Chap 1

Database System Architecture

- The ***Architecture of a database system is greatly influenced by the computer system on which the database system runs***. Such as ***Networking, Parallelism & Distribution***
  Networking of computers allows some tasks to be executed on a ***Server System***, & some tasks to be on ***Client Systems***.
- This division of work has led to the development of ***Client-Server Database Systems.***
- ***Parallel Processing*** within a Computer System allows ***Database-System*** activities to be speeded up, allowing faster response to transactions, as well as more transactions per second. The need for ***parallel query processing*** has led to the development of ***Parallel database systems.***
- ***Distributed Data*** across sites in an organization allows those data to reside where they are generated or most needed, but still to be accessible from other sites. Keeping multiple copies of the database across different sites also allows large organizations to continue their database operations even when one site is affected by a natural disaster such as earthquake, flood & fire. ***Distributed Database Systems*** have been developed to handle geographically or administratively distributed data spread across multiple database systems.

1.1 Centralized & Client Server Architecture

- ***Centralized database systems*** are those that run on a ***Single Computer System*** & do not interact with other Computer Systems. Such system spans (covers) a range from Single-user Database Systems running on Personal Computers to high–performance database systems running on Mainframe Systems. ***Client–server systems*** on the other hand, have functionality split between a server system & multiple client systems.

