DaCe: Data Centric parallel programming for heterogeneous platforms

Tiziano De Matteis
Vrije Universiteit Amsterdam

Who am I?

Tiziano De Matteis

Assistant Professor at VU Amsterdam (joined March 2023)

Research Interests

I am interested in High-Performance Computing, Energy Efficiency, Systems.

Current focus: abstractions, methods, and systems to overcome the end of Moore's law.

Previously:

- PhD University of Pisa (on Parallel Streaming Analytics)
- PostDoc ETH Zurich (on FPGA for HPC)

Contributions to the FPGA-HPC community

Focus on methods and tools to productively program (with HLS) FPGAs for High-Performance Computing

Libraries/Tools

To increase programming **productivity**

SMI [1]

FBLAS [2]

hlslib [3]

Temporal Vectorization [4]

Programming Model

To provide **performance portability** in Heter. Architectures (not only FPGA!)

DaCe [5]

Applications

To show the potentials of reconfigurable hardware in **applications**

DaCeML

StencilFlow [6]

[1] T. De Matteis, J. de Fine Licht, J. Beránek, T. Hoefler. "Streaming Message Interface: High-Performance Distributed Memory Programming on Reconfigurable Hardware". SC'19

[2] T. De Matteis, J. de Fine Licht, T. Hoefler. "FBLAS: Streaming Linear Algebra on FPGA". SC'20

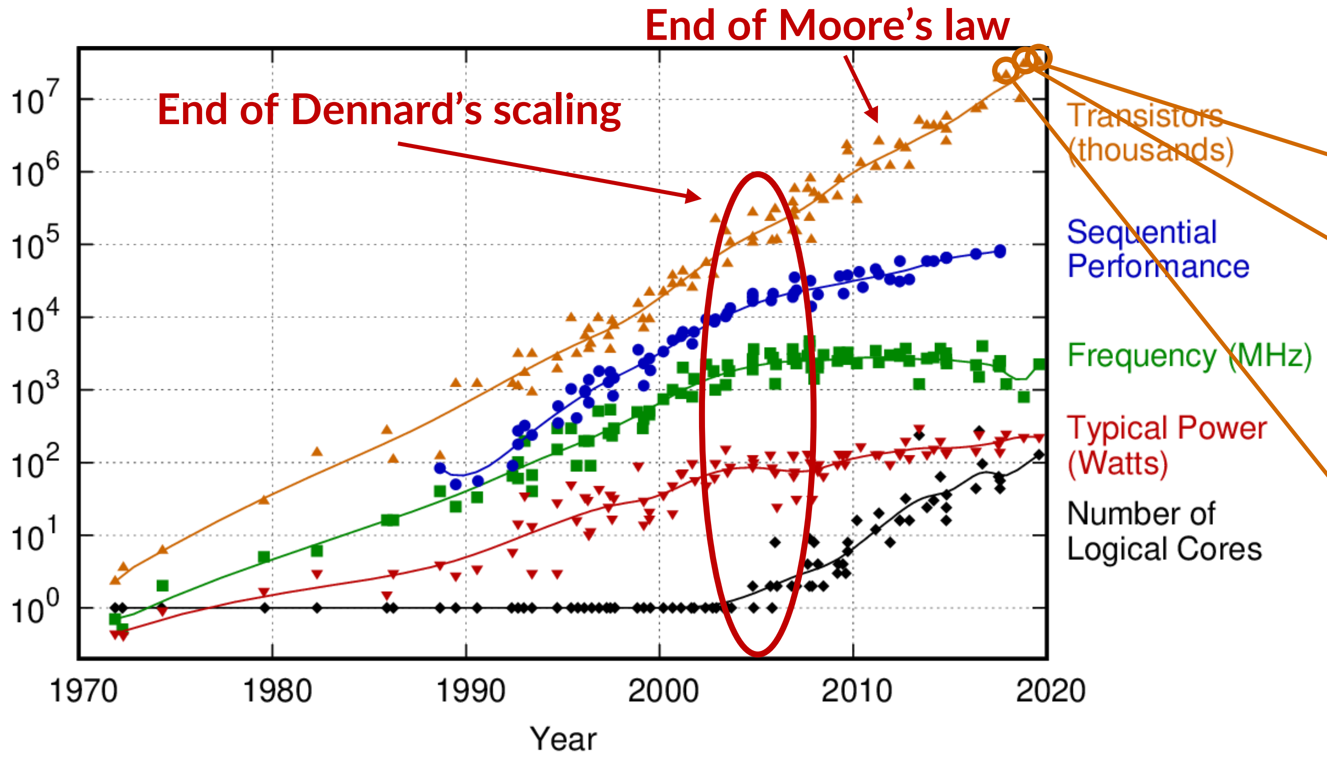
[3] J. de Fine Licht, T. Hoefler. "hlslib: Software Engineering for Hardware Design". H2RC'19

[4] C. Johnsen, T. De Matteis, T. Ben-Nun, J. de Fine Licht, T. Hoefler de Fine Licht, T. Hoefler. "Temporal Vectorization: A Compiler Approach to Automatic Multi-Pumping". ICCAD'22

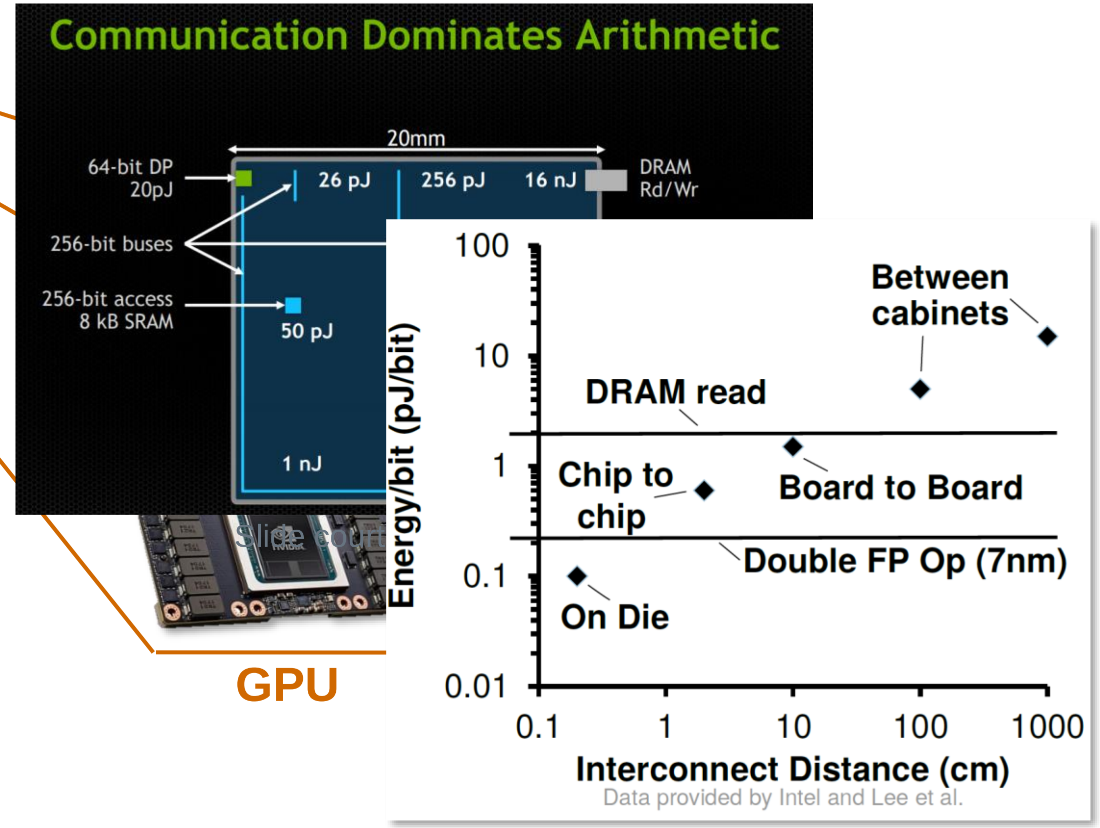
[5] A. N. Ziogas, T. Schneider, T. Ben-Nun, A. Calotoiu, T. De Matteis, J. de Fine Licht, L. Lavarini, T. Hoefler. "Productivity, Portability, Performance: Data-Centric Python". SC '21.

[6] J. de Fine Licht, A. Kuster, T. De Matteis, T. Ben-Nun, D. Hofer, T. Hoefler . "StencilFlow: Mapping Large Stencil Programs to Distributed Spatial Computing Systems". CGO '21

Changes in the Computing Landscape



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp



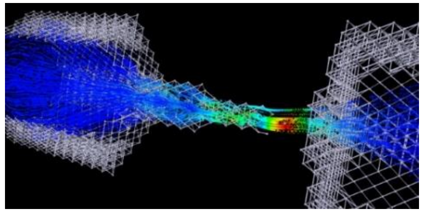
The holy grail of Performance Portability!

High-Performance Optimization -> Data Movement Reduction

Changes in the Software Landscape – The Scientific Languages

seaborn
matplotlib
SymPy
SciPy
pandas
NetworkX
Network Analysis in Python

MD ANALYSIS
GridTools



TensorFlow
PyTorch

Vast software ecosystem → **Productivity**
Performance?
Portability?

DaCe Overview

Domain Scientist

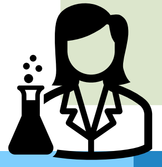
Problem Formulation

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

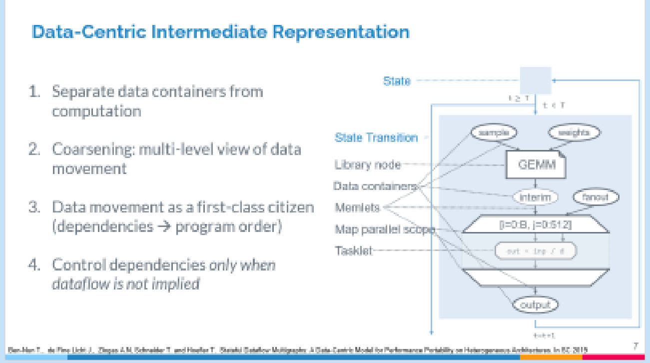
Python	DSLs
PyTorch	C

...

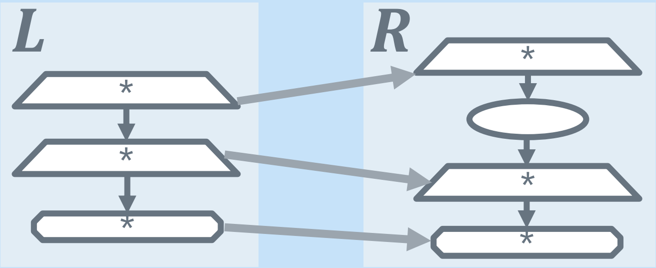
Scientific Frontend



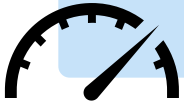
Performance Engineer



Data-Centric Intermediate Representation (SDFG)



Graph Transformations



Transformed Dataflow

Performance Results

System

Hardware Information

Compiler

Runtime

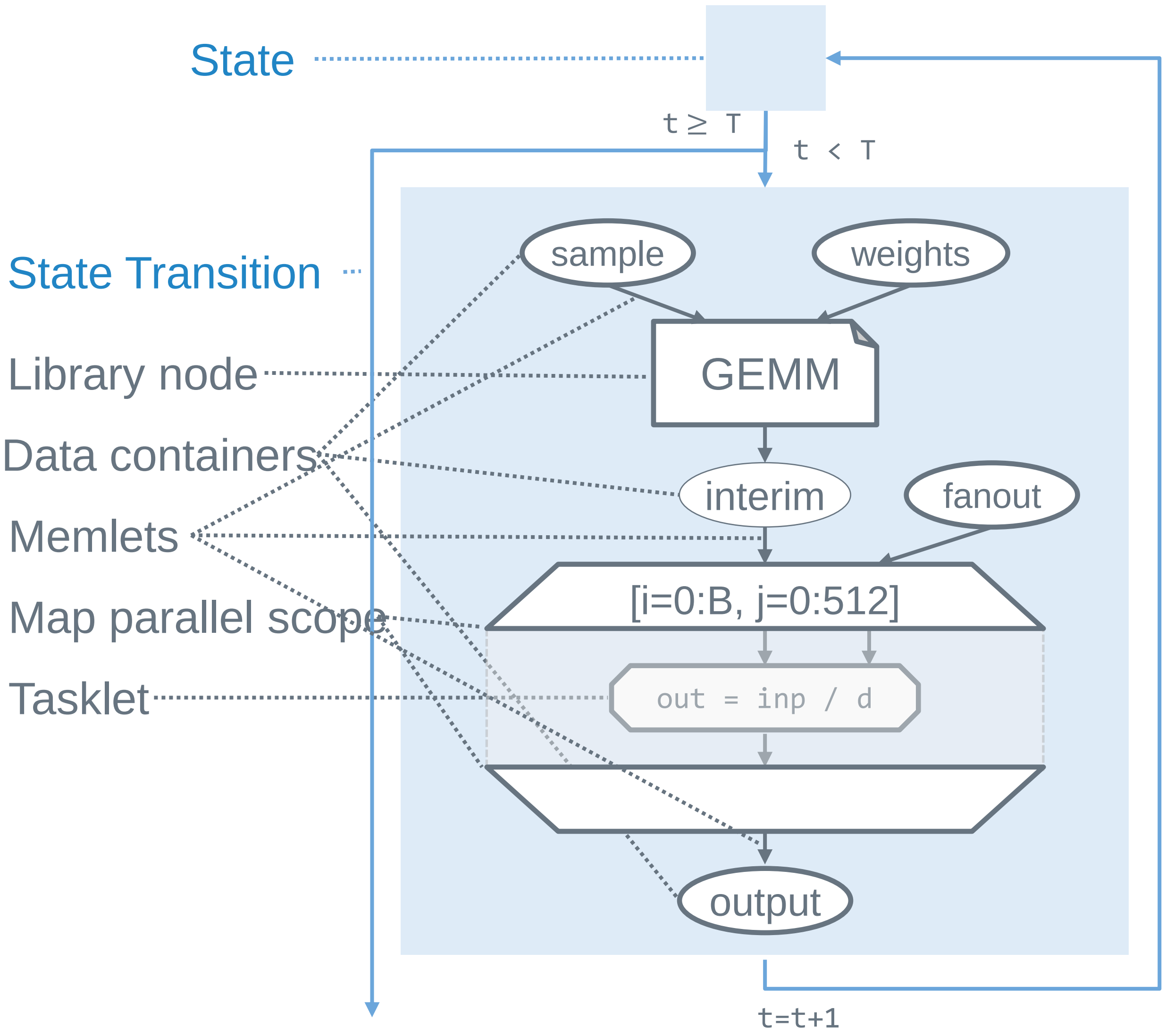
CPU Binary

GPU Binary

FPGA Modules

Data-Centric Intermediate Representation

1. Separate data containers from computation
2. Coarsening: multi-level view of data movement
3. Data movement as a first-class citizen (dependencies → program order)
4. Control dependencies *only when dataflow is not implied*



Ben-Nun T., de Fine Licht J., Ziogas A.N, Schneider T. and Hoefler T. Stateful Dataflow Multigraphs: A Data-Centric Model for Performance Portability on Heterogeneous Architectures. In SC 2019

From source code to SDFG

Code

```
glob_b = ...
class ClassA:
    def __init__(self, arr):
        self.q = arr

    @dace.method
    def __call__(self, a):
        return a * self.q + glob_b
```

From source code to SDFG

Code	Preprocess

```python
glob_b = ...
class ClassA:
    def __init__(self, arr):
        self.q = arr

    @dace.method
    def __call__(self, a):
        return a * self.q + glob_b
```

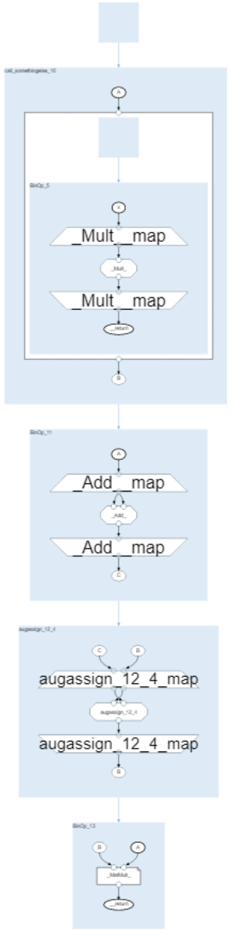
```python
@dace.program
def ClassA__call__(a, __g_self_q, __g_glob_b):
    return a * __g_self_q + __g_glob_b
```

Closure Resolution → Constant Folding → Conditional/Dead Code Elimination → Call Tree Evaluation → Array Consolidation

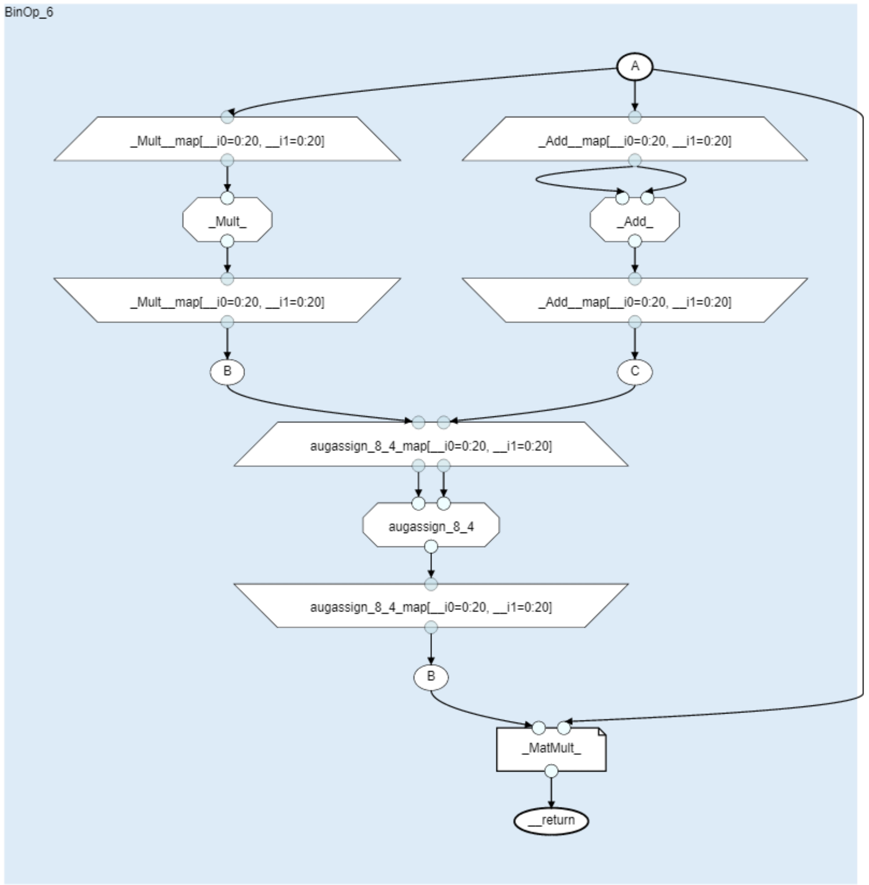
Slide courtesy Tal Ben-Nun

From source code to SDFG

Code | **Preprocess** | **Simplify**



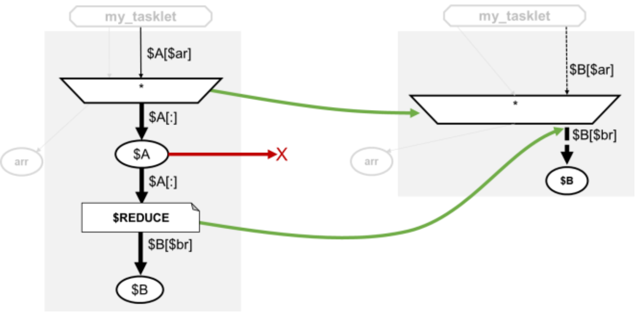
Control flow coarsening passes



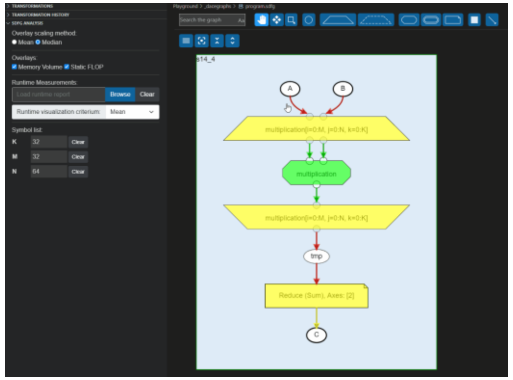
Slide courtesy Tal Ben-Nun

From source code to SDFG

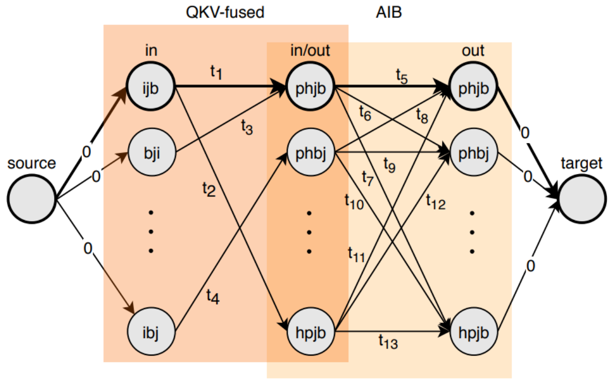
Code → **Preprocess** → **Simplify** → **Optimize**



Graph Rewriting Transformations



Interactive Transformation and Instrumentation



Local and Global Tuning Interface

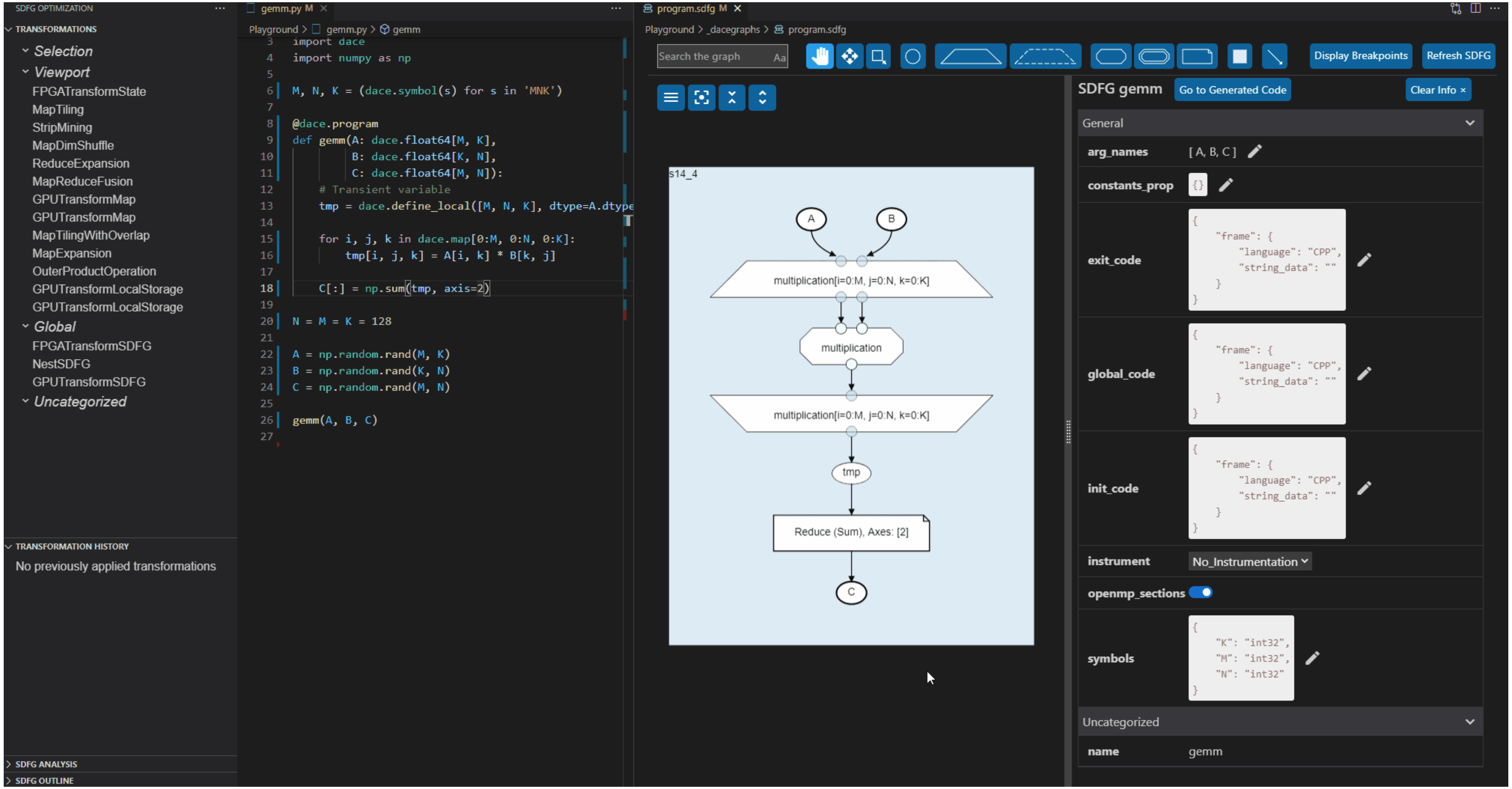
```python
class ClassA:
    def __init__(self, arr):
        self.q = arr

    @dace.method(auto_optimize=True, device=dace.DeviceType.GPU)
    def __call__(self, a):
        return a * self.q + glob_b
```

Automatic Optimization Heuristics

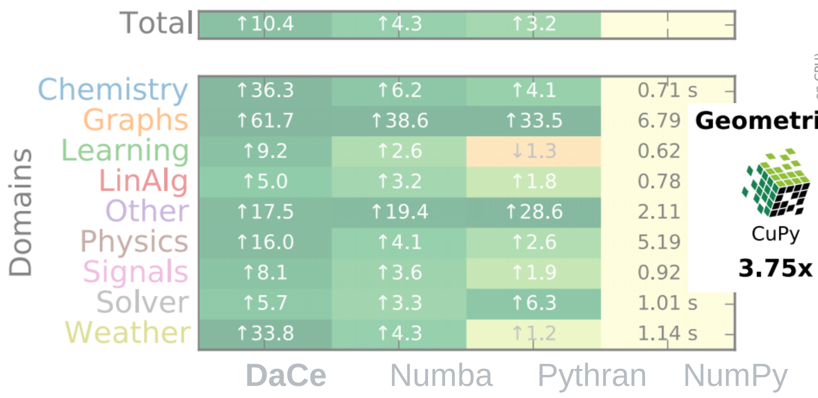
Slide courtesy Tal Ben-Nun

Visual Studio Code Integration

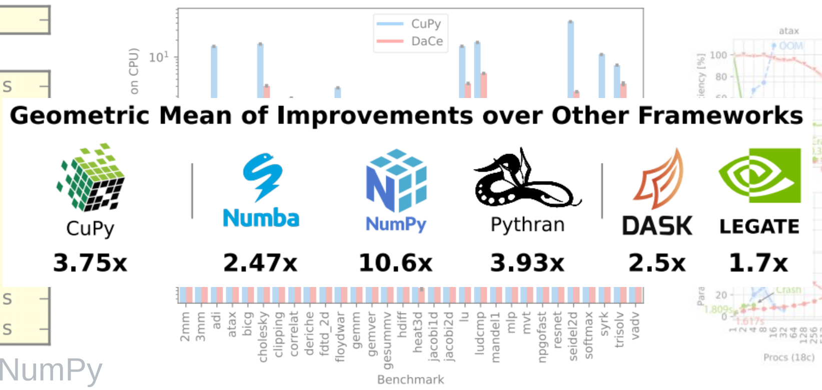


DaCe Performance

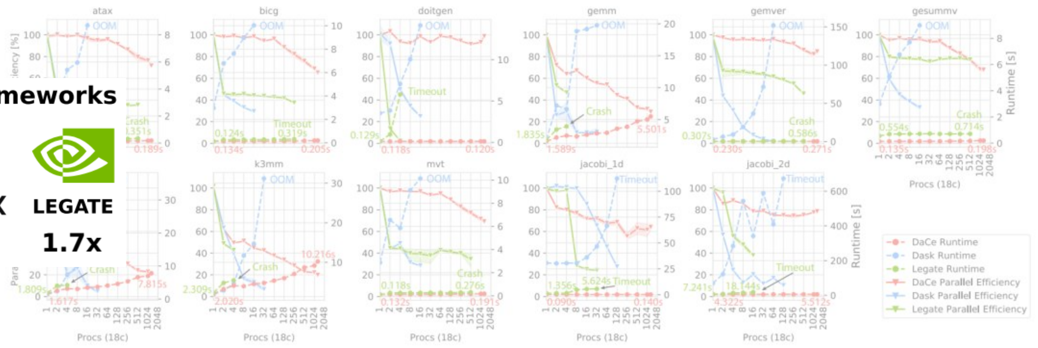
https://github.com/spcl/npbench



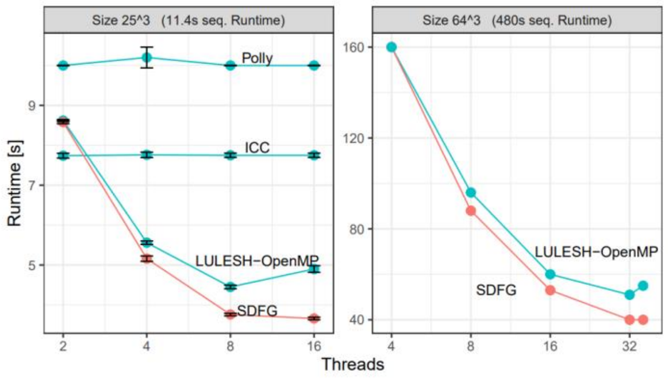
NumPyBench



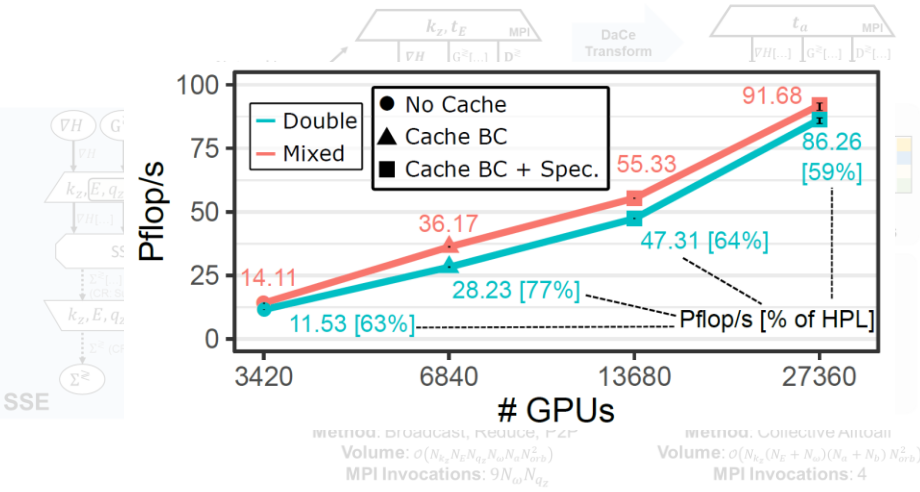
NumPyBench GPU



NumPyBench Distributed



Unstructured Hydrodynamics (LULESH)



Quantum Transport Simulation

A. N. Ziogas, T. Schneider, T. Ben-Nun, A. Calotoiu, T. De Matteis, J. de Fine Licht, L. Lavarini, T. Hoefler. “Productivity, Portability, Performance: Data-Centric Python”. SC ‘21.
A. Calotoiu, T. Ben-Nun, G. Kwasniewski, J. de Fine Licht, T. Schneider, P. Schaad, T. Hoefler. “Lifting C semantics for Dataflow Optimization ”. ICS ‘22.
A .Ziogas, et al. A data-centric approach to extreme-scale ab initio dissipative quantum transport simulations”. SC’19, **Gordon Bell Award**

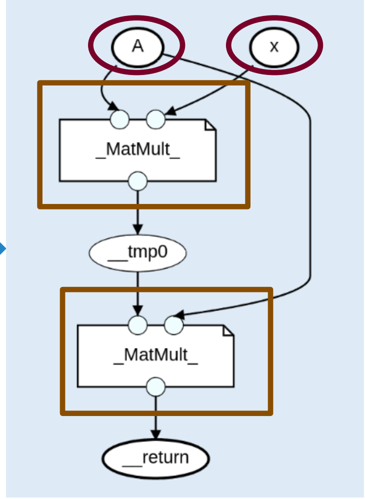
DaCe and FPGA

Let's use DaCe to compute $y = A^T Ax$ (ATAX)

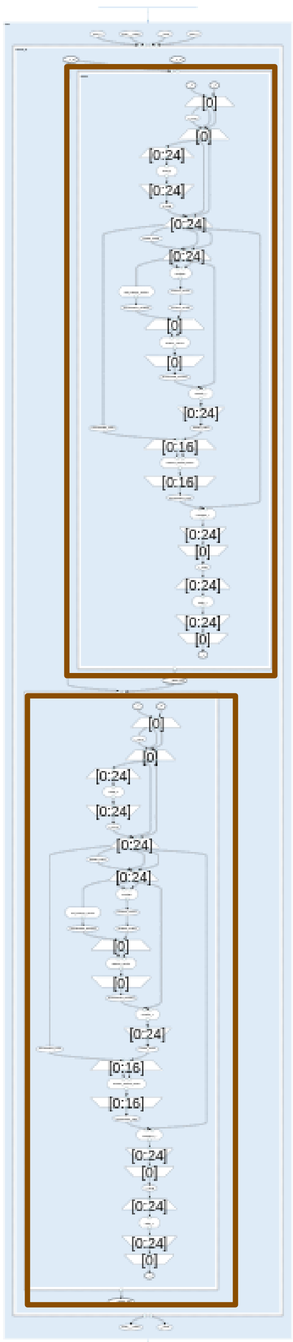
```python
import dace

M, N  = 24, 24
@dace.program
def atax(A: dace.float32[M, N],
                    x:dace.float32[N]):
    return (A @ x)@A
```

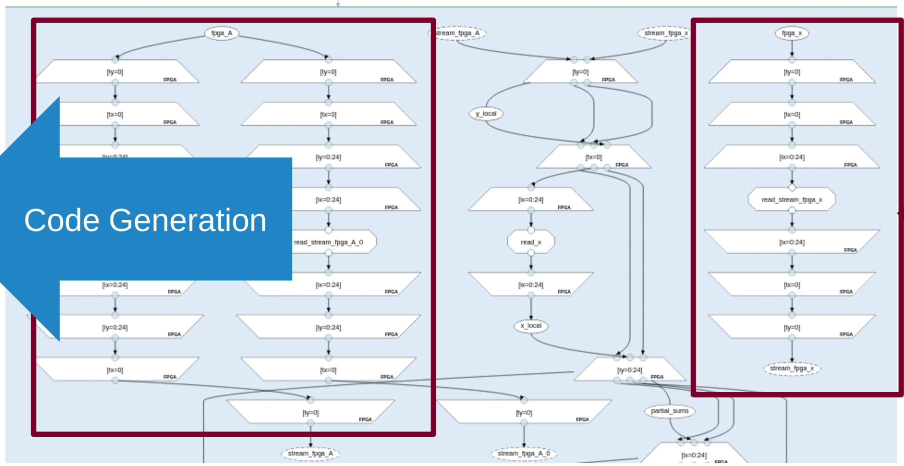
SDFG Parsing and Simplification



Specialization



Streaming Memory



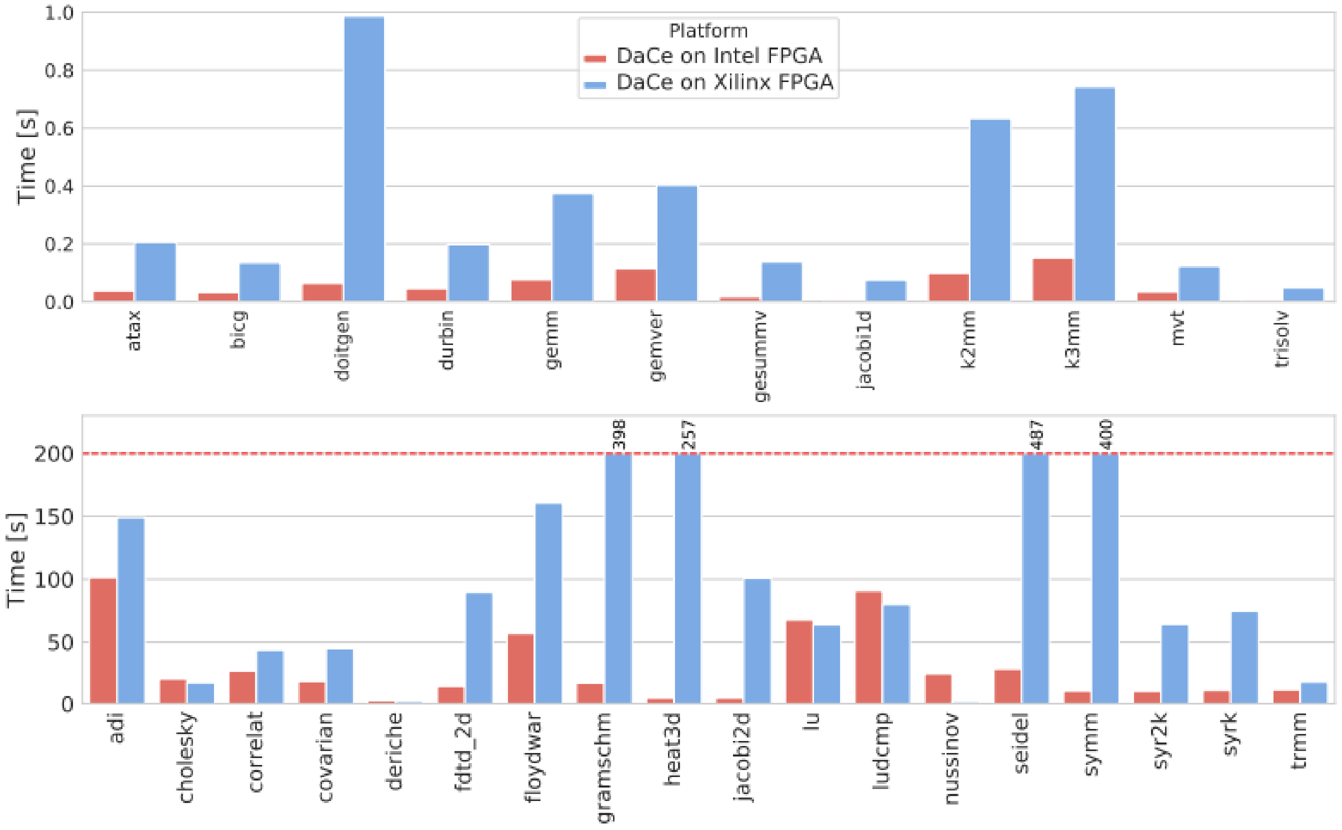
Code Generation

```c
void _MatMult_gemv_sdfg_1_0_3(const
float* _A, const float* _x, float* _y,
int M, int N) {
  #pragma HLS INLINE
  for (int _o0 = 0; _o0 < M; _o0 += 1) {
    #pragma HLS PIPELINE II=1
    #pragma HLS LOOP_FLATTEN
    float out = 0
     …

//450+ lines of code + Make files
```

Apply other optimizations

DaCe and FPGA

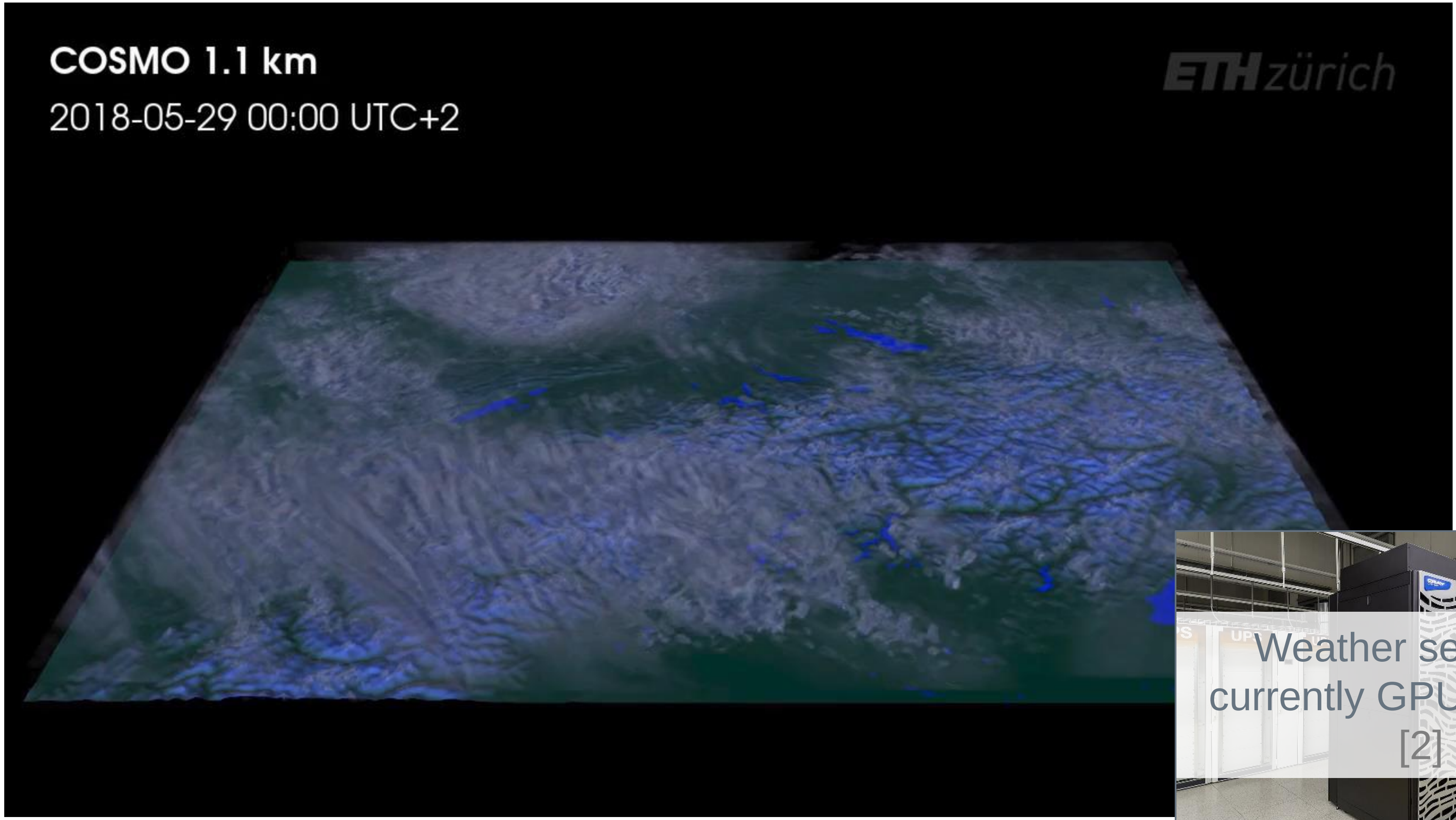


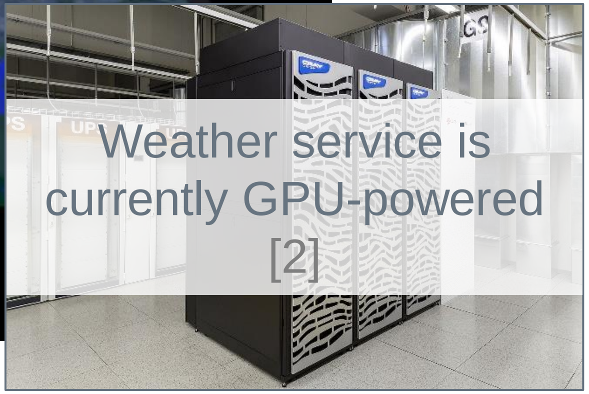
Polybench kernels code generated from Numpy for both Xilinx and Intel

A. N. Ziogas, T. Schneider, T. Ben-Nun, A. Calotoiu, T. De Matteis, J. de Fine Licht, L. Lavarini, T. Hoefler. "Productivity, Portability, Performance: Data-Centric Python". SC '21.

Weather simulation at MeteoSwiss

[1]

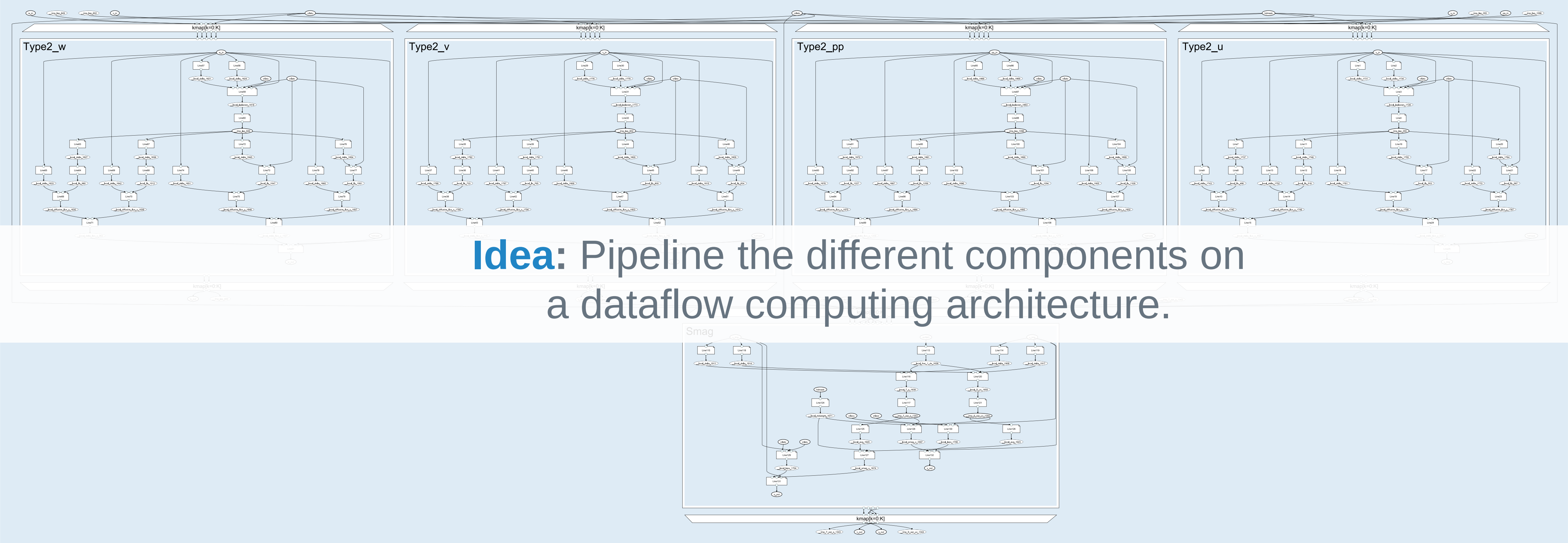




Weather service is currently GPU-powered [2]

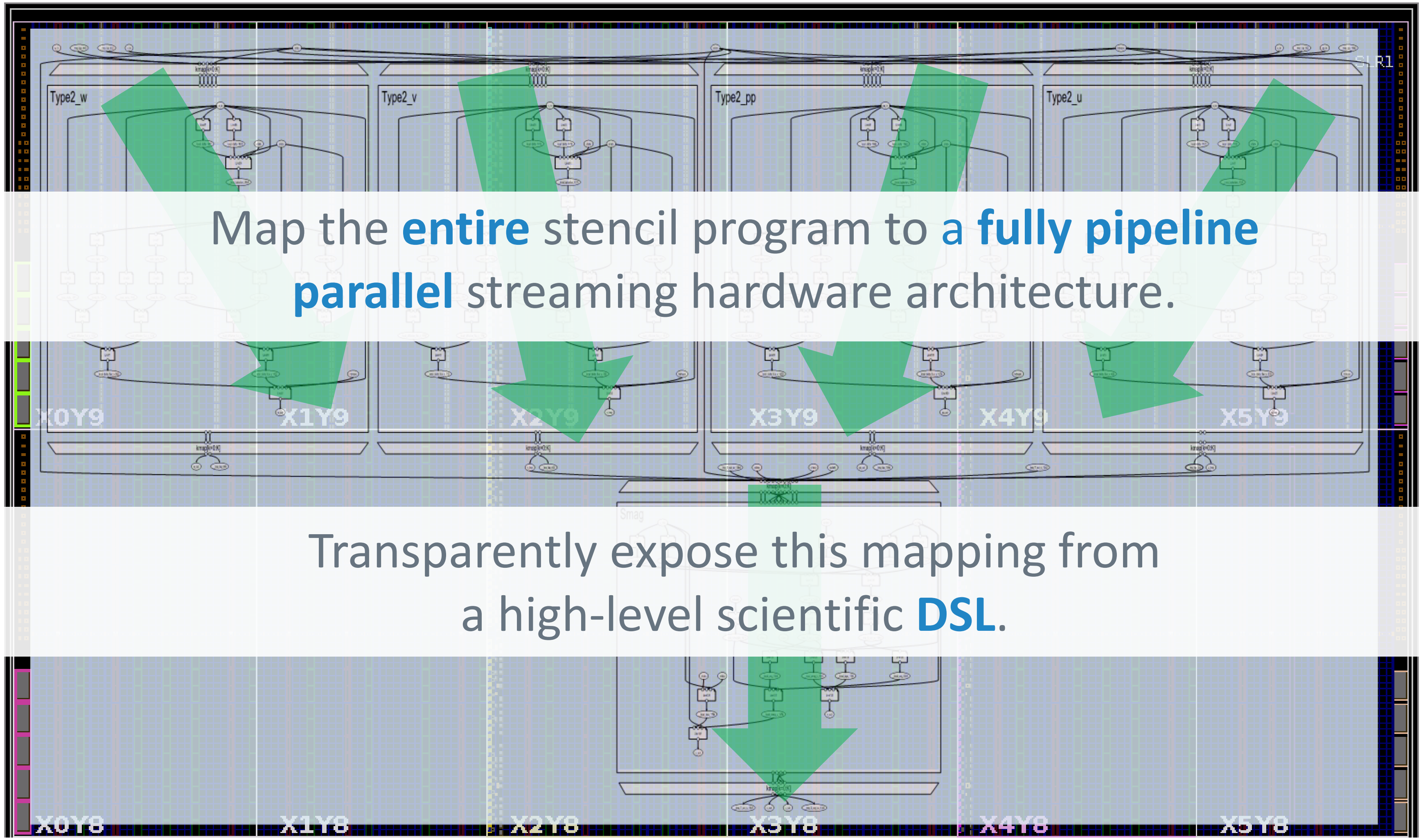
[1] Institute for Atmospheric and Climate Science and Computer Graphics Laboratory, ETH Zürich [https://vimeo.com/389292423]
[2] Swiss National Supercomputing Center (CSCS) [https://www.cscs.ch/computers/arolla-tsa-meteoswiss]

Weather programs



Idea: Pipeline the different components on a dataflow computing architecture.

StencilFlow – FPGAs for Climate Modeling



Map the **entire** stencil program to a **fully pipeline parallel** streaming hardware architecture.

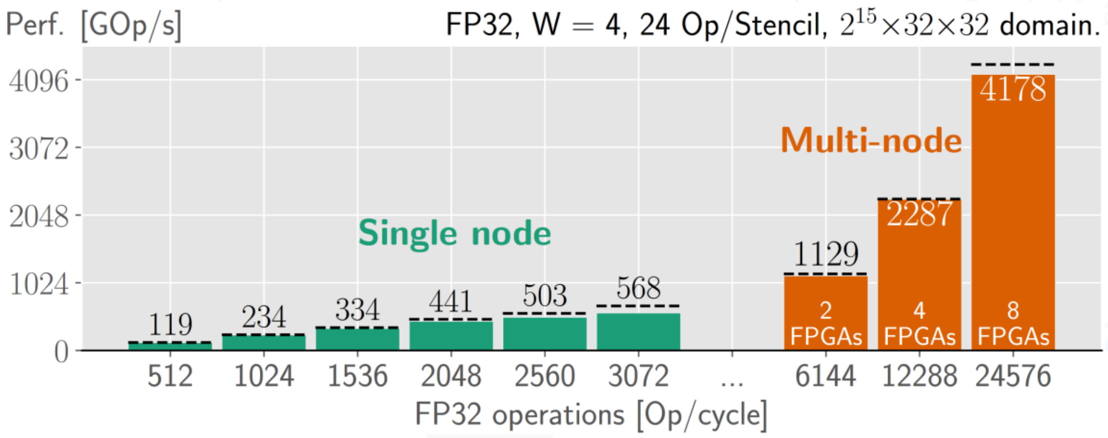
Transparently expose this mapping from a high-level scientific **DSL**.

J. de Fine Licht, A. Kuster, T. De Matteis, T. Ben-Nun, D. Hofer, T. Hoefler. "StencilFlow: Mapping Large Stencil Programs to Distributed Spatial Computing Systems". CGO'21

Stencilflow

J. de Fine Licht, A. Kuster, T. De Matteis, T. Ben-Nun, D. Hofer, T. Hoefler. "StencilFlow: Mapping Large Stencil Programs to Distributed Spatial Computing Systems". CGO'21

Stencilflow



Iterative stencils (e.g., Jacobi, Diffusion, ...) on multiple FPGA devices using SMI

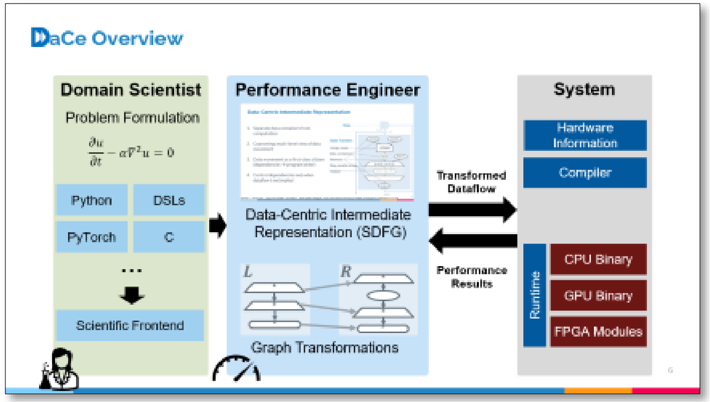
	Runtime	Performance	Peak BW.	%Roof.
Stratix 10	1,178 µs	145 GOp/s	77 GB/s	52%
Stratix 10*	332 µs	513 GOp/s	∞ GB/s	–
Xeon 12C	5,270 µs	32 GOp/s	68 GB/s	13%
P100	810 µs	210 GOp/s	732 GB/s	8%
V100	201 µs	849 GOp/s	900 GB/s	26%

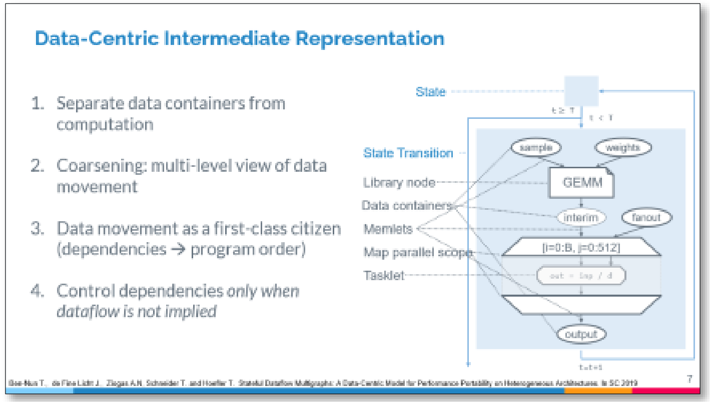
*Without memory bandwidth constraints.

Stencil from Weather Prediction models (Horizontal Diffusion)

J. de Fine Licht, A. Kuster, T. De Matteis, T. Ben-Nun, D. Hofer, T. Hoefler. “StencilFlow: Mapping Large Stencil Programs to Distributed Spatial Computing Systems”. CGO’21

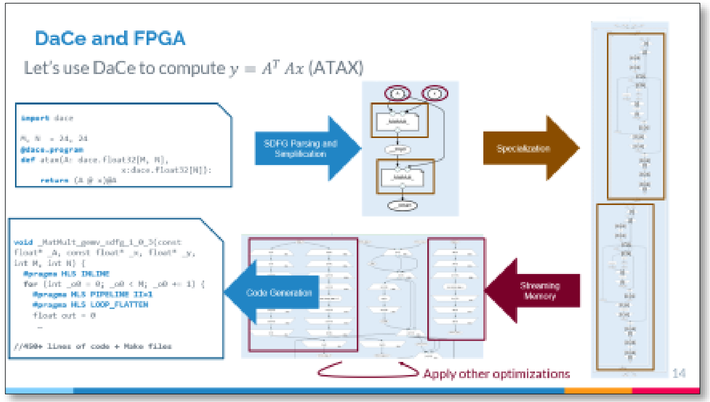
Conclusion

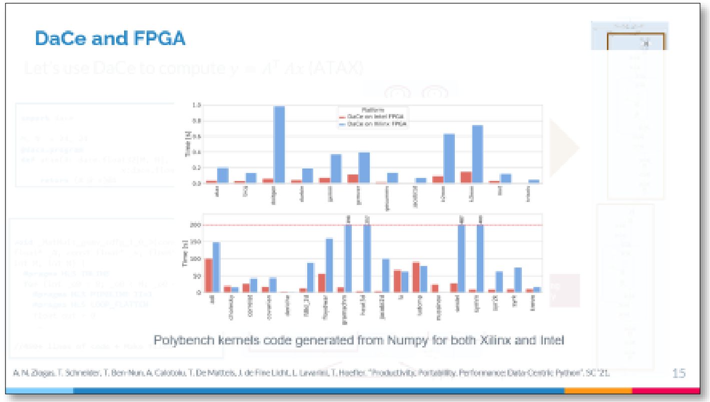


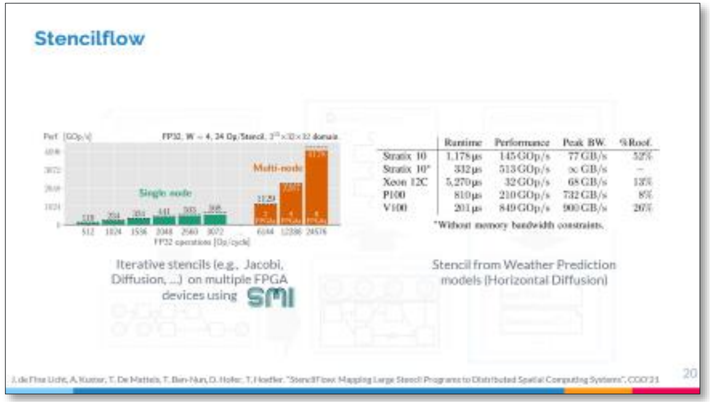


DaCe is in active development.
If you want to know more/contribute:

github.com/spcl/dace







t.de.matteis@vu.nl