SYCL in HPC

Peter Žužek, Senior Software Engineer, SYCL Performance

Workshop on Efficient Computing for High Energy Physics, Edinburgh February 18th 2020

Leadership Products Enabling Advanced Applications on Complex Processor Systems

Company

High-performance software solutions for custom heterogeneous systems

Enabling the toughest processor systems with open-standards-based tools and middleware

Established 2002 in Scotland, UK

Markets

Vision Processing
Machine Learning
Data Compute

High Performance Computing (HPC)
Automotive (ISO 26262)
IoT, Smartphones & Tablets
Medical & Industrial

Products

ComputeCpp™

C++ platform with SYCL, enabling vision and machine learning applications e.g. TensorFlow™

ComputeAorta™

The heart of Codeplay's compute technology, enabling OpenCL™, SPIR™, HSA™ and Vulkan™

Partners

AMD
ARM
BROADCOM
Imagination
QUALCOMM
Movidius
Many Global Companies

Codeplay staff are closely involved in defining new open standards

Codeplay lead the definition of the SYCL standard in Khronos to target modern C+ AI acceleration

The ISO C++ body trust Codeplay to lead the transition to Machine Learning & heterogeneous parallelism in C++

Codeplay has joined AUTOSAR to deliver new autonomous drive capabilities

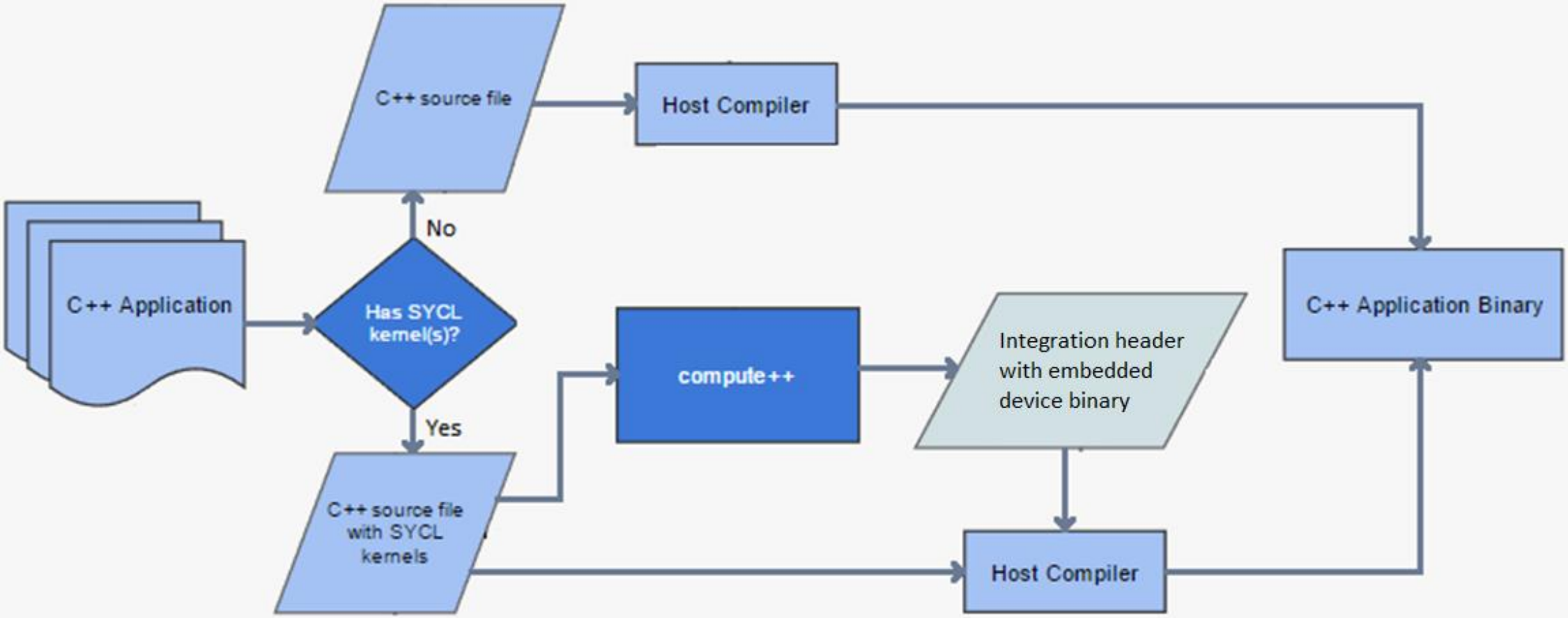
Codeplay lead the definition of the SYCL standard in Khronos to target modern C+ AI acceleration

