# Accelerating Foreign-Key Joins using Asymmetric Memory Channels

Holger Pirk
holger@cwi.nl

Stefan Manegold
manegold@cwi.nl

Martin Kersten
mk@cwi.nl

CWI

monetdb

# Why?

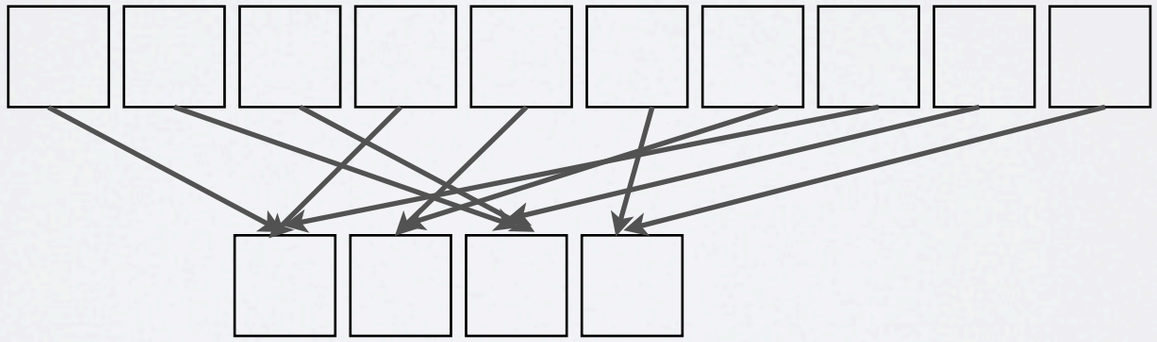
- Trivia: Joins are important
- But: Many Joins are (Indexed) Foreign Key Joins
- Important example: Data Warehouse Star Joins
  - Large Fact Table (many Gigabytes) with Foreign Keys on
  - Small Dimension Tables (hundreds of Megabytes)
  - Joins are Performed for Projection or Condition Evaluation

# A History of In Memory Join Processing (and incidentally of this talk)

- Foreign-Key Joins on CPUs
  - Naïve (Indexed Hash) Foreign-Key Joins
  - Clustered Foreign-Key Joins
- Foreign-Key Joins on GPGPUs
  - Clustered Foreign-Key Joins
  - Naïve (Asymmetric) Foreign-Key Joins

# Premises for this talk

- Focus is on Indexed Foreign Key Joins
- We assume an unclustered Foreign Key Index Column
- i.e., Pointers to Matching Tuples of the Target Table

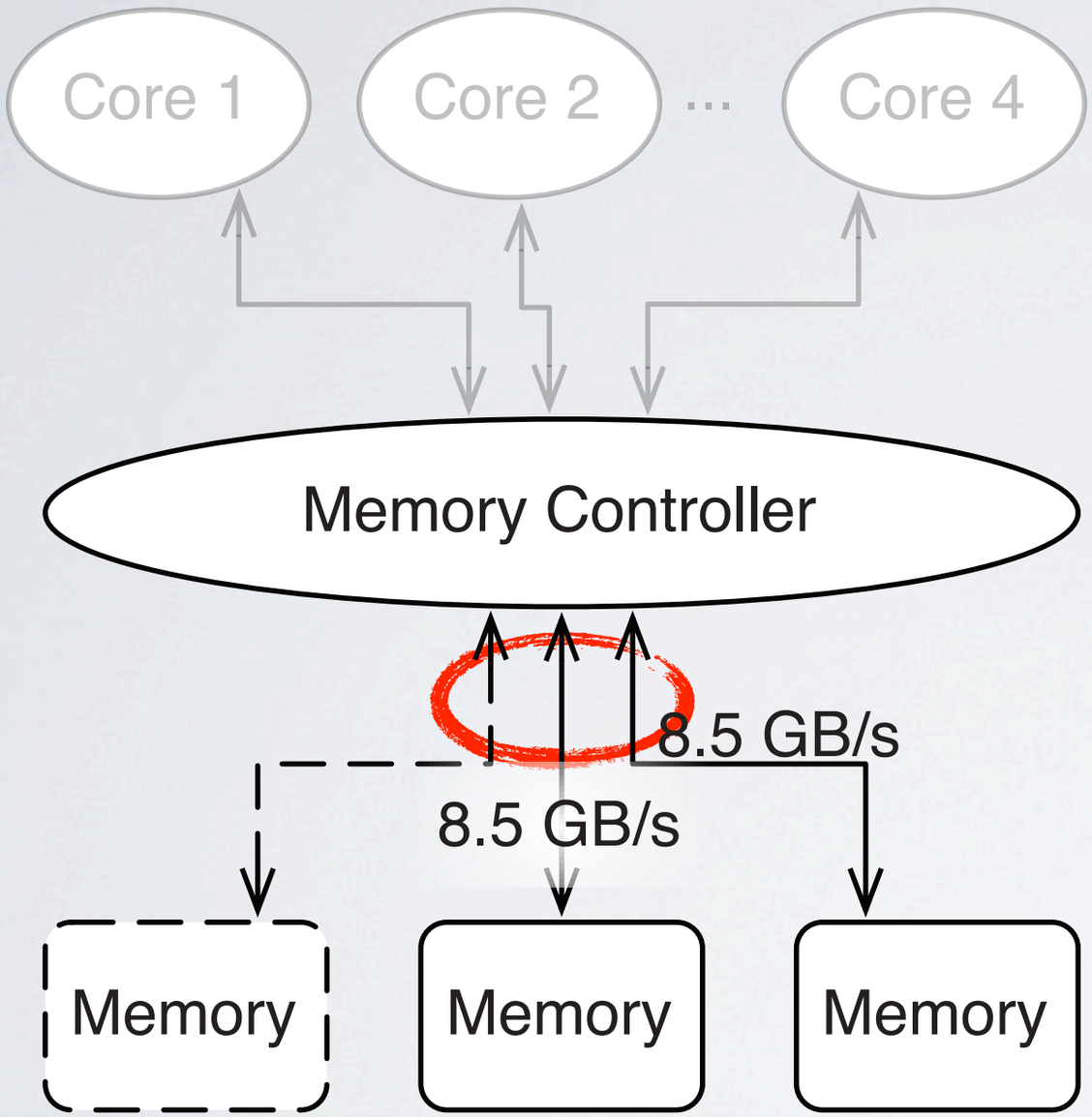


CWI

monetdb

# Foreign-Key Joins on CPUs

# The Problem: Bandwidth Bound CPUs




- Bottleneck for almost all relational (Memory Resident) DBMS operations
- Valuable Target for Optimization

