Principles of Distributed Database Systems

M. Tamer Özsu

Patrick Valduriez

© 2020, M.T. Özsu & P. Valduriez

Outline

- Introduction
- Distributed and Parallel Database Design
- Distributed Data Control
- Distributed Query Processing
- Distributed Transaction Processing
- Data Replication
- Database Integration – Multidatabase Systems
- Parallel Database Systems
- Peer-to-Peer Data Management
- Big Data Processing
- NoSQL, NewSQL and Polystores
- Web Data Management

© 2020, M.T. Özsu & P. Valduriez

Outline

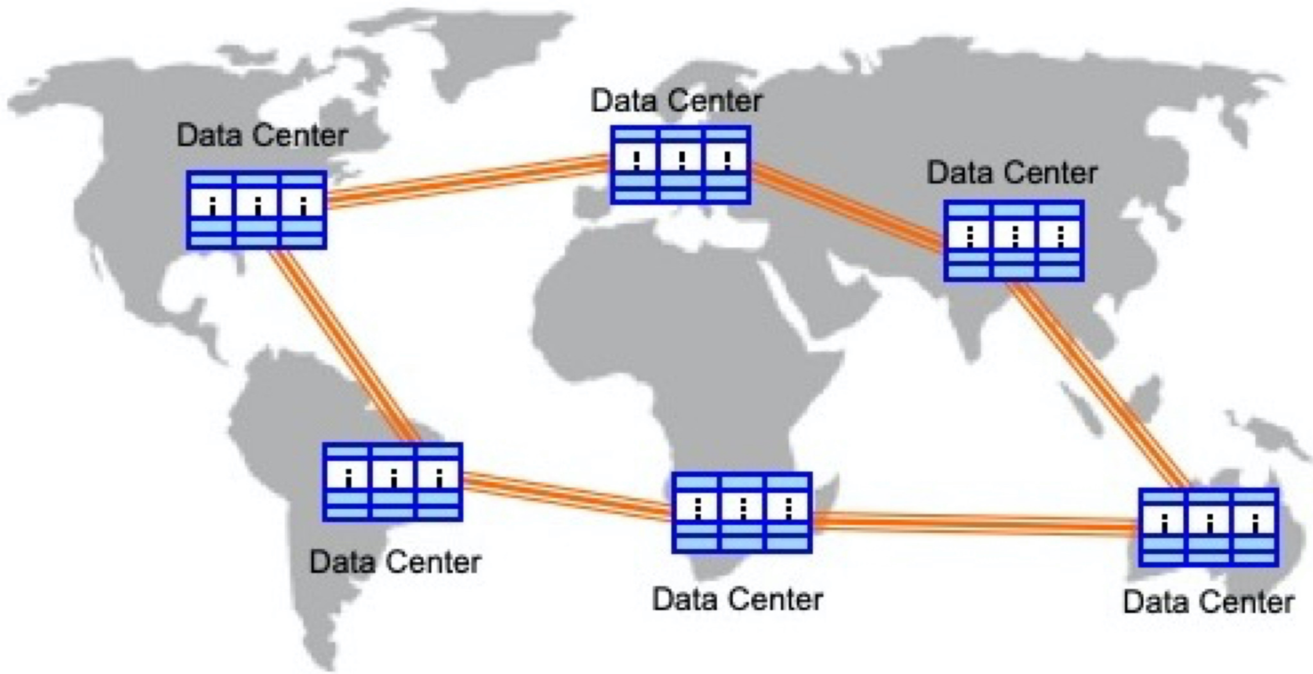
- Introduction
  - What is a distributed DBMS
  - History
  - Distributed DBMS promises
  - Design issues
  - Distributed DBMS architecture

© 2020, M.T. Özsu & P. Valduriez

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.
- What is being distributed?
  - Processing logic
  - Function
  - Data
  - Control

Current Distribution – Geographically Distributed Data Centers



© 2020, M.T. Özsu & P. Valduriez

What is a Distributed Database System?

