



das

A Python Framework for DAS

The ultimate use case:



3 interrogators



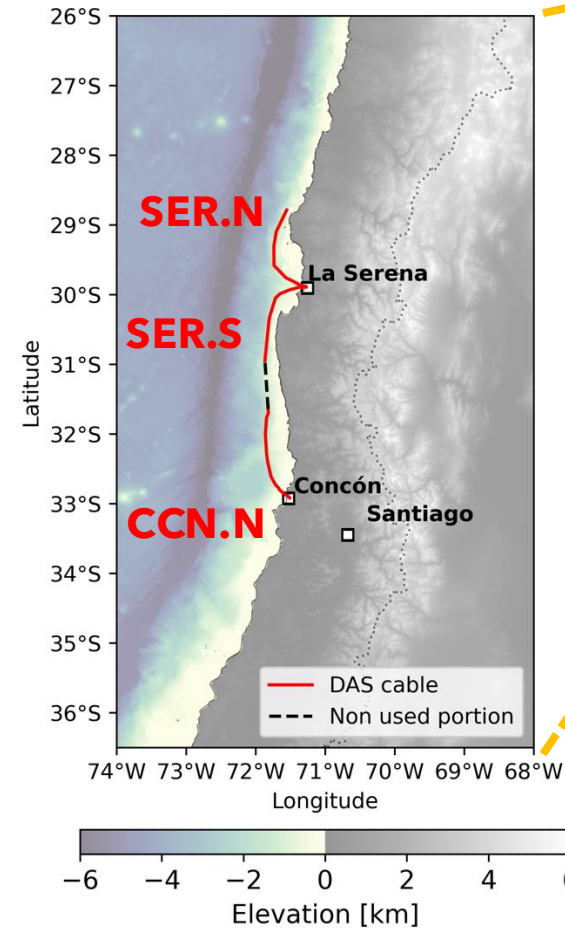
450 km / 30 000 sensors



400 GB / day
25 000 files / day

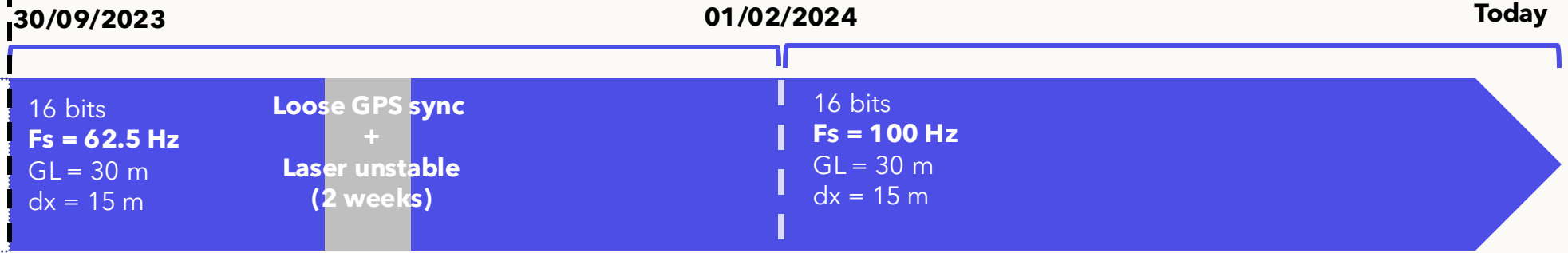


5 years

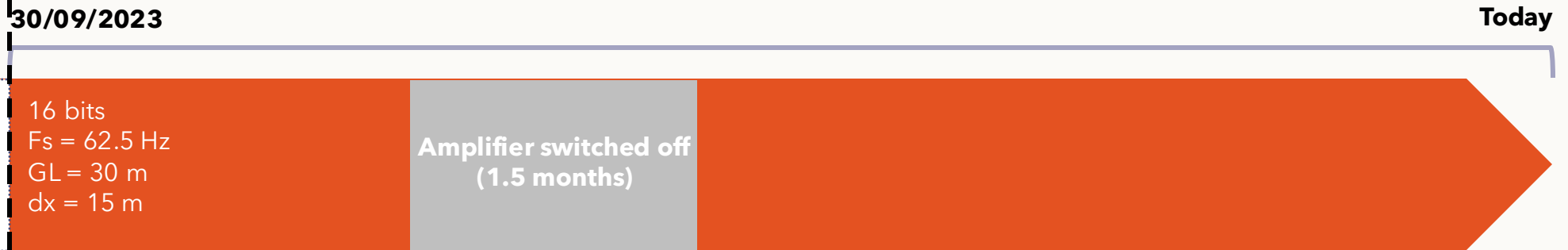


SER.N

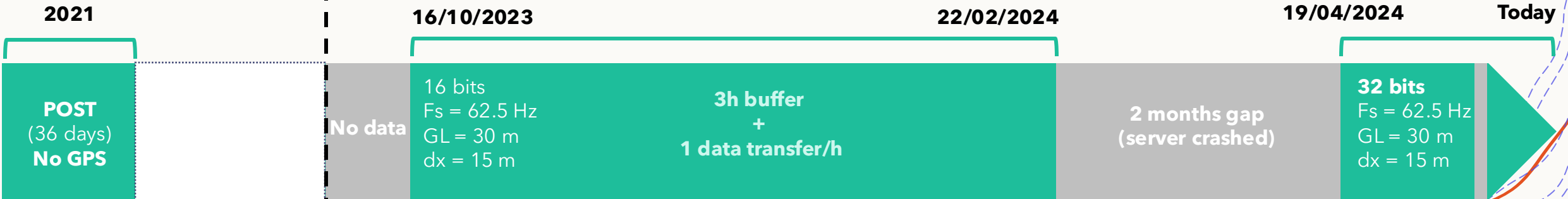
Begining of ABYSS project



SER.S



CCN.N



Key Features of das

- + Seamless manipulation of any multi-file DAS datasets
- + Larger-than-memory computing (online processing)
- + Easy and Extensible (similar to Numpy/Scipy/Xarray)
- + Efficient signal processing routines (multi-threading)
- + Real-time capabilities

