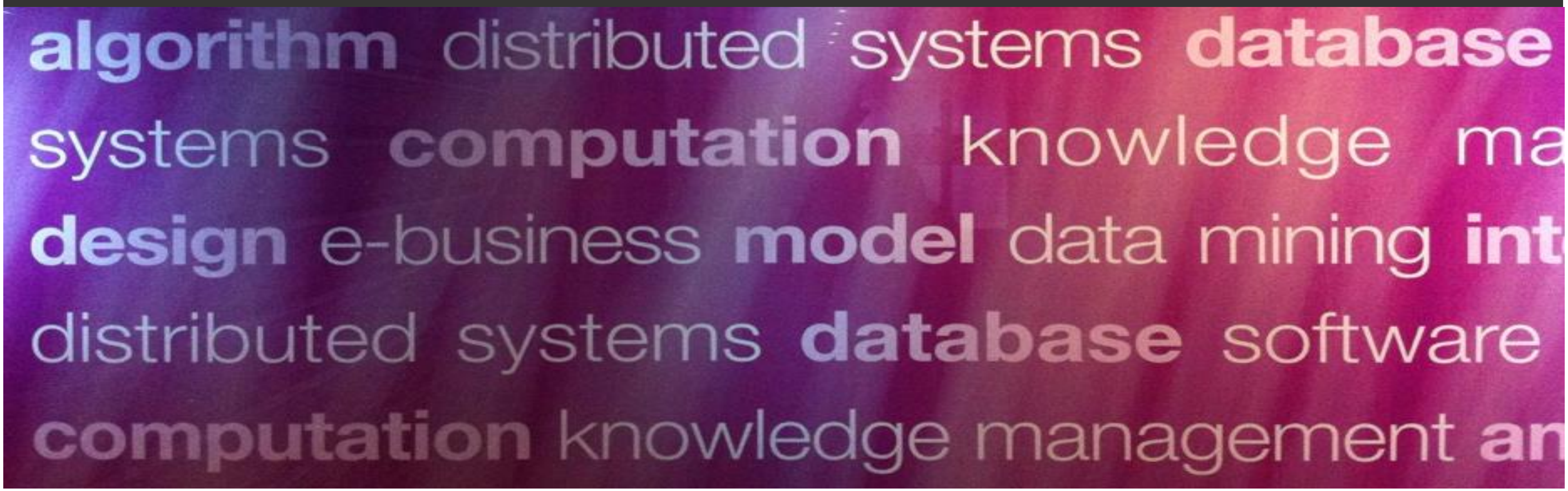
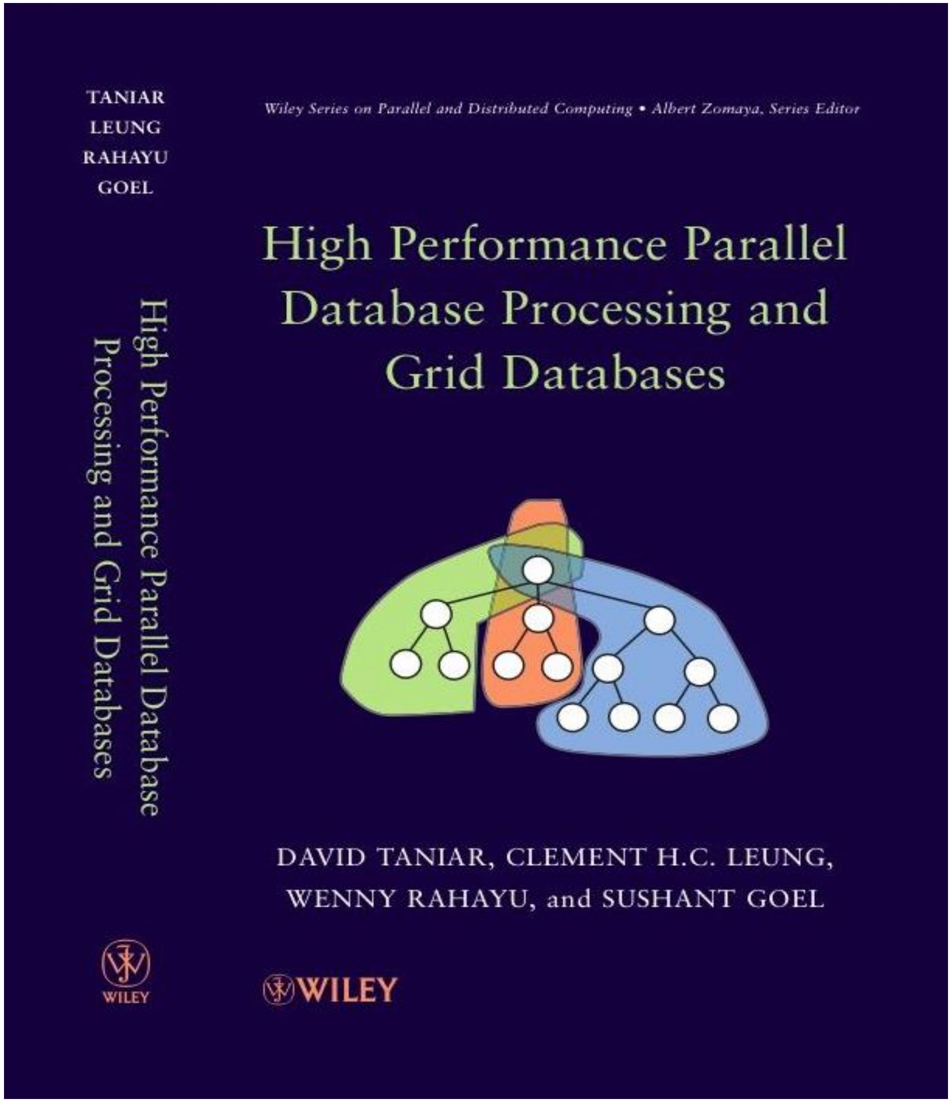
MONASH University

Information Technology

FIT5202 (Volume I - Introduction)

Week 2a – Introduction to Parallel Databases





Chapter 1 Introduction

1.3. Objectives (cont'd)

- **Parallel Obstacles**
  - Start-up and Consolidation costs,
  - Interference and Communication, and
  - Skew

