YAKUP KORAY BUDANAZ

SoftHier Progress Report February 24

Overiew of the Topics:

- Struct-of-Array Flattening
- End-to-end Optimization Stacks (+ How DaCe Compares to Them)
- DaCe + SoftHier
- Transformation Updates
- Outlook For Next Weeks

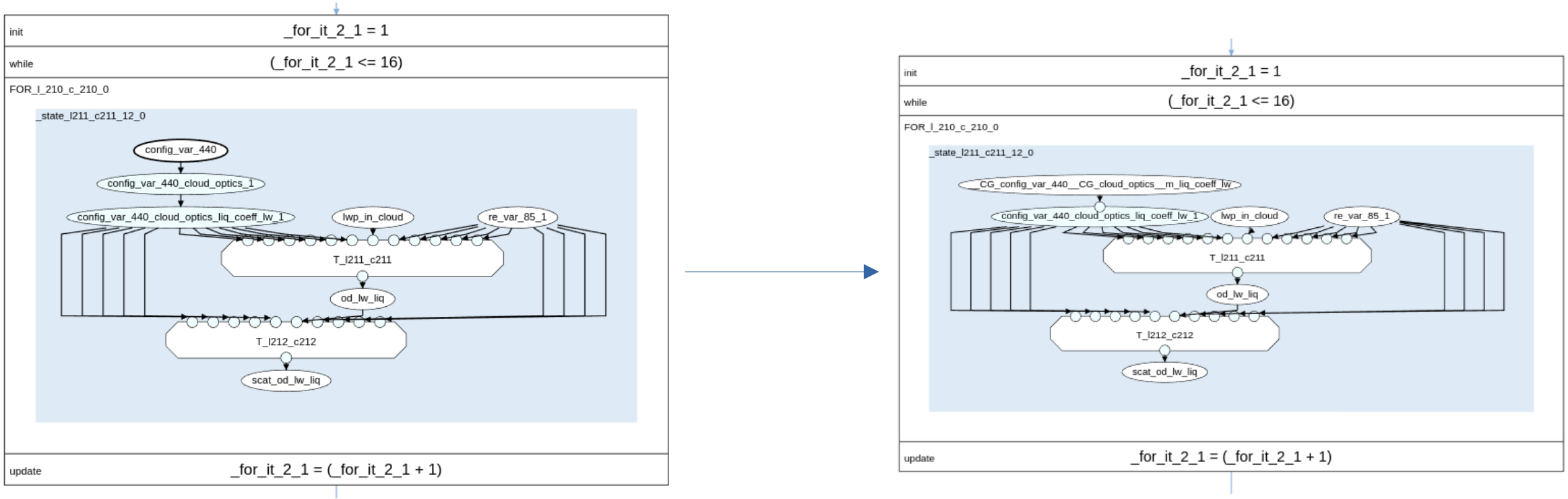
- **Struct-of-Array Flattening**

Struct-of-Array Flattening

- After multiple bugfixes and extensions, I was able to apply the flattening pass on SDFGs obtained from ECRAD.
- Also provided an *interface mode* to the pass that enables to call the SDFG with the original structs the untransformed SDFG would use, but the computation uses SoA arrays.

Struct-of-Array Flattening

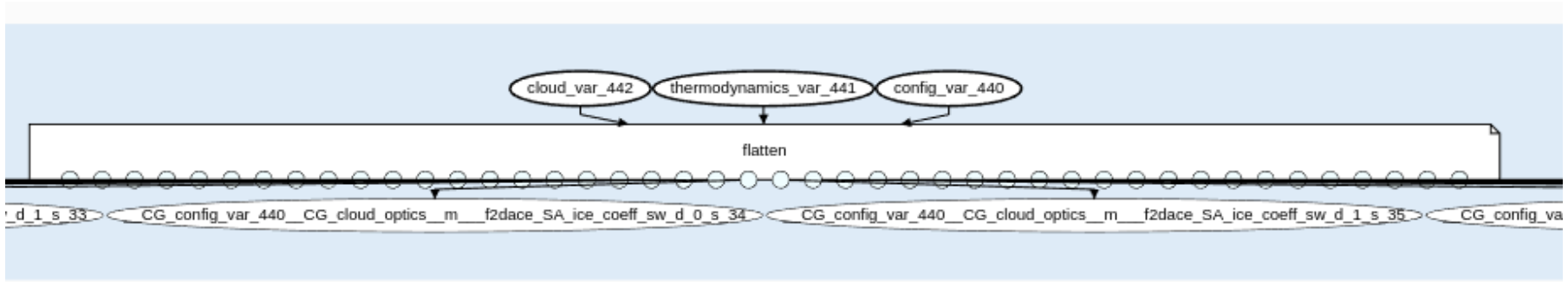
1) Analyze struct hierarchy and register corresponding multi-dimensional arrays.
2) Preprocess the struct accesses to a form easier to replace.
3) Replace all view-chains with an SoA array and a view that reduces the dimensionality.



Struct-of-Array Flattening

4) (Optional) Generate Struct-to-SoA copy-in and SoA-to-Struct copy-out functions
5) Add the copy-in and copy-out procedures to the beginning and the end of the SDFG

...



