Efficient Eiger Data Processing

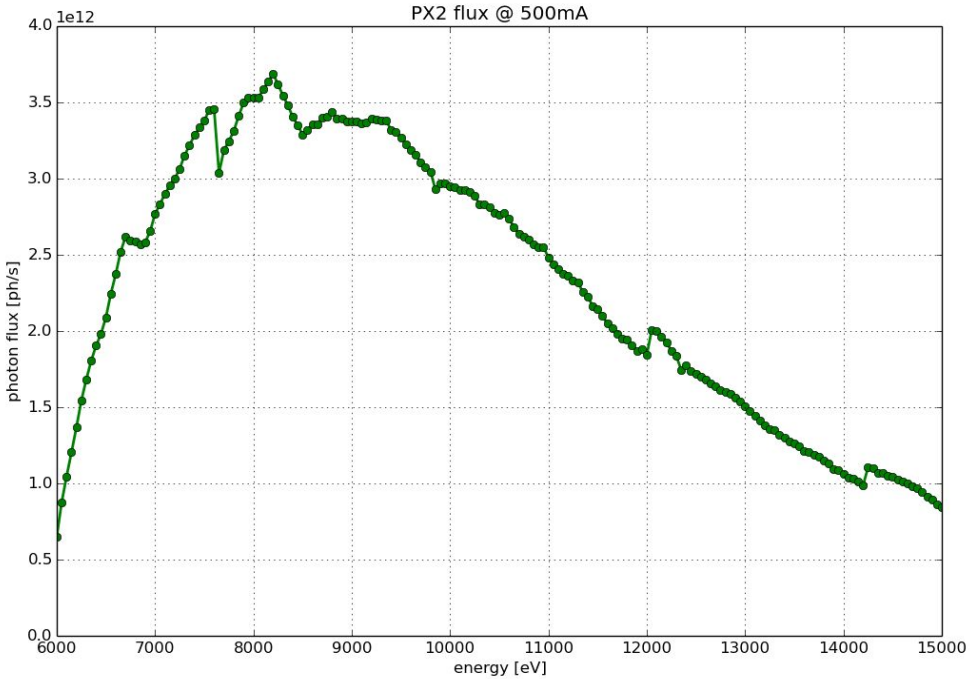
Martin Savko
savko@synchrotron-soleil.fr

Overview

- Beamline
- Eiger Setup
- Computational infrastructure
- Eiger Data Processing
- Conclusion

Proxima 2A

- Users since March 2013
- Microfocus (5x10um), linearly polarized
- 3.6e12 ph/s @ 8.1keV
  - Tunable 6 - 17 keV
- MD2 goniometer
- CATS sample changer (144 samples)
- Eiger X 9M detector



Eiger X 9M Installation and Commissioning

- Installation November 2015
- User operation since December 2015
- bslz4 compression
- Max speeds
  - 238Hz @ 9M
  - 750Hz @ 4M ROI (stable as of SIMPLON API 1.6.2)

Infrastructure

- 10Gbit network
- Storage (Active Circle based), NFS access
  - Tiered system
    - 10TB local SSD
    - 20TB local SAS
    - 1PB remote

Processing infrastructure

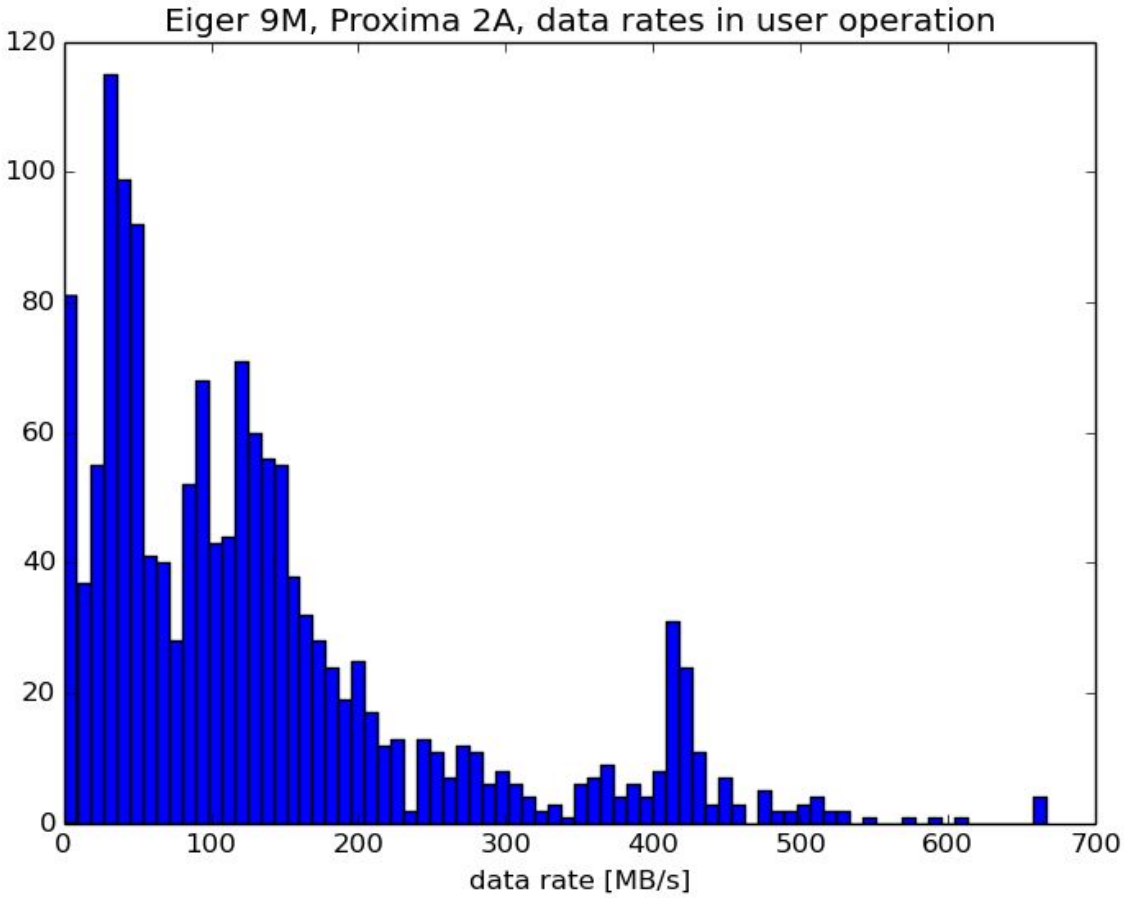
- Huawei FusionServer RH8100 V3 Rack Server *
  - 8 x XEON E7-8890 v3 @ 2.5GHz
  - 144 cores, 288 threads
  - 2.56 TB RAM (DDR4 1866MHz)
  - 4 x 10GBe
  - 5.76 TFlops (estimated)
  - 8U form factor
- System dedicated to the single beamline

* http://e.huawei.com/en/products/cloud-computing-dc/servers/rh-series/rh8100-v3

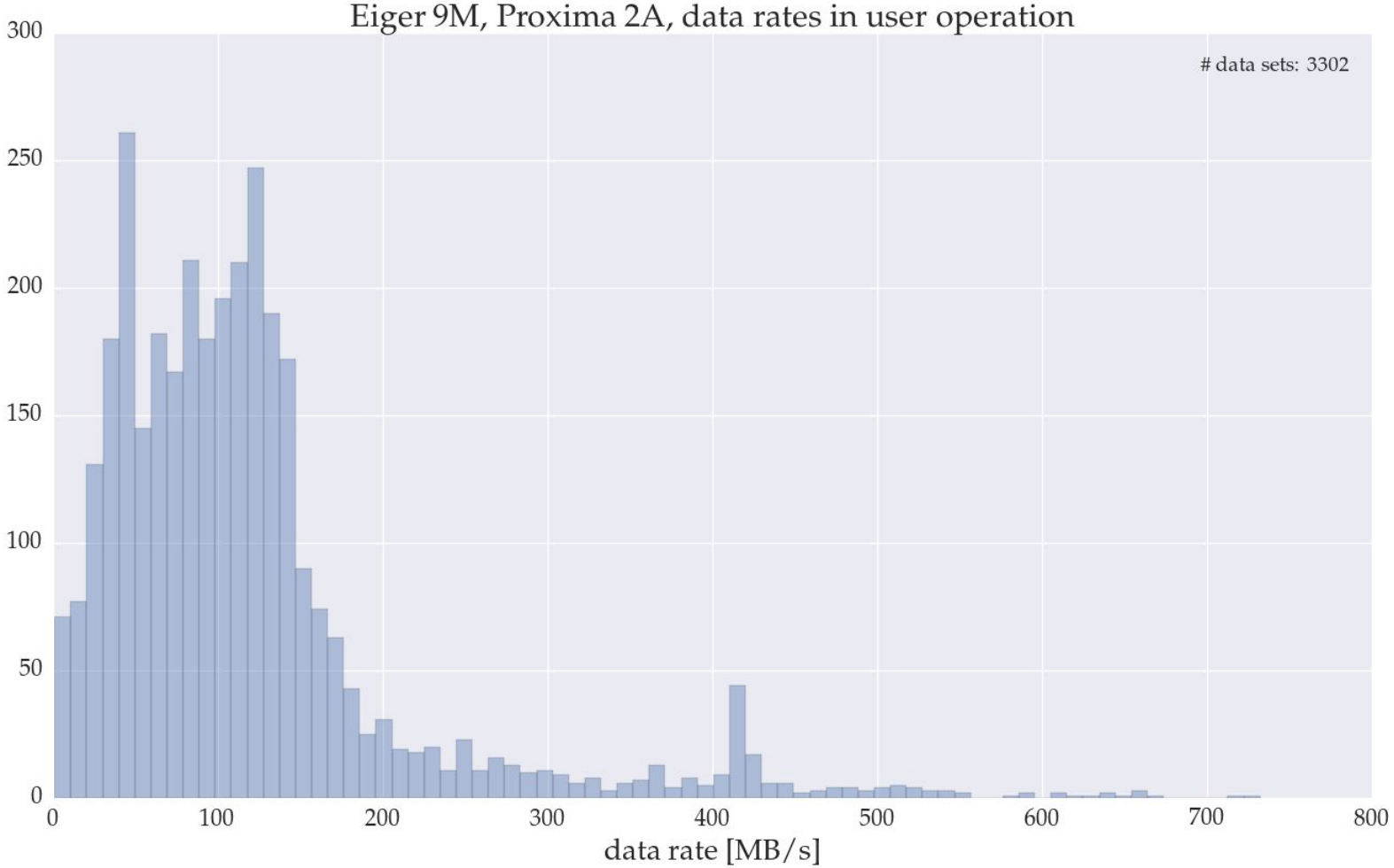
Performance of the setup

- ~ 1000 MB/sec download speed
  - Two 10Gbit ports for getting data out of DCU
- ~ 120MB/s is the average data rate
  - Maximum observed data rate ~ 732 MB/s
  - In practice no data transfer bottleneck thanks to bitshuffle lz4
- The server has RAM cache of 170 GB
  - ~ 20 min autonomy assuming average data rate in bslz4 compression
- 12.75 is the average observed bslz4 compression ratio
  - x 13.34 per 32bit -- average compressed image size ~3 MB
  - x 10.13 per 16bit -- average compressed image size ~2 MB

