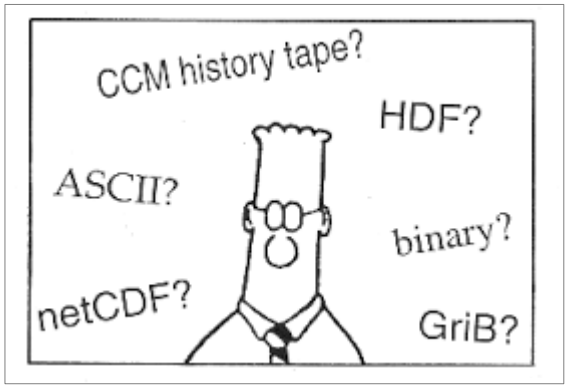
An introduction to netCDF diagnostics in GEOS-Chem



Bob Yantosca
Senior Software Engineer
05 Oct 2017

Overview

- What are diagnostics?
- Why do we need new diagnostics?
- Design considerations
- Building blocks: Fields and Collections
- Examples: Scheduling diagnostic output
- Near-future work

What are diagnostics?

Types of GEOS-Chem outputs

- GEOS-Chem (GC) creates several output files:
  - Restart files
    - For GEOS-Chem species
    - For HEMCO emissions quantities
  - Diagnostic output files
    - Time-averaged
    - Timeseries (several options)
    - “Plane-following”
  - See *GEOS-Chem Output Files* page on the wiki
    - GC wiki pages are located at wiki.geos-chem.org

Restart files

- GEOS-Chem restart files
  - **`GEOSChem_restart.YYYYMMDDhhmm.nc`**
  - Archive species concentrations (mol/mol dry air)
  - These files are used to start the next stage of a long GEOS-Chem simulation (which has to be split up)
- HEMCO restart files (for emissions)
  - **`HEMCO_restart.YYYYMMDDhhmm.nc`**
  - Archive several quantities for the MEGAN biogenic emissions and soil NOx emissions

Diagnostics

- Diagnostics are outputs from GC that represent various physical quantities
- We use this output to assess how well GC is representing the atmosphere (e.g. “benchmarks”)
- GC diagnostic output includes:
  - Species concentrations
  - Chemical reaction rates (including photolysis rates)
  - Transport, convective, PBL mixing fluxes
  - Loss of species by drydep and wetdep, etc.

Time-averaged diagnostics

- Most diagnostics in GC are time-averaged
  - e.g. **`trac_avg.geosfp_4x5_standard.YYYYMMDDhhmm`**
  - Set in the "DIAGNOSTIC MENU" section of `input.geos`
  - Minimum averaging period = 1 day
  - Can schedule output for any day of the year
  - More info on *GEOS-Chem Input Files* page on wiki

Timeseries diagnostics

- Can save output more frequently than 1 day
  - Several different options:
    - ND40: “Plane-following” timeseries
    - ND48: Instantaneous output at individual points
      - NOTE: Implementation has been problematic
    - ND49: Instantaneous lat-lon diagnostic output
    - ND50: 24-hour time-averaged output
    - ND51: “Satellite” timeseries
      - Facilitates comparison with sun-synchronous satellite data, which overpass a specific spot on the globe at the same local time each orbital period

