

Bringing Parallelism to Large-Scale Climate Data Analysis and Visualization

Robert Jacob

Climate and Earth System Modeling PI meeting

September 23rd, 2011

Washington, D.C.

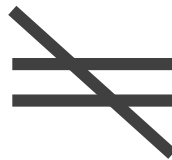


parvis

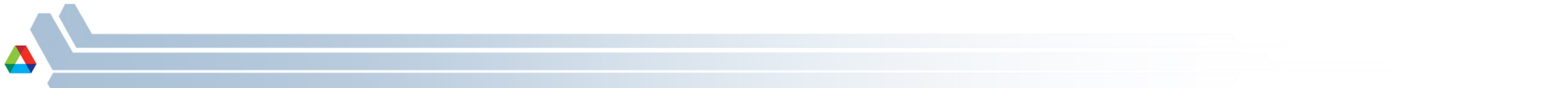
(Parallel Analysis Tools and New Visualization Techniques for Ultra-Large Climate Data Sets)

Motivation:

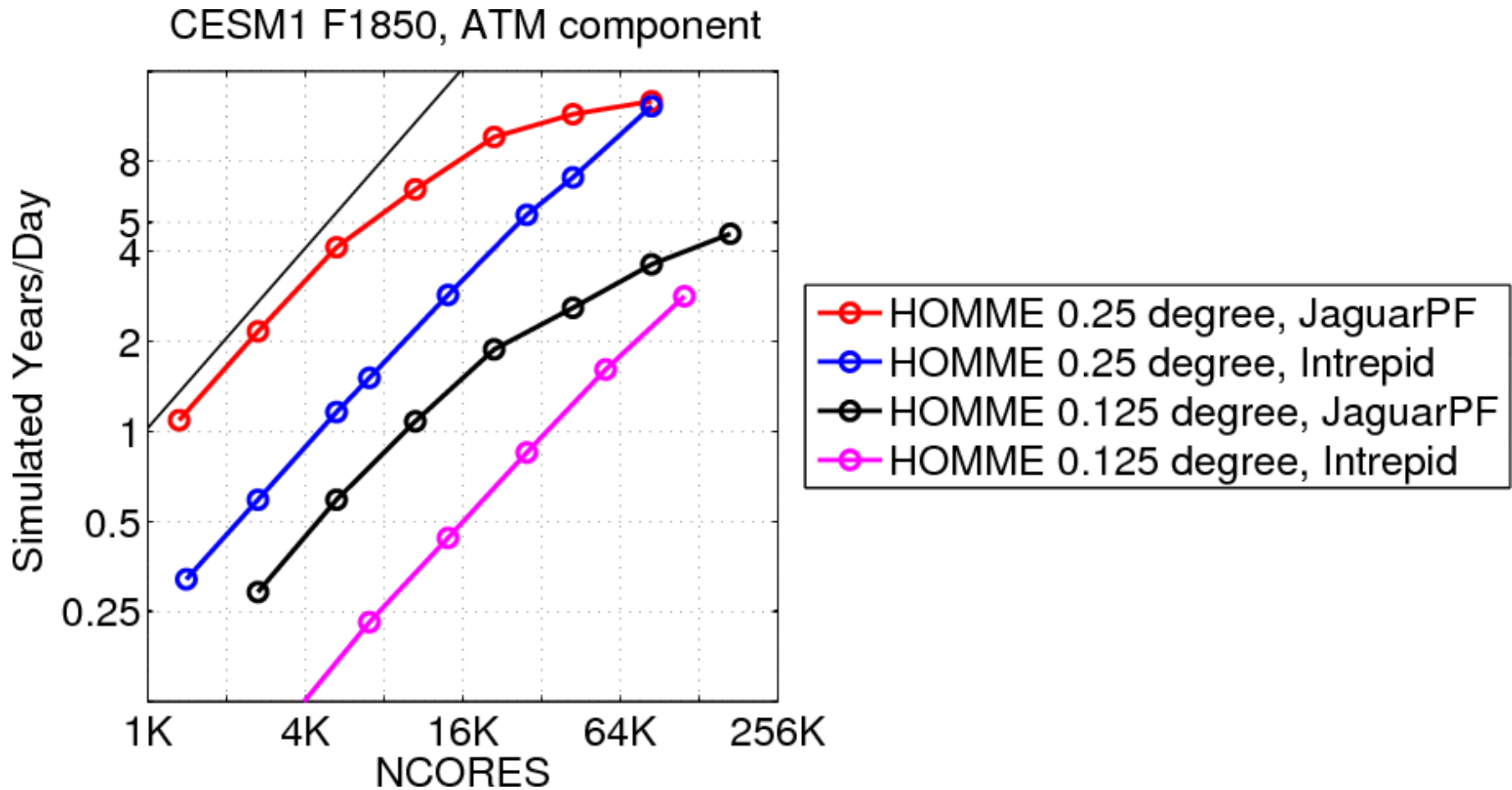
Ability to gain insight from current and future climate data sets



Capability of current tools



Climate models are now running on 100K cores...



From Mark Taylor, SNL



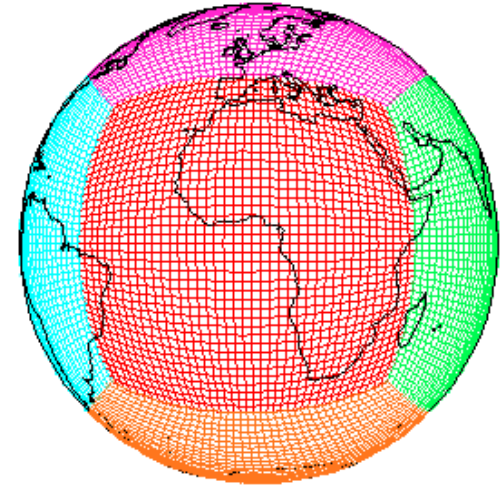
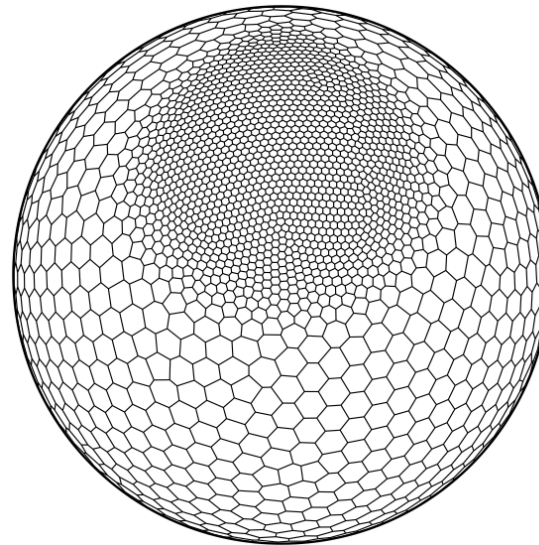
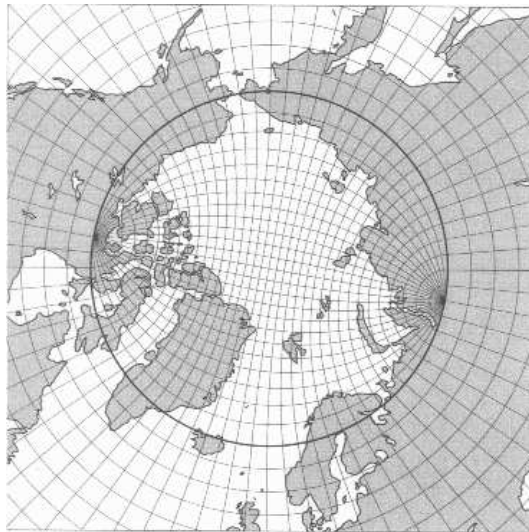
...and they are outputting lots of data.

- CAM-SE at 0.125 degrees
 - Single 3D variable: 616 MB
 - Single 2D variable: 25 MB
 - Single history file: 24 GB
 - 1 year of monthly output: 288 GB
 - 100 years of monthly: 28.8 TB

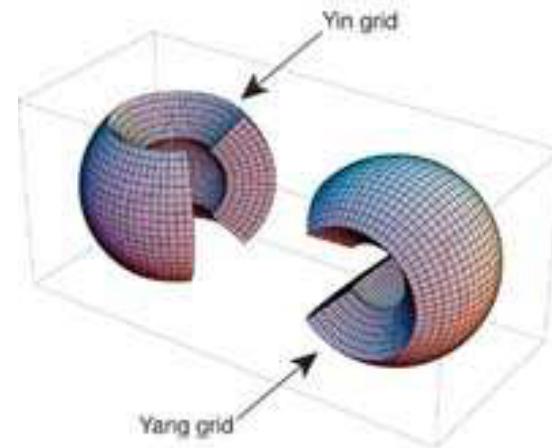
- CSU GCRM 4km horizontal, 100 levels
 - Single 3D variable (cell center): 16 GB
 - Single 3D variable (cell edge): 50.3 GB
 - Single history file: 571 GB
 - 1 year of monthly output: 6 TB
 - 100 years of monthly: .6 PB



and the data is coming out on new, unstructured or semi-structured grids.



