GPU Databases—The New Modality of Data Analytics

Xiangyao Yu

University of Wisconsin-Madison

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Outline

GPU hardware trend

Demo of Camelot

GPU database optimizations

- Tile-based execution **(SIGMOD 2020)**
- Data Compression **(SIGMOD 2022)**
- Heterogeneous CPU-GPU DBMS **(VLDB 2022)**
- Accelerating UDF on GPUs **(DaMoN@SIGMOD 2023)**
- Multi-GPU database (on-going)

GPU for SQL Data Analytics

GPU target applications:

- ➢ There are many, many threads
- ➢ Threads perform very similar operations
- ➢ Threads have simple control flow
- ➢ Threads are mostly independent (minimal synchronization)

GPU for SQL Data Analytics

GPU target applications:

- There are many, many threads
- Threads perform very similar operations
- Threads have simple control flow
- Threads are mostly independent (minimal synchronization)

What about SQL data analytics???

GPU for SQL Data Analytics

GPU target applications:

- ➢There are many, many threads ✓
- ➢Threads perform very similar operations ✓
- ➢Threads have simple control flow ✓
- ➢Threads are mostly independent (minimal synchronization) ✓

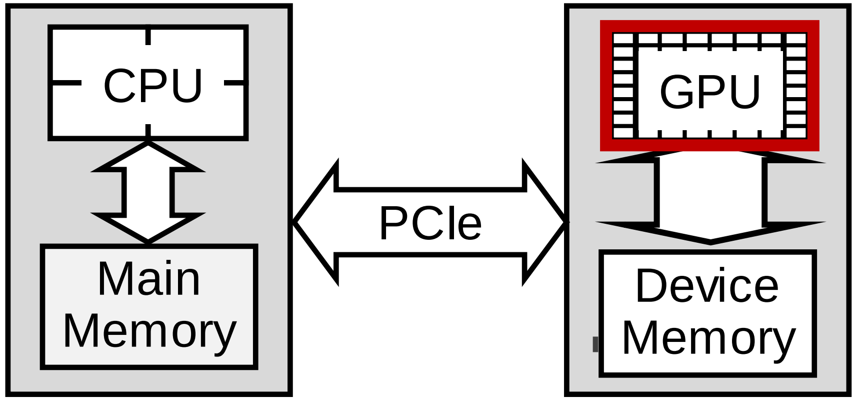
What about SQL data analytics???

GPU is very suitable for SQL data analytics

Running SQL analytics on GPUs can give 10–25x speedup over CPU [1]

[1] Anil Shanbhag, Samuel Madden, Xiangyao Yu, *A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics*, **SIGMOD 2020**

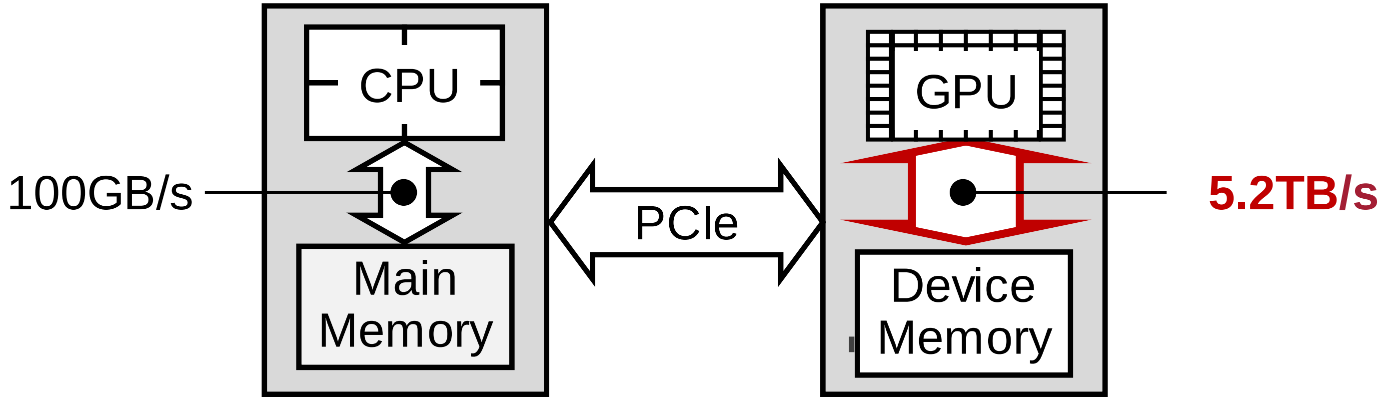
Advantages of GPU for Data Analytics



Advantage 1: **Higher computation power**

- GPU has massive parallelism using SIMT model

Advantages of GPU for Data Analytics

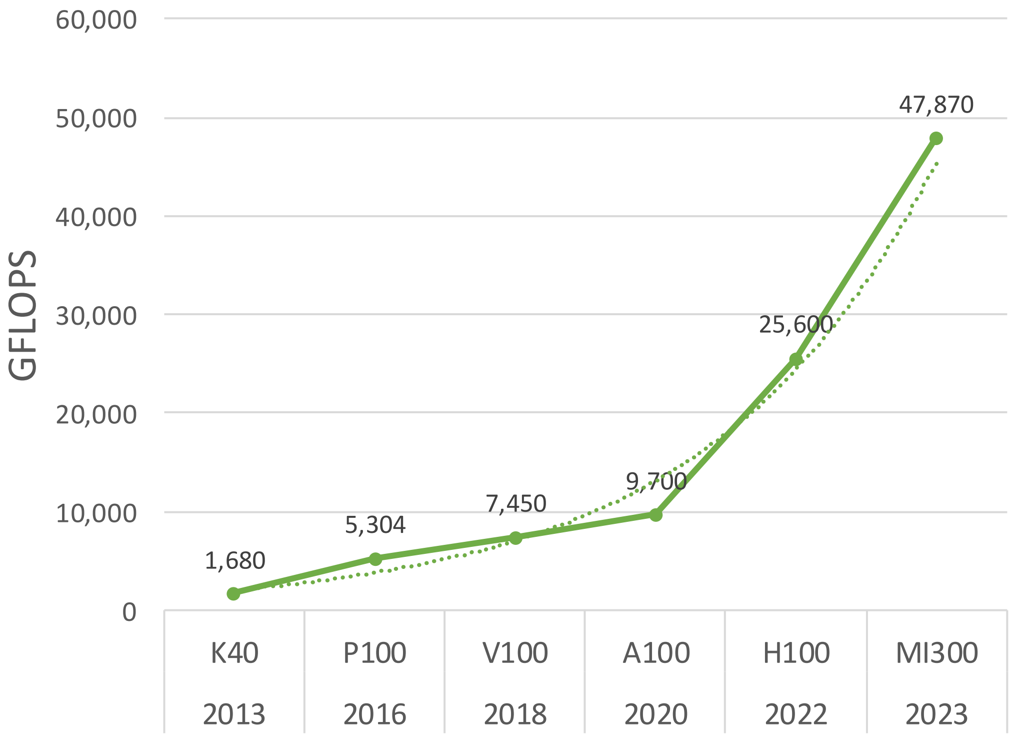


Advantage 1: **Higher computation power**

Advantage 2: **Higher memory bandwidth**

- GPU memory bandwidth is an order-of-magnitude higher than CPU.

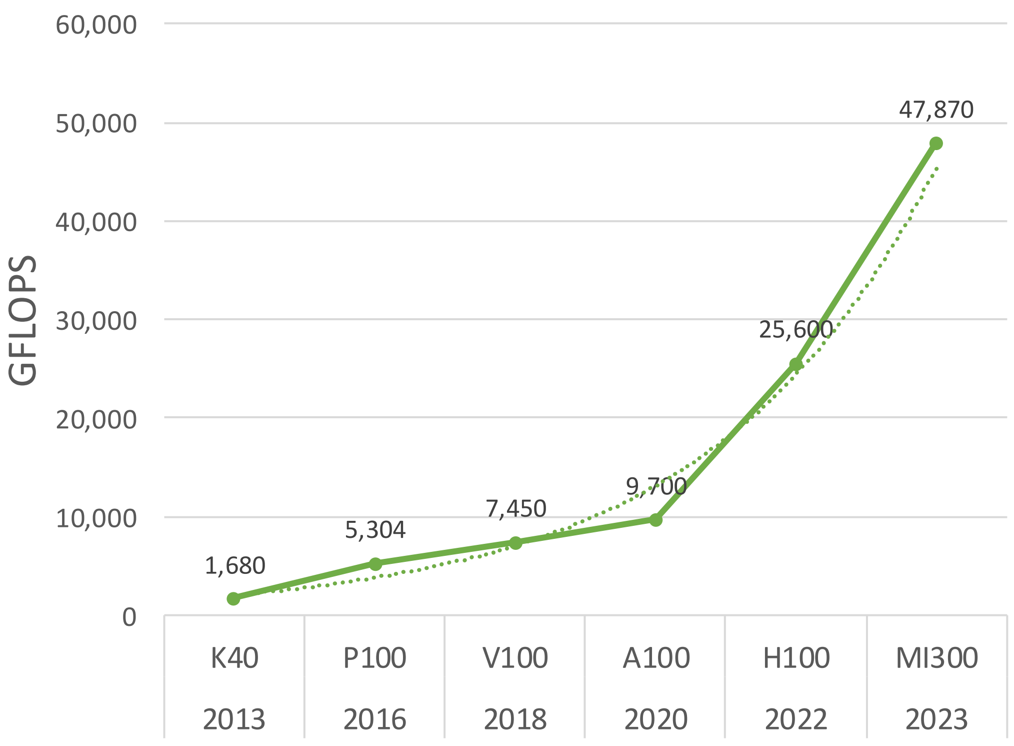
GPU Trend



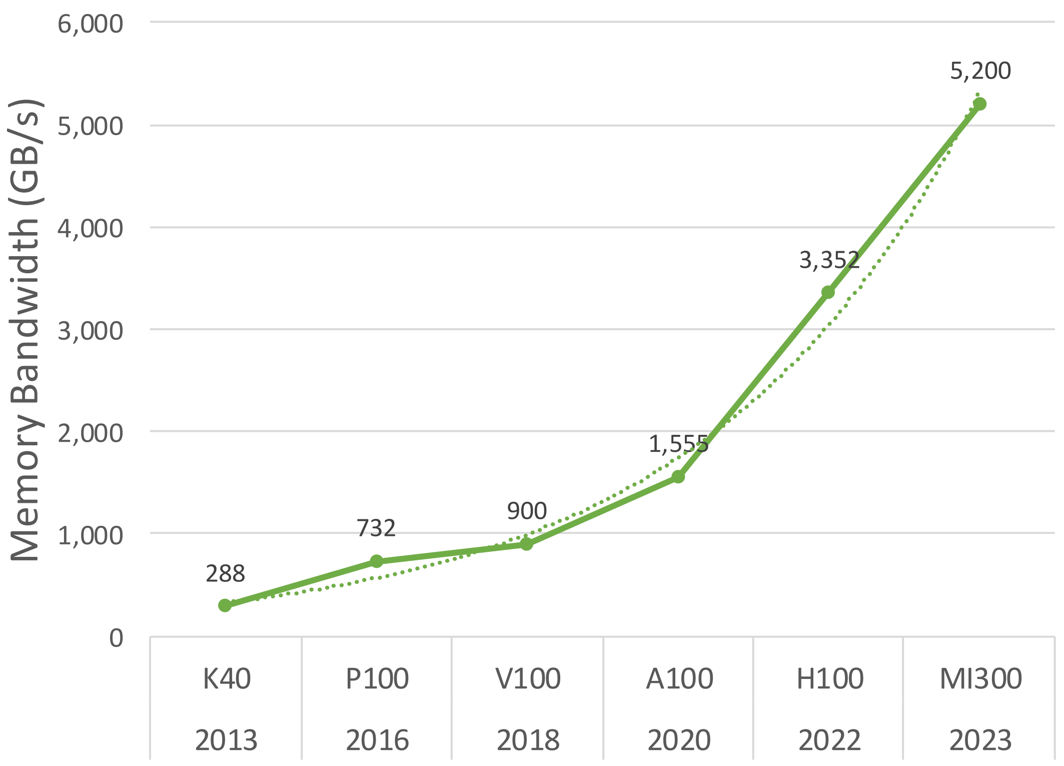
(a) GPU Peak Performance

GPU **peak performance increase by 5x** from 2020 to 2023**.**

GPU Trend



(a) GPU Peak Performance

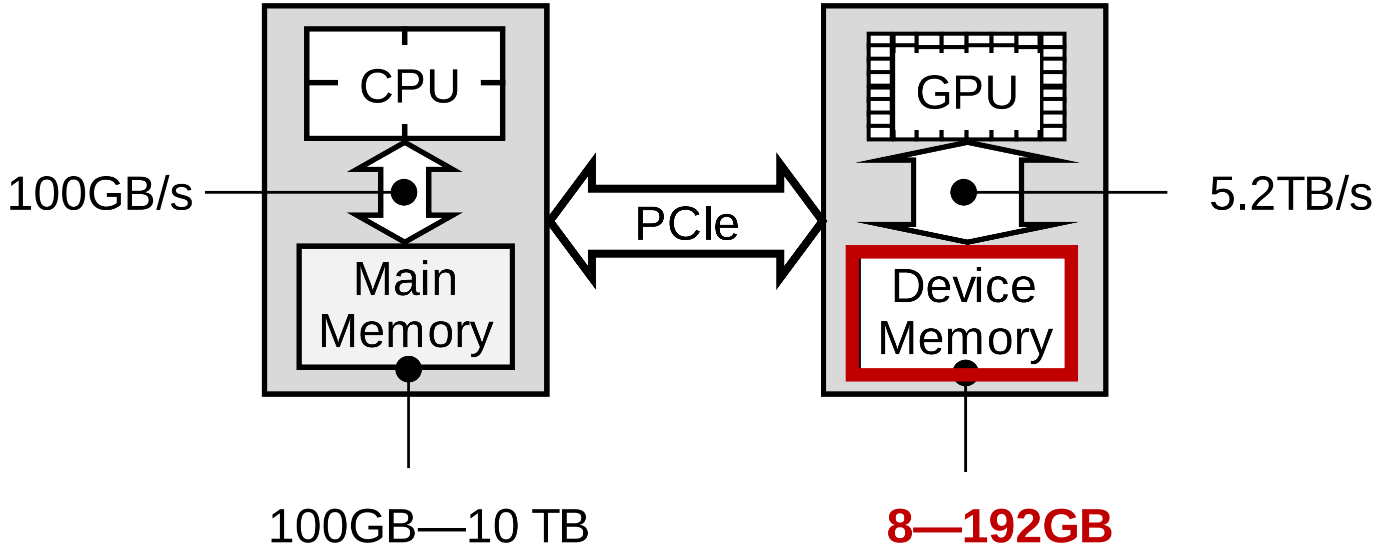


(b) GPU Memory Bandwidth

GPU **peak performance increase by 5x** from 2020 to 2023.

GPU **memory bandwidth increase by 3.5x** from 2020 to 2023.

