Towards Memory Disaggregation via NVLink C2C: **Benchmarking CPU-Requested GPU Memory Access**

Felix Werner, **Marcel Weisgut**, Tilmann Rabl

Hasso Plattner Institute, University of Potsdam, Germany

Motivation

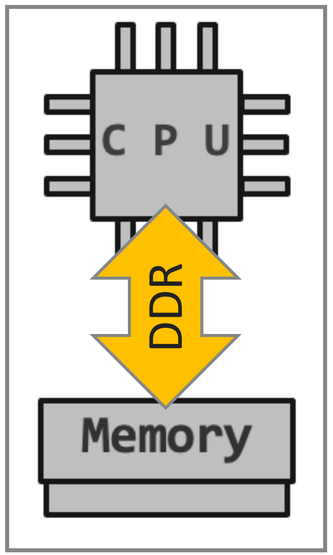
Separation of Resources

- Database systems have widely adapted separation of compute & storage
  - Elastic scaling of storage on demand
  - Avoid resource over-provisioning → cost reduction

Motivation

Separation of Resources

- Database systems have widely adapted separation of compute & storage
  - Elastic scaling of storage on demand
  - Avoid resource over-provisioning → cost reduction
- Compute & memory tightly coupled
  - Result: stranded memory



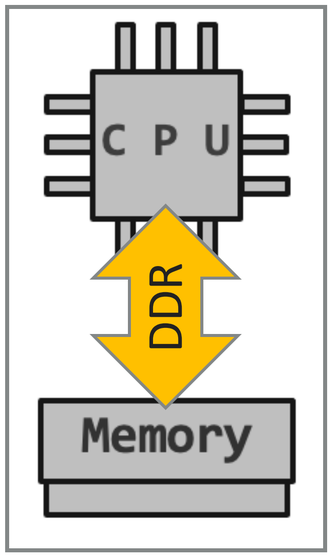
Motivation

Separation of Resources

- Database systems have widely adapted separation of compute & storage
  - Elastic scaling of storage on demand
  - Avoid resource over-provisioning → cost reduction
- Compute & memory tightly coupled
  - Result: stranded memory

Memory Disaggregation

- Initiatives towards memory disaggregation to separate memory from compute resources
  - Compute Express Link (CXL) as a recent option



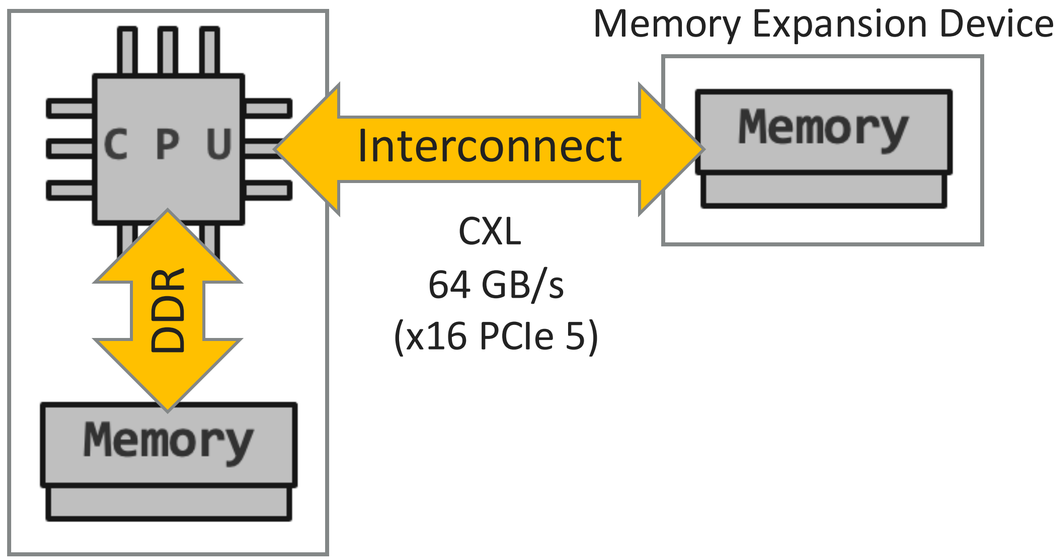
Motivation

Separation of Resources

- Database systems have widely adapted separation of compute & storage
  - Elastic scaling of storage on demand
  - Avoid resource over-provisioning → cost reduction
- Compute & memory tightly coupled
  - Result: stranded memory

Memory Disaggregation

- Initiatives towards memory disaggregation to separate memory from compute resources
  - Compute Express Link (CXL) as a recent option



Motivation

Separation of Resources

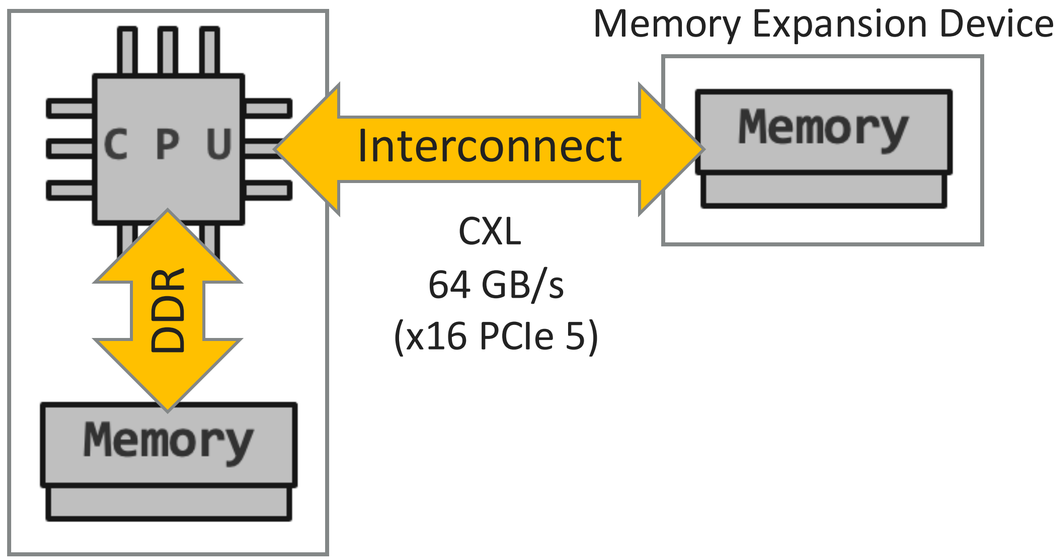
- Database systems have widely adapted separation of compute & storage
  - Elastic scaling of storage on demand
  - Avoid resource over-provisioning → cost reduction
- Compute & memory tightly coupled
  - Result: stranded memory

Memory Disaggregation

- Initiatives towards memory disaggregation to separate memory from compute resources
  - Compute Express Link (CXL) as a recent option

NVLink versions

Version	Data rate per link [GB/s]	# Lanes per link	# Links	Theoretical bandwidth	Architecture
1	20	8	4	80	Pascal
2	25	8	6	150	Volta
3	25	4	12	300	Ampere
4 / C2C	25	4	18	**450**	Hopper



