COSMO model, external parameters are e.g. the orography, the leaf area index, the root depth or the land-sea fraction,

b. import of the external parameters of the *driving model*, and

c. definition of the internal setup and pre-calculation of variables required for the interpolation. Depending on the setup and on the fields provided by the *driving model*, missing fields are calculated from other available fields.

The external parameters defined on the COSMO grid (item a) are usually constant in time. Thus they are read only during start or *restart* of a simulation. The import of the external parameters of the *driving model* (item b) is replaced by the data exchange via MMD. Usually during the import, logicals are set indicating, which fields are provided by the *driving model* and which have to be calculated. As the import procedure is skipped in the on-line coupled setup, these switches are set within MMDCLNT according to the data sent via MMD instead. The last section (item c) is processed as in the stand-alone version.

The *INT2LM inherent fields* are interpolated first, calling the original interpolation routine `org_coarse_interpol` of INT2LM for the horizontal interpolation. The vertical interpolation of the *INT2LM inherent fields* is accomplished by the same subroutines as in the stand-alone version.

Afterwards, the *additional fields* are interpolated in the same way. First, each vertical layer (and each number dimension) of a field is horizontally interpolated according to the interpolation type chosen in the namelist. This takes also into account the settings for monotonicity and positive definiteness (see Sect. 8.1). Second, the field is interpolated vertically, if requested.

8.3.2 Data transfer to COSMO variables

After finishing all interpolations the resulting *intermediate fields* are copied to the *target fields*. MMDCLNT distinguishes three destination types for the data (see introduction to Sect. 8):

a. *initial fields*: These are only required for the initialisation of the COSMO model,

b./c. *boundary* and *input fields*: These are updated periodically during the model integration.

As fields of destination type (a) are only copied in the initialisation phase, two independent subroutines perform the data transfer for data of type (a) and (b/c).

Moreover, there are two kinds of initial data:

a1. scalar variables defining the vertical model grid and the reference atmosphere of the COSMO model:
In the stand-alone INT2LM and COSMO model the vertical grid and the reference atmosphere are defined by namelist settings in INT2LM. The resulting variables are dumped into the initial file and the COSMO model reads its grid and the reference atmosphere definitions. In case of the on-line coupling, these variables are also defined by INT2COSMO namelist settings, but as COSMO does not read any file for input anymore, these variables also have to be transferred to the respective COSMO variables[14].

a2. 2-D-, 3-D- or 4-D-fields for the initialisation:
For these fields the contents of the *intermediate field* are copied to the *target* variable.

The subroutine `move_initial_arrays_to_COSMO` copies both types of initial data to their counterparts of the COSMO/MESSy model.

During the integration phase two data destinations are distinguished:

b. the *boundary fields* for prognostic variables;
c. the *input fields*.

The most important difference between the on-line and the off-line coupling of the models is evident in the treatment of the boundary data. In the off-line setup boundary data are typically available for discrete time intervals (e.g. 6 hourly). The data at the beginning and the end of this time interval are read and the current boundary data in each time step are linearly interpolated between these two. The on-line coupling works differently. To permit the same implementation as in the off-line mode, the server model would have to be one coupling time interval ahead. This would be possible for the 1-way-coupling. But the ultimate goal of our model developments is the implementation of a 2-way-nesting. For this, the server model must not be ahead of the client model, otherwise the feedback to the larger scale model would not be possible. For simplicity, the two time layers of the *boundary* data are filled with the same value. As the on-line coupling allows for much higher coupling frequencies, no further interpolation in time is required.

9 Implementation details

This section conveys some important details about the technical implementation itself.

9.1 Changes in the original codes of COSMO, INT2COSMO and ECHAM5/MESSy

All changes in the original COSMO and INT2LM code have been introduced with pre-processor directives. As different

[14]The variables in this category are `vcflat`, `p0sl`, `t0sl`, `dt0lp`, `nfltvc`, `svc1`, `svc2`, `ivctype`, `irefatm`, `delta_t` and `h_scal`.

model configurations are possible, three pre-processor directives have been introduced:

- `MESSY`
- `I2CINC`
- `MESSYMMD`

The directive `MESSY` is used for the implementation of the MESSy interface into the COSMO model as described in Kerkweg and Jöckel (2012). According to Jöckel et al. (2005), all MESSy specific entry points in the COSMO model are encapsulated in

```
#ifdef MESSY
 CALL messy_...
#endif
```

directives. Those parts of the COSMO model, which become obsolete by using the MESSy interface are enclosed in

```
#ifndef MESSY
...
#endif
```

directives. Thus, it is always possible to use the original stand-alone COSMO model by simply not defining the pre-processor directive `MESSY` for the compilation.

