Using XDMF to read HDF5

From VisItusers.org

HDF5 files can be used to store a wide variety of array data and many complex structures can be decomposed into array data and stored to HDF5 files. When an application needs to read that data at the more complex level of objects instead of data arrays, the conventions needed to reassemble the array data into an object must be known. Often, these conventions can be very complex. The XDMF (http://www.xdmf.org) file format provides support for creation of an XML schema file that can be used to tell an application how to read and reassemble HDF5 array data into higher level constructs.

The basic idea behind using the XDMF reader for VisIt is that you have existing HDF5 files and you create an XML schema *(.xmf file)* that tells VisIt how to read your data files. From there, you open the *.xmf* file in VisIt and VisIt reads your HDF5 data through the XDMF database reader plugin.

Contents

Examples

This section contains examples on how to use XDMF to read HDF5 data into VisIt.

2D curvilinear mesh

This example provides a simple C++ program to create an HDF5 data file and the XML schema *(.xmf)* file needed to read the data into VisIt as a 2D curvilinear mesh. The X,Y coordinate arrays are stored separately and there are fields defined on the mesh: Pressure *(cell-centered)*, VelocityX *(node-centered)*. The C++ code to store the HDF5 data is included to demonstrate the types of arrays that would be used with the provided schema. Most users of XDMF would already have HDF5 files and would only need to create the XML schema.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include <hdf5.h>

// The number of cells in the X, Y dimensions
#define NX 30
#define NY 20

void
write_hdf5_data()
{
    hid_t     file_id;
    file_id = H5Fcreate("xdmf2d.h5", H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);

    // Create the coordinate data.
    float *x = (float *) malloc((NX+1)*(NY+1) * sizeof(float));
    float *y = (float *) malloc((NX+1)*(NY+1) * sizeof(float));
    int ndx = 0;
    for (int j = 0; j < NY+1; j++)
    {
        float yt = float(j) / float(NY);
        float angle = yt * M_PI;
        for (int i = 0; i < NX+1; i++)
        {
            float xt = float(i) / float(NX);
            float R = (1.-xt)*2. + xt*5.;

            x[ndx] = R * cos(angle);
            y[ndx] = R * sin(angle);
            ndx++;
        }
    }

    // Create the scalar data.
    float *pressure = (float *) malloc(NX*NY * sizeof(float));

    for (int j = 0; j < NY; j++)
    {
        for (int i = 0; i < NX; i++)
        {
            int ndx = j * NX + i;
            pressure[ndx] = (float) j;
        }
    }

    float *velocityx = (float *) malloc((NX+1)*(NY+1) * sizeof(float));

    for (int j = 0; j < NY+1; j++)
    {
        for (int i = 0; i < NX+1; i++)
        {
            int ndx = j * (NX+1) + i;
            velocityx[ndx] = (float) i;
        }
    }
```

```
    // Write the data file.
    hid_t     dataset_id, dataspace_id;
    hsize_t   dims[3];
    herr_t    status;
    const char *coordNames[] = {"/X", "/Y"};

    /* Write separate coordinate arrays for the x and y coordinates. */
    for(int did = 0; did < 2; ++did)
    {
        dims[0] = (NY + 1);
        dims[1] = (NX + 1);
        dataspace_id = H5Screate_simple(2, dims, NULL);

        dataset_id = H5Dcreate(file_id, coordNames[did], H5T_NATIVE_FLOAT, dataspace_id,
                               H5P_DEFAULT);

        status = H5Dwrite(dataset_id, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL,
                          H5P_DEFAULT, did == 0 ? x : y);

        status = H5Dclose(dataset_id);

        status = H5Sclose(dataspace_id);
    }

    // Write the scalar data.
    dims[0] = NY;
    dims[1] = NX;
    dataspace_id = H5Screate_simple(2, dims, NULL);

    dataset_id = H5Dcreate(file_id, "/Pressure", H5T_NATIVE_FLOAT,
                           dataspace_id, H5P_DEFAULT);

    status = H5Dwrite(dataset_id, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL,
                      H5P_DEFAULT, pressure);

    status = H5Dclose(dataset_id);

    status = H5Sclose(dataspace_id);

    dims[0] = NY + 1;
    dims[1] = NX + 1;
    dataspace_id = H5Screate_simple(2, dims, NULL);

    dataset_id = H5Dcreate(file_id, "/VelocityX", H5T_NATIVE_FLOAT,
                           dataspace_id, H5P_DEFAULT);

    status = H5Dwrite(dataset_id, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL,
                      H5P_DEFAULT, velocityx);

    status = H5Dclose(dataset_id);

    status = H5Sclose(dataspace_id);

