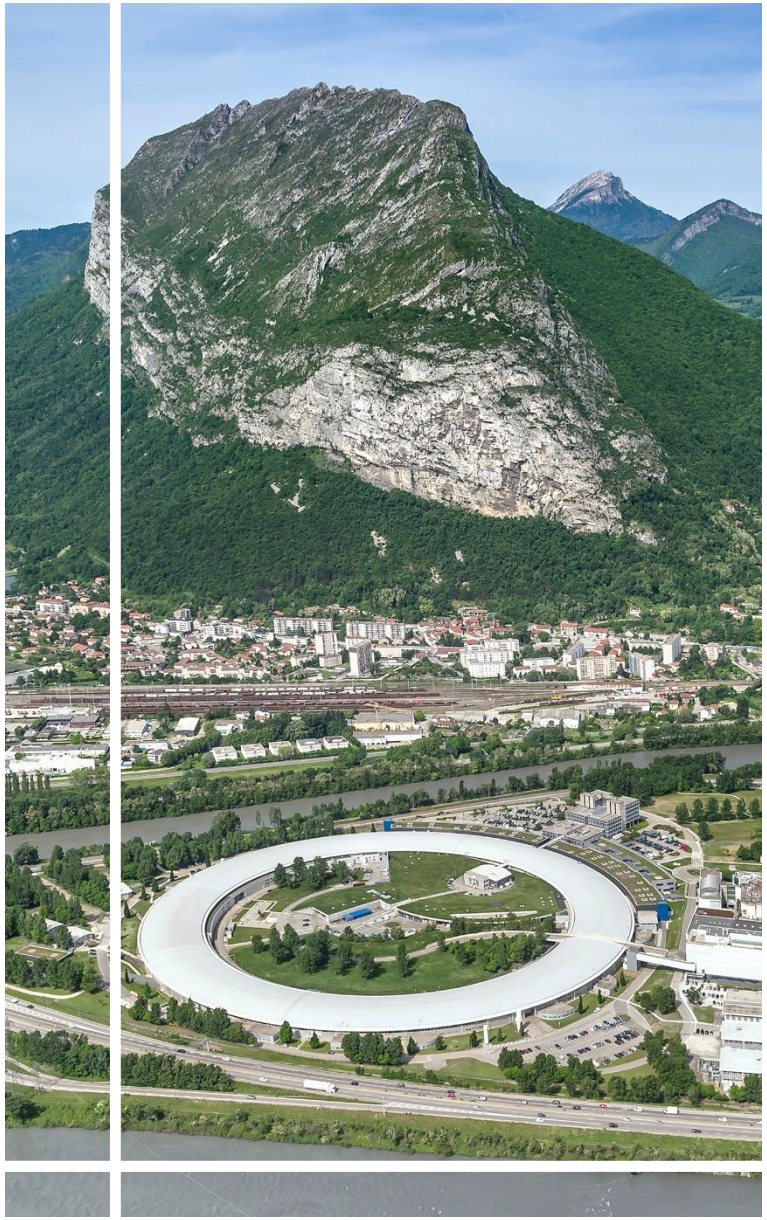
ESRF | The European Synchrotron



HDF5 at the ESRF

Wout De Nolf
ESRF Data Analysis Unit
HDF5 writing in BLISS

HDF5 AT THE ESRF: DATA COLLECTION

Transient storage
- Metadata
- 0D/1D data
- 2D URI's

Persistent storage
- Metadata
- 0D to 3D HDF5 datasets

Data collection

https://gitlab.esrf.fr/bliss/bliss/

URI's

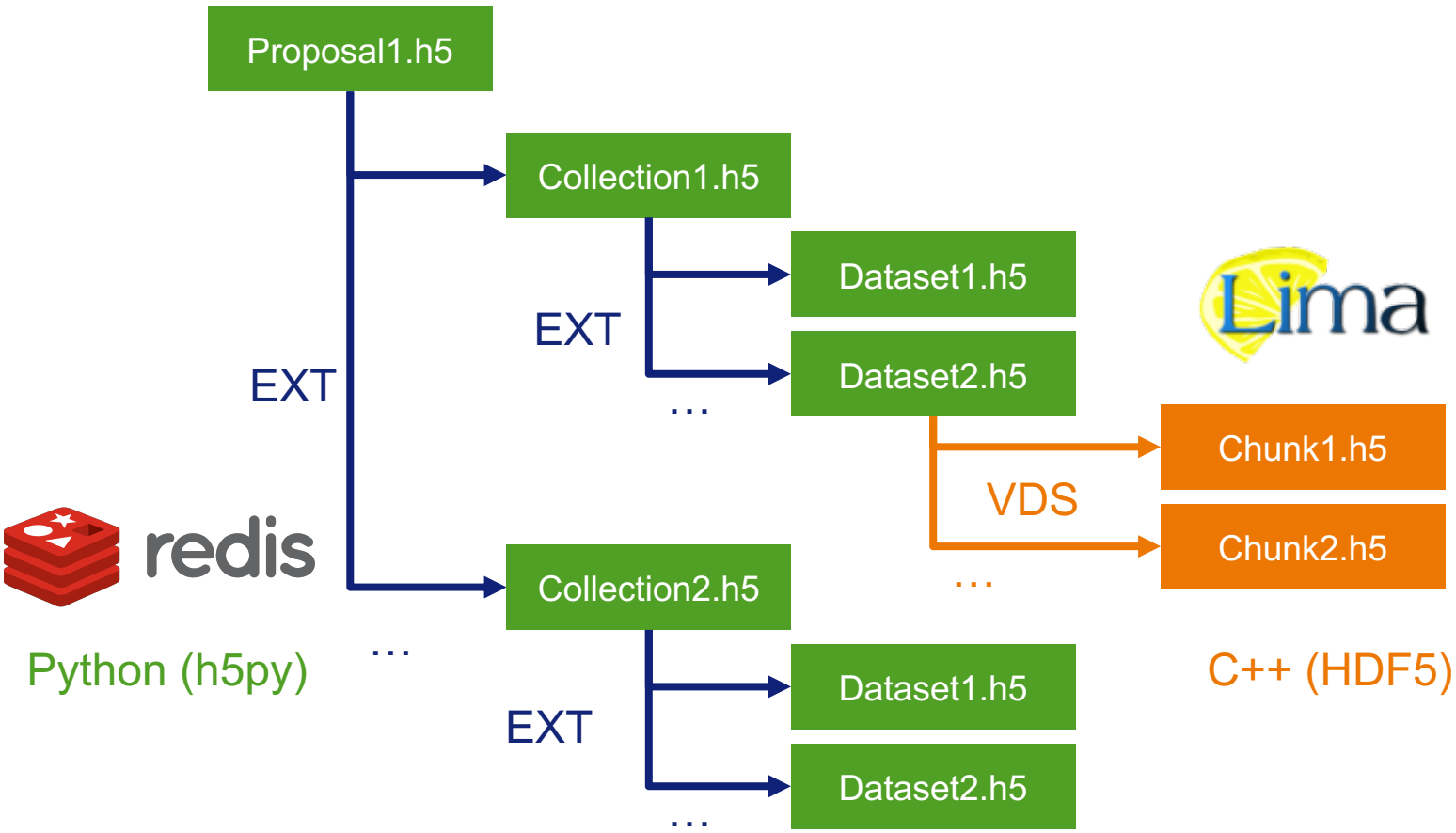
VDS

Transient storage
- Metadata
- 2D data

https://github.com/esrf-bliss/Lima

Distributed file system
- Writing: GPFS, NFS
- Reading: GPFS, NFS, SMB

On Distributed File System



HDF5 AT THE ESRF: DATA COLLECTION

NeXus Data Format

Collection1.h5

Dataset1.h5

Name	Description	Type
sample_0006.h5		NXroot
1.1	"ascan robx 0 3 5 0.03"	NXentry
2.1	"ct 0.02"	NXentry
3.1	"amesh robx -1 2 5 roby 0.5 3 6 0.03"	NXentry
