PHOTONIC – Photon data focused

Photon Science - **H**armonized **O**nline **T**ools for **O**pen, **N**avigable, **I**nteractive **C**omputing

Tim Wetzel, Patrick Fuhrmann, Sophie Servan, Paul Millar, Shruti Ghargi, Sebastian Paripsa, Linus Pithan, Devin Burke – **DESY**

Luca Gelisio, Thomas Michelat – **EuXFEL**

Nicolas Hayen – **CAU Kiel**

Slides courtesy by Patrick Fuhrmann

PHOTONIC https://photonic-project.org/

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Forschung, Technologie und Raumfahrt unter dem Förderkennzeichen C3-K2500180 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei der Autorin / beim Autor

Gefördert vom

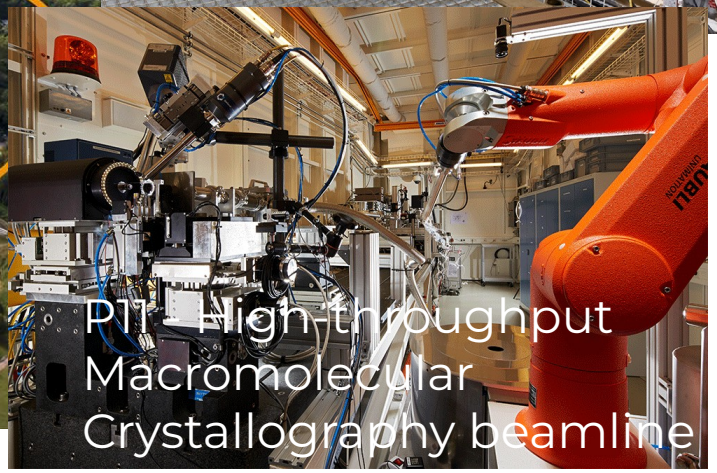
Federal Ministry of Research, Technology and Space

