Grid enabling legacy applications for scalability – Experiences of a production application on the UK NGS

Anjan Pakhira, Ronald Fowler, Lakshmi Sastry and Toby Perring

Rutherford Appleton Laboratory, Chilton, Didcot, UK.

Abstract

ISIS is the world's brightest pulsed neutron facility which supports research into condensed matter, bio-molecular sciences and advanced materials. New instruments are coming on line which will be able to record tens of Gigabytes of data within hours and scientists wish to be able to process and analyse this rapidly to guide follow on experiments. Existing software needs to be adapted to scale up to these new data levels and to provide near real-time response to the user. This paper describes the reimplementation of one important ISIS data analysis application to run on grids such as the UK NGS and the experiences of the developers and end users.

1. Introduction

The ISIS facility at the Rutherford Appleton Laboratory is the world's most powerful pulsed spallation neutron source. It provides beams of neutrons and muons that enable scientists to probe the structure and dynamics of condensed matter on scales ranging from the sub-atomic to the macromolecular. The scientific domain covers soft condensed matter, biomolecular sciences and advanced materials science.

Current work at ISIS involves the installation of seven new instruments in the second target station which is under construction. A key feature of many new machines is faster data gathering at higher resolution, which means that there is more data to analyse.

Figure 1 shows the new Merlin instrument which will produce of the order of 20 GBytes of data for each experimental run. Analysis of the data as it is obtained would allow the scientist to decide what, if any, further measurements should be made in the experimental run.

One of the remits of the CCLRC e-Science department is to help grid enable computational and data intensive applications such as these. Hence a close collaboration was established between ISIS instrument scientists and members of e-Science to develop a tool that would provide easy access to grid resources as well as giving a greatly enhanced user interface for the software.

Initial results of this work were presented previously [1]. In this paper we give a more detailed review of the final version of the software, along with experiences of both developers and end users of the grid enabled application.

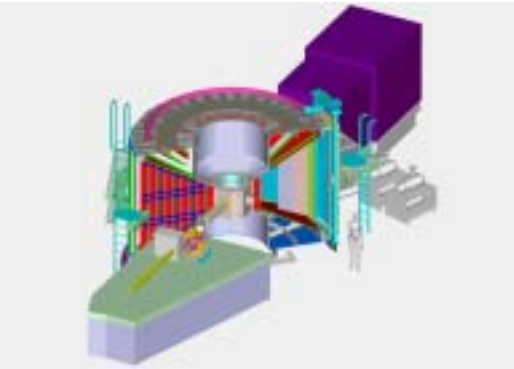

Figure 1: Diagram of the Merlin detector

2. The Application

The data analysis used for a class of ISIS instruments such as MAPS and Merlin [2] is based on the three packages, shown in Figure 2. Homer does initial data reduction, Mslice[6] is used for visualisation and selection of 1D and 2D subsets of the data. TobyFit [3] is the most compute intensive part of the data analysis software and is a specialised non-linear least squares fitting package including Monte Carlo simulation of instrument effects. It is a Fortran command line application of about 32,000 lines, with more than 100 subroutines.

The scientist provides a parameterised form of the expected scattering and background functions and, by selecting some or all of the

data sets available, tries to find the best fit. Currently only a fraction of the total data is actually used in the fit due to the time taken on a single PC, which could run into many hours or even days. As the data rate increases substantially with new instruments this becomes a major problem.

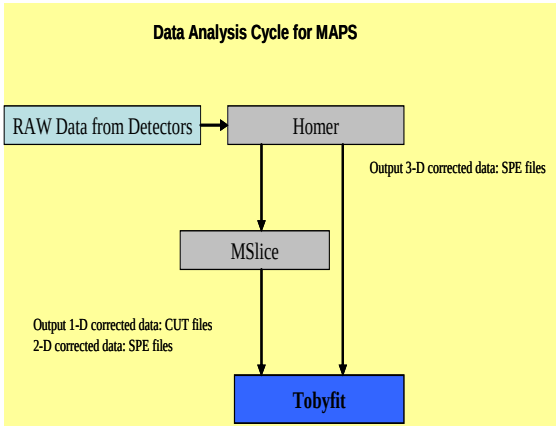

Figure 2: MAPS data analysis process. Homer and MSlice are used to pre-process and select data to fit or simulate with TobyFit.