Recall:

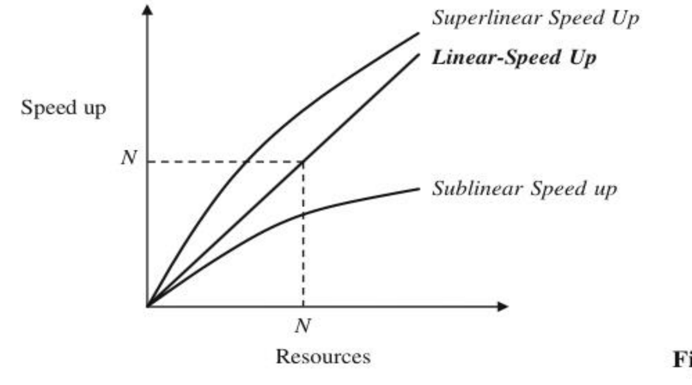


Figure 1.1 Speed up

1.3. Objectives (cont'd)

- **Start-up and Consolidation**
  - Start up: initiation of multiple processes
  - Consolidation: the cost for collecting results obtained from each processor by a host processor

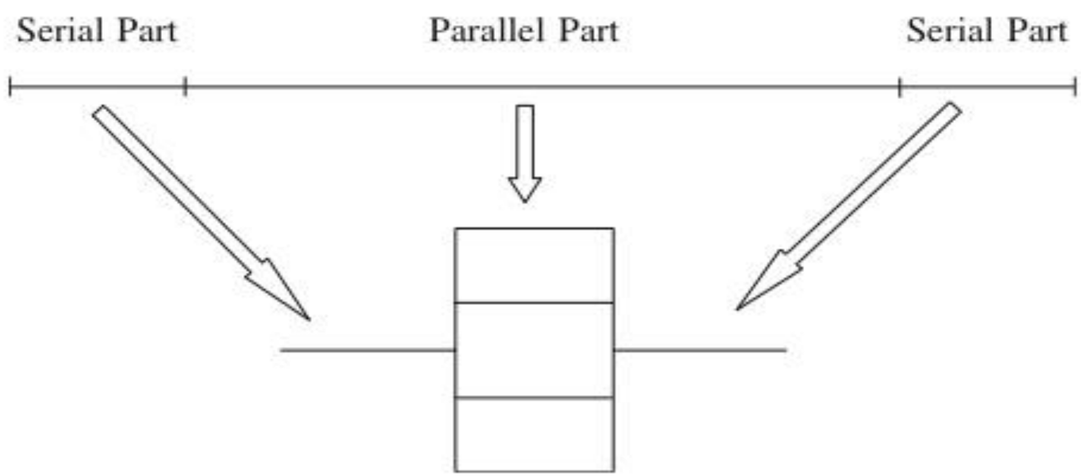


Figure 1.3 Serial part vs. parallel part

1.3. Objectives (cont'd)

- **Interference and Communication**
  - Interference: competing to access shared resources (e.g., disk & memory)
  - Communication: one process communicating with other processes, and often one has to wait for others to be ready for communication (i.e. waiting time).

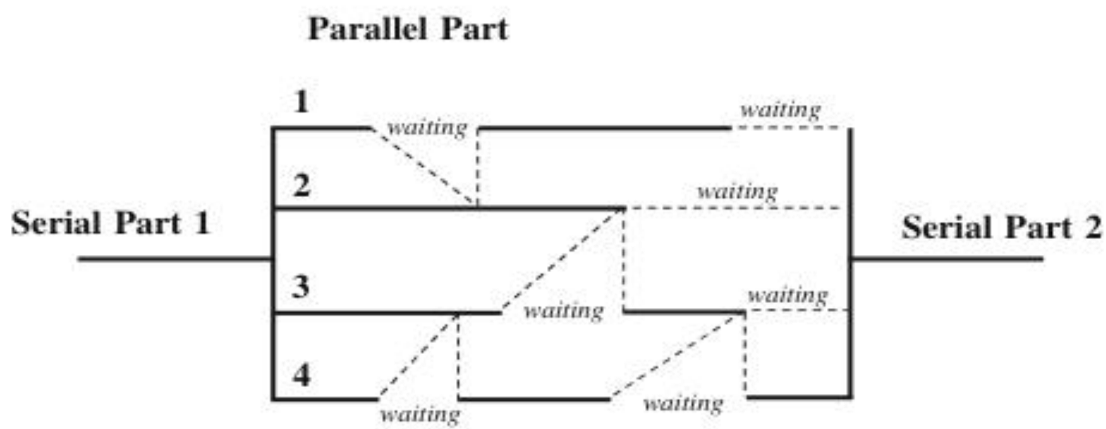


Figure 1.4 Waiting period

- **Exercise**
  - There is a job that will take 1 hour to complete, if this is done by 1 processor.
  - The serial part of this job is 10%
  - There are 4 processors to use in this job, but each processor will have an overhead of 20% due to waiting time, communication time, etc.
  - What type of **speed up** do we get?

There is a job that will take 1 hour to complete, if this is done by 1 processor. The serial part of this job is 10%. There are 4 processors to use in this job, but each processor will have an overhead of 20% due to waiting time, communication time, etc. What is the speed up? (5 Minutes)**

$$\text{Speed up} = \frac{\text{elapsed time on uniprocessor}}{\text{elapsed time on multiprocessors}}$$

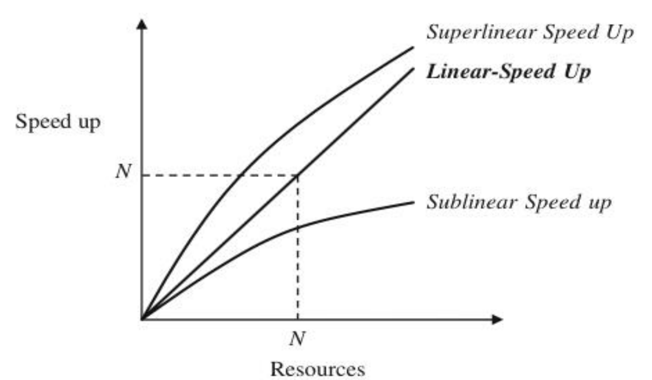


Figure 1.1 Speed up

Solution:

1 processor = 60min
Serial part = 10% = 6min
Parallel part = 54min

4 processors = 54min/4 = 13.5min

Overhead = 20%
Hence, parallel processing part = 13.5min + 20%overhead = 13.5min+2.7min = 16.2min

Total time = 6min (serial) + 16.2min (parallel) = 22.2min

Speed up = 60min / 22.2min = 2.7
Linear speedup should be 4
Speed up of 2.7 is Sub-Linear Speedup

1.3. Objectives (cont'd)

- **Skew**
  - Unevenness of workload
  - Load balancing is one of the critical factors to achieve linear speed up

- ❑ Data skew: Uneven distribution of data in terms of size, allocated to processors
- ❑ Processing skew: Uneven processing time of the processors due to data skew.

