GPU and network innovations

John W. Romein

Motivation

- goals:
  - more powerful instruments ← higher bandwidth
  - improve energy efficiency
  - reduce implementation effort

Agenda

- GPU innovation
- network innovation

} integrated approach

Tensor cores

- hardware matrix-multiplication units
  - limited-precision input data
  - 5-10x faster than regular GPU cores
- accelerates deep learning
- since ~5 years

Use tensor cores for signal processing?

- yes, if:
  - algorithm translates to matrix-matrix multiplications
    - correlator: ✔
    - beam former: ✔, ✘
    - FIR filter: (likely) ✘
    - non-uniform Fourier transform: ✔
    - FFT: ✘ (radix 8: ✔)
  - operates on few bits ✔

The *Tensor-Core Correlator*[1]

- GPU library
  - performs (tensor-core) computations
  - not full application (no I/O etc.)
  - highly optimized
  - hides nasty details
- open source[2]
- rapidly adopted by radio telescopes worldwide

1) J.W. Romein, The Tensor-Core Correlator, A&A 656(A52), Dec 2021
2) https://git.astron.nl/RD/tensor-core-correlator

Implementation challenges

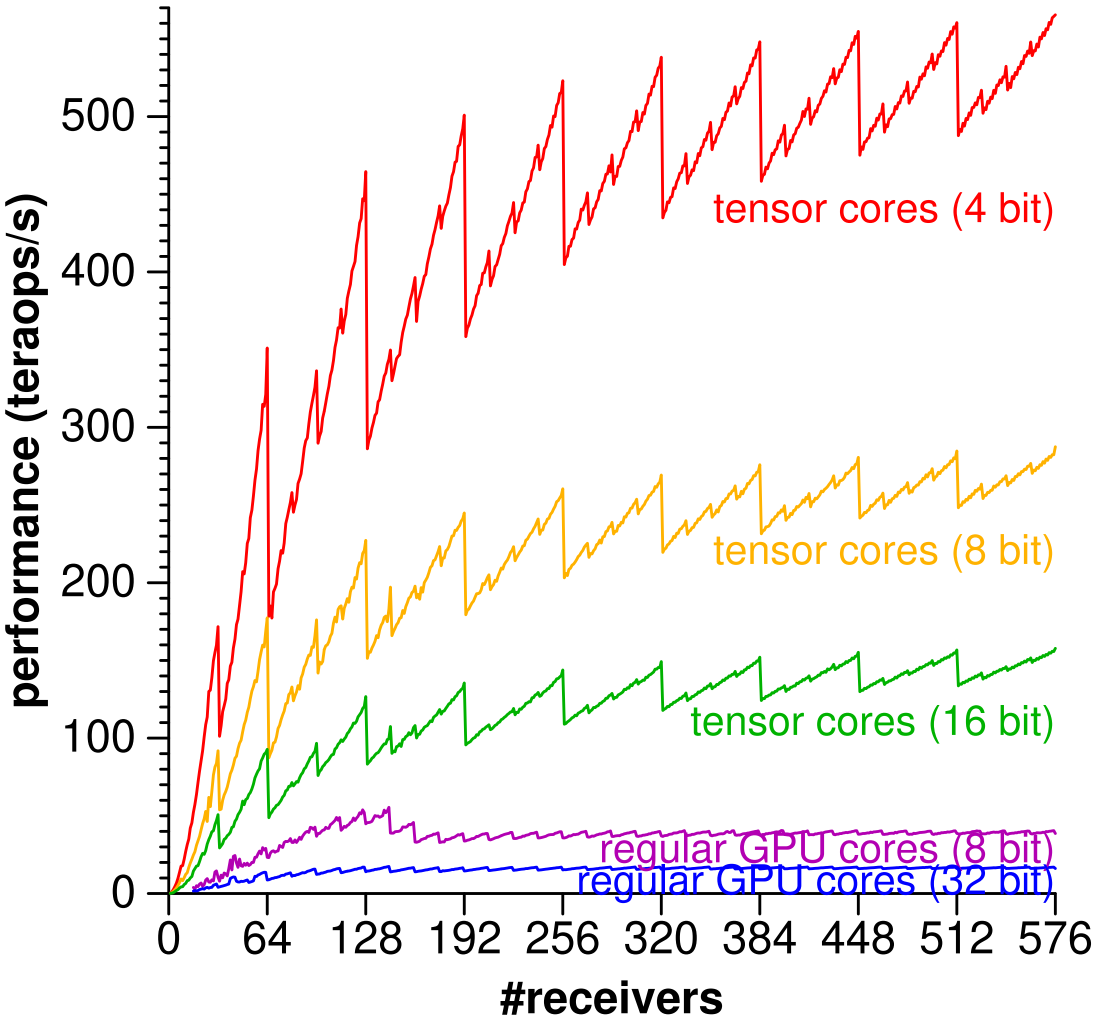
1) complex numbers
2) triangular output format

