Tampere University

15+ Years of PoCL

Past Lessons, Present Reality, Future Directions

IWOCL 2026 | Heilbronn, Germany | May 6 2026

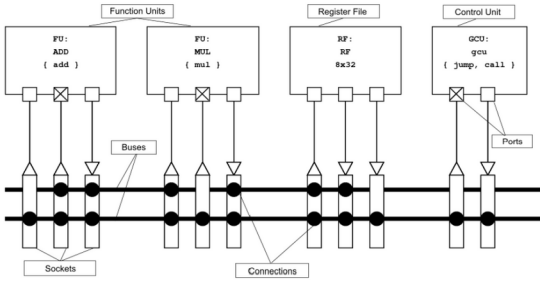
Pekka Jääskeläinen

Customized Parallel Computing group, Tampere University, Finland

2008-2009: The first steps towards PoCL

- Origins in a GPU (for graphics) research project in around 2008-2010: **GLSL** as input to a static instruction-level parallel optimized processing element (PE)
  - Doctoral research collaboration with Carlos S. de La Lama
  - PE: Transport Triggered Architecture (TTA, basically exposed datapath VLIW)
- Switched to OpenCL immediately after the first spec release in 2009 and shifted my own doctoral research focus towards GPGPU
  - OpenASIP / TCE

"Programmable and Scalable Architecture for Graphics Processing Units" in SAMOS 2009.

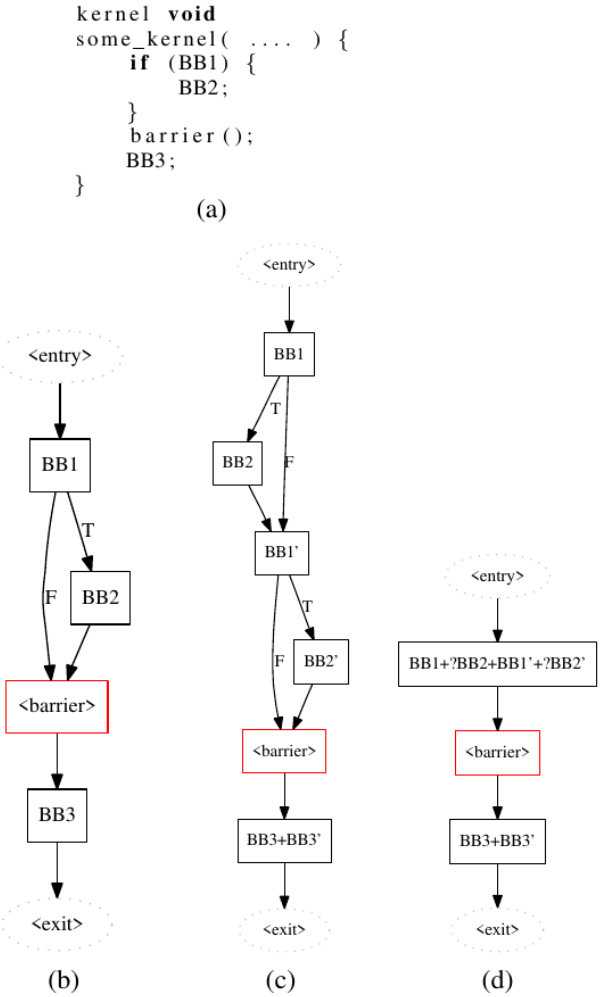


"OpenCL-based design methodology for application-specific processors" in SAMOS 2010.

```
#ifdef cl_TCE_ADDSUB
 clADDSUBTCE(a, b, c, d);
#else
 c = a + b;
 d = a - b;
#endif
```

PoCL kernel compiler: Original goals

- Compile multi-work-item (WI) work-groups (WG) to extract a lot of **instruction level parallelism** when unrolled or pipelined
  - The original compiler ”replicated” WIs (c) to provide free operations to hide static latencies in the VLIW pipeline
  - Aggressive predication for ”global scheduling” (d)
- **Data level parallelism** / SIMD was *not* of primary interest at that point