Xdas workflow

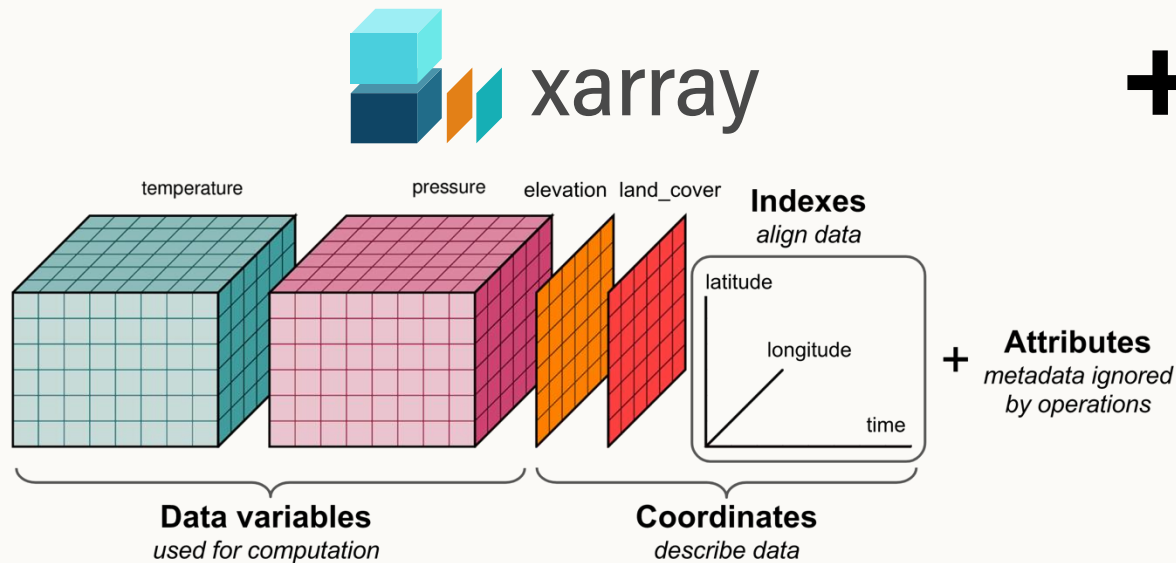


Xdas workflow



Sources of Inspiration

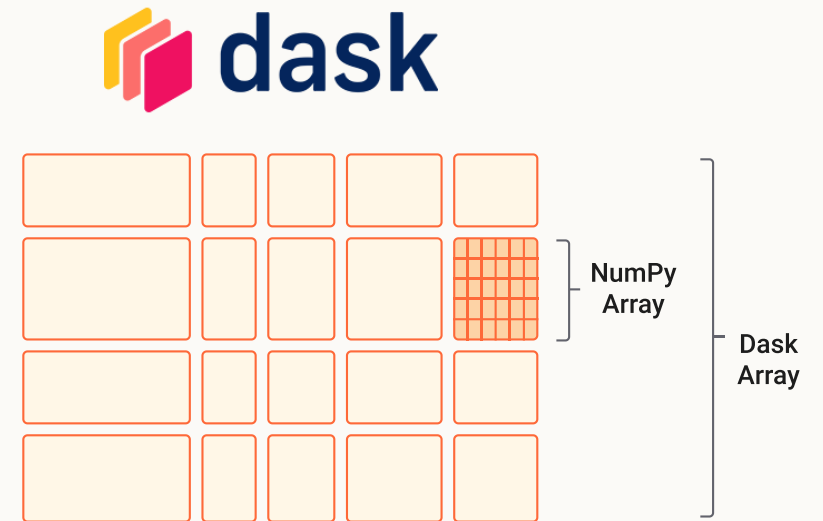
N-D labeled arrays (DataArray)



Only support **dense** coordinates

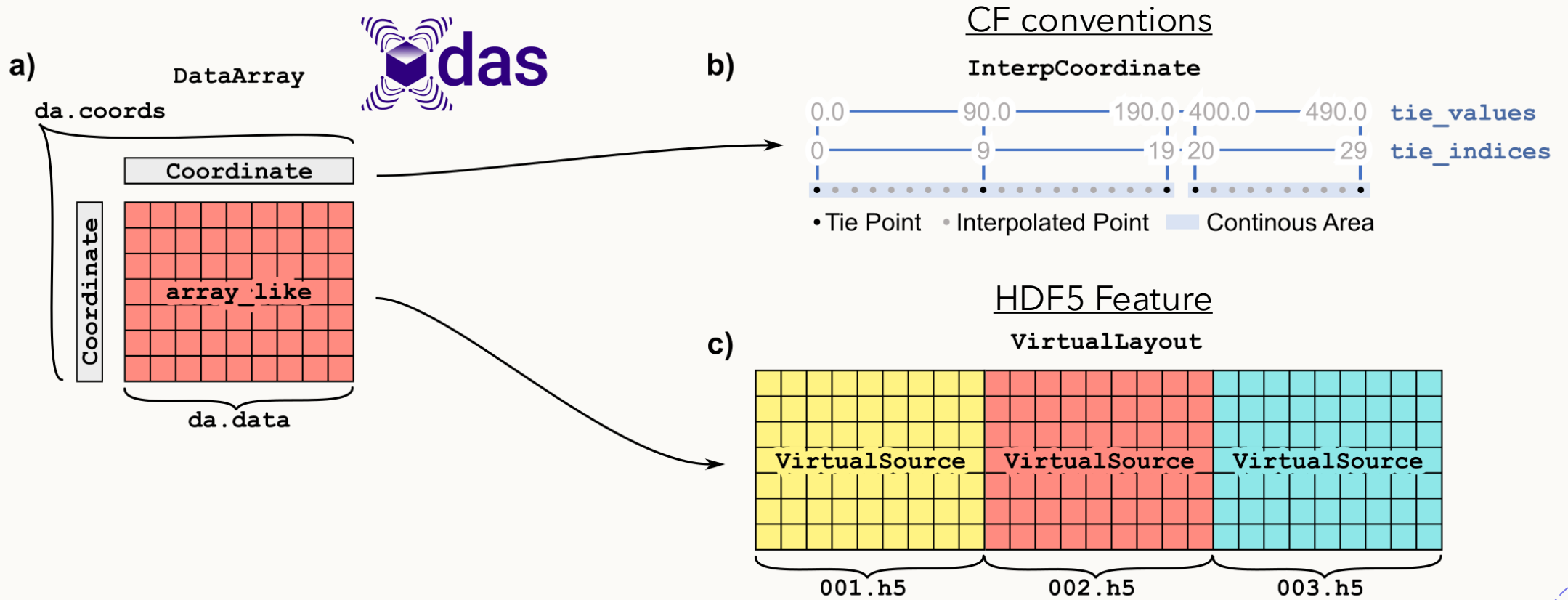
→ Reimplement a subset of Xarray with **interpolated** coordinates and **virtual** dataset

Chunked processing



One task object per chunk make **enormous redundant graphs**

Data structure: DataArray



Code Example: Link files

```
import xdas as xd

# Link virtual dataset
da = xd.open_mfdataarray("abyss/CCN/N/20240101/*.hdf5", engine="asn", tolerance=None)

<xdas.DataArray (time: 5400000, distance: 10000)>
VirtualStack: 100.6G (int16)
Coordinates:
  * time (time): 2024-01-22T00:00:07.000 to 2024-01-23T00:00:06.984
  * distance (distance): 0.0 to 153179.709

# save consolidation
da.to_netcdf("da.nc")
```

Code Example: Label based selection

```
da = da.sel(  
    time=slice("2024-01-22T22:38:30", "2024-01-22T22:39:00"),  
    distance=slice(None, 120_000),  
)
```

```
<xdas.DataArray (time: 1875, distance: 7834)>
```

```
VirtualStack: 28.0M (int16)
```

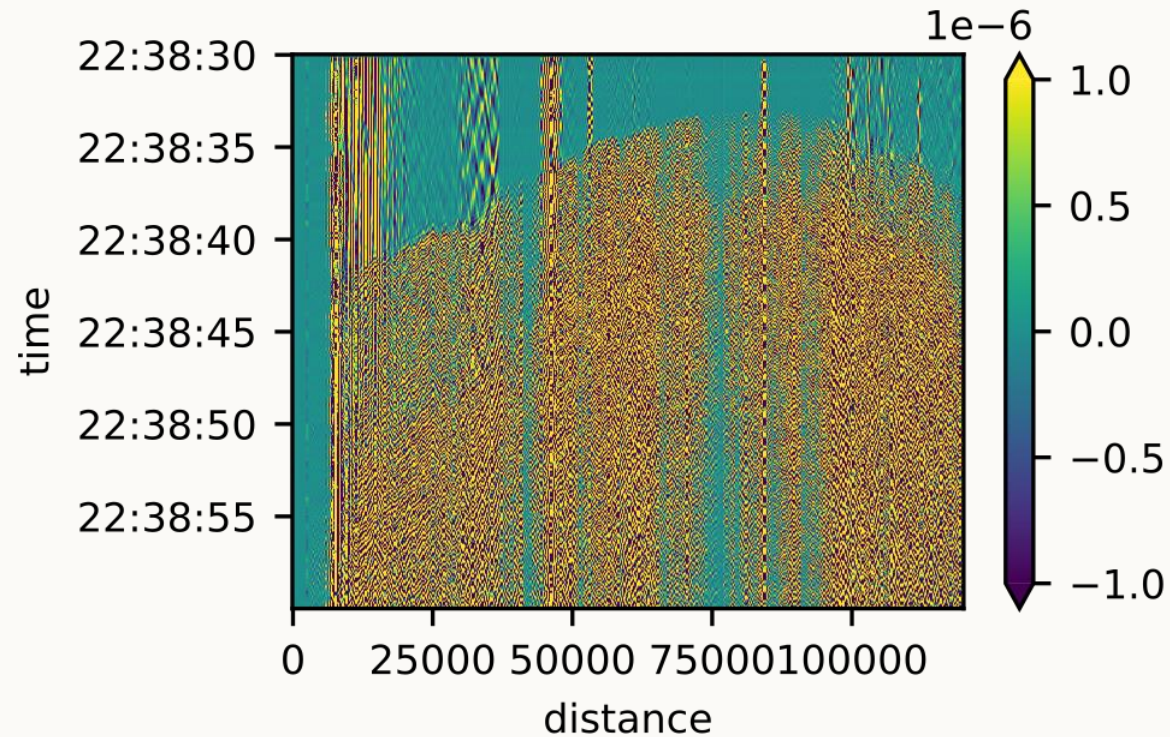
```
Coordinates:
```

```
* time (time): 2024-01-22T22:38:30.008 to 2024-01-23T22:38:30.992
```

```
* distance (distance): 0.0 to 119997.6662
```

