# The ORNL 5 year plan for Mantid

Prepared by P.F. Peterson, S.E. Hahn, A.T. Savici, R.M.F. Leal, R.E. Whitfield, and G.E. Granroth

December 7, 2015



The overarching Goal for Mantid, as applied to neutron scattering at ORNL, is a core set of routines that can easily be integrated with other systems. Examples of this include, but are not restricted to:

- Inclusion in a workflow where models or simulations are calculated then compared to measured data. This feedback is used to modify and iteratively improve parameters in the model.
- Providing feedback to data acquisition to decide that a particular measurement is done. One example would be reducing live data to S(Q) and fitting individual peak parameters.
- Instrument scientists can try out new ideas using the vast array of packages available for python, like IPython, NumPy, SciPy, pandas, SymPy, etc. In this case Mantid, as accessed by the python interface, becomes a neutron specific package along side the others.
- Scientists are interested in a SciPy [1] style library for neutron scattering. This library would be easily incorporated into other projects (small dependency list) and would provide a gateway for external contribution.

This is not to suggest that the features of the Mantid project beyond the framework and python extensions, i.e. MantidPlot, are not useful or not needed. Instead, ORNL is directing the project to enable Mantid to be better integrated within other systems.

The necessary tasks required to reach this goal are summarized in sections on the Following document. As a summary the section titles are listed here.

- Separating user interfaces from core functionality
- Mantid as a SciPy styled python package
- User centric documentation
- Extensible and flexible instrument geometry and units
- The workspace types need to more efficiently reflect the science
- Experiment steering
- Advanced visualization
- Mantid connecting to advanced analysis and modeling
- Utilization of distributed and heterogeneous computing environments

# 1 Separating user interfaces from core functionality

## 1.1 Current situation

This is arguably the number one priority for ORNL concerning Mantid. The original design document for Mantid, contains a figure at the beginning of section 3.1 in [2] that describes a modular software that has a core written in C++, a well defined API for language bindings and graphical user interfaces (GUIs). This original goal has changed over the life of the project, as more necessary behavior is written directly in python. Also, the GUIs have become more tightly coupled to the framework than was originally designed.

## 1.2 Objectives

Mantid has been designed to be modular and this is an effort to re-emphasize that [2]. The refinement of the figure at the beginning of section 3.1 in [2] is that there will be more inside the Mantid Framework, and the division with things outside will be more distinct. Done properly, this would allow for more rapid development of standard alternate user interfaces as well as more technique specific user interfaces.

The main criteria for inclusion in Framework are things needed for data reduction on distributed computers (e.g. fermi.ornl.gov). This includes much of what is currently in the Framework package of the source code, including Python, and dropping scripting for other languages.

The two biggest reasons for this is to (1) promote encapsulation of calculations to simplify embedding Mantid into other systems without losing functionality and (2) to help simplify the code base in order to promote maintenance and adding capabilities.

There is already discussion of future user interfaces and how they would fulfill the needs of different users. Some examples include IPython Notebooks, ParaView web, custom web applications, Eclipse RCP applications, and custom GUIs that incorporate analysis software to help the user. The current user interfaces are only beginning to branch out into serving other types of users. The GUI workbench is useful for sci-

entists who want to explore functionality; python scripting helps scientists script and automate their workflow; technique specific custom GUIs that guide novice users. These ideas need to be expanded on and make the results of neutron scattering measurements more accessible to the different types of users with varying access to computation resources.

### 1.3 Milestones

1. Reorganize code to clarify the boundary between "framework" and "user interfaces" (4 months).

2. Rework rpm/deb installers to support sub-projects (2 months).

## 2 Mantid as a SciPy styled python package

### 2.1 Current situation

There have been several instances where ORNL staff wanted to use pieces of Mantid functionality in situations when a full installation is not necessary. The first example python script used Mantid for performing coordinate transformations and matplotlib for visualization during data acquisition. The second example used Mantid to augment NumPy and SciPy for exploring concepts in data reduction before being passed on to the Mantid team for incorporating in the software properly. In both of these cases, progress was difficult for a variety of reasons including the number of dependencies, overhead of initializing the framework, and accessing/modifying in memory data.