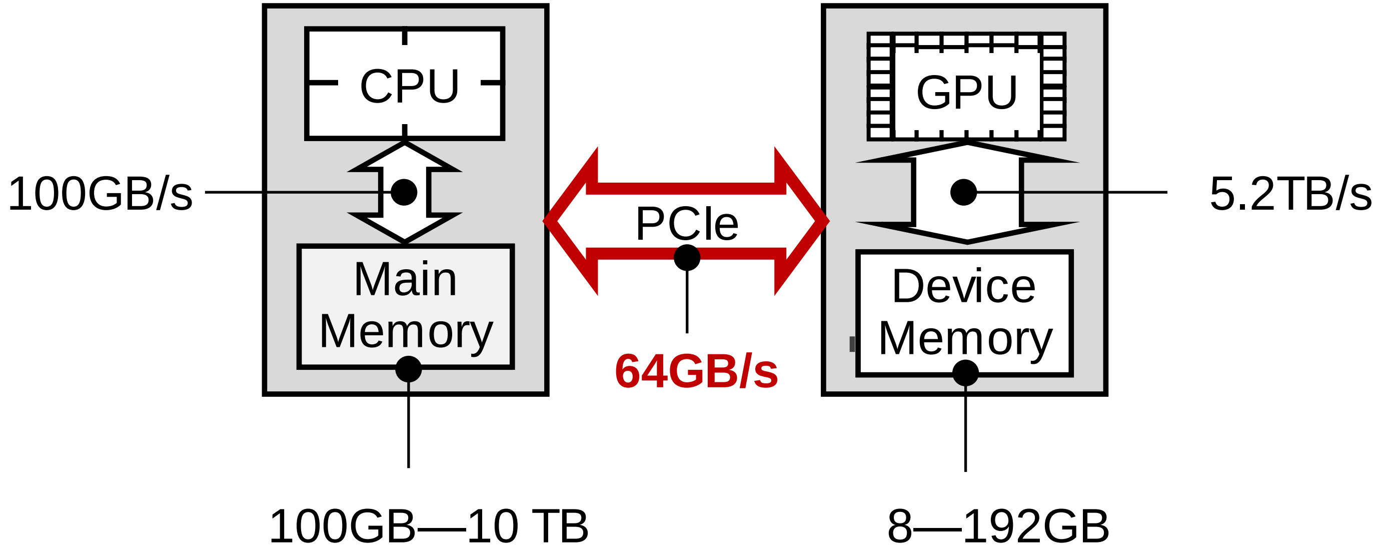
Challenges of GPU for Data Analytics



Challenge 1: **Limited memory capacity**

- Some data sets do not fit in GPU memory

Challenges of GPU for Data Analytics

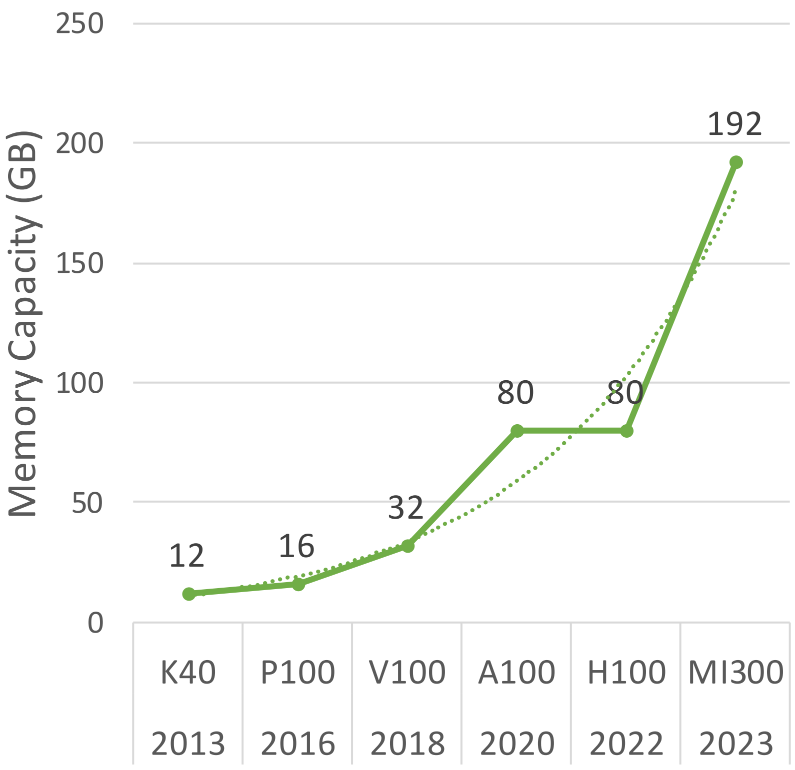


Challenge 1: **Limited memory capacity**

Challenge 2: **Limited interconnect bandwidth**

- Transferring data from CPU can be expensive

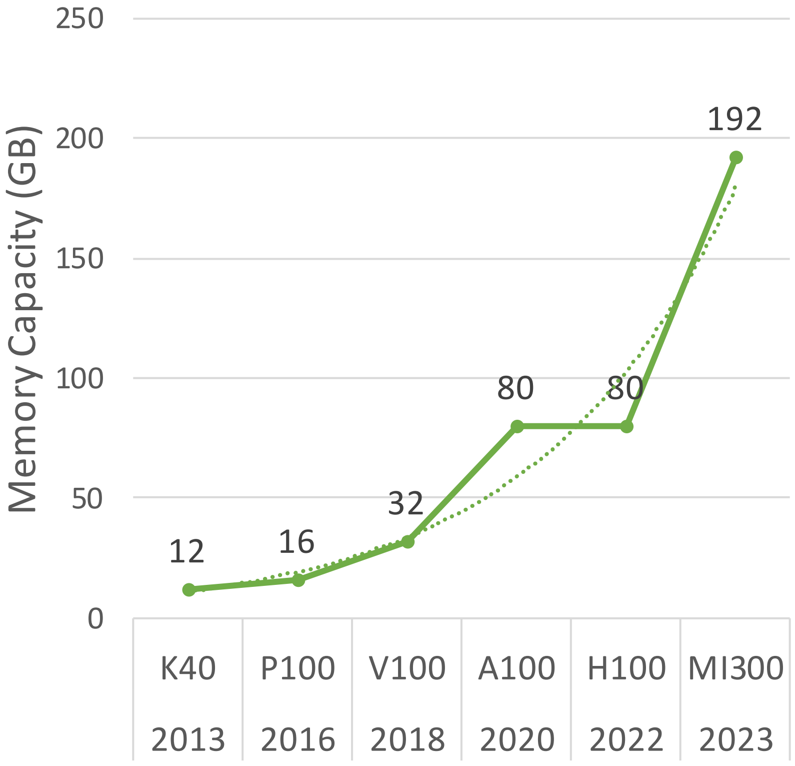
GPU Trend



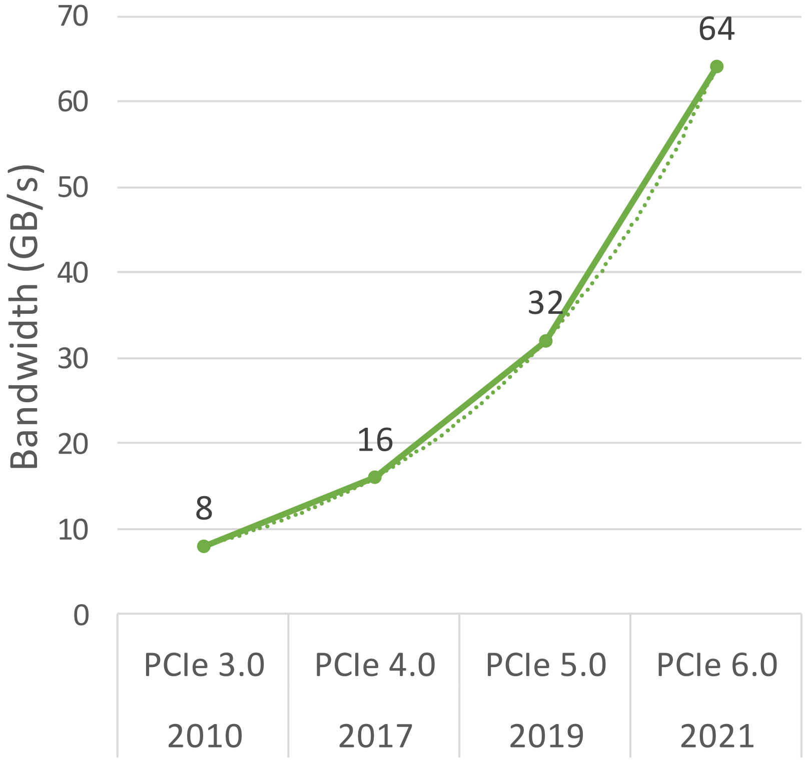
(a) GPU Memory Capacity

GPU **memory capacity increase by 6x** in the last 5 years**.**

GPU Trend



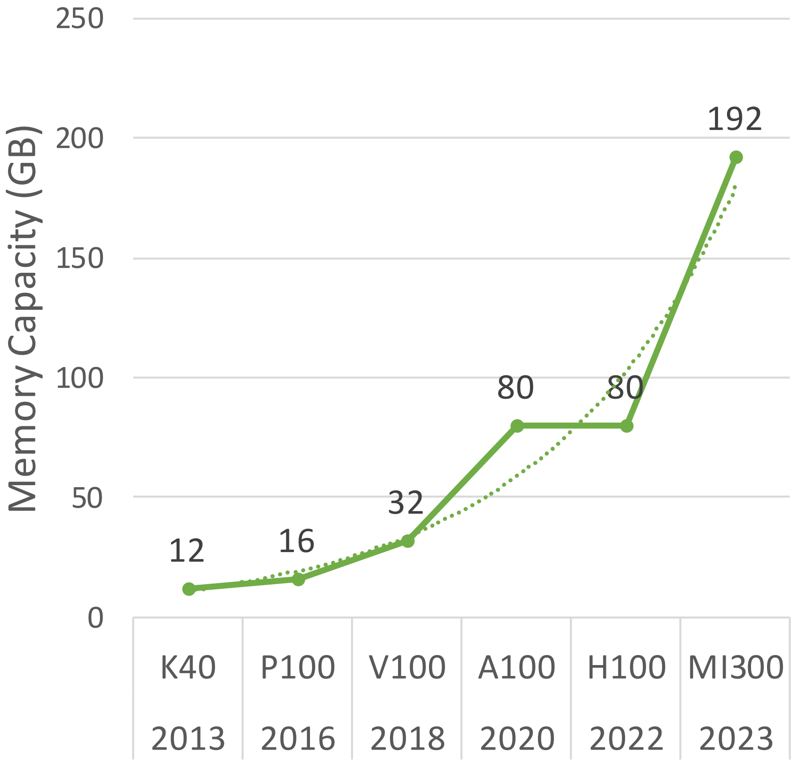
(a) GPU Memory Capacity



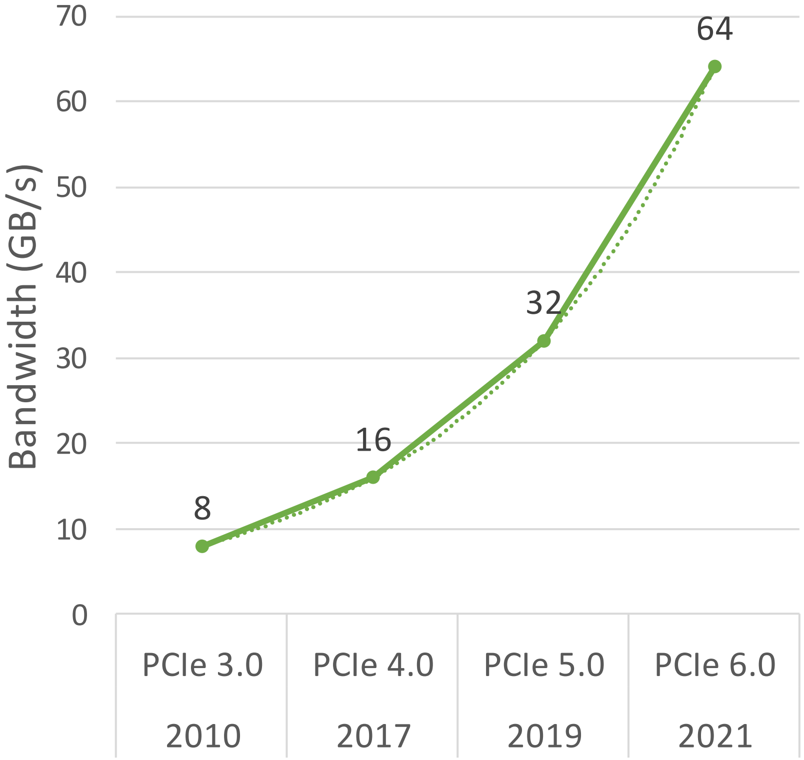
(b) PCIe Bandwidth

GPU **memory capacity increase by 6x** in the last 5 years**.**
PCIe **increase by 2x** every two years.

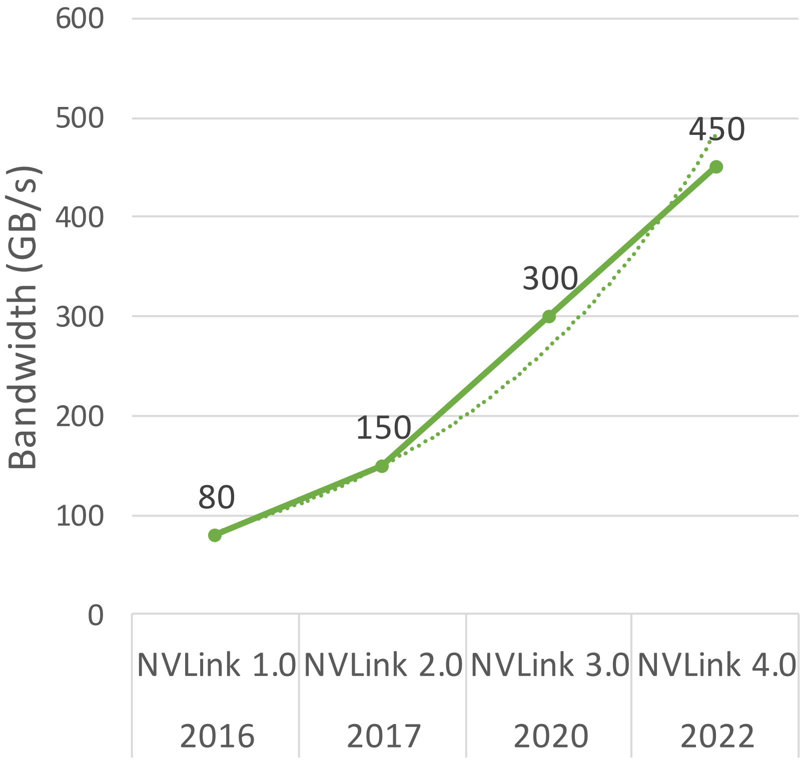
GPU Trend



(a) GPU Memory Capacity



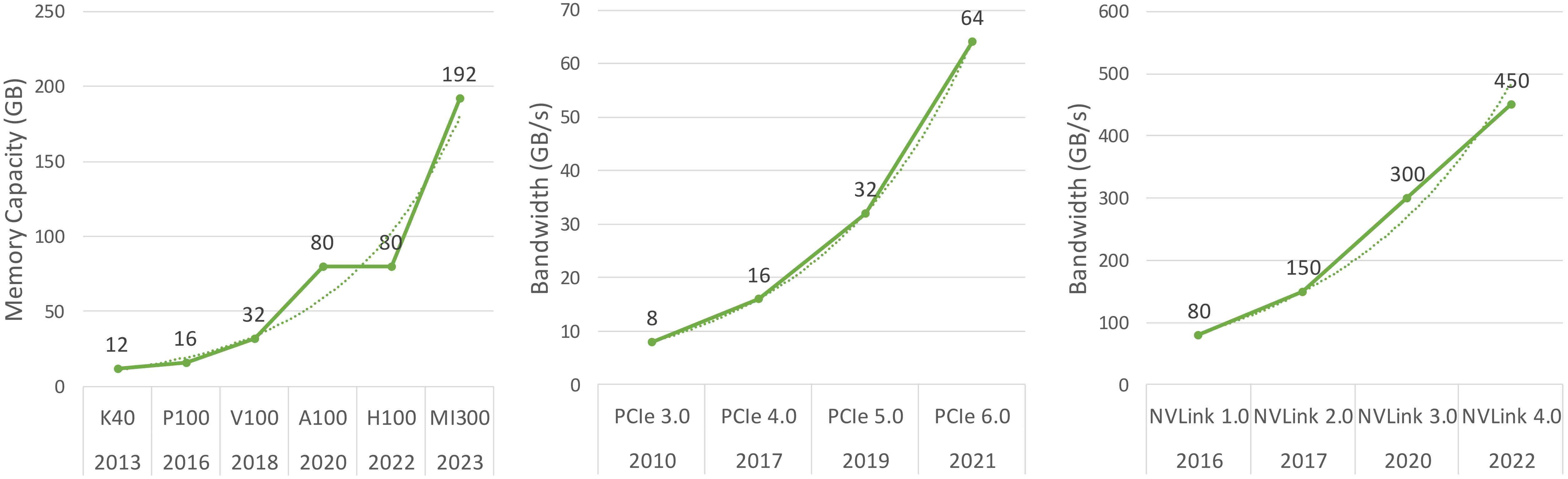
(b) PCIe Bandwidth



(c) NVLink Bandwidth

GPU **memory capacity increase by 6x** in the last 5 years**.**
PCIe **increase by 2x** every two years.
NVLink Bandwidth **increase by 3x in 5 years.**

GPU Trend



(a) GPU Memory Capacity **(b) PCIe Bandwidth** **(c) NVLink Bandwidth**

GPU **memory capacity increase by 6x** in the last 5 years**.**

PCIe **increase by 2x** every two years.

NVLink Bandwidth **increase by 3x in 5 years.**

NVLink C2C (2022) connect NVIDIA GPU and NVIDIA CPU **(450 GB/s).**

Outline

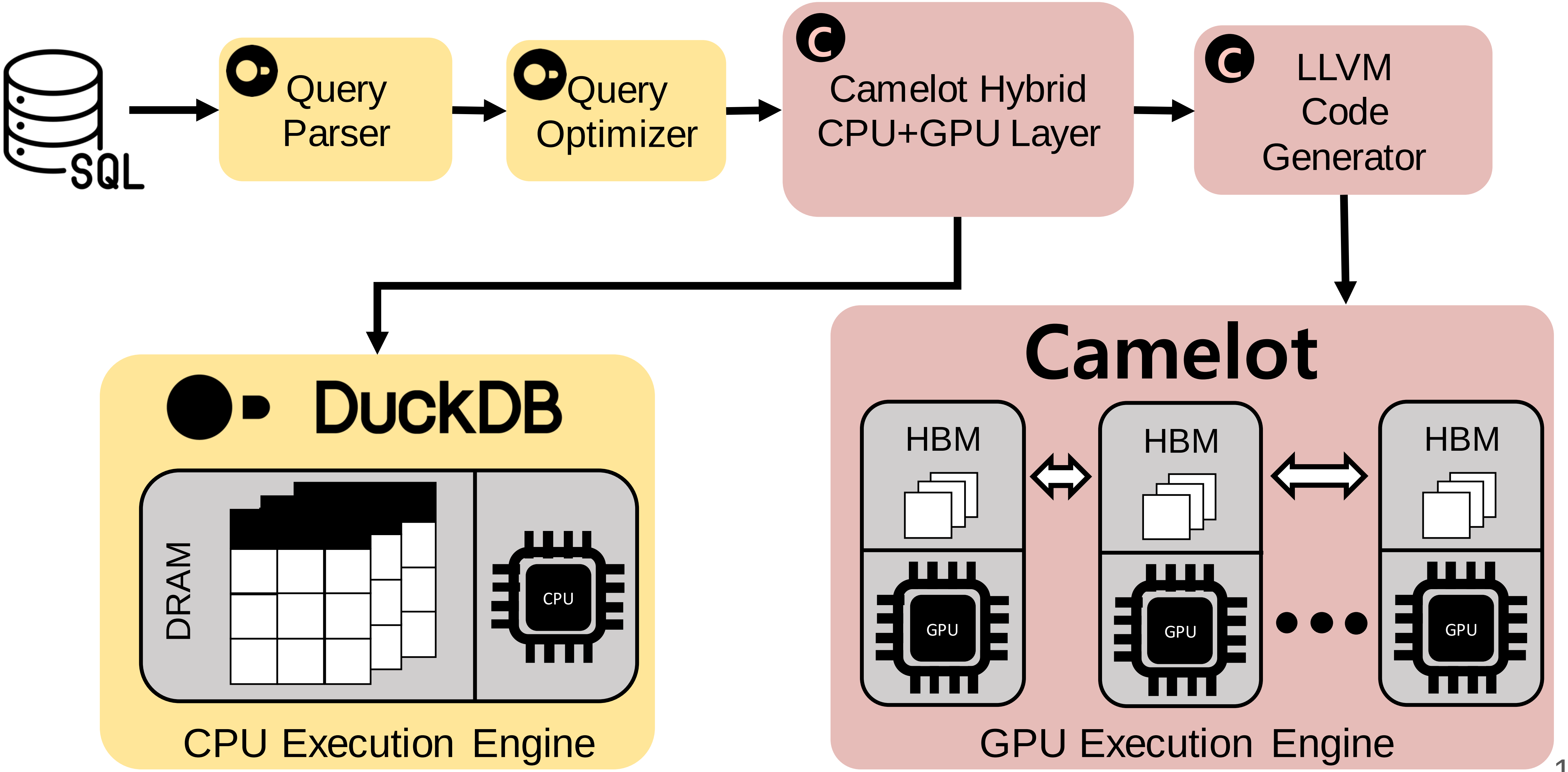
GPU hardware trend

Demo of Camelot

GPU database optimizations

- Tile-based execution **(SIGMOD 2020)**
- Data Compression **(SIGMOD 2022)**
- Heterogeneous CPU-GPU DBMS **(VLDB 2022)**
- Accelerating UDF on GPUs **(DaMoN@SIGMOD 2023)**
- Multi-GPU database (on-going)

Camelot v0.1 → Demo!



Outline

GPU hardware trend

Demo of Camelot

GPU database optimizations

- Tile-based execution **(SIGMOD 2020)**
- Data Compression **(SIGMOD 2022)**
- Heterogeneous CPU-GPU DBMS **(VLDB 2022)**
- Accelerating UDF on GPUs **(DaMoN@SIGMOD 2023)**
- Multi-GPU database (on-going)

GPU Database Optimizations

Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- Crystal: tile-based execution **(SIGMOD 2020[1])**
- Data Compression **(SIGMOD 2022[2])**
- Heterogeneous CPU-GPU DBMS **(VLDB 2022[3])**
- Multi-GPU DBMS (ongoing)

Thrust 2: Enhancing the **Practicality** of GPU Databases

- Accelerating UDF on GPUs **(DaMoN@SIGMOD 2023[4])**
- Code Generation for GPU DBMS (ongoing)

[1] Anil Shanbhag, Samuel Madden, Xiangyao Yu. *A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics*, **SIGMOD 2022**
[2] Anil Shanbhag*, Bobbi Yogatama*, Xiangyao Yu, and Samuel Madden. *Tile-based Lightweight Integer Compression in GPU*, **SIGMOD 2022**
[3] Bobbi Yogatama, Weiwei Gong, Xiangyao Yu. *Orchestrating data placement and query execution in heterogeneous CPU-GPU DBMS*, **VLDB 2022**
[4] Bobbi Yogatama et al. *Accelerating User-Defined Aggregate Functions with Block-wide Execution and JIT Compilation on GPUs*, **DaMoN@SIGMOD 2023**

GPU Database Optimizations

Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- Crystal: tile-based execution **(SIGMOD 2020[1])**
- Data Compression **(SIGMOD 2022[2])**
- Heterogeneous CPU-GPU DBMS **(VLDB 2022[3])**
- Multi-GPU DBMS (ongoing)

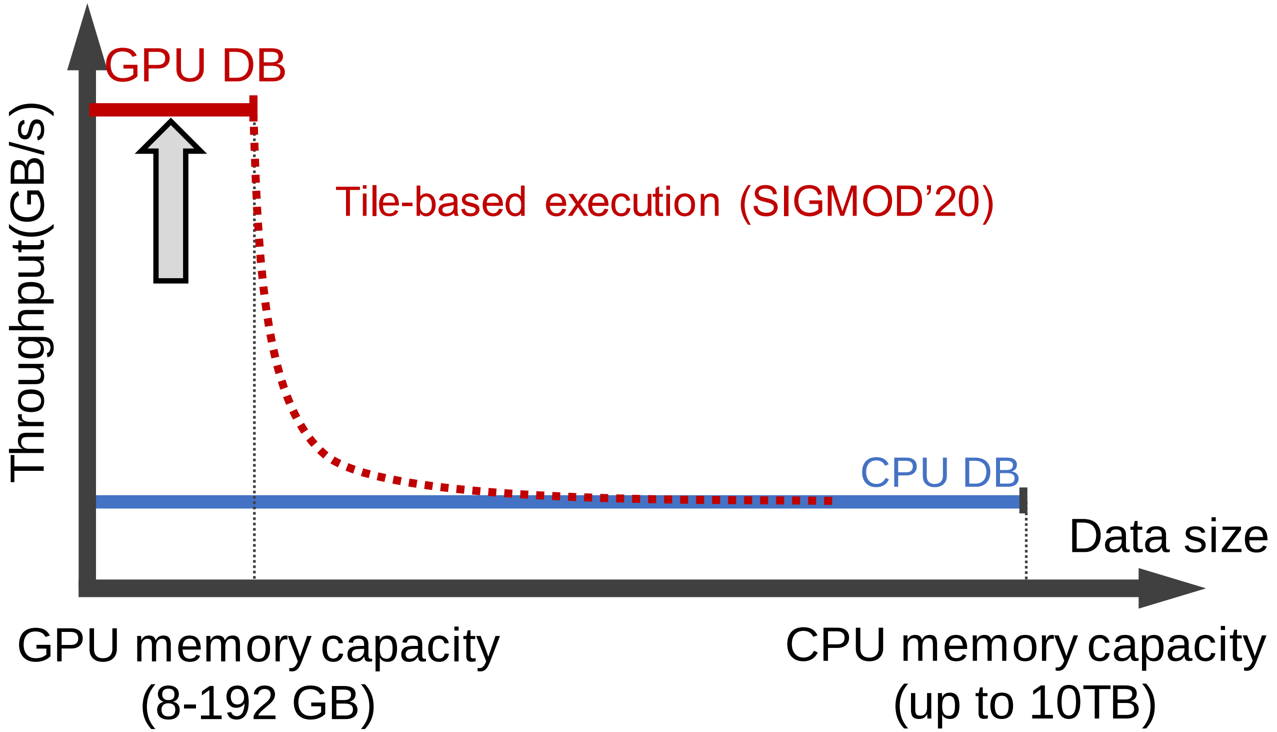
Thrust 2: Enhancing the **Practicality** of GPU Databases

