

---

---

## DMSC-DRAM: LIVE DATA PROCESSING

---

---

	Name	Role/Title
<b>Owner</b>	Torben R Nielsen	Scientist, DRAM
<b>Reviewer</b>	Thomas Holm Rod	Group leader, DRAM
<b>Approver</b>	Jonathan Taylor	Head of DMSC

<b>TABLE OF CONTENT</b>	<b>PAGE</b>
EXECUTIVE SUMMARY .....	4
1. INTRODUCTION .....	5
1.1. DMSC High-Level-Milestones.....	5
1.2. Infrastructure setup: Pushing live data through the different services.....	5
1.3. DRAM related HLM.....	9
2. HLM-1: DATA SET AND RATE FOR ACCEPTANCE TEST DEFINED .....	10
3. HLM-2: DATA REDUCTION ON THE FLY.....	11
3.1. Mantid workflow .....	11
3.2. Data set 1 - LoKI Low count rate .....	12
3.2.1. Instrument View .....	13
3.2.2. Mantid system performance .....	13
3.3. Data set 3 - SANS2D Low count rate .....	15
4. HLM-3: INTEGRATION OF ANALYSIS & REDUCTION SOFTWARE .....	15
4.1. Data Analysis .....	16
4.2. Data set 1 - LoKI Low count rate .....	16
4.3. Data set 3 – SANS2D_LOW count rate .....	17
5. HLM-4: SOFTWARE PERFORMANCE AND INTEGRATION TESTING VALIDATED .....	18
5.1. Data can be reduced and analyzed on the fly for expected flux of <INSTR> when in full operation .....	18
5.1.1. In-Kind contribution from ISIS .....	18
5.1.2. Work done by DMSC .....	19
5.2. Post batch processing can be performed on CPH cluster....	21
6. HLM-5: PROOF OF OPERATION (REDUCTION) .....	21
6.1. Data are received from DAQ & Streaming system .....	21
6.2. Data can be post re-reduced on cluster.....	22

6.3.	Define the scaling parameter for the architecture.....	23
7.	HLM-6: PROOF OF OPERATION (ANALYSIS) .....	23
7.1.	Data can be analyzed on the fly for flux.....	23
7.2.	Post batch processing can be performed on cluster .....	25
8.	FINAL COMMENTS.....	26
9.	ACKNOWLEDGEMENT .....	27
10.	GLOSSARY.....	27
11.	REFERENCES .....	27
12.	DOCUMENT REVISION HISTORY .....	28
13.	APPENDIX: SOME PRACTICAL DETAILS.....	28
13.1.	LoKI – Dataset 1.....	28
13.2.	LoKI – Dataset 2.....	30
13.3.	Batch fitting on DMCS cluster .....	31

Review

## EXECUTIVE SUMMARY

This document describes each of the six milestones assigned to the Data Reduction, Analysis and Modelling (DRAM) group in to document completion of the ESS construction phase. It goes through the milestones and describes one by one how they have been fulfilled. It is documented that all milestones are completed and that it can run on infrastructure provisioned by the Data Systems and Technologies Group at ESS-DMSC. A detailed LoKI instrument model has been used as test case for demonstrating the most demanding milestones in terms of data rates. Changes to Mantid has been required in order to make it able to cope with the expected LoKI flux and workflows needs to be designed carefully. Hence, Mantid is running at its limit, but with anticipated improvements in hardware performance and the ongoing work on improving the underpinning work space architecture (i.e. the scipp project) this is currently not considered high risk. The Mantid improvements and associated performance tests are discussed in detail in a report from the ISIS in kind partner. In conclusion, ESS is now ready, from a software perspective, to run data reduction and analysis both live and in post processing mode on infrastructure that the Data Systems and Technology group plans to provision for this purpose.

The execution of the project leading up to these milestones have highlighted a few issues that warrants to be addressed in future work; Resolution functions for SANS are generally not fully understood for spallation sources, a fact that so far has been ignored at other spallation sources. However, because ESS is a long pulse source this approach will most likely not be satisfactory for ESS. Work on obtaining better resolution functions has been initiated with the LoKI instrument team. Moreover, the DMSC hardware resources have been stretched to its limit for demonstrating these milestones, i.e. becoming ready for operation of LoKI. Because we need to go through the same procedure for other instruments, of which some have higher expected flux than LoKI, more hardware resources are required for testing in the initial phase.

The work presented here has been performed in collaboration with members of the Data Systems and Technologies Group, the Experiment Control and Data Curation Group, and the ISIS in kind partner to DMSC.

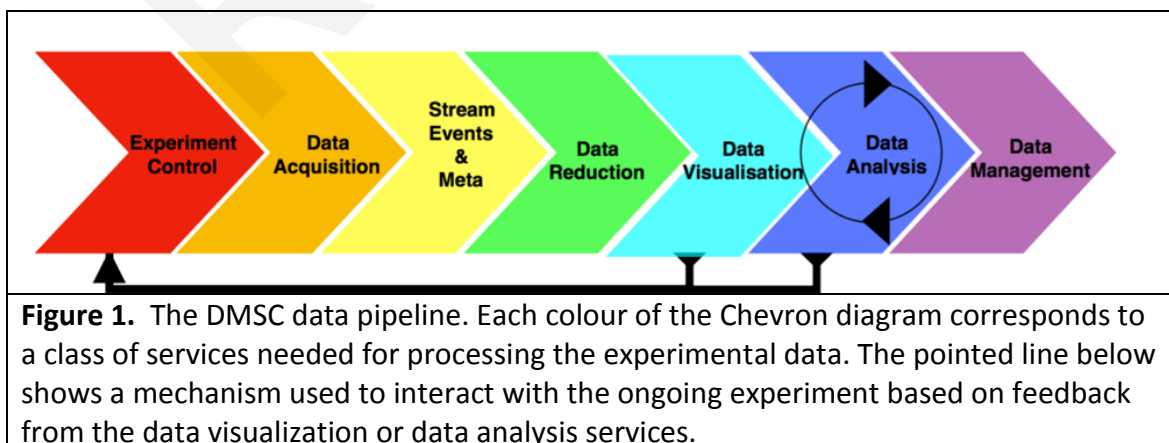
## 1. INTRODUCTION

This document goes through each of the DMSC high level milestones assigned to DRAM in the ESS construction phase (i.e. until the end of 2019). The DMSC milestones are listed and described on the Confluence page [High Level Milestones](#). The purpose is to document that the milestones have been fulfilled and to serve as a future reference. Each milestone is discussed in a separate section together with a discussion of how they have been fulfilled and potential bottlenecks. The document concludes with recommendations for future work.

### 1.1. DMSC High-Level-Milestones

Generally speaking the DRAM milestones address the question of DMSC being able to deliver a data reduction and analysis framework that can cope with the expected high neutron count rates at ESS. Stated differently: Do we see any bottlenecks in the current design for live and post data processing. Figure 1 shows the DMSC vision for the data flow during an experiment, from running and steering the experiment, over to data acquisition to data analysis and data management as the last steps. At each step in the pipeline data is received and then passed on to the next data service. The HLM focus on demonstrating that we are able to process the data flow for the expected high neutron flux at the ESS instruments.

The LoKI instrument with a neutron count rate of  $>1E7$  counts/sec was chosen as the test case for these milestones. This choice reflects two major aspects: the data pipeline must be able to smoothly process data from beamlines with a high neutron count rate and detectors with high number of pixels. The LoKI instrument will steer the transformation of the DMSC design towards a working DMSC architecture for live data.



**Figure 1.** The DMSC data pipeline. Each colour of the Chevron diagram corresponds to a class of services needed for processing the experimental data. The pointed line below shows a mechanism used to interact with the ongoing experiment based on feedback from the data visualization or data analysis services.

### 1.2. Infrastructure setup: Pushing live data through the different services

To test the data pipeline for LoKI we setup a virtual experiment and a data pipeline to mimic the data flow shown in Figure 1. However, compared to the pipeline shown in

Figure. 1, we do not address experimental control and data management in the DRAM-HLMs. The purpose of the “Data Acquisition” part is to take data of the detector and deliver “good” neutron events. In the virtual setup we replace this part with McStas simulations. The purpose of the “Stream Events & Meta data” part is to distribute the data stream to other services in the data pipe line. This part is taken care of by the Kafka cluster. The purpose of the “Data Reduction” part is to reduce event data and make it ready for a latter data analysis. The data reduction step is done by Mantid. The purpose of the “Data Analysis” part is to study the experimental data and compare to a theoretical model. For the SANS beamlines, SASVIEW is used for the data analysis.

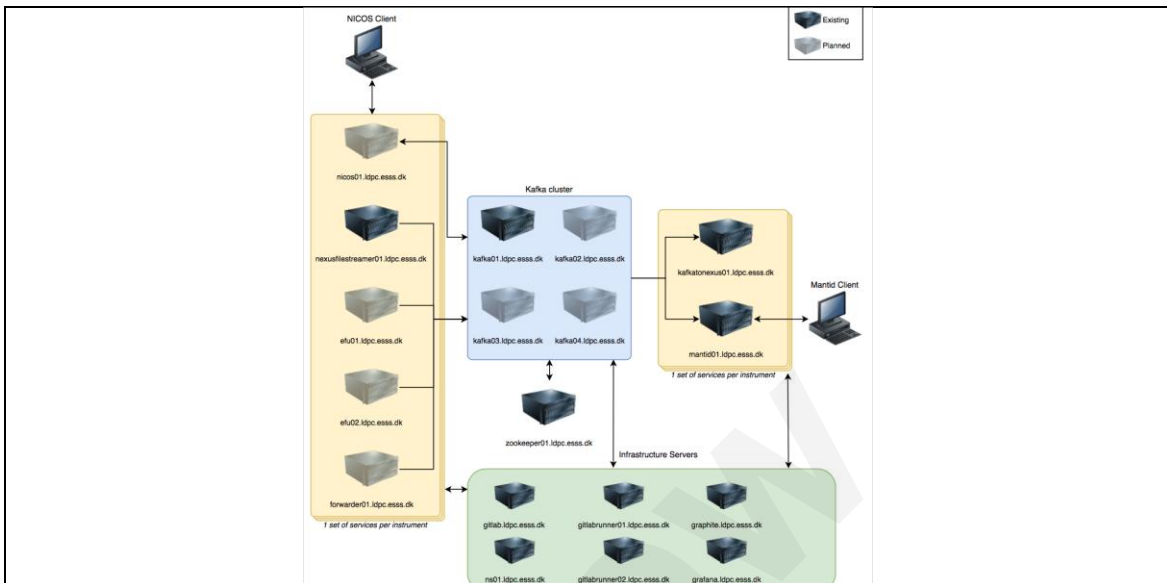
As mentioned above, each part in the data pipeline, the Chevron diagram in Figure 1, corresponds to a class of data processing services. All services will run on the DMSC compute infrastructure. The infrastructure setup used for the DRAM-HLM has been developed, tested and implemented by the DMSC Live-Data Processing-Coordination ([LDPC](#)) project. Figure 2 below shows the network diagram for processing Live Data, and how the services are interconnected. All clients used in the DRAM-HLM are provisioned using [Foreman](#) and configured using [Puppet](#) where all the configuration files for the clients are under version control hosted at the DMSC GitLab server ([gitlab-ldpc](#)). The LDPC Puppet setup can deploy all the services needed for live data on either the DMSC virtualization hosts or on the DMSC cluster system, i.e. on different networks/hardware setup. This gives a huge degree of flexibility for testing different aspects of the full data pipeline. If, we e.g. need a “beefy” Mantid client with much memory and many cores we can by choice deploy on one of the DMSC cluster nodes.

