Investigations into using HDF5 for product model data – B-Spline and Cartesian Point data in HDF5

Vailin Choi and Mike Folk

The HDF Group

January 2007

1 Introduction

The work reported here is part of an investigation into the viability of using a general purpose format such as HDF5 for preserving engineering data, in particular product model data. This work is part of a project at the National Center for Supercomputing Applications (NCSA) and The HDF Group, supported by the National Archives and Records Administration (NARA), to investigate the use of scientific data formats to support long term preservation of certain collections of high volume, complex federal records.

For this study, NARA provided a sample collection that exemplifies the types of engineering data that it must be able to archive – a collection of engineering data relating to ship called Torpedo Weapons Receiver (TWR). The overall goal of the project is

> *Using NARA's TWR collection, investigate the possibilities and limitations of using HDF5 to preserve and access heterogeneous collections of records, with special attention to STEP data.*

The work reported on in this paper focuses on particular types of common product model data objects found in the TWR engineering data: Cartesian points and B-splines. Both are of interest here because they can occur in very large volumes, and thus challenge the scalability of the existing formats.

The objectives of this work are to (a) map datatypes and structures found in the TWR collection to HDF5 objects and structures, exploiting features of HDF, and (b) assess the benefits and costs of storing this data in HDF5.

2 Background

Current formats for managing product model data are based on the STEP format[1]. The semantics of data stored in STEP is described in the EXPRESS data description language, an object-oriented information modeling language.

Currently, EXPRESS objects are instantiated in STEP as ASCII text. Alternative XML implementations are also available. STEP is a successful ISO standard[1], and is well accepted and supported. However, ASCII-based implementations do not adapt well for highly voluminous, complex data, such as data used finite element analysis and computational fluid dynamics applications. STEP is also not suitable for the heterogeneous supporting data found in collections of product model data, such as digital photographs, blueprints, and formatted text.

These shortcomings of STEP have led to a number of attempts to develop an alternate binary format. One such effort is EuroSTEP's EXPRESS/Binary Project, whose goal is to "standardize a mapping into a more efficient (i.e. binary) file

---

[1] STEP (**ST**andard for the **E**xchange of **P**roduct data) refers to the "International Standard ISO 10303 *Industrial systems and integration - Product data representation and exchange*."

representation" for EXPRESS data. It is hoped that this alternate file representation may satisfy the needs of several industries such as Thermal Analysis and Finite Element Analysis, which utilize very large datasets for their information models. [1][2] [3]

The EuroSTEP Project identified a number of criteria for such a new file representation:

- Available everywhere – open source, free
- Architecture-independent specification and extensibility
- Platform-independent implementation
- General purpose capability
  - Custom Structure Definition/Representation
  - Alignment with EXPRESS (datatypes, structures, rules?)
  - Support for mixed content
- Fast read/write performance
- Out-of-core data access (partial I/O)
- Large files support, compression
- Applicability for long term archiving
- Pedigree, viability, widespread usage and some standardization
- Support, training, high quality documentation

The EXPRESS/Binary Project selected HDF5 (Hierarchical Data Format) as the binary format for initial investigation. The project has developed mappings from EXPRESS to HDF5, and has recently presented a prototype of these mappings to the International Standards Organization (ISO).

The work reported here builds on the EuroSTEP Project, and was done in collaboration with the Project. It uses some of the EXPRESS-HDF5 mappings to investigate the viability of using HDF5 as a binary format for certain product model data.

3 Mapping of EXPRESS Data to HDF5

The two basic constructs of the EXPRESS language are **entities** and **types**. **Types** in EXPRESS include primitive types such as *integer* and *string*, as well as *enumeration*, *aggregation*, and *select* types. An **entity**, which represents a real-world object, is composed of named properties called attributes, each of which has an associated data type. An instance of an entity has an identifier as well as values associated with each declared attribute. A **schema** is a collection of EXPRESS entity declarations while a **data population** consists of entity instances that conform to the entity data type declarations.

HDF5 is a general purpose file format for storing large or complex volumes of scientific data. It consists of two primary objects: datasets and groups. "A **dataset** is essentially a multidimensional array of data elements, and a group is a structure for organizing objects in an HDF5 file. Datasets and groups can be stored in different ways so as to improve storage or I/O efficiency, and both can have associated metadata in the form of HDF5 **attributes**. Using these as basic building blocks, one can create and store almost any kind of scientific data structure in HDF5, such as images, arrays of vectors, and structured and unstructured grids." [4]

The representation of EXPRESS-driven data using HDF5 is specified by relating EXPRESS data concepts to HDF5 data concepts. An EXPRESS entity is similar in structure to an HDF5 compound datatype. The components of an EXPRESS entity are called "attributes". The corresponding components in an HDF5 compound datatype are called "fields," so each attribute in an EXPRESS entity corresponds to a

field in an HDF5 compound type. The actual datatypes of EXPRESS entity attributes correspond to HDF5 atomic types. Hence, an EXPRESS entity declaration is represented as an HDF5 compound data type whose fields consist of HDF5 atomic types.

The collection of instances of an EXPRESS entity type is treated as a dataset in HDF5 and each population of an EXPRESS schema is represented as an HDF5 group. Details of mapping from EXPRESS data to HDF5 is described in *EXPRESS Data as HDF5 Mapping Specification Version 0.3.1*. [3]

4 Testing Data

4.1 Collections

Two sets of data are used for this study:

A. The first set is a collection provided by NARA that consists of data concerning the US Navy on Torpedo Weapon Retriever (TWR) 841, which is a 120 foot boat used in support of underwater acoustic submarine operations and torpedo exercises. The design data in the collection covers the hull, main deck, lower platform, and all major transverse and longitudinal bulkheads in the hold. There are altogether 26 images and 3 STEP files in this collection. The three STEP files are AP216 files which are populated based on the *ship_moulded_form* schema.

B. The second set is obtained from a colleague from Electric Boat, consisting of four AP209 files that are populated based on the *structural_analysis_design* schema. They contain geometry, finite element data and results information.

4.2 B-splines and Cartesian points

To investigate the appropriateness of a binary format such as HDF5, we need to identify EXPRESS entities that may create difficulties when instantiated in STEP, and seem not likely to create the same difficulties when instantiated in HDF5. In the two collections, two particular entities were identified: B_SPLINE_SURFACE_WITH_KNOTS, and CARTESIAN_POINT.

A “b-spline” is an important mathematical function used to describe curves and surfaces, such as the surfaces of a ship’s hull. A b-spline is represented in STEP as a set of points through which the corresponding curve or surface is to be computed. The entity B_SPLINE_SURFACE_WITH_KNOTS occurs often in the two STEP collections. This entity was chosen for conversion into HDF5 because it contains a variety of data types, and can sometime be large, thus providing a good example in demonstrating the mapping from EXPRESS to HDF.

The entity B_SPLINE_SURFACE_WITH_KNOTS references another entity, CARTESIAN_POINT, as one of its attributes. The CARTESIAN_POINT entity is of interest in this study because there can often be a very large number of Cartesian points stored in a STEP file. Furthermore, Cartesian points are almost always referenced from some other entity instance in STEP files, and hence need to be accessed directly, or randomly.

4.3 Mappings from EXPRESS to HDF5

The entity B_SPLINE_SURFACE_WITH_KNOTS is represented as a compound type in HDF5. Table 1 lists this entity’s attributes with the associated EXPRESS

types and the corresponding HDF5 data types in the compound structure. The mapping for the CARTESIAN_POINT entity is shown in Table 2.

Attribute	EXPRESS types	HDF5 types
Name	label (STRING)	H5T_C_S1
u_degree	INTEGER	H5T_NATIVE_INT
v_degree	INTEGER	H5T_NATIVE_INT
control_points_list	LIST of LIST of cartesian_point (ENTITY)	variable length of variable length of H5T_NATIVE_INT or H5T_STD_REF_DSETREG
surface_form	B_spline_surface_form (ENUM)	Enum
u_closed	LOGICAL	Enum
v_closed	LOGICAL	Enum
self_intersect	LOGICAL	Enum
u_multiplicities	LIST of INTEGER	variable length of H5T_NATIVE_INT
v_multiplicities	LIST of INTEGER	variable length of H5T_NATIVE_INT
u_knots	LIST of parameter_value (REAL)	variable length of H5T_NATIVE_FLOAT
v_knots	LIST of parameter_value (REAL)	variable length of H5T_NATIVE_FLOAT
knot_spec	knot_type (ENUM)	Enum

Table 1 Mapping for B_SPLINE_SURFACE_WITH_KNOTS

Attribute	EXPRESS types	HDF5 types
name	label (STRING)	H5T_C_S1
coordinates	LIST [1:3] of length_measure (REAL)	Array of H5T_NATIVE_FLOAT

Table 2 Mapping for CARTESIAN_POINT

The mappings shown in Table 1 and Table 2 describe compound datatypes in HDF5. These two HDF5 datatypes are the datatypes of one-dimensional HDF5 arrays corresponding, respectively, to a collection of “B-spline surface with knots” and a collection of Cartesian points. In the B-spline array, there are many instances of B-splines. The “control_points_list” of each B-spline instance identifies the Cartesian points to be used as control points, and hence references points in the Cartesian points array. This relationship is illustrated in Figure 1.

