Using SYCL for the next generation heterogeneous systems

Mehdi Goli

16 May 2024

codeplay

Enabling AI & HPC To Be Open, Safe & Accessible To All

Established 2002 in **Edinburgh, Scotland.**

Grown successfully to around 100 employees.

In 2022, we became a **wholly owned subsidiary** of Intel.

oneAPI

Committed to expanding the **open ecosystem** for heterogeneous computing.

Through our involvement in oneAPI and SYCL governance, we help to **maintain and develop** open standards.

Developing at the forefront of **cutting-edge research.**

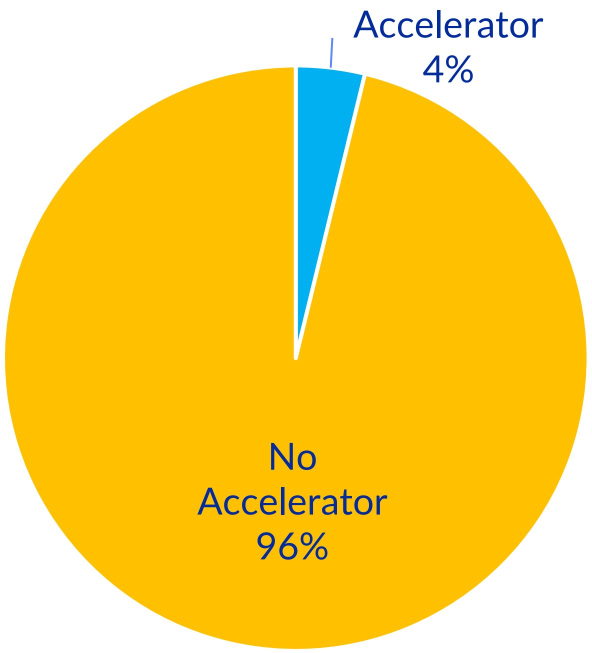
Currently involved in two research projects - **SYCLOPS** and **AERO**, both funded by the Horizon Europe Project.

Agenda

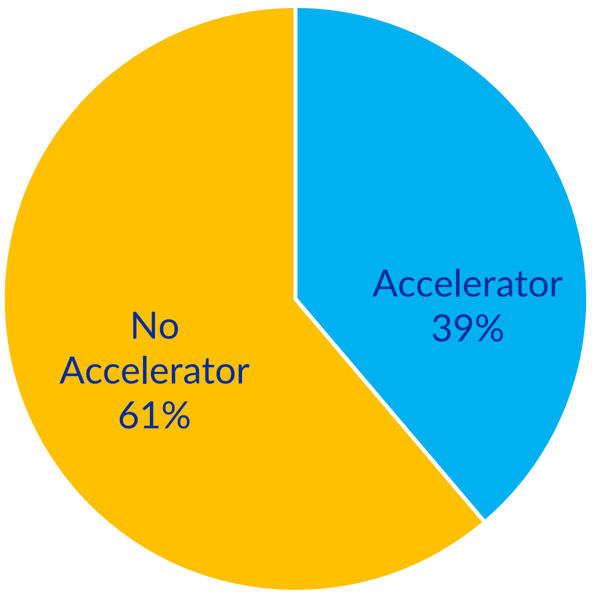
- New generation of computers and HPC
- How to achieve Portability and Performance
- Example SYCL usage in real world applications
- SYCL As a solution for enabling New architecture
- Active EU research projects

Heterogeneous Computing is Ubiquitous

Systems with Accelerator in Top500 – June 2011



Systems with Accelerator in Top500 – May 2024



https://www.top500.org/statistics/list/

Hardware: From CPU to GPU to other Accelerators

CPUs	GPUs	Other Processors
• Great at latency-sensitive tasks (reacting quickly) • Huge, well-developed ecosystem	• Great at graphics • Great at processing lots of floating-point data • Huge, well-developed ecosystem for graphics • Single-vendor ecosystem for non-graphics data-intensive applications	• Great at specific tasks • Much longer time-to-market • No open ecosystem for software • High performance in Specific domain only (e.g AI)

Large Language Models (LLMs)

- Text Classification
- Question Answering
- Document Summarization
- Text Generation

- Retail
- Finance
- Entertainment

- Large, general-purpose language models
- Can be pre-trained and then fine-tuned for specific purposes

Scientific Applications

- GROMACS
- The GROMACS molecular simulation toolkit.
- 1M line of code
- https://gitlab.com/gromacs/gromacs

- ATLAS Athena Project
- Research project at CERN
  - Focus on HPC
  - Scientific Computing
- 4M line of code
- https://gitlab.cern.ch/atlas/

The Opportunity vs the Challenge

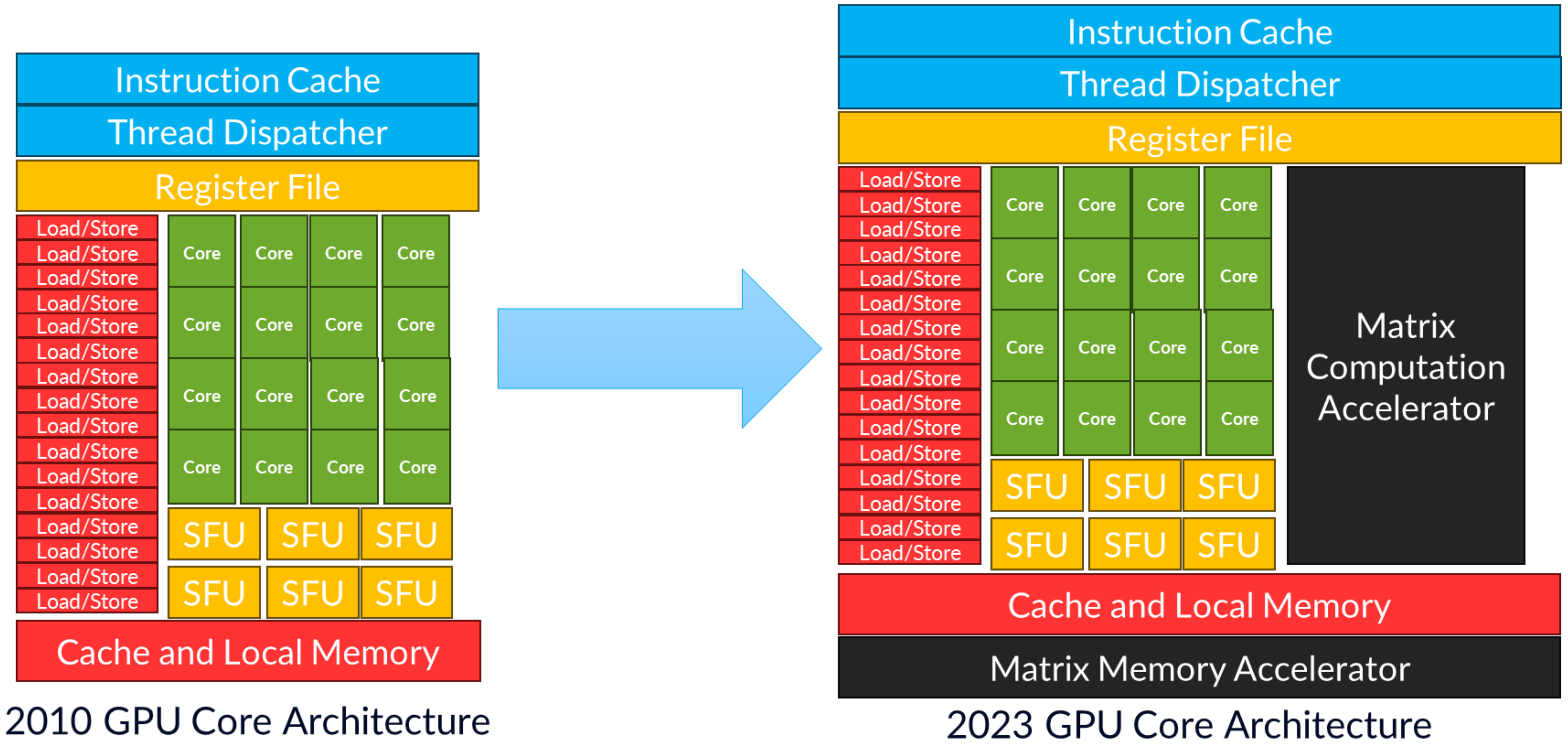
Accelerators give the performance we need for future software innovations

