Karlsruhe Institute of Technology

UFO + Concert – High-throughput data stream processing

Matthias Vogelgesang

POOW '14 — PaNdata ODI Open Workshop

What do we try to do?

In-situ and *in-vivo* experiments via

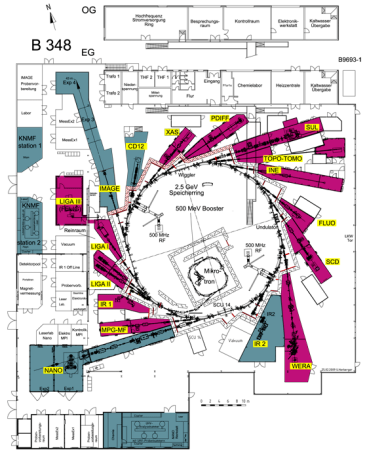
New high-speed X-ray radio- and tomography beamline at ANKA

High-speed camera streaming

RT reconstruction and analysis

On-line experiment control

Fast data storage



Main objectives

Technical: keep stream in memory, process fast

Sociological: include *all* users

DATA ACQUISITION

Data acquisition

Requirements

Generic access to all employed cameras

Low latencies, high throughput

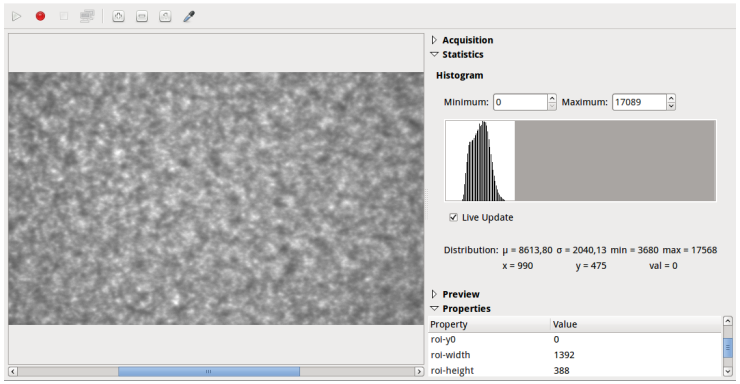
Fast remote access

How?

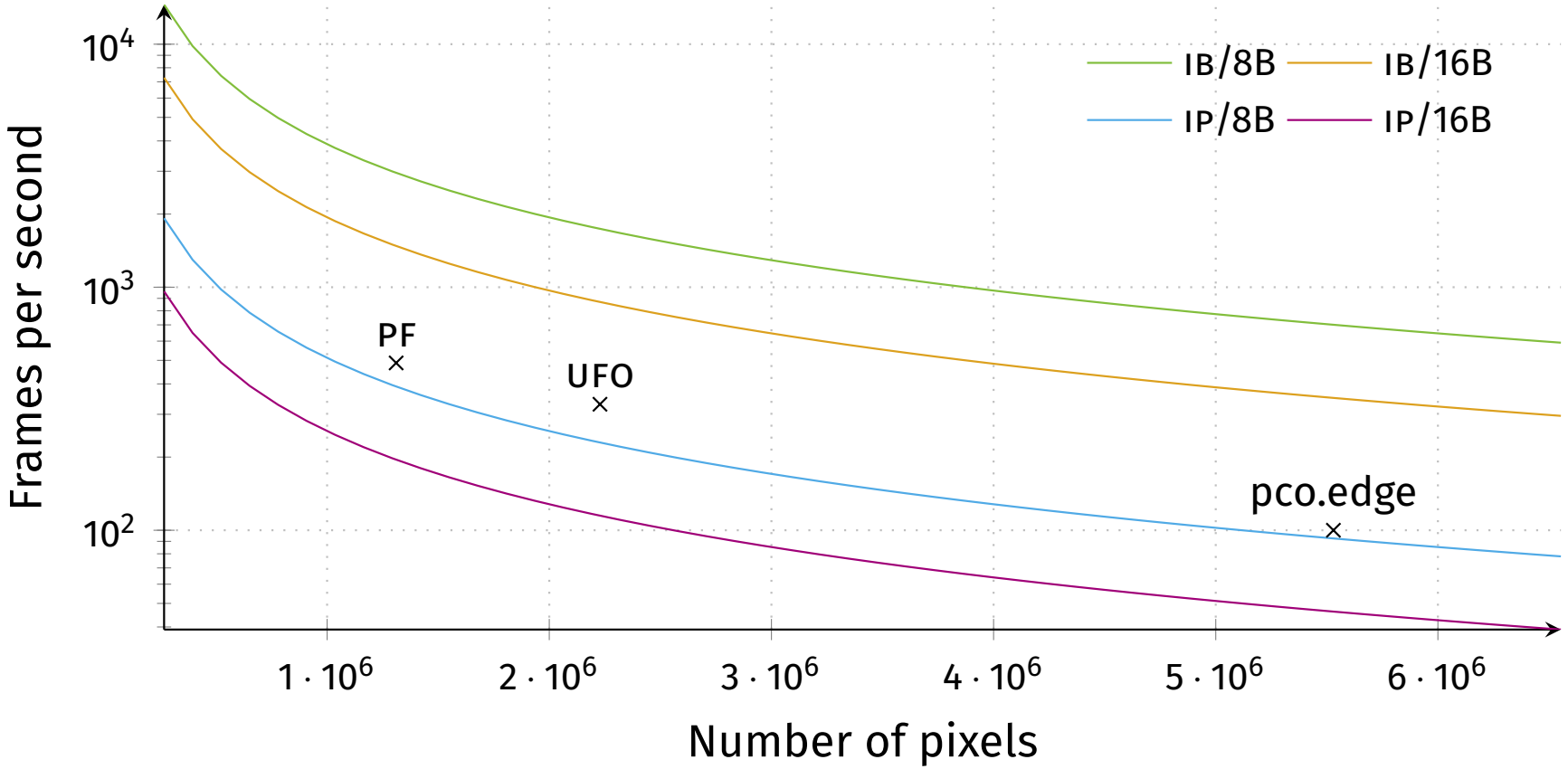
Thin API through low-level C library

Zero-copy data transfers

Remote control via TANGO, data transfer via InfiniBand protocol



InfiniBand streaming



DATA PROCESSING

Heterogeneous data processing

Requirements

Process image streams on the fly

Use heterogeneous compute systems

How?

Define tasks of work

Connect processing workflows

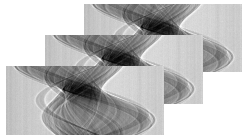
Let a run-time schedule tasks

Heterogeneous data processing

Requirements

Process image streams on the fly

Use heterogeneous compute systems



How?

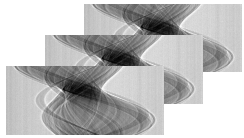
Define tasks of work

Connect processing workflows

Let a run-time schedule tasks

Heterogeneous data processing

Requirements



Process image streams on the fly

Use heterogeneous compute systems

How?

Define tasks of work

Connect processing workflows

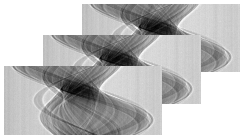
Let a run-time schedule tasks

Heterogeneous data processing

Requirements

Process image streams on the fly

Use heterogeneous compute systems



How?

Define tasks of work

Connect processing workflows

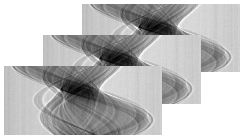
Let a run-time schedule tasks

Heterogeneous data processing

Requirements

Process image streams on the fly

Use heterogeneous compute systems



How?

Define tasks of work

Connect processing workflows

Let a run-time schedule tasks

Heterogeneous data processing

Requirements

Process image streams on the fly

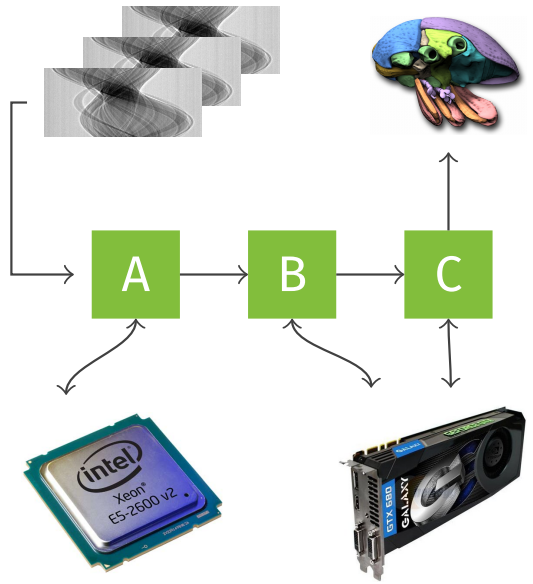
Use heterogeneous compute systems

How?

Define tasks of work

Connect processing workflows

Let a run-time schedule tasks



Integration

Written as a framework in C + OpenCL ...