2.1 MATLAB and MSlice

Many scientists within ISIS make use of MATLAB or IDL as a convenient problem solving environment for their data analysis needs. Mslice is a tool that has been developed in MATLAB, which is used interactively to explore the data and select which parts of it to fit. A simple GUI allows easy data selection and plotting and enables experienced users to employ their own MATLAB functions on the data. Hence there is a desire that any grid enabled interface to the TobyFit application should be made more user friendly with a MATLAB GUI to it. This of course is not a requirement of "grid enabling" software, and does require a significant amout of effort, but is seen as an important aspect of making the grid based version more accessable to the end user. Without the benefit of the GUI, the new version would be much less attractive to new users.

2.2 Data storage

At present most data is simply stored on the user's local machine. ISIS is moving towards use of the Storage Resource Broker (SRB) for their data, along with a metadata model to fully describe all data. In future it will be the case that the primary repository for experimental data will be on an SRB system with high speed links to computational resources such as the UK NGS. The CCLRC Data Portal and its meta-data model will be used to track results.

The use of a central data store has advantages of been easier to back up and to share with other users. With rapidly increasing amounts of data to store the limitations of transfer time to the local machine may become a bottleneck.

3. The Solution

The approach taken to solving this problem is designed to allow the scientist to replace the existing command line interface with a powerful Graphical User Interface (GUI) built inside of the preferred MATLAB environment. The GUI allows setting and manipulation of the numerous run time parameters required in the fitting process. When these are as required, a job can be submitted to an appropriate grid resource. As the computational task will be a significant one, it is vital to provide a parallel implementation of the fitting process that can be run on any available grid resource. Since large amounts of data are likely to be utilized in many fitting runs, the Storage Resource Broker will be used to manage the data where possible.

A diagram of the architecture used is given in Figure 3. The various components are discussed below.

3.1 MATLAB GUI

The MATLAB environment allows easy manipulation of vector and array objects through a high level language. There is also support for GUI development using Guide or Java Swing classes. The Java option was used as it is better suited to the complex interface that was needed by the application.

Typical views of the GUI are shown in Figures 4 and 5. As can be seen there are several separate tabs to control the various aspects of the data fitting process. One or more data sets from a given experiment can be selected, and each of these may be cuts (1D), slices (2D) or an entire 3D data set. In addition one can select to fit to the data from several separate runs. This means there can be very many parameters to fit as it possible to have different background models for each run, in addition to the many values that can be used to describe the foreground scattering.

The data structures to store and process the options defined in the GUI have been converted from Fortran into standard MATLAB commands. This enables all the existing options to be accessed through the GUI.

The other requirements of the interface were:

- To provide visualisation of the input data, fitted results and errors. This was done using the MSlice[6] package and direct MATLAB functions (Figure 4).
- To allow job and data submission to any grid resource.
- The ability to monitor and return output from these jobs.

These last two points are discussed in the following sections.

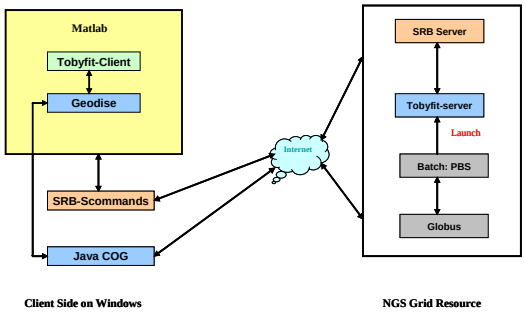

Figure 3: Diagram of the interaction of the MATLAB client with NGS compute nodes and the SRB.

