Workshop Proceedings

Coupling Technologies for Earth System Modelling: Today and Tomorrow

December 15-17, 2010

Toulouse, France

Sophie Valcke/CERFACS, Rocky Dunlap/Georgia Tech

CERFACS Technical Report TR-CMGC-11-39 (May 2011)

On December 15th-17th 2010, CERFACS and the Georgia Institute of Technology organized a workshop on the theme of "Coupling Technologies for Earth System Modelling: Today and Tomorrow" at CERFACS in Toulouse, France. The first thrust of the workshop was to assess the state of the art in Earth System Modelling (ESM) coupling frameworks, including current assumptions about coupled geophysical models, the nature of the capabilities provided, and the range of software architectures currently employed. A primary objective was to explore and more deeply understand the trade-offs involved in the range of approaches to coupling in use throughout the community. The second thrust was to discuss a vision for coupling in the year 2020. This involved positing what new requirements will arise in the next ten years and how existing coupling frameworks must evolve to meet those needs.

45 people from different countries attended the workshop: 20 from France, 12 from other EU countries, 11 from the US, and 2 from China. The workshop programme was established by a Programme Committee composed of V. Balaji (Princeton University), Cecelia DeLuca (NOAA), Rocky Dunlap (Georgia Tech), Rupert Ford (University of Manchester), Sophie Valcke (CERFACS) and Mariana Vertenstein (NCAR). The first day and a half of the workshop was devoted to detailed presentations of current coupling technologies from developers of ESMF, CESM, MCT, PALM, OASIS, FMS, BFG, OpenMI and OOPS. The rest of the workshop included presentations on coupled modelling perspectives at different centres, coupling related issues (e.g., data assimilation, metadata, and education), software and hardware challenges for coupling technologies, and two round tables to allow in-depth, interactive discussions.

The present proceedings include:

- the detailed workshop programme (p.2),
- a summary of the round table discussions that took place both during the workshop and in the following weeks via interactive wiki pages (pp.3-6),
- a set of workshop conclusions (p.7),
- an extended abstract for each of the presentations (pp.8-66),
- the list of participants (p.67).

All details about the workshop including presentation slides and interactive wiki pages are available on the workshop web site at the following web address:

https://verc.enes.org/models/software-tools/oasis/general-information/events

Workshop programme

December 15th:
Introduction and welcome, by Bijan Mohammadi (CERFACS director)
Current coupling technologies and developments I; chair: Sophie Valcke (CERFACS)

- 09h15 – 10h00: The Earth System Modeling framework ESMF, by Ryan O'Kuinghttons (NOAA/CIRES)
- 10h00 – 10h45: The Community Earth System Model CESM1, by Tony Craig (NCAR))
- 11h15 – 12h00: The Model Coupling Toolkit by Robert Jacob (Argonne National Laboratory)
- 12h00 – 12h45: The dynamic parallel PALM coupler by Andrea Piacentini (CERFACS)

Current coupling technologies and developments II; chair: Bob Oehmke (NOAA)

- 14h00 – 14h45: OASIS, a coupler for climate modelling, by Sophie Valcke (CERFACS)
- 14h45 – 15h30: The GFDL Flexible Modeling System FMS, by Balaji (Princeton University)
- 16h00 – 16h45 : The Bespoke Framework Generator BFG, by Rupert Ford (U. Manchester)
- 16h45 – 17h30 : The OpenMI interface for flexible, dynamic coupling, by Stef Hummel (Deltares)
- 17h30 – 18h15 : OOPS - An Object Oriented Framework for Coupling Data Assimilation Algorithms to Models, by Mike Fisher (ECMWF)

December 16th:
Current coupling technologies and developments III; chair: Mariana Vertenstein (NCAR)

- 09h00 – 09h45 : C-Coupler: A coupler for Earth System Modeling, by Xiaoge Wang (Tsinghua University, Beijing, China)
- 09h45 – 10h30 : The Model for Prediction Across Scales (MPAS),, by Michael Duda (NCAR/MMM)
- 11h00 – 11h30 : Feature modeling of coupling technologies, by Rocky Dunlap, Spencer Rugaber, and Leo Mark (Georgia Tech)

11h30 – 12h30: Round table 1 : How do the different coupling technologies fit the different application needs and constraints; chair: Mariana Vertenstein and Sophie ValckeCoupling at the boundaries ; chair: Rocky Dunlap (Georgia Tech)

- 15h00 – 15h30 : Data assimilation and coupling, by Andrea Piacentini (CERFACS)
- 16h00 – 16h30 : Web based experiments with Earth system models of different complexity used for education at Freie University Ingo Kirchner (FU Berlin)
- 16h30 – 17h00 : Metadata and coupling, by Rupert Ford (Manchester University)

December 17th:

Coupled modeling perspectives at different centers ; chair: Rupert Ford (Manchester University)

- 09h00 – 09h30 : Leveraging the New CESM1 CPL7 Architecture - Current and Future Challenges, by Mariana Vertenstein (NCAR)
- 09h30 – 10h00 : Coupled models at the Max-Planck-Institute for Meteorology, by René Redler (MPI Meteorology)
- 10h00 – 10h30 : Infrastructure requirements in support of Met Office models, by Steve Mullerworth (MetOffice)

Software and hardware challenges for coupling technologies; chair: Mick Carter (Met Office)

- 11h00- 11h30: Addressing the Challenge of Exaflopic Computation, by Jean-Yves Berthou (EDF R&D)
- 11h30 – 12h00: Designing HPC Software for an Uncertain World of Hardware, by Wael Elwasif and David Bernholdt (Oak Ridge National Laboratory)
- 13h30 – 14h00: Future directions for coupling technology in Earth system modeling, by Balaji (GFDL)

14h00 – 15h15: Round table 2: software and hardware challenges for coupling technologies; chair: Rob Jacob

Conclusions and recommendations

Summary of the round table discussions

What follows is a summary of the round table discussions that took place during the workshop and in the following weeks via interactive wiki pages.

The theme of the first round table was a comparative analysis of current ESM coupling technologies including a discussion of how the technologies address different application needs and constraints. The following questions were proposed for the first round table:

- Is interoperability an unreachable dream? Should we even aim for it?
- What should be the role of the coupling software layer (exchange of data, interpolation/transformation of data, weights-and-address calculation, process management, code generation, load balancing, etc.)?
- How far/deep should we (try to) go in standardizing the coupling infrastructure (technical interface compatibility, superstructure/framework layer, scientific interface, etc.)?

The theme of the second round table was software and hardware challenges for coupling technologies. The questions proposed for the discussion were:

- How can more levels of parallelism be exposed in coupling operations?
- How can we increase concurrency in the coupled system?
- What is the impact of smaller memory per node on the coupling-related functions?
- If exascale needs new languages and rewrite of component models, what will be the impact on coupling?

Workshop discussions typically proceeded in an organic manner, sometimes jumping from one topic to the next and then back again. To improve coherence, instead of recreating the original flow of discussion we have organized the workshop content into several high-level topics: the definition of coupling, the scope of couplers, the current approaches to coupling, interoperability, and future coupler developments. In the concluding section, we summarize the major takeaways of the workshop and offer some recommendations.

The Definition of Coupling

For some workshop participants, "coupling" refers to the process of making two originally independent components interact through a separate architectural layer (as opposed to native subroutine calls) and necessarily involves data copy or transfer (and possibly other transformations such as rearrangement and/or regridding). Under this definition of coupling, the modularity of the original components is preserved, but the use of a separate architectural layer implies a potential loss of performance. Other workshop participants adhered to a more general definition of coupling: the action of making two originally independently modelled processes interact, regardless of the implementation used to realize this interaction (e.g., via data copy/transfer or via shared memory accesses or adding a process as a subroutine).

The Scope of Couplers

Since workshop participants represented a wide range of scientific and technical backgrounds, an important question centred on the scope of couplers. Which functions should couplers provide and which functions should be left to other software components in the system?

There was widespread agreement on certain basic functions, such as data transfer and interpolation/regridding, although not all software packages represented provided these functions and many rely on external packages for calculating interpolation weights. 2D linear, higher order, conservative and user-defined regridding functions were widely accepted as essential, and 3D volumetric regridding was also indentified as important for some applications. The type of grids supported should include all logically-rectangular grids (latitude-longitude, stretched, rotated, etc.) but also unstructured grids such as Gaussian reduced and icosahedral grids which are becoming more popular in the climate modelling community. Coupling of components with adaptive grids is foreseen in the near/mid- term. In this case, the coupler should then be able to recalculate the regridding neighbours and weights during the run and efficiently manage the impact on the communication.

There are other capabilities currently supported by a subset of the coupling technologies. Some couplers allow external process control of the components being coupled (e.g., time stepping), where others just ensure transformation and exchange of coupling data without affecting the execution of the component models per se. It is also interesting to note that in some cases the coupling layer can manage ensemble runs of the coupled model. This allows one ensemble of runs to be considered as one application with an added level of parallelism (the number of runs). Discussions did not lead to a firm conclusion on whether or not the coupling layer should also manage the dynamic (re-)load balancing of the coupled model components.

Current Approaches to Coupling

The coupling technologies presented during the workshop can roughly be split into two main categories. With the "multiple executable" approach (e.g., OASIS, O-PALM), the original components are run as separate concurrent executables, and their main characteristics, such as memory management or internal parallelisation, remain practically untouched with respect to their standalone mode. The exchange of coupling data is performed through in-place "put" and "get" instructions which are configured externally for a particular run (e.g., the source and the target, the coupling frequency). In this case, the components expose only a data interface to the coupler and it is the user's responsibility to ensure that the component models coherently define some global parameters such as the total run duration, the calendar, etc. The main advantage of this approach is that it requires minimal intrusion into or restructuring of existing legacy codes. The drawback is that it is less flexible and potentially less efficient because it constrains the way components can be mapped to the computing hardware. Therefore this approach can lead to a waste of resources if the components are "naturally sequential" (i.e., if one component necessarily waits for an input while the other is doing some calculation and vice-versa) and if they are run on separate sets of processors. Furthermore, in a multiple executable system, it is not possible to pass coupling data by reference, which would generally be faster.

In the "integrated" mono-executable approach (e.g. CESM1, ESMF, FMS) each model source code is decomposed into init, run, and finalize units with argument lists that match the interface standard expected by the coupling layer. The coupling data are made available as input and/or output at each calling interface. Conforming to component interfaces is typically achieved by creating wrappers that are distinct from user code. The internals of user code, including data structures and parallelisation, are not affected by the wrappers. This approach is more flexible and in some cases more efficient as the component models can be executed concurrently, sequentially, or in some hybrid mode and coupling exchanges can be optimized as shared memory accesses. Components can be nested within other components allowing many possible configurations of couplers and components. However, this approach requires that components expose both data and control interfaces. A driver or parent component controls the couplers and components, and it also enforces a consistent subset of global parameters (such as run duration) across component models. If components do not already have clear initialize, run, and finalize units, significant restructuring may be required. The places where data transfers can happen is restricted, and this may affect the control flow and require scientific reformulation.

Research in Generative Programming (such as BFG2) proposes ways in which the "multiple-executable" and "integrated" approaches can be combined.

Interoperability and componentization

Interoperability was an important theme throughout the workshop and was the particular point of focus during the first round table discussion. Participants recognized the advantages of interoperability in the context of Earth System Modelling, especially the ability to reuse code in new contexts and facilitate coupling of external components into existing models.

Most participants agreed that working toward some level of interoperability is a worthwhile goal. However, this did not imply that full-fledged "plug-and-play" compatibility is a viable goal. Plug-and-play compatibility implies that component models use not only the same *technical* coupling interface (how to exchange data), but also must agree on a standard *scientific* coupling interface (what data to exchange). The European PRISM project made an attempt to agree on such two-level standard but this proved unsuccessful. In fact, even if the technical coupling interface can in principle be standardized (all groups can agree to use the same coupling software), defining standard scientific coupling interfaces turned out to put burdensome constraints on the science itself. Therefore, PRISM did not put further effort into the standardization of the scientific interfaces, as keeping diversity in the science (by minimizing scientific constraints) was considered more important. Furthermore, in many cases scientists do not want to change different components frequently as climate applications are sensitive complex systems that need to be tuned and evaluated in detail for each particular configuration.

All agree that componentization[1] makes sense at some level of coupling. Componentization means wrapping a particular scientific model with a *clear set of functions* and *well-defined interfaces* to provide a convenient method for coupling and use. Componentization facilitates program understanding and promotes interoperability; in fact, the difficult step in coupling is usually identifying and harmonizing coupling fields across components, not adapting model codes to the same technical interface (i.e., the same coupling layer). Componentization also provides a logical organization of the source code and helps prevent evolution toward a monolithic code. However, at some point, componentization may incur a penalty of performance, or reduce the code readability.

Further coupler developments in the short, mid and long term

Existing couplers have been developed by different groups with different goals, priorities and constraints in mind. In the short and mid term, this is likely to continue. Therefore, it will be important to continue to develop, in parallel, a reduced number of coupling technologies, each having a significant amount of resources (~5 FTEs at minimum), and each targeting a different coupling approach. Of course, sharing some building blocks (e.g., the conservative remapping algorithm) among these technologies should be encouraged. The coupler development teams should include computing scientists interacting frequently with climate modelling scientists.

In all cases, an effort should be made to identify, share, and promote best practices in coupling, such as the calculation of fluxes at the resolution of the "exchange grid" (see FMS presentation) or adaptation of the surface tiles in the atmosphere model to fit the ocean model coastline.

In the longer term, the coupling technologies will have to adapt to future computer architectures. While the individual arithmetic processors are probably going to remain at ~1 GHz (10^9), there will most likely be a massive increase in the number of cores, with increased heterogeneity (e.g., a mix between CPUs and GPUs) and modest increases in available memory per core and inter-node communication throughput. One way to reach the exascale ($O(10^{18})$ FLOPS) expected before the end of the decade is to expose new levels of parallelism. The following proposal was presented, assuming that processors remain at $O(10^9)$ clock speed:

- *Increase the resolution*. This will allow a higher level of parallelism within individual model components, which we expect to execute on $O(10^5)$ processors. But the throughput of the model is expected to continue to decrease with increasing resolution.
- *Increase the number of concurrent components*. O(10) concurrent components should be run in one coupled application; concurrent coupling using a forward-only timestep (i.e., $X(t+1) = X(t) + f[Y(t)]$) could be applied to sub-components (e.g., physics time

[1] Componentization in geoscience modeling community is akin to software *modularization*. ESM components should not be confused with the notion of components as defined by the Component-based Software Engineering community.

dependencies in the atmosphere) in addition to the ocean and atmosphere components for which it is traditionally used.

- *Increase the number of members in ensemble simulations*. One ensemble simulation with O(100) members should be considered as one climate application with the different members adding one level of parallelism.
- *Increase the concurrency in the workflow*. Additional parallelism of O(10) should be reached at the level of the different workflow tasks (pre-processing, execution, post-processing, analysis).

Parallelization along the time dimension and run of multi-model ensembles were also mentioned as other ways to increase the parallelism.

Many aspects of running with increased concurrency of components were discussed. With more concurrent components, load balancing will become even more difficult but this might not be as great a concern as the cost of the CPU cycle will most probably no longer be the limiting factor. Memory movement (e.g. coupling data transfer) will likely become more expensive and so it may be necessary for models to stop decomposing their domains independently (letting the coupler sort it out) and instead co-locate points from the same region on the same node as much as possible. Participants proposed several other ways to reduce the coupling-related communication costs. Overlay of calculation and communication (e.g., a communication is started, some computation is executed at the same time, and then the communication is checked for completion) and redundant calculations (i.e., defining very wide halos and doing the calculation for the same grid points by multiples processes) are options that should be explored. Of course, we can always hope for faster hardware networks and improved communication implementations using MPI or possible alternatives.

Regarding future platforms, the multiple executable coupling approach may still be the choice of a part of the community. Therefore, it is important to ensure that future operating systems support the MPMD mode.

Finally, the impact of new languages (beyond F90 and MPI) is hard to evaluate. Some current programming models (e.g., CUDA for GPUs) are not amenable to a driver-kernel programming model. In all cases, the community will have to get organized to ensure that the new languages and technologies are adopted at the same pace by the developers of the different components. If the new programming languages and paradigms force the community to rewrite model code, this should be seen as an opportunity to consider the adoption of community-wide technical standards that would facilitate the coupling or even to unify coupling approaches and share developmental costs.

Conclusions and recommendations

- The advantages of *interoperability* and *componentization* in the context of Earth System Modelling are recognized. However, plug-and-play compatibility should not be an objective per se, as it requires a standardization of the scientific coupling interfaces among the components (i.e., what field data to exchange). Furthermore, swapping model components quickly and frequently is not of primary importance in the climate modelling community as climate models are complex systems that require detailed tuning and validation for each combination of components.
- Current coupling technologies can roughly be split into two main categories. The "multiple executable" approach is somewhat less flexible and can be less efficient in some cases but is straightforward to implement requiring minimal modification to individual models. The "integrated" mono-executable approach, which requires the original codes to be split into init, run and finalize units and some standardization of the resulting component interfaces, limits the places in code where data exchanges can happen. Although this can simplify program flow, it can also affect time sequencing and require scientific reformulation. However, because components can be run sequentially or concurrently, there are additional opportunities for performance optimization
- For maximum coupling flexibility and efficiency, all climate component models should be re-factored into init, run and finalize units. Where the norm is a multiple executable approach, such as the European climate modelling community, it may be difficult to achieve the agreement on component interfaces required for integrated coupling. To satisfy all cases, an "ideal" coupling technology should therefore offer both approaches in order allow an easy assembling of legacy code but also provide more efficient and flexible coupling when interface agreements can be reached. Current research in Generative Programming explores approaches that may enable such an "ideal" coupling technology to be built.
- Existing coupling technologies have been developed with different priorities and constraints. In the short term, it is recommended to keep the parallel development of a reduced number of coupling technologies, each one with a significant amount of resources (~5 FTEs at a minimum). The coupler development teams should include computing scientists interacting closely with climate modelling scientists.
- In all cases, the different coupler developers should interact more closely and share more infrastructure building blocks (e.g. remapping/regridding algorithms, decomposition descriptions, metadata utilities, parallel I/O libraries, performance timers, etc.). Best practices in coupling should also be discussed, identified, and promoted.
- On the longer term, increased parallelism is seen as essential to exploit the exaflop platforms expected before the end of the decade. It will then be crucial to limit the load of the associated data communication by carefully distributing the coupled components over available processes or by finding ways to diminish its impact (overlay of communication and calculation, redundant calculations, etc.).
- If model rewrites are required in new programming languages in the years to come, we should take advantage of that opportunity to better agree on coding and coupling standards that will better facilitate the coupling of Earth System components between different groups. As future hardware will probably require significant changes in coding approaches, leveraging/combining our resources as much as possible to address the new hardware challenges should be seriously considered.
- The challenge of leveraging the exascale for climate modelling should be addressed with significant manpower and funds so to ensure that climate science remains a major driver for high performance exascale computing.

Extended abstracts

The Earth System Modeling framework ESMF

by Ryan O'Kuinghttons (NOAA/CIRES)

General Overview

The Earth System Modeling Framework (http://www.earthsystemmodeling.org) is open source software for building modeling components, and coupling them together to form weather prediction, climate, coastal, and other applications. ESMF was motivated by the desire to exchange modeling components amongst centers and to reduce costs and effort by sharing codes.

The ESMF package is comprised of a superstructure of coupling tools and component wrappers with standard interfaces, and an infrastructure of utilities for common functions, including calendar management, message logging, grid transformations, and data communications. The project is distinguished by its strong emphasis on community governance and distributed development, and by a diverse customer base that includes modeling groups from universities, major U.S. research centers, the National Weather Service, the Department of Defense, and NASA. The ESMF development team is centered at the NOAA Earth System Research Laboratory and the Cooperative Institute for Research in Environmental Science at the University of Colorado.

Rationale and History

The ESMF collaboration had its roots in the Common Modeling Infrastructure Working Group (CMIWG), an unfunded, grass-roots effort to explore ways of enhancing collaborative Earth system model development. The CMIWG attracted broad participation from U.S. weather and climate modeling groups at research and operational centers. In a series of meetings held from 1998 to 2000, CMIWG members established general requirements and a preliminary design for a common software framework.

In September 2000, a critical mass of CMIWG participants developed a coordinated response to a NASA solicitation that called for the creation of an “Earth System Modeling Framework.” They received awards for linked proposals that covered development of the framework and its incorporation into modeling and data assimilation applications. As the ESMF project gained momentum, it replaced the CMIWG as the focal point for developing community modeling infrastructure in the U.S. The second major development cycle for ESMF saw the framework emerge as a multi-agency effort. Major new grants came from NASA, the Department of Defense, NOAA, and the National Science Foundation, and many smaller ESMF-based application adoption projects were funded in domains as diverse as space weather and sediment transport. During this project phase, the central data structures in ESMF were completely rewritten to improve flexibility and extensibility.

