Recent Updates to the CFD General Notation System (CGNS)

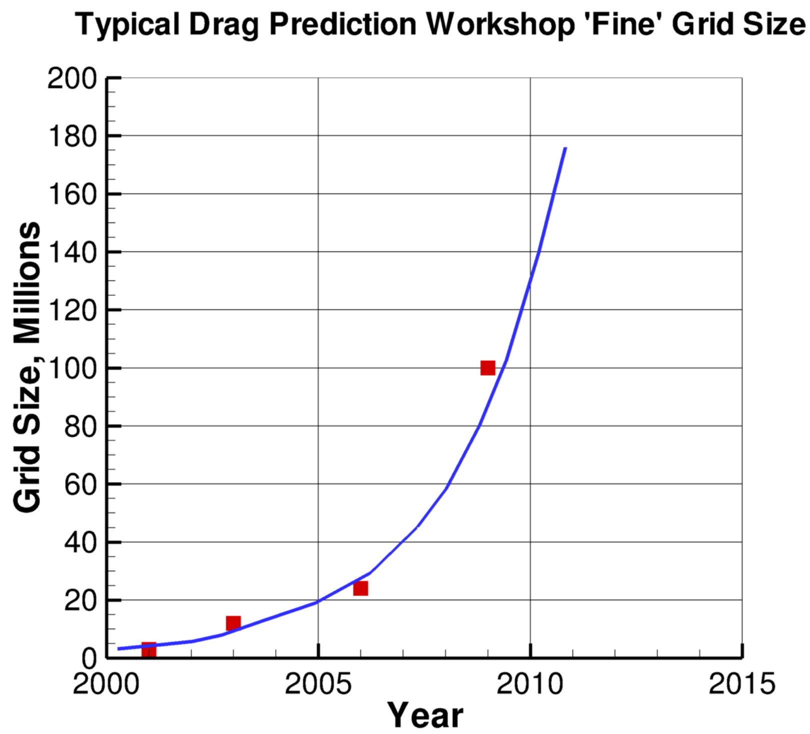
C. L. Rumsey
NASA Langley Research Center
B. Wedan
Computational Engineering Solutions
T. Hauser
University of Colorado
M. Poinot
ONERA

AIAA-2012-1264, 50th AIAA Aerospace Sciences Meeting, Nashville, TN, January 2012

Purpose of this talk

- Provide an overview of CGNS
- Provide a summary of recent developments & current status

Motivation



- Need to handle larger grid file sizes
  - 32-bit integer limit too low -> solution: 64-bit integers
  - I/O times increasing -> solution: parallel I/O
- CGNS can help… a lot of recent progress

What is CGNS?

- Standard for defining & storing CFD data
  - Self-descriptive
  - Machine-independent
  - Very general and extendable
  - Administered by international steering committee
- AIAA recommended practice (AIAA R-101A-2005)
- Free and open software
- Well-documented
- Discussion forum: cgnstalk@lists.nasa.gov
- Website: cgns.org

Introduction

- CGNS provides a general, portable, and extensible standard for the description, storage, and retrieval of CFD analysis data
- Principal target is data normally associated with computed solutions of the Navier-Stokes equations & its derivatives
- But applicable to computational field physics in general (with augmentation of data definitions and storage conventions)

History

- CGNS was started in the mid-1990s as a joint effort between NASA, Boeing, and McDonnell Douglas
  - Initially funded by NASA's Advanced Subsonic Technology (AST) program
  - Arose from need for common CFD data format for improved collaborative analyses between multiple organizations
- Version 1.0 of CGNS released in May 1998
- Current Version 3.1 released in April 2011

History, cont'd

- CGNS steering committee was formed in 1999
  - Voluntary public forum
  - International members from government, industry, academia
  - Became a sub-committee of AIAA Committee on Standards in 2000
- Value:
  - According to ohloh.net project cost calculator: in terms of value, CGNS represents a project with 94 person-years effort, $5.2M
  - CGNS committee continues to operate as a dedicated core of volunteers who contribute to CGNS as part of, or in addition to, their regular jobs
    - Many are enabled because their organizations place a high value on CGNS
    - NASA has also provided occasional contracts for software development

Steering committee

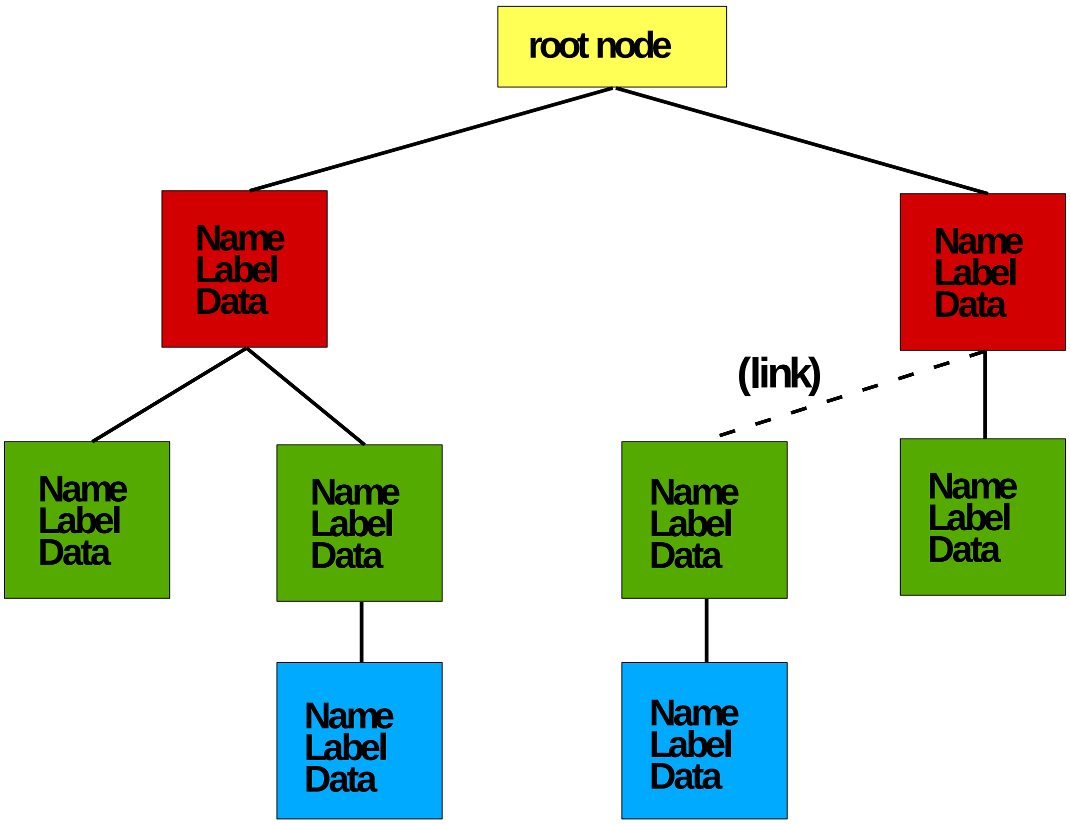
- CGNS Steering committee is a public forum
- Responsibilities include (1) maintaining software, documentation, and website, (2) ensuring free distribution, and (3) promoting acceptance
- Current steering committee make-up (20 members):