} not supported by tensor cores

3) fast data fetching
4) 4-bit mode requires ptx assembly hacking

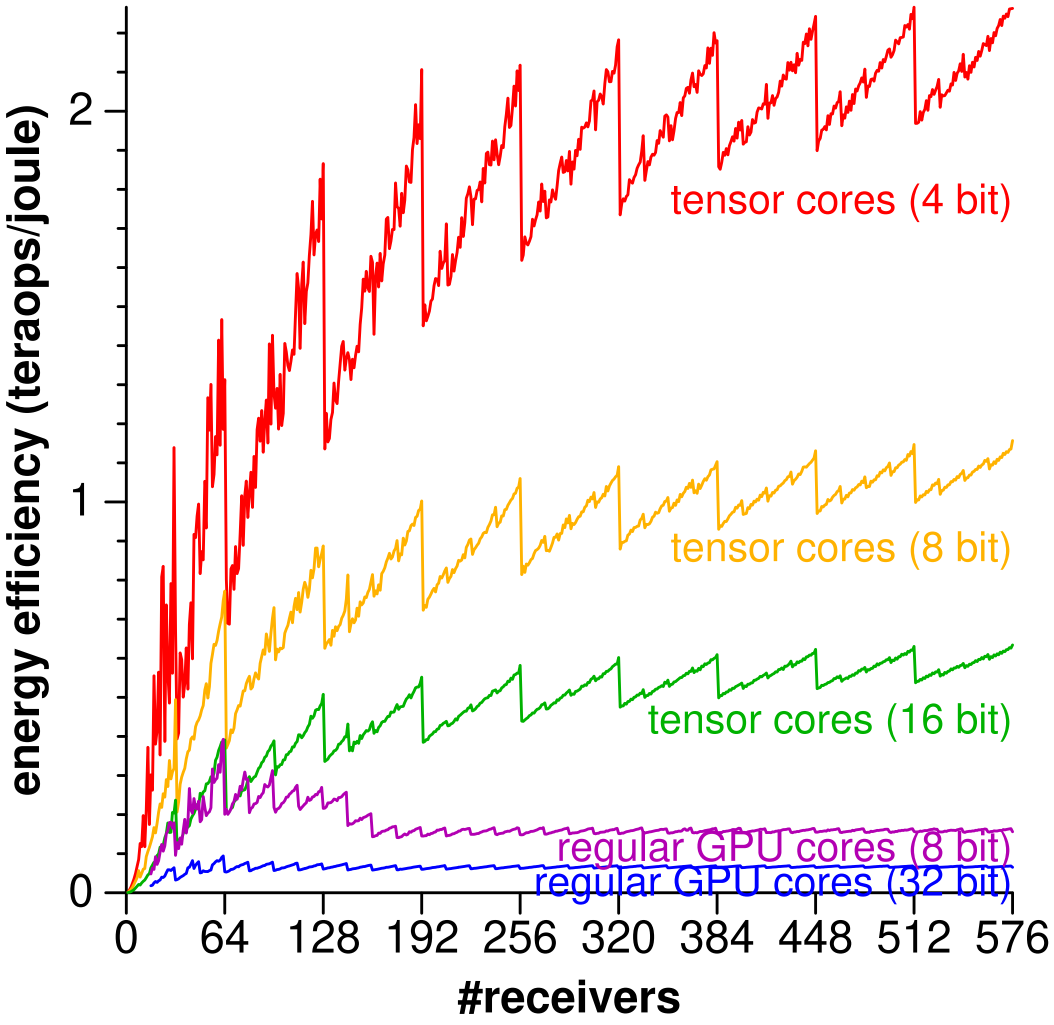
- all hidden from the user

Performance



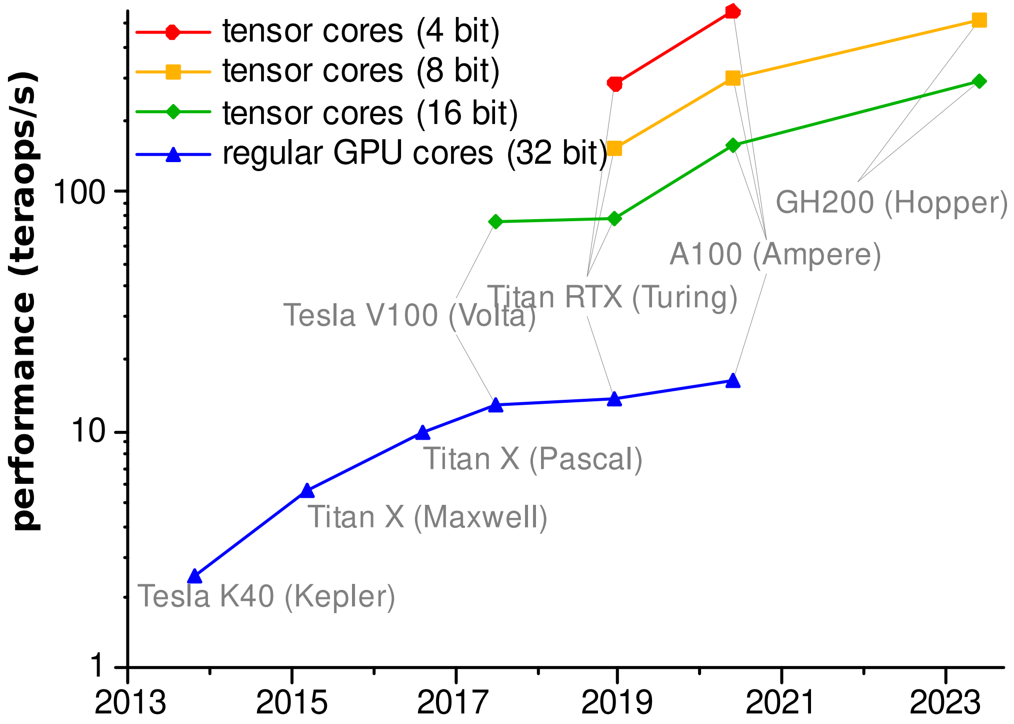
NVIDIA A100 PCIe 40 GB

Energy efficiency



NVIDIA A100 PCIe 40 GB

Tensor cores: answer to the demise of Moore's law

