Assessing Memory Bandwidth on ARCHER2 and LUMI Using CAMP

CUG 11.5.23, Helsinki

Wenqing Peng, Adrian Jackson and Evgenij Belikov*
EPCC, The University of Edinburgh

* corresponding author: e.belikov@epcc.ed.ac.uk

Outline

- Motivation
- CAMP Overview
- CAMP custom kernel example (dot product)
- Experimental Setup
- Results on ARCHER2, LUMI and NEXTGenIO
- Conclusions and Future Work
- Discussion

Machine Imbalance

- Memory Wall and growing machine imbalance
- NUMA architectures with increasing number of cores
- Many apps that were considered compute-bound now memory-bound

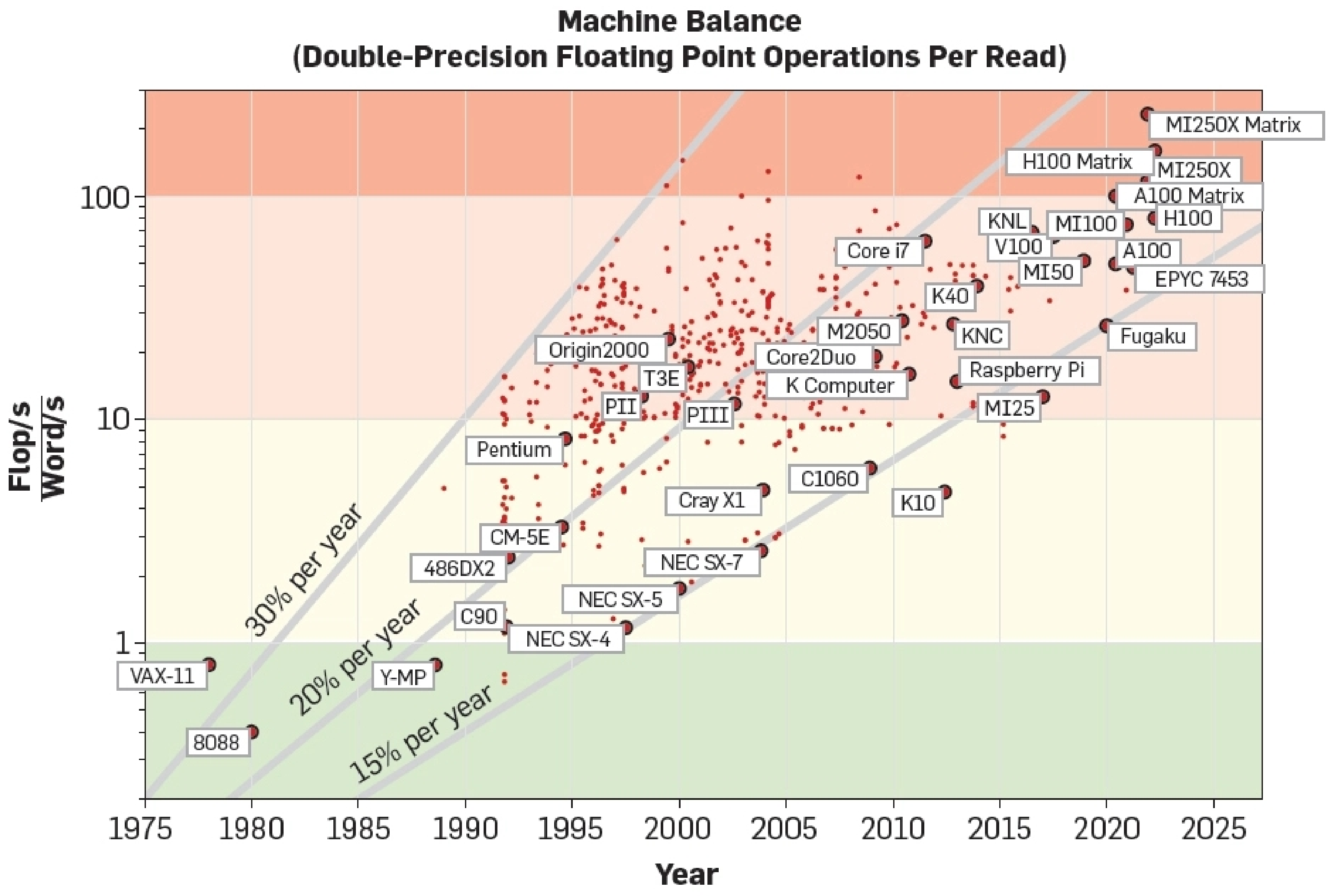
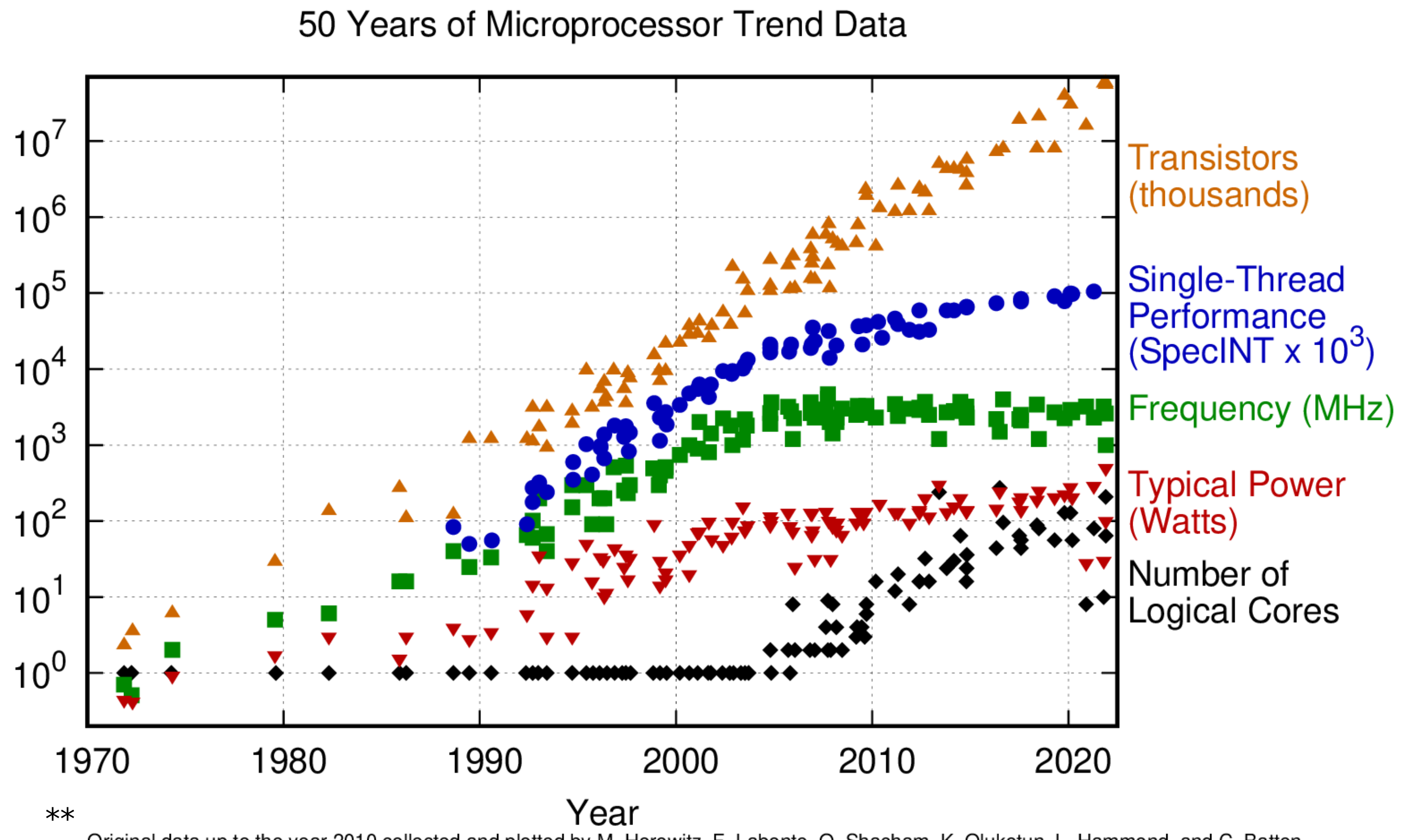


Fig.: J Dongarra, Turing Award lecture: "The evolution of mathematical software." CACM 65.12 (2022): 66-72