ADAPCO
ANSYS-CFX
ANSYS-ICEM CFD
Airbus
Boeing Commercial
Computational Eng. Solutions
Concepts NREC
Intelligent Light
NASA Glenn
NASA Langley
ONERA
Pointwise
Pratt & Whitney
Rolls-Royce Allison
Stanford University
Stony Brook University
Tecplot
TTC Technologies
University of Colorado
U.S. Air Force / AEDC

CGNS main features

- Hierarchical data structure : quickly traversed and sorted, no need to process irrelevant data
- Database is universal and self-describing
- Data may encompass multiple files through the use of symbolic links
- Portable ANSI C software, with complete Fortran, C, and Python interfaces
- Modular layered system, so that many of the data structures are optional

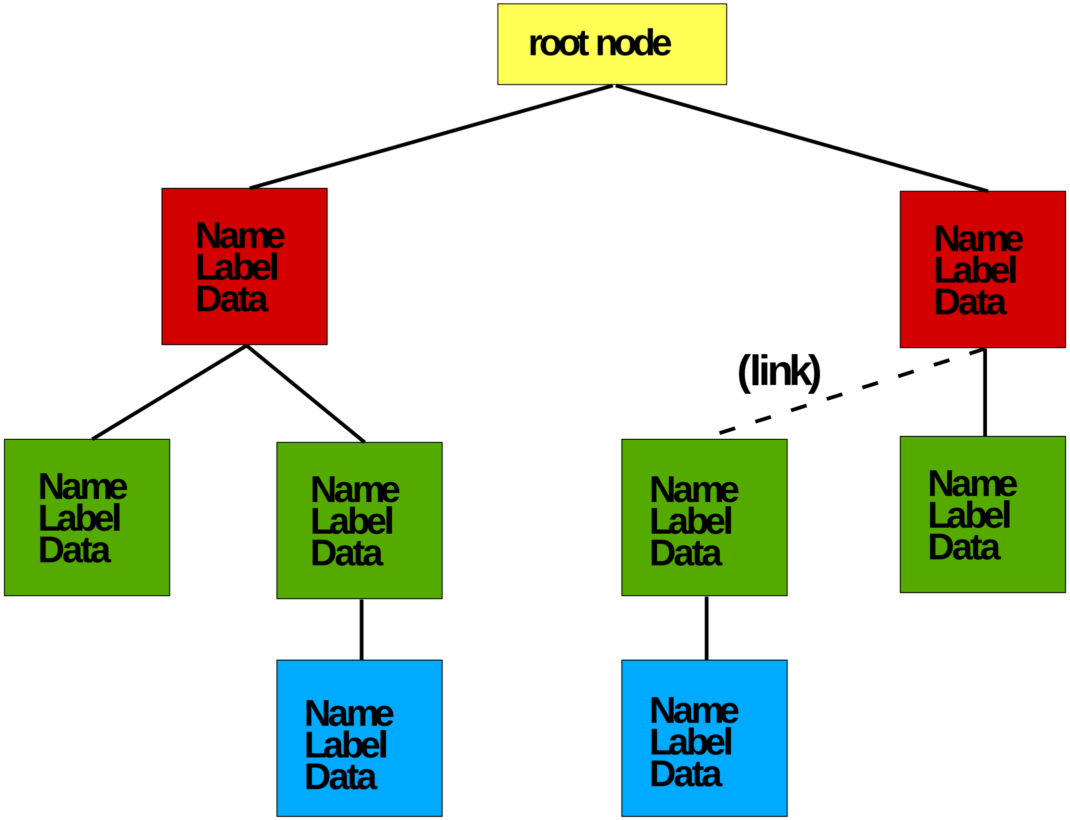
CGNS file layout



CGNS main features

- CGNS software is always backward compatible
- Supported by its users, via CGNSTalk e-mail exchange group
  - approx 350 registered users from 22 countries
- Standard Interface Data Structures (SIDS) is the core of CGNS – defines the intellectual content
- Architecture-independent application programming interface (API)
  - written as a mid-level library (MLL)

