Database System Concept (CSE 3103)
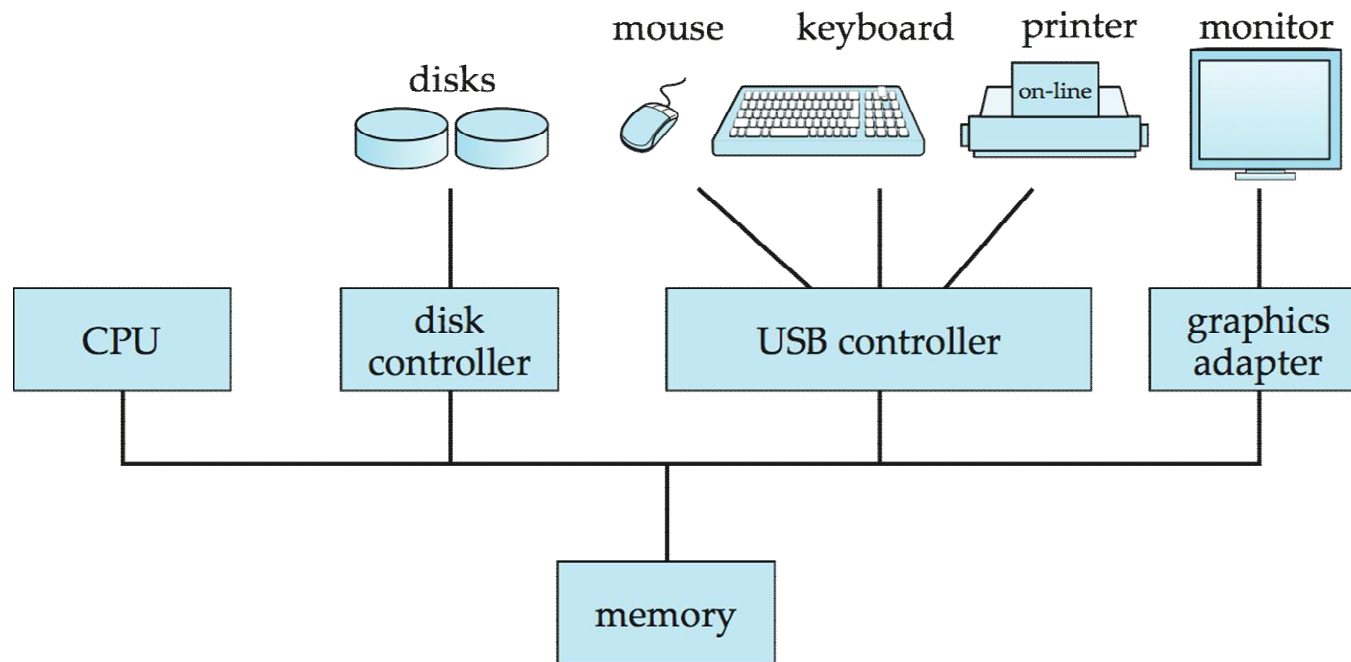
Lecture 09-Day 04

Nazmus Sakib, Assistant Professor, Dept. of CSE, AUST

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.
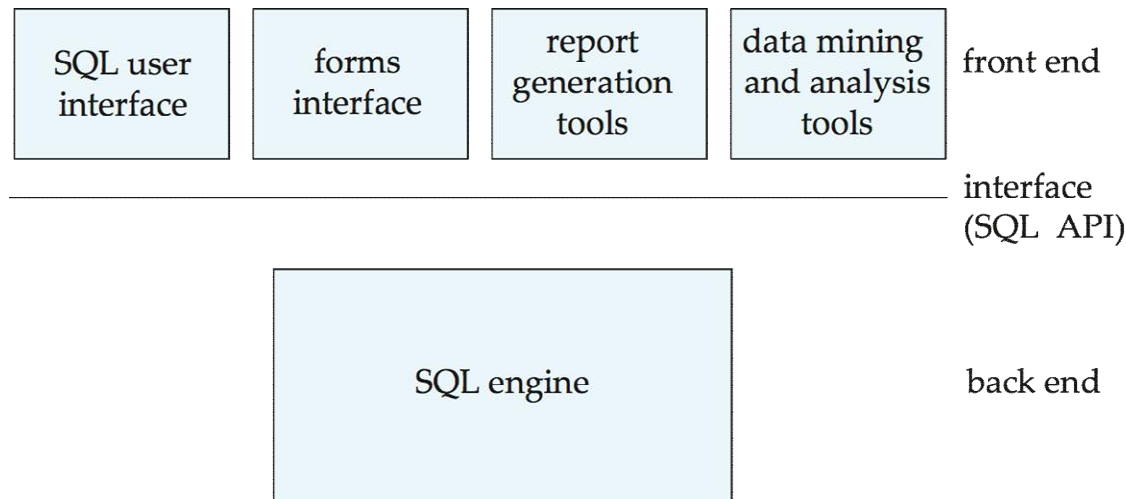
A Centralized Computer System

disks mouse keyboard printer monitor

on-line

CPU	disk controller	USB controller	graphics adapter

memory

Client-Server Systems

- Server systems satisfy requests generated at *m* client systems, whose general structure is shown below:
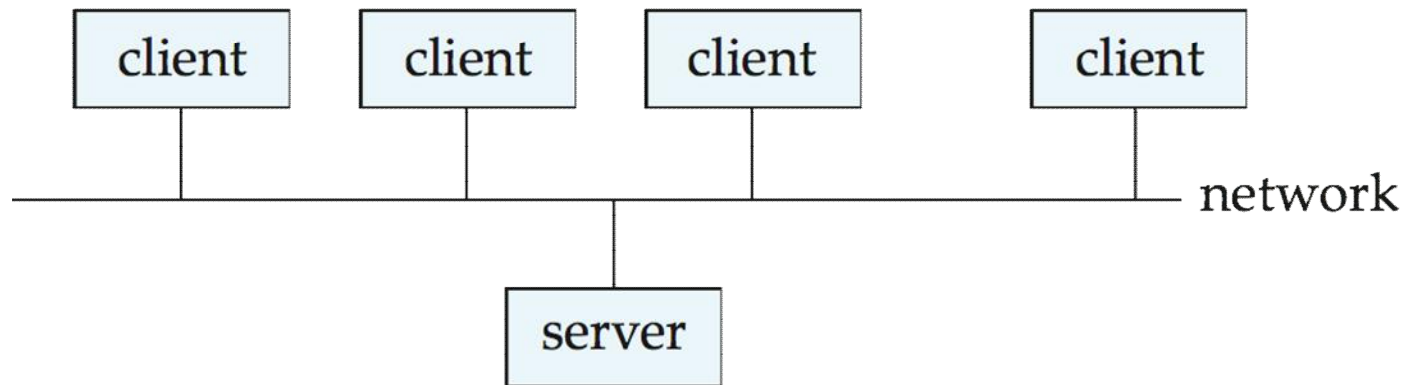
Client-Server Systems (Cont.)

- 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.

SQL user interface	forms interface	report generation tools	data mining and analysis tools	front end

interface (SQL API)

|	
SQL engine	back end

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

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.
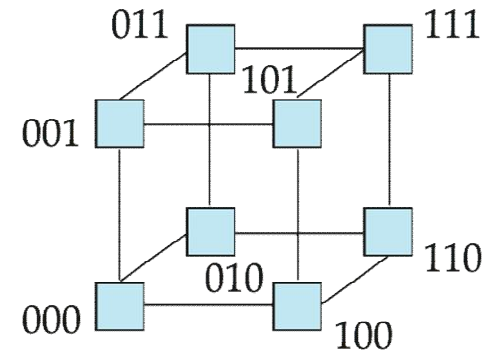
Interconnection Architectures
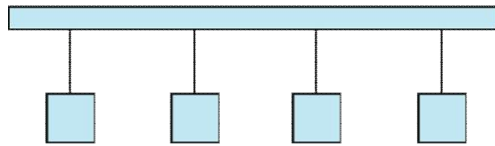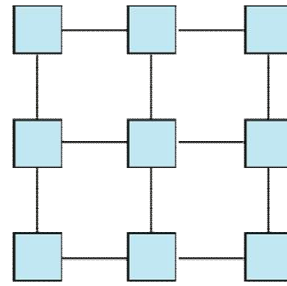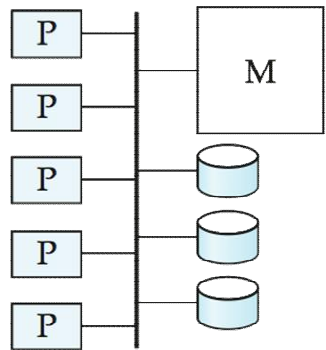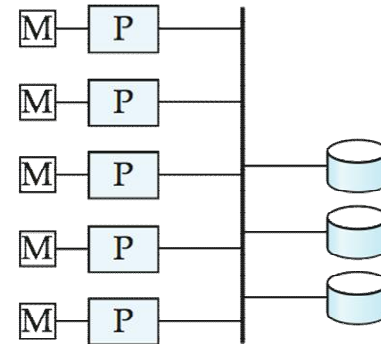

(a) bus
(b) mesh
(c) hypercube

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
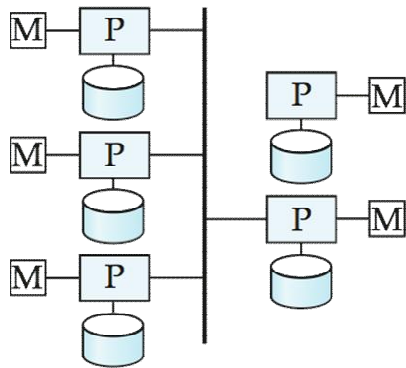- **Hierarchical** -- hybrid of the above architectures

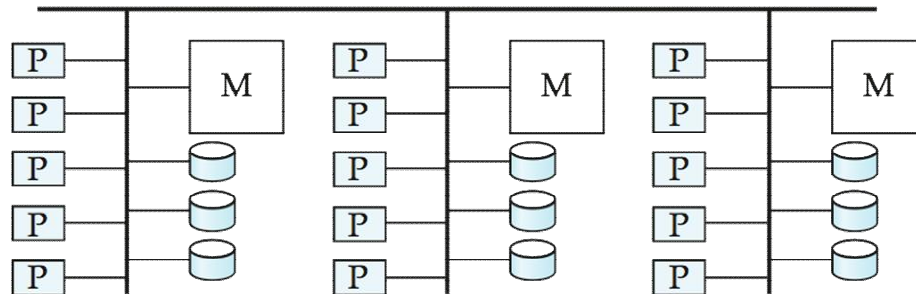# Parallel Database Architectures

(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)**