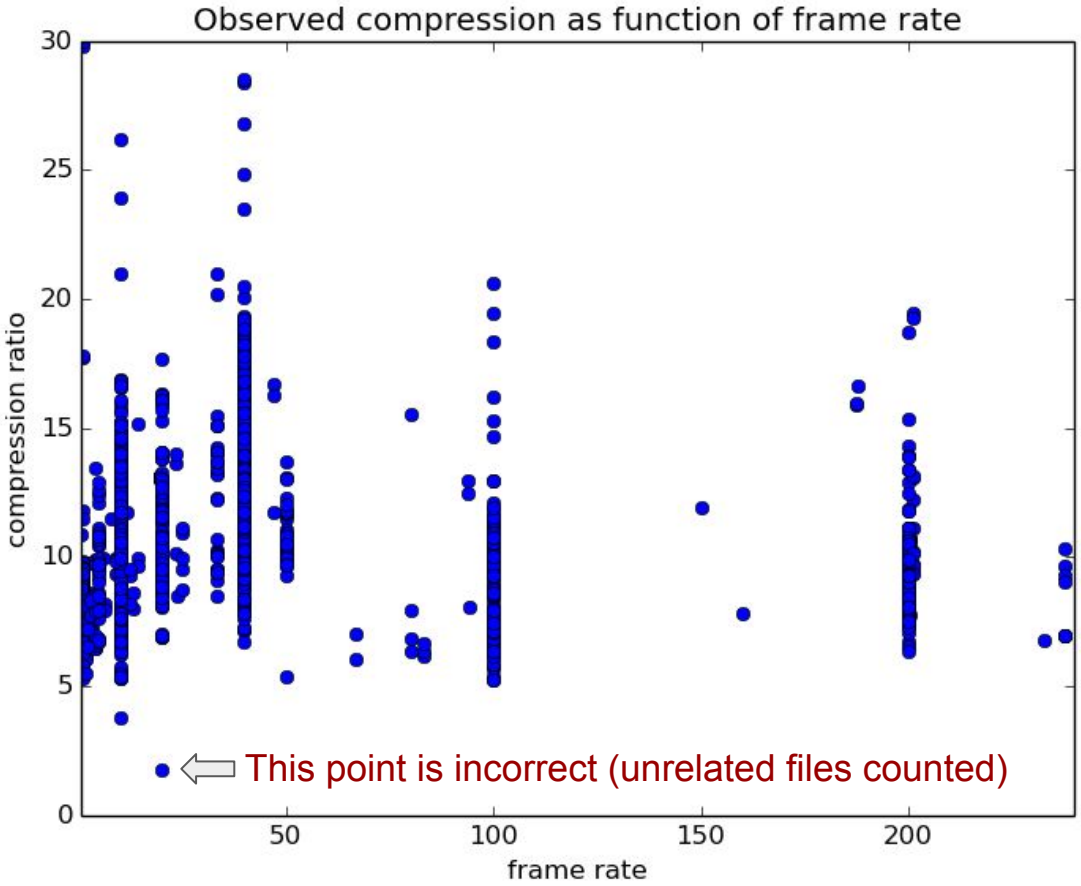
May 2016


Eiger 9M, Proxima 2A, data rates in user operation
120
100
80
60
40
20
0
0
100
200
300
400
500
600
700
data rate [MB/s]

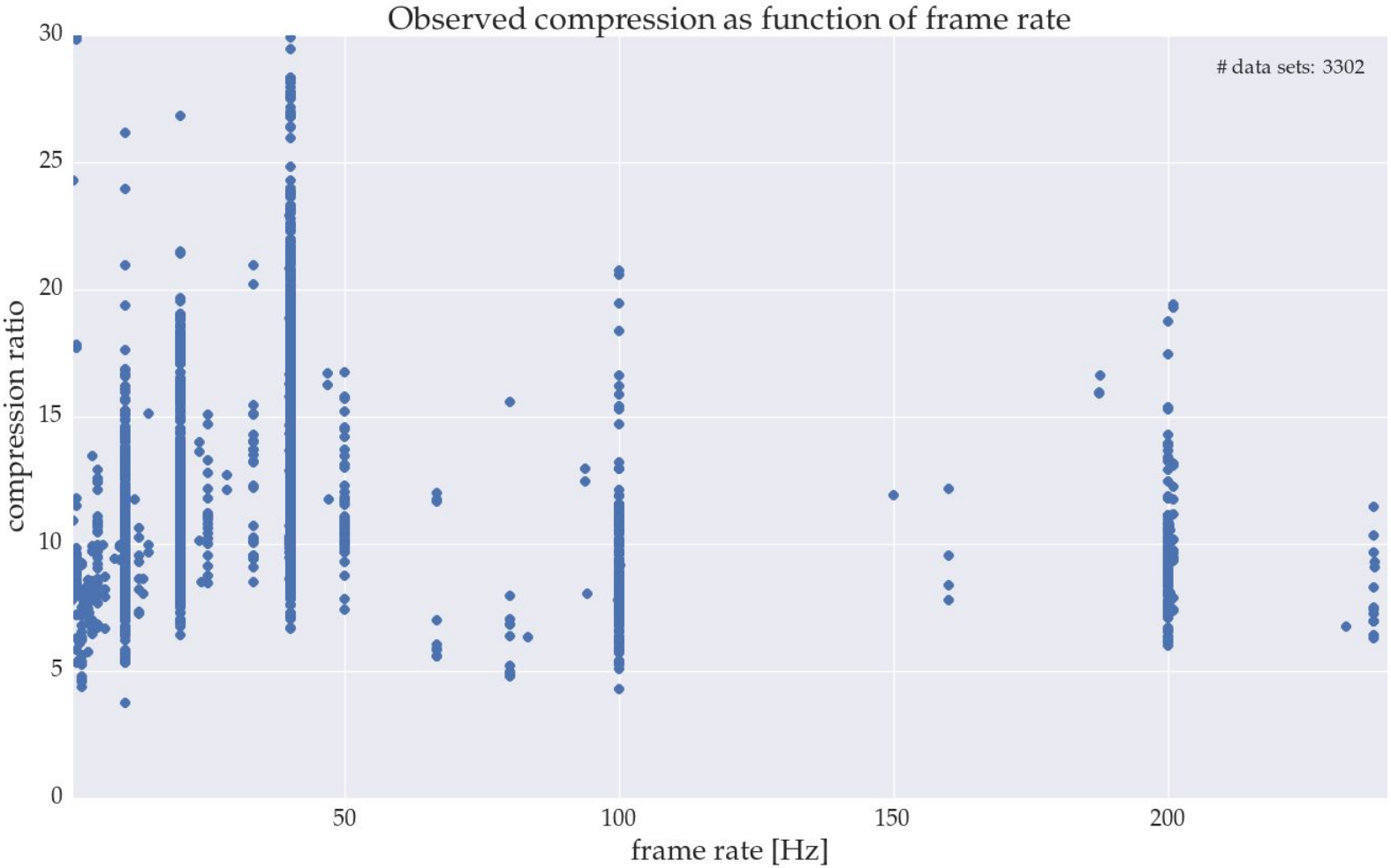
September 2016



May 2016


Observed compression as function of frame rate
compression ratio
frame rate
0
5
10
15
20
25
30
50
100
150
200
This point is incorrect (unrelated files counted)

September 2016

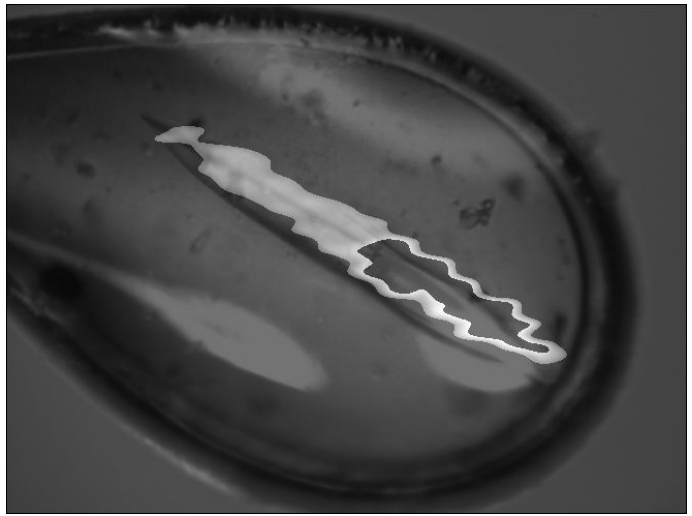


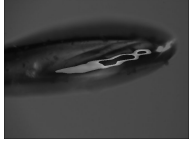
Eiger Data Processing

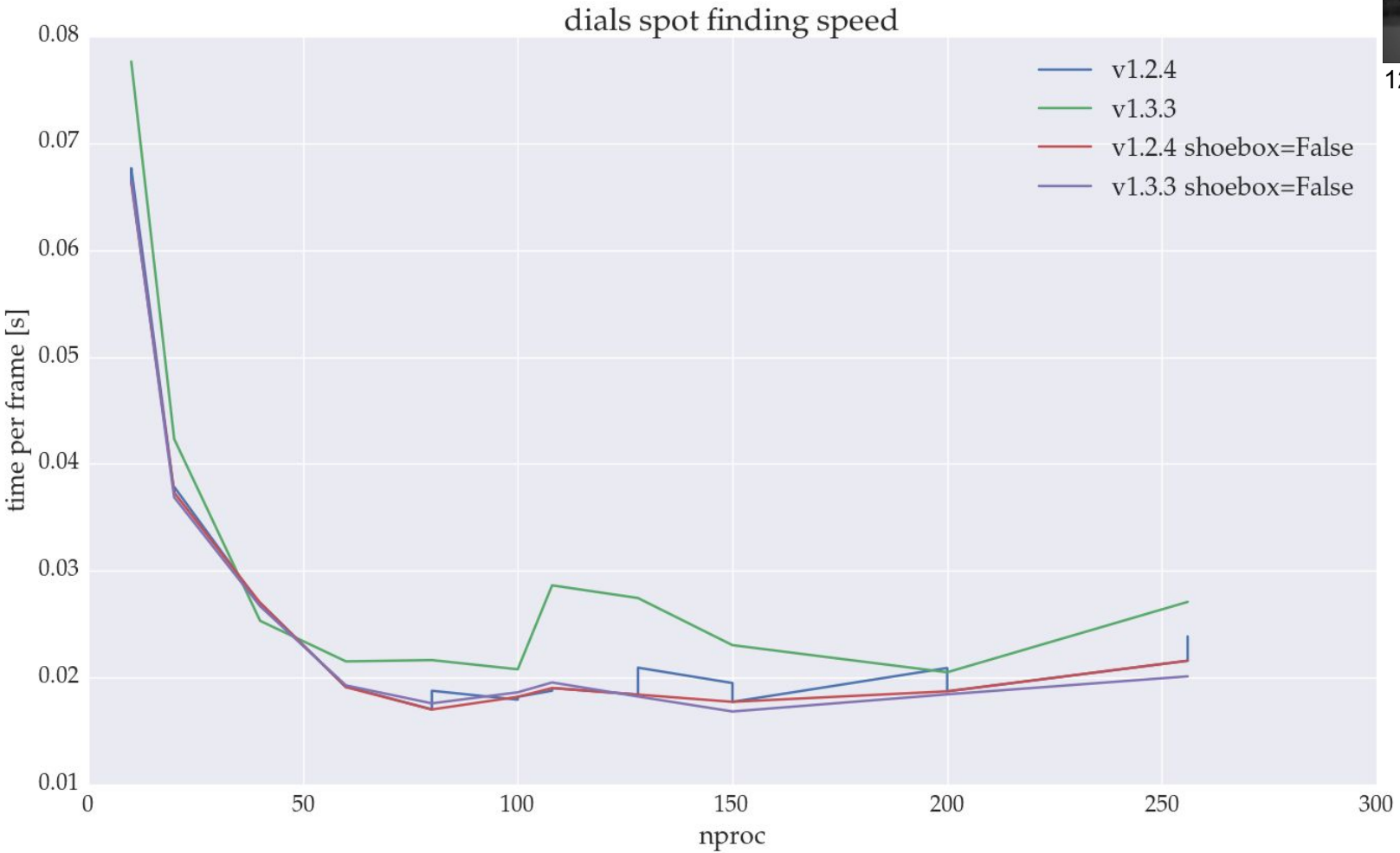
- Oscillation data processed via XDS
- Raster scan analysis via DIALS (`dials.find_spots`)
- Recent XDS for efficient data caching
  - Useful tips at http://strucbio.biologie.uni-konstanz.de/xdswiki/index.php/Eiger
- Processing HDF5 data compared to CBF equivalent with XDS is slower
  - at least 20% overall penalty, often we see penalty closer to 50%
  - beware of what is running on the computer at the same time (avoid virtualbox :) !
  - Importance of cache management: #sync; echo 3 > /proc/sys/vm/drop_caches
- Conversion from HDF5 to CBF
  - ~100Hz
  - H5ToXDS run in parallel via python wrapper to generate correct mini-cbf header
  - Generation of temporary CBFs makes sense if data need to be accessed repeatedly

Raster scans

- 5x10 micrometer beam
- 40 Hz default frame rate
- fast axis speed ~0.5 mm/s
- typical grid size 0.1 mm^2 ~1000 images
- typical acquisition time 40 seconds
- processing time 20 seconds
  - dials.find_spots ~ 0.02s/image
  - native support for HDF5




1200 images


dials spot finding speed
v1.2.4
v1.3.3
v1.2.4 shoebox=False
v1.3.3 shoebox=False
time per frame [s]
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
50
100
150
200
250
300
nproc

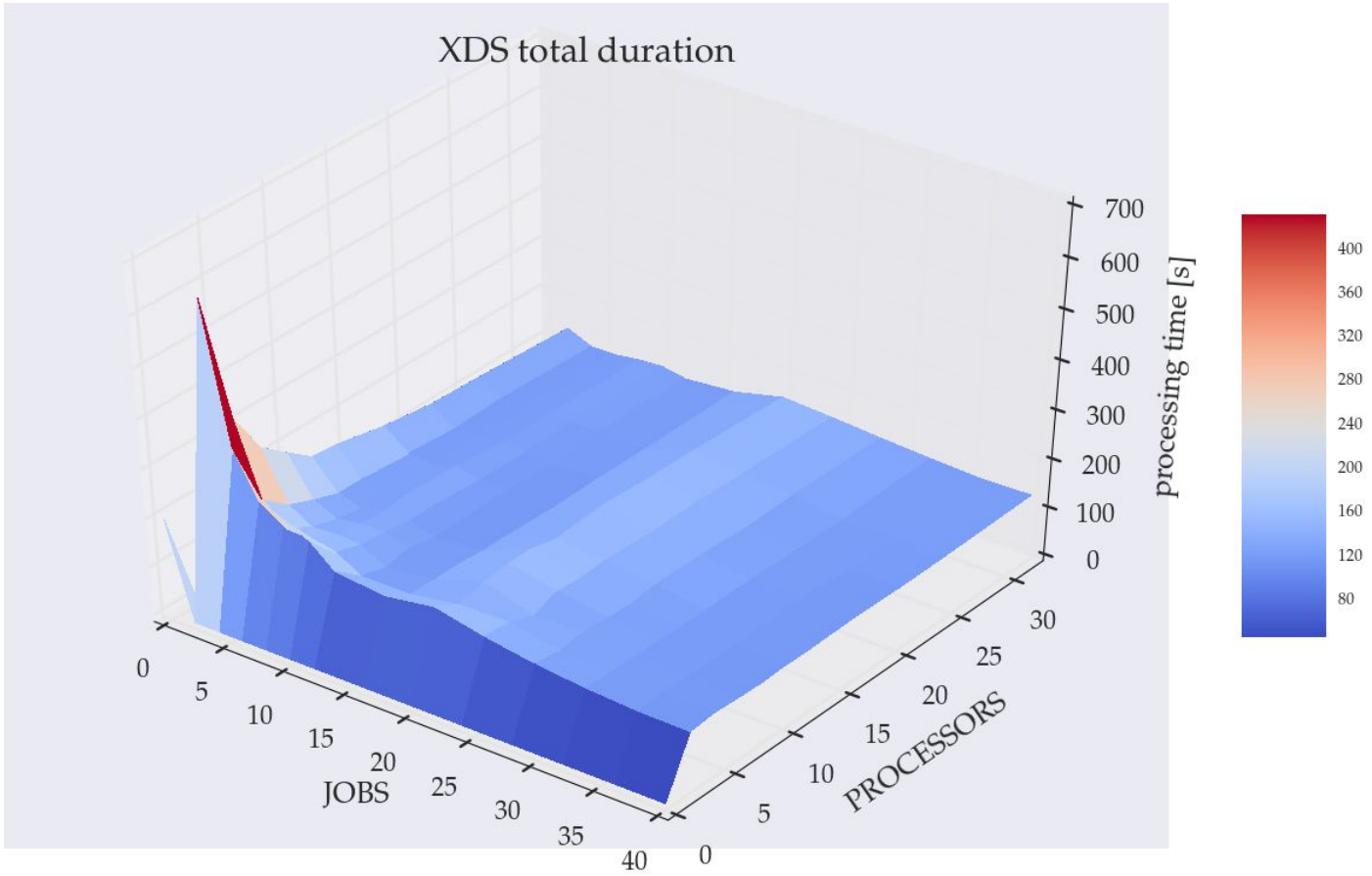
Evaluating FusionServer RH 8100 v3

- Datasets: https://www.dectris.com/datasets.html
- Parameters influencing processing time the most
  - MAXIMUM_NUMBER_OF_JOBS
  - MAXIMUM_NUMBER_OF_PROCESSORS
  - CBF vs. HDF5
  - Long term performance of the system

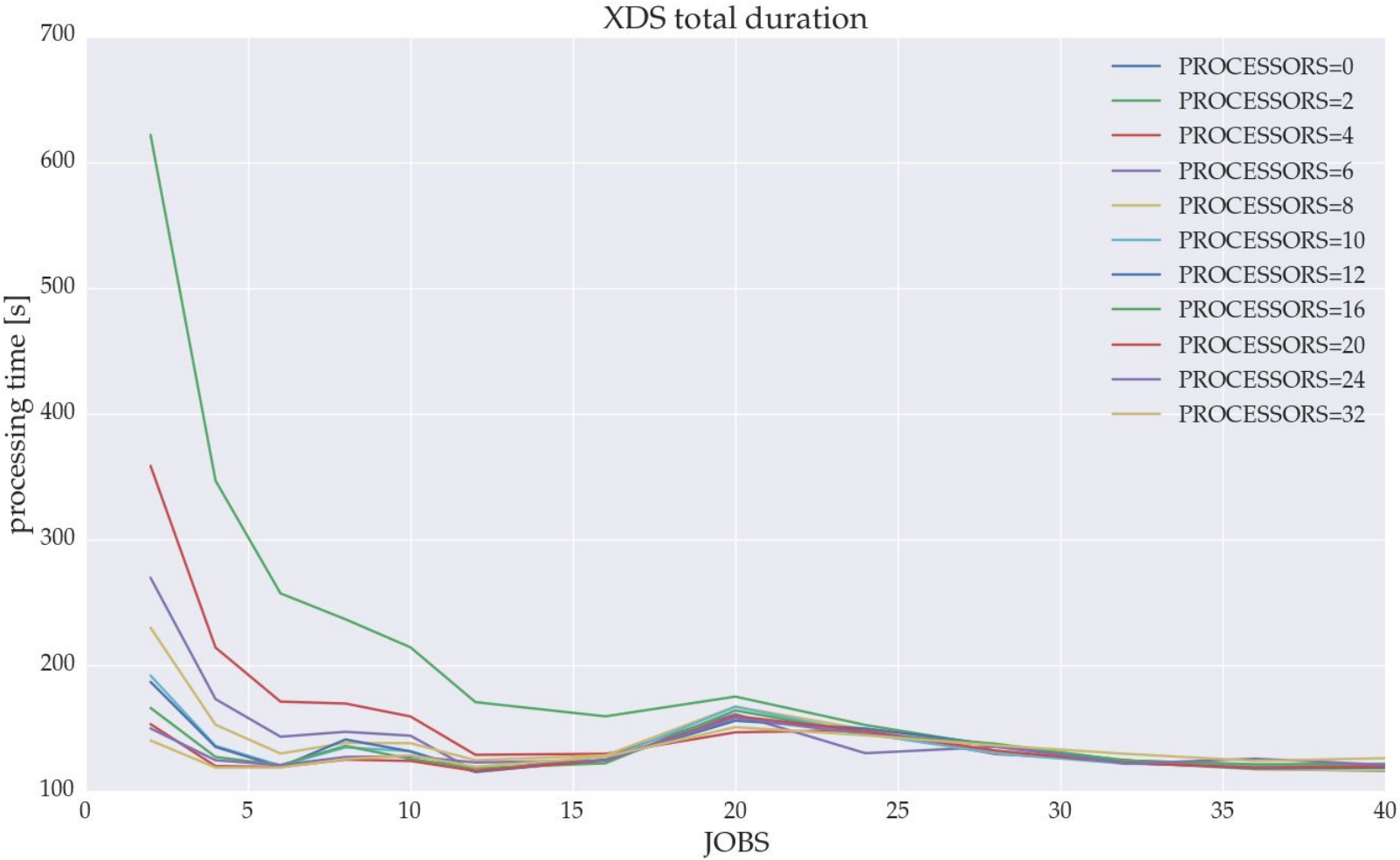
Eiger X 9M dataset

- 1800 frames, 180 degrees, 0.1 degree oscillation, frame rate 200Hz
- Evaluation of influence of combination of `MAXIMUM_NUMBER_OF_JOBS` and `MAXIMUM_NUMBER_OF_PROCESSORS` on data processing duration
- Let’s first look at the total time and then individual stages

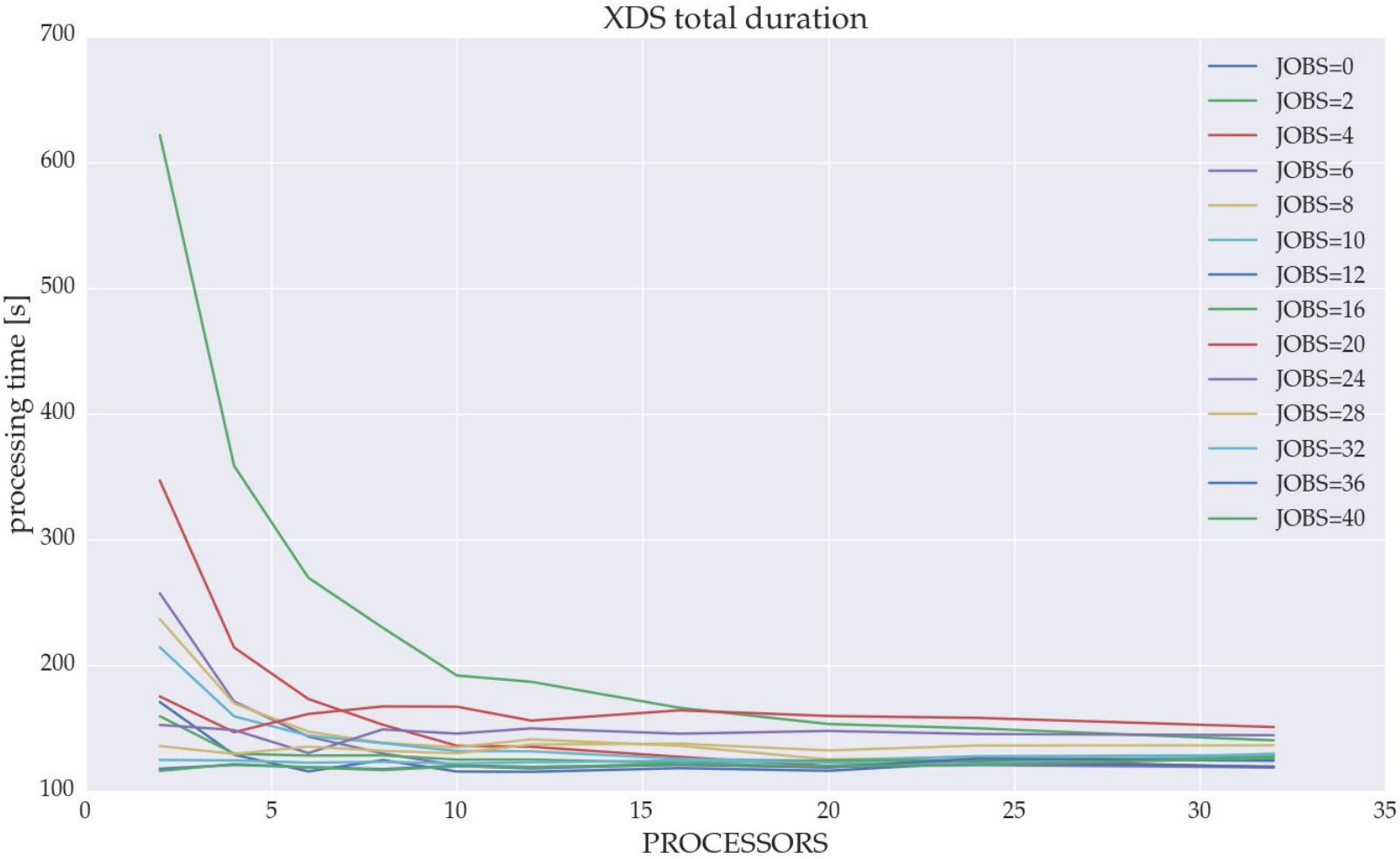
HDF dataset


XDS total duration
processing time [s]
700
600
500
400
300
200
100
0
JOBS
0
5
10
15
20
25
30
35
40
PROCESSORS
0
5
10
15
20
25
30
400
360
320
280
240
200
160
120
80

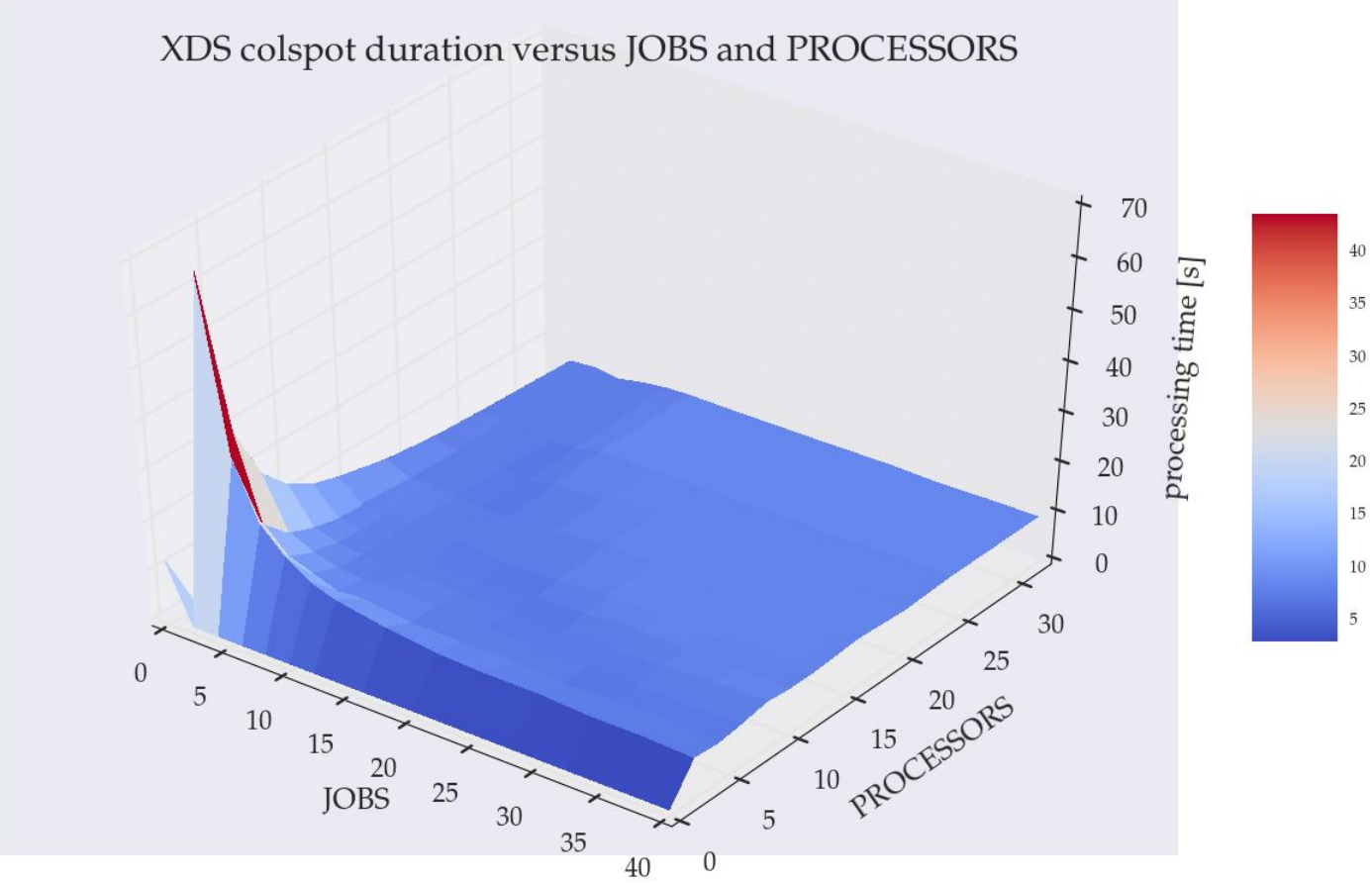
HDF dataset


XDS total duration
processing time [s]
JOBS
700
600
500
400
300
200
100
0
5
10
15
20
25
30
35
40
PROCESSORS=0
PROCESSORS=2
PROCESSORS=4
PROCESSORS=6
PROCESSORS=8
PROCESSORS=10
PROCESSORS=12
PROCESSORS=16
PROCESSORS=20
PROCESSORS=24
PROCESSORS=32

