Efficient Join Processing across Heterogeneous Processors

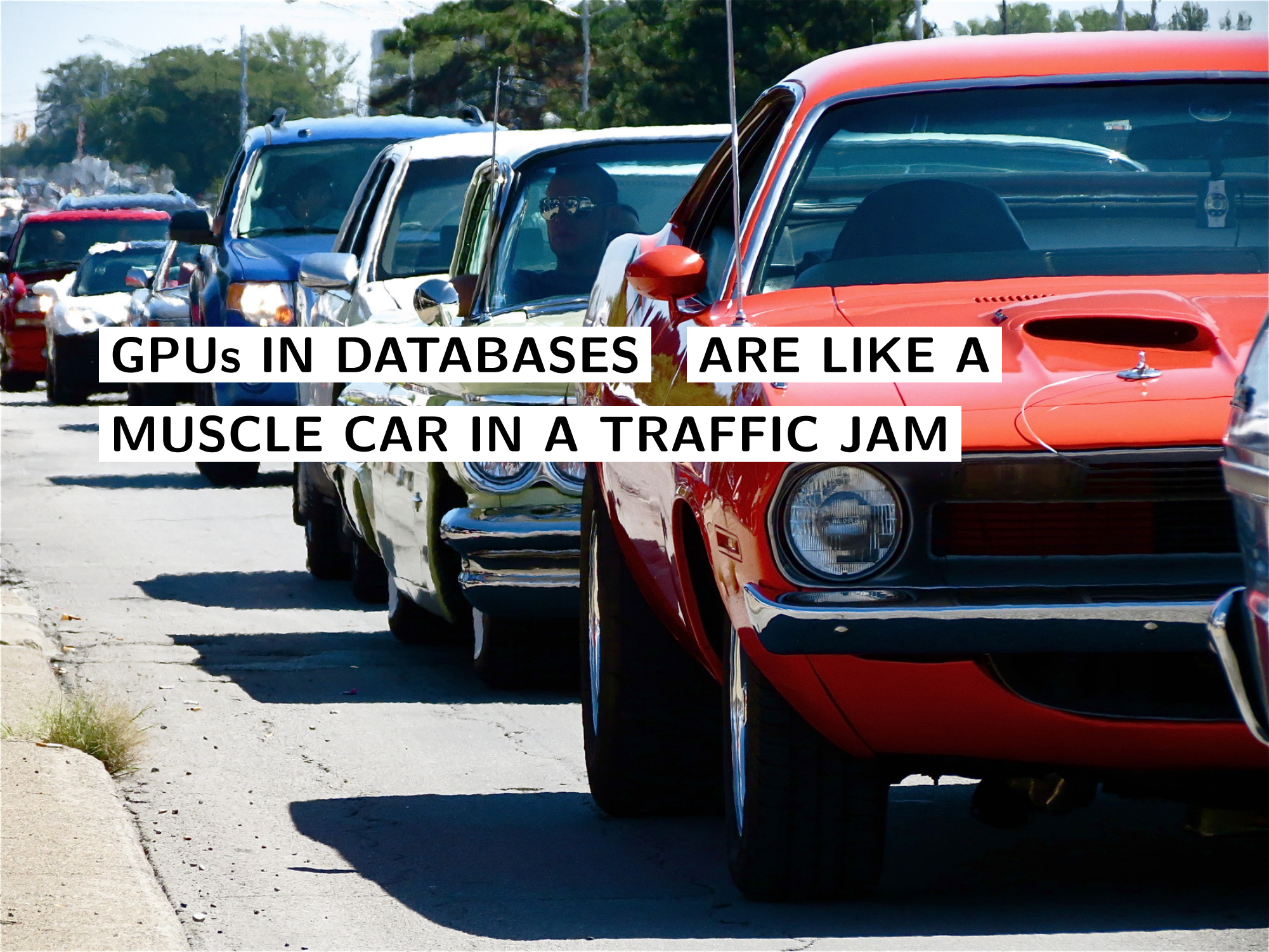
Henning Funke, Sebastian Breß, Stefan Noll, Jens Teubner

