Parallel Database Systems: The Future of High Performance Database Processing

Presented by: Iman Elghandour

Overview

- **Parallel Databases**
  - Machines are physically close to each other, e.g., same server room.
  - Machines connects with dedicated high-speed LANs and switches.
  - Communication cost can be ignored.

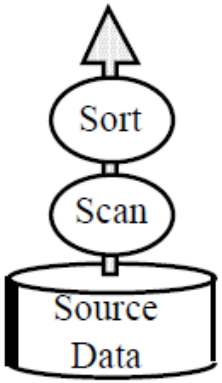
    shared-memory, shared-disk, or shared-nothing architecture
- **Distributed Databases**
  - Machines are not be physically located at the same place.
  - Can be connected using public network, e.g., Internet.
  - Communication cost and problems cannot be ignored.

    shared-nothing architecture

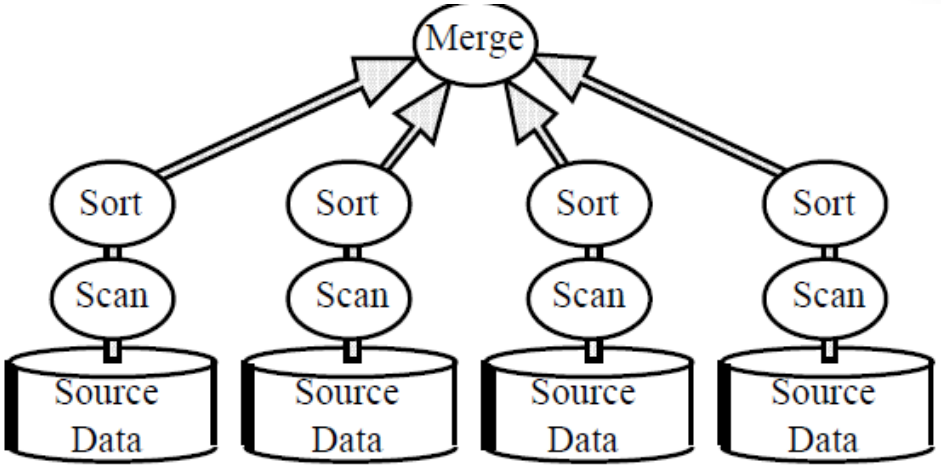
Motivation

- Large data
- Nature of the application
- Relational queries are ideally suited to parallel execution
- Improve the query execution performance
  - Speedup and Scaleup
- Special-purpose database machines (mainframes) vs parallel database built using commodity disks, processors, and memories
- Partitioning allows multiple processors to scan large relations in parallel without needing any exotic I/O devices

Pipelined and Partitioned Parallelism



pipeline parallelism

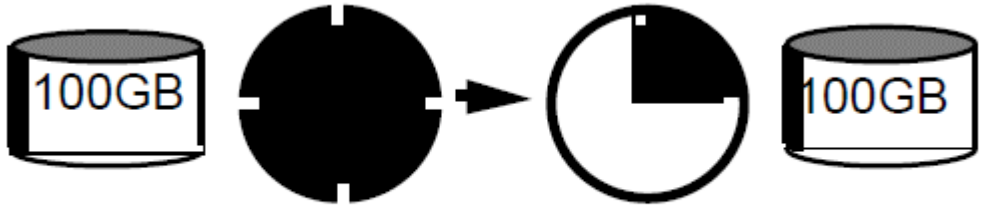


partitioned data allows partitioned parallelism

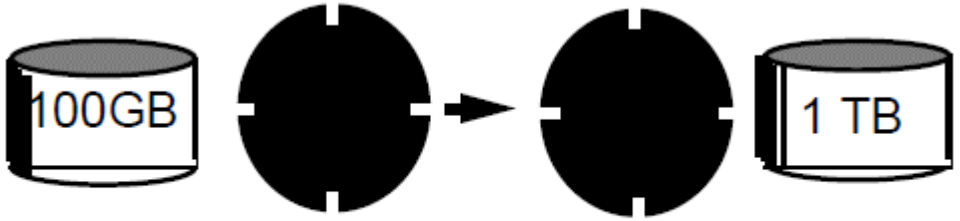
Outline

- Parallelism Goals and Metrics
- Hardware Architecture
- Parallel Dataflow Approach to SQL Software
  - Data Partitioning
  - Parallelism within Relational Operators

Parallelism Goals and Metrics: Speedup and Scaleup



Speedup



Batch Scaleup

Speedup

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

- Linear Speedup: *N*-times large or more expensive system yields a speedup of *N*.

Scaleup

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

- Scaleup: *N*-times larger system performs an *N*-times larger job in the same elapsed time
- Linear Scaleup: scaleup evaluates to 1
- Transaction-scaleup: *N*-times as many clients submit *N*-times as many requests against *N*-times larger database
- Batch-scaleup: *N*-times larger computer is used to solve *N*-times larger problem

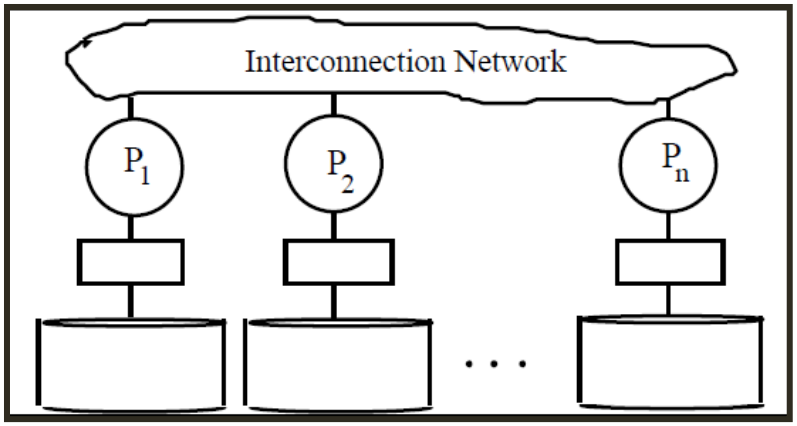
Limitations of Linear Speedup and Linear Scaleup

- **Startup**: starting up thousands of operators can dominate the computation time
- **Interference**: slowdown of processes because of accessing shared resources
- Skew: the variance of each parallel step increases, and therefore the total elapsed time to finish a job slightly improves

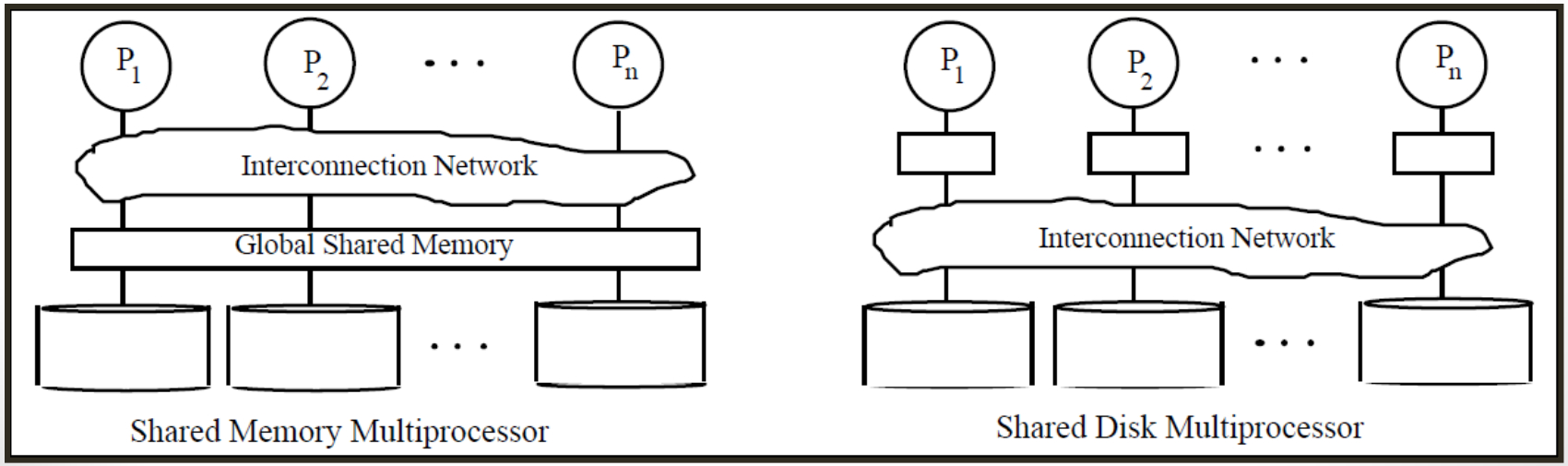
Outline

- Parallelism Goals and Metrics
- **Hardware Architecture**
- Parallel Dataflow Approach to SQL Software
  - Data Partitioning
  - Parallelism within Relational Operators

Systems with Multiprocessors



Shared-nothing design



Shared-nothing Architecture

- Each memory and disk is owned by some processor
- Advantages:
  - Minimizes interference by minimizing resource sharing
  - Exploits commodity processors and memory
  - Does not need powerful interconnection network
  - Network is used for transferring queries and answers
  - Can scaleup to hundreds/ thousands of processors
  - Can achieve near linear speedups and scaleups

Shared-memory and Shared-disk

- Do not scale well on database applications
- Interference is a major problem
- The interconnection network must have the bandwidth of the sum of processors and disks
- Usually shared-memory architectures adopt a shared disk-architecture
- Affinity scheduling

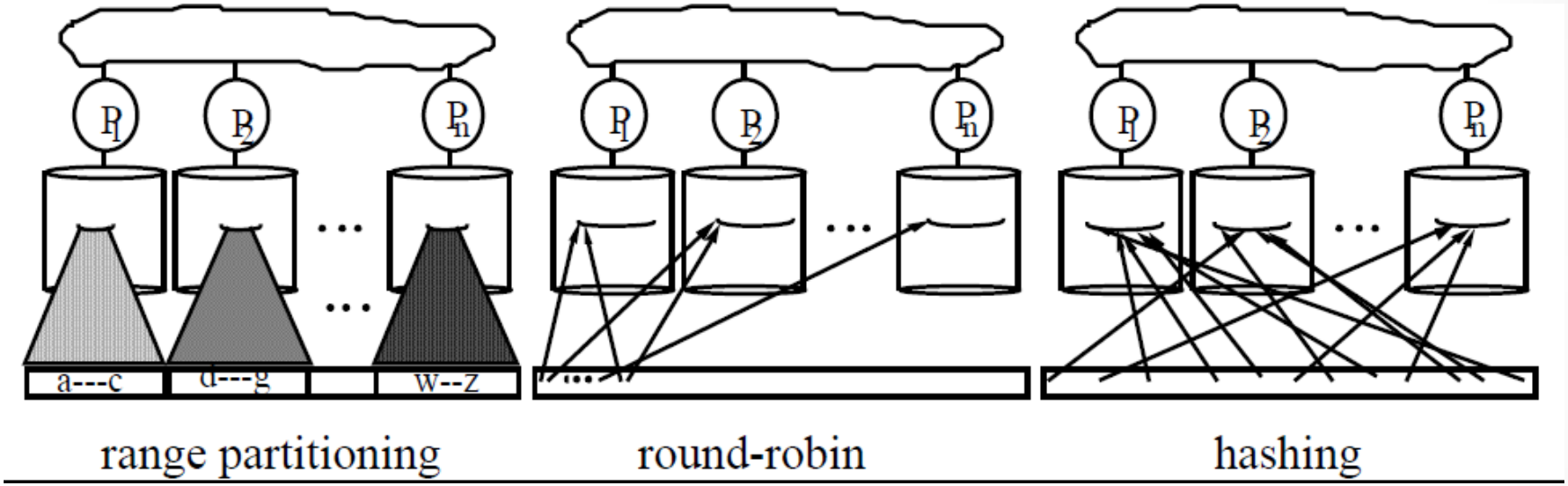
Outline

- Parallelism Goals and Metrics
- Hardware Architecture
- **Parallel Dataflow Approach to SQL Software**
  - Data Partitioning
  - Parallelism within Relational Operators

Parallel Dataflow Approach to SQL Software

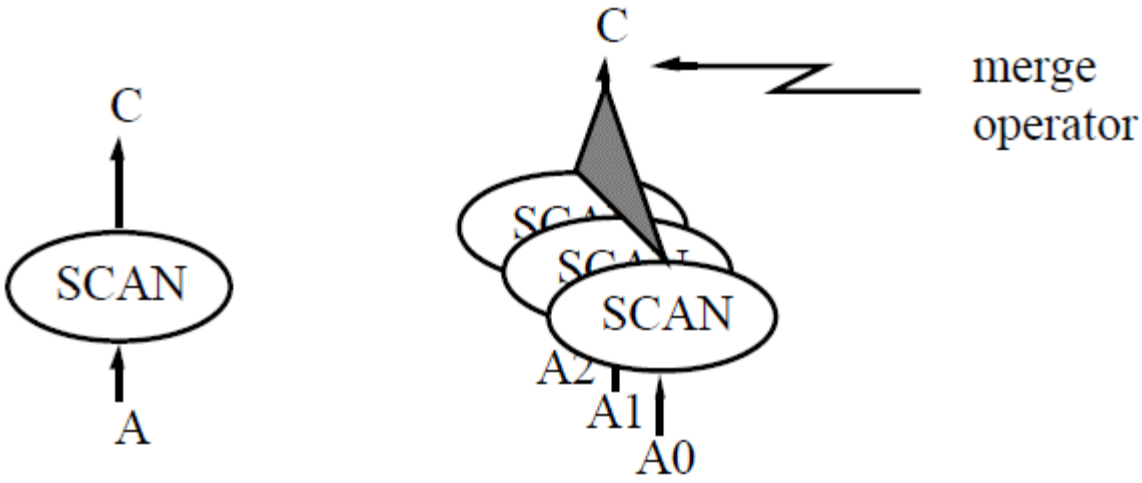
- Queries dataflow graphs can use pipelined parallelism and partitioned parallelism
- Limitations of pipelined parallelism:
  - Relational pipelines are rarely very long
  - Some operators do not emit their output until they have consumed all their input
  - Execution costs of operators are not equal

Data Partitioning



- Design issues:
  - Each relation must have a partitioning strategy
  - Increasing the degree of partitioning usually reduces the response time of an individual query and increases the whole throughput of the system

Parallelism within Relational Operators



Partitioned data parallelism

Example of a Relational Dataflow Graph