C++OpenCL4TVM: Support C++OpenCL Kernel for TVM NN Operators

Po-Yao Chang, Tai-Liang Chen, Yu-Tse Huang, Meng-Shiun Yu, and Jenq-Kuen Lee

Department of Computer Science,
National Tsing Hua University, Hsinchu, Taiwan
{pychang, tlchen, ythuang, msyu}@pllab.cs.nthu.edu.tw, jklee@cs.nthu.edu.tw
Outline

• Motivation - Parallelism TS merged to C++17 and C++20

• Objective

• Rewriting in TVM codegen and OpenCL modules accommodating C++17/20 construct

• Experiment

• Ongoing work

• Summary
Motivation - Parallelism TS merged to C++17 and C++20

• A detour: C++ release model
  • ISO C++ “train model”: standardize what’s fully-baked, iterate on immature features.
  • Very similar to how LLVM operates: development all happens on the main branch. A “release branch” is created twice a year. After some testing and cherry-picking then ship the release.
• A time-based instead of feature-based release model

• C++17 、 C++20 parallel algorithms
  • Parallelism TS: proposed by Nvidia folks; brings parallel algorithms to C++
  • Incorporate into TVM generated kernel code
Objective: TVM Introduction

Computational graph, e.g., ONNX

Optimization

TIR

Codegen

input

1x7x224x224

Conv

W (64x7x3x3)

1x64x222x222

output
Objective: add C++ construct into TVM-generated code

```c
__kernel void permute_kernel0(__global float* __restrict TTranspose, __global float* __restrict A, int n, int stride, int stride1, int stride2, int stride3) {
    for (int ax1 = 0; ax1 < n; ++ax1) {
        TTranspose[((int)get_group_id(0) * stride2) + (ax1 * stride3))] = A[((ax1 * stride) + (((int)get_group_id(0) * stride1))]);
    }
}
```

```c
#include "uoc1_algorithm"
#include "helper_iterator.hpp"

__kernel void permute_kernel0(__global float* __restrict TTranspose, __global float* __restrict A, int n, int stride, int stride1, int stride2, int stride3) {
    std::for_each_n(std::execution::unseq, CountFromZero{}, n, [&] (int ax1) {
        TTranspose[((int)get_group_id(0) * stride2) + (ax1 * stride3))] = A[((ax1 * stride) + (((int)get_group_id(0) * stride1))]);
    });
}
```
CountFromZero: a helper iterator providing ++ and *

class CountFromZero {
    int counter = 0;

public:
    CountFromZero& operator++() {
        ++counter;
        return *this;
    }

    int& operator*() { return counter; }
};
Rewriting in TVM Codegen modules

Inside CodeGenC::VisitStmt_(const ForNode* op)

```cpp
if (op->body->type_index() == 169) {
    // __builtin_debugtrap();
    stream << "std::for_each_n(std::execution::unseq, CountFromZero{}, " << extent << ", [&](int \\
        << vid << ") {\n";
    int for_scope = BeginScope();
    PrintStmt(op->body);
    this->EndScope(for_scope);
    PrintIndent();
    stream << "});\n";
} else {
```
Revision for SPIR-V (OpenCL::InstallKernel)

```cpp
{
    std::ofstream clcpp{DIRPREFIX FILENAME "\.

    auto it = std::ostream_iterator<char>{clcpp};

    const char includes[] = R"(#include "uoocl_algorithm""
        "\n"
        R"(#include "helper_iterator.hpp"
        "\n"
        std::copy(std::begin(includes), std::prev(std::end(includes)), it);
    std::copy(data_.begin(), data_.end(), it);

    std::system(TOOLSREFIX "clang -c -cl-std=CLC++ -target spir64 -emit-llvm -I" IPATH
        " DIRPREFIX FILENAME ".clcpp & & " TOOLSREFIX
        "llvm-spirv " DIRPREFIX FILENAME ".bc"));

    std::ifstream clcpp{DIRPREFIX FILENAME ".spv"};
    const std::vector<char> spv(std::istreambuf_iterator<char>{clcpp}, {});
    program_ = clCreateProgramWithIL(w->context, spv.data(), spv.size(), &err);
```
Experiment Environment

• Compiler for kernel source file: Clang 13.0.1

• SPIR-V translator: patched llvm-spirv 13.0.1

• OpenCL Device: Intel(R) HD Graphics 630/OpenCL 3.0 NEO

• TVM [https://github.com/apache/tvm/commit/701d2c32759c95](https://github.com/apache/tvm/commit/701d2c32759c95)
IWOCL 2022 - The 10th International Workshop on OpenCL

llvm-spirv workaround

• Official llvm-spirv can’t handle freeze

• Workaround it as intel does: https://github.com/intel/llvm/blob/b2d4d67d5e34/llvm-spirv/lib/SPIRV/SPIRVRegularizeLLVM.cpp#L626-L631
## Speedup after vectorization (unseq)

<table>
<thead>
<tr>
<th>Function</th>
<th>Base version (not specify unseq)</th>
<th>Vectorized (specify unseq)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>convolution</td>
<td>21954</td>
<td>20540 us</td>
<td>6%</td>
</tr>
<tr>
<td>topi.transpose</td>
<td>3001 us</td>
<td>2160 us</td>
<td>38%</td>
</tr>
<tr>
<td>topi.matmul</td>
<td>1204 us</td>
<td>1112 us</td>
<td>8%</td>
</tr>
</tbody>
</table>
Ongoing work: layout/view

- Provide layout and view for sparse abstractions for TVM with OpenCL C++

```cpp
View<Sparse_CSR> v{row_indices, col_indices, elements};
foo::bar(
    arg_indicating_execution_policy_or_matrix_sparsity,
    args...);
```

Listing 2: Code fragment TVM-generated C++ for OpenCL code
Encapsulate into a sparse\_dense template specialization

```c
kernel void fused nn sparse dense kernel0(_global float* __restrict compute, _global int* __restrict placeholder, _global float* __restrict placeholder1, _global float* __restrict placeholder2)
    if (((int)get_group_id(0) * 64) + ((int)get_local_id(0)) < 1000)
        compute[(((int)get_group_id(0)) * 64) + ((int)get_local_id(0))]] = 0.00000e+00f;
    
    for (int elem_idx = 0; elem_idx < (placeholder[(((int)get_group_id(0)) * 64) + ((int)get_local_id(0)) % 1000 + 1)] - placeholder[(((int)get_group_id(0)) * 64) + ((int)get_local_id(0)) < 1000])
        compute[(((int)get_group_id(0)) * 64) + ((int)get_local_id(0))]] = (compute[(((int)get_group_id(0)) * 64) + ((int)get_local_id(0))]]) + (placeholder1[(((
```
Summary

• We C++-ify TVM-generated OpenCL kernels

• Building upon our work from last year which is bringing C++ unsequenced execution policy to OpenCL kernel, we put “unseq” into TVM-generated code.

• Identify a “StoreNode” in a loop body and transform it.

• Neither does performance improve significantly, nor does it degrade.

• Ongoing work to provide layout and view abstraction to help facilitate sparse computations for TVM

• Investigating TVM backends with more Khronos APIs is of interests.