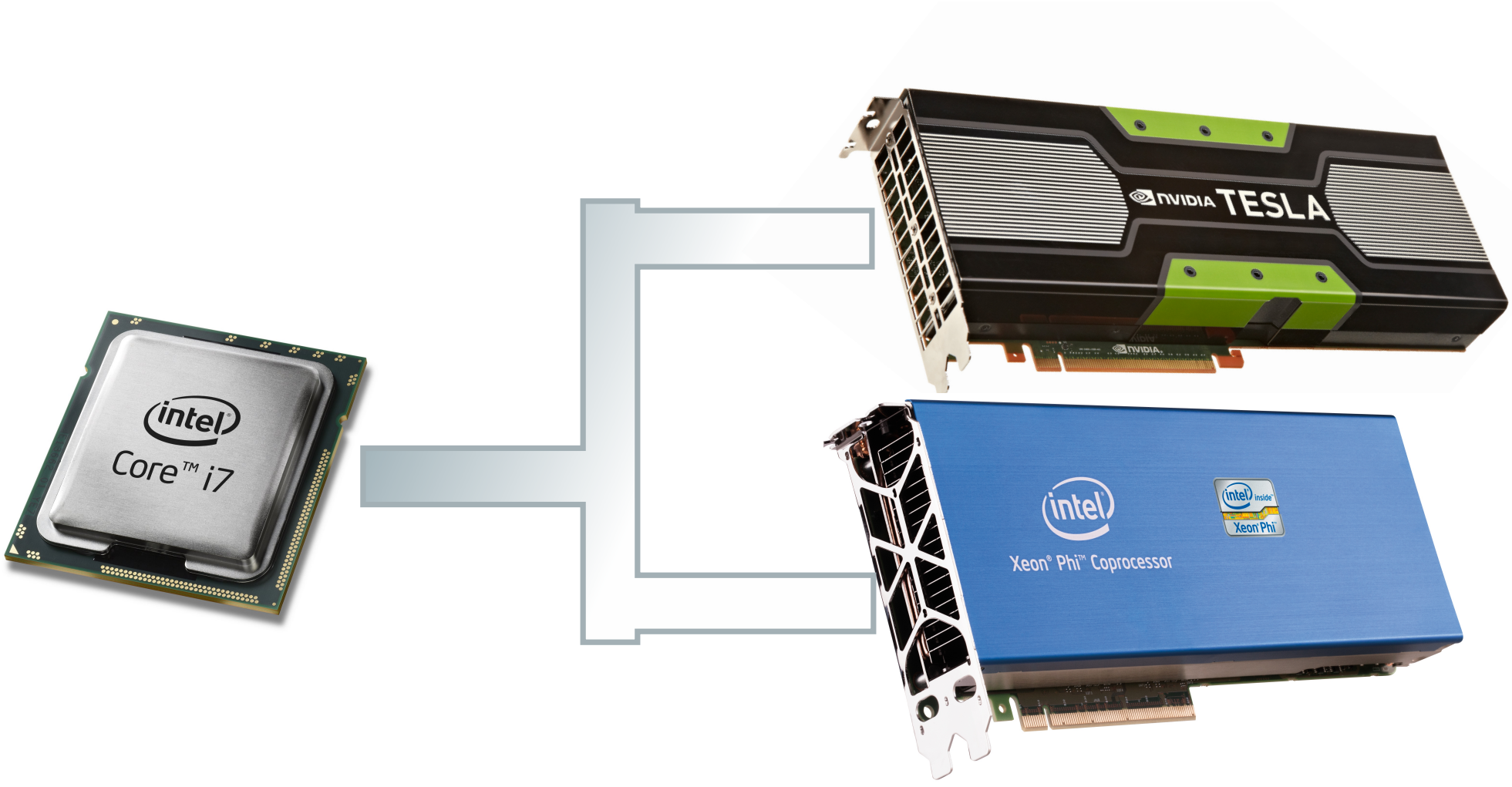
December 15, 2015

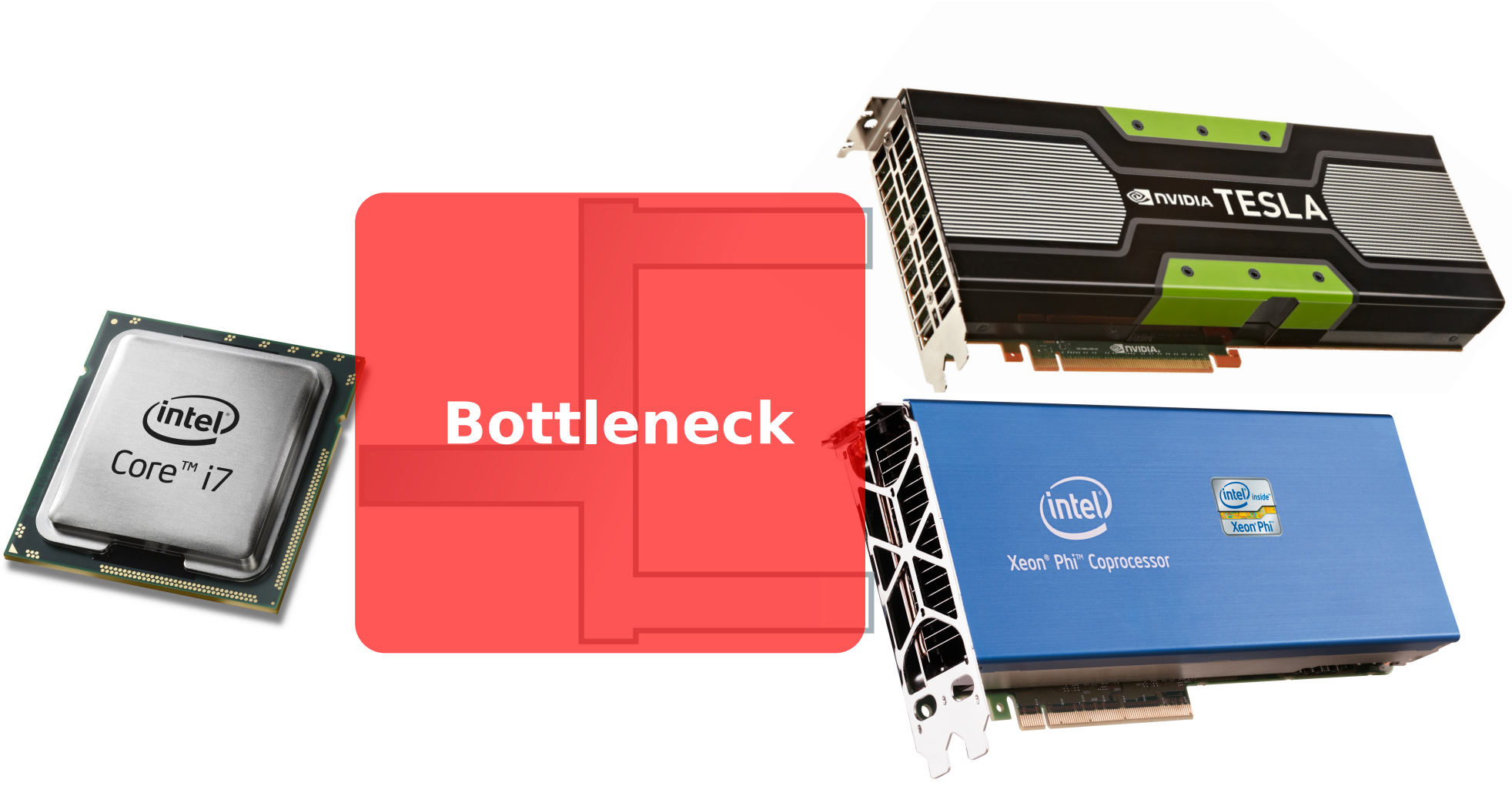
technische universität dortmund

Lehrstuhl VI Datenbanken und Informationssysteme

German Research Center for Artificial Intelligence

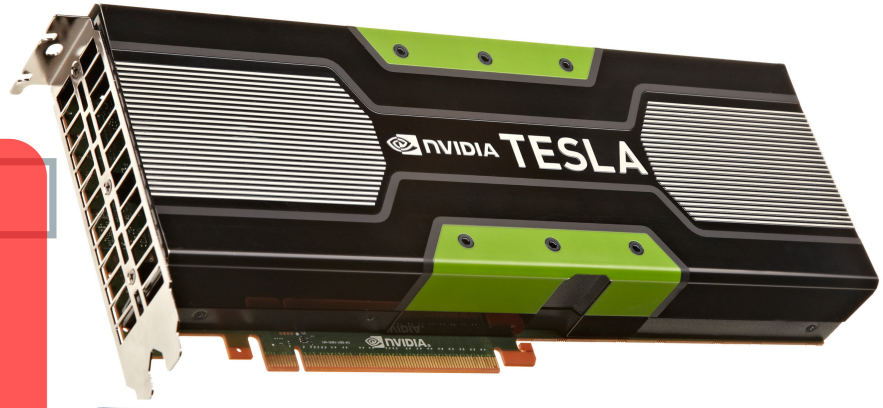

GPUs IN DATABASES ARE LIKE A
MUSCLE CAR IN A TRAFFIC JAM

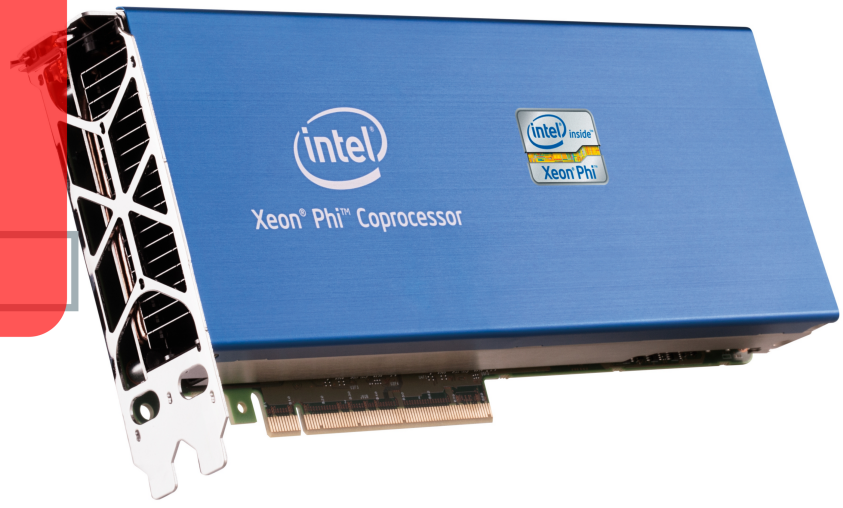

intel
Core™ i7
NVIDIA TESLA
intel
Xeon® Phi™ Coprocessor
intel inside
Xeon Phi


intel
Core™ i7
Bottleneck
NVIDIA TESLA
intel
Xeon® Phi™ Coprocessor
intel inside
Xeon Phi

Bottleneck