If we don't enable an accelerator ecosystem, we provide the performance but not the software innovations

The Heterogenous Programming Challenge

- Heterogeneous computing is inevitable, …
- …but
  - It’s hard
    - Requires **expert knowledge** and **significant experience** to achieve **maximum performance**
    - Need to **learn** about underlying **device architecture** to get right
    - Different strategies required for different types of accelerators
  - Continues to get more complicated
    - Accelerator vendors keep on innovating to give us faster devices
    - Vendors include more specialized functional units
    - The accelerators in a heterogeneous system become heterogeneous themselves!

The Heterogeneous Programming Challenge



The solutions

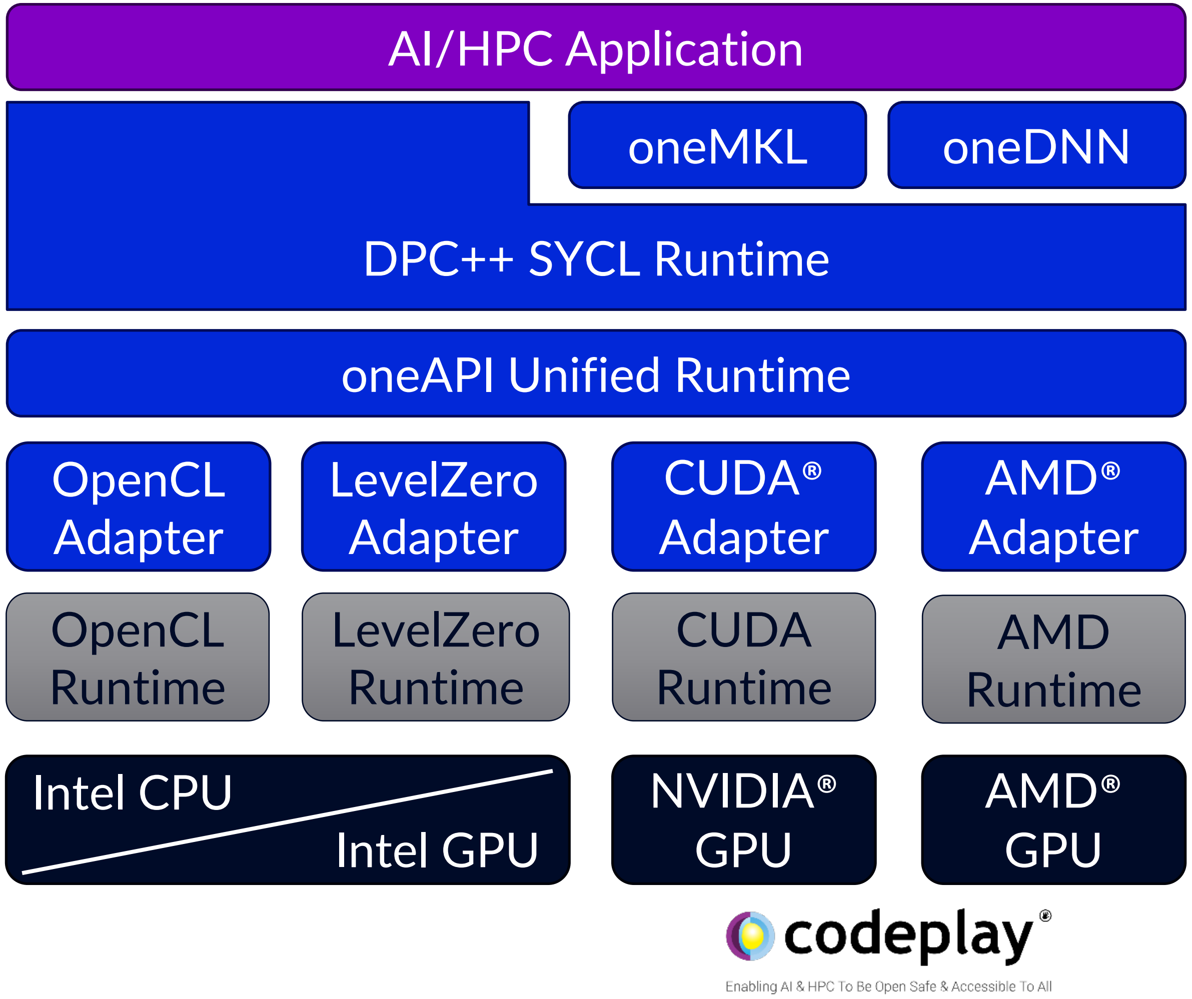
- We know the technology answers
  - There isn’t one solution for everyone, so we need to integrate the best-in-class technology solutions into one ecosystem-ready integrated solution

- We know the organizational answers
  - We know that industry standards and open-source projects can deliver ecosystem-friendly, innovator-friendly platforms for innovation