- Accelerating UDF on GPUs **(DaMoN@SIGMOD 2023[4])**
- Code Generation for GPU DBMS (ongoing)

GPU Database Optimizations

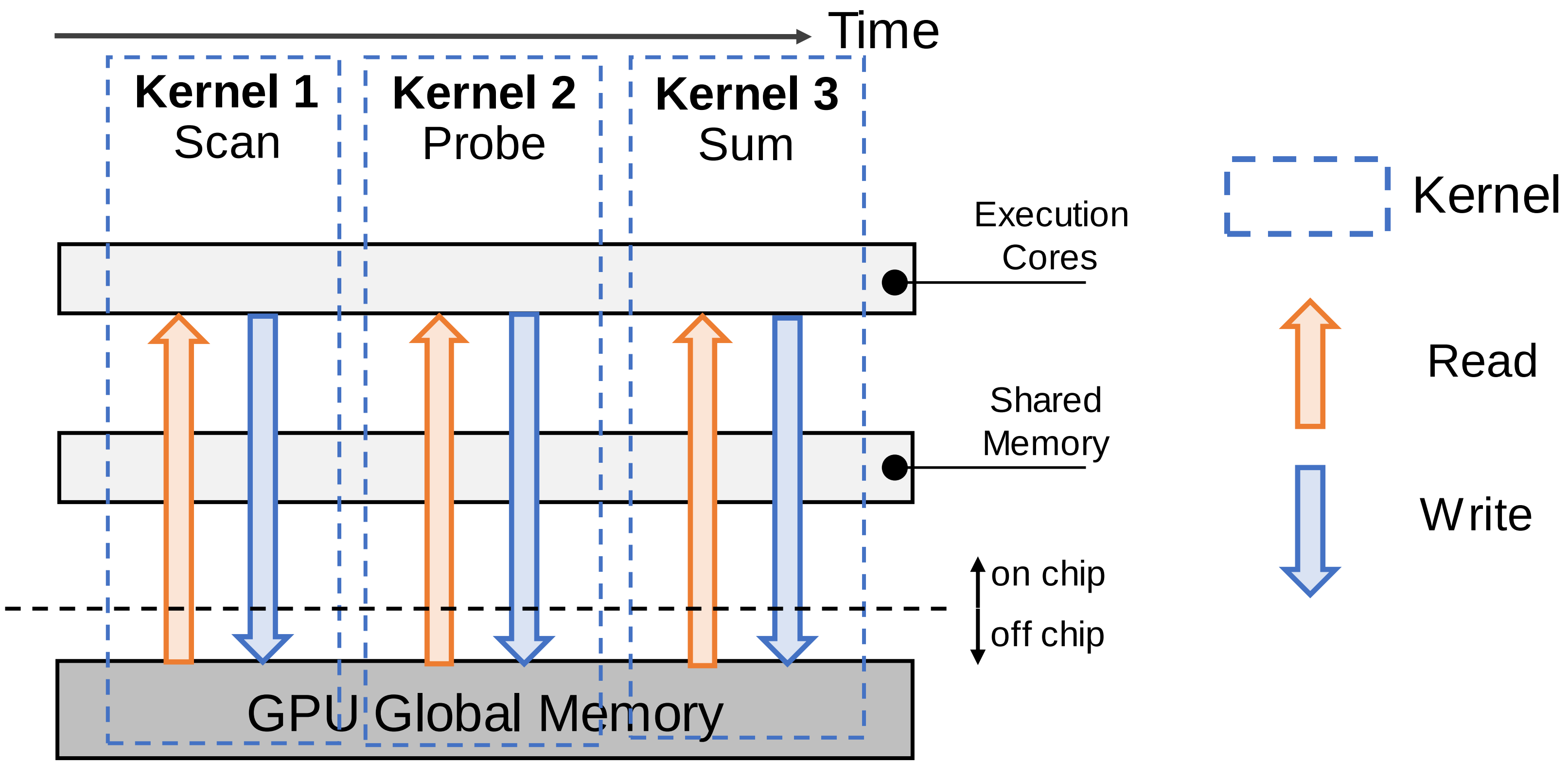
Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- **Crystal: tile-based execution (SIGMOD 2020[1])**
- Data Compression (SIGMOD 2022[2])
- Heterogeneous CPU-GPU DBMS (VLDB 2022[3])
- Multi-GPU DBMS (ongoing)



Crystal Library: Tile-Based Execution Model

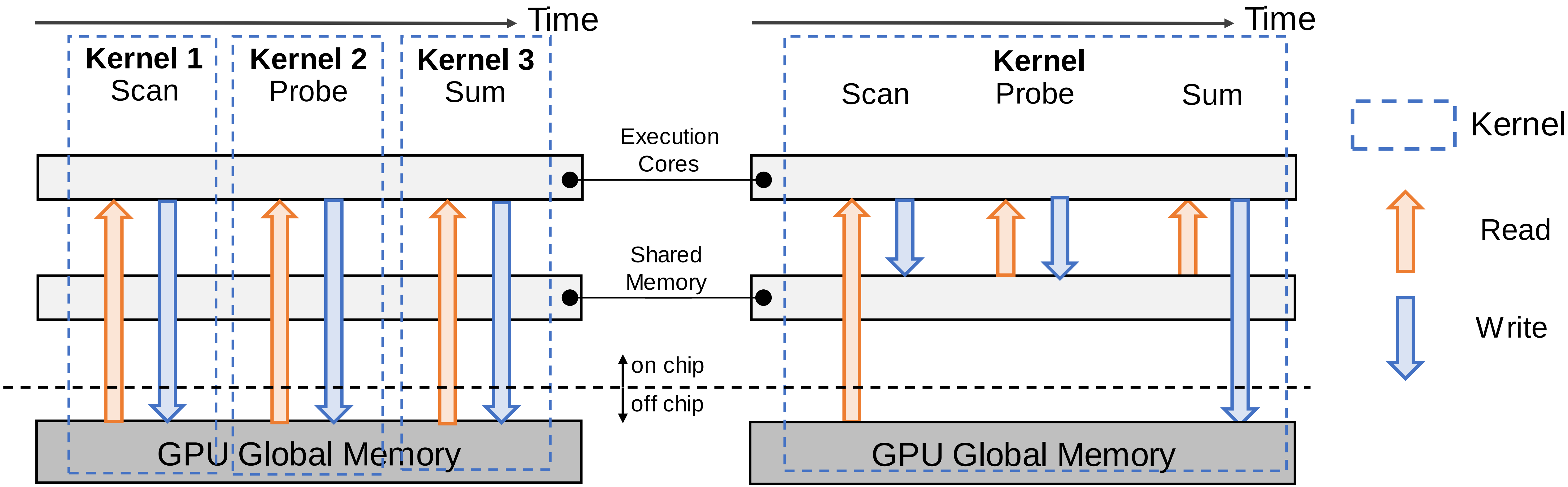
Problem: Conventional execution model incurs excessive memory traffic for reading/writing intermediate results



(a) Conventional execution model

Crystal Library: Tile-Based Execution Model

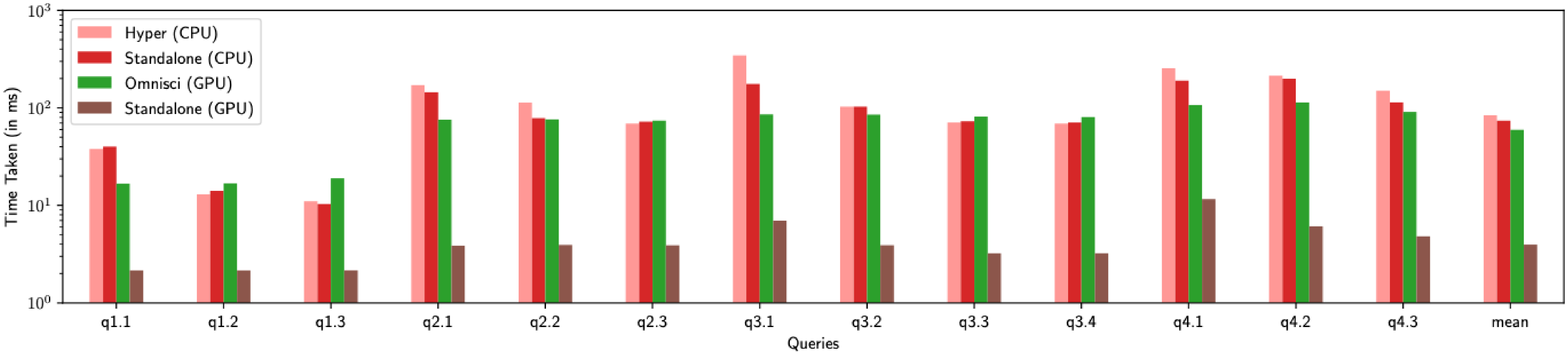
Key Idea: Partition data into small tiles and store intermediate results in the shared memory (~10x faster)



(a) Conventional execution model

(b) Tile-based execution model

Experimental Results



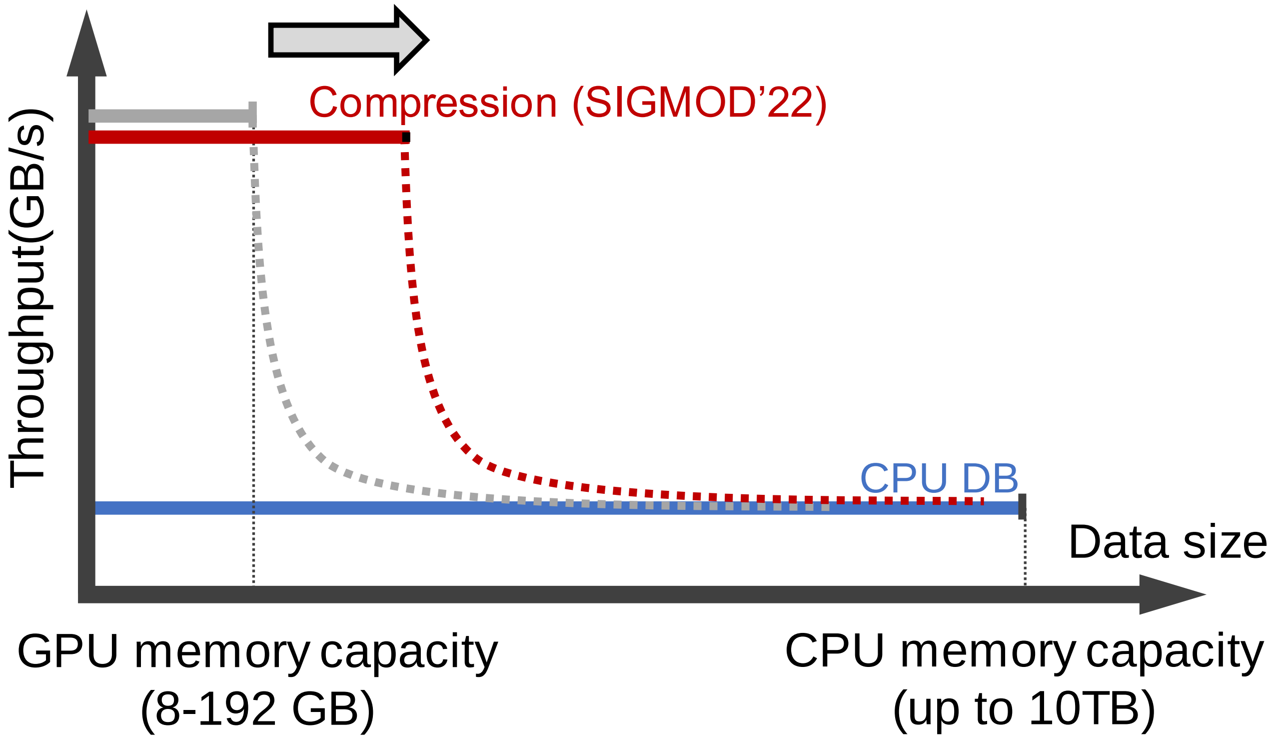
With Crystal, GPU is on average **25X** faster than CPU running Star-Schema Benchmark (SSB).

Hardware: NVIDIA V100 GPU. Intel i7-6900 CPU (8 cores)
Benchmark: Star Schema Benchmark (SF = 20).

GPU Database Optimizations

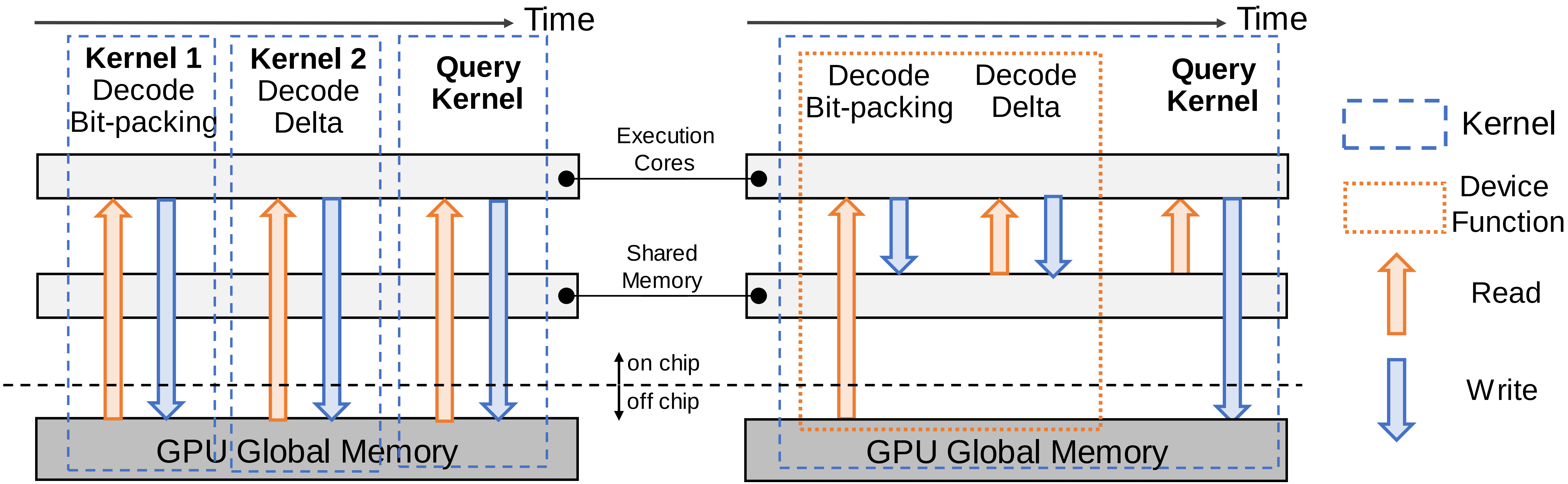
Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- Tile-based execution (SIGMOD 2020[1])
- **Data Compression (SIGMOD 2022[2])**
- Heterogeneous CPU-GPU DBMS (VLDB 2022[3])
- Multi-GPU DBMS (ongoing)



Idea 1: Tile-Based Decompression

Tile-based execution to keep intermediate results in shared memory



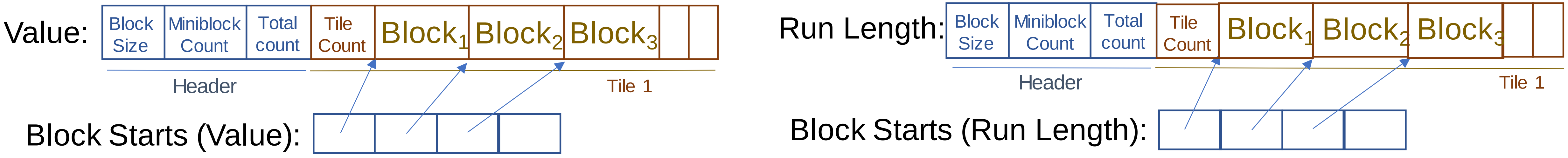
(a) Conventional decompression model

(b) Tile-based decompression model

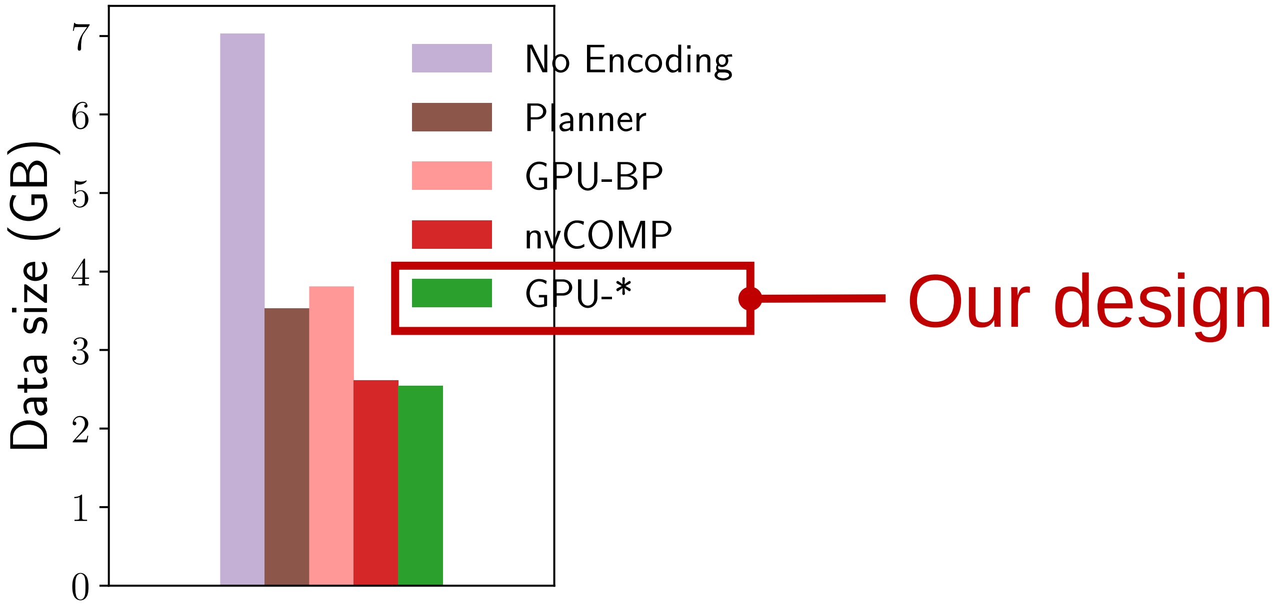
Idea 2: GPU-Optimized Compression Format

Compact data format that can **fully saturates GPU memory bandwidth** during decompression

- **GPU-FOR**: Frame of Reference + **Bit-Packing**
- **GPU-DFOR**: **Delta encoding** + Frame of Reference + **Bit-Packing**
- **GPU-RFOR**: **Run-length encoding** + Frame of Reference + **Bit-Packing**



GPU Data Compression – Evaluation

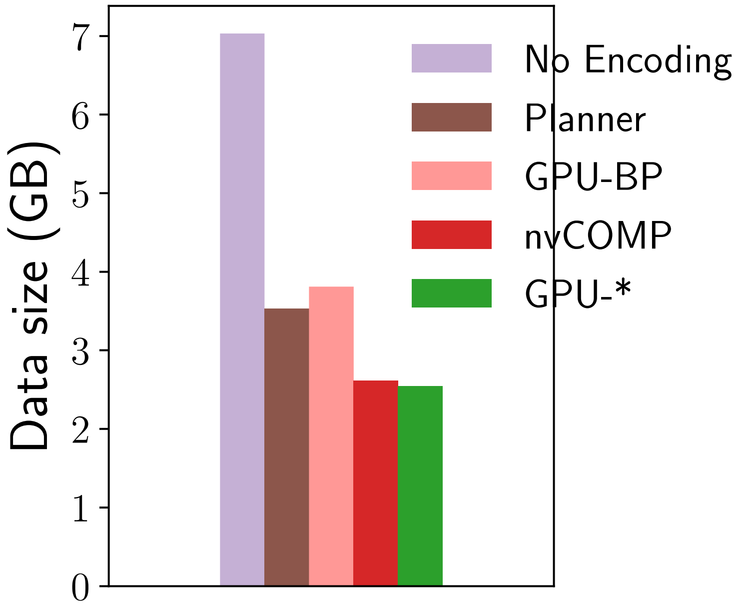


(a) Compressed data size

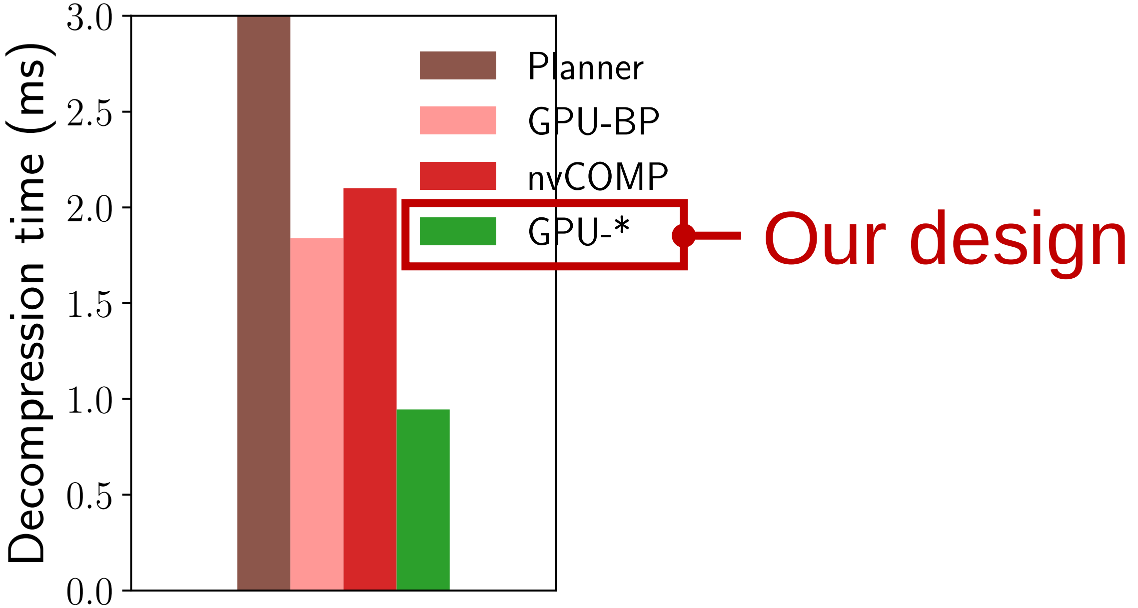
Compression rate comparable to the best-previous scheme (i.e. nvCOMP)

Hardware: NVIDIA V100 GPU.
Benchmark: Star Schema Benchmark (SF = 20).

