MPAS Halo Exchange

February 17, 2012

Contents

Chapter 1

Introduction

This document proposes a more flexible methodology for the halo exchanges used in MPAS. The current framework supports only the exchange of the entire halo region, while the proposed changes make partial halo exchanges possible. The motivation for partial halo exchanges is to reduce halo communication for instances when only a subset of the halo stencil is required. There already exists known instances in the atmospheric dynamical core where only the layer of halo cells immediately bordering a block's owned cells need values updated, and conversely instances where only the outer halo stencil requires updates. The intent of this proposal is to generalize the specification of halo layers so that, where appropriate, it would be straightforward to adjust any exchange to take advantage of reduced stencil communication.

The modifications outlined in this document also simplify the interface for calling halo exchanges and provide an argument list that is compatible with halo exchange operating on multiple blocks of cells per MPI task.

Chapter 2

Definition of Halo Layers

It is necessary that we establish a definition of what elements a halo layer contains. For cells, this definition is straightforward. The first halo layer consists of the ghost cells immediately surrounding a block's owned cells. The second halo layer consists of the ghost cells immediately surrounding the first halo layer. MPAS currently defines only two cell halo layers, though were this to be expanded, additional halo layers would be defined in the same manner. Figure 2.1 provides an illustration of two layers of halo cells. Naturally, a block's entire halo region is the union of the halo layers.

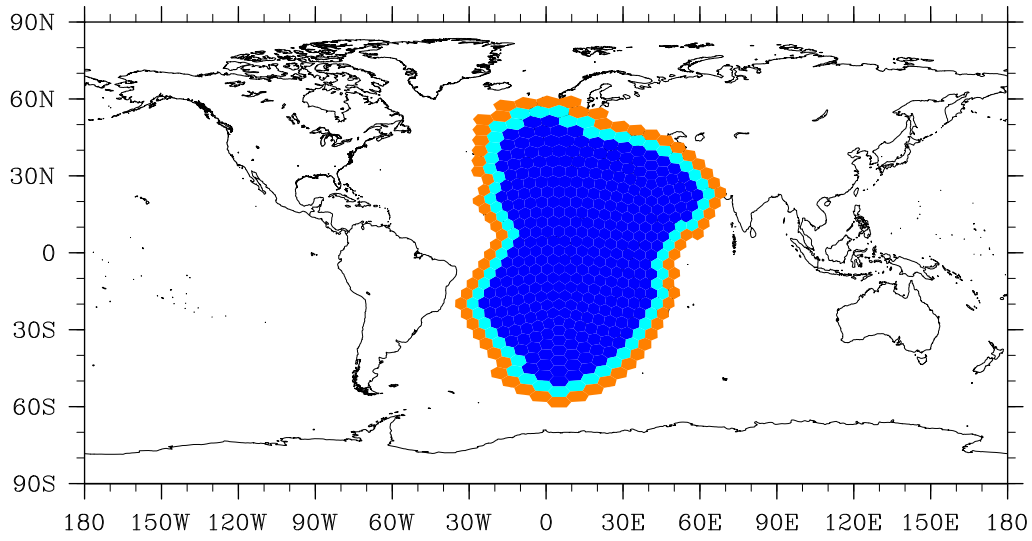

Figure 2.1: The dark blue cells are the block's owned cells. The light blue are the first halo layer cells and the orange are the second halo layer cells.

For edges, the definition of a halo layer is more nuanced. The number of edge halo layers is always one more than the number of cell halo layers. To see why this is, consider the current case where the halo consists of two layers of ghost cells; the three edge halo layers would be defined as follows:

- layer 1 : every edge bordering an owned cell that is not an owned edge
- layer 2 : edge of a first-layer ghost cell that is not a first-layer ghost edge
- layer 3 : every edge of a second-layer ghost cell that is not a second-layer ghost edge

The definition of a vertices halo layer is essentially the same as that of a halo edge. Consider the three edge layers defined above, only replace the word "edge" with "vertex". Figure 2.2 provides an illustration of three layers of halo edges corresponding the cell halo layers in Figure 2.1.

