Revisions to MPAS block decomposition routines

January 24, 2012

Contents

Chapter 1

Introduction

In order to support multiple blocks of cells per MPI task, there are a number of development issues that need to be addressed:

1. Update/extend the fundamental derived types in mpas_grid_types.F. In order for other parts of the infrastructure to handle multiple blocks per task in a clean way, we'll need to be able to pass a head pointer to a field into a routine, and have that routine loop through all blocks for that field, with information about which cells/edges/vertices in that field need to be communicated.

2. Decide on a new MPAS I/O abstraction layer, which will provide a high-level interface to the PIO layer for the rest of MPAS. This layer should work with blocks of fields, and make it possible to define an arbitrary set of I/O streams at run-time.

3. Add a new module to parse a run-time I/O configuration file that will describe which fields are read or written to each of the I/O streams that a user requests via the file. This module will make calls to the new MPAS I/O layer to register the requested fields for I/O in the requested streams.

4. Update the mpas_dmpar module to support communication operations on multiple blocks per task. This will likely involve revising the internal data structures used to define communication of cells between tasks, and also require revisions to the public interface routines themselves.

5. Modify the block_decomp module to enable a task to get a list of cells in more than one block that it is to be the owner of. Implemented in the simplest way, there could simply be a namelist option to specify how many blocks each task should own, and the block_decomp module could look for a graph.info.part.n file, with n=num_blocks_per_task*num_tasks, and assign blocks k, 2k, 3k, ..., num_blocks_per_task*k to task k.

This document concerns the last item, namely, the extensions to the block decomposition module that will be necessary for supporting multiple blocks per task in other infrastructure modules.

Chapter 2

Requirements

The changes to the block decomposition module should enable an MPI task to get a list of it's owned cells, as well as the block number each of those cells lives on within it's task.

- Domain decomposition information needs to be read from a file based on number of total blocks as opposed to number of MPI tasks.
- Block decomposition routines need to provide a list of cells in a on a processor, as well as a list of blocks on a processor, and the cells that make up each block.

Chapter 3

Design

We propose several changes to the block decomposition module in order to support multiple blocks per MPI task. Currently, in order to support the case where there are multiple blocks per MPI task a namelist parameter needs to be added that will allow these two values to differ.

The namelist.input files will have the value config_number_of_blocks added as an integer field to the io section. This option will have a default value of 1.

Inside mpas_block_decomp.F, the mpas_block_decomp_cells_for_proc needs to be changed. To not only read in all cells in all blocks, but also their block numbers.

The meaning of the contents of graph.info.part.N needs to change from the processor ID that owns a cell, to the global block number for a cell. This means the file that is read in with have N = config_number_of_blocks.

Given a graph.info.part.N file, the global block number needs to be mapped into both an owning processor number, and a local block id. The local block id does not need to be computed within mpas_block_decomp_cells_for_proc as long as the mapping is available or known.

The api for mpas_block_decomp_cells_for_proc will change from

```
subroutine mpas_block_decomp_cells_for_proc(dminfo, partial_global_graph_info, local_cell_list)
```

to

```
subroutine mpas_block_decomp_cells_for_proc(dminfo, partial_global_graph_info, local_cell_list, local_block_list )
```

where local_cell_list is a list of cells owned by a processor, and local_block_list is a list of global block ids that describes which block owns a cell.

mpas_block_decomp_cells_for_proc will perform the same regardless of number of processors to enable the use of multiple blocks on a single processor.

To begin, the mapping from global block id to owning processor number and local block number is as follows.

```
blocks_per_proc = config_number_of_blocks / dminfo % nprocs

local_block_id = mod(global_block_id, blocks_per_proc)
owning_proc = (global_block_id) / blocks_per_proc
```

An array called global_block_list needs to be added, which will be similar to global_cell_list except that instead of storing cell ids it will store global block ids that should be paired with the cell id. For example, global_block_list(10) gives the global block number that owns the cell from global_cell_list(10).

An additional mpas_dmpar_scatter_ints needs to be added to communicate the global_block_list into nprocs copies of local_block_list containing a list of the global block ids that a processor owns.

Chapter 4

Implementation

Should we outline a plan for implementing these changes?