The starting point for this feasibility study is a McStas generated NeXus event file. This will play the role of an event list generated by the detector event formation unit (EFU). The McStas generated event list is read by the NeXusFileStreamer client. The NeXusFileStreamer will then feed all the neutron events into the data pipeline, further on to the Mantid client and finally to the analysis client. Metrics for each client system performance can be monitored by [Grafana](#). For testing the system, one needs in principle only to login to two clients: 1) start the event data stream on the NeXusFileStreamer client, and 2) start the reduction and data analysis client in the Mantid client. See information on Confluence on how-to use the live-data pipeline ([Quick start guide](#)).

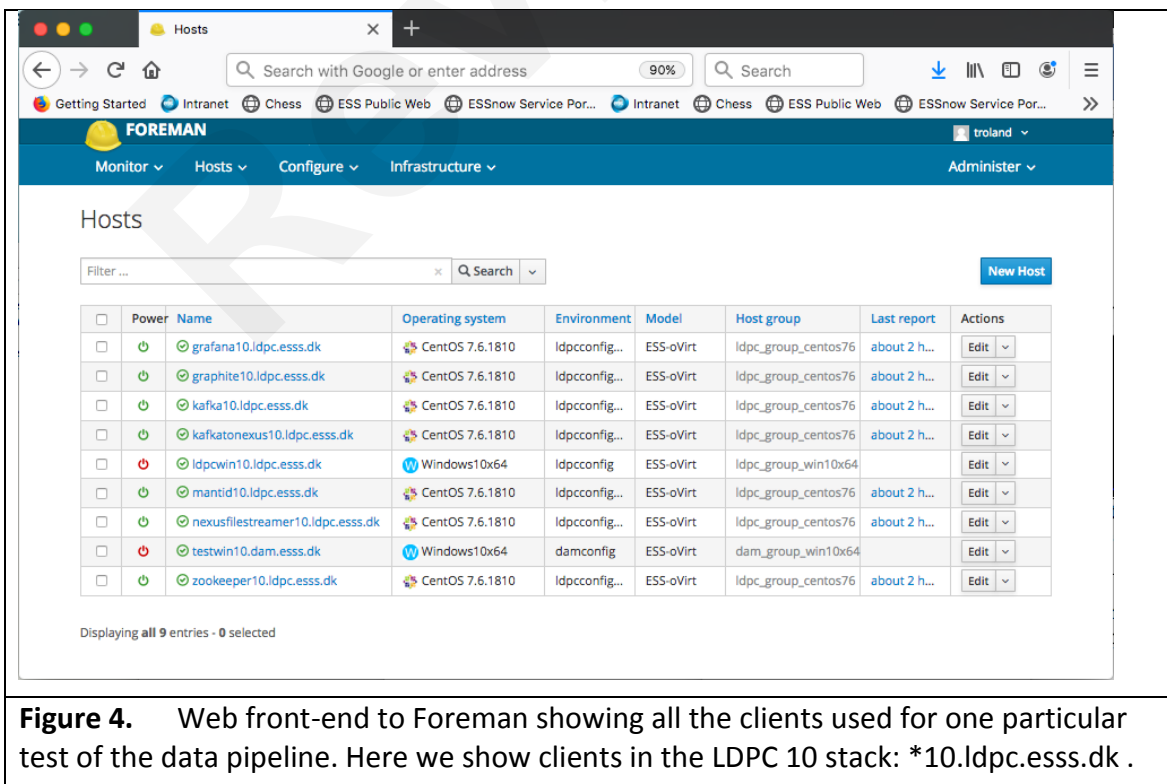
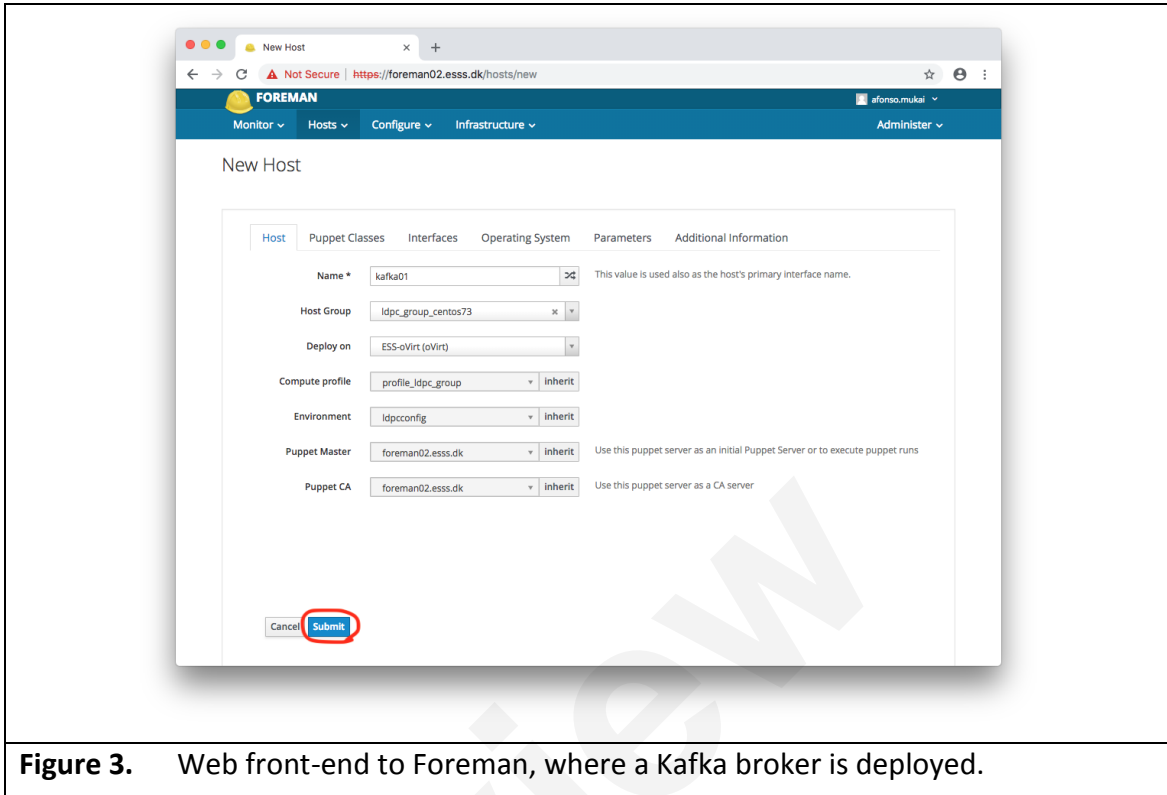
The data reduction part in the data pipe-line is potentially the limiting factor for the DRAM-HLM. The clients are all connected on a 1Gbs network, which is capable of keeping up with the data rates listed in Table 2. Similar, writing to disk at the data rates in Table 2 is also feasible with current technologies. Thus, in the DRAM-HML focus is on the data reduction part.

Given the many different clients used in the data pipeline, we need a fast and reproducible method to rebuild one or all services if either a new feature has been added and/or if a new system dependency is required somewhere along the data chain. Here Foreman and Puppet are used to leverage this inter-dependency problem. Figure 3 shows how to deploy a new Kafka broker using the Foreman web-frontend. Foreman can also be used to rebuild an existing client, and during this rebuild any new features or dependencies will be deployed on the Kafka broker. The status of the VM’s can also be

monitored in Foreman, where specification for the VMs are listed along any error reports. Figure 4 shows the list of VMs used for one particular test case.



**Figure 2.** Network diagram showing all the different clients which constitute the infrastructure needed for processing live data. The event data stream is started on the NeXusFileStreamer, which sends data to the KafkaBroker, which then passes data further onto Mantis and the FileWriter. All the clients use in data pipeline is referred to as the LDPC stack, consisting of 7 services: Zookeeper, Kafka, NexusFileStreamer, NexusToKafka (file writer), Mantis, Graphite, and Grafana.





### 1.3. DRAM related HLM

The DMSC HLM are listed on Confluence, see [High-level DMSC Milestones \(old versions\)](#).  
 The relevant HML for the DRAM group are listed in Table 1 below.

<b>Table 1 HIGH LEVEL MILESTONES RELATED TO THE DMSC-DRAM GROUP</b>	
<b>No. / Title</b>	<b>PCC / Definition of Readiness</b>
<b>HLM-1</b> Data set and rate for acceptance test defined	<b>ALL</b> Data set and rate for acceptance test is well defined but may still need to be generated.  Candidates are SANS2D modified to have flux expected from LoKI, or LoKI instrument model data, and test beamline for low flux.  Called <INSTR> in other milestones.
<b>HLM-2</b> Data reduction on the fly	<b>ID</b> Data can be reduced on the fly for <INSTR> for acceptance test but only low flux
<b>HLM-3</b> Integration of Analysis & Reduction Software	<b>ID / DAM</b> Data can be reduced and analyzed on the fly for limited flux for instrument(s) chosen for acceptance test
<b>HLM-4</b> Software performance and integration testing validated	<b>ID / DAM</b> 1. Data can be reduced and analyzed on the fly for expected flux of <INSTR> when in full operation  2. Post batch processing can be performed on CPH cluster
<b>HLM-5</b> Proof of operation (reduction)	<b>ID</b> 1. Data can be reduced on the fly for expected flux of <INSTR> when in full operation. Data are received from DAQ & Streaming system  2. Data can be post re-reduced on cluster  3. Need to be able to define the scaling parameter for the architecture rather than actually physically demonstrate

<p><b>HLM-6</b>          Proof of operation (analysis)</p>	<p><b>DAM</b></p> <ol style="list-style-type: none"> <li>1. Data can be analyzed on the fly for flux of &lt;INSTR&gt; when in full operation. Data received from Mantid, which in turn receives data from DAQ &amp; Streaming system.</li> <li>2. Post batch processing can be performed on cluster</li> </ol>
--	--

## 2. HLM-1: DATA SET AND RATE FOR ACCEPTANCE TEST DEFINED

*Data set and rate for acceptance test is well defined but may still need to be generated.*

To demonstrate the DRAM HLMs the [LoKI](#) and [SANS2D](#) beamlines were chosen. The LoKI beamline was chosen as it is one of the first beamlines to come online at ESS. The SANS2D beamline was chosen for comparison reasons and secondly it is a beamline the DRAM group has studied before.