really?





GPU – Hashjoin Algorithm

Cuckoo Hashing Implementation[1]

→ Strict limit on number probes per query key

- Pipeline join probe and result compaction in shared memory

[1] Based on: Alcantara, Dan Anthony Feliciano. *Efficient hash tables on the GPU.* University of California at Davis, 2011.

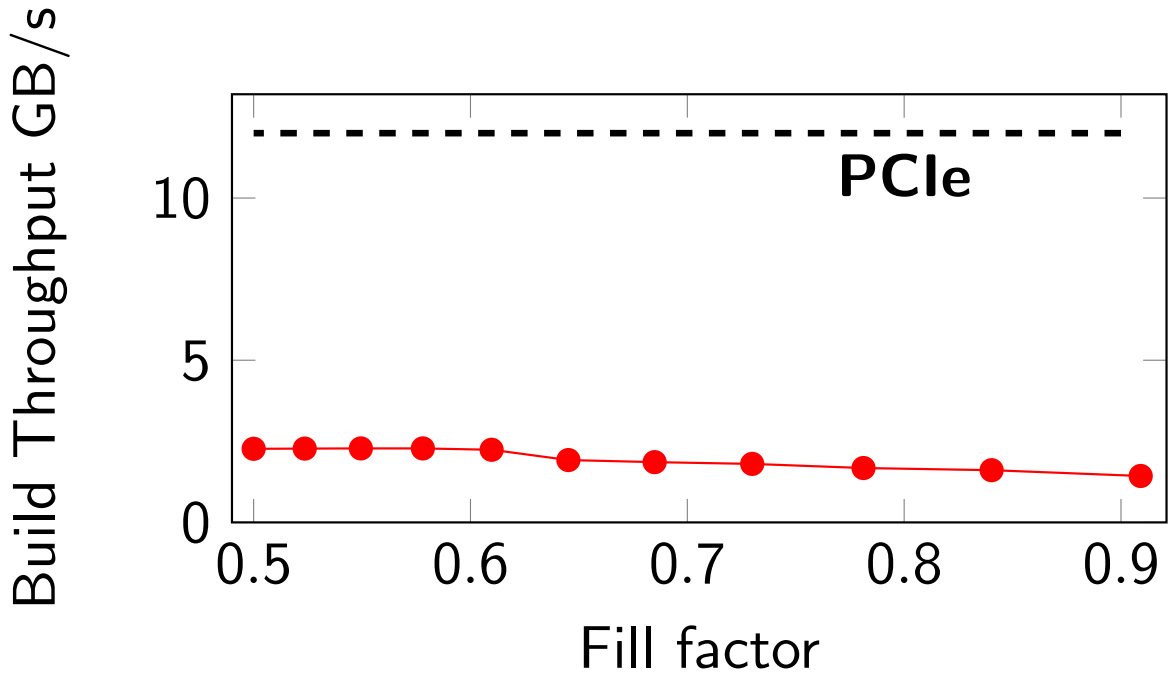
GPU – Hashjoin Algorithm

Cuckoo Hashing Implementation[1]

→ Strict limit on number probes per query key

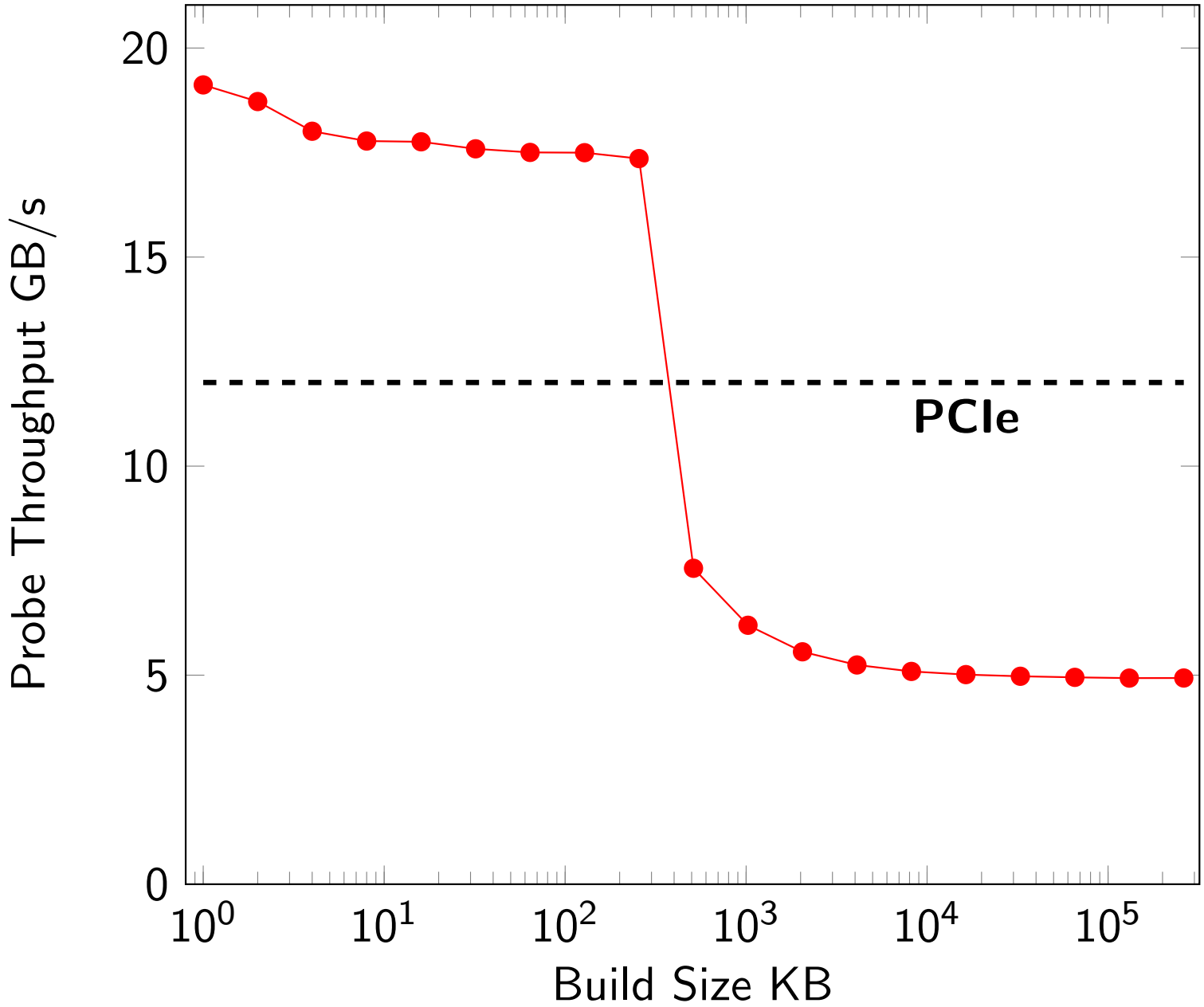
- Pipeline join probe and result compaction in shared memory