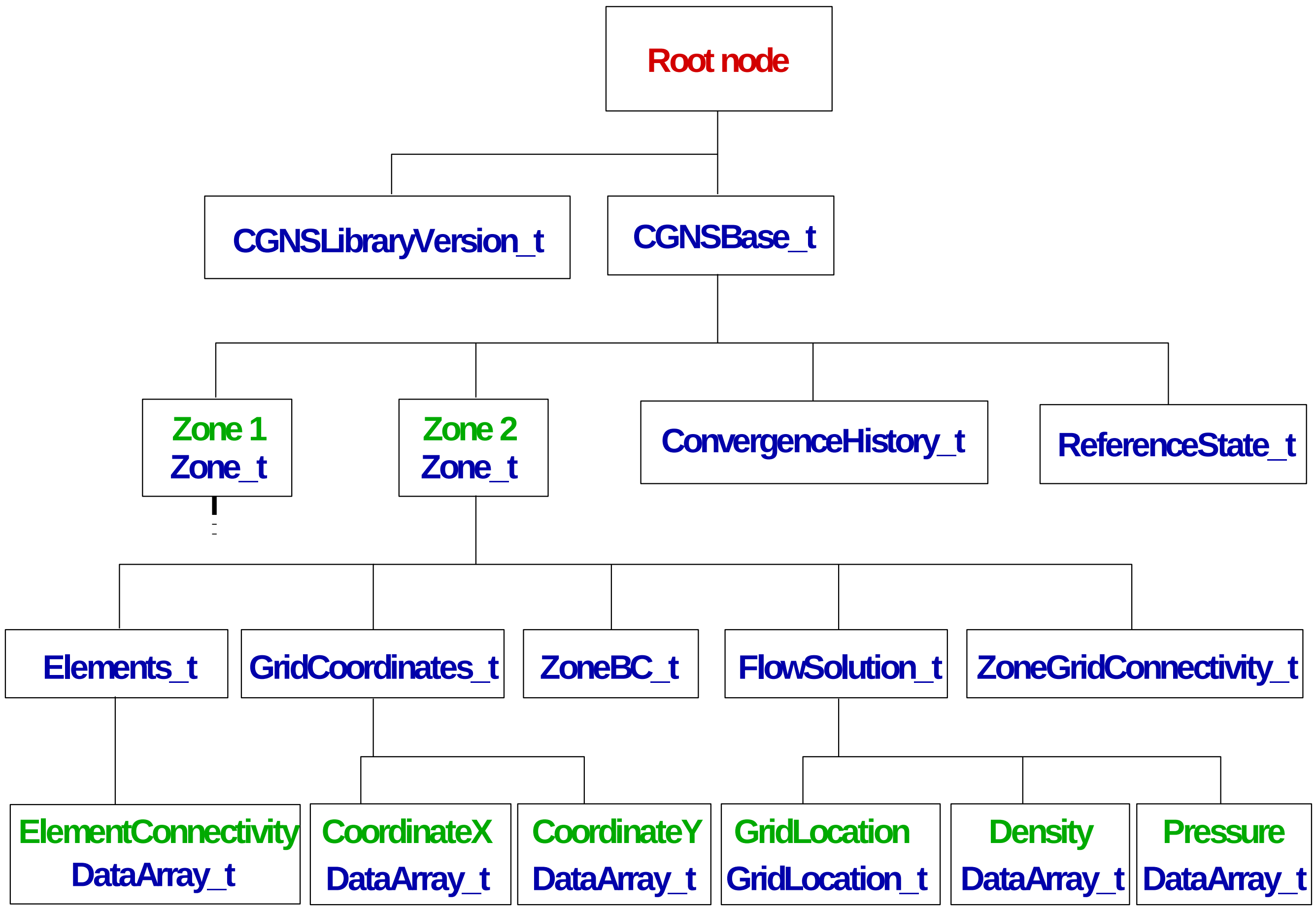
CGNS file layout



SIDS dictates the conventions of "what goes where," so that someone else reading the file knows where to find things.

The MLL implements these conventions.

Typical CGNS file



CGNS usefulness

- Improves longevity (archival quality) of data
  - Self-descriptive
  - Allows for unlimited user-defined comments & data
  - Machine-independent
- Easy to share data files between sites
  - Eliminates need to translate between different data formats
  - Rigorous standard means less ambiguity about what the data means
  - For example:
    - CGNS being used in part for AIAA Drag Prediction Workshop (DPW) and High Lift Prediction Workshop (HiLiftPW)
    - ONERA has adopted CGNS for its CFD-based workflow process
    - Pointwise has adopted CGNS as its default file type for grid generation
- Saves time and money
  - For example, easy to set-up CFD runs because files include grid coordinates, connectivity, and BC information
- Easily extendible to include additional types of data
  - Solver-specific or user-specific data can easily be written & read – file remains CGNS-compliant (others can still read it!)
  - Once defined & agreed upon, new data standards can be added

CGNS usefulness

- Improves longevity (archival quality) of data
  - Self-descriptive
  - Allows for unlimited user-defined comments & data
  - Machine-independent
- Easy to share data files between sites
  - Eliminates need to translate between different data formats
  - Rigorous standard means less ambiguity about what the data means
  - For example:
    - CGNS being used in part for AIAA Drag Prediction Workshop (DPW) and High Lift Prediction Workshop (HiLiftPW)
    - ONERA has adopted CGNS for its CFD-based workflow process
    - Pointwise has adopted CGNS as its default file type for grid generation
- Saves time and money
  - For example, easy to set-up CFD runs because files include grid coordinates, connectivity, and BC information
- Easily extendible to include additional types of data
  - Solver-specific or user-specific data can easily be written & read – file remains CGNS-compliant (others can still read it!)
  - Once defined & agreed upon, new data standards can be added

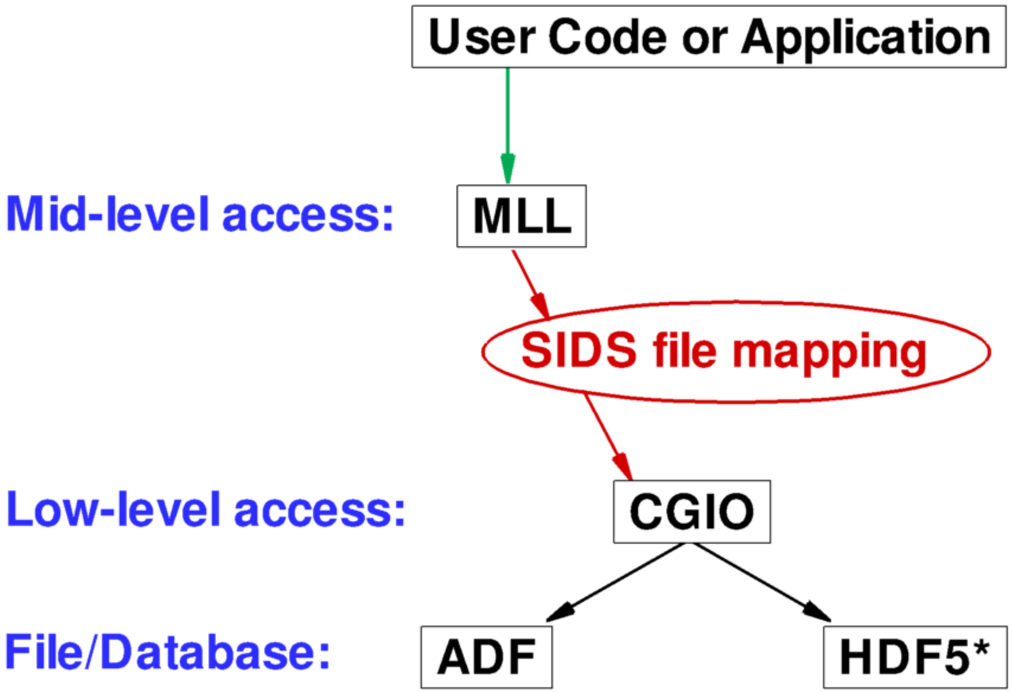
CGNS usefulness

- Improves longevity (archival quality) of data
  - Self-descriptive
  - Allows for unlimited user-defined comments & data
  - Machine-independent
- Easy to share data files between sites
  - Eliminates need to translate between different data formats
  - Rigorous standard means less ambiguity about what the data means
  - For example:
    - CGNS being used in part for AIAA Drag Prediction Workshop (DPW) and High Lift Prediction Workshop (HiLiftPW)
    - ONERA has adopted CGNS for its CFD-based workflow process
    - Pointwise has adopted CGNS as its default file type for grid generation
- Saves time and money
  - For example, easy to set-up CFD runs because files include grid coordinates, connectivity, and BC information
- Easily extendible to include additional types of data
  - Solver-specific or user-specific data can easily be written & read – file remains CGNS-compliant (others can still read it!)
  - Once defined & agreed upon, new data standards can be added

CGNS usefulness

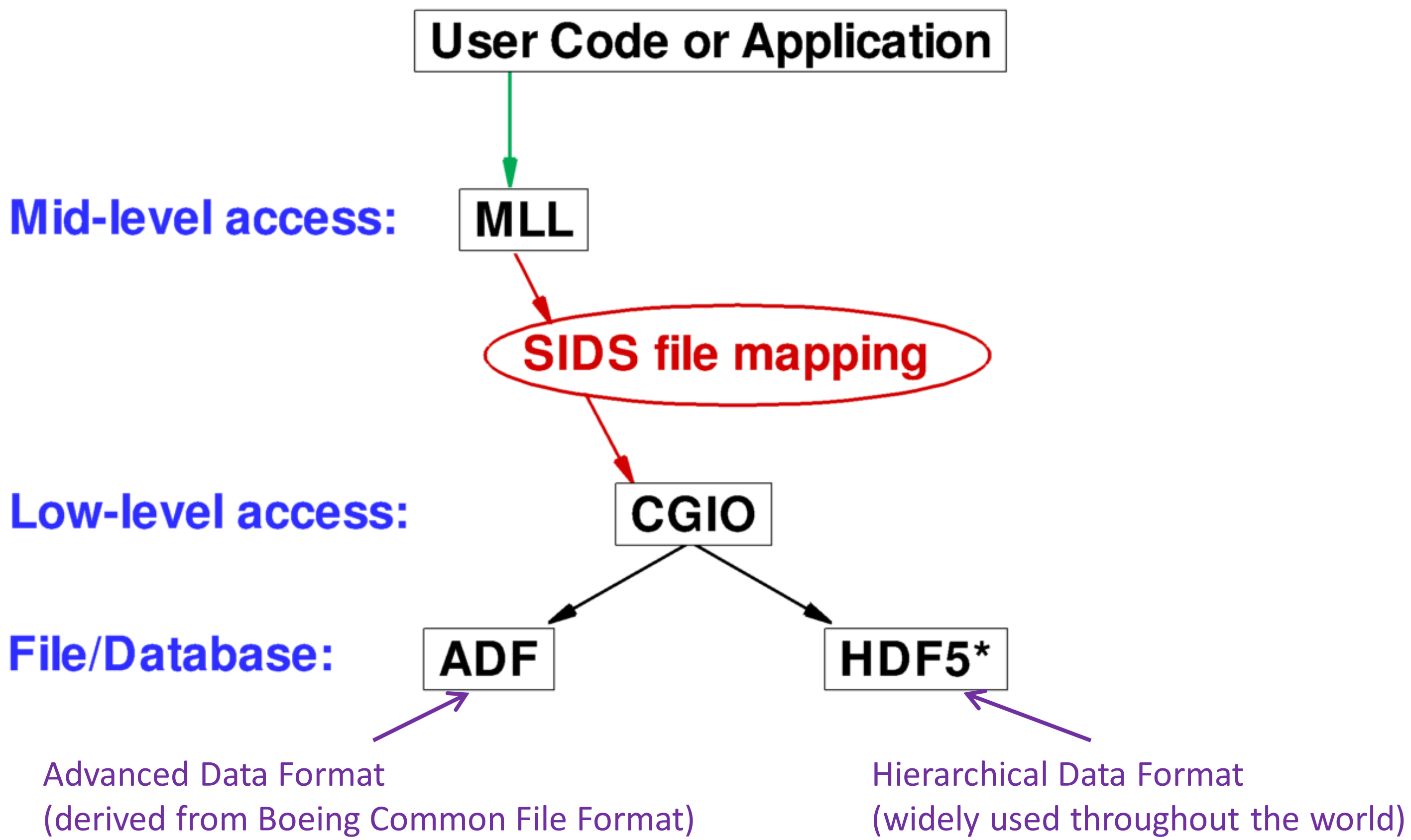
- Improves longevity (archival quality) of data
  - Self-descriptive
  - Allows for unlimited user-defined comments & data
  - Machine-independent
- Easy to share data files between sites
  - Eliminates need to translate between different data formats
  - Rigorous standard means less ambiguity about what the data means
  - For example:
    - CGNS being used in part for AIAA Drag Prediction Workshop (DPW) and High Lift Prediction Workshop (HiLiftPW)
    - ONERA has adopted CGNS for its CFD-based workflow process
    - Pointwise has adopted CGNS as its default file type for grid generation
- Saves time and money
  - For example, easy to set-up CFD runs because files include grid coordinates, connectivity, and BC information
- Easily extendible to include additional types of data
  - Solver-specific or user-specific data can easily be written & read – file remains CGNS-compliant (others can still read it!)
  - Once defined & agreed upon, new data standards can be added

This is the most commonly-used access method



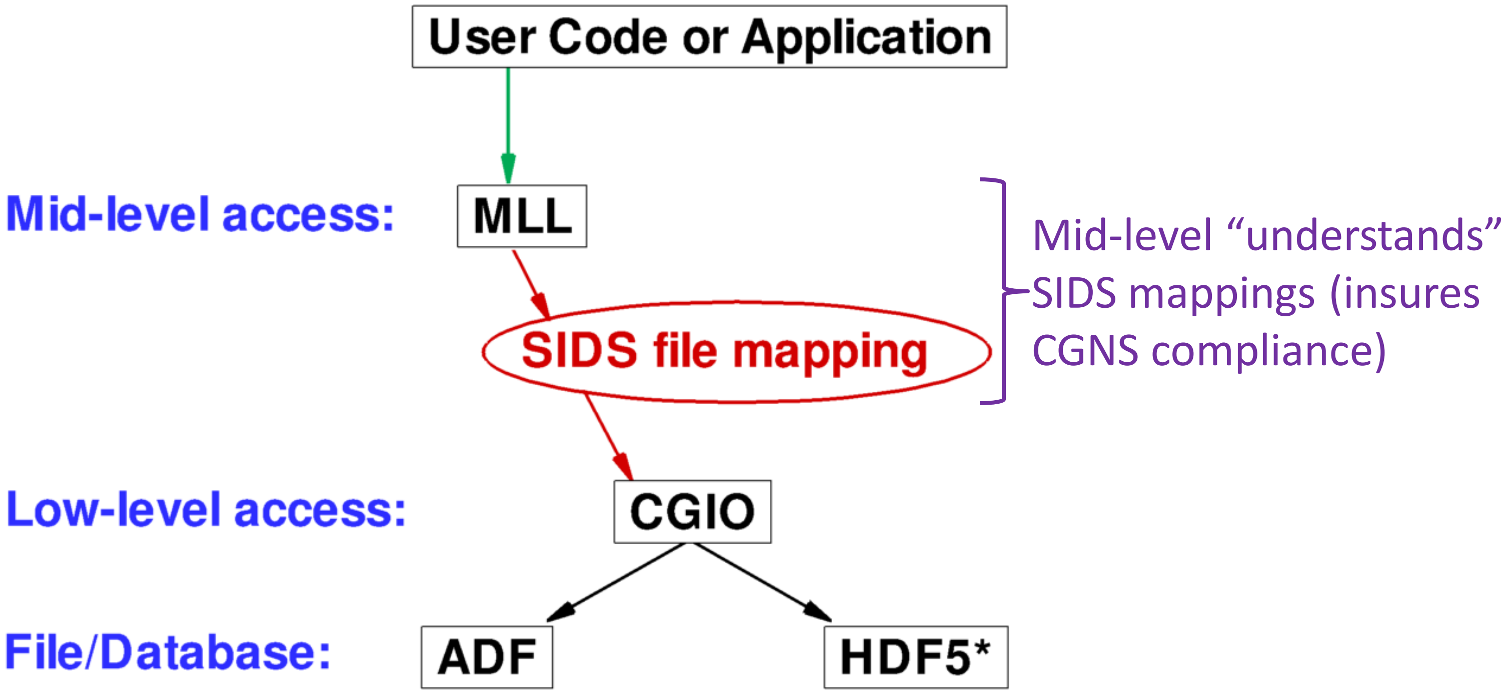
* available separately from CGNS distribution

This is the most commonly-used access method



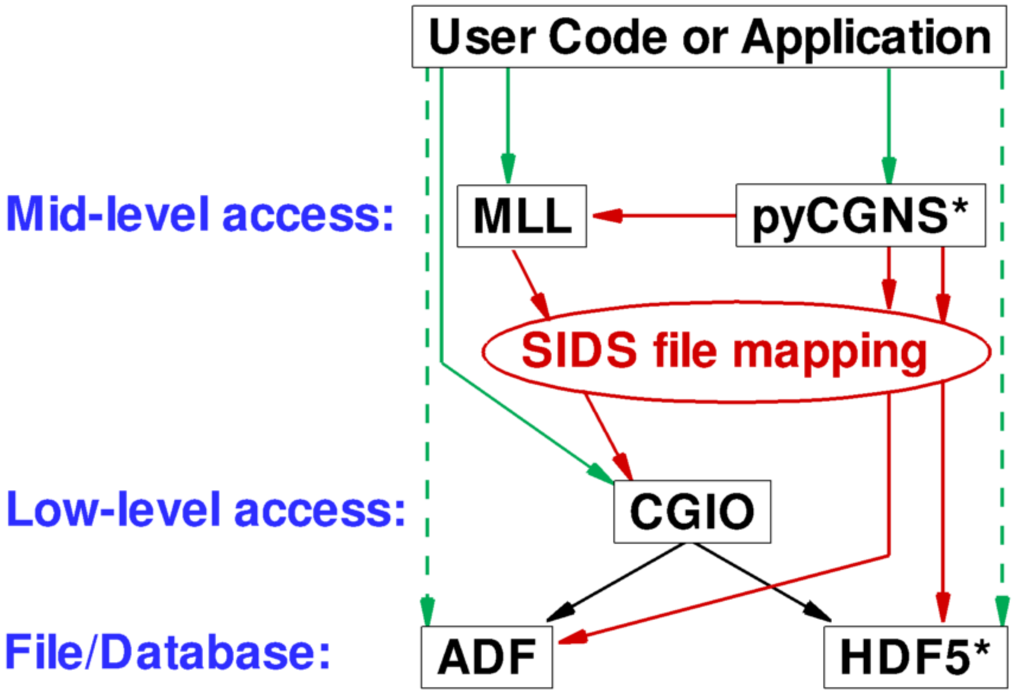
* available separately from CGNS distribution

This is the most commonly-used access method



* available separately from CGNS distribution

Other access options



* available separately from CGNS distribution

How CGNS works

- Users must download the free CGNS software
  - This includes low-level software (basic I/O operations)
  - Also includes MLL software (for ease of SIDS-compliant implementation)
  - Software will work with either HDF5 or ADF
- Users are encouraged to use the MLL to read and write their data (helps ensure CGNS-compatibility)
- Files are portable across computer platforms
- Tools (such as CGNSview) allow user to “see” what is in the CGNS file
- Most commercial pre- and post-processing software support CGNS format

What's new in CGNS?

(recent enhancements)

- File-level changes
  - Official adoption of HDF5
  - 64-bit integer capability
- Low-level access changes
  - CGIO interface
- SIDS changes
  - Unstructured polyhedral element capability
  - Time-dependent connectivities
  - General SIDS improvements
  - Regions
- Mid-level access changes
  - New MLL functions
  - Parallel CGNS MLL
  - Python mapping
- New applications
  - CGNSview

What's new in CGNS?

(recent enhancements)

- File-level changes
  - Official adoption of HDF5
  - 64-bit integer capability
- Low-level access changes
  - CGIO interface
- SIDS changes
  - Unstructured polyhedral element capability
  - Time-dependent connectivities
  - General SIDS improvements
  - Regions
- Mid-level access changes
  - New MLL functions
  - Parallel CGNS MLL
  - Python mapping
- New applications
  - CGNSview

Only some of these will be discussed here…
see paper for the others

Use of HDF5

- Original ADF file format works well, but does not have parallel I/O or data compression like HDF5
- HDF5 is now a world-wide standard for storing scientific data
- HDF5 is now considered the CGNS “standard”
  - But for purpose of backward compatibility, both ADF and HDF5 will continue to be supported indefinitely
  - Both are handled transparently through new low-level CGIO interface
  - User can easily choose and switch between them

64-bit integer capability

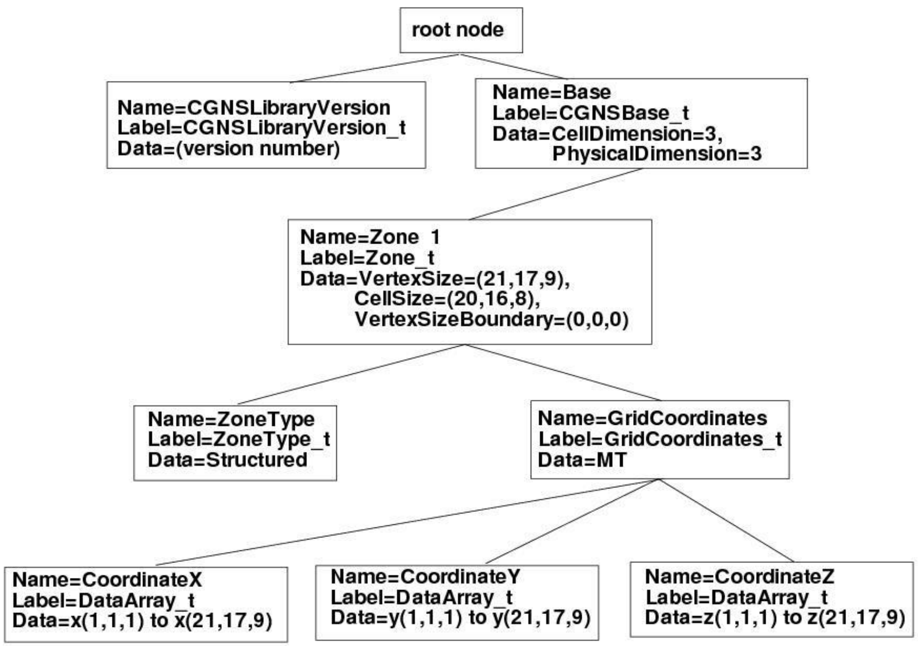
- Indexing arrays for very large CFD grids can exceed the 32-bit integer limit
  - 32-bits handles integer addresses up to $\pm 2{,}147{,}483{,}648$
  - E.g., tetrahedral grid with 90M nodes and approx 540M cells; its cell-to-node pointer array exceeds the limit (2.16B integers)
- 64-bits handles addresses up to $\pm 9.2 \times 10^{18}$
- 64-bit integers handled in CGNS software through new data type cgsize_t
- CGNS V3.1 can compile in either 32-bit or 64-bit mode
  - Backward compatible
  - “Legacy mode” creates files readable by earlier versions of CGNS

SIDS: unstructured polyhedral element capability

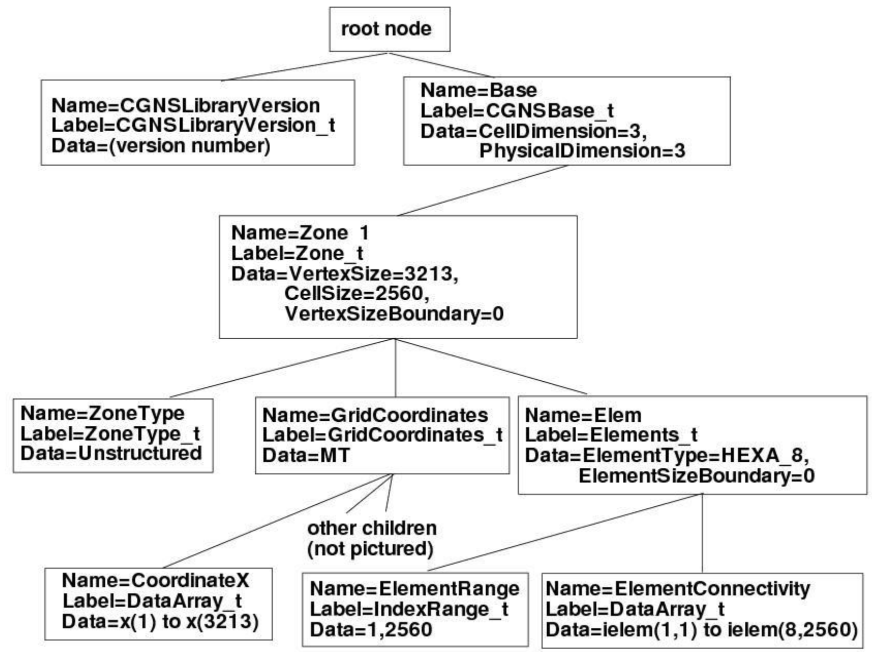
- CGNS can write both structured and unstructured grids
  - Unstructured grids typically made up of "standard" element types: hexahedra, tetrahedra, pyramids, pentahedra, etc.

