Low-Level Containers

`array` and `gridded` Containers

Jeremy Lloyd Conlin

jlconlin@lanl.gov

Los Alamos National Laboratory

October 30, 2014

Array Container

- Arrays can be nested—require indices to map child array to parent array
- Necessary child elements are defined by combinations of attributes

Array Container
Attributes

storageOrder `"row-major"` (default) or `"column-major"`

symmetry `"upper"` (default) or `"lower"` indicating triangular symmetry and representation

shape Comma-separated list of integers representing the length of the array along each dimension

Array Container
Attributes

compression Indication of sparseness. Allowed values:

- **none** (default) All values are explicitly given
- **coordinate** Assign coordinates to every non-zero value
- **upper-triangular** Only store the non-zero upper portion of array
- **lower-triangular** Only store the non-zero lower portion of array
- **diagonal** Non-zero values are in bands parallel to main diagonal.
- **embedded** Non-zero portion of array are structured as blocks
- **flattened** Used for higher-dimensional arrays

indices Needed for `embedded` arrays

Symmetry

- Don't need to store redundant data
- Only kind of symmetry is upper- and lower-triangular
- Must be careful with combination of upper/lower symmetry and row/column major storageOrder

  ```
  <tag xData="array"symmetry="upper" storageOrder="row-major">
  ```

  has equivalent <values...> element as

  ```
  <tag xData="array" symmetry="lower" storageOrder="column-major">
  ```

- Important not to confuse with triangular compression

Symmetry

$$S = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 6 & 7 \\ 3 & 6 & 8 & 9 \\ 4 & 7 & 9 & 10 \end{bmatrix}$$

Visual Symmetry

```
<tag xData="array" shape="4,4"
    symmetry="upper"
    storageOrder="row-major">
  <values xData="numeric">
    1  2  3  4
       5  6  7
          8  9
             10
</values></tag>
```

Symmetry

$$S = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 6 & 7 \\ 3 & 6 & 8 & 9 \\ 4 & 7 & 9 & 10 \end{bmatrix}$$

Visual Symmetry

```
<tag xData="array" shape="4,4"
    symmetry="upper"
    storageOrder="row-major">
  <values xData="numeric">
    1  2  3  4
       5  6  7
          8  9
             10
</values></tag>
```

Symmetry

```
<tag xData="array" shape="4,4"
    symmetry="upper"
    storageOrder="row-major">
  <values xData="numeric">
  1  2  3  4  5  6  7  8  9 10
  </values>
</tag>
```

Sparse Storage

- Avoid storing zeros
- Should use standard formats where available

compression `none` indicates no compression; i.e., full, explicit storage of every value

Coordinate

- Straightforward avoidance of storing zeros—works for every 1-D & 2-D arrays, but may not be most efficient
- Need to store three things for every array element:
    1. Numerical value
    2. Row index
    3. Column index

Coordinate

2-D Example

$$C = \begin{bmatrix} 1 & 1 & 0 & 3 & 0 \\ 2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ 4 & 0 & 2 & 7 & 0 \\ 0 & 8 & 0 & 0 & -5 \end{bmatrix}$$

```
<tag xData="array" shape="5,5" format="coordinate">
    <values xData="numeric">
        1 -1 -3 -2  5  4  6  4 -4  2  7  8 -5</values>
    <rowIndex xData="numeric">
        0  0  0  1  1  2  2  2  3  3  3  4  4 </rowIndex>
    <columnIndex xData="numeric">
        0  0  0  1  1  2  2  2  3  3  3  4  4</columnIndex>
</tag>
```

Coordinate

1-D Example

$$V = \begin{bmatrix} 0 & 0 & 1 & 2 & 3 & 0 & 0 \end{bmatrix}$$

```
<tag xData="array" shape="7" format="coordinate">
    <values xData="numeric"> 1 2 3</values>
    <rowIndex xData="numeric"> 0 0 0</rowIndex>
    <columnIndex xData="numeric"> 2 3 4</columnIndex>
</tag>
```

Triangular

- Only store the non-zero elements
- No indexing required
- Important not to confuse with triangular symmetry

Triangular

Examples

$$U = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 10 \end{bmatrix} \qquad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 10 \end{bmatrix}$$

```
<tag xData="array" shape="4,4"
    compression="upper-triangular">
  <values xData="numeric">
    1  2  3  4
       5  6  7
          8  9
             10
</values></tag>
```

```
<tag xData="array" shape="4,4"
    compression="lower-triangular">
  <values xData="numeric">
    1
    2 3
    4 5 6
    7 8 9 10
</values></tag>
```

Diagonal

$$D = \begin{bmatrix} \color{red}{1} & 0 & \color{blue}{3} & 0 \\ \color{purple}{1} & \color{red}{2} & 0 & \color{blue}{4} \\ 0 & \color{purple}{2} & \color{red}{0} & 0 \\ 0 & 0 & \color{purple}{3} & \color{red}{4} \end{bmatrix}$$

- Store the non-zero diagonal bands
- The "distance" for a diagonal is how many diagonals from the main diagonal.

