Centralized and Client-Server Systems:
Introduction:

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.

Client Server Systems:

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

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 Server**

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.
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 More on this

Later Database writer process Output modified buffer blocks to disks continually.

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.

Shared memory contains shared data

- Buffer pool
- Lock table
- Log buffer
- Cached query plans
Data Server

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.
Used in many object-oriented database systems

Issues:
Page-Shipping versus Item-Shipping
Locking
Data Caching
Lock Caching
1. Page-shipping versus item-shipping
   Smaller unit of shipping ⇒ more messages
   Worth prefetching related items along with requested item
   Page shipping can be thought of as a form of prefetching
2. 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.
3. 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)
4. Lock Caching
   Locks can be retained by client system even in between transactions
   Transactions can acquire cached locks locally, without contacting server
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:
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.

There are two factors influence the performance of Parallel systems:
1. Speedup
2. Scaleup

Speedup and scaleup are often sublinear due to:
- Startup costs: Cost of starting up multiple processes may dominate computation time, if the degree of parallelism is high.
- 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.

Parallel database system architecture
- Shared memory -- processors share a common memory.
  - Processors and disks have access to a common memory, typically via a bus or through an interconnection network.
  - 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 -- processors share a common 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.

- Shared nothing -- processors share neither a common memory nor common disk.
  - 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-n CUBE
- Hierarchical -- hybrid of the above architectures.
  - 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.

**Distributed Systems**

Data spread over multiple machines (also referred to as sites or nodes). Network interconnects the machines. Data shared by users on multiple machines
Homogeneous distributed databases
Same software/schema on all sites, data may be partitioned among sites Goal:
provide a view of a single database, hiding details of distribution

Heterogeneous distributed databases
Different software/schema on different sites
Goal: integrate existing databases to provide useful functionality

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.

Characteristics of DDB  Disadvantage
• Sharing Data  Software Development cost
• Autonomy  Greater potential for bugs.
• Higher availability  Increased processing overhead

Implementation issues of Distributed databases
Atomicity needed even for transactions that update data at multiple sites The
two-phase commit protocol (2PC) is used to ensure atomicity
Basic idea: each site executes transaction until just before commit, and the leaves final
decision to a coordinator
Each site must follow decision of coordinator, even if there is a failure while waiting for
coordinators decision
2PC is not always appropriate: other transaction models based on persistent messaging, and
workflows, are also used.
Distributed concurrency control (and deadlock detection) required Data items may be
replicated to improve data availability.

This is a SAMPLE (Few pages have been extracted from the complete notes:-It’s meant to show you the topics covered in the full notes and as per the course outline

Download more at our websites:
www.naarocom.com
To get the complete notes either in softcopy form or in Hardcopy (printed & Binded) form, contact us on:

Call/text/whatsapp +254 719754141/734000520

Email:
naarocom@gmail.com
info@naarocom.com
sales@naarocom.com

Get news and updates by liking our page on facebook and follow us on Twitter

Sample/preview is NOT FOR SALE