Karlsruhe Institute of Technology

High-speed tomography with UFO

Matthias Vogelgesang et al.

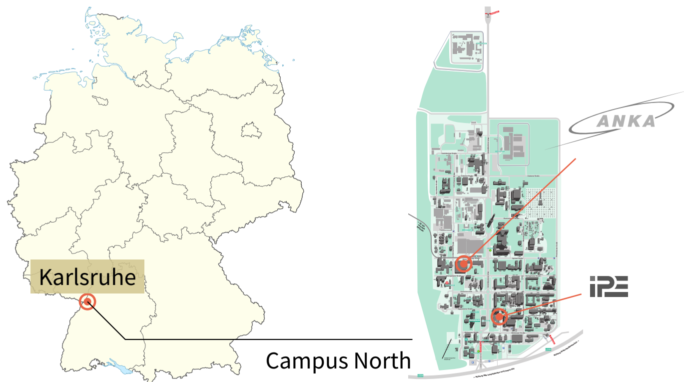
matthias.vogelgesang@kit.edu

Institute for Data Processing and Electronics

Karlsruhe Institute of Technology

KIT Karlsruhe Institute of Technology

Karlsruhe Institute of Technology



UFO at a glance

Karlsruhe Institute of Technology

- A **library** not a **program** to define **general purpose** processing pipelines and graphs
- Pipelines are composed of individual building blocks
- **Real-time** focus due to streamed data processing

- C and Python APIs, tools built on top
- Runs (at least) on Linux and MacOS
- Is free and open source

Falling for the NIH syndrome?

Karlsruhe Institute of Technology

Requirements

- High-throughput and real-time X-ray imaging
- Exclusive access to compute node with one or more GPUs
- Automation of the entire data processing pipeline

Falling for the NIH syndrome?

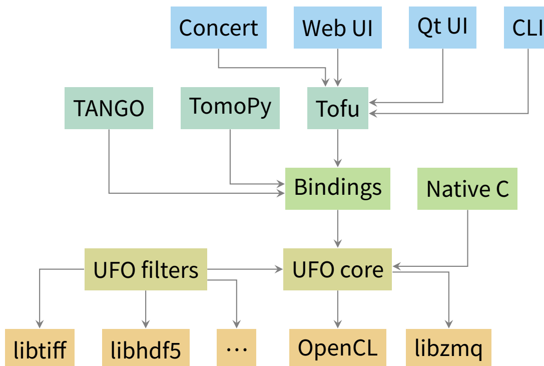
Karlsruhe Institute of Technology

Requirements

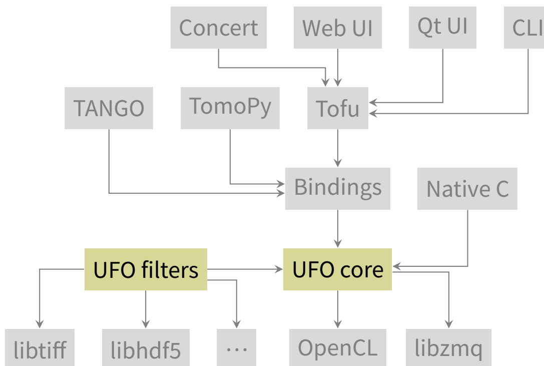
- High-throughput and real-time X-ray imaging
- Exclusive access to compute node with one or more GPUs
- Automation of the entire data processing pipeline

In 2010, no open source solution in sight that could fit the bill

Where are we now



Where are we now



Architecture

Karlsruhe Institute of Technology

Core framework

- Single C library based on GObject and OpenCL
- Manages hardware resources
- Loads and schedules task plugins

Plugins

- Implemented as shared libraries with a defined interface
- Produce, process, reduce or consume data streams
- State depends solely on input data and internal parameters

