ODIN - A CONTROL AND DATA ACQUISITION FRAMEWORK FOR EXCALIBUR 1M AND 3M DETECTORS

G. Yendell, U. Pedersen, N. Tartoni, S. Williams, Diamond Light Source, Oxfordshire, UK
T. Nicholls, STFC/RAL, Chilton, Didcot, Oxon, UK
A. Greer, OSL, Cambridge UK

Abstract

Detectors currently being commissioned at Diamond Light Source (DLS) bring the need for more sophisticated control and data acquisition software. The Excalibur 1M and 3M are modular detectors comprised of rows of identical stripes. The Odin framework emulates this architecture by operating multiple file writers on different server nodes, managed by a central controller. The low-level control and communication is implemented in a vendor supplied C library with a set of C-Python bindings, providing a fast and robust API to control the detector nodes, alongside a simple interface to interact with the file writer instances over ZeroMQ. The file writer is a C++ module that uses plugins to interpret the raw data and provide the format to write to file, allowing it to be used with other detectors such as Percival and Eiger. At DLS we implement an areaDetector driver to integrate Odin with the beamline EPICS control system. However, because Odin provides a simple HTTP Rest API, it can be used by any site control system. This paper presents the architecture and design of the Odin framework and illustrates its usage as a controller of complex, modular detector systems.

INTRODUCTION

Diamond Light Source (DLS) are currently developing data acquisition and control software for several modular, high-performance detectors. Excalibur [1] is the result of a collaboration between DLS and STFC and has been implemented for the X-ray Imaging and Coherence beamline I13 to make use of the small pixel size of the detector in coherence diffraction imaging. The Hard X-ray Nanoprobe beamline I14 has more recently chosen a 3M Excalibur system for nanoscale microscopy. Another collaboration, between DLS, Elettra, the Pohang Light Source and STFC, is ongoing to develop the Percival detector [2] for soft x-ray experiments. At the same time, DLS is exploring commercial options in the Eiger from Dectris. Currently, the VMXi (Versatile Macromolecular Crystallography in-situ) beamline is commissioning the first of these; a 4M Eiger X detector [3]. With a multitude of modular and scalable detector systems in development concurrently, an opportunity arose to develop shared control and data acquisition software stacks to drive the systems, designed from the very beginning to be detector agnostic, but with a set of specific use cases to guide the design process.

EXCALIBUR DETECTORS

Excalibur [1] detectors are made up of identical sensor 'stripes', with 8 Medipix3 readout chips. Each stripe has its own FPGA data acquisition card, known as a front-end module (FEM), with a 10Gbit/s optical link. These stripes are combined into a pair to create what is called a 'module'; a 1M 2048 x 512 pixel sensor. A 3M simply consists of 3 stacked modules producing a 2048 x 1536 sensor. A primary feature of the Excalibur is the small pixel size of the sensors, at 55 um x 55 um. A schematic of the 3M Excalibur DAQ system is shown in Fig. 1. The design follows a generic data acquisition framework for detectors, where the Linux cluster receiving the data simply sees a set of parallel data links. This allows the software supporting the framework a large amount of abstraction, simplifying the architecture. The Excalibur is currently operated at DLS with an EPICS areaDetector driver [4] controlling and acquiring data from each individual FEM, with a top level IOC presenting PVs wired through to the underlying processes. However, this system is limited both in its control flexibility and its data throughput and is intended to be replaced by the Odin software stack, described in this paper.

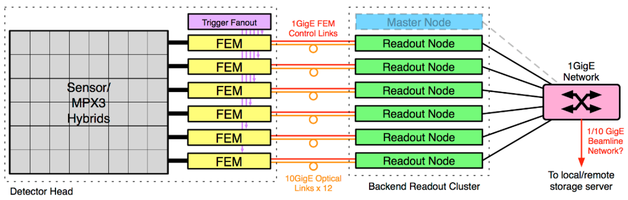


Figure 1: A schematic of the Excalibur 3M system [5].

AN OVERVIEW OF ODIN

Devices consisting of multiple individual parts can lead to complications in the control layer trying to get operate them together in unity. The Odin software framework is designed specifically for this modular architecture by mirroring the structure within its internal processes. The data acquisition modules have the perspective of being one of many nodes built into the core of their logic. This makes it straightforward to operate multiple file writers on different server nodes working together to write a single acquisition to disk, all managed by a single point of control. It also means that the difference in the data acquisition stack of a 1M system and a 3M system can be as little as duplicating a few processes and modifying the configuration of the central controller. Given the collaborative nature of the detector development, the software framework has been designed to generic, allowing its integration with control systems used at different sites.