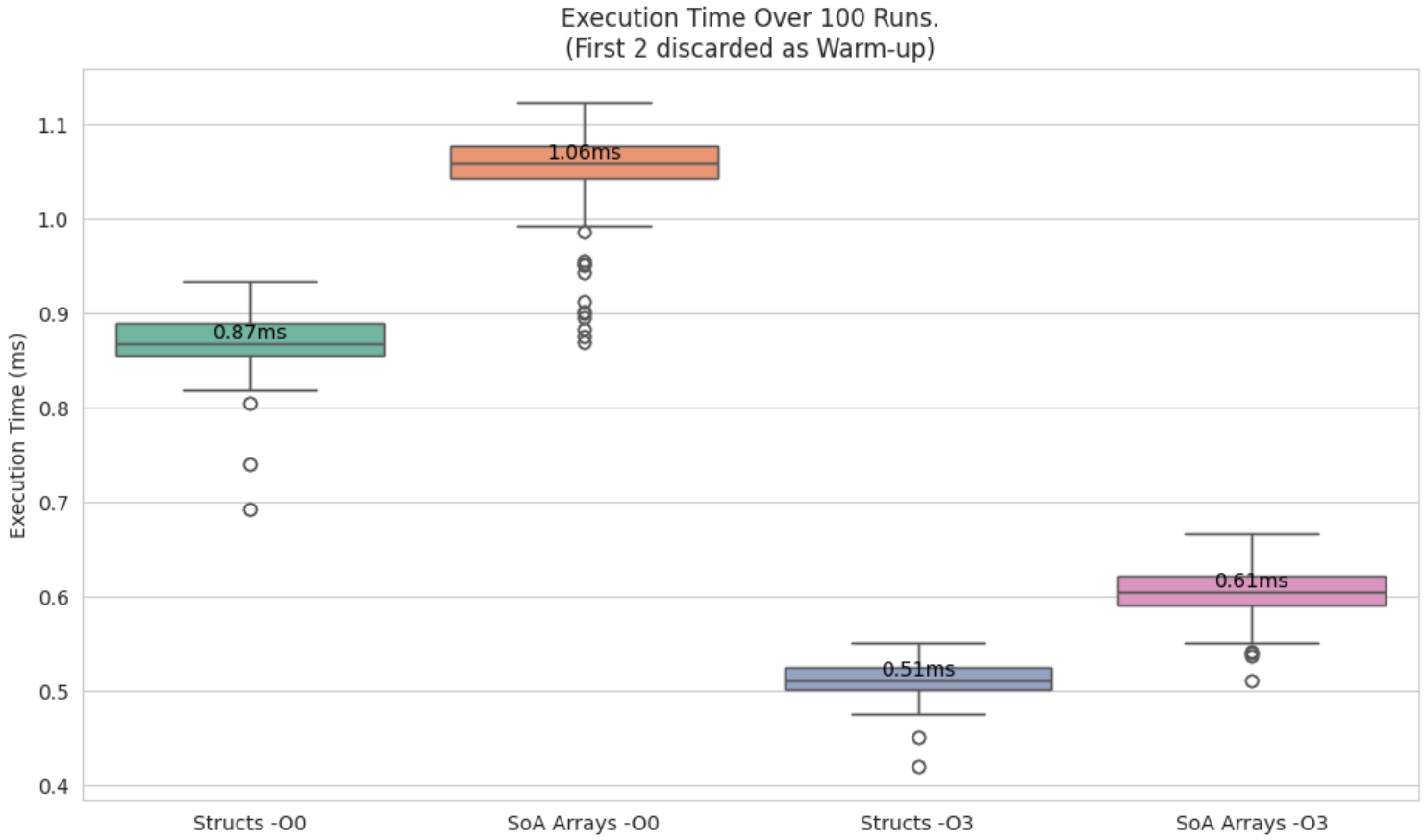
...

(It can generate a lot of SoA arrays)

(flatten → copy-in, deflatten → copy-out, looks the same)

Struct-of-Array Flattening

- How much is the overhead?
- Since I run it on a small example, I ran it using single core.
- Ofc. Impact depends on the struct hierarchy and how parallelizable the copy is.

