Software Engineering and the Parallel Climate Analysis Library (ParCAL)

Robert Jacob and Xiabing Xu
Mathematics and Computer Science Division
Argonne National Laboratory

SEA Software Engineering Conference

February 23rd, 2012

Boulder, CO

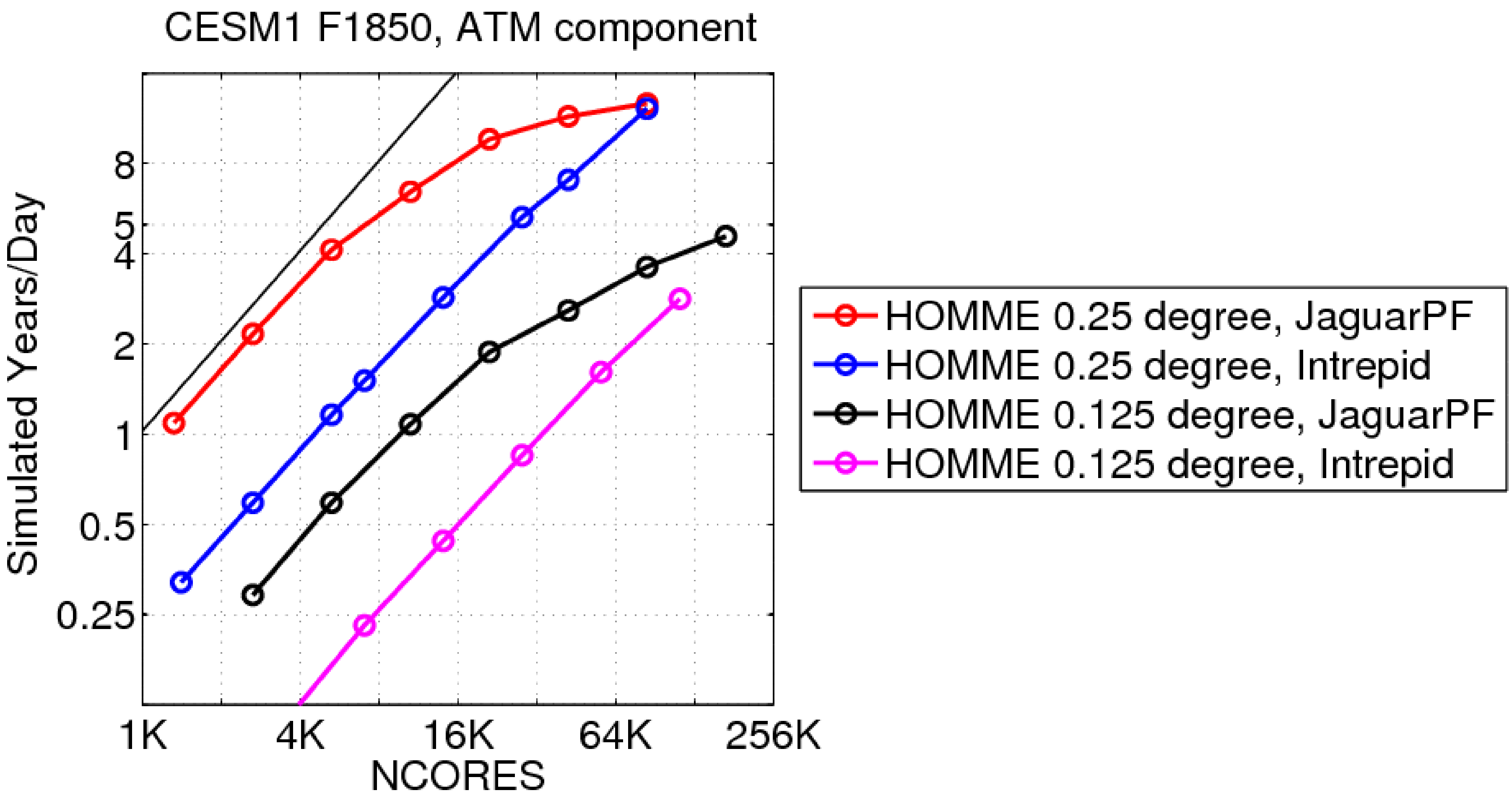
parVis

(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 $\neq$ Capability of current tools