- Single source heterogeneous programming model
  - Designed to run on host device, CPU, GPU, FPGA, DSP, accelerators …
- Cross platform, open standard alternative to CUDA
- Entirely standard C++
  - No #pragma, restrict, <<<range>>>
  - Long term aim: ISO C++

```
#include <CL/sycl.hpp>
using namespace cl::sycl;

#include <vector>
using std::vector;

vector<int> add_vectors(const vector<int>& a,
                        const vector<int>& b);


int main() {
  vector<int> a{1, 2, 3, 4, 5};
  vector<int> b{6, 7, 8, 9, 10};
  auto c = add_vectors(a, b);
  return 0;
}
```

```
vector<int> add_vectors(const vector<int>& a,
                        const vector<int>& b) {
  const auto N = a.size();
  buffer<int, 1> bufA(a.data(), range<1>{N});
  buffer<int, 1> bufB(b.data(), range<1>{N});

  vector<int> c(N);
  buffer<int, 1> bufC(c.data(), range<1>{N});

  queue myQueue;

  myQueue.submit([&](handler& cgh) {
    auto A = bufA.get_access<access::mode::read>(cgh);
    auto B = bufB.get_access<access::mode::read>(cgh);
    auto C = bufC.get_access<access::mode::write>(cgh);

    cgh.parallel_for<class add>(
      range<1>{N},
      [=](id<1> i) {
        C[i] = A[i] + B[i];
      }
    );
  });

  return c;
}
```

ComputeCpp compilation



Asynchronous Execution

- Command queue submissions are asynchronous by default
  - Can throw asynchronous exceptions -> async_handler
- Automatic management of data movement
- Can wait on events for manual synchronization

