Tensor Data Model

Brett Viren

July 23, 2024

Contents

1 Introduction **1**

2 Generic tensor data model **2**

3 Persistent tensor data models **2**

4 Specific tensor data model **3**
4.1 pcarray 4
4.2 pcdataset 4
4.3 pcgraph 4
4.4 pcnamedset 4
4.5 pctree 5
4.6 trace 6
4.7 tracedata 6
4.8 frame 7
4.9 cluster 7
4.10 clnodeset 7
4.11 cledgeset 8

5 Tensor archive files **8**

6 Hierarchical files **8**

7 C++ API **8**
7.1 `TensorDM.h` 9

1 Introduction

The Wire-Cell Toolkit (WCT) defines and supports a *tensor data model*. This model is factored into two layers:

- The **generic tensor data model** is transiently represented as a concrete implementations of the `ITensorSet` and `ITensor` C++ data interface classes and serialized (persisted) according to this document.
- The **specific tensor data model** defines conventions on the generic tensor data model in order to map certain other data types to the generic tensor data model.

The meaning of the term "tensor" here is not that of the strict mathematical sense. It describes a collection of objects with an optional metadata object and an optional array object.

2 Generic tensor data model

The generic tensor data model maps between the transient `ITensorSet` and `ITensor` model and a persistent one. The elements of the tensor data model are:

- A **tensor set** is an aggregation of zero or more **tensors** and a optional **metadata object**.
- A **tensor** is the combination of an **array** and a **metadata object**.
- A **metadata object** associates **attributes** in a structure that follows the JSON data model.
- An **array** is a contiguous block of memory holding numeric data that is stored with associated **shape**, layout **order** and array element **type** and **size**. An array may be empty or null.

A tensor may also associate to an aggregate of other tensors as described below.

While the tensor data model supports the JSON data model for metadata objects, serializing to some formats are limited. In particular, HDF5 metadata "attributes" set must be a "flat object" with all values scalar data.

3 Persistent tensor data models

Two persistent variants are supported:

serial data is sent through WCT iostreams to/from Zip or Tar archive files.

hierarchical objects are persisted in a tree structure such as to/from HDF5 files.

In both persistent forms a hieachy is expressed. In the serial form, each `ITensorSet` or `ITensor` representation carries its location in the hierarchy by a path-like stream in the reserved **metadata object** attribute **datapath**. While, in the hierarchical form, the **datapath** forms a path through the tree structure to locate the object.

In the serial form, a complex objects may be represented as a tensor that references an aggregate of other tensors. This aggregation is represented as a list of **datapath** values held in a **metadata object attribute**. For example, a tensor set may provide a metadata object attribute called **tensors** with an array value of **datapath** elements to its tensors (see below). Other aggregations are described in the specific tensor data model below. The hierarchy form may utilize HDF5 object or region references.

For I/O optimization, the **serial** variant will also associate tensor set with its tensors using a file (in-archive) naming and ordering convention as illustrated:

```
tensorset_0_metadata   # the tensor set ident=0 metadata object
tensor_0_0_metadata    # the first tensor metadata object (no array)
tensor_0_1_metadata    # the second tensor metadata object
tensor_0_1_array       # the second tensor array object
tensor_0_2_metadata    # etc....
tensor_0_2_array
tensor_0_3_metadata
tensor_0_3_array
tensor_0_4_metadata
tensor_0_4_array
```

*The user may adopt a convention that **datapath** informally implies some association. For example a point cloud (see below) may result in the following **datapath** set:*

```
pointcloud/0           # main tensor datapath
pointcloud/0/arrays/t  # datapath to first PC attribute
pointcloud/0/arrays/x  # etc...
pointcloud/0/arrays/y  # etc...
pointcloud/0/arrays/z  # etc...
```

4 Specific tensor data model

The specific tensor data model maps additional meaning to one or more tensors in terms of transient WCT data types by defining a number of conventions on top of the generic tensor data model.

To start with, every tensor has provides an associated **datatype** attribute of value string. Only values listed in this document are in the model:

pcarray a `PointCloud::Array` (`pointclouds/<ident>/arrays/<name>`)

pcdataset a `PointCloud::Dataset` (`pointclouds/<ident>`)

pcgraph a `PointGraph` (`pointgraphs/<ident>` with `pointgraphs/<ident>/{nodes,edges}`)

pctree A `PointTree`

trace one `ITrace` as 1D array or multiple `ITrace` as 2D array. (`frames/<ident>/traces/<number>`)

tracedata tagged trace indices and summary data. (`frames/<ident>/tracedata`)

frame an `IFrame` as aggregate of traces and/or traceblocks. (`frames/<ident>`)

cluster an `ICluster` (`clusters/<ident>`)

clnodeset an array of attributes for set of monotypical `ICluster` graph nodes.

cledgeset an array describing a set of `ICluster` graph edges between all nodes of one type to all nodes of another.

Where pertinent, the recommended **datapath** root path for the type is given in parenthesis.

The tensor set has a **datatype** of **tensorset** and is merely a generic container of tensors produced in some context (eg an "event"). The **tensorset** may provide an optional attribute **tensors** to reference **datapath** of the tensors. For persistent serial files, such references are redundant with the file (in-archive) naming convention described below. For hierarchical files, such references will form "softlinks" or "aliases" if the format supports them (as does HDF5).

A tensor type may itself represent an **aggregate** of other tensors. The aggregate is defined by a **datatype** specific metadata attribute holding an array of the **datapath** of the agregated tensors.

The remaining sections describe additional requirements specific to for each **datatype**.

4.1 pcarray

The **datatype** of **pcarray** indicates a tensor representing one `PointCloud::Array`. The tensor array information shall map directly to that of `Array`. A **pcarray** places no additional requirements on its tensor MD.

4.2 pcdataset

The **datatype** of **pcdataset** indicates a tensor representing one `PointCloud::Dataset`. The tensor array shall be empty. The tensor MD shall have the following attributes:

arrays an object representing the named arrays. Each attribute name provides the array name and each attribute value provides a **datapath** to a tensor of type **pcarray** holding the named array. Additional user application `Dataset` metadata may reside in the tensor MD.

4.3 pcgraph

The **datatype** of **pcgraph** indicates a tensor representing a "point cloud graph". This extends a point cloud to include relationships between pairs of points. The array part of a **pcgraph** tensor shall be empty. The MD part of a **pcgraph** tensor shall provide reference to two **pcdataset** instances with the following MD attributes:

nodes a **datapath** refering to a **pcdataset** representing graph vertex features.

edges a **datapath** refering to a **pcdataset** representing graph edges and their features.

In addition, the **pcdataset** referred to by the **edges** attribute shall provide two arrays of integer type with names **tails** and **heads**. Each shall provide indices into the **nodes** point cloud representing the tail and head endpoint of graph edges. A node or edge dataset may be shared between different **pcgraph** instances.

4.4 pcnamedset

The **datatype** of **pcnamedset** indicates a tensor representing a `std::map<std::string, PointCloud::Dataset>`. The tensor array shall be empty. The tensor MD shall have the following attributes:

items an object representing the named point cloud set. Each attribute name provides the name of the point cloud and the value provides the **datapath** to a **pcdataset**.

4.5 pctree

> *All node-local point clouds of a given name must be comprised of arrays of the same names and types. An earlier data model called* ***pctreenode*** *did not carry this restriction but results in many small records which led to ruinously slow conversions.*

The **datatype** of **pctree** indicates a tensor representing an n-ary tree such as represented in C++ with `WireCell::NaryTree::Node` with a value type of `WireCell::PointCloud::Tree::Points`.

The **pctree** has an array part that represents the tree structure as a flattened **parentage map**. The metadata of **pctree** has these attributes:

pointclouds the point cloud datasets of the tree held as a **pcnamedset**

lpcmaps the local point cloud mapping into the concatenated **pointclouds** as a **dataset**.

To describe the **parentage map** array and the **pointclouds** and **lpcmaps** elements of a **pctree** consider the point cloud tree shown in figure 1. It consists of six nodes which are labeled in the order they are visited in a depth-first descent. Each node has two local point clouds of zero or more points. When represented as a `NaryTree::Node` with `Tree::Points` type, any PC with zero points are omitted but these empty point clouds may be conceptually included.

