University of BRISTOL

Performance Portability for Next-Generation Heterogeneous Systems

Dr Tom Deakin

Lecturer in Advanced Computer Systems

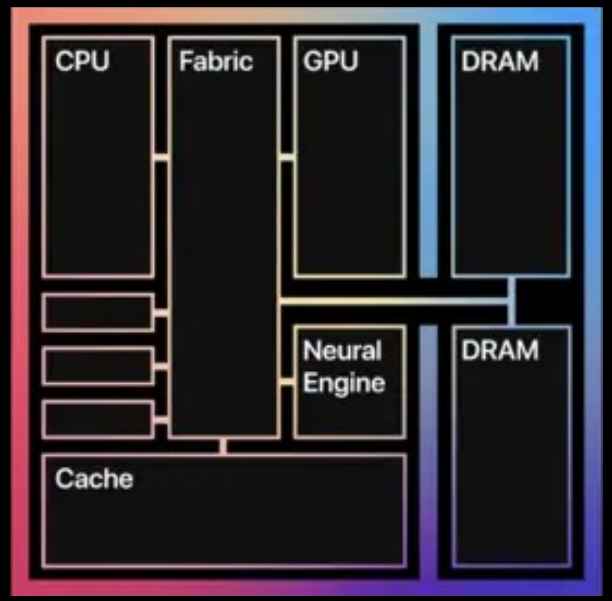
University of Bristol

Nov'23 Top500 Rank	System	Accelerator
1	Frontier	✅
2	Aurora	✅
3	Eagle	✅
4	Supercomputer Fugaku	❌
5	LUMI	✅
6	Leonardo	✅
7	Summit	✅
8	MareNostrum 5 ACC	✅
9	Eos NVIDIA DGX SuperPOD	✅
10	Sierra	✅

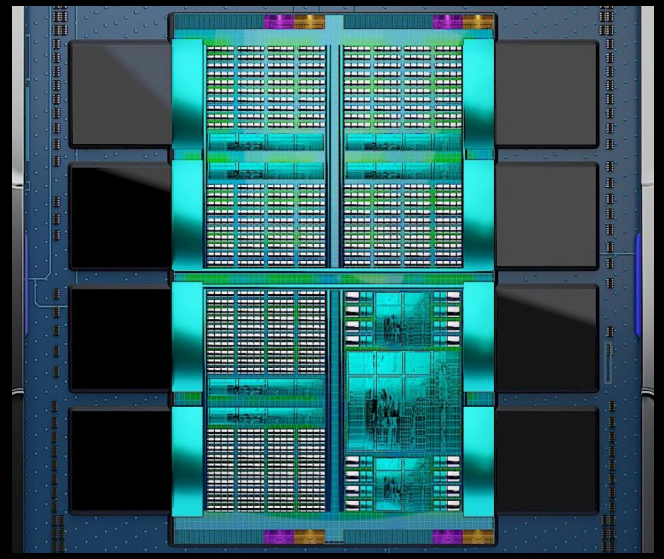
- “Complex” cores
- Instruction Level Parallelism
- Deep cache hierarchy
- NUMA
- Wide SIMD
- In-core accelerators

- More “simple” cores
- Very wide SIMD
- Fast context switching
- Programable memory hierarchy
- Latest memory technology