3.2 Job Submission and Monitoring

The MATLAB GUI needs to be able to submit jobs to current grid resources which are built on Globus Toolkit 2, from both Linux and Windows hosts. The Java Cog toolkit provides such functionality, but a more convenient interface to it was found to be the Geodise Computational Toolbox [2] which can be called easily from MATLAB. This allows submission of jobs to any of the NGS clusters [4] or the CCLRC dedicated system Scarf. These are batch systems so the GUI provides a query option to monitor the jobs' progress in the queue and return the results when complete.

The user has the option to either specify the resource to use and the number of processors required, or to let this be automatically selected. At the moment the automatic option is fairly crude, but in future it will make use of a common queue for all NGS machines, when this becomes available.

3.3 SRB and data management

Since very large data files often need to be fitted, it makes sense to exploit the SRB [3] service that is part of the NGS to store these. The user has the option to up load these from the client to the SRB, if needed. Subsequent sessions then make use of these for any fitting jobs and return results to the to the SRB, which the client can access for visualisation.

3.4 Parallel implementation of fitting

The non-linear least squares fitting process had to be implemented in parallel to exploit the full power of the distributed memory clusters. To retain compatibility with the existing software, and avoid re-implementing vast amounts of code, MPI calls were added to time critical loops so that the objective function is evaluated in parallel, with much of the remaining logic still being serial. As much as possible of the existing Fortran was retained, so the scientist could still work with known code, while achieving reasonable scaling for tens of processors. Moving to 64 bit architectures, such as Scarf, should deal with increasing memory requirements.

Figure 4: One of the main panels of the MATLAB GUI, with the advanced panel. Many different parameters can be selected for fitting.

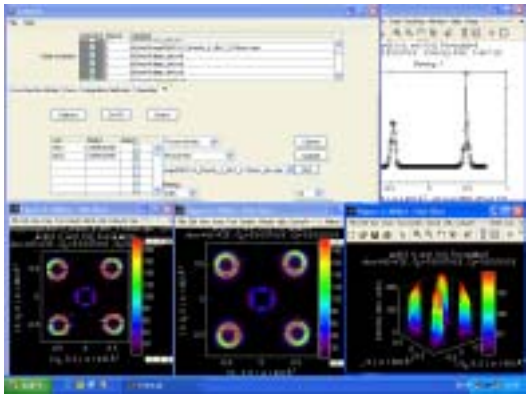

Figure 5: Display on the client showing the job control window with a list of submitted jobs. 2D cuts from the data and the fitted simulation are compared.

4. Experiences from Implementation

The grid enabled version of Tobyfit had three main development stages:

- Implementation of a parallel version of the fitting code suitable for use on clusters such as the NGS.
- Implementation of the client GUI within Matlab.
- Implementation of a data management and job submission scheme, between client and server.

Each of these presented challenges and required collaboration with end users to develop a useful system.

4.1 Experiences from development of Parallel Tobyfit

4.1.1 Language and parallel methodology

Fortran 95 was selected as the development language for the use in the parallel code. This is reasonable as all the existing software can be used without change and NGS provides the PGC and Intel compilers.

As already mentioned, a parallel implementation based on MPI was made. This is a widely supported standard library which will be available on virtually all parallel architectures. While OpenMP may have offered a simpler parallelisation, it is only applicable to shared memory nodes and requires extra compiler support.

The downside is that significant code changes are required with MPI compared to OpenMP and it is important to get the software maintainer – such as the ISIS instrument scientist - to accept these as an important part of the software. Otherwise the parallel code will not be kept up to date.

4.1.2 Analysis and profiling of code

The first challenge in analysis of the serial version of the code was to identify the behaviour for a set of benchmark cases and to understand the control flow of the program. This required getting a set of test problems that accurately represent both current uses of the code and future requirements in terms of size.

