

Databroker / Tiled for Accelerator Physics Experiments

Daniel Allan

Data Engineering Group Lead

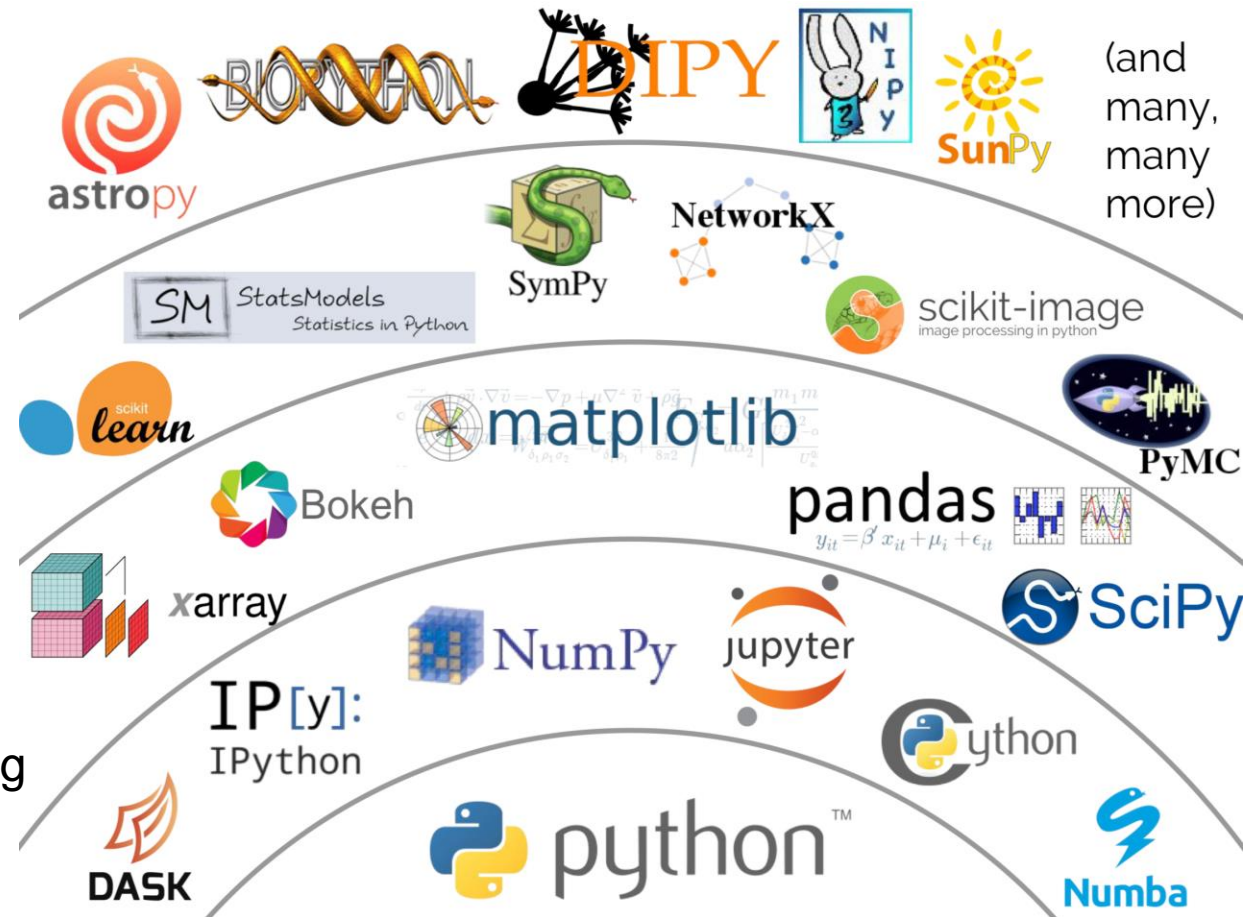
Data Science and Systems Integration Program

National Synchrotron Light Source II

whoami

- Ph.D. in Experimental Condensed Matter Physics (Johns Hopkins University)
- In 2015, started the *Bluesky* project, a green-field (blue sky...) open-source Python approach to data acquisition and data access experimental science, with collaborators Thomas Caswell (NSLS-II) and Ken Lauer (formerly LCLS)
- *Bluesky* has grown within NSLS-II and around the world
- Focusing on data access, management at Data Engineering Group Lead