# Naïve Foreign-Key Joins on CPUs

# Naïve Foreign-Key Joins on CPUs

- Foreign-Key Index is scanned sequentially

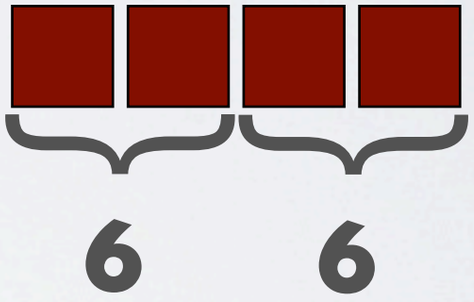
- Target Table is randomly accessed

# Naïve Foreign-Key Joins on CPUs

- Optimization Targets:
  - Foreign-Key Index is scanned sequentially
  - Index Cache lines accessed once, i.e., not much to optimize

# Naïve Foreign-Key Joins on CPUs

- Large Target Tables cause heavy Cache Thrashing
- Assume a (theoretical) 1-Slot Cache with 2-Value Cache Lines
- How many Cache Misses do you get?
    - 80% of are Capacitive Misses




- Target Table Lookups are rewarding optimization Targets

# Clustered Foreign-Key Joins on CPUs

CWI

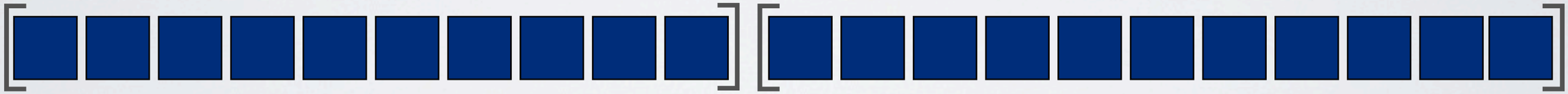
monetdb

# Clustered Foreign-Key Joins on CPUs

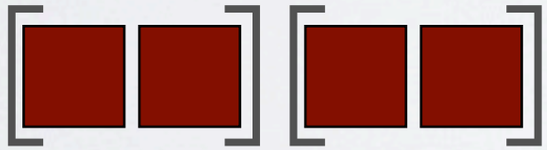
- Require a Value-partitioned Foreign-Key Index
- Foreign-Key Index Partitions are scanned
- Target Table Random Access Locality is improved

# Clustered Foreign-Key Joins on CPUs

- Require a Value-partitioned Foreign-Key Index
- Foreign-Key Index Partitions are scanned



- Target Table Random Access Locality is improved

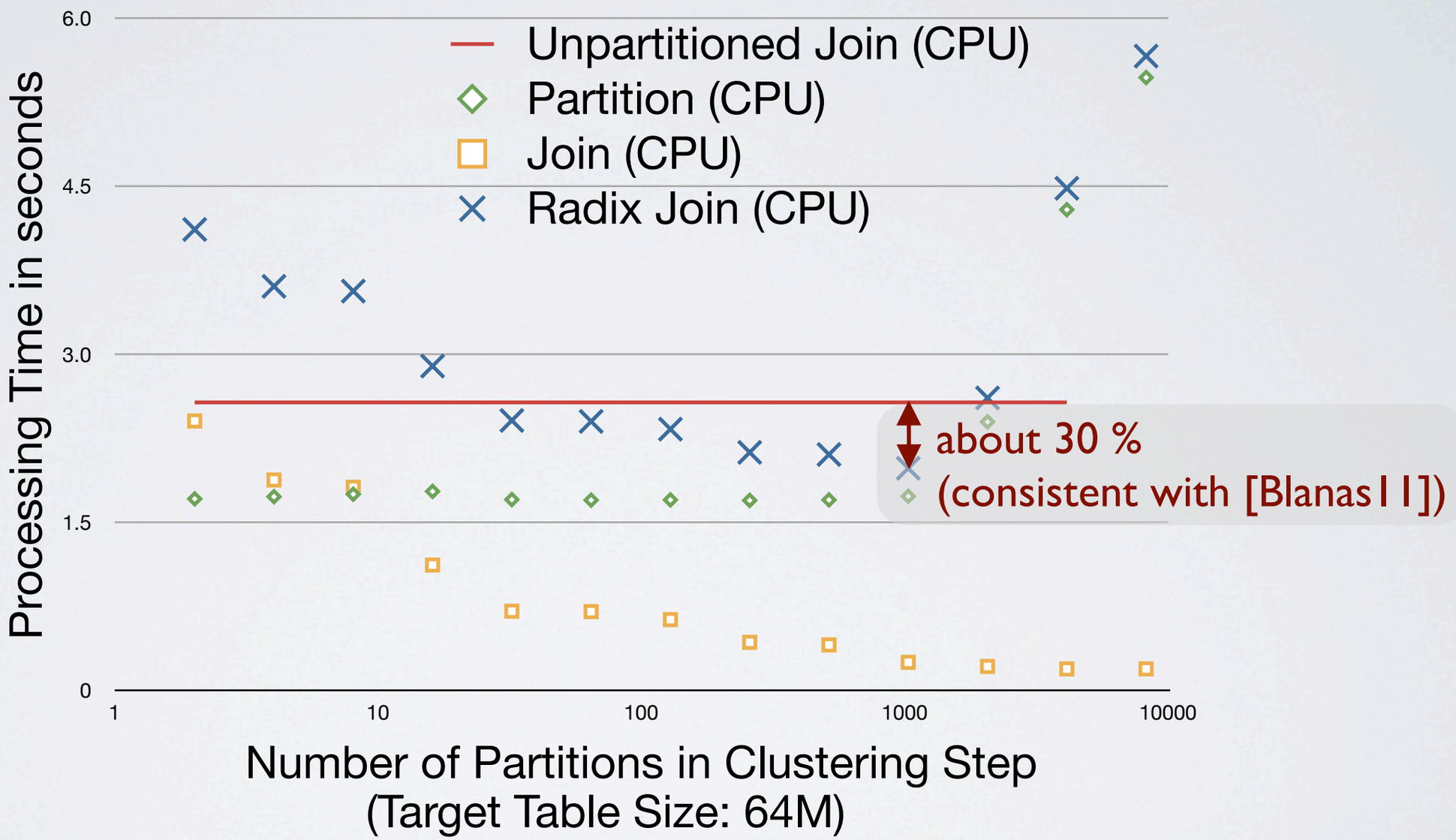


# Clustered Foreign-Key Joins on CPUs

- What is the number of Target Table Cache Misses?
- So Cache behavior is greatly improved

