Journal of Applied Crystallography

ISSN 0021-8898

Received 0 XXXXXXX 0000
Accepted 0 XXXXXXX 0000
Online 0 XXXXXXX 0000

Dataflow: a Web-based Data Reduction Framework for Neutron Scattering

William Ratcliff,[a]* Brian Maranville,[a] Paul Kienzle,[a] Andrew Tracer,[a,b] Elakian Kanakaraj,[a] Brendan Rowan[a] and Alex Yee[a]

[a]National Institute of Standards and Technology, Gaithersburg, Maryland 20899 USA, and [b]Princeton University, Princeton, NJ 08544 USA. Correspondence e-mail: william.ratcliff@nist.gov

Neutron scattering is a technique which necessarily requires centralized facilities, with distributed access (a user facility). At such a facility, outside users collect data on one or more instruments, which is typically recorded in instrument-specific coordinates. Each user needs to do two things with this data: convert it into more universal (physical) coordinates, and correct for any instrumentation-specific artifacts in the data. This process is termed "data reduction". A common paradigm for data reduction is for the instrument operator to create a new program and interface for each instrument. The source data files are reduced at user facility, where there is direct access to both the programs and the data. The reduced output files are then carried home by the users. This project aims to simplify this part of the user experience. A visual-programming interface is created in the user's browser, allowing reduction of the source data by remote control. Reduction protocols are built up using facility-supplied functions (in icon form) wired together to create a data flow diagram. The user (or instrument operator) can change the reduction protocol, and repeat the procedure as needed, and the resultant reduced files can be downloaded by the users at will. Standard reduction protocols are made available by the facility and instrument operator. Changes to the protocol (and the addition of new instruments and new protocols) will not require changes to the user interface, and instrument-specific functions for reduction will be maintained by the instrument operators, rather than a full-blown unique reduction program for each instrument or type of instrument.

1. Introduction

Every dataset that is produced at a neutron user facility must be reduced to physical coordinates, with artifacts removed and error bars added, before it can be analyzed or modeled. As a result, every instrument is typically equipped with a software routine for performing this reduction. In addition, if the reduction programs are to be distributed to the users so that they might work with the raw data at their home institutions after the experiment is complete, each program must be compiled for multiple target platforms (Microsoft Windows, Mac OS, Unix, Linux, etc.). At a large facility this requires the writing and maintenance of dozens of programs, with multiple target platforms for each program, just to perform reduction. Updates to software are also on a per-instrument basis, requiring notification of all the users who might have downloaded or copied the programs, requesting that they download a newer version.

For this reason, the Dataflow framework was designed to consist of two parts: the user interface which exists solely within the user's web browser, and a centralized server which is maintained by the facility staff. Updates to the server are implemented without needing interaction from the user, and updates to the user interface code (in the browser) require only that the user refresh their view of the reduction site.

2. Client

By presenting the user interface in a web browser, the installation requirement for users is shifted to the browser providers. All that is required of the user is that they run a modern browser. Current browser support is indicated in Table 1. The user interface is built in Javascript, mainly using the WireIt and ExtJS libraries. Plotting is accomplished through custom-built plugins for the jqPlot library (`http://www.jqplot.com/`), which is built on another common Javascript library, jQuery.

3. Server

The Dataflow system was designed with a backend server written in Python (Django, `http://www.djangoproject.com`), which makes the connection between the client (web browser) and the data reduction modules, which are written in Python. There is no particular requirement that the reduction code be written in Python, just that the code be callable from Python. Because of the large available body of numerical libraries in Python, it was not necessary to use any other languages in the project.

3.1. Instrument-specific modules

In the client and server, every reduction pipeline is categorized in terms of a single "instrument". For each instrument, all the reduction subroutines and data formats are defined. Typically this will consist of one or more "loaders" which import raw data from datafiles, and "filters" which operate on the data. It is necessary that the loaders and filters provide compatible objects within each instrument's namespace, but it is not required that the filters for one instrument operate on the data of another instrument. Thus every reduction problem is able to be addressed individually on the backend without requiring modification of the data or file types already in use.

All that is required is that existing algorithms for data reduction be ported to individual Python modules that act as filters, passing modified data to the next filter in the chain.

3.1.1. Triple-Axis Spectrometer Enter Alex Yee.

3.1.2. Small-Angle Neutron Scattering By Elakian. For instance, in the Small-Angle Neutron Scattering (SANS) reduction, many of the algorithms used were adapted from exisiting Igor Pro code (Kline, 2006).

3.1.3. Offspecular I imagine Brendan will contribute something here.

3.2. Caching

In many reduction processes, intermediate results can be costly to calculate. In order to have a responsive, flexible system, small changes to the dataflow that do not require recalculation of previous steps should not force this to happen.

Appendix A
Libraries used in Dataflow Application

A.1. Client

A.1.1. WireIt

A.1.2. jqPlot

A.1.3. extJS

A.2. Server

A.2.1. Django

A.2.2. PostgreSQL

A.2.3. Redis

A.2.4. Numpy

Acknowledgements

This work utilized facilities supported in part by the National Science Foundation under Agreement No. DMR-0944772. Andrew Tracer was supported as part of the Summer Undergraduate Research Fellowship (SURF); Elakian Kanakaraj, Brendan Rowan and Alex Yee were supported by thed Summer High-school Internship Program (SHIP), both of which are also part of the above grant.

The authors would like to thank Andrew Jackson, Steve Kline and Julie Borchers of the NIST Center for Neutron Research for their invaluable assistance on this work.

References

Kline, S. R. (2006). *Journal of Applied Crystallography*, **39**(6), 895–900.
URL: *http://dx.doi.org/10.1107/S0021889806035059*

Supported browsers, at time of publication

Browser	Min. Version	Link
Firefox	5.0	`http://www.firefox.com`
Chrome	13.0	`http://www.google.com/chrome`
Internet Explorer	None	Not supported