Getting Started with GCHP v11-02c

Lizzie Lundgren
GEOS-Chem Support Team
geos-chem-support@as.harvard.edu

September 2017

Overview

1) What is GCHP and why use it?
2) Common Misconceptions
3) Useful Tips
4) Basic Tutorial
5) Introduction to GCHP Source Code
6) Resources

What is GCHP and why use it?

- GCHP features the same science as GEOS-Chem using the standard "classic" capability except:
  1. It operates on a cubed-sphere grid
  2. It is parallelized using a message-passing interface (MPI)
- GCHP improves upon GCC by:
  - Enabling more accurate transport
  - Providing efficient scaling making finer resolution global simulations possible

Common Misconceptions about GCHP

- I need a high performance compute cluster for GCHP
  - Not true! You can run GCHP on as little as one machine with 6 cores. We are working towards eventually being able to run it on a single core.

- I can only perform high-resolution runs with GCHP
  - Not true! GCHP can run at c24 resolution which is the cubed-sphere equivalent of 4°x5°.

- I need met fields at the same resolution as my run
  - Not true! You may use GCHP with 2°x2.5° meteorology for up to at least c180 (0.5° res), although we recommend keeping the met resolution to no more than twice your run resolution equivalent to ensure quality output. If the met wind fields are too coarse relative to your internal resolution then polar divergence will occur.

Useful Tip #1: Grid Resolutions

- Cubed-sphere resolution “cN” means each of the six faces are divided into N x N segments.
- An easy rule-of-thumb for resolution mapping is to divide 90 by N to determine the approximate lat-lon degree resolution.

Standard lat-lon resolution	Approximate CS equivalent(s)
4° x 5°	c24
2° x 2.5°	c48
1° x 1.25°	c90
0.5° x 0.625° [1]	c180
0.25° x 0.3125° [2]	c360
0.125° x 0.15625°	c720 [3]

[1] Native resolution of MERRA-2 product from GMAO
[2] Native resolution of GEOS-FP product from GMAO
[3] Native cubed-sphere resolution of GEOS-5

Useful Tip #2: Resource Allocation

- Number of nodes and number of faces are independent
- Number of cores and number of faces are NOT independent
- Total number of cores must be divisible by 6!
- How does it work?
  - Each NxN face is divided into NX x NY/6 segments, each comprised of approximately N/NX x N*6/NY cubed-sphere grid cells.
  - Each segment is processed by a single core
  - NX * NY must therefore equal the total number of cores
  - NX * NY/6 would ideally be square to minimize required communication between cores
- NX and NY are manually set in config file GCHP.rc but are over-written by NX and NY of your choosing in utility script runConfig.rc. Setting them will soon be automatic.
- More on this topic later in the presentation.

GCHP Tutorial

1. Downloading Source Code
2. Create a Run Directory
3. Set Link to Source Code
4. Load Environment
5. Compile GCHP
6. Configure Run Settings
7. Run a Simulation
8. Analyze Output Data
9. Reuse a Run Directory

Step 1: Download Source Code

- You need two repositories for GCHP:
  1. GEOS-Chem “classic” (GCC) code as your main directory
  2. GCHP code as a subdirectory within it

- Use the GC and GCHP master branches and checkout the version tag for v11-02c

```
git clone -b master https://bitbucket.org/gcst/gc_bleeding_edge Code.v11-02c_gchp
cd Code.v11-02_gchp
git checkout tags/v11-02c
git clone -b master https://bitbucket.org/gcst/gchp GCHP
cd GCHP
git checkout tags/v11-02c-1yr-bm
```

Step 2: Create a Run Directory

- Download a GCHP run directory from GEOS-Chem Unit Tester
- Use the UT_Bleeding_Edge master branch
  - If you have the repository already:
    - `git checkout master`
    - `git pull`
    - `git checkout tags/v11-02c`
  - If you do not have the repository:
    - `git clone -b master https:/bitbucket.org/gcst/ut_bleeding_edge UT`
- To create run directory, modify UT/perl/CopyRunDirs.input:

```
#
# %%% Target directory and copy command %%%
#
   COPY_PATH       : {HOME}/GC/rundirs
   COPY_CMD        : cp -rfL
#
```

