In memoriam

Amy Langenhorst  1977-2016
Overview

• Intro to GFDL, FMS, FRE
• Climate modeling workflow achievements
• How we’re approaching a workflow rewrite
• Challenges, lessons learned
Geophysical Fluid Dynamics Laboratory

• Mission: *To advance scientific understanding of climate and its natural and anthropogenic variations and impacts, and improve NOAA's predictive capabilities*

• Joint Institute with Princeton University

• About 300 people organized in groups and teams
  • 8 scientific groups
  • Technical Services
  • Modeling Services
    • Framework for coupled climate modeling (FMS)
    • Workflow software (FRE), Curator Database
    • Liaisons to scientific modeling efforts
Flexible Modeling System

- FMS is a software framework which provides infrastructure and interfaces to component models and multi-component models.

- Started ~1998, development timelines:
  - Component models by scientific group, continuous
  - Annual FMS city releases, 200+ configurations tested
Workflow Goals

- Reproducibility
- Perturbations
- Differencing
- Robustness
- Efficiency

Error handling: user level & system level

From the climate model run...

...through postprocessing...

...to automatically generated figures.
FRE is a set of workflow management tools that read a model configuration file and take various actions.

Development cycle is independent of FMS but responds to scientific needs.

- Acquire Source Code → Compile
- Configure Model → Transfer Input Data
- Transfer Output Data
- Regrid Diagnostic Output → Average Diagnostic Output
- Create Figures

Development Timeline:
- Started ~2002
- Wide adoption ~2004
FRE Technologies

- XML model configuration files, and schema
  - User and canonical XML
- Perl command line utilities read XML file
- Site configuration files
  - Script templates
  - Conventions for file names and locations
- Environment modules to manage FRE versions
- Moab / Torque to schedule jobs across sites
- Transfer tools: globus + local gcp, HSM tools
- NCO Operators + local fregrid, plevel, timavg
- Jenkins for automated testing
- Coming soon: Cylc
FRE Features

- Encapsulated support for multiple sites
  - Compiler, scheduler, paths, user defaults
- Model component-based organization of model configuration
- Integrated bitwise model testing
  - Restarts, scaling (vs. reference and self)
- Experiment inheritance for perturbations
- User code plug-ins
- Ensemble support
- Postprocessing and publishing, browsable curator database
FRE Job Stream Overview

- Experiments run in segments of model time
  - Segment length is user-defined
  - More than one segment per main compute job is possible
- After each segment:
  - State, restart, diagnostic data is saved
  - Model can be restarted (checkpointing)
  - Data is transferred to long term storage
  - A number of post-processing jobs may be launched, highly task parallel
Flow Through The Hardware

1. Remote transfer of input data
2a. Transfer/preprocess input data
2b. Model execution
2c. Transfer/postprocess output data
3. Remote transfer of output data
4. Post-processing

Image: Tara McQueen
GFDL Data Stats

- Networking capacity ORNL to GFDL: two 10gig pipes, 120 TB/day theoretical for each pipe
- Analysis cluster:
  - ~100 hosts with 8 to 16 cores, 48GB to 512GB memory, local fast scratch disk of 9TB to 44TB
  - 4 PB/week throughput
- Tape-based data archive: 60PB
  - ~2PB disk attached
  - Another ~2PB filesystem shared among hosts for caching intermediate results
- 1300 auto-generated figures from atmos model
- An early configuration of CM2.6 (0.1 degree ocean, 0.5 degree atmos) runs on 19,000 cores
  - a year of simulation takes 14 hours and generates 2TB of data per simulation year, ran 300 simulation years
Post-processing Defined

- Preparing diagnostic output for analysis
  - Time series: Hourly, daily, monthly, seasonal, annual
  - Annual and Seasonal climatological averages
  - Horizontal interpolation: data on various grids can be regridded to lat-lon with “fregrid”
  - Vertical interpolation: to standard pressure levels
  - Hooks to call user scripts to create plots or perform further data manipulation
  - Enter the model into the curator database
- Requirement: must run faster than the model
- Self healing workflow: state is stored; tasks know their dependencies and resubmit as needed
10 Years Later...

Oh, the clusters you’ll run on and the lessons you’ll learn...
FMS FRE Chaco

- Rewrite of FRE begining with post processing
- Maintain compatibility and historical behavior, yet standardize tool behavior
  – Old (user) interfaces available, e.g. ‘drop in’ replacement where possible
- Improve visibility and control of experiments
Chaco Major Goals

- Robustness and reliability
- Support for high resolution resource requirements
  - CM2.6, a higher resolution model, generates 2TB per simulation year, completes 2 sim years/day
- Support for discrete toolset that can be used without running end to end “production frepp”
- Monitoring
- Increased task parallelism
- Maintain existing functionality
- Keep pace with data flow rate from remote computing sites (gaea/theia/titan…)
High Resolution Strategy

- Reduce memory and disk space requirements
- Initially break all diagnostic data up into “shards”
  - “Shard” files contain one month’s worth of data for one variable
  - Shards can be on model levels or regridded
  - Perform data manipulations on one variable at a time, then combine data later if necessary
  - Operations on shards can be highly parallelized
- Make intelligent use of disk cache with intermediate data
- Reducing data movement is key
  - Overwhelming majority of the time spent in postprocessing is simply moving data
Black box to discrete toolset

Postprocessing

• Outside looks the same

frepp -v -c split -s -D -x FILE.XML -P SITE.PLATFORM-COMPILER -T TARGET-STYLE -d MODEL_DATA_DIRECTORY -t START_TIME EXPERIMENT_NAME

• Inside nothing is the same
<table>
<thead>
<tr>
<th>Bronx</th>
<th>Chaco</th>
</tr>
</thead>
<tbody>
<tr>
<td>fremake</td>
<td>fre cylc run</td>
</tr>
<tr>
<td>frerun</td>
<td>fre cylc prepare</td>
</tr>
<tr>
<td>frepp</td>
<td>fre list</td>
</tr>
<tr>
<td>frelist</td>
<td>fre monitor</td>
</tr>
<tr>
<td>frestatus</td>
<td></td>
</tr>
<tr>
<td>frecheck</td>
<td></td>
</tr>
<tr>
<td>frepriority</td>
<td></td>
</tr>
<tr>
<td>freppcheck</td>
<td></td>
</tr>
<tr>
<td>frescrub</td>
<td></td>
</tr>
<tr>
<td>fredb</td>
<td></td>
</tr>
</tbody>
</table>

- fremake: obtain code, create/submit compile scripts
- frerun: create and submit run scripts
- frepp: create and submit post-processing scripts
- frelist: list info about experiments in an xml file
- frestatus: show status of batch compiles and runs
- frecheck: compare regression test runs
- frepriority: change batch queue information
- freppcheck: report missing post-processing files
- frescrub: delete redundant post-processing files
- fredb: interact with Curator Database
Bronx Design

• Bronx implementation: monolithic, linear perl script re-invoking itself over subsequent segments of model data. Steps to produce desired products hardcoded in blocks of shell which are assembled and submitted to batch scheduler.
Chaco Design, 1

• Discrete tools in a unified code base: data movement, refinement, interpolation, timeseries and timeaverage
  
  \texttt{fre data \{get, split, zinterp, xyinterp, refineDiag, analysis, timeaverage, timeseries\}}

• Cylc-based cycle for experiment duration and model segments

• Per-cycle tool runs
Chaco Cylc

- Cylc for the dependency and scheduling engine
- Cylc suites generated by FRE Chaco
  - Allows for different grouping of tasks based on arguments, XML, or other factors.
  - User accessible / modifiable, but ‘hidden’
- Leverage Cylc task management and job submission
- Cylc GUI gives visibility and control
Chaco Design, 2

- Segments run in parallel:
  - Multiple MOAB jobs on separate nodes

- Tool commands
  - Utilize FRED to store parsed XML, diagfield info
  - Run concurrently over model components and model variables (shards)
  - Log all stdout, stderr, cpu, memory utilization, execution time
Chaco Tool Outline

• Modularize the monolithic workflow
• Standardize interface
  – ‘fre CATEGORY ACTION [OPTIONS]’
• Can operate independent of workflow
• On-disk data structure (FRED) stashes parsed experiment details
• File inventory and tool run databases
• Tools know input and output files
Chaco Tech

• Perl OO via Moose and friends
  – Path to Perl 6
• CPAN all things
• Test driven development with continuous integration tactics.
• Strong source code practices, code documentation and project management tools to enable the team.
• Get, Split, and Stage
  Retrieve data from storage, separate into variable shards, copy to shared temp filesystem.