In 2008, ESMF was chosen as the technical basis for the National Unified Operational Prediction Capability (NUOPC), a consortium of U.S. operational weather and climate centers that aims to deliver an ESMF-based, managed, multi-model ensemble. The emergence of this large-scale national project marks the beginning of the framework’s third phase.

Component Architecture

ESMF is based on principles of component-based software engineering. The components within an ESMF software application usually represent large-scale physical domains such as the atmosphere, ocean, cryosphere, or land surface. Some models also represent specific processes (e.g. ocean biogeochemistry, the impact of solar radiation on the atmosphere) as components. In ESMF, components can create and drive other components so that an ocean biogeochemistry component can be part of a larger ocean component.

ESMF offers two kinds of components: a Gridded Component (GridComp), which is associated with a physical domain, and a Coupler Component (CplComp), for transforming and transferring data between GridComps. ESMF components exchange information with other components only through State objects. A State contains data types representing fields, arrays, or other States. Each Gridded Component is associated with an import State, containing the data required for it to run, and an export State, containing the data it produces.

In order to adopt ESMF, modelers must decide how to organize their code as a set of GridComps and CplComps, then split these components into standard ESMF methods (initialize, run, and finalize, each of which may have multiple phases). The next step is to wrap native model data structures with ESMF data structures. This can be done either in index space, using a very general ESMF Array class, or in physical space, in which case model grids must be expressed using the ESMF Grid class. If Grids are used, ESMF can generate the interpolation weights needed for remapping between components.

ESMF enables components to run sequentially, concurrently, or in a mixed mode. Applications usually run with all components linked into a single executable program, but there is also support for running separate components as multiple executables. ESMF is written mainly in C++, and has Fortran and C interface bindings.

Coupling and Other Capabilities

Grid remapping is a central function of ESMF, and the framework supports a wide variety of grids and remapping options. Generation of interpolation weights and their application is fully parallel. ESMF supports first order conservative, bilinear, and a higher-order finite element-based patch recovery method for remapping. Logically rectangular and unstructured grids are both supported, in 2D and 3D. There is a range of options with respect to masking, handling poles, and behavior of unmapped points. The remapping system is flexible and modular, in that the calculation of interpolation weights can be performed either during a model run or offline, and the subsequent application of weights can be made as a separate call.

The ESMF team is actively developing extensive metadata handling capabilities. This effort was motivated by the growing need to carefully document the provenance of the data produced by climate and other simulations, and by the desire to automate aspects of coupling to enhance cross-institutional interoperability. ESMF has a class that represents metadata as name-value pairs within either prefabricated or custom “Attribute packages.” Methods of this class can be used to aggregate, store and output metadata. Metadata schemata follow community conventions such as the Climate and Forecast (CF) conventions, ISO standards, and the METAFOR Common Information Model (CIM).

Overview of Results

Since its inception in 2002, the ESMF effort has steadily grown, attracting new users, new offshoots, and new sponsors. Its success can be measured by the increasingly robust and fully-featured software that it has delivered, by the growing pool of ESMF components and applications in the community, and by the emergence of new partnerships facilitated by this shared infrastructure.

Timing results for a variety of codes show that the overhead of using ESMF components is typically negligible (< 3% of runtime), and that key operations have good scaling to tens of thousands of processors. Grid remapping and parallel communications are highly scalable and extensible to many new grid types. The framework is very robust and is exhaustively tested nightly on 24+ platforms with a suite of over 4000 tests (covering remapping accuracy, API correctness, use test cases, etc). ESMF customers are now finding that they are increasingly able to achieve results using the framework that they cannot through other means.

There are currently more than 70 ESMF components and applications in the community. The largest ESMF systems are the GEOS-5 model at NASA Goddard Space Flight Center, which is structured as a deeply nested component hierarchy; the whole Earth system developed by the Battlespace Environments Institute, which combines coastal, watershed, ocean, atmosphere, and space weather components into multiple models; and the new numerical weather prediction system at the National Centers for Environmental Prediction, which will be a key part of a next-generation operational multi-model ensemble. These activities have been deeply integrative, bringing to bear the resources of multiple organizations on problems too large for any one of them to address alone.

Future Plans

In the future, ESMF will continue to improve and extend its functionality, improve training materials, and expand and support its customer base. The project is also evolving to address new concerns. ESMF initially focused on coupling components intended to run on the same computer, with performance as the foremost concern. In response to changing science requirements and technical trends, future plans focus on leveraging the interface and metadata standardization implicit in ESMF adoption in order to enable ESMF components to operate in more heterogeneous environments. One aspect of this is linking ESMF components to web-based coupling technologies. Another is introducing ESMF components and models into science gateways that catalog and integrate diverse, distributed resources.

The integration of modeling with data services is a key part of this vision. The Curator project, initiated in 2005 with NSF funding and continued under NASA and NOAA, pairs ESMF leads with collaborators from the NOAA Geophysical Fluid Dynamics Laboratory, the Earth System Grid (ESG) data distribution portal project, the Georgia Institute of Technology, and the E.U. METAFOR project. The Curator group is creating a user interface for ESG based on the METAFOR Common Information Model. This will provide access to much more structured, searchable information about the models used to generate climate datasets than has ever been available before. The portal will be used to support the fifth Coupled Model Intercomparison Project, part of the next Intergovernmental Panel on Climate Change assessment.

A New Flexible Coupler Designed for Earth System Modeling for CCSM4/CESM1

by Anthony Craig (NCAR[2] Earth System Laboratory)

The Community Climate System Model (CCSM) development is based at the National Center for Atmospheric Research (NCAR) in Boulder, Colorado, USA. CCSM is a state-of-the-art global climate model consisting of four fundamental physical components: an atmosphere model, an ocean model, a land surface model, and a sea ice model. In addition, a coupler (or driver) is used to exchange boundary data between the components and to coordinate the time evolution of the physical models. CCSM is used to understand the Earth's global climate system, to predict the effects of climate change, and to understand past climates. It is developed as a high performance computing application but is used on a wide variety of platforms. Various fully prognostic (active) data (where coupling fields are derived from input files), dead (for testing), and stub models are available for use in the system. The Community Earth System Model (CESM) is an extension of CCSM that includes an additional land-ice component, a higher altitude atmosphere model option, land and ocean biogeochemistry capabilities, and an atmospheric chemistry model. For the purposes of this discussion, CCSM4 and CESM1 are the same model and the name "CCSM" or "CCSM4" will be used to describe capabilities in both models.

In general, couplers carry out critical but limited functions within coupled systems. These functions normally include the support of data communication between components, the sequencing and integration control of the system as a whole, and the execution of coupling methods such as mapping (interpolation), merging of fields, and diagnostics. Sequencing and integration control are associated with the time evolution of the system and deal with issues such as the temporal sequencing of components and the coupling frequencies and lags between components. Separately, there are several basic characteristics of coupled systems that have implications on the system design. The first characteristic is whether the system is a based on a single or multiple executable design. The second is how components can be laid out on processors; whether sequential, concurrent, or mixed. The third characteristic is whether data is communicated between components directly or through a central "hub". And finally, the fourth important characteristic is whether the system is coupled via a top-level driver or through coupling calls from inside components directly. The coupler functions and design characteristics play an integral role in determining the climate model system implementation.

There is a long history of building coupled climate models at NCAR. CCM development started in the 1980s and was one of the first coupled climate models. It consisted of prognostic atmosphere and land models coupled to simpler prescribed ocean and ice models. The model was fully sequential, all components ran on the same grid, there was no memory parallelism, and surface models were called from the atmosphere component directly. In this model, the atmosphere component acted as the top-level driver and communication was through interface calls without any need for interpolation. In the mid-1990s, NCAR started to develop more sophisticated coupled climate models with fully prognostic components in ways that required distinct coupling software. Initially, there were two parallel efforts. PCM was a single executable, memory parallel implementation where the atmosphere/land, ocean, and sea ice were all on distinct grids but running on the same processors. The initial target platforms for PCM were CM5, T3D, T3E, and SGI Origin. Coupling was carried out with a

[2] NCAR is sponsored by the National Science Foundation

distinct driver, components placed coupling data in a coupler common block, and coupler functionality such as mapping was carried out as part of the top-level driver implementation. The other mid-1990s NCAR coupled climate model was CSM1. CSM1 was a multiple executable, shared memory parallel implementation with distinct grids for the atmosphere/land and ocean/ice. In CSM1, all components ran on distinct hardware processors and there was a separate hub coupler that mediated communication, mapped fields, and implicitly handled time integration. Coupling in CSM1 was done via calls directly in components. The initial target platforms for this implementation were Cray vector platforms such as the YMP and C90 as well as SGI Origin systems. In around 2000, an effort was made to merge the best features of PCM and CSM1 into a single model called CCSM2. The CCSM2 coupling architecture largely followed CSM1 with a multiple executable, concurrent design using a separate hub coupler. But the CCSM2 components were now partly based on the work of PCM with respect to memory parallel capabilities, although the coupler in CCSM2 was still a single processor application. CCSM2 targeted general distributed shared memory systems. The coupler became a fully memory parallel component with the release of CCSM3 in 2004.

With the CCSM4 release in 2010, a completely new approach was taken to coupling climate models. CCSM4 is a single executable implementation that contains a top-level driver, components are coupled via standard init/run/finalize interfaces, and individual components in CCSM4 can be laid out on processors in relatively arbitrary ways such that components can be run on identical or independent hardware processors. The top-level driver that runs on all processors controls the processor layout and time sequencing of the components. A separate coupler component that can run on a subset of all the processors still exists in the system to map, merge, and carry out other coupler functions. Components in CCSM4 are parallelized with MPI and OpenMP. The driver/coupler grids, component layout on processors, decompositions, and configuration options are set at run-time based on Fortran namelist inputs and communication with components at initialization. The CCSM4 driver/coupler uses Model Coupling Toolkit (MCT) datatypes and methods extensively, mapping weights are generated offline, and a new parallel IO (PIO) library is being used to support improved I/O performance particularly in the area of memory scaling.

There were several reasons for migrating to this new coupling approach in CCSM4. The new implementation improves performance because of greater flexibility in laying out components on hardware processors compared to the prior concurrent-only CCSM3 system. The new processor layouts also allow models to be coupled more tightly when needed without worries over concurrent performance. CCSM4 provides an ability to run on a single processor without MPI sequentially but is more memory and performance scalable for runs at much higher resolution. Finally, implementing CCSM4 using a top-level driver with init/run/finalize interfaces allows compatibility with the Earth System Modeling Framework (ESMF) superstructure.

As mentioned above, components can be laid out on processors in relatively arbitrary ways, at least in a technical sense. In addition, model results are independent of the component layout on hardware processors because the driver sets the temporal evolution of the system not the processor layout. However, in the current implementation of the driver, components cannot actually be run completely concurrently. For scientific reasons related to consistency of the surface albedos and the atmospheric radiation calculation, the atmosphere model run method is always called after both the sea ice and land models' run method. This is the only constraint on concurrency in the system at the present time. But as a result, the optimal component processor layouts usually have the atmosphere model overlapping hardware processors used by the sea ice and land models.

The scaling of the CCSM4 coupler has been evaluated using four basic coupler kernels. A flux calculation, a merge operation on the ocean grid, a rearrange of ocean data between two different decompositions, and a mapping of data between an atmosphere and ocean grid were run at two different resolutions and on three hardware platforms from 1 to 10,000 processors. Time in seconds for each test case was collected. The timers were aggregated over the full 20-day dead model test, and barriers were used to eliminate the impact of load imbalance outside the kernel.

The scaling of the flux kernel, which is trivially parallel (there is no communication) and very FLOP intensive, is excellent. Linear scaling is demonstrated across all processor counts, resolutions, and machines. The merge kernel is also trivially parallel but is primarily a memory intensive operation. In the merge, several fields are copied out of memory and combined using a simple mult-add operation. Those merged fields are then copied back into memory. The scaling of the merge operation is linear at lower processor counts but then flattens out at higher processor counts as the number of gridcells per processor decreases below a few hundred. It's likely the scaling of the merge is influenced by cache line efficiency in this regime.

The rearrange kernel is dominated by communication. In this kernel, fields are rearranged between two different decompositions on a common set of processors. This is almost an all-to-all communication operation. As expected, the scaling is sub-linear at lower processor counts and then quickly flattens out. Scaling performance is highly dependent on the machine and resolution in this case. The rearrange kernel speeds up on all machines out to 100 to 500 processors. At higher processor counts, the scaling is flat or worse. The mapping kernel scaling is similar to the rearrange kernel. The mapping operation involves rearranging data and applying mapping weights in a mult-add loop to derive fields on new grids. At lower processor counts, the mult-add plays a relatively larger role in the total mapping cost. But at higher processor counts, the scaling of the mapping is consistent with the scaling of the rearrange where performance flattens out and depends heavily on resolution and hardware architecture.

The improvements in the memory and performance scaling capability of CCSM4 compared to CCSM3 are significant. Performance improvements in the coupler were needed to keep up with improvements in the rest of the system and to achieve new science goals. The model is now being run at global resolutions of around one tenth of a degree on tens of thousands of processors with reasonable performance and scaling.

Continual improvement in climate model performance in the future may become more difficult. Most of the gains in the last decade came from faster hardware on a per processor basis and improvements in basic grid decompositions, memory parallelization strategies, and communication algorithms. Unfortunately, future generation hardware is likely to consist of orders of magnitude more processors that are slower, heterogeneous, and with less and slower memory. If so, future hardware will require models to scale to even higher processor counts just to maintain today's model throughput at current resolutions. New component, physics, gridcell, tracer, or time parallelism may need to be found in the system, and new communication strategies such as one-sided or the overlapping of non-blocking communication with computation may need to be implemented.

The Model Coupling Toolkit

by Robert Jacobs and Jay Larson (Argonne National Laboratory)

MCT was designed to be an application neutral approach to solving parallel coupling problems in multiphysics applications. “Application neutral” means not only that MCT can be used in other applications besides climate modeling but also MCT does not impose any structure on the overall application. Choices such as timestepping, number of models or number of fields and overall architecture can affect the possible science done and should be left to the coupled application developer. The developer also should be able to choose a la carte which pieces of the toolkit to use in the coupled application. Application neutrality and a library approach were central to our design philosophy in creating the Model Coupling Toolkit (MCT). The application-neutral elements of coupling include data description, efficient movement of that data in parallel and support for parallel data transformation and interpolation.

MCT does not prescribe how communication and process management is done within the model. Instead, MCT provides a datatype to describe the assembled coupled system to MCT. MCT assumes MPI-based parallelism, but includes a small MPI-replacement library that allows MCT to be used in non-parallel applications. MCT’s main datatype is a field data object supporting storage of arbitrary numbers of real- or integer-valued fields indexible using string tokens. This holds all data to be transferred during coupling. Another important datatype is a highly flexible domain decomposition descriptor that employs virtual linearization to represent multidimensional index spaces. This datatype is used by MCT to automatically derive communications schedules for parallel data transfer and redistribution. Calls to simple send/receive pairs using the MCT data storage type and the derived communication schedule as inputs perform all of the data transfer. Distributed storage for pre-computed interpolation coefficients is also provided and MCT is able to query the coefficients to derive communication schedules for parallel interpolation. All of the aforementioned classes have comprehensive method support.

MCT’s native API is Fortran-based, but bindings for C++ and Python are also available. MCT includes a highly portable build system based on GNU autoconf. MCT’s programming model derives from Fortran90, comprising module use to access MCT classes and methods, declaration of variables of MCT’s datatypes, and invocation of MCT’s methods to perform coupling operations. To use MCT, the coupled model developer first uses their application knowledge to locate logical interaction points in the legacy subsystem model. Once this assessment is made, the user first adds code to the model to declare and initialize MCT’s main datatypes for coupling. The user inserts MCT handshaking calls between model pairs to initialize the communication schedules. With the “run” methods of the model, the user places the calls to load the model’s data in to the MCT datatype and then call MCT’s parallel communication and/or interpolation methods

The most difficult steps conceptually are defining the virtual linearization of mesh and index spaces to support the MCT communication methods. We find that most new users of MCT are able to build their own MCT-based parallel coupled models after experimenting with the short example codes bundled in the MCT distribution, and reading and refactoring their source code. The ease-of-use is the primary benefit of using MCT while its main limitation is a lack of support for internal computation of interpolation weights and tools for MPI communicator construction. MCT is robust; it is the basis of the coupler in all versions of the U.S. Community Climate System Model since 2004 (CCSM3, CCSM4 and CESM1), supporting literally thousands of model years of coupled climate integration by a community of hundreds of scientists.

When considering MCT's future development, the arrival of exascale presents the most challenges. The paucity of per-core memory at exascale means copying field data and replication of domain decomposition descriptor data will have to be revisited. Field data copying could be obviated through use of general mesh representation software in all components. The domain decomposition descriptor problem may be attacked by using space-filling curves as virtual linearizations. Tolerating faults and changing pools of processors mean MCT's assumption of a static pool will also have to be revisited. At present, MCT supports application on tens of thousands of processors and is well positioned for future coupled model challenges.

The dynamic parallel O-PALM coupler

by Andrea Piacentini, Thierry Morel, Anthony Thévenin, Florent Duchaine (CERFACS)

Since 1996, CERFACS is developing the PALM coupler, which is currently used for more than 50 research and industrial projects, ranging from operational data assimilation, to multi-physics modelling, from climate change impact assessment to fluid and structure interactions. O-PALM is the open source version of PALM, distributed under the LGPL license. This new distribution policy arises from the consideration that the coupler has reached a high degree of maturity and stability. This makes easier to set up new coupled applications and allows other developers to contribute to the coupler evolution.

At the very origin of the project there was the commitment to provide the software backbone for the implementation of operational data assimilation suites. In particular, the French oceanography project MERCATOR was facing the challenge to set up a completely new operational forecast and assimilation system. The choice of the model configuration was not yet finalized, there were several candidate assimilation methods to test, there were different kinds of observations to handle and the same system should have been used for research and operations. All these needs of flexibility lead to the implementation of the assimilation suite as a coupling between model, observations handling, error statistics and algebra instead of hard-coding data assimilation routines in the model, or vice-versa.

Some data assimilation algorithms are based on an iterative minimization: this implies the repeated execution of the tasks and the total number of iterations is not necessarily known beforehand. Moreover, in some configurations some tasks are activated only if some observations are available at run-time. This specific requirement imposed to conceive a coupler of independent parallel codes capable to deal with complex coupling algorithms allowing for the conditional and/or repeated execution of the coupled components. The main goals and constraints were user friendliness, modularity, portability and high performances on parallel computers. OASIS was not a suitable choice for the lack of the dynamic aspects. On the other side, not all the OASIS features were needed for data assimilation at that time, in particular all the grid to grid interpolation issues. This lead to the design of a new MPMD dynamic parallel coupler based on the MPI message passing and process management standard library.

In our definition, a dynamic coupler has to fulfil three main requirements:

- process management: this means that the coupler has to be able to start and synchronise the tasks and to handle algorithms with loops and conditional switches.
- buffered communications: in order to grant full flexibility, avoiding deadlocks dependencies on the production and reception order, at least the production side of a communication has to be non blocking. This requires the explicit handling of a storage space for pending communications. This feature allows for some extra possibilities, such as the linear combination of cumulated fields and the explicit permanent storage of objects that are to be repeatedly received.
- object versioning: the flexible use of a temporary storage space for parallel communications requires special care to grant the coherency of the stored global objects. The Last In Only Out paradigm is adopted: every new version of an object replaces the previous ones. Nevertheless, for parallel communications, we count a new version of an object only when all the processes of the producing code have provided their contributions. For loosely synchronised codes, it implies the introduction of stamps to keep track of what version of an object new contributions belong to.

The same way, in a parallel coupling, a coupler has to deal with two levels of parallelism:

- Concurrent tasks parallelism: independent tasks can run concurrently on separate sets of processes. The coupler has to deal with the concurrent execution, to establish all the needed intercommunication contexts and to grant synchronisation.
- Distributed coupled codes: as a second level of parallelism we account for the inner parallelism of the coupled codes, mostly related to data distribution. The coupler has to grant private and robust intracommunication contexts and, most important, to be able to manage the data exchanges between sets of processes, including the remapping between codes with different distributions of the same physical objects.

Since one of the main aims of coupling is the reuse of legacy codes, we tried to reduce the intrusiveness of the coupling instructions in the source codes. For this reason we adopted the so-called end point communication paradigm: the producer of an object does not know anything about the recipients (if any) and the other way round. The coupler makes the matching. In order, once more, to minimize the interventions in the codes, we defined a reduced set of multi-language API calls, complemented by a very detailed Graphic User Interface: most information - such as the coupling algorithm, the communication patterns and the parallel distributions - is easily described in the graphic interface. The changes in the code have minimal impact and, because of the use of the one-sided communications paradigm, they are independent of the specific coupling algorithm.

The last constraints that drove the design of PALM are related to its uses and diffusion. The operational usage imposes robustness and high performances. This not only determined some implementation choices, but it also lead to the integration in PALM of a real-time monitor, allowing to display in the graphic user interface the status of the execution while running and of a performance analyser that works on trace files and helps tuning and optimising the coupled application. For research applications there are other criteria, such as portability, that imposed to rely on standard coding and message passing techniques) and user friendliness. The latter not only drove the design of the Graphic User Interface, but it lead also the the introduction of an algebra toolbox providing a palette of predefined generic algebraic operations ranging from BLAS to parallel linear algebra solvers and minimisers that can be coupled to any other user defined code.

The coupler implementation went through several steps. At the very beginning of the project, when the MPI2 standard was recently published, but hardly any complete and robust implementation was available, we implemented an MPI1 emulation based on a pool of idle processes, released under the name of PALM_RESEARCH, later changed into PALM_SP. It was dedicated to functional tests, but in practice it proved to be very effective in some cases and it still used for some full size applications. Some interesting features of this first implementation could now be seen as possible optimisations under some conditions and will be hopefully reintroduced in the current PALM version in a near future.

In 2003 we released the first fully MPMD version of PALM under the name PALM_MP. It was based on the MPI2 process management and communication layer. The main components of PALM_MP are:

- The scheduler that handles the process management and the execution of the coupled components accordingly to the algorithm described in the user interface. PALM can schedule several parallel codes to run concurrently to perform independent tasks if enough resources are available. Since starting an independent executable always causes a overhead, PALM offers the option to merge into a single executable the coupled components that are started in a sequence.

- The optimised communication scheme managed by a driver that takes care of the data transfer between parallel programs. This is one of the most evolved components of PALM and handles very complex communication patterns with some very practical features, such as the remapping of objects exchanged by parallel codes with different distributions, the selection of object subsets entirely from the user interface, the presence of an explicitly managed permanent repository for objects to be repeatedly received.

Since then the coupler has been constantly enhanced and optimised. With respect to PALM_MP, the current O-PALM release offers

- the possibility to interface commercial black-box codes (such as Fluent, Abaqus MSC/MARC) by the use of external dynamic libraries and/or a socket based layer
- a simplified working mode entirely compliant with the MPI-1 library
- the optimisation of repeated well synchronised communications that don't require the intervention of the driver
- the enhancement of the parallel algebra toolbox that is soon going to include the CWIPI interpolation library from ONERA for the grid to grid remapping.

Current PALM applications largely go beyond data assimilation and cover many fields of multi-physics coupling ranging from oceanography to hydraulics, from hydrology to agronomy, from aeronautics to space engineering and so on. Some of them are particularly representative of the advantages coming from the dynamic coupling and from the user friendliness of the API's and of the user interface. For instance we could mention the use of PALM for the shape optimisation of a combustor cooling system. In this kind of applications, several instances of a distributed CFD code run in parallel and are dynamically driven by a minimisation algorithm. Another significant application is the coupling of an adaptive 2D surface biosphere model to different parallel atmosphere circulation limited area models. In such a case, not only the dynamic features of PALM can easily take into account the adaptive model, but also the compact syntax used to describe data exchanges allows for a quite generic implementation with different atmosphere models. Some full size, near real time applications, like the operational ocean data assimilation and forecast suite of the MERCATOR operational oceanography centre or the air quality data assimilation and forecasting system Valentina, based on the MOCAGE chemistry and transport model, provide a very satisfactory test bench for the PALM performances in large scale parallel applications. Finally we should mention the recent use of PALM for the implementation of a demonstrative data assimilation suite based on a 1D hydraulic model used in flood forecasting. The graphic algorithm representation proves to be a very useful pedagogical tool. Furthermore, the generic formalism allows for the application of the demo suite to real life applications with no changes in the code lines.

The open source distribution of O-PALM is the most suitable environment to accept collaborations and contributions on the coupler development. Among the most important technical challenges for the evolution of O-PALM towards exascale applications, there is the search for the best trade-off between a centralised and a fully distributed approach. If on one side process management and monitoring is a key issue for dynamic coupling, it imposes an overhead and a risk of bottlenecks for MPP applications. Optimisations bypassing the PALM scheduler and launcher are under study, but they have to keep the capability of reorganizing the layout of the application in case, for instance, of automatic load balancing or adaptive meshes. The same considerations apply for the communication handling: to obtain very effective parallel communications on MPP configurations, we'll have to look the best compromise between flexibility and monitoring on one side and performances on the other.

In particular, for climate modelling applications, the use of O-PALM and the CWIPI interpolation library (with specific enhancements) has to be thoroughly studied and evaluated on test cases of increasing complexity and size.

OASIS, a coupler for climate modelling

by Sophie Valcke (CERFACS)

1. Design goals and strategy

In 1991, CERFACS began to contribute to climate modelling by assembling ocean and atmosphere General Circulation Models developed independently by different groups. After an initial period of investigation, a decision was made to implement a technical coupling layer between the ocean and atmosphere components in the form of an external coupler, i.e. a separate executable performing the regridding of the coupling fields and a coupling library linked to the component codes. This choice ensured a minimal level of interference in the existing codes (low intrusiveness) while focussing on modularity and portability. Two years later, a first version of the OASIS coupler was distributed to the community. The OASIS3 version [Valcke2006], widely used in the climate modelling community today, is the direct evolution of this first version. In 2001, in the framework of the EU PRISM[3] project, during which active collaboration took place with NEC Laboratories Europe (NLE-IT), SGI and the French Centre National de la Recherche Scientifique (CNRS), the development of a new fully parallel coupler, OASIS4 [Redler2010], began targeting higher resolution climate simulations on massively parallel platforms. Parallelism and efficiency drove OASIS4 developments, but the concepts of portability, flexibility and low intrusiveness that made OASIS3 a success were maintained.

2. Implementation

OASIS3 and OASIS4 are portable sets of Fortran and C routines. After compilation, they form a separate Driver & Transformer executable (D&T) and a model coupling interface library, the PSMILe, that needs to be linked to and used by the component models.

Coupling configuration

At run time, the D&T first reads the coupled run configuration defined by the user and distributes the corresponding information to the different component models. This user-defined configuration contains all coupling options for a particular coupled run, e.g. the source and target components, the exchange period, and the regridding chosen for each coupling exchange. During the run, the Driver-Transformer executable and the component model's coupling interface perform appropriate exchanges based on this configuration. With OASIS3, the configuration information is contained in a text file while with OASIS4 it is provided in XML[4] files. A Graphical User Interface (GUI) facilitates the creation of those XML files.

Process management

In a coupled run using OASIS3 or OASIS4, the component models generally remain separate executables with main characteristics, such as the general code structure and memory management, untouched with respect to the uncoupled mode. It is therefore the user's responsibility to ensure that the component models coherently define some global parameters such as the total run duration, the calendar, etc. If a complete implementation of the MPI2 is

3 http://prism.enes.org

4 Extensible Markup Language, http://www.w3.org/XML/

available, the user has to start only the OASIS D&T which then launches all remaining component executables; the advantage of the MPI2 approach is that each component keeps its own internal communication context unchanged with respect to the standalone mode. If only MPI1 is available, the OASIS Driver and the component model executables must be all started at once in the job script in a "multiple program multiple data" (MPMD) mode; in this case, OASIS needs to recreate a component model communicator that must be used by the component model for its own internal parallelisation

Communication: the OASIS PSMILe library

The OASIS3 PSMILe *prism_put* interface (see below) provides a method to send a field into the OASIS3 D&T which then can gather the whole coupling field, transform or regrid it, and redistribute it to the target component model processes. The OASIS3 D&T can be parallel, each process treating a subset of complete coupling fields; this results in a pseudo-parallelisation of OASIS3 D&T on a field-per-field basis.

With the OASIS4 PSMILe, the communication is more efficient. In a first step, envelopes of the grid partitions residing on each process are exchanged between source and target processes and intersections are identified. For each target point falling into an intersection, a multigrid algorithm with a refinement factor of two in each direction is then used to perform the neighbourhood search, i.e. to identify the source cell containing the projection of the target point and its source neighbours. When the source grid is partitioned, the OASIS4 PSMILe performs an additional search step called the *parallel global search*: for the target grid points falling near the source partition border, the neighbours are also searched for on adjacent partitions. This ensures that the regridding result is independent of the source partitioning. At the end of the neighbourhood search, each source process holds different lists, each list containing the information about the target points located in the intersection of a target process domain with its local domain and about the source neighbour points needed for the regridding of these target points. These lists are equally distributed over the D&T processes, resulting in an effective parallelisation of the D&T over the lists. During the exchange phase, each D&T process receives the grid point field values corresponding to its list(s), calculates the regridding weights and applies the weights. The data are sent upon request from the respective target process (i.e. when a *prism_get* is called in the target component code). The OASIS4 D&T therefore acts as a parallel buffer into which the transformations take place.

Coupling field transformation and regridding

OASIS3 offers different transformations for 2D coupling fields expressed on grids in the Earth spherical coordinate system that are regular in longitude and latitude, stretched, rotated, Gaussian reduced, and unstructured. The regridding algorithms available, taken from the Spherical Coordinate Remapping and Interpolation Package (SCRIP) library [Jones1999] are distance-weighted-nearest-neighbour, bilinear, bicubic, conservative remapping and user-defined regridding (the weights and addresses are pre-defined by the user in an external file). Additional transformations such as time accumulation or averaging, correction with external data read from a file, linear combination with other coupling fields, addition or multiplication by a scalar and global conservation are also available.

OASIS4 offers the same regridding options as OASIS3 for 2D coupling fields. In addition, OASIS4 supports 3D distance-weighted-nearest-neighbour and trilinear regriddings that are 3D extensions of the 2D SCRIP algorithms. The 3D implementations are currently being validated. Time accumulation or averaging, addition or multiplication by a scalar, and gathering/scattering (required when the grid definition includes all masked and non masked points but when the coupling field itself gathers only non masked points) are also available. An important limitation of OASIS4 is that unstructured grids are not yet supported.

3. How to use the software, community

To communicate with other component models, a component model needs to call few specific OASIS PSMILe routines. The PSMILe API function calls for both OASIS3 and OASIS4 can be split into three phases. The first phase includes calls for the coupling initialisation, the definition of the grids (i.e. the grid point and corner longitudes and latitudes), the description of the local partition in a global index space, and the coupling field declaration; the second phase comprises receiving and sending of the coupling fields (by calling respectively a *prism_get* or a *prism_put* routine) usually implemented in the model timestepping loop, while the third phase terminates the coupling. For both OASIS3 and OASIS4, the sending and receiving of data follow the principle of “end-point" data exchange. The target component of a *prism_put* or the source of a *prism_get* for each field as well as the exchange frequency is defined by the user in the configuration file and the coupling exchanges take place according to the user external specifications. The target and the source can be another component or a file as the PSMILe library also supports disk I/O based on GFDL mpp_io library [Balaji2001].

Today, both the widely used OASIS3 coupler and the new fully parallel OASIS4 coupler are available. OASIS3 is used today by about 30 different climate modelling groups in Europe, Australia, Asia and North America. The current user community of OASIS4 is growing, and use of OASIS4 has already shown promising results in different configurations. OASIS4 has been used for 3D coupling between atmosphere and atmospheric chemistry models at ECMWF, KNMI and Météo-France in the framework of the EU GEMS project and is still used in the following EU MACC project. Currently, OASIS4 is used at SMHI for regional ocean-atmosphere coupling applied to the Arctic region, at the Bureau of Meteorology (BoM) in Australia for regional ocean-atmosphere coupling, and at the Alfred Wegener Institute, (Bremerhaven, Germany) for 2D global ocean-atmosphere coupling. Global ocean-atmosphere coupled models are also being currently set-up with OASIS4 at the MPI-M and at CERFACS.

4. Benefits and limitation

OASIS3 performances

OASIS success up to now can be explained by its great flexibility, by its low intrusiveness in the component codes, by the active support offered by the development team to the users, and the great care taken to constantly integrate the community developments in the official version.

The OASIS3 coupler is certainly limited in parallelism and will eventually become a bottleneck in the simulation on massively-parallel platforms. However, thanks to its pseudo-parallelisation on a field-per-field basis, OASIS3 has been used recently in a few high-resolution coupled simulations without introducing significant overhead in the simulation elapsed time. For example, OASIS3 is used in the high-resolution version of the Hadley Centre coupled model to couple the atmospheric Unified Model (UM) with a horizontal resolution of 432 x 325 grid points to the ocean NEMO model at a horizontal resolution of ¼ degree; the coupling frequency is 3 hours, XXX coupling fields are exchanged, and the coupled model is run on an IBM power6 192 cpus for the UM, 88 cpus for NEMO, and 8 cpus for OASIS3. In this configuration, the coupling overhead was observed to be less than 2% in elapsed time.

Recently, the resolution of EC-Earth was increased for the atmospheric model IFS to T799 (~25 km) and to the ORCA0.25 configuration (~¼ degree) for NEMO ocean model. 20 coupling fields were exchanged at a coupling frequency of 3 hours. This was run on the Ekman cluster (1268 nodes of 2 quadripro AMD Opteron, i.e. a total of 10144 cores) with different numbers of cores for IFS, NEMO and OASIS3. It was observed that OASIS3 elapse time of ~6 seconds is non negligible when it runs in mono-processor mode. In this case, the coupling induces a significant overhead of ~13% in elapse time with respect to the IFS standalone run; this is true even if OASIS3 interpolates the fields when the fastest component waits for the slowest as the OASIS3 cost itself is larger than the component imbalance. But when the parallelism of OASIS3 increases (going from 1 to 10 processes), OASIS3 elapsed time decreases and its cost is nearly "hidden" by the component imbalance. In this case, the overhead decreases to less than 3%. Of course, this way of "hiding" the cost of OASIS3 works only if there is some imbalance of the components elapsed time which allows OASIS3 to interpolate the fields when the fastest component waits for the slowest. If the components were perfectly load balanced, then the OASIS3 cost, even if lower when OASIS3 is used in the pseudo-parallel mode, would be directly added in the coupled model elapse time.

OASIS4 performances

The performance of the OASIS4 multi-grid search was analysed in detail by comparing it to the OASIS3 sequential search (see [Redler2010]). Even at relatively low resolution (2244 and 4692 grid points for the atmosphere and the ocean), it was observed that OASIS3 is about two times slower than OASIS4. The difference gets bigger with increasing resolution: in fact, the time required for the neighbourhood search increases with $O(N^2)$ for OASIS3 where as it increases only with $O(N)$ for OASIS4. With ~300 000 grid points for the atmosphere and for the ocean, the search in OASIS3 is about 170 slower than the search in OASIS4. This clearly demonstrates the benefit of the multi-grid neighbourhood search when compared to a classical search and the increased general performance of OASIS4 over OASIS3 even in this simple non-parallel case.

Regarding the scalability of OASIS4, some first tests of the PSMILe library and D&T scalability were done and are reported in [Redler2010]. Up to 16 cpus, the PSMILe and the D&T show a good scalability. These first tests on the PSMILe and the Transformer scalability are encouraging and can be used as a proof-of-concept. Additional tests on much greater numbers of processes will need to be carried out before any firm conclusions can be drawn.

5. Future plans

In conclusion, one can say that OASIS3 is stable and well debugged, but it is more performance limited than OASIS4, which continues to undergo validation, especially in the fully parallel cases. Within the framework of funded projects work continues to establish comprehensive services around OASIS through a portal offering documentation, user guides, tutorial, FAQs, user forum and tips for best practices, and to extend the existing functionality. One example of such an initiative is the InfraStructure for the European Network for Earth System Modelling (IS-ENES), a 4-year project started in March 2009 that brings about 90 person-months of funding for OASIS development and user support, and into which a fruitful collaboration with the Deutsches Klimarechenzentrum GmbH (DKRZ) is currently taking place.

Currently, CERFACS and CNRS are committed to support the development and maintenance of the OASIS software. However, CERFACS and CNRS permanent resources devoted to the OASIS development are most probably undersized given the large OASIS user community and the always evolving complexity of computing platforms used in climate modelling. Therefore CERFACS's current objective is to establish an official Memorandum of Understanding between the largest institutions using OASIS into which each partner would engage in spending some permanent resources on OASIS. Thanks to this MoU, OASIS hopefully will remain for the coming years a great example of successful community software.

Acknowledgements: Part of this work has been funded by the EU FP7 IS-ENES project (Contract GA No: 228203).

References

- [Redler2010] R. Redler and S. Valcke, 2010. OASIS4, A Coupling Software for Next Generation Earth System Modelling. Geosci. Model Dev., 3, 87-104.
- [Valcke2006] S. Valcke, 2006. OASIS3 User Guide (prism_2-5), Technical Report TR/CMGC/06/73, PRISM Report No 2, CERFACS, Toulouse, France. 60 pp
- [Balaji2001] Balaji, 2001: Parallel Numerical Kernels for Climate Models, ECMWF TeraComputing Workshop 2001, World Scientific Press, Reading.
- [Jones1999] Jones, P.: Conservative remapping: First- and second-order conservative remapping, Monthly Weather Review, 127, 2204–2210, 1999.

The GFDL Flexible Modeling System FMS

by Balaji (Princeton University)

The GFDL Flexible Modeling System (FMS) is an early example of a modeling framework, a comprehensive programming model and toolkit for the construction of coupled climate models. The "sandwich" architecture is fairly typical of such frameworks. User code, that is to say a set of routines expressing scientific algorithms, is written following the conventions of a standard infrastructure layer that provides useful and common technical services such as I/O, exception handling, and most importantly, operations on distributed grids and fields. Such standard high level expressions of parallelism, independent of the underlying hardware architecture, and uniformly expressed on all platforms, are an area of keen research interest. Balaji and Numrich (2005) provide an overview of the field

The climate system is composed of hierarchies of interacting physical components, and it is natural to think of constructing models of such systems out of interacting code components. Component-based design of model codes is based on defining standards for what a component interface looks like: a community of scientists and developers that agree to adhere to a given standard component interface (SCI) set can then distribute development amongst themselves, confident that their own independently developed components will interact correctly with others within the same modeling framework.

The FMS coupler is a domain-specific SCI: it is written quite narrowly to support ESMs. It is designed to address the question of how different components of the Earth system, say atmosphere and ocean, are discretized. Earlier generations of climate models used the same discretization, or simple integer refinement, for all components: thus, data exchange between components was a relatively simple point-to-point exchange. But any limitation on resolution of one component necessarily imposed itself on the other as well. Now it is increasingly common for each model component to make independent discretization choices appropriate to the particular physical component being modeled. In this case, how is, say a sea surface temperature from an ocean model made available to an atmosphere model that will use it as a boundary condition on a different spatial grid ?

This is the regridding problem, subject to the following constraints when specialized to Earth system models:

- Quantities must capable of being globally conserved : if there is a flux of a quantity across an interface, it must be passed conservatively from one component to the other. This consideration is less stringent when modeling weather or short-term (intraseasonal to interannual) climate variability, but very important in models of secular climate change, where integration times can be $O(10^6) - O(10^8)$ timesteps.

- The numerics of the flux exchange must be stable, so that no limitation on the individual component timestep is imposed by the boundary flux computation itself.

- There must be no restrictions on the discretization of a component model. In particular, resolution or alignment of coordinate lines cannot be externally imposed. This also implies a requirement for higher-order interpolation schemes, as low-order schemes work poorly between grids with a highly skewed resolution ratio. Higher-order schemes may require that not only fluxes, but their higher-order spatial derivatives as well, be made available to regridding algorithms. The independent discretization requirement extends to the time axis: component models may have independent timesteps. (We do have a current restriction that a coupling timestep be an integral multiple of any individual model timestep, and thus, timesteps of exchanging components may not be co-prime).

• The exchange must take place in a manner consistent with all physical processes occurring near the component surface. This requirement is highlighted because of the unique physical processes invoked near the planetary surface: in the atmospheric and oceanic boundary layers, as well as in sea ice and the land surface, both biosphere and hydrology.

• Finally, we require computational efficiency on parallel hardware: a solution that is not rate-limiting at the scalability limits of individual model components. Components may be scheduled serially or concurrently between coupling events.

The specificity of the problem that the FMS coupler is designed to address distinguishes it from more general component frameworks such as ESMF. Unlike an ESMF application, which can be recursively constituted out of components performing any function at all, the FMS coupler recognizes only a few components that may be on independent grids: an atmosphere, an ocean surface, a land surface, and an ocean. The ocean surface also represents the sea ice. Any other components inherit a grid from these, e.g atmospheric physics and chemistry from the atmosphere; terrestrial biosphere, river and land ice components from the land surface; marine biogeochemistry from the ocean.

The SCI for FMS is therefore not phrased in terms of a generic "component" as in ESMF. Instead, there are interfaces or "slots" for each of the specific components listed above. For instance, an ocean model would encode its state in terms of specific data structures to hold the fields it exchanges with other components, called ocean boundary type and ocean data type. It must provide calls named ocean model init and ocean model end for initialization and termination, and a routine called update ocean model that steps the model forward for one coupling timestep. These calls all have a specific syntax. Each slot also includes the possibility of a null or "stub" component if that component is not needed, as well as a "data" component (where for instance the ocean is replaced by a dataset). In addition we provide a data override capability for fine-tuned sensitivity studies, where individual fields in the model can be overridden by a dataset.

Fluxes at the surface often need to be treated using an implicit timestep. Vertical diffusion in an atmospheric model is generally treated implicitly, and stability is enhanced by computing the flux at the surface implicitly along with the diffusive fluxes in the interior. Simultaneously we must allow for the possibility that the surface layers in the land or sea ice have vanishingly small heat capacity. This feature is key in the design of the FMS coupler. This is a tridiagonal matrix inversion which can be solved relatively efficiently using an up-down sweep. The problem is that some of the layers are the atmosphere and others are in the land. Moreover, if the components are on independent grids, the key flux computation at the surface, to which the whole calculation is exquisitely sensitive, is a physical process (e.g Monin and Obukhov, 1954) that must be modeled on the finest possible grid without averaging. Thus, the exchange grid, on which this computation is performed, emerges as an independent model component for modeling the surface boundary layer.

The general procedure for solving vertical diffusion is thus split into separate up and down steps. Vertically diffused quantities are partially solved in the atmosphere (known in FMS as the "atmosphere down" step) and then handed off to the exchange grid, where fluxes are computed. The land or ocean surface models recover the values from the exchange grid and continue the diffusion calculation and return values to the exchange grid. The computation is then completed in the up-sweep of the atmosphere. Note that though we are computing vertical diffusion, some spurious horizontal mixing can occur as the result of regridding. The exchange grid is described in detail in Balaji et al (2006).

Data assimilation for coupled models is an exciting emergent field of research. Data assimilation includes a class of methods known as ensemble filters (Kalnay, 2002), which involve sampling the error space of observations by running a model ensemble: multiple copies of a model perturbed to span that space.

The FMS coupled modeling system includes a sophisticated data assimilation system, the parallel ensemble adjustment Kalman filter (Zhang et al, 2005). The parallel filter consists of running each ensemble member as a concurrent component on an independent set of processors, and the slot replaced by the filter.

The ensemble filter has been used on 24-member ensembles of CM2.1 with no loss of performance versus ensemble size. A recent development has been the construction of a coupled data assimilation system (CDAS) where ensemble methods simultaneously assimilate both atmosphere and ocean data.). The CDAS has been the basis for a set of pioneering studies showing the influence of initial conditions on simulations of recent climate history on decadal timescales.

In summary, this section has presented a review of the key features of how coupling is performed in the GFDL Flexible Modeling System. A standard coupling interface with slots for atmosphere, land surface, ocean surface, and ocean components is coupled along with a surface boundary layer component on an exchange grid (Balaji et al, 2006). Components live within a single executable, but can be scheduled serially or concurrently with others. The code has been shown to be scalable to O(1000) processors, with fast surface processes coupling every atmospheric timestep (typically □ 15 min) and slow processes coupling every ocean timestep (typically 1 hour). Coupling is conservative to up to second-order accuracy. The FMS superstructure also includes support for data assimilation using ensemble filter methods. The coupled data assimilation system has been run on IPCC-class models assimilating both atmosphere and ocean fields. An ensemble size of up to 24 has been used with no significant loss of scaling.

At the moment of writing, FMS and its coupler have been in active use for over a decade. Its feature list and its performance still place it at the forefront of the field.

References

- Balaji V, Numrich RW (2005) A Uniform Memory Model for Distributed Data

Objects on Parallel Architectures. In: Zwieflhofer W, Mozdzynski G (eds) Use of High-Performance Computing in Meteorology, World Scientific Publishing Co., pp 272–294

- Monin, A. and A. Obukhov,1954. Basic laws of turbulent mixing in the ground layer of the atmosphere. Tr Geofiz Inst Akad Nauk SSSR 151:163–187
- Balaji V., J. Anderson, I. Held, M. Winton, J. Durachta, S. Malyshev, R. J. Stouffer, 2006. The Exchange Grid: a mechanism for data exchange between Earth System components on independent grids. In: Deane A, Brenner G, Ecer A, Emerson D, McDonough J, Periaux J, Satofuka N, , Tromeur-Dervout D (eds) Parallel Computational Fluid Dynamics: Theory and Applications, Proceedings of the 2005 International Conference on Parallel Computational Fluid Dynamics, May 24-27, College Park, MD, USA, Elsevier
- Kalnay E (2002) Atmospheric Modeling, Data Assimilation and Predictability. Cambridge University Press
- Zhang S., M.J. Harrison, A.T. Wittenberg, A. Rosati, J.L. Anderson, V. Balaji, 2005. Initialization of an ENSO Forecast System using a Parallelized En- semble Filter. Mon Wea Rev 133:3176–3201

The Bespoke Framework Generator BFG

by Rupert Ford (U. Manchester)

1. Design goals and strategy

BFG was originally developed as a potential solution to an analysis of the Met Office's coupling requirements [REQS]. These requirements included high performance (in-part to support both the Climate and NWP communities), flexibility (in terms of coupling models together and integrating external models) and future-proofing (to avoid major changes to the scientific software in the future). BFG was subsequently extended to support the requirements of the GENIE community model [BFG2] and the CIAS Integrated Assessment System [CIAS].

Rather than being a coupler in its own right, BFG is designed to allow the user to choose the coupling technology, i.e. a specific coupler and/or communications infrastructure, they would like to use for a coupled model run. Given the required information, in the form of metadata, BFG generates bespoke wrapper code which can be compiled and linked with the users science code and the coupling technology of choice. Regardless of which coupling technology the user chooses for their coupled model run, the scientific code remains unchanged. BFG can, therefore, be thought of as a Meta-coupler.

By separating the implementation of the coupler from the science code the user is given an additional layer of flexibility. This flexibility can help in terms of portability, performance, maintenance and future-proofing of the code.

In BFG code developers are encouraged to input and output coupling data in their internal storage format. A consequence of this philosophy is that data may need to be transformed when being transferred between models (one important class being re-gridding in ESM). In BFG, transformation code is specified and treated in the same way as model code. This approach allows transformations to be mapped to the underlying resources in the most appropriate manner. In the case that the target framework supports intrinsic transformations (such as OASIS4) then these transformations can be indicated as being intrinsic in the BFG metadata and BFG will generate appropriate code (or configuration files) to use these.

BFG concepts have been purposely designed to be relatively generic. A notable example is the model interface where one is able to specify models written in a variety of languages. A related design philosophy is to avoid requiring any domain specific features and to treat them as optional additions (for example grids in ESM). Thus BFG should be applicable to other domains and also between domains. Early indications are that this is indeed the case [CIAS][MD].

2. Implementation (regarding process management, communication/data exchange, regridding)

The current version of BFG (BFG2) supports models written in Fortran90. Each model must be written as a module containing one or more subroutines. BFG uses the associated metadata to generate the required calling (control) code. The coupled model behaviour is specified by a schedule described in the metadata which supports arbitrary nested loops.

As models are written as modules, BFG2 is also able to map models within the coupled model to a single executable, multiple executables or any combination thereof.

BFG2 supports data being passed to and from module subroutines via arguments and/or in-place put/get calls. The former approach is similar to that used by ESMF and CPL7 and the latter to OASIS3, OASIS4 and TDT. Each coupling connection can be initialised (primed) in a variety of ways including from a model or a file.

In the current implementation, data can be passed between models in the following ways:

- Argument passing
- MPI
- OASIS4 calls.

The OASIS4 implementation also supports the specification of grids using an XML representation of the Gridspec [REF] and the use of intrinsic OASIS4 transformations.

3. How to use the software, community

To use BFG2 you must make your model code conform to the coding rules mentioned earlier and describe your models interface in (definition) metadata. You then specify how the models (and transformations) are connected together scientifically (the composition) and finally you specify how to map the models onto the available hardware and software resources (the deployment). BFG2 takes the metadata descriptions and translates these descriptions to appropriate (bespoke) code using xsl transformations with python wrappers.

BFG has thus far been primarily a prototyping tool. Currently its one use is within CIAS, a Community Integrated Assessment System [CIAS] where it is used to couple over 20 different model configurations. However, BFG2 is maturing into a tool that could be used more widely.

4. Benefits and limitations

The BFG approach has the primary benefit of combining the isolation of the science from the infrastructure and the implementation of a code generation system to provide flexibility in model composition and deployment onto the available hardware and software resources.

One key feature of BFG is that it is able to achieve the same performance as hand written code [BFG2]. The API and the high performance offered by BFG opens up the opportunity for much finer grain coupling than is typically performed at the moment.

The current BFG2 implementation has a number of limitations:

- One needs to regenerate the framework code whenever any part of the coupling metadata changes
- The xslt generation software is complex and difficult to manage
- BFG2 does not currently support data partition information which means it can not support parallel models which are the norm in ESM
- BFG2 only supports one target in a coupled model (with the notable exception of argument passing) so it is not possible to pass data using ESMF and OASIS4 (for example) in the same coupled model.

5. Future plans

We are currently working on extending BFG2 to support models that have been written in a less modular way. In particular, codes which are main programs, codes with internal control and codes where the source code is not available and must, therefore, be treated as a black box. We are also working on extending BFG2 to support a number of different languages in order to satisfy the requirements of the IAM community. As an example, economics models are typically written in GAMS. We are planning to add support for parallel partitions and subsequently parallel models. For the MPI target we will use MCT [REF] for the resultant "mxn" communication that will occur. We are also planning to extend BFG2 to support ESMF, CPL7 and TDT as targets.

Finally, in the slightly longer term, we are interested in the feasibility of using BFG2 to couple models that are coded to conform to other frameworks by generating appropriate adaptor code.

References

[BFG2] C. W. Armstrong, R. W. Ford and G. D. Riley, Coupling integrated Earth System Model components with BFG2, Concurrency and Computation: Practice and Experience, Vol. 21 No. 6, pp. 767--791, 2009, DOI: 10.1002/cpe.1348.

[CIAS] R. Warren, S. de la Nava Santos, N.W. Arnell, M. Bane, T. Barker, C. Barton, R. Ford, H.M. Fussel, Robin K.S. Hankin, Rupert Klein, C. Linstead, J. Kohler, T.D. Mitchell, T.J. Osborn, H. Pan, S.C.B. Raper, G. Riley, H.J. Schellnhuber, S. Winne, D. Anderson. Development and illustrative outputs of the Community Integrated Assessment System (CIAS), a multi-institutional modular integrated assessment approach for modelling climate change. In J. Environmental Modelling and Software, Vol. 23, No. 5, May 2008. ISSN: 1364-8152.

[BFG1] R.W. Ford, G.D. Riley, M.K. Bane, C.W. Armstrong and T.L. Freeman, 'GCF: A General Coupling Framework', Concurrency and Computation: Practice and Experience 18(2), pp. 163--181, 2006.

[MD] R. Delgado-Buscalioni, P.V. Coveney, G.D. Riley and R.W. Ford, 'Hybrid Molecular-Continuum Models under the General Coupling Framework' Philosophical Transactions of the Royal Society of London Series A, 363(1833), pp. 1975--1985, 2005.

[REQS] R.W. Ford and G.D. Riley, Towards the Flexible Composition and Deployment of Coupled Models. In proc. Tenth ECMWF Workshop on the Use of High Performance Computing in Meteorology; Realizing TeraComputing. ECMWF, Reading, England, 4-8 November 2002. World Scientific, pp. 189--195, 2003.

The OpenMI interface for flexible and dynamic coupling

by Stef Hummel and Bert Jagers (Deltares)

Introduction

Integrated analysis often requires integrated modeling. This can be done by developing all-inclusive models, but it is preferable to be able to flexibly combine individual models or model components, that address specific domains, at run time. This can be realized by implementing models as shared libraries with a common standardized interface. In the water sector, in a series of EU-projects that focused on river basin management, the Open Modeling Interface (OpenMI, see [1] and [2]) has been developed in order to link together model components from various origins. OpenMI provides a standard model interface, a reference implementation of that standard, and utilities to support existing models in adhering to that standard. The OpenMI standard is published by the OpenMI Association; the reference implementation and the utilities (the so called SDK, Software Development Kit) are provided as an open source project by the the OATC, OpenMI Association Technical Committee (see [3] and [4]).

The first version of the Open Modeling Interface (OpenMI) was launched at the end of 2005. Since that time, the user and development community has grown steadily, and various well known models have become compliant. Because of limitations of this first version, some of the models did not follow the OpenMI standard interfaces exactly, but used slight deviations to achieve their goal in a similar style. Improvements were necessary to become a general interface standard that would not only cover water related applications, but also other domains. Over the past few years, starting in 2007, a core group of six institutes has worked on an upgrade of the OpenMI towards version 2.0. Based on a limited number of use cases as general guidance for improvement, a long list of improvements was composed. These changes have made OpenMI suitable for a large range of applications, from non-time dependent Geographical Information Systems (GIS) towards e.g. master-slave controlled modeling frameworks. The resulting version 2.0 of the OpenMI standard (see [5] and [6]) has been released in December 2010.

OpenMI concepts

OpenMI provides standardized interfaces to define, describe and transfer data between software components that run simultaneously, thus supporting systems where feedback between the modeled processes is necessary in order to achieve physically sound results. A software component that implements OpenMI standard is called a Linkable Component. OpenMI allows the linking of models with different spatial and temporal representations: for example, linking river models and groundwater models, where the river model typically uses a one-dimensional grid and a short time step and the groundwater model uses a two- or three-dimensional grid and a longer time step.

The OpenMI standard consists of a set of interface classes, specified in both Java and C#, that define the behavior of a model component, and that define which quantities can be exchanged by that component, on which locations and in what time frame.

What, where, when

The run time data exchange between model components is done by means of a GetValues(…) call, where the argument of this call specifies:

- *What* is exchanged? This is defined by the *IQuantity* and the *IQuality* interfaces below.
- *Where* is it exchanged? The location is specified by the so called *IElementSet*, a set of ID-based or Geo-referenced locations (see table below).
- *When*, i.e. at what times is the data needed? This is expressed by the *ITimeSet*, a list of time stamps or time spans.

A *quantity* is specified by

- Caption ("Runoff")
- Description (optional explanatory description)
- Value Type (double, integer, etc.)
- Unit:
    - Caption ("CFS")
    - Description ("Cubic feet per second")
    - ConversionFactorToSI (0.0283168439)
    - OffsetToSI (0)
    - Dimension (e.g. $L^3 T^{-1}$)

A *quality* is defined by its:

- Caption ("Soil Type")
- Description (optional explanatory description)
- Categories:
    - Caption ("sand 1")
    - Description ("coarse sand")
- IsOrdered

For the definition of locations, the *ElementSet*, various types are available:

ElementType	*Description*
IDBased	ID-based (string comparison).
Point	geo-referenced point in the horizontal (XY)-plane or in in the 3-dimensional (XYZ)-space.
PolyLine	geo-referenced polyline connecting at least two vertices in the horizontal (XY)-plane or in the 3-dimensional (XYZ)-space. The begin- and end-vertex indicate the direction of any fluxes. Open entity with begin- and end-vertex not being identical.
Polygon	geo-referenced polygons in the horizontal (XY)-plane or in the 3-dimensional (XYZ)-space. Vertices defined anti-clockwise. Closed entity with one face, begin- and end-vertex being identical.

Linking components

A component specifies its data exchange capabilities by defining input items and output items. After initialization, the lists of input and output items supported by the Linkable Component can be queried via the OpenMI interface. Each input item and output item specifies its quantity or quality, its element set and its time set.

The actual data exchange between components is established by defining provider/consumer relationships between output items and input items (see Figure 1). The GetValues() call mentioned above is performed on the output items.

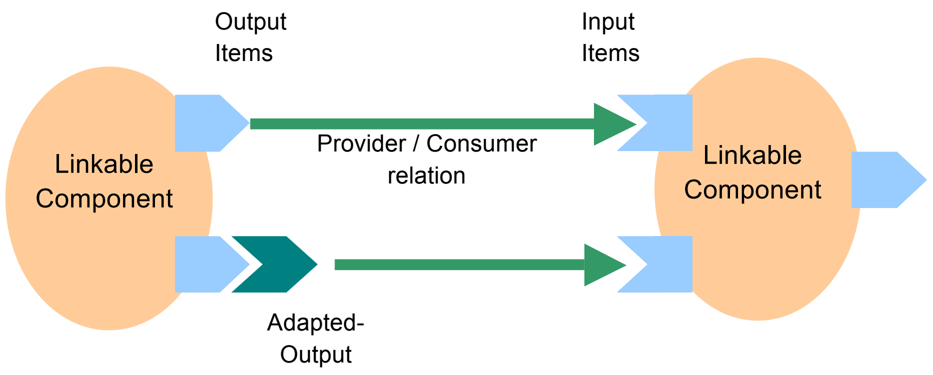

Figure 1: Linking output and input items

If the quantity, the ElementSet or the TimeSet of a certain output does not fit the way the input item requires it, the output can be adapted by adding an AdaptedOutput to the output. As the name indicates, an adapted output in its turn is just an output again, so subsequent adapted output items can be added to the initially created adapted output.

Figure 2 shows an example of some sequences of adapted outputs. It may be clear that the adapted output approach offers great flexibility in defining the steps that have to be taken to transform that data from, for example, Output-1 to Input-2 and Input-a.

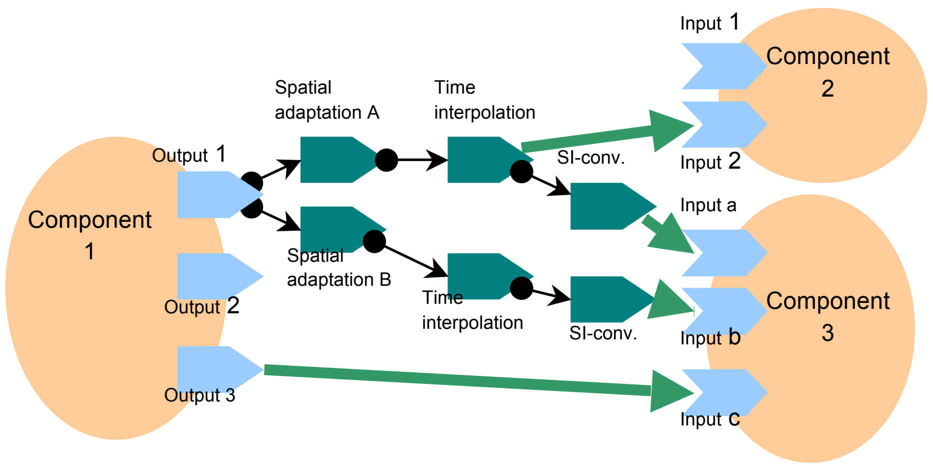

Figure 2: Flexibility in data transformations by means of Adapted Outputs

The first release of OpenMI included only the GetValues() call to transfer data. Although in OpenMI 2.0 a SetValues() call has been added to support e.g. parameter sensitivity studies and data assimilation, the GetValues() call remains the main work horse for OpenMI. It is closely related to the time progress and synchronization of the overall configuration of Linkable Components.

Migrating models

From the very start the OpenMI has been designed in such a way that it supports the easy migration of existing modeling systems. Generally speaking, a model in any programming language is made OpenMI compliant by re-organizing its code in such a way that it has a separate initialization, computation and finalization routine, and can accept and provide input data and results, after which – based on the current reference implementations – a Java and/or C# wrapper is put around it (see Figure 3). The OpenMI SDK offers utilities to facilitate the development of such a wrapper.

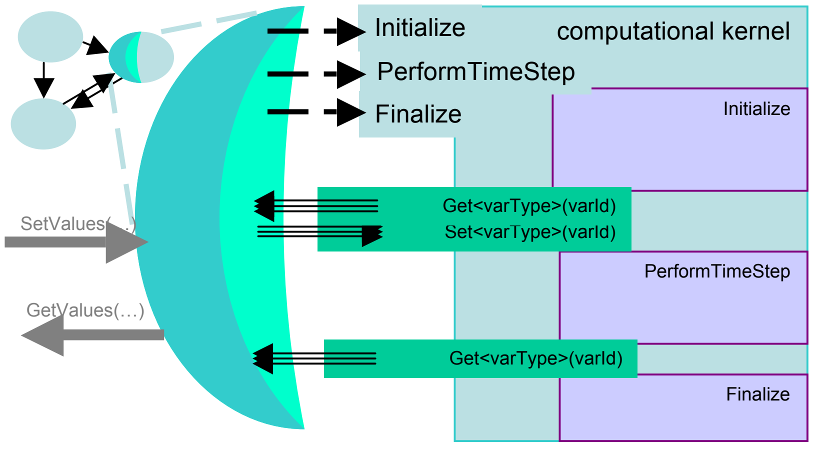

Figure 3: Wrapping native (e.g. Fortran) code in an OpenMI wrapper

References

[1] The OpenMI Standard, published by the OpenMI Association.
Distribution, news and publications: http://www.openmi.org/,
Developers and users: http://wiki.openmi.org/.

[2] Gregersen, J.B. , P.J.A. Gijsbers and S.J.P. Westen (2007) OpenMI: Open modelling interface, Journal of Hydroinformatics Vol.9 No 3. pp 175–191

[3] The OpenMI SDK (System Development Kit), provided by the OpenMI Association Technical Committee [3].
Information on developments: http://wiki.openmi.org/
Source development: http://sourceforge.net/projects/openmi/

[4] The OpenMI Association Technical Committee (AOTC),
OpenMI's core developing team, participants (amongst others):
Deltares (NL), Alterra (NL), DHI (DK), MWHSoft (GB), Bundesanstalt für Wasserbau (D).
http://wiki.openmi.org/OpenMI+Association+Technical+Committee

[5] Gijsbers, P., Hummel S., Vaneçek S., Groos J., Harper A., Knapen R., Gregersen J.

Schade P., Antonelli A., Donchyts, G. (2010) From OpenMI 1.4 to 2.0. International Congress on Environmental Modelling and Software, July 5 - 8 2010, Ottawa, Ontario, Canada

[6] Donchyts, G., Hummel S., Vaneçek S., Groos J., Harper A., Knapen R., Gregersen J. Schade P., Antonelli A., Gijsbers P. (2010) OpenMI 2.0 What's New. International Congress on Environmental Modelling and Software, July 5 - 8 2010, Ottawa, Ontario, Canada

OOPS - An Object Oriented Framework for Coupling Data Assimilation Algorithms to Models

by Mike Fisher (ECMWF)

Data assimilation involves a close coupling between the assimilation algorithm and a numerical model. Observation operators are required to determine model-equivalents of observations, and the assimilation algorithm must integrate the assimilating model (and possibly its tangent linear and adjoint) with specific initial and boundary conditions.

Despite this close coupling with the model, the high-level mathematical description of data assimilation algorithms is essentially model-independent. Moreover, these algorithms can typically be described in terms of a few operators and vectors.

In the IFS/Arpege code, the close connection between model and data assimilation has lead to a single monolithic code in which the boundaries between model and assimilation algorithm have become blurred, and in which it difficult to identify the vectors and operators of the mathematical description of the algorithm. This makes it difficult to develop new algorithms, and does not allow the assimilation code to be used with different models.

The Object Oriented Prediction System (OOPS) attempts to address this problem by clearly identifying the various operators and vectors required by the assimilation algorithm, and providing clean interfaces between these entities and the model. Since these operators and vectors are explicitly available, it is straightforward to re-combine them into new algorithms.

The data assimilation algorithm is expressed in a way that is independent of any specific model. This allows, for example, algorithms to be developed with simple models and then transferred directly, without any re-coding, to more complex models.

C-Coupler: A coupler for Earth System Modeling

by Xiaoge Wang[1], Li Liu[1], Guangwen Yang[1,2]

[1] Department of Computer Science and Technology
[2] Center for Earth System Science
Tsinghua University, Beijing, China, 100084

Abstract: This presentation will introduce the project C-Coupler. It will include an overview of the project, the design goals and technical aspects of C-Coupler.

Project C-Coupler is funded by National High Technology Research and Development Program of China (863 program) started this year for 3 years. Its goal is to support the earth system modeling research. In addition to the functions of a typical coupler, such as data transfer and re-grid between grid components, flux computation, and driver of the system, C-Coupler needs to support ensemble modeling and embedded regional modeling, and to run efficiently on large scale parallel computers.

The earth system modeling research community in China currently uses mainly NCAR cpl5 and cpl6 and OASIS3 and OASIS4 for coupling different component models. The demand for a coupler with more functions and more flexibility and user friendliness is getting higher.

In design of C-Coupler, there are some considerations:

(1) The effort of transform from current couplers to the new one should be minimized.

(2) The advantages of current in-use couplers should be preserved.

(3) The software technologies used in the new coupler should support component-based programming and software configuration management;

(4) The new features include: more configurable (using GUI and script), Interactive ensemble, 3D coupling, regional coupling.

(5) High performance in communication, system level load balancing and parallel I/O operations.

The architecture design of C-Couple is presented in Figure 1 and Figure 2. As shown in Figure 1, the coupled earth system model (ESM) is consisted of three parts: on the top, there are the model components, such as atmosphere model, ocean model, etc.; on the bottom, there are function components and data components; all the components connected to the center part, C-Coupler, via predefined interfaces. C-Coupler is consisted of interfaces to the components, configure files and coupling driver code. The configure files and part of the driver code are generated by configuration system, which is shown in details in Figure 2. As shown in Figure 2, configuration system is used by component builders, ESM builders and users to configure the model components, function components, coupling cases and ESM cases through GUI or script language. The configuration system would access the components repository and manage the configurations. With the capability of configuration and code generation, we expect C-Coupler to be more flexible and user friendly. The coupling function of C-Coupler is also expected to be more extensible.

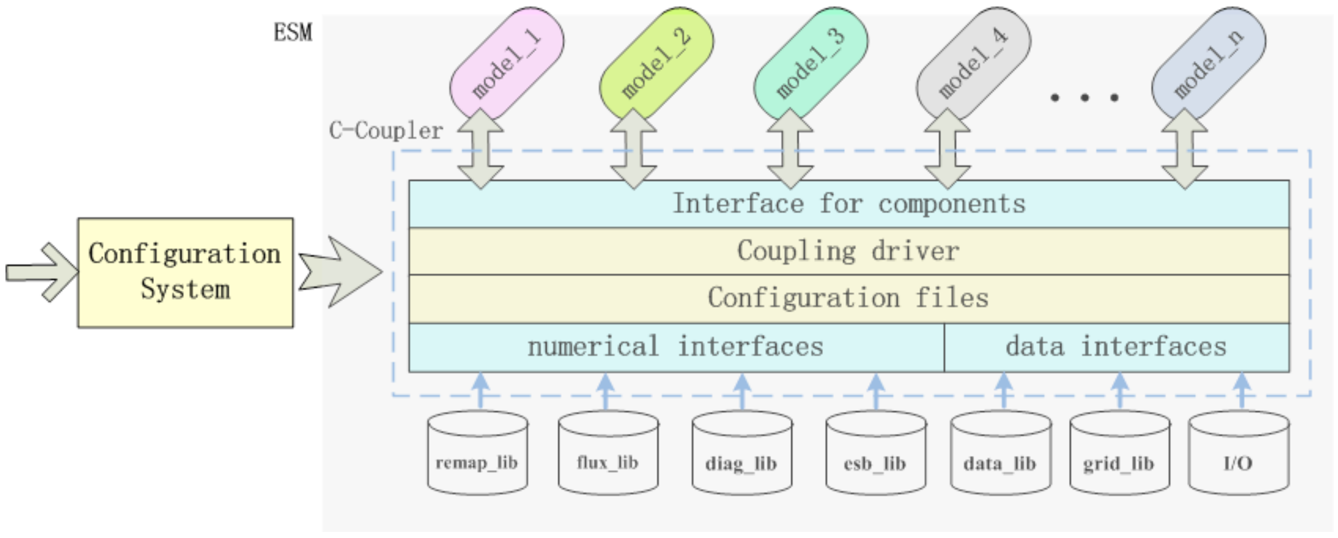

Figure 1: Overview of C-Coupler and coupled earth system model.

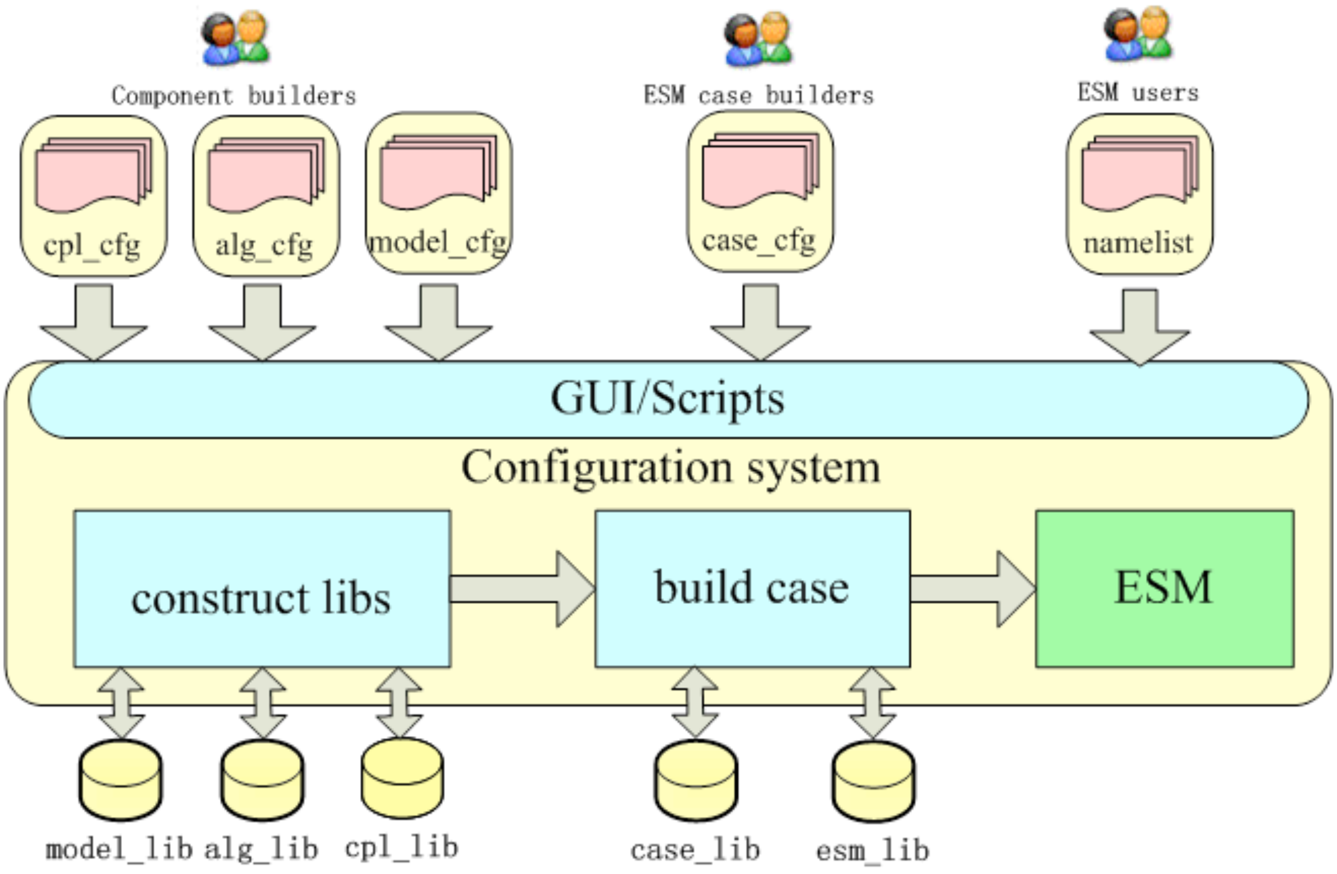

Figure 2: Overview of C-Coupler configuration system

Some details of configuration are presented. They are model configuration, coupling algorithm configuration, coupler configuration and ESM case configuration. With the enough configuration information collected during the configuration stage, the system could generate the sequence of coupling algorithm objects. This sequence would be used to in the C-Coupler driver to control the model coupling. The driver uses time-driven policy to drive the components. It uses a global timer, model components' time steps, coupling frequencies to manage the execution of components.

The implementation of first version of C-Coupler is planed to complete by early next year. It will implement some basic coupling functions that will allow the current version of model FGOALS to run under C-Coupler. Some more advanced functions, such the support for interactive ensemble, 3D remapping, embedded regional model coupling, global load balancing etc., would be implemented in the late version. The evaluation of C-Coupler will follow the implementation step. It will cover the testing and evaluation on the system integrity and robustness. After the first version, more efforts will be put on the system scalability, portability and adaptability.

The Model for Prediction Across Scales: meshes and software framework

Michael Duda[+], Todd Ringler[*], and William Skamarock[+]

[+]National Center for Atmospheric Research[5]
[*]Los Alamos National Laboratories

1. Introduction

The Model for Prediction Across Scales (MPAS) is a collaborative effort between LANL (COSIM) and NCAR (MMM) to develop climate, regional climate, and numerical weather prediction components within a common framework. Currently, a nonhydrostatic atmosphere model and an ocean model are under development in MPAS, and there are plans to develop an ice sheet model in the near future. Although the physical domains over which each of these models simulate are quite distinct, all of the models have in common their use of centroidal Voronoi tessellations (CVT) with a C-grid staggering, i.e., with the prognosed velocity field defined in terms of velocities normal to grid cell faces, as their horizontal meshes. The consequent need for software infrastructure to support finite volume-type modeling on CVT meshes has motivated the development of a common software framework for MPAS. Of particular interest to the scientific goals of MPAS is the ability of CVT meshes to provide smooth mesh refinement according to a user-defined density function, though with this flexibility come challenges for software infrastructure, and, most likely, for model couplers as well. In this presentation, we first describe the construction and use of CVTs in MPAS, and we then outline the MPAS software architecture, pointing out how we anticipate interacting with coupling packages.

2. CVT meshes in MPAS

As their name implies, centroidal Voronoi tessellations are tessellations of a domain where each of the cells is a Voronoi region for some generating point; when the generating points are also the mass centroids of the Voronoi regions with respect to a specified density function, the Voronoi tessellation is a centroidal Voronoi tessellation. A detailed review of CVTs is given in Ju et al. (2010). It is precisely the flexibility to specify the density function that enables MPAS meshes to possess smoothly-changing resolution, and the centroidal requirement of CVTs leads to meshes — both uniform and variable-resolution — of high quality. Figure 1a provides an illustration of an SCVT[6] mesh with higher resolution targeted over a region of the Northern Hemisphere; it is worth noting that the mesh is unstructured, since the cells are not constrained to have a specified number of sides.

For any CVT mesh, the dual mesh, or Delaunay triangulation, provides a connectivity graph of the cells of the mesh, and by applying existing graph partitioning algorithms to the connectivity graph, we arrive at a partition of the cells among processors. The collection of cells in a partition is referred to as a block, and each block is assigned to a parallel task; the parallel decomposition of an SCVT into 64 blocks is illustrated in Figure 1b. In the MPAS architecture, blocks represent the basic level of mesh decomposition.

[5] NCAR is sponsored by the National Science Foundation

[6] An SCVT is is a spherical centroidal Voronoi tessellation, where the generating points are constrained to lie on the surface of a sphere

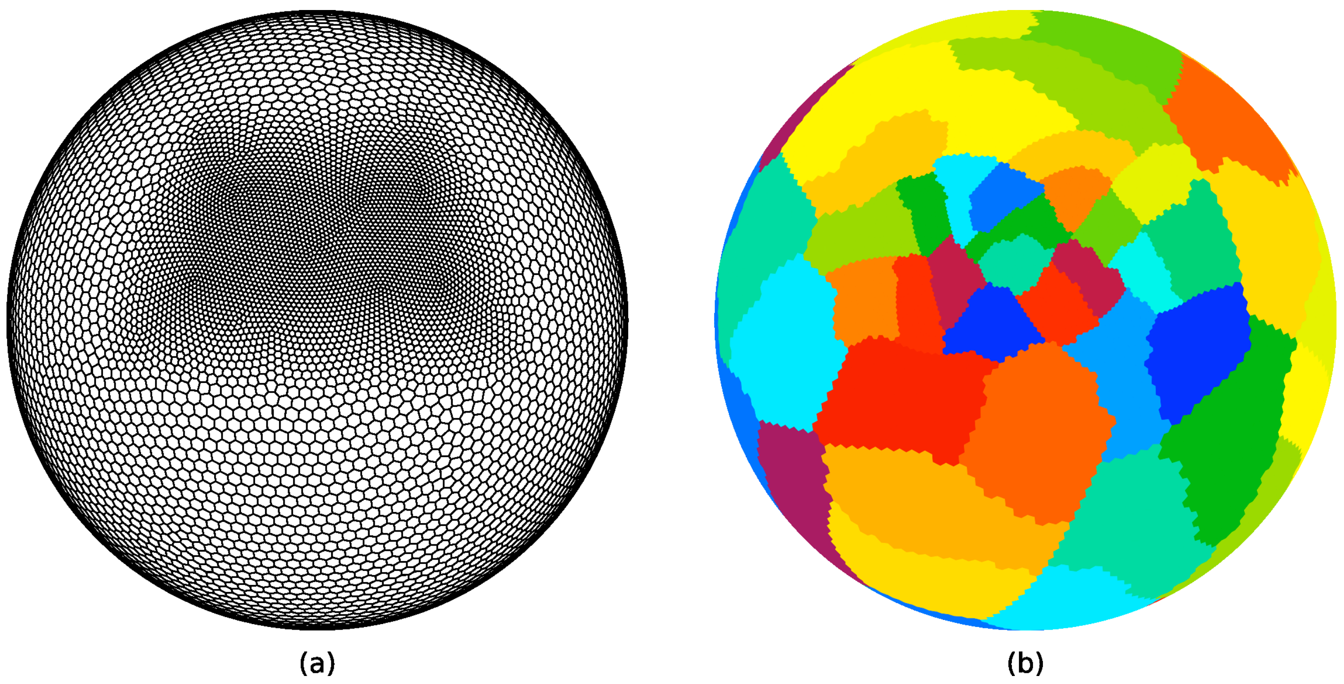

Figure 1: (a) An example of an SCVT with refinement targeted over a region in the Northern Hemisphere. (b) A parallel decomposition of the SCVT into 64 bocks of cells.

3. The MPAS software architecture

At the coarsest level, the MPAS architecture contains three main parts: a driver layer, a model core, and software infrastructure. Figure 2 illustrates the connections between components of the MPAS architecture. As in the figure, the driver layer is divided into two distinct parts. The top-level driver essentially calls init, run, and finalize routines, which are implemented in the sub-driver. In turn, the sub-driver interacts with both the model core and the infrastructure in the course of performing work appropriate to the init, run, and finalize routines. The rationale behind the division of the driver layer into a top-level driver and a sub-driver is heavily influenced by the desire to run MPAS models as components of larger system models. The top-level driver may be removed, and its role fulfilled by a coupler or a component driver in another Earth system model. The routines implemented by the sub-driver may need to be augmented, depending on the requirements of the driver, though the sub-driver should remain independent of any particular MPAS core. With this split between top-level driver and sub-driver, as much driver-level code can be shared between cores as possible, while the amount of code that needs to be replaced by another high-level driver layer is minimized.

TheMPAS core, which lies between the driver and infrastructure, contains all computational work that is specific to a particular model. This work can obviously include that of a dynamical core and physics parameterizations; however, it can also be envisioned as the work of creating initial conditions or of postprocessing simulation output, for example. In this way, most of the MPAS data flow — from the generation of initial conditions, to model simulation, to post-processing — can reuse the MPAS software infrastructure, gaining access to parallelism, I/O, and fundamental data types.

The infrastructure part of the MPAS architecture is roughly divided into four parts: definitions of derived types, input and output, parallelism, and operators. A domain type encapsulates the complete computational state for an MPAS task, including information for distributed-memory parallelism (principally, anMPI communicator), as well as the data to be operated upon by the task. The data for a task is comprised of one or more blocks, with each block constituting the fields defined on the partitions of the mesh assigned to the task plus information about which grid cells of the blocks need to be communicated.

The operators in the MPAS architecture represent, e.g., differential operators for CVT meshes, interpolation routines, advection operators, and other code that can be re-used by different MPAS cores. In order to generate customized infrastructure and other code that would ordinarily require tedious work from the developer of a core, MPAS has adopted a computer-aided software engineering (CASE) tool called the Registry, which is modeled on a tool by the same name in the Weather Research and Forecasting model (Michalakes et al. (2004)). At compile time, the Registry program is first built; then, the Registry parses a text file — called a registry file — specific to each MPAS core, and, based on the contents of the registry file, generates Fortran code for core-specific data types, data allocation and deallocation calls, and I/O calls.

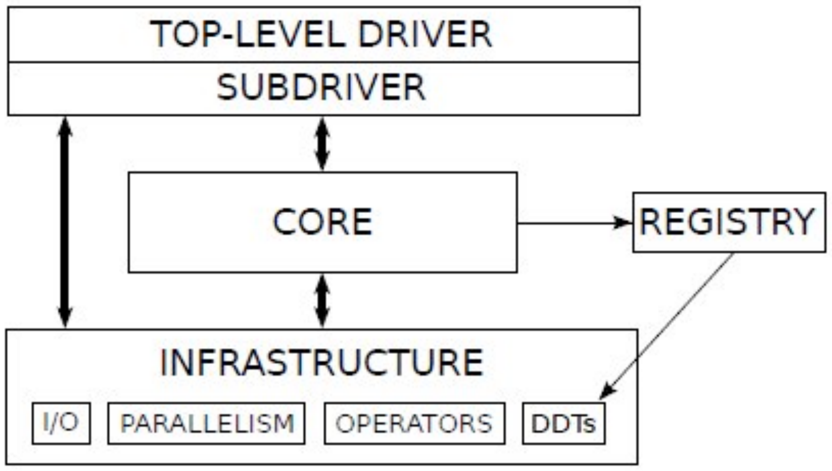

Figure 2: The high-levelMPAS architecture with its three main components: the driver layer, a model core, and model infrastructure; the Registry is a CASE tool used to generate customized DDTs as well as code that would be otherwise tedious to write and maintain.

4. Coupling in MPAS

With the MPAS software in a relatively immature state — the current working framework is still considereda first prototype, in fact — we have attempted to maintain architectural flexibility so that

MPAS models can be coupled using the largest possible range of coupling packages. One method for coupling MPAS models might involve wrapping the MPAS model core and its supporting infrastructure code into a component; coupled fields would be exchanged through import and export states of components, and the control of MPAS execution would be delegated to a higher-level coupler or coupledsystem driver; this approach is facilitated by, e.g., the Earth System Modeling Framework. To support coupling in this manner, we envision replacing the top-level driver in MPAS by an external coupler or driver, and augmenting the implementation of the MPAS sub-driver with routines for importing and exporting coupled fields. The adaption of the MPAS driver layer to this approach is shown in Figure 3a.

Another approach to coupling might involve running MPAS as an independent executable, with new calls to send and receive coupled field placed at appropriate points in the MPAS code. If coupled fields are exchanged at most once per MPAS time step, a flexible implementation of the MPAS I/O subsystem to handle the sending and receiving of coupled fields in the same manner as the input and output of fields may be feasible; this approach is illustrated in Figure 3b. Of course, other paradigms for model coupling also exist, and these will need to be considered as we continue to evaluate the design of the MPAS software.

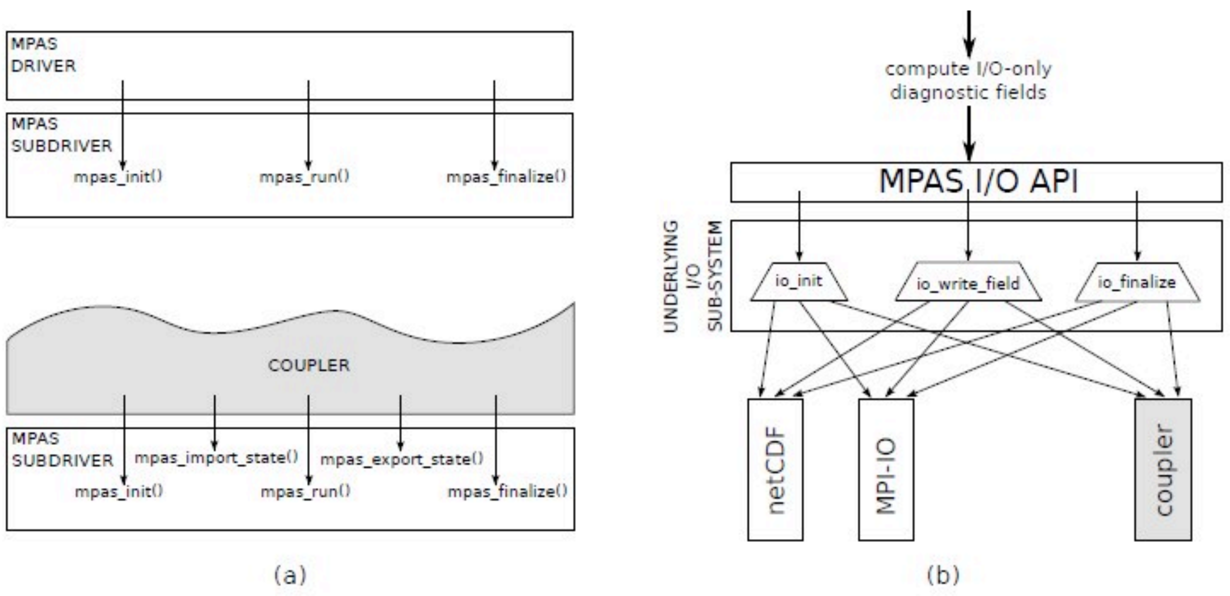

Figure 3: (a) Coupling with MPAS as a component may be accomplished by replacing the top-level driver with a coupler or driver from a larger Earth-system model, and implementing additional routines in the sub-driver. (b) Coupling via sends and receives of fields could be accomplished by implementing these calls as I/O.

5. Conclusions

Given that all MPAS models share the same CVT mesh technology, the development of a common software framework to support modeling on CVT meshes is a logical step. From a coupling perspective, this common framework implies that, if the software challenges of coupling one of the MPAS models can be worked out, then coupling any of the other MPAS models comes at virtually no additional cost, at least from a technical standpoint; we recognize that coupling each model comes with its own scientific issues. The flexibility of CVT meshes poses challenges for theMPAS software infrastructure, and any model coupler used by MPAS must also support horizontally unstructured meshes. To the coupling community, MPAS may present opportunities to test couplers in areas such as re-gridding, since the meshes for MPAS models could be either configured to have coincident cells or completely independent meshes at different resolutions.

References

Ju, L., T. Ringler, andM. Gunzburger, 2010: Voronoi tessellations and their application to climate and global modeling. Chapter to appear in Numerical Techniques for Global Atmospheric Models, Lecture Notes in Computer Science, draft.

Michalakes, J., J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang, 2004: The weather research and forecast model: Software architecture and performance. Proceedings of the 11th ECMWF Workshop on the Use of High Performance Computing in Meteorology, G. Mozdzynski, ed., Reading, U.K.

A Feature Model of Coupling Technologies for Earth System Models

by Rocky Dunlap, Spencer Rugaber, Leo Mark (College of Computing, Georgia Institute of Technology)

Coupling is essential for implementing multi-physics models made up of two or more interacting computer simulations. A quintessential example of such coupled models is an Earth System Model (ESM), which involves several interacting components simulating the Earth's atmosphere, oceans, land, and sea ice systems. The software components that link together and mediate interactions between these models are called couplers. Couplers are well-known abstractions in the geophysical and other scientific communities, although their implementations differ vastly. With respect to ESMs, no standardized reference architecture has emerged. Instead, couplers are designed to address particular modeling situations. The design space of couplers is constrained by properties of the existing models, such as software architecture, dependencies on third party libraries, numerical and scientific characteristics, as well as the nature of the target computational environment.

Because coupling numerical modeling components is a common need, a number of technologies have emerged in the form of reusable software assets to facilitate building coupled scientific applications. Developing couplers is difficult requiring expertise in geoscience, numerical methods, and programming for high performance environments. Therefore, we seek to support geoscience communities building coupled models through generative reuse. Specifically, our approach is based on Generative Programming, a software engineering method for automatically generating members of software families by assembling reusable components into final products based on a declarative requirements specification. Couplers can be seen as members of a family of modules with similar requirements (e.g., they coordinate data communication among models, transform and interpolate field data based on the numerical properties of the constituent models, and manage use of parallel computing resources).

A prerequisite to creating couplers generatively is the need to understand the space (domain) of possible couplers. What features do couplers require? What features are common across couplers and what features vary? How should those features be implemented to address the structure of existing modeling components? A key step in generative programming is feature analysis in which similarities and variations among members of a family of systems are made explicit. Feature analysis determines a multi-dimensional design space for describing a family of applications. The output is a feature model that identifies a concise and descriptive set of common and variable properties of domain concepts. The feature model represents the intention of a software family and can be used to infer the set of possible family instances, called the extension. Once a feature model has been produced, elements can be selected to produce a configuration, describing a desired family member. An automated generator can then be used to produce the actual code for that member.

One way to view a domain is as a set of related software applications. Taking this view, a feature analysis of couplers involves studying existing software systems used for coupling ESMs. The ESM community has already developed reusable software assets in the form of coupling libraries and frameworks, and we have conducted a feature analysis of these existing software assets in support of a generative programming tool we are building. While no two systems are identical, our analysis has revealed significant overlap in the features supported by these coupling technologies. However, there are also significant variations in what features are supported and how the features are implemented. A feature model of couplers makes these similarities and differences explicit.

Our work is similar to the domain analyses done by the Earth System Curator and Metafor projects, but we focus specifically on couplers and coupling technologies for ESMs. Our starting point is existing couplers and coupling technologies, which gives credibility to the analysis and ensures that the results are a true reflection of state-of-the-practice models. Feature analysis allows us to uncover the breadth of features supported by coupling technologies while leaving room to go deeply into one particular feature when desired. Features are abstract, supporting the specification of relevant aspects of coupling technologies, without being tied to specific programming constructs or architectural structures. Features may be functional or non-functional in nature - that is, we can specify not only what kinds of operations are supported, but how they are accomplished (e.g., features related to performance and security). The same feature may manifest itself quite differently across the range of coupling frameworks. Therefore, we can specify that a feature exists without saying too much about how it is implemented.

Coupling Technologies Analyzed

The coupling technologies we analyzed are currently used in Earth System Models or are under active development. Our goal is to paint a relevant picture of the state of the practice for ESM couplers. Table 1 lists the coupling technologies we considered. It is important to note that the studied technologies each have a different scope of use. As such, this is not an apples-to-apples comparison but is intended to reveal the set of features that are relevant when writing couplers for ESMs and, ultimately, for generating them.

Acronym	**Full Name**	**Reference**	**Latest Released Version**
BFG2	Bespoke Framework Generator		bfg2-beta
ESMF	Earth System Modeling Framework		ESMF_4_0_0rp2
FMS	Flexible Modeling System		Riga (internal)
MCT	Model Coupling Toolkit		2.6.0
OASIS/PSMILe	Ocean Atmosphere Sea Ice Soil / PRISM System Model Interface Library		OASIS4
TDT	Typed Data Transfer		12 June 2008

Table 1 - Analyzed Coupling Technologies

The feature analysis we conducted is based on information found in technical documentation that accompanies the coupling technologies (e.g., programming guides, user manuals) as well as articles that describe the technologies and their uses. The initial feature analysis was conducted in a bottom-up fashion by gathering a large list of features that couplers support. The resulting feature diagram contained over one hundred features at the leaf level. We dealt with this complexity by abstracting related sub-features into common higher-level features, sometimes producing a hierarchy several levels deep. To deal with uncertainty in the way certain features are represented, we created an issues list describing alternative feature representations. In working through the issues list, the feature model has undergone several refactorings. During the feature modeling process, we have defined a vocabulary that describes the space of features supported by couplers for ESMs. When alternative terms were found in the literature, we either chose one of the terms or selected a different term which we felt encompassed the semantics of the set of alternatives. In an attempt to appeal to a broad audience of researchers and scientific modelers interested in coupling technologies, we have tried to avoid jargon terms that are only well-known within highly specialized communities.

The full feature model is available in a technical report.

References

[1] Metafor Home Page. Available: http://metaforclimate.eu/

[2] C. W. Armstrong, R. W. Ford, and G. D. Riley, "Coupling integrated Earth System Model components with BFG2," Concurrency and Computation: Practice and Experience, vol. 21, pp. 767-791, 2009.

[3] V. Balaji, "The FMS Manual: A developer's guide to the GFDL Flexible Modeling System," December 17, 2002 2002.

[4] V. Balaji, B. Boville, S. Cheung, N. Collins, T. Craig, C. Cruz, A. d. Silva, C. DeLuca, R. d. Fainchtein, B. Eaton, B. Hallberg, T. Henderson, C. Hill, M. Iredell, R. Jacob, P. Jones, E. Kluzek, B. Kauffman, J. Larson, P. Li, F. Liu, J. Michalakes, S. Murphy, D. Neckels, R. O. Kuinghttons, B. Oehmke, C. Panaccione, J. Rosinski, W. Sawyer, E. Schwab, S. Smithline, W. Spector, D. Stark, M. Suarez, S. Swift, G. Theurich, A. Trayanov, S. Vasquez, J. Wolfe, W. Yang, M. Young, and L. Zaslavsky, "ESMF User Guide Version 3.1," 2009.

[5] K. Czarnecki and U. W. Eisenecker, Generative Programming: Methods, Tools, and Applications: Addison-Wesley, 2000.

[6] R. Dunlap, L. Mark, S. Rugaber, V. Balaji, J. Chastang, L. Cinquini, C. DeLuca, D. Middleton, and S. Murphy, "Earth System Curator: Metadata Infrastructure for Climate Modeling," Earth Science Informatics, vol. 1, pp. 131-149, 2008.

[7] R. Dunlap, S. Rugaber, and L. Mark, "A Feature Model of Coupling Technologies for Earth System Models," Georgia Institute of Technology GT-10-18, October 5 2010.

[8] J. Larson, R. Jacob, and E. Ong, "The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models," International Journal for High Performance Computing Applications, vol. 19, pp. 277-292, 2005.

[9] C. Linstead, "Typed Data Transfer (TDT) User's Guide," Potsdam Institute for Climate Impact Research, Potsdam2004.

[10] R. Redler, S. Valcke, and H. Ritzdorf, "OASIS4--A Coupling Software for Next Generation Earth System Modelling," Geoscientific Model Development, vol. 3, pp. 87-104, 2010.

[11] M. Simos, D. Creps, C. Klinger, L. Levine, and D. Allenmang, "Organization Domain Model (ODM) Guidebook, Version 2.0," 1996.

Data assimilation and coupling

by Andrea Piacentini (CERFACS)

From the point of view of modellers, the aim of Data Assimilation techniques is to incorporate information drawn from observations into a time evolving model, in order to enhance its accuracy. Typical application examples are the determination of the optimal initial condition and of a set of parameters for a forecast model or the integration of large quantities of validated observational data in long term reanalyses.

There is a full collection of data assimilation methods and techniques, but they all share some basic components, such as a model, its linear tangent and adjoint (if available), an observation operator, its linear tangent and adjoint, and so on. This allows for the implementation of modular data assimilation suites, provided that the interfaces between components are coherently defined.

The computational burden due to the assimilation process is usually higher or even much higher than the cost of the direct modelling itself. The best performances are clearly obtained when the data assimilation operators are integrated since the beginning in the model design. On the contrary, implementing a data assimilation procedure inside an existing model can easily turn out to be a cumbersome task and the resulting suite usually lacks of flexibility.

In this situation it is recommended to implement the data assimilation suite as a coupled application where all the components keep a certain amount of independence and the assimilation algorithms is described by their order of execution and by the way they exchange data. With respect to a canonical climatic coupling, the coupled data assimilation suite application has to account for the repeated or conditional execution of the components and therefore for complex communication patterns. We define the former canonical coupling as *static* and the latter kind of coupling as *dynamic*. The PALM dynamic coupler, developed at CERFACS, grants the needed amount of flexibility.

Examples of data assimilation suites implemented as couplings exist in the domains of atmospheric chemistry, oceanography and flood forecasting and the same approach could be easily extended to data assimilation in coupled models.

Webbased Experiments With Earth System Models of Dierent Complexity Used for Education at Freie Universitat

by Ingo Kirchner and Ulrich Cubasch, Institut of Meteorology, Freie Universitat Berlin

1. Introduction

The WEKUW system (Webbased experiments with climate and weather models) gives the students the opportunity to work with typical earth system models without too much technical background. Normally an earth system scientist performs simulations with complex models on super computers. In relation to the modeling, the earth system scientist switches between three different activities (see Fig.1). As analyst he analyzes various model experiments. As modeler he is setting up the model and performing the experiments, as developer he programs and changes the model.

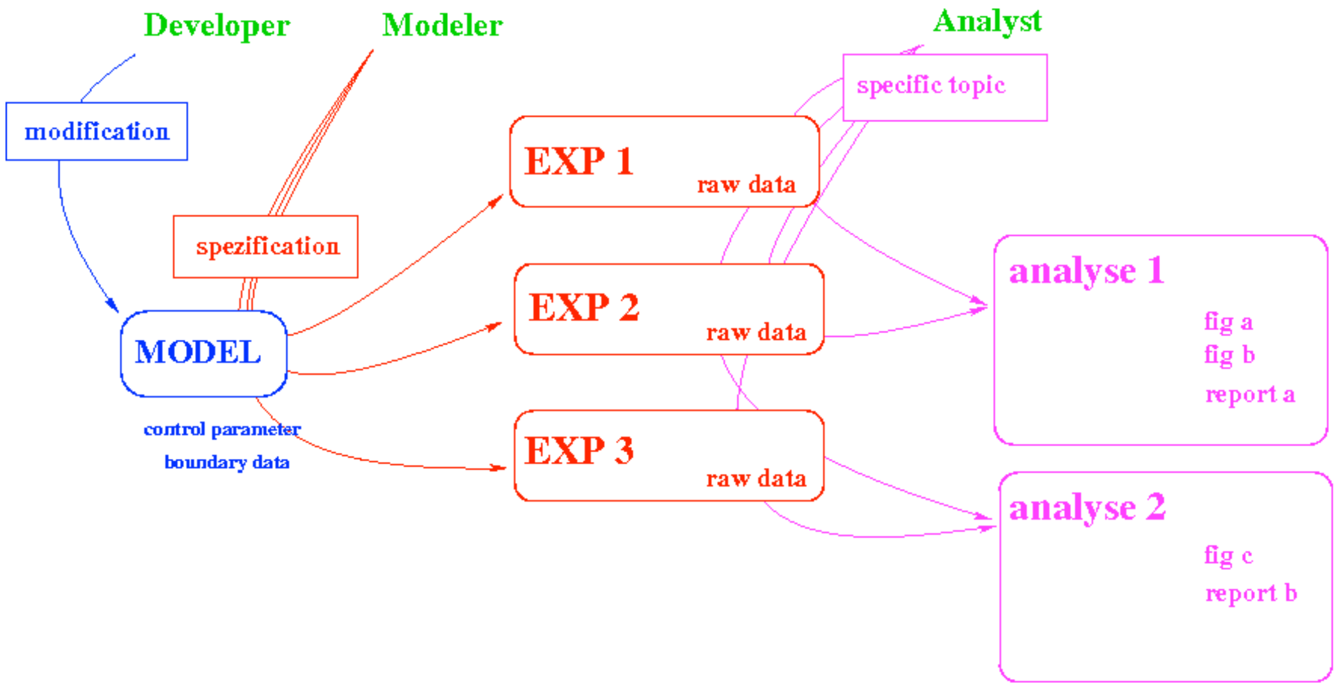

Figure 1: The various tasks of an earth system scientist to perform an experiment

For these tasks he needs in addition to the earth science background good skills in computational sciences and technical programming. These skills are missing by most of the meteorology students. The WEKUW system hides most of the technical tasks from the user. The system builds an application layer between the models and the user interface.

2. Climate Modeling with WEKUW

The concept of the WEKUW system has the main focus on typical scientific questions. The user performs the simulations and learns the basic knowledge for the application of the models by doing the experiments. The minimum technical requirements are an internet connection and a web browser on the user side. The server connects to all different models in an unique way and do the simulation uncoupled from the user login.