The C++ approach: SYCL

- SYCL is a royalty-free vendor-neutral industry standard C++ for parallel software and accelerator processors
- **SYCL takes proven C++ performance ideas & super-charges them for a heterogeneous processing world**
- Now we can:
  - Build our own C++ SYCL compilers for a variety of new processors
  - We can design our own optimizations
  - We can build C++ libraries that can adapt to the performance requirements of lots of different systems
  - We can integrate native compilation for different processors in one source file

The oneAPI Software Stack for SYCL

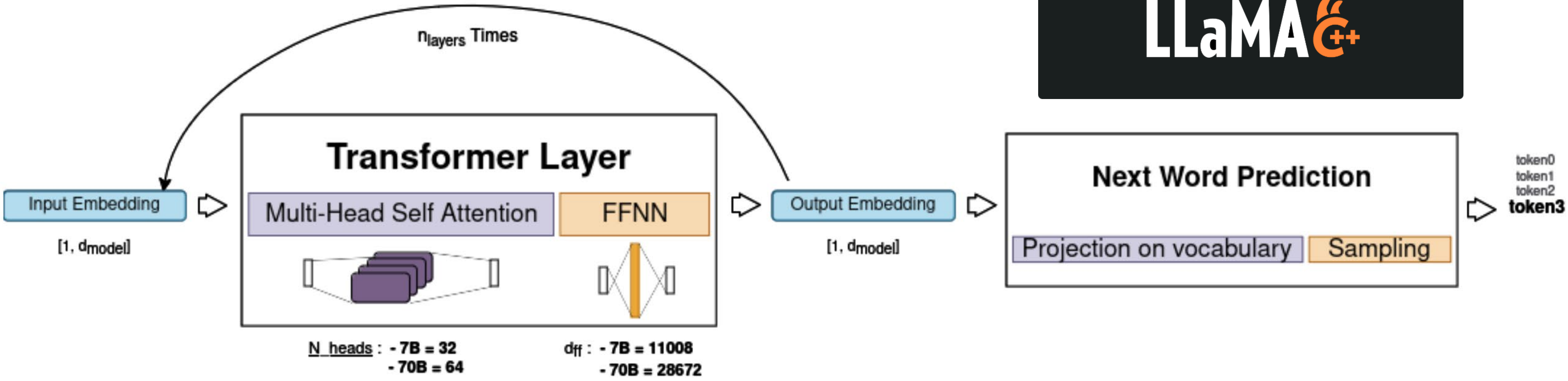


Example of SYCL Usage for Real world applications

Llama.cpp and GROMACS

What is Llama.cpp

- Llama (Large Language Model Meta AI) is a family of autoregressive large language models released by Meta AI starting in February 2023. The latest version is Llama 3 released in April 2024.
- Llama.cpp is an Inference of Meta's LLaMA model (and others) in pure C/C++
- The main goal of llama.cpp is to enable LLM inference with minimal setup and state-of-the-art performance on a wide variety of hardware - locally and in the cloud.



General Structure

- Llama.cpp generates a graph from a model file
- The graph is executed using a GGML backend
- Backends are
  - CUDA® /metal/Vulkan/OpenCL
- Contribution:
  - Adding SYCL support for NVIDIA® and Intel
  - ggml-sycl.h and ggml-sycl.cpp
- Current Work
  - Improving SYCL kernel performance
  - Enabling SYCL support for AMD® backend

LLM inference in C/C++

llama ggml

Readme

MIT license

Security policy

Activity

58.2k stars

508 watching

8.3k forks

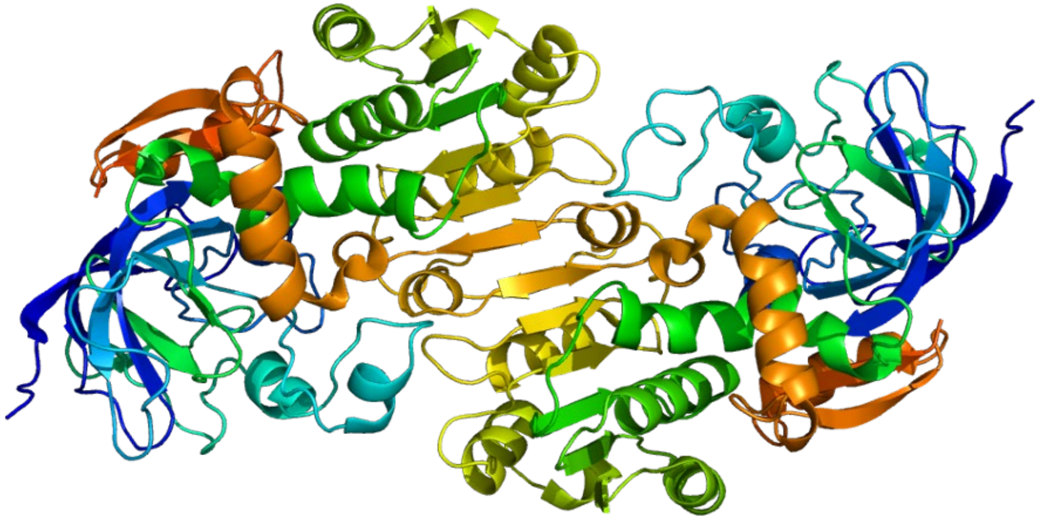
What is GROMACS?

Molecular dynamics

- Proteins
- Lipids
- Nucleic acids

Free and open source

Computationally very expensive



ADH5 protein

https://en.wikipedia.org/wiki/Alcohol_dehydrogenase#/media/File:Protein_ADH5_PDB_1m6h.png

GROMACS interactions

Bonded interactions

- Interaction between a fixed list of atoms. E.g. covalent bonds
- Can be accelerated with GPU kernels