GPU Data Compression – Evaluation



(a) Compressed data size



(b) Decompression time

Compression rate comparable to the best-previous scheme (i.e. nvCOMP)

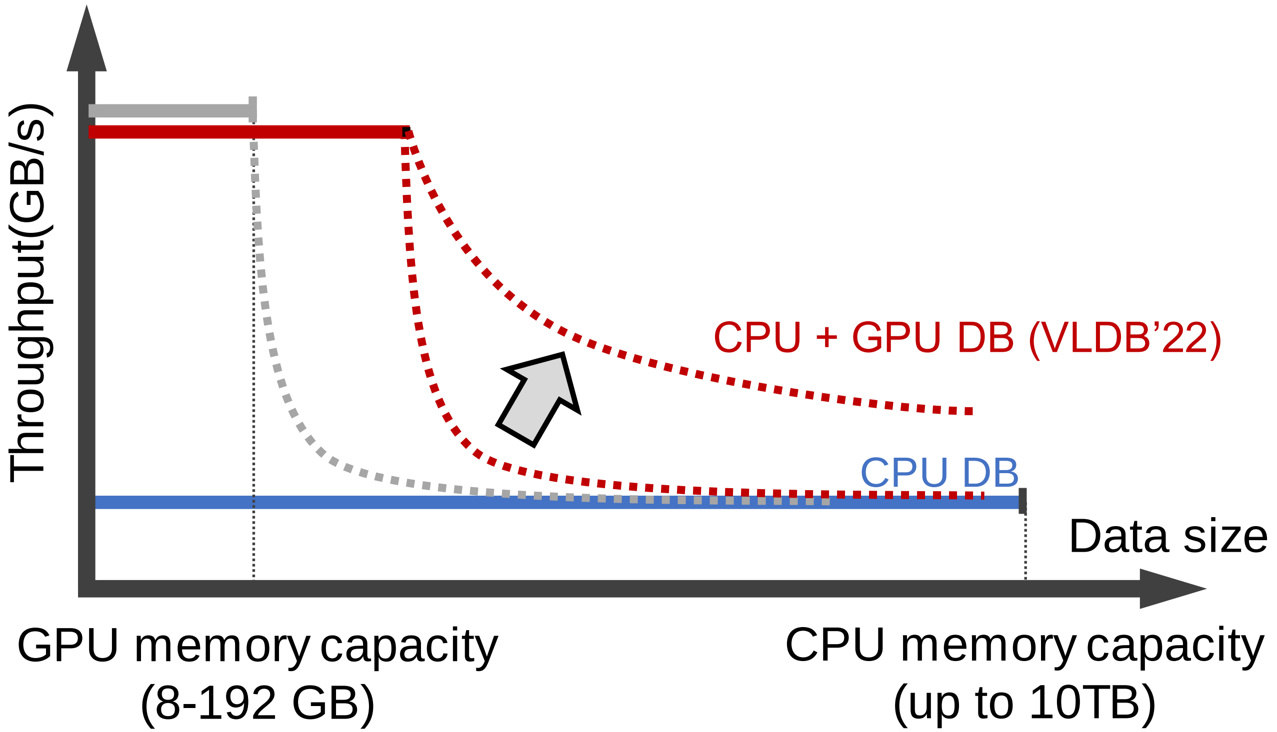
Decompression time is 2.2x faster than the best-previous scheme

Hardware: NVIDIA V100 GPU.
Benchmark: Star Schema Benchmark (SF = 20).

GPU Database Optimizations

Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- Tile-based execution (SIGMOD 2020[1])
- Data Compression (SIGMOD 2022[2])
- **Heterogeneous CPU-GPU DBMS (VLDB 2022[3])**
- Multi-GPU DBMS (ongoing)



Challenges in Heterogeneous CPU-GPU Model

We aim to answer the **existing challenges** in heterogeneous CPU-GPU DBMS:

1. **Data Placement**

   → How do we partition data between CPU and GPU?

2. **Heterogeneous Query Execution**

   → How to coordinate query execution between CPU and GPU?

Challenges in Heterogeneous CPU-GPU Model

We aim to answer the **existing challenges** in heterogeneous CPU-GPU DBMS:

1. **Data Placement**

   → How do we partition data between CPU and GPU?

2. **Heterogeneous Query Execution**

   → How to coordinate query execution between CPU and GPU?

Data Placement

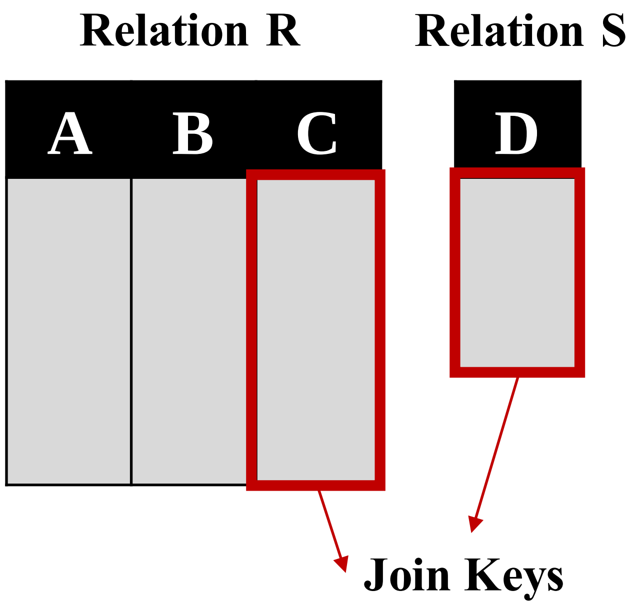
We treat data placement as a **caching problem** → the complete data set resides in CPU memory and a mirrored subset of data is **cached** in GPU.

Key design decision: **cache replacement policy**?

Data Placement

We treat data placement as a **caching problem** → the complete data set resides in CPU memory and a mirrored subset of data is **cached** in GPU.

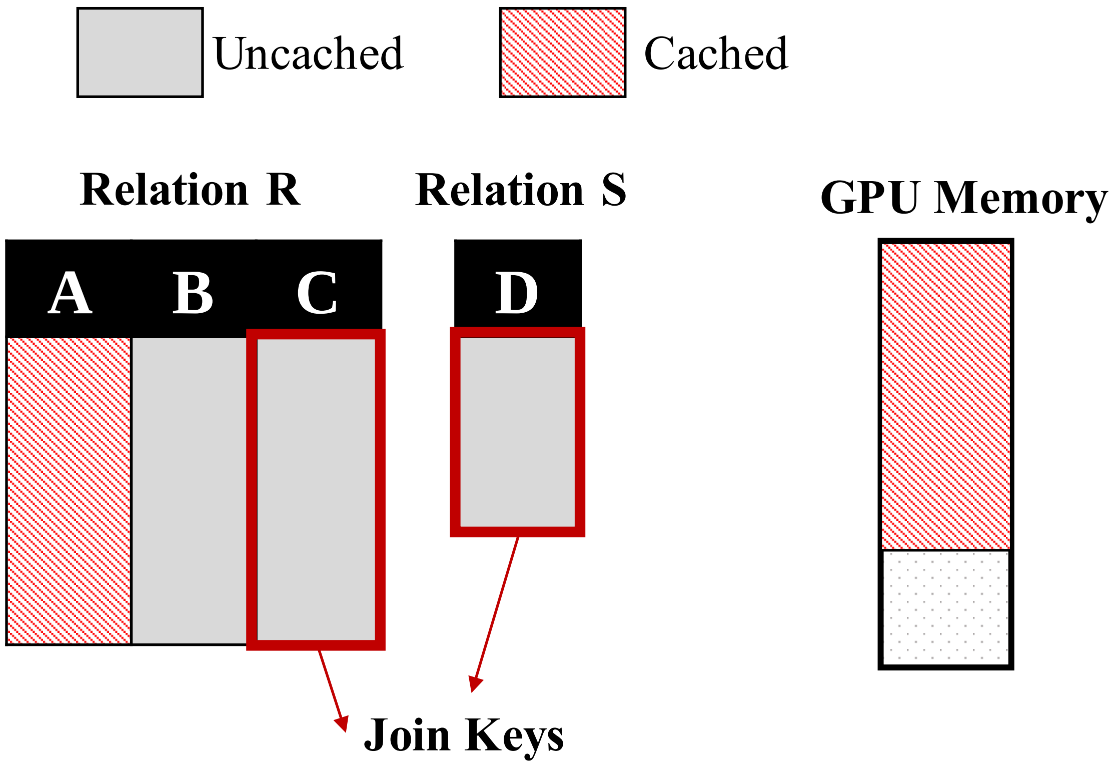
Key design decision: **cache replacement policy**?



Data Placement

Key design decision: **cache replacement policy**?

Previous works did a **column-granularity frequency-based/timestamp-based** policy.



(a) Coarse-grained caching (LFU/LRU)[2]

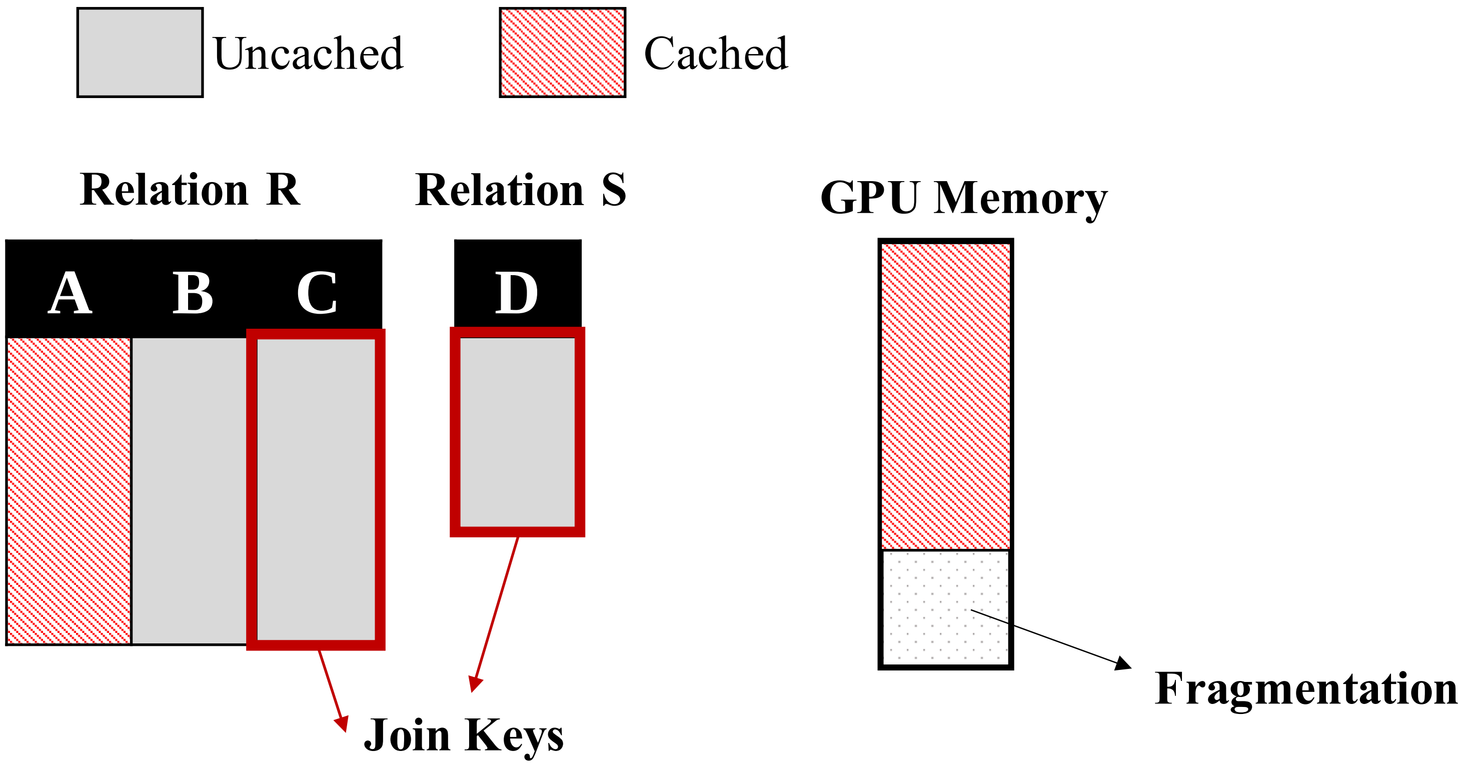
[2] Sebastian Breß, Henning Funke, and Jens Teubner. *Robust Query Processing in Co-Processor-Accelerated Databases*, **SIGMOD 2016.**

Data Placement

Key design decision: **cache replacement policy**?

Previous works did a **column-granularity frequency-based/timestamp-based** policy.

Limitation: Fragmentation

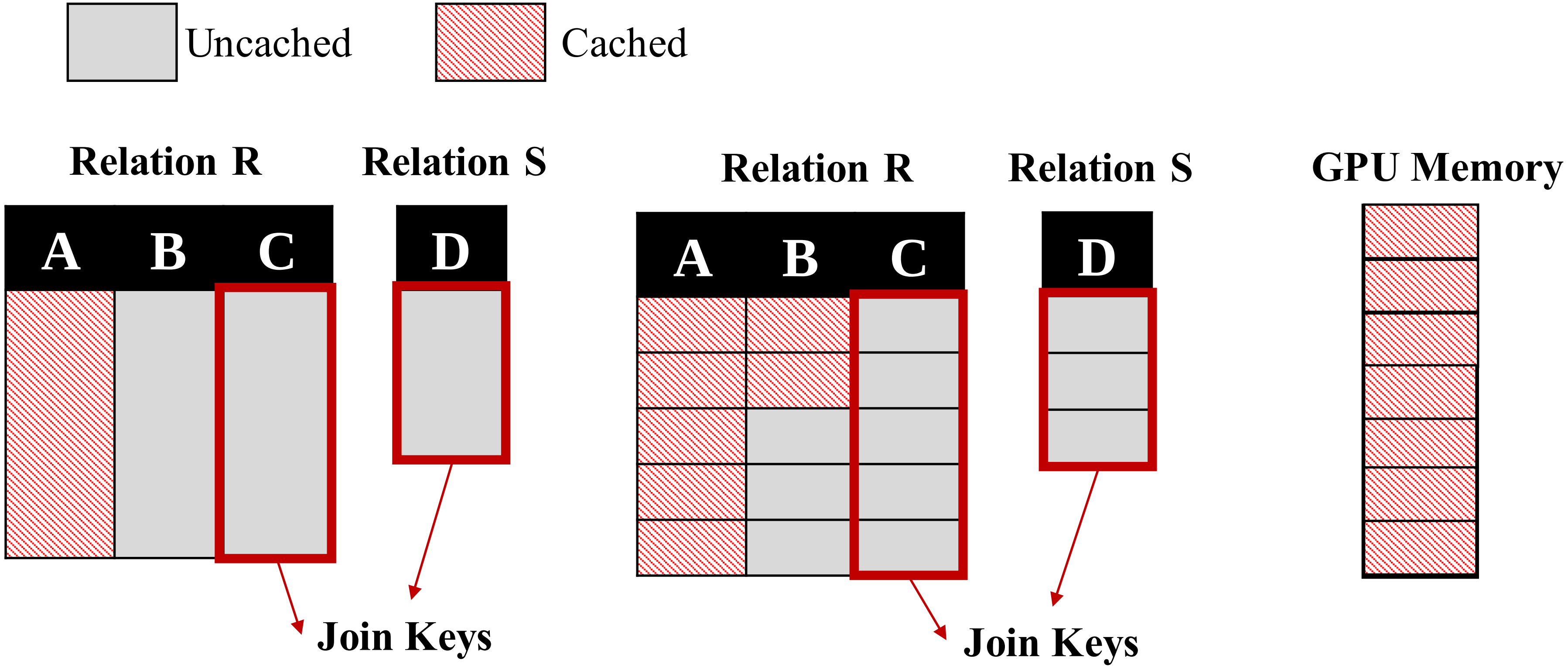


(a) Coarse-grained caching (LFU/LRU)

Data Placement

Key design decision: **cache replacement policy**?

A sub-column (segment) **fine-grained** policy can improve caching efficiency.



(a) Coarse-grained caching (LFU/LRU) (b) Fine-grained caching (LFU/LRU)[3]

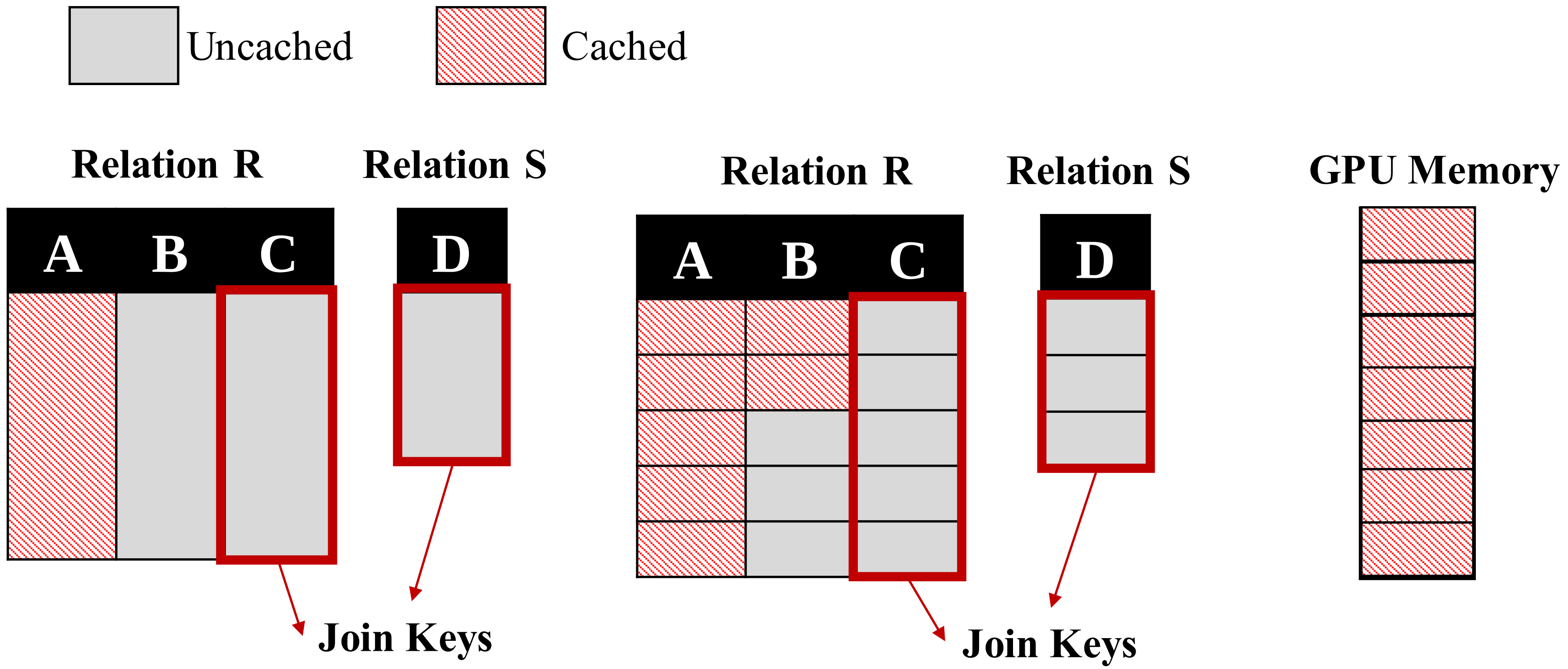
[3] Todd Mostak. *An Overview of MapD (Massively Parallel Database)*.