Climate models are now running on 100K cores…

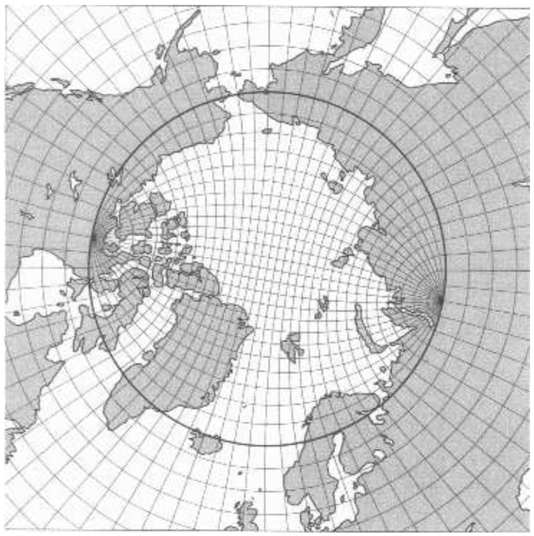


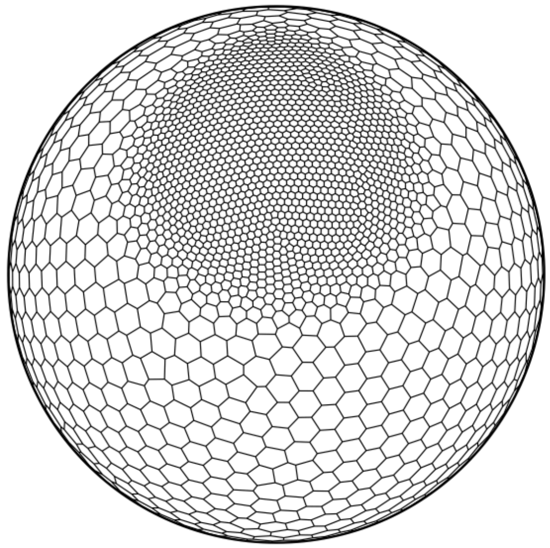
From Mark Taylor, SNL

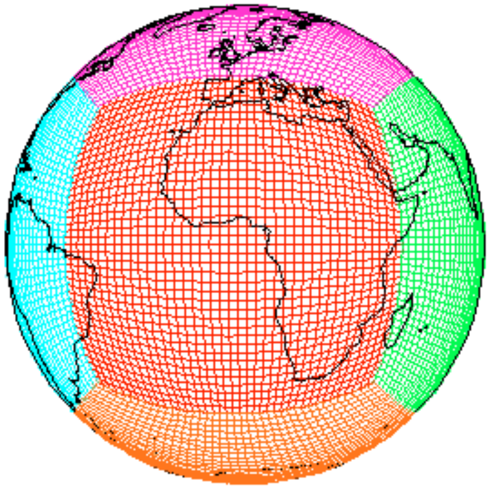
…and they are outputting lots of data.

- 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

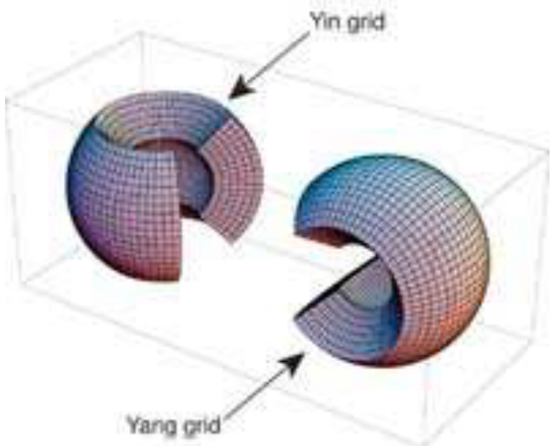
and the data is coming out on new, unstructured or semi-structured grids.