Our Data (light) sources



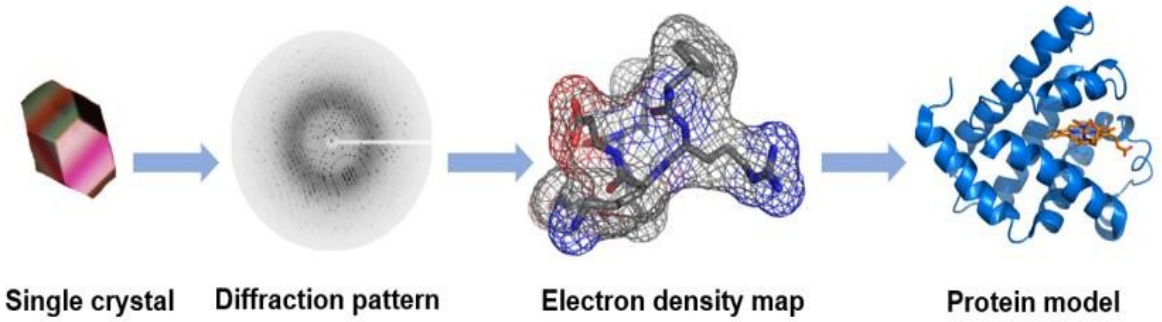




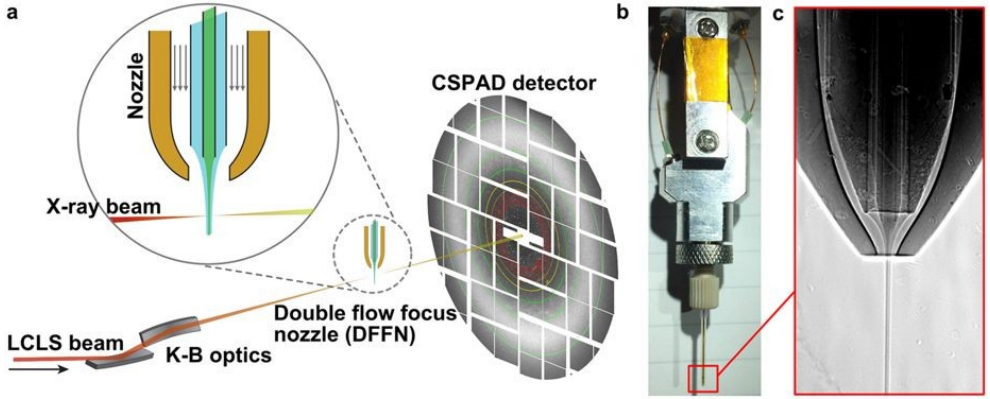


Using Photons and Neutrons

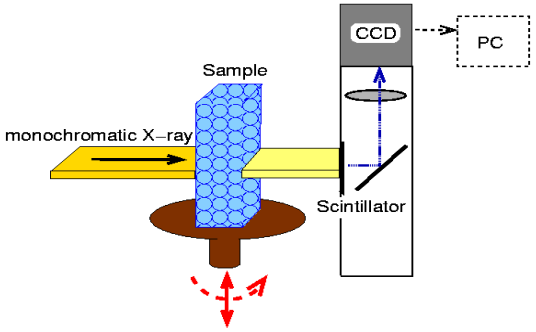
Conservative Crystallography



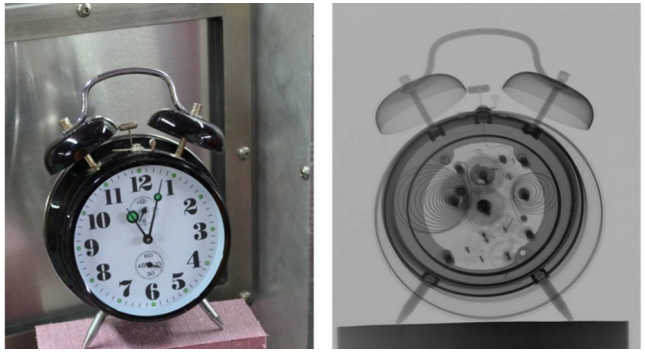
Serial Crystallography



Photon Tomography



Neutron Tomography



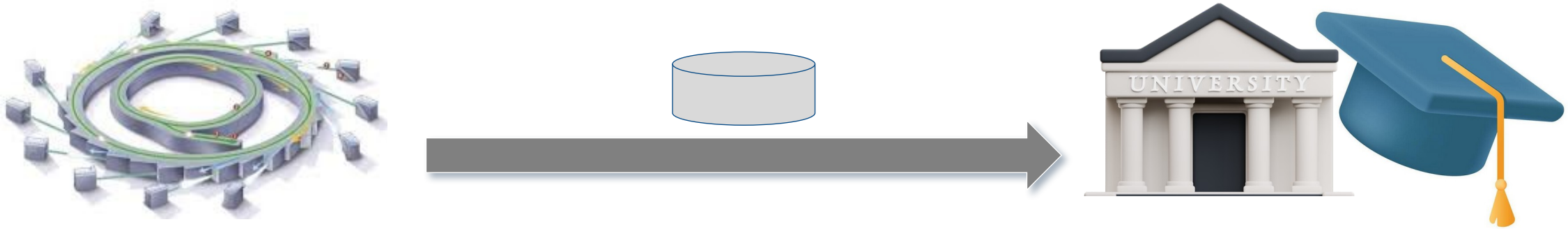
A new Neutron Radiography / Tomography / Imaging Station DINGO at OPAL

Plasma Physics, Quantum Physics,

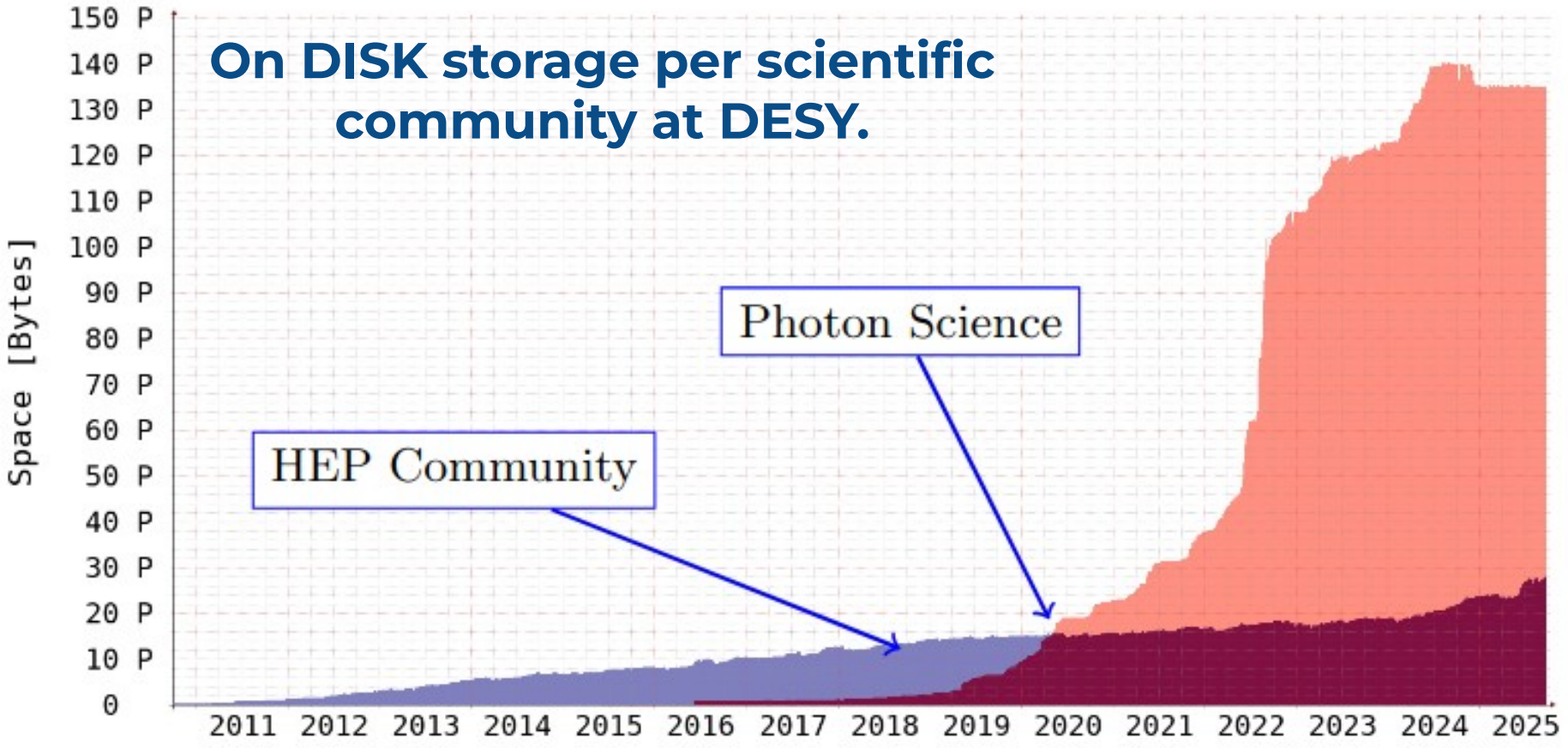
See Lightsources.org

What is the issue we need to solve!

- Beamlines produce a huge amount of data, but the time and capacity of the remote compute infrastructure (mostly laptops) is limited.
- And a large portion of science is done by research groups (students) at the Universities.
- **GOAL**: Provide universities with access to scientific data generated by high-volume facilities, such as synchrotrons and enable remote visualization and analysis.

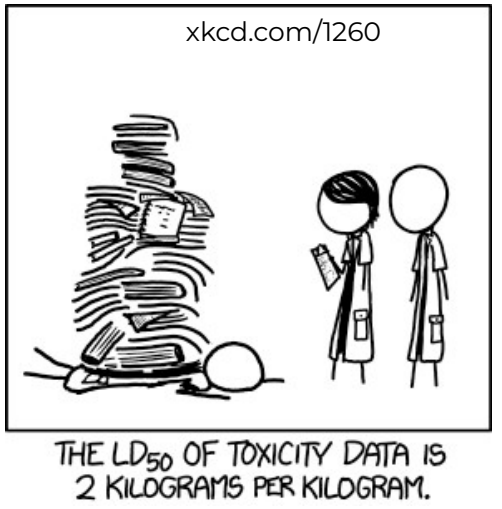


Why would that be an issue ?



On DISK storage per scientific community at DESY. With the start of the EU-XFEL and PETRA III, the Photon Science community dominated the storage needs (significantly)

There are techniques (beamlines) creating petabytes of data per beamtime within a couple of days.



X-Ray Reflectivity (XRR) Analysis

From Nicolas

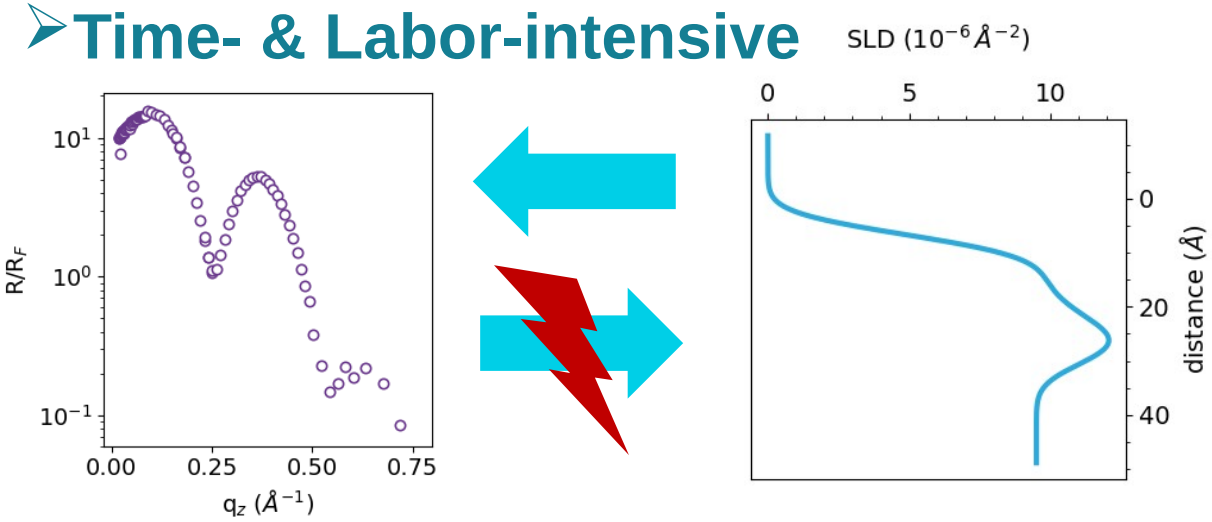
PHOTONIC

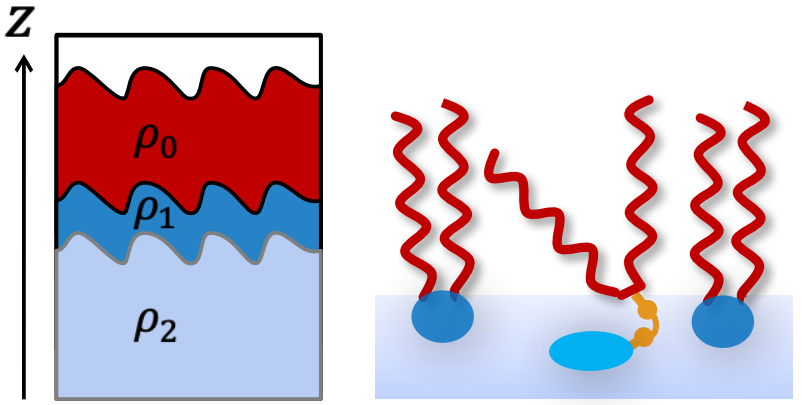
X-ray reflectivity measurements (XRR) is an established and widely used technique for investigating nanoscale structure at solid and liquid interfaces and surfaces, e.g. semiconductor components and lipid Langmuir films. From the characteristic curve of XRR, conclusions can be drawn about the **nature of the sample interfaces, in particular the electron density, layer thickness, and roughness of the interfaces down to atomic resolution.**

What can XRR reveal about surfaces and interfaces?

- Reflectivity curves are converted to density (Scattering Length Density) profiles
- Phase problem: XRR does not uniquely imply SLD
- Modeling with prior assumptions + parameter optimization

➢ **Time- & Labor-intensive**

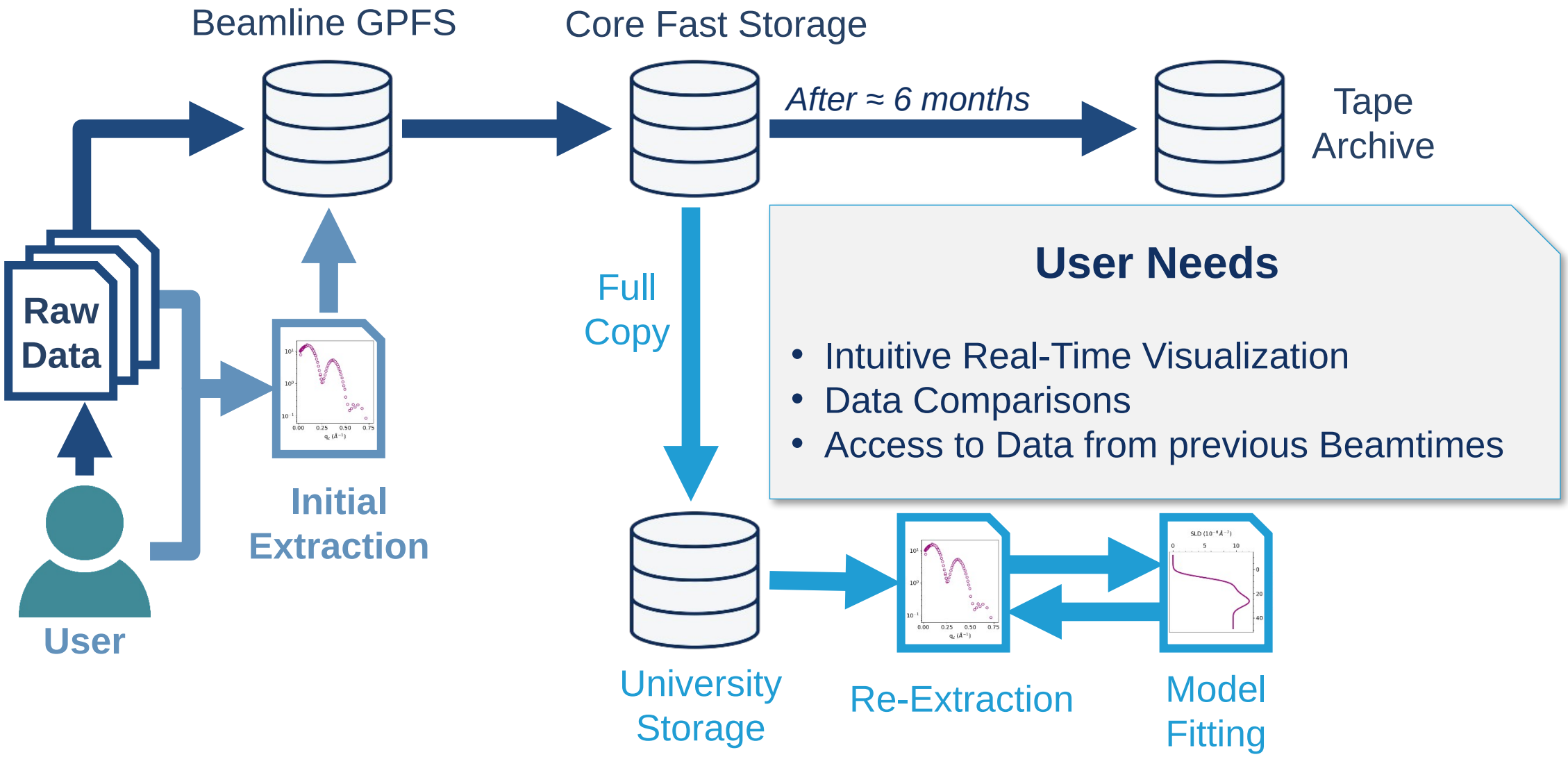




XRR Data Management

From Nicolas

PHOTONIC



User Needs

- Intuitive Real-Time Visualization
- Data Comparisons
- Access to Data from previous Beamtimes

XRR Summary

From Nicolas

PHOTONIC

XRR Requirements

- XRR Users need interactive data exploration and visualization
- Reusable analysis tools need **common standards and stable interfaces**

PHOTONIC's work

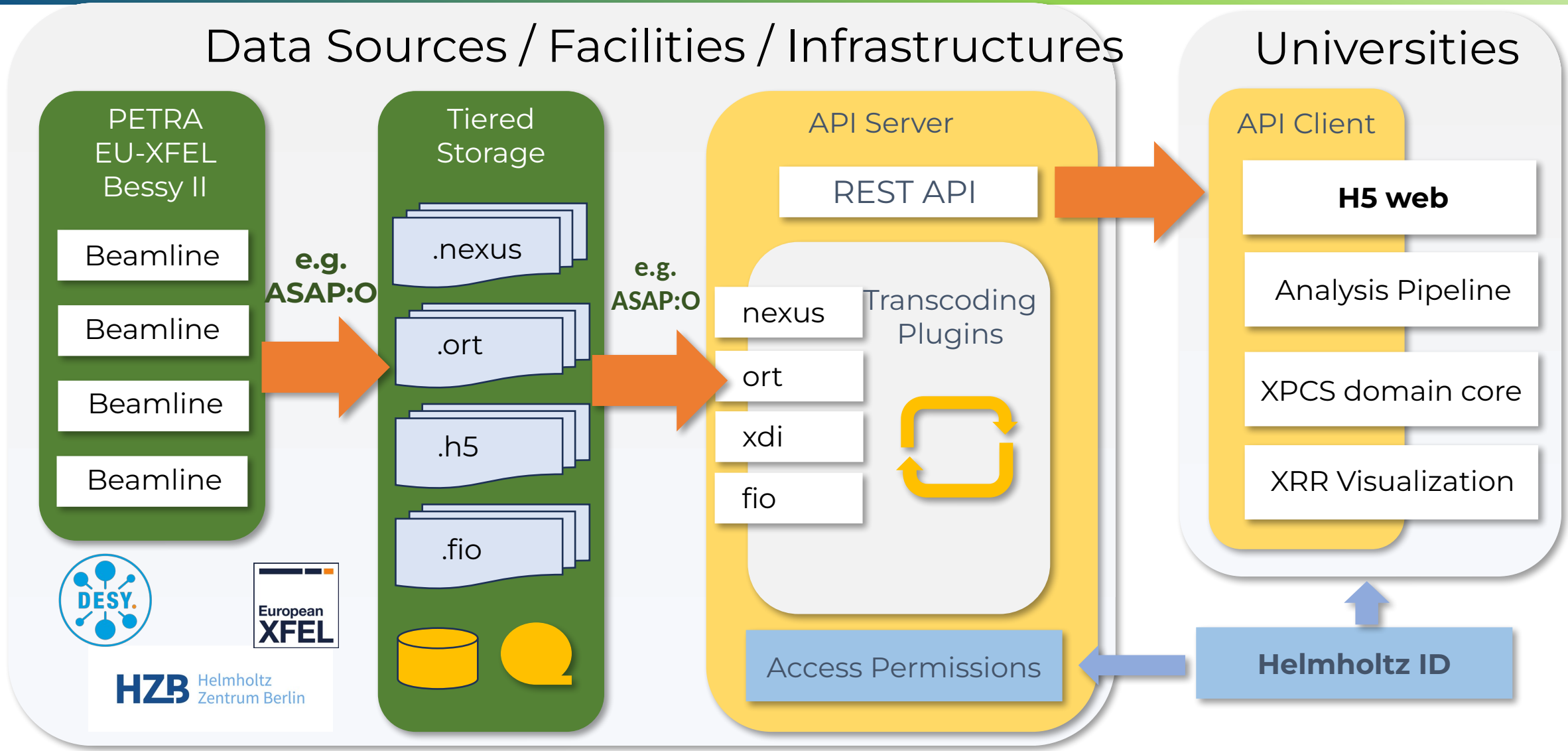
- Standard XRR data **interoperability formats**
- Accessing & **transcoding data**
- Integration with new **client software**

High level goals (more specific)

- Accessing specific slices of data from a recent or ongoing experiment at their home institutions for visualization or **ad-hoc data analysis**.
- Using a personal laptop during beamtime (e.g., via eduroam Wi-Fi) to conveniently **access live data** generated during the ongoing measurement.
- Exploring experimental data after completing a beamline experiment, **using facility-provided computing clusters** for initial data reduction and analysis to identify publishable results.
- **Training on open datasets** to prepare for and optimize upcoming beamtime experiments.
- Supporting **Open and FAIR data principles** by enabling the download of datasets from other experiments once the embargo period has ended

The Architecture

Existing | New Components



Tiled by Bluesky

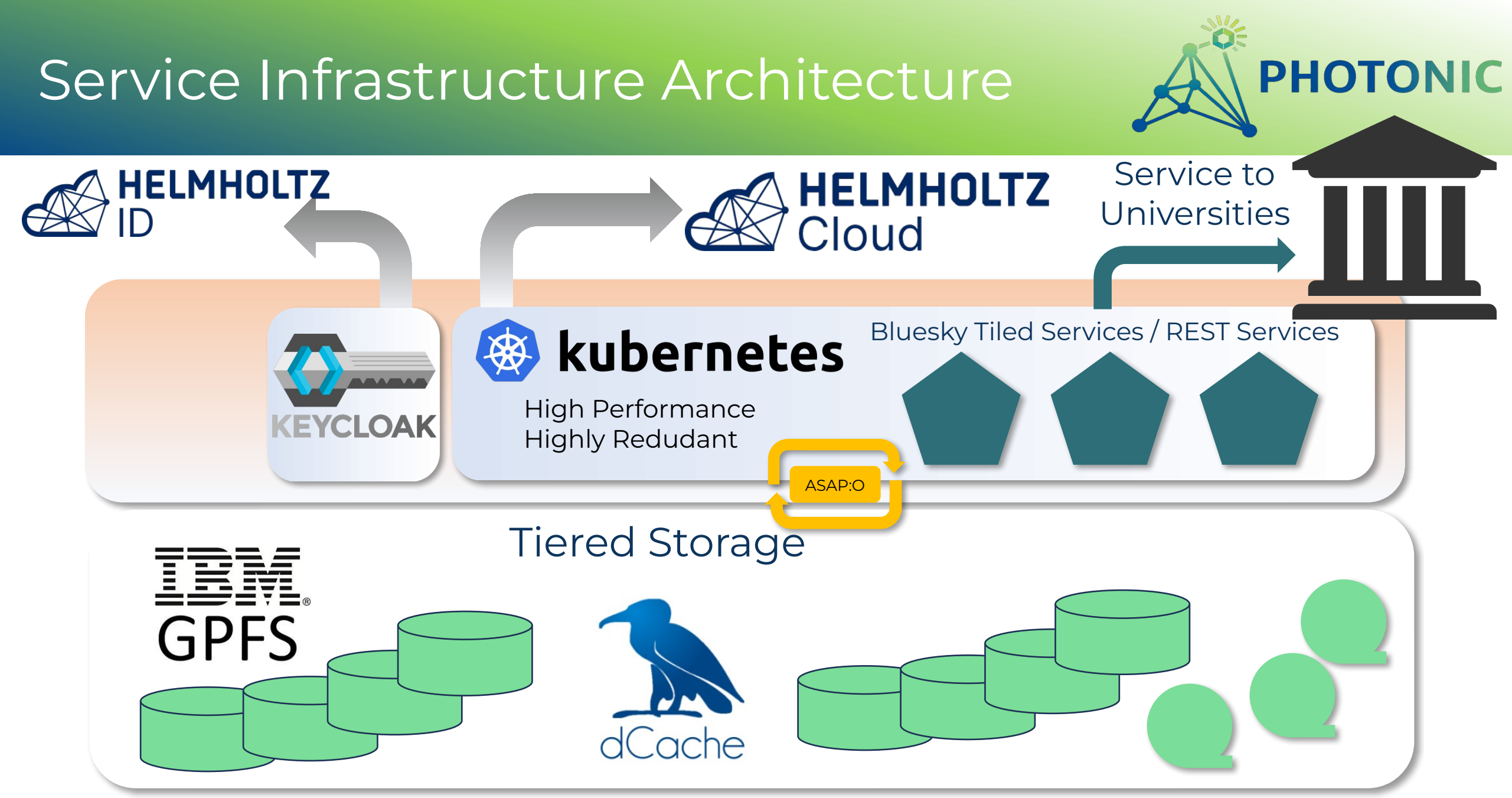
https://github.com/bluesky/tiled
https://blueskyproject.io/tiled/

PHOTONIC

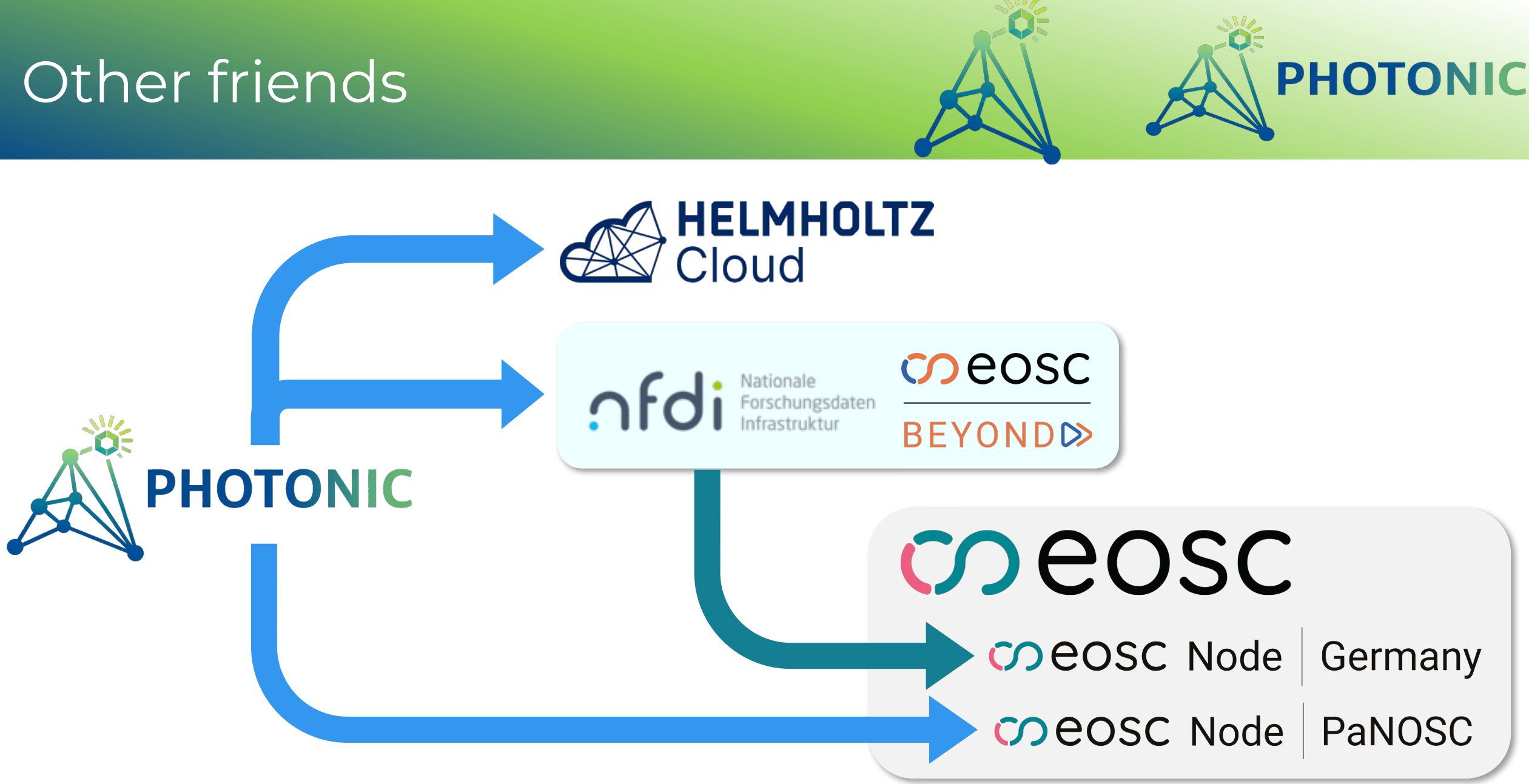
From Devin

- Tiled is a service that enables secure, structured access to scientific data.
- Supports search, **remote slicing, format conversion, and live data streaming**.
- Developed at **NSLS-II**, it is seeing production beamlines at American facilities and experimental use at **Diamond** and **HZB BESSY-II**.
- Ships with a python client API for easy integration into server and client systems.
- **Two development instances** now run in our DMZ-photonic Kubernetes cluster at DESY.

Service Infrastructure Architecture



Other friends



Already done ....

From Shruti

PHOTONIC

- Checking server – tiled client in a Jupyter Notebook

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

[2]: client = from_uri("http://localhost:8000")

[3]: client

[3]: <Container {'scalars', 'nested', 'tables', 'structured_data', ...} ~8 entries>

[4]: client['nested/images']

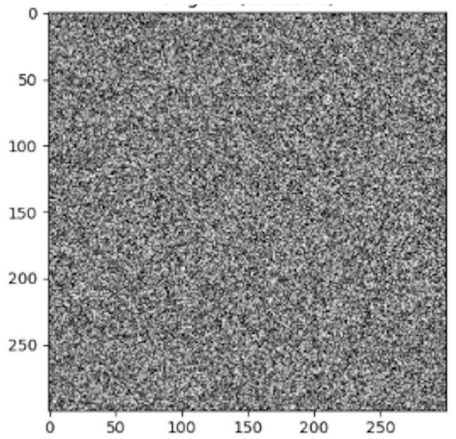
[4]: <Container {'tiny_image', 'small_image', 'medium_image', ...} ~4 entries>

[5]: client['nested/images/small_image']

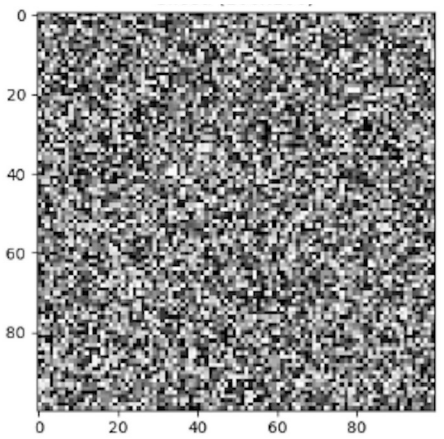
[5]: <ArrayClient shape=(300, 300) chunks=((300,), (300,)) dtype=float64>

[6]: sliced_image = client['nested/images/small_image'][0:100, 0:100]
```

Original (300 x 300)



Slice (100 x 100)



Client understands the data structure from the server side!

Client only downloads the slices needed!

Already done ....

From Shruti

PHOTONIC

- Browsing, visualising & exporting to different formats

2. Browse the catalog tree

- `list(client)` shows top-level entries called nodes.
- Nodes can nest arbitrarily deep (processed → integrated → 2theta ...)
- No need to know the full path in advance. It is possible to explore interactively

[2]:
```
top_level_folders = list(client)
print(f"{len(top_level_folders)} top-level entries")
top_level_folders[:10]
```

```
2 top-level entries
['p10', 'p02.1']
```

[3]:
```
# Drill into one of the entries to see how the tree ne
client["p02.1"]
```

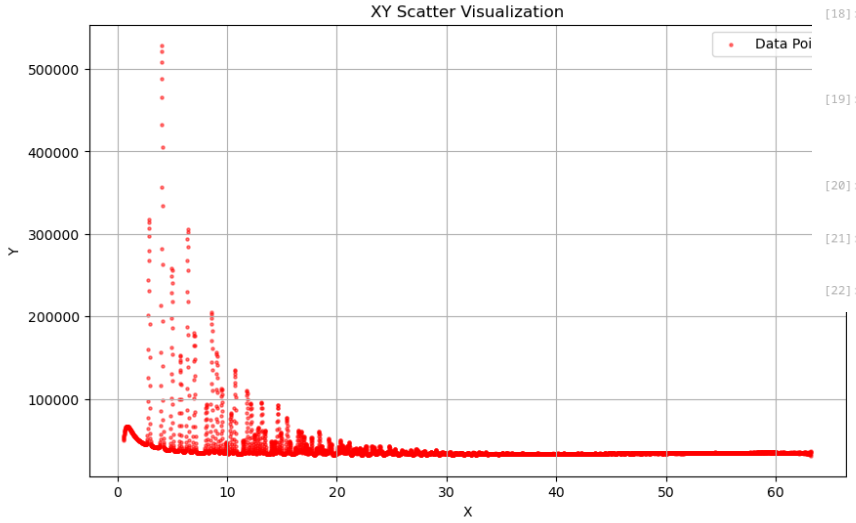
```
<Container {'raw', 'processed', 'lost+found'}>
```

[4]:
```
# Navigate into subfolders
sub_folders = client["p02.1"]["processed"]["merged"]
top_level = list(sub_folders)
print(f"{len(top_level)} files")
top_level[:10]
```

```
80 files
['empty_0p6_sg_007_250K_avg_123_',
 'empty_0p6_sg_015_450K_sum_243_',
 'Cu3Bi2S4Cl_0p6_sg_004_175K_avg_368_',
 'Cu3Bi2S4Cl_0p6_sg_006_225K_sum_398_',
 'Ni_0p6_sg_avg_596_',
 'empty_0p6_sg_002_125K_sum_48_',
 'empty_0p6_sg_018_300K_avg_288_',
 'Cu3Bi2S4Cl_0p6_sg_014_425K_sum_518_',
 'empty_0p6_sg_001_100K_avg_33_',
 'empty_0p6_sg_006_225K_sum_108_']
```

[12]:
```
# Access XY file through custom XY adapter
xy_file = client["p02.1"]["processed"]["integrated"]["2theta"]["xy"]["300_LaB6_660c_0p8_sg_avg_17_17_2thet
# data[:, 0] is all rows for the first column and data[:, 1] is all rows for the second column
x = xy_file[:, 0]
y = xy_file[:, 1]

