UNIT V

Parallel and Distributed Databases

Introduction to Architectures

Multi-User DBMS Architectures

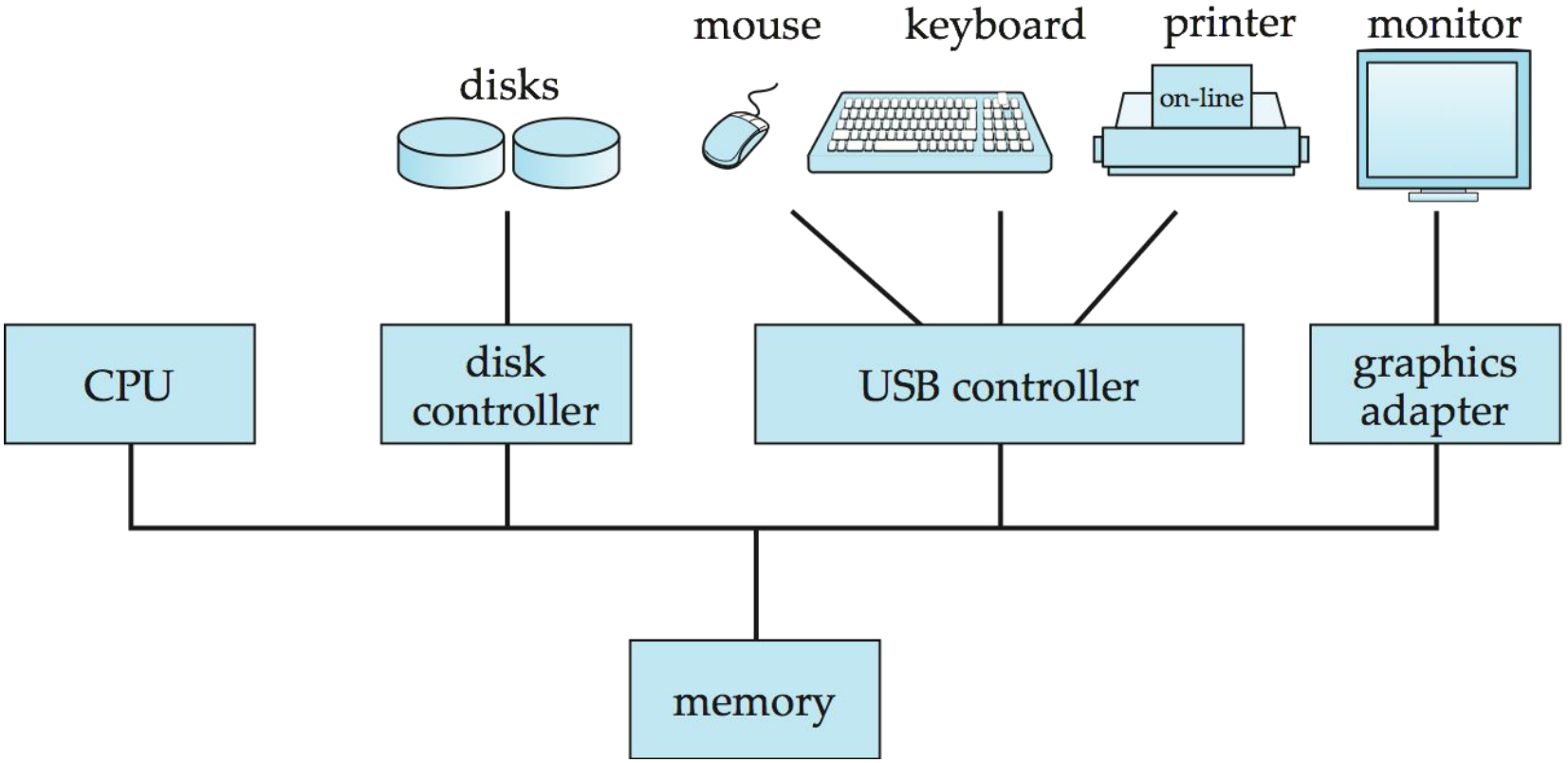
Database System Architectures

- Centralized and Client-Server Systems
- Server System Architectures
- Parallel Systems
- Distributed Systems
- Network Types

Centralized Systems

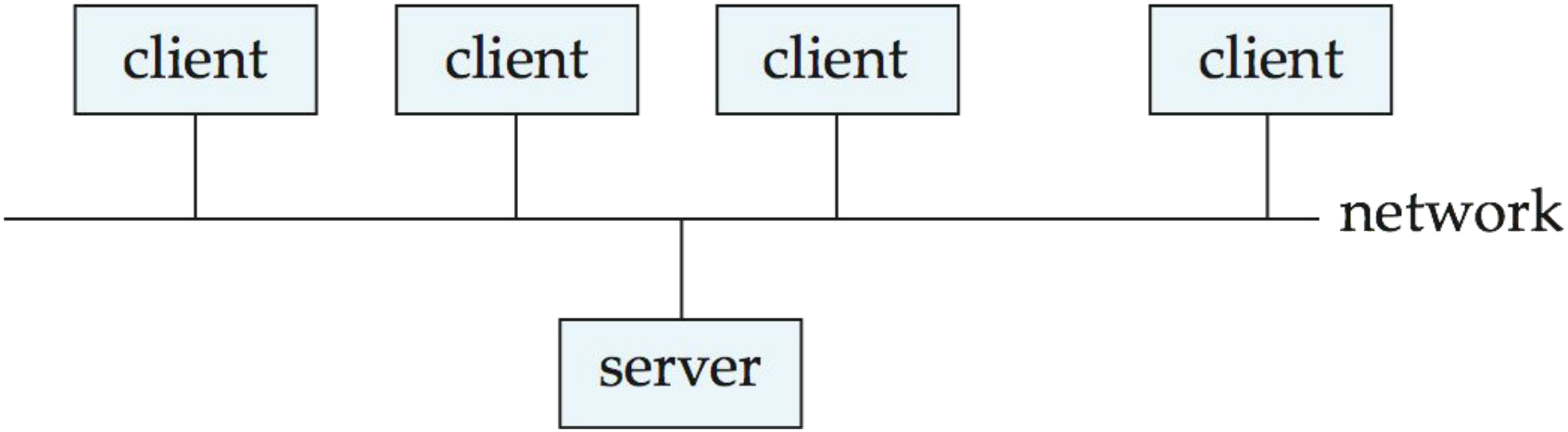
- Run on a single computer system and do not interact with other computer systems.
- General-purpose computer system: one to a few CPUs and a number of device controllers that are connected through a common bus that provides access to shared memory.
- Single-user system (e.g., personal computer or workstation): desk-top unit, single user, usually has only one CPU and one or two hard disks; the OS may support only one user.
- Multi-user system: more disks, more memory, multiple CPUs, and a multi-user OS. Serve a large number of users who are connected to the system vie terminals. Often called *server* systems.

A Centralized Computer System



Client-Server Systems

- Server systems satisfy requests generated at *m* client systems, whose general structure is shown below:



Client-Server Architectures

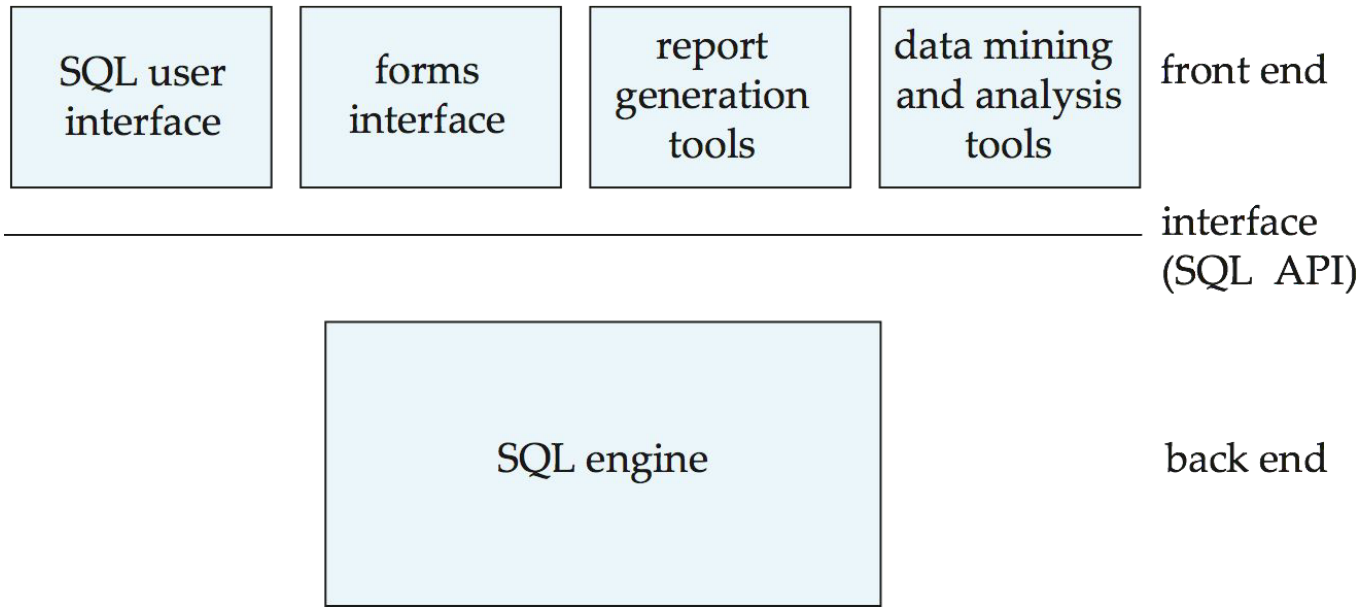
The **client/server architecture** was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, database servers, Web servers, e-mail servers, and other software and equipment are connected via a network. The idea is to define **specialized servers** with specific functionalities.

For example, it is possible to connect a number of PCs or small workstations as clients to a **file server** that maintains the files of the client machines. Another machine can be designated as a **printer server** by being connected to various printers; all print requests by the clients are forwarded to this machine.

Web servers or **e-mail servers** also fall into the specialized server category. The resources provided by specialized servers can be accessed by many client machines.

Client-Server Systems (Cont.)