Low-level access

```
from ufo import (Reader,
                 Backproject,
                 Writer)

read = Reader(path='/mnt/raid/scan16')
reco = Backproject(axis=1021.54)
write = Writer(path='/mnt/raid/scan16')

write(reco(read())).run().wait()
```

High-level access

```
for result in reco(read()):
    print(np.mean(result))
```

High-level GPU extension

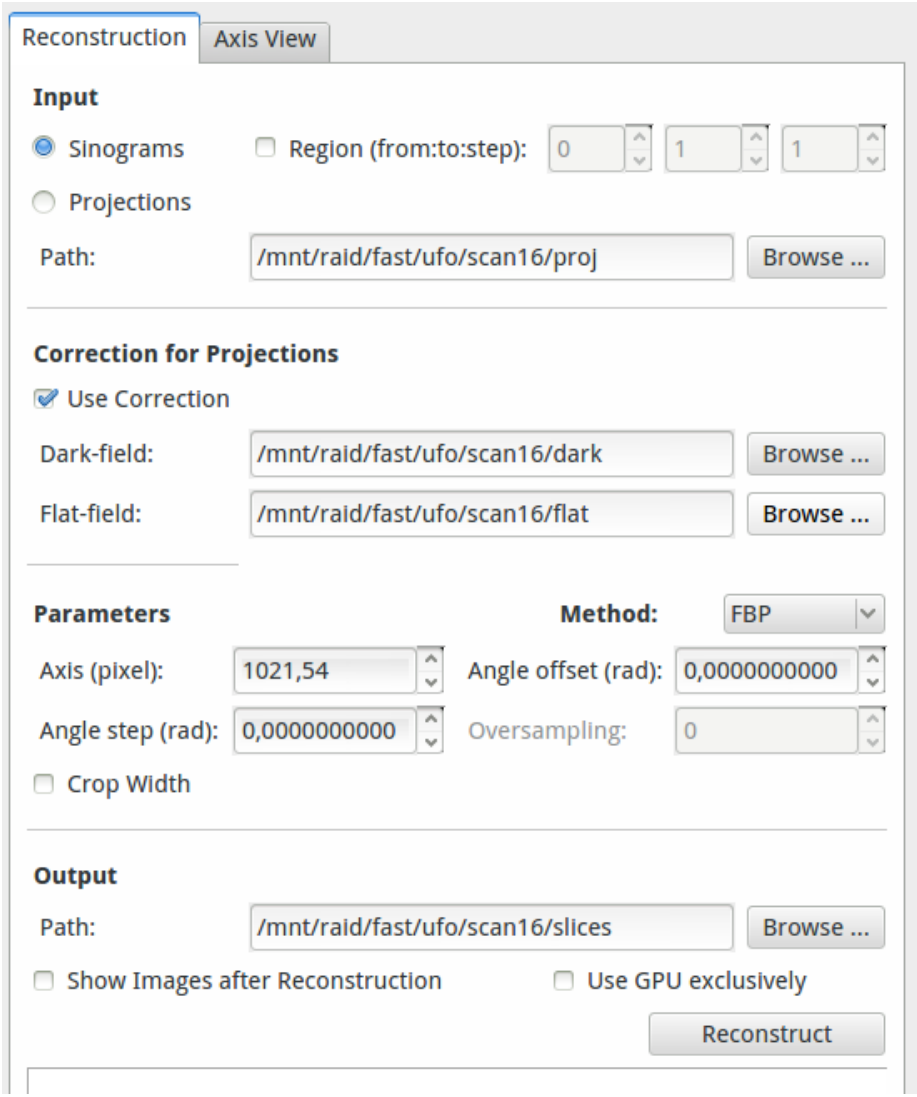
Writing GPU kernel code can be a pain, so we wrote a Python-to-OpenCL translator ...

```
@jit
def saxpy(a, b, y):
    return a * b + y
```