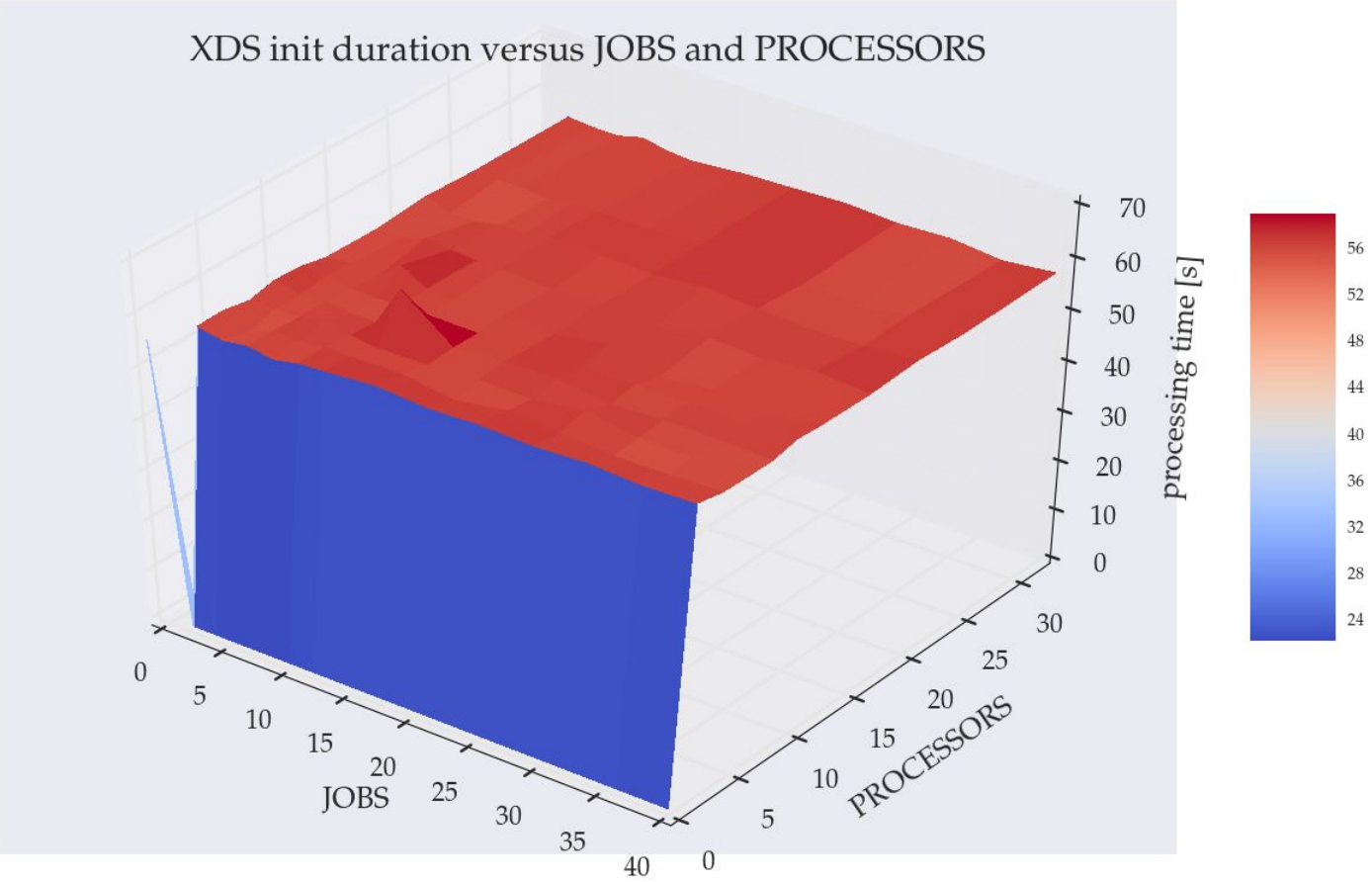
HDF dataset


XDS total duration
processing time [s]
PROCESSORS
700
600
500
400
300
200
100
0
5
10
15
20
25
30
35
JOBS=0
JOBS=2
JOBS=4
JOBS=6
JOBS=8
JOBS=10
JOBS=12
JOBS=16
JOBS=20
JOBS=24
JOBS=28
JOBS=32
JOBS=36
JOBS=40

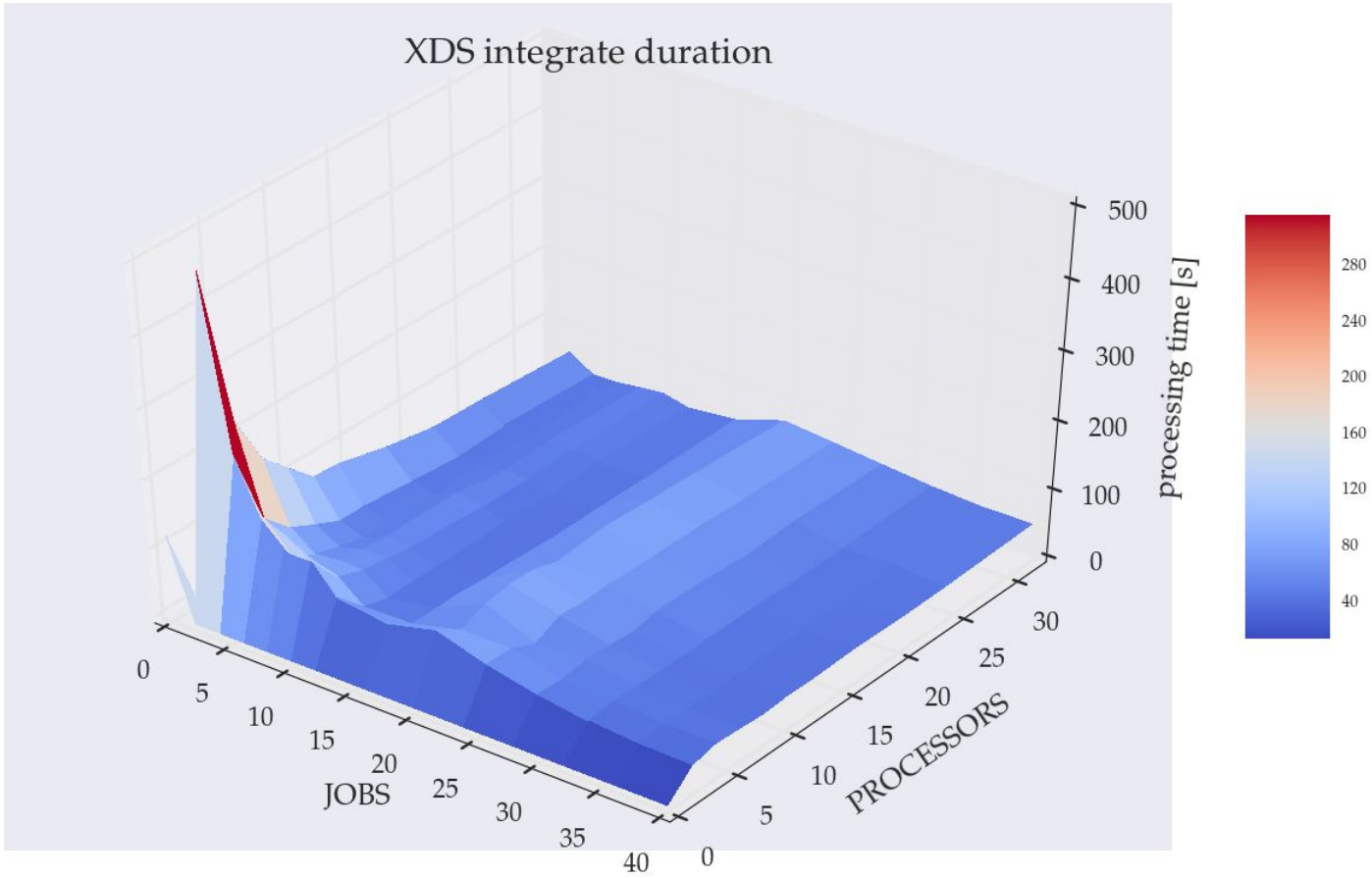
HDF dataset


XDS colspot duration versus JOBS and PROCESSORS
processing time [s]
70
60
50
40
30
20
10
0
JOBS
0
5
10
15
20
25
30
35
40
PROCESSORS
0
5
10
15
20
25
30
40
35
30
25
20
15
10
5

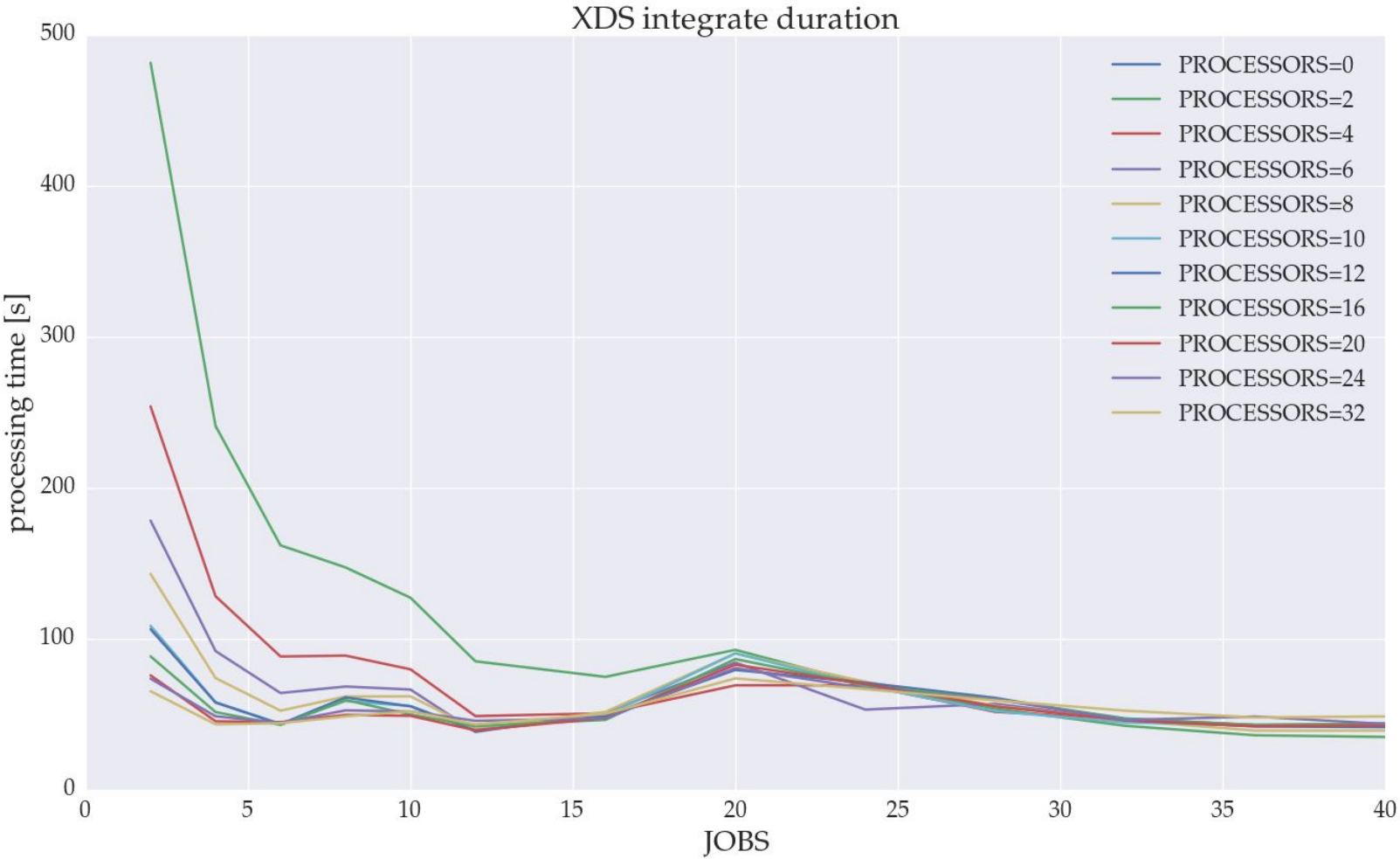
HDF dataset


XDS init duration versus JOBS and PROCESSORS
JOBS
0
5
10
15
20
25
30
35
40
PROCESSORS
0
5
10
15
20
25
30
processing time [s]
0
10
20
30
40
50
60
70
56
52
48
44
40
36
32
28
24

HDF dataset


XDS integrate duration
processing time [s]
500
400
300
200
100
0
JOBS
0
5
10
15
20
25
30
35
40
PROCESSORS
0
5
10
15
20
25
30
280
240
200
160
120
80
40

HDF dataset


XDS integrate duration
processing time [s]
JOBS
0
100
200
300
400
500
0
5
10
15
20
25
30
35
40
PROCESSORS=0
PROCESSORS=2
PROCESSORS=4
PROCESSORS=6
PROCESSORS=8
PROCESSORS=10
PROCESSORS=12
PROCESSORS=16
PROCESSORS=20
PROCESSORS=24
PROCESSORS=32