While is easy to get small and medium sized demonstration data sets, it takes more work and time to obtain test cases that represent future requirements. This is because such cases are simply too expensive to run with the current software on one processor. The application scientist had to take time to generate new files for this.

4.1.3 Level of parallel implementation

There are a number of different levels at which parallelism can be exploited in the fitting process. This choice obviously influences both efficiency and scalability of the code. Since many of the test examples showed cases where a one dimensional cut was fitted, which maps to a single spectra internally, we chose to parallelise the code at the level of individual spectra.

With hindsight this is not always the best choice. The reason for this is that fitting to three dimensional data, there are a very large number of individual spectra (order of 30000), but each has relatively few points in it. This presents limitations to the performance (see 4.1.4).

With this chosen level of parallelism it was possible to limit changes to a few tens of routines, so that the bulk of the code is identical to the serial case. Hence future updates to the serial version should easily carry across.

4.1.4 Parallel performance

Figure 6 shows the observed speed up for the parallel code on one of the NGS clusters. These results are for up to 16 processors, which is the level of resource which should be readily available on such clusters. Larger requests often have to wait in the batch queue, which defeats the intended interactive nature of this tool.

While the scaling is not as close to linear as one might hope, the performance is a major

improvement over the performance seen running the serial code on a local PC.

The scaling should improve when the amount of data in a slice is increased, since this will improve the load balance and ratio of computation to communication. Similar results are obtained when fitting to 1D cuts where the number of data points in each spectra is also high.

When the accuracy of the Monte-Carlo integration is increased, the ratio of computation to communication will also increase. This will give improved scaling when more accurate results are needed.

The scaling of the parallel code is not so good when looking at fitting 3D data sets for full SPE files. The reason for this is that the number of data points in each spectra is still rather limited, even for large data sets. The scaling in this case would be much better if it was made parallel at a higher level. This will be investigated in the future.

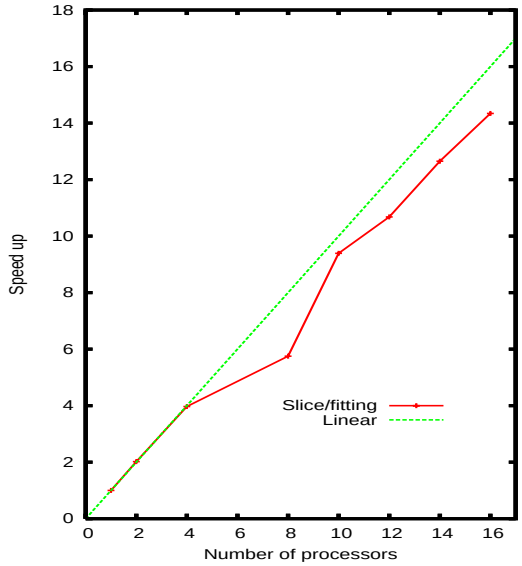

Figure 6: Measured speed up for the parallel code for fitting to a 2D slice, using up to 16 nodes on the NGS.

4.2 Development experience of the parallel implementation

The challenges in optimising and implementing a parallel version of the compute intensive fitting code included:

- identification of compute intensive subroutines
- mapping the data files needed
- understanding the data structures
- following the interaction between least squares code and the data simulation.

One requirements set for parallel version was maintenance of existing structures and format of input output data. Adopting existing structures and file formats made development quicker. However this meant a significant loss in flexibility of memory management.

The compute intensive fitting component was developed into a non interactive program, which could accept multiple inputs as command line arguments. One feature of the old version and the new parallel version is the use of environment variables to define input file paths. This feature was easily ported to the serial version. However MPI does not have a standard way to handle user environment variables. MPIRUN and MPIEXEC interfaces to launch a parallel job do not have a standard interface to pass user defined environment variables. This indicates that depending on non-standard features of a language, such as environment variables, is never a good idea. When launching jobs via the GT2, the necessary environment variables can be set from the RSL.

