Accelerating Foreign-Key Joins using Asymmetric Memory Channels

Holger Pirk  
tholger@cwi.nl

Stefan Manegold  
manegold@cwi.nl

Martin Kersten  
mk@cwi.nl
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
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
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

Processing Time in seconds

Unpartitioned Join (CPU)
Partition (CPU)
Join (CPU)
Radix Join (CPU)

Number of Partitions in Clustering Step
(Target Table Size: 64M)

about 30 %  
(consistent with [Blanas11])
Clustered Foreign-Key Joins on CPUs

![Bar chart showing partition time in seconds for 256 and 65536 clusters, comparing CPU and GPU performance.]

- **256 Clusters**
  - CPU: 1.695271 seconds
  - GPU: 11.040 seconds

- **65536 Clusters**
  - CPU: 6.492860 seconds
  - GPU: 20.512 seconds
Foreign-Key Joins on GPUs
GPU Architecture Overview

Expensive Atomic Operations

Global Memory
The Problem: Bandwidth Bound GPUs

- PCI-E Bottleneck is a big problem for GPUs

---

**Diagram:**

- Memory Controller
  - Core 1
  - Core 2
  - ... Core 4

- QPI (25.6 GB/s)
  - 8.5 GB/s

- GPU
  - PCI-E x 16 (8 GB/s)
  - 177.4 GB/s

- Graphics Memory

---

CWI
Clustered Foreign-Key Joins on GPUs
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

![Graph showing processing time vs. number of partitions for different join algorithms on GPUs.](image-url)
Naïve Foreign-Key Joins on GPUs
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
Asymmetric Foreign-Key Joins on GPUs

- Join (GPU)
- Partition (CPU)
- Radix Join (CPU/GPU)
Asymmetric Foreign-Key Joins on GPUs

Processing Time in seconds vs. Number of Partitions in Clustering Step

- Unpartitioned Join (GPU)
- Join (GPU)
- Partition (CPU)
- Radix Join (CPU/GPU)

About factor 3.5
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?
But what about Parallelization

- Total Costs
- Data transfer Costs

Processing time in Seconds

Number of Utilized Cores per Processor (Work Group Size)

About 41% (not enough)
Foreign-Key Joins on GPUs vs CPUs

Processing Time in Seconds

- Intel i7 860
- ATI Radeon HD 5850
- PCI-E Data Transfer Costs

Dimension Table Size
Foreign-Key Joins on GPUs vs CPUs

![Graph showing processing time in seconds vs dimension table size, comparing Intel i7 860, ATI Radeon HD 5850 (processing only), and PCI-E data transfer costs.](image-url)
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
Thank you for Listening
Questions? Feedback?