All climate DAV tools require lat-lon grids for their internal analysis functions.

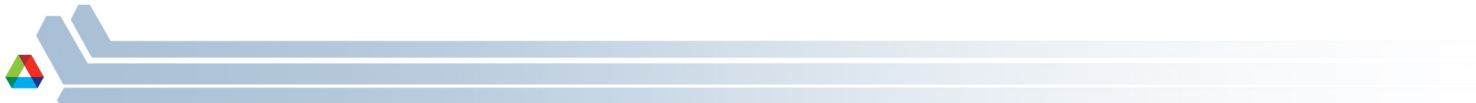




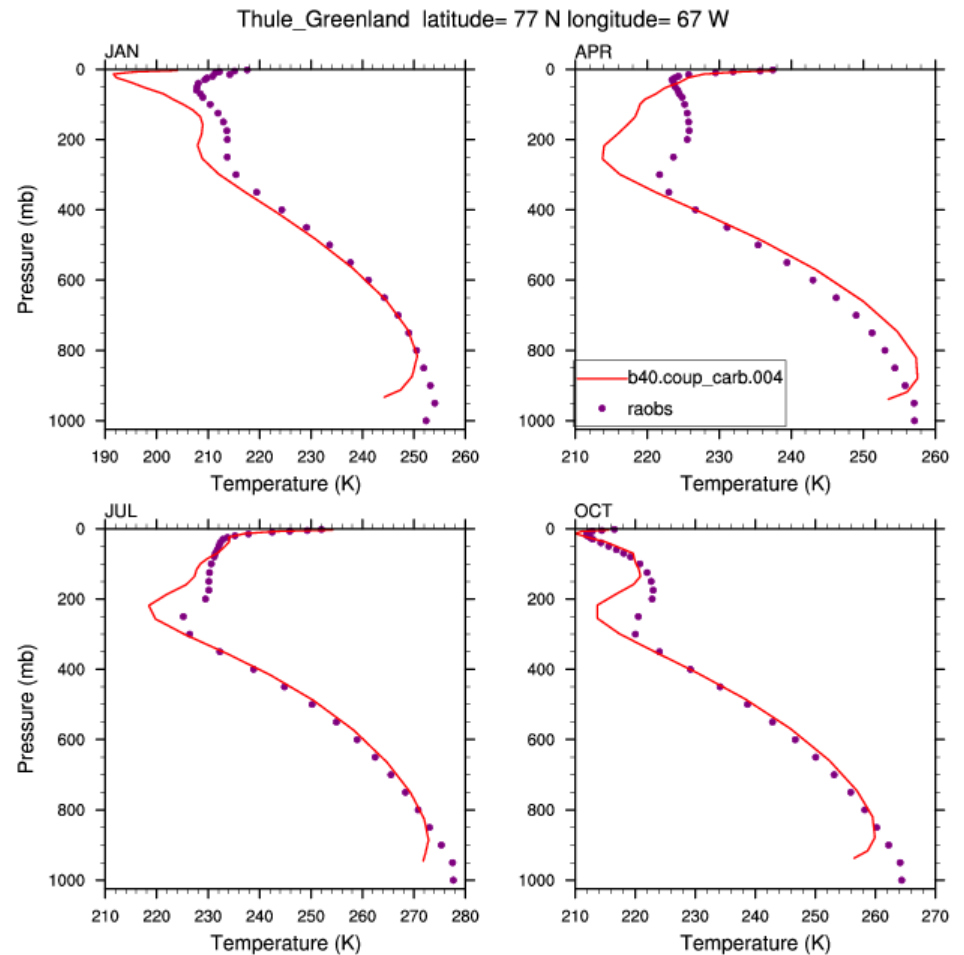
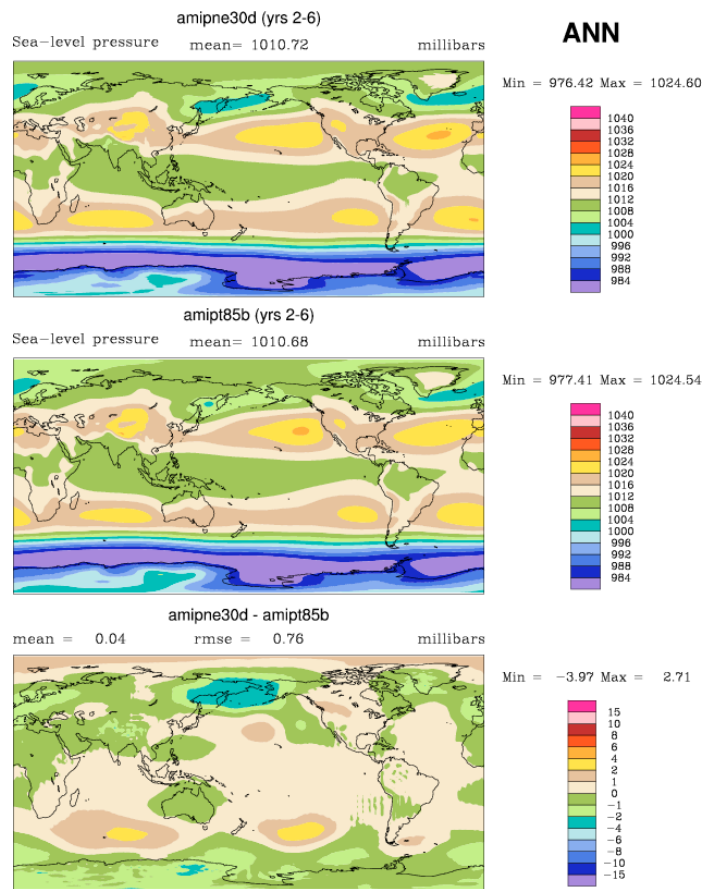
Existing Data Analysis and Visualization (DAV) tools have not kept up with growth in data sizes and grid types.

- NCAR Command Language (NCL)
 - Climate Data Analysis Tools (CDAT)
 - Grid Analysis and Display System (GrADS)
 - Ferret
- } No parallelism

ParVis will speed up data analysis and visualization through data- and task-parallelism AND natively support multiple grids.




ParVis philosophy: Insight about climate comes mostly from computationally undemanding (to plot) 2D and 1D figures.



Why? The atmosphere and ocean have a small aspect ratio; 10,000 km vs. 10 km.





(Philosophy cont'd) The problem is more in making the data for the figures (not directly from the models).

- Post-processing is an inextricable part of visualization of climate models.
- It is the post-processing where the introduction of parallelism could have the largest impact on climate science using current visualization practice
- BUT we should keep in mind changes/limitations in hardware (GPU co-processors, cloud computing, network bandwidth)





Approach

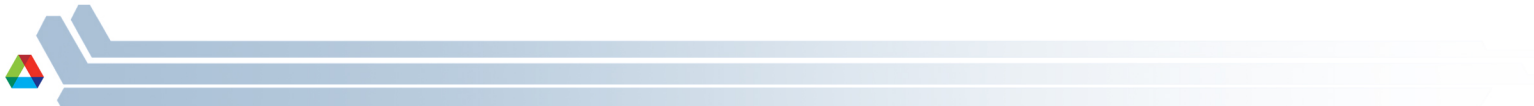
- Use existing tools to speed-up development.
- As much as possible, preserve well-established workflows for analyzing climate data, just speed them up.
- There is a problem *right now* so provide both immediate and long-term help
- Assemble a multi-disciplinary and multi-institutional team to carry out the work.





Personnel

- At Argonne:
 - Rob Jacob, Xiabing Xu, Jayesh Krishna, Sheri Mickelson, Tim Tautges, Mike Wilde, Rob Ross, Rob Latham, Jay Larson, Mark Hereld, Ian Foster
- At Sandia:
 - Pavel Bochen, Kara Peterson, Dennis Ridzal, Mark Taylor
- At PNNL
 - Karen Schuchardt, Jian Yin
- At NCAR
 - Don Middleton, Mary Haley, Dave Brown, Rick Brownrigg, Dennis Shea, Wei Huang, Mariana Vertenstein
- At UC-Davis
 - Kwan-Lu Ma, Jinrong Xie



ParCAL - Parallel Climate Analysis Library

- The main product from ParVis.
 - Data parallel C++ Library
 - Typical climate analysis functionality (such as found in NCL)
 - Structured and unstructured numerical grids
- Built upon existing tools
 - MOAB
 - Intrepid
 - MOAB and Intrepid have already solved the hard problem of how to represent and operate on structured and unstructured grids distributed over processors.
 - PnetCDF
 - MPI
- Will provide data-parallel core to perform typical climate post-processing currently done by either NCL or NCO.
- **Will be able to handle unstructured and semi-structured grids in all operations by building on MOAB and Intrepid. Will support parallel I/O by using PnetCDF.**





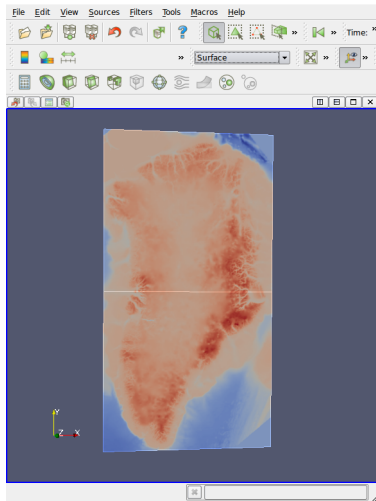
PNetCDF: NetCDF output with MPI-IO

- Based on NetCDF
 - Derived from their source code
 - API slightly modified
 - Final output is indistinguishable from serial NetCDF file
- Additional Features
 - Noncontiguous I/O in memory using MPI datatypes
 - Noncontiguous I/O in file using sub-arrays
 - Collective I/O
- Unrelated to netCDF-4 work

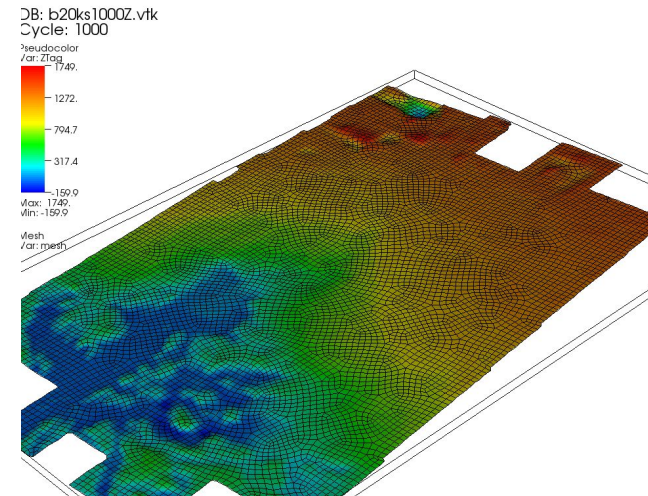


Mesh-Oriented datABase (MOAB)

- MOAB is a library for representing structured, unstructured, and polyhedral meshes, and field data on those meshes
- Uses array-based storage, for memory efficiency
- Supports MPI-based parallel model
 - HDF5-based parallel read/write on (so far) up to 16k processors (IBM BG/P)
- Interfaces with other important services
 - Visualization: ParaView, VisIt
 - Discretization: Intrepid (Trilinos package)
 - Partitioning / load balancing: Zoltan



Greenland ice bed elevation (in Paraview/MOAB)



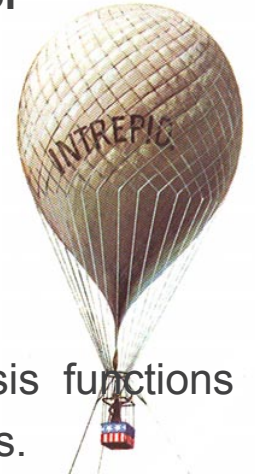
Jakobshavn ice bed (in VisIt/MOAB)



Intrepid *INteroperable Tools for Rapid dEvelopment
of compatible Discretizations*

A Trilinos package for compatible discretizations: a suite of stateless tools for

- Cell topology, geometry and integration
- Discrete spaces, operators and functionals on cell worksets
- Up to order 10 $H(\text{grad})$, $H(\text{curl})$ and $H(\text{div})$ FE bases on Quad, Triangle, Tetrahedron, Hexahedron, and Wedge cell topologies
- High quality cubature, e.g., positive weights only on Tri and Tet cells
- Flexible and extensible design: easy to add tools for new cell shapes and basis functions
- Common API for Finite Element, Finite Difference and Finite Volume methods.

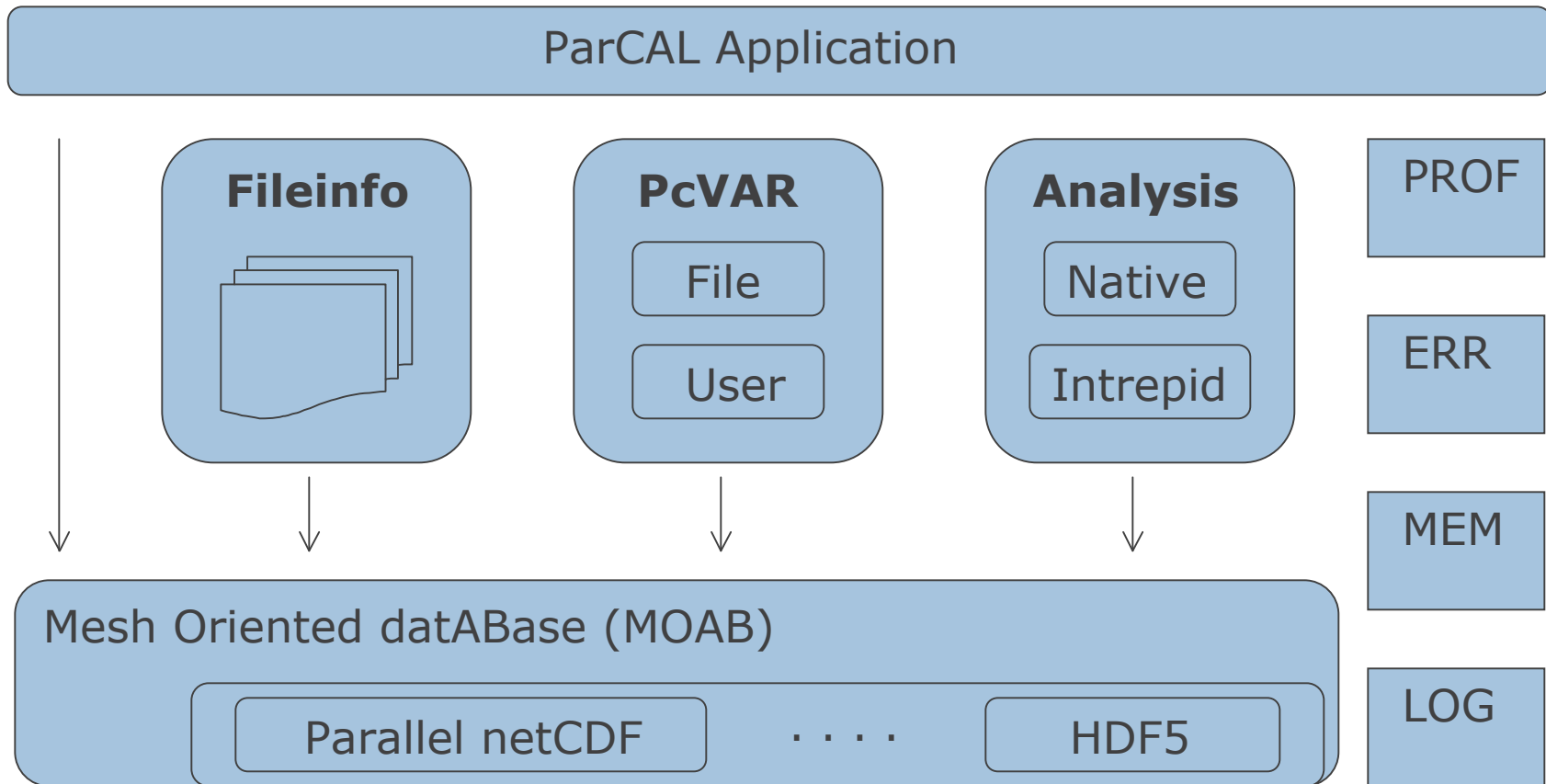


Design features

- Multi-indexed **Scalar** value is the “only” data type in Intrepid. Implemented as multi-dimensional array (MDA): contiguous data layout with multi-index access.
- optimized **multi-core** kernels; optimized **assembly**
- Can compute div, grad, curl on structured or unstructured grids maintained by MOAB.



ParCAL Architecture



ParCAL Architecture - contd

- Fileinfo
 - Abstraction of multiple files
- PcVAR
 - File Variables
 - User Variables
 - Read/write data through **MOAB**
- Analysis
 - Native: dim_avg_n, max, min (already implemented)
 - **Intrepid**
- MOAB
 - Parallel IO/Storage
- Misc utilities
 - MEM, ERR, LOG, PROF



