Research Track – Data Management on Novel Hardware

CXL Memory Performance for In-Memory Data Processing

Marcel Weisgut
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

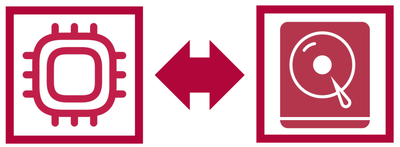
Daniel Ritter
SAP, Walldorf, Germany

Pınar Tözün
IT University of Copenhagen, Copenhagen, Denmark

Lawrence Benson
Technische Universität Munich, Munich, Germany

Tilmann Rabl
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

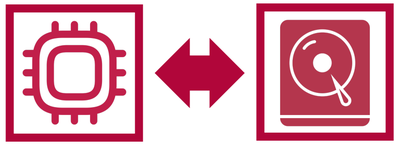
Resource Disaggregation



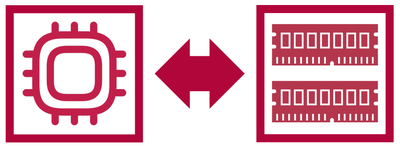
Individual resource scaling → improves utilization → reduces TCO [1].
Widely adopted.

[1] Pang and Wang, 2024, Understanding the Performance Implications of the Design Principles in Storage-Disaggregated Databases, PACMMOD.
[2] Lerner and Alonso, 2024, CXL and the Return of Scale-Up Database Engines, PVLDB.
[3] Wang et al., 2022, The case for distributed shared-memory databases with RDMA-enabled memory disaggregation, PVLDB.
[4] Chronis et al., 2025, Databases in the Era of Memory-Centric Computing, CIDR.

Resource Disaggregation



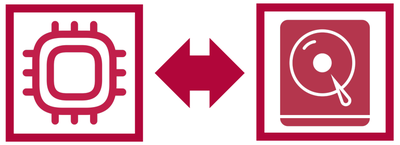
Individual resource scaling → improves utilization → reduces TCO [1].
Widely adopted.



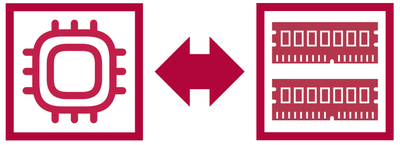
Increasing efforts on memory-disaggregated database systems [2, 3].

[1] Pang and Wang, 2024, Understanding the Performance Implications of the Design Principles in Storage-Disaggregated Databases, PACMMOD.
[2] Lerner and Alonso, 2024, CXL and the Return of Scale-Up Database Engines, PVLDB.
[3] Wang et al., 2022, The case for distributed shared-memory databases with RDMA-enabled memory disaggregation, PVLDB.
[4] Chronis et al., 2025, Databases in the Era of Memory-Centric Computing, CIDR.

Resource Disaggregation



Individual resource scaling → improves utilization → reduces TCO [1].
Widely adopted.



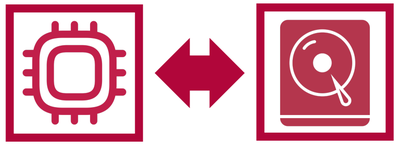
Increasing efforts on memory-disaggregated database systems [2, 3].

CXL

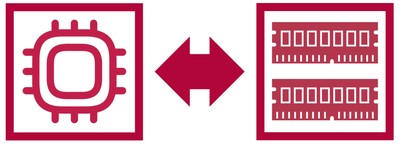
Interconnect technology to implement memory disaggregation [2, 4].

[1] Pang and Wang, 2024, Understanding the Performance Implications of the Design Principles in Storage-Disaggregated Databases, PACMMOD.
[2] Lerner and Alonso, 2024, CXL and the Return of Scale-Up Database Engines, PVLDB.
[3] Wang et al., 2022, The case for distributed shared-memory databases with RDMA-enabled memory disaggregation, PVLDB.
[4] Chronis et al., 2025, Databases in the Era of Memory-Centric Computing, CIDR.

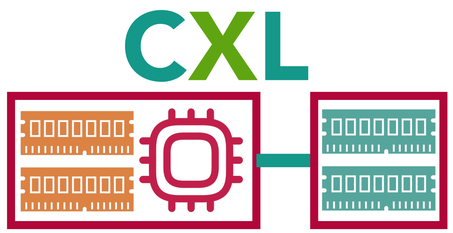
Resource Disaggregation



Individual resource scaling → improves utilization → reduces TCO [1].
Widely adopted.



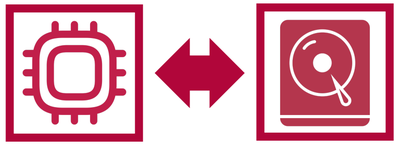
Increasing efforts on memory-disaggregated database systems [2, 3].



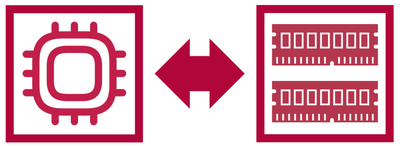
Interconnect technology to implement memory disaggregation [2, 4].
→ Breaks assumption that memory is close to CPU [2].

[1] Pang and Wang, 2024, Understanding the Performance Implications of the Design Principles in Storage-Disaggregated Databases, PACMMOD.
[2] Lerner and Alonso, 2024, CXL and the Return of Scale-Up Database Engines, PVLDB.
[3] Wang et al., 2022, The case for distributed shared-memory databases with RDMA-enabled memory disaggregation, PVLDB.
[4] Chronis et al., 2025, Databases in the Era of Memory-Centric Computing, CIDR.

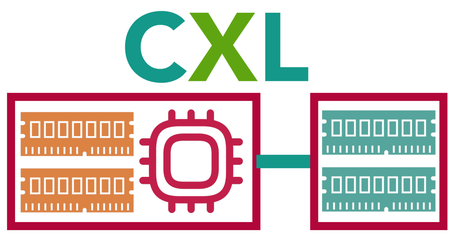
Resource Disaggregation



Individual resource scaling → improves utilization → reduces TCO [1].
Widely adopted.



Increasing efforts on memory-disaggregated database systems [2, 3].



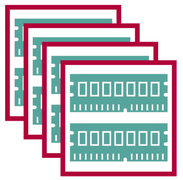
Interconnect technology to implement memory disaggregation [2, 4].
→ Breaks assumption that memory is close to CPU [2].

Extensive experimentation required to identify what part of a database engine can be stored in CXL memory and what part should reside in CPU memory [2].

[1] Pang and Wang, 2024, Understanding the Performance Implications of the Design Principles in Storage-Disaggregated Databases, PACMMOD.
[2] Lerner and Alonso, 2024, CXL and the Return of Scale-Up Database Engines, PVLDB.
[3] Wang et al., 2022, The case for distributed shared-memory databases with RDMA-enabled memory disaggregation, PVLDB.
[4] Chronis et al., 2025, Databases in the Era of Memory-Centric Computing, CIDR.

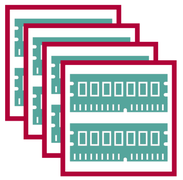
Contributions

Contributions



Memory access performance evaluation on a server with four CXL memory devices.

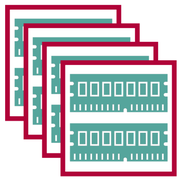
Contributions



Memory access performance evaluation on a server with four CXL memory devices.

Column scans & B⁺-Tree workloads with data across up to four CXL devices.

Contributions



Memory access performance evaluation on a server with four CXL memory devices.

Column scans & B⁺-Tree workloads with data across up to four CXL devices.

TPC-H evaluation with simple data placement heuristics on the in-memory DBMS Hyrise.

Contributions

Memory access performance evaluation on a server with four CXL memory devices.

Column scans & B⁺-Tree workloads with data across up to four CXL devices.

TPC-H evaluation with simple data placement heuristics on the in-memory DBMS Hyrise.

Economic viability study of CXL memory devices.

Contributions

Memory access performance evaluation on a server with four CXL memory devices.

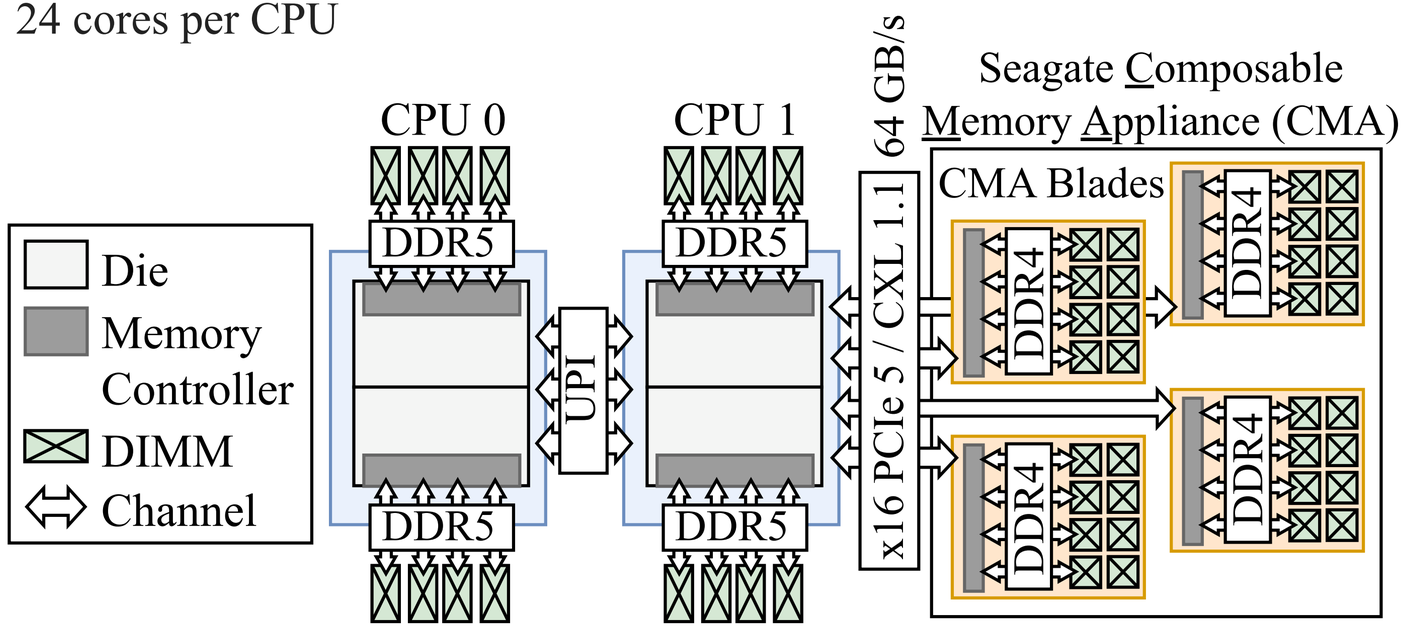
Column scans & B⁺-Tree workloads with data across up to four CXL devices.

TPC-H evaluation with simple data placement heuristics on the in-memory DBMS Hyrise.

Economic viability study of CXL memory devices.

Hardware Setup

2× Intel Xeon Gold 6542Y (Emerald Rapids)
24 cores per CPU

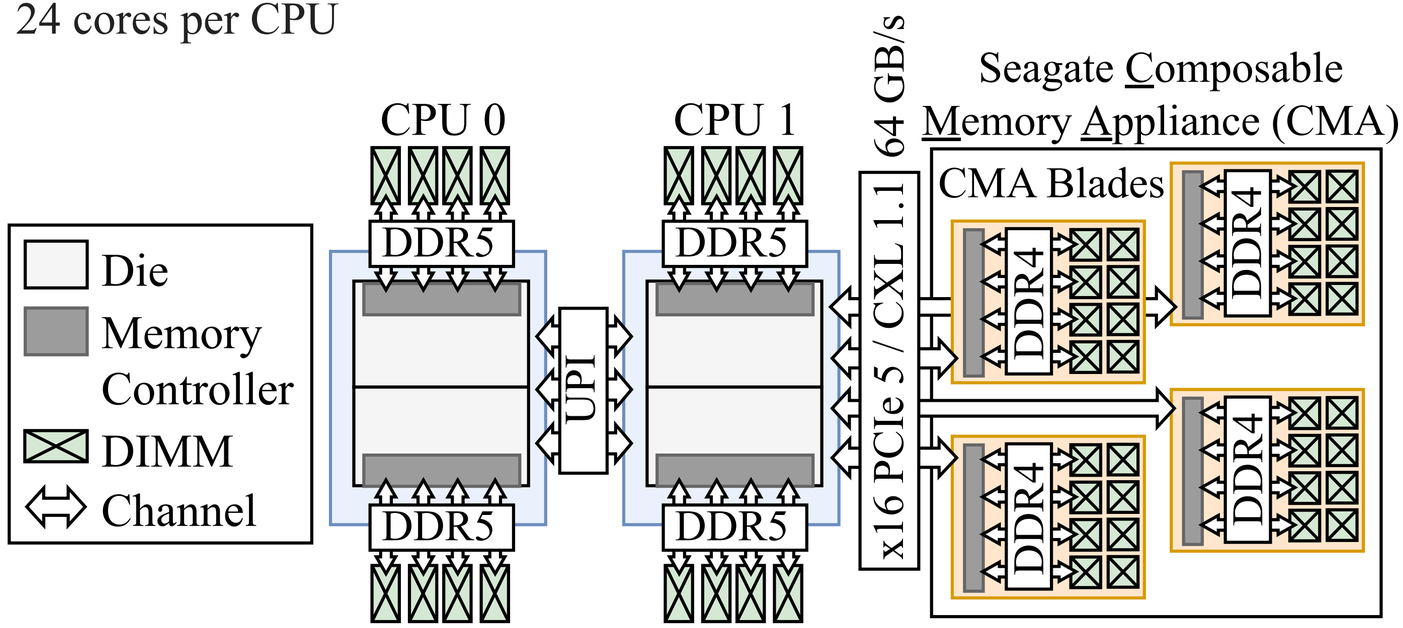


Memory per CPU:
8 x 32 GiB DDR5 (4800 MT/s)

Memory per CMA blade:
8 x 128 GiB DDR4 (1866 MT/s)

Hardware Setup

2× Intel Xeon Gold 6542Y (Emerald Rapids)
24 cores per CPU



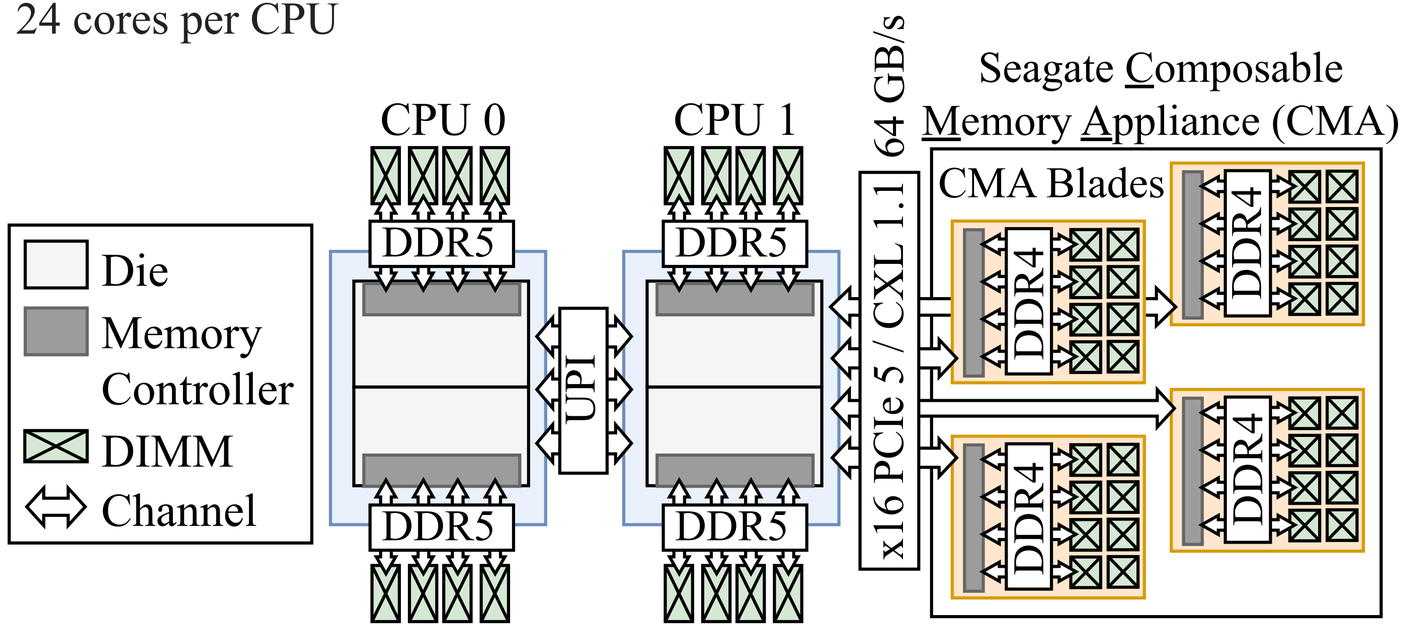
Memory per CPU:
8 x 32 GiB DDR5 (4800 MT/s)

Memory per CMA blade:
8 x 128 GiB DDR4 (1866 MT/s)

Memory of each CXL device configured as CPU-less NUMA node (system RAM).

Hardware Setup

2× Intel Xeon Gold 6542Y (Emerald Rapids)
24 cores per CPU



Memory per CPU:
8 x 32 GiB DDR5 (4800 MT/s)

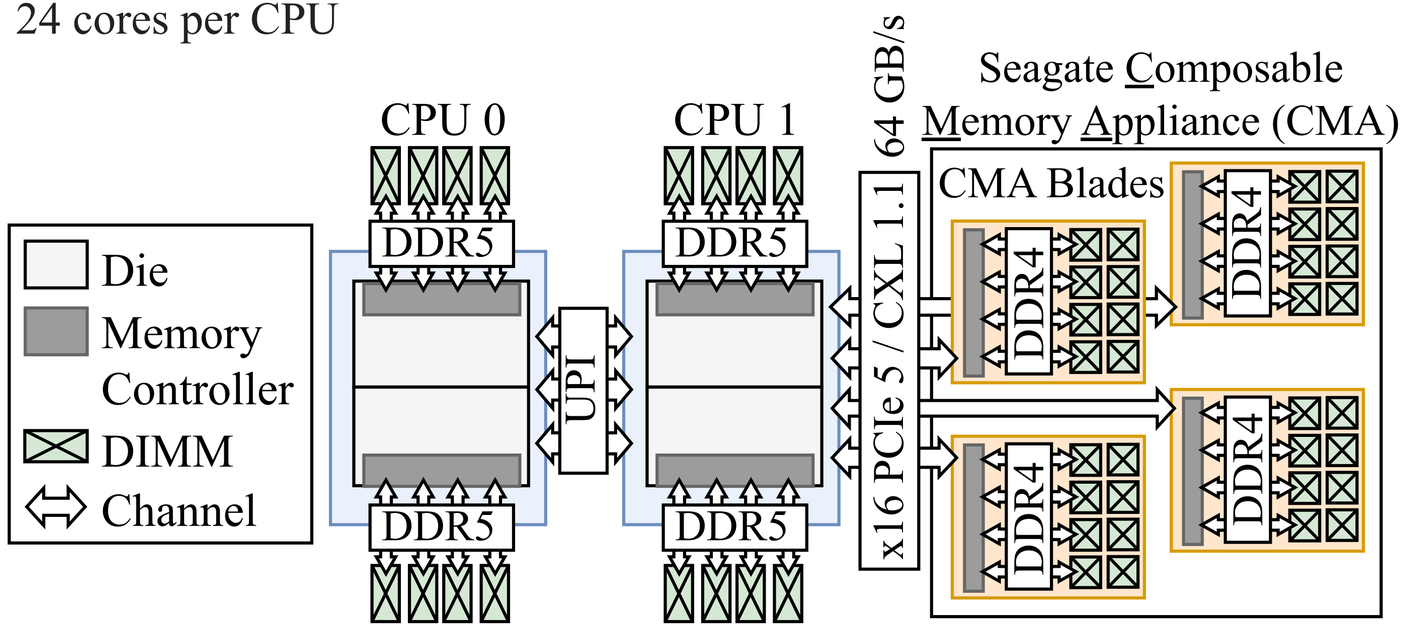
Memory per CMA blade:
8 x 128 GiB DDR4 (1866 MT/s)

Memory of each CXL device configured as CPU-less NUMA node (system RAM).

Impact of storing table data in CXL memory ?

Hardware Setup

2× Intel Xeon Gold 6542Y (Emerald Rapids)
24 cores per CPU



Memory per CPU:
8 x 32 GiB DDR5 (4800 MT/s)

Memory per CMA blade:
8 x 128 GiB DDR4 (1866 MT/s)

Memory of each CXL device configured as CPU-less NUMA node (system RAM).

Impact of storing table data in CXL memory ?

