# SuperGlue: Standardizing Glue Components for HPC Workflows

Jay Lofstead*, Alexis Champsaur†, Jai Dayal†, Matthew Wolf†, Greg Eisenhauer†
*Sandia National Laboratories †School of Computer Science, Georgia Institute of Technology

## I. Introduction

Existing workflow engines, such as Kepler [2] and DAGMan [3], offer flexible ways to assemble components with rich functionality to manage the control flow. What they both lack is a way to easily deploy and manage the glue code required to connect the various components. One example illustrating the complexities comes from the Oak Ridge Leadership Computing Facility (OLCF). Kepler was used for several workflows for the fusion simulation users. While the initial goal was that an internal, expert resource would create the workflow, including glue components, it should be able to be maintained easily by the application scientists. Unfortunately, the expert was required regularly as the configuration evolved and scaled. The complexities of making and maintaining the glue components as the output shifted and managing the deployment was too high. Falling back to Python scripts managed by the application scientist proved easier and faster to maintain. While this approach using the parallel file system to stage intermediate data was sufficient, it is quickly becoming infeasible. The IO overhead for using the parallel file system is exceeding acceptable runtime percentages forcing a reduction in output and making scientific insights more difficult to discover.

To address the performance mismatch, Integrated Application Workflows (IAWs) are being developed. The easiest way to think of these IAWs is the Unix/Linux shell pipe operator to connect commands. The shell connects stdout of one program to stdin of the next with the assumption that each component in the chain can operate in this mode. For tasks at this scale, this approach works well. For the scientific workflows we are targeting, we have processes spread across potentially 10,000s of nodes connected to other components also running on multiple nodes.

Several efforts to work through some of the issues related to IAWs have been investigated such as Catalyst, Libsim, Glean, FlexPath, Bredala, and PreDatA. One key observation, however, is that there is a lack of portability to the resulting implementations; they require a great deal of tuning and/or runtime placement control to make them function as desired.

This poster describes our work on SuperGlue, a set of generic, reusable components for composing scientific workflows. These are distributed data analysis and manipulation tools that can be chained together to form a variety of real-time workflows providing analytical results during the execution of the primary scientific code. Unlike existing components used in IAWs, SuperGlue components do not have a fixed data type. This one change enables using these components on completely different kinds of simulations that share nothing in their output format. Key to making this work is using a typed transport mechanism between different components. Many options exist for these transports and the particular mechanism selected is not critical.

## II. Workflows

We designed and implemented two realistic real-time workflows based on scientific codes having large user bases: the LAMMPS Newtonian particle simulator [4] and GTCP, a particle-in-cell Tokamak simulator [1]. While both of these workflows eventually turn the simulation data into histograms of certain quantities of interest, how they arrive at their final result varies significantly. Creating similar types of results, and this using some of the same components but in significantly different ways, has allowed us to gain important insight into how best to design glue components that can be used in a wide variety of workflows.

In typical scientific workflows today, custom glue code is written for selecting relevant data and writing it. Then, potentially additional custom glue code will fix the histogram calculation into something that can be rendered or saved as desired. In this work, we demonstrate general, reusable components capable of handling all three intermediate operations.

## III. Design

In this work, we offer some insight in the design of generic data manipulation and analysis components from our implementation of two workflows. These workflows are driven by two different scientific codes, yet they share some of the same components. We present our insights.

### A. Insights: Overview

By evaluating the presented workflows and considering other workflows with which the authors are familiar, four particular insights are revealed.

1) To allow for the greatest variety of workflows, data manipulation primitives and data analysis components should be packaged in similar ways – that is, regardless of their individual complexity, the pieces that make up these workflows should export compatible interfaces as much as possible.

2) The ability to handle multi-dimensional data, along with the consistent labeling of dimensions and quantities as metadata, allows for components that are highly adaptable and simple to use.

2168-9253/16 $31.00 © 2016 IEEE
DOI 10.1109/CLUSTER.2016.90

3) While different types of components understand varying levels of semantics, maintaining a high level of semantics (i.e., labeling quantities and dimensions as much as possible) early on and when passing through components that do not necessarily require all of these labels allows for the most functionality downstream.