The directive `I2CINC` (**I**NT**2C**OSMO **IN C**OSMO) is used for the modifications of the code required for the implementation of INT2COSMO as MESSy sub-submodel. As the COSMO model and INT2LM contain many redundant code parts, most changes in INT2COSMO exclude redundant code and variable definitions. The only changes in the COSMO code occur within the file `src_input.f90`, where the reading of the initial and boundary files is omitted.

The directive `MESSYMMD` indicates that the MMD library is used. In this case, more than one model instance runs concurrently within the same MPI environment. Therefore, the MPI-communicators used in each basemodel must be modified.

`MESSYMMD` and `I2CINC` are two completely independent directives. In future the MMD library might be used to couple other models than ECHAM5 and COSMO. Thus, the directive `MESSYMMD` does not imply that `I2CINC` must also be defined. Vice versa, INT2COSMO in COSMO (and thus `I2CINC`) without defining `MESSYMMD` will be applicable in future to include the possibility to drive COSMO/MESSy off-line, directly with ECHAM5/MESSy or COSMO/MESSy output. Instead of receiving the data on-line from MMD, files containing the required data from the coarser model are imported and interpolated on-line by INT2COSMO.

9.2 Implementation of INT2LM as MESSy sub-submodel MMDCLNT-INT2COSMO

For the on-line coupling method described here, the interpolation of the coarse grid data to the smaller scale COSMO model is performed during the time integration. Therefore INT2LM is implemented as sub-submodel MMDCLNT-INT2COSMO into the MESSy submodel MMDCLNT. Consequently, INT2COSMO co-exists within the COSMO/MESSy model itself.

Many subroutines and tools are part of both, the INT2LM code as well as of the COSMO model code. Those subroutines available in both models are used from the COSMO code for both model parts. Technically, INT2LM was included into the MESSy code distribution as stand-alone basemodel INT2COSMO within the MESSy basemodel directory (mbm). For the inclusion in COSMO/MESSy a directory parallel to the source code directory of COSMO (`cosmo/src`) called `cosmo/src_i2c` was created, which contains links to those source code files of the mbm model INT2COSMO, which are required in the MMDCLNT submodel INT2COSMO.

To reduce the MPI communication overhead resulting from the different parallel decompositions of the INT2COSMO and the COSMO model grid, the core regions of the COSMO and the INT2COSMO grid have to be congruent[15]. The "MMD user manual" (see Supplement) contains a detailed explanation of the procedure used to adjust the parallel decomposition of the INT2COSMO grids to the parallel decomposition of the COSMO model grids.

At the time being, the only server models available within the MESSy system are the ECHAM5/MESSy and the COSMO/MESSy model. Therefore, the module files of INT2LM, which are only relevant for other *driving models*, are unused in INT2COSMO so far, but can be easily activated if required.

More details about the implementation of INT2COSMO into MMDCLNT are provided in the "MMD user manual" in the Supplement.

10 Remarks on optimising the run-time performance efficiency

The main challenge in efficiently using the available computer resources, (the CPU time) with MECO(n), is to find the optimum distribution of MPI tasks among the participating model instances. This optimum distribution depends on the chosen model setups, the model resolutions, the number of instances etc. and usually underlies further constraints.

[15]This is not trivial, as the "inner" INT2COSMO grid is the "outer" COSMO grid. Thus, using the standard routines calculating the parallel decomposition of the model grid for both parts independently would result in a shift between the model domains across the MPI tasks.

Table 1. Fraction of MPI tasks for the different MECO(2) instances estimated for a maximum efficiency from the number of grid boxes and time step lengths.

Instance	# grid boxes		time step	fraction of
	horizontal	vertical	(seconds)	tasks
ECHAM/MESSy-T106L31	360×160	31	360	0.03
COSMO-40/MESSy	122×122	40	120	0.03
COSMO-7/MESSy	421×381	40	40	0.94

Such constraints can be both, intrinsic to the models (e.g. if the number of usable tasks has a resolution dependent upper limit caused by the chosen parallel decomposition), or posed by the used computer system (e.g. if only specific task numbers (e.g. full nodes) are selectable). As a consequence, a general statement on the optimum distribution of tasks is hardly possible.

To facilitate the optimisation process, both MMD submodels measure on-line the waiting times between the corresponding requests and the time they actually read (MMDCLNT) and wrote (MMDSERV) the MMD exchange buffer. These times are provided as *channel objects* and therefore (optionally) subject to output. These times in combination with the on-line measurement of wall-clock and CPU-times of the submodel QTIMER (Jöckel et al., 2010) provide valuable information to assess the run-time performance efficiency.

With an example, we provide a general guideline. The chosen MECO(2) setup is the one used by Kerkweg and Jöckel (2012) and Hofmann et al. (2012), i.e. two COSMO/MESSy instances nested into each other are nested into the global ECHAM/MESSy. The COSMO/MESSy domains are the bluish coloured domains over Europe in Fig. 1.

