IGC7 Clinic:

GEOS-Chem in massively parallel and ESM environments, GEOS-5 CTM

Agenda:

1) Getting Set-up and Logged in for the clinic.
2) GCHP Structure
3) Software requirements
4) How HPC works for GEOS-Chem
5) ESMF basics
6) MAPL basics
7) How to build it*
8) Experiment Setup*
9) Model Execution*

* Preliminary – this is likely to change (for simplicity and for the better).

IGC7: Getting Set-up and Logged in for the clinic

1) Log into <username>@neologin.seas.harvard.edu using the ssh command:
   e.g. $ ssh –X igc7-00@neologin.seas.harvard.edu

2) Log in to a compute session:
   WITH active X Display: $ qsh
   WITHOUT active X Display: $ qlogin

3) Copy over the GCHP source code
   $ cp –r /scratch/global/igc7/shared/GCHP ./

4) Copy over the experiment directory
   $ cp –r /scratch/global/igc7/shared/RunDir ./

5) OK – sit tight. Time to talk.

IGC7: GCHP Structure

```
[mlong@login04 GCHP]$ ls
GIGC       GeosCore  Headers   Makefile              REVISIONS  help
GTMM       GeosUtil  ISOROPIA  Makefile_header.mk    bin        lib
GeosApm    HEMCO     KPP       NcdfUtil              doc        mod
```

IGC7: GCHP Structure

[mlong@login04 GCHP]$ cd GIGC

[mlong@login04 GCHP]$ ls

Chem_GridCompMod.F90
ESMF
FVdycoreCubed_GridComp
GCSA-HowTo.docx
GEOSChem.F90
GEOS_HyCoords.H
GEOS_ctmEnvGridComp.F90
GIGC.mk
GIGC_Connections.H
GIGC_GridCompMod.F90
HEMCO_Includes_AfterRun.H
HEMCO_Includes_BeforeRun.H
Includes_After_Dyn.H
Includes_After_Run.H
Includes_Before_Dyn.H
Includes_Before_Run.H
Makefile
Registry
Shared
gc_land_interface.F90
gigc_chem_utils.F90
gigc_chunk_mod.F90
gigc_diagnostics_mod.F90
gigc_finalization_mod.F90
gigc_initialization_mod.F90
gigc_mpi_wrap.F90
gigc_test_utils.F90
gigc_type_mod.F

IGC7: GCHP Structure

[mlong@login04 GCHP]$ cd GIGC

[mlong@login04 GCHP]$ ls

Chem_GridCompMod.F90
ESMF
FVdycoreCubed_GridComp
GCSA-HowTo.docx
GEOSChem.F90
GEOS_HyCoords.H
GEOS_ctmEnvGridComp.F90
GIGC.mk
GIGC_Connections.H
GIGC_GridCompMod.F90
HEMCO_Includes_AfterRun.H
HEMCO_Includes_BeforeRun.H
Includes_After_Dyn.H
Includes_After_Run.H
Includes_Before_Dyn.H
Includes_Before_Run.H
Makefile
Registry
Shared
gc_land_interface.F90
gigc_chem_utils.F90
gigc_chunk_mod.F90
gigc_diagnostics_mod.F90
gigc_finalization_mod.F90
gigc_initialization_mod.F90
gigc_mpi_wrap.F90
gigc_test_utils.F90
gigc_type_mod.F