    // Free the data.
    free(x);
    free(y);
    free(pressure);
    free(velocityx);

    status = H5Fclose(file_id);
}

void
write_xdmf_xml()
{
    FILE *xmf = 0;

    /*
     * Open the file and write the XML description of the mesh..
     */
    xmf = fopen("xdmf2d.xmf", "w");
    fprintf(xmf, "<?xml version=\"1.0\" ?>\n");
    fprintf(xmf, "<!DOCTYPE Xdmf SYSTEM \"Xdmf.dtd\" []>\n");
    fprintf(xmf, "<Xdmf Version=\"2.0\">\n");
    fprintf(xmf, " <Domain>\n");
    fprintf(xmf, "   <Grid Name=\"mesh1\" GridType=\"Uniform\">\n");
    fprintf(xmf, "     <Topology TopologyType=\"2DSMesh\" NumberOfElements=\"%d %d\"/>\n", NY+1, NX+1);
    fprintf(xmf, "     <Geometry GeometryType=\"X_Y\">\n");
    fprintf(xmf, "       <DataItem Dimensions=\"%d %d\" NumberType=\"Float\" Precision=\"4\" Format=\"HDF\">\n", (NY+1), (NX+1));
    fprintf(xmf, "        xdmf2d.h5:/X\n");
    fprintf(xmf, "       </DataItem>\n");
    fprintf(xmf, "       <DataItem Dimensions=\"%d %d\" NumberType=\"Float\" Precision=\"4\" Format=\"HDF\">\n", (NY+1), (NX+1));
    fprintf(xmf, "        xdmf2d.h5:/Y\n");
    fprintf(xmf, "       </DataItem>\n");
    fprintf(xmf, "     </Geometry>\n");
    fprintf(xmf, "     <Attribute Name=\"Pressure\" AttributeType=\"Scalar\" Center=\"Cell\">\n");
    fprintf(xmf, "       <DataItem Dimensions=\"%d %d\" NumberType=\"Float\" Precision=\"4\" Format=\"HDF\">\n", NY, NX);
    fprintf(xmf, "        xdmf2d.h5:/Pressure\n");
    fprintf(xmf, "       </DataItem>\n");
    fprintf(xmf, "     </Attribute>\n");
    fprintf(xmf, "     <Attribute Name=\"VelocityX\" AttributeType=\"Scalar\" Center=\"Node\">\n");
    fprintf(xmf, "       <DataItem Dimensions=\"%d %d\" NumberType=\"Float\" Precision=\"4\" Format=\"HDF\">\n", NY+1, NX+1);
    fprintf(xmf, "        xdmf2d.h5:/VelocityX\n");
    fprintf(xmf, "       </DataItem>\n");
    fprintf(xmf, "     </Attribute>\n");
    fprintf(xmf, "   </Grid>\n");
    fprintf(xmf, " </Domain>\n");
    fprintf(xmf, "</Xdmf>\n");
    fclose(xmf);
}

int
main(int argc, char *argv[])
{
    write_hdf5_data();
    write_xdmf_xml();

    return 0;
}
```

The XML schema

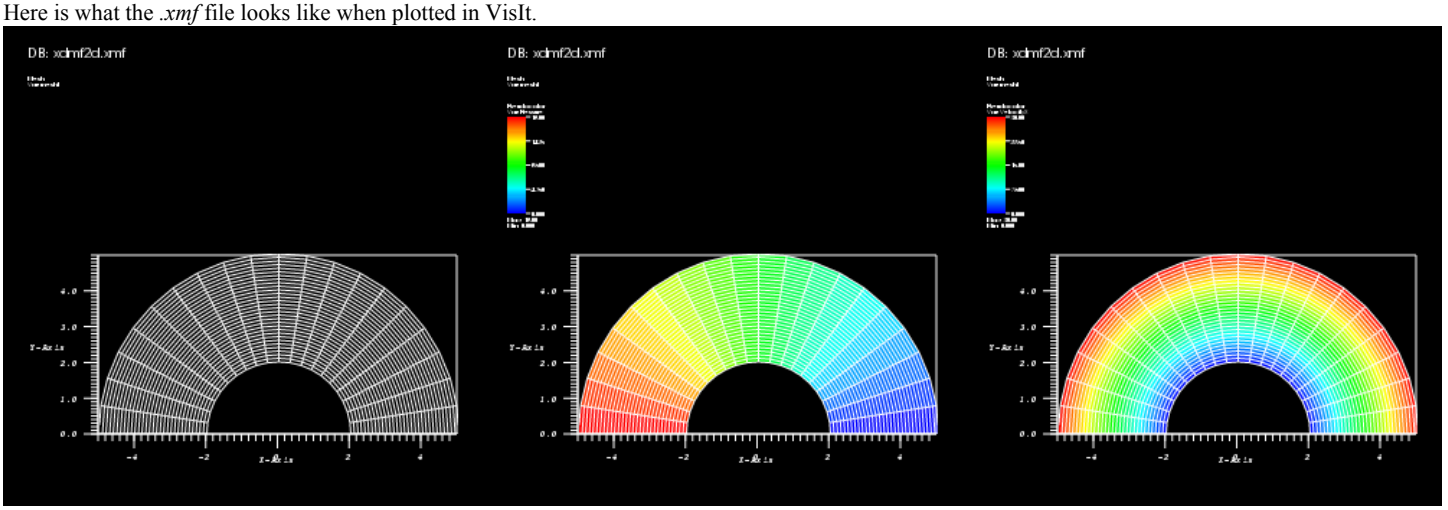
The XML schema produced by the program indicates that data should be read from the *xdmf2d.h5* file. The mesh is *2DSmesh* indicating that the mesh is 2D and structured. The mesh coordinates use the *X_Y* syntax, which indicates that each coordinate field is stored in a different HDF5 array. The *Attribute* tags in the XML specify both of the fields that are defined on the mesh. Note that the order of the coordinates when using HDF5 and the XML file is: Z, Y, X *(with Z being optional)*.

Another important point is that you should take care to specify the number of nodes for most dimensions of the mesh. The number of cells only seems important for specifying the dimensions of cell-centered variables. The (NX+1) in the above source code example program means the number of nodes in X since NX is the number of cells in X. Thus (NX+1) is used to write any node-sized value *(with respect to X)* to the XML file.

```
<?xml version="1.0" ?>
<!DOCTYPE Xdmf SYSTEM "Xdmf.dtd" []>
<Xdmf Version="2.0">
 <Domain>
   <Grid Name="mesh1" GridType="Uniform">
     <Topology TopologyType="2DSMesh" NumberOfElements="21 31"/>
     <Geometry GeometryType="X_Y">
       <DataItem Dimensions="21 31" NumberType="Float" Precision="4" Format="HDF">
        xdmf2d.h5:/X
       </DataItem>
       <DataItem Dimensions="21 31" NumberType="Float" Precision="4" Format="HDF">
        xdmf2d.h5:/Y
       </DataItem>
     </Geometry>
     <Attribute Name="Pressure" AttributeType="Scalar" Center="Cell">
       <DataItem Dimensions="20 30" NumberType="Float" Precision="4" Format="HDF">
        xdmf2d.h5:/Pressure
       </DataItem>
     </Attribute>
     <Attribute Name="VelocityX" AttributeType="Scalar" Center="Node">
       <DataItem Dimensions="21 31" NumberType="Float" Precision="4" Format="HDF">
        xdmf2d.h5:/VelocityX
       </DataItem>
     </Attribute>
   </Grid>
 </Domain>
</Xdmf>
```

The results

Here is what the *.xmf* file looks like when plotted in VisIt.



Categories: User Documentation | Reading/Writing Data

- This page was last modified 18:26, 29 April 2010.