Table of Contents

1Introduction 2

1.1Purpose of this Document 2

1.2Scope of this Document 2

1.3Context of this Issue 2

1.4Definition of Terms 2

2Detailed Design 3

2.1Context 3

2.2Overview 3

2.3Automated Processing 3

2.4Configuration of Processing Task 3

2.5Splitting up Processing Task 3

2.6Parallel Extensions to Mantid 3

2.7Mantid Task Submission 3

2.7.1Overview 3

2.7.2GUI 4

2.7.3Data Transfer 4

2.7.4Configuration / Setup / Security 4

2.8Parallel IO 4

[1.1Purpose of this Document 2](#__RefHeading__125_1632753529)

[1.2Scope of this Document 2](#__RefHeading__125_1632753529)

[1.3Context of this Issue 2](#__RefHeading__125_1632753529)

[1.4Definition of Terms 2](#__RefHeading__125_1632753529)

[2Detailed Design 3](#__RefHeading__125_1632753529)

[2.1Context 3](#__RefHeading__125_1632753529)

[2.2Overview 3](#__RefHeading__125_1632753529)

[2.3Automated Processing 3](#__RefHeading__125_1632753529)

[2.4Configuration of Processing Task 3](#__RefHeading__125_1632753529)

[2.5Splitting up Processing Task 3](#__RefHeading__125_1632753529)

[2.6Parallel Extensions to Mantid 3](#__RefHeading__125_1632753529)

[2.7Mantid Task Submission 3](#__RefHeading__125_1632753529)

[2.8Parallel IO 4](#__RefHeading__125_1632753529)

# Introduction

## Purpose of this Document

This document describes the detailed design of the components required for parallel data analysis of the Mantid Framework.

It is based on the design of the Mantid Framework specified in the Architectural Design Document [ADD]

It will form the basis of the development of this aspect of the framework and act as a guide for maintaining the system.

## Scope of this Document

These requirements cover the development of the parallel/batch data analysis aspect of the Mantid Framework and MantidPlot.

## Context of this Issue

This is the first draft of the LED-DDD derived from the ADD and after internal review will be updated and used as a basis for the development of the system.

## Definition of Terms

|  |  |
| --- | --- |
| ADD | The Architectural Design Document (this document), the high level design document for the entire system. |
| URD | The User Requirements Document, records the users’ requirements for the system. |
| SRD | The Software Requirements Document specifies the behaviour of the software system. |
| API | Application Programming Interface defines the interface through which two programs may interact. |

# Detailed Design

## Context

While live data analysis of the neutron data stream will provide users with much needed timely feedback and the ability to make informed decisions on steering subsequent experiments, a number of challenges must be addressed for the Mantid software to carry out analysis of datasets in the 10-100’s of gigabytes or more. These datasets are currently processed in Mantid using big-memory analysis nodes (128-256 GB RAM) on which Mantid currently provides node-level parallelism through the use of OpenMP. As dataset sizes continue to increase, the memory requirements alone will require parallel analysis techniques. In addition to a large memory footprint, many analysis workloads currently require staging datasets from one file system to another, prior to parallel data analysis, in such an environment, the time for this staging will dominate the analysis workflow. At the SNS there are plans to implement a parallel file system to help alleviate this bottleneck. Analysis tasks that are heavily compute or memory footprint bound will be accelerated by developing parallel analysis capabilities within Mantid. In this architecture, Mantid analysis tasks can be run concurrently on an analysis cluster and the local experimental workstation with summary analysis results displayed on the experimenter’s workstation.

## Overview

Once the NeXus raw data file has been written (and is accessible to be read) an automated (user defined) process will be launched using appropriate computational resources. This processing job can either be a simple mantid ‘serial’ process, which may use thread parallelisation, or split up into separate parallel jobs.

We will need a set of strategies for intelligently splitting up larger jobs into smaller ones, this will also be of use in the future for running reduction procedures on machines of limited resources, for which case we can just run each of the jobs in serial when we don’t have access to a parallel resource (but this use case is outside the scope of this project).

Various parts of the Mantid Framework will need modification in order to efficiently make use of any parallel resource (such as a cluster). Some prototype work has been done in this area and is detailed in the document <https://github.com/mantidproject/documents/blob/master/Design/ParallelAnalysis/MPI_Prototype.odt>).

We will also look at how a user will easily submit and monitor jobs using a backend parallel compute resource.

## Automated Processing

Pete / Shelly…

## Configuration of Processing Task

Pete / Shelly…

## Splitting up Processing Task

Pete

## Parallel Extensions to Mantid

Russell / Sarp

## Mantid Task Submission

### Overview

We want to be able to submit and monitor parallel jobs from within MantidPlot. This implies some kind of network communication between the client running MantidPlot and the parallel cluster.

Condor is a cluster-management package from the University of Wisconsin (<http://research.cs.wisc.edu/condor/>) that should fit our needs quite well. It is designed to schedule jobs on otherwise unused computing resources (for example, idle computers in a student lab), but will function just fine on a dedicated cluster. More importantly for the Mantid project, Condor is designed to allow remote job submission. It has a complete API for submitting jobs, monitoring jobs and retrieving results over the network via SOAP calls. That means that we only need to write a minimal amount of code for MantidPlot to actually call the Condor API's.

### GUI

I'm envisioning another box on the right side of the window below the algorithms box. This one would be “Remote Algorithms” and would list the tools that Russell has developed – see 2.6. (The list of available algorithms could either be hard-coded, or - with a little bit of work - we could probably query the cluster for the list of available algorithms.) Clicking on one of these remote algorithms would open a dialog box containing user-selectable options (number of processes, input files, etc...) and a “Submit” button.

There should also be a “Job Monitor” button (or perhaps a menu choice) that will bring up a dialog showing current and completed jobs.

### Data Transfer

There's going to be some interesting issues with transferring data. (Or, more specifically, avoiding data transfers wherever possible.) If necessary, Condor's API's do allow for transferring the input (and output) data to the cluster, but it would be far more efficient to access it locally via a shared filesystem. Ideally, MantidPlot would know if the data it's working on is located on the shared filesystem. Otherwise, that is something that will have to be specified in the options dialog before the job is submitted.

### Configuration / Setup / Security

We will need a dialog box in MantidPlot to hold various configuration options, such as the address of the cluster head node. Given that there is only one cluster right now and that there will probably never be more than a few, it seems practical to hard-code all the necessary configuration options and then let the user just select the cluster from a drop-down list.

### User Authentication and Authorization

Condor accepts a number of different authentication/authorization options. We would like something that would allow the user to authenticate once yet allow multiple connections to the cluster. For example, forcing the user to enter a password every time he/she wanted to check on the status of a running job would not be ideal.

One possible option is to use X.509 certificates. We can set up a server (either the cluster head node or some other convenient server) that will issue X.509 certificates. The user logs in and obtains a certificate that is valid for a reasonable amount of time (12 hours, for example). This certificate is then used for all of the Condor API calls. We can also probably integrate the act of obtaining the certificate into MantidPlot so that needn't even be aware that certificates are in use.

## Parallel IO

In order to make most efficient use of a parallel file system we may need to implement addition techniques and methodology for some IO tasks within Mantid. Until we have access to a Parallel File System, it is impossible to determine what these changes are and if indeed they are necessary.

This will be revisited once we have done sufficient testing and parameterisation using a parallel file system.