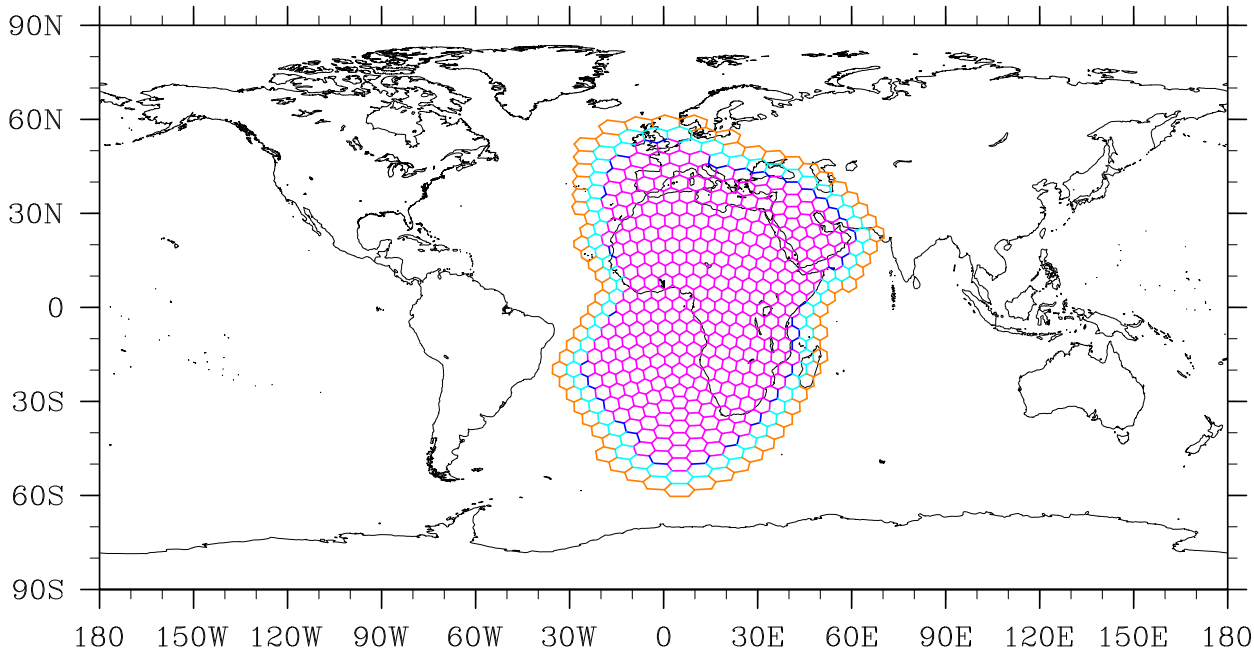

Figure 2.2: The pink edges are the block's owned edges. The dark blue edges on the pink perimeter are the first halo layer edges, the light blue are the second halo layer edges, and the orange are the third halo layer edges.

Chapter 3

Requirements

The halo exchange in MPAS has the following requirements.

- Allow the user to specify an exchange of subsets of the halo layers (e.g., only layer 1, only layer 2, layers 1 and 2)
- Support an arbitrary number of halo layers
- Work on field derived data types rather than Fortran arrays to support multiple blocks per MPI task

Chapter 4

Interface

The halo exchange requirements described in the previous section require modification to the exchange halo subroutines.

- The current exchange subroutines take a number of parameters (`dminfo, arrayIn, arrayOut, dim1, dim2, nOwnedList, nNeededList, sendList, recvList`) that will be replaced with the singe parameter `field` of the appropriate field type. The field types have been modified such that they contain all necessary information for the exchange, and passing the field into the subroutines will enable the future implementation of multiple blocks per MPI task. Additionally, the modified interface simplifies the calling of the subroutine and avoids unintentionally calling the exchange routines with incorrect dimensions or exchange lists.
- Currently, a separate halo exchange routine must be called for every field type, for example, `mpas_dmpar_exch_halo_field1dInteger`, `mpas_dmpar_exch_halo_field2dReal`, etc. The proposed changes would encapsulate these various subroutines in an interface, so that for any field type a call can be made to `mpas_dmpar_exch_halo_field`.
- The new exchange interface would include an additional optional parameter that specifies the particular halo layers to exchange. This parameter is an array of the halo indices to exchange. If this parameter is not included when calling the exchange subroutine, the current behavior of communicating the entire halo region is the default.

---

subroutine

mpas_dmpar_exch_halo_field(field, haloIndices)

Input

fieldType :: field — *the field to exchange*

integer, dimension(:), optional :: haloIndices — *the indices of the halo layers to be exchanged*

- Note that this is an interface, and the fieldType can be of the following: field1DReal, field2DReal, field3DReal, field1DInteger, field2DInteger, field3DInteger.

Chapter 5

Implementation

- Update the `mpas_dmpar` module to match the interface described in Chapter 4
- Add to the `mpas_dmpar` module a routine to aggregate specified halo indices into a single exchange list per block. When no halo indices are specified, aggregate all halo layers.
- Update the exchange halo calls throughout the dynamical cores to match the new interface
- Update the `mpas_io_input` module to divide the halo regions into appropriate layers
- Remove obsolete block loops that surround calls to the exchange routines