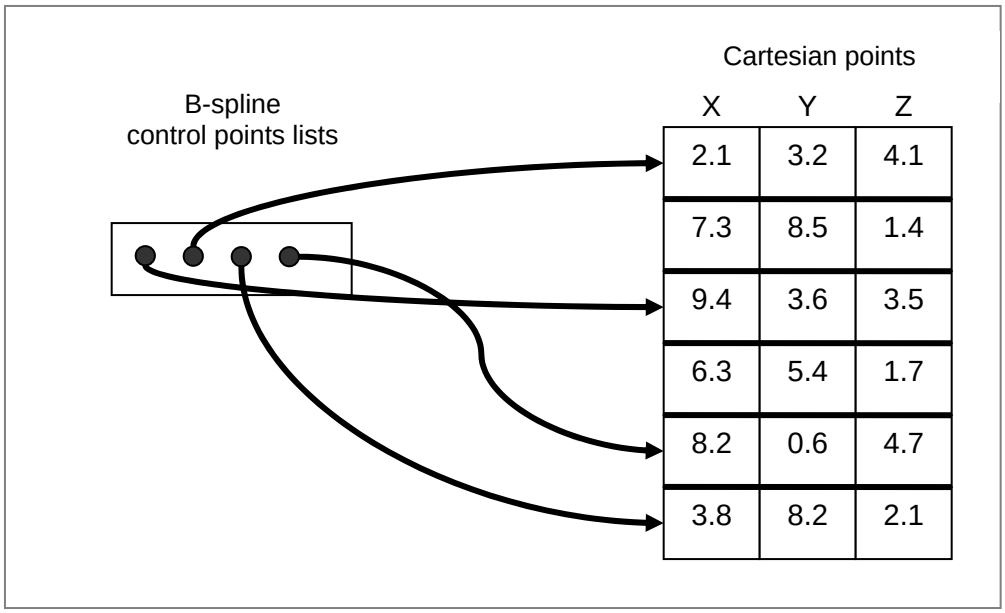


Figure 1. Relationship between B_SPLINE_SURFACE_WITH_KNOTS and CARTESIAN_POINT datasets.

4.4 Data characteristics

The number of B_SPLINE_SURFACE_WITH_KNOTS and CARTESIAN_POINT objects extracted from collections A and B are shown in Table 3.

file	total # of B_SPLINE_SURFACE_WITH_KNOTS	total # of Cartesian points referenced by B_SPLINE_SURFACE_WITH_KNOTS	total # of CARTESIAN_POINT
#A1	9	1,116	3,297
#A2	87	812	29,588
#A3	5	484	20,355
#B1	54	13,704	15,235
#B2	8	2,133	3,146
#B3	9	959	4,387
#B4	216	2,600	20,062

Table 3. Numbers of objects in files from collections A and B

5 Storage Formats in HDF5 -

Indexing versus regional references. The entity B_SPLINE_SURFACE_WITH_KNOTS has an attribute named *control_points_list*, which references the entity CARTESIAN_POINT. These references are implemented in two different ways in HDF5, as follows:

- *Method R* uses the HDF5 *regional reference*. A region reference is a data type that describes a region of a second dataset in the HDF5 file. In this instance, each control point in the *control_points_list* is mapped to a region in the dataset containing all of the Cartesian points. That region consists of the point being referred to.

- *Method I* uses *simple indexing*. In this case, instead of storing region references in the *control_points_list*, we simply stored the index (offset) of the Cartesian point in the Cartesian points dataset.

Method R is the most general way to map region references to HDF5. With this method, it is possible to store references consisting of any number of Cartesian points. But Method R has certain potential disadvantages for performance:

- Because region references are potentially variable in size, they are stored in a way that requires two accesses to an HDF5 file. Namely, a region reference in a dataset is stored as a pointer into another data structure (called a "heap"), which containis the action information associated with the region reference.
- Furthermore, because a region reference can be very complex, there is significant overhead associated with storing a region reference in HDF5.
- And finally, much of the data associated with region references is stored in a location in HDF5 files that cannot be compressed.