The demonstrations of the milestones are split into two parts. The first part, a feasibility study using limited neutron flux on the detectors, showing that we can push live data through the different services constituting the entire data pipeline from detected neutrons to analyzed data. The second part, a volumetric study using high neutron flux on the detectors, showing that the data pipeline can cope with the expected detected neutron flux once ESS is in full operations. [McStas](#) will be used to generate the event data for the LoKI beamline in “low” and “high” count rate limits. For SANS2D we will for comparison reasons also study the “low” count rate limit. For both SANS beamlines we will use the scattering sample [SANS benchmark2](#), as it has been used previously by the Detector Group in Lund for benchmarking studies in previous Toll Gates reports for LoKI. Table 2 below shows a list of the event data used in this document. For data set 1 and 2, all instrument settings are kept fixed, except the sample which is allowed to change. The scattering sample SANS\_benchmark2.comp is a multi-sample McStas component with a collection of different SANS samples commonly used. The data sets are available on the DMSC [gitlab](#) repository, and see Section 13 (Appendix) for further information on the test cases.

Data set no.	Instrument	Scattering Sample	Detector Neutron Count Rate
1	LoKI	SANS_benchmark2.comp no. 5	Low (21.000 #/sec)
2	LoKI	SANS_benchmark2.comp no. 3	High (1.20E7 #/sec)
3	SANS2D	SANS_benchmark2.comp no. 5	Low (1.000 #/sec)

### 3. HLM-2: DATA REDUCTION ON THE FLY

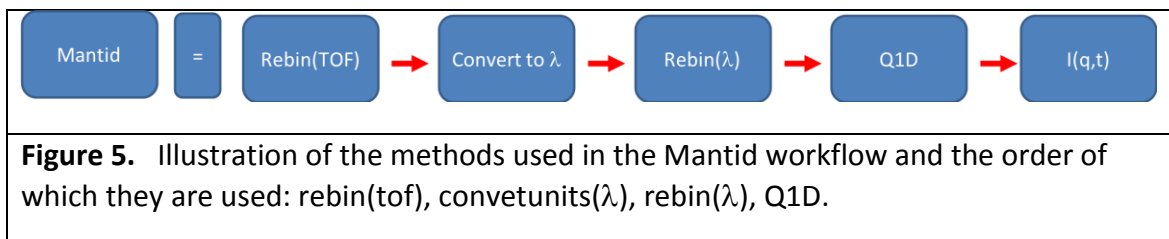
*Data can be reduced on the fly for <INSTR> for acceptance test but only low flux*

This milestone is a feasibility study designed to show that a limited neutron flux on the detectors can be pushed through the different live data services constituting the entire data pipe-line from detected neutrons to analyzed data. For this milestone we will use data set 1 and 3 defined in Section 2.1.

#### 3.1. Mantid workflow

A Mantid workflow involves linking datasets and operations on the datasets in a predefined way. Each operation on the dataset can be configured in different ways; e.g. one needs to decide between working with the data in event mode or in histogram mode, and for histogram data one also needs to decide on the total number of bins used. The total “Mantid Workflow” can thus be configured in several different ways, and if needed one could try to optimize the workflow for speed and better usage of system memory.

Figure 5 shows the schematics of the Mantid workflow used in the DRAM-HLM. First events are rebinned in time-of-flight, followed by a conversion to wavelength, then a rebinning in wavelength, and finally then a reduction method to give  $I$  vs  $q$  (intensity versus scattering vector  $q$ ). Here both the TOF and lambda workspace are kept, to have more than one workspace in the workflow. The reduction method Q1d is used for generating the scattering curve. (Notice that e.g. beam alignment or sample can subtraction is not needed here to process event data from the idealized virtual McStas experiment.)

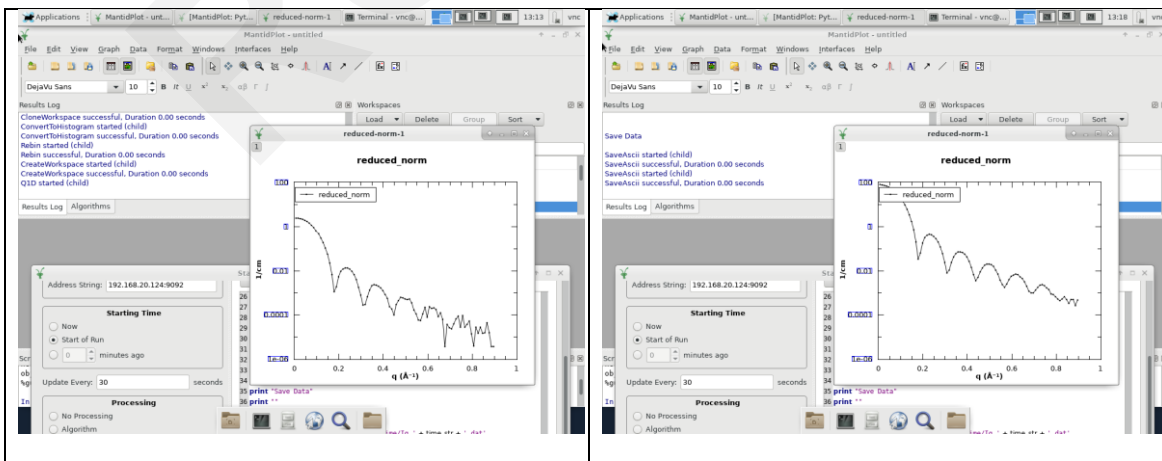


**Figure 5.** Illustration of the methods used in the Mantid workflow and the order of which they are used: rebin(tof), convetunits( $\lambda$ ), rebin( $\lambda$ ), Q1D.

### 3.2. Data set 1 - LoKI Low count rate

The McStas simulation for the “low” count rate regime is based on using the McStas small angle neutron scattering kernel [SANS\\_benchmark2.com](http://SANS_benchmark2.com) (model no. 5) and the full McStas instrument model for [LoKI](http://LoKI). The details of the detector layout are based on input from STFC LoKI team. A detector resolution of 2.5mm x 2.5mm were chosen for all nine detector panels. This gives a total number of pixels of more than 1E6 detector pixels. The total neutron count rate for this particular setup is > 21.000 #/sec. A corresponding event list were generated with a total count time of 305 seconds.

As described in Section 2.2 the data-pipeline is tested by starting the NeXusFileStreamer, and then after a manual configuration of Mantid Live Data, the time evolution of detected intensity and the reduced data can be monitored in Mantid. The details of the reduction workflow are described in Section 8. Loosely speaking, Mantid Live Data has a build-in Kafka subscriber client which can fetch neutron events from the Kafka cluster and then ingest the events into a Mantid event workspace. Once the streamed data are in a Mantid workspace, we are back to the “normal” situation and one can start to use Mantid reductions methods on the dataset as if the data were loaded from a file on disk. The fetching of data from the Kafka cluster is updated at a given fixed rate; e.g. at every 30 seconds. Figure 6 shows two snapshots of a running Mantid Live Data session for the LoKI instrument using the SANS\_benchmark2.com model no. 5 event data. This demonstrates that the Mantid client in the data pipeline is capable of fetching the event data from the Kafka cluster and subsequently perform the data reduction workflow.

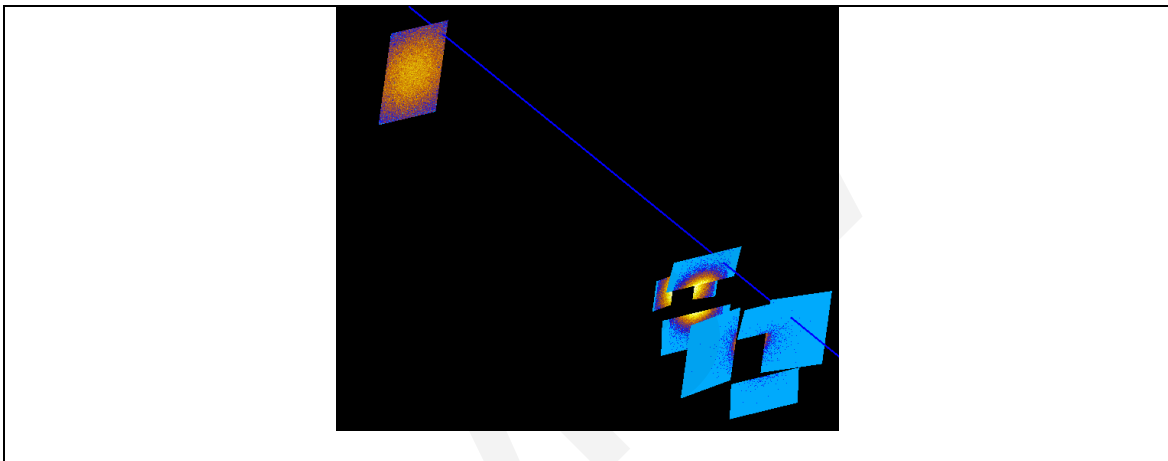


**Figure 6.** The Mantid GUI showing the reduced data,  $I(q)$  for different times after the data stream has started. Left: just after the data stream has started (+60 seconds). Right: all events have been received by Mantid (+306 seconds.) Notice the improvement in the statistics at high  $q$ -values when all events are captured in Mantid (i.e. for longer exposure time).

The event data contains only data from the scattered beam, e.g. effects of background scattering are on purpose not included in this virtual experiment in order to focus on the reduction process.

### 3.2.1. Instrument View

Mantid is also able to auto-update the Instrument View during an experiment and in this way one can see build-up of intensity on the detector panels as more neutrons are coming in. Figure 7 shows the LoKI detector fully illuminated at the end of the run, when all neutron events have reached Mantid.



**Figure 7.** Mantid instrument view of the event data used in Fig. 6. Notice that all 9 detector banks are illuminated.

### 3.2.2. Mantid system performance

As mentioned above, the Mantid workflow can be configured in many ways, which will influence the performance of the workflow. In addition, the workflow performance also depends on the specifics of the data coming in; namely the total neutron rates and the number of detector pixels. The Mantid Live Data method is centred around using a fixed update rate for processing the event data. Thus, the Mantid client has only a fixed time to perform all the operations in the workflow for the data in the first frame, before the second data frame comes in. This is schematically shown in Figure 8, where the CPU usage of two subsequent data frames in Mantid are illustrated as a function of time. If the compute resources are fast enough to process all events within the update time frame, the CPU usage should at the end of the time frame decrease (to a minimum), before the next data data frame needs to be processed.

