## Parallel Databases

- increased transaction requirements
- increased volumes of data (particularly in data-warehousing
- Many queries lend themselves easily to parallel execution
- Can reduce time required to retrieve relations from disk by partitioning relations onto a set of disks
- Horizontal partitioning usually used. Subsets of a relation are sent to different disks

## Round Robin

- Assume $n$ disks.
- With Round Robin: Relation is scanned in some order. The $i^{th}$ relation is sent to disk $D_i mod n$
- Guarantees an even distribution.

## Hash Partitioning

- choose attributes to act as partitioning attributes.
- define a hash function with range $0 \dots n - 1$
- Each tuple is placed according to the result of the hash function

## Range Partitioning

- partitioning attribute is chosen
- Partitioning vector is defined $< v_0, v_1, \ldots v_{n-2} >$
- tuples are placed according to value of partitioning attribute. If t[partitioning attribute] < $v_0$, place tuple t on disk $D_0$

## Query Types

Common types of queries

1. Scanning entire relation (batch processing)
2. Point-Queries (return all tuples that match some value)
3. Range-Queries (return all tuples with some value in some range)

## Comparison of Partitioning techniques

- Round Robin
  - useful for batch processing
  - not very suitable for point or range querying as all disks have to be accessed.

## Comparison of Partitioning techniques

- Round Robin
  - useful for batch processing
  - not very suitable for point or range querying as all disks have to be accessed.
- Hash partitioning
  - very useful if point query based on partitioning attribute.
  - usually useful for batch querying is a fair hash function is used
  - poor for range querying

## Comparison of Partitioning techniques

- Round Robin
  - useful for batch processing
  - not very suitable for point or range querying as all disks have to be accessed.
- Hash partitioning
  - very useful if point query based on partitioning attribute.
  - usually useful for batch querying is a fair hash function is used
  - poor for range querying
- Range Partitioning
  - Useful for point and range querying
  - Can lead to inefficiency in range querying if many tuples satisfy condition

## Inter-query Parallelism

- different transactions run in parallel on different processors
- Transaction throughput is increased
- The times for individual queries remains the same
- easiest form of parallelism to implement

### Intra-query parallelism

- Can run a single query in parallel on multiple processors (and disks)
- Can speed up running time of query
- Can achieve parallel execution by parallelising individual components ( intra-operation parallelism)
- Can also achieve parallel execution by evaluating portions of the query in parallel (inter-operation parallelism)
- Can also combine both

## Parallel Sorting

- **Range-Partitioning Sort**
- Distribute the relation using a range-partitioning strategy on the sort attribute
- Each subset is sorted in parallel. The final merge is not expensive due to the range partitioning strategy chosen

## Parallel External Sort-Merge

- Relation is partitioned.
- Each processor $P_i$ sorts the tuples at $D_i$
- The sorted runs are then merged in parallel.
- Sorted runs are range-partitioned across a set of processors.
- Each processor performs a merge on the incoming streams
- These sorted runs are then concatenated.

## Parallel Join

Wish to compute r $\bowtie$ s

## Partitioned Join

- Partition relations across the n processors
- Compute $r_0 \bowtie s_0$ at at processor $P_0$, $r_1 \bowtie s_1$ at processor $P_1$ etc.
- can partition relations using hash or range partitioning
- suitable for equi-joins; not suitable for other types.

## Parallel Join

Wish to compute r $\bowtie$ s

## Partitioned Join

- Partition relations across the n processors
- Compute $r_0 \bowtie s_0$ at at processor $P_0$, $r_1 \bowtie s_1$ at processor $P_1$ etc.
- can partition relations using hash or range partitioning
- suitable for equi-joins; not suitable for other types.

## Fragment and Replicate

- Wish to calculate $r \bowtie_{x>y} s$
- partition $r$ across the processors
- $s$ is replicated at all processors
- $r_i \bowtie_{x>y} s$ is calculated at all processors