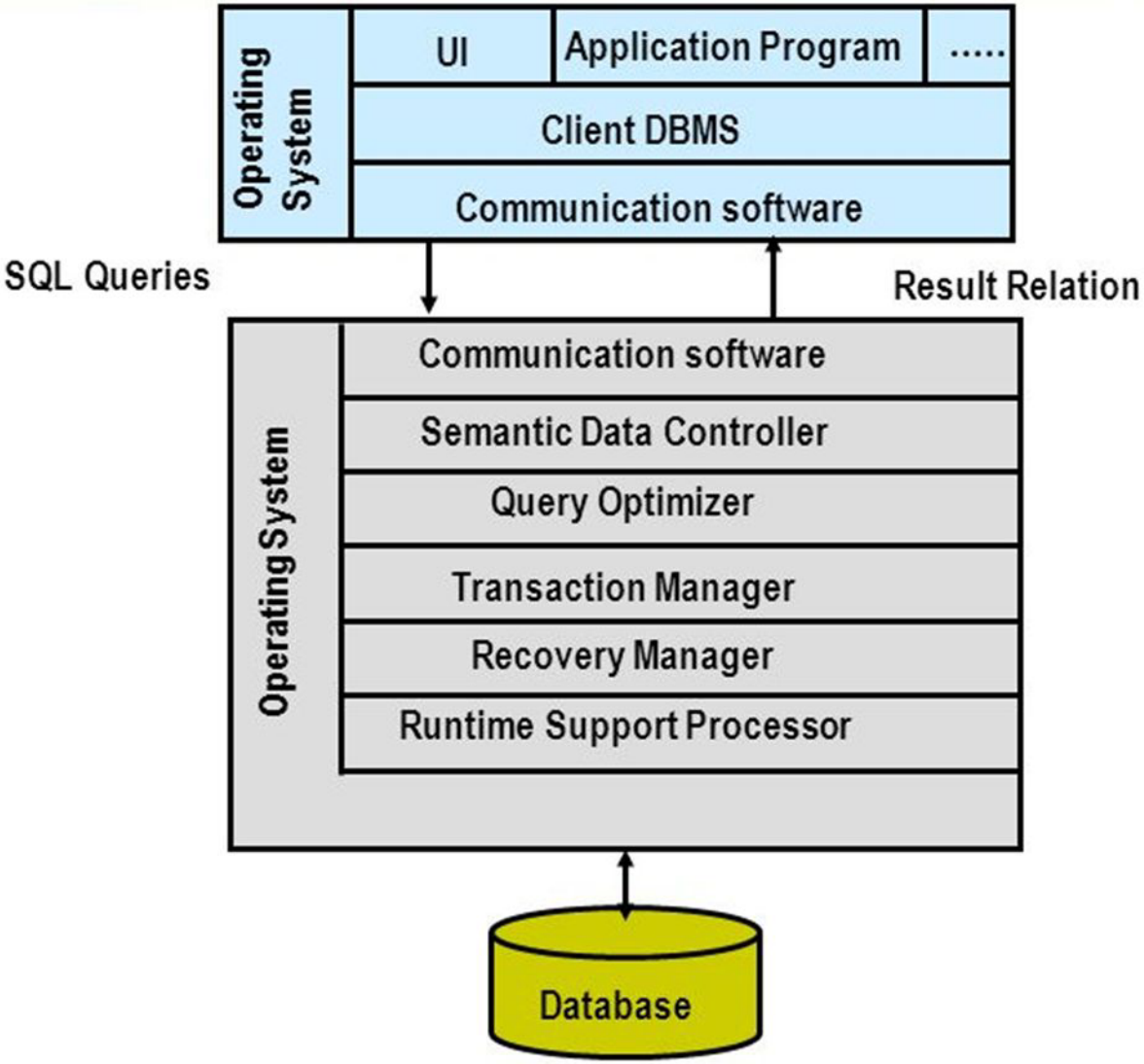
- Database functionality can be divided into:
  - **Back-end**: manages access structures, query evaluation and optimization, concurrency control and recovery.
  - **Front-end**: consists of tools such as *forms*, *report-writers*, and graphical user interface facilities.
- The interface between the front-end and the back-end is through SQL or through an application program interface.



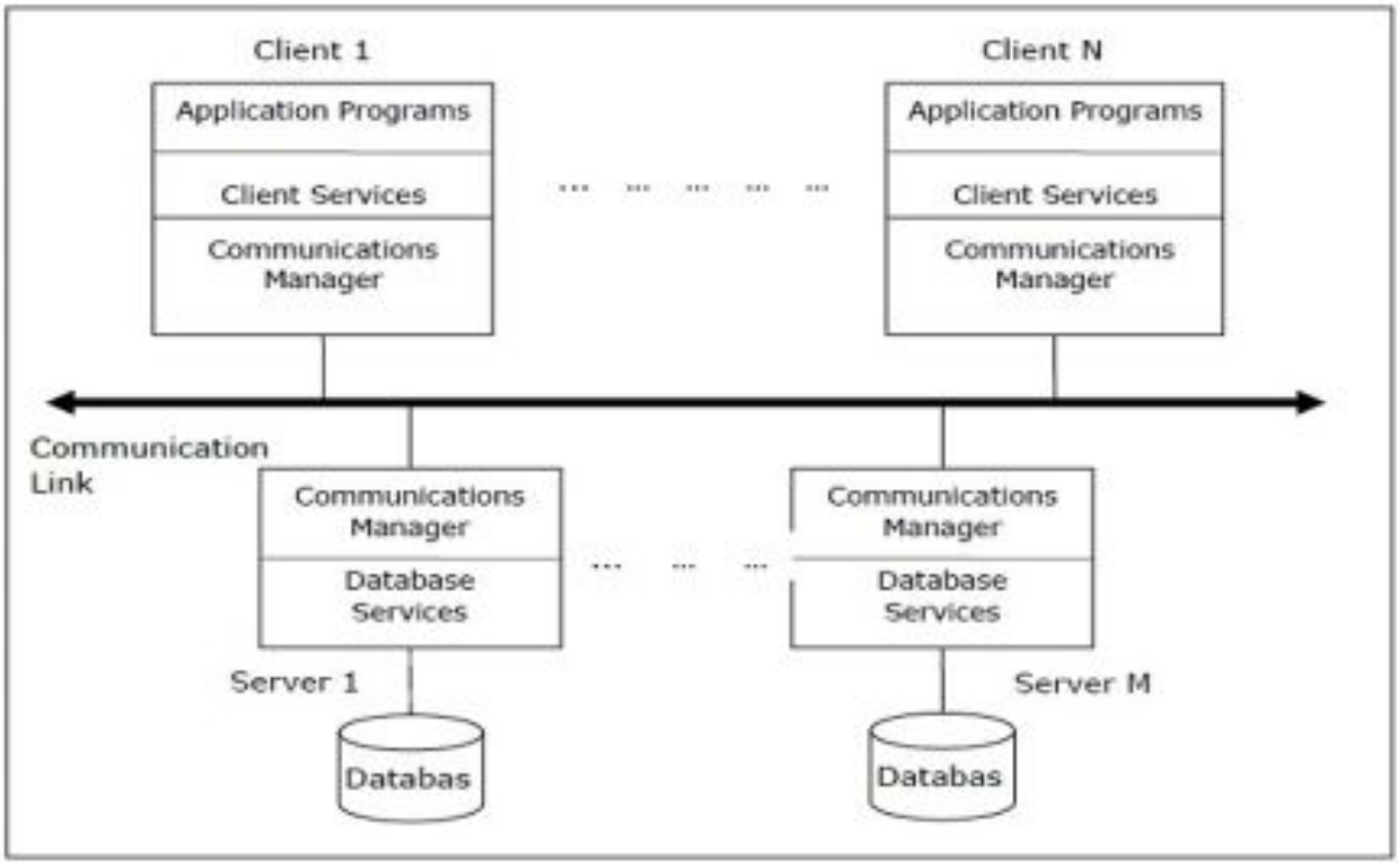
Client-Server Systems (Cont.)

- Advantages of replacing mainframes with networks of workstations or personal computers connected to back-end server machines:
  - better functionality for the cost
  - flexibility in locating resources and expanding facilities
  - better user interfaces
  - easier maintenance

Components of Client / Server Architecture



Multiple Server Multiple Client



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

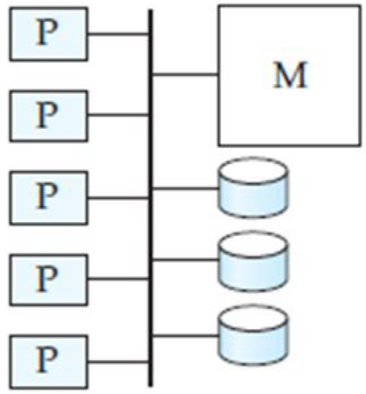
Parallel Databases

Parallel database system improves performance of data processing using multiple resources in parallel, like multiple CPU and disks are used in parallel.

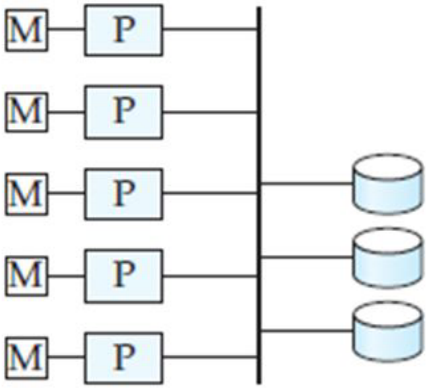
There are several architectural models for parallel machines.
These models are:

- **Shared memory**
- **Shared disk**.
- **Shared nothing**
- **Hierarchical**.

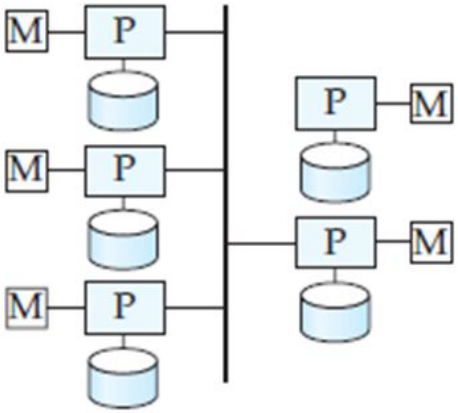
Parallel Database Architectures



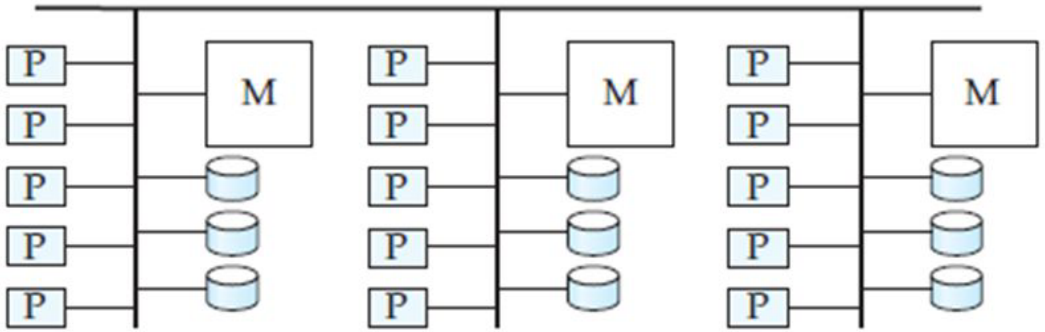
(a) shared memory



(b) shared disk



(c) shared nothing



(d) hierarchical

a)Shared-Memory :

In a **shared-memory** architecture, the processors and disks have access to a common memory, typically via a bus or through an interconnection network. The benefit of shared memory is extremely efficient communication between processors - data in shared memory can be accessed by any processor without being moved with software. A processor can send messages to other processors much faster by using memory writes (which usually take less than a microsecond) than by sending a message through a communication mechanism.

The downside of shared-memory machines is that the architecture is not scalable beyond 32 or 64 processors because the bus or the interconnection network becomes a bottleneck (since it is shared by all processors). Adding more processors does not help after a point, since the processors will spend most of their time waiting for their turn on the bus to access memory.

Shared-memory architectures usually have large memory caches at each processor, so that referencing of the shared memory is avoided whenever possible.

b) Shared Disk:

In the **shared-disk** model, all processors can access all disks directly via an interconnection network, but the processors have private memories.

There are two advantages of this architecture over shared-memory architecture:

1) Since each processor has its own memory, the memory bus is not a bottleneck.

2) It offers a cheap way to provide a degree of **fault tolerance**: If a processor (or its memory) fails, the other processors can take over its tasks, since the database is resident on disks that are accessible from all processors.

The shared-disk architecture has found acceptance in many applications.

The main problem with a shared-disk system is again scalability. Although the memory bus is no longer a bottleneck, the interconnection to the disk subsystem is now a bottleneck; it is particularly so in a situation where the database makes a large number of accesses to disks compared to shared-memory systems.

c) Shared Nothing:

In a **shared-nothing** system, each node of the machine consists of a processor, memory, and one or more disks. The processors at one node may communicate with another processor at another node by a high-speed interconnection network.

Shared-nothing architectures are more scalable and can easily support a large number of processors. The main drawbacks of shared-nothing systems are the costs of communication and of nonlocal disk access, which are higher than in shared-memory or shared-disk architecture since sending data involves software interaction at both ends.

d)Hierarchical:

The **hierarchical architecture** combines the characteristics of shared-memory, shared-disk, and shared-nothing architectures. At the top level, the system consists of nodes that are connected by an interconnection network and do not share disks or memory with one another. Thus, the top level is a shared-nothing architecture. Each node of the system could actually be a shared-memory system with a few processors. Alternatively, each node could be a shared-disk system, and each of the systems sharing a set of disks could be a shared-memory system. Thus, a system could be built as a hierarchy, with shared-memory architecture with a few processors at the base, and a shared-nothing architecture at the top, with possibly a shared-disk architecture in the middle.

Parallel Systems

- Parallel database systems consist of multiple processors and multiple disks connected by a fast interconnection network.
- A **coarse-grain parallel** machine consists of a small number of powerful processors
- A **massively parallel** or **fine grain parallel** machine utilizes thousands of smaller processors.
- Two main performance measures:
  - **throughput** --- the number of tasks that can be completed in a given time interval
  - **response time** --- the amount of time it takes to complete a single task from the time it is submitted

Speed-Up and Scale-Up

- **Speedup**: a fixed-sized problem executing on a small system is given to a system which is *N*-times larger.
  - Measured by:

    $$speedup = \frac{small\ system\ elapsed\ time}{large\ system\ elapsed\ time}$$

  - Speedup is **linear** if equation equals N.
- **Scaleup**: increase the size of both the problem and the system
  - *N*-times larger system used to perform *N*-times larger job
  - Measured by:

    $$scaleup = \frac{small\ system\ small\ problem\ elapsed\ time}{big\ system\ big\ problem\ elapsed\ time}$$

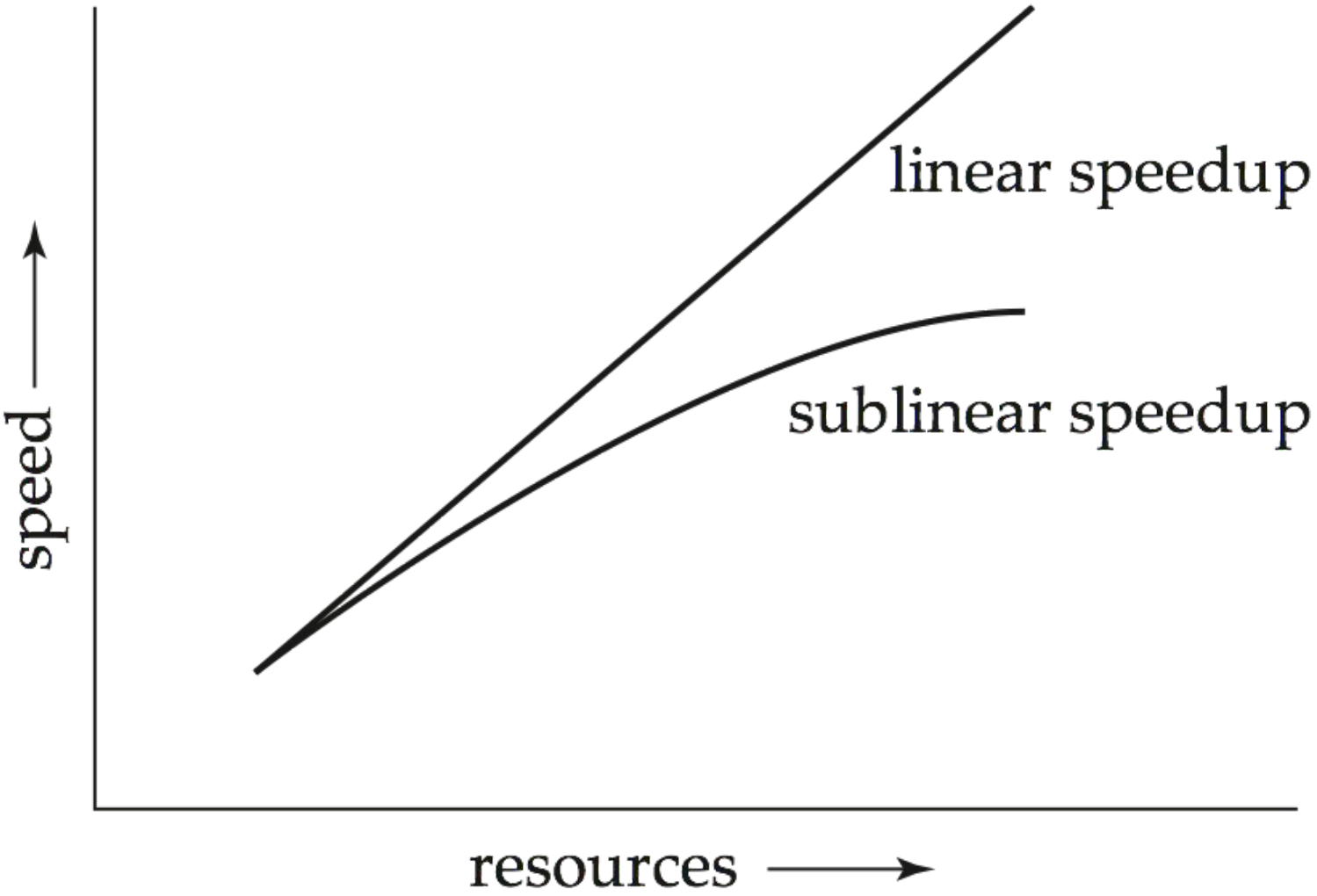
  - Scale up is **linear** if equation equals 1.

Running a given task in less time by increasing the degree of parallelism is called speedup

Consider a database application running on a parallel system with a certain number of processors and disks. Now suppose that we increase the size of the system by increasing the number or processors, disks, and other components of the system. The goal is to process the task in time inversely proportional to the number of processors and disks allocated. Suppose that the execution time of a task on the larger machine is TL, and that the execution time of the same task on the smaller machine is TS.

The speedup due to parallelism is defined as TS/TL. The parallel system is said to demonstrate linear speedup if the speedup is N when the larger system has N times the resources (CPU, disk, and so on) of the smaller system. If the speedup is less than N, the system is said to demonstrate sublinear speedup. Figure illustrates linear and sublinear speedup..

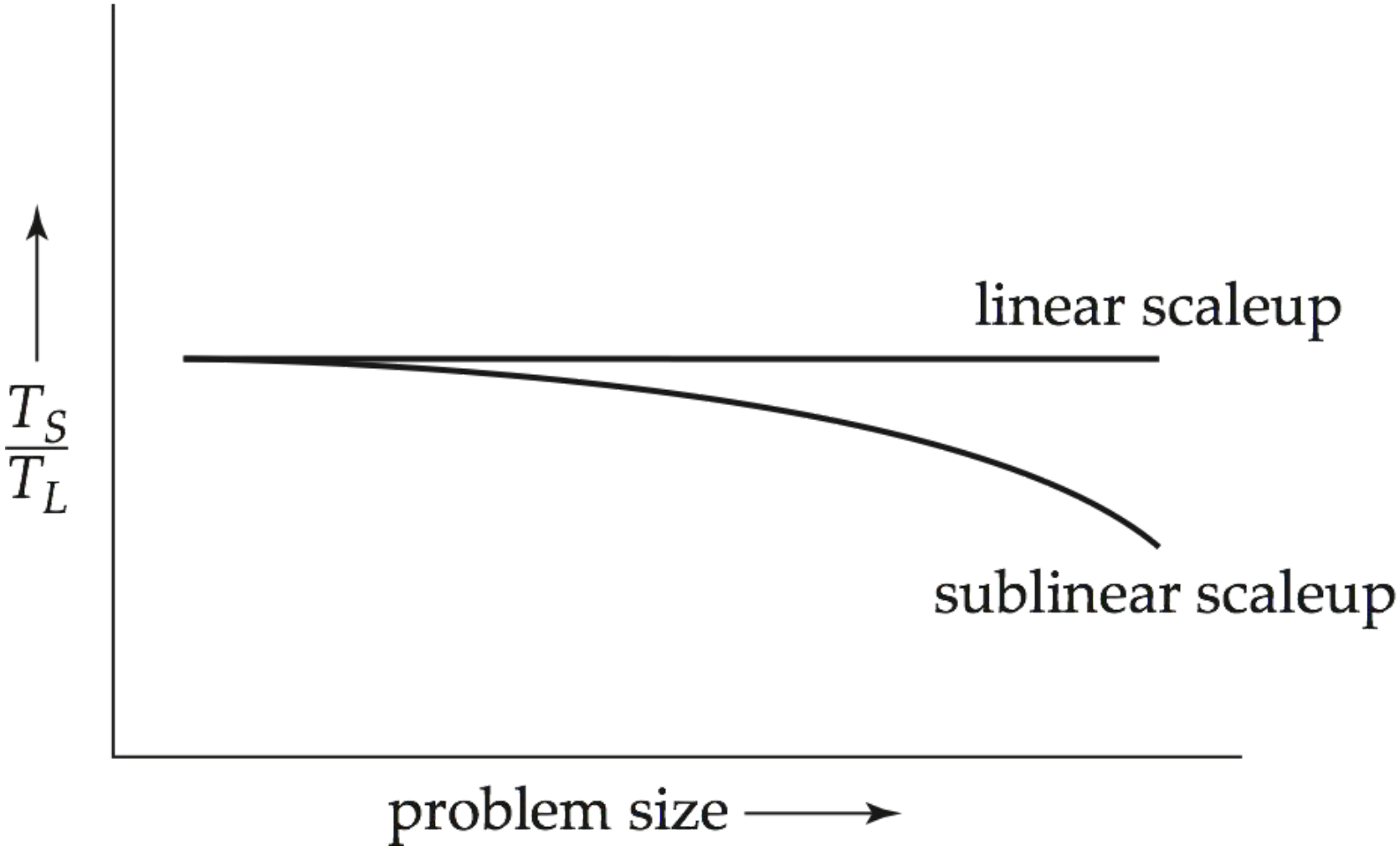
Speedup



Handling larger tasks by increasing the degree of parallelism is called **scaleup**.

Scaleup relates to the ability to process larger tasks in the same amount of time by providing more resources. Let Q be a task, and let QN be a task that is N times bigger than Q. Suppose that the execution time of task Q on a given machine MS is TS, and the execution time of task QN on a parallel machineML, which is N times larger than MS, is TL. The scaleup is then defined as TS/TL. The parallel system ML is said todemonstrate linear scaleup on task Q if TL = TS. If TL > TS, the system is said to demonstrate sublinear scaleup.

Scaleup



Batch and Transaction Scaleup

- **Batch scaleup:**
  - A single large job; typical of most decision support queries and scientific simulation.
  - Use an *N*-times larger computer on *N*-times larger problem.
- **Transaction scaleup**:
  - Numerous small queries submitted by independent users to a shared database; typical transaction processing and timesharing systems.
  - *N*-times as many users submitting requests (hence, *N*-times as many requests) to an *N*-times larger database, on an *N*-times larger computer.
  - Well-suited to parallel execution.

Factors Limiting Speedup and Scaleup

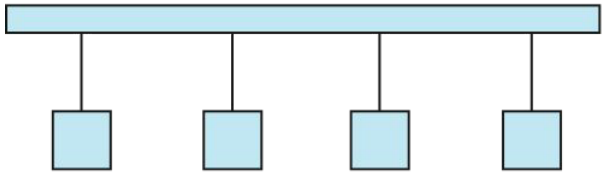
Speedup and scaleup are often sublinear due to:

- **Startup costs**: Cost of starting up multiple processes may dominate computation time, if the degree of parallelism is high.
- **Interference**: Processes accessing shared resources (e.g., system bus, disks, or locks) compete with each other, thus spending time waiting on other processes, rather than performing useful work.
- **Skew**: Increasing the degree of parallelism increases the variance in service times of parallely executing tasks. Overall execution time determined by **slowest** of parallely executing tasks.

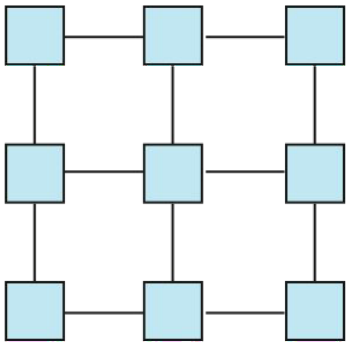
Interconnection Network Architectures

- **Bus**. System components send data on and receive data from a single communication bus;
  - Does not scale well with increasing parallelism.
- **Mesh**. Components are arranged as nodes in a grid, and each component is connected to all adjacent components
  - Communication links grow with growing number of components, and so scales better.
  - But may require $2\sqrt{n}$ hops to send message to a node (or $\sqrt{n}$ with wraparound connections at edge of grid).
- **Hypercube**. Components are numbered in binary; components are connected to one another if their binary representations differ in exactly one bit.
  - *n* components are connected to *log(n)* other components and can reach each other via at most *log(n)* links; reduces communication delays.

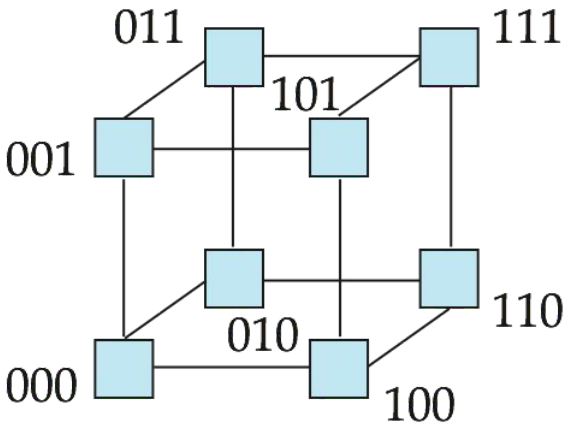
Interconnection Architectures



(a) bus



(b) mesh

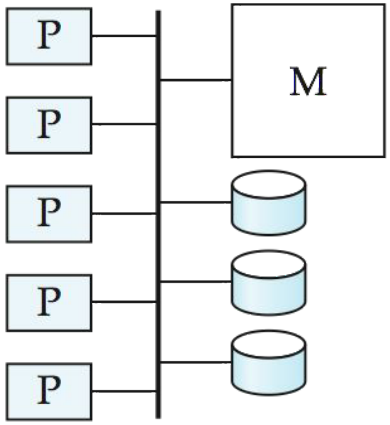


(c) hypercube

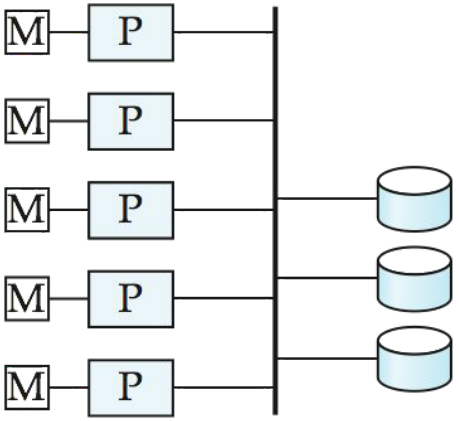
Parallel Database Architectures

- **Shared memory** -- processors share a common memory
- **Shared disk** -- processors share a common disk
- **Shared nothing** -- processors share neither a common memory nor common disk
- **Hierarchical** -- hybrid of the above architectures

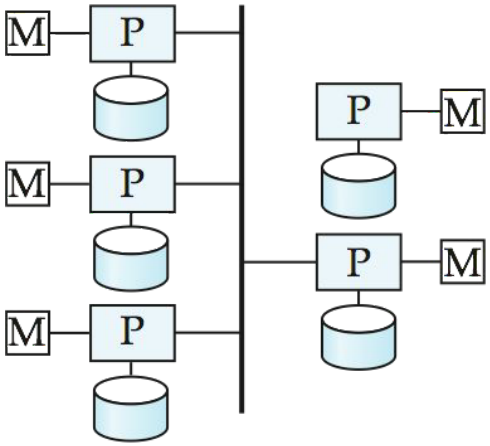
Parallel Database Architectures



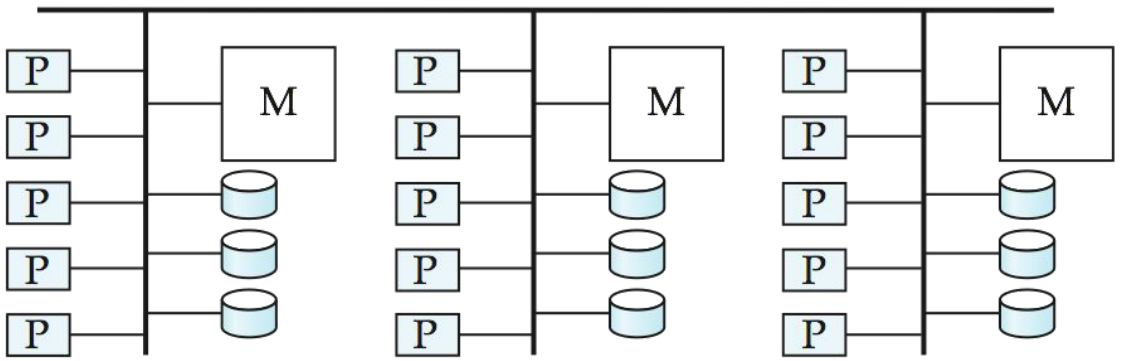
(a) shared memory



(b) shared disk



(c) shared nothing



(d) hierarchical

Shared Memory

- Processors and disks have access to a common memory, typically via a bus or through an interconnection network.
- Extremely efficient communication between processors — data in shared memory can be accessed by any processor without having to move it using software.
- Downside – architecture is not scalable beyond 32 or 64 processors since the bus or the interconnection network becomes a bottleneck
- Widely used for lower degrees of parallelism (4 to 8).

Shared Disk

- All processors can directly access all disks via an interconnection network, but the processors have private memories.
  - The memory bus is not a bottleneck
  - Architecture provides a degree of **fault-tolerance** — if a processor fails, the other processors can take over its tasks since the database is resident on disks that are accessible from all processors.
- Examples: IBM Sysplex and DEC clusters (now part of Compaq) running Rdb (now Oracle Rdb) were early commercial users
- Downside: bottleneck now occurs at interconnection to the disk subsystem.
- Shared-disk systems can scale to a somewhat larger number of processors, but communication between processors is slower.

Shared Nothing

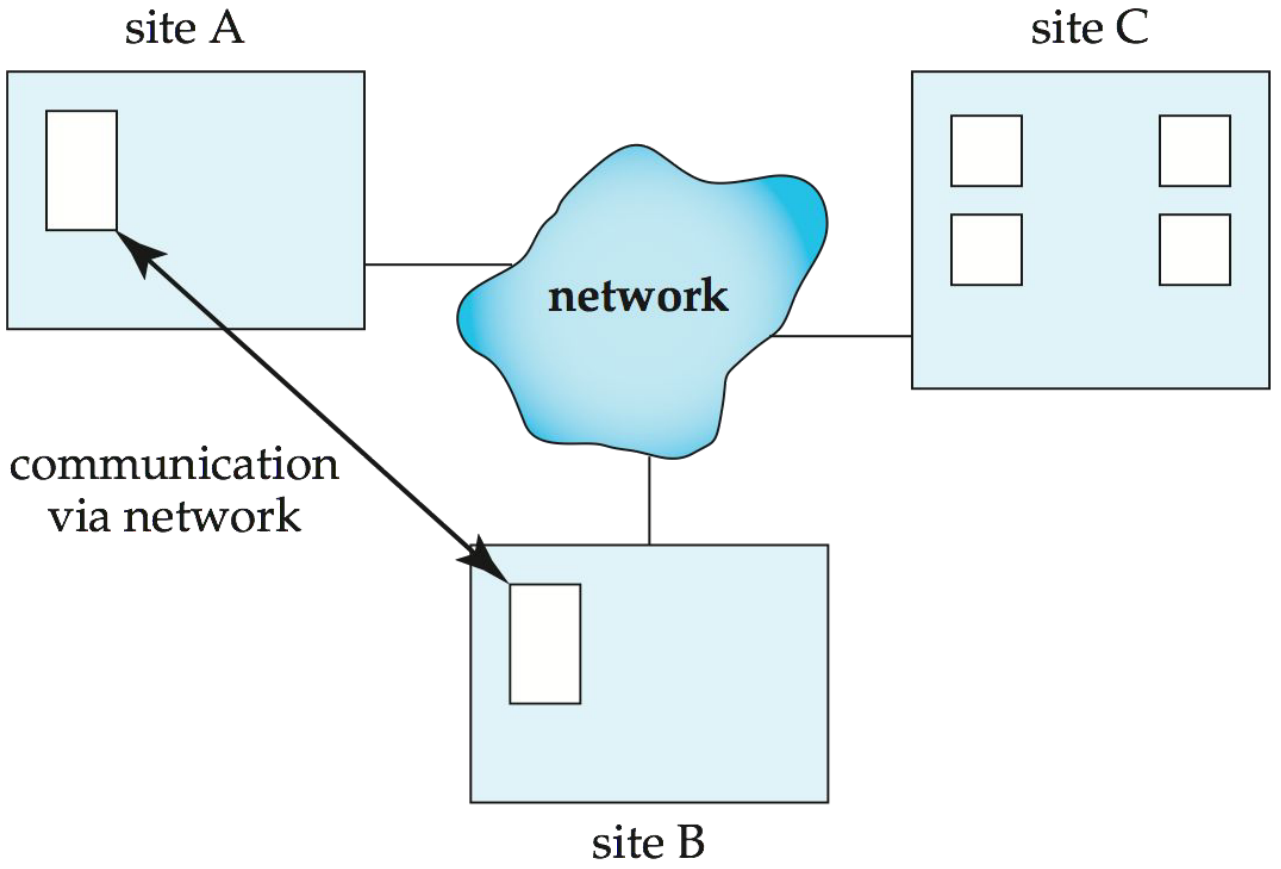
- Node consists of a processor, memory, and one or more disks. Processors at one node communicate with another processor at another node using an interconnection network. A node functions as the server for the data on the disk or disks the node owns.
- Examples: Teradata, Tandem, Oracle-n CUBE
- Data accessed from local disks (and local memory accesses) do not pass through interconnection network, thereby minimizing the interference of resource sharing.
- Shared-nothing multiprocessors can be scaled up to thousands of processors without interference.
- Main drawback: cost of communication and non-local disk access; sending data involves software interaction at both ends.

Hierarchical

- Combines characteristics of shared-memory, shared-disk, and shared-nothing architectures.
- Top level is a shared-nothing architecture – nodes connected by an interconnection network, and do not share disks or memory with each other.
- Each node of the system could be a shared-memory system with a few processors.
- Alternatively, each node could be a shared-disk system, and each of the systems sharing a set of disks could be a shared-memory system.
- Reduce the complexity of programming such systems by **distributed virtual-memory** architectures
  - Also called **non-uniform memory architecture (NUMA)**

Distributed Systems

- Data spread over multiple machines (also referred to as **sites** or **nodes**).
- Network interconnects the machines
- Data shared by users on multiple machines



Distributed Databases

- Homogeneous distributed databases
  - Same software/schema on all sites, data may be partitioned among sites
  - Goal: provide a view of a single database, hiding details of distribution
- Heterogeneous distributed databases
  - Different software/schema on different sites
  - Goal: integrate existing databases to provide useful functionality
- Differentiate between *local* and *global* transactions
  - A **local transaction** accesses data in the *single* site at which the transaction was initiated.
  - A **global transaction** either accesses data in a site different from the one at which the transaction was initiated or accesses data in several different sites.

Trade-offs in Distributed Systems

- Sharing data – users at one site able to access the data residing at some other sites.
- Autonomy – each site is able to retain a degree of control over data stored locally.
- Higher system availability through redundancy — data can be replicated at remote sites, and system can function even if a site fails.
- Disadvantage: added complexity required to ensure proper coordination among sites.
  - Software development cost.
  - Greater potential for bugs.
  - Increased processing overhead.

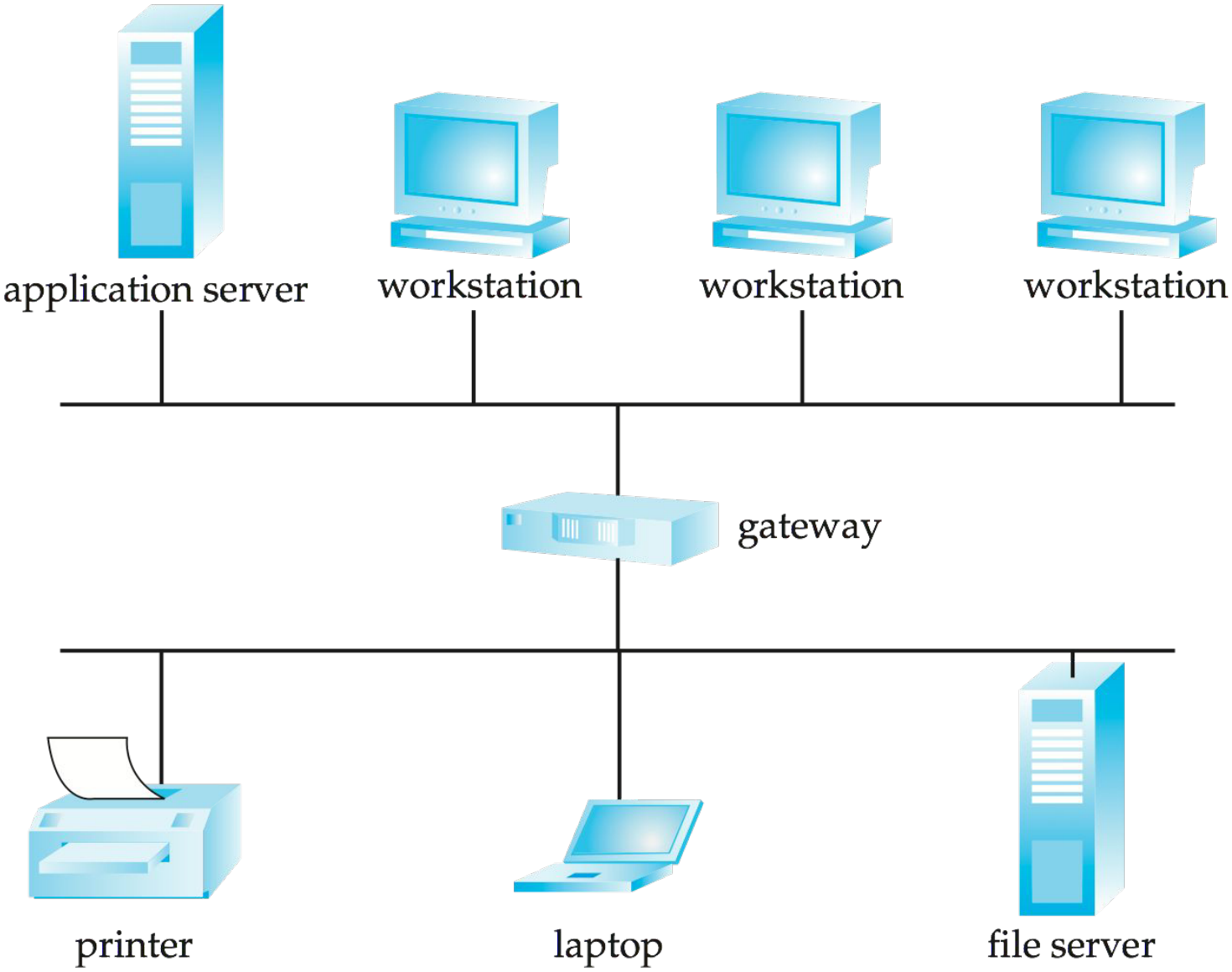
Implementation Issues for Distributed Databases

- Atomicity needed even for transactions that update data at multiple sites
- The two-phase commit protocol (2PC) is used to ensure atomicity
  - Basic idea: each site executes transaction until just before commit, and the leaves final decision to a coordinator
  - Each site must follow decision of coordinator, even if there is a failure while waiting for coordinators decision
- 2PC is not always appropriate: other transaction models based on persistent messaging, and workflows, are also used
- Distributed concurrency control (and deadlock detection) required
- Data items may be replicated to improve data availability
- Details of above in Chapter 22

Network Types

- **Local-area networks (**LANs) – composed of processors that are distributed over small geographical areas, such as a single building or a few adjacent buildings.
- **Wide-area networks (**WANs) – composed of processors distributed over a large geographical area.

Local-area Network



Networks Types (Cont.)

- WANs with continuous connection (e.g., the Internet) are needed for implementing distributed database systems
- Groupware applications such as Lotus notes can work on WANs with discontinuous connection:
  - Data is replicated.
  - Updates are propagated to replicas periodically.
  - Copies of data may be updated independently.
  - Non-serializable executions can thus result. Resolution is application dependent.

End of Chapter 17

Figure 17.01

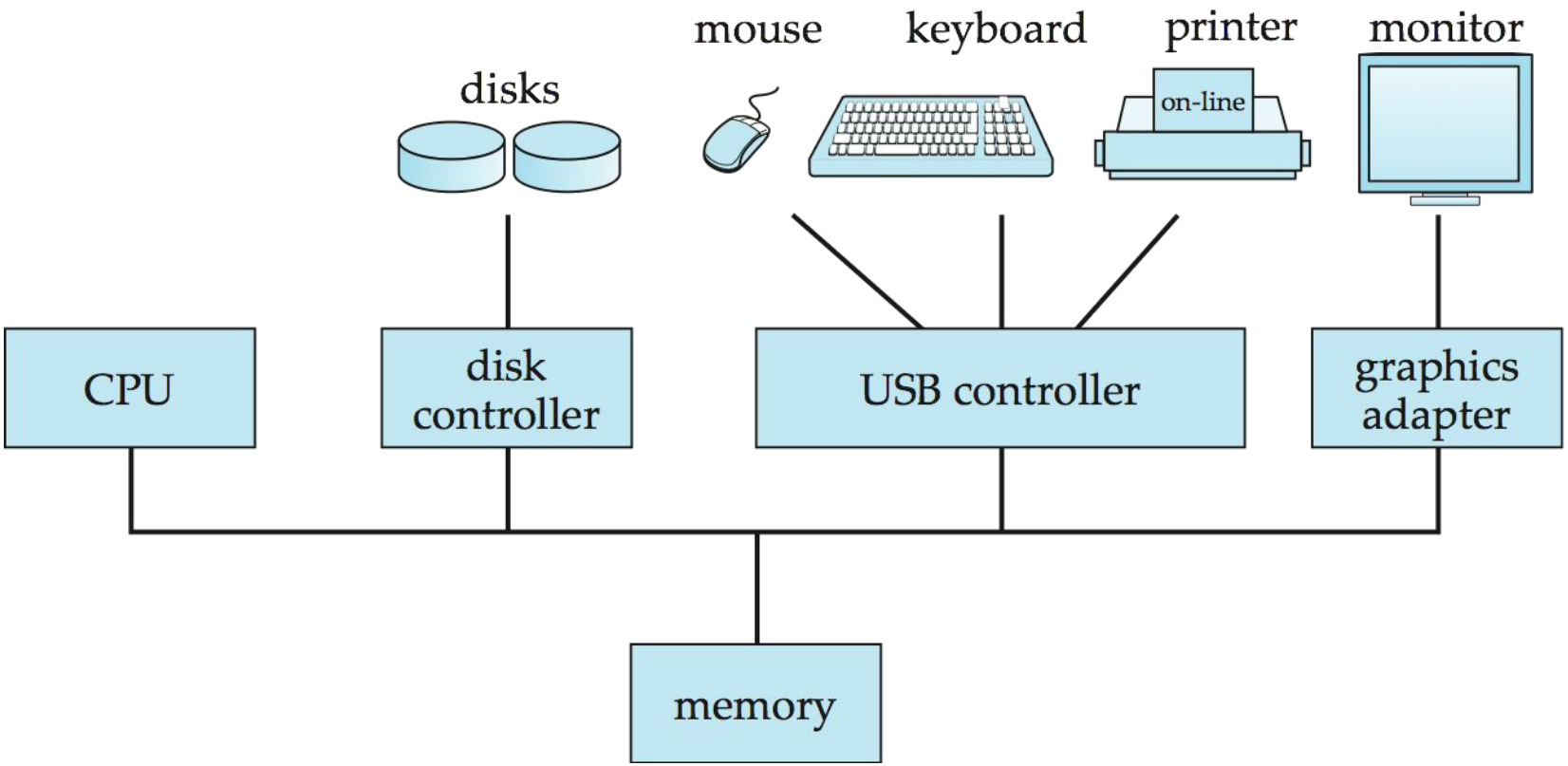


Figure 17.02

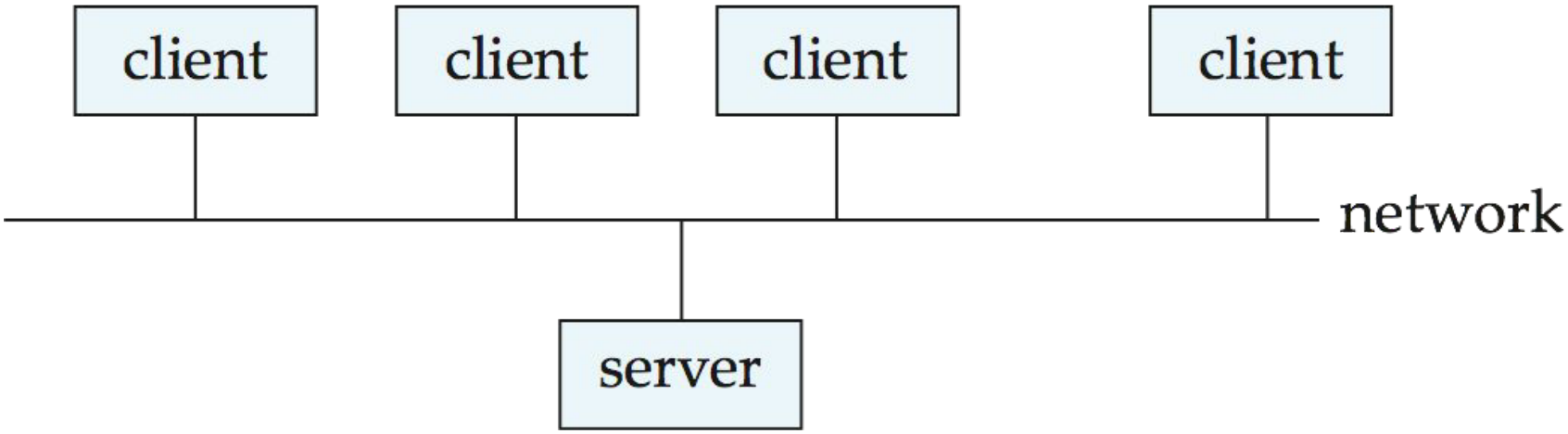


Figure 17.03

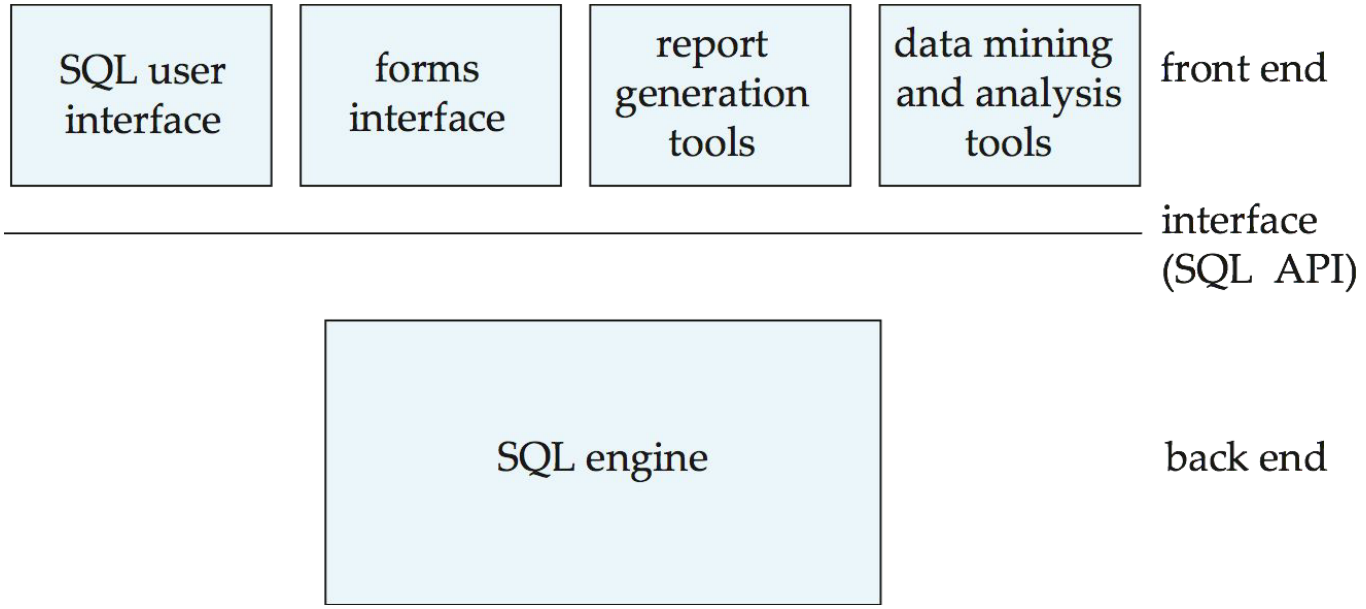

SQL user interface
forms interface
report generation tools
data mining and analysis tools
front end
interface (SQL API)
SQL engine
back end

Figure 17.04

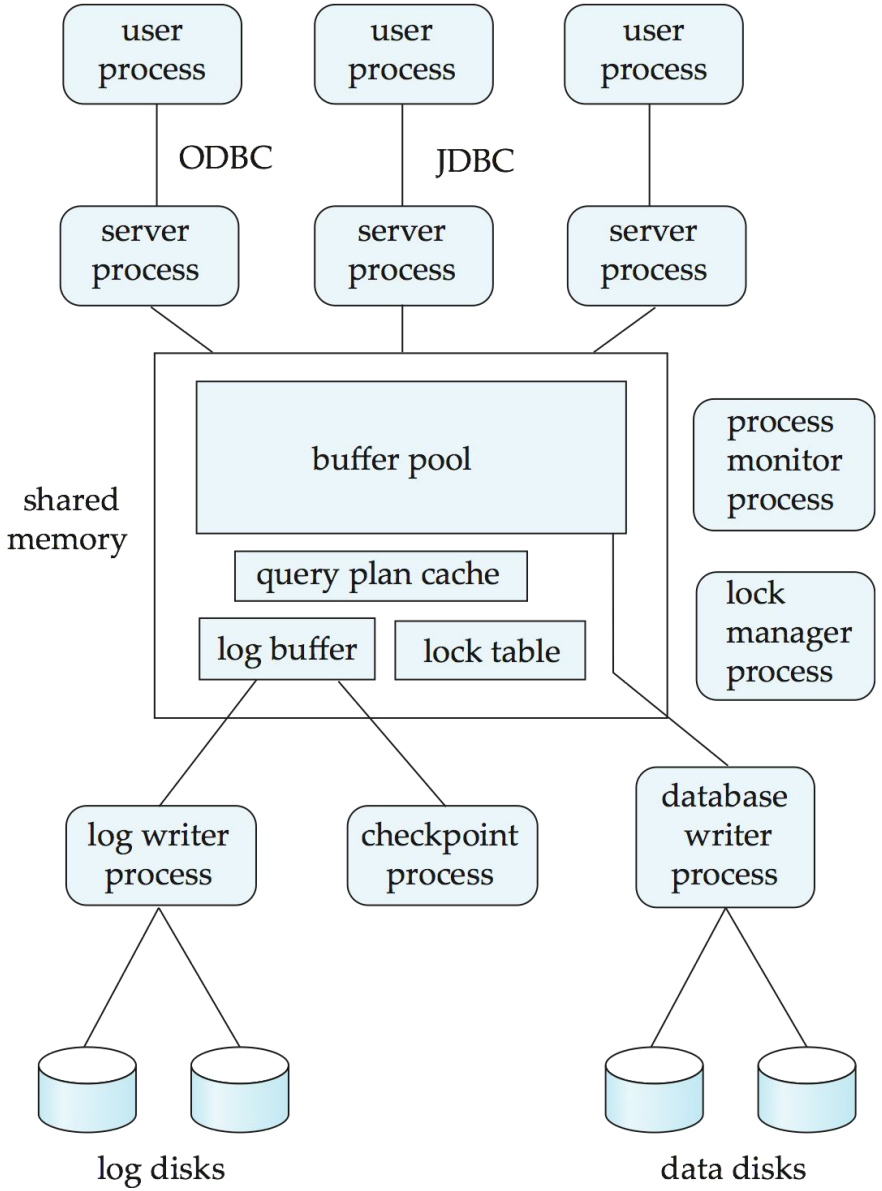


Figure 17.05

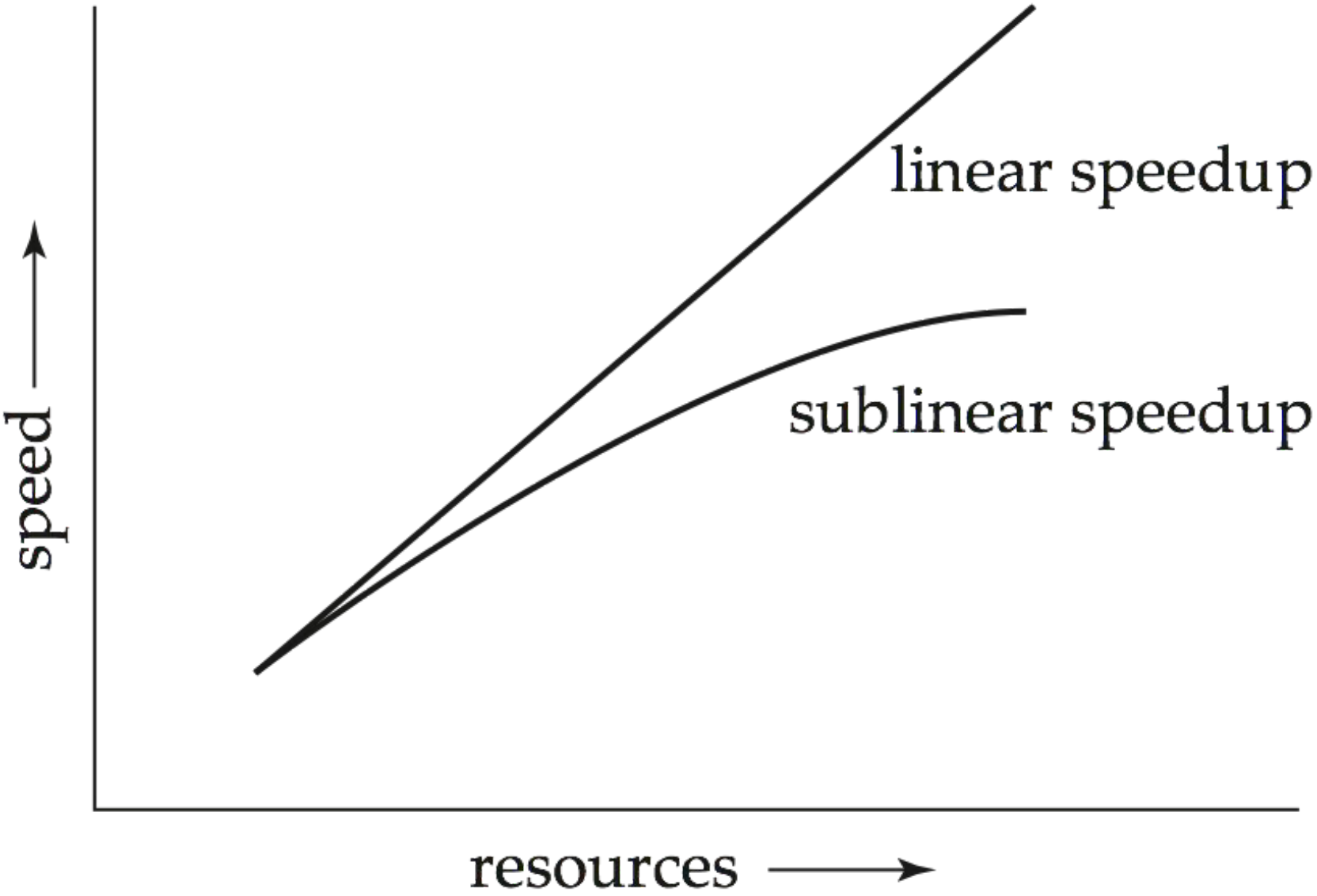


Figure 17.06

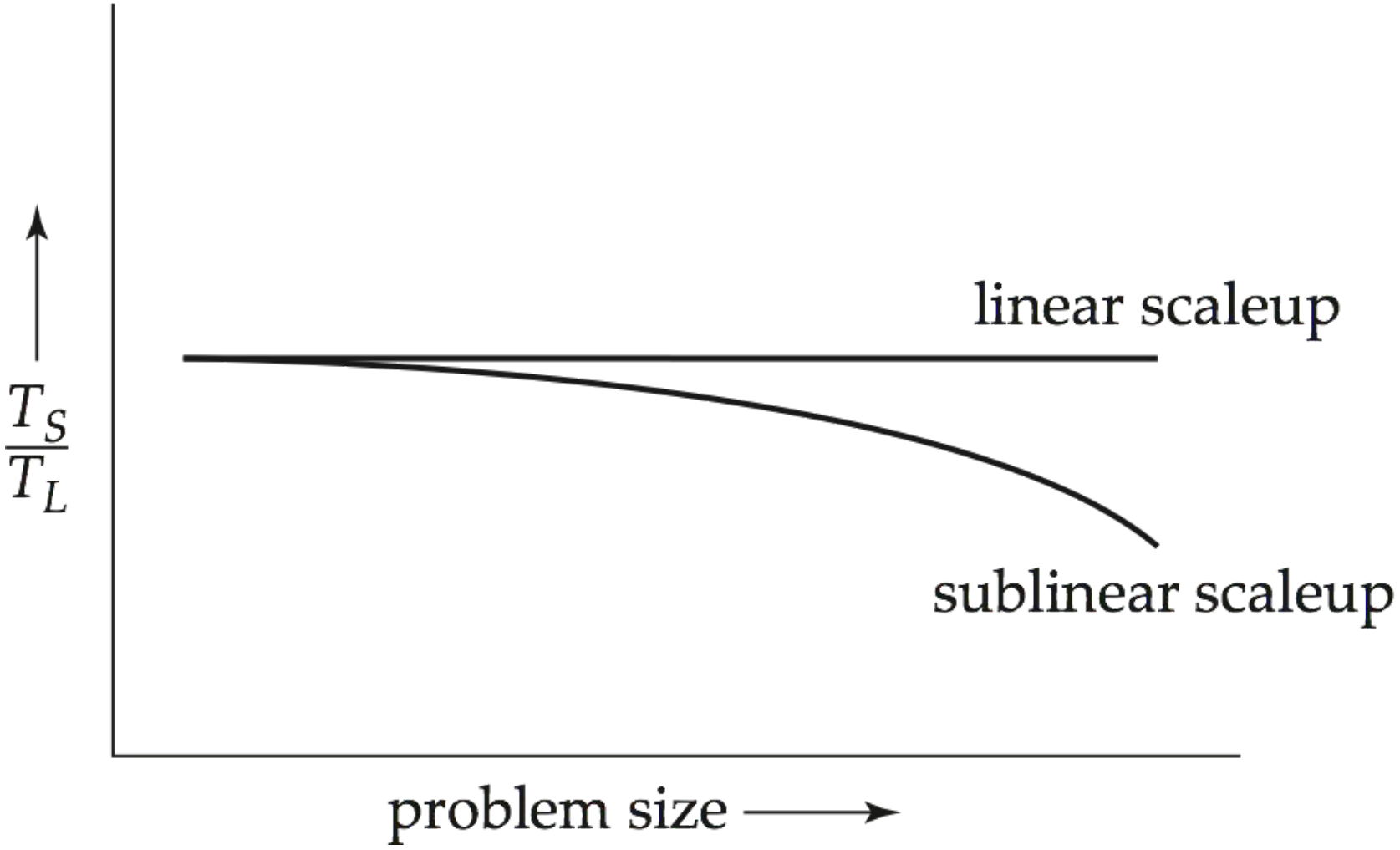
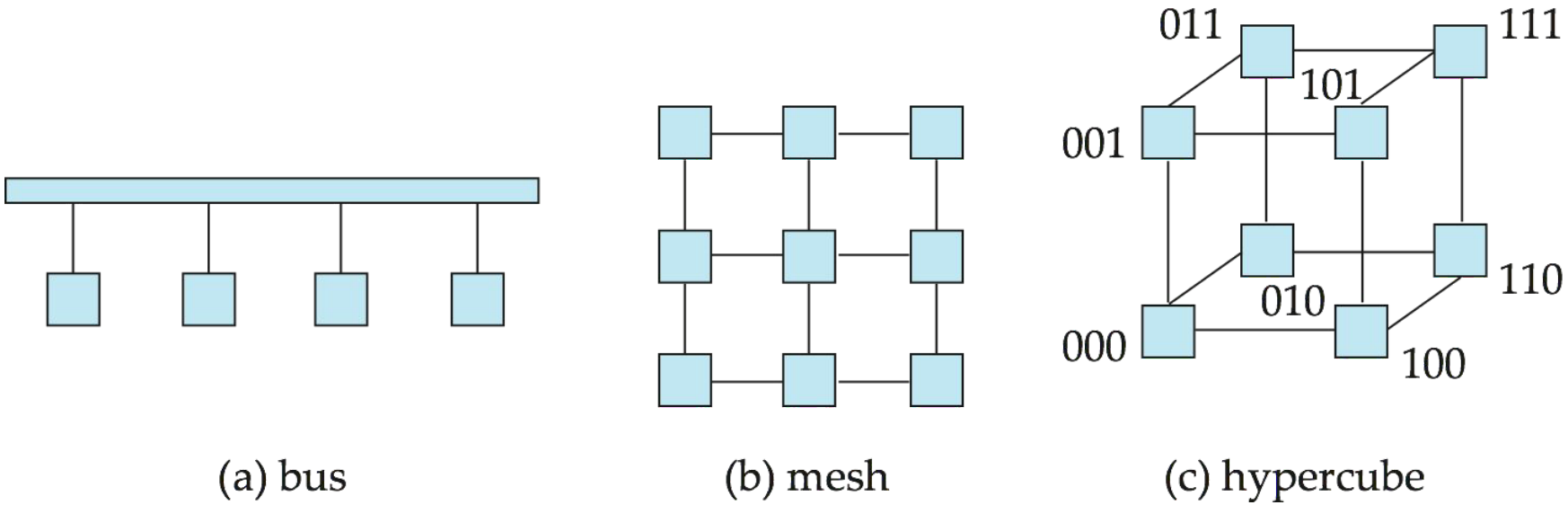
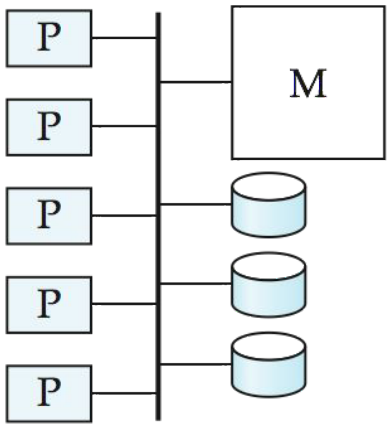


Figure 17.07

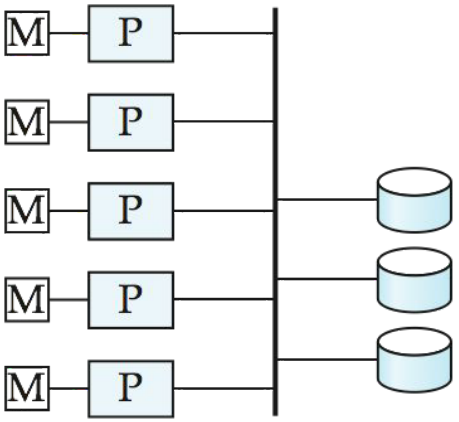


(a) bus (b) mesh (c) hypercube

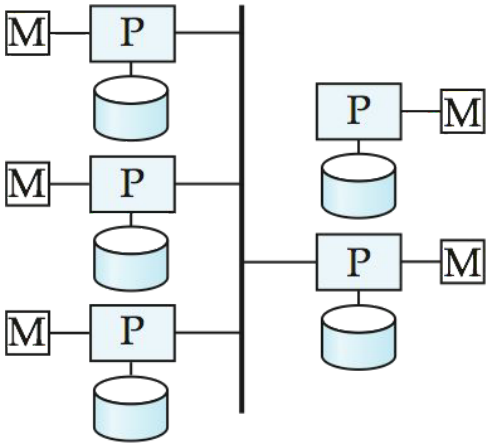
Figure 17.08



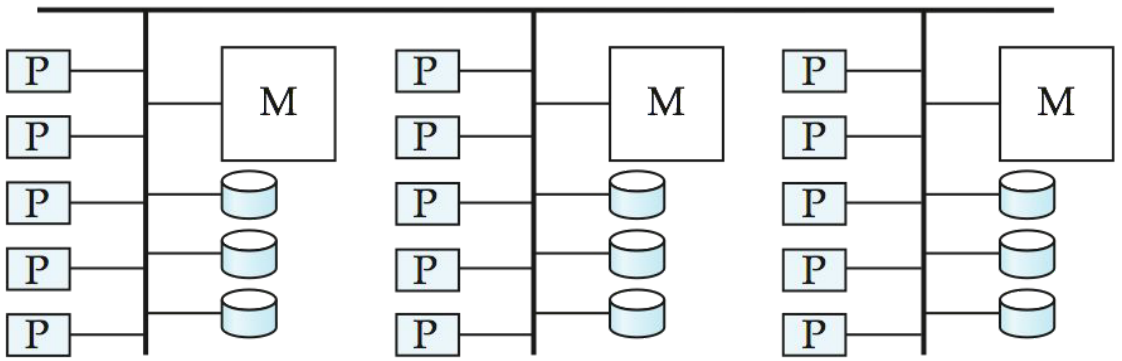
(a) shared memory



(b) shared disk



(c) shared nothing



(d) hierarchical

Figure 17.09

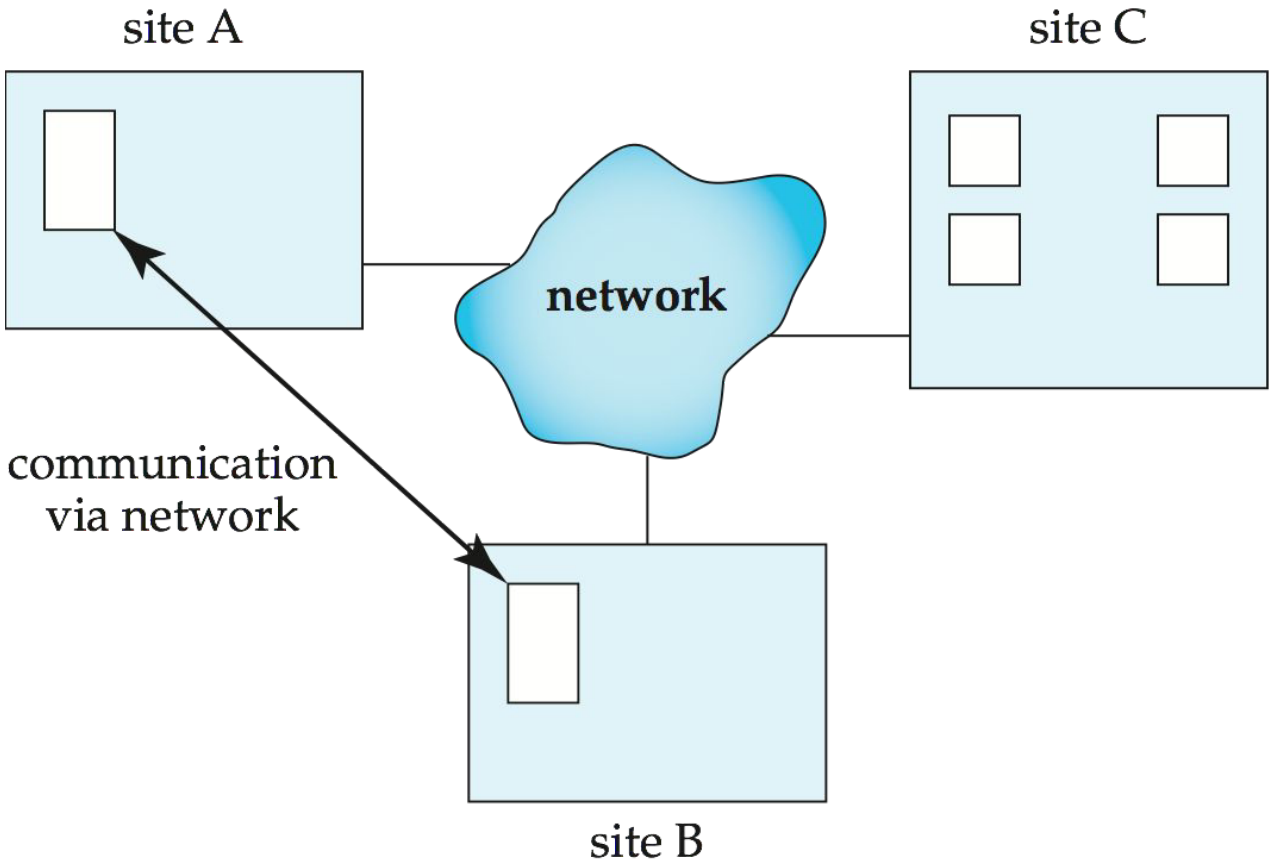


Figure 17.10

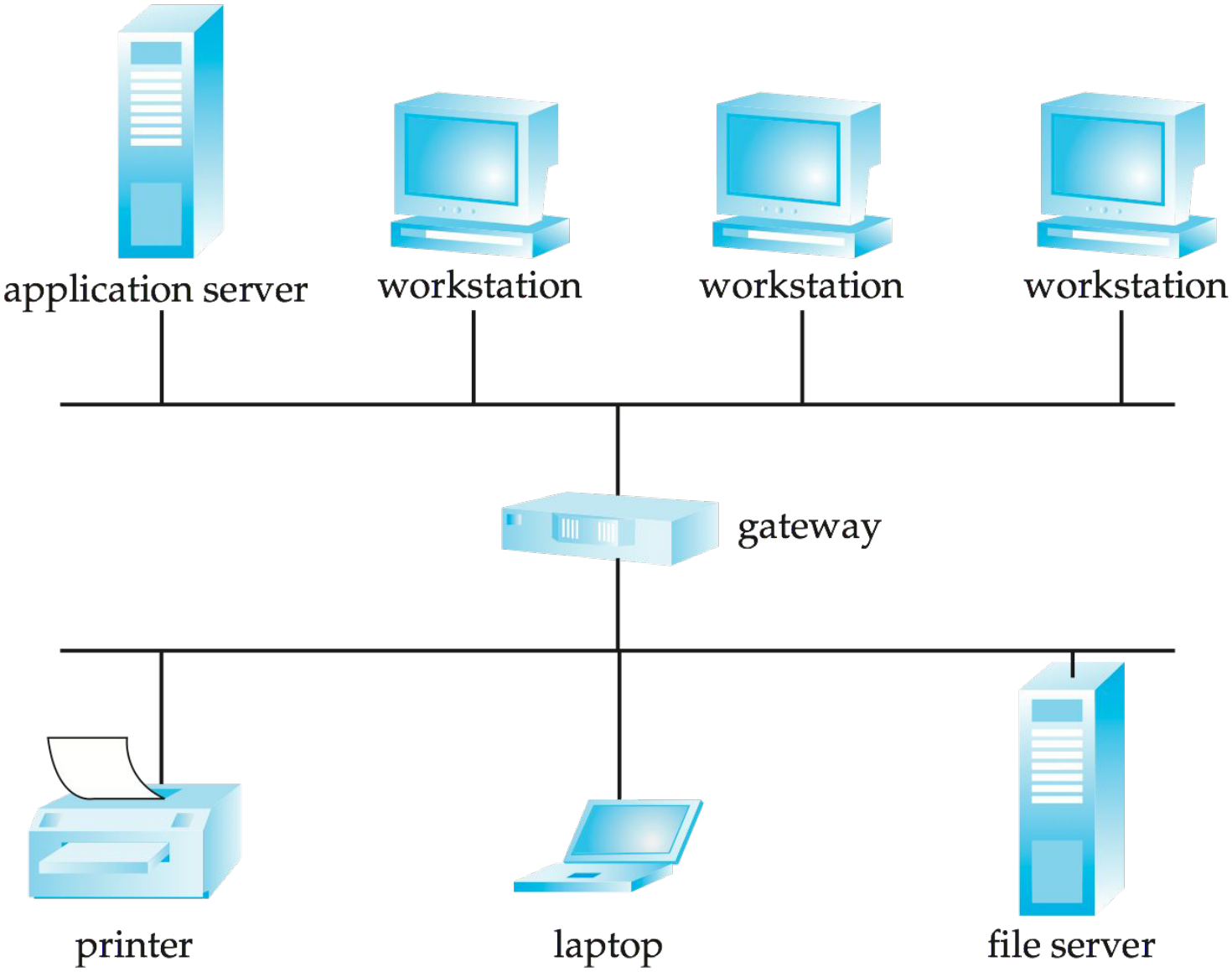
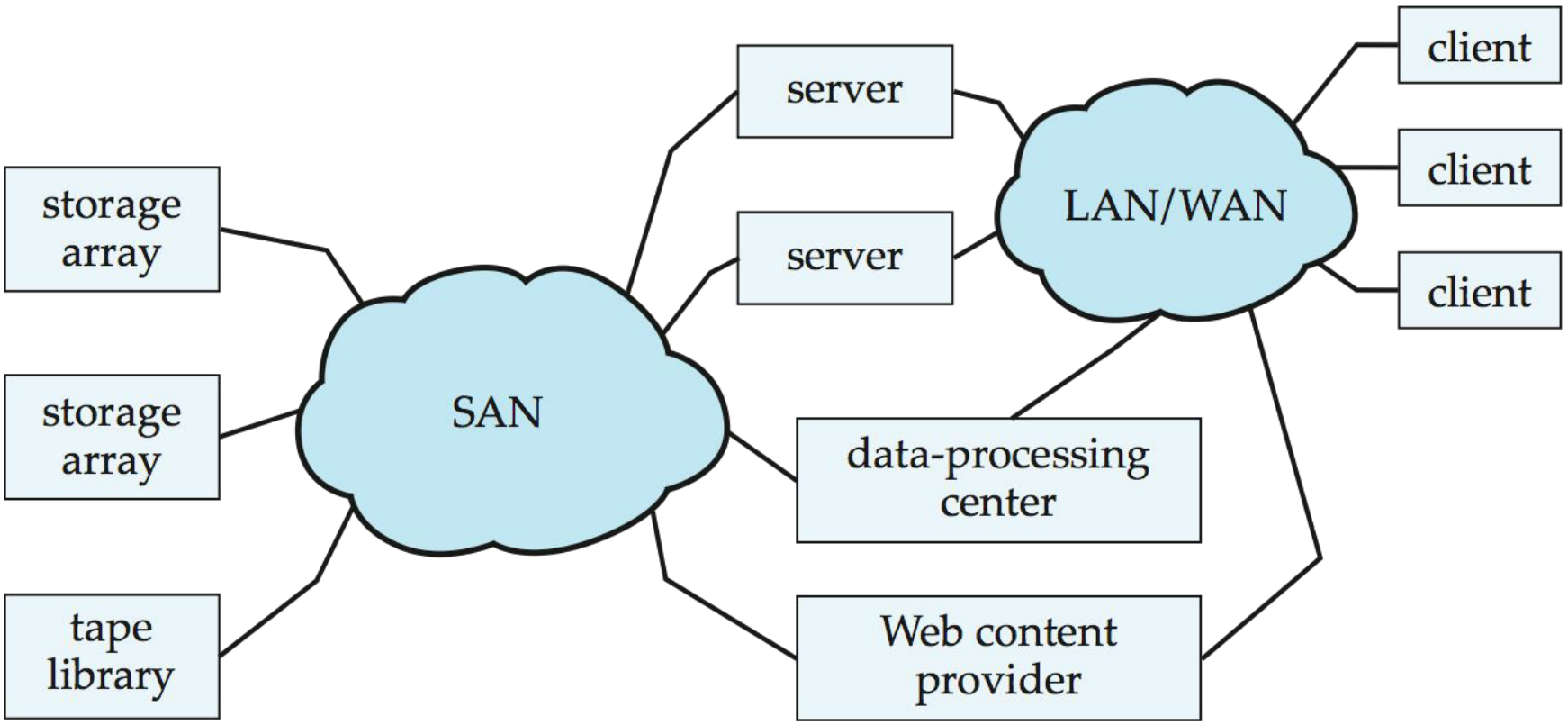

application server
workstation
workstation
workstation
gateway
printer
laptop
file server

Figure 17.11


client
client
client
server
server
LAN/WAN
storage array
storage array
tape library
SAN
data-processing center
Web content provider