Method I, in contrast, is much less flexible, but has no associated overhead, allows direct access to the corresponding Cartesian point, and can be compressed in its entirety.

Effects of using compression. No matter what format is used, data compression can be beneficial in saving storage space and improving transmission time. In comparing a text-based format with a binary format, such as STEP with HDF5, there are some special factors to consider.

Although binary data generally is more space-efficient than text data, text tends to compress much better than binary data, and often ultimately takes up less space. This is particularly true for so-called real numbered, or floating point, data.

However, many text formats use variable-sized fields and records for storing numeric data, to avoid excessive file sizes. STEP is such a format. When variable-sized fields and records are used, it becomes difficult to know the location of an item in a STEP file, and hence any access to an object in a STEP file requires that the file be read from its start, up to the location of the desired object. This in turn, means that it would be difficult with the current STEP format to compress individual objects within a STEP file, and still achieve efficient access to individual items in the file. If compression is to be used with STEP, the entire file should be compressed, and if there is a need to access an individual record, the entire file would need to be uncompressed.

HDF5 was designed to avoid this problem. In HDF5, indexes are created to tell where the data records are, and the data records are stored separately from the indexes. Furthermore, in HDF5, the data records can be individually compressed without compressing the indexes, making it possible to access data by only uncompressing the data records, not the whole file. Finally, large data aggregates in HDF5 can also stored in individually-compressed "chunks", so that only those chunks need be uncompressed that contain the data records that are of interest.

For these reasons, and since our interest is in storing and accessing STEP data in which there are very large arrays of data, we only consider the case in which we would not compress a STEP file, but *would* compress individual objects in an HDF5 file.

For each of the variations above, those entity instances that have potentially high volume are stored with and without compression, namely B_SPLINE_SURFACE_WITH_KNOTS and CARTESIAN_POINT.

6 Results

Table 4 shows the results from the study, including the sizes of the original STEP files, and the sizes of the corresponding HDF5 files with and without compression.

File name	STEP	Method I		Method R	
		HDF	HDF Compression	HDF	HDF Compression
#A1	207,319	195,000	116,102	268,728	189,830
#A2	2,743,083	1,954,712	1,125,294	1,995,672	1,166,254
#A3	1,917,877	1,322,488	759,150	1,355,256	791,918
#B1	1,149,145	966,904	499,742	1,835,256	1,368,094
#B2	276,129	181,976	100,571	313,048	231,643
#B3	366,803	242,168	132,224	307,704	197,760
#B4	1,777,968	1,404,248	847,247	1,568,088	1,011,087

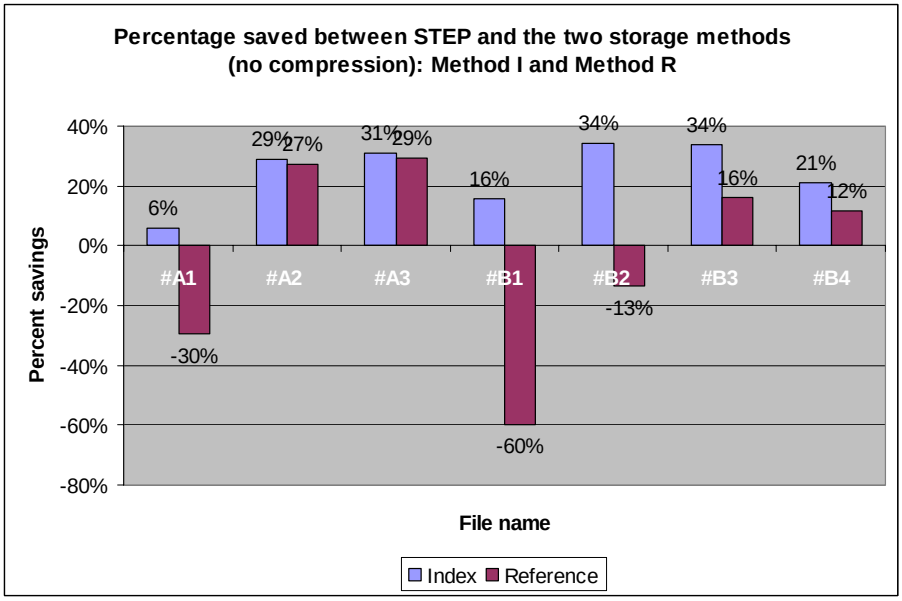
Table 4 File sizes for Method I in collection set A.