IGC7: Software Requirements

Primary Requirements

Fortran Compiler
- Intel-11.1; Intel-13.1; Intel-15.0

C Compiler
- gcc-4.6.3; gcc-4.8.0

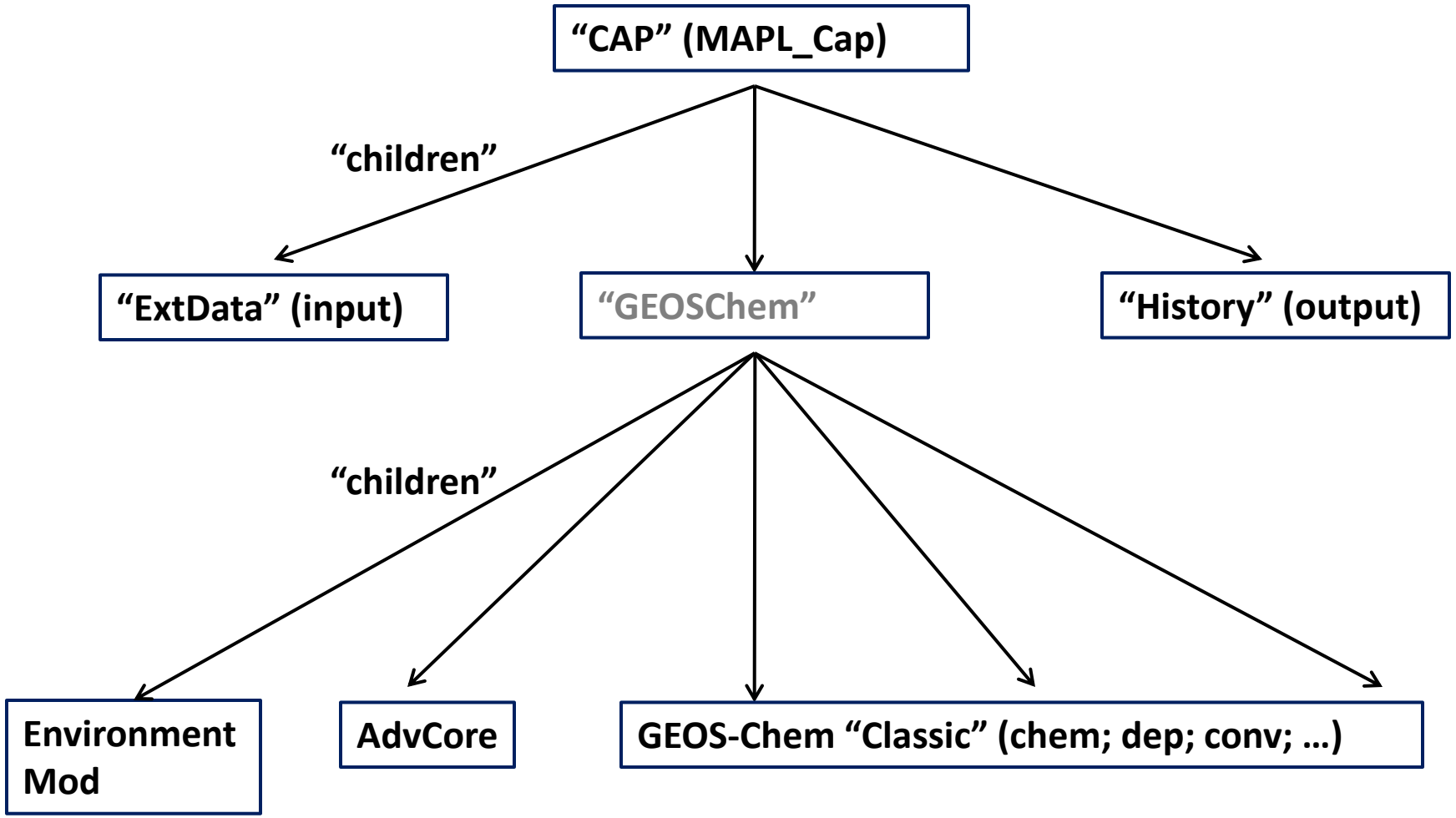
MPI Compiler
- OpenMPI-1.4.1; 1.7.2 (Ifort/gcc)
- MVAPICH2-1.8.1 (Intel-13.1; Infiniband)
- SGE-MPT-2.11

Secondary Requirements

NetCDF
- If NetCDF-3, then you're OK.

- If NetCDF-4 w/ parallel I/O enabled
 -- Curl
 -- ZLib
 -- HDF5 (compiled with mpicc)

IGC7: How HPC works for GEOS-Chem



"Parent" controls communication between "children"

The Rules for *Writing* an ESMF Gridded Component Module

- It must contain the four routines (methods):
  - SetServices
  - Initialize
  - Run
  - Finalize
- The only public quantity is the SetServices routine.
- SetServices must register the three IRF methods with ESMF.
- The argument list of all four routines is prescribed by ESMF

Argument lists of the four ESMF Methods--SetServices plus three IRF Methods

```
type(ESMF_GridComp)        :: GC
type(ESMF_State)           :: IMPORT, EXPORT
type(ESMF_Clock)           :: CLOCK
integer                    :: RC

call SetServices(GC,rc)

call ESMF_GridCompInitialize (GC, import, export, clock, rc)

call ESMF_GridCompRun       (GC, import, export, clock, rc)

call ESMF_GridCompFinalize  (GC, import, export, clock, rc)
```