Motivation

Separation of Resources

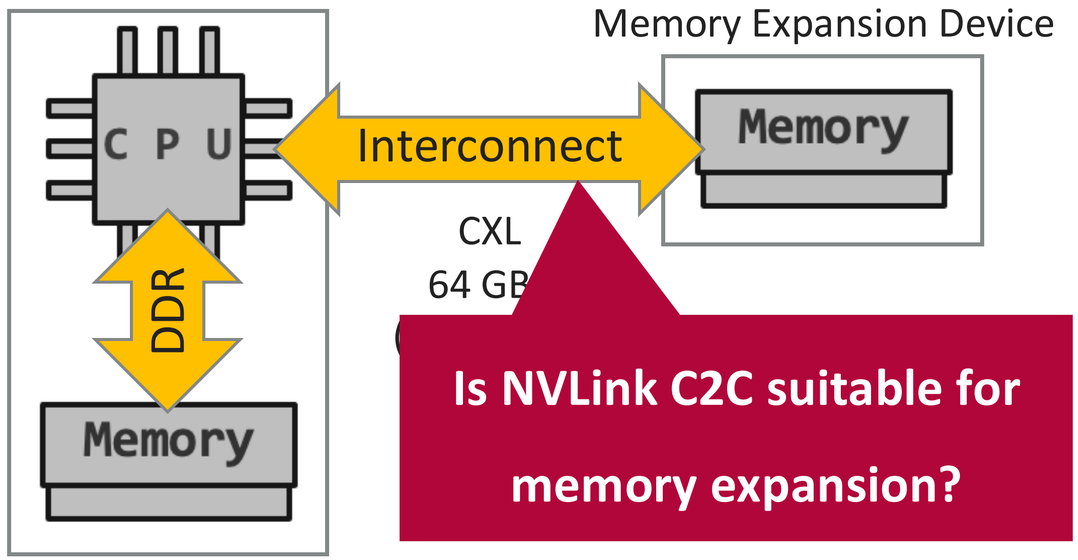
- Database systems have widely adapted separation of compute & storage
  - Elastic scaling of storage on demand
  - Avoid resource over-provisioning → cost reduction
- Compute & memory tightly coupled
  - Result: stranded memory

Memory Disaggregation

- Initiatives towards memory disaggregation to separate memory from compute resources
  - Compute Express Link (CXL) as a recent option

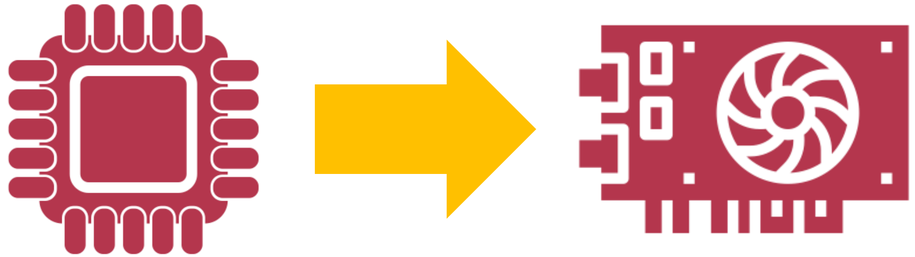
NVLink versions

Version	Data rate per link [GB/s]	# Lanes per link	# Links	Theoretical bandwidth	Architecture
1	20	8	4	80	Pascal
2	25	8	6	150	Volta
3	25	4	12	300	Ampere
4 / C2C	25	4	18	**450**	Hopper



Contributions

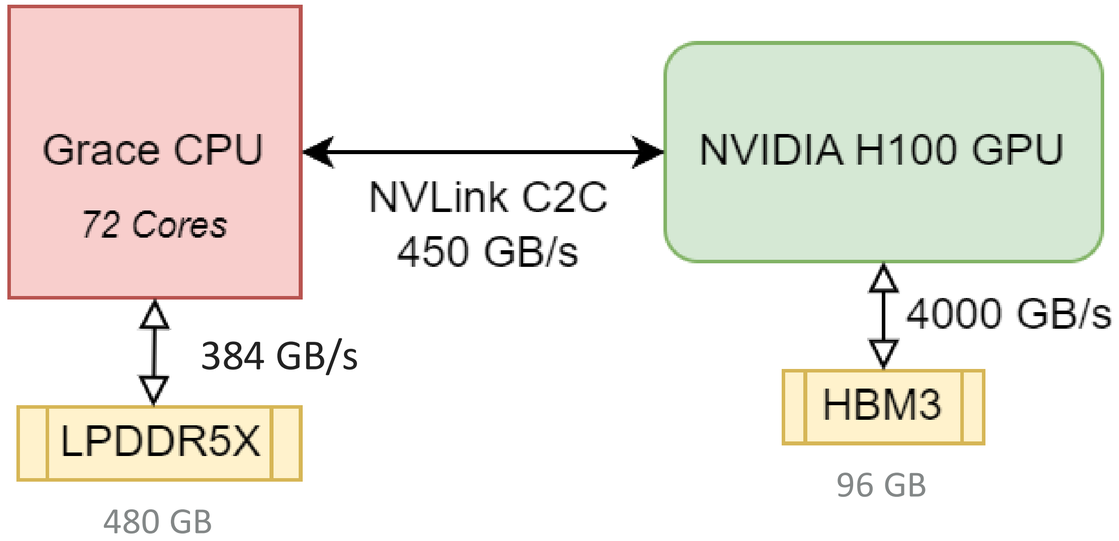
- Performance evaluation of memory access performance over NVLink C2C
  - CPU→GPU memory: throughput, latency
  - CPU→CPU & GPU memory: throughput expansion



- Impact of storing data in GPU memory on database operations
- Discuss suitability of NVLink-attached memory for memory expansion

Hardware Setup

NVIDIA Grace-Hopper Superchip



Bandwidth & Latency

Sustained Bandwidth [GB/s]

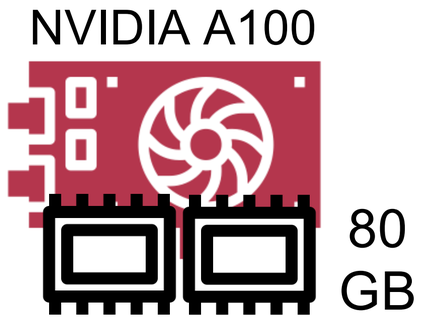
	CPU→CPU Memory		CPU→GPU Memory	
	Seq Read	Seq Write	Seq Read	Seq Write
	362	370	130 36% CPU→CPU	163
	Rnd Read	Rnd Write	Rnd Read	Rnd Write
	345	370	128	**168** 45% CPU→CPU

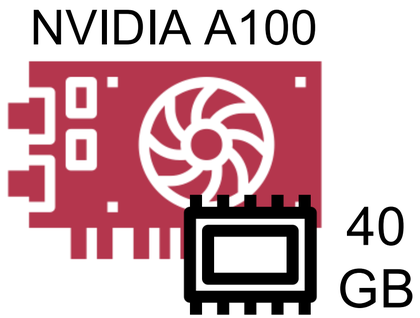
Bandwidth & Latency

	CPU→CPU Memory		CPU→GPU Memory	
Sustained Bandwidth [GB/s]	Seq Read	Seq Write	Seq Read	Seq Write
	362	370	130 36% CPU→CPU	163
	Rnd Read	Rnd Write	Rnd Read	Rnd Write
	345	370	128	**168** 45% CPU→CPU
Latency [ns] (random reads)	Idle		Idle	
	220		810 (3.7x)	
	Loaded		Loaded	
	440		1020 (2.3x)	