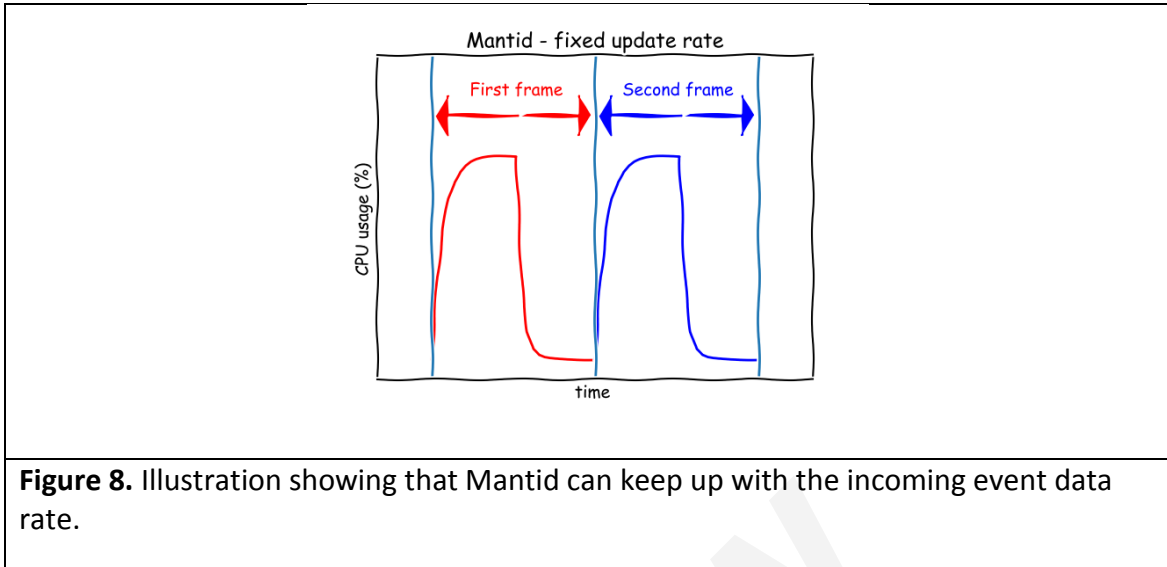
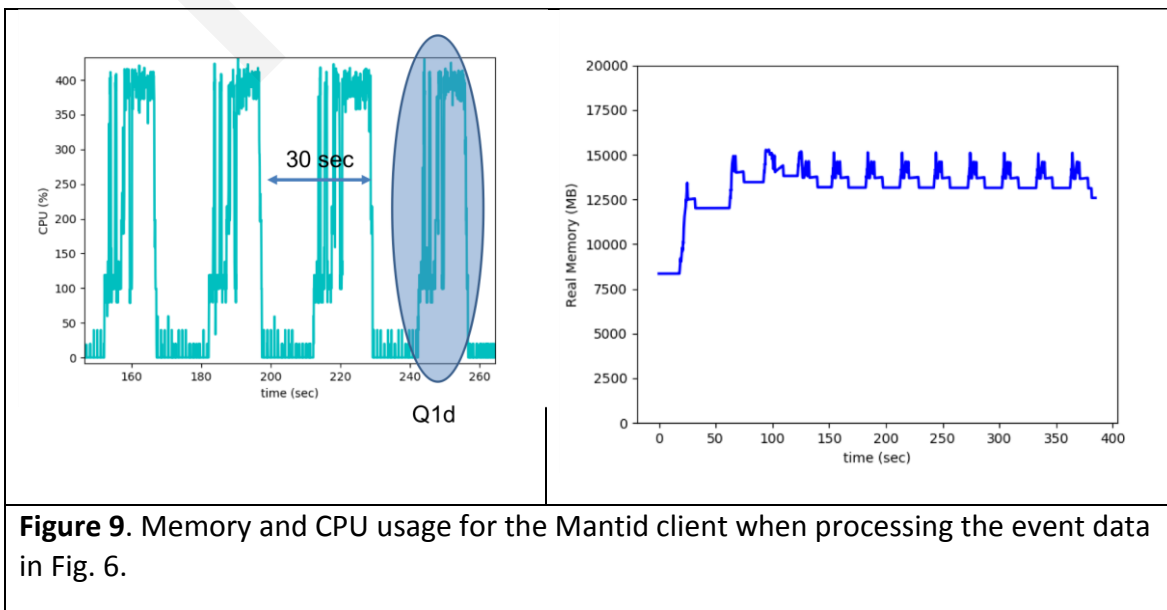
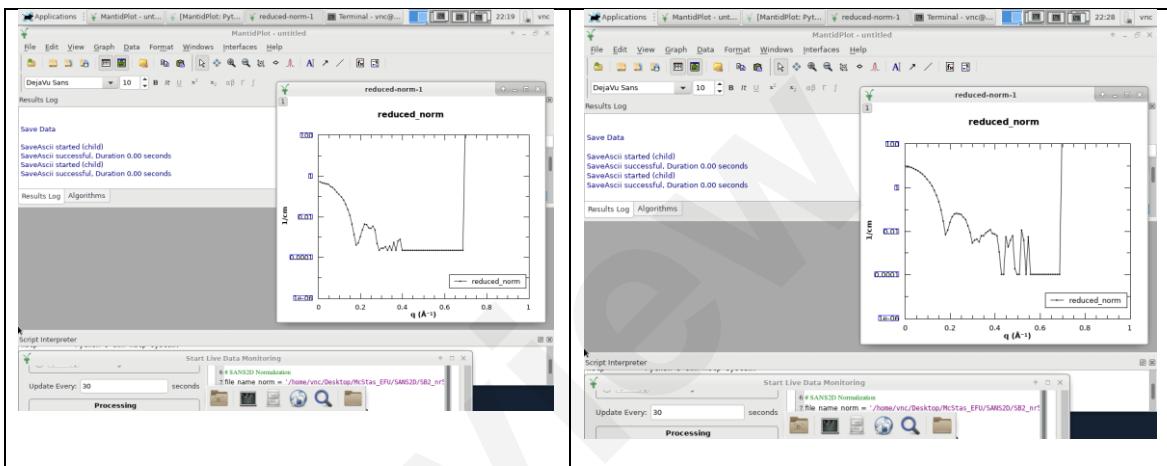


Figure 9 shows the CPU and memory usage used for the Mantid client processing the event data shown in Figure 6, with the update time set to 30 seconds. The Mantid VM for this test has 16 GB memory and 4 CPU cores. For the workflow settings used here, we notice that Mantid is running on all 4 cores for a short time and then is running idle for some time. We also notice that the CPU and memory performance is (almost) reproduced for every 30 seconds. Another way of seeing this is to monitor the output log from Mantid. Here the timestamp and time difference between two identical calls to the same method is no longer than the predefined update rate, say e.g. 30 seconds. Thus, Mantid can keep up with the event rate. Since we are using histogram workspaces, we also see the memory usage is more or less constant during the entire data stream.



### 3.3. Data set 3 - SANS2D Low count rate

For comparison we show here the results of using the same McStas scattering kernel, SANS\_benchmark2.comp no. 5, for the SANS2D instrument. The full McStas instrument description is based on previously developed instrument models for [SANS2D](#). For simplicity only, a single detector bank is included in the simulations, with a 7.8mm x 7.8mm detector resolution (128x128 detector bins), giving a total of 16384 detector pixels. The total detected neutron count rate for this particular setup is > 1.000 #/sec. A corresponding event list were generated with a total count time of 1126 seconds. Figure 10 shows the reduced data for two snapshots while the data is being streamed.



**Figure 10.** SANS2D: The Mantid GUI showing the reduced data,  $I(q)$  for different times after the data stream has started. Left: just after the data stream has started (+60 seconds). Right: all events have been received by Mantid (+1100 seconds.)

Compared to Fig. 6, we observe that the scattering curve  $I(q)$  is less well resolved even though the data collection time is longer for the SANS2D event list. This is of course related to the higher flux impinging in the sample at the LoKI beam line. The difference in the  $q$ -coverage range is also to be expected two instrument files were not configured to show this, e.g. sample-to-detector distance differ.

## 4. HLM-3: INTEGRATION OF ANALYSIS & REDUCTION SOFTWARE

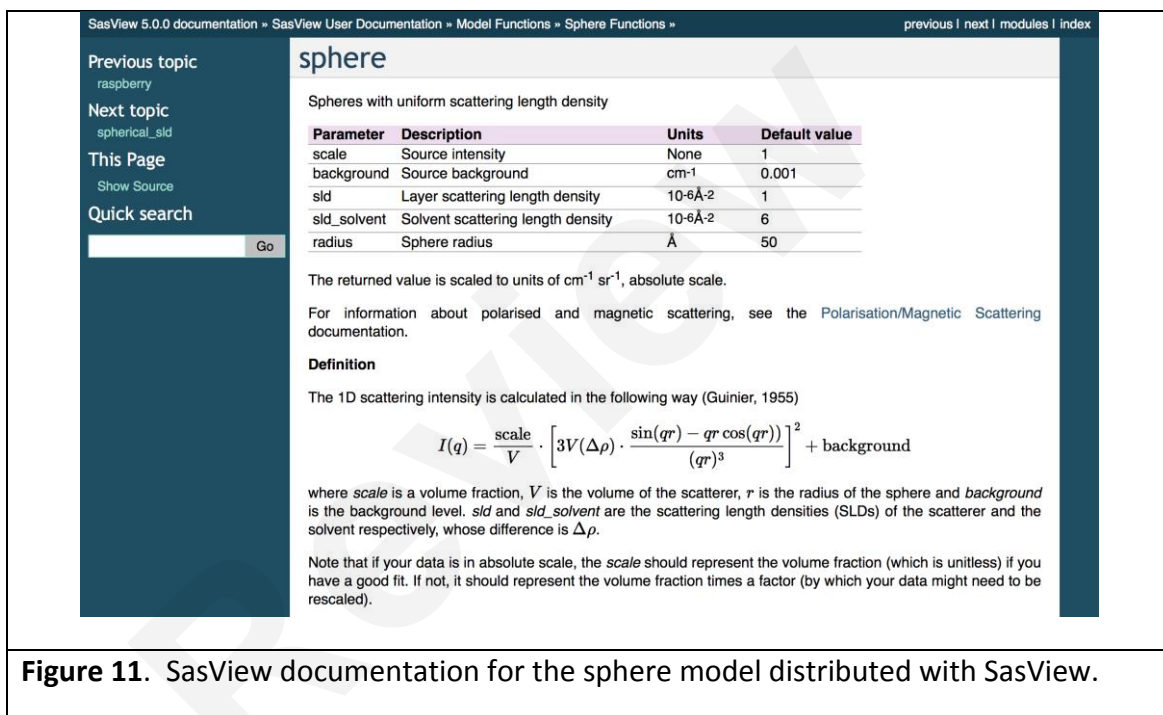
*Data can be reduced and analyzed on the fly for limited flux for instrument(s) chosen for acceptance test*

This milestone is a feasibility study designed to show that a limited neutron flux on the detectors can be pushed through the different live data services constituting the entire data pipe-line from detected neutrons to analyzed data. For this milestone we will use data set 1 and 3 defined in Section 2.1.

