codeplay

Enable AI & HPC to be Open, Safe and Accessible to All

Intro to SYCL and DPC++

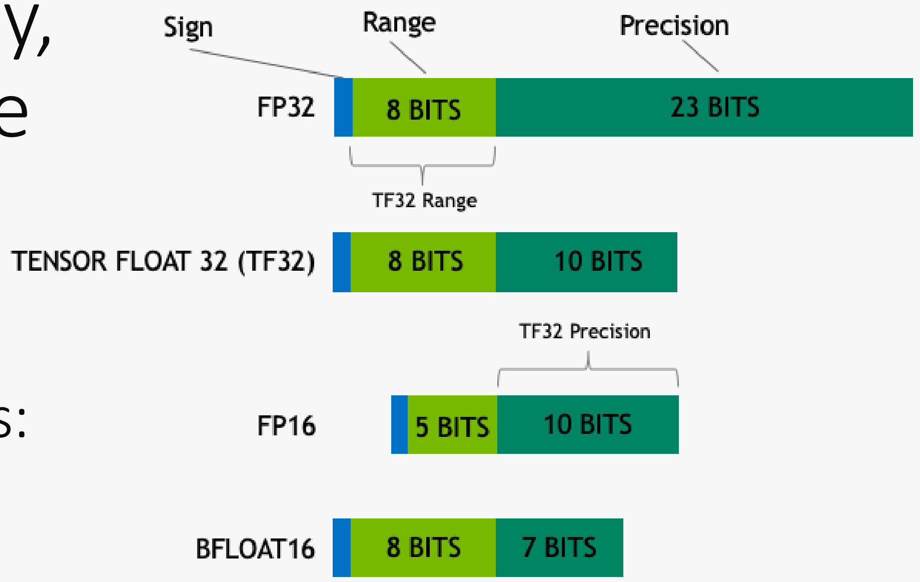
Hugh Delaney – Software Engineer

Intel Bayncore Training – 28th April

- Nvidia is a registered trademark of NVIDIA Corporation
- AMD is a registered trademark of Advanced Micro Devices Corporation
- Intel is a registered trademark of Intel Corporation
- SYCL, SPIR are trademarks of the Khronos Group Inc. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos

Introduction

- I am a software engineer at Codeplay, working on the CUDA backend of the Intel SYCL compiler (DPC++).
- Recent work:
  - Extensions to add support for new datatypes: tensorfloat32, bfloat16 to be used in tensor core mixed precision calculations.
  - Designing the SYCL joint_matrix API to allow for different precisions

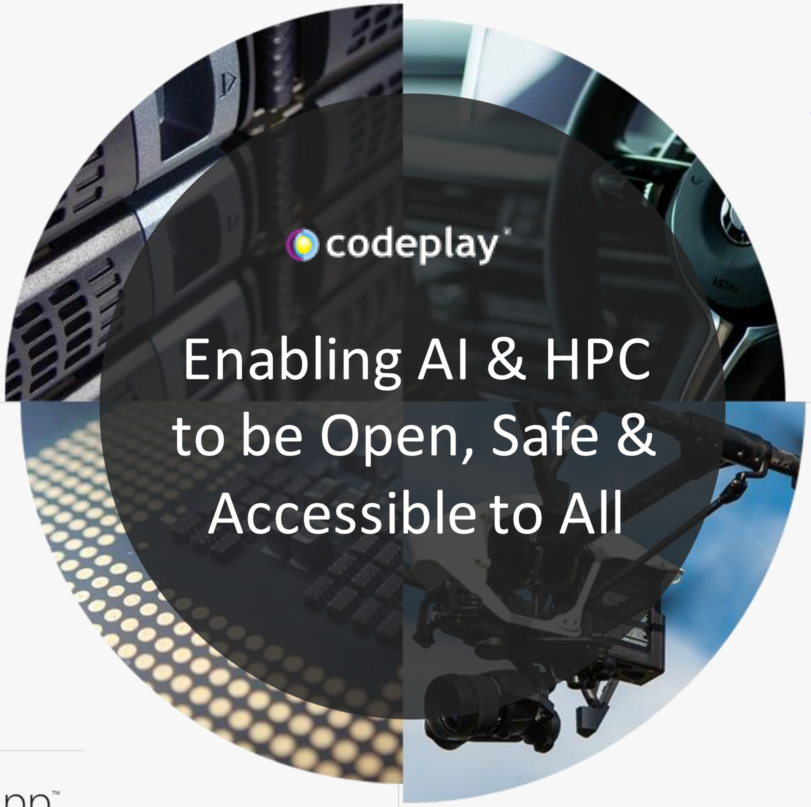


Company

Leaders in enabling high-performance software solutions for new AI processing systems