Current diagnostic settings in input.geos file

```
------------------------+-----------------------------------------------------
%%% DIAGNOSTIC MENU %%% :
Binary punch file name  : trac_avg.merra2_4x5_standard.YYYYMMDDhhmm
Diagnostic Entries ---> :  L   Tracers to print out for each diagnostic
ND01: Rn/Pb/Be source   :  0   all
ND02: Rn/Pb/Be decay    :  0   all
ND03: Hg emissions, P/L :  0   all
ND04: CO2 Sources       :  0   all
ND05: Sulfate prod/loss : 72   all
ND06: Dust aer source   :  1   all
ND07: Carbon aer source : 72   all
ND08: Seasalt aer source:  1   all
ND09: -                 :  0   all
ND10: -                 :  0   all
ND11: Acetone sources   :  1   all
ND12: BL fraction       :  0   all
ND13: Sulfur sources    : 72   all
ND14: Cld conv mass flx :  0   all
ND15: BL mix mass flx   :  0   all
ND16: LS/Conv prec frac :  0   all
ND17: Rainout fraction  :  0   all
ND18: Washout fraction  :  0   all
ND19: CH4 loss          :  0   all
ND21: Optical depths    : 72   all
ND22: J-Values          : 72   all
      => JV time range  :      11 13
ND24: E/W transpt flx   :  0   all
ND25: N/S transpt flx   :  0   all
ND26: U/D transpt flx   :  0   all
ND27: Strat NOx,Ox,HNO3 :  0   1 2 7
ND28: Biomass emissions : 72   all
ND29: CO sources        : 72   all
ND30: Land Map          :  0   all
ND31: Pressure edges    : 73   all
ND32: NOx sources       : 72   all
ND33: Column tracer     :  0   all
ND34: Biofuel emissions :  1   all
ND35: Tracers at 500 mb :  0   all
ND36: Anthro emissions  : 72   all

...etc...
```

```
%%% OUTPUT MENU %%%     : 123456789.123456789.123456789.1--1=ZERO+2=BPCH
Schedule output for JAN : 300000000000000000000000000000
Schedule output for FEB : 300000000000000000000000000000
Schedule output for MAR : 300000000000000000000000000000
Schedule output for APR : 300000000000000000000000000000
Schedule output for MAY : 300000000000000000000000000000
Schedule output for JUN : 300000000000000000000000000000
Schedule output for JUL : 300000000000000000000000000000
Schedule output for AUG : 300000000000000000000000000000
Schedule output for SEP : 300000000000000000000000000000
Schedule output for OCT : 300000000000000000000000000000
Schedule output for NOV : 300000000000000000000000000000
Schedule output for DEC : 300000000000000000000000000000

%%% ND49 MENU %%%       :
Turn on ND49 diagnostic : F
Inst 3-D timeser. file  : tsYYYYMMDD.bpch
Tracers to include      : 94
Frequency [min]         : 120
IMIN, IMAX of region    :  70  30
JMIN, JMAX of region    :  23  46
LMIN, LMAX of region    :   1   1

%%% ND50 MENU %%%       :
Turn on ND50 diagnostic : F
24-hr avg timeser. file : ts_24h_avg.YYYYMMDD.bpch
Output as HDF5?         : F
Tracers to include      : 82 83 84 85 86 87
IMIN, IMAX of region    :   1  72
JMIN, JMAX of region    :   1  46
LMIN, LMAX of region    :   1   1
```

http://wiki.geos-chem.org/GEOS-Chem_Input_Files#The_input.geos_file

Why do we need new diagnostics?

Reason to replace GC diagnostics #1

- “NDxx” diagnostic structure is historical baggage!
  - Taken from the old 9-layer Harvard-CTM (1980's-90's)
  - Diags were implemented in an “ad-hoc” fashion
  - Diagnostic arrays and counters are scattered haphazardly throughout the code
    - diag_mod.F, ndxx_setup_mod.F, initialize.F, etc.
  - Timeseries diagnostics were an afterthought
    - And also haphazardly implemented

Reason to replace GC diagnostics #2

- Some issues with "binary punch" (bpch) format
  - Bpch has been the format for GC diagnostic outputs for the past 20 years
  - But, bpch files only contain data, but limited metadata
    - Metadata in separate "diaginfo.dat", "tracerinfo.dat" files
  - And, bpch requires GAMAP for visualization
    - GAMAP requires Interactive Data Language (IDL)
    - IDL requires $$$
    - Proprietary software like IDL is a barrier to cloud computing

Reason to replace GC diagnostics #3

- We are developing a capability for GC to take advantage of High-Performance Computing environments (named “GCHP”)
- Bpch data I/O cannot be efficiently done in High-Performance Computing environments
  - Bpch (which is a sequential, unformatted stream of bytes) has to be written in one go, from start to end, on a single CPU (unlike netCDF)

Design considerations

Design consideration #1

- Use NetCDF file format
  - NetCDF is a set of software libraries and machine-independent data formats that promote the sharing of array-oriented scientific data.
    - Long story short, it's a “Self-describing file format”
  - NetCDF stores data arrays and related “metadata” (i.e. descriptions about the data) in the same file.
  - Data in netCDF files can be compressed to minimize file storage requirements.

```
netcdf AEIC.47L.gen.1x1 {
dimensions:
        lon = 360 ;
        lat = 180 ;
        lev = 47 ;
        time = UNLIMITED ; // (12 currently)
variables:
        float lon(lon) ;
                lon:units = "degrees_east" ;
        float lat(lat) ;
                lat:units = "degrees_north" ;
        float lev(lev) ;
                lev:positive = "up" ;
                lev:long_name = "GEOS-Chem level" ;
                lev:units = "level" ;
        double time(time) ;
                time:units = "days since 2005-01-01 00:00:00" ;
                time:calendar = "standard" ;
        float FUELBURN(time, lev, lat, lon) ;
                FUELBURN:units = "kg/m2/s" ;
                FUELBURN:long_name = "AEIC aircraft fuel burned" ;
        float NO2(time, lev, lat, lon) ;
                NO2:units = "kg/m2/s" ;
                NO2:long_name = "AEIC aircraft emitted NO2" ;
        ... etc ...

// global attributes:
                :description = "AEIC emissions. Regridded from original AEIC levels onto
                 standard GEOS-Chem levels using routine `aeic_vertgrid.py`." ;
                :history = "Created by Christoph Keller, Wed Jan 28 13:10:01 2015" ;

Data:
 lon = -179.5, -178.5, -177.5, -176.5, -175.5, -174.5, -173.5, -172.5, ... 179.5
 lat = -89.5, -88.5, -87.5, -86.5, -85.5, -84.5, -83.5, -82.5, -81.5, -80.5, ... 89.5
 lev = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ... 47
 time = "2005-01-01", "2005-02-01", "2005-03-01", "2005-04-01", "2005-05-01", "2005-06-01",
        "2005-07-01", "2005-08-01", "2005-09-01", "2005-10-01", "2005-11-01", "2005-12-01"
```