## 4.1. Data Analysis

The McStas scattering kernel SANS\_benchmark2.comp model no. 5 mimics scattering from a perfect sample consisting of nano-spheres with a radius of 25 Å. This sample was chosen as it gives a relatively low intensity on detector and since the scattering curve  $I(q)$  is varying slowly wrt. the scattering vector  $q$ .

For the live data analysis, we will use SasView to fit the Mantid reduced data to the sphere model. Figure 11, shows a screen shot of the SasView documentation for the Sphere model. We will fit the Mantid reduced data to the SasView sphere model using three parameters: radius, scale, and background.



The data analysis is for simplicity performed on the same client where Mantid is running. The Mantid generated reduced data file,  $I(q)$ , is read from disk each time Mantid has processed a new data frame. In this way on can monitor results of the virtual experiment as data are being streamed through the system.

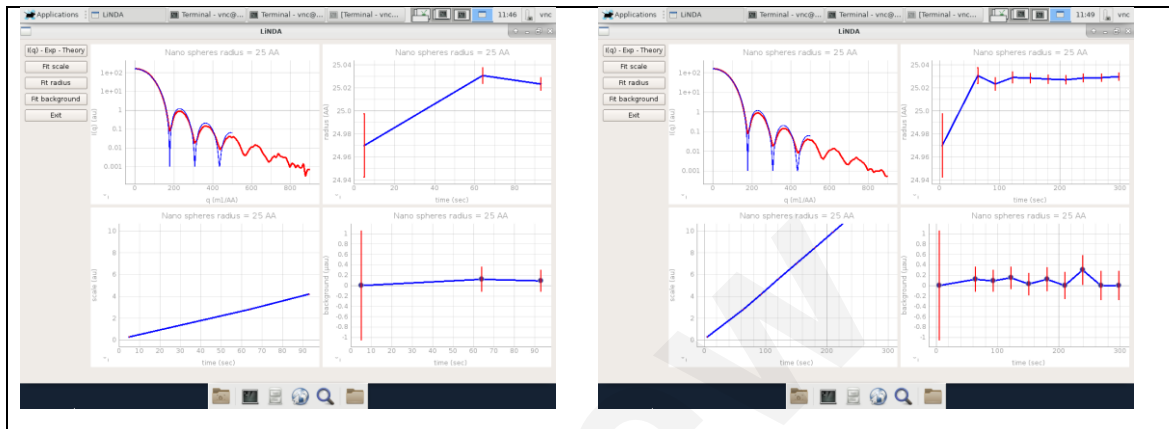
Initially, the live data analysis was wrapped in a PyQtGraph widget as shown here in Figure 12 and 13 below. Later, as shown in Figure 18, Section 7.1, the data analysis was run from a Jupyter Notebook which opens up for a more interactive user experience.

## 4.2. Data set 1 - LoKI Low count rate

The results of the live data analysis are shown in Fig. 12 below. The four sub-plots show  $I(q)$  from Mantid and the theoretical prediction, radius, scale and background parameters. Here we use the DREAM fitting method from SasView to analyse the reduced data. The results of the fitted parameters and the associated error bars are plotted as a function of time. The figure to the left shows the results shortly after event stream has



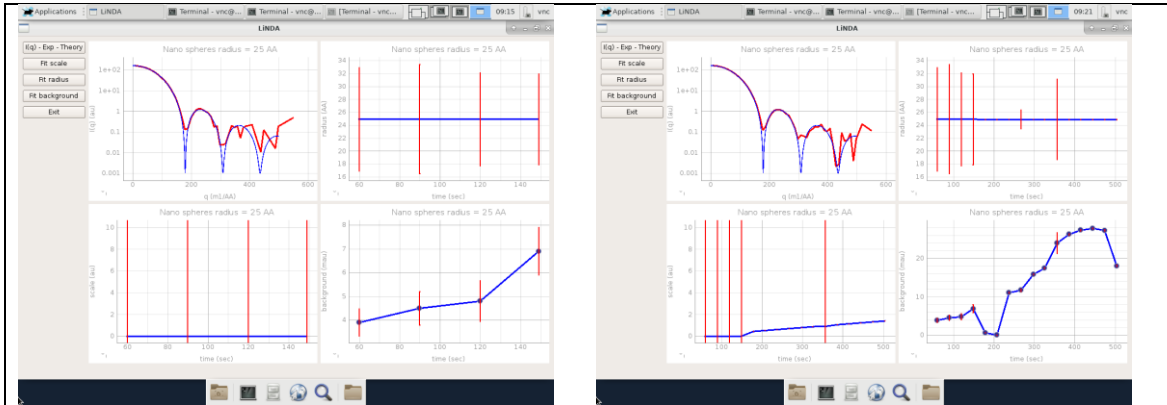
started, while the figure to the right shows the result at the end of the data stream. We observe that as more neutrons are coming in the fitted parameter for the radius converge towards the theoretical value of 25Å used in the McStas scattering kernel, being only a few % off. We notice that the error bars are also converging. The scale parameter is expected to grow linear with time with the number of acquired events. The background parameter is expected to be zero.



**Figure 12.** LoKI: Live Analysis GUI showing the reduced data,  $I(q)$  and fit parameters for different times after the data stream has started. Left: just after the data stream has started (+60 seconds). Right: all events have been received and reduced by Mantid (+306 seconds.)

### 4.3. Data set 3 – SANS2D\_LOW count rate

For comparison reasons we also show the live analysis for the SANS\_benchmark2.comp no. 5 model used for a virtual SANS2D experiment, see Fig. 13. Again, we use the DREAM fitting method from SasView to analyse the reduced data. As before, the results of the fitted parameters and the associated error bars are plotted as a function of time. The figure to the left shows the results shortly after event stream has started, while the figure to the right shows the result at the middle of the data stream. Compared to the similar LoKI setup used in Fig. 12 we notice that the reduced scattering curve is less well resolved for the virtual SANS2D experiment. We also notice the error bars for the fitted parameters are larger for the SANS2D case. This is to be expected as the neutron rate is lower for the SAND2D beamline.



**Figure 13.** SANS2D: Live analysis GUI showing the reduced data,  $I(q)$  and fit parameters for different times after the data stream has started. Left: after a few data frames have been processed by Mantid (+150 seconds). Right: roughly half of all events have been received and reduced by Mantid (+500 seconds.)

## 5. HLM-4: SOFTWARE PERFORMANCE AND INTEGRATION TESTING VALIDATED

1. Data can be reduced and analyzed on the fly for expected flux of  $\langle \text{INSTR} \rangle$  when in full operation
2. Post batch processing can be performed on CPH cluster

This milestone is related to showing that the data pipe-line can handle and process the high data rate coming of the LoKI beam line. As mentioned earlier, the potential bottleneck is foreseen to be the Mantid data reduction part. A detailed study of how Mantid performs for the LoKI high-count rates scenario is done by in-kind partners at ISIS/STFC. Complementary to this, the DRAM group has also studied the high-count rate limit and tested it, making sure that the new features developed at ISIS will also work at the DMSC infrastructure (together with data analysis for this; see Section 6).

### 5.1. Data can be reduced and analyzed on the fly for expected flux of $\langle \text{INSTR} \rangle$ when in full operation

#### 5.1.1. In-Kind contribution from ISIS

STFC in-kind partners has written a thorough report on [LOKI Live Reduction Performance in Mantid](#) based on a detailed modelling of the LoKI detector coupled to Geant4 simulations. The in-kind report has been uploaded to CHES; see CHES document no. ESS-0060903. [Not there yet. To be done at a later time.]

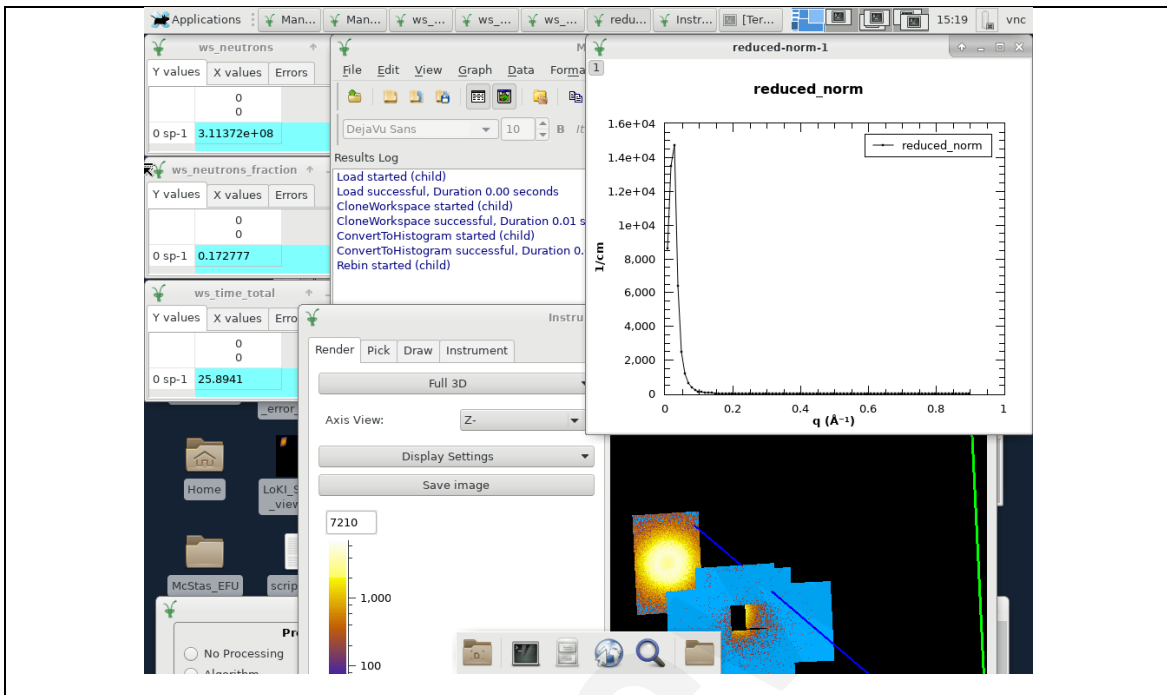
The **conclusion** from the report is: “The final optimizations added to Mantid have facilitated live streaming which will meet expected requirements for coping with projected instrument flux. The scalability of the solution also demonstrates that as processor architectures improve and scale in future iterations, we will be in a good position to handle increasing flux. There are currently no recommended actions required for parallelisation/distribution of Mantid for live streaming.”