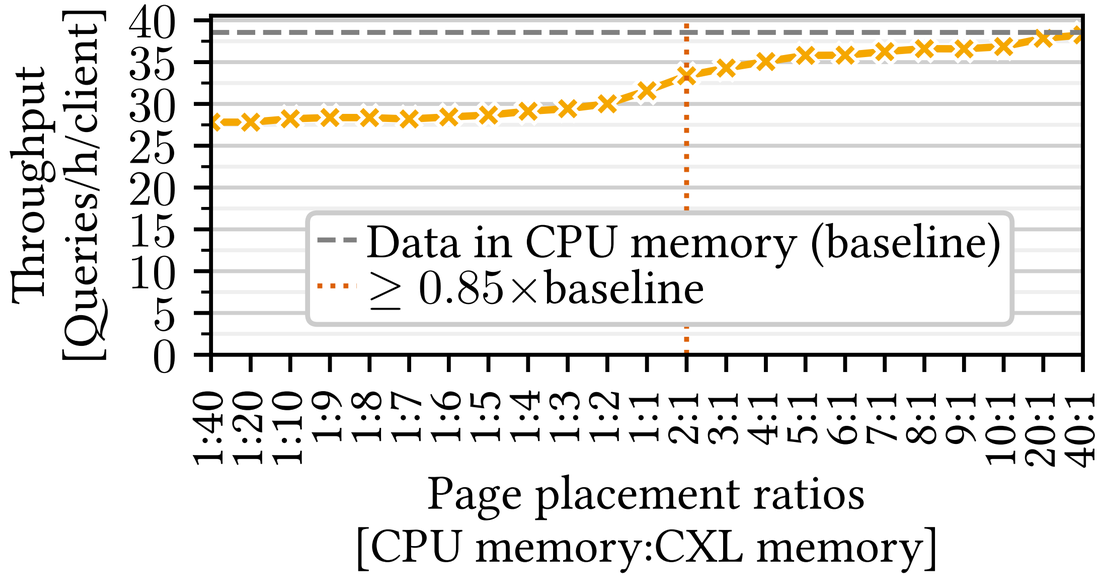
HYRISE [5]

[5] Dreseler et al., Hyrise Re-engineered: An Extensible Database System for Research in Relational In-Memory Data Management, EDBT.

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

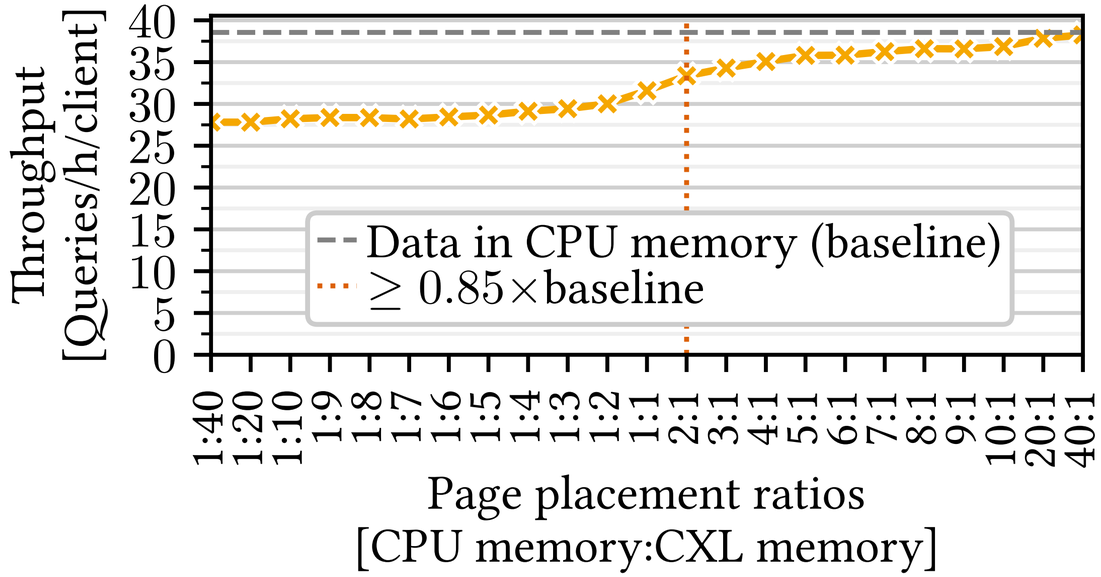
TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

Black Box Approach (Page Granularity)



TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

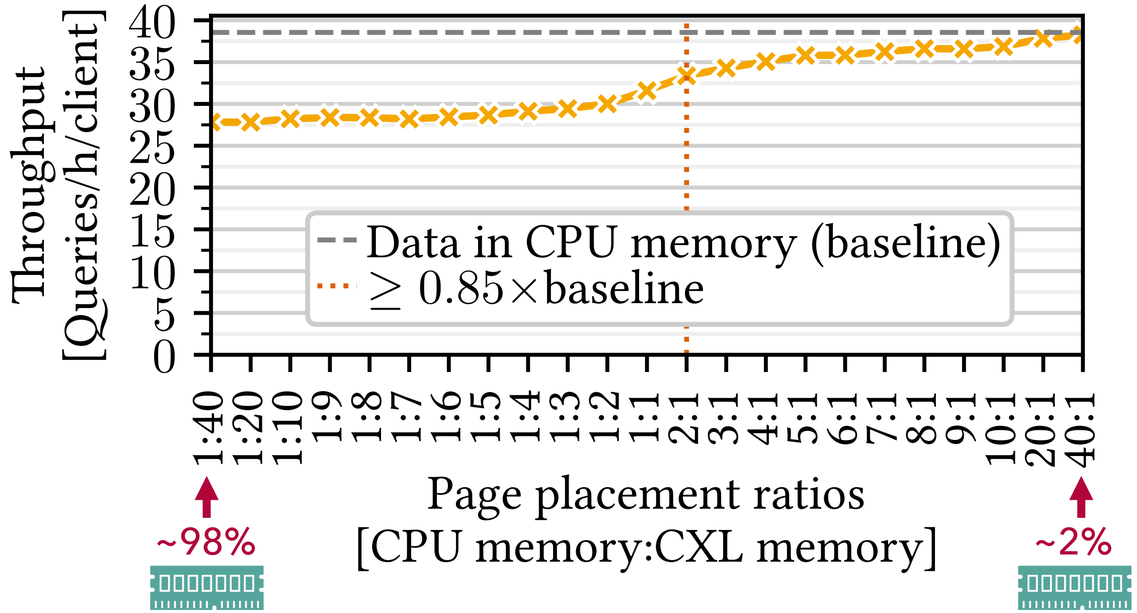
Black Box Approach (Page Granularity)



- Data placement via page interleaving across CPU memory and one CXL device

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

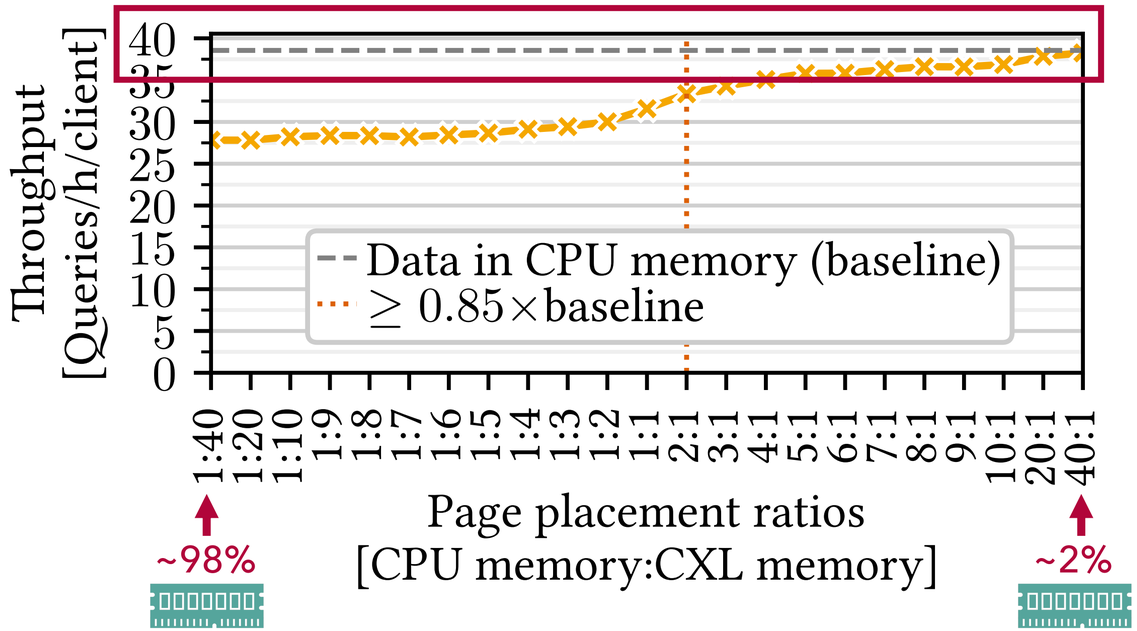
Black Box Approach (Page Granularity)



- Data placement via page interleaving across CPU memory and one CXL device

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

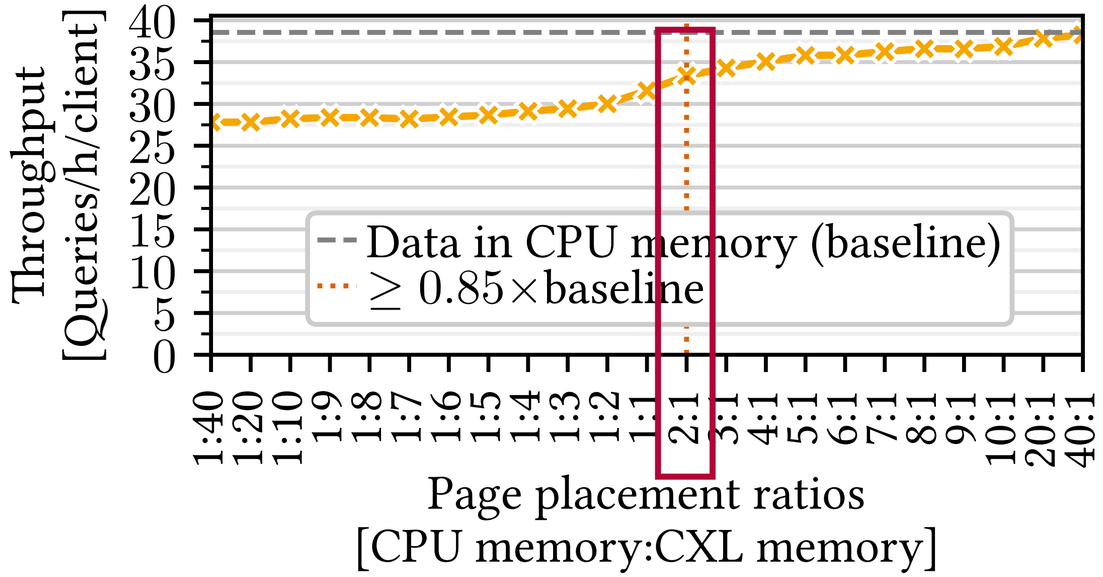
Black Box Approach (Page Granularity)



- Data placement via page interleaving across CPU memory and one CXL device

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

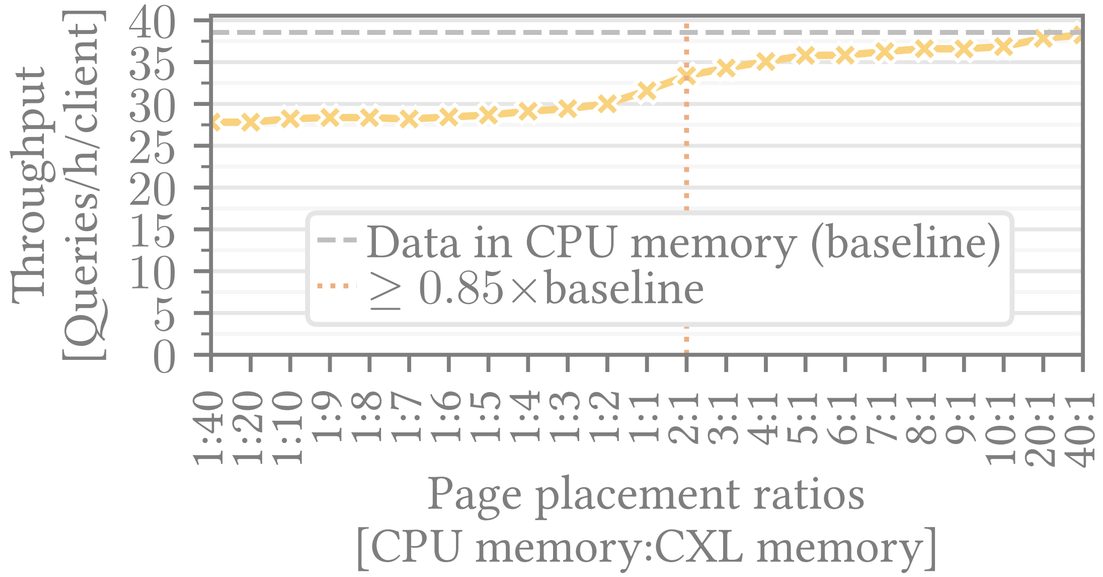
Black Box Approach (Page Granularity)



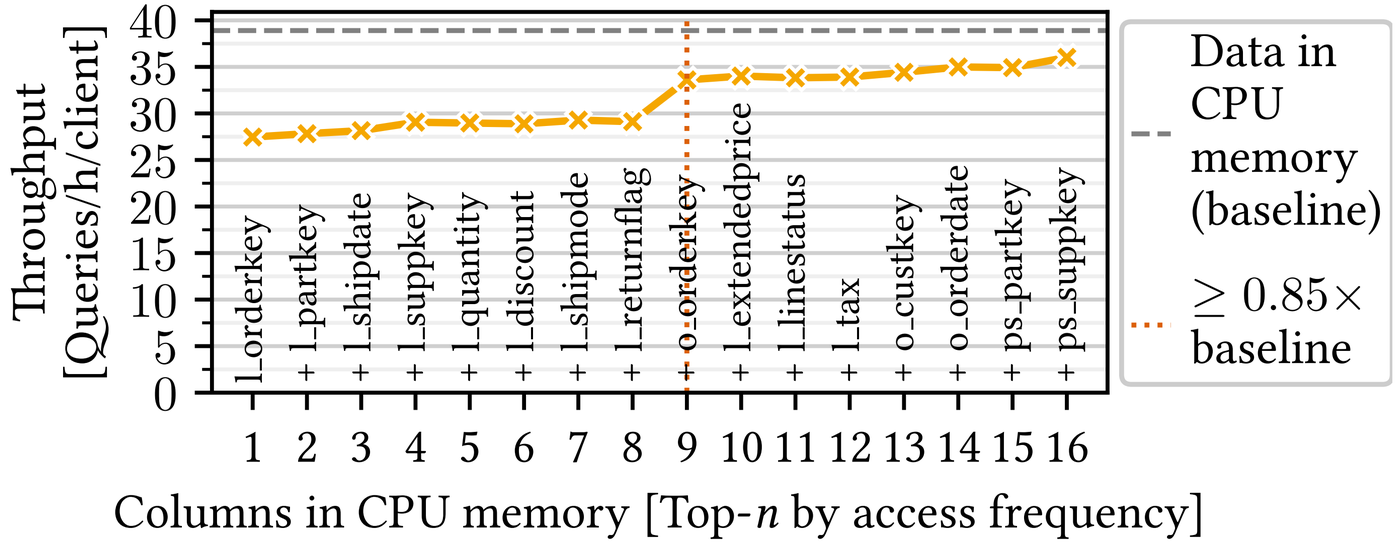
- Data placement via page interleaving across CPU memory and one CXL device
- 2:1 stores only $\frac{1}{3}$ pages in CXL memory

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

Black Box Approach (Page Granularity)



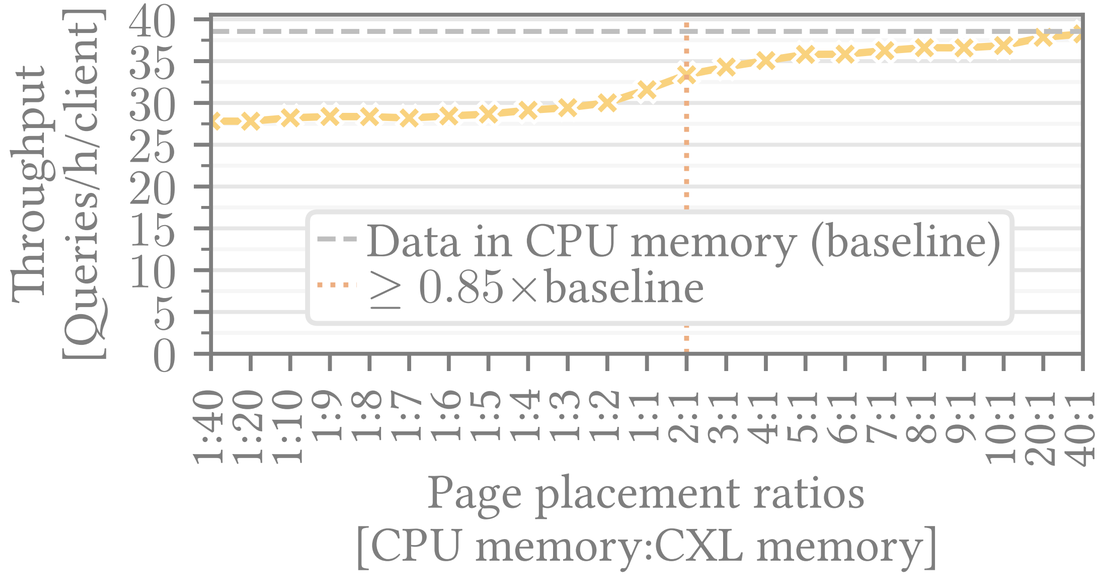
Access Frequency-Based (Column Granularity)



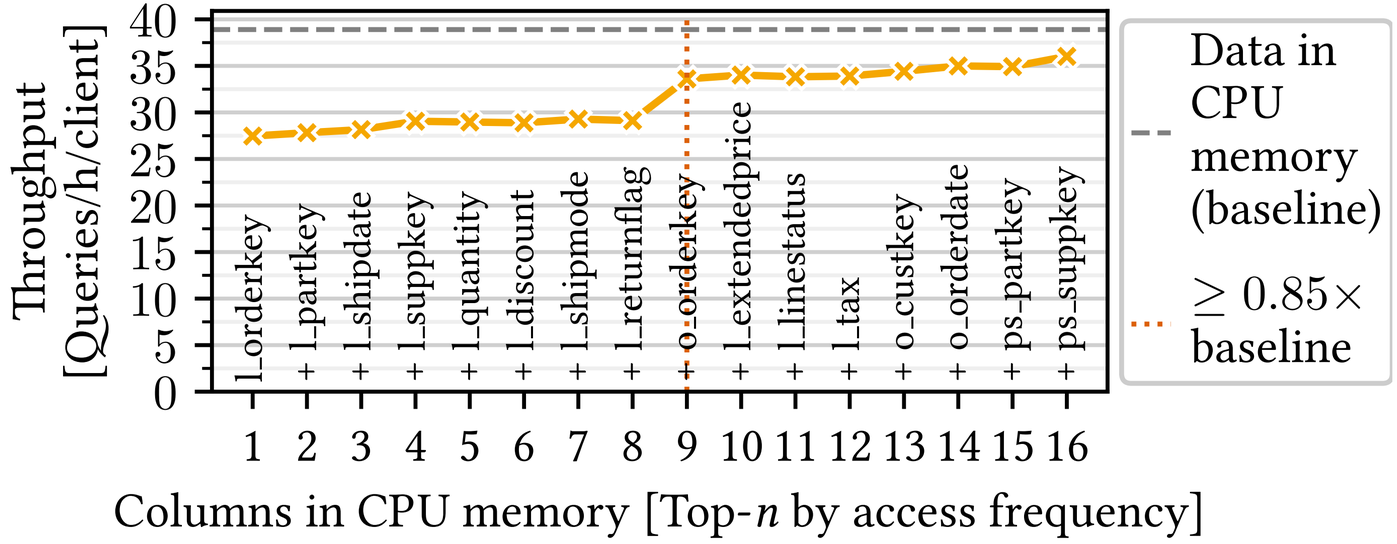
- Accumulate data accesses per column

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

Black Box Approach (Page Granularity)



