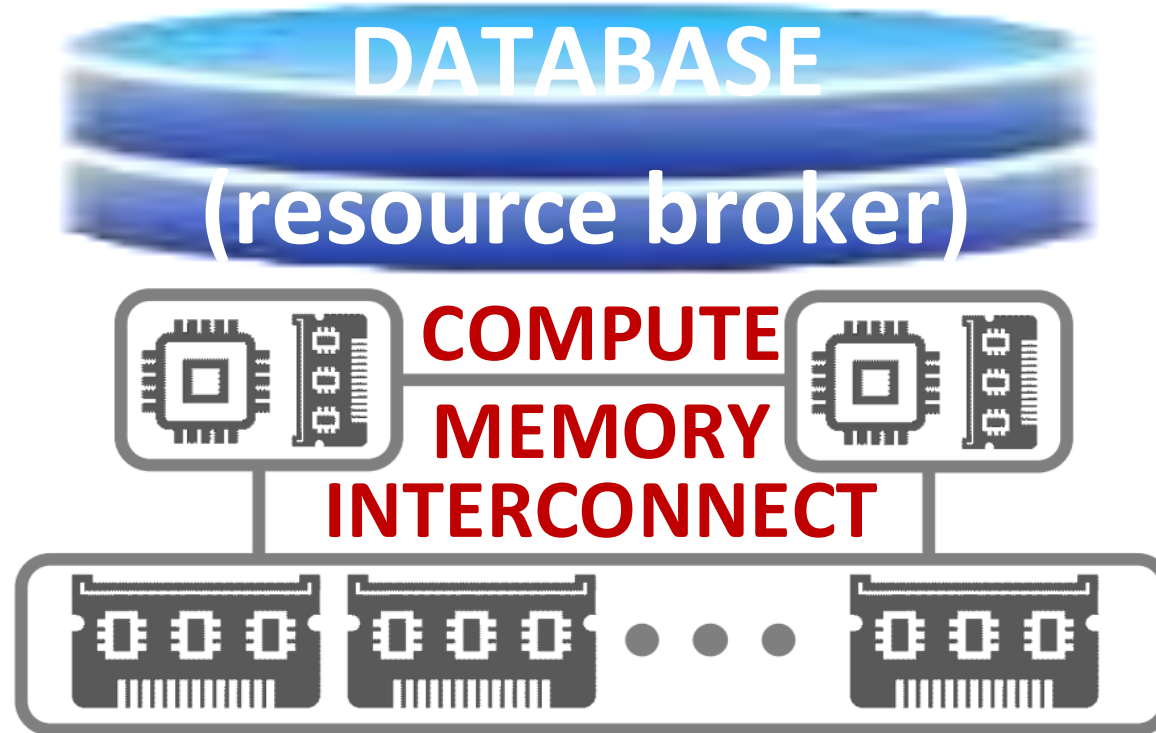
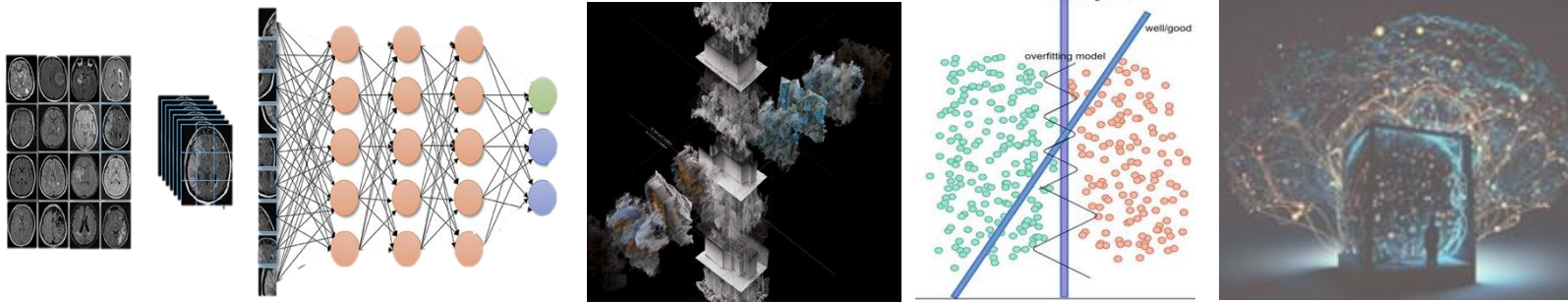


No more promises: ROI-Guided Resource-Centric Systems

Anastasia Ailamaki

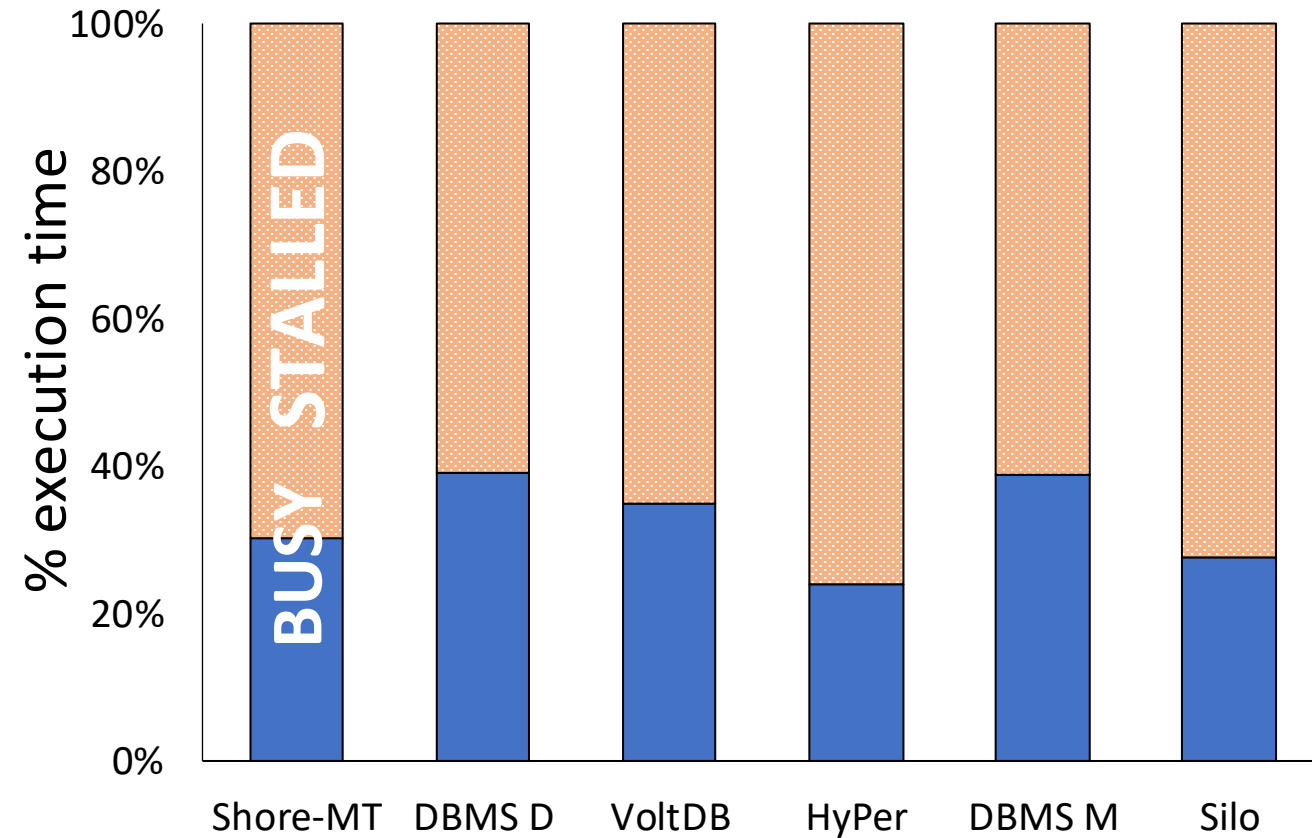
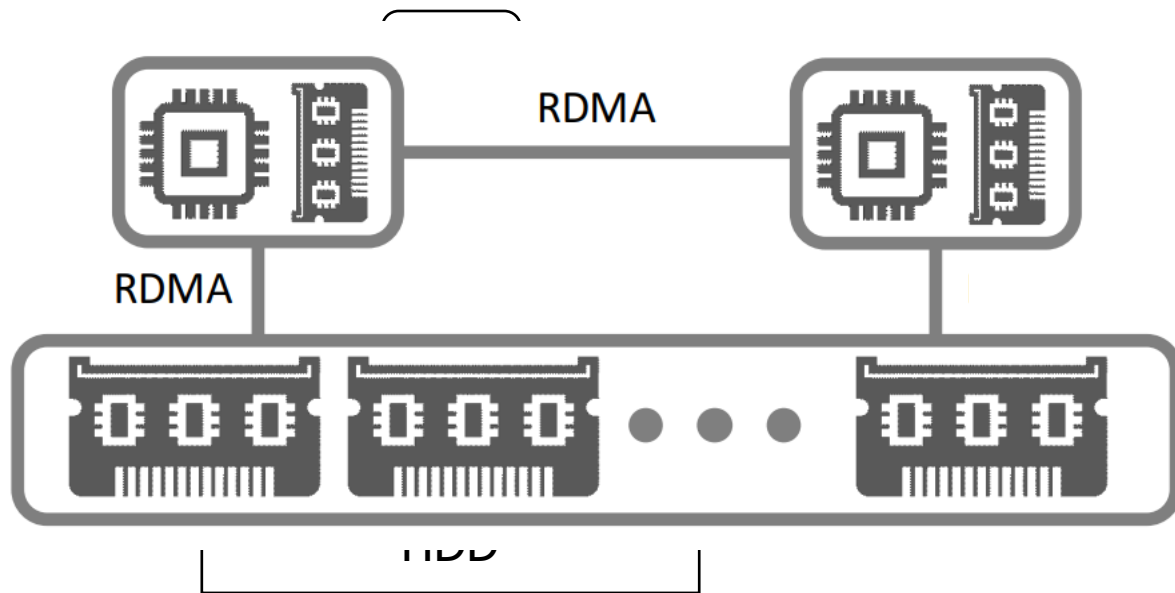
*Data-Intensive Applications and Systems Lab
École Polytechnique Fédérale de Lausanne*

