Advanced Data Management for Data Analysis

Stefan Manegold

Data Management @ LIACS

Group leader Database Architectures
Centrum Wiskunde & Informatica (CWI)
Amsterdam

s.manegold@liacs.leidenuniv.nl
http://www.cwi.nl/~manegold/

ADM: Agenda

- 07.09.2022: Lecture 1: **Introduction**
- 14.09.2022: Lecture 2: **SQL Recap**

  (plus Assignment 1 [in groups; 3 weeks]: TPC-H benchmark)
- 21.09.2022: Lecture 3: **Column-Oriented Database Systems (1/6) - Motivation & Basic Concepts**
- 28.09.2022: Lecture 4: **Column-Oriented Database Systems (2a/6) - Selected Execution Techniques (1/2)**
- 05.10.2022: Lecture 5: **Column-Oriented Database Systems (2b/6) - Selected Execution Techniques (2/2)**

  (plus Assignment 2 [in groups; 4 weeks]: Compression techniques)
- 12.10.2022: Lecture 6: **Column-Oriented Database Systems (3/6) - Cache Conscious Joins**
- 19.10.2022: Lecture 7: **Column-Oriented Database Systems (4/6) - "Vectorized Execution"**
- ~~26.10.2022~~: ***No lecture!***
- 02.11.2022: Lecture 8: **DuckDB: An embedded database for data science (1/2) (guest lecture & *hands-on*)**

  (plus Assignment 3 [individual; 2 weeks]: Analysing NYC Cab dataset with DuckDB)
- 09.11.2022: Lecture 9: **DuckDB: An embedded database for data science (2/2) (guest lecture & *hands-on*)**
- 16.11.2022: Lecture 10: **Branch Misprediction & Predication**

  (plus Assignment 4 [individual; 2 weeks]: Predication)
- 23.11.2022: Lecture 11: **Column-Oriented Database Systems (5/6) - Adaptive Indexing**
- 30.11.2022: Lecture 12: **Column-Oriented Database Systems (6/6) - Progressive Indexing**

ADM: Literature

- **Column-Oriented Database Systems (3/6) - Cache Conscious Joins**
  - “Cache Conscious Algorithms for Relational Query Processing”. Shatdal, Kant, Naughton. VLDB’94.
  - “Fast Joins using Join Indices”. Li and Ross. VLDBJ 8:1-24, 1999.
  - “Optimizing main-memory join on modern hardware”. Boncz, Manegold, Kersten, TKDE 14(4): 709-730, 2002.
    - “Database Architecture Optimized for the New Bottleneck: Memory Access”. Boncz, Manegold, Kersten. VLDB’99.
    - “What Happens During a Join? Dissecting CPU and Memory Optimization Effects”. Manegold, Boncz, Kersten. VLDB’00.
    - “Optimizing database architecture for the new bottleneck: memory access”. Manegold, Boncz, Kersten. VLDB J. 9(3): 231-246, 2000.
    - “Generic Database Cost Models for Hierarchical Memory Systems”. Manegold, Boncz, Kersten. VLDB’02.
  - “Cache-Conscious Radix-Decluster Projections”. Manegold, Boncz, Nes. VLDB’04.

Join Algorithms

- Nested Loop: $O(n^2)$
  - sequential access to both outer and inner input
  - repetitive scan of inner input

- Sort-Merge: $O(n \log n)$
  - random access during sort ("investment")
  - single sequential scan during merge ("benefit")

- Hash-Join: $O(n)$
  - sequential scan over inputs
  - random access to hash table (build & probe)

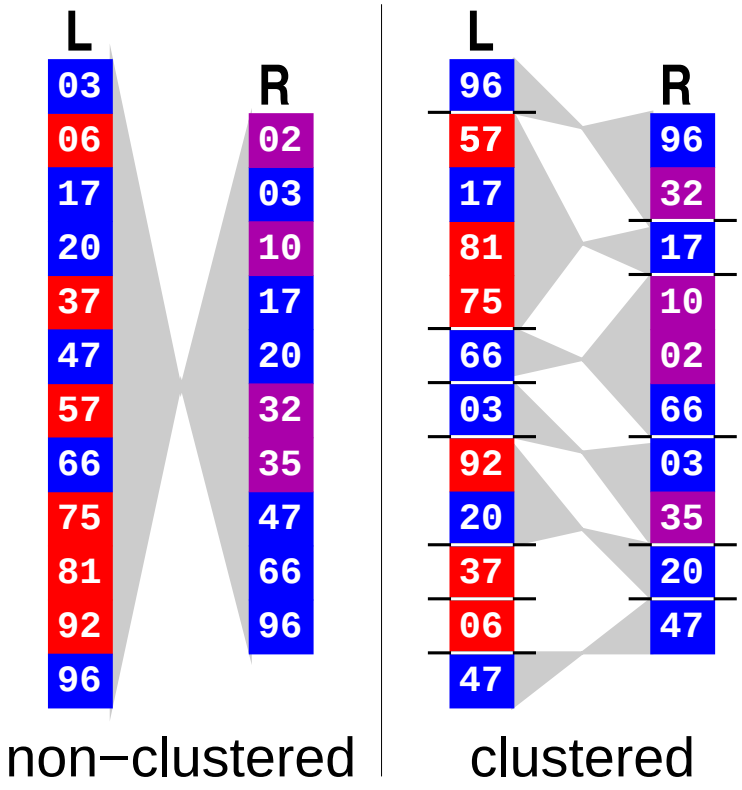
Partitioned Joins: Idea

Phase 1:

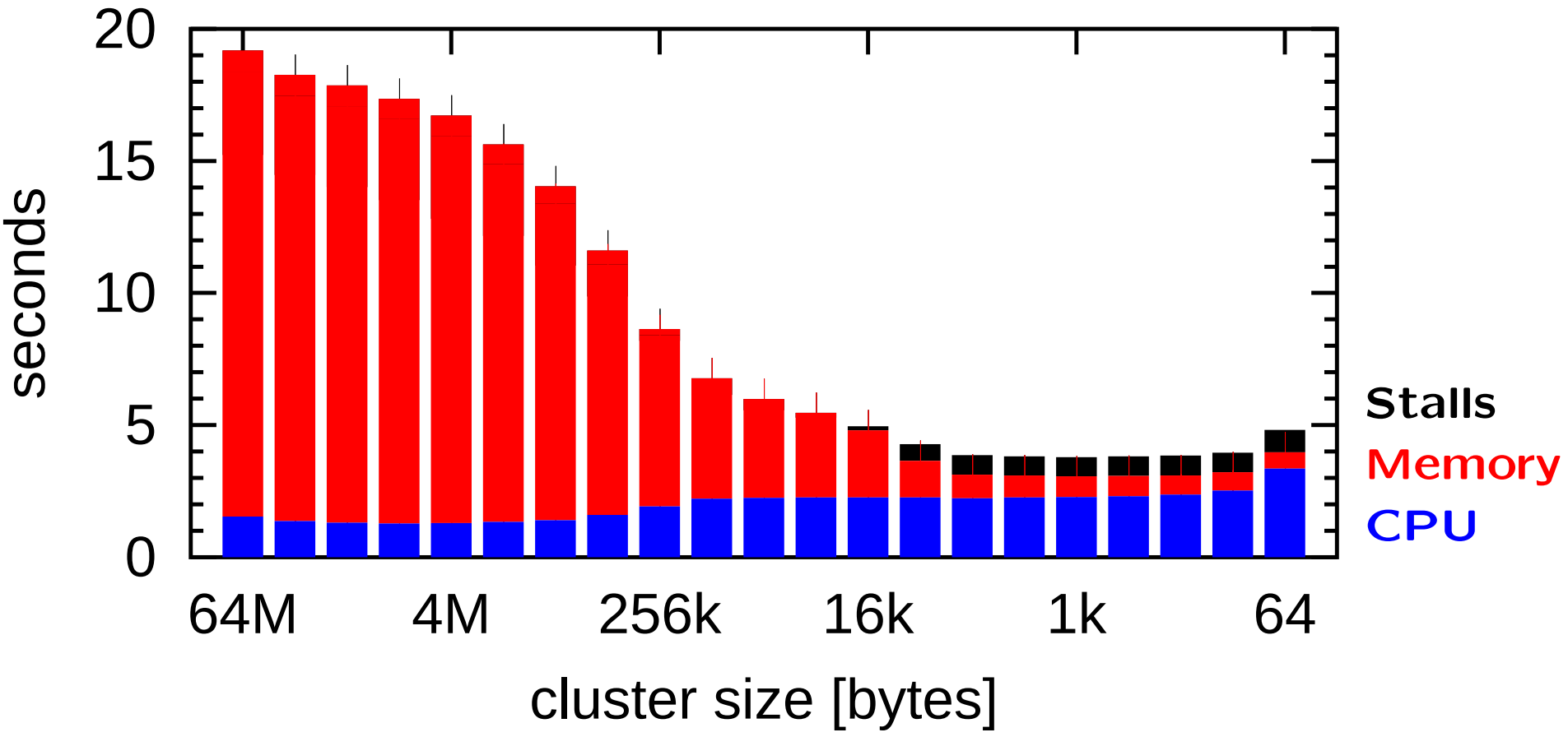
- Cluster both input relations
- Create clusters that fit in memory cache
- Restrict random memory access to smallest cache
- Avoid cache capacity misses

Phase 2:

- Join matching clusters



Partitioned Joins: Phase 2: Join matching clusters



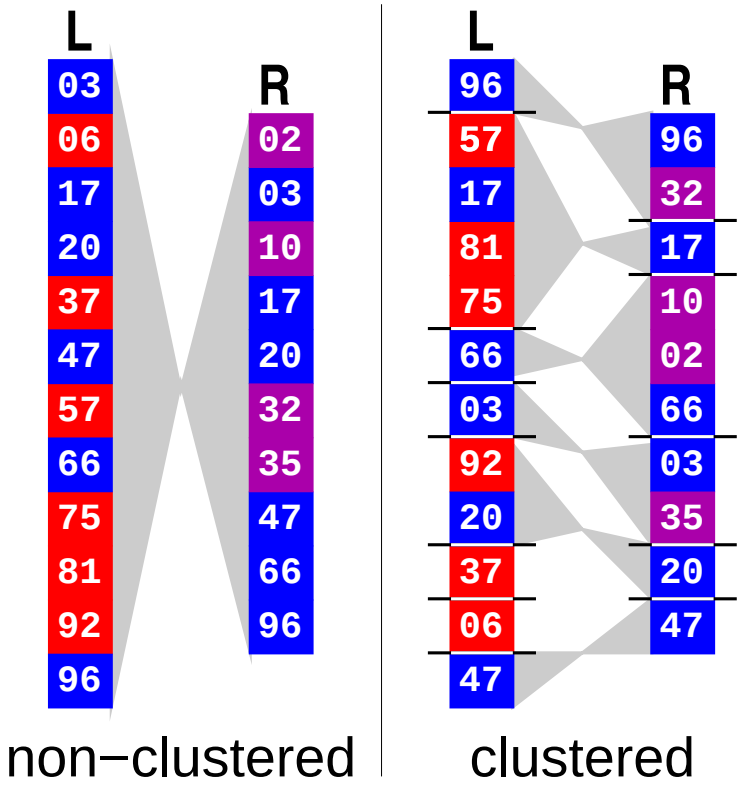
Partitioned Joins: Idea

Phase 1:

- Cluster both input relations
- Create clusters that fit in memory cache
- Restrict random memory access to smallest cache
- Avoid cache capacity misses

Phase 2:

- Join matching clusters

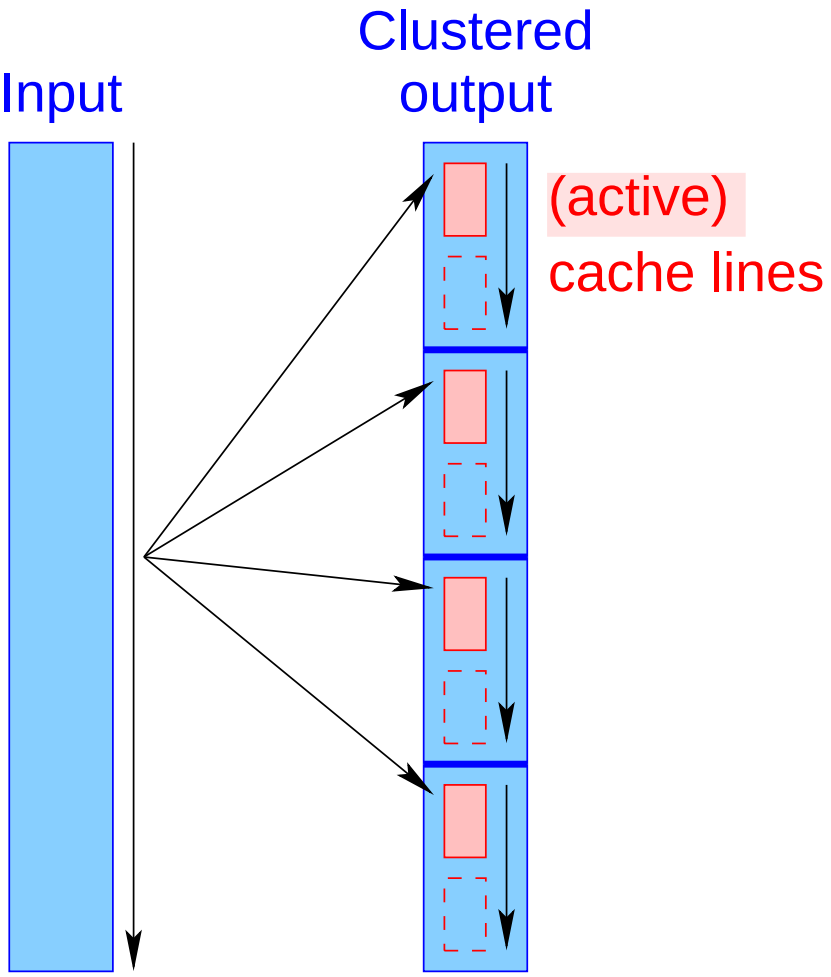


$\Longrightarrow$ Phase 2 is "solved", but what about Phase 1?

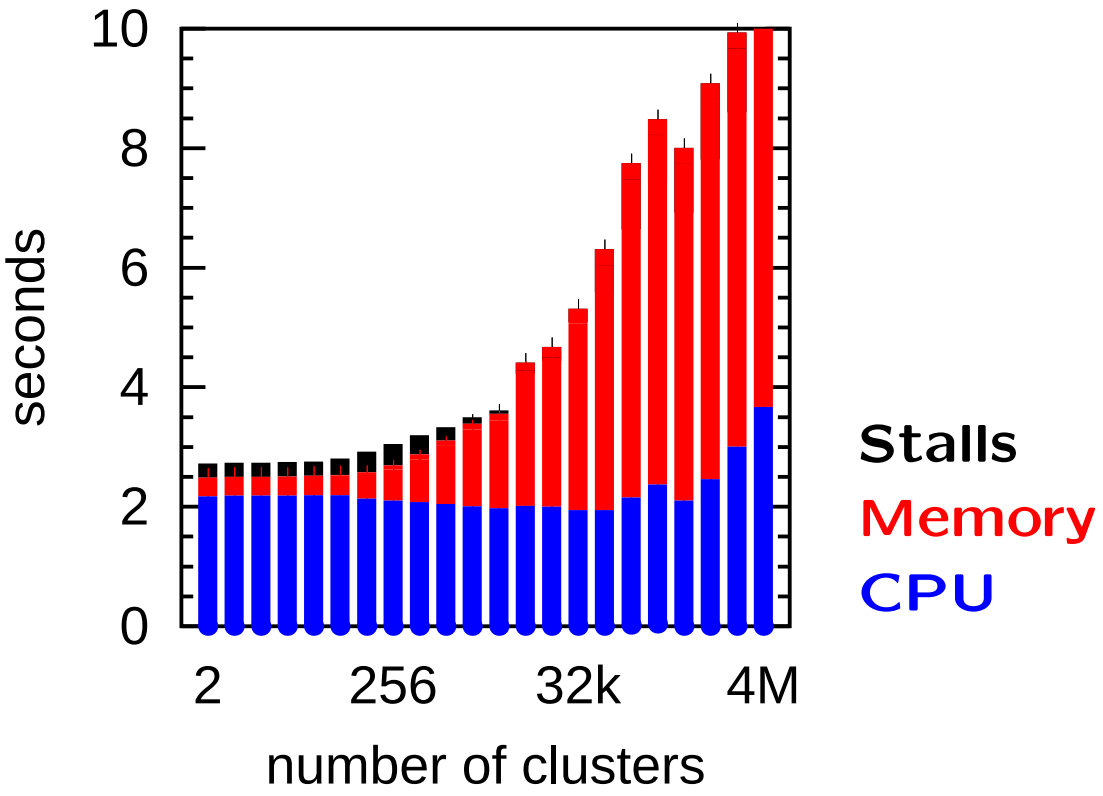
Partitioned Joins: Straightforward Clustering

- Problem:
  Number of clusters exceeds number of cache lines
  $\Rightarrow$ cache thrashing

- Solution:
  Multi-pass clustering

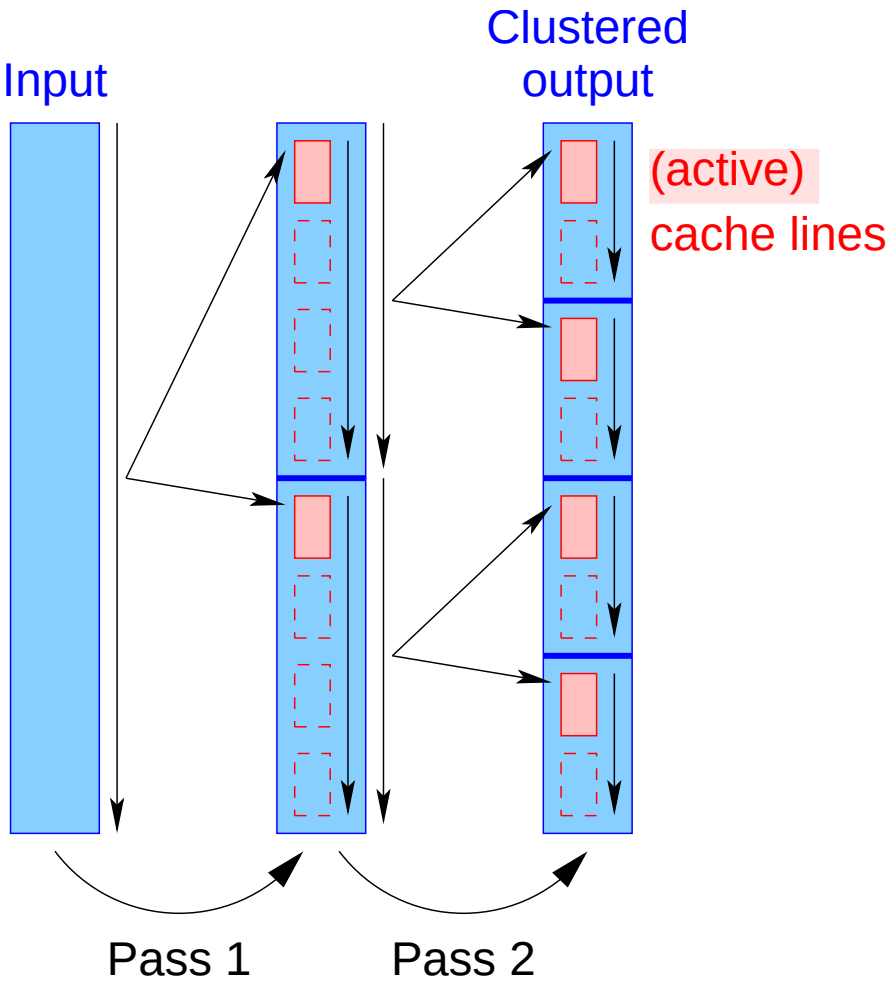


Partitioned Joins: Phase 1: Cluster both input relations

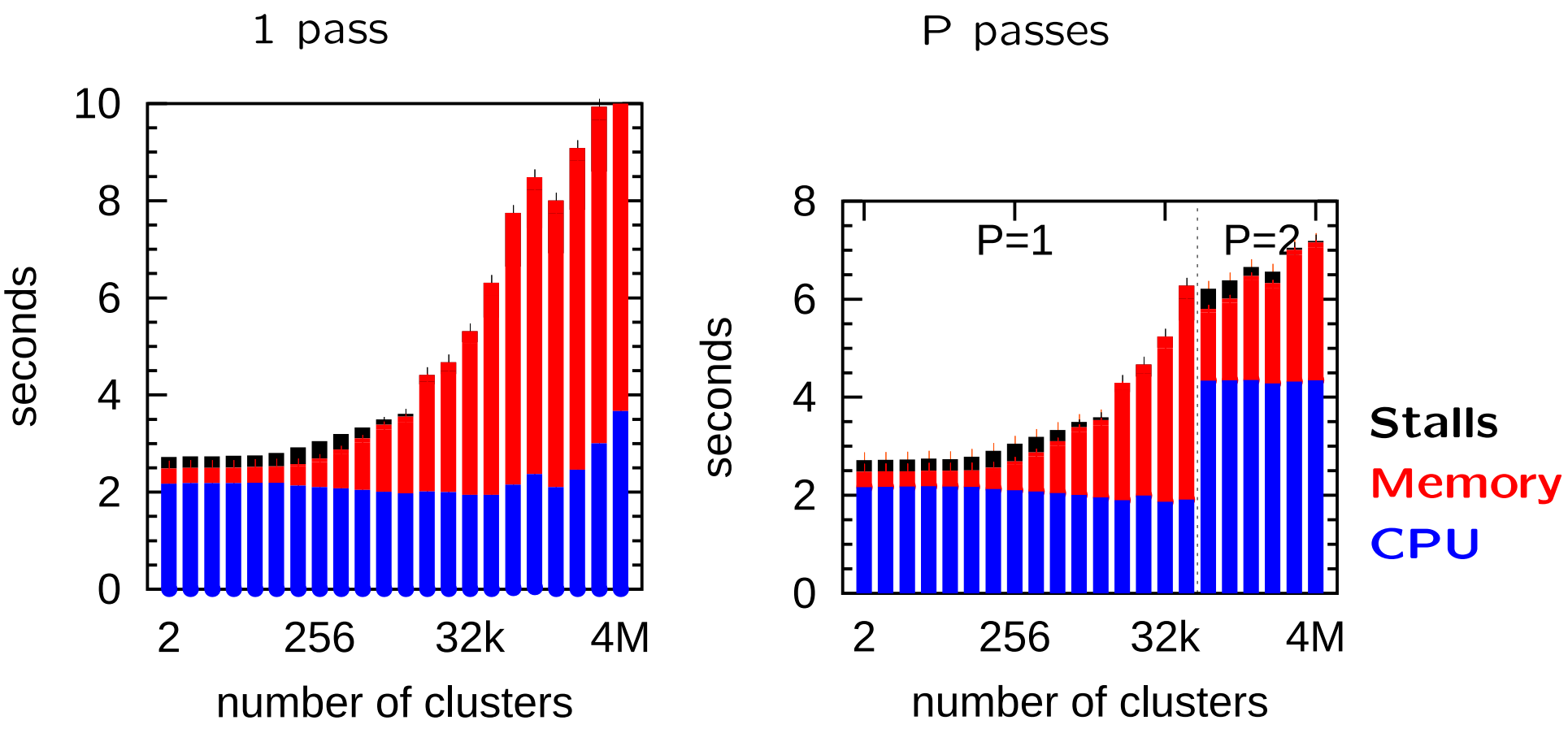


Partitioned Joins: Multi-Pass Clustering

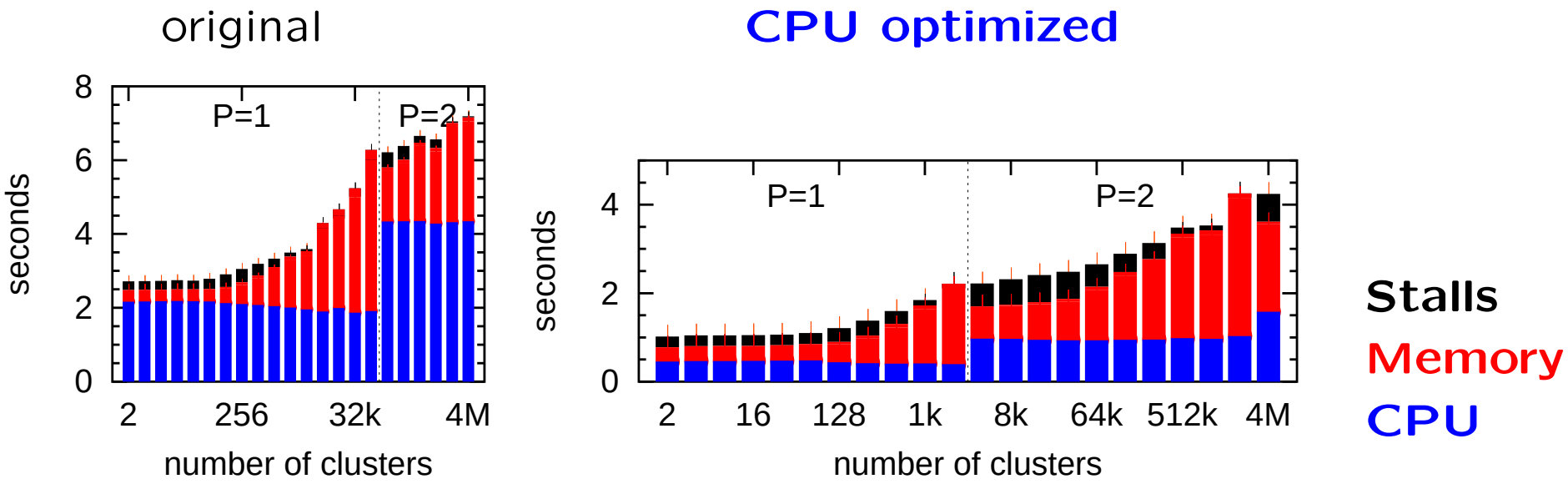
- Limit number of clusters per pass
- Avoid cache/TLB thrashing
- Trade memory cost for CPU cost



Partitioned Joins: Phase 1: Cluster both input relations



Partitioned Joins: Phase 1: Cluster both input relations



Cache-Conscious Joins

Radix-Cluster

- Radix-Partitioned Hash Join
  - create partitions << CPU cache
  - small partitions ➔ many partitions
  - many partitions ➔ multiple passes needed

•“Database Architecture Optimized for the New Bottleneck: Memory Access” VLDB’99
•“Generic Database Cost Models for Hierarchical Memory Systems”, VLDB’02 (all Manegold, Boncz, Kersten)

Cache-Conscious Joins

Radix-Cluster

- Radix-Partitioned Hash Join
  - create partitions << CPU cache
  - small partitions ➔ many partitions
  - many partitions ➔ multiple passes needed

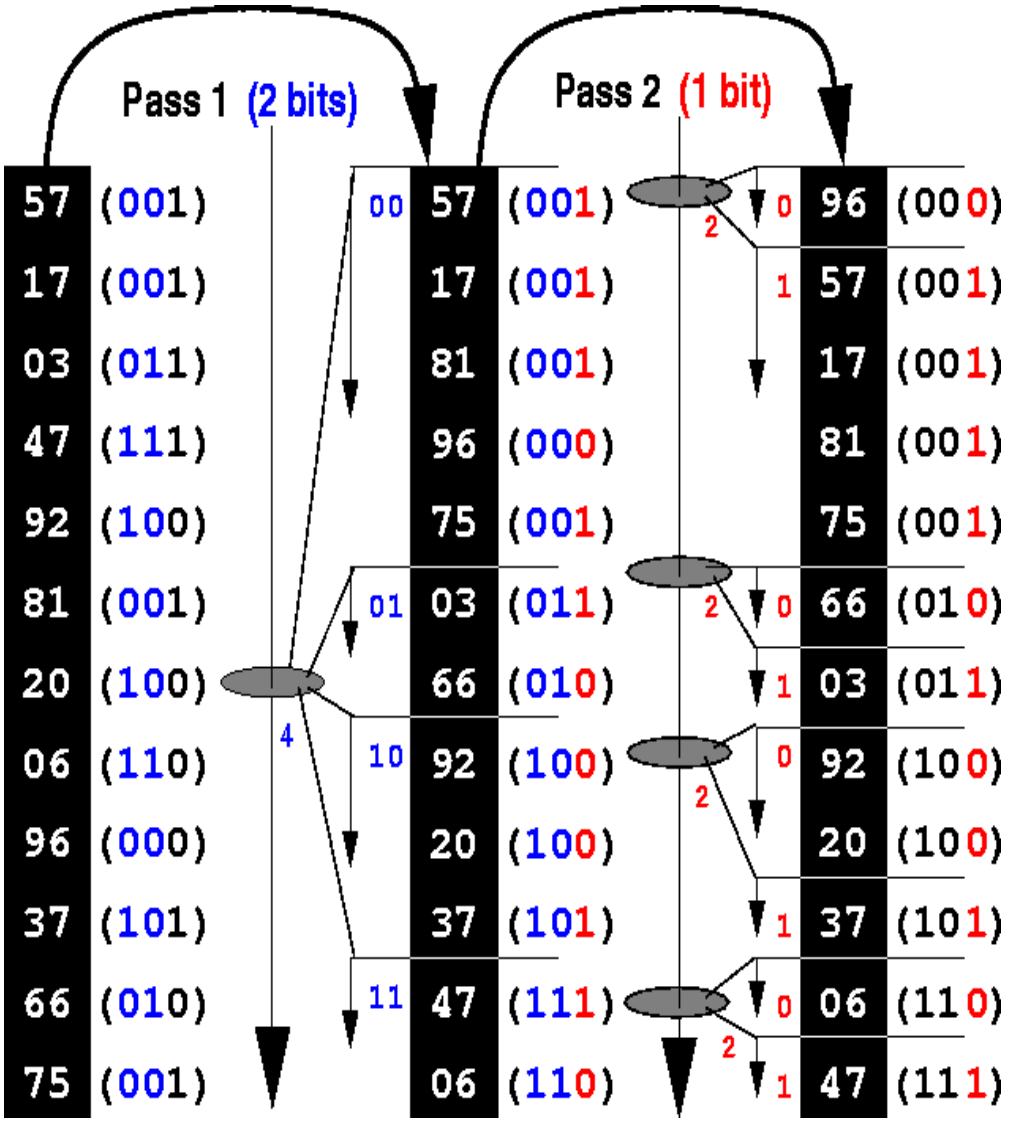
- Radix-Cluster
  - Radix-Sort with early stopping
    - Each pass looks at Bi higher-most radix bits
    - Splitting each input cluster into 2^{Bi} output clusters
  - leaves relation partially ordered

•“Database Architecture Optimized for the New Bottleneck: Memory Access” VLDB’99
•“Generic Database Cost Models for Hierarchical Memory Systems”, VLDB’02
(all Manegold, Boncz, Kersten)

Cache-Conscious Joins

Radix-Cluster

- Multiple clustering passes
- Limit number of clusters per pass
- Avoid cache/TLB trashing
- Trade memory cost for CPU cost
- Any data type (hashing)

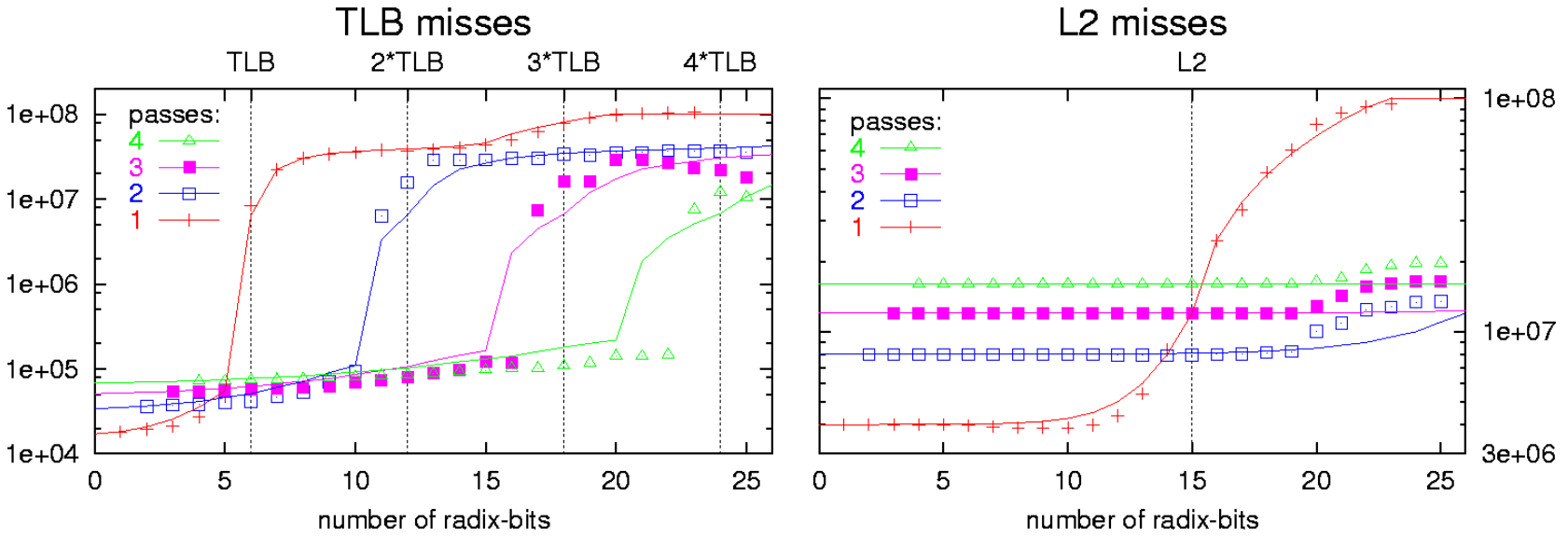


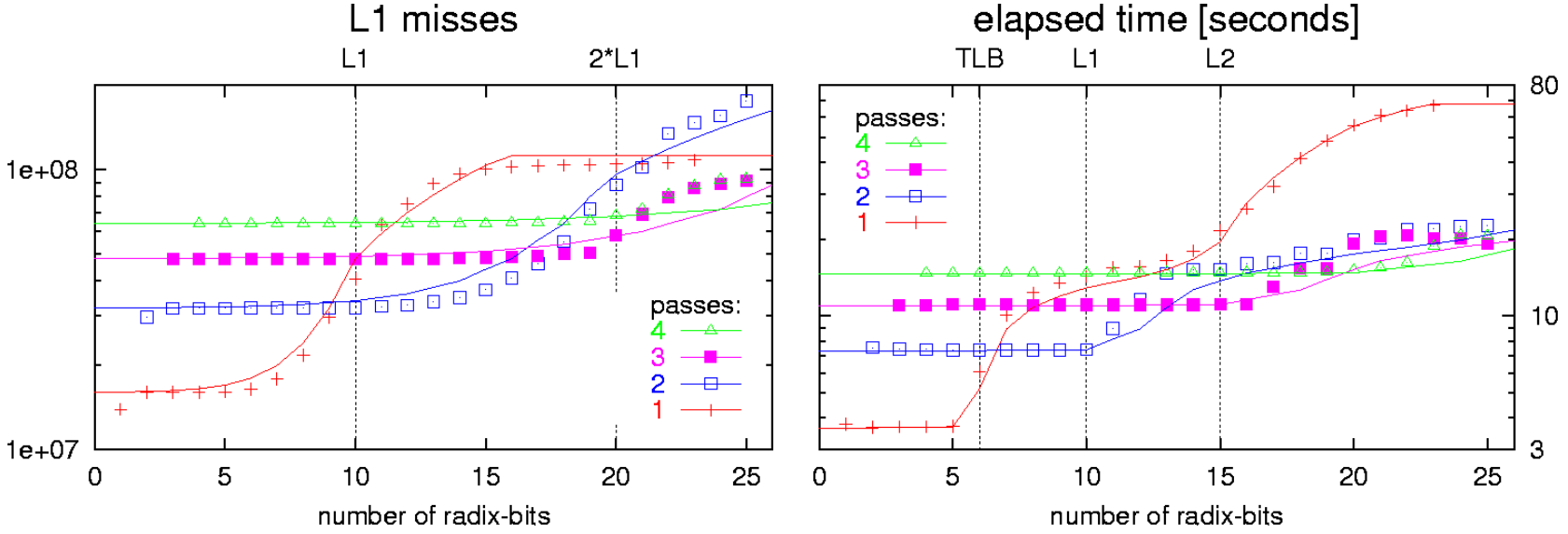
•"Database Architecture Optimized for the New Bottleneck: Memory Access" VLDB'99

•"Generic Database Cost Models for Hierarchical Memory Systems", VLDB'02 (all Manegold, Boncz, Kersten)

Cache-Conscious Joins

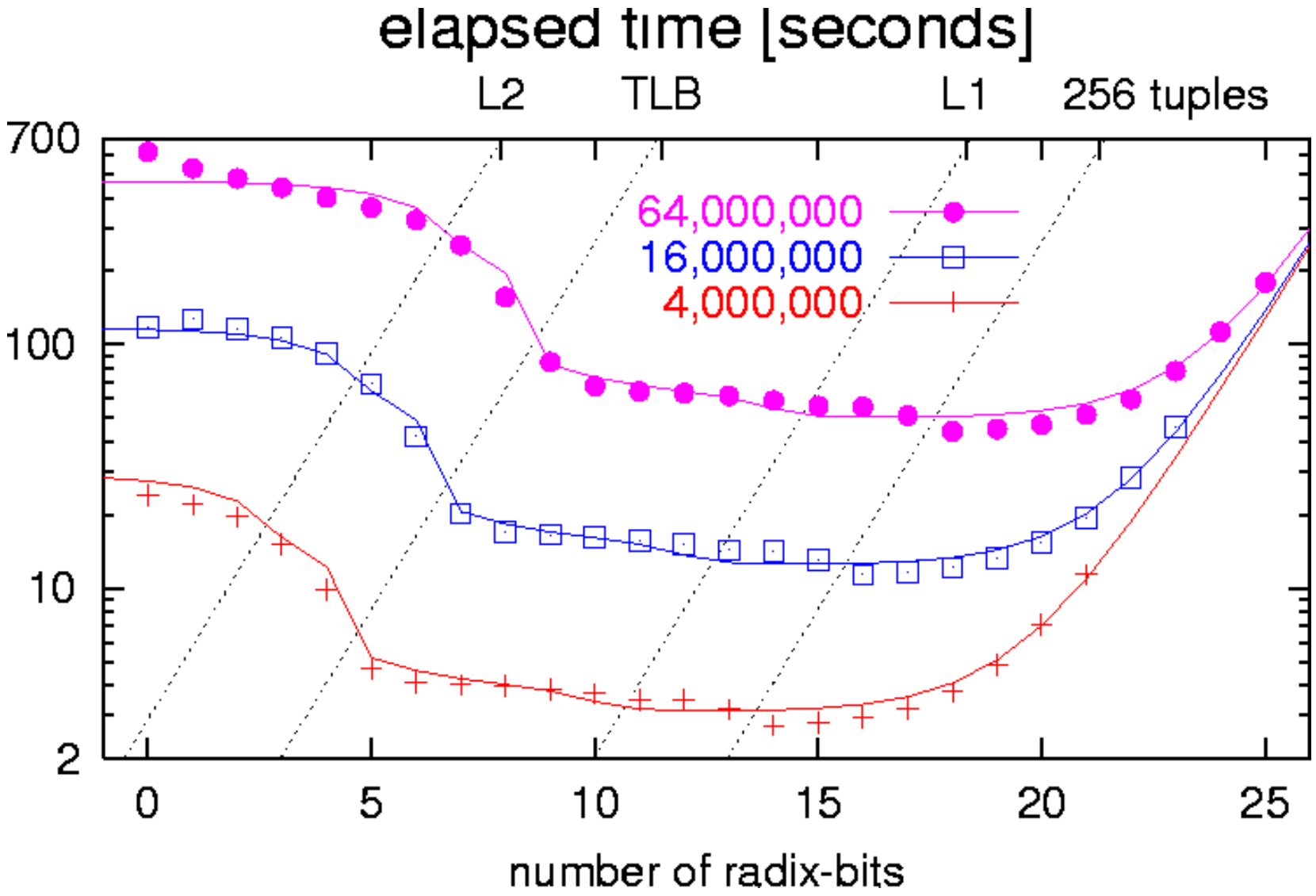
Accurate Cache Miss Cost Modeling





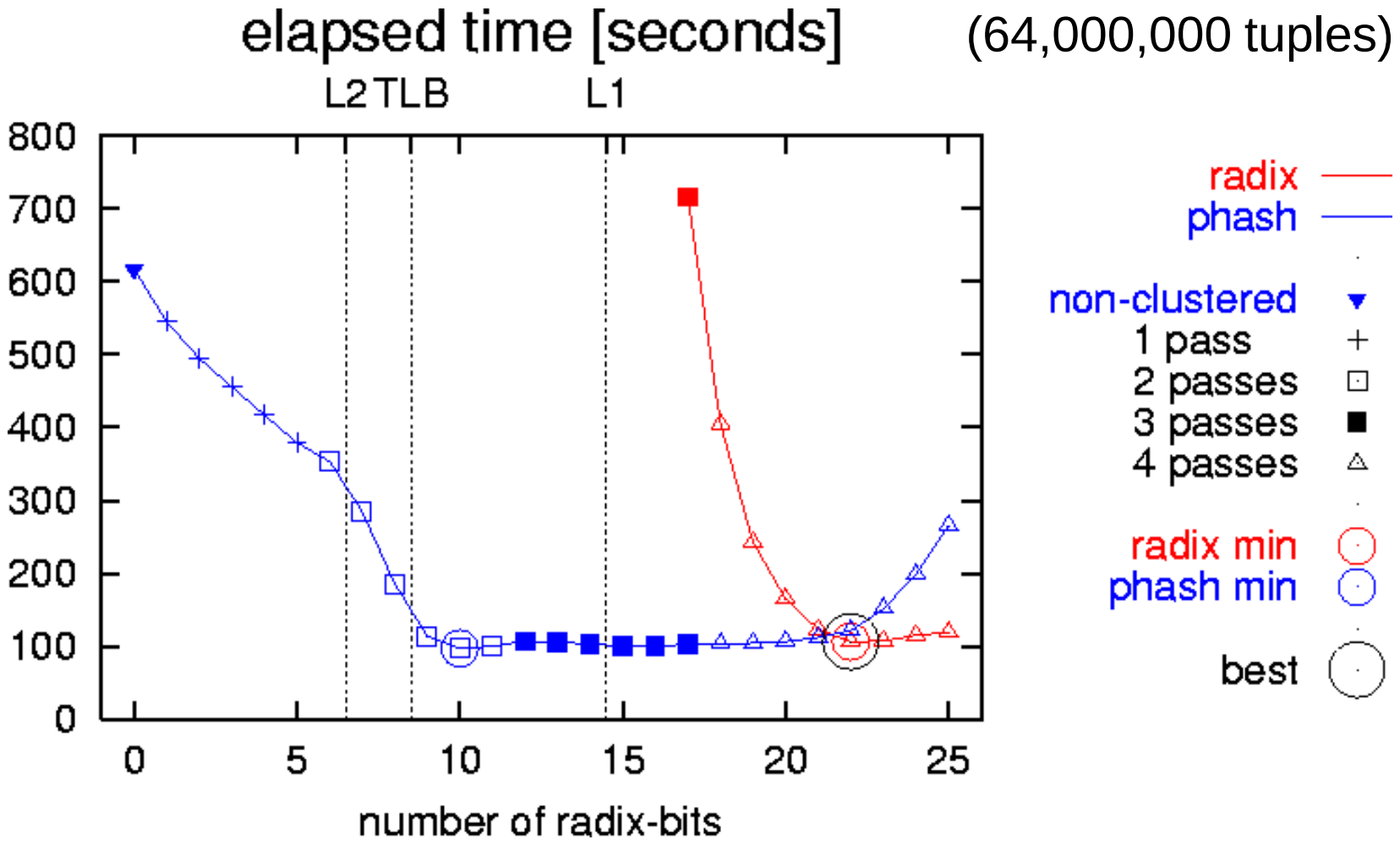
Cache-Conscious Joins

Partitioned Hash Join



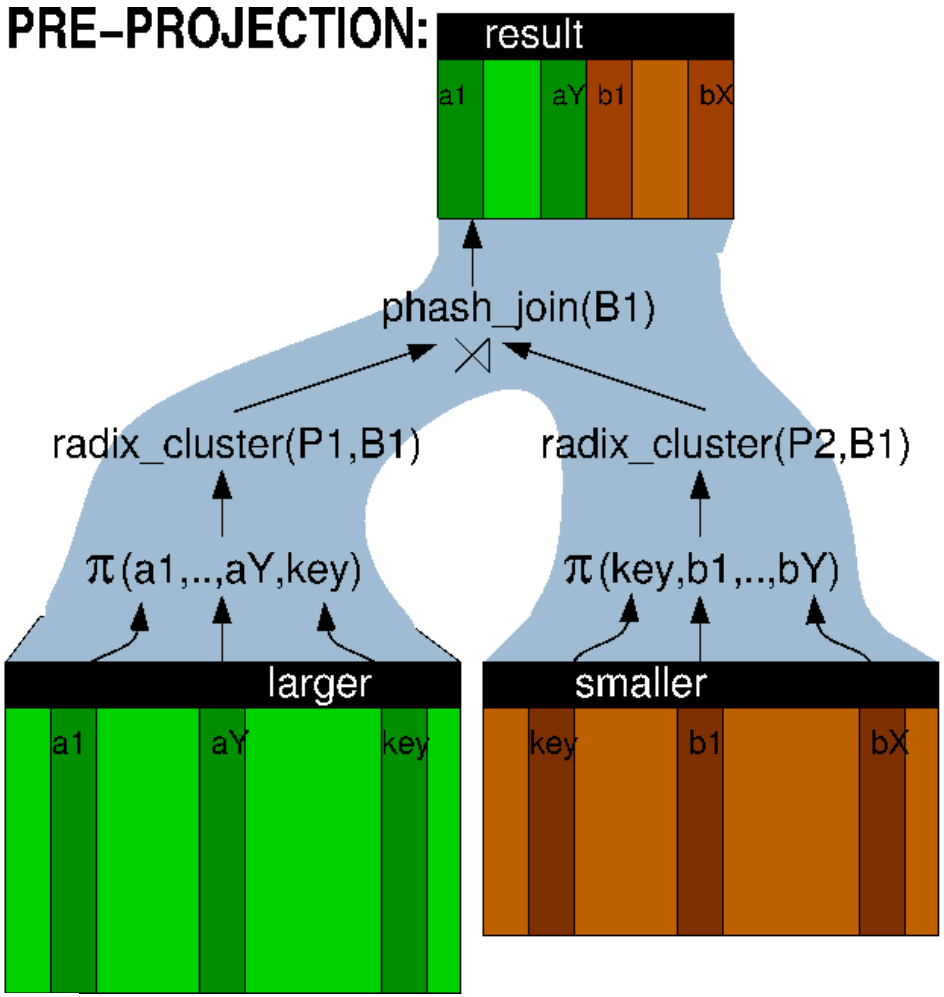
Cache-Conscious Joins

Radix-Clustered Hash-Join: overall perf



Cache-Conscious Joins

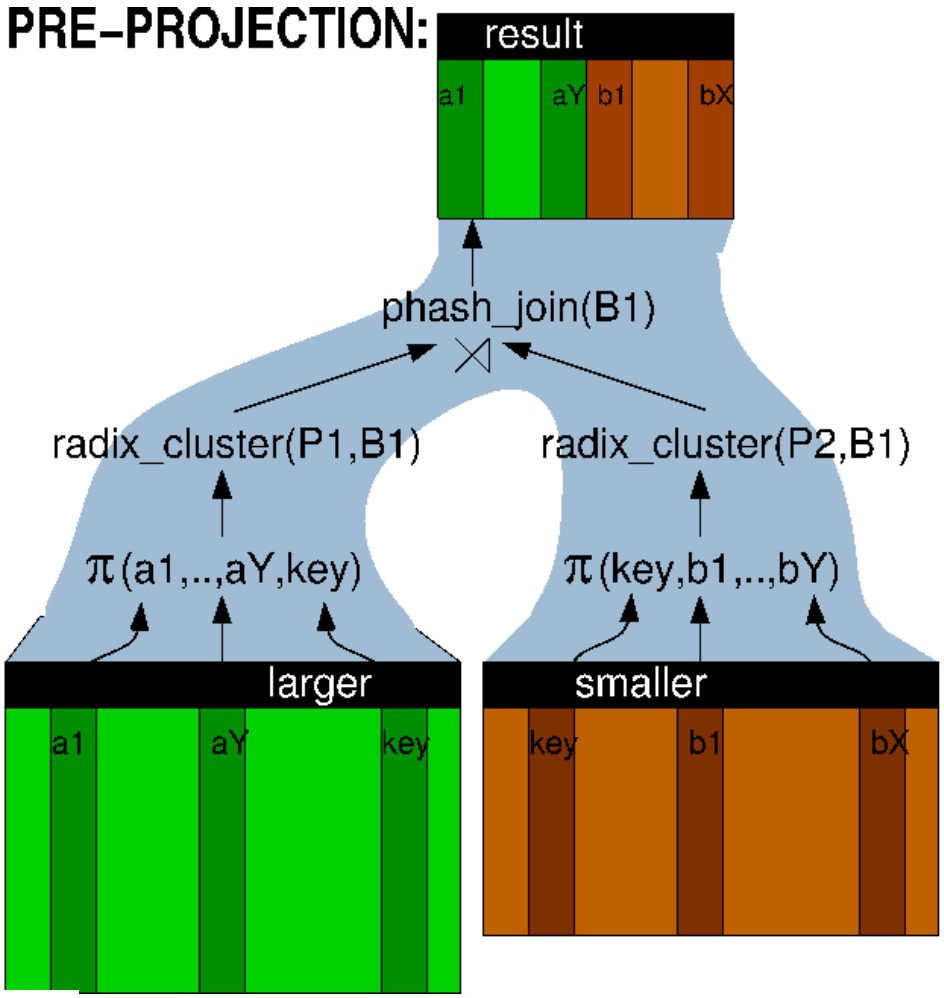
Pre-Projection vs. Post-Projection

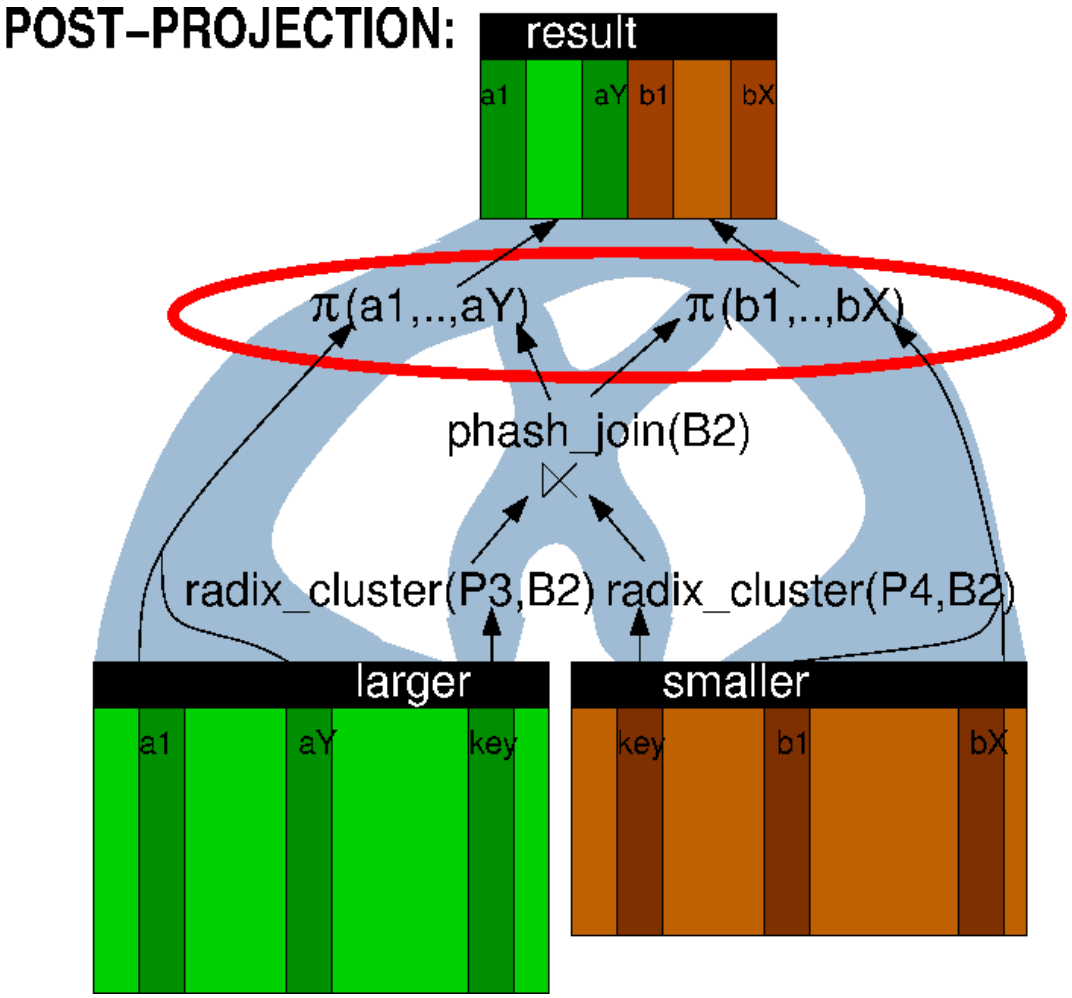


Cache-Conscious Joins

Pre-Projection vs. Post-Projection

- Radix-Decluster =
  cache-conscious post-projection





Cache-Conscious Joins

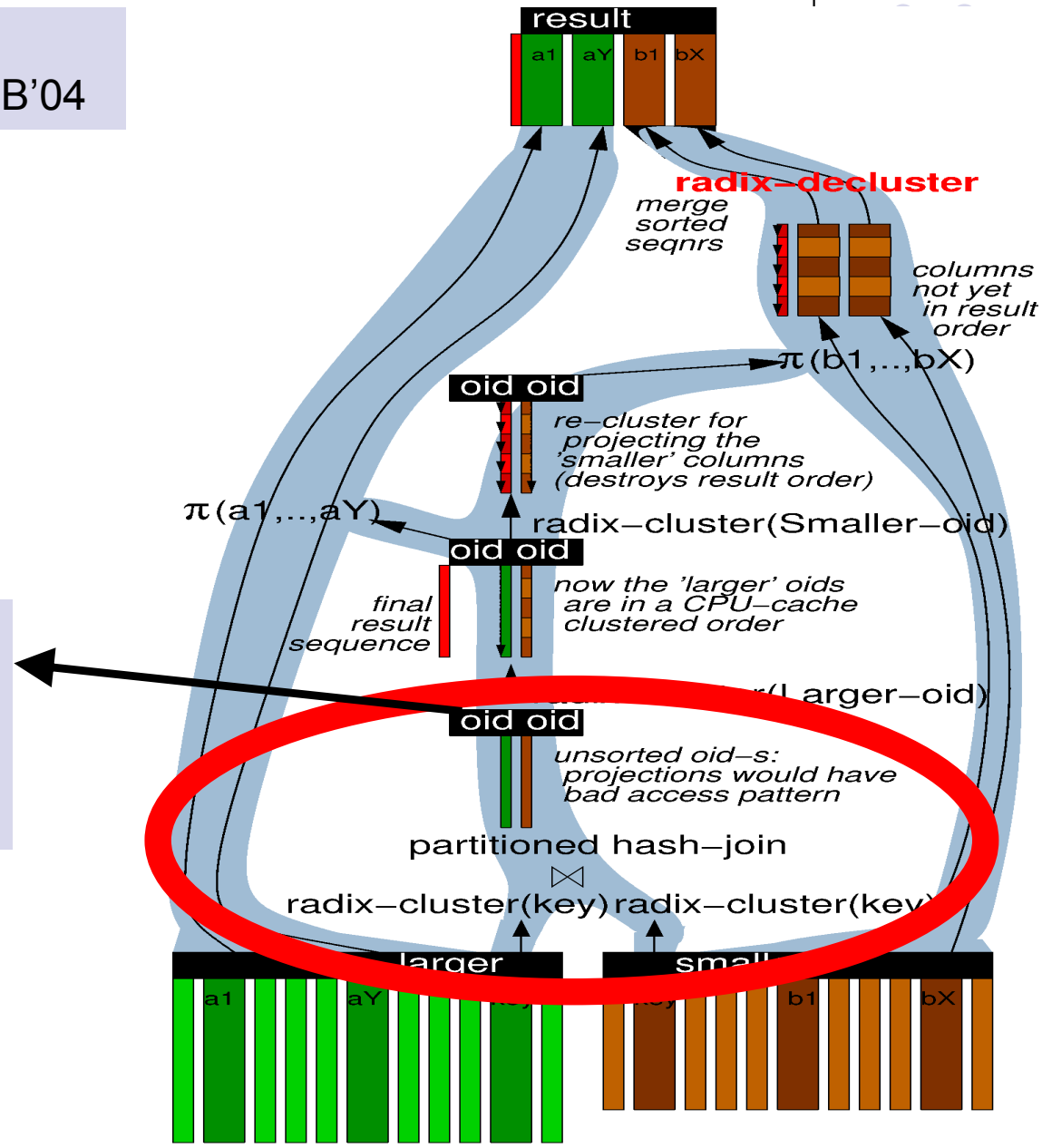
Post-Projection

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

- Partitioned Hash-Join

Join Index!