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

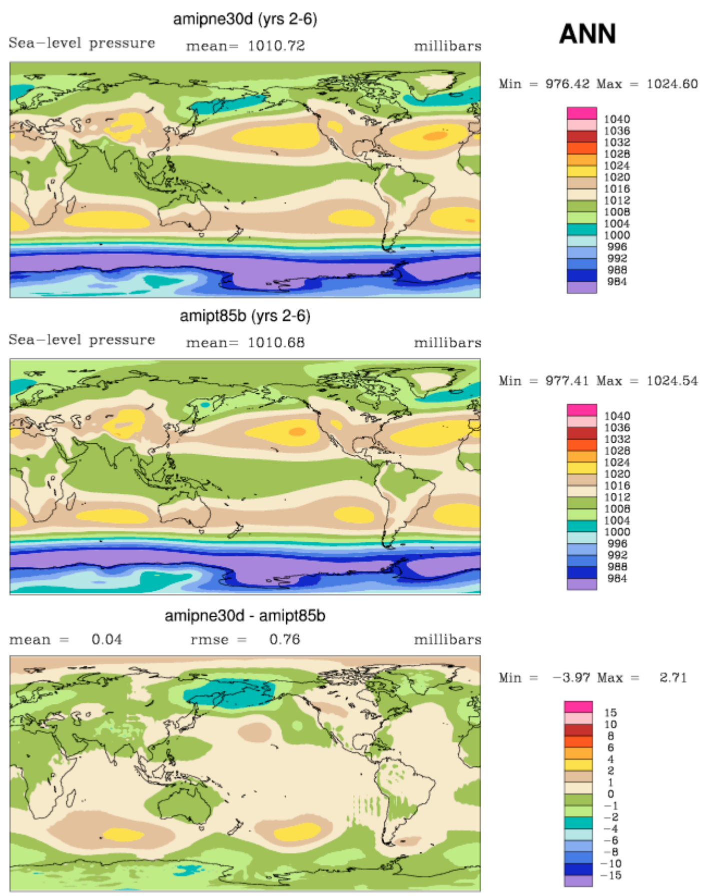


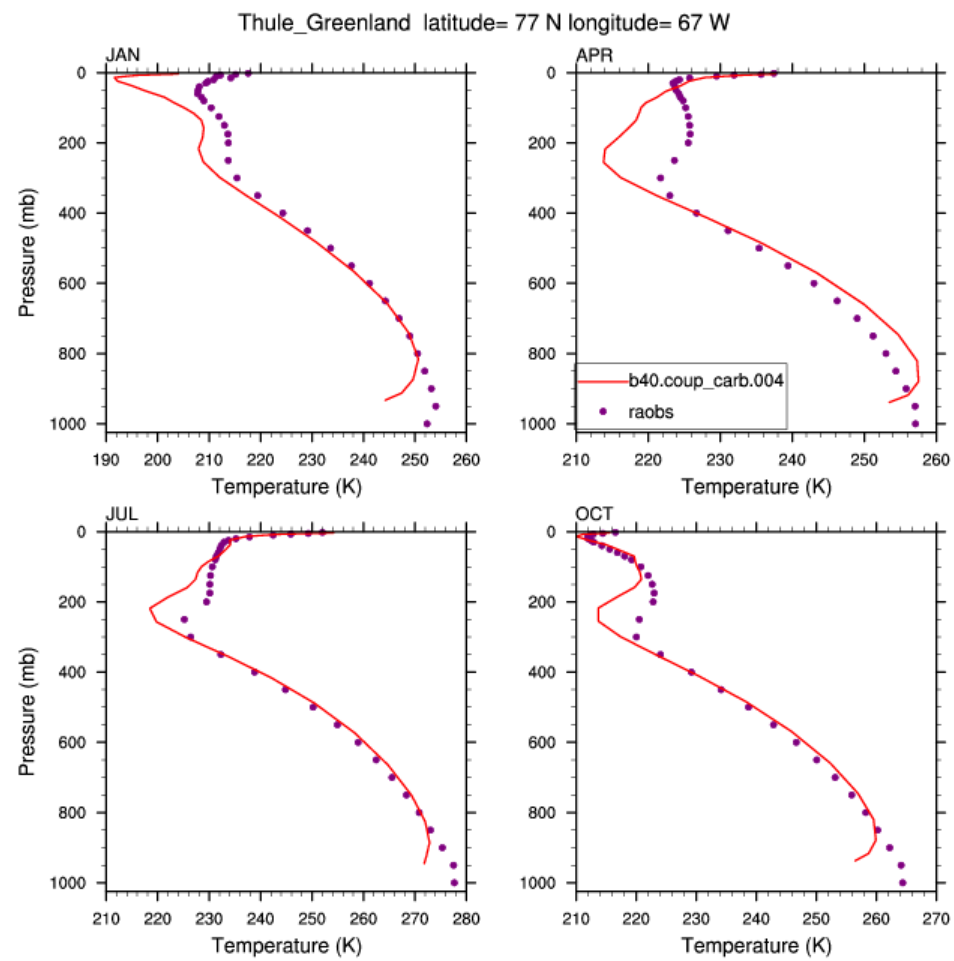
Existing Data Analysis and Visualization (DAV) tools have not kept up with growth in data sizes and grid types.

- NCAR Command Language (NCL)
- Climate Data Analysis Tools (CDAT)
- Grid Analysis and Display System (GrADS)
- Ferret

No parallelism

ParVis 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 climate viz problem for ultra-large data is mostly in post-processing the data for the figures.

(data used does not come out directly from the models)

- *Post-processing* is an inextricable part of *visualization* of climate model output.

- It is the post-processing where the introduction of parallelism could have to largest impact on climate science using current visualization practice