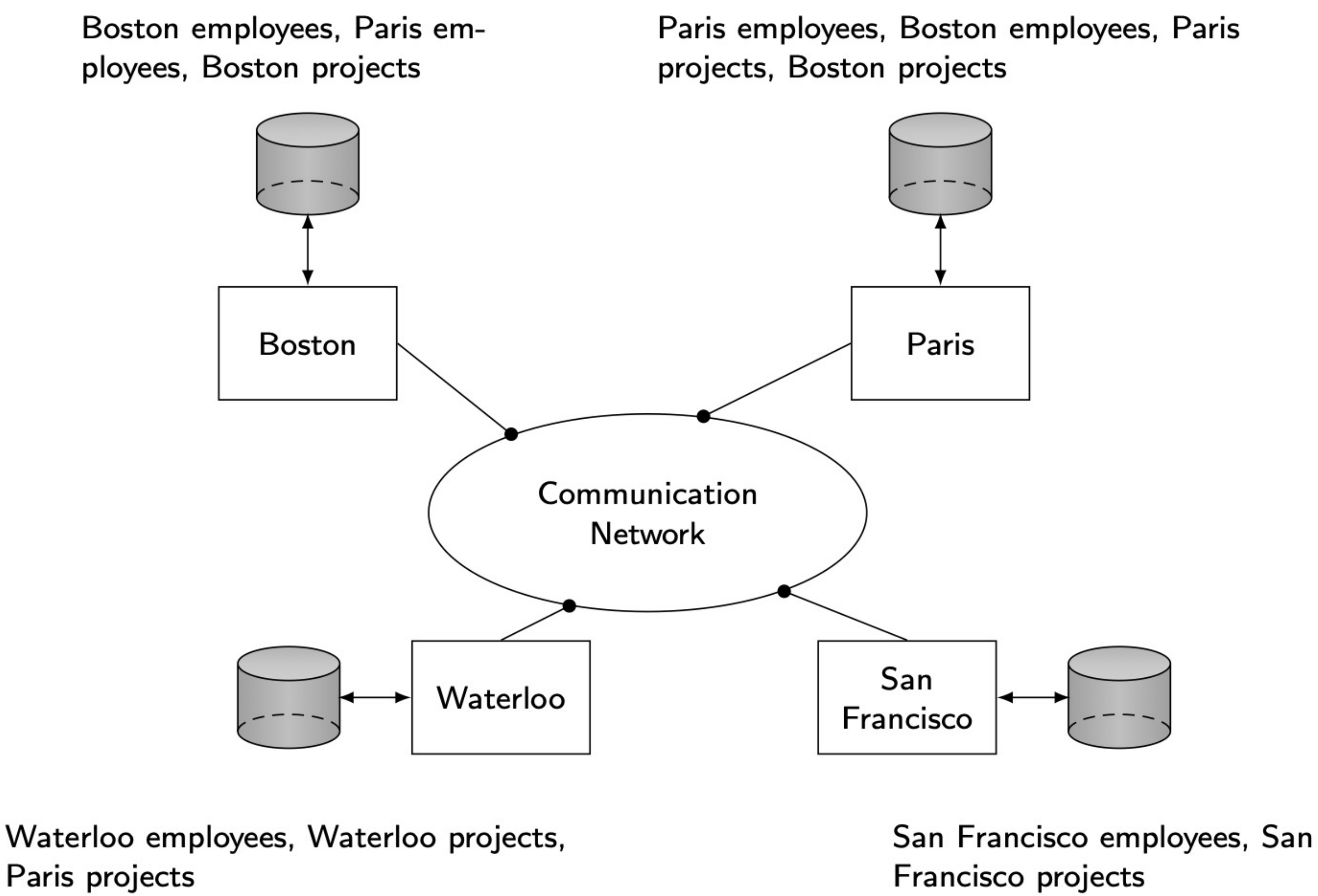
A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network

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

What is not a DDBS?

- A timesharing computer system
- A loosely or tightly coupled multiprocessor system
- A database system which resides at one of the nodes of a network of computers - this is a centralized database on a network node

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
- Distributed DBMS is a full-fledged DBMS
  - Not remote file system, not a TP system

© 2020, M.T. Özsu & P. Valduriez

Important Point

Logically integrated

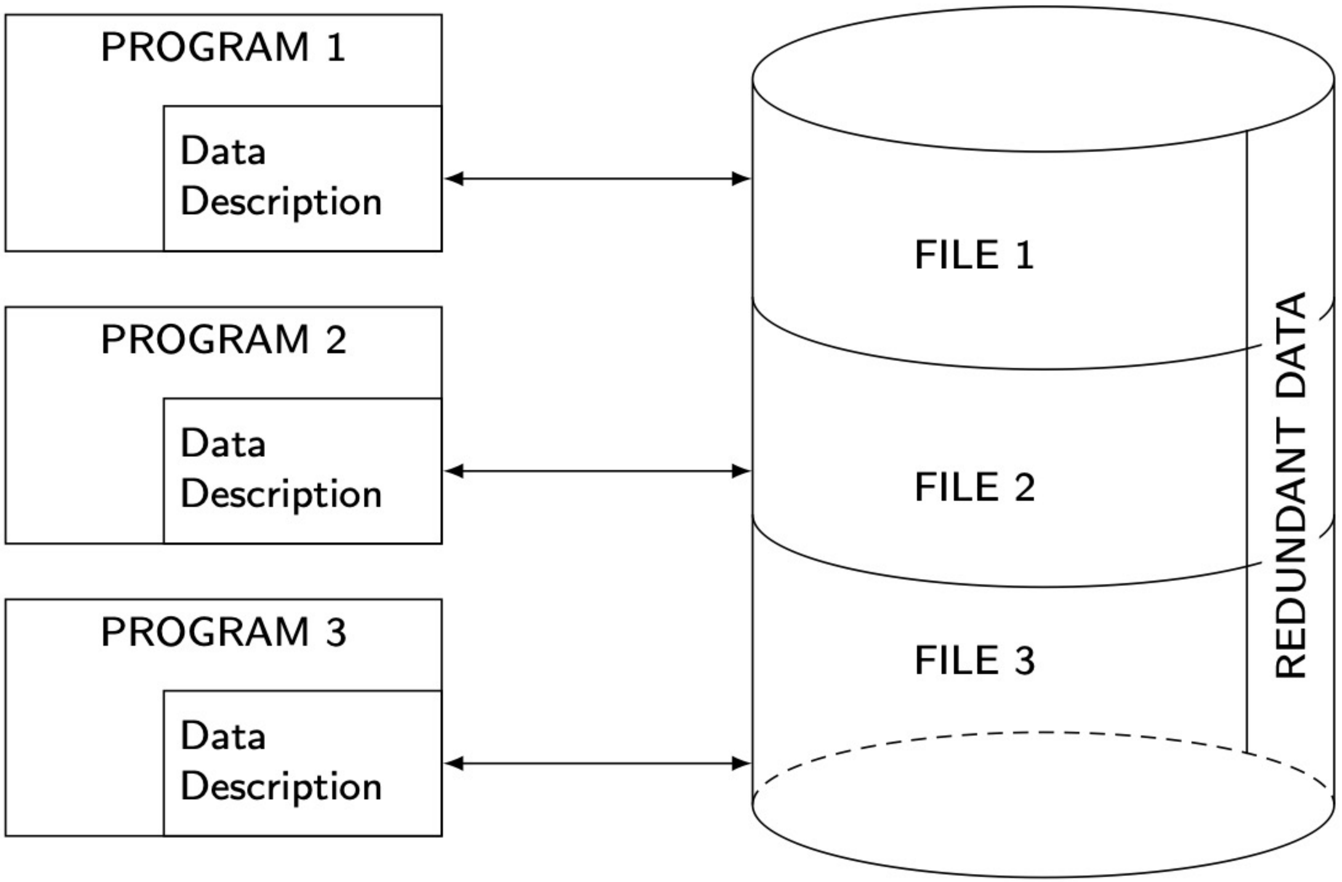
but

Physically distributed

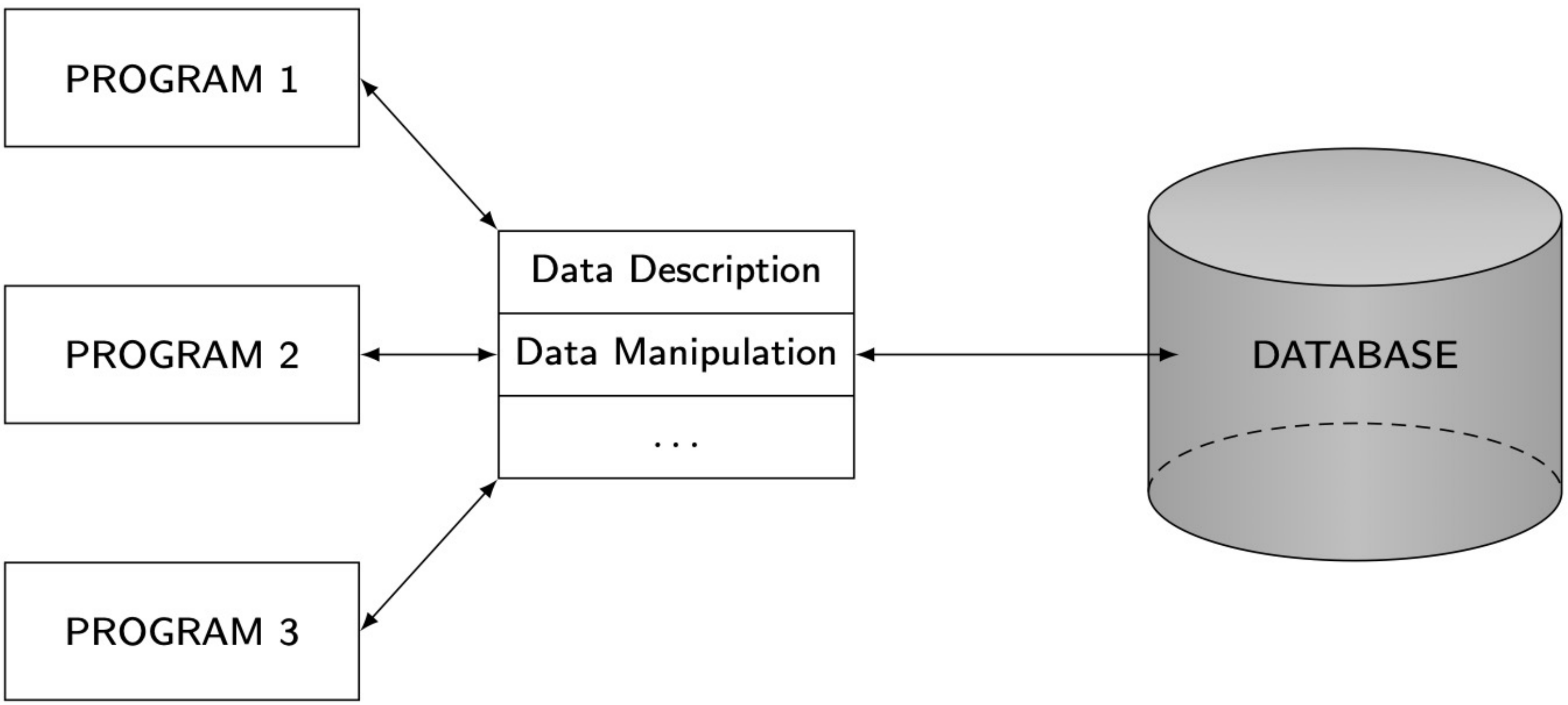
Outline

- Introduction
  - What is a distributed DBMS
  - History
  - Distributed DBMS promises
  - Design issues
  - Distributed DBMS architecture

History – File Systems



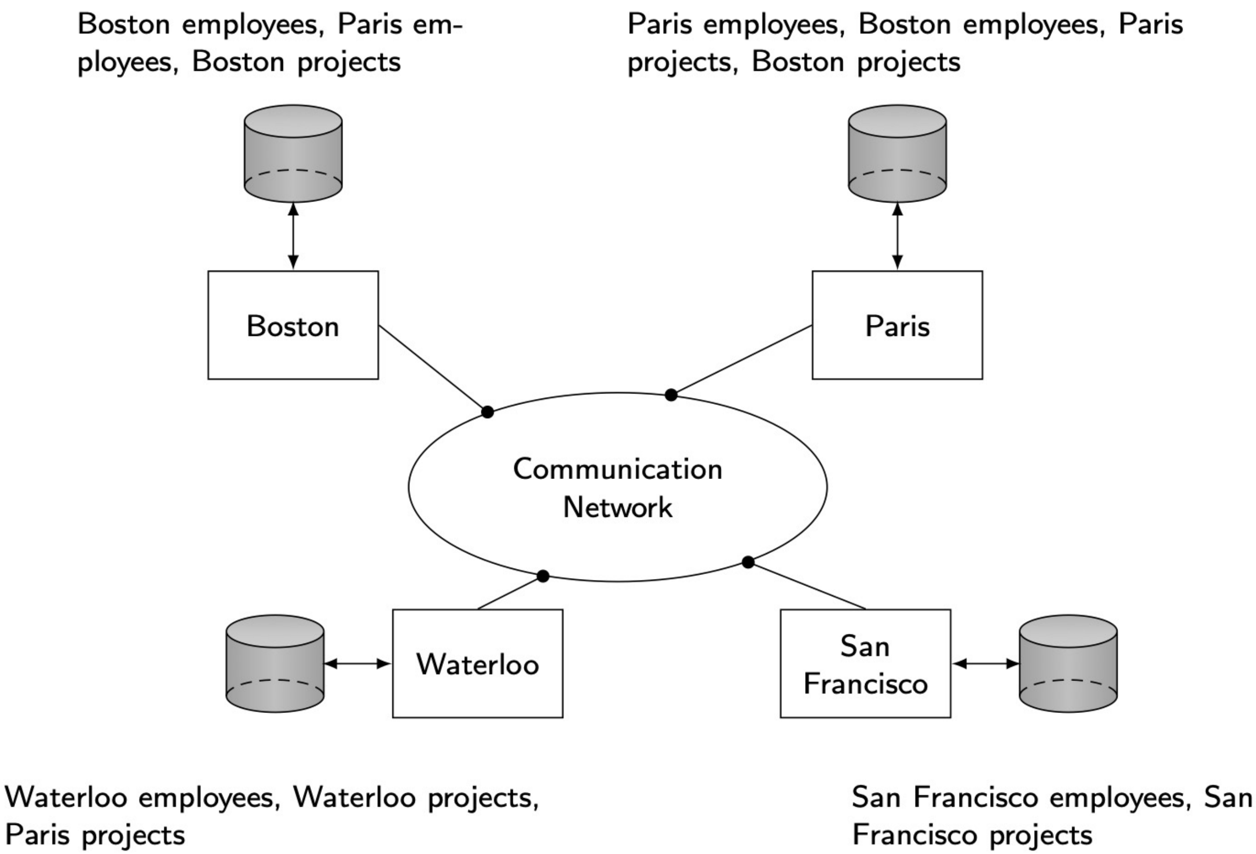
History – Database Management



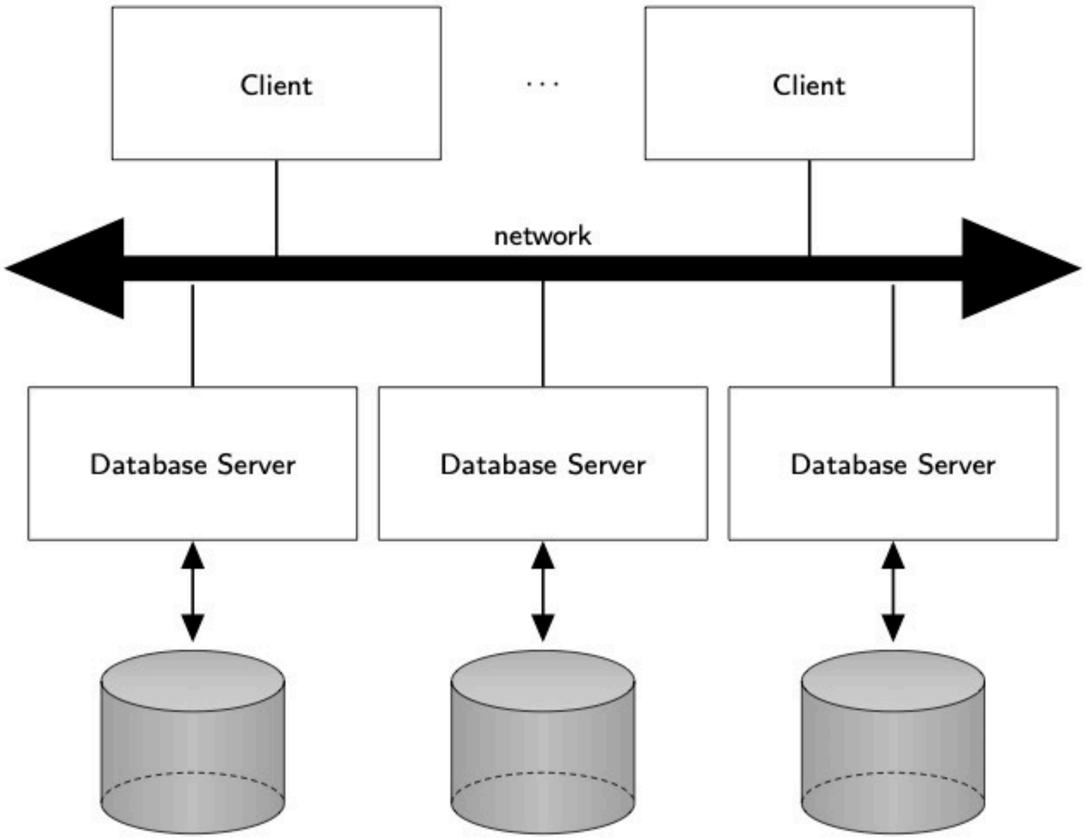
© 2020, M.T. Özsu & P. Valduriez

History – Early Distribution

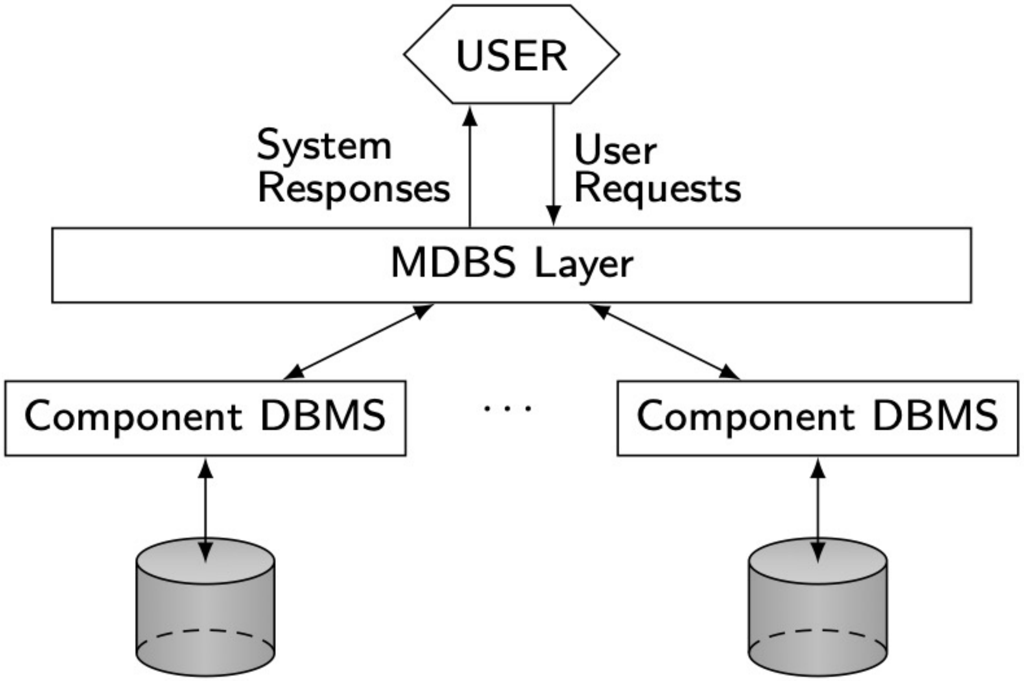
Peer-to-Peer (P2P)



History – Client/Server



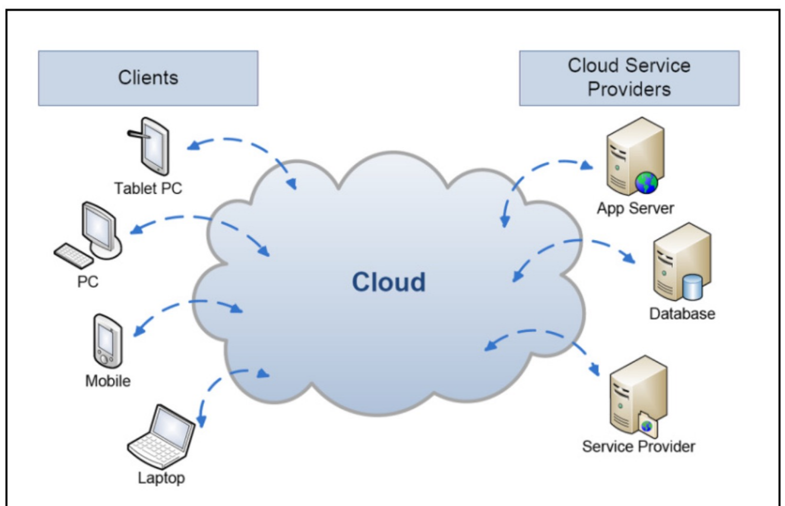
History – Data Integration



History – Cloud Computing

On-demand, reliable services provided over the Internet in a cost-efficient manner

- Cost savings: no need to maintain dedicated compute power
- Elasticity: better adaptivity to changing workload



© 2020, M.T. Özsu & P. Valduriez

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

Outline

- Introduction
  - What is a distributed DBMS
  - History
  - Distributed DBMS promises
  - Design issues
  - Distributed DBMS architecture

Distributed DBMS Promises

❶ Transparent management of distributed, fragmented, and replicated data

❷ Improved reliability/availability through distributed transactions

❸ Improved performance

❹ Easier and more economical system expansion

© 2020, M.T. Özsu & P. Valduriez

Transparency

- Transparency is the separation of the higher-level semantics of a system from the lower level implementation issues.
- Fundamental issue is to provide

  data independence

  in the distributed environment
  - Network (distribution) transparency
  - Replication transparency
  - Fragmentation transparency
    - horizontal fragmentation: selection
    - vertical fragmentation: projection
    - hybrid

© 2020, M.T. Özsu & P. Valduriez

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

