Computing Environment Support

Nancy Collins nancy@ucar.edu

or

dart@ucar.edu

Roadmap

- Getting code, requirements
- Installing and compiling
- Converting observations
- Running models
- Diagnostics
- Batch system, Memory, Parallel Considerations

Getting DART

- DART Web Pages
  - http://www.image.ucar.edu/DAReS/
- Download link
- Gives subversion instructions
- Need ‘svn’ subversion client
  - http://www.image.ucar.edu/~thoar/svn_primer.html

Installing and Compiling

- DART requires a Fortran 90 compiler
  - Intel, PGI, IBM, gfortran, pathscale
- Also requires NetCDF libraries
  - http://www.unidata.ucar.edu/software/netcdf
- If running in parallel, requires MPI libraries
  - Usually standard on clusters & supercomputers
  - NOT required, DART can be compiled without it

Installing and Compiling (cont)

- We use generic Fortran 90 constructs
  - Usually compiles easily on new platforms
- Most problems are linking with NetCDF
  - The NetCDF libs MUST BE COMPILED WITH SAME FORTRAN COMPILER AS YOU ARE USING FOR DART. Also must include the Fortran interfaces.
  - At NetCDF build time, can choose to have a single libnetcdf.a lib, or both libnetcdf.a and libnetcdff.a (note 2 Fs at end of name)

Installing and Compiling (cont)

- Use the ‘mkmf’ utilitity
- cd into the `DART/mkmf` directory and find the closest `mkmf.template.xxx` file
- Copy that into `mkmf.template`
- cd into a `DART/model/xxx/work` directory
- Run `quickbuild.csh` to compile

DART Documentation

- Web pages:
  - http://www.image.ucar.edu/DAReS/
- Documentation comes with subversion checkout:
  - index.html in top level DART directory points to all other DART documentation throughout the directory tree
  - doc/html/Lanai_release.html
- Browse documentation directly from SVN server:
  - https://proxy.subversion.ucar.edu/DAReS/DART/releases/Lanai

Converting Observations

- DART uses a proprietary file format (for now)
- Existing observation converters are found in the DART tree under `DART/observations`
- Most frequently used atmospheric observation types already have converters available

Running Models - OSSEs

- OSSE observations:
  - create_obs_sequence
  - create_fixed_network
- OSSE execution:
  - perfect_model_obs
  - filter
- OSSE diagnostics:
  - obs_diag
  - obs_seq_to_netcdf

Running Models – Real Observations

- Observations:
  - NCEP
  - MADIS
  - SSEC
- Execution:
  - filter
- Diagnostics:
  - obs_diag
  - obs_seq_to_netcdf

Fortran 90 Namelists

- All DART programs use Fortran namelists for run-time configurations
- Always in a single file, always named 'input.nml'
- Each part of the system has its own namelist
- Documented in the html files

Diagnostics

- DART includes a large number of MATLAB scripts
  - Interactive GUI
  - “Offline” batch mode
  - Plotting “observation space” values
  - Looking at “state space” mean and spread

Parallelism

- DART is a parallel program that uses MPI
- Multiple tasks (copies of the same program) are running at the same time, exchanging data
- Small models can be compiled without MPI
- Larger models use parallelism for speed and to have access to more memory
- You generally need a cluster or supercomputer for parallel jobs

Batch Systems

- Job submission software on large parallel machines
  - LSF, PBS, Moab/Torque, SGE
- Most common issue is memory size on an individual node
  - Multiple CPUs often share memory on a ‘node’
  - Machines are connected networks of ‘nodes’
  - Tasks on the same node share node memory
  - Starting fewer tasks than CPUs gives each task more memory

Modes of Running

- If model is single-executable and memory issues not a problem, DART can advance the model from inside filter
- If model is large and parallel, may want to run model advances and DART as separate jobs

Modes of Running (cont)

- 'filter' uses the observation sequence file to decide how long to run and what model data is needed
  - Long obs_seq files good when filter runs model
  - Single-assimilation-window files good for scripted advances
- Scripted advances mean a job script first runs N model advances, then runs filter, then repeats
  - Generally more flexible and easier when the model is itself a parallel program

Modes of Running (cont)

- The number of MPI tasks to get best performance from the model will be different than the number of MPI tasks for “filter”
- “filter” requires enough tasks and memory to process all N ensemble members at once
  - Often need to run with fewer tasks than CPUs on a node to get sufficient memory