Odin comprises two parts; OdinControl, a central controller, and OdinData, a data acquisition stack, both of which can be used independently of each other as well as in conjunction. These modules are described, separately, below.

ODIN CONTROL

OdinControl is a HTTP based server host providing a framework that device-specific adapters can be implemented for to interface to the control channel of a device. The architecture of OdinControl for the Excalibur use case is shown in Fig. 2. Adapters can be loaded in an Odin Server instance, which then provides a REST API that can be operated using just a web page providing the appropriate GETs and PUTs, corresponding to the attributes and methods in the adapter API. See Fig. 3 for an example web page for Percival. This can be extended to a RESTful client library that can be integrated into a higher level control system such as EPICS. Once an Odin Server instance is running with a set of adapters loaded, a parameter tree is created in the API defining the different devices, duplicates of the same devices and finally the endpoints for those devices, producing logical paths to the parameters and methods of a collection of separate systems. With OdinControl and a device adapter, a control system agnostic, consistent API is created that can be used in a wide range of applications. OdinControl provides a simple Python API, enabling rapid development of device adapters. Because of the generic architecture of OdinControl it does not need a tight coupling to OdinData; it is also interfaced via adapters, just like Excalibur or Percival, to provide a REST API for a set of methods. This keeps the two parts of Odin entirely separate, achieving a good software design with loose coupling and high cohesion.

ODIN DATA

OdinData gathers incoming frames from a data stream and writes them to disk as quickly as possible. It has a modular architecture making it simple to add functionality and extend its use for new detectors. The function of the software itself is relatively simple, allowing a higher-level supervisory control process to do the complex logic defined by each experimental situation and exchanging simple configuration messages to perform specific operations. This makes it easy for the control system to operate separate systems cooperatively.

OdinData consists of two separate processes. These are the FrameReceiver (FR) and the FrameProcessor (FP). The FR is able to collect data packets on various input channel types, for example UDP and ZeroMQ [6], construct data frames and add some useful meta data to the packet header before passing it on to the FP through a shared memory interface. The FP can then grab the frame, construct data chunks in the correct format and write them to disk. The two separate processes communicate via inter-process communication (IPC) messages over two ZMQ channels. When the FR places a frame into shared memory, it sends a message over the ready channel, the FP consumes the frame and once it is finished passes a message over the release channel allowing the FR to re-use the frame memory. The use and re-use of shared memory reduces the copying of large data blobs and increases data throughput. This logic is shown visually in Fig. 4.

The overall concept is to allow a scalable, parallel data acquisition stack writing data to a individual files in a shared network location. This allows fine tuning of the process nodes for a given detector system, based on the image size and frame rate, to make sure the beamline has the capacity to carry out its experiments and minimise the data acquisition bottleneck.

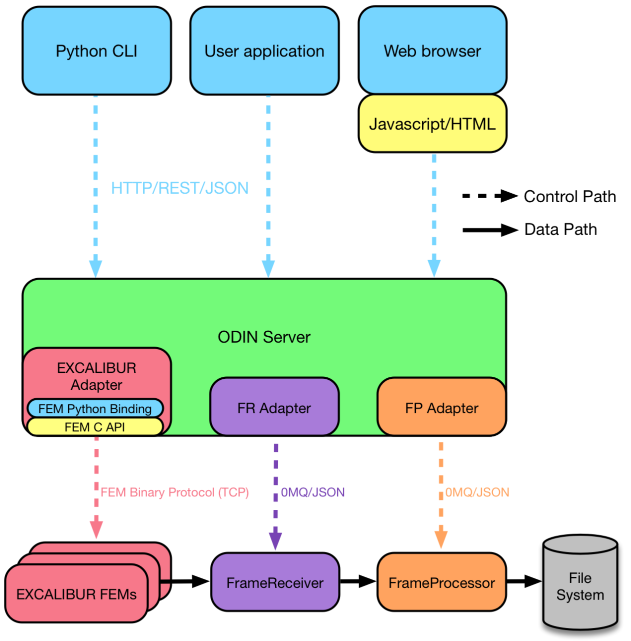
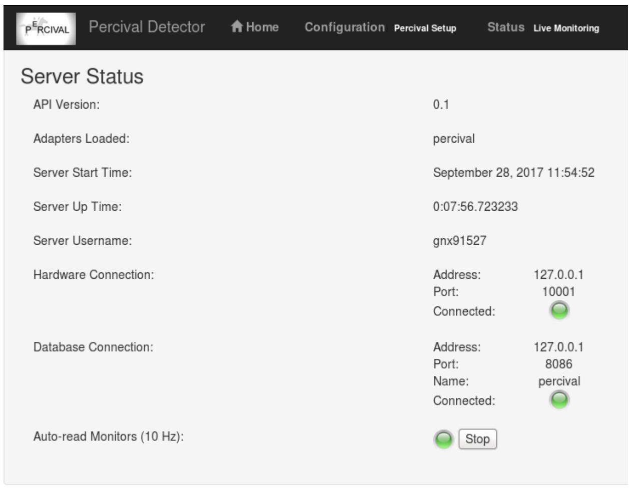


