Stand-Alone Grid-Independent GEOS-Chem

Developers Document

M. Long Dec., 10 2014 – First crack.

R. Yantosca – 12 Dec 2014 – Update

**About this Document**

This document is intended to be a working/living overview meant to help users/testers of the Beta verion of the Stand-alone HPC-capable GEOS-Chem CTM. We are all learning about and developing this beast together. Therefore, as you, the user and tester, learn about, develop, and improve this system, please edit this document accordingly. It is tracked by GIT in the GIGC repository.

**General Description**

To be added …

**How to Get & Update the Code**

1. Clone the gigc\_download package to your local disk space:

$ git clone <https://github.com/GCST/gigc_download.git>

This will create a gigc\_download directory in your disk space. This directory contains a single Makefile.

(2) Use GNU Make to start the download process. Specify a directory where you want GEOS-Chem to be installed.

$ cd gigc\_download

$ make GEOSCHEM\_DIR=../GEOS-Chem GEOSCHEM\_BRANCH=MegaChem\_1.0

This will clone the “MegaChem\_1.0” branch of the GC\_Bleeding\_Edge repository. If you omit the GEOSCHEM\_BRANCH option, then the downloader will give you the “Dev” branch of the GC\_Bleeding\_Edge repository.

 If the download is successful, you will see output such as this:

Cloning into '../GEOS-Chem'...

remote: Counting objects: 24926, done.

remote: Compressing objects: 100% (4482/4482), done.

remote: Total 24926 (delta 21071), reused 24141 (delta 20424)

Receiving objects: 100% (24926/24926), 20.53 MiB | 4.59 MiB/s, done.

Resolving deltas: 100% (21071/21071), done.

Checking connectivity... done.

Checking out files: 100% (595/595), done.

###################################################

### GEOS-Chem has been downloaded successfully! ###

###################################################

. . .

Cloning into '../GEOS-Chem/GIGC'...

remote: Counting objects: 7199, done.

remote: Compressing objects: 100% (3899/3899), done.

remote: Total 7199 (delta 3017), reused 7199 (delta 3017)

Receiving objects: 100% (7199/7199), 25.39 MiB | 5.05 MiB/s, done.

Resolving deltas: 100% (3017/3017), done.

Checking connectivity... done.

Checking out files: 100% (5406/5406), done.

###################################################

### GIGC has been downloaded successfully!      ###

###################################################

Note that the GIGC directory (which contains the ESMF, MAPL, and FVdycore code) is currently built as a subdirectory of GEOS-Chem.  In a future release, we may try to let you install that into a directory path of your choice.

**Model Structure**

The stand-alone HPC version of GEOS-Chem is in two parts: The first part is simply the GEOS-Chem code. It exists as-you-get-it, and includes all of the hooks to operate in the HPC/ESMF environment. The code is enabled by c-preprocessor switches. But, it will not work without the second part. The second part contains the HPC guts, and a few more important pieces. It lives in a directory labeled GIGC, and is an independent git repository and code base than GEOS-Chem itself. Most of the description will pertain to the structure of the GIGC.

Within GIGC are three base library subdirectories, a Registry directory and the coupling code.

1. Directories
   1. GIGC/ESMF: This directory is an exact copy of the ESMF code v5.2.0rp2
   2. GIGC/Shared: This directory contains NASA-GMAO's MAPL and Shared library packages. These packages are used to facilitate coupling between components and provide the primary interface with the ESMF and I/O software.
   3. GIGC/FvdycoreCubed\_GridComp: This is the HPC-capable cubed-sphere dynamics core. It is taken from the GEOS-5 and modified for use offline in a CTM.
   4. GIGC/Registry: The Registry directory contains information used by MAPL at compile time to generate the fortran interface between the various quantites needed by GEOS-Chem and the ESMF/MAPL/FVDyCore code.
2. Fortran-90 Code

The F90 files in the GIGC directory are what supplement the use of GEOS-Chem's “main.F” code. They consist of ESMF & MAPL interface code that couple GEOS-Chem's routines in an ESMF environment. Developers will be expected to modify these files, but doing so will require some skill with using MAPL & ESMF.