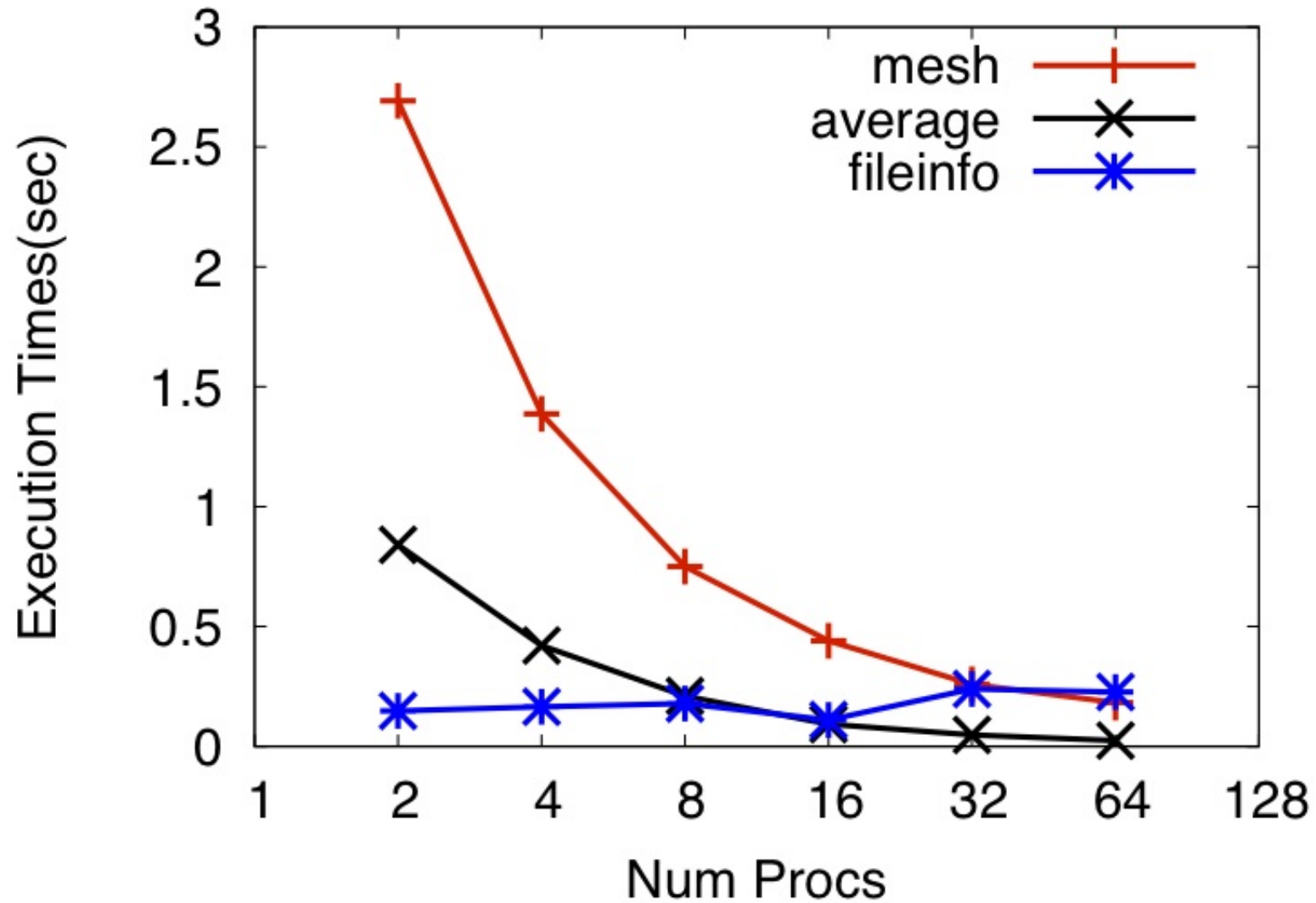
ParCAL test of dim_avg_n

- Input
 - 0.1 degree Atmosphere (1800x3600x26) up-sampled from a ¼ degree CAM-SE cubed sphere simulation
- Environment: Argonne “Fusion” cluster
 - OS: Red Hat Enterprise Linux 5.4
 - Compiler: Intel-11.1.064
 - Optimization Level: -O2
 - MPI: Mvapich2 1.4.1

NCO's ncea takes 90 seconds to calculate average of 3D T field.

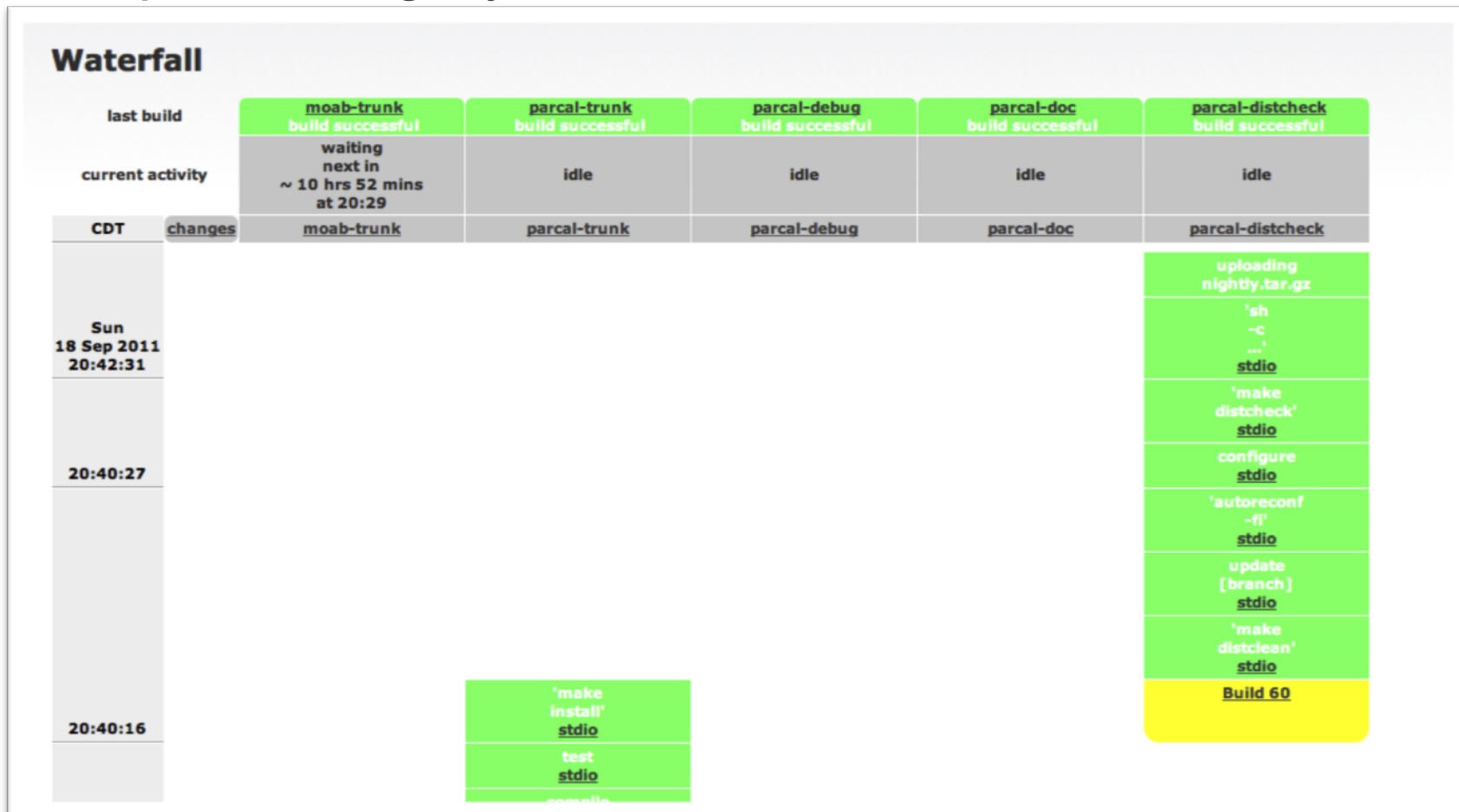


ParCAL dim_avg_n Performance



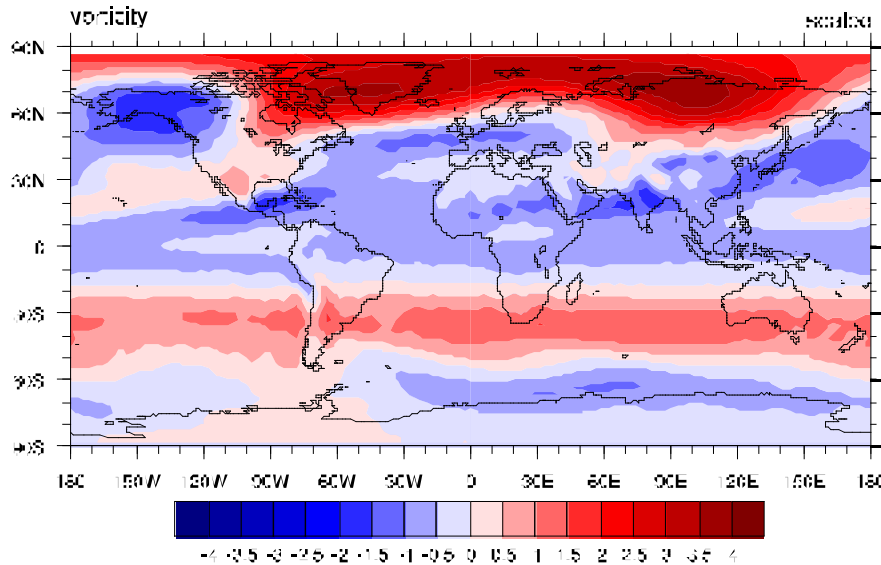
ParCAL is using software development best practices: Doxygen, Boost unit test, nightly build.