Performance was the major reason behind development of the parallel version. Performance improvement in speed of computation was one factor, while the other factor was memory management to enable exploration of larger datasets.

The ability to explore larger datasets than hitherto possible has been partially achieved. The maintenance of old data structures meant static memory allocation, which eventually hit the i386 limit of 2GB per node. With this amount of space software handles 5 times as much data as the original serial version, which is sufficient for the present. However, future detectors amy require more than this. This should be possible by changing the current implementation to be based on spectra for 3D cases, and only map the required specrtra to each worker node.

4.3 Development experience of MATLAB client

The MATLAB GUI for Tobyfit is complex because it reflects the command line functionalities of it sequential version. The serial version has a large number of parameters and options which the user may want to alter. Often the extra options are only used occasionally by the advanced user, but must be included in some way for a complete interface.

The Java Swing GUI of the scale developed for Tobyfit is unusually complex and pushes the software quite hard. Although Matlab makes available some functionality to use Java classes, it does not support all the features of Swing, such as event handling. Matlab's single thread of execution structure also prevented use of sophisticated event handling mechanisms.

4.3.1 Geodise Compute Toolbox

The Geodise toolkit gives a Matlab interface to Globus. Geodise also exploits feature of Matlab – Java interface. It implements a subset of features for job submission, query and result retrieval. The experience of using Geodise has been largely positive. The implementation of job polling, and result retrieval needs a multithreaded implementation, which will enhance user experience.

4.3.2 Job submission

The job submission, and result retrieval has complex bookkeeping to manage multiple simultaneous job submission to multiple NGS resources. The use of gridFTP to transfer data had long timeouts which are thought to be due to firewall issues. These made it very difficult to use this protocol for jobs requiring interactive response. Hence SRB was investigated as a more reliable data transfer mechanism.

It was found that dynamic resource query and discovery did not function robustly. The application therefore depends on the user choosing a resource from a list.

4.3.3 SRB

Data management for both input and results was implemented using SRB. The SRB features of remote synchronisation improved file transfer performance substantially. Backward compatibility of SRB is an issue. SRB has a Java interface, which was initially to be used within Matlab, however we found it difficult to use. As a result a subset of SRB functionalities required by the application was developed by wrapping S-commands within Matlab. The use of SRB solved bulk transfer related issues, but some sessions display arbitrary behaviour leading to partial file transfers, or long delays.

4.3.4 Grid requirements for interactive jobs

Applications of the nature of Tobyfit require short bursts of compute power, and have quick return times. The queue management of grid resources have been a major performance issue. All jobs sharing the same queue depending on the number of processors requested is not optimal. Quasi interactive grid applications of the class of Tobyfit require implementation of intelligent queuing solutions.

Long queue related delays, was a motivating factor behind a history logging scheme developed with in Matlab. This effectively gave the user the ability to launch a number of grid jobs and close a session. The results for the grid jobs could then be retrieved during a fresh client session.

Any grid application of this nature ultimately dependents on the fabric layer for performance from the usability point of view. Network configurations and firewall policies of different NGS sites have been variable. We found site specific policies often were a point of failure.

4.3.4 MATLAB client performance issues

Having described specific features of the client development, it is necessary to evaluate how the client side software performed as a unit with in Matlab. The client side code consists of over 20,000 lines of Matlab scripts, excluding the toolkits. Our experience with Matlab and recently with IDL show that large scale development with in a scripting environment leads to memory management issues. We found Matlab required large amounts of memory to function, and quite often software response became sluggish. The garbage collection mechanisms did not always work as expected, and some memory was not de-allocated even after end of session. This was experienced more often when external java classed were used.

The data management features implemented using SRB, improved reliability of bulk data transfer, but backward compatibility, and server failures were difficult to tackle.