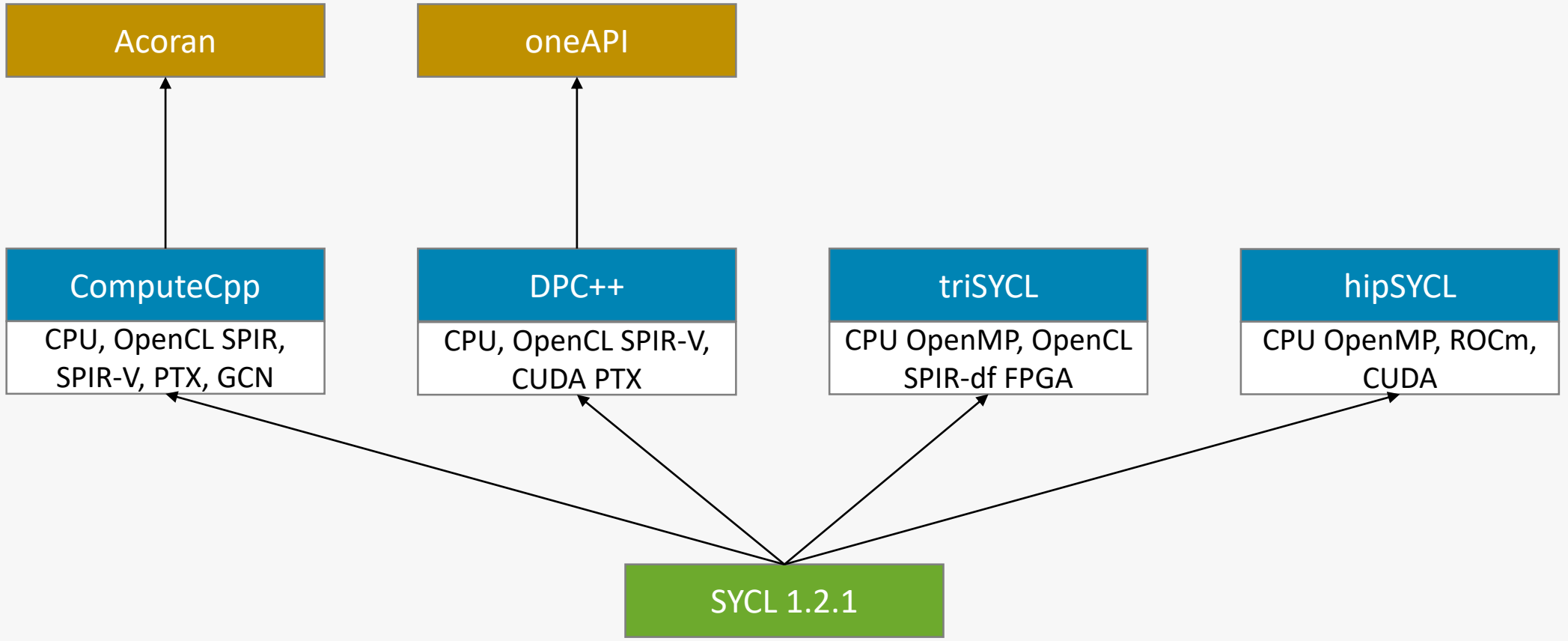
```
vector<float> add_vectors(const vector<float>& a, const vector<float>& b) {
  const auto N = a.size();
  const auto bufRange = range<1>{N};

  queue myQueue;

  buffer<float> bufA{bufRange};
  myQueue.submit([&](handler& cgh) {
    auto acc = bufA.get_access<access::mode::discard_write>();
    cgh.copy(a.data(), acc);
  });

  buffer<float> bufB{bufRange};
  myQueue.submit([&](handler& cgh) {
    auto acc = bufB.get_access<access::mode::discard_write>();
    cgh.copy(b.data(), acc);
  });

  vector<float> c(N, 0.f);
  bufA.set_final_data(c.data());

  myQueue.submit([&](handler& cgh) {
    auto A = bufA.get_access<access::mode::read_write>();
    auto B = bufB.get_access<access::mode::read>();
    cgh.parallel_for<class vec_add>(bufRange, [=](id<1> i) { A[i] += B[i]; });
  });
  return c;
```

SYCL Ecosystem

A range of open source project use SYCL to accelerate execution of complex algorithms

