Parallel DBMS

Chapter 22, Part A

Slides by Joe Hellerstein, UCB, with some material from Jim Gray, Microsoft Research.
Why Parallel Access To Data?

At 10 MB/s
1 Terabyte
1.2 days to scan

1,000 x parallel
1 Terabyte
1.5 minute to scan.

Parallelism:
divide a big problem into many smaller ones to be solved in parallel.

Bandwidth:
10 MB/s
Parallel DBMS: Intro

- Parallelism is natural to DBMS processing
  - Pipeline parallelism: many machines each doing one step in a multi-step process.
  - Partition parallelism: many machines doing the same thing to different pieces of data.
  - Both are natural in DBMS!

Pipeline

Partition
Some Terminology

- **Speed-Up**
  - More resources means proportionally less time for given amount of data.

- **Scale-Up**
  - If resources increased in proportion to increase in data size, time is constant.
Architecture Issue: Shared What?

Shared Memory (SMP)
- Easy to program
- Expensive to build
- Difficult to scale up
- Sequent, SGI, Sun

Shared Disk
- Hard to program
- Cheap to build
- Easy to scale up
- VMScluster, Sysplex

Shared Nothing (network)
- Easy to program
- Cheap to build
- Easy to scale up
- Tandem, Teradata, SP2
Different Types of DBMS

- Intra-operator parallelism
  - get all machines working to compute a given operation (scan, sort, join)
- Inter-operator parallelism
  - each operator may run concurrently on a different site (exploits pipelining)
- We’ll focus on intra-operator parallelism
Automatic Data Partitioning

Partitioning a table:
- **Range**
- **Hash**
- **Round Robin**

Good for equijoins, range queries, and group-by

Good for equijoins

Good to spread load

Shared disk and memory less sensitive to partitioning, Shared nothing benefits from "good" partitioning
Parallel Scans

- Scan in parallel, and merge.
- Selection may not require all sites for range or hash partitioning.
- Indexes can be built at each partition.
- Question: How do indexes differ in the different schemes?
  - Think about both lookups and inserts!
  - What about unique indexes?
Parallel Sorting

- Idea:
  - Scan in parallel, and range-partition as you go.
  - As tuples come in, begin “local” sorting on each
  - Resulting data is sorted, and range-partitioned.
  - Problem: *skew!*
  - Solution: “sample” the data at start to determine partition points.
Parallel Joins

- **Nested loop:**
  - Each outer tuple must be compared with each inner tuple that might join.
  - Easy for range partitioning on join cols, hard otherwise!

- **Sort-Merge (or plain Merge-Join):**
  - Sorting gives range-partitioning.
    - But what about handling skews?
  - Merging partitioned tables is local.
In first phase, partitions get distributed to different sites:
- A good hash function \textit{automatically} distributes work evenly!

Do second phase at each site.

Almost always the winner for equi-join.
Dataflow Network for \( \Join \)
Complex Parallel Query Plans

- Complex Queries: Inter-Operator parallelism
  - Pipelining between operators:
    - note that sort and phase 1 of hash-join block the pipeline!!
  - Bushy Trees
Parallel Query Optimization

- Common approach: 2 phases
  - Pick best sequential plan (System R algorithm)
  - Pick degree of parallelism based on current system parameters.
- “Bind” operators to processors
  - Take query tree, “decorate” as in previous picture.
What’s Wrong With That?

- Best serial plan != Best | | plan! Why?
- Trivial counter-example:
  - Table partitioned with local secondary index at two nodes
  - Range query: all of node 1 and 1% of node 2.
  - Node 1 should do a scan of its partition.
  - Node 2 should use secondary index.

- SELECT *
  FROM telephone_book
  WHERE name < “NoGood”;

Table Scan

Index Scan

A..M

N..Z
Parallel DBMS Summary

- |-ism natural to query processing:
  - Both pipeline and partition |-ism!

- Shared-Nothing vs. Shared-Mem
  - Shared-disk too, but less standard
  - Shared-mem easy, costly. Doesn’t scaleup.
  - Shared-nothing cheap, scales well, harder to implement.

- Intra-op, Inter-op, & Inter-query |-ism all possible.
- Data layout choices important!
- Most DB operations can be done partition-
  - Sort.
  - Sort-merge join, hash-join.
- Complex plans.
  - Allow for pipeline- | | ism, but sorts, hashes block the pipeline.
  - Partition | | -ism achieved via bushy trees.
DBMS Summary, cont.

- Hardest part of the equation: optimization.
  - 2-phase optimization simplest, but can be ineffective.
  - More complex schemes still at the research stage.

- We haven’t said anything about Xacts, logging.
  - Easy in shared-memory architecture.
  - Takes some care in shared-nothing.