GPU-based data processing for ultrafast X-ray imaging — KSETA Plenary Workshop 2014

Matthias Vogelgesang

Institute for Data Processing and Electronics

Motivation

Synchrotron radiation from ANKA

- X-ray source producing beam with broad spectrum and high brilliance
- X-ray scattering and diffraction, high-resolution and high-speed radiography, **tomography** and laminography
- New IMAGE beamline used to investigate fast processes in life and material sciences

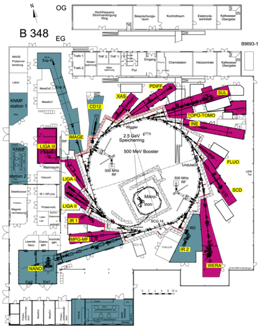


Figure: Floor plan of ANKA electron ring with 16 tangential *beamlines*

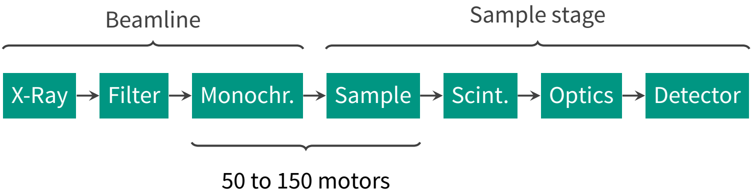
X-ray imaging experiments

Karlsruhe Institute of Technology

Beamline

X-Ray → Filter → Monochr.

X-ray imaging experiments

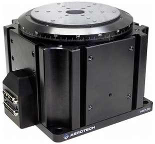


X-ray imaging experiments

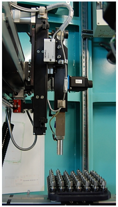
Beamline

Sample stage

X-Ray → Filter → Monochr. → Sample → Scint. → Optics → Detector



Rotary stage



Sample changer

X-ray imaging experiments

Karlsruhe Institute of Technology

Beamline

Sample stage

X-Ray → Filter → Monochr. → Sample → Scint. → Optics → Detector

Convert X-ray photons into visible light, magnify and detect them with CCD or CMOS.

X-ray imaging experiments

Beamline: X-Ray → Filter → Monochr.

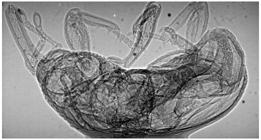
Sample stage: Sample → Scint. → Optics → Detector

X-ray imaging experiments

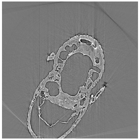
Beamline: X-Ray → Filter → Monochr.

Sample stage: Sample → Scint. → Optics → Detector

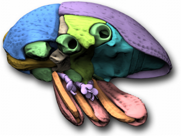
→ Process.



Projections



Reconstructed slice



Visualization

Challenges

Karlsruhe Institute of Technology

Fast, high volume acquisition

- Up to hundreds of samples per experiment
- Many thousand frames per second per sample
- Up to eight mega pixels at 16 bit per frame

Challenges

Karlsruhe Institute of Technology

Fast, high volume acquisition

- Up to hundreds of samples per experiment
- Many thousand frames per second per sample
- Up to eight mega pixels at 16 bit per frame

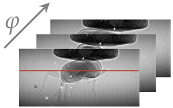
Efficient, fast data processing

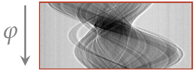
- On-site data processing for visual quality assurance
- Use reconstructed data for early feedback
- Use existing computing infrastructure

Tomographic reconstruction

Problem

From a series of projections …

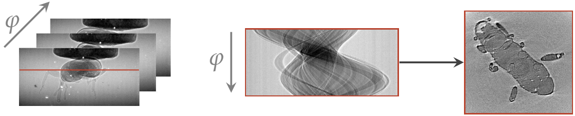




Tomographic reconstruction

Problem

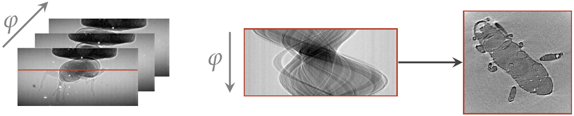
From a series of projections … reconstruct *unknown* slice information



Tomographic reconstruction

Problem

From a series of projections … reconstruct *unknown* slice information



Solutions

- Solve analytically using the Fourier-slice theorem (DFI)
- Filter projections and smear back into empty volume (FBP)
- Model detection as a linear system and solve algebraically (ART)

Improving reconstruction

There is no “one-size-fits-all” solution

- Quality: DFI < FBP < ART
- Speed: DFI > FBP > ART[1]

CPUs are still too slow

- Up to hours for very large data sets
- On-line and on-site inspection assessment inconvenient
- Fast feeback impossible

[1] $O(n \log n) \subset O(n^3) \subseteq O(cn^3)$

Why GPUs?

A superficial comparison

Processor	Purpose	Cores	Performance	Bandwidth	Price
CPU	General	12	0.5 TFLOPs	60 GB/s	> € 2200
GPU	Graphics	2880	5.0 TFLOPs	336 GB/s	€ 430

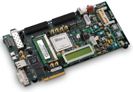
But …

- Work differently than a CPU
- Cover restricted problem domains

More challenges

Heterogeneous computing

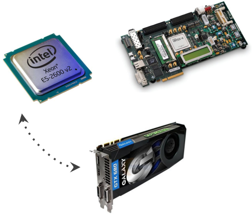
- Systems with different processors and topologies (CPUs, GPUs, FPGAs)



More challenges

Heterogeneous computing

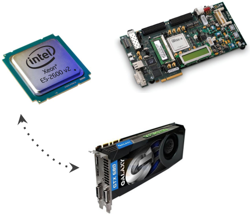
- Systems with different processors and topologies (CPUs, GPUs, FPGAs)
- Needs precise scheduling and resource allocation as well as architecture-specific code
- Avoid unnecessary copies between devices



More challenges

Heterogeneous computing

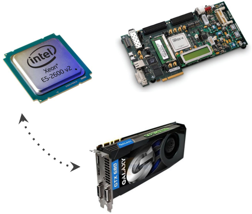
- Systems with different processors and topologies (CPUs, GPUs, FPGAs)
- Needs precise scheduling and resource allocation as well as architecture-specific code
- Avoid unnecessary copies between devices



More challenges

Heterogeneous computing

- Systems with different processors and topologies (CPUs, GPUs, FPGAs)
- Needs precise scheduling and resource allocation as well as architecture-specific code
- Avoid unnecessary copies between devices

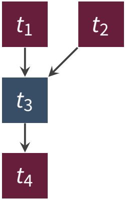


Data streaming

- Common in trigger systems
- Low latency and high throughput desirable
- Must consider clusters

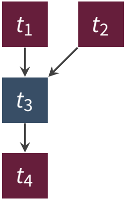
Basic approach

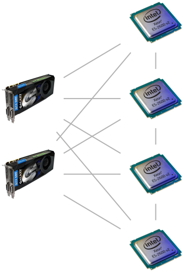
- Define algorithmic computation *and* data flow as a directed graph



Basic approach

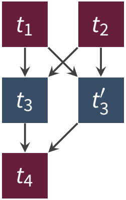
- Define algorithmic computation *and* data flow as a directed graph
- Determine local system of CPUs and GPUs

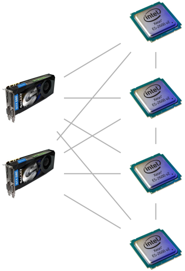




Basic approach

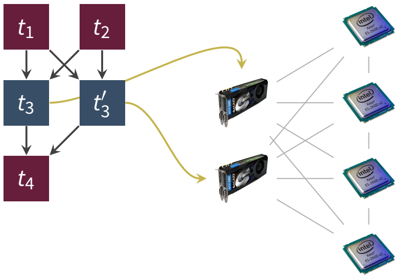
- Define algorithmic computation *and* data flow as a directed graph
- Determine local system of CPUs and GPUs
- Transform graph to accomodate for additional processors





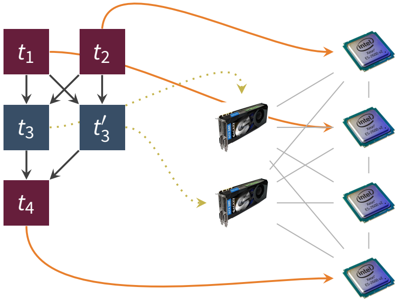
Basic approach

- Define algorithmic computation *and* data flow as a directed graph
- Determine local system of CPUs and GPUs
- Transform graph to accomodate for additional processors
- Assign tasks to GPUs



