

CS 7270: Advanced Database Systems Fall 2025

Lecture 26

Databases on Modern Hardware

Prashant Pandey

p.pandey@northeastern.edu

Acknowledgement: Prof. Xiangyao Yu, University of Wisconsin

Moore's law

Often expressed as:

“X doubles every 18-24 months”

where X is:

“performance”

CPU clock speed

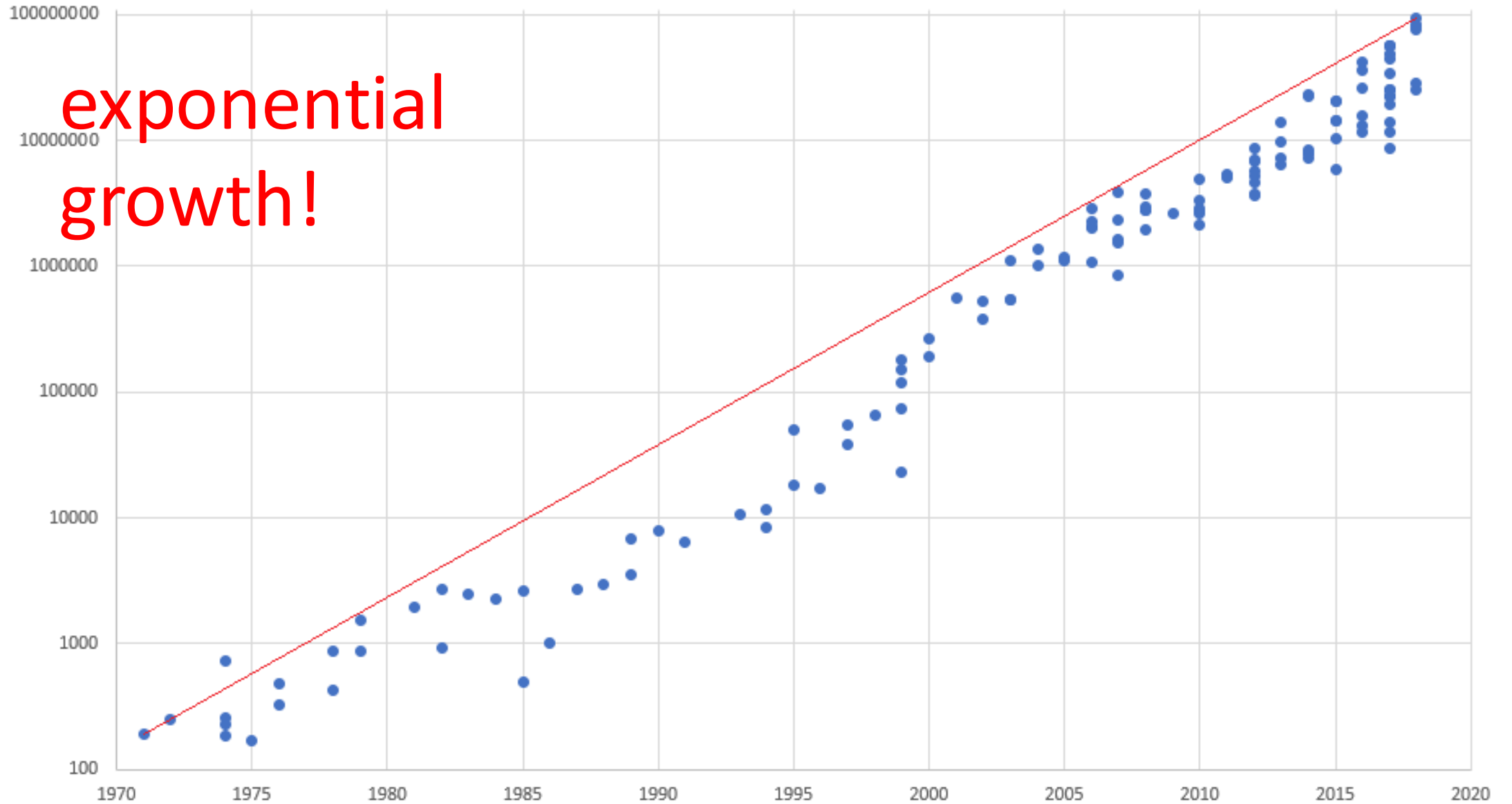
the number of transistors per chip

which one is it?

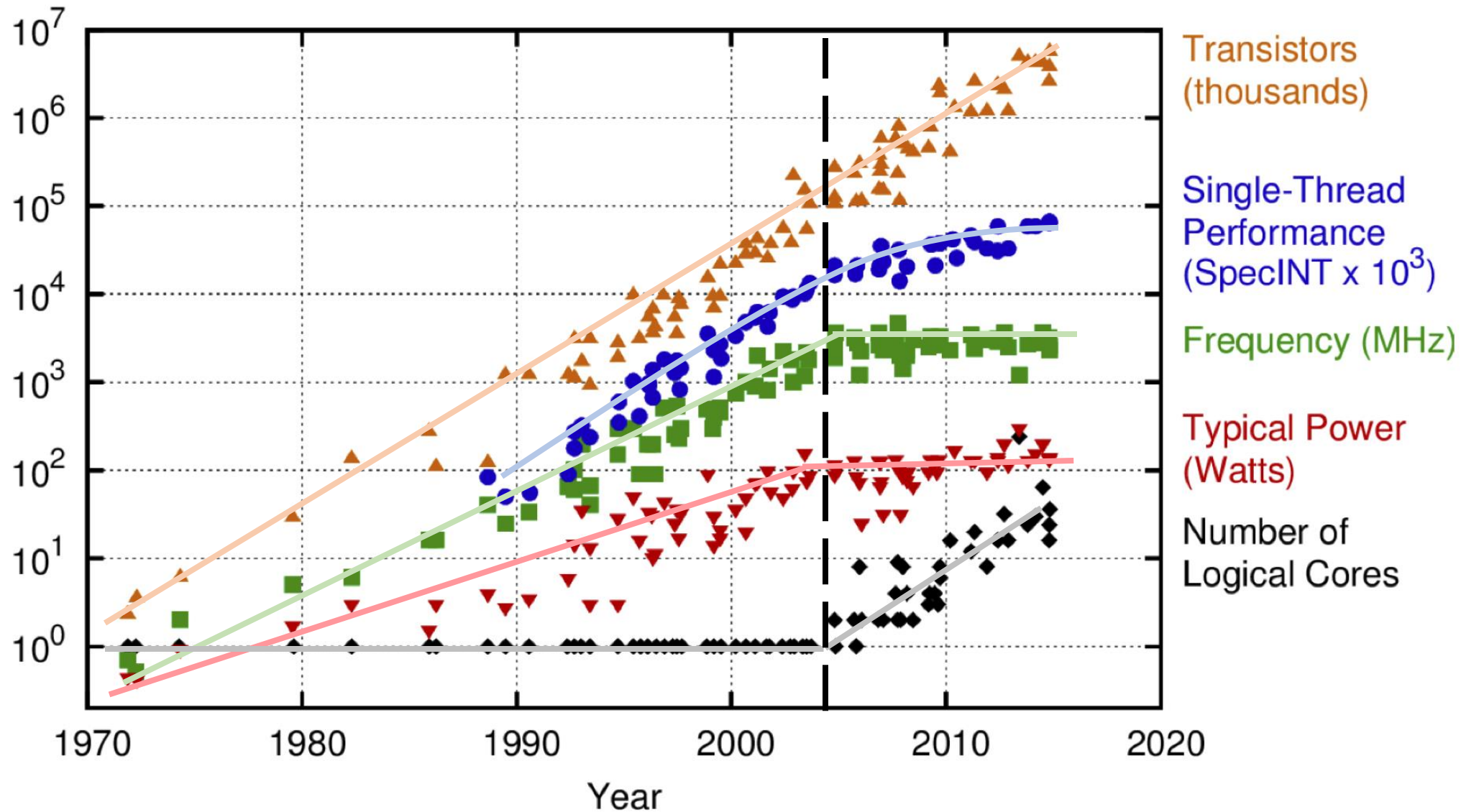
Moore's Law is Alive and Well!
Transistors per Square Millimeter by Year

but ...

exponential
growth!



40 Years of Microprocessor Trend Data



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 2010-2015 by K. Rupp

Can (a single) CPU cope with increasing application complexity?