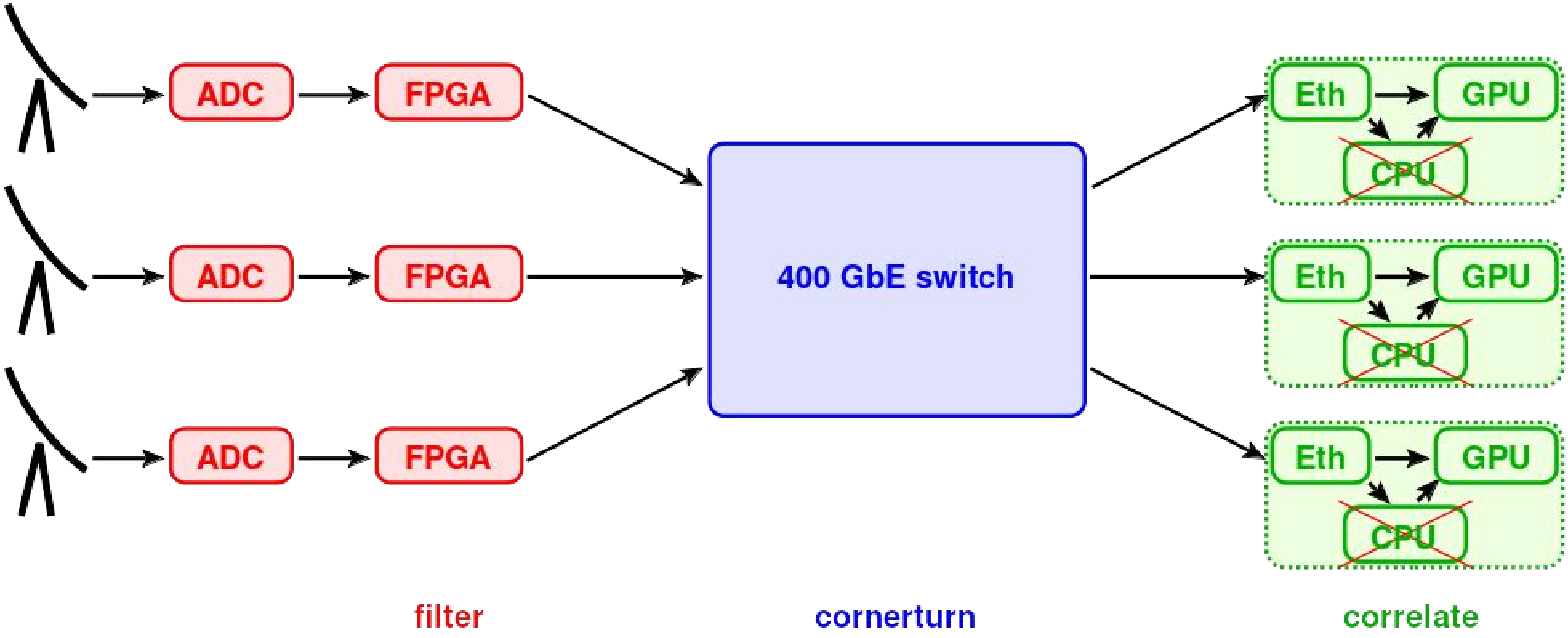


- historical best-case correlator performance

The I/O challenge

- GPU correlators in last decade: up to 100x performance
- I/O must scale proportionally
- receiving >40 Gb/s data: prohibitive OS overhead
  - ~~wait for faster processors~~
  - need new I/O techniques

Data Transport



- digitizer → corner turn → correlator
- stream data: FPGA → switch → GPU

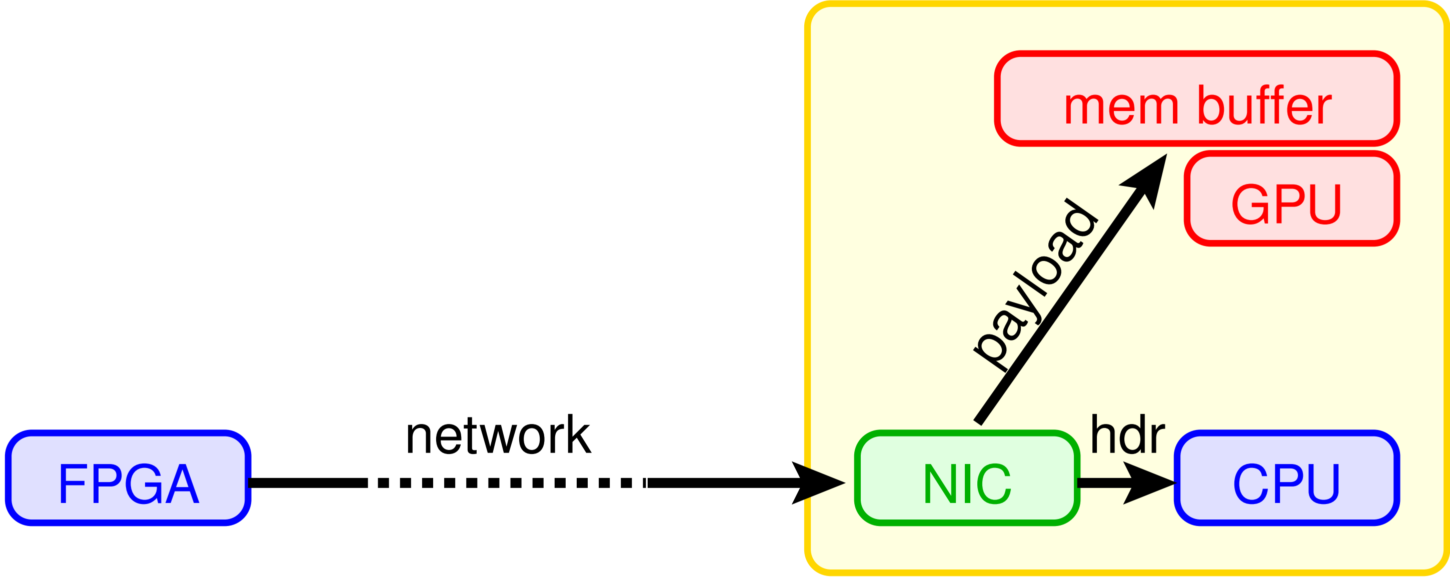
Data Transport

- explore new methods
  - Remote Direct Memory Access (RoCE v2)
  - Data Plane Development Kit