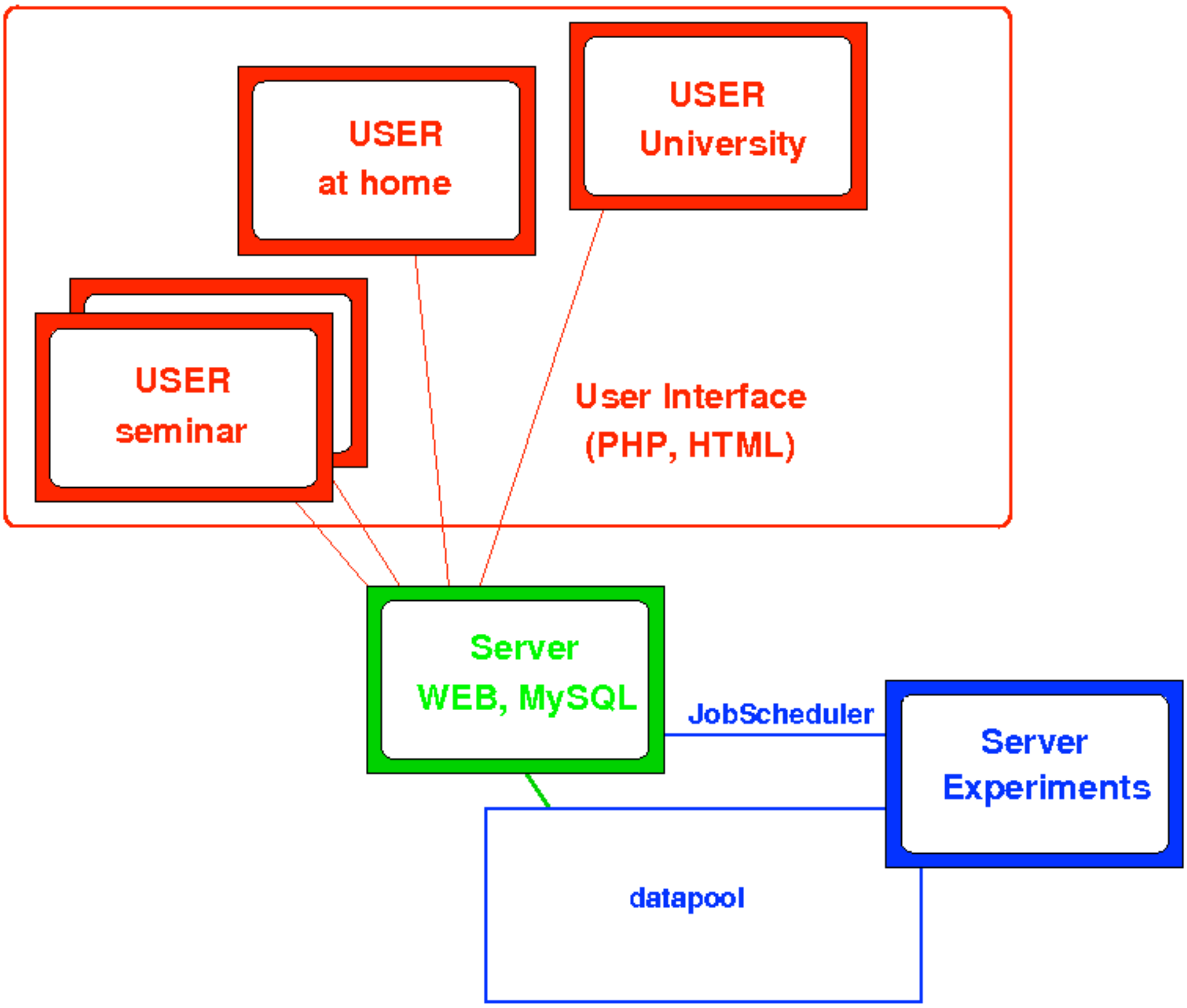

Figure 2: Components of the WEKUW system

The WEKUW system can be divided into the following components (see Fig.2):

- The WEKUW server provides the basic functions for the uniform operation with the models. This includes the online help, starting the experiments, the data evaluation and the management of the user accounts.
- The WEKUW models are a compilation of the individual models for different applications. For every model exists a catalog of scientific questions, a package for the local installation of the models and a chapter in the content area of the server.
- The WEKUW training course follows the personal learning path, or is embedded in the curriculum. Normally a selection is made by the tutor based on the pool of the models with their question catalogs. A training course for example is managed on the platform of the FUB (http://wekuw.met.fuberlin.de/WEKUW/current/). Another typical application scenario of the WEKUW system is the goal-oriented involvement of experiment selection and is used for demonstration of practical activities.

The user side of the server supports two roles, the tutor and the student. The tutor can control the educational process of each student individually. He will switch the models on or off, unlock the experiments for the student and he can comment the experiment results of each student in the web-environment. The student has access to the documentation of each model and each experiment. He can start the experiments which are unlocked for him. For each experiment type the user can store notes inside the web-environment and the tutor can comment these individually via the web interface.

3. Example Workow with WEKUW

The web interface allows to modify the model control parameters, to start the models, to control the experiments and to analyse the results of each simulation (see Fig.3).

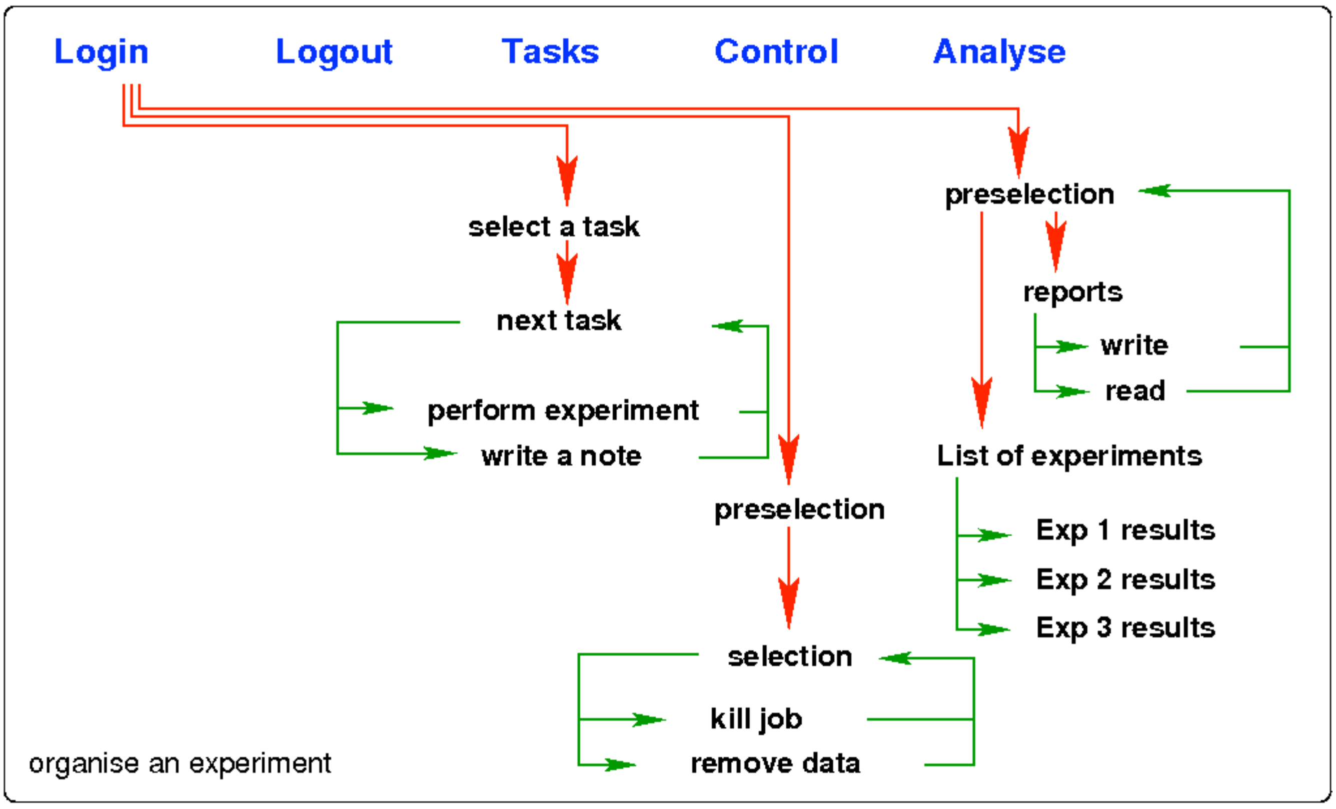

Figure 3: The basic workow in the user mode of the WEKUW system

For each model a series of different experiments is available. This list is continuously expanded and includes currently:

- EBM an energy balance model, see e.g. McGue/Henderson-Sellers (1997)
- PUMA the Portable University Model of the Atmosphere of the University of Hamburg, see Fraedrich et al. (2001)
- ECHAM4 the 4th generation of the atmospheric global circulation model of MPIM Hamburg, see Roeckner et al. (1996)
- PLASIM the planet simulator of the University of Hamburg, see Lunkeit et al. (2004 and 2005)
- RCG the regional chemical transport mode of the TrUmF group (tropospheric environmental sciences) of the Institute of Meteorology at FUB, see Stern (2003 and 2004)

Here a short session scenario will be described to demonstrate a typical experiment workflow. After individual registration, the tutor will unlock a specific scientific question, e.g. task D/03 in the ECHAM4 section “What is the effect of the parameterization of radiation, friction, and clouds?". Now the student will login with his own account and will go to the selected task. On the introduction page of the experiment the student will see additional references and a short explanation of the scientific background. Here he can fill in a note, e.g. writing down the individual experiment plan. By using the “execute experiment" button, the user opens the conguration page. In our example the user will make two experiments, for the first one only the experiment name will be changed to “control" and the experiment will be submitted. For the second experiment the name is changed to “noclouds", the cloud parameterization is switched off, the experiment is submitted and the student can logout. The simulations will be executed in the background. After few hours the student will analyse the experiments. He will login again, chooses the “analysis" button from the main WEKUW menu and opens the experiment result pages of both experiments. Now he can compare the preprocessed figures, download and analyses the raw data or can read more details about the runscript and the model, e.g. browsing in the source code of the model.

4. Summary and Outlook

The WEKUW system is used since 2003 for the training of meteorology students. In the training course the students start with an energy balance model (EBM), continue with experiments of a simple general circulation model (PUMA), with an aerosol chemistry transport model (RCG), with a complex global atmosphere model (ECHAM4) or with a simple earth system model (PlaSim). There is no limitation for an extention of the system with new models, as each new model can be implemented in a modular way.

5. References

Fraedrich, K., D.Kirk und F.Lunkeit (2001): Portable university model of the atmosphere. Veröffentlichungen des Meteorologischen Instituts der Universität Hamburg, 37 pages.

Lunkeit, F., et al. (2004): Planet simulator - Reference manual. 102 pages. Lunkeit, F., et al. (2005): Planet simulator - User's guide. 82 pages.

McGue, K., and A. Henderson-Sellers (1997): A climate modelling primer,John Wiley & Sons, Serie: Research and Development in Climate and Climatology, 253 pages.

Roeckner, E., et al. (1996): The atmospheric general circulation model ECHAM-4: Model description and simulation of present-day climate. Max-Planck Report No.218.

Stern, R. (2003): Entwicklung und Anwendung des chemischen Transportmodells REM-CALGRID. Abschlussbericht zum FuE-Vorhaben 298 41 252 des Umweltbundesamts Modellierung und Prüfung von Strategien zur Verminderung der Belastung durch Ozon.

Stern, R. (2004): Weitere Entwicklung und Anwendung des chemischen Transportmodells REM-CALGRID für die bundeseinheitliche Umsetzung der EU-Rahmenrichtlinie Luftqualität und ihrer Tochterrichtlinien. Abschlussbericht zum FuE-Vorhaben 201 43 250 des Umweltbundesamts Änwendung modellgestützter Beurteilungssyteme für die bundeseinheitliche Umsetzung der EU-Rahmenrichtlinie Luftqualität und ihrer Tochterrichtlinien.

WEKUW (2002): http://wekuw.met.fu-berlin.de/WEKUW/current the webpage of the WEKUW installation at the Freie Universität Berlin.

Metadata and coupling

by Rupert Ford (Manchester University)

This talk concentrates on two potential uses of Metadata for Coupling. The first is to configure a set of model components into a particular coupled model run. The second is to capture the provenance of a particular coupled model run.

For coupled model configuration we compare and contrast three coupling systems, the two most widely used ESM coupling systems (ESMF and OASIS) and BFG, which is fully metadata driven.

For coupled model provenance we introduce and propose the Common Information Model (CIM) that is being developed by the METAFOR project.

Finally we present a vision of using the CIM (or some future derivative) as the "lingua franca" for both provenance and configuration Metadata.

Leveraging the New CESM1 CPL7 Architecture - Current and Future Challenges

by Mariana Vertenstein (NCAR)

New CESM science will build on the scalability, flexibility and extensibility of the new CPL7 architecture. Key examples of this will be outlined in this talk. CESM1 is now targeting unprecedented global resolutions for all components. The CESM1 coupling infrastructure has been extended in order to provide data assimilation capability in order to obtain better ocean initial conditions for the upcoming decadal prediction runs that are part of the CMIP5 experimental protocol suite. A new land ice sheet component has been added that will provide the capability of better predictions of sea-level rise. The CPL7 paradigm has also been leveraged to create a new "ocean" component that permits the nesting of a regional model, ROMS, in the global ocean component, POP. The flexible inclusion of a new isotope capability will help provide new insights in the global hydrological cycle and of ocean model circulation. Finally, the incorporation of super-parameterization in CAM, and the accompanying changes to pass new surface fields to CAM, will result in better simulations of clouds, one of the largest sources of uncertainly in climate models. These topics are summarized in more detail below.

Ultra High Resolution: Scalability out to tens of thousands of processors has already been demonstrated with the addition of the HOMME dynamical core in CCSM4 (CAM4). An outstanding goal is to approach a global horizontal scale of 10km across CESM components. Critical to the ability to achieve this scaling has been the introduction of memory and performance scalability that could not have been obtained without the use of a new and efficient parallel I/O library, PIO, that was designed and implemented by CESM collaborators at NCAR, ANL, ORNL and LLNL. Previous to PIO, external storage accesses was limited to a single master process, thereby creating a serial bottleneck, degrading parallel performance scalability of the application as a whole, and exhausting local memory at ultra high model resolutions. A parallel solution was therefore needed that had more generality than having all processes access the external storage, especially to access to the same file. The latter case can lead to failure or very poor performance when thousands or hundreds of thousands of processes are involved.

PIO was initially designed to allow better memory management for very high-resolution simulations by relaxing the requirement for retaining the memory corresponding to the global 2-d horizontal resolution on the master I/O task. Since then, PIO has developed into a general purpose parallel I/O library that serves as a software interface layer designed to encapsulate the complexities of parallel I/O and to make it easier to replace the lower level software backend. PIO has been implemented throughout the entire CESM system and currently supports serial I/O using netCDF and parallel I/O using pnetCDF. PIO calls are collective. An MPI communicator is set in a call to the PIO initialization routine and all tasks associated with that communicator must participate in all subsequent calls to PIO.

One of the key features of PIO is that it takes the model's decomposition and redistributes it via a generic framework-independent rearranger to an I/O "friendly" decomposition on the requested number of I/O tasks. It is important to note that there is no imprinting of the model decomposition in the resulting I/O file. In using the PIO library, the user must specify the number of I/O tasks to be used, the stride or number of tasks between I/O tasks and the backend type (e.g. pnetCDF). By increasing the number of I/O tasks, the user can easily reduce the serial I/O memory bottleneck even with the use of serial netCDF.

Extending the coupler infrastructure - data assimilation: Short-term decadal prediction runs are part of the CMIP5 protocol suite and will be much more sensitive to ocean initial conditions. New ocean data assimilation has been incorporated into the model system by extending the coupling architecture in order to permit the instantiation of more than one instance of a model component within the single model executable. This new capability is utilized to perform ocean data assimilation using Kalman Ensemble filtering via the Data Assimilation Research Testbed (DART). To perform ocean data assimilation, DART is combined with the Community Atmosphere Model (CAM) and the Parallel Ocean Program (POP) to create loosely coupled ensemble analyses of the ocean that are consistent with the analyses of the atmosphere. Ocean data assimilation has been carried out using 48 members of POP initially drawn from model climatology and 48 members of a data atmosphere that comes from an independent CAM ensemble member analysis (also using DART) where observed ocean temperature and salinity is assimilated every midnight. The large ensemble and diverse atmospheric forcing lead to improvements in the ensemble mean ocean analysis.

Introduction of new model component – land ice sheet: The ice sheets of Greenland and Antarctica are strongly coupled to the ocean, land and atmosphere, and are expected to play a pivotal role in the global sea-level rise in the 21st century. Consequently improved sea-level predictions are needed for mitigation and adaptation strategies. A new dynamic ice sheet component has been added to the CESM1 system that will enhance the ability to predict changes in ice sheets and sea level rise on the decade to century time scales. CESM 1.0 now includes Glimmer-CISM (v1.6), the Community Ice Sheet Model and is the first publicly released IPCC-class model to include a dynamic ice sheet model. The current ice-sheet component, is serial and does not include the higher-order dynamics required for modeling fast-flowing ice streams and outlet glaciers. However, it does include a new surface-mass-balance scheme in the land component that passes numerous new fields through the coupler.

Currently, only one-way coupling between the land surface and ice sheets is utilized. The surface mass balance of ice sheets is computed in CLM and downscaled to Glimmer-CISM within the ice-sheet model. The ice sheet then evolves in time, but the topography and surface types in CLM are held fixed. Furthermore, ice-sheet/ocean coupling is also not supported and is needed to simulate interactions between oceans and floating ice shelves. A near term goal is the addition of a parallel, fully coupled ice sheet model with advanced dynamics that will send data such as ice area and elevation to other CESM components and receive boundary conditions such as ice accumulation and melting rates from land and ocean models.

Introduction of embedded regional models – NRCM: CESM1, along with many current global climate models shows significant biases in properties such as sea surface temperature in upwelling regions. Increasing the atmospheric resolution helps, but is not sufficient. One approach that is being taken in CESM1 to mitigate these biases is to embed a high-resolution limited domain model, the Regional Ocean Model System (ROMS), within POP, in each of the upwelling regions. The hope is that embedding regional 1/10° ROMS models in a global system containing a 1° ocean (POP) will permit the resolution of regional scale processes as well as their influence on the large-scale climate, thus leading to improved simulations in these regions.

The approach taken for NRCM is to create a new “hybrid” OCN component that in effect serves as a driver and coupler for a POP-ROMS system. The OCN component couples to the CPL7 driver as if it was a standard CESM1 component. However, the sea-surface temperatures (SSTs) passed back to the driver correspond to merged POP/ROMS SSTs. The

POP model still receives atmosphere/ocean fluxes that are computed in the coupler code. However, ROMS computes its own atmosphere/ocean fluxes thereby requiring a new time series of atmosphere forcing fields on the atmosphere grid to be passed to the OCN component.

Introduction of flexibly specified new fields – addition of isotopes: CESM is targeting the addition of isotope tracers in order to better understand the flow of water, carbon and nitrogen in the model system. From a scientific perspective, water isotopes are used to provide new insights into the global hydrological cycle and cloud processes. The addition of these isotopes to CESM would also enable for the first time a direct comparison of model output to paleoclimate archives. Similarly, the addition of carbon isotopes to CESM will lead to new understanding of ocean model circulation and deep-water mass formation processes. The goal is to extend the current coupling scheme to allow for the run time specification of isotopes and have the coupler automatically pass the tracers between components.

Introduction of new physics – addition of super-parameterization in CAM: The modeling of clouds is one of the major sources of uncertainty in global climate models. One approach to removing this uncertainty is to migrate to global cloud resolving models. However, this currently results in a prohibitive computational cost. An intermediate approach, that is believed to provide more accurate simulations of cloud fields, is to replace only the parameterized moist physics in each model grid column with a "small" cloud resolving model. This approach is referred to as super-parameterization. From a coupling perspective, the introduction of super-parameterization in CESM/CAM will require new flux information from the surface components. In particular, in addition to state and flux fields, the introduction of super-parameterization will also require the sending of higher-order moments of these fields that describe spatial variability information for these fields.

Finally, new CESM science will also require addressing new computational challenges such as the incorporation of new parallel workflow and post-processing functionality and determining ways to leverage GPU functionality to benefit model performance.

