

# 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}@pplab.cs.nthu.edu.tw, jklee@cs.nthu.edu.tw



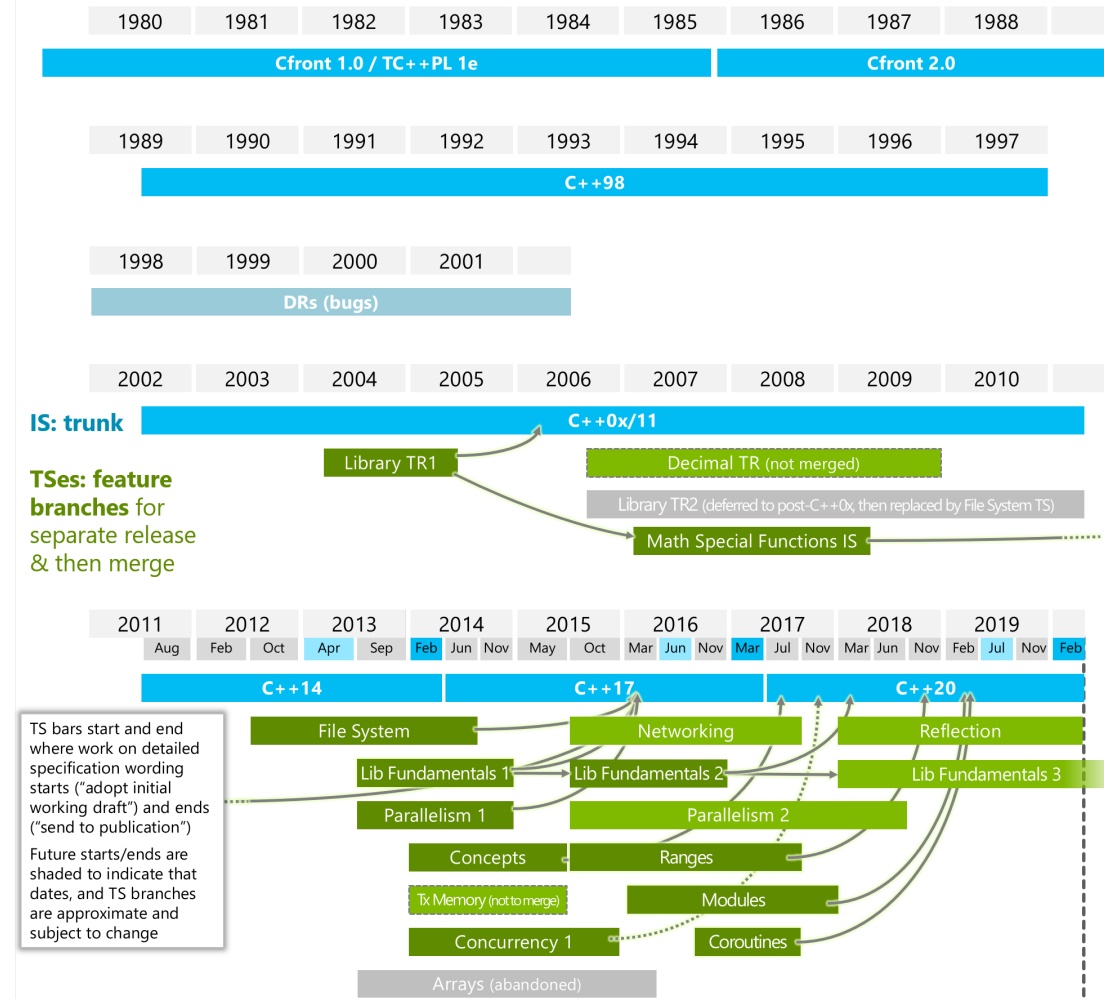
# Outline

---

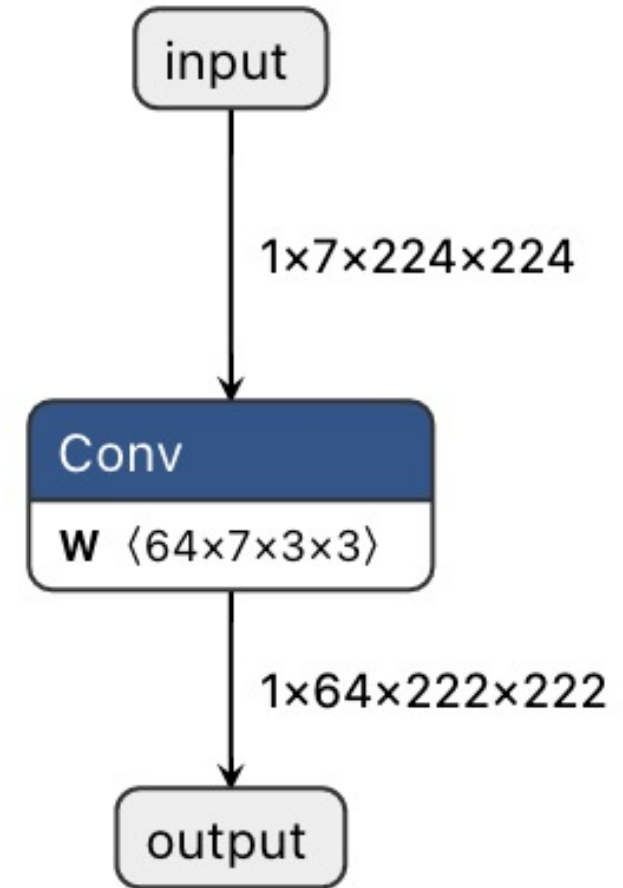