The user is always focal point in any development of this nature. The software development apart from complexities of the technologies, had to iteratively allow for user requirement variations on the client side, and base software changes on the server side. This introduced delays. There is a need for users to specify requirements with a sufficiently long term view and be flexible to basic structural changes in the grid enabled version so that application performance requirements can be met.

5. Limitations of the middleware

The current version of the application makes use of various grid middleware components as discussed above. These have been made to work together to give a usable production tool for data analysis on ISIS. There are of course some areas in which these tools could be improved.

5.1 Geodise Computational Toolbox

The version of the toolkit we used (v0.7) had a few features that could be improved in future releases. For example, gd_createproxy clears the Matlab space and requires the user to type return into the command window. Also there were some issues with file transfers with respect to fire walls and transferring whole directories. Overall the software was very useful and gave easy access to Globus resources. It would be useful if the toolbox could be extended to support direct access to MDS information. The inclusion of an SRB interface would also be useful.

5.2 GT2.4

The clusters used in this work were all running the VDT distribution of GT2.4. This provided a stable interface for job submission which supports MPI based parallel jobs. It also hides the differences between job managers such as PBS and LSF. There were some problems with the environment that the user can expect for a job. For example dynamic libraries were not always found. Care also had to be taken to ensure that jobs could run on systems with different paths to the user's file store.

While it is possible to query resources to see how busy they are, it would be useful if there was a single jobmanager for all NGS resources which would run a job on any cluster that had spare capacity. Such a queuing system may become available in the future.

5.3 SRB

The version of SRB we used (v3.2) had a few problems. These included occasional slow response times, difficulty in finding a GSI based version of the S-commands for Windows and bugs in the Windows version of the Srsync command. Hopefully these should be fixed in later versions.

6. End user experience

The grid enabled version of the software has been delivered to the instrument scientist who has installed and used it to fit new and larger data sets. The software will be released to more users in the near future.

Obtaining a UK e-Science certificate and installing it in the correct location for the client is seen as more complex then it might be. Also the need to set up SRB access using Encrypt1 password, rather than GSI authentication, was a draw back.

Problems with some NGS resources becoming overloaded with work was identified. This should be addressed at least in part by a single queue for all NGS resources, or use of a resource broker.

The GUI was seen as a very important step in making the fitting process easier to understand and simplifying grid access. While the built in visualization is sufficient, there is a desire to include more advanced techniques.

7. Conclusions and future work

A substantial amount of effort has been put into providing a grid enabled data analysis tool which is being used by scientists in ISIS. This includes development of a sophisticated MATLAB GUI which, while not directly grid related, was seen as necessary to hide some of the complexity of grid operations from the end user.

Feedback from initial users has been very positive and the tool enables the user to fit significantly larger data sets then before and visualize the results locally. Future areas for refinement of the software are in improvements to scaling for performance and memory use for 3D problems. There is also a need to provide an intelligent resource broker service that will automatically select the best computational resource to use.

Visualization of very large data sets using grid resources is another area of research that we are investigating. The framework for this and the data analysis should be applicable to other instruments at ISIS and related facilities.

8. References

[1] *EVE – A grid enabled data analysis environment for neutron scattering experiments*, T.G.Perring, R.F.Fowler, A.Pakhira, M.Sastry, UK e-Science AHM2004, Nottingham.

[2] http://www.isis.rl.ac.uk/excitations/maps/maps.htm, and http://www.isis.rl.ac.uk/isis2002/developments/developments.htm
[3] *Tobyfit Version 2.0: Least squares fitting to single crystal data on HET, MARI and MAPS*, T.G.Perring, 2000. Available at http://www.isis.rl.ac.uk/excitations/documents/tobyfit.pdf.
[4] UK National Grid Service: http://www.ngs.ac.uk
[5] The Geodise Computational toolbox: http://www.geodise.org/toolboxes/generic/toolkit_matlabcond.htm
[6] *Mslice: A Data analysis program for time of flight neutron spectrometers*. R.Coldea, http://www.isis.rl.ac.uk/excitations/mslice/index.htm