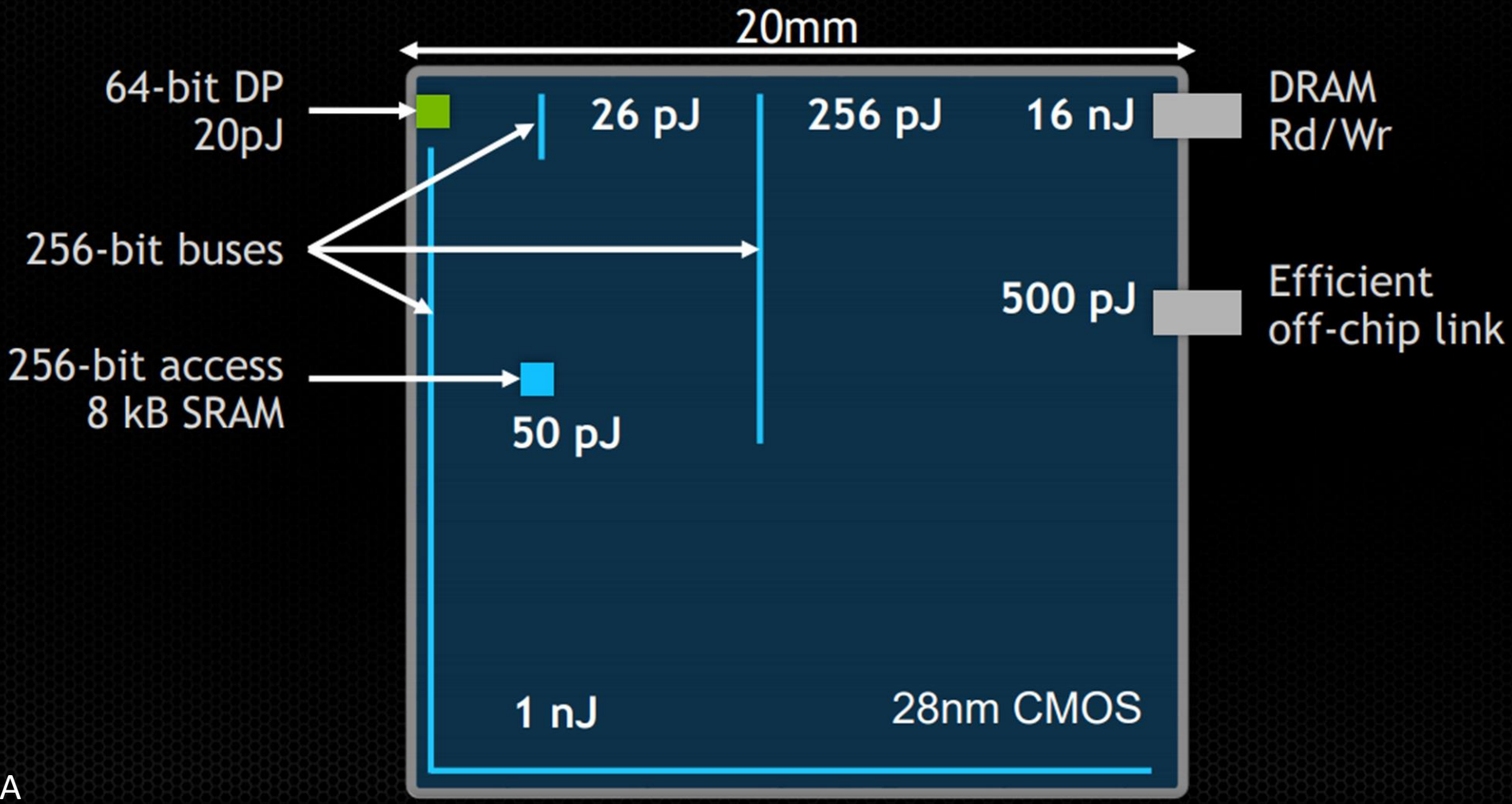
Data-Centric Python - Productivity, portability and all with high performance!

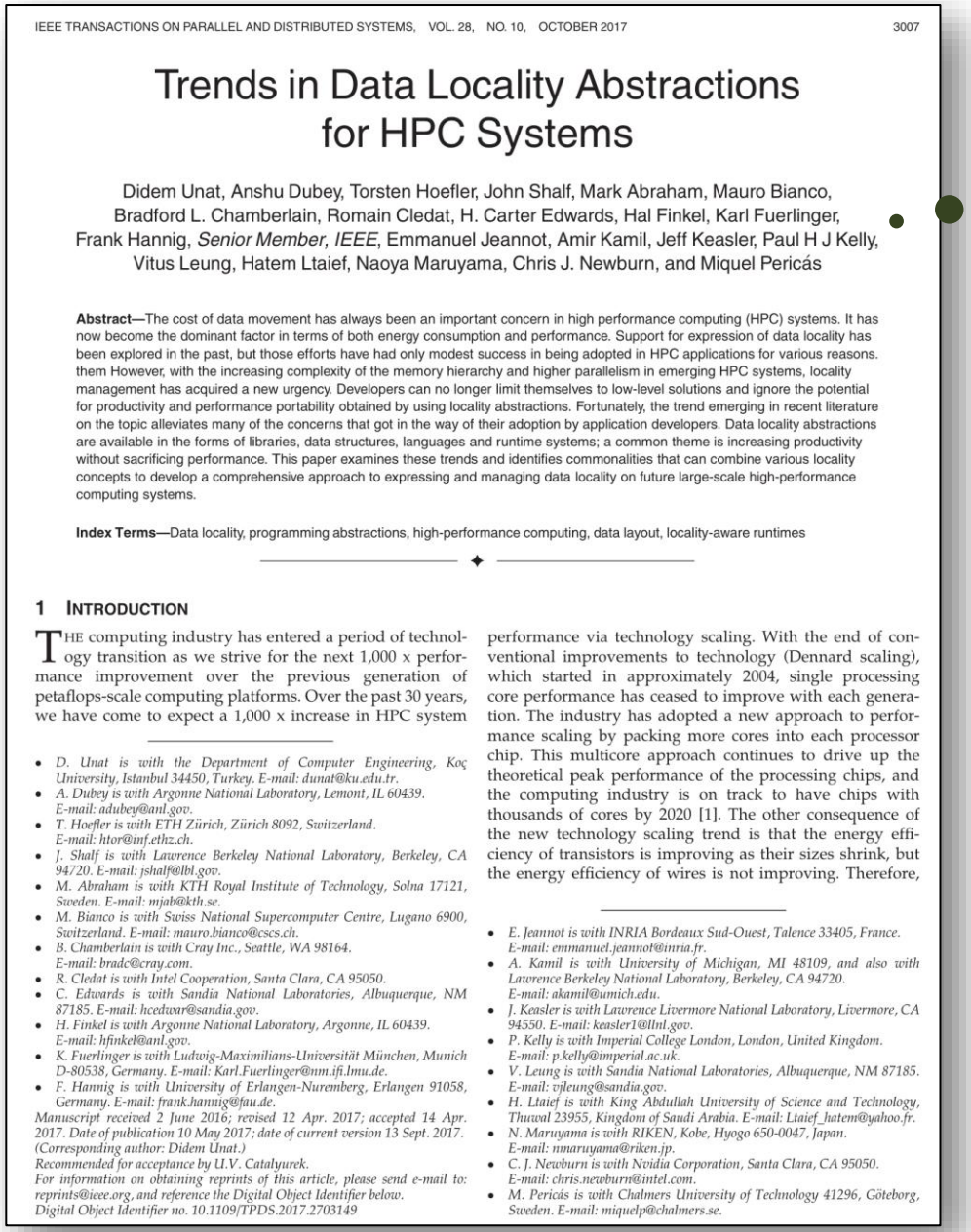
Torsten Hoefler, keynote at the Russian Supercomputing Days (virtual)

Tal Ben-Nun, Alexandros Ziogas, Johannes de Fine Licht, Tiziano de Matteis, Timo Schneider, Andreas Kuster, Manuel Burger, Philip Schaad, Dominic Hofer and the whole DAPP team @ SPCL

Communication Dominates Arithmetic



Courtesy NVIDIA



IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 28, NO. 10, OCTOBER 2017 3007

Trends in Data Locality Abstractions for HPC Systems

Didem Unat, Anshu Dubey, Torsten Hoefler, John Shalf, Mark Abraham, Mauro Bianco, Bradford L. Chamberlain, Romain Cledat, H. Carter Edwards, Hal Finkel, Karl Fuerlinger, Frank Hannig, *Senior Member, IEEE*, Emmanuel Jeannot, Amir Kamil, Jeff Keasler, Paul H J Kelly, Vitus Leung, Hatem Ltaief, Naoya Maruyama, Chris J. Newburn, and Miquel Pericás

Abstract—The cost of data movement has always been an important concern in high performance computing (HPC) systems. It has now become the dominant factor in terms of both energy consumption and performance. Support for expression of data locality has been explored in the past, but those efforts have had only modest success in being adopted in HPC applications for various reasons. them However, with the increasing complexity of the memory hierarchy and higher parallelism in emerging HPC systems, locality management has acquired a new urgency. Developers can no longer limit themselves to low-level solutions and ignore the potential for productivity and performance portability obtained by using locality abstractions. Fortunately, the trend emerging in recent literature on the topic alleviates many of the concerns that got in the way of their adoption by application developers. Data locality abstractions are available in the forms of libraries, data structures, languages and runtime systems; a common theme is increasing productivity without sacrificing performance. This paper examines these trends and identifies commonalities that can combine various locality concepts to develop a comprehensive approach to expressing and managing data locality on future large-scale high-performance computing systems.

Index Terms—Data locality, programming abstractions, high-performance computing, data layout, locality-aware runtimes

1 Introduction

The computing industry has entered a period of technology transition as we strive for the next 1,000 x performance improvement over the previous generation of petaflops-scale computing platforms. Over the past 30 years, we have come to expect a 1,000 x increase in HPC system performance via technology scaling. With the end of conventional improvements to technology (Dennard scaling), which started in approximately 2004, single processing core performance has ceased to improve with each generation. The industry has adopted a new approach to performance scaling by packing more cores into each processor chip. This multicore approach continues to drive up the theoretical peak performance of the processing chips, and the computing industry is on track to have chips with thousands of cores by 2020 [1]. The other consequence of the new technology scaling trend is that the energy efficiency of transistors is improving as their sizes shrink, but the energy efficiency of wires is not improving. Therefore,

- *D. Unat is with the Department of Computer Engineering, Koç University, Istanbul 34450, Turkey. E-mail: dunat@ku.edu.tr.*
- *A. Dubey is with Argonne National Laboratory, Lemont, IL 60439. E-mail: adubey@anl.gov.*
- *T. Hoefler is with ETH Zürich, Zürich 8092, Switzerland. E-mail: htor@inf.ethz.ch.*
- *J. Shalf is with Lawrence Berkeley National Laboratory, Berkeley, CA 94720. E-mail: jshalf@lbl.gov.*
- *M. Abraham is with KTH Royal Institute of Technology, Solna 17121, Sweden. E-mail: mjab@kth.se.*
- *M. Bianco is with Swiss National Supercomputer Centre, Lugano 6900, Switzerland. E-mail: mauro.bianco@cscs.ch.*
- *B. Chamberlain is with Cray Inc., Seattle, WA 98164. E-mail: bradc@cray.com.*
- *R. Cledat is with Intel Cooperation, Santa Clara, CA 95050.*
- *C. Edwards is with Sandia National Laboratories, Albuquerque, NM 87185. E-mail: hcedwar@sandia.gov.*
- *H. Finkel is with Argonne National Laboratory, Argonne, IL 60439. E-mail: hfinkel@anl.gov.*
- *K. Fuerlinger is with Ludwig-Maximilians-Universität München, Munich D-80538, Germany. E-mail: Karl.Fuerlinger@nm.ifi.lmu.de.*
- *F. Hannig is with University of Erlangen-Nuremberg, Erlangen 91058, Germany. E-mail: frank.hannig@fau.de.*
- *E. Jeannot is with INRIA Bordeaux Sud-Ouest, Talence 33405, France. E-mail: emmanuel.jeannot@inria.fr.*
- *A. Kamil is with University of Michigan, MI 48109, and also with Lawrence Berkeley National Laboratory, Berkeley, CA 94720. E-mail: akamil@umich.edu.*
- *J. Keasler is with Lawrence Livermore National Laboratory, Livermore, CA 94550. E-mail: keasler1@llnl.gov.*
- *P. Kelly is with Imperial College London, London, United Kingdom. E-mail: p.kelly@imperial.ac.uk.*
- *V. Leung is with Sandia National Laboratories, Albuquerque, NM 87185. E-mail: vjleung@sandia.gov.*
- *H. Ltaief is with King Abdullah University of Science and Technology, Thuwal 23955, Kingdom of Saudi Arabia. E-mail: Ltaief_hatem@yahoo.fr.*
- *N. Maruyama is with RIKEN, Kobe, Hyogo 650-0047, Japan. E-mail: nmaruyama@riken.jp.*
- *C. J. Newburn is with Nvidia Corporation, Santa Clara, CA 95050. E-mail: chris.newburn@intel.com.*
- *M. Pericás is with Chalmers University of Technology 41296, Göteborg, Sweden. E-mail: miquelp@chalmers.se.*

Manuscript received 2 June 2016; revised 12 Apr. 2017; accepted 14 Apr. 2017. Date of publication 10 May 2017; date of current version 13 Sept. 2017. (Corresponding author: Didem Unat.)
Recommended for acceptance by U.V. Catalyurek.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2017.2703149

How do we bring these ideas into productive performance computing?

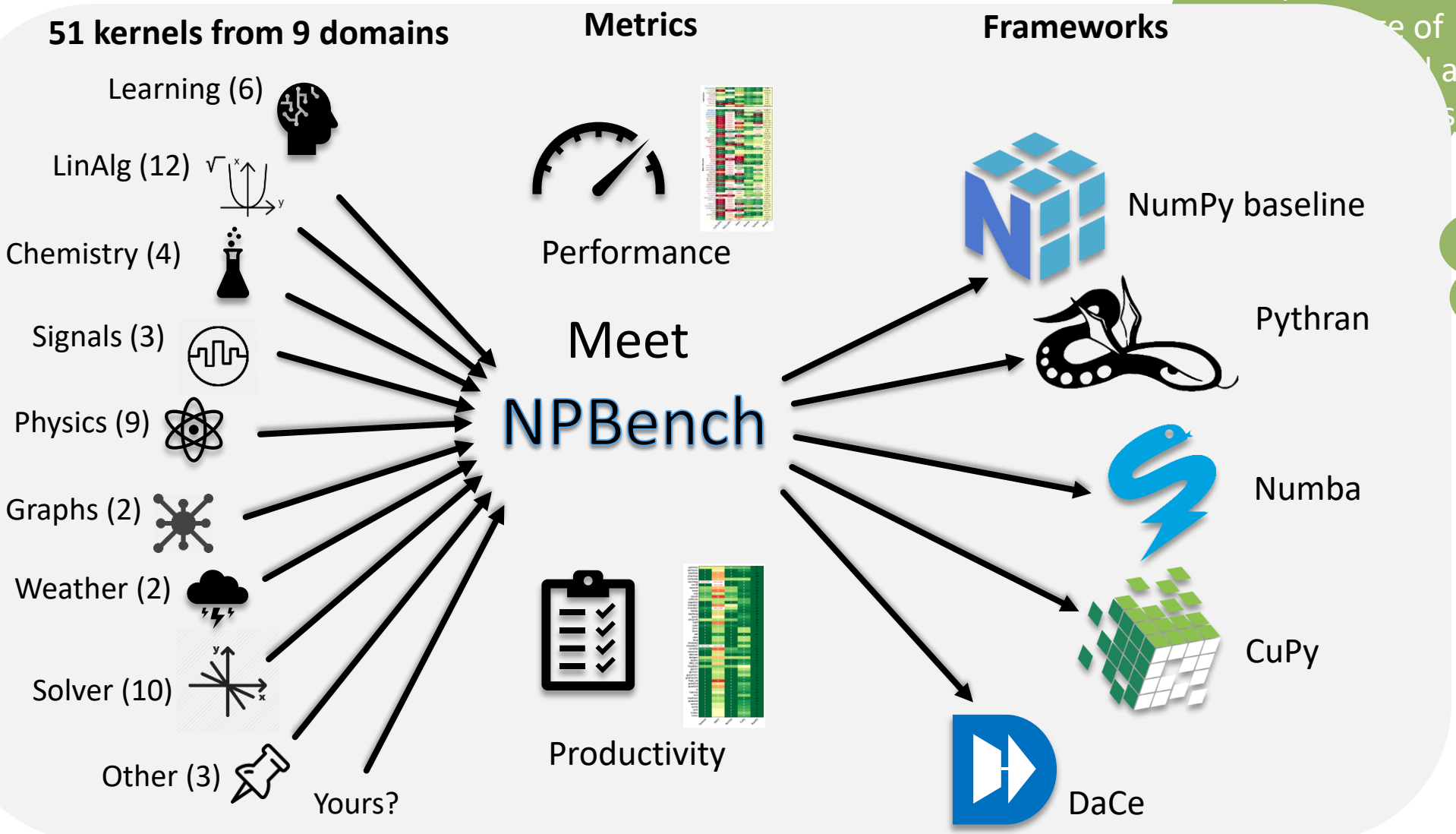
What is the fastest growing high-performance user-base to adopt GPUs even faster than HPC?

TensorFlow PyTorch

build on

python™

Comparing fast python implementations



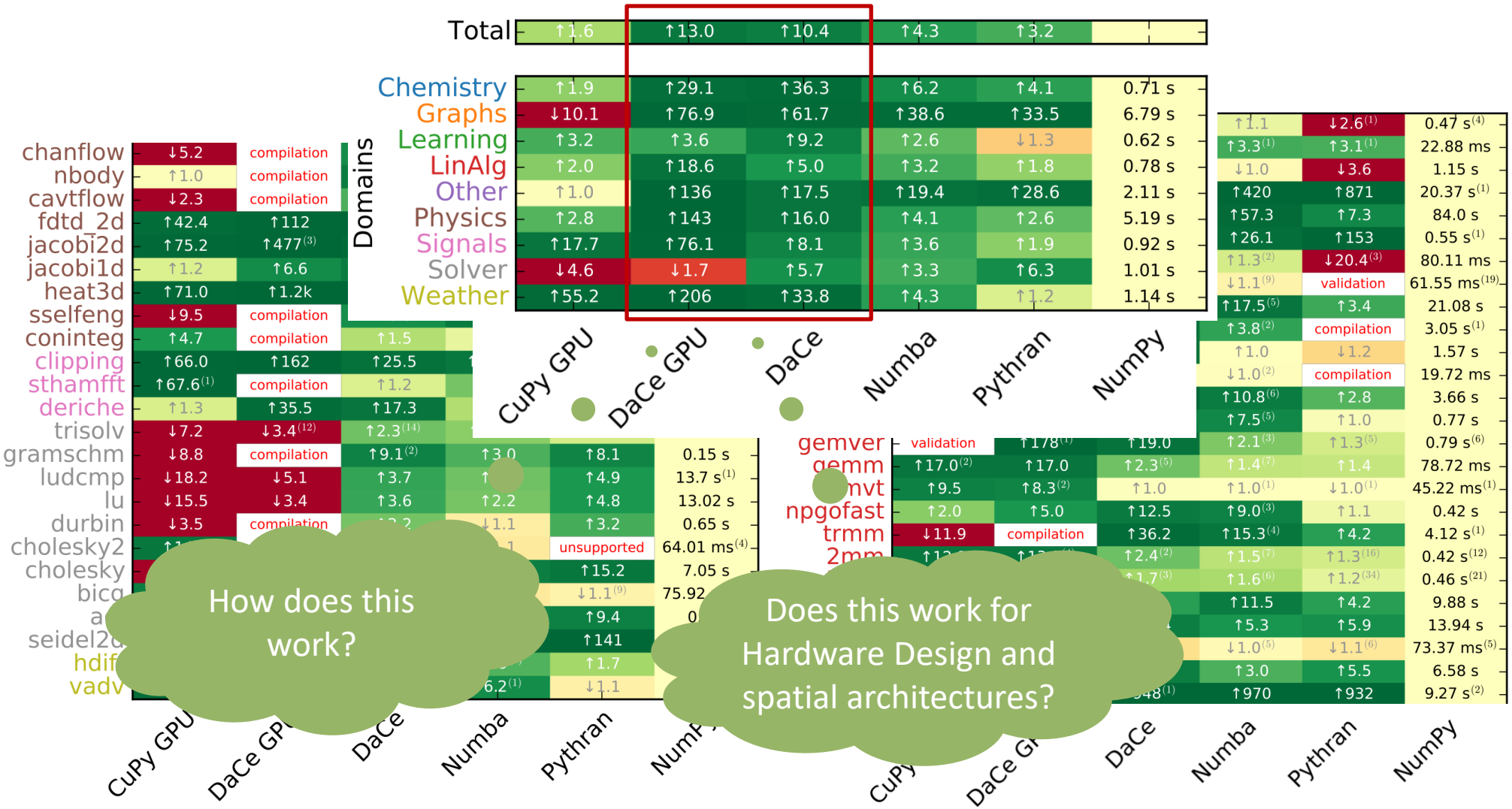
Python is the … of … and …

We need a fair comparison between frameworks

https://github.com/spcl/npbench

NPBench results

Machine with two 16-core Intel Xeon Gold 6130 processors and an Nvidia V100 GPU with 32GB of memory



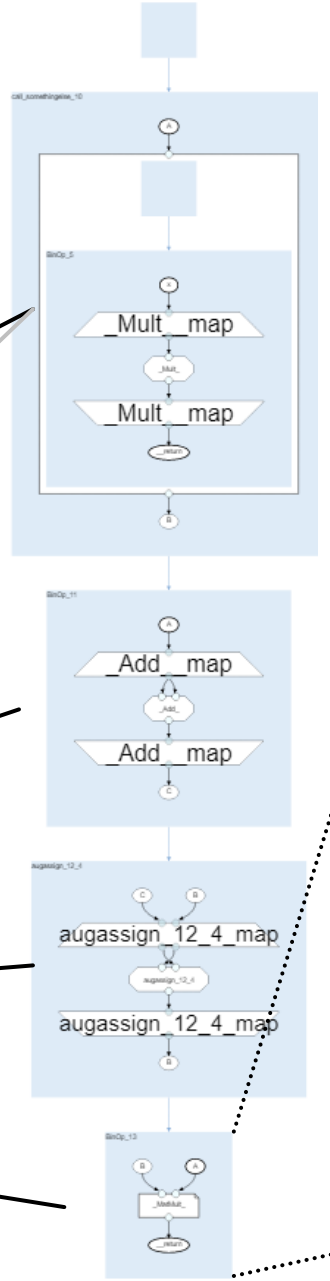
Ziogas et al.: NPBench: A Benchmarking Suite for High-Performance NumPy, ACM ICS'21

Data-Centric Python

python™

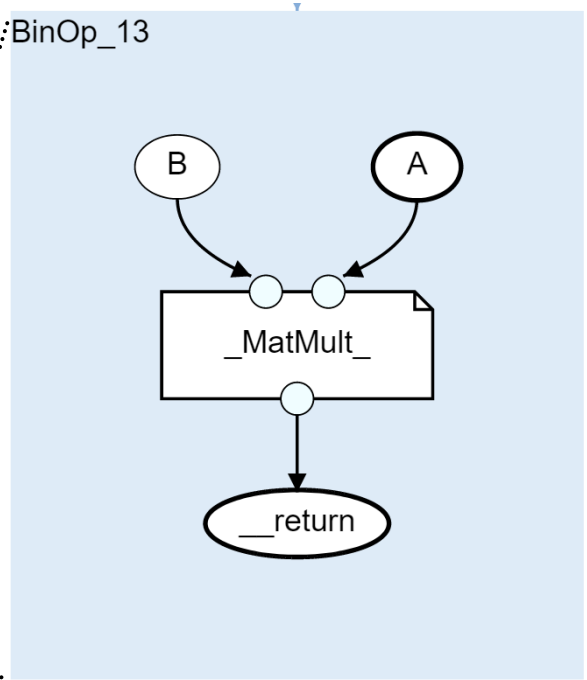
```
import dace

@dace.program
def somethingelse(x):
    return x * 5


@dace.program
def example(A: dace.float64[20, 20]):
    B = somethingelse(A)
    C = A + A
    B += C
    return np.dot(B, A)
```

imperative code



Dace

parametric dataflow representation



Data-Centric Python Vision – Performance Portability

10s of SLOC

Domain Scientist

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

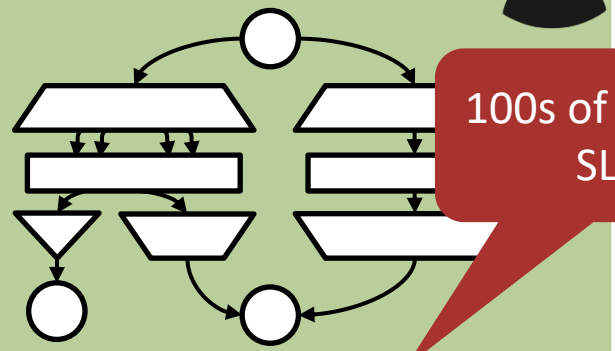
NumPy TensorFlow PyTorch MATLAB DSLs

Applied Scientist

translate DSL into parametric dataflow graphs

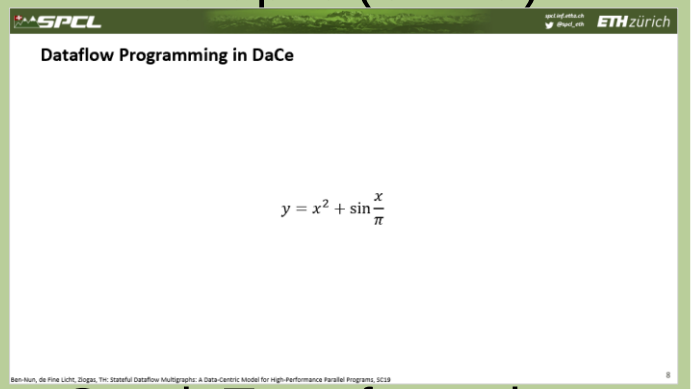
SDFG Builder API
Multi-Level Library Nodes

Performance Engineer



100s of reusable SLOC

Parametric Dataflow Graphs (SDFG)



Graph Transformations (API, Interactive)

Transformed Dataflow

Performance Results

1000s of auto-generated SLOC

Specialized Code Generation

Runtime

CPU Code

GPU Code

FPGA Code

C++ code generation/runtime

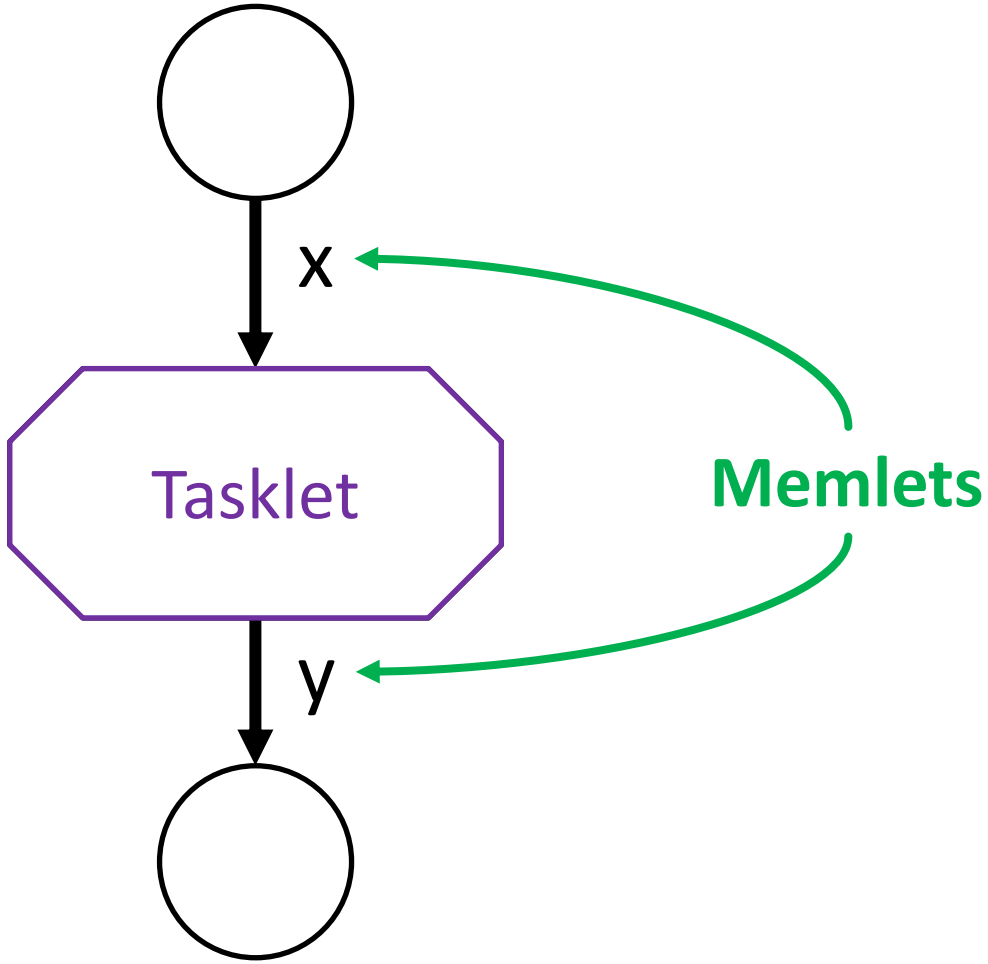
Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Dataflow Programming in DaCe

$$y = x^2 + \sin\frac{x}{\pi}$$

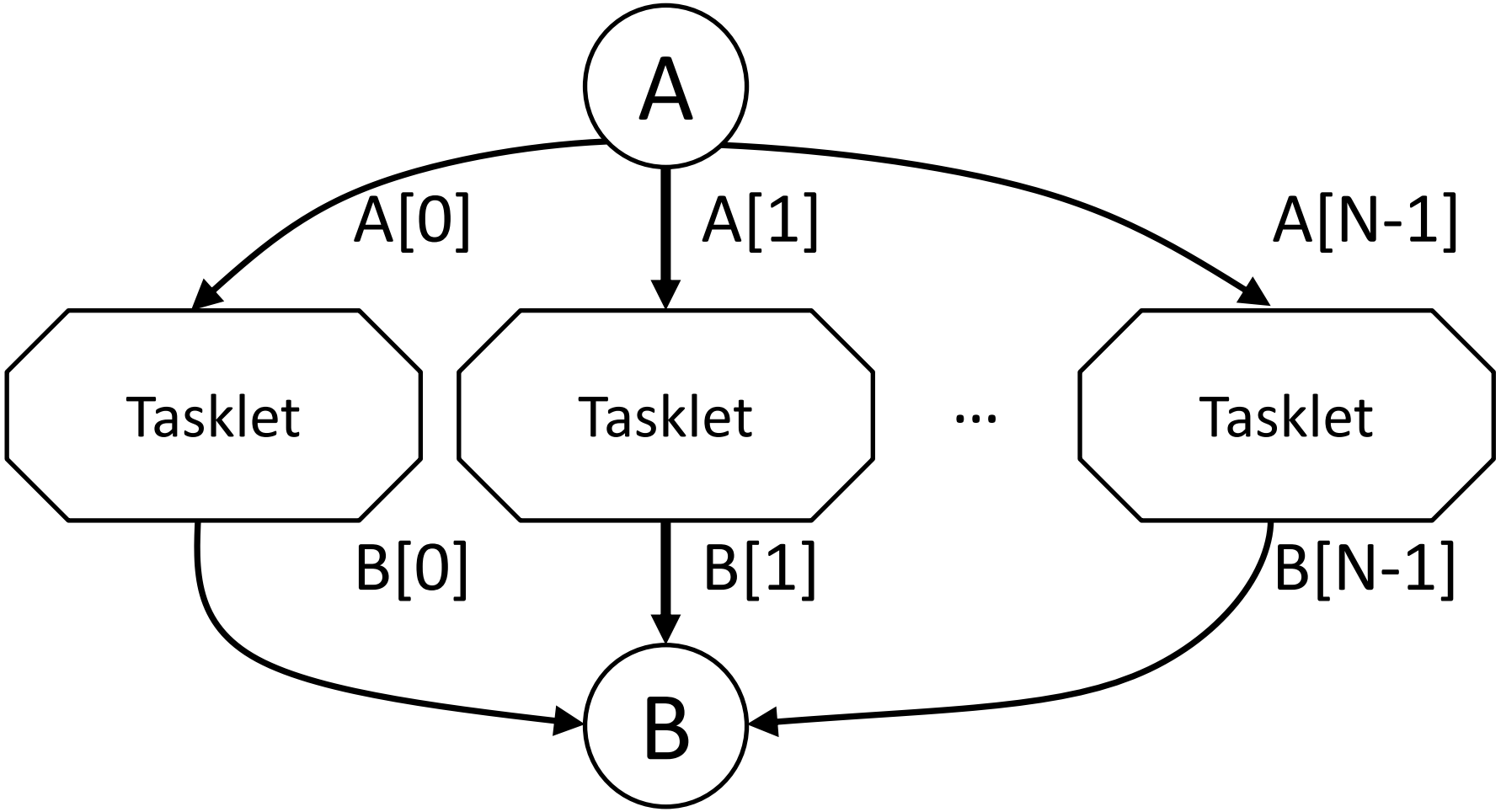
Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Dataflow Programming in DaCe



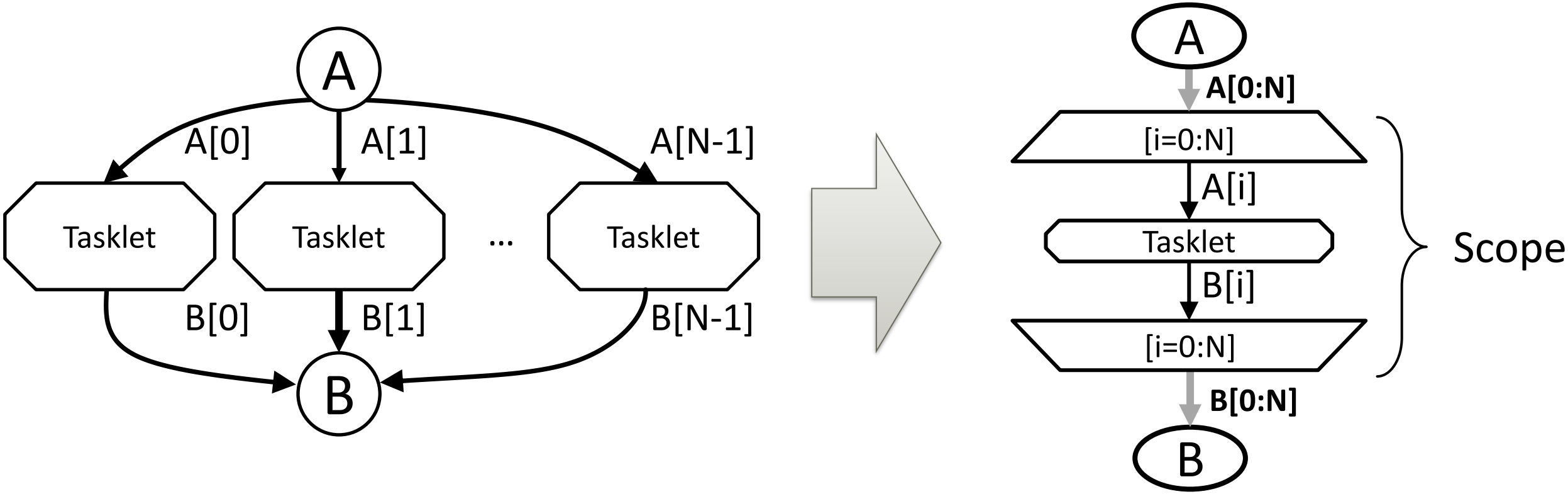
Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Parallel Dataflow Programming

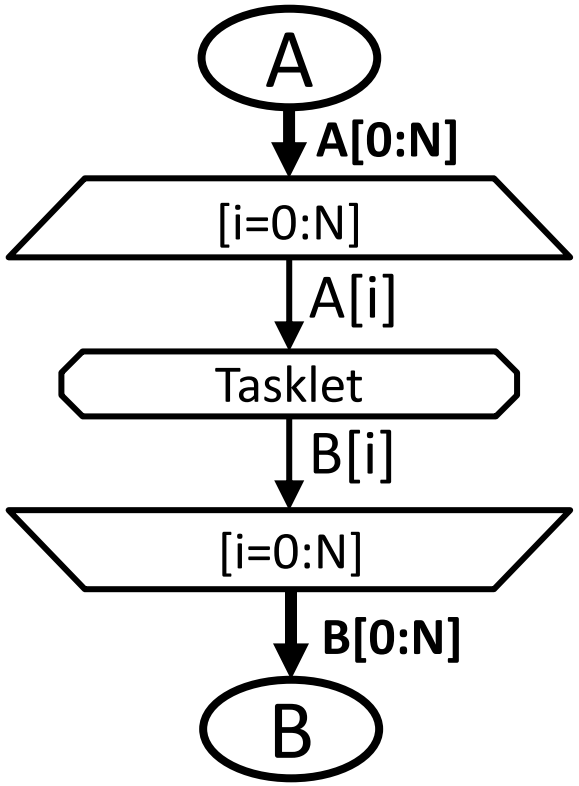


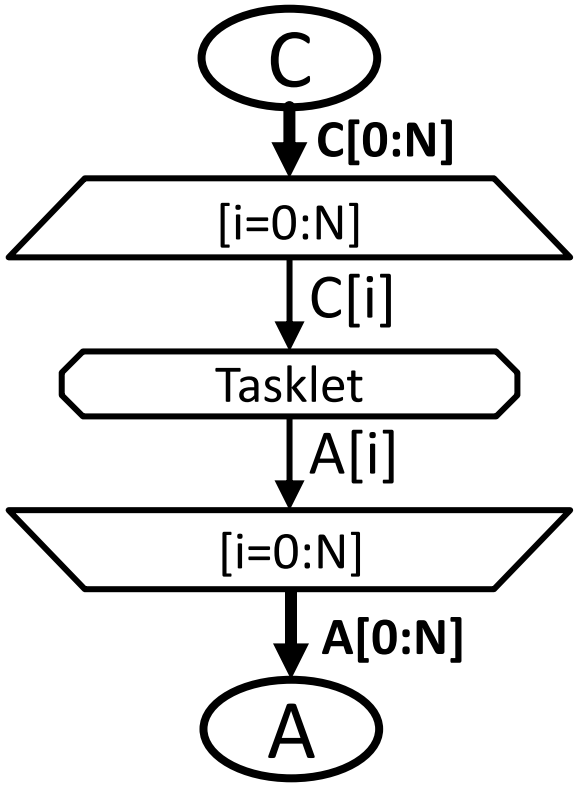
Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Parallel Dataflow Programming



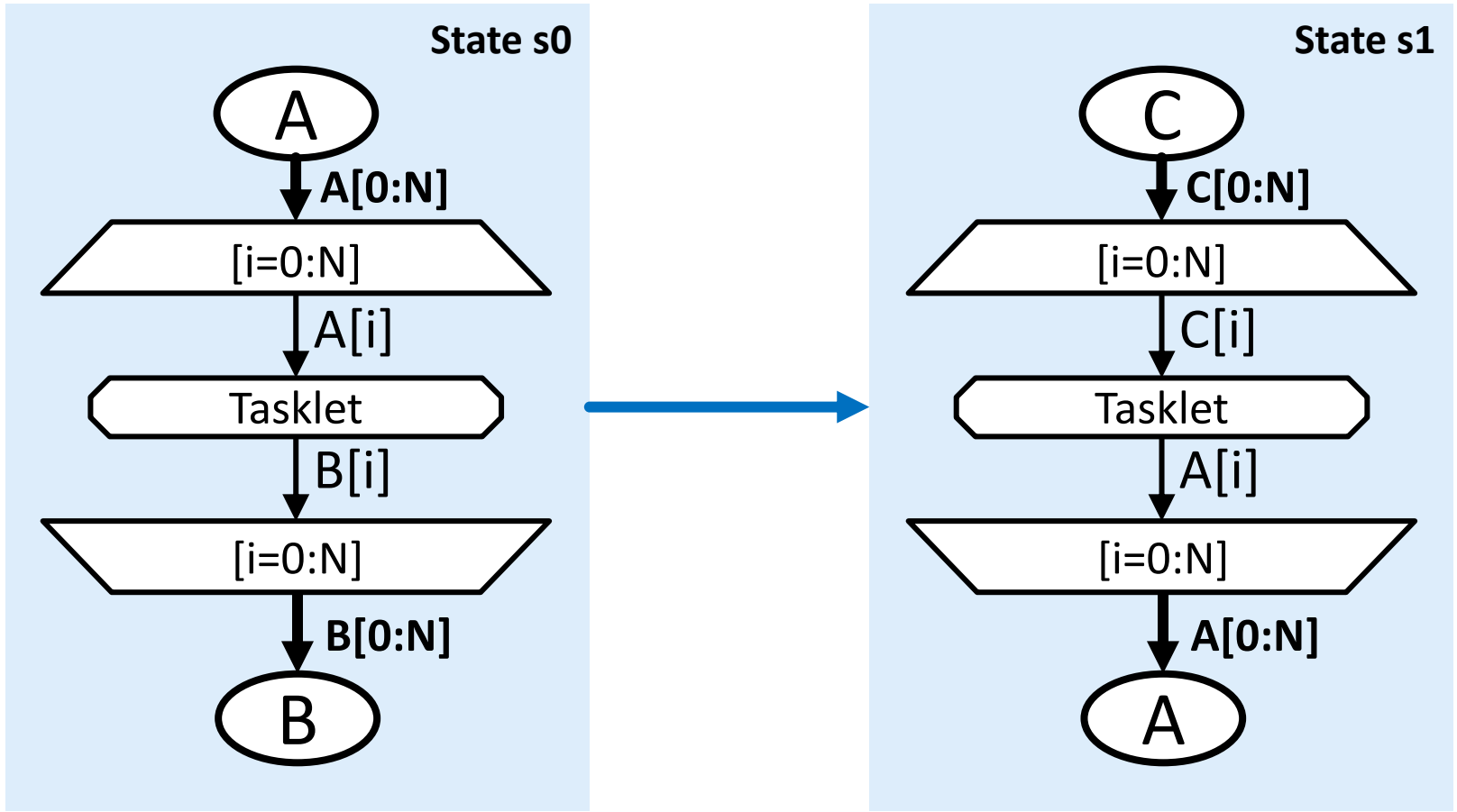
Stateful Parallel Dataflow Programming



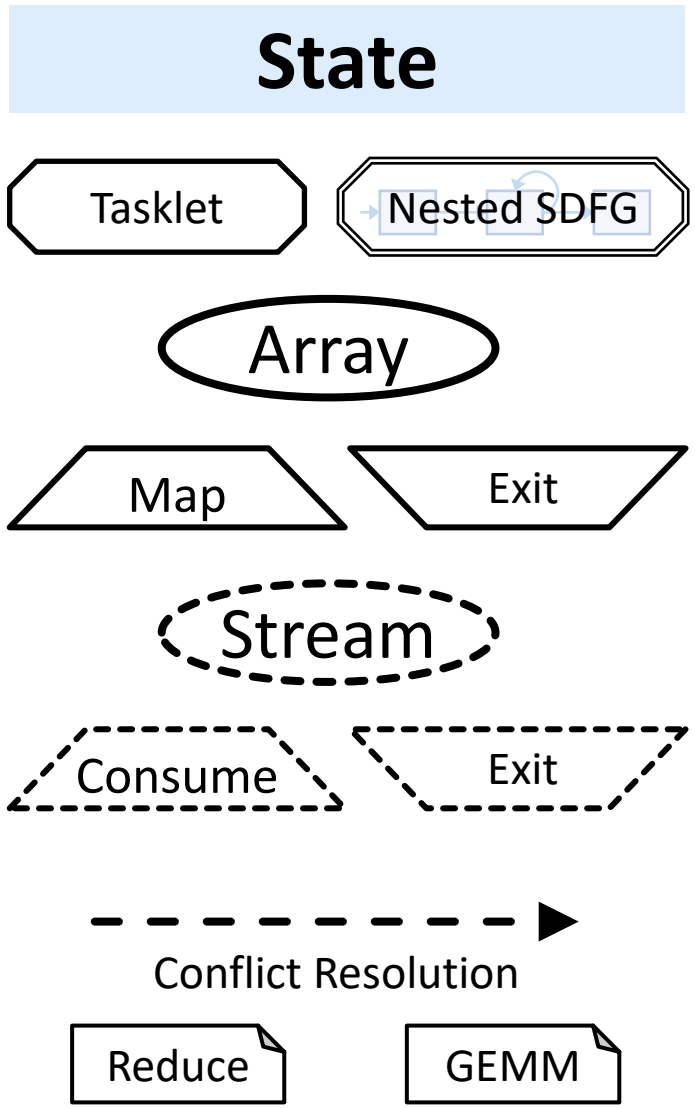


Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Stateful Parallel Dataflow Programming



Meet the Nodes



State machine element

Fine-grained computational block

N-dimensional data container

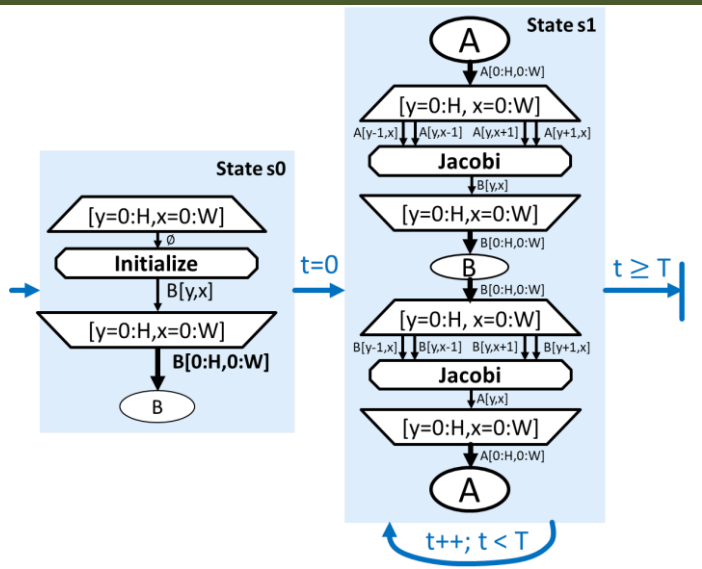
Parametric graph abstraction for parallelism

Streaming data container

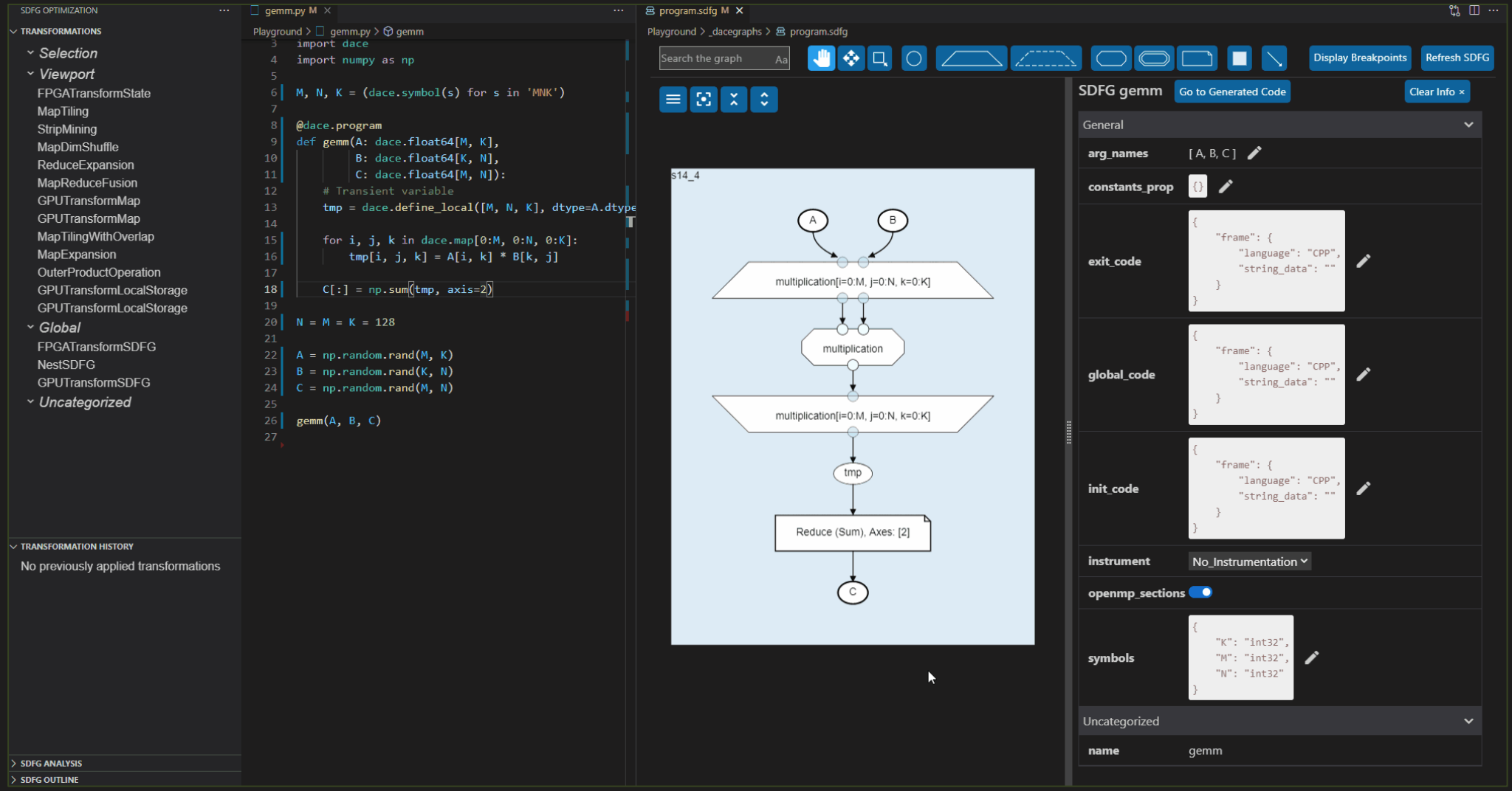
Dynamic mapping of computations on streams

Defines behavior during conflicting writes

Customizable computation with multiple implementations

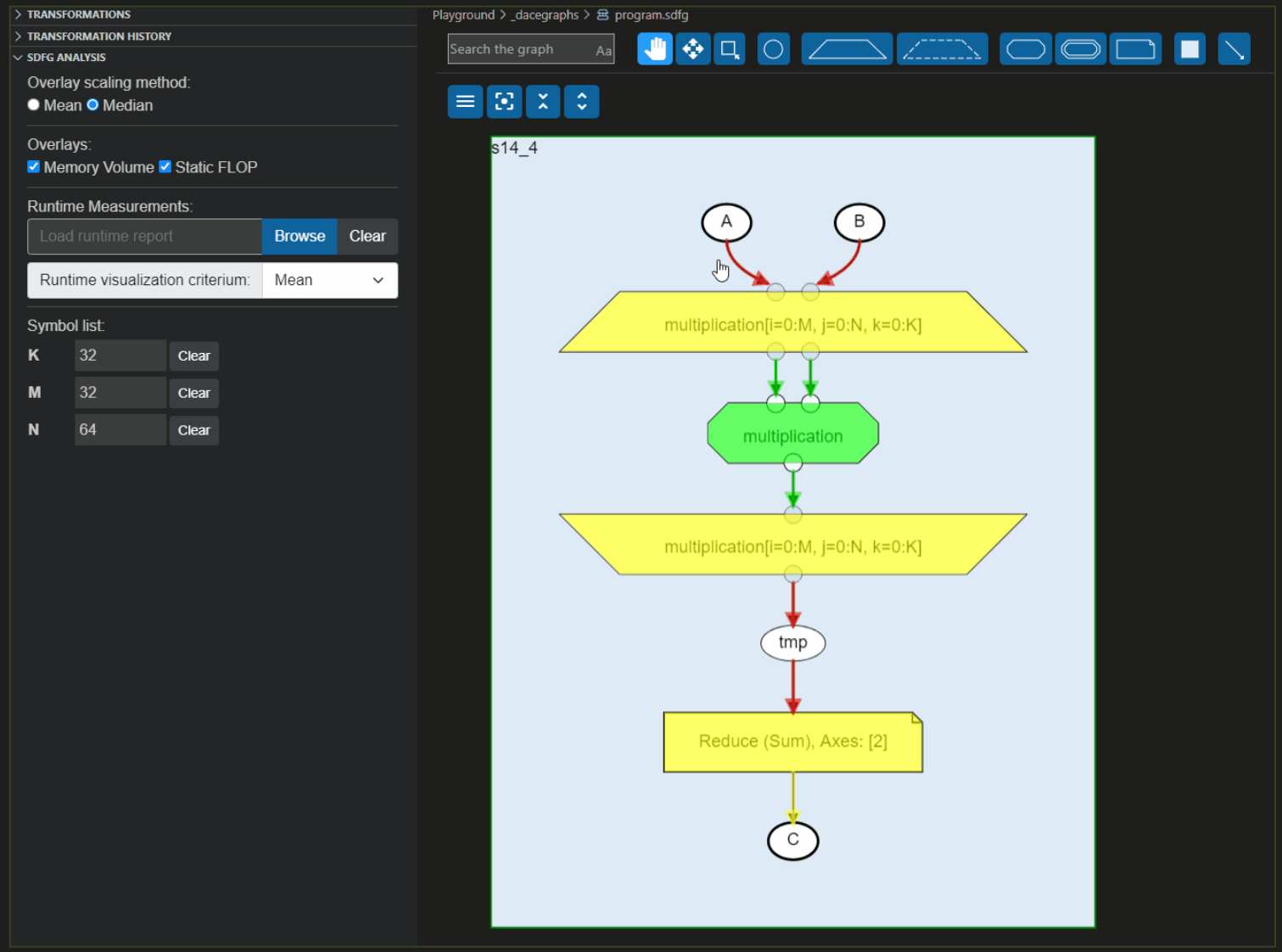


Programmer/Performance Engineer view: Visual Studio Code Integration



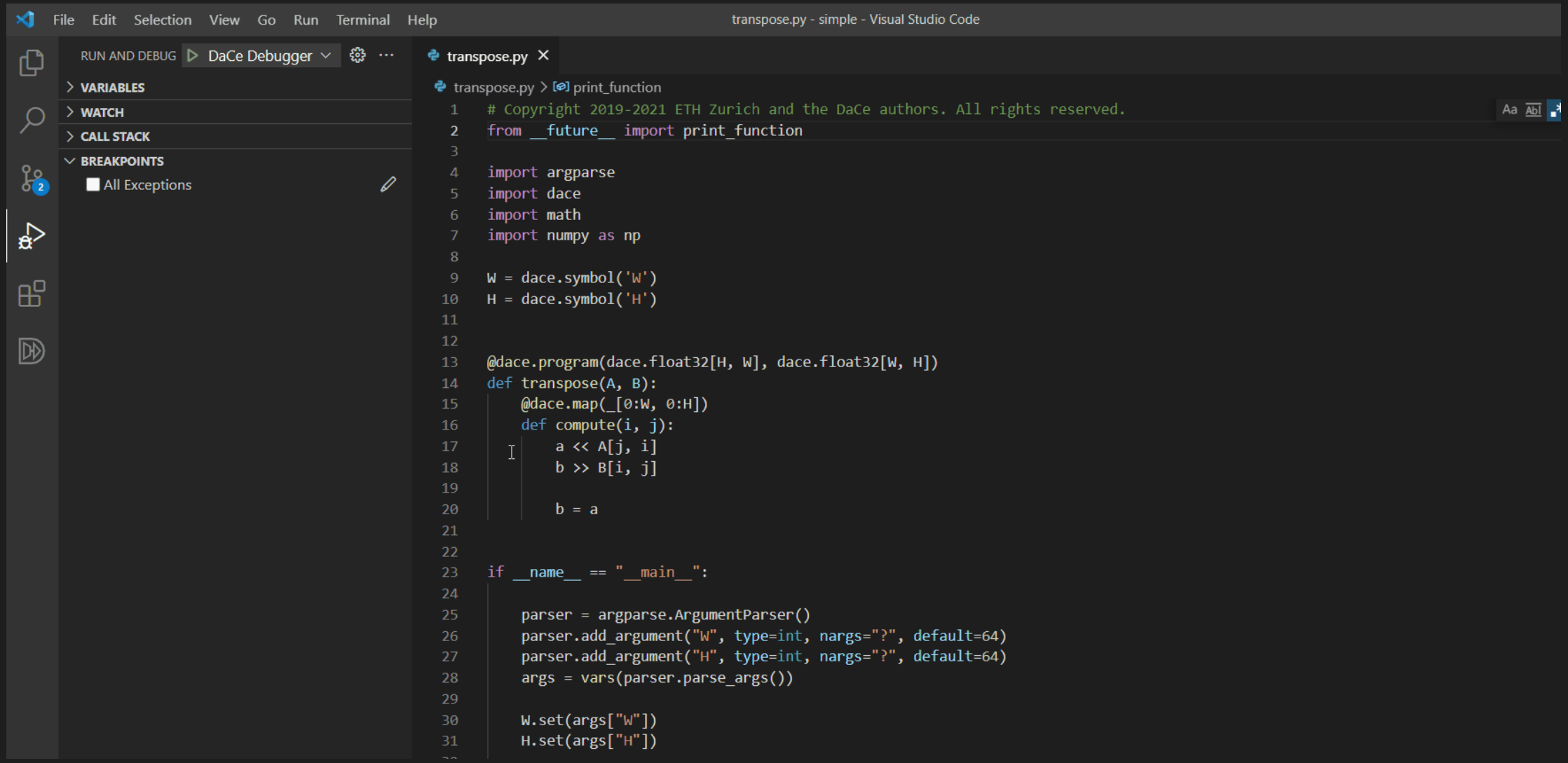
Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Programmer/Performance Engineer view: Analyzing data flows



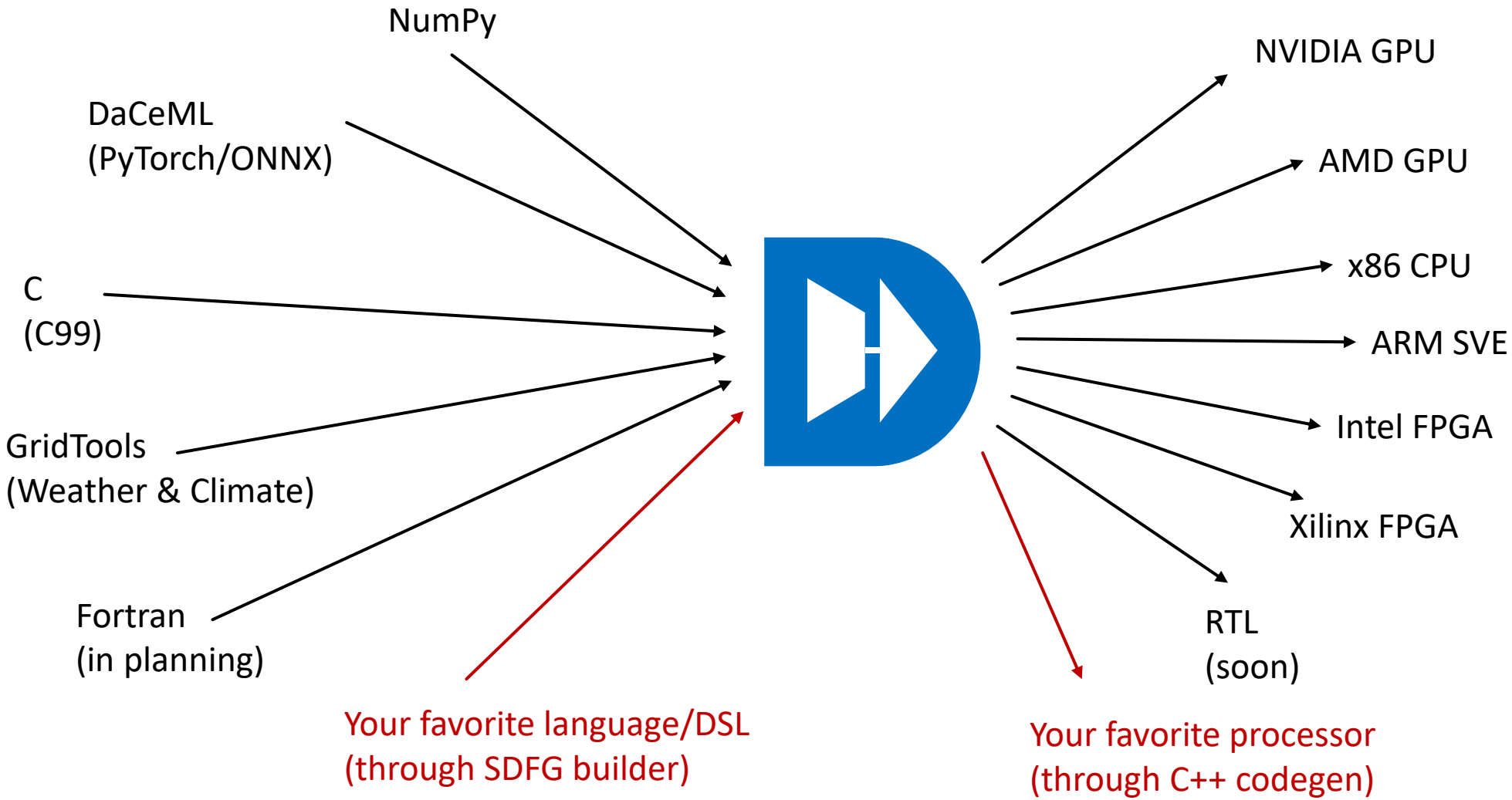
Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Programmer/Performance Engineer view: Debugging



Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

DaCe is a versatile platform



Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

Mapping NumPy to CPU

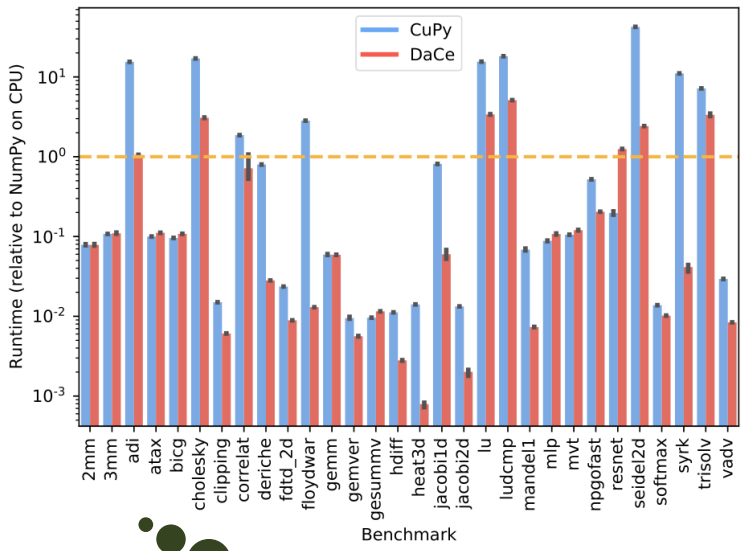
Key principle: keep dataflow of vectorized code around and use it for mapping

PolyBench/C

the Polyhedral Benchmark suite

	DaCe	GCC	ICC	Numba	Pythran	NumPy
Total	↑10.6	↑1.1	↑4.5	↑4.3	↑2.7	
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$^{(1)}$

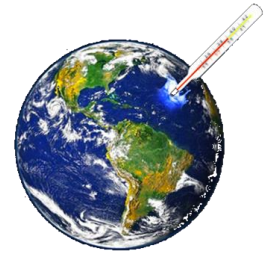
NumPy can even beat the C versions of PolyBench codes!



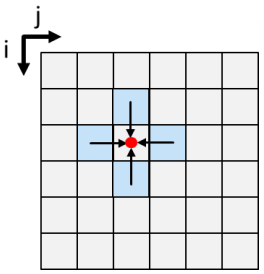
DaCe outperforms CuPy in most cases

Ziogas et al.: Productivity, Portability, Performance: Data-Centric Python, SC21

Mapping Weather/Climate Stencils (in a Python DSL) to FPGAs



Key principle: spatial layout and pipelining using streams and delay buffers



Combined Spatial and Temporal Blocking for High-Performance Stencil Computation on FPGAs Using OpenCL

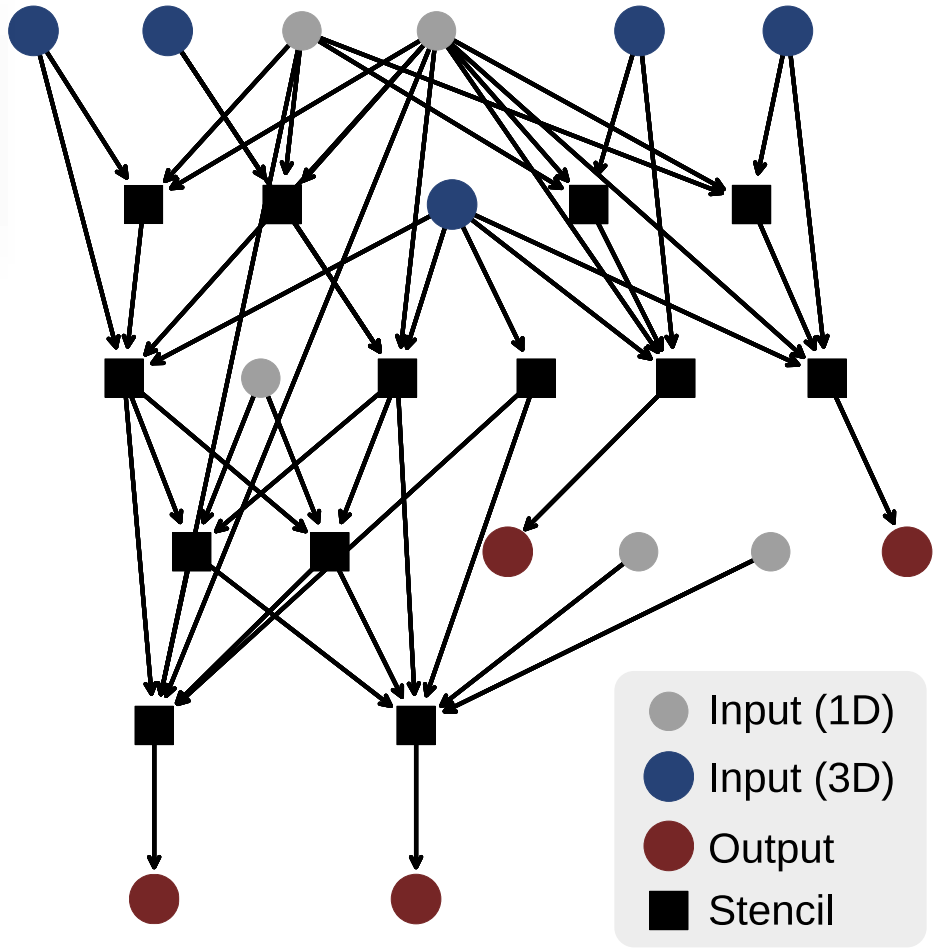
Hamid Reza Zohouri, Artur Podobas, Satoshi Matsuoka
Tokyo Institute of Technology, Tokyo, Japan
{zohouri.h.aa@m,podobas.a.aa@m,matsu@is}.titech.ac.jp

ABSTRACT

Recent developments in High Level Synthesis tools have attracted software programmers to accelerate their high-performance which has prevented large-scale adoption of FPGAs in the High Performance Computing (HPC) community. However, with the recent improvements in High Level Synthesis (HLS), especially the

	Performance	ALM	FF	M20K	DSP
Total		103 M	3.7 M	11.7 K	5760
Avail.		692 K	2.8 M	8.9 K	4468
Jacobi 3D (Ours)	265 GOp/s	233 K 33.6%	534 K 19.3%	1495 16.7%	784 17.6%
Jacobi 3D W=8 (Ours)	921 GOp/s	437 K 63.1%	1207 K 43.6%	2285 25.5%	3072 68.8%
Diffusion 2D W=8 (Ours)	1,313 GOp/s	449 K 64.8%	1329 K 48.0%	2565 28.6%	2304 51.6%
Diffusion 3D W=8 (Ours)	1,152 GOp/s	567 K 81.9%	1606 K 57.9%	5357 59.8%	3072 68.8%
Diffusion 2D (Zohouri et. al. [8])	913 GOp/s	471.4 K 68.0%	1173.6 K 42.3%	2204 24.6%	3844 86.0%
Diffusion 3D (Zohouri et. al. [8])	934 GOp/s	450.5 K 65.0%	1078.2 K 38.9%	8684 97.0%	3592 80.4%

23-44% faster!

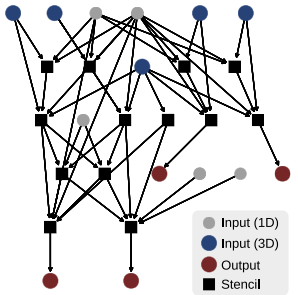


de Fine Licht et al: StencilFlow: Mapping Large Stencil Programs to Distributed Spatial Computing Systems, CGO'21

Mapping Weather Stencils (in a Python DSL) to FPGAs

Key principle: spatial layout and pipelining using streams and delay buffers

318 MHz at 48% DSP utilization



	Runtime	Performance	Peak BW.	%Roof.
Stratix 10	1,178 µs	145 GOp/s	77 GB/s	52%

all memory bound

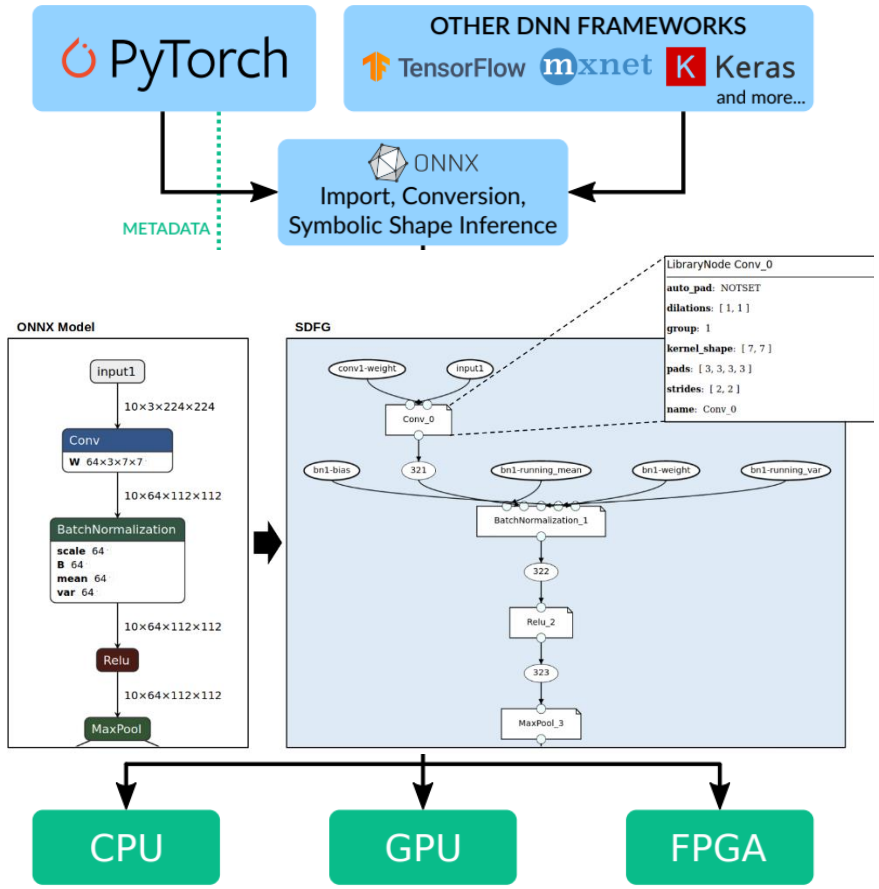
The Stratix 10 is held back by insufficient bandwidth.

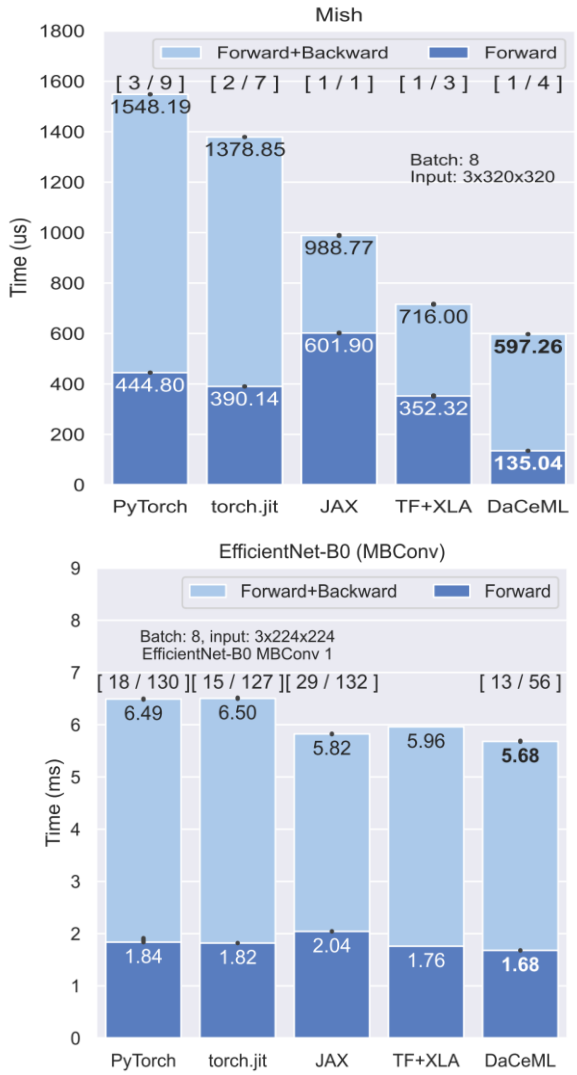
FPGAs are good at deterministically **exploiting bandwidth**, but require a lot of **pipeline parallelism**.

de Fine Licht et al: StencilFlow: Mapping Large Stencil Programs to Distributed Spatial Computing Systems, CGO'21

Mapping Deep Learning Codes to GPUs

Key principle: minimize data movement and optimize data layout





Data Movement Is All You Need: A Case Study on Optimizing Transformers

Andrei Ivanov*, Nikoli Dryden*, Tal Ben-Nun, Shigang Li, Torsten Hoefler
ETH Zürich
firstname.lastname@inf.ethz.ch
* Equal contribution

MLSys'20

arXiv:2007.00072v2 [cs.LG] 2 Jul 2020

***Abstract*—Transformers have become widely used for language modeling and sequence learning tasks, and are one of the most important machine learning workloads today. Training one is a very compute-intensive task, often taking days or weeks, and significant attention has been given to optimizing transformers. Despite this, existing implementations do not efficiently utilize GPUs. We find that data movement is the key bottleneck when training. Due to Amdahl's Law and massive improvements in compute performance, training has now become memory-bound. Further, existing frameworks use suboptimal data layouts. Using these insights, we present a recipe for globally optimizing data movement in transformers. We reduce data movement by up to 22.91% and overall achieve a 1.30× performance improvement over state-of-the-art frameworks when training BERT. Our approach is applicable more broadly to optimizing deep neural networks, and offers insight into how to tackle emerging performance bottlenecks.**

***Index Terms*—Data movement, high-performance computing, deep learning, transformers**

I. Introduction

Transformers [1] are a class of deep neural network architecture for sequence transduction [2], similar to recurrent neural networks [3] and LSTMs [4]. They have recently had a major impact on natural language processing (NLP), including language modeling [5]–[7], question-answering [8], translation [1], and many other applications. The significant improvement in accuracy brought by transformers to NLP tasks is comparable to the improvement brought to computer vision by AlexNet [9] and subsequent convolutional neural networks. Transformers have also begun to be applied to domains beyond NLP where RNNs would previously have been used, including speech recognition [10], reinforcement learning [11], molecular property prediction [12], and symbolic mathematics [13].

challenges such as artificial general intelligence [27]. Thus, improving transformer performance has been in the focus of numerous research and industrial groups.

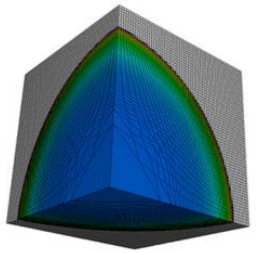
Significant attention has been given to optimizing transformers: local and fixed-window attention [28]–[32], more general structured sparsity [33], learned sparsity [34]–[36], and other algorithmic techniques [19], [37] improve the performance of transformers. Major hardware efforts, such as Tensor Cores and TPUs [38] have accelerated tensor operations like matrix-matrix multiplication (MMM), a core transformer operation. Despite this, **existing implementations do not efficiently utilize GPUs**. Even optimized implementations such as Megatron [18] report achieving only 30% of peak GPU flop/s.

We find that **the key bottleneck when training transformers is data movement**. Improvements in compute performance have reached the point that, due to Amdahl's Law and the acceleration of tensor contractions, training is now memory-bound. Over a third (37%) of the runtime in a BERT training iteration is spent in memory-bound operators: While tensor contractions account for over 99% of the flop performed, they are only 61% of the runtime. By optimizing these, we show that the overhead of data movement can be reduced by up to 22.91%. Further, while MMM is highly tuned by BLAS libraries and hardware, we also find that **existing frameworks use suboptimal data layouts**. Using better layouts enables us to speed up MMM by up to 52%. Combining these insights requires moving beyond peephole-style optimizations and **globally optimizing data movement**, as selecting a single layout is insufficient. Overall, we achieve at least 1.30× performance improvements in training over general-purpose deep learning frameworks, and 1.08× over DeepSpeed [39], the state of the art manually-tuned implementation of BERT.

Rausch et al. "A Data-Centric Optimization Framework for Machine Learning", to appear

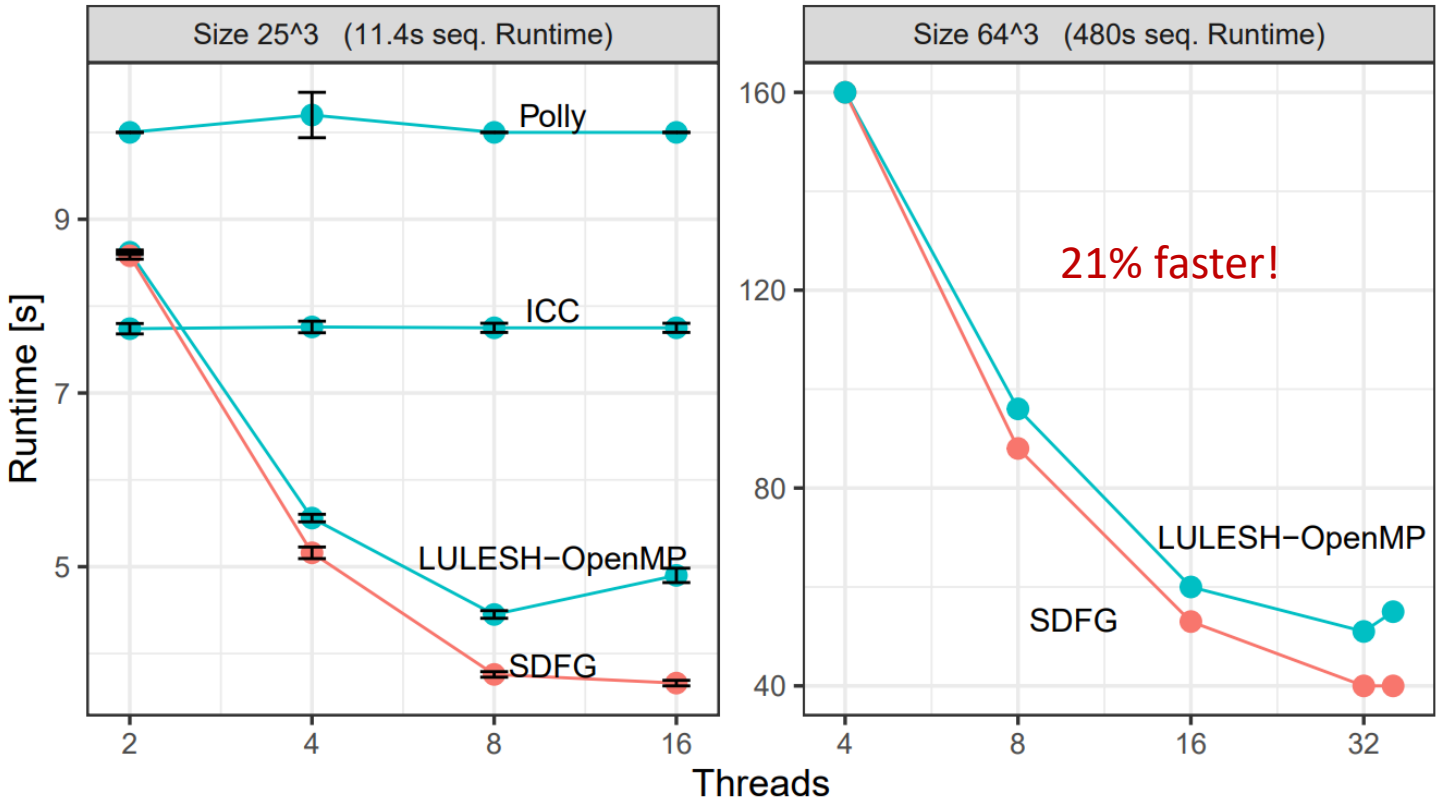
Bonus: Mapping C/C++ (not just Python!) to CPUs

Key principle: find parallel regions using parametric dataflow



LLNL's LULESH C++ benchmark

dual-socket 2×18 core Intel Xeon Gold 6154



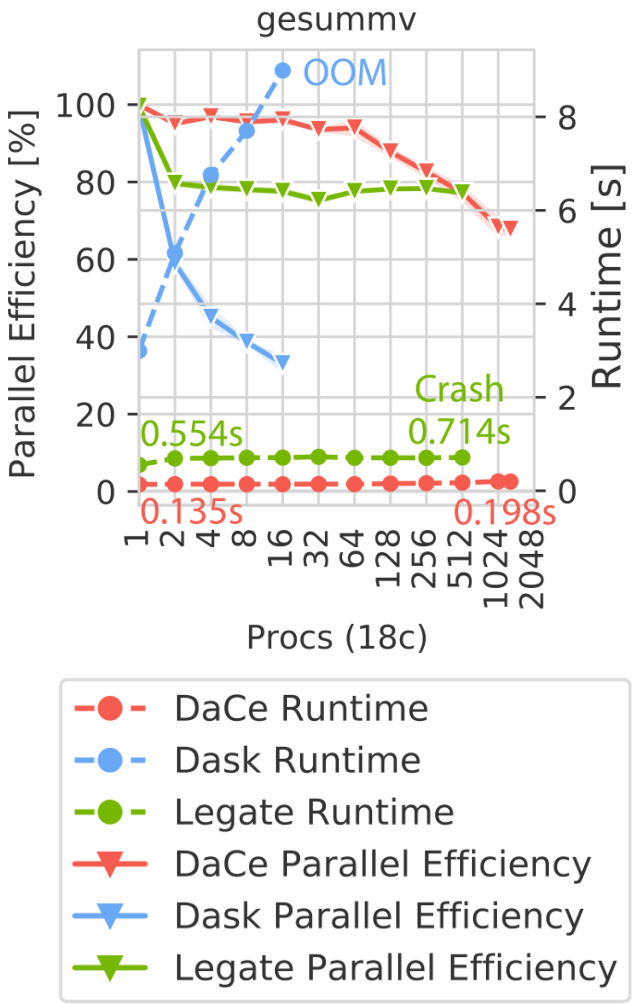
Calotoiu et al: High Performance C — Let Dataflow Do the Heavy Lifting, to appear

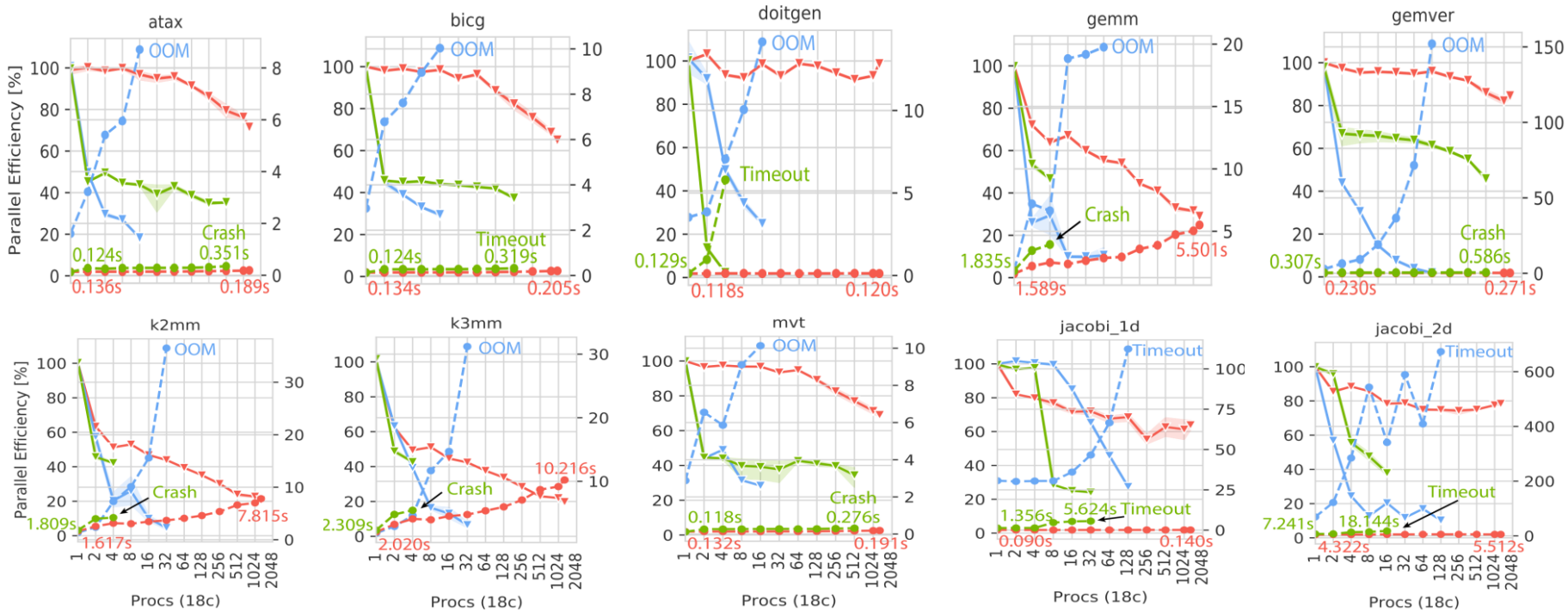
Mapping NumPy to large-scale CPU clusters

DASK

nv-legate/
legate.numpy

Key principle: communication-minimizing data mapping to nodes





Using MPI library nodes

(MPI-like interface)

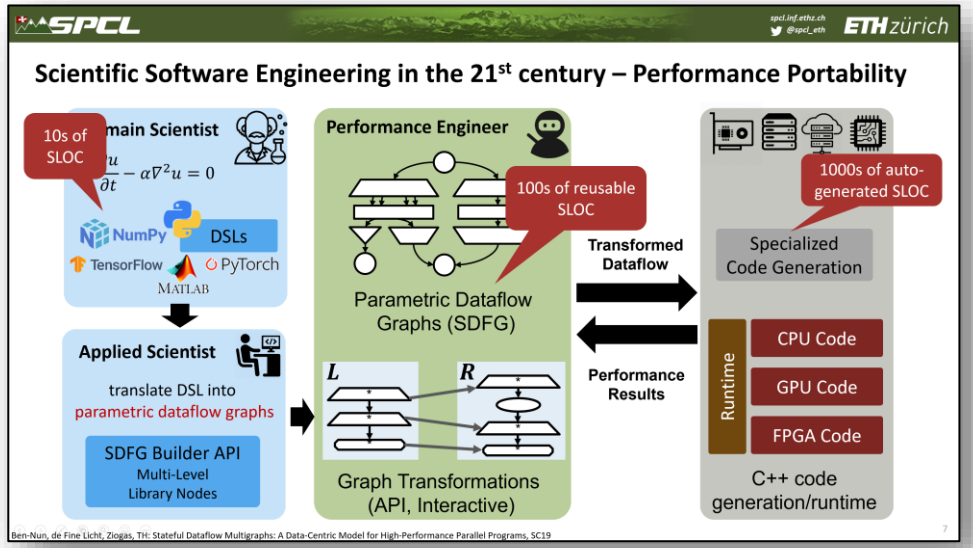
Gordon Bell Prize 2019 on ORNL's Summit (Top-1 machine run with >21k GPUs)

- `pip install dace`
- **Gordon Bell Prize 2019**
  - Quantum Nano Transport simulation

    Design of future micro-processors
- **Now working on large-scale:**
  - Deep Learning (transformers)
  - Climate (COSMO, icon, fv3)
  - Green's functions solvers
  - … your project?

http://spcl.inf.ethz.ch/DAPP

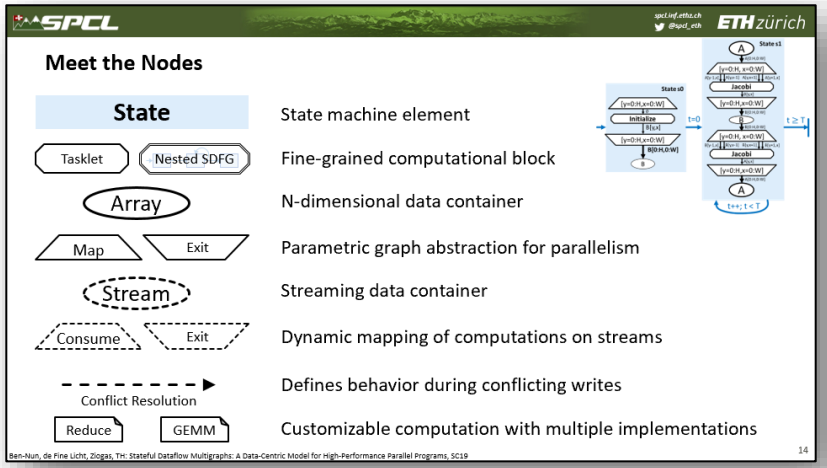
Overview and wrap-up

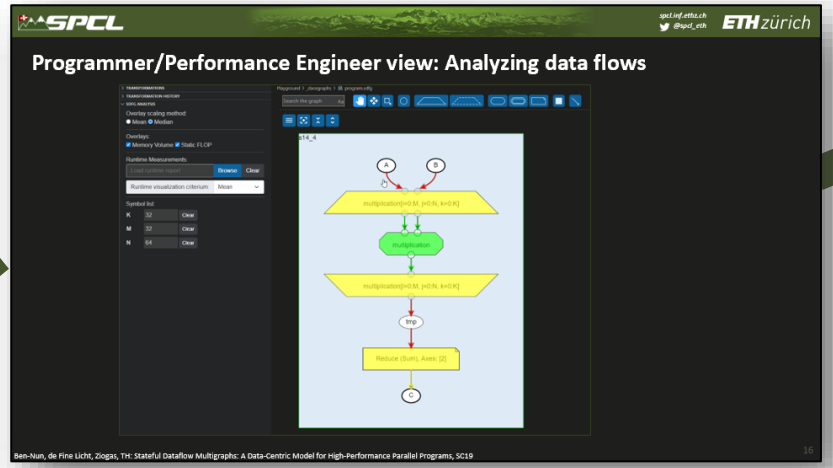


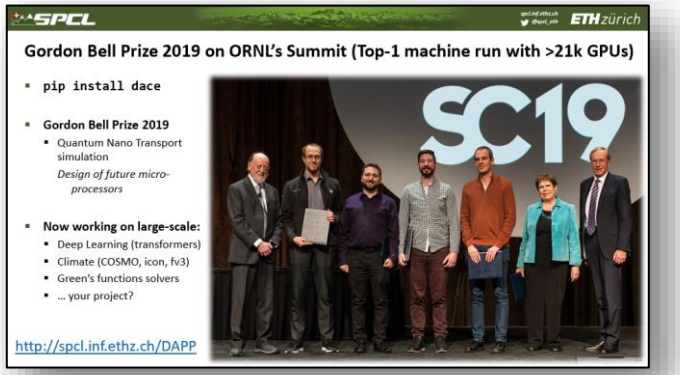
https://www.github.com/spcl/dace

```
pip install dace
```

Open PhD and Postdoc positions:
https://spcl.inf.ethz.ch/Jobs/

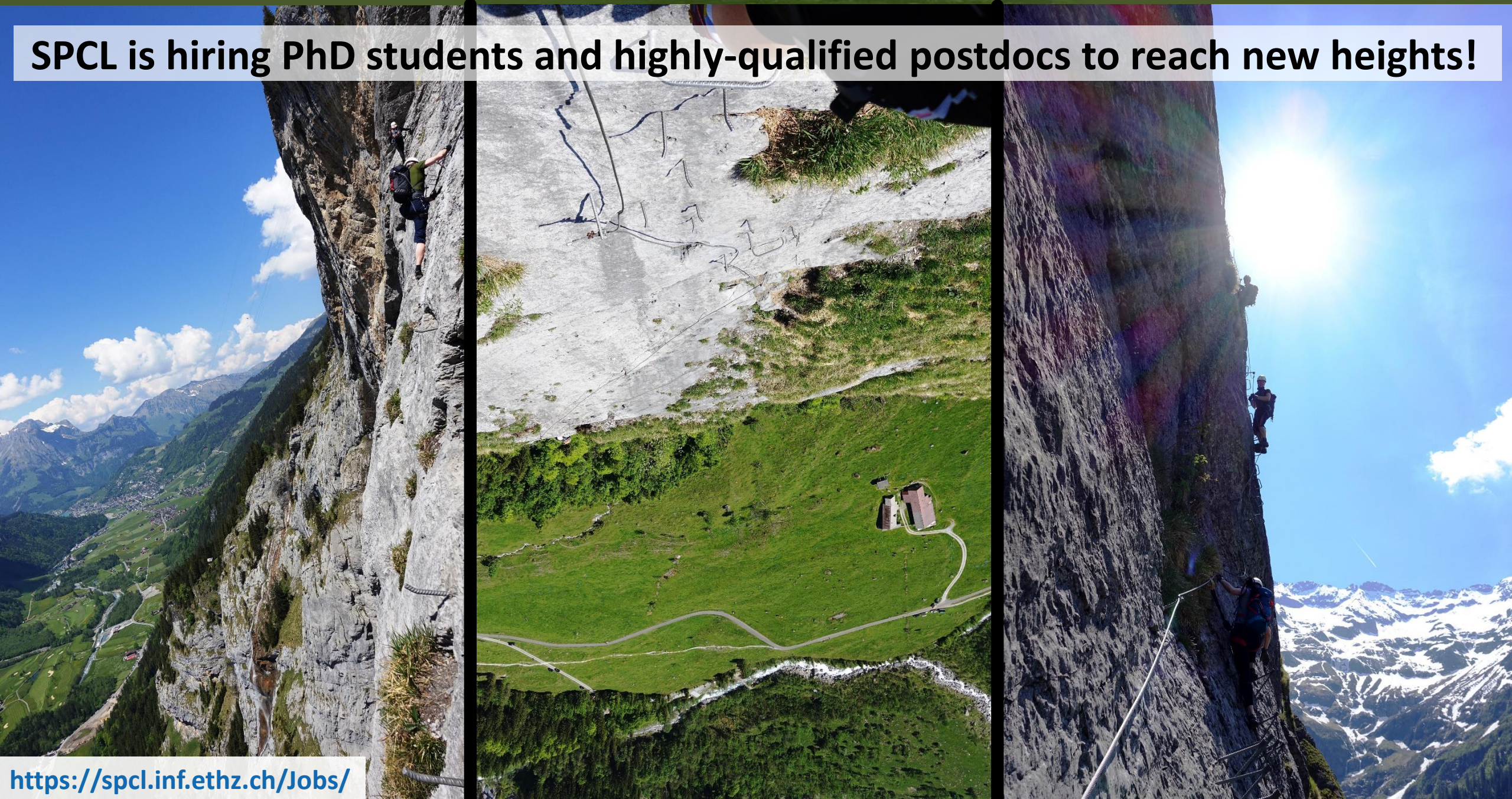






This project has received funding from the European Research Council (ERC) under grant agreement "DAPP (PI: T. Hoefler)“ and DEEP-SEA, No. 955606.

SPCL is hiring PhD students and highly-qualified postdocs to reach new heights!



https://spcl.inf.ethz.ch/Jobs/