“Join Indices”
Valduriez, TODS’87

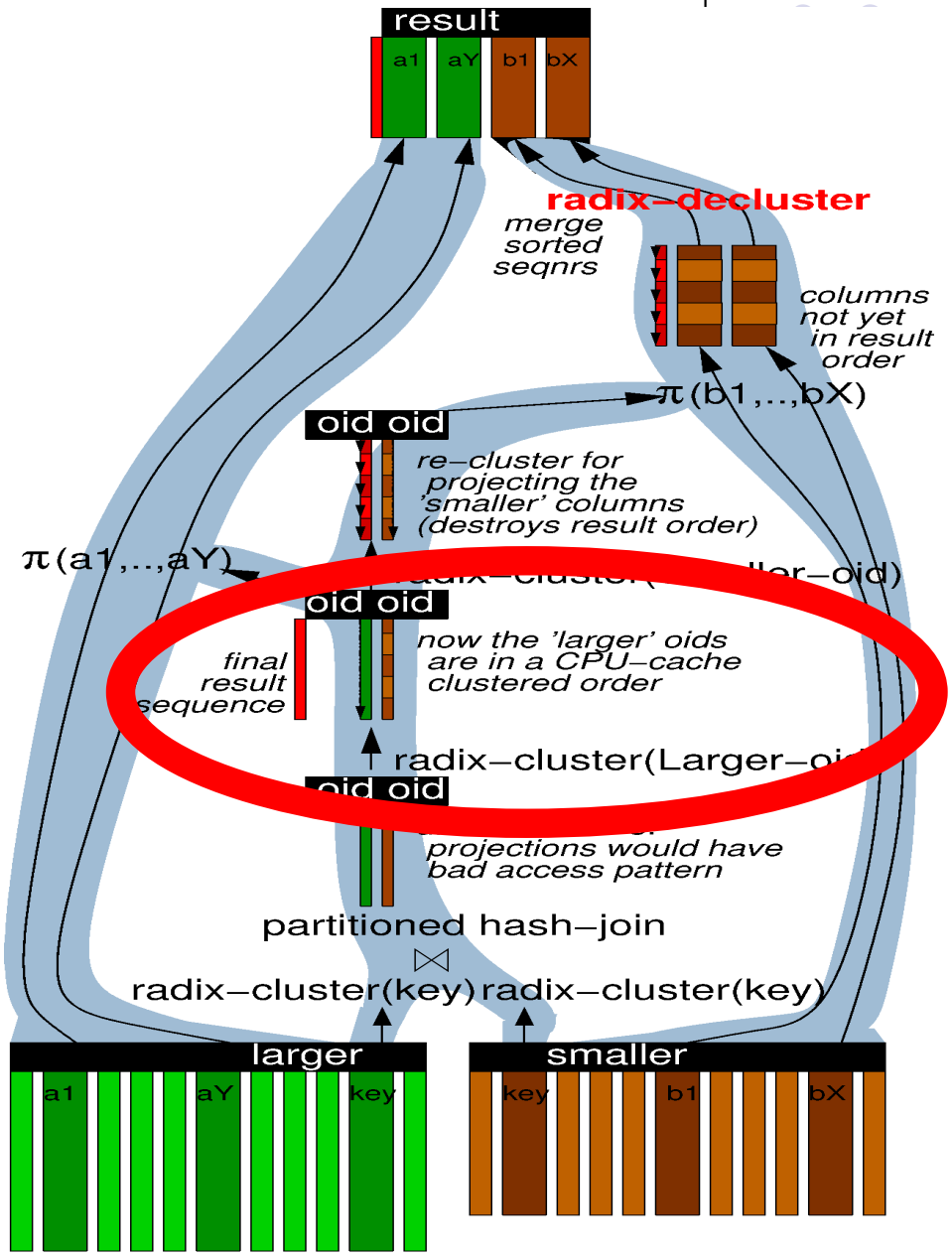


Cache-Conscious Joins

Post-Projection

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

- Partitioned Hash-Join
- Cluster on Left

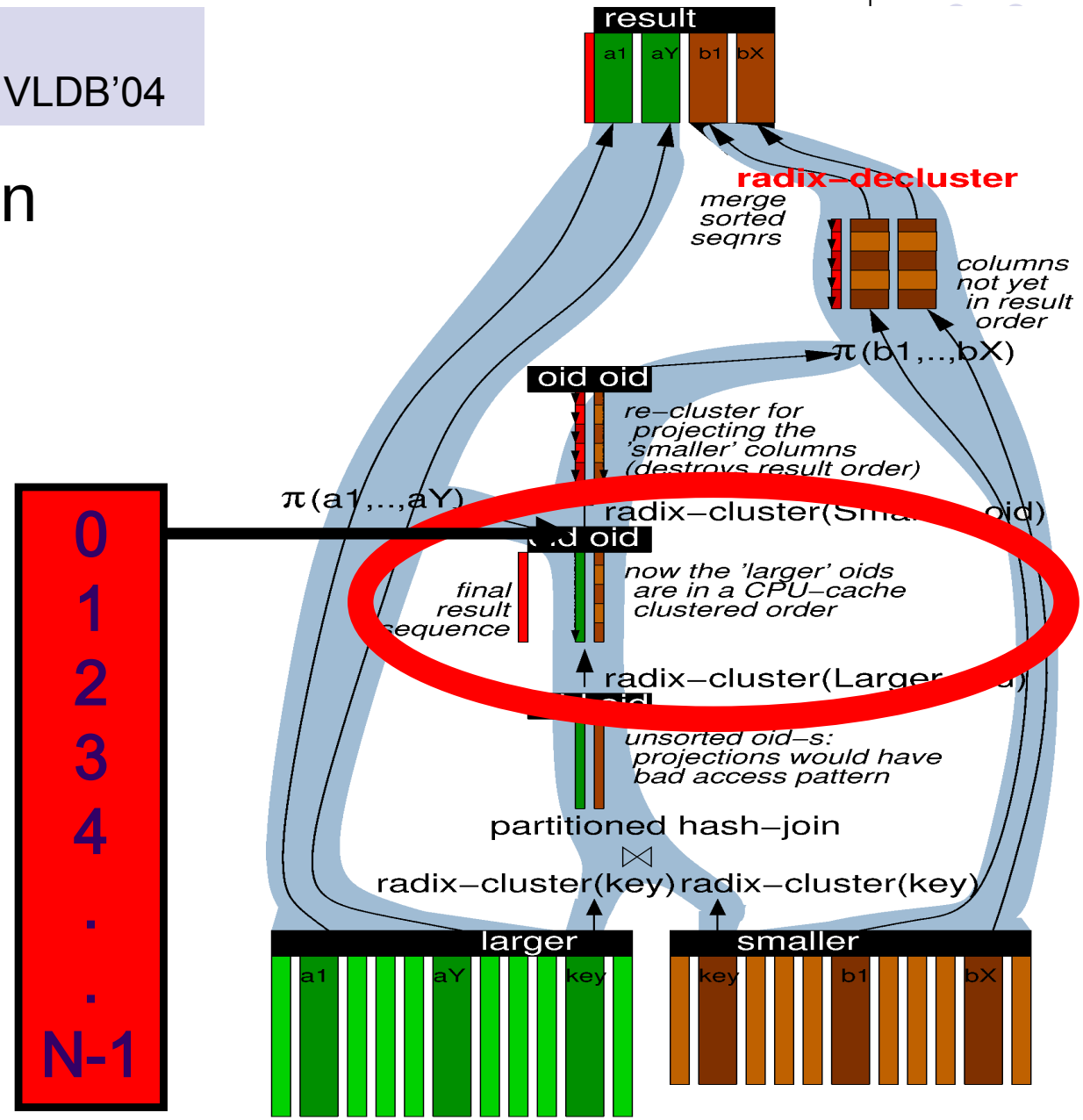


Cache-Conscious Joins

Post-Projection

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

- Partitioned Hash-Join
- Cluster on Left

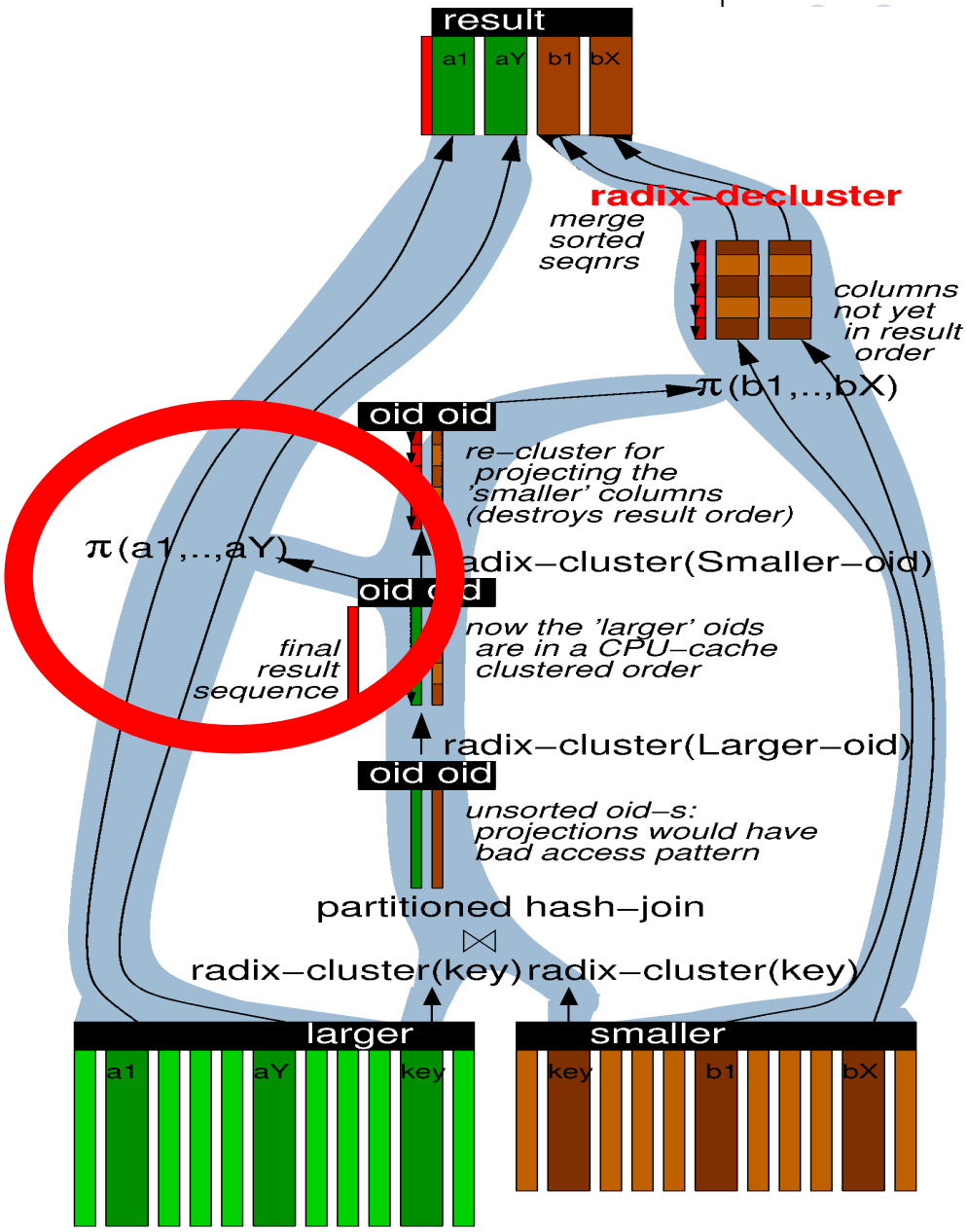


Cache-Conscious Joins

Post-Projection

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

- Partitioned Hash-Join
- Cluster on Left
- Project Left

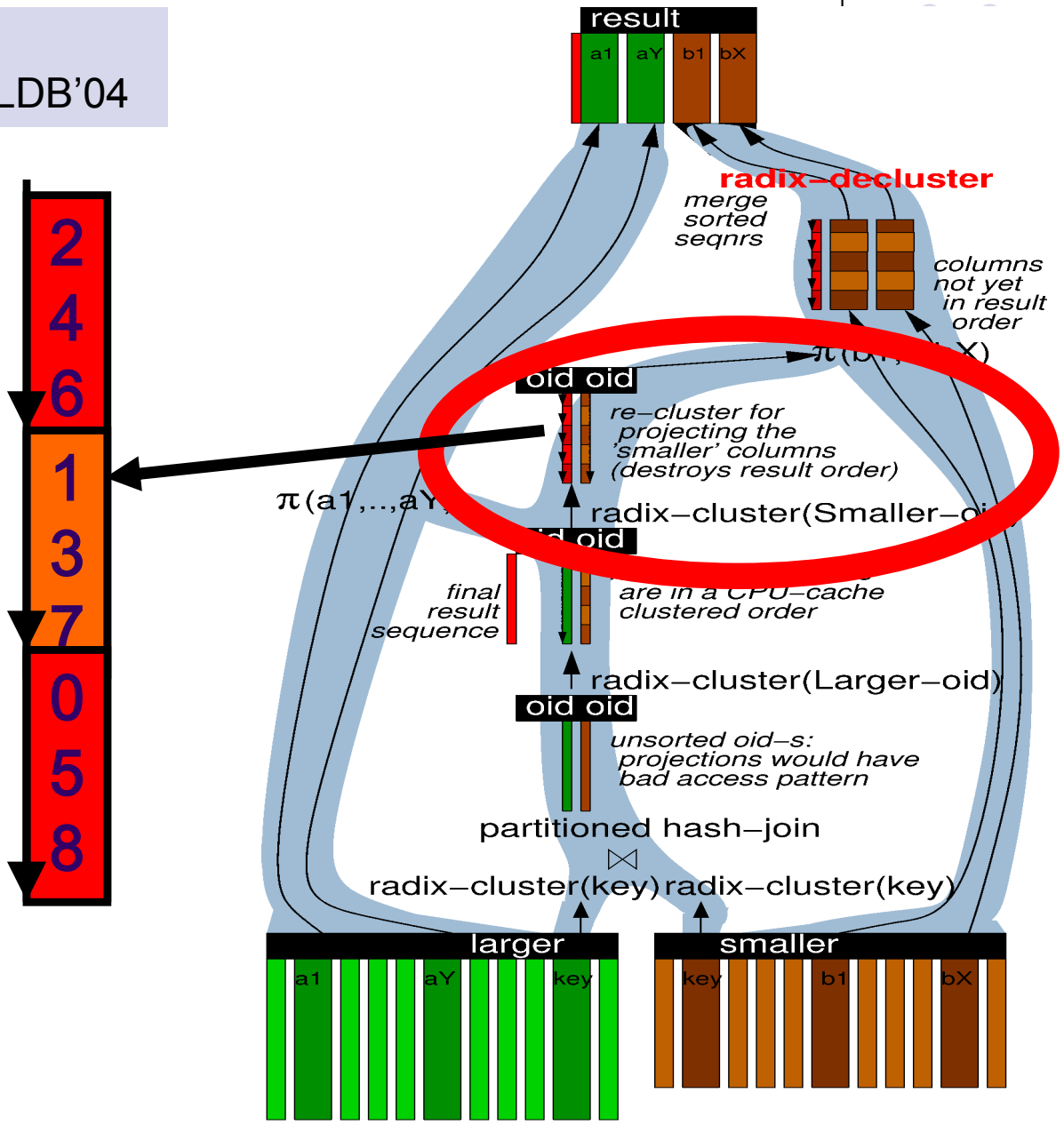


Cache-Conscious Joins

Post-Projection

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

- Partitioned Hash-Join
- Cluster on Left
- Project Left
- Cluster on Right

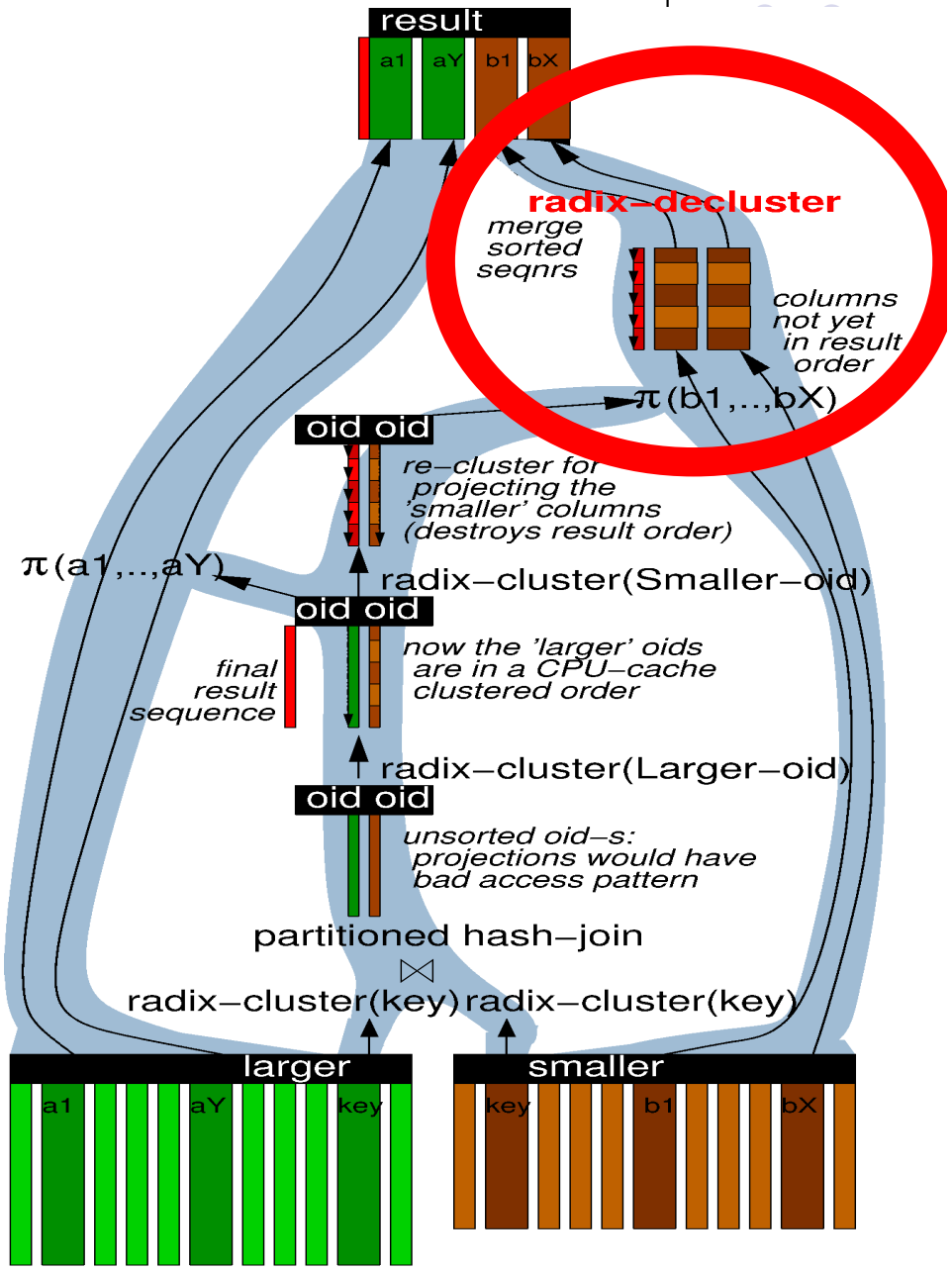


Cache-Conscious Joins

Post-Projection

•"Cache-Conscious Radix-Decluster Projections", Manegold, Boncz, Nes, VLDB'04

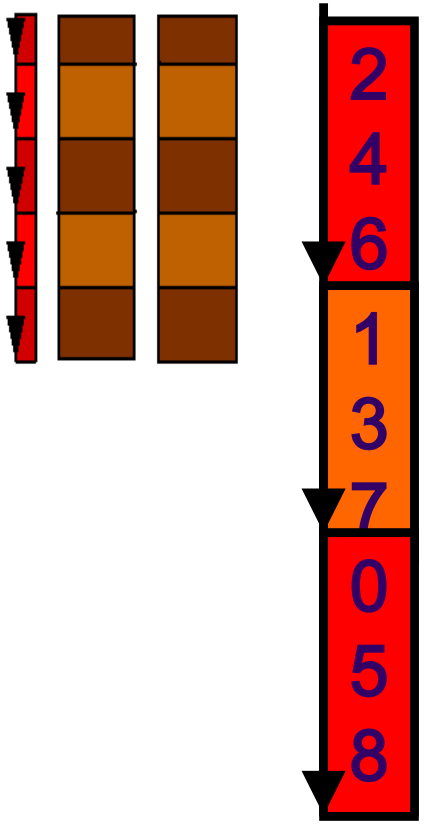
- Partitioned Hash-Join
- Cluster on Left
- Project Left
- Cluster on Right
- Project Right
  - ➔ Radix-Decluster



Cache-Conscious Joins

Radix-Decluster

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

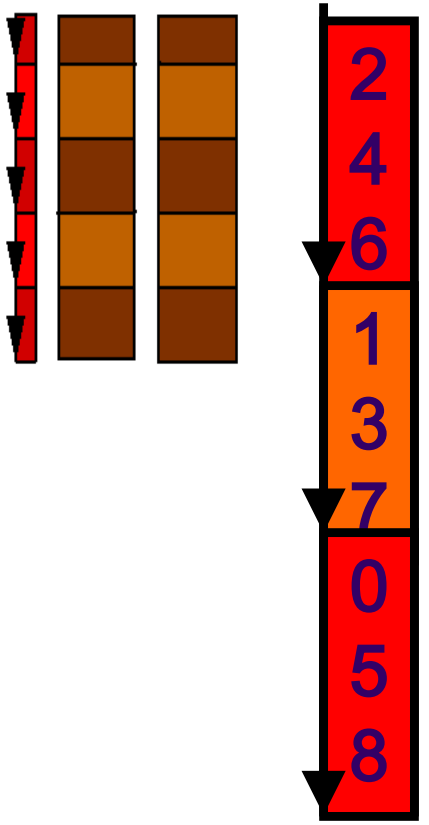


- Red column forms a dense domain
  - {0,1,2,…,N-1}
- All subsequences are ordered
  - e.g.
- Task: merge subsequences into dense sequence

Cache-Conscious Joins

Radix-Decluster

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

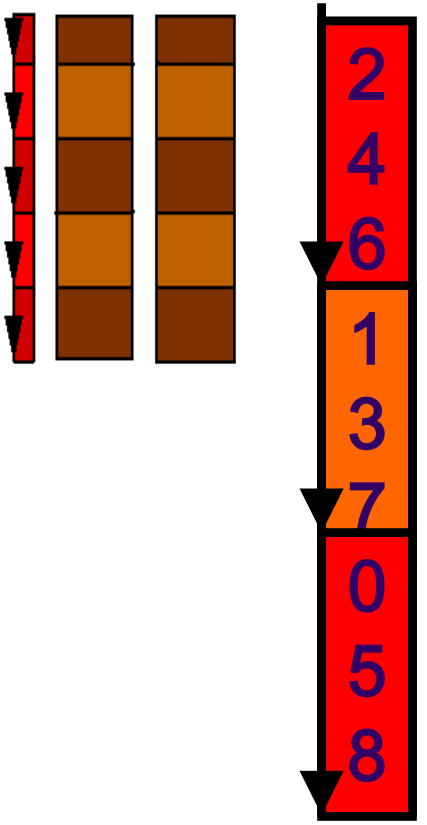


- Red column forms a dense domain
  - {0,1,2,…,N-1}
- All subsequences are ordered
  - e.g.
- Task: merge subsequences into dense sequence
  - Approach 1: merge $H=2^B$ lists
    - ☹ cost O(N log(H))
    - many (H) cursors needed ➔cache thrashing

Cache-Conscious Joins

Radix-Decluster

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

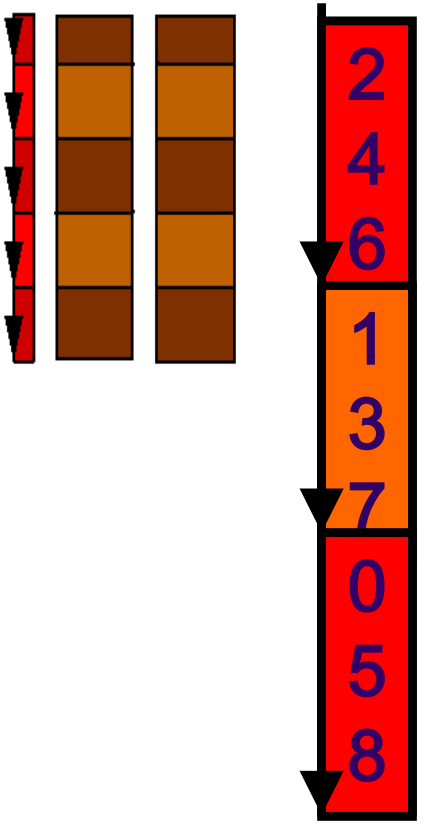


- Red column forms a dense domain
  - {0,1,2,…,N-1}
- All subsequences are ordered
  - e.g.
- Task: merge subsequences into dense sequence
  - Approach 1: merge $H=2^B$ lists
    - ☹ cost O(N log(H)),
    - many (H) cursors needed ➔cache thrashing
  - Approach 2: insert by position
    - ☹ many (H) sparse passes over the result ➔no cache reuse

Cache-Conscious Joins

Radix-Decluster

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



- Red column forms a dense domain
  - {0,1,2,…,N-1}
- All subsequences are ordered
  - e.g.
- Task: merge subsequences into dense sequence
  - Approach 1: merge $H=2^B$ lists
    - ☹ cost O(N log(H))
    - many (H) cursors needed ➔cache thrashing
  - Approach 2: insert by position
    - ☹ many (H) sparse passes over the result ➔ no cache reuse
  - Radix-Decluster: insert by position with sliding window

Cache-Conscious Joins

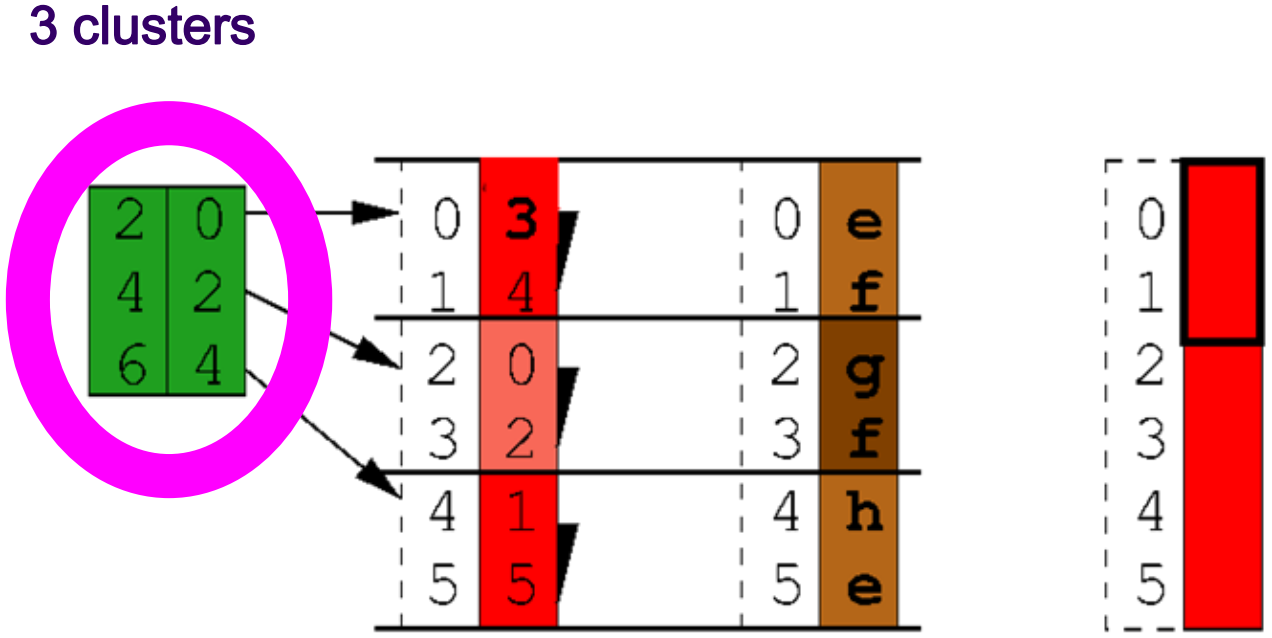
Radix-Decluster In Action

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

Cache-Conscious Joins

Radix-Decluster In Action

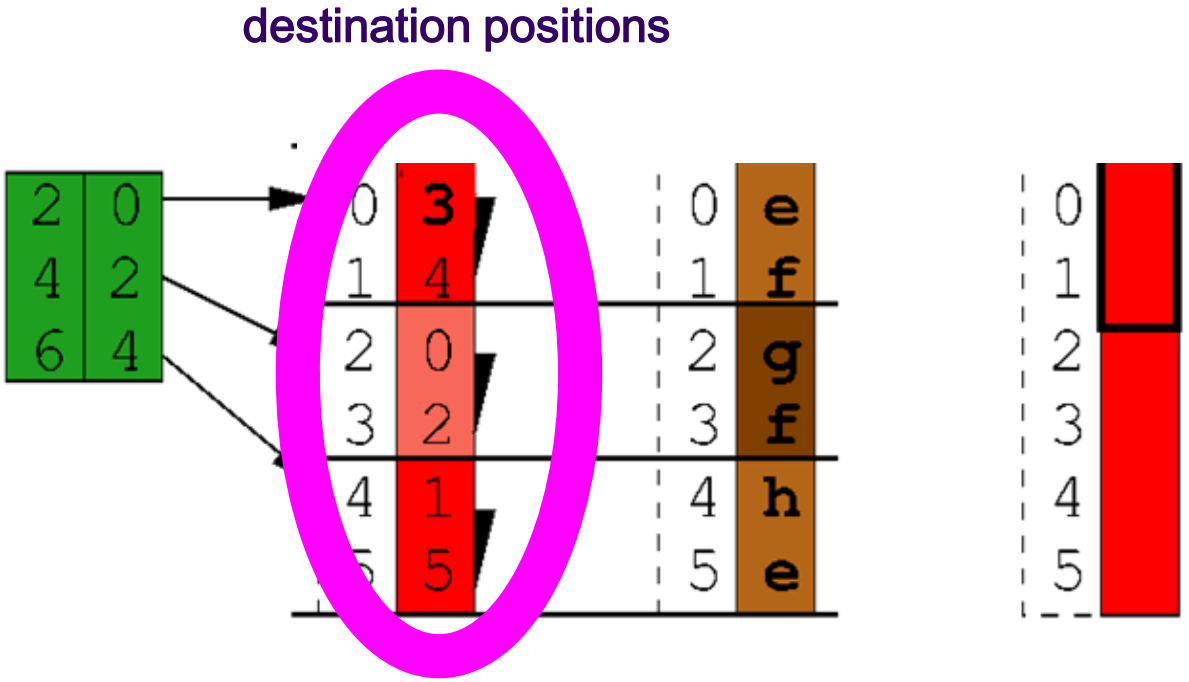
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

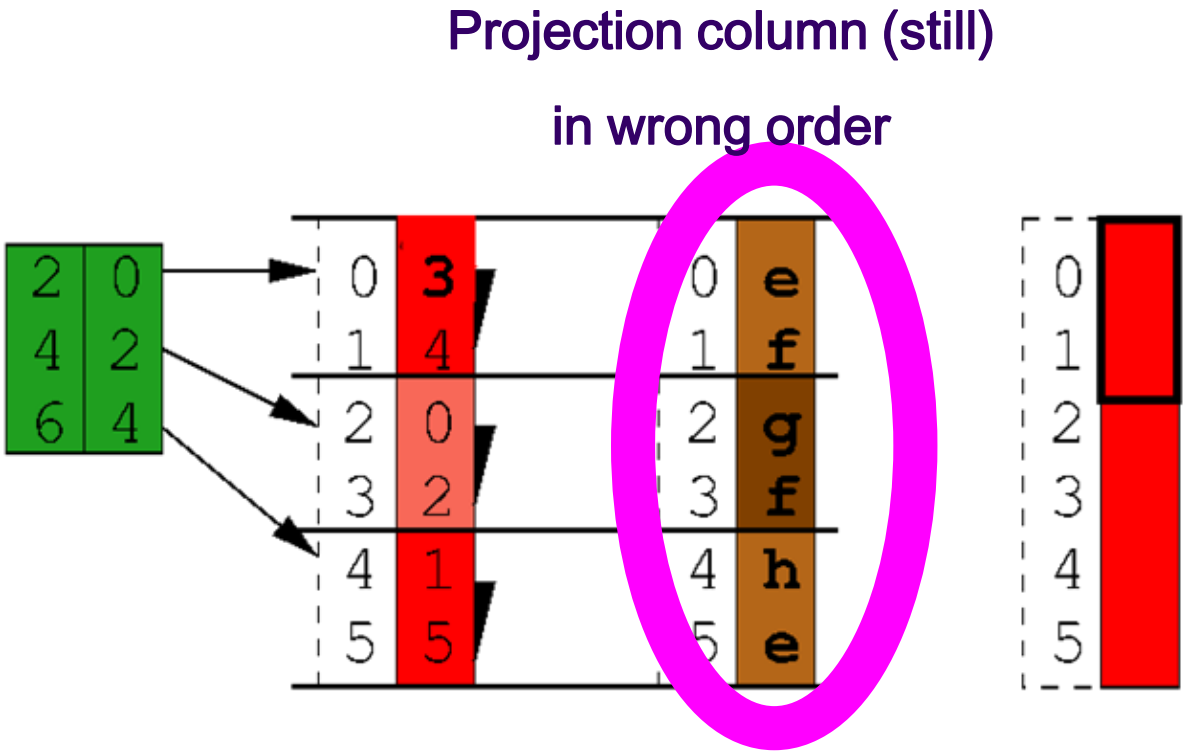
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