Set target directory

Set start and end dates

```
## ======= GCHP ==========================================================
# gchp     -          -      benchmark      2013070100    2013080100     -
# gchp     -          -      standard       2013070100    2013070101     -
# gchp     -          -      RnPbBe         2013070100    2013070101     -
```

Uncomment simulation of interest

- Like GCC, each simulation has a different run directory
- Unlike GCC, simulation resolution is configurable at run-time

Available GCHP Run Directories

- Like GCC, each simulation has a different run directory
- Unlike GCC, simulation resolution is configurable at run-time
- Run directories available in v11-02c:
  1. Standard simulation
     - Use for full chemistry science simulations
     - Includes symbolic links to restart files for four resolutions: c24, c48, c90, c180
     - Restart files incorporate UCX initial mixing ratios applied internally in GCC
  2. Rn-Pb-Be7 simulation
     - Use for Rn-Pb-Be7 and passive tracer simulations
     - Includes symbolic links to restart files for four resolutions: c24, c48, c90, c180
  3. Benchmark simulation
     - Used for benchmarking simulations
     - Like the new GCC benchmark simulation, both simple and complex SOA are on
     - Includes symbolic links to c24 restart file only
     - Restart files does NOT incorporate UCX initial mixing ratios applied internally in GCC

GCHP Standard Run Directory: Contents

- Configuration files
- Chemistry `*.dat` files
- Sample environment files subdirectory
- Sample run scripts subdirectory
- Utility bash scripts
- Output data subdirectory
- README
- Makefile
- Symbolic links to:
  - Restart files
  - Input data directories
  - Cube sphere <-> lat-lon regridding files (tile files)

```
OD ~/gchp_standard $ ls
bashrcSamples/     h2so4.dat                                MetDir@
brc.dat            HEMCO_Config.rc                          org.dat
build.sh*          HISTORY.rc                               OutputDir/
CAP.rc             initial_GEOSChem_rst.c180_standard.nc@   README
ChemDataDir@       initial_GEOSChem_rst.c24_standard.nc@    runConfig.sh*
dust.dat           initial_GEOSChem_rst.c48_standard.nc@    runScriptSamples/
ExtData.rc         initial_GEOSChem_rst.c90_standard.nc@    setCodeDir*
FJX_j2j.dat        input.geos                               so4.dat
FJX_spec.dat       input.nml                                soot.dat
fvcore_layout.rc   jv_spec_mie.dat                          ssa.dat
GCHP.rc            MainDataDir@                             ssc.dat
gchp.run*          Makefile*                                TileFiles@
```

GCHP Run Directory: Environment Setup Files

- Sample environment setup files are stored in the **`bashrcSamples`** subdirectory

```
OD ~/gchp_standard $ cd bashrcSamples/
OD ~/gchp_standard/bashrcSamples $ ls
gchp.gfortran_MVAPICH2.bashrc
gchp.ifort13_openmpi_glooscap.bashrc
gchp.ifort15_mvapich2_odyssey.bashrc
```

- The sample files provided are custom for Harvard and Dalhousie Universities
- For your own system you may use these files as examples to build your own
- Please note that the gfortran compiler and the MVAPICH2 and OpenMPI implementations are all open-source
- Contact the GCST if you and your system administrators get stuck

GCHP Run Directory: Config Files

- GCHP Specific:
  - **`Cap.rc`**
    - start/end dates, and more
  - **`ExtData.rc`**
    - external data information
  - **`fvcore_layout.rc`**
    - transport-related settings
  - **`GCHP.rc`**
    - general settings
  - **`HISTORY.rc`**
    - output data settings

- Same as GCC:
  - **`input.geos`**
  - **`HEMCO_Config.rc`**

- **WARNINGS:**
  - Not all fields in **`input.geos`** and **`HEMCO_Config.rc`** are used
  - Some settings must be set in multiple files (**use runConfig.sh**)

```
OD ~/gchp_standard $ ls
bashrcSamples/     h2so4.dat                                  MetDir@
brc.dat            HEMCO_Config.rc                            org.dat
build.sh*          HISTORY.rc                                 OutputDir/
CAP.rc             initial_GEOSChem_rst.c180_standard.nc@     README
ChemDataDir@       initial_GEOSChem_rst.c24_standard.nc@      runConfig.sh*
dust.dat           initial_GEOSChem_rst.c48_standard.nc@      runScriptSamples/
ExtData.rc         initial_GEOSChem_rst.c90_standard.nc@      setCodeDir*
FJX_j2j.dat        input.geos                                 so4.dat
FJX_spec.dat       input.nml                                  soot.dat
fvcore_layout.rc   jv_spec_mie.dat                            ssa.dat
GCHP.rc            MainDataDir@                               ssc.dat
gchp.run*          Makefile*                                  TileFiles@
```

