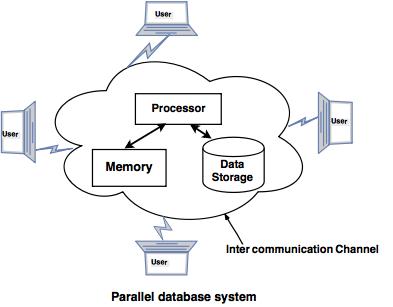
Introduction to Parallel Databases

Companies need to handle huge amount of data with high data transfer rate. The client server and centralized system is not much efficient. The need to improve the efficiency gave birth to the concept of Parallel Databases.  
  
Parallel database system improves performance of data processing using multiple resources in parallel, like multiple CPU and disks are used parallely.  
  
It also performs many parallelization operations like, data loading and query processing.

Goals of Parallel Databases

**The concept of Parallel Database was built with a goal to:**  
  
**Improve performance:**  
The performance of the system can be improved by connecting multiple CPU and disks in parallel. Many small processors can also be connected in parallel.  
  
**Improve availability of data:**  
Data can be copied to multiple locations to improve the availability of data.  
**For example:** if a module contains a relation (table in database) which is unavailable then it is important to make it available from another module.  
  
**Improve reliability:**  
Reliability of system is improved with completeness, accuracy and availability of data.  
  
**Provide distributed access of data:**  
Companies having many branches in multiple cities can access data with the help of parallel database system.  


Parallelism in Databases !

Data can be partitioned across multiple disks for parallel I/O. !

Individual relational operations (e.g., sort, join, aggregation) can be executed in parallel ! data can be partitioned and each processor can work independently on its own partition.

Queries are expressed in high level language (SQL, translated to relational algebra)

makes parallelization easier.

Different queries can be run in parallel with each other. Concurrency control takes care of conflicts.

**I/O Parallelism**

It is simplest form, I/O parallelism refers to reducing the time required to retrieve relations from disk by partitioning the relations on multiple disks.

The most common form of data partitioning in a parallel database environment is horizontal partitioning.

In horizontal partitioning, the tuples of a relation are divided (or declustered) among many disks, so that each tuple resides on one disk.

***Partitioning Techniques***

Assume that there are n disks, D0,D1, . . .,Dn−1, across which the data are to be partitioned.

* **Round-robin.** This strategy scans the relation in any order and sends the ith tuple to disk number Di mod n.
* The round-robin scheme ensures an even distribution of tuples across disks; that is, each disk has approximately the same number of tuples as the others.
* **Hash partitioning**. This declustering strategy designates one or more attributes from the given relation’s schema as the partitioning attributes. A hash function is chosen whose range is {0, 1, . . . , n − 1}. Each tuple of the original relation is hashed on the partitioning attributes. If the hash function returns i, then the tuple is placed on disk Di.
* **Range partitioning**. This strategy distributes contiguous attribute-value ranges to each disk. It chooses a partitioning attribute, A, as a partitioning vector. The relation is partitioned as follows.
* Let [v0, v1, . . . , vn−2] denote the partitioning vector, such that, if i < j, then vi < vj.
* Consider a tuple t such that t[A] = x. If x < v0, then t goes on disk D0. If x ≥ vn−2, then t goes on disk Dn−1. If vi ≤ x < vi+1, then t goes on disk Di+1.

For example, range partitioning with three disks numbered 0, 1, and 2 may assign tuples with values less than 5 to disk 0, values between 5 and 40 to disk 1, and values greater than 40 to disk 2.

Evaluate how well partitioning techniques support the following types of data access:

1. Scanning the entire relation.

2. Locating a tuple associatively – point queries. – E.g., r.A = 25.

3. Locating all tuples such that the value of a given attribute lies within a specified range – range queries.

Round-robin. – Best suited for sequential scan of entire relation on each query.

All disks have almost an equal number of tuples; retrieval work is thus well balanced between disks.

– Range queries are difficult to process ∗ No clustering – tuples are scattered across all disks

Hash partitioning. – Good for sequential access

∗ Assuming hash function is good, and partitioning attributes form a key, tuples will be equally distributed between disks

∗ Retrieval work is then well balanced between disks. – Good for point queries on partitioning attribute

∗ Can lookup single disk, leaving others available for answering other queries.

∗ Index on partitioning attribute can be local to disk, making lookup and update more efficient – No clustering, so difficult to answer range queries

**Handling of Skew !**

The distribution of tuples to disks may be skewed — that is, some disks have many tuples, while others may have fewer tuples.

**Types of skew:**

**Attribute-value skew.** " Some values appear in the partitioning attributes of many tuples;

all the tuples with the same value for the partitioning attribute end up in the same partition. " Can occur with range-partitioning and hash-partitioning.

**Partition skew**. " With range-partitioning, badly chosen partition vector may assign too many tuples to some partitions and too few to others. " Less likely with hash-partitioning if a good hash-function is chosen

## Query parallelism \*\*\*

There are two types of query parallelism: interquery parallelism and intraquery parallelism.

## The two techniques used in query evaluation are as follows: 1. Inter-query Parallelism

* This technique allows to run multiple queries on different processors simultaneously.
* Pipelined parallelism is achieved by using inter query parallelism, which improves the output of the system.

It is a form of parallelism where many different Queries or Transactions are executed in parallel with one another on many processors.

It is also called as parallel transaction processing.

#### Advantages

It increases Transaction Throughput. That is, number of transactions executed in a given time can be increased.

It scales up the Transaction processing system. Hence, best suited for On-Line Transaction Processing (OLTP) systems.

#### Supported Parallel Database Architectures

It is easy to implement in [Shared Memory](http://exploredatabase.blogspot.in/2014/02/parallel-database-architectures.html) Parallel System. Lock tables and Log information are maintained in the same memory. Hence, it is easy to handle those transactions which shares locks with other transactions. Locking and logging can be done efficiently.

In other parallel architectures like [Shared Disk and Shared Nothing](http://exploredatabase.blogspot.in/2014/02/parallel-database-architectures.html), the locking and logging must be done through message passing between processors, which is considered as costly operation when compared Shared Memory Parallel architecture. [**Cache coherency**](http://exploredatabase.blogspot.in/2014/02/some-definitions-in-parallel-database.html) problem would occur.

#### Example Database systems which support Inter-query Parallelism

Oracle 8 and Oracle Rdb

**2. Intra Query Parallelism**

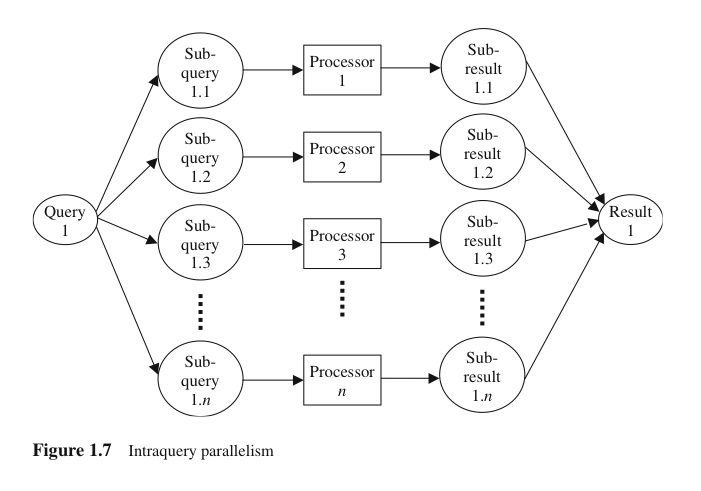
* In this technique query is divided in sub queries which can run simultaneously on different processors, this will minimize the query evaluation time.
* Intra query parallelism improves the response time of the system.

**For Example:** If we have 6 queries, which can take 3 seconds to complete the evaluation process, the  total time to complete the evaluation process is 18 seconds. But We can achieve this task in only 3 seconds by using intra query evaluation as each query is divided in sub-queries.

#### Advantages

To speed up a single complex long running queries.

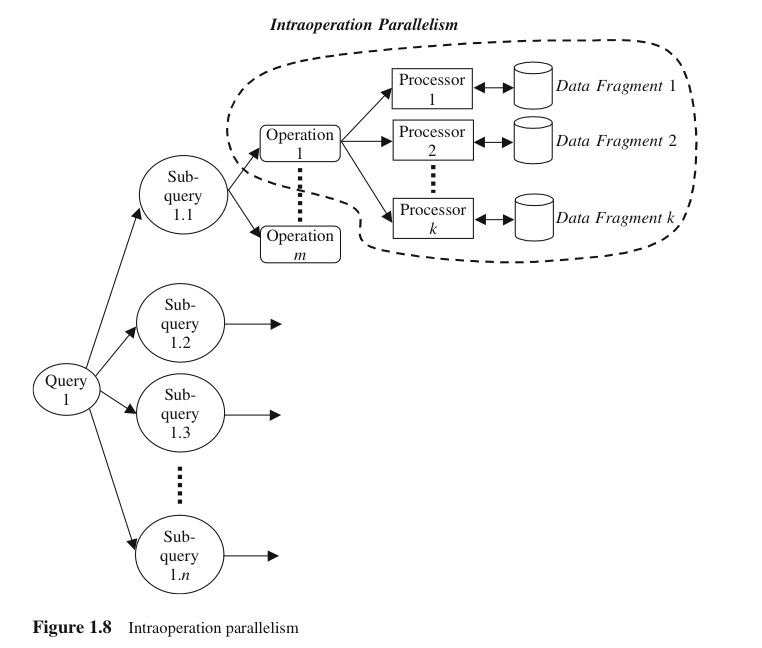
Best suited for complex scientific calculations (queries).



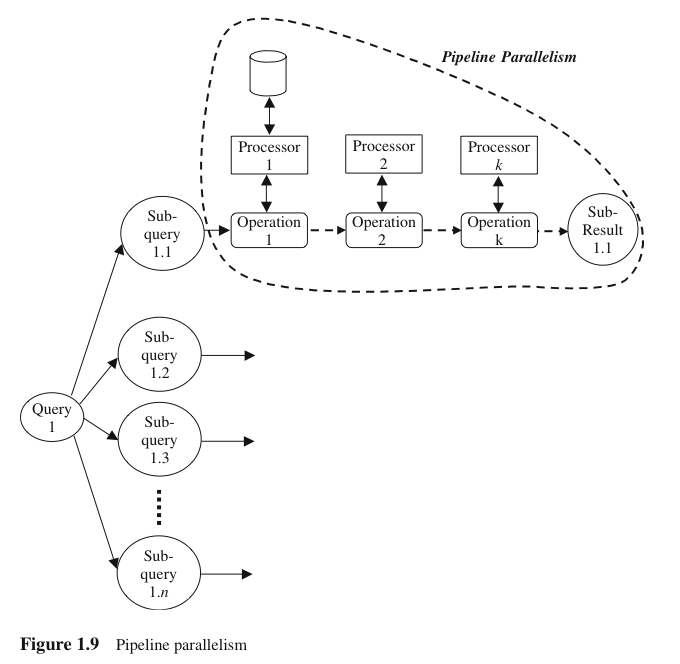
#### Example Database systems which support Intra-query Parallelism