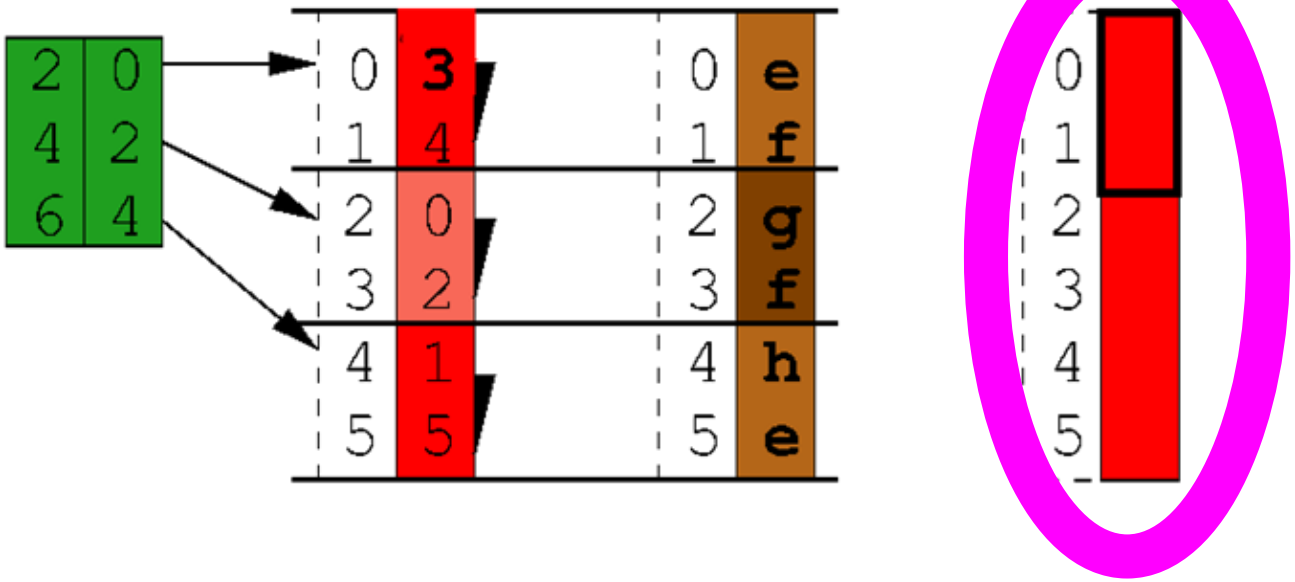
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

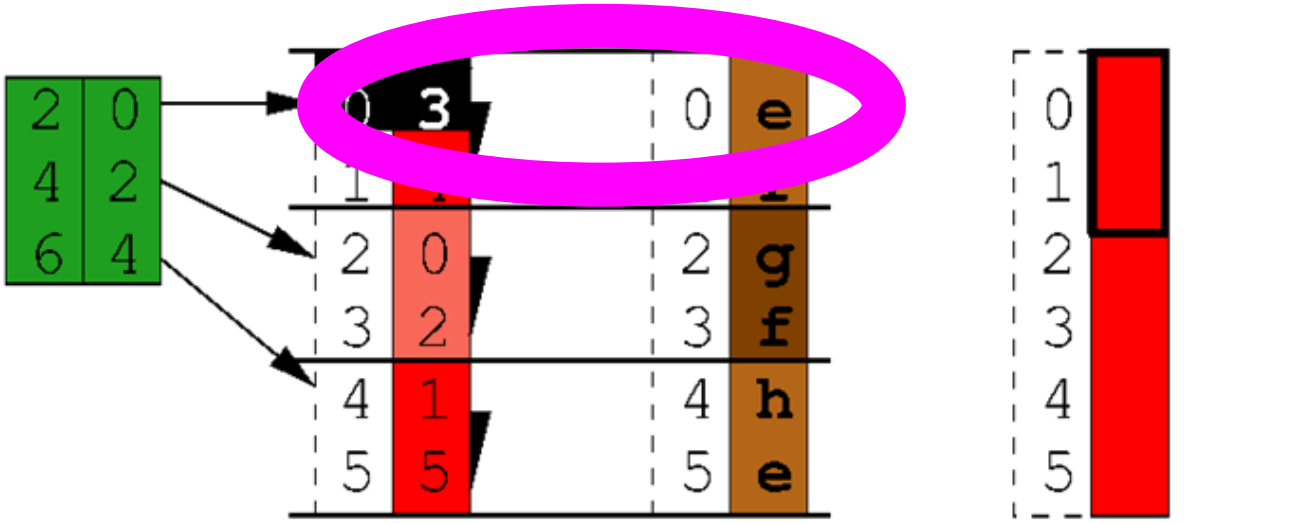
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

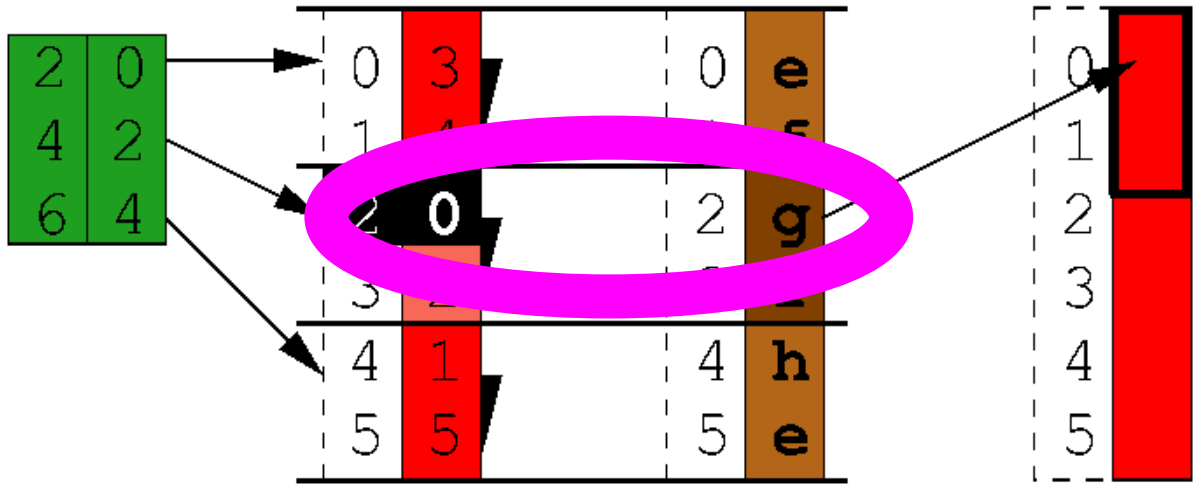


next cluster (3 is outside window)

Cache-Conscious Joins

Radix-Decluster In Action

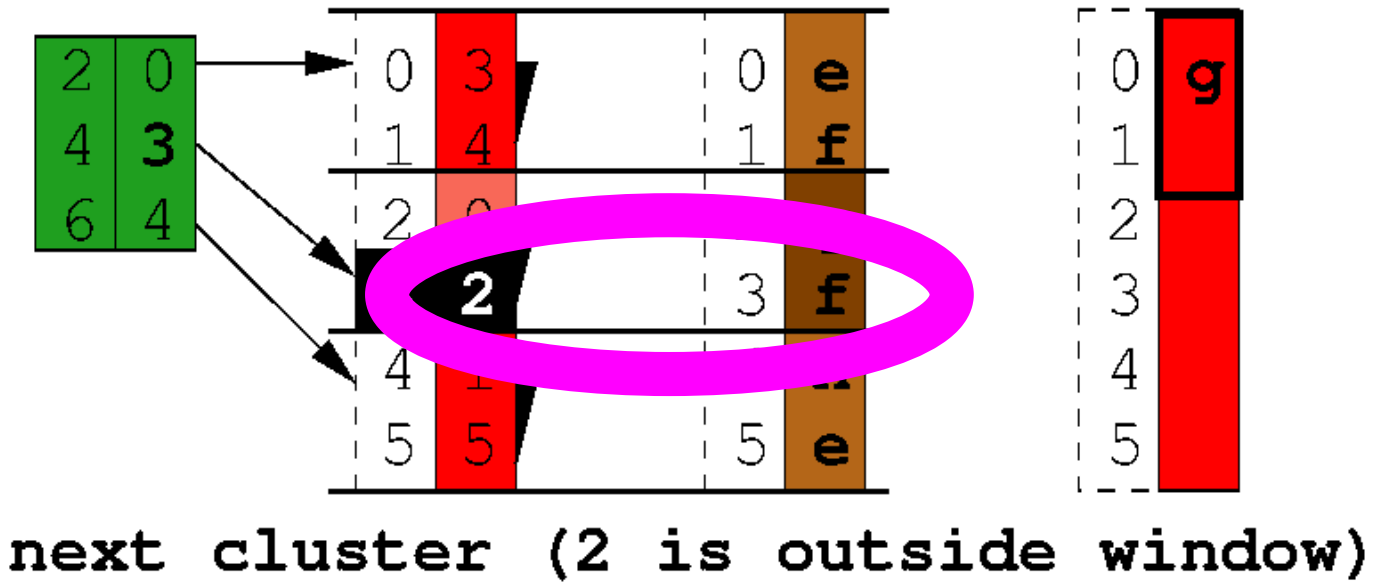
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

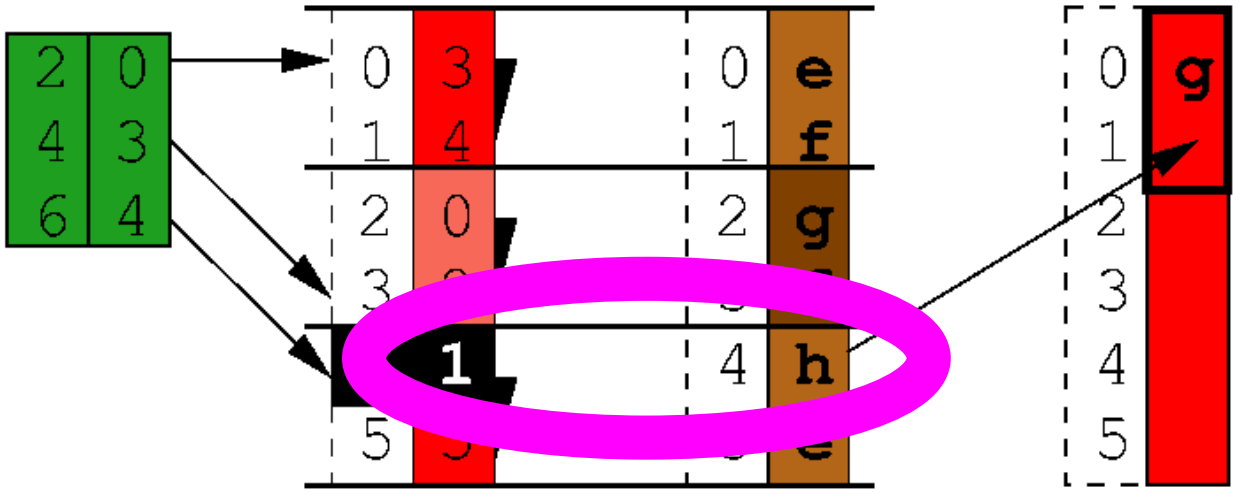
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

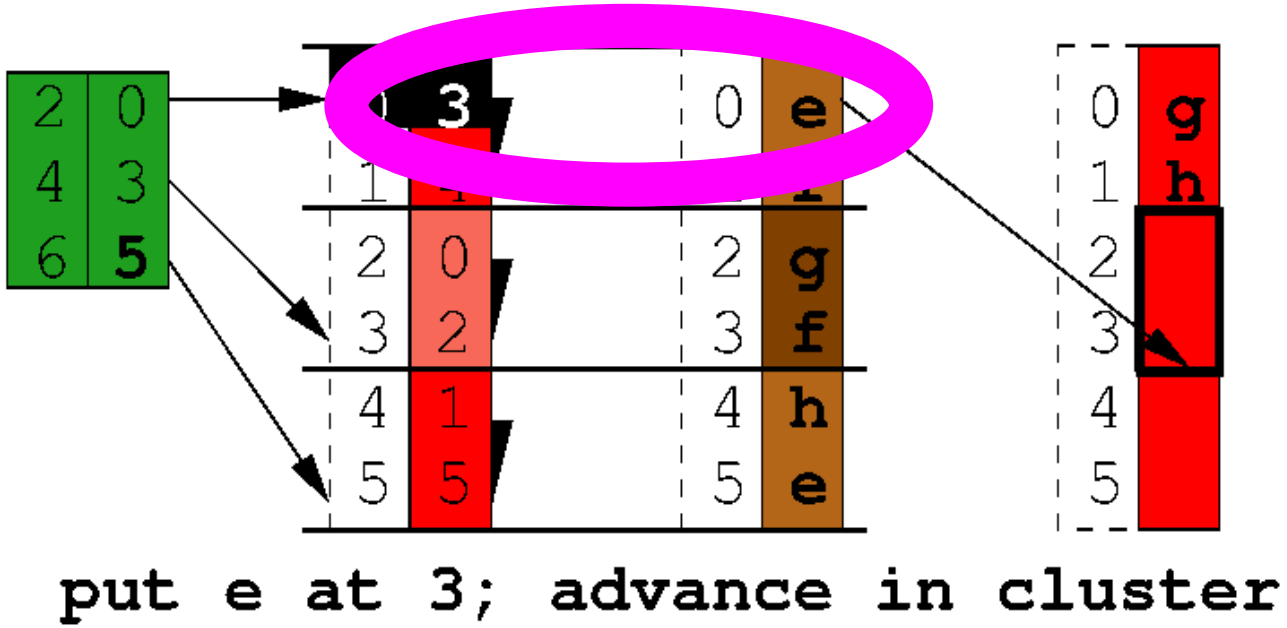


put h at 1; advance window; reset

Cache-Conscious Joins

Radix-Decluster In Action

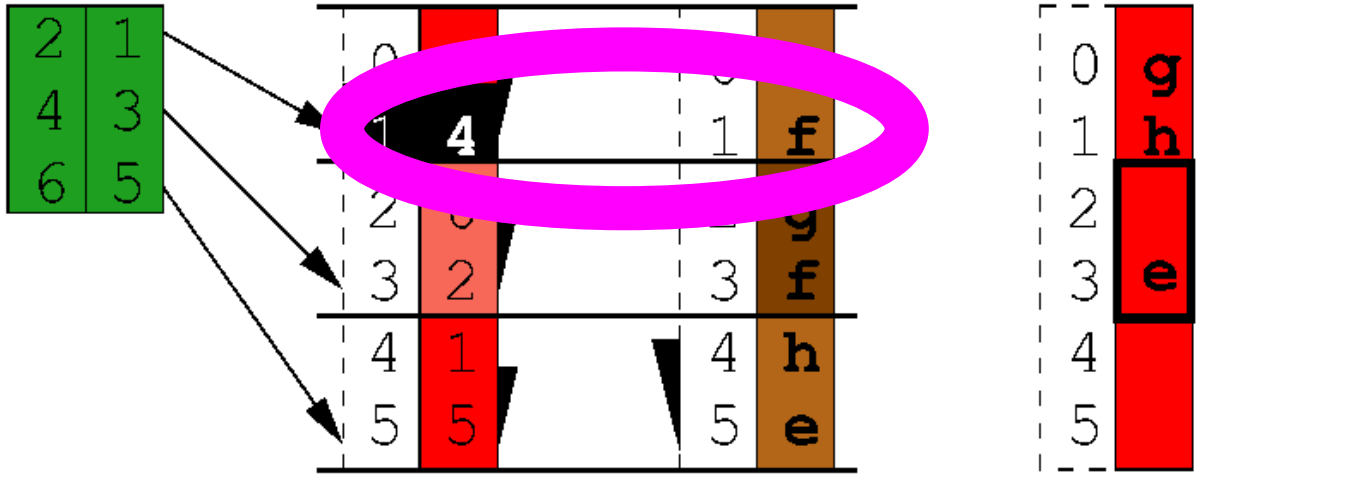
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