2010-2011: Open sourcing

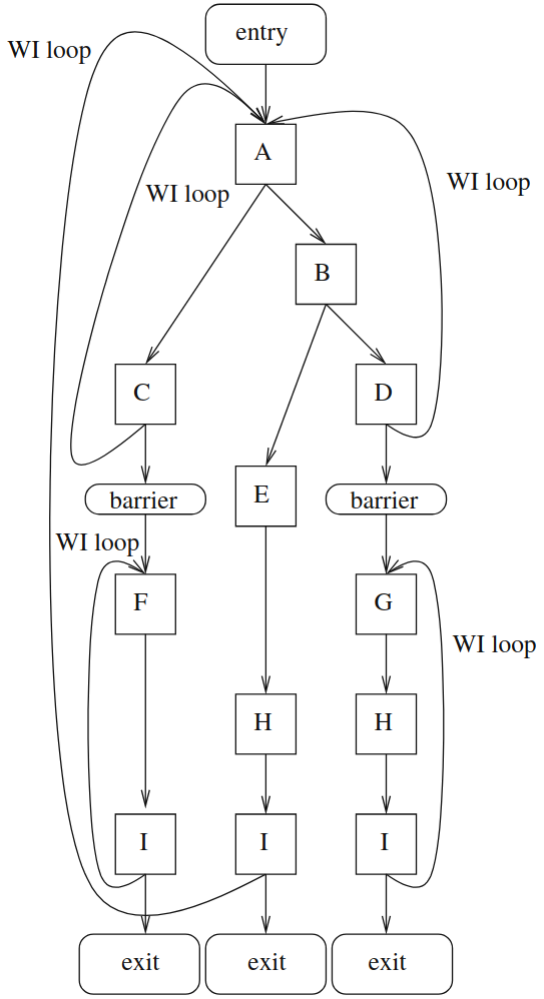
- Compiler work on LLVM IR generally useful to any ILP and also CPU-SIMD target?
  - Replicated WIs could be already vectorized with basic block vectorizers
- Open source project goal: a modular OpenCL kernel compilation flow for **any** LLVM-supported CPU/SIMD architecture
  - ...along with a reusable OpenCL runtime/builtin framework
- Originally called **Portable OpenCL (POCL)**, renamed in 2013 to **Portable Computing Language (PoCL)** to avoid ™ disputes and to adhere to Khronos conformance rules (not conformant yet!)

```
commit f86c535f9c995274eeaa56f05fba5cce8d8fad18
Author: Carlos Sánchez de La Lama <csanchezdll@gmail.com>
Date:   Fri Oct 14 13:31:27 2011 +0300

    Names changed, prepared to open source release.
```

2012-2014: Vectorizable work-item loops

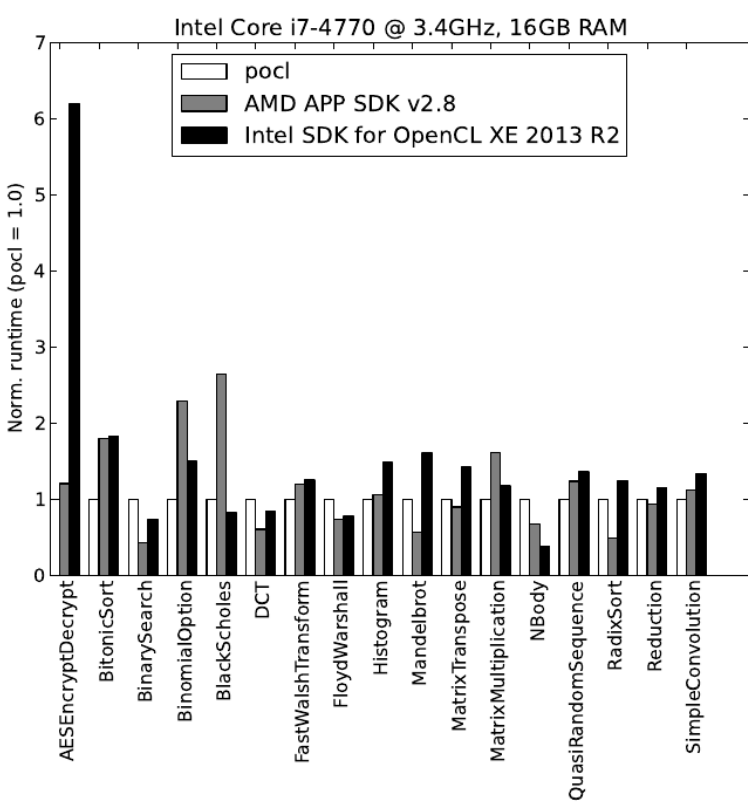
- To exploit scalable DLP of various SIMD instruction sets it would be better to utilize loop vectorizers
  - Generate parallel work-item loops instead of directly replicating WIs
  - Various transformations to make such static looping feasible
  - Express loop parallelism using a new LLVM metadata to remove the need for loop dependency analysis for the WI loops

