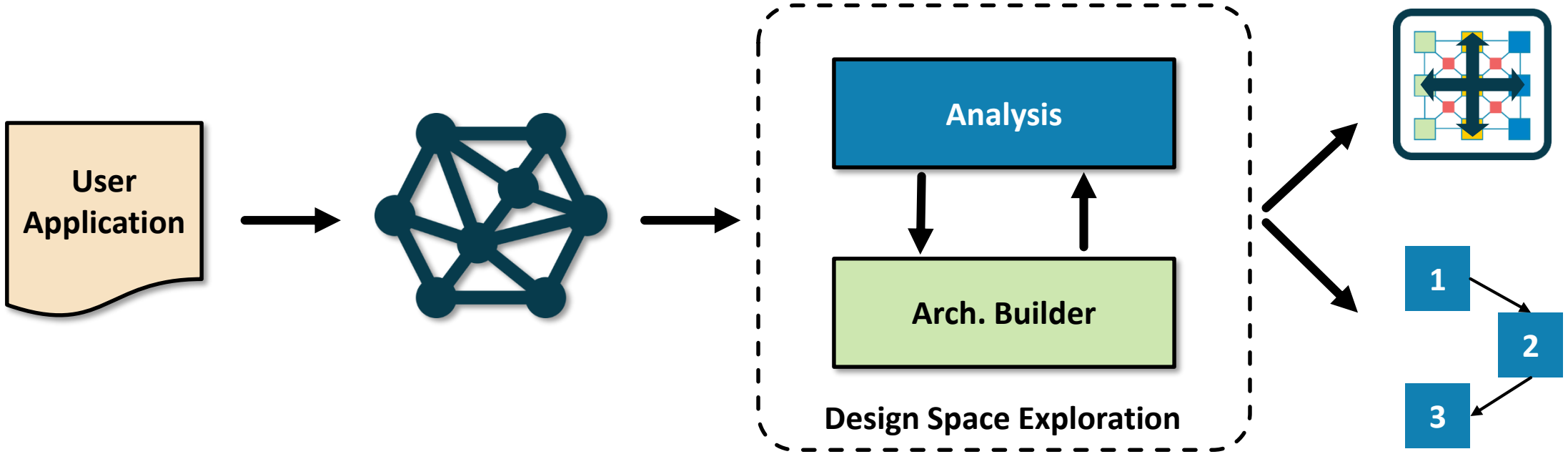
ASA: Application-Specific Architecture

Tiziano De Matteis, Torsten Hoefler

Motivations

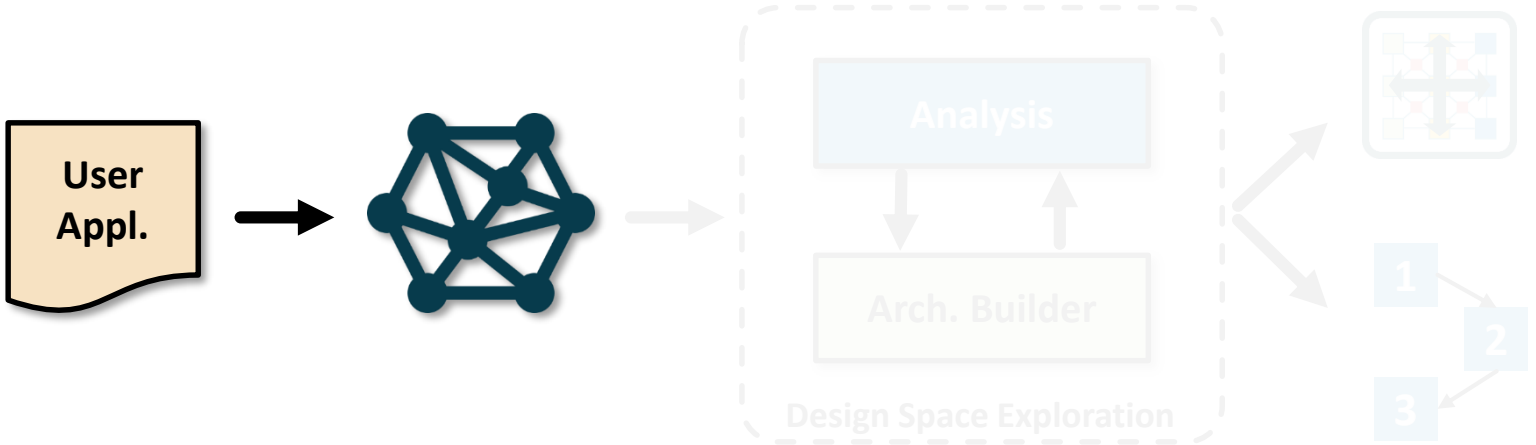
Instead of having an application being adapted and optimized for a given architecture (DSA or SoC), **how we can build a Spatial Specialized Architecture to efficiently support a given application? How we can map the application on such architecture?**