The **key feature** for being able to handle the high-count rates is to introduce an internal neutron event buffer in Mantid where the incoming events are stored and later sorted, before being send to an EventWorkspace; as explained in the report: “Instead of received events (from the Kafka stream) being immediately inserted into an EventWorkspace they are instead stored in a buffer held by the decoder. When this buffer is full (where full is a number of events set by the user) it is sorted to place events from the same detector in the same period adjacent to one another. This locality is then used to allow those events to be inserted into the EventWorkspace in parallel.”

### 5.1.2. Work done by DMSC

The McStas simulation for the “high” count rate regime is based on the same setup used for HLM-1 (see Section 3.2) but with the scattering kernel SANS\_benchmark2.comp (model no. 3). With this setup, the detected total neutron count rate is  $> 1.2E7$  #/sec. A corresponding event list were generated with a total count time of 150 seconds. This scenario for the virtual experiment does require more compute power and for this reason we have deployed the live data pipeline on the DMSC cluster network with a single dedicated cluster node for the Mantid reduction client. Here the Mantid client is running on bare metal, while the rest of the clients are virtualized. The event data is stored on group data and thus data are streamed over the network

Figure 14 below shows a screenshot of the Mantid client after a few 10’s of seconds event data being streamed. The setup for Mantid is done in the same way as described in Section 3.2.



**Figure 14.** Mantid GUI after approximately 30 seconds of data being streamed. Upper right part shows the reduced scattering curve  $I(q)$ , while the Mantid instrument View is shown below.

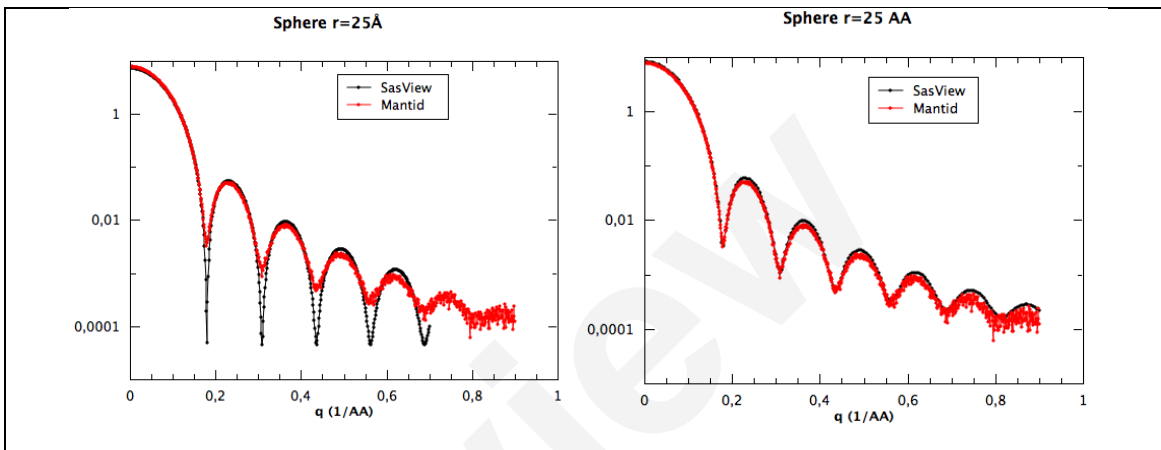
One major outcome of the in-kind work done by STFC, see section 5.1.1, was to introduce a buffer for the events, and the associated BufferThreshold. As a practical consequence of this there are two guidelines that must be fulfilled:

- The number of cores in use for parallel OpenMP methods should not exceed 80% of the total number of core available on the Mantid client, in order not to exhaust the system and leave enough CPU power to the rest of the Mantid GUI application.
- The buffer threshold value needs to be adjusted manually to give the best performance, and one needs to try out a few threshold values to find the optimum value. As a first guess the buffer threshold value should approximately equal the expected count-rate times the update time interval.

A note for the local experts: Due to the dual setup for the DMSC compute facilities at the end of 2019; the event data do a small round trip, when we are testing the data pipeline. The NexusFileWriter, located at DMSC, reads data from storage at HCØ, then sends the data to Kafka at DMCS, and then finally the Mantid cluster node at HCØ reads the event data from the Kafka cluster. Thus, data travels from HCØ-> DMSC -> HCØ.

## 5.2. Post batch processing can be performed on CPH cluster

The event data from data set no. 1 and no. 2 are designed to be used with the NeXusFileStreamer, and cannot directly load into Mantid. By restreaming the event data into Mantid and keeping them in event format one can save the captured data as an event list to be used for later processing. Then using a python script, the event data can be post-processed by one Mantid cluster node. Figure 15 shows the result of a post reduction, alongside two fits using SasView with and without poly dispersion (radius  $pd=0.03$ ) for the data set no. 1.



**Figure 15.** Scattering intensity as a function scattering vector  $q$ ;  $I(q)$ . Post-data reduction of the data set no. 1, using the Mantid client. Without (left) and with poly dispersion (right). The black curves are from SasView, while the red curves are from Mantid.

## 6. HLM-5: PROOF OF OPERATION (REDUCTION)

1. Data can be reduced on the fly for expected flux of  $\langle INSTR \rangle$  when in full operation. Data are received from DAQ & Streaming system
2. Data can be post re-reduced on cluster
3. Need to be able to define the scaling parameter for the architecture rather than actually physically demonstrate

### 6.1. Data are received from DAQ & Streaming system

This part is covered by the work done at the HZB V20 beamline by DMSC staff during 2018/2019. It is not considered to be a part of this report. See documentation elsewhere on [Confluence](#), e.g.: [Update on 2019 V20 experiments](#).

## 6.2. Data can be post re-reduced on cluster

The post data reduction described in section 5.2 were done on a single cluster node, having 32 cores and 128 GB memory. To go beyond this kind of number of cores and memory used, would require even more cores and memory squished into a single HPC node. Another more standard HPC approach would be to use an MPI based version of Mantid.

As a proof of data reduction using MPI, we have set up a two-node cluster system with Mantid compiled to use [MPI](#). SLURM was used as the job scheduler. Figure 16 below shows the output of an MPI run reduction script, demonstrating the we can also use an MPI based version of Mantid if needed. The script is based on the SANS2D data set SANS\_test.nxs which is distributed together with the NexusFileStremer. The reduction workflow is on purpose designed to show just the basic features of a Rebin, SumSpectra and then SaveAscii. Then by comparing the total number of neutrons listed in the saved ascii file to the total number of neutrons listed in the NeXus files, we observe that they match.

```
>> sbatch mantidsbatch.sh
```

```
>> more mantid.err
```

```
FrameworkManager-[Notice] Welcome to Mantid 4.1.20191106.2323
```

```
FrameworkManager-[Notice] Please cite: http://dx.doi.org/10.1016/j.nima.2014.07.029 and this release:  
http://dx.doi.org/10.5286/Software/Mantid
```

```
Load-[Notice] Load started
```

```
Load-[Notice] Load successful, Duration 5.26 seconds
```

```
Rebin-[Notice] Rebin started
```

```
Rebin-[Notice] Rebin successful, Duration 0.12 seconds
```

```
SumSpectra-[Notice] SumSpectra started
```

```
SumSpectra-[Notice] SumSpectra successful, Duration 0.38 seconds
```

```
SaveAscii-[Notice] SaveAscii started
```

```
SaveAscii-[Notice] SaveAscii successful, Duration 0.31 seconds
```

**Figure 16.** MPI data reduction of the test data set SANS\_test.nxs using two cluster nodes. The job was submitted with the SLURM scheduler. (N.B. the Mantid output was dumped in the error file mantid.err. There are however no errors in the run here, it is just related to the way std. in/out is handled.)

### 6.3. Define the scaling parameter for the architecture

The scaling aspects were addressed in the report, [LOKI Live Reduction Performance in Mantid](#), by the STFC in-kind partner, see section 5.1.1. They conclude:

*“The scalability of the solution also demonstrates that as processor architectures improve and scale in future iterations, we will be in a good position to handle increasing flux. There are currently no recommended actions required for parallelisation/distribution of Mantid for live streaming.”*

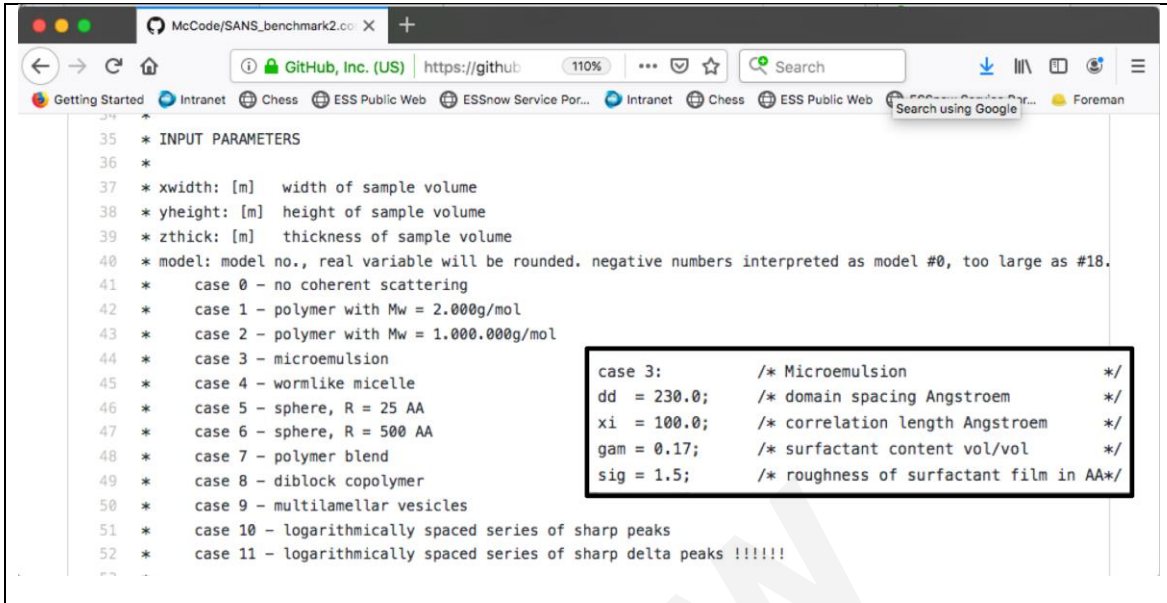
## 7. HLM-6: PROOF OF OPERATION (ANALYSIS)

1. *Data can be analyzed on the fly for expected flux of <INSTR> when in full operation. Data received from Mantid, which in turn data from DAQ & Streaming system*
2. *Post batch processing can be performed on cluster*

### 7.1. Data can be analyzed on the fly for flux

The McStas scattering kernel SANS\_benchmark2.comp model no. 3 mimics scattering from a microemulsion sample. This sample was chosen as it gives a high intensity on the detector.

