A short introduction to the MPAS framework

Matt Hoffman
5/16/2022
MPAS History

- Spearheaded by Todd Ringler (LANL) and Bill Skamrock (NCAR) following development of TRSK discretization scheme for geophysical fluids (2010)
- Vision: shared framework for building geophysical fluid models using unstructured, global, variable resolution Voronoi grids
- Initially: MPAS-Shallow Water, MPAS-Ocean (LANL), MPAS-Atmosphere (NCAR)
- Later: MPAS-Seaice, MPAS-Land Ice (now MALI)
MPAS Repository history

- **Original svn repo:** [https://github.com/MPAS-Dev/MPAS-Legacy](https://github.com/MPAS-Dev/MPAS-Legacy)
  - First commit: Mar 4, 2010
- **First git repo (originally private):** [https://github.com/MPAS-Dev/MPAS](https://github.com/MPAS-Dev/MPAS)
  - I believe contains all the svn history
- **Second git repo (entirely public):** [https://github.com/MPAS-Dev/MPAS-Model](https://github.com/MPAS-Dev/MPAS-Model)
  - Continuous git history with previous repo
  - Was a git submodule within E3SM (main motivation for going public)
- **E3SM repo:** [https://github.com/E3SM-Project/E3SM](https://github.com/E3SM-Project/E3SM)
  - Transitioned May 17, 2021
  - Older history not accessible - refer to previous repo
  - Framework (including standalone Makefile) as a ‘component’ at: components/mpas-framework/
  - MPAS-Ocean, MPAS-Seaice, MPAS-Albany-Land Ice each have separate components
  - MALI development primarily in a fork: [https://github.com/MALI-Dev/E3SM](https://github.com/MALI-Dev/E3SM)
MPAS Framework history

- Shared framework for data structures, mesh operations, parallelization, i/o, configuration parsing, timekeeping, physics-agnostic operations
- Primary authors:
  - Doug Jacobsen (LANL, now at Google)
  - Michael Duda (NCAR)
- Desire to make use of modern software features (templating, operator overloading, classes) but still use Fortran due to familiarity for domain experts
  - many custom implementations, resulting in challenges for long term maintenance, ‘unusual’ Fortran code, mixed language code, etc.
- Desire to be very general - features any geophysical ‘core’ could use
  - some features are overly general for any given physics application and perhaps more cumbersome than necessary for a specific application
- Currently in Framework:
  - 72,781 lines of Fortran code
  - 1845 lines of include files
  - 9554 lines of C code
Documentation

- Original documentation repo: https://github.com/MPAS-Dev/MPAS-Documents
- Framework design documents: https://github.com/MPAS-Dev/MPAS-Documents/tree/master/shared
Development guidelines

As developers of MPAS, we attempt to make the code look as uniform as we can across the entire code-base. In order to enforce this, there are a set of guidelines developers should follow.

- Each core has a name, and an abbreviation. For example, the shallow water core is called sw and its abbreviation is sw, but the ocean core is called ocean and its abbreviation is ocn.

- All subroutines should be named in a manner which prevents namespace conflicts. Shared functions/subroutines are simply named mpas.subroutine.name. Core specific functions/subroutines are named mpas.abbrev.subroutine.n (where abbrev is replaced with the cores abbreviation).

  e.g. mpas.atm.time_integration

- Subroutine names should all be lower case, with underscores in place of spaces (e.g. see above).

- Variable names should be mixed case (e.g. cellsOnCell rather than cells.on.cell).

- In general, variable names should be self-descriptive (e.g. nCells rather than n).

- Subroutines and modules should be appropriately documented. Shared portions of MPAS code use doxygen comments, but core developers are free to decide what method of documenting they prefer.

- Development of shared parts of MPAS need reviews from multiple core maintainers prior to a merge.

- Development within a core should be approved by other core developers before being merged into that core.

- Development within a core should follow the practices of that core’s developer group, for documentation etc.

- Core related testing is the responsibility of that core’s maintainers/developers.
Framework structure

components/mpas-framework/src

- core_landice  ->  ../../mpas-albany-landice/src
- core_ocean  ->  ../../mpas-ocean/src
- core_seaice  ->  ../../mpas-seaice/src
- core_sw
- core_test
- driver
- external
  - esmf_time_f90  Frozen version of ESMF timekeeper
  - ezxml  XML parser for streams
- framework  Basic MPAS functionality
- operators  Useful algorithms that are not core-specific (e.g. geometric operations)
- tools
  - input_gen  Tool to generate default namelist & streams files
  - registry  Parser of Registry files (incl. conversion to Fortran code)
Registry:
- XML file defining model dimensions, namelist options, stream definitions, variables
- parser converts to Fortran (and some C) code

Data structures:
- `var_struct` - grouping of model variables
  - `Var_array`
  - `Var`
Registry

Allowable attributes defined in: src/tools/registry/Registry.xsd

- dimensions
- nml records & options
- streams
- var_structs, var_arrays, vars
  - name
  - type
  - dimensions
  - name_in_code
  - units
  - description
  - default_value
  - persistence
  - packages
  - (time_levs, array_group)
Data structures

Internal MPAS code:

As the blocklist structure is a linked list of blocks, one can iterate over the list of blocks in the following manner.

```fortran
    type (block_type), pointer :: block_ptr

    block_ptr => domain % blocklist
    do (while(associated(block_ptr))
        ... do stuff on block ...
        block_ptr => block_ptr % next
    end do
```

Figure 4.1: Visual diagram of MPAS derived data type layout.
Data structures: Internal MPAS code: ‘pools’

- similar to a class/derived type
  - each data structure has attributes and methods, child members, etc.
  - implemented from scratch with a lot of linked lists
define pointer variables as destinations for pools retrievals

mpas_pool_get_subpool to retrieve structs

mpas_pool_get_config to retrieve nl options

mpas_pool_get_array to retrieve actual arrays

Notes:

- Pool routines are case sensitive even though Fortran is not!
- Typos/faulty retrievals commands cannot be detected at compile time, only at run time! (Major limitation of the pools data structure)
- Compiling with DEBUG=true necessary to get useful error message
type mpas_pool_member_type
integer :: keyLen
integer :: contentType
integer (8) :: data => null()
type (mpas_pool_data_type), pointer :: data => null()
type (mpas_pool_member_type), pointer :: next => null()
type (mpas_pool_member_type), pointer :: iteration_next => null()
type (mpas_pool_member_type), pointer :: iteration_prev => null()
end type mpas_pool_member_type

! For storing fields

type mpas_pool_type
integer :: size
integer (8) :: table => null()
type (mpas_pool_member_type), pointer :: iterator => null()
type (mpas_pool_member_type), pointer :: iteration_head => null()
type (mpas_pool_member_type), pointer :: iteration_tail => null()
end type mpas_pool_type

! For storing fields

type mpas_pool_data_type
integer :: contentsDimensions
integer (8) :: contentsTimeLevs

! For storing config options, dimensions, and packages

character (len=StrKind) :: key
integer (8) :: next => null()
type (mpas_pool_data_type), pointer :: data => null()
type (mpas_pool_member_type), pointer :: next => null()
type (mpas_pool_member_type), pointer :: iteration_next => null()
type (mpas_pool_member_type), pointer :: iteration_prev => null()
end type mpas_pool_data_type

mpas_field_types.inc
src/framework
- .F -> code
- .inc -> derived type definitions
MPAS Timekeeping

- xtime is most fundamental time variable
  - string of format: ‘YYYY-MM-DD hh:mm:ss’
- cores have defined something like ‘daysSinceStart’ as a real variable but not actually used by framework
- MPAS_Time_type (ESMF_Time) used for most actual time operations
- MPAS wraps and old, frozen version of ESMF timekeeper
  - (E3SM might use a more current version?)
  - handles Y,M,D,h,m,s, including different calendars (e.g. leap years) and overloaded mathematic operators
  - concept of ‘time’, ‘interval’
  - concept of ‘clock’, ‘alarm’
  - Addition of a CF compliant time variable may require adjusting timekeeping routines (or possibly updating the version of the ESMF timekeeper?)
mpas_timekeeping_types.inc

```fortran
integer, parameter :: MPAS_MAX_ALARMS = 40

integer, parameter :: MPAS_NOW = 0, &
  MPAS_START_TIME = 1, &
  MPAS_STOP_TIME = 2

integer, parameter :: MPAS_FORWARD = 1, &
  MPAS_BACKWARD = -1

integer, parameter :: MPAS_GREGORIAN = 0, &
  MPAS_GREGORIAN_NOLEAP = 1, &
  MPAS_360DAY = 2

type MPAS_Time_type
  type (ESMF_Time) :: t
end type

type MPAS_TimeInterval_type
  type (ESMF_TimeInterval) :: ti
end type

type MPAS_Alarm_type
  character (len=ShortStrKIND) :: alarmID
  logical :: isRecurring
  logical :: isSet
  type (MPAS_Time_type) :: ringTime
  type (MPAS_Time_type) :: prevRingTime
  type (MPAS_TimeInterval_type) :: ringTimeInterval
  type (MPAS_Alarm_type), pointer :: next => null()
end type

type MPAS_Clock_type
  integer :: direction
  integer :: nAlarms
  type (ESMF_Clock) :: c
  type (MPAS_Alarm_type), pointer :: alarmListHead => null()
end type
```

mpas_timekeeping.F

```fortran
module mpas_timekeeping

use mpas_kind_types
use mpas_derived_types
use mpas_dmpar
use mpas_threading
use mpas_log
use ESMF

subroutine mpas_timekeeping_init
subroutine mpas_timekeeping_finalize
subroutine mpas_timekeeping_set_year_width!
subroutine mpas_create_clock
subroutine mpas_destroy_clock
subroutine mpas_set_clock_direction
subroutine mpas_set_clock_timestamp
subroutine mpas_advance_clock
subroutine mpas_set_clock_time
subroutine mpas_add_clock_alarm
subroutine mpas_remove_clock_alarm
subroutine mpas_minimum_alarm_interval
subroutine mpas_print_alarm
subroutine mpas_get_clock_ringing_alarms
subroutine mpas_reset_clock_alarm
subroutine mpas_adjust_alarm_to_reference_time
subroutine mpas_calibrate_alarms
subroutine mpas_set_time
subroutine mpas_get_time
subroutine mpas_set_time_interval
subroutine mpas_get_time_interval
subroutine mpas_interval_division
subroutine mpas_split_string
subroutine mpas_get_month_day
subroutine mpas_expand_string!

end module mpas_timekeeping
```