### 2.2 Objectives

A solution to all of these problems is to generate a significantly smaller product that, as a simple example, contains the ability to convert units of a supplied vector of values, with a supplied total flight path and scattering angles. This would have no workspace objects, geometry system, or logging framework. The library would use prefer using

objects native to python or NumPy to promote interoperability with other packages. To make this available, much of the Mantid code base would have to undergo major rework. Much of what is desired is currently inside of Algorithms. The kernel of these would be moved into non-member C-functions that could be exported to python, and included in the existing Algorithms. The python bindings would follow principles similar to SciPy. A side effect of the new core is cleaner, simpler code that is easier to test and maintain.

### 2.3 Milestones

1. Determine example application for the library (1 month).
2. Explore technologies to be used for the application and determine what Mantid would need to provide (2 months).
3. Build example application (6 months).
4. Generalize decisions to make library available outside of example (12 months).

## 3 User centric documentation

### 3.1 Current situation

Recently the Mantid project made large improvements for documentation of the algorithms and a start at connecting them to the objects exposed in python. However, this is a work in progress.

The online documentation is generated automatically by one of the build servers. We already have a very good experience with documenting algorithms, where we have documentation tests to check if the relevant parameters are changing. If some optional packages are missing, some of the algorithms are not built. Their documentation is therefore missing from the online version. This is the case of the MPI related algorithms, and for python algorithms that require non standard packages (e.g. DSFInterp). In the case of python algorithms, since they are shipped with Mantid, it is possible to have a running algorithm with no documentation.

We have documentation about setting up a system for a new developer, but it is sometimes out of date.

Mantid has an overwhelmingly large amount of functionality, and this is an appropriate place to limit what users see in order to help the relevant information become more visible. Users for SANS might not be interested in algorithms specific for inelastic scattering. At the same time, for a new user is almost impossible to read documentation about 600 algorithms, several user interfaces, and tens of concepts.

## 3.2 Objectives

We need to generate documentation for all algorithms that are (or can be) available to users, and to explicitly state which dependencies are not satisfied for a particular algorithm.

There is a need to expand the linking to relevant data objects (e.g. specific workspace types, lattices, run objects), and general improvements to the documentation of things that are not algorithms. This documentation has to be available offline, and has to be automatically tested.

We would like that all documentation for users and developers to be available offline, with the online version generated and tested by the build servers. Documentation about setting up a system for a new developer is sometimes out of date. Keeping all development documents in one place, maybe with a required workflow testing before each release (either manual or automatic), will ensure that anyone can be up and running in a very short time.

The Mantid tutorials should also be automatically generated, to include up to date screenshots, and be automatically tested to see if the syntax/ parameters have been changed. At the moment, it seems like the stumbling point is the ability to generate nice pdf printouts. Similarly, we should automatically generate screen casts.

The future documentation ORNL would like to have is customized towards specific techniques and, as appropriate, specific instruments. The future emphasis of the documentation should be on aggregating the relevant knowledge to perform a particular type of experiment. The starting point should be the physics of that particular measurement. The documentation should include a description of the input data, the

output file format, the relevant physics that transform time-of-flight (or reactor) data to the desired quantity, the algorithms and workflow used in Mantid, and links to additional information, such as detector diagnostic, sample alignment, event filtering.

For many instruments, both at SNS and ISIS, we started to implement autoreduction scripts, and these should be documented according to the principles exposed above.

At this point, the developers have enough knowledge to diagnose a series of problems with the instrument and data acquisition. It would be useful to develop a comprehensive plan to share this information with instrument staff. This should include notes about plotting relevant areas of the detector space, diagnose wrong timing, calibration of the instrument, and so on.

## 3.3 Milestones

1. Automatically generate documentation for all algorithms shipped with Mantid, irrespective if the algorithm can run on a build server (12 months).

2. Expand offline documentation of non-algorithm objects, including documentation tests (ongoing).

3. Automatically/manually tested offline documentation for system setup for new developers (12 months).

4. Automatically generated screen shots and tests for Mantid tutorials (12 months).

5. Gathering technique specific documentation requirements (12 months).

6. Generating technique specific documentation (12 months after milestone 5).

7. Generate database and documentation for diagnosing problems with instruments (12 months - ongoing).

# 4 Extensible and flexible instrument geometry and units

As ORNL expands its usage of Mantid in support the instruments at HFIR, the topics of instrument geometries and units become increasingly important.

## 4.1 Geometry

### 4.1.1 Current situation

Instrument geometry is defined as a tree structure. The tree nodes represent physical components (subclasses of Component). In the case of a detector, the tree structure consists typically of panels and tubes which ultimately break into pixels leaf nodes.