parVis

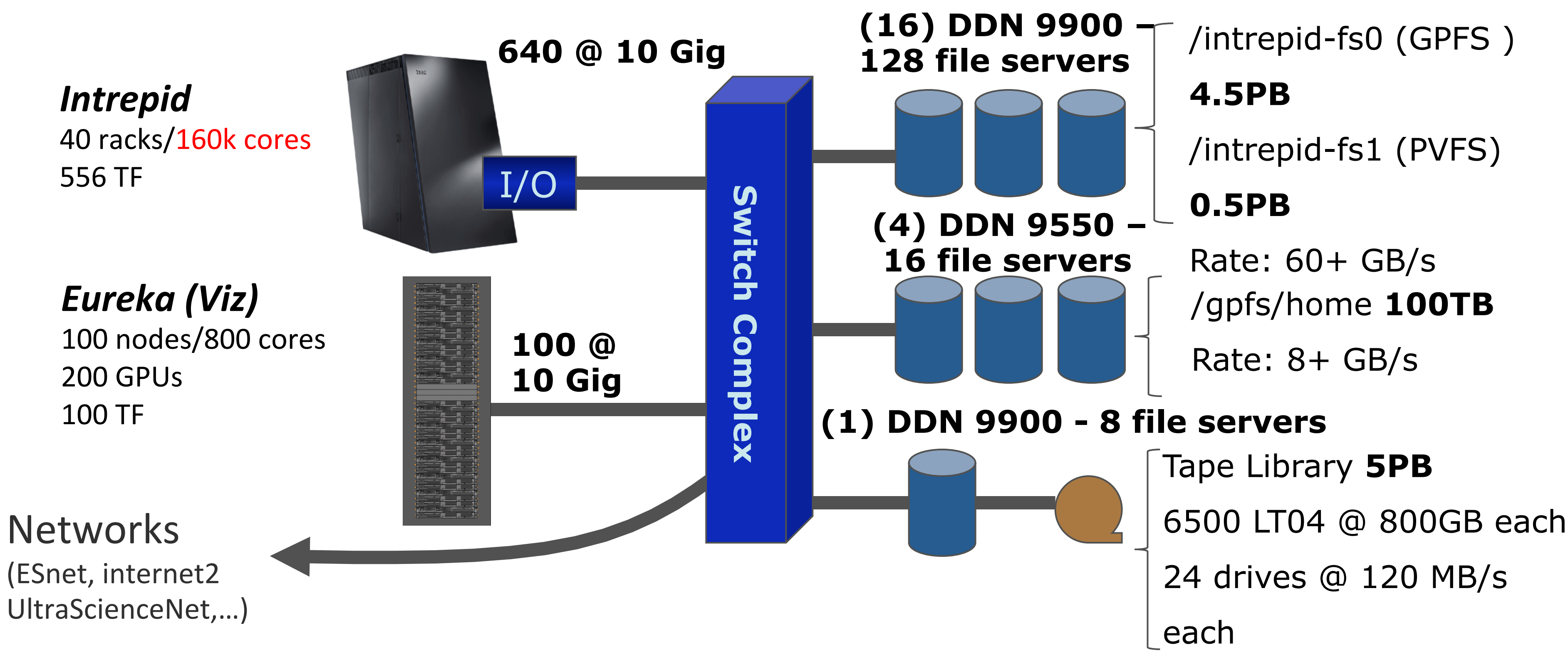
- Two-pronged approach:
  - Provide immediate help by speeding up current workflows with Task-parallelism
  - Build a new data-parallel library for performing climate analysis on both structured AND unstructured grids.

Parvis Hardware Model

- Data Analysis Center's (connected to large compute centers) will be a main venue for performing climate-model post-processing.
  - Eureka connected to Intrepid at ALCF (Argonne National Lab)
  - Lens connected to JaguarPF at NCCS (Oak Ridge National Lab)
  - DAV system connected to Yellowstone at NWSC (NCAR)

- Your desktop.
  - No longer any such thing as a single-processor workstation
  - Your desktop/laptop has 4, 8, 16 or more cores. Will increase. Your DAV tools likely not taking advantage of extra cores.

Argonne Leadership Computing Facility Hardware Layout



Parvis will provide 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. Can use multiple cores on your desktop/laptop.
- Have rewritten CESM Atmospheric Model Working Group diagnostics with Swift
  - "The AMWG diagnostics package produces over 600 postscript plots and tables in a variety of formats from CESM (CAM) monthly netcdf files."
- Timings with 10 years of 0.5 degree CAM data comparing with observations:
  - Original csh version on one core: 71 minutes
  - Swift and 16 cores: 22 minutes

ParCAL - Parallel Climate Analysis Library

- The main features
  - Data parallel C++ Library
  - Typical climate analysis functionality (such as found in NCL)
  - Structured and unstructured numerical grids
- Built upon existing Libraries
  - MOAB
  - Intrepid
  - PnetCDF
  - MPI
- Will provide data-parallel core to perform typical climate post-processing functions.

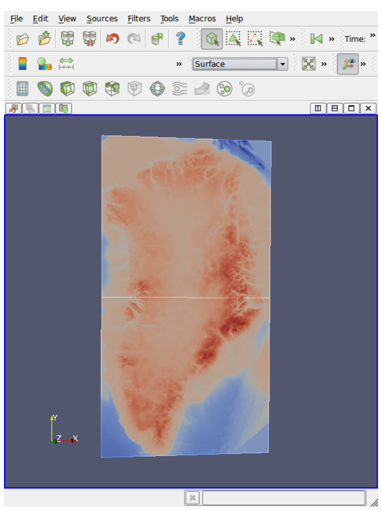
- **Will be able to handle unstructured and semi-structured grids in all operations by building on MOAB and Intrepid. Will support parallel I/O by using PnetCDF.**