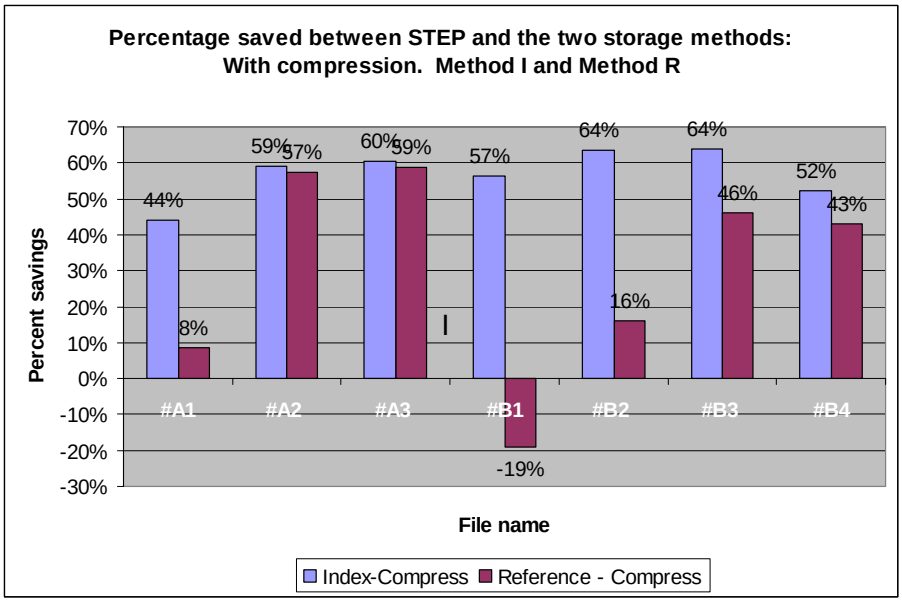
7 Analysis

The following observations are made from the resulting data:

7.1 Comparison between STEP and the two storage methods

The following two graphs show the percentage of space saved by using HDF5 storage rather than STEP, for both the indexing and region reference methods. As explained in Section 5, compression is used only with HDF5 files, not with the corresponding STEP files, because compressing STEP files would not permit partial file access, which is one of the primary reasons for using HDF5 files.





The graphs slow that HDF5 is generally superior to STEP for storage. Two exceptions to this are files A1 and B1 using region references. When compression is used, the differences are even more dramatic, showing that HDF5 with the indexing approach generally takes approximately half the space that STEP requires.

The two graphs also show that Method I, *indexing*, is a better choice than Method R, *regional reference*. This is more obvious for collection B, which may be due to the larger number of Cartesian points being referenced by B_SPLINE_SURFACE_WITH_KNOTS in this data set. For collection set A, the same scenario is observed for file #A1.

7.2 Other lessons

In the process of this study, other useful lessons were learned about the process of converting EXPRESS data to a binary format such as HDF5:

- All EXPRESS/STEP entities could be mapped to HDF5 in a straightforward manner.
- When converting large objects between formats, the best conceptual mapping (region references) was not the best in terms of storage and accessibility. Valuable improvements were achieved by taking the time to investigate available indexing as a storage option to using HDF5 region references.

8 Conclusion and future work

This study is an initial effort to explore the possibility of HDF5 as a binary representation for EXPRESS-driven data. The result shows that HDF5 is a feasible alternative to STEP, and can save considerable storage space when large numbers of Cartesian points are to be stored.

The next phase of this research will investigate another type of object that presents scalability challenges: finite element modeling (FEM) data. FEM is a very common method used in engineering design. The ability to understand materials behavior is greatly enhanced by running finite element models at increasingly higher resolutions, which can produce enormous amounts of data. Thus, as our computational resources

grow dramatically, we can expect FEM data volumes to grow in equal measure, challenging our ability to store and access such data in the STEP format. The next phase of this investigation will address the question of how HDF5 might accommodate this growth to provide effective storage and access.

9 References

[1] P. Wilson. STEP and EXPRESS. http://deslab.mit.edu/DesignLab/dicpm/step.html

[2] D. Rivers-Moore. XML and EXPRESS as schema definition languages. http://xml.coverpages.org/rivers-moore-k0603.html

[3] Welcome to EXPRESS-Binary. http://www.exff.org/express_binary/index.html

[4] What is HDF? http://hdf.ncsa.uiuc.edu.