When a component has parameters (e.g., position in space), the component is said to be parametrized. Its location in memory (memory address) indexes a map (ParameterMap) with the respective parameters.

As pointed out by an internal ESS report [3], this implementation is highly inefficient. Algorithms that need to access detector/instrument position can spend between 5% and more than 30% of time in calls to the functions getPos() and getDetector() rather than in functionality of the algorithm itself. As a stopgap solution, some algorithms have overcome this inefficiency by creating local caches of instrument geometry.

In addition the geometry of an instrument is tightly coupled to its data. The duplication of a workspace implies the duplication of the instrument structure, even if it was not changed. Moreover, this coupling particularly hinders the possibility of dealing with data collected at different geometries, i.e., handling instrument components that move during a single scan.

Scanning instruments, such as some powder diffractometers at reactor based sources, have datasets formed by data collected at different positions. To accomplish wider $2\theta$ ranges, the detector is usually rotated around the sample and several diffraction images are taken. Thus a single dataset is usually formed by joining data acquired at different detector positions. Initial Mantid developments were targeted to TOF instruments. This initial choice has made workspace very rigid. It implies an *one-to-*

*one* relationship between a dataset and a position of the detector.

The Instrument Tree Structure of Mantid was only thought with three top components: *source*, *sample* and *detector*. Indirect geometry instruments, along with Triple Axis Spectrometers (TAS) and a new polarizer configuration for HYSPEC, have an additional component sitting in between the sample and the detector.

### 4.1.2 Objectives

**Scanning instruments** The instrument geometry needs a significant redesign. For fixed wavelength instruments, the main issue is the inability to handle instrument components that move during a single scan (e.g. area detector on a moving arm).

Workspaces formed by different detector positions will require changes to the core API of Mantid.

**Inverse geometry instruments** Inverse geometry instruments, such as BASIS and VISION, have a separate engineering and neutronic geometry. This constrains the connectedness and hierarchy of components to match between the two representations (e.g. detectors must be cylinders in both representations).

**Extra components** It is important to rethink the 3 component model (source, sample and detector) and extend it to cases where more components are needed.

The new paradigm of an additional component must take into consideration that deviations of the scattered beam can be accomplished through analyzer crystals (back scattering or triple axis) or polarizers (HYSPEC). Ideally it should be flexible enough to add more additional components without any complex developments.

This new type of components will have intrinsic characteristics (e.g. crystal analyzer space group) and behavior (e.g. polarization function). The new design has to consider those parameters as part of the component.

It is worth noting that for reactor sources, there are cases where the source component (a mandatory component!) has no added value. Usually the unique meaningful components are the sample, detector and eventually any component sitting in between

these two.

**Multi-technique instruments** Multi-technique instruments together with instruments that have beamlines that split or branch are another use case that should be accounted in the new system.

A new geometry can incorporate ideas from Monte Carlo codes such as McStas and McVine. In the next design of the geometry, a careful eye should be kept on making easy things easy to do (e.g. getting flight path and scattering angle of a detector pixel) while allowing for the flexibility to handle these other cases. The future implementation must aim at maintenance since most changes to the geometry system require changes to large portions of the rest of Mantid.

We envisage a complete separation of data and instrument. Both should be able to exist independently. A workspace can then be seen as the union of data and metadata linked to an instrument geometry.

Mantid should also be able to read an Instrument Definition File (IDF) in a standard 3D description format. As an example, the COLLADA (COLLAborative Design Activity) specification provides a standard to transport 3D assets between different tool sets. Software such as Blender, Maya, 3ds Max, etc can export data in the COLLADA format. A clear advantage of this approach is that the design descriptions of the instruments can be directly converted to a standard format read by Mantid. In addition, the user is not locked into Mantid to see the instruments in 3D. Although COLLADA supports constraints and physical materials, if more information is needed, the IDF can be complemented with parameters file, in a standard human readable format (e.g. Json or YAML).

## 4.2 Units

### 4.2.1 Current situation

Units is linked to geometry throughout the algorithm ConvertUnits. The current system is, understandably, heavily biased towards time-of-flight units.

### 4.2.2 Objectives

It needs to be expanded to more easily handle units that are used at fixed wavelength sources as well as conversion of log values (e.g. between Celsius and Kelvin, between picoCoulumbs and microAmp-hour).

