Parallel Database Systems

The Future of High Performance Database Systems

Authors: David DeWitt and Jim Gray

Presenter: Jennifer Tom

Based on Presentation by Ajith Karimpana

Outline

- Why Parallel Databases?
- Parallel Database Implementation
- Parallel Database Systems
- Future Directions and Research Problems

Why Parallel Databases?

- Relational Data Model – Relational queries are ideal candidates for parallelization

- Multiprocessor systems using inexpensive microprocessors provide more power and scalability than expensive mainframe counterparts

Why do the relational queries allow for parallelization?

Properties of Ideal Parallelism

- Linear Speedup – An N-times large system yields a speedup of N

$$Speedup = \frac{small\_system\_elapsed\_time}{big\_system\_elapsed\_time}$$

- Linear Scaleup – An N-times larger system performs an N-times larger job in the same elapsed time as the original system

$$Scaleup = \frac{small\_system\_elapsed\_time\_on\_small\_problem}{big\_system\_elapsed\_time\_on\_big\_problem}$$

Why are linear speedup and linear scaleup indicators of good parallelism?

Types of Scaleup

- **Transactional** – N-times as many clients submit N-times as many requests against an N-times larger database.
- **Batch** – N-times larger problem requires an N-times larger computer.

Should we look at both transactional scaleup and batch scaleup for all situations when assessing performance?

Barriers to Linear Speedup and Linear Scaleup

- **Startup:** The time needed to start a parallel operation dominates the total execution time
- **Interference:** A slowdown occurs as new processes are added when accessing shared resources
- **Skew:** A slowdown occurs when the variance exceeds the mean in job service times

Give example situations where you might come across interference and skew barriers?

Shared-Nothing Machines

- **Shared-memory** – All processors have equal access to a global memory and all disks
- **Shared-disk** – Each processor has its own private memory, but has equal access to all disks
- **Shared-nothing** – Each processor has its own private memory and disk(s)

Why do shared-nothing systems work well for database systems?

Parallel Dataflow Approach

- Relational data model operators take relations as input and produce relations as output
- Allows operators to be composed into dataflow graphs (operations can be executed in parallel)

How would you compose a set of operations for an SQL query to be done in parallel?

Pipelined vs. Partitioned Parallelism

- **Pipelined:** The computation of one operator is done at the same time as another as data is streamed into the system.
- **Partitioned:** The same set of operations is performed on tuples partitioned across the disks. The set of operations are done in parallel on partitions of the data.

Can you give brief examples of each? Which parallelism is generally better for databases?

Data Partitioning

- **Round-robin** – For n processors, the i^{th} tuple is assigned to the "i mod n" disk.
- **Hash partitioning** – Tuples are assigned to the disks using a hashing function applied to select fields of the tuple.
- **Range partitioning** – Clusters tuples with similar attribute values on the same disk.

What operations/situations are ideal (or not ideal) for the above partitioning types?

What issues to database performance may arise due to partitioning?

Skew Problems

- **Data skew** – The number of tuples vary widely across the partitions.
- **Execution skew** – Due to data skew, more time is spent to execute an operation on partitions that contain larger numbers of tuples.

How could these skew problems be minimized?

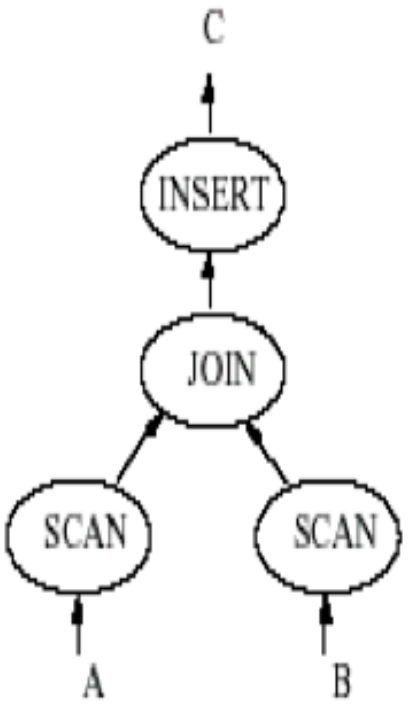
Parallel Execution of Relational Operators

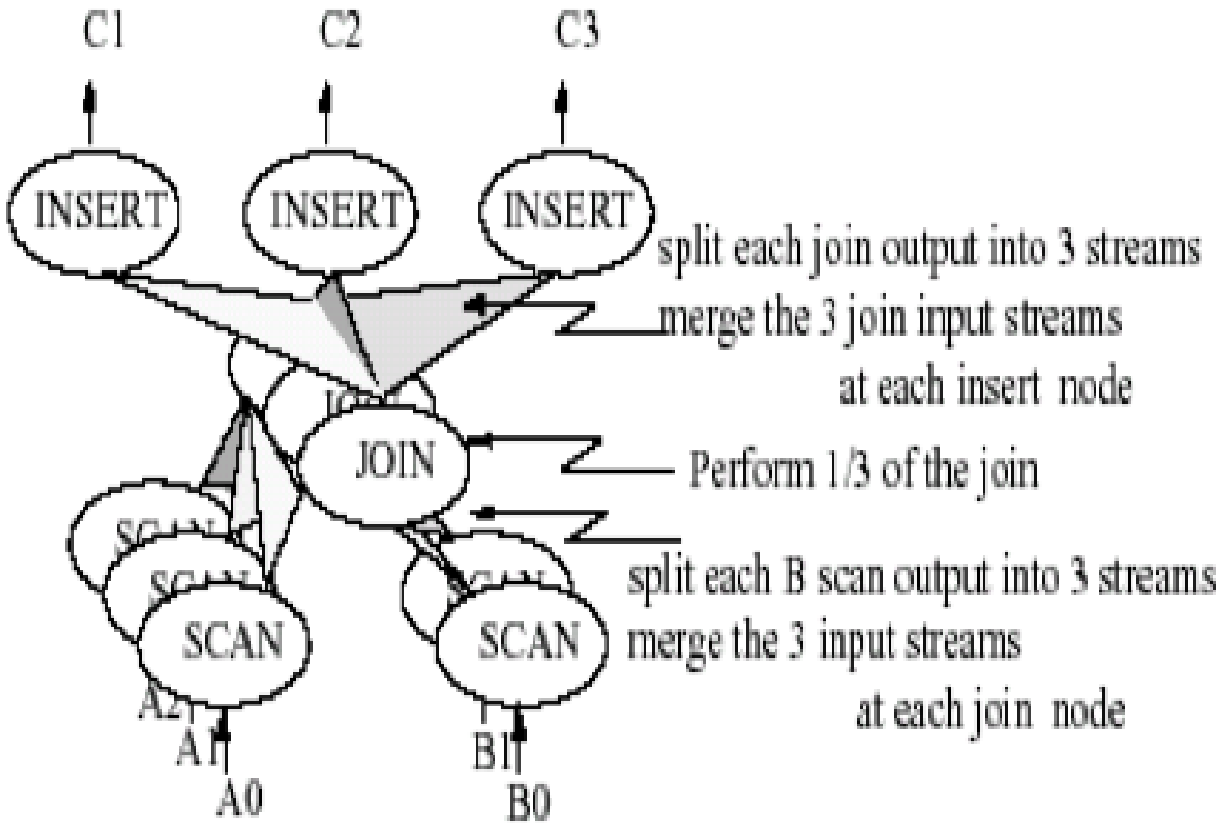
- Rather than write new parallel operators, use parallel streams of data to execute existing relational operators in parallel
- Relational operators have (1) a set of **input ports** and (2) an **output port**
- **Parallel dataflow** – partitioning and merging of data streams into the operator ports

Are there situations in which parallel dataflow does not work well?

Simple Relational Dataflow Graph

```
insert into C
  select    *
  from      A, B
  where     A.x = B.y;
```





Parallelization of Hash-Join

- Sort-merge join does not work well if there is data skew
- **Hash-join** is more resistant to data skew
- To join 2 relations, we divide the join into a set of smaller joins
- The union of these smaller joins should result in the join of the 2 relations

How might other relational algorithms/operations (like sorting) be parallelized to improve query performance?

Parallel Database Systems

- **Teradata** – Shared-nothing systems that use commodity hardware. Near linear speedup and scaleup on relational queries.
- **Tandem NonStop SQL** – Shared-nothing architecture. Near linear speedup and scaleup for large relational queries.
- **Gamma** – Shared-nothing system. Provides hybrid-range partitioning. Near linear speed and scaleup measured.
- **The Super Database Computer** – Has special-purpose components. However, is shared-nothing architecture with dataflow.
- **Bubba** – Shared-nothing system, but uses shared-memory for message passing. Uses FAD rather than SQL. Uses single-level storage

Should we focus on the use of commodity hardware to build parallel database systems?

Database Machines and Grosch's Law

- **Grosch's Law** – The performance of computer systems increase as the square of their cost. The more expensive systems have better cost to performance ratios.
- Less expensive shared-nothing systems like Gamma, Tandem, and Teradata achieve near-linear performance (compared to mainframes)
- Suggests that Grosch's Law no longer applies

Future Directions and Research Problems

- **Concurrent execution of simple and complex queries**
  - Problem with locks and large queries
  - Priority scheduling (priority inversion scheduling)
- **Parallel Query Optimization** – Optimizers do not take into account parallel algorithms. Data skew creates poor query plan cost estimates.
- **Application Program Parallelization** – Database applications are not parallelized. No effective way to automate the parallelization of applications.

Future Directions and Research Problems

- **Physical Database Design** – Tools are needed to determine indexing and partitioning schemes given a database and its workload.
- **On-line Data Reorganization** – Availability of data for concurrent reads and writes as database utilities create indices, reorganize data, etc. This process must be (1) online, (2) incremental, (3) parallel, and (4) recoverable.

Which of these issues may be extremely difficult to resolve?