4) Because programming languages understand multi-dimensional data as being in a specific order in memory, there is a need for glue components that re-arrange data and re-label its dimensions without necessarily changing its size. Indeed, when data is stored in a database on disk, it is simple to gain a desired view of the data, for example by using SQL. However, in the middle of a real-time workflow, data must be presented to the components in a format that they expect and understand. This requires a specific ordering of data in memory.

These insights guide the design for the reusable glue and analysis components presented in this paper. From a general perspective, designing a smaller number of components to assemble workflows with finer step decomposition allows for more general processing and more accurate performance expectations than designing more numerous components each having more complex functionality.

## IV. Reusable Components

This section provides greater details about the individual SuperGlue components and how a small number of parameters allow the them to operate (a) on a variety of different input data formats, and (b) in a user-specified way.

**Select** Given an input stream that includes an array with any number of dimensions, Select extracts certain indices from one of the dimensions. Thus, it outputs an array with the same number of dimensions, but with the dimension of interest having a smaller size. In order to select the quantities of interest, the component uses a header which must be passed by the previous component in the workflow. The header is a list of strings that name the quantities in the dimension of interest. This allows for easy selection of quantities at runtime when Select is launched.

**Dim-Reduce** Dim-Reduce is a glue component that removes one dimension from its input array, "absorbing" it into another dimension without modifying the total size of the data. The other dimensions are left unchanged. This component can work with an input array having any number of dimensions. The output is an array with one dimension removed and with another dimension that has been re-defined. When using this component, the user must specify which dimension to eliminate and which to grow.

**Magnitude** In our current implementation, Magnitude expects a two-dimensional array as input, where one dimension spans the data points at each time step (particles in the case of LAMMPS and grid points in the case of GTC) and the other dimension spans any number of components of the same vector, for example the three-dimensional components of velocity in the LAMMPS workflow. Magnitude calculates the magnitudes of the vectors from the values of their individual components and outputs a one-dimensional array of the new values. Which dimension is which in the input array is specified by the user at runtime. A small number of changes and a few start-up parameters could generalize this code to perform any number of common operations that calculate a quantity from many, applying a known formula over a two-dimensional dataset, thus allowing this component to fit into a variety of scientific workflows.

**Histogram** The processes that make up the Histogram component partition among themselves a one-dimensional array of data. They communicate to discover the global minimum and maximum values in the array, create a number of bins between these two extremes, and then communicate again to count the number of values in the globally partitioned array that fall in each bin. The number of bins to use must be passed to the component when it is launched.

**Dumper** While this component was not created in time for this paper, the value of a component specifically designed to be the endpoint of a workflow is clear. The key goal for this component is to offer a way to write an ADIOS stream into an output file using some particular format. Whether to write the workflow output as HDF5, ADIOS-BP, or a simple text file could simply be selected by the user as an option, requiring no modifications to existing components, and no re-compilation.

**Plotter** Another component that would be of value would be one with a graph plotting capability. For example, GNU Plot takes a simple text input description and generates a graph. Rather than having the graphing component write to disk, it should also push out a stream to some other consumer. An additional Dumper that writes an image file in a particular format would be a valuable addition.

Initial evaluation results are on the poster itself.

## Acknowledgments

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

This work was supported by Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357, program manager Lucy Nowell.

## References

[1] Z. Lin, T. S. Hahm, W. W. Lee, W. M. Tang, and R. B. White. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, 281(5384):1835–1837, September 1998.

[2] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.

[3] G. Malewicz, I. Foster, A. Rosenberg, and M. Wilde. A tool for prioritizing DAGMan jobs and its evaluation. *High Performance Distributed Computing, 2006 15th IEEE International Symposium on*, pages 156–168, 0-0 2006.

[4] S. Plimpton, R. Pollock, and M. Stevens. Particle-mesh ewald and rrespa for parallel molecular dynamics simulations. In *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, PPSC 1997, March 14-17, 1997, Hyatt Regency Minneapolis on Nicollet Mall Hotel, Minneapolis, Minnesota, USA*. SIAM, 1997.