IGC7: ESMF Basics

```
Module MyWorldGridCompMod

use ESMF_mod

private
implicit none
public SetServices

contains
!==============================================

Subroutine SetServices(GC,RC)
 type(ESMF_GridComp), intent(INOUT) :: GC
 integer, optional,           intent(OUT)    :: RC

   call ESMF_GridCompSetEntryPoint ( GC, ESMF_SETRUN,          Run,          &
                                     ESMF_SINGLEPHASE, rc)
   If(RC /= ESMF_SUCCESS) return

   call ESMF_GridCompSetEntryPoint ( GC, ESMF_SETINITIALIZE,  Initialize,    &
                                     ESMF_SINGLEPHASE, rc)
   If(RC /= ESMF_SUCCESS) return

   call ESMF_GridCompSetEntryPoint ( GC, ESMF_SETFINALIZE,    Finalize,     &
                                     ESMF_SINGLEPHASE, rc)

End Subroutine SetServices
```

Import & Export States

- User components under ESMF use special interface objects for component to component data exchanges:
  - Import state: what a component needs to run
  - Export state: what a component can produce for the consumption of other components
  - Import/export states are defined on the *grid* of the component, with no knowledge of the coupling environment
- The contents of Import/Export State are self-describing.
- High-performance considerations:
  - Import State and Export State contents may use references or pointers to component data, so that costly data copies of potentially very large data structures can be avoided where possible.

What is MAPL?

- A toolkit for writing ESMF gridded components
- A system for building complex applications from MAPL/ESMF gridded components
- A set of utilities that extend ESMF functionality

MAPL Evolved from GEOS-5 development

MAPL_Generic

- Provides generic versions of SetServices, Initialize, Run, and Finalize methods
- These can be used as ingredients in a standard recipe for writing ESMF gridded components.
  - When you write a GC,
    - You write a SetServices, that calls Generic_SetServices
    - You can then default any of the IRF methods to a MAPL_Generic version
    - Or you can write your own IRF method and call the generic version from there
- Creates an Internal ESMF state in the gridded component that complements the ESMF Import and Export states
- Provides a means of describing the contents of the three (Im/Ex/In) states
- Manages the creation, initialization, and destruction of the three states

MAPL_History

- An ESMF gridded component within MAPL that can service the exports states of an entire hierarchy
- Writes data collections with selectable horizontal and vertical grids, resolution, frequency, and time-averaging
- Uses MAPL_CFIO as its I/O layer and can thus produce netcdf, hdf, grads,…

MAPL_CFIO

- An I/O layer for ESMF Array, Field, Bundle, and State classes.
- Does not rely on any other aspect of MAPL
- Writes create CF-compliant output stream coordinates from ESMF grids.
- Reads can automatically create ESMF Fields and Bundles from data files.
- Data is as CF-Compliant as the bundle it came from.
- Currently supports netcdf, hdf, and grads, and can translate between them.

