





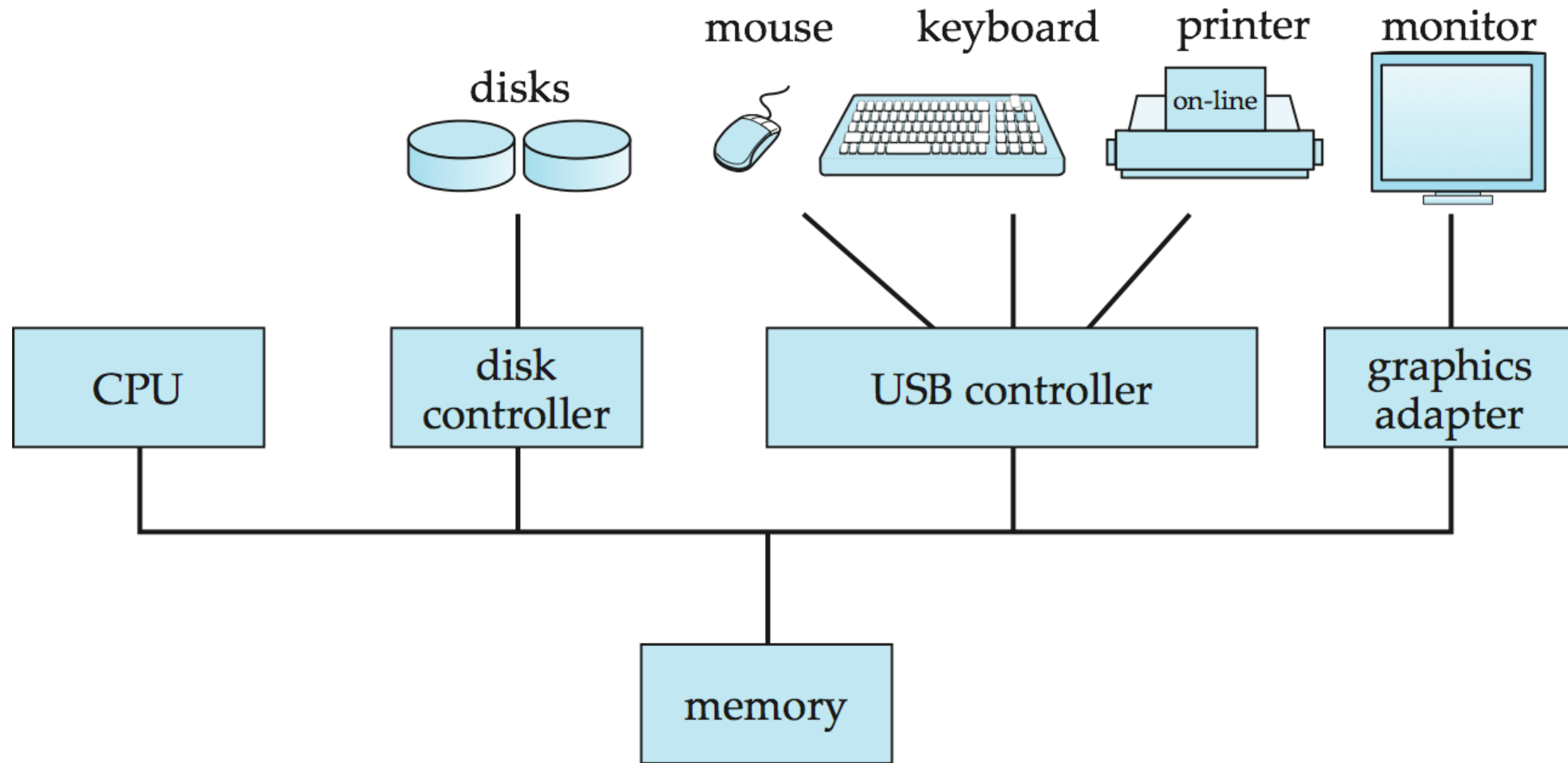
CSE3103 : Database

Nazmus Sakib
Assistant Professor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Centralized Systems

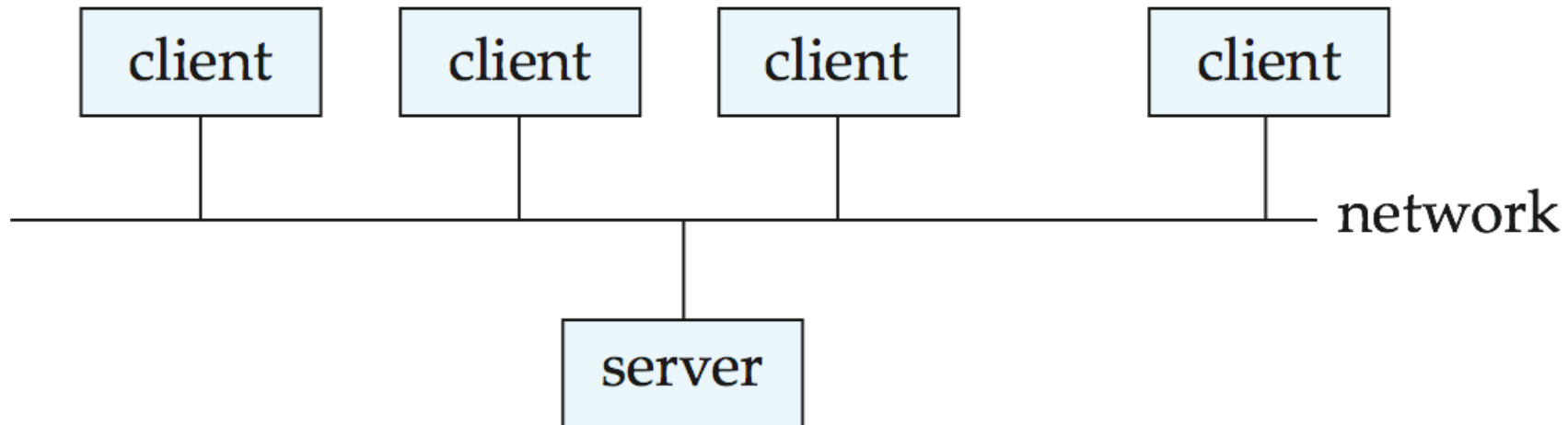
- Run on a single computer system and do not interact with other computer systems.
- General-purpose computer system: one to a few CPUs and a number of device controllers that are connected through a common bus that provides access to shared memory.
- Single-user system (e.g., personal computer or workstation): 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 via terminals. Often called *server* systems.

A Centralized Computer System



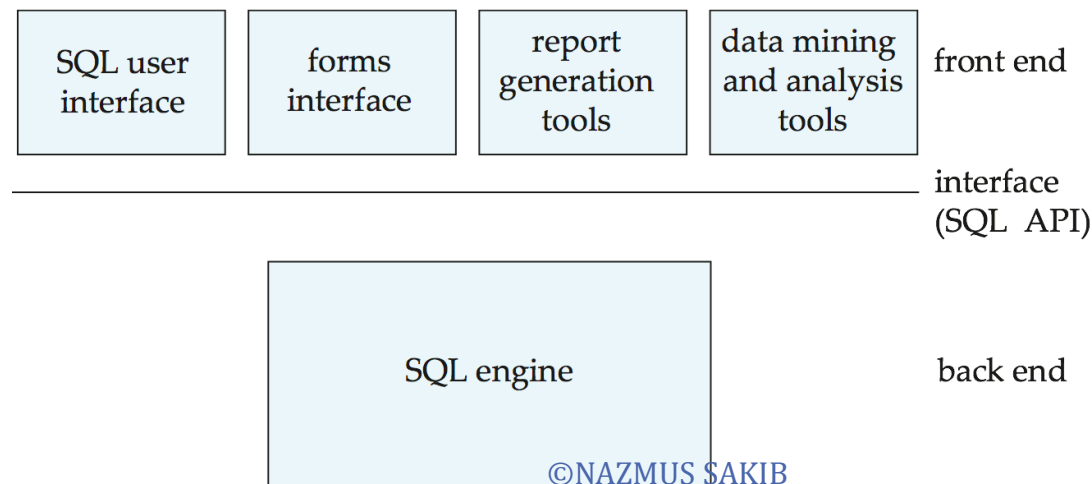
Client-Server Systems

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



Client-Server 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

Server System Architecture

- Server systems can be broadly categorized into two kinds:
 - **transaction servers** which are widely used in relational database systems, and
 - **data servers**, used in object-oriented database systems

Transaction Servers

- Also called **query server** systems or *SQL server* systems
 - Clients send requests to the server
 - Transactions are executed at the server
 - Results are shipped back to the client.
- Requests are specified in SQL, and communicated to the server through a *remote procedure call* (RPC) mechanism.
- Transactional RPC allows many RPC calls to form a transaction.
- *Open Database Connectivity* (ODBC) is a C language application program interface standard from Microsoft for connecting to a server, sending SQL requests, and receiving results.
- JDBC standard is similar to ODBC, for Java

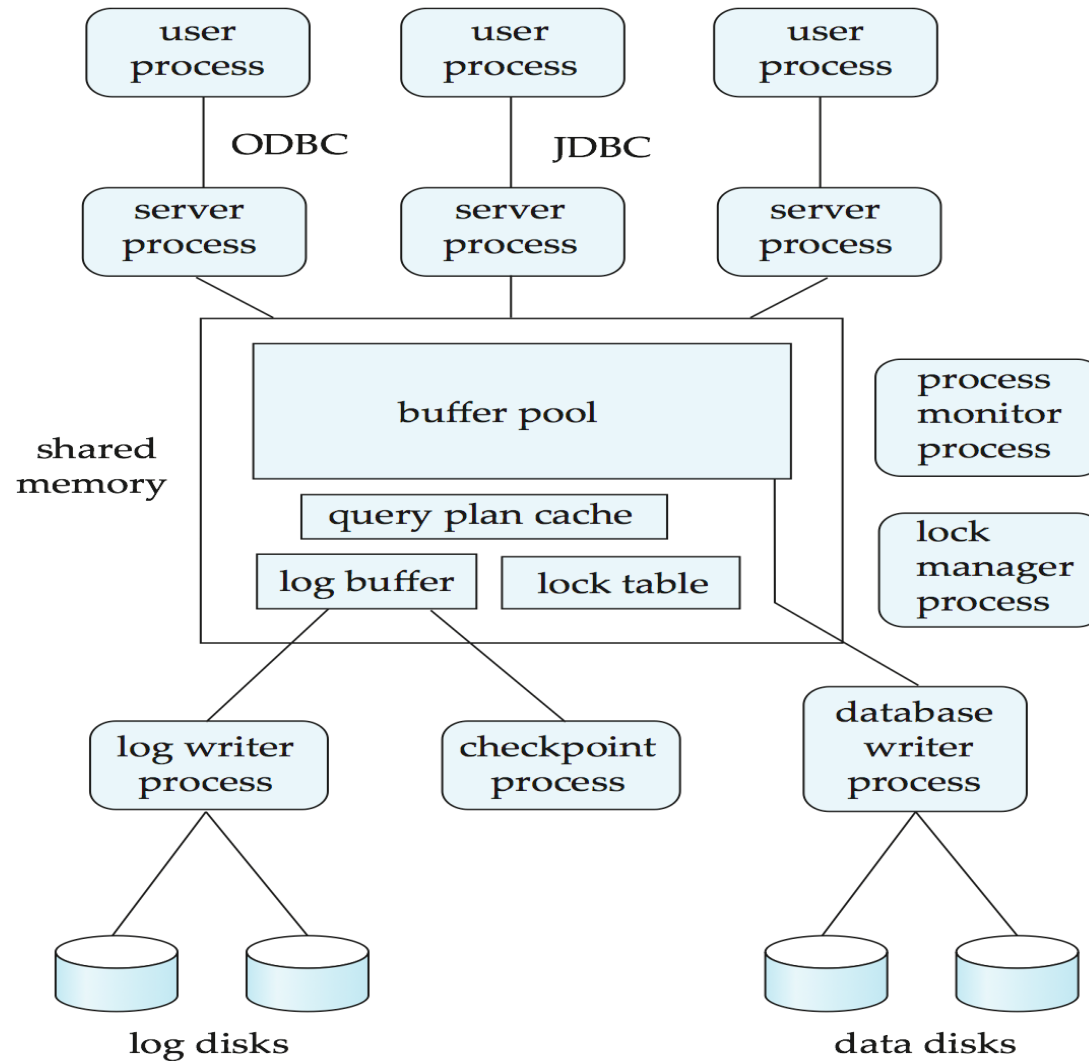
Transaction Server Process Structure

- A typical transaction server consists of multiple processes accessing data in shared memory.
- Server processes
 - These receive user queries (transactions), execute them and send results back
 - Processes may be **multithreaded**, allowing a single process to execute several user queries concurrently
 - Typically multiple multithreaded server processes
- Lock manager process
- Database writer process

Transaction Server Processes (Cont.)

- Log writer process
 - Server processes simply add log records to log record buffer
 - Log writer process outputs log records to stable storage.
- Checkpoint process
 - Performs periodic checkpoints
- Process monitor process
 - Monitors other processes, and takes recovery actions if any of the other processes fail
 - E.g., aborting any transactions being executed by a server process and restarting it

Transaction System Processes (Cont.)



Data Servers

- Used in high-speed LANs, in cases where
 - The clients are comparable in processing power to the server
 - The tasks to be executed are compute intensive.
- Data are shipped to clients where processing is performed, and then shipped results back to the server.
- This architecture requires full back-end functionality at the clients.
- Issues:
 - Page-Shipping versus Item-Shipping
 - Locking
 - Data Caching
 - Lock Caching

Data Servers (Cont.)

- **Page-shipping** versus **item-shipping**
 - Smaller unit of shipping \Rightarrow more messages
 - Worth **prefetching** related items along with requested item
 - Page shipping can be thought of as a form of prefetching
- Locking
 - Overhead of requesting and getting locks from server is high due to message delays
 - Can grant locks on requested and prefetched items; with page shipping, transaction is granted lock on whole page.
 - Locks on a prefetched item can be P{called back} by the server, and returned by client transaction if the prefetched item has not been used.

Data Servers (Cont.)

- **Data Caching**

- Data can be cached at client even in between transactions
- But check that data is up-to-date before it is used (**cache coherency**)
- Check can be done when requesting lock on data item

- **Lock Caching**

- Locks can be retained by client system even in between transactions
- Transactions can acquire cached locks locally, without contacting server
- Server **calls back** locks from clients when it receives conflicting lock request. Client returns lock once no local transaction is using it.
- Similar to deescalation, but across transactions.

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:

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

- Speedup is **linear** if equation equals N .

- **Scaleup**: increase the size of both the problem and the system

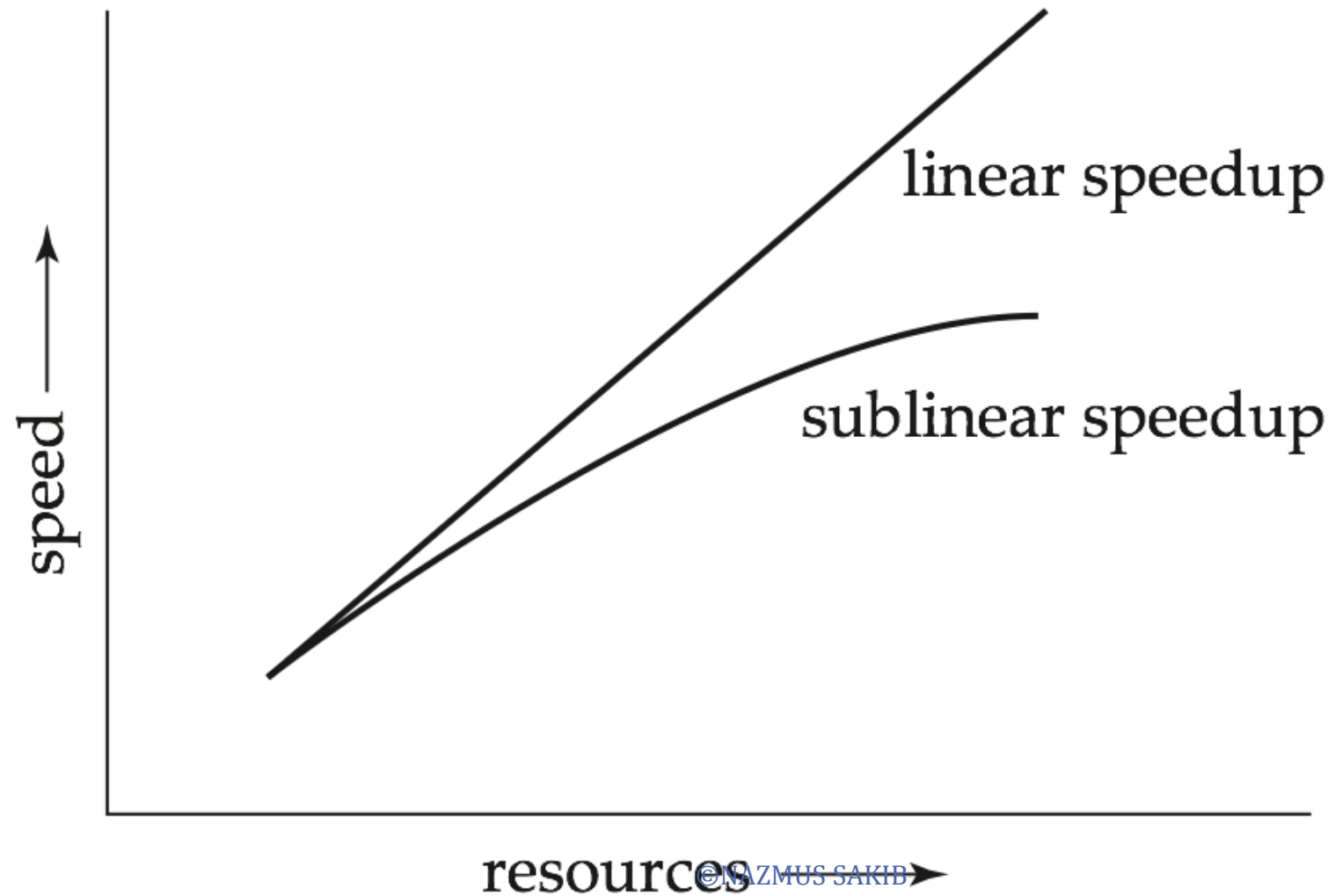
- N -times larger system used to perform N -times larger job

- Measured by:

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

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

Speedup



Scaleup