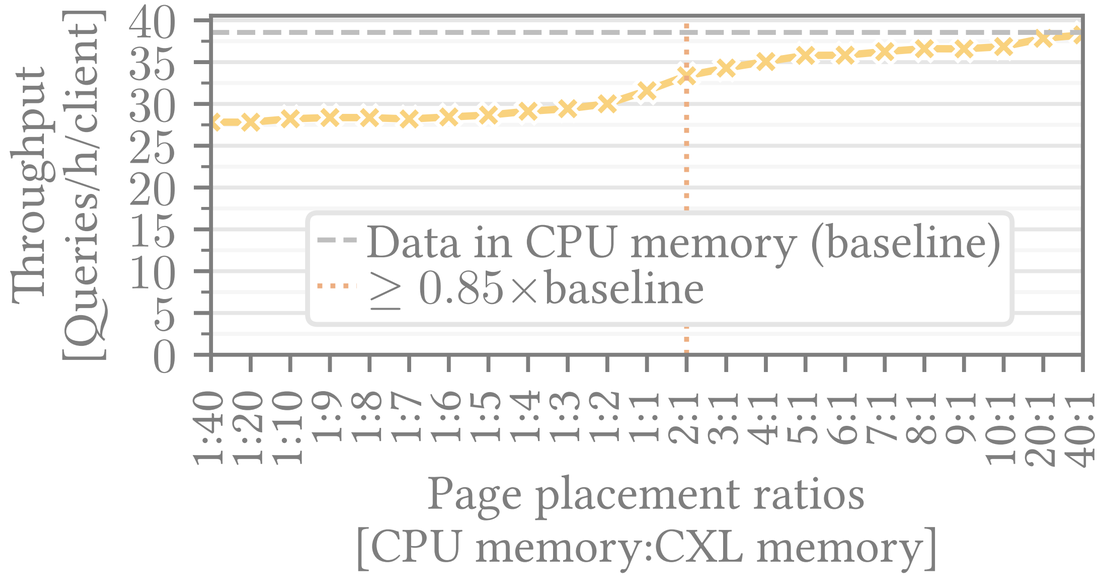
Access Frequency-Based (Column Granularity)



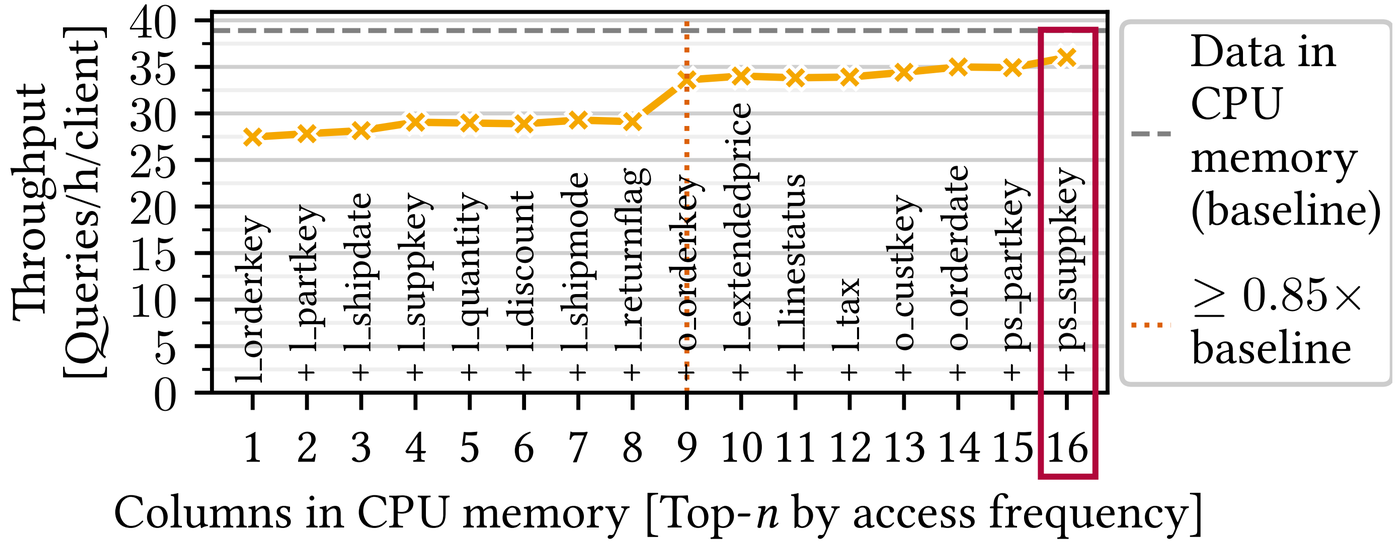
- Accumulate data accesses per column
- Columns with top-n access counts in CPU memory

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

Black Box Approach (Page Granularity)



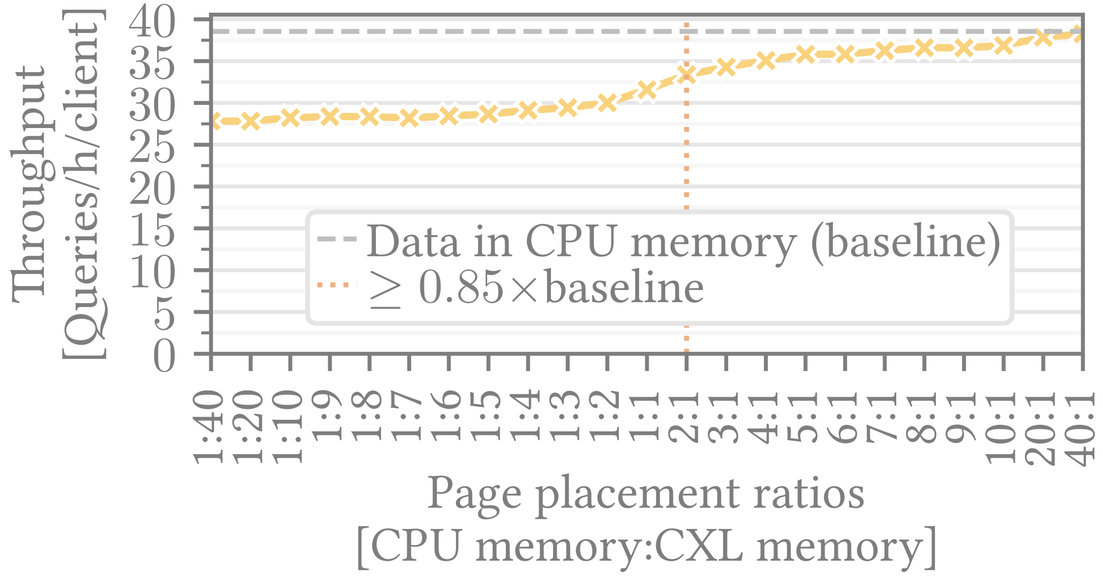
Access Frequency-Based (Column Granularity)



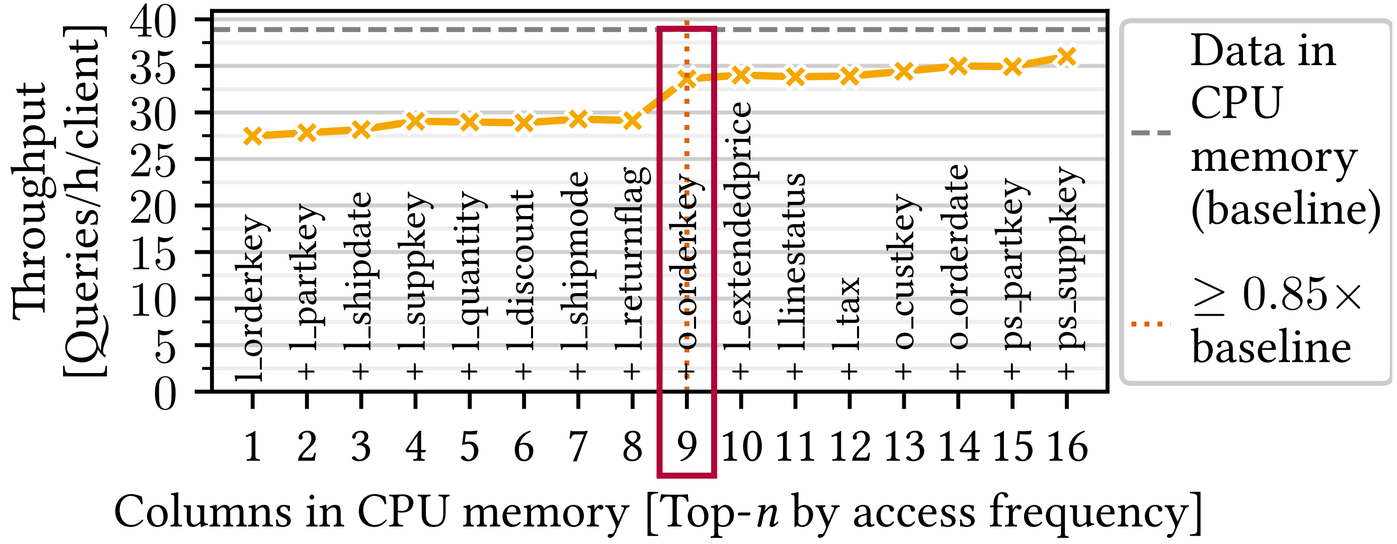
- Accumulate data accesses per column
- Columns with top-n access counts in CPU memory
- Top-16 (23% of table data size) in CPU memory results in 94% baseline performance

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

Black Box Approach (Page Granularity)



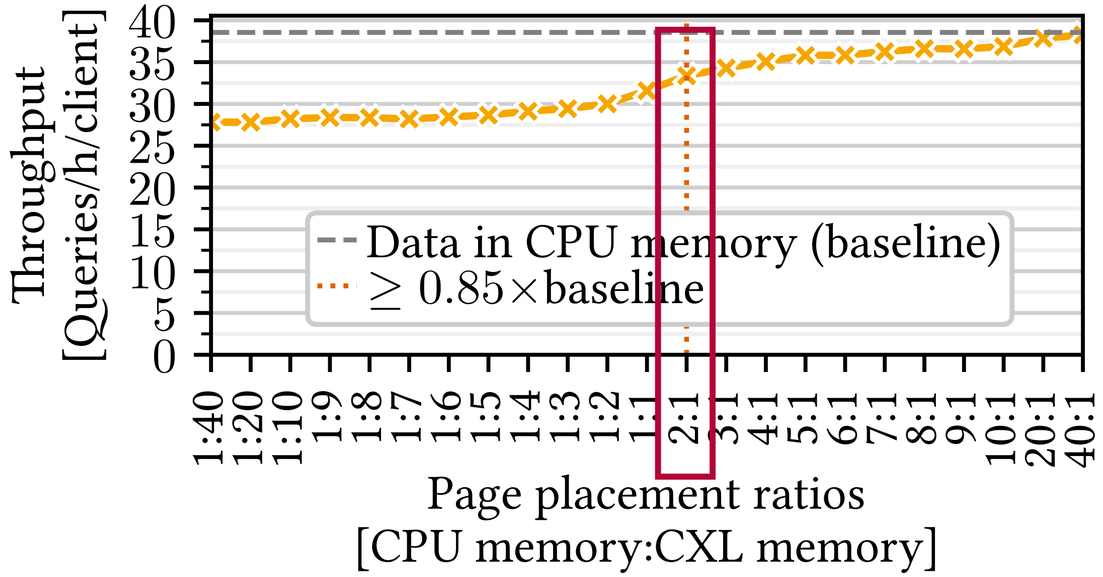
Access Frequency-Based (Column Granularity)



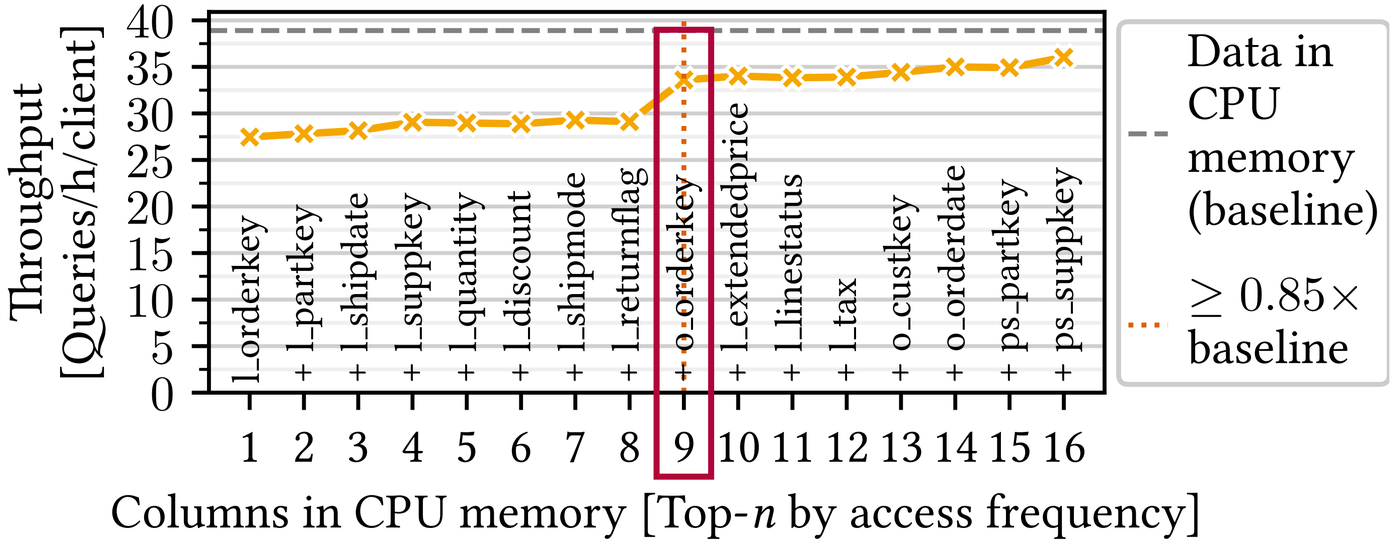
- Accumulate data accesses per column
- Columns with top-n access counts in CPU memory
- Top-16 (23% of table data size) in CPU memory results in 94% baseline performance
- Top-9 (16%) results in 87% baseline performance

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

Black Box Approach (Page Granularity)



Access Frequency-Based (Column Granularity)



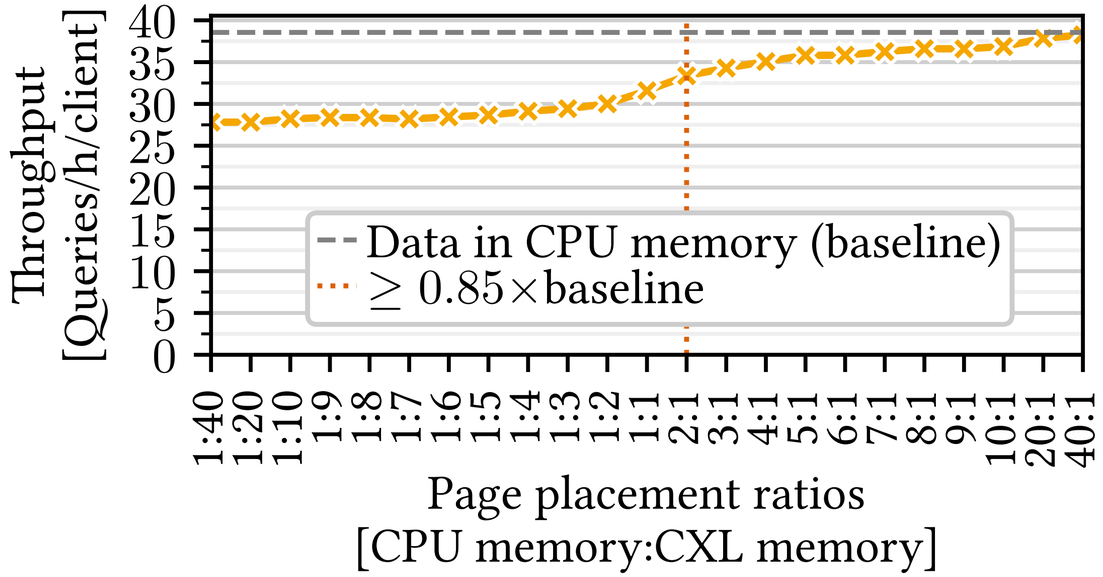
33% in CXL

vs.

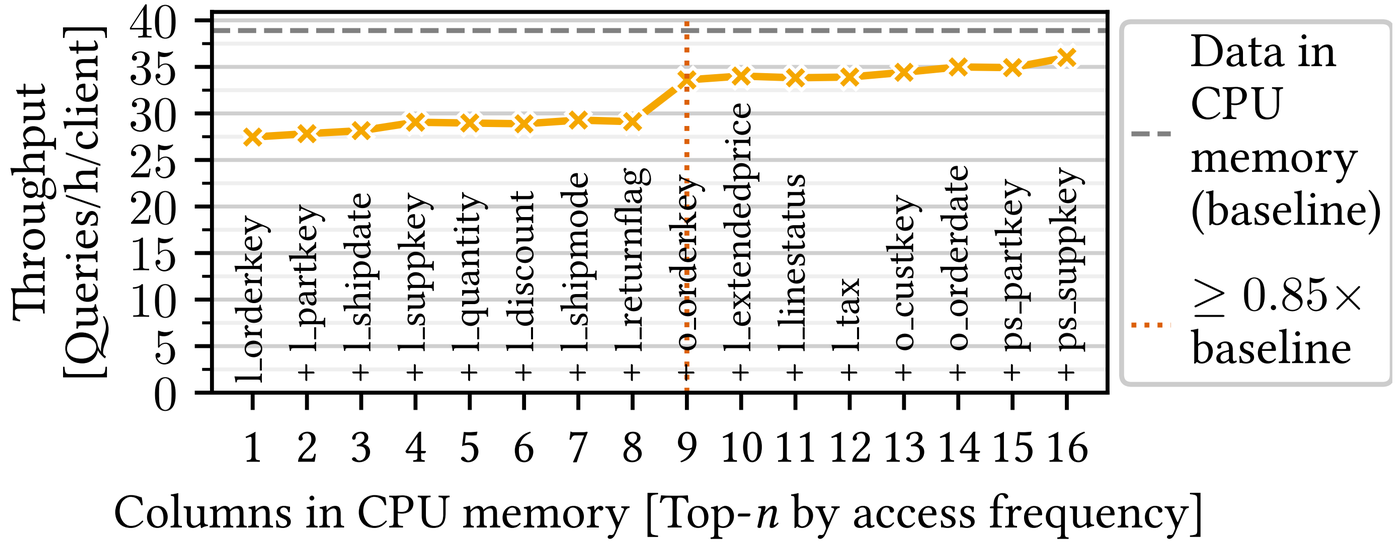
84% in CXL

TPC-H on Hyrise | SF 100 | 10 simultaneous query streams | 20 minutes

Black Box Approach (Page Granularity)



Access Frequency-Based (Column Granularity)



Conclusion

CXL memory for cold and warm data with hot data in CPU memory can lead to high performance when a DBMS uses both CPU and CXL memory.

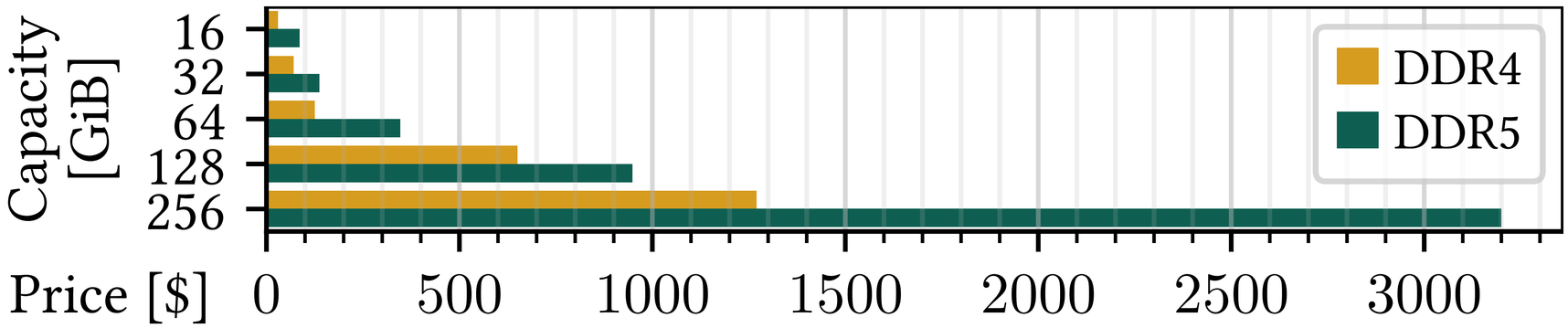
Economic Viability

Calculate expenses of CPUs and memory DIMMs.

We collected prices of NEMIX DDR4/DDR5 288-PIN RDIMMs on NewEgg.com in March 2025.

