CSE 453 High Performance Database System

Dr. Abu Sayed Md. Latiful Hoque
Professor, CSE, BUET

Contact: mobile 01556346357
email: asmlatifulhoque@cse.buet.ac.bd

About Course

- **Course Summary:**
  1. Parallel Database Architecture, partitioning, replication, indexing and Query processing
  2. Distributed Database Architecture, Storage and Query processing, transaction management, concurrency control and Design
  3. Query optimization in centralized and distributed database
  4. High performance data models: NoSQL, semi-structured and column-oriented etc.
  5. Big data and Data Analytics: overview, Data warehousing: Storage structure and star schema, design and OLAP

Books

- Text books:

Database System Concepts 7th Edition

By

Abraham Silberschatz, Henry F. Korth, S. Sudarshan,

- Reference books:

Fundamentals of Database Systems 7th Edition

By

Ramez Elmasri and Shamkant B. Navathe

Weightage Distribution among Assessment Tools

- As per academic council guidelines

- Class performance
- Assignment on Data Analytics
- Assignment on High Performance Data Models
- Class Tests
- Final Examination

Tentative Mode of Online Class

Lecture will be interactive using ppt slides.

1. After a few slides, there shall be some analytical questions to answer or some problems to solve.
2. Each student must write the answer in A4 size white paper with black ball point pen clearly and chronologically.
3. The answer of the questions or the solution of the problems will be discussed and guided so that the students can write answer/solve the question/problem.
4. At the end of the class, students will be given some time to upload the scanned/image of the sheets as per moodle requirements.
5. These uploads will be considered as class performance.

Basic towards High Performance DBMS

- What you have leant in Basic DBMS?
  1. Data model - Relational model
  2. Query Languages
     - Relational Algebra
     - Relational Calculus
     - SQL
  3. Database Design (ERD)
  4. Refinement of Database Design (Normalization)

Basic towards High Performance DBMS

- What you have learnt in Basic DBMS?

5. Database Storage Management -> High Performance
6. Indexing -> High Performance
7. Query Processing and Optimization -> High Performance
8. Transaction management -> High Performance
9. Concurrency Control of Transactions -> High Performance
10. Distributed Database
11. -> High Performance Data Models

Classification of Physical Storage Media

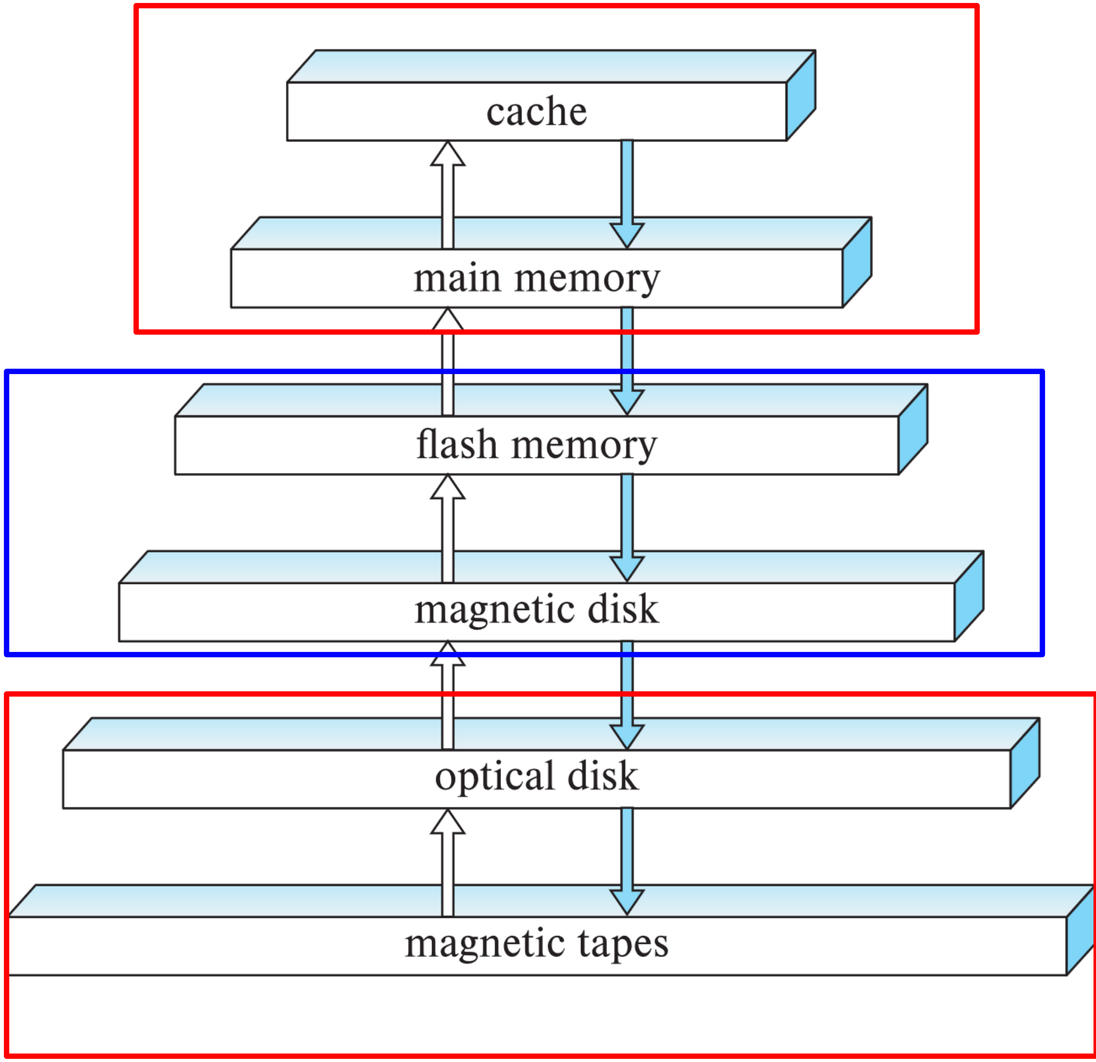
- Can differentiate storage into:
  - **volatile storage:** loses contents when power is switched off
  - **non-volatile storage**:
    - Contents persist even when power is switched off.
    - Includes secondary and tertiary storage, as well as batter- backed up main-memory.
- Factors affecting choice of storage media include
  - Speed with which data can be accessed
  - Cost per unit of data
  - Reliability

Storage Hierarchy

primary storage: Fastest media but volatile (cache, main memory).