Typical CGNS file

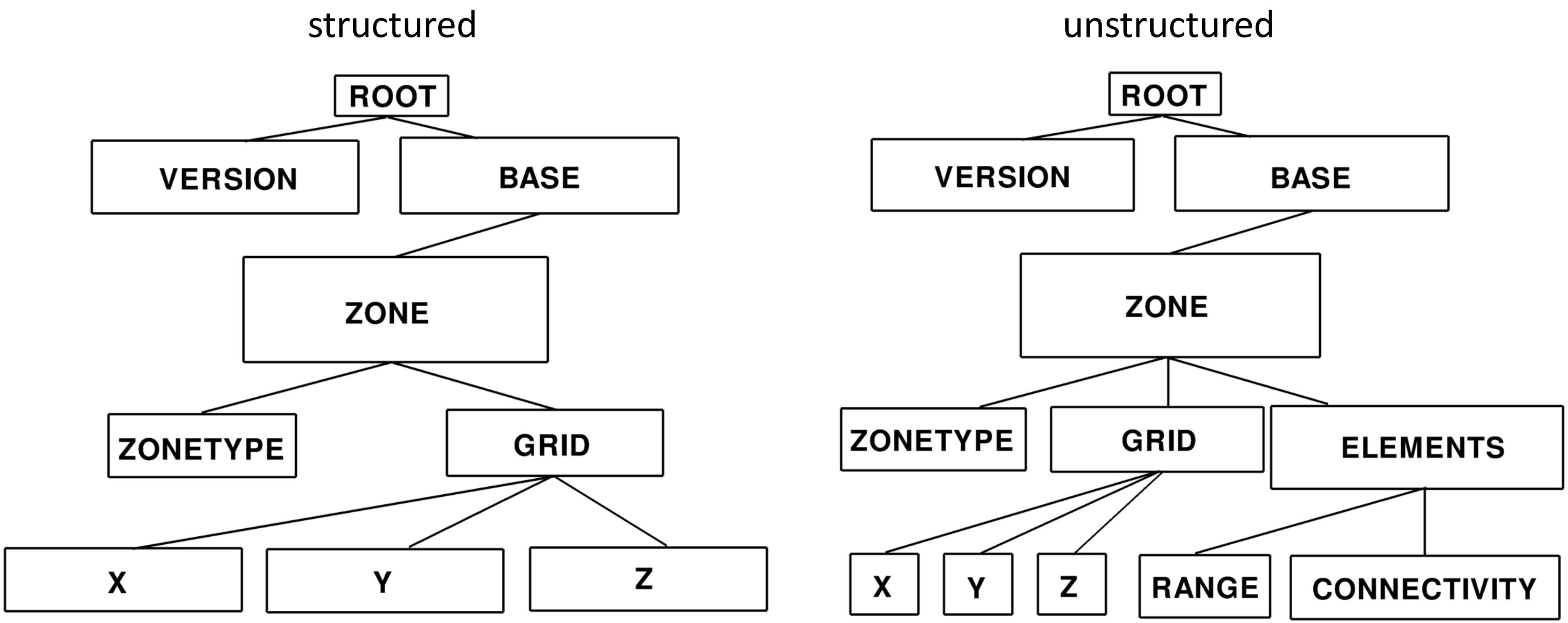
structured



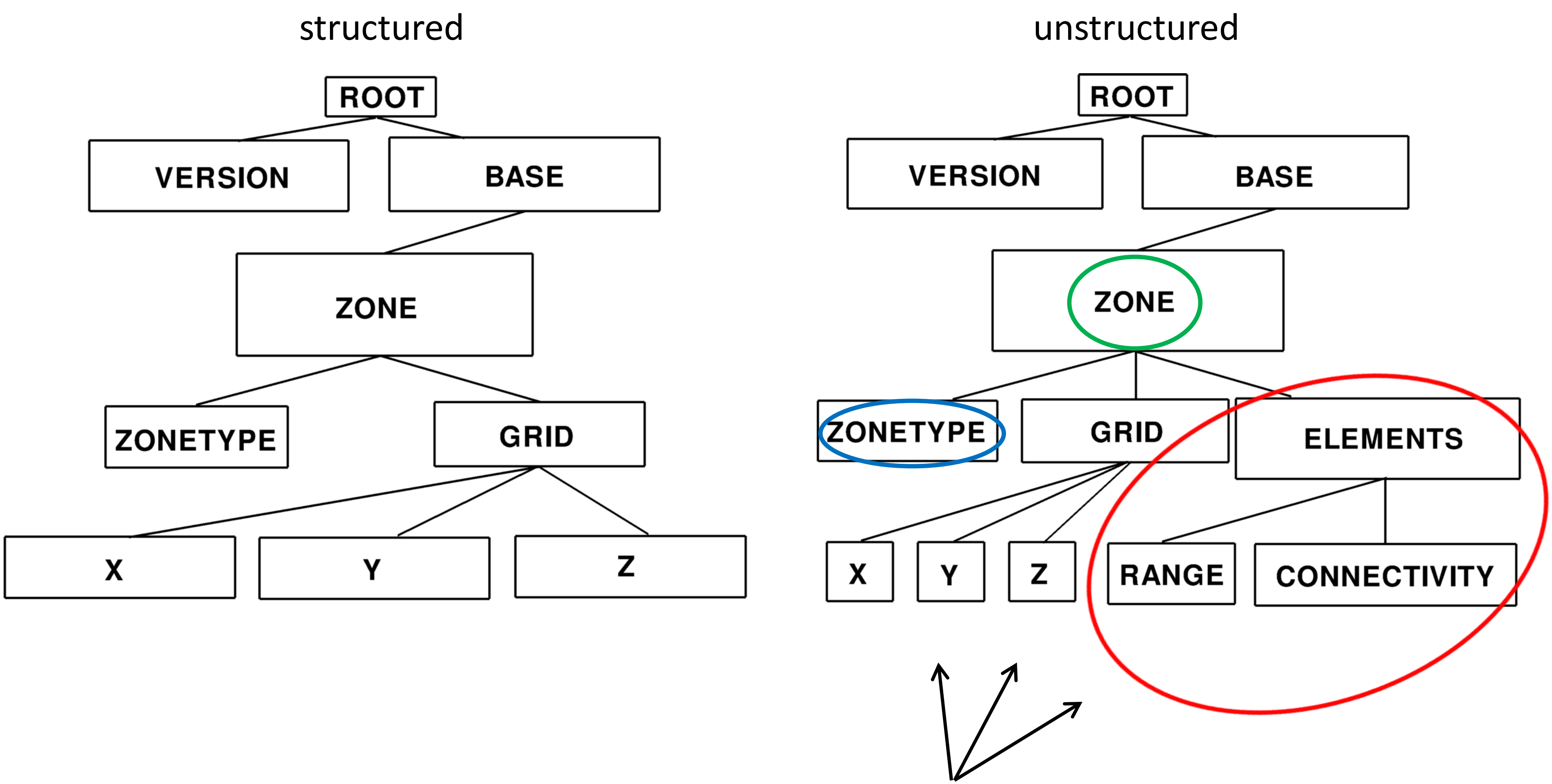
unstructured



Typical CGNS file (simplified)



Typical CGNS file (simplified)



(what's different between structured and unstructured)

SIDS: unstructured polyhedral element capability

- New arbitrary polyhedral elements are described by 2 element types:
  - Faces: each face is listed in terms of its component nodes

    and

  - Elements: each element is listed in terms of its component faces (the sign indicates the direction of the face normal relative to the element)

Simple example

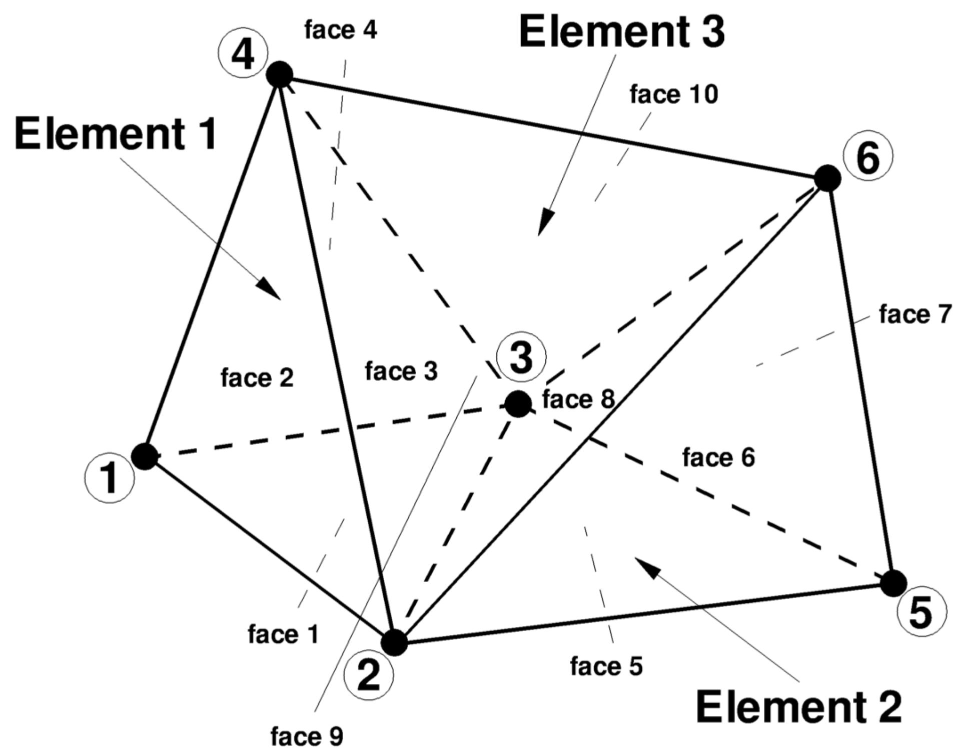
(can be any shape – in this case all faces are triangles and all elements are tetrahedra)

There are 10 face elements, with node numbers:

- Face 1: 1,3,2
- Face 2: 1,2,4
- Face 3: 2,3,4
- Face 4: 3,1,4
- Face 5: 2,3,5
- Face 6: 2,5,6
- Face 7: 5,3,6
- Face 8: 3,2,6
- Face 9: 2,6,4
- Face 10: 6,3,4

and 3 volume elements, with face numbers:

- Volume 1: 1,2,3,4
- Volume 2: 5,6,7,8
- Volume 3: -8,9,10,-3

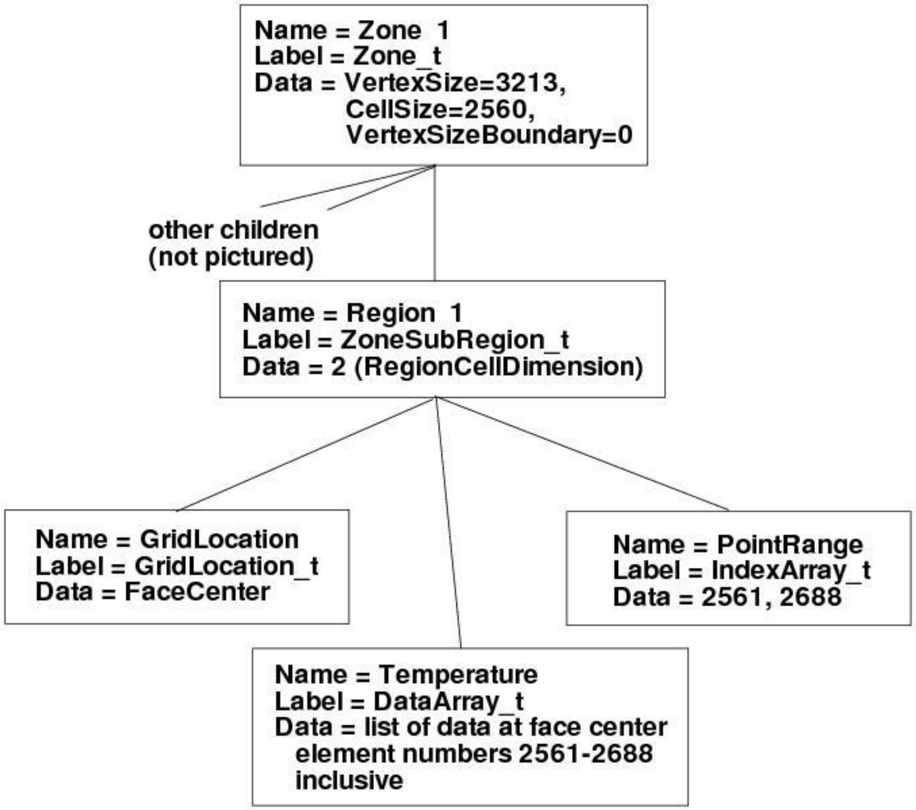


SIDS: Regions

- Previously, CGNS only allowed flowfield or other information associated with the grid to be given over the *entire* grid
- New ZoneSubRegion_t node allows user to store subsets over boundaries, portions of boundaries, or portions of the volume

Example

storing sub-region of temperatures at specific boundary elements



Parallel CGNS MLL

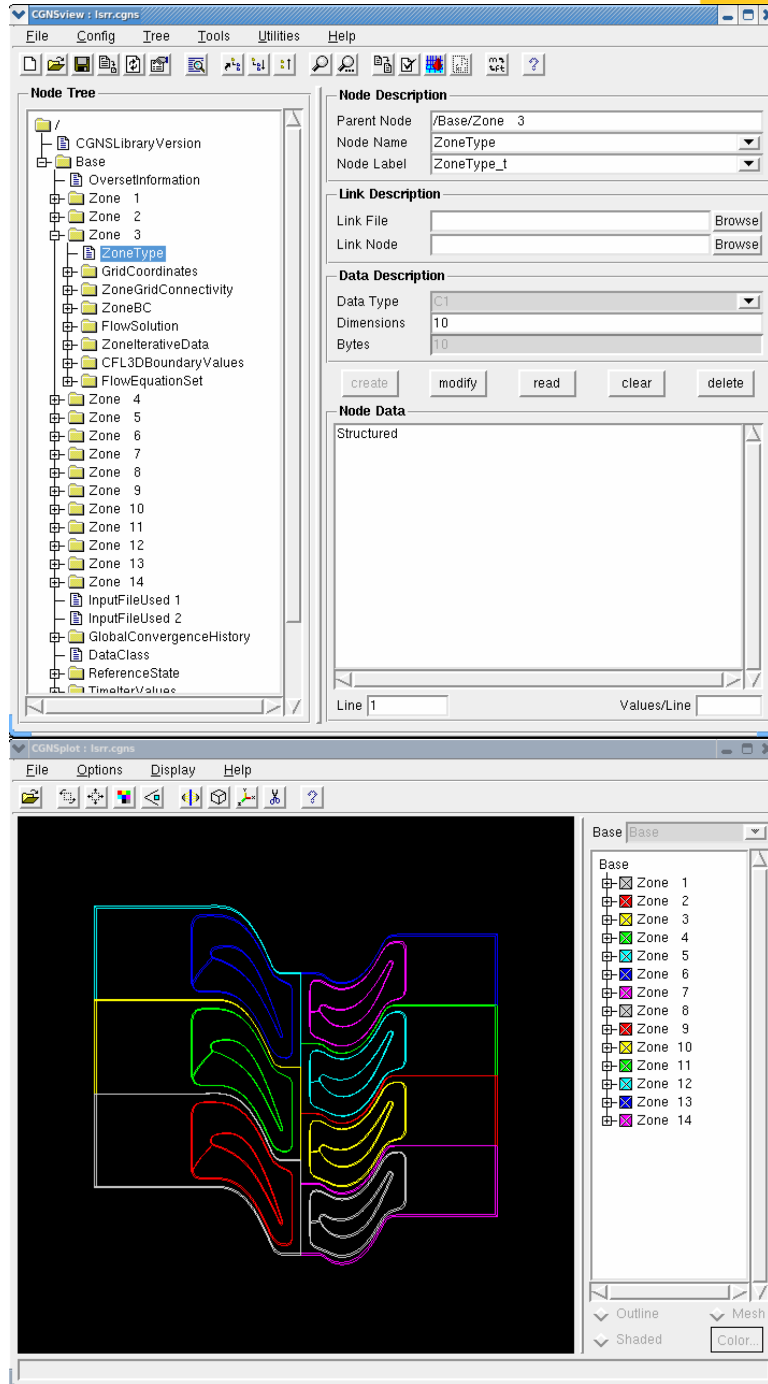
- Tied into HDF5 parallel capability directly
  - Takes advantage of the collective I/O support in recent version of HDF5 (using MPI-IO)
- Parallel MLL writes grids, solution, and other data arrays in parallel
  - Meant to be a supplement to the non-parallel MLL
- Parallel CGNS currently available as “alpha” release (not fully integrated or supported)
- Plan to fully integrate during early 2012

Python mapping

- Python facilitates the use of CGNS in multi-physics simulations
  - Used extensively at ONERA to create inter-operability between different simulation codes
  - A complete CGNS tree can be defined as a Python “list of lists” using raw Python types and NumPy arrays
  - Not well-suited for time-consuming computations; but very useful for setting up complex workflow processes

CGNSview

- Viewer/editor based on Tck/Tk
- Uses CGIO interface, so can read both ADF and HDF5 files
- Includes:
  - Plotting capability
  - CGNS file validator
  - Import, export, data conversion, subset extraction, and interpolation utilities



Concluding remarks

- Some recent enhancements to the CGNS standard (and its surrounding ecosystem) have been described
- CGNS has proved to be long-lasting and stable, yet readily-extensible to handle new types of data
- Its recent upgrade to 64-bit integer capability (*while still remaining backward compatible*) expands its range of usefulness as the grid sizes used in the CFD community continue to grow
- The Steering committee is open to new members who are interested in actively contributing to the future of CGNS

End