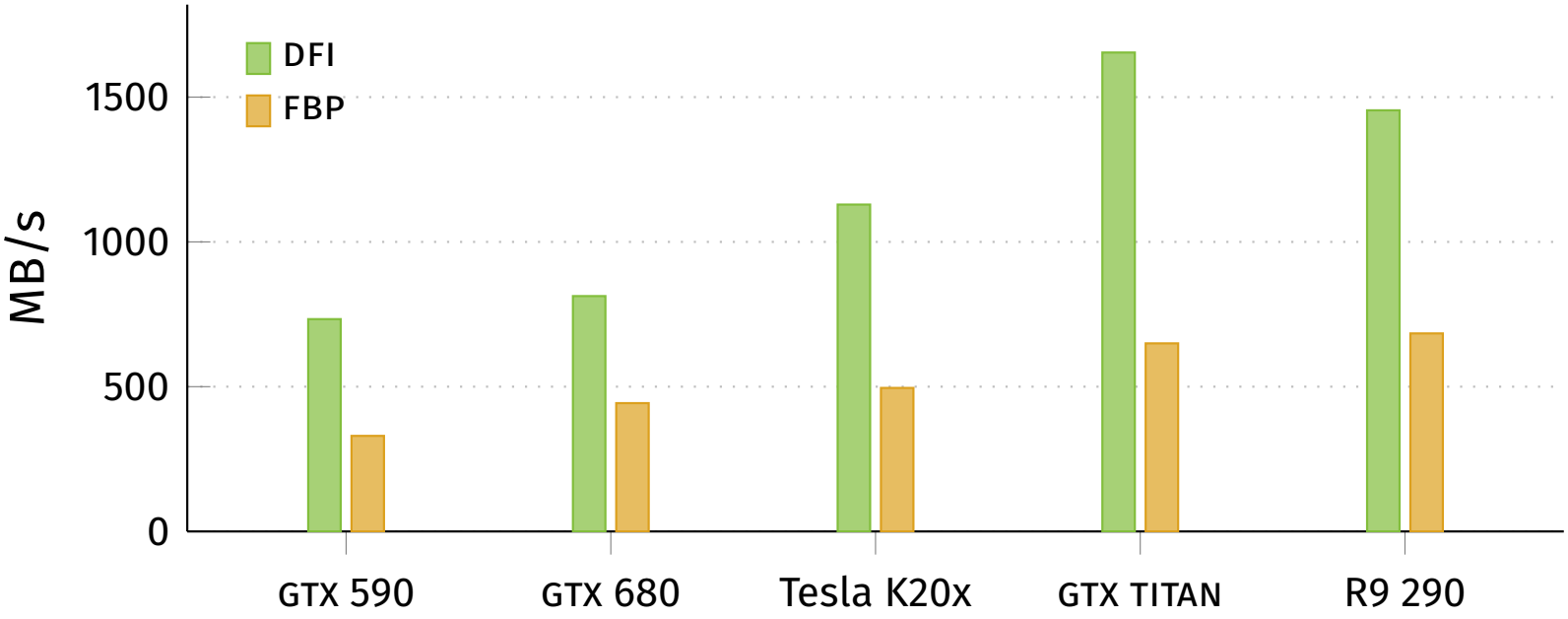
Notes

- Dynamic data type introspection
- AST optimizations
- Used stand-alone or with heterogeneous compute environment

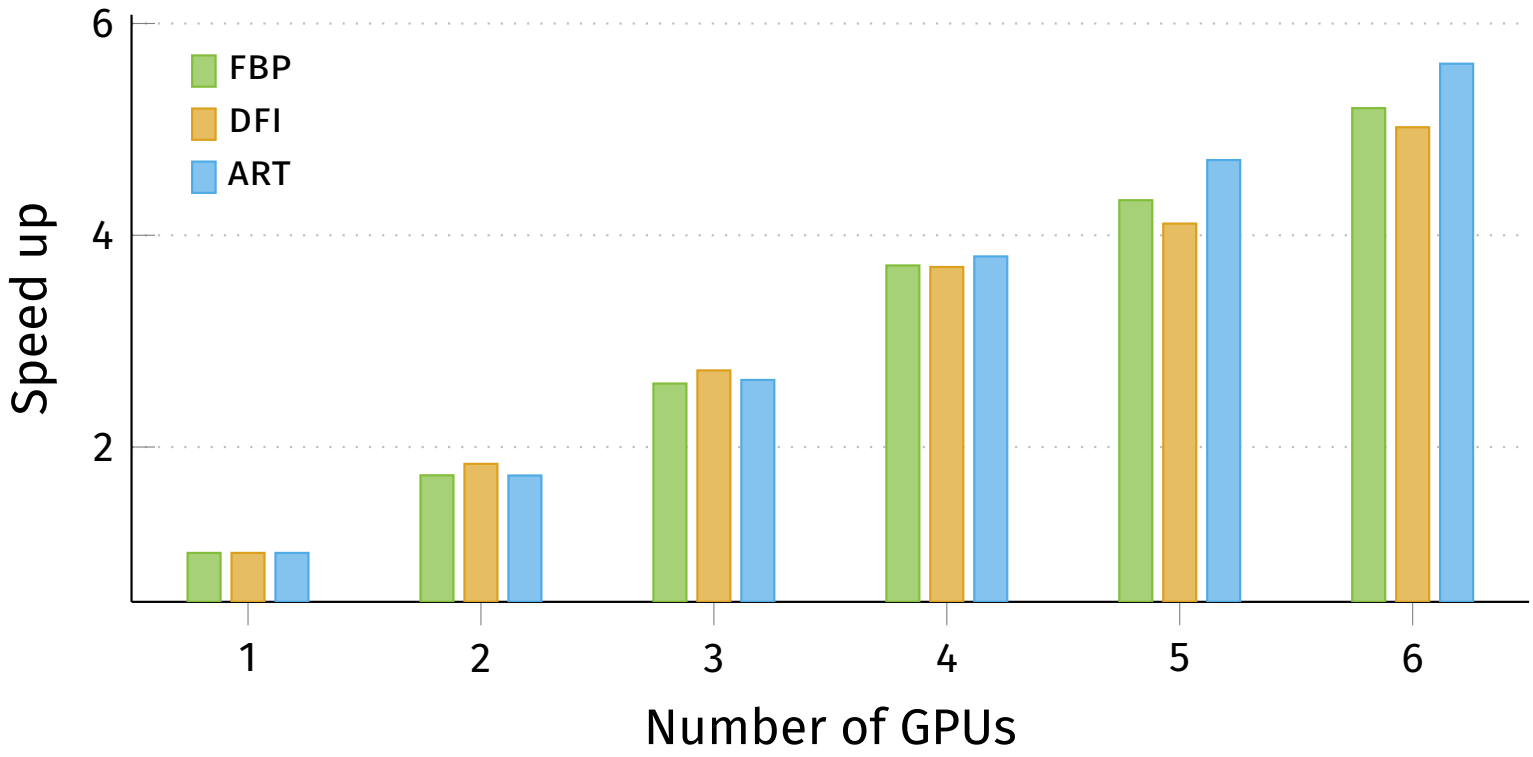
Application-specific interfaces



Reconstruction throughput



Reconstruction scalability



CONTROL

Experiment control

Requirements

- Process streamed data and control experiment
- Provide high-level interface
- Direct access to devices and compute resources

How?

- Python experiment control system *Concert*
- Provides in-process access to data and devices
- Wraps device access in future* objects
- Wraps compute pipelines in coroutines

* either native Python threads or Greenlets

Asynchronous I/O and compute

Why futures?

Concurrency model suitable for compute and device access

Different implementations available

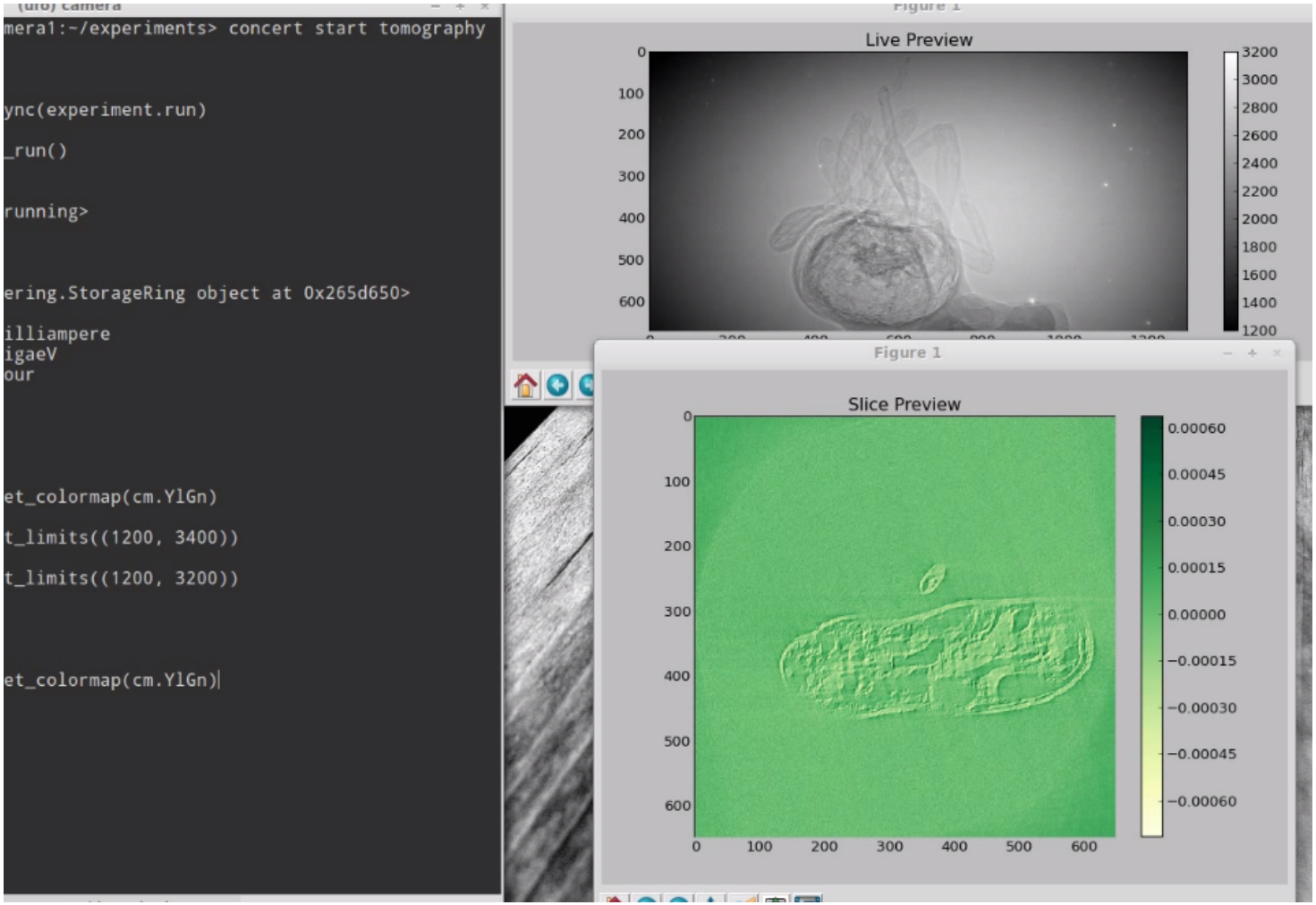
Why coroutines?