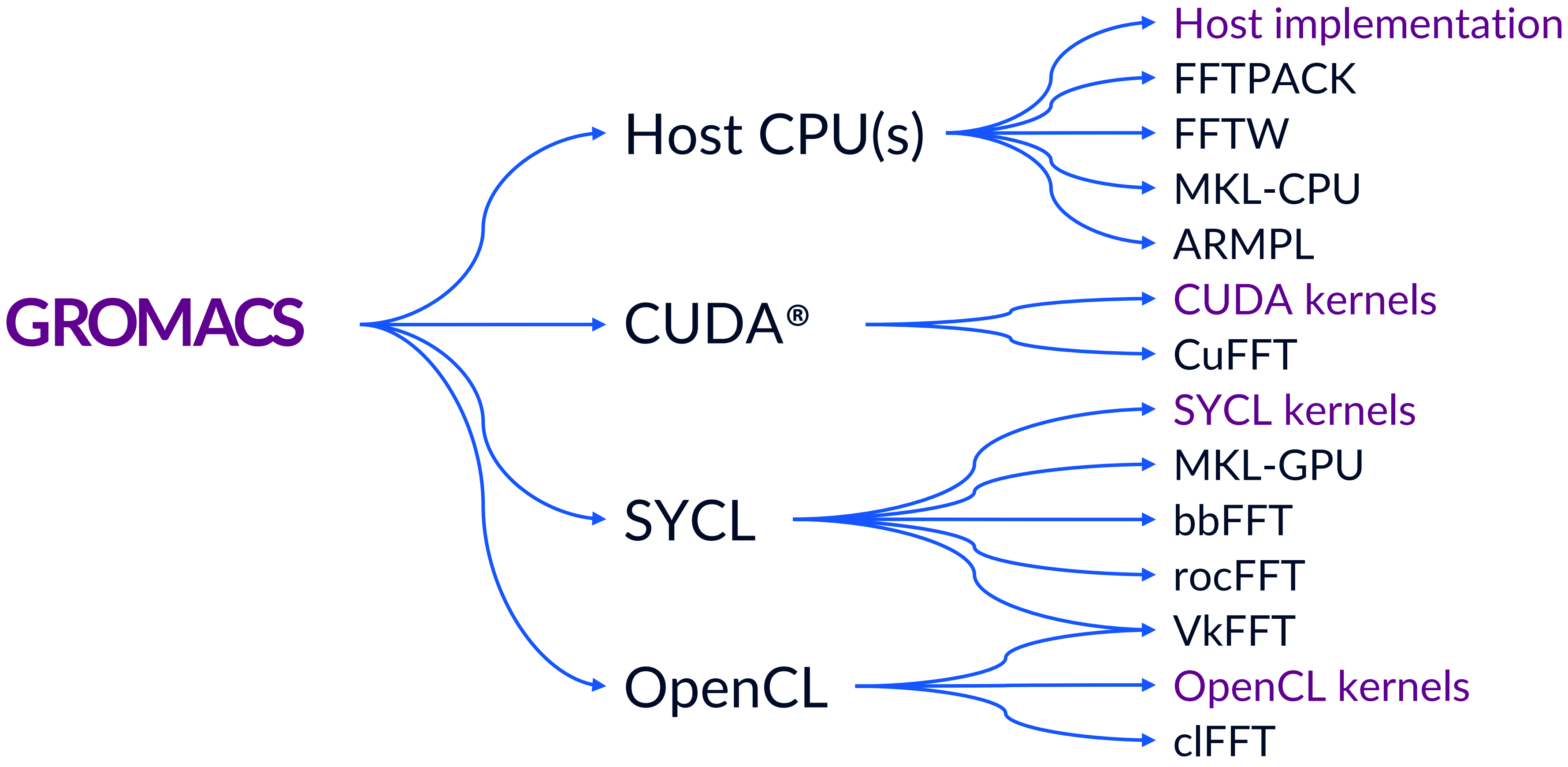
Contributions

- Enabling oneMKL interface to abstract out the call to FFT library on various hardware
- Improving DPCPP runtime to optimise the event time and interoperability with underlying libraries
- Improving GROMACS' SYCL based special kernels for short-range non-bonded forces.

Non-bonded interactions

- Interactions can be long-range, with every atom interacting with every other atom. E.g. electrostatic interactions between charged particles
- Convolutions can be FFT accelerated after interpolation onto a grid for particles that are far apart
- A short-range non-bonded force kernel (aka *NBNXM*) is used for close-together particles. This is usually GROMACS' longest-running kernel.

GROMACS' portability means duplication



... And that's not even every FFT backend!

CUDA is a registered trademark of Nvidia Corporation

A lower maintenance alternative

GROMACS → SYCL

- Custom kernels
  - NBNXM
  - Interpolation
  - And more!
- oneMKL Interface Library
  - MKL-CPU
  - CuFFT
  - MKL-GPU
  - rocFFT
  - portFFT

Write SYCL kernels that are portable across multiple vendors

Take the vendor's optimized FFT library with you

Improvements to DPC++ and Codeplay's plugins for AMD® and NVIDIA® GPUs have the potential to improve every HPC code.

Performance

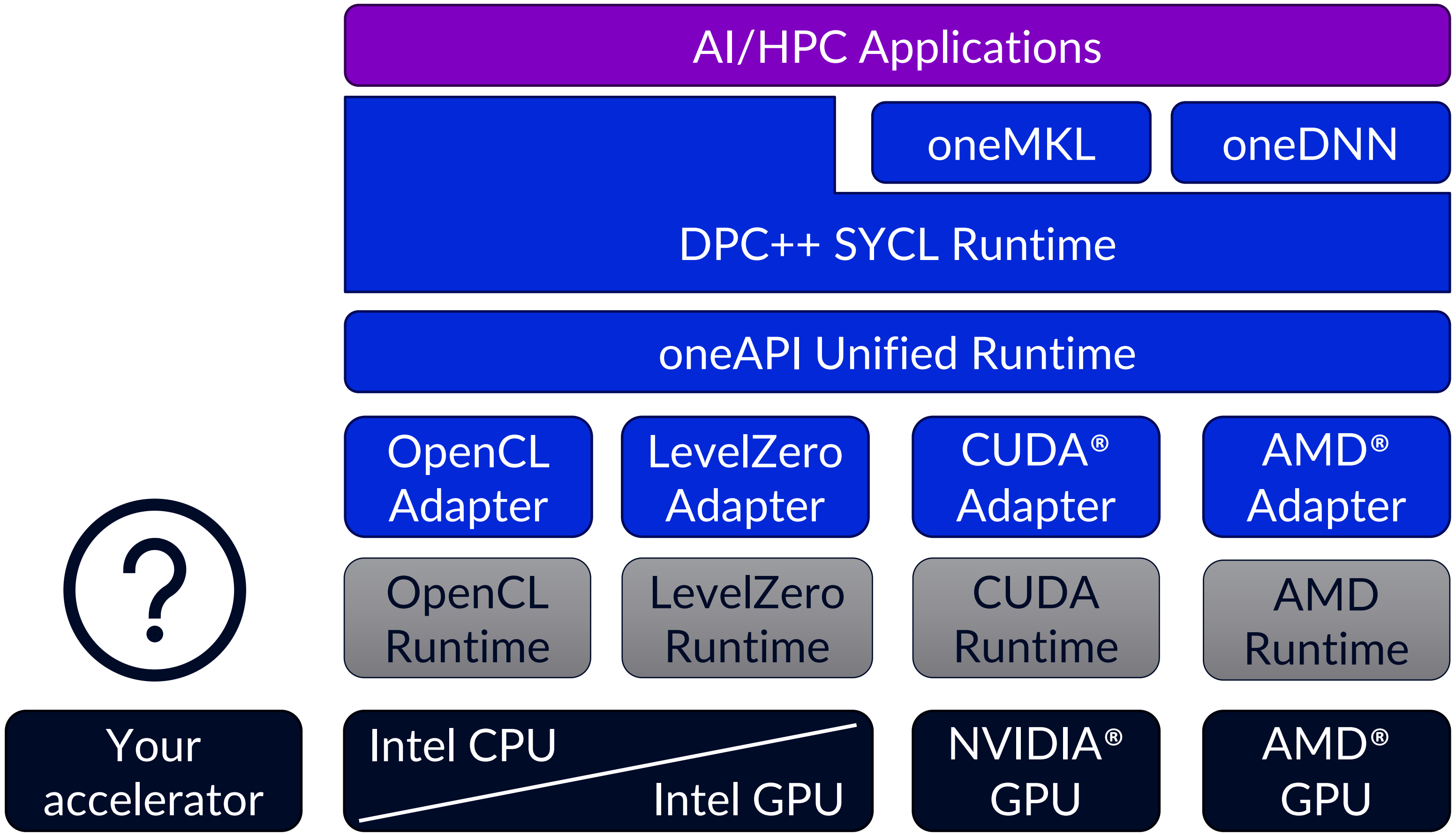
- Comparing native backends with oneMKL backend on 2 benchmarks: BenchMEM and ADHdodec using oneAPI DPC++ 2024.1.
  - Reaching **91%** geomean relative performance on A100 out of the box.
  - Reaching **84%** geomean relative performance on MI210 out of the box.
- Further optimisations:
  - Optimising the specific kernels written in SYCL(e.g *NBNXM*). This work improves the geomean relative performance to **94%** on A100.
  - Optimising the DPCPP runtime for host_task used for interoperability. This work will contribute to eliminate for ~20% of the performance difference on AMD.
  - Optimising events on DPCPP runtime for. This work will contribute to eliminate ~37% of the performance difference on AMD.

