Java Scientific Containers - an open source generic large data library for visualization applications

Piotr Wendykier*, Bartosz A. Borucki, Krzysztof S. Nowiński (IEEE Member)

Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw

ABSTRACT

Java Scientific Containers (JSciC) is an open source Java library providing a generic data type for scientific datasets, primarily for visualization systems applications. A generic concept of field represents the dataset with three components - structure, geometry and values - allowing to represent the majority of numerical data used in the scientific environment. Not only regular and irregular (unstructured) grids with implicit or explicit geometry are supported, but also multiple numeric data components (multivariate fields) with scalar or vector elements. Flexible time support is provided by independent time steps. Additional functionality, including interpolation, slicing and statistics is also available in the library. Parallel execution arithmetic on data components is incorporated with full physical units support. JSciC library serves as a base for generic data representation in VisNow platform.

Keywords: JSciC, scientific visualization, generic data types, Java, arithmetic, unit support

1 INTRODUCTION

Java Scientific Containers (JSciC) is an open source pure Java library for generic representation of scientific data. Since JSciC originated from VisNow [1] - a generic visualization platform in Java - it is significantly focused on the generic data types for visualization systems, with multiple integrated analytics or simple processing tools and numeric I/O data format.

One of the key components of each universal visualization system is an internal generic data type. In order to support multiple datasets from different fields of applications and from multiple file formats in a single visualization system it is crucial to translate every supported external format to a single internal data type that can be processed in a unified environment by the components of a visualization system.

Scientific visualization is the domain where datasets are (usually) embedded in real space and described by multiple variables, often time dependent. A generic data type applicable for such solutions must cover at least the following functionalities: representation of one-, two- or three-dimensional geometry, representation of multivariate data, support for different numeric data types, support for time variability and support for different structural representations.

In practice, a generic data type can be implemented as a multilevel object hierarchy, usually with several alternative types at each hierarchy level.

2 BACKGROUND

The current standard of such generic approach to data types in visualization applications is the format of VTK library [3]. However, from technical perspective VTK is a C/C++ library and even though it has multiple wrappers or derived libraries for different programming languages, including Java, there is no popular native Java alternative. VTK includes a complete set of visualization functions, while this work focuses only on scientific data containers. In addition, a mixed language programming, such as using a VTK wrapper in Java, is more error prone and much harder to debug than pure Java solution. Performance comparison of JSciC and VTK is currently under development.

ImgLib2 [4] is a Java library for n-dimensional data representation and manipulation. It provides various implementations for pixel data in a discrete n-dimensional grid as well as popular image processing algorithms. However, ImgLib2 is not designed for scientific visualization applications and it does not support unstructured grids.

JSciC is our proposition of pure-Java implementation of a generic data type applicable for visualization purposes supporting large data structures. From technical perspective, due to Java object memory management, JSciC follows an assumption of storing numerical and indexation data in flat (one dimensional) arrays as an alternative to storing object hierarchy (e.g. field → cellset → cell → wall → edge → cell → wall → edge → node). As Java lacks true multidimensional arrays, most of numerical libraries in Java use similar approach to store multidimensional data. However, to overcome Java limitation of 2^{31} elements in a single array (equivalent of a 3D regular grid of 1290^3 scalar elements) JSciC uses JLargeArrays library [2] to store native memory arrays up to 2^{63} elements. At the same time JLargeArrays provides the functionality of numeric data types of logic, unsigned byte, short, int, long, float, double, float complex, double complex, String and Object. Benchmarks results [2] show that JLargeArrays generally outperforms both Fastutil library [5] and native Java arrays especially when multiple threads are used and array sizes are larger than 2^{31}.

From substantial perspective JSciC follows an assumption that to represent data in a generic manner it is required to define the generic field object and to provide three elements of the description - geometry, structure and values. The geometry part of the data object describes spatial location of points (nodes), the structure part defines relationships between nodes, forming a data grid, and finally the values represent certain measurements on each given node.

3 GENERIC FIELD CONCEPT

JSciC defines a generic data type as "field" and provides two main subtypes of the field - a regular field and an irregular field.

Nodes in a regular field are organized as one- two- or three-dimensional regular grid (or array), thus the structure of a such field is completely determined by its dimensions. The geometry of a regular field can be determined either by its origin p and unit cell vectors v_0, v_1, v_2 (a point with the indices i,j,k will be located in $p+i*v_0+j*v_1+k*v_2$) or by explicitly given coordinate array in the case of a regular curvilinear field. JSciC distinguishes these two types of geometry in a regular field and all visualization algorithms in VisNow exploit that optimization.

An irregular field, often called an unstructured field, is determined by a set of nodes and one or more cell sets. JSciC supports point, segment, triangle, quadrangle, tetrahedron, pyramid, prism and hexahedron as basic cell types. In addition to node data, cell data can be defined with different data on each cell set.

*p.wendykier@icm.edu.pl

Multivariate values can be defined over nodes or cells and are related to as "components". Multiple components can be defined over one field. Each component brings the numerical values of the dataset (in one of types determined by JLargeArrays library) with possibility of linear mapping to physical range with units support and together with basic statistics on the data. Each data component can be scalar or vector.
Additionally, field nodes can be masked as valid (visible) or invalid (invisible) with use of binary mask.
Time dependency is defined for geometry, mask and data components (structure dependency is under development). The key concept of time in JSciC is to treat time independently for each variable (component), mask and geometry, thus multiple different time steps can be defined for each of them. Additionally the library provides proper interpolation functionality for any time moment.