To start with a first guess, we make the crude assumption that each grid box in each model instance at each time step requires the same computing time. To achieve an optimal load balance between the model instances, this defines the relative distribution (fraction) of tasks among the model instances, so far without further constraints. For the chosen example the numbers are listed in Table 1, meaning that, under the assumption above, load balance is achieved, if out of one hundred tasks, three are assigned to ECHAM/MESSy, three to the coarse COSMO-40/MESSy instance, and 94 to the finer COSMO-7/MESSy instance.

Next, we assess the constraints exposed by our computer system, the IBM Power6 system "blizzard" at the "Deutsches Klimarechenzentrum", Hamburg (DKRZ, http://www.dkrz.de/). This computer has 16 dual core processors per node and we can only use complete nodes, i.e. multiples of 32 tasks[16]. According to Table 1, we assign ECHAM/MESSy and COSMO-40/MESSy 4 tasks each, and hereafter assess the run-time performance efficiency for four different setups, namely with 24, 56, 88 and 120 tasks for COSMO-7/MESSy (i.e. using in total between 1 and 4 nodes with 32 CPUs each). The resulting waiting times are displayed in Fig. 6. A perfect load balance and synchronisation between the different model instances would result in zero waiting times. This, however, is virtually impossible to be achieved, since the required computing time per model time step varies depending on the model setup. For instance, radiation is not calculated every time step and output is only triggered at distinct time intervals (hourly in the example here).

As Fig. 6 shows, the COSMO-7/MESSy client is much too slow in the 4-4-24 setup, as both servers wait, while the clients do not. In the 4-4-56 setup the server waiting times are reduced by a factor of approximately 2 (implying a nearly linear scaling) and waiting times for the coarser client emerge. For the 4-4-88 setup, both clients experience idle times regularly exceeding the server waiting time, except at full hours, when the output is written. For the 4-4-120 setup the maximum waiting times of server and clients are similar. But, because the servers run with less tasks, the overall idle time in this last setup is longer as in the other setups. Figure 7 displays for the four setups the average waiting time per task calculated according to

$$\bar{T}_{\mathrm{w}} = \frac{1}{\sum_{i=1}^{3} n_i} \left(\sum_{s=1}^{2} T_{\mathrm{w},s} n_s + \sum_{c=2}^{3} T_{\mathrm{w},c} n_c \right) \quad (1)$$

where n is the number of tasks used for each model instance (index i), distinguished for the servers (index s) and for the clients (index c), respectively and T_{w} are the individual waiting times. While the 4-4-24 (black line) setup produces relatively constant idle times of 3 seconds on average per task, the average waiting times in the setups with more tasks for the COSMO-7/MESSy client (4-4-88 and 4-4-120, green and blue lines, respectively) peak to 4 or 7 seconds, respectively, with smaller waiting times in between these peaks. The best balanced setup is the 4-4-56 setup (red line) as it produces the shortest average waiting times.

This best balanced setup is also the most efficient, as is confirmed by another measure, based on the on-line QTIMER recording of wall-clock and consumed CPU-times

[16] For simplicity, we do not use the possible Simultaneous Multithreading (SMT) Mode with 64 tasks per node here and only assign one task per core.