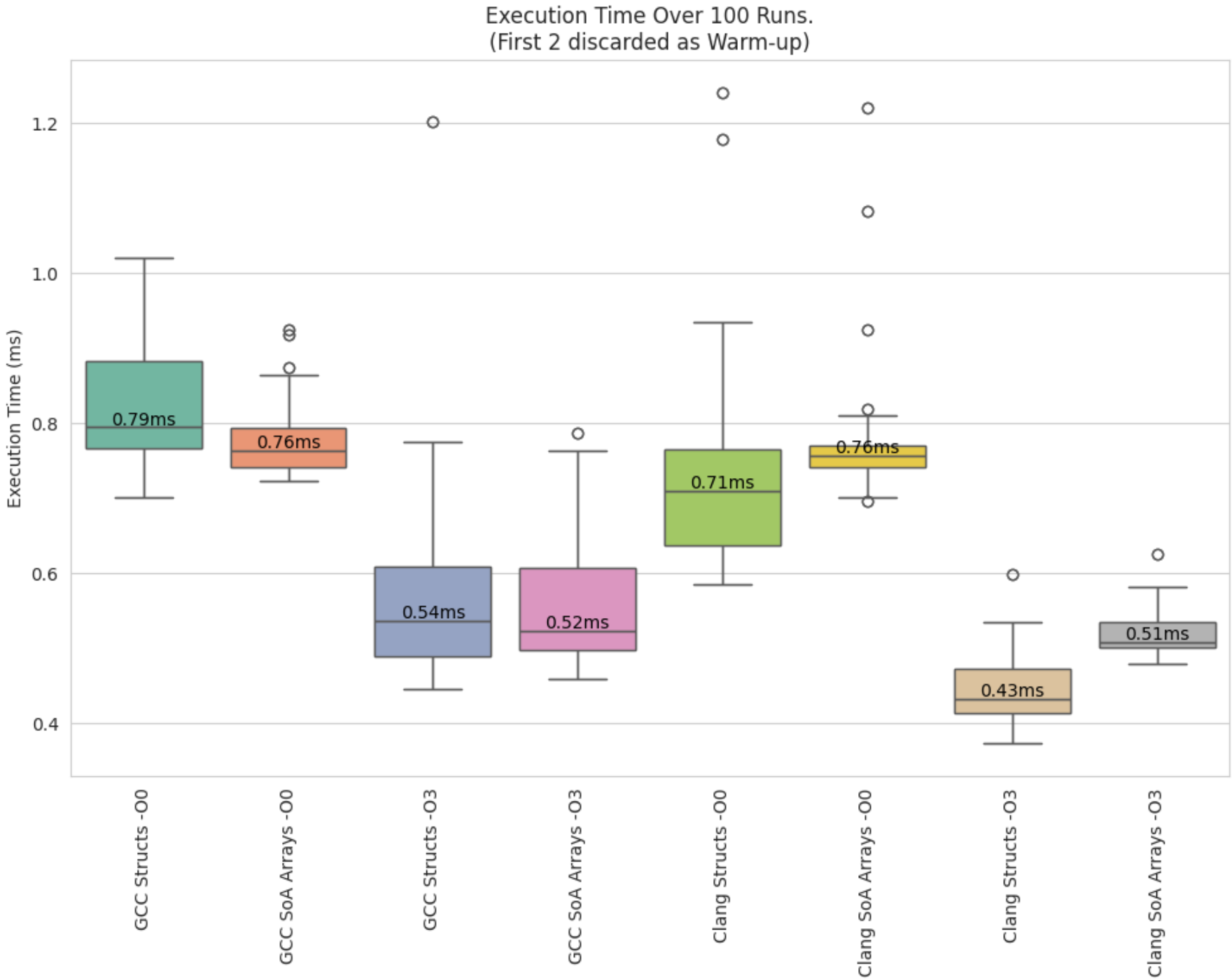


AMD EPYC 7742
GCC@10.2

Struct-of-Array Flattening

- Clang vs GCC!
- Since I run it on a small example, I ran it using single core.
- My local laptop (waited 10s between runs of different configurations)
- Why local laptop? I am compiling LLVM and GCC on Aul currently.

AMD Ryzen 7 8845HS
Clang@18.1.3
GCC@13.3



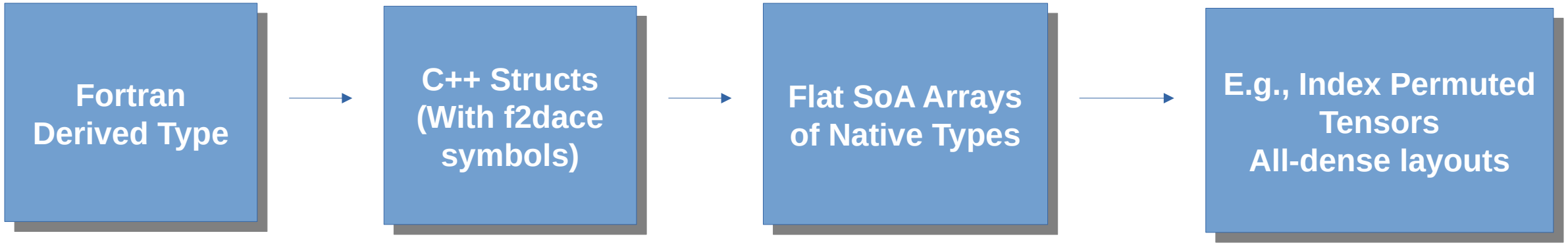
Struct-of-Array Flattening

- I am compiling a new GCC version on AULT
- And also compiling a new version of LLVM (+Clang) on AULT
  - To test compiler's impact on code performance.

Velocity Tendencies Optimizations

Fortran derived type to C++ type copy-in and copy-out functions were already developed.

This allows easier layout transformations that will enable the efficient use of vector units



C++ Structs to SoA Arrays of native types copy-in and copy-out functions I have completed in these weeks.