4.1	"amesh robx -1 2 6 roby 0.5 3 4 0.03"	NXentry
5.1	"loopscan"	NXentry
6.1	"ct 0.01"	NXentry
7.1	"amesh robx -1 2 5 roby 0.5 3 6 0.03"	NXentry
8.1	"loopscan"	NXentry
9.1	"amesh robx -1 2 6 roby 0.5 3 4 0.02"	NXentry
10.1	"amesh robx -1 2 6 roby 0.5 3 3 0.03"	NXentry

…

External Links

"SCANS"

HDF5 AT THE ESRF: DATA COLLECTION

instrument		NXinstrument	
diode2alias		NXdetector	
diode3		NXdetector	
diode4		NXdetector	
diode5		NXdetector	
diode6		NXdetector	
diode7		NXdetector	
diode8		NXdetector	
diode9alias		NXdetector	
elapsed_time		NXpositioner	
epoch		NXpositioner	
lima_simulator		NXdetector	
lima_simulator2		NXdetector	
acq_parameters		NXcollection	
ctrl_parameters		NXcollection	
data	3D data	uint32	6 × 1024 × 1024
type	"lima"	string	scalar
lima_simulator2_bpm		NXdetector	
acq_time	Compressed 1D d		6
fwhm_x	Compressed 1D d		6
fwhm_y	Compressed 1D data	float64	6
intensity	Compressed 1D data	float64	6
type	"lima"	string	scalar
x	Compressed 1D d		6
y	Compressed 1D d		6
lima_simulator2_roi1		NXdetector	
avg	Compressed 1D data	float64	6
data	Compressed 1D data	float64	6
max	Compressed 1D data	float64	6
min	Compressed 1D data	float64	6
selection		NXcollection	
std	Compressed 1D data	float64	6
type	"lima"	string	scalar

Parameters and settings

External data

Internal data

VDS

External data: 3D

Chunk1.h5

Chunk2.h5

…

Lima

C++ (HDF5)

Internal data: 0D, 1D, 2D

redis

Python (h5py)

HDF5 AT THE ESRF: DATA COLLECTION

HDF5 features used for data collection

- Vanilla HDF5 (Groups, Datasets, Attributes, Softlinks)
- External Links (EXT)
- Virtual DataSets (VDS)
- Variable length data types: only for scalar strings
- Growing datasets during acquisition
- Chunking and compression

