Unified Data Model: A Library for Parallel I/O and Data Management

William W. Dai, Robbie Aulwes, HPC-4

The Unified Data Model (UDM) is an original library for parallel input/output (I/O) and data management that we developed under the Advanced Simulation and Computing (ASC) program for code projects. The goal of the library is to provide sustainable, interoperable, efficient, scalable, and convenient tools for parallel I/O and data management for high-level data structures in applications, and to provide tools for the connection between applications, i.e., files generated from one application to be used in another application as inputs.

Functionality for Array and Structured Mesh

The UDM library provides I/O tools for single and multidimensional arrays used in numerical simulations. It supports arrays with any number of dimensions. The minimum requirement to write an array in parallel is to set the size in each dimension on its own computer processor, and the starting address of data in memory. As a result, each processor contributes a part of a multidimensional array. The library also supports ghost cells, which are typically involved in simulations in parallel computer environments.

The capability for structured meshes in the library is very similar to the one for multidimensional arrays, but association between a structured mesh and a set of variables will be automatically built through the library. Variables may also be defined on different structured meshes, and they may be defined on nodes, faces, and mesh elements.

Functionality for Unstructured Meshes and Variables

One of the important and powerful functions in the UDM library is the management of unstructured meshes and variables. The library supports a broad range of unstructured meshes, which include meshes with fixed shapes, arbitrary polygons, and arbitrary polyhedrons. A mesh element may be a zone, or face, or edge; i.e., zone-mesh, face-mesh, and edge-mesh. An edge-mesh may be 1-D, 2-D, or 3-D; and a face-mesh may be 2-D or 3-D. Mesh elements of a zone-mesh may be made directly from nodes, the elements may be made from edges, and the elements may also be made from faces, and the faces are made from either edges or nodes. The UDM library also supports ghost mesh elements, boundary faces, boundary edges, boundary nodes, slip faces, slip edges, slip nodes, and other options. The variables the library supports include node-, or edge-, or face-, or zone-variables, and the variables may be scalars, vectors, and tensors.

Although the UDM library covers a broad range of unstructured meshes, a user only has to set up one mesh definition and all other mesh definitions will be hidden from the user. For example, for an unstructured zone-mesh made from nodes, only a list of nodes for each element is needed, if the elements are of a fixed shape, such as prisms. If mesh elements are arbitrary polyhedrons made from nodes, two arrays are needed; one for the numbers of nodes for each element, and the other for the list of nodes for each element. Like the

capability for structured meshes, the association between a mesh and a set of variables is automatically built into the library.

Functionality for Querying
A file written through the UDM library is self-descriptive. All the information in the file may be queried by calls to the library. For example, for a given file, users may find the number of arrays, meshes, and variables, the description of each array, mesh, and variable, and any association between meshes and variables. Through querying function calls in the library, meshes and variables may be viewed through graphics tools.

After a data object, such as array, or mesh, or variable, is written into a file, users may read any part of the data object, for example, as global IDs, or processor rank, or space domain. Figure 1 illustrates the capability for reading three parts of an unstructured mesh with 1.6 billion elements. The left image is a part read through a processor rank, the middle one is a part identified through global IDs, and the right image is read based on a space domain.

I/O Performance
The UDM library is built on the top of message-passing inferface in/out (MPI-IO). The performance of the library is approximately 95% that of the best MPI-IO performance. To get that performance, we first make sure that the data on each computer processor are contiguously written onto a disk file, and therefore there is no movement of data during writing. Second, the library collects all metadata, i.e., the description of data, and writes the collection only when a file is closed. Third, when reading a file, the library reads all the metadata together and reads it only once.

For more information contact William W. Dai at dai@lanl.gov.

Funding Acknowledgements
This research was supported by the NNSA tri-Lab Advanced Simulation and Computing Program.

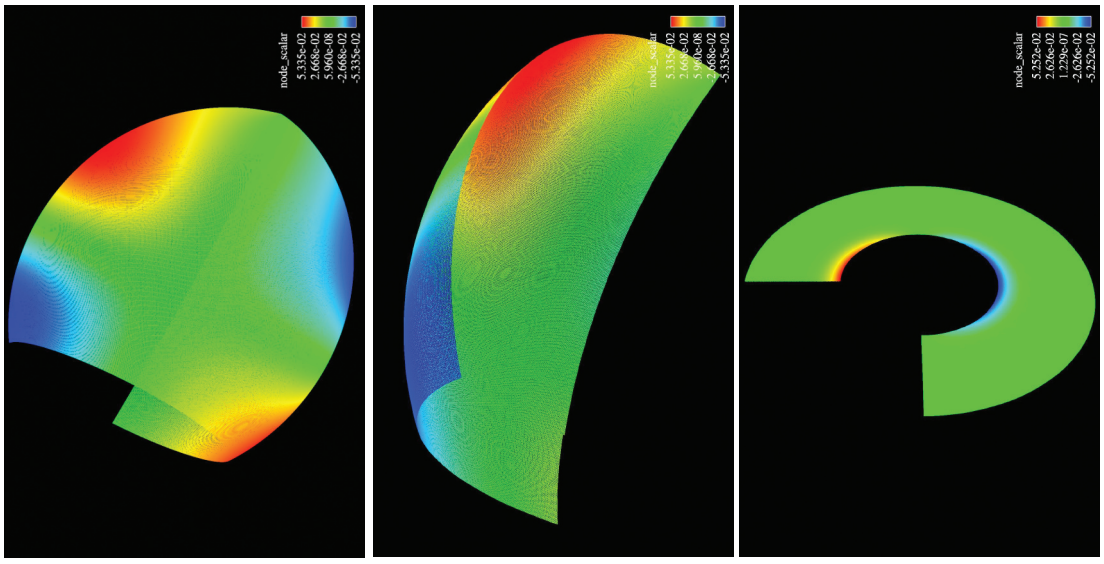

Fig. 1.
The three parts of an unstructured mesh and the variables defined on the three parts. The original full mesh contains 1.6 billion elements. Each part of the mesh is read from the original mesh through three different capabilities. The left image is read through a computer processor rank, the middle one is obtained through a set of global IDs, and the right image is obtained from a user-defined space domain.