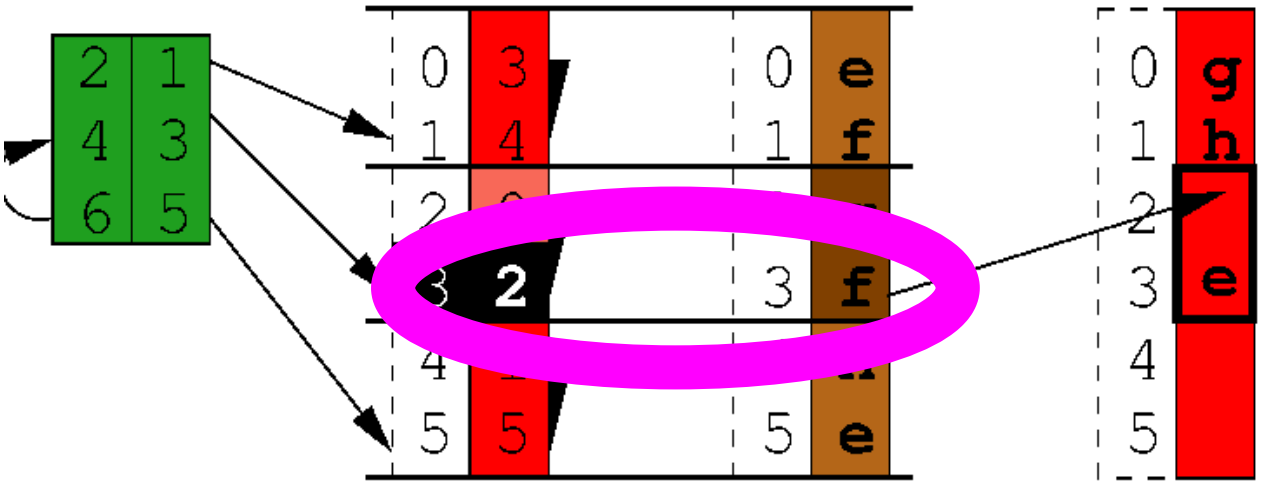
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

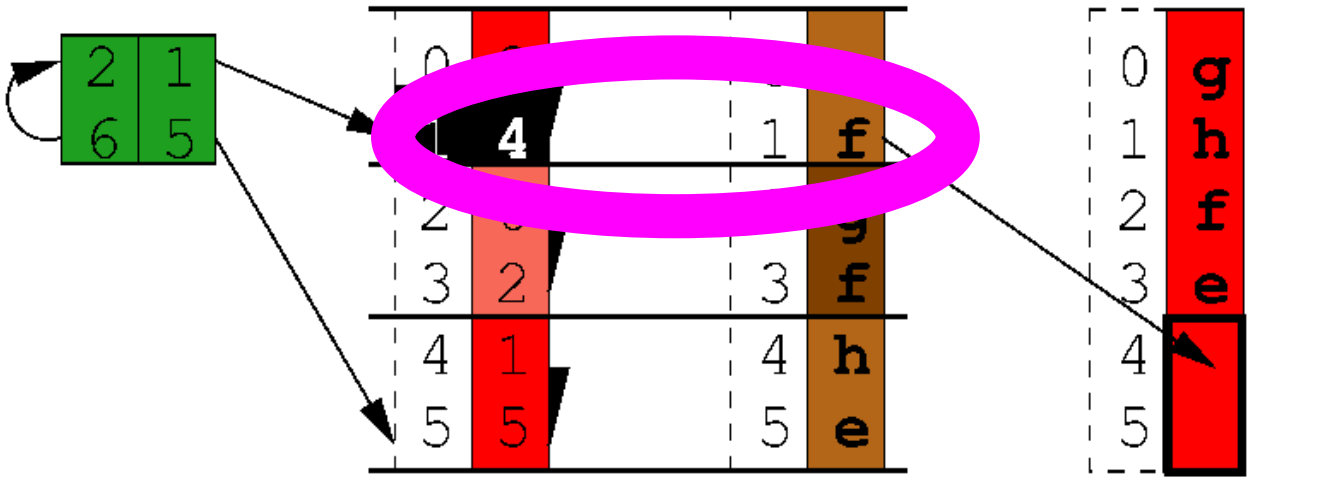


put f at 2; delete empty cluster;
advance window; reset

Cache-Conscious Joins

Radix-Decluster In Action

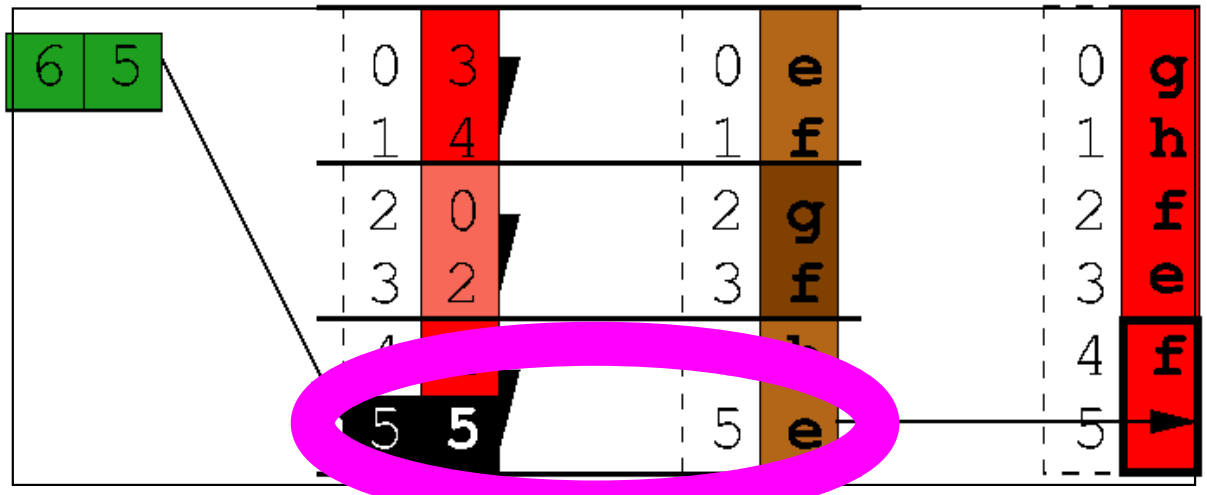
•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04



Cache-Conscious Joins

Radix-Decluster In Action

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

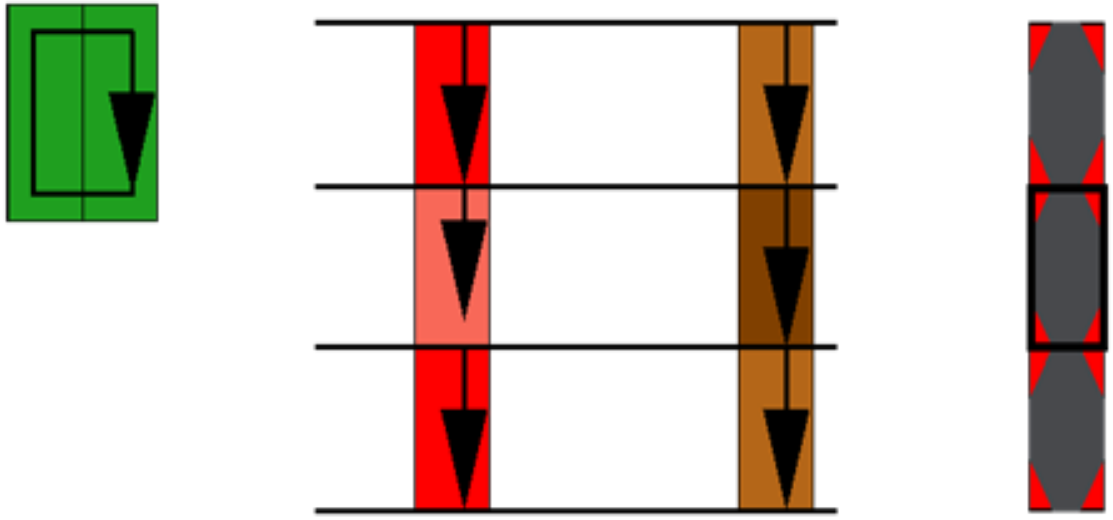


Cache-Conscious Joins

Radix-Decluster Memory Access Pattern

•“Cache-Conscious Radix-Decluster Projections”, Manegold, Boncz, Nes, VLDB’04

- Random access only to sliding window(<< cache size)



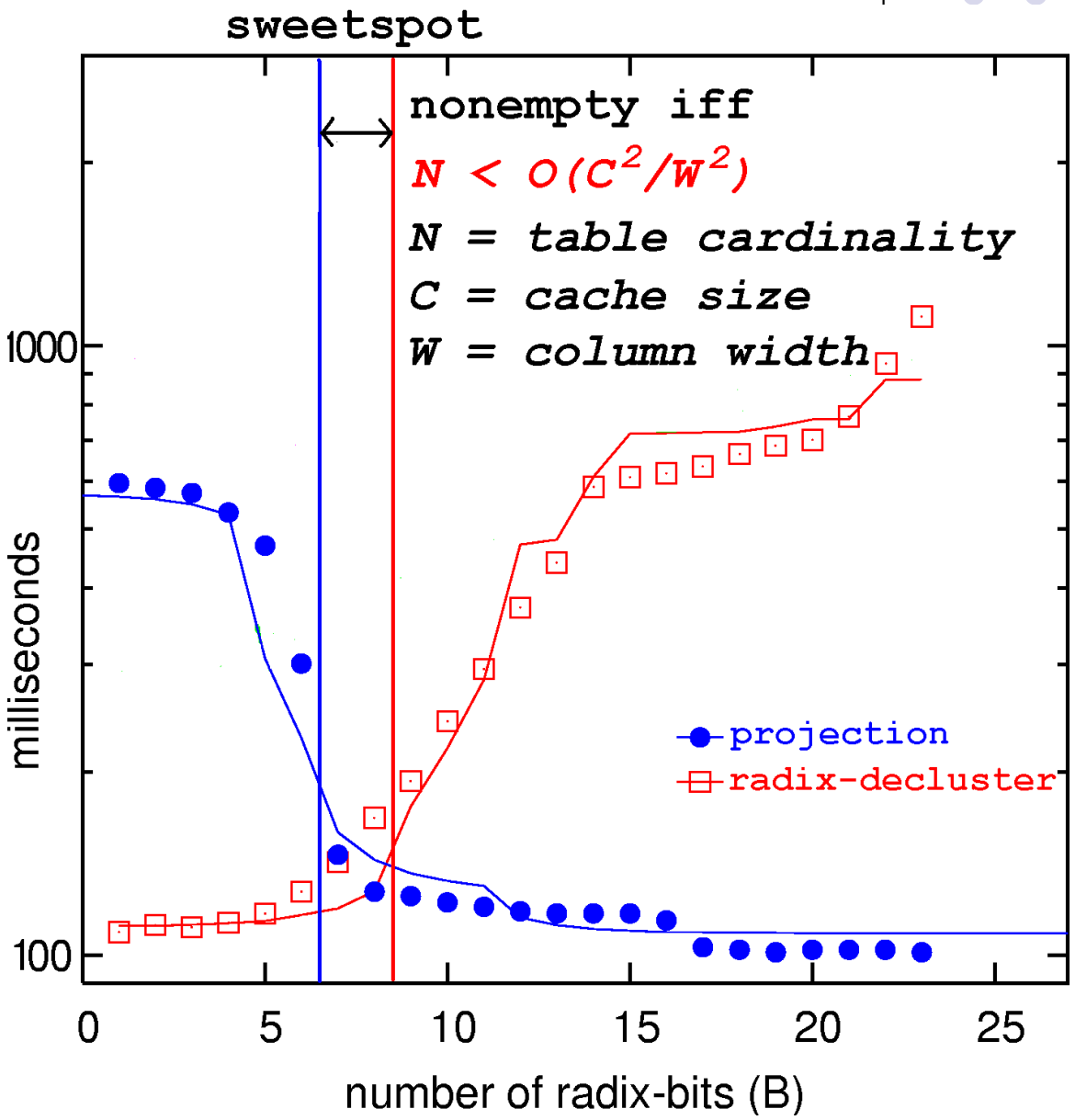
- Only compulsary misses ➔ cache-conscious

Cache-Conscious Joins

Radix-Decluster Performance Tradeoff

- Radix-Decluster prefers small W

  (i.e. vertical fragmentation)



Read Also:
- Jive Join
- Slam Join

"Fast Joins Using Join Indices"
Ross, Lei, VLDBJ'98

ADM: Literature

- **Column-Oriented Database Systems (3/6) - Cache Conscious Joins**
  - “Cache Conscious Algorithms for Relational Query Processing”. Shatdal, Kant, Naughton. VLDB’94.
  - “Fast Joins using Join Indices”. Li and Ross. VLDBJ 8:1-24, 1999.
  - “Optimizing main-memory join on modern hardware”. Boncz, Manegold, Kersten, TKDE 14(4): 709-730, 2002.
    - “Database Architecture Optimized for the New Bottleneck: Memory Access”. Boncz, Manegold, Kersten. VLDB’99.
    - “What Happens During a Join? Dissecting CPU and Memory Optimization Effects”. Manegold, Boncz, Kersten. VLDB’00.
    - “Optimizing database architecture for the new bottleneck: memory access”. Manegold, Boncz, Kersten. VLDB J. 9(3): 231-246, 2000.
    - “Generic Database Cost Models for Hierarchical Memory Systems”. Manegold, Boncz, Kersten. VLDB’02.
  - “Cache-Conscious Radix-Decluster Projections”. Manegold, Boncz, Nes. VLDB’04.

ADM: Agenda

- 07.09.2022: Lecture 1: **Introduction**
- 14.09.2022: Lecture 2: **SQL Recap**

  (plus Assignment 1 [in groups; 3 weeks]: TPC-H benchmark)
- 21.09.2022: Lecture 3: **Column-Oriented Database Systems (1/6) - Motivation & Basic Concepts**
- 28.09.2022: Lecture 4: **Column-Oriented Database Systems (2a/6) - Selected Execution Techniques (1/2)**
- 05.10.2022: Lecture 5: **Column-Oriented Database Systems (2b/6) - Selected Execution Techniques (2/2)**

  (plus Assignment 2 [in groups; 4 weeks]: Compression techniques)
- 12.10.2022: Lecture 6: **Column-Oriented Database Systems (3/6) - Cache Conscious Joins**
- 19.10.2022: Lecture 7: **Column-Oriented Database Systems (4/6) - "Vectorized Execution"**
- ~~26.10.2022~~: ***No lecture!***
- 02.11.2022: Lecture 8: **DuckDB: An embedded database for data science (1/2) (guest lecture & *hands-on*)**

  (plus Assignment 3 [individual; 2 weeks]: Analysing NYC Cab dataset with DuckDB)
- 09.11.2022: Lecture 9: **DuckDB: An embedded database for data science (2/2) (guest lecture & *hands-on*)**
- 16.11.2022: Lecture 10: **Branch Misprediction & Predication**

  (plus Assignment 4 [individual; 2 weeks]: Predication)
- 23.11.2022: Lecture 11: **Column-Oriented Database Systems (5/6) - Adaptive Indexing**
- 30.11.2022: Lecture 12: **Column-Oriented Database Systems (6/6) - Progressive Indexing**

ADM: Literature

- **Column-Oriented Database Systems (4/6) - “Vectorized Execution”**

  - “MonetDB/X100: Hyper-Pipelining Query Execution”. Boncz, Zukowski, Nes. CIDR’05.

  - “Buffering Database Operations for Enhanced Instruction Cache Performance”. Zhou and Ross. SIGMOD’04.

  - “Block oriented processing of relational database operations in modern computer architectures”. Padmanabhan, Malkemus, Agarwal. ICDE’01.

  - “Balancing Vectorized Query Execution with Bandwidth Optimized Storage”. Zukowski. PhD Thesis. CWI 2008.