Code Example: Plotting

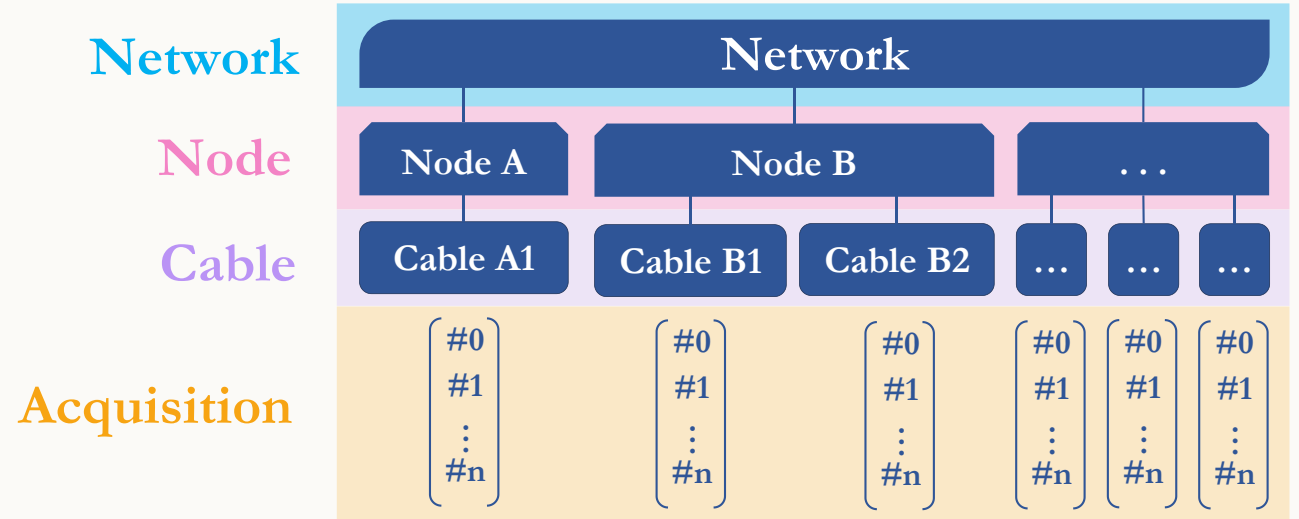
```
da.plot(yincrease=False, vmin=-1e-6, vmax=1e-6)
```



Data structure: DataCollection

Hierarchy of DataArray objects

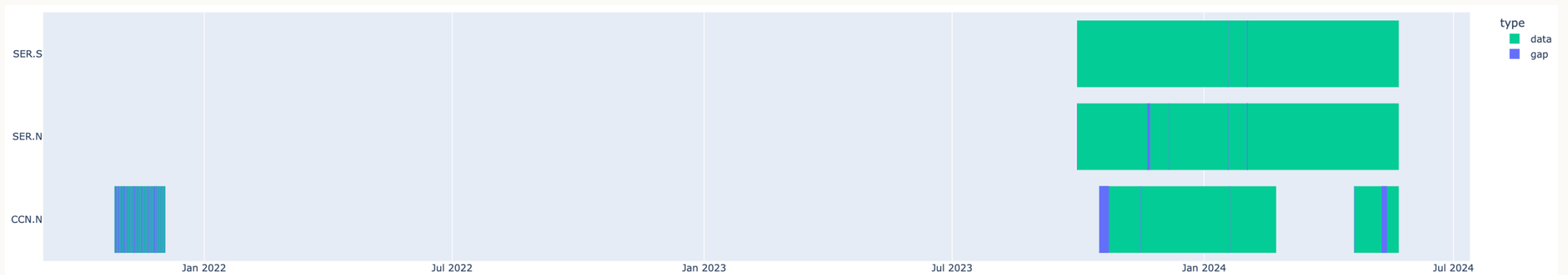
- + Multiple instruments
- + Catalog of events
- + Inhomogeneous acquisitions



Code Example: data availability

```
import xdas as xd
```

```
dc = xd.open_datacollection("abyss.nc")  
xd.plot_availability(dc)
```

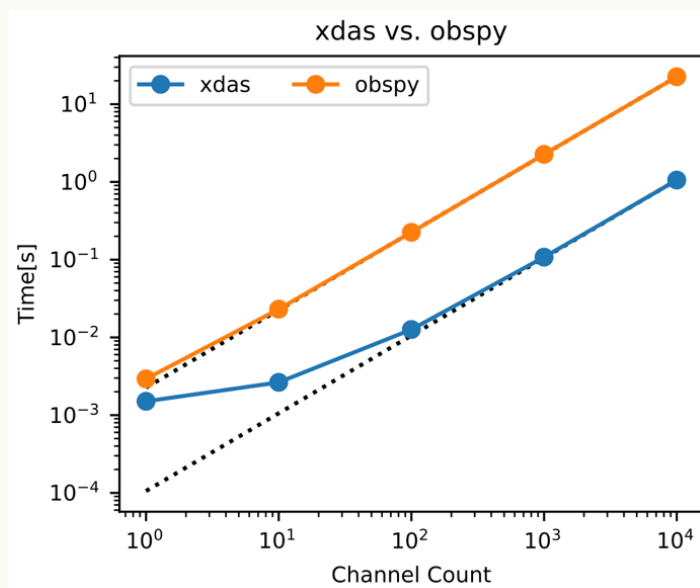


Xdas workflow

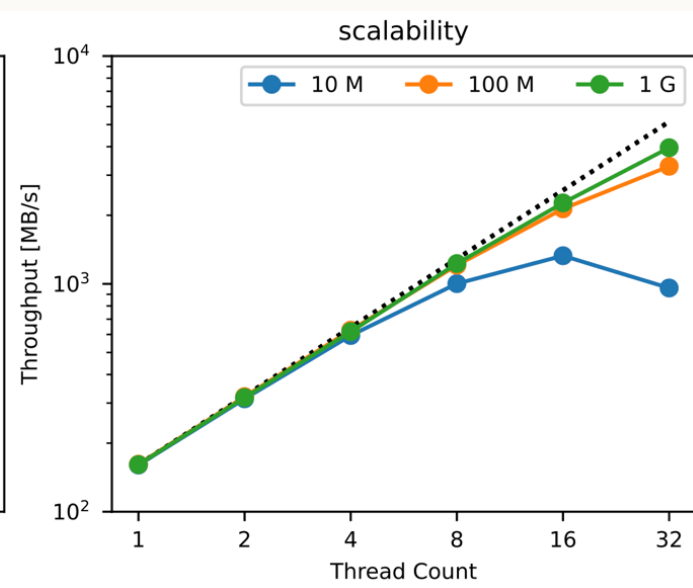


Multi-threaded signal processing routines

- + Implemented:
scipy.signal and
scipy.fft routines
- + Target:
Single machine
servers
- + Strategy:
split & join



