CIS 560 – Database System Concepts

Lecture 27

Operator Algorithms

November 6, 2013

Credits for slides: Chang, Ullman, Whitehead.

Outline

- Query execution 15.1-15.6
- Query optimization 16

2

Planning

- Assignment 8 (indexes) due 11/8
- Project - DB design revision due 11/11
  - No class that day – use the time to work on project
- Assignment 9 (query optimization) due 11/15
- Exam 2 (assignments 6-9) – 11/20
- Project - DB implementation and queries due 11/22
- Quiz from special topics – 12/06
- Project presentations 12/9, 12/11, 12/13
- Project reports – finals week

3

Types of Algorithms

- One-pass algorithms (Sec. 15.2 and 15.3)
- Index-based algorithms (Sec 15.6)
- Two-pass algorithms (Sec 15.4 and 15.5)

Hash Join

Hash join: $R \bowtie S$

- Scan R, build buckets in main memory
- Then scan S and join
- Cost: B(R) + B(S)

- One-pass algorithm when $B(R) \leq M$

Sort-Merge Join

- Sort-merge join: $R \bowtie S$
  - Scan R and sort in main memory
  - Scan S and sort in main memory
  - Merge R and S

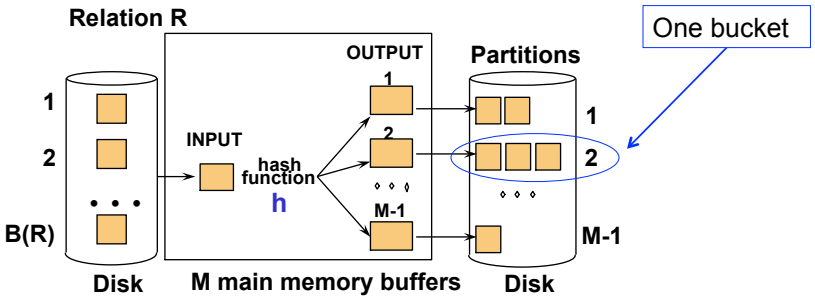
- Cost: B(R) + B(S)
- One pass algorithm when $B(S) + B(R) \leq M$
- Typically, this is NOT a one pass algorithm

Two-Pass Algorithms

- What if data does not fit in memory?
- Need to process it in multiple passes
- Two key techniques
  1. Hashing
  2. Sorting

Partitioned Hash Algorithms

- Idea: partition a relation R into buckets, on disk
- Each bucket has size ≈ B(R)/M



Does each bucket fit in main memory?
Yes, if B(R)/M <= M, i.e. B(R) <= M^2

Partitioned Hash Join

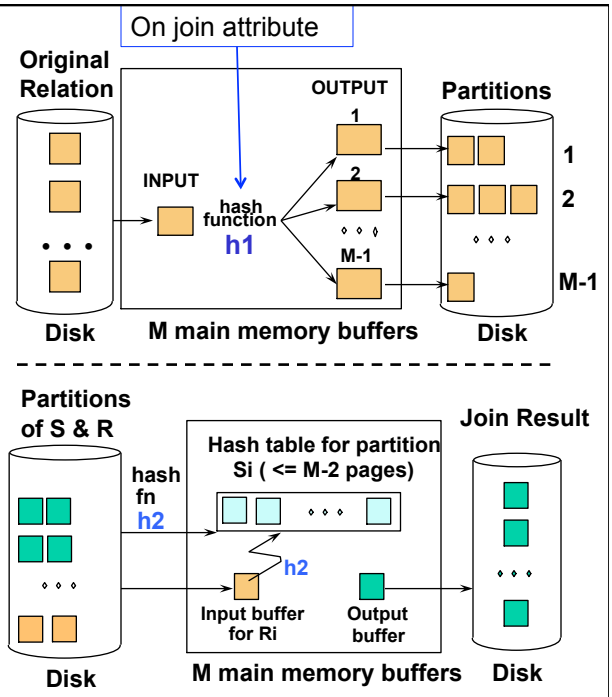
R ⋈ S

- Step 1:
  - Hash S into M-1 buckets
  - Send all buckets to disk
- Step 2
  - Hash R into M-1 buckets
  - Send all buckets to disk
- Step 3
  - Join every pair of buckets

Cost: 3B(R) + 3B(S)
Assumption: $\min(B(R),B(S)) \leq M^2$

Partitioned Hash-Join

- Partition both relations using hash function **h1**: R tuples in partition i will only match S tuples in partition i.
- Read in a partition of S, hash it using **h2 (<> h1)**. Scan matching partition of R, search for matches.



Cost of Partitioned Hash Join

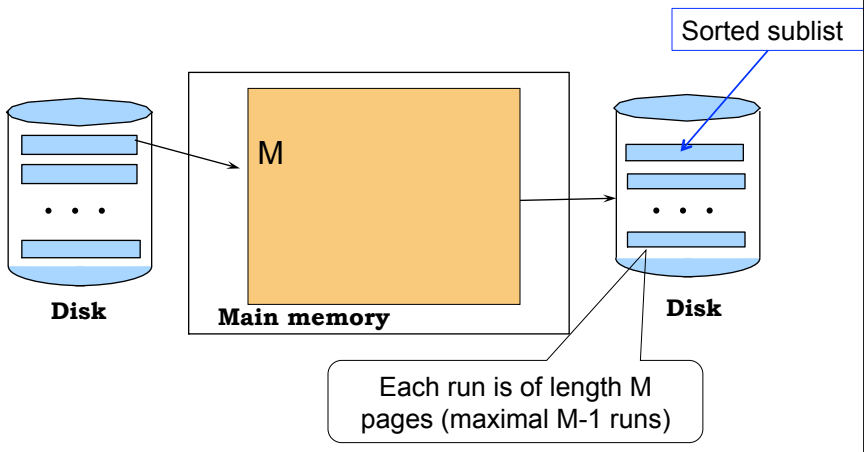
- Read+write+read = 3B(R) + 3B(S)
- Assumption: min(B(R),B(S)) <= M^2

External Sorting (Two-Phase, Multiway Merge Sort)

- Problem:
  - Sort a file of size B with memory M
- Where we need this:
  - ORDER BY in SQL queries
  - Several physical operators
  - Bulk loading of B+-tree indexes.
- Sorting is two-pass when $B \leq M^2$

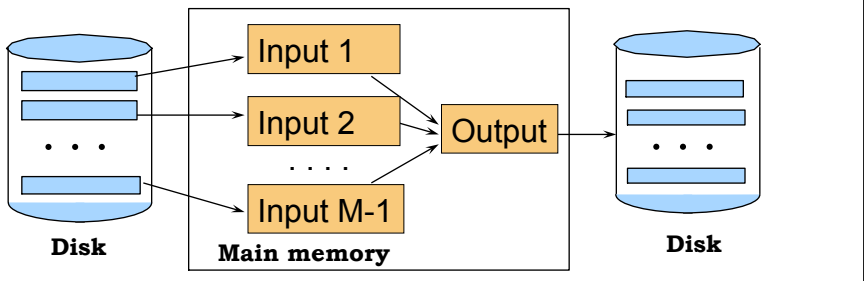
External Merge-Sort: Step 1

- Load M pages in memory, sort



External Merge-Sort: Step 2

- Merge M – 1 runs (sorted sublists) into a new run (sorted list)
  - M-1 runs of length M: $M(M-1) \approx M^2$ => $B <= M^2$



Cost of External Merge Sort

- Read+write+read = 3B(R)
- Assumption: B(R) <= M^2
- Other considerations
  - In general, a lot of optimizations are possible.

Two-Pass Join Algorithm Based on Sorting

Join R ⋈ S

- Step1: Sort both R and S on the join attribute:
  - Cost: 4B(R)+4B(S) (because need to write to disk)
- Step2: Read both relations in sorted order, match tuples
  - Cost: B(R)+B(S)
- Total cost: 5B(R)+5B(S)
- Assumption: B(R)<=M^2,B(S)<=M^2
- All tuples with a common value for the join attribute must fit in the memory.

Two-Pass Algorithms Based on Sorting

Join R ⋈ S

- If the number of tuples in R matching those in S is small (or vice versa), we can compute the join during the second phase of sort (sort-merge-join)
- Total cost: 3B(R)+3B(S)
- Assumption: B(R) + B(S) <= M^2

Summary of Join Algorithms

- Nested Loop Join: B(R)+B(R)B(S)/M
  - Assuming block-at-a-time refinement, with one block-at-a time, the cost is: B(R)+ B(R)B(S)
- Hash Join: 3B(R) + 3B(S)
  - Assuming: min(B(R), B(S)) ≤ M^2
- Sort-Merge Join: 3B(R) + 3B(S)
  - Assuming B(R)+B(S) ≤ M^2
- Index Nested Loop Join: B(R) + T(R)B(S)/V(S,a)
  - Assuming S has clustered index on attribute a

Exercise

Consider joining two relations R(x, y) and S(x, z) on their common attribute x. The size of relation R is 1000 blocks and the size of relation S is 500 blocks. Attribute x of relation R has 50 different values and the values are evenly distributed in R. Attribute x of relation S also has the same 50 different values and the values are evenly distributed in S. Suppose that neither relation is sorted by attribute x. Furthermore, suppose that the memory buffer has 101 blocks.

- Briefly explain if/how the two relations R and S can be joined using a two-pass sort-merge join algorithm and compute the cost of join.
- Similarly, explain how the two relations R and S can be joined using a two-pass partitioned hash-join algorithm and compute the cost of join.