Benchmarked by HJA Bird on 15/04/2024 to 17/04/2024. A100 system: Intel Xeon Gold 6326 with Nvidia A100 PCIe 40GB. MI210 system: 2x EPYC 7402 with AMD Instinct MI210. Reference A100 binary compiled with CUDA 12.3, GCC 12.2 and GROMACS recommended configuration. Reference AMD binary compiled using AdaptiveCpp v23.10.0, ROCm 6.0.0, rocFFT GPU FFT library, HIP target GFX90a and GROMACS recommended configuration. Both DPC++ binaries were compiled with ICPX 2024.1 and matching Codeplay plugins for Nvidia and AMD GPU; CUDA 12.2 and ROCm 5.4.3 as appropriate; oneMKL main at commit f6263bc962; cluster pair splitting enabled for MI210; sycl targets for nvidia_gpu_sm80 and gfx90a, and the GROMACS recommended configuration. The AHDdodec and benchMEM benchmarks were used. All possible work was assigned to GPU. The number of OpenMP threads was set equal to the core count on a single socket. A single MPI domain was used. Performance was measured in ns/day, as reported by GROMACS counters, for the second 15000 steps of a 30000-step run.
Your costs and results may vary.

SYCL for Specialized Hardware

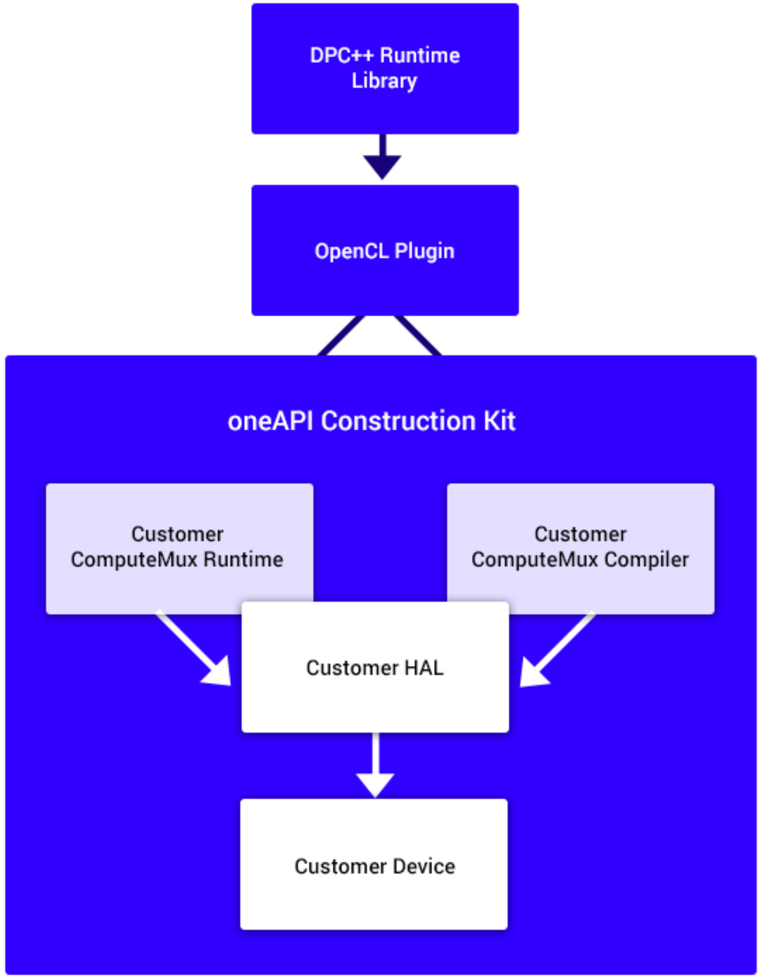
RISC-V and PIM

The oneAPI Software Stack for SYCL

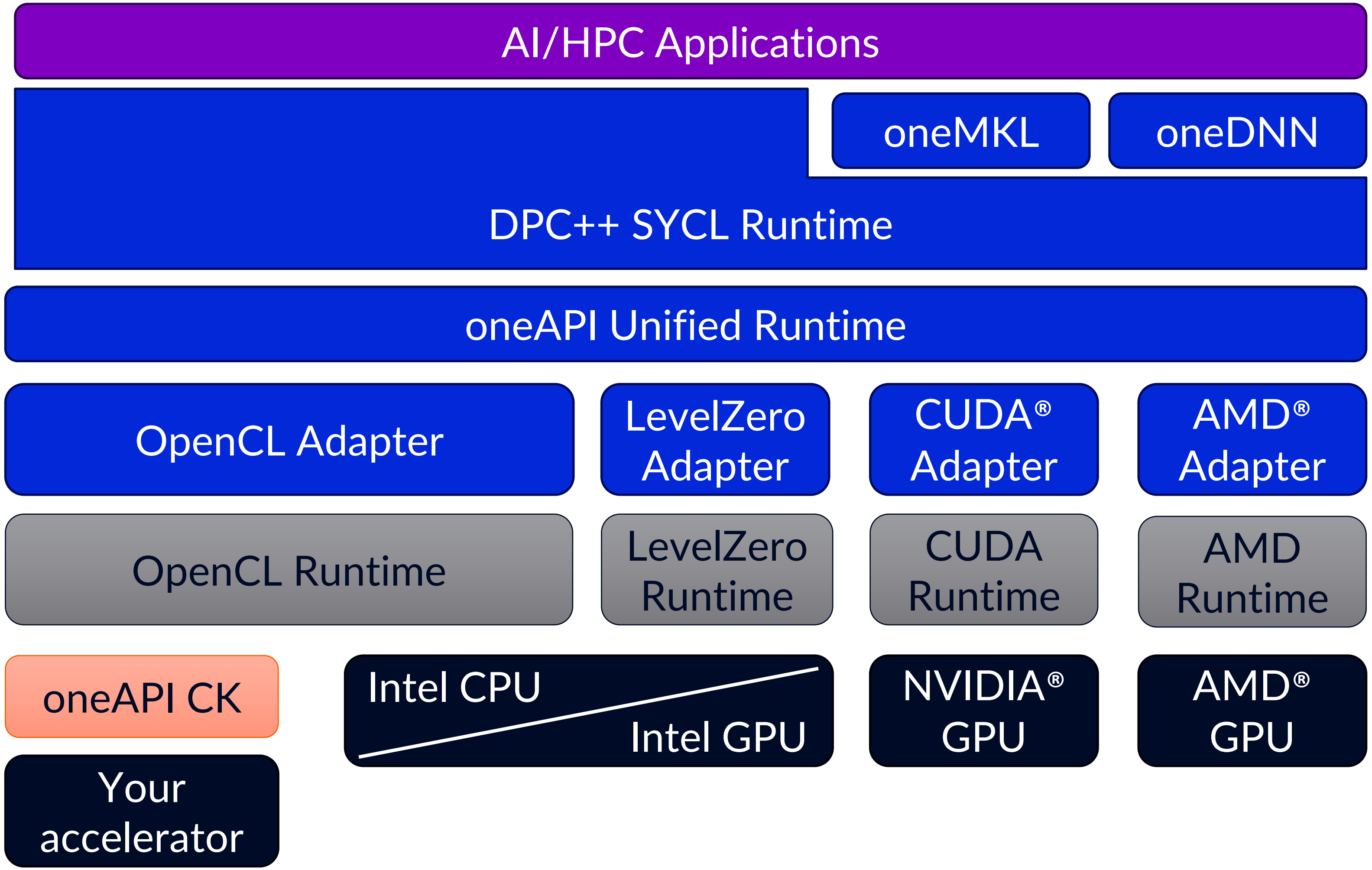


oneAPI Construction Kit

- oneAPI Construction Kit enables integration of custom accelerators into the oneAPI software stack
- Only need to provide
  - Runtime component
    - e.g., data movement between host and accelerator
  - Device binary compiler
    - Sufficient to compile from SPIR-V to accelerator binary
    - Prior compilation from SYCL to SPIR-V handled by DPC++ compiler
- oneAPI Construction Kit is open-source!
  - https://github.com/codeplaysoftware/oneapi-construction-kit



The oneAPI Software Stack for SYCL



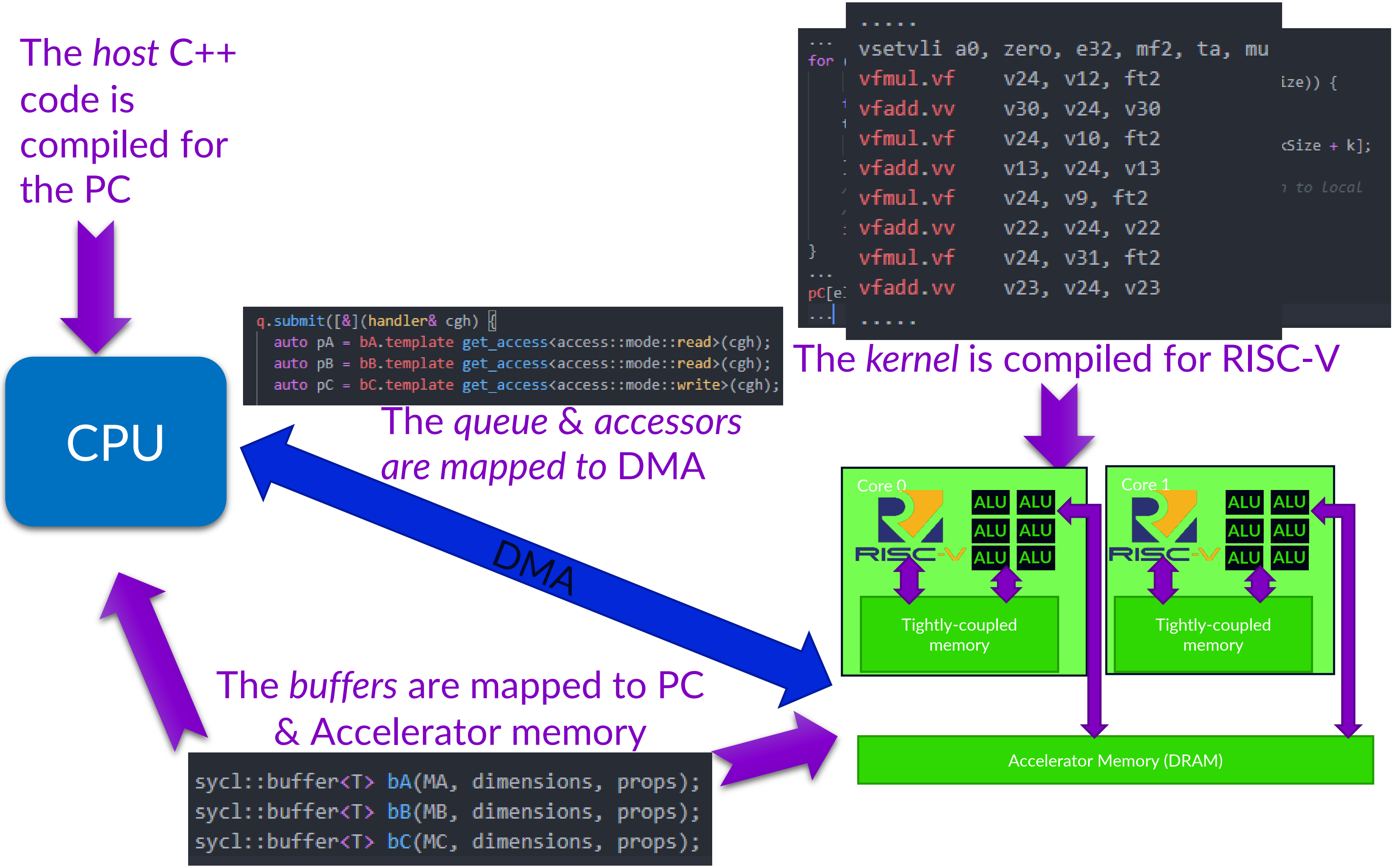
Enable SYCL and oneAPI for your Processor