Economic Viability

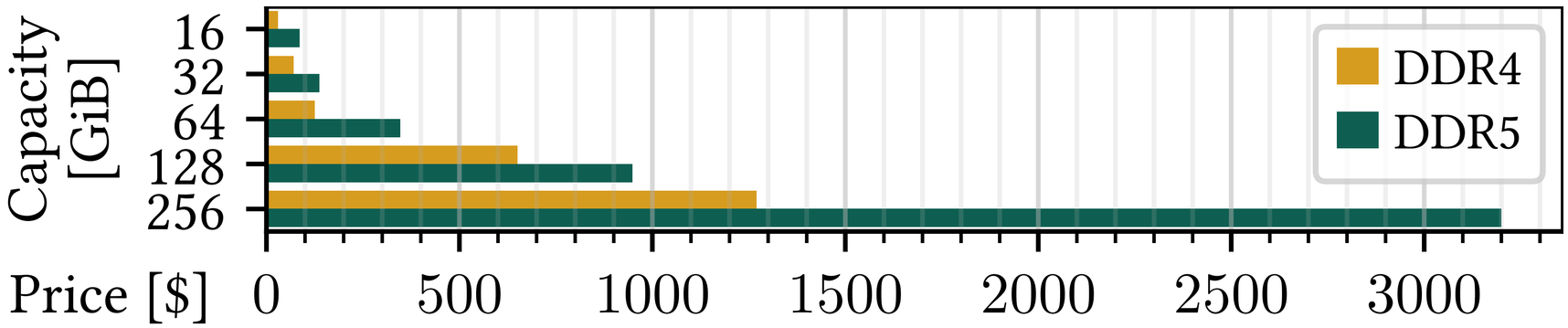
Calculate expenses of CPUs and memory DIMMs.



We collected prices of NEMIX DDR4/DDR5 288-PIN RDIMMs on NewEgg.com in March 2025.

Economic Viability

Calculate expenses of CPUs and memory DIMMs.

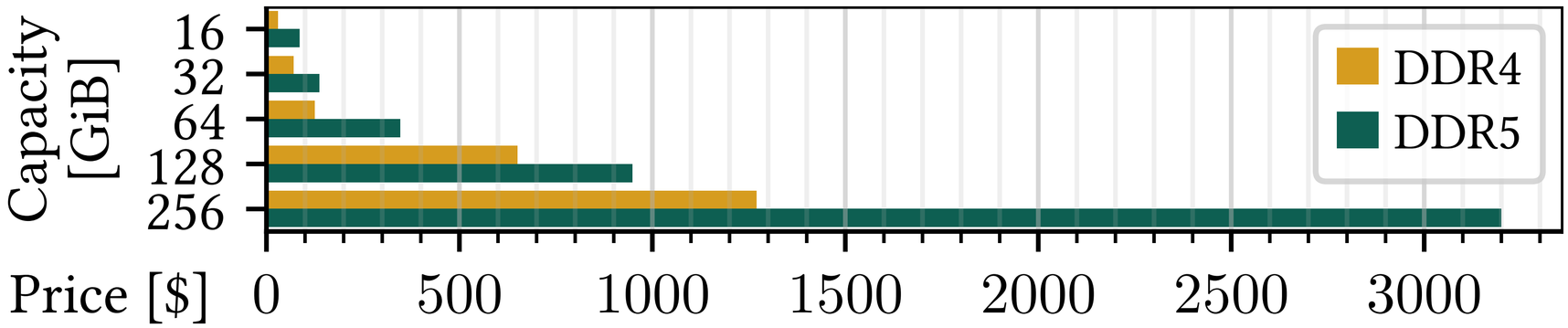


Price [$] per GiB	
DDR4	DDR5
1.9	5.3
2.2	4.3
2.0	5.4
5.1	**7.4**
5.0	**12.5**

We collected prices of NEMIX DDR4/DDR5 288-PIN RDIMMs on NewEgg.com in March 2025.

Economic Viability

Calculate expenses of CPUs and memory DIMMs.

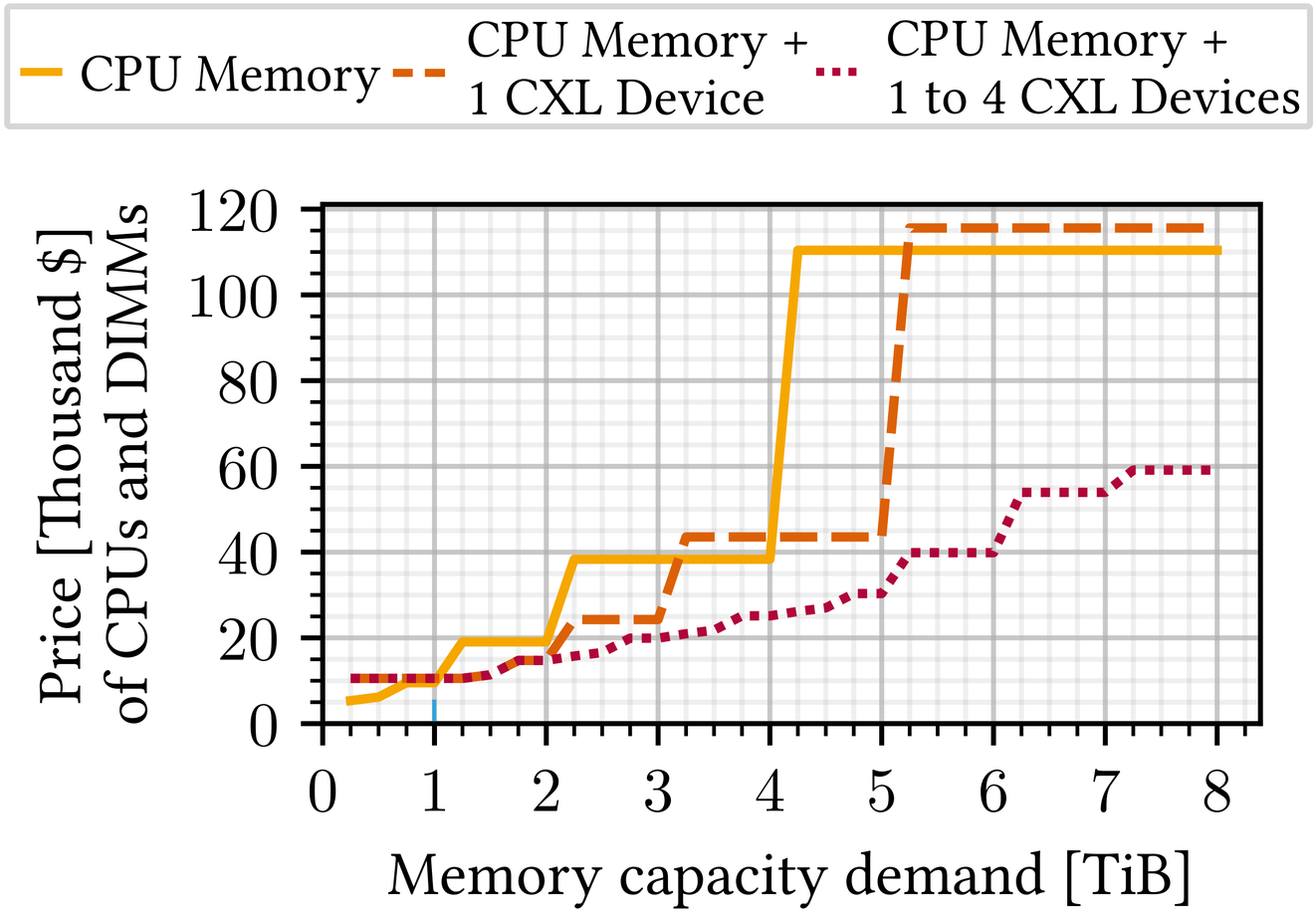


Price [$] per GiB	
DDR4	DDR5
1.9	5.3
2.2	4.3
2.0	5.4
5.1	**7.4**
5.0	**12.5**

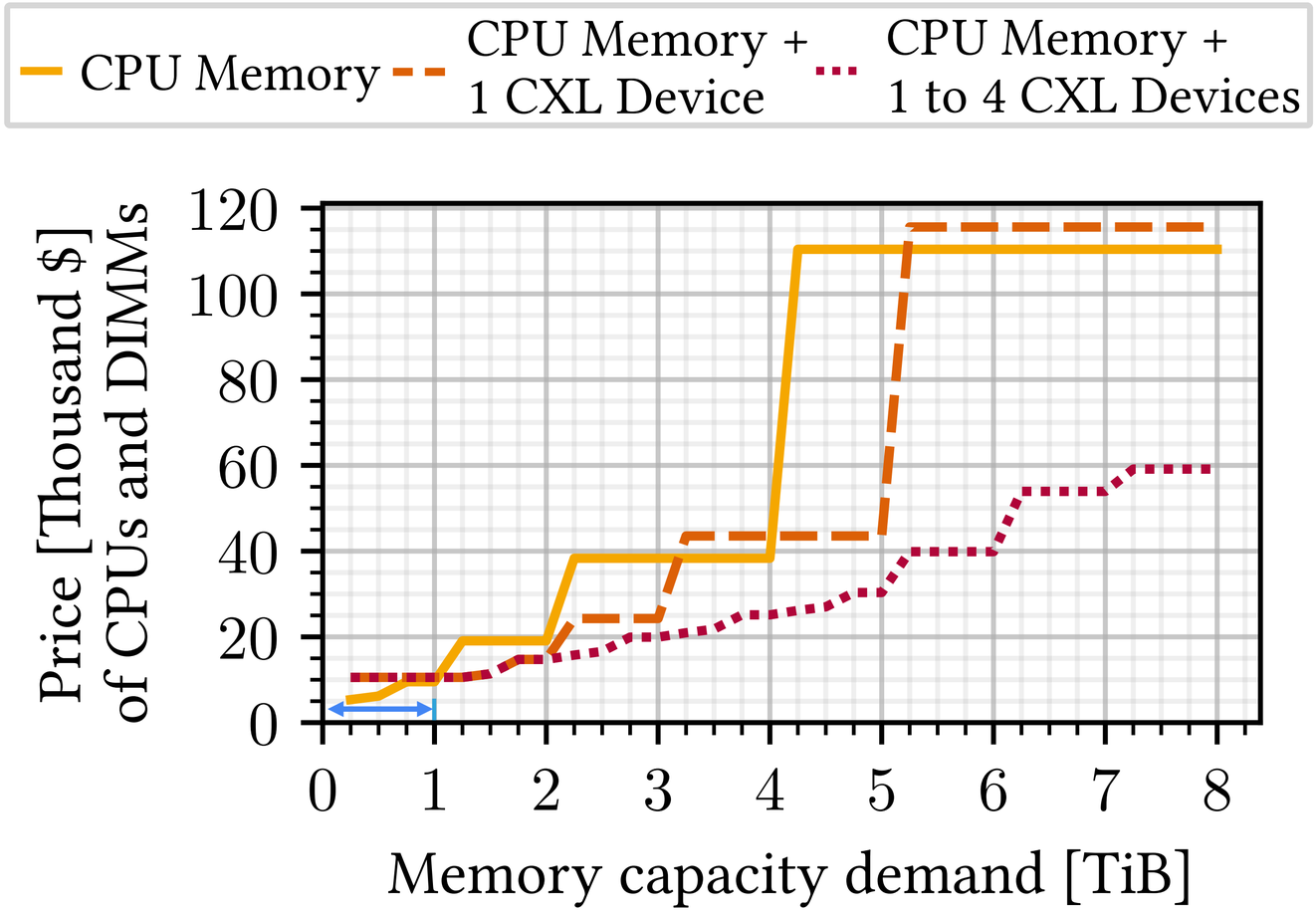
→ Avoid 128 GiB / 256 GiB DIMMs to minimize memory expenses.

We collected prices of NEMIX DDR4/DDR5 288-PIN RDIMMs on NewEgg.com in March 2025.

Economic Viability



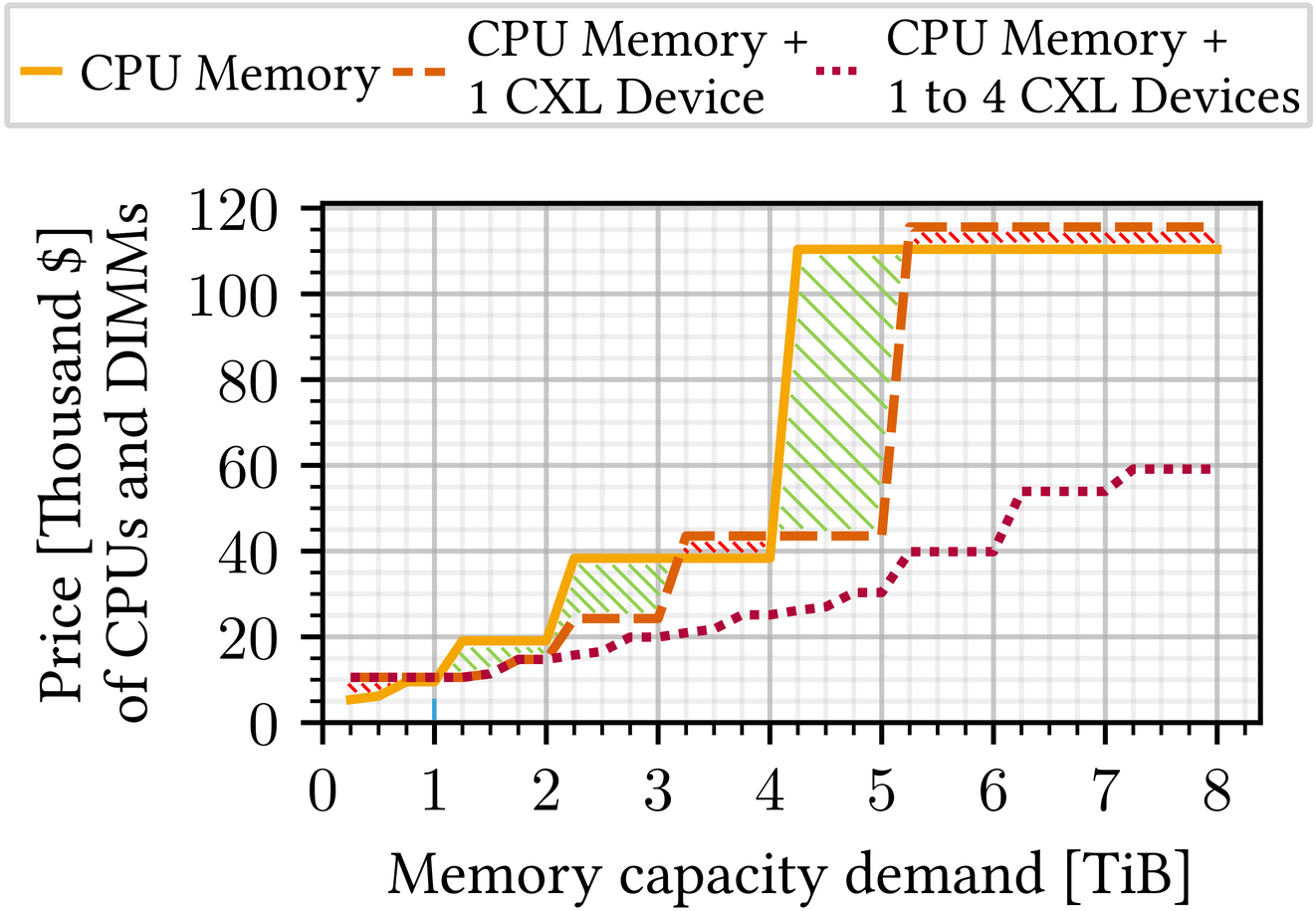
Economic Viability



Observations

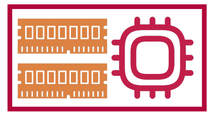
CPU memory cheapest for demand (D) ≤ 1 TiB.

Economic Viability

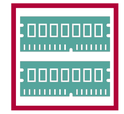


Observations

CPU memory cheapest for demand (D) ≤ 1 TiB.

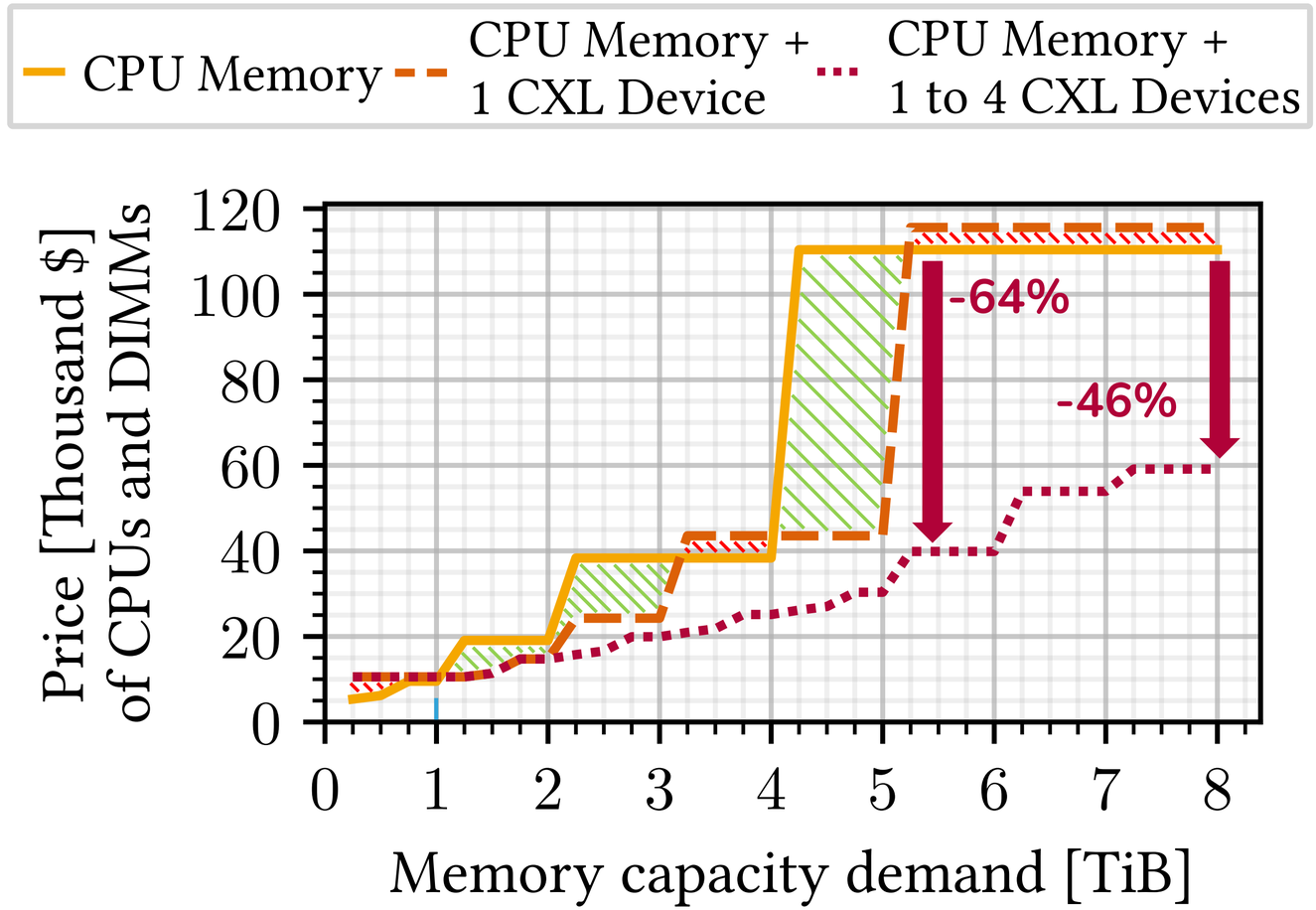


Up to 61% cost reduction for 4 TiB < D ≤ 5 TiB.



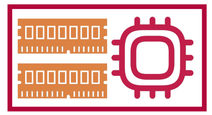
+ 1 device

Economic Viability

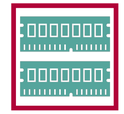


Observations

CPU memory cheapest for demand (D) ≤ 1 TiB.

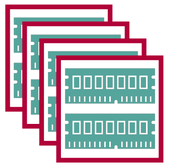


Up to 61% cost reduction for 4 TiB < D ≤ 5 TiB.