PNetCDF: NetCDF output with MPI-IO

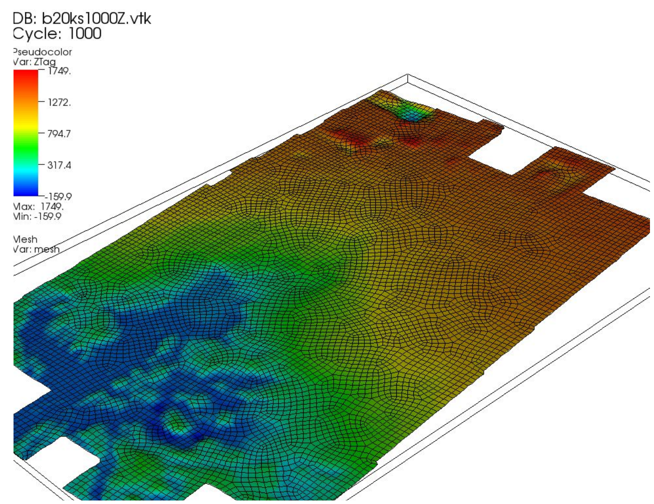
- Based on NetCDF
  - Derived from their source code
  - API slightly modified
  - Final output is indistinguishable from serial NetCDF file
- Additional Features
  - Noncontiguous I/O in memory using MPI datatypes
  - Noncontiguous I/O in file using sub-arrays
  - Collective I/O
- Unrelated to netCDF-4 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 (software, not to be confused with the BlueGene/P at ALCF)

INteroperable Tools for Rapid dEveloPment of compatIble Discretizations

A **Trilinos package** for compatible discretizations:

- An extensible library for computing operators on discretized fields
- Will compute div, grad, curl on structured or unstructured grids maintained by MOAB!

When fully deployed (~2012) will provide

- support for **more discretizations** (FEM, FV and FD)
- optimized **multi-core** kernels
- optimized **assembly** (R. Kirby)

Developers: P. Bochev, D. Ridzal, K. Peterson, R. Kirby

http://trilinos.sandia.gov/packages/intrepid/

Software Engineering Practices in developing ParCAL (starting almost from scratch)

- Version Control System
  - Subversion (SVN)
- Automatic Documentation System
  - Doxygen
- Automatic Configuration
  - Autotools (Autoconf, Automake, libtool)
- Unit Tests
  - Boost Unit Testing Framework
- Automatic Nightly Tests
  - Buildbot System
- Project Management, Issue and Bug Tracking
  - Trac-based system

Common Tools

Version Control with svn

- svn co https://svn.mcs.anl.gov/repos/parvis/parcal/trunk
- Repository Layout (directories)
  - Branches: for different branch development
  - Tags: for different release versions
  - Trunk: main development repository
- Our sysadmins (at MCS) make it easy to set up an svn repo.

Doxygen In-Source Documentation System

- Support various programming languages (C, Java, Python, Fortran etc.)
- Automatic generation of on-line webpages and off-line reference manual
- Can be configured to document code structure (class, subclass)

Unit Test

Boost Test Library - Unit Test Framework

C++ framework for unit test implementation and organization

Very widely used.   www.boost.org

Features:

- Simplify writing test cases by providing various testing tools
- Organize test cases into a test tree
- Relieve users from messy error detection, reporting duties and framework runtime parameters processing

Boost Test Tools:
You write a test program and call Boost functions

- Test Organization
  - BOOST_AUTO_TEST_SUITE (test_name)
  - BOOST_AUTO_TEST_CASE (test1)
  - BOOST_AUTO_TEST_CASE (test2)
  - BOOST_AUTO_TEST_SUITE_END ()
- Predefined Macros
  - BOOST_WARN(), BOOST_CHECK(), BOOST_REQUIRE()
- Pattern Matching
  - Compare against a golden log file
- Floating Point comparison
  - BOOST_CHECK_CLOSE_FRACTION (left-value, right-value, tolerance-limit)
- Runtime Parameter Options (handled by the Boost main())
  - --log_level : specify the log verbosity
  - --run_test : specify test suites and test cases to run by names

Buildbot System

Goal:

- Automate the compile/test cycle to validate code changes
- trac.buildbot.net

Features:

- Run builds on a variety of platforms
- Arbitrary build process: handles projects using C, C++, Python etc.
- Status delivery through web pages, email etc.
- Track builds in progress, provide estimated completion time
- Specify the dependency of different test configurations
- Flexible configuration to test on a nightly basis or on every code changes
- Debug tools to force a new build

Nightly Build

Nine different configurations:

mpich, openmpi, gcc, intel, warning test, document generation etc.

Nightly build homepage

- http://crush.mcs.anl.gov:8010/

Doxygen API:

- http://crush.mcs.anl.gov:8010/parcal-docs/

Doxygen PDF

- http://crush.mcs.anl.gov:8010/refman.pdf

Nightly build from trunk

- http://crush.mcs.anl.gov:8010/parcal-nightly.tar.gz

Nightly Test Screenshot

Project Management with Trac

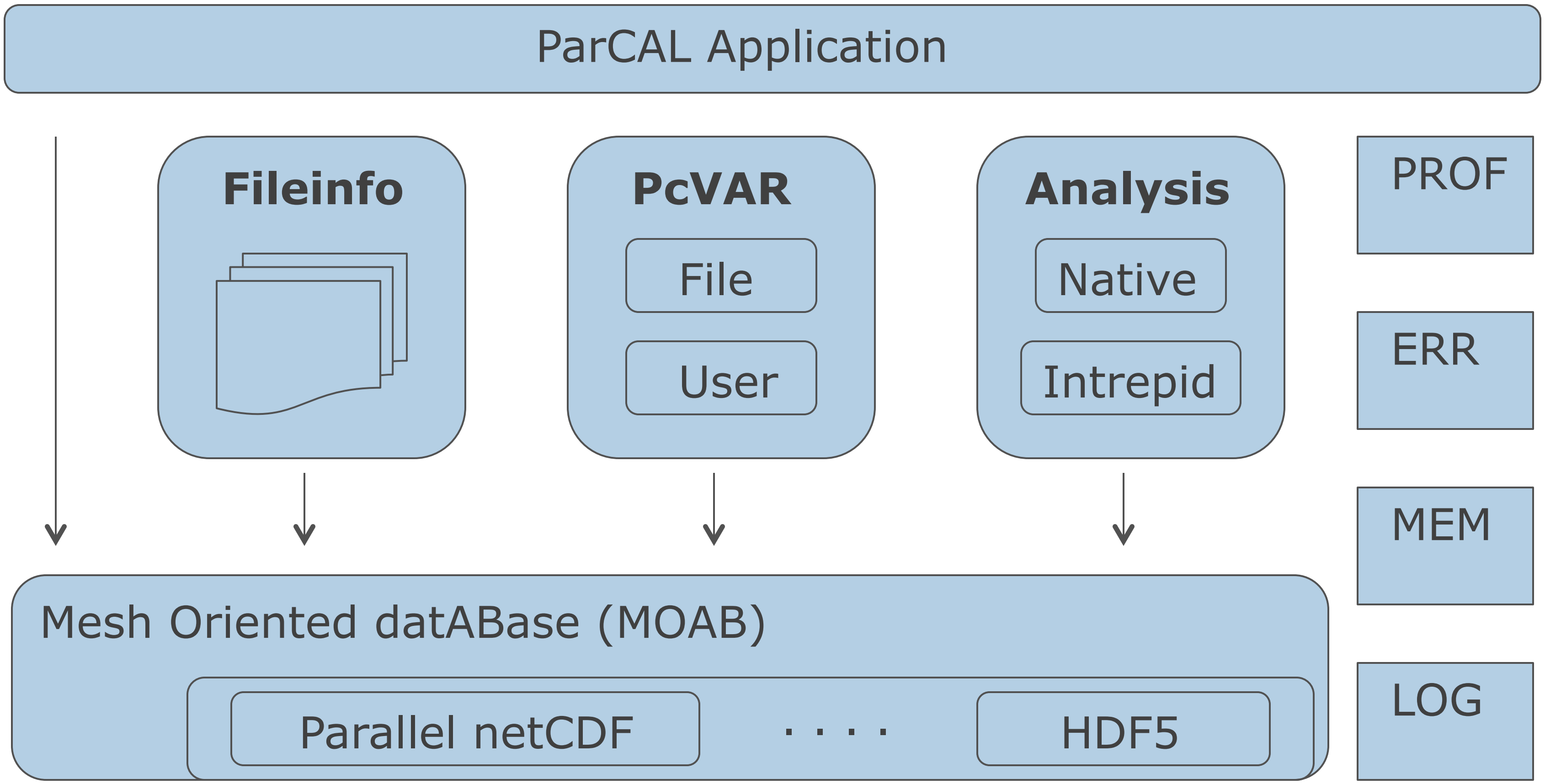
Web-based software project management system. Like svn, easy setup provided by MCS sysadmins

http://trac.edgewall.org/

- Built-in Wiki System
- Connects with svn repo
  - Browsing source code
  - Viewing changes to source code
  - Viewing change history log
- Ticket Subsystem
  - Using tickets for project tasks, feature requests, bug reports etc.

