DECTRIS®

detecting the future

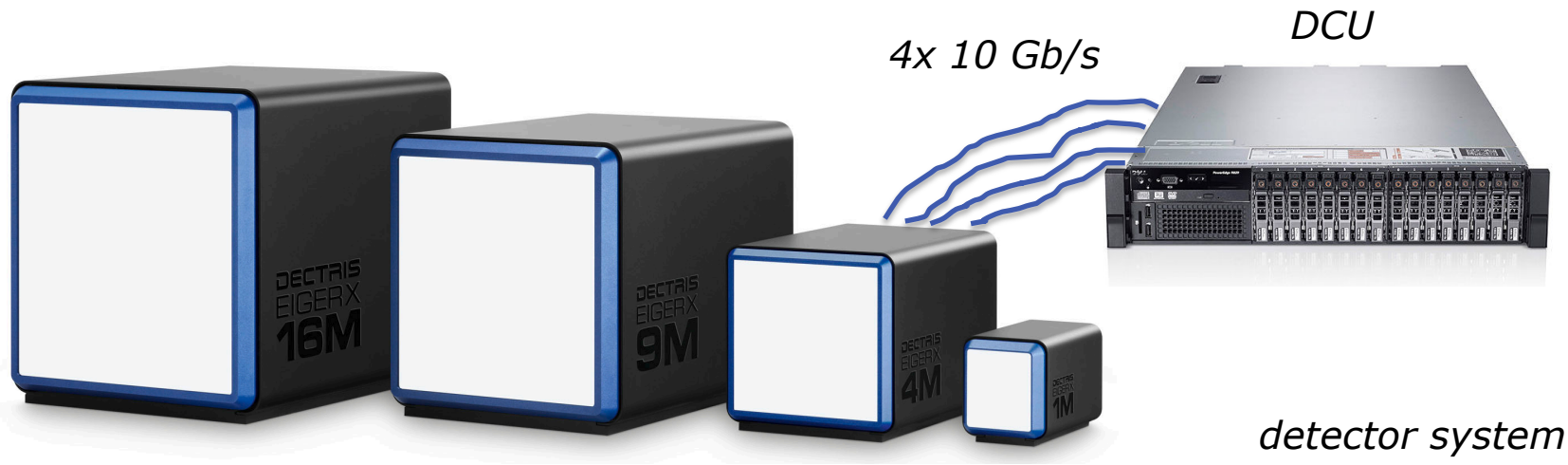
EIGER Data

Andreas Förster – Application Scientist MX

High Data Rate MX, 2 Sep 2016

DECTRIS Ltd.
5405 Baden-Dättwil
Switzerland
www.dectris.com

High internal data rates



	Maximum frame rate	Size of single image	Maximum data rate
16M	133 Hz	36 MB (@ 16-bit)	36.0 Gbit/s
9M	238 Hz	20 MB (@ 16-bit)	36.1 Gbit/s
4M	750 Hz	6.7 MB (@ 12-bit)	37.6 Gbit/s
1M	3000 Hz	1.6 MB (@ 12-bit)	36.8 Gbit/s

Efficient compression required

	Sustained frame rate	Size of single image	Uncompressed data rate	Required compression
16M	66 Hz	36 MB (@ 16-bit)	17.8 Gbit/s	1.78
9M	119 Hz	20 MB (@ 16-bit)	18.0 Gbit/s	1.80
4M	375 Hz	9.0 MB (@ 16-bit)	25.1 Gbit/s	2.51
1M	1500 Hz	2.2 MB (@ 16-bit)	24.5 Gbit/s	2.45

Bit-shuffle LZ4 compression

Kiyoshi Masui (github.com/kiyo-masui/bitshuffle)
=> Numbers from EIGER X 16M @ SLS (X06SA)