Informix, Terradata

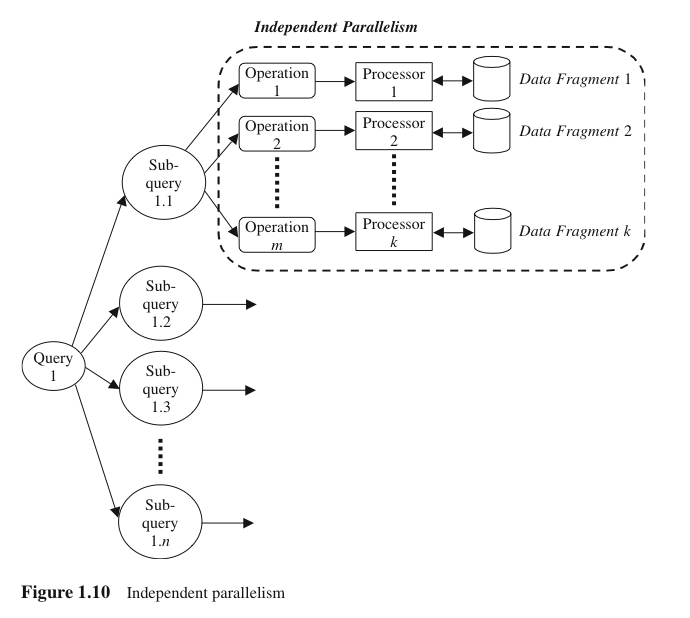
* Execution of a single query can be parallelized in two ways:
  + **Intraoperation parallelism**: Speeding up the processing of a query by parallelizing the execution of each individual operation (e.g. parallel sort, parallel search, etc)

****

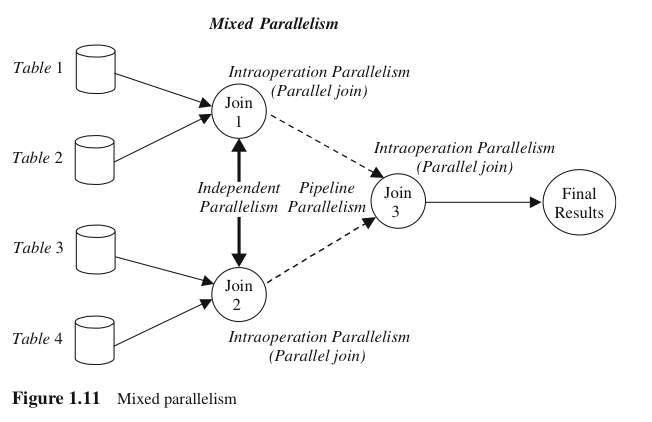
* **Interoperation parallelism: Parallelism created by concurrently executing different operations within the same query or transaction**
  + **Pipeline parallelism : Output record of one operation *A* are consumed by a second operation *B*, even before the first operation has produced the entire set of records in its output**
  + **Multiple operations form some sort of assembly line to manufacture the query results**
  + **Useful with a small number of processors, but does not scale up well**

****

* + **Independent parallelism Operations in a query that do not depend on one another are executed in parallel**
  + **Does not provide a high degree of parallelism**

****

* **Mixed Parallelism**
  + In practice, a mixture of all available parallelism forms is used.



Optimization of Parallel Query

* Parallel Query optimization is nothing but selecting the efficient query evaluation plan.
* Parallel Query optimization plays an important role in developing system to minimize the cost of query evaluation.

**Two factors play a very important in parallel query optimization.**  
  
a) total time spent to find the best plan.  
b) amount of time required to execute the plan.

Goals of Query optimization.

**Query Optimization is done with an aim to:**

* Speed up the queries by finding the queries which can give the fastest result on execution.
* Increase the performance of the system.
* Select the best query evaluation plan.
* Avoid the unwanted plan.

Approaches of Query Optimization.

**Following are the three approaches to Query Optimization:**  
  
**1. Horizontal partitioning:** Tables are created vertically using columns.  
**2. Vertical partitioning:** Tables are created with fewer columns and partition the table row wise.  
**3. De-normalization:** In this approach multiple tables are combined into one table.