100x faster than obspy*



10x faster than scipy*

Code Example: signal processing

```
import xdas.signal as xs

# apply gain
da *= 8.36e-11

# convert to velocity
da = xs.integrate(da, dim="distance")
da = xs.sliding_mean_removal(da, wlen=1000.0)

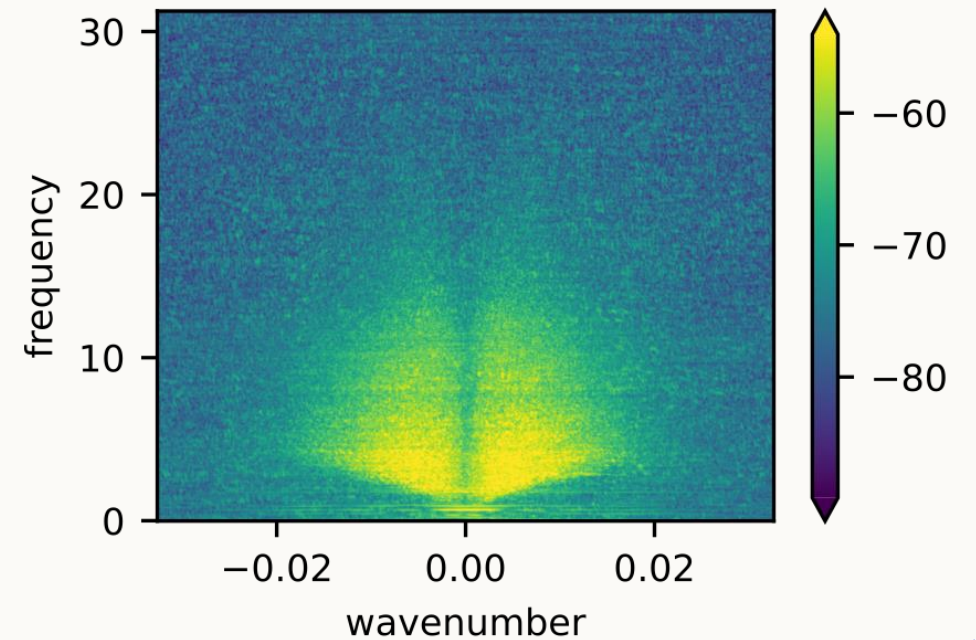
# decimate in time and space and plot
da = xs.decimate(da, 16, ftype="fir", dim="distance")
da = xs.decimate(da, 4, ftype="iir", dim="time")
```


Code Example: FFT processing

```
import xdas.fft as xfft

# compute FK representation
fk = xs.taper(da, dim="distance")
fk = xs.taper(fk, dim="time")
fk = xfft.rfft(fk, dim={"time": "frequency"})
fk = xfft.fft(fk, dim={"distance": "wavenumber"})
fk = 20 * np.log10(np.abs(fk)) # numpy ufuncs

# plot
fk.plot(robust=True, interpolation="antialiased")
```

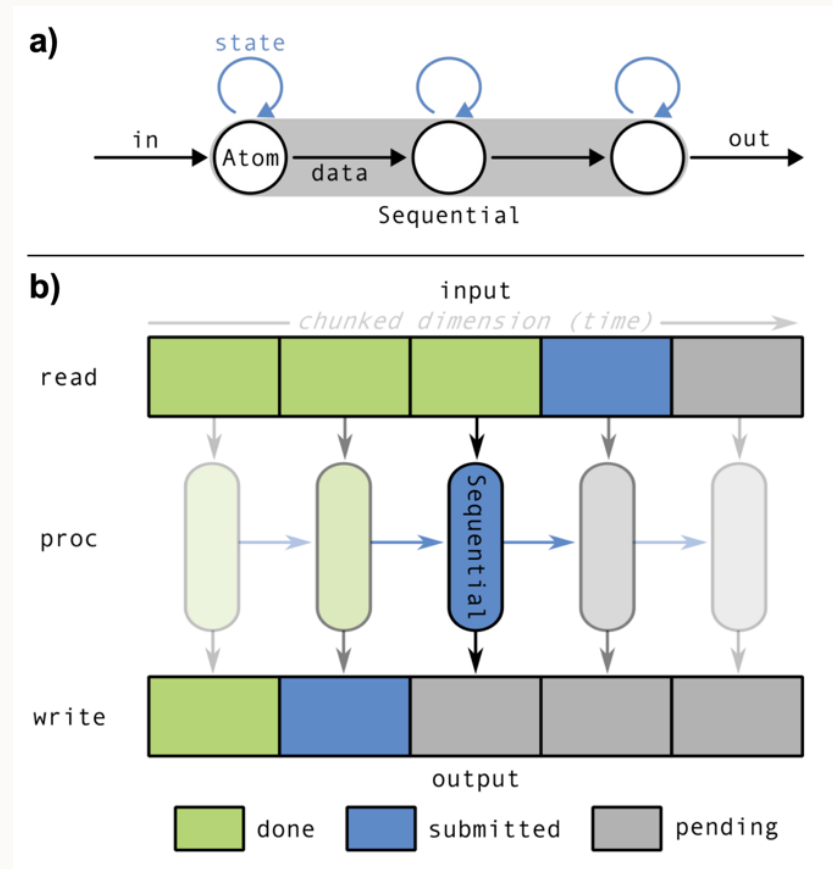


Xdas workflow



Online processing

- + Strategy:
Pipes & Filters
- + Application:
Larger-than-memory &
Real-time
- + Model:
Atom objects with state
memory/passing



Code Example: Atoms

```
import xdaskit.atoms as xa

atom = xa.Sequential(
    [
        xs.taper(..., dim="distance"), # replace input by ...
        xa.DownSample(2, dim="time"), # use stateful atoms
        np.square, # already unary in/out operation
    ]
)

result = atom(da)
```

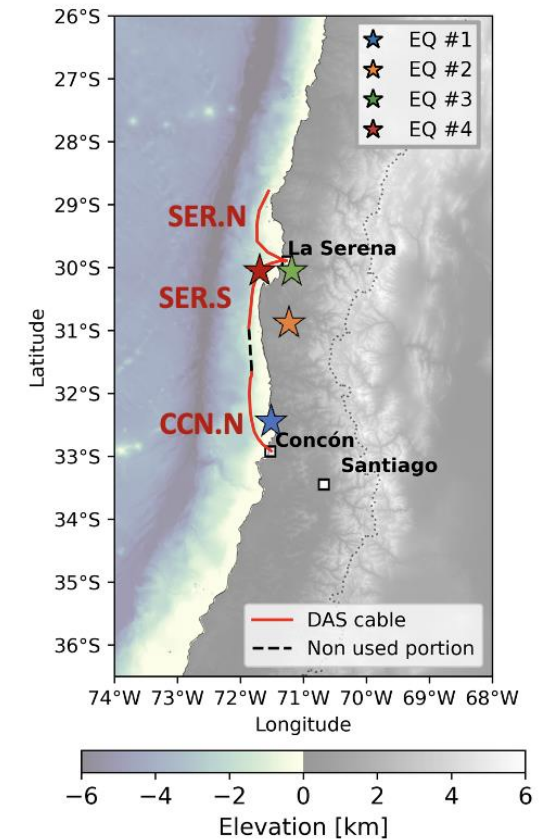
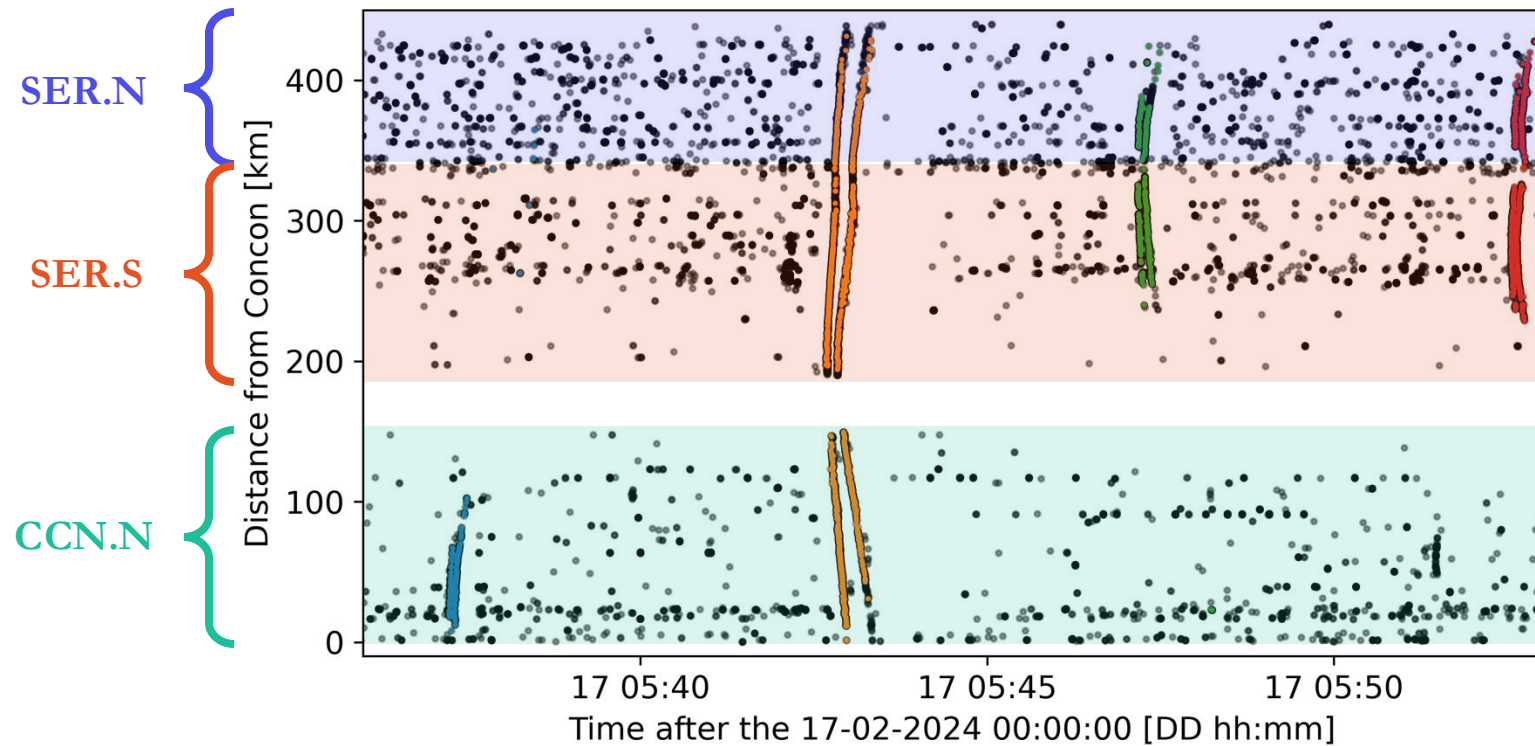
Code Example: Massive processing

```
import xdas.processing as xp

dl = xp.DataArrayLoader(da, chunks={"time": 10_000})
dw = xp.DataArrayWriter(path="outputs/")

result = xp.process(atom, dl, dw)
```