Data Placement

Key design decision: **cache replacement policy**?

A sub-column (segment) **fine-grained** policy can improve caching efficiency.

Limitation: unaware of **query semantic**.

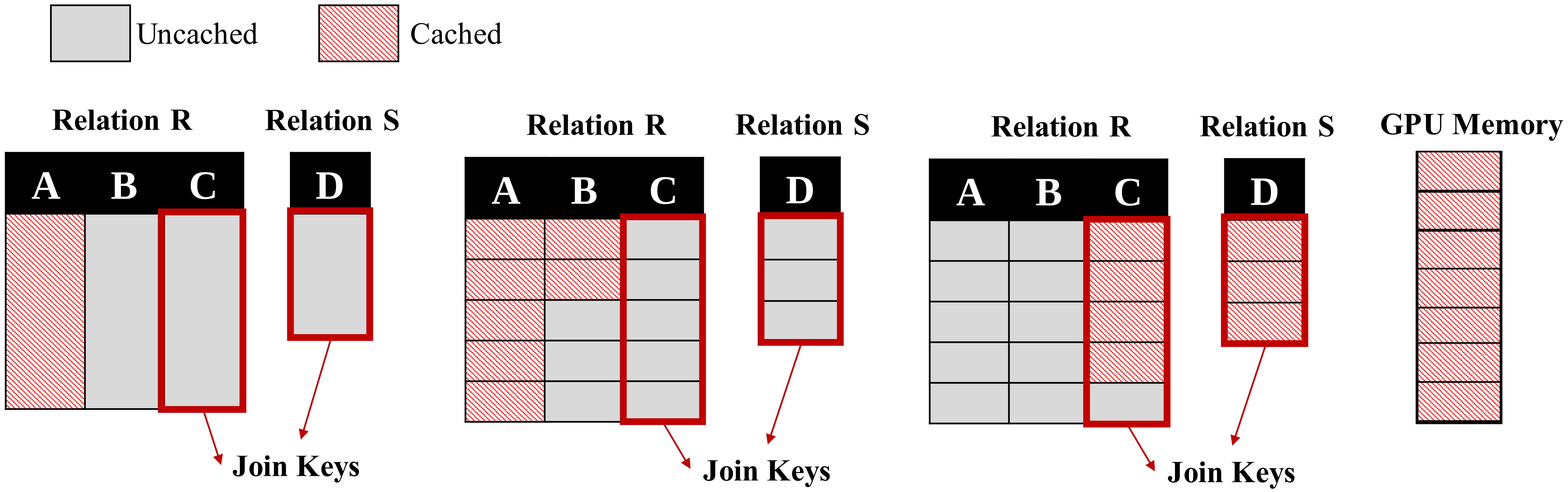


(a) Coarse-grained caching (LFU/LRU) (b) Fine-grained caching (LFU/LRU)

Data Placement

Key design decision: **cache replacement policy**?

A sub-column (segment) **fine-grained** policy can improve caching efficiency.
Semantic-aware replacement leads to better performance.



(a) Coarse-grained caching (LFU/LRU) (b) Fine-grained caching (LFU/LRU) (c) Fine-grained + semantic-aware caching

Semantic-Aware Fine-Grained Caching

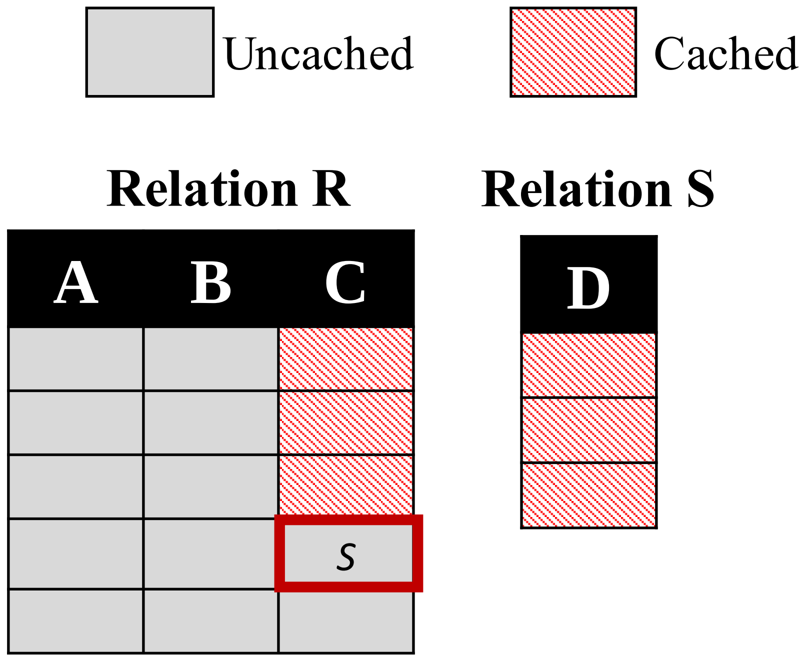
Extend conventional LFU with **weighted frequency counters.**

Weight reflects the potential **benefits of caching the segments** and is derived using cost model.

Semantic-Aware Fine-Grained Caching

Extend conventional LFU with **weighted frequency counters.**

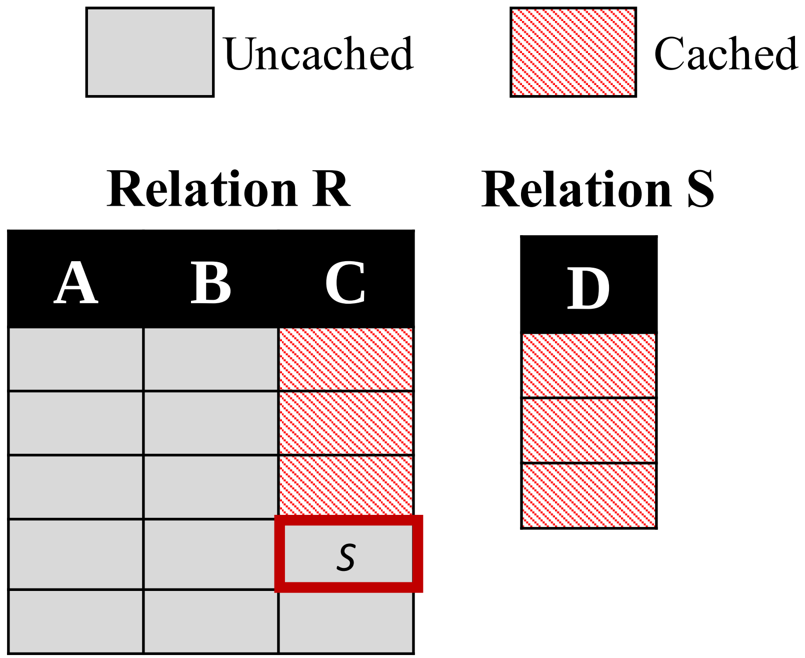
Weight reflects the potential **benefits of caching the segments** and is derived using cost model.



Semantic-Aware Fine-Grained Caching

Extend conventional LFU with **weighted frequency counters.**

Weight reflects the potential **benefits of caching the segments** and is derived using cost model[1].



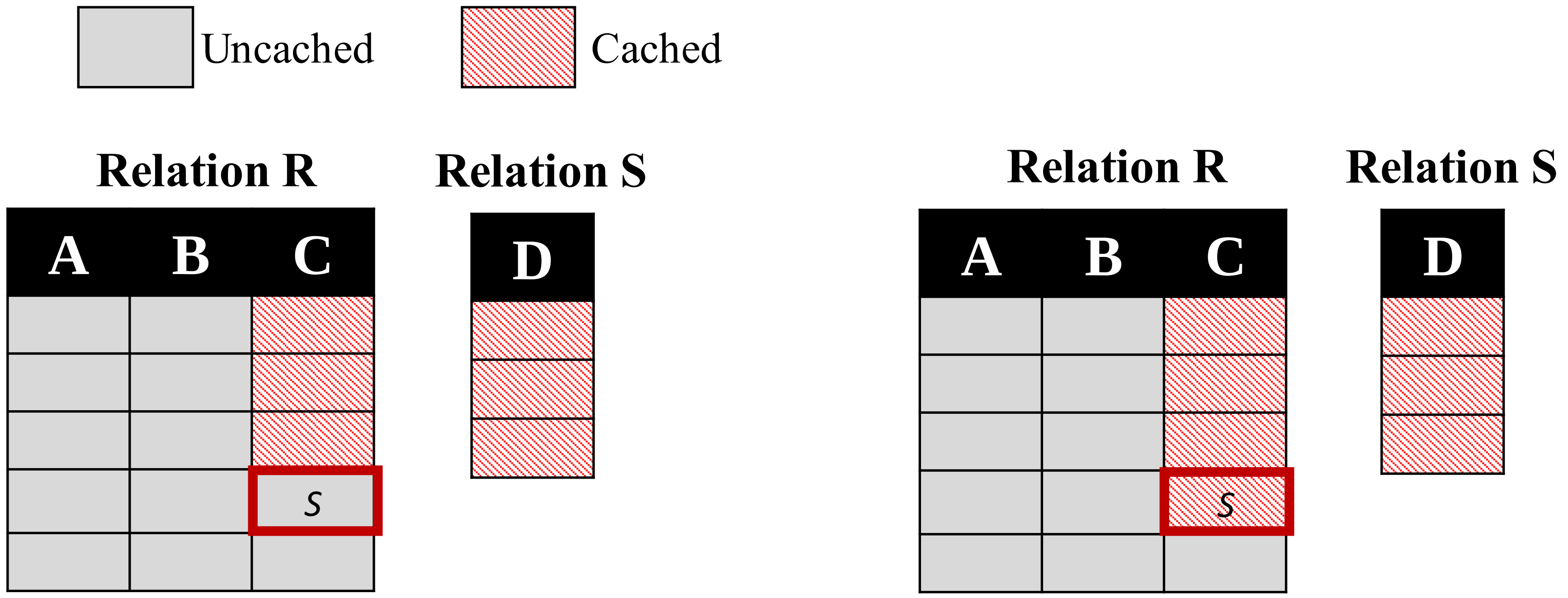
$$\boxed{RT_{uncached} = estRuntime(cached\ segments/S)}$$

[1] Anil Shanbhag, Samuel Madden, Xiangyao Yu, *A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics*, **SIGMOD 2020**

Semantic-Aware Fine-Grained Caching

Extend conventional LFU with **weighted frequency counters.**

Weight reflects the potential **benefits of caching the segments** and is derived using cost model[1].



$$RT_{uncached} = estRuntime(cached\ segments / S)$$

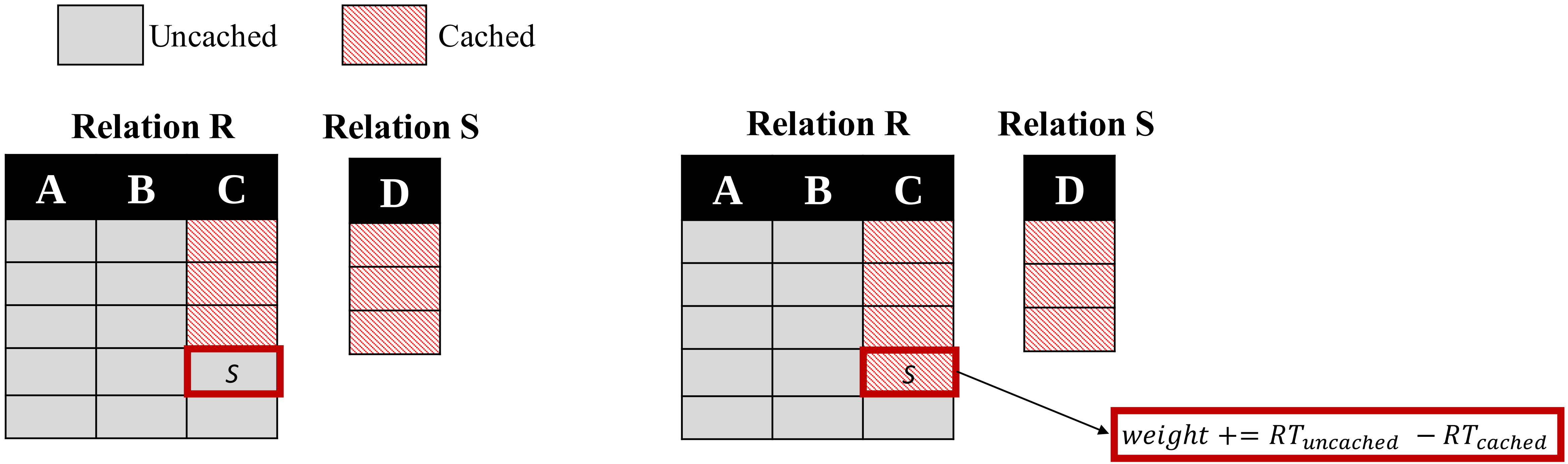
$$RT_{cached} = estRuntime(cached\ segments \cup S)$$

[1] Anil Shanbhag, Samuel Madden, Xiangyao Yu, *A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics*, **SIGMOD 2020**

Semantic-Aware Fine-Grained Caching

Extend conventional LFU with **weighted frequency counters.**

Weight reflects the potential **benefits of caching the segments** and is derived using cost model[1].



$$RT_{uncached} = estRuntime(cached\ segments / S)$$

$$RT_{cached} = estRuntime(cached\ segments \cup S)$$

[1] Anil Shanbhag, Samuel Madden, Xiangyao Yu, *A Study of the Fundamental Performance Characteristics of GPUs and CPUs for Database Analytics*, **SIGMOD 2020**

Challenges in Heterogeneous CPU-GPU Model

We aim to answer the **existing challenges** in heterogeneous CPU-GPU DBMS:

1. **Data Placement**

   → How do we partition data between CPU and GPU?

2. **Heterogeneous Query Execution**

   → How to coordinate query execution between CPU and GPU?

Heterogeneous Query Execution

Challenges:

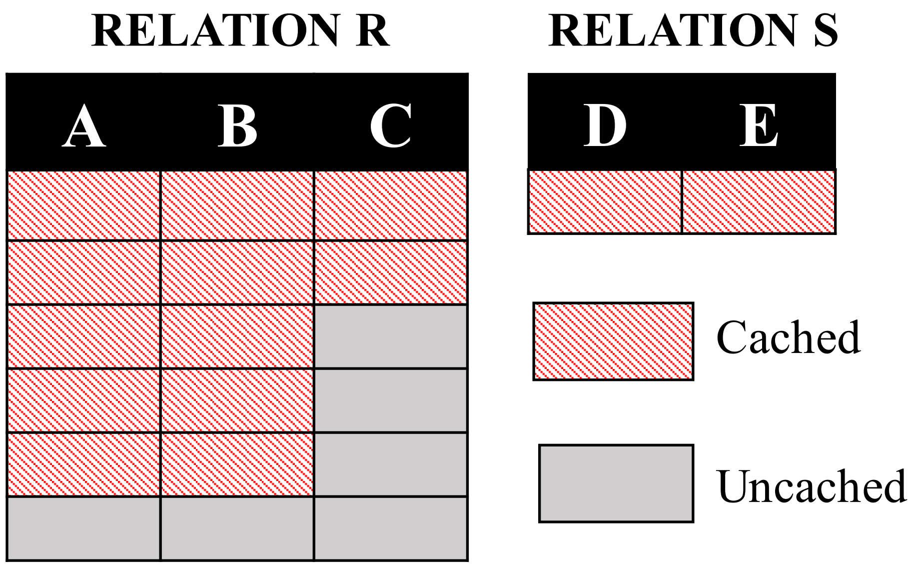
1. Exploit intra-device and inter-device **parallelism** in both CPU and GPU.
2. Minimize inter-device **data transfer**.

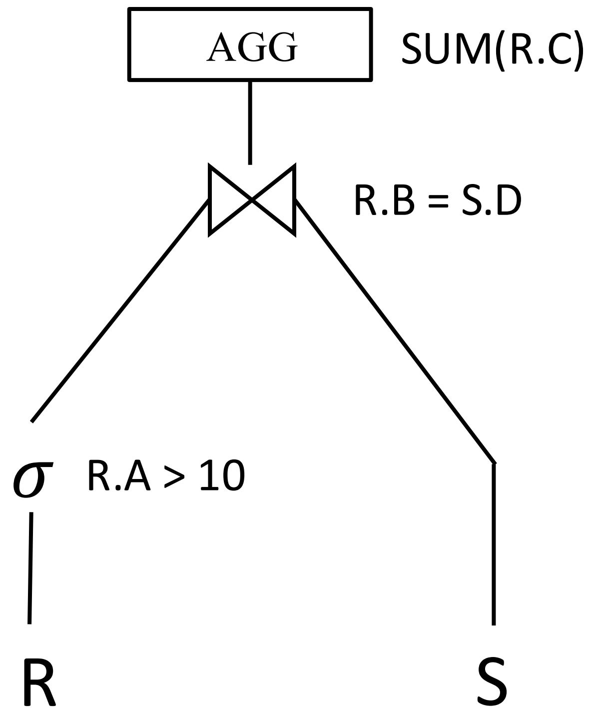
Heterogeneous Query Execution

Challenges:

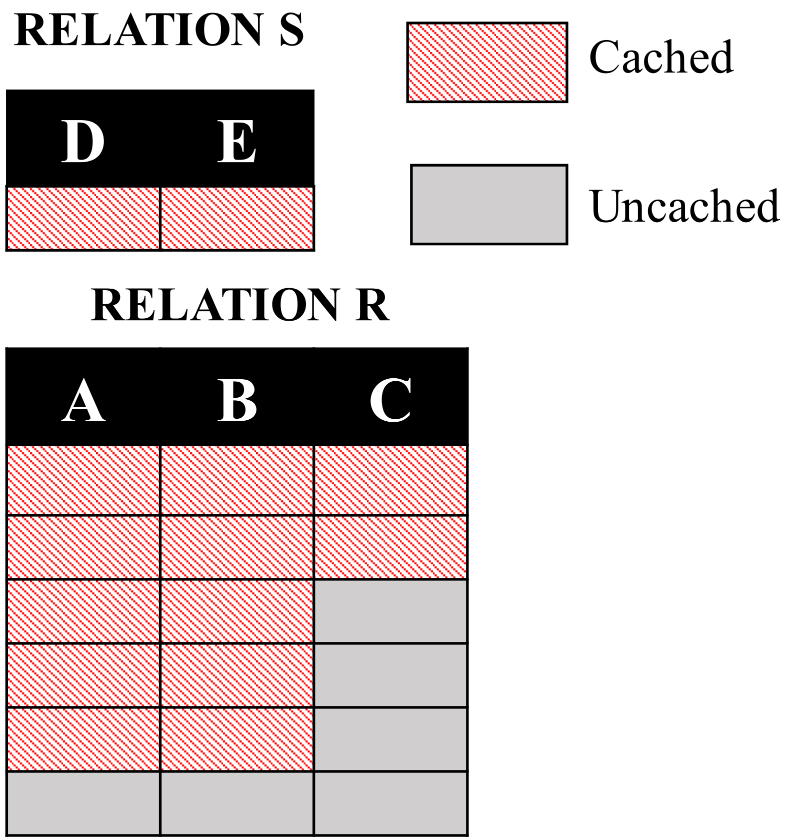
1. Exploit intra-device and inter-device **parallelism** in both CPU and GPU.
2. Minimize inter-device **data transfer**.

