# Post-Moore's Law Fusion

## High-Bandwidth Memory, Accelerators, and Native Half-Precision Processing for CPU-Local Analytics

**Viktor Sanca**, Anastasia Ailamaki

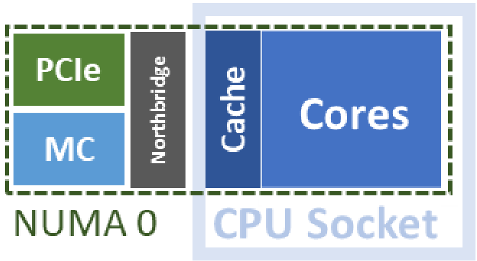
ADMS 2023

viktor-sanca
viktor.sanca@epfl.ch

# CPU Evolution: The Day of Reckoning

[Dennard Scaling]

Multi-Core

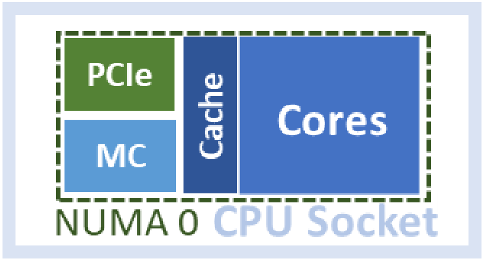


Miniaturization

Low-effort benefits

2008 →

Many-Core



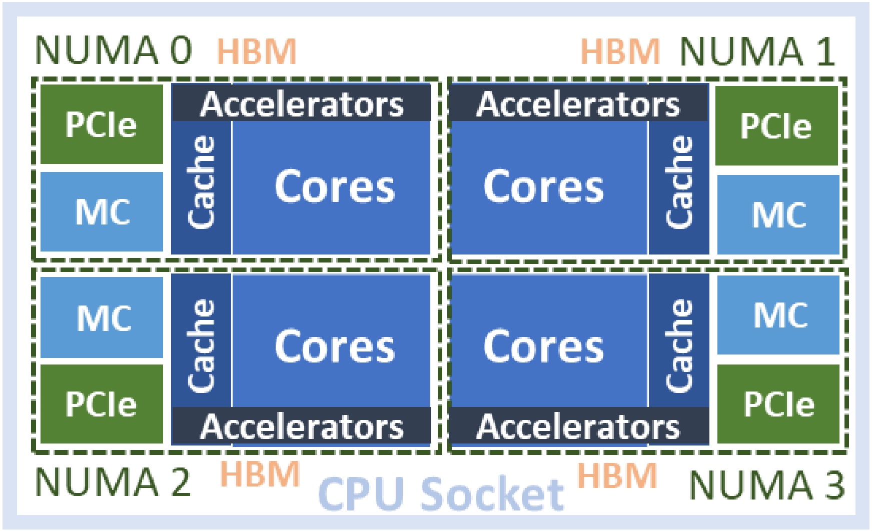
Generalization

Parallelization + vectorization

2015 →

[Moore's Law]

Chiplets

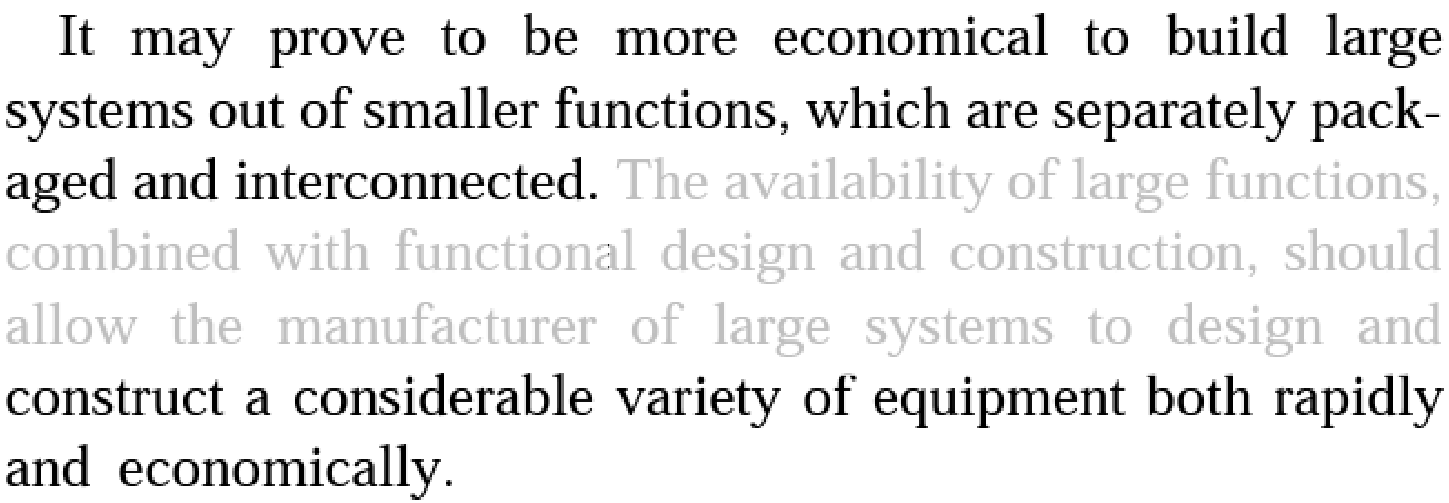


Specialization [Intel, AMD, IBM, Apple, …]

Heterogeneous + specialized unit interactions

# CPU Evolution: The Day of Reckoning

G. Moore: Cramming Components Onto Integrated Circuits (1965)

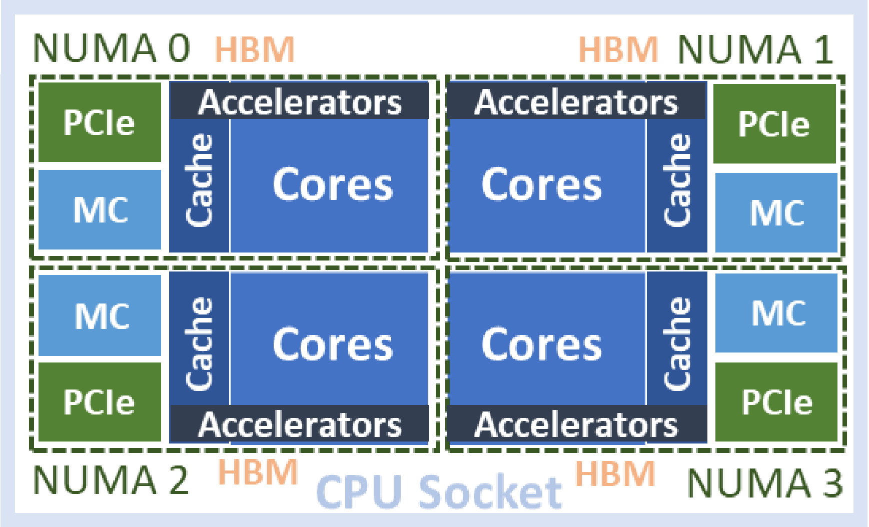


It may prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected. The availability of large functions, combined with functional design and construction, should allow the manufacturer of large systems to design and construct a considerable variety of equipment both rapidly and economically.

*Section VIII: Day of Reckoning*

[Moore's Law]

Chiplets



Specialization [Intel, AMD, IBM, Apple, …]

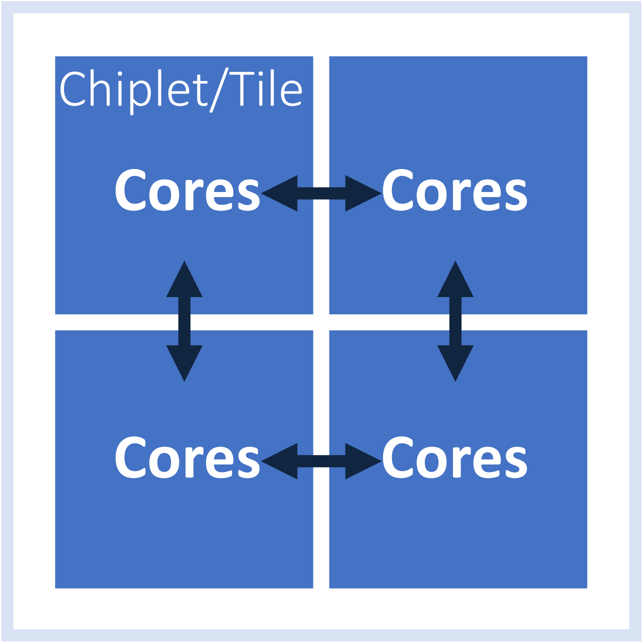
Heterogeneous + specialized unit interactions

Fusion and mix of CPU components: bottleneck shift and novel tradeoffs

# Large System Out of Small Functions

Monolithic CPU

# Large System Out of Small Functions



AMD

Infinity Fabric

Intel

Embedded Multi-die Interconnect Bridge

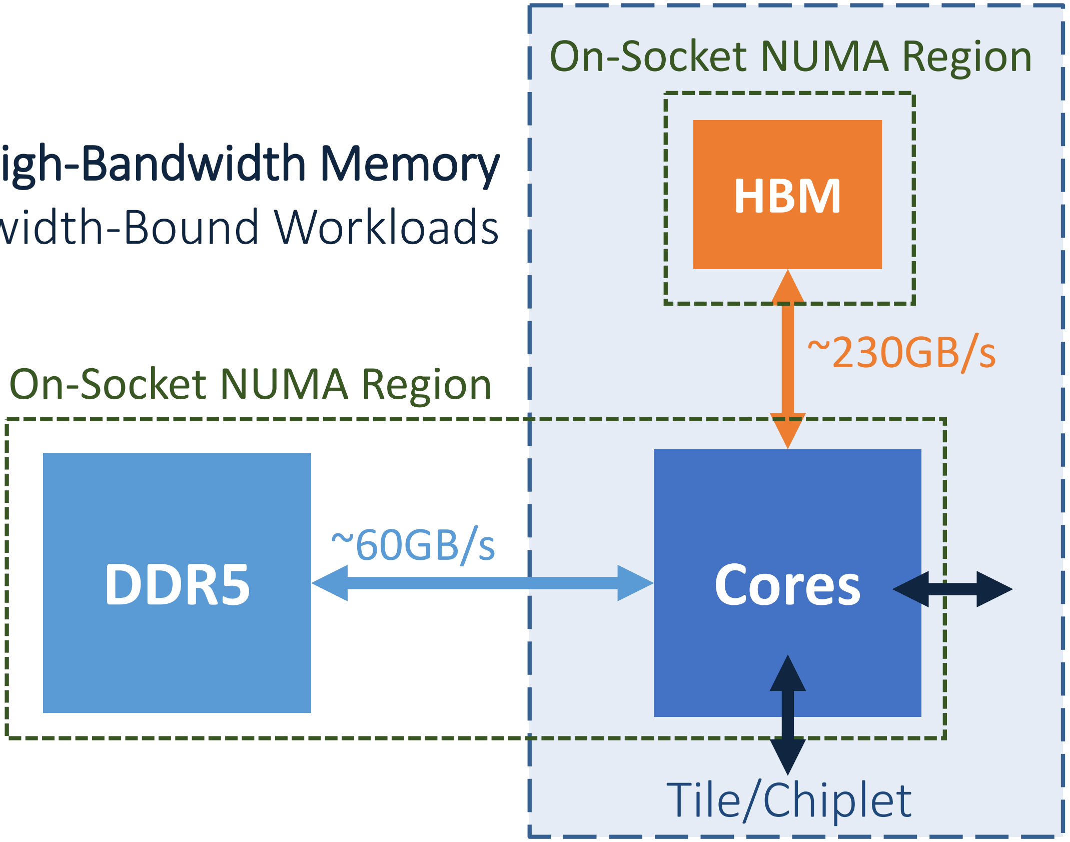
Open Chiplet Ecosystem

Universal Chiplet Interconnect Express (UCIe)

Interconnected Chiplets: Increased On-Socket NUMA Granularity

# A Fusion of Components for Modern Workloads

**1. High-Bandwidth Memory**
Bandwidth-Bound Workloads



**2. Native Half-Precision Types**
Optimized Computation and Vectorization

**3. Specialized Hardware Accelerators**
Offload CPU Cores With Specialized Units

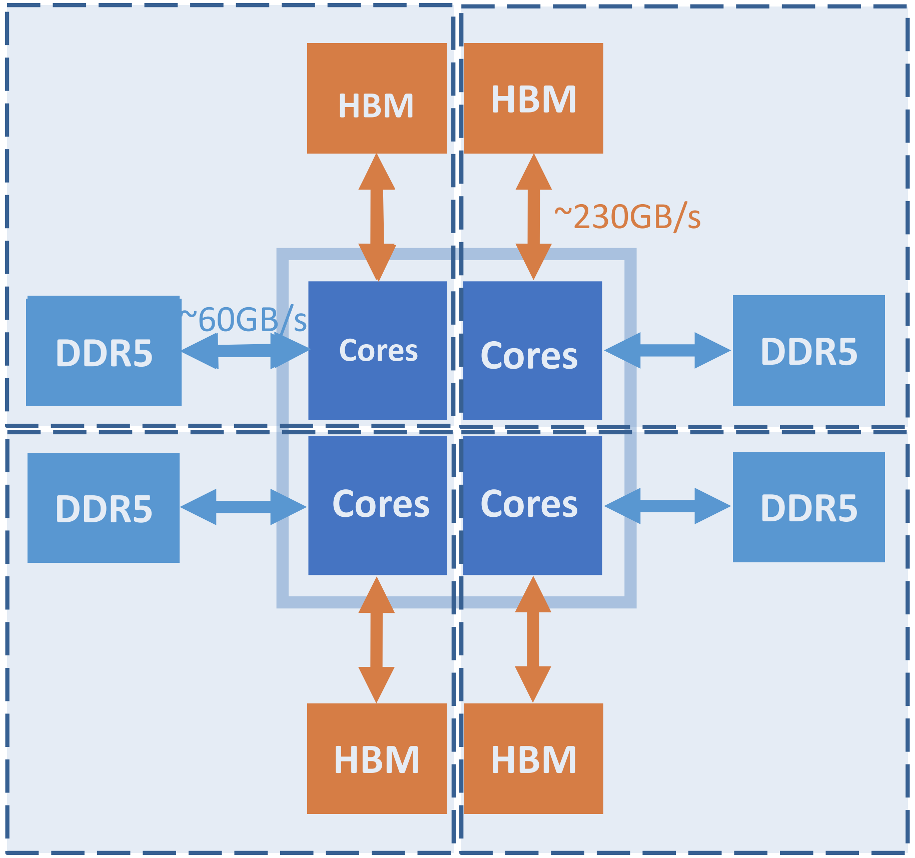
Complex interplay of novel memory + computation non-uniformity

# The Big Picture: Intel Sapphire Rapids

**Intel Xeon 9480 MAX**
4 Chiplets/Tiles Configuration

**Per Tile**
14 cores (28HT), 16GB HBM2e, 64GB DDR5

**Total**
56 cores (112HT), 64GB HBM2e, 256GB DDR5



# The Big Picture: Intel Sapphire Rapids

**Intel Xeon 9480 MAX**

4 Chiplets/Tiles Configuration

**Per Tile**

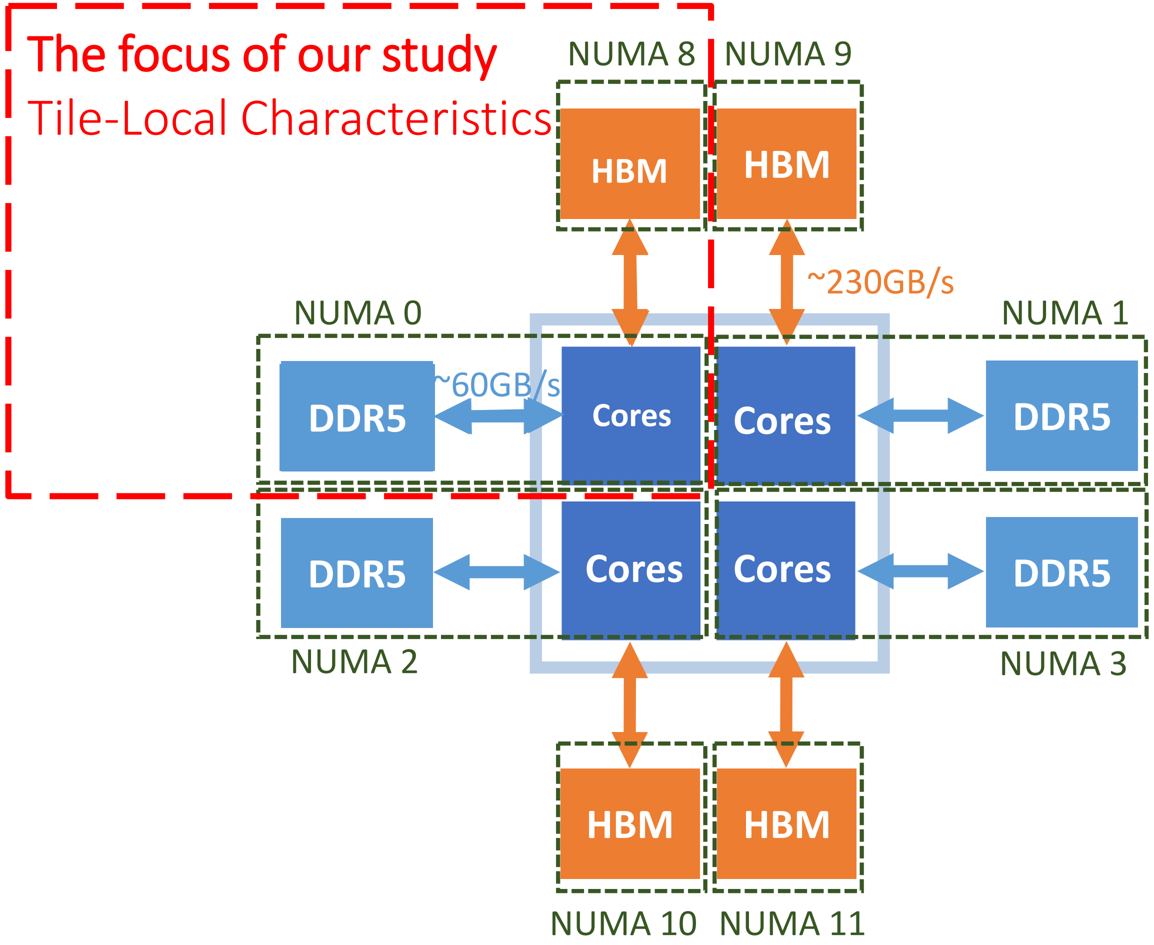
14 cores (28HT), 16GB HBM2e, 64GB DDR5

**Total**

56 cores (112HT), 64GB HBM2e, 256GB DDR5

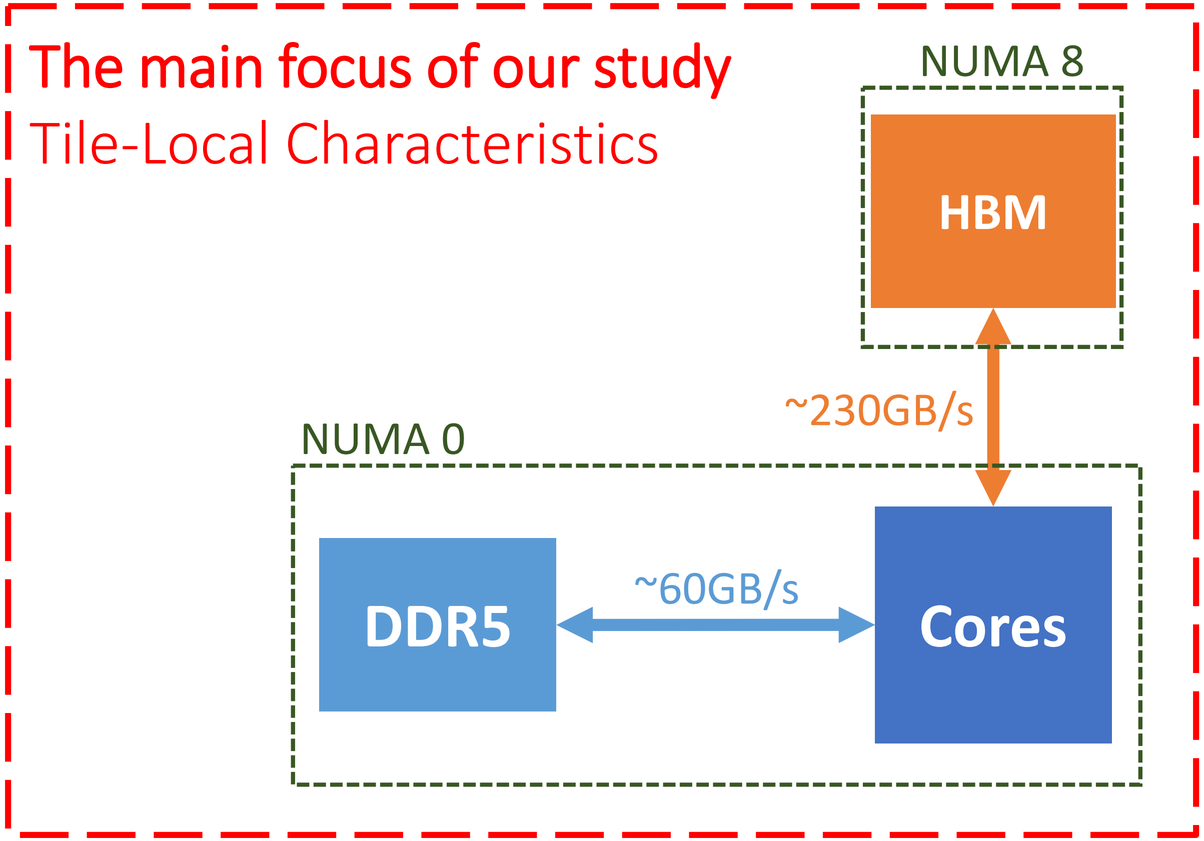
**NUMA**

8 regions per socket: 4 HBM + 4 CPU/DRAM



Evolution: from monolithic CPU resources to fine-grained control

# Interaction of Individual Functionalities



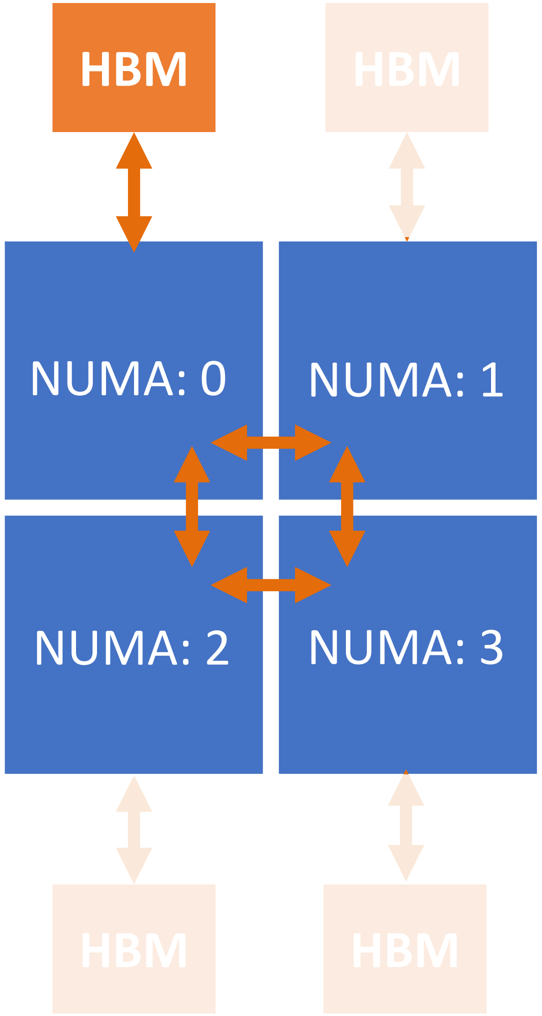
1. **High-Bandwidth Memory**
Bandwidth-Bound Workloads

2. Native Half-Precision Types
Optimized Computation + Vectorization

3. Specialized Hardware Accelerators
Offload CPU Cores By Specialized Units

Evaluate the interplay of granular memory and computational decisions

# HBM vs. DRAM: Extending the Memory Hierarchy



Intel Memory Latency Checker (MLC v3.10) Bandwidth Matrix

HBM Bandwidth

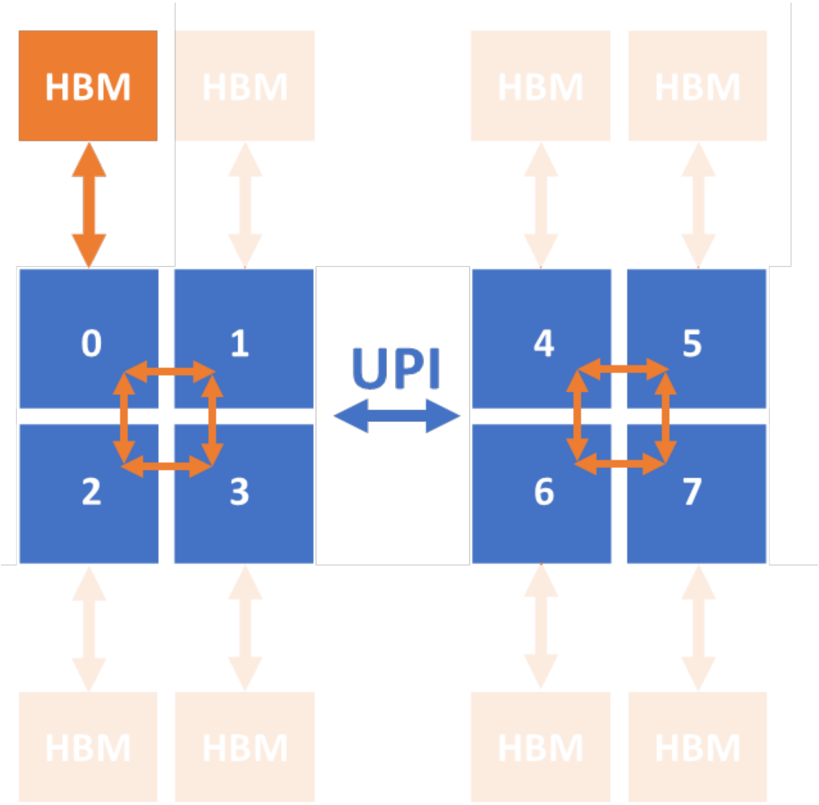
| NUMA | GB/s |
|---|---|
| 0 | 220.92 |
| 1 | 144.27 |
| 2 | 126.16 |
| 3 | 122.46 |

Increase over DRAM

| NUMA | % |
|---|---|
| 0 | 367.95 |
| 1 | 238.55 |
| 2 | 209.06 |
| 3 | 203.05 |

2-3.7x bandwidth increase on socket for shifting the DRAM bottleneck

# HBM vs. DRAM + Interconnects: Latency Slowdown



Intel Memory Latency Checker (MLC v3.10) Latency Matrix

HBM Latency

| NUMA | ns |
|---|---|
| 0 | 120.4 |
| 1 | 127.1 |
| 2 | 133.3 |
| 3 | 142.6 |
| 4 | 233.4 |
| 5 | 234 |
| 6 | 235.4 |
| 7 | 235.9 |

Slowdown over DRAM

| NUMA | % |
|---|---|
| 0 | 28.22 |
| 1 | 22.33 |
| 2 | 19.02 |
| 3 | 20.24 |
| 4 | 1.88 |
| 5 | 1.61 |
| 6 | 1.16 |
| 7 | 0.81 |

EMIB/Tile Mesh

UPI/Remote

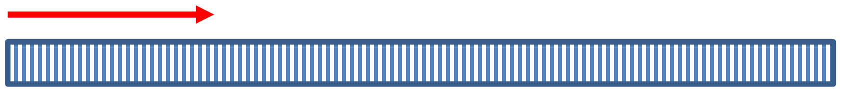
Up to 30% higher latency over DRAM for EMIB, negligible for UPI

# The Impact of HBM: Data Access Patterns

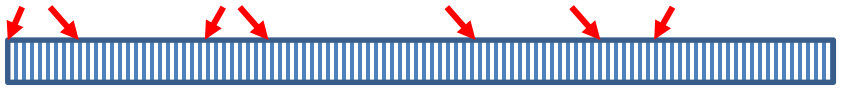
**Workload:** 1B FP32 elements + Tile-local processing (up to 28 threads)

Access patterns

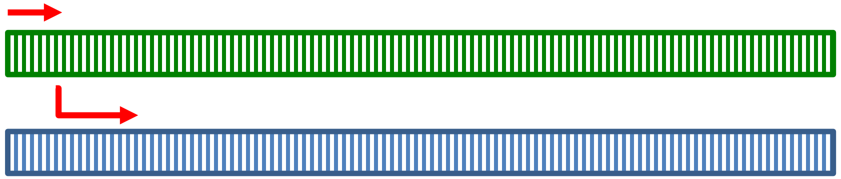
**SCAN:** full sequential scan (bandwidth)



**RANDM:** random access (latency)



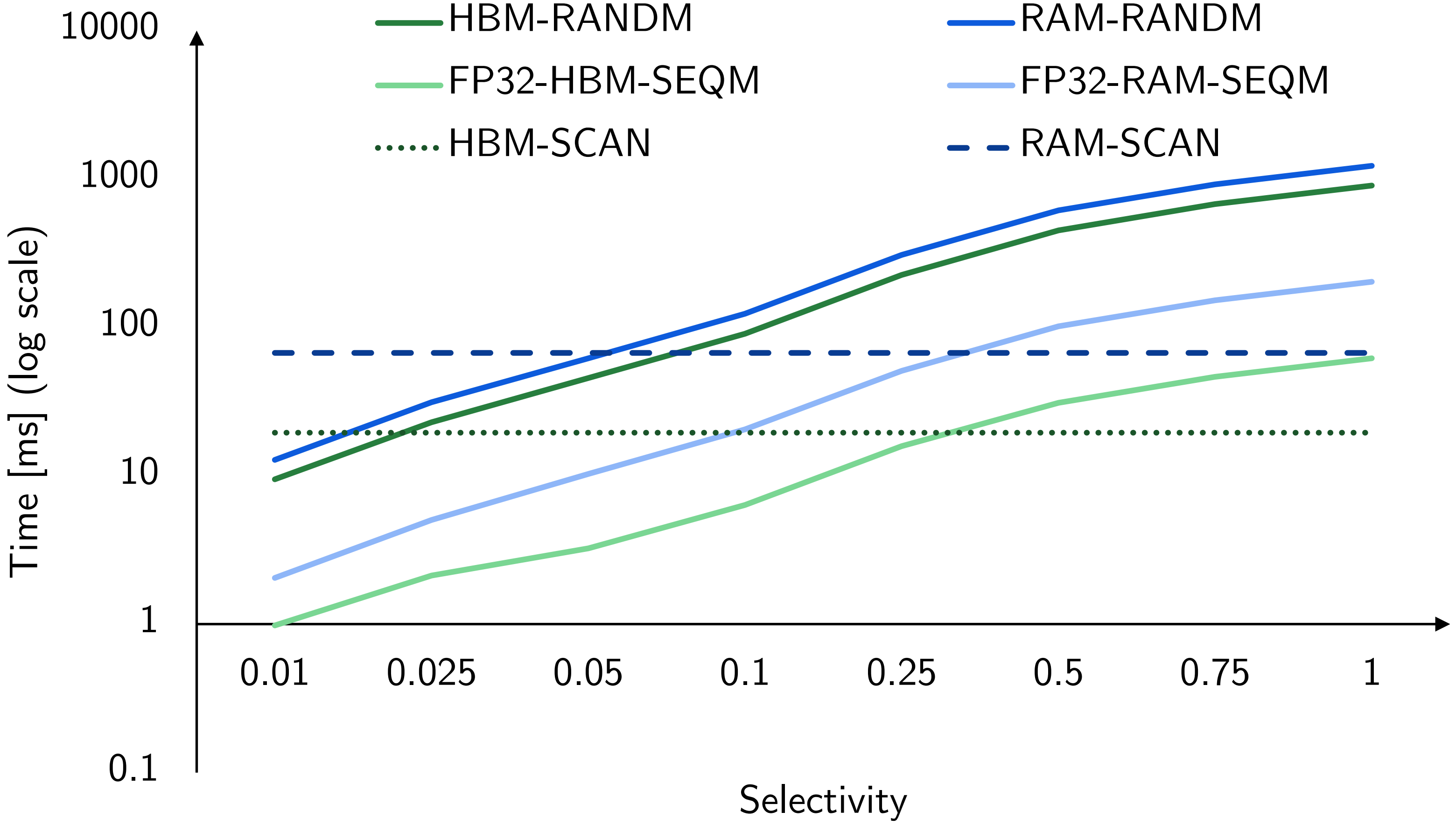
**SEQM:** sequential scan with indirection (mix)



Evaluate extra bandwidth and higher latency on generalized patterns

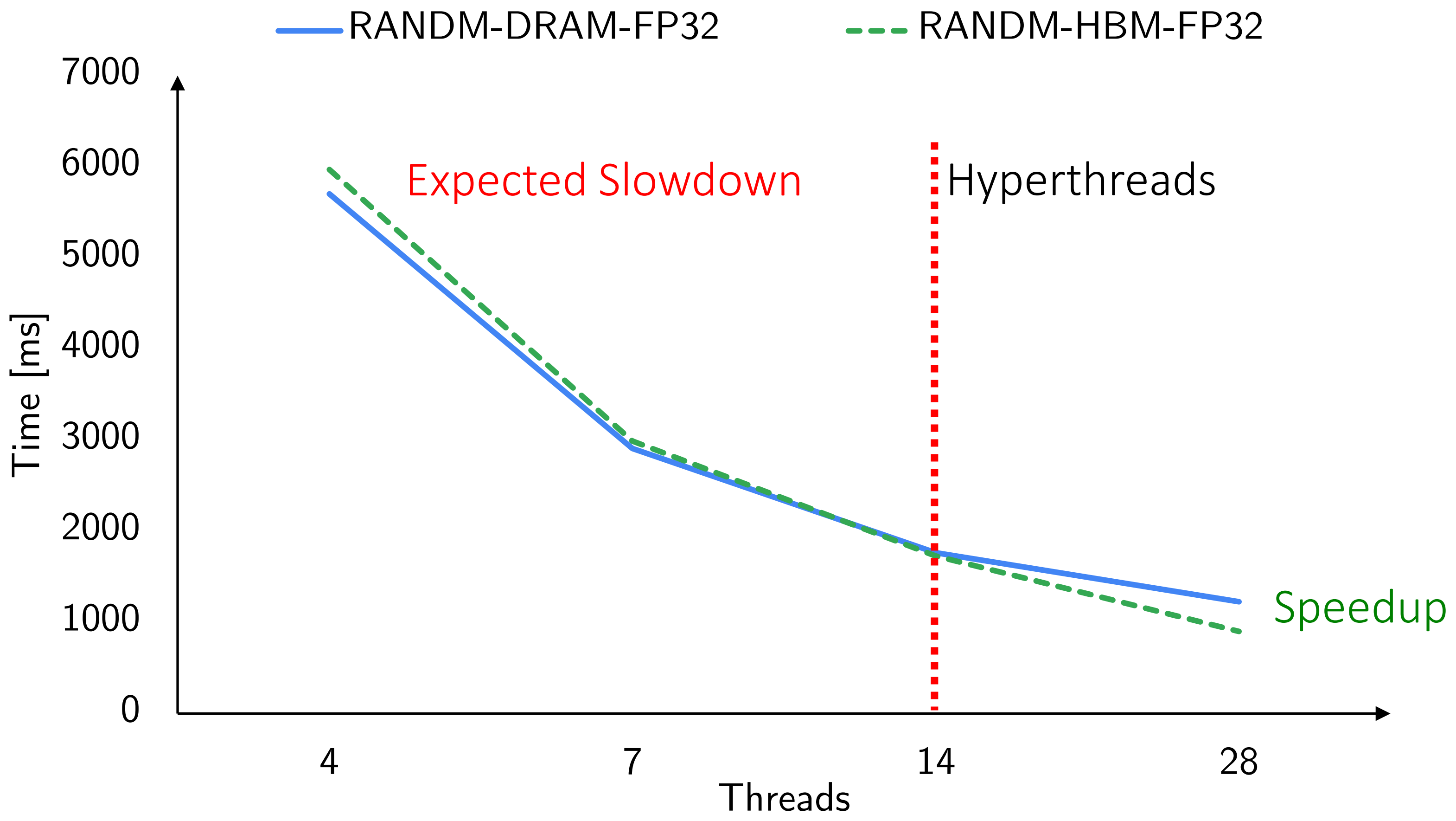
# Data Access Pattern: Bottleneck Shift

SCAN: sequential scan, RANDM: random access, SEQM: sequential scan with indirection; 1B elements, 28 threads



HBM provides additional resources with similar scalability characteristics

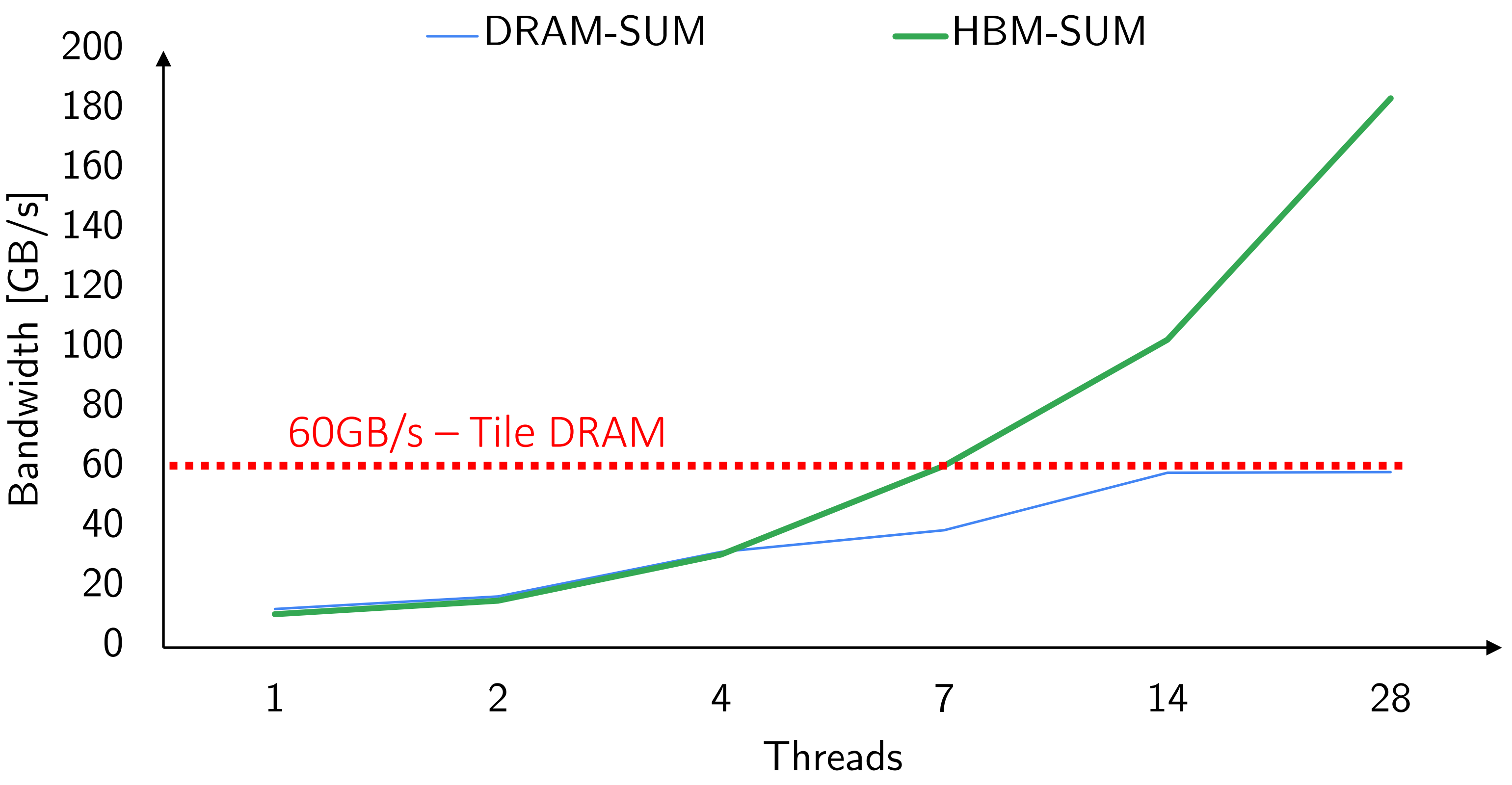
# Higher HBM Latency + Random Access Improvement



HBM scales with additional resources consuming/starving for data

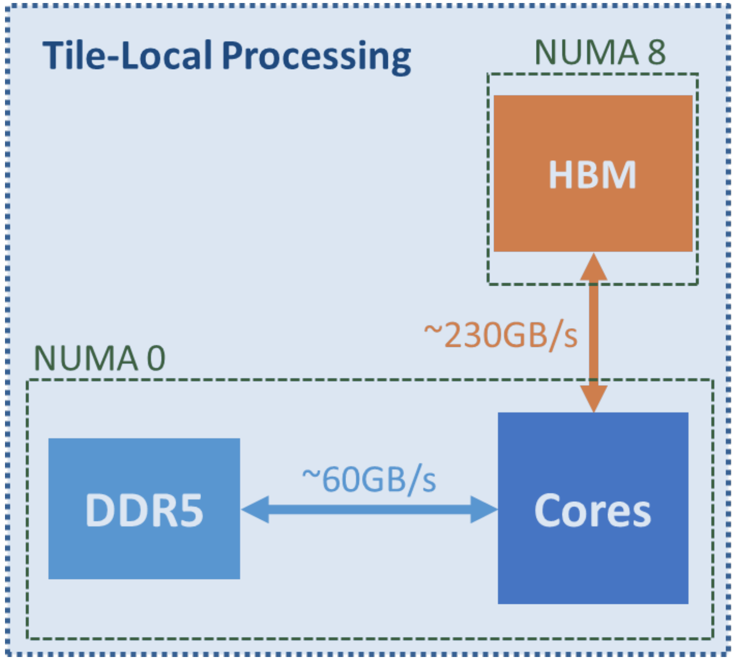
# Scaling The Bandwidth Wall

SUM: summing up 1B elements, up to 28 threads on a single tile, DRAM/HBM local data.



Breaking the DRAM bandwidth wall with the benefit of data + core locality

# Native Half-Precision Types: ML-Driven Opportunity



1. High-Bandwidth Memory
Bandwidth-Bound Workloads and Access Patterns

2. Native Half-Precision Types
Optimized Computation + Vectorization

3. Specialized Hardware Accelerators
Offload CPU Cores By Specialized Units + Accelerate

Hardware-supported types enable fine-grained memory + compute tuning

# Reducing Transfer Size and Computation Footprint

**Workload:** 1B elements SUM-IF, varying the data type and placement in HBM/DRAM

**Double precision:** FP64

64 bits

**Single precision:** FP32

32 bits

**Half precision:** FP16, BF16

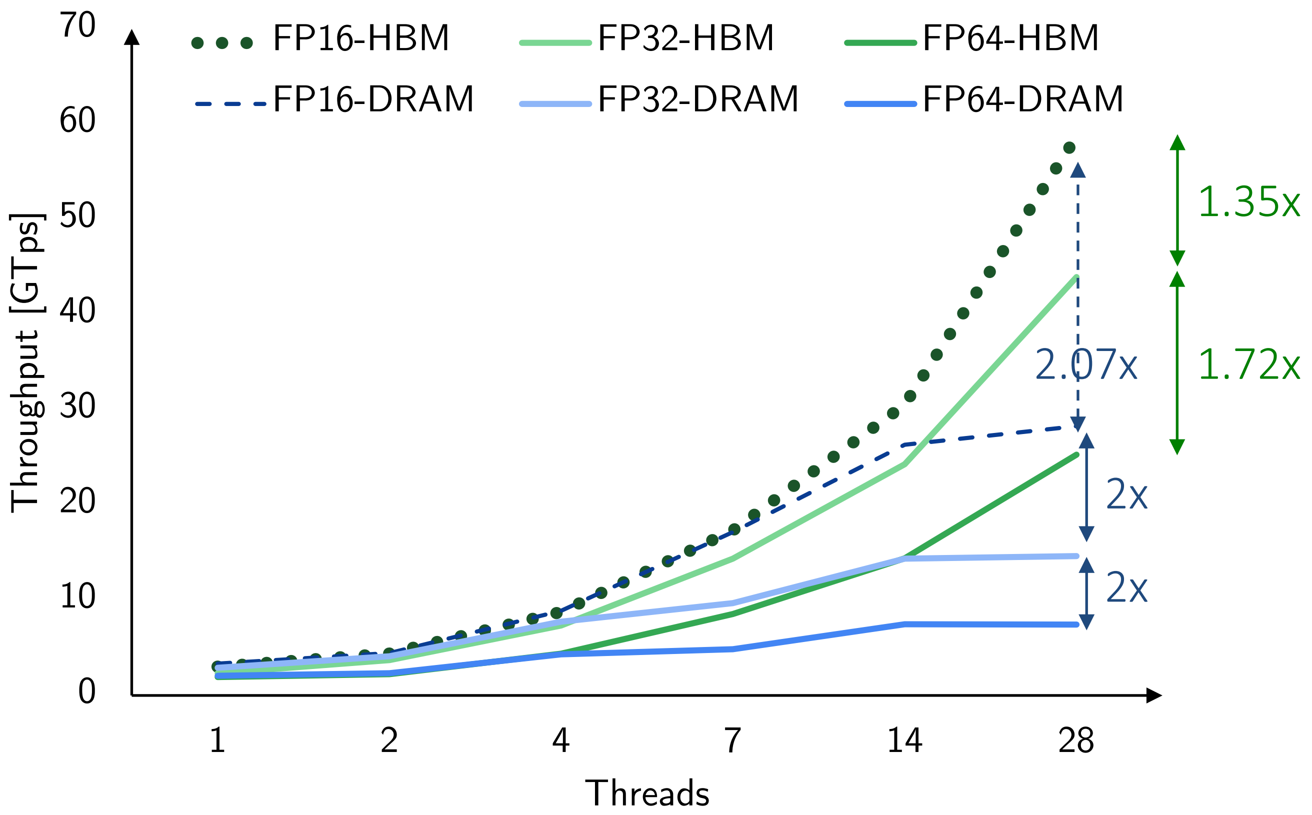
16 bits

**Flexible data types tailored to the workload**

Trade off range + precision for performance

**Hardware Instructions + Vectorization**

Intrinsics and compiler support

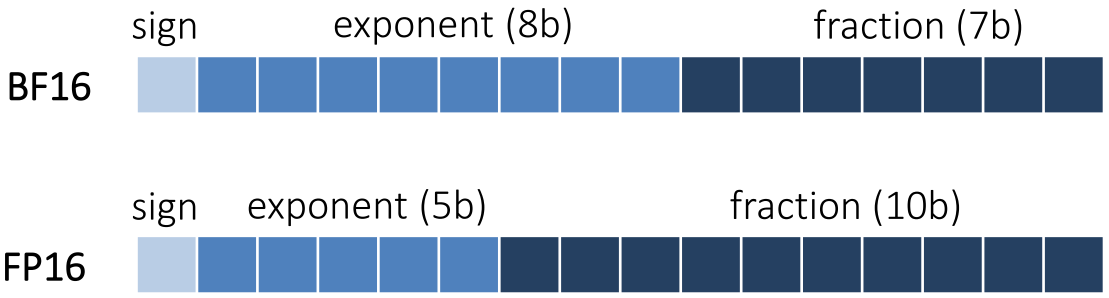


HBM + Types: benefit depends on the shifted memory + compute bottleneck

# On-The-Fly Intermediate Type Conversion

**Workload:** 1B elements, pair-wise multiply-add, FP32->BF16 and FP16 only, DRAM + HBM

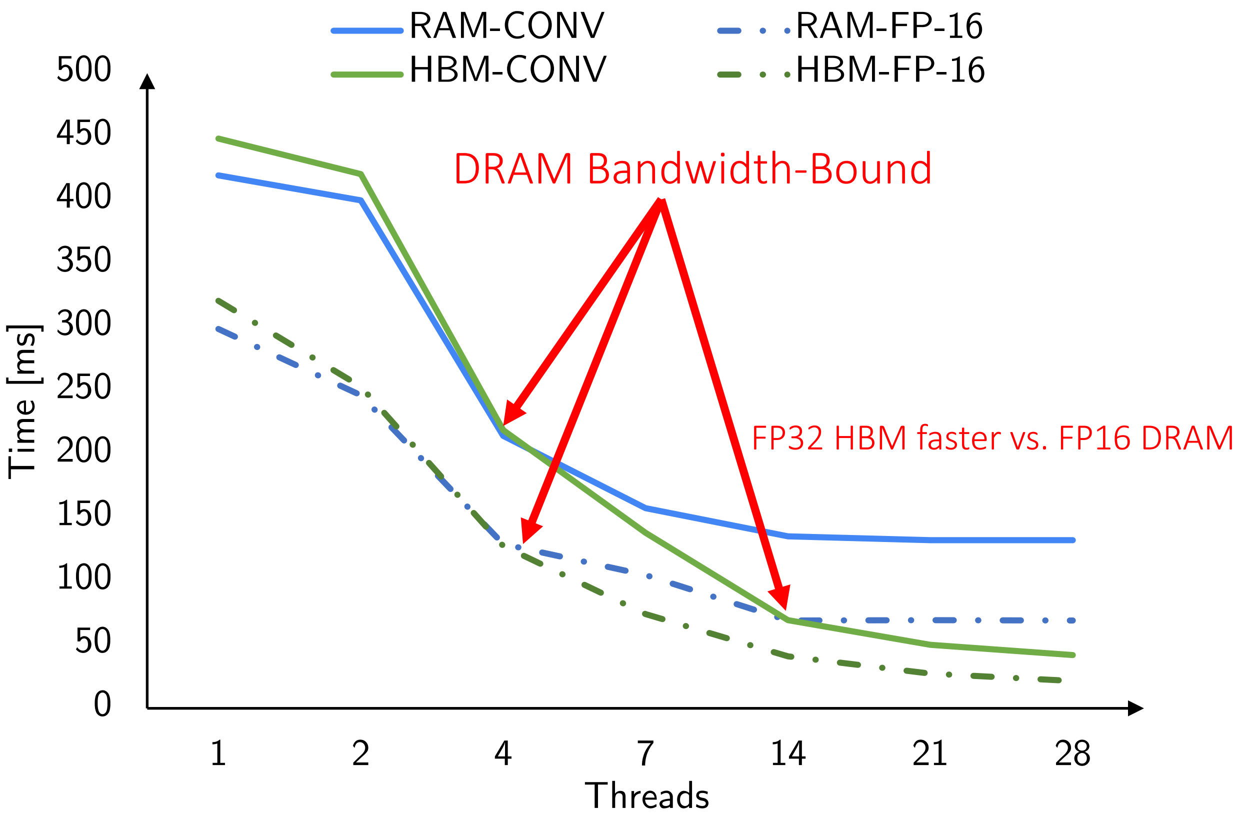
Trade precision for range – ML-driven: no silver bullet!



**BF16:** only intermediate data type for computation

Requires on-the-fly conversion

Optimized computation and intrinsics

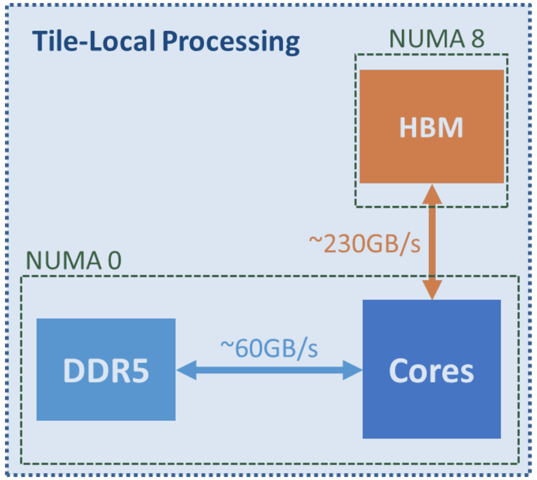


DRAM Bandwidth-Bound

FP32 HBM faster vs. FP16 DRAM

HBM alleviates the data movement bottleneck for efficient computation

# Accelerators: Advanced Matrix Extensions (AMX)

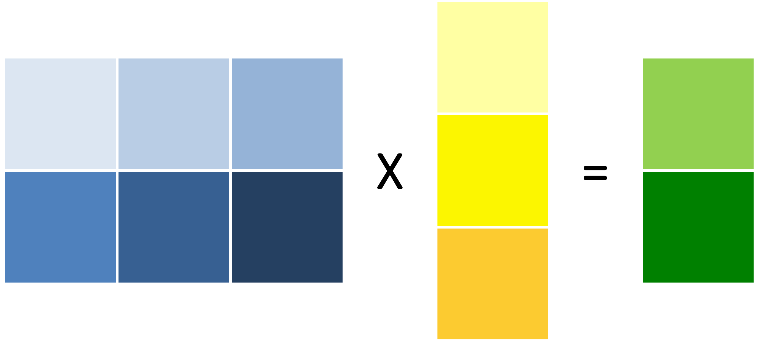


1. High-Bandwidth Memory
Bandwidth-Bound Workloads and Access Patterns

2. Native Half-Precision Types
Optimized Computation + Vectorization

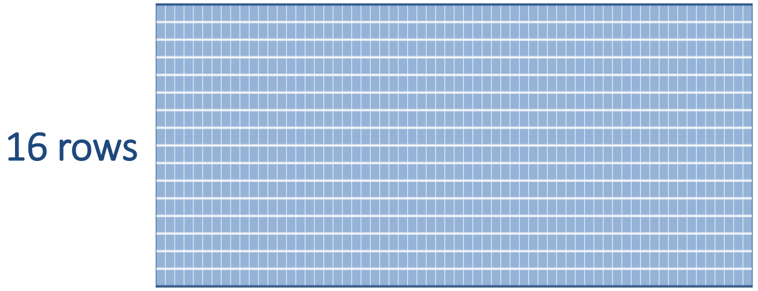
3. Specialized Hardware Accelerators
Offload CPU Cores By Specialized Units + Accelerate Workloads

**Tile Matrix Multiply (TMUL):** Dot Product



ML: Matrix Operations, Convolution, …

64 bytes = 32 x BF16, 64 x INT8



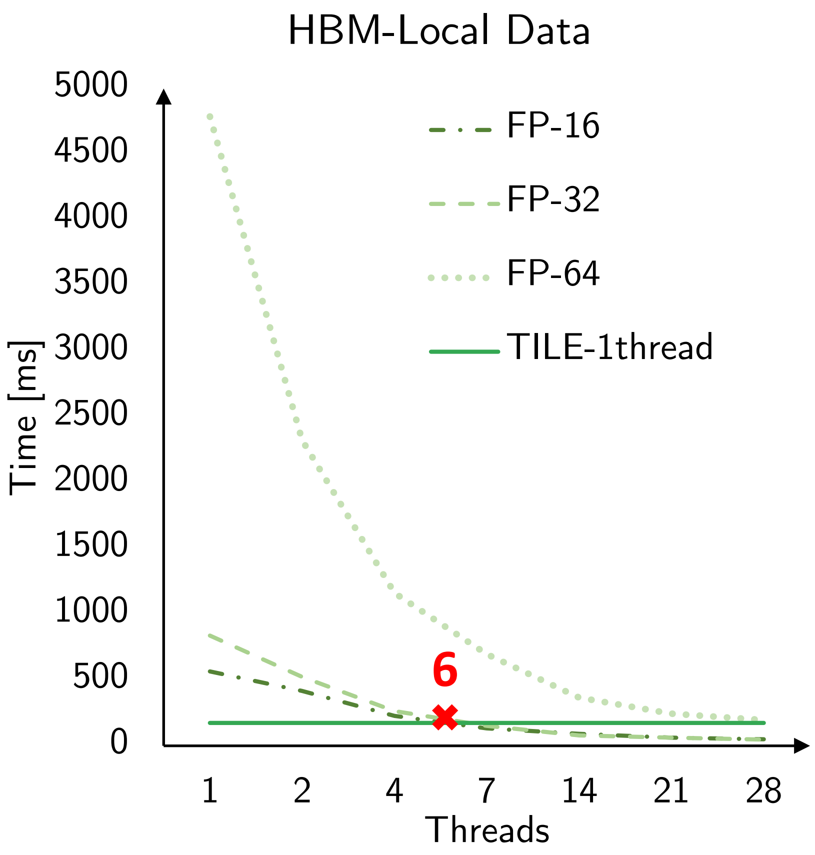
1KByte Tile Register

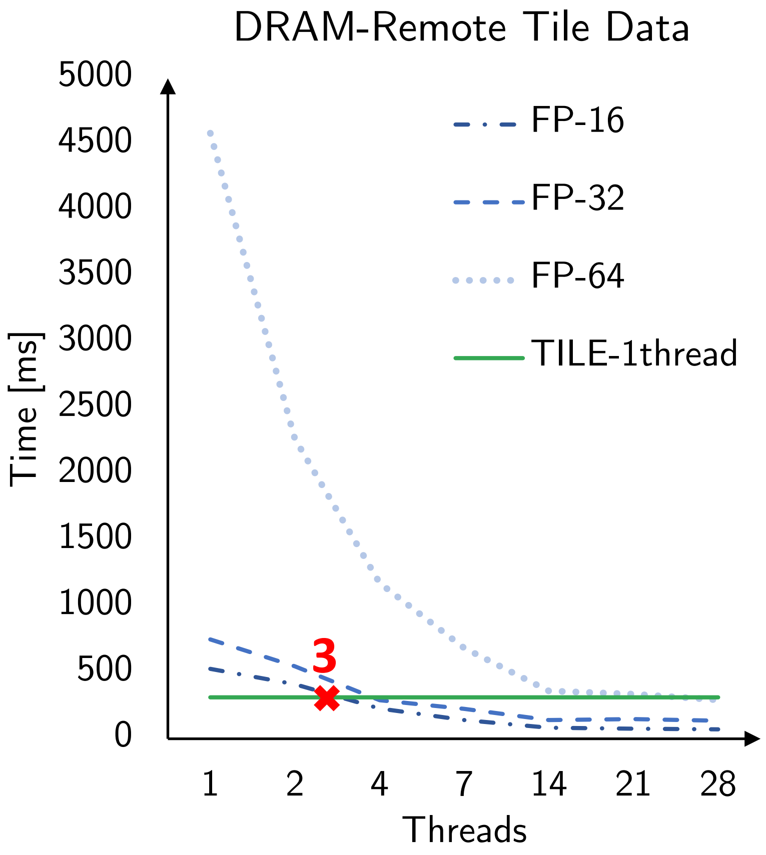
x 8 Register Files + TMUL

Mix-and-match: specialized core-local resources added to design space

# Use Case: Accelerating Vector Computations

**Workload:** 1M tuples x 512-D vector, computing dot products against 512-D vector (on-the-fly BF16 conversion for AMX)





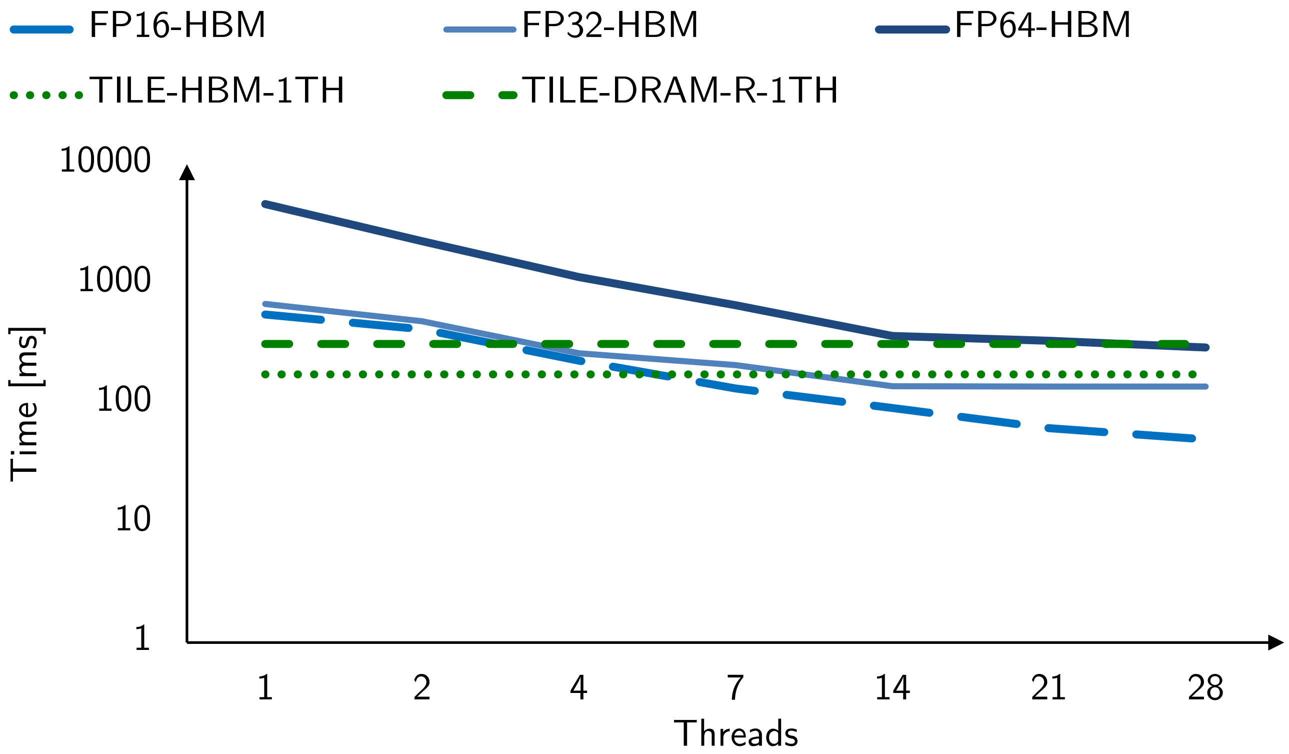
Offload computation from cores: complex decisions inside single socket

# Growing CPU Compute and Storage Heterogeneity

**Workload:** 1M tuples x 512-D vector, computing dot products against 512-D vector (on-the-fly BF16 conversion for AMX)

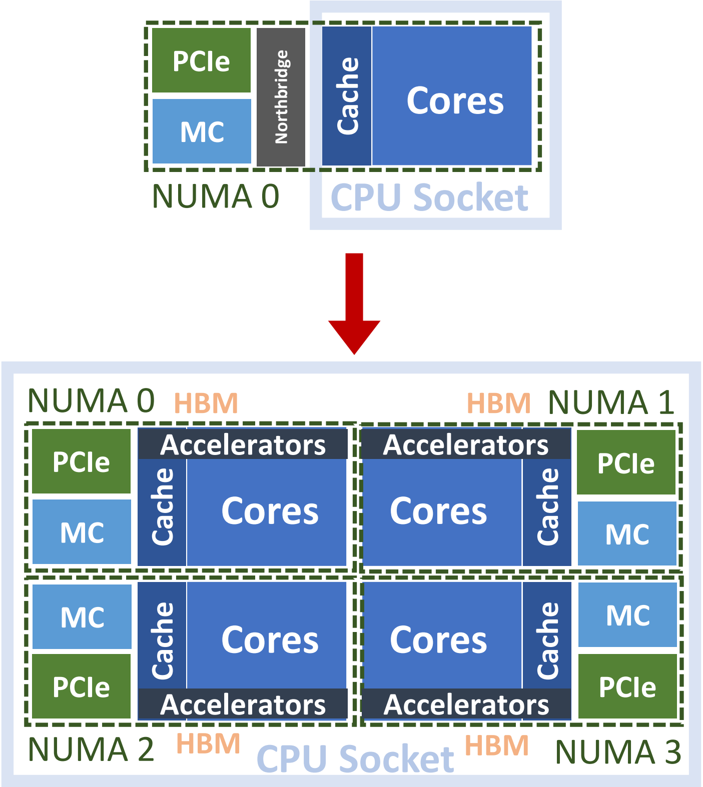
1. High-Bandwidth Memory ⟷ 2. Half-Precision Types ⟷ 3. Specialized Accelerators

Larger Design Space: Interactions + Tradeoffs



Goal: transparent system adaptation to the novel HW interactions

# **Expected** Moore's Law: Large System of Small Functions



**From Monolithic to Complex Heterogeneous CPUs**

On-the-fly system adaptation for any hardware

**Complex Memory and Compute Interactions**

Automating workload benchmarking and tuning

[Chaosity@TPC-TC'23]

**Tailored and Optimized Data Structures and Algorithms**

Using novel hardware fusion with principled design

in viktor-sanca

viktor.sanca@epfl.ch

Build adaptive and hardware-conscious systems for inevitable complexity