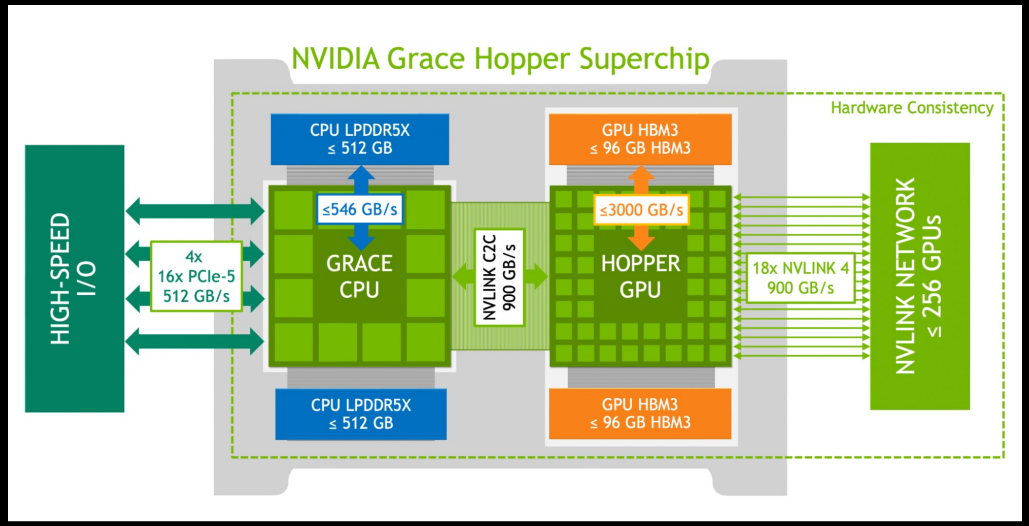
Apple M1



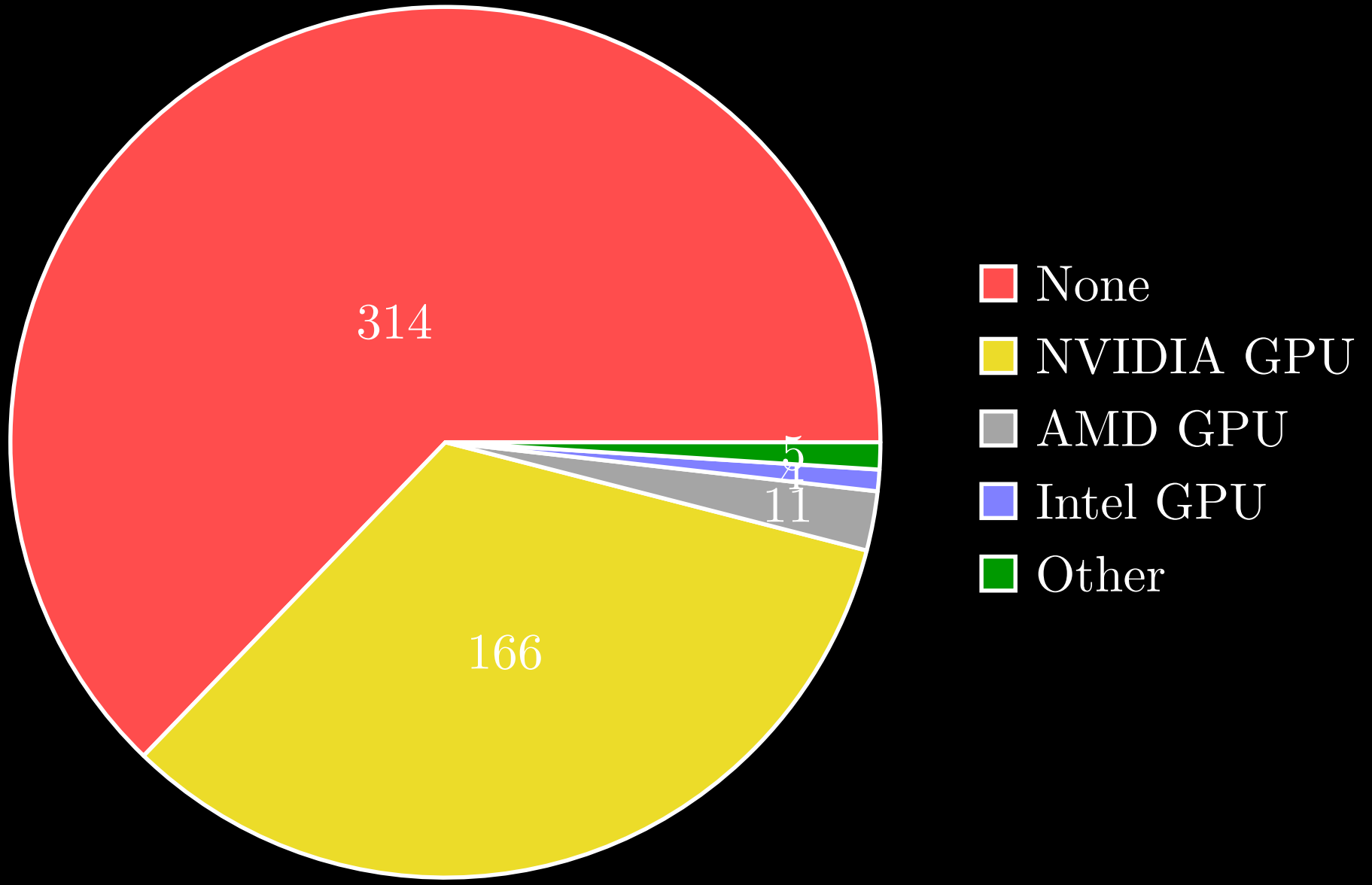
AMD MI300A



NVIDIA Grace-Hopper



Images belong to their respective owners



Data: TOP500 November 2023
Updated version of chart from: doi.org/10.1109/P3HPC56579.2022.00006

Tension between migrating to next system (which may be GPUs), and keeping running on current system

Performance, Portability, and Productivity

“A code is performance portable if it can achieve a similar fraction of peak hardware performance on a range of different target architectures”.