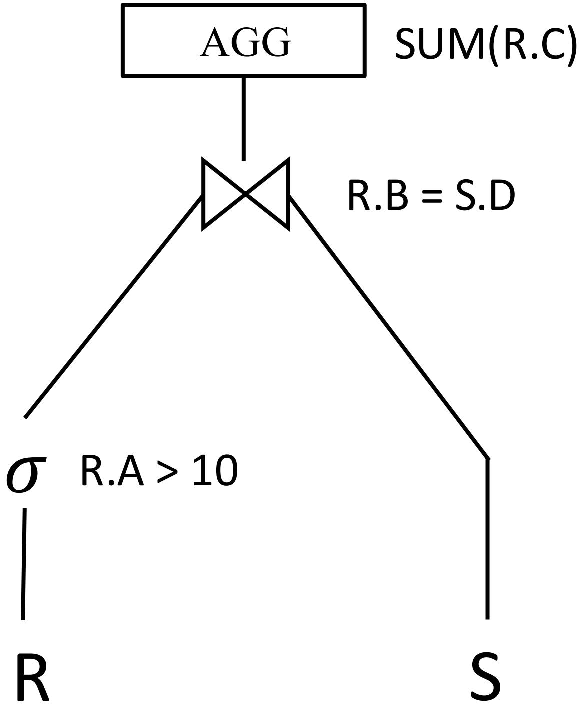
Solution: Introduce segment-level query execution





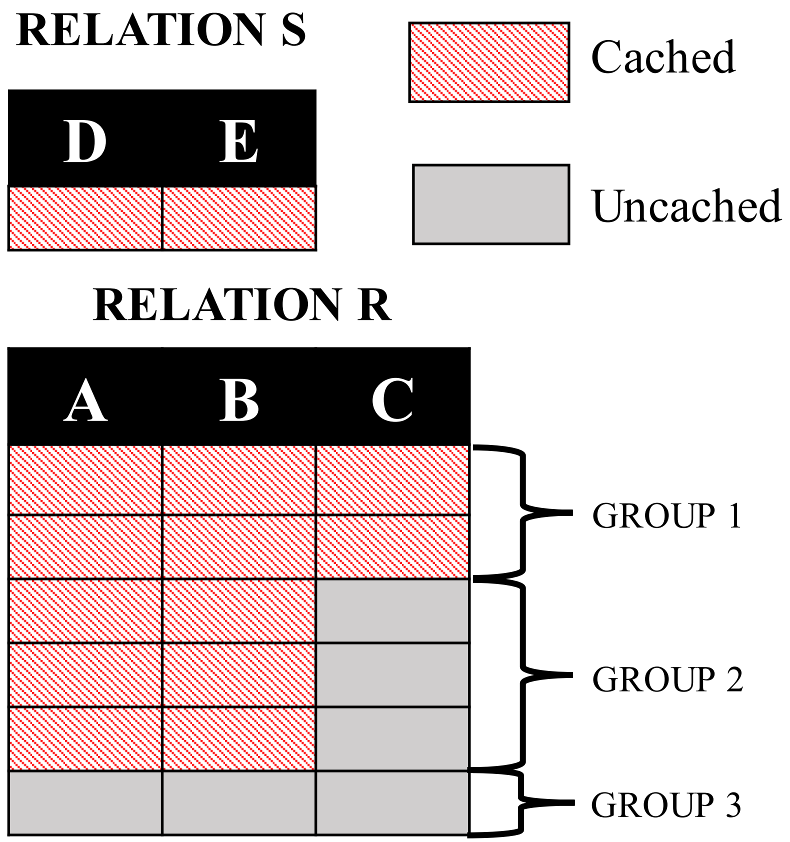
Segment-level Query Execution

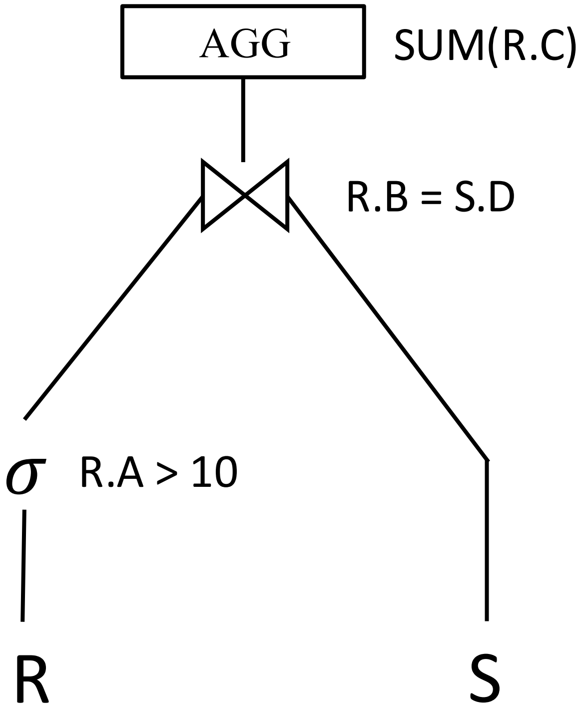




Segment-level Query Execution

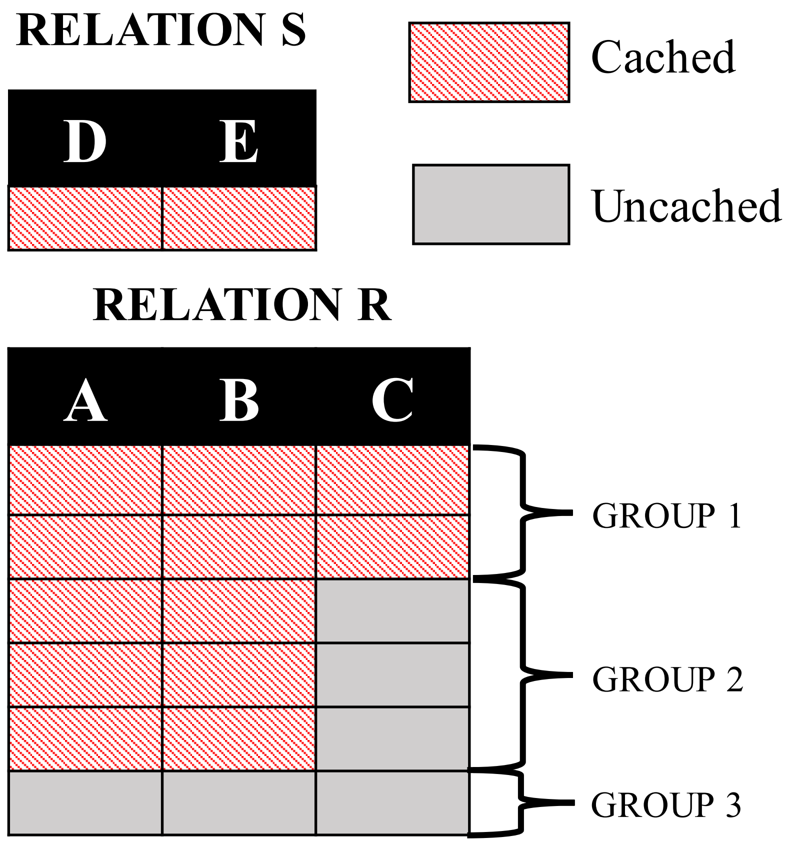
Group segments with the same execution plan into **segment groups.**

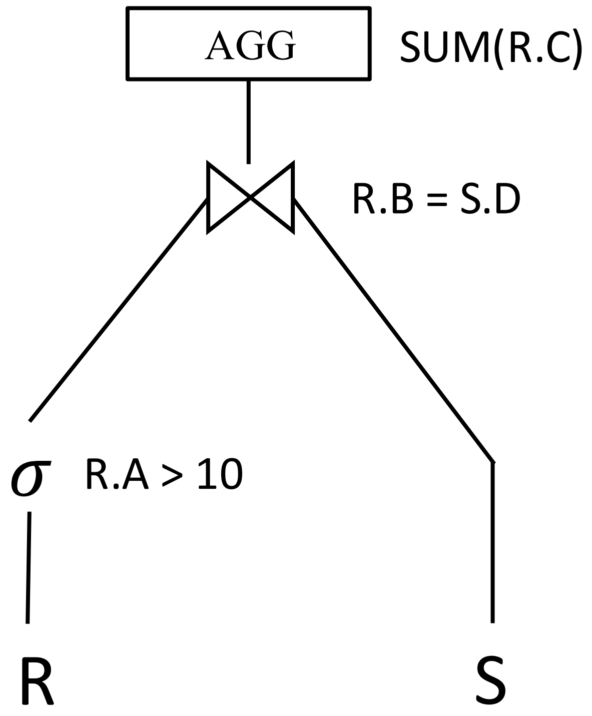




Segment-level Query Execution

Group segments with the same execution plan into **segment groups.**
Execute segment groups **in parallel** following ***data-driven operator placement***[2] → execute operators in GPU only if all input segments reside in GPU.

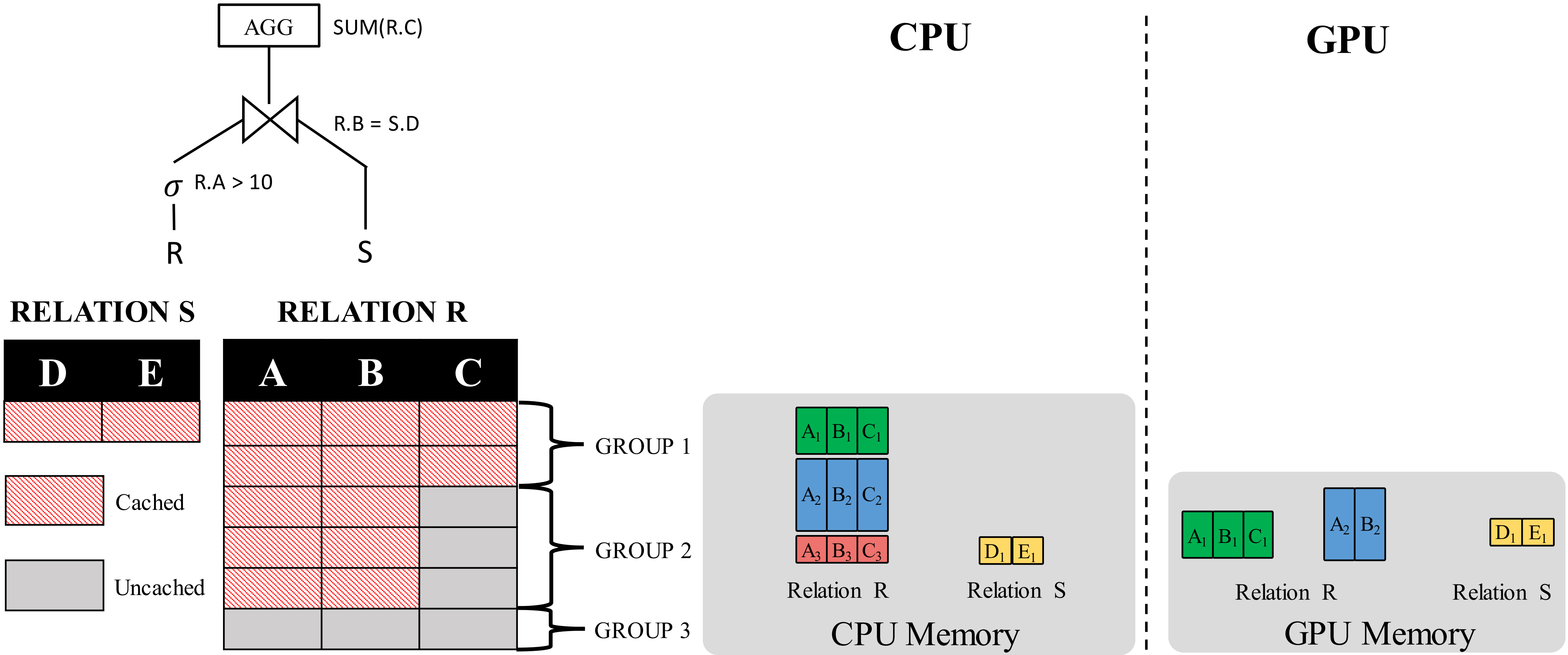




[2] Sebastian Breß, Henning Funke, and Jens Teubner. *Robust Query Processing in Co-Processor-Accelerated Databases*, **SIGMOD 2016.**

Segment-level Query Execution

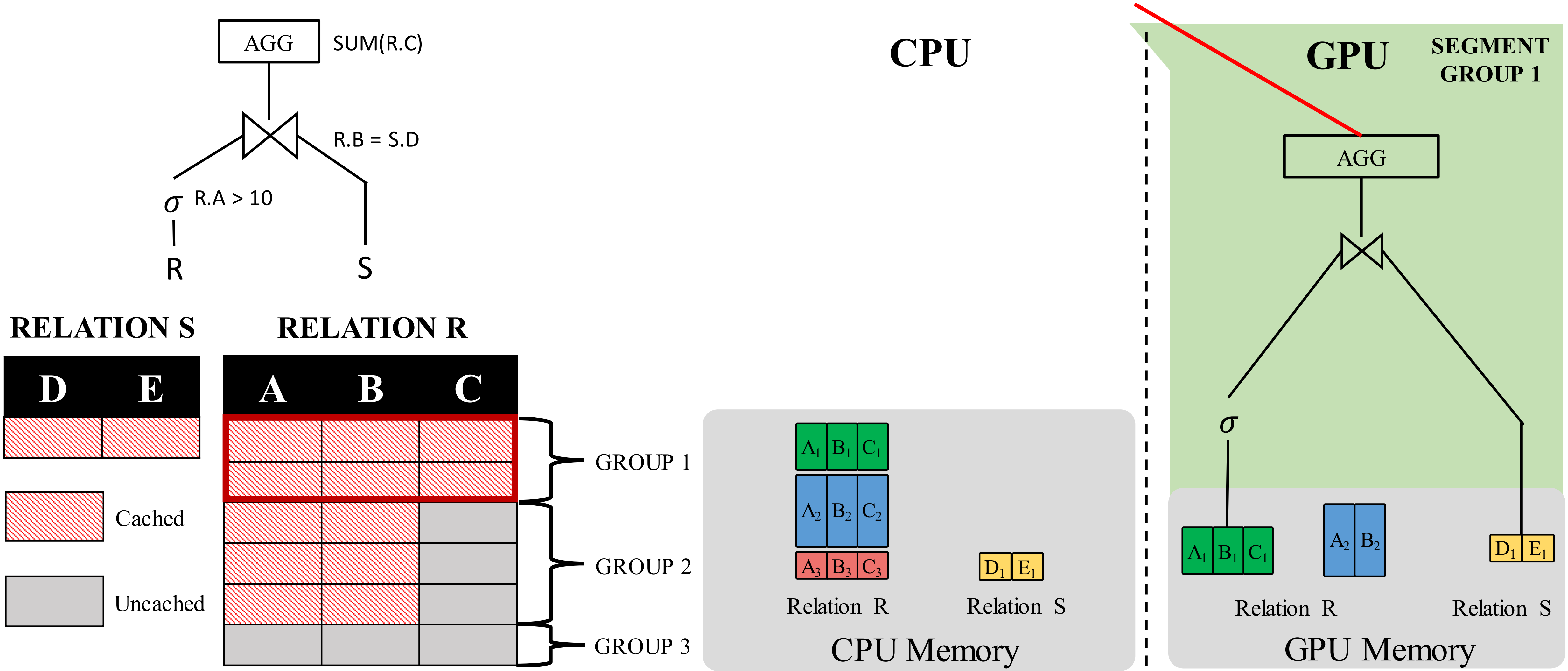
Group segments with the same execution plan into **segment groups.**
Execute segment groups **in parallel** following ***data-driven operator placement***[2] → execute operators in GPU only if all input segments reside in GPU.



[2] Sebastian Breß, Henning Funke, and Jens Teubner. *Robust Query Processing in Co-Processor-Accelerated Databases*, **SIGMOD 2016.**

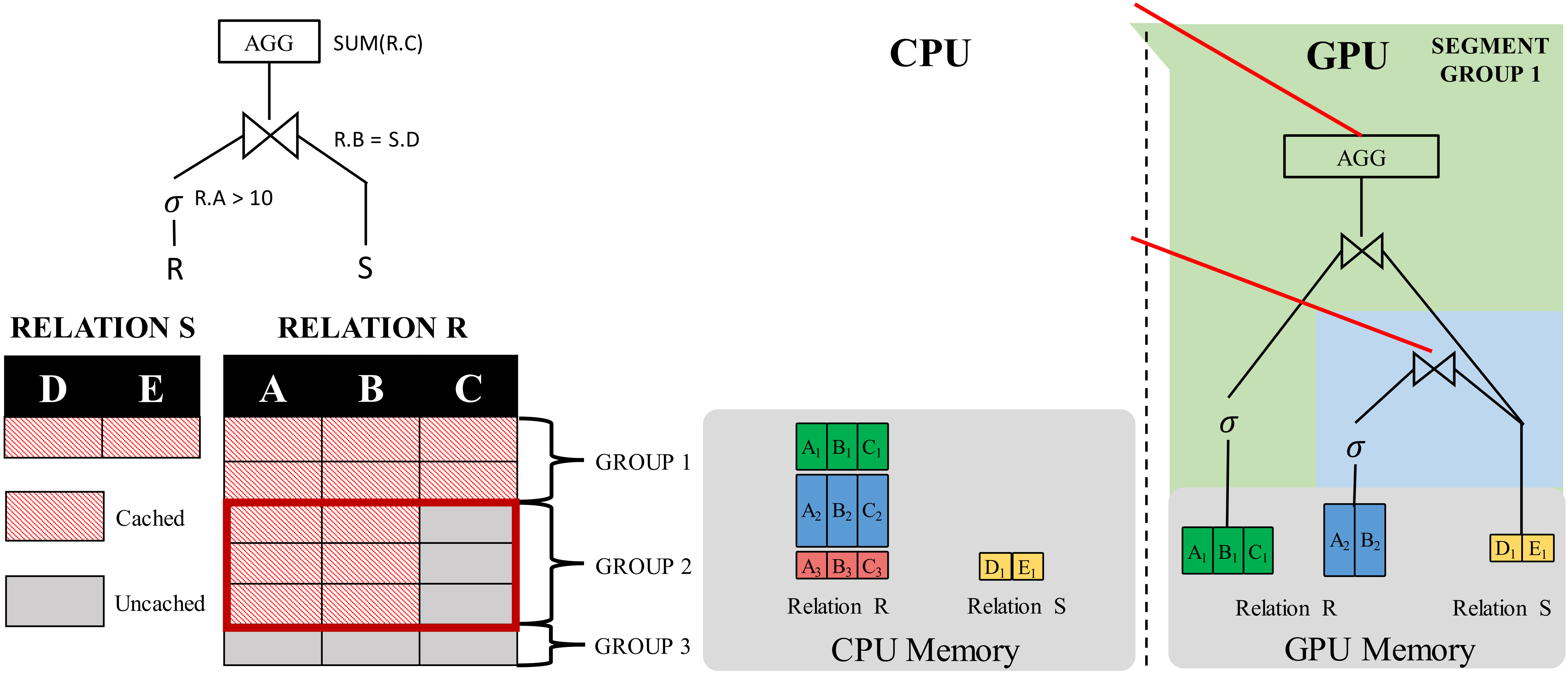
Segment-level Query Execution

Group segments with the same execution plan into **segment groups.**
Execute segment groups **in parallel** following ***data-driven operator placement.***



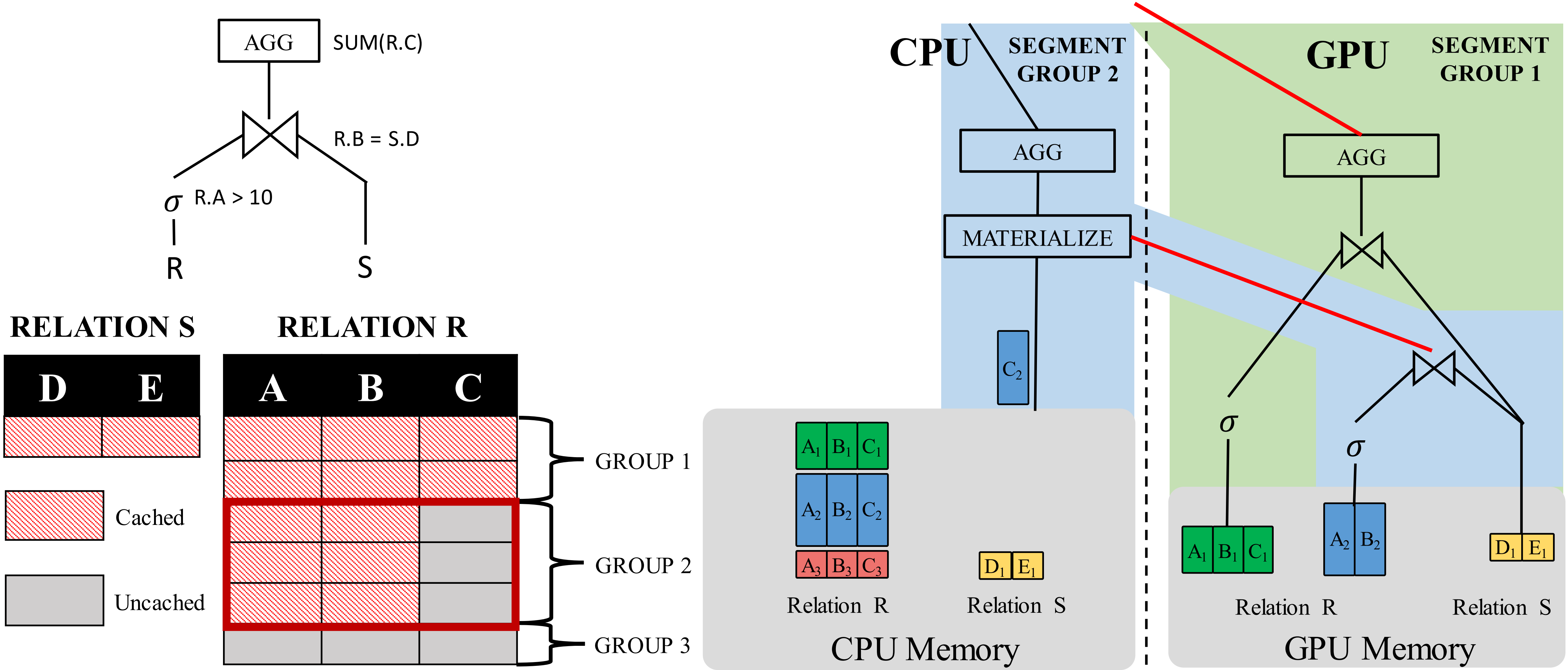
Segment-level Query Execution

Group segments with the same execution plan into **segment groups.**
Execute segment groups **in parallel** following ***data-driven operator placement.***