- **End-to-end Optimization Stacks (+ How DaCe Compares to Them)**

End-to-end Optimization Stacks (+ How DaCe Compares to Them)

- There is only one competitive end-to-end optimization stack that works: **TVM**

End-to-end Optimization Stacks (+ How DaCe Compares to Them)

- I am integrating TVM for GPUs and CPUs to NPBench
- Our HPC microbenchmarking suite
- I also starting comparing my auto-tiling to TVM template based auto-tile and auto-scheduler

End-to-end Optimization Stacks (+ How DaCe Compares to Them)

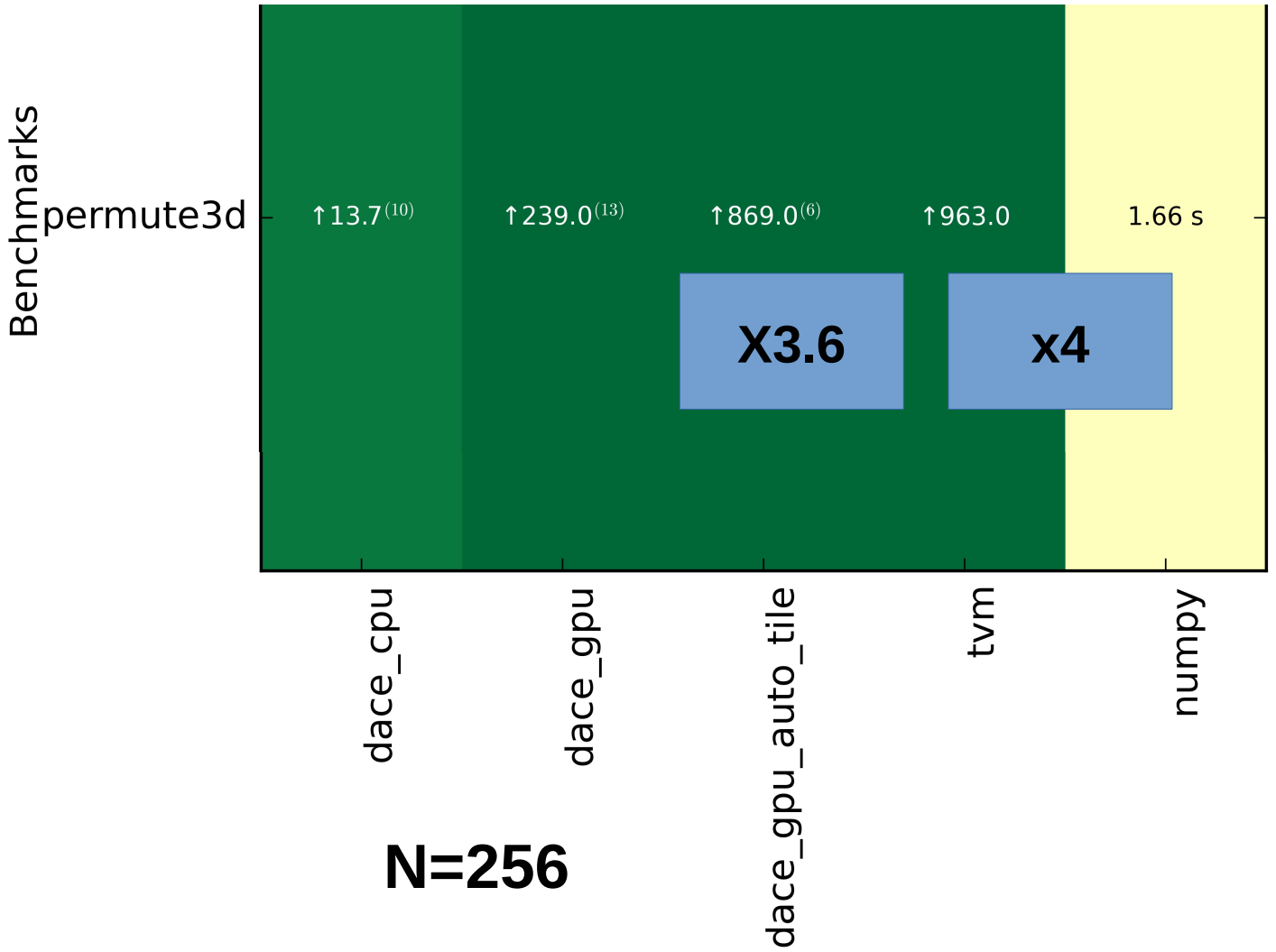
- Template based tiling of TVM works very good.
- It is faster than DaCe code for the same tiling – I am investigating why.
- The un-templated auto-scheduler (also known as Ansor), does not find a proper candidate program and is slow. I am also investigating this.

End-to-end Optimization Stacks (+ How DaCe Compares to Them)

Even if the same tiling parameters are enforced, TVM kernel is faster.

Results from RTX4050 because I am currently trying a lot of things with the Ansor auto-scheduler and improving the launch latency of DaCe.

Median of 10 runs

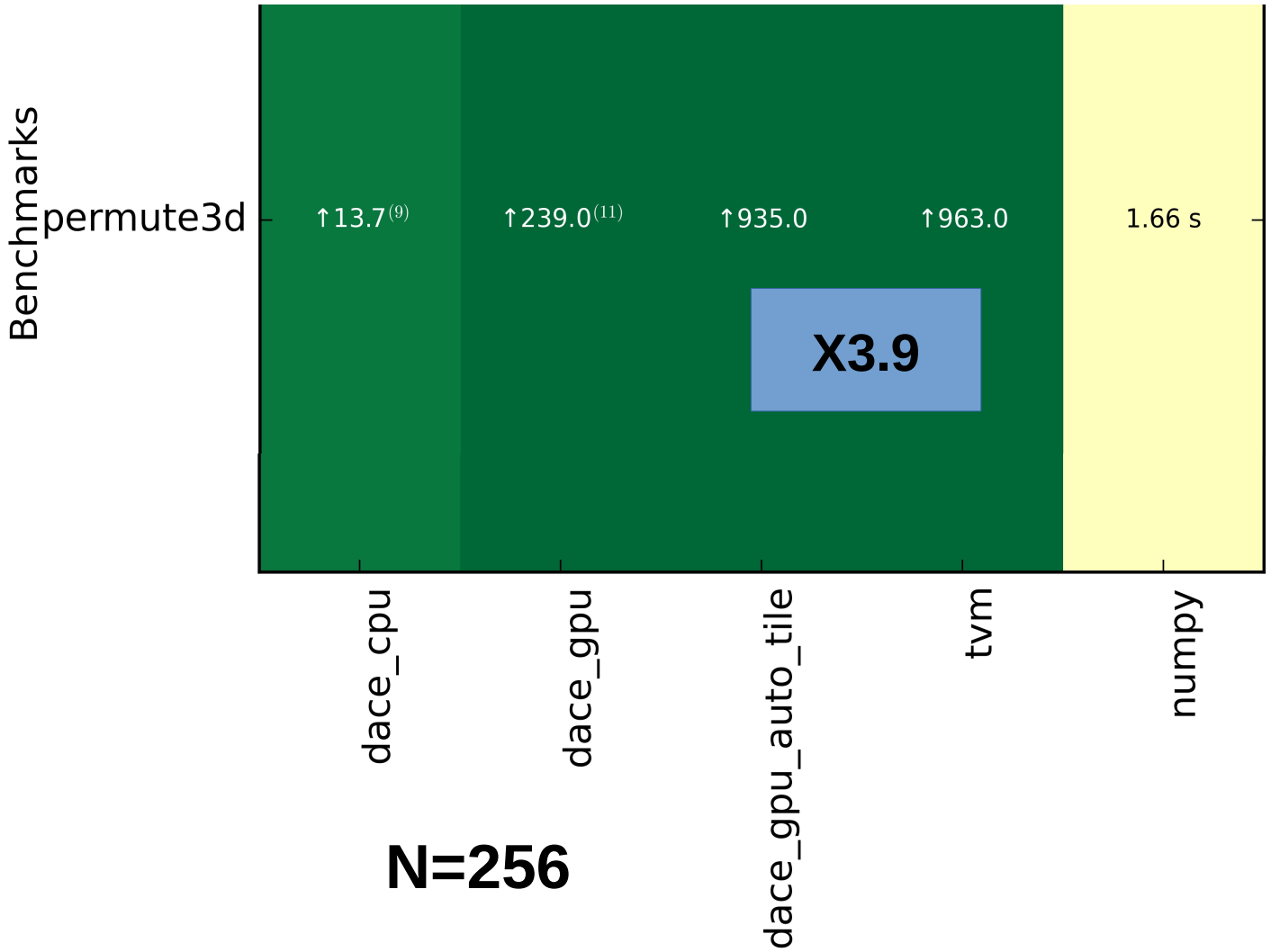


End-to-end Optimization Stacks (+ How DaCe Compares to Them)

The input sanitation at launch is adding overhead for short-lived kernels.

But why is TVM still slightly faster? I am investigating it.