GCHP Run Directory: Run Scripts and Output Data

- **`OutputDir/`**
  - All GCHP output data configured in **`HISTORY.rc`** are saved here
  - Restart file is not saved here
  - Do not remove or rename! GCHP will hang without an error message

- **`runScriptSamples/`**
  - For Odyssey (Harvard):
    - **`GCHP_slurm.run`**
  - For Glooscap (Dalhousie):
    - **`GCHP_gridengine.run`**
  - For other Systems
    - Use these as examples to build your own

```
OD ~/gchp_standard $ ls
bashrcSamples/     h2so4.dat                                MetDir@
brc.dat            HEMCO_Config.rc                          org.dat
build.sh*          HISTORY.rc                               OutputDir/
CAP.rc             initial_GEOSChem_rst.c180_standard.nc@   README
ChemDataDir@       initial_GEOSChem_rst.c24_standard.nc@    runConfig.sh*
dust.dat           initial_GEOSChem_rst.c48_standard.nc@    runScriptSamples/
ExtData.rc         initial_GEOSChem_rst.c90_standard.nc@    setCodeDir*
FJX_j2j.dat        input.geos                               so4.dat
FJX_spec.dat       input.nml                                soot.dat
fvcore_layout.rc   jv_spec_mie.dat                          ssa.dat
GCHP.rc            MainDataDir@                             ssc.dat
gchp.run*          Makefile*                                TileFiles@
```

GCHP Run Directory: Utility Scripts

- **`setCodeDir`**
  - creates symbolic link to source code path
  - Pass full path (without links) as an argument
- **`build.sh`**
  - cleans and compiles code
  - executed in Makefile
- **`runConfig.sh`**
  - single location to update common run settings
  - overwrites config files
  - executed in run scripts

> 90% of GCHP errors are due to incorrect or inconsistent config file settings. Use bash script **`runConfig.sh`** to avoid common errors.

```
OD ~/gchp_standard $ ls
bashrcSamples/      h2so4.dat                                MetDir@
brc.dat             HEMCO_Config.rc                          org.dat
build.sh*           HISTORY.rc                               OutputDir/
CAP.rc              initial_GEOSChem_rst.c180_standard.nc@   README
ChemDataDir@        initial_GEOSChem_rst.c24_standard.nc@    runConfig.sh*
dust.dat            initial_GEOSChem_rst.c48_standard.nc@    runScriptSamples/
ExtData.rc          initial_GEOSChem_rst.c90_standard.nc@    setCodeDir*
FJX_j2j.dat         input.geos                               so4.dat
FJX_spec.dat        input.nml                                soot.dat
fvcore_layout.rc    jv_spec_mie.dat                          ssa.dat
GCHP.rc             MainDataDir@                             ssc.dat
gchp.run*           Makefile*                                TileFiles@
```

Step 3: Set Link to Source Code

- GCHP uses a symbolic link to source code called **`CodeDir`**
- Run bash shell script **`setCodeDir`** to set symbolic link:

```
OD ~/gchp_standard $ ./setCodeDir /n/home08/elundgren/Code.v11-02c_gchp/
CodeDir: symbolic link to `/n/home08/elundgren/Code.v11-02c_gchp/'
```

- Things to note:
  - Specify the path to the GEOS-Chem top-level directory, not the GCHP subdirectory
  - Do not include symbolic links in your source code path
  - Unlike GCC, do not edit the Makefile with your source code path

Step 4: Load GCHP Environment

- Set up your environment prior to compiling and/or running
- On Odyssey:

```
OD ~ $ source GCHP.ifort15_mvapich2_odyssey.bashrc
Loading modules for GCHP on Odyssey, please wait ...

Due to MODULEPATH changes the following have been reloaded:
  1) gd/2.0.28-fasrc01


Currently Loaded Modules:
  1) perl/5.10.1-fasrc04            4) intel/15.0.0-fasrc01    7) zlib/1.2.8-fasrc03
  2) perl-modules/5.10.1-fasrc12    5) gd/2.0.28-fasrc01       8) hdf5/1.8.12-fasrc12
  3) git/2.1.0-fasrc02              6) mvapich2/2.2-fasrc01    9) netcdf/4.1.3-fasrc09
```