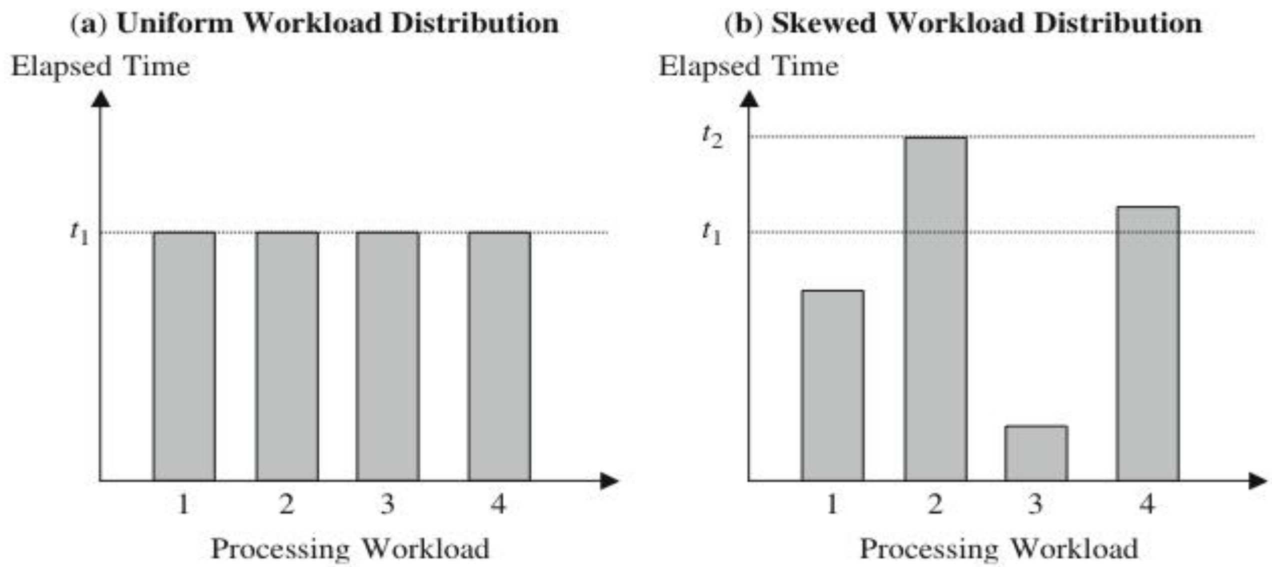


Figure 1.5 Balanced workload vs. unbalanced workload (skewed)

1.3. Objectives (cont'd)

- **Skew**
  - *Zipf* distribution model to model skew. Measured in terms of different sizes of fragments allocated to the processors

$$|R_i| = \frac{|R|}{i^{\theta} \times \sum_{j=1}^{N} \frac{1}{j^{\theta}}} \qquad \text{where } 0 \leq \theta \leq 1 \qquad (2.1)$$

  - The symbol θ denotes the degree of skewness, where $\theta = 0$ indicates no skew, and **θ = 1 indicates highly skewed**
  - **|*R*|** is number of records in the table, **|*Ri*| is number of records in processor *i***, and ***N*** is number of processor (*j* is a loop counter, starting from 1 to *N*)
  - Example: |*R*|=100,000 records, *N*=8 processors

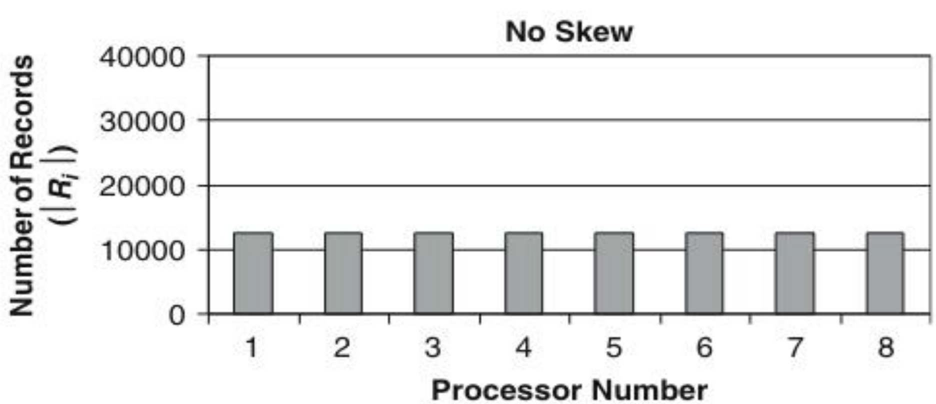


Figure 2.1 Uniform distribution (no skew)

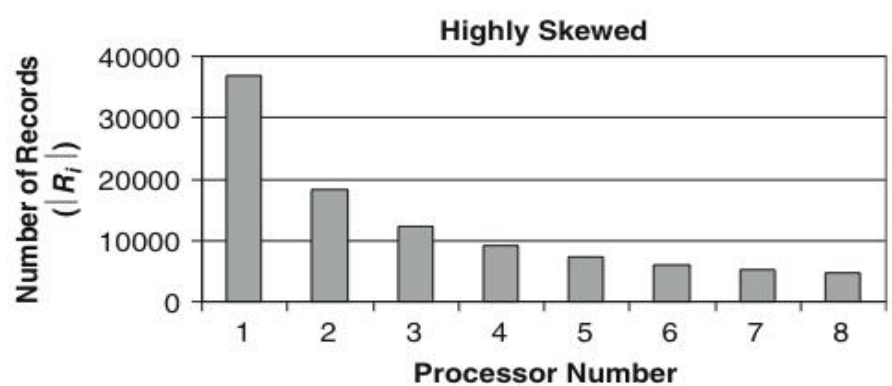


Figure 2.2 Highly skewed distribution

1.3. Objectives (cont'd)

- No skew vs. highly skewed

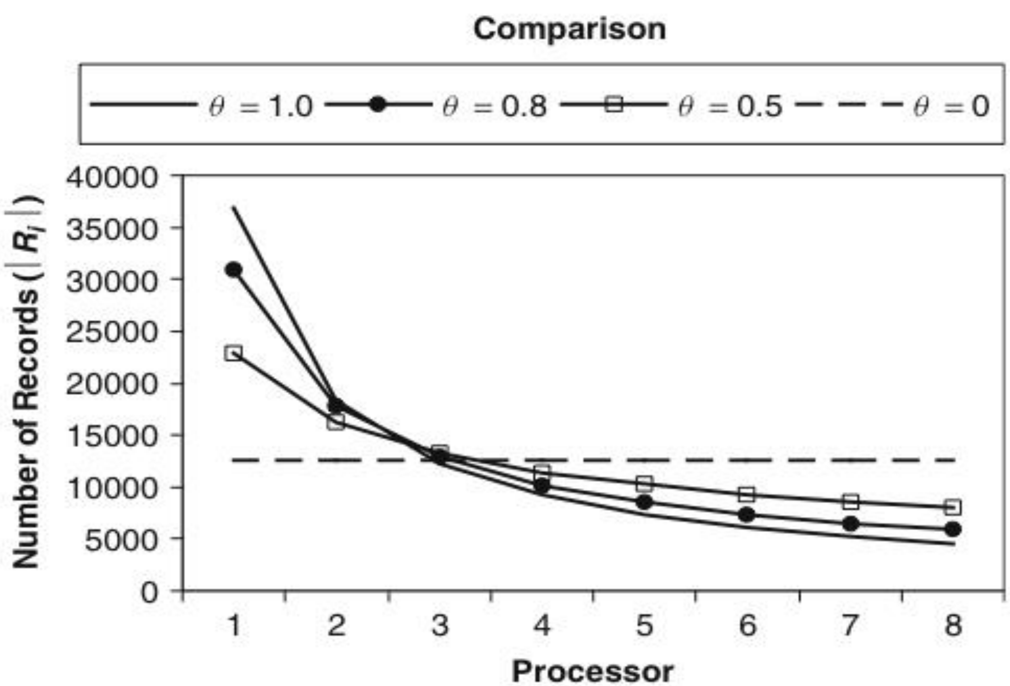
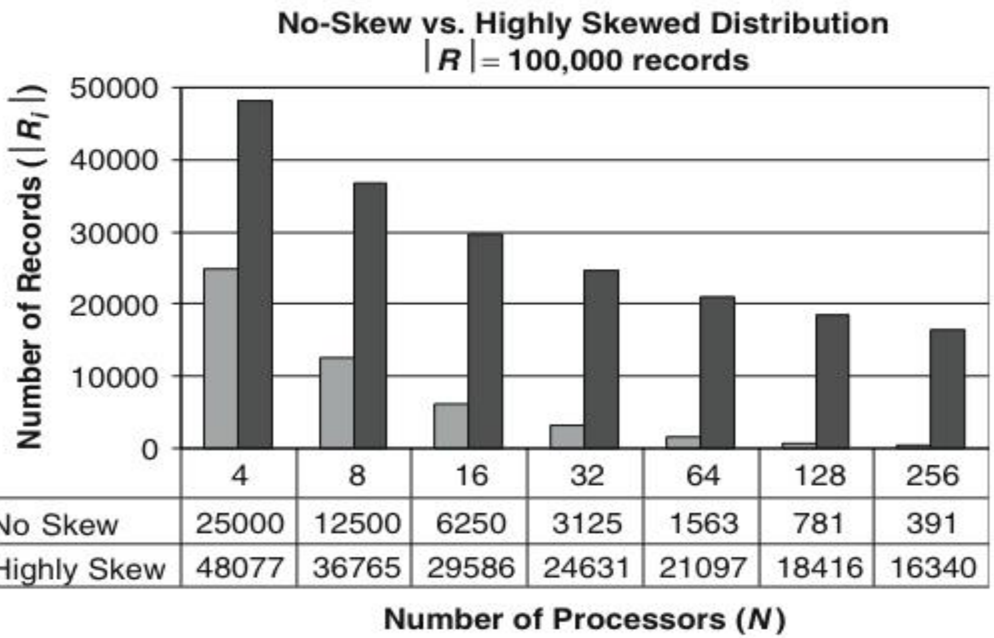


Figure 2.3 Comparison between highly skewed, less skewed, and no-skew distributions

1.3. Objectives (cont'd)

- **No skew vs. highly skewed**



	4	8	16	32	64	128	256
No Skew	25000	12500	6250	3125	1563	781	391
Highly Skew	48077	36765	29586	24631	21097	18416	16340

Figure 2.4 Comparison between the heaviest loaded processors using no-skew and highly skewed distributions

1.3. Objectives (cont'd)

- **No skew vs. highly skewed**

Table 2.2 Divisors (with vs. without skew)

N	4	8	16	32	64	128	256
Divisor without skew	4	8	16	32	64	128	256
Divisor with skew	2.08	2.72	3.38	4.06	4.74	5.43	6.12

- **Exercise 7**
  - There 100,000 records in the table to be distributed to 32 processors. Assuming that the skewness degree is high ($\theta = 1$), what is the estimated number of records in the heaviest processor?

  - A. 48,000 records
  - B. 29,000 records
  - C. 24,000 records
  - D. It is not possible to predict

1.4. Forms of Parallelism

- Forms of parallelism for database processing:
  - Interquery parallelism
  - Intraquery parallelism
  - Interoperation parallelism
  - Intraoperation parallelism
  - Mixed parallelism

1.4. Forms of Parallelism (cont'd)

- **Interquery Parallelism**
  - "Parallelism among queries"
  - Different queries or transactions are executed in parallel with one another
  - Main aim: scaling up transaction processing systems

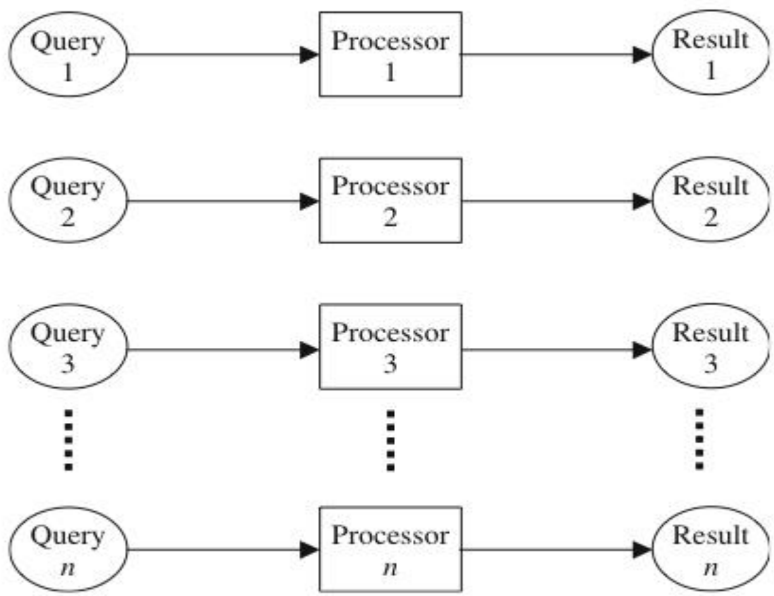


Figure 1.6 Interquery parallelism

1.4. Forms of Parallelism (cont'd)

- **Intraquery Parallelism**
  - "Parallelism within a query"
  - Execution of a single query in parallel on multiple processors and disks
  - Main aim: speeding up long-running queries

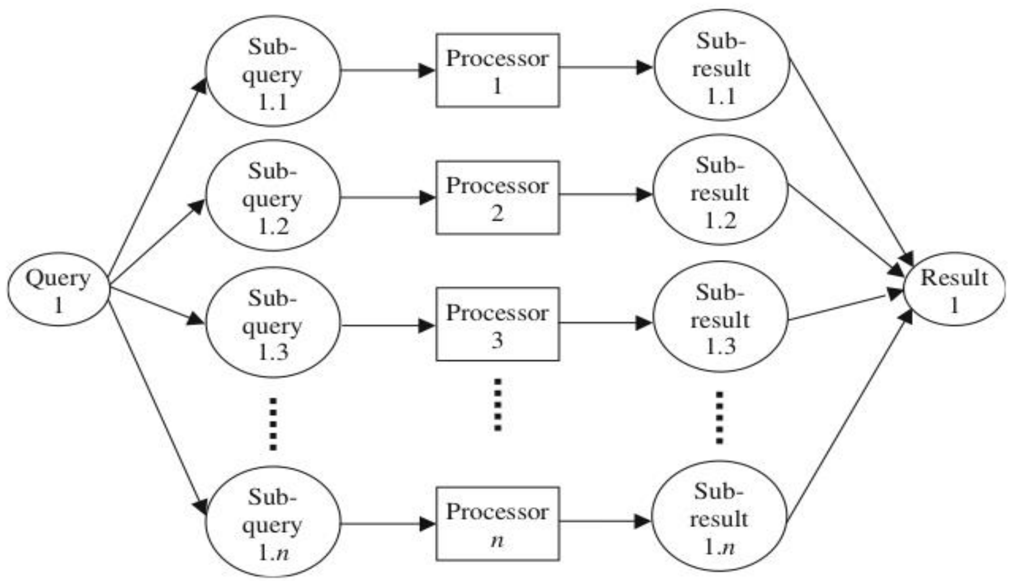


Figure 1.7 Intraquery parallelism

1.4. Forms of Parallelism (cont'd)

- 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**: Speeding up the processing of a query by executing in parallel different operations in a query expression (e.g. simultaneous sorting or searching)

1.4. Forms of Parallelism (cont'd)

- **Intraoperation Parallelism**
  - Parallelize execution of each individual operation
  - "Partitioned parallelism"
  - Parallelism due to the data being partitioned
  - Since the number of records in a table can be large, the degree of parallelism is potentially enormous

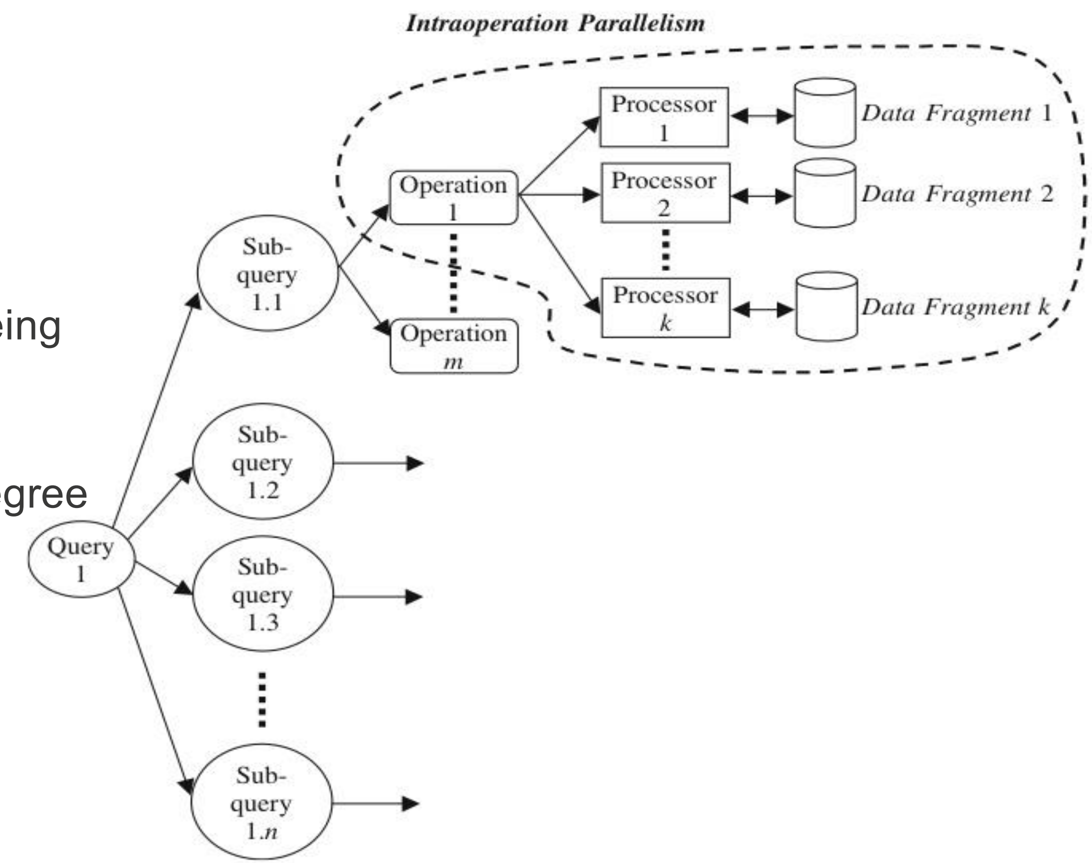


Figure 1.8 Intraoperation parallelism

1.4. Forms of Parallelism (cont'd)

- **Interoperation parallelism**: Parallelism created by concurrently executing different operations within the same query or transaction
  - Different operations are executed in parallel
    - Pipeline parallelism
    - Independent parallelism

1.4. Forms of Parallelism (cont'd)

- **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

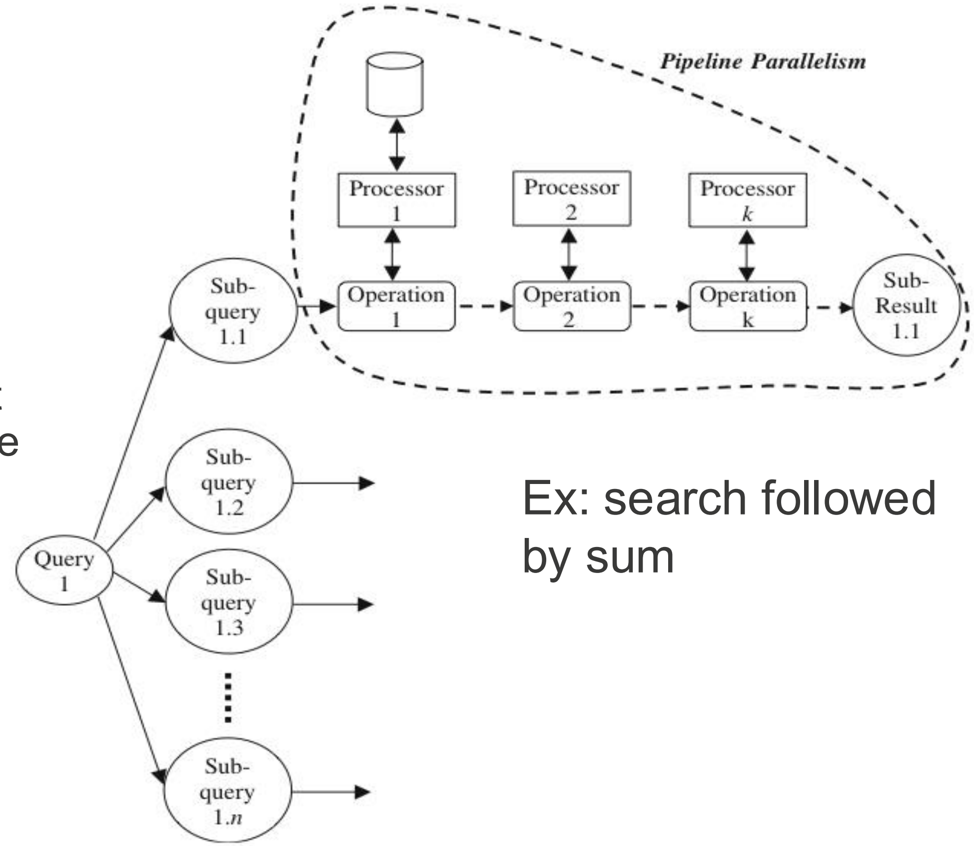


Figure 1.9 Pipeline parallelism

Ex: search followed by sum

1.4. Forms of Parallelism (cont'd)

- **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

    Parallelism is limited by the number of operations available

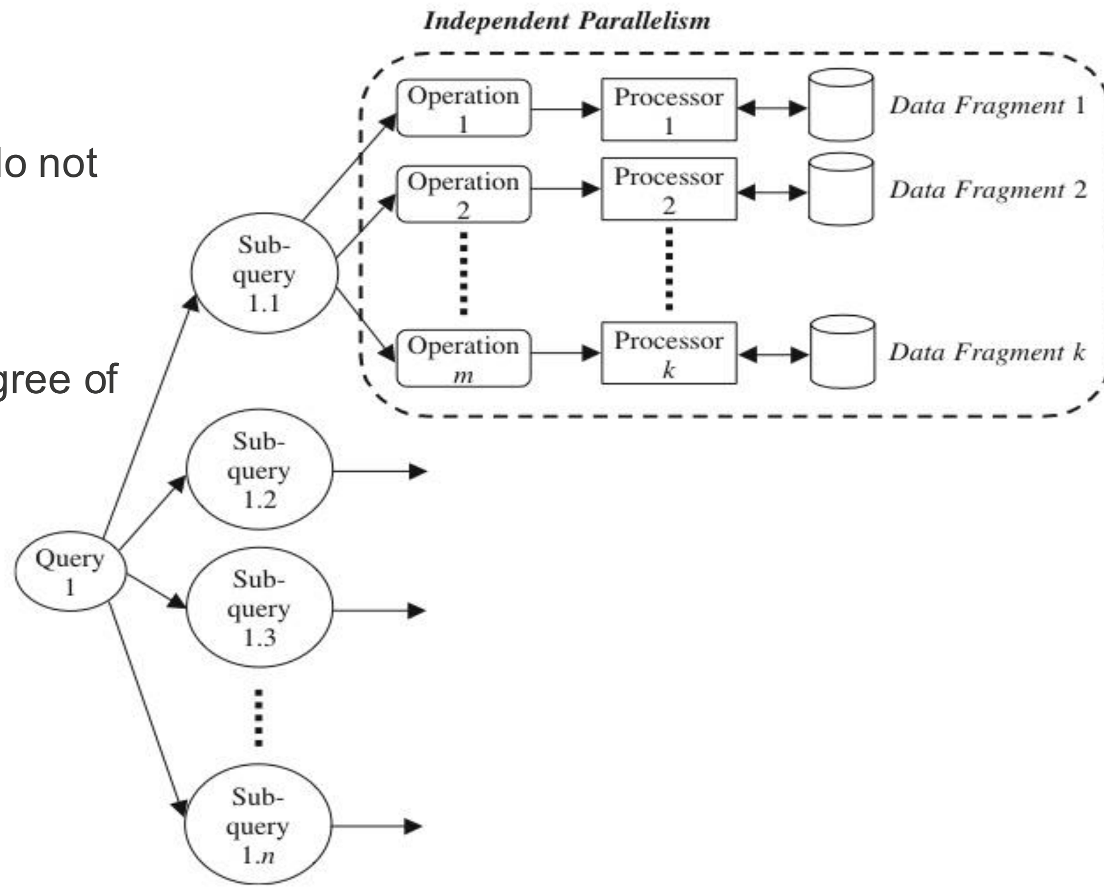


Figure 1.10 Independent parallelism

1.4. Forms of Parallelism (cont'd)

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

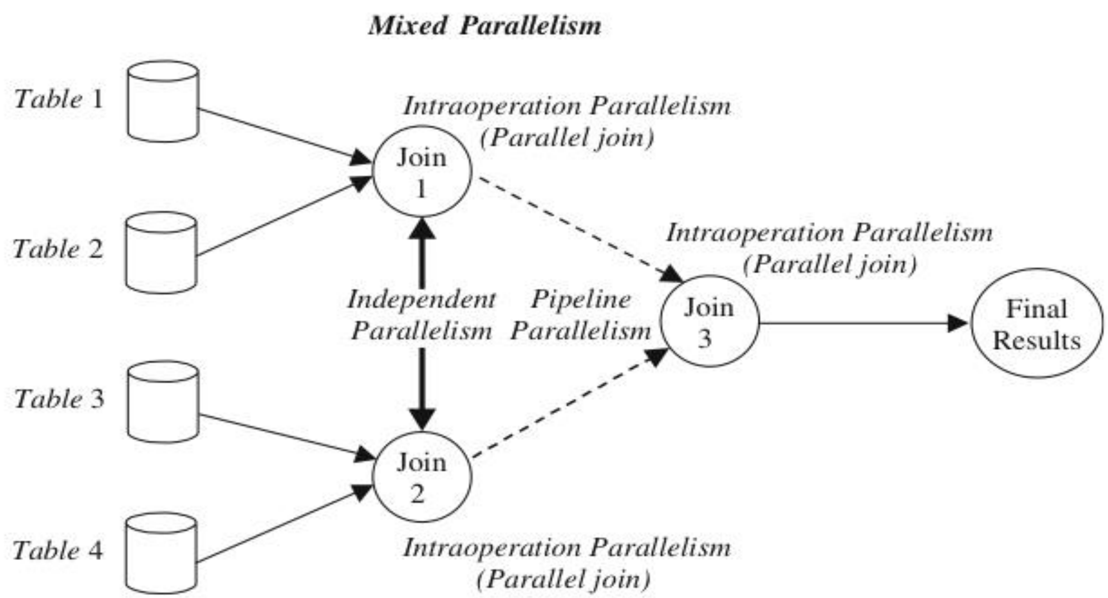


Figure 1.11 Mixed parallelism

1.5. Parallel Database Architectures

- Parallel computers are no longer a monopoly of supercomputers
- Parallel computers are available in many forms:
  - Shared-memory architecture
  - Shared-disk architecture
  - Shared-nothing architecture
  - Shared-something architecture

1.5. Parallel Database Architectures (cont'd)

- **Shared-Memory** and **Shared-Disk Architectures**
  - Shared-Memory: all processors share a common main memory and secondary memory
  - Load balancing is relatively easy to achieve, but suffer from memory and bus contention
  - Shared-Disk: all processors, each of which has its own local main memory, share the disks

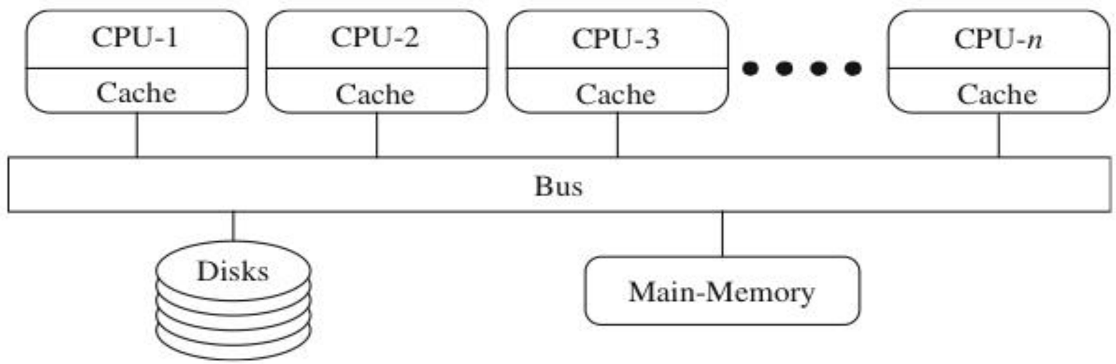


Figure 1.12 An SMP architecture

Load balancing: process of distributing a set of tasks over a set of computing units

1.5. Parallel Database Architectures (cont'd)

- **Shared-Nothing Architecture**
  - Each processor has its own local main memory and disks
  - Load balancing becomes difficult (because data is placed locally in each processor, and each processor may have an unequal load)

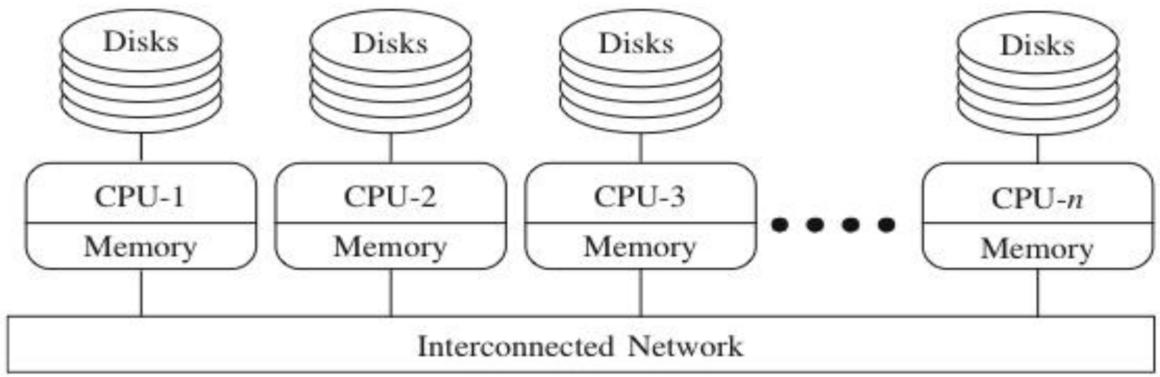


Figure 1.13 A shared-nothing architecture

1.5. Parallel Database Architectures (cont'd)

- **Shared-Something Architecture**
  - A mixture of shared-memory and shared-nothing architectures
  - Each node is a shared-memory architecture connected to an interconnection network aka shared-nothing architecture

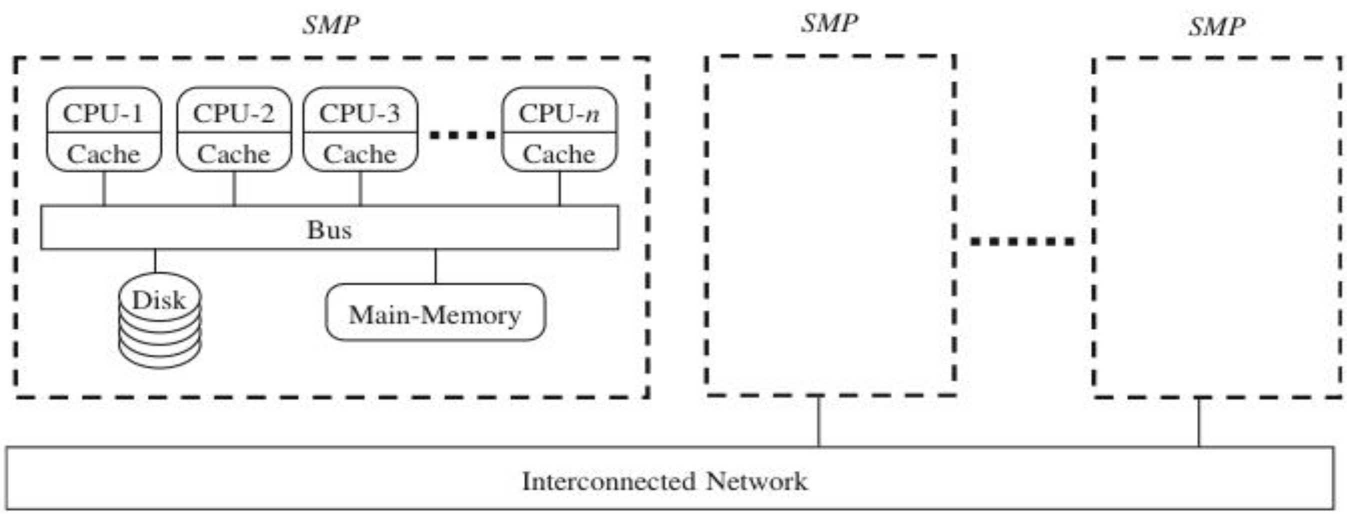


Figure 1.14 Cluster of SMP architectures

1.5. Parallel Database Architectures (cont'd)

- **Interconnection Networks**
  - Bus, Mesh, Hypercube

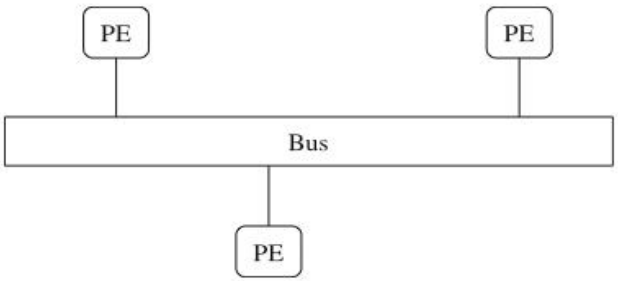


Figure 1.15 Bus interconnection network

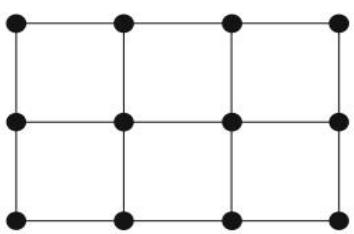


Figure 1.16 Mesh interconnection network

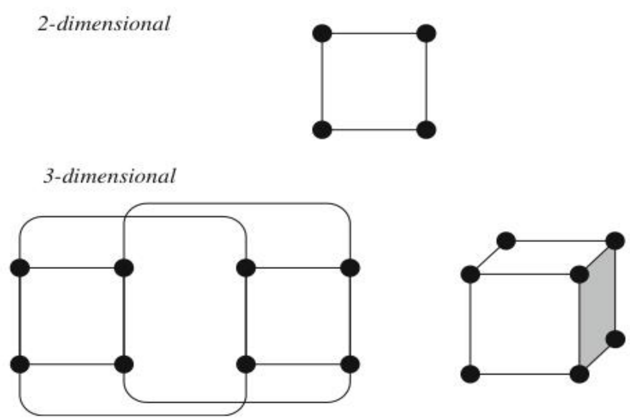


Figure 1.17 Hypercube interconnection network

1.8. Summary

- **Why**, **What**, and **How** of parallel query processing:
  - Why is parallelism necessary in database processing?
  - What can be achieved by parallelism in database processing?
  - How parallelism performed in database processing?
  - What facilities of parallel computing can be used?