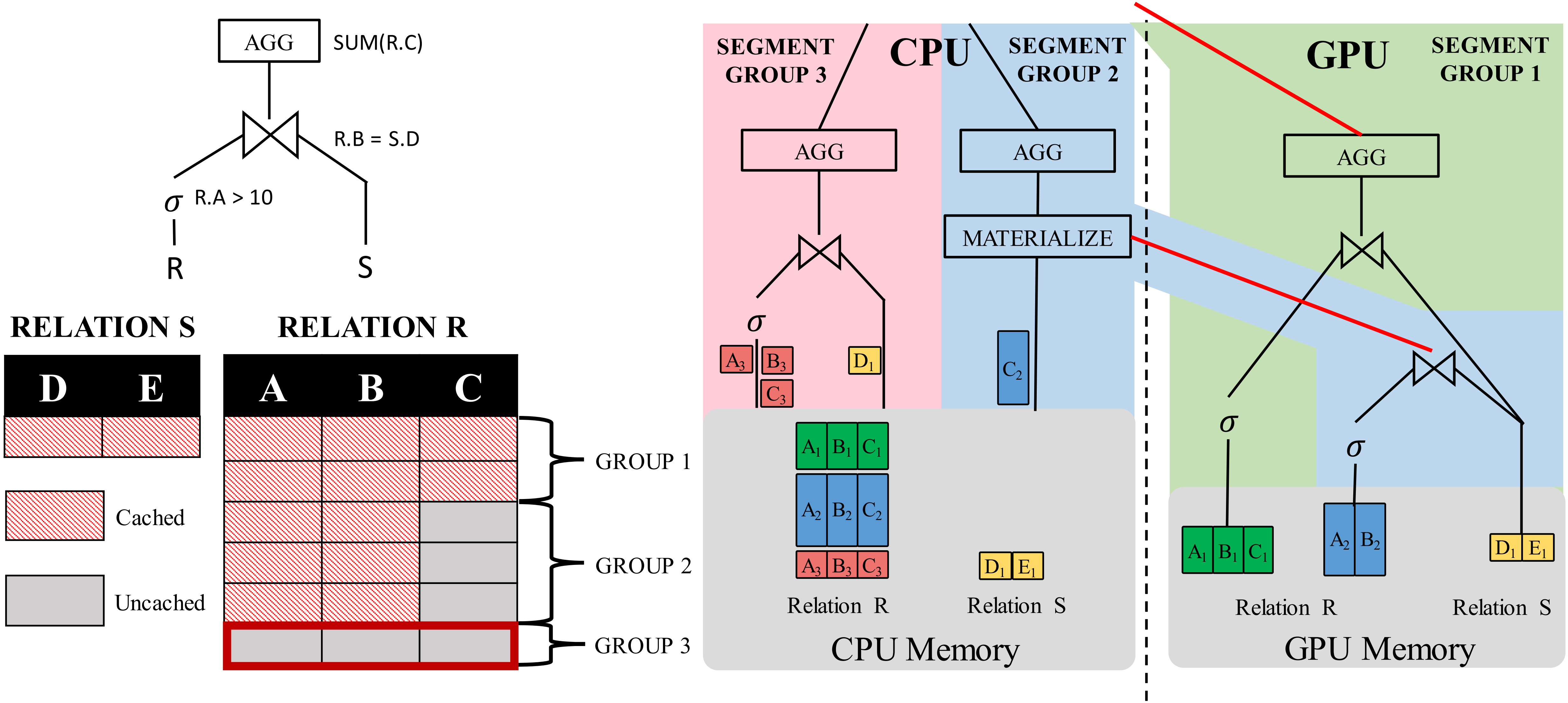
Segment-level Query Execution

Group segments with the same execution plan into **segment groups.**
Execute segment groups **in parallel** following ***data-driven operator placement.***



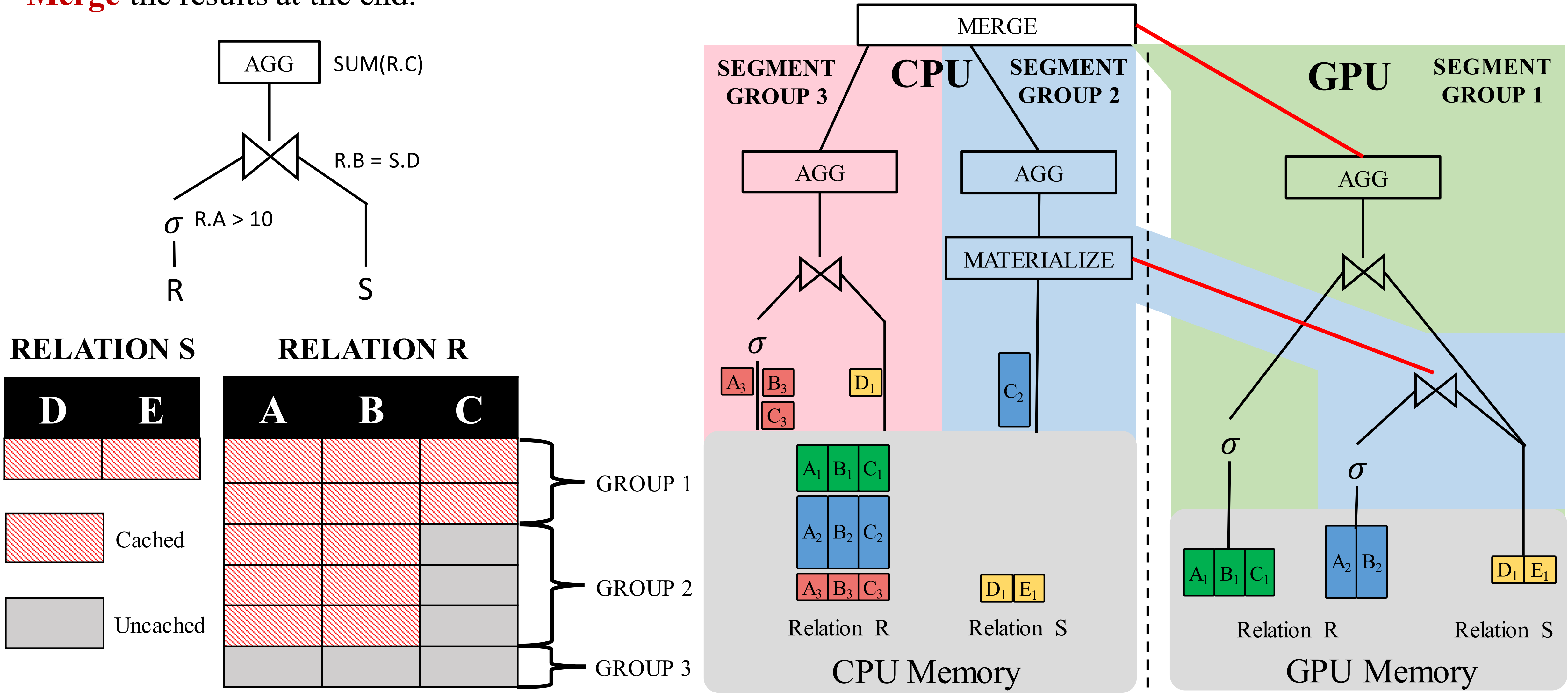
Segment-level Query Execution

Group segments with the same execution plan into **segment groups.**
Execute segment groups **in parallel** following ***data-driven operator placement.***



Segment-level Query Execution

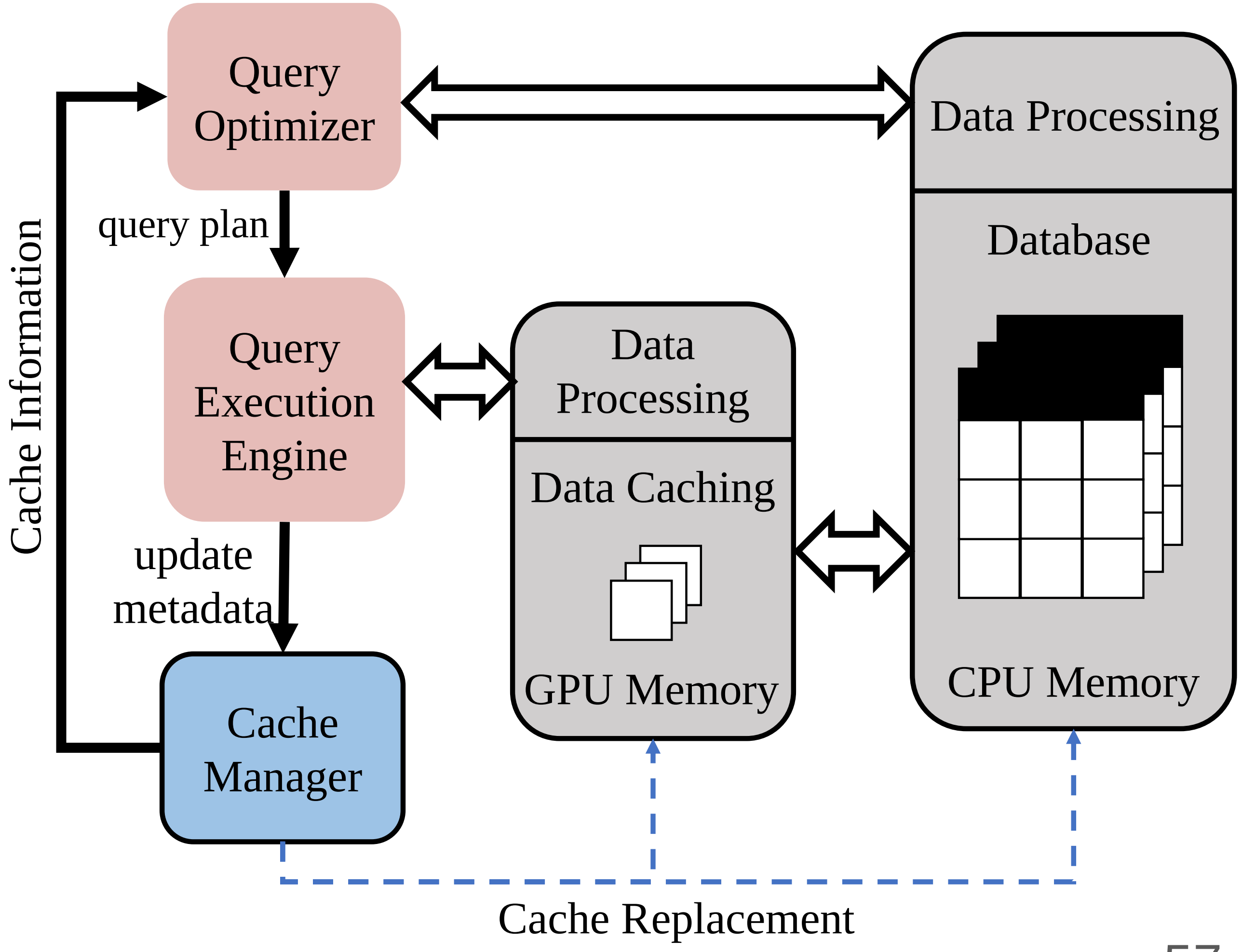
Group segments with the same execution plan into **segment groups.**
Execute segment groups **in parallel** following ***data-driven operator placement.***
Merge the results at the end.



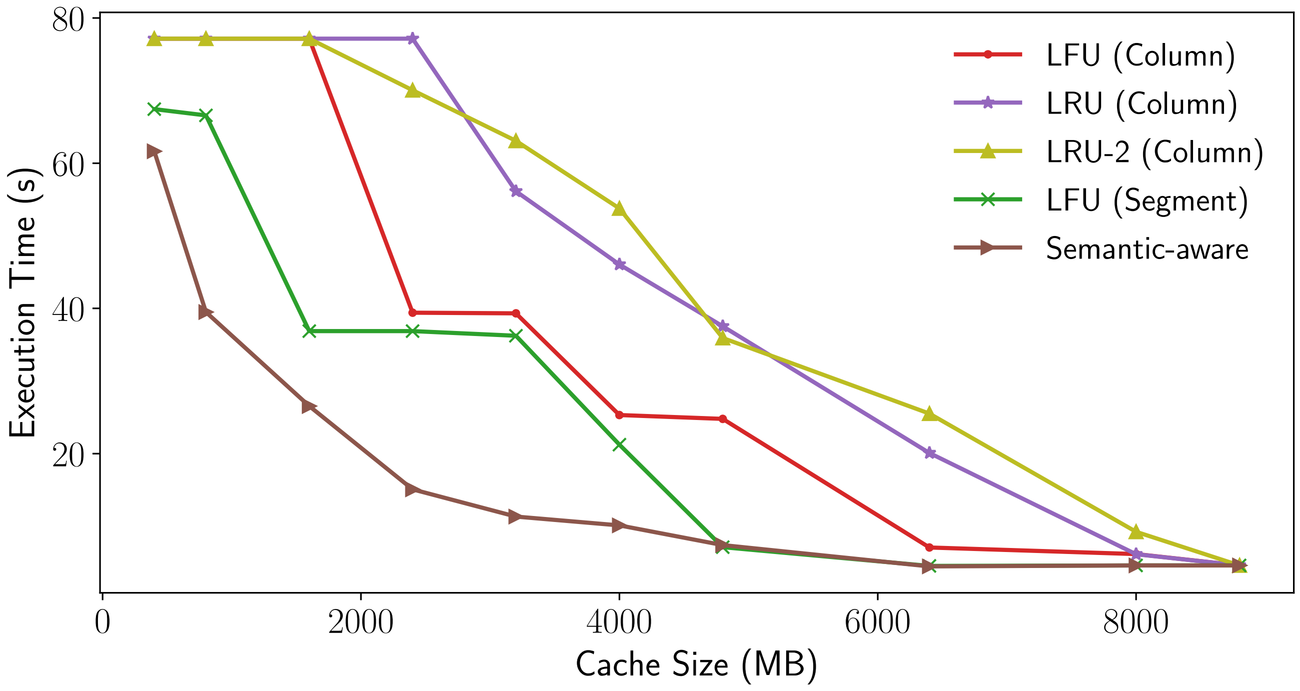
Mordred: A Hybrid CPU-GPU DBMS

Three components:

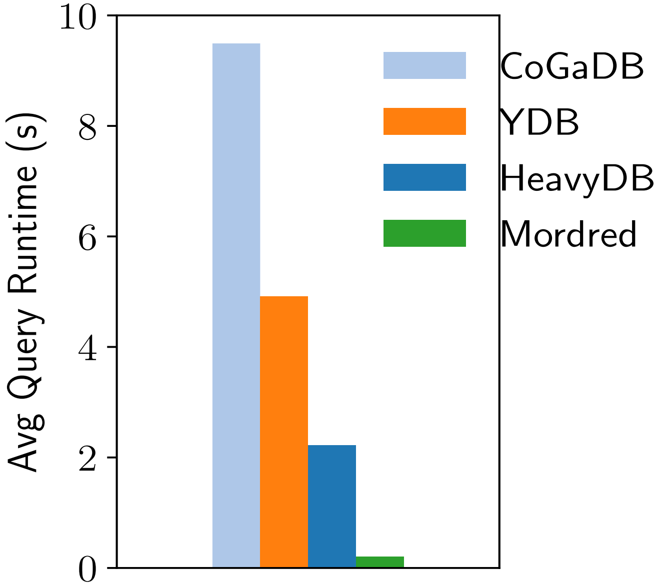
- **Cache Manager**
  - Semantic-Aware Caching Policy
- **Query Optimizer**
  - Data-driven Operator Placement
- **Query Execution Engine**
  - Segment-level Query Execution
  - Tile-based Execution Model
  - Late Materialization
  - Segment Skipping



Hybrid CPU-GPU DBMS – Evaluation



(a) Comparison of Different Data Placement Policies



(b) End-to-end Performance

Semantic-aware caching outperforms the best prior policy **by up to 3x.**

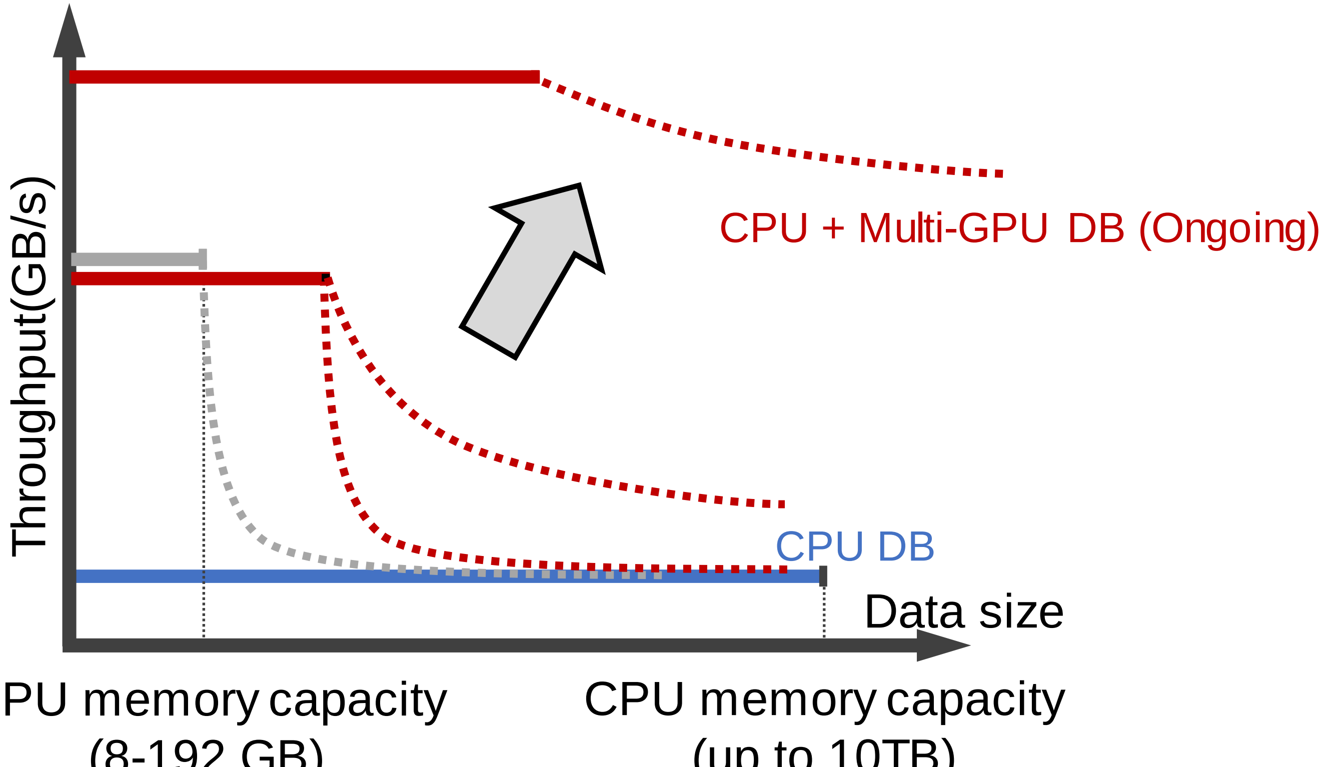
Mordred is 11x faster than the best existing GPU DBMS.

Hardware: NVIDIA V100 GPU. Intel Xeon Platinum CPU (24 Cores).
Benchmark: Star Schema Benchmark: (1) SF = 40 → Figure a, (2) SF = 160 → Figure b.