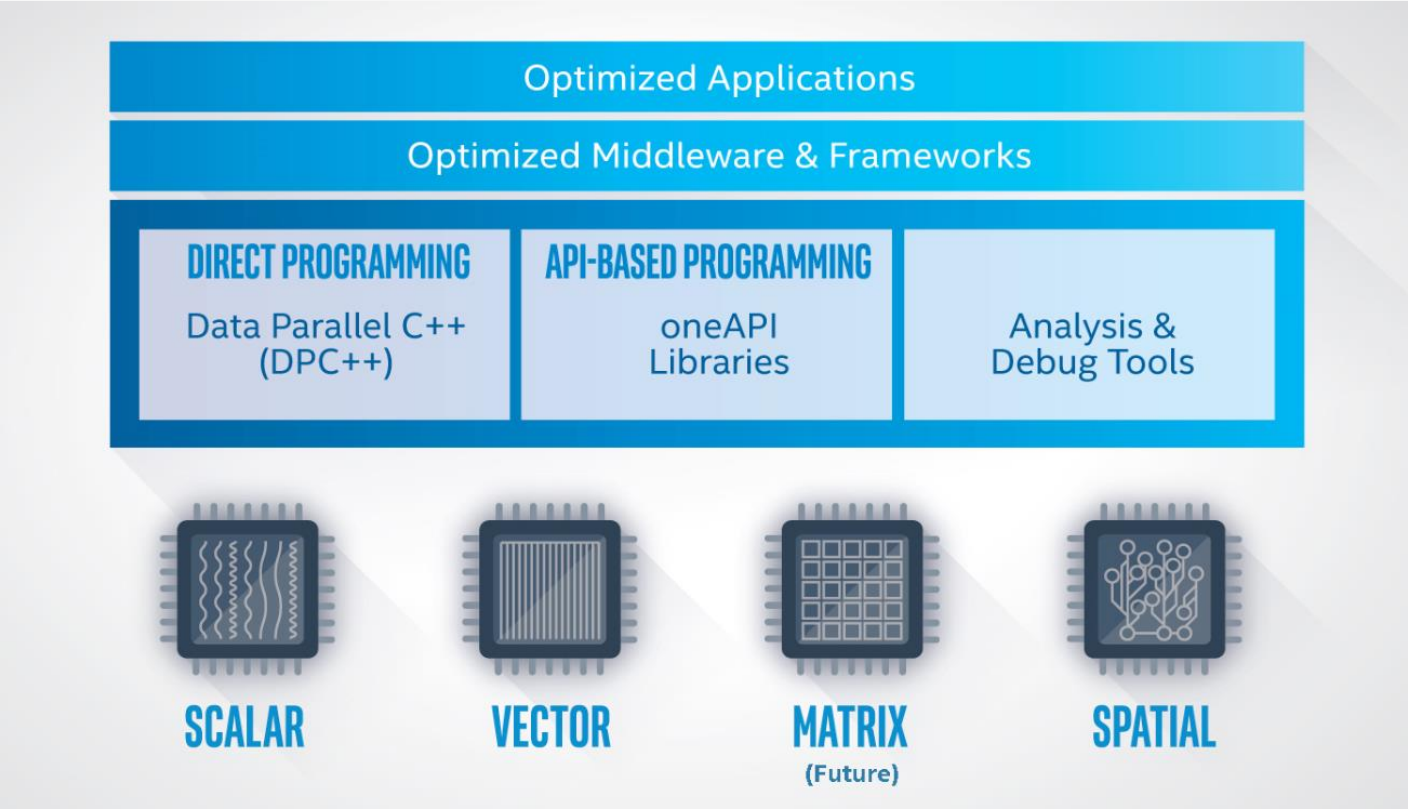
Linear Algebra	AI	Imaging	std C++	Benchmarks
Eigen	TensorFlow	VisionCpp	SYCL-ParallelSTL	BabelStream
SYCL-BLAS	SYCL-DNN	OpenGL Mathematics (GLM)		SLAMbench
	SYCL-ML			RSBench
	TensorOpt			ParRes

SYCL Implementations

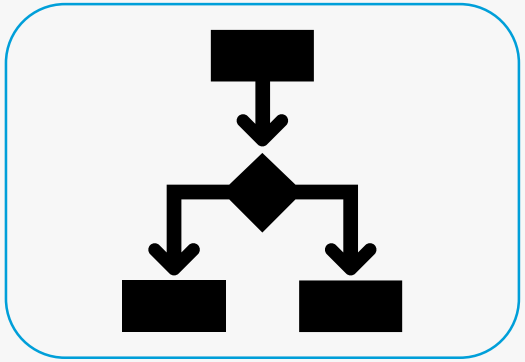


oneAPI

- Unified, Cross-Architecture Programming Model
- SYCL with DPC++ and extensions
- oneAPI libraries
- Optimized for Intel hardware

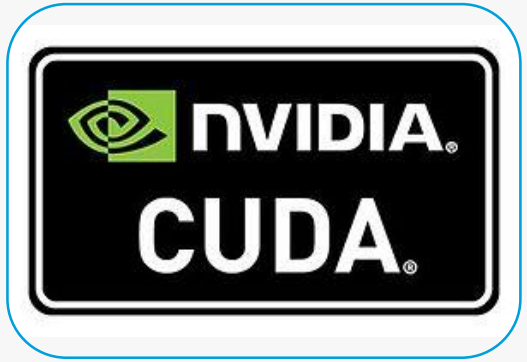


Software platforms for AI acceleration



Device-specific graph compilers

- Limited capabilities, rarely used in production
- Non-standard, hard to integrate

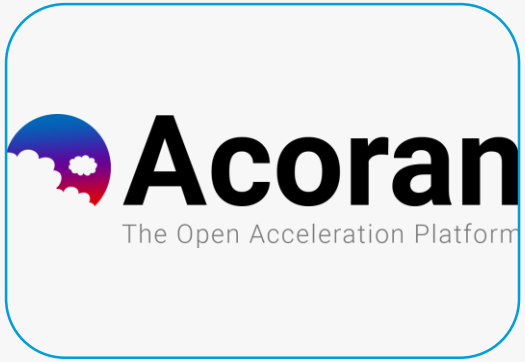


NVIDIA CUDA

- Widely adopted, full-featured
- Proprietary, locked to NVIDIA GPUs

Intel oneAPI

- Industry-standard, full-featured
- Optimized for Intel processors; GPU, AI, FPGA