Bluesky is a bridge from data acquisition to the SciPy ecosystem



SciPy ecosystem:
Diverse science fields
encountering problems
that "rhyme" and sharing
solutions that layer and
compose well

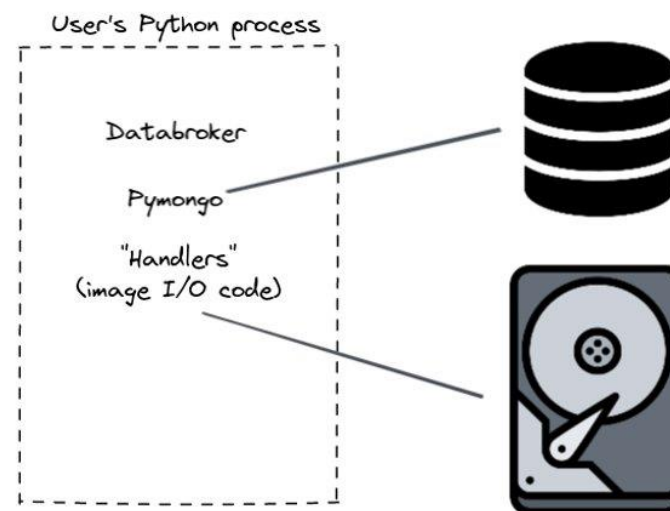
Bluesky worked downward from data analysis to data acquisition

- A collaboration between science Ph.Ds and Controls experts
- Building what we had wanted as graduate students
- Working downward:
 - scikit-beam
 - databroker
 - bluesky (orchestration)
 - ophyd
 - [caproto]



Databroker: a Python I/O abstraction

- The name *Databroker* came from EPICS co-founder Bob Dalesio
- The idea was to abstract over I/O for data and metadata coming from many different sources
- Successfully did this!
- But with limitations...
 - Python-specific
 - No remote access
 - No security model



There is a Broad Effort to Build Data *Services*

- Earthmover: *Arraylake*
- CERN: *Rucio*
- IRIS-HEP: *ServiceX*
- Pangeo: *Pangeo Forge*
- Janelia: *DVID*
- Multiple efforts at Microsoft, NASA, etc.
- Uncountable in-house industry projects

"Today we have lots of data, yes, but it's not universally accessible or understandable. We should invest in data access technologies and metadata standards to allow the world to ask questions without munging."

- Andy Mueller, Scikit-Learn / Microsoft

Source:

<http://matthewrocklin.com/blog/work/2022/07/19/scipy-mission>

Data Services streamline these steps

- **Find** the data
- Access the **specific part** we need (maybe much less than all of it)
- Obtain it in a **format** our data analysis program can read
- Understand what the **names** mean
- Follow **relationships** between data sets
- As needed, provide **open** access or secure, **authorized** access
- Meet the **F.A.I.R.** principles achievable with reasonable effort

Synchrotron science has particular challenges

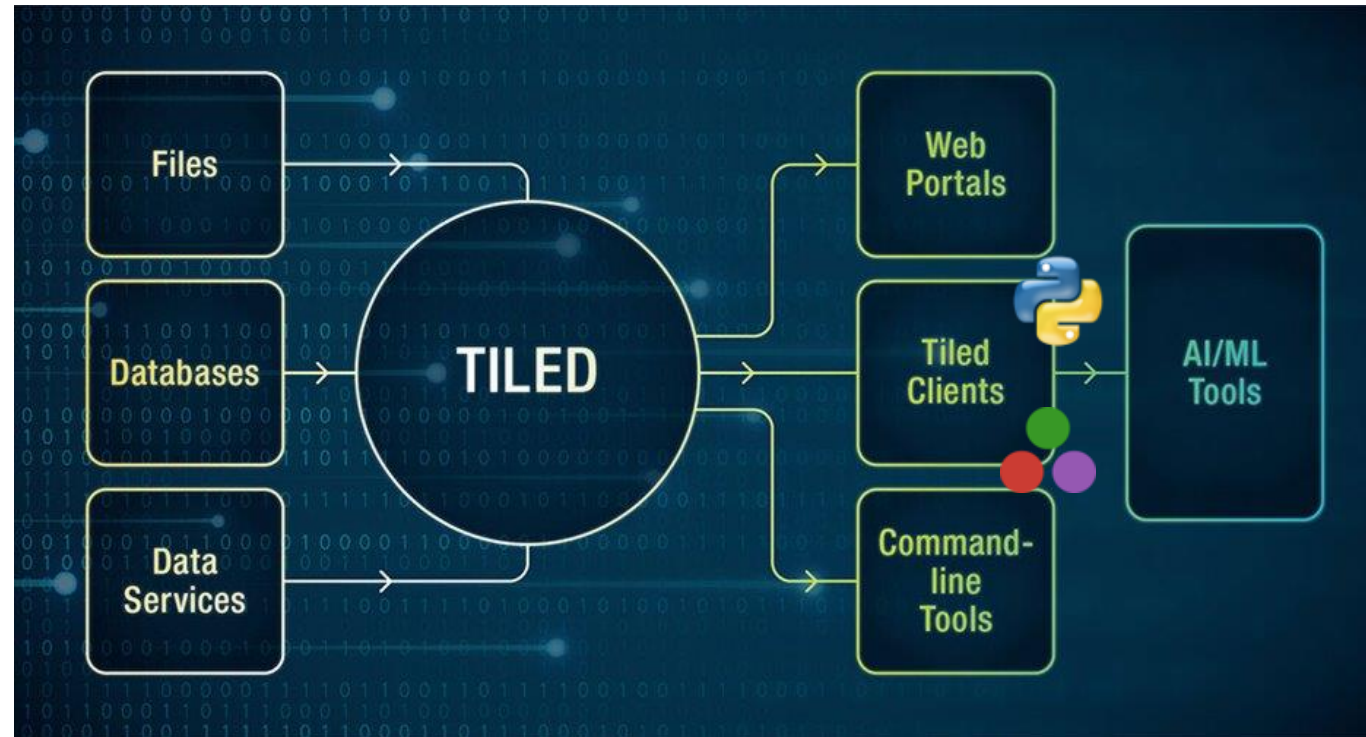
- Large variety of...
 - Shapes (tables, images, image time series, hierarchical structures)
 - Sizes (kilobyte to terabyte)
 - Access patterns
- Highly dynamic requirements
 - Equipment added and removed
 - Evolving structures (framing detectors to event-based detectors)
 - Evolving formats and schema evolution
 - In other domains (astrophysics, climate science, business analytics) the data structure are much more static and defined in advance.

We have built a Data Service called *Tiled*

- We have spoken with many, many external groups through the design and development process to better understand this space, our own requirements, and opportunities
- *Tiled* passed routine security scans and a *specifically targeted* cybersecurity "pen test" with a clean bill of health
- Development began in February 2021. Tagged "beta" in May 2024.

Tiled: A Structured API to Data

- **Search** on metadata
- **Slice** into remote datasets
- **Transcode** between formats
- **Download** partial or whole datasets
- Or **find** data storage location(s) for direct access
- Implement web **security** standards and authorization



Emphasis on a fixed set of well-defined *structures* makes this tractable

- Strided array (like numpy)
- Column-major table (like Apache Arrow)
- Wide, nested structure of these (like a directory or HDF5 group)
- Sparse array (different enough semantics that it gets its own thing)
- Awkward array (nested, variable-sized arrays, *a la* HEP ROOT)
 - We are just beginning to use event-based detectors at synchrotrons
 - We are interested in using Awkward / ROOT examples to drive big-scale performance tests while the project is still malleable

Data directly into an analysis program

```
In [1]: from tiled.client import from_uri

In [2]: c = from_uri("https://tiled-demo.blueskyproject.io/api")

In [3]: c
Out[3]: <Node {'generated', 'csx', 'bmm', 'fxi', 'um2022'}>

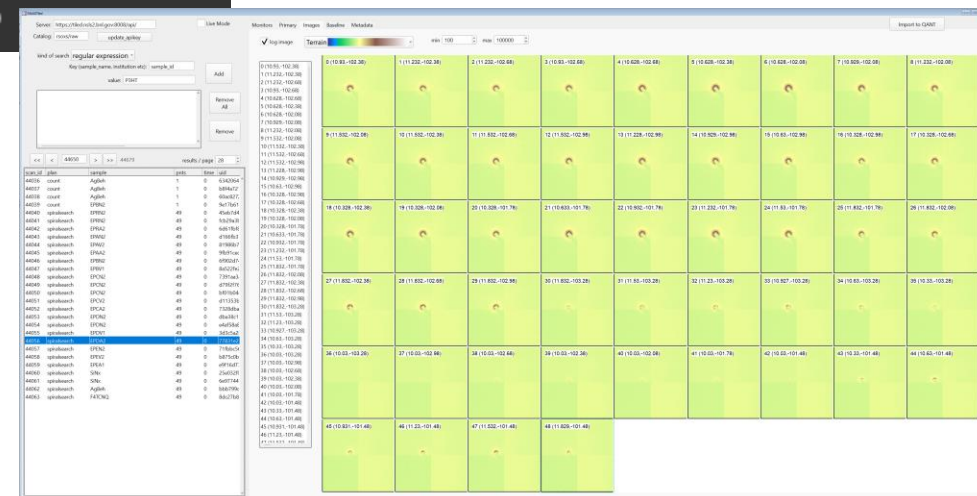
In [4]: run = c['bmm']['raw'].values().last()

In [5]: run
Out[5]: <BlueskyRun {'baseline', 'primary'} scan_id=32450 uid='10cefec0' 2022-03-27 09:29>

In [6]: run['primary']['data']['IO'][:5]
Out[6]: array([68.24125113, 68.09572678, 68.09231882, 67.90571976, 67.71874719])
```

A custom (scientist-written!) Igor program loads data from Tiled over HTTP, integrating search and viz

Tiled's officially-supported Python client loads data efficiently into numpy, pandas, xarray



Data in a web browser, on a phone

TILED

BROWSE

Top / fxi / raw / 1b0b4d73-6d87-43ab-8d62-ed035c51b9b4 / primary / data / Andor_image

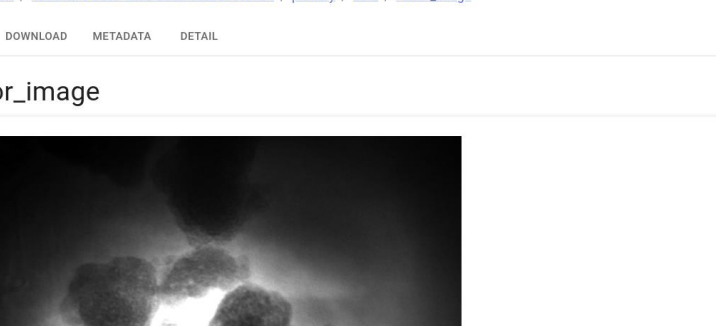
VIEW

DOWNLOAD

METADATA

DETAIL

Andor_image



Choose a planar cut through this 4-dimensional array.

ⓘ This large array has been downsampled by a factor of 2. Use the "Download" tab to access a full-resolution image.

022

044

010

19

TILED

BROWSE

Top / fxi / raw /

1b0b4d73-6d87-43ab-8d62-ed035c51b9b4 /

primary / data / Andor_image

VIEW

DOWNLOAD

METADATA

DETAILS

Andor_image



Choose a planar cut through this 4-dimensional array.

① This large array has been downsampled by a factor of 3. Use the "Download" tab to access a full-resolution image.

0

44

22

0

19

10

TILED

BROWSE

[Top](#) / [fxi](#) / [raw](#) / [1b0b4d73-6d87-43ab-8d62-ed035c51b9b4](#) / [primary](#) / [data](#) / [Andor_image](#)

VIEW

DOWNLOAD

METADATA

DETAIL

Dimensions: 45 × 20 × 1080 × 1280

Slice (Optional)

EXAMPLES

If blank, access entire array

Format *

CSV

DOWNLOAD

LINK

OPEN

Link

<https://tiled-demo.blueskyproject.io/a>

File browsers that can't "see inside" the files present barriers to science

- The format data is **written in** (e.g. by a detector) is not necessarily always the most **convenient** or **performant** one.
- **Often, no partial access** ("Just download the first couple points from each of a bunch of files so I can look at them.")
- No clean way to express **relationships between data across files**, especially linking derived data to raw data
- No way to **preview** many scientific formats in a web browser
- **Search** capabilities are limited.