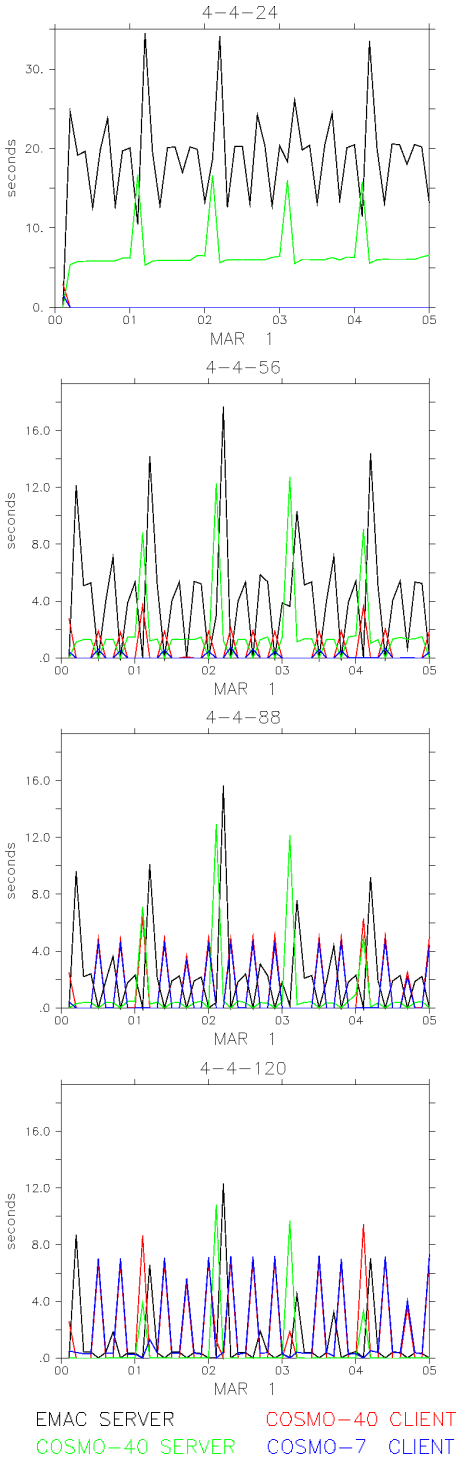

Fig. 6. Waiting times of the server and clients required for the four task distributions 4-4-24, 4-4-56, 4-4-88 and 4-4-120. The black and the green lines denote the time the ECHAM/MESSy and the COSMO/MESSy server tasks wait, respectively. The red and the blue lines indicate the waiting times of the COSMO-40/MESSy and the COSMO-7/MESSy clients, respectively.

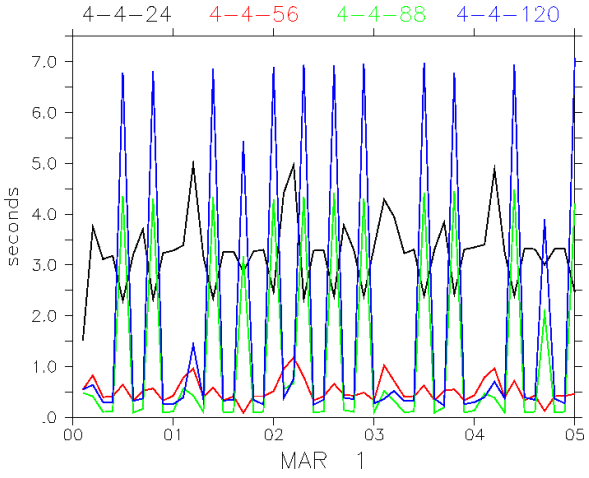

Fig. 7. Average waiting time per task (according to Eq. 1) for the different setups (black: 4-4-24, red: 4-4-56, green: 4-4-88 and 4-4-120: blue).

as shown in Table 2. The table lists the average wall-clock times (in seconds) per simulated hour and the corresponding consumed CPU-time (in hours). As to be expected, as long as the communication overhead (including the waiting times) does not outperform the calculation effort, more tasks result in shorter wall-clock times. However, the speedup for the assessed MECO(2) cascade slows down considerably for the 4-4-120 setup, implying that no further speedup can be expected, if only the number of tasks for the COSMO-7/MESSy instance is further increased. As Table 2 further shows, the best efficiency is achieved with the 4-4-56 setup, because the costs in terms of used CPU-time per simulated hour show a minimum. This is the setup, for which we deduced above that it accumulates the least idle times and is therefore best balanced.

As a result, our first guess based on the crude assumption, (see Table 1) is not the best (i.e. most efficient) choice, because we need to take into account the accumulated waiting times. As this example shows, the optimum distribution of tasks cannot be estimated straightforwardly. However, the MMD submodels MMDCLNT and MMDSERV are equipped with internal watches to measure their waiting times for the data exchange between the model instances. In combination with the MESSy submodel QTIMER this provides a convenient way for empirically determining the optimum choice with only a few short model simulations in the desired MECO(n) setup.

11 Summary and outlook

The newly developed 1-way on-line coupled model system ECHAM5/MESSy ($\rightarrow$ COSMO/MESSy)^n is presented. For easier reference to the on-line coupled system in future applications we refer to the entire MESSy model

Table 2. Average wall-clock (seconds) and corresponding CPU-times (hours) per simulated hour for the different setups. The CPU-times are the product of the wall-clock times and the number of used cores. In our setup the number of cores equals the number of MPI tasks.

Setup Setup	wall-clock (s)	CPU-time (h)
4-4-24	280.0	2.49
4-4-56	133.6	2.38
4-4-88	114.3	3.05
4-4-120	102.8	3.77

cascade as MECO(n) (**M**ESSy-fied **E**CHAM and **CO**SMO nested **n** times). Hence, the example in Fig. 1 shows a MECO(12) setup, i.e. 12 COSMO/MESSy instances nested into ECHAM5/MESSy. To further visualise the structure of the nested instances "n" (here $n = 12$) can be written as the sum of the individual cascades. For the example shown in Fig. 1 one can write MECO$(2+2(1+1)+2(1+2))$ as two single COSMO/MESSy ("2"), 2 COSMO/MESSy instances with one further nest ("$2(1+1)$") and 2 COSMO/MESSy instances with 2 nests on the next nesting level ("$2(1+2)$") are nested into ECHAM5/MESSy. Further examples of MECO(2) model setups are presented in the companion articles (Kerkweg and Jöckel, 2012; Hofmann et al., 2012).