Open-Standards, Driven by Industry Collaboration

Take Advantage of a Huge Existing Ecosystem

Enabling SYCL based MatMul for a Vector RISC-V Accelerator using OCK

```cpp
{   ....
    range<1> dimensions(matSize * matSize);
    const property_list props = {property::buffer::use_host_ptr()};
    buffer<T> bA(MA, dimensions, props);
    buffer<T> bB(MB, dimensions, props);
    buffer<T> bC(MC, dimensions, props);

    q.submit([&](handler& cgh) {
      auto pA = bA.template get_access<access::mode::read>(cgh);
      auto pB = bB.template get_access<access::mode::read>(cgh);
      auto pC = bC.template get_access<access::mode::write>(cgh);
      auto localRange = range<1>(blockSize * blockSize);

      accessor<T, 1, access::mode::read_write, access::target::local> pBA(
          localRange, cgh);
      accessor<T, 1, access::mode::read_write, access::target::local> pBB(
          localRange, cgh);

      cgh.parallel_for<mxm_kernel>(
          nd_range<2>{range<2>(matSize, matSize),
                      range<2>(blockSize, blockSize)},
          [=](nd_item<2> it) {
            // Current block
            int blockX = it.get_group(1);
            int blockY = it.get_group(0);

            // Current local item
            int localX = it.get_local_id(1);
            int localY = it.get_local_id(0);

            // Start in the A matrix
            int a_start = matSize * blockSize * blockY;
            // End in the b matrix
            int a_end = a_start + matSize - 1;
            // Start in the b matrix
            int b_start = blockSize * blockX;

            // Result for the current C(i,j) element
            T tmp = 0.0f;
            // We go through all a, b blocks
            for (int a = a_start, b = b_start; a <= a_end;
                 a += blockSize, b += (blockSize * matSize)) {
              // Copy the values in shared memory collectively
              pBA[localY * blockSize + localX] =
                  pA[a + matSize * localY + localX];
              // Note the swap of X/Y to maintain contiguous access
              pBB[localX * blockSize + localY] =
                  pB[b + matSize * localY + localX];
              it.barrier(access::fence_space::local_space);
              // Now each thread adds the value of its sum
              for (int k = 0; k < blockSize; k++) {
                tmp +=
                    pBA[localY * blockSize + k] * pBB[localX * blockSize + k];
              }
              // The barrier ensures that all threads have written to local
              // memory before continuing
              it.barrier(access::fence_space::local_space);
            }
            auto elemIndex = it.get_global_id(0) * it.get_global_range()[1] +
                             it.get_global_id(1);
            // Each thread updates its position
            pC[elemIndex] = tmp;
          });
    });
  }
  return false;
}
```



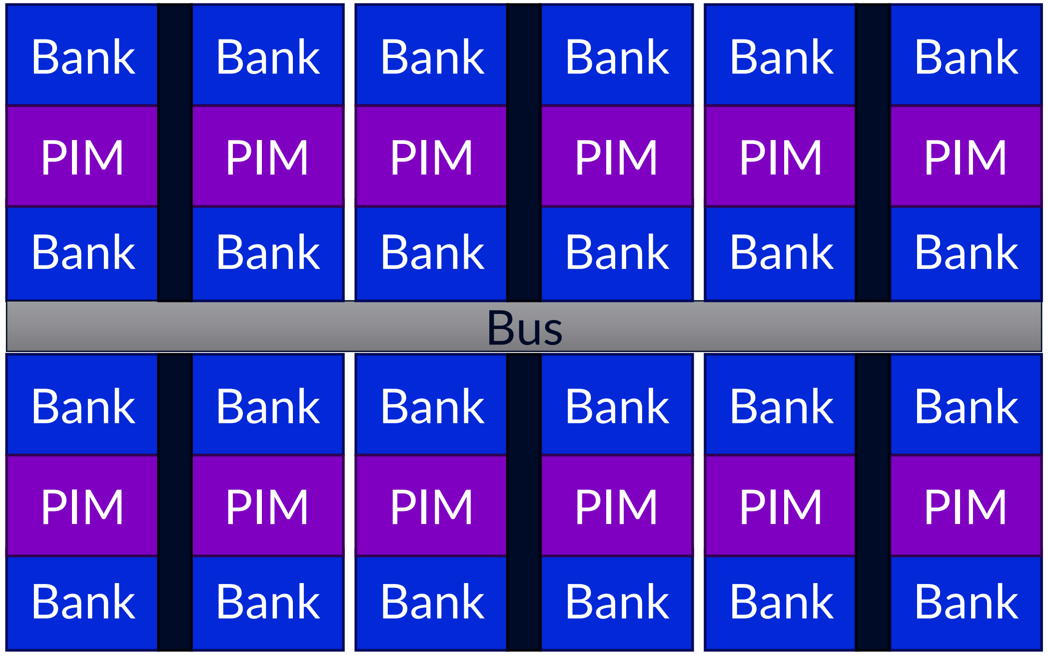
Processing in Memory (PIM)

- Despite improvements such as HBM, large gap between processor speed and memory speed
  - Memory access becomes bottleneck for applications with lower arithmetic intensity
- Memory speed limited by physical constraints, e.g, number of pins
  - Memory has much higher bandwidth internally than available via bus to processor
- Idea of PIM: Instead of moving data to processor, move processing to the data
  - Can significantly improve application performance for applications with lower arithmetic intensity

Samsung PIM

- PIM units integrated into HBM
- PIM units can perform arithmetic operations on fp16 data with much higher bandwidth
- Can be integrated with processors without modification of processor
  - Same physical dimensions
  - JEDEC compliant, no changes to memory controller on processor required