The current workspace was thought with spectra in mind. The X units, for example, only support *bins*, i.e., the X axis has to be at least two values (the boundaries of the bin). In the case of fixed wavelength instruments with minimal wavelength spread the *bins* have no use.

## 4.3 Milestones

1. Make workspaces independent from the instrument geometry.
2. Functionality to concatenate, append or merge workspaces with different instrument geometries to create a new multiple geometry workspace.
3. Flexible implementation of instrument definitions. Specially for top components (e.g. *Analyzer*) without mandatory elements.
4. Add top components with intrinsic characteristics (e.g polarizer, crystal, etc).
5. Rethink the instrument definition files. Perhaps using standard language allowing viewing/editing in other tools.
6. Rethink the Units architecture in Mantid with additional flexibility to create new units using the "Open Closed Principle". I.e. the Units will allow its behaviour to be extended (i.e. create a new Unit) without modifying its source code.

# 5 The workspace types need to more efficiently reflect the science

## 5.1 Current status

Data containers in Mantid are called workspaces. In addition to the neutron data, these workspaces contain information about instrument geometry and parameters,

and experimental logs, such as temperature, chopper setings, and so on. The main workspace type in Mantid is the "matrix workspace". In the simplest form, the "workspace2D", it is just a collection of equal length arrays of numbers. It was originally designed to mimic the "raw" data files produced at ISIS. The structure of this container is ideal to store minimally processed time-of-flight data. When ORNL joined the project, the matrix workspace was extended to be able to store "event data". The event workspaces contain information about the time-of-flight, and maybe the external time when each neutron is detected. To keep compatibility with workspace2Ds, a histogram representation of the event list is also stored in the workspace.

In order to be able to process data related to single crystal diffraction and single crystal spectroscopy, which require multiple coordinates for a given point, the multi-dimensional workspace or "MDWorkspace" was introduced. There are two types of such containers, the "MDEventWorkspace" which contains a set of discrete points in a multi-dimensional space, and the "MDHistoWorkspace", which contains data on a regular grid.

For various uses in Mantid, there are other workspace types. "TableWorkspaces" can store information about fitting parameters. "PeakWorkspaces" contain representations of single crystal peaks.

## 5.2 Objectives

The design of workspace types and hierarchy was done on an ad-hoc basis, and will benefit from a major rethinking and reorganization. We believe that we should start with a complete review of the physics of neutron scattering, starting from data acquisition, and up to analysis, to better understand the requirements for data containers and interactions between them. We should also keep performance as a major part of the design.

As the Mantid project started to develop, several deficiencies of this workspace type become apparent. There is a need to remove the limitation of same number of bins. Instruments can have detectors at different distances, and different angles. This results in measurements with different ranges, and different resolutions in different spectra of a workspace. While having different ranges is possible at the moment,

Mantid currently requires the same number of bins. This means that for a particular spectrum one either has to pad/crop data, or use a bin size that is either too small or too large. Similarly, monochromatic monitors in a direct geometry scattering instrument have a very narrow time window where the count is non-zero. Keeping them in the same workspace, with the same number of bins, as the detector data is wasteful.

While manipulating MDHistoWorkspaces in python is relatively straightforward, the python interface for MDEvents is seriously underdeveloped. The interaction with ParaView would be simplified if the multidimensional (MD) workspaces borrowed concepts and/or interfaces from VTK.

As described in the hybrid computing (section 9), more work is needed to support for handling chunking and partial datasets.

Many of the changes and additions need to be made for moving instruments at constant wavelength sources. The current instrument geometry is tightly coupled to the idea that there is one position for every component (e.g. source, sample, detectors) for a measurement.

Finally, workspaces only contain one type of data (histogram, events, table, etc.) and users are required to keep track of sets of workspaces. Mantid needs a clear way to support linking workspaces together (e.g. monitor with data, data with normalization). For processed data, the correct physical representation would dictate the requirement to store both data and statistical weights. Keeping them in separate but linked workspaces would be a major step forward in bringing data structures to more closely reflect the science.



## 5.3 Milestones

1. As part of the documentation for Mantid, we intend to generate technique specific documentation. Once the physics requirements are identified, we should generate a list of containers that can hold each of the data types requested, in a way to be as close as possible to the scientific understanding (24 months).

   (a) Design workspaces for scanning instruments, at continuous neutron sources (12-24 months)

2. Expose more functionality of MD workspaces to python (12 months).

3. Allow linking of workspaces, and implement algorithms that use these links (12-24 months)

4. Support for chunking and partial data sets (24 months)

# 6 Experiment steering

## 6.1 Current situation

Currently, most data is measured for an amount of time or proton charge based on heuristics from experience with the instrument being used. We have no method to enable collection of data until a required statistical significance, or any other criteria, is meet.

There is very limited feedback from analysis into the current experiment. The use of other tools for crystal alignment (finding the UB) requires manual effort to feed back into the experiment. Other tools, such as CrystalPlan which is used to maximise the reciprocal space collection, also requires an antiquated method to transfer this plan back into the DAS controls.

## 6.2 Objectives

This could be greatly improved by adding a feedback loop from Mantid to the data acquisition that provides for smart stopping. This would allow for arbitrary criteria to determine that an individual measurement is complete. In the early stages, criteria will focus on things like fitted peak parameters. Future criteria could be stopping when the R-value of a Rietveld refinement is below a certain value.

Once this feedback mechanism is in place, the work can expand the capabilities. Mantid could be used to steer the experiment itself to more closely explore areas of interest. An example of this is configuring a temperature ramp to look for a phase transition then using Mantid to identify the temperature of the phase transition and

concentrate subsequent measurements in that area. This functionality would promote optimizing the measurement scheme during the experiment.

This can be achieved by integrating small components from Mantid in to the instrument control software used at SNS, Control System Studio [4] (CSS), or by allowing the remote execution of Mantid from within CSS with the returning parameters guiding the experiment.

The capabilities if the auto-reduction can be expanded also, with the correct metadata included in the data files (sample info, vanadium/background runs etc), complete automatic analysis could be achieved for a number of different, e.g. Rietveld fitting powder data.

### 6.3 Milestones

1. Develop a feedback method for Mantid $\rightarrow$ DAS for stopping criteria (12 months)

## 7 Advanced visualization

While the first neutron scatters had to manually setup their instrument and record neutron counts – in handwriting – one single point at a time, users at a modern time of flight spectrometer rapidly collect hundreds of gigabytes of data in less than one day. Rapid and effective viewing of of multi-dimensional (N$\geq$3) data should utilize all three dimensions of visual space. Lower-dimensional views not only show a much smaller quantity of the measured volume, but also introduce biases. In single crystal diffraction or inelastic scattering, for example, cuts in one and two dimensions primarily focus on lines and planes along high symmetry directions in the Brillouin zone.

### 7.1 Current situation

The VATES project is successful in bringing advanced visualization into Mantid. ParaView's visualization framework provides much of the functionality inside the VATES Simple Interface (VSI), and ParaView continues to undergo heavy development in

areas useful to Mantid. The current VSI implementation, however, is limited to serial operation on the local machine. Interactive volume rendering often exceeds the capabilities of a single CPU core and GPU. Views in the VSI primarily consist of two-dimensional cuts. Volume rendering is at most a final step to generate a high-impact figure after most analysis is already complete.

While much of Mantid is exposed to python, scripting within the VSI remains undeveloped. Recent work has brought Horace-style commands for generating multidimensional data, but interactions in the VSI remain heavily tied to the graphical user interface. Without the visualization capabilities available in MantidPlot and the VSI, IPython notebooks will remain a limited, secondary interface to Mantid.

The instrument view panel is also unavailable from python. Loading it requires is a manual step, limiting its usefulness in auto-reduction.

## 7.2 Objectives

ParaView's client/server mode would allow better use of ORNL's computing infrastructure and resources and scale from multicore workstations to petascale supercomputers. In this configuration, the Vates Simple Interface (VSI)is a separate application, replacing the pvclient GUI and providing a customized interface specifically tailored to data reduced in Mantid. The pvserver would either be run locally through Mantid (looking almost identical to the current setup) remotely on a larger workstation or cluster.

Refactoring the VSI to separate pvclient and pvserver binaries will allow use of departmental clusters such as CADES [5] and High Performance Computing (HPC) clusters such as TITAN [6, 7] for visualization. Ideally, it should be easy to build only the ParaView plugins. Submitting this limited package to Kitware for inclusion in future releases of ParaView would further lower the barrier for system administrators managing large HPC resources.