1.1.1 Centralized System

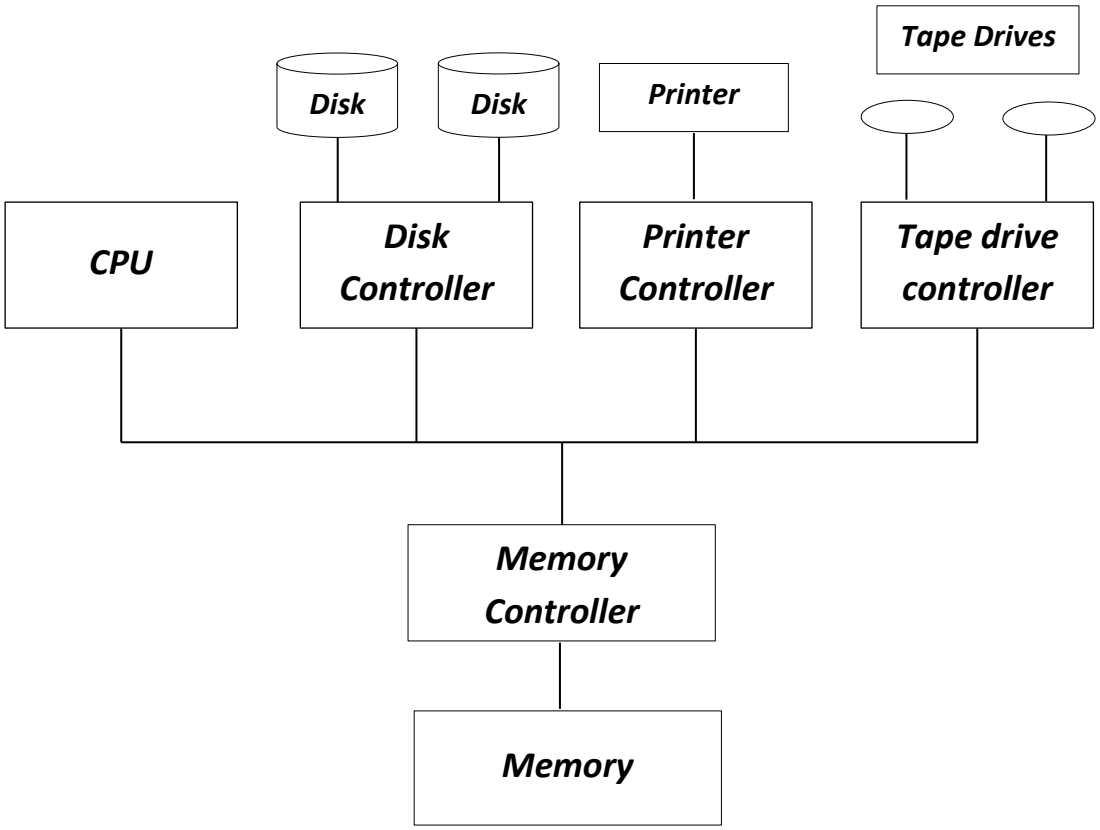


Fig. 1.1 Centralized system

- A modern, general purpose computer system consists of one to a few CPUs and a number of ***device controllers*** that are connected through a common bus that provides access to ***shared memory.*** The CPUs have ***local cache memories*** that stores local copies of parts of the memory, to speed up access to data. ***Each device controller is in charge of a specific type of device***. The ***CPU & device controllers*** can execute concurrently, competing for memory access.
- ***Cache memory*** reduces the ***Contention of Memory*** access, since it reduces the number of times that the ***CPU*** needs to access the ***Shared Memory***. We distinguish two ways in which Computers (*Centralized Computers*) are used, as: ***Single-User System & Multi-User Systems.***
- ***Personal Computer & Workstations*** falls into the first category. A typical single-user is a desktop unit used by a single person has only one CPU & one or two hard disks, & has ***an OS that may support only one user***.

- ***A typical multiuser system***, on the other hand, has more disks & more memory, may have multiple CPUs and has a ***multiuser OS.*** It serves a large number of users who are connected to the system via terminals. Such systems are often called ***Server System***.
- Database systems designed for ***Single-User Systems***, such as personal computers do not provide many of the facilities that a ***multiuser database*** provides. In particular they do not support ***Concurrency Control***, *which is not required when only a single user can generate updates.* The ***crash recovery*** *facilities in such systems are either absent or primitive* (simple) for e.g. simply making a backup of the database, before any update.

1.1.2 Client-Server Systems

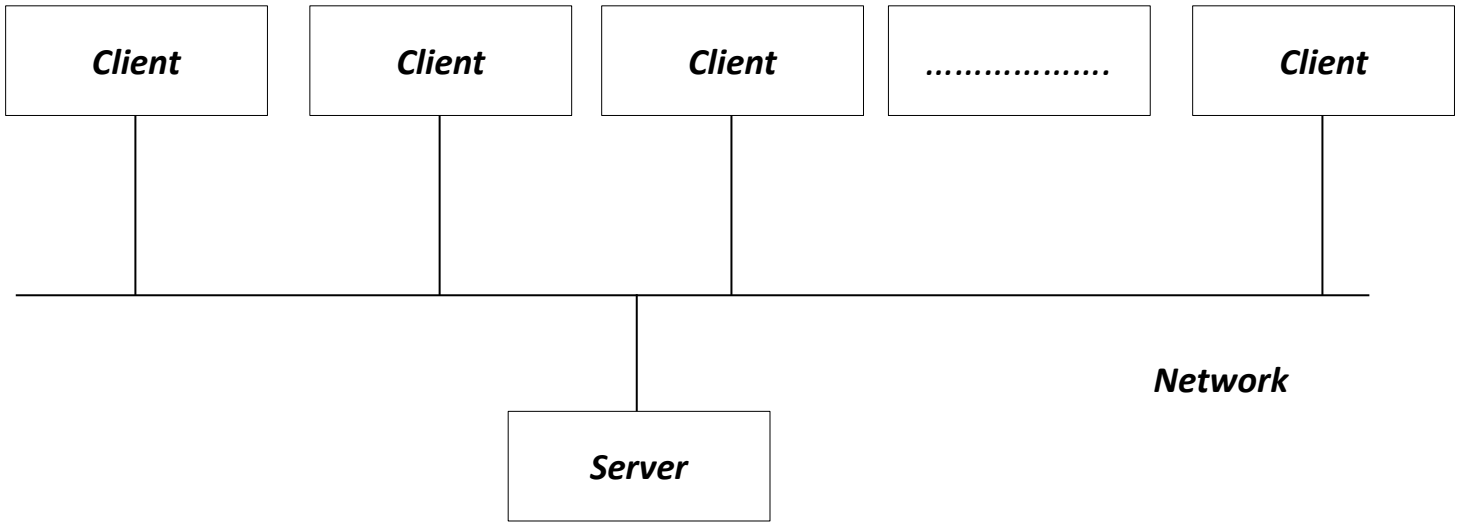


Fig 1.2 Client Server system

- As ***Personal Computers*** *became faster, more powerful & cheaper, there was a shift away from the* ***Centralized system architecture***. ***Personal Computers*** supplanted(replaced) ***Terminals*** *connected to* ***centralized systems***. *Centralized system today as a* ***Server Systems*** that satisfy requests generated by ***client systems***. Above Fig shows a general structure of a client-server system.