For the live data analysis, we will use SasView to fit the Mantid reduced data by use of a python custom made model of the SANS\_benchmark2.comp model no. 3 sample. Figure 17 shows a screenshot of the McStas documentation for the SANS\_benchmark2.comp. In the following we will fit the Mantid reduced data to the microemulsion model using three parameters: domain spacing, scale, and background.



```
35 * INPUT PARAMETERS
36 *
37 * xwidth: [m] width of sample volume
38 * yheight: [m] height of sample volume
39 * zthick: [m] thickness of sample volume
40 * model: model no., real variable will be rounded. negative numbers interpreted as model #0, too large as #18.
41 * case 0 - no coherent scattering
42 * case 1 - polymer with Mw = 2.000g/mol
43 * case 2 - polymer with Mw = 1.000.000g/mol
44 * case 3 - microemulsion
45 * case 4 - wormlike micelle
46 * case 5 - sphere, R = 25 AA
47 * case 6 - sphere, R = 500 AA
48 * case 7 - polymer blend
49 * case 8 - diblock copolymer
50 * case 9 - multilamellar vesicles
51 * case 10 - logarithmically spaced series of sharp peaks
52 * case 11 - logarithmically spaced series of sharp delta peaks !!!!!
```

```
case 3:      /* Microemulsion      */
dd = 230.0;  /* domain spacing Angstroem    */
xi = 100.0;  /* correlation length Angstroem */
gam = 0.17;  /* surfactant content vol/vol   */
sig = 1.5;   /* roughness of surfactant film in AA*/
```

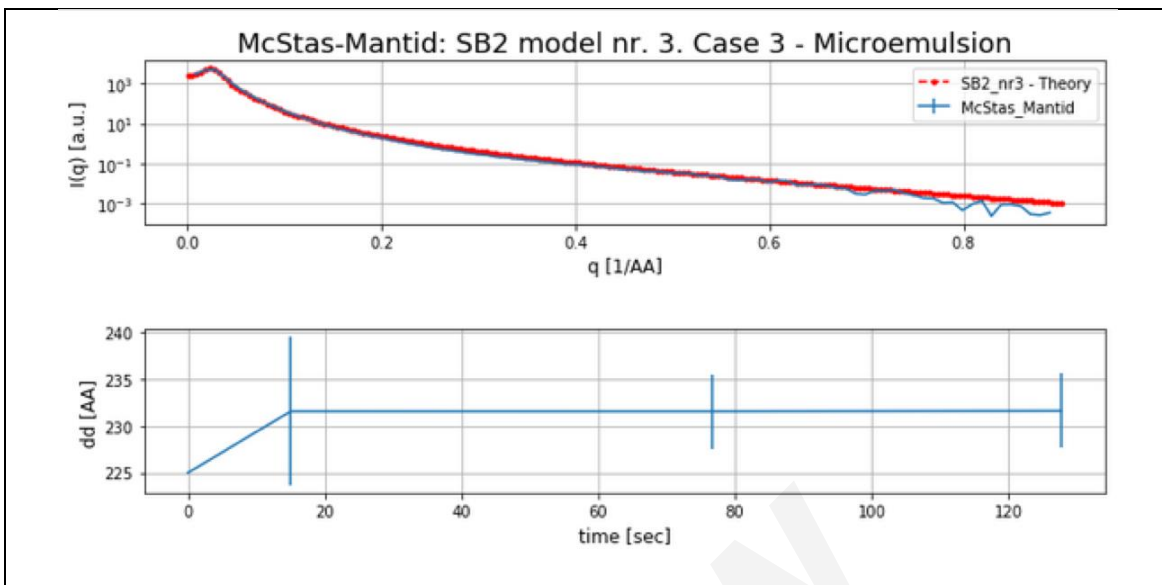
**Figure 17.** SANS\_benchmark2.comp documentation. Here we will use model no. 3.

The data analysis is for simplicity performed on the Mantid client, where the Mantid reduced data file,  $I(q)$ , is read from disk each time Mantid has processed a new data frame. In this way one can monitor results of the virtual experiment as data are being streamed through the system.

For the fitting we use a Conda installation of SasView and a Jupyter notebook which opens up for a more interactive user experience. As usual we use the DREAM fitting method from SasView to analyse the reduced data. The details of the Jupyter notebook setup can be found in Section 8.

The results of the live data reduction and analysis are shown in Fig. 18 below. The top figure shows the reduced data with error bars as it comes from Mantid (blue curve), while the red curve is the result from the SasView model using the latest model parameters from the fit. The bottom figure shows the results of the fitted parameter domain spacing,  $dd$ , as a function of time. Due to the high neutron flux on the detector and the simplicity of the model the fit converges quite fast. The fitted parameter for the domain spacing converges towards the theoretical value of  $230\text{\AA}$  used in the McStas scattering kernel, being only a few % off.



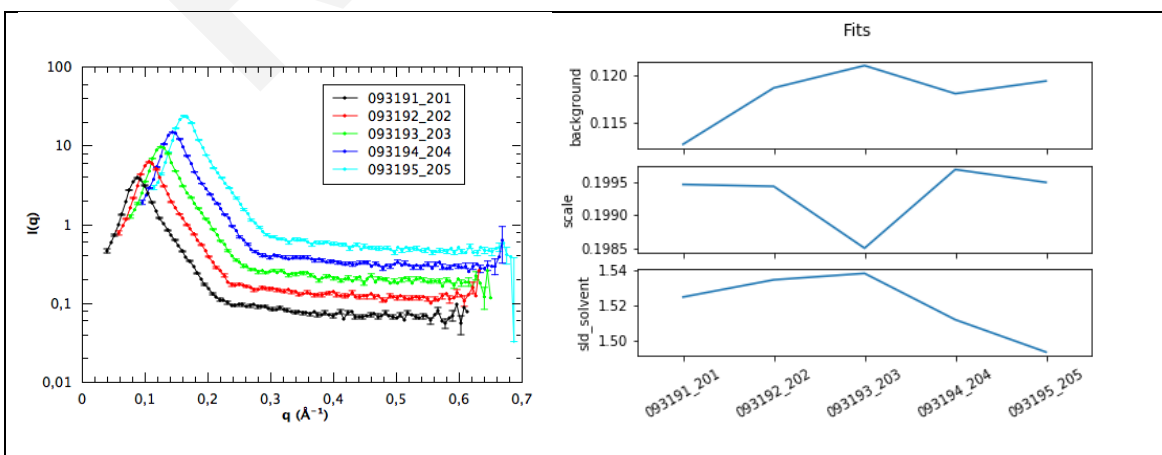


**Figure 18.** Live data reduction and fit of virtual LoKI SANS\_benchmark2.comp no. 3. experiment.

## 7.2. Post batch processing can be performed on cluster

The primary use case for the live data analysis setup is considered to be used for on-the-fly sanity checks of the data coming in. For more advanced model fitting, it is advantageous to be able to submit fitting jobs to a compute cluster.

Batch fitting of a series of scattering curves can be performed on the DMSC cluster. See Section 8 for details. Here a series of recorded scattering curves are fitted to the same model, see Figure 19.



**Figure 19.** Waterfall plot of a series of scattering curves (left). All data files are fitted to the same model, ellipsoid.py, and the fitted parameters (right).

Using the time series obtained for the scattering curve in the Live Data reduction, one could imagine a use case for time-resolved SANS studies where the time dependent scattering curves could be analysed doing batch fitting on a cluster by fitting all data sets to a single model. That is, any change of model parameters can then be identified from the outcome of the fit results.

## 8. FINAL COMMENTS

Based on the finding in this report we have some general comments and recommendations for further work, related to the question of “Do we see any bottlenecks in the current design for live data.”

**Live Data:** Live Data should be thought of as data per time frame or time interval. Obviously, the limiting factor comes down to how many computations can be performed in a given time interval? Given a fixed set of hardware specifications there is an upper limit to the number of CPU cycles that can be handled in a given time span. Again, one needs to think carefully about the data reduction workflow and not waste CPU cycles.

**Mantid:** In the case of Mantid, the requirements for dealing with the high-count rates depends on many parameters. E.g. the number of detector pixels, how events are stored in Mantid (event or histogram), the update rate, the numbers of bins in e.g. TOF, wavelength  $\lambda$ , momentum transfer  $q$  and of how many workspaces are typically used in a workflow. Using inputs from instrument scientists and users we can make an educated guess with respect to, the hardware needed for running a live data pipeline (data reduction workflow). But, there is no “one size fits all”.

**DMSC compute resources:** The current tests for the high-count rates has stretched the DMSC compute resources, and more resources are needed as we work our way through the rest of the ESS instrument suite. This goes for compute resources with high-memory, many cores, fast network connections and large file storage with high read/write access. The LDPC Puppet setup can deploy all the services needed for live data on either the DMSC virtualization hosts or on the DMSC cluster system. If we e.g. need a “beefy” Mantid client with much memory and many cores we can by choice deploy on one of the cluster nodes, but it does require the hardware for doing so.

**Securing knowledge transfer:** The current studies have shown that a detailed instrument modelling of the beamlines is essential for finding “blind spots” in the virtual data pipeline, as well as for securing knowledge transfer between the many stakeholder of a particular beamline.

**Optimize reduction and analysis workflows:** Equally important is the need for optimizing Mantid workflows, and test the usability of the reduction methods for the expected ESS data rates and detector layouts. Finally, for data analysis it is important to strike a balance between being able to fit data within a given time frame (the live data aspect) using a model that is just detailed enough to capture the essence of the experiments; a full fledged detailed model would most likely require fitting times longer than the update

data frame time. Using analysis tools with a clearly defined CLI have proven to make it much easier to setup the data analysis.

**Instrument resolution functions:** Although not part of the milestones, the work has shown that the effects of the instrument resolution must be included in the data analysis and that the ESS specific resolution functions are not well described.

## 9. ACKNOWLEDGEMENT

The following DMSC and ISIS in-kind partners have contributed to the work summarized in this report:

ESS-DMSC-ECDC:	Afonso Mukai
ESS-DMSC-DST:	Kareem Galal and Lottie Greenwood
ESS-DMSC-DRAM:	Peter Willendrup, Wojciech Potrzebowski, Simon Heybrock, and Torben Nielsen
STFC-ISIS:	Daniel Nixon, Lamar Moore, Michael Hart, Oven Arnold, and Mathew D. Jones.

## 10. GLOSSARY

Term	Definition
HLM	High-Level-Milestones
DRAM	Data Reduction, Analysis and Modelling
LDPC	Live-Data-Processing-Coordination
GUI	Graphical User Interface
TOF	Time-of-flight
CLI	Command Line Interface

## 11. REFERENCES

- [1] Link to STFC document which is now already on [github](#). Should also be in CHESS.