Selection of plugins

Karlsruhe Institute of Technology

Input and output

- Image I/O (Raw, TIFF, HDF5)
- In-memory readers/writers
- stdin/stdout

Pre- and post-processing

- Flat-field-correction
- Denoising and artefact removal
- Averaging and median filtering

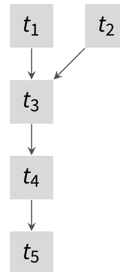
Processing

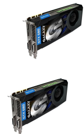
- Phase retrieval
- Tomographic reconstruction (FBP, Fourier-based and IR methods)
- Laminographic reconstruction (FBP and IR)

Streamed processing on multiple GPUs

Static assignment

- Partitioning of tasks
- Depth-first search for path identification
- Insertion of duplicates
- Mapping and execution





Benefits

- Avoids unnecessary data transfers
- Scales pretty well

Streamed processing on multiple GPUs

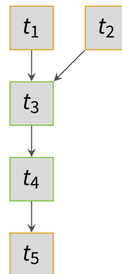
Karlsruhe Institute of Technology

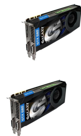
Static assignment

- Partitioning of tasks
- Depth-first search for path identification
- Insertion of duplicates
- Mapping and execution

Benefits

- Avoids unnecessary data transfers
- Scales pretty well





Streamed processing on multiple GPUs

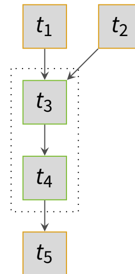
Karlsruhe Institute of Technology

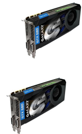
Static assignment

- Partitioning of tasks
- Depth-first search for path identification
- Insertion of duplicates
- Mapping and execution

Benefits

- Avoids unnecessary data transfers
- Scales pretty well



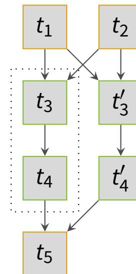


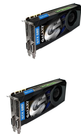
Streamed processing on multiple GPUs

Karlsruhe Institute of Technology

Static assignment

- Partitioning of tasks
- Depth-first search for path identification
- Insertion of duplicates
- Mapping and execution





Benefits

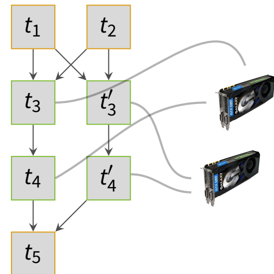
- Avoids unnecessary data transfers
- Scales pretty well

Streamed processing on multiple GPUs

Karlsruhe Institute of Technology

Static assignment

- Partitioning of tasks
- Depth-first search for path identification
- Insertion of duplicates
- Mapping and execution



Benefits

- Avoids unnecessary data transfers
- Scales pretty well

Streamed processing within a cluster

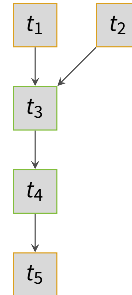
Karlsruhe Institute of Technology

Strategy

- Proxy task represents subpath
- Instantiate subpath remotely
- Send and receive input and output

Benefits

- Local multi GPU optimization
- Maps equally well to MPI and ZeroMQ



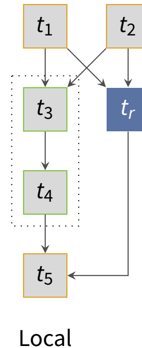
Streamed processing within a cluster

Strategy

- Proxy task represents subpath
- Instantiate subpath remotely
- Send and receive input and output

Benefits

- Local multi GPU optimization
- Maps equally well to MPI and ZeroMQ



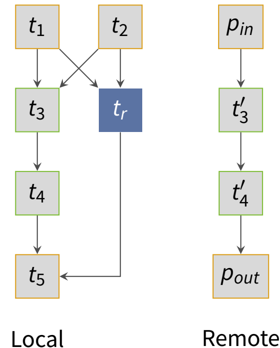
Streamed processing within a cluster

Strategy

- Proxy task represents subpath
- Instantiate subpath remotely
- Send and receive input and output

Benefits

- Local multi GPU optimization
- Maps equally well to MPI and ZeroMQ



Streamed processing within a cluster

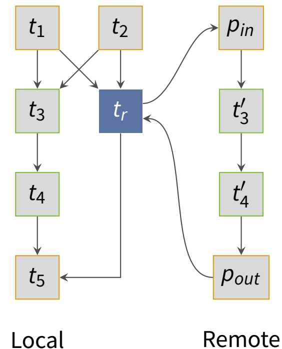
Karlsruhe Institute of Technology

Strategy

- Proxy task represents subpath
- Instantiate subpath remotely
- Send and receive input and output

Benefits

- Local multi GPU optimization
- Maps equally well to MPI and ZeroMQ



Improving latency and throughput

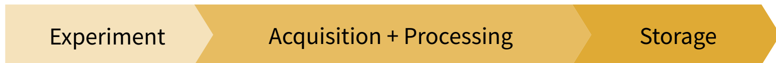
Karlsruhe Institute of Technology

- Real-time control requires low latencies and high throughput

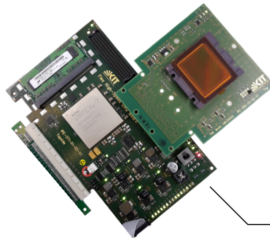
Improving latency and throughput

Karlsruhe Institute of Technology

- Real-time control requires low latencies and high throughput
- Reduce latencies by coupling acquisition and processing



Aside: our acquisition platform

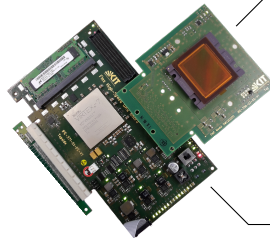


2 HPC FMC connectors

2 GB DDR3 memory

Xilinx Virtex 7 FPGA

Aside: our acquisition platform



CMOS image sensor

$5.5\,\mu m^2$ to $6.4\,\mu m^2$

up to 300 frames/s

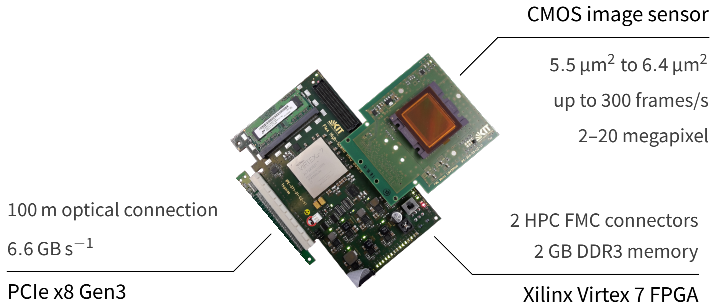
2–20 megapixel

2 HPC FMC connectors

2 GB DDR3 memory

Xilinx Virtex 7 FPGA

Aside: our acquisition platform



Direct memory access

Regular copy

Result

Direct memory access

DirectGMA

Result

Direct memory access with OpenCL

Karlsruhe Institute of Technology

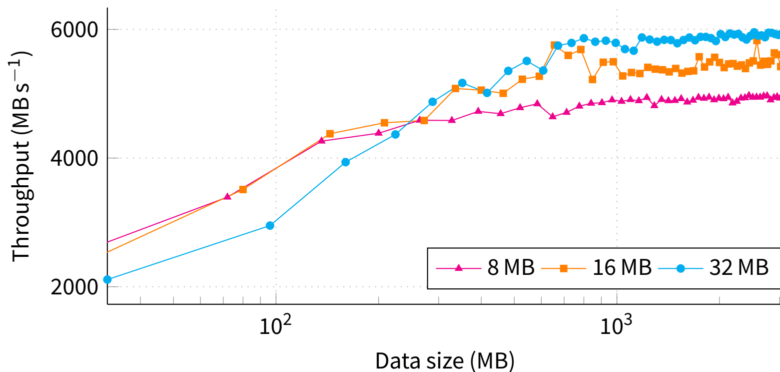
Writing to GPU

1. Create OpenCL buffer with `CL_MEM_BUS_ADDRESSABLE_AMD`
2. Determine address by passing buffer to `clEnqueueMakeBuffersResidentAMD()`
3. Set address and paging information in FPGA control registers
4. Busy-wait until FPGA flag signals completion

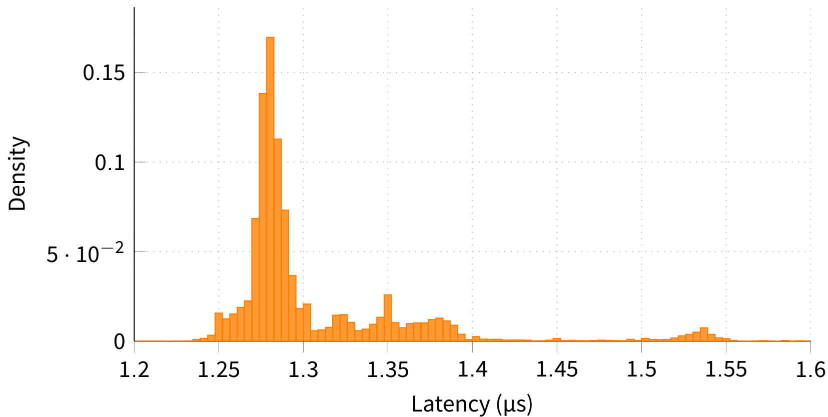
Writing to FPGA

1. Create OpenCL buffer with `CL_MEM_EXTERNAL_EXTERNAL_PHYSICAL_AMD`
2. Determine physical address of FPGA's "memory space"
3. Pass buffer and address to `clEnqueueMakeBuffersResidentAMD()`
4. Writes from the GPU are proxied transparently to the FPGA

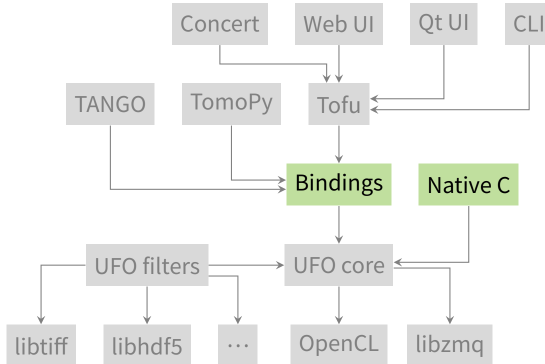
DirectGMA throughput



DirectGMA latency



Arbitrary pipeline construction



Running workflows the UNIX way

Karlsruhe Institute of Technology

Read and write data

```
ufo-launch read path=folder/sino*.tif !
    write filename=multi.tif
```

Running workflows the UNIX way

Karlsruhe Institute of Technology

Flat field correction

```
ufo-launch [read path=radios/, read path=darks/, read path=flats/] !
    flat-field-correct !
    write filename=output.h5:/correct
```

Running workflows the UNIX way

Karlsruhe Institute of Technology

Apply OpenCL expressions

```
ufo-launch [read path=radios/, read path=darks/, read path=flats/] !
    flat-field-correct !
    calculate expression="-log(v)" !
    write filename=output.h5:/log
```

Running workflows the UNIX way

Karlsruhe Institute of Technology

Remove vertical stripes

```
ufo-launch [read path=radios/, read path=darks/, read path=flats/] !
    flat-field-correct !
    calculate expression="-log(v)" !
    fft dimensions=2 ! filter-stripes ! ifft dimensions=2 !
    write filename=output.h5:/clean
```

Running workflows the UNIX way

Karlsruhe Institute of Technology

Compute filtered backprojection

```
ufo-launch [read path=radios/, read path=darks/, read path=flats/] !
    flat-field-correct !
    calculate expression="-log(v)" !
    fft dimensions=2 ! filter-stripes ! ifft dimensions=2 !
    fft ! filter ! ifft ! backproject !
    write filename=output.h5:/slices
```

Running workflows the UNIX way

Karlsruhe Institute of Technology

Going full circle

```
ufo-launch [read path=radios/, read path=darks/, read path=flats/] !
    flat-field-correct !
    calculate expression="-log(v)" !
    fft dimensions=2 ! filter-stripes ! ifft dimensions=2 !
    fft ! filter ! ifft ! backproject !
    stdout | gzip > reco.raw.gz
```

Language bindings

- Constructing the pipeline using the C API is possible but cumbersome
- Introspection mechanism enables third-party language integration …
- … for example JavaScript, Python, Ruby, Lua, Go and Haskell
- Our primary target for now is Python

Python bindings resemble C API

Karlsruhe Institute of Technology

```
from gi.repository import Ufo

pm = Ufo.PluginManager()
read = pm.get_task('read')
rescale = pm.get_task('rescale')
write = pm.get_task('write')

read.set_properties(path='folder/sino*.tif')
rescale.set_properties(factor=0.5)
write.set_properties(filename='output.h5:/raw')

g = Ufo.TaskGraph()
g.connect_nodes(read, rescale)
g.connect_nodes(rescale, write)

sched = Ufo.Scheduler()
sched.run(g)
```

Improving Python support

Karlsruhe Institute of Technology

Unlocking the Global Interpreter Lock

- GIL would block Python interpreter during computation
- GIL is released during execution and insertion of data

Interfacing with NumPy

- C module converts between UFO and NumPy
- Alternatively data pointers can be re-used

High-level abstractions

- `ufo` module wraps filters during import
- More magic but cleaner instantation and setup

High-level Python

Karlsruhe Institute of Technology

```python
from ufo import Read, Write, Rescale

read = Read(path='folder/sino*.tif')
rescale = Rescale(factor=0.5)
write = Write(filename='output.h5:/raw')

# wait for execution to finish
write(rescale(read())).run().wait()
```

High-level Python

Karlsruhe Institute of Technology

```
from ufo import Read, Rescale

read = Read(path='folder/sino*.tif')
rescale = Rescale(factor=0.5)

# use result immediately
for image in rescale(read()):
    print(np.mean(image))
```

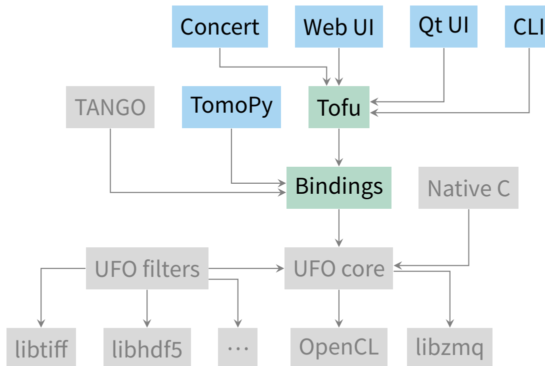
High-level Python

Karlsruhe Institute of Technology

```python
from ufo import Rescale

data = [np.ones((1024, 1024)) * i for i in range(10)]
rescale = Rescale(factor=0.5)

# insert NumPy arrays
for image in rescale(data):
    print(np.mean(image))
```


Concert
Web UI
Qt UI
CLI
TANGO
TomoPy
Tofu
Bindings
Native C
UFO filters
UFO core
libtiff
libhdf5
...
OpenCL
libzmq

Tofu reconstruction

Karlsruhe Institute of Technology

Idea

- Move reconstruction-related code to single Python module
- Simplify setup and execution of reconstruction pipelines using UFO
- Provide visualization widgets based on PyQtGraph

Focus

- Dark current and flat field correction
- Tomographic reconstruction with FBP, DFI and IR method
- Laminographic reconstruction with FBP
- Manual and automatic axis alignment

Command line interface

Karlsruhe Institute of Technology

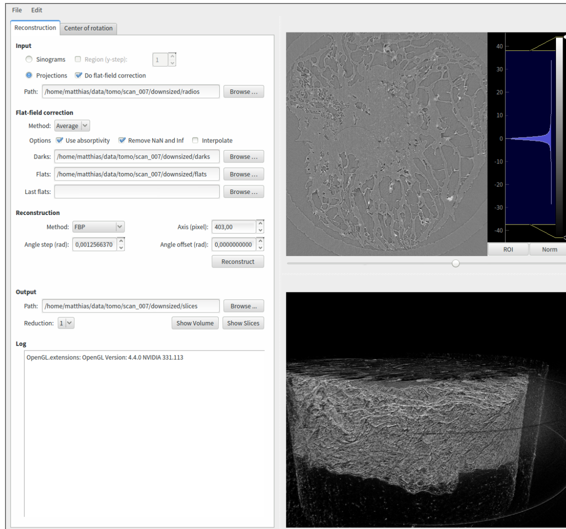
- Offline reconstruction for power users
- Parameters are stored in a configuration file

```
$ ufo-reconstruct init
$ vi reco.conf
$ ufo-reconstruct tomo
```

- Command line arguments can override parameters

```
$ ufo-reconstruct tomo --axis=234.5
```

Qt GUI



TomoPy + Gridrec


```
import tomopy, dxchange

proj, flat, dark, theta = dxchange.read_aps_32id('tooth.h5', sino=(0, 2))
proj = tomopy.normalize(proj, flat, dark)
proj = tomopy.minus_log(proj)
center = tomopy.find_center(proj, theta, init=290, ind=0, tol=0.5)
recon = tomopy.recon(proj, theta, center=center,
                     algorithm='gridrec')
```

TomoPy + UFO

Karlsruhe Institute of Technology

```python
import tomopy, dxchange, ufo.tomopy

proj, flat, dark, theta = dxchange.read_aps_32id('tooth.h5', sino=(0, 2))
proj = tomopy.normalize(proj, flat, dark)
proj = tomopy.minus_log(proj)
center = tomopy.find_center(proj, theta, init=290, ind=0, tol=0.5)
recon = tomopy.recon(proj, theta, center=center,
                     algorithm=ufo.tomopy.fbp, ncore=1)
```

Honest assessment

The Good

- Compute model typically fits X-ray imaging
- We can cover all aspects we come across
- Performance is good

- Complete manual, manpages and reference
- Clean, portable code
- Open development yet QA

Honest assessment

The Bad

- Streaming on cluster works but currently is a beta at best
- Larger-than-memory problems need higher level solution
- Getting contributors up to speed takes a while

Honest assessment

The Ugly

- openSUSE packaging is KIT-specific
- Awkward implementation of IR methods
- New plugins have to be written in C/OpenCL

Karlsruhe Institute of Technology

Questions?

Performance

KIT
Karlsruhe Institute of Technology

Projection size	Projections	Slices / s
512×2048	512	893
1024×2048	1024	238
2048×2048	2048	35

Measured on an older machine with two K20xm and four GTX Titan

Real-time quality assessment

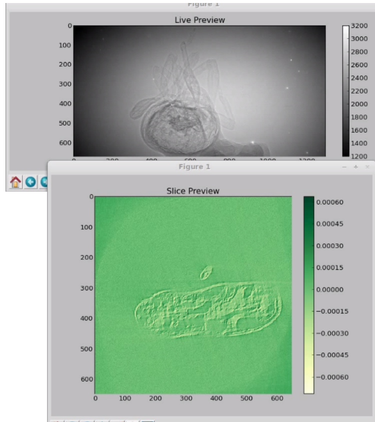


Figure : Live preview and updated slice