Problem

Application

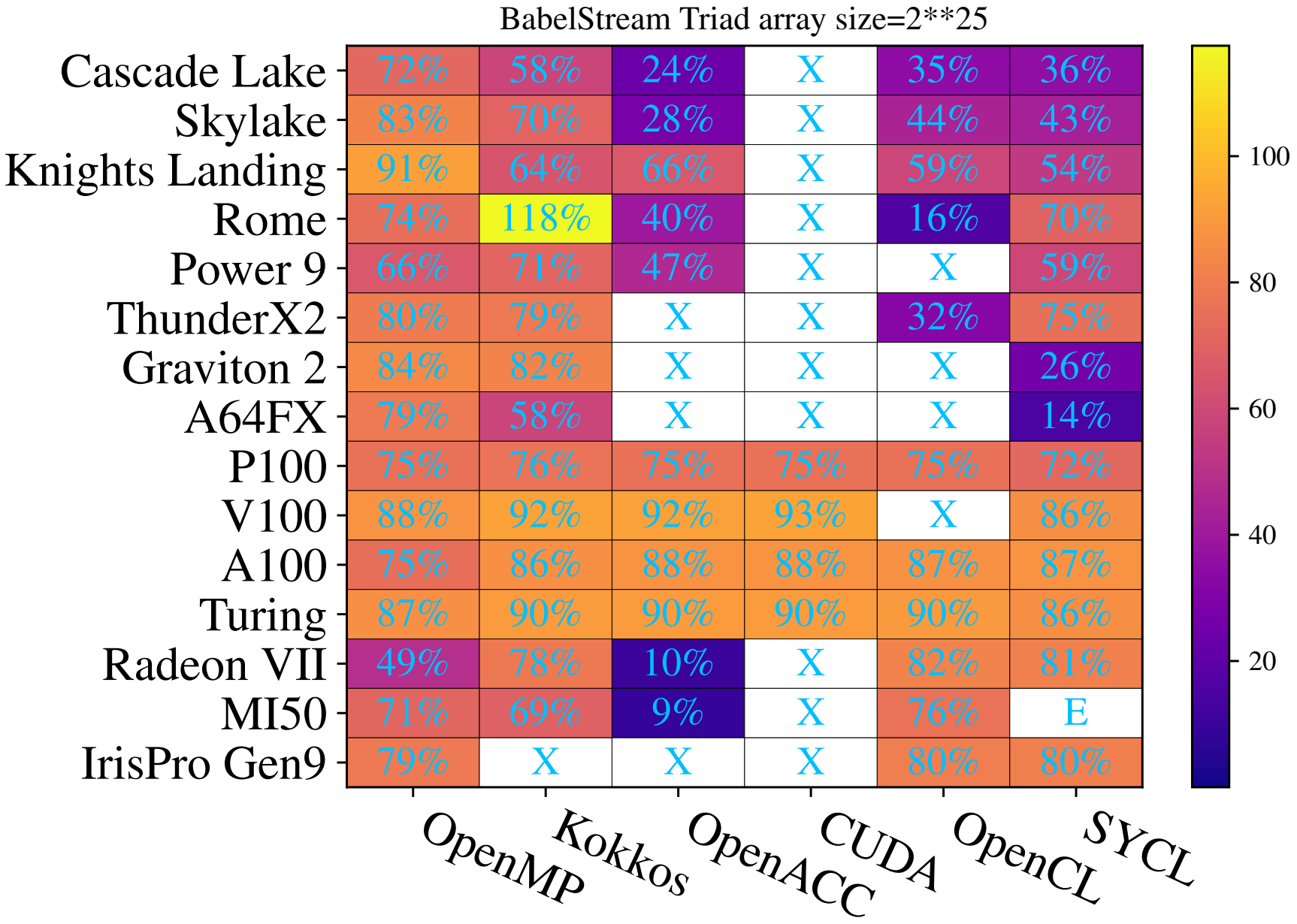
Platform

Efficiency

More details in doi.org/10.1109/P3HPC51967.2020.00007

$$\Phi(a,p,H) = \begin{cases} \dfrac{|H|}{\displaystyle\sum_{i \in H} \dfrac{1}{e_i(a,p)}} & \text{if, } \forall i \in H \\ & e_i(a,p) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