4 FUNCTIONALITY

JSciC library provides multiple embedded structure, geometry and data processing tools as generic utilities or fields' functionality. Field resampling, cropping, triangulation and slicing is included in the library itself, as well as interpolation and arbitrary probing. In more technical layer, hashcodes and fingerprints are provided as fast comparison methods.

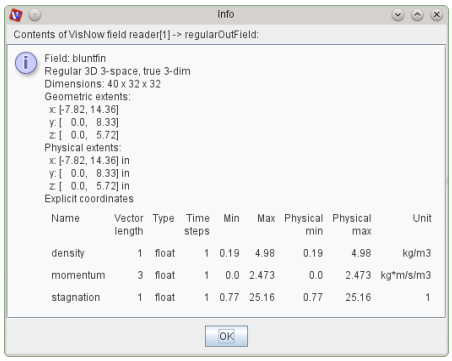

Figure 1: Example JSciC field description in VisNow of a 3D regular grid with three data components.

Data components provide basic or extended statistics, including histograms. Type conversion of data components is also possible with additional configurable support of NaN and Inf values.
Moreover, library API provides basic data arithmetic at the level of data components (equivalent of point by point or array operations), including support for both scalar and vector components. More importantly, as JSciC supports physical units, all arithmetical operations also incorporate proper unit calculations (e.g. mass in kg multiplied by acceleration in m/s2 results in force defined in Newtons). Most common physical constants with units are provided in the library for inside calculations. Execution of arithmetic is as fast as possible, not only utilizing automatic parallelization on multi-core and multi-CPU architectures, but also with fallback to fast JLargeArrays library arithmetic where possible. More advanced arithmetic at fields level (e.g. spatial gradient) is under migration.
I/O functionality is provided for JSciC fields - reading of all types of regular and selected irregular fields is possible via the ASCII header, while the writing format is predefined, similarly in a two-file manner - header and binary data.
As JSciC development follows the concept of open software, the source code is available for download from GitLab (https://gitlab.com/groups/ICM-VisLab/JSciC/).

5 APPLICATIONS AND VISUALIZATION CONTEXT

As JSciC library originated from VisNow environment [1], it was designed to serve the functionality required for the generic visualization system. Currently it provides the basic internal data structures for the VisNow modular visualization system. All functionalities of JSciC library find its representation in VisNow system at higher layer and with proper user interface. Detailed information on field and the components may be presented in description window (see example in Figure 1.). Both regular and irregular fields, serving the structure and geometry, provide the visualization system with primitives and basic structures for visualization purposes (e.g. 3D cells, see Figure 2.) and represent the generic fields in regular and irregular grids (see examples in Figure 3.). Multivariate field data - multiple components - provide a choice of data for colormapping or other visual layer representation of a single field. For example, a multivariate 3D field can be represented with volume rendering technique, where three data components are mapped to RGB colors and one component is mapped as transparency. Data arithmetic at component level provided by JSciC is utilized as component calculator with expression parser within VisNow module, serving as data generator or calculator for derived components. VisNow Field data format (VNF) being a native file format for data fields is a direct usage of JSciC provided I/O.
JSciC can be useful in applications that require generic data types defined in a real vector space such as physical simulations and engineering tools. Provided I/O functionality may be utilized in simulation codes, where real time visualization can be obtained without conversion of data formats. Finally, JSciC can be used in all Java applications that process very large datasets common in microscopic, medical and cosmological sciences.

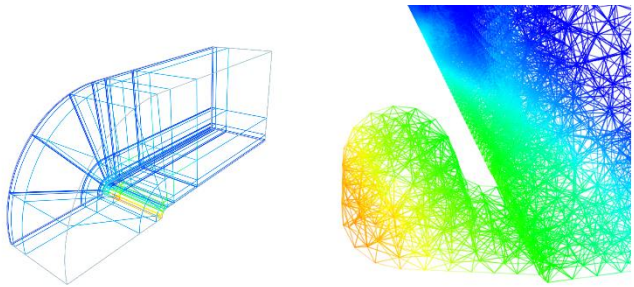

Figure 2: Examples of curvilinear regular field (left) and irregular field on tetrahedral cells (right) as presented in VisNow.

REFERENCES

[1] K.S. Nowiński, B.A. Borucki. VisNow - a Modular Extensible Visual Analysis Platform. *Proc. of 22nd Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision WSCG2014*, pages 73-76. 2014.

[2] P. Wendykier, B.A. Borucki, K.S. Nowinski. Large Java arrays and their applications. *High Performance Computing & Simulation (HPCS), 2015 International Conference on*, pages 460-467. July 2015.

[3] W. Schroeder, K. Martin, B. Lorensen. The Visualization Toolkit (4th ed.), Kitware, 2004, ISBN 978-1-930934-19-1.

[4] T. Pietzsch, S. Preibisch, P. Tomancak, S. Saalfeld. ImgLib2 - Generic Image Processing in Java. *Bioinformatics*, vol. 28, no. 22, pages 3009–3011. Nov. 2012.

[5] S. Vigna. Fastutil: Fast and Compact Type-specific Collections for Java. http://fastutil.dsi.unimi.it/. 2016.