No, because CPUs (cores) are **not** getting faster!!!

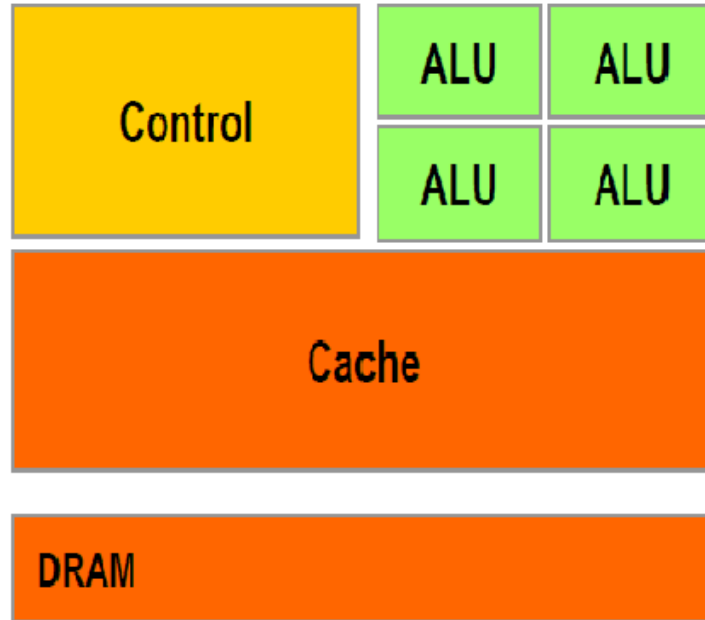
.. but they are getting more and more (**parallel**)

Research Challenges

how to handle them?

how to parallel program?

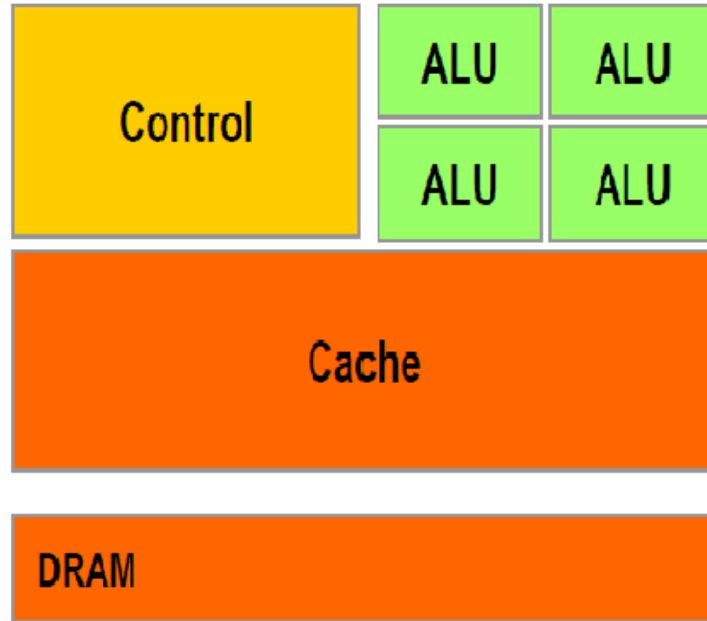
CPU vs. GPU



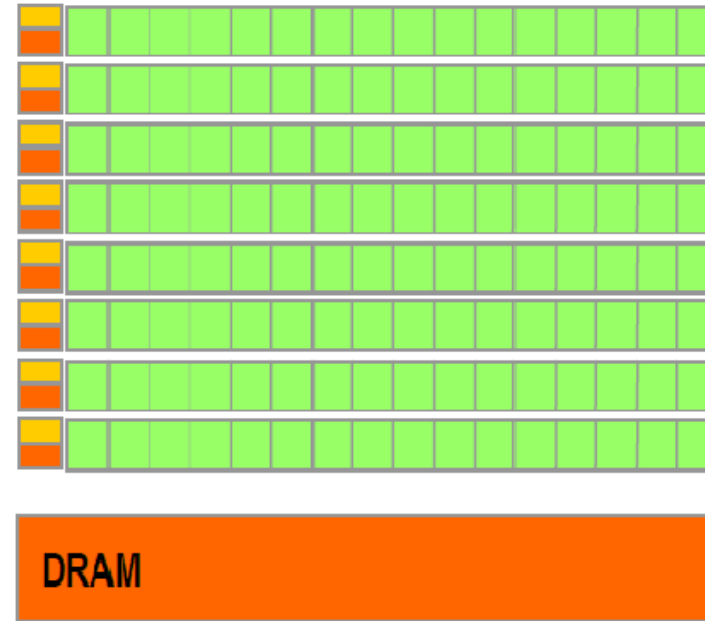
CPU

CPU: A few powerful cores with large caches. Optimized for sequential computation

CPU vs. GPU



CPU



GPU

CPU: A few powerful cores with large caches. Optimized for sequential computation

GPU: Many small cores. Optimized for parallel computation

GPU architecture

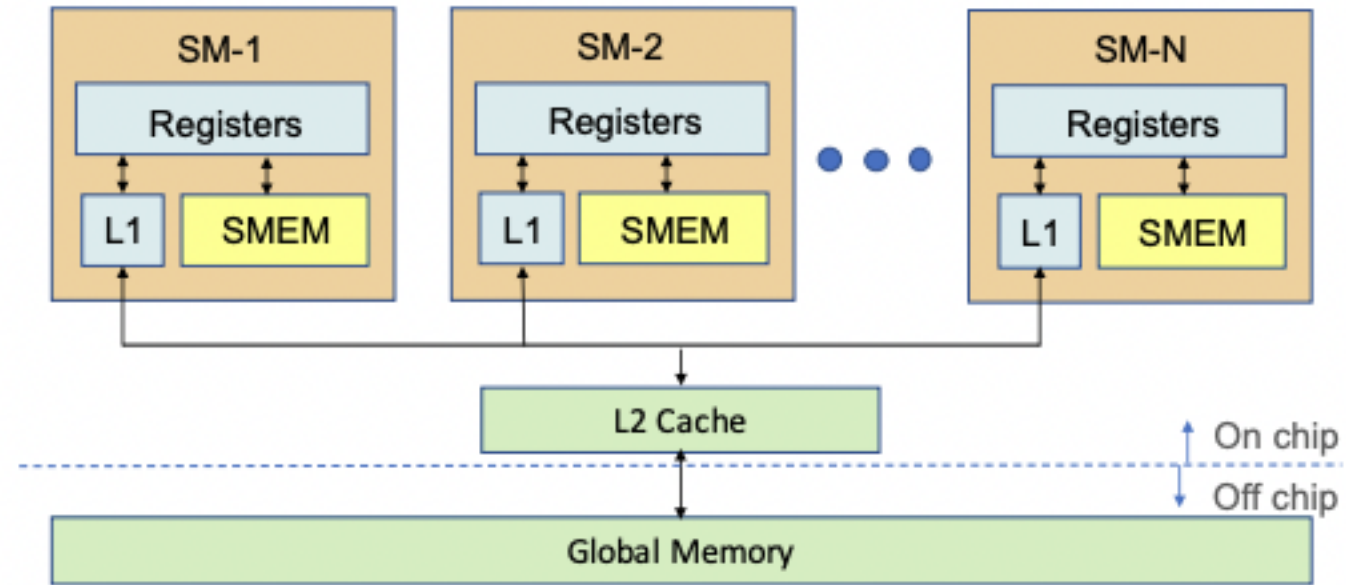
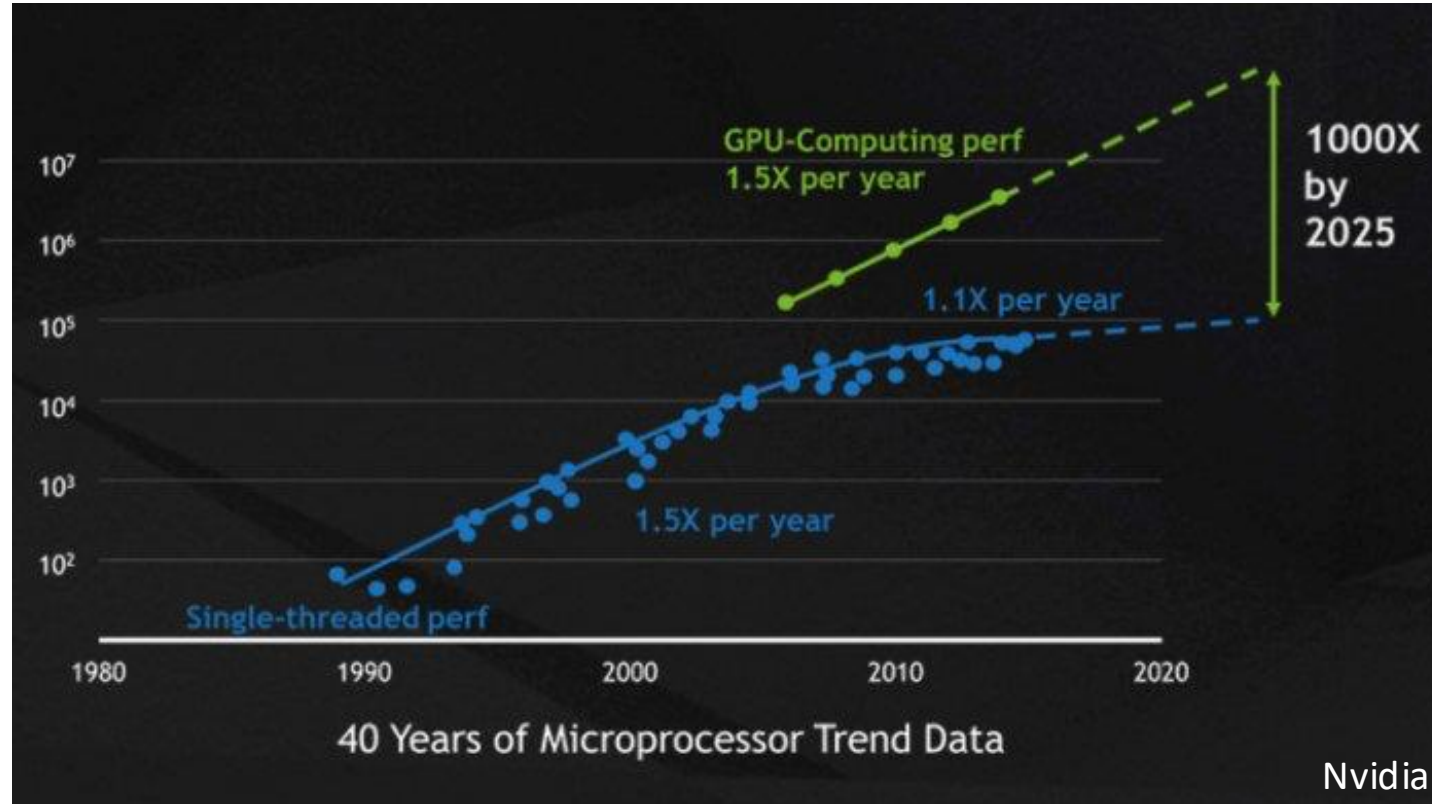