• Interpolate in Z, and XY
  Stage shards in, interpolate, stage out

• Generate Timeseries
  Stage shards in, timeseries, store product

• Generate Timeaverage
  Stage shards in, timeseries, store product
FRE PostProcessing, 2

Reality check!

• Get, “RefineDiag”, Split, and Stage
  Retrieve data from storage, run user-specified scripts to “refine data”, separate into variable shards, copy to shared temp filesystem.

• Interpolate in Z, and XY
  Stage shards in, interpolate, stage out

• Generate Timeseries, timeaverage
  Stage shards in, timeseries / timeavg, store product

• “Analysis”
  Run user-specified scripts to generate figures, etc
Dependency and Recovery check!

• Add knowledge of inputs and outputs to tools

• A Tool Operation framework for recovery and dependency-checking
  • Check for output (in final and local locations)
  • Check for inputs (in local and staged locations)
  • Run tool
FRE Operation

Inputs

Staged Inputs

could be 1:1 file copy or untarring

Outputs

Tool actions
- Split
- Interp
- TimeAvg
- TimeSeries

TOOL

Located on shared filesystems or archive

Staged Outputs

currently tape archive, but could be ptmp
Running an Operation

- **Do staged outputs exist?**
  - no
- **Do local outputs exist?**
  - no
- **Do local inputs exist?**
  - no
- **Do staged inputs exist?**
  - no

**Goal:** work avoidance

- **Stage out**
  - yes
- **run Commands**
  - yes
  - no
  - Error
- **Stage in**
  - yes

**Done**
**Operation Granularity**

### Split → Interp → Products

- **Split**
  - xyinterp
  - zinterp

- **Interp**
  - one per comp history file

- **Products**
  - one per segment
  - TS, TA, CS
  - one per product

### Granularity
- one per comp history file
- one per segment
- one per product

### Typical Run
- 10 / yr
- 5 / yr
- 100 / 100 yr experiment

### Goals
- maximize outer-parallel
- make each be able to be a Cylc task without performance cost
- work avoidance

### Requests in XML (Examples)
- 5 yr monthly TS ocean regridded to 1 deg
- All statics from land
- 10 yr seasonal TA atmos tracer native grid
Goal: maximize inner-parallel by using tiers to isolate dependent commands

Example: One operation to create one annual timeaverage file containing all variables

<table>
<thead>
<tr>
<th>Tier</th>
<th>Action</th>
<th>Tool</th>
<th>Parallelization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier 1</td>
<td>Concatenate shards to create TS</td>
<td>ncrcat</td>
<td>Number of variables</td>
</tr>
<tr>
<td>Tier 2</td>
<td>Average over TS timesteps</td>
<td>timavg (FRE tool)</td>
<td>Number of variables</td>
</tr>
<tr>
<td>Tier 3</td>
<td>Combine per-variable TA files into single file</td>
<td>ncks</td>
<td>1</td>
</tr>
</tbody>
</table>
FRE Workflow, 3yr test

3 year test and verify experiment
FRE Workflows

Simplified 1yr cycle of 100yr experiment

load_diagfields +0007610101T0000Z

trigger_cycle_lyr +0007610101T0000Z

get_split_stage_lyr +0007610101T0000Z

static_and_interpolation_atmos_lyr +0007610101T0000Z

static_and_interpolation_atmos_4xdaily_lyr +0007610101T0000Z

static_and_interpolation_atmos_8xdaily_lyr +0007610101T0000Z

static_and_interpolation_atmos_8xdaily_reined_lyr +0007610101T0000Z

static_and_interpolation_atmos_daily_lyr +0007610101T0000Z

static_and_interpolation_atmos_daily_reined_lyr +0007610101T0000Z

static_and_interpolation_atmos_diurnal_lyr +0007610101T0000Z

frc.verona.SM2_Control-1990.multiple.07610101-08600101
Current Status

- Alpha release available: `module load fre/chaco-alpha`
  - Have achieved full functionality for “quickstart” AM2, simplified c48 AM3, ESM2G, OM4, AWG
  - Differences in netcdf output vs. bronx exist but are understood
  - Workflow recovery dependency analysis
  - `refineDiag`: Create custom variables between run and postprocessing
  - Performance test performed; results currently under review
- Current work includes:
  - Adding ability to run in production from Bronx runscript
  - Review of performance data
  - Evaluate concurrency and parallelism
  - Adding experiment configurations to our test suite
Cool! So all the workflow issues are solved! Right??
When is the workflow broken?