Deep Memory Hierarchies

- Deeper memory hierarchies to accommodate larger core counts
- Memory access: one of the main intra-node bottlenecks
- Various access patterns and operational intensities in kernels → CAMP

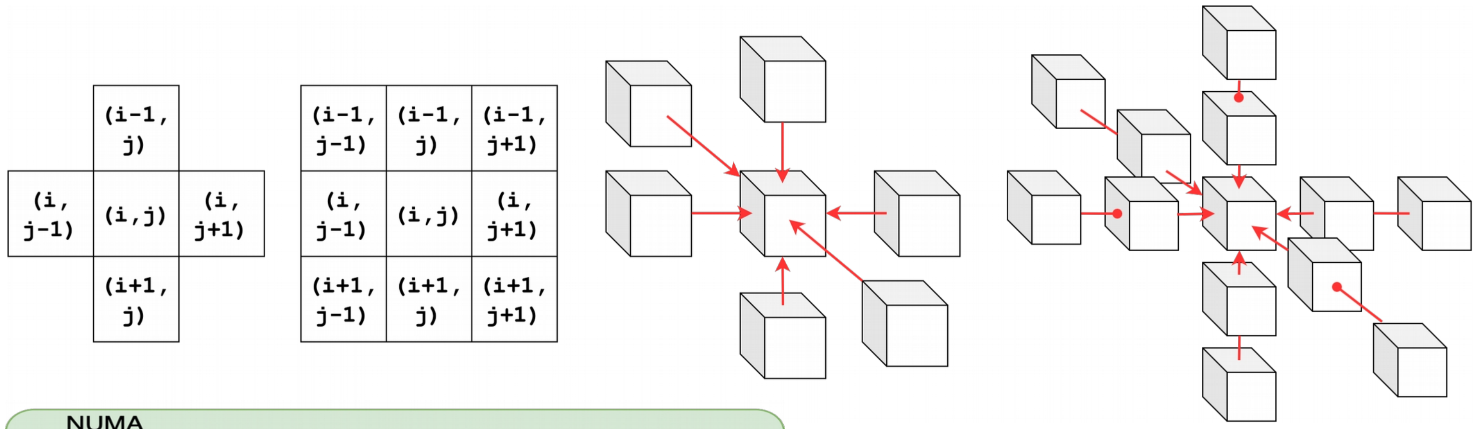


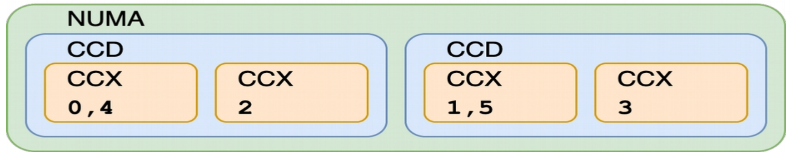
** Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

** https://github.com/karlrupp/microprocessor-trend-data

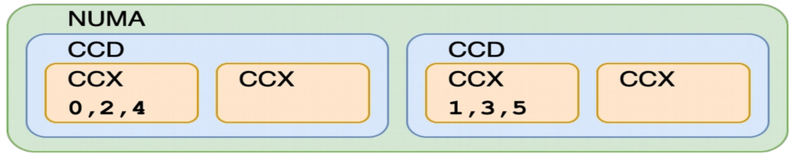
Introduction to CAMP (Configurable App for Memory Probing)

- Operational intensity (OI)
- Different access patterns reflecting *spatial* and *temporal locality*
- Thread pinning →
- Custom kernels
- Performance Counters
- Portability

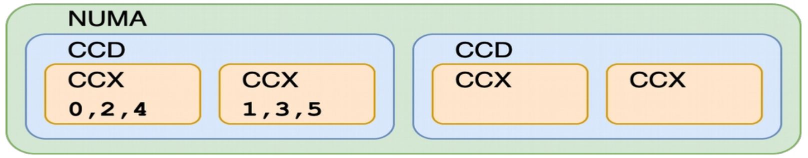




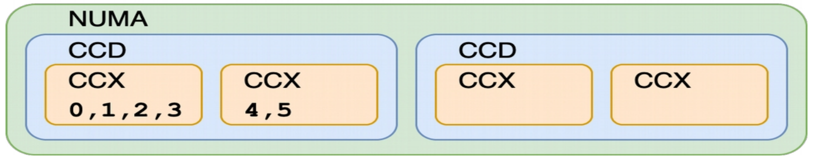
CCD and CCX aware



CCX ignored



CCD ignored



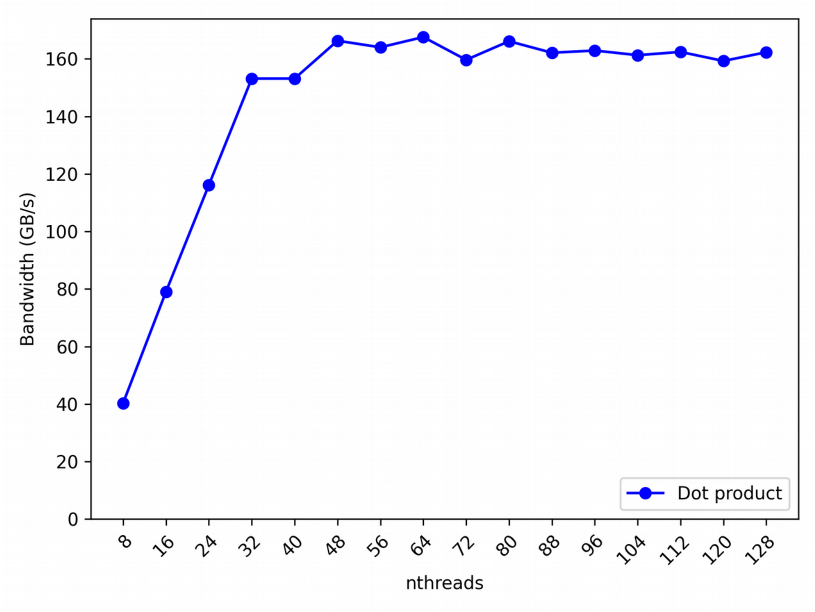
CCD and CCX ignored

* for more details see:
CAMP: a Synthetic Micro-Benchmark for Assessing Deep Memory Hierarchies. HiPar@SC'22: 28-36

epcc

CAMP: Extensions

- More fine-grained OI (from 0.08 to 0.02)
- Energy consumption analysis
- Support for custom kernels: *dot product*
- New way to configure runs via a config file and using an outer-level driver script



Algorithm 1 CAMP custom kernel (pseudocode)