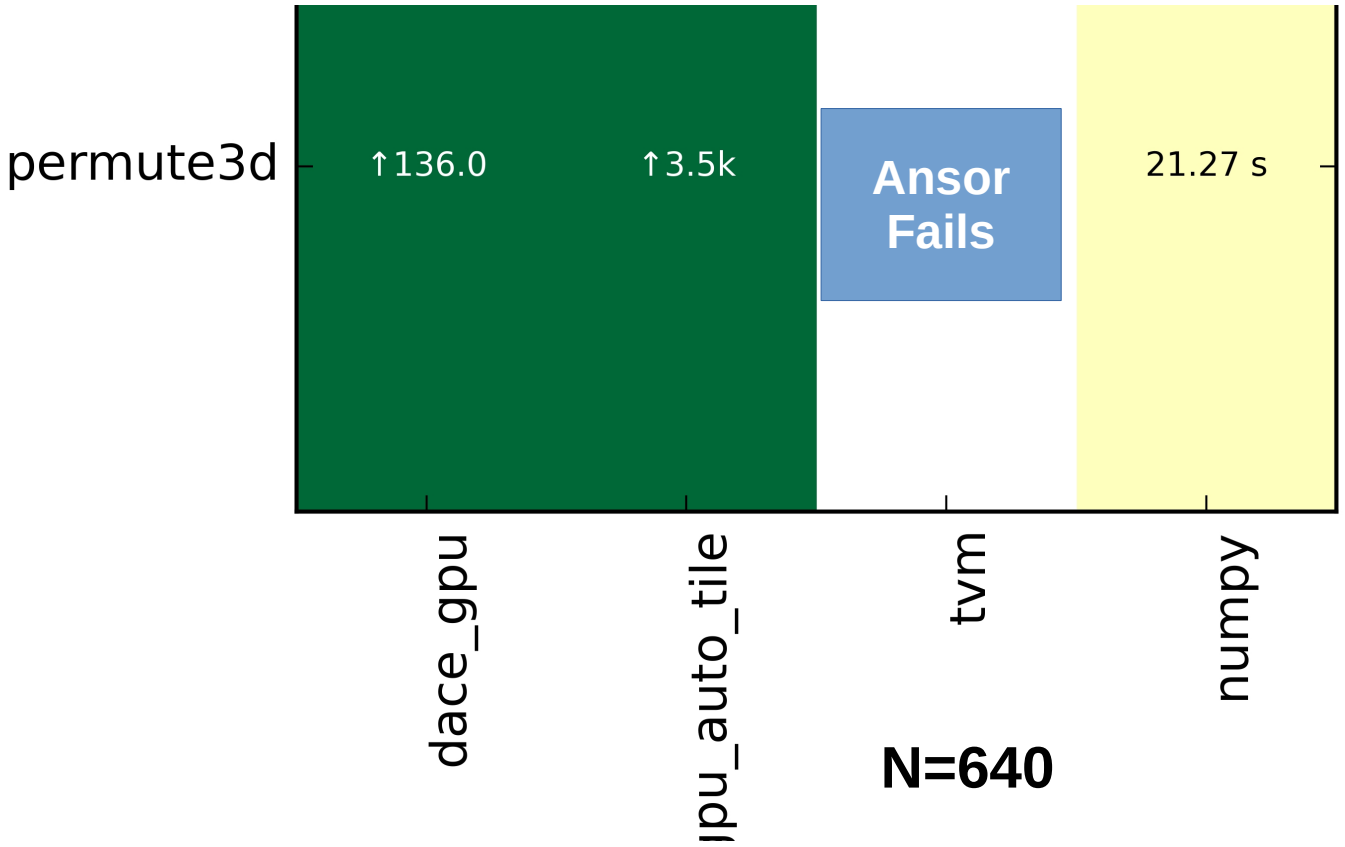
I have seen flags such as TVM_L2_PREFETCH etc. that might be improving the behavior.



End-to-end Optimization Stacks (+ How DaCe Compares to Them)

Results are from a V100 Node on AULT Preset XL.

I am also investigating why Ansor fails to find a schedule.

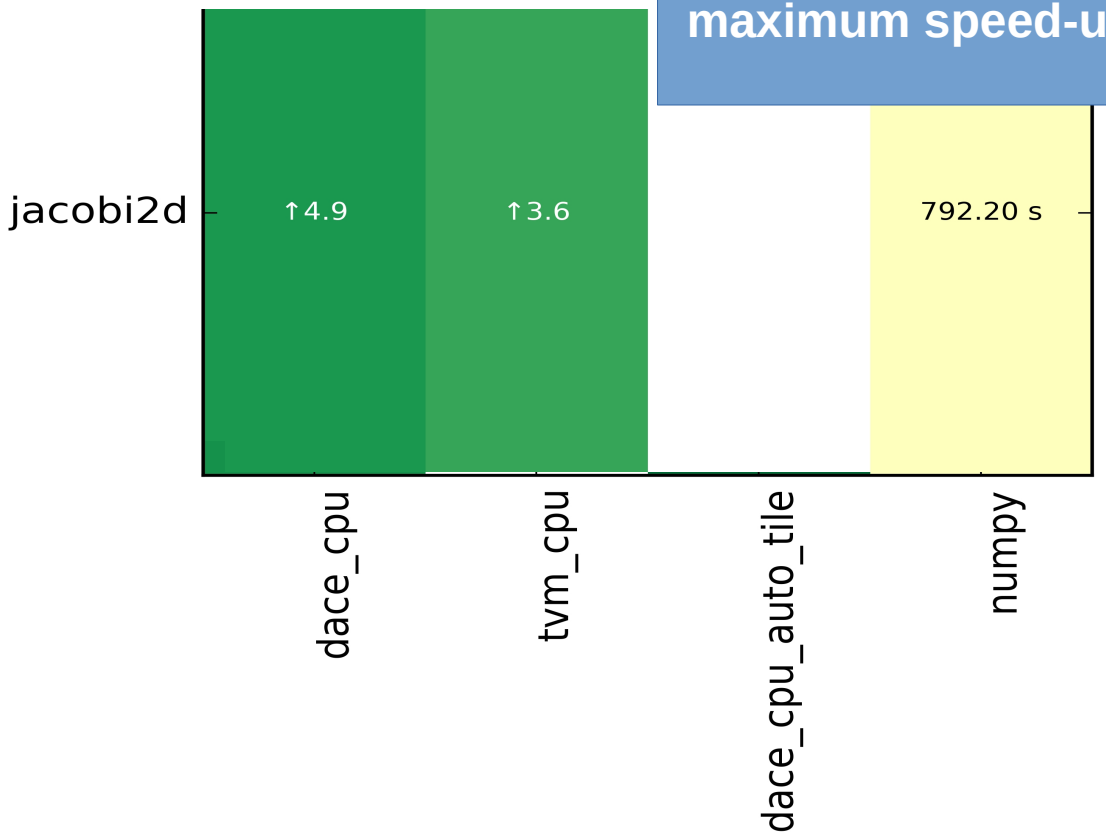


End-to-end Optimization Stacks (+ How DaCe Compares to Them)

TVM CPU looks to perform not so good.