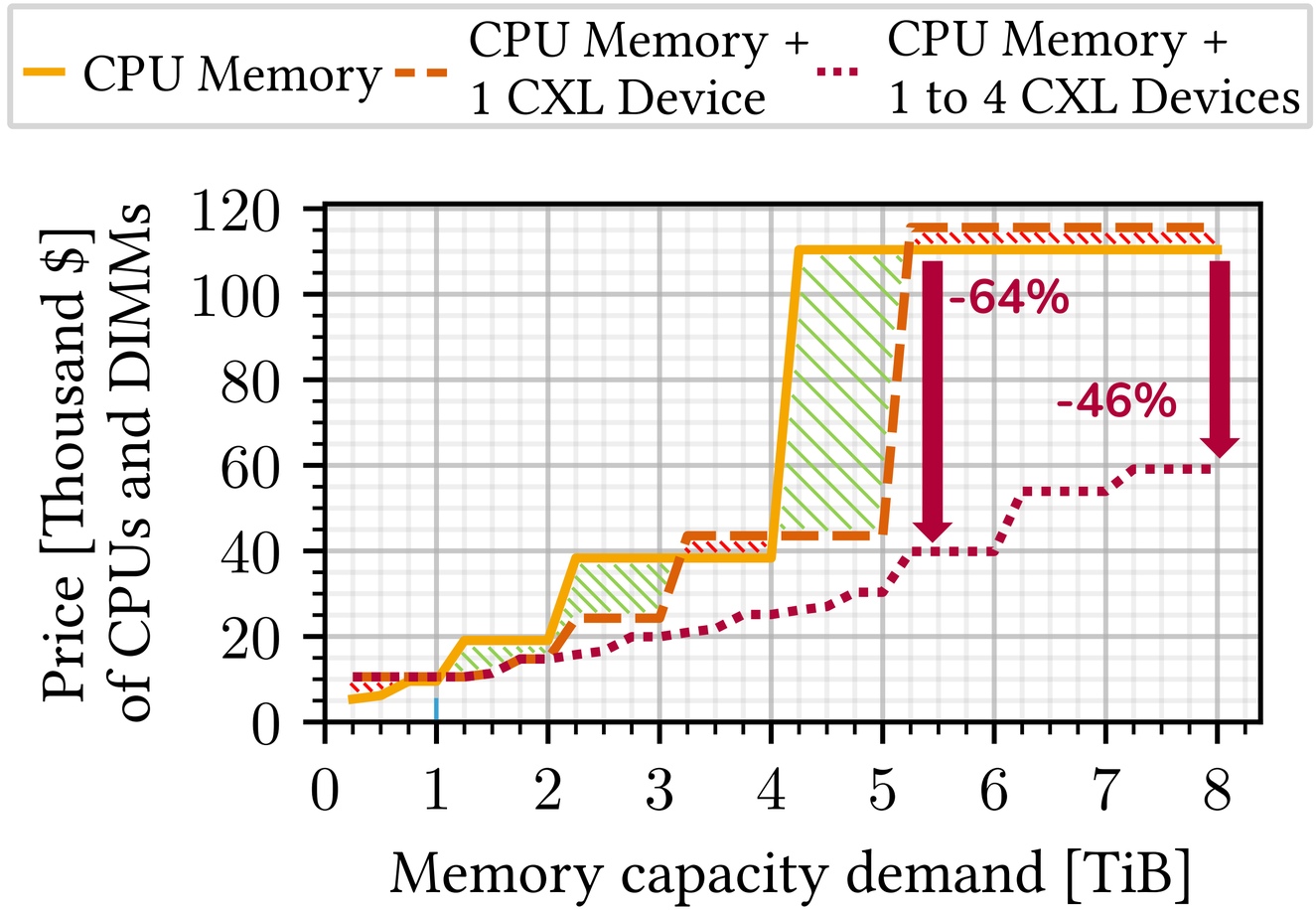
+ 1 device

Up to 64% cost reduction for D > 5 TiB.



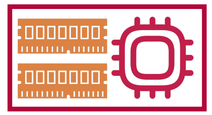
+ 1 to 4 devices

Economic Viability

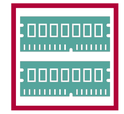


Observations

CPU memory cheapest for demand (D) ≤ 1 TiB.

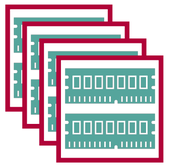


Up to 61% cost reduction for 4 TiB < D ≤ 5 TiB.



+ 1 device

Up to 64% cost reduction for D > 5 TiB.



+ 1 to 4 devices

Reusing DIMMs of old servers further reduces cost of CXL setups.

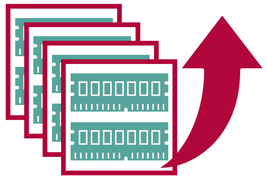
Conclusion

Conclusion

CXL memory can reduce cost while providing high OLAP throughput with hot data in CPU memory.

Conclusion

CXL memory can reduce cost while providing high OLAP throughput with hot data in CPU memory.

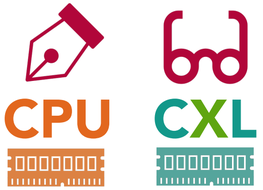


Bandwidth-bound workloads benefit from CXL memory bandwidth of multiple devices.

Conclusion

CXL memory can reduce cost while providing high OLAP throughput with hot data in CPU memory.

Bandwidth-bound workloads benefit from CXL memory bandwidth of multiple devices.

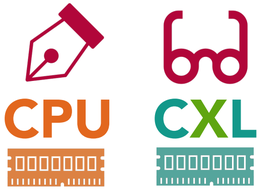


Write-throughput limited workloads can benefit from writes to CPU memory and reads from CXL memory.

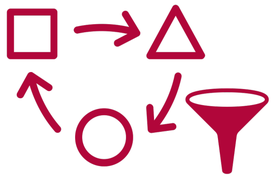
Conclusion

CXL memory can reduce cost while providing high OLAP throughput with hot data in CPU memory.

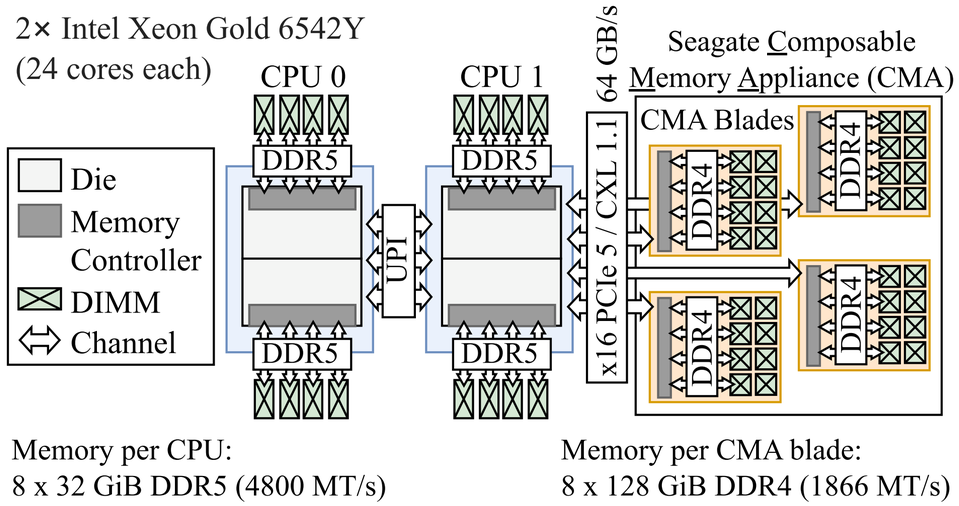
Bandwidth-bound workloads benefit from CXL memory bandwidth of multiple devices.

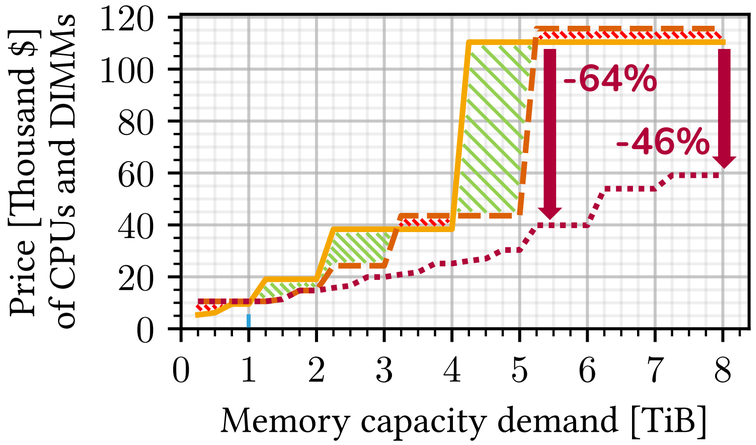


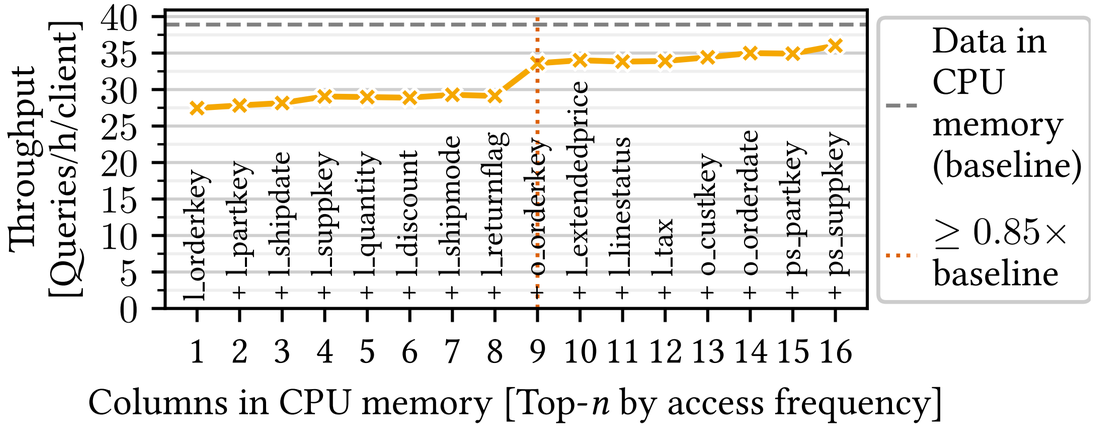
Write-throughput limited workloads can benefit from writes to CPU memory and reads from CXL memory.



Changing data structures according to their memory bottleneck improves performance.







- CXL memory can reduce cost while providing high OLAP throughput with hot data in CPU memory.
- Bandwidth-bound workloads benefit from CXL memory bandwidth of multiple devices.
- Write-throughput limited workloads can benefit from writes to CPU memory and reads from CXL memory.
- Changing data structures according to their memory bottleneck improves performance.

Contact

marcel.weisgut@hpi.de

weisgut.com

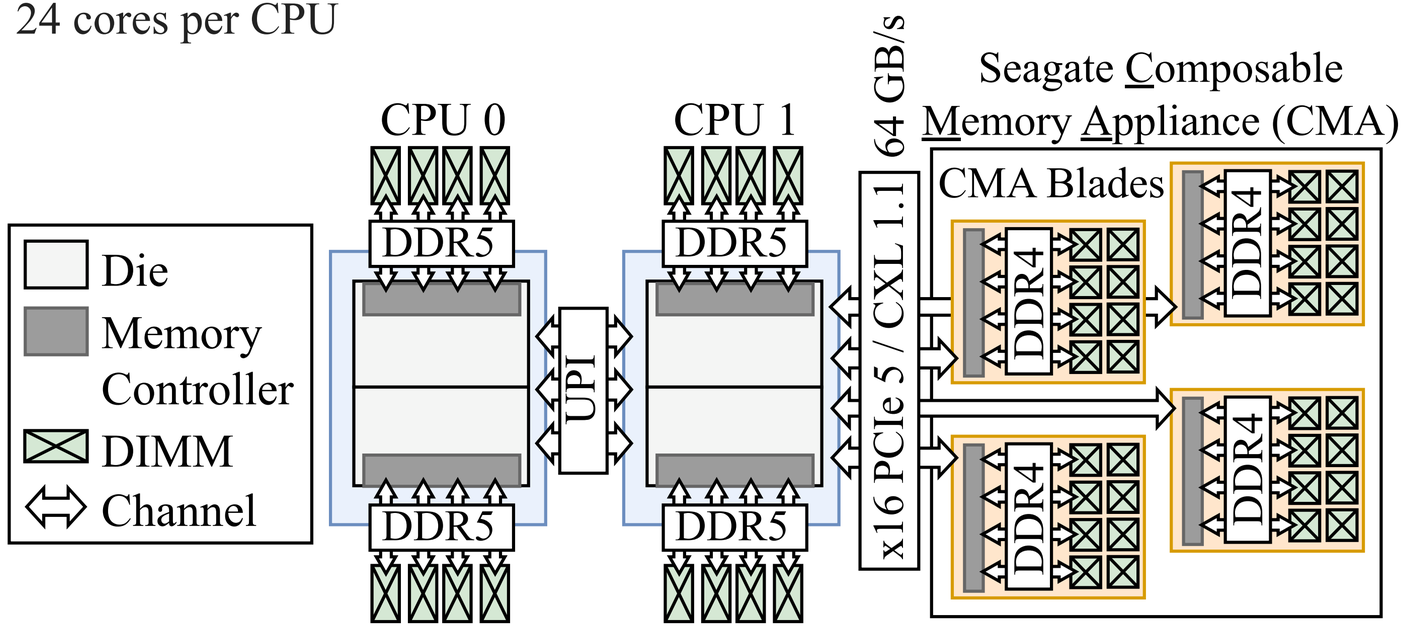
Data Engineering Systems Group @ HPI

hpi.de/rabl

BACKUP

Hardware Setup

2× Intel Xeon Gold 6542Y (Emerald Rapids)
24 cores per CPU

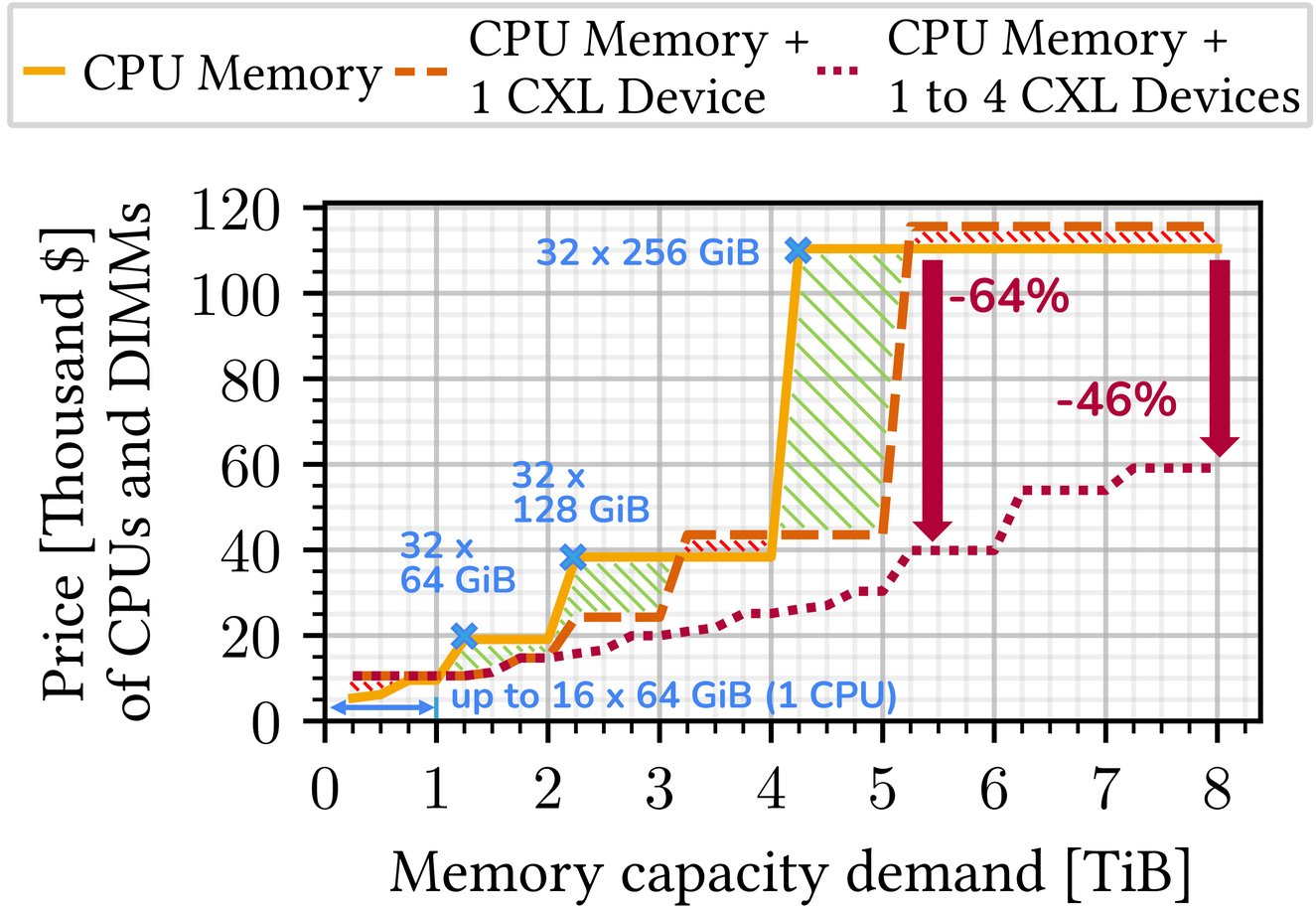


Memory per CPU:
8 x 32 GiB DDR5 (4800 MT/s)

Memory per CMA blade:
8 x 128 GiB DDR4 (1866 MT/s)

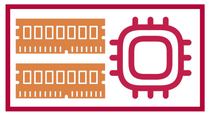
Server	Supermicro SYS-741GE-TNRT
CPUs	2 Intel 5th Gen Xeon Scalable Gold 6542Y with 24 cores
Caches	L1i: 32 KiB, L1d: 48 KiB, L2: 2 MiB, L3: 60 MiB
Memory	8 32 GB DDR5 with a speed of 4800 MT/s
OS	Ubuntu 24.04, Kernel 6.13

Economic Viability

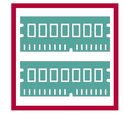


Observations

CPU memory cheapest for demand (D) ≤ 1 TiB.

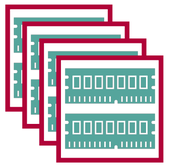


Up to 61% cost reduction for 4 TiB < D ≤ 5 TiB.



+ 1 device

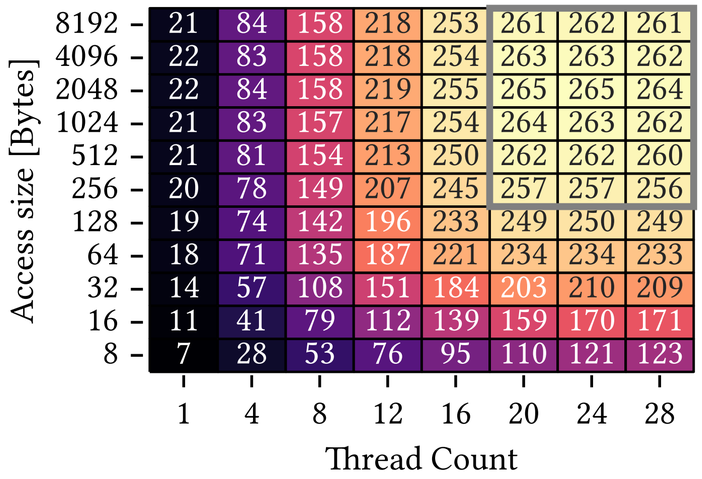
Up to 64% cost reduction for D > 5 TiB.



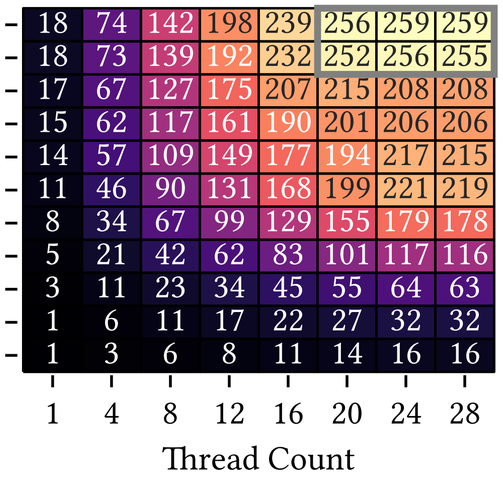
+ 1 to 4 devices

Reusing DIMMs of old servers further reduces cost of CXL setups.

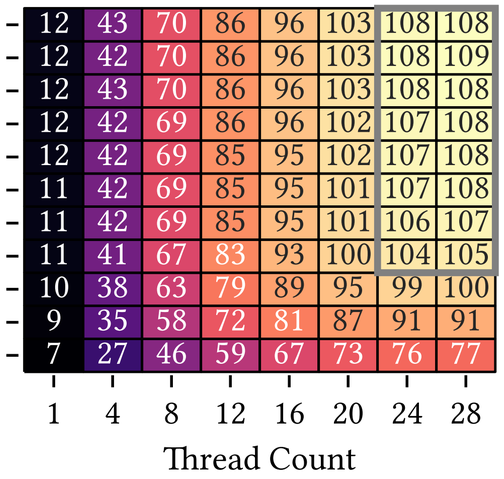
Throughput – CPU Memory & CXL Device Memory



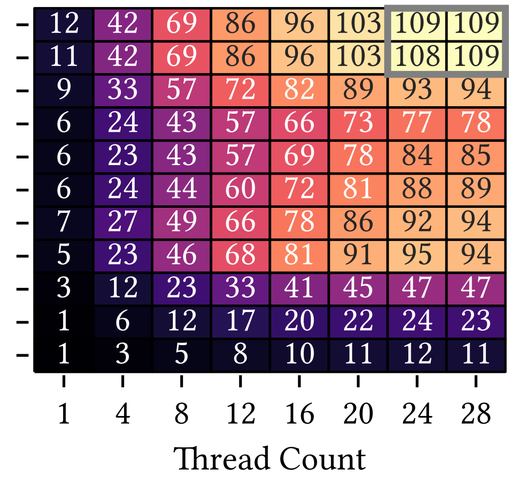
a) Sequential Reads CPU)



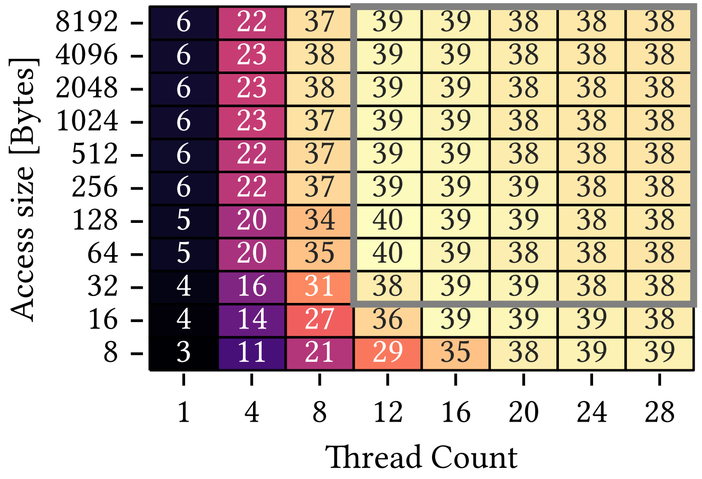
b) Random Reads CPU)



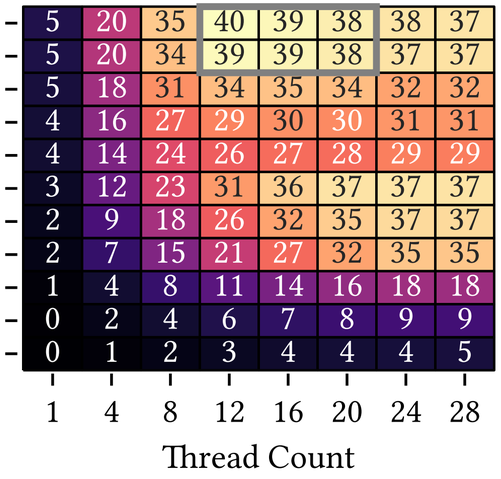
c) Sequential Writes CPU)



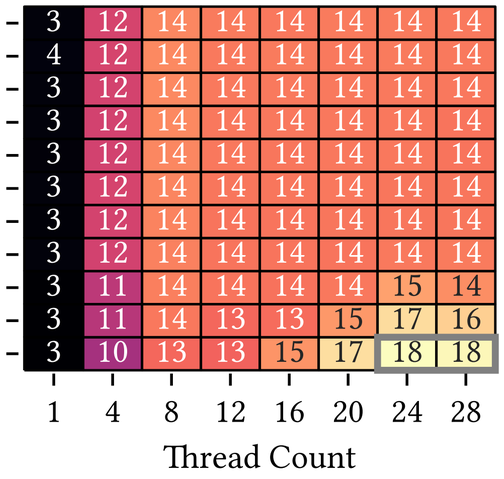
d) Random Writes CPU)



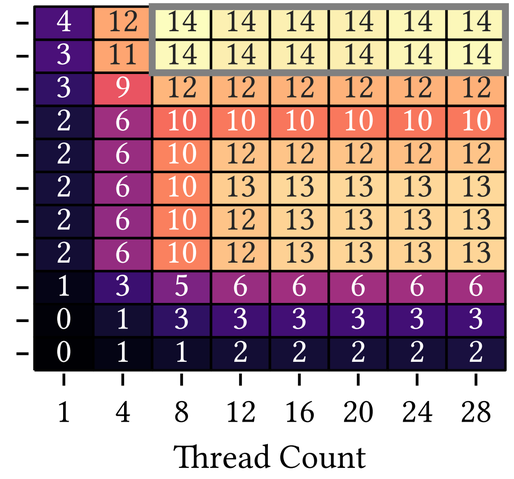
e) Sequential Reads CXL)



f) Random Reads CXL)

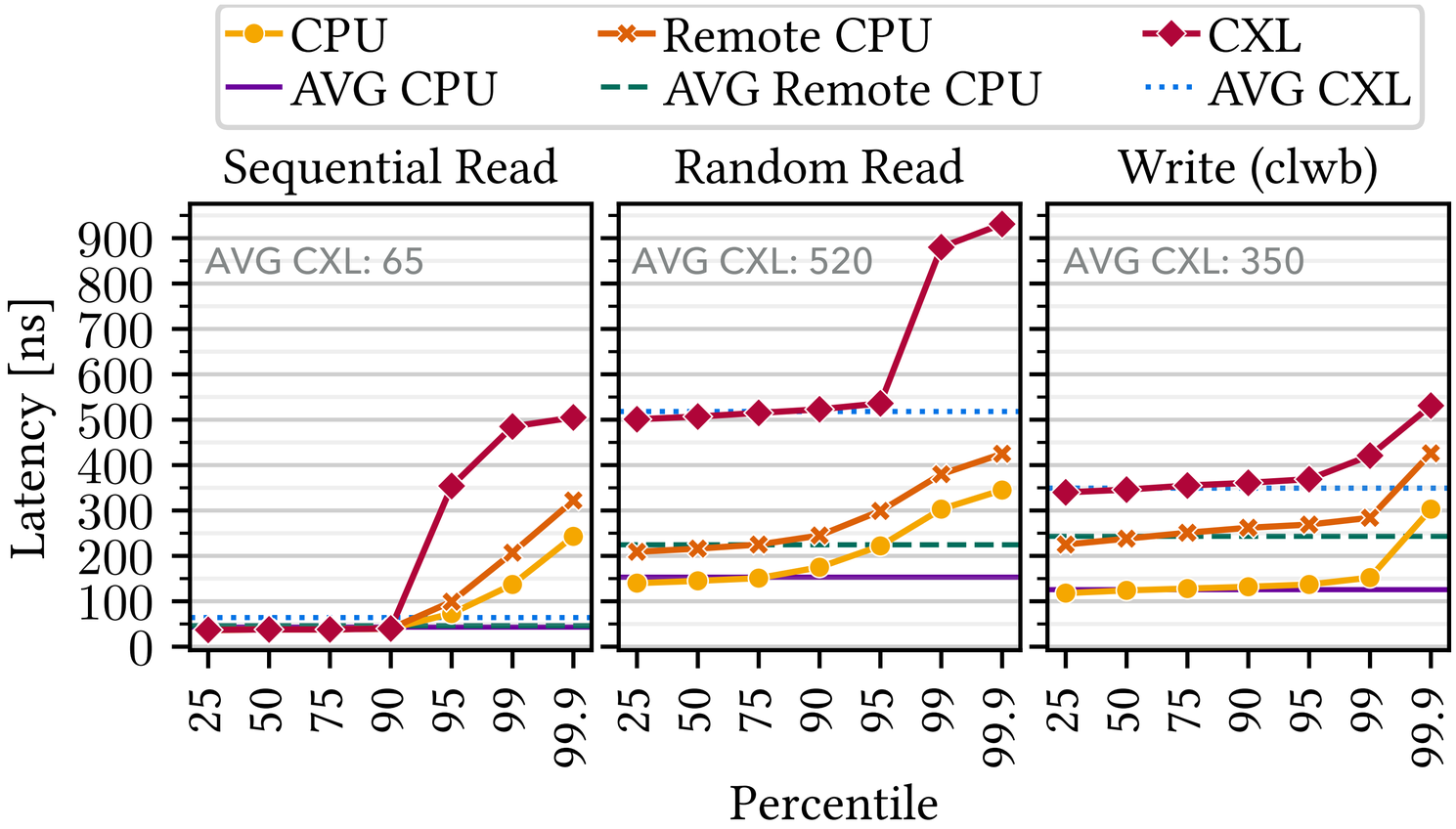


g) Sequential Writes CXL)



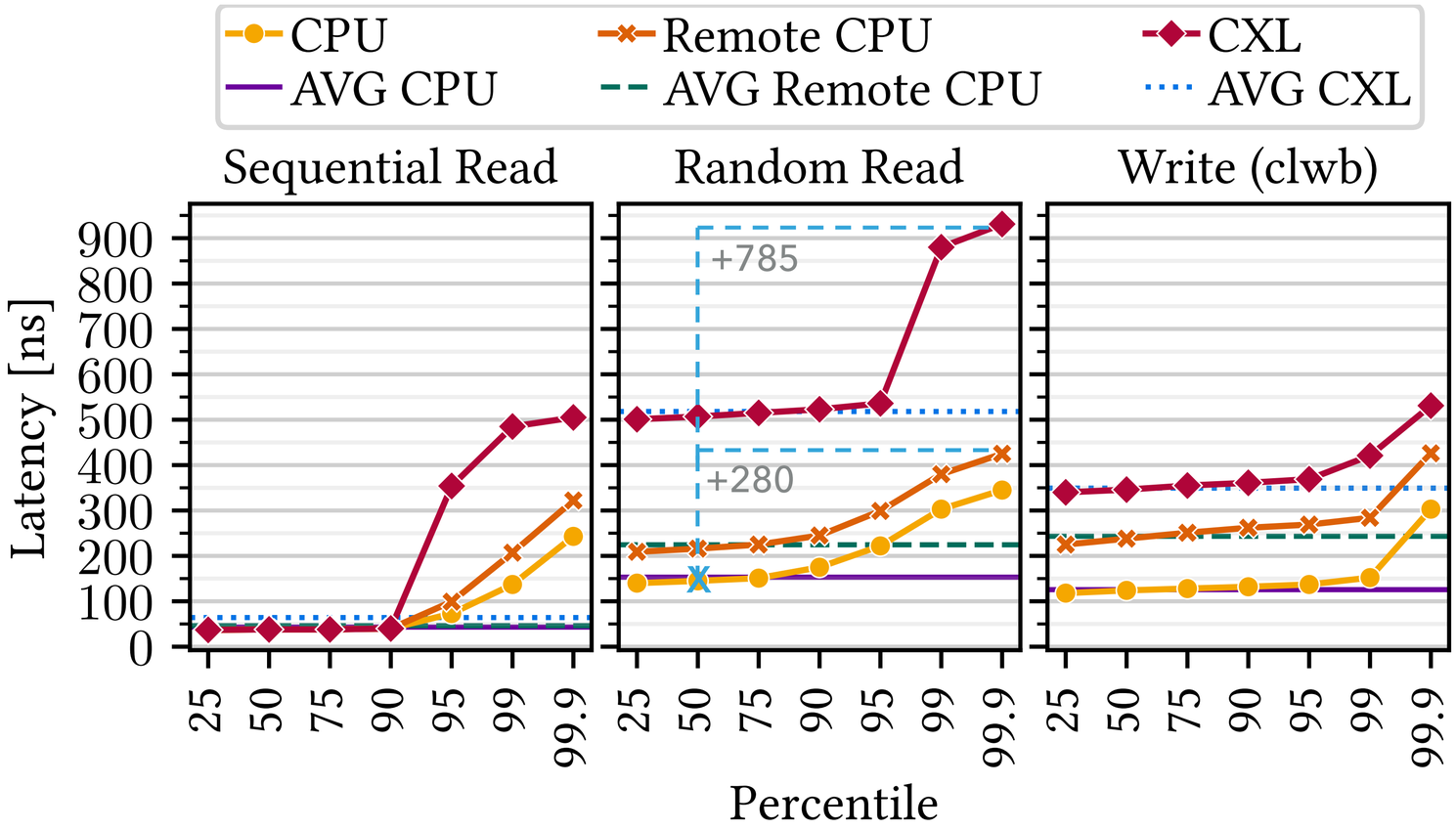
h) Random Writes CXL)

Latency



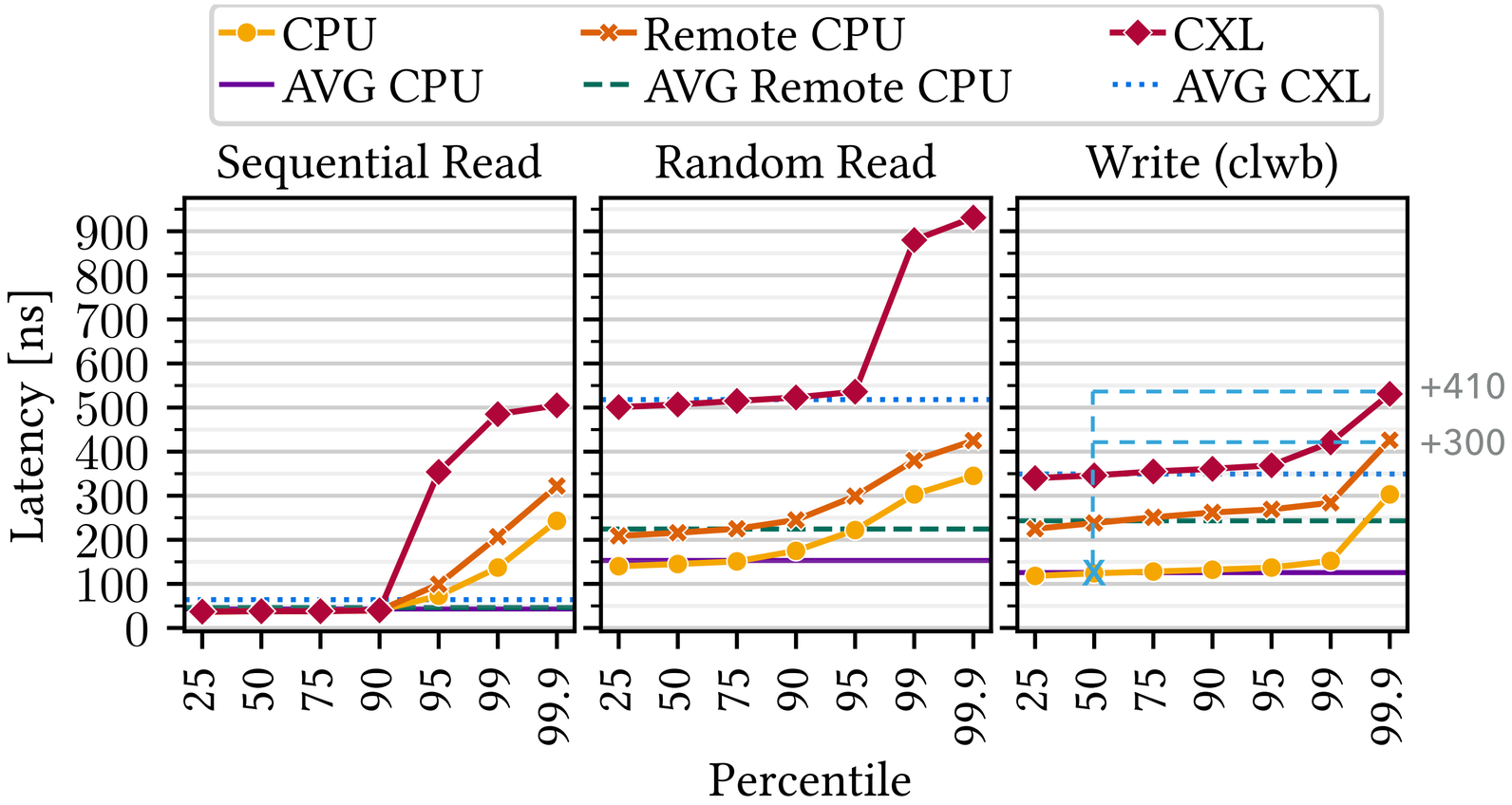
- 8 B reads/writes
- Reads: load + fence
- Writes: first load cache line, then measure store + clwb + fence

Latency



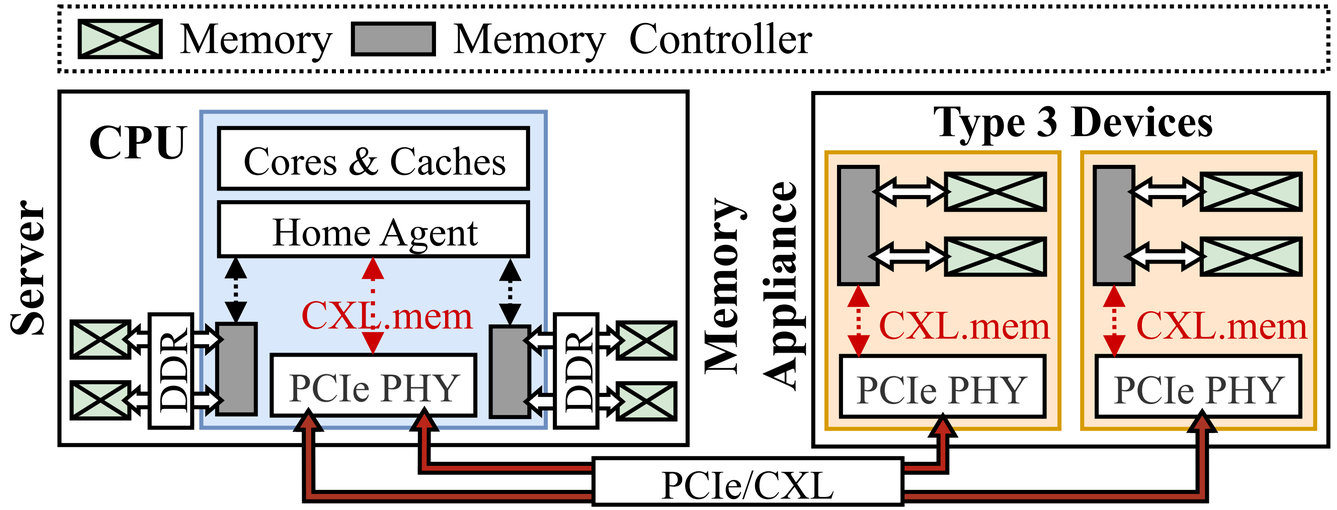
- 8 B reads/writes
- Reads: load + fence
- Writes: first load cache line, then measure store + clwb + fence

Latency

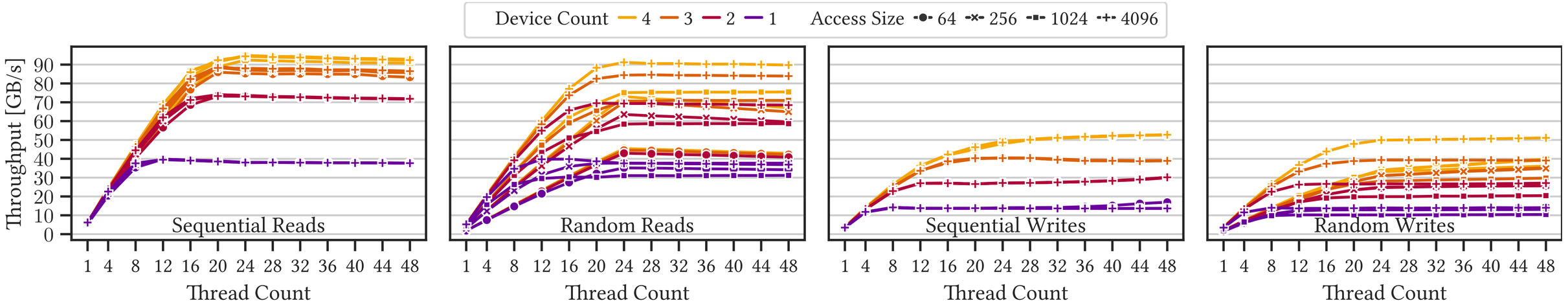


- 8 B reads/writes
- Reads: load + fence
- Writes: first load cache line, then measure store + clwb + fence

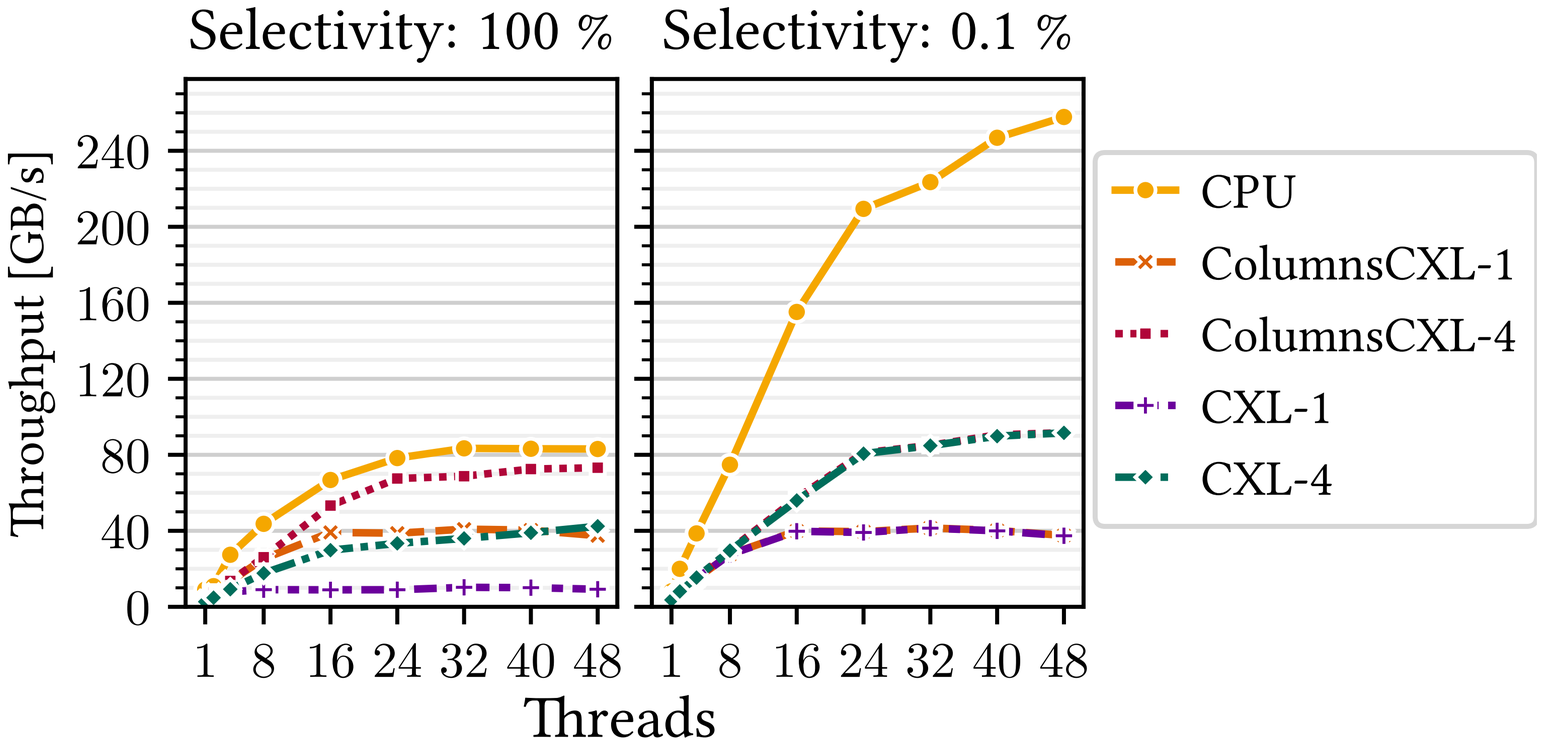
Memory Expansion with Multiple CXL Devices



Throughput – Multiple CXL Devices

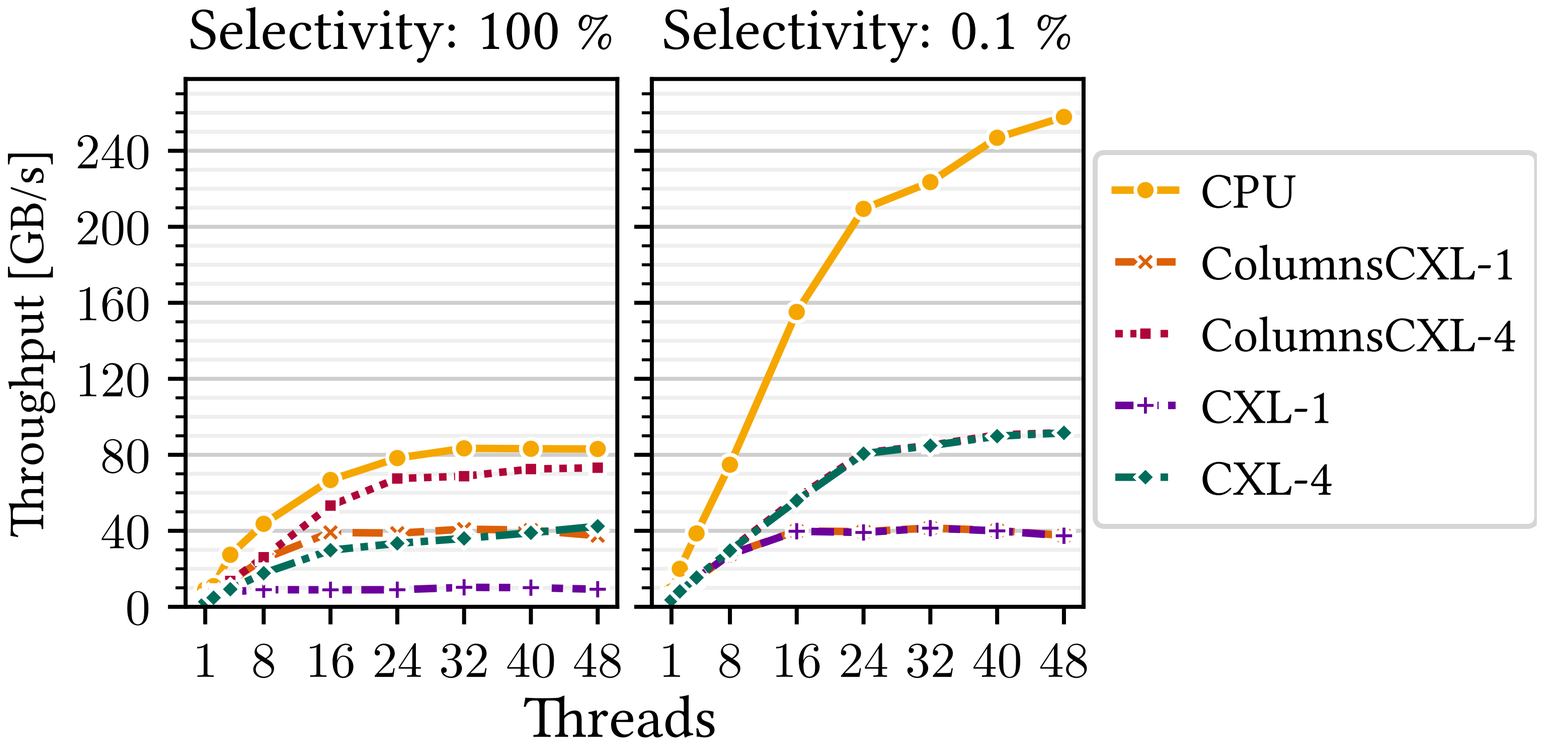


In-Memory SIMD Column Scan



- 4-B unsigned integer (uint) values
- Each thread scans separate column (512 MiB, ~134 M values)
- Scan writes 4-B uint tuple identifiers (TIDs) for matches
- Allocate memory region for each column and each TID list

In-Memory SIMD Column Scan

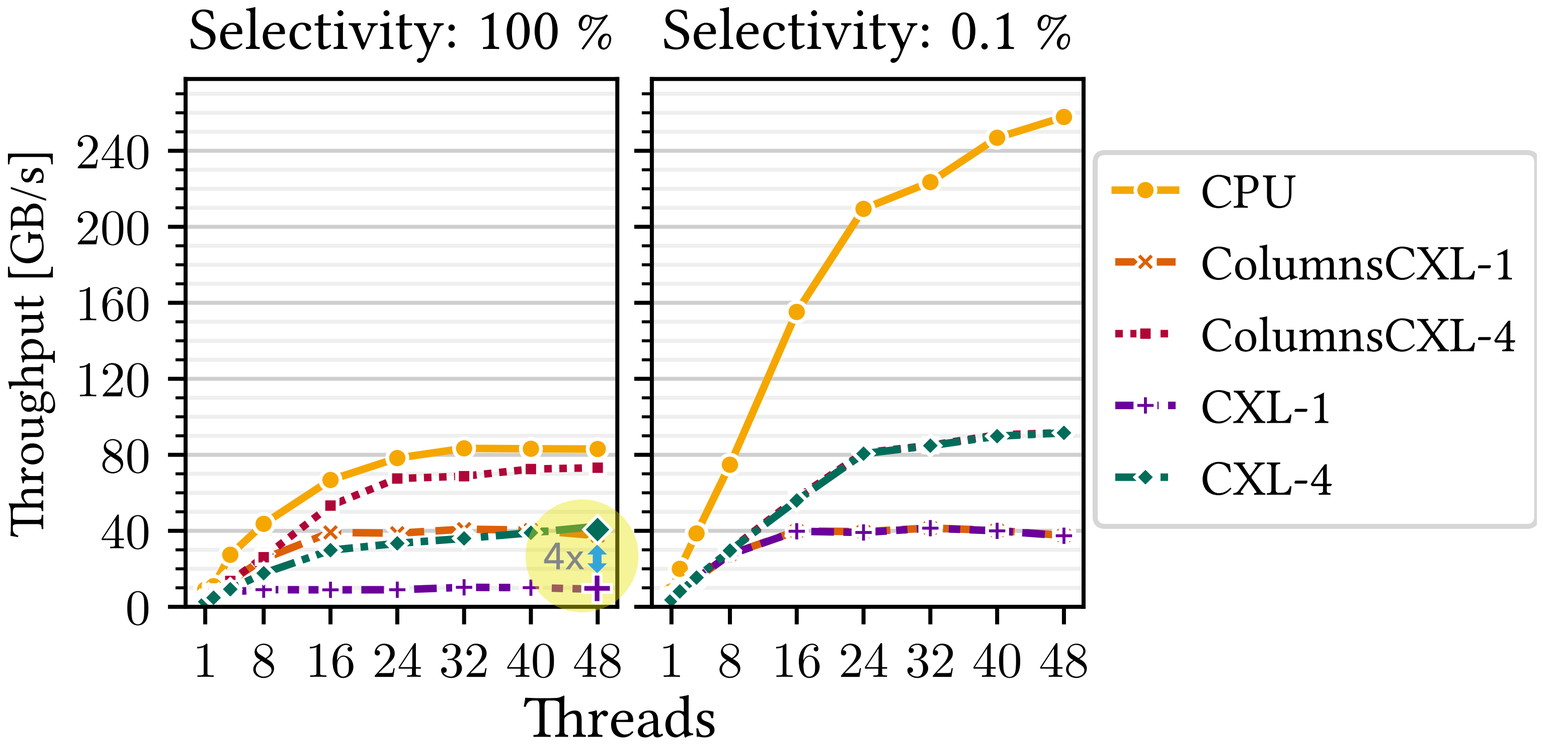


Observations

- Lower selectivity → fewer writes → higher performance
- Write throughput limits scan since it writes a 4-B TID for each read value
- More devices yield higher scan throughput

- 4-B unsigned integer (uint) values
- Each thread scans separate column (512 MiB, ~134 M values)
- Scan writes 4-B uint tuple identifiers (TIDs) for matches
- Allocate memory region for each column and each TID list

In-Memory SIMD Column Scan

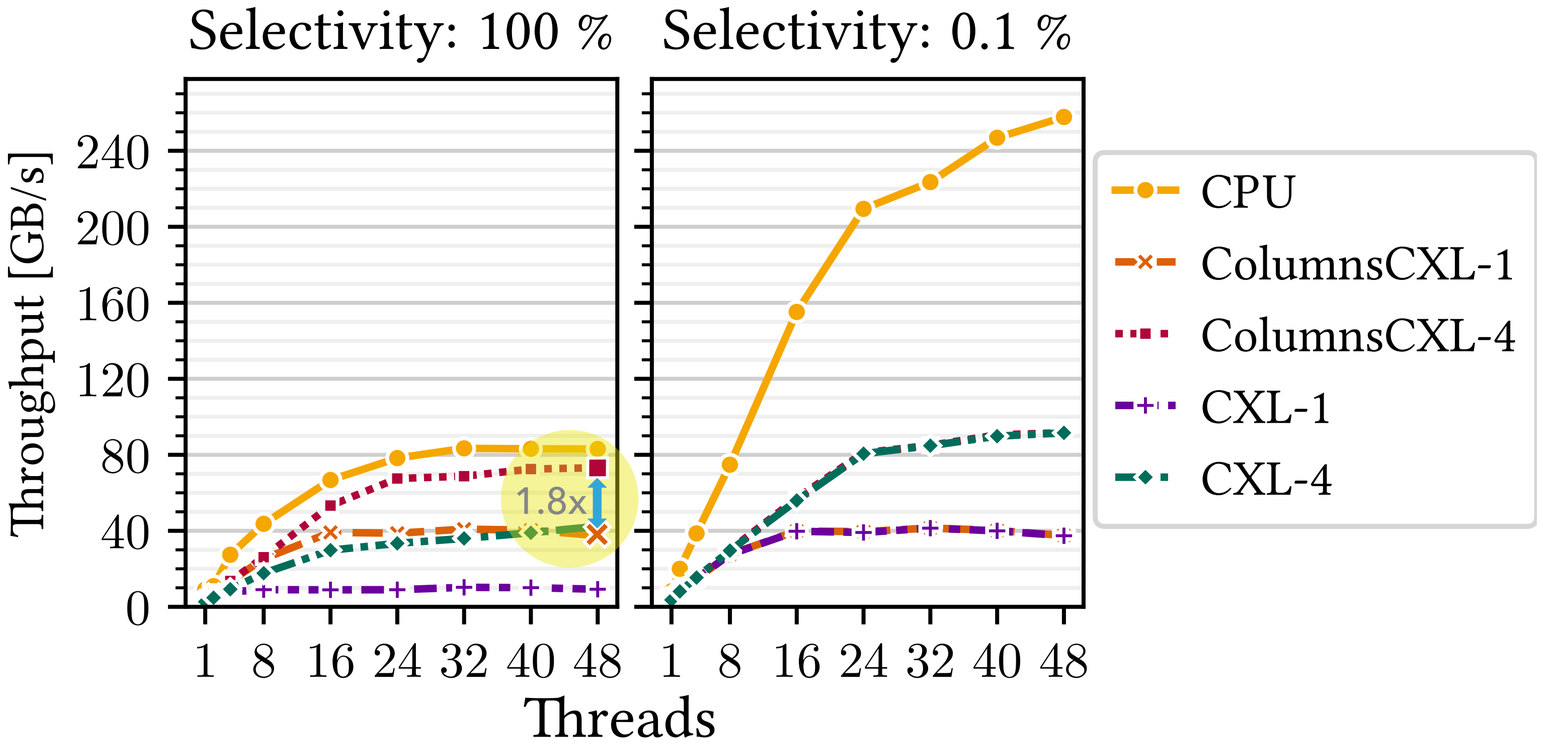


Observations

- Lower selectivity → fewer writes → higher performance
- Write throughput limits scan since it writes a 4-B TID for each read value
- More devices yield higher scan throughput

- 4-B unsigned integer (uint) values
- Each thread scans separate column (512 MiB, ~134 M values)
- Scan writes 4-B uint tuple identifiers (TIDs) for matches
- Allocate memory region for each column and each TID list

In-Memory SIMD Column Scan

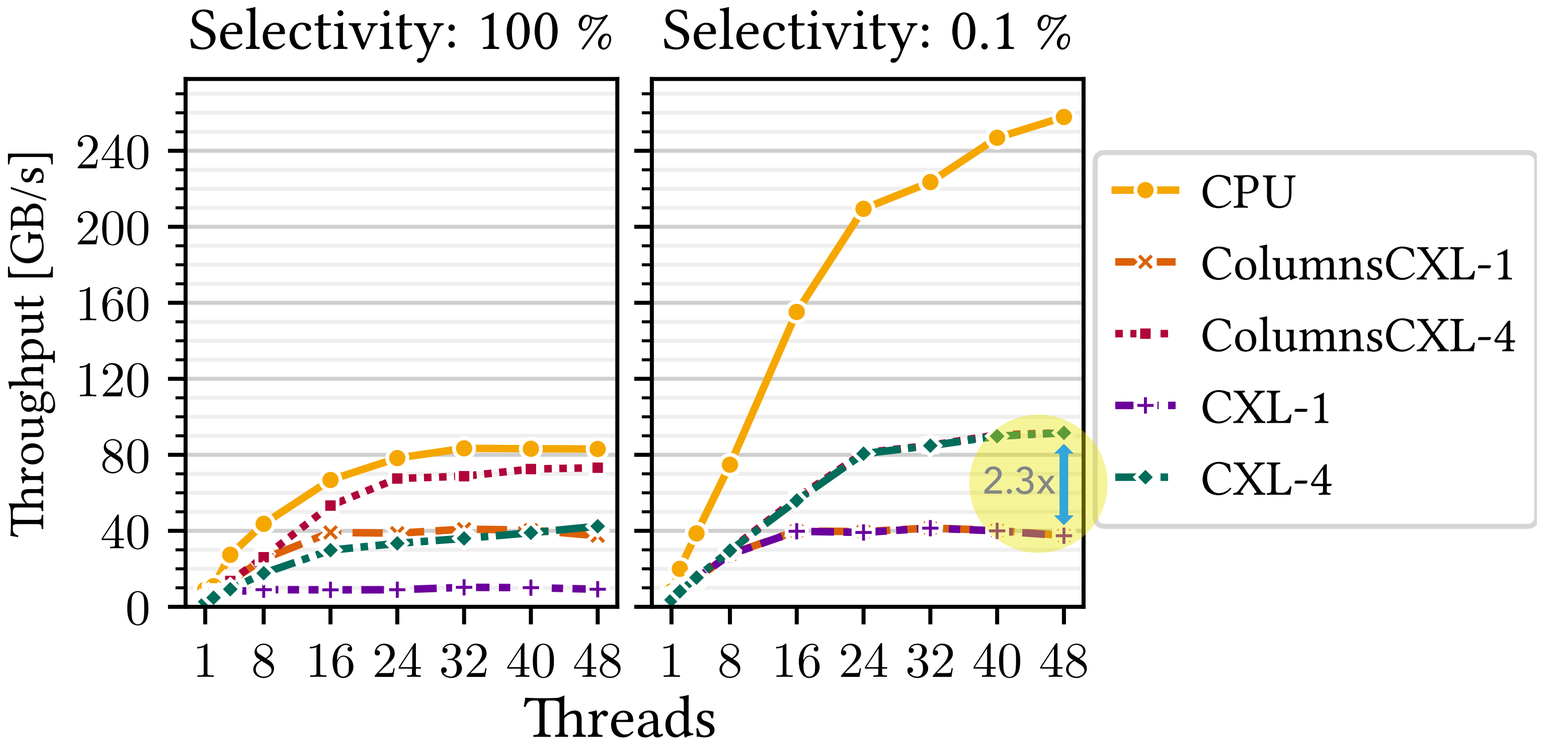


Observations

- Lower selectivity → fewer writes → higher performance
- Write throughput limits scan since it writes a 4-B TID for each read value
- More devices yield higher scan throughput

- 4-B unsigned integer (uint) values
- Each thread scans separate column (512 MiB, ~134 M values)
- Scan writes 4-B uint tuple identifiers (TIDs) for matches
- Allocate memory region for each column and each TID list

In-Memory SIMD Column Scan

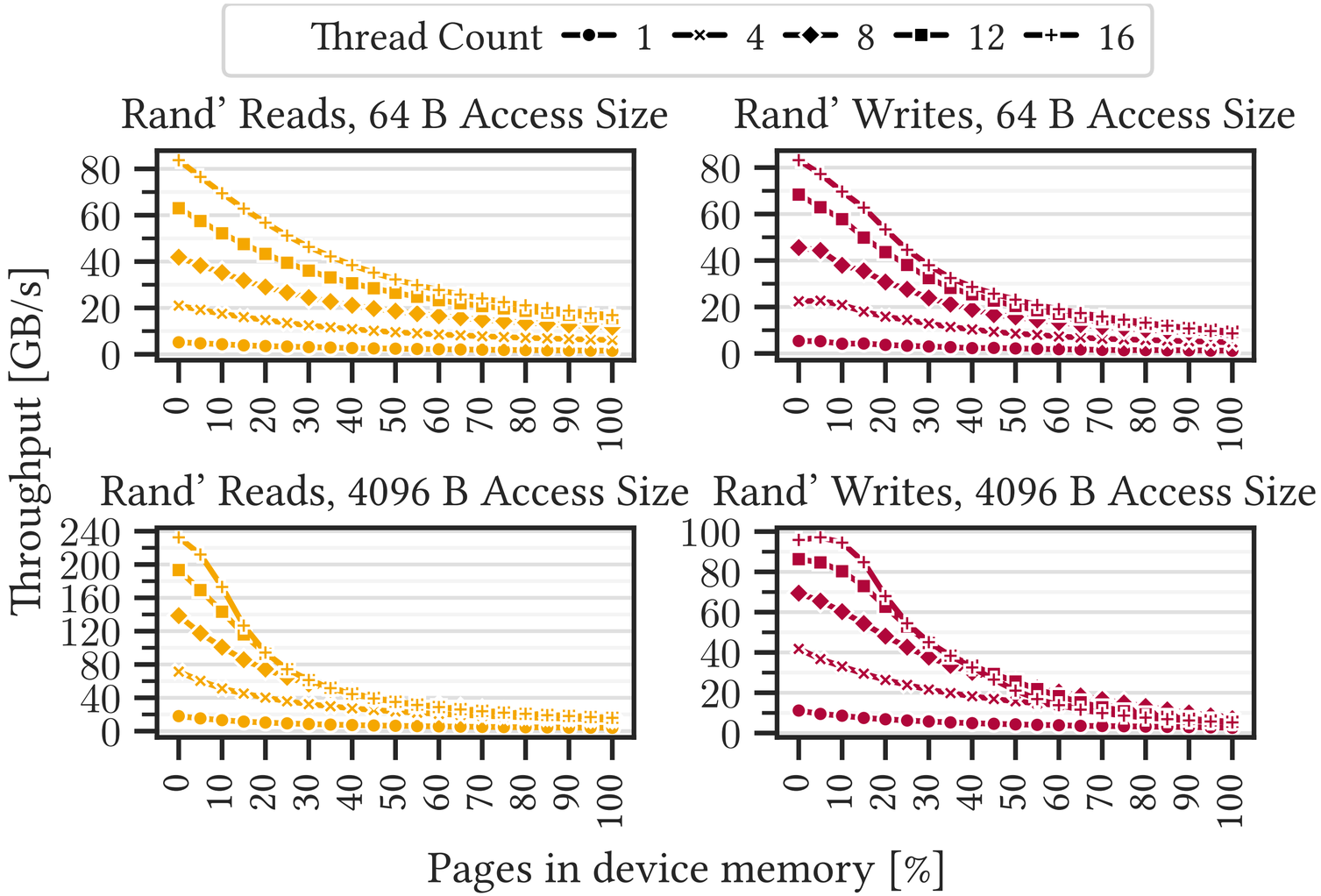


- 4-B unsigned integer (uint) values
- Each thread scans separate column (512 MiB, ~134 M values)
- Scan writes 4-B uint tuple identifiers (TIDs) for matches
- Allocate memory region for each column and each TID list