Typical netCDF file structure (using COARDS conventions)

Dimensions are in red.

Index arrays (or axis arrays) are in blue. These specify the grid.

Data arrays are in green.

Attributes (i.e. descriptive text about the data and/or file) are in magenta.

This output was generated with:
`ncdump -ct AEIC.47L.gen.1x1.nc`

Design consideration #2

- Many tools have been developed for netCDF
  - NetCDF operators (NCO)
  - Climate data operators (CDO)
  - Ncview
  - Panoply
  - Several Python modules (see wiki)
    - Python has been gaining in popularity for sci. computing
  - GCPy (in development)
  - Matlab also has netCDF capability

Design consideration #3

- We wanted to simplify the input file that is used to schedule diagnostic outputs
  - GC diagnostic input options will be read a new file called **`HISTORY.rc`** instead of from **`input.geos`**
    - We “stole” HISTORY.rc from GCHP!
  - This will allow us to define all diagnostics (either instantaneous or time-averaged) in the same way, in the same file

Building blocks of the new GEOS-Chem diagnostics: Fields and Collections

```
------------------------+-----------------------------------------------------
%%% DIAGNOSTIC MENU %%% :
Binary punch file name  : trac_avg.merra2_4x5_standard.YYYYMMDDhhmm
Diagnostic Entries ---> :  L   Tracers to print out for each diagnostic
ND01: Rn/Pb/Be source   :  0   all
ND02: Rn/Pb/Be decay    :  0   all
ND03: Hg emissions, P/L :  0   all
ND04: CO2 Sources       :  0   all
ND05: Sulfate prod/loss : 72   all
ND06: Dust aer source   :  1   all
ND07: Carbon aer source : 72   all
ND08: Seasalt aer source:  1   all
ND09: -                 :  0   all
ND10: -                 :  0   all
ND11: Acetone sources   :  1   all
ND12: BL fraction       :  0   all
ND13: Sulfur sources    : 72   all
ND14: Cld conv mass flx :  0   all
ND15: BL mix mass flx   :  0   all
ND16: LS/Conv prec frac :  0   all
ND17: Rainout fraction  :  0   all
ND18: Washout fraction  :  0   all
ND19: CH4 loss          :  0   all
ND21: Optical depths    : 72   all
ND22: J-Values          : 72   all
      => JV time range  :      11 13
ND24: E/W transpt flx   :  0   all
ND25: N/S transpt flx   :  0   all
ND26: U/D transpt flx   :  0   all
ND27: Strat NOx,Ox,HNO3 :  0   1 2 7
ND28: Biomass emissions : 72   all
ND29: CO sources        : 72   all
ND30: Land Map          :  0   all
ND31: Pressure edges    : 73   all
ND32: NOx sources       : 72   all
ND33: Column tracer     :  0   all
ND34: Biofuel emissions :  1   all
ND35: Tracers at 500 mb :  0   all
ND36: Anthro emissions  : 72   all

...etc...
```

We are removing the diagnostic inputs from the `input.geos` file!

```
%%% OUTPUT MENU %%%     : 123456789.123456789.123456789.1--1=ZERO+2=BPCH
Schedule output for JAN : 300000000000000000000000000000
Schedule output for FEB : 300000000000000000000000000000
Schedule output for MAR : 300000000000000000000000000000
Schedule output for APR : 300000000000000000000000000000
Schedule output for MAY : 300000000000000000000000000000
Schedule output for JUN : 300000000000000000000000000000
Schedule output for JUL : 300000000000000000000000000000
Schedule output for AUG : 300000000000000000000000000000
Schedule output for SEP : 300000000000000000000000000000
Schedule output for OCT : 300000000000000000000000000000
Schedule output for NOV : 300000000000000000000000000000
Schedule output for DEC : 300000000000000000000000000000

%%% ND49 MENU %%%       :
Turn on ND49 diagnostic : F
Inst 3-D timeser. file  : tsYYYYMMDD.bpch
Tracers to include      : 94
Frequency [min]         : 120
IMIN, IMAX of region    :  70  30
JMIN, JMAX of region    :  23  46
LMIN, LMAX of region    :   1   1

%%% ND50 MENU %%%       :
Turn on ND50 diagnostic : F
24-hr avg timeser. file : ts_24h_avg.YYYYMMDD.bpch
Output as HDF5?         : F
Tracers to include      : 82 83 84 85 86 87
IMIN, IMAX of region    :   1  72
JMIN, JMAX of region    :   1  46
LMIN, LMAX of region    :   1   1
```