Enabling the toughest processors with tools and middleware based on open standards

Established 2002 in Scotland with ~80 employees



Partners

And many more!

Products

Acoran

Integrates all the industry standard technologies needed to support a very wide range of AI and HPC

ComputeAorta™

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

ComputeCpp™

C++ platform via the SYCL™ open standard, enabling vision & machine learning e.g. TensorFlow™

Markets

High Performance Compute (HPC)
Automotive ADAS, IoT, Cloud Compute
Smartphones & Tablets
Medical & Industrial

Technologies: Artificial Intelligence
Vision Processing
Machine Learning
Big Data Compute

Learning Objectives

1

Learn why you should use SYCL.

2

Understand why industry wants SYCL.

3

Learn some neat features of DPC++.

SYCL First Example

```
#include <CL/sycl.hpp>
#include <vector>

constexpr size_t N = 10;

#define PRINT_ARRAY(a)                                                              \
  for (auto &e : a)                                                                 \
    std::cout << e << " ";                                                          \
  std::cout << std::endl;

using T = float;

int main() {
  auto A = std::vector<T>(N);
  for (auto i = 0; i < N; i++) {
    A[i] = (T)i;
  }

  std::cout << "Before:\n";
  PRINT_ARRAY(A);
  auto q = sycl::queue{};

  {
    auto buf = sycl::buffer(A);
    q.submit([&](sycl::handler &cgh) {
      auto acc = buf.get_access(cgh);
      cgh.parallel_for(sycl::range<1>(N),
                 [=](sycl::id<1> idx) { acc[idx] *= 2; });
    });
  } // memory is updated once buffers are destructed

  std::cout << "After:\n";
  PRINT_ARRAY(A);
}
```

Kernel Code

What is SYCL?

- High level, standard C++ programming model enabling hardware acceleration.
- Accelerators:
  - GPUs (Nvidia, AMD, Intel)
  - SIMD processors
  - Some FPGAs
  - Other niche hardware

Why use SYCL?

- Code Portability
  - Target GPUs from Nvidia, Intel, AMD
  - Run on CPU, FPGAs, exotic hardware
- Performance Portability
- "But I already know how to use CUDA!"

Why use SYCL?

- Code Portability
  - Target GPUs from Nvidia, Intel, AMD
  - Run on CPU, FPGAs, exotic hardware
- Performance Portability
- ~~"But I already know how to use CUDA!"~~
- You basically already know how to use SYCL!

Motivation

- Hardware becoming more and more diverse, specialized to specific tasks.
  - GPUs specialized to graphics/deep learning.
  - FPGAs specialized to niche tasks.
- The programmer of the future will need to interface with lots and lots of different types of hardware.

Motivation

- However:
  - Interfacing with specialized hardware is hard.
  - Not all hardware vendors can make software like CUDA.

oneAPI

- SYCL enables interface with GPUs, FPGAs, and more through the same programming model.
- Programmer can avoid hardware-specific languages, APIs.
    - Increased productivity.
    - High level SYCL API using existing C++ syntax and language features.

SYCL Increases productivity

```cpp
// CUDA version
__device__ int getGlobalIdx_3D_3D() {
  int blockId =
      blockIdx.x + blockIdx.y * gridDim.x + gridDim.x * gridDim.y * blockIdx.z;
  int threadId = blockId * (blockDim.x * blockDim.y * blockDim.z) +
                 (threadIdx.z * (blockDim.x * blockDim.y)) +
                 (threadIdx.y * blockDim.x) + threadIdx.x;
  return threadId;
}
```

```cpp
// SYCL version
template <size_t N>
inline size_t getGlobalIdx(sycl::item<N> my_item) {
  return my_item.get_global_linear_id();
}
```

- SYCL uses a high-level modern C++ API.
  - Likes templates, lambdas and other modern C++ features

oneAPI

- Code written in SYCL can easily migrate between hardware from different vendors.
- Hardware purchasers not locked-in to long term hardware/software commitments.
    - Especially relevant in age of supply-chain shocks.