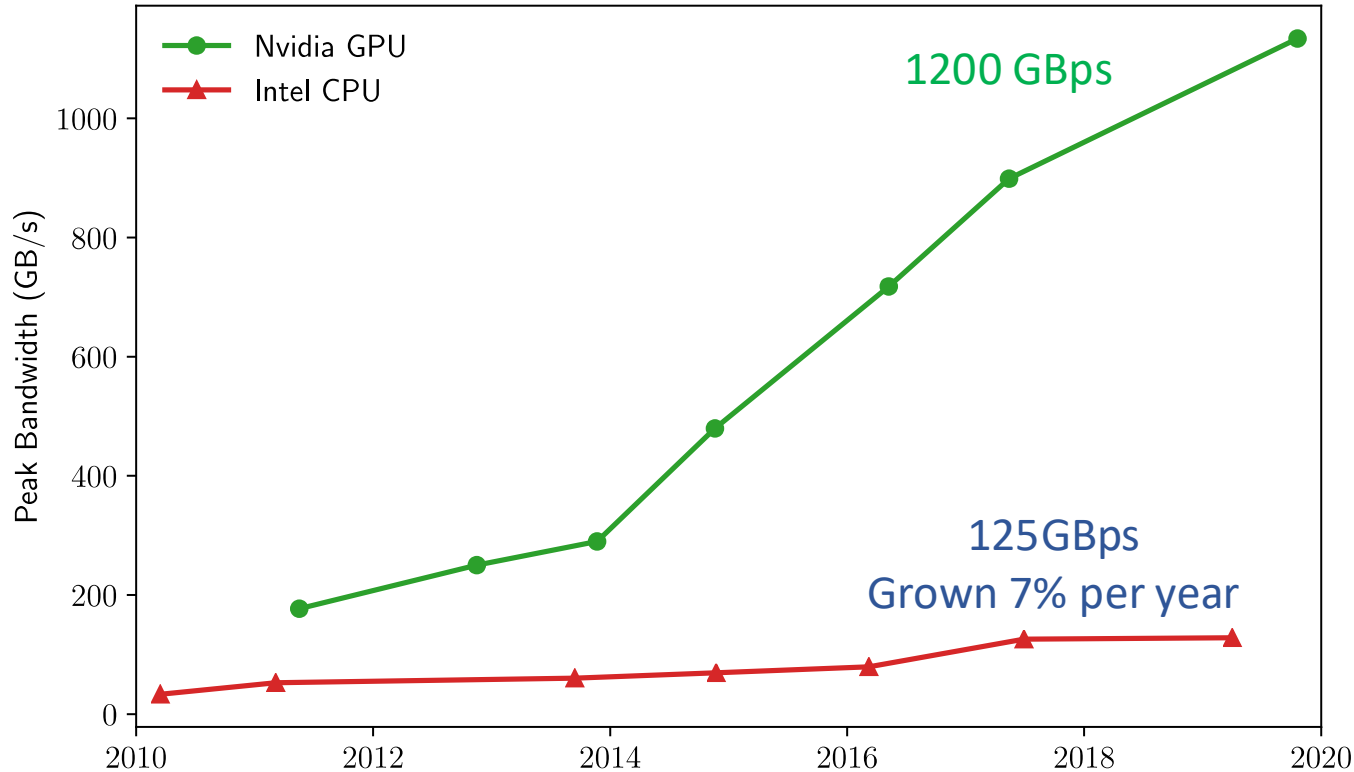
Figure 1: GPU Memory Hierarchy

CPU vs. GPU – Processing Units



	Throughput	Power	Throughput/Power
Intel Skylake	128 GFLOPS/4 Cores	100+ Watts	~1 GFLOPS/Watt
NVIDIA V100	15 TFLOPS	200+ Watts	~75 GFLOPS/Watt

CPU vs. GPU — Memory Bandwidth



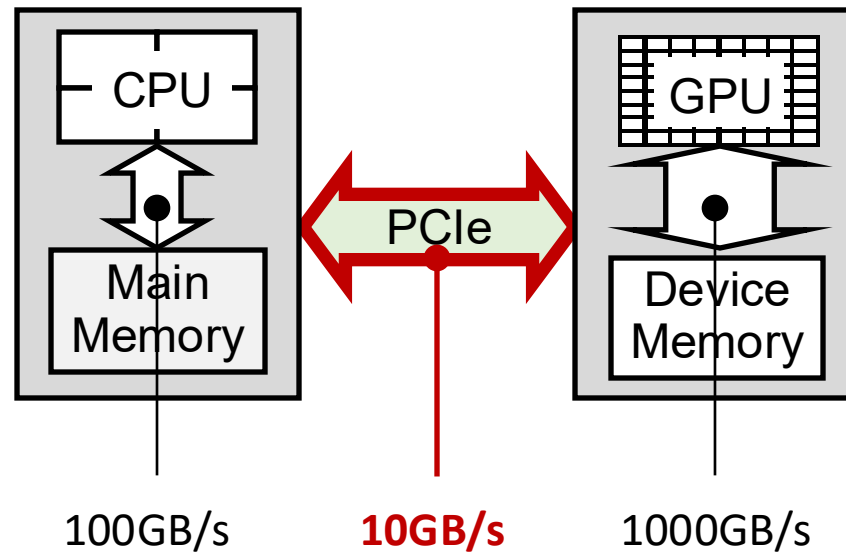
GPU has one order of magnitude higher memory bandwidth than CPU

