DATABROKER: AN INTERFACE FOR NSLS-II DATA MANAGEMENT SYSTEM

A. Arkilic, L. Dalesio, D. Allan, D. Chabot Brookhaven National Laboratory, NSLSII
Upton, NY, USA

Abstract

A typical experiment involves not only the raw data from a detector, but also requires additional data from the beamline. This information is largely kept separated and manipulated individually, to date. A much more effective approach is to integrate these different data sources, and make these easily accessible to data analysis clients. NSLS-II data flow system contains multiple back-ends with varying data types. Leveraging the features of these (metadatastore, filestore, channel archiver, and Olog), this library provides users with the ability to access experimental data. The service acts as a single interface for time series, data attribute, frame data access and other experiment related information.

INTRODUCTION

The current and projected NSLS-II beamlines will support a wide range of disciplines from biology to physics with a wide range of techniques. Higher beam quality and new generations of area detectors are increasing data rates quickly. Increasingly all disciplines are using multi-element detectors to measure and record the results of the experiment. Current state-of-the-art detectors are capable of producing megapixel images with frame-rates in the kilohertz range which results in data rates of up to 800 Mb.s-1. Such data-rates are fully compatible with current off-the-shelf RAID storage technology, effectively allowing the data to be written to a storage medium as it is produced. The challenge for NSLS-II, however is to be able to process and analyse the data at this rate, with these volumes to allow the experimenter to not only make the "first-cut" for decision making during the experiment, but to retrieve data and analyse it for publication. NSLS-II middle layer beamline applications (Fig. 1), powered by MongoDB, the General Parallel File System (GPFS), and EPICS provide such capabilities.

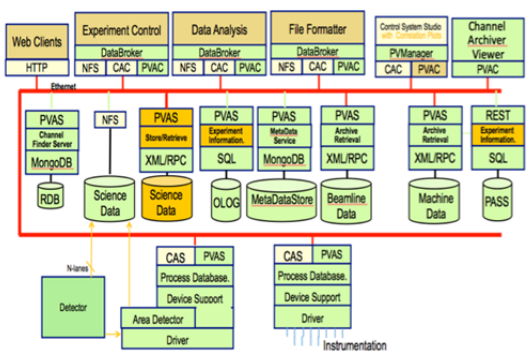


Figure 1: An overview of NSLS-II middle-layer applications.

Given the modular architecture and complexity of these various applications, a need for an easy point of access for all experimental data has occurred. In order to simplify the access to critical experimental data, we have developed databroker.

ARCHITECTURE

databroker is the pathway for data analysis tools to access data without any knowledge of experimental data. The library in production is implemented in Python and provides users with broker objects that contain Pandas data frames, allowing semi and un-structured experimental data to be accessed within scientific Python framework with ease via its narrow interface.

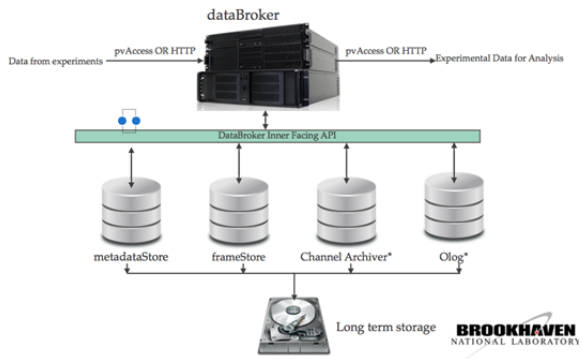


Figure 2: databroker Architecture Overview.

databroker uses the client APIs of metadatastore, filestore, channel archiver, and Olog (Fig. 2). Its primary goal is to provide users with run-based information. A run is defined as an end-to-end scan or sweep that is performed and captured either via NSLS-II run engine, BlueSky, or custom data acquisition framework that utilizes NSLS-II middle layer services. Run based data management is common to all beamlines and it provides users with the desired abstraction.

Deployment Strategy and Access Environment

All python clients that are part of NSLS-II middle layer get deployed to each beamline using the Anaconda python package management toolkit. The sandbox approach allows us to deploy packages that are required but not natively supported by the Debian 8. IPython notebook is also part of this deployment and it is the primary platform to access databroker. In addition to local access to databroker, NSLS-II beamlines provide access to an identical databroker session remotely using jupyter hub.

ISBN 978-3-95450-148-9

Searching by ID or Recency¶

Here is a summary of the "Do What I Mean" slicing supported by `DataBroker`.

syntax	meaning
`DataBroker[-1]`	most recent header
`DataBroker[-5]`	fifth most recent header
`DataBroker[-5:]`	all of the last five headers
`DataBroker[108]`	header with scan ID 108 (if ambiguous, most recent is found)
`DataBroker[[108, 109, 110]]`	headers with scan IDs 108, 109, 110
`DataBroker['acsf3rf']`	header with unique ID (uid) beginning with `acsf3rf`

Time-based Queries¶

Runs that took place sometime in a given time interval are also supported.

syntax	meaning
`DataBroker(start_time='2015-01')`	all headers from January 2015 or later
`DataBroker(start_time='2015-01-05', end_time='2015-01-010')`	between January 5 and 10

Complex Queries¶

Finally, for advanced queries, the full MongoDB query language is supported. Here are just a few examples:

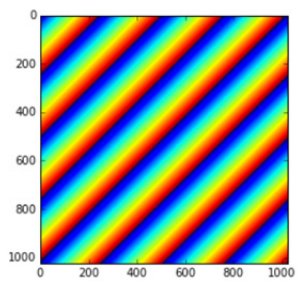
syntax	meaning
`DataBroker(sample={'$exists': True})`	headers that include a custom metadata field labeled 'color'
`DataBroker(scan_type={'$ne': 'DeltaScan'})`	headers where the type of scan was not a `DeltaScan`

Figure 3: Querying capabilities overview.

Using Databroker

As described in Fig. 3, databroker provides different querying methods such as, unique identifier based, time based, and complex/aggregated. This sort of querying approach provides users with immense data mining capabilities that were not available as a part of most legacy systems. Once a query is submitted to databroker yields to a metadatastore query. The results of this query allow databroker to compose a **header** that is composed of a RunStart, RunStop, and a set of EventDescriptor documents [1]. databroker lazily loads the Event documents, which hold the actual experiment metadata, using a generator object. This approach is performance savvy as in most cases users only want to access the header of the experimental data and request the data payload if needed. In addition to data in metadatastore, databroker is capable of locating images that are related to each data point within specific metadatastore events. As shown in the jupyter session in Fig. 4, querying and accessing complex experimental data requires a handful of comments, allowing users to get to analysis stage with very little effort.

```
In [1]: from dataportal.broker import DataBroker

In [2]: from dataportal.muxer import DataMuxer

In [3]: header = DataBroker[-1]

In [4]: events = DataBroker.fetch_events(header)

In [5]: dm = DataMuxer.from_events(events)

In [6]: dm.sources
Out[6]: {u'None_acquire_period': u'PV:ADSIM:cam1:AcquirePeriod_RBV',
         u'None_acquire_time': u'PV:ADSIM:cam1:AcquireTime_RBV',
         u'None_image_lightfield': u'PV:ADSIM:',
         u'None_stats_total1': u'PV:ADSIM:Stats1:Total_RBV',
         u'None_stats_total2': u'PV:ADSIM:Stats2:Total_RBV',
         u'None_stats_total3': u'PV:ADSIM:Stats3:Total_RBV',
         u'None_stats_total4': u'PV:ADSIM:Stats4:Total_RBV',
         u'None_stats_total5': u'PV:ADSIM:Stats5:Total_RBV',
         u'm1': u'PV:LSBR-DEV:m1.RBV',
         u'sclr_ch2': u'PV:AISIM:ai1'}

In [7]: images = dm[u'None_image_lightfield']

In [8]: import matplotlib.pyplot as plt

In [11]: %matplotlib inline

In [12]: plt.imshow(images.values[0])
Out[12]: <matplotlib.image.AxesImage at 0x7f1a24adf290>
```




```
In [ ]:
```

Figure 4: A sample jupyter Session and access to sample experiment via databroker interface.

databroker does not only support experimental data saved in filestore and metadatastore. NSLS-II beamlines archive all process variables within the scope of an experiment in channel archiver. By providing process variable based search, databroker provides access to this experimental data. This allows users to investigate factors they left out during data acquisition stage. Another important aspect of databroker is its capability to access Olog, the rich logging environment [2]. Users can create log entries in which they can denote interesting findings of their experiments. This is very useful as Olog provides users with CS-Studio and web browser views.

CONCLUSION

databroker provides access to complex, high-performance, and modular NSLS-II middle layer framework by providing a well-defined narrow interface that is accessible both locally and remotely. Its intuitive and simple API allows end-users to start analyzing

ISBN 978-3-95450-148-9

experimental data during their experiment,. This allows them to discover interesting attributes of their samples while they still have the chance to investigate them further.

REFERENCES

[1] metadatastore: A primary data store for NSLS-II beamlines, A. Arkilic, L. Dalesio, D. Allan, W. K. Lewis, Presented at the ICAPLEPCS 2015.

[2] Olog and the CSStudio: A rich logging environment, K. Shroff, L. Dalesio, A. Arkilic, E. Berryman Presented at the 2013 ICALEPCS.

ISBN 978-3-95450-148-9