SYCL and DPC++ Enables Supercomputers

"this work supports the productivity of scientific application developers and users through performance portability of applications between Aurora and Perlmutter."

Codeplay works in partnership with US National Laboratories to enable SYCL on exascale supercomputers

Enables a broad range of software frameworks and applications

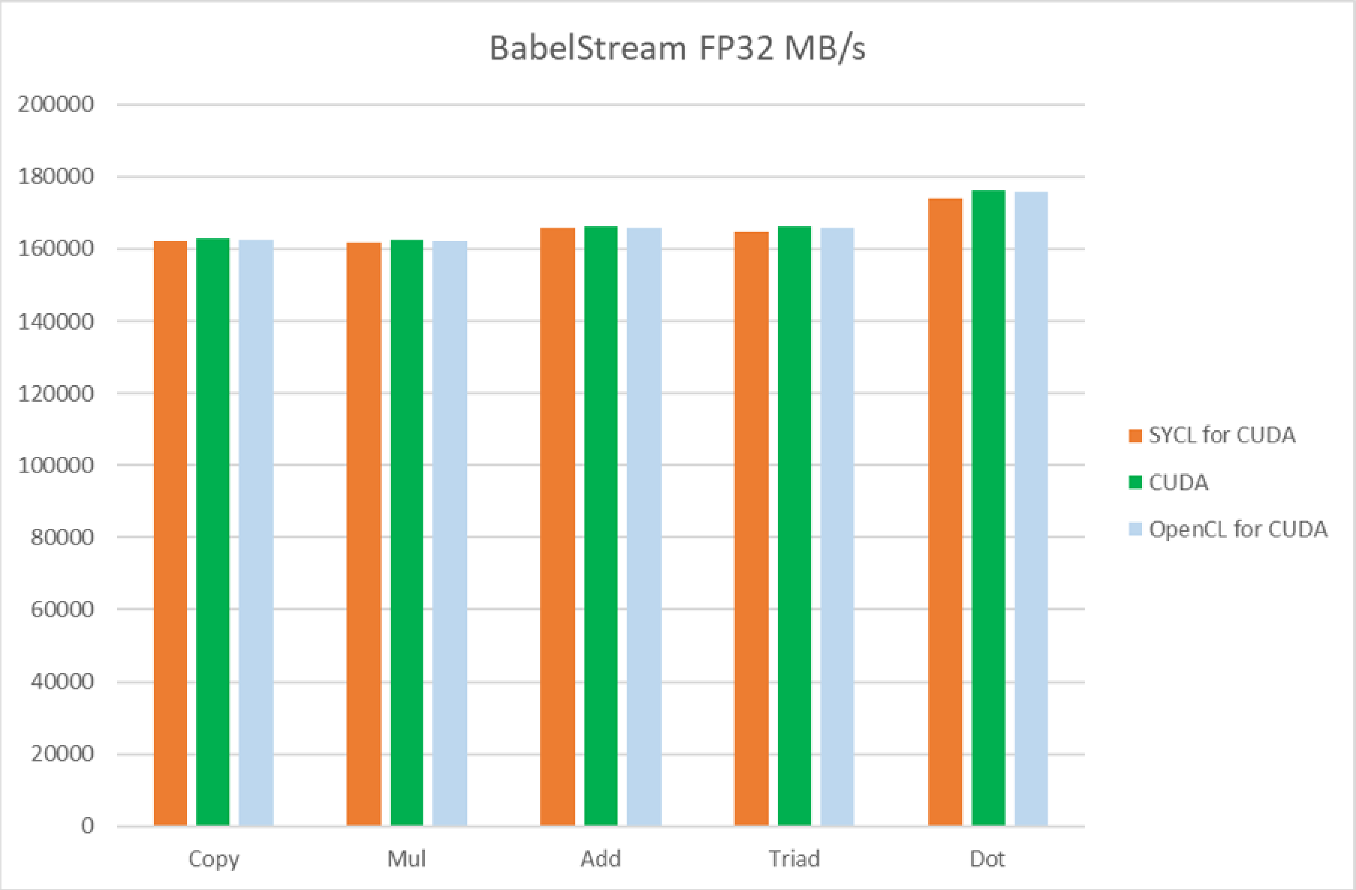
kokkos
RAJA
Celerity
alpaka
BabelStream
LAMMPS
GROMACS
fast, flexible & free

