ICANS XIV
14th Meeting of the International Collaboration on Advanced Neutron Sources
June 14-19, 1998
Starved Rock Lodge, Utica, IL

Remote Data Collection and Analysis

Tom Worlton
Argonne National Laboratory

1. Introduction

Neutron sources are getting more intense and the quantities of data produced for a given experiment are increasing. Some experiments can be done in a few minutes. The number of experiments which can be performed and the number of papers which can be written is limited by speed of analysis. Instrument Scientists at the major facilities spend much of their time helping users, not only with data collection, but also with data analysis. Ideally, the outside users should be able to take a larger share of the responsibility for data collection and analysis. This would allow the Instrument Scientists to work with more users and more papers could be produced.

One of the factors limiting data collection and analysis by outside users is the difficulty in using the software that has been developed. The software generally is written in such a way that the user must type a lot of input. The software needs to become easier and more intuitive to use. The other factor limiting data collection and analysis by outside users is the need to travel to the facility to perform the experiment and analyze the data. Allowing this to be done remotely could greatly expand the number of users of the neutron facilities.

2. Remote Data Collection and Analysis Project

In order to enable scientists to more easily collect and analyze data and to allow them to do experiments remotely, we have begun a project to provide remote user-friendly data access, viewing, and manipulation. We also plan to provide an interface between this data viewing and manipulation software and specialized analysis programs. This project is possible because of recent developments in computer technology such as html and Java.

Steps in data collection and analysis are shown in Table 1. Most steps can be performed locally, but the steps in the shaded areas will still need to be performed locally.

Table 1 Remote data collection and analysis. Most steps of the data analysis procedures can be done remotely, but the shaded steps must still be performed locally.

Data Collection	Data Analysis
Load sample	Archive data
Enable user control	Read data
Set up data collection	View spectra
Start collection	Operate on Spectra
Control Sample Environment	Analyze data or prepare analysis input files
Stop Collection	Collaborate with Instrument Scientist

For remote data collection and analysis, samples will still need to be loaded locally, but a sample changer could allow batch loading of samples. Because more than one user might be using the instrument, it will be necessary for the facility to enable data collection for a specific user. Otherwise, it would be possible for one user to interfere with another or for an arbitrary person to interfere with data collection and instrument operation. Although environmental parameters could be controlled remotely, there would need to be hardware or software limits imposed by the facility.

Archiving of data should be handled by the facility and could be done automatically, by connecting the data transfer to the data collection process. Because CDROM drives are now found on nearly all computers, CD Recordable (CDR) is a very desirable storage and distribution medium. CDR disks will last much longer than tape and it will probably be many years before CDROM drives disappear. By that time, DVD drives will have replaced CDROM drives and they will also be able to read CDR disks. The other advantage of using CDR is that remote users cannot accidentally (or intentionally) overwrite the data once it has been archived. Data can be archived and accessed without operator intervention, if a CD jukebox is used.

We have chosen to write most software in the Java programming language because Java is designed as a network-based, operating system-independent language (see http://www.javasoft.com/). Java is also secure, object-oriented, and easier to write than C++ code. Java is also more suitable for use by non-professional programmers because it manages memory and does not have pointers. This prevents the programmer from creating memory leaks or accessing memory used by other programs. If the software is intended to run only on the server, it can be written in a native language such as C, C++, or Fortran.

Java programs can be compiled either as "applications" or as "applets". An applet runs inside a web browser, while an application can be run independent of a web browser. Java achieves machine and operating system independence by defining a "byte code" which can be run by a Java virtual machine installed on the local computer. In the case of applets, the virtual machine is provided by the browser. The use of an interpreted byte code means Java programs run slower than native language programs such as C and C++ that are compiled for the target machine. However, Java Just In Time compilers can be used to compile byte code into machine code and speed up execution.

One of the attractive features of Java is the ease with which one can write Graphical User Interfaces which can run on different platforms with a native look and feel. The Java GUI routines are in the Active Windowing Toolkit which appeared in Java version 1.1. Another class called SWING, provides even more powerful GUI functions. Swing will become standard in Java version 1.2. Sample menus showing Java menus and the File dialog box provided by Java are shown in Figs. 1 and 2.