http://wiki.geos-chem.org/GEOS-Chem_Input_Files#The_input.geos_file

```
# Code in PURPLE is only used for GCHP, not GC-Classic
EXPID:  OutputDir/GCHP
EXPDSC: GEOS-Chem_devel
CoresPerNode: 6

COLLECTIONS: 'inst',
             'avg6hr',
              ::
  inst.filename:       './GEOSChem.inst.%y4%m2%d2.nc4',
  inst.frequency:       010000,
  inst.duration:        240000,
  inst.mode:           'instantaneous',
  inst.fields:         'SpeciesConc_NO',   'GIGCchem',
                       'SpeciesConc_O3',   'GIGCchem',
                       'SpeciesConc_PAN',  'GIGCchem',
                       'SpeciesConc_CO',   'GIGCchem',
                       ... ETC ...
                        ::
  avg6hr.filename:     './GEOSChem.avg6hr.%y4%m2%d2.nc4',
  avg6hr.frequency:    002000,
  avg6hr.duration:     240000,
  avg6hr.mode:         'time-averaged',
  avg6hr.fields:       'Met_U10M',          'GIGCchem',
                       'Met_T',             'GIGCchem',
                       'SpeciesConc_CO',    'GIGCchem'
                        ::
```

The new `HISTORY.rc` file

Fields in PURPLE are only used by GCHP, can be ignored here.

Two diagnostic COLLECTIONS are defined:

- **inst (hourly instantaneous output), in red**
- **avg6hr (6-hr time-averaged output), in green**

Each COLLECTION contains several FIELDS:

- **Blue text = slices from array `State_Diag%SpeciesConc`**
- **Magenta text = met fields stored in `State_Met`**

Double colons :: are separators

Fields

- A FIELD represents a diagnostic quantity that will be saved to a netCDF file
  - FIELDS can be:
    - Species concentrations
    - Met fields
    - Other diagnostics, e.g.
      - J-values,
      - Aerosol OD's
      - Chemical rxn rates or P/L rates
      - Drydep velocities and fluxes
      - Amount of species lost to wet scavenging
      - etc.