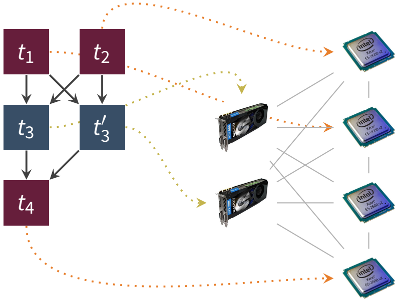
Basic approach

- Define algorithmic computation *and* data flow as a directed graph
- Determine local system of CPUs and GPUs
- Transform graph to accomodate for additional processors
- Assign tasks to GPUs and CPUs



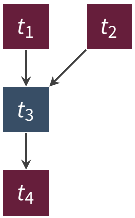
Basic approach

- Define algorithmic computation *and* data flow as a directed graph
- Determine local system of CPUs and GPUs
- Transform graph to accomodate for additional processors
- Assign tasks to GPUs and CPUs



Scaling to clusters

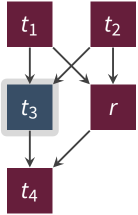
- Use algorithmic description



Local master

Scaling to clusters

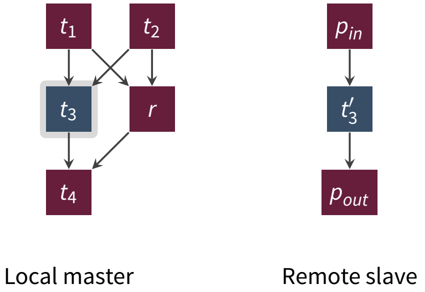
- Use algorithmic description
- Replicate sub-graph



Local master

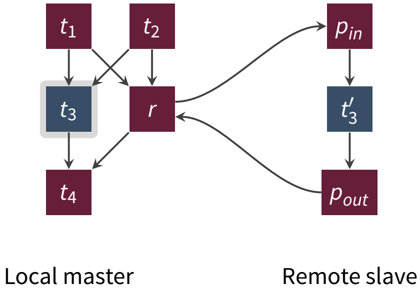
Scaling to clusters

- Use algorithmic description
- Replicate sub-graph
- Instatiate tasks on remote nodes



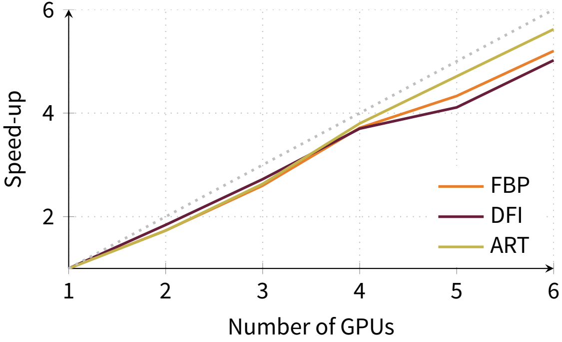
Scaling to clusters

- Use algorithmic description
- Replicate sub-graph
- Instatiate tasks on remote nodes
- Forward data and receive results



Improvements

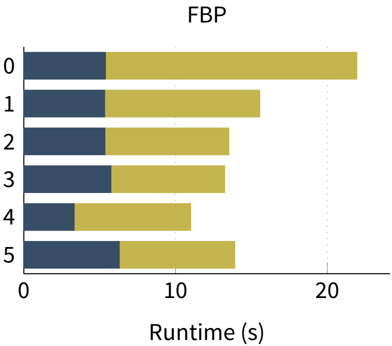
Single node multi-GPU



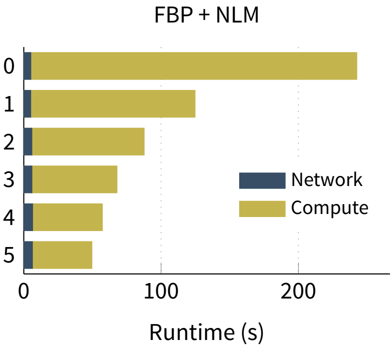
Good scalability with near linear speed-up for up to 6 NVIDIA GTX 580’s.

Multi node cluster

Scalability strongly depends on “computation / data transfer” ratio …



High network overhead



Almost linear speedup

Let’s do better

Improving user experience

Performance problems but ...

- Yet another compute language
- Hardware knowledge required for best performance
  - Work group layout,
  - Memory access patterns
  - Cache hierarchies etc. pp. ...

Improving user experience

Performance problems but …

- Yet another compute language
- Hardware knowledge required for best performance
  - Work group layout,
  - Memory access patterns
  - Cache hierarchies etc. pp. …

GPU programming is **hard!**

Improving user experience

Karlsruhe Institute of Technology

Performance problems but …

- Yet another compute language
- Hardware knowledge required for best performance
  - Work group layout,
  - Memory access patterns
  - Cache hierarchies etc. pp. …

GPU programming is **hard!**

Use Python instead

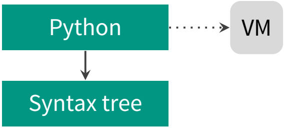
- Straightforward: Use NumPy's vectorized expressions
- Clever: Run the same code on GPU/CPU/FPGA

Compiling Python into GPU code

- Python interprets statements on an abstract virtual CPU

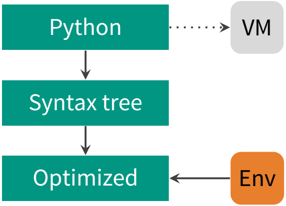
Compiling Python into GPU code

- Python interprets statements on an abstract virtual CPU
- Parse Python code into syntax tree



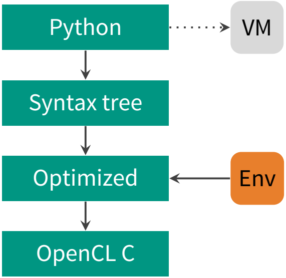
Compiling Python into GPU code

- Python interprets statements on an abstract virtual CPU
- Parse Python code into syntax tree
- Optimize tree according to environment



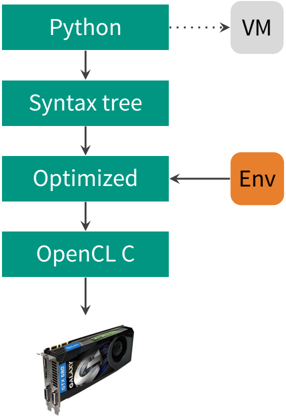
Compiling Python into GPU code

- Python interprets statements on an abstract virtual CPU
- Parse Python code into syntax tree
- Optimize tree according to environment
- Generate OpenCL C code



Compiling Python into GPU code

- Python interprets statements on an abstract virtual CPU
- Parse Python code into syntax tree
- Optimize tree according to environment
- Generate OpenCL C code
- Run generated code on GPU(s)



Benefits

- Non-invasive changes

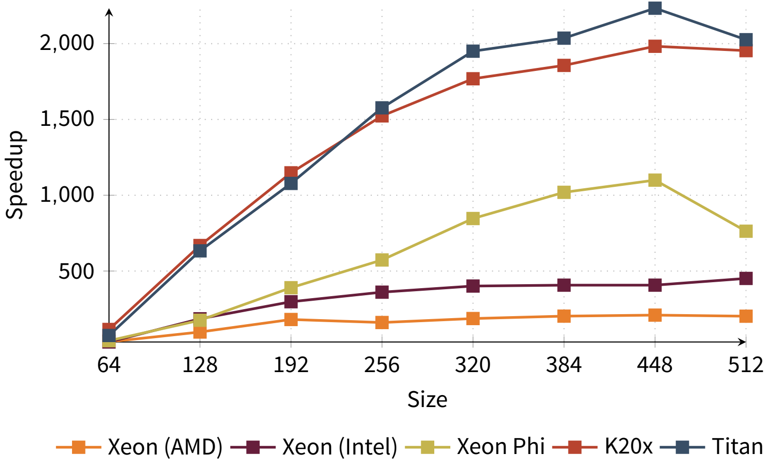
```python
def calc(x, y):
    return np.cos(x) + np.sin(y)
```

becomes

```python
@jit
def calc(x, y):
    return np.cos(x) + np.sin(y)
```

- Additional optimization opportunities

Speedup compared to NumPy



Conclusion

- Heterogeneous systems can be utilized for streamed data

Conclusion

- Heterogeneous systems can be utilized for streamed data
- Scaling out to clusters works well

Conclusion

- Heterogeneous systems can be utilized for streamed data
- Scaling out to clusters works well
- No one needs to write GPU kernels

Thank you. Any questions?