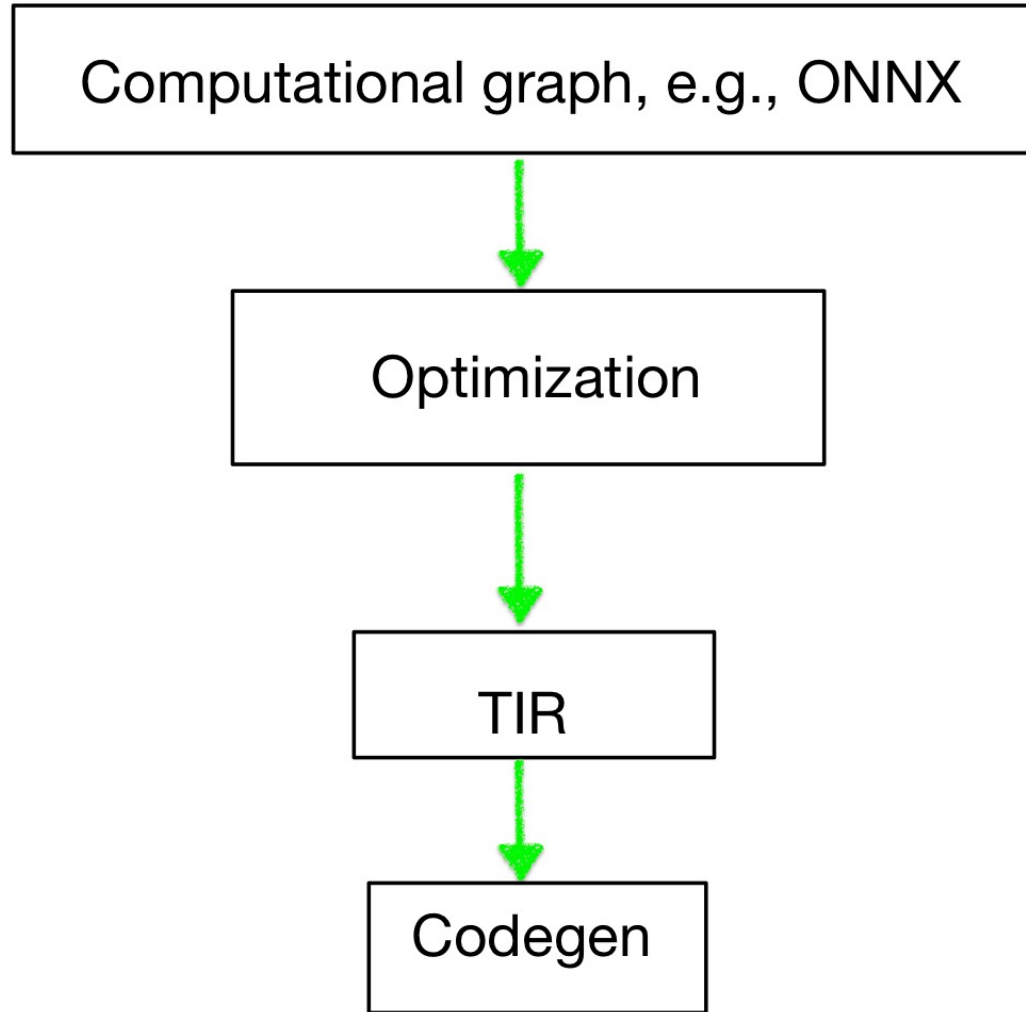
- Motivation - Parallelism TS merged to C++17 and C++20
- Objective
- Rewriting in TVM codegen and OpenCL modules accommodating C++17/20 construct
- Experiment
- Ongoing work
- Summary