Codeplay Acoran

- Industry-standard, full-featured
- Optimized for a wide range of platforms
- Available for new processors

Compatible

Unified Shared Memory extension

- USM pointers
  - Same representation and same location for all devices
- Easier to port HPC code
- 4 levels
  - Explicit, Restricted, Concurrent, System

```
vector<float> add_vectors(const vector<float>& inputA, const vector<float>& inputB) {
  const auto N = inputA.size();
  const auto bufRange = range<1>{N};

  queue myQueue;

  constexpr std::size_t alignment = 64;
  using allocator_t = usm_allocator<float, usm::alloc::device, alignment>;
  const auto alloc = allocator_t{myQueue};

  auto dataA = vector<float, allocator_t>{alloc};
  dataA.reserve(N);
  event copyA = myQueue.memcpy(dataA.data(), inputA.data(), N);

  auto dataB = vector<float, allocator_t>{alloc};
  dataB.reserve(N);
  event copyB = myQueue.memcpy(dataB.data(), inputB.data(), N);

  event kernelEvent = myQueue.submit([&](handler& cgh) {
    auto A = dataA.data();
    auto B = dataB.data();
    cgh.depends_on({copyA, copyB});
    cgh.parallel_for<class vec_add>(bufRange, [=](id<1> ID) {
      auto i = ID[0];
      A[i] += B[i];
    });
  });

  vector<float> c(N, 0.f);
  kernelEvent.wait();
  myQueue.memcpy(c.data(), dataA.data(), N).wait();
  return c;
```

ComputeCpp™

- SYCL 1.2.1 Conformant (CTS)
- Optimized data movement
- Application tuning via configuration file
- Vendor specific optimizations
- Address space deduction
- Targets SPIR and SPIR-V devices
  - Experimental support for PTX and GCN

Who is using SYCL?

https://sycl.tech/research/

- Wigner Institute, Heriot Watt University, Stellar Group, UWS, Argonne NL, Lawrence Berkeley NL, DOE, Oak Ridge NL, Abertay University, TU Dresden….

Research Papers & Benchmarks

If you are looking at using SYCL as a programming model for heterogeneous and parallel software development there are a wide variety of published independent research papers. Here we have linked details for some of these papers.

Research Papers

SYCL Code Generation for Multigrid Methods

Authors: Stefan Groth, Christian Schmitt, Jürgen Teich, and Frank Hannig

Multigrid methods are fast and scalable numerical solvers for partial differential equations (PDEs) that possess a large design space forimplementing their algorithmic components. Code generation ap-proaches allow formulating multigrid methods on a higher level of abstraction that can then be used to define a problem and hardware-specific solution. Since these problems have considerable implementation variability, it is crucial to define...

View Paper

Performance Portability of Multi-Material Kernels

Authors: Istvan Z. Reguly

Trying to improve performance, portability, and productivity ance portability and code divergence metrics, contrasting performance, portability, and productivityof an application presents non-trivial trade-offs, which are often difficult to quantify. Recent work has developed metrics for performance portability, as well some aspects of productivity - in this case study, we present a set of challenging computational kernels nd their implementations from...

View Paper

Performance portability of a Wilson Dslash Stencil Operator Mini-App using Kokkos and SYCL

Authors: Balint Joo, Thorsten Kurth, M. A. Clark, Jeongnim Kim, Christian R. Trott, Dan Ibanez, Dan Sunderland, Jack Deslippe

We describe our experiences in creating mini-apps for the Wilson-Dslash stencil operator for Lattice Quantum Chromo dynamics using the Kokkos and SYCL programming models. In particular we comment on the performance achieved on a variety of hardware architectures, limitations we have reached in both programming models and how these have been resolved by us, or may be resolved by the...