Economic Viability

- HBM is on-package memory → not modular like DRAM DIMMs
- Price estimation attempt: compare GPU models with different memory capacities

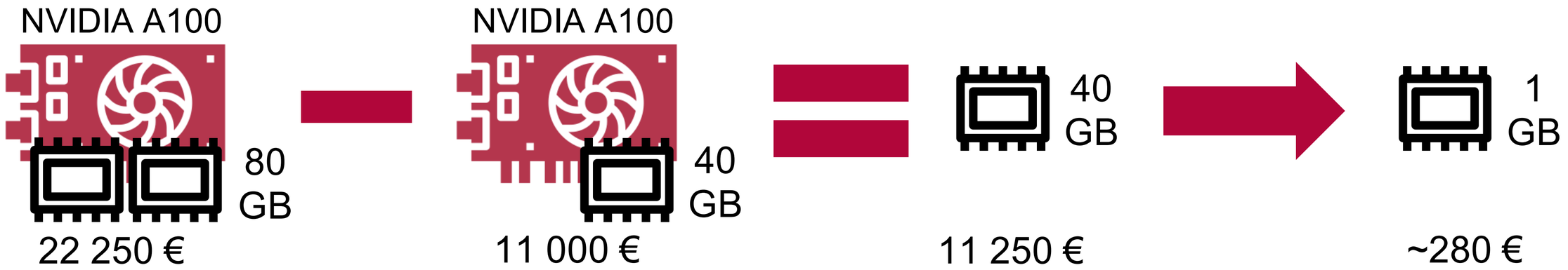




Prices retrieved from geizhals.eu in February 2025.

Economic Viability

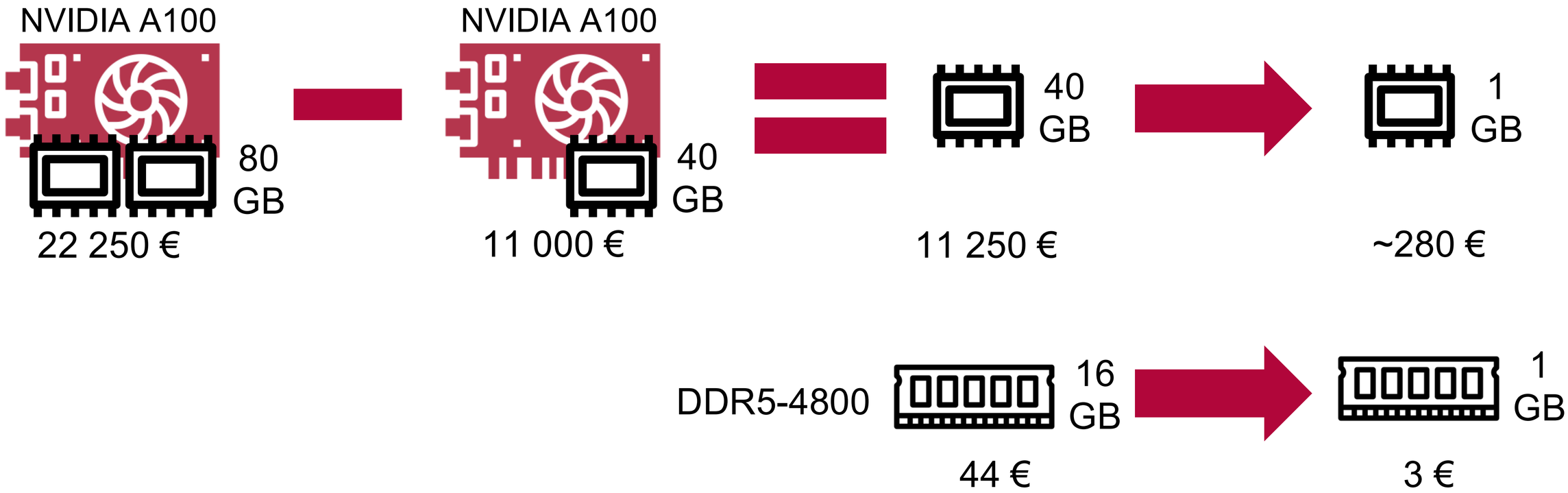
- HBM is on-package memory → not modular like DRAM DIMMs
- Price estimation attempt: compare GPU models with different memory capacities



Prices retrieved from geizhals.eu in February 2025.

Economic Viability

- HBM is on-package memory → not modular like DRAM DIMMs
- Price estimation attempt: compare GPU models with different memory capacities

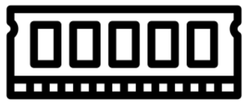


Prices retrieved from geizhals.eu in February 2025.

Economic Viability

1 GB

~280 €

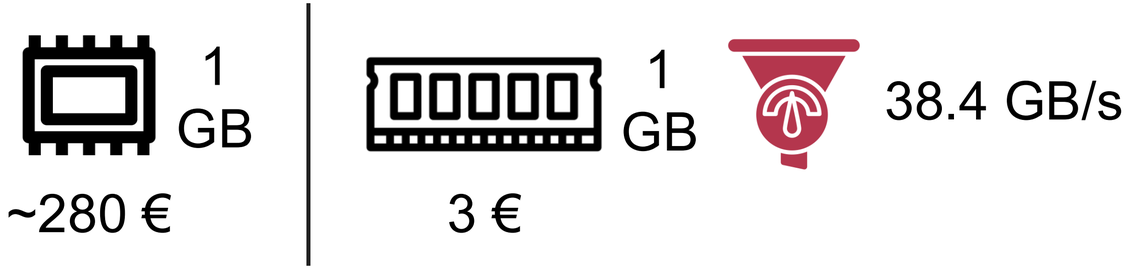


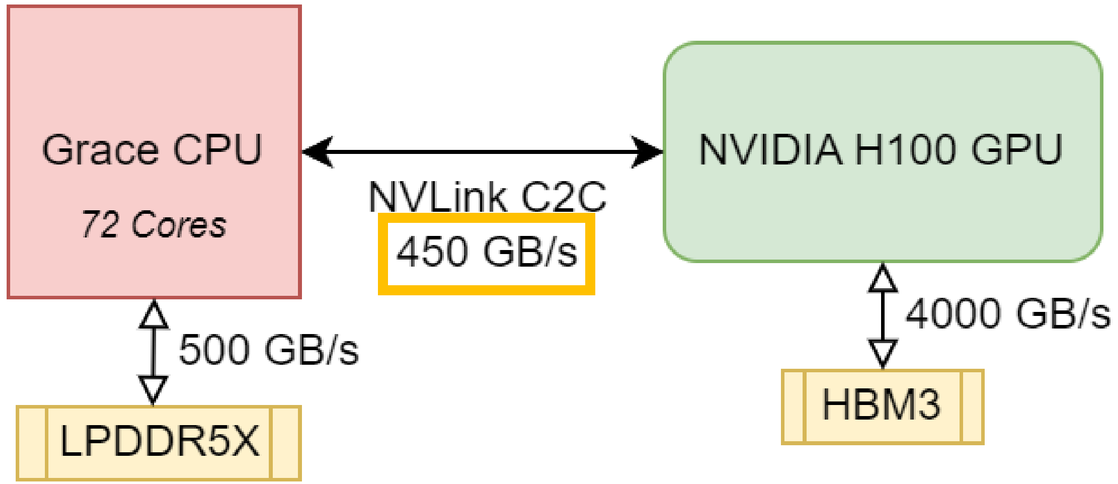
1 GB

3 €

Prices retrieved from geizhals.eu in February 2025.