- Also: (disjoint) Partitions can be processed in parallel
  - Parallelism is generally good for GPUs
- But: The Index has to be Value (e.g., Radix)-partitioned first

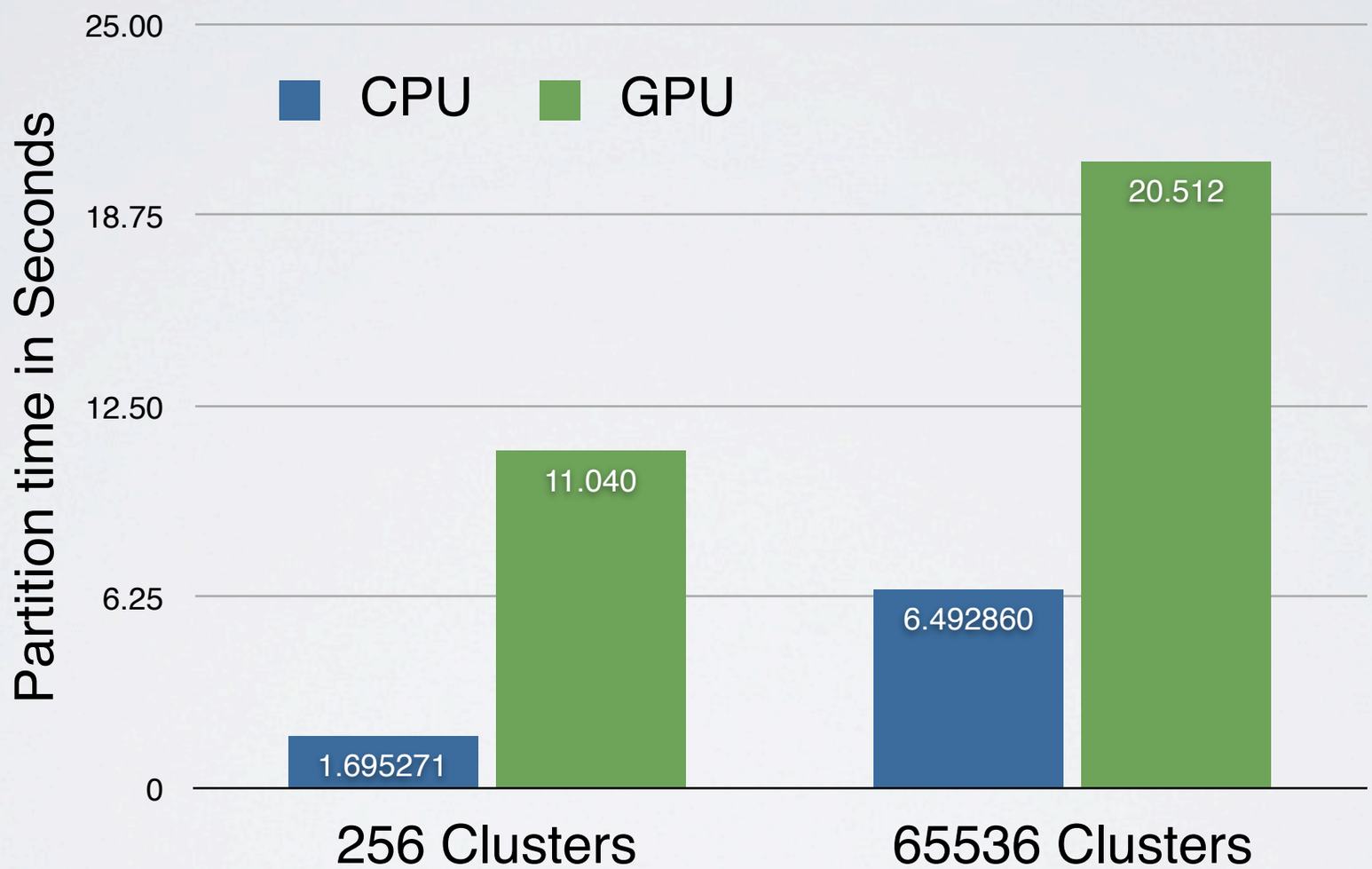
# Clustered Foreign-Key Joins on CPUs




CWI

monetdb

# Clustered Foreign-Key Joins on CPUs

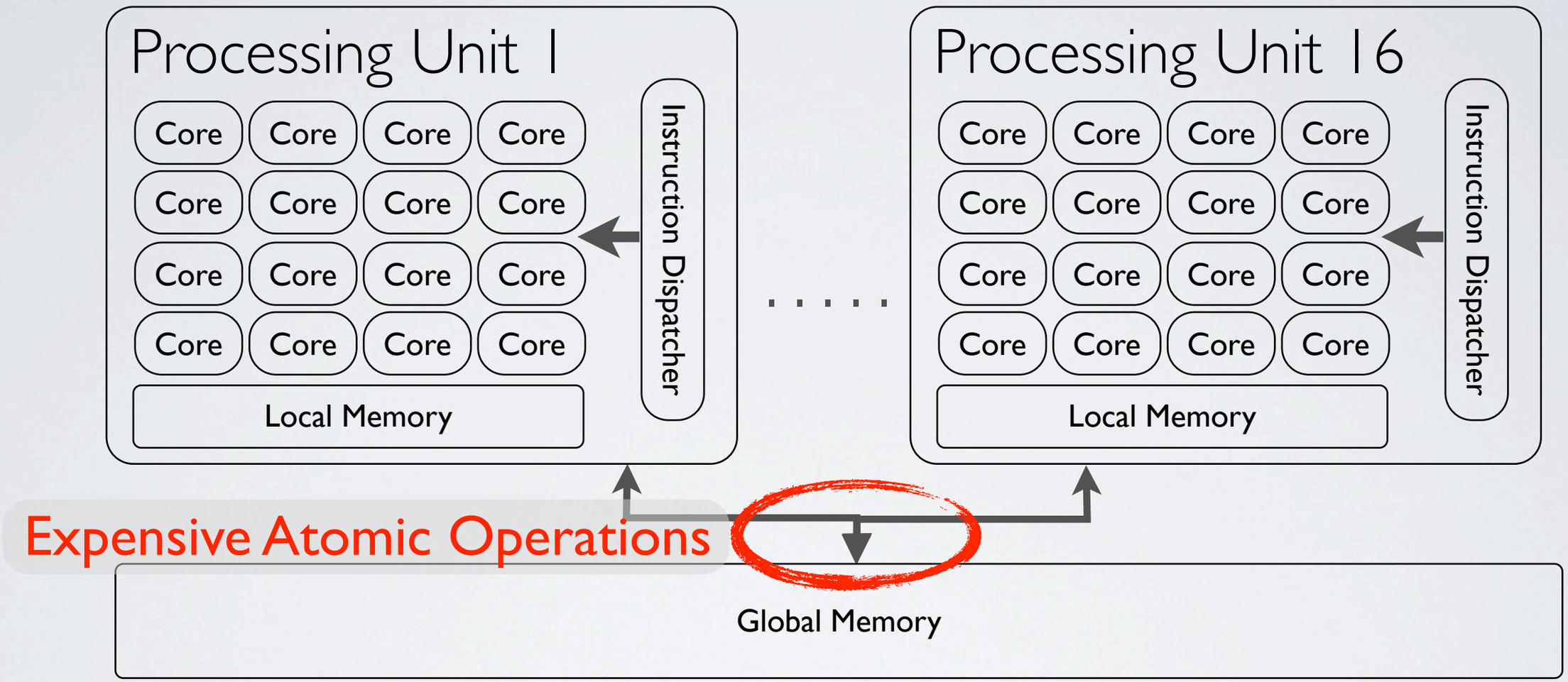


# Foreign-Key Joins on GPUs

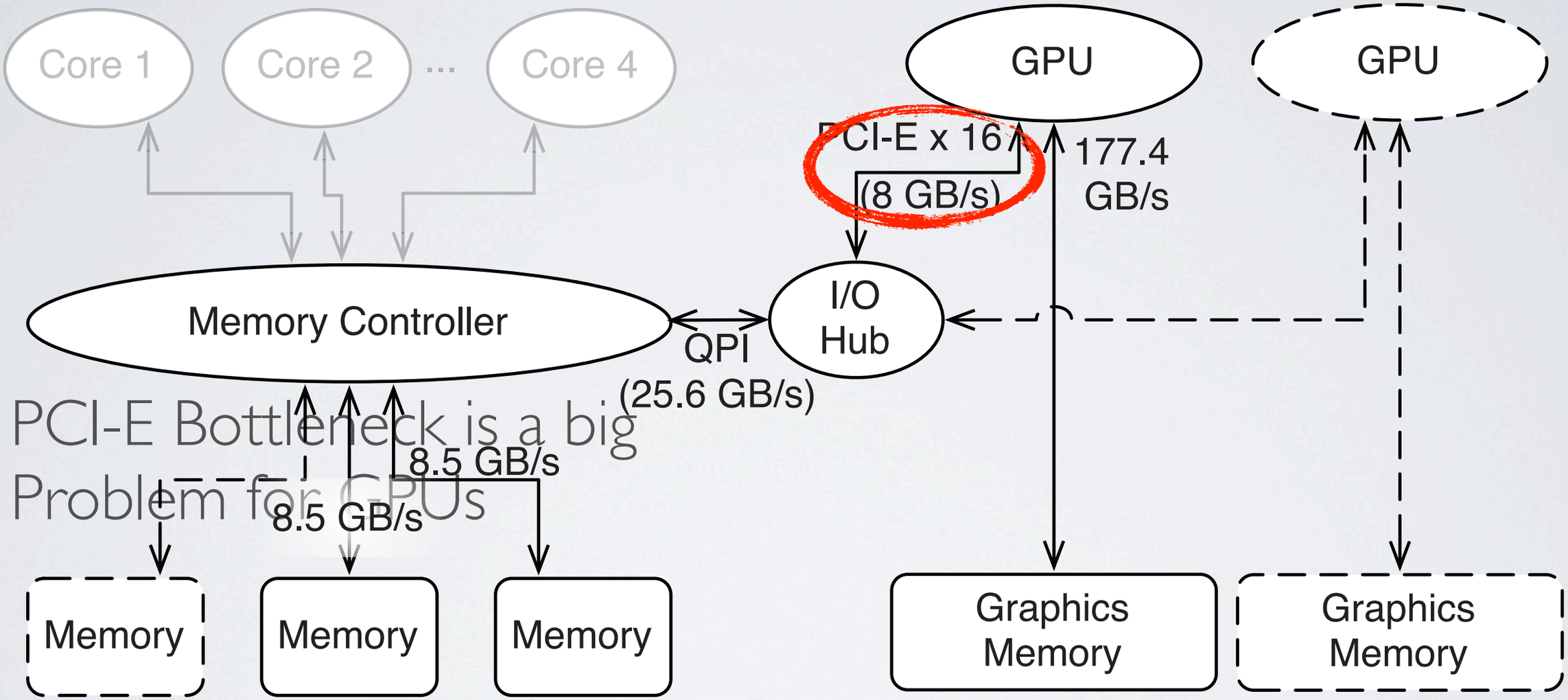