- Functionality provided by database can be broadly divided into two parts ***Front end & Backend.*** The ***Backend*** manages ***access structures, query evaluation & optimization, concurrency control, & recovery***. The ***Front end*** of a database system consists of tools such ***as SQL user interface, forms interfaces, report generation tools & data mining & analysis tools.*** The interface between the ***front end & backend*** is through ***SQL***, or through an ***application program***.

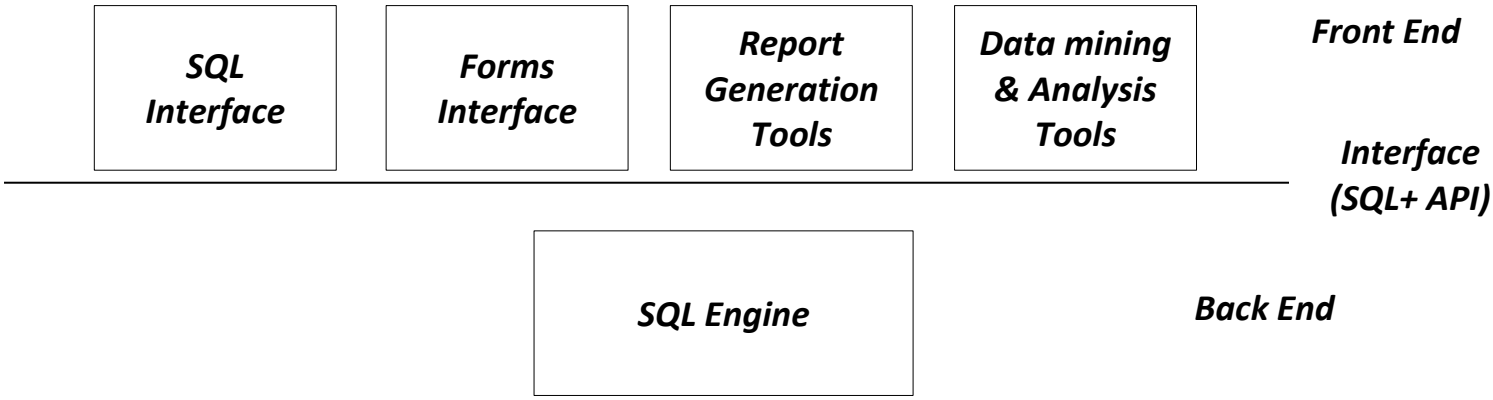


Fig 1.3 Front end & Back end Functionality

1.2 Server System Architecture

- Servers can be broadly categorized as ***Transaction Servers & the Data Servers, Transaction Servers*** systems, also called ***Query-Server Systems***, *provide an interface to which clients can send requests, to perform an action, in response to which they execute the action & send back result to the client.*
- *Usually,* ***client machines ship transactions to the server systems, where those transactions are executed & results are shipped back to clients.*** Requests may be specified by using ***SQL*** or through a ***specialized Application Program Interface***.
- ***Data Servers*** systems allow clients ***to interact with the servers by making requests to read or update data, in units such as files or pages.*** For e.g. *File Server provide a file-system interface where clients can create, update, read, & delete files. (google drive) Data servers for database systems offer much more functionality; they support units of data such as pages, tuples or objects that are smaller than a file.*

1.3 Parallel Systems

- ***Parallel systems improve processing & I/O speeds using multiple CPUs & disk in parallel.***
- *The* ***driving force*** *behind* ***parallel database systems is the demands of applications that have to query extremely large databases*** *or* ***that have to process an extremely large number of transactions per second****.* ***Centralized & client-server systems are not powerful enough to handle such applications.***
- *In parallel processing, many operations are performed simultaneously, as opposed to a serial processing, in which computational steps are performed sequentially.*
- There are two main measures of performance of a database system:
    1. ***Throughput:*** *The number of tasks that can be completed in a given time interval.*
    2. ***Response time***: *The amount of time it takes to complete a single task from the time it is submitted.*

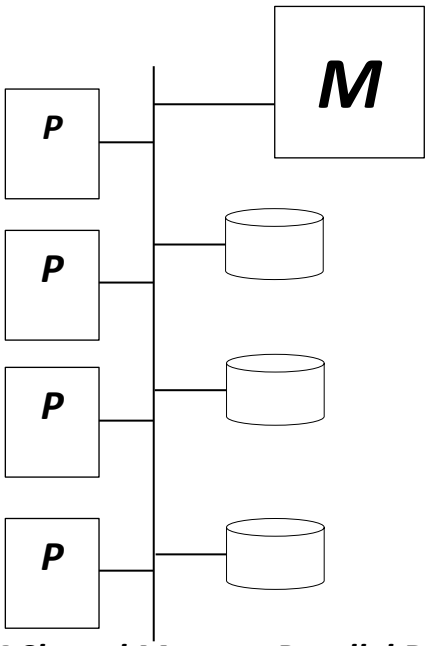
1.3.1 Speedup & Scaleup

- Two important issues in studying parallelism are ***speedup & scaleup***.
- *Running a given task in less time by increasing the degree of parallelism is called* ***speedup****. Handling larger tasks by increasing the degree of parallelism is called* ***scaleup.***

1.3.2 Parallel Database Architectures

- There are several architectural models for Parallel ***Machines***.
    - ***Shared Memory***
    - ***Shared Disk***
    - ***Shared Nothing***
    - ***Hierarchical***

1.3.2.1 Shared Memory



P- *Processor*

M- *Memory*

Fig 1.4 Shared Memory Parallel Database Architecture

- In a ***Shared Memory*** architecture, the ***processors & 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 using memory writes,*** which messages are much faster than are messages sent with a communication mechanism.
- The ***downside of Shared-Memory machines is that the architecture is not scalable beyond 32 or 64 processors,*** since the Bus or interconnected network becomes a bottleneck (as 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.*** *However,* ***at least some of the data will not be in the cache*** *& accesses will have to go to the shared memory.*

- *Moreover, the caches need to be kept coherent; that is , if a processor performs a write to a memory location , the data in that memory location should be either updated at or removed from any processor where the data are cached. Maintaining cache coherency becomes an increasing overhead with increasing number of processors.* Consequently, *shared memory machines are not capable of scaling up beyond a point; current shared memory machines cannot support more than 64 processors.*

1.3.2.2 Shared Disk

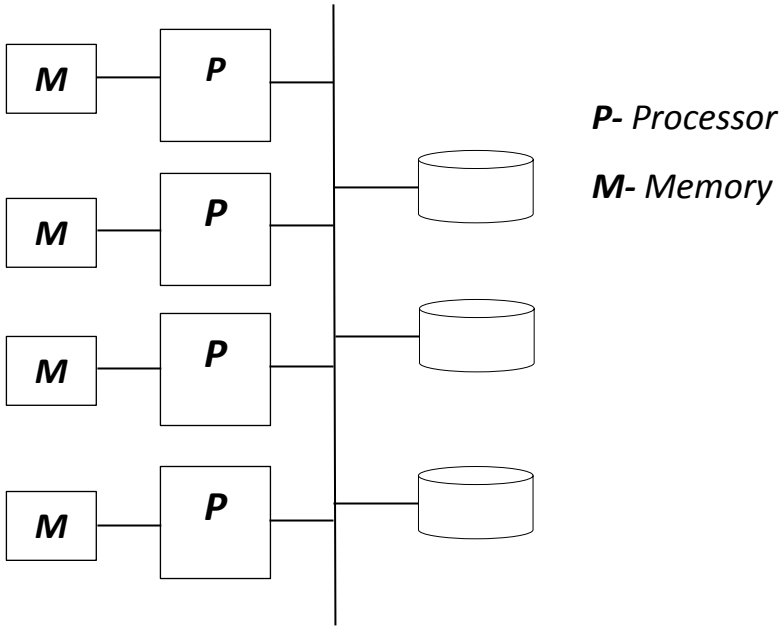


Fig 1.5 Shared Disk Parallel Database Architecture

- 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 benefits for shared-disk architecture over shared memory architecture.
- ***First,*** *since processor has its own memory,* ***the memory bus is not a bottleneck.***
- ***Second****, this architecture offers* ***a cheap way to provide a degree of fault tolerance****:* if ***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 **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, shared-disk systems** can scale to a somewhat larger number of processors, but communication across processors is slower since it has to go through a communication network*

1.3.2.3 Shared Nothing

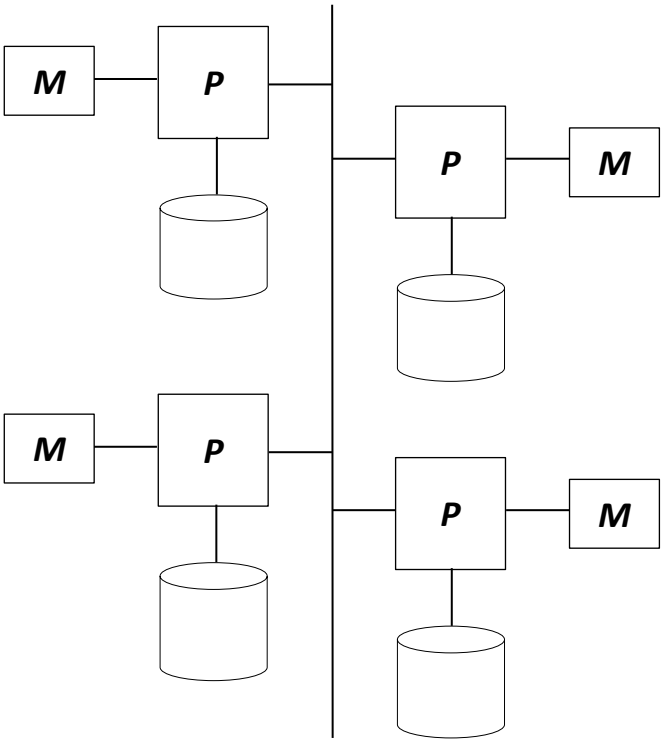


Fig 1.6 Shared Nothing Parallel Database Architecture

- In a ***shared nothing system***, *each node of the machine consists of a processor, memory & one or more disks. The processors at one node may communicate with another processor at another node by a high speed interconnection network.* ***A node functions as the server for data on the disk or disks that the node owns.***
- *Since local disk references are serviced by local disks at each processor, the shared-nothing model overcomes the disadvantage of requiring all I/O to go through a single interconnection network; only queries, accesses to nonlocal disks & result relations pass through the network*.

1.3.2.4 Hierarchical

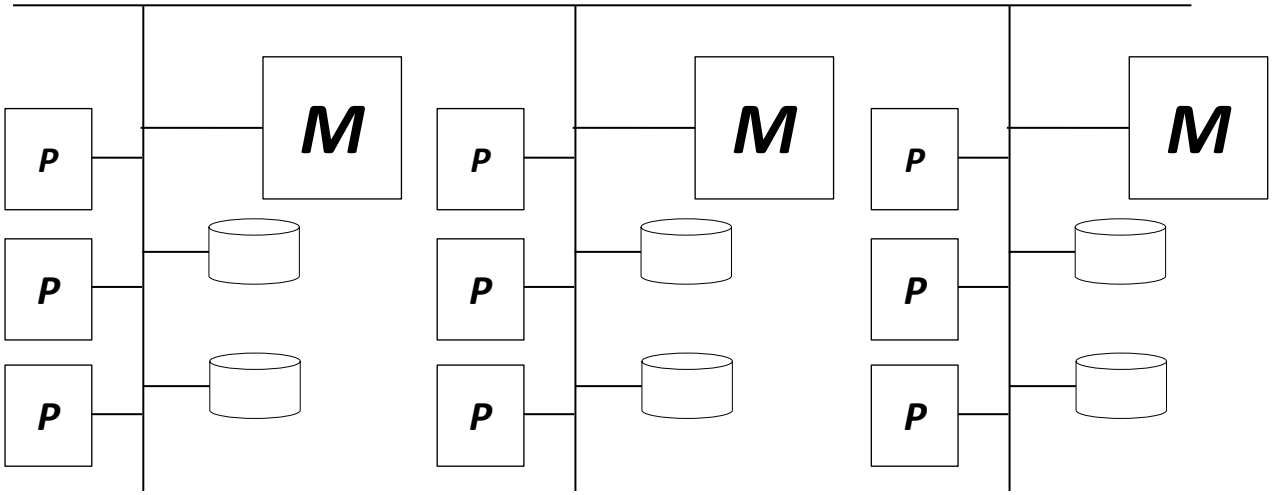
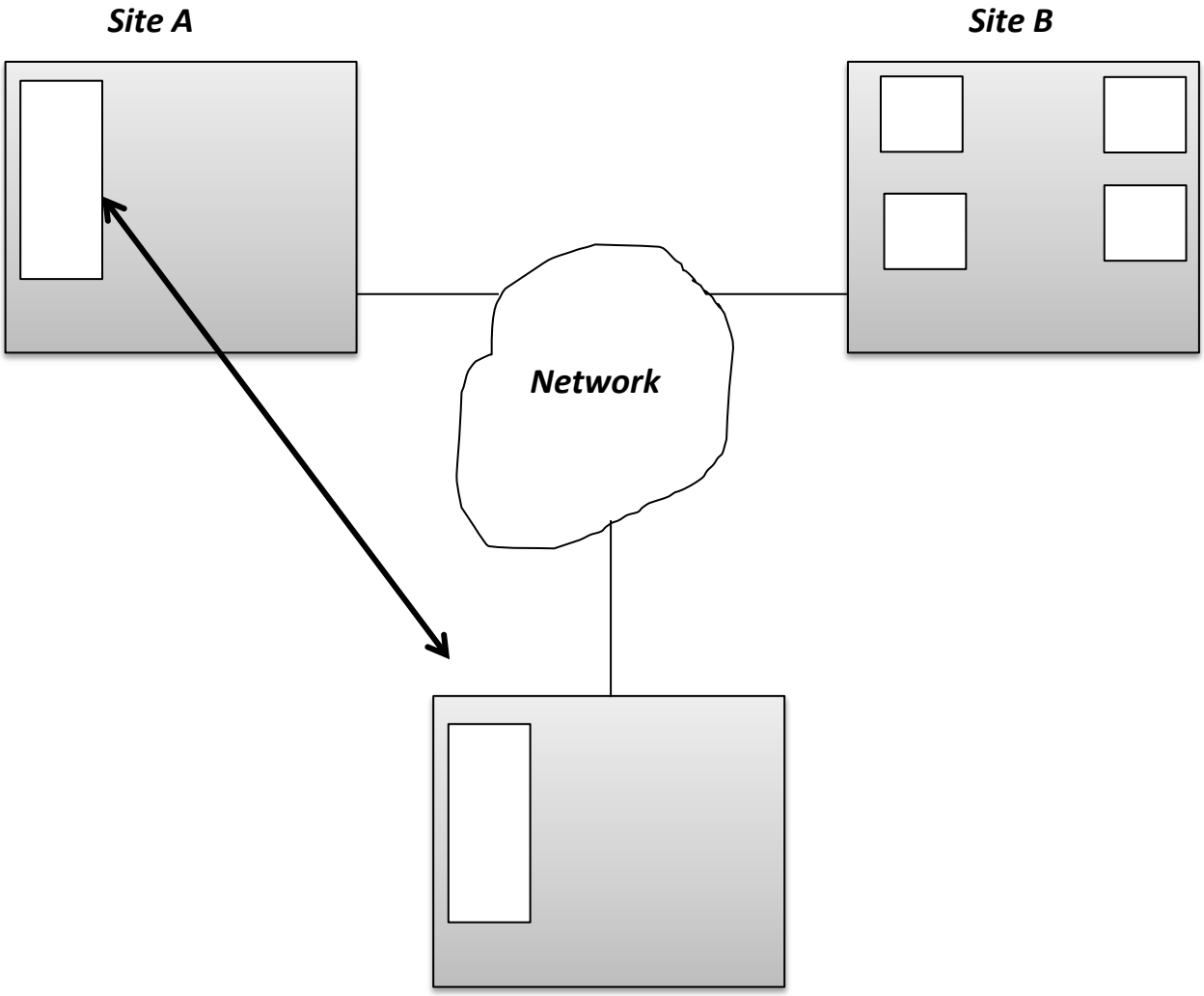
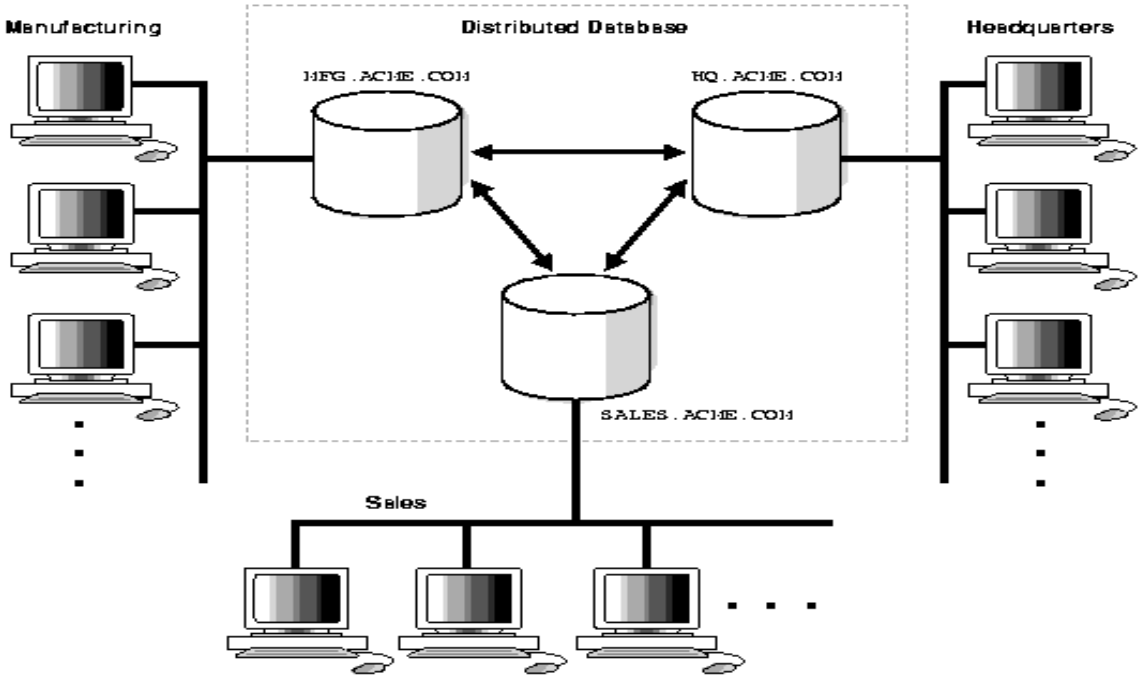


Fig 1.7 Hierarchical Parallel Database Architecture

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

1.4 Distributed Database Systems





- In ***a distributed database system***, ***the database is stored on several computers.***
- *The computers in distributed system communicate with one another through various communication media, such as high speed networks or telephone lines, they do not share memory or disk. The computers in distributed system are referred by a number of names, such as* ***sites or nodes.***
- *The* ***main difference between shared nothing parallel databases & distributed databases are that distributed database are typically geographically separated, are separately administered, & have a slower interconnection****. Another major difference is that, in* ***a distributed database we differentiate between local & global transactions.***
- ***A Local Transaction*** *is one that accesses data only from sites where the transaction was initiated.*
- ***A Global Transaction*** *on the other hand, is one that either accesses data in a site different from the one at which the transaction was initiated, or accesses data in several different sites.*
- There are ***several reasons for building distributed database systems,*** including ***sharing of data, autonomy, & availability***.
  1. ***Sharing Data*****:** *The major advantage in building* ***a distributed database system*** *is the provision of an environment where* ***users at one site may be able to access the data residing at other sites.***
  2. ***Autonomy:*** The ***primary advantage of sharing data by means of data distribution is that each site is able to retain a degree of control over data that are stored locally.*** *In a centralized system,* ***the database administrator of the central site controls the database.*** *In a distributed system, there is* ***global database administrator responsible for the entire system.*** A part of these responsibilities is delegated (passed on) to the ***local database administrator*** for each site. .

3. ***Availability:*** If ***one site fails in a distributed system, the remaining sites may be able to continue operating.*** In particular, if data items are ***replicated in several sites***, a transaction needing a particular data item may find that item in any of several sites. Thus, ***the failure of a site does not necessarily imply the shutdown of the system.***

- *The distributed database system running the same Database Management System S/w is called an* ***Homogeneous Distributed Database System.***
- *The distributed database system that runs different Database Management System S/w is called an* ***Heterogeneous Distributed Database System.***

1.5 Network Types

- *LAN (Local Area Network)*
- *MAN (Metropolitan Area Network)*
- *WAN (Wide Area Network*