Simon Ebner :: Paul Scherrer Institut

HDF5 @PSI

ICALEPCS HDF5 Workshop 08.10.2017

… the actual work was done by …

Andrej Babic

Leonardo Sala

Heiner Billich

…

[1]

[1] http://assets.creativity-online.com/images/work/print/p/a/n/Pantone_MinionYellow15.jpg

PAUL SCHERRER INSTITUT

Overview

- Direct Chunk Writing in h5py
- Virtual Data Sets - Performance
- PSI HDF5 File Writer

PAUL SCHERRER INSTITUT

PSI

Direct Chunk Writing - h5py

PAUL SCHERRER INSTITUT

Direct Chunk Writing - h5py

Available with h5py version 2.7.0

https://github.com/h5py/h5py/releases/tag/2.7.0

Direct Chunk Writing - h5py

```
filename = "test_direct_chunk_write.hdf5"
filehandle = h5py.File(filename, "w")
dataset = filehandle.create_dataset("data", (100, 100, 100), maxshape=(None, 100,
100), chunks=(1,100,100), dtype='float32')

# Create test array
array = numpy.random.rand(100, 100)
array = array.astype('float32')

# Direct chunk write
index = 0
dataset.id.write_direct_chunk((index, 0, 0), array.tobytes(), filter_mask=1)

# File need to be closed to have the changes been visible
filehandle.close()
```

*https://github.com/simongregorebner/hdf5_examples/blob/master/HDF5%20Direct%20Chunk%20Write.ipynb

PAUL SCHERRER INSTITUT PSI

Direct Chunk Writing - h5py

```
filename = 'test_direct_chunk_write_bitshuffle.hdf5'
filehandle = h5py.File(filename, "w")

block_size = 2048
dataset = filehandle.create_dataset("data", (100, 100, 100), maxshape=(None, 100, 100),
compression=bitshuffle.h5.H5FILTER, compression_opts=(block_size, bitshuffle.h5.H5_COMPRESS_LZ4),
chunks=(1,100,100), dtype='float32')

# Create random numbers
array = numpy.random.rand(100, 100)
array = array.astype('float32')

def compress_as_chunk(array, block_size=2048):
    compressed_bytes = bitshuffle.compress_lz4(array, block_size)
    bytes_number_of_elements = struct.pack('>q',
(array.shape[0]*array.shape[1]*array.dtype.itemsize))
    bytes_block_size = struct.pack('>i', block_size*array.dtype.itemsize)
    all_bytes = bytes_number_of_elements + bytes_block_size + compressed_bytes.tobytes()
    return all_bytes

compressed = compress_as_chunk(array, block_size=block_size)

# Direct chunk write
index = 0
dataset.id.write_direct_chunk((index, 0, 0), compressed)

filehandle.close()
```

*https://github.com/simongregorebner/hdf5_examples/blob/master/HDF5%20Direct%20Chunk%20Write%20Bitshuffle.ipynb

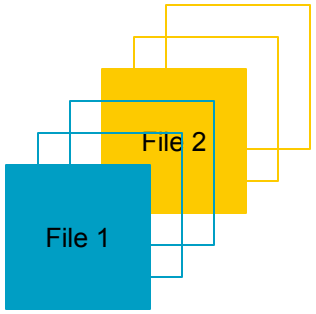
PAUL SCHERRER INSTITUT

PSI

Virtual Data Sets - Performance

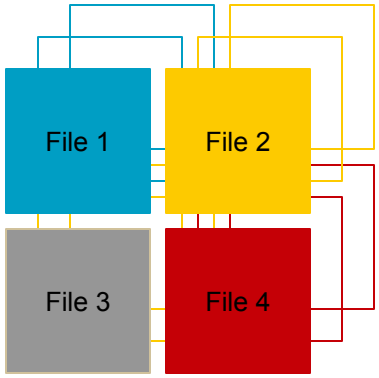
Virtual Data Sets - Performance

Time Series Chunking



Usable

Image Quadrant Chunking



Not Usable

Up to **240%** performance impact

PAUL SCHERRER INSTITUT

Virtual Data Sets - Performance

Code

https://github.com/paulscherrerinstitute/hdf5_vds_performance_tests

Results

http://nbviewer.jupyter.org/github/paulscherrerinstitute/hdf5_vds_performance_tests/blob/master/results/analysis.ipynb

PAUL SCHERRER INSTITUT

PSI HDF5 Writer

HDF5 Writer - Used For

- Pilatus
- Eiger
- GigaFRoST
- Jungfrau
- ...