**What Is Being Executed**

At this point, not everything is being actively run. The system is barebones, at this point.

1. Chemistry – FullChem is executed.
2. Dry Deposition
3. Wet Scavenging
4. HEMCO

**What ISN'T Being Executed**

There are no active diffusion or convection routines running at this point. This is a challenge that we hope you can help us resolve.

UCX is not active. Ask me why!

**How To Compile**

1. Software Environment
   1. **Operating System:**
      1. Linux, SLES with BASH, CSH or TCSH.
   2. **Fortran Compiler:**
      1. To date, the system has only been tested with the Intel Fortran Compiler. It has been shown to compile successfully with major versions 11 and 13. Version 14 fails to compile the prerequisite libraries successfully, and version 12 was not tested.
   3. **NetCDF:**
      1. Some version of netcdf is required. To date, netcdf version 3.6.X and versions 4.1.1 and 4.1.4 have been successful. Versions higher than v4.1 generally fail, though it is not clear why. At this point, reading IN netCDF restart files requires version 4 and requires that it be compiled with parallel operations enabled. This also requires that HDF5 be compiled with parallel ability. Version 3 does not have parallel capabilities.
      2. A serial-netcdf READ capacity has been written by Ben Auer at GMAO for our needs, and it will be incorporated in MAPL soon. It's not there yet (as of Dec. 10, 2014)!
   4. **MPI:** 
      1. GIGC has only been tested with OpenMPI with successful compilation and execution with versions 1.4.1, 1.7.2 and 1.8.1. OpenMPI is compiled with FC=ifort and CC=gcc. The system fails with CC=icc for unknown reasons.
   5. **What DOESN'T Work or HASN'T Worked**
      1. To date, MVAPICH has not worked. It would be worthwhile to get it to work since it is optimized for Infiniband interconnects, and this system is decidedly I/O heavy.
2. Environment Variables
   1. Much effort has been spent making this system build and run without much environmental configuration. At this point, there are only seven (7) variables that must be set by the user in order to compile: For example in bash, these definitions would look like:

export COMPILER=ifort

export GC\_BIN=/path/to/netcdf/bin/

export GC\_INCLUDE=/path/to/netcdf/include/

export GC\_LIB=/path/to/netcdf/lib

export ESMF\_COMPILER=intelgcc

export ESMF\_COMM=openmpi

export ESMF\_BOPT=O

Where:

* + 1. COMPILER=ifort tells GEOS-Chem to use commands for Intel Fortran. The other option, pgi, is not yet supported for running GEOS-Chem in ESMF/MPI mode.
    2. GC\_BIN points the binary directory of your local netcdf installation.
    3. GC\_INCLUDE points to the include directory of your local netcdf installation
    4. GC\_LIB points to the library directory of your local netcdf installation
    5. ESMF\_COMPILER specifies the Fortran and C compilers to use. Using intelgcc means that we will build the libraries with Intel Fortran but GNU C/C++.
    6. ESMF\_COMM specifies the type of MPI you are using. (We have only tried openmpi).
    7. ESMF\_BOPT specifies the Optimization level that you want ESMF to use. Using “O” will build optimization into ESMF. Using “g” will build ESMF with debugging features, which will execute much more slowly.

For example, on the Harvard system, my netCDF settings look like

export GC\_INCLUDE=/home/mlong/tools/netcdf/include

export GC\_LIB=/home/mlong/tools/netcdf/lib

export GC\_BIN=/home/mlong/tools/netcdf/bin

1. Compile command
   1. **Build:** Building GEOS-Chem coupled to the GIGC simply requires the inclusion of an additional tag in the make command: *hpc*. Appending *hpc* causes the make procedure to look for and revert to the contents of GIGC, and bypasses much of the serial GEOS-Chem code. A typical make command would look like  
        
      $ make MET=geosfp GRID=4x5 NO\_REDUCED=yes UCX=no *hpc*  
        
      This command compiled GEOS-Chem at 4x5 resolution for GEOS-FP meteorology on a full vertical grid, and with NO UCX stratospheric chemistry. Note, the 'UCX=no' must be included because, at this time, the code is not in place to appropriately load the necessary files and initialize the system.  
        
      Upon the first build using the '*hpc'* tag, the ESMF, MAPL, and FVDyCore libraries are compiled. They are only compiled one time after which a stub-file (e.g. mapl.install) is created in the ESMF, Shared and FVDyCore directories indicating that they have been built, and signaling the make procedure to bypass them on subsequent attempts. This minimizes overhead for small changes in the GIGC code.
   2. **Clean:** To remove the object and module files in the GEOS-Chem directory without removing the executable, type:  
        
      make clean  
        
      To remove all GEOS-Chem object, module, library files, and executables, type:  
        
      make realclean  
        
      as you would when using the serial GEOS-Chem code.
   3. **Wipeout:** Typing make clean or make realclean only removes GEOS-Chem compiled code and libraries. It does not remove any of the ESMF, MAPL or FVdycore libraries in the GIGC directory. If you should ever need to remove these libraries, then you will have to issue the following commands. PLEASE PROCEED WITH CAUTION!!!  
        
      This command will remove ESMF + MAPL + FVdycore in one fell swoop.  
        
      cd GIGC  
      make the\_nuclear\_option  
        
      or if you would like to remove these one at a time you can type:  
        
      cd GIGC  
      make wipeout\_esmf  
      make wipeout\_mapl  
      make wipeout\_fvdycore  
        
      Once you issue these commands, the ESMF + MAPL + FVdycore libraries will evaporate. You will need to rebuild these from scratch before running the Stand-alone Grid-Independent GEOS-Chem again.

**How To Run**

*Stay Tuned*

1. File IO
2. Diagnostics

**What NOT To Do**

1. Do NOT edit any files in ESMF, Shared, or FvdycoreCubed\_GridComp.

**Final Caveat Statement**

In a perfect world, you are able to glaringly see what we could do or should be done *better.* I personally cannot imagine a perfect scenario where something like this ever comes off without errors. In this perfect world, I would be out of a job! As such, please do NOT hesistate to chop and improve, but please follow general guidelines for clean and elegant code writing. Thank you for your time and effort.

**System-dependent notes**

odyssey.rc.fas.harvard.edu: Use these settings in your .bashrc file:

source new-modules.sh

module purge

module load git

module load legacy

module load intel openmpi

module load ncview nco netcdf/4.1.3-fasrc01

export ARCH=`uname -s`

export COMPILER=ifort

export NETCDF\_HOME=`nc-config --prefix`

export GC\_BIN="$NETCDF\_HOME/bin"

export GC\_INCLUDE="$NETCDF\_HOME/include"

export GC\_LIB="$NETCDF\_HOME/lib"

export ESMF\_COMPILER=intelgcc

export ESMF\_COMM=openmpi

export ESMF\_BOPT=O

discover.nccs.nasa.gov: Use these settings in your .cshrc file:

source /usr/share/modules/init/csh

module purge

module load comp/intel-13.1.2.183

module load other/comp/gcc-4.6.3-sp1

module load other/mpi/openmpi/1.7.2-intel-13.1.2.183

module load lib/mkl-13.0.1.117

module load other/SIVO-PyD/spd\_1.10.0\_gcc-4.6.3-sp1

setenv ARCH `uname –s`

setenv COMPILER ifort

setenv GC\_BIN /gpfsm/dnb32/mslong1/tools/netcdf/bin/

setenv GC\_INCLUDE /gpfsm/dnb32/mslong1/tools/netcdf/include/

setenv GC\_LIB /gpfsm/dnb32/mslong1/tools/netcdf/lib

setenv ESMF\_COMPILER intelgcc

setenv ESMF\_COMM openmpi

setenv ESMF\_BOPT O

Feel free to add settings for other systems here.