While the regional atmospheric chemistry model COSMO/MESSy (Kerkweg and Jöckel, 2012) was developed to provide a limited-area model for atmospheric chemistry applications, the main goal for the development of MECO(n) is to provide consistent dynamical and chemical boundary conditions to a regional chemistry model in a numerically efficient way and thus providing a zooming capability. Chemical calculations demand a much higher number of exchanged fields (all tracers except the very short lived species need boundary conditions) and a higher frequency for the provision of the boundary data (necessary to capture the diurnal variations of chemical compounds). Therefore, an off-line coupling (as usually performed for the mere meteorological simulations) becomes impracticable due to the enormous amount of data and numerous pre-processing steps. Thus, we couple the models on-line in a client-server approach: the server model provides the input data via MPI based communication during the integration phase and the client model interpolates these data to get its boundary and input data. Hence, with this new approach

- no additional disk storage is required to keep the files containing the preprocessed initial and boundary data available during the regional model simulation,
- a higher frequency for boundary data provision becomes possible.
- the data exchange is faster, because data exchange via memory is much faster compared to disk input/output,
- no stand-alone interpolation program needs to be run for each time step, for which boundary data is provided to the regional model, i.e. no more pre-processing steps are required, that could only be performed sequentially, whereas in our approach the models run concurrently,
- the input and boundary data files, as used by the stand-alone COSMO model for the meteorological variables, are no more needed in the on-line coupled setup. This leads to a further reduction of the required disk storage,
- a prerequisite for two-way nesting, i.e. feedback from the smaller to the larger scales is fulfilled.

Thus, much less disk storage is required for an on-line coupled simulation and manual data processing to produce the boundary files is largely reduced.

On the other hand, the on-line coupled setup requires more computing power at once compared to the stand-alone setup, as all models run concurrently in the same MPI environment. Nevertheless, nowadays at super-computing centres, it is easier to access a large number of computing cores, than large amounts of permanently available disk storage.

The on-line data exchange is managed by the newly developed Multi-Model-Driver (MMD) library and two corresponding MESSy submodels, MMDCLNT and MMDSERV. During the initialisation phase the models communicate via MMD using `MPI_send` or `MPI_recv` and the group communicators defined for one client-server pair. During the integration phase, the exchange of the data fields required by the client is coded as point-to-point, single-sided, non-blocking MPI communication: the server fills a buffer and continues its integration, while the client reads the data stored by the server.

The MMD library could be used to couple other models as well, but the interfaces to the ECHAM/MESSy and the COSMO/MESSy models (i.e. the submodels MMDSERV and MMDCLNT) are rather specific for MECO(n) and need to be adapted; in particular, as MMDCLNT includes INT2COSMO, which is tailor made for grid-transformations to the COSMO model grid. Nevertheless, the adaption is straightforward due to the standardised MESSy infrastructure.

The partitioning of the available MPI tasks between the model instances for an efficient usage of resources is left to the user, since its optimum strongly depends on the individual model setups. To facilitate this task, the MMD submodels are equipped with internal stopwatches to measure the performance. With this information, the optimisation procedure as shown, requires only a few short simulations with different task distributions.

In a next step we plan to extend the 1-way on-line coupled model system ECHAM5/MESSy$(\rightarrow$ COSMO/MESSy$)^n$ into a two-way nested atmospheric chemistry model system.

Finally, we emphasise that even though the technical structure looks complicated at the first glance, the user only needs to edit the run-script and the MMDCLNT namelist files (one per client instance) to run MECO(n).

Appendix A

Classification of different coupling approaches

In the Earth system modelling community the terms *coupling*, *coupler* and *coupled* are widely used to describe a variety of aspects of connections within an Earth System Model (ESM), though rarely precised. In general, these terms are used to express some way of how different *components* influence each other. For instance the basic equations describing the system form a set of *coupled* differential equations, however, these equations are commonly not regarded as *components* of the model.

This intrinsic level of *coupling* is complemented by a second level of *coupling*, which arises from the (numerical) necessity to formulate and implement complex ESMs in an *operator* splitting approach: the different *operators* need to be *coupled* together to form an integral model: one operator after each other modifies the state variables of the system. Technically, the simplest *operator* is formed by a subroutine, which calculates a specific output for the provided input. The granularity of this decomposition into operators can be formalised in a way that a set of subroutines describes a specific natural *process* of the system (which in MESSy is called a submodel, Jöckel et al., 2005). Such a *process* can still be regarded as an *operator*. Even further, a set of *processes* describing a specific *domain* of the Earth system, such as the atmosphere or the ocean subsystem, can still be regarded as an *operator*.