View Paper

Innovative language extensions for accelerator cards using the example of SYCL, HC, HIP and CUDA: research on usability and performance

Authors: Jan Stephan, Dr. Wolfgang E. Nagel

Translated from German: "The purpose of this work is a comparative analysis of the programming models CUDA, SYCL and ROCm (or HC and HIP) on GPUs of the

Celerity: High-level C++ for Accelerator Clusters

Authors: Peter Thoman, Philip Salzmann, Biagio Cosenza, and Thomas Fahringer

In the face of ever-slowing single-thread performance growthfor CPUs, the scientific and engineering communities increasingly turn toaccelerator parallelization to tackle growing application workloads. Ex-isting means of targeting distributed memory accelerator clusters

Improving the Performance of Medical Imaging Applications using SYCL

Authors: Zheming Jin

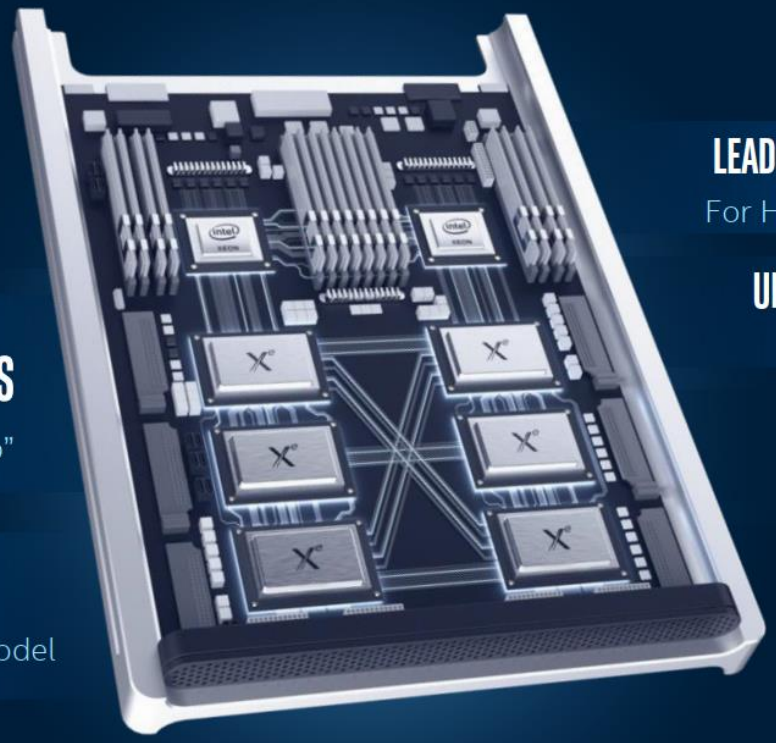
In this report, we are interested in applying the SYCL programming model to medical imaging applications for a study on performance portability and programming productivity. The SYCL standard specifies a cross-platform abstraction layer that enables programming of heterogeneous computing systems using standard C++. As

Building the Foundation for Exascale Computing

2 INTEL XEON SCALABLE PROCESSORS
"Sapphire Rapids"

6 X^E ARCHITECTURE BASED GPU'S
"Ponte Vecchio"

ONEAPI
Unified programming model



DELIVERED IN 2021

LEADERSHIP PERFORMANCE
For HPC, data analytics, AI

UNIFIED MEMORY ARCHITECTURE
Across CPU & GPU

ALL-TO-ALL CONNECTIVITY WITHIN NODE
Low latency, high bandwidth

UNPARALLELED I/O SCALABILITY ACROSS NODES
8 fabric endpoints per node, DAOS

U.S. DEPARTMENT OF ENERGY | Argonne NATIONAL LABORATORY | intel | CRAY a Hewlett Packard Enterprise company

SYCL Academy

Learning Materials for SYCL

- An open source project
- Learning materials for high performance programming
- Teaches common C++ features and uses open standards

More information at https://sycl.tech

Try out SYCL

- https://developer.codeplay.com/
- https://support.codeplay.com/
- https://software.intel.com/en-us/oneapi
- SYCL extensions
  - https://github.com/codeplaysoftware/standards-proposals
  - https://github.com/intel/llvm/tree/sycl/sycl/doc/extensions

We're Hiring!

codeplay.com/careers/

codeplay

Enabling AI to be Open, Safe & Accessible to All

@codeplaysoft

info@codeplay.com

codeplay.com