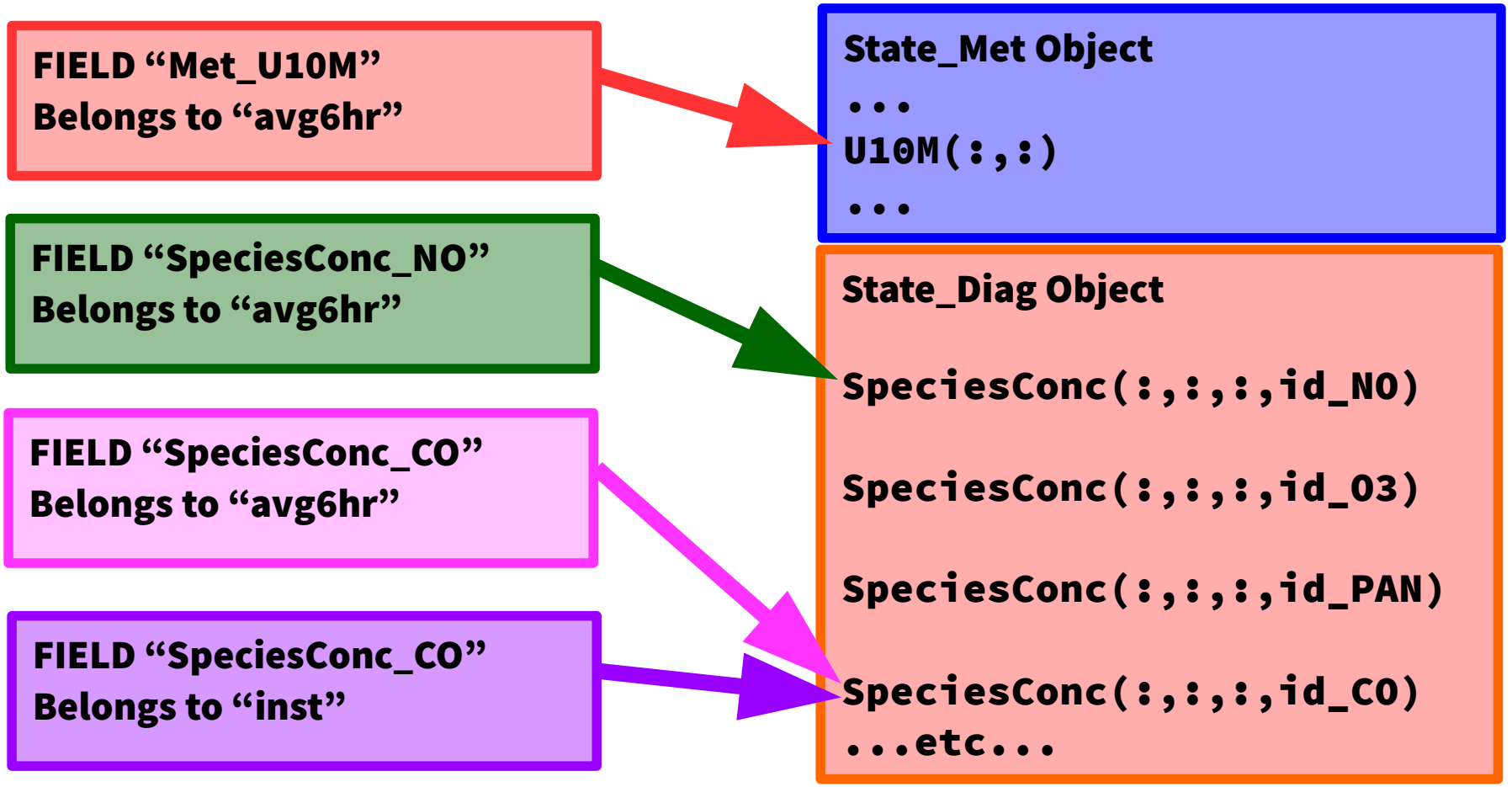
Fields

- FIELDS archive data from any of these modules:
  - **`State_Chm (state_chm_mod.F90)`**
    - FIELD name begins with “CHEM_”, “Chem_”, or “chem_”
  - **`State_Met (state_met_mod.F90)`**
    - FIELD name begins with “MET_”, “Met_”, or “met_”
  - **`State_Diag (state_diag_mod.F90)`**
    - This module will contain “target” arrays for diagnostic quantities that aren't already in State_Chm or State_Met
    - New in v11-02!

Data structure for each FIELD

- Each FIELD contains:
  - Identifying info (name, ID number etc.)
  - NetCDF variable metadata (`long_name`, `units`, etc.)
  - "Rank" of the data (0D, 1D, 2D, or 3D)
  - "Kind" of the data (`INTEGER`, `REAL*4`, `REAL*8`)
  - Arrays to hold data values (0D, 1D, 2D, or 3D)
  - Pointers to the "target" data (e.g. in `State_Chm` etc.)
  - A counter to increment the # of times the FIELD is updated

FIELDS point to data arrays in GC



Each FIELD belonging to a COLLECTION points to a “target” member of one of the State_Chm, State_Met, or State_Diag objects. FIELDS in different COLLECTIONS can point to the same “target”, thus reducing the amount of memory required.

Collections

- A COLLECTION is a series of netCDF files that contain diagnostic output
  - Files corresponding to a COLLECTION can be saved to disk, hourly, monthly, daily, etc., depending on the settings in HISTORY.rc
  - Each collection contains one or more FIELDS
  - You can have as many collections as you wish

Collection Properties

- COLLECTIONS have 2 properties:
  - **Instantaneous (aka "Timeseries")**
    - Similar to e.g. ND49 timeseries diagnostic.
    - Each FIELD gets new data from its "target", which is then held in an array, then later written to disk later in the timestep.
  - **Time-averaged**
    - Similar to diags in "DIAGNOSTIC MENU" of the input.geos file
    - Each FIELD gets new data from its "target", which is added into an "accumulator array".
    - The number of updates is also incremented, so that the time-average of the FIELD can be computed.

Collection Operations

- COLLECTIONS have 3 associated operations:
  - **Update**
    - FIELDS are updated with new values from their "targets" (i.e. arrays in State_Chm, State_Met, or State_Diag)
  - **File Close**
    - The currently-open netCDF file is closed
    - The netCDF file for the next diagnostic interval is created
  - **File Write**
    - FIELDS are averaged (for time-averaged COLLECTIONS only)
    - FIELDS are written to the netCDF file