CWI

monetdb

# GPU Architecture Overview



# The Problem: Bandwidth Bound GPUs




- PCI-E Bottleneck is a big Problem for GPUs

# Clustered Foreign-Key Joins on GPUs

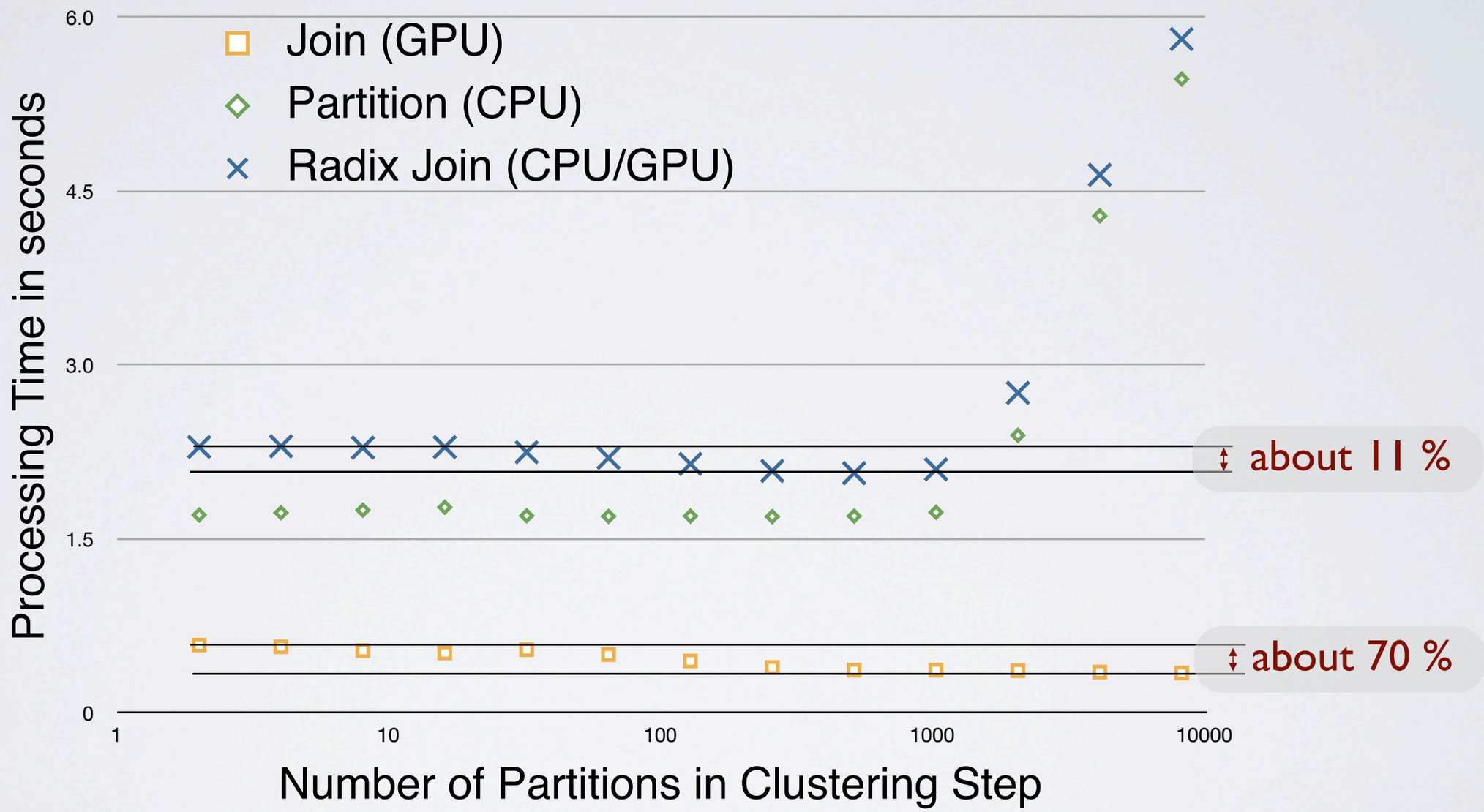
CWI

monetdb

# Clustered Foreign-Key Joins on GPUs

- Port of the Radix-Join from CPU to GPU [He08]
- Improved Cache Performance is Important on GPUs too
- Allows efficient parallel processing of Partitions

# Clustered Foreign-Key Joins on GPUs




CWI

monetdb

# Naïve Foreign-Key Joins on GPUs

CWI

monetdb

# Naïve Foreign-Key Joins on GPUs

- Primary Bottleneck is PCI-E, not GPU Memory
- Idea: Accelerate FK Joins by exploiting the Strength of GPUs:
  - Random Access to target table
- Limit the PCI-E traffic to a minimum:
  - Stream only the Index