- Elsewhere:
  - Create a **`.bashrc`** file based on sample files in the run directory
  - Using the libraries above is recommended but other combos are possible
    - OpenMPI
    - Intel MPI
    - Gfortran
    - Other NetCDF library versions

Step 5: Compile GCHP

- Like GCC, compile GCHP from the run directory using the Makefile

- First time compilation (30-60 min): **`make clean_compile`**
  - Warnings, error messages, and pauses are normal
  - Signs of successful compilation:
    - **`"### GCHP compiled Successfully ###"`**
    - The following files exist:
      - **`GCHP/ESMF/esmf.install`**
      - **`GCHP/FVdycoreCubed_GridComp/fvdycore.install`**
      - **`GCHP/Shared/mapl.install`**

- Subsequent compilation: **`make clean_standard`**
  - For updates to GC base code or GCHP top-level directory
  - Not for updates to GCHP subdirectories (e.g. GCHP/Shared)

Step 6: Configure Run Settings

- Use utility bash script **`runConfig.sh`** for select config settings:
  - Compute resources (e.g. # nodes and cores)
  - Internal grid resolution
  - Restart file
  - Simulation start/end times
  - Output diagnostic file frequency, duration, and mode
    - e.g. hourly (frequency) that is time-averaged (mode) and contained in daily files (duration)
  - Component on/off switches, including mixing scheme
  - Time-step intervals
  - Debug level for MAPL
- Manually change individual config files for all other settings
- Important things to understand about **`runConfig.sh`**
  - Run scripts execute **`runConfig.sh`** prior to executing **`geos`**
  - It overwrites **`input.geos`** and **`*.rc`** files (BEWARE!!!)
  - Does not updates **`HEMCO_Config.rc`** or **`Ext_Data.rc`** (yet)

runConfig.sh: Default Settings Part 1

```
#!/bin/bash

# runSettings.sh: Update select settings in *.rc and input.geos config files
#
# Usage: ./runSettings.sh
#
# E. Lundgren, 8/17/2017

##########################
####  Configurables   ####
##########################

#### COMPUTE RESOURCES
NUM_NODES=1
NUM_CORES_PER_NODE=6
NY=6                    # NY must be an integer and a multiple of 6
NX=1                    # NX*NY must equal total number of cores
                        # Choose NX and NY to optimize NX x NY/6 squareness
                        # within contraint of total # of CPUs
                        # e.g., (NX=2,NY=12) if 24 cores, (NX=4,NY=12) if 48

#### INPUT MET RESOLUTION
INPUT_MET_RES=placeholder # not yet implemented

### INTERNAL CUBED-SPHERE RESOLUTION
# Please note that lightning and dust emissions in GCHP are tuned to c24.
# Removing the online resolution dependency of lightning and dust by using
# offline lightning and dust emissions will be a future update. Until
# then, running at resolutions other than c24 will have degraded accuracy
# due to the resolution dependency of emissions. Please contact the GCST
# for more information.
CUBE_SPHERE_RES=24      # 24~4x5, 48~2x2.5, 90~1x1.25, 180~0.5x0.625

### INITIAL RESTART FILE
INITIAL_RESTART=default  # specify default or your own restart filename
```

See Useful Tip #2 at start of slides. Always check that your resources here match your run script settings!

Set simulation resolution

Set restart filename if not using symbolic links included in the run directory

runConfig.sh: Default Settings Part 2

```
#### SIMULATION TIMES
Start_Time="20130701 000000"
End_Time="20130701 010000"
Duration="00000000 010000"

#### OUTPUT
cs_frequency="010000"
cs_duration="010000"
cs_mode="'time-averaged'"
ll_frequency="010000"
ll_duration="010000"
ll_mode="'time-averaged'"

#### TURN COMPONENTS ON/OFF
Turn_on_Chemistry=T
Turn_on_emissions=T
Turn_on_Dry_Deposition=T
Turn_on_Wet_Deposition=T
Turn_on_Transport=T
Turn_on_Cloud_Conv=T
Turn_on_PBL_Mixing=T

#### TIMESTEPS
TransConv_Timestep_min=10
ChemEmis_Timestep_min=20

#### PBL MIXING
Use_nonlocal_PBL=T

#### DEBUG OPTIONS
MAPL_DEBUG_LEVEL=0     # 0 is none, output increases with higher values (to 20)
#GC_ND70="0 all"       # requires special handling; omit for now
```

Set simulation start and end times. Set duration to the difference, or multiples of the difference if doing segmented runs.