Data structure for each COLLECTION

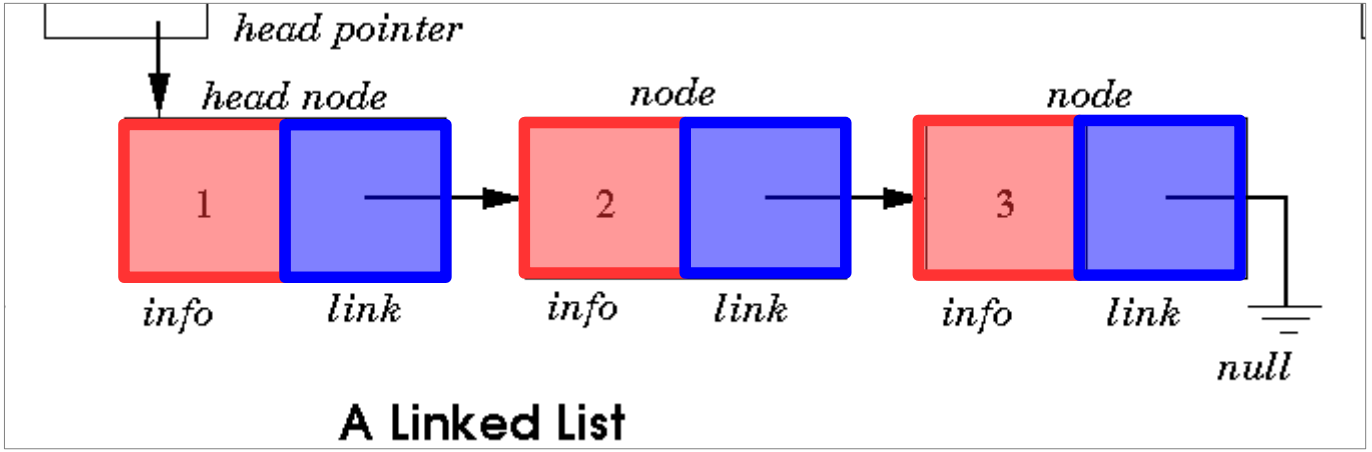
- Each COLLECTION contains:
  - NetCDF file info (name, file ID #, dimension ID #'s, etc.)
  - NetCDF metadata (aka the global attributes)
  - List of FIELDS to be saved out to the netCDF file
  - Timing information:
    - When to do the Update, File Write, File Close operations
  - Are FIELDS defined on vertical level edges or centers?
    - By convention, one or the other is allowed per file, but not both

Making the master data structure

- We have defined data structures (“objects”) for individual FIELDS and COLLECTIONS, which is great!
- But what we really need is a way of arranging these into ordered lists:
  - A master list of COLLECTIONS, and
  - Each COLLECTION has a list of FIELDS
- For this, we rely on the “linked list” concept.

Linked lists

- A linked list is a data structure that has two parts
  - Data
    - Can be any type of variable
  - A pointer to the next node in the list
    - You can have an unlimited number of nodes in the list



More about linked lists ...

Essentials: Pointer Power! - Computerphile

Computerphile 80K views • 1 month ago

Pointers are fundamental in programming and Professor Brailsford couldn't live without them! Professor Brailsford's Code: ...

4K CC

Linked Lists - Computerphile

Computerphile 82K views • 8 months ago

Linked Lists explained: Dr Alex Pinkney returns to **Computerphile**. Apolo gies for the traffic noise on this episode - we tried filming ...

Arrays vs Linked Lists - Computerphile

Computerphile 106K views • 2 months ago

Which is faster? The results *may* just surprise you. Dr 'Heartbleed' Bagley gives us an in depth shoot-out - Arrays vs **Linked Lists** ...

4K CC

The **Computerphile** channel on Youtube has several excellent videos on how linked lists and pointers work. Check them out!

Schematic of master diagnostics linked list. Derived type names are in parentheses.

Etc. add more types of diagnostic collections (hourly, monthly, restart, etc.)
NOTE: Naming convention not 100% finalized as of this writing

Looping through Collections & Fields

- Loop through each COLLECTION in the master list
  - If it's time for **Update**
    - Loop over each FIELD In the COLLECTION
      - Update each FIELD in the COLLECTION w/ new data from its “target”
  - If it's time for **File Close**
    - Close the netCDF file specified by this COLLECTION
    - Open the netCDF file for the next diagnostic interval
  - If it's time for **File Write**
    - Loop over each FIELD in the COLLECTION
      - Compute the time average of each FIELD (if necessary)
      - Write each FIELD in the COLLECTION to the netCDF file

When diagnostics are called

Beginning of "heartbeat' timestepping loop in main.F (time = T)

Transport
Dry Deposition
Emissions
PBL Mixing
Cloud Convection
Chemistry
Wet Deposition

UPDATE FIELDS IN EACH COLLECTION (if it's time)

