arm

Reaching even richer C++ in OpenCL kernels with use of libclcxx

IWOCL'22

Anastasia Stulova, Ishfaq Wardag

10-12 May 2022

Motivation

- The biggest limitation of C++ for OpenCL kernel language is the absence of the libraries support
  - This prevents accessing many C++ features

- Previous work has focused on reusing libcxx project with successful *<type_traits>* library ported to OpenCL
  - Its use is not straightforward from libcxx project as it is missing OpenCL target
  - It also does not contain OpenCL specific traits

About libclcxx

- This is a new public project hosted on GitHub
  - https://github.com/KhronosGroup/libclcxx
- The main goal is to gather and expose OpenCL-specific C++ libraries
- It integrates libcxx and is made to work with Clang frontend
- Type traits is the initial library feature
- Testing of new functionality is integrated and enabled in CI
- Doxygen documentation is available for the users, similar to the C++ reference pages

<opencl_type_traits>

- Type traits are vital to exploit the power of metaprogramming in C++ fully
- This new type trait library imports all standard C++ type traits from libcxx
- Some specializations of C++ traits have been added for OpenCL specific type system
- Some custom type traits for OpenCL have been added:
  - vectors, address spaces, images, half precision float
- Thanks to reuse of existing type traits implementation from C++, these new traits could be added with minimal development effort

<opencl_type_traits> internals - standard trait specialization

```
// import all type traits from C++
#pragma OPENCL EXTENSION __cl_clang_function_pointers  : enable
#pragma OPENCL EXTENSION __cl_clang_variadic_functions : enable
#include <type_traits>
#pragma OPENCL EXTENSION __cl_clang_function_pointers  : disable
#pragma OPENCL EXTENSION __cl_clang_variadic_functions : disable
namespace std {
/**
* @see <a href="https://en.cppreference.com/w/cpp/types/is_signed">is_signed</a>
*/
template <> struct is_signed<int2> : public true_type {};
/**
* @see <a href="https://en.cppreference.com/w/cpp/types/is_unsigned">is_unsigned</a>
*/
template <> struct is_unsigned<int2> : public false_type {};
```

<opencl_type_traits> internals – new traits

```
/**
* Provides the constant member variable value equal to the
* the number of elements of a vector T if T is a vector and
* equal to 0 otherwise.
* @tparam T - a vector type.
*/
template <typename T> struct vector_size : public
integral_constant<size_t, 0> {};
template <> struct vector_size<int2> : public
integral_constant<size_t, 2> {};
// traits for vector types from int3 to int8 are omitted
template <> struct vector_size<int16> : public
integral_constant<size_t, 16> {};
```

Demonstration – Address space traits

```
1 #include <opencl_type_traits>
2 template<typename PTR_TYPE>
3 auto foo(PTR_TYPE ptr) {
4   using pointee_type = std::remove_pointer<PTR_TYPE>::type;
5   //pointee_type tmp;  /*error: pointee_type qualified with non-private address space*/
6   std::remove_address_space<pointee_type>::type tmp;
7   tmp = *ptr + 1;
8   return tmp;
9 }
10 // foo() works generically for different pointer types and address spaces
11 void bar(global int * glob_int_ptr, local float * loc_float_ptr) {
12   *glob_int_ptr = foo(glob_int_ptr);
13   *loc_float_ptr = foo(loc_float_ptr);
14 }
```

Standard C++ trait

OpenCL-specific trait

Demonstration – Vector type traits

$$r = \sum_{i=0}^{N} x_i$$

```
__kernel void partial_reduction(__global elem_type *input,
                                __global elem_type *output, uint n) {
  auto offset = get_global_id(0);
  elem_type sum = 0;
  // every work item needs to sum up N elements
  for (auto i = offset; i < offset + n; i += vector_size ? vector_size : 1)
    sum += add_elems(&input[i]);
  output[offset] = sum;
}
```

Demonstration – Vector type traits

$$r = \sum_{i=0}^{N} x_i$$

```
__kernel void partial_reduction(__global elem_type *input,
                                __global elem_type *output, uint n) {
  auto offset = get_global_id(0);
  elem_type sum = 0;
  // every work item needs to sum up N elements
  for (auto i = offset; i < offset + n; i += vector_size ? vector_size : 1)
    sum += add_elems(&input[i]);
  output[offset] = sum;
}
```

```
clang -I<path to libclcxx>/include -DTYPE=int   reduction.clcpp
clang -I<path to libclcxx>/include -DTYPE=char4 reduction.clcpp
```

Demonstration – Vector type traits (cont'd)

```
1  #include <opencl_type_traits>
2  constexpr auto vector_size = std::vector_size<TYPE>::value;
3  using elem_type = std::scalar_type<TYPE>::type; // scalar_type obtains vec elem type
4  auto add_elems(elem_type *ptr) {
5   if constexpr (vector_size == 2) {
6     auto vec = vload2(0, ptr);
7     return vec.s0 + vec.s1;
8   } else if constexpr (vector_size == 3) {
9     // analogous handling of vectors with sizes from 3 to 8 are omitted
10  } else if constexpr (vector_size == 16) {
11    auto vec = vload16(0, ptr);
12    return vec.s0 + /* omitted other components from s1 to s14 */ vec.s15;
13  }
14  return *ptr; // not a vector!
15 }
```

Conclusions and feedback

- A new space for C++ libraries for OpenCL has been created
- First library (OpenCL type traits) is derived from the C++ implementation allowing reducing the development cycles and providing best synergy
- Developers are invited for experimenting and contributing
- Any feedback is greatly appreciated and can be submitted using libclcxx GitHub project page

arm

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה