Chapter 16: Database System Architectures

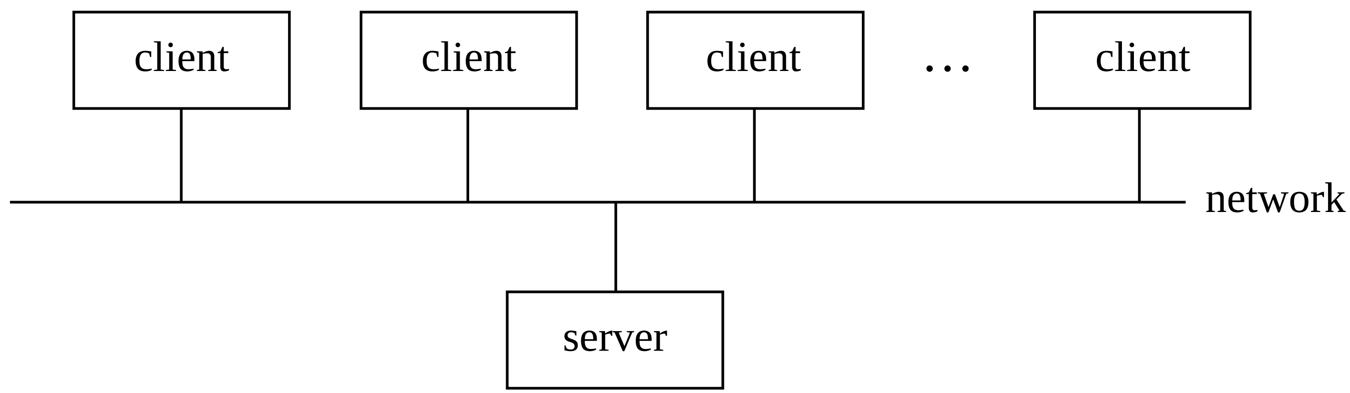
- Centralized Systems
- Client–Server Systems
- Parallel Systems
- Distributed Systems
- Network Types

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, 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 called *server* systems.

Client-Server Systems

- Server systems satisfy requests generated at *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.
- Server systems can be broadly categorized into two kinds: *transaction servers* and *data servers*.

Transaction Servers

- Also called *query server* systems or SQL *server* systems; clients send requests to the server system where the transactions are executed, and results are shipped back to the client.
- Requests specified in SQL, or through an application program interface such as ODBC, using a *remote procedure call* mechanism.
- 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

Data Servers

- Used in LANs: very high speed connection between the clients and the server, the client machines are comparable in processing power to the server machine, and the tasks to be executed are compute intensive.
- Ship data to client machines where processing is performed, and then ship results back to the server machine.
- This architecture requires full back-end functionality at the clients.
- Issues:
  - Page-Shipping versus Item-Shipping
  - Locking
  - Data Caching
  - Lock Caching

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” 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 larger system which is N-times larger.
  - Measured by:
    $$speedup = \frac{small\ system\ elapsed\ time}{large\ system\ elapsed\ time}$$
  - Speedup is linear if equation equals N.
- Scaleup: increase the size of both the problem and the system
  - N-times larger system performs N-times larger job in same time as original system.
  - Measured by:
    $$scaleup = \frac{small\ system\ small\ problem\ elapsed\ time}{big\ system\ big\ problem\ elapsed\ time}$$
  - Scaleup is linear if equation equals 1.

Batch and Transaction Scaleup

Batch scaleup:

- A single large job; typical of most database queries and scientific simulation.
- Use an N-times larger computer on N-times larger problem.

Transaction scaleup:

- Numerous small queries submitted by independent users to a shared database; typical transaction processing and timesharing systems.
- N-times as many users submitting N-times as many requests to an N-times larger database.
- Well-suited to parallel execution.

Factors Limiting Speedup and Scaleup

- **Startup**: Cost of initiating processes may dominate computation time.

- **Interference**: New processes accessing shared resources impose overall slowdown.

- **Skew**: Increasing the number of parallel steps increases the variance in service times.

Interconnection Networks

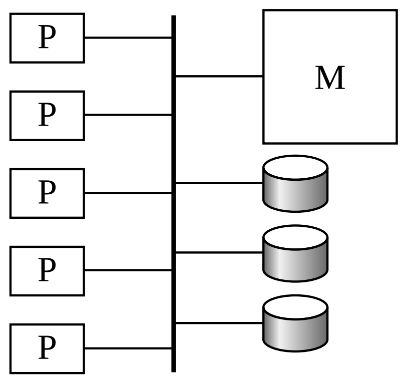
- **Bus**. System components send data on and receive data from a single communication bus; do not scale well with increasing parallelism.
- **Mesh**. Components are arranged as nodes in a grid, and each component is connected to all adjacent components; communication links grow with growing number of components, and so scales well.
- **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.

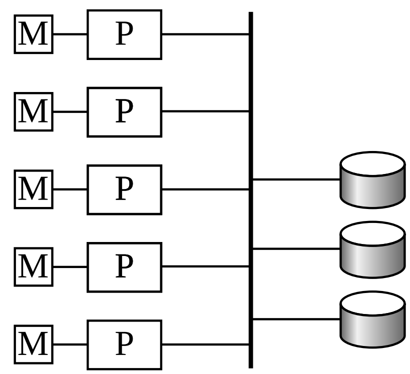
Parallel Database Architectures

- 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

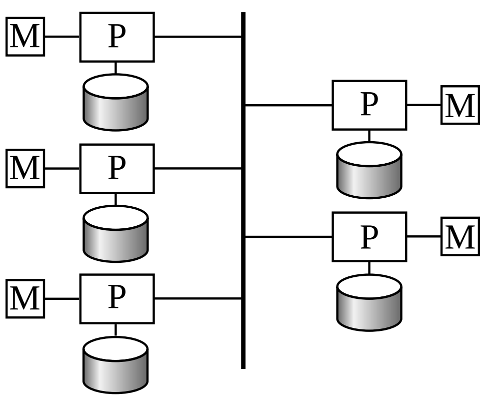
Architecture Structure



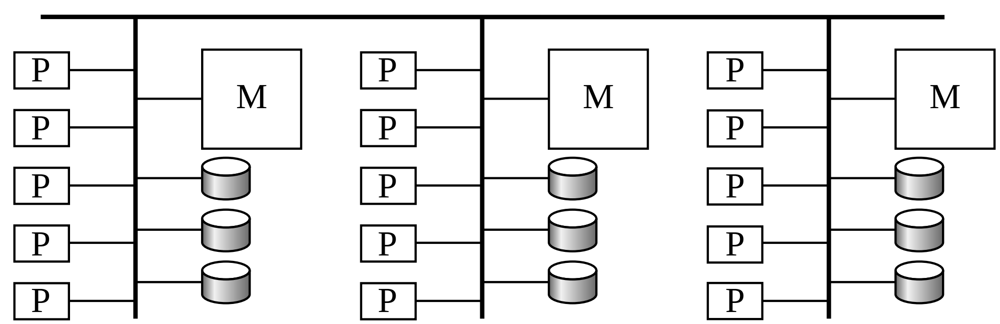
(a) Shared memory



(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.
- Examples: IBM/370, VAX, Sequent
- 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

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, Digital VAXcluster; DEC clusters running Rdb were early commercial users (now Oracle Rdb)
- 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-nCUBE
- Only queries and answers are passed through the 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.

Distributed Systems

- Data spread over multiple machines
- Shared data by users on multiple machines
- Network
- Illusion of a non-distributed system
- Differentiate between *local* and *global* transactions
  - A local transaction accesses data in the *single* site at which the transaction was initiated.
  - A global transaction either accesses data in a site different from the one at which the transaction was initiated or accesses data in several different sites.

Trade-offs in Distributed Systems

- Sharing data – users at one site able to access the data residing at some other sites.
- Autonomy – each site is able to retain a degree of control over data stored locally.
- Availability through redundancy.
- Disadvantage: added complexity required to ensure proper coordination among sites.
  - Software development cost.
  - Greater potential for bugs.
  - Increased processing overhead.

Network Types

- Local-area networks (LANs) – composed of processors that are distributed over small geographical areas, such as a single building or a few adjacent buildings.
- Wide-area networks (WANs) – composed of processors distributed over a large geographical area.
  - Discontinuous connection – WANs, such as those based on UUCP, that are connected only for part of the time.
  - Continuous connection – WANs, such as the Internet, where hosts are connected to the network at all times.