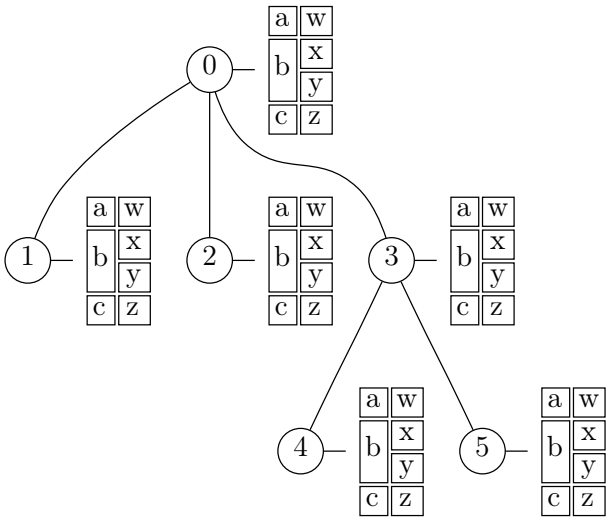


Figure 1: Example point cloud tree with nodes labeled in order of depth-first descent. Each node has a three local points clouds ("a", "b" and "c")

This tree will have a **parentage map** array as shown in table 1. Each element of the array represents a node as visited in depth-first descent order. The value of the element is the index of the parent of the node. To indicate the non-existent parent of a root node, the element value is set to the element index. This representation allows for a tree to have multiple roots (disjoint subtrees).

If we assume that only node 0 has non-empty PC "a", only nodes 1, 2 and 3 have non-empty PC "b" and only nodes 4 and 5 have non-empty PC "c" the **lpcmaps dataset** will be as illustrated in table 2. For simplicity, this example associates (non-empty) PCs with a "layer" in the tree. However, the representation is not limited and can represent point clouds of a given name that span layers.

Table 1: Example parentage map for the six-node example pctree.

0	1	2	3	4	5	node index
0	0	0	0	3	3	parent index

Table 2: Example local point clouds map for the six-node example pctree assuming a distribution of sizes of local point clouds across the nodes.

0	1	2	3	4	5	(node index)
n_0	0	0	0	0	0	a (sizes)
0	n_1	n_2	n_3	0	0	b (sizes)
0	0	0	0	n_4	n_5	c (sizes)

This example will have an entry "a" in **pointclouds** consisting of array "w" that has total size n_0, entry "b" with arrays "x" and "y" of total size $n_1 + n_2 + n_3$ and entry "c" with array "z" with total size "$n_4 + n_5$. From these concatenated **pointclouds** and the **lpcmaps** and the depth-first descent ordering of the **parentage map** the individual local point clouds may be reconstructed.

4.6 trace

The **datatype** of **trace** indicates a tensor representing a single `ITrace` or a collection of `ITrace` which have been combined.

The tensor array shall represent the samples over a contiguous period of time from traces.

The tensor array shall have dimensionality of one when representing a single `ITrace`. A collection of `ITrace` shall be represented with a two-dimensional array with each row representing one or more traces from a common channel. In such a case, the full trace content associated with a given channel may be represented by one or more rows.

The array element type shall be either `"i2"` (`int16_t`) or `"f4"` (`float`) depending on if ADC or signals are represented, respectively.

The tensor MD may include the attribute **tbin** with integer value and providing the number of sample periods (ticks) between the frame reference time and the first sample (column) in the array.

4.7 tracedata

The **datatype** of **tracedata** provides per-trace information for a subset of. It is similar to a **pcdataset** and in fact may carry that value as the **datatype** but it requires the following differences.

It defines additional MD attributes:

tag optional, a trace tag. If omitted or empty string, dataset must span total trace ordering.

The following array names are recognized:

chid channel ident numbers for the traces.

index provides indices into the total trace ordering.

summary trace summary values.

A chid value is require for every trace. If the **tracedata** has no **tag** then a **chid** array spanning the total trace ordering must be provided and neither **index** nor **summary** is recognized. If the **tracedata** has a **tag** it must provide an **index** array and may provide a **summary** array and may provide a **chid** array each corresponding to the traces identified by **index**.