Economic Viability



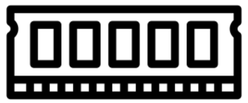


Prices retrieved from geizhals.eu in February 2025.

Economic Viability

1 GB

~280 €

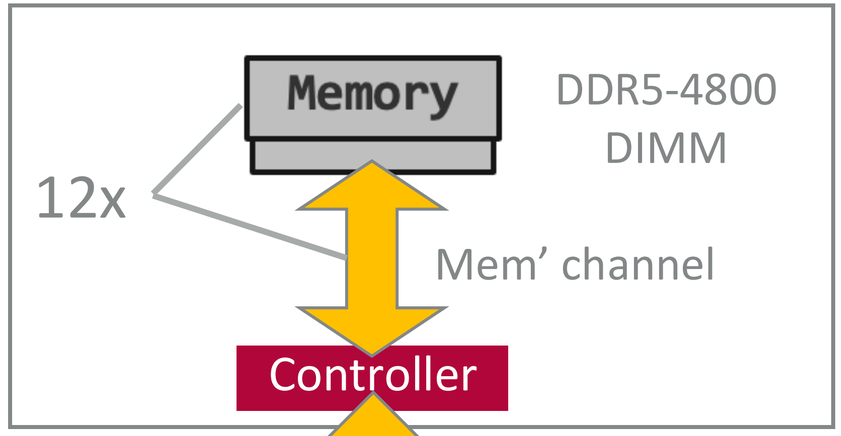


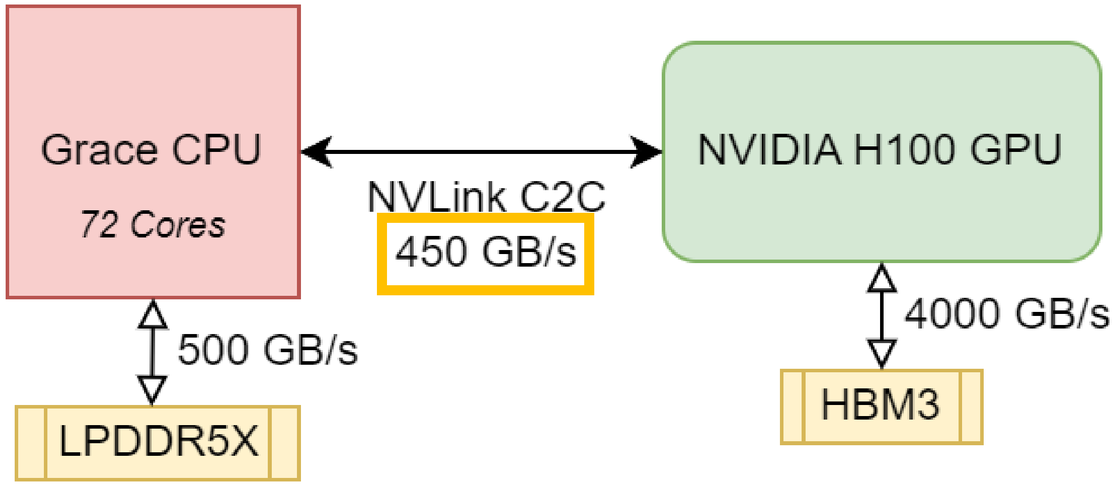
1 GB

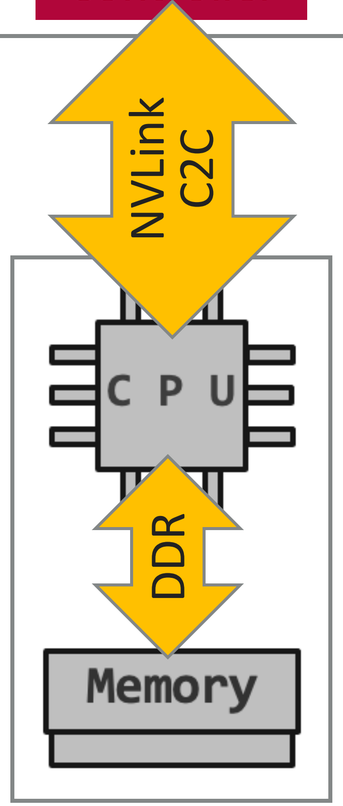
3 €

38.4 GB/s

DIMM setup
- 460.8 GB/s
- 192 GB
- 528 €







Prices retrieved from geizhals.eu in February 2025.

Summary

- CPU→GPU achieves 36% and 45% of the CPU→CPU read/write throughput
- CPU→GPU has access latencies of between ~800 ns to ~1000 ns (3.7x / 2.3x of CPU→CPU)
- Bandwidth expansion study shows improvements between of 1.3x for reads and 1.7x for writes
- Multiple DDR DIMMs more cost-efficient for pure memory expansion than GPU HBM

