CPEX #42: Storing the Bounding Box of a grid

Proposal author : Mickael Philit (mickey.phy@gmail.com)

Sponsor : Marc Poinot (marc.poinot@safrangroup.com)

1. Introduction

CGNS currently do not store explicitly the bounding box of a grid defined in a GridCoordinates_t node. This information can be computed from coordinates arrays but it would be convenient to store and retrieve it quickly as algorithms could then harness this data (for load on demand, search process or). For instance the following paper shows usage of bounding box in an HPC context :
http://www.cie.bgu.tum.de/publications/masterthesis/2015_Ertl_Master.pdf

As a matter of fact, major existing cfd file formats already provide this metadata :

- Xdmf is using Information node to store bounds : http://www.xdmf.org/index.php/XDMF_Model_and_Format#Information
- Ensight gold format supports model extents : http://www3.ensight.com/EnSight92_Docs/UserManual.pdf section "EnSight Gold Geometry File Format"
- VTK XML file format uses xml attribute to keep Extent of the grid
- …

The proposal is thus to add the bounding box as an optional data of GridCoordinates_t node.

2. Proposed extension, CGNS/SIDS

The simple modification consists in attaching the Bounding Box as an optional data of GridCoordinates.

```
GridCoordinates_t< int IndexDimension, int VertexSize[IndexDimension], int PhysicalDimension > :=

{
   DataType[PhysicalDimension, 2] BoundingBox ;                    (o)

   List( Descriptor_t Descriptor1 ... DescriptorN ) ;              (o)

   Rind_t<IndexDimension> Rind ;                                   (o/d)

   List( DataArray_t<DataType, IndexDimension, DataSize[]>
        DataArray1 ... DataArrayN ) ;                              (o)

   DataClass_t DataClass ;                                         (o)

   DimensionalUnits_t DimensionalUnits ;                           (o)

   List( UserDefinedData_t UserDefinedData1 ... UserDefinedDataN ) ; (o)
} ;
```

The Bounding Box array will store minimum and maximum of coordinates values sorted by an alphabetical order for cartesian and cylindrical coordinates while Spherical and Auxiliary coordinates retain a specific order to be coherent with the existing SIDS notation :

- Bounding Box for 3D cartesian coordinates :
  [[Min (CoordinateX), Max(CoordinateX)],
  [Min (CoordinateY), Max(CoordinateY)],
  [Min (CoordinateZ), Max(CoordinateZ)]]

- Bounding Box for 3D cylindrical coordinates :
  [[Min (CoordinateR), Max(CoordinateR)],
  [Min (CoordinateTheta), Max(CoordinateTheta)],
  [Min (CoordinateX), Max(CoordinateX)]]
  or
  [[Min (CoordinateR), Max(CoordinateR)],
  [Min (CoordinateTheta), Max(CoordinateTheta)],
  [Min (CoordinateY), Max(CoordinateY)]]
  or
  [[Min (CoordinateR), Max(CoordinateR)],
  [Min (CoordinateTheta), Max(CoordinateTheta)],
  [Min (CoordinateZ), Max(CoordinateZ)]]

- Bounding Box for 3D spherical coordinates :
  [[Min (CoordinateR), Max(CoordinateR)],
  [Min (CoordinateTheta), Max(CoordinateTheta)],
  [Min (CoordinatePhi), Max(CoordinatePhi)]]

- Bounding Box for 3D Auxiliary coordinates :
  [[Min (CoordinateXi), Max(CoordinateXi)],
  [Min (CoordinateEta), Max(CoordinateEta)],
  [Min (CoordinateZeta), Max(CoordinateZeta)]]

Thus, all coordinate systems are handled in a deterministic way. For 2D coordinates the order is kept the same as for 3D.

1. Modification of CGNS/Filemap

The proposal consists in storing the Bounding Box information directly in GridCoordinates_t Data (that was not used before). Thus GridCoordinates_t mapping will be the following :

Node Attributes

Name: GridCoordinates or user defined
Label: GridCoordinates_t
DataType: MT or R4 or R8
Dimension: 2
Dimension Values: PhysicalDimension, 2
Data : BoundingBox
Children: See GridCoordinates_t figure
Cardinality: 0,N
Parameters: IndexDimension, VertexSize , PhysicalDimension
Functions: DataSize

3. Modification of CGNS/MLL

A new function should be introduced for reading bounding box while keeping compatibility with existing code:

ier = cg_grid_boundingbox_read(**int fn**, **int B**, **int Z**, **int G**, ***DataType_t DataType, void *Data***);

Input/***Output***

fn	CGNS file index number.
B	Base index number, where 1 ≤ B ≤ nbases.
Z	Zone index number, where 1 ≤ Z ≤ nzones.
G	Grid index number, where 1 ≤ G ≤ ngrids.
DataType	Type of data in the bounding box. The admissible types are RealSingle, RealDouble.
Data	The bounding box data array.
ier	Error status.

In case the bounding box information is not present, Data will remain untouched and a warning will be emitted.

A similar function is specified for writing bounding box to an existing Grid :

Ier = cg_grid_boundingbox_write(**int fn**, **int B**, **int Z**, **int G**, **DataType_t DataType, void* Data**) ;

4. Conflict and compatibility concern

Modifying GridCoordinates_t in CGNS/SIDS does not seem to create conflict with existing CGNS convention. Indeed, an existing CGNS file without Bounding box information in GridCoordinates nodes would be read and emits a warning only when trying to access the bounding box information and this situation can be handle properly. In case of CGNS file with Bounding Box information, existing codes will just ignore the data.

5. Observations

To determine the system of coordinates being used for the GridCoordinates_t node and thus the bounding box, reading the coordinate array names is needed. Adding a ReferenceFrame notion would be a great feature to remove this pattern.