  } different advantages & disadvantages

- implementing demo correlator

Data Plane Development Kit

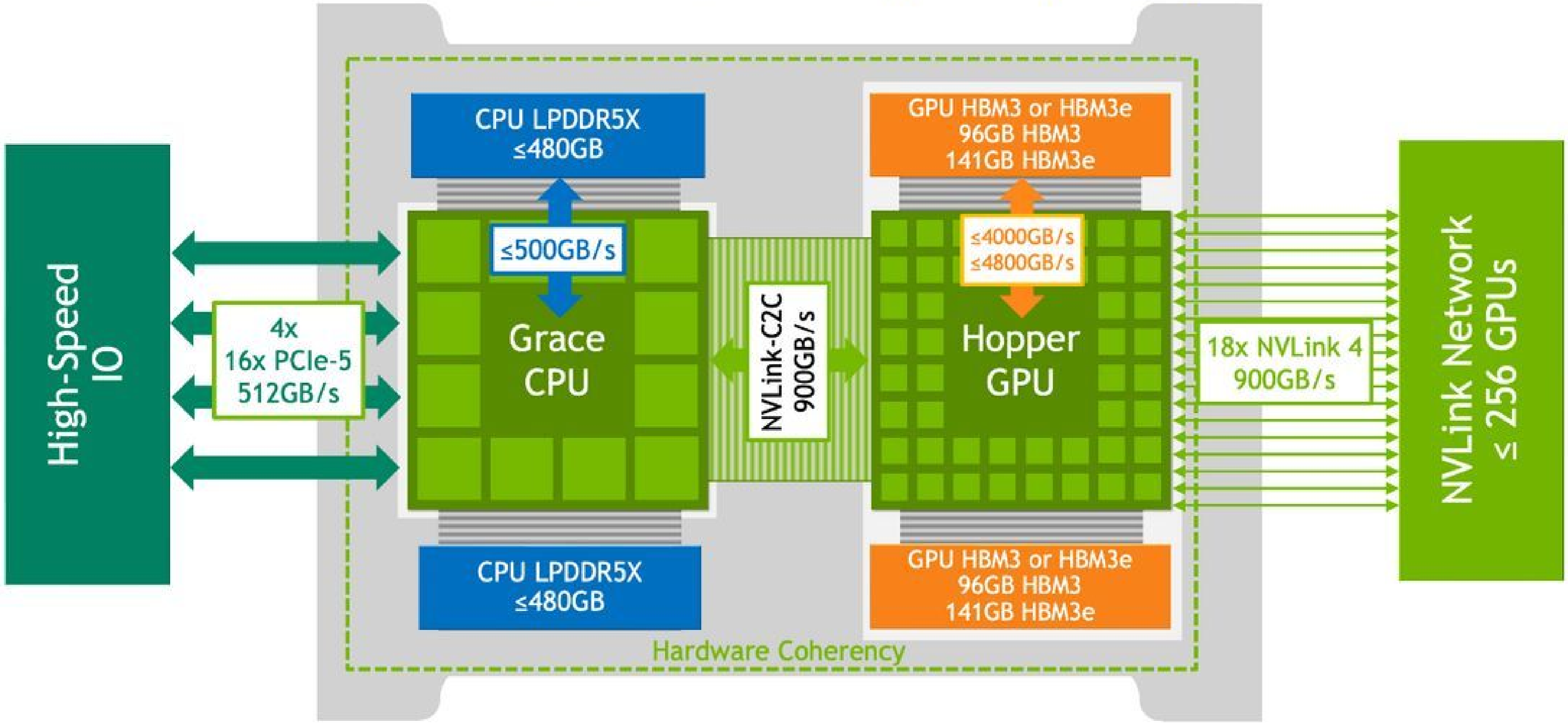


- ~~OS~~ application controls NIC → no OS overhead
- application sends/receives Ethernet packets