	Compression method	Angular increment [°]	Frame rate [Hz]	Image size [MB]	Compression factor	Data per 360° [GB]	Data rate [MB/s]
	none	-	-	72	1	-	
	CBF	0.25	4	18	4.0	25	72
32-bit	bs-LZ4	0.25	4	11	**6.5**	15	44
	bs-LZ4	0.1	10	6.5	**11.1**	23	65
	bs-LZ4	0.05	20	5.5	**13.1**	39	110
16-bit	bs-LZ4	0.02	50	3.5	**10.3**	62	175
	bs-LZ4	0.01	100	2.7	**13.3**	95	270
	bs-LZ4	0.01	133	2.5	**14.4**	88	**333**
	CBF	0.01	133	18	2.0	633	2394
	none	-	-	36	1	-	

Data rates are manageable

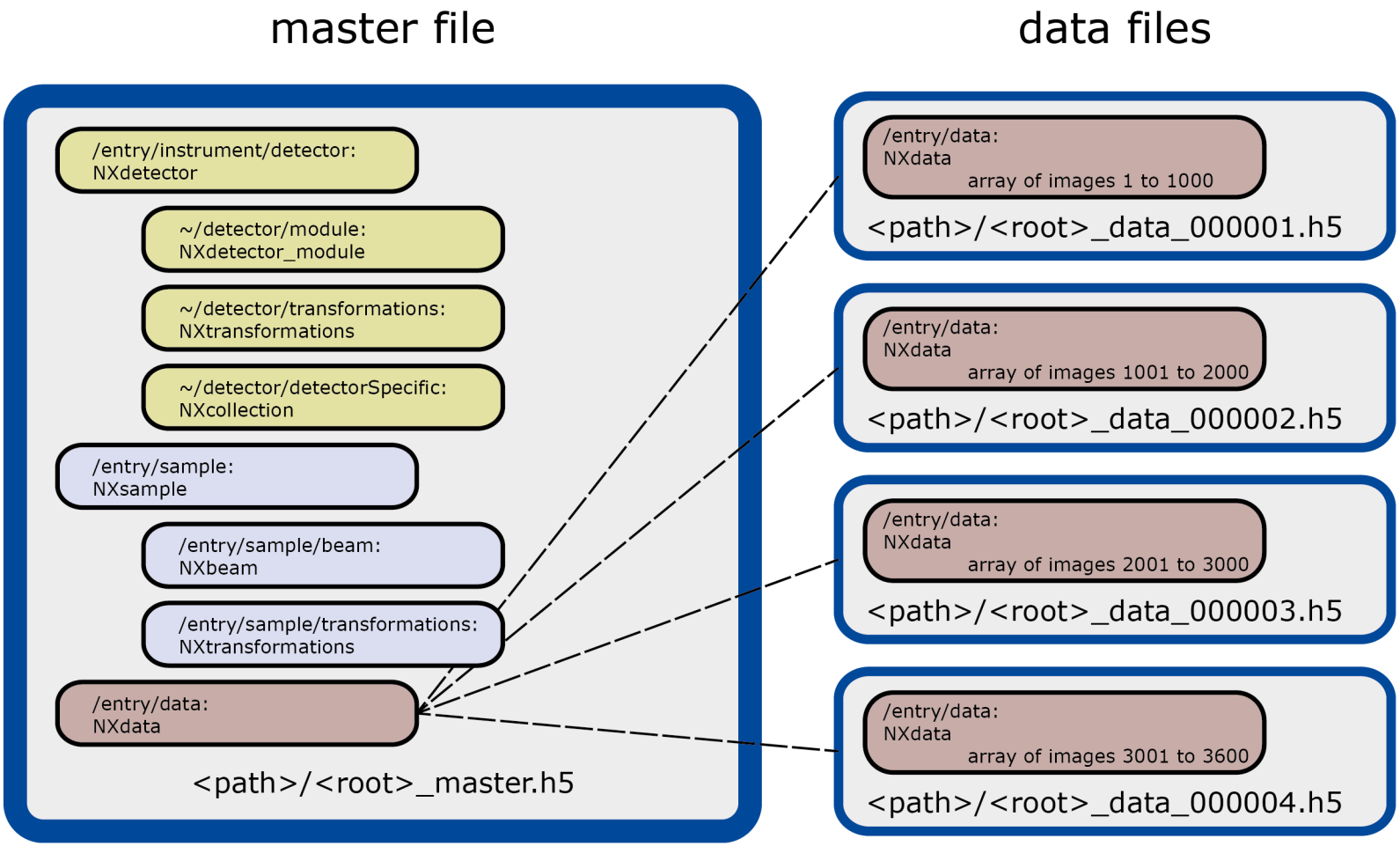
	Sustained frame rate	Uncompressed data rate	Minimal compression	Effective data rate
16M	66 Hz	17.8 Gbit/s	4	570 MB/s
9M	119 Hz	18.0 Gbit/s	4	576 MB/s
4M	375 Hz	25.1 Gbit/s	4	803 MB/s
1M	1500 Hz	24.5 Gbit/s	4	784 MB/s