From Pennycook, Sewall and Lee: doi.org/10.1016/j.future.2017.08.007



From doi.org/10.1109/P3HPC51967.2020.00006

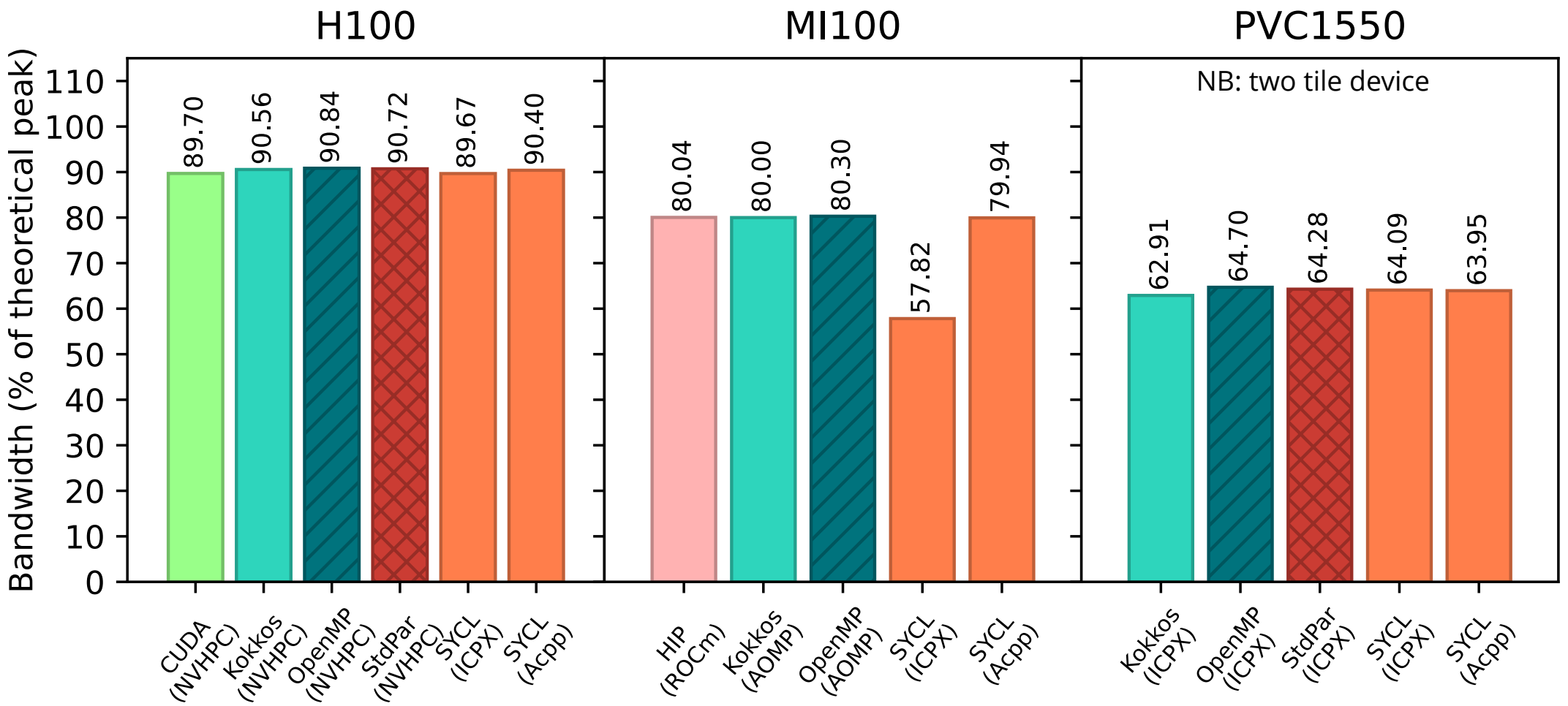
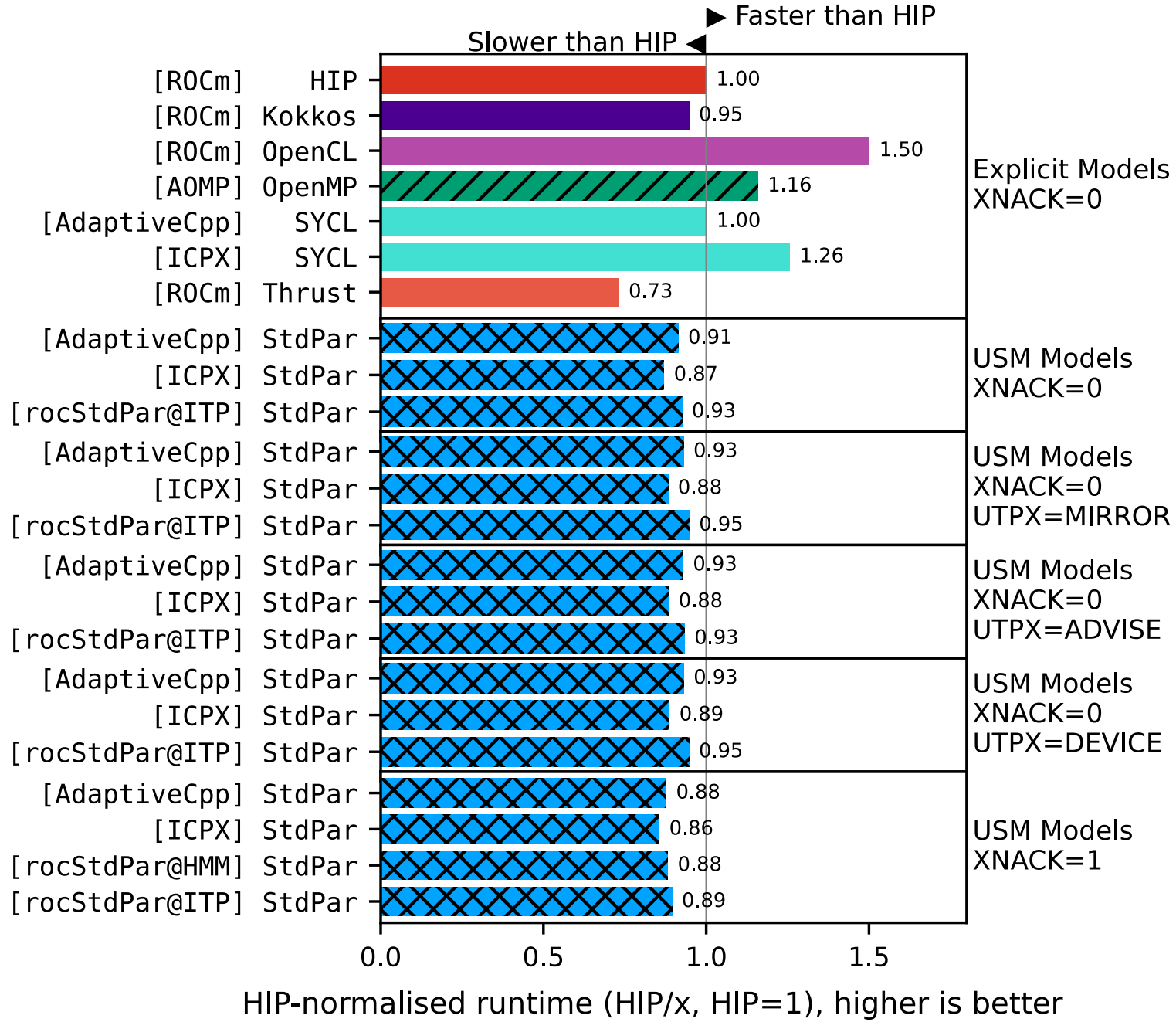