4.8 frame

> *See the topic frames as tensors for details about representing frames with tensors.*

The **datatype** of **frame** represents an `IFrame`.

The tensor array shall be empty.

The tensor MD aggregates tensors of datatype **trace** and **tracedata** and provides other values as listed;

ident the frame ident number (required)

tags an array of string giving frame tags

time the reference time of the frame (required)

tick the sample period of the traces (required)

masks channel mask map (optional)

traces a sequence of datapath references to tensors of datatype **trace**. The order of this sequence, along with the order of rows in any 2D **trace** tensors determines the *total order of traces*.

tracedata a sequence of datapath references to tensors of datatype **tracedata**

In converting an `IFrame` to a **frame** tensor the sample values may be truncated to **type** `"i2"`.

A frame tensor of type `"i2"` shall have its sample values inflated to type `float` when converted to an `IFrame`.

4.9 cluster

The **datatype** of **cluster** indicates a tensor representing one `ICluster`. The tensor array shall be empty. The tensor MD shall have the following attributes:

ident the `ICluster::ident()` value.

nodes an object with attributes of *cluster array schema* **node type code** and values of a **datapath** of a **clnodeset**. The **node type code** is in single-letter string form, not ASCII char value.

edges an object with attributes of *cluster array schema* **edge type code** and values of a **datapath** of a **cledgeset**. The **edge type code** is in double-letter string form, not packed short integer.

The **cluster** tensor MD holds all references required to assemble the nodes and edges into an `ICluster`. The **nodes** and **edges** tensors hold no identifiers and require the **cluster** tensor to provide context.

4.10 clnodeset

The **datatype** of **clnodeset** indicates a tensor representing one type of node array in *cluster array schema*. The array is of type f8~~ and is 2D with each row representing one node and columns representing node attributes. The tensor MD may be empty.

4.11 cledgeset

The **datatype** of **cledgeset** indicates a tensor representing an edge array in *cluster array schema*. The array is of type `i4` and is 2D with each row representing one edge. First column represents edge tail and second column edge head. Values are row indices into a **clnodeset** array. The tensor MD may be empty.

5 Tensor archive files

WCT provides the DFP graph node components `TensorFileSink` and `TensorFileSource` that persist `ITensorSet` through an archive file (Zip or Tar, with optional compression) using WCT iostreams. The archive file will contain files with names matching these patterns:

```
<prefix>tensorset_<ident>_metadata.json
<prefix>tensor_<ident>_<index>_metadata.npy
<prefix>tensor_<ident>_<index>_array.json
```

The `<prefix>` is arbitrary, the `<index>` identifies a tensor set and `<index>` identifies a tensor in a set.

6 Hierarchical files

Currently, only the serial variant of the persistent data model is implemented. The general data model is intentionally similar to HDF5 and there is a conceptual mapping between the two:

- HDF5 group hierarchy ↔ `ITensor` metadata attribute providing a hierarchy path as array of string.
- HDF5 group ↔ No direct equivalent in that **datapath** patterns do not imply grouping but rather explicit metadata arrays do.
- HDF5 references ↔ Aggregation through array of **datapath** in metadata attribute.
- HDF5 dataset ↔ `ITensor` array.
- HDF5 dataspace and datatype ↔ `ITensor` methods `shape()`, `dtype()`, etc.
- HDF5 group or dataset attribute ↔ `ITensor` metadata attribute

7 C++ API

The WCT `aux` sub-package provides `WireCellAux/TensorDM.h` API for converting instances of WCT `IData` and more concrete classes to and from `ITensor` representation. This API is used in with components named like `XxxTensor` and `TensorXxx` to apply the conversions in the context of a WCT data flow graph. Additionally, the `sio` package provides `TensorFileSink` and `TensorFileSource` to serialize `ITensor` representations with files.

7.1 TensorDM.h

This API collects conversion methods which follow the forms:

```
ITensor::pointer as_tensor(<concrete types>);
ITensor::vector as_tensors(<concrete types>);
<concrete types> as_<concrete name>(<ITensor types>);
```