Writing of EIGER data

```
619M Nov 11  2015 insu6_1_data_000001.h5
618M Nov 11  2015 insu6_1_data_000002.h5
620M Nov 11  2015 insu6_1_data_000003.h5
624M Nov 11  2015 insu6_1_data_000004.h5
631M Nov 11  2015 insu6_1_data_000005.h5
639M Nov 11  2015 insu6_1_data_000006.h5
648M Nov 11  2015 insu6_1_data_000007.h5
654M Nov 11  2015 insu6_1_data_000008.h5
657M Nov 11  2015 insu6_1_data_000009.h5
350M Nov 11  2015 insu6_1_master.h5
```

- ***File Writer***
  - *Writes metadata and images (compressed*
- ***Stream***
  - *Produces stream of compressed data*
  - *Preceded by metadata from detector configuration*
- ***Monitor***
  - *Low-performance, low-bandwidth*
  - *Inspection of single images*
  - *No metadata*

Organization of EIGER data



Streaming of EIGER data

Data exchange via ZeroMQ sockets

- ***What is ZeroMQ?***
  - *Looks like an embeddable networking library*
  - *Acts like a concurrency framework*
  - *Fast, lightweight messaging library*
  - *Supercharged TCP sockets*
  - *Various protocols to exchange messages between server and client(s)*

Streaming of EIGER data

Three kinds of ZeroMQ messages

- ***Global header data***
  - *Detector configuration*
  - *Flat field, pixel mask*
- ***Image data***
  - *Series and frame ID*
  - *Data type and compression*
  - *Data as binary large object*
  - *Time stamp*
- ***End-of-series***
  - *That's it, folks*

EIGER Stream interface

Global header data

- ***Multipart ZeroMQ message***
  1) *Description of header detail*
  2) ***Detector configuration***
  3) *Flatfield header*
  4) ***Flatfield** blob*
  5) *Pixel mask header*
  6) ***Pixel mask** blob*
  7) *Count rate table header*
  8) *Count rate table blob*
  9) *Optional appendix*

EIGER Stream interface

Image data

- ***Multipart ZeroMQ message***
  1) Series and frame ID
  2) Data type, size and compression
  3) ***Image data*** *as blob*
  4) Time stamp in ns relative to start
  5) Optional appendix