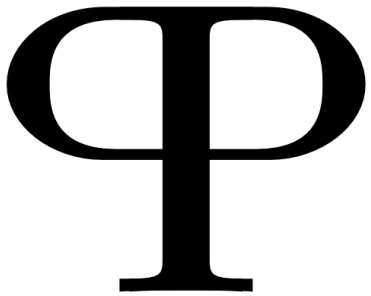
Figure from forthcoming publication

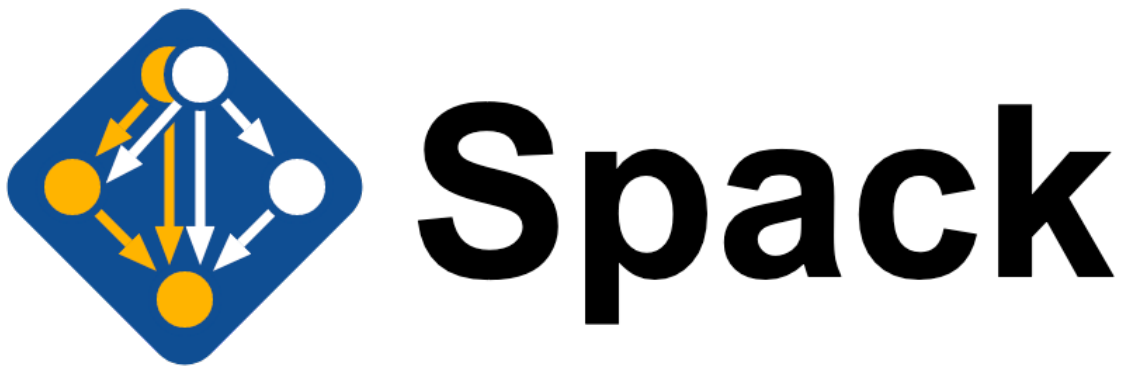
MiniBUDE

2024



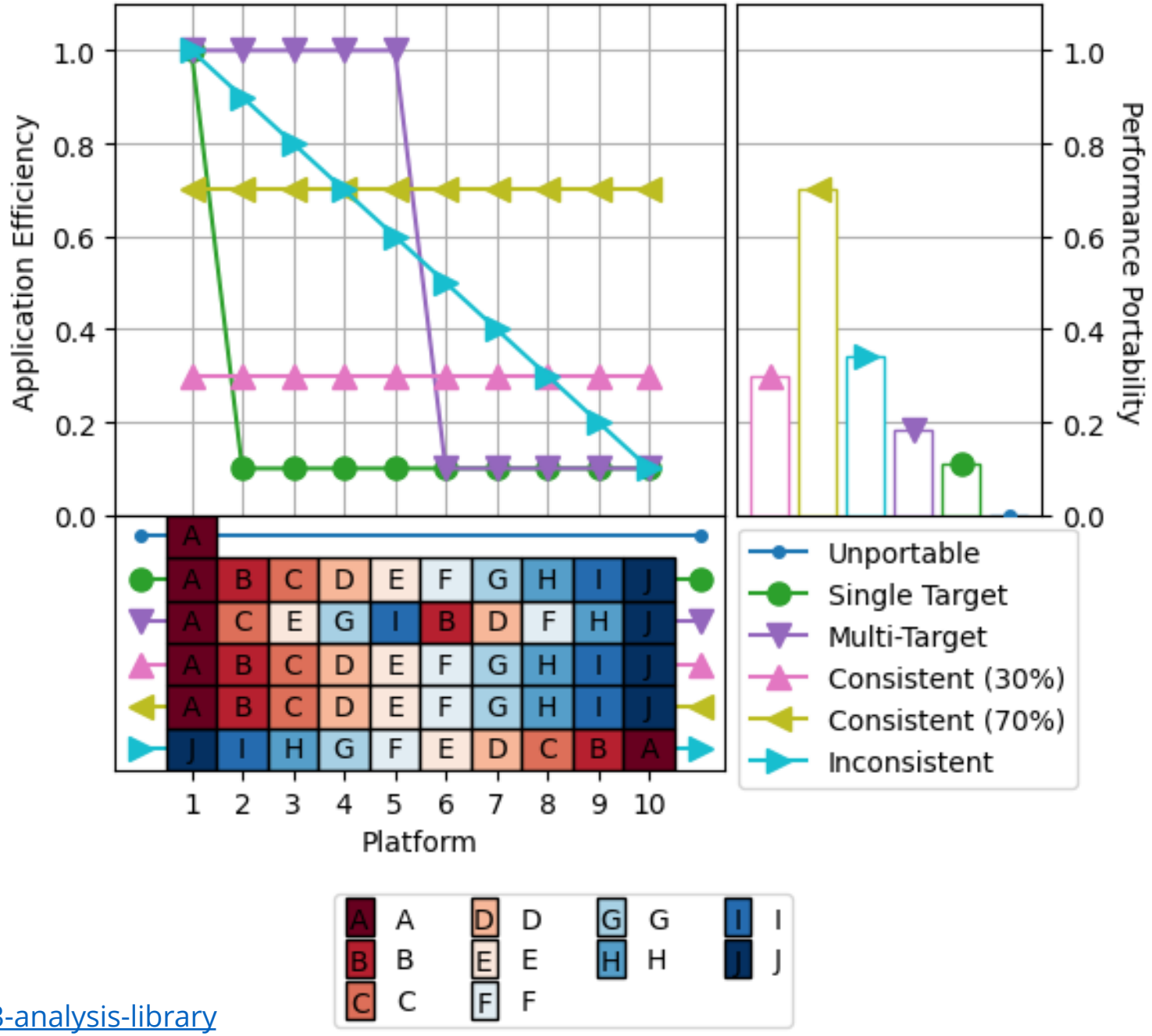
From doi.org/10.48550/arXiv.2401.02680



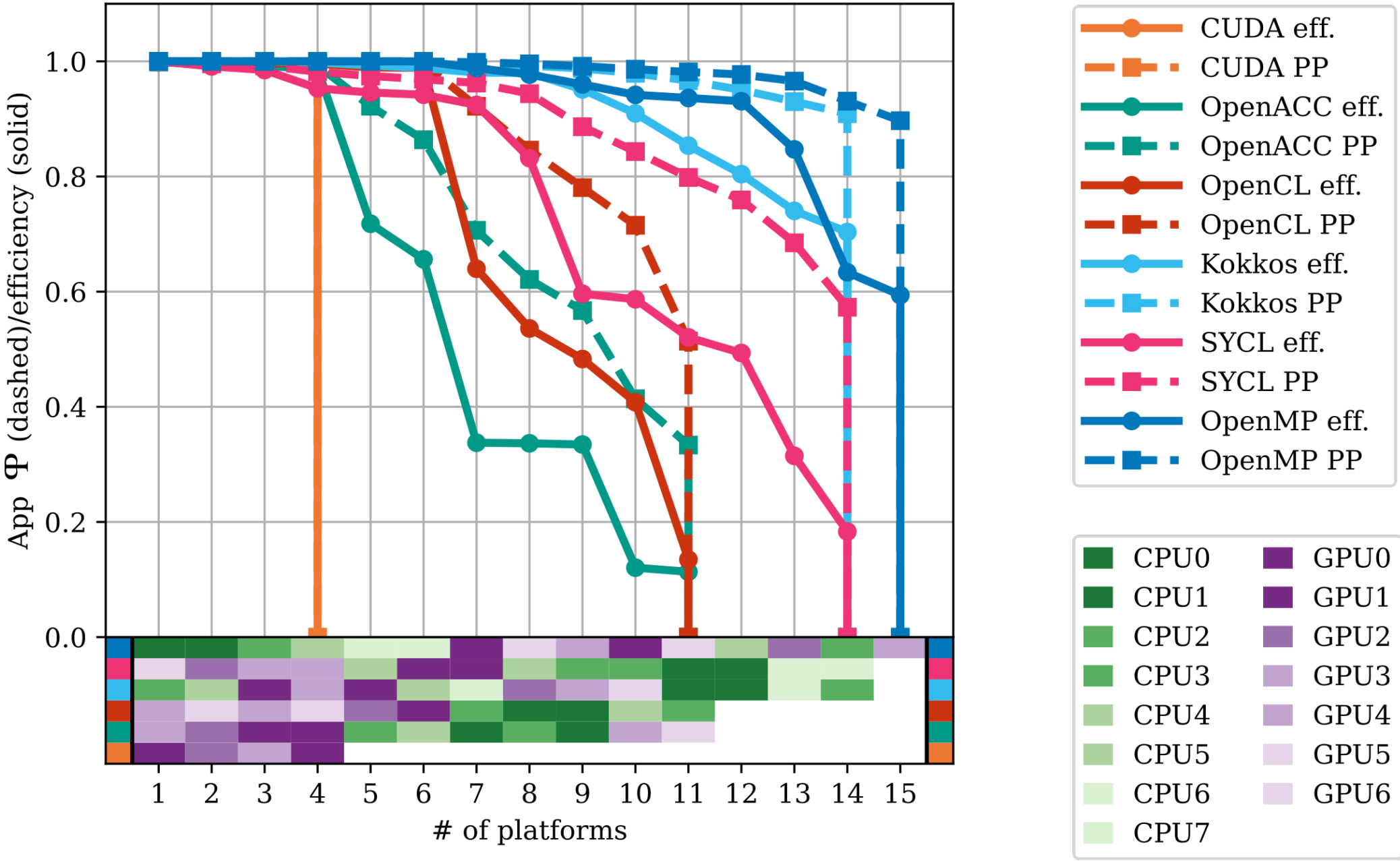


For more details, see doi.org/10.1145/3624062.3624133
and https://github.com/ukri-excalibur/excalibur-tests