After these tasks are completed, then ORNL can better take advantage of advances in ParaView and VTK. Improvements to VTK and ParaView take advantage of heterogeneous computing. The OpenGL and OpenGL2 rendering engines offload some computation from the CPU to the graphics card. VTK-m [8] is a rewrite of VTK to provide support for fine-grained concurrency and heterogeneous computing.

With IPython Notebooks becoming a first-class interface to Mantid, the visualization capabilities available in MantidPlot and the VSI must be accessible from the notebook. IPython notebook integration is available through ParaView web [9]. ParaView web uses HTML5 technologies to render images in a web browser rather than requiring installed software. An additional set of wrapper scripts will simplify the amount of code required to perform common tasks within the VSI.

The instrument view panel should also be available from python. This capability will allow rendering of the instrument view to be part of autoreduction, providing a quick check for users.

## 7.3 Milestones

1. Modify Mantid to separate pvclient and pvserver [10] components, ensuring that all communication goes through ParaView's remote connection. Revise the VSI to open independently and be able to connect to a local or remote Mantid pvserver. (12 months, may require assistance from Kitware)
2. Rewrite Mantid's ParaView plugins to depend only on a small subset of the Mantid codebase (Requires section 1) (3 months).
3. Ensure that Mantid and ParaView's python libraries can both be loaded in the same python instance. Develop a set of wrapper scripts to more easily perform common tasks with the ParaView API. Try adding ParaView's python trace capability into the VSI. (Two months for initial work and to make further recommendations).
4. Study ParaView web and attempt to get rendering within the IPython window itself. (two weeks to study issue and make recommendation).
5. Before refactoring the instrument view panel, we should study whether this could be more simply written using VTK. Utilizing the same high-level libraries could simplify development work, unify the user interface, and potentially share improvements made for the VSI and other parts of Mantid. (two weeks to study issue and make recommendation).
6. Modify the instrument view so that it can be started as its own process. Investigate options for creating and saving a screen shot. Add python bindings necessary for use in an autoreduce script (3 months).
7. Replace generic ParaView properties panel with a custom panel better suited for the VSI. Consider removing the pipeline browser (3 months)

8. Allow multiple VSI windows to be open simultaneously with different workspaces and/or views in each window. (3+ months, requires assistance from Kitware)
9. Determine a strategy for either replicating functionality between slice viewer and spectrum viewer, or merging the two tools. (2 months)

# 8 Mantid connecting to advanced analysis and modeling

## 8.1 Current situation

## 8.2 Objectives

Ultimate goal would be to enable a workflow that allows *live* analysis of data as part of the experiment as demonstrated in the CAMM project [11]. The ability for a visiting scientist to not only perform the experiment during their visit but complete most of their data analysis would be a truly great benefit.

Include Mantid in the refinement framework for optimizing simulations to model experimental data. Mantid should be easily be called by existing workflow software (*e.g.* Kepler workflow and Dakota optimization) to allow the comparison between simulations and experimental data to be done within Mantid, along with the option of using Mantid curve fitting if required. Mantid can also be used as a common visualization tool for both experimental and simulation data.

To be able to take data collected and analysis it on a supercomputer with ease. To achieve this a workflow would need to be able to ...

Mantid components should be easy to include into any other software to allow easy utilization of Mantid's file load, fitting routines, etc. This would most easily be achieved by either a connecting to a smaller set of libraries (section 1) or SciPy styled library (section 2).

Having a more standard saved data file (e.g. NeXus) would allow easy integration into other analysis packages. And to allow easier combined analysis of data collected on different instruments. Making analysis of parameterized data collections runs easier.

Bindings into other languages would need to be easier to develop with to allow easy integration into analysis packages.

Algorithms developed in Mantid should be easily usable in other analysis software.

### 8.3 Milestones

## 9 Utilization of distributed and heterogeneous computing environments

Neutron event data are a significant challenge. Data sets can be so large, or complex, that traditional data processing applications are inadequate. These large datasets pose great opportunities as well as new challenges.

### 9.1 Current Situation

It is a fact that the attainable performance of Mantid is limited. Current computation times do not work well for multi-dimensional calculations or with large number of neutron events. If many datasets, or many chained algorithms, are used the processing time scale can reach tens of minutes for some instruments. Mantid has tried tackling this problem by making direct use of OpenMP [12], thread pools and limited use of MPI [13]. While OpenMP and thread pools are invisible to the user, MPI requires job submission to often a remote host (a cluster).