Processing from HDF5

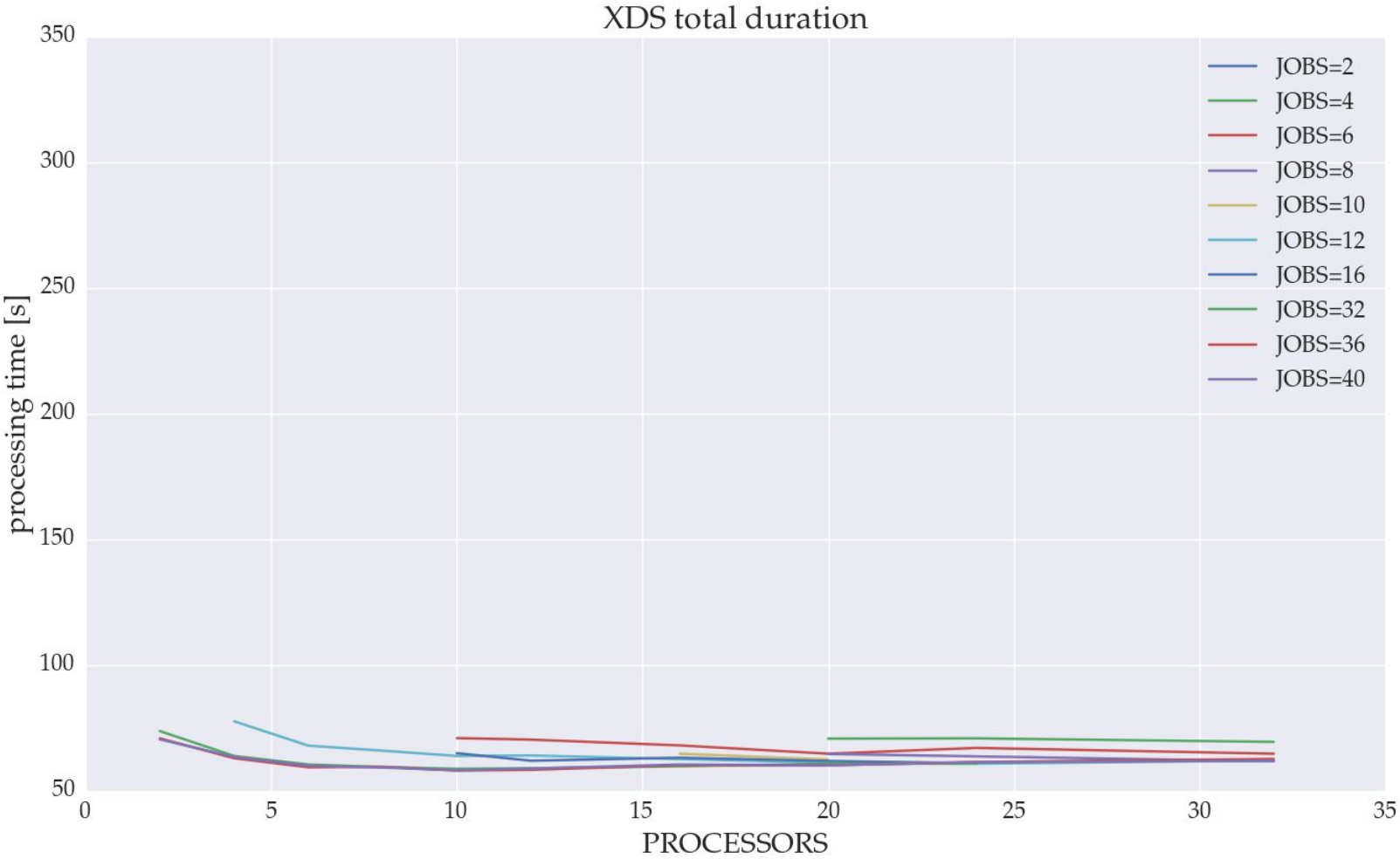
- The speed gain from increasing `MAXIMUM_NUMBER_OF_JOBS` and `MAXIMUM_NUMBER_OF_PROCESSORS` levels off at between 6 - 12 for both parameters
- For practical purposes we keep low end values close to optimum as it permits parallel run of multiple jobs

CBF versus HDF

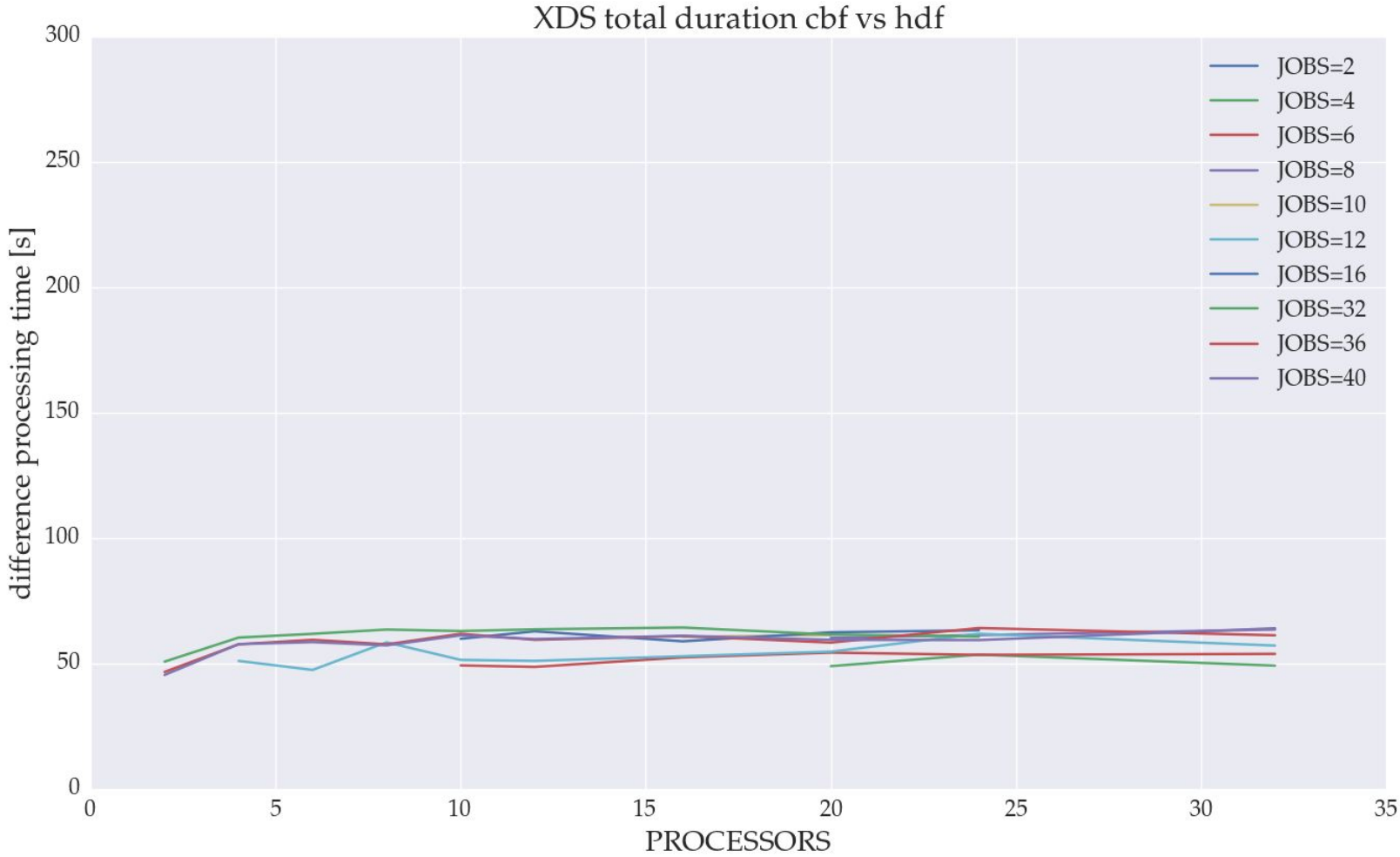
- The dataset was converted to CBF format and analogous set of processing runs executed

CBF = 45% speedier!

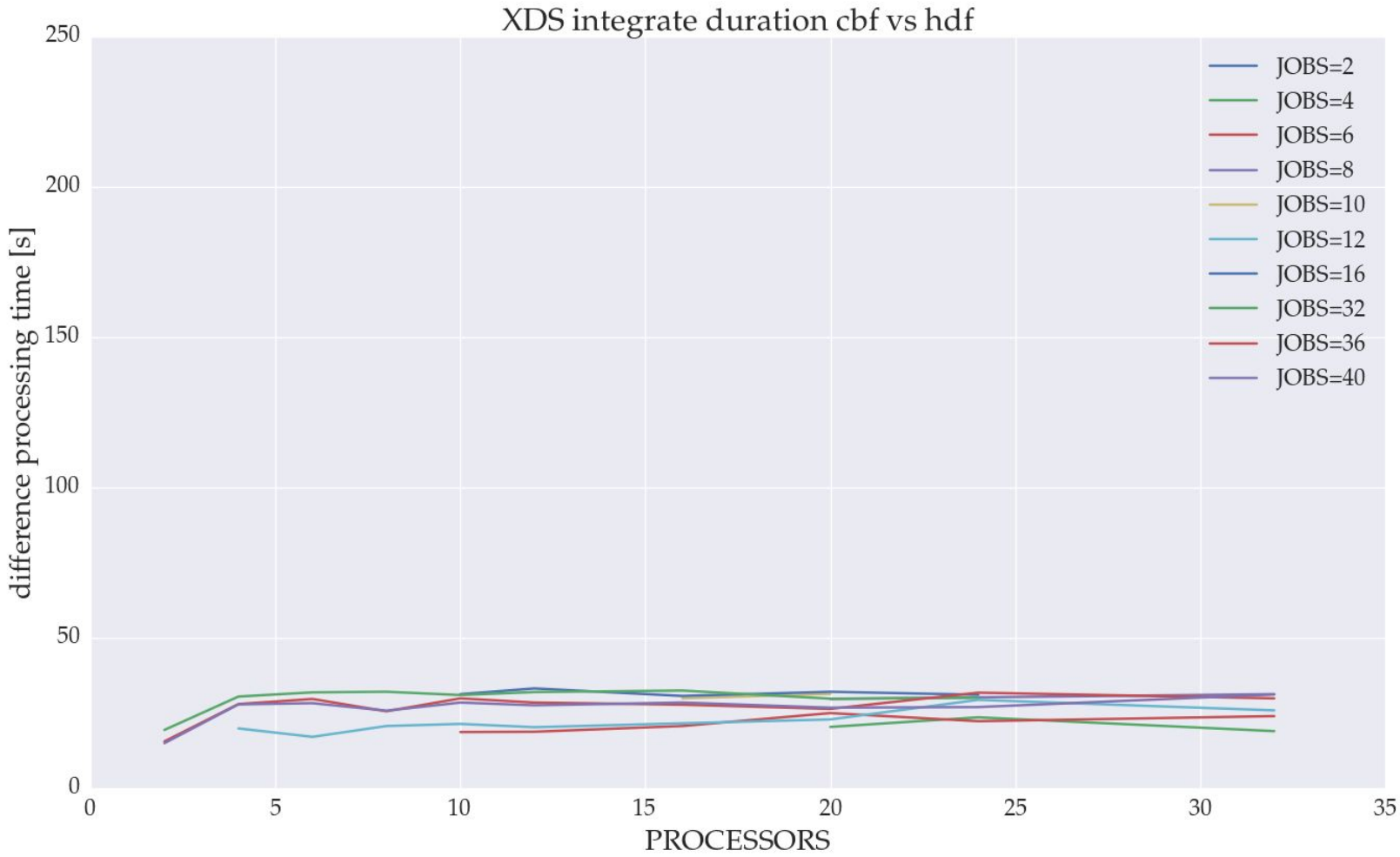
CBF dataset



CBF vs HDF


XDS total duration cbf vs hdf
difference processing time [s]
PROCESSORS
0
50
100
150
200
250
300
0
5
10
15
20
25
30
35
JOBS=2
JOBS=4
JOBS=6
JOBS=8
JOBS=10
JOBS=12
JOBS=16
JOBS=32
JOBS=36
JOBS=40