- new: GPU support → GPU handles packets

DPDK demo correlator

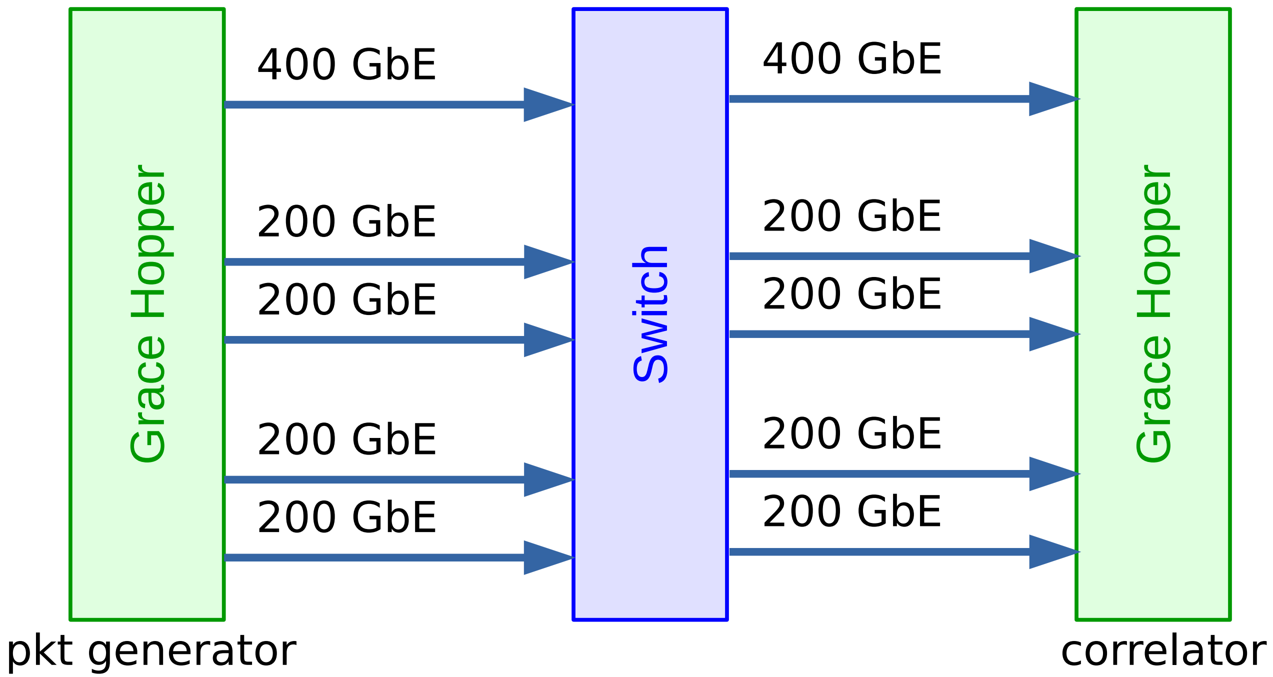
- correlator *partially* implemented
  - packet receipt ✔
  - delay input streams ✘
  - GPU filtering ✘
  - GPU correlator ✔
  - writing correlations ✔