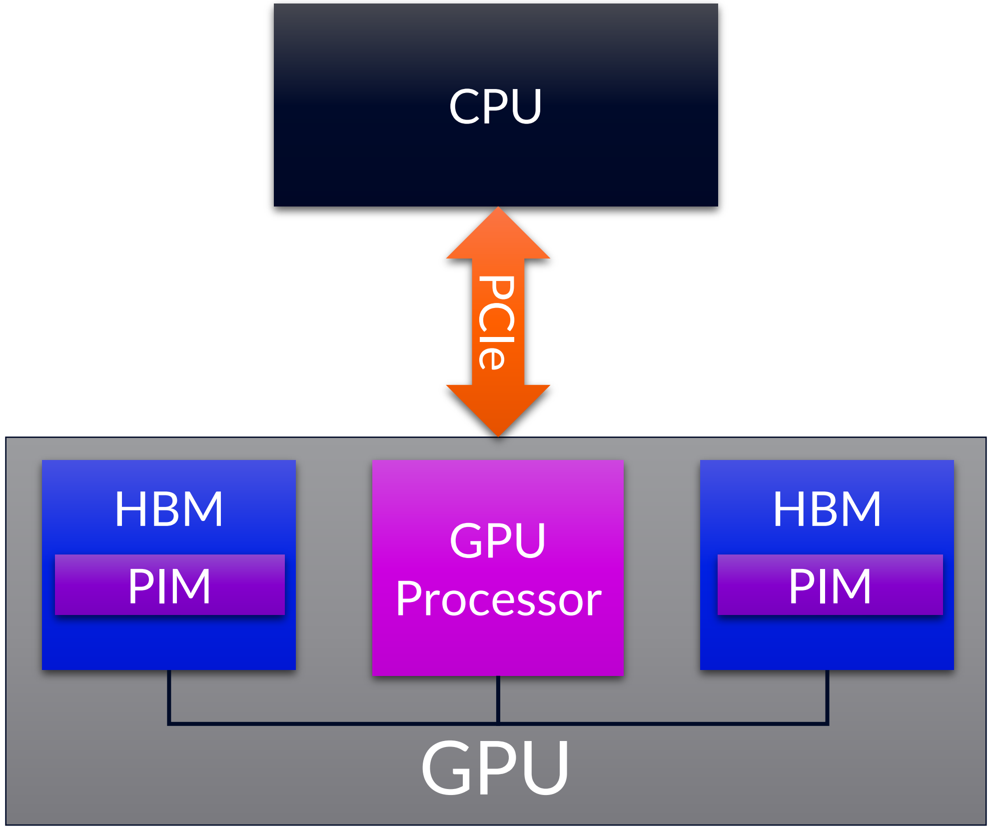
HBM DRAM Die



S. Lee et al., "Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology", 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)

Samsung PIM

- Example system: GPU HBM extended with PIM units
- PIM execution controlled through GPU memory controller
- Heterogeneous system becomes even more heterogeneous
- New challenge for programming models: How can we expose additional heterogeneity to user?



Integrating PIM into SYCL

- Variant 1 – host-initiated PIM operations

- PIM operations as separate SYCL command invoked from host
  - Execution still controlled by GPU

- Full integration into SYCL runtime requirements and dependency tracking

```
queue q{ext::pim_selector{}};
half a = 2.0;
half x[dataSize];
half y[dataSize];

// y[i] = a + x[i]
{
   buffer<sycl::half> bX{x, range{dataSize}};
   buffer<sycl::half> bY{y, range{dataSize}};
   q.submit([&](ext::pim_handler &ph) {
      ext::pim_accessor pAccX{bX, ph, sycl::read_only};
      ext::pim_accessor pAccY{bY, ph, sycl::read_write};
      ph.elementwise_add(pAccY, a, pAccX);
   });
}
```

H. Hong et al., "Programming Model Extensions for General-Purpose Processing-in-Memory", 2024 ISC-HPC

Integrating PIM into SYCL

- Variant 2 – PIM as group functions
- PIM operations fully integrated into SYCL kernels as PIM operations
- Allows mix of GPU and PIM execution in the same kernel

```
queue q{pim_selector{}};
{
   buffer<half> b_a{a, range{dataSize}};
   buffer<half> b_b{b, range{dataSize}};
   buffer<half> b_c{c, range{dataSize}};
   q.submit([&](handler &cgh) {
      auto acc_a = b_a.get_access<access_mode::read>(cgh);
      auto acc_b = b_b.get_access<access_mode::read>(cgh);
      auto acc_c = b_c.get_access<access_mode::write>(cgh);
      cgh.parallel_for<class Kernel>(nd_range<1>
         {range<1>{10}, range<1>{2}}, [=](nd_item<1> item) {
           auto groupID = item.get_group(0);
           auto* a = &acc_a[groupID * 128];
           auto* b = &acc_b[groupID * 128];
           auto* c = &acc_c[groupID * 128];
           joint_transform<64>(item.get_group(), a, b, c,
                                       sycl::plus<>(), 2);
      });
   });
}
```

H. Hong et al., "Programming Model Extensions for General-Purpose Processing-in-Memory", 2024 ISC-HPC

SYCLOPS EU project

Vision - Enable better solutions for AI/data mining for extremely large and diverse data by

- democratizing AI acceleration using open standards, and
- enabling a healthy, competitive, innovation- driven ecosystem

AccelOm

CERN

Codasip

codeplay

EURECOM Sophia Antipolis

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

hiro MICRODATACENTERS

inesc id lisboa

Three-year project launched in January 2023 with eight leading organizations

GRANT AGREEMENT NUMBER: 101092877

https://www.syclops.org/

UK Research and Innovation

Funded by the European Union

AERO EU project

Vision - enabling the future heterogeneous EU cloud infrastructure.

- Upbring and optimize the software ecosystem based on
  - managed programming languages
  - Native programming languages
  - OS, driver, and virtualization support infrastructure for EU cloud deployment

Three-year project launched in January 2023 with 12 leading organizations

Funded by the European Union. Views and opinions are however those of the author(s) only and do not necessarily reflect those of the European Union or the HaDEA. Neither the European Union nor the granting authority can be held responsible for them. Project number: 101092850.

AERO has also received funding from UKRI under grants no. 10048318 and 10048915, and the Swiss State Secretariat for Education, Research, and Innovation.

UK Research and Innovation

Funded by the European Union

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

https://aero-project.eu/

codeplay®

Social Media

Don’t forget to follow us for the latest updates!

@codeplaysoft

@codeplaysoftware

codeplay-software

codeplay-software

codeplay®

Disclaimers

A wee bit of legal

Performance varies by use, configuration and other factors.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details.

No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.