Enabling the Use of C++20 Unseq Execution Policy for OpenCL

Po-Yao Chang, Tai-Liang Chen, and Jenq-Kuen Lee
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
{pychang, tlchen}@pllab.cs.nthu.edu.tw, jklee@cs.nthu.edu.tw

Motivation

- C++ for OpenCL was announced in 2020, but without the support of the standard library as stated in the C++ standard.
- We explore the use of execution policy as in the C++ parallel library (focused on execution::unseq from C++20).
- Inspired by OpenCL vector, this paper supports C++ template of execution::unseq based on OpenCL vector.

Procedural Steps

Step 1 Define unseq object
This step defines the types as follows and a global object unseq of type unsequenced_policy accordingly.

```cpp
struct unsequenced_policy {};
struct sequenced_policy {};
constexpr sequenced_policy unseq();
```

Step 2 OpenCL kernel with execution policy
Overload functions with execution policy types.

```cpp
#define __kernel void add(...)

__kernel void add(...)
{
    auto v = a[idx] + b[idx];
}
```

Step 3 Using directive to vector
- Clang would then inline the function object call operator as in f(*first) and vectorize the loop with clang directive.
- The resulting LLVM bitcode would contain LLVM vector types.
- OpenCL vector types also get lowered to LLVM vector type.

Unseq with OpenCL Vector

- This for_each call may be vectorized.

```cpp
for (auto v = a[idx] + b[idx];
```

OpenCL vector are mapped to LLVM vector in LLVM IR layer.

Experimental Results

Experiment Environments

- Platform:
  - OpenCL 2.1
  - Clang 10.0.1
  - Spirv compiler: llvm-spirv (built against LLVM 10.0.1)

OpenCL devices:
- Intel(R) CPU Runtime for OpenCL(TM) Applications/Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
- Intel(R) OpenCL HD Graphics/Intel(R) Gen9 HD Graphics NEO

- In the case of SAD on GPU, vector width 4 results in a speedup of 3.4, and vector width 16 results in 6.9X speedup