GPU Database Optimizations

Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- ➢Tile-based execution (SIGMOD 2020[1])
- ➢Data Compression (SIGMOD 2022[2])
- ➢Heterogeneous CPU-GPU DBMS (VLDB 2022[3])
- ➢**Multi-GPU DBMS (ongoing)**



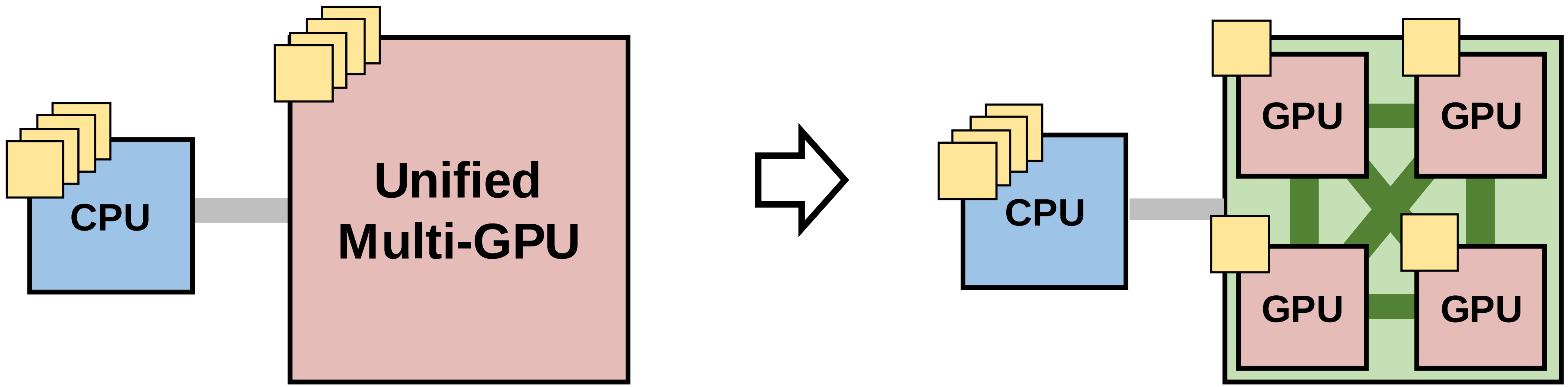
Heterogeneous CPU + Multi-GPU DBMS

Idea 1: Unified Multi-GPU Abstraction

- ➢Views multi-GPU as a single large monolithic GPU.

Idea 2: Capacity-aware Replication Policy

- ➢Intelligent replication policy to reduce data transfer between GPUs.



(a) Data Caching **(b) Data Partitioning/Replication**

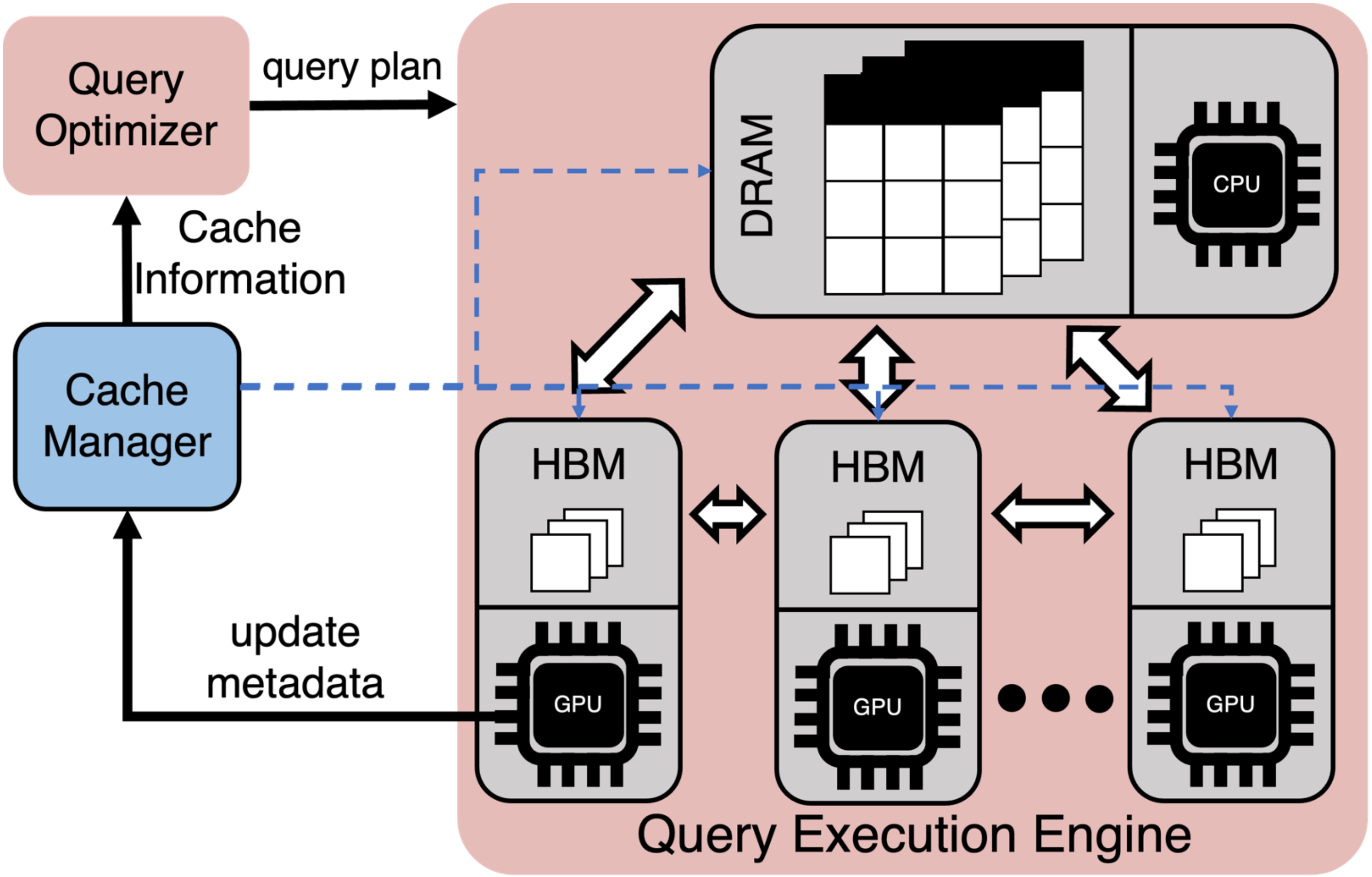
Lancelot: A Hybrid CPU + Multi-GPU DBMS

Three components:

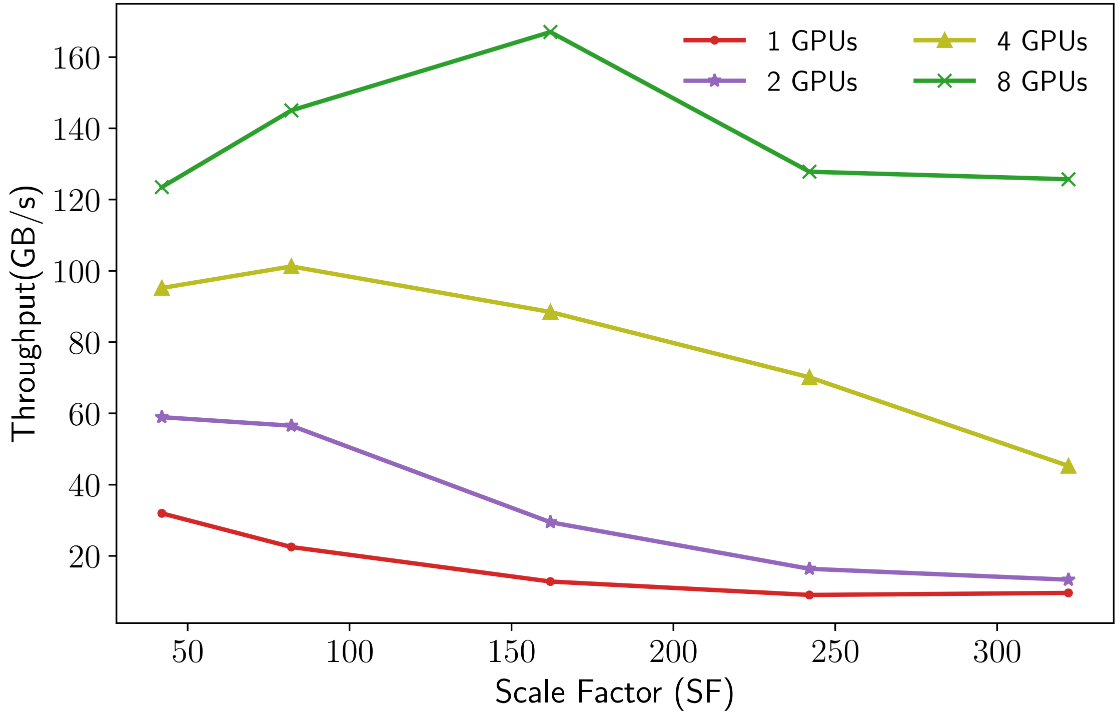
➢ **Cache Manager**
- Semantic-Aware Caching Policy
- Capacity-Aware Replication Policy

➢ **Query Optimizer**
- Data-driven Operator Placement

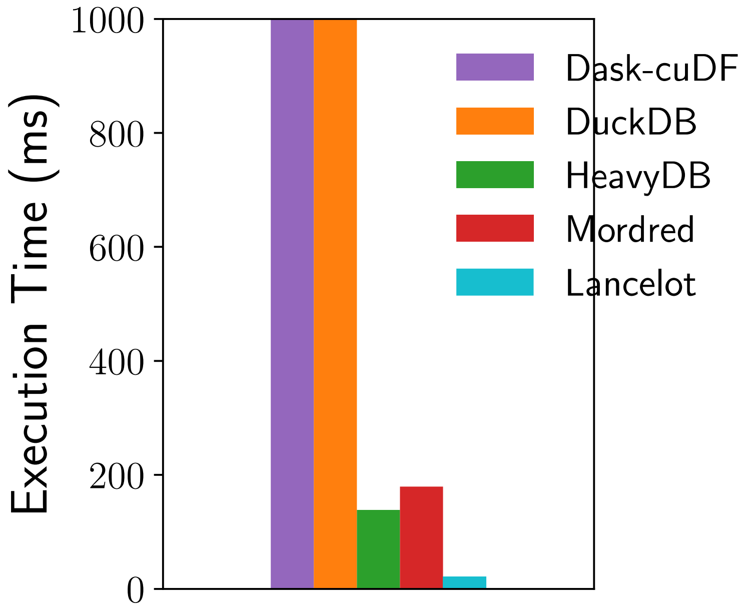
➢ **Query Execution Engine**
- Segment-level Query Execution
- Late Materialization
- Adaptive Query Execution
- Join Reordering
- Tile-based Execution Model
- Segment Skipping



Multi-GPU DBMS – Evaluation



(a) Scaling to Multiple GPUs



(b) End-to-end Performance

Lancelot can scale Mordred to Multiple GPUs.

Lancelot is 7x faster than the best existing multi-GPU DBMS.

Hardware: NVIDIA V100 GPU. Intel Xeon Platinum CPU (104 Cores).
Benchmark: Star Schema Benchmark: SF = 160 (Figure b)

GPU Database Optimizations

Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- Crystal: tile-based execution **(SIGMOD 2020[1])**
- Data Compression **(SIGMOD 2022[2])**
- Heterogeneous CPU-GPU DBMS **(VLDB 2022[3])**
- Multi-GPU DBMS (ongoing)

Thrust 2: Enhancing the Practicality of GPU Databases

- Accelerating UDF on GPUs **(DaMoN@SIGMOD 2023[4])**
- Code Generation for GPU DBMS (ongoing)

GPU Database Optimizations

Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- Crystal: tile-based execution **(SIGMOD 2020[1])**
- Data Compression **(SIGMOD 2022[2])**
- Heterogeneous CPU-GPU DBMS **(VLDB 2022[3])**
- Multi-GPU DBMS (ongoing)

Thrust 2: Enhancing the Practicality of GPU Databases

- Accelerating UDF on GPUs **(DaMoN@SIGMOD 2023[4])**
- Code Generation for GPU DBMS (ongoing)

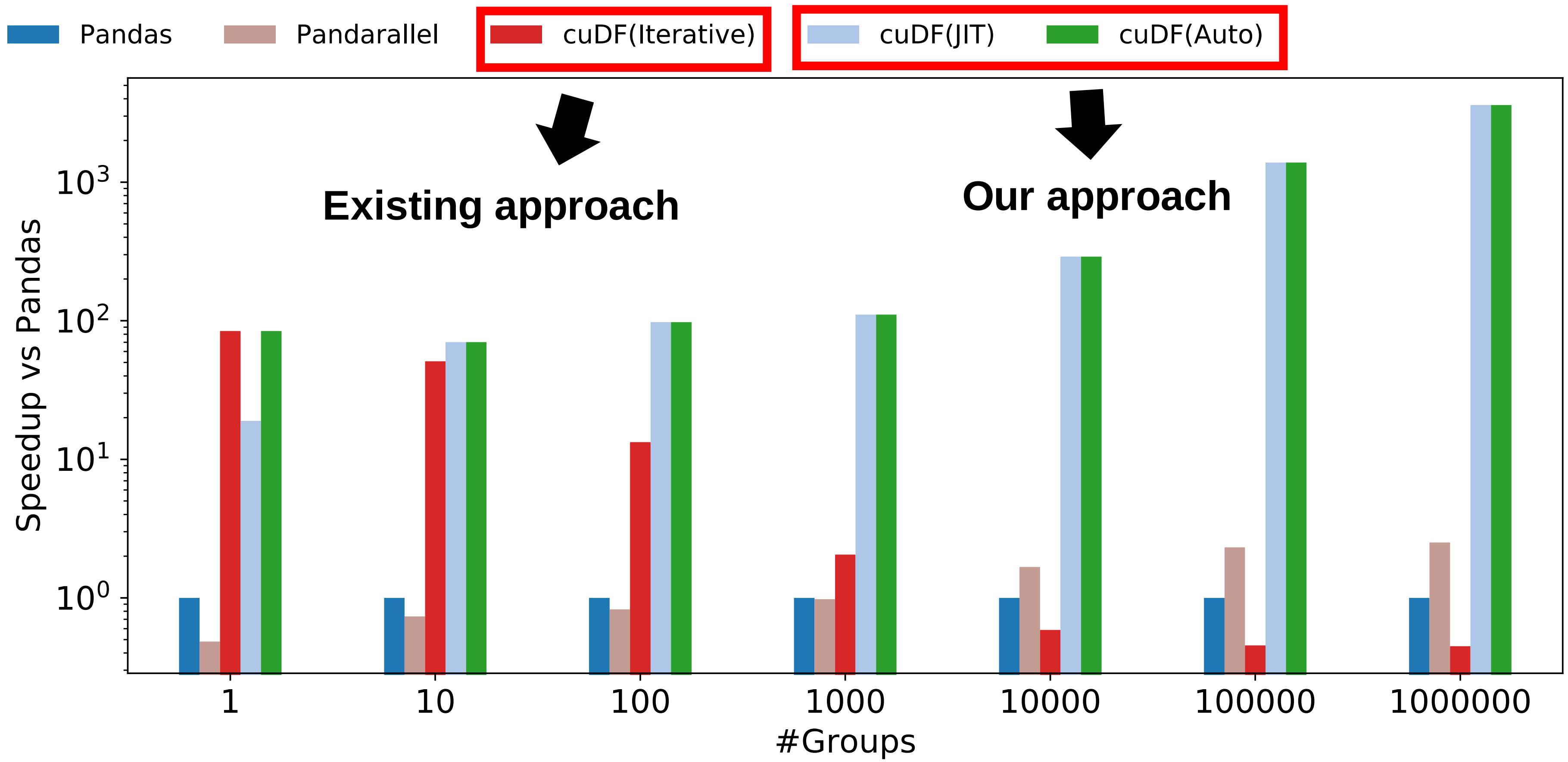
Accelerating UDAF on GPUs

Challenge: UDAF Execution is slow on GPUs (sometimes even slower than CPU).

Accelerating UDAF on GPUs

Challenge: UDAF Execution is slow on GPUs (sometimes even slower than CPU).

We **introduce a novel UDAF execution framework** with **Tile-based Execution and JIT Compilation.**



We are up to **8000×** faster against existing approach (on NVIDIA V100 GPU).

Fully integrated and released in **NVIDIA RAPIDS cuDF v23.02**[2].

Confronting Challenges in GPU DBMS

Goal: Solve these challenges with **two research thrusts**

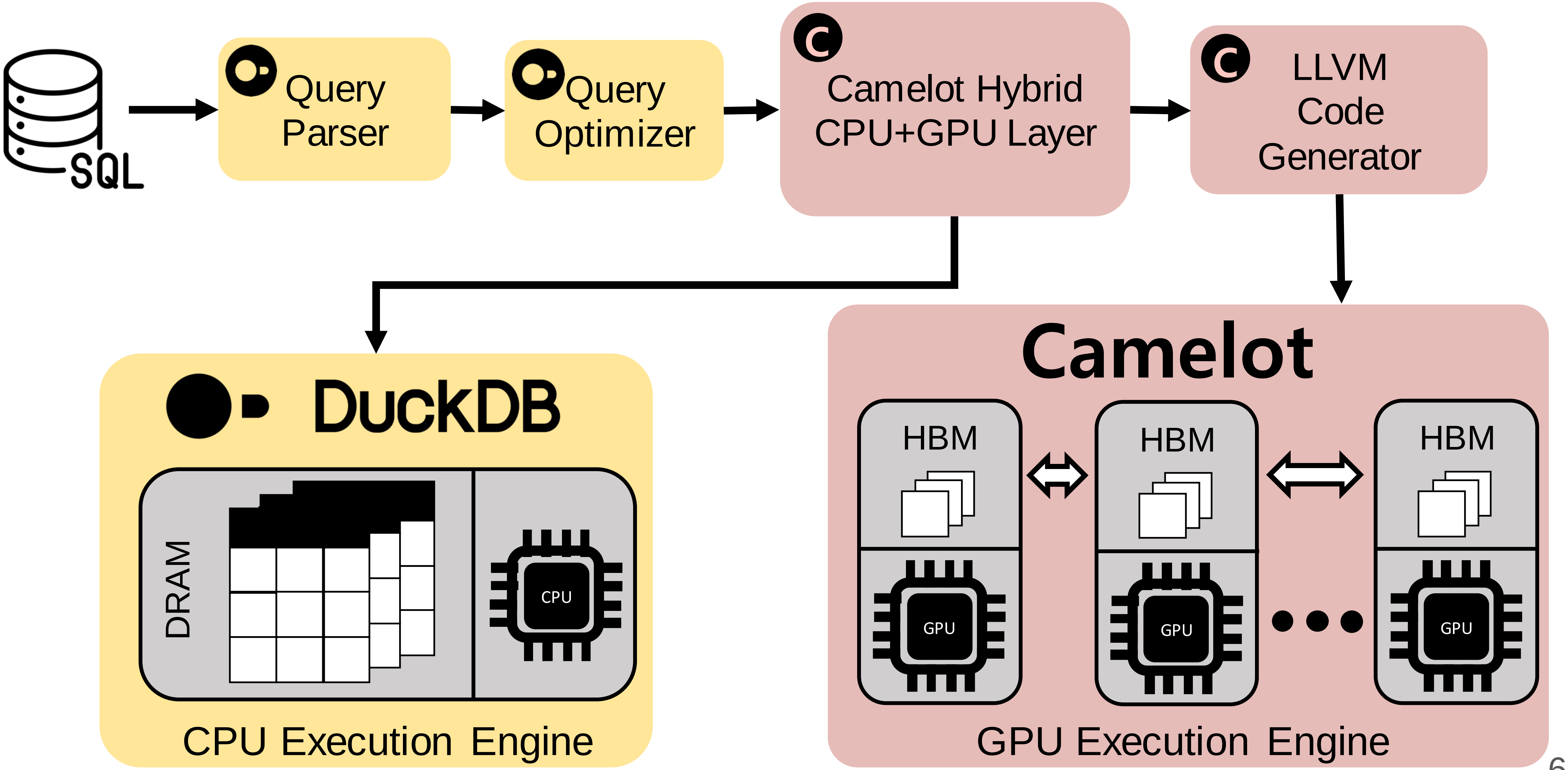
Thrust 1: Enabling **Large-Scale** Data Analytics on GPUs

- Data Compression (SIGMOD 2022[1])
- Heterogeneous CPU-GPU DBMS (VLDB 2022[2])
- Multi-GPU DBMS (ongoing)

Thrust 2: Enhancing the Practicality of GPU Databases

- **Accelerating UDF on GPUs (DaMoN@SIGMOD 2023[3])**
- **Code Generation for GPU DBMS (ongoing)**

Camelot v0.1


SQL
Query Parser
Query Optimizer
Camelot Hybrid CPU+GPU Layer
LLVM Code Generator
DuckDB
DRAM
CPU
CPU Execution Engine
Camelot
HBM
HBM
HBM
GPU
GPU
GPU
GPU Execution Engine

Conclusion

GPU is becoming the new modality of SQL analytics