Apache Arrow Tensor Arrays

A toolchain for tensor transport and storage

Rok Mihevc, Alenka Frim

Apache Arrow committers

February 1, 2025

Overview

1. Arrow Extension Types
2. Fixed Shape Tensor
3. Variable Shape Tensor
4. FixedShapeTensorArray and NumPy ndarray
5. DLPack protocol

Arrow Extension Types

- Arrow allows for user extension types
- Arrow project also provides some "well-known extension types" or canonical extension types in the `arrow.*` namespace
- Current extension types are: `arrow.fixed_shape_tensor`, `arrow.variable_shape_tensor`, `arrow.json`, `arrow.opaque`, `arrow.bool8`

Fixed Shape Tensor - memory layout

type: extension<arrow.fixed_shape_tensor[value_type=int32, shape=[2,2]]>
pyarrow array: [[[1,2,3,4],[10,20,30,40],[100,200,300,400]]]

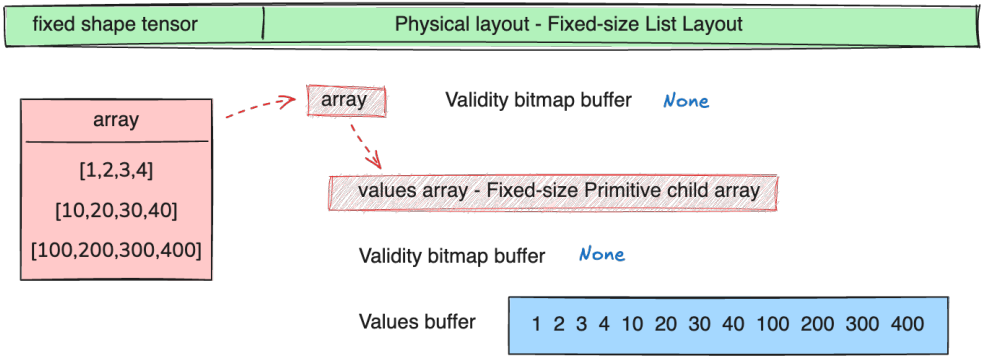


Figure: Fixed Shape Tensor memory layout

Fixed Shape Tensor

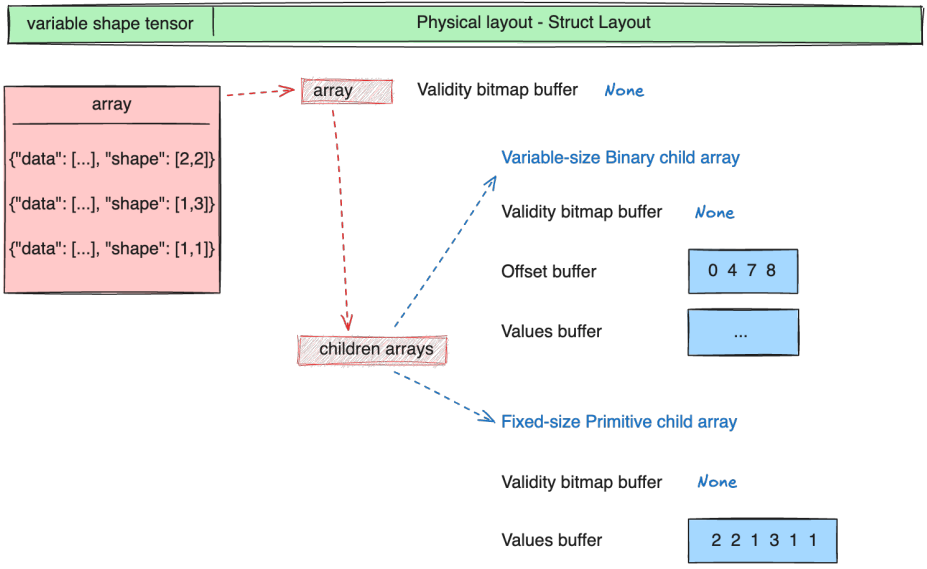
- Based on fixed size list array layout (`FixedSizeList[value_type, list_size]`)
- Every cell of the array is a multidimensional tensor of the same shape and type
- Type parameters: `data type` and `shape` of individual tensor elements
- Optional parameters: `dim_names`, `permutation`
- Elements in a fixed shape tensor extension array are stored in row-major/C-contiguous order

FixedShapeTensorArray metadata serialized

Examples

```
{
    "shape":[100, 200, 500 ],
    "dim_names": ["C", "H", "W"],
    "permutation": [2, 0, 1]
}
```

Variable Shape Tensor - memory layout



Variable Shape Tensor

- Based on struct array layout (`StructArray[List[value_type], FixedSizeList<int32>[ndim]]`)
- Every cell of the array is a multidimensional tensor with the same type and number of dimensions
- Type parameters: `data type`
- Data are stored as `StructArray`
  - `data` is a List holding tensor elements
  - `shape` is a `VariableSizeList<int32>[ndim]`
- Optional parameters: `dim_names`, `permutation` and `uniform_shape`
- Elements in a variable shape tensor extension array are stored in row-major/C-contiguous order

FixedShapeTensorArray metadata serialized

Examples

```
{
    "dim_names": ["H", "W", "C"],
    "uniform_shape": [400, null, 3],
    "permutation": [2, 0, 1]
}
```

Create a FixedShapeTensorArray

Examples

```
>>> import pyarrow as pa
>>> tensor_type = pa.fixed_shape_tensor(pa.int32(), (2, 2))
>>> arr = [[1, 2, 3, 4], [10, 20, 30, 40], [100, 200, 300, 400]]
>>> storage = pa.array(arr, pa.list_(pa.int32(), 4))
>>> tensor_array = pa.ExtensionArray.from_storage(tensor_type, storage)
```

Create a FixedShapeTensorArray

Examples

```
>>> tensor_array
<pyarrow.lib.FixedShapeTensorArray object at ...>
[
  [
    1,
    2,
    3,
    4
  ],
  ...
]
```

Move to NumPy ndarray

Examples

```
>>> tensor_array.to_numpy_ndarray()
array([[[  1,   2],
        [  3,   4]],

       [[ 10,  20],
        [ 30,  40]],

       [[100, 200],
        [300, 400]]], dtype=int32)
```

Move back to PyArrow

Examples

```
>>> pa.FixedShapeTensorArray.from_numpy_ndarray(
...       tensor_array.to_numpy_ndarray()
... )
<pyarrow.lib.FixedShapeTensorArray object at ...>
[
  [
    1,
    2,
    3,
    4
  ],
  ...
]
```

DLPack protocol

- Enables device aware data interchange between array/tensor libraries
- Currently producer side of DLPack implemented for pyarrow Array
- Future plan: Implementation of producing and consuming part for `Tensor` class and `FixedShapeTensorArray.to_tensor()` method to connect `FixedShapeTensorArray` with libraries supporting DLPack (NumPy, CuPy, Tensorflow, PyTorch, JAX, MXNet, TVM, mpi4py, Paddle, etc.)

The End