Snapshot of nightly build

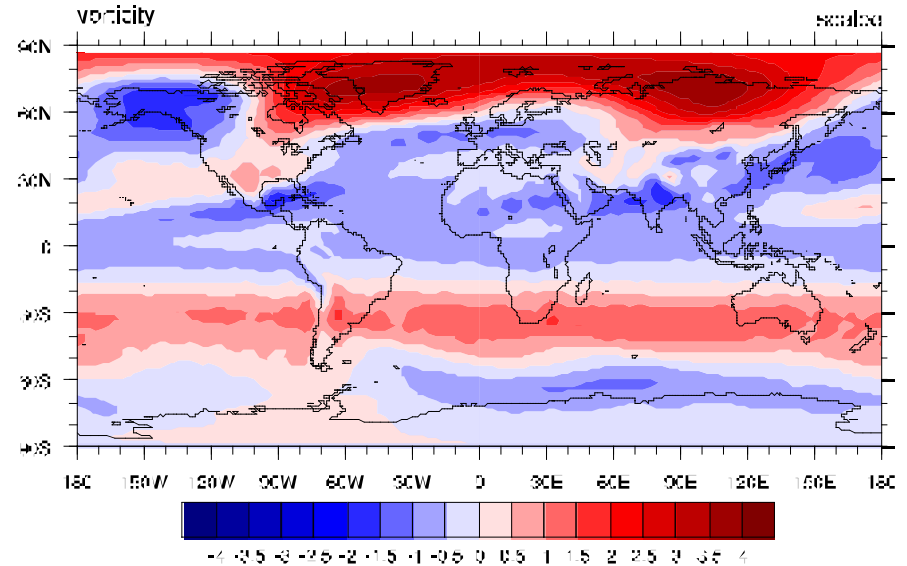


Calculating Vorticity with Intrepid

Intrepid



NCL (uv2vrG_Wrap)



- Calculated locally on each element
- Easily parallelizable
- Global data not required

- Uses spherical harmonics
- Requires global data

$$vorticity = \frac{1}{r \cos \phi} \frac{\partial v}{\partial \lambda} - \frac{1}{r} \frac{\partial u}{\partial \phi} + \frac{u}{r} \tan \phi$$

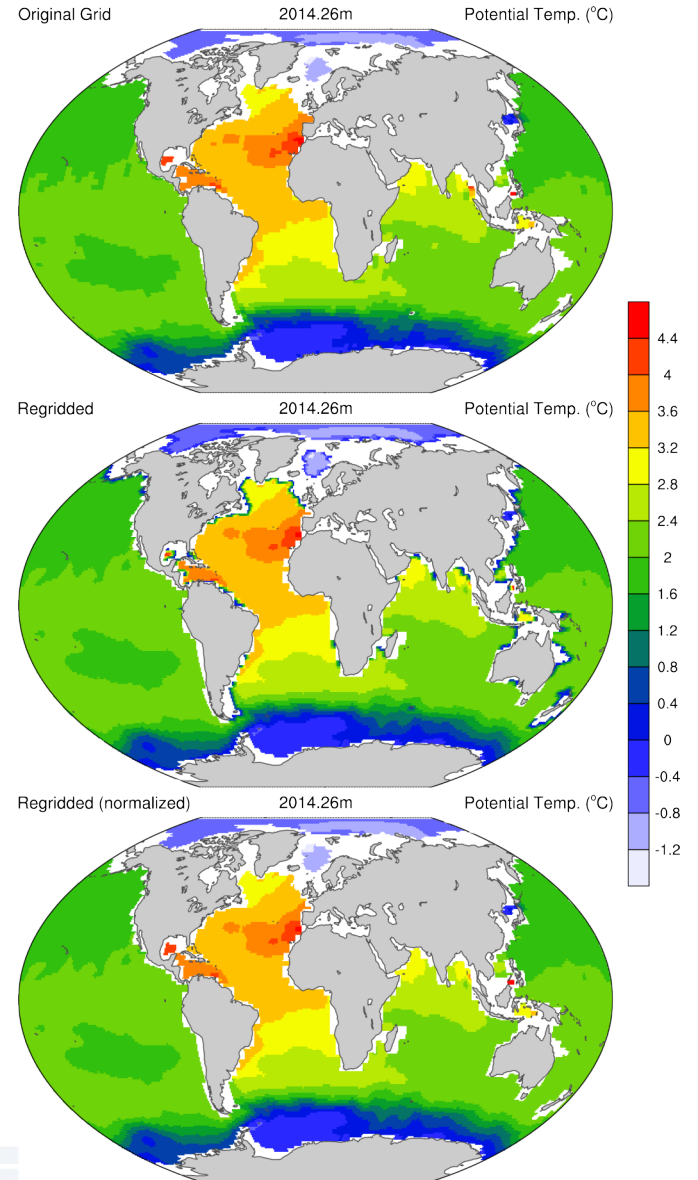


NCAR Command Language (NCL)

A scripting language tailored for the analysis and visualization of geoscientific data

1. Simple, robust file input and output
2. Hundreds of analysis (computational) functions
3. Visualizations (2D) are publication quality and highly customizable

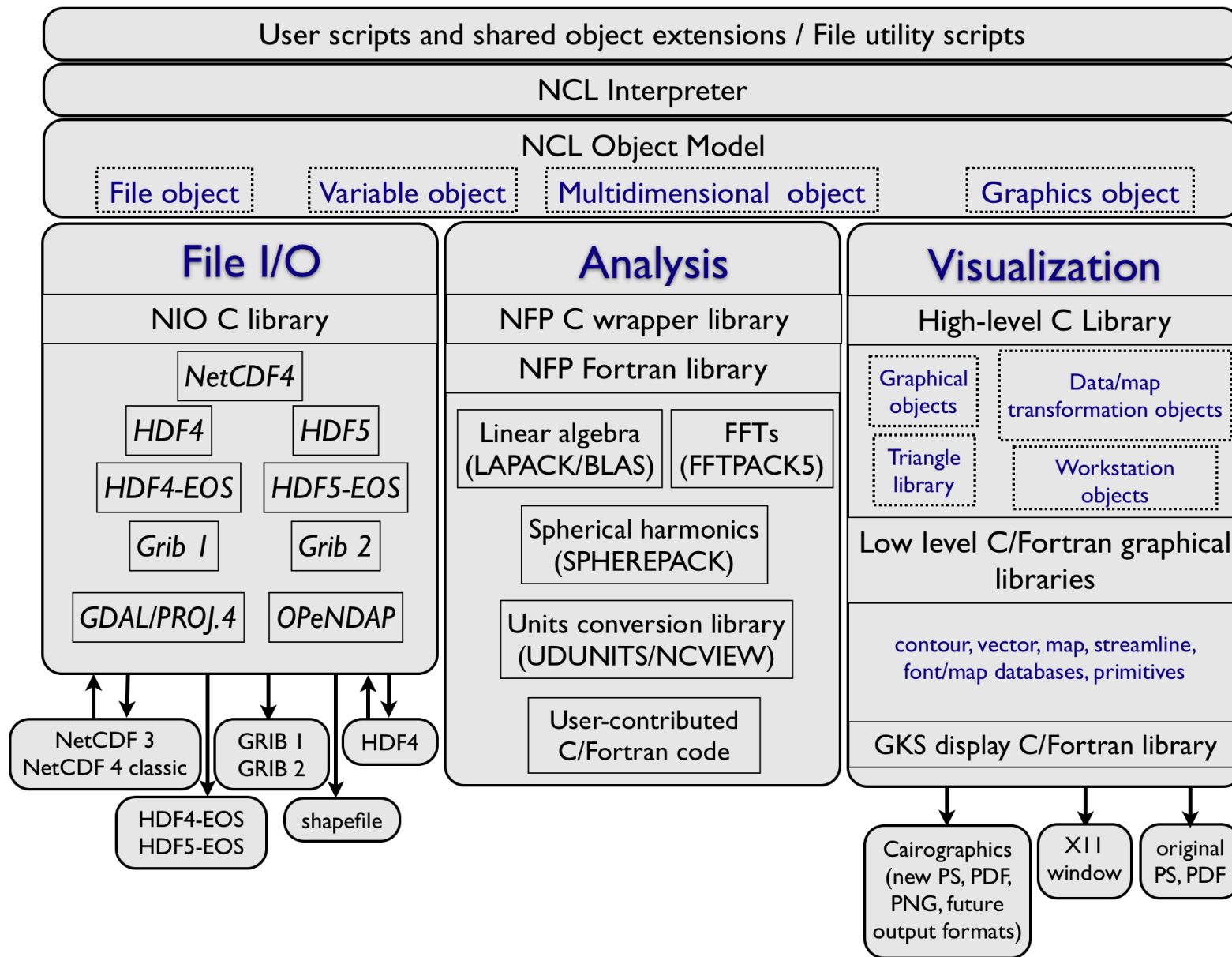
- **Community-based tool**
- Widely used by CESM developers/users
- UNIX binaries & source available, free
- Extensive website, **regular workshops**