Low resource requirements

Simplifies process structuring

Enhances decoupling and modularity

Online reconstruction



Storage

Requirements

Store raw and processed data on-the-fly

$\approx$ 1250 MB/s with pco.edge

Get semantics right

Throughput

Method*	Disk	SSD
TIFF/Single	77	1129
TIFF/Multi	135	1157
HDF5/Array	137	1364
HDF5/Groups	138	1398
dd	—	2192

* libhdf 1.8.11 and libtiff 4.0.3

Storage

Requirements

Store raw and processed data on-the-fly

$\approx$ 1250 MB/s with pco.edge

Get semantics right

Throughput

Method*	Disk	SSD
TIFF/Single	77	1129
TIFF/Multi	135	1157
HDF5/Array	137	1364
HDF5/Groups	138	1398
dd	—	2192

* libhdf 1.8.11 and libtiff 4.0.3

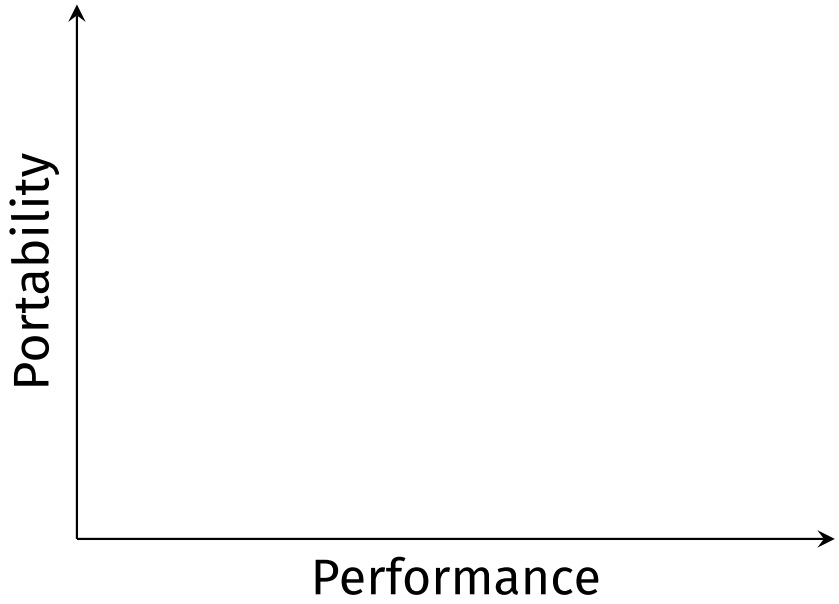
Performance is not a real argument for either format but ...