We build compute-centric systems



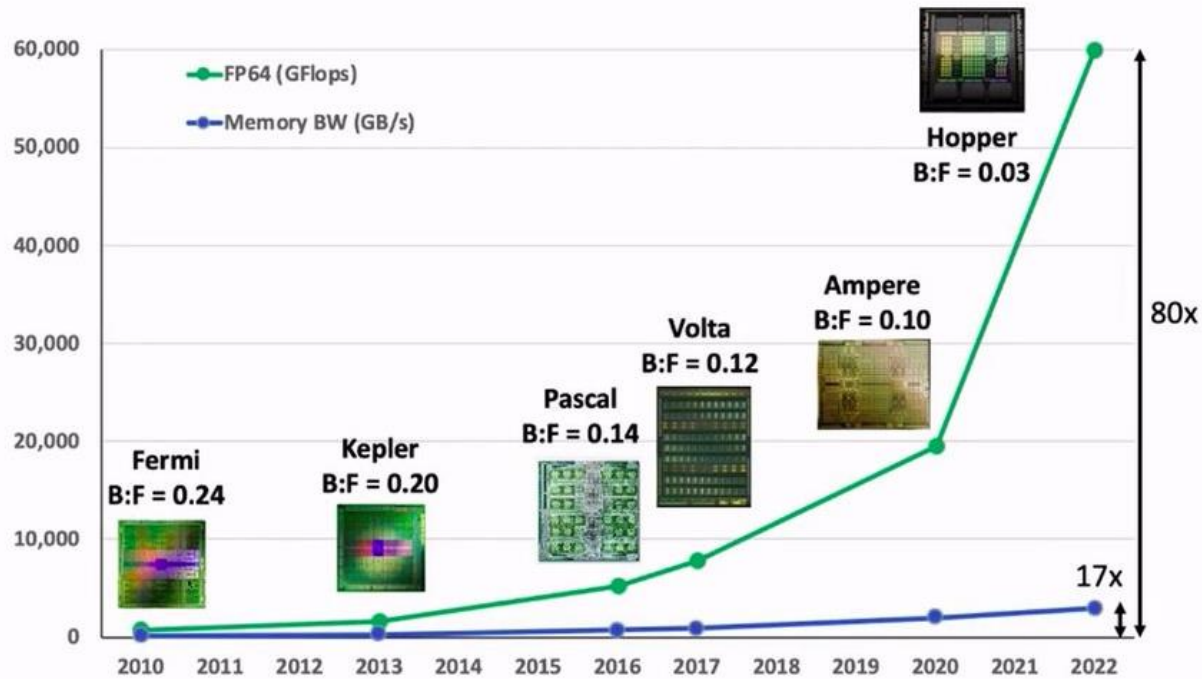
Compute-centric design is wasteful

[Sirin et al, VLDB 2020]



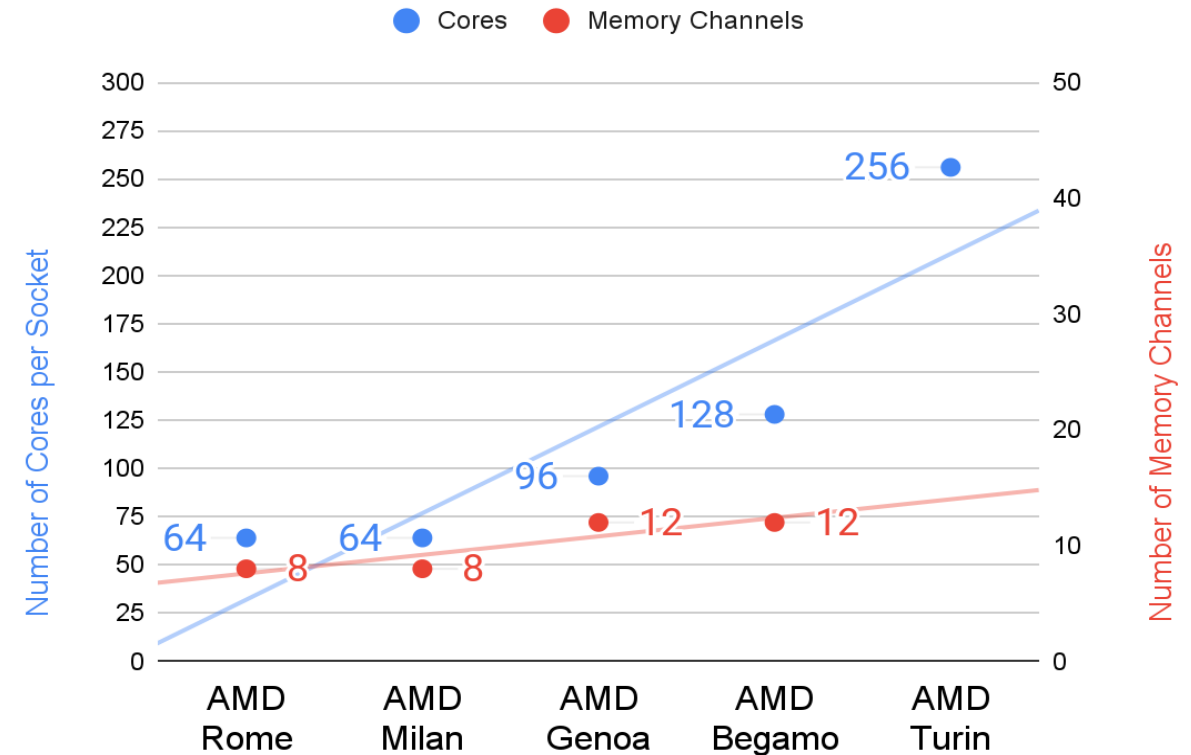
Processors compute faster than memory feeds them

Memory Latency Bandwidth-bound



[Chronis et al, CIDR'25]

