THG Pilot Study: Dynamic Transformations to HDF5 Data

Ruth Aydt

Introduction

The HDF Group (THG) is conducting a pilot study to assess the feasibility of applying user-defined transform operations written in a dynamic programming language, initially Python, to data being read from an HDF5 file.

This work is being funded by a customer of THG. The general results of the study will be available on THG's website. At this time, there is no commitment by either THG or the customer to pursue the work further.

Interested parties are encouraged to provide feedback to THG on the ideas presented in this document by sending comments to the HDF Forum mailing list: hdf-forum@hdfgroup.org.

1 Background and Motivation

The HDF5 library provides APIs that allow applications to effectively and efficiently store data in the HDF5 binary format, and to access that data on a variety of platforms. HDF5 files are frequently shared among users with widely-varying interests, and often contain data that has value for decades or longer. HDF5's ability to manage large and complex data makes it ideal for storing data from instruments, experiments, and simulations.

THG customers have expressed the desire to manipulate the "raw" data stored in HDF5 files in a variety of ways as the data is read. Based on the needs of the application reading the data, different transformations may need to be applied to the data. Storing the raw data and applying transformations on-the-fly as the data is read, rather than saving both raw and transformed values, saves storage space. In addition, it allows transformations to be defined and refined at any time throughout the lifetime of the initial data product.

For example, an application writing data to an HDF5 file may record temperature using the Fahrenheit scale. One application using this data may want to report the temperature on the Celsius scale, while another application using the data may want to report using the Kelvin scale. One approach would be to store the data in all three temperature scales in the file, allowing the applications using the data to read and report the appropriate values. This approach incurs a storage space penalty, and the conversion to alternative scales would be added work for the recording program. Alternatively, the applications reading the data could each be responsible for reading the data from the HDF5 file where it is saved in degrees Fahrenheit, and then converting it into either Celsius or Kelvin. This seems reasonable if there is only one "reader" application for each temperature scale, but does not allow for easy reuse of the conversion routine if multiple applications would be applying the same conversions. Neither of these approaches easily supports reporting in a different scale at some future date.

2 Proposed "Transformers"

The pilot study will explore the feasibility of allowing conversions such as those described above (or more generally, transformations) to be dynamically defined and applied as the data is being read. The focus of the initial study will be on defining "transformers" using the dynamic object-oriented language Python. Python was chosen for a variety of reasons – it is open source, widely used on a variety of platforms, and has an excellent numerical package.

In the example given above, both a 1) Fahrenheit to Celsius transformer and a 2) Fahrenheit to Kelvin transformer would be written. The application reporting in Celsius would specify the first transformer be applied as the data is read, and the application reporting in Kelvin would use the second transformer. Assuming multiple applications might want to report in Celsius, they could share the common Python transformer. Perhaps later an application wants to process the data using the FooBar temperature scale – as long as a transformer could be written from Fahrenheit to FooBar, the application can easily take advantage of it. Transformers can be defined – and refined – at any time.

One can foresee multiple possible use cases for such transformers, including calibration of raw data based on collection instrument settings, conversions between currencies, binning numbers, and table lookup with the raw values serving as indices into the table values. It may also be desirable to apply a series of transformers to the data being read. The original data remains intact in the HDF5 file, and can be read directly without any the application of any transformers when desired.

3 Scope

For the pilot study, the term "raw form" shall refer to integer types, including packed fields of integers. "Transformed form" refers to both integer and floating point types, including n-bit integers where n is not a multiple of 8. In the pilot study, the datatypes will be scalar – compound types will not be included. Currently, THG is planning to implement value transforms by passing arrays of raw value elements to transform operations written in Python. Several additional requirements/limitations for converting raw values to transformed values are desired:

- The focus for applying these operations is on reading the raw values. Converting from transformed values to raw values (possibly during writing) is outside the scope of this task.
- The operations can be applied to raw values stored in HDF5 datasets, but will not be able to be applied to raw values stored in HDF5 attributes.
- There can be one or more operations applied to the raw values when converting them to transformed values (a "list of operations" to apply).
- There should be a simple way to access the raw values without applying transform operations (i.e. a straightforward way to bypass the data transform operation step).

4 Pilot Study Activities

THG will implement a framework for applying the data operations required to change raw values into transformed values, according to the requirements and limitations listed above.

The framework will support the application of “transformers” to the raw values stored in the HDF5 file. The transformers will perform the types of operations described above. These transformers will change the numeric value of the data element, and may possibly change the HDF5 datatype of the data element, but their focus will be on the numeric value change.

It is expected that the raw values will be stored in an HDF5 file using the smallest/best datatype that describes them, and tranformers will be applied to the raw values in order to coerce them into transformed values as they are read into user applications.

THG will investigate the use of Python to define the transformers. Prototype code will be used to evaluate the suitability of this approach for the data value and data type manipulations. Performance tests will be conducted as part of this pilot study. One design goal of the “transformer” framework is to allow for transformers in languages other than Python at a future date.