MAX IV – UCPH

Collaboration Workshop

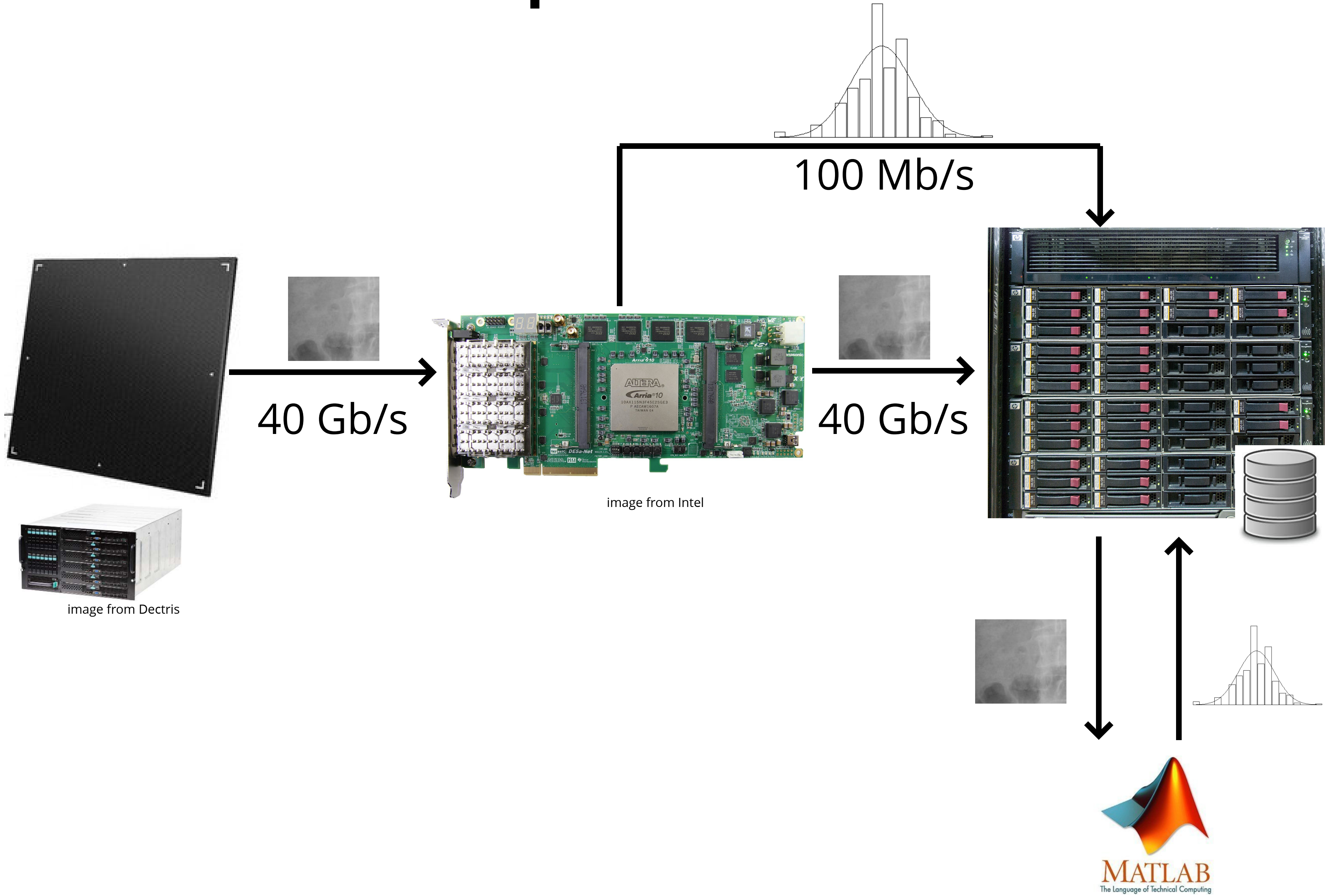
Kenneth Skovhede

MAX IV, 2019-12-09

SME / OpenCL

Status

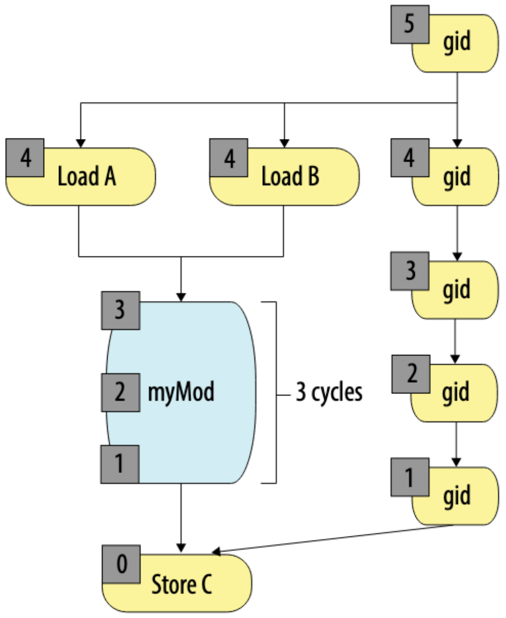
The setup

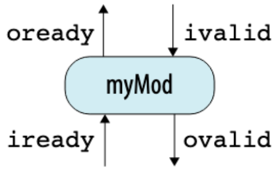

100 Mb/s
40 Gb/s
40 Gb/s
image from Intel
image from Dectris
MATLAB
The Language of Technical Computing

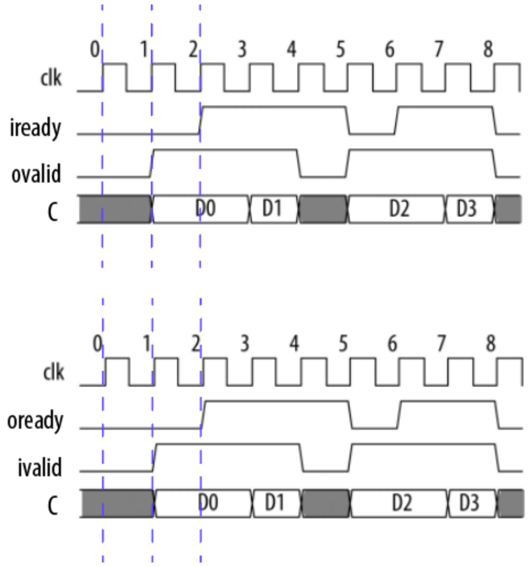
Overview

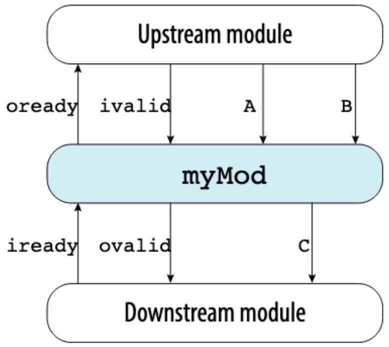
```
extern int myMod(int, int);
void kernel pe(global int* A,
               global int* B,
               global int* C){

    int gid = get_global_id(0);
    int a = A[gid];
    int b = B[gid];
    C[gid] = myMod(a, b);
}
```









images copyright Intel

Status so far

- Builds take +3 hours
- Essentially no debug information
- Appears unable to hold state

- New SDK 19.03, build time ~20min
- No binary for 19.x builds due to license issues
- Spent more than 2 months, incl. support from Intel, and still no working 19.x binary produced