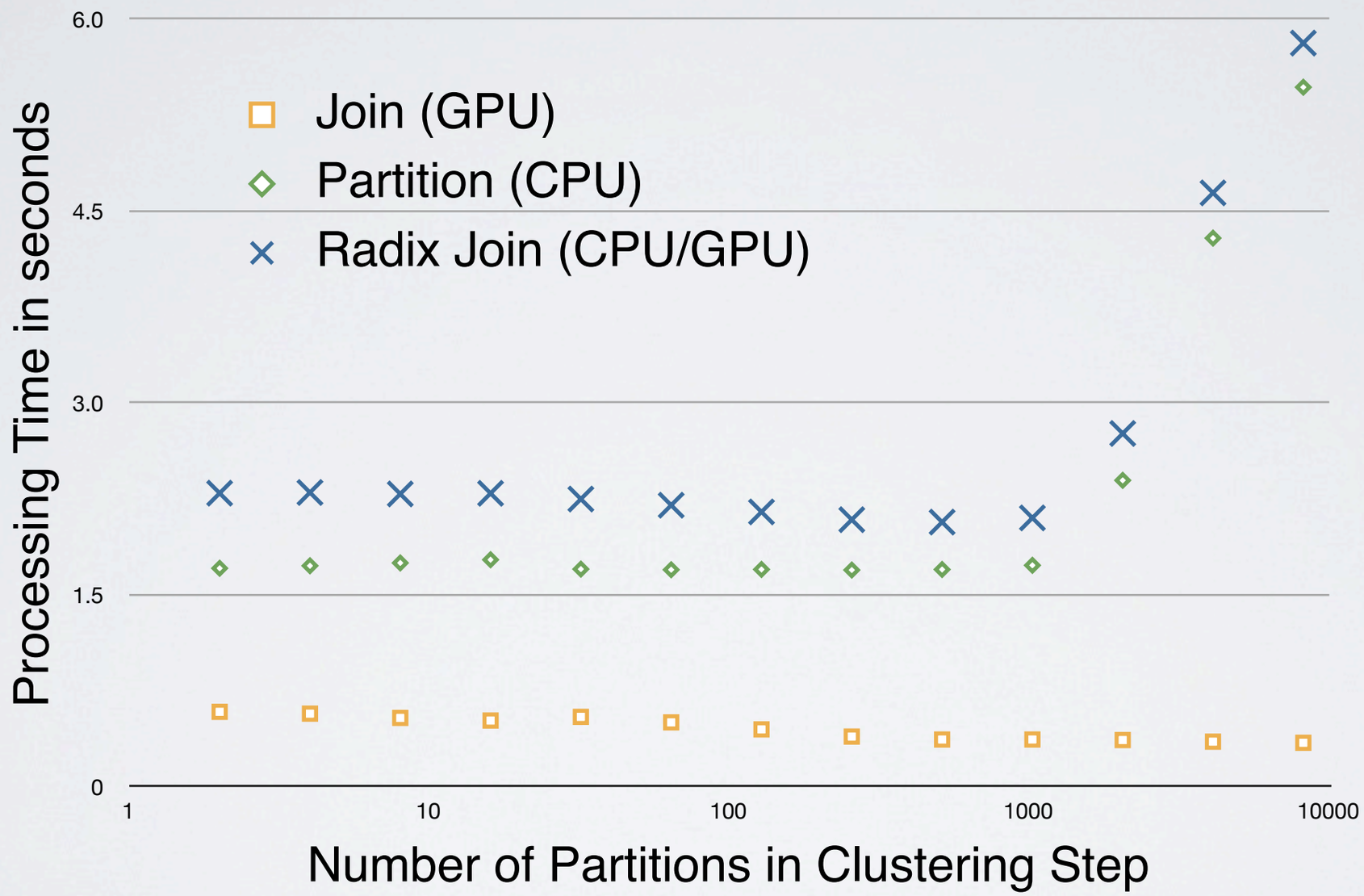
# Asymmetric Foreign-Key Joins on GPUs

- Primary Bottleneck is PCI-E, not GPU Memory
- Idea: Accelerate FK Joins by exploiting the Strength of GPUs:
  - Random Access to target table
- Limit the PCI-E traffic to a minimum:
  - Stream only the Index

CWI

monetdb

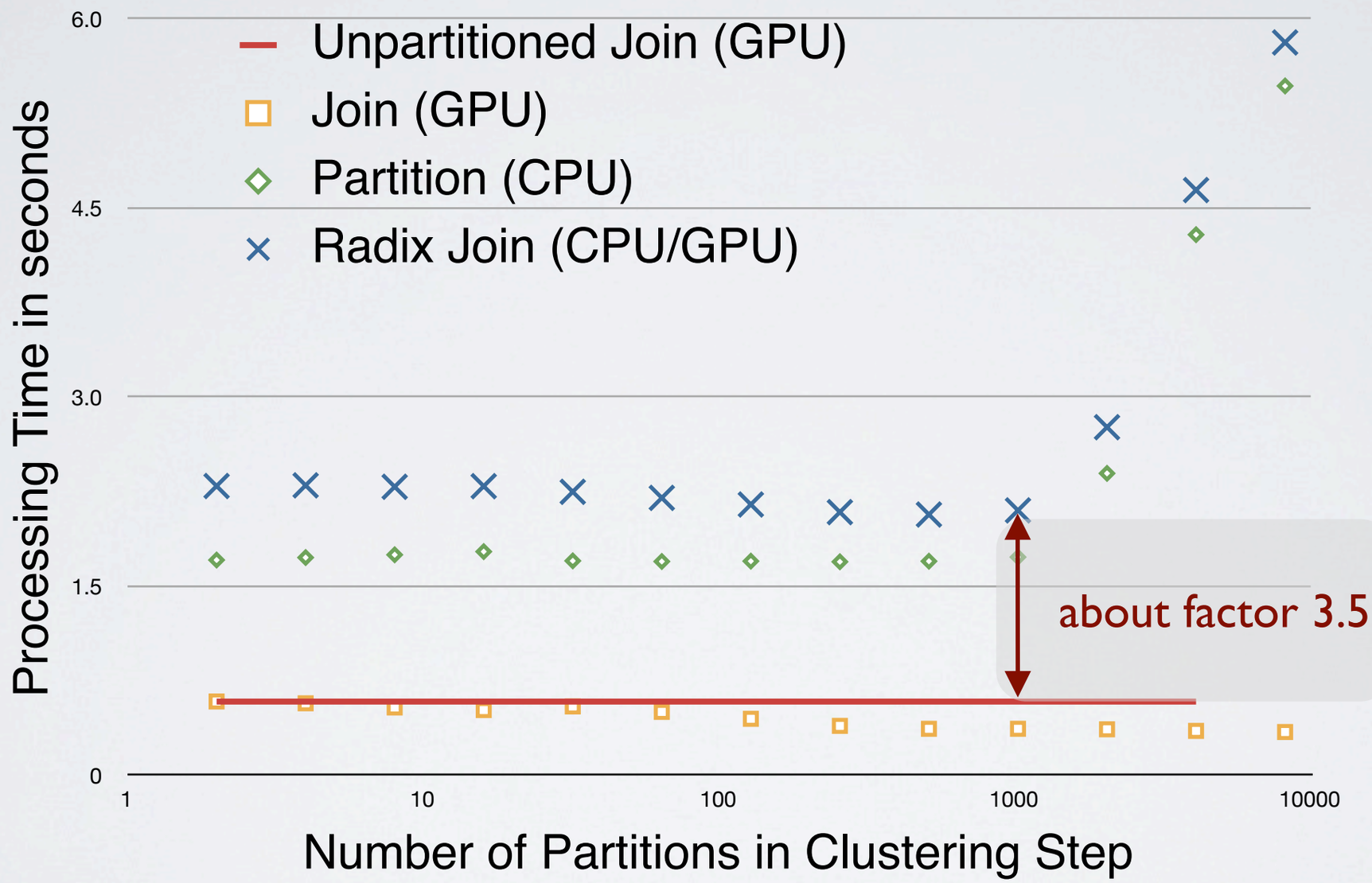
# Asymmetric Foreign-Key Joins on GPUs




CWI

monetdb

# Asymmetric Foreign-Key Joins on GPUs




CWI

monetdb

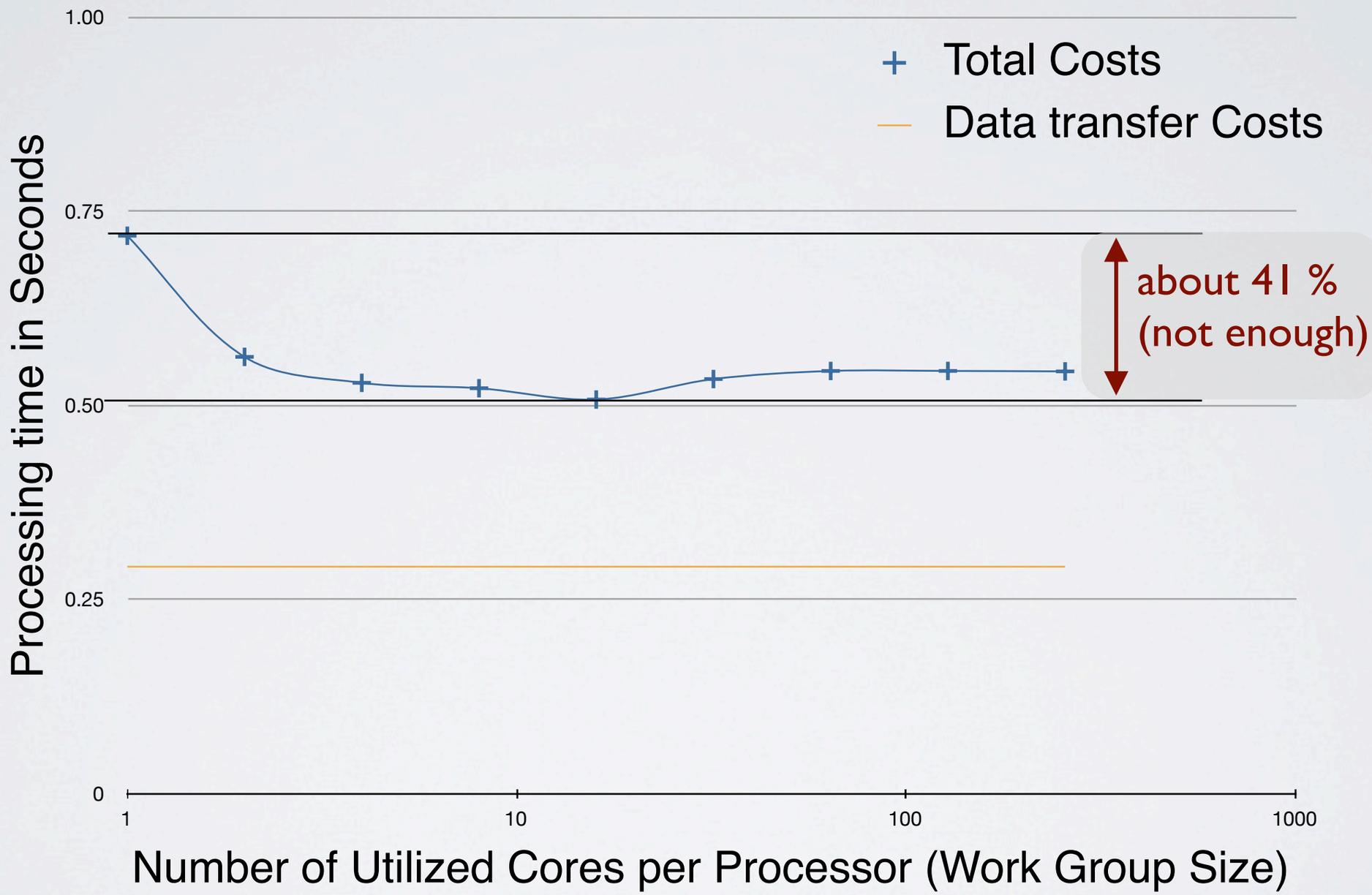
# Asymmetric Foreign-Key Joins on GPUs

- Message: Clustering improves performance on GPUs
- But: Cluster Costs don't pay off in Cache Friendliness
- But what about parallel processing of Partitions?

CWI

monetdb

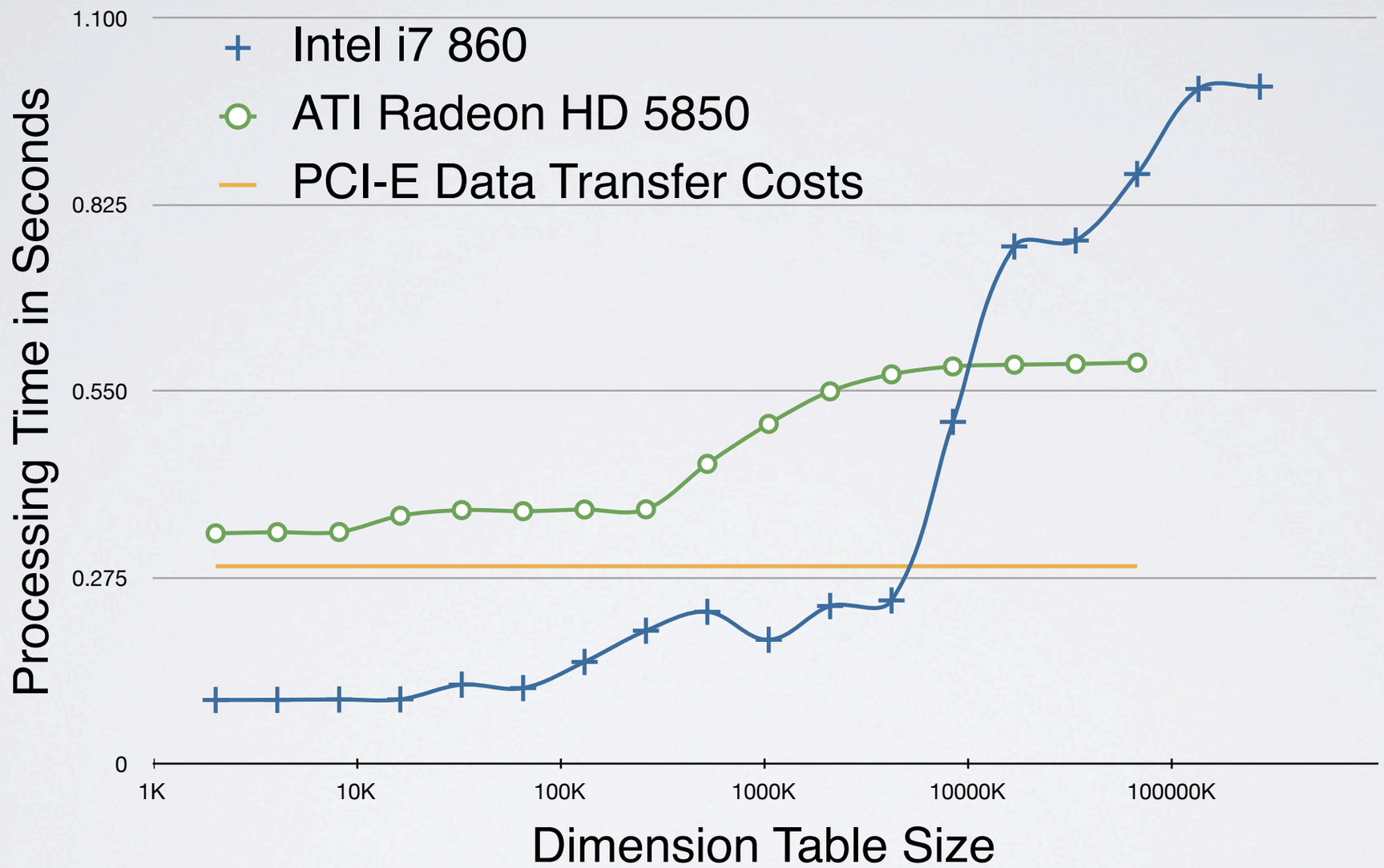
# But what about Parallelization




CWI

monetdb

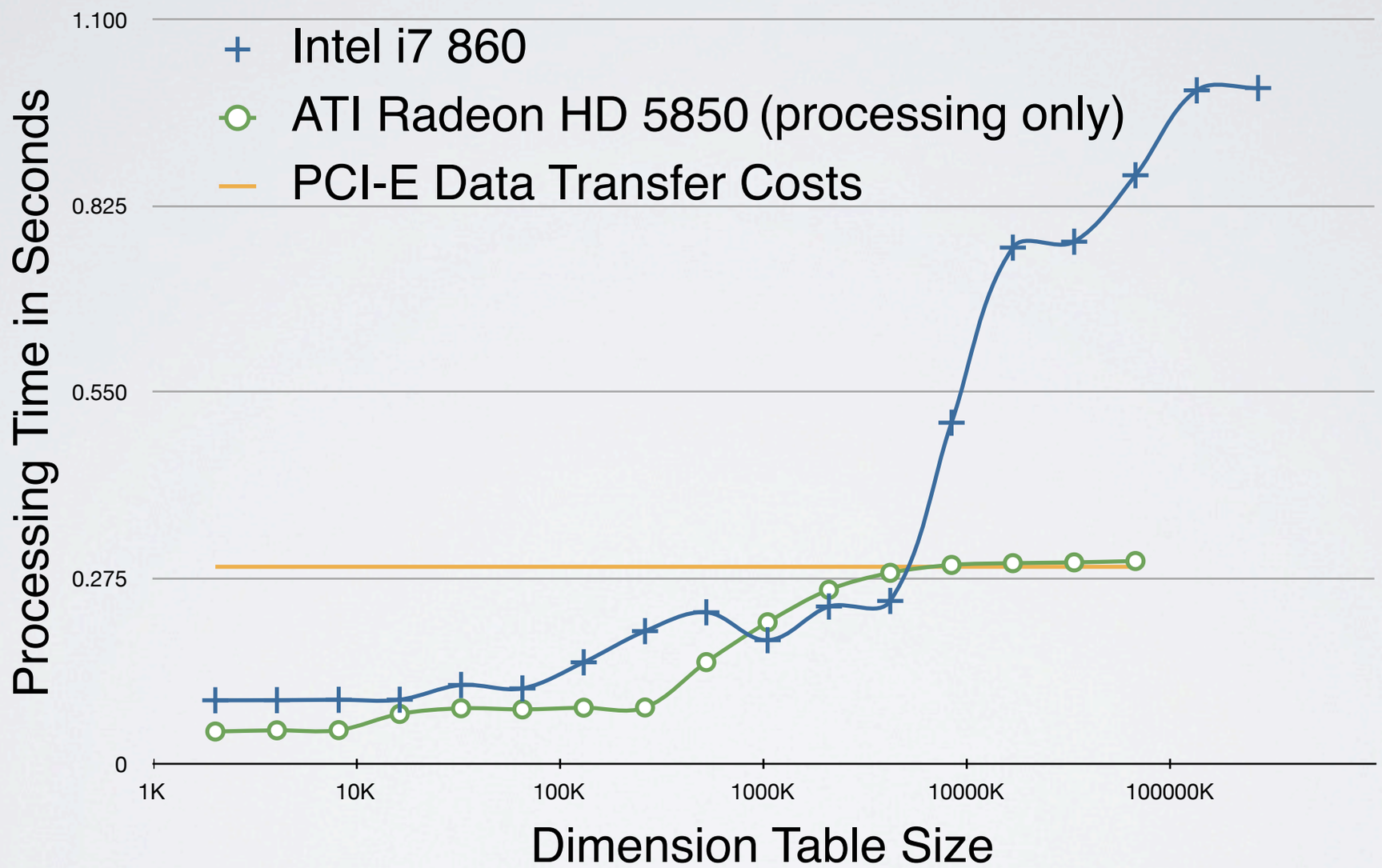
# Foreign-Key Joins on GPUs vs CPUs




CWI

monetdb

# Foreign-Key Joins on GPUs vs CPUs




CWI

monetdb

# Conclusion

- Unclustered FK-Joins Cause Heavy Cache Thrashing
- Thrashing can be offloaded to fast GPU memory
- PCI-E Bandwidth can be saturated without Clustering
- Naïve FK-Joins on GPUs outperform Clustered Joins on CPUs

CWI

monetdb

# Thank you for Listening

CWI

monetdb

Questions? Feedback?

CWI

monetdb