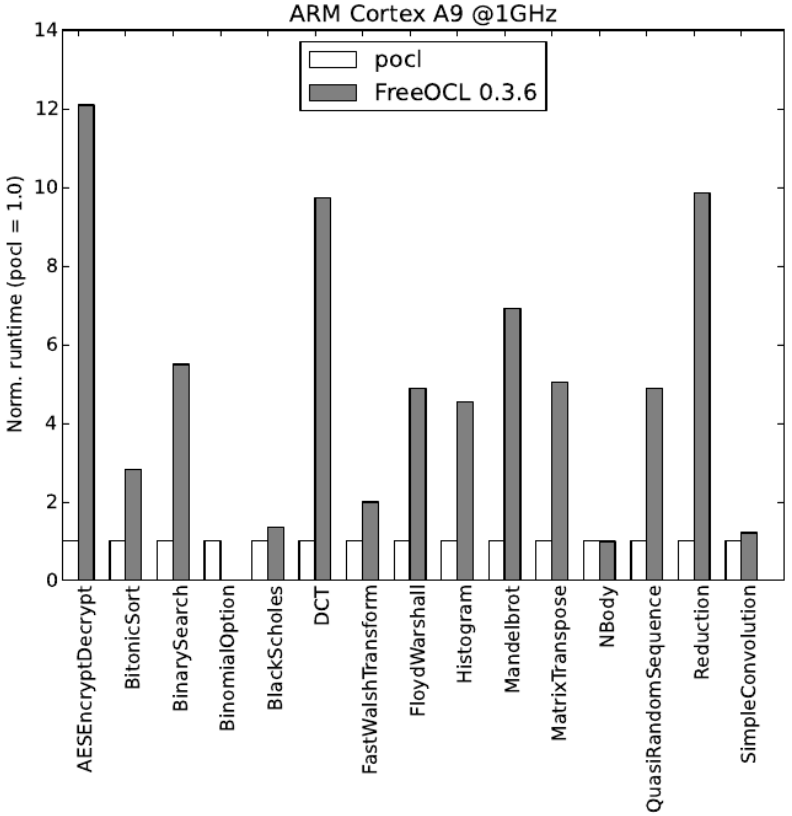


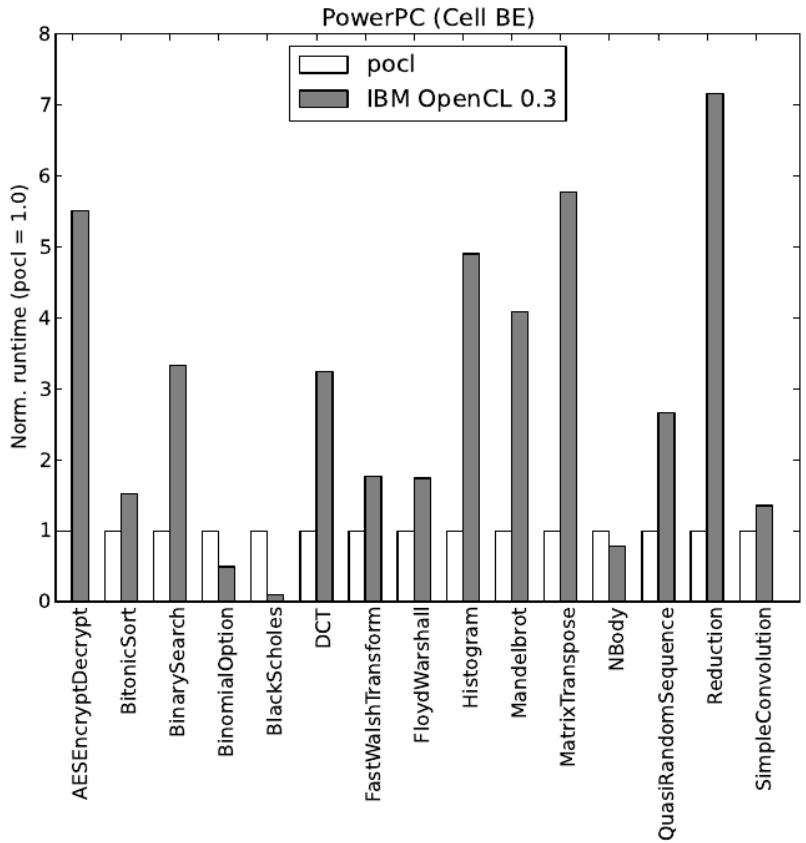
Stratton et al. *"MCUDA: An Efficient Implementation of CUDA Kernels for Multi-core CPUs"* in International Workshop on Languages and Compilers for Parallel Computing (2008)

"pocl: A Performance-Portable OpenCL Implementation" in International Journal of Parallel Programming (Vol 43, 2014).

Original PoCL WILoops performance (2014)



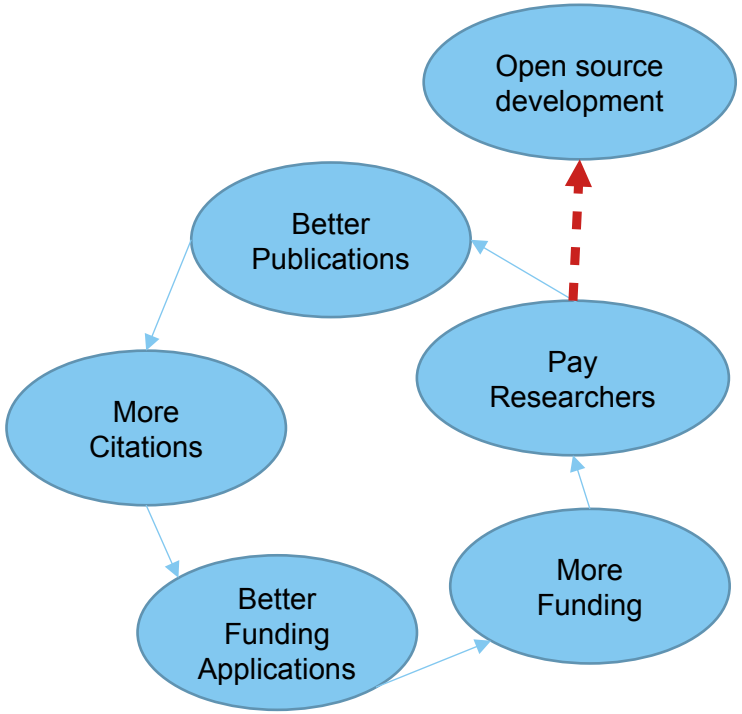




"pocl: A Performance-Portable OpenCL Implementation" in International Journal of Parallel Programming (Vol 43, 2014).

University Research: ”The Game”

- The ”end score” *should be* the number of **citations to one’s publications**
  - Considered a key measure of research quality / relevance in global rankings
  - Not ideal as it favors popular topics of a time and leads to ”hot topic snowballing”
- To get citations, attention should be paid to
  - The forum: Prefer well-known respected conferences / journals:
    - Likely more critical peer reviewers meaning **harder to get the paper accepted**, but when accepted, other researchers **trust** the publication more and are **more likely to cite it** in theirs
    - Top forums are read by a wider range of researchers
  - However, a publication in a ”smaller forum” is **much better** than no publication at all, as long as it is indexed and can be advertised, it’s always a proof that something potentially interesting was done by the group
- The citations and publications help to **get more funding** to hire more skilled people to produce even better publications
  - In some funding sources they also evaluate the merits of the principal investigators
- Only publications (papers) get cited
  - Open source project development as such is considered ”mere engineering work” (not research) and is as such not valued by the Ministry of Education etc. who quantify the performance of the University by ”traditional scientific merits”

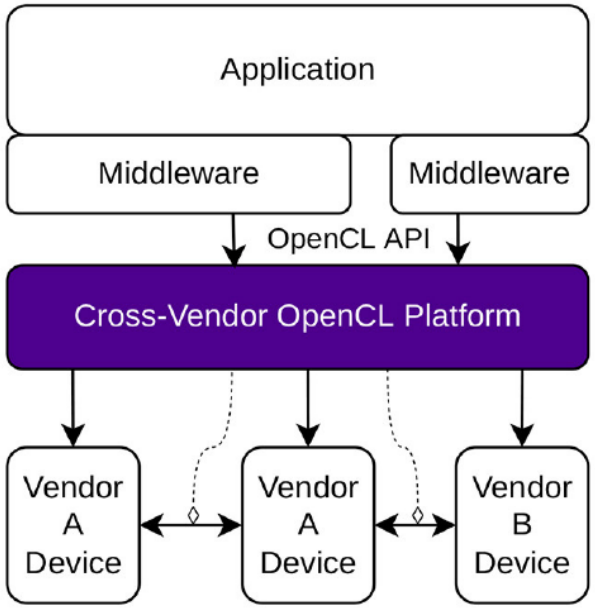


2014-2021: Mostly maintenance mode

- PoCL open source code base was mostly maintained on the side of research projects
  - Periodical releases to keep the project alive, up to date with the latest LLVM releases and usable for further research
- First (nearly) full-time PoCL developer (Michal Babej) hired in 2016
  - Supported by HSA Foundation grants (HSA backend, HIPCL) and via a few European projects (PoCL-R, ...)
- Some engineering work through my consultancy company (Parmance) on a custom proprietary driver
- **None of the funding was for improving the CPU+SIMD vectorization nor working on the spec conformance level while the community was (still is) using PoCL mostly as an open source OpenCL implementation for CPU+SIMD machines.**

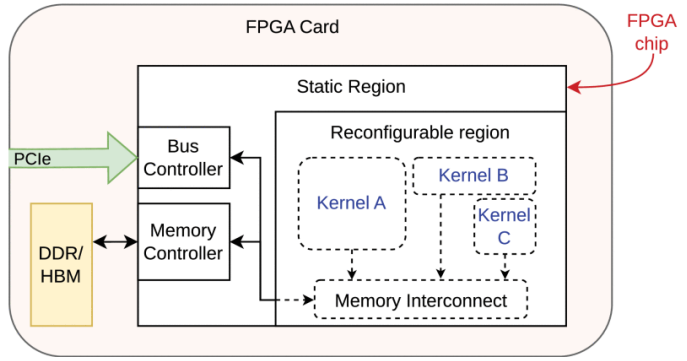
2022-2025: Working on PoCL in Intel

- The Parmance team of 3 was acquired by Intel to work on certain open source projects of Intel's interest at that time
- Could focus more on PoCL and CPU+SIMD optimization + multi-device platform experiments without publication pressures
- Goal: Heterogeneous (3-device) platform full-system optimization within a single OpenCL runtime using command buffers, etc.
- PoCL improvements done:
  - A new experimental Level Zero driver (PoCL-LZ)
  - Finally (!) OpenCL 3.0 conformance stamp for x86 and Intel iGPU via PoCL-LZ in PoCL v7.0
  - Interfacing with Intel NPUs using Defined Built-in Kernels (DBK) https://github.com/KhronosGroup/OpenCL-Docs/pull/1007
  - Rework of the CPU kernel compiler started



2022-2025: Meanwhile in Tampere Uni [1/2]

- Portable programming of FPGAs by using standard OpenCL as the middleware API
  - Interfacing with fixed function IP blocks using Defined Built-in Kernels (DBK)
  - Bitstream database mechanism for exchanging pre-designed FPGA IPs implementing DBKs
  - Study on the FPGA-related pain points of OpenCL 2.0 pipe spec
  - OpenCL C + command buffers to RTL synthesis using open source components
- Topi Leppänen's doctoral thesis project, successfully defended on April 30



"Unified OpenCL Integration Methodology for FPGA Designs" in IEEE Nordic Circuits and Systems Conference (11/2021).

"Cross-vendor programming abstraction for diverse heterogeneous platforms" in Frontiers in Computer Science (10/2022).

"Efficient OpenCL system integration of non-blocking FPGA accelerators" in Microprocessors and Microsystems (3/2023).

"AFOCL: Portable OpenCL Programming of FPGAs via Automated Built-in Kernel Management" in IEEE Nordic Circuits and Systems Conference (11/2023).

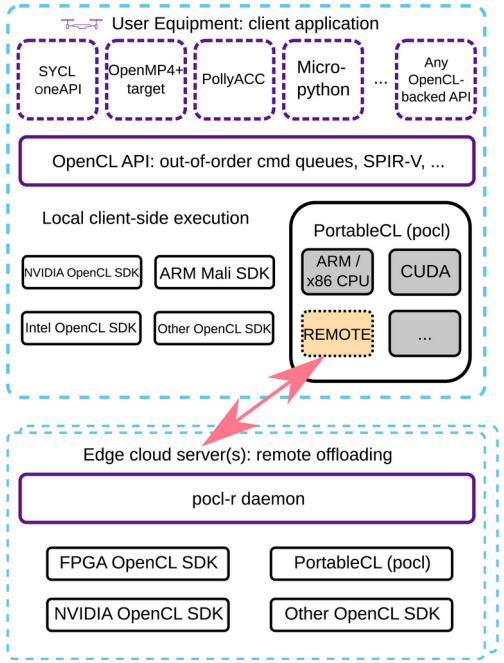
"Towards Efficient OpenCL Pipe Specification for Hardware Accelerators" in 12th International Workshop on OpenCL and SYCL (4/2024).

"Bitstream Database-Driven FPGA Programming Flow Based on Standard OpenCL" in IEEE Transactions on Very Large Scale Integration (9/2024).

"Composable Open-Source Toolchain for Synthesizing Hardware Accelerators from OpenCL Command Buffers" " in ACM Transactions on Reconfigurable Technology and Systems (11/2025).

2022-2025: Meanwhile in Tampere Uni [2/2]

- PoCL-Remote: A PoCL driver that enables using remote OpenCL devices (PoCL-based or not) like they were local devices
  - Optimizations for server-side buffer management, peer-to-peer, buffer RDMA, etc.
  - Command buffers to capture remotely (re-)invoked task graphs
  - Use cases in edge offloading and massive heterogeneous parallelization: Multi-device OpenCL programs can be made multi-node without code modifications (no MPI needed)
  - Preliminarily tested also “below” Java/Android, OpenMP target offload and multi-device SYCL applications



“PoCL-R: A Scalable Low Latency Distributed OpenCL Runtime” in SAMOS XXI (7/2021).

“Open Software Stack for Compression-Aware Adaptive Edge Offloading” in IEEE Wireless Communications and Networking Conference (3/2025).

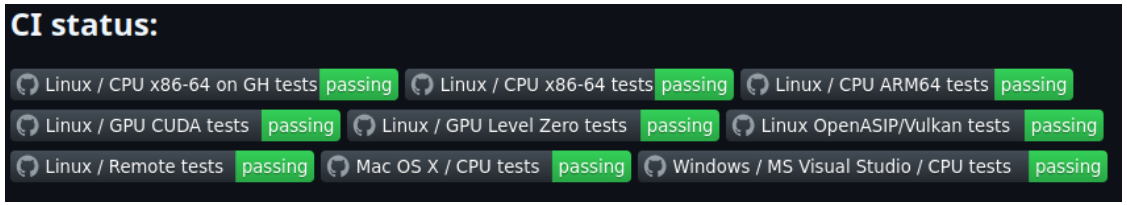
“Latency Reduction Potential of Server-Side Command Buffers in OpenCL-Based Edge Offloading” in 3th International Workshop on OpenCL and SYCL (4/2025).

“PoCL-R: An open standard based heterogeneous offloading layer with server side scalability” in International Journal of High Performance Computing Applications (9/2025).

https://tinyurl.com/poclr-videos

2026 May: PoCL technical status

- Considered stable: large part of the reported bugs end up actually being application-side bugs
- Various ”device drivers” in different levels of implementation completeness/stability:
  - CPU on x86-64 the most stable one, ARM64 and RISC-V follow
  - Layered GPU support via CUDA, Level Zero and Vulkan
  - Experimental-level special drivers: OpenASIP (the original target) and the remote
- Linux, Mac OS X, Windows (MinGW, MSVC), Android
- Multiple options for task runtime implementations:
  - Pthread, OpenMP and OneTBB

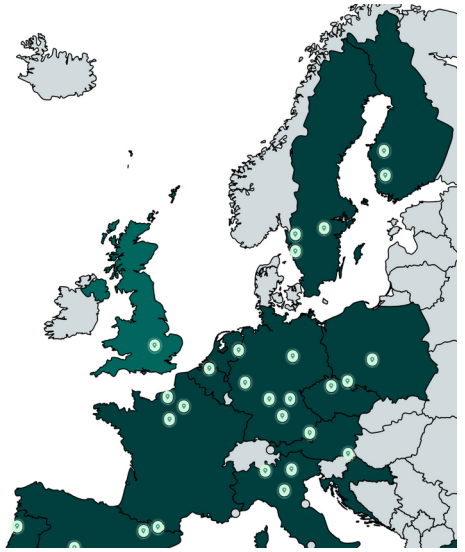


2026 May: Adoption status

- Solid adoption in open source and academic communities
  - Included in various Linux distros https://repology.org/project/pocl/versions
  - 2 open source GPUs (Vortex and Ventus) using it
- Commercial adopters – the ones I know about:
  - Texas Instruments, Kalray, Think Silicon
- Number of pull requests from outside my research group is still quite low
  - **Any suggestions how to increase?** Maybe we are too actively fixing issues by ourselves?

Future prospects: EU, HPC and open standards

- PoCL work in CPC group is currently funded mainly by the DARE SGA1 EU project
  - About 3.5 full time effort assigned on working on the software stack
  - Runs until 2/2028
- Layered software architecture based on open standards
  - OpenCL C and SPIR-V input
  - SYCL input via OpenCL & SPIR-V through AdaptiveCpp
  - CUDA/HIP input via OpenCL & SPIR-V through chipStar
  - Modular autovectorization optimized for RISC-V V (v1.0) using WILoops/DeSPMD + LLVM loop vectorizers