PAUL SCHERRER INSTITUT PSI

HDF5 Writer - In a Nutshell

Generic HDF5 writer

- Can be used with any detector supporting streaming
- Can be used for various data formats
- Can be extended with additional functionality (e.g. different compression algorithms)

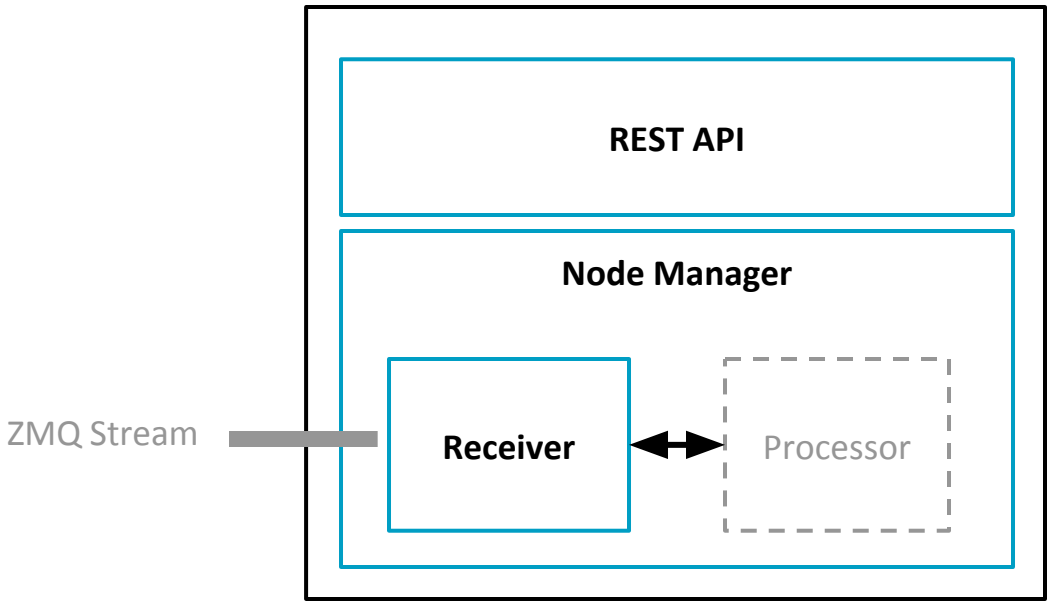
REST API

- All functionality exposed
- Provides real time feedback

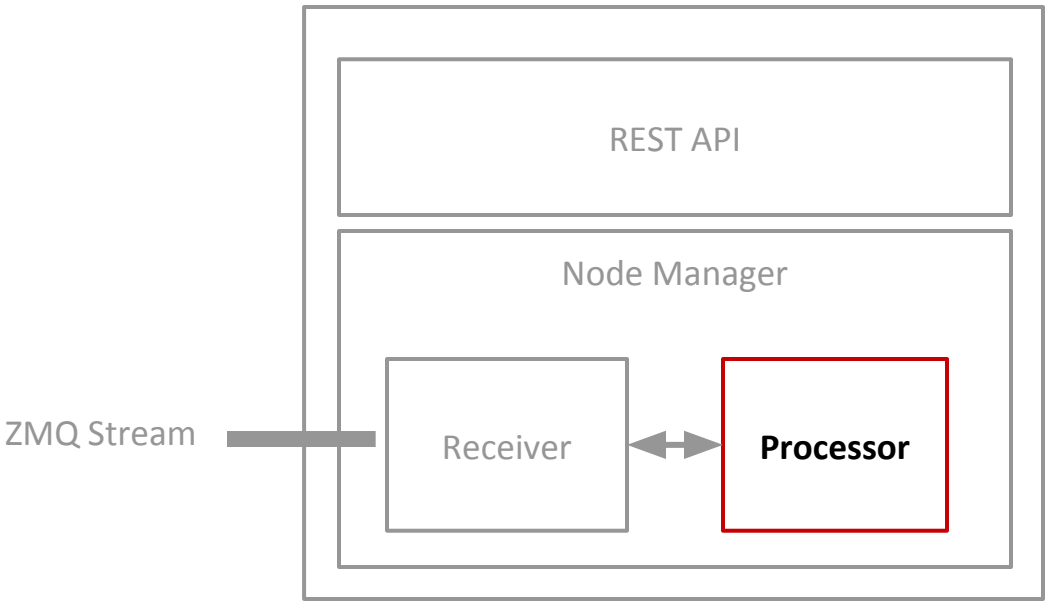
Take full advantage of infrastructure/hardware

- Parallel receiving and writing of data
- Ability to have multiple writers
- Make full use of the capabilities of the storage infrastructure

mflow Node - Generic Part



mflow Processor - Specific Part



mflow Processor - Specific Part

HDF5 Processor

- **Generic** stream HDF5 writer
- Writes data files with frame rollover
- Writes metadata (attributes / additional datasets)