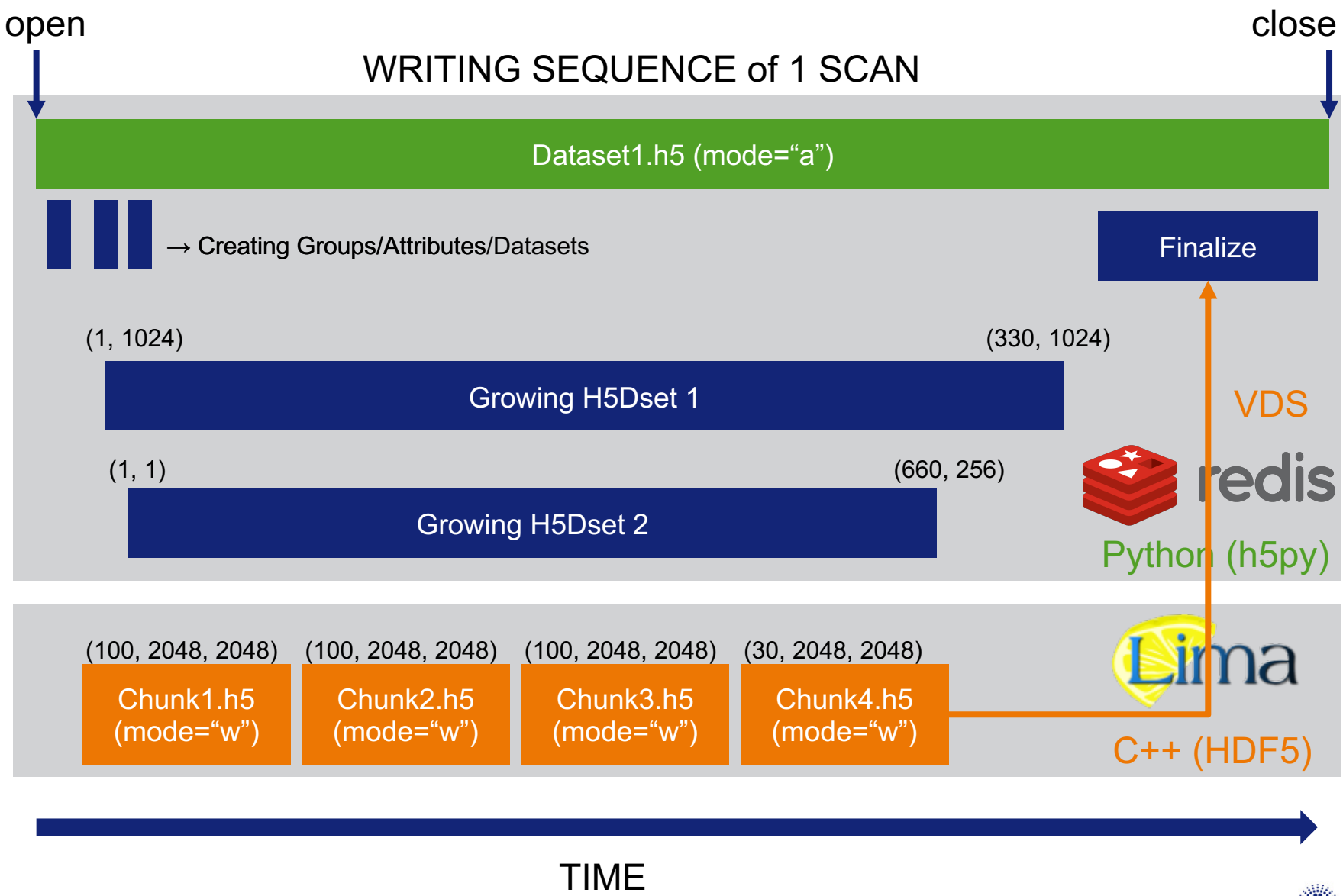
No SWMR
No Parallel HDF5

Distributed file system

- Writing: GPFS, NFS
- Reading: GPFS, NFS, SMB

No control over readers and their access mode

HDF5 AT THE ESRF: DATA COLLECTION



HDF5 AT THE ESRF: DATA COLLECTION

Why we don’t use SWMR

- Non-POSIX file systems (NFS, SMB)

- No control over the readers
  - Writer: open – SWMR ON – SWMR OFF – close
  - Reader: can come in at any time in any mode

- POV of the writer architecture: you may not know whether you have created all necessary groups, datasets and attributes so you can never enable SWMR

- Many readers are stuck on HDF5 1.8.x (Matlab, IDL, …)

HDF5 AT THE ESRF: DATA COLLECTION

Current situation at the ESRF

Writer: file open in append mode for the duration of 1 scan

Readers:

1. HDF5_USE_FILE_LOCKING=TRUE (mode=<any>)
   - File will not be corrupted
   - Prevent a writer/acquisition from starting
   - No access during writing

2. HDF5_USE_FILE_LOCKING=FALSE (mode="r")
   - File will not be corrupted
   - Never influences the writer/acquisition
   - Capture+retry exceptions or segfaults

http://www.silx.org/doc/silx/latest/Tutorials/io.html#concurrent-hdf5

➡ Warning: changing an environment variable affects the entire process

HDF5 AT THE ESRF: DATA COLLECTION

Current situation at the ESRF

Data corruption (no ACID guarantees)