Performance: Build (GTX970)

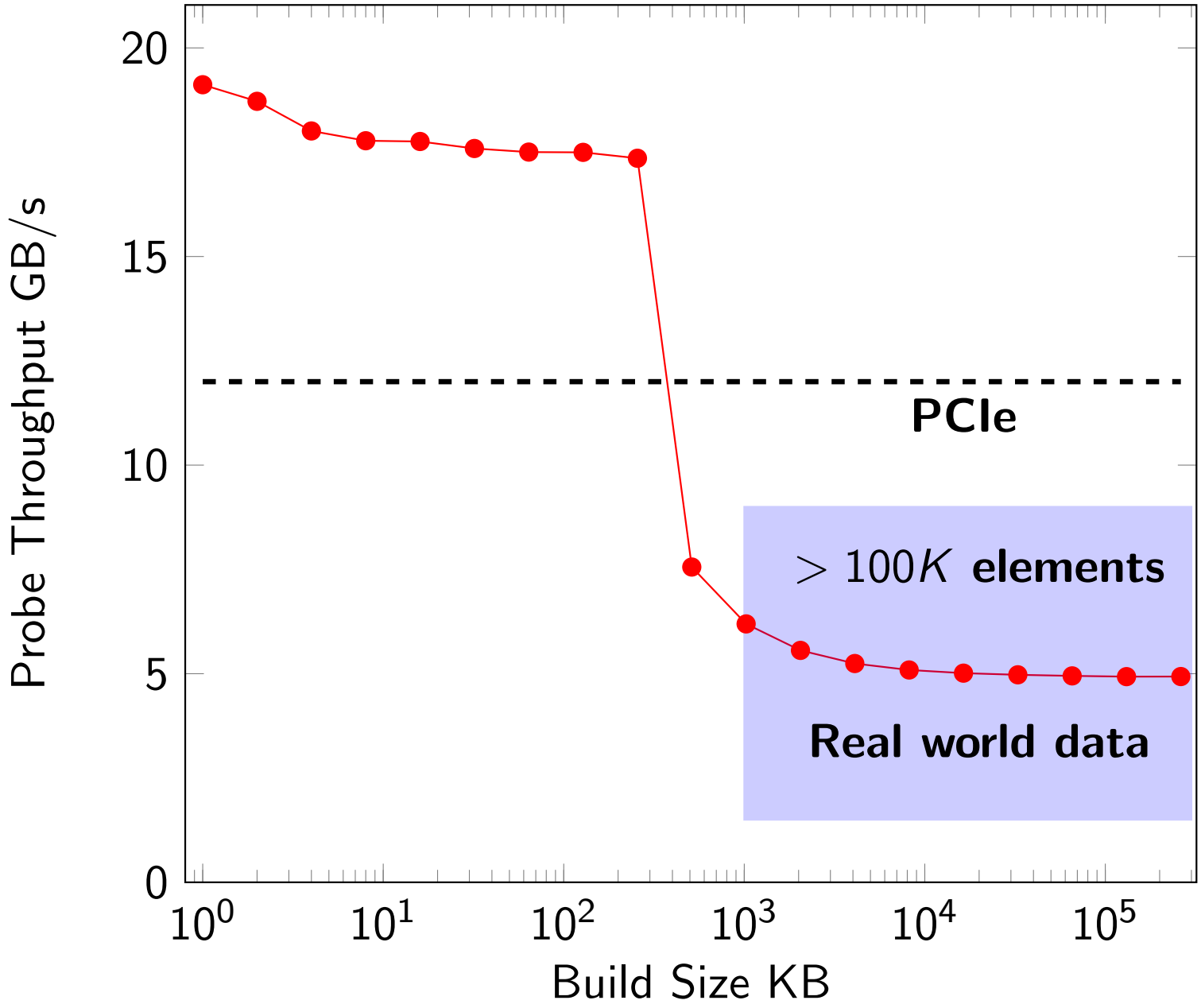


[1] Based on: Alcantara, Dan Anthony Feliciano. *Efficient hash tables on the GPU.* University of California at Davis, 2011.

Performance: GPU Join Probe (GTX970)



Performance: GPU Join Probe (GTX970)



