16th Annual CESM Workshop’s Software Engineering Working Group

Parallel Analysis of GeOscience Data Status and Future

Jeff Daily

PI: Karen Schuchardt, in collaboration with Colorado State University’s Dave Randall et al

http://svn.pnl.gov/gcrm/wiki/pagoda
pagoda-dev@googlegroups.com
Motivation

Data sets approaching PB range
- **GCRM = 1.4 PB**
  - 4 km resolution
  - 3 hourly, 1 simulated year
- **CCSM4 = 100 TB**
  - 0.1 degree ocean
  - 0.24 degree atmosphere
  - 1 simulated century
- 1 PB / 10GB/s = 28 hours
- 1 PB / 300MB/s > 40 days
- IO bandwidth is the bottleneck
- 64bit offsets needed to describe file

Support GCRM’s geodesic grid
- Semi-structured
- Explicit topology variables

Image courtesy of Prabhat at Berkeley Lab using VisIt tool on GCRM data.
Approach

- Parallel IO part of the design
  - Parallel NetCDF
  - NetCDF4/HDF5

- Data-parallel versus task parallel
  - Data distribution and communication using Global Arrays library
  - Task parallel is still possible

- C++ API for custom analysis
  - Similar to Java NetCDF API
  - IO and Array operations

- Command-line tools for immediate use
  - Mimic an established interface
  - Drop-in replacement for NetCDF Operators

- Support geodesic and regular grids
Original GCRM Data Model

CF compliant cell-based data model
- Needed variables defined on corners, edges
- Redundancy in the data (cell bounds are logically shared)
- Cannot traverse cells for visualization (e.g. isolines) or analysis

Enhancements to NetCDF Operators
- Auxiliary coordinate support for cell-based grids
- Performance improvements for hyperslabbing (subsetting)
- No support for explicit topology

Dimensions:
- cell = 10240
- nv = 6

Variables:
- float center_lon(cell)
- float center_lat(cell)
- float corner_lon(cell, nv)
- float corner_lat(cell, nv)

ncks -X lon_min,lon_max,lat_min,lat_max vs.
ncks -d lon,min,max -d lat,min,max
// Dummy scalar for grid discovery
int grid;
    
    grid:standard_name = "grid";
    grid:external_ref = "some uri";
    grid:cell_type = "hex";
    grid:index_start = 0;
    // topology references
    grid:cell_edges = "cell_edges";
    grid:cell_corners = "cell_corners";
    grid:cell_cells = "cell_neighbors";
    grid:edge_corners = "edge_corners";
    // geometry references
    grid:coordinates_cells = "center_lon center_lat";
    grid:coordinates_corners = "corner_lon corner_lat";
    grid:coordinates_edges = "edge_lon edge_lat";
Data Model (cont.)

dimensions:
  cells      =  41943042
  corners    =  83886080
  edges      = 125829120
  time       = UNLIMITED
  layers     =  24
  interfaces =  25

variables:
  float temperature(time,cells,layers)
  float wind(time,edges,interfaces)
  float center_lon(cells)
  float center_lat(cells)
  float corner_lon(corners)
  float corner_lat(corners)
  float edge_lon(edges)
  float edge_lat(edges)
  int    cell_corners(cells, cellcorners=6)
  int    cell_edges(cells, celledges=6)
  int    edge_corners(edges, edgecorners=6)
  int    cell_neighbors(cells, cellneighbors=6)
Morton-ordering within panels keeps logically adjacent cells adjacent in memory.
Subsetting the Geodesic Grid

- Can’t use start+count or strided NetCDF API (data is unstructured)
- Mask-based (arbitrary subset regions)
- Maintain whole cells (renumber topology variables)
- “subsetter” was the first pagoda command-line tool
NetCDF Operators

<table>
<thead>
<tr>
<th>NCO</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>ncks</td>
<td>subsetting, text display</td>
</tr>
<tr>
<td>ncra</td>
<td>record average</td>
</tr>
<tr>
<td>ncea</td>
<td>ensemble average</td>
</tr>
<tr>
<td>ncwa</td>
<td>weighted average</td>
</tr>
<tr>
<td>ncbo</td>
<td>binary arithmetic</td>
</tr>
<tr>
<td>ncflint</td>
<td>file interpolation</td>
</tr>
<tr>
<td>ncrcat</td>
<td>record concatenation</td>
</tr>
<tr>
<td>ncecat</td>
<td>ensemble concatenation</td>
</tr>
<tr>
<td>ncrename</td>
<td>rename vars/dims</td>
</tr>
<tr>
<td>ncatted</td>
<td>edit attributes</td>
</tr>
<tr>
<td>ncpdq</td>
<td>permute dimensions</td>
</tr>
<tr>
<td>ncap/ncap2</td>
<td>scripted processor</td>
</tr>
</tbody>
</table>

- Serial command-line tools
  - Use interactively
  - As part of a script
- Task-parallel versions
  - No parallel IO
- Extremely portable
- Not intended as library
  - “To my knowledge, though, only NCO programs use libnco”
  - Installs libnco but no headers
  - Potential for code reuse
## Pagoda Command-line Tools