Performance

- But doesn't 'one size fits all' code equate to poor performance?
    - SYCL code written for GPUs will not be performant on FPGAs.
- Code written for GPUs usually gives good performance across vendors.
- We still need to think about the hardware we are targeting.

SYCL performance on Nvidia GPUs

- DPC++ calls the underlying CUDA runtime API/native proprietary API.
- Performance is comparable



http://uob-hpc.github.io/BabelStream

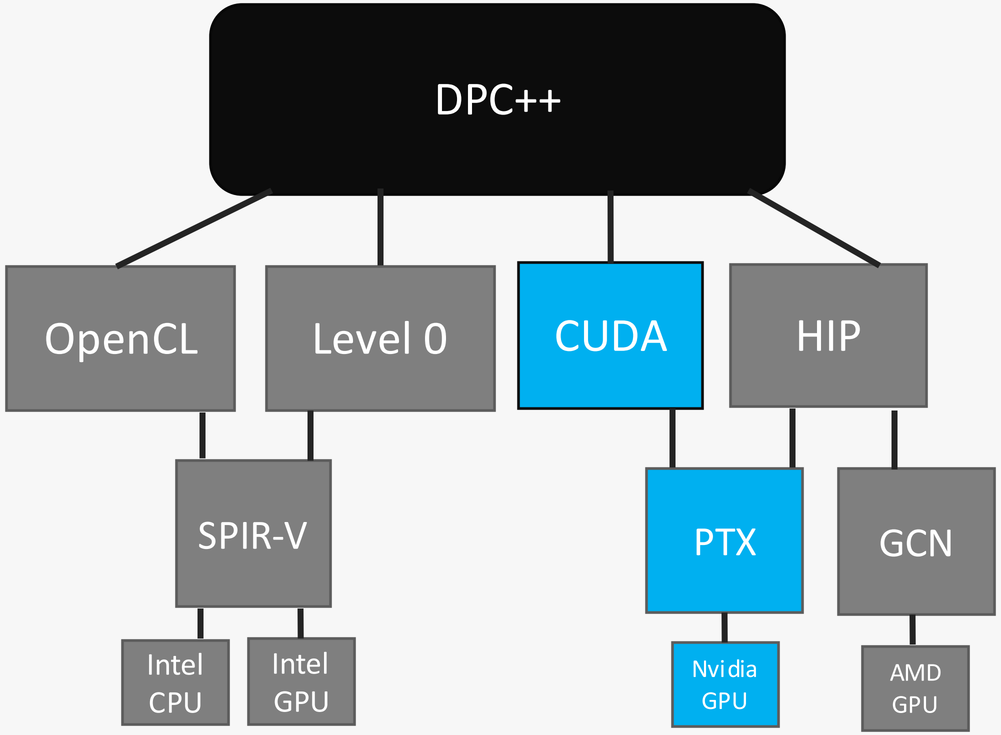
Run on GeForce GTX 980 with CUDA 10.1

What is DPC++

- DPC++ is an Intel-backed open-source implementation of SYCL.
- Built and designed by users.