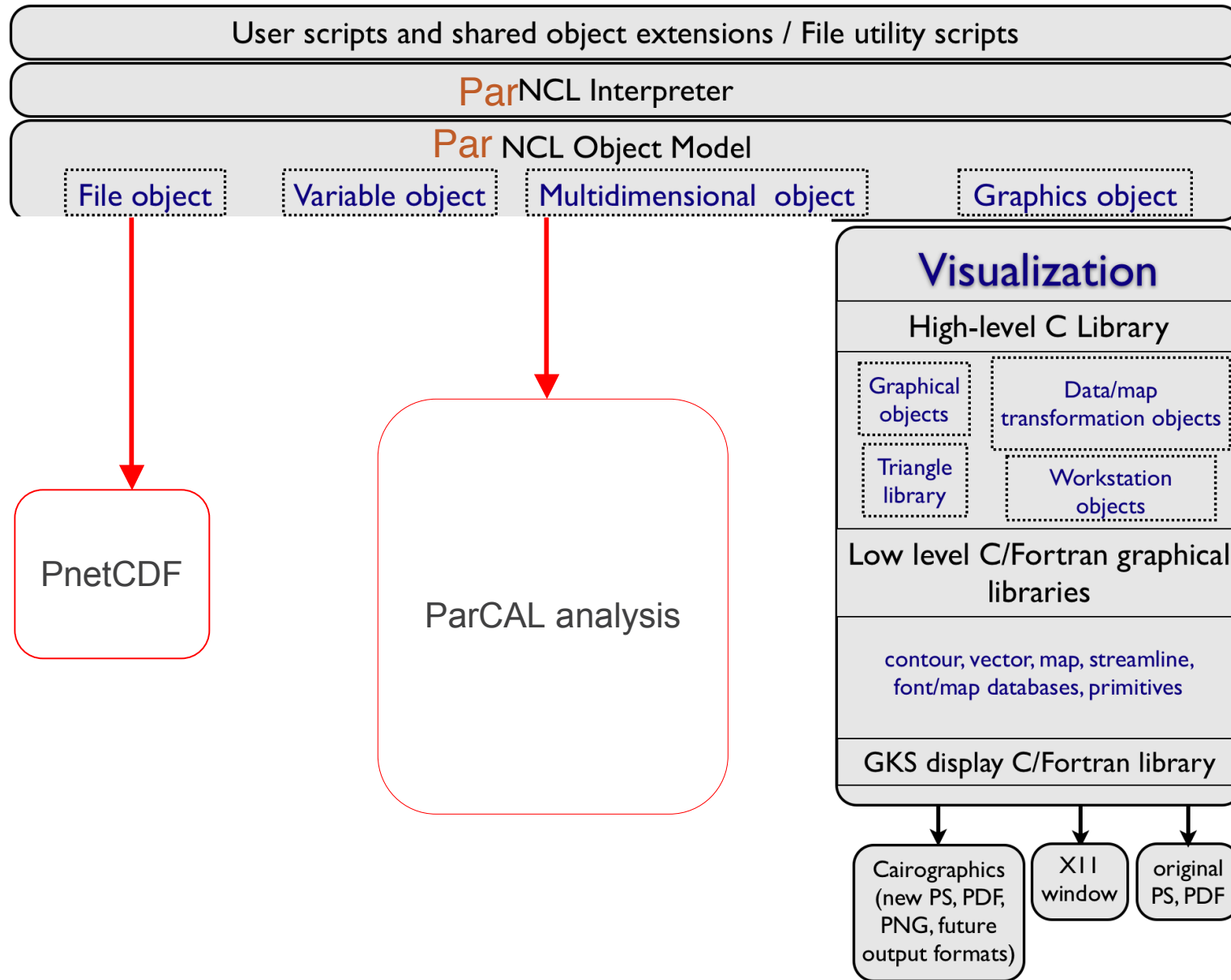
<http://www.ncl.ucar.edu/>



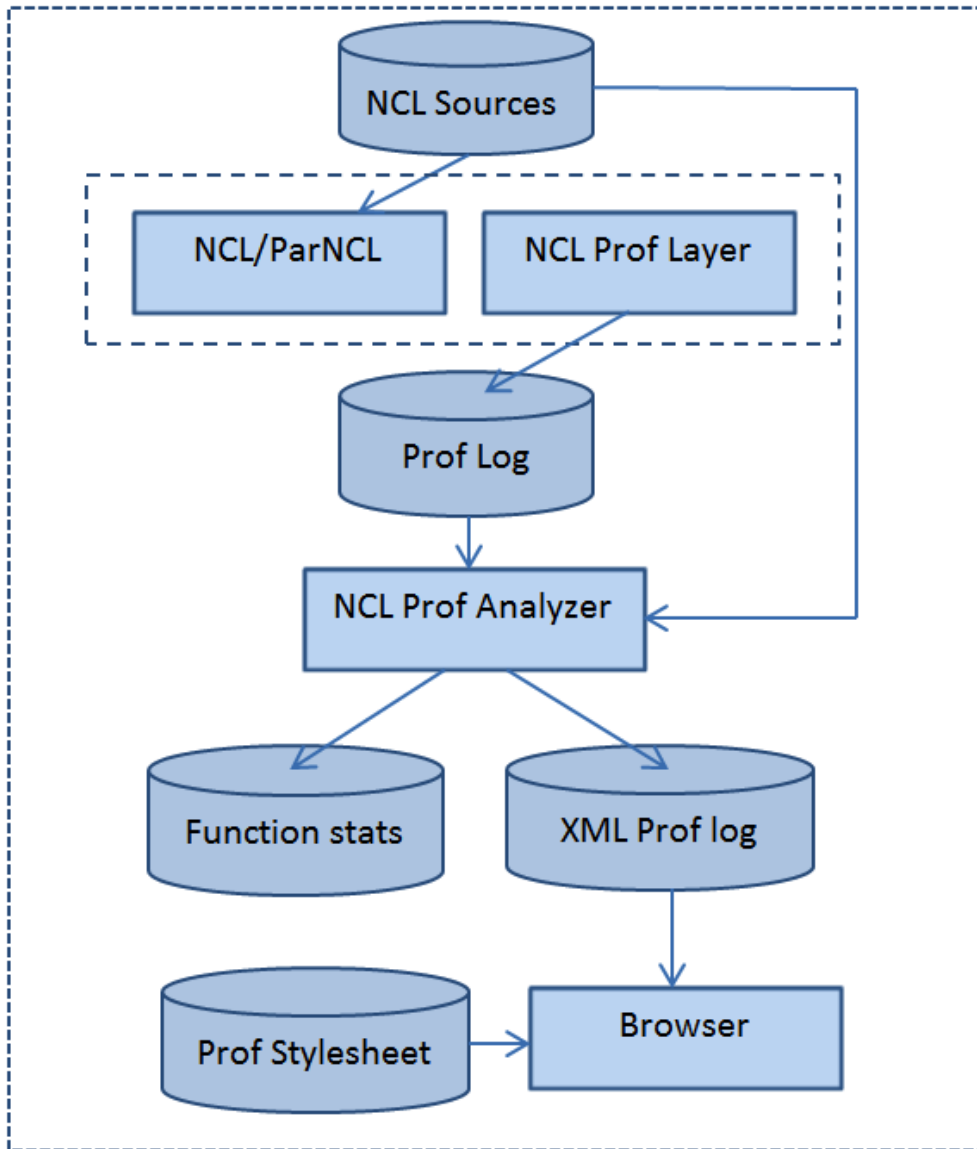
NCL architecture



ParNCL architecture



ParNCL Profiling Layer



- Manual modification of NCL script requires a lot of knowledge
 - Code semantics
 - NCL internals
- The profiling layer profiles scripts automatically
 - Records profile events at runtime
 - Useful for diagnostic pkgs
- Event analyzer script creates XML file containing usage statistics
 - NCL Script lines color coded based on time taken
 - Browser can be used to view XML

ParNCL Profiling Layer

NCL Line Statistics

avg_mfile.ncl

			load "\$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
3	2.31266021728516e-05s	0.000%	diri = "/fusion/group/climate/Parvis/atmos/b40.1850.track1.1deg.006/"
5	0.0303809642791748s	0.062%	fili = systemfunc("cd "+diri+" ; ls b40.1850.track1.1deg.006.cam2.h0.01*nc") ; 132 files
6	6.07967376708984e-05s	0.000%	setfileoption("nc","SuppressClose",False) ; user can tell NCL to alter default mode
8	10.6223649978638s	21.824%	f = addfiles(diri+fili, "r") ;
10	34.8738298416138s	71.649%	t = f[:]->T ; [time 132]x[lev 26]x[lat 192]x[lon 288]
12	3.0847008228302s	6.338%	Tavg=dim_avg_n(t,0) ;average t over the 0th dimension which is time.
14	0.061568021774292s	0.126%	printVarSummary(Tavg)

- XML profile log of script *avg_mfile.ncl* (serial run) viewed using Firefox
- Line no, time taken (T), % of total time & code is shown
- TM = median time calculated by the analyzer script
- Eg: Lines taking > 100 times the median time, like line 8 above, is colored red.

$T \leq TM/100$

$TM/100 < T \leq TM/10$

$TM/10 < T \leq TM$

$TM < T \leq TM*10$

$TM*10 < T \leq TM*100$

$T > TM*100$





ParNCL will be first application written with ParCAL

- A data-parallel version of the popular NCAR Command Language
- Will be backwards-compatible with current NCL scripts.
- Only parallelizes analysis functions of NCL
- Like NCL, will be freely available and open source.