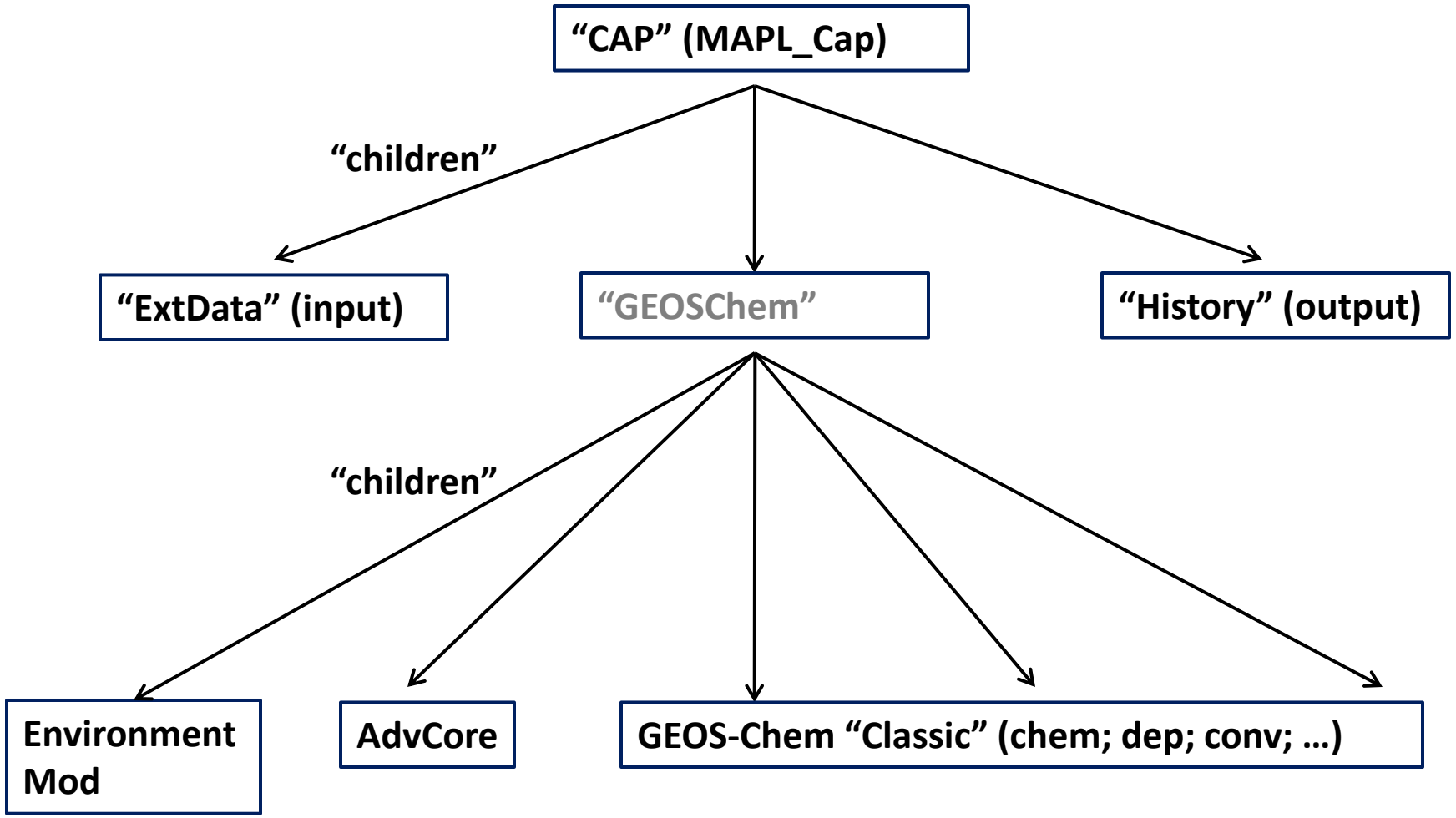
MAPL_Utils

- Miscellaneous utility packages, including
  - Simple profiler
  - Simple error handler that implements a traceback through MAPL routines
  - A very general astronomy, based on ESMF clock/calendar and field classes

IGC7: MAPL Basics

```
#define I_AM_MAIN

#include "MAPL_Generic.h"

Program GIGC_Main

  use MAPL_Mod
  use GIGC_GridCompMod, only:  ROOT_SetServices => SetServices

  implicit none

  integer          :: STATUS
  character(len=18) :: Iam="GIGC_Main"

  logical          :: AmIRoot

  call MAPL_CAP(ROOT_SetServices, AmIRoot=AmIRoot, rc=STATUS)
  VERIFY_(STATUS)

  call exit(0)

end Program GIGC_Main
```

IGC7: How HPC works for GEOS-Chem



"Parent" controls communication between "children"

IGC7: How to Build It

Compiling GEOS-Chem Classic

$ make **COMPILER**=ifort **EXTERNAL_GRID**=yes **DEVEL**=yes **MET**=geos-fp **GRID**=4x5 **DEBUG**=yes

IGC7: How to Build It

Compiling GEOS-Chem HP

$ make **COMPILER**=ifort **EXTERNAL_GRID**=yes **DEVEL**=yes **MET**=geos-fp **GRID**=4x5 **DEBUG**=yes ***hpc***

IGC7: How to Build It

Compiling GEOS-Chem HP

$ make **COMPILER**=ifort **EXTERNAL_GRID**=yes **DEVEL**=yes **MET**=geos-fp **GRID**=4x5 **DEBUG**=yes ***hpc***

If it can't find GIGC:

../Makefile_header.mk:729: *** "Unable to find the GIGC configuration file. Have you downloaded the GIGC?". Stop.