Set cubed-sphere file duration, data frequency, and mode, either "'time-averaged'" or "'instantaneous'". These update the "center" collection in HISTORY.rc. Ignore the "ll_" options.

Set high-level input.geos options

For MAPL debugging, set level as high as 20. Be aware this will come at great performance cost!

Step 7: Run GCHP (single node)

- Two run scripts are provided as examples of how to run GCHP:

```
OD ~/gchp_standard $ cd runScriptSamples/
OD ~/gchp_standard/runScriptSamples $ ls
gchp_gridengine.run*  gchp_slurm.run*
```

- Both run **`runConfig.sh`** prior to executing **`geos`**
- **`runConfig.sh`** will exit with an error if your settings do not make sense or if your restart file does not exist, thereby preventing GCHP from starting and hanging
- It is up to you, however, to check that your compute resource settings in your run script match those in **`runConfig.rc`**
- GCHP standard output is sent to a log called **`gchp.log`**
- You may change the log name in your run script.

Step 8: Analyze Output

- All GCHP output is in netCDF-4 format
- Two outputs:
  - Restart file
    - Stored in top-level of run directory
    - Filename: **`-gcchem_internal_checkpoint_c24.nc`** (configured in **`GCHP.rc`**)
    - Cubed-sphere grid
  - One or more diagnostic files
    - Output files are stored in **`OutputDir`**
    - File format: **`GCHP.{collection}.YYYYMMDD.nc4`**
    - Different files for each collection
    - Collections configured in **`HISTORY.rc`**
- Regrid from cubed-sphere to lat-lon using tools developed by Seb Eastham:
  - CSRegridTool (FORTRAN): https://bitbucket.org/sdeastham/csregridtool
  - CSRegrid (Matlab): https://bitbucket.org/gcst/csgrid

Step 9: Reuse a Run Directory

- You can reuse your GCHP run directory but MUST do one of the following prior to rerunning to avoid a seg fault:
  - **`make cleanup_output`**
  - Delete file **`cap_restart`**
- Invoking **`cleanup_output`** will remove all simulation log and output files but will not delete your executable or last build info log
- Sample run scripts include several aliases to reduce typing if running repeatedly
- Experiment with different run settings in **`runConfig.sh`**
- If changing # of cores and/or # of nodes:
  - Remember to update **`runConfig.sh`** as well as your run script
  - Choose NX and NY such that NX by NY/6 is roughly square
  - See next slide for an example

GCHP with Multiple Nodes

- 24 cores across 2 nodes
  - Run script:

```
#!/bin/bash

#SBATCH -n 24
#SBATCH -N 2
```

  - runConfig.sh:

```
#### COMPUTE RESOURCES
NUM_NODES=2
NUM_CORES_PER_NODE=12
NY=12                    # NY must be an integer and a multiple of 6
NX=2                     # NX*NY must equal total number of cores
                         # Choose NX and NY to optimize NX x NY/6 squareness
                         # within contraint of total # of CPUs
                         # e.g., (NX=2,NY=12) if 24 cores, (NX=4,NY=12) if 48
```

- 48 cores across 3 nodes
  - Run script:
    - –n 48
    - –N 3
  - runConfig.sh:
    - NUM_NODES=3
    - NUM_CORES_PER_NODE=16
    - NY=12
    - NX=4