plt.figure(figsize=(10, 6))
plt.scatter(x, y, s=5, color="red", alpha=0.5, label="Data Points")
plt.title("XY Scatter Visualization")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid(True)
plt.legend()
plt.show()
```



[ ]:
```
import os

os.makedirs("files", exist_ok=True)
```

[16]:
```
# `.formats` lists all the supported mimetypes/file formats the container can be exported to.
spec_file = client["p10"]["spec"]["align2"]["2"]["Time"]
spec_file.formats
```

[16]:
```
['application/json',
 'application/octet-stream',
 'application/vnd.ms-excel',
 'application/x-hdf5',
 'image/png',
 'image/tiff',
 'text/csv',
 'text/html',
 'text/plain',
 'text/x-comma-separated-values',
 'text/x-xdi']
```

[ ]:
```
# Export SPEC to HDF5
print (spec_file)
print(type(spec_file))
list(spec_file)
spec_file.export("exported-files/spec_h5.h5", format="application/x-hdf5")
```

```
<ArrayClient shape=(5,) chunks=((5,)) dtype=float64>
<class 'tiled.client.array.ArrayClient'>
```

[18]:
```
# Export by file extension.
#Tiled infers the target format automatically from file extension.
xy_file = client["p02.1"]["processed"]["integrated"]["2theta"]["xy"]["300_LaB6_660c_0p8_sg_avg_17_17_2theta.xy"]
xy_file.export("exported-files/xy_csv_export.csv")
```

[19]:
```
# Export by Mimetype.
#Specify a format (format="text/x-xdi") to force a specific encoding type.
file = xy_file
file.export("exported-files/xy_xdi_export.xdi", format="text/x-xdi")
```

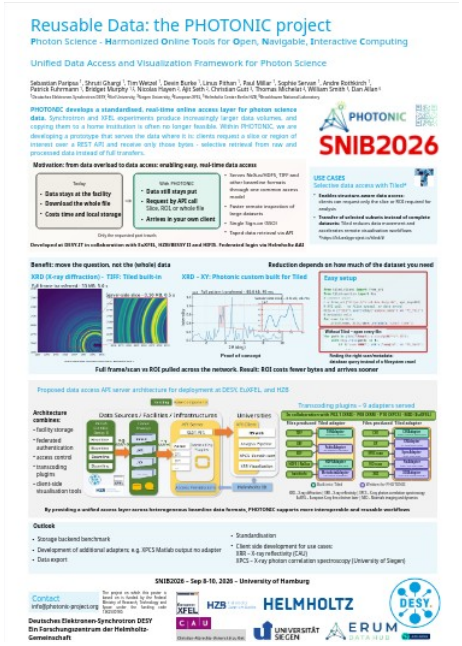
[20]:
```
edf_file = client["p02.1"]["processed"]["pyFAI"]["300_LaB6_660c_0p8_sg-00017_mask.edf"]
edf_file.export("exported-files/edf_xdi_export.xdi", format="text/x-xdi")
```

[21]:
```
xy = client["p02.1"]["raw"]["20260122112119_Beamline"]["alignment"]["20260122113503_s2_cy.xy"]
xy.export("exported-files/xy_hdf5_export.h5")
```

[22]:
```
edf_file = client["p02.1"]["processed"]["pyFAI"]["300_LaB6_660c_0p8_sg-00017_mask.edf"]
edf file.export("exported-files/edf h5 export.h5", format="application/x-hdf5")
```

Thanks

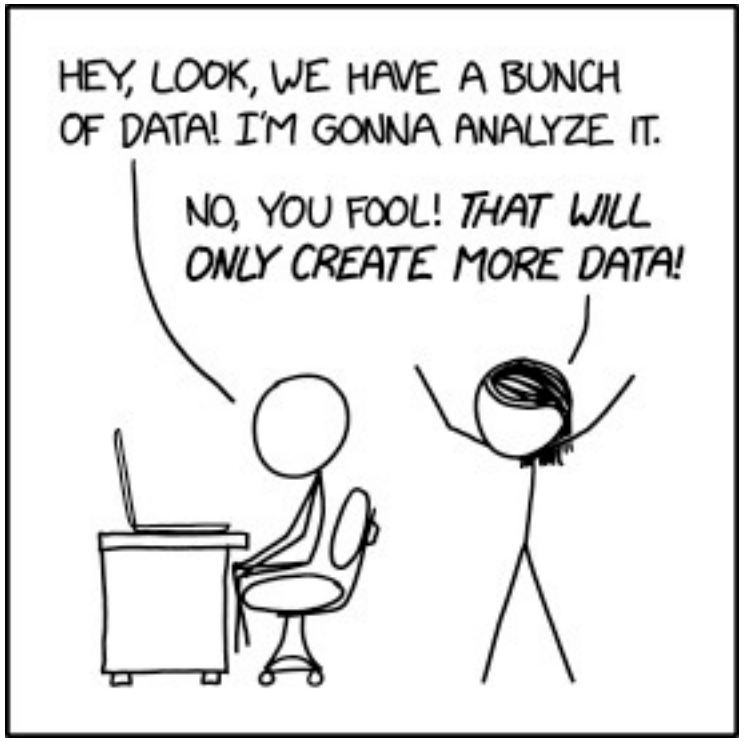
Questions?



Our poster at SNIB2026:

https://doi.org/10.5281/zenodo.22769688

Contact:
Tim Wetzel
tim.wetzel@desy.de



xkcd.com/2582