# 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

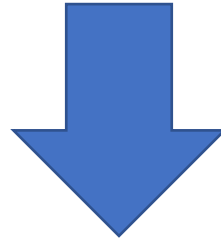


# Objective: TVM Introduction



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

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



```
#include "uocl_algorithm"  
#include "helper_iterator.hpp"  
  
__kernel void permute_kernel0(__global float* __restrict T_transpose, __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) {  
        T_transpose[(((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)

```
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 (OpenCLModuleNode::InstallKernel)

```
{
    std::ofstream clcpp{DIRPREFIX FILENAME ".clcpp"};
    auto it = std::ostream_iterator<char>{clcpp};
    constexpr char includes[] = R"(#include "uocl_algorithm")"
                                "\n"
                                R"(#include "helper_iterator.hpp")"
                                "\n\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>

# 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>

```
~/iwocl/iwocl_2022 master !3 ?19  
> llvm-spirv unseq.bc  
InvalidInstruction: Can't translate llvm instruction:  
%54 = freeze i32 %20  
~/iwocl/iwocl_2022 master !3 ?19
```

```
338 // Remove optimization info not supported by SPIRV  
339 if (auto BO = dyn_cast<BinaryOperator>(&II)) {  
340     if (isa<PossiblyExactOperator>(BO) && BO->isExact())  
341         BO->setIsExact(false);  
342 }  
343  
344 // FIXME: This is not valid handling for freeze instruction  
345 if (auto *FI = dyn_cast<FreezeInst>(&II)) {  
346     FI->replaceAllUsesWith(FI->getOperand(0));  
347     FI->dropAllReferences();  
348     ToErase.push_back(FI);  
349 }  
350  
351 // Remove metadata not supported by SPIRV  
352 static const char *MDs[] = {  
353     "fpmath",  
354     "tbaa",  
355     "range",  
356 };
```

# Speedup after vectorization (unseq)

|                | Base version (not specify unseq) | Vectorized (specify unseq) | Speedup |
|----------------|----------------------------------|----------------------------|---------|
| convolution    | 21954                            | 20540 us                   | 6%      |
| topi.transpose | 3001 us                          | 2160 us                    | 38%     |
| topi.matmul    | 1204 us                          | 1112 us                    | 8%      |

# Ongoing work: layout/view

---

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

---

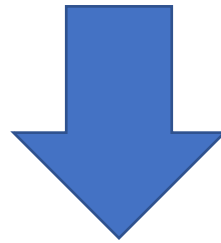
```
1 View<Sparse_CSR> v{row_indices, col_indices, elements};  
2  
3 foo::bar(  
4     arg Indicating_execution_policy_or_matrix_sparsity,  
5     args...);
```

---

**Listing 2: Code fragment TVM-generated C++ for OpenCL code**

# Encapsulate into a sparse\_dense template specialization

```
kernel void fused_nn_sparse_dense_kernel0(__global float* __restrict compute, __global int* __restrict placeholder, __global float* __restrict placeholder1, __gl
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))] = 0.000000e+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))
    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))] = (compute[((((((int)get_group_id(0)) * 64) + ((int)get_local_id(0)))] + (placeholder1[(((
    }
}
}
```



```
kernel void default_function_kernel0(__global float* __restrict compute, __global int* __restrict placeholder, __global float* __restrict placeholder1, __global float* __restr
csr_format csr_data(placeholder1, placeholder2, placeholder3, placeholder);
View<csr_format, int> sprase_kernel (csr_data, 17);
sprase_kernel.do_sparse_dense_group(compute, 64, 1000);
}
```

# 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 degrades.
- 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.