• As complexity increases, a smaller and smaller percentage of error is tolerable
  • If 1% of jobs fail and your workflow launches thousands of jobs, you still have lots of work

• Our strategy: standard logging, retries, recovery, error handling, Cylc, automated test infrastructure in Jenkins…
Analyzing Chaco 1

• Analyze timing and resource metrics

  *Performance Analysis of Chaco*

• Tracking details at many levels
  • Individual commands (ncks, gcp)
  • Tiers of commands (run concurrently)
  • Command groupings
  • Segment runs
  • Tool runs
Analyzing Chaco 2

• What can we determine?
  • Variability
  • Data movement costs
  • Resource usage

• Areas under investigation
  • Per-node concurrency
  • Coalescing tasks
Pros and cons: user scripts

- Users can plug in shell scripting in several places
  - In runscript, before & after postprocessing
- Enables scientific exploration
- Enables great flexibility within FRE
- Often terrible for the workflow, if best practices are not followed
- Error catching/handling is difficult
Aiming for disk affinity

- Data movement can be reduced if jobs that will reuse the same source data are launched on the same host with data in the attached local disk.
- Implemented to a small degree in FRE, as a high-res option.
- Must tune task parallelism vs. reduced transfers.
- Difficult batch scheduling task to load balance the system.

job schooling

Art by zazerkale, redbubble.com
Workflow Discussion Points

• How can we reduce data movement?
• How much disk affinity can we achieve?
• Robustness: can we detect / recover from new error modes?
• How can we better leverage each other’s work?
  • Automation vs. HPC security concerns
• How can we manage user plug-ins to the workflow?
• Do you trust your state files or check the filesystem?
• How do we handle user changes during a run?
Questions?
Answers?
Sample XML: Setup

```xml
<experimentSuite>
  <setup>
    <platform name="gfdl.default">
      <directory stem="subdirectory/paths"/>
      <csh>
        module load intel
        module load fre
      </csh>
      <property name="FMS_ARCHIVE_ROOT" value="/archive/fms"/>
    </platform>
  </setup>
  <experiment name=""/>
</experimentSuite>
```
Sample XML: Experiment

```xml
<experiment name="CM2.5A_2" inherit="CM2.5">
  <component name="fms">
    <source vc="cvs" root="/home/fms/cvs">
      <codeBase version="riga">shared</codeBase>
      <csh>
        cvs up -r riga_arl shared/.../file
      </csh>
    </source>
  </component>
  <compile>
    <cppDefs>-Duse_netCDF</cppDefs>
  </compile>
</component>
<component name="mom4p1"> ...
```
<input>
  <namelist name="vert_diff_driver_nml">
    do_conserve_energy = .true.
  </namelist>
  <dataFile label="input" target="INPUT/">
    checksum="" size="" timestamp="">
      $(FMS_ARCHIVE_ROOT)/am2/cover_type_field
  </dataFile>
  <diagTable>
    diagnostic variable table
  </diagTable>
</input>
<runtime>

<production simTime="26" units="years"
npes="120" runTime="12:00:00">
  <segment simTime="12" units="months"
runTime="06:00:00"/>
</production>

<regression name="basic">
  <run days="8" npes="60" runTime="00:15:00"/>
</regression>

<dataFile label="reference">
  $(FMS_ARCHIVE_ROOT)/am2/riga/19950108.tar
</dataFile>
</runtime>
Sample XML: Postprocessing

<postProcess>
  <component type="atmos" source="atmos_month" sourceGrid="atmos-cubedsphere"
    xyInterp="90,144" zInterp="era40">
    <timeSeries freq="monthly"
     (chunkLength="5yr">
      <variables> precip, temp </variables>
      <analysis script="/home/template.csh"/>
    </timeSeries>
    <timeAverage source="monthly"
      interval="5yr"/>
  </component>
</postProcess>
/archive/$USER/fre/CM2.1U_Control-1990_E1.M_3A
|-- ascii
|-- restart
|-- history
`-- pp
  |-- atmos
  |  |-- ts
  |  |  |-- monthly
  |  |  |  `-- 5yr
  |  |  |-- atmos.000101-000512.precip.nc
  |  |  `-- atmos.000601-001012.temp.nc
  |  `-- av
  |     |-- monthly_5yr
  |     |  |-- atmos.0001-0005.01.nc
  |     |  `-- atmos.0001-0005.02.nc
  ...