This work was supported by the Engineering and Physical Sciences Research Council as part of ExCALIBUR Hardware & Enabling Software [EP/X031829/1]



From https://intel.github.io/p3-analysis-library
Based on doi.org/10.1109/MCSE.2021.3097276



From doi.org/10.1109/MCSE.2021.3097276

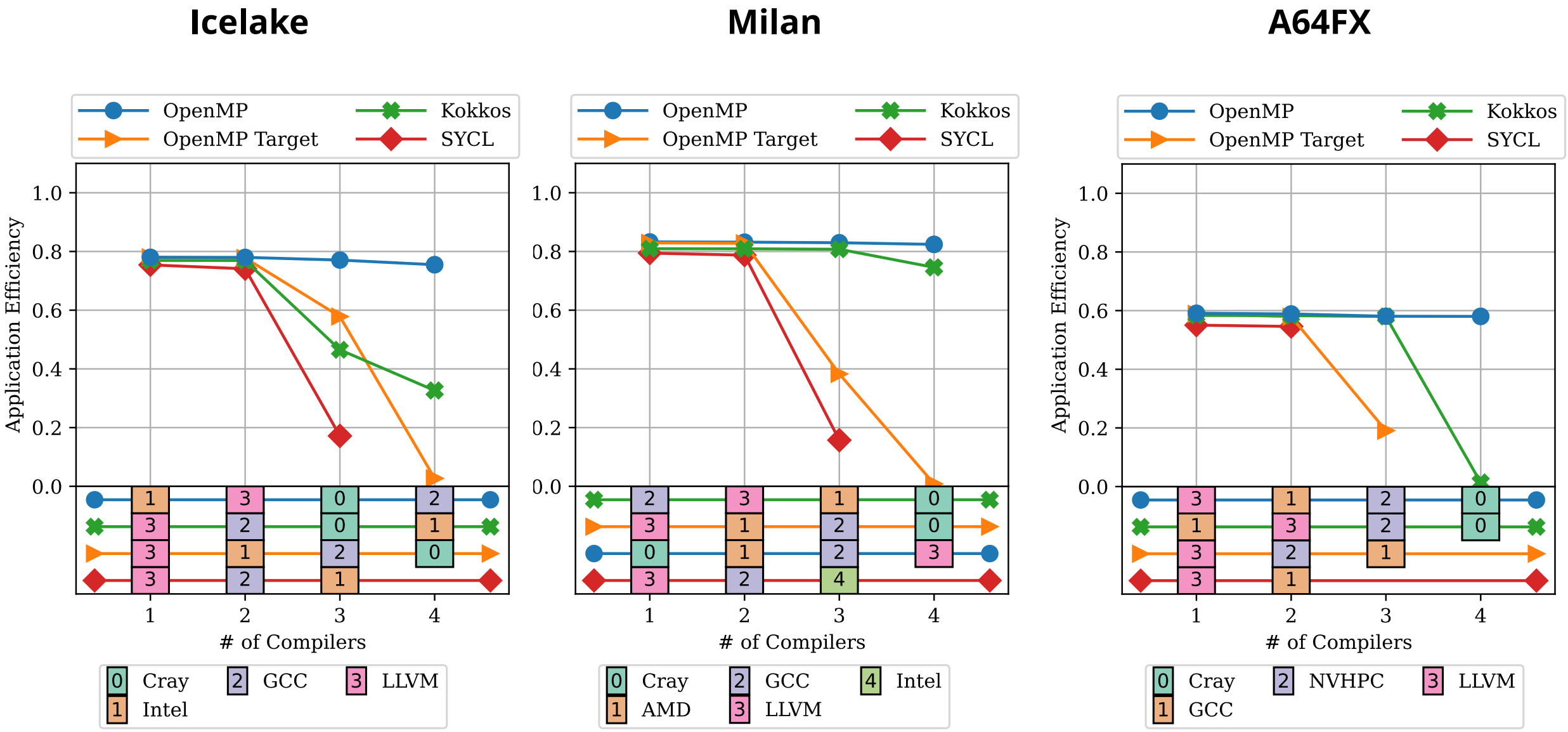
Specialisation?