EFCL Proposal



SDRs and Deep Learning as driving use cases

First steps



To facilitate further analysis, we want to use data-flow languages to represent the application:

- that naturally describe the **business logic** of the application
- and its **data movements**

We want to exploit **both** these types of information → DaCe

DaCe Overview

Domain Scientist

Problem Formulation

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

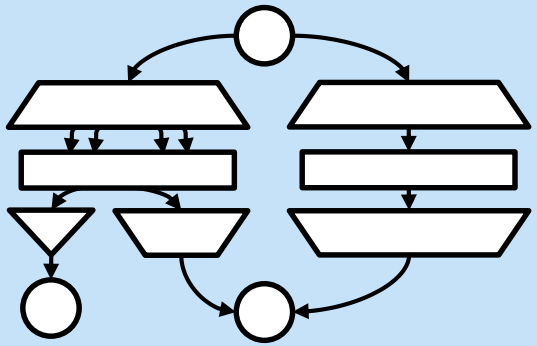
python

NumPy ONNX

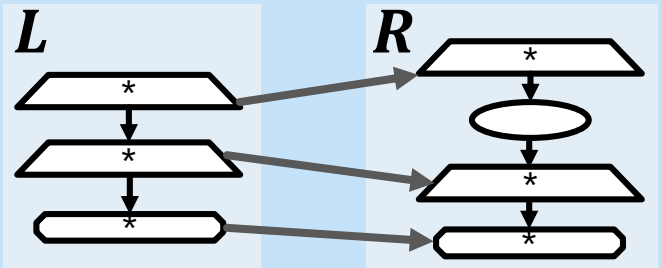
...

Scientific Frontend

Performance Engineer



Data-Centric Intermediate Representation (SDFG)



Graph Transformations

Transformed Dataflow

Performance Results

System

Hardware Information

Compiler

Runtime

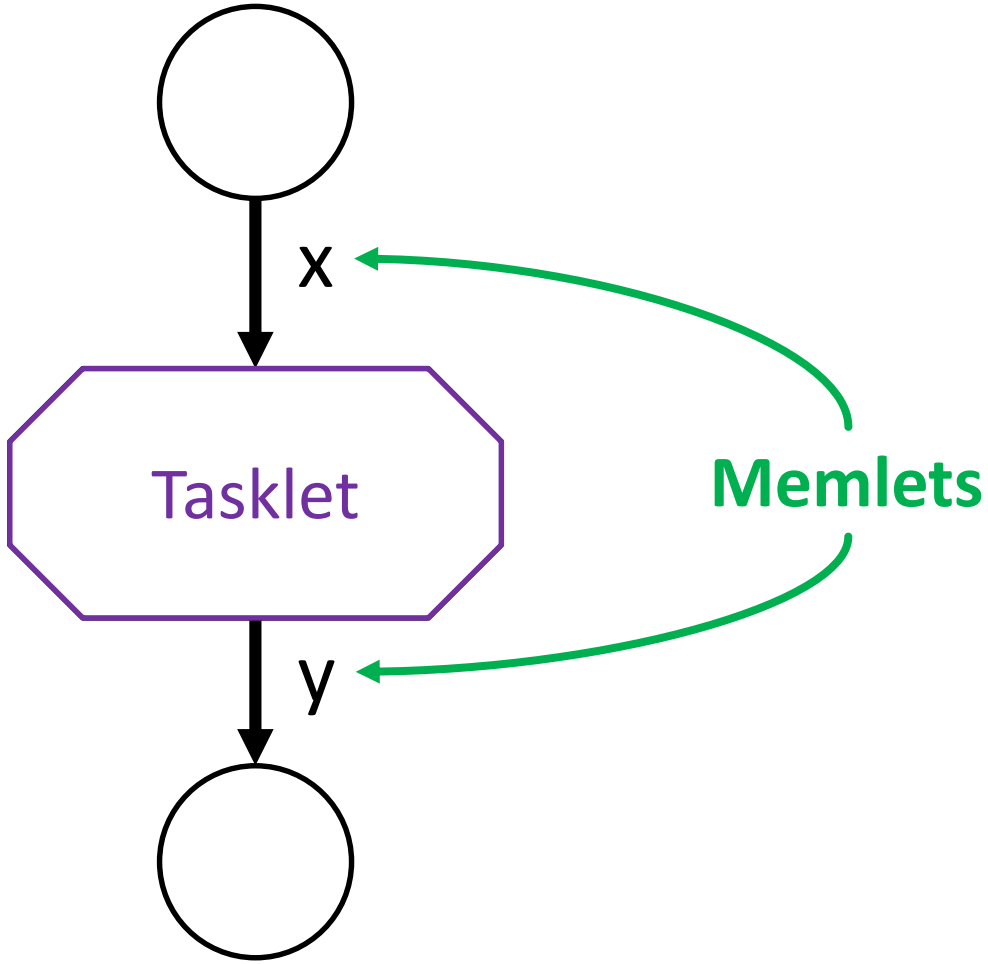
CPU Binary

GPU Binary

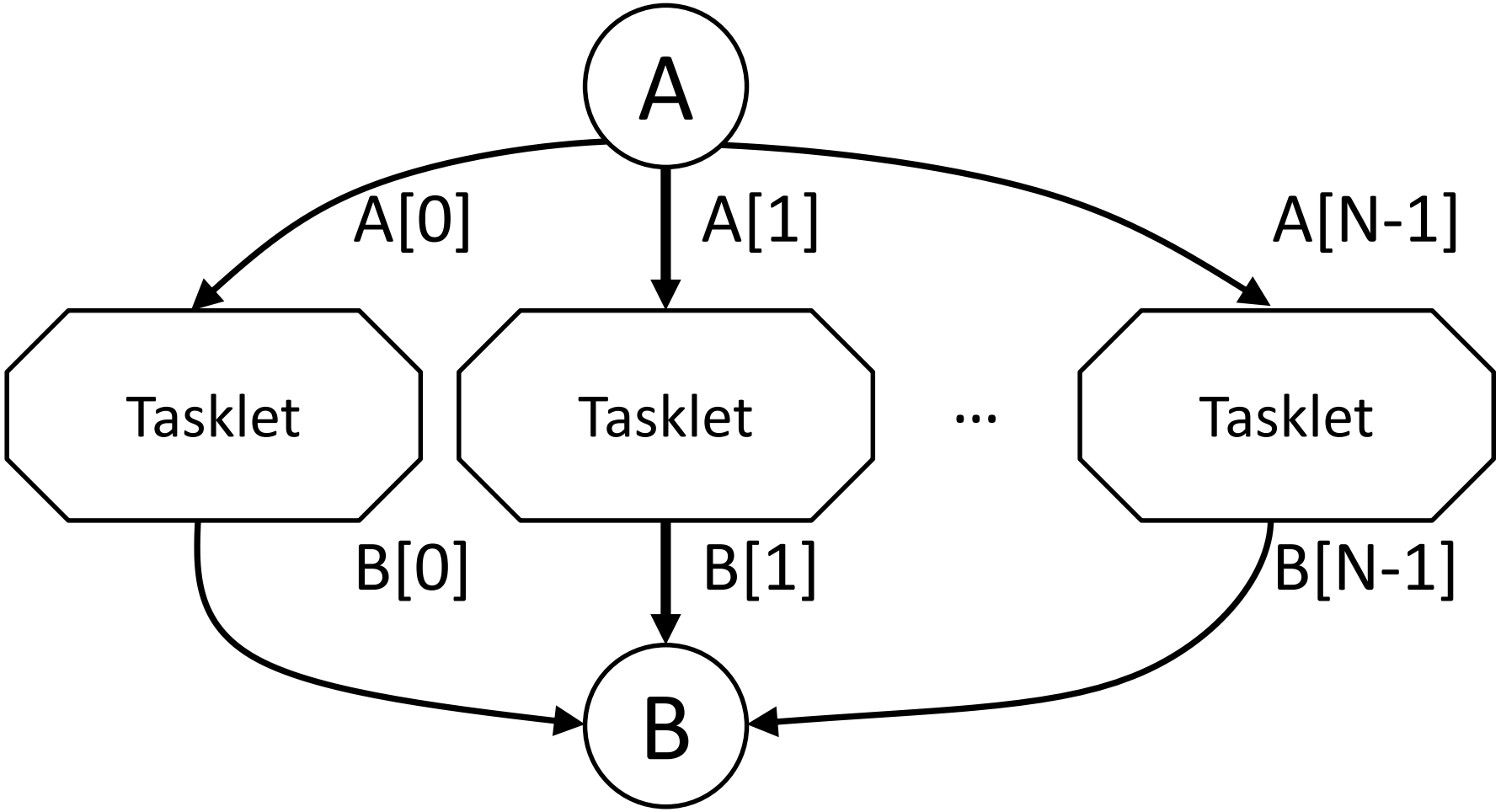
FPGA Modules

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

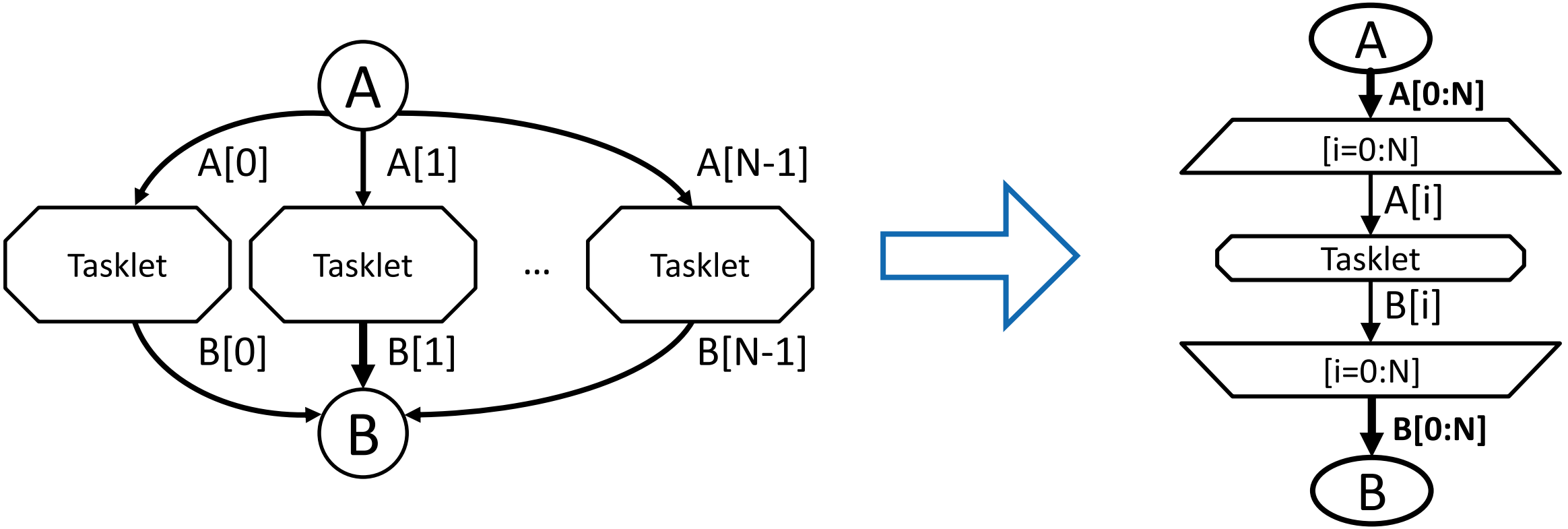
Dataflow Programming in DaCe



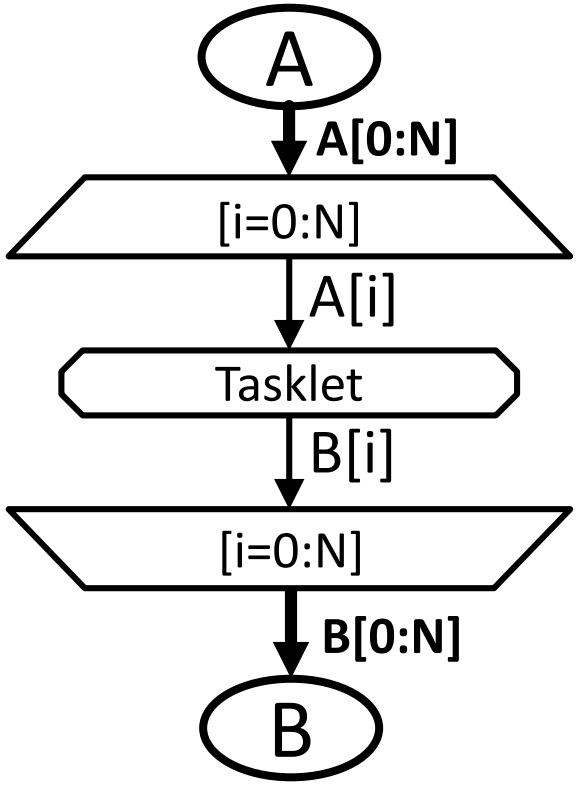
Parallel Dataflow Programming

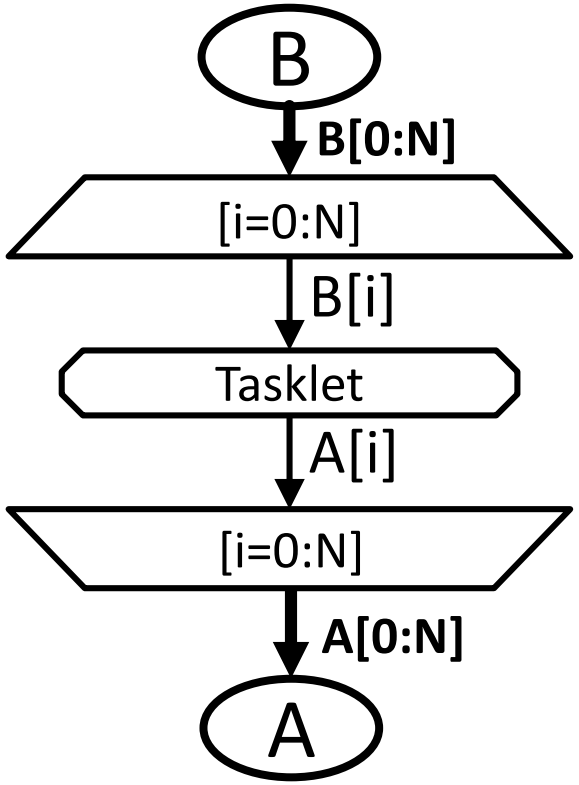


Parallel Dataflow Programming

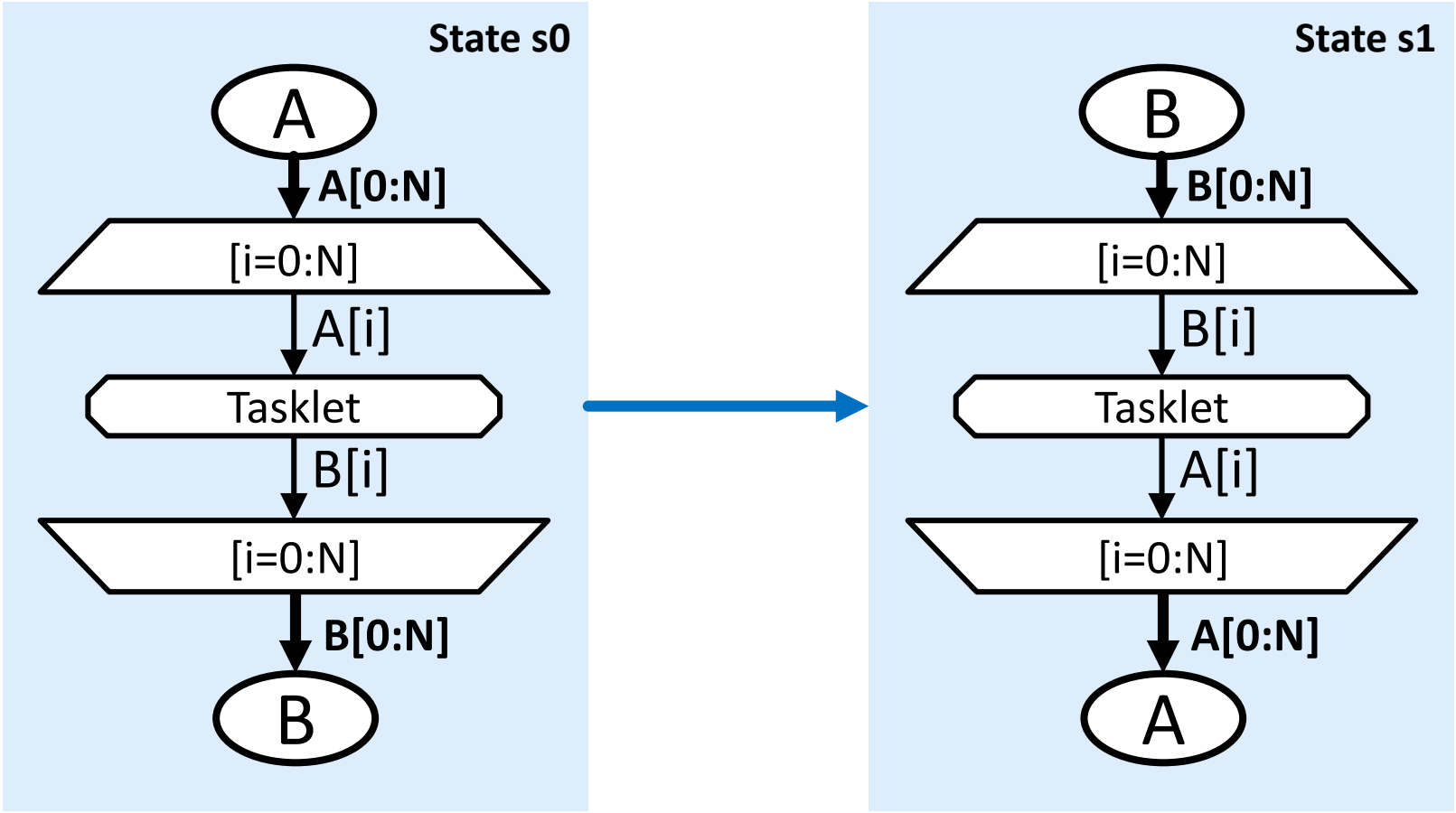


Stateful Dataflow Parallel Programming in DaCe





Stateful Dataflow Parallel Programming in DaCe



Meet the Nodes

State

State machine elem

Tasklet

Nested SDFG

Fine-grained compu

Array

N-dimensional data

Map

Exit

Parametric graph al

Stream

Streaming data con

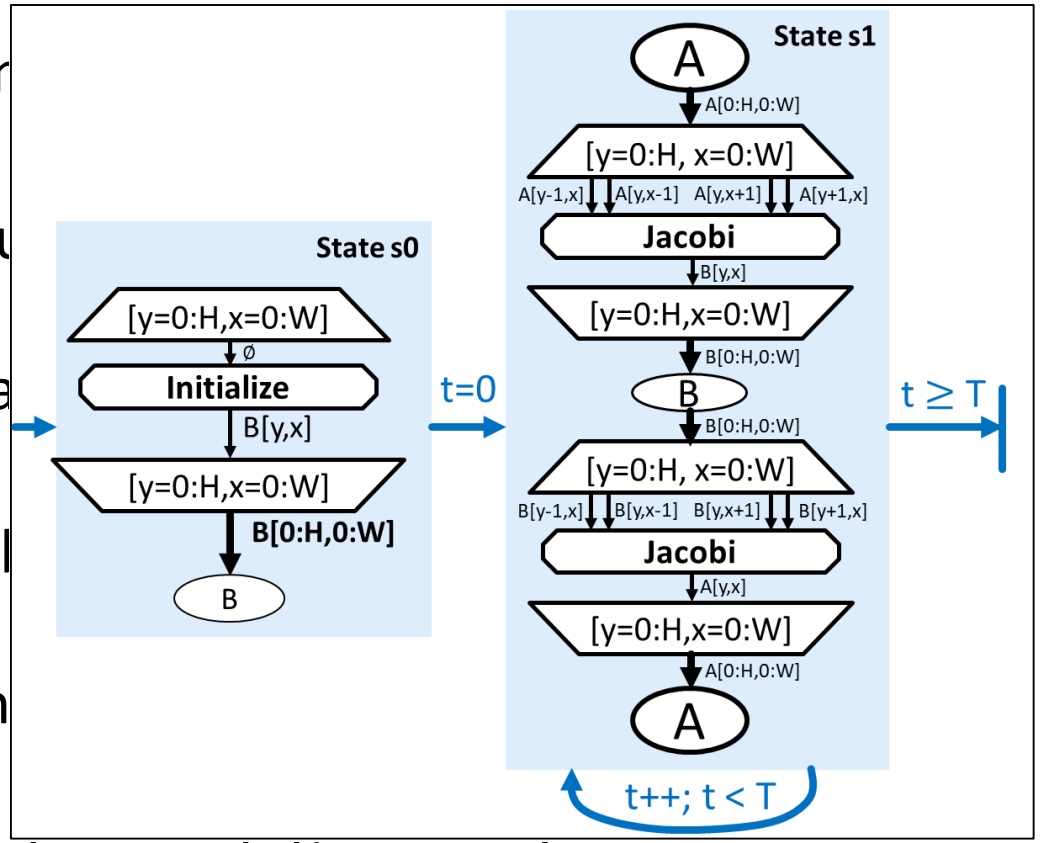
Consume

Exit

Dynamic mapping of computations on streams

Conflict Resolution

Defines behavior during conflicting writes

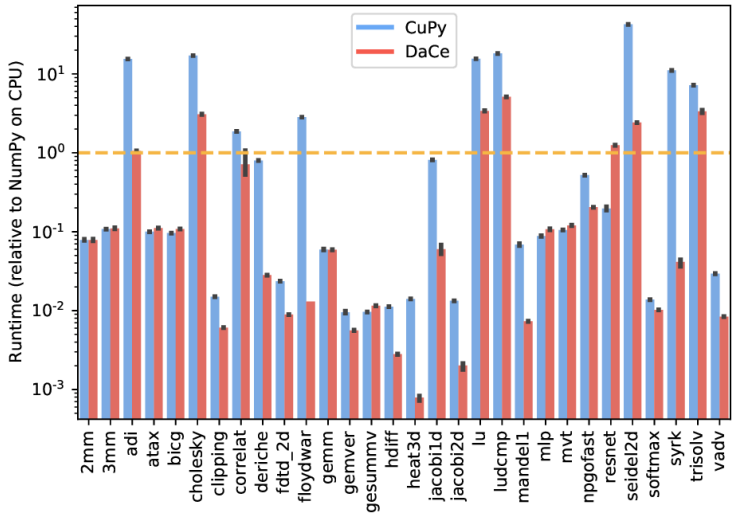


DaCe – Multi Backend Code Generation

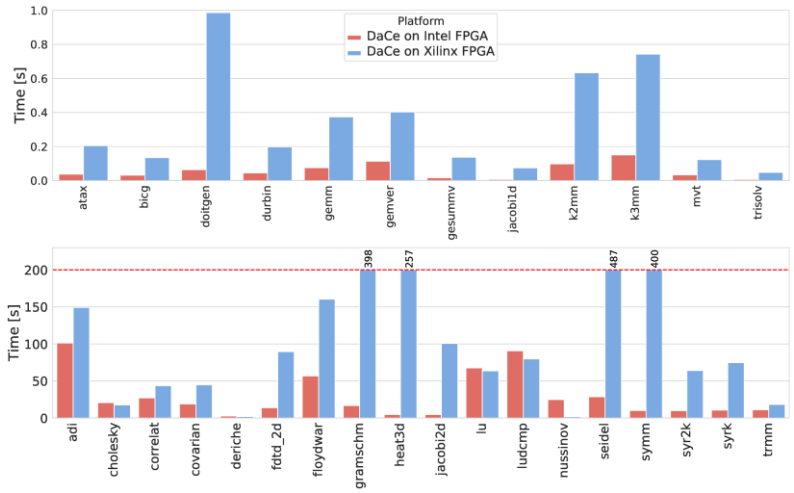
CPU Results

	DaCe	GCC	ICC	Numba	Pythran	NumPy
gramschm	↑9.1[2]	↑9.3	↑7.2[54]	↑3.0	↑8.1	0.15 s
heat3d	↑454[42]	↑24.0[9]	↑179[7]	↑50.1	↑2.3	50.35 s
jacobi1d	↑11.4[1]	↑3.7[8]	↑3.9[13]	↓1.2	↑3.1	0.45 s
jacobi2d	↑56.2	↑7.1[28]	↑58.6[7]	↑18.2	↑21.8	174.61 s
lu	↑3.6	↑2.9	↑4.5	↑2.2	↑4.8	13.02 s
ludcmp	↑3.7	↑5.4	↑5.4	↑2.3	↑4.9	13.7 s[1]
syr2k	↑10.4	↑10.5	↑287[14]	↑5.3	↑5.9	13.94 s
nussinov	↑612	↑1.1k	↑909[18]	↑420	↑871	20.37 s[2]
seidel2d	↑185	↑158	↑78.1[2]	↑95.8	↑141	15.87 s[1]
symm	↑4.2	↑6.4	↑74.9[19]	↑11.5	↑4.2	9.88 s
syrk	↑10.0	↑9.3	↑233[28]	↑3.0	↑5.5	6.58 s
gesummv	↑8.6[2]	↑1.8[6]	↑1.8[3]	↑7.5[5]	↑1.0	0.77 s
mvt	↑1.0	↓52.6[23]	↓13.9[4]	↑1.0[1]	↓1.0[1]	45.22 ms[1]
gemver	↑19.1[13]	↑1.5[3]	↑2.5[1]	↑2.1[3]	↑1.3[5]	0.79 s[5]
atax	↓1.2[7]	↓12.0[2]	↓6.2[6]	↓1.0[5]	↓1.1[8]	73.37 ms[5]
floydwar	↑116[2]	↑3.9[5]	↑2.0	↑57.3	↑7.3	84.0 s
fdtd_2d	↑170[98]	↑3.7[11]	↑41.3[24]	↑4.1	↑1.3	7.4 s
durbin	↑2.2	↑5.0[1]	↑5.9[10]	↓1.1[2]	↑3.2	0.65 s
doitgen	↑39.7[46]	↓11.8	↓2.3[3]	↑1.1	1.0[3]	0.47 s[5]
deriche	↑17.3	↑2.7[6]	↑55.1[9]	↑1.5	↑1.6	2.83 s
covarian	↑1.6[16]	↓21.4	↓5.0[6]	↑1.3[2]	↓20.4[3]	80.11 ms
correlat	↑1.6	↓27.8[2]	↓6.6[4]	↓1.1[9]	1.0[19]	61.55 ms[19]
cholesky	↑12.1	↑3.7	↑3.4	↑10.3	↑15.2	7.05 s
bicg	↓1.1[8]	↓17.2[2]	↓17.5[1]	↓1.1[7]	↓1.1[8]	75.92 ms[9]
trisolv	↑2.3[14]	↓1.6[2]	↓1.8[4]	↑1.9[23]	↑1.3	0.13 s[14]
adi	↑16.5	↑9.1	↑13.7[6]	↑8.2	↑9.4	0.9 s
3mm	↑1.7[3]	↓1.7k[11]	↓1.1[2]	↑1.6[6]	↑1.2[33]	0.46 s[21]
2mm	↑2.4[2]	↓942[22]	↓17.1	↑1.5[10]	↑1.3[16]	0.42 s[12]
gemm	↑2.3[5]	↓133[22]	↓4.8[11]	↑1.4[7]	↑1.4	78.72 ms
trmm	↑36.2	↑4.1	↑50.9[18]	↑15.3[4]	↑4.2	4.12 s[1]

GPU Results



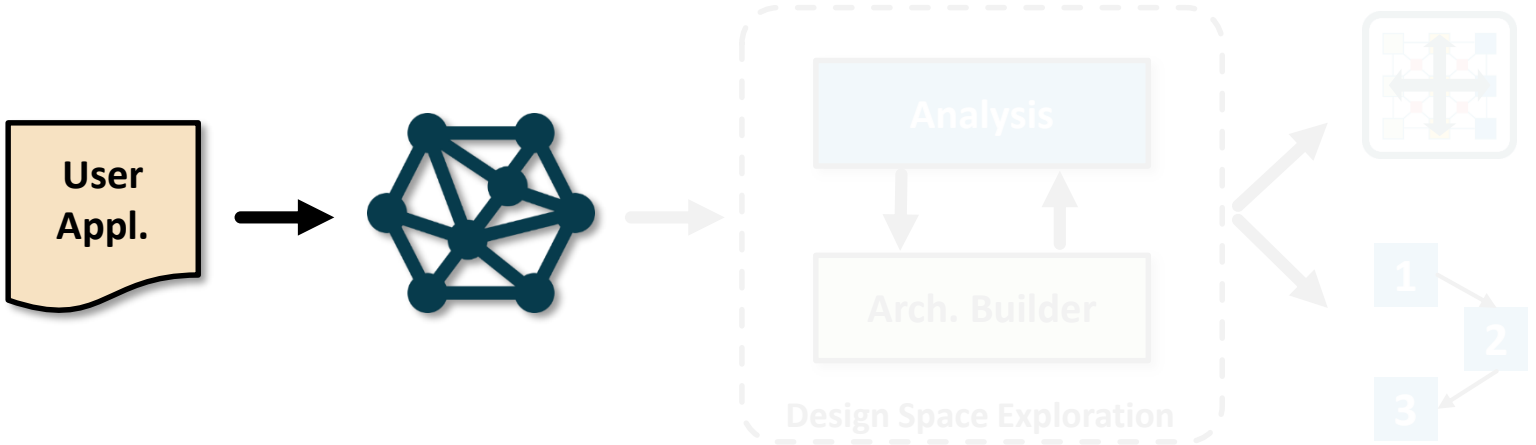
FPGA Results



Distributed Memory Results

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.

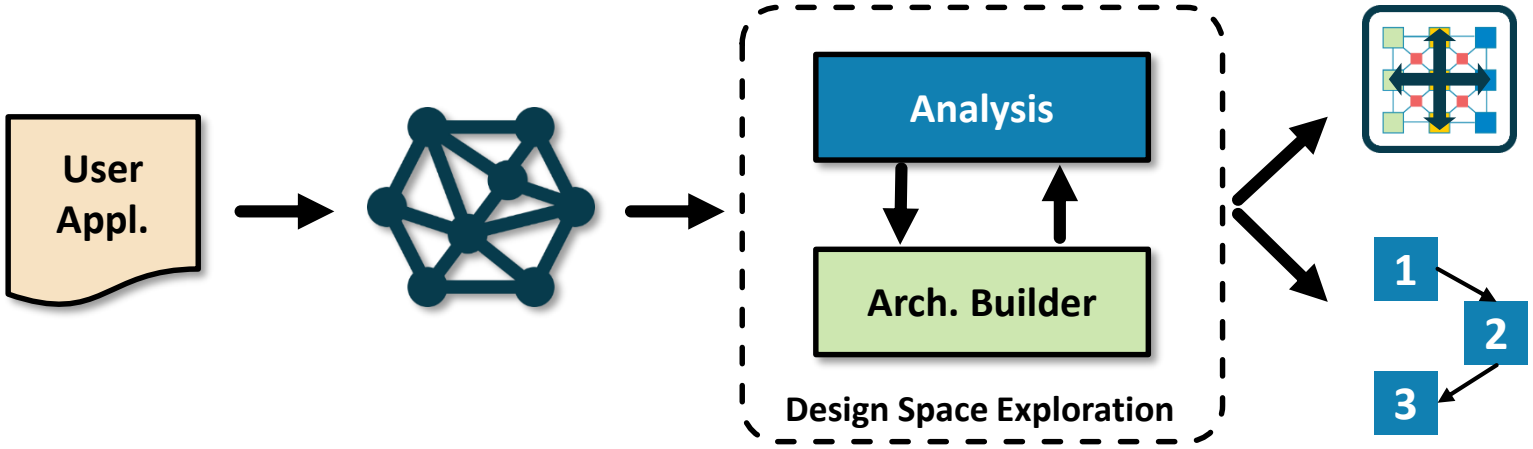
First steps



In this project we want to utilize DaCe to:

- exploit **its Data-Centric Intermediate Representation**
- (later on) use it as **“frontend”**, to let user express her own application using a high-level formalism

First steps



Understand how we can efficiently **map an application on a given architecture**:

- we want to consider **heterogeneity**
- we want to deal with architecture specific features, such as the presence of a **fast on-chip interconnect**

Your inputs regarding typical workloads/applications/use cases will be important

Thank you!

spcl.inf.ethz.ch
@spcl_eth

ETH zürich