The Model for Prediction Across Scales: meshes and software framework

Michael Duda*[1], Todd Ringler[2], and William Skamarock[1]

[1]National Center for Atmospheric Research†
[2]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 non-hydrostatic 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[1] 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.

*Corresponding author e-mail address: *duda@ucar.edu*
†NCAR is sponsored by the National Science Foundation

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

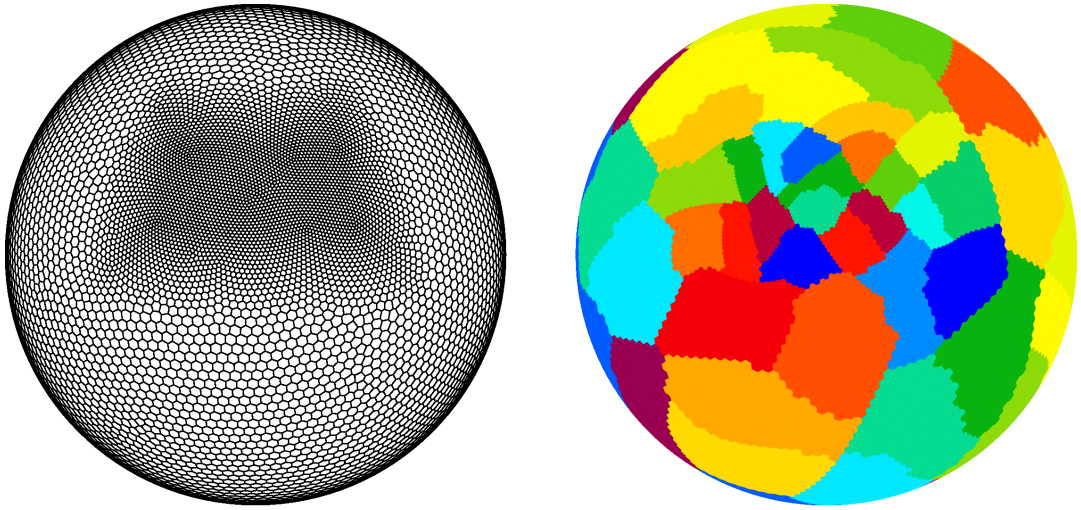

Figure 1: (a) An example of an SCVT with refinement targeted over a region in the Northern Hemisphere. (b) A parallel decomposition of the SCVT into 64 bocks of cells.

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.

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.

The MPAS 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 post-processing 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, an MPI 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

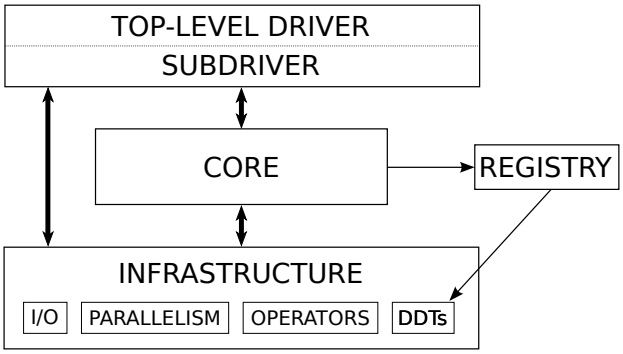

Figure 2: The high-level MPAS architecture with its three main components: the driver layer, a model core, and model infrastructure; the Registry is a CASE tool used to generate customized DDTs as well as code that would be otherwise tedious to write and maintain.

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.

4. Coupling in MPAS

With the MPAS software in a relatively immature state — the current working framework is still considered a 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 coupled-system 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.

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 soft-

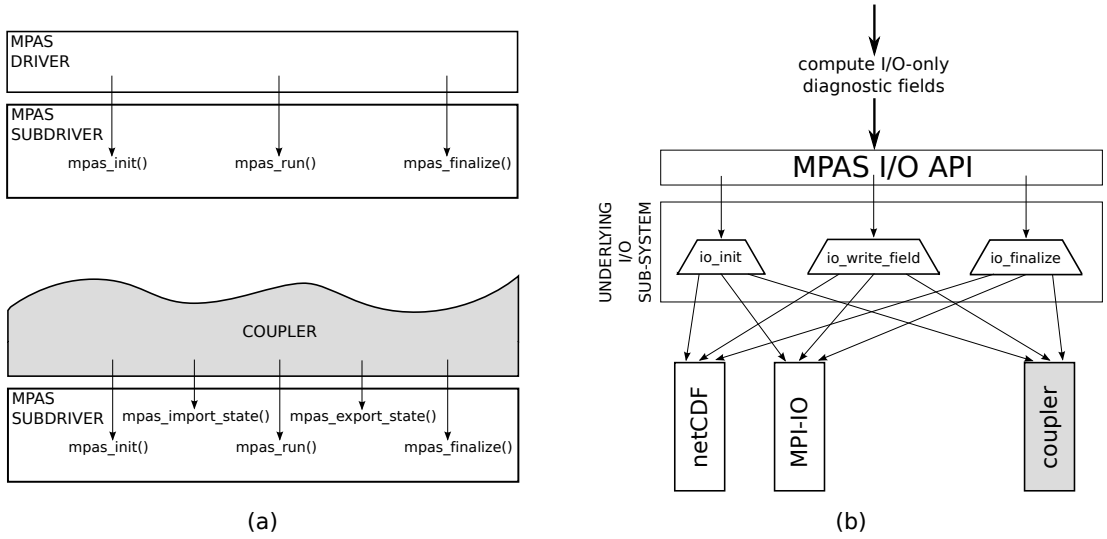

Figure 3: (a) Coupling with MPAS as a component may be accomplished by replacing the top-level driver with a coupler or driver from a larger Earth-system model, and implementing additional routines in the sub-driver. (b) Coupling via sends and receives of fields could be accomplished by implementing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

ware 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 the MPAS 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, and M. 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.