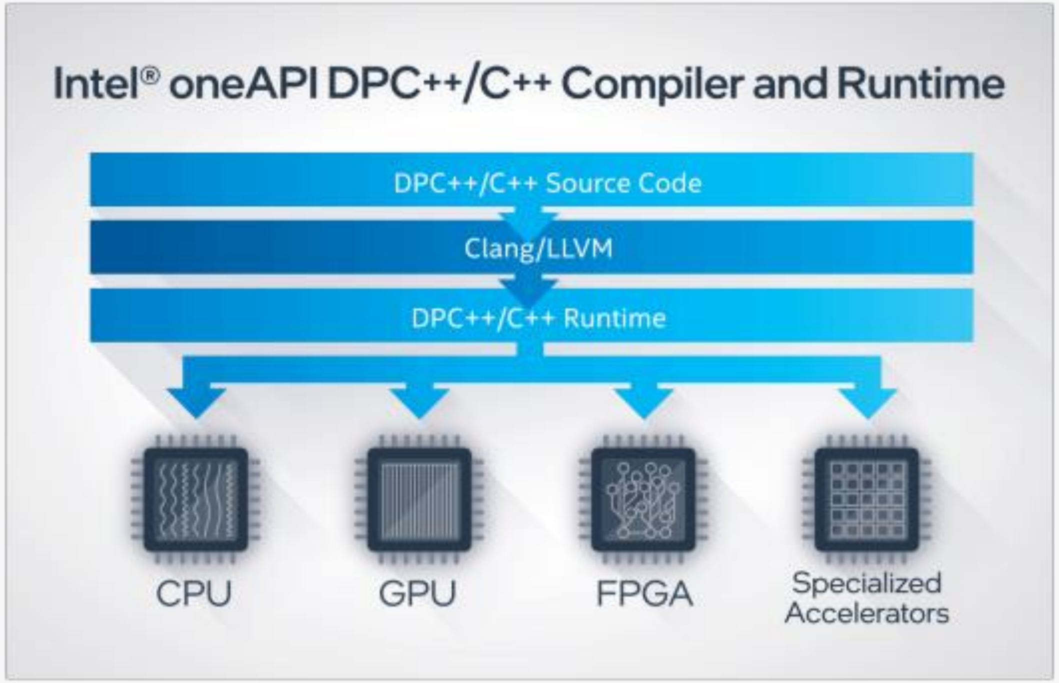
What is DPC++

- DPC++ currently implements 4 GPU backends:
  - OpenCL
  - Level Zero
  - CUDA
  - HIP



What is DPC++

- Includes extensions for specific hardware features (tensor cores etc).
- Supports oneAPI libraries oneMKL, oneDNN.



Porting your brain to SYCL

- Most CUDA concepts map to a SYCL equivalent
- SYCL is an abstraction layer on top of CUDA & others

CUDA	SYCL
CUstream	sycl::queue
CUcontext	sycl::context
CUdevice	sycl::device
CUevent	sycl::event

Where to get started with DPC++

- Build from source:
  - https://github.com/intel/llvm
  - https://intel.github.io/llvm-docs/GetStartedGuide.html
- Intel Devcloud:
  - https://devcloud.intel.com/oneapi/get_started/baseToolkitSamples/

Using DPC++ Compiler

- Check for available devices using sycl-ls

```
➜ cuda01 ~ /opt/llvm/build/bin/sycl-ls
[ext_oneapi_cuda:gpu:0] NVIDIA CUDA BACKEND, NVIDIA GeForce RTX 2080 SUPER 0.0 [CUDA 11.4]
[ext_oneapi_cuda:gpu:1] NVIDIA CUDA BACKEND, Tesla K40c 0.0 [CUDA 11.4]
[host:host:0] SYCL host platform, SYCL host device 1.2 [1.2]
➜ cuda01 ~
```

First Compilation

```
→ cuda01 ~ /opt/llvm/build/bin/clang++ -fsycl -fsycl-targets=nvptx64-nvidia-cuda hello.cc
warning: linking module '/opt/llvm/build/lib/clang/15.0.0/../../clc/remangled-l64-signed_char.libspirv-nvptx64
--nvidiacl.bc': Linking two modules of different target triples: '/opt/llvm/build/lib/clang/15.0.0/../../clc/r
emangled-l64-signed_char.libspirv-nvptx64--nvidiacl.bc' is 'nvptx64-unknown-nvidiacl' whereas 'hello.cc' is 'n
vptx64-nvidia-cuda'
 [-Wlinker-warnings]
1 warning generated.
→ cuda01 ~ LD_LIBRARY_PATH=/opt/llvm/build/lib ./a.out
Running on device: NVIDIA GeForce RTX 2080 SUPER
→ cuda01 ~
```