Bohrium + SME

Status

Same setup

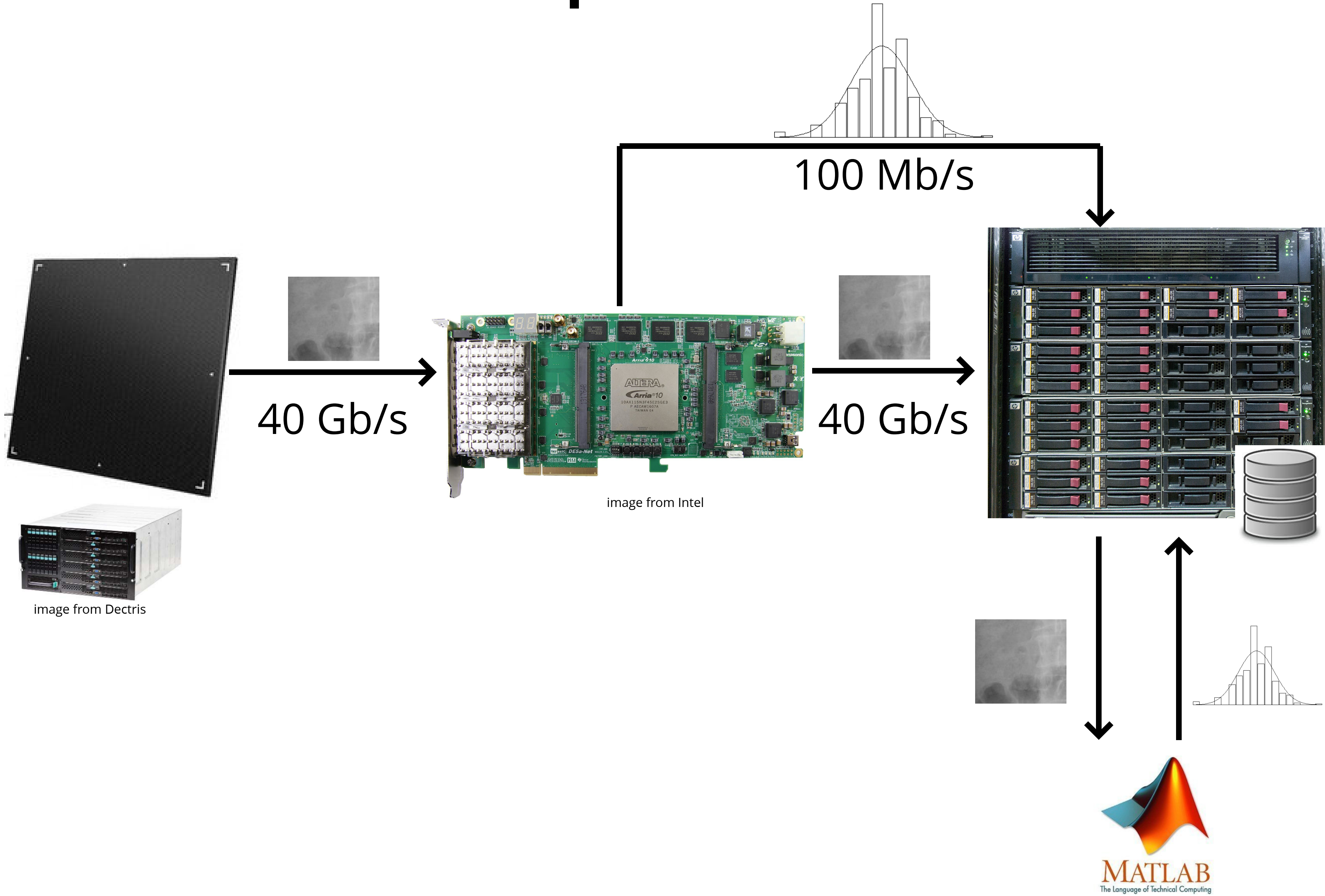


image from Dectris

image from Intel

Bohrium

Bohrium provides automatic acceleration of array operations in Python/NumPy, C, and C++ targeting multi-core CPUs and GPGPUs.

```
> python compute.py
```

$\downarrow$

```
> python -m bohrium compute.py
```

```python
import numpy as np

a = np.arange(10)
b = (a + 2) * a
c = np.histogram(b)
```

```
a = [10]
t0 = [10]
arange a, 10
add, t0, a, 2
mul b, t0, a
hist c, b
```

image from Intel

OpenMP

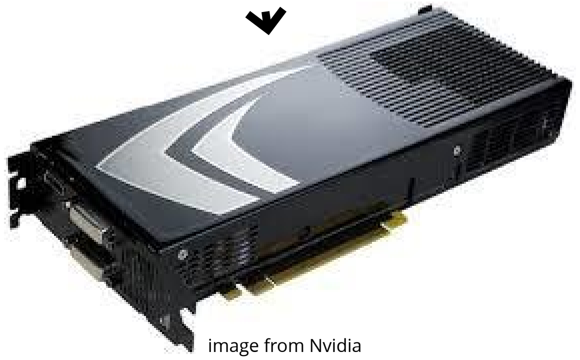


image from Nvidia

OpenCL CUDA

Goal

```
import numpy as np

def kernel(source, sink):
  b = (a + 2) * a
  sink = np.histogram(b)

a = np.arange(10)
c = np.empty(5)

fpga_execute(kernel, [a, c])
```

image from Xilinx

Current

```python
import numpy as np

a = np.arange(10)
b = (a + 2) * a

print b
```

