Yakup Koray Budanaz

SoftHier September 2

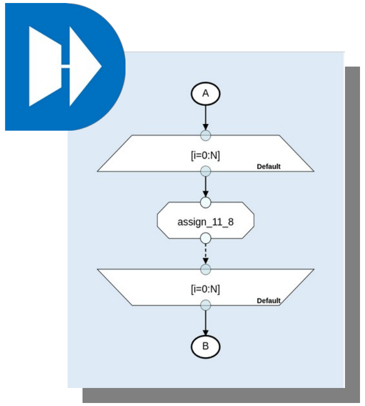
Overiew of the Topics:

- DaCe + ICON Gordon Bell Submission
- Current Work & And Future Student Projects

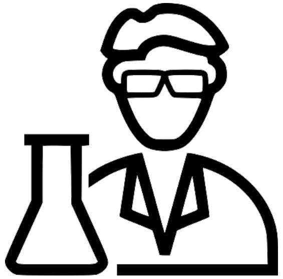
Click to add Title

- ICON and Gordon Bell Submission

The DaCe Story:

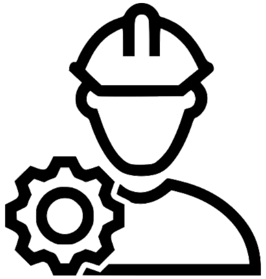


Lifting Source Code to a graph-based dataflow representation

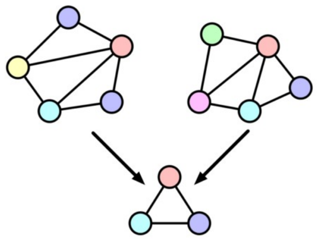


Domain Scientist

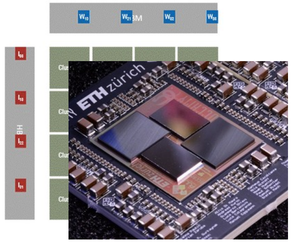
NumPy

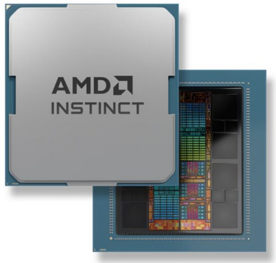


Performance Engineer



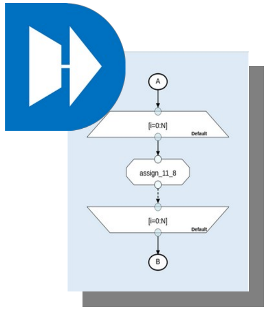
Reusable Transformations





More Accelerators...

ICON + DaCe Story = Gordon Bell Run

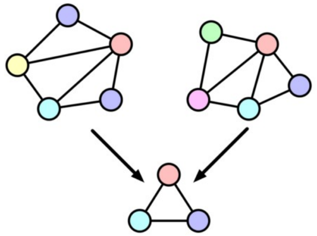


GH200

Domain Scientist

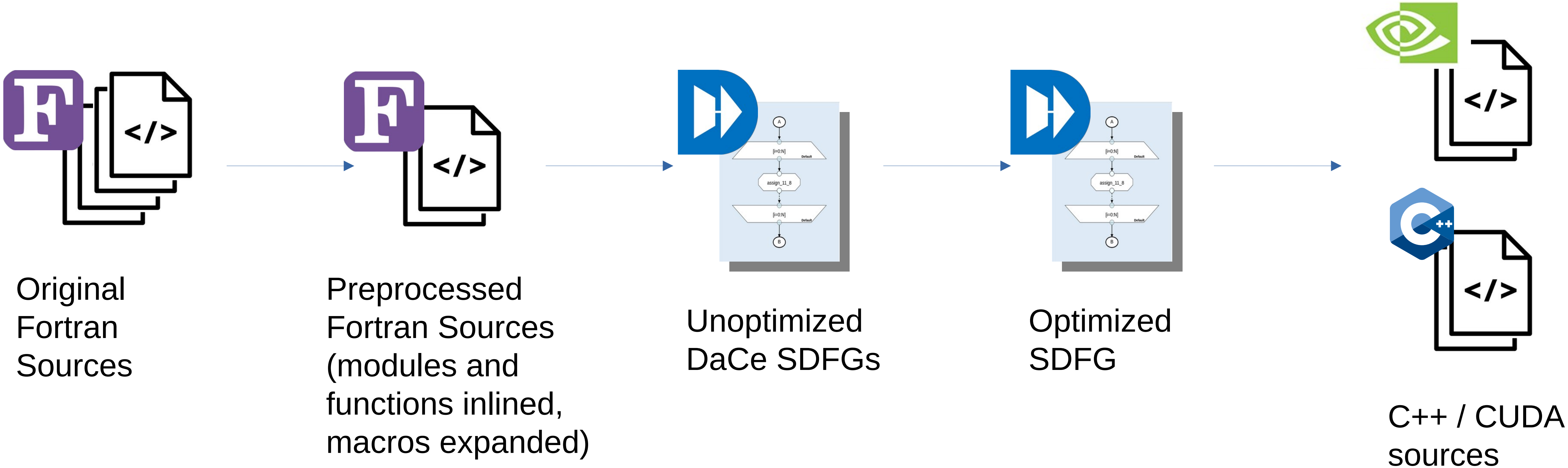
Performance Engineer

ICON sources
and configuration



Single-use transformations
due to the time-frame and
scale of the run

Decoupling Hardware-Specific Optimizations From Source Code



Original Fortran Sources

Preprocessed Fortran Sources (modules and functions inlined, macros expanded)

Unoptimized DaCe SDFGs

Optimized SDFG

C++ / CUDA sources

Note: manual fixes have been omitted (there are many of them)

Click to add Title

Step 1: Propagate Configuration-specific constants

Step 2: Simplify the structure of the SDFG to be more compatible with existing transformations

Step 3: Obtain the maximally parallel SDFG and improve data layouts

Note: manual fixes have been omitted (there are many of them)

Click to add Title

Step 1: Propagate Configuration-specific constants

- **Propagate known constants**
- **Propagate value sets (e.g. x={1,2} then we create two specialized calls)**

Step 2: Simplify the structure of the SDFG to be more compatible with existing transformations

- **Flatten Structs (AoS) to SoA layout**
- **Eliminate trivial control flow**
- **Move if-conditions within / outside maps**

Step 3: Obtain the maximally parallel SDFG and improve data layouts

- **LoopToMap + MapFusion**
- **Move if-conditions outside maps (again)**
- **Change layouts**

Note: manual fixes have been omitted (there are many of them)

Click to add Title

Step 4: Specialize for GPUs

Step 5: Further architecture and configuration specific optimizations

Step 6: Generate bindings and integrate SDFG into the ICON / Fortran modules

Note: manual fixes have been omitted (there are many of them)

Click to add Title

Step 4: Specialize for GPUs

- **Offload to GPU**
- **Assign Streams**
- **Memcpy / Memset kernel to API calls**