Joins on Multiple Heterogeneous Processors

Challenges

- Scalability to large data
- Communication
- Local and remote resources

Related work

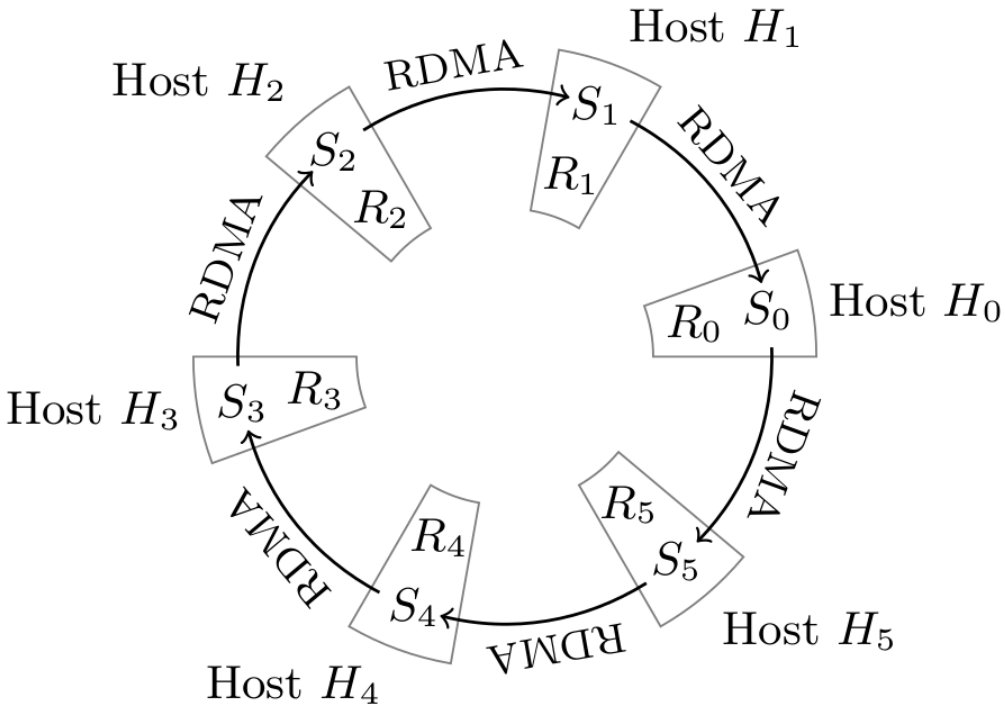
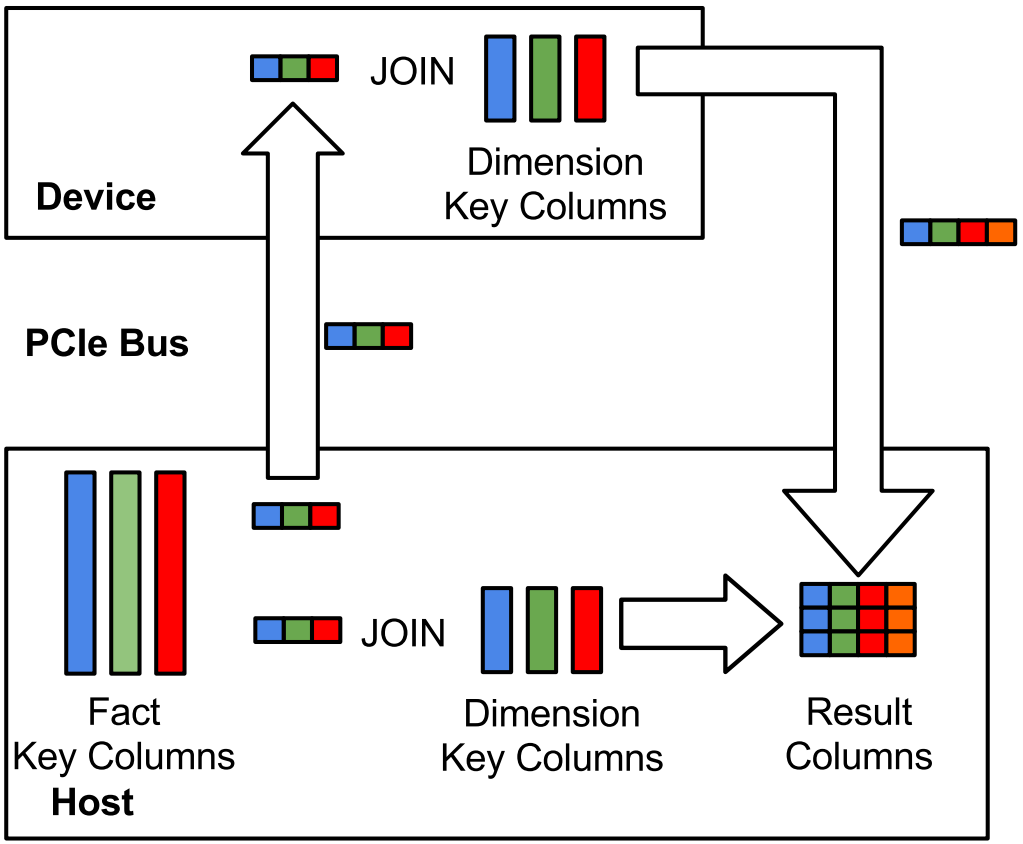


Figure : P. Frey, R. Goncalves, M. L. Kersten, and J. Teubner. Spinning Relations: High-Speed Networks for Distributed Join Processing. In DaMoN, 2009.

