Parallel and Distributed Databases

Parallel or Distributed?

Goal: move DBMS computations to multiple CPUs.

Two reasons for it:

- **Performance bottlenecks**: want to make DBMS faster when processing queries.
- **Data ownership**: different data used in the database is owned/operated by different entities within an enterprise.

Parallel databases: extension of DBMS functionality to computations on multiple CPUs in order to address the **performance bottlenecks** issue.

Distributed databases: extension of DBMS functionality to computations on multiple CPUs in order to address the **data ownership** issue.

Parallel Databases

Goal: make DBMS perform better.

Performance characteristics. The following performance characteristics can be improved:

- **Throughput.** Number of transactions completed per unit of time.
- **Response time.** Amount of time it takes to complete each transaction.

Speedup and scaleup. Two performance metrics for parallel databases.

speedup: running a given task in less time by increasing parallelism.

scaleup: handling larger tasks by increasing parallelism.

Parallel Database Architectures.

Three key parallel database architectures and their combination:

1. Shared memory
2. Shared disk
3. Shared nothing
4. Hierarchical

Shared Memory. Multiple CPUs are interconnected via a bus to shared RAM, and, by its virtue, shared disk space.

Advantages.

- Efficient communication between processors.
- No need to distribute data.
- Straightforward extensions of single-CPU algorithms for some types of parallelism (see below).

Disadvantages.

- Shared memory access is a *bottleneck*.
- Cannot scale to more than 32-64 CPUs.
- On-board CPU caches used to avoid main memory access. Updates to data by one CPU may cause *stale caches*.

Shared Disk. Each CPU has its own private RAM, but they are connected to the same disk/collection of disks.

Advantages.

- No bottleneck for memory access.
- *Falut tolerance*: CPUs can fail.
- RAID array disk storage makes disks fault tolerant too.

Disadvantages.

- Slower communiction between processors.
- Disk access is a *bottleneck*.
- Scalability. Better than for *shared memory* but not as good as in *shared nothing*.

Shared Nothing. Each CPU comes with its own private RAM and its own private disk space. *Shared nothing* architecture is essentially a cluster of commodity computers connected via a network.

Advantages.

- Cost. Commodity hardware.
- Scalability. Scales to any number of CPUs/hardware units.
- No in-system performance bottlenecks.

Disadvantages.

- Communication between CPUs is slow.
- May require non-local disk access.
- Requires data partition/replication.

Types of Parallelism

Parallelism in databases can be achieved in three different ways (or via a combination of the three methods below):

1. **I/O Parallelism.** Partitioning of relations to multiple disks.
2. **Interquery Parallelism.** Executing transactions in parallel on multiple CPUs.
3. **Intraquery Parallelism.** Executing the same transaction on multiple CPUs.

I/O Parallelism.

Types of queries:

1. Scans. (`SELECT * FROM Employees;`)
2. Point Queries. (`SELECT * FROM Employees WHERE Id = 2034;`)
3. Range Queries. (`SELECT * FROM Employees WHERE Salary > 10000 AND Salary < 40000;`)

Types of relation partitioning:

1. **Round-robin.** Spread tuples into partitions in a round-robin fashion.

   Advantages: partitions are balanced in size. works well for scans.

   Disadvantages: no order; not good for point or range queries.

2. **Hash partitioning**. Hash a subset of attributes to values from 0 to $n-1$ where n is a number of disks. Place tuples into appropriate partition.

   Advantages: excellent for point queries, works for scans.

   Disadvantages: partitions may be not balanced in size; does not work for range queries; hard to rebalance.

3. **Range Partitioning.** Pick an attribute, separate its values into n ranges, assign each range to a partition.

   Advantages: excellent for range queries, works for scans and point queries on the partition attribute.

   Disadvantages: partitions may be not balanced in size; does not work for point queries on non-partition attributes;

   Note: Dynamic rebalancing for range is often used.

Interoperation Parallelism

Allows transaction manager to schedule different transactions on different CPUs.

Advantages.

- Straightforward extension of RDBMS technology.
- Does not require rewrites of relational algebra toolkit.
- Works with *shared memory* and *shared disk* architechtures.
- Increases *throughput*, and, thus, *scalability*.

Disadvantages.

- Does not increase *response time*, and thus, *speedup* much.

Intraoperation Parallelism

Execute different portions of the same transaction on different CPUs.

Advantages.

- Increases *response time*, improves *speedup*.
- Works with *shared nothing* architectures well.

Disadvantages.

- Does not have much effect on *throughput* and *scalability*.
- Requires parallel implementations of relational algebra operations.