Data Engineering Systems Group @ HPI

hpi.de/rabl

marcel.weisgut(at)hpi.de

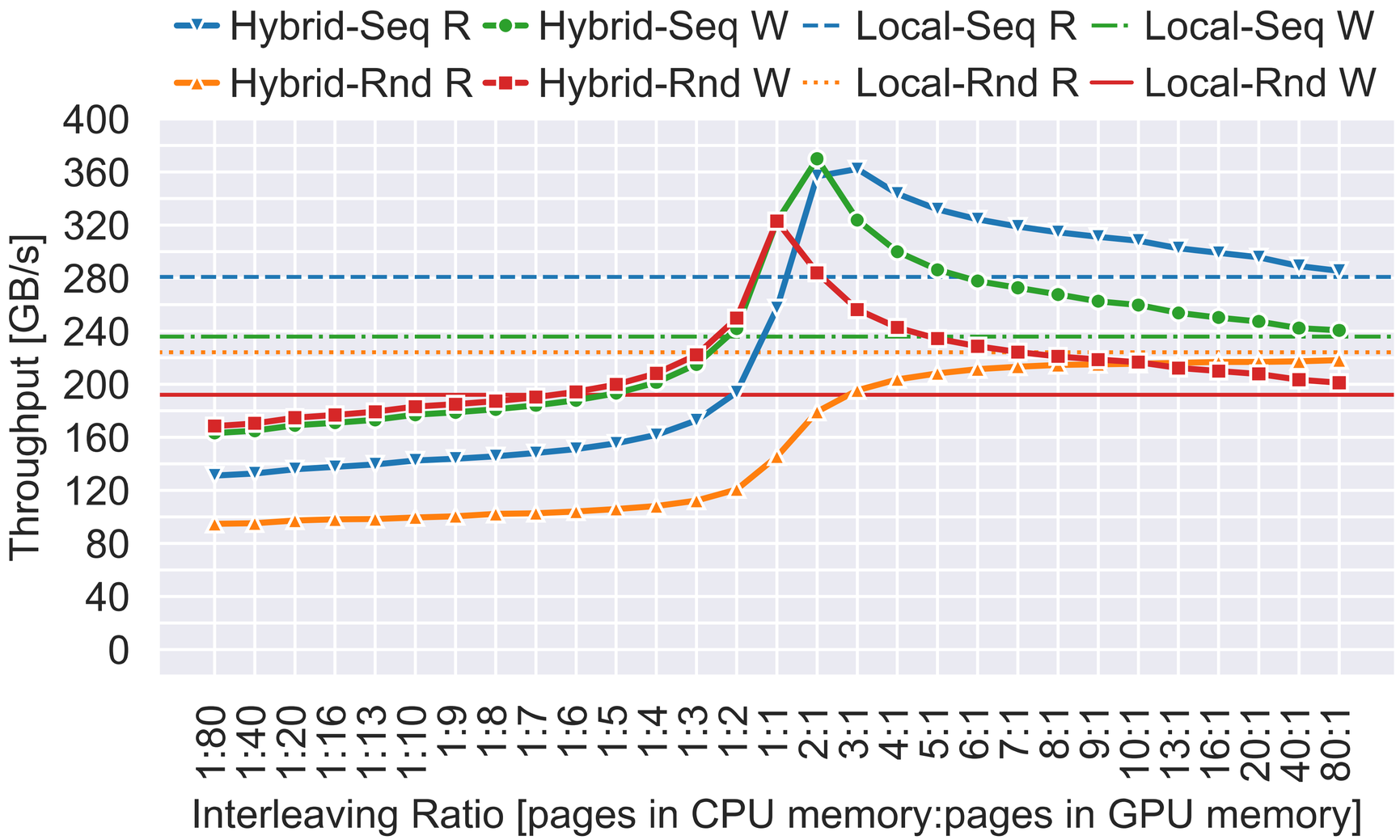








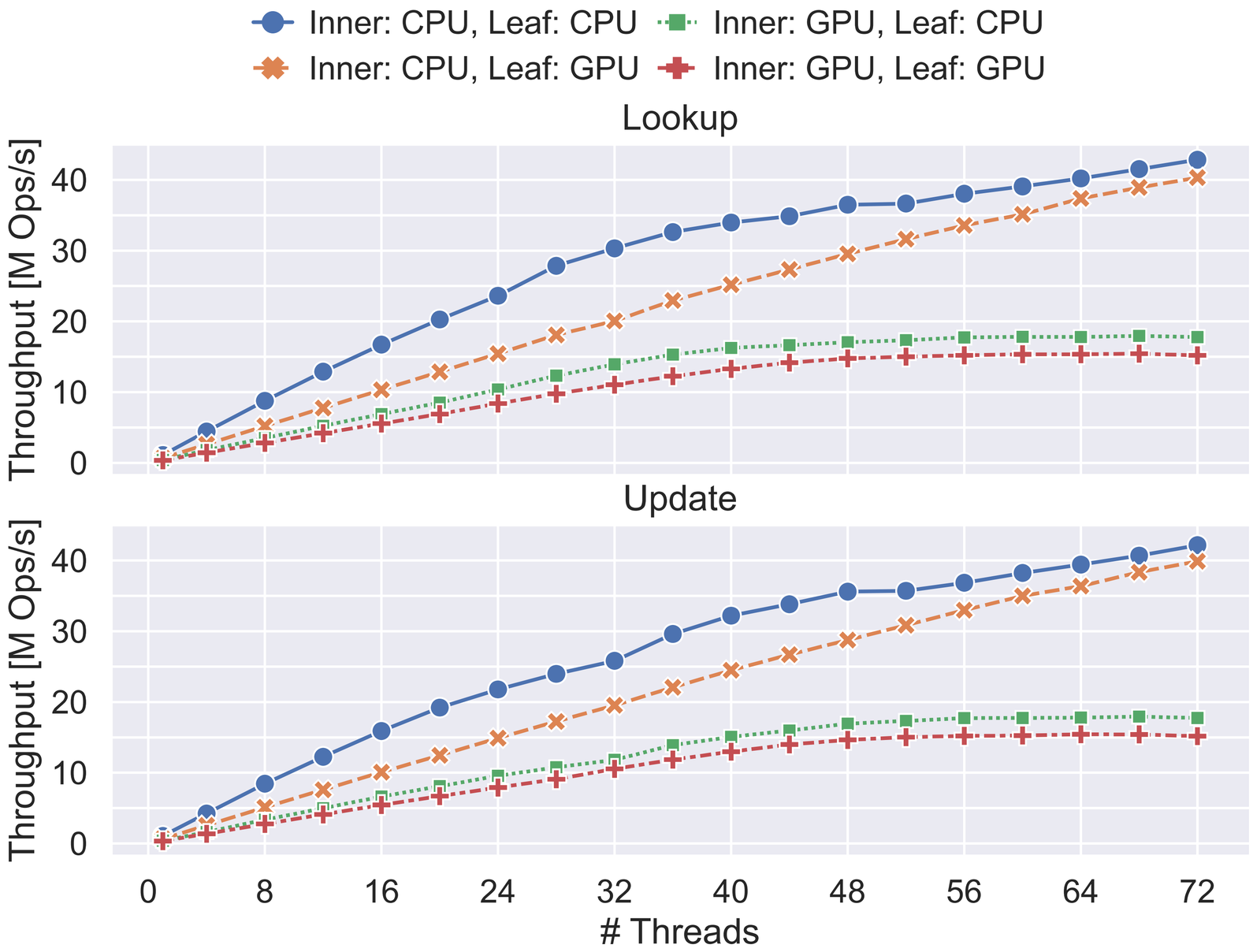

Bandwidth Expansion



Rnd writes: 1.7x
Seq writes: 1.6x
Seq reads: 1.3x

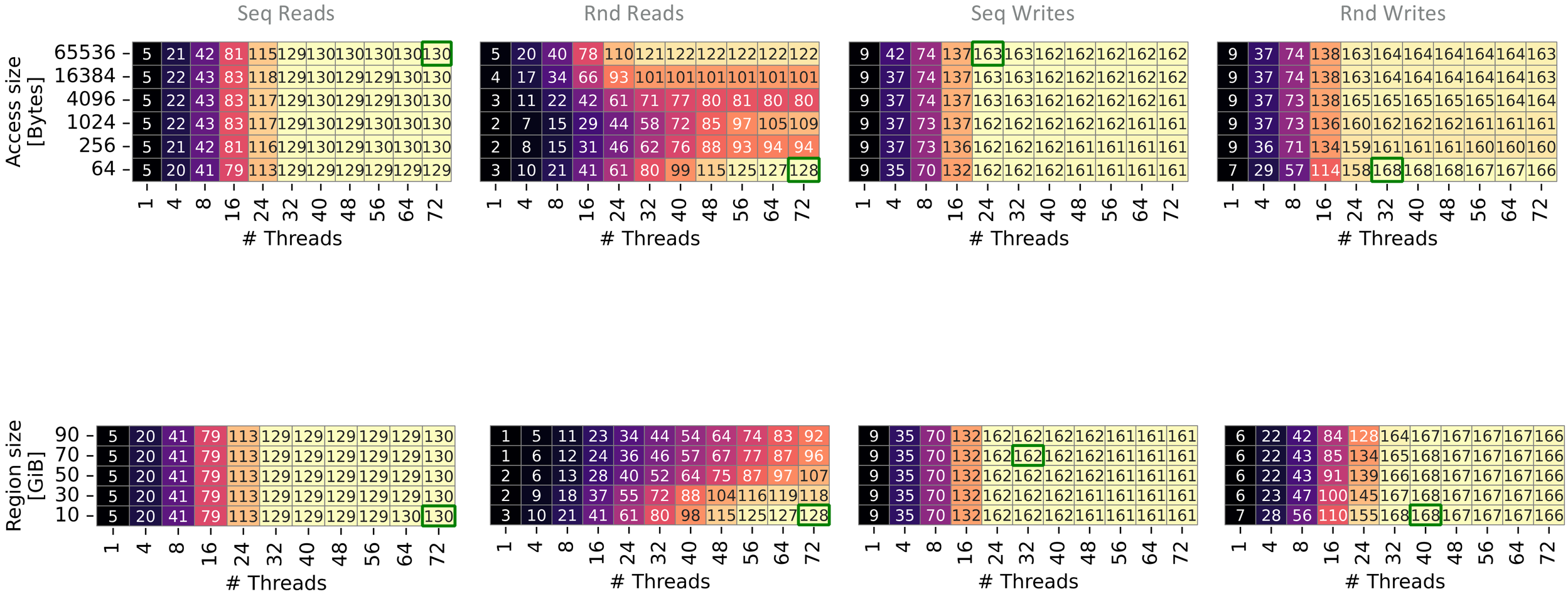
Access size: 64 B
Region size: 90 GiB

Reads: ld1
Writes: vst1q_u8_x4


Inner: CPU, Leaf: CPU
Inner: GPU, Leaf: CPU
Inner: CPU, Leaf: GPU
Inner: GPU, Leaf: GPU
Lookup
Update
Throughput [M Ops/s]
0
10
20
30
40
0
8
16
24
32
40
48
56
64
72
Threads

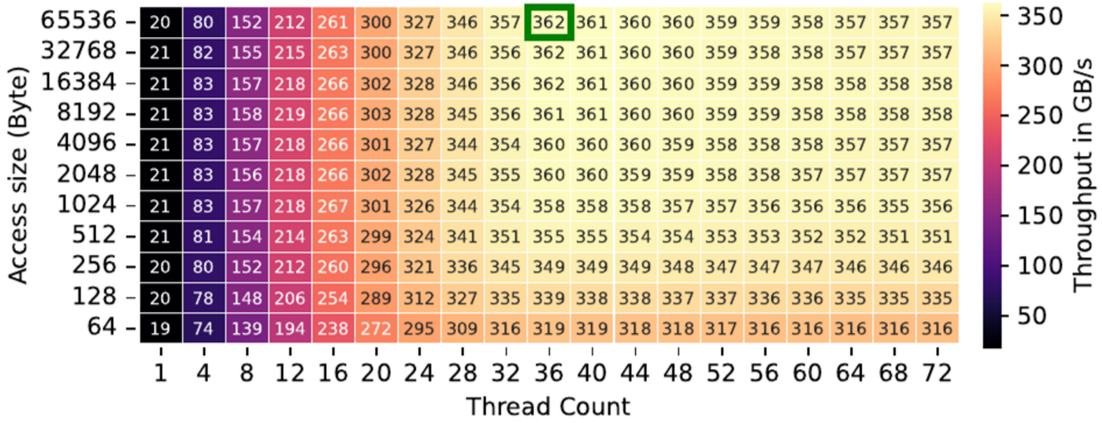
CXL-Bench

- Support for fundamental and chains of memory access operations
- Memory region preparation
  - Pin memory region to NUMA node(s) (**`mbind`**) – such as CXL and (modern) GPU memory
  - Multiple regions, partitions, and NUMA nodes supported
  - Round-robin & weighted page interleaving supported
- Task execution
  - Thread pinning to set of cores (**`pthread_setaffinity_np`**)
  - Memory access via scalar & SIMD instructions (using compiler vector intrinsics)
- Configurable with YAML files

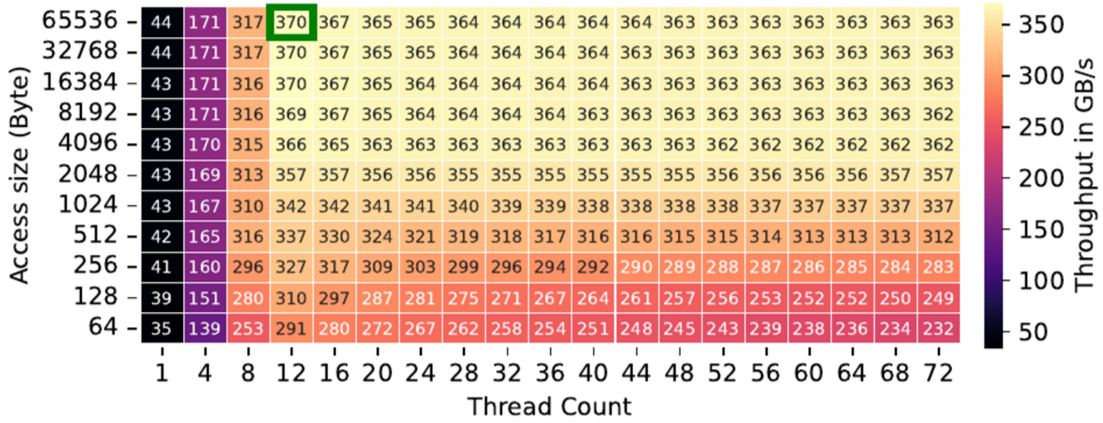

Seq Reads
Rnd Reads
Seq Writes
Rnd Writes
Access size [Bytes]
65536 5 21 42 81 115 129 130 130 130 130 130
16384 5 22 43 83 118 129 130 129 129 130 130
4096 5 22 43 83 117 129 130 129 129 130 130
1024 5 22 43 83 117 129 130 129 130 130 130
256 5 21 42 81 116 129 130 129 130 130 130
64 5 20 41 79 113 129 129 129 129 129 129
5 20 40 78 110 121 122 122 122 122 122