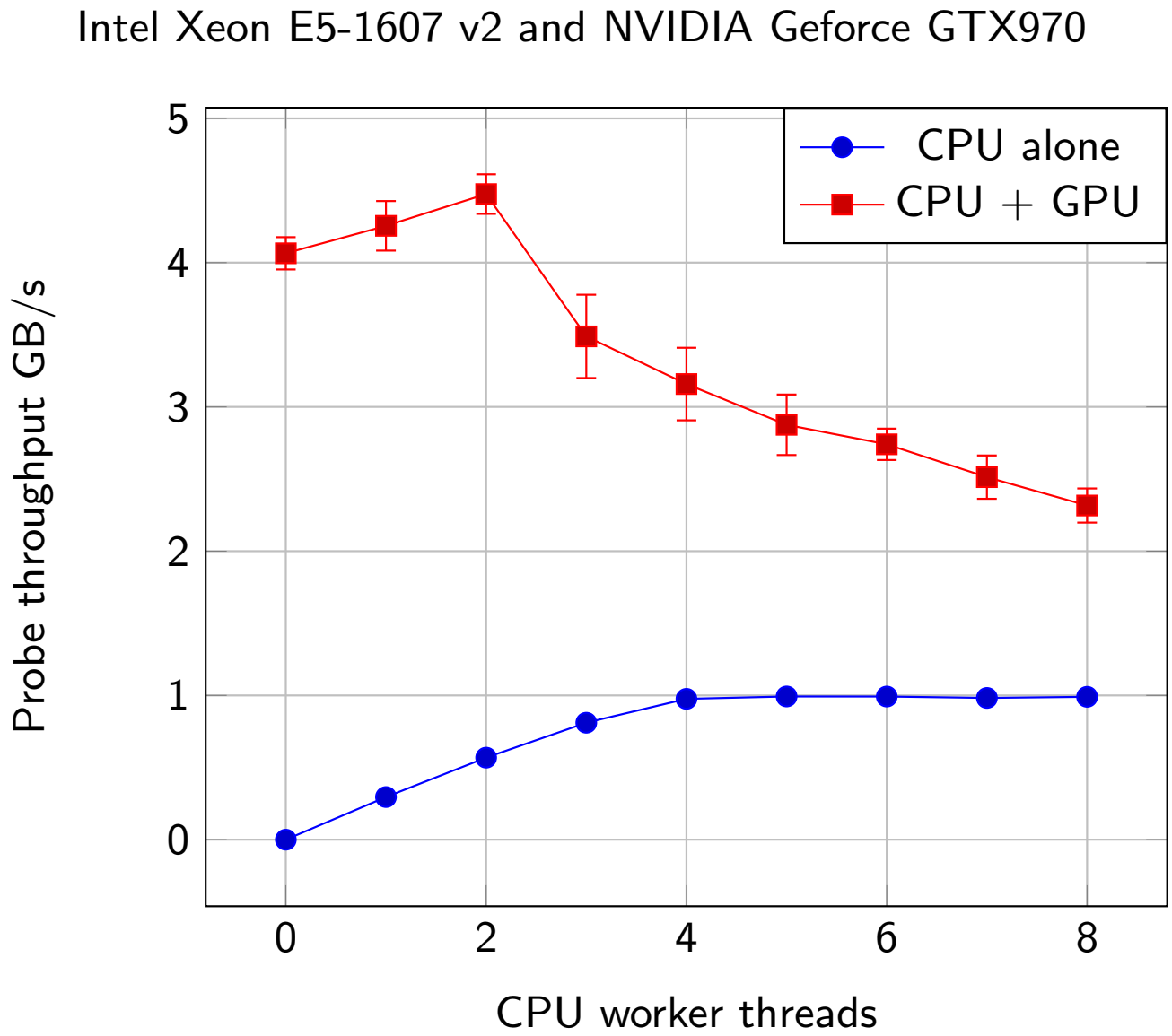
Star Join on Heterogeneous Processors



Processing Strategy

- → Allocate smaller tables on all devices
- → Asynchronous data transfers at computation speed
- → Merge results into continuous arrays

Performance: Join Probe Across Heterogeneous Processors



Coprocessor Control Thread Scheduling

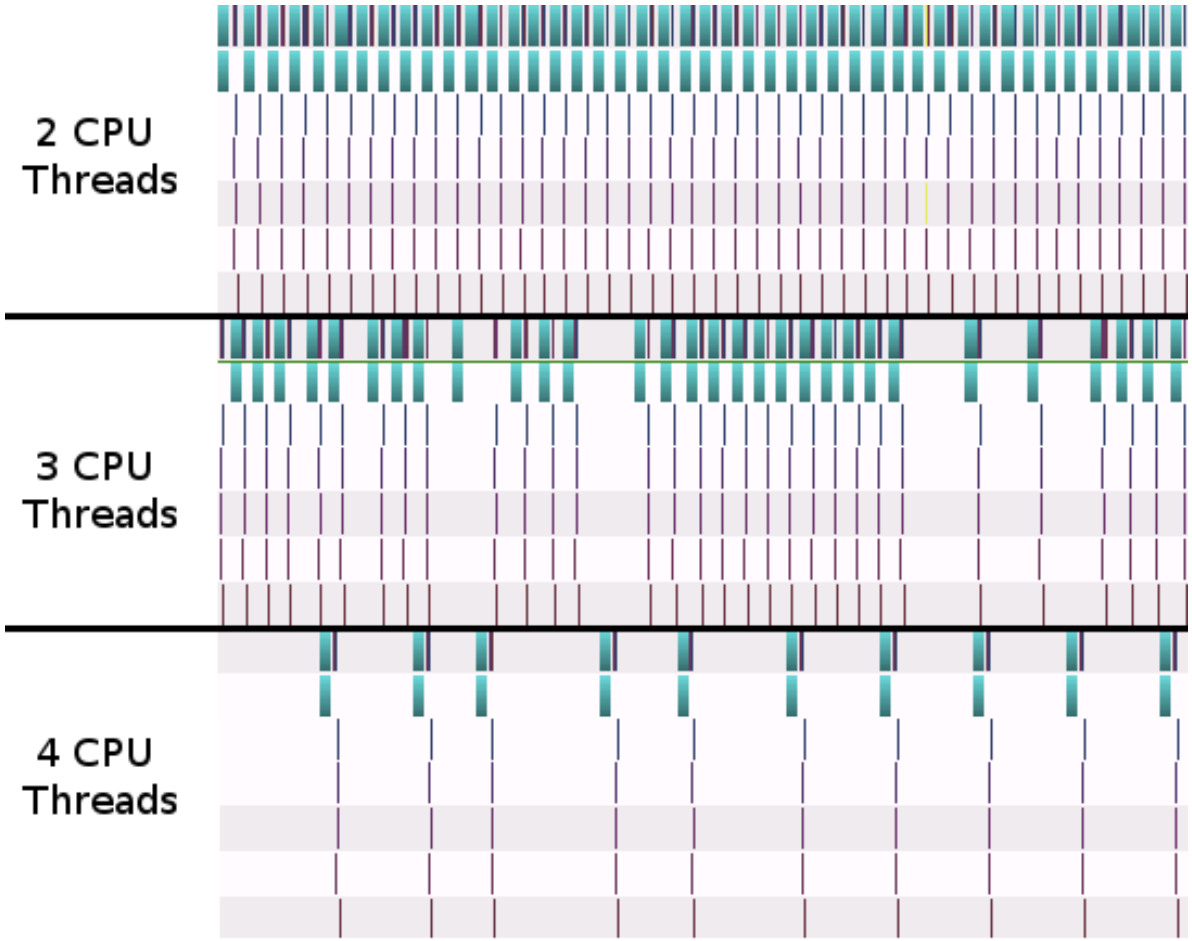
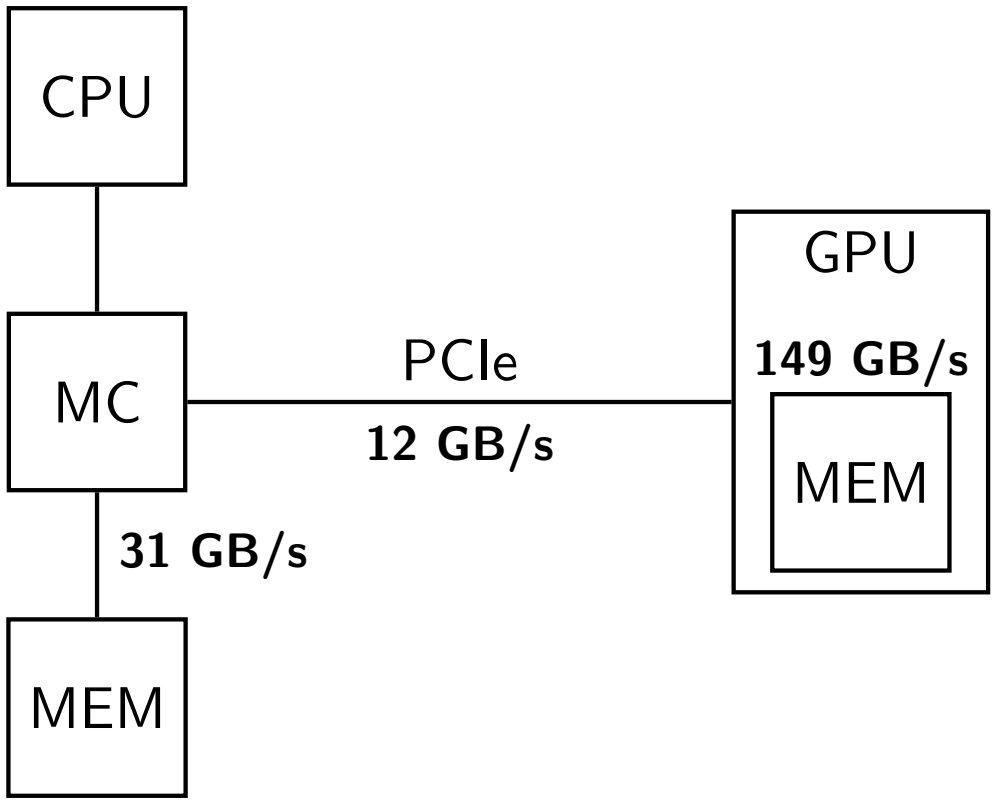