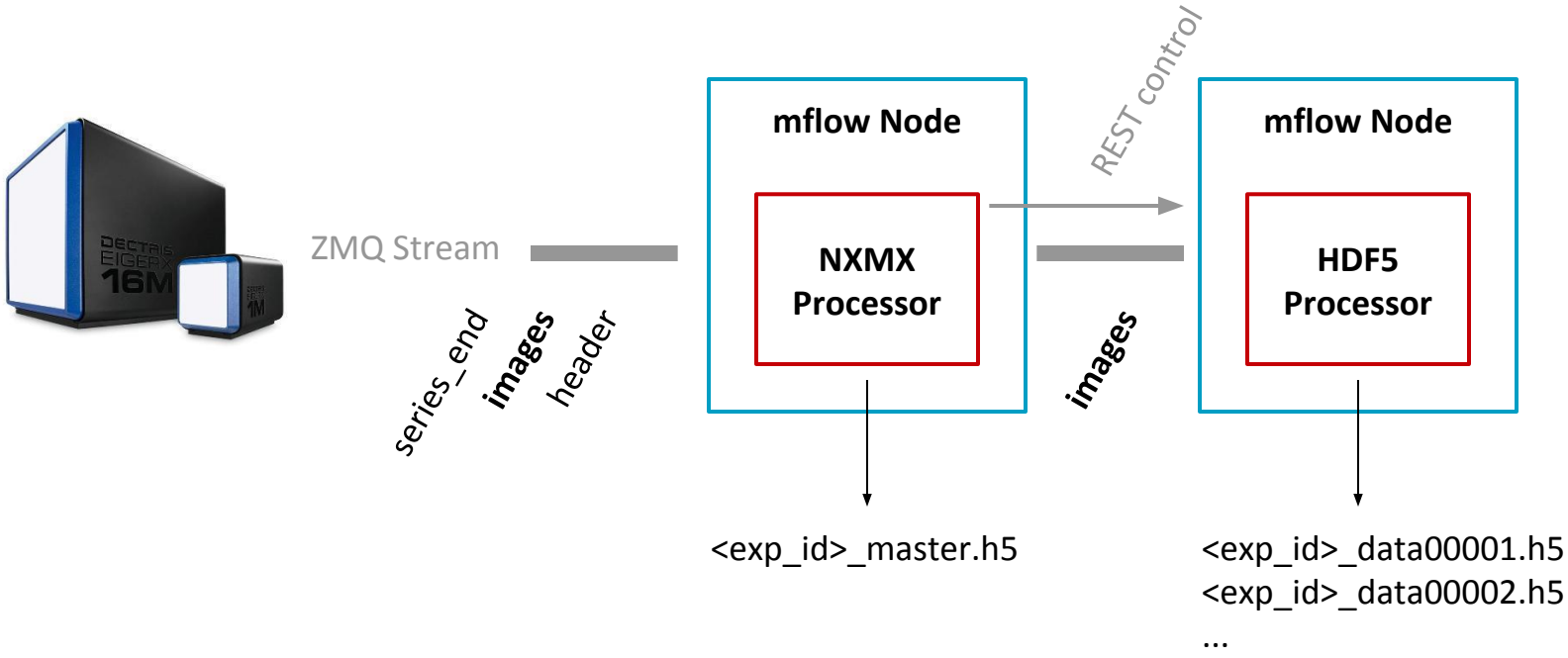
NXMX Processor

- Knows about the Dectris Eiger NXMX format
- Controls the HDF5 writer
- Writes master file
- Filters the stream and forwards it to the writer