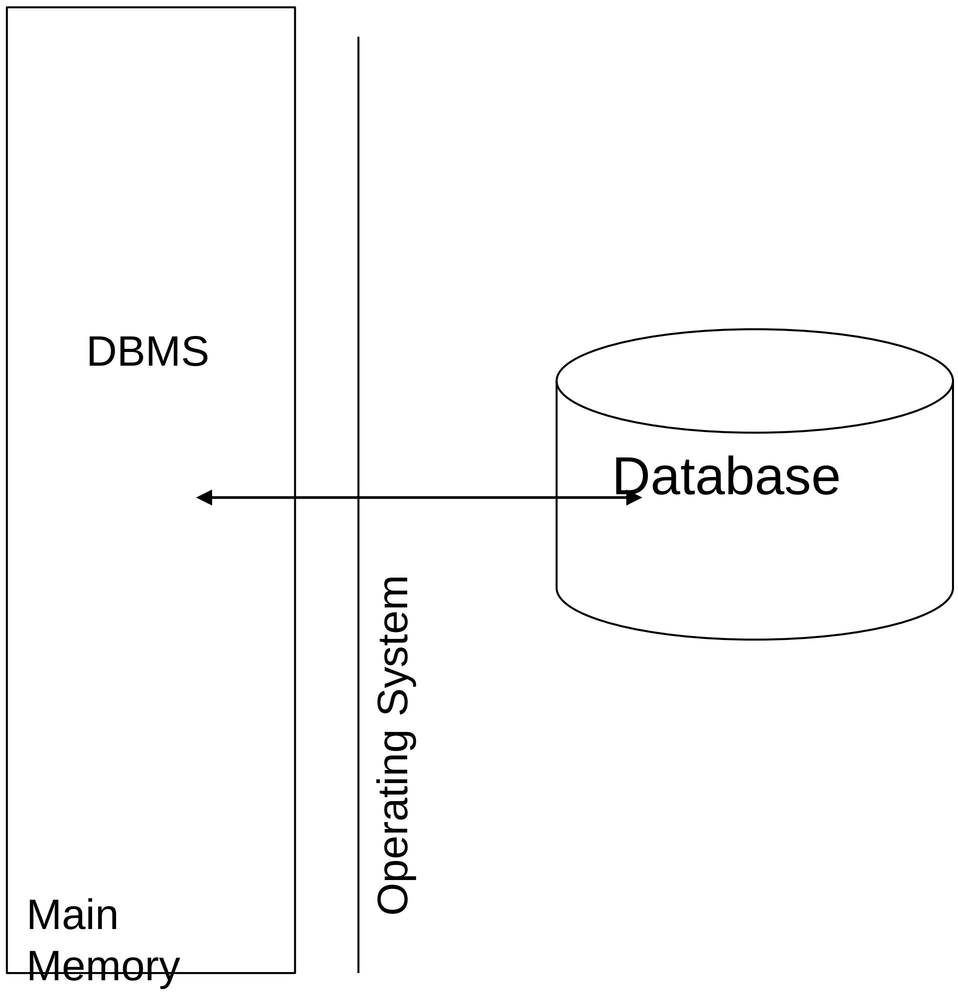
secondary storage: next level in hierarchy, non-volatile, moderately fast access time
Also called **on-line storage**
E.g., flash memory, magnetic disks

tertiary storage: lowest level in hierarchy, non-volatile, slow access time
also called **off-line storage** and used for **archival storage**
e.g., magnetic tape, optical storage



DBMS System Architecture

- Main Memory DBMS
- Disk-Based DBMS

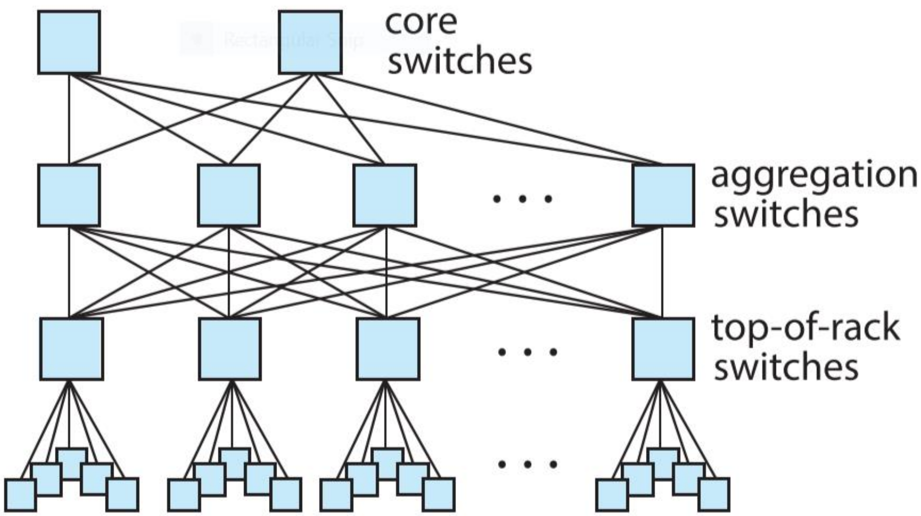


Question 1-1: Explain the implications of storage to MMDBMS and Disk-based DBMS

Parallel Database System Architecture (Data Center Server System)

Tree-like or Fat-Tree Topology:

- widely used in data centers today
- DC are typically mounted in racks
- Each rack has approx. 40 nodes
- Top of rack switch for approx 40 machines in rack
- Each top of rack switch connected to multiple aggregation switches.
- Aggregation switches connect to multiple core switches.



(e) tree-like topology

Parallel Systems

- Parallel database systems consist of multiple processors and multiple disks connected by a fast interconnection network.
- Motivation: handle workloads beyond what a single computer system can handle
- High performance **transaction processing**
  E.g. handling user requests at web-scale
- **Decision support** on very large amounts of data
  E.g. data gathered by large web sites/apps

Parallel Systems (Cont.)

- 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.
  Typically hosted in a **data center**
- 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

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

Measured by

$$\text{Speed up} = \frac{\textit{small system elapsed time}}{\textit{large system elapsed time}}$$

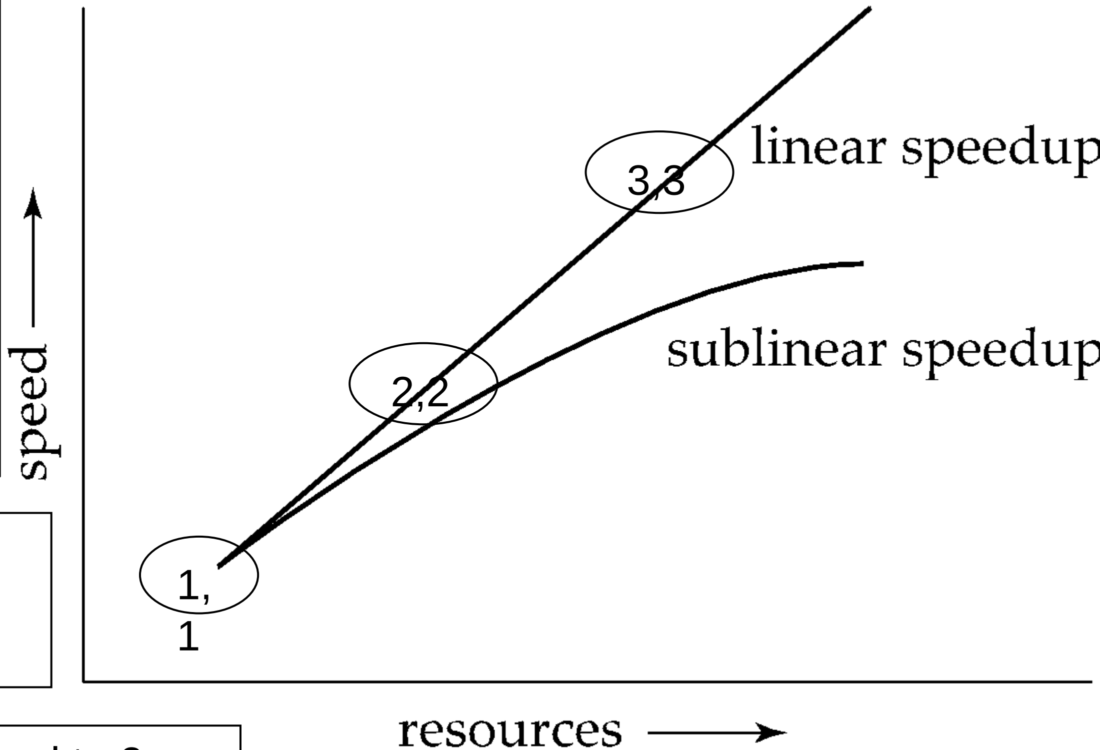
Speedup is **linear** if equation equals N.

Example: A server system has 1 node (small system) and elapsed to solve problem P is 10ms.

1. The number of node has been increased to 2. The time is 5ms. Speed up = 10 /5 = 2

2. The number of node has been increased to 3. The time is 3.333ms. Speed up = 10/3.33 = 3

Linear Speed up 1, 2, 3 …



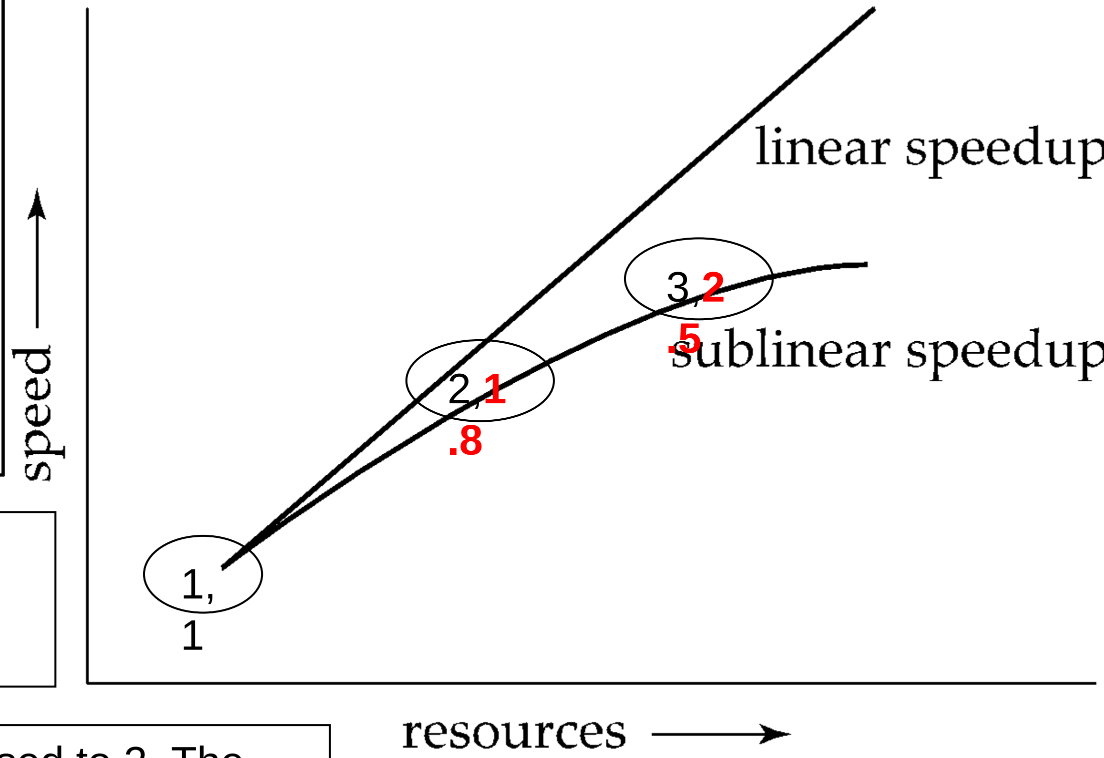
Speed-Up

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

Measured by

$$\text{Speed up} = \frac{\textit{small system elapsed time}}{\textit{large system elapsed time}}$$

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



Example: A server system has 1 node (small system) and elapsed to solve problem P is 10ms.

1. The number of node has been increased to 2. The time is 5.55ms. Speed up = 10 / 5.55 = 1.8

2. The number of node has been increased to 3. The time is 4ms. Speed up = 10/4 = 2.5

SubLinear Speed up 1, 2, 3 …

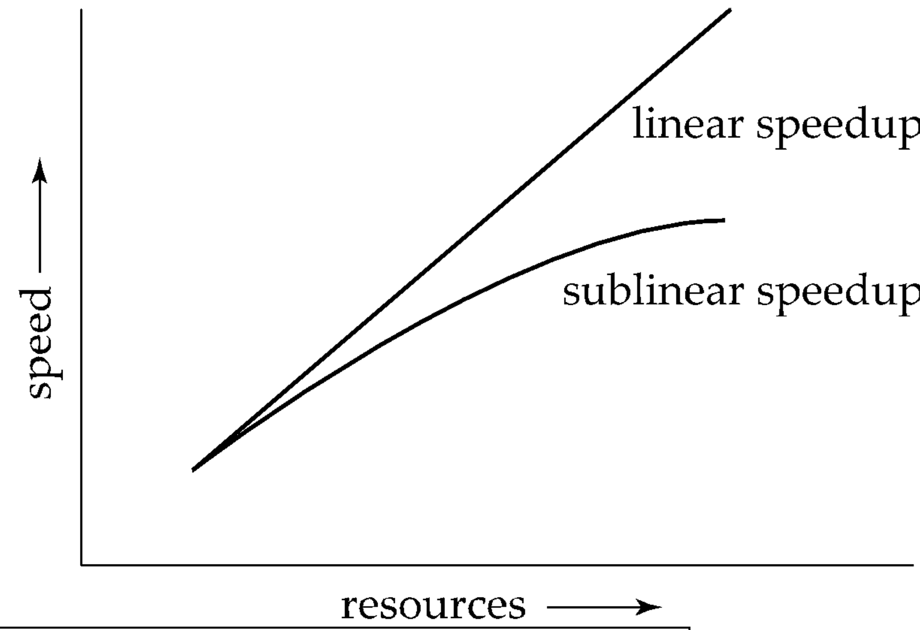
Speed-Up

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

Measured by

$$\text{Speed up} = \frac{\textit{small system elapsed time}}{\textit{large system elapsed time}}$$

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



Question 2-1: A server system has 1 node (small system) and elapsed to solve problem P is 10ms.

1. The number of node has been increased to 2. The time to solve P is 5ms
2. The number of node has been increased to 3. The time to solve P is 4ms
3. The number of node has been increased to 4. The time to solve P is 3ms

Find the type of speedup graph for the above system and explain.

Scale-Up

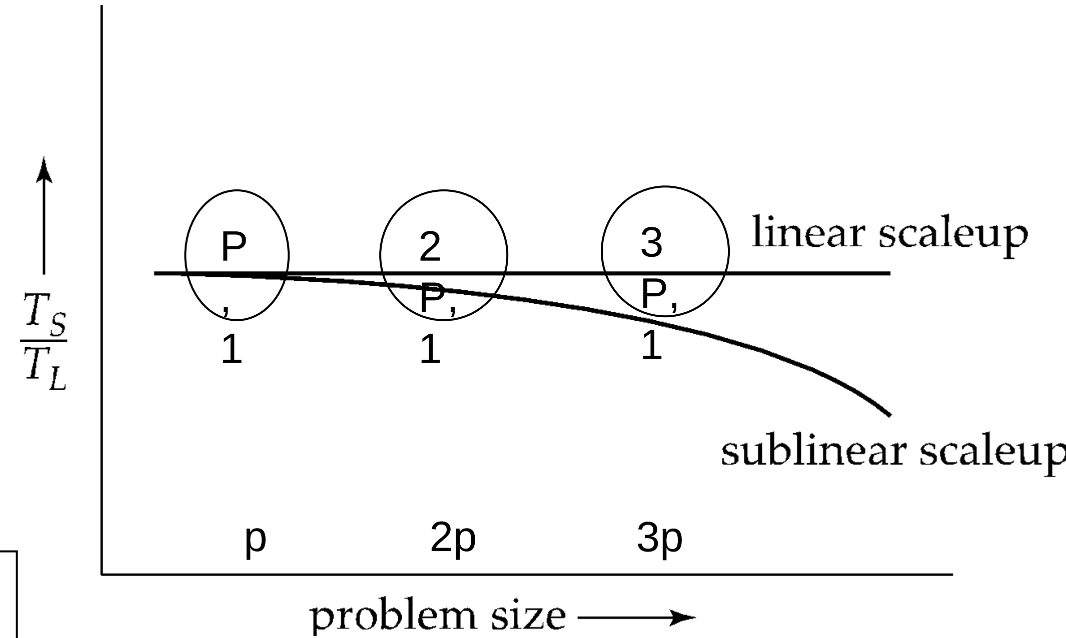
Scaleup: increase the size of both the problem and the system *N*-times larger system used to perform *N*-times larger job

Measured by:

Scale up =

$$\frac{\textit{small system small problem elapsed time}}{\textit{big system big problem elapsed time}}$$

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



Example: A server system has 1 node (small system) and elapsed to solve problem P is 10ms.

1. The number of node has been increased to 2. The size of the problem = 2p, elapsed time = 10ms, Scale up = 10/10 = 1

1. The number of node has been increased to 3. The size of the problem = 3p, elapsed time = 10ms, Scale up = 10/10 = 1

Linear Scale up

Scale-Up

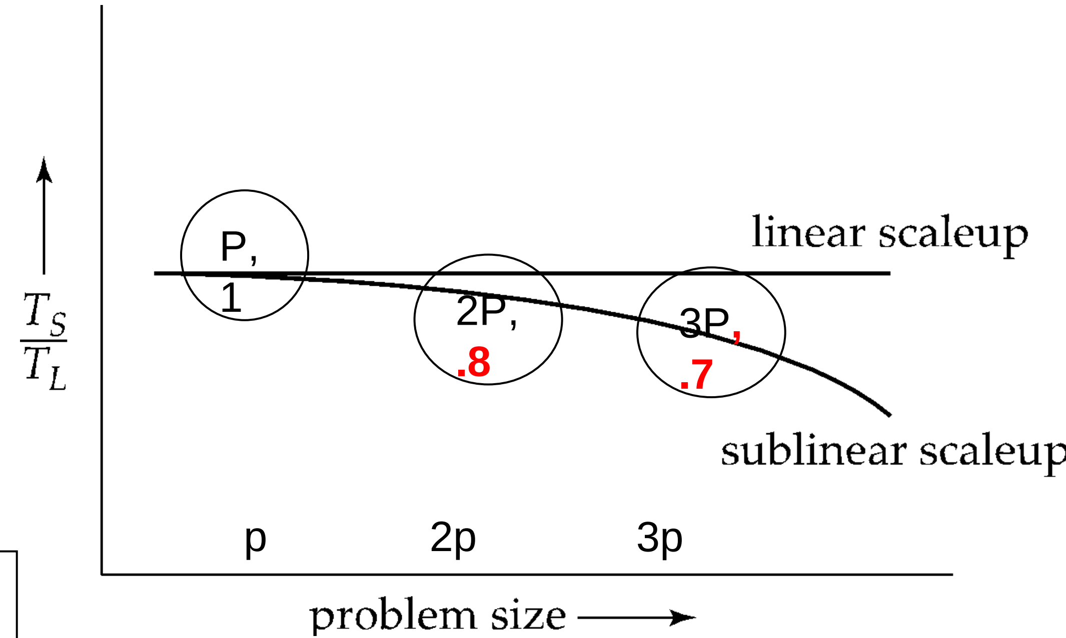
Scaleup: increase the size of both the problem and the system *N*-times larger system used to perform *N*-times larger job

Measured by:

Scale up =

$$\frac{\textit{small system small problem elapsed time}}{\textit{big system big problem elapsed time}}$$

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



Example: A server system has 1 node (small system) and elapsed to solve problem P is 10ms.

1. The number of node has been increased to 2.
   The size of the problem = 2p, elapsed time = 12ms, Scale up = 10/12 = 0.8

2. The number of node has been increased to 3.
   The size of the problem = 3p, elapsed time = 14 ms, Scale up = 10/14 = 0.7

SubLinear Scale up

Scale-Up

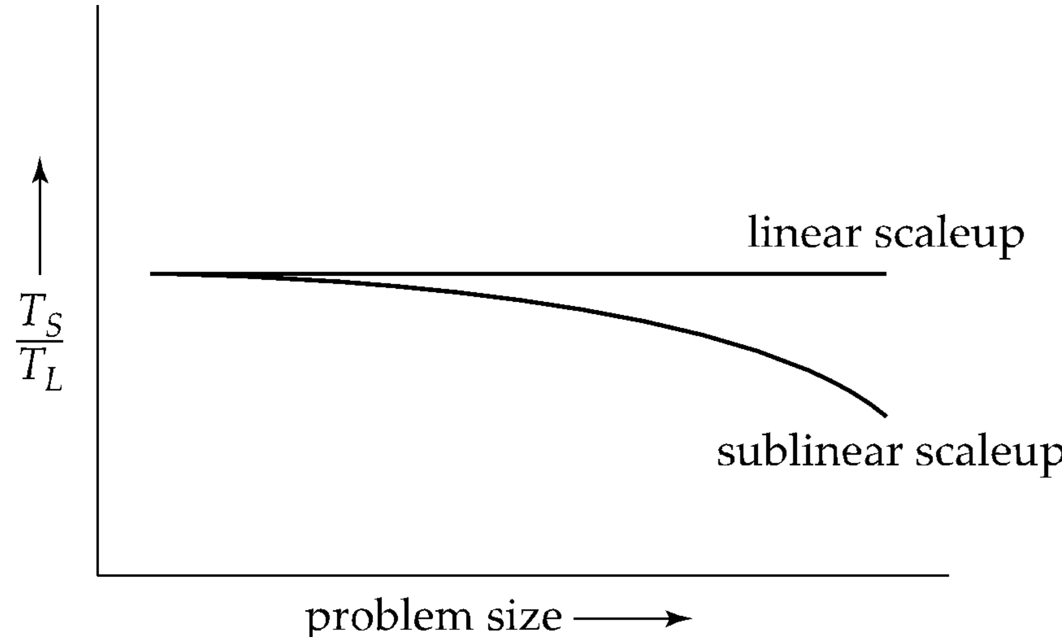
Scaleup: increase the size of both the problem and the system *N*-times larger system used to perform *N*-times larger job

Measured by:

Scale up =

$$\frac{\textit{small system small problem elapsed time}}{\textit{big system big problem elapsed time}}$$

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



Question 2-2: A server system has 1 node (small system) and elapsed to solve problem P is 10ms.

1. The number of node has been increased to 4.The size of the problem = 4p, elapsed time is 40ms
2. The number of node has been increased to 8.The size of the problem = 8p, elapsed time is 80ms

Find type of scale up graph for the above system and explain

Factors Limiting Speedup and Scaleup

Speedup and scaleup are often sublinear due to:

Startup/sequential costs: Cost of starting up multiple processes, and sequential computation before/after parallel computation

May dominate computation time, if the degree of parallelism is high

Suppose sequentially, a task completion time = T

Fraction of T that can be executed parallel by *n* nodes = *p*

Sequential time = (1-p)T ***Parallel time = (pT)/n***

Elapsed time = (1-p)T +pT/n

Amdahl's law: speedup = $\dfrac{T}{(1-p)T + pT/n}$

$$= \frac{1}{(1-p) + p/n}$$

Factors Limiting Speedup and Scaleup

Suppose sequentially, a task completion time = T

Fraction of T that can be executed parallel by *n* nodes = *p*

Sequential time = (1-p)T ***Parallel time = (pT)/n***

Elapsed time = (1-p)T +pT/n

Amdahl's law: speedup = $$\frac{T}{(1-p)T + pT/n}$$

$$= \frac{1}{(1-p) + p/n}$$

Question 2-3: Explain speedup as per Amdahl's law for the following cases:

Case 1: Full fraction of T can be executed in parallel (p =1)
Case 2: No fraction of T can be executed in parallel (p =0)
Case 3: A fraction of T can be executed in parallel (0 < p <1)

Factors Limiting Speedup and Scaleup

Speedup and scaleup are often sublinear due to:

Startup/sequential costs: Cost of starting up multiple processes, and sequential computation before/after parallel computation

May dominate computation time, if the degree of parallelism is high

Suppose sequentially, a task completion time = T

Fraction of T that can be executed parallel by n nodes = p

Problem size = nT

Sequential time = (1-p)nT ***Parallel time = (pnT)/n = pT***

Elapsed time = (1-p)nT +pT

Gustafson's law: scaleup = $\frac{T}{(1-p)nT + pT}$

$$= \frac{1}{(1-p)n + p}$$

Factors Limiting Speedup and Scaleup

Suppose sequentially, a task completion time = T

Fraction of T that can be executed parallel by *n* nodes = *p*

Problem size = nT

Sequential time = (1-p)nT ***Parallel time = (pnT)/n = pT***

Elapsed time = (1-p)nT +pT

Gustafson's law: scaleup = $\frac{T}{(1-p)nT + pT}$

$$= \frac{1}{(1-p)n + p}$$

Question 2-2: Explain scaleup as per Gustafson's law for the following cases:

Case 1: Full fraction of T can be executed in parallel (p =1)
Case 2: No fraction of T can be executed in parallel (p =0)

Factors Limiting Speedup and Scaleup

Speedup and scaleup are often sublinear due to:

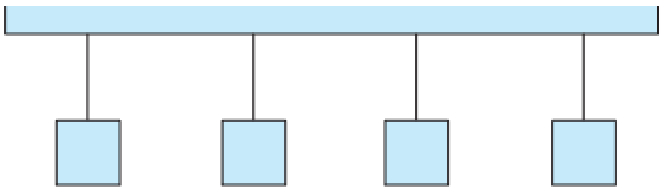
Interference: Processes accessing shared resources (e.g.,system bus, disks, or locks) compete with each other, thus spending time waiting on other processes, rather than performing useful work.

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.



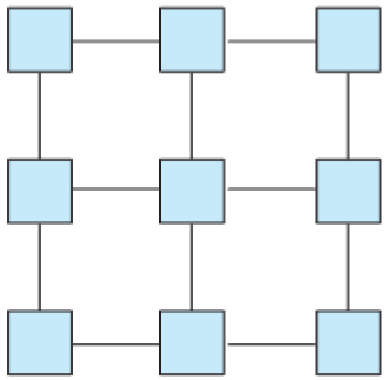
(a) bus

Interconnection Network Architectures

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 $2(\sqrt{n}\text{-}1)$ hops to send message to a node

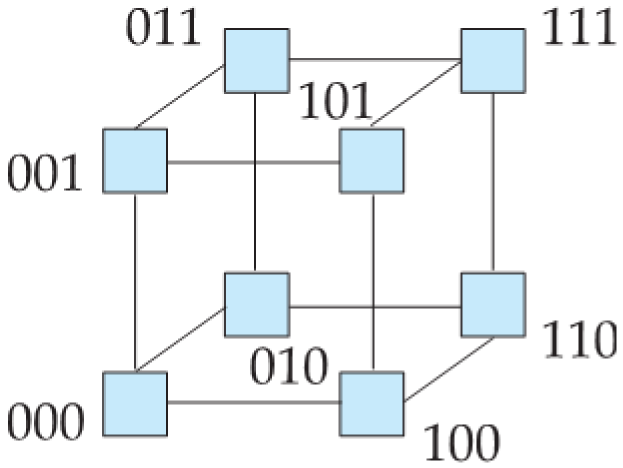


(c) mesh

Interconnection Network Architectures

Hypercube. Components are numbered in binary; components are connected to one another if their binary representations differ in exactly one bit.

Each of n components are connected to *log(n)* other components and can reach each other via at most *log(n)* links; reduces communication delays.

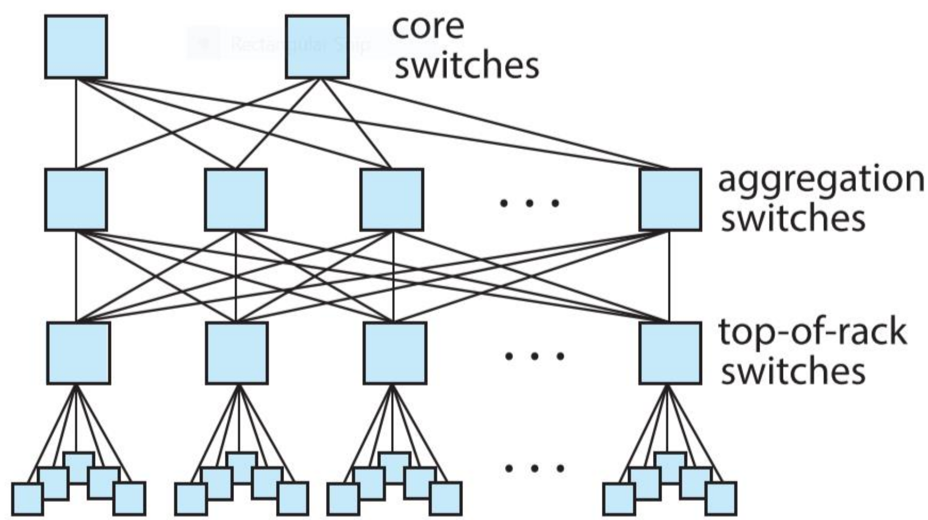


(d) hypercube

Interconnection Network Architectures (Data Center Server System)

Tree-like or Fat-Tree Topology:

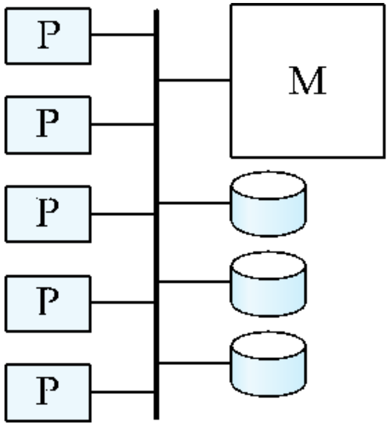
- widely used in data centers today
- DC are typically mounted in racks
- Each rack has approx. 40 nodes
- Top of rack switch for approx 40 machines in rack
- Each top of rack switch connected to multiple aggregation switches.
- Aggregation switches connect to multiple core switches.



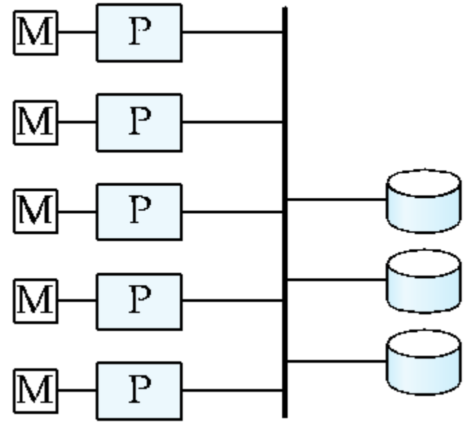
(e) tree-like topology

Question 3-1: Discuss comparative advantages and Disadvantages of BUS, Mesh, Hypercube and Tree topology.

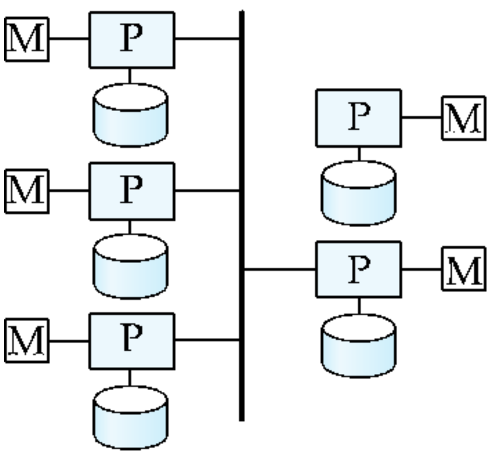
Parallel Database Architectures



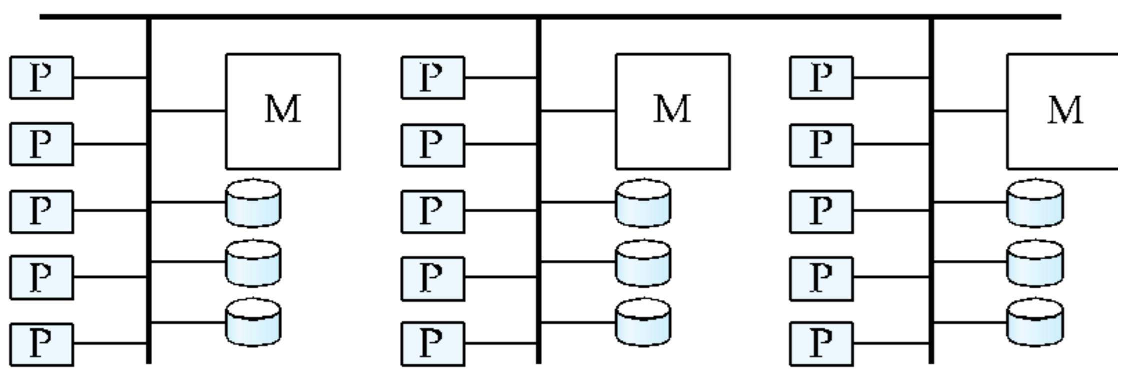
(a) shared memory



(b) shared disk



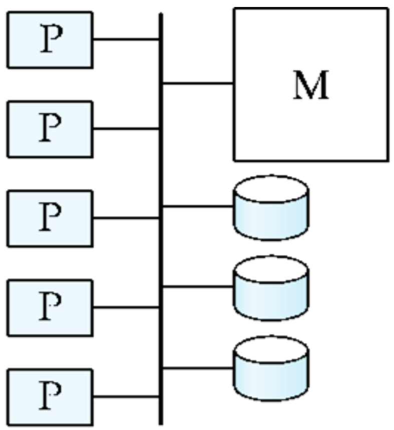
(c) shared nothing



(d) hierarchical

Shared Memory

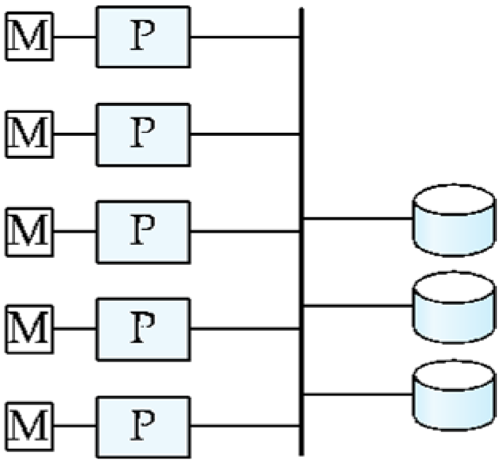
- Processors (or processor cores) and disks have access to a common memory
  - Via a bus in earlier days, through an interconnection network today
- Extremely efficient communication between processors
- Downside: shared-memory architecture is not scalable beyond 64 to 128 processor cores
  - Memory interconnection network becomes a bottleneck



(a) shared memory

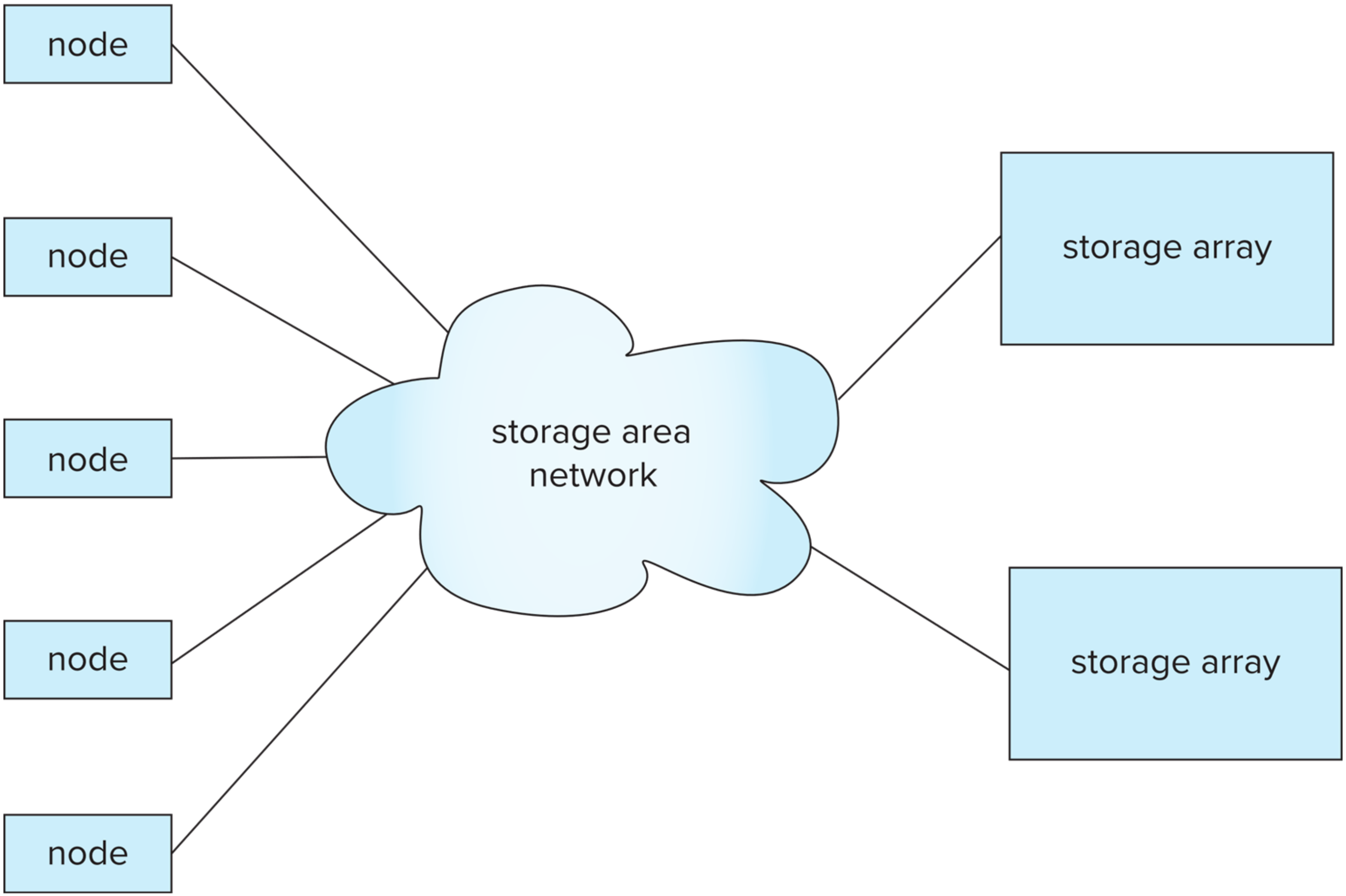
Shared Disk

- All processors can directly access all disks via an interconnection network, but the processors have private memories.
  - Architecture provides a degree of **fault-tolerance** — if a processor fails, the other processors can take over its tasks
    - the data of the failed processor is resident on disks that are accessible from all processors.
- Downside: bottleneck now occurs at interconnection to the disk subsystem.



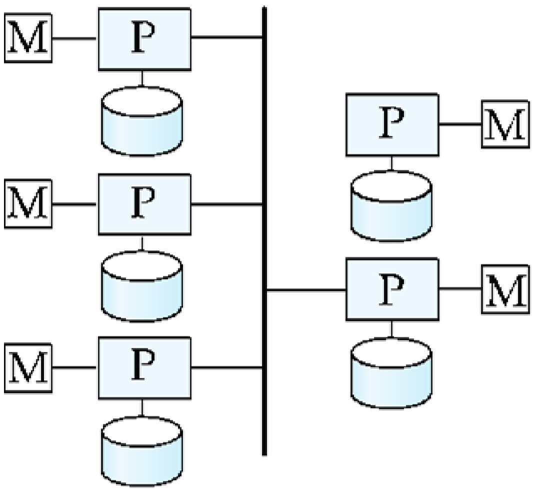
(b) shared disk

Modern Shared Disk Architectures: via Storage Area Network (SAN)



Shared Nothing

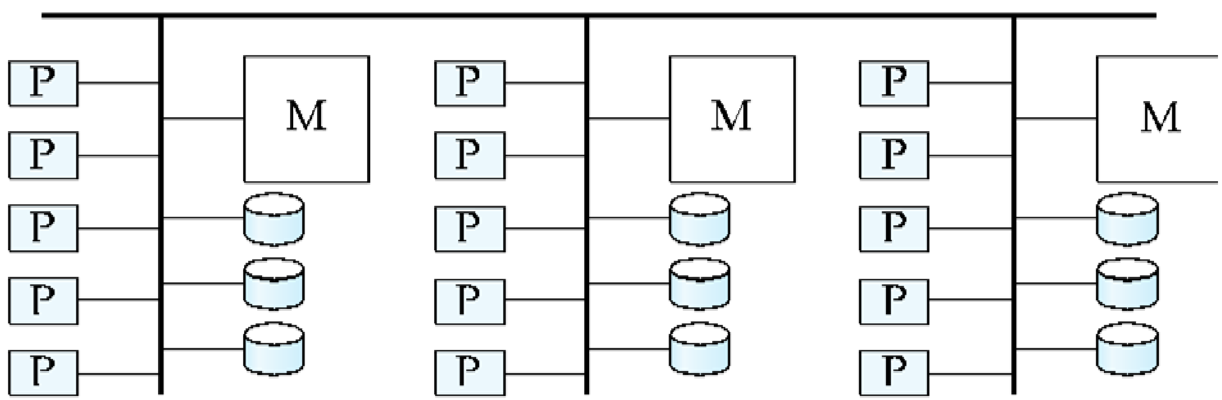
- Node consists of a processor, memory, and one or more disks
- All communication via interconnection network
- Can be scaled up to thousands of processors without interference.
- Main drawback: cost of communication and non-local disk access; sending data involves software interaction at both ends.



(c) shared nothing

Hierarchical

- Combines characteristics of shared-memory, shared-disk, and shared-nothing architectures.
  - Top level is a shared-nothing architecture
    - With each node of the system being a shared-memory system
  - Alternatively, top level could be a shared-disk system
    - With each node of the system being a shared-memory system



(d) hierarchical

Question 3-2: Show the hierarchical architecture with top level be a shared-disk system and each node of the system being a shared-memory system.

Shared-Memory Vs Shared-Nothing

- Shared-memory internally looks like shared-nothing!
  - Each processor has direct access to its own memory, and indirect (hardware level) access to rest of memory
  - Also called **non-uniform memory architecture (NUMA)**
- Shared-nothing can be made to look like shared memory
  - Reduce the complexity of programming such systems by **distributed virtual-memory** abstraction
  - **Remote Direct Memory Access (RDMA)** provides very low-latency shared memory abstraction on shared-nothing systems
    - Often implemented on top of i**nfiniband** due it its very-low-latency
  - But careless programming can lead to performance issues