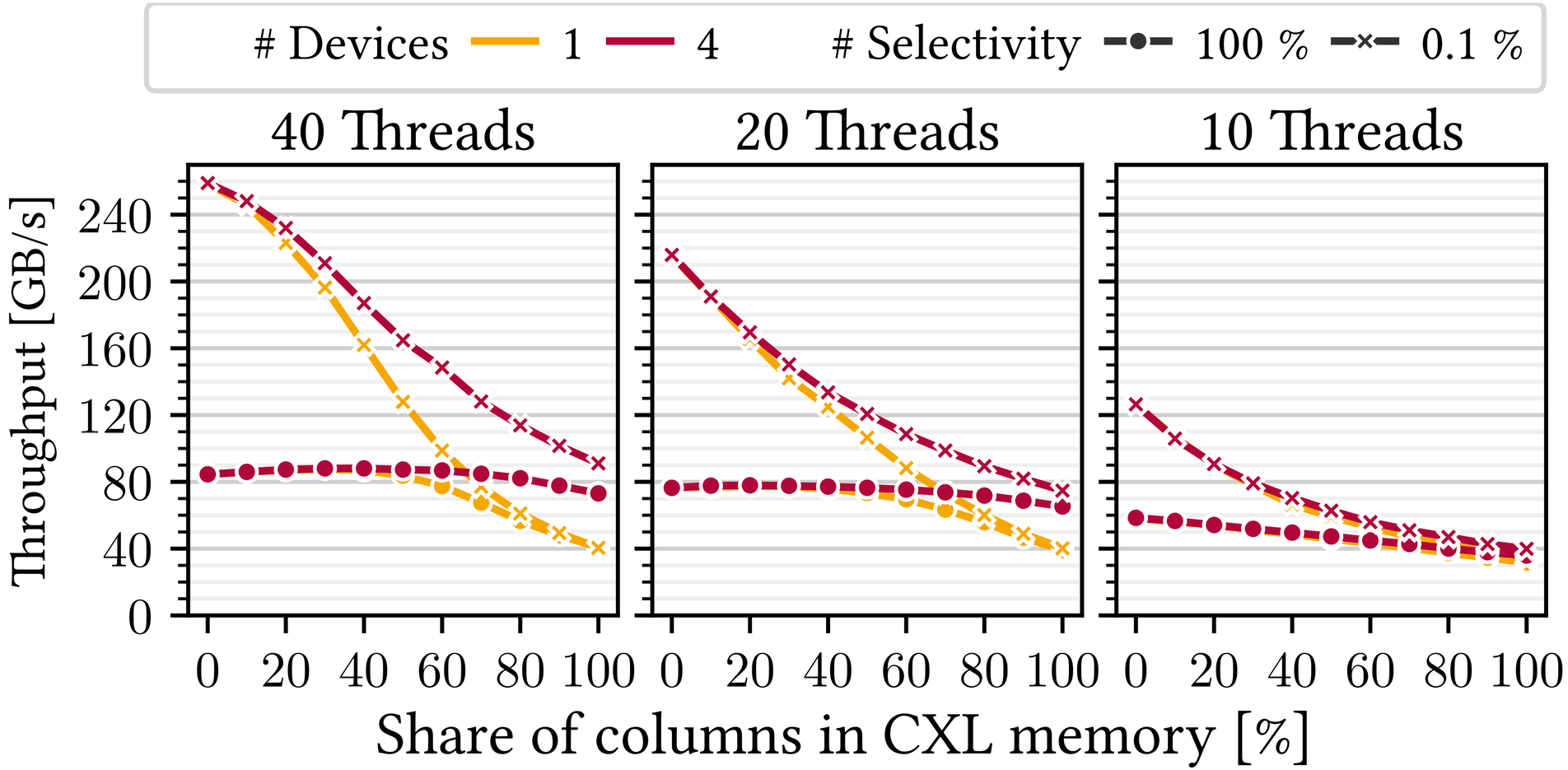
Observations

- Lower selectivity → fewer writes → higher performance
- Write throughput limits scan since it writes a 4-B TID for each read value
- More devices yield higher scan throughput

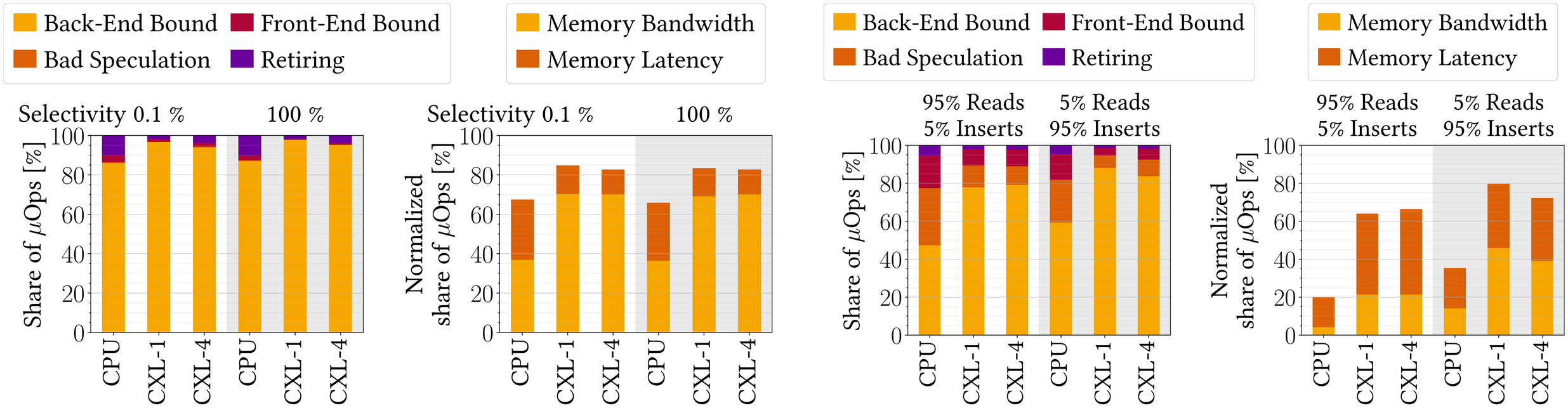
Impact of Storing Pages in CXL Device Memory



Scans with a Share of Columns in CXL Memory

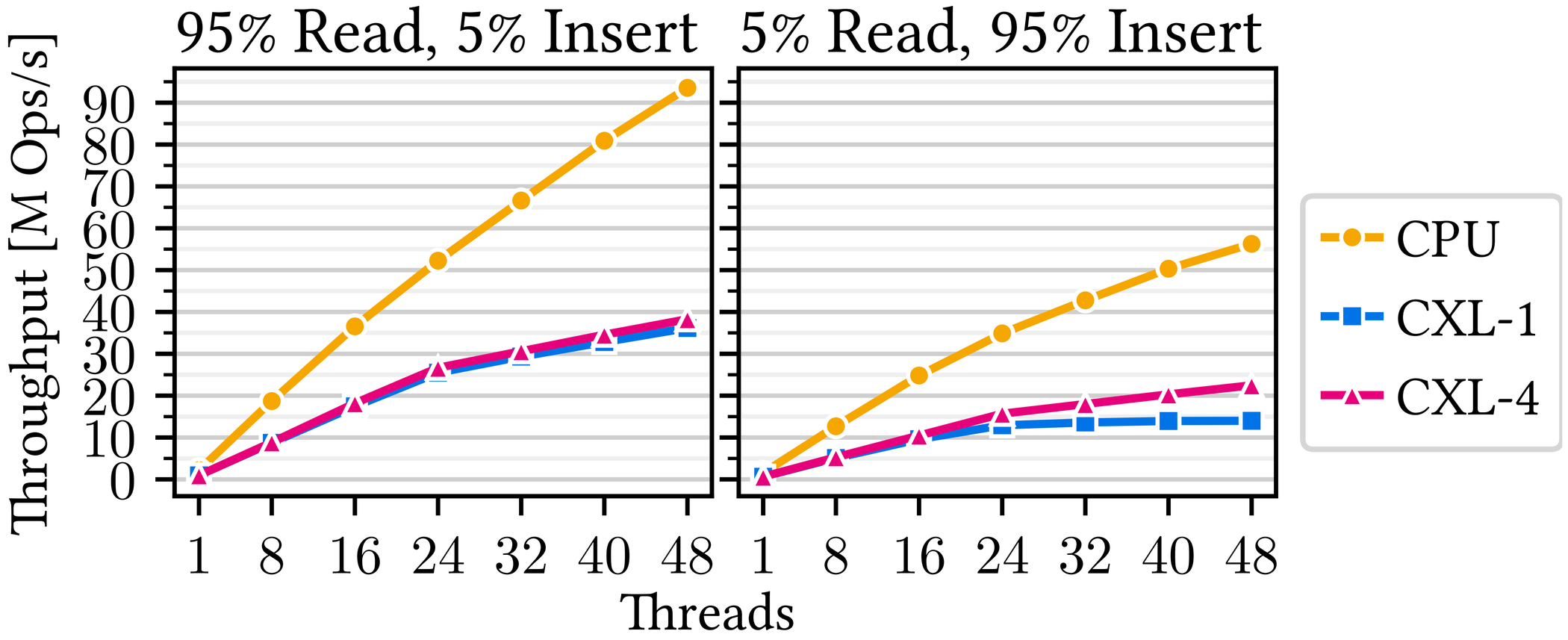


Top-Down Microarchitecture Analysis: Scans & B+-Tree Workloads

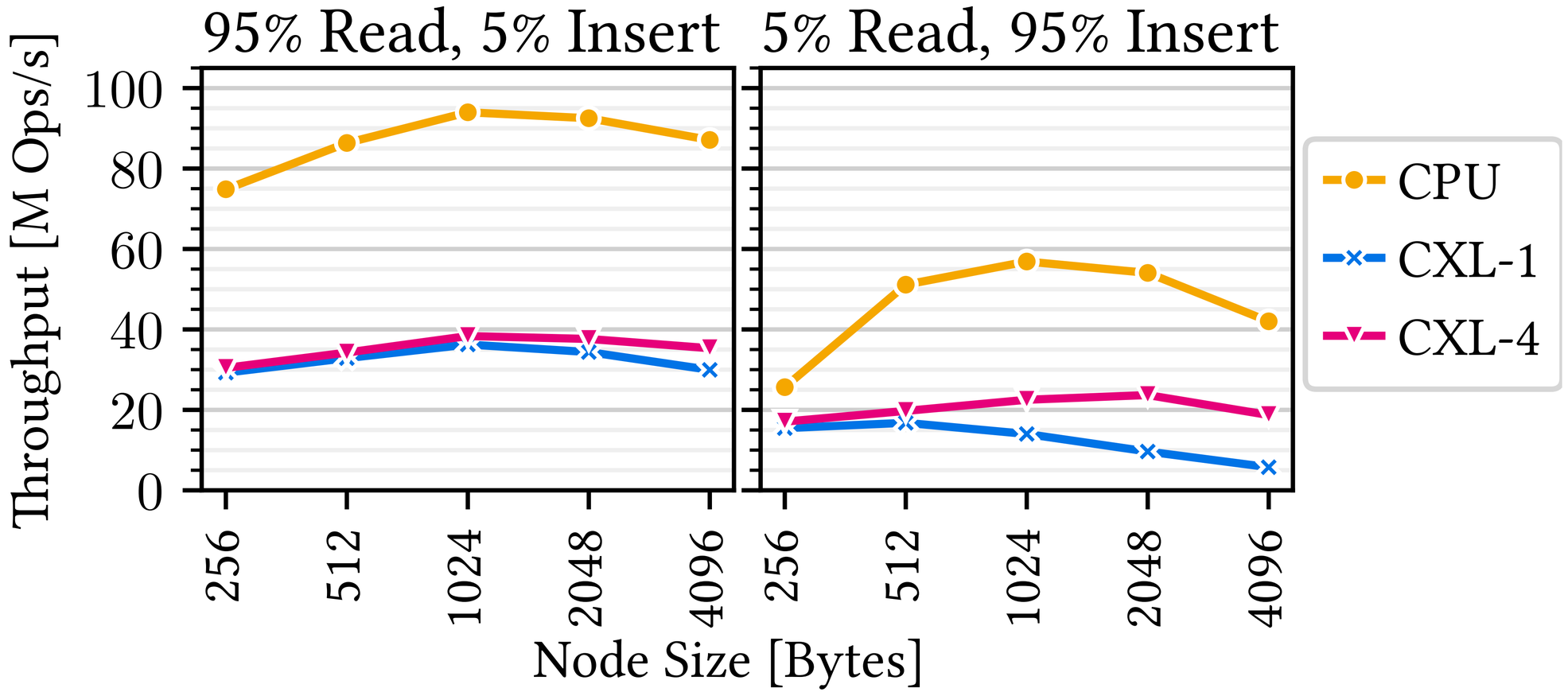


Performance breakdown with 40 threads and all data on one device (CXL-1), four devices (CXL-4), or CPU memory (CPU) with normalized bandwidth- and latency-bound shares.

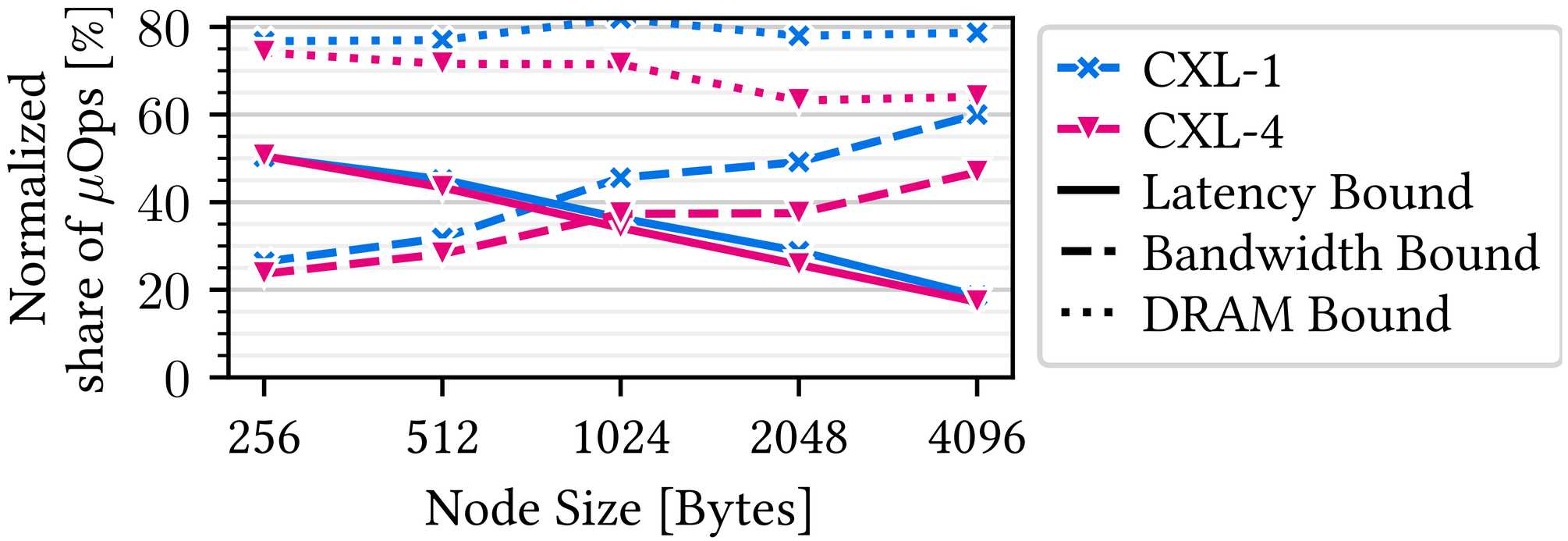
BTreeOLC Performance with Different Data Placement Configurations (1024 B Nodes)



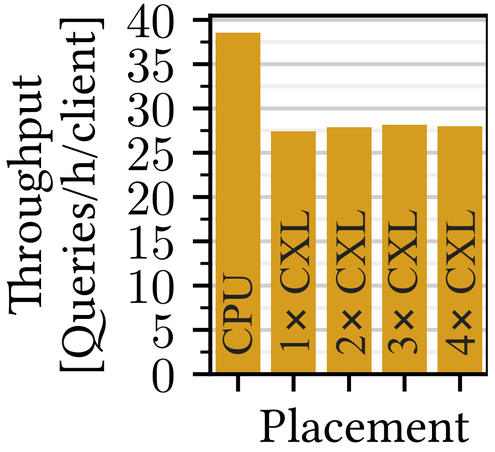
BTreeOLC Performance with Different Data Placements and Node Sizes for 48 Threads



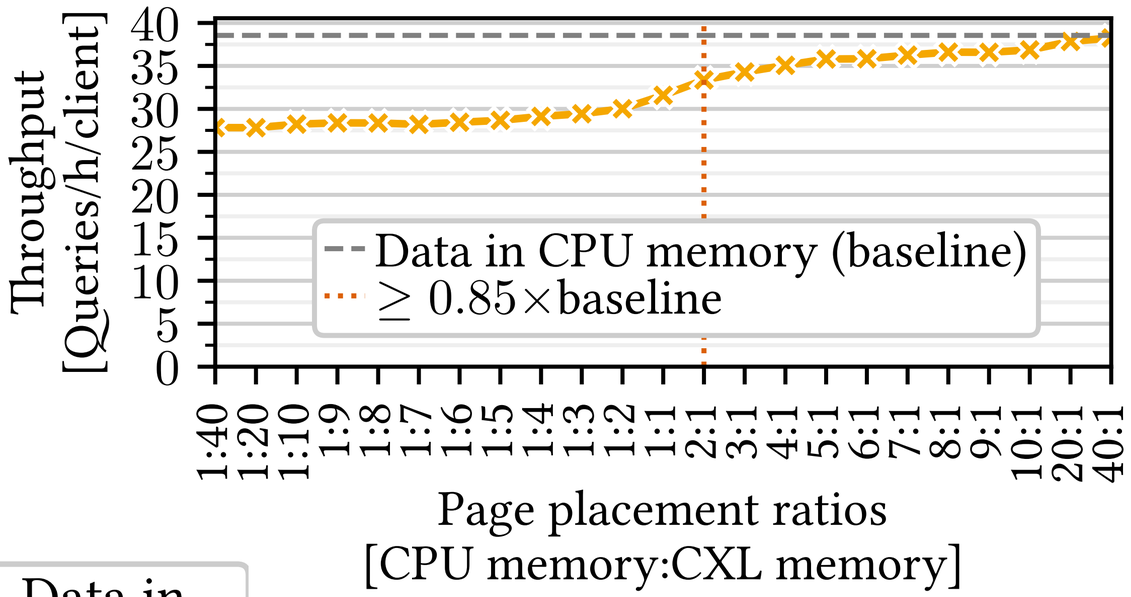
Latency- and Bandwidth-Bound μOps of the Write-Heavy Workload for Different Node Sizes



TPC-H on Hyrise



Black Box Approach (Page Granularity)



Access Frequency-Based (Column Granularity)