Figure 2: OdinControl architecture showing Excalibur use case.



Server Status

API Version:	0.1
Adapters Loaded:	percival
Server Start Time:	September 28, 2017 11:54:52
Server Up Time:	0:07:56.723233
Server Username:	gnx91527
Hardware Connection:	Address: 127.0.0.1 Port: 10001 Connected: ●
Database Connection:	Address: 127.0.0.1 Port: 8086 Name: percival Connected: ●
Auto-read Monitors (10 Hz):	● Stop

Figure 3: Example OdinControl client webpage for Percival.

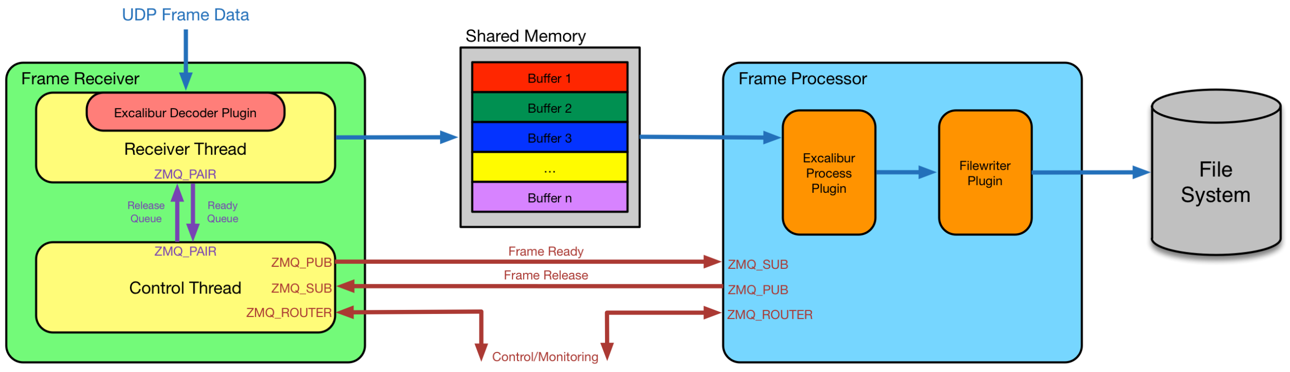


Figure 4: OdinData architecture showing Excalibur use case.

Plugins

OdinData is extensible by the implementation of plugins. The Excalibur detector has a module with two plugins built against the OdinData library. One for the FR and one for the FP. These provide the implementation of a decoder of the raw frame data as well as the processing required to define the data structure written to disk. As an example, the Excalibur plugins implement some algorithms [4] to perform transforms to rotate chunks of the input data, due to the physical orientation of the chips on the detector. The implementation of any other detector would simply require the two plugins to be replaced with equivalents, to process the output data stream; the surrounding logic would remain exactly the same.

API

OdinData provides a python library with simple methods for initialising, configuring and retrieving status from the FP and FR processes at runtime. These can be integrated with a wider control system, but can also be used directly in a simple python script or interactively from a python shell. This is how OdinData integrates with OdinControl; there is no special access granted, the interface is generic allowing it to be integrated with other control systems.

HDF5 Features

To take advantage of the high data rates of modern detectors, OdinData seeks to write data to disk quickly with minimal processing overhead. To achieve this, the built-in FileWriterPlugin employs some of the latest features of the HDF5 library.