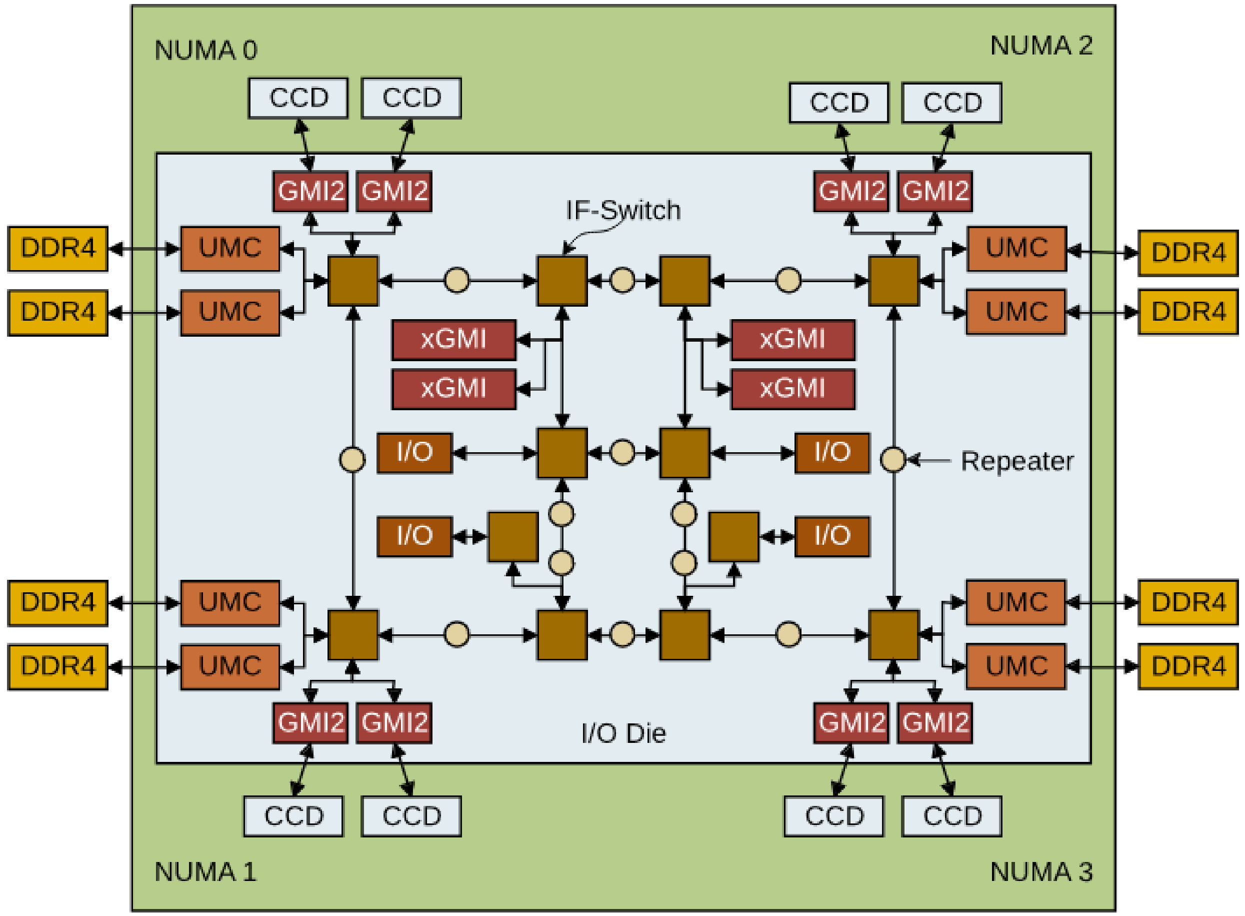
```
procedure CAMP_PREPROCESS
    #pragma omp parallel default(none) shared(a, b) {
    int tid = omp_get_thread_num();
    a[tid] = malloc(size);
    b[tid] = malloc(size);
    for i in range(1, size) do
        a[tid][i] = 1.0;
        b[tid][i] = 2.0;
    end for
    }
end procedure

procedure CAMP_KERNEL
    #pragma omp parallel default(none) shared(a, b) reduction(+:sum) {
    int tid = omp_get_thread_num();
    for i in range(1, size) do
        sum += a[tid][i] * b[tid][i];
    end for
    }
end procedure

procedure CAMP_POSTPROCESS
    #pragma omp parallel default(none) shared(a, b) {
    int tid = omp_get_thread_num();
    free(a[tid]);
    free(b[tid]);
    }
end procedure
```

Experimental Setup

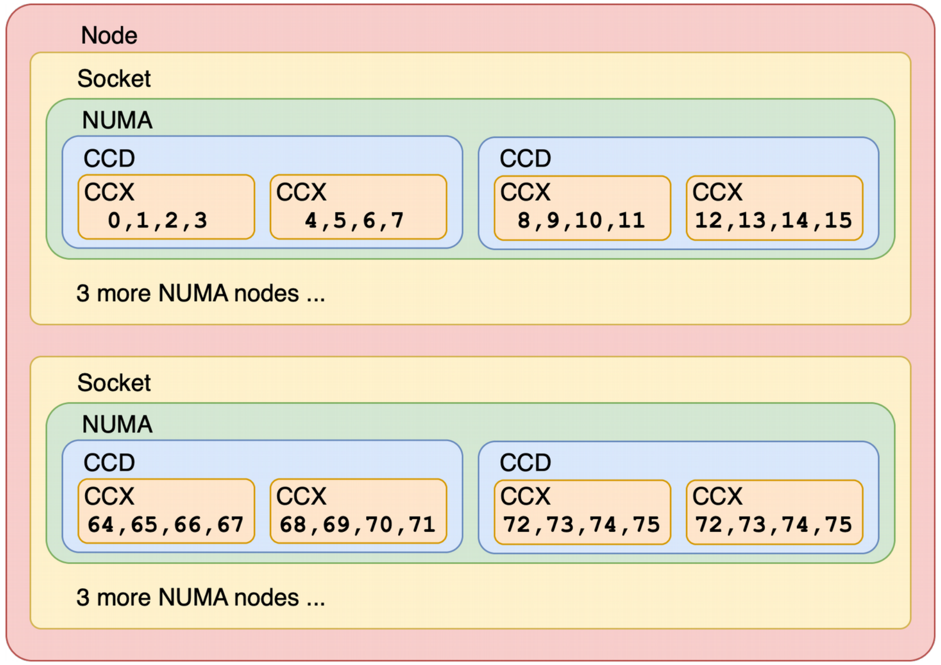
- ARCHER2: AMD Rome, node hierarchy on the right*
- LUMI: AMD Milan
- NEXTGenIO: Intel Ice Lake
- CAMP cyclic distribution (similar to OpenMP *spread*)
- data size >> L3 cache size
- optimisation on, vectorisation on
- best of 10 runs (raw data kept)



* from Velten et al.,"Memory Performance on AMD Rome and Intel Cascade Lake SP Server Processors", ICPE'22

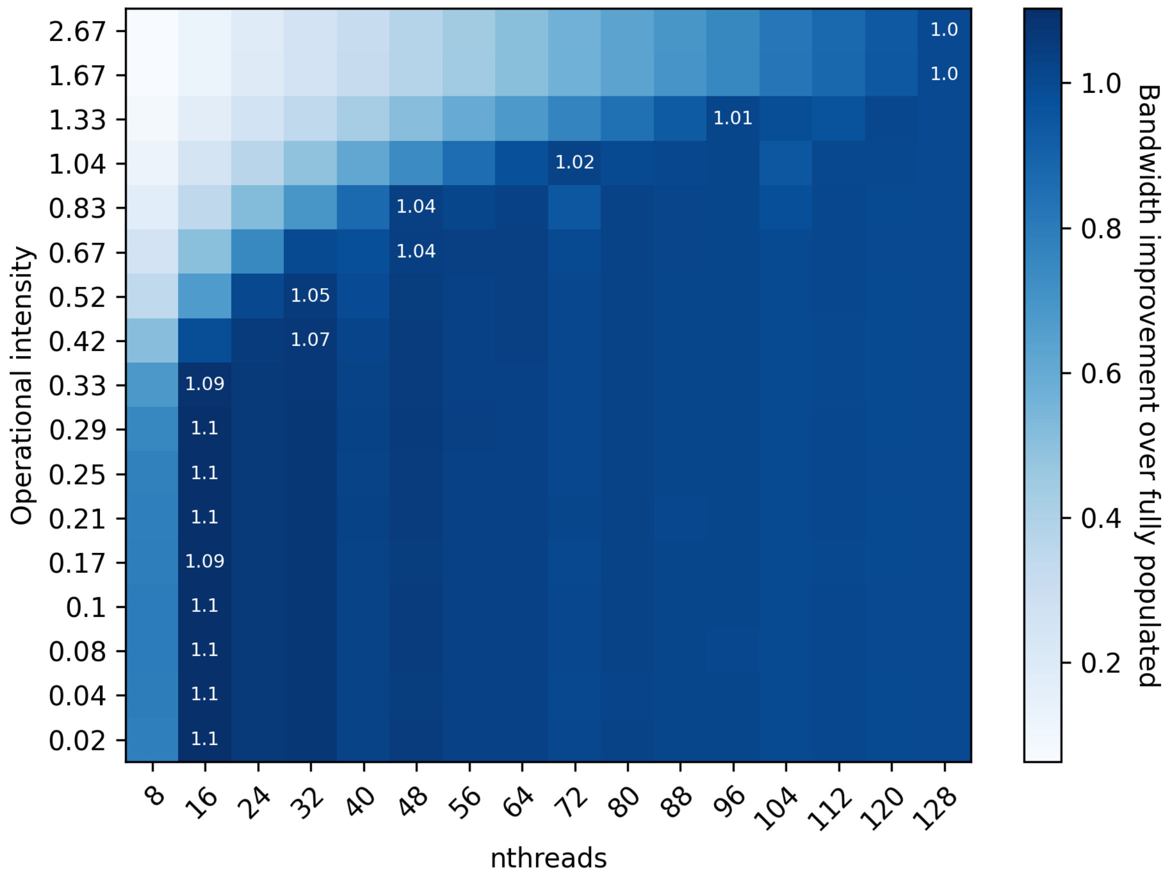
Systems Overview: ARCHER2, LUMI-C, NEXTGenIO-icx

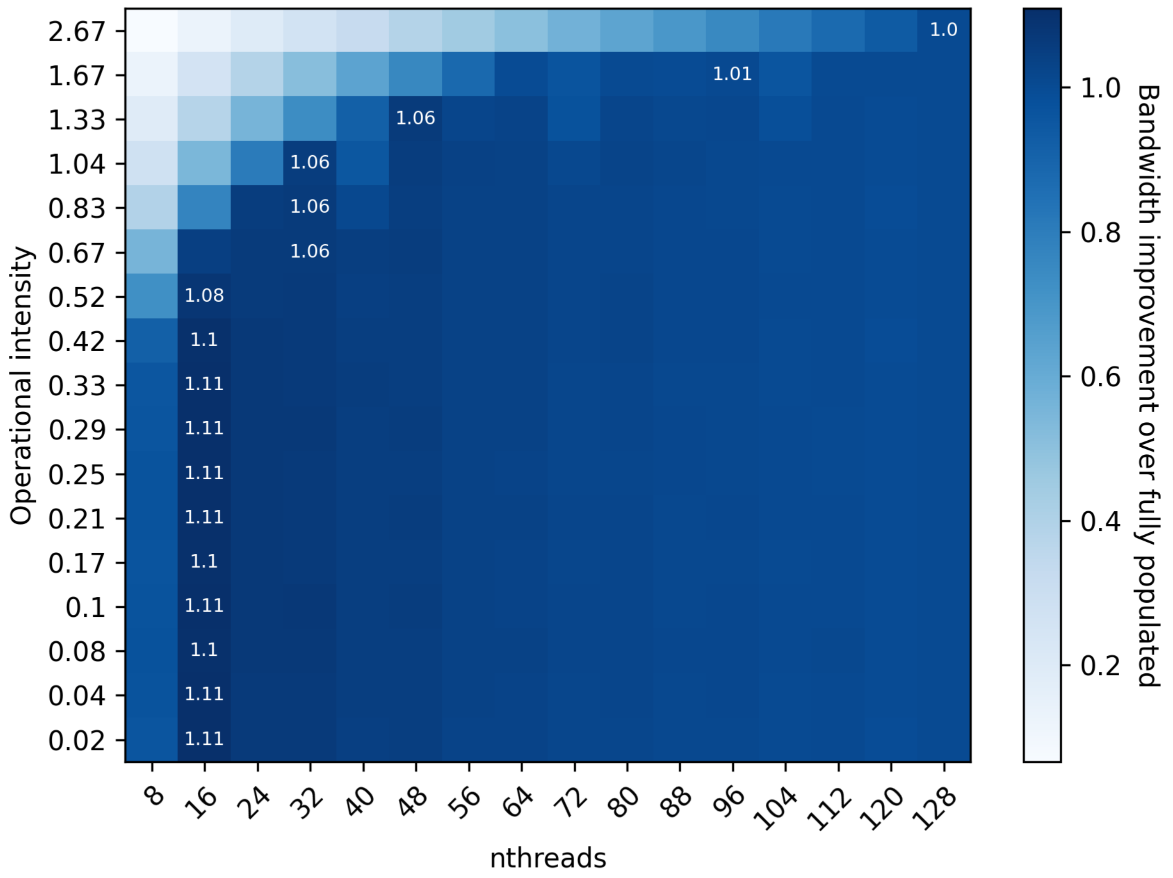
	ARCHER2	LUMI-C	NEXTGenIO-icx
Hardware			
Architecture	AMD Rome	AMD Milan	Intel Ice Lake
Model	EPYC 7742	EPYC 7763	Xeon Gold 6330
Cores per node	128 (2x64)	128 (2x64)	56 (2x28)
CPU freq. (GHz)	2.25	2.45	2.00
L1$ (KB)	32	32	48
L2$ (KB)	512	512	1280
L3$ (MB)	16x16	8x32	42
RAM (GB)	256	256	256
Software			
OS	SLES 15	SLES 15	CentOS 7.9
Compiler	PrgEnv-cray	PrgEnv-cray	Intel oneAPI 2023
Python	3.8.5	3.9.13	3.8.5
numpy	1.24.2	1.20.0	1.24.2
pandas	1.5.3	1.3.4	1.5.3
matplotlib	3.7.1	3.7.1	3.7.1



ARCHER2 node hierarchy

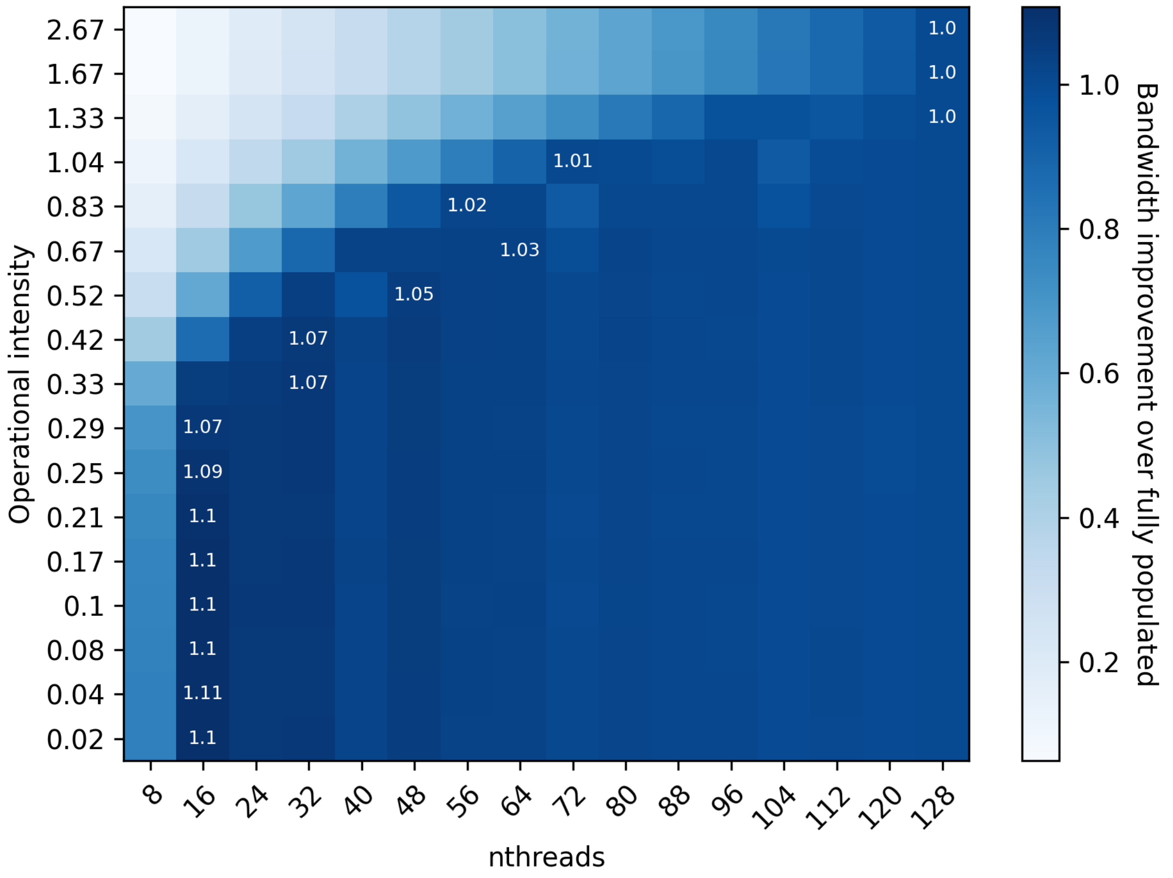
ARCHER2 Results: Contiguous (1.50 GHz vs 2.25 GHz CPU freq.)

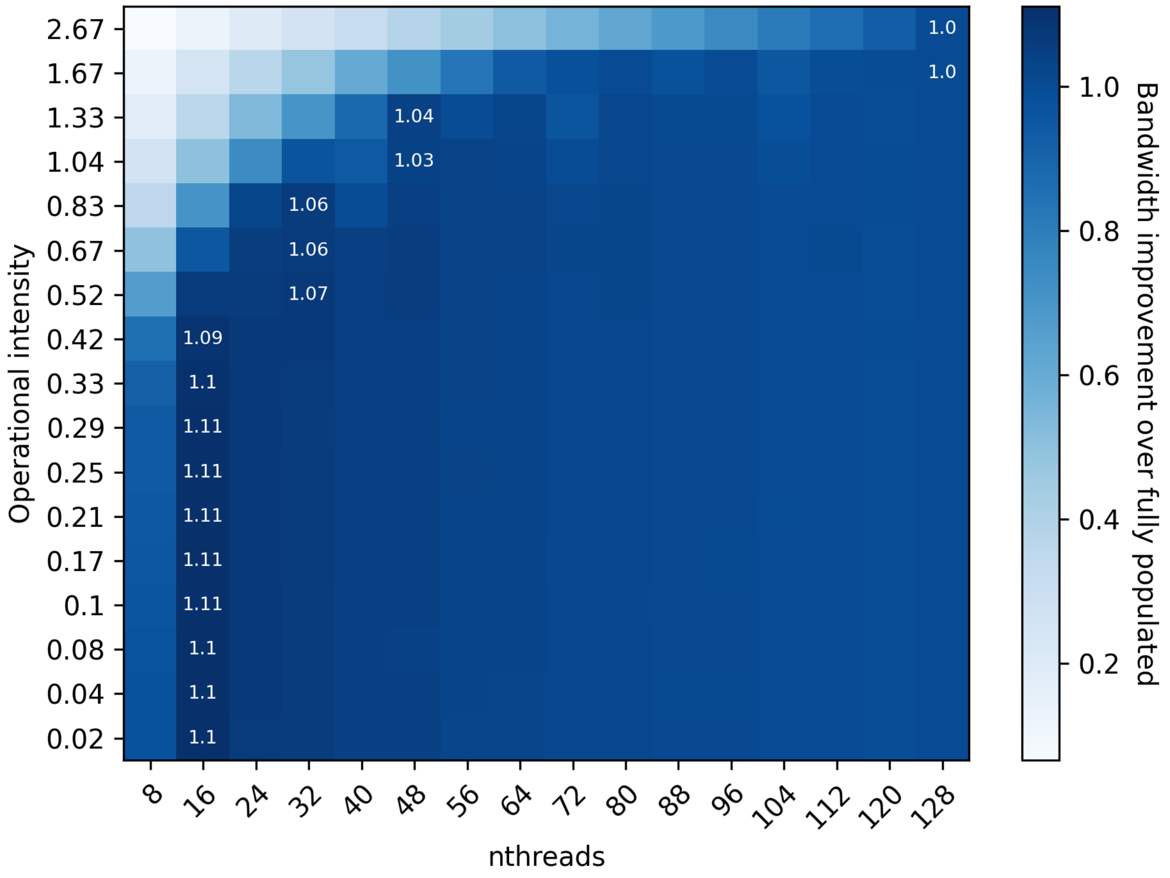




- Highest bandwidth in most cases with 16 threads (up to 11% increase); stronger effect for higher CPU freq.

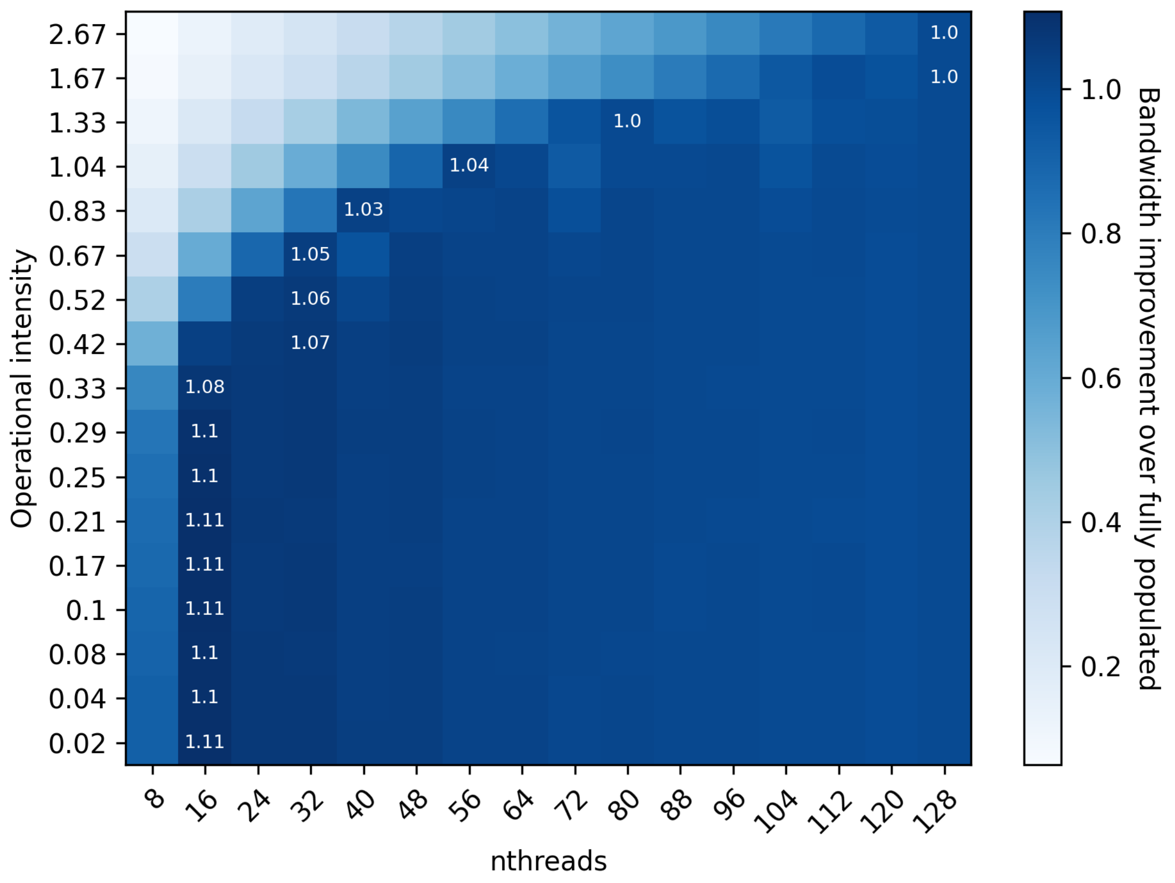
ARCHER2 Results: Stride4 (1.50 GHz vs 2.25 GHz CPU freq.)

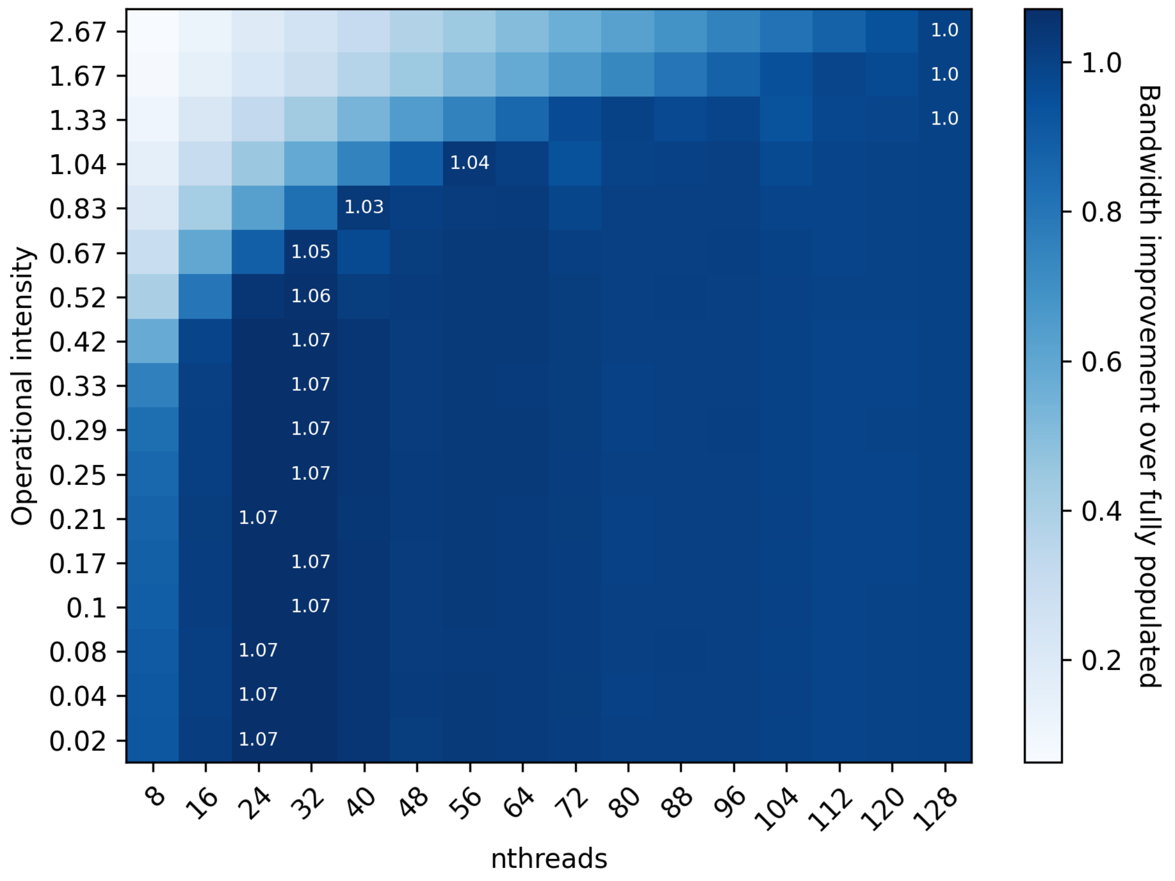




- Highest bandwidth in most cases with 16 threads (up to 11% increase); stronger effect for higher CPU freq.

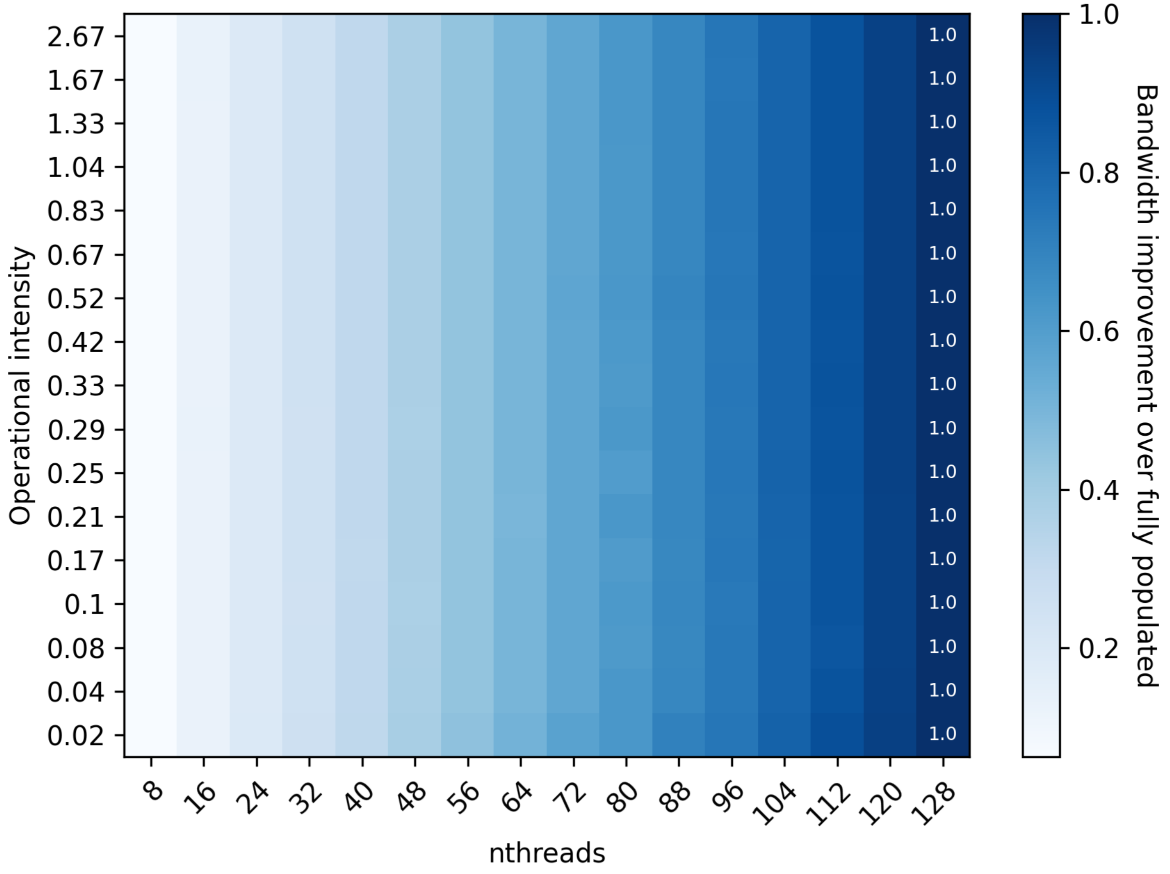
Effect of sub-optimal hierarchy setting (ARCHER2 Stride4 2 GHz)

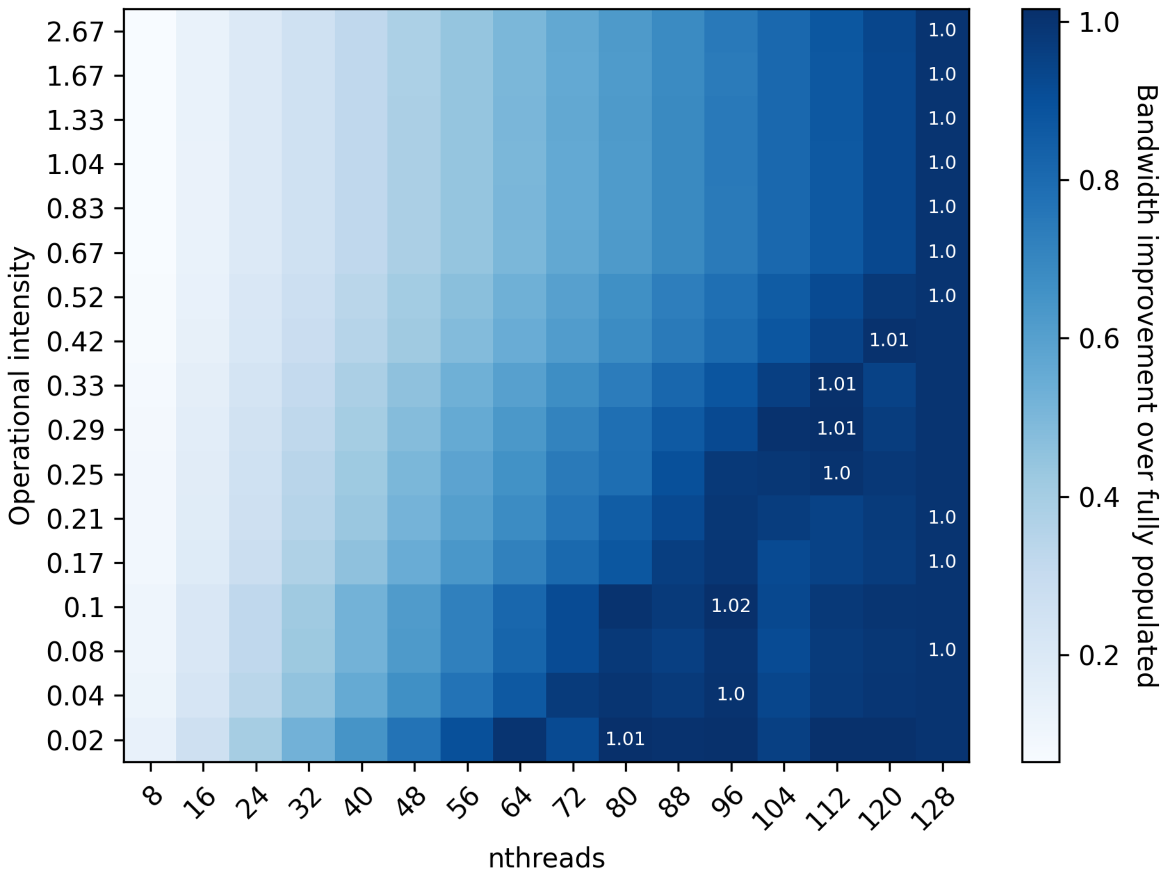




- Need to take case when specifying thread placement at sub-NUMA region level (NB: poor settings on the right)

ARCHER2 Results: Stencil5 (1.50 GHz vs 2.25 GHz CPU freq.)

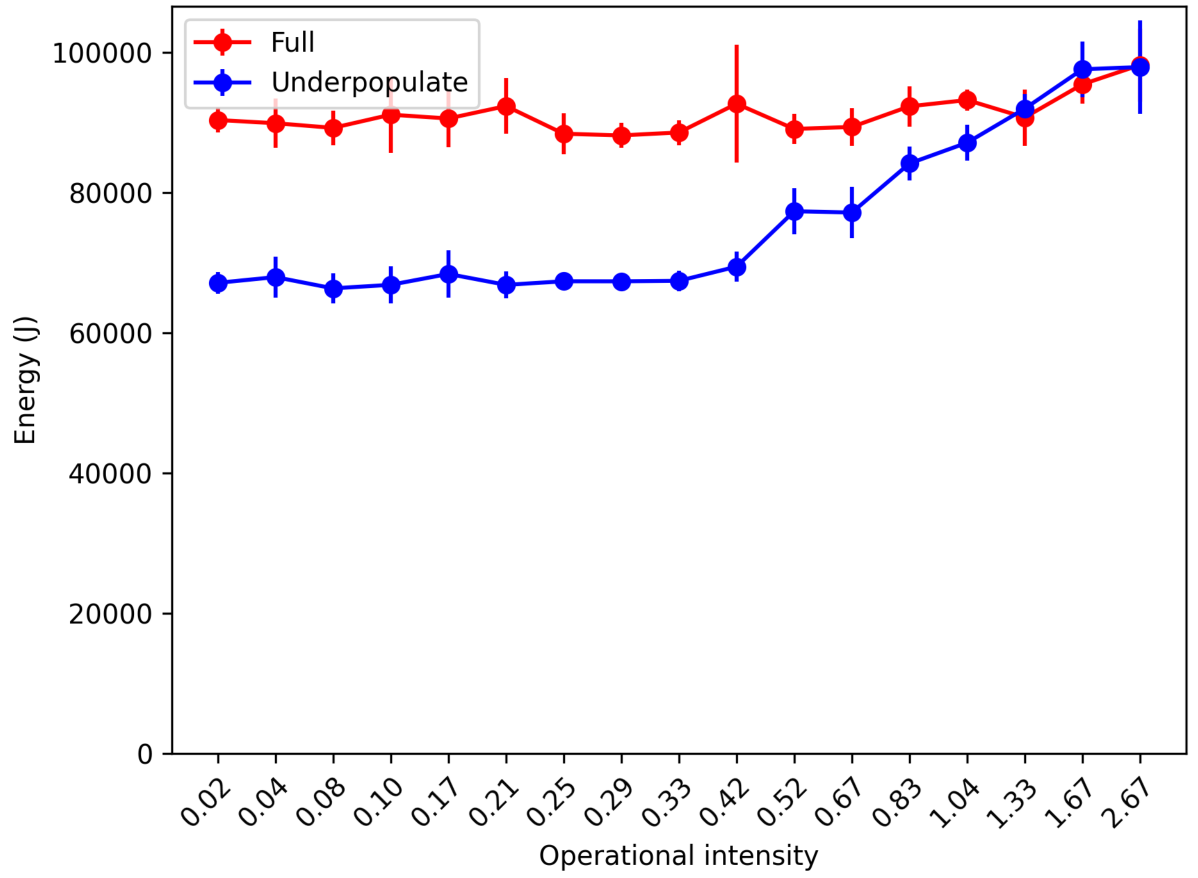


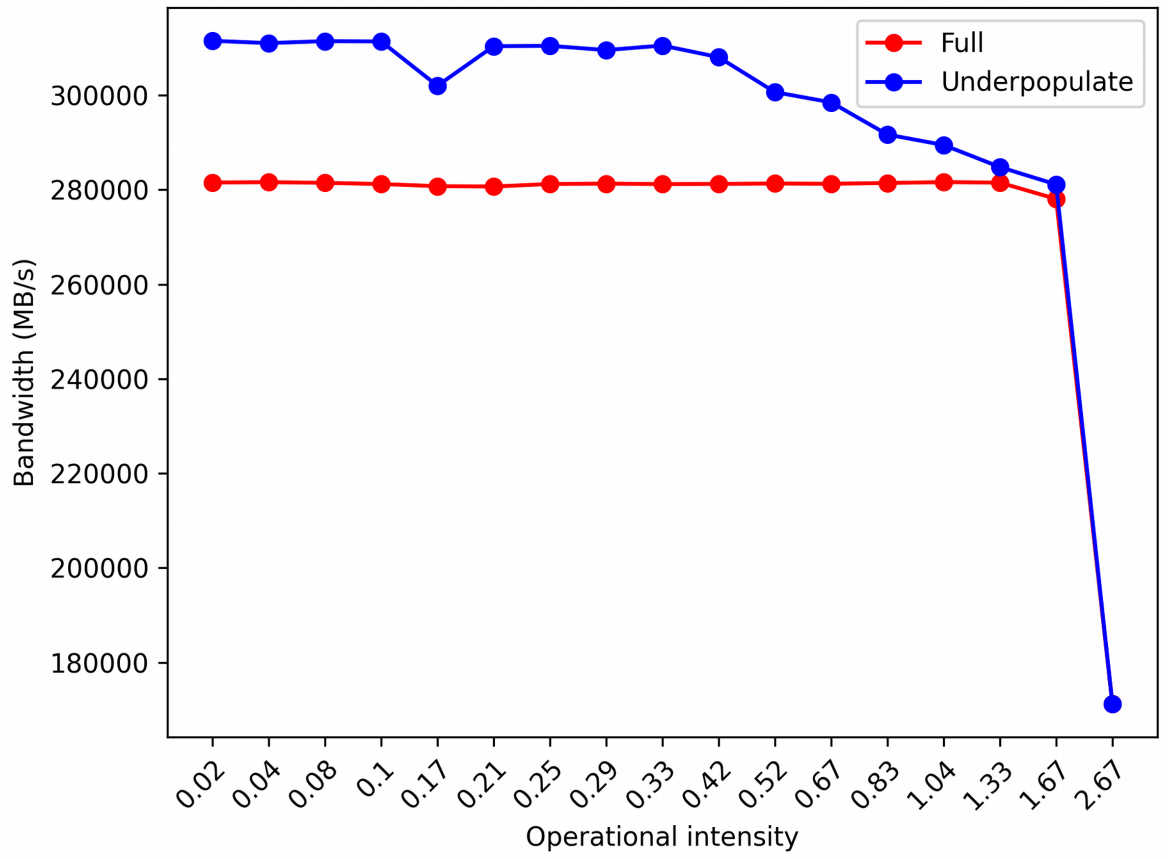


- Effect only visible for higher CPU freq.
  (e.g. +2% using 96 threads for operational intensity of 0.1)

epcc

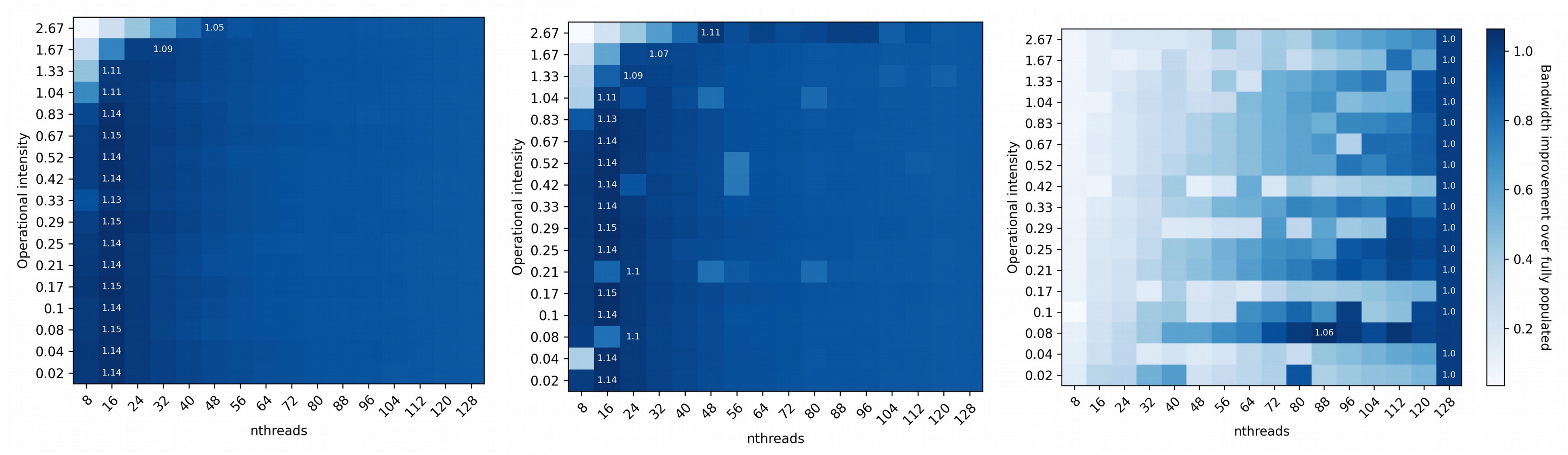
ARCHER2: Energy Consumption and Absolute Bandwidth





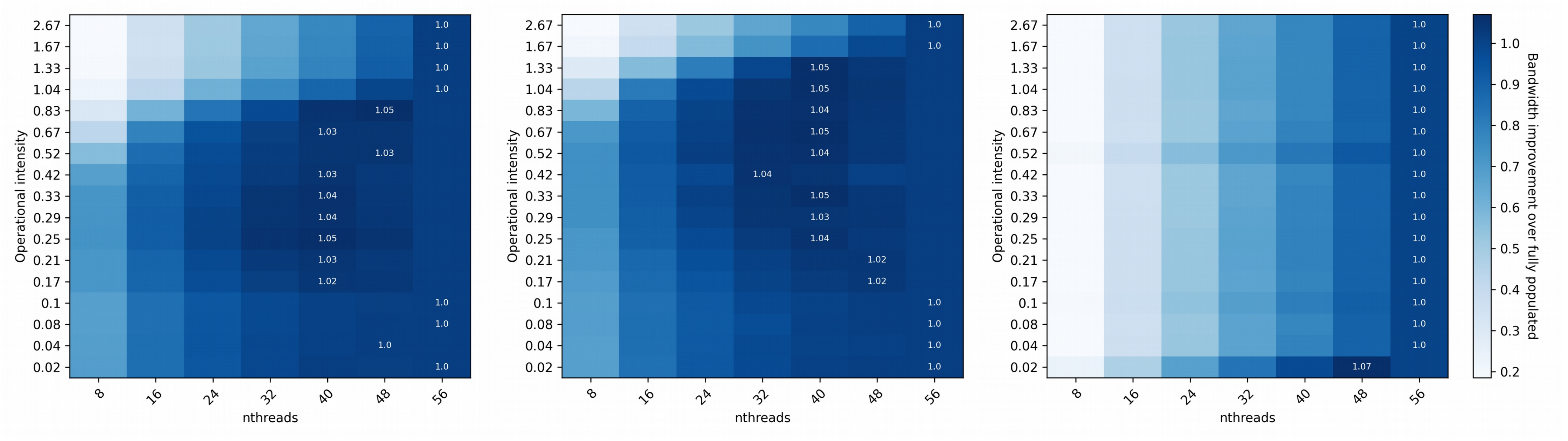
- Contig. access: underpopulated case has lower energy cost (r=1000)
- Absolute bandwidth higher for underpopulated setup

LUMI Results: Contiguous, Stride4, Stencil5 (default CPU freq)



- Highest bandwidth in most cases with 16 threads (up to +15%) for contiguous and strided access patterns
- Unclear patchy behaviour for stencil (for <128 threads)

NEXTGenIO Results: Contiguous, Stride4, Stencil5



- Highest bandwidth in most cases with 40 threads (up to +5%) for contiguous and strided access patterns
- For stencil +7% only for smallest operational intensity

Conclusions

- CAMP is an open-source tool complementing existing benchmarks with focus on assessing intra-node memory hierarchy and support for different access patterns with tunable operational intensity
  → *https://github.com/CAMP-benchmark/CAMP*
- We observe underpopulation effects (up to 11% and 15% increased bandwidth on ARCHER2 and LUMI-C respectively, when using only 16 threads rather than 128); the effect appears stronger for higher CPU frequency settings
- Using lower CPU frequency setting on ARCHER2 results in increased runtime and energy consumption
- Underpopulation may help reduce energy consumption

Future Work

- Next we plan to extend CAMP by adding GPU support
- Increase the number of supported access patterns
- Grow the library of kernels and compare to efficient implementations (e.g. BLAS, FFT)
- Apply CAMP to cost modelling for applications

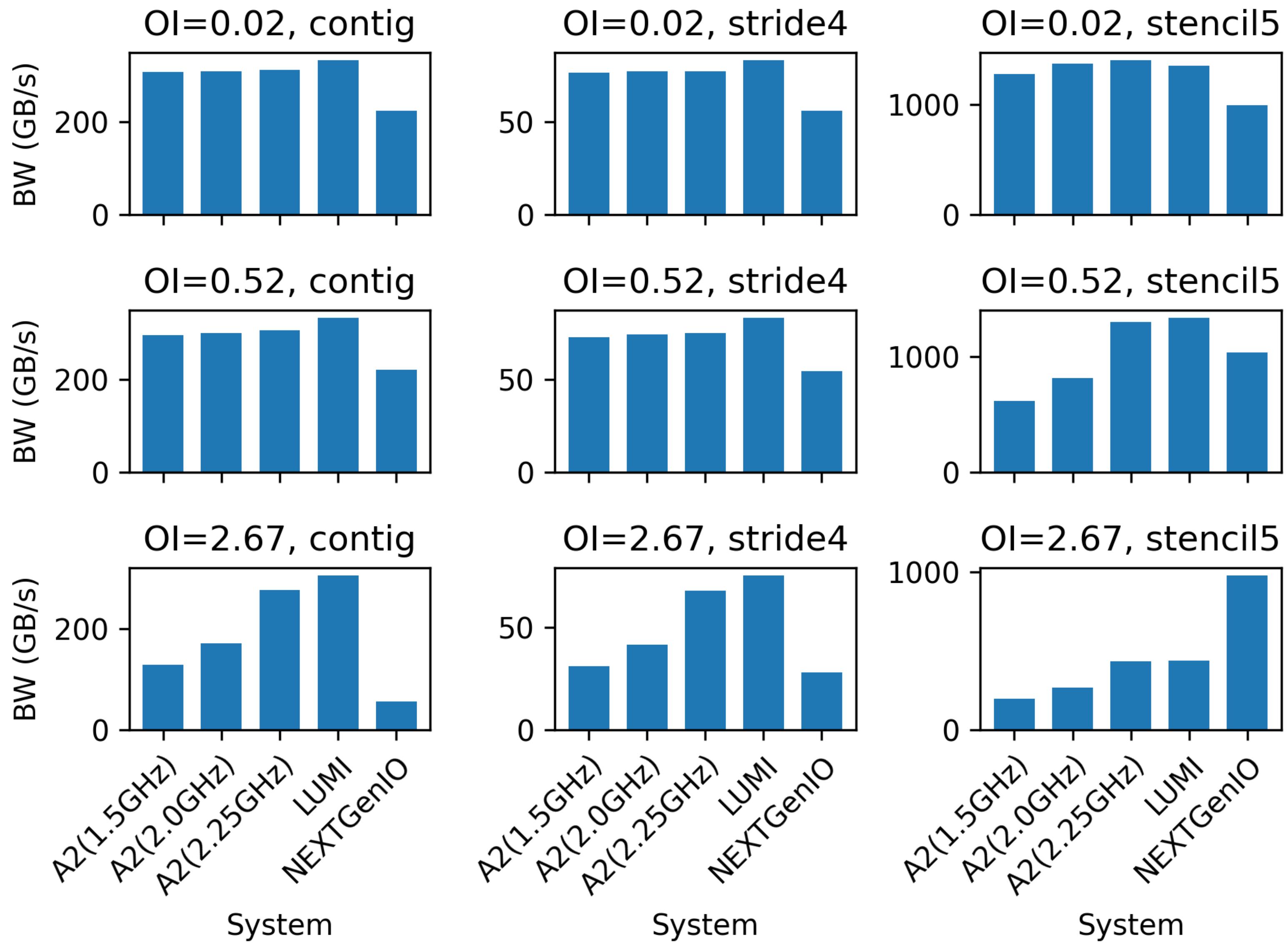
Acknowlegments

This work used the ARCHER2 UK National Supercomputing Service (https://www.archer2.ac.uk), the NEXTGenIO system (http://www.nextgenio.eu) and LUMI (https://www.lumi-supercomputer.eu).

The NEXTGenIO system was funded by the European Union's Horizon 2020 Research and Innovation program under Grant Agreement no. 671951, and supported by EPCC, The University of Edinburgh.

We acknowledge the EuroHPC Joint Undertaking for awarding this project access to the EuroHPC supercomputer LUMI, hosted by CSC (Finland) and the LUMI consortium through a EuroHPC Regular Access call.

Some absolute numbers


OI=0.02, contig
OI=0.02, stride4
OI=0.02, stencil5
OI=0.52, contig
OI=0.52, stride4
OI=0.52, stencil5
OI=2.67, contig
OI=2.67, stride4
OI=2.67, stencil5
BW (GB/s)
A2(1.5GHz)
A2(2.0GHz)
A2(2.25GHz)
LUMI
NEXTGenIO
System