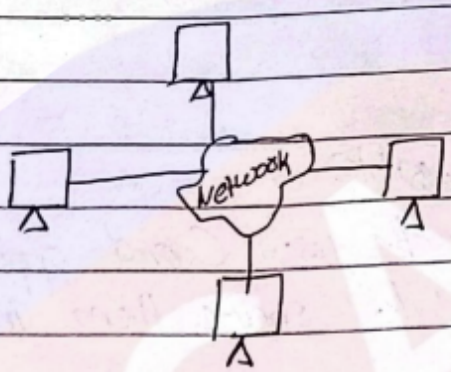
Distributed System

→ It is a collection of autonomous computing elements that appears to its users as a single coherent system.



- a collection of computing element (nodes : HW device or s/w process) each being able to behave independently of each other.
- users (people or applications) believe they are dealing with a single system.

Characteristics

a) Collection of autonomous computing elements

→ In a DS there are multiple components that may be decomposed further.

→ These components are autonomous, i.e they possess full control over their parts at all times.

b) Single Coherent System

→ In a single coherent system the collection of nodes as a whole operates the same, no matter where, when and how interaction bet^n a user and the system takes place.

Design Goals

a) Making resources Accessible
b) Distribution Transparency
c) Openness
d) Scability
e) Pitfalls

Making Resources Accessible

→ to make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way
→ Resources :- printers, computers, storage facilities, data, files, web pages, and networks, etc.

Distribution Transparency

→ to hide the fact that its processes & resources are physically distributed across multiple computers
→ In other words, it takes to make the distribution of processes and resources transparent, that is, invisible, to end users and applications.

Types

→ Access
→ location
→ Relocation
→ migration
→ Replication
→ Concurrency
→ Failure

Openness

- Open distributed system → system that offers services according to standard rules that describe the syntax and semantics of those services.
- For example, in computer networks, standard rules govern the format, contents, and meaning of messages sent and received. Such rules are formalized in protocols. In distributed systems, services are generally specified through interfaces, which are often described in an Interface Definition Language (IDL).

Scalability

→ " of a system can be measured along at least three different dimensions:

- A system can be ~~scal~~ scalable with respect to its size
→ add more users and resources to the system
- A system can be administratively scalable
→ it can still be easy to manage even if it spans many independent administrative organizations
- A geographically scalable system
→ add more users and resources geographically different locations to the system

Pitfalls when developing DS

→ False assumptions that everyone makes when developing a distributed application for the first time:

a) The network is reliable
b) " " " secure
c) " " " homogeneous
d) The topology does not change
e) Latency is zero
f) Bandwidth is infinite
g) Transport cost is zero
h) There is one administrator

Types of DS

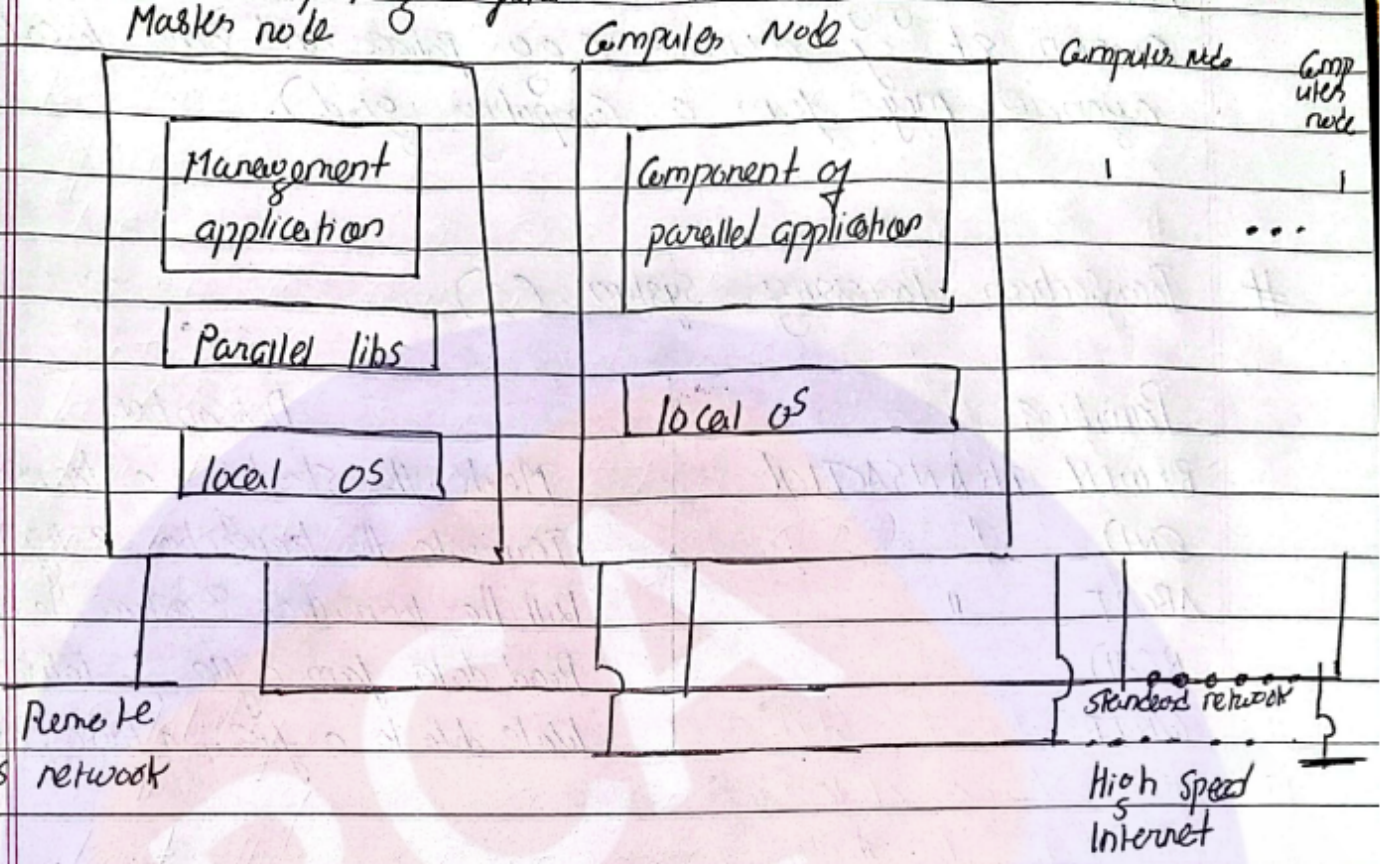
1. Distributed computing system: Focus on computation
   - Cluster computing system
   - Grid computing system

2. Distributed Information System: Focus on interoperability
   - Transaction Processing system
   - Enterprise Application Integration (Exchange info via RPC or RMI)

3. Distributed Pervasive system (usually small, battery-powered systems, Mobile & Wireless): Focus on mobile, embedded, communicating
   - Home system (e.g. Smart phones, PDAs)
   - Electronic Health Care systems (Heart monitors, BAN: Body Area Networks)
   - Sensor Networks (distributed Database connected wirelessly)

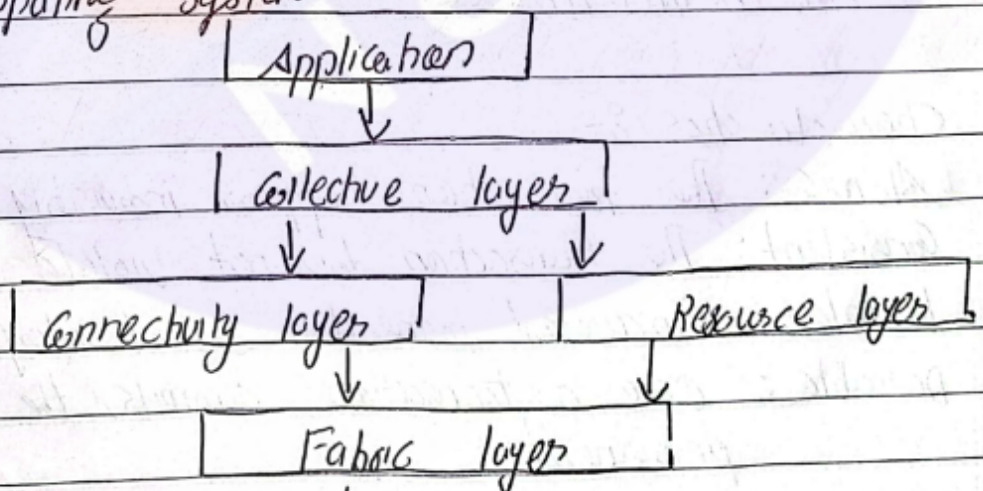
Date
Page

To # Cluster computing system



- Hooking up a collection of simple computers via high-speed networks to build a supercomputing environment
- Mostly homogenous
- Example : Server clusters at Banks

Grid computing System



-> have a high degree of heterogeneity

-> Users and resources from different organizations are brought together to allow collaboration (ie a V.O = Virtual Organization

- Members belonging to the same V.O have access rights to a common set of resources (e.g Police and some local agencies may form a computing grid).

Transaction Processing System (1)

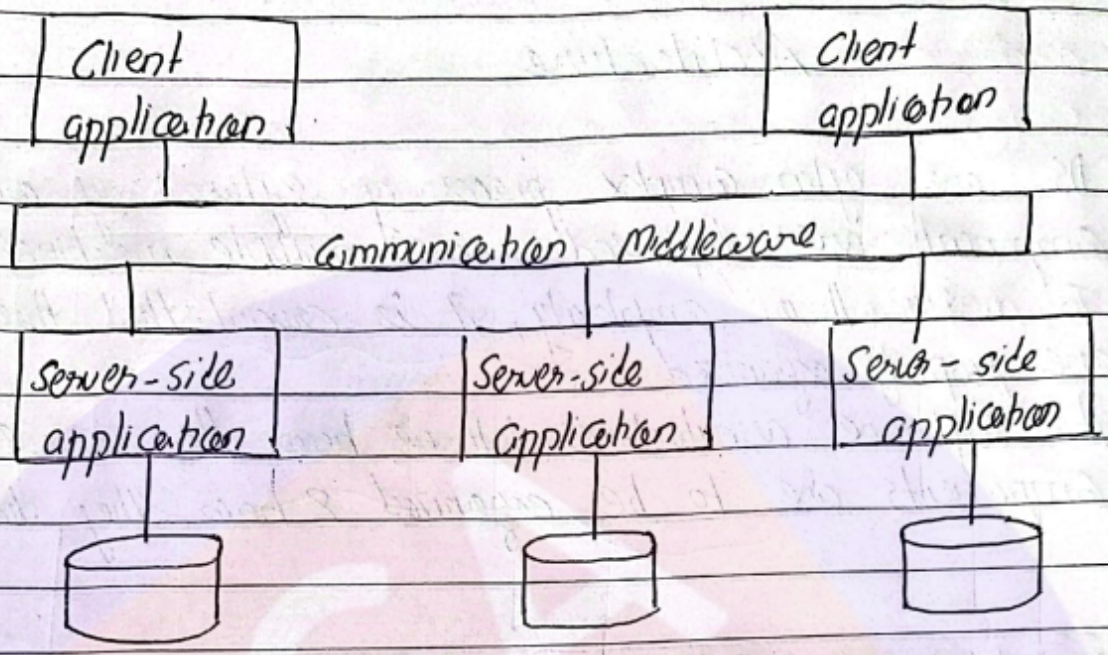
Primitive	Description
BEGIN_TRANSACTION	Mark the start of a transaction
END_ "	Terminate the transaction & try to commit
ABORT_ "	Kill the transaction & restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise

```
BEGIN_TRANSACTION
Salary_1 = doctor1.getSalary()
doctor1.setSalary (salary_1 + bonus)
Salary2 = doctor2.getSalary()
doctor2.setSalary (salary2 - bonus)
END_TRANSACTION
```

Characteristics :-

- Atomic : The transaction happens indivisibly
- Consistent : The transaction does not violate system invariants
- Isolated : Concurrent transactions do not interfere with each other
- Durable : once a transaction commits, the changes are permanent

Enterprise Application Integration



- RPC (Remote Procedure Call)
- RMI (Remote Method Invocation)

Questions → Assignment (1)

i) Define DS. Explain its main characteristics.
ii) Explain in brief about the design goals of DS.
iii) Compare DS with centralized system. What are the advantages & disadvantages of DS over centralized system.
iv) What are the major types of DS. Explain in brief with block diagram.