Increment Elapsed Time

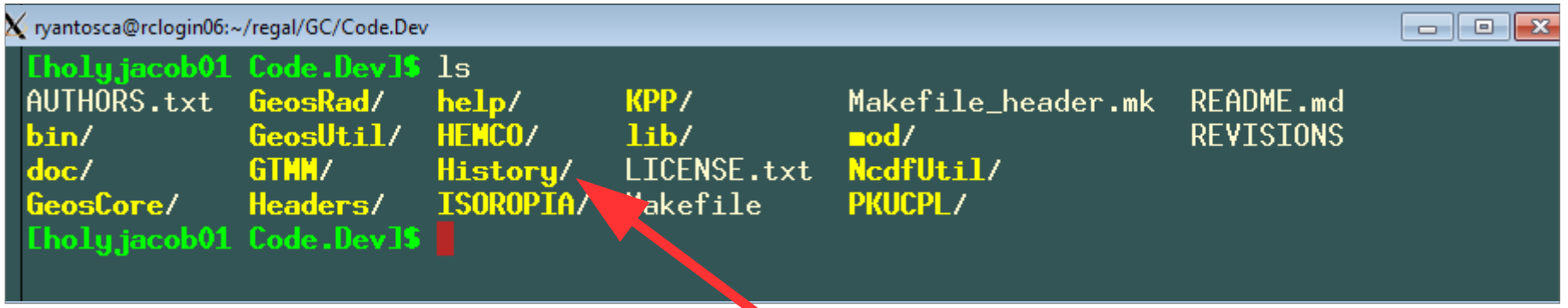
CLOSE FILE / OPEN NEXT FILE FOR EACH COLLECTION (if it's time)
WRITE FIELDS FOR EACH COLLECTON TO FILE (if it's time)

- **NOTE: Data will be timestamped with end-of-timestep time = T + ΔT**

End of "heartbeat" timestepping loop in main.F (time = T + ΔT)

Location of netCDF diag code

- NetCDF diagnostics code for GEOS-Chem “Classic” lives in the History folder of the source code:



Location of netCDF diagnostic code (in v11-02 and later versions

Examples: How to schedule diagnostic output?

Example 1: Instantaneous collection

```
inst.filename:   './GEOSChem.inst.%y4%m2%d2.nc4',
inst.frequency:  010000,
inst.duration:   240000,
inst.mode:       'instantaneous',
inst.fields:     'SpeciesConc_NO',  'GIGCchem'
                 'SpeciesConc_O3',  'GIGCchem',
                 'SpeciesConc_PAN', 'GIGCchem',
                 'SpeciesConc_CO',  'GIGCchem',
                  ... ETC ...
                  ::
```

Result:
FIELDS are updated and saved out to disk each hour.

A new file is created each day.

Each file will have 24 time values.

File Write
Interval is defined with the **"frequency"** tag.

Update
This interval is automatically set equal to the **File Write** interval.

File Close
Interval is defined with the **"duration"** tag.

Instantaneous
Defined by the **"mode"** tag.

Example 2: Time-averaged collection

```
avg6hr.filename:      './GEOSChem.avg6hr.%y4%m2%d2.nc4',
avg6hr.frequency:     060000,
avg6hr.duration:      240000,
avg6hr.mode:          'time-averaged',
avg6hr.fields:        'Met_U10M',          'GIGCchem',
                      'Met_T',             'GIGCchem',
                      'SpeciesConc_CO',    'GIGCchem'
                      ::
```

Result:
Fields are updated every 10 minutes, and averaged into 6-hour intervals.

A new file is written each day.

Each file will have 4 time values (6-hr intervals).

File Write
Interval is defined with the **"frequency"** tag.
This also determines the averaging period for the data.

Update
Interval is set by default to the dynamic "heartbeat" timestep, which is set in the `input.geos` file (= 10 min for most simulations)

File Close
Interval is defined with the **"duration"** tag.

Time-averaged
Defined by the **"mode"** tag.

Example 3: Monthly mean output

```
avg6hr.filename:      './GEOSChem.avg6hr.%y4%m2%d2.nc4',
avg6hr.frequency:     000100 000000,
avg6hr.duration:      010000 000000,
avg6hr.mode:          'time-averaged',
avg6hr.fields:        'Met_U10M',          'GIGCchem',
                      'Met_T',             'GIGCchem',
                      'SpeciesConc_CO',    'GIGCchem'
                      ::
```

Two sets of 6 digits are interpreted as **YYMMDD hhmmss**.

One se t of 6 digits is interpreted as **hhmmss**.

The default time format of `HISTORY.rc` is `hhmmss` (hrs/mins/secs).

We also allow the **Update**, **File Write** and **File Close** operations to occur at intervals of **1 month** or **1 year**. (Longer periods are harder to implement, as we have to be concerned about straddling leap years, etc.)

Result: Fields will be updated every 20 minutes, and averaged into monthly intervals. A new file will be created once per year.

Example 4: Wildcards!

```
avg6hr.filename:      './GEOSChem.avg6hr.%y4%m2%d2.nc4',
avg6hr.frequency:     000100 000000,
avg6hr.duration:      010000 000000,
avg6hr.mode:          'time-averaged',
avg6hr.fields:        'Met_U10M',             'GIGCchem',
                      'Met_T',                'GIGCchem',
                      'SpeciesConc_?ADV?',    'GIGCchem'
                      ::
```

You can also specify wild cards for species names. This will prevent you from having to list several species individually.

?ADV? = all advected species
?AER? = all aerosol species
?GAS? = all gas-phase species
?DRY? = all drydep species
?WET? = all wetdep species
?PHO? = all phtolysis species
?KPP? = all species in KPP mechanism
?VAR? = all active KPP species
?FIX? = all inactive KPP species
?ALL? = all species

Example 5: Structure of the created netCDF files

```
netcdf GEOSChem.inst.20130701 {
dimensions:
      time = UNLIMITED ; // (1 currently)
      lev = 72 ;
      ilev = 73 ;
      lat = 46 ;
      lon = 72 ;
variables:
      float AREA(lat, lon) ;
            AREA:long_name = "Surface area" ;
            AREA:units = "m2" ;
      double time(time) ;
            time:long_name = "Time" ;
            time:units = "minutes since 2013-07-01 00:00:00 UTC" ;
            time:calendar = "gregorian" ;
            time:axis = "T" ;
      double lev(lev) ;
            lev:long_name = "hybrid level at midpoints ((A/P0)+B)" ;
            lev:units = "level" ;
            lev:axis = "Z" ;
            lev:positive = "up" ;
            lev:standard_Name = "atmosphere_hybrid_sigma_pressure_coordinate" ;
            lev:formula_terms = "a: hyam b: hybm p0: P0 ps: PS" ;
      double ilev(ilev) ;
            ilev:long_name = "hybrid level at interfaces ((A/P0)+B)" ;
            ilev:units = "level" ;
            ilev:positive = "up" ;
            ilev:standard_Name = "atmosphere_hybrid_sigma_pressure_coordinate" ;
            ilev:formula_terms = "a: hyai b: hybi p0: P0 ps: PS" ;
      double lat(lat) ;
            lat:long_name = "Latitude" ;
            lat:units = "degrees_north" ;
            lat:axis = "Y" ;
```

Example 5: Structure of the created netCDF files

```
double lon(lon) ;
      lon:long_name = "Longitude" ;
      lon:units = "degrees_east" ;
      lon:axis = "X" ;
double hyam(lev) ;
      hyam:long_name = "hybrid A coefficient at layer midpoints" ;
      hyam:units = "hPa" ;
double hybm(lev) ;
      hybm:long_name = "hybrid B coefficient at layer midpoints" ;
      hybm:units = "1" ;
double hyai(ilev) ;
      hyai:long_name = "hybrid A coefficient at layer interfaces" ;
      hyai:units = "hPa" ;
double hybi(ilev) ;
      hybi:long_name = "hybrid B coefficient at layer interfaces" ;
      hybi:units = "1" ;
double P0 ;
      P0:long_name = "reference pressure" ;
      P0:units = "hPa" ;
float SpeciesConc_NO(time, lev, lat, lon) ;
      SpeciesConc_NO:long_name = "SPC_NO concentration" ;
      SpeciesConc_NO:units = "mol/mol dry" ;
      SpeciesConc_NO:_FillValue = -1.e+31f ;
      SpeciesConc_NO:averaging_method = "instananeous" ;


... Etc ...
```

Near-future work

Stuff we are still working on …

- Investigate how the new diagnostic structure can be used for checkpointing
  - Creating restart files
  - Creating nested-grid boundary condition files
  - Creating checkpoint files for the adjoint simulations
- Validation
  - Comparisons with existing bpch diagnostics
- Documentation
  - e.g. how to add new diagnostic quantities

References

- Official netCDF site
  - Download netCDF versions from here (but we recommend to use the netCDF modules that are on Odyssey).
- Preparing data files for use with HEMCO (GC wiki)
  - General info about netCDF files, how to manipulate them
- Python code for GEOS-Chem (GC wiki)
  - Packages for visualizing netCDF output, written in Python
- GCPy
  - Describes our Python package for GEOS-Chem, which is currently under development

References

- GEOS-Chem Input Files (GC wiki)
  - Lists the diagnostic options in the input.geos file
- GEOS-Chem Output Files (GC wiki)
  - Lists the restart and diagnostic files created by GEOS-Chem
- GAMAP manual, Chapter 6.2
  - Describes the binary punch file format in detail