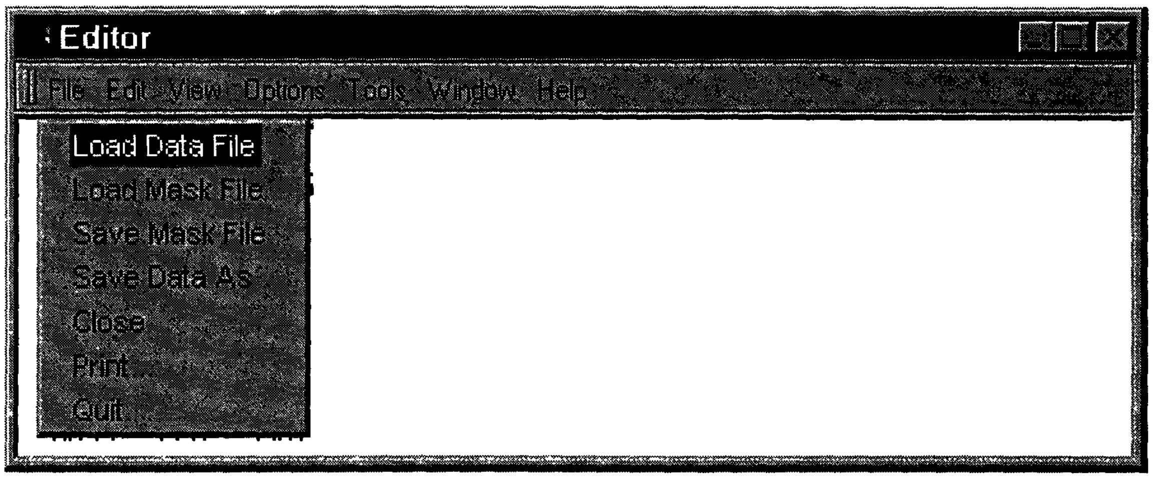
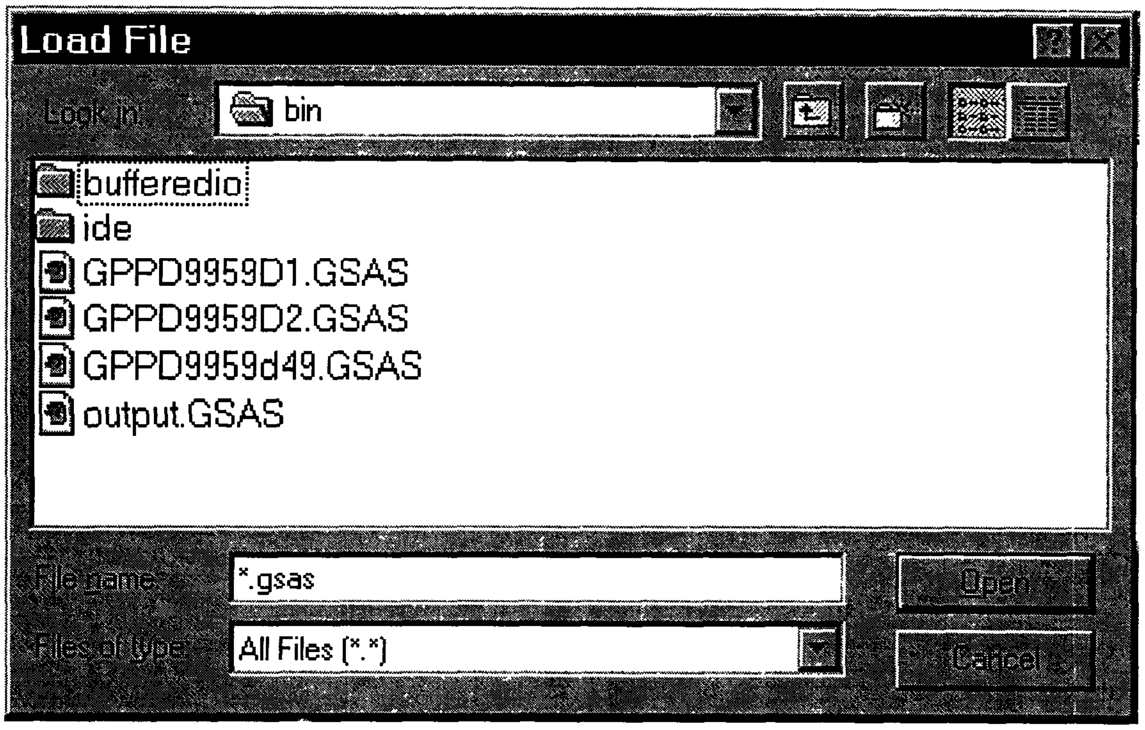


Figure 1 A sample Java menu bar and pull-down File menu.