Use case: ML picking

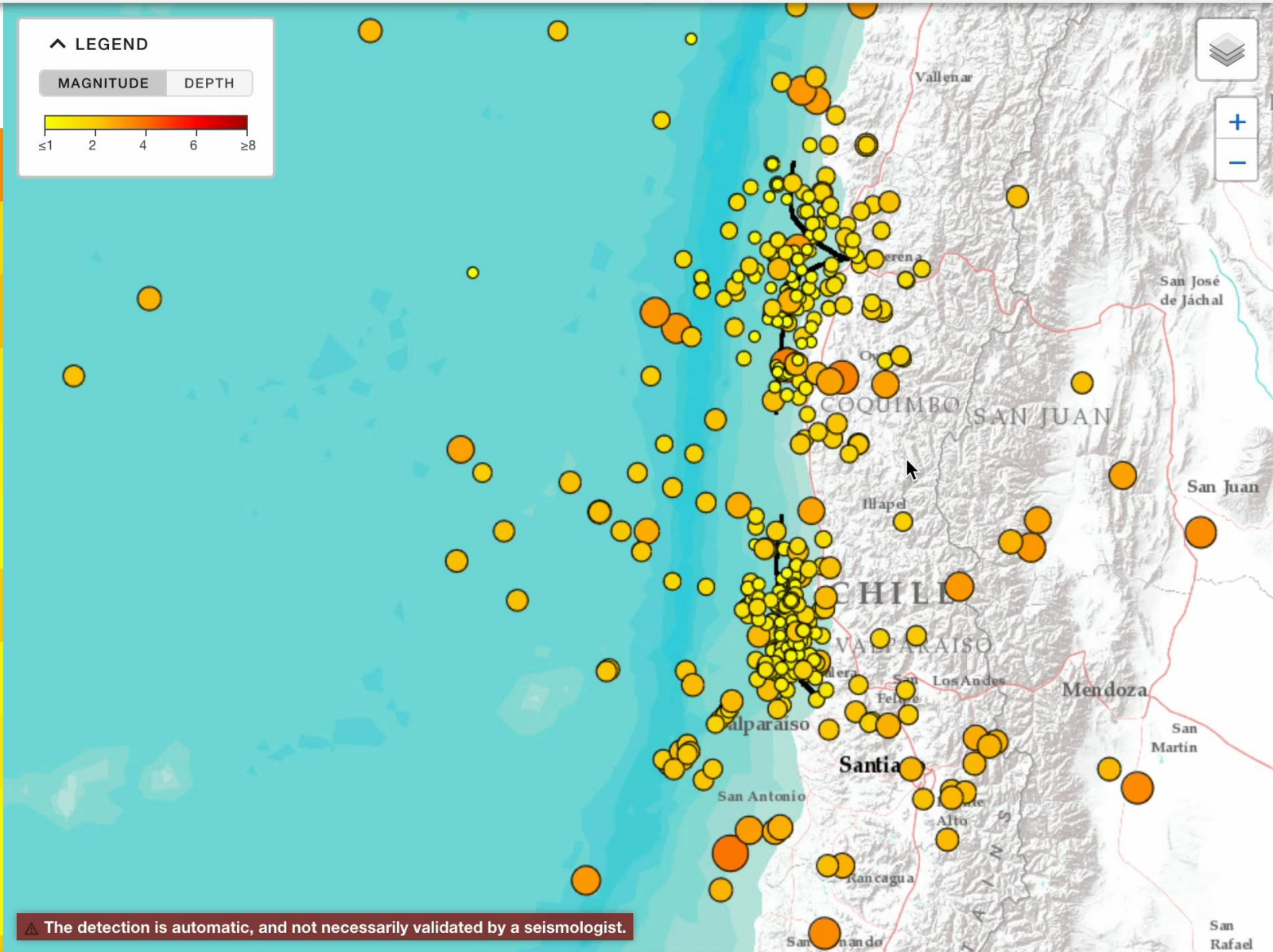
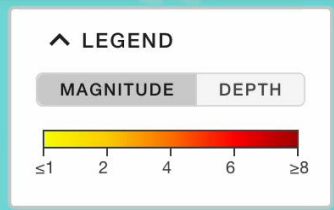


