Distributed DB architectures: An Introduction

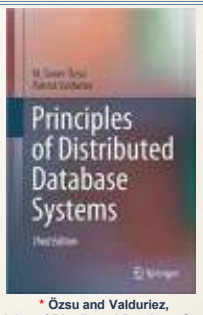
Data Management for Big Data
2018-2019 (spring semester)

Dario Della Monica

These slides are a modified version of the slides provided with the book
Özsu and Valduriez, *Principles of Distributed Database Systems* (3rd Ed.), 2011
The original version of the slides is available at: extras.springer.com

Outline (distributed DB)

- Introduction (Ch. 1) *
  - What is a distributed DBMS
  - Distributed DBMS Architecture
- Distributed Database Design (Ch. 3) *
- Distributed Query Processing (Ch. 6-8) *
- Distributed Transaction Management (Ch. 10-12) *



* Özsu and Valduriez, *Principles of Distributed Database Systems*, 3rd edition, 2011

Outline (today)

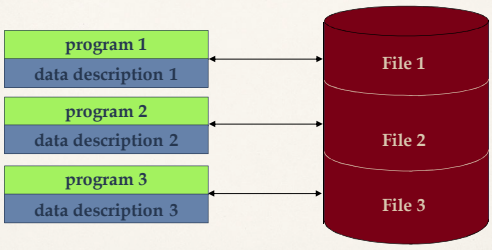
- Introduction (Ch. 1) *
  - Introduction to distributed processing
  - What is (not) a Distributed Database System (DDBS)
  - Data delivery alternatives
  - Promises of DDBS
    - Transparency and its levels
    - Reliability (introduction to the problem of distributed transactions)
    - Improved performances
    - Easier system expansion
  - Design issues (deriving for the promises)
  - Classification of Distributed DBMS (D-DBMS)
    - Dimensions of the classification (autonomy, distribution, heterogeneity)
    - Different D-DBMS Architectures

* Özsu and Valduriez, *Principles of Distributed Database Systems* (3rd Ed.), 2011

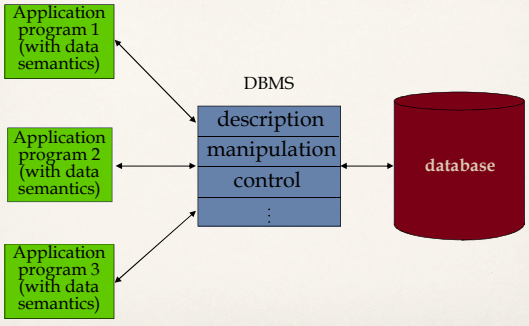
Centralization, distribution, integration

- Database philosophy
  - Separation between application logic and data
  - Centralization (integration) of data
  - Transparency, data independence, access control
- Computer network
  - Distributed applications
  - Distribution of data (big data)
  - Concurrency, redundancy (backup), localization/proximity
- Distributed databases
  - Centralization data is logical
  - Distribution of data is physical
  - Integration

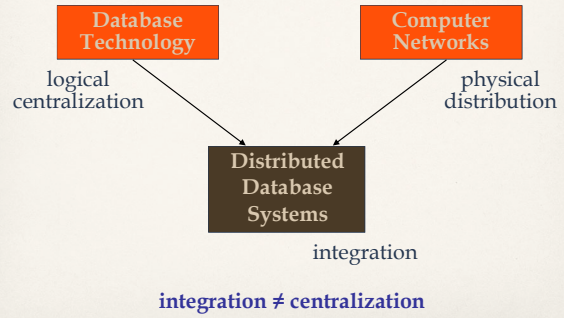
File Systems



Database Management



Motivation



integration ≠ centralization

Distributed Computing

- Some forms of distributed processing are everywhere (CPU vs. I/O, parallel computation, multi-processor, multi-core)
- **Definition (distributed computing).** A number of autonomous processing elements (not necessarily homogeneous) that are interconnected by a computer network and that cooperate in performing their assigned tasks

Distribution: what and why?

- What is being distributed?
  - Processing logic (pieces of computations)
  - Function (tasks that are specific to a piece of hardware or software)
  - Data
  - Control
- Why to distribute?
  - Widely distributed (physically) enterprises
  - Reliability
  - More responsive systems
  - More importantly: nowadays applications are intrinsically distributed and so is the data (web-based, e-commerce, social)
  - In one word (actually two): big data → divide-et-impera (divide-and-conquer)

distr. DB systems : distr. processing = DB systems : centr. processing

What is a Distributed Database System?

A distributed database (DDB) is a collection of multiple, *logically interrelated* databases distributed over a *computer network*.

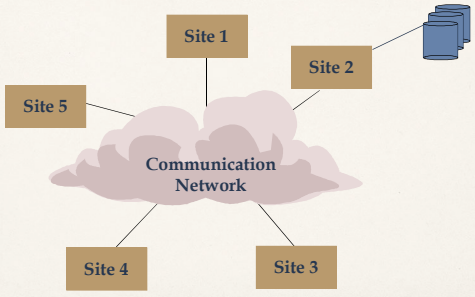
A distributed database management system (D–DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users.

Distributed database system (DDBS) = DDB + D–DBMS

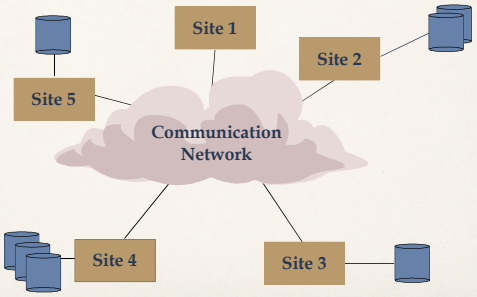
What is not a DDBS?

- Files distributed over a network (missing structure and common logical access interface) → logical interrelation
- A number of related DB that reside on the same system → physical distribution (not necessarily over a wide area, network communication)
- A loosely coupled multiprocessor system → even though communication issues are similar to the one over network (no disk or memory shared), they are not enough heterogeneous (identical processors and OS's) → **parallel DB systems**
- A database system which resides at one of the nodes of a network of computers - this is a centralized database on a network node → multiple DB → **client/server DB systems**

Centralized DBMS on a Network



Distributed DBMS Environment



Implicit Assumptions

- Data stored at a number of sites → each site *logically* consists of a single processor.
- Processors at different sites are interconnected by a computer network → not a multiprocessor system
  - ➡ Parallel database systems
- Distributed database is a database, not a collection of files → data logically related as exhibited in the users' access patterns
  - ➡ Relational data model
- D-DBMS is a full-fledged DBMS
  - ➡ Not remote file system, not a TP system

Data Delivery Alternatives

- Delivery modes
  - ➡ Pull-only
  - ➡ Push-only
  - ➡ Hybrid
- Frequency
  - ➡ Periodic
  - ➡ Conditional
  - ➡ Ad-hoc or irregular
- Communication Methods
  - ➡ Unicast
  - ➡ One-to-many
- Note: not all combinations make sense

Distributed DBMS Promises

1. Transparent management of distributed, fragmented, and replicated data
2. Improved reliability/availability through distributed transactions (tolerance to, e.g., network communication failure and concurrent access)
3. Improved performance
4. Easier and more economical system expansion

Transparency

- Transparency is the separation of the higher level semantics of a system from the lower level implementation issues
- 4 types of transparency. Fundamental issue is to provide
  data independence
  in the distributed environment
  - ➡ Network transparency (or distribution transparency)
    - Location transparency
    - Naming transparency
  - ➡ Replication transparency
  - ➡ Fragmentation transparency
    - horizontal fragmentation: selection
    - vertical fragmentation: projection
    - hybrid

Example

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

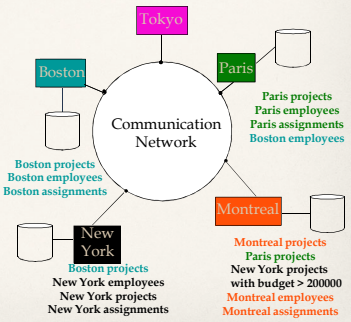
PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

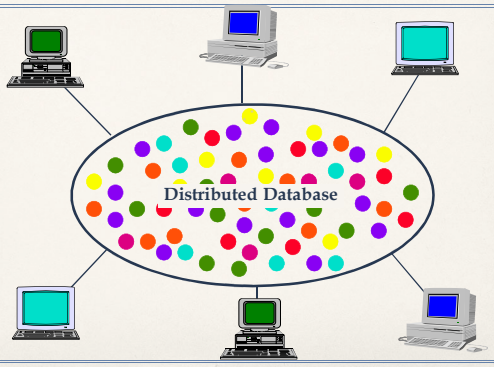
TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Transparent Access

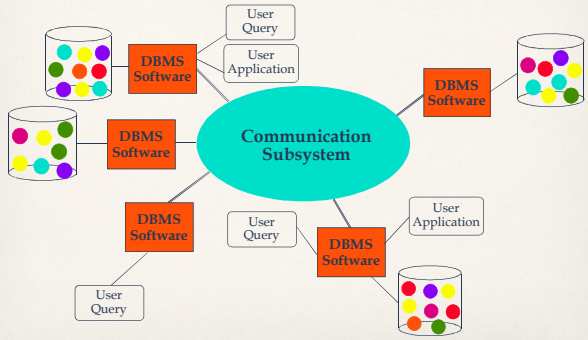
```
SELECT ENAME,SAL
FROM   EMP,ASG,PAY
WHERE  DUR > 12
AND    EMP.ENO = ASG.ENO
AND    PAY.TITLE = EMP.TITLE
```



Distributed Database - User View



Distributed DBMS - Reality



Reliability Through Transactions

- Replicated components and data should make distributed DBMS more reliable
- Transactions: sequences of DB operations executed as atomic actions
- (Distributed) transactions provide
  - ➡ Consistency: bring the DB from a consistent state to another consistent one
  - ➡ Reliability: Concurrency transparency, Failure atomicity
- Distributed transaction support requires implementation of
  - ➡ Distributed concurrency control protocols
  - ➡ Commit protocols
- Data replication (we will not deal with it)
  - ➡ Great for read-intensive workloads, problematic for updates
  - ➡ Replication protocols

Distributed systems are intrinsically problematic

CAP Theorem states that it is impossible for a distributed DB (where data is fragmented and replicated) to provide simultaneously:

- Consistency (always the updated data is read)
- Availability (every request receives a response)
- Partition tolerance (tolerance to network communication failures)

Potentially Improved Performance

- Proximity of data to its points of use
  - ➡ Requires some support for fragmentation and replication
- Parallelism in execution
  - ➡ Inter-query parallelism
  - ➡ Intra-query parallelism

Parallelism Requirements

- Have as much of the data required by *each* application at the site where the application executes
  - ➡ Full replication
- How about updates?
  - ➡ Mutual consistency
  - ➡ Freshness of copies

Easier System Expansion

- Issue is database scaling
- Expansion: as easy as adding a new node in the network (with its own DBMS)
  - ➡ Easy thanks to transparency
- Expanding a centralized DBMS is more difficult than adding a new one

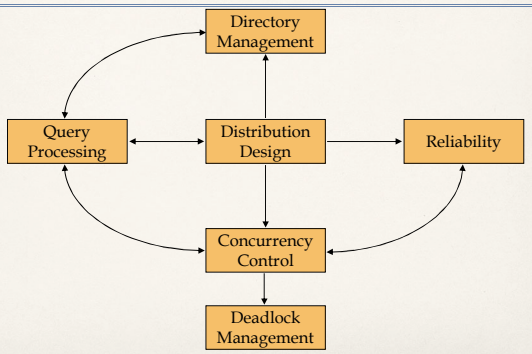
Distributed DBMS Issues

- **Distributed Database Design**
  - ➡ How to distribute the database
  - ➡ Replicated & non-replicated database distribution
  - ➡ A related problem in **directory management**
- **Query Processing**
  - ➡ Convert user transactions to data manipulation instructions
  - ➡ Optimization problem
    - ✦ min{cost = data transmission + local processing}
  - ➡ General formulation is NP-hard

Distributed DBMS Issues

- **Concurrency Control**
  - ➡ Synchronization of concurrent accesses
  - ➡ Consistency and isolation of transactions' effects
- **Distributed Deadlock management**
  - ➡ Prevention, avoidance, detection/recovery techniques
- **Reliability**
  - ➡ How to make the system resilient to failures
  - ➡ Atomicity and durability
- **Replication**
  - ➡ Consistency of the information (eager and lazy protocols)

Relationship Between Issues



Recent Developments

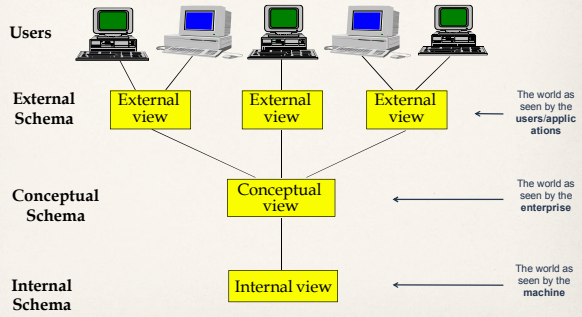
- So far, and in the rest of the course: focus on "traditional" DDBS
- Recent directions
  - ➡ **Multidatabase systems**, aka federated databases, aka data integration systems - *not in this course (Ch. 1.10, 4, 9)* *
    - ✦ Heterogeneity of data source
  - ➡ **Peer-to-peer** "reloaded" **and web data management** - *not in this course (Ch. 16, 17)* *
    - ✦ Rethinking of an old architecture, oriented to needs of data sharing over the web
  - ➡ **Parallel databases** – *maybe at the end of the course (Ch. 14)* *
    - ✦ Specific issues (not really distributed)

* Özsu and Valduriez, *Principles of Distributed Database Systems* (3rd Ed.), 2011

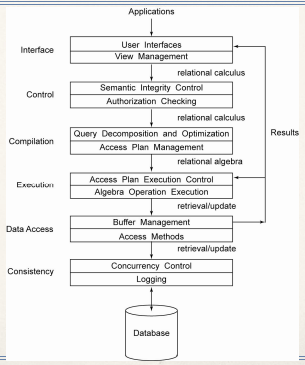
Architecture

- Defines the structure of the system
  - components identified
  - functions of each component defined
  - interrelationships and interactions between components defined

ANSI/SPARC Architecture



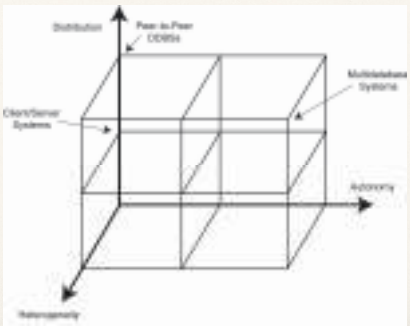
Generic DBMS Architecture



Dimensions of the Problem

- Distribution (of data)
  - Whether the components of the system are located on the same machine or not
    - Non-distributed/*client-server*/***fully distributed*** (peer-to-peer)
- Heterogeneity
  - Hardware, communications, **data model**, **query language**, **transaction management**
- Autonomy (distribution of control)
  - Various versions
    - Design autonomy: ability of a component DBMS to decide on issues related to its own design.
    - Communication autonomy: ability of a component DBMS to decide whether and how to communicate with other DBMSs and what information to share with them.
    - Execution autonomy: ability of a component DBMS to execute transaction operations in any manner it wants to
  - Classification along this dimension:
    - Tight integration (one unique logical view of the DB, control is centralized for each user request)
    - Semiautonomous (independent DBMS, selective data sharing, common communication protocols)
    - Total isolation (*multidatabase systems* – totally independent, unaware of each other and of communication systems, no global control, no unified DB view, each DB decides what to share with who)

DBMS Implementation Alternatives



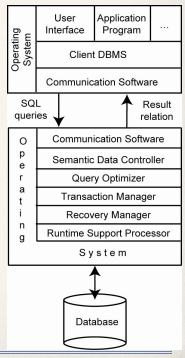
Terminology disambiguation

- In our discussion, by distributed DBMS and DDBS we mostly refer to fully distributed, **peer-to-peer** data management systems
- In particular, the term "peer-to-peer" is overloaded in the book. It is used to refer to:
  - "traditional" peer-to-peer DBMS (also referred to as fully distributed DBMS) → our focus is on these systems [see previous two slides]
  - "modern" peer-to-peer data management systems [see slide slide titled "Recent developments"]
    - These refer to an evolution of the traditional peer-to-peer architecture, to cope with the need of data sharing, e.g., over the web
    - Ongoing research area, related issues still being investigated
    - Out of the scope of this course (see Ch. 16) *

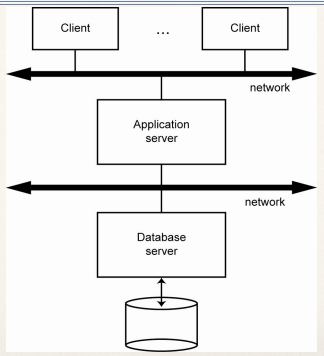
* Özsu and Valduriez, *Principles of Distributed Database Systems* (3rd Ed.), 2011

Client/Server Architecture

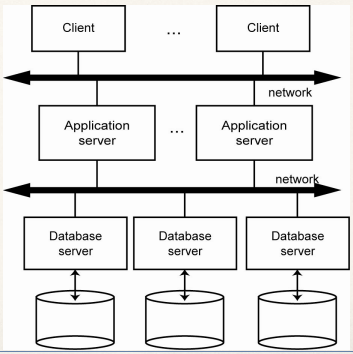
- Not fully distributed: data and control are distributed among few servers (multiple client/multiple servers) or not distributed at all (multiple client/single server)
- Nodes of the network are not identical (divided into two groups: clients and servers)
- DBMS is on server(s) only
- Most of the work is done by server(s)
- On clients: only a DBMS client for sending queries and receiving/displaying results
- Multiple client/single servers: like centralized DBMS (no distribution-related issues)
- Multiple client/multiple servers: mild form of distribution
- Our discussion mainly refers to fully distributed DBMS, but often applies to client/server architectures



Database Server



Distributed Database Servers



Multidatabase system architecture

- Multidatabase system (MDBS) are beyond the scope of this course
- In a nutshell
  - ➡ Characterized by presence of several fully autonomous, independent, and heterogeneous DBMS that are located apart from each other
  - ➡ Unaware of each other existence
  - ➡ Systems do not know how to communicate with other ones
  - ➡ No (explicit) interrelation between DBMS
  - ➡ Each DBMS provides an export schema to make (some of) its data available to (some of) the other systems
  - ➡ In DDBS: unique logical view of the entire DB (transparency) resulting from the "union" of parts of DB stored locally at each node of the network
    In MDBS: not even a clear notion of "entire DB"
  - ➡ Architecture design is built bottom-up (in DDBS it is built top-down)
- If interested, see the discussion at Ch. 1.10, 4, 9 *

* Özsu and Valduriez, *Principles of Distributed Database Systems* (3rd Ed.), 2011

MDBS Components & Execution