<table>
<thead>
<tr>
<th>NCO</th>
<th>Pagoda</th>
</tr>
</thead>
<tbody>
<tr>
<td>ncks</td>
<td>pgsub</td>
</tr>
<tr>
<td>ncra</td>
<td>pgra</td>
</tr>
<tr>
<td>ncea</td>
<td>pgea</td>
</tr>
<tr>
<td>ncwa</td>
<td>(soon, v0.7)</td>
</tr>
<tr>
<td>ncbo</td>
<td>pgbo</td>
</tr>
<tr>
<td>ncflint</td>
<td>pgflint</td>
</tr>
<tr>
<td>ncrcat</td>
<td>N/A*</td>
</tr>
<tr>
<td>ncecat</td>
<td>N/A*</td>
</tr>
<tr>
<td>ncrename</td>
<td></td>
</tr>
<tr>
<td>ncatted</td>
<td></td>
</tr>
<tr>
<td>ncpdq</td>
<td></td>
</tr>
<tr>
<td>ncap/ncap2</td>
<td></td>
</tr>
</tbody>
</table>

- **Current version is 0.6**
- **Output verified against NCO**
  - Tested GCRM data
    - 8km resolution
  - Tested against ANL data
    - 1/8 degree CAM HOMME
    - 19 8.5GB files (15 variables each)
    - 19 2.5GB files (4 variables each)
  - Assumes NCO infallible
- **Scriptable (but not as simple)**
- *Don’t concatenate, aggregate*
pgra Strong Scaling on eureka.alcf.anl.gov

19 8.5GB files
- 15 variables each
- 1/8 degree CAM HOMME
- (lev=26,ncol=3110402)

Timings
- 1-core ncra 16:06
- 2-core pgra 17:41
- 4-core pgra 9:27
- 8-core pgra 5:12
- 16-core pgra 3:03

Thanks to Sherri Mickelson at ANL for the data and for performing these runs.
pgra Strong Scaling on eureka.alcf.anl.gov

19 2.5GB files
- 4 variables each
- 1/8 degree CAM HOMME
  (lev=26, ncol=3110402)

Timings
- 1-core ncra 4:54
- 2-core pgra 4:39
- 4-core pgra 2:56
- 8-core pgra 1:40
- 16-core pgra 1:04

Thanks to Sherri Mickelson at ANL for the data and for performing these runs.
Scalability depends on dimension order and data distribution:

- Using pnetcdf-1.2.0
- netcdf4/hdf5 may not be impacted
- (lev=26, ncol=3110402)
- Permuted i.e. (ncol, lev)
- Only distributing first dimension when smaller than number of cores

Thanks to Sherri Mickelson at ANL for the data and for performing these runs.
Strong Scaling on franklin.nersc.gov

- 12 GB per step, 24 steps
  - >288 GB
- Shown to scale up to 2K cores
- Shows that IO is a major bottleneck
- Write bandwidth nearly 5GB/s on franklin
- Our first optimization shows importance of efficient use of IO
<table>
<thead>
<tr>
<th>Function</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_File_read_all()</td>
<td>60.822</td>
</tr>
<tr>
<td>MPI_File_open()</td>
<td>4.804</td>
</tr>
<tr>
<td>MPI_Barrier()</td>
<td>3.971</td>
</tr>
<tr>
<td>MPI_File_write_all()</td>
<td>1.697</td>
</tr>
<tr>
<td>MPI_Finalize()</td>
<td>1.35</td>
</tr>
<tr>
<td>MPI_Bcast()</td>
<td>0.305</td>
</tr>
<tr>
<td>MPI_File_close()</td>
<td>0.292</td>
</tr>
<tr>
<td>MPI_File_set_view()</td>
<td>0.239</td>
</tr>
<tr>
<td>void ncrei::get_vara_all(int, int, const std::vector&lt;long long, std::allocator&lt;long long&gt;&gt;&amp;)</td>
<td>0.195</td>
</tr>
<tr>
<td>void GlobalArray::operate_add(int) [(GlobalArray.C) {418,1}−{426,1}]</td>
<td>0.194</td>
</tr>
<tr>
<td>MPI_Init()</td>
<td>0.134</td>
</tr>
<tr>
<td>MPI_Allreduce()</td>
<td>0.094</td>
</tr>
<tr>
<td>void pagoda::initialize(int *, char ***) [(Bootstrap.C) {36,1}−{64,1}]</td>
<td>0.07</td>
</tr>
<tr>
<td>void pagoda::finalize() [(Bootstrap.C) {67,1}−{87,1}]</td>
<td>0.026</td>
</tr>
<tr>
<td>void GlobalArray::create() [(GlobalArray.C) {124,1}−{152,1}]</td>
<td>0.021</td>
</tr>
</tbody>
</table>

- Using franklin.nersc.gov, 1K cores, 40 OSTs
- Variable: wind(time=8*17, edges=7864302, layers=26)
  - ~0.76 GB per timestep
  - >100 GB total (not including grid variables)
- Overwhelming majority of time spent in IO
  - Using non-blocking pnetcdf API to aggregate small IO
  - Working with IO library developers to optimize
Future

- “make it easy” – A higher level API
- New language bindings? Python (via Cython)? Fortran?
- Handle more conventions e.g. scale_value, add_offset
- Finish pgwa (ncwa)
- Grid interpolation, integrating ESMF code
- Use MPI comm. to read multiple files simultaneously
- Other operators e.g. pdq, ncap/ncap2
  - What if pnetcdf’s “CDF5” format is used?
- We need more users and user input on what’s needed
  - Already in use/testing by CSU, ANL, NCAR
  - In testing as replacement for NCO tools in nightly NCAR script
This research is supported by the U. S. Department of Energy’s Office of Science under the Scientific Discovery through Advanced Computing (SciDAC) program.
Thanks

16th Annual CESM Workshop’s Software Engineering Working Group

Parallel Analysis of GeOscience Data Status and Future

Jeff Daily

PI: Karen Schuchardt, in collaboration with Colorado State University’s Dave Randall et al

http://svn.pnl.gov/gcrm/wiki/pagoda
pagoda-dev@googlegroups.com