AMD Chiplet CPUs: Cores and Memory channels

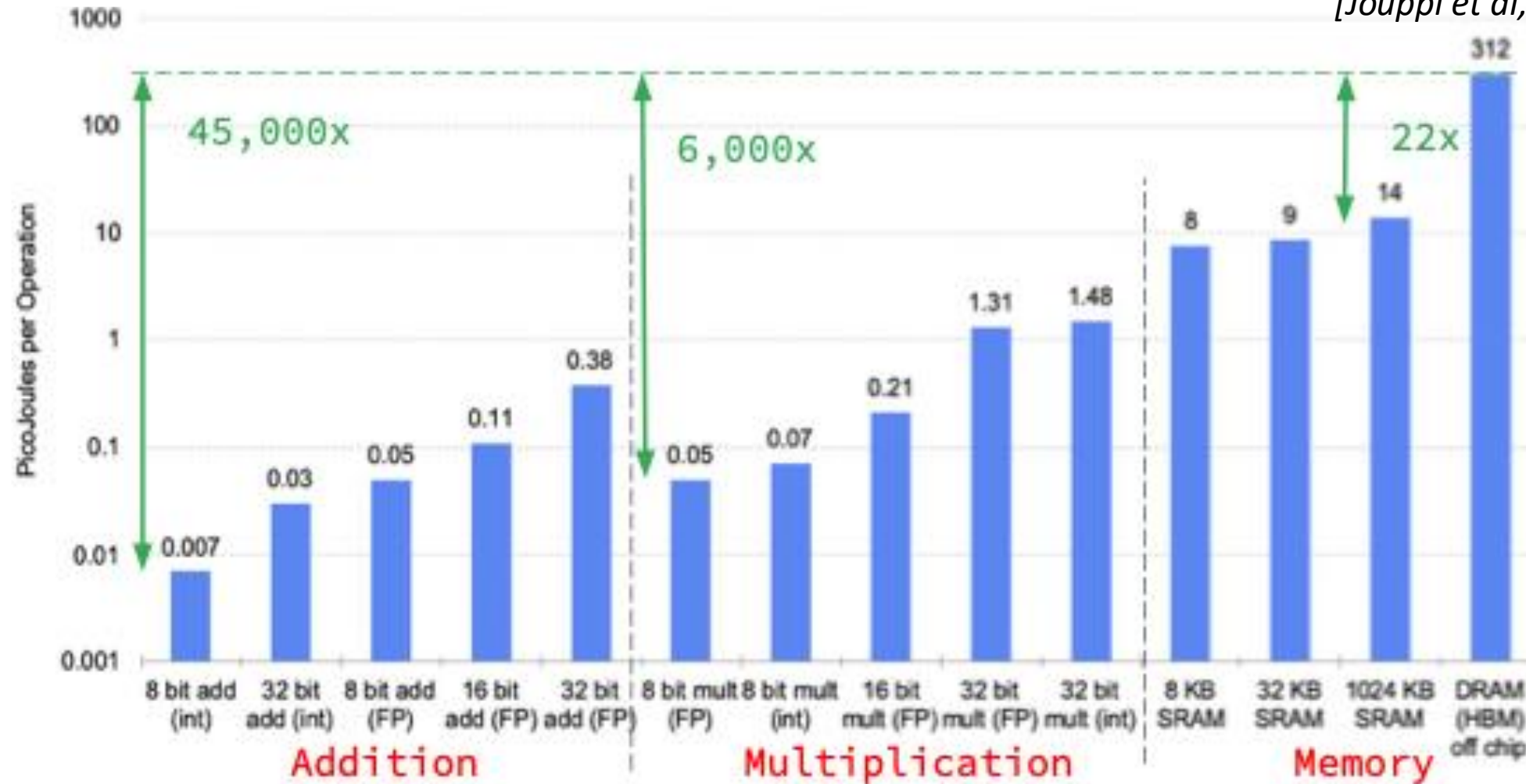


https://twitter.com/david_schor/status/1579456724125089792

Data movement dominates memory efficiency

Memory accesses are energy-hungry

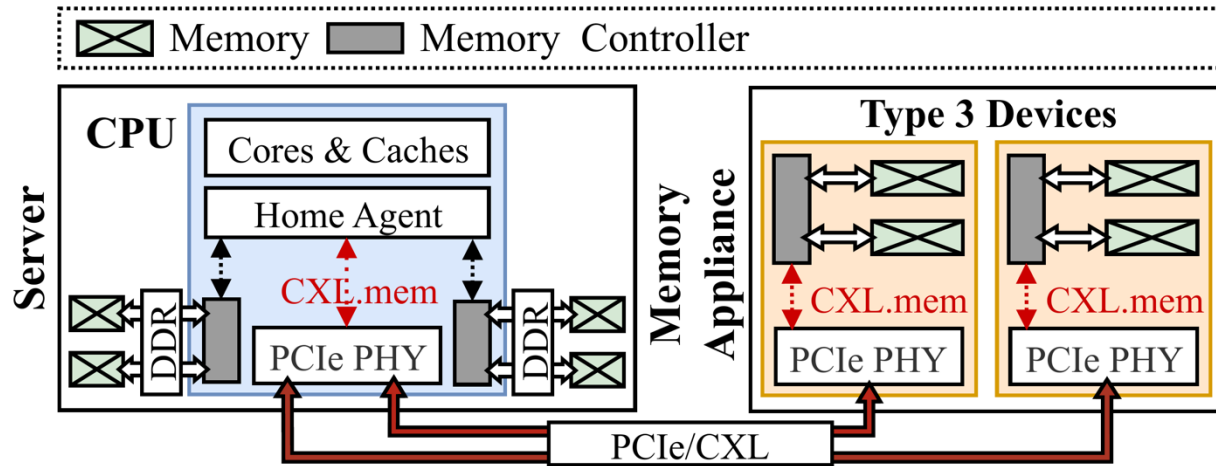
[Jouppi et al, ISCA 2021]



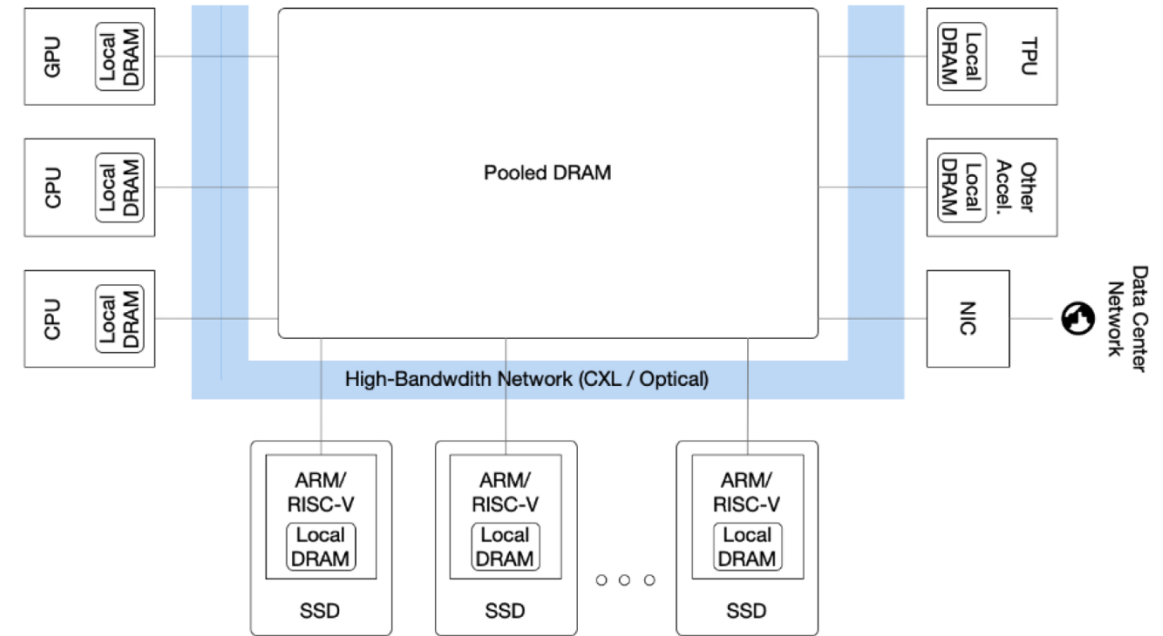
Data movement dominates energy budget

CXL: Pooled Memory over PCIe

[Chronis et al, CIDR'25]



CXL memory expansion [Weisgut et al.]



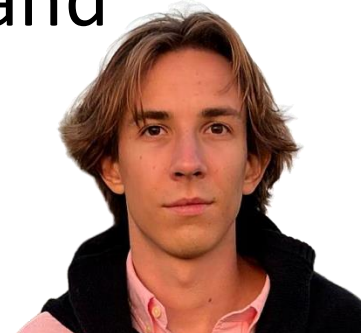
Pooled DRAM [Chronis et al., CIDR'25]

Not just slower DRAM: non-uniform latency and bandwidth

CXL bandwidth expansion: Not as simple as it looks

[Lebedev et al, ADMS'25]

- OS transparently interleaves pages 1:1 across DRAM and CXL
- May *hurt* performance (SSB loses 20% vs DRAM-only)
- Optimal DRAM:CXL interleaving ratio varies by up to 88%
- Workload-aware management reclaims 25% stranded DRAM
 - database must communicate runtime memory demand
 - not just react to page faults



ROI-guided runtime interleaving

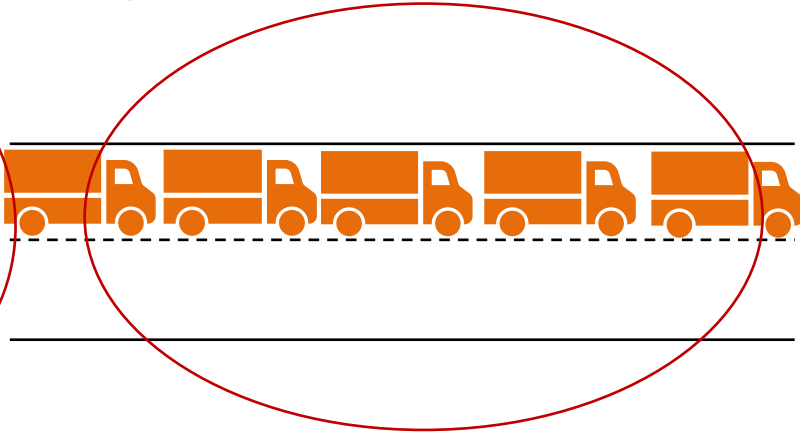
Scan time dominates out-of-memory analytics