Figure : Profiling coprocessor kernel invocations

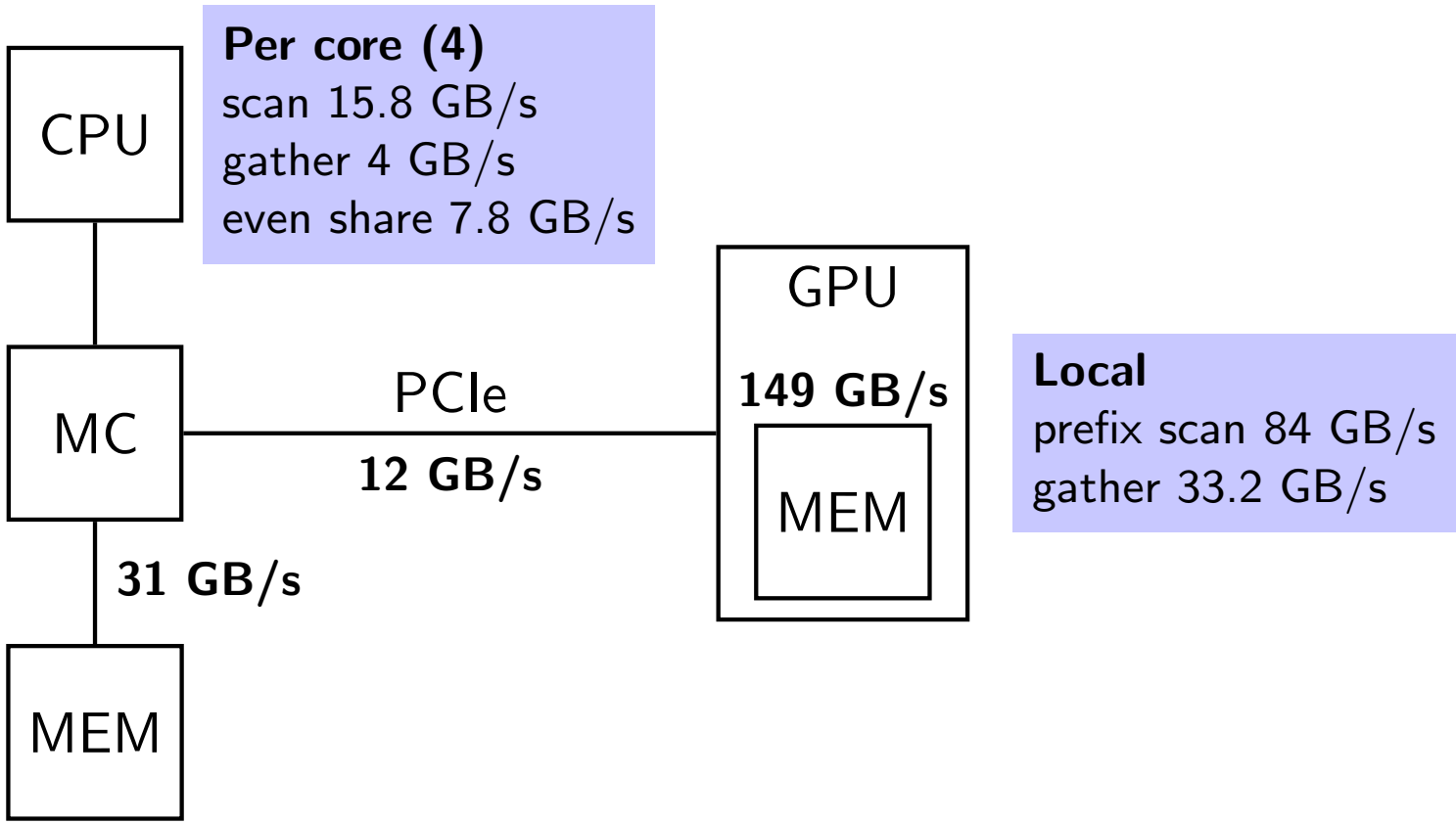
→ **Steer control flow from coprocessor itself**