EIGER Stream interface

End of series

- ***Single ZeroMQ message***
  - json string with series ID

Streaming of EIGER data

Data stream out of detector

- ***Lost if not collected***

Reception of EIGER stream

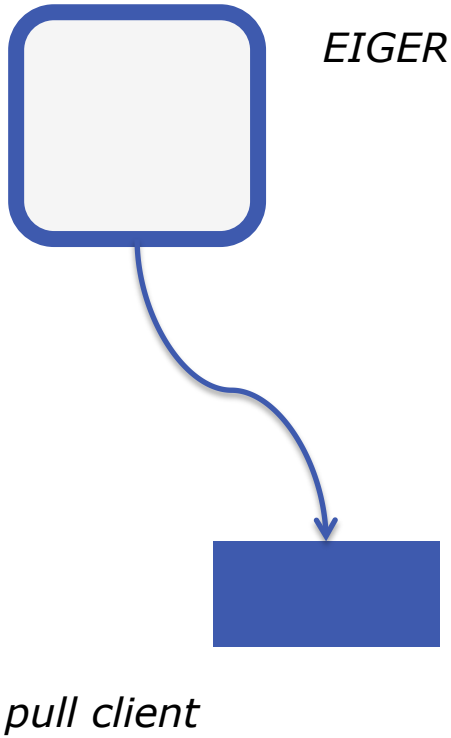
Data stream out of detector

- **Stream receiver**
  - *Listens to stream*
  - *Does something to data*
  - *Multiple stream receivers possible*
- **Data transfer**
  - *Handover to processing software*
  - *Use for online processing*
  - *Save to HDF5*
  - *Poorly tested in HDXMR environments*

ZeroMQ transfer patterns

Push/pull transfer pattern

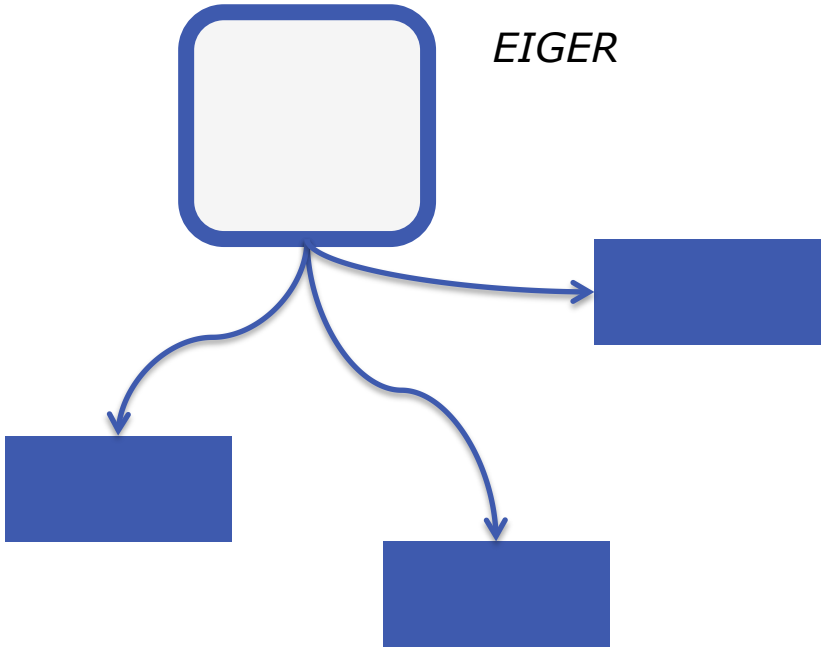
- ***Single client***
  - Client listens to stream



ZeroMQ transfer patterns

Push/pull transfer pattern

- ***Single client***
  - *Client listens to stream*
- ***Any number of clients***
  - *Load balancing among clients*
  - *Useful for parallel spot finding?*
  - *Reassembly of dataset required*



Conclusion

Three ways of data

- ***Stream for maximum flexibility***
  - *Process data in memory*
  - *Integrate into beamline infrastructure*
  - *Write bespoke HDF5 files (standard!!)*
  - *Do things in parallel*
- ***File Writer for simplicity***
  - *Writes standard HDF5 files*
  - *Follows NeXus standard*
  - *Archiving and processing*
- ***Monitor for overview***
  - *Inspect single images*

DECTRIS®

detecting the future

Thank you for your attention!

www.dectris.com

andreas.foerster@dectris.com

DECTRIS Ltd.
5405 Baden-Dättwil
Switzerland
www.dectris.com