[Nicholson et al, ICDE'26]

Persistent Storage

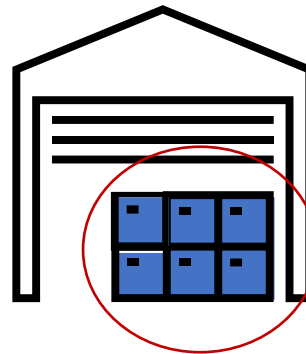


Device BW

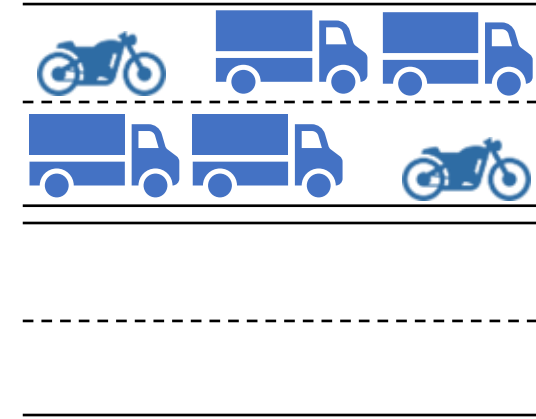


Interconnect BW

Memory



Memory capacity



Compute



Processing throughput

48% of CPU time spent on scans on customer workloads

Szlang VLDB'25, Renen VLDB'23



85% of time spent on scans in Velox-CUDF on TPC-H

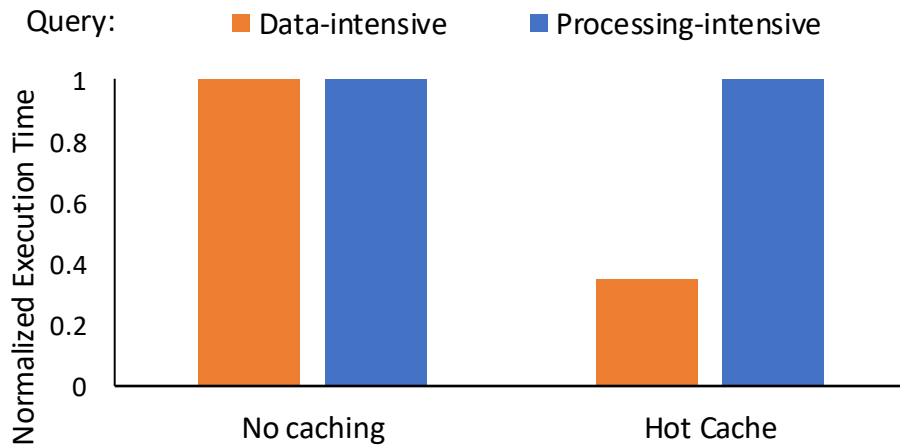
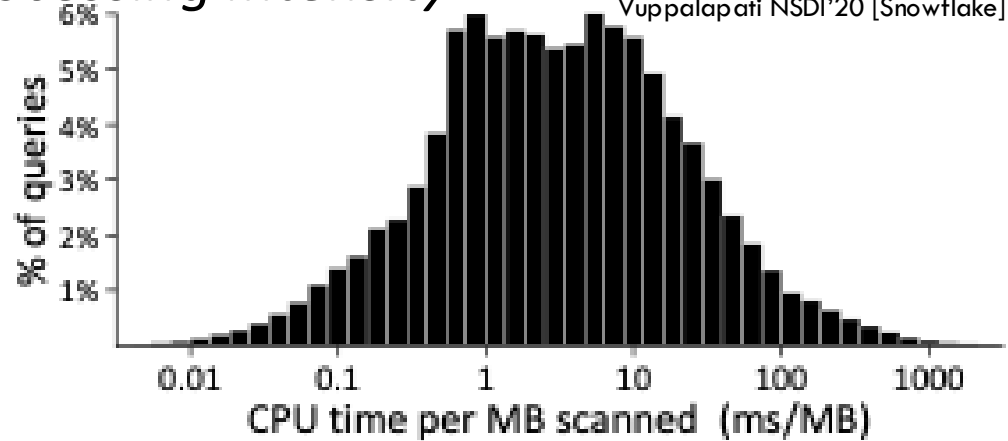
Kimball VeloxCon'25



The scan bottleneck is not fixed

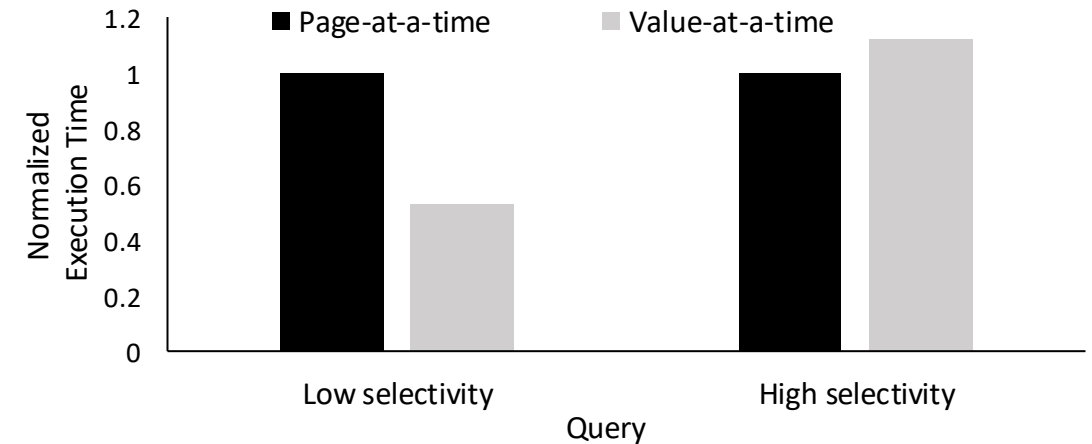
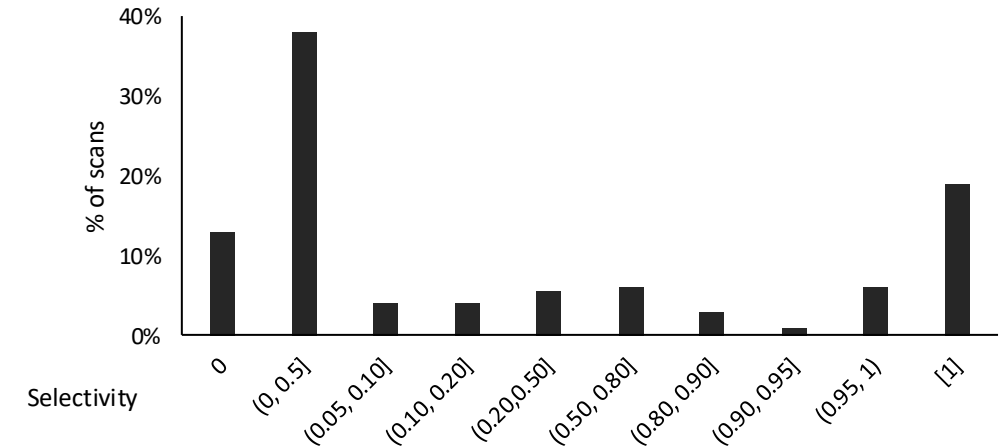
Impossible to make a-priori movement decisions

Processing Intensity



Selectivity

[Nicholson et al, VLDB'24]



Every query property matters

Decide caching based on throughput *potential*

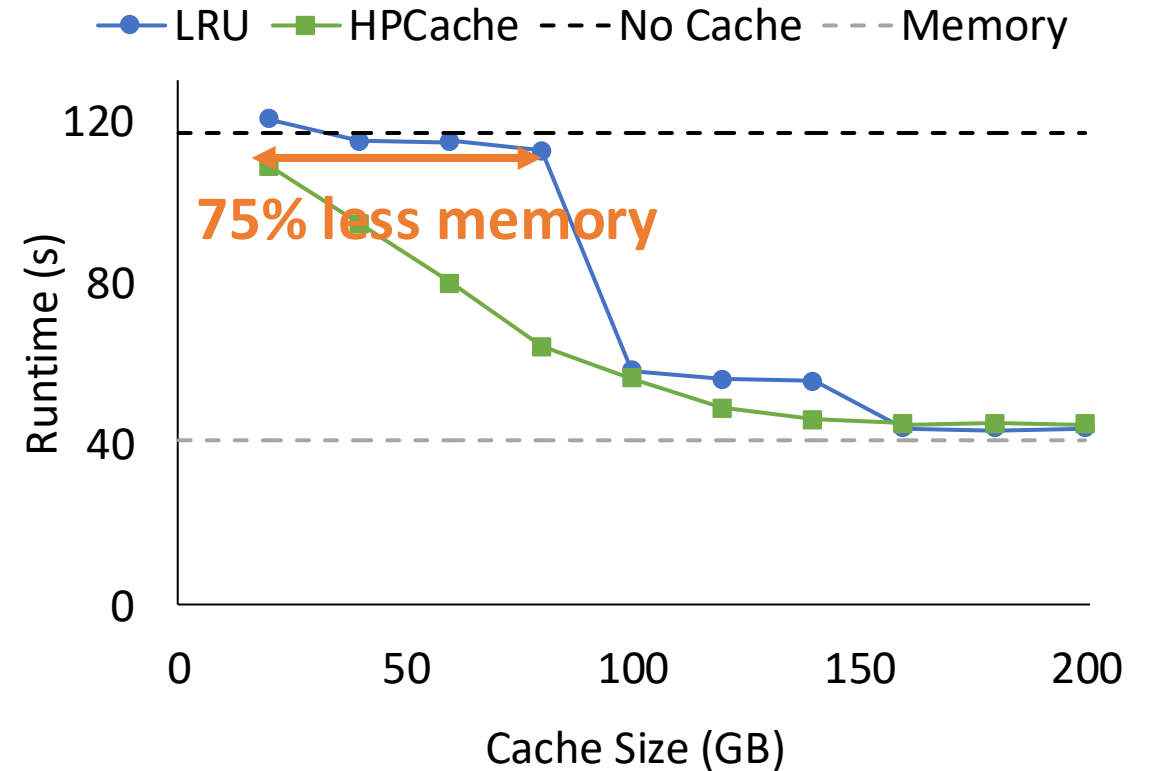
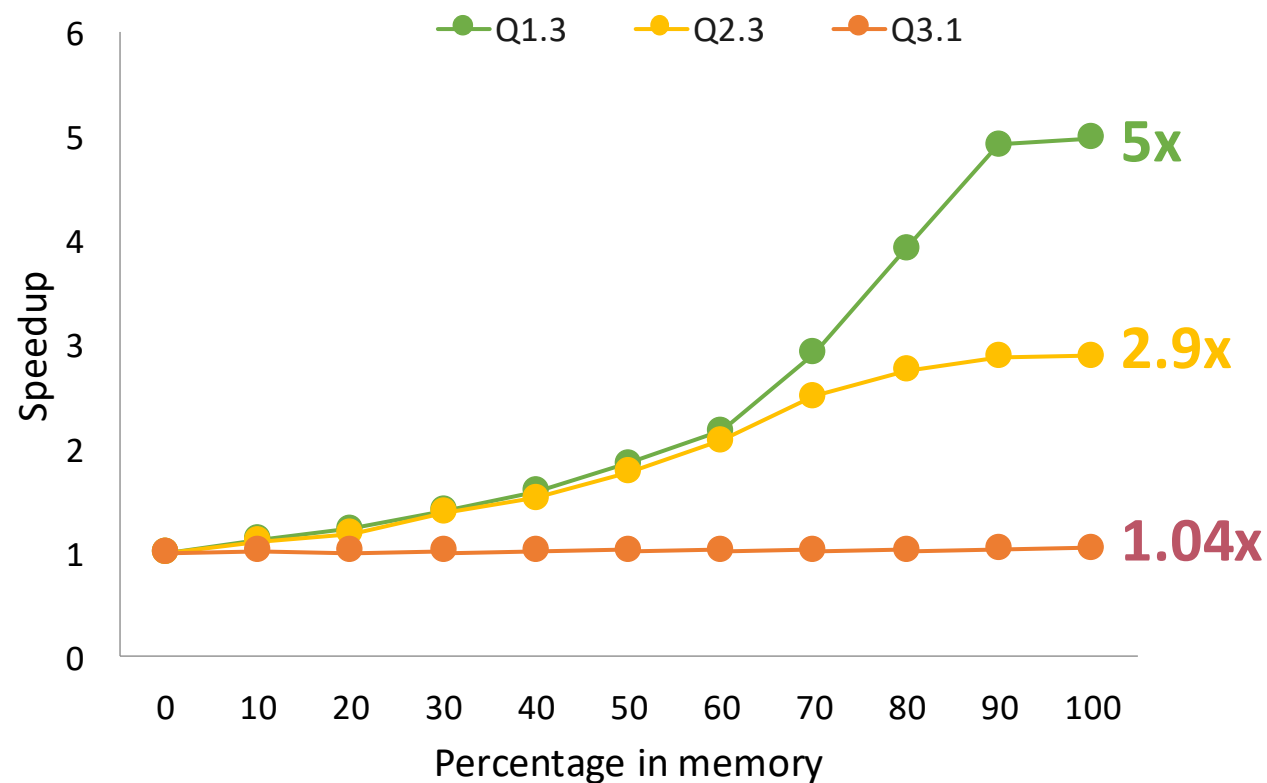
2x 12-core Xeon Gold 5118 (Skylake)

4x (NVMe, PCIe 3 x8)

SSB1000

96GB working set

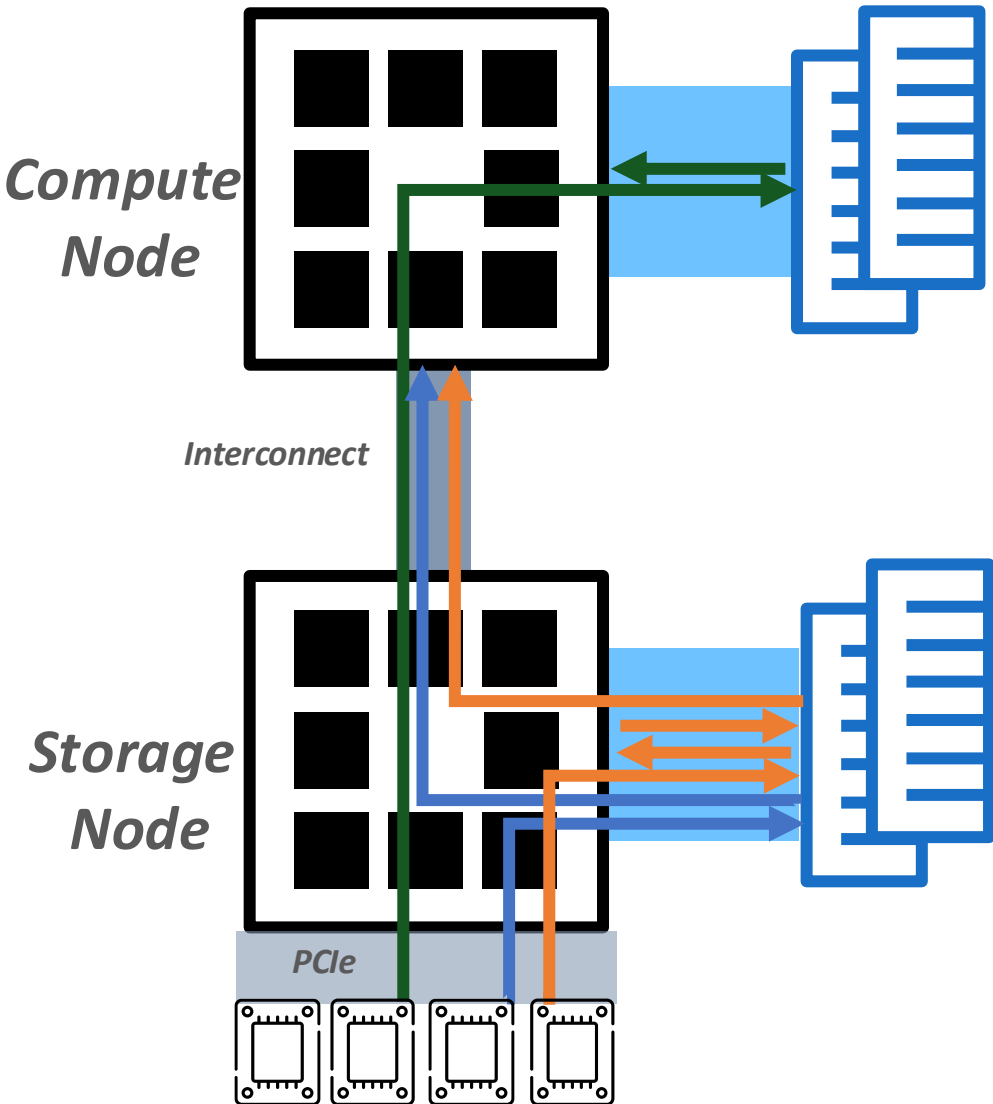
[Nicholson et al, VLDB'24]



HPCache: cache what most accelerates pipeline

Multiple paths from storage to compute

[Nicholson et al, ICDE'26]



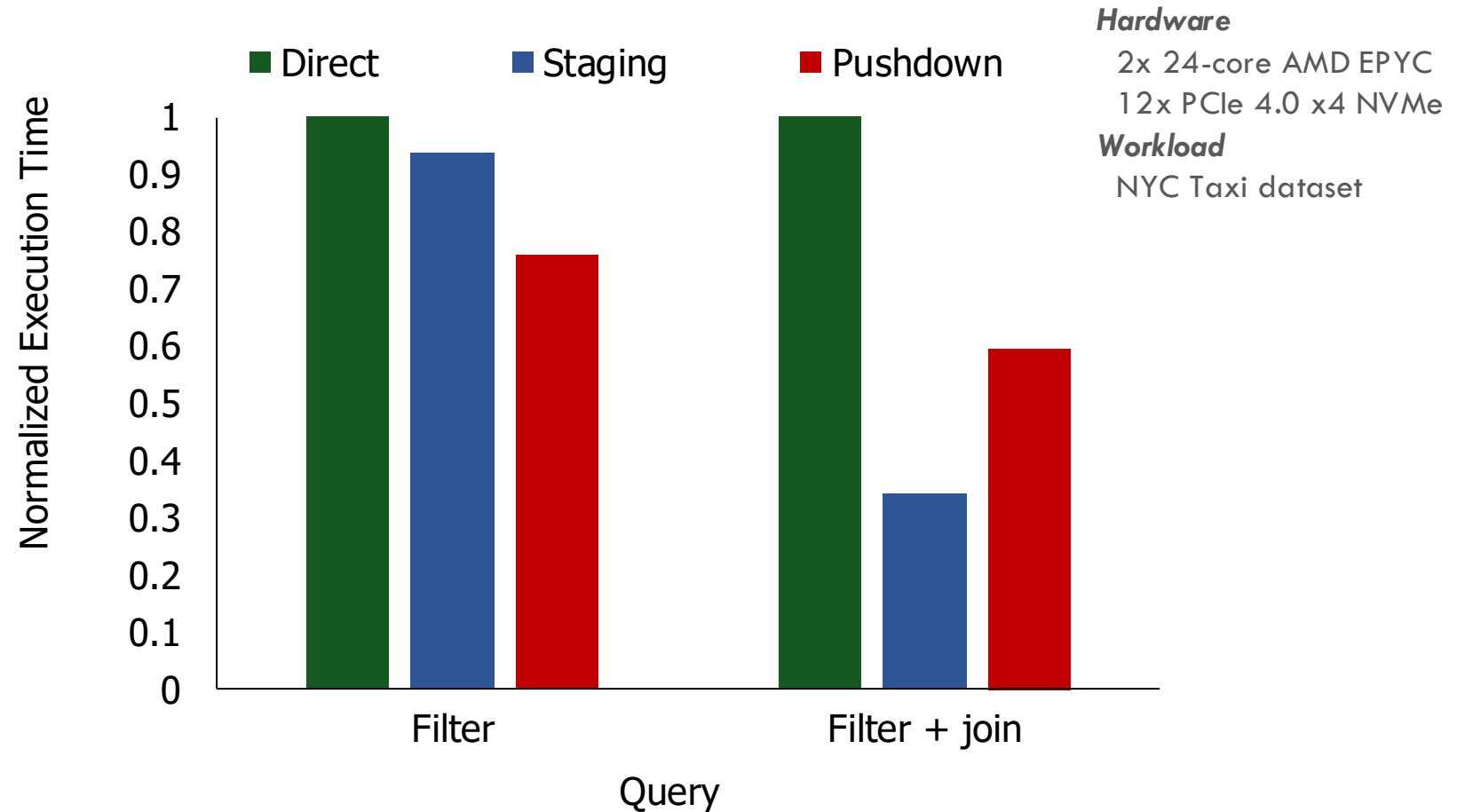
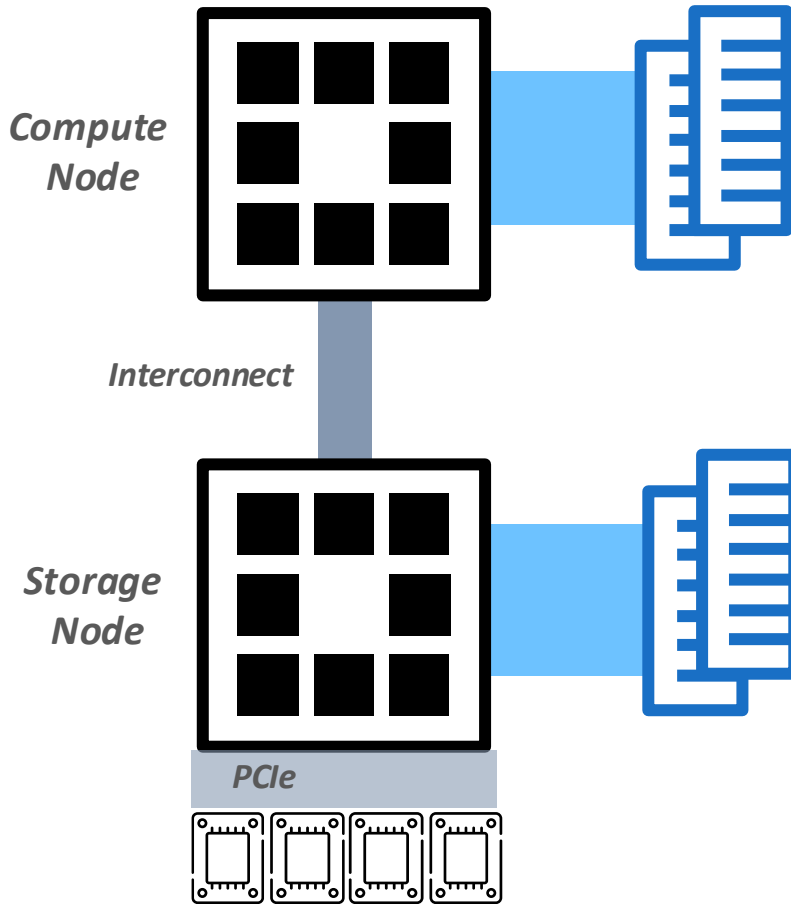
1. Direct

2. Staging

3. Pushdown

Multiple paths from storage to compute

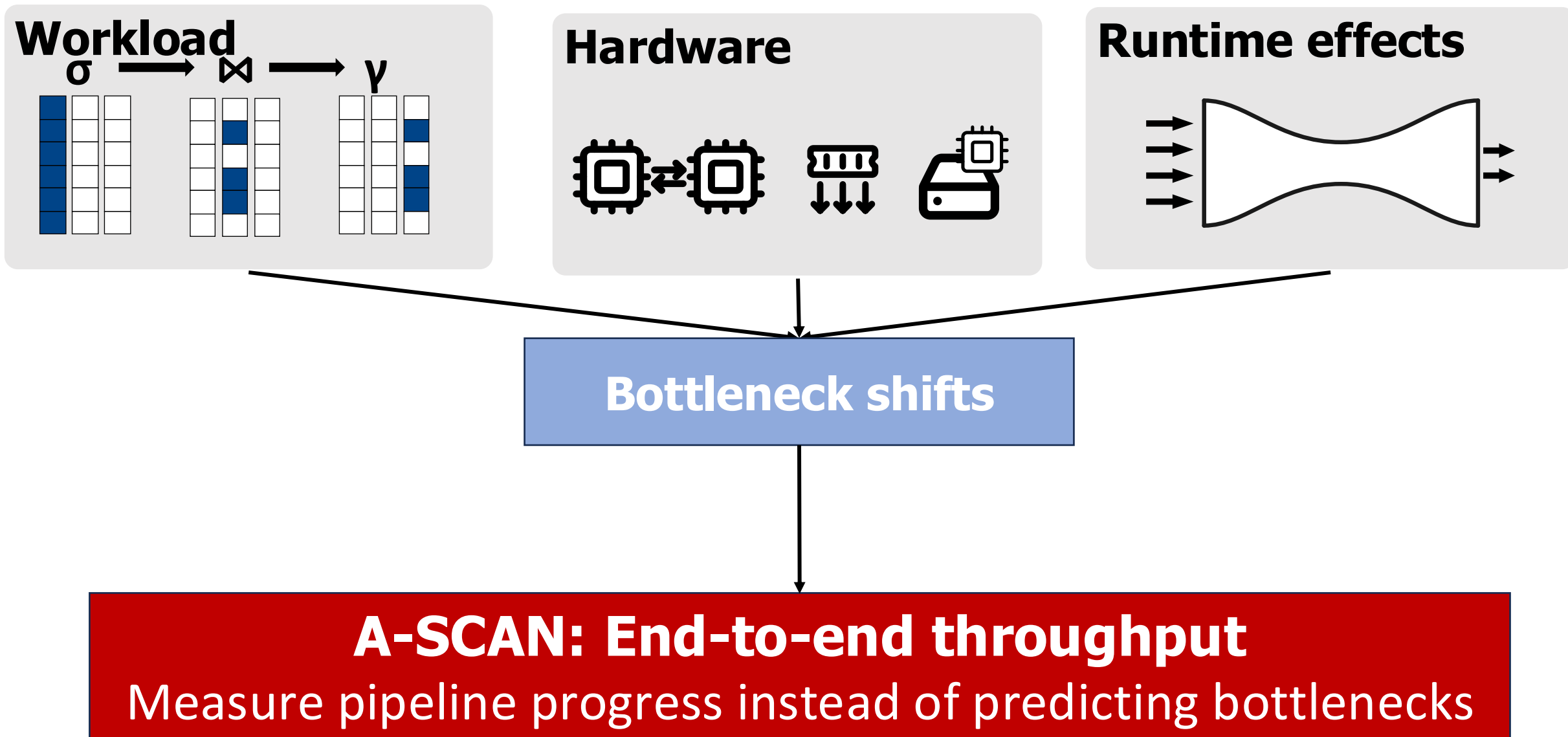
[Nicholson et al, ICDE'26]



Fastest strategy depends on query and hardware

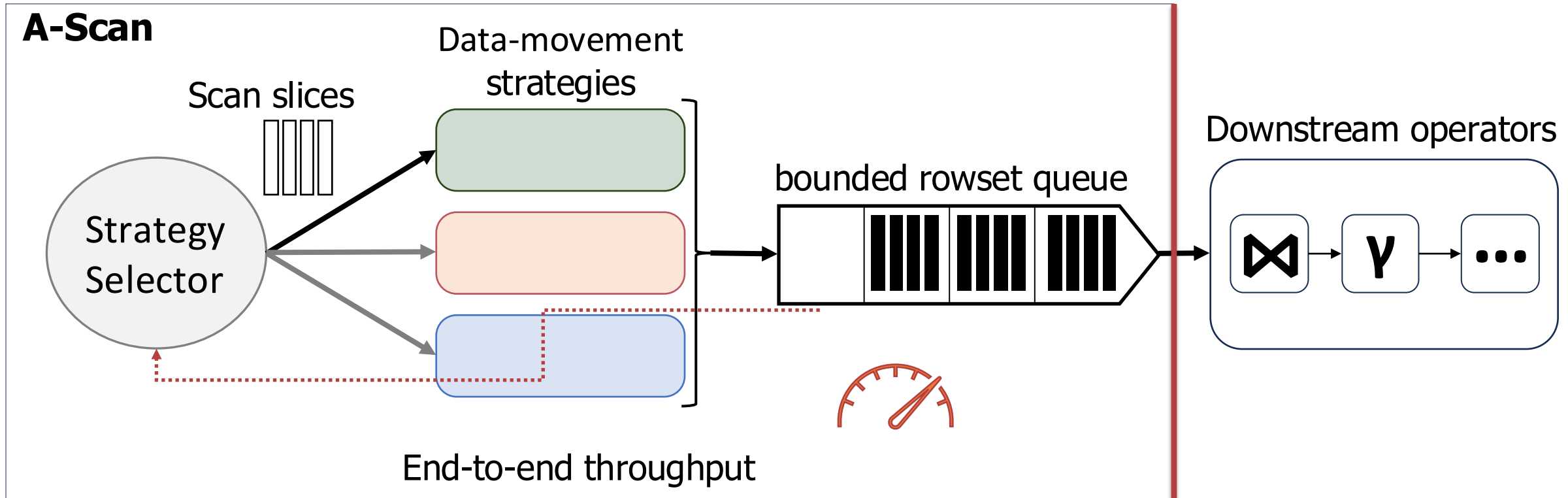
Too many interacting factors to model robustly

[Nicholson et al, ICDE'26]



Query plan stays fixed; data movement strategy changes

[Nicholson et al, ICDE'26]



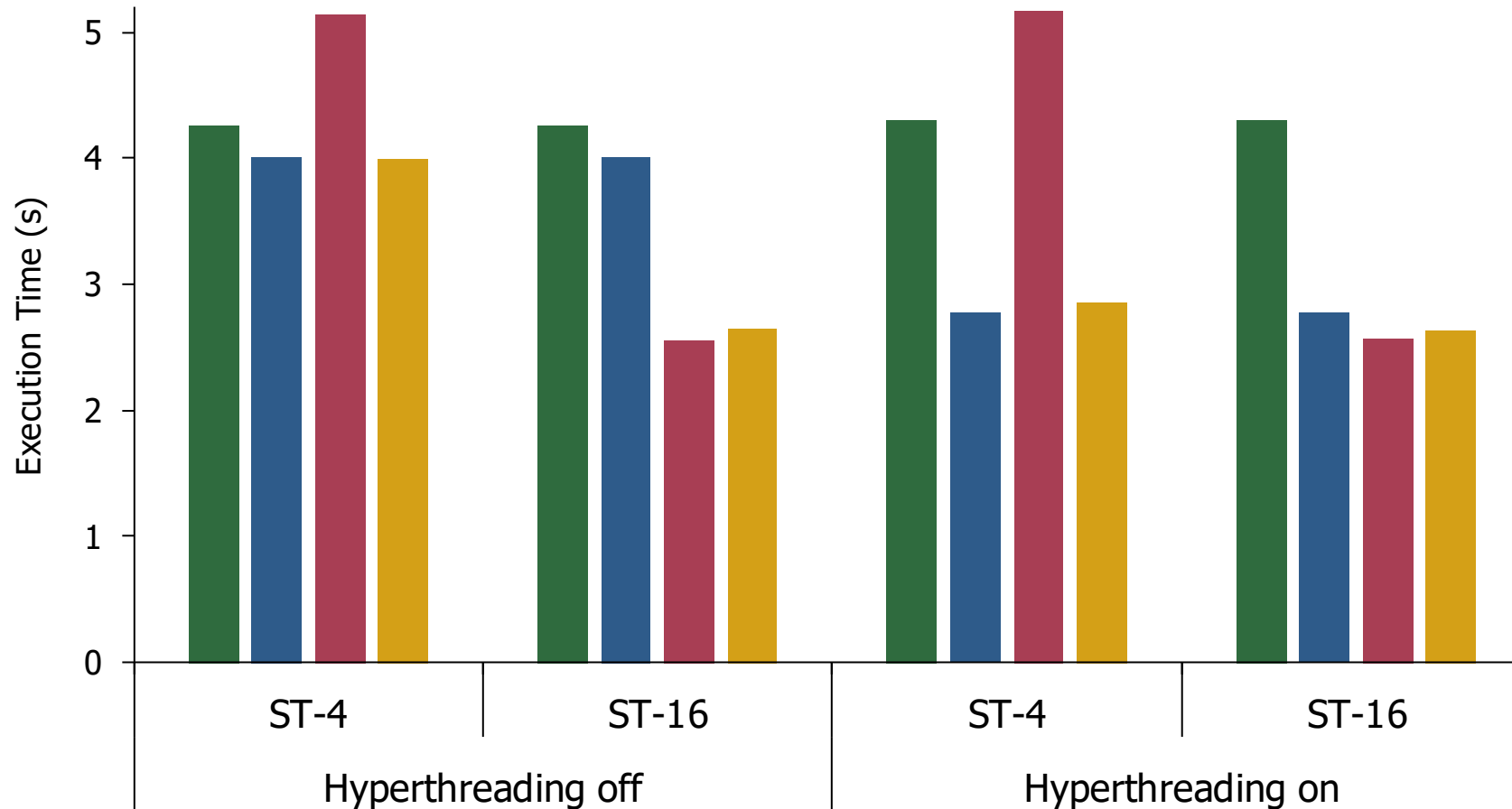
sample strategies → exploit fastest → resample on drift

End-to-end throughput guides data movement

A-Scan follows the best strategy

[Nicholson et al, ICDE'26]