The Virtual Dataset (VDS) [7] enables the file writing to be delegated to a number of independent, parallel processes, because the data can all be presented as a single file at the end of an acquisition using VDS to link to the raw datasets. Secondly, with Single Writer Multiple Reader (SWMR) [7] functionality, datasets are readable throughout the acquisition and live processing can be carried out while frames are still being captured, greatly reducing the overall time to produce useful data. Though the real benefit comes when these two features are combined. A VDS can be created anytime before, during or after and acquisition, independent of when the raw datasets and created. Then, as soon as the parallel writers begin writing to each raw file, the data appears in the VDS as if the processes were all writing to the same file and can be accessed by data analysis processes in exactly the same way.

A more straightforward improvement in the form of a data throughput increase is found by the use of Direct Chunk Write [7]. With a little extra effort in the formatting of the data chunk, this allows the writer to skip the processing pipeline that comes with the standard write method and write a chunk straight to disk as provided. This reduces the processing required and limits data copying. For the Eiger use case specifically, great use is made of the Direct Chunk Write to allow writing of pre-compressed images from the detector to file. Due to the considerable data rate of the detector, compression is used to reduce network and file writing load by around a factor of four, depending on the sensor exposure. Reader applications can use Dynamically Loaded Filters [7] to read the datasets.

USE CASES

Odin is now starting to be integrated with beamline and lab detector systems at DLS. The most progress has been made with Eiger and Excalibur. The two implementations detailed below are quite different from each other, showing the flexibility of the software.

Excalibur

Excalibur is the first detector to be fully supported by Odin. It has FR and FP plugins allowing an OdinData process to serve each FEM of the detectors, as well as an adapter for OdinControl to provide a complete HTTP API for controlling the nodes. The latest milestone reached was using OdinData and OdinControl to acquire 10,000 frames from a 1M detector at 100Hz. The Surface and Interface Diffraction beamline I07 will be the first to commission an Excalibur operated with this software stack.

Eiger

The first Eiger detector at DLS is currently being commissioned on the VMXi beamline using OdinData, alongside an EPICS areaDetector [8] driver used for control only. In this system, frames are fanned out systematically from the Eiger ZeroMQ [6] stream to four process nodes, each running an OdinData stack writing one quarter of the frames. These raw datasets are then wrapped in a VDS along with the meta data captured in parallel by a fifth, separate process. This produces a final HDF5 file containing a single, coherent view of all the data from an acquisition. It is able to fully support the 10 Gbit link of the 4M; the bottleneck being pushing frames down the data link from the detector, rather than writing them to disk.

CONCLUSION

Odin has the scope to become an integral part of beamline controls at DLS. It serves many currently missing use cases, such as those that involve very high data rates or require more flexible control. The usage to up to this point in time has been successful, but there is some way to go before it can be deployed more widely. The next steps are to improve stability and to develop some new features required for future use cases. One of these being the ability to rewind a scan and continue from part way through, which will require overwriting specific chunks of data through computation of relative offsets in the HDF5 datasets based on configuration parameters. This will eventually be integrated into Malcolm [9], another ongoing development at Diamond to allow concurrent control of multiple different systems, specifically for, but not limited to, hardware-triggered scans and mapping [10] of samples.

REFERENCES

[1] J. Marchal *et al.*, "Excalibur: a small-pixel photon counting area detector for coherent x-ray diffraction - front-end design, fabrication and characterisation," in *Journal Of Physics: Conference Series*, vol. 425, pp. 530–533, 2013.

[2] C. B. Wunderer *et al.*, "The percival soft x-ray imager," in *JINST*, vol. 9, 2014.

[3] EIGER X 4M, `https://www.dectris.com/products/eiger/eiger-x-for-synchrotron/details/eiger-x-4m/`.

[4] J. Thompson *et al.*, "Controlling the excalibur detector," in *Proc. ICALEPCS'11*, pp. 894–897.

[5] N. Tartoni *et al.*, "Excalibur: A three million pixels photon counting area detector for coherent diffraction imaging based on the medipix3 asic," in *IEEE Nuclear Science Symposium Conference Record*, 2012.

[6] ZeroMQ, `http://zeromq.org/`.

[7] N. Rees *et al.*, "Developing hdf5 for the synchrotron community," in *Proc. ICALEPCS'15*, pp. 845–848.

[8] B. Martins, ADEiger, `http://github.com/brunoseivam/ADEiger/`.

[9] T. Cobb, M. Basham, G. Knap, C. Mita, and G. Yendell, "Malcolm: A middlelayer framework for generic continuous scanning," in *Proc. ICALEPCS'17.* paper TUPHA159, this conference.

[10] R. Walton *et al.*, "Mapping developments at diamond," in *Proc. ICALEPCS'15*, pp. 1111–1114.