4 17 34 66 93 101 101 101 101 101 101
3 11 22 42 61 71 77 80 81 80 80
2 7 15 29 44 58 72 85 97 105 109
2 8 15 31 46 62 76 88 93 94 94
3 10 21 41 61 80 99 115 125 127 128
9 42 74 137 163 163 162 162 162 162 162
9 37 74 137 163 163 162 162 162 162 162
9 37 74 137 163 163 162 162 162 162 161
9 37 73 137 162 162 162 162 162 161 161
9 37 73 136 162 162 162 161 161 161 161
9 35 70 132 162 162 162 161 161 161 161
9 37 74 138 163 164 164 164 164 164 163
9 37 74 138 163 164 164 164 164 164 163
9 37 73 138 165 165 165 165 165 164 164
9 37 73 136 160 162 162 162 161 161 161
9 36 71 134 159 161 161 161 160 160 160
7 29 57 114 158 168 168 168 167 167 166
1 4 8 16 24 32 40 48 56 64 72
Threads
Region size [GiB]
90 5 20 41 79 113 129 129 129 129 129 130
70 5 20 41 79 113 129 129 129 129 129 130
50 5 20 41 79 113 129 129 129 129 129 130
30 5 20 41 79 113 129 129 129 129 129 130
10 5 20 41 79 113 129 129 129 129 130 130
1 5 11 23 34 44 54 64 74 83 92
1 6 12 24 36 46 57 67 77 87 96
2 6 13 28 40 52 64 75 87 97 107
2 9 18 37 55 72 88 104 116 119 118
3 10 21 41 61 80 98 115 125 127 128
9 35 70 132 162 162 162 162 161 161 161
9 35 70 132 162 162 162 162 161 161 161
9 35 70 132 162 162 162 161 161 161 161
9 35 70 132 162 162 162 161 161 161 161
9 35 70 132 162 162 162 161 161 161 161
6 22 42 84 128 164 167 167 167 167 166
6 22 43 85 134 165 168 167 167 167 166
6 22 43 91 139 166 168 167 167 167 166
6 23 47 100 145 167 168 167 167 167 166
7 28 56 110 155 168 168 167 167 167 166
1 4 8 16 24 32 40 48 56 64 72
Threads