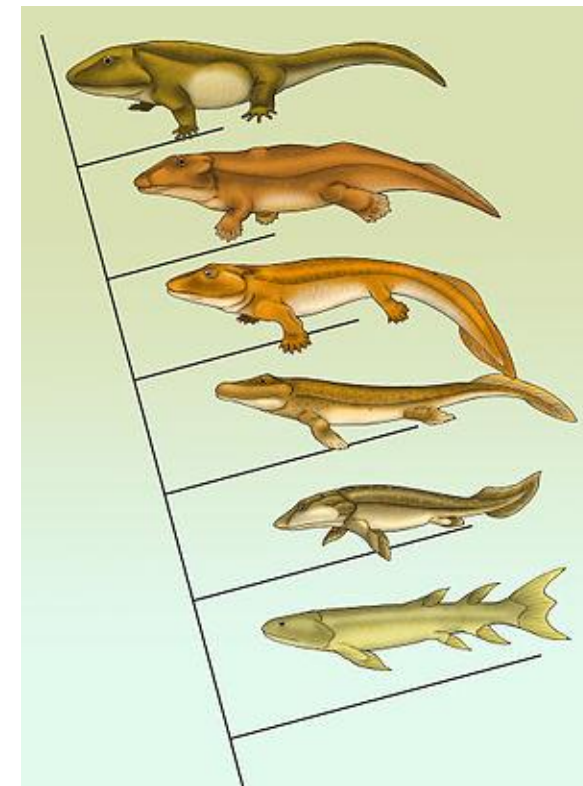
Index of /

Name	Last modified	Size	Description
 folder1/	2014-09-30 22:08	-	
 folder2/	2014-08-06 19:14	-	

Apache/2.4.7 (Ubuntu) Server at localhost Port 80

Wading out of the Data Lake

0. Heap of bytes (at least you *have* the data...)
1. A database of filepaths (or URIs) with associated metadata would let us *search*.
2. But if the server knows enough about the files to *open* and *read* them and describe their *structure* to clients, clients don't know have to know the storage details to slice into the data structurally.
3. And, for extra credit, if the markup follows some convention that we can communicate to the client (e.g. "XDI" or "NXcanSAS") we can go further still.



Tiled can provide slices of data as...

- Custom, one-off text format designed to be parsed by a **30-year-old bash script** that still works ("if it ain't broke...")
- Traditional formats like CSV, TIFF, or HDF5 to be opened by **Igor, Origin, ImageJ, PyMCA,**
- **Web-friendly** image or data formats to be directly displayed by a web browser or web application
- Chunks of compressed **C** or **Arrow**-encoded buffers to be fetched on demand and fed zero-copy to **Tensorflow**
- Or, just use Tiled as a database of filepaths (or URIs)



We can meet all users where they are!

Like Jupyter Notebook, Tiled is something a single user can easily run

- Easy to run on a laptop:
 1. `pip install tiled[all]`
 2. `tiled serve directory my_files/`
 3. Access data from your web browser or Python or any Internet-connected program.
- Tiled can also scale up for large **multi-user deployments**, like JupyterHub.

What about *Databroker* then?

- We have reimplemented *Databroker's* Python API on top of an HTTP client, for a backward-compatible user API.
- *Databroker* will soon go into "maintenance mode" and be maintained for at least 1 U.S. Ph.D-length (~5 years).
- The future is *Tiled*.

***Tiled* incorporates the input and experience of many people**

- **NSLS-II**: Dan Allan, Stuart Campbell, Thomas Caswell, Marcus Hanwell, Juan Marulanda, Eugene Matviychuk, Padraic Shafer
- **ALS**: Dylan McReynolds, Joseph Kleinhenz
- Builds on open-source collaboration with Martin Durant (**Anaconda, Inc.**) with involvement from Garrett Bischof (**NSLS-II**).
- Contributions from the **IRIS-HEP** collaboration to add support for AwkwardArray

Support Infrastructure

- The project is 3-clause BSD licensed and covered by the Bluesky Project multi-facility governance model github.com/bluesky/governance
- It is supported as an open source project by NSLS-II's Data Science and Systems Integration Program.
- NSLS-II deploys it on the public web at [tiled.nsls2.bnl.gov](https:// tiled.nsls2.bnl.gov). (BNL login is required to access any of the data in it.)
- **At NSLS-II we are betting our data access on this.**

Links

Demo: <https://tiled-demo.blueskyproject.io/>

Documentation: <https://blueskyproject.io/tiled>

Code: <https://github.com/bluesky/tiled>

Contact: Daniel Allan <dallan@bnl.gov>