Diagonal
Example: Implicit

$$D = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 1 & 2 & 0 & 4 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 \end{bmatrix}$$

```xml
<tag xData="array" shape="4,4" compression="diagonal">
    <diagonals xData="numeric" shape="3,4">
        1  1  3
        2  2  4
        3  0  0
        0  4  0
    </diagonals>
    <distance xData="numeric"> -1 0 2 </distance>
</tag>
```

Diagonal

Example: Explicit

$$D = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 1 & 2 & 0 & 4 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 \end{bmatrix}$$

```
<tag xData="array" shape="4,4" compression="diagonal">
    <diagonal xData="numeric" distance="-1"> 1 2 3</diagonal>
    <diagonal xData="numeric" distance="0"> 1 2 0 4</diagonal>
    <diagonal xData="numeric" distance="2"> 3 4</diagonal>
</tag>
```

Embedded

$$C = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 & 10 \\ 0 & 0 & 11 & 12 & 13 \end{bmatrix} \qquad C = \begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}$$

- Useful when arrays can be broken up into multiple sub arrays
- Simply allow for the `xData="array"` to be nested

Embedded

Example

$$C = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 \\ 0 & 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 & 10 \\ 0 & 0 & 11 & 12 & 13 \end{bmatrix}$$

$$C = \begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix}$$

```
<tag xData="array" shape="5,5" format="embedded">
    <sub xData="array" shape="2,2" indices="0,0">
        <values xData="numeric">
            1 2
            3 4 </values></sub>
    <sub xData="array" shape="3,3" indices="2,2">
        <values xData="numeric">
             5  6  7
             8  9 10
            11 12 13 </values></sub> </tag>
```

Flattened

- No standard for sparse storage for arrays with dimensions >2
- Data stored:
  - Row index
  - Column index
  - How many numbers follow
- Examples are modified from original LLNL idea

Flattened

Example: 1-D

$$V = \begin{bmatrix} 0 & 0 & 1 & 2 & 3 & 0 & 0 \end{bmatrix}$$

```
<tag xData="array" shape="7" format="flattened">
    <values xData="numeric">
        0 2 3  1 2 3</values>
</tag>
```

Flattened
Example: Implicit

$$C = \begin{bmatrix} 1 & 1 & 0 & 3 & 0 \\ 2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ 4 & 0 & 2 & 7 & 0 \\ 0 & 8 & 0 & 0 & -5 \end{bmatrix}$$

Implicit

```
<tag xData="array" shape="5,5" format="flattened">
    <values xData="numeric">0 0 2  1 1
                            0 3 4  3 0 2 5
                            2 2 3  4 6 4
                            3 0 4  4 0 2 7
                            4 1 1  8
                            4 4 1 -5 </values></tag>
```

Flattened

Example: Explicit

$$C = \begin{bmatrix} 1 & 1 & 0 & 3 & 0 \\ 2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 4 & 6 & 4 \\ 4 & 0 & 2 & 7 & 0 \\ 0 & 8 & 0 & 0 & -5 \end{bmatrix}$$

Explicit

```
<tag xData="array" shape="5,5" format="flattened">
    <values xData="numeric" rowIndex="0" columnIndex="0" N="2">  1 1      </values>
    <values xData="numeric" rowIndex="0" columnIndex="3" N="4">  3 0 2 5 </values>
    <values xData="numeric" rowIndex="2" columnIndex="2" N="3">  4 6 4    </values>
    <values xData="numeric" rowIndex="3" columnIndex="0" N="4">  4 0 2 7 </values>
    <values xData="numeric" rowIndex="4" columnIndex="1" N="1">  8        </values>
    <values xData="numeric" rowIndex="4" columnIndex="4" N="1"> -5        </values>
</tag>
```

Discussion

- Other possible formats?
    - Many other "standard" formats
- Should some formats be removed?