OpenMP = OpenMP 1 + OpenMP 4/5 (+tasks) ?

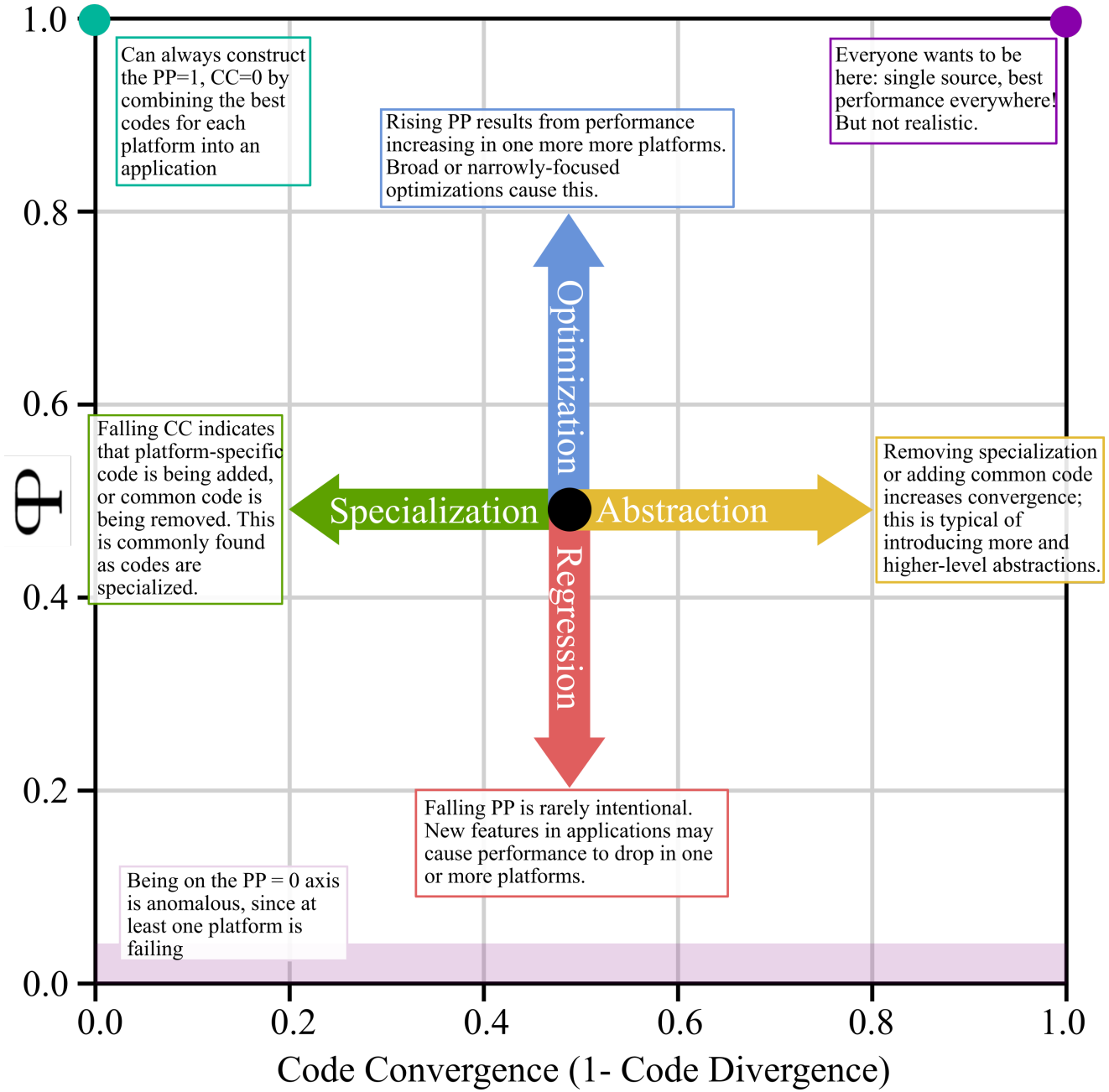
```
#pragma omp parallel for
for (int i = 0; i < N; ++i) {
    C[i] = A[i] + B[i];
}
```

```
#pragma omp target enter data \
map(alloc: A[:N], B[:N], C[:N])

#pragma omp target
#pragma omp loop
for (int i = 0; i < N; ++i) {
    C[i] = A[i] + B[i];
}

#pragma omp target exit data \
map(from: C[:N]) \
map(release: A[:N], B[:N])
```

BabelStream



From doi.org/10.1109/P3HPC56579.2022.00006



From doi.org/10.1109/P3HPC56579.2022.00006

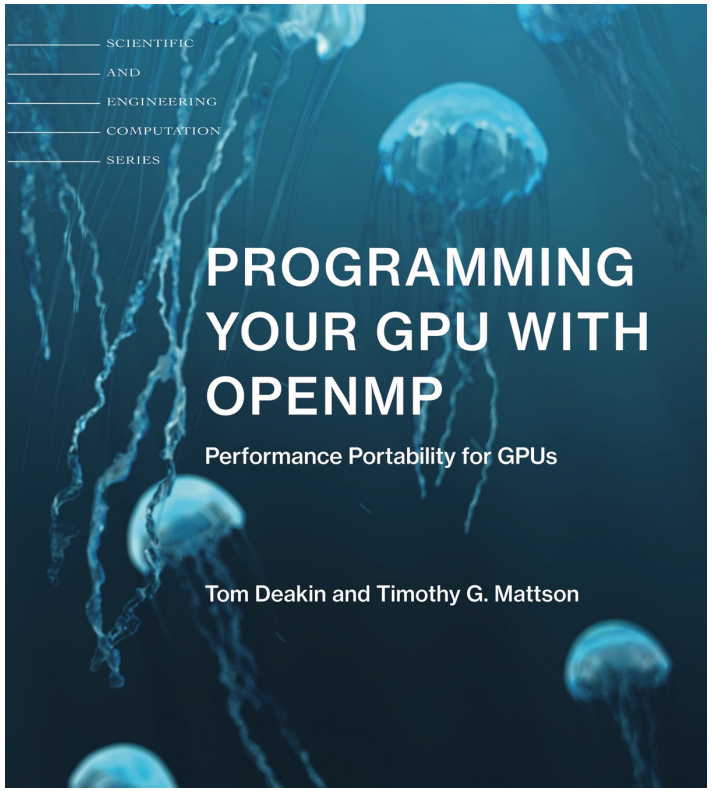
Device discovery and control

Data location and movement in discrete memory spaces

Expressing concurrent and parallel work

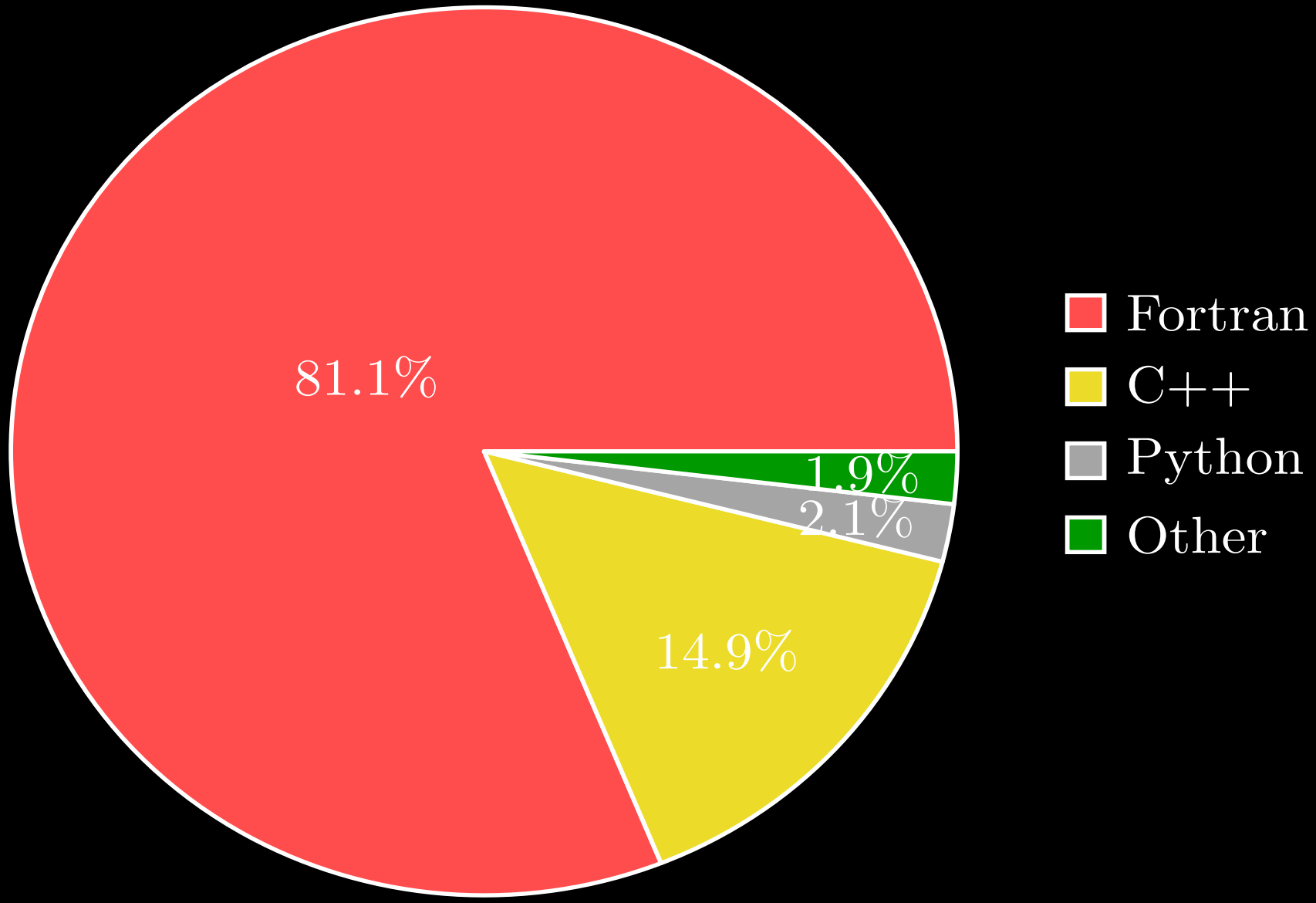
SYCL™

OpenMP®
Enabling HPC since 1997




IWOCL 2024
12th International workshop on open computing with OpenCL and SYCL
April 8-11, 2024 - Chicago, USA
iwocl.org
Full Program of Speakers and Registration
Supported by the
KHRONOS GROUP
SYCL
OpenCL

ARCHER2 Usage: March-August 2022



From doi.org/10.1109/PMBS56514.2022.00013

Prompt: A fight between parallel programming languages
Generated with AI on Microsoft Bing Image Creator · 6 December 2023 at 11:58 am

Develop with P3 in mind with Standard Parallelism

Use open-standards as confluent off-ramp to be productive today

Express all concurrent work asynchronously

Build in tuning parameters

Test all compilers & runtimes, on all systems, all the time

Tell your vendor

https://hpc.tomdeakin.com