CBF vs HDF

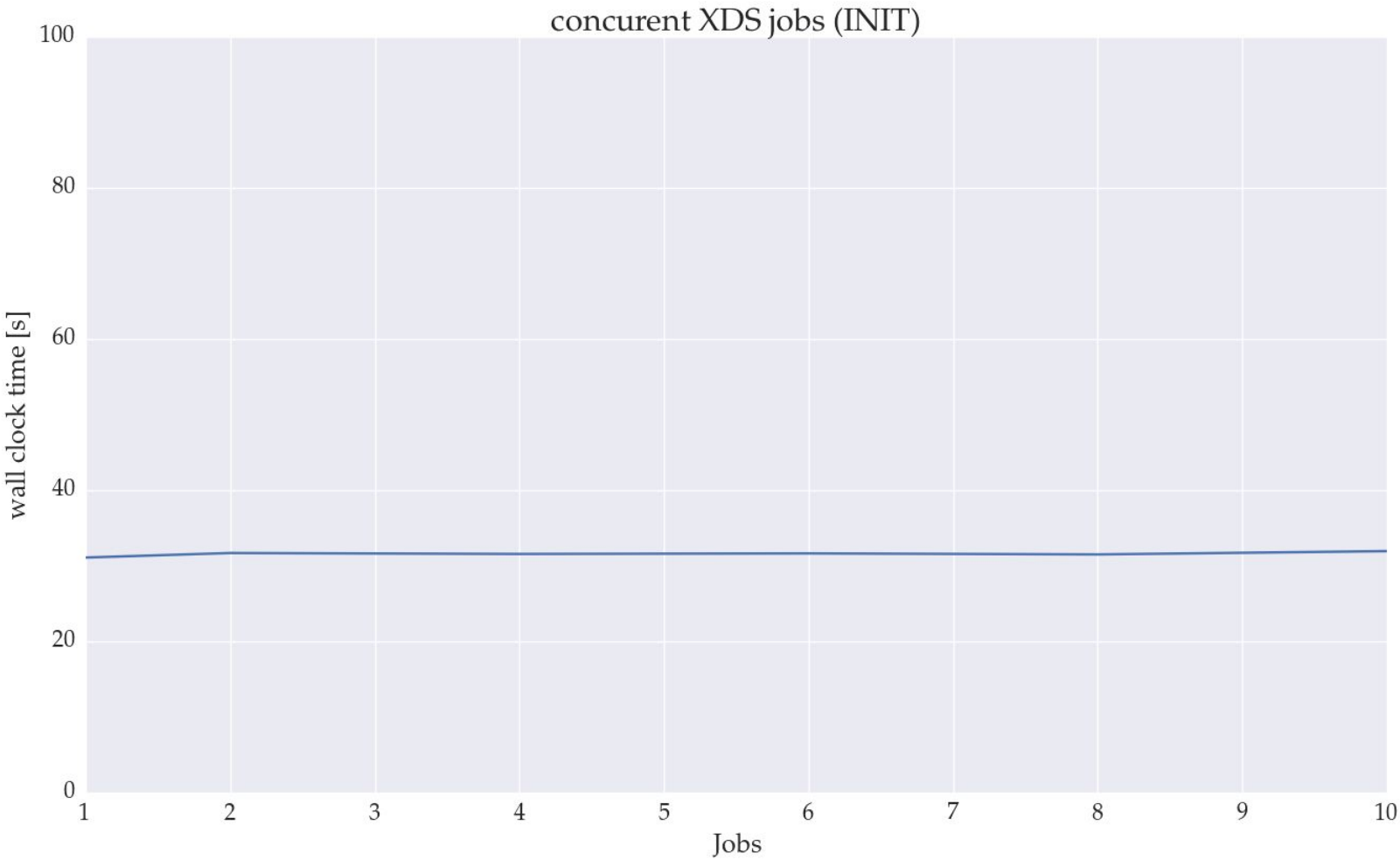

XDS integrate duration cbf vs hdf
difference processing time [s]
PROCESSORS
0
50
100
150
200
250
0
5
10
15
20
25
30
35
JOBS=2
JOBS=4
JOBS=6
JOBS=8
JOBS=10
JOBS=12
JOBS=16
JOBS=32
JOBS=36
JOBS=40

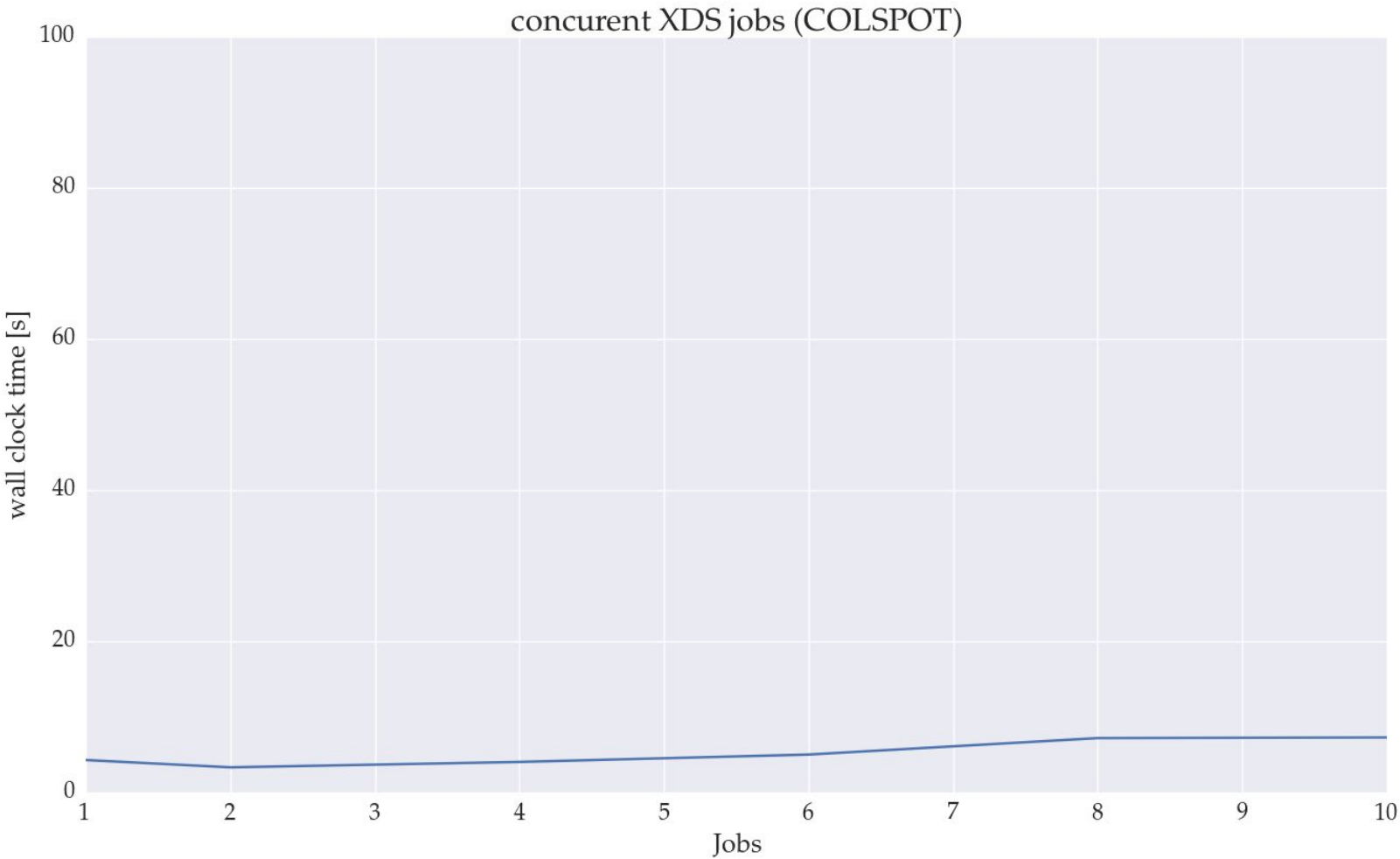
Conclusions

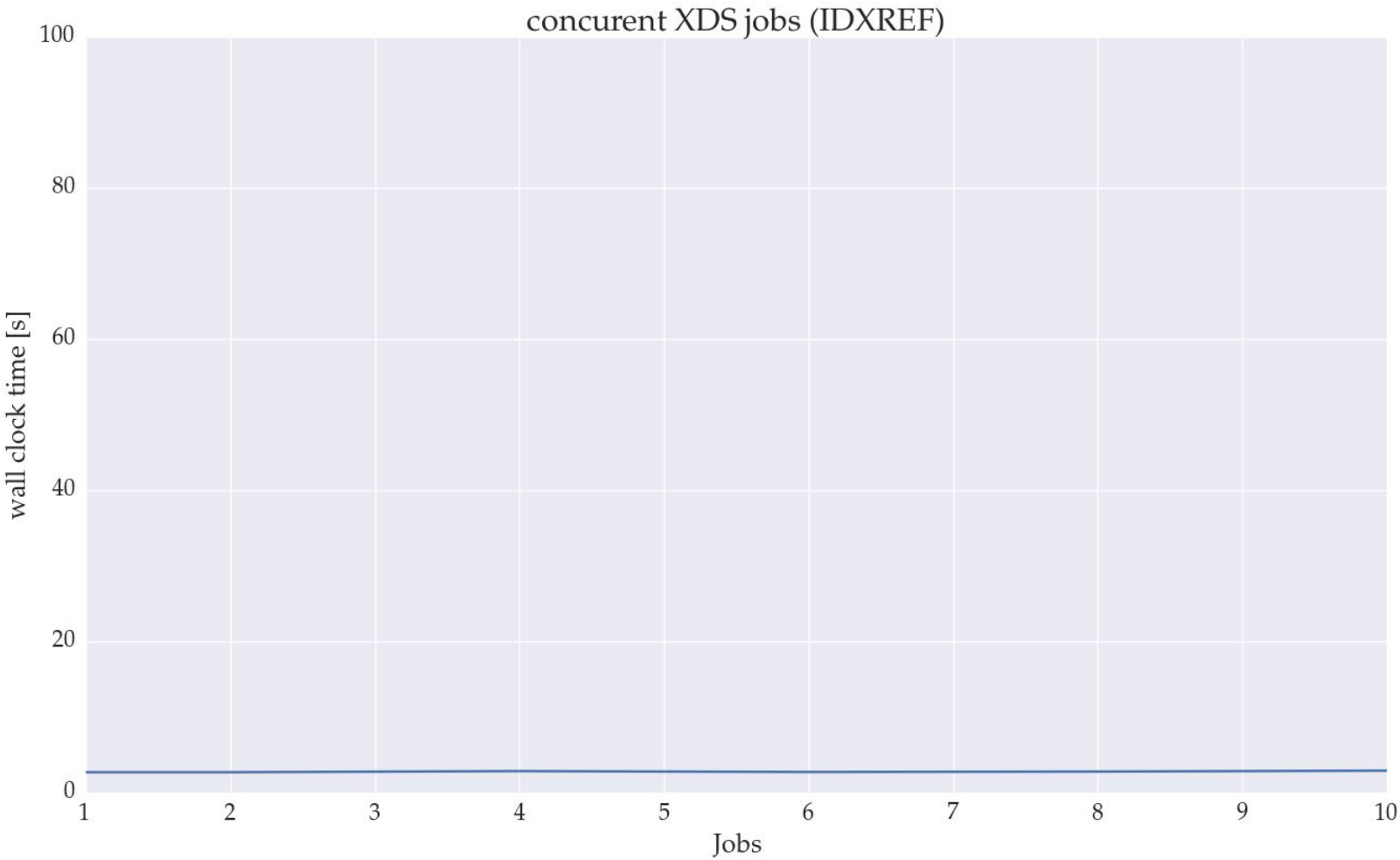
- Processing with CBF is faster by ~45 % across evaluated parameters
- Penalty mostly in `INIT` (20%) and `INTEGRATE` (25%) steps
- The conversion time 36s -- worth doing even for a single XDS run