http://trac.mcs.anl.gov/projects/parvis

ParCAL Architecture



ParCAL Architecture - contd

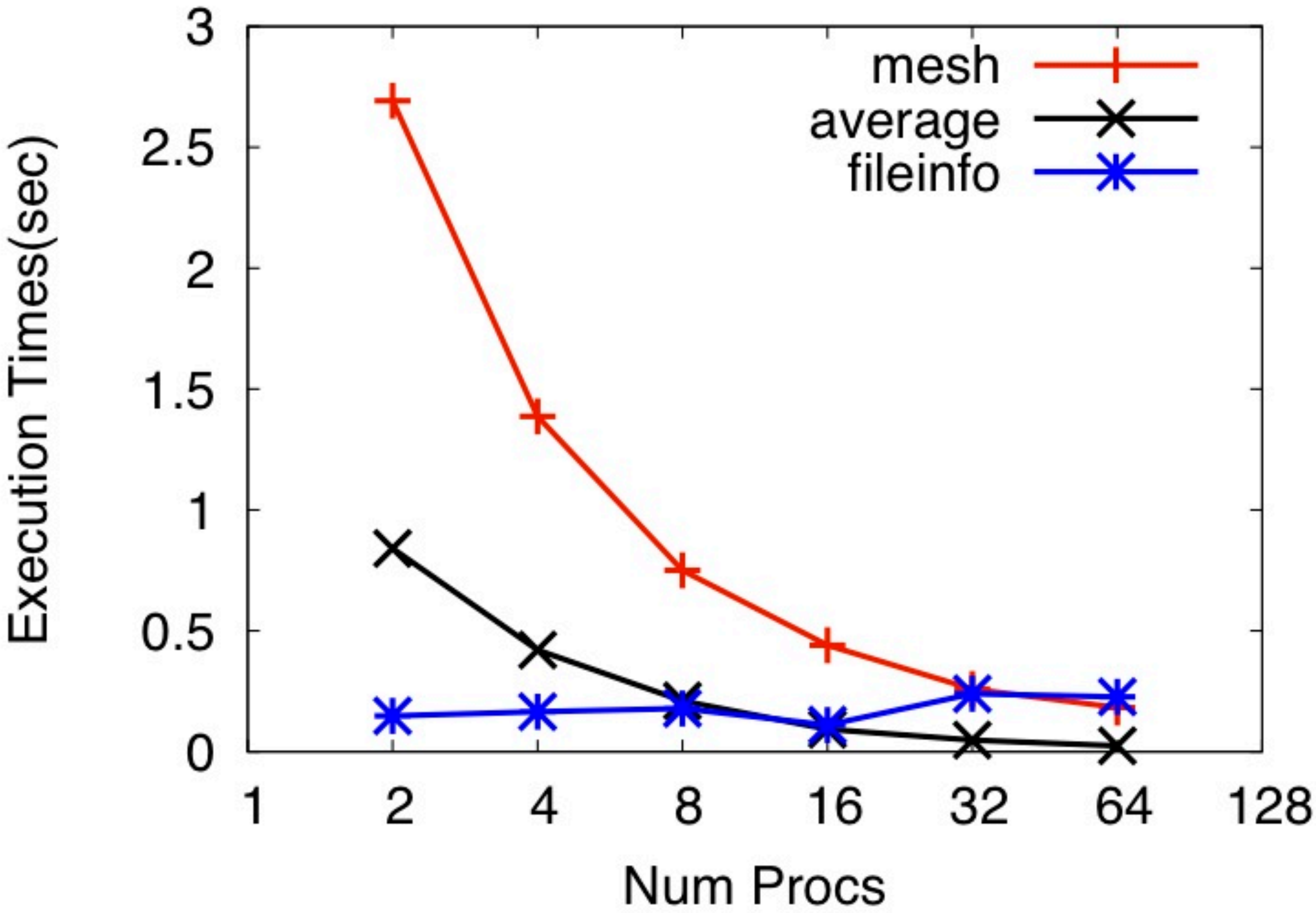
- Fileinfo
  - Abstraction of multiple files
- PcVAR
  - File Variables
  - User Variables
  - Read/write data through **MOAB**
- Analysis
  - Native: dim_avg_n, max, min (already implemented)
  - **Intrepid**
- MOAB
  - Parallel IO/Storage
- Misc utilities
  - MEM, ERR, LOG, PROF

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

ParCAL dim_avg_n Performance



ParCAL and ParNCL

- ParCAL is a library.

- ParNCL is an application written using that library. A parallel version of NCL (which also allows computations on unstructured grids).

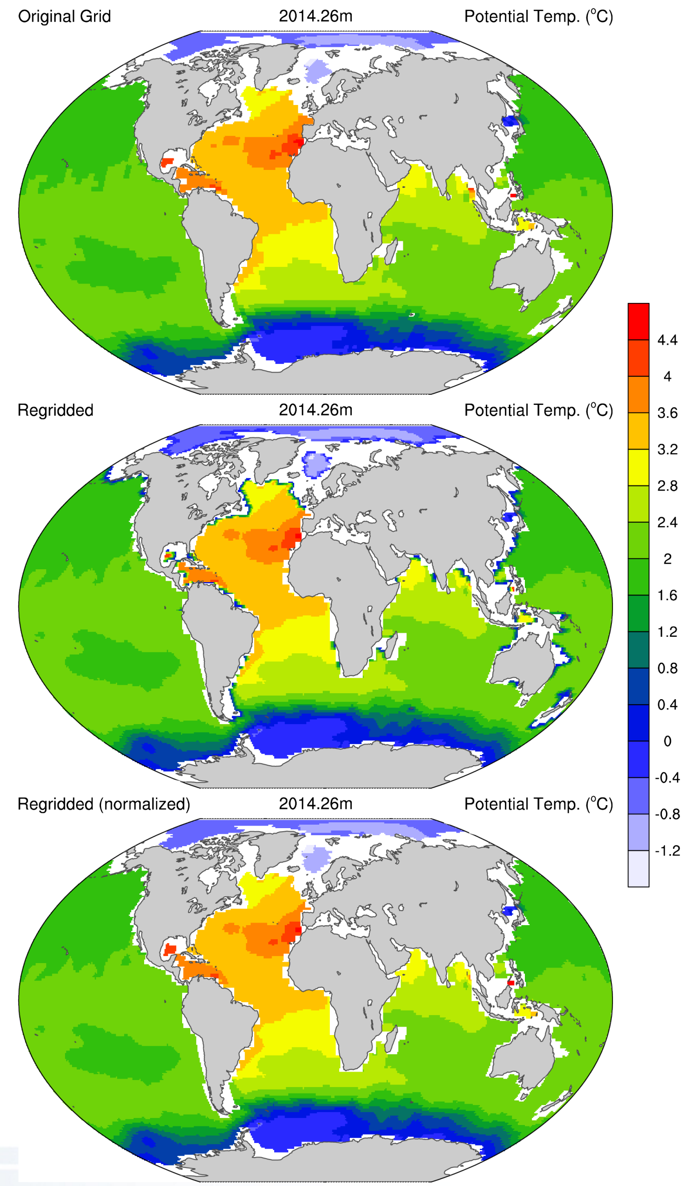
NCAR Command Language (NCL)