Memory Bandwidth is the bottleneck for in-memory analytics

A natural idea: **use GPUs for data analytics**

GPU-DB Limitations

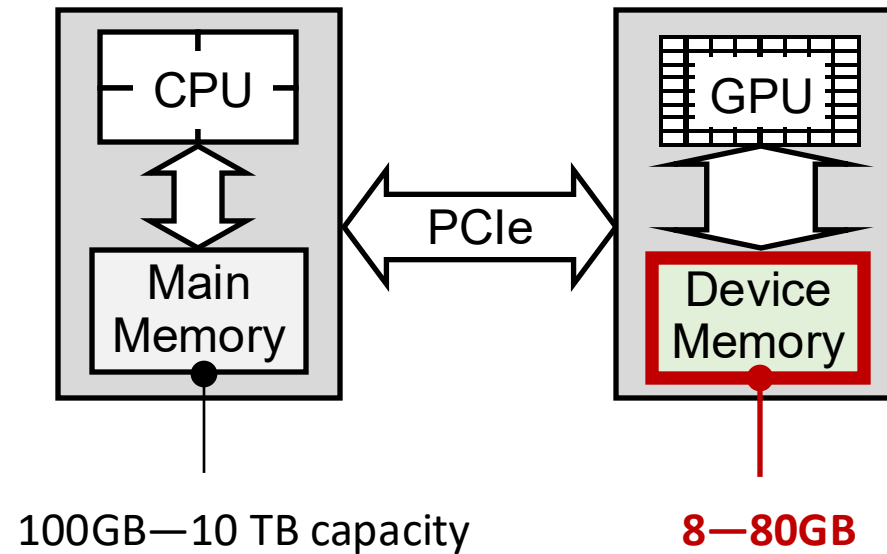
Limitation 1: Low interconnect bandwidth



