HIGH PERFORMANCE STENCIL CODE GENERATION WITH LIFT

Bastian Hagedorn[†] | Larisa Stoltzfus[†] | Michel Steuwer[‡] | Sergei Gorlatch[†] | Christophe Dubach[†]

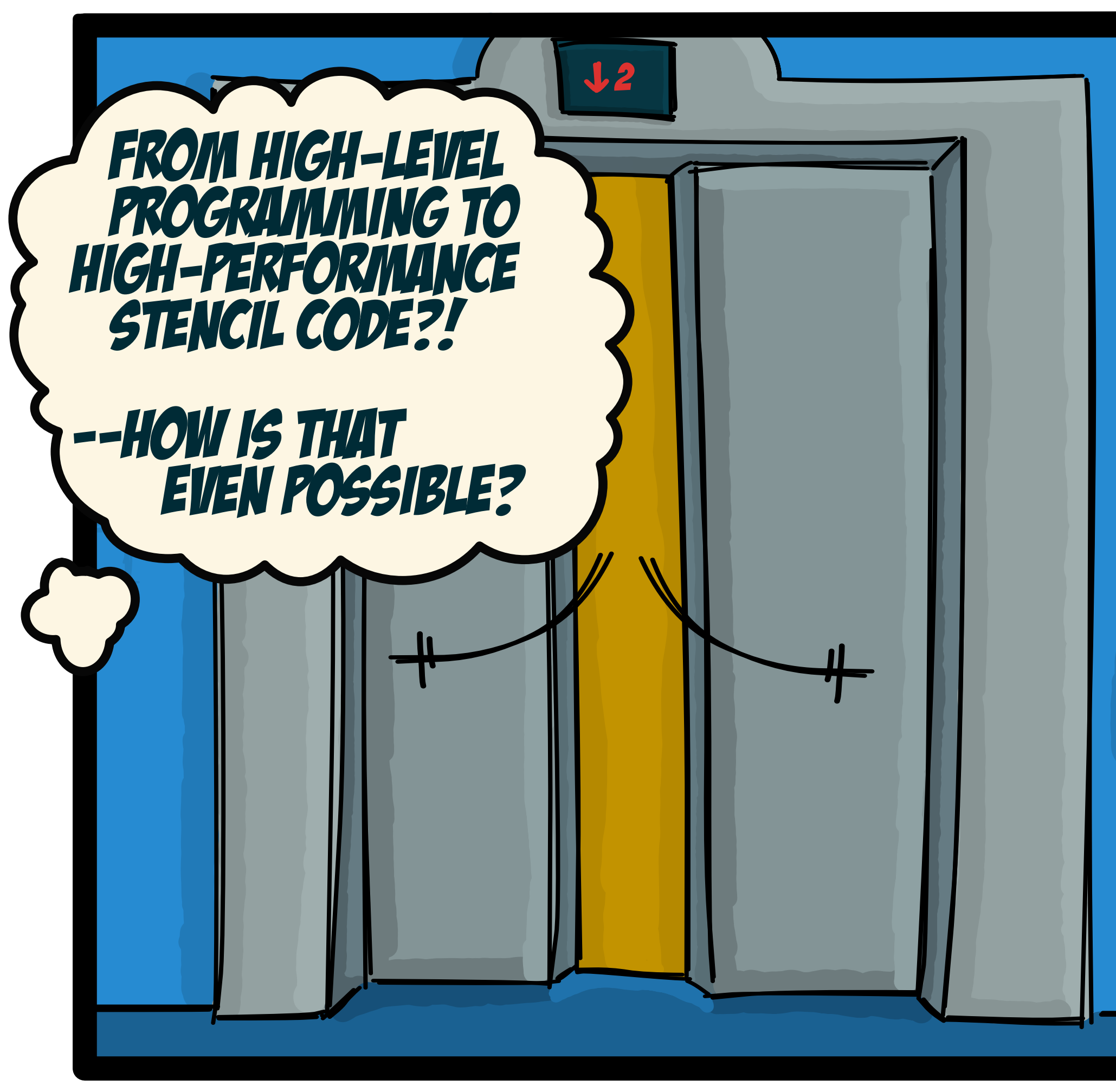
University of Münster, Germany | University of Edinburgh, UK | University of Glasgow, UK

ABSTRACT

STENCIL COMPUTATIONS ARE USED IN A WIDE RANGE OF APPLICATIONS FROM PHYSICAL SIMULATIONS TO MACHINE-LEARNING. OPTIMIZING AND TUNING THEM FOR PARALLEL HARDWARE REMAINS CHALLENGING.

LIFT IS A NEW APPROACH TO ACHIEVING PERFORMANCE PORTABILITY BASED ON A SMALL SET OF REUSABLE PARALLEL PRIMITIVES. ITS KEY NOVELTY IS ENCODING OF OPTIMIZATION AS A SYSTEM OF REWRITE RULES WHICH DEFINE THE OPTIMIZATION SPACE.

WE EXTEND LIFT WITH SUPPORT FOR STENCIL COMPUTATIONS BY ADDING A SMALL NUMBER OF PRIMITIVES TOGETHER WITH A FEW REWRITE RULES TO ACHIEVE PERFORMANCE PORTABILITY FOR STENCIL COMPUTATIONS. PERFORMANCE RESULTS ON SEVERAL APPLICATIONS SHOW THAT THIS APPROACH LEADS TO HIGH PERFORMANCE.


FROM HIGH-LEVEL PROGRAMMING TO HIGH-PERFORMANCE STENCIL CODE?!
--HOW IS THAT EVEN POSSIBLE?
↓2

LIFT

- 2. HIGH-LEVEL PROGRAMMING
- 1. LOW-LEVEL OPTIMIZATIONS
- G. HIGH PERFORMANCE

WWU MÜNSTER (†)

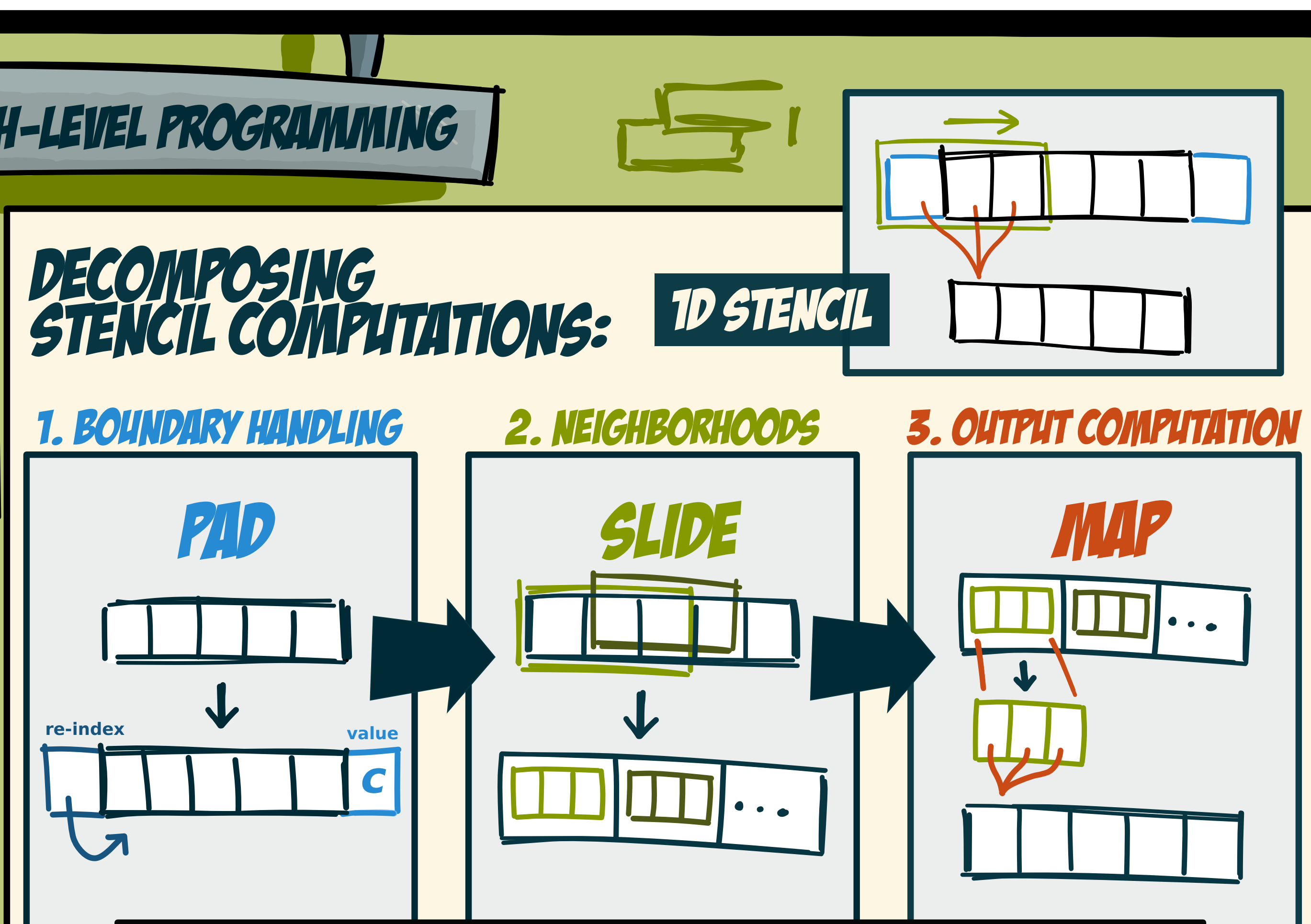
University of Glasgow (‡)

THE UNIVERSITY of EDINBURGH (†)

FUNDED BY EUROLAB-4-HPC AND HiPEAC

HIGH-LEVEL PROGRAMMING

DECOMPOSING STENCIL COMPUTATIONS:

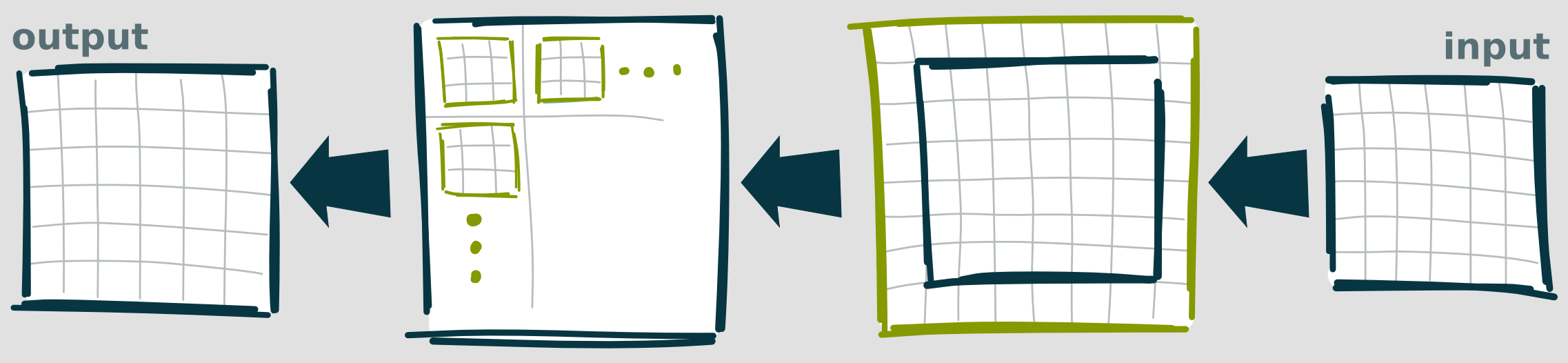


1D STENCIL IN LIFT:

```
fun(ArrayType(Float, N), input =>
map( reduce (+) 0 ) o
  slide(3,1) o
    pad(1,1, clamp) $ input
```

MULTI-DIMENSIONAL STENCIL COMPUTATIONS

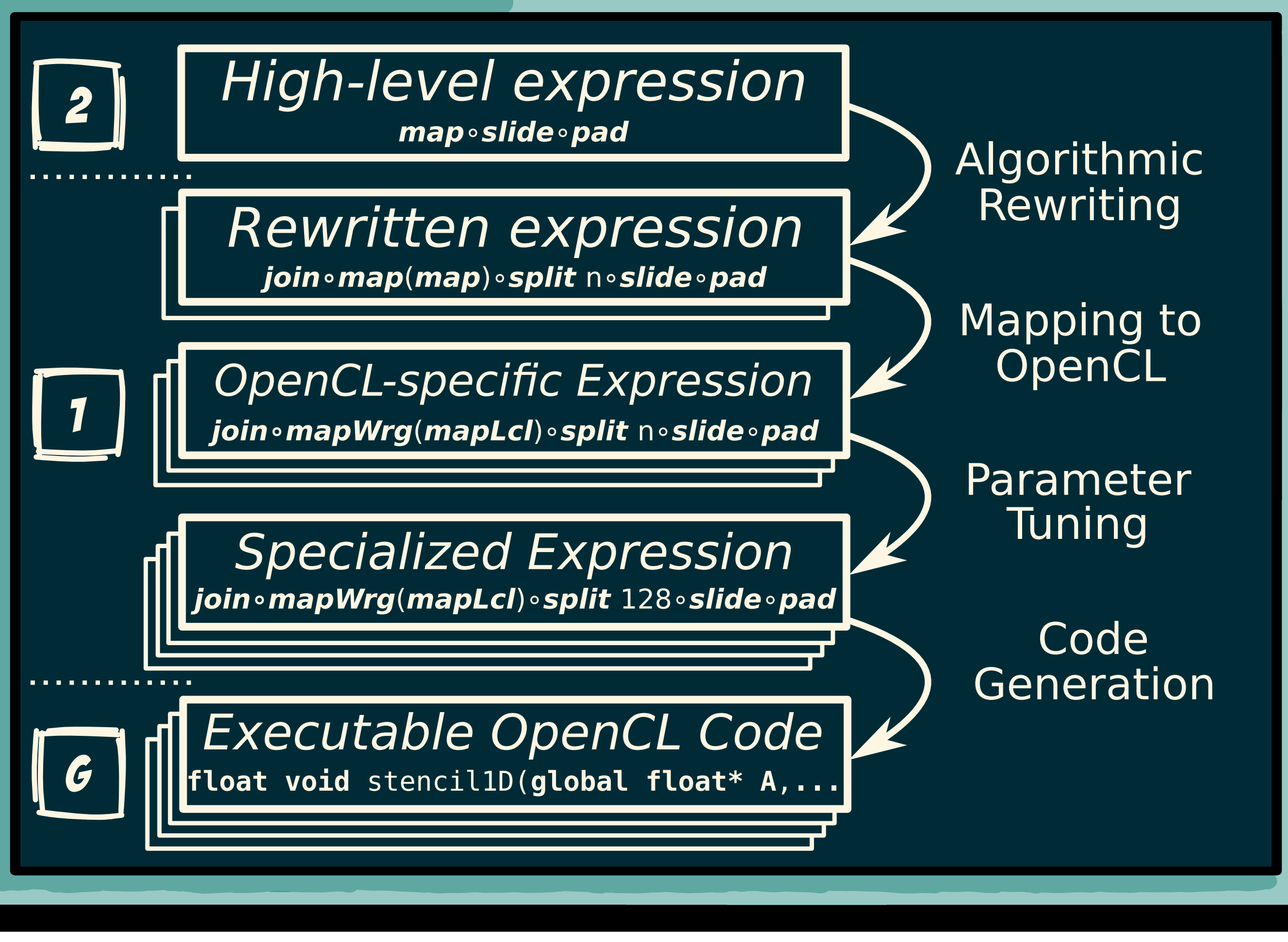
are expressed as compositions of inituitive, generic 1D primitives



$$map_2(f) \circ slide_2(size, step) \circ pad_2(l, r, h)$$

$=map(map(f))$ $\quad =map(transpose) \circ slide(size, step) \circ map(slide(size,step))$ $\quad =map(pad(l,r,h)) \circ pad(l,r,h)$

