Particle track reconstruction on heterogeneous platforms with SYCL

Bartosz Sobol, Jagiellonian University
Grzegorz Korcyl
Outline

- Software development challenges in experimental physics
- Potential for SYCL
- Project overview
- Performance evaluation
- Conclusions and potential future works
Software challenges in experimental physics

● Long-living software
  ○ Experiment lifespan >10 years
  ○ In rapidly changing hardware platform landscape
● Low manpower and high rotation
● Often developed by non professionals
  ○ Developing parallel/GPU code can be hard
● Often use obsolete APIs/libraries or in-house solutions
Potential for SYCL

- Supports wide range of platforms
  - Vendor agnostic GPU support is crucial
  - Always possible to fall-back to CPU when necessary
  - Easy to test CPU vs GPU performance
- Easy basics (range, buffer, lambda, submit)
- Standardized
- Living ecosystem
  - Evolving standard, growing community, open-source implementations
Project background

- **Best case scenario:**
  All data stored for *offline* analysis

- **Data-rates increase**

- **Additional online (live) processing step required**
Project overview

- PANDA - particle physics experiment under construction at FAIR facility, Darmstadt
- Track reconstruction algorithm for one of PANDA detectors - Forward Tracker
  - Input: list of particle interactions with detector - hits
  - Estimating lines - free particle, and circles - in EM field from hits
  - Matching linear parts with circular
- Goals:
  - Determine on what platform it performs best
  - Use and learn SYCL
  - Develop guides for porting existing single-threaded code
SYCL implementation strategy

- Start with plain C++ single-threaded code
- Port to SYCL with minimal possible effort
- Introduce optimisations
  - Mainly data-flow and memory layout
  - Try to keep kernel code similar to initial version
  - Try to stay within simpler SYCL interfaces (buffers, ranges)
- Result: 7 Kernels + helper functions, ~1.5k lines of accelerated code
- Single code for different platforms
Performance evaluation

- Modern hardware from all leading vendors
  - Rome, Milan, Cascade Lake
  - V100, A100, MI250
  - Alveo U280
- Two major implementations:
  - hipSYCL (0.9.4)
  - DPC++ (2023.0, 2023.1 - MI250)
    - triSYCL/sycl - U280
- Compared with native CUDA implementation on NVIDIA GPUs
CPU performance

hipSYCL, omp backend

- EPYC 7742
- EPYC 7763
- Xeon Platinum 8268
GPU performance
Performance summary

- **EPYC 7742**: 100K events/s
- **EPYC 7763**: 120K events/s
- **Xeon Platinum 8268**: 80K events/s
- **V100**: 150K events/s
- **A100**: 200K events/s
- **MI250**: 250K events/s

*Legend:*
- CPU hipSYCL threads = cores
- CPU hipSYCL 16 threads
- GPU DPC++
Performance summary

- Alveo U280 performance ~2 orders of magnitude worse
  - We didn’t introduce any FPGA-specific optimisations
  - Adventure making the code compile and run
    - Still has some issues - potentially compiler bugs
  - Great to have SYCL available on such platforms
    - Certain algorithms can highly benefit
    - We hope development on the toolchain will continue
- Intel FPGAs not tested (yet) - tools probably more mature
Performance summary

- Algorithm itself isn’t ideal for GPU
  - Lot’s of branches and not parallelizable short loops
  - More data-bounded than compute-heavy
- CPU parallelization is quite good
  - Up to ~16 threads; TODO: compare with OpenMP
- GPU performance is mediocre compared to CPU
  - Probably wouldn’t improve significantly even with further fine-tuning
  - GPU optimisations also positively affect CPU performance
  - With SYCL, we have an efficient CPU version for free
Future PANDA computing pipeline

- In final PANDA significant part of data processing will be conducted online (live)
  - On HPC computing nodes with multicore CPUs and GPUs (APUs, FPGAs, ?)
- Up to 300 GB/s of raw data in final system
  - From different subsystems
    - Processed with diverse algorithms - some suitable for GPU acceleration, some not
- Software stack will have to be (re)designed
- Existing algorithms optimized/parallelized and reimplemented
- Development slowed-down by war in Ukraine
Conclusions

- We believe SYCL is a promising option for the use case
  - For prototyping, evaluating performance over platforms AND production use
  - Can provide satisfying and competitive performance with portability
- Presented work is a case study
  - For individual algorithm implementation and porting
  - We propose our methods and what we learned for final general solution
Particle track reconstruction on heterogeneous platforms with SYCL

IWOCL & SYCLcon, 20.04.2023

Bartosz Soból
bartosz.sobol@doctoral.uj.edu.pl