Also, default OpenMP schedule seems to work very good most of the times.

Tiler Runs, but there are some issues in returning the complete tiled program.
In the individual kernel level, the maximum speed-up is x1.3 over Dace



- **DaCe + SoftHier**

DaCe + SoftHier

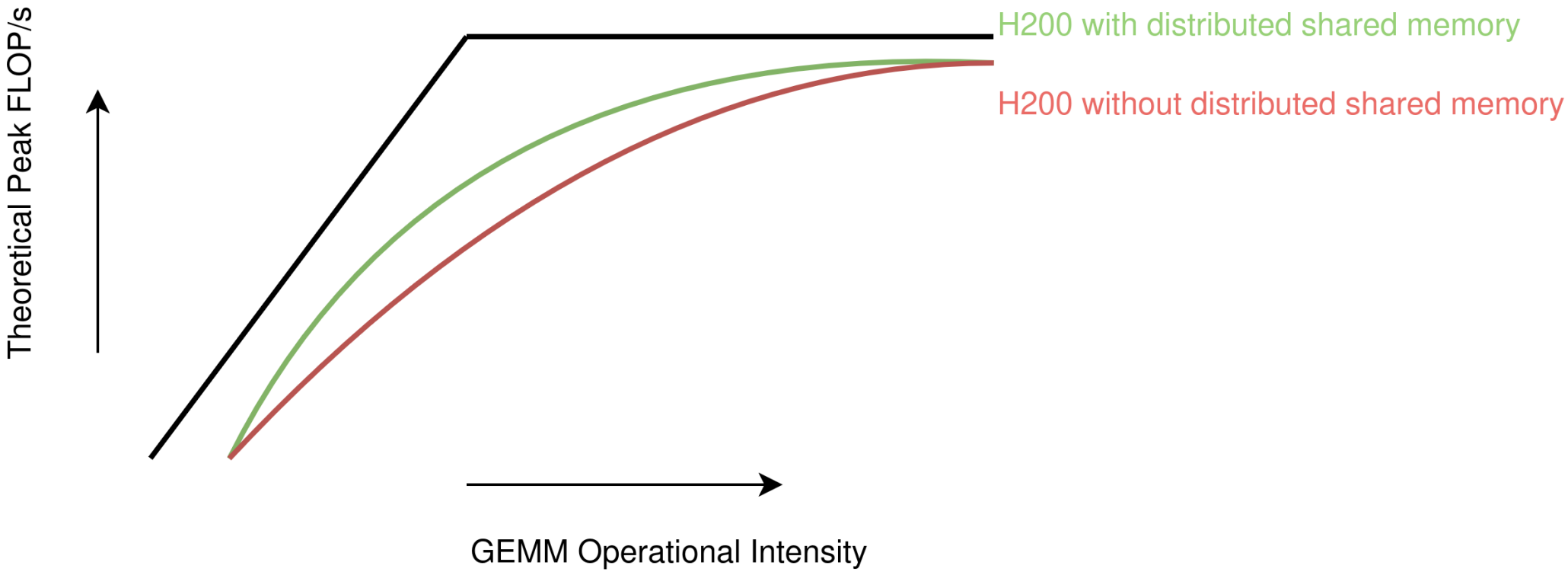
- With Aofeng we are working towards a paper to submit to HPCA in August.
- The plan is to run a lot of simulations of the SoftHier architecture (many many configurations)
- With multiple different kernel implementations (multiple schedules of Kernels) and incorporating the auto-tiler to the procedure (many many kernels).

DaCe + SoftHier

- After this work, we plan on a second paper where we will explicitly program AMD GPUs with multiple distinct programmable NUMA domains and Nvidia H100 GPUs with on-chip interconnect to show to impact of on-chip NoC.
- We also plan to use SoftHier simulator to mimic the properties of A100 and 910B to showcase how the kernel performance can be improved if on-chip interconnect was available on these devices in a second paper.