GCHP Source Code: ESMF, MAPL, FVdycore

ESMF and transport directories: these are compiled once and then you shouldn't need to touch them

```
OD ~/Code.v11-02/GCHP $ ls
Chem_GridCompMod.F90        GIGC_Connections.H              HEMCO_Includes_BeforeRun.H
ESMF/                       gigc_diagnostics_mod.F90        Includes_After_Dyn.H
FVdycoreCubed_GridComp/     gigc_finalization_mod.F90       Includes_After_Run.H
gchp_utils.F90              GIGC_GridCompMod.F90            Includes_Before_Dyn.H
gc_land_interface.F90       gigc_initialization_mod.F90     Includes_Before_Run.H
GCSA-HowTo.docx             GIGC.mk                         Makefile
GEOSChem.F90                gigc_mpi_wrap.F90               Registry/
GEOS_ctmEnvGridComp.F90     gigc_test_utils.F90             Shared/
GEOS_HyCoords.H             gigc_type_mod.F*
gigc_chunk_mod.F90          HEMCO_Includes_AfterRun.H
```

This is also compiled once. Most run directory issue errors will point you here.

```
OD ~ $ cd Code.v11-02/GCHP/Shared/
OD ~/Code.v11-02/GCHP/Shared $ ls
Chem_Base/     CVS/          GFDL_fms/    GMAO_hermes/    GMAO_pilgrim/   MAPL_cfio/
Chem_Shared/   GEOS_Shared/  GMAO_etc/    GMAO_mpeu/      GNUmakefile*
Config/        GEOS_Util/    GMAO_gfio/   GMAO_perllib/   MAPL_Base/
```

Especially here.

GCHP Source Code: MAPL_Base

Error messages may lead you here…

Resource setup or time issues

Input data issues

Output data issues

Tile file issues (lat-lon <-> CS)

```
OD ~/Code.v11-02/GCHP/Shared/MAPL_Base $ ls
allgather.H                 MAPL_Cap.F90                MAPL_SatVapor.F90
allgatherv.H                MAPL_CFIO.F90               MAPL_ShmemMod.F90*
allreducemax.H*             MAPL_CFIOServer.F90         MAPL_SimpleBundleMod.F90
allreducemin.H*             MAPL_Comms.P90              MAPL_Sort.F90
allreducesum.H*             MAPL_Constants.F90          mapl_stub.pl*
arraygather.H               MAPL_ErrLog.h               MAPL_stubs.F90
arraygatherRcvCnt.H         MAPL_ErrLogMain.h           MAPL_sun_uc.P90
arrayscatter.H              MAPL_Exceptions.h           mapl_tree.py
arrayscatterRcvCnt.H        MAPL_ExtDataGridCompMod.F90 MAPL_VarSpecMod.F90
bcast.H                     MAPL_GenericCpiComp.F90     mapl_vlist.py*
c_mapl_locstream_F.c        MAPL_Generic.F90            memuse.c*
CubeToLatLon.F90            MAPL_Generic.h              overload.macro*
CVS/                        MAPL_Hash.F90               Python/
eqsat.H                     MAPL_HeapMod.F90            qsatice.H
eqsat_verification.dat      MAPL_HistoryGridComp.F90    qsatlqu.H
esatice.H                   MAPL_HorzTransform.F90      read_parallel.H
esatlqu.H                   MAPL_InterpMod.F90          recv.H
ESMFL_Mod.P90               MAPL_IO.P90                 red_ma.pl*
gather.H                    MAPL_LoadBalance.F90        Sample_ExtData.rc
GetPointer.H                MAPL_LocStreamMod.F90       scatter.H
getrss.c                    MAPL_MaxMinMod.F90          send.H
GNUmakefile*                MAPL_MemUtils.F90*          sendrecv.H
hash.c                      MAPL_Mod.F90                sort.c
hinterp.F                   MAPL_NewArthParser.F90*     sun.H
HorzBinning.F90             MAPL_NominalOrbitsMod.F90   tests/
mapl_acg.pl*                MAPL_OrbGridCompMod.F90     TeX/
MAPL_Base.F90               MAPL_OrbGridComp.rc         tstqsat.F90
MAPL_base.mk*               MAPL_Profiler.F90           write_parallel.H
```

Review your run directory setup before trying to change MAPL code!

Resources

- GCHP Links:
  - Main Wiki Page
  - Online Tutorial
  - v11-02: new features, benchmarks, open and resolved issues
  - Working Group and Users
  - Timing Tests
- Other Useful Links:
  - Interactive construction of a cubed-sphere grid
  - GMAO MAPL User's Guide (info may be outdated)
  - GEOS-5 wiki page for ExtData (info may be outdated)