### 9.2 Objectives

As time passes more opportunities become available. However, the physical size of the files will continue to be quite large if not larger. It is thus clear that Mantid needs to be fairly high performant to allow new state of the art data reduction and therefore leverage science.

### 9.2.1 Parallel Processing

Current Mantid version possesses algorithms that take advantage of Parallel Processing technologies. Carefully inspection of CPU usage when running some algorithms shows that its usage is very sub-optimal. These bottlenecks must me investigated in the first place and the code should be refactored where possible to allow better CPU usage.

Ultimately, and if needed, the way as the data is stored, organised and accessed inside a workspace must be revised and improved to allow better parallel access. E.g., data should be kept independent (e.g. at spectra or eventually at detector level) for as long as possible in the data reduction process.

**OpenMP and Thread Pools** As originally specified, OpenMP is primarily suited to loop-level parallelism. If data structures allow, it is usually easy to transform a serial loop into a parallel loop by simply using compiler directives.

Since Thread Pools has an analogous functionality, investigation is needed to deprecate one of the technologies and keep standards unique. Ultimately both technologies may be deprecated if a highly efficient solution will be found (e.g. Parallel STL [14] for C++, multiprocessing or C extensions that release the GIL for Python). More research is needed...

**MPI** MPI main objective is to run CPU intense jobs in a remote cluster. Contrary to OpenMP/Threading (optimal for loop level fine grain parallelism), generally MPI is considered optimal for process-level coarse parallelism. Since remote job submission implies some non negligible communication overhead, combining MPI and OpenMP/Threading parallelization to construct a hybrid program appears to be an entirely satisfactory solution to take into consideration.

Care must be taken and trade-offs must be considered when juggling this too technologies. MPI introduces communication overhead while OpenMP/Threading introduces overhead due to thread creation and increased memory bandwidth contention.

**GPU** Altough GPU processing is not supported in the current Mantid versions, as proof of concept, there is an algorithm for OpenCL [15], and ideas for using OpenACC [16].

GPU processing technologies (CUDA [17] and OpenCL) need a considerable effort in the development process. In addition, it may be not applicable to a wide range of algorithms as to notice some improvements at least 10000 intense operations have to be launched at the same time.

OpenACC is a recent technology which aims to simplify parallel programming of heterogeneous CPU/GPU systems by creating a set of annotations (similar to OpenMP) which allow the code to run on either CPU or GPU. It is only available through commercial compilers from Cray and The Portland Group (PGI). Since ORNL possesses PGI licenses, OpenACC implementation should not be fully discarded. OpenACC directives are very similar to those used by OpenMP, thus requires less development effort than CUDA or OpenCL. In the case where algorithms are unique to the SNS, can be parallelized and are suited for GPU processing ($\gtrsim$ 10000 threads), then OpenACC should eventually be considered.

### 9.2.2 MPI and HPC

Coupling Mantid with the High-Performance Computing (HPC) facilities available at the ORNL, may be the best short term strategy to implement new high performant data reduction strategies at the SNS. This will lead to new opportunities, and thus, new paradigms can be seen ahead of time and integrated into the core data reduction strategy. Live data analysis can take full advantage of the HPC with several multi-parameter data reduction jobs launched simultaneously in parallel. The results of these jobs will eventually be fed back to the data collection system to better gain control over the experiment. Ultimately this will make data collections shorter and optimize beamtime.

An intelligent load balancing and queue system will make sure that the user obtains the results in a timely manner. A queuing system will to make users aware of their jobs position in queue and respective waiting time. Jobs for local users or live data analysis can eventually be attributed high priority on accessing the HPC system.

It is worth noting that Mantid will always be available as as desktop application.

However, the user, whenever possible, has the choice to submit the reduction jobs to a HPC cluster. In addition to the ORNL HPC, there is also the possibility of performing calculations using cloud services (e.g. AWS [18], OpenStack [19]).

### 9.2.3 Load on demand

Mantid datasets can be larger than the machine physical memory, forcing interactive workflows to include the hard drive. I/O speed will increase at a significantly slower pace than computation speed and number of cores. Users of Mantid have highlighted excessive memory usage and slow read/write times to the hard disk.

Instead of loading all the dataset into physical memory, the paradigm must be loading only what it is necessary, thus improving performance of disk I/O.

Some of the Mantid Loaders read the data file and load all the data to memory. While this procedure is adequate for small datasets, it can be very resource consuming for large datasets. The computer's physical memory can overflow to disk, leading to disk subsystem waits.