The *coupling* then describes, how these operators communicate which each other. Two operators are *internally coupled*, if they are part of the same program unit, i.e. the same executable, and exchange information *on-line*, i.e. via the working memory. In contrast to that, operators are *externally coupled*, if they are split into different program units, i.e. different executables. This implies that information to be exchanged between the operators necessarily need to leave the program unit bounds and are either exchanged *off-line* via storage and import to/from an external (e.g. disk-based) file system, or *on-line* via the working memory. Whereas *off-line external coupling* is computationally inefficient due to the low input/output bandwidths and hardly allows a two-way data exchange, *on-line external coupling* requires special software and/or a model infrastructure for the communication between two (or more) executables (such as for instance the message passing interface, MPI).

The *external on-line coupling* approach can be further differentiated into the *indirect external on-line coupling*, in which two different program units communicate via an additional intermediate program unit (meaning that another executable is required, which is not identical to the message passing environment process) and into the *direct external on-line coupling*, where the two program units exchange their information without such an additional intermediate program unit (e.g. within the same message passing environment). The intermediate program unit performing the data exchange in the *indirect external on-line coupling* approach is commonly called a *coupler* and, in most implementations, does not only perform the data exchange (including transpositions) but contains further services, such as for instance rediscretisations, i.e. transformations of data from one model grid to another.

Appendix B

Glossary

- *additional field*: an *additional field* is a field requested in the MMDCLNT namelist in addition to the fields already taken into account within INT2COSMO.

- *boundary field*: it is used to prescribe the variables at the model domain boundaries.

- *break event*: the *break event* is an *event* that is triggered each server time step in order to receive the information from the server, whether the server model is going to be interrupted after the current time step or not.

- *channel*: the generic submodel CHANNEL manages the memory and meta-data and provides a data transfer and export interface (Jöckel et al., 2010). A *channel* represents sets of "related" *channel objects* with additional meta information. The "relation" can be, for instance, the simple fact that the *channel objects* are defined by the same submodel.

- *channel object*: it represents a data field including its meta information and its underlying geometric structure (*representation*), e.g. the 3-dimensional vorticity in spectral *representation*, the ozone mixing ratio in Eulerian *representation*, the pressure altitude of trajectories in Lagrangian *representation*.

- *coupling event*: this is an *event* scheduling the data exchange from the server to the client. Its time interval has to be a common multiple of the client and the server time step length.

- *coupling field*: a *coupling field* is either an *exchange field* or a field required by the client model that is calculated during the interpolation procedure in INT2COSMO, i.e. the fields deduced from the external parameters, e.g. `lai`, `rootdp`, etc.

– *dimensions*: they represent the basic geometry, e.g. the number of latitude points, the number of trajectories, etc.

– *driving model*: the coarse grid model (= server) that provides the *in-fields* to INT2LM/INT2COSMO.

– *event*: This is a data type provided by the generic submodel TIMER, which is used to schedule processes at specific (regular) time intervals, e.g. to trigger regular output or input during a simulation. The *event* control is part of the MESSy generic submodel TIMER. The Supplement of Jöckel et al. (2010) comprises a manual for TIMER and details about the *event* definition.

– *exchange field*: an *exchange field* is a field requested within the `mmdclnt.nml` namelist file and provided by the server to the client. An *exchange field* can either be a field which is interpolated and copied to a client variable, or a field required for the interpolation itself.

– *in-coming grid*: the *in-coming grid* is the grid on which the *in-fields* are defined, i.e. a subpart or the full server grid.

– *in-field*: the *in-fields* are those fields provided by the server or *driving model*, which are still defined on the server grid, but on the client side. In other words, *in-fields* are the *exchanged fields* before the interpolation.

– *INT2LM inherent field*: this is a field which is considered and interpolated within INT2COSMO or INT2LM (it is part of the variable table in INT2LM).

– *initial fields*: one destination type of data field provided by MMDCLNT to the client model. *Initial fields* are only used to initialise variables at the very beginning of the simulation.

– *input fields*: one destination type of data field provided by MMDCLNT to the client model. *Input fields* are *additional fields*. The newly interpolated field replaces the field in the client model, e.g. an emission flux field, that is down-scaled from the server.

– *intermediate field*: the *intermediate field* is the "working space" of INT2COSMO. It contains the fields after horizontal and/or vertical interpolation. Its vertical dimension is the maximum of the level numbers of the client and the server grid.

– *mandatory field*: this is an *in-field* absolutely required either by the COSMO model setup, or for the interpolation itself.

– *master server*: The coarsest model in the model cascade is called the *master Server*. It determines the time settings of all other model instances.