NVIDIA GH200 Grace Hopper Superchip



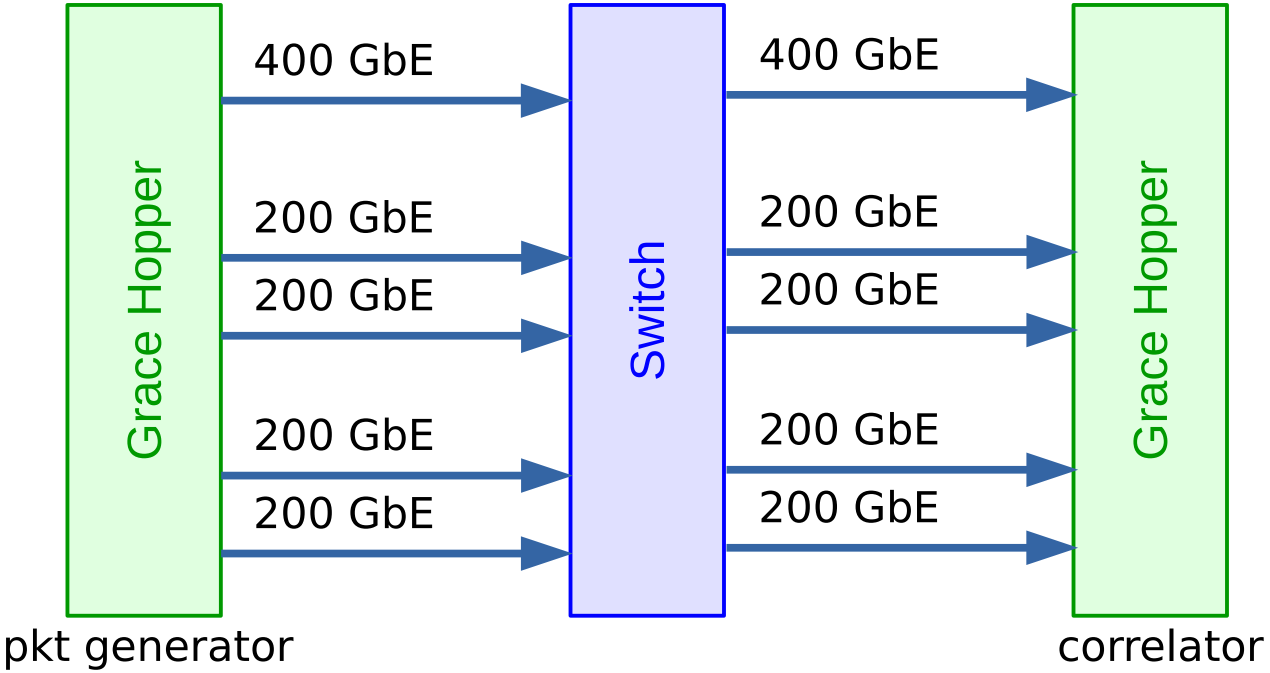
- most powerful GPU ever
- 14x more CPU ↔ GPU bandwidth than PCIe gen4

Grace Hopper systems setup



- all PCIe slots filled with NICs
  - 1200 Gb/s total

Grace Hopper DPDK results



- result: *1196 Gb/s received in GPU memory*
  - DPDK (software innovation): 40 → 199 Gb/s (PCIe gen4 NIC/GPU)
  - Grace Hopper (hardware innovation): 199 → 1196 Gb/s

To do

- finish the demo application
  - buffer input data to apply delays
  - properly channelize data

Motivation revisited

- goals:
  - more powerful instruments ← higher bandwidth *promising results*
  - improve energy efficiency *tensor cores: ✔*
    DPDK: okay, but RDMA may be better
  - reduce implementation effort *complex, but hidden inside Radio Block*

Summary

- ~10 years:
  - innovations like tensor cores → up to 100x performance
  - compute bound → I/O bound
- integrated GPU / network approach looks very promising