LOW-LEVEL OPTIMIZATIONS



- 2 — High-level expression: map∘slide∘pad
- Algorithmic Rewriting
- Rewritten expression: join∘map(map)∘split n∘slide∘pad
- Mapping to OpenCL
- 1 — OpenCL-specific Expression: join∘mapWrg(mapLcl)∘split n∘slide∘pad
- Parameter Tuning
- Specialized Expression: join∘mapWrg(mapLcl)∘split 128∘slide∘pad
- Code Generation
- G — Executable OpenCL Code: `float void stencil1D(global float* A,...`

$map(f) = join \circ map(map\ f) \circ split\ n$

DIVIDE & CONQUER

$map(f) \circ map(g) = map(f \circ g)$

MAP-FUSION

Lift is a code generation approach based on a high-level, data-parallel intermediate language.

It is designed as a target for DSLs and exploits functional principles to produce high-performance GPU code. Optimizations are all encoded as formal, semantics-preserving rewrite rules.

These rules define an optimization space which is automatically searched for high performance code.

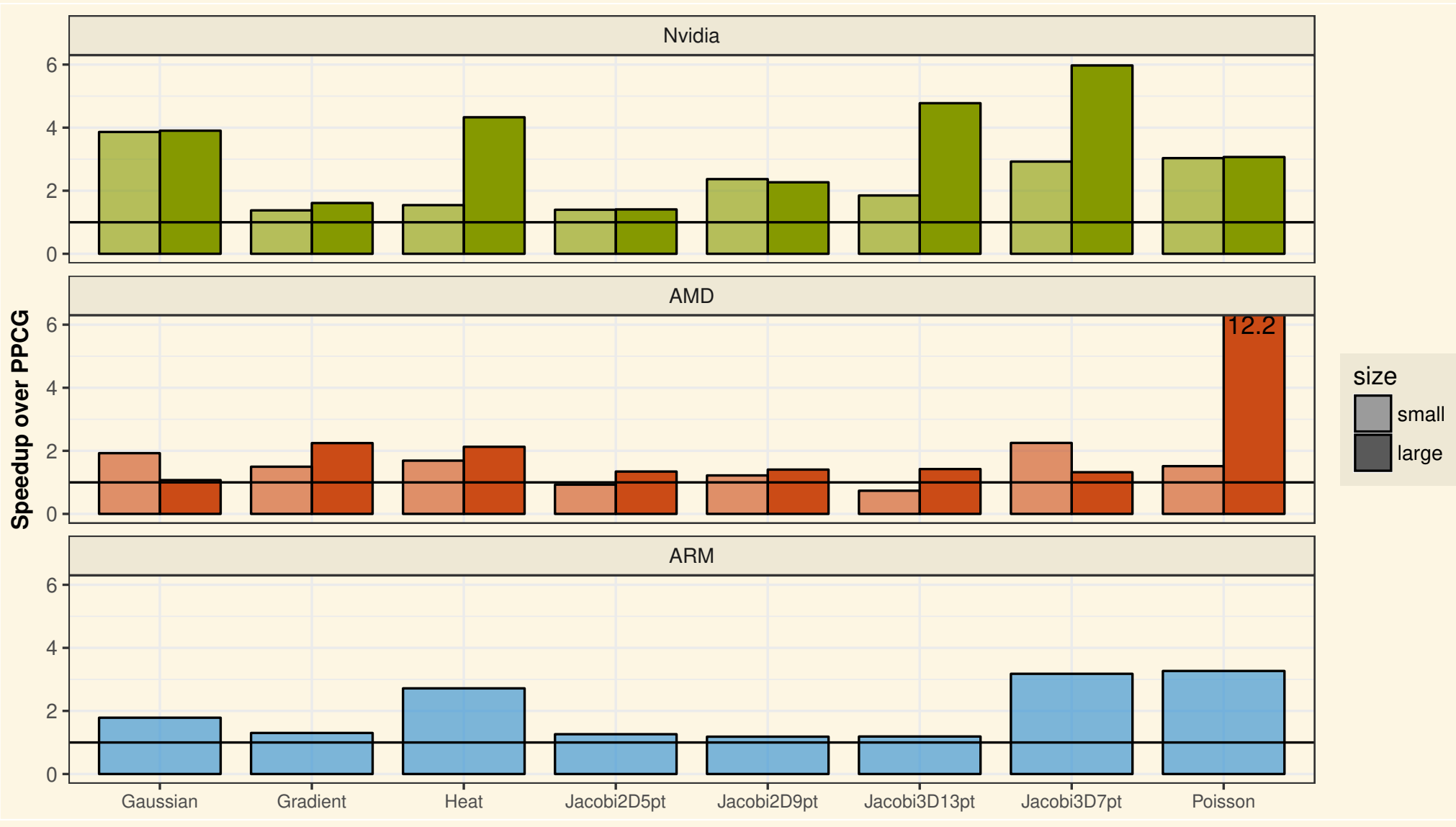
This approach liberates programmers from the tedious process of re-writing and tuning their code for each new domain or hardware.

IDEOLOGY & CONCEPT

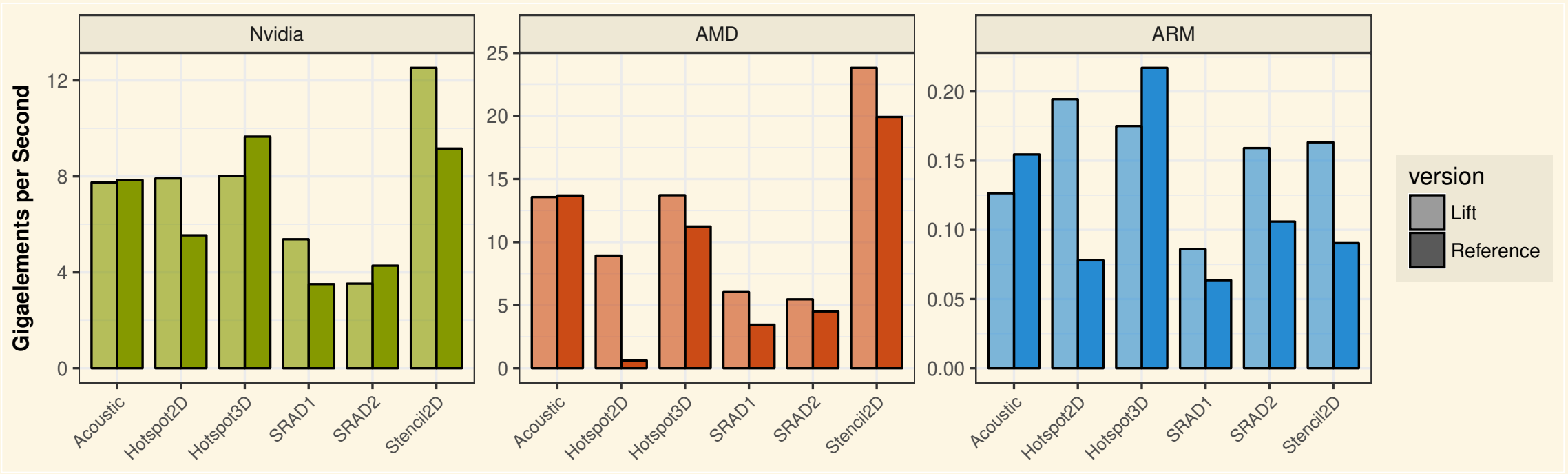
HIGH-PERFORMANCE

SPEEDUP OVER POLYHEDRAL COMPILATION (PPCG)

Lift achieves significant speedups compared to a state-of-the-art polyhedral compiler on 3 architectures



COMPARISON WITH HANDWRITTEN BENCHMARKS



Lift achieves the same - or even better - performance than hand optimized code

ACCEPTED AT CGO'18

B.HAGEDORN@WWU.DE
WWW.LIFT-PROJECT.ORG

TO BE CONTINUED...