- Home
- Catalog
- Stations
- Live

Events in the study area (last 7 days)
Event time in UTC

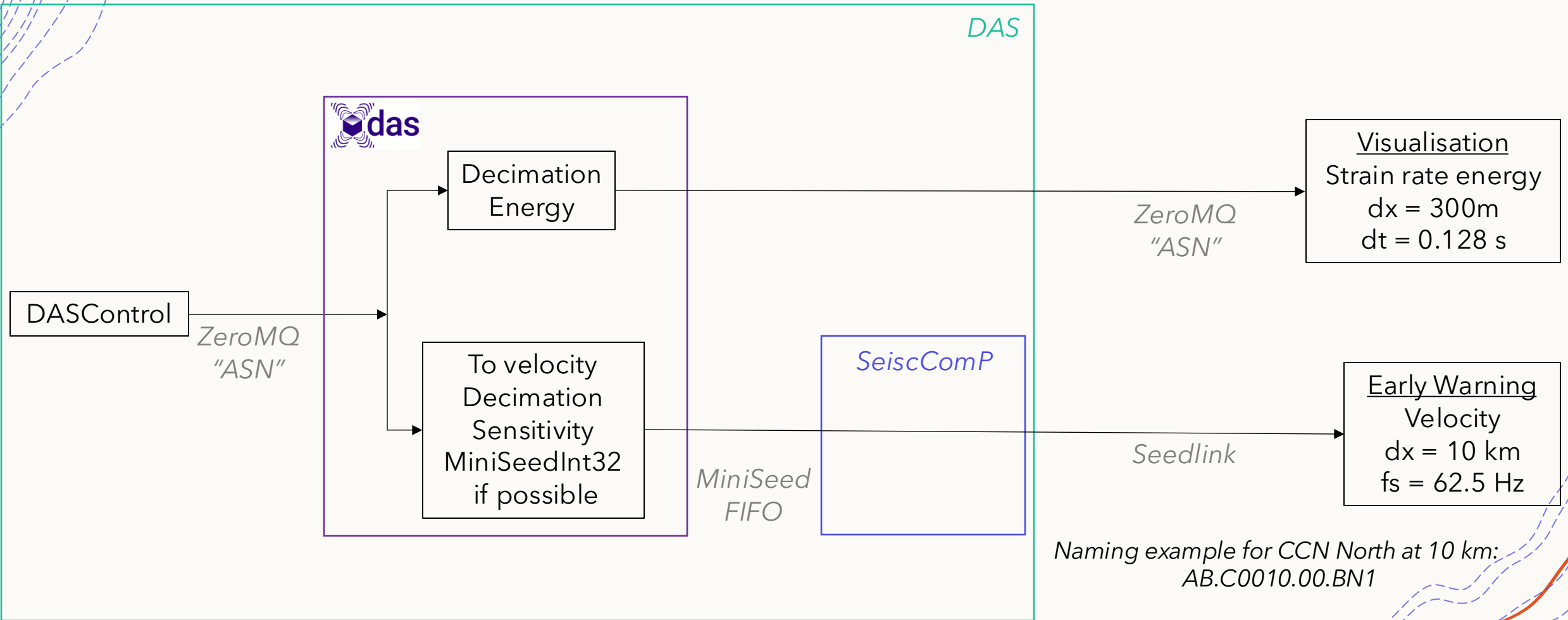
MAG

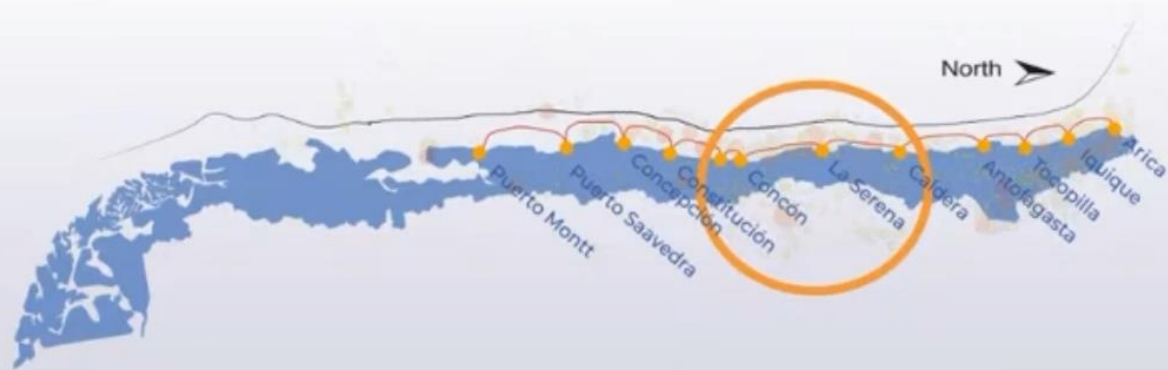
3.4	2024-10-10 23:57:07 (14h 5... MENDOZA, ARGENTINA	
1.7	2024-10-10 23:33:12 (15h 1... OFF THE COAST OF CENTR...	
2.4	2024-10-10 19:55:56 (18h 5... NEAR THE COAST OF CENT...	
1.5	2024-10-10 15:23:37 (23h 2... NEAR THE COAST OF CENT...	
1.3	2024-10-10 15:07:02 (23h 4... OFF THE COAST OF CENTR...	
1.2	2024-10-10 15:05:22 (23h 4... NEAR THE COAST OF CENT...	
2.1	2024-10-10 15:03:56 (23h 4... OFF THE COAST OF CENTR...	
1.3	2024-10-10 14:39:41 (1j 11m) NEAR THE COAST OF CENT...	
1.5	2024-10-10 14:22:00 (1j 28m) NEAR THE COAST OF CENT...	
1.2	2024-10-10 14:15:14 (1j 35m) NEAR THE COAST OF CENT...	
1.2	2024-10-10 13:26:04 (1j 1h ... NEAR THE COAST OF CENT...	



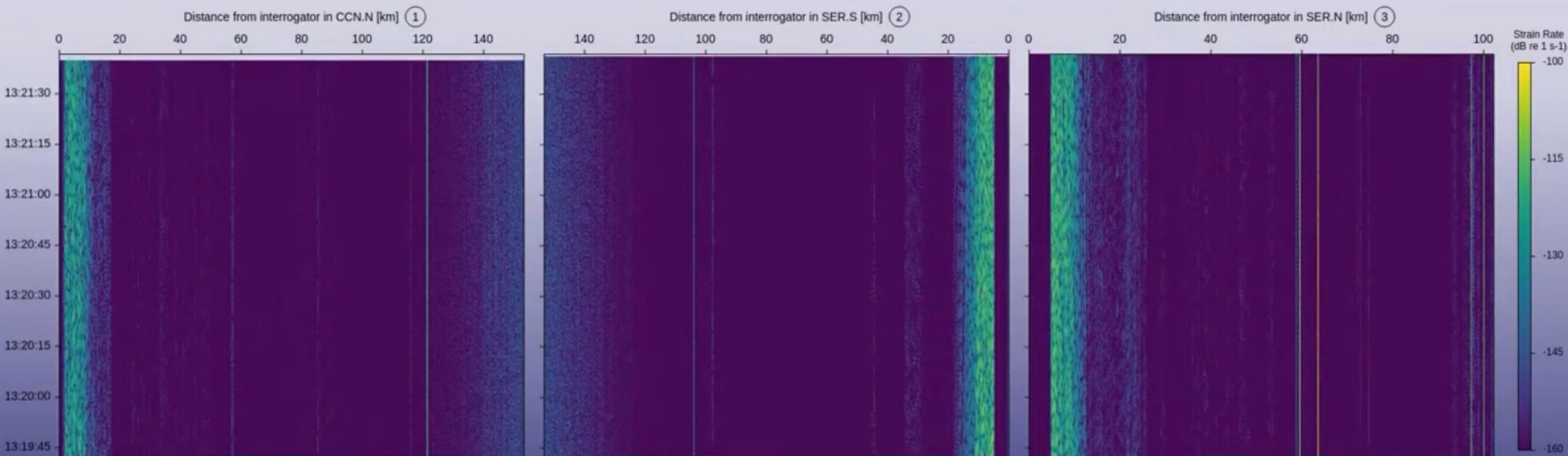
The detection is automatic, and not necessarily validated by a seismologist.

Real-time streaming

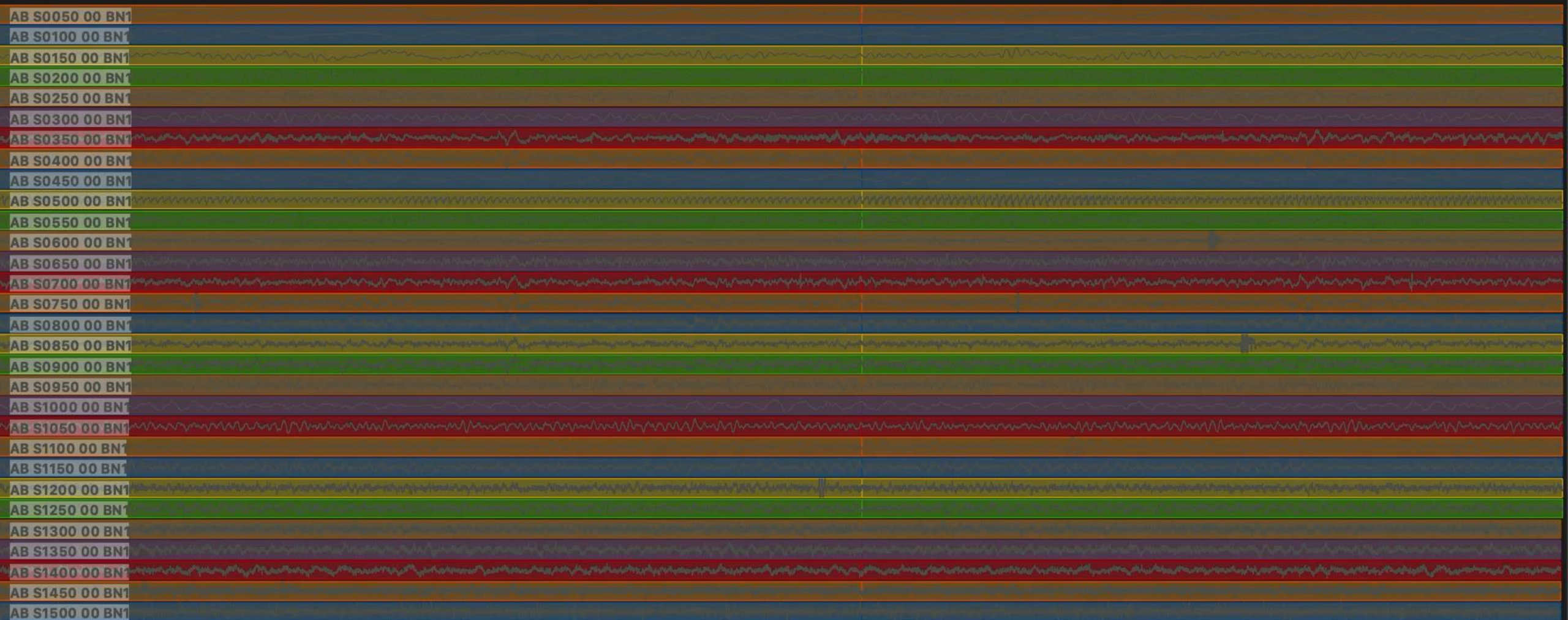




Live stream of seismic movement in **central Chile** along submarine fiber optic cables



Snuffler



15:48:30

15:49:00

15:49:30

15:50:00

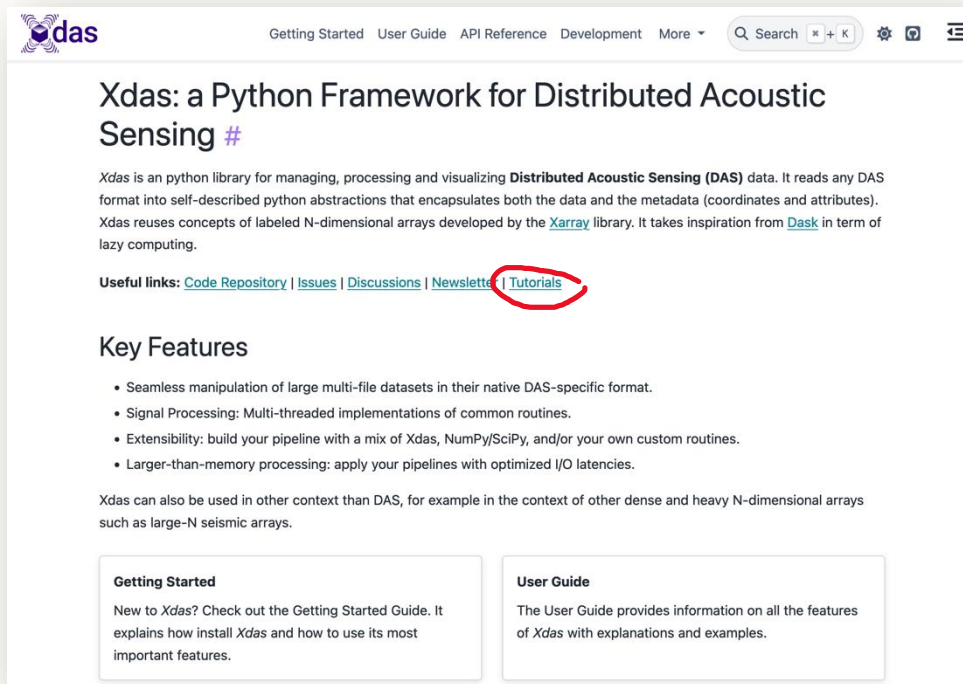
15:50:30

15:51:00

Nov 18, 2024

Resources

Documentation



The screenshot shows the Xdas documentation website. At the top, there is a navigation bar with links for 'Getting Started', 'User Guide', 'API Reference', 'Development', and 'More'. A search bar is also present. The main heading is 'Xdas: a Python Framework for Distributed Acoustic Sensing #'. Below this, a paragraph describes Xdas as a Python library for managing, processing, and visualizing DAS data. A list of 'Useful links' includes 'Code Repository', 'Issues', 'Discussions', 'Newsletters', and 'Tutorials', with 'Tutorials' circled in red. A 'Key Features' section lists several bullet points. At the bottom, there are two boxes: 'Getting Started' and 'User Guide', each with a brief description.

Xdas: a Python Framework for Distributed Acoustic Sensing #

Xdas is a python library for managing, processing and visualizing **Distributed Acoustic Sensing (DAS)** data. It reads any DAS format into self-described python abstractions that encapsulates both the data and the metadata (coordinates and attributes). Xdas reuses concepts of labeled N-dimensional arrays developed by the [Xarray](#) library. It takes inspiration from [Dask](#) in term of lazy computing.

Useful links: [Code Repository](#) | [Issues](#) | [Discussions](#) | [Newsletters](#) | [Tutorials](#)

Key Features

- Seamless manipulation of large multi-file datasets in their native DAS-specific format.
- Signal Processing: Multi-threaded implementations of common routines.
- Extensibility: build your pipeline with a mix of Xdas, NumPy/SciPy, and/or your own custom routines.
- Larger-than-memory processing: apply your pipelines with optimized I/O latencies.

Xdas can also be used in other context than DAS, for example in the context of other dense and heavy N-dimensional arrays such as large-N seismic arrays.

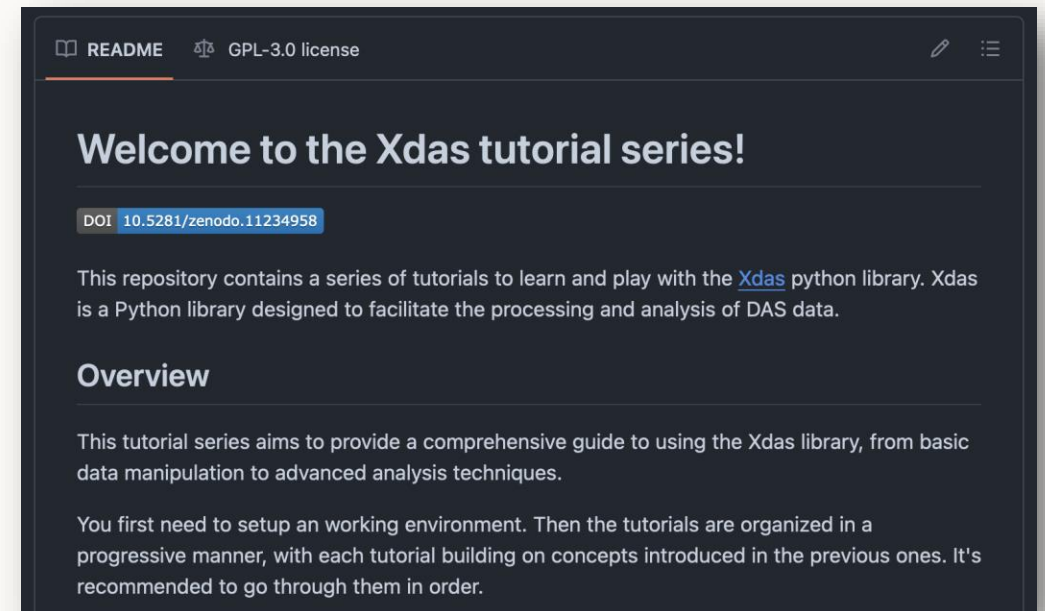
Getting Started

New to Xdas? Check out the Getting Started Guide. It explains how install Xdas and how to use its most important features.

User Guide

The User Guide provides information on all the features of Xdas with explanations and examples.

Tutorials



The screenshot shows the README for the Xdas tutorial series. It features a dark theme. At the top, there are links for 'README' and 'GPL-3.0 license'. The main heading is 'Welcome to the Xdas tutorial series!'. Below this, a DOI is provided: 'DOI 10.5281/zenodo.11234958'. A paragraph explains that the repository contains a series of tutorials to learn and play with the Xdas python library. An 'Overview' section follows, stating that the tutorial series aims to provide a comprehensive guide to using the Xdas library, from basic data manipulation to advanced analysis techniques. The final paragraph mentions that users first need to setup a working environment and that the tutorials are organized in a progressive manner.