Comparison

Single TIFFs are portable but slow

Multi-page TIFFs have potential 2 GB limit

HDF5 is fastest but high-level tooling is lacking

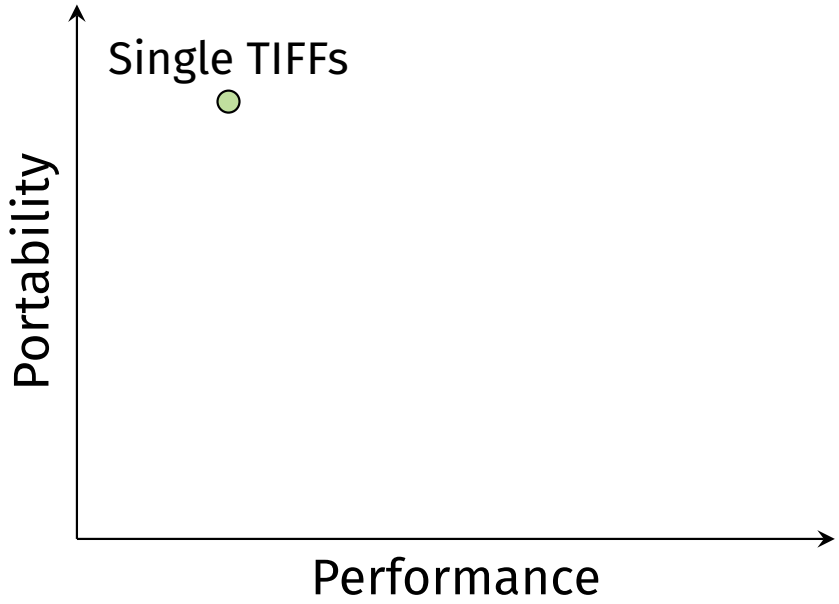


Comparison

Single TIFFs are portable but slow

Multi-page TIFFs have potential 2 GB limit

HDF5 is fastest but high-level tooling is lacking

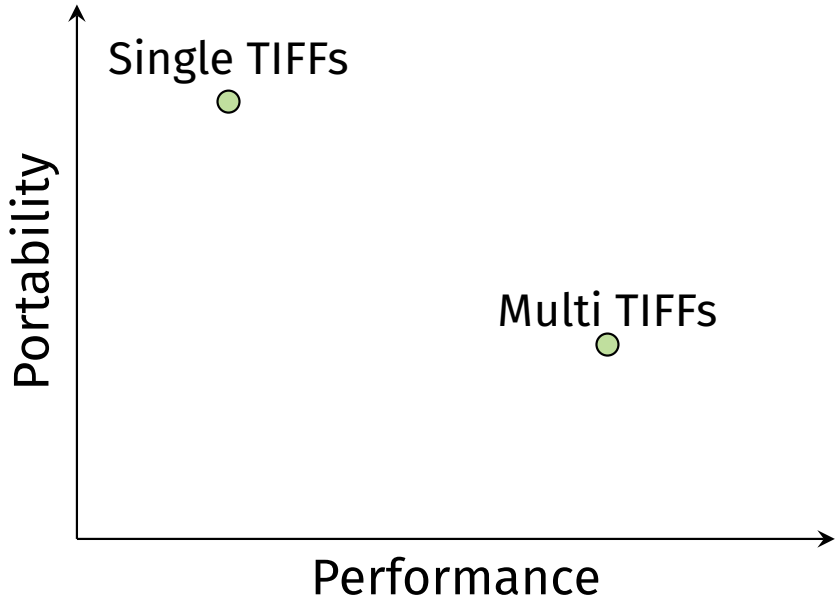


Comparison

Single TIFFs are portable but slow

Multi-page TIFFs have potential 2 GB limit

HDF5 is fastest but high-level tooling is lacking

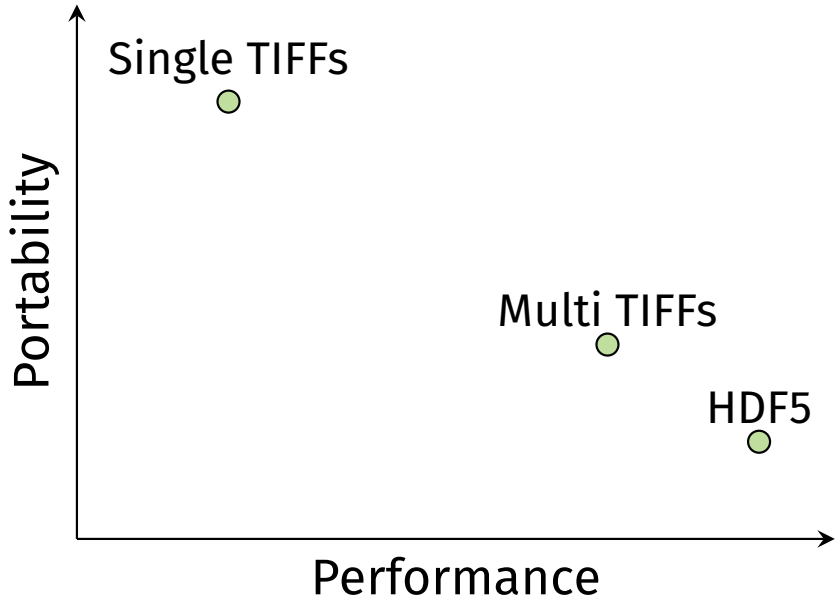


Comparison

Single TIFFs are portable but slow

Multi-page TIFFs have potential 2 GB limit

HDF5 is fastest but high-level tooling is lacking

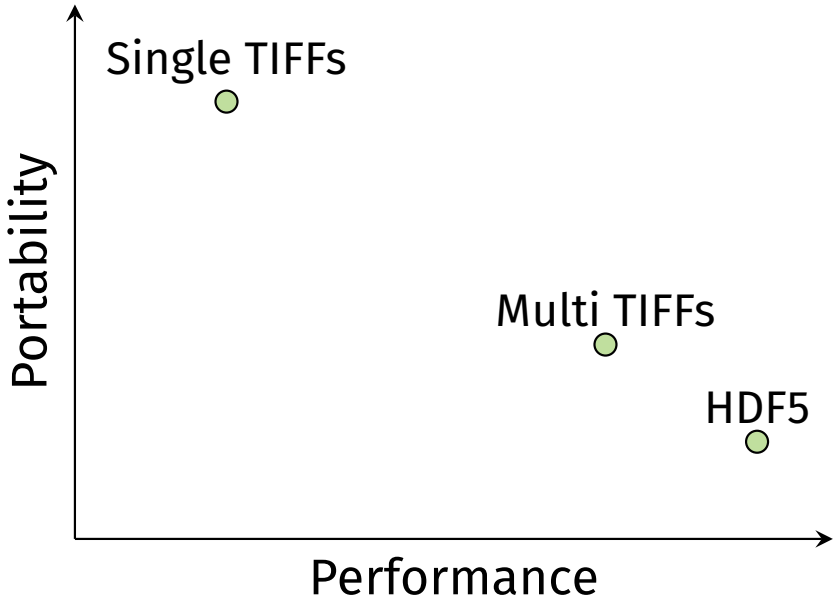


Comparison

Single TIFFs are portable but slow

Multi-page TIFFs have potential 2 GB limit

HDF5 is fastest but high-level tooling is lacking or not?



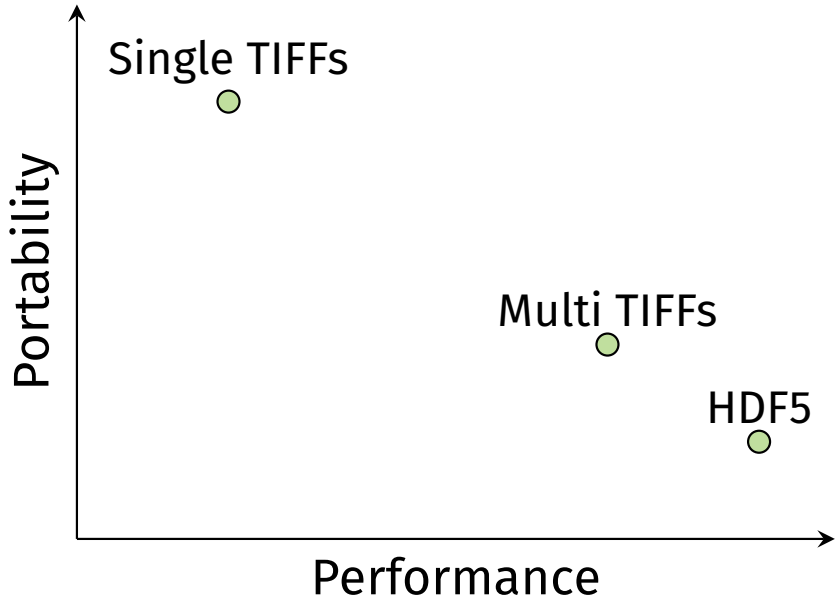
Comparison

Single TIFFs are portable but slow

Multi-page TIFFs have potential 2 GB limit

HDF5 is fastest but high-level tooling is lacking or not?

→ Abstract away the differences



Data structuring

Meta data hierarchies

Abstract directory and group hierarchies as walkers

```
walker.ascend('scan')
stream(walker.write())
```

Writes structured TIFF files or HDF5 dsets

Data structuring

Meta data hierarchies

Abstract directory and group hierarchies as walkers

```
walker.ascend('scan')
stream(walker.write())
```

Writes structured TIFF files or HDF5 dsets

So ...

Solves technical problems

...but still no idea about the structure

CONCLUSION

Karlsruhe Institute of Technology

Summary

Local and remote high throughput data acquisition

Summary

Local and remote high throughput data acquisition ✓

Scalable heterogeneous computing

Karlsruhe Institute of Technology

Summary

Local and remote high throughput data acquisition ✓

Scalable heterogeneous computing ✓

Flexible control

Summary

- Local and remote high throughput data acquisition ✓
- Scalable heterogeneous computing ✓
- Flexible control ✓
- Data and meta data storage

Summary

- Local and remote high throughput data acquisition ✓
- Scalable heterogeneous computing ✓
- Flexible control ✓
- Data and meta data storage ?

Summary

- Local and remote high throughput data acquisition ✓
- Scalable heterogeneous computing ✓
- Flexible control ✓
- Data and meta data storage ?

More information

- Data acquisition: `github.com/ufo-kit/libuca`
- Compute framework: `github.com/ufo-kit/ufo-core`
- Control system: `github.com/ufo-kit/concert`
- Python/OpenCL: `github.com/ufo-kit/pina`