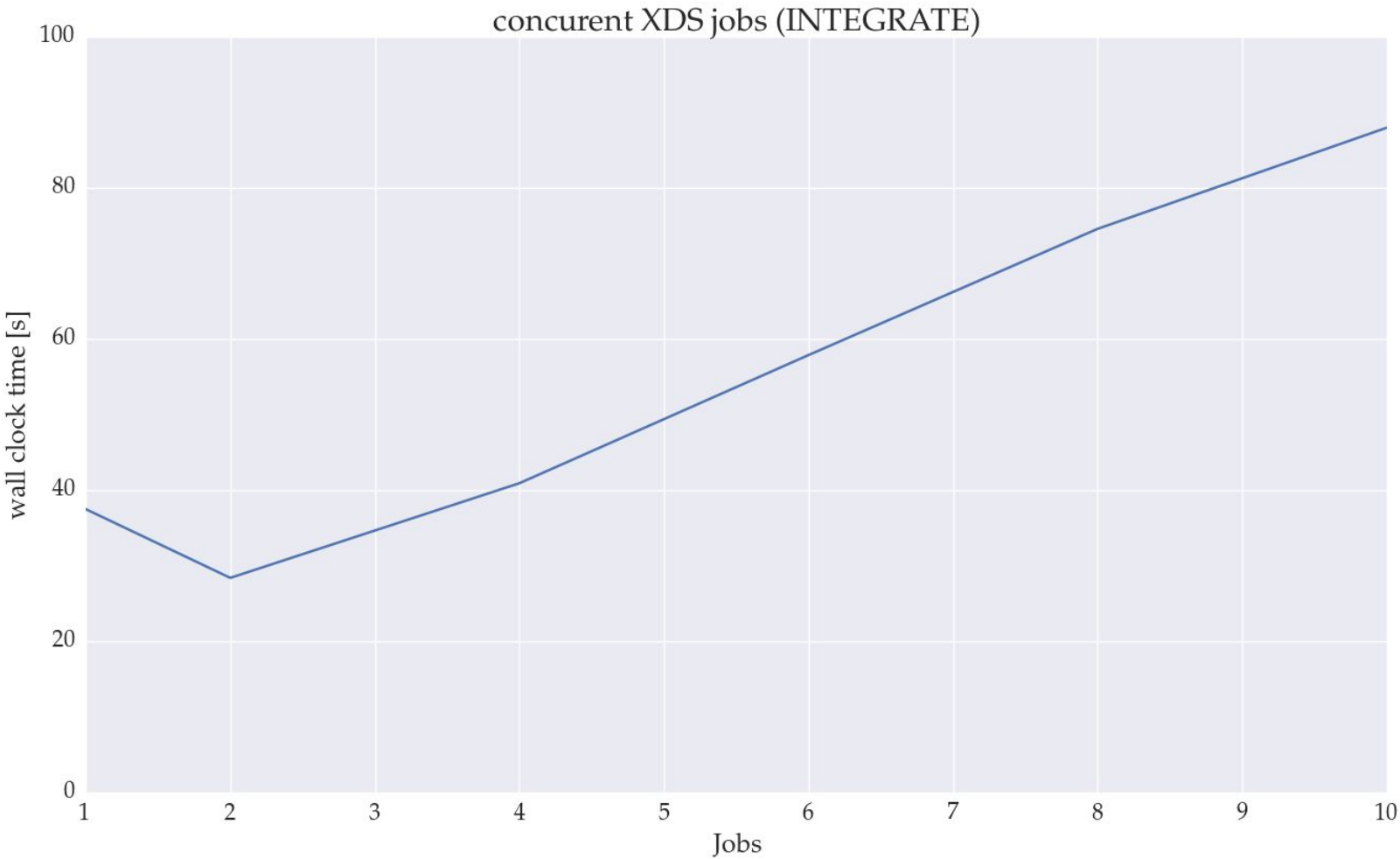
Concurrent XDS processing

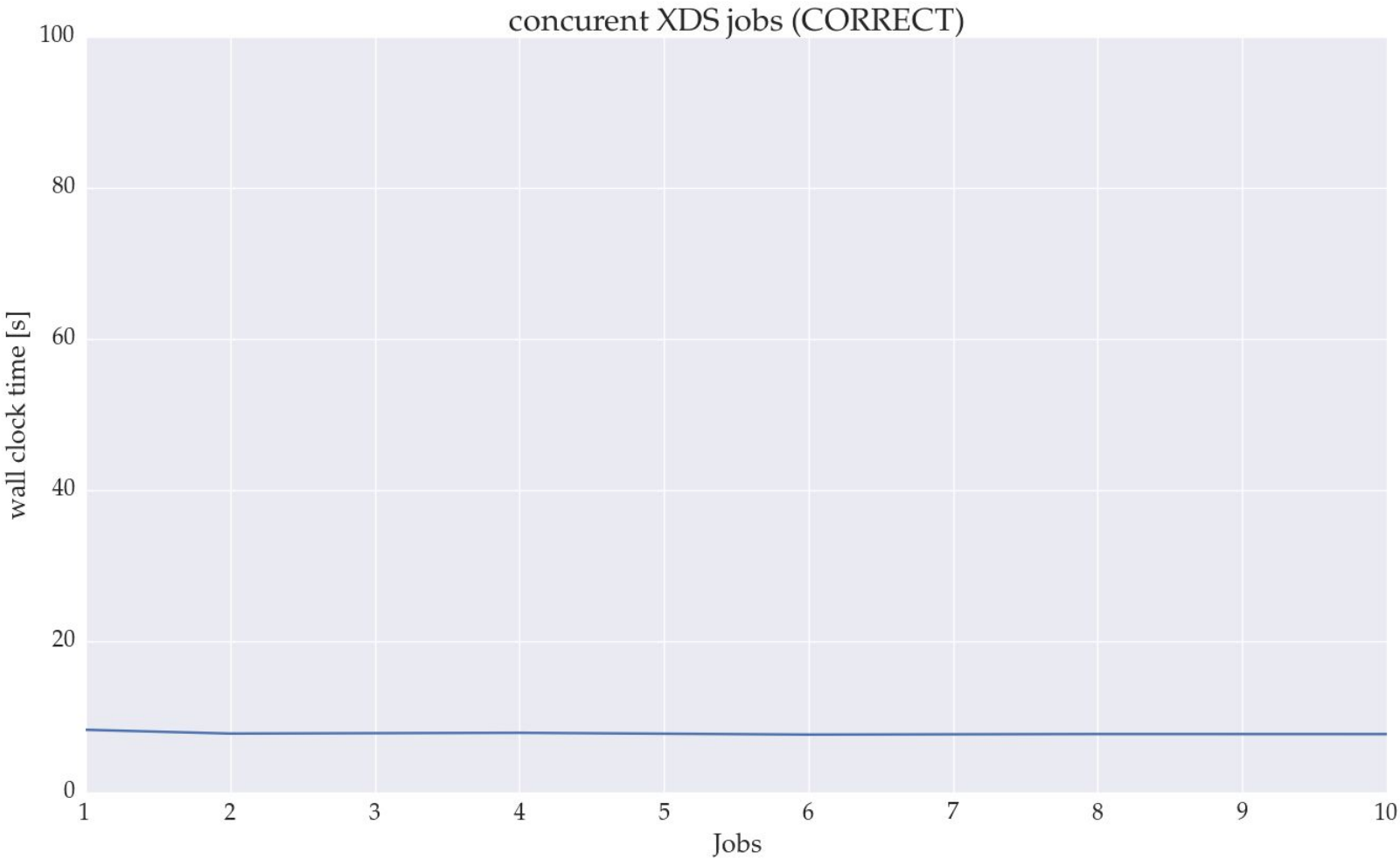
- Choose `MAXIMUM_NUMBER_OF_PROCESSORS` AND `_JOBS` close to optimum: `6,6` for our machine
- How many jobs can we run at the same time ?


concurent XDS jobs (INIT)
wall clock time [s]
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
Jobs


concurent XDS jobs (COLSPOT)
wall clock time [s]
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
Jobs


concurent XDS jobs (IDXREF)
wall clock time [s]
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
Jobs


concurent XDS jobs (INTEGRATE)
wall clock time [s]
Jobs
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10


concurent XDS jobs (CORRECT)
wall clock time [s]
100
80
60
40
20
0
1
2
3
4
5
6
7
8
9
10
Jobs

Concurrent XDS processing

- Running up to 10 concurrent XDS jobs is still efficient on this system
- What is the point at which it is better to serialize tasks?
- NUMA control ?

For Steady Processing Performance

```
# sync; echo 3 > /proc/sys/vm/drop_caches
```

Acknowledgements

- Bill Shepard
- Gavin Fox
- Enrico Stura (CEA)
- Laurent Gadea
- Patrick Gourhant
- Pierre Legrand
- Leo Chavas
- Arkadiusz Dawiec
- Philippe Martinez
- Idrissou Chado
- Stephane Le
- Alain Buteau
- Arafat Nouredinne
- David von Stetten
- Sebastien Petitdemange
- Paul-Antoine Douissard