## 12. DOCUMENT REVISION HISTORY

Revision	Reason for and description of change	Author	Date
1	First issue	Torben Nielsen	2019-12-19
2	Internal DRAM review	Torben Nielsen	2020-01-13

## 13. APPENDIX: SOME PRACTICAL DETAILS

This appendix shows how to use the data sets this report is based upon. This is intended as a background information to give a short look-and-feel for the data pipeline, without having the reader required to log-on to the DMSC infrastructure and perform all the steps needed to push data through the system.

The test data and scripts for running the live data stream, data reduction and analysis used in this report, can be obtained from a DMSC [gitlab](#) repository. Using these data sets and scripts and the [Quick start guide](#), one should be able to test the virtual data pipeline. For data set 1 and 2 there are described two ways for running the Mantid data reduction, whereas all the data analysis is done exclusively by using Jupyter notebooks. The Mantid data reduction can be run in the standard Mantid desktop application (GUI mode). In this user mode a few manual configuration steps are needed before Mantid can hook up to the data stream. In this mode we have full access to all the methods of Mantid; e.g. to show the build-up of intensity on the detectors by the method "ShowInstrument". One can also run Mantid headless by using the "mantidpython" to execute a python script.

### 13.1. LoKI – Dataset 1

The data set no. 1 is used in Section 3.2 and 4.2. Compared to the data analysed in Section 4.2, Figure 22 below shows the same data set and time dependent fitting, now using a Jupyter notebook. The Mantid data reduction is run headless by using Python scripts. First the a Jupyter notebook server is started and the notebook file Fit.ipynb is executed (see Figure 20), then the NeXusFileStreamer (see Figure 21) is started and finally the Mantid python script is executed (see Figure 22).

```
>> jsmith@mantid01:~$ jupyter notebook
```

**Figure 20.** Start the Jupyter notebook from a terminal.

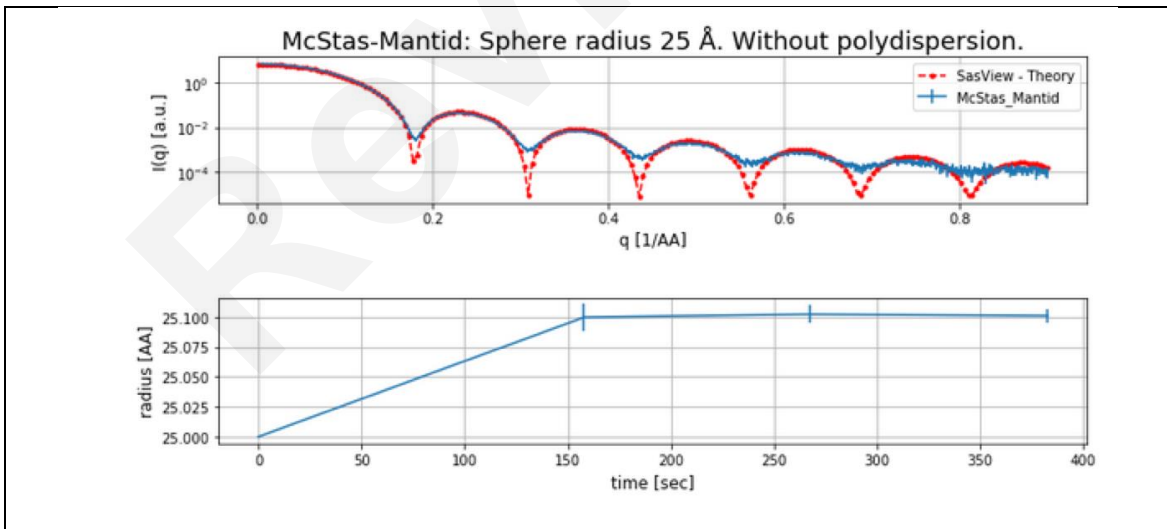
```
>> jsmith@nexusfilestreamer01:~$ cd /opt/dm_group/NeXus-Streamer/bin  
>> jsmith@nexusfilestreamer01:~$ LD_LIBRARY_PATH=./lib ./nexus-streamer -f /opt/dm_group/NeXus-Streamer/data/SANS_test.nxs -d /opt/dm_group/NeXus-Streamer/data/spectrum_gastubes_01.dat -b 1 92.168.20.124:9092 -i ISIS_Kafka_Event -z
```

**Figure 21.** Start the NeXusFileStreamer from a terminal. This example shows how to stream the event list “SANS\_test.nxs”. For streaming the data set no. 1, 2, and 3 the paths to the event list and spectrum file must be adjusted accordingly.

```
>> jsmith@mantid01:~$ mantidpython run_live.py
```

**Figure 22.** Start the Mantid data reduction from a terminal

At the end of the data stream we see the results for the fitted radius as a function of time, see Fig. 23:

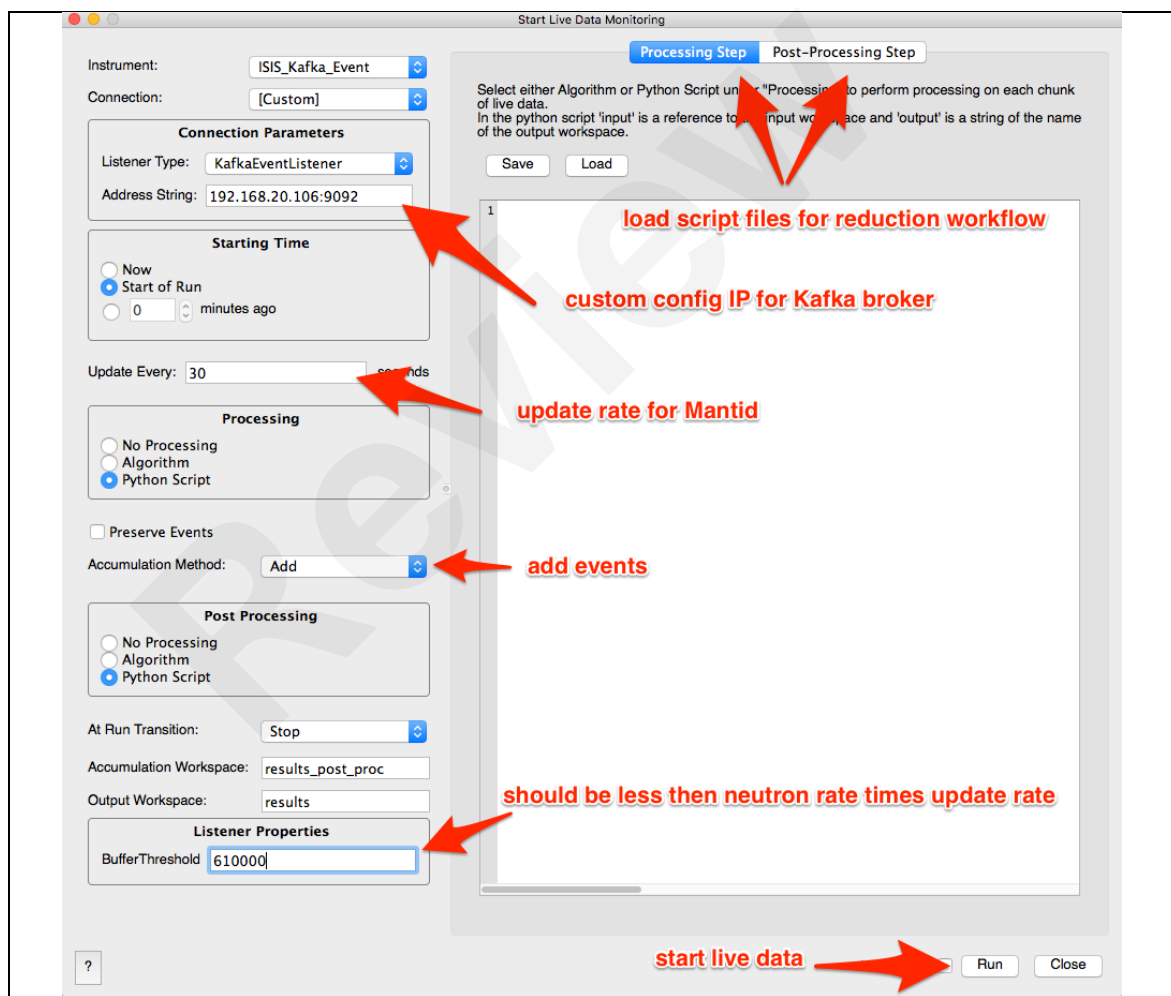


**Figure 23.** Jupyter notebook results for the live data analysis of data set no. 1, as done in Section 4.2

The repository holds all data sets needed to test the data pipeline using data set no. 1: event list, IDF, spectrum mapping, normalisation spectrum, Mantid scripts and SasView scripts.

## 13.2. LoKI – Dataset 2

Figure 23 shows how to configure the Mantid GUI for Live Data. In the main Mantid GUI widget press “Load” and select “Live Data”, then the widget shown in Figure 24 appears. The left panel is related to how-to collect data from Kafka, while the right side of the panel is related to the data reduction workflow. For both the processing and the post-processing tab a python script can be loaded. Then as described in Section 13.1 the Jupyter notebook is first started, then the NexusFileStreamer, and finally the Mantid Live Data is started by the “Run” button, see Figure 24. The results of the fitting by the Jupyter notebook is shown in Figure 18.



**Figure 24.** Mantid widget for configuring and running the Live Data.

The repository holds most of the data sets needed to test the data pipeline using data set no. 1: IDF, spectrum mapping, normalisation spectrum, Mantid scripts and SasView scripts. The event list is only stored on DMSC hardware, as it is more 14 GB.

### 13.3. Batch fitting on DMCS cluster

SasView supports batch fitting, using either the GUI or by a python script. An example of how-to use the python CLI for batch fitting is shown on the SasView [repository](#). Using the DMCS HPC job scheduler SLURM, fitting jobs can be submitted to the HPC compute resources. Figure 25 and 26 show how to run the batch fitting using a python script. The python script can in turn be submitted to the HPC cluster using the SLURM job scheduler.

```
>> jsmith@mantid01:~$ source activate sasview_env
```

**Figure 25.** How to run batch fitting from CLI. Activate the Conda python environment for SasView.

```
>> jsmith@mantid01:~$ (sasview_env) python test.py
```

**Figure 25.** How to run batch fitting from CLI. In the Conda environment “sasview\_env” run the python script “test.py”.

The current implementation is based on a Conda installation of SasView. According to SasView documentation for the [fit engine being used](#), a custom build version of the fit engine tailored to the HPC cluster in use, could add additional speed-up for the fitting.