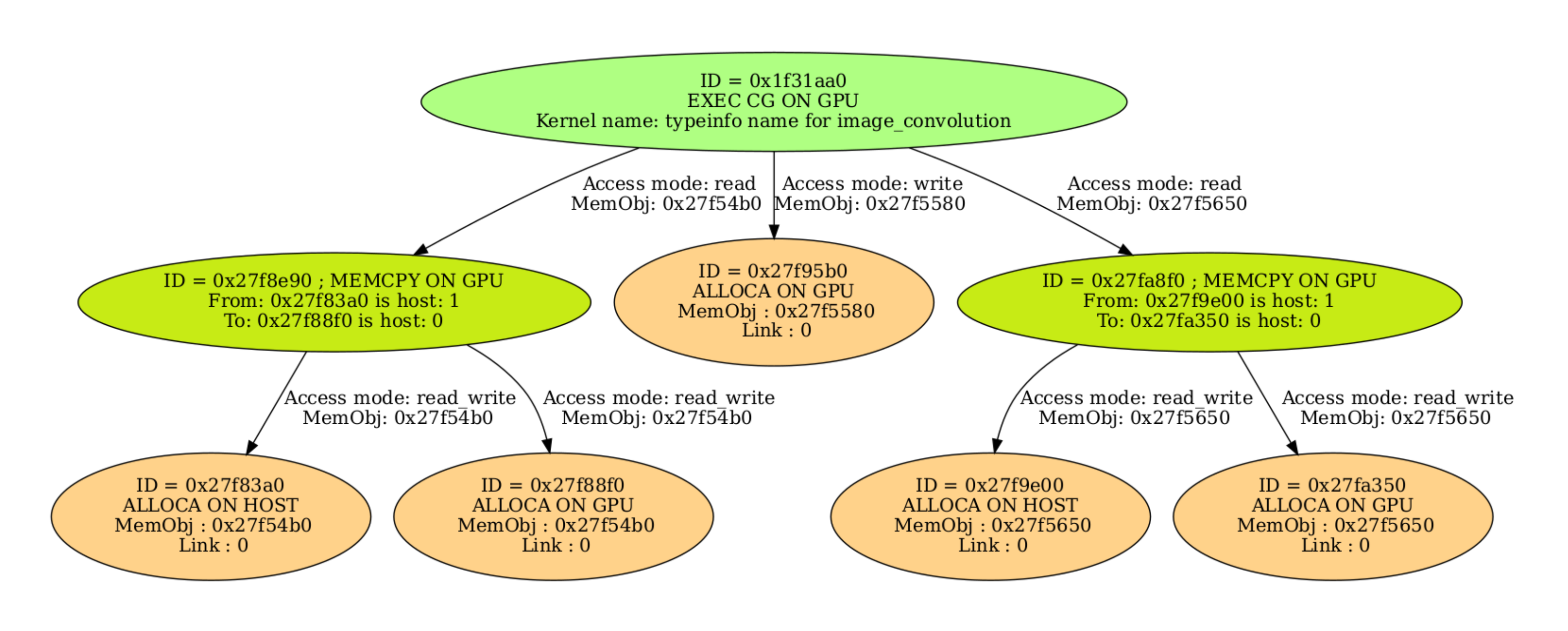
SYCL Device Filter

- Use SYCL_DEVICE_FILTER to specify one of the devices detected by sycl-ls.

```
→ cuda01 ~ SYCL_DEVICE_FILTER=cuda LD_LIBRARY_PATH=/opt/llvm/build/lib ./a.out
Running on device: NVIDIA GeForce RTX 2080 SUPER
→ cuda01 ~ SYCL_DEVICE_FILTER=host LD_LIBRARY_PATH=/opt/llvm/build/lib ./a.out
Running on device: SYCL host device
→ cuda01 ~
```

Neat DPC++ Environment Variables

- SYCL_PI_TRACE
- SYCL_PRINT_EXECUTION_GRAPH



We're Hiring!
codeplay.com/careers/

codeplay

Enable AI & HPC to be Open, Safe and Accessible to All

Questions

@codeplaysoft

info@codeplay.com

codeplay.com