A sample Java file dialog box is shown in Fig. 2. The built-in classes in the Java AWT and Swing provide a simple method for interactively selecting and reading files.



bin

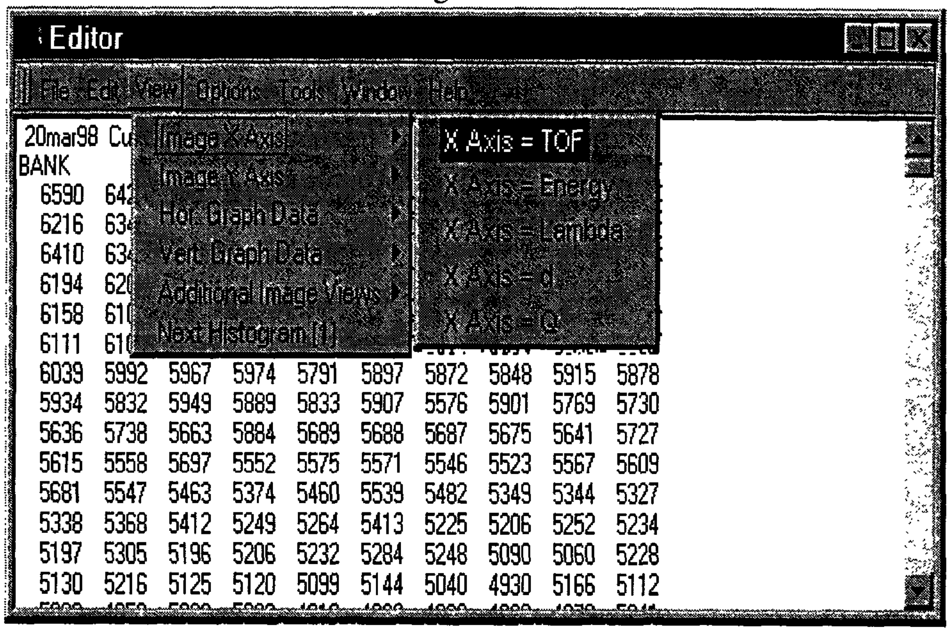
```
bufferedio
ide
GPPD9959D1.GSAS
GPPD9959D2.GSAS
GPPD9959d49.GSAS
output.GSAS
```

File name: *.gsas

Files of type: All Files (*.*)

Figure 2 A sample Java File dialog box for selecting and reading files.

Menu bars and pull-down menus are quite familiar to most computer users now because of the ubiquity of Microsoft Windows and the Macintosh as personal computers. One of the goals of new software design is to make the use intuitively obvious so that new or inexperienced users can find their way through the operations provided by the program without assistance from other users and without the use of a user manual. This can be accomplished through the use of a familiar menu structure such as shown in Figs. 1-3.



```
20mar98 Cu
BANK
  6590  642
  6216  634
  6410  634
  6194  620
  6158  610
  6111  610
  6039  5992  5967  5974  5791  5897  5872  5848  5915  5878
  5934  5832  5949  5889  5833  5907  5576  5901  5769  5730
  5636  5738  5663  5884  5689  5688  5687  5675  5641  5727
  5615  5558  5697  5552  5575  5571  5546  5523  5567  5609
  5681  5547  5463  5374  5460  5539  5482  5349  5344  5327
  5338  5368  5412  5249  5264  5413  5225  5206  5252  5234
  5197  5305  5196  5206  5232  5284  5248  5090  5060  5228
  5130  5216  5125  5120  5099  5144  5040  4930  5166  5112
```

Figure 3 Sample Java menus for selecting viewing options.