$\longrightarrow$

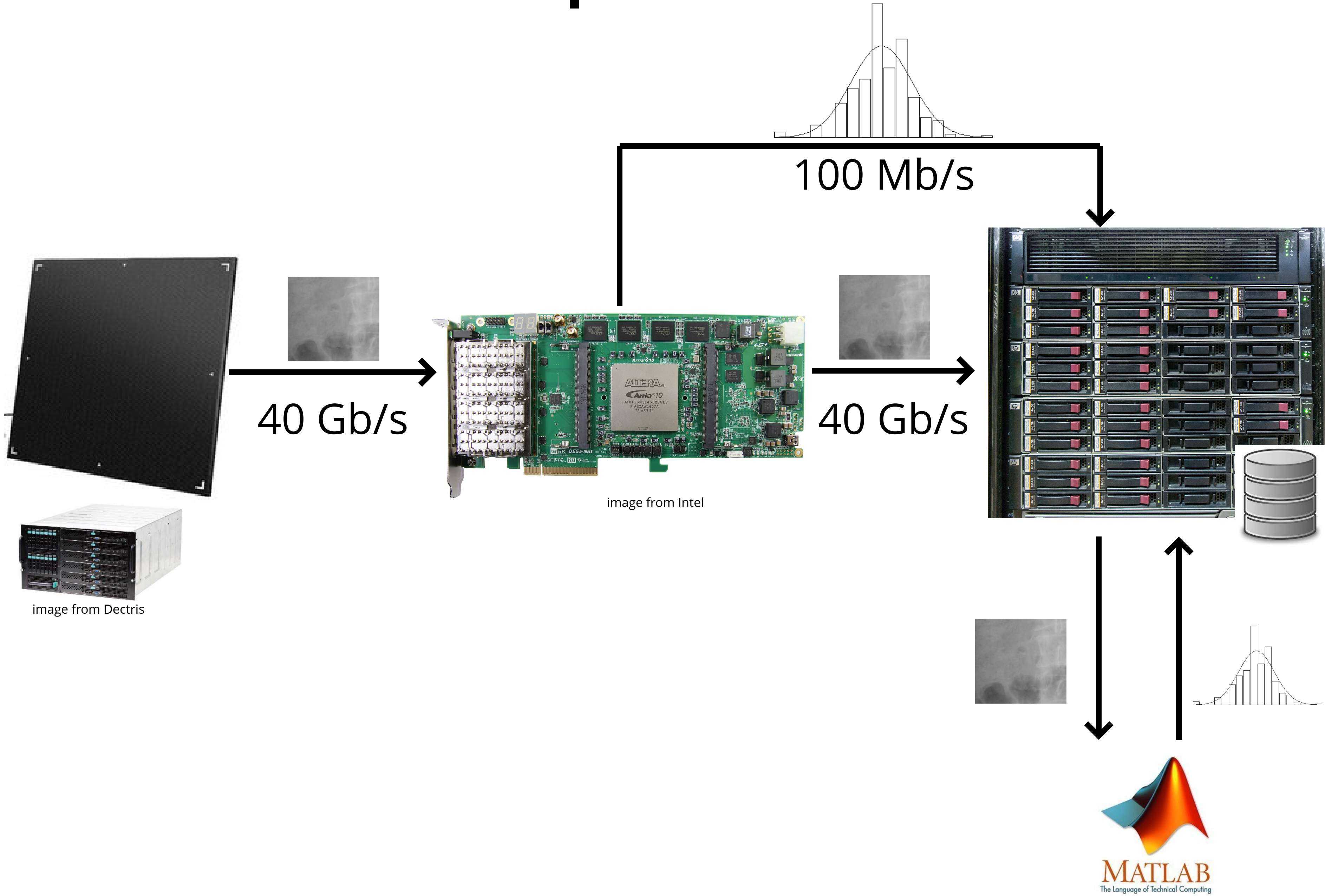
```
a = [10]
t0 = [10]
arange a, 10
add, t0, a, 2
mul b, t0, a
```

$\longrightarrow$

SME

FPGA & TCP

Same setup


100 Mb/s
40 Gb/s
40 Gb/s
image from Dectris
image from Intel
MATLAB
The Language of Technical Computing

TCP on FPGA

TCP is generally a swiss army knife:

- Handles packet loss
- Handles packet out-of-order
- Flow control/congestion
- Etc

Which we don't need on a controlled network...

Unfortunately, the detector vendors decide the protocol: TCP

TCP on FPGA

2x Msc Students did not manage to get a working TCP solution in 6 months

Open Source TCP implementations are board specific

A newer Open Source solution exists, but it fills most of the board just for handling buffers and concurrent connections

Proprietary ToE solutions exist, but are expensive and requires vendor lock-in

General advice is: don't do TCP in FPGA logic

TCP offload Engine (ToE)

Cards exist, but low-level documentation is lacking and card specific

Seems to favor the "Chimney" solution, due to security implications with a duplicated TCP stack

Future plans

DPDK

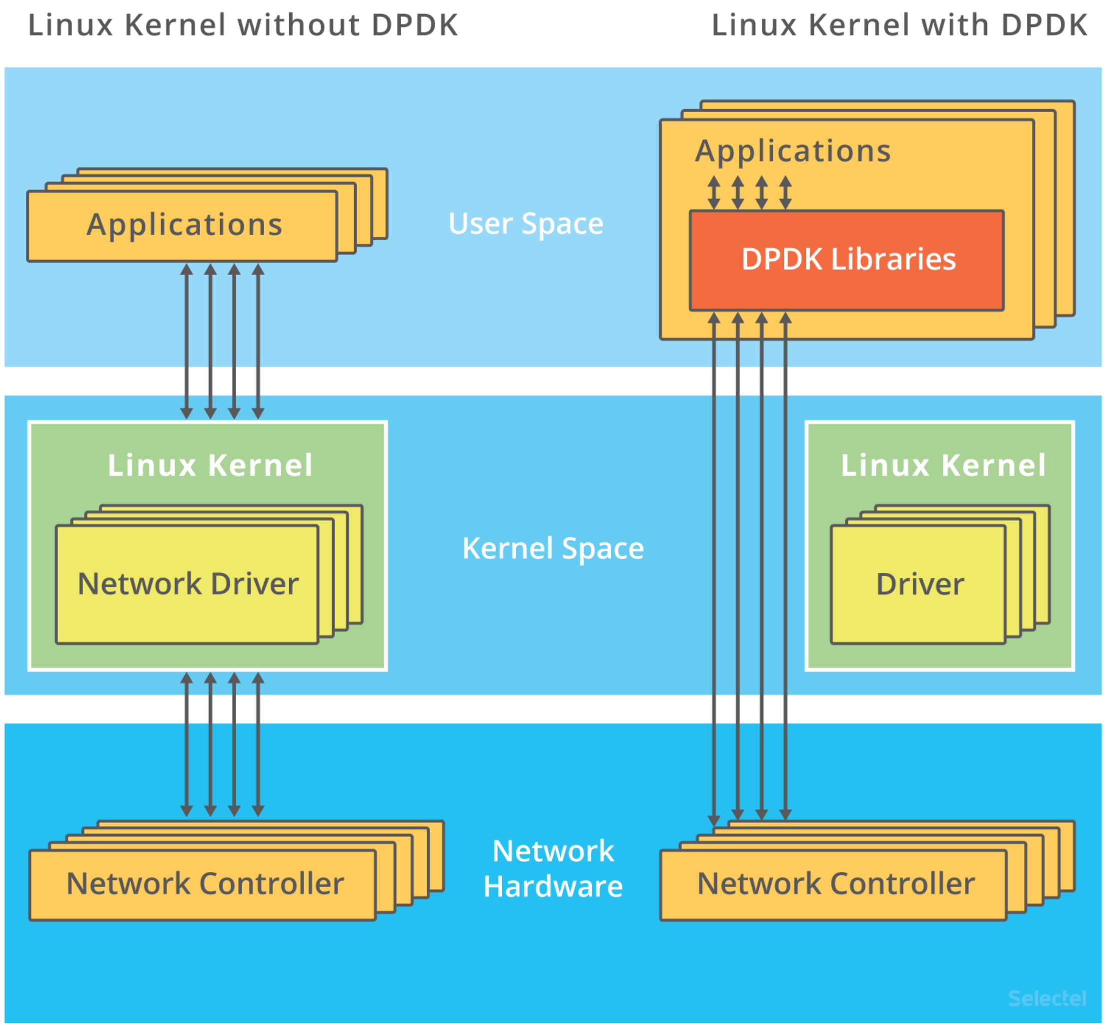


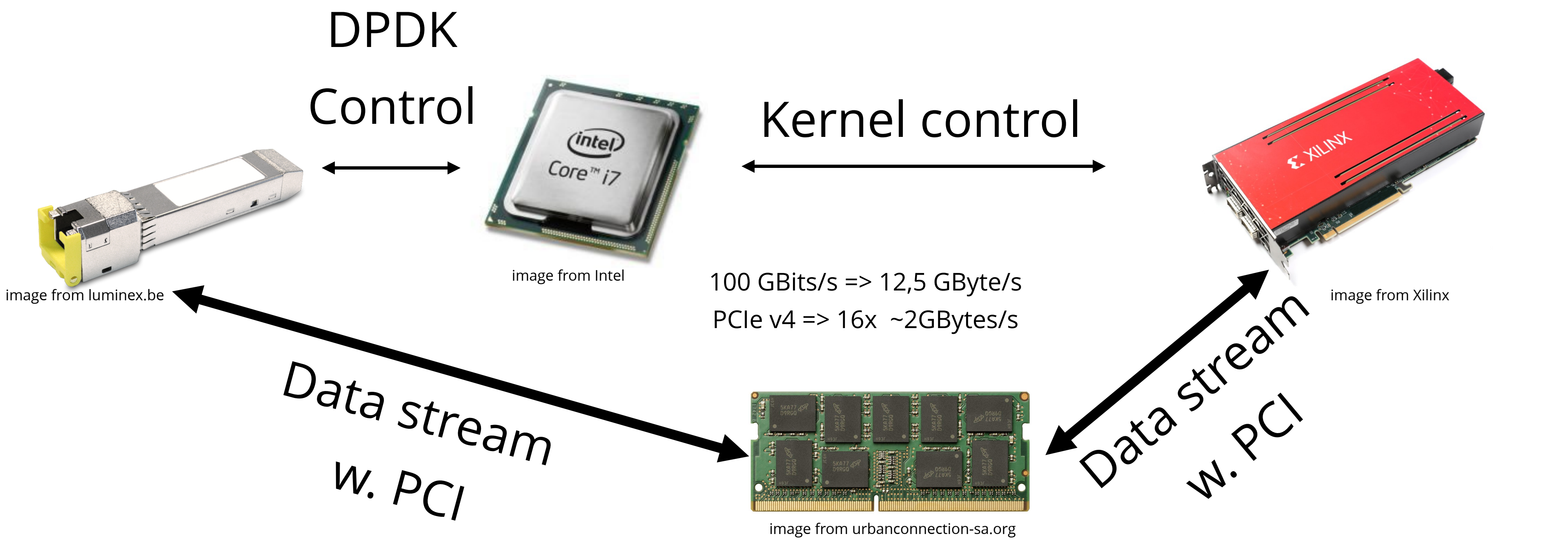
image from DPDK

Combining CPU + FPGA

DPDK
DATA PLANE DEVELOPMENT KIT

OpenCL

SYCL™



100 GBits/s => 12,5 GByte/s
PCIe v4 => 16x ~2GBytes/s