LOCAL MEMORY ACCESS THROUGHPUT

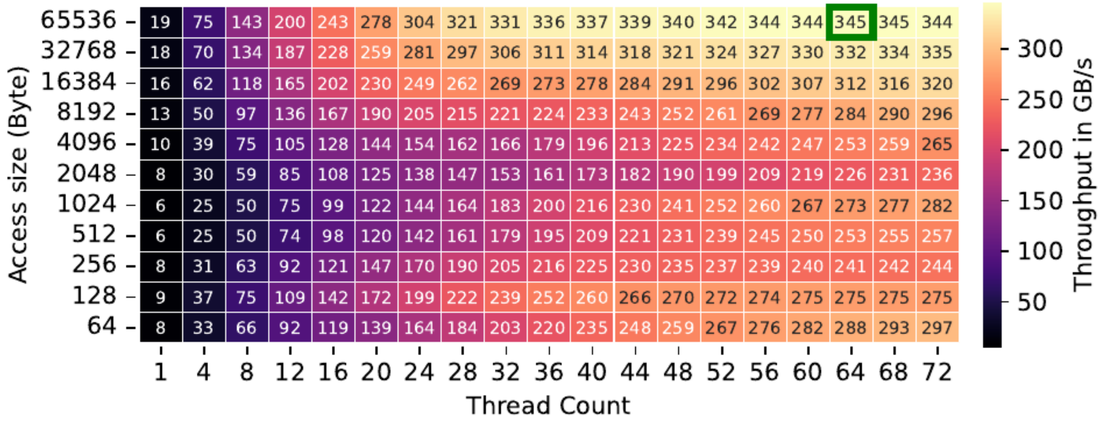
Sequential Reads



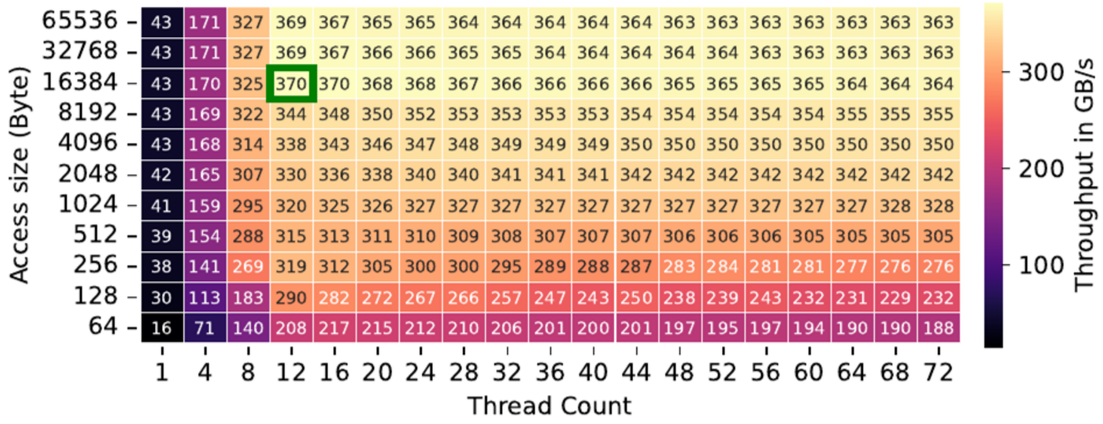
Sequential Writes

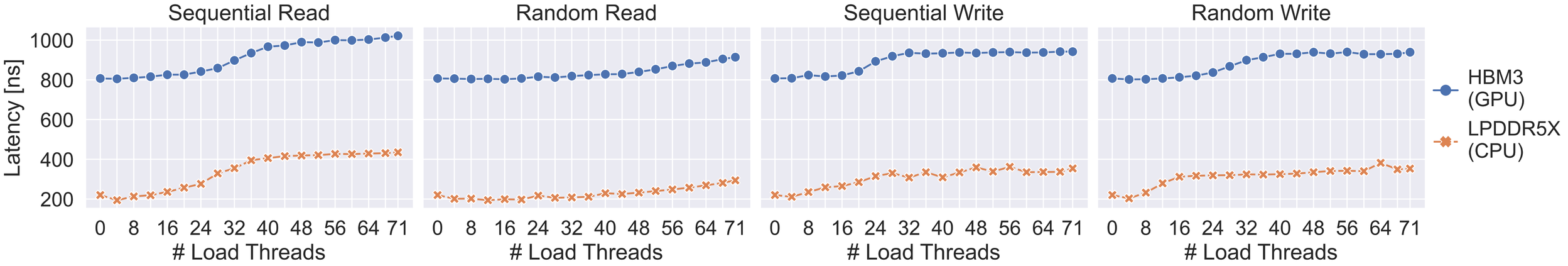


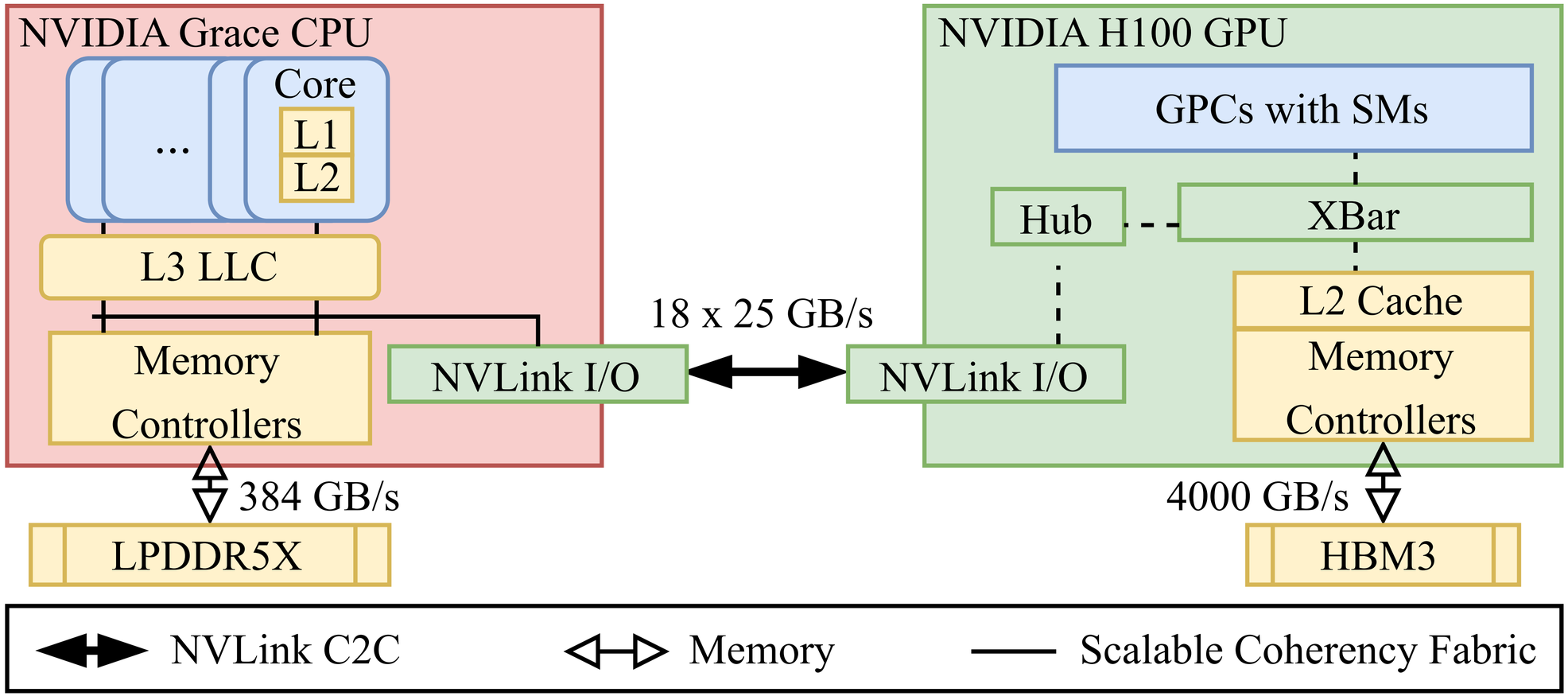
Random Reads



Random Writes



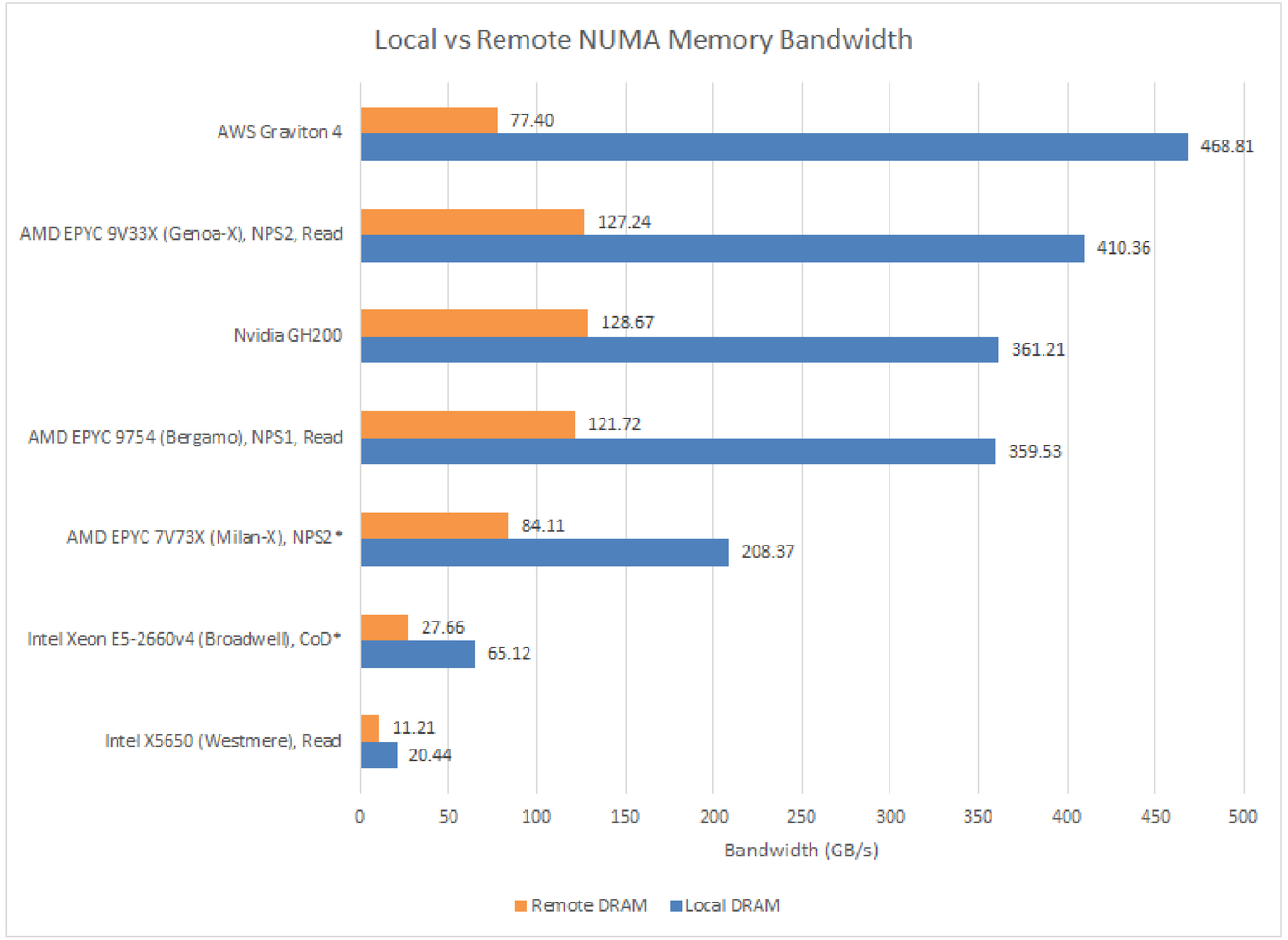

Sequential Read
Random Read
Sequential Write
Random Write
Latency [ns]
1000
800
600
400
200
0 8 16 24 32 40 48 56 64 71
Load Threads
HBM3 (GPU)
LPDDR5X (CPU)


NVIDIA Grace CPU
Core
L1
L2
...
L3 LLC
Memory Controllers
NVLink I/O
18 x 25 GB/s
NVLink I/O
NVIDIA H100 GPU
GPCs with SMs
Hub
XBar
L2 Cache
Memory Controllers
384 GB/s
4000 GB/s
LPDDR5X
HBM3
NVLink C2C
Memory
Scalable Coherency Fabric

NVLink Evolution

Version	Data rate per link [GB/s]	# Lanes per link	# Links	Theoretical bandwidth	Architecture
1	20	8	4	80	Pascal
2	25	8	6	150	Volta
3	25	4	12	300	Ampere
4 / C2C	25	4	18	**450**	Hopper

NVLINK C2C VS CPU INTERCONNECTS



https://chipsandcheese.com/2024/07/31/grace-hopper-nvidias-halfway-apu/