IGC7: Experimental Setup

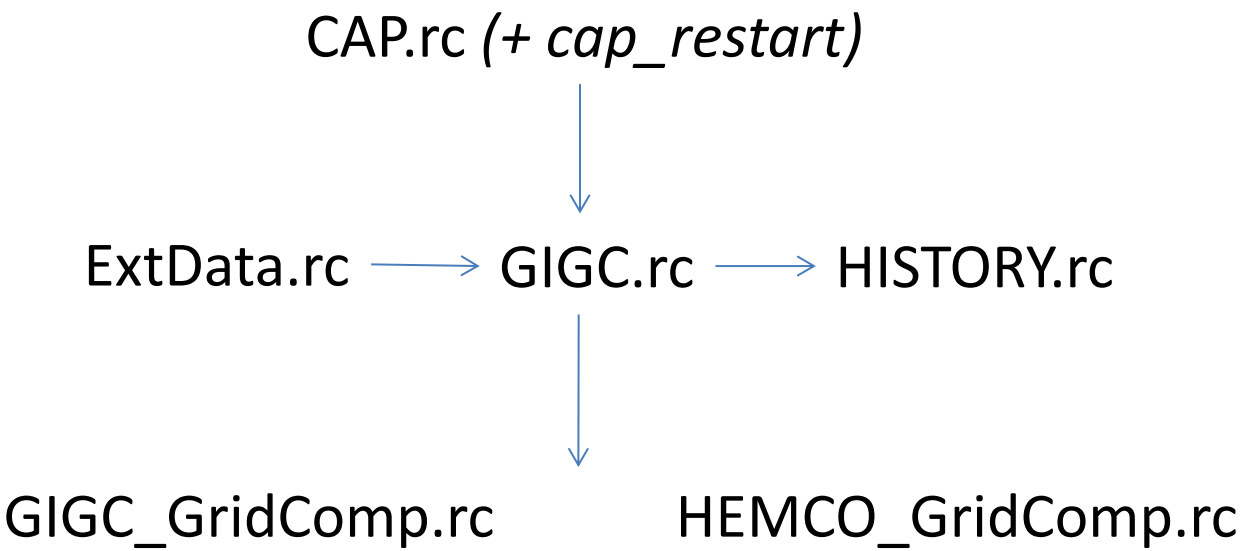
```
[mlong@login04 RunDir]$ ls
CAP.rc                          chemga.dat.rc
Chem_Registry.rc                fvcore_layout.rc
ExtData.rc                      geoschemchem_internal_rst.nc4
FJX_j2j.dat.rc                  globchem.dat.rc
FJX_spec.dat.rc                 input.geos.rc
GIGC.rc                         jv_spec_aod.dat.rc
GIGC_GridComp.rc                jv_spec_mie.dat
HEMCO_Config.rc                 jv_spec_mie.dat.rc
HEMCO_GridComp.rc               mglob.dat.rc
HISTORY.rc                      ratj.d.rc
tracerinfo.dat.rc               cap_restart
```

IGC7: Experimental Setup

CAP.rc :

```
[mlong@login04 tmpDir]$ more CAP.rc
MAPLROOT_COMPNAME: GIGC
       ROOT_NAME: GIGC
ROOT_CF: GIGC.rc
HIST_CF: HISTORY.rc

BEG_DATE:     20130301 000000
END_DATE:     20130910 000000
JOB_SGMT:     00000007 000000
NUM_SGMT:     20

DEFAULT_DURATION: 3
STOP_DATE: 20130307
POSTPROC: 0

HEARTBEAT_DT: 1800

MAPL_ENABLE_TIMERS: YES
MAPL_ENABLE_MEMUTILS: YES
PRINTSPEC: 0  # (0: OFF, 1: IMPORT & EXPORT, 2: IMPORT, 3: EXPORT)
```

IGC7: Model Execution



IGC7: Model Execution

Preconditions:
1)Number of CPUs = NX x NY (from GIGC.rc)
2)NY = factor of NX and 6 (Cubed-sphere Constraint)

NX x NY = 2 x 6 = 12

$mpirun –n 12 geos 2>&1 | tee run.log