README [GPL-3.0 license](#)

Welcome to the Xdas tutorial series!

DOI [10.5281/zenodo.11234958](#)

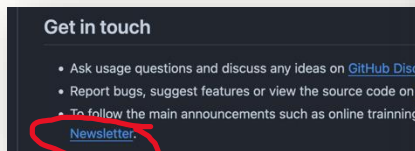
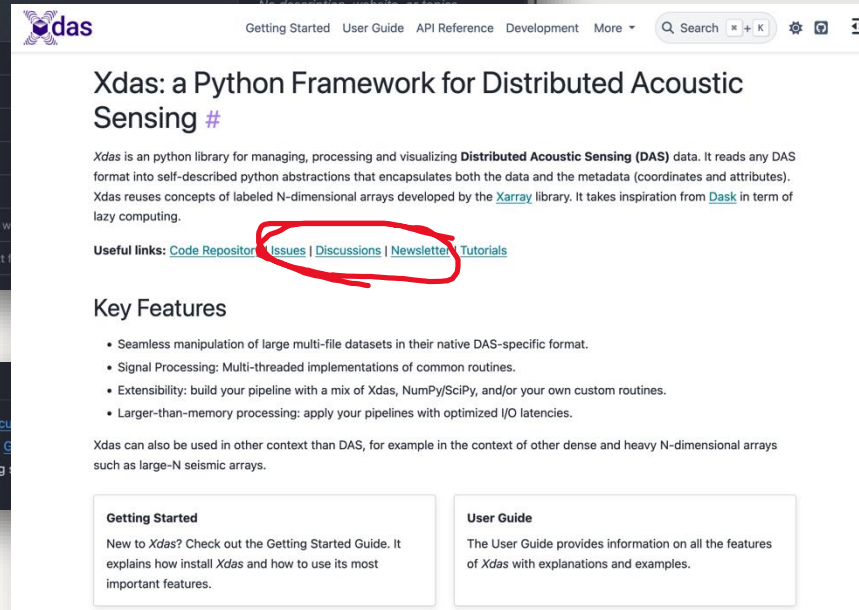
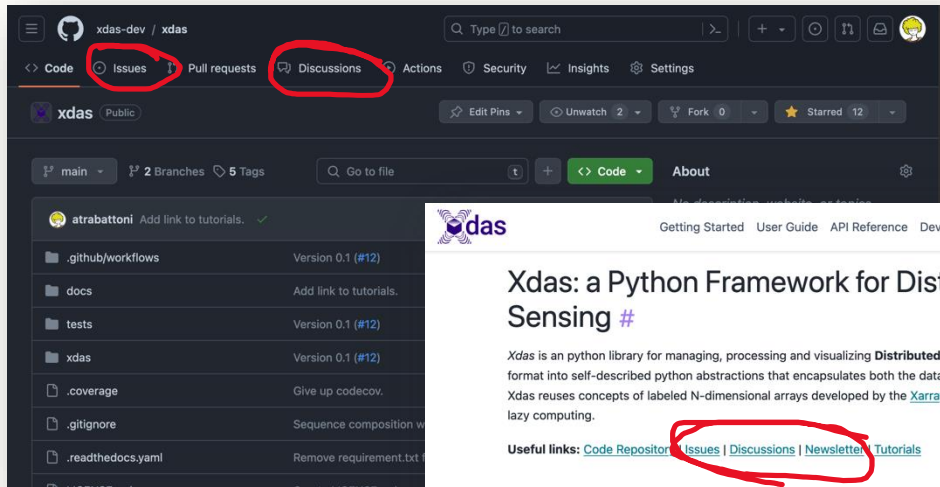
This repository contains a series of tutorials to learn and play with the [Xdas](#) python library. Xdas is a Python library designed to facilitate the processing and analysis of DAS data.

Overview

This tutorial series aims to provide a comprehensive guide to using the Xdas library, from basic data manipulation to advanced analysis techniques.

You first need to setup an working environment. Then the tutorials are organized in a progressive manner, with each tutorial building on concepts introduced in the previous ones. It's recommended to go through them in order.

Get in touch

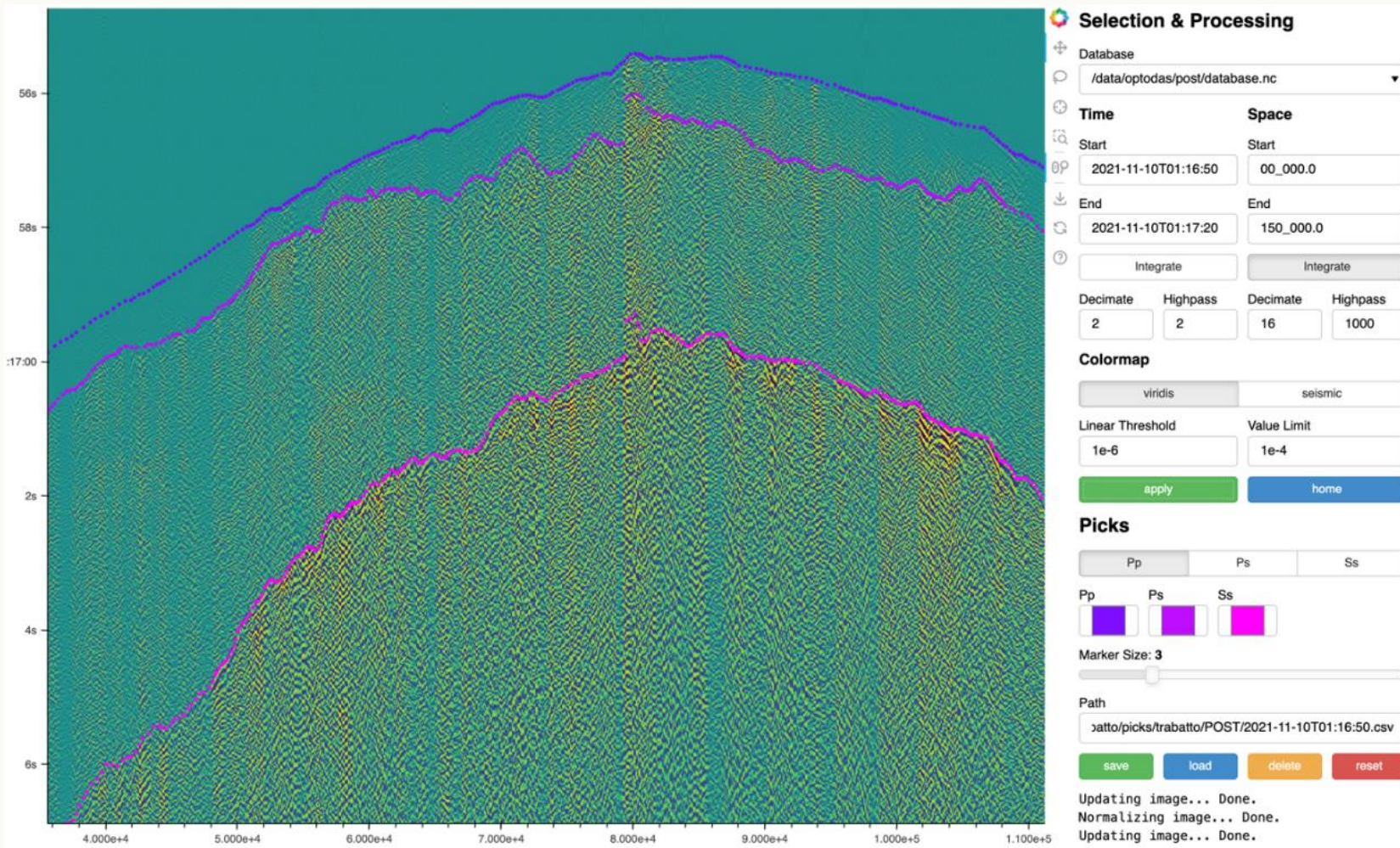


- **Newsletter:** get the main announcements.
- **Discussions:** ask questions, share ideas.
- **Issues:** Report a bug, request a feature.

Two ways to contribute

1. Join the team and improve Xdas
2. Create your toolbox within the xdas-dev github organization

Additional toolboxes: Xpick





Questions? Comments?
Requests? Ideas?