Future developments should access and improve the disk performance of some Loaders (e.g. LoadEventNexus). HDF file format can play a crucial role in improving I/O performance. HDF has several native functionalities that can improve I/O performance: binary format storage, compressed data (less data being transferred from the disk), indexed and partitioned data. Partitioned data – data chunking – is generally beneficial to I/O performance in very large arrays as it may reduce the number of seeks through a data array.

To take advantage of these native features, an HDF library must be used (e.g. H5Py [20]). Despite the practicality of the NeXus library for neutron data treatment, it exposes very little functionality necessary to tune the I/O performance.

It is known that a key to I/O performance in HDF is the number of disk accesses that must be made during any I/O operation. The choice of the the chunk size plays a major role in this effect. The chunk size should be approximately equal to the average expected size of the data block needed by the application [21]. It is thus essential to coordinate efforts with the DAS group to set standards and share knowledge of how chunked data arrays are created and accessed.

Note that this model should be aligned with the parallel processing developments, by intelligently streaming data from disk and by leveraging all the cores of a modern CPU. Given the architecture of the instruments and the organization of the NeXus files (data grouped independently by detector), only one detector/panel/block of the detector should be loaded to memory and assigned to a core/CPU at a time. Resources should be released after usage to avoid memory overflow. Whenever possible, the algorithms should operate on blocks rather than on entire row or column, so that data loaded into the faster levels of the memory hierarchy are reused.

### 9.3 Milestones

Initial developments should focus on selecting algorithms that treat the spectra independently (pixel/bank/neighbor independent) and thus can be theoretically parallelized. Those that can run in parallel must take full advantage of OpenMP. Even if they use OpenMP already, the bottlenecks should be found and optimizations should be applied to achieve maximum benefit. Algorithms that are hardly executable on a Laptop PC, in addition to OpenMP, should have a double implementation in MPI. Users should not be forced to run algorithms using the SNS infrastructure, however that possibility should be available.

1. Research Multithreading functionalities. Substitute thread pools by OpenMP or use a new Multithreading framework.
2. Access slow algorithm to find eventual bottlenecks
3. Parallelize straightforward tasks (independent spectra / detectors tasks).
4. Implement hybrid OpenMP/Treading/MPI where OpenMP/Treading alone is not enough.
5. Access data structures and Load algorithms to load to memory only the essential data.
6. Prepare the data structures for chunk processing taking advantage of the HDF data chunking when NeXus files are available.
7. Access current algorithms using parallel processing and identify the bottlenecks.

## References

[1] Scipy homepage. `https://www.scipy.org/`.

[2] Mantid framework architectural design document. http://github.com/mantidproject/documents/blob/master/Design/ArchitectureDesignDocument.doc.

[3] Simon Heybrock. *Performance Analysis of Mantid for ESS*. 2015.

[4] Control system studio. http://controlsystemstudio.org/.

[5] Cades. http://cades.ornl.gov/.

[6] Titan. https://www.olcf.ornl.gov/titan/.

[7] Titan becomes worlds largest gpu-powered visualization system for scientific discovery. http://blogs.nvidia.com/blog/2015/11/17/titan-largest-gpu-visualization/.

[8] Vtk-m homepage. http://m.vtk.org/index.php/Main_Page.

[9] Paraview web. http://www.paraview.org/web/.

[10] Paraview server documentation. http://www.paraview.org/Wiki/Setting_up_a_ParaView_Server.

[11] *Live CAMM* simulations of anharmonic phonons in $SrTiO_3$ measured on HYSPEC. http://camm.ornl.gov/highlights/1/.

[12] Openmp homepage. http://openmp.org/wp/.

[13] Mpi homepage. http://www.mcs.anl.gov/research/projects/mpi/.

[14] Parallel stl. https://parallelstl.codeplex.com.

[15] Opencl homepage. https://www.khronos.org/opencl/.

[16] Openacc. http://www.openacc.org/.

[17] Cuda homepage. http://www.nvidia.com/object/cuda_home_new.html.

[18] Amazon web services. https://aws.amazon.com/.

[19] Openstack. https://www.openstack.org/.

[20] Hdf5 for python. http://www.h5py.org/.

[21] Hdf performance issues. https://www.hdfgroup.org/release4/doc/UsrGuide_html/UG_Perform.html.