Coupled models at the Max-Planck-Institute for Meteorology

by René Redler (MPI Meteorology)

We start with a description of the current production version of the Earth System Model (ESM) developed and used by the Max Planck Institute for Meteorology (MPI-M) , now named MPI-ESM Version 1, with special emphasis on the coupling of the atmosphere and ocean model via the OASIS coupler. The short technical description of MPI-ESM1 serves as the basis for an introduction to MPI-ESM2, the new coupled model under development at MPI-M.

The MPI Earth System Model Version 1

MPI-ESM1 is designed to simulate the full Earth System over periods of hundreds to a few thousand years. The Earth system model is the major tool for experiments addressing questions for example on the internal variability of the system or its susceptibility to natural or anthropogenic perturbations, including major volcanic eruptions, variations in the solar irradiation or anthropogenic greenhouse gas emissions. The MPI-ESM1 consists of four main components: the atmospheric ECHAM6 model, the land model JSBACH, the ocean model MPIOM and the ocean-biogeochemistry-model HAMOCC. A sea-ice model, the fifth major component in our system, is included in MPIOM. The MPIOM and HAMOCC on the one hand and the ECHAM6 and JSBACH models on the other hand are coupled directly, while the air-sea-exchange in the coupled MPI-ESM1 is taking place via OASIS3. Technically, this is solved by a coupling between MPIOM and ECHAM6. The atmosphere part collects contributions from land (e.g. river runoff) while the ocean collects data from the ice and the ocean-biogeochemistry model. While for the internal coupling data are exchanged at every time step, for the external coupling the interval for data exchange between the ocean and the atmosphere is set to one day. Coupling fields are accumulated at each time step within the sending model components locally on each process. Data are averaged before sending them to the OASIS3 coupler.

The boundary values that are exchanged between the atmosphere and the ocean are :

<u>Ocean/Sea-Ice/Ocean-Biogeochemistry to Atmosphere/Land</u>

- sea surface temperature
- sea ice concentration
- sea ice thickness
- ocean horizontal surface velocity
- CO2 flux

<u>Atmosphere/Land to Ocean/Sea-Ice/Ocean-Biogeochemistry</u>

- solar and non-solar heat fluxes
- precipitation, evaporation, river runoff
- snow fall
- horizontal wind stress
- CO2 concentration

As an alternative to the fully coupled model configuration the software environment allows to run subsets of the model system (see Table 2).

ECHAM6 – JSBACH – **MPIOM** – HAMOCC	coupled with **OASIS3**
ECHAM6 – JSBACH – **MPIOM**	coupled with **OASIS3**
ECHAM6 – JSBACH	
MPIOM – HAMOCC	
MPIOM	
JSBACH	

Table 2: Possible model configurations of MPI-ESM version 1

In terms of horizontal and vertical resolution, the coupled MPI-ESM1 and it subcomponents can be assembled in several different model configurations:

Horizontal and vertical resolution		# of horizontal grid points	
Atmosphere	Ocean	Atmosphere	Ocean
T31L19	GR30L40	96 x 48	121 x 101
T63L47	GR15L40	192 x 96	256 x 220
T127L95	TP04L40	384 x 192	802 x 404
T255L199	TPM6L80	768 x 384	3586 x 1800

Table 3: Possible coupled model configurations of MPI-ESM1

In production mode the T127L95 ECHAM6/JSBACH is typically run with up to 32 x 24 MPI processes and two OpenMP threads. MPIOM/HAMOCC (TP04L40) typically uses 16 x 8 MPI processes. For our setup we use the parallel version of OASIS3 as it is provided by CERFACS with up to 21 OASIS3 processes, one for each coupling field. Even though we collect the data on the component root processes prior to the exchange with OASIS3 and redistribute them after having them received, the cost for coupling is reasonably low (less than 2% of the total time) even at this quite high resolution. Efficient gather and scatter routines as well as the local accumulation of coupling fields all done inside our model components contribute to these low costs.

Currently we investigate to replace OASIS3 with OASIS4, especially for targeting the high-resolution configuration T255L199-TPM6L80. While it is quite straightforward to replace the OASIS3 interface with an OASIS4 interface in both our model components, we notice problems with the coupling of the MPIOM grid in the Arctic region. While the neighbourhood search in OASIS4 is quite efficient, it fails to generate proper interpolation stencils for target cells that intersect with the edge of the northern compute domain because no information is provided about the connectivity of points. With OASIS3 this problem is still overcome by using the conservative remapping where the less efficient search provides correct results. While OASIS3 would only fail to provide correct results for a bilinear or bicubic interpolation, with OASIS4 even the conservative remapping would fail under certain conditions. This problem can either be solved by providing appropriate information about the connectivity rather than relying on the implicit data structure, or by extending the OASIS4 functionality in such a way that it identifies missing cells and points in these critical regions by evaluating the geographical information provided by the user API.

The MPI Earth System Model Version 2

ICOsahedral **N**on-hydrostatic General Circulation Models (ICON) is a joint project of MPI-M and the German Weather Forecast Service (DWD), with the goal to develop a new generation of general circulation models for the atmosphere and the ocean in a unified framework. These models use unstructured grids derived from an icosahedral base grid. Parametrizations are inherited from MPI-ESM1 and so will be the component models HAMOCC and JSBACH; the coupling fields exchanged between ocean and atmosphere are similar to MPI-ESM1 as well.

The first version of the coupled model will employ identical horizontal grids in atmosphere and ocean. While for each wet ocean point there is a corresponding grid point at the same geographical position in the atmosphere, grid points are partitioned in a different way in both media. A search is required to locate the process on which the remote grid point is located; in this simple case we do not require any interpolation. For this first simple coupling we have developed a customized light-weighted coupler which exploits the known grid hierarchy and geometry, and thus allows for an efficient search (and interpolation) on ICON grids. In our preliminary configuration grid points on identical geographical positions will have identical grid point indices which allows us to reduce the search to a 1d search along one integer array.

The current plan is to extend this coupling software with growing needs of an advanced version of the MPI-ESM2 and use this as a simple test-bed to implement the functionality required like particular interpolation schemes or modifications of the user API. As the icosahedral grid has a hierarchical structure we plan to use this hierarchy for an efficient search strategy similar to the multi-level approach taken in OASIS4 for block-structured grids. In a third step we plan to integrate our software development into OASIS4 and provide an OASIS4 interface for the individual components to allow for a coupling of any of the individual MPI-ESM2 components to external models working on block-structured grids like MPIOM, ECHAM and others.

Infrastructure requirements in support of Met Office models

by Steve Mullerworth (MetOffice)

The Met Office is developing its next generation of global coupled models, earth system models and forecast production models around the use of the OASIS coupler. In common with other models, higher resolution versions of these models urgently need more scalability in both the model formulation and the coupling framework.

The coupled configuration currently under development comprises the Met Office Unified Model (UM) atmosphere, the NEMO ocean model and the CICE sea ice model and is called HadGEM3-AO. While the development of the climate configuration (N96, which is roughly 1.5 degree atmosphere and 1 degree ocean) is still continuing, HadGEM3-AO is already being used to provide seasonal forecasts in production (GLOSEA4), with plans to migrate to a higher resolution version of this model (N216 and 0.25 degree ocean).The OASIS3 coupler is currently used for all resolutions of the model. The N216 resolution version uses 8 instances of OASIS3 to manage the coupling of approximately 20 fields each-way. Plans are underway to migrate HadGEM3 to the parallel OASIS4.

In the future, the HadGEM3 configuration is likely to form the basis of a range of models, from low resolution fast Earth System Models,through providing the mid-resolution scenario runs for the IPCC process, to providing a myriad of climate and forecasting services in a tight operational schedule.

The UM atmosphere currently benefits from a highly flexible and configurable post-processing system that allows users to select individual diagnostics, to select in-line time and domain processing individually for each chosen diagnostic, and to allow output of results to a choice of files. Asynchronous IO is currently being implemented to manage the output of these multiple files.

We foresee addition of multiple other components into the current model system, such as wave models, chemistry models and new land surface schemes, each of which have differing demands on the coupler and model infrastructure. Flexibility in choices of how to couple and deploy components developed both at the Met Office and by our collaborators and external groups is likely to be an important requirement.

Addressing the Challenge of Exaflopic Computation

by Jean-Yves Berthou, EDF R&D - European Exascale Software Initiative Coordinator, and Jean-Claude André, Vice-Chair of the EESI Working Group on industrial and engineering applications

Abstract

Exaflopic systems, composed of millions of heterogeneous cores will appear at the end of this decade. This technological breakthrough will engage the HPC community in defining new generations of applications and simulation platforms. The challenge is particularly severe for multi-physics, multi-scale simulation platforms that will have to combine massively parallel software components developed independently from each others. Another difficult issue is to deal with legacy codes, which are constantly evolving and have to stay in the forefront of their disciplines. This will also require new compilers, libraries, middleware, programming environments (including debuggers and performance optimizers), languages, as well as numerical methods, code architectures, and pre- and post-processing tools (e.g., for mesh generation or visualization).

The goal of the European Exascale Software Initiative project (EESI) is to build a European roadmap along with a set of recommendations to address the challenge of performing scientific computing on this new generation of computers. This paper presents the objective and first results of EESI as well as the international context on which this effort is conducted.

Introduction

Exaflopic computer (10^18 floating point operations per second) composed of millions of heterogeneous cores are expected at the end of this decade. These incredible capabilities lead to outstanding technological breakthrough possibilities opening unknown areas in designing new products or optimizing existing ones in almost all society domains [5,6,7,8,11,12,15,18].

These massively parallel systems will engage the HPC community for the next 20 years in defining new generations of applications and simulation platforms. The challenge is particularly severe for multi-physics, multi-scale simulation platforms that will have to combine massively parallel software components developed independently from each others. Another difficult issue is to deal with legacy codes, which are constantly evolving and have to stay in the forefront of their disciplines. This will require new numerical methods, code architectures, mesh generation tool, visualization tool. In addition to the applications, all the software layers between the applications and the hardware need to be revisited. Many challenges are to be addressed: scalability, fault tolerance, programming models, to cite a few. As examples of scalability challenges, currently, none of the runtime environment allows executing an application on one million of nodes and there is no known solution to launch one million of processes on large scale machines in less than 5 minutes. Fault tolerance is another very important challenge to solve before being able to run applications on one million of nodes for hours. Several recent keynote addresses in top level conferences [11][12] have raised two significant issues: 1) the MTBF of very large computers is diminishing rapidly and will reach soon the time required for fault tolerance systems to only restart the application, 2) the exponential increase of the number of transistors and their exponential reduction in size with time, will increase significantly the number of "masked errors" that could be not detected by any system (silent soft errors). A third challenge is on the programming approaches for Peta-Exascale computer. Programming environments should deal with hierarchy, heterogeneity, flexibility and help the programmer for making his program scalable.

Performance, computational precisions, energy saving, etc. are also challenges to be addressed [3,4,13,14,16,20,21,22].

The International Exascale Software Project (IESP)

Community of researchers in HPC software is convinced that there is no way for a single continent alone (America, Europe or Asia) to design and develop all the software needed for these computers. In USA, the Department Of Energy (Office of Science) has launched at the end of 2008 the International Exascale Software Project (IESP)[10,24] and has invited the international community, mainly US, Europe and Japan, to work together for providing the necessary tools, software and methods for new generation of HPC applications. This initiative, led by Jack Dongarra and Pete Beckman, claims that "…although the investments in these separate software elements have been tremendously valuable, a great deal of productivity has also been lost because of the lack of planning, coordination, and key integration of technologies necessary to make them work together smoothly and efficiently, both within individual PetaScale systems and between different systems…. It seems clear that this completely uncoordinated development model will not provide the software needed to support the unprecedented parallelism required for peta/Exascale computation on millions of cores, or the flexibility required to exploit new hardware models and features, such as transactional memory, speculative execution, and GPUs. We believe the community must work together to prepare for the challenges of Exascale computing, ultimately combing their efforts in a coordinated International Exascale Software Project".

EESI, building a European vision and roadmap

Even if the European participation to IESP is significant, many experts and stakeholders in Europe are not involved in this roadmapping activity. Moreover, we are convinced that an Exaflopic roadmap should be conducted also at the European level, including the technological dimension as well as the applicative one. This lead us to propose to the European Commission to fund the European Exascale Software Initiative, EESI [25]. EESI was launched the June 1, 2010 for a 18 months duration.

The EESI goal is to build a European vision and roadmap to address the challenge of performing scientific computing on multi Petaflop performances in the next few years and Exaflop performances in 2020. EESI is investigating where Europe stands in the overall international HPC landscape, what are its strengths and weaknesses, what are the priority actions, and what cooperation modes should be implemented between Europe and the international community. EESI is also identifying the sources of competitiveness for Europe induced by the use of Peta/Exascale software. It is investigating and will propose programs in education and training for the next generation of computational scientists.

A first mapping of the major HPC projects and organizations has been achieved. This mapping has been extended world-wide using IESP inputs and international contacts. It is available on the EESI web site [1].

The EESI work plan is progressing in two directions. A first set of four working groups is targeting the technological computing domain challenges: hardware and associated software, computer science, numerical analysis and applicative software (ie. scientific and engineering codes). Each working group will produce its own roadmap by June 2011. A second set of working groups will target the applicative side by looking for major grand challenges in Climate and Weather forecasting, Industrial application (focus on Transportation and Energy), Physics and Engineering sciences and Life science-Health-BPM. Each of these four working groups will also produce its own roadmap integrating technological inputs identified by the first four working groups. The economic dimension and impact on European competitiveness of these challenges will be particularly under study. To ensure close collaboration and sharing, one internal workshop will be held in February 2011 where each working group will be invited to present its first results and roadmaps.

An overall synthesis will be produced and be presented at a large final public conference in Barcelona in October 2011.

Bibliography

[1] EESI Web site, www.eesi-project.eu/media/download_gallery/EESI_Investigation_on_Existing_HPC__Initiatives_EPSRC_D2%201_FF.pdf

[3] "Simulation-Based Engineering Science - Revolutionizing Engineering Science through Simulation" , mai 2006, www.nsf.gov/pubs/reports/sbes_final_report.pdf

[4] ORNL/TM-2007/44, National Centre for Computational Sciences, Oak Ridge National Laboratory, « COMPUTATIONAL SCIENCE REQUIREMENTS FOR LEADERSHIP COMPUTING »,July 2007, Douglas Kothe-Ricky Kendall

[5] ORNL/TM-2007/238, National Centre for Computational Sciences, Oak Ridge National Laboratory, « Scientific Application Requirements for Leadership Computing at the Exascale », December 2007, Computing Requirements Team

[6] ORNL/TM-2007/232, National Centre for Computational Sciences, Oak Ridge National Laboratory, « Science Prospects and Benefits with Exascale Computing » December 2007, Douglas B. Kothe

[7] "Enquête sur les frontières de la simulation numérique », mai 2005, www.academie-technologies.fr/V2/ecrit05/Simulation/rapport090505_2.pdf

[8] "Getting ready for petaflop capacities and beyond: a utility perspective", 2008 J. Phys.: Conf. Ser. 125 012001, July 2008, Jean-François Hamelin and Jean-Yves Berthou

[10] http://www.exascale.org/iesp/Main_Page

[11] "Reflections on Failure in Post-Terascale Parallel Computing", ICPP 2007, Keynote, Garth Gibson

[12] "Fault Tolerance for PetaScale Systems: Current Knowledge, Challenges and Opportunities", Europar 2008, Keynote, Franck Cappello

[13] ExaScale Computing Software Study: Software Challenges in Extreme Scale Systems, a DARPA/IPTO Report (September 14, 2009), http://users.ece.gatech.edu/%7Emrichard/ExascaleComputingStudyReports/ECSS%20report%20101909.pdf

[14] International Assessment ofesearch in Simulation-Based Engineering and Science, a World Technology Evaluation Centre (WTEC) panel report, sponsored by the NSF and other US Govt. Agencies, chapter 5 (*Next Generation Architectures and Algorithms* by George Em Karniadakis) and chapter 6 (*Software Development* by Martin Head-Gordon), http://www.exascale.org/mediawiki/images/2/20/SBES-InitialFullDraftReport-30April2009_BW.pdf

[15] Modeling and Simulation at the Exascale for Energy and the Environment: Report on the Advanced Scientific Computing Research Town Hall Meetings on Simulation and Modeling at the Exascale for Energy, Ecological Sustainability and Global Security (E3) (2008), *Horst Simon(LBNL), Thomas Zacharia (ORNL), Rick Stevens (ANL). Sponsored by the DOE Office of Science, http://www.sc.doe.gov/ascr/ProgramDocuments/Docs/TownHall.pdf*

[16] ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, (DARPA/IPTO, 2008), http://users.ece.gatech.edu/%7Emrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf

[18] Parallel Computing Research at Illinois The vision and research agenda of the Universal Parallel Computing Research Centre at the University of Illinois at Urbana-Champaign, http://www.upcrc.illinois.edu/whitepaper.php

[20] ICT Infrastructures for eScience, Communication from the Commission to the European Parliament, The Council, the European Economic and Social Committee, and the Committee of the Regions, http://www.exascale.org/mediawiki/images/4/4c/ICT-Infrastructures-eScience.pdf

[21] High Performance Computing & Numerical Analysis Applications/Algorithms Roadmap Version 1.0, A. Trefethen, N. Higham, I. Duff, P.V. Coveney - Engineering and Physical

[22] Fundamentals of Technology Roadmapping, Marie L. Garcia and Olin H. Bray, Sandia National Laboratory

[24] The International Journal of High Performance Computing Applications Volume 23 Issue 4, October 2009 Special Issue dedicated to The International Exascale Software Project

[25] www.eesi-project.eu

Designing HPC Software for an Uncertain World of Hardware

by Wael Elwasif and David Bernholdt (Oak Ridge National Laboratory)

Historically, the one constant in HPC hardware architectures is change, but the pace of change is, arguably, accelerating of late. During the transition from terascale to petascale, we've rather suddenly seen power concerns come to dominate, forcing a transition to horizontal scaling (increasing cores). The world-wide HPC community is now planning an ambitious decade-long push to exascale with multiple (broad) paths forward, and significant technical uncertainty as to which one(s) will ultimately succeed in reaching the stated performance goals.

Creating software which is simultaneously stable and functional enough to serve the long-term needs of its users, and flexible enough to respond to the changing hardware environment without requiring undo effort will be one of the prominent challenges of the coming decade (and beyond).

In this talk, we will examine some of the key trends in hardware architecture and other issues facing software developers, and discuss techniques on the software side that can help developers adapt to and benefit from the rapid advances in computing power expected over the next decade.

List of participants

Balaji	V.	Princeton University
Berthou	Jean-Yves	EDF
Carter	Mick	Met Office
Caubel	Arnaud	IPSL
Coquart	Laure	CNRS-CERFACS
Coulaud	Olivier	INRIA
Craig	Tony	NCAR
Duchaine	Florent	IMFT - CERFACS
Duda	Michael	NCAR/MMM
Dunlap	Rocky	Georgia Tech
Elwasif	Wael	Oak Ridge National Laboratory
Esnard	Aurelien	INRIA HiePACS / Cerfacs
Fisher	Mike	ECMWF
Fladrich	Uwe	SMHI
Ford	Rupert	Manchester University
Foujols	Marie-Alice	IPSL
Gurol	Selime	CERFACS
Hanke	Moritz	DKRZ
Hill	Richard	Met Office
Hummel	Stef	Deltares
Jacob	Robert	Argonne National Laboratory
Kirchner	Ingo	Freie Universitat Berlin, Institute of Meteorology
Larson	Jay	Argonne National Laboratory
Liu	Li	Tsinghua University
Maisonnave	Eric	CERFACS
Mohammadi	Bijan	CERFACS
Moine	Marie-Pierre	CERFACS
Morel	Thierry	CERFACS
Mullerworth	Steve	Met Office
Oehmke	Robert	NOAA
O'Kuinghttons	Ryan	NOAA/CIRES
Osprey	Annette	NCAS-CMS, University of Reading
Piacentini	Andrea	CERFACS
Ralph	Adam	ICHEC
Ramos Buarque-Giordani	Silvana	Meteo-FRANCE
Redler	Rene	MPI Meteorology
Riley	Graham	University of Manchester
Riviere	Olivier	Meteo-FRANCE (CNMR/GMAP)
Senesi	Stephane	Meteo-FRANCE
Sevault	Florence	METEO-FRANCE
Urzay	Javier	CERFACS
Valcke	Sophie	CERFACS
Vertenstein	Mariana	National Center for Atmospheric Research
Vuchener	Clement	INRIA HiePACS
Wang	Xiaoge	Tsinghua University
Yang	Guangwen	Tsinghua University