Can already execute small NCL script that calls ParCAL's `dim_avg_n!`



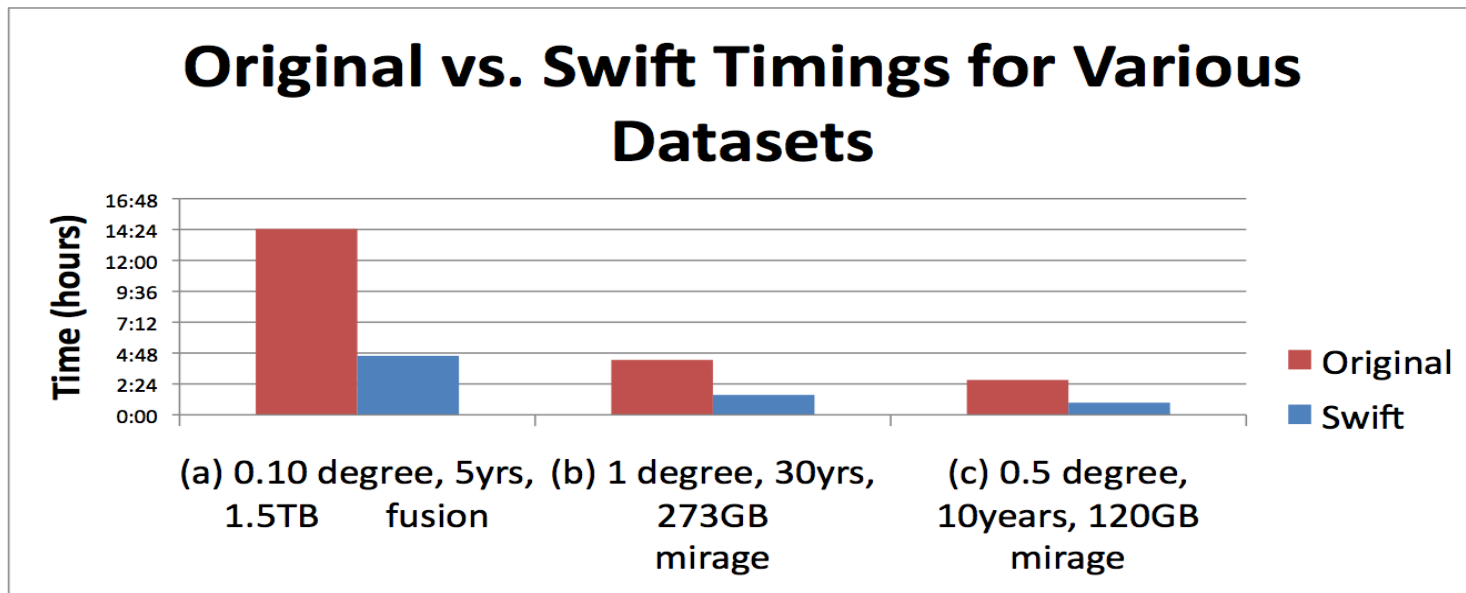
ParVis is providing immediate help with task-parallel versions of diagnostic scripts using *Swift*

- **Swift is a parallel scripting system for Grids and clusters**
 - for loosely-coupled applications - application and utility programs linked by exchanging files
- **Swift is easy to write:** simple high-level C-like functional language
 - *Small Swift scripts can do large-scale work*
- **Swift is easy to run:** a Java application. Just need a Java interpreter installed.
- **Swift is fast:** Karajan provides Swift a powerful, efficient, scalable and flexible execution engine.
 - *Scaling close to 1M tasks – .5M in live science work, and growing*
- **Swift usage** is growing:
 - *applications in neuroscience, proteomics, molecular dynamics, biochemistry, economics, statistics, and more.*



ParVis is rewriting diagnostic packages with Swift.

- Converting AMWG diagnostics package
 - Compares 2 CAM simulations or compares one CAM simulation to observational data
 - Controlled from a top level C-Shell script that calls NCO functions and NCL to create climate average files and over 600 plots that are browsable through a web interface
- Using Swift in both the “data” and “graphics” parts of diagnostics
- Swift version **now available** for download by anyone for beta testing.
- See **poster #8** for more information.



ParVis and Hardware

- Data Analysis Center's (Eureka, Lens) will continue to be a main venue for performing climate-model post-processing.
 - Eureka at ALCF
 - 100 compute nodes: each with (2) 2.0 GHz quad-core Xeon servers with 32 GB RAM
 - 200 NVIDIA Quadro FX5600 GPUs in 50 S4s
 - Memory: More than 3.2 terabytes of RAM
 - Peak Performance: More than 111 mostly single precision teraflops of computation use a fraction of electricity compared to alternative architectures.
 - Shares same disk as the big BlueGene/P
- Parallelism is a big step but other hardware concerns should be accounted for
 - Disk space and network bandwidth not growing as fast as data.
 - Cloud computing will play a role in providing capacity cycles
 - GPU's are everywhere. Should find role for 3D.



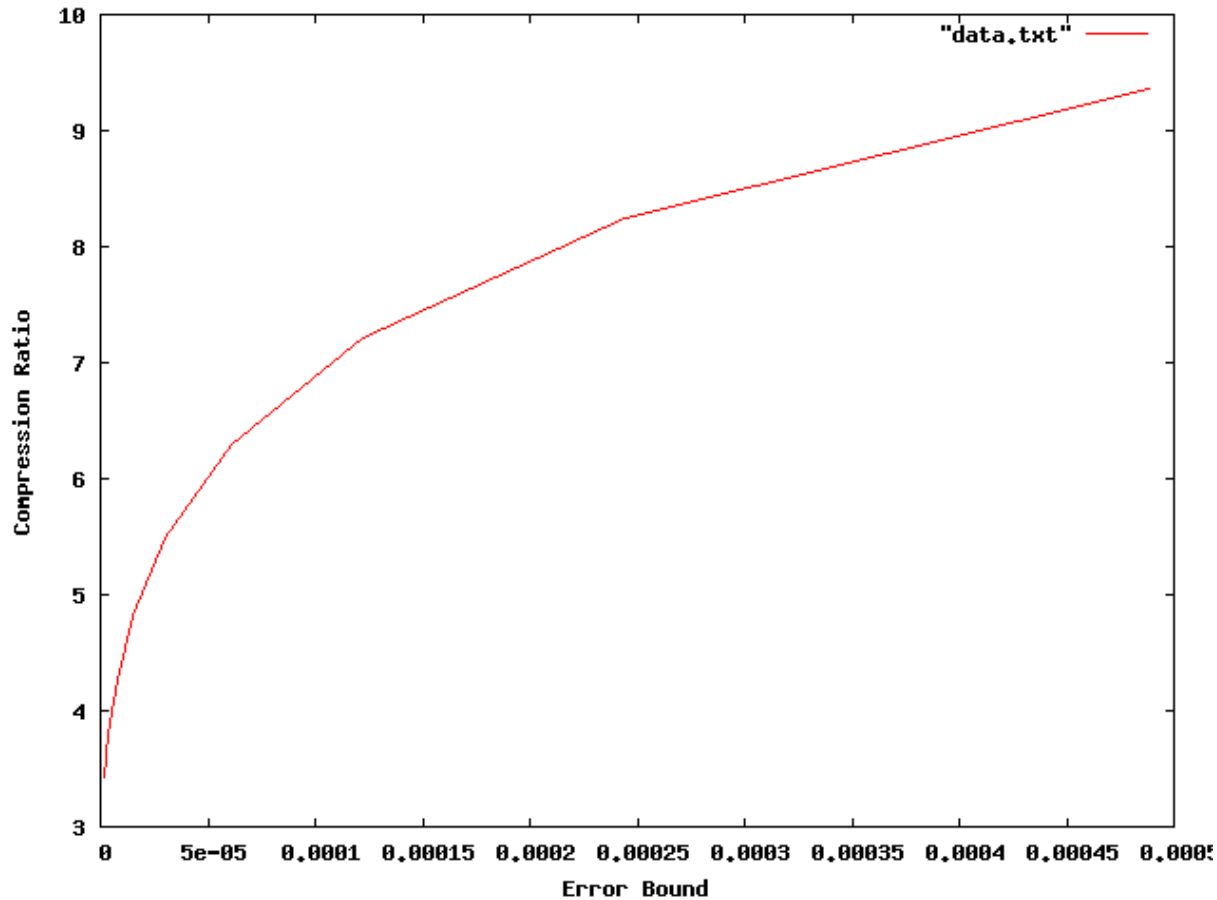


Compression can help with growing data sizes

- Completely random data can not be compressed without information loss but many climate output fields are smooth, not random.
- **Lossless** compression can reduce volume of the climate data without information loss
 - Reduce storage, memory, and network requirements to store, process, and transfer the data
 - Compression can potentially speedup analysis and visualization applications
 - Light weight and Integrate well with the applications
- **Lossy** compression can achieve higher compression ratio
 - May be appropriate for some applications.