– *optional field*: this is an optional *in-field*. It can be either an *additional field* or an *INT2COSMO inherent field* not absolutely required by the COSMO basemodel or for the interpolation.

– *representation*: it describes multidimensional geometric structures (based on *dimensions*), e.g. Eulerian (or grid point), spectral, Lagrangian.

– *restart*: a *restart* is performed, if the computing time allowed by a scheduler of a super-computer is too short for the complete simulation. In this case, the simulation is interrupted in between and restarted in a new job. To achieve binary identical results for simulations with and without interruption, restart files are written, of which the contents fully determine the state of a model simulation. These files are read in the initialisation phase during a model *restart*.

– *target field*: This term specifies those fields, on which the results of INT2COSMO are written, i.e. the variables used in the COSMO/MESSy simulation itself.

Supplementary material related to this article is available online at: http://www.geosci-model-dev.net/5/111/2012/gmd-5-111-2012-supplement.zip.

Acknowledgements. This work was founded by the German Science Foundation (DFG) under the project name MACCHIATO (WE 2943/4-1). The authors thank the "application support for high computer performance" team of the MPG computer centre Garching, in particular I. Weidl, R. Hatzky, W. Nagel and H. Lederer. The development of MECO(n) was supported within the project CheSS by the DEISA (Distributed European Infrastructure for Supercomputer Applications) Extreme Computing Initiative (DECI): we therefore thank the DEISA Consortium (www.deisa.eu), co-funded through the EU FP6 project RI-031513 and the FP7 project RI-222919. We feel grateful to K. Ketelsen for setting up the C-code part and the basic outline of the Fortran part of the MMD library. Many thanks to J. Lelieveld for supporting this model development effort, especially for the financial backup of K. Ketelsens work. Last but not least, we very much appreciate the fruitful discussions with our colleagues A. Baumgaertner and H. Tost.

Edited by: V. Grewe

References

Collins, N., Theurich, G., DeLuca, C., Suarez, M., Trayanov, A., Balaji, V., Li, P., Yang, W., Hill, C., and da Silva, A.: Design and Implementation of Components in the Earth System Modeling Framework, Int. J. High. Perform. C, 19, 341–350, 2005.

Collins, W. D., Bitz, C. M., Blackmon, M. L., Bonan, G. B., Bretherton, C. S., Carton, J. A., Chang, P., Doney, S. C., Hack,

J. J., Henderson, T. B., Kiehl, J. T., Large, W. G., McKenna, D. S., Santer, B. D., and Smith, R. D.: The Community Climate System Model Version 3 (CCSM3), J. Climate, 19, 2122–2143, doi:10.1175/JCLI3761.1, 2006.

Craig, A. P., Jacob, R., Kauffman, B., Bettge, T., Larson, J., Ong, E., Ding, C., and He, Y.: CPL6: The New Extensible, High Performance Parallel Coupler for the Community Climate System Model, Int. J. High. Perform. C, 19, 309–327, 2005.

Doms, G. and Schättler, U.: The nonhydrostatic limited-area model LM of DWD. Part 1: Scientific documentation, Deutscher Wetterdienst, Offenbach, available at: www.cosmo-model.org (last access: 20 June 2011), 1999.

Gent, P. R., Danabasoglu, G., Donner, L. J., Holland, M. M., Hunke, E. C., Jayne, S. R., Lawrence, D. M., Neale, R. B., Rasch, P. J., Vertenstein, M., Worley, P. H., Yang, Z.-L., and Zhang, M.: The Community Climate System Model Version 4, J. Climate, 24, 4973–4991, doi:10.1175/2011JCLI4083.1, 2011.

Hofmann, C., Kerkweg, A., Wernli, H., and Jöckel, P.: The 1-way on-line coupled atmospheric chemistry model system MECO(n) – Part 3: Meteorological evaluation of the on-line coupled system, Geosci. Model Dev., 5, 129–147, doi:10.5194/gmd-5-129-2012, 2012.

Jöckel, P., Sander, R., Kerkweg, A., Tost, H., and Lelieveld, J.: Technical Note: The Modular Earth Submodel System (MESSy) – a new approach towards Earth System Modeling, Atmos. Chem. Phys., 5, 433–444, doi:10.5194/acp-5-433-2005, 2005.

Jöckel, P., Tost, H., Pozzer, A., Brühl, C., Buchholz, J., Ganzeveld, L., Hoor, P., Kerkweg, A., Lawrence, M. G., Sander, R., Steil, B., Stiller, G., Tanarhte, M., Taraborrelli, D., van Aardenne, J., and Lelieveld, J.: The atmospheric chemistry general circulation model ECHAM5/MESSy1: consistent simulation of ozone from the surface to the mesosphere, Atmos. Chem. Phys., 6, 5067–5104, doi:10.5194/acp-6-5067-2006, 2006.