1. Disk full
2. Network issues
3. Writer crash

Salvage corrupt files:

Pure Python HDF5 reader to browse the file and skip corrupt sections
https://github.com/jjhelmus/pyfive
Use h5py to reader the actual datasets

HDF5 AT THE ESRF: DATA COLLECTION

Priorities for HDF5 development:

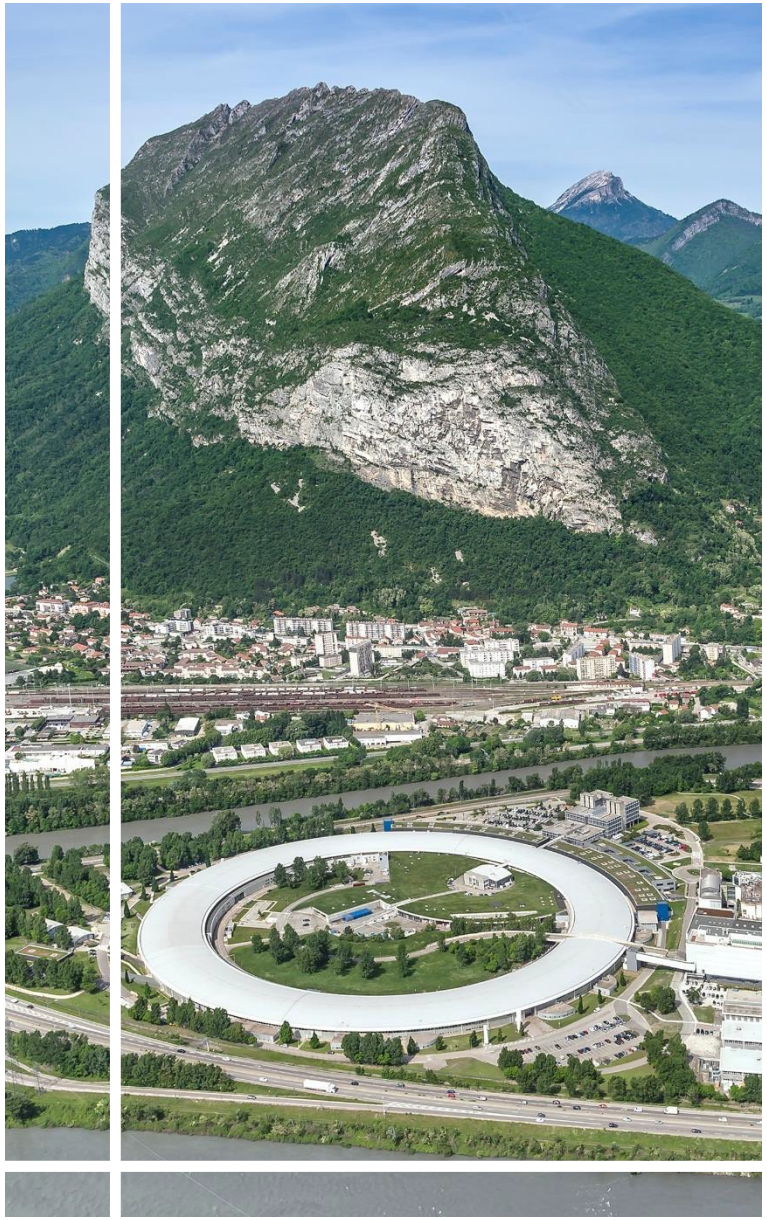
1. VFD SWMR (!!!!!!!)
   - Supported on NFS/SMB?
   - Writing can be done regardless of readers and their access mode?
   - Support for virtual datasets?
   - Support variable length data types (scalar strings only) ?

2. Growing Virtual Datasets (specifically a growing number of external sources, not growing external sources)

3. Tool for recovering data from corrupt files

4. HDF5_USE_FILE_LOCKING from API instead of environment variable

5. Not discussed: parallel writing and why MPI is not used



HDF5 at the ESRF