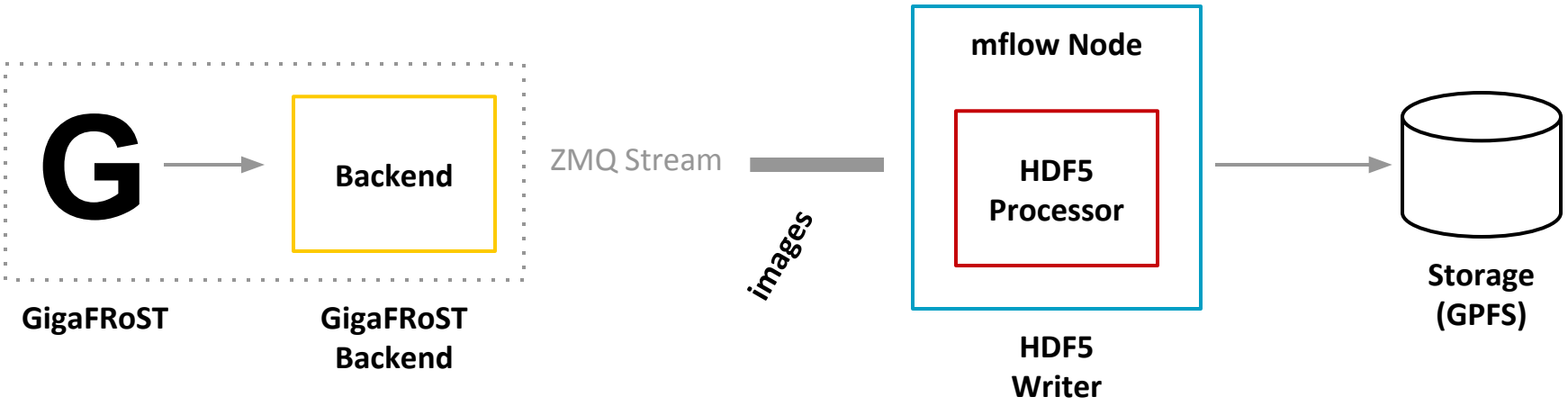
U-NAME-IT Processor

- Simple Python class that implements start, stop, process_message
- Set parameters via REST api

Overview - Eiger NXMX



Overview Performance - GigaFRoST



Overview Performance - GigaFRoST

	GigaFRoST		Backend		HDF5 Writer		Storage
Hardware/ Software		8x 10Gb Ethernet SFP+	16 receiving processes 1 sending process	40Gb Ethernet	2 receiving threads 1 writing thread	56Gb Infiniband	GPFS
Protocol		UDP		TCP/ZMQ **mflow**		GPFS	
Actual Performance		7.6GB/s		**2.8GB/s**		2.8GB/s	
Achieved Performance			generated data	3.3GB/s	generated data	4.2GB/s	
Theoretical Possible Performance		10GB/s		5GB/s		7GB/s	

PAUL SCHERRER INSTITUT

PSI HDF5 Writer

Important Links

https://github.com/datastreaming

https://github.com/datastreaming/mflow_nodes

https://github.com/datastreaming/mflow_node_processors

Questions ?

Advertisement

Mo 09.10.2017 10:45-11:00 - **MOAPL04** - SwissFEL Control System - Overview, Status, and Lessons Learned - Elke Zimoch - Paul Scherrer Institut - Talk

Tue 10.10.2017 14:45-15:00 - **TUCPL04** - SwissFEL Timing System: First Operational Experience - Babak Kalantari - Paul Scherrer Institut - Talk

Tue 10.10.2017 15:15-15:30 - **TUCPA06** - SwissFEL - Beam Synchronous Data Acquisition - The First Year - Simon Gregor Ebner - Paul Scherrer Institute

Tue 10.10.2017 17:45-18:15 - **TUSH102** - PShell: from SLS beamlines to the SwissFEL control room - Alexandre Gobbo - Paul Scherrer Institut - Poster

Thu 12.10.2017 16:15-16:30 - THDPL02 - GigaFRoST (Gigabyte Fast Read-Out System for Tomography): Control and DAQ System Design - Tine Celcer - Paul Scherrer Institut - Talk

PAUL SCHERRER INSTITUT

Contact

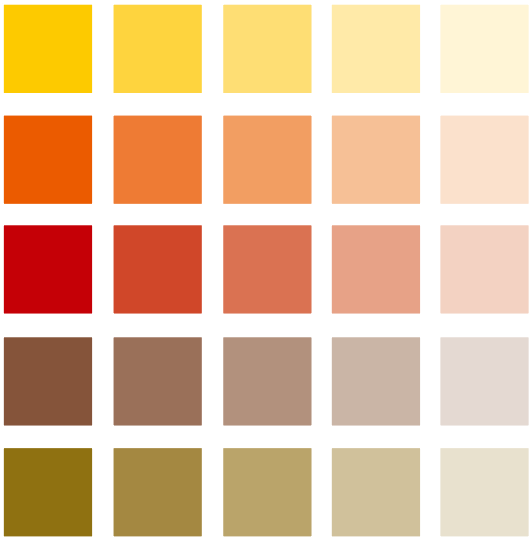
Simon Ebner
Paul Scherrer Institute
WBGB/001
5232 Villigen PSI

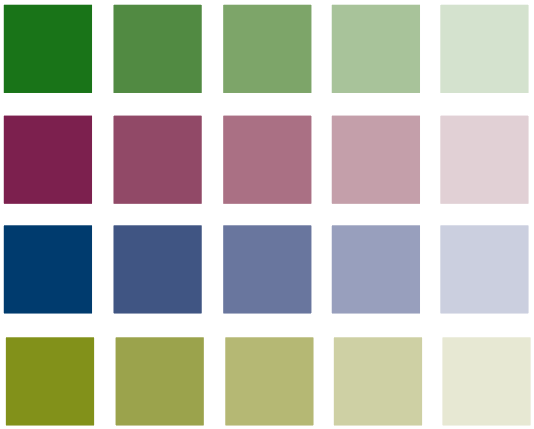
simon.ebner@psi.ch

PSI Colour Scheme

PSI's basic colours

Colour options for graphs:
1st choice





Colour options for graphs:
2nd choice