→ **Increase block size**

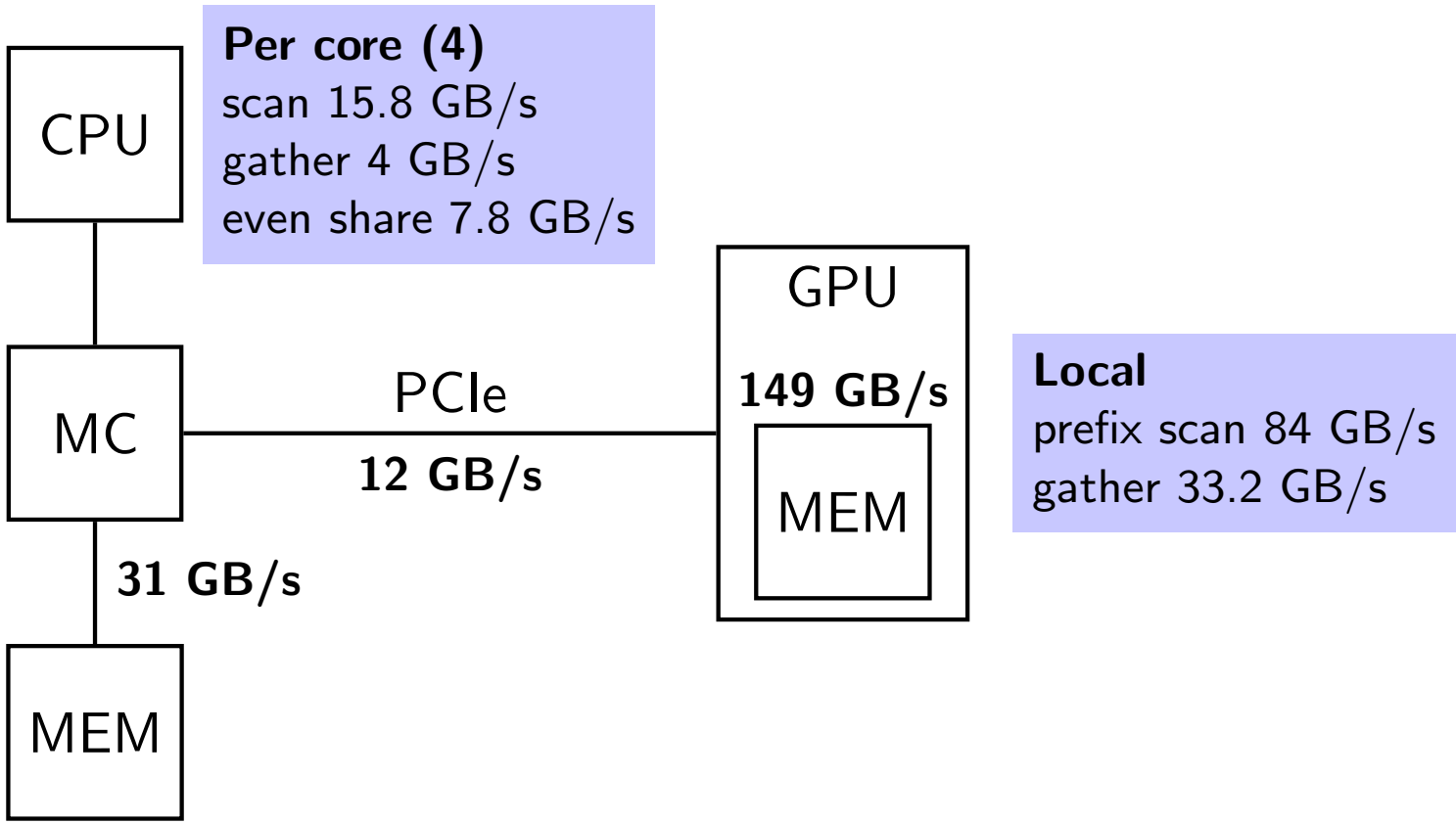
Hardware Schema – Memory Bandwidth Utilization



Hardware Schema – Memory Bandwidth Utilization



Hardware Schema – Memory Bandwidth Utilization



- PCI express bus **and** main memory can become a bottleneck
- Take bandwidth footprint and throughput into account

  → Instead of bulk processing, **apply dataflow perspective**

Improving Resource Utilization

Cache awareness

- Materialize tuples in hash table
  $\rightarrow$ Useful payload in cacheline
- Order probe data by hash function

GPU Optimizations

- Pipeline data between GPU kernels
- Concurrent kernel execution

Key Insights

- **PCIe is not the dominating bottleneck** for GPU joins
- Dataflow oriented processing
  $\rightarrow$ Join **arbitrary outer relation sizes**
- Move part of probes to coprocessor
  $\rightarrow$ **Save memory bandwidth**
  $\rightarrow$ **Gain throughput**

Future Work

- Join processing $\rightarrow$ query processing
- Compile pipelined operator sequences
- Stream arbitrary columns

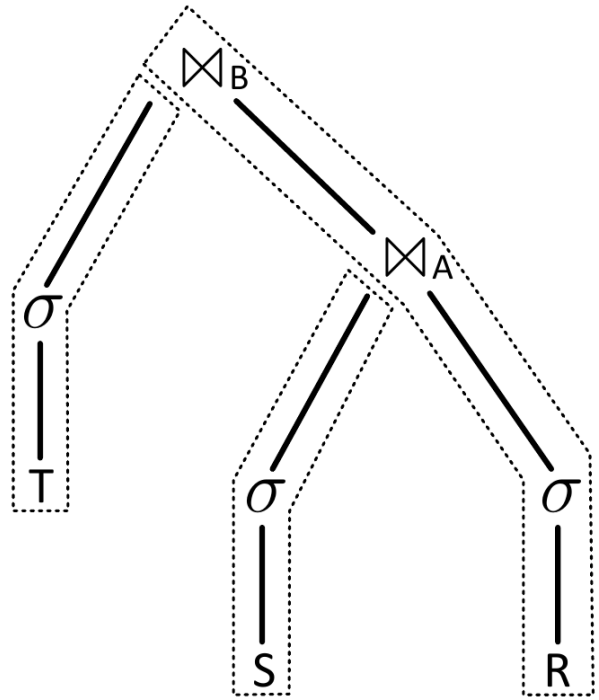


Figure : Operator pipelines. From Leis et al. *Morsel-driven parallelism* SIGMOD 2014

Key Insights

- **PCIe is not the dominating bottleneck** for GPU joins
- Dataflow oriented processing
  → Join **arbitrary outer relation sizes**
- Move part of probes to coprocessor
  → **Save memory bandwidth**
  → **Gain throughput**

Future Work

- Join processing → query processing
- Compile pipelined operator sequences
- Stream arbitrary columns

Thank you!

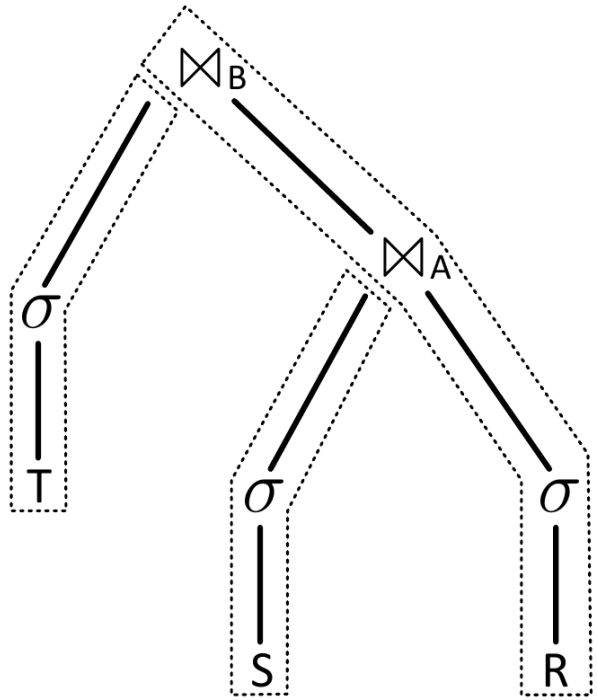


Figure : Operator pipelines. From Leis et al. *Morsel-driven parallelism* SIGMOD 2014

SFB 876 Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung