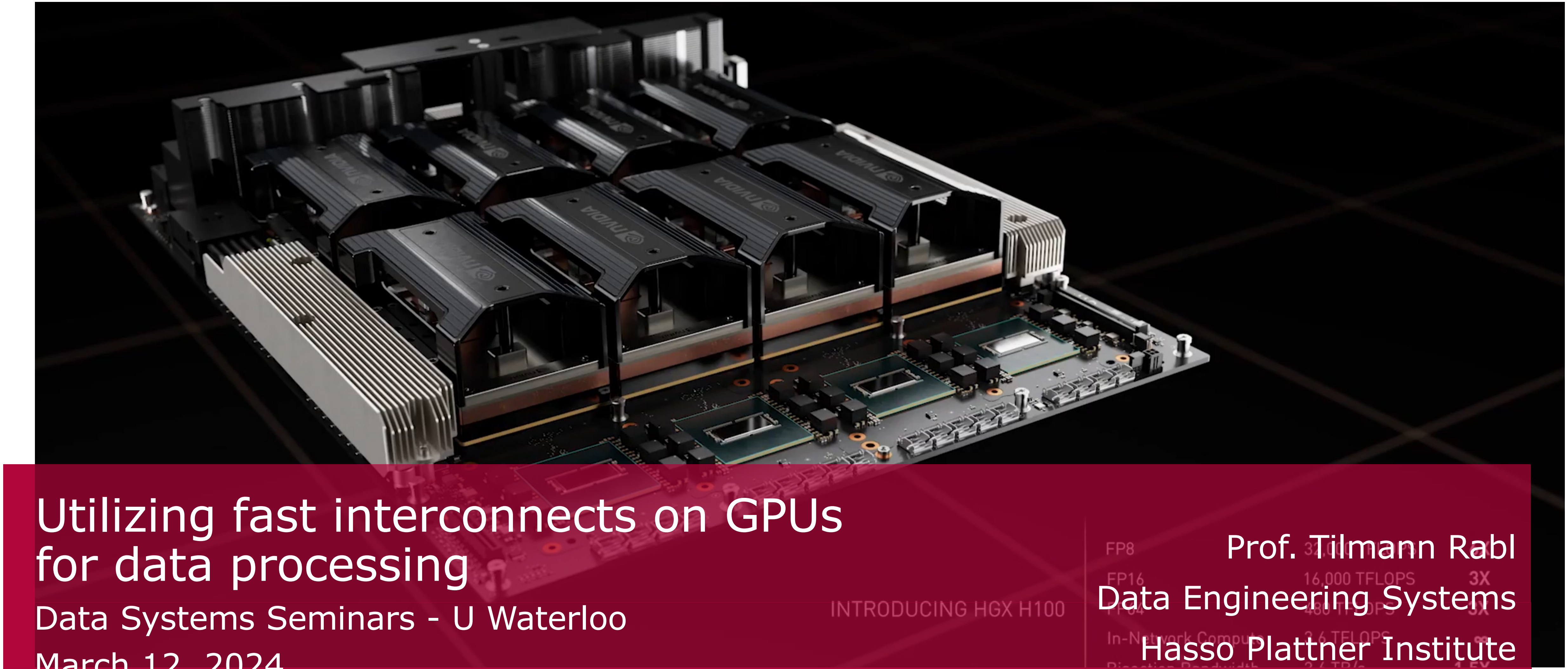
DES DATA ENGINEERING SYSTEMS GROUP

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam



Utilizing fast interconnects on GPUs for data processing

Data Systems Seminars - U Waterloo

March 12, 2024

Prof. Tilmann Rabl
Data Engineering Systems
Hasso Plattner Institute

Who am I?

- Until 2011: PhD in CS at University of Passau
  - Distributed databases
- Until 2015: Postdoc at University of Toronto
  - Big data systems / benchmarking
- Until 2019
  - Visiting Professor & Research Director at DIMA group, TU Berlin
  - Deputy Director of Department IAM at DFKI
  - Scientific Coordinator of the Berlin Big Data Center
- Since Mai 2019
  - Professor for Data Engineering Systems, Digital Engineering Faculty, HPI, U Potsdam

- Other activities
  - HPI Ombudsperson
  - Director of HPI Data Center

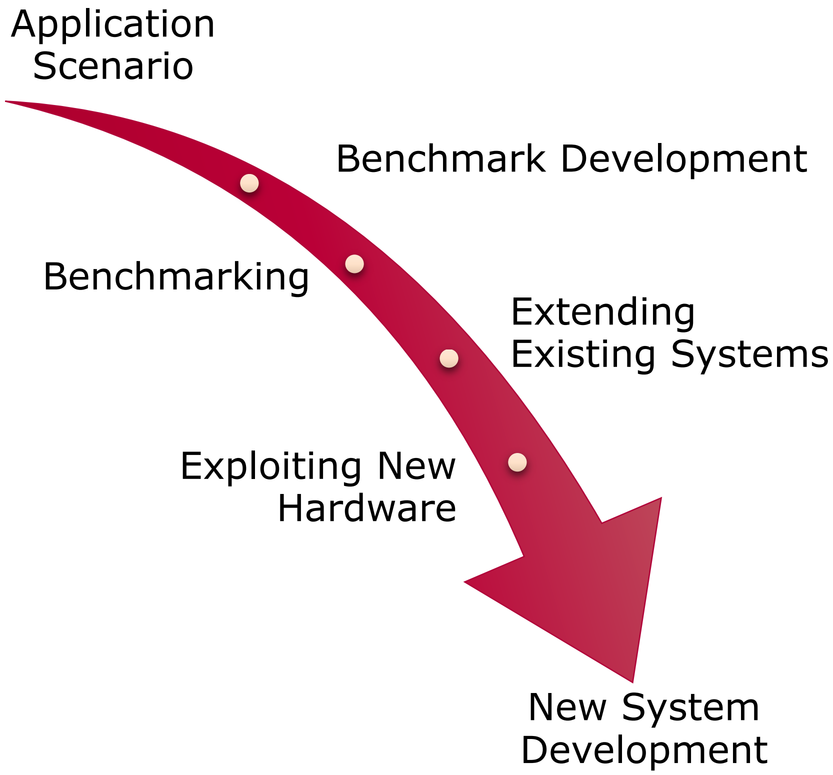
Hasso Plattner Institute

- Computer Science Institute in Potsdam, Germany
- B.Sc. and M.Sc. degree programs
- Department of University of Potsdam

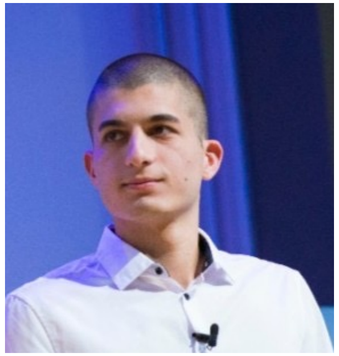
Research Topics

- Database Systems (on Modern Hardware)
  - ICDE 21, PVLDB 21, SIGMOD 22, SIGMOD 23
- Stream Processing / Real-time Analytics
  - PVLDB 20, TODS 21, SIGMOD 22, EDBT 23
- Machine Learning Systems
  - SIGMOD 20, ICDE 21, EDBT 22, EDBT 23
- Benchmarking
  - TPCTC 21, SIGMOD 21, PVLDB 22, SIGMOD 23

Research Approach



This work



- Clemens Lutz
- Tobias Maltenberger
- Ivan Ilic

This Talk

1. Quick 101 on data processing on GPU

2. Scalable Joins on a single GPU

3. Mult-GPU Sorting

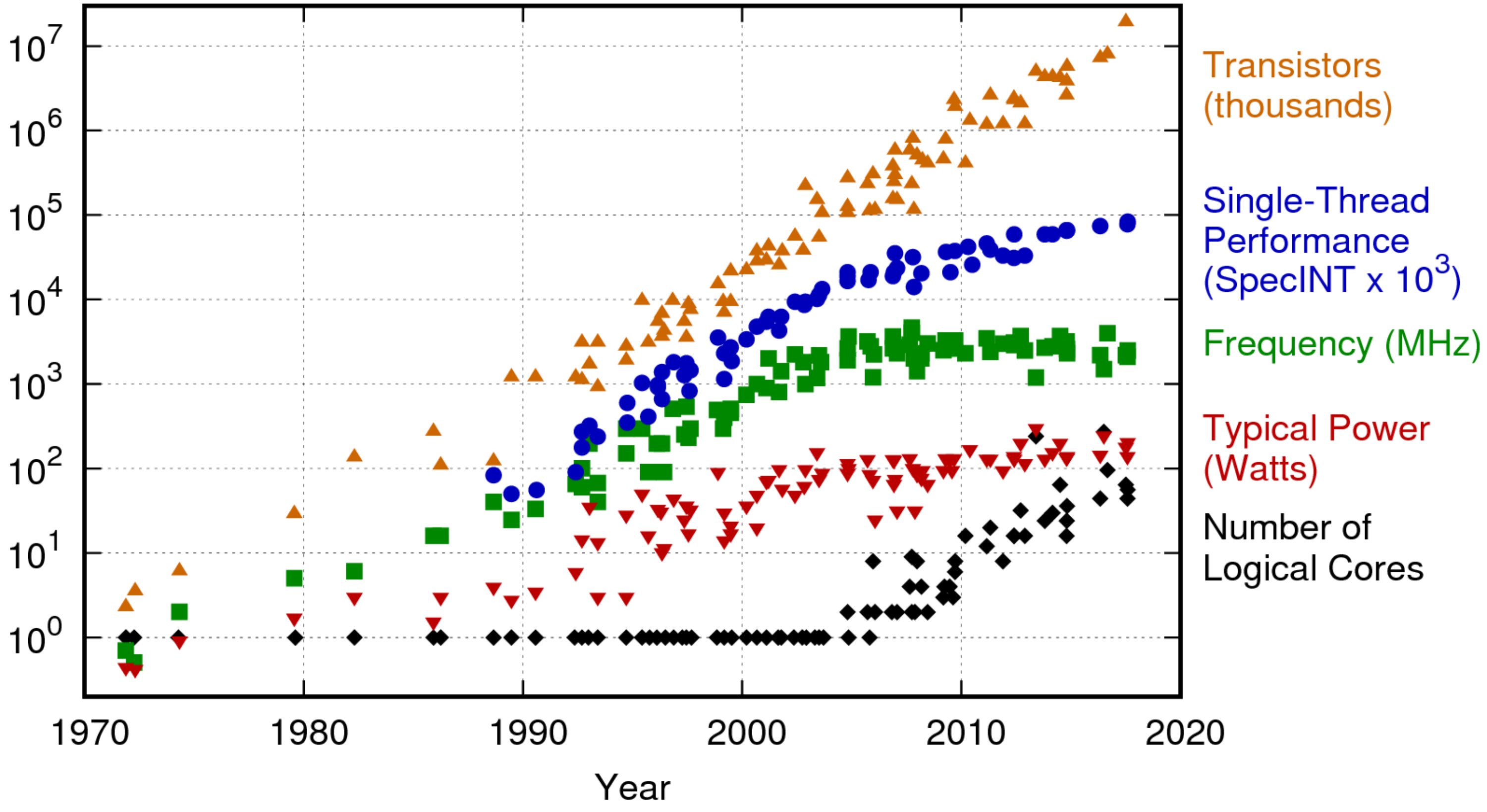
Big Fast Data Analysis

- Data is growing
- Messages, tweets, social networks (statuses, check-ins, shared content), blogs, click streams, various logs, …
- Everyone is interested!
- **The value of data is decreasing with its age!**

The Hardware Challenge

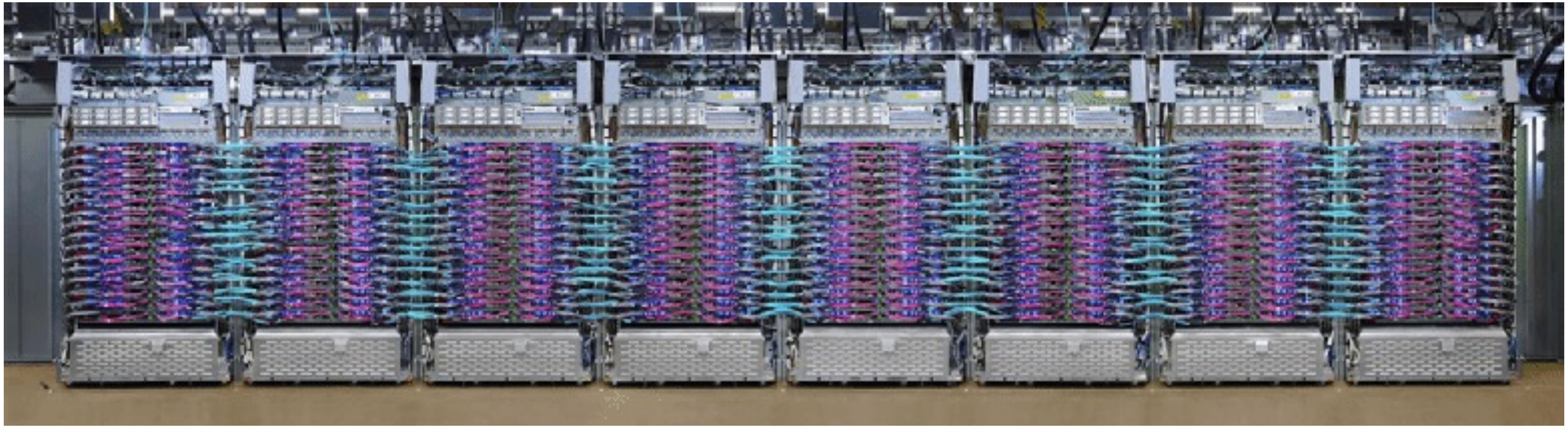
- Hardware performance is not simply increasing anymore
- Single thread -> multi thread
- Multi core -> specialization
- It is getting harder to be efficient

42 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 2010-2017 by K. Rupp

Hardware – Industry Trend

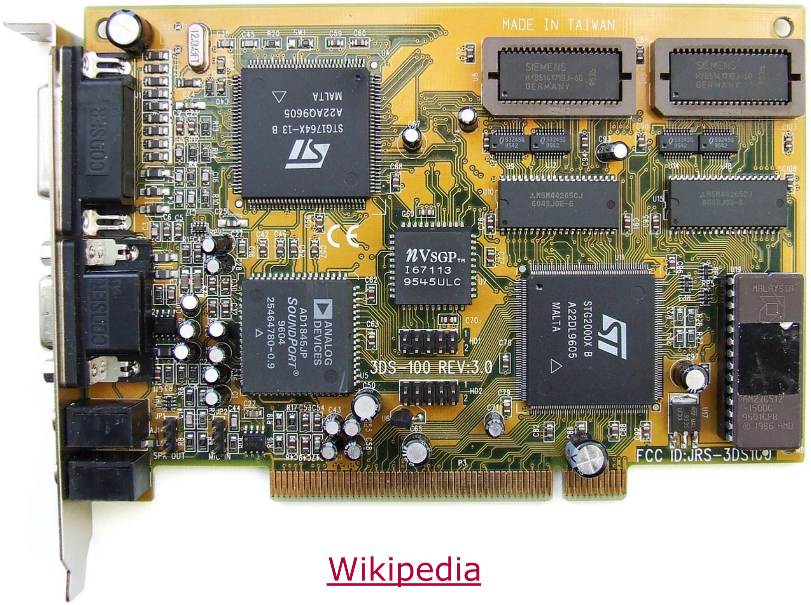


- TPU cloud, tight coupling, scale up, special hardware
- “Only way to meet growing compute demands” - Amin Vahdat (Google)
- *We need to be hardware-conscious and make systems efficient - Tilmann*

GPU-accelerated Data Processing

Graphics Processing Unit (GPU)

- Highly specialized coprocessing chip
- Image rendering -> parallel computations
- Performance increased ~2.4x yearly
- During Moore's law CPU's performance increased 1.8x

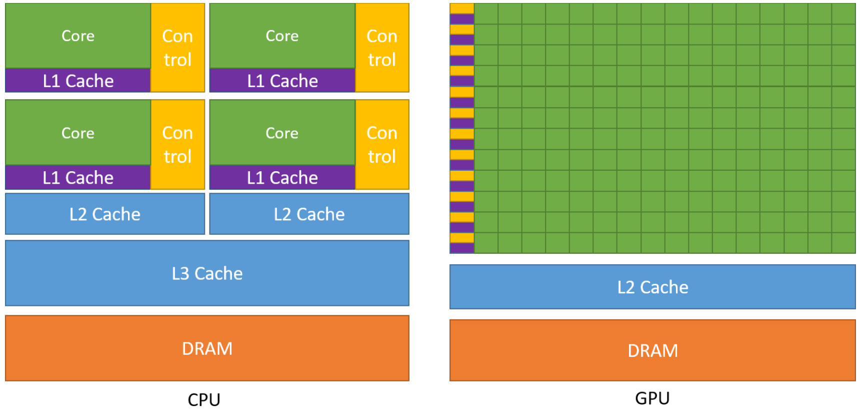


Wikipedia

- Programming model was based on DirectX
- Targeting workloads for CPUs required major adaptations

GPGPUs

- CPU cores != GPU cores -> Fundamental design differences
- CPU cores
  - Minimizing the latency of arithmetic operations
  - Latency-oriented design
  - Tens of cores
- GPU cores
  - Large number of FLOPS and memory accesses
  - Throughput-oriented design (1 555 – 3 000 GB/s)
  - Thousands of cores
- Massive parallelism
  - CPU bandwidth -> ~100s GB/s
  - GPU bandwidth -> ~1555 GB/s (A100) – 3000 GB/s (H100 - 2023)



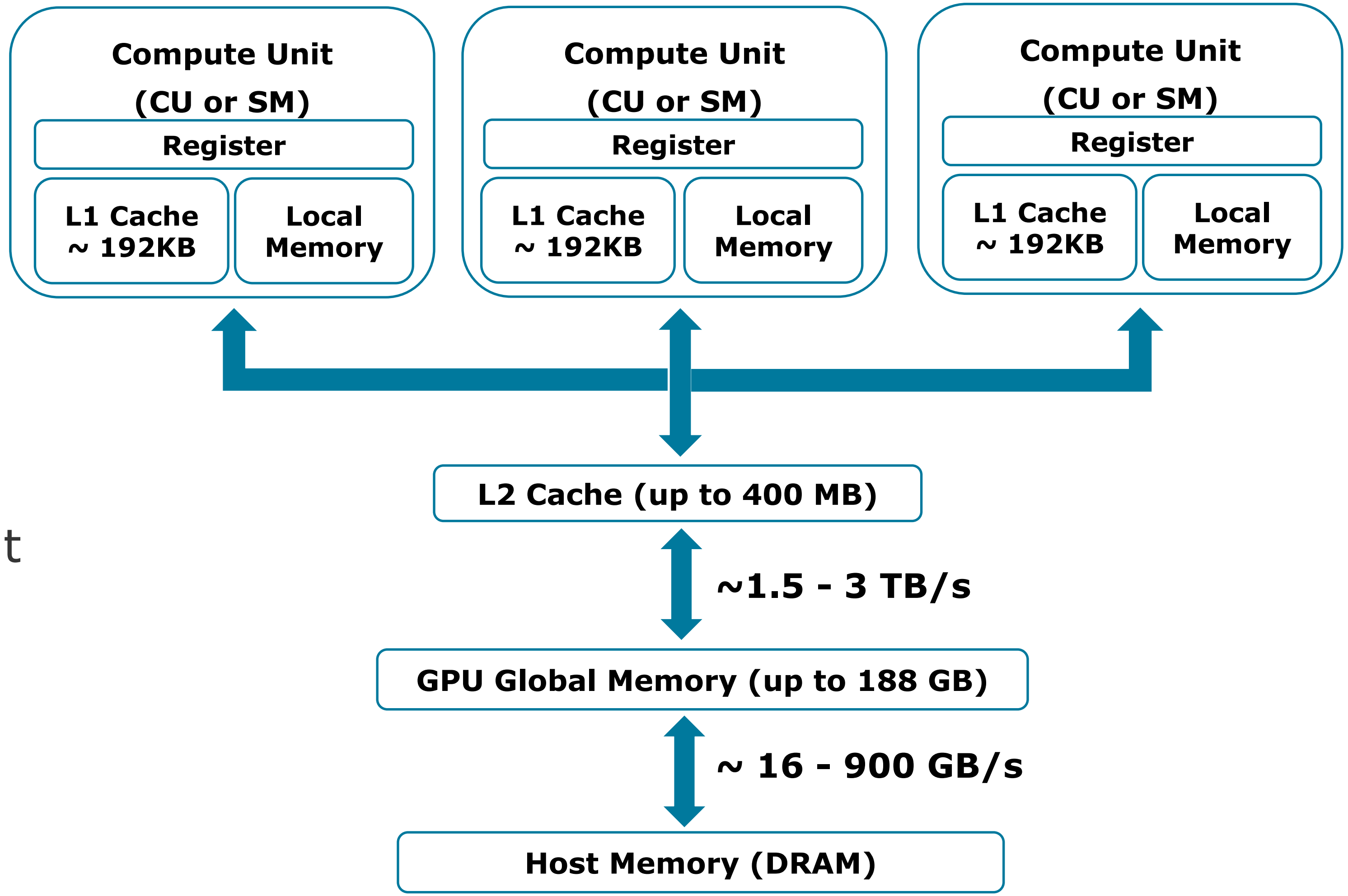
CUDA Programming Guide

GPU Architecture

- Compute Units
  - Computation cores
  - Register files
  - L1 cache
  - Shared memory

- Shared L2 cache
  - All compute units can access it

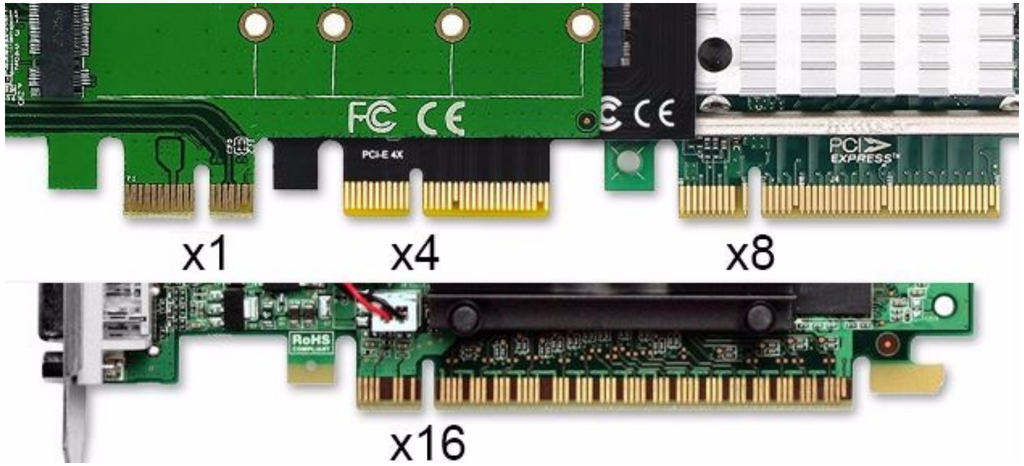
- Global Memory



GPU Interconnects

System Interconnects

- PCI Express
  - Connects CPU to storage, memory and coprocessors
  - Most commonly used CPU-GPU interconnect
  - Current systems use versions 3.0 and 4.0
  - x1, x4, x8, x16 lanes
  - Not necessarily all lanes are used
  - PCIe 3.0x16 ->16 GB/s, PCIe 4.0x16 -> 32 GB/s



PCI Express x1, x4, x8, x16

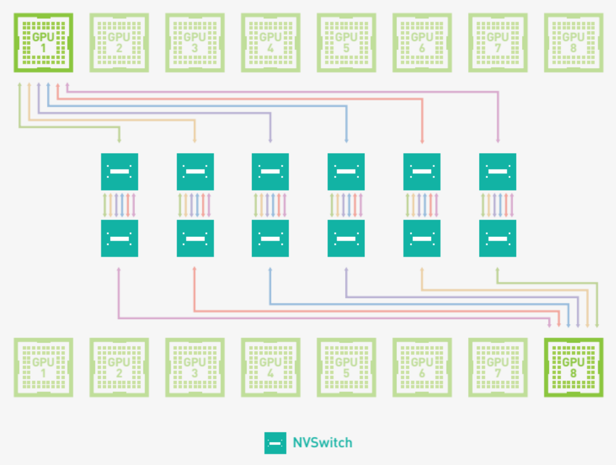
- From PCIe 5.0 also Compute Express Link (CXL)
  - Cache coherent access DtoH and HtoD (like NVLink)

GPU Interconnects

- NVIDIA NVLink
  - Mainly used as an inter-GPU interconnect
  - High bandwidth P2P data transfers
  - NVLink 2.0 -> 150 GB/s, NVLink 3.0 -> 300 GB/s
  - NVLink 4.0 -> 900 GB/s

NVLink Bridge 2.0 (left) and 3.0 (right)

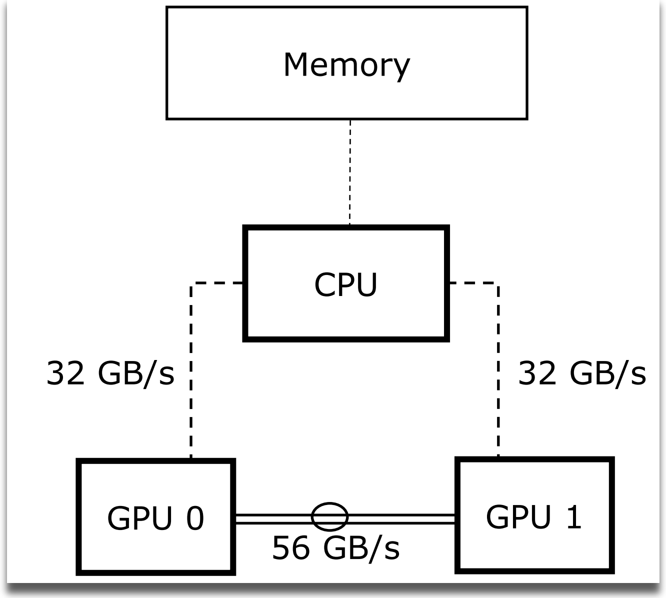
- NVIDIA NVSwitch
  - Switching element connecting up to 18 GPUs
  - Enables all-to-all GPU interconnectivity in a system
  - Non-blocking data transfers
  - GPU-GPU bandwidth -> up to 900 GB/s



NVSwitch

Single CPU – 2 GPU

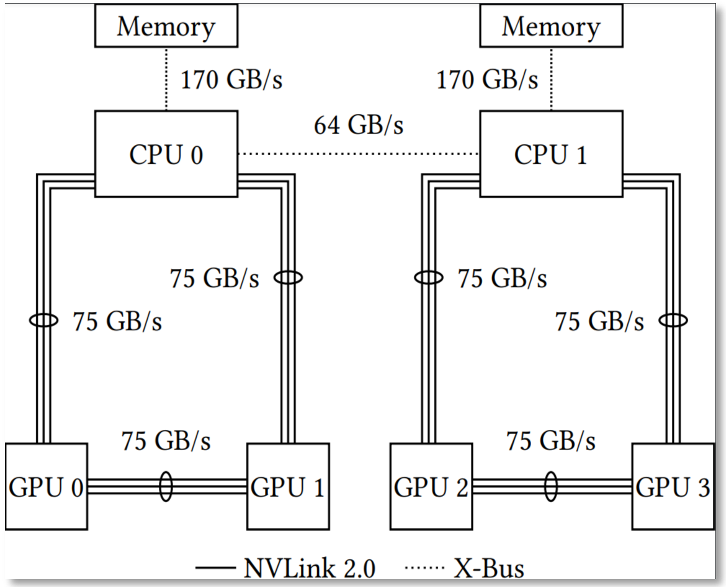
- Single socket CPU System
- Direct main memory access
- No NUMA effects
- Single/Shared PCIe x.0 CPU-GPU lane



- Thinkstation P620
  - CPU -> AMD Threadripper PRO 3995WX 64 Core
  - GPU -> 2x NVIDIA A5000 24GB NVLink 3.0 connected (56 GB/s)
  - CPU-GPU -> Single PCIe 4.0x16 lane (32 GB/s)

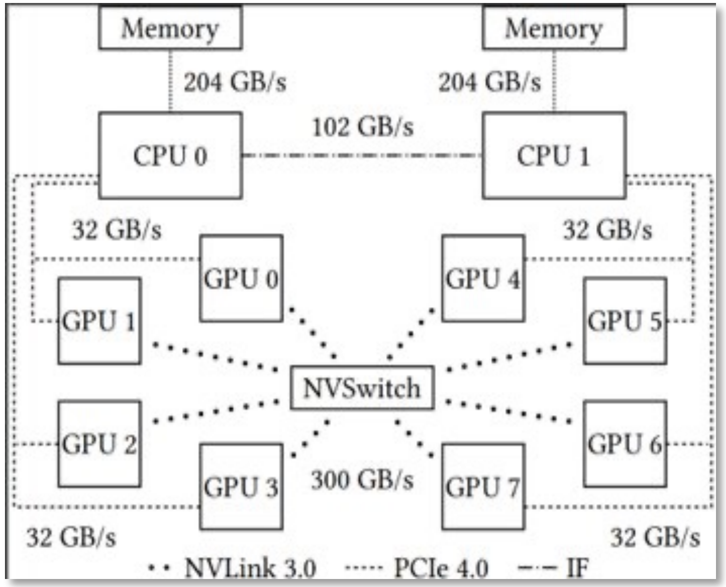
Dual CPU – 4 GPU

- Dual socket CPU System
  - CPUs interconnected via a proprietary interconnect
  - NUMA effects
- Dedicated main memory access
- **CPU-GPU connected via NVLink**
- **Equal bandwidth across all processors**
- IBM AC922
  - CPU -> 2x IBM Power 9
  - GPU -> 4 x NVIDIA V100 32GB NVLink 2.0 connected (75 GB/s)
  - CPU-GPU -> NVLink 2.0 interconnect (75 GB/s)
  - CPU-CPU -> IBM XBus 64 GB/s



Dual CPU – 8 GPU

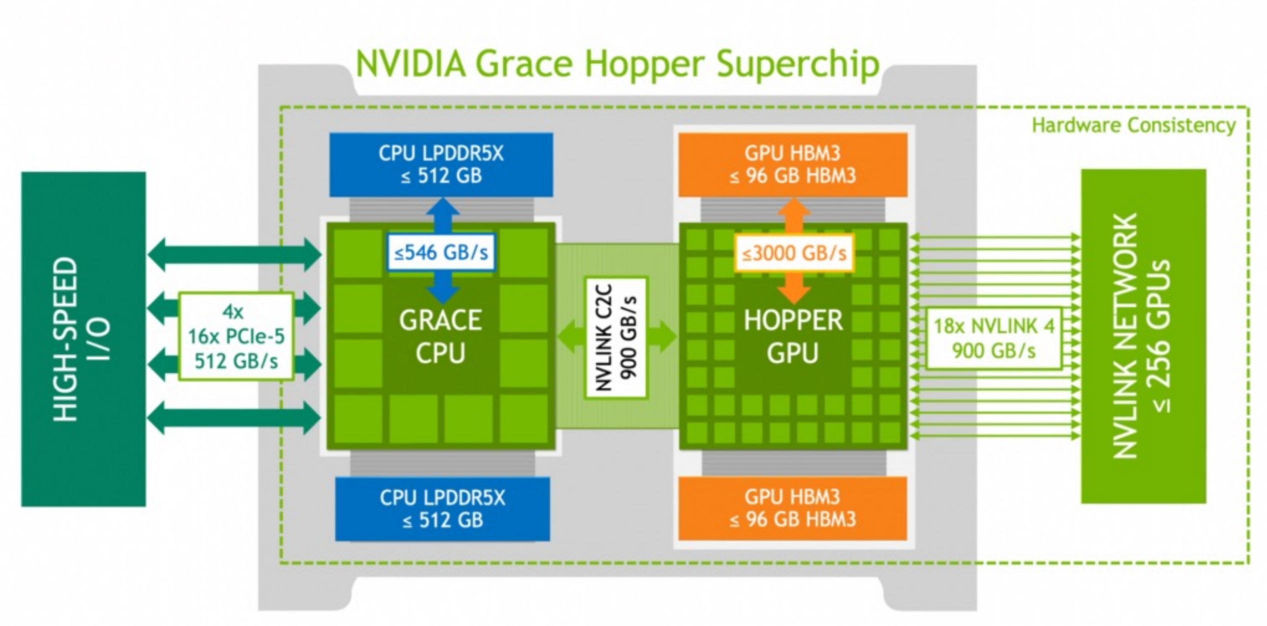
- Single/Shared PCIe 4.0 CPU-GPU lane
- Switching elements between GPUs
- All to all GPU communication
- Full GPU-GPU Bandwidth
- GPU-GPU bandwidth close to global memory bandwidth

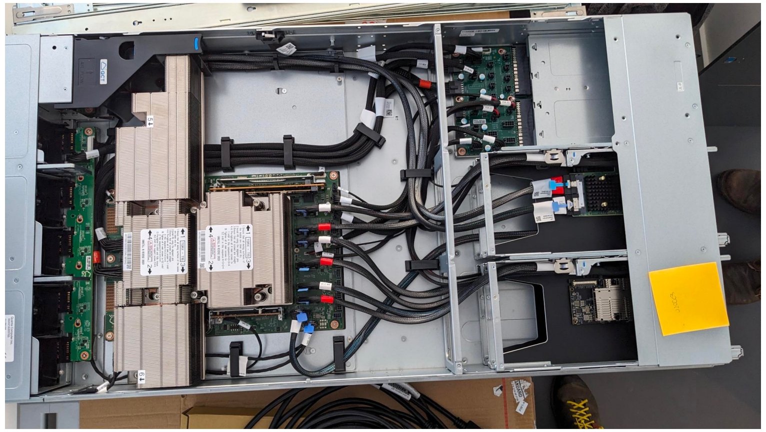


- DGX A100
  - CPU -> 2x AMD EPYC 7742
  - GPU -> 8x NVIDIA A100 40 GB NVLink 3.0 connected (300 GB/s)
  - CPU-GPU -> **Shared** PCIe 4.0x16 lane (32 GB/s)
  - CPU-CPU -> AMD Infinity Fabric (102 GB/s)

NVIDIA Grace Hopper

- Grace CPU
  - ARM
  - 72 Cores
  - 480 GB DDR5x
  - 64x PCIe 5 lanes
- Hopper GPU
  - 96 GB HBM3
- NVLink 4.0
  - 900 GB/s





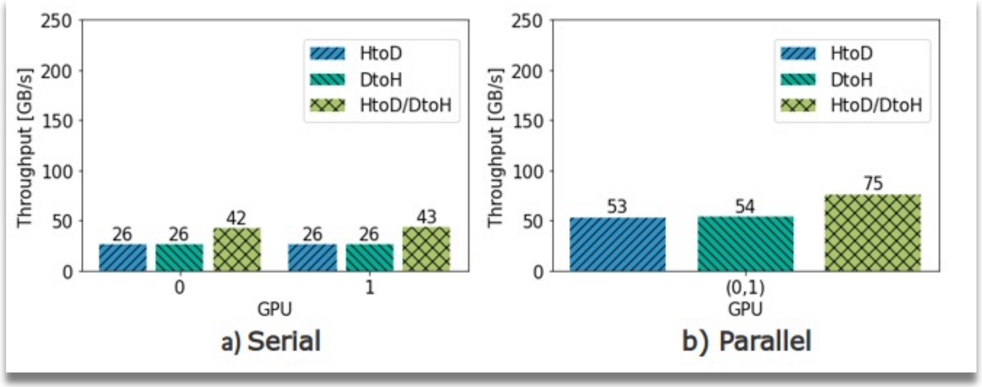
Data Transfers – ThinkStation P620

- Single PCIe 4.0 channel per GPU enables efficient CPU-GPU parallel transfers

- Performance drops for bi-directional CPU-GPU transfers

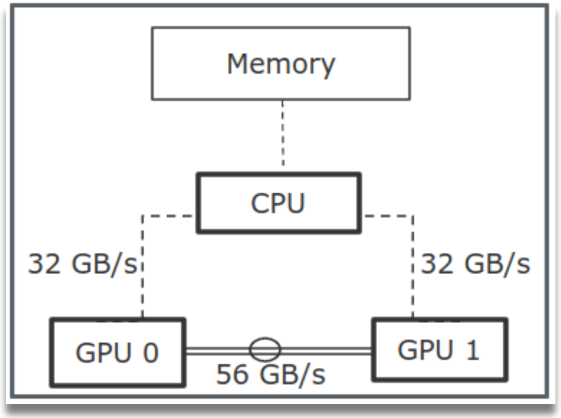
- Direct communication between the two GPUs

- 2x performance discrepancy between CPU-GPU and P2P data transfers



a) Serial

b) Parallel



Memory

CPU

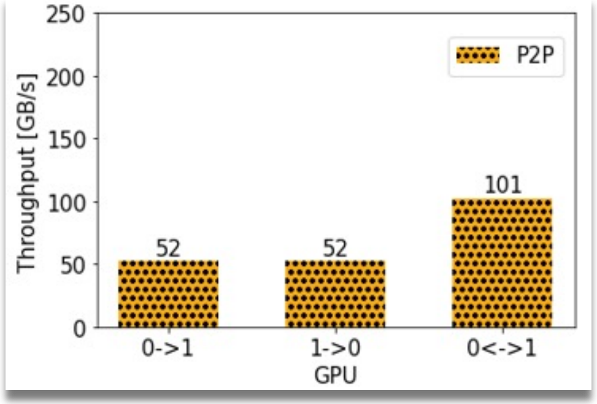
32 GB/s

32 GB/s

GPU 0

GPU 1

56 GB/s



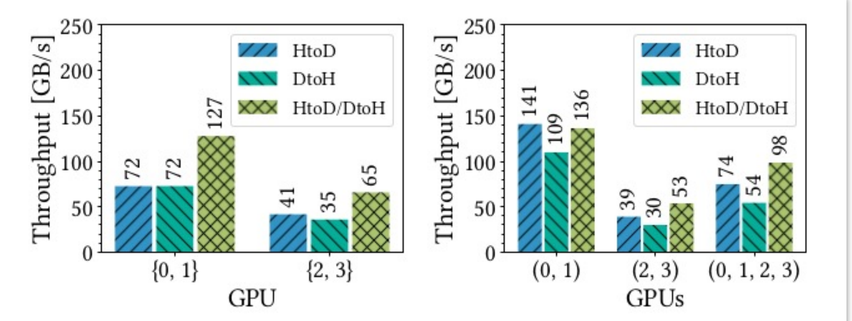
Data Transfers - AC 922

- Significant performance drops due to NUMA effects

- CPU-CPU interconnect is the main bottleneck

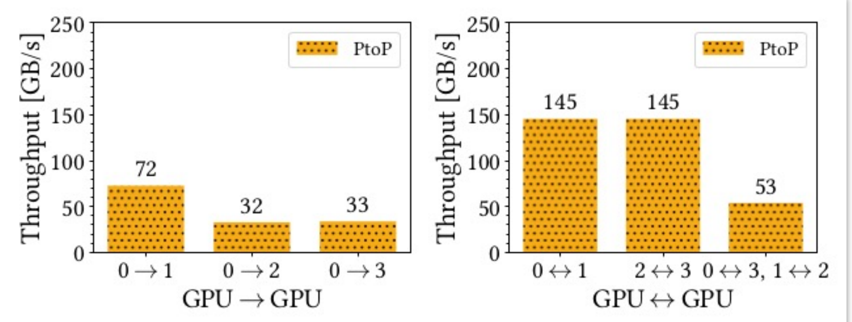
- Using two GPUs often faster than all four

- No performance discrepancy between CPU-GPU and P2P data transfers



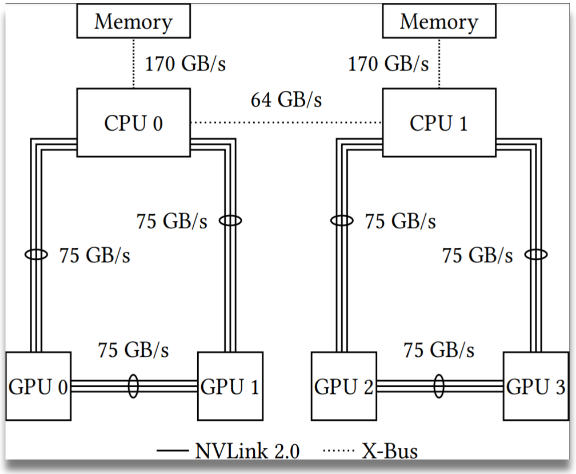
(a) Serial

(b) Parallel



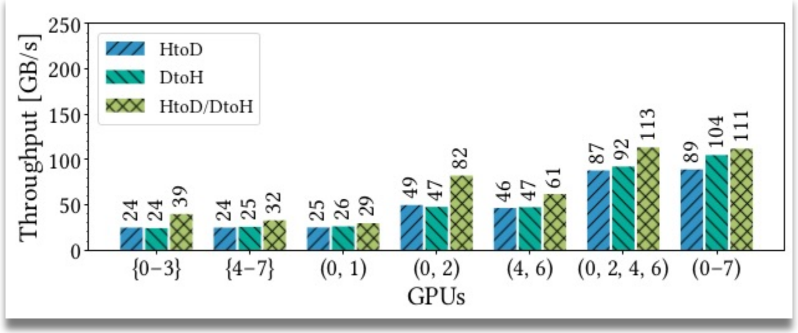
(a) Serial

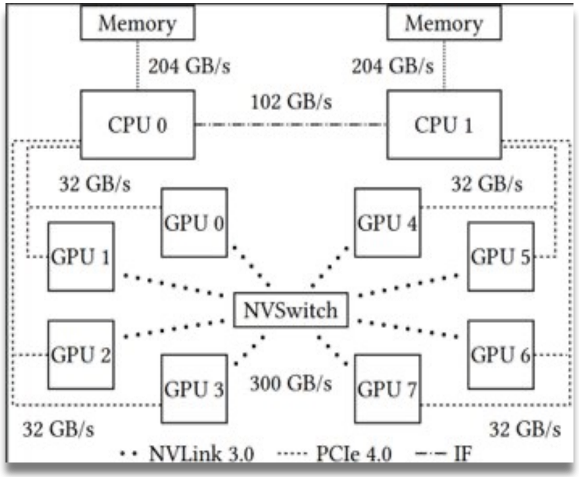
(b) Parallel

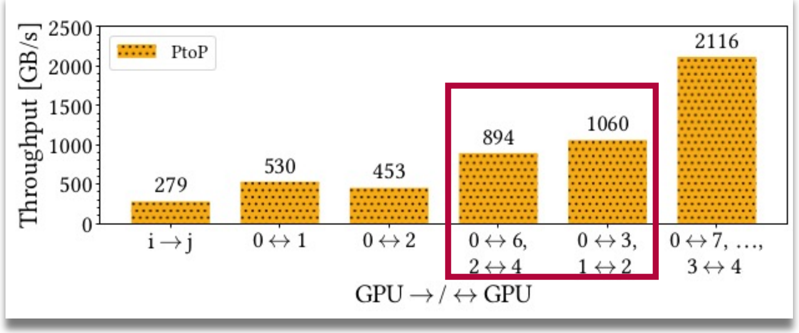


Data Transfers - DGX A100

- Consistent P2P bandwidth
- Main memory transfers limited by the PCIe interconnect
- Parallel transfers limited by shared PCIe bus
- Despite having NVSwitch, parallel P2P transfers are faster between co-located GPUs
- >10x performance discrepancy between CPU-GPU and P2P data transfers

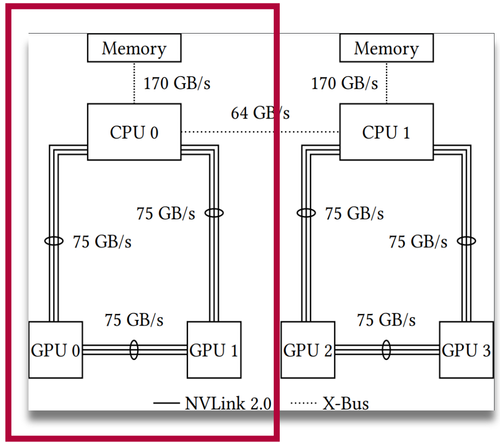


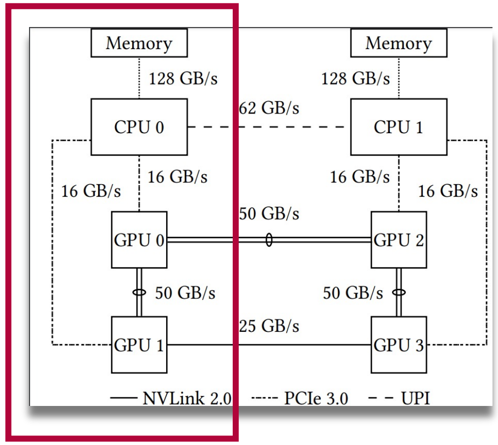




Topology-aware Algorithm Design

- Pre-allocate memory and reduce data transfers
- Organize the P2P communication
- Maximize the bandwidth according to the system's topology
- When data is located on one socket, keep the computation on the same NUMA node





Scalable GPU-based Join

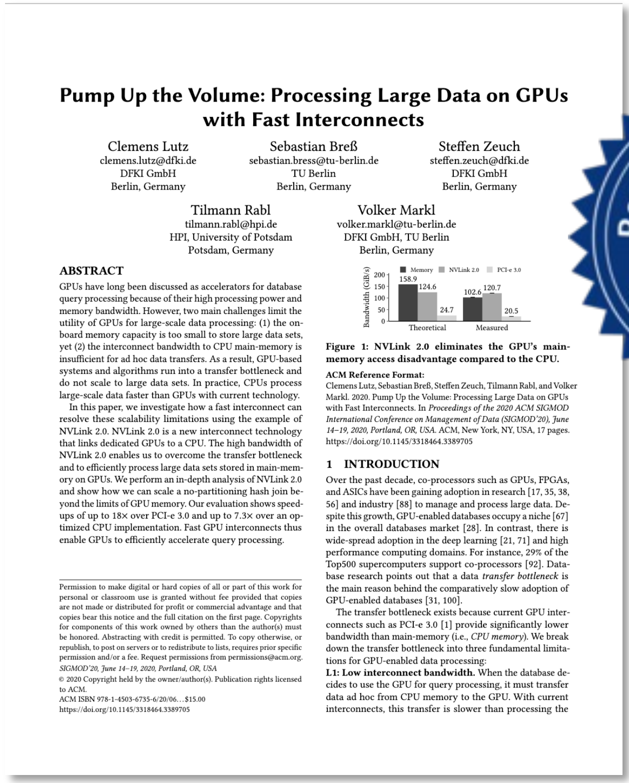
DES

Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects

Clemens Lutz, Sebastian Breß, Steffen Zeuch, Tilmann Rabl, Volker Markl

SIGMOD 2020 – Best Paper

Follow-up: Triton Join - SIGMOD 2022



Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects

Clemens Lutz
clemens.lutz@dfki.de
DFKI GmbH
Berlin, Germany

Sebastian Breß
sebastian.bress@tu-berlin.de
TU Berlin
Berlin, Germany

Steffen Zeuch
steffen.zeuch@dfki.de
DFKI GmbH
Berlin, Germany

Tilmann Rabl
tilmann.rabl@hpi.de
HPI, University of Potsdam
Potsdam, Germany

Volker Markl
volker.markl@tu-berlin.de
DFKI GmbH, TU Berlin
Berlin, Germany

ABSTRACT

GPUs have long been discussed as accelerators for database query processing because of their high processing power and memory bandwidth. However, two main challenges limit the utility of GPUs for large-scale data processing: (1) the onboard memory capacity is too small to store large data sets, yet (2) the interconnect bandwidth to CPU main-memory is insufficient for ad hoc data transfers. As a result, GPU-based systems and algorithms run into a transfer bottleneck and do not scale to large data sets. In practice, CPUs process large-scale data faster than GPUs with current technology.

In this paper, we investigate how a fast interconnect can resolve these scalability limitations using the example of NVLink 2.0. NVLink 2.0 is a new interconnect technology that links dedicated GPUs to a CPU. The high bandwidth of NVLink 2.0 enables us to overcome the transfer bottleneck and to efficiently process large data sets stored in main-memory on GPUs. We perform an in-depth analysis of NVLink 2.0 and show how we can scale a no-partitioning hash join beyond the limits of GPU memory. Our evaluation shows speedups of up to 18× over PCI-e 3.0 and up to 7.3× over an optimized CPU implementation. Fast GPU interconnects thus enable GPUs to efficiently accelerate query processing.

Figure 1: NVLink 2.0 eliminates the GPU's main-memory access disadvantage compared to the CPU.

ACM Reference Format:
Clemens Lutz, Sebastian Breß, Steffen Zeuch, Tilmann Rabl, and Volker Markl. 2020. Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3318464.3389705

1 INTRODUCTION

Over the past decade, co-processors such as GPUs, FPGAs, and ASICs have been gaining adoption in research [17, 35, 38, 56] and industry [88] to manage and process large data. Despite this growth, GPU-enabled databases occupy a niche [67] in the overall databases market [28]. In contrast, there is wide-spread adoption in the deep learning [21, 71] and high performance computing domains. For instance, 29% of the Top500 supercomputers support co-processors [92]. Database research points out that a data *transfer bottleneck* is the main reason behind the comparatively slow adoption of GPU-enabled databases [31, 100].

The transfer bottleneck exists because current GPU interconnects such as PCI-e 3.0 [1] provide significantly lower bandwidth than main-memory (i.e., *CPU memory*). We break down the transfer bottleneck into three fundamental limitations for GPU-enabled data processing:

L1: Low interconnect bandwidth. When the database decides to use the GPU for query processing, it must transfer data ad hoc from CPU memory to the GPU. With current interconnects, this transfer is slower than processing the

SIGMOD'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3389705

Results Reproduced
acm
V1.1

TU berlin

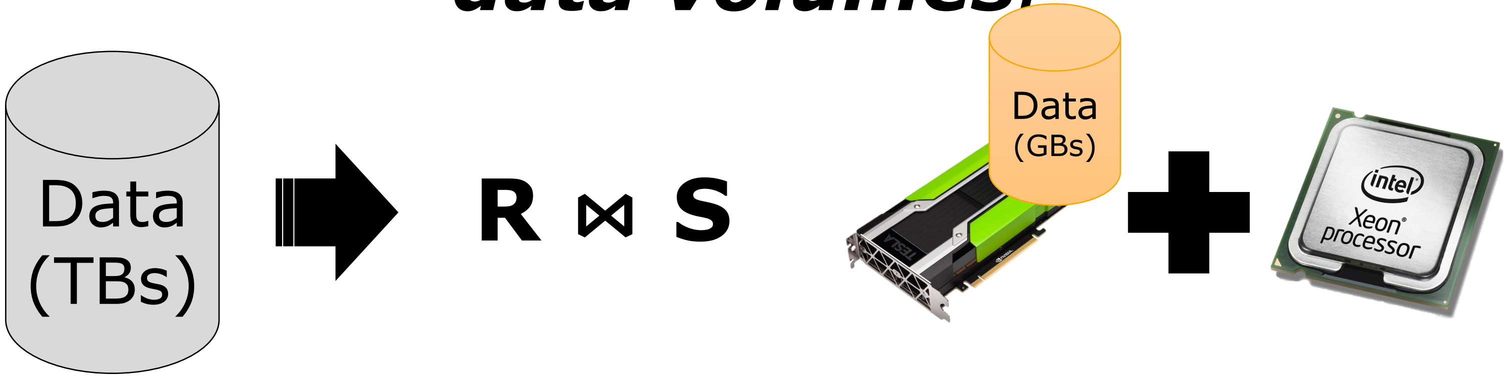
DFKI German Research Center for Artificial Intelligence

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

PORTLAND ACM SIGMOD +PODS 2020
ACM SIGMOD/PODS International Conference on Management of Data
June 14-19th, 2020
Portland, Oregon USA

Goal

Scale GPU-accelerated data management to ***arbitrary data volumes.***

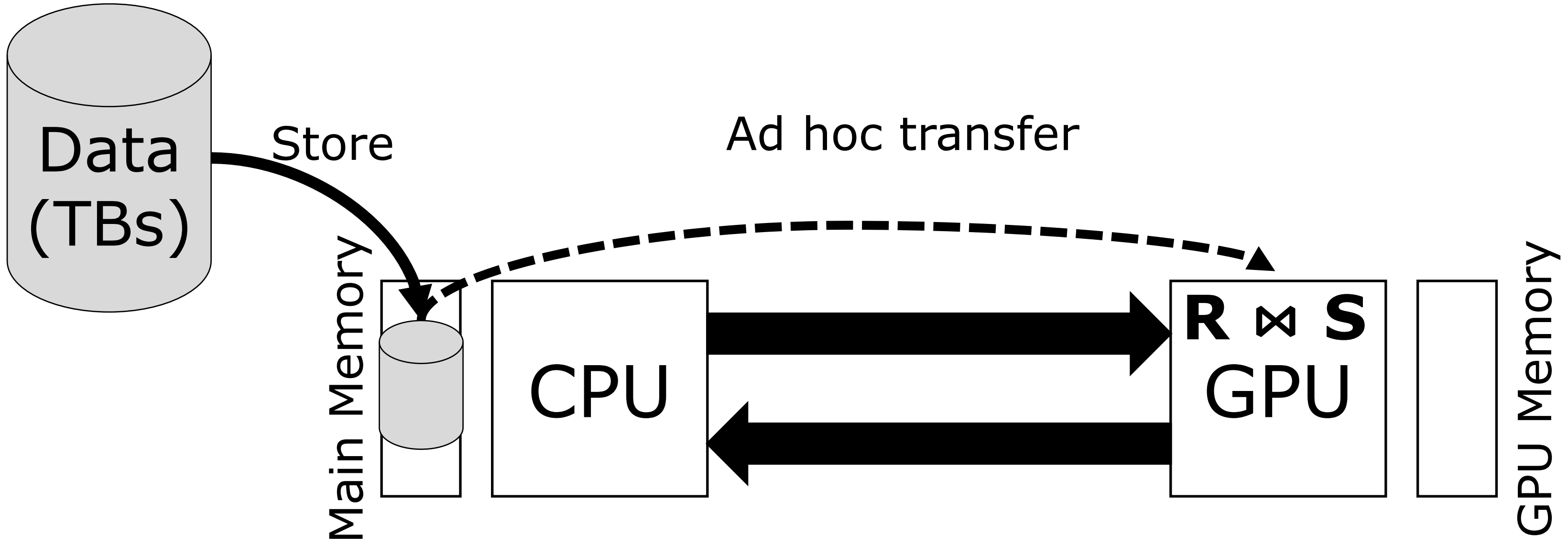


Data (TBs)

R ⋈ S

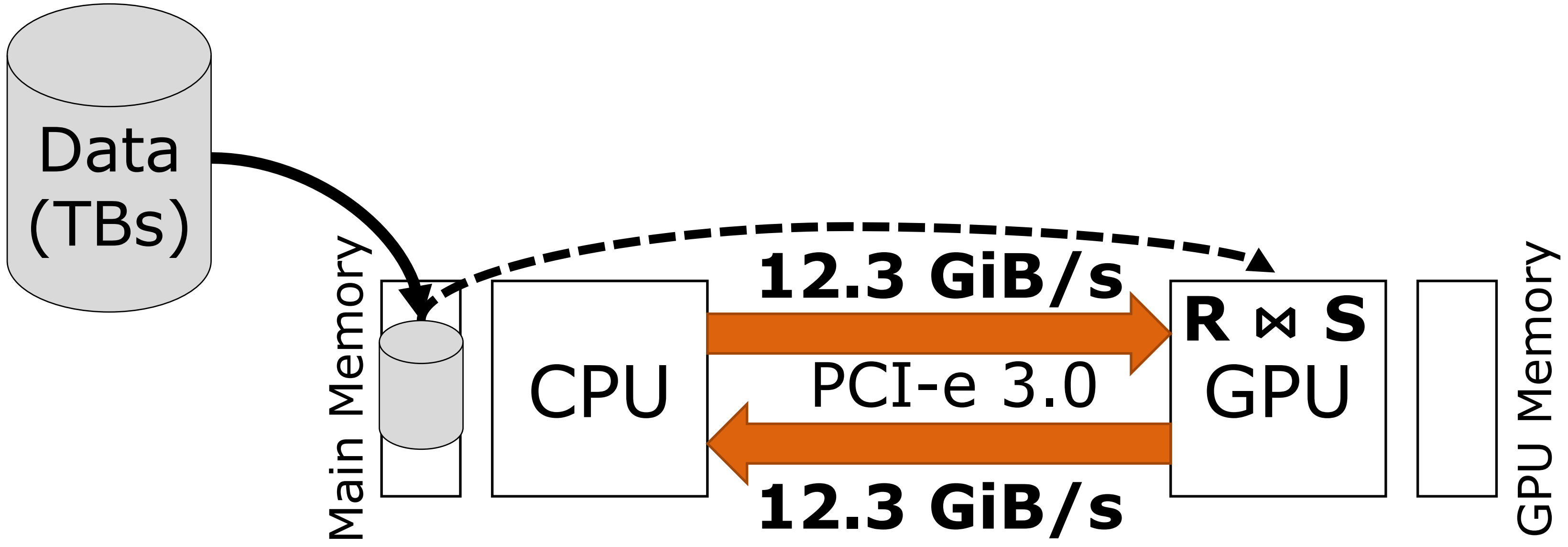
Data (GBs)

Problem: Transfer Bandwidth



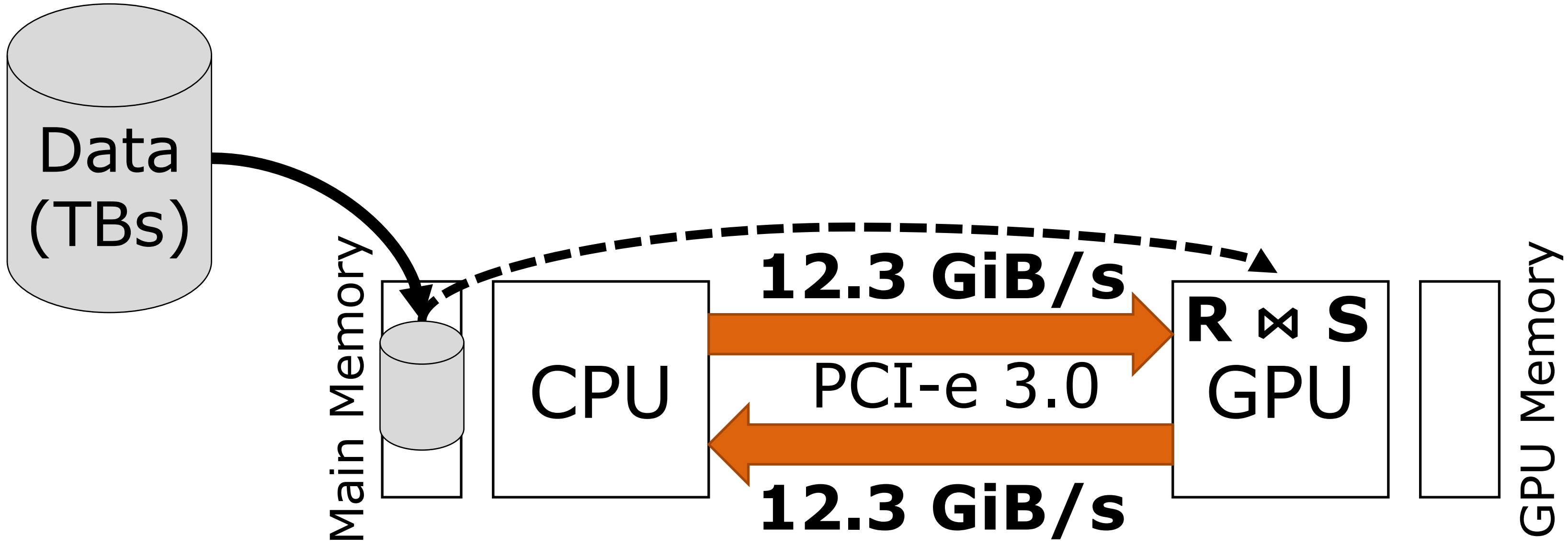
- Today’s GPU databases:
- Store data in main memory
- Perform data processing on GPU

Problem: Transfer Bandwidth



- Ad hoc data transfer over PCI-e 3.0
- GPU capable of much higher throughput

Problem: Transfer Bandwidth

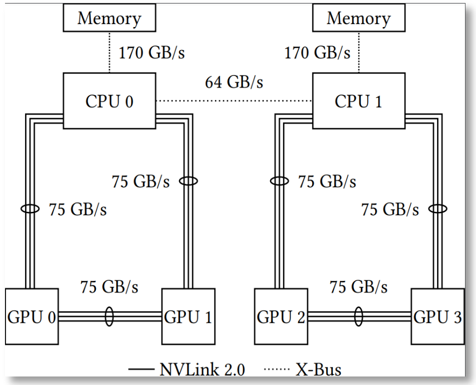


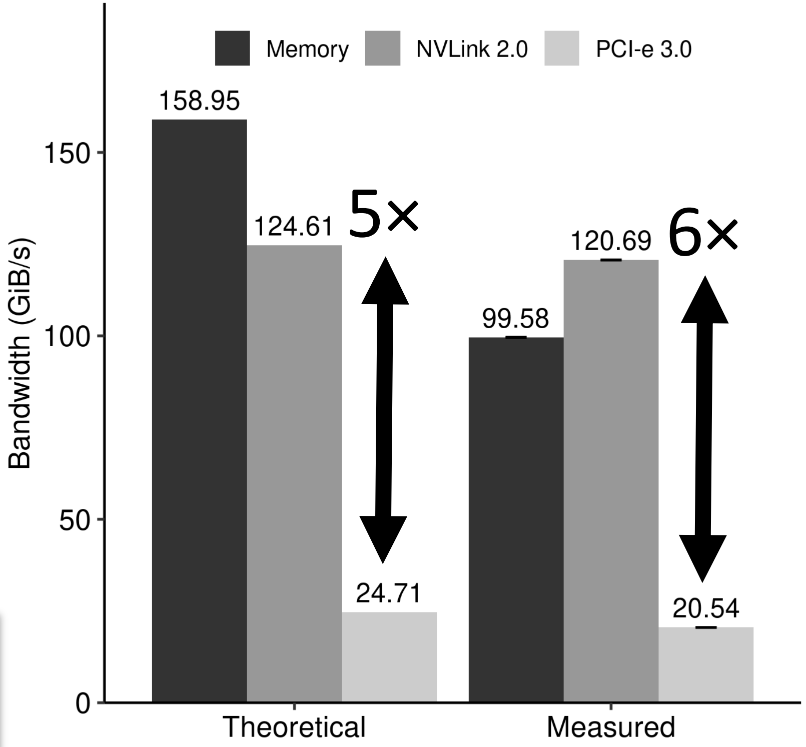
Interconnect bandwidth & GPU memory capacity limit scalability

"Transfer bottleneck"

Game Changer

- Fast interconnects
  - e.g., NVLink 2.0, Infinity Fabric, CXL
- High bandwidth (124 GB/s total)
- System-wide cache-coherence
  - data-dependent memory access
  - fine-grained CPU+GPU cooperation
- E.g., AC 922



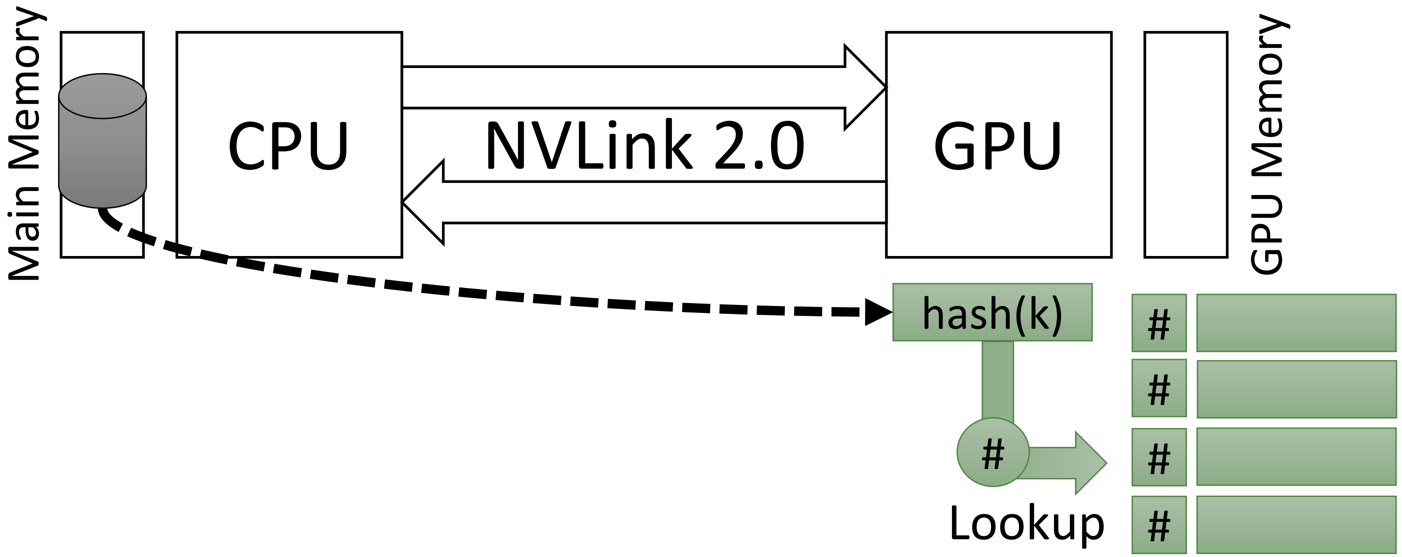


Solution

Hash Join

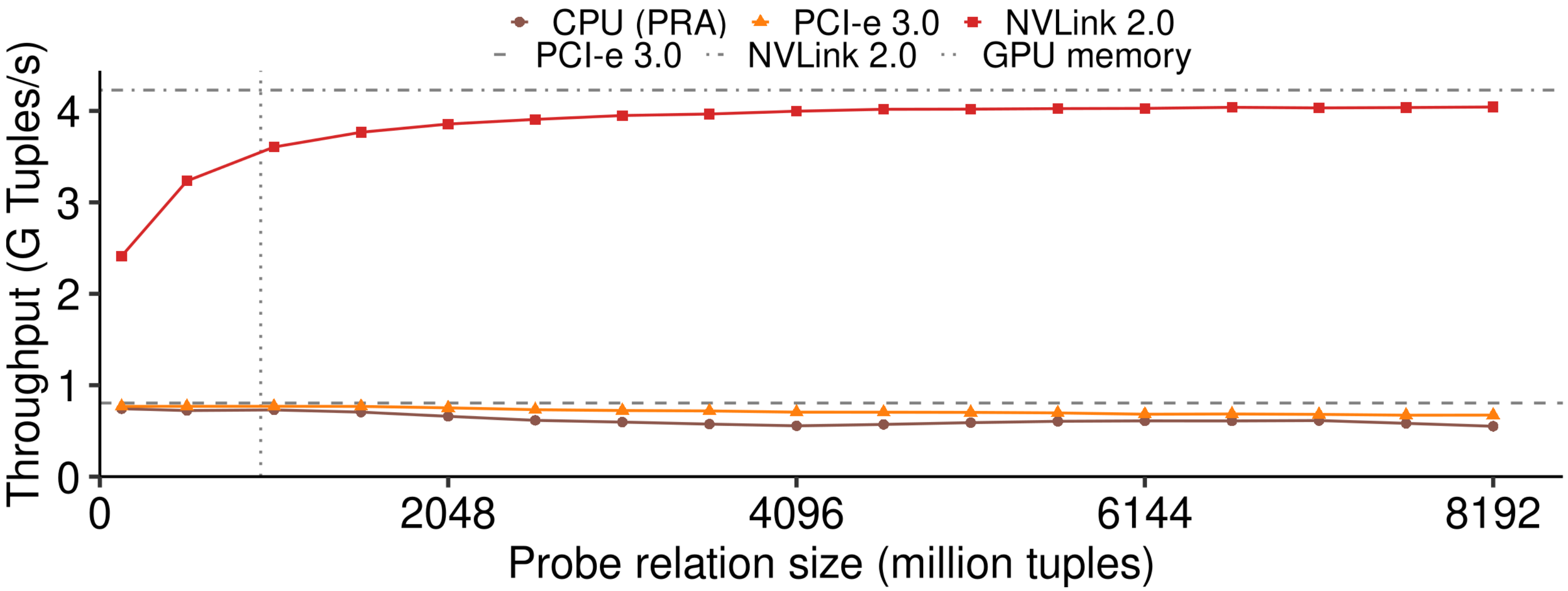
- Probe-side scaling
- Build-side scaling
- GPU+CPU cooperation

Probe-Side Scaling



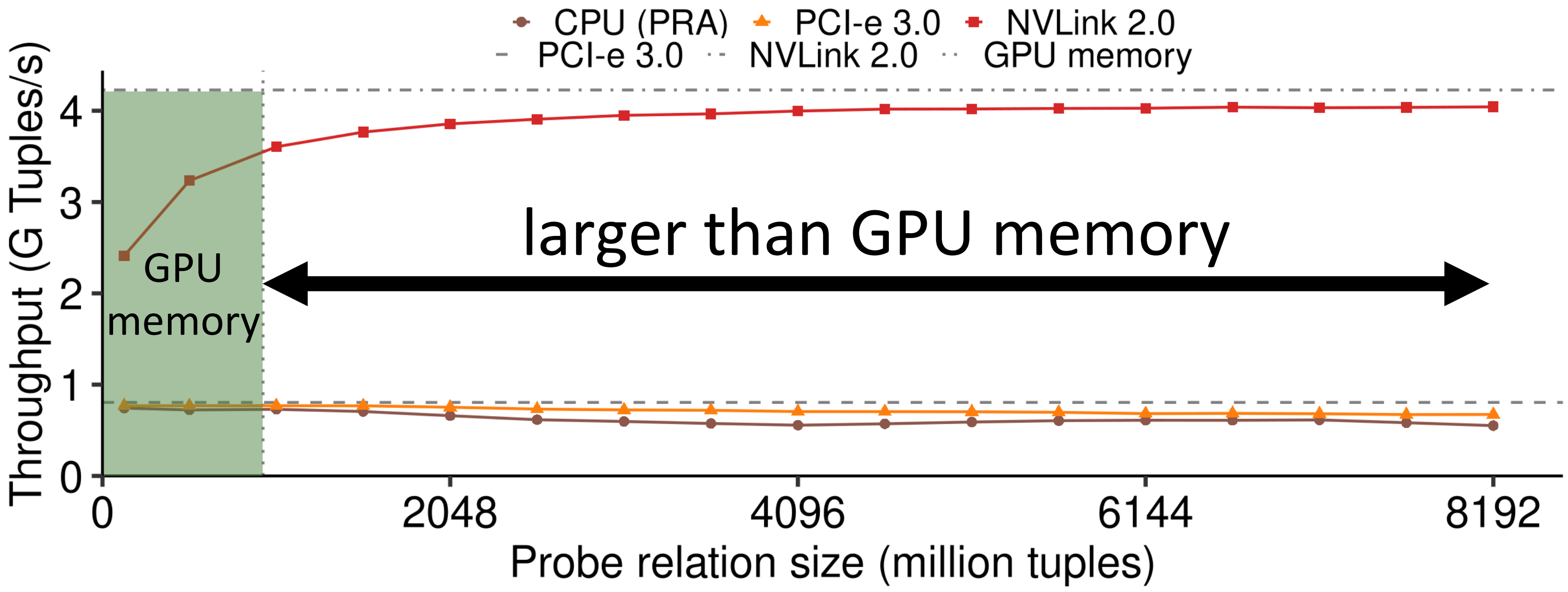
Interconnect feature: High bandwidth

Probe-Side Scaling



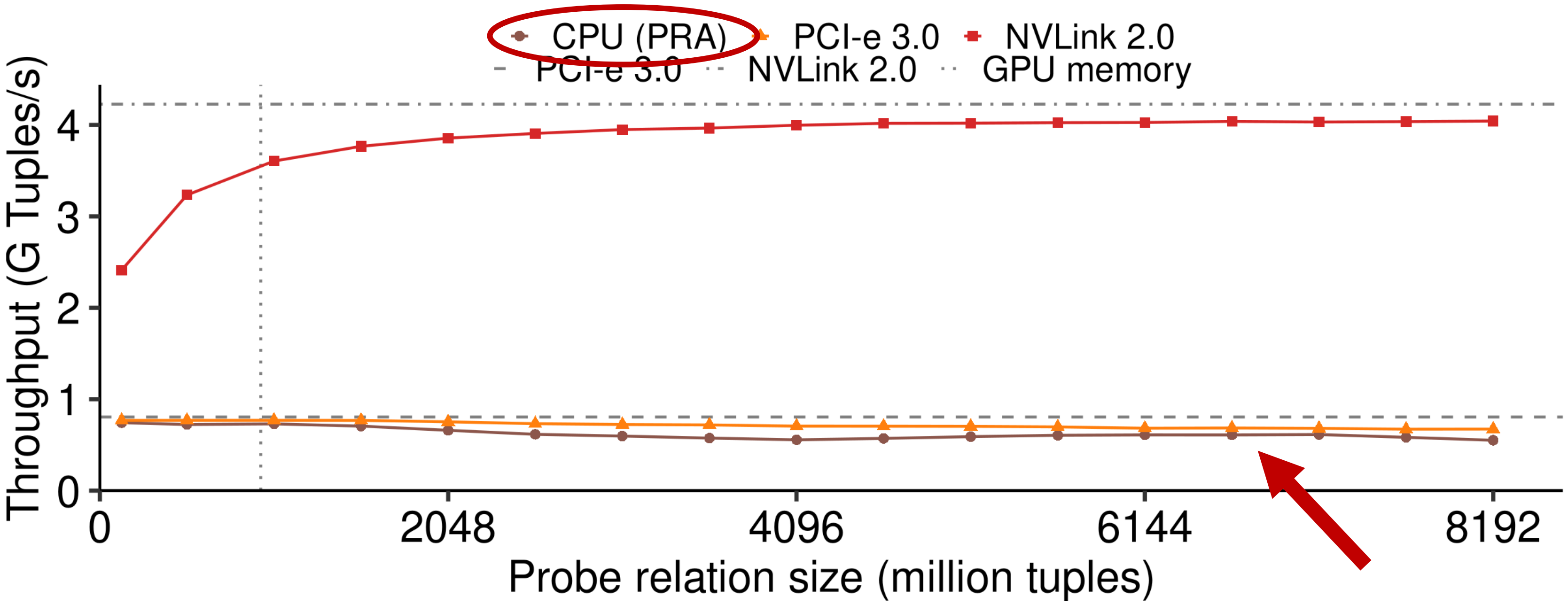
- Up to 2⋈122 GB

Probe-Side Scaling



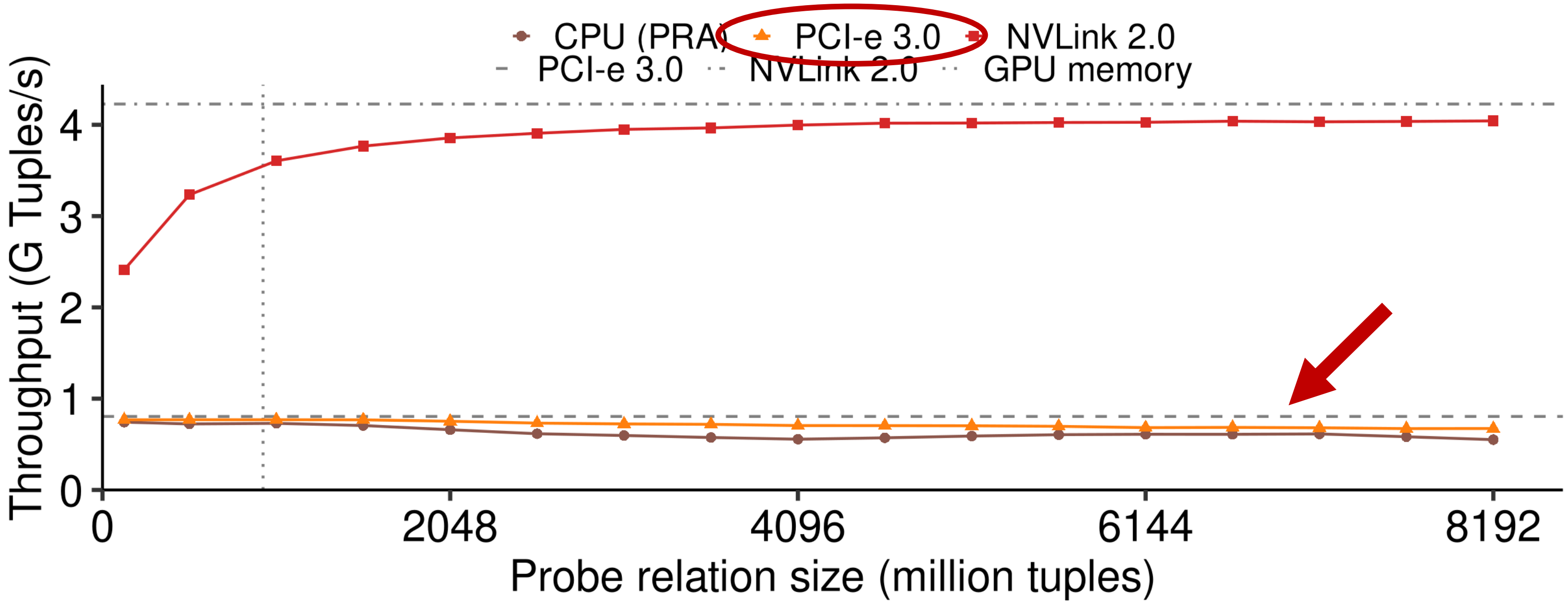
- Up to 2⋈122 GB

Probe-Side Scaling



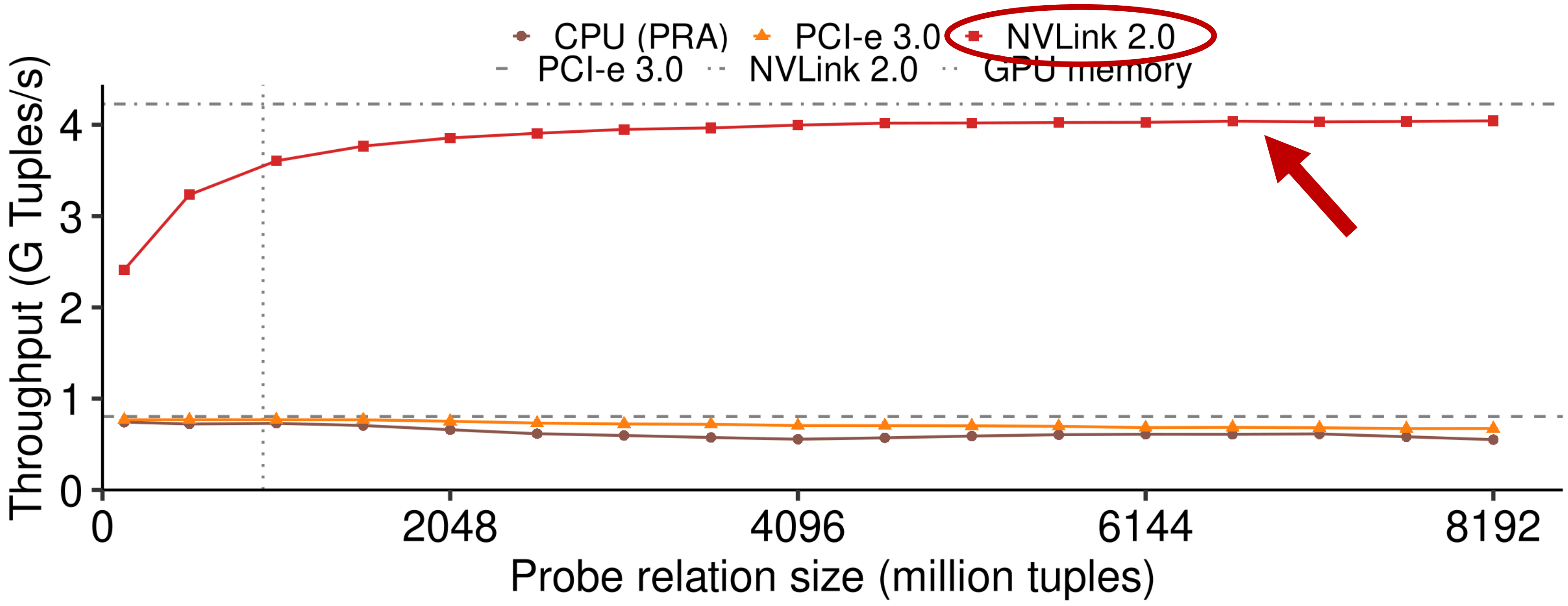
- Up to 2⋈122 GB
- CPU baseline: Radix-partitioned hash join

Probe-Side Scaling



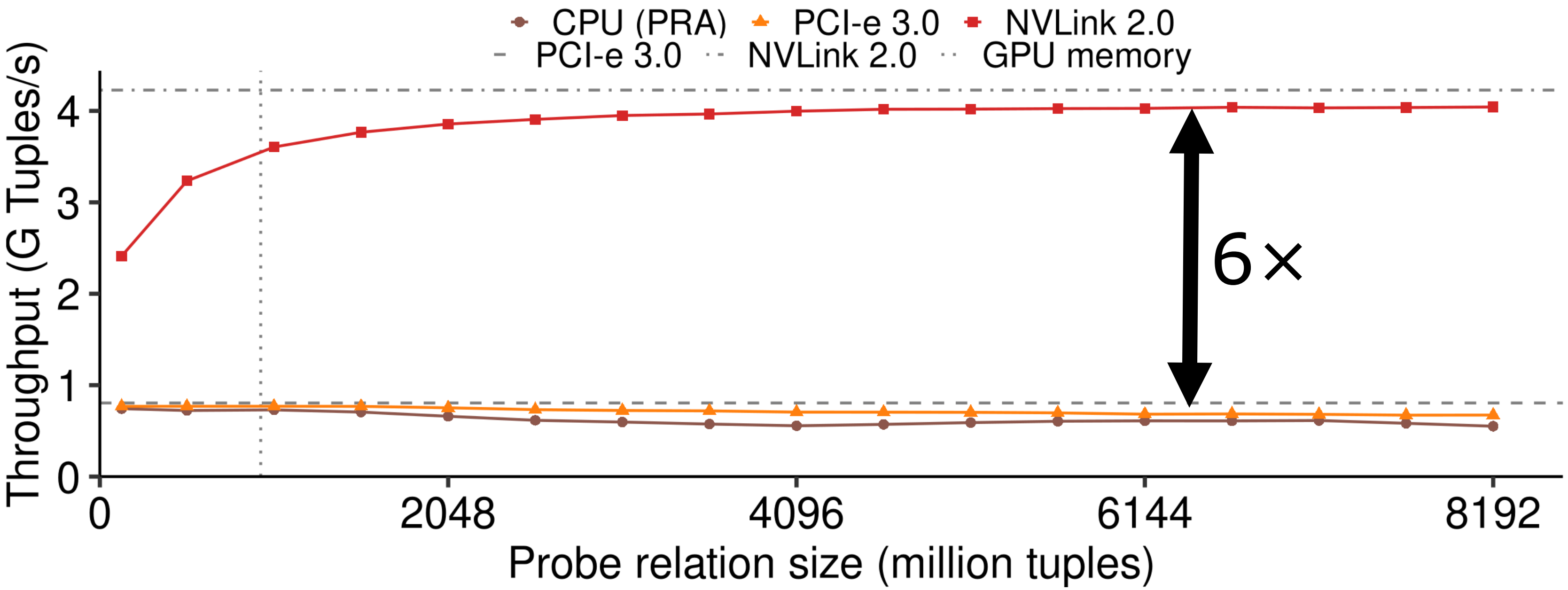
- Up to 2⋈122 GB
- CPU baseline: Radix-partitioned hash join

Probe-Side Scaling



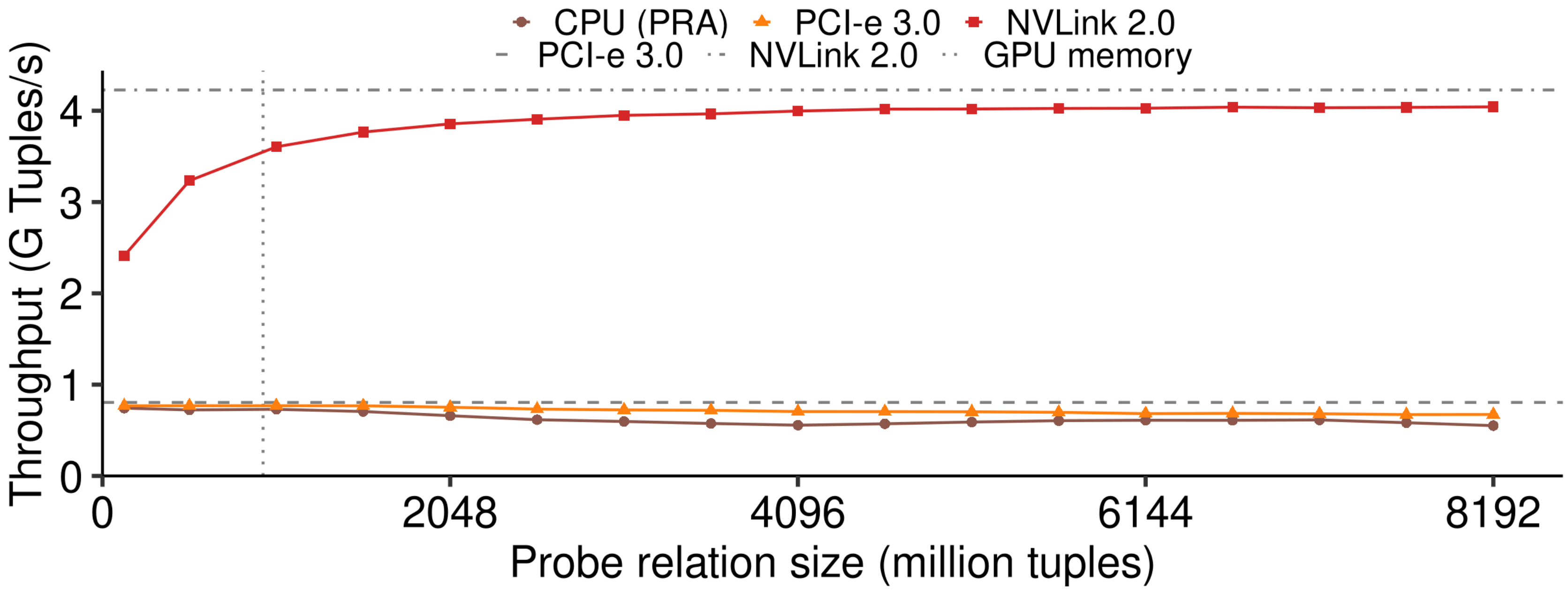
- Up to 2⋈122 GB
- CPU baseline: Radix-partitioned hash join

Probe-Side Scaling



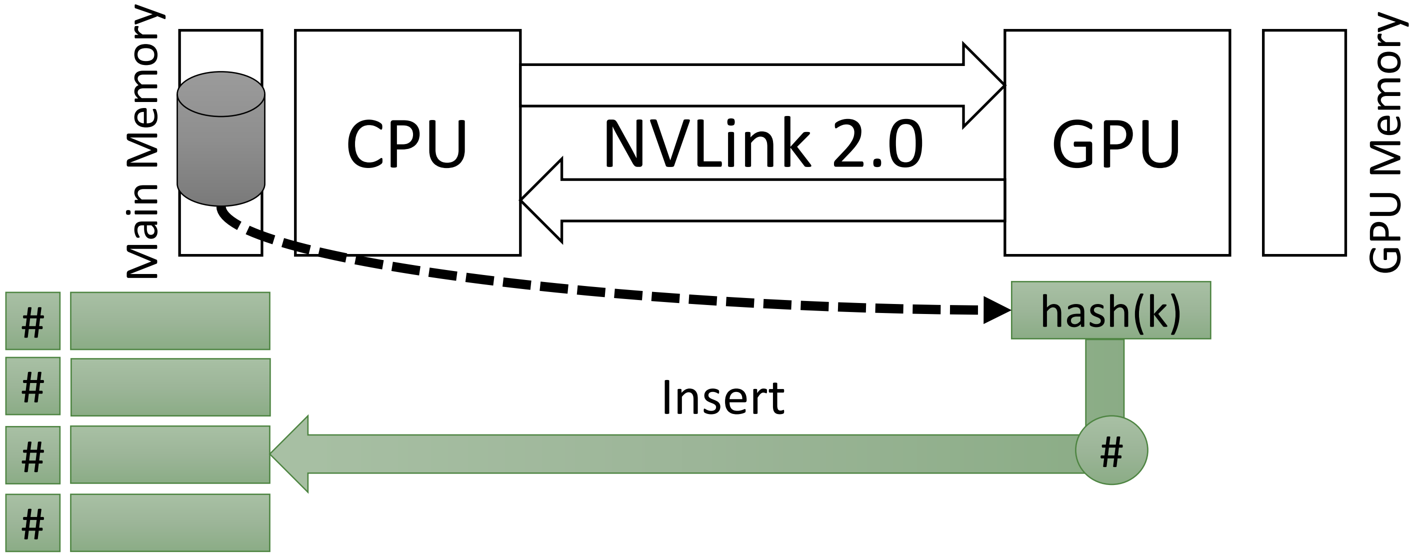
- Up to 2⋈122 GB
- CPU baseline: Radix-partitioned hash join

Probe-Side Scaling



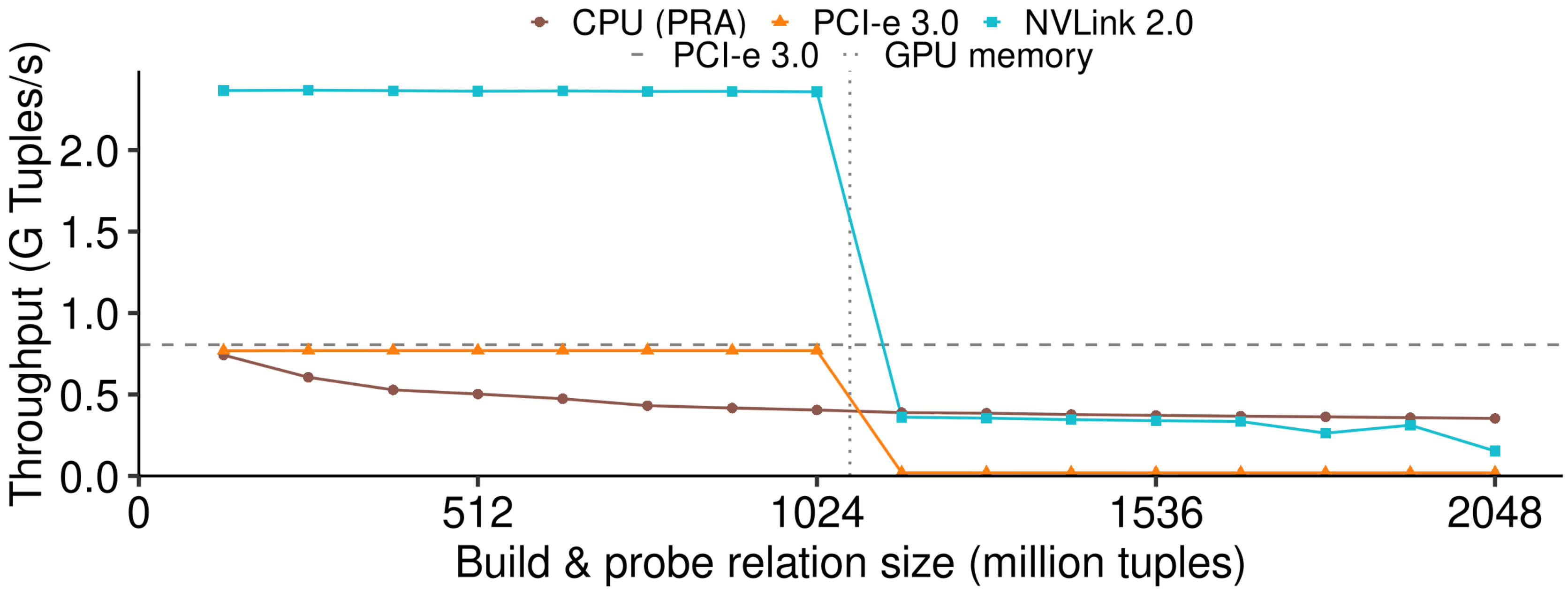
GPUs can efficiently process large, out-of-core data

Build-Side Scaling



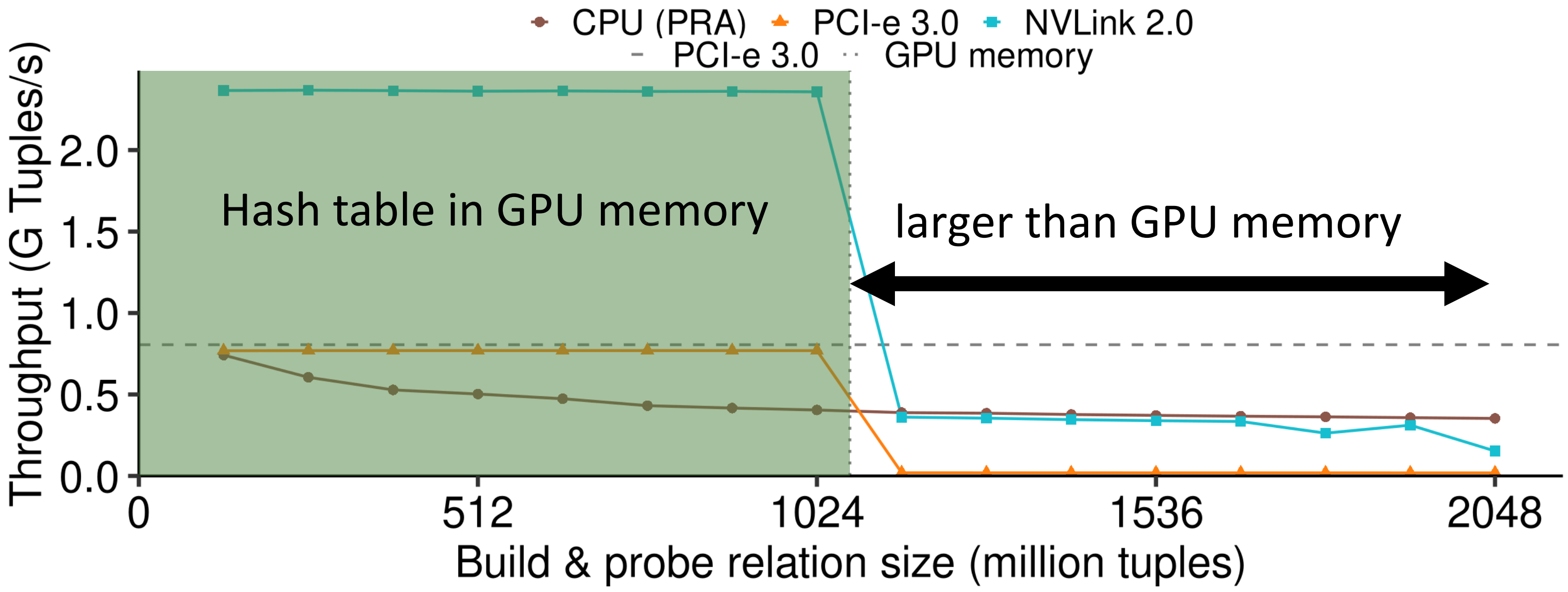
Interconnect feature: Data-dependent memory access

Build-Side Scaling



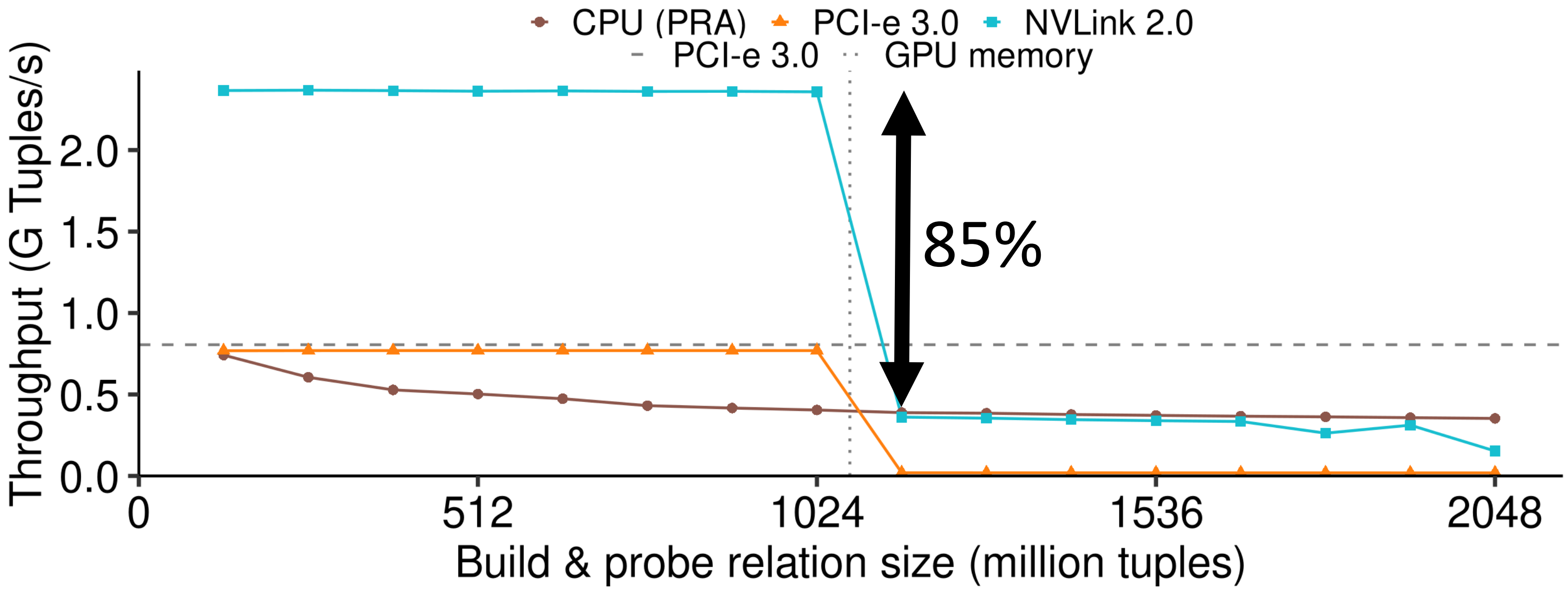
- Up to 30⋈30 GB with a 30 GB hash table **= 90 GB**

Build-Side Scaling



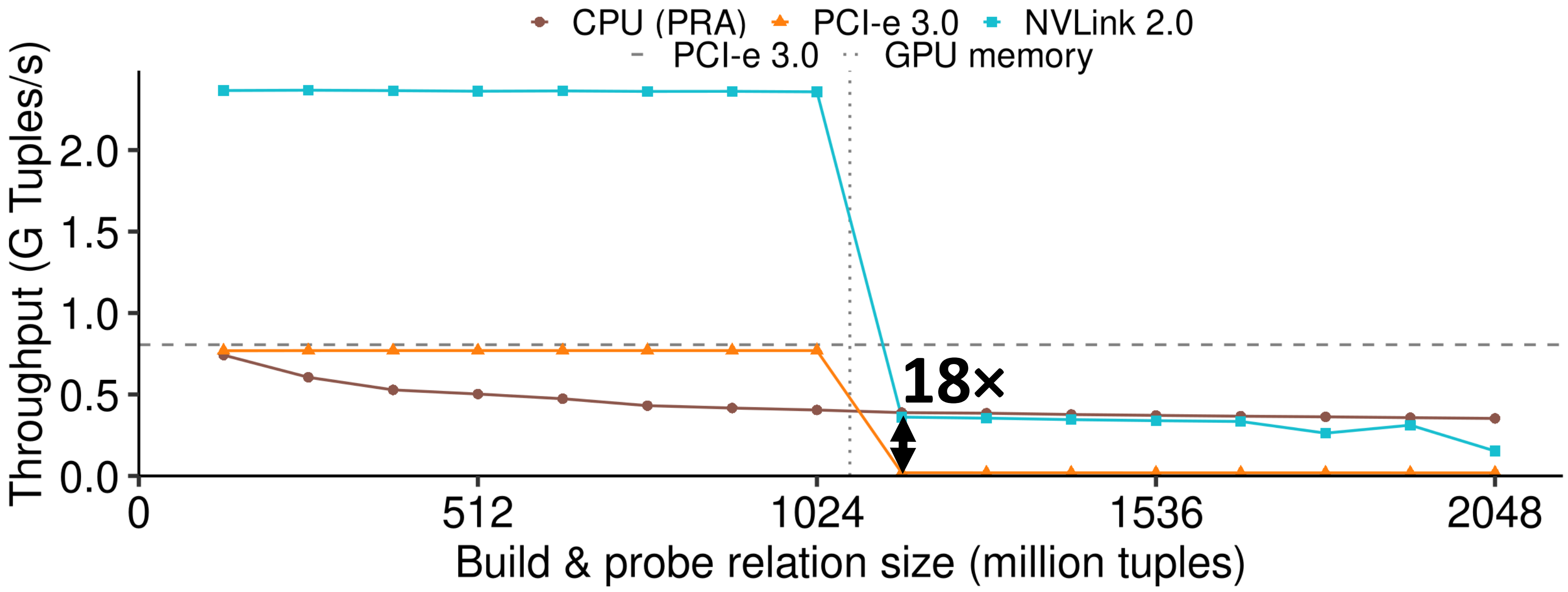
- Up to 30⋈30 GB with a 30 GB hash table **= 90 GB**

Build-Side Scaling



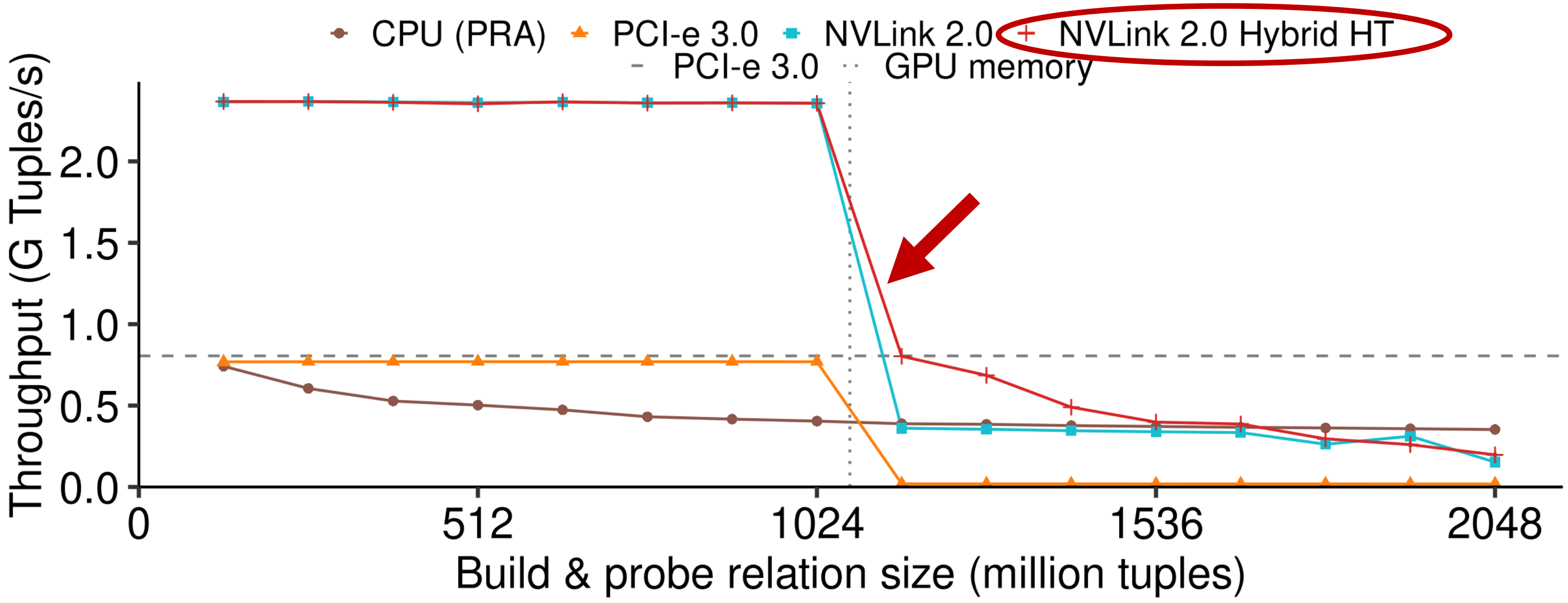
- Up to 30⋈30 GB with a 30 GB hash table **= 90 GB**

Build-Side Scaling



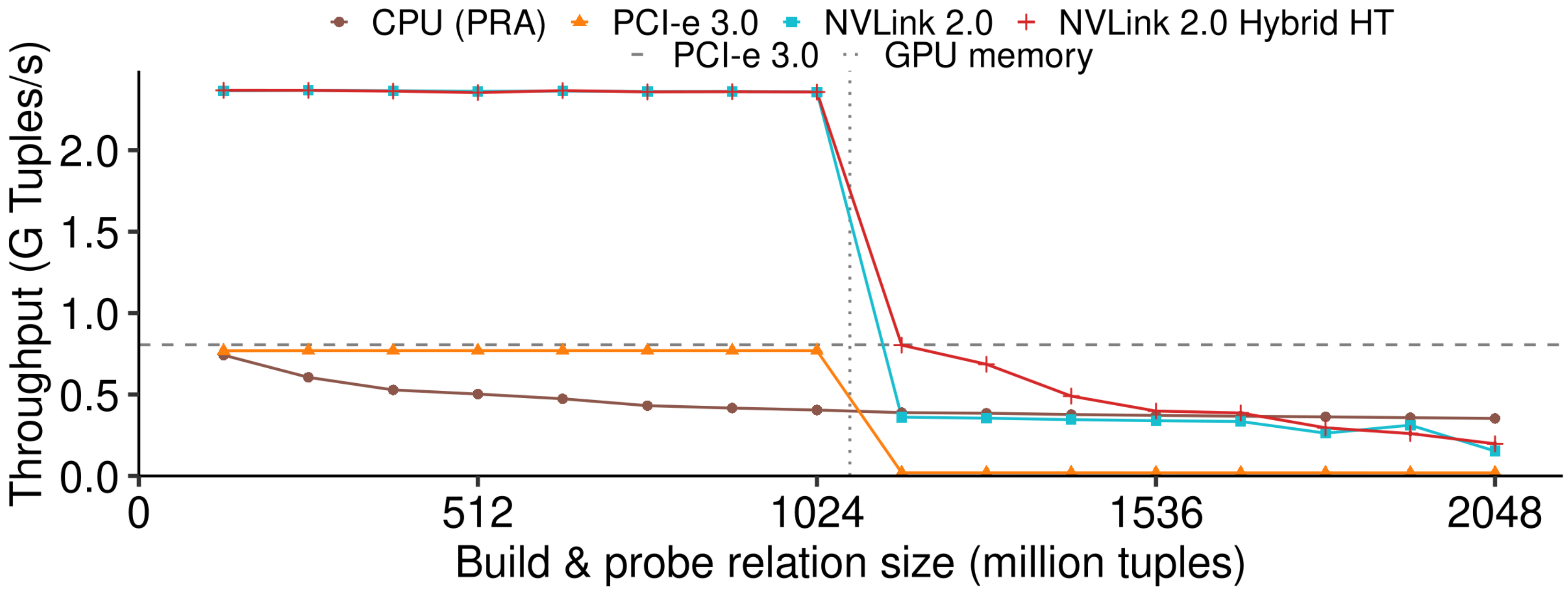
- Up to 30⋈30 GB with a 30 GB hash table **= 90 GB**

Build-Side Scaling



- Up to 30⋈30 GB with a 30 GB hash table **= 90 GB**
- Hybrid hash table spills to CPU memory

Build-Side Scaling



GPUs are able to operate on large, out-of-core data structures

… but should cache data structures in GPU memory

Conclusion

We explore ***in which ways*** *fast interconnects* ***benefit databases****:*

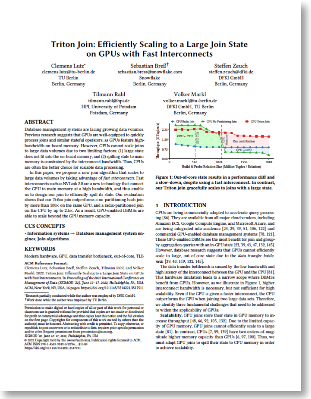
- Out-of-core **data sets**
- Out-of-core **data structures**
- **Fine-grained** cooperative co-processing

Key technology enablers:

- Hardware accelerators
- Fast interconnects

Sneak Peak – Triton Join – SIGMOD 2022

- Radix-partitioned GPU hash join
- Hierarchical partitioning



Triton Join: Efficiently Scaling to a Large Join State on GPUs with Fast Interconnects

Clemens Lutz[*]
clemens.lutz@tu-berlin.de
TU Berlin
Berlin, Germany

Sebastian Breß[†]
sebastian.bress@snowflake.com
Snowflake
Berlin, Germany

Steffen Zeuch
steffen.zeuch@dfki.de
DFKI GmbH
Berlin, Germany

Tilmann Rabl
tilmann.rabl@hpi.de
HPI, University of Potsdam
Potsdam, Germany

Volker Markl
volker.markl@tu-berlin.de
DFKI GmbH, TU Berlin
Berlin, Germany

ABSTRACT

Database management systems are facing growing data volumes. Previous research suggests that GPUs are well-equipped to quickly process joins and similar stateful operators, as GPUs feature high-bandwidth on-board memory. However, GPUs cannot scale joins to large data volumes due to two limiting factors: (1) large state does not fit into the on-board memory, and (2) spilling state to main memory is constrained by the interconnect bandwidth. Thus, CPUs are often the better choice for scalable data processing.

In this paper, we propose a new join algorithm that scales to large data volumes by taking advantage of *fast interconnects*. Fast interconnects such as NVLink 2.0 are a new technology that connect the GPU to main memory at a high bandwidth, and thus enable us to design our join to efficiently spill its state. Our evaluation shows that our *Triton* join outperforms a no-partitioning hash join by more than 100× on the same GPU, and a radix-partitioned join on the CPU by up to 2.5×. As a result, GPU-enabled DBMSs are able to scale beyond the GPU memory capacity.

CCS CONCEPTS

• **Information systems → Database management system engines; Join algorithms.**

KEYWORDS

Modern hardware, GPU, data transfer bottleneck, out-of-core, TLB

ACM Reference Format:
Clemens Lutz, Sebastian Breß, Steffen Zeuch, Tilmann Rabl, and Volker Markl. 2022. Triton Join: Efficiently Scaling to a Large Join State on GPUs with Fast Interconnects. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3514221.3517911

[*] Research partially conducted while the author was employed by DFKI GmbH.
[†] Work done while the author was employed by TU Berlin.

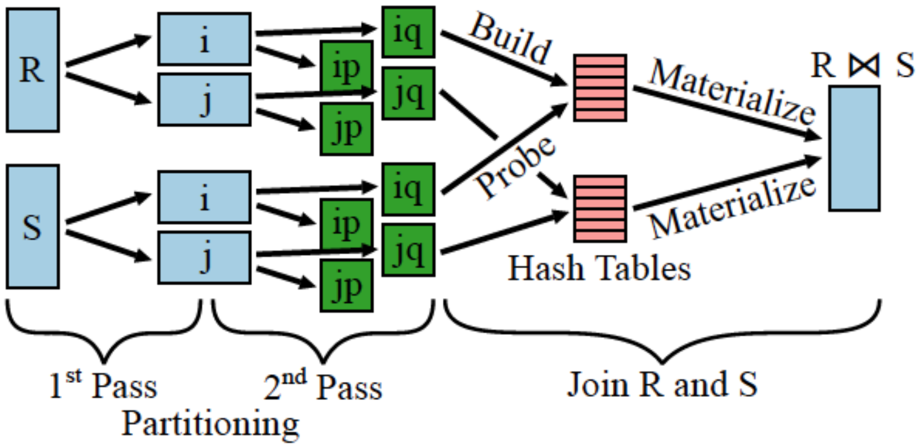
Figure 1: Out-of-core state results in a performance cliff and a slow-down, despite using a fast interconnect. In contrast, our Triton join gracefully scales to joins with a large state.

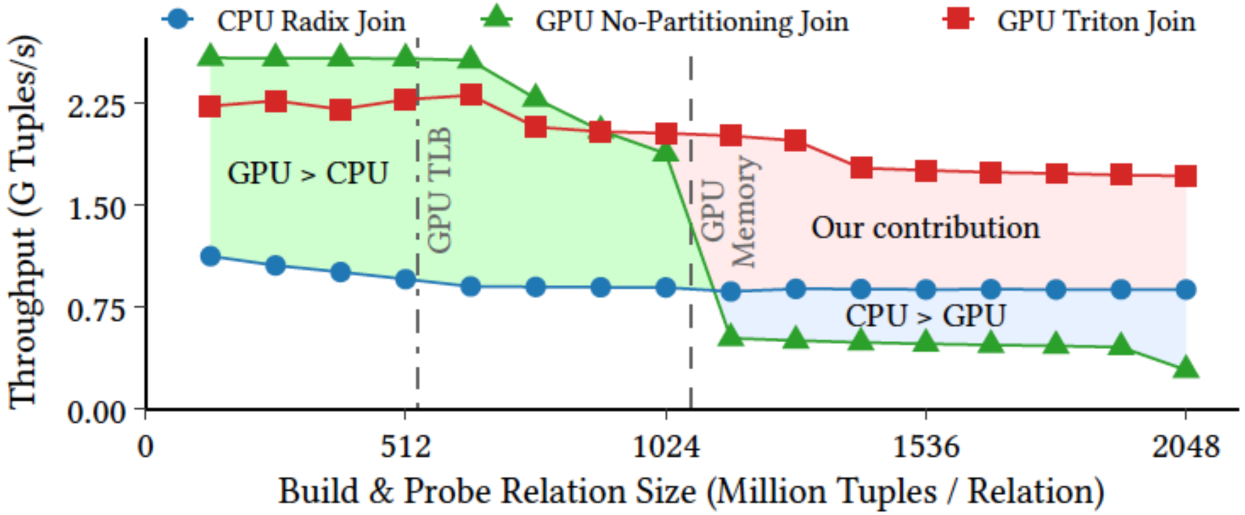
1 INTRODUCTION

GPUs are being commercially adopted to accelerate query processing [86]. They are available from all major cloud vendors, including Amazon EC2, Google Compute Engine, and Microsoft Azure, and are being integrated into academic [24, 29, 39, 51, 106, 122] and commercial GPU-enabled database management systems [78, 121]. These GPU-enabled DBMSs see the most benefit for join and group-by aggregation queries with an in-GPU state [23, 39, 43, 47, 132, 145]. However, database research suggests that GPUs cannot efficiently scale to large, out-of-core state due to the *data transfer bottleneck* [39, 43, 119, 132, 145].

The data transfer bottleneck is caused by the low bandwidth and high latency of the interconnect between the GPU and the CPU [81]. This hardware limitation leads to a narrow scope where DBMSs benefit from GPUs. However, as we illustrate in Figure 1, higher interconnect bandwidth is necessary, but not sufficient for high scalability. Even if the GPU is given a faster interconnect, the CPU outperforms the GPU when joining two large data sets. Therefore, we identify three fundamental challenges that need to be addressed to widen the applicability of GPUs:

Scalability. GPU joins store their state in GPU memory to increase throughput [48, 66, 92, 105, 132]. Due to the limited capacity of GPU memory, GPU joins cannot efficiently scale to a large state [81]. In contrast, CPUs [7, 59, 139] have two orders-of-magnitude higher memory capacity than GPUs [6, 97, 100]. Thus, we must adapt GPU joins to spill their state to CPU memory in order to achieve scalability.