Jöckel, P., Kerkweg, A., Buchholz-Dietsch, J., Tost, H., Sander, R., and Pozzer, A.: Technical Note: Coupling of chemical processes with the Modular Earth Submodel System (MESSy) submodel TRACER, Atmos. Chem. Phys., 8, 1677–1687, doi:10.5194/acp-8-1677-2008, 2008.

Jöckel, P., Kerkweg, A., Pozzer, A., Sander, R., Tost, H., Riede, H., Baumgaertner, A., Gromov, S., and Kern, B.: Development cycle 2 of the Modular Earth Submodel System (MESSy2), Geosci. Model Dev., 3, 717–752, doi:10.5194/gmd-3-717-2010, 2010.

Jungclaus, J. H., Keenlyside, N., Botzet, M., Haak, H., Luo, J.-J., Latif, M., Marotzke, J., Mikolajewicz, U., and Roeckner, E.: Ocean Circulation and Tropical Variability in the Coupled Model ECHAM5/MPI-OM, J. Climate, 19, 3952–3972, doi:10.1175/JCLI3827.1, 2006.

Kerkweg, A. and Jöckel, P.: The 1-way on-line coupled atmospheric chemistry model system MECO(n) – Part 1: Description of the limited-area atmospheric chemistry model COSMO/MESSy, Geosci. Model Dev., 5, 87–110, doi:10.5194/gmd-5-87-2012, 2012.

Pozzer, A., Jöckel, P., Kern, B., and Haak, H.: The Atmosphere-Ocean General Circulation Model EMAC-MPIOM, Geosci. Model Dev., 4, 771–784, doi:10.5194/gmd-4-771-2011, 2011.

Redler, R., Valcke, S., and Ritzdorf, H.: OASIS4 – a coupling software for next generation earth system modelling, Geosci. Model Dev., 3, 87–104, doi:10.5194/gmd-3-87-2010, 2010.

Rockel, B., Will, A., and Hense, A.: The regional climate model COSMO-CLM (CCLM), Meteorol. Z., 17, 347–348, 2008.

Roeckner, E., Arpe, K., Bengtsson, L., Christoph, M., Claussen, M., Dümenil, L., Esch, M., Giorgetta, M., Schlese, U., and Schulzweida, U.: The atmospheric general circulation model ECHAM-4: Model description and simulation of the present-day climate, Tech. Rep. 218, Max-Planck-Inst. für Meteorologie, Hamburg, Germany, 1996.

Sander, R., Kerkweg, A., Jöckel, P., and Lelieveld, J.: Technical note: The new comprehensive atmospheric chemistry module MECCA, Atmos. Chem. Phys., 5, 445–450, doi:10.5194/acp-5-445-2005, 2005.

Steppeler, J., Doms, G., Schättler, U., Bitzer, H. W., Gassmann, A., Damrath, U., and Gregoric, G.: Meso-gamma scale forecasts using the non-hydrostatic model LM, Meteorol. Atmos. Phys., 82, 75–96, doi:10.1007/s00703-001-0592-9, 2003.

Tost, H., Jöckel, P., Kerkweg, A., Sander, R., and Lelieveld, J.: Technical note: A new comprehensive SCAVenging submodel for global atmospheric chemistry modelling, Atmos. Chem. Phys., 6, 565–574, doi:10.5194/acp-6-565-2006, 2006.

Tost, H., Lawrence, M. G., Brühl, C., Jöckel, P., The GABRIEL Team, and The SCOUT-O3-DARWIN/ACTIVE Team: Uncertainties in atmospheric chemistry modelling due to convection parameterisations and subsequent scavenging, Atmos. Chem. Phys., 10, 1931–1951, doi:10.5194/acp-10-1931-2010, 2010.

Valcke, S., Guilyardi, E., and Larsson, C.: PRISM and ENES: a European approach to Earth system modelling, Concurrency Computat., Pract. Exper., 18, 231–245, doi:10.1002/cpe.915, 2006.

Vogel, B., Vogel, H., Bäumer, D., Bangert, M., Lundgren, K., Rinke, R., and Stanelle, T.: The comprehensive model system COSMO-ART — Radiative impact of aerosol on the state of the atmosphere on the regional scale, Atmos. Chem. Phys., 9, 8661–8680, doi:10.5194/acp-9-8661-2009, 2009.

Wong, D. C., Pleim, J., Mathur, R., Binkowski, F., Otte, T., Gilliam, R., Pouliot, G., Xiu, A., Young, J. O., and Kang, D.: WRF-CMAQ two-way coupled system with aerosol feedback: software development and preliminary results, Geosci. Model Dev. Discuss., 4, 2417–2450, doi:10.5194/gmdd-4-2417-2011, 2011.