GPU-DB Limitations

Limitation 1: Low interconnect bandwidth

Limitation 2: Small GPU memory capacity

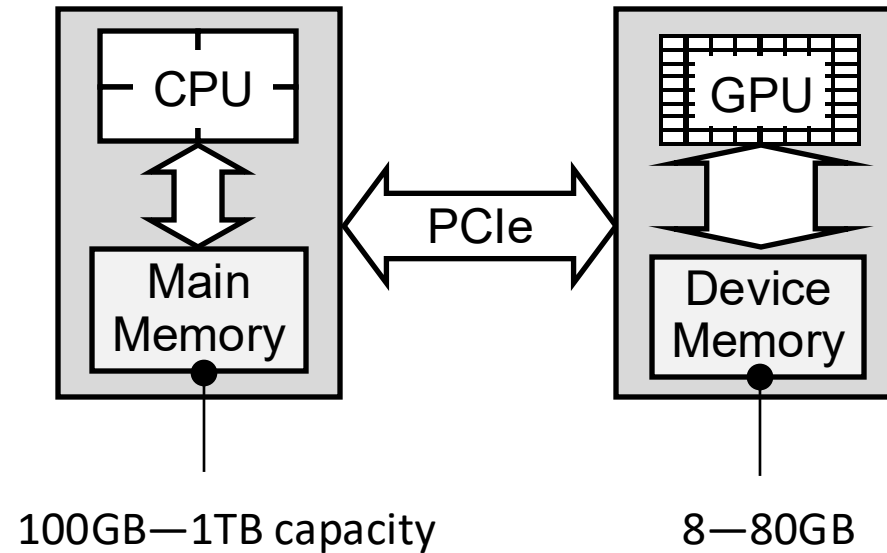


GPU-DB Limitations

Limitation 1: Low interconnect bandwidth

Limitation 2: Small GPU memory capacity

Limitation 3: Coarse-grained cooperation of CPU and GPU

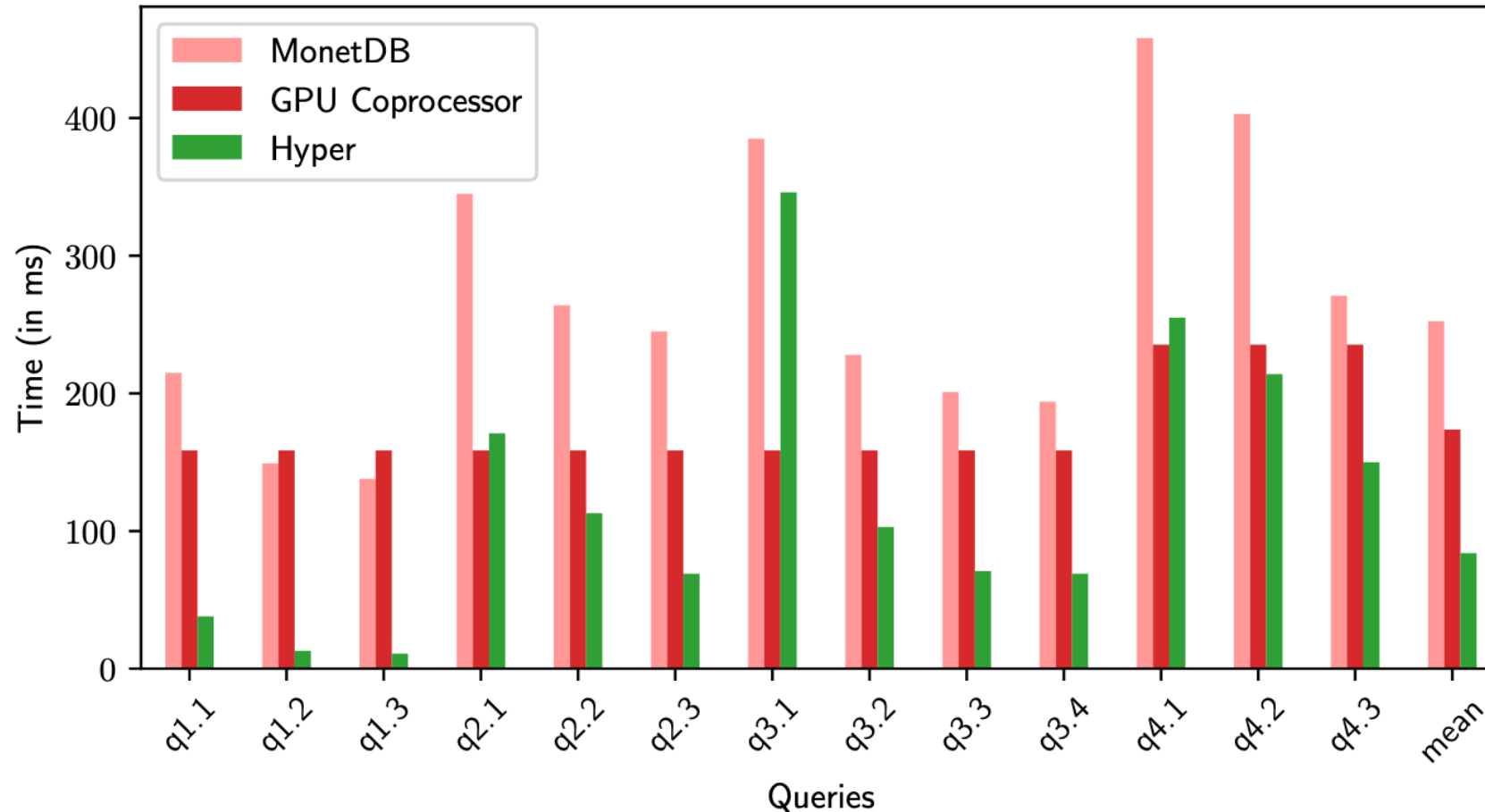


GPU Database Operation Mode

Coprocessor mode: Every query loads data from CPU memory to GPU

GPU-only mode: Store working set in GPU memory and run the entire query on GPU

CPU-only vs. Coprocessor



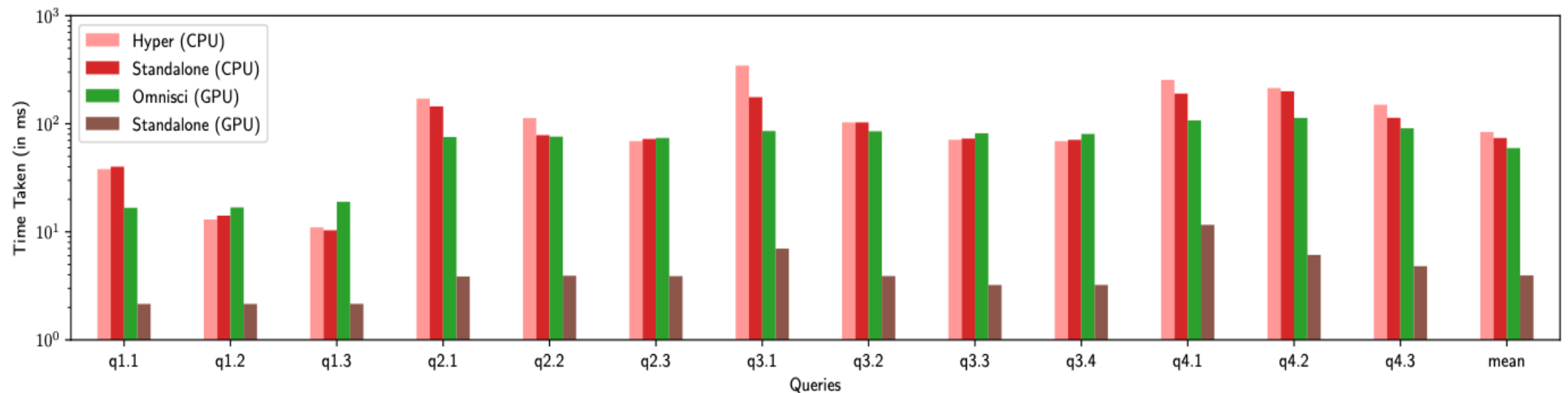
Key observation: With efficient implementations that can saturate memory bandwidth
GPU-only > coprocessor > CPU-only

Star-Schema Benchmark

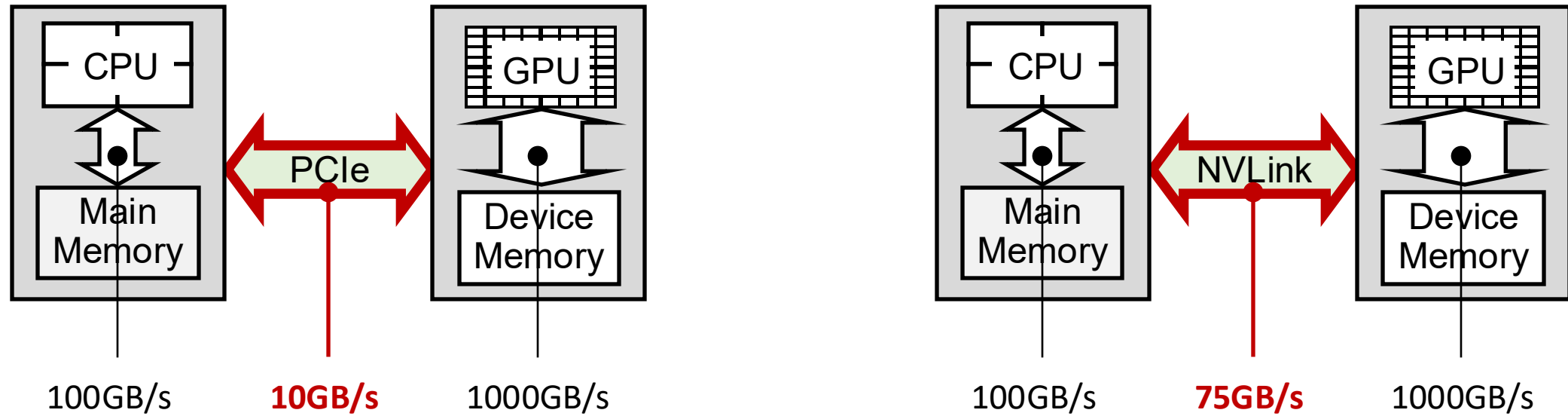
Platform	CPU	GPU
Model	Intel i7-6900	Nvidia V100
Cores	8 (16 with SMT)	5000
Memory Capacity	64 GB	32 GB
L1 Size	32KB/Core	16KB/SM
L2 Size	256KB/Core	6MB (Total)
L3 Size	20MB (Total)	-
Read Bandwidth	53GBps	880GBps
Write Bandwidth	55GBps	880GBps
L1 Bandwidth	-	10.7TBps
L2 Bandwidth	-	2.2TBps
L3 Bandwidth	157GBps	-

Crystal-based implementations always saturate GPU memory bandwidth

GPU is on average **25X** faster than CPU



Emerging Fast Interconnect



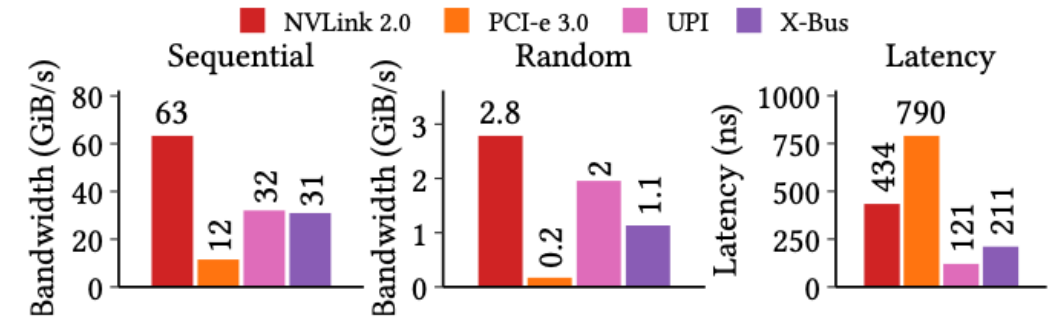
Fast Interconnect can solve the PCIe bottleneck

Emerging alternative interconnect technologies:

- NVLink
- Infinity Fabric
- Compute Express Link (CXL)

NVLink Bandwidth and Latency

NVLink has much higher bandwidth than PCIe

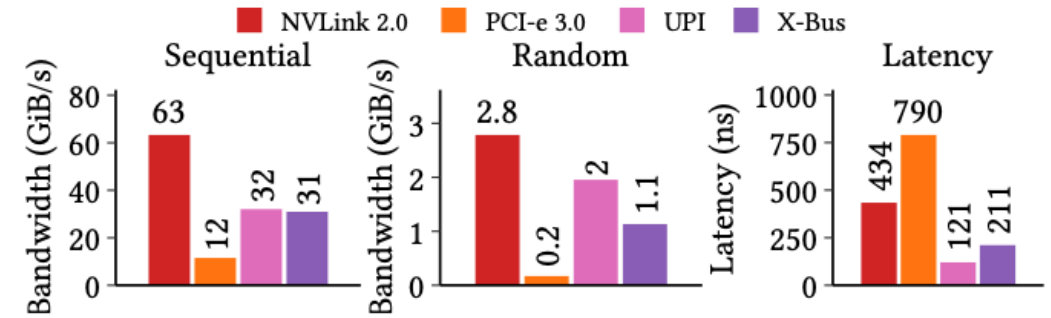


(a) NVLink 2.0 vs. CPU & GPU Interconnects.

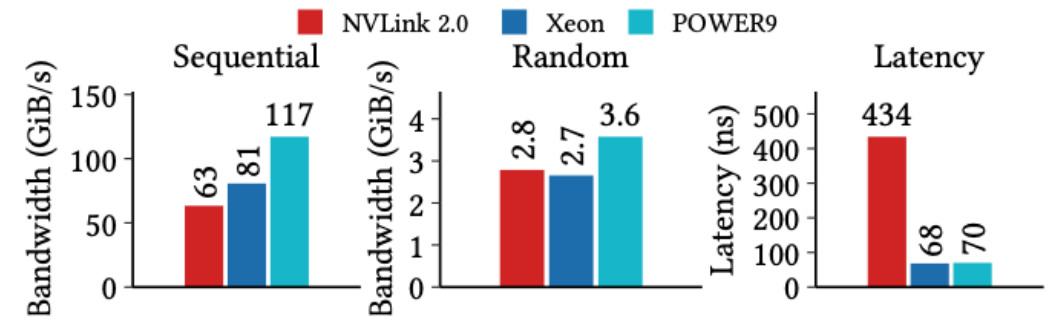
NVLink Bandwidth and Latency

NVLink has much higher bandwidth than PCIe

NVLink has comparable bandwidth as CPU local memory



(a) NVLink 2.0 vs. CPU & GPU Interconnects.



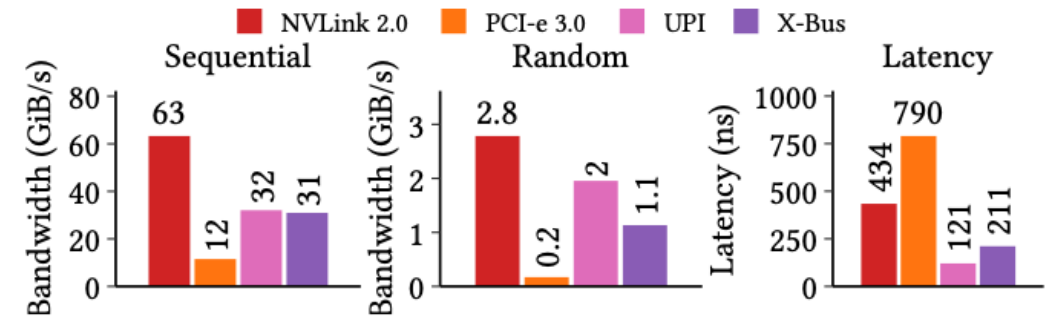
(b) NVLink 2.0 vs. CPU memory.

NVLink Bandwidth and Latency

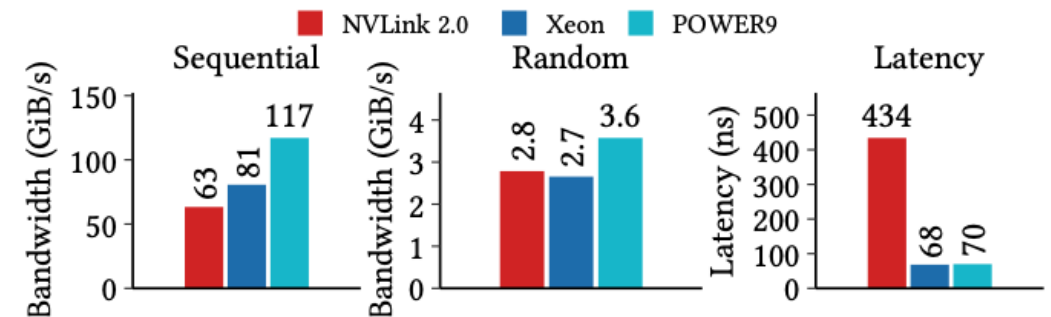
NVLink has much higher bandwidth than PCIe

NVLink has comparable bandwidth as CPU local memory

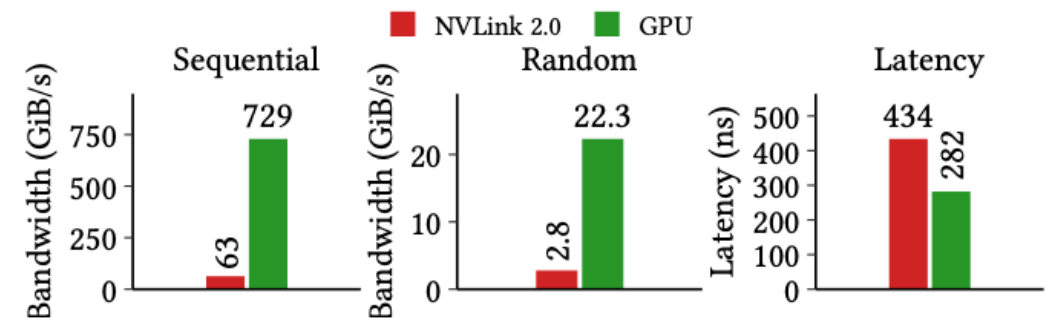
NVLink bandwidth has much lower bandwidth than GPU memory



(a) NVLink 2.0 vs. CPU & GPU Interconnects.



(b) NVLink 2.0 vs. CPU memory.



(c) NVLink 2.0 vs. GPU memory.

GPU Transfer Methods

Table 1: An overview of GPU transfer methods.

Method	Semantics	Level	Granularity	Memory
Pageable Copy	Push	SW	Chunk	Pageable
Staged Copy				
Dynamic Pinning				
Pinned Copy				
UM Prefetch	Pull	OS	Page	Unified
UM Migration				
Zero-Copy		HW	Byte	Pinned
Coherence				

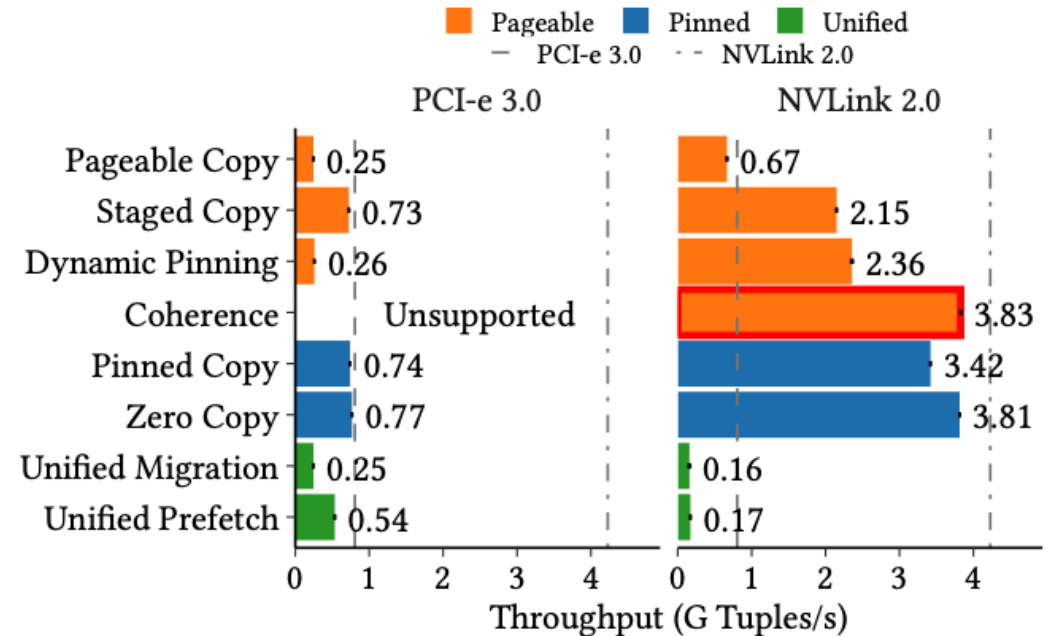


Figure 12: No-partitioning hash join using different transfer methods for PCI-e 3.0 and NVLink 2.0.