Although we are writing our code in Java, we may still execute some functions through the web browser and CGI for simplicity. The web browser can execute CGI scripts written in Perl, TCL, JavaScript, Visual Basic, or some other scripting language native to the machine being used as a web server (See http://www.w3.org/MarkUp/, http://hoohoo.ncsa.uiuc.edu/cgi/, http://www.perl.org/,. http://www.scriptics.com/scripting/)

3. Project Status

Initially we planned to put the data in a database so we could use standard Java DataBase Connectivity (JDBC) tools to access the data. We tested this with the free Postgress database and found it too slow and inefficient. We then switched to Microsoft SQL server and found quite good response and efficiency. This is an acceptable method for instruments with moderate amounts of information, but for instruments with very large data sets, we decided that it is better to read the spectra from the raw data files. The raw data files will be read from the disk drives on the data collection computers or from the CDR data archive. Storing the header information in the database is still preferred for convenience in searching for desired data sets.

The Java routines for searching the database and reading the data from the database are working. Programs to load the data into the database are also working. We used C for writing data to the database since the database is local, and populating the database does not require software to run on a client machine. Java was used for the routines to read from the database since they will typically be run remotely on the client machines.

For highest storage efficiency and speed of access, we have written routines to read the data directly from the raw IPNS Run Files. Since access routines for these files were already available in C, the routines were written in C with a Java interface to the native methods. We may later rewrite these routines in pure Java to allow users to download the raw data and access it using our read routines. We will also write data retrieval routines for other types of data such as NeXUS [P. Klosowski, et al. 1997].

We have designed Data objects to contain a single spectrum and DataSet objects to hold multiple Data objects with common independent variables. The Data object includes an XScale, an array of y-values, and an array of errors. It can also contain Attributes such as temperature or pressure. Attributes are optional and can be any string or float parameter. The Attribute class has a comparison method that allows a program to sort without needing to read and compare the attributes. Methods available for Data objects include add, subtract, multiply, divide, rebin, set attribute, and get attribute. The error array in the Data objects contains the statistical uncertainty for each data value. Raw neutron scattering data has a statistical uncertainty equal to the square root of the total number of counts, so a method has been included to set the errors to those values. As the data is manipulated, the data errors will be combined to properly propagate errors.

Each IPNS run would map to a DataSet with the spectrum from each detector element mapping to a Data object within the DataSet. DataSet objects include methods to add and remove Data objects, to set and get labels, IDs, titles, and axis units, and to get data ranges. The DataSet object includes a log of operations that have been performed on the Data in the DataSet. We will also have a DataSetContainer that allows handling multiple runs such as a series of measurements at different temperatures.

Programs have been written to display an x-y plot of a selected Data from a DataSet, and to display an image composed of all the data from a DataSet. The intensities of points in the image are represented by the pixel color and each row of the image represents a Data. The image display we have produced is similar to that generated in the TOF_VIS program [Mikkelson and Worlton, 1997].

Currently the rows of the image are sorted by detector element number and the data all comes from one DataSet. We are working on generalizing the selection and sorting of the different Data objects that make up the image so they can be selected from different DataSets in a DataSetContainer. This will allow us to display an image composed of spectra sorted by temperature or other variable.

We initially were uncertain whether Java would work well with cursor interaction, but have been pleasantly surprised. We are able to generate x-y plots and display selected values in real time as the cursor is moved over the image.

4. Conclusion

An increase in the neutron scattering user community is necessary to make full use of neutron sources. Software that is easier to use and that can be run remotely will facilitate involving new users with neutron scattering. World Wide Web browsers and Java make it possible to design software that can run on multiple computing platforms and can run over the Internet. IPNS has begun a project to design software to allow remote data collection and analysis. A database is being used for storing and organizing information about experiments. Java code has been written to search and retrieve information about experiments and Java classes have been written for reading IPNS data. Methods are included to view, manipulate, and save data sets. Work on the project is ongoing.

5. Acknowledgements

The funding for this project came through a grant from the office of the Director of Physical Research at Argonne National Laboratory as result of a proposal submitted by C.-K. Loong, J.P. Hammonds, R. Osborn, T. Morgan, and T. Diesz. Joe Nipko, Dennis Mikkelson, and Dhar Desai have written software most of the software described here.

6. References

P. Klosowski, M. Koennecke, J.Z. Tischler, R. Osborn, NeXus: A common format for the exchange of neutron and synchrotron data, Physica B: Physics Of Condensed Matter (241-243)1-4 (1998) pp. 151-153

D. Mikkelson, T. Worlton, TOF-VIS, software for interactive exploration of time-of-flight data, Physica B: Physics Of Condensed Matter (241-243)1-4 (1998) pp. 142-144