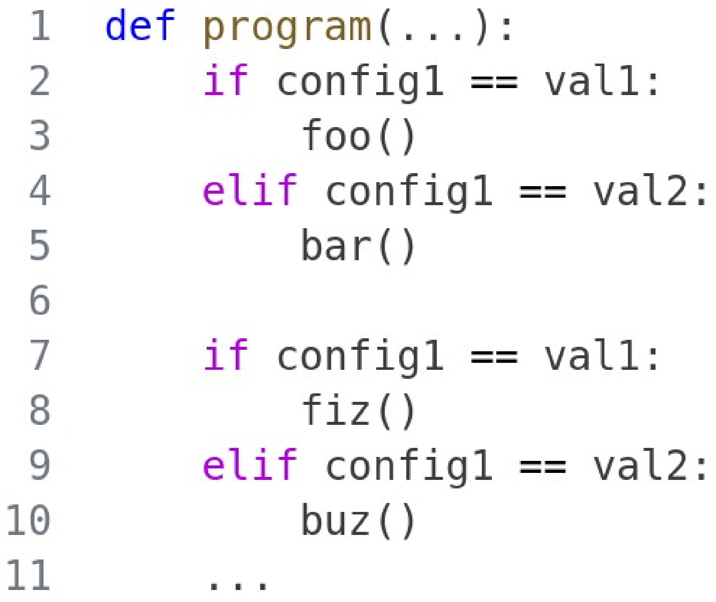
Step 5: Further architecture and configuration specific optimizations

- **Lower Bitwidth of Neighbor Lists**
- **Replace single-value array with Constants**

Step 6: Generate bindings and integrate SDFG into the ICON / Fortran modules

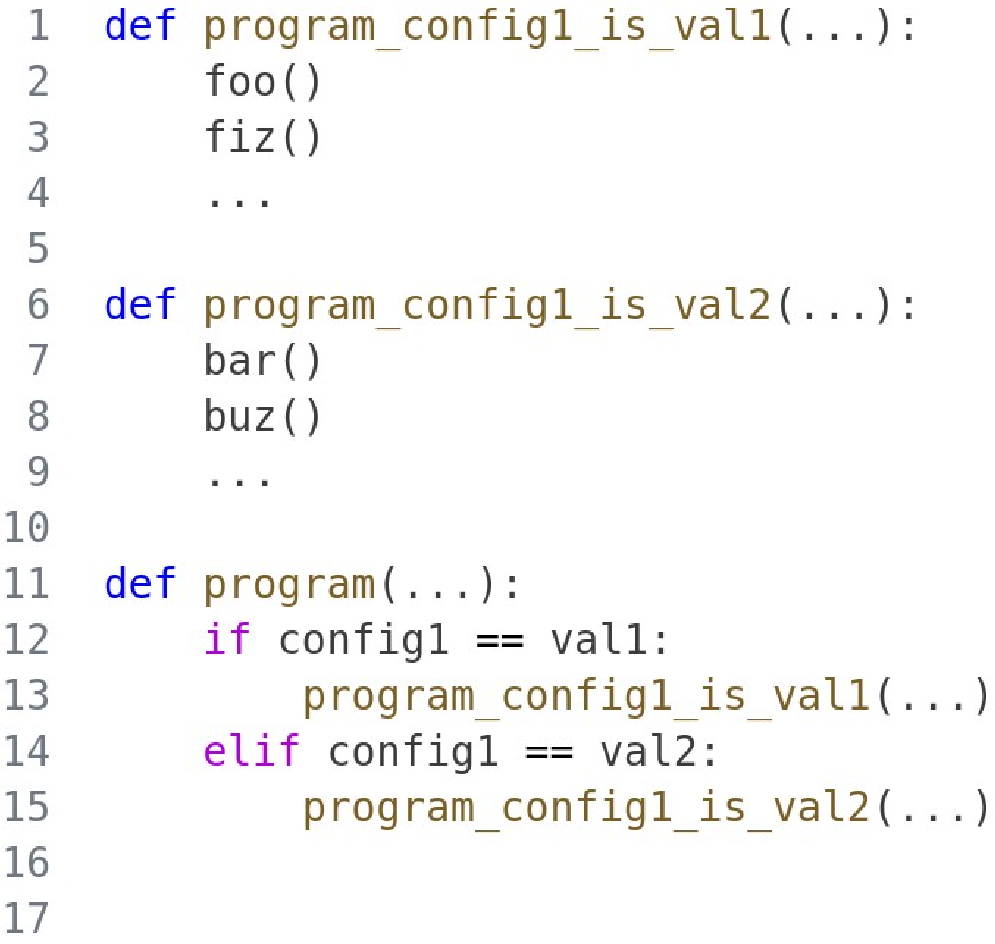
- **Performed in multiple scripts**
- **(Mainly manual)**

Step 1: Propagate Configuration-specific constants



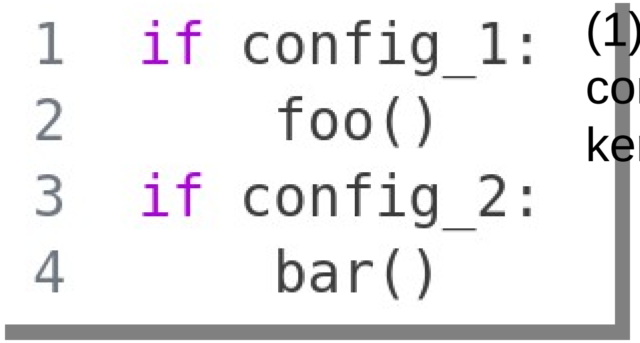
```python
def program(...):
    if config1 == val1:
        foo()
    elif config1 == val2:
        bar()

    if config1 == val1:
        fiz()
    elif config1 == val2:
        buz()
    ...
```

Exposes more opportunities to fuse kernels and decreases the chance of encountering bugs by simplifying the structure of the SDFG



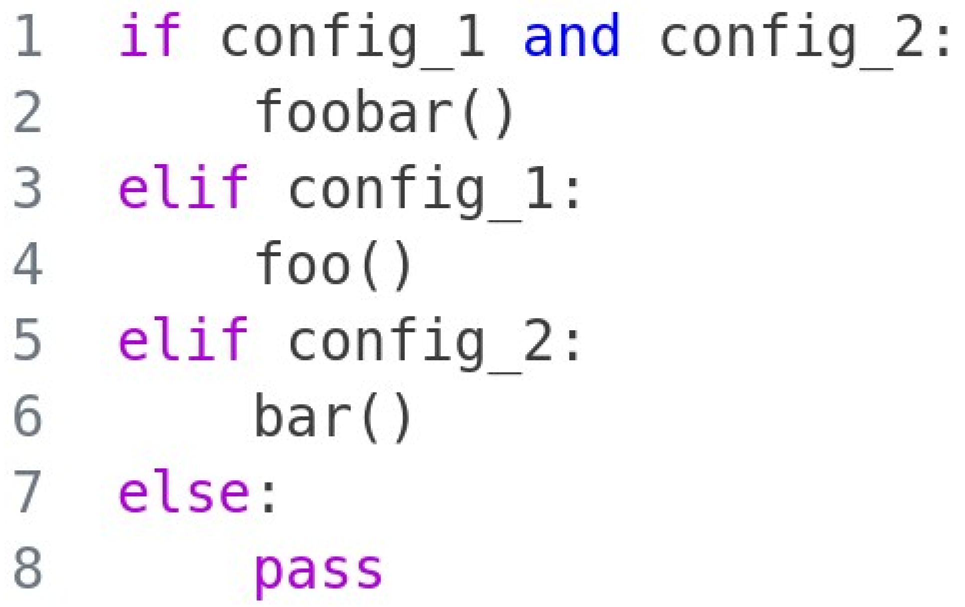
```python
def program_config1_is_val1(...):
    foo()
    fiz()
    ...

def program_config1_is_val2(...):
    bar()
    buz()
    ...

def program(...):
    if config1 == val1:
        program_config1_is_val1(...)
    elif config1 == val2:
        program_config1_is_val2(...)


```

Step 1: Propagate Configuration-specific constants & Step 2: Map Fusion



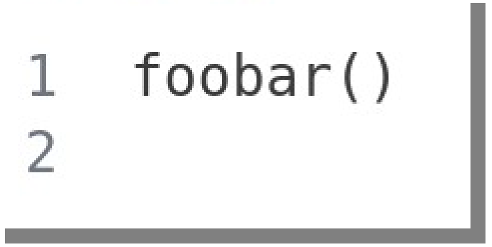
```
if config_1:
    foo()
if config_2:
    bar()
```

(1) Propagate configurations and fuse kernels



```
if config_1 and config_2:
    foobar()
elif config_1:
    foo()
elif config_2:
    bar()
else:
    pass
```

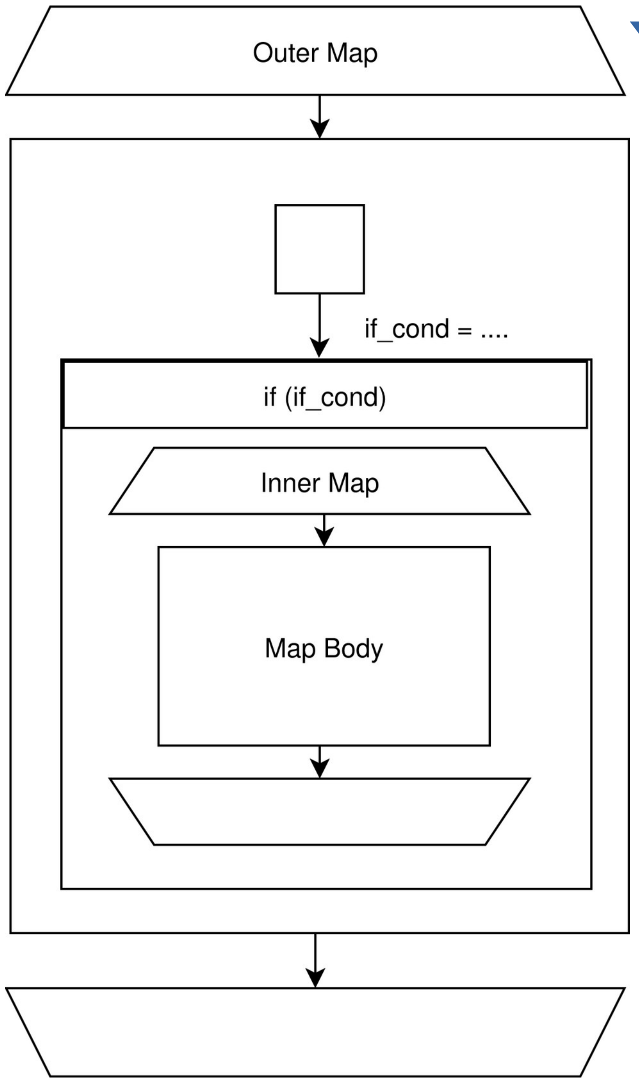
(2) Filter trivially false configurations



```
foobar()

```

*Due to the time-frame we first propagated / filtered the known configuration and then fused kernels.

Step 2: Simplify the structure of the SDFG to be more compatible with existing transformations

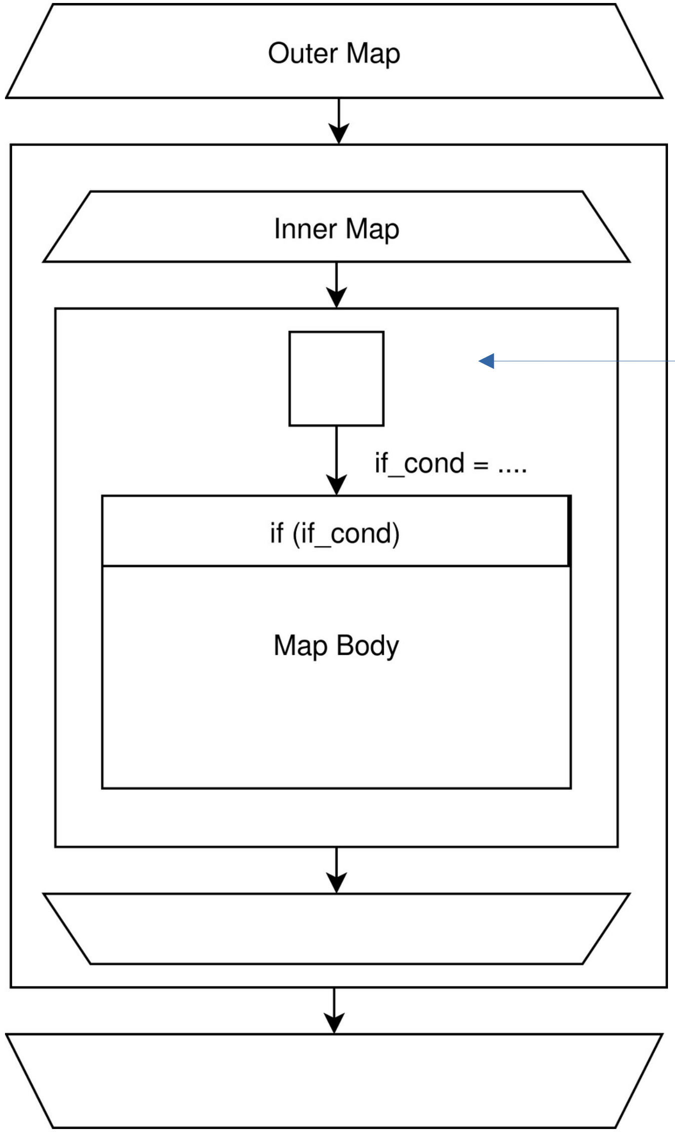


Outer Map can be offloaded to the accelerators

Inner computation depends on a condition. Inner body has to be a sequential loop executed by a thread (only option due to how DaCe handles offloading).

Step 2: Simplify the structure of the SDFG to be more compatible with existing transformations

The outer and the inner map can be collapsed together now



Every thread duplicates the computation of the if condition

Step 2: Simplify the structure of the SDFG to be more compatible with existing transformations

- We identified the working combinations for the movement of ifs through manual inspections and the performance obtained through profiling runs

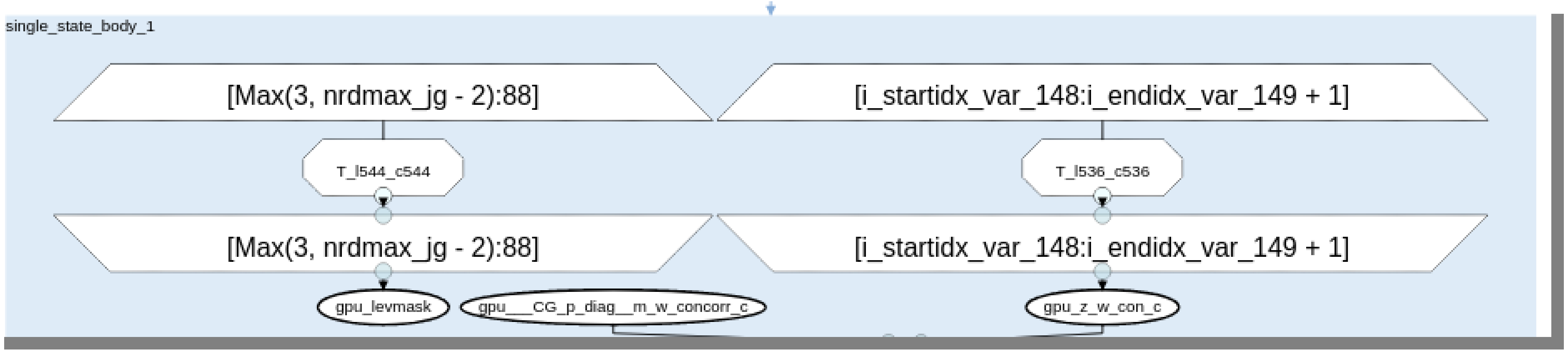
Step 3: Obtain the maximally parallel SDFG and improve data layouts

- Identifying parallelism automatically and manually to expose as much as parallel regions as possible

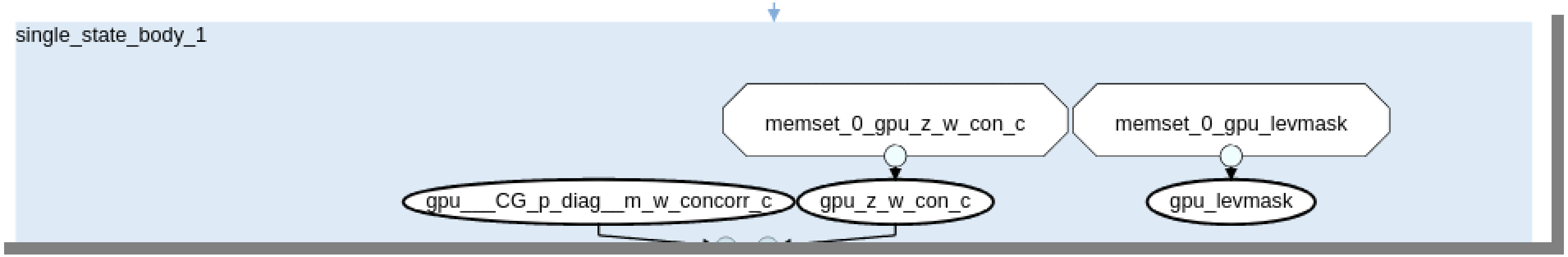
Step 4: Specialize for GPUs

- DaCe’s GPU Offloading transformation was not capable of offloading ICON SDFGs. I developed an offloading implementation optimized for the assumption that all computation resides on the GPU. (Makes the development of the transformation much simpler)
- I have also designed an improved approach that supports cases where part of the computation remains on the CPU.
  - Based on this, I created a project and am looking for students to implement it.

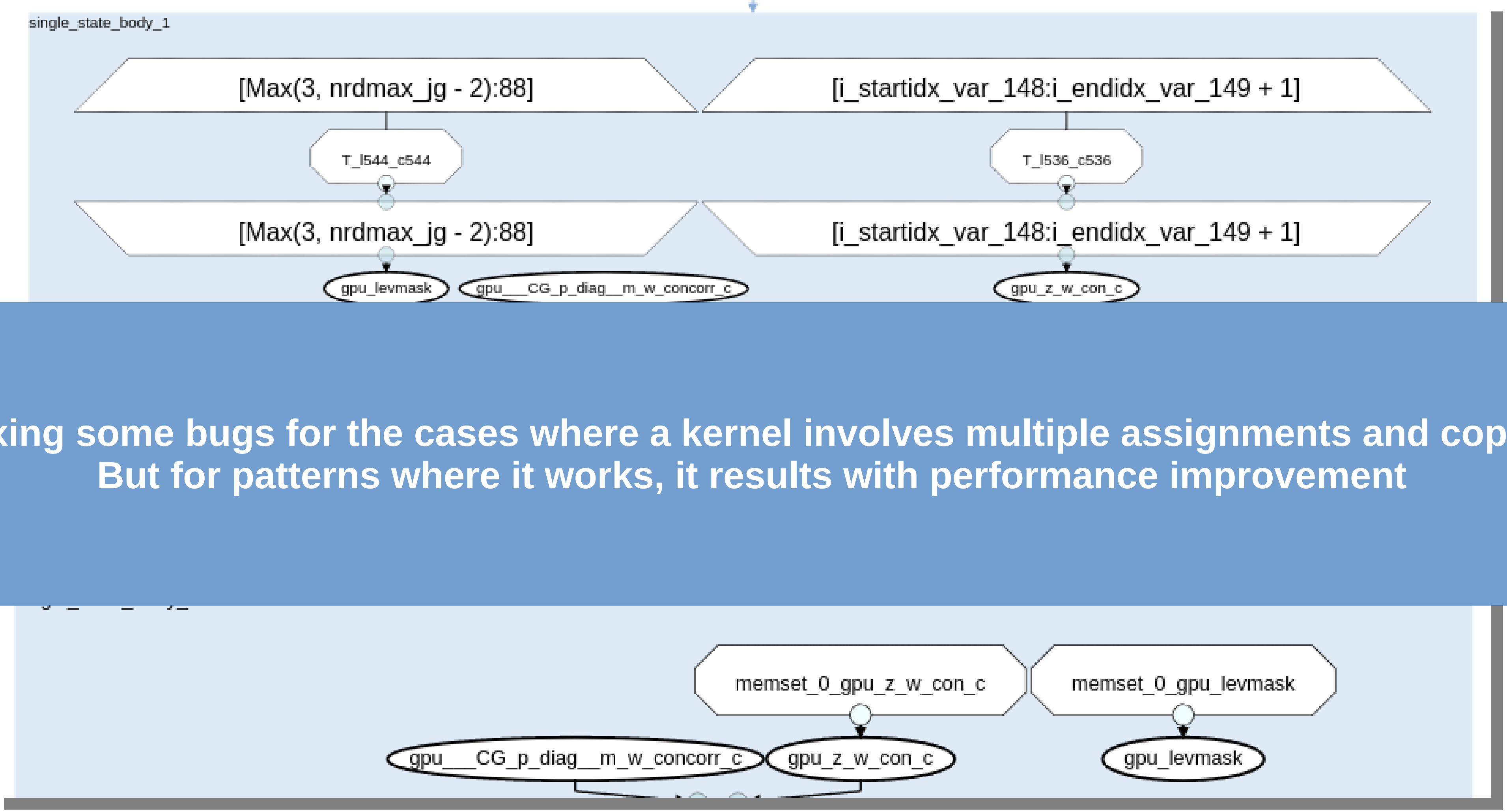
Step 4: Specialize for GPUs


single_state_body_1
[Max(3, nrdmax_jg - 2):88]
[i_startidx_var_148:i_endidx_var_149 + 1]
T_l544_c544
T_l536_c536
[Max(3, nrdmax_jg - 2):88]
[i_startidx_var_148:i_endidx_var_149 + 1]
gpu_levmask
gpu___CG_p_diag__m_w_concorr_c
gpu_z_w_con_c

```
AssignmentAndCopyKernelToMemcpyAndMemset().apply_pass(sdfg)
```

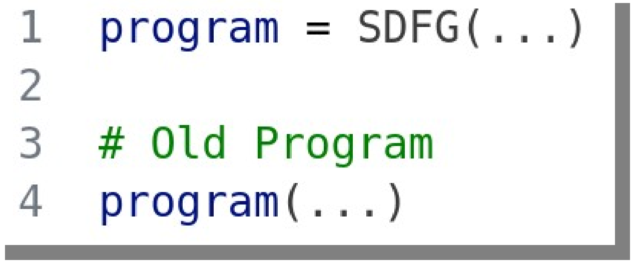


Step 4: Specialize for GPUs


single_state_body_1
[Max(3, nrdmax_jg - 2):88]
[i_startidx_var_148:i_endidx_var_149 + 1]
T_l544_c544
T_l536_c536
[Max(3, nrdmax_jg - 2):88]
[i_startidx_var_148:i_endidx_var_149 + 1]
gpu_levmask
gpu___CG_p_diag__m_w_concorr_c
gpu_z_w_con_c
memset_0_gpu_z_w_con_c
memset_0_gpu_levmask
gpu___CG_p_diag__m_w_concorr_c
gpu_z_w_con_c
gpu_levmask

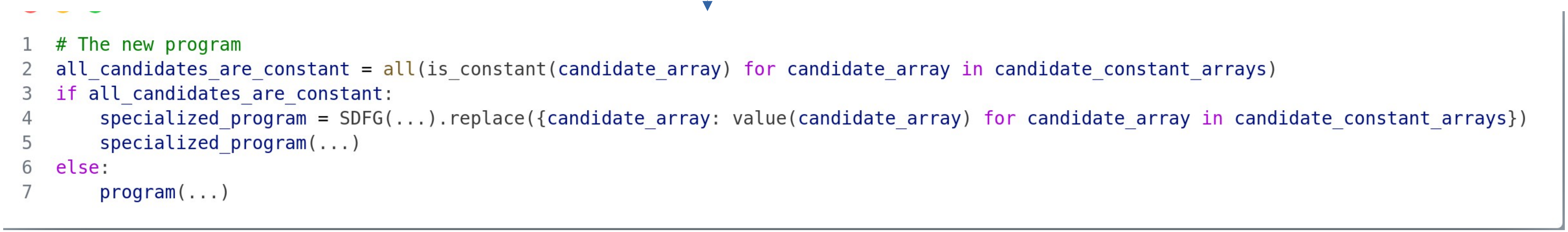
Fixing some bugs for the cases where a kernel involves multiple assignments and copies
But for patterns where it works, it results with performance improvement

Step 5: Further architecture and configuration specific optimizations



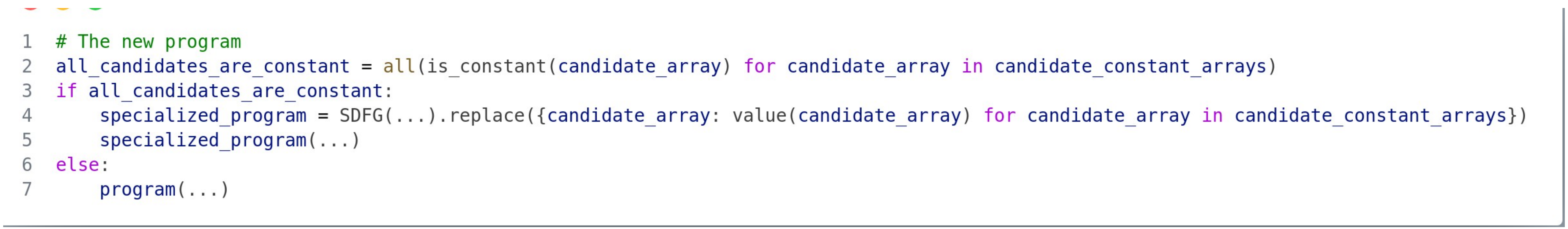
```
program = SDFG(...)

# Old Program
program(...)
```

Instrument the program to check the values of candidate arrays – and create two specialized programs.



```
# The new program
all_candidates_are_constant = all(is_constant(candidate_array) for candidate_array in candidate_constant_arrays)
if all_candidates_are_constant:
    specialized_program = SDFG(...).replace({candidate_array: value(candidate_array) for candidate_array in candidate_constant_arrays})
    specialized_program(...)
else:
    program(...)
```

Step 5: Further architecture and configuration specific optimizations

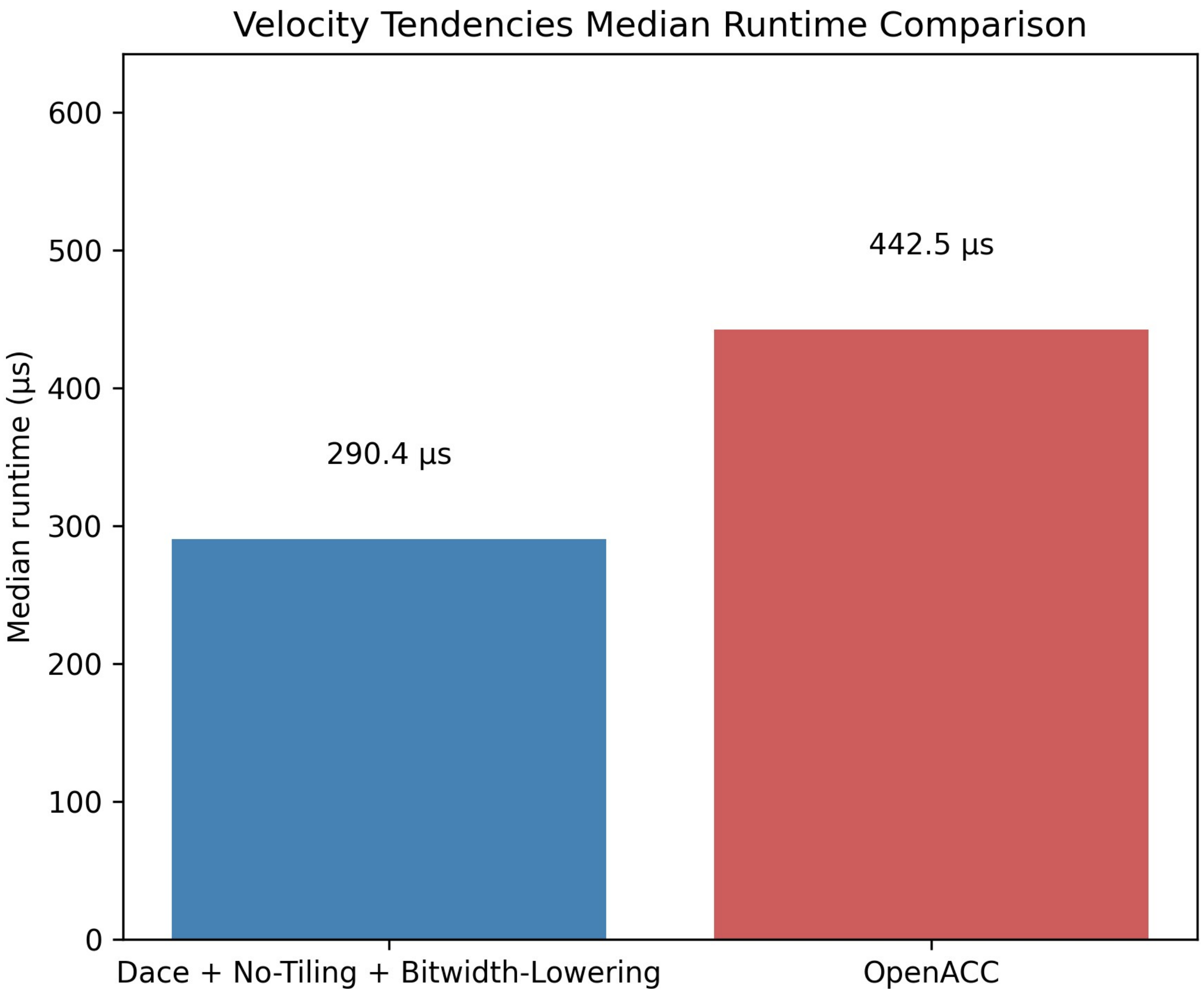


```python
# The new program
all_candidates_are_constant = all(is_constant(candidate_array) for candidate_array in candidate_constant_arrays)
if all_candidates_are_constant:
    specialized_program = SDFG(...).replace({candidate_array: value(candidate_array) for candidate_array in candidate_constant_arrays})
    specialized_program(...)
else:
    program(...)
```

When the SDFG is called for the first time, the check is performed.

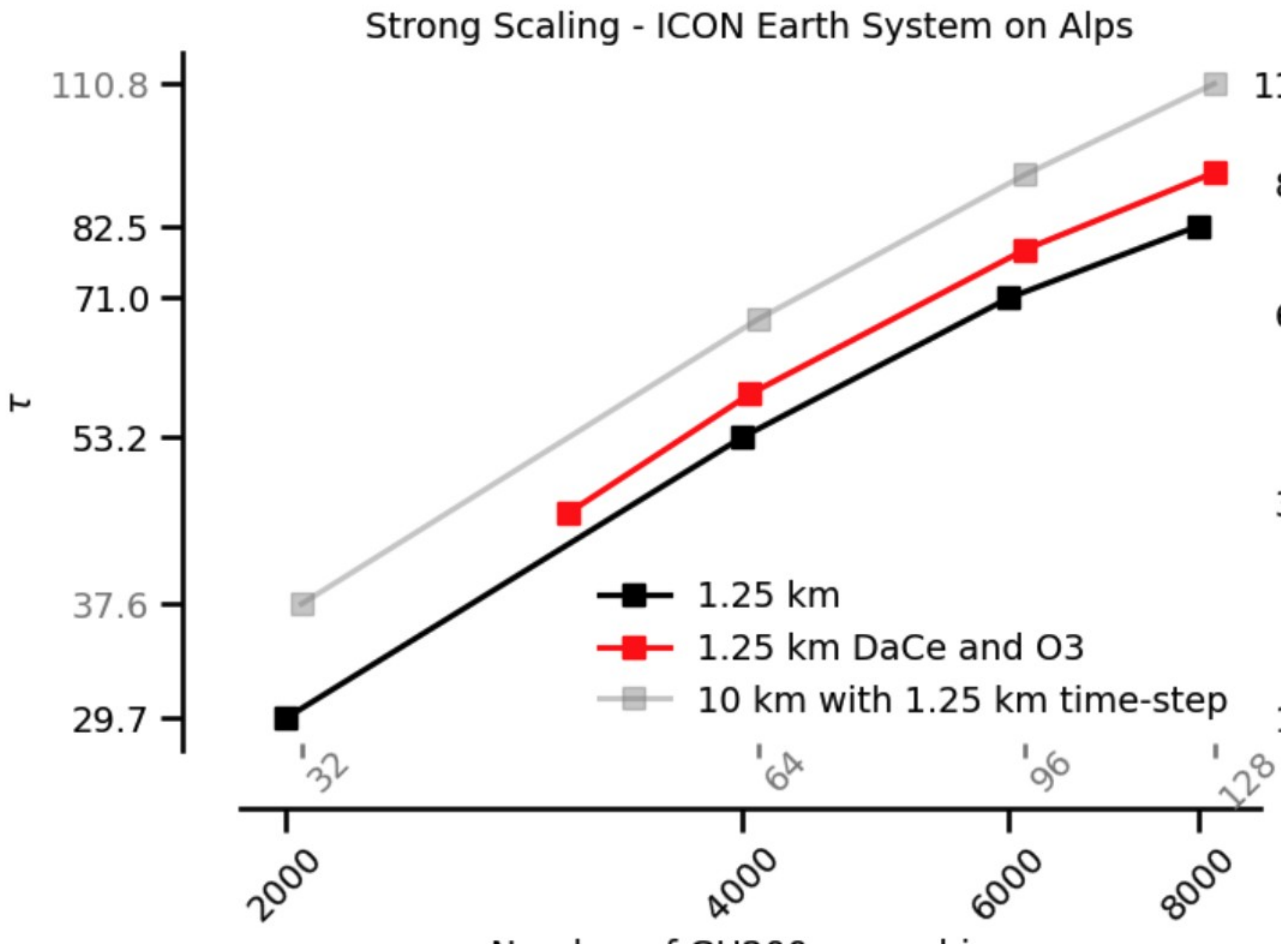
The transformation currently assumes these arrays are constant across program invocations.

Runtime Results:



1.5x speedup

Runtime Results:



Solve NH performance is improved by 20%

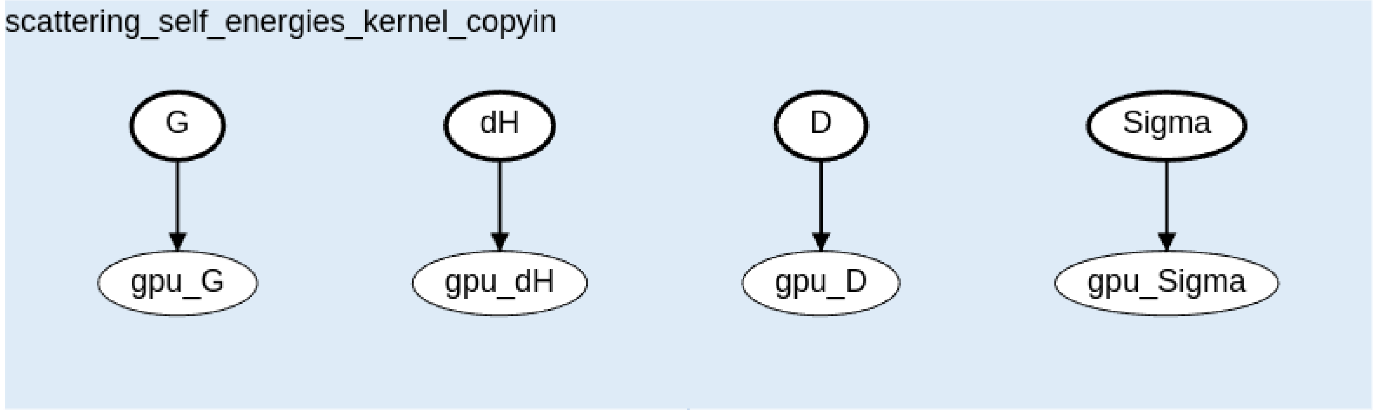
Total runtime is improved by 10%

Click to add Title

- **Improvements in the new GPU Code generation & And current Projects**

Click to add Title

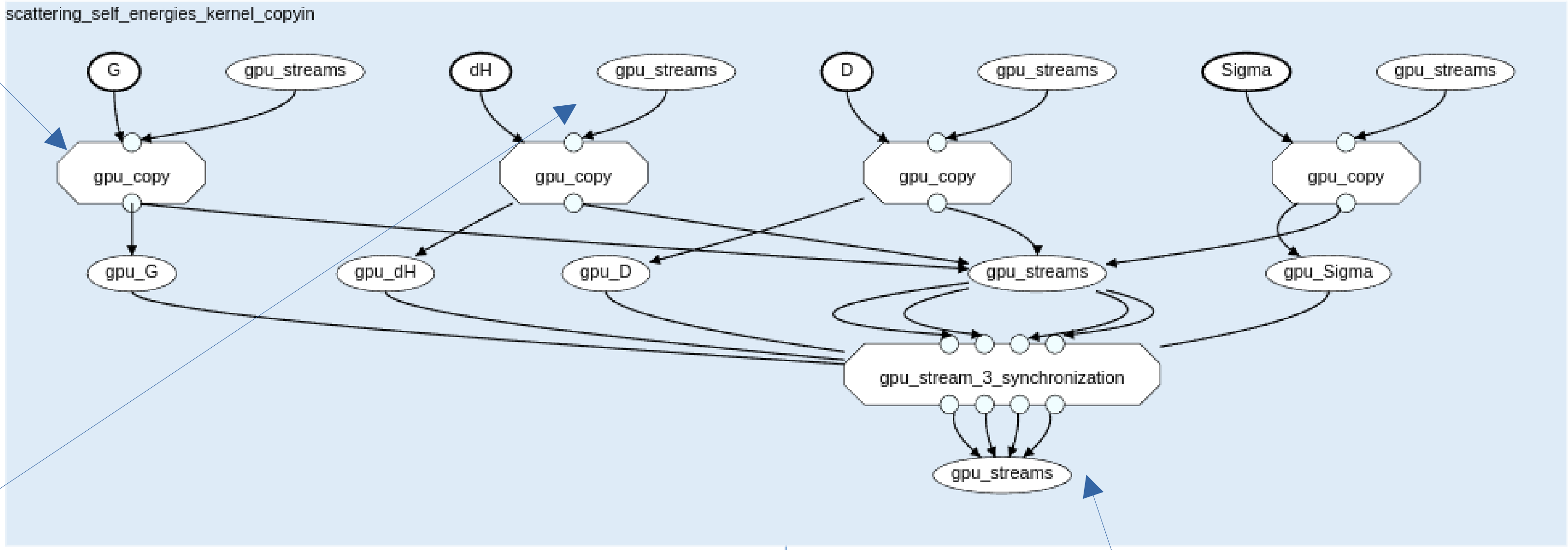
- **New GPU Codegen:** *In the previous codegen, copies were generated and synchronized implicitly*



Click to add Title

- **New GPU Codegen:** *In the previous codegen, copies were generated and synchronized implicitly*

GPU copies are lowered explicitly to copy-tasklets to expose them to different scheduling possibilities



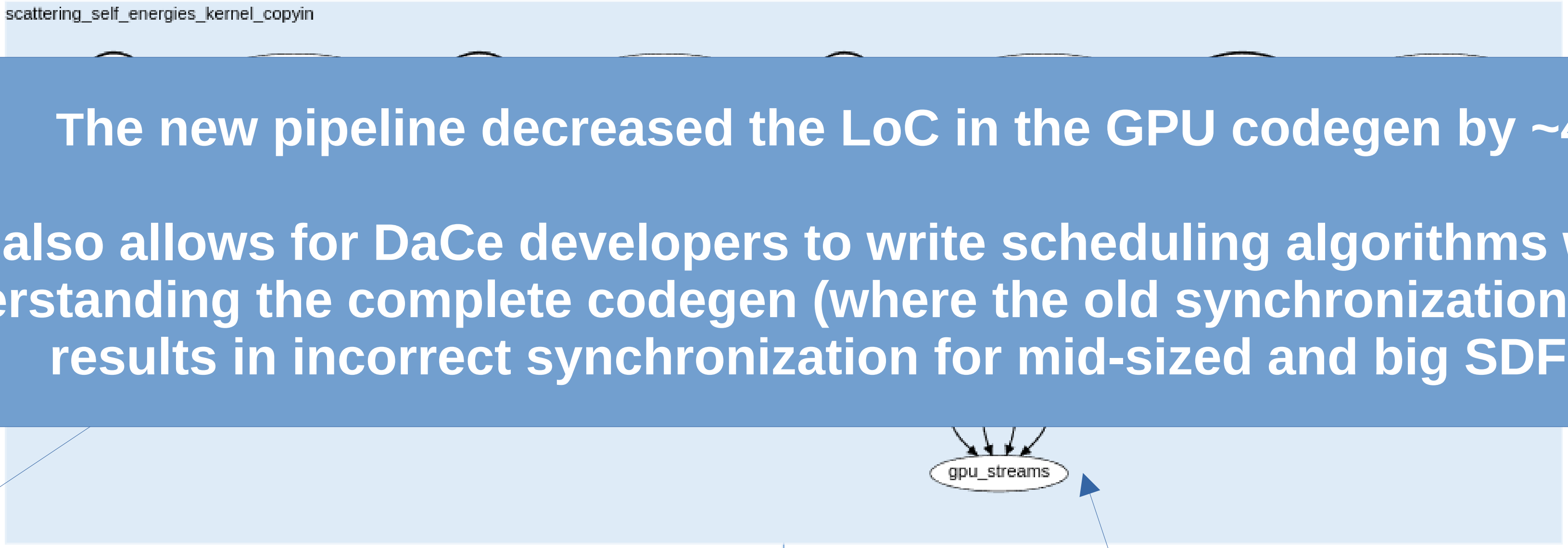
Scheduling implementations assign streams to tasklets and maps that require a stream

SDFG's state semantics require work of a state to be completed at state exit

Click to add Title

- **New GPU Codegen:** *In the previous codegen, copies were generated and synchronized implicitly*

GPU copies are lowered explicitly to copy-tasklets to expose them to different scheduling possibilities



The new pipeline decreased the LoC in the GPU codegen by ~400

It also allows for DaCe developers to write scheduling algorithms without understanding the complete codegen (where the old synchronization algorithm results in incorrect synchronization for mid-sized and big SDFGs)

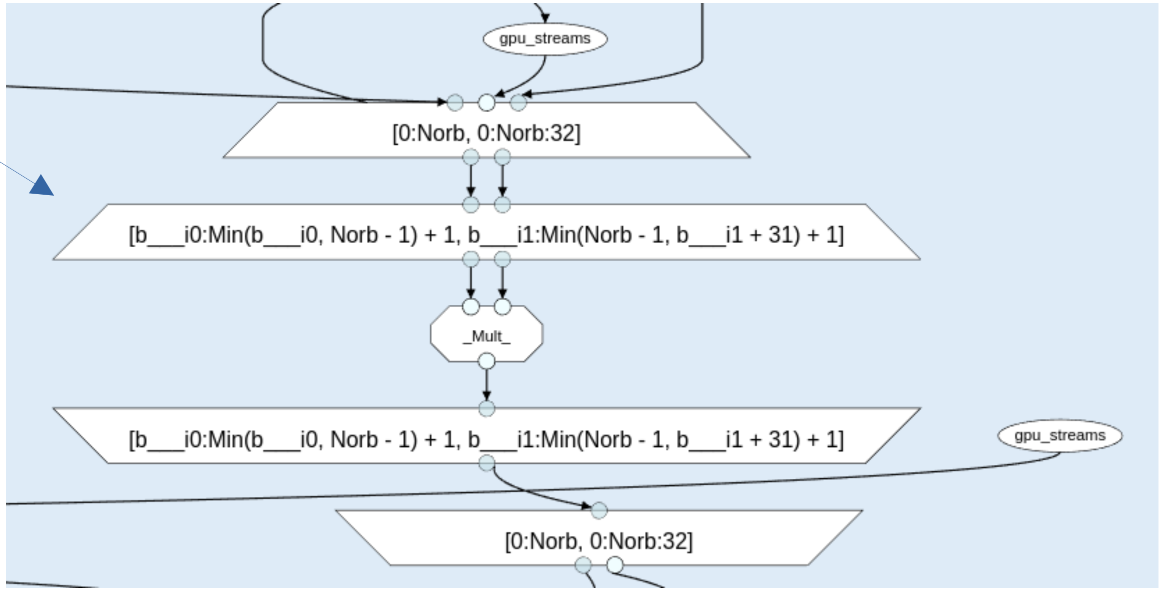
Scheduling implementations assign streams to tasklets and maps that require a stream

SDFG’s state semantics require work of a state to be completed at state exit

Click to add Title

- **New GPU Codegen:** *We enforce all GPU kernels to have an explicit thread block map*

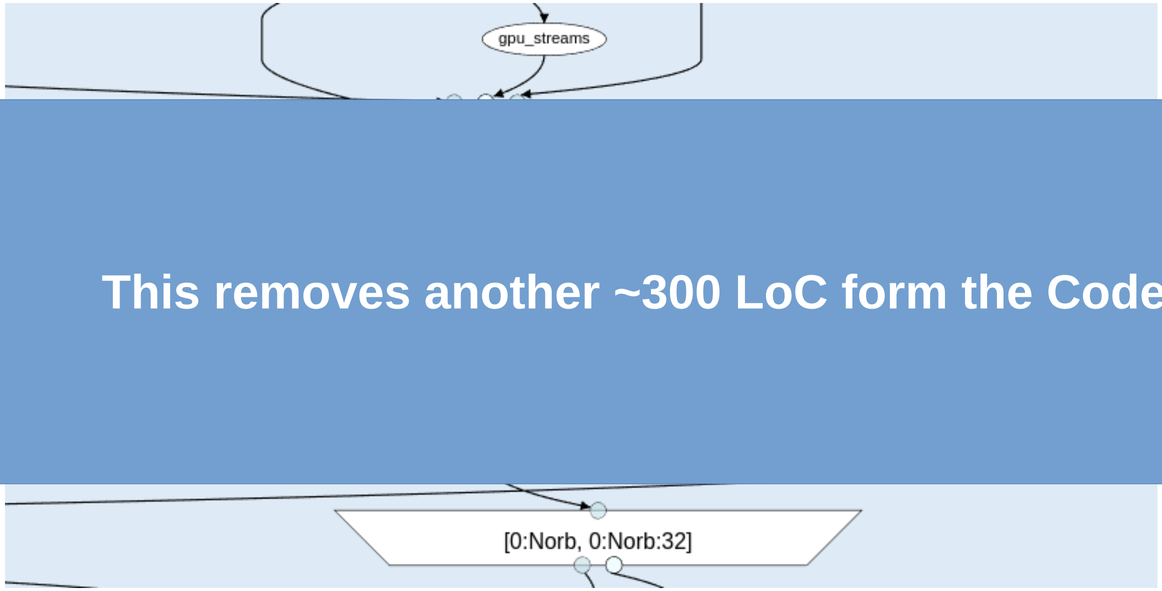
Threadblock map (in this case we have a thread block dimension of (32,1,1))


gpu_streams
[0:Norb, 0:Norb:32]
[b___i0:Min(b___i0, Norb - 1) + 1, b___i1:Min(Norb - 1, b___i1 + 31) + 1]
Mult
[b___i0:Min(b___i0, Norb - 1) + 1, b___i1:Min(Norb - 1, b___i1 + 31) + 1]
gpu_streams
[0:Norb, 0:Norb:32]

Click to add Title

- **New GPU Codegen:** *We enforce all GPU kernels to have an explicit thread block map*

  Threadblock map (in this case we have a thread block dimension of (32,1,1))



This removes another ~300 LoC form the Codegen

Click to add Title

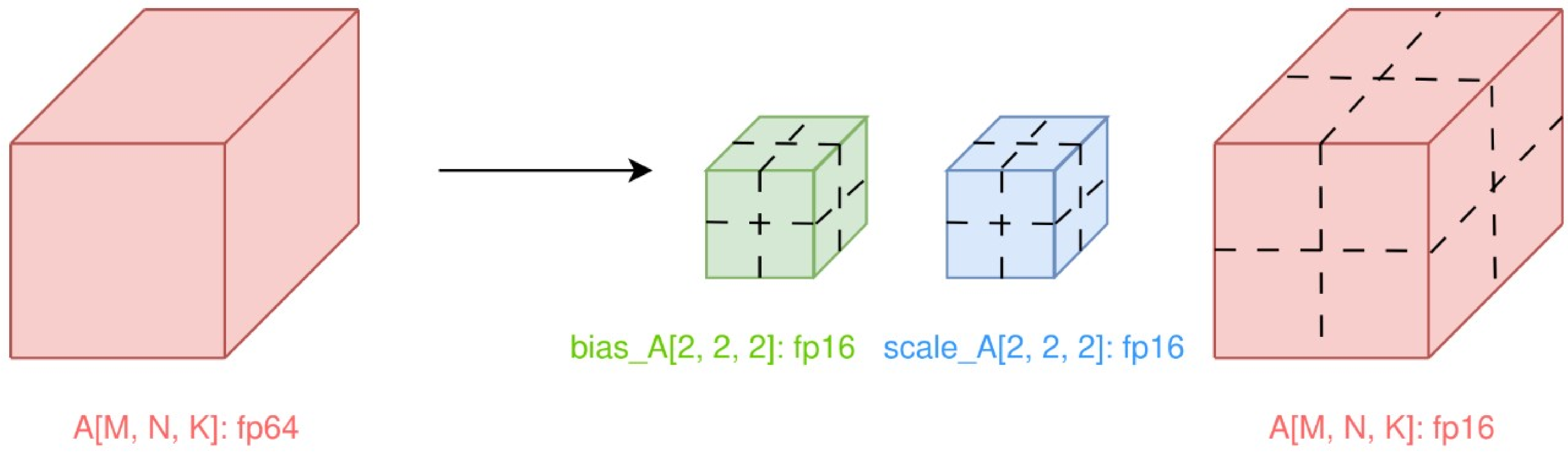
- **Current Open Projects**

Click to add Title

- **Blocked FP Formats**

Click to add Title

- *Recruited a student to work on the project. Proposal can be found under this link.*
- *First aim is to represent FP64 arrays as a block of FP16-arrays accompanied with FP64-bias and FP64-scale arrays.*



Click to add Title

- **New Offloading Transformation**
- **DaCe + SoftHier Backend**

Click to add Title

- **DaCe + SoftHier Backend:**
  - *Sad news, the student chose another project. Now I am interviewing candidates.*
  - *This project is not so popular with students. Some projects gets >5 applicants in the first week, this got 1 but the student was not good enough.*
  - *Project proposal can be found under this link and on the SPCL website.*
- **New Offloading Transformation:**
  - *Uploaded it couple of days ago to the SPCL website. Waiting for candidates.*
  - *Design document can be found under this link.*

Click to add Title

- **NPBench Modernization:**
  - *I have student working on integrating CUDA libraries, Performance counters (LIKWID, PAPI) and microbenchmarks (Stress-ng, GPU-Stream, Stream, etc.) to NPBench to automatically generate performance plots*
  - *I will also create student projects to extend existing kernels to Triton and TVM.*
  - *Project proposal can be seen under this link.*
- **WMMA / CUTLASS / CuTe GEMM Expansions in DaCe:**
  - *In the upcoming months I will also start a project to extend DaCe GEMM expansions to support using WMMA / CUTLASS / CuTe backends*