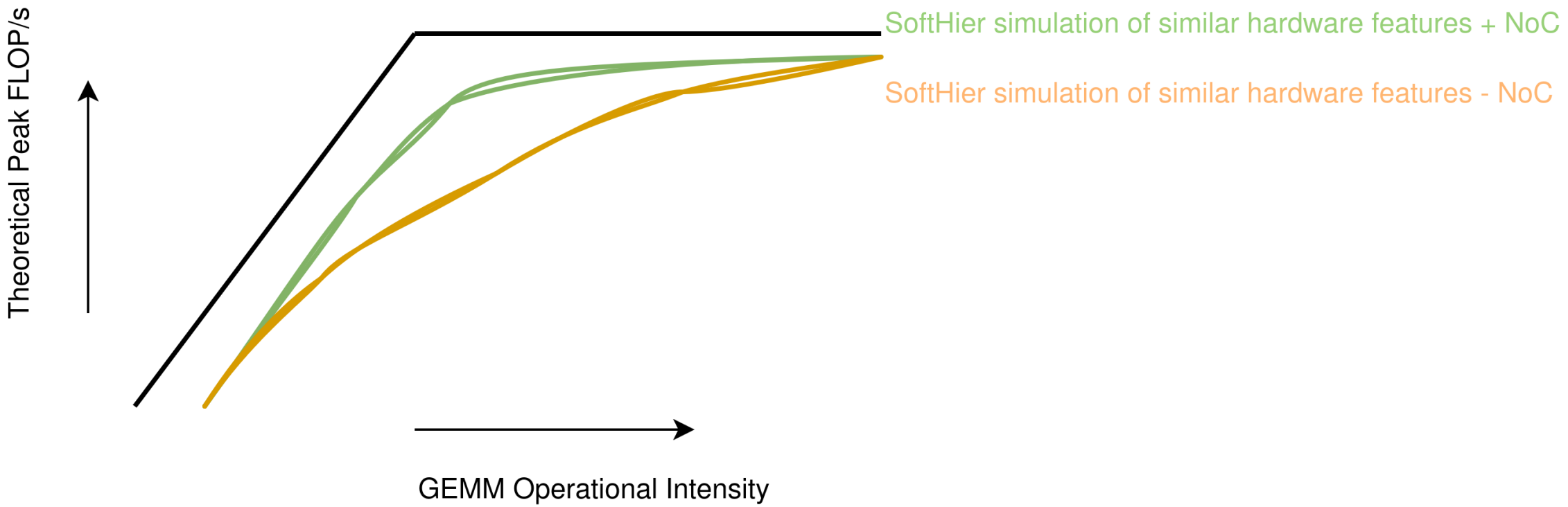
DaCe + SoftHier

- Step 1: Implement kernels that can benefit from NoC on hardware that has programmable NoC, using both NoC and not using NoC
- Small stencils that can fit into GPU SRAM, GEMM kernels



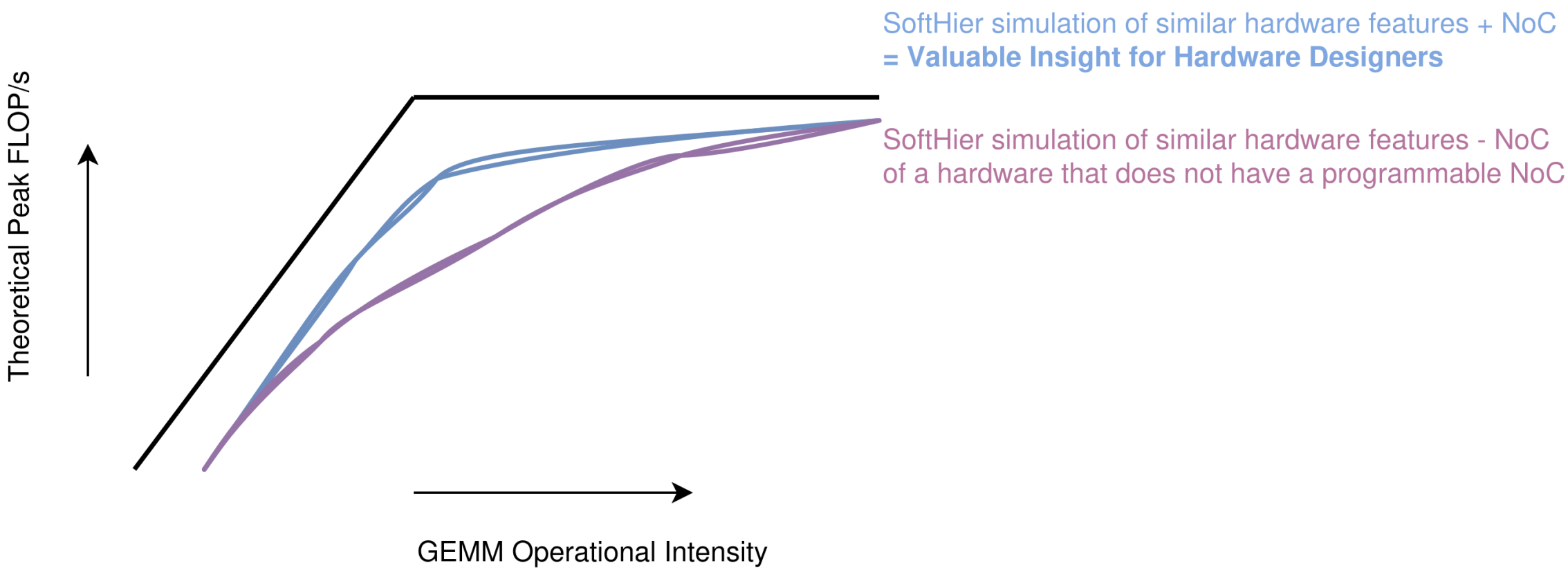
DaCe + SoftHier

- Step 2: Verify the SoftHier simulation is correct by simulating the kernels on SoftHier simulator



DaCe + SoftHier

- Step 3: Extend the simulator to other AI accelerators that do not provide a programmable NoC, e.g., 910B and show the possible improvements



- **Transformation Updates**

Transformation Updates

- B

- **Outlook for the Next Weeks**

Outlook for the Next Weeks

- B