Ben Barsdell (Harvard)
Mike Clark (NVIDIA)
Lincoln Greenhill (Harvard)
Outline

- Modern radio astronomy meets HPC
- Early lessons learned
- A scalable pipeline
- Implementation concepts
- How to go even faster
Modern Radio Astronomy
Why build a 10,000-input array?
Why build a 10,000-input array?
Why build a 10,000-input array?

- Microwave astronomy
- Low-freq radio astronomy
- Optical astronomy

**Timeframes:**
- $3 \times 10^5$ yr
- $5 \times 10^8$ yr
- $10^9$ yr
- $10^{10}$ yr

Ben Barsdell 03/27/2014
Cross-Correlation

- Interferometry
- Compute-intensive streaming application
- E.g., 512-inputs @ 60 MHz => 250 Gb/s
Modern Correlators

- **FX design**: channelise (F), cross-correlate (X)
- **Hardware**: ASICs, FPGAs, CPUs, GPUs

Input $\rightarrow$ ADC $\rightarrow$ PFB $\rightarrow$ Corner turn $\rightarrow$ Cross-correlate $\downarrow$ Cal & image
Modern Correlators

- FX design: channelise (F), cross-correlate (X)
- Hardware: ASICs, FPGAs, CPUs, GPUs

E.g., VLA, ALMA

Input → ADC → PFB → Corner turn → Cross-correlate → Cal & image

ASIC

CPU/GPU
Modern Correlators

- **FX design**: channelise (F), cross-correlate (X)
- **Hardware**: ASICs, FPGAs, CPUs, GPUs

E.g., SMA
Modern Correlators

**FX design**: channelise (F), cross-correlate (X)

**Hardware**: ASICs, FPGAs, CPUs, GPUs

E.g., LEDA, MWA, PAPER

![Diagram of FX design and hardware components]

- **Input** → **ADC** → **PFB** → **Corner turn** → **Cross-correlate** → **Cal & image**
- **FPGA**: 10/40 GbE switch
- **GPU**: CPU/GPU
Boarding the HPC Train

- Software provides scalability, portability and **flexibility**
- Potentially huge reductions in development time
- But poses **challenges** for the astro community...

**Diagram:**
- Input → ADC → PFB → Corner turn → Cross-correlate → Cal & image
1st PeXC prototype: lessons learned

- Hybrid CPU-GPU pipeline on Titan
- Sustained multi-PFLOP/s, but ran into problems
1st PeXC prototype: lessons learned

- Hybrid CPU-GPU pipeline on Titan
- Sustained multi-PFLOP/s, but ran into problems
  - Poor CPU F-engine perf. (struggled with 0.4% of the work!)
  - MPI did not like our pipeline infrastructure
  - MPI_Alltoall (OpenMPI, bipartite) scaling stalled at 2k nodes
New PeXC design

- GPU F-engine
- Thread-based pipeline infrastructure
- Unified correlator architecture
  - Uniform nodes; each does both F and X stages
  - Corner turn becomes simple MPI_Alltoall(v)
  - Flexible BW:N^2 ratio
  - Strong-scale to achieve speed (BW)∝ Nnodes
Implementation plans

- Lean heavily on existing libraries:
  - cuFFT, xGPU, MPI_Alltoall

- Only a few custom parts needed:
  - Various re-ordering/packing kernels
  - Flexible ring buffer implementation
Pipeline Model

- **Implicit overlap** of parallel resources between tasks
  - E.g., CPU, DMA engines, concurrent kernels, network
- Each task in own CPU thread
  - Don’t block the GPU
  - Use own stream + async API + cudaStreamSync...
Pipeline Model

- **Implicit overlap** of parallel resources between tasks
  - E.g., CPU, DMA engines, concurrent kernels, network
- Each task in own CPU thread
  - Don’t block the GPU
  - Use own stream + async API + cudaStreamSync...
- **Ring Buffer**
  - Single-publisher, multiple (dynamic) subscribers
    - Enables flexible branching pipelines
  - Custom allocators for system/pinned/device mem
  - Built-in support for ‘overlapping’ buffers
    - Easily implement convolution-like tasks
F-Engine

- Copy digitised data host → device
  - 8/12/16 bits
- Polyphase filterbank
  - FIR filter
  - 1D R2C FFT
Corner Turn

- MPI\_Alltoallv with uniform size distribution
- In-memory re-order before and after network transmission to maximise message sizes
- GPU-aware MPI makes things very easy
- Scalability of MPI\_Alltoall is critical to application performance
X-engine

- Lots of CMACs!
- **xGPU** library [1]
  - Highly-tuned CUDA library for this algorithm
- Uses *hierarchical memory tiling* approach to minimise memory overhead
- Several other optimisations
  - (see GTC’13 talk for details)

\[ C_{vij} = \sum_{t} a_{tvi} a_{tvj} \]

[1] https://github.com/GPU-correlators/xGPU
How to go even faster?

- PCI-E v3
- GPU Direct RDMA
- Maxwell!
- NVLink?
- Decreasing required memory bandwidth
  - Real-time compression?
  - Kernel fusion
  - 8- and 4-bit loads and stores instead of 32
- 4-bit compute?
4-bit FFMA trick (experimental)

- xGPU uses 32-bit FMA: highest throughput
- But we only need 4-bit multiply-add

**Float hacks?**
- Pack two 4-bit values into a 32-bit float
- 24-bit mantissa operates in \(|\|) on both 4-bit values
- Can accumulate 18 times before overflow

- Unfortunately, advantage killed by overhead
- What about doubles?
4-bit DFMA trick (experimental)

- $\text{DFMA} \rightarrow (A_1 \cdot A_2, A_1 \cdot B_2 + B_1 \cdot A_2, B_1 \cdot B_2)$

- This is (most of) a CMAC
  - Implement as preproc, dgemm, postproc

- 4-for-1, but with reduced DP throughput

- On Kepler, 4 * 1/3 $\Rightarrow$ 33% speed-up
  - Also, power consumption decreases by ~10%
Conclusions

- GPUs now powering some of the world’s largest radio telescope arrays
- HPC is a highly competitive solution for radio astronomy and related signal-processing apps
- A purely software approach has many advantages over firmware/hardware
- Many upcoming SW and HW advances, which can be integrated trivially