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

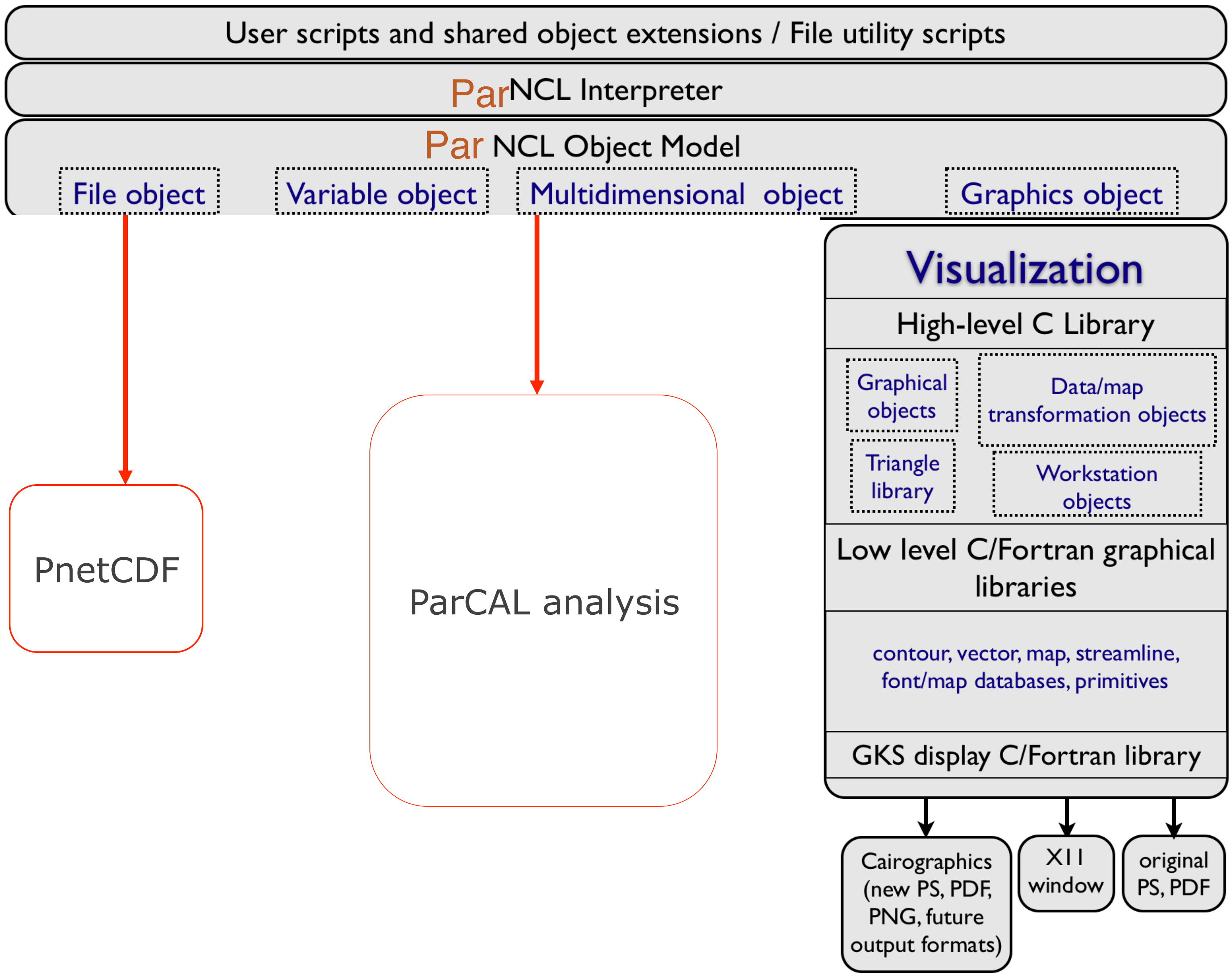
1. Simple, robust file input and output
2. Hundreds of analysis (computational) functions
3. Visualizations (2D) are publication quality and highly customizable

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

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



ParNCL architecture



Summary

- ParCAL and ParNCL beta versions exist. A few functions implemented.
  - Working on improving coverage, testing
  - Prioritizing NCL built in functions (300+) to implement

- Swift version of Atmospheric Model Working Group diagnostics released to user community.

http://trac.mcs.anl.gov/projects/parvis