Lossy Compression results



- Error for each value is bounded
- Preliminary results show that we can achieve a compression ratio around 10 when the error bound is 0.1%
- Further improvement is possible with improvement in the second part of our two-stage compression





Compression will help with data transfer.

Also helping - Globus Online

- Globus Online provides a managed file transfer service
 - features include: File Movement, Globus Connect, Performance Optimization, Error Retry, Monitoring, Endpoint Management, and Conditional Transfers and Integrity Checking.
- We used Globus Online to transfer files for testing from NCAR to Argonne
 - 144 528MB files (75GB total) in 10.5 minutes
- Many endpoints supported by default (NERSC, Teragrid, ...)
 - Can add your own by downloading app to your laptop or workstation (Mac, Linux, Windows)
- Get free account at www.globusonline.org



Using cloud computing for climate analysis with MapReduce

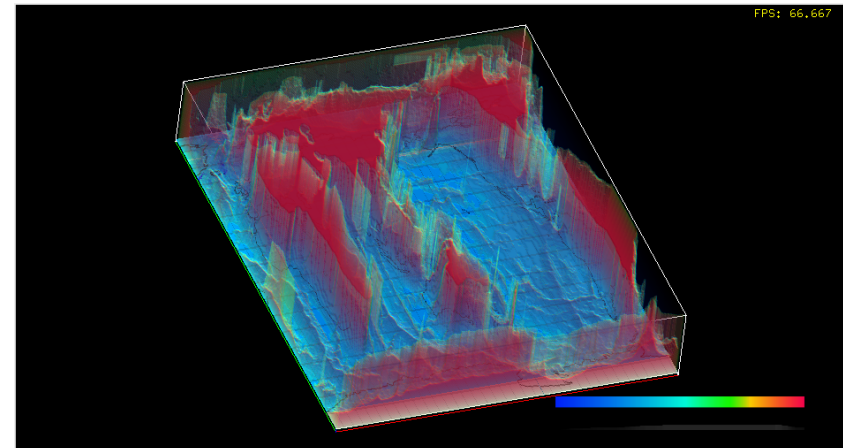
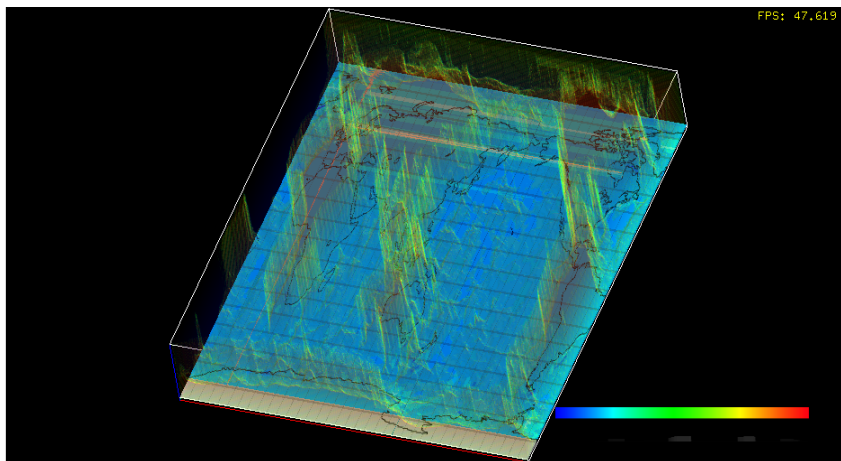
- MapReduce: Functional Programming pattern popularized by Google
 - User writes two functions **Map ()** and **Reduce ()** :
 - **Map ()** - processes **key/value** pairs to generate intermediate **key/value** pair representation (e.g., <filename,field> to <location, field value>)
 - **Reduce ()** - merges all intermediate values sharing the same **key** (e.g., compute average, et cetera)
 - Example: ensemble upper-air *T* average across large numbers of history files, **Map ()** emits <(lat,lon,lev), *T*>, **Reduce ()** computes average *T* at each location
 - Highly scalable to large data volumes (>1TB) and thousands of work units
 - Apache Hadoop open-source system including distributed file system
 - Native API in Java
 - Streaming Hadoop supports Map/Reduce executable images
- Map and Reduce archetypes have been identified for a large class of analyses
 - Individual **Map ()** / **Reduce ()** functions reusable
- Implementation of netCDF input reader class and Map/Reduce functions in progress.



ParVis taking advantage of all that GPU power...

- Designing and evaluating an interactive correlation analysis and visualization tool (ICAV).
- GPU acceleration and parallelization of all correlation and visualization calculations.
- Interactive user interface.
- Developing hooks between (Par)NCL and ICAV to enable superimposing/linking NCL and ICAV views.

Figures below: volume rendering by ICAV, map by NCL

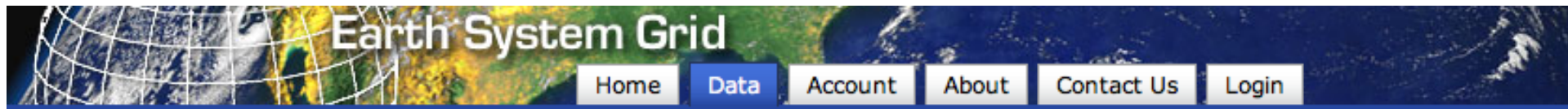


Future work...(lots of coding)

- Continue to develop ParCAL.
 - Integrate Intrepid functionality.
 - Implement parallel versions of more common climate analysis functions.
 - Parallel NetCDF writer for MOAB
- Expand test coverage of ParNCL
- Extend Swift approach to other diagnostic packages.
 - Currently working on Ocean. **Almost done converting ocean diagnostics from IDL to NCL.**
- Compression
 - Bring techniques in to PnetCDF (and thus in to ParCAL)
 - Experiment with multi-layer compression
 - Increase throughput with pipeline parallelism between read of compressed data and decompression
- Expanding ICAV capabilities
 - Develop rendering solutions for 3D hexagonal grid data.
 - Enhance ICAV UI



Further Future: ParNCL with ESG via LAS



Advanced Search

Search: for:

To conduct a search, select a category from the pull down menu and/or enter free text into the the text box.

Search Categories

- Project
 - < [Any Project](#)
 - CMIP5
- Institute
 - < [Any Institute](#)
 - NCAR
- + Model
- + Experiment
- + Frequency
- + Product
- + Realm
- + Variable

Total Number of Results: 27

1-10 of 27 results | [11-20](#) | [21-27](#)

- [project=CMIP5, model=NCAR Community Climate System Model, CCSM version 4, experiment=historical, time_frequency=mon, modeling_realm=atmos, ensemble=r1i1p1, version=1](#)
 Data Center: ESG-NCAR
Access: [NCAR LAS](#)
- [project=CMIP5, model=NCAR Community Climate System Model, CCSM version 4, experiment=historical, time_frequency=mon, modeling_realm=atmos, ensemble=r2i1p1, version=1](#)
 Data Center: ESG-NCAR
Access: [NCAR LAS](#)
- [project=CMIP5, model=NCAR Community Climate System Model, CCSM version 4, experiment=historical, time_frequency=mon, modeling_realm=atmos, ensemble=r3i1p1, version=1](#)
 Data Center: ESG-NCAR
Access: [NCAR LAS](#)
- [project=CMIP5, model=NCAR Community Climate System Model, CCSM](#)

Download datasets. Or expression (example fi



Live Access Server (LAS) with NCL Backend Service

Live Access Server About LAS OPeNDAP (F

NCAR ESG-LAS Confluence Server (v7.3.1)

Choose dataset **Update Plot** Set plot options Animate Compare Google Earth Show Values Export to Desktop Application Save As ... Link To ... Print

cmip5.output1.NCAR.CCSM4.historical.mon.atmos.Amon.r3i1p1.ts.1.aggregation / cmip5.output1.NCAR.CCSM4.historical.mon.atmos.Amon.r3i1p1.ts.1.aggregation - Subset 1

ts

89 N
0 E 3 W
90 S

MAPS

- Latitude-Longitude

HOVMOLLER PLOTS

- Longitude-Time
- Latitude-Time

LINE PLOTS

- Time Series
- Longitude
- Latitude

Date: Jan 1850

[Apply analysis](#)

ts K

60N
30N
0
30S
60S
90S

0 30E 60E 90E 120E 150E 180 150W 120W 90W 60W 30W

231.514 240.778 250.043 259.307 268.571 277.835 287.1 296.364

- Can drop in ParNCL for NCL



What you can do...

- Let us know:
 - Where bottlenecks are in your analysis workflow. What NCL commands take too long or need too much memory?
 - What kind of post processing analysis would you like to do but can't?
 - When do you have to interpolate to some other grid as part of your analysis?
- Attend our session at 2011 Fall AGU (Thursday morning, Dec 8th)
 - IN06: Challenges in Analysis and Visualization of Large Earth Science Data Sets.
 - Conveners: Robert Jacob, Dean Williams and Wes Bethel
- Check the website: trac.mcs.anl.gov/projects/parvis
 - Subscribe to ParVis announcement mailing list: parvis-ann



parvis