Pinned copy and **zero copy** can saturate PCIe bandwidth

Coherence can saturate NVLink bandwidth

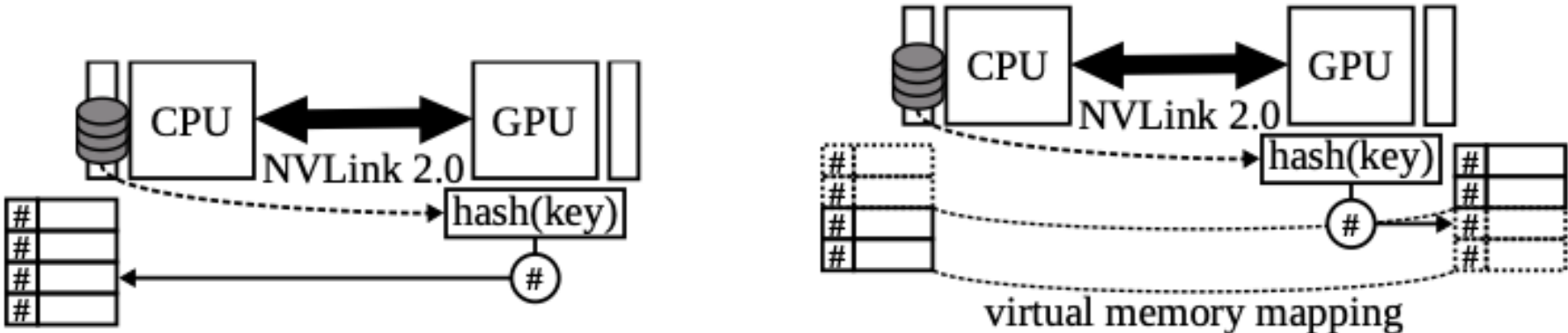
Non-Partitioned Hash Join Methods

Build phase: build the hash table using inner relation R

Probe phase: lookup hash table for each record in outer relation S

Hash Join – Build Phase

Build phase: build the hash table using inner relation R

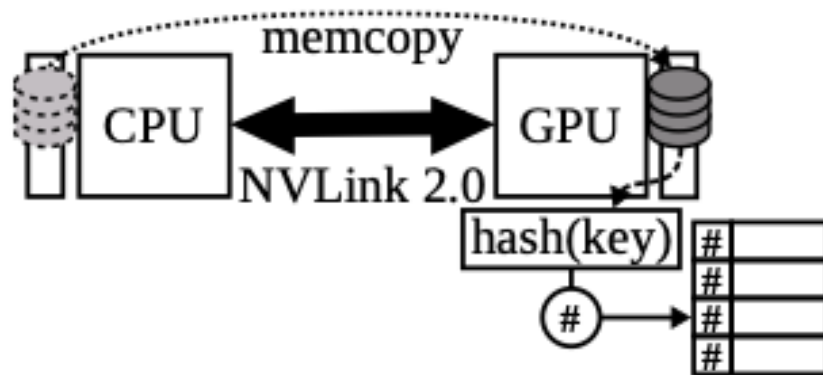


(a) Data and hash table in CPU memory.

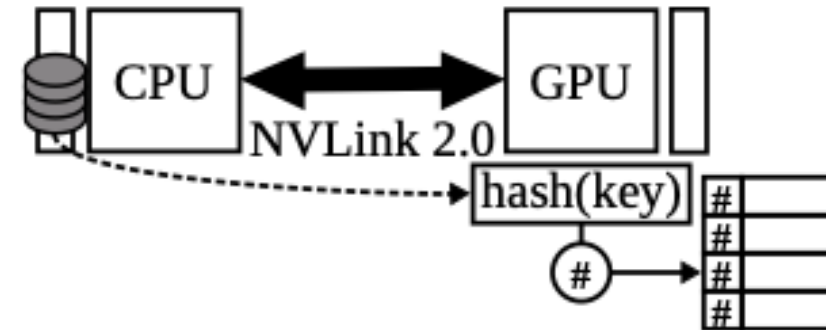
(b) Data in CPU memory and hash table spills from GPU memory into CPU memory.

Hash Join – Probe Phase

Probe phase: lookup hash table for each record in outer relation S

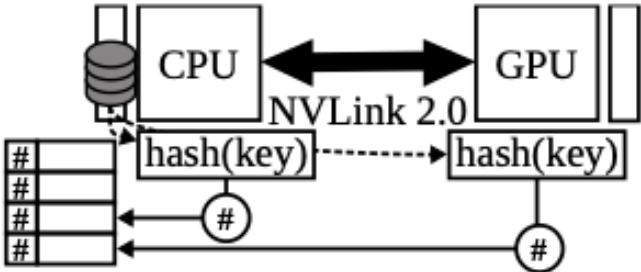


(a) Data and hash table in GPU memory.

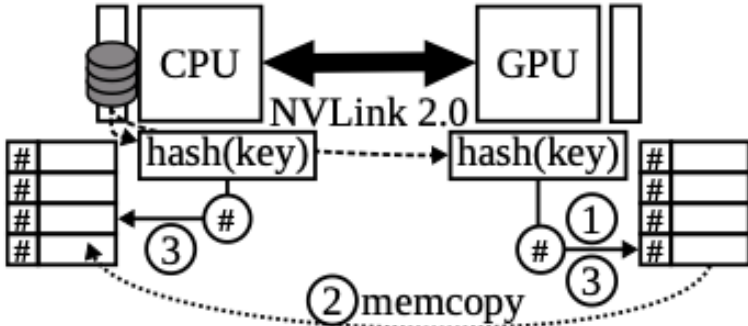


(b) Data in CPU memory and hash table in GPU memory.

Hash Join



(a) Cooperatively process join on CPU and GPU with hash table in CPU memory.



(b) Build hash table on GPU, copy the hash table to processor-local memories, and then cooperatively probe on CPU and GPU.

This **hybrid design** subsumes the previous designs in the paper

- Dynamically schedule tasks to both CPU and GPU

Hash Table Locality

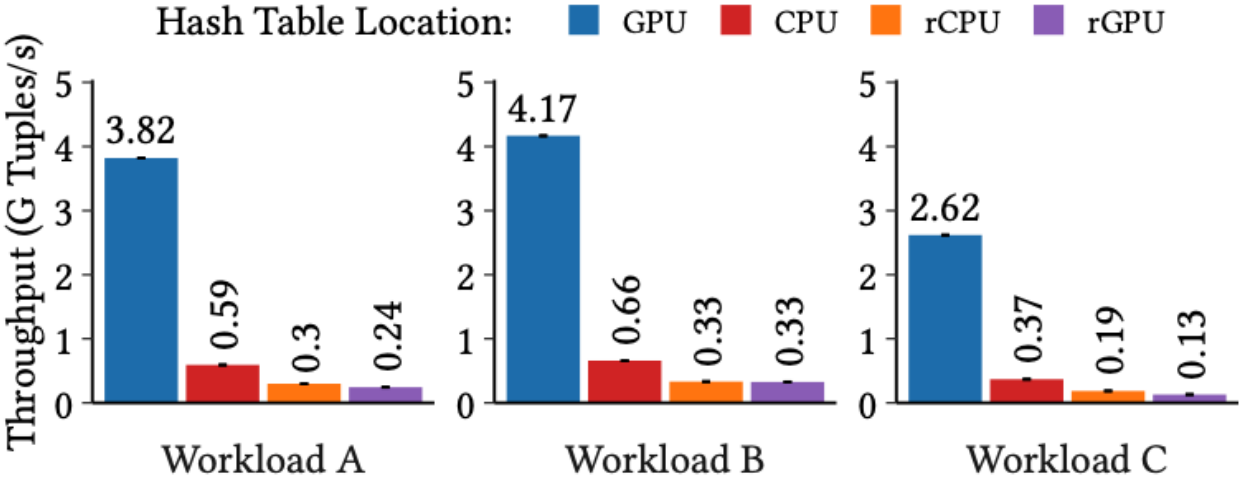


Table 2: Workload Overview.

Property	A (from [10])	B	C (from [54])
key / payload	8 / 8 bytes	8 / 8 bytes	4 / 4 bytes
cardinality of R	2^{27} tuples	2^{18} tuples	$1024 \cdot 10^6$ tuples
cardinality of S	2^{31} tuples	2^{31} tuples	$1024 \cdot 10^6$ tuples
total size of R	2 GiB	4 MiB	7.6 GiB
total size of S	32 GiB	32 GiB	7.6 GiB

Figure 14: Join performance of the GPU when the hash table is located on different processors, increasing the number of interconnect hops from 0 to 3.

Best performance achieved when the hash table is in GPU memory

Scaling Data Size in TPC-H Q6

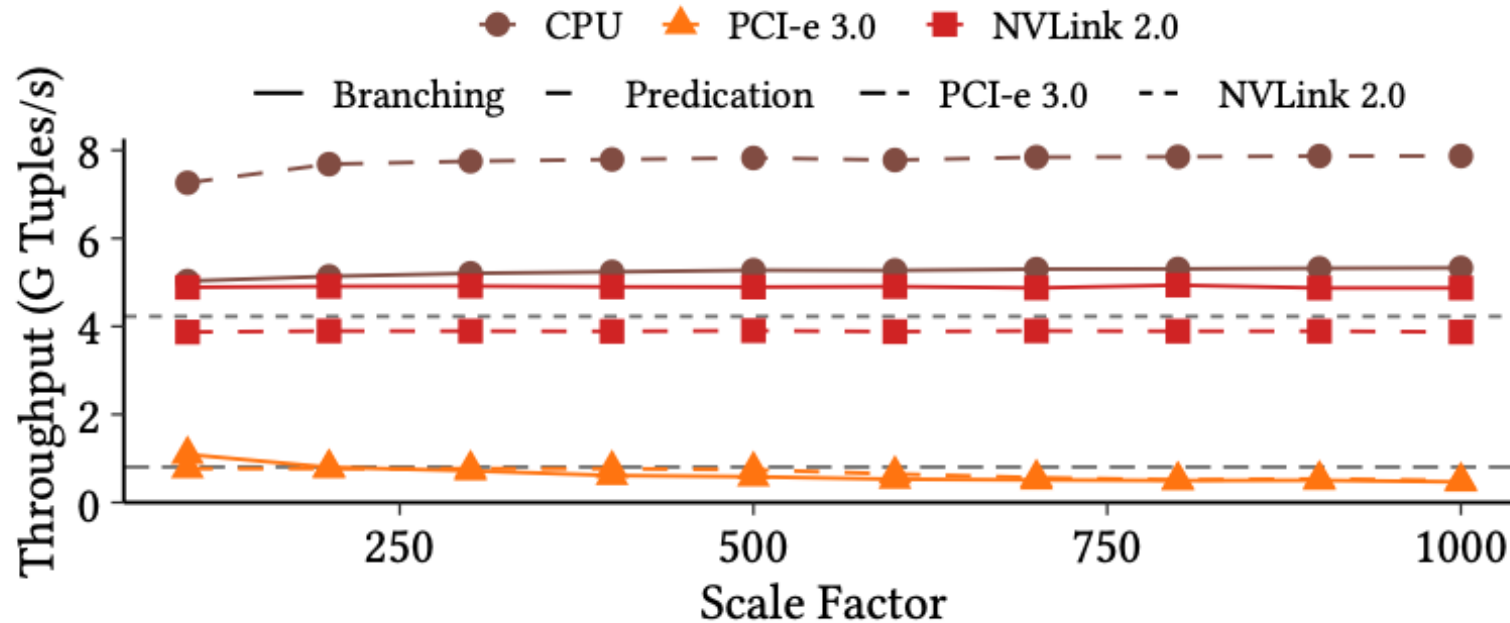


Figure 15: Scaling the data size of TPC-H query 6.

TPC-H Q6 contains a simple scan + aggregation with no join
Running the query on CPU leads to the highest performance

Scaling the Probe Side Relation

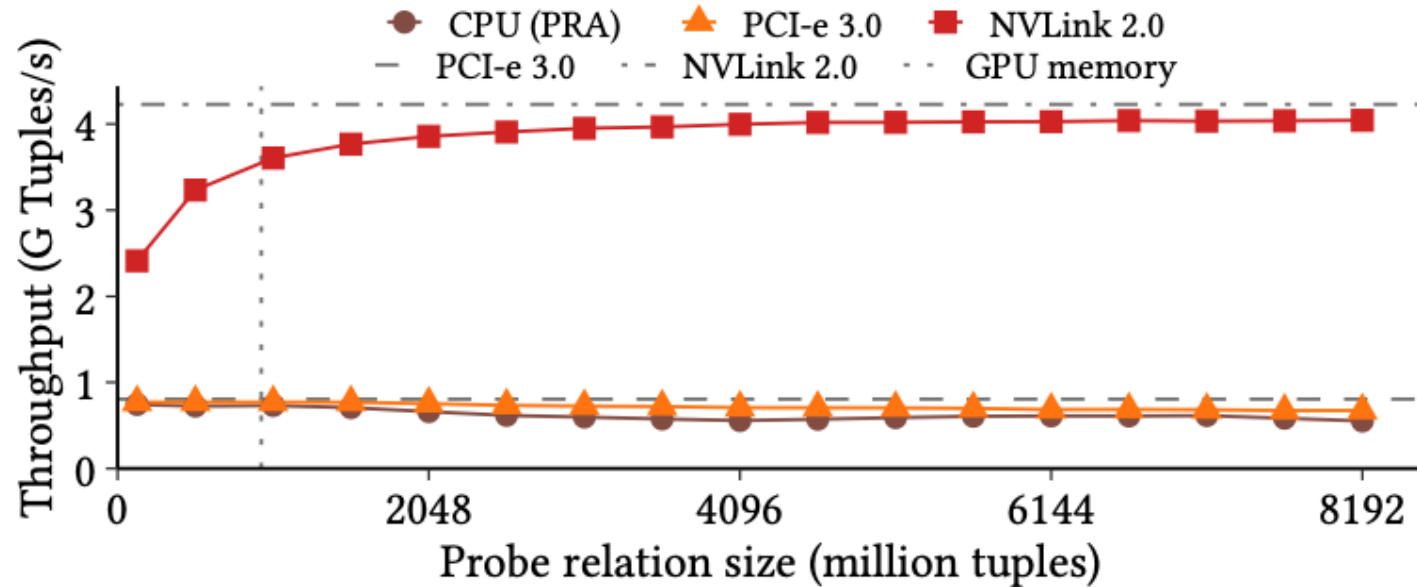


Figure 16: Scaling the probe-side relation.

NVLink is faster than both PCIe and CPU only

Scaling the Build Side Relation

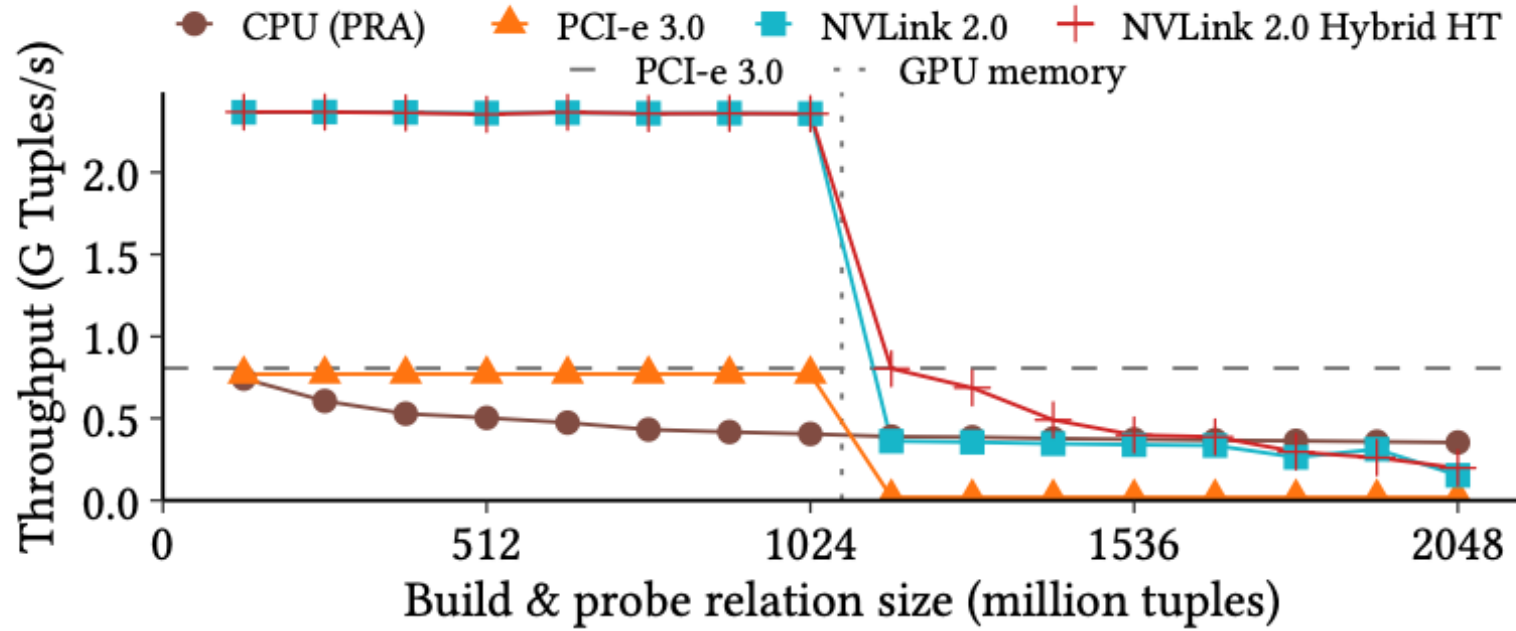


Figure 17: Scaling the build-side relation.

Performance drops when the hash table does not fit in GPU memory

Discussion

	Crystal	NVLink
Query Type	SPJA analytical queries	Non-partitioned hash join
Execution Model	Data fits in GPU memory	Coprocessor
Interconnect	PCIe 3.0	NVLink 2.0

Research question: How to maximize GPU database performance with different interconnect technology?