■ Direct ■ Staging ■ Pushdown ■ A-Scan



Hardware

2x 24-core AMD EPYC 7413

12x PCIe 4.0 x4 NVMe

Workload

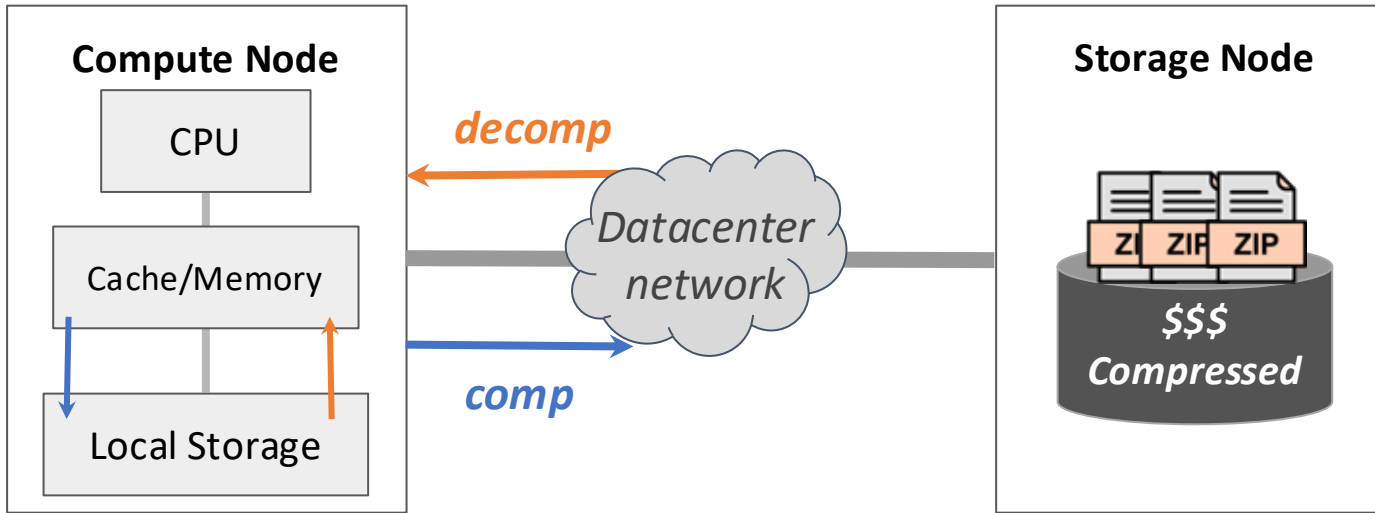
SSB Q3.2 SF=1000

ST-n = n storage threads (for pushdown)

Throughput-guided adaptation across hardware setups

Reduce data: HW compression

[Jiang et al, ADMS'26]



- + Reduce storage footprint
- + Minimize I/O and data movement overhead

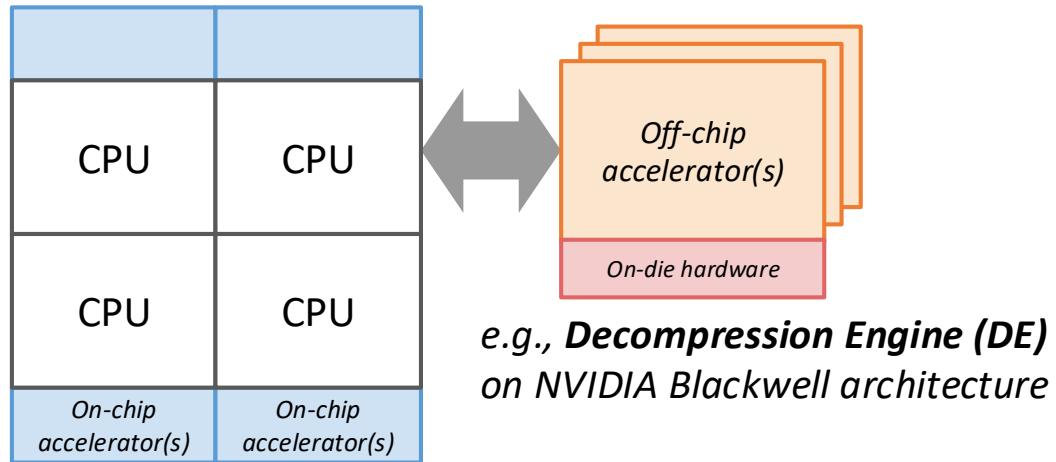
Datacenter tax: 32-40% CPU cycles
De-/compression: 14-31% of the Datacenter tax
[Mishra et al, ISCA'23]

- Additional critical path component
- Wasted CPU cycles
- LLC & memBW contention

De-/compression on critical path: schedule efficiently

ROI-guided decompression paths

[Jiang et al, ADMS'26]



e.g., Granite Rapids (Intel Xeon 6)

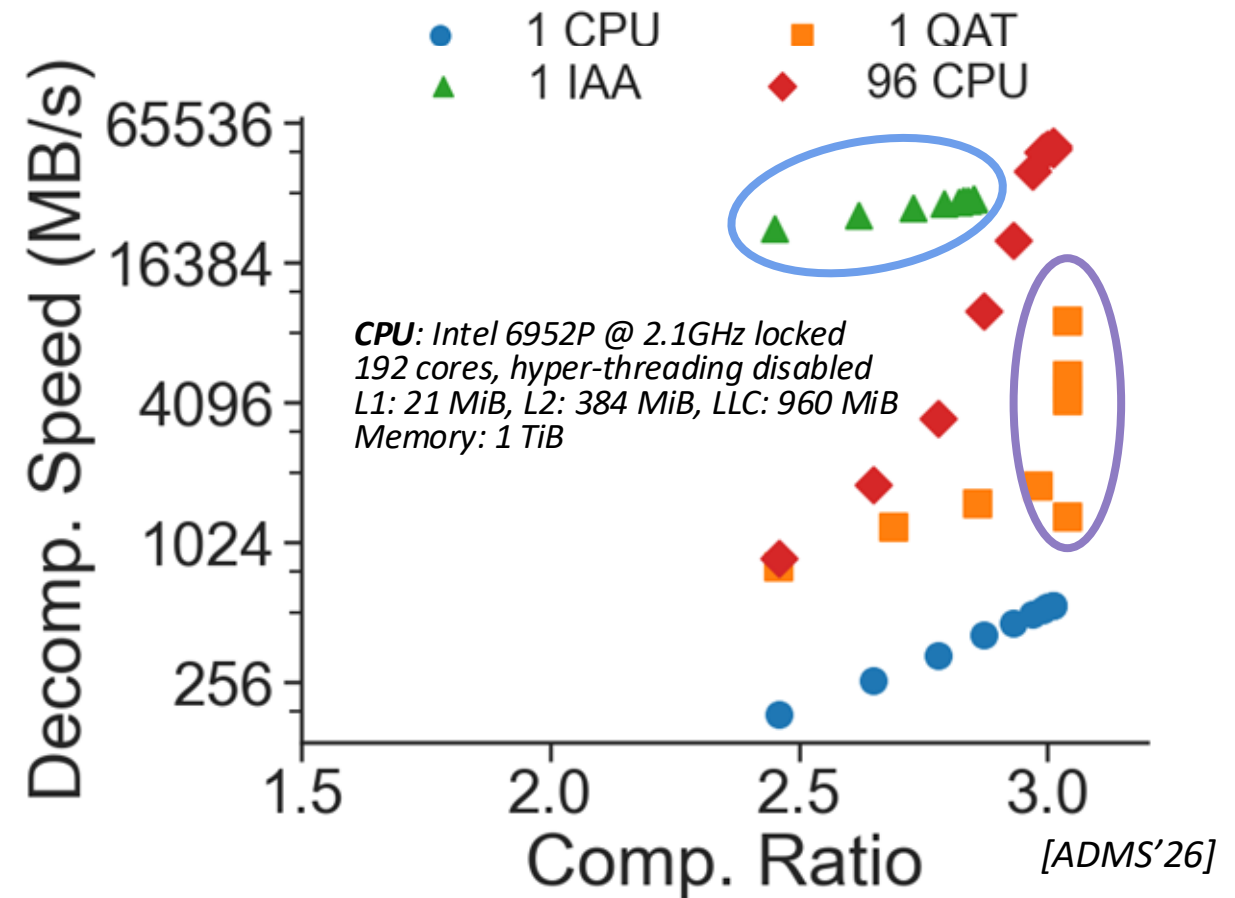
IAA: In-memory Analytics Accelerator

- Analytics primitives, CRC, **de-/compression**
- **Deflate** compression algorithm only

QAT: Quick Assist Technology

- de-/encryption and **de-/compression**
- Multiple compression algorithms

Deflate (dynamic)



Offload de-/compression adaptively for efficiency

ROI-guided resource-centric systems

	Compute-centric	Resource-centric
Core focus	Instruction throughput	Work per joule and per dollar
Data movement	Static plans, fixed hierarchy	Topology-aware, throughput-guided
Memory policy	Access frequency, fixed partitions	Bandwidth- and benefit-aware tiering
Advantage	Simple; fastest when compute-bound	Reclaims stranded resources; portable
Drawback	Portability debt, stranded memory	Runtime complexity

**A system is only as efficient
as its least adaptive component.**