Triton Join: Efficiently Scaling to a Large Join State on GPUs with Fast Interconnects – Clemens Lutz et al. SIGMOD 2022

Multi-GPU Sorting

Multi-GPU Data Processing @ DES

Evaluating Multi-GPU Sorting with Modern Interconnects @ SIGMOD 2022

Tobias Maltenberger, Ivan Ilic, Ilin Tolovski, Tilmann Rabl

RMG Sort: A Radix-Partitioning-Based Multi-GPU Sorting Algorithm @ BTW 2023

Ivan Ilic, Ilin Tolovski, Tilmann Rabl

Evaluating Multi-GPU Sorting with Modern Interconnects

Tobias Maltenberger*, Ivan Ilic*, Ilin Tolovski, Tilmann Rabl
Hasso Plattner Institute, University of Potsdam
{tobias.maltenberger,ivan.ilic}@student.hpi.de,{ilin.tolovski,tilmann.rabl}@hpi.de

ABSTRACT

Figure 1: Sorting 16 GB on the DGX A100: CPU vs. GPUs

B. König-Ries et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2023),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 1

RMG Sort: Radix-Partitioning-Based Multi-GPU Sorting

Ivan Ilic[1] Ilin Tolovski[2] Tilmann Rabl[3]

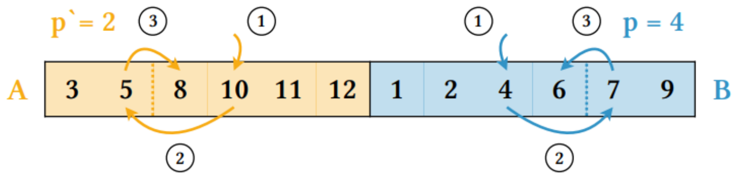
Peer-to-Peer-based Sorting

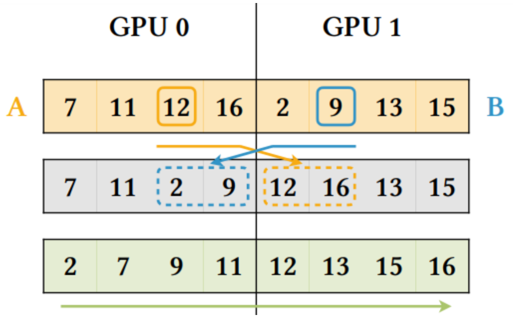
- Algorithm stages:
  1. Split data across the GPUs
  2. Find the pivot on each GPU
  3. Exchange data between GPUs
  4. Sorted subarrays

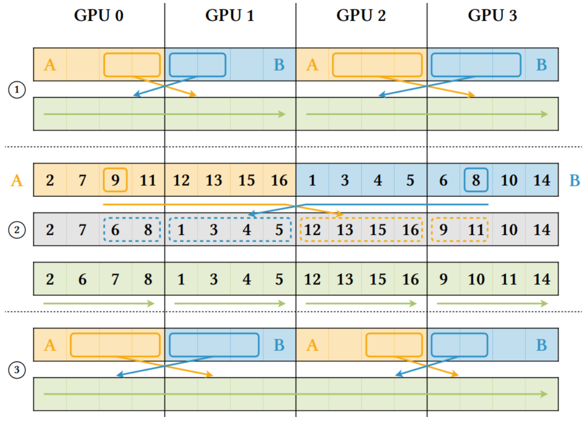
- In-place sorting algorithm

- Algorithm limited by the total GPU memory

- Max. data to sort: ~50% of total GPU memory

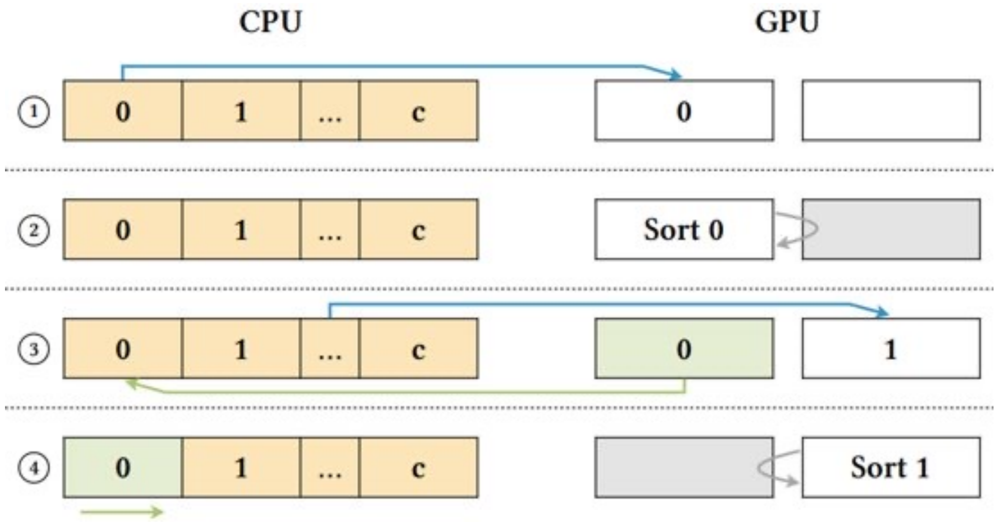






Heterogenous CPU-GPU Sorting

- Algorithm stages:
  1. Fill a single GPU with data
  2. Execute an in-place sort
  3. Bi-directional CPU-GPU data exchange
  4. CPU Merge & GPU Sort executed in parallel
  5. Final merge on CPU

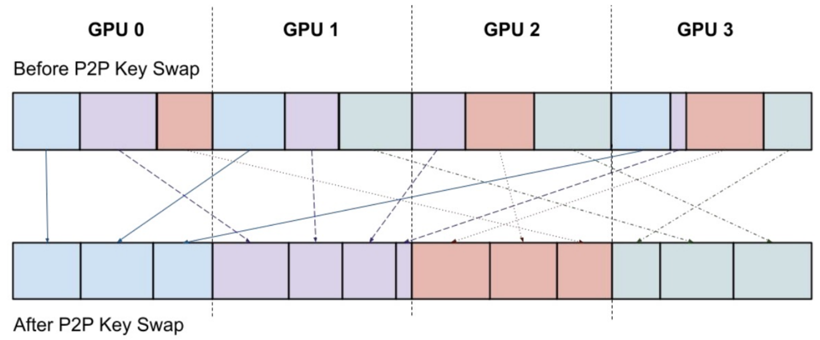


- Sorted data can be larger than combined GPU capacity
- Main memory bound algorithm

Radix-Partitioning-based Sorting

- Algorithm stages:
  1. Partition keys based on radix value (MSB)
  2. Exchange buckets between all GPUs via P2P transfers
  3. Sort buckets on each GPU using a single-GPU sorting primitive

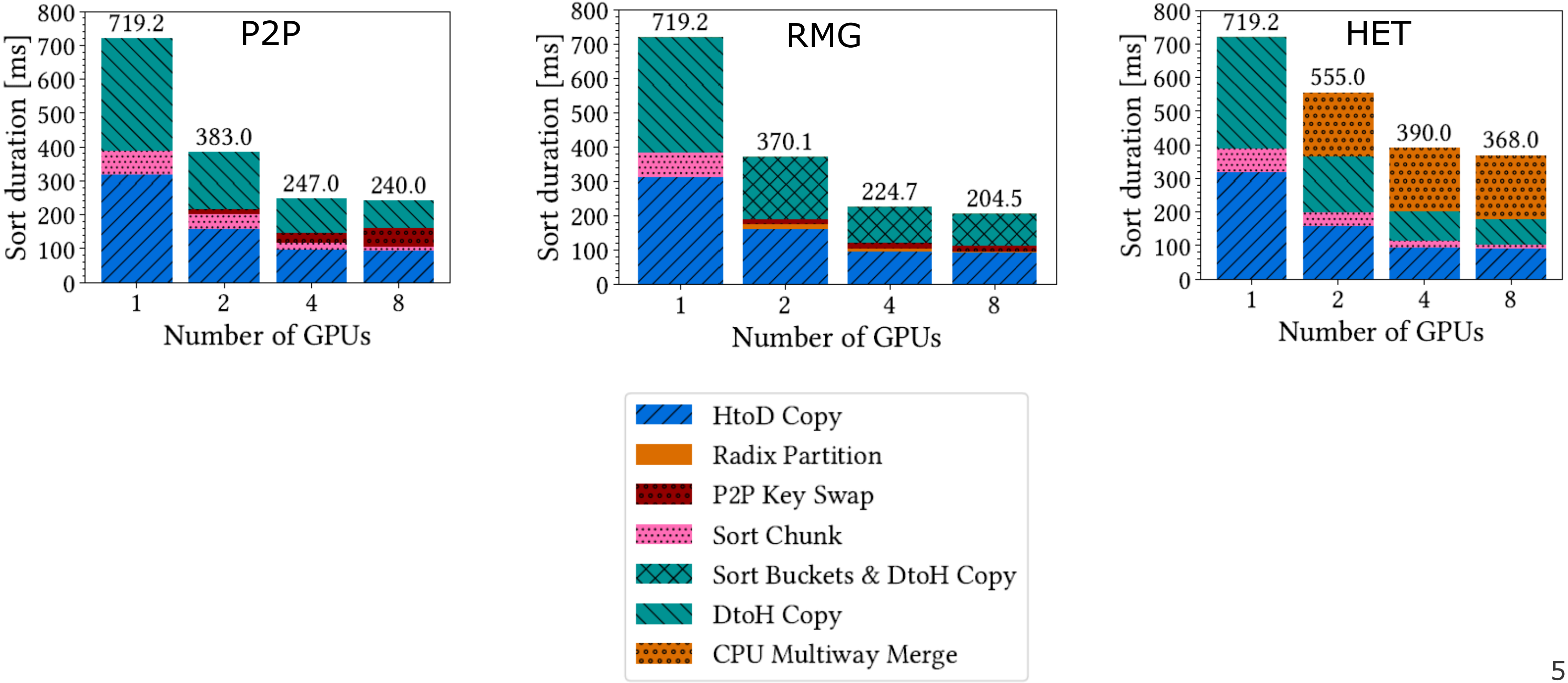
```
00000010 00111010 01110001 01100010  →  Bucket [2]
…
00000001 10101011 01011000 00110101  →  Bucket [1]
```



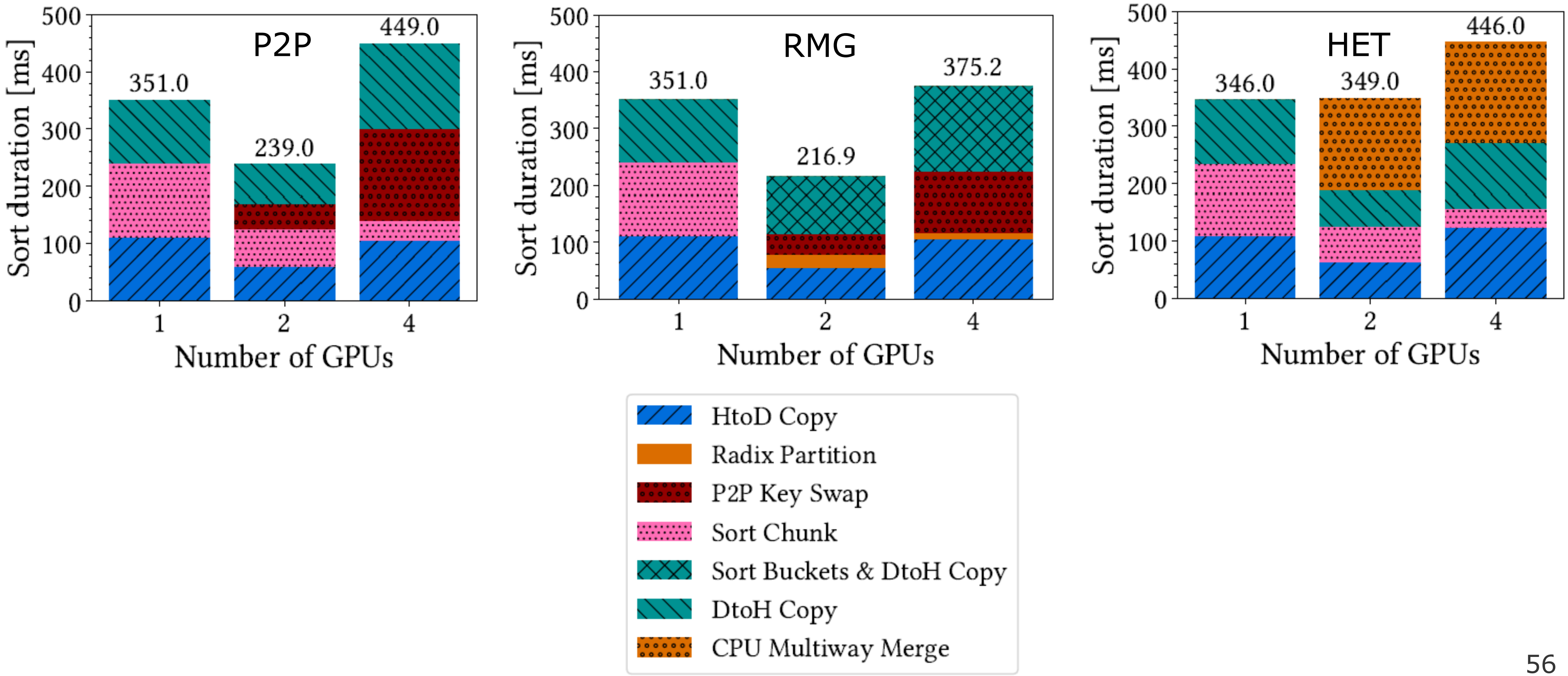
- In-place sorting algorithm
- Algorithm limited by the total GPU memory
- Max. data to sort: ~50% of total GPU memory

```
Bucket [1]:     00000001 00000000 00000000 00011010
                …
                00000001 10101011 01011000 00110101
                …
                00000001 11111111 11111111 11110101
```

Performance Analysis – DGX A100

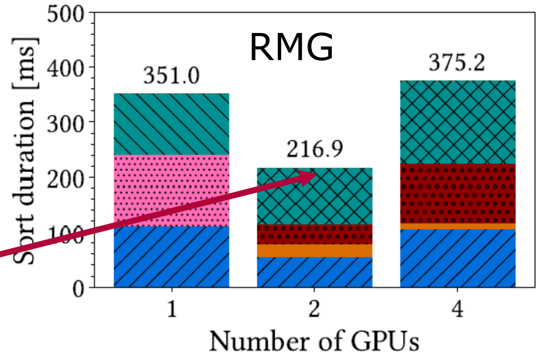


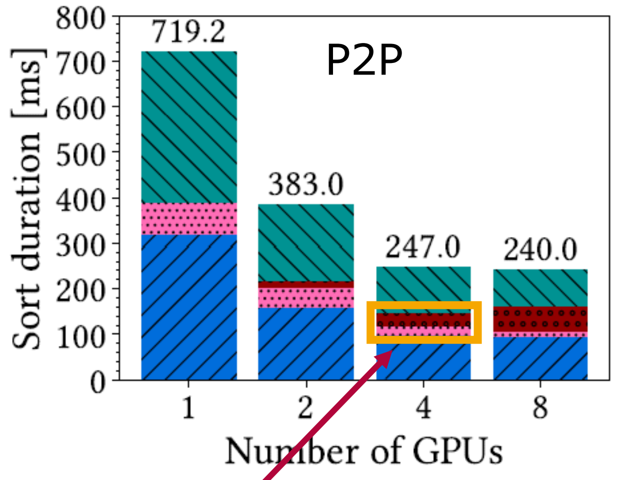
Performance Analysis - AC922



Takeaways

- Scaling operators on Multi-GPU systems has limited benefits
- Often the best performance is achieved on 2 or 4 GPUs
  - **Overall fastest solution – 2 GPU mode on AC922**
- OLAP workloads rely on data transfers to GPU Memory
  - Short computation time – majority spent on data transfers
- Bound by the interconnect bandwidth
  - **DGX A100 -> P2P throughput: 279 GB/s, HtoD: 24 GB/s**
- Topology awareness and algorithm adaptation is essential





Sorting + merging is only 10-15% of the total runtime

Summary

- GPU-accelerated data processing
- Scalable GPU Joins
- Multi-GPU Sorting
- Coming next: Multi-GPU Joins
- Questions?
  - tilmann.rabl@hpi.de