dare

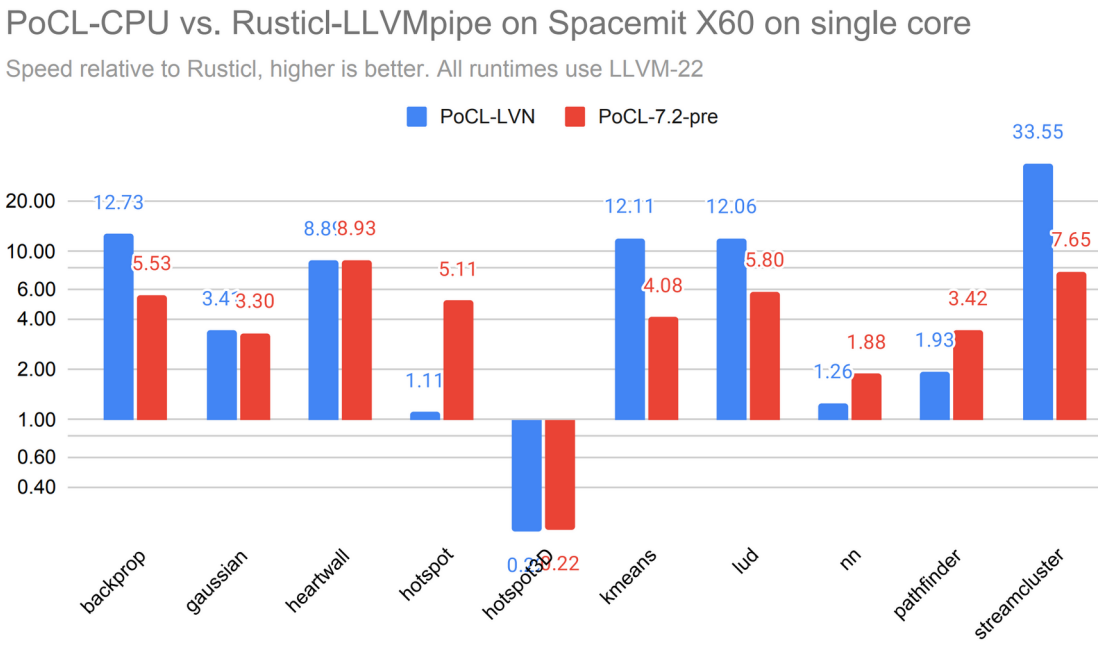
http://dare-riscv.eu/

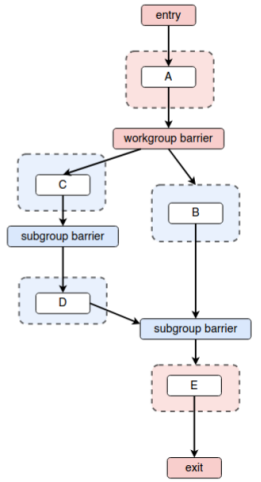
Coming soon...

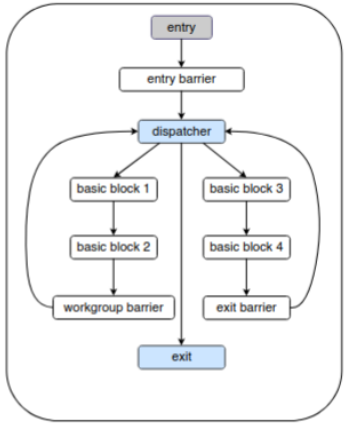
- More optimized and secure PoCL-R
  - Addressing multiple scaling bottlenecks and testing on supercomputer level clusters
  - Automated attestation and use of remote security features
- OpenCL v3.1 conformance on
  - RISC-V
  - x86-64
  - ARM64 (unsure)

Coming soon: DeSPMD

- Nearly complete rewrite of the kernel compiler for WG autovectorization, also known as DeSPMD or **L**oop**V**ec **N**ext (LVN)
  - Significant performance low hanging fruits picked
  - Proper subgroup support
  - Cleaned up, simplified, less bug-prone code base
  - Fiber fallback for the weirdest CFG cases
- LLVM upstreaming to benefit other SPMD languages and OpenCL CPU implementations (LLVMPipe?)
- Journal article in works, after that is out, the code will be pushed to the public PoCL repo







Conclusions

- PoCL started as a VLIW compiler research project, then generalized to become an OpenCL implementation framework
- Getting to the state where it is now, has been a long journey, sometimes quite of a struggle
  - Lots of engineering work without getting a paper published is difficult to justify in University setting
- Now conformant and stable for CPUs
  - Valid bug reports are rare
- Will get an even better kernel compiler **Soon**

Tampere University

Thank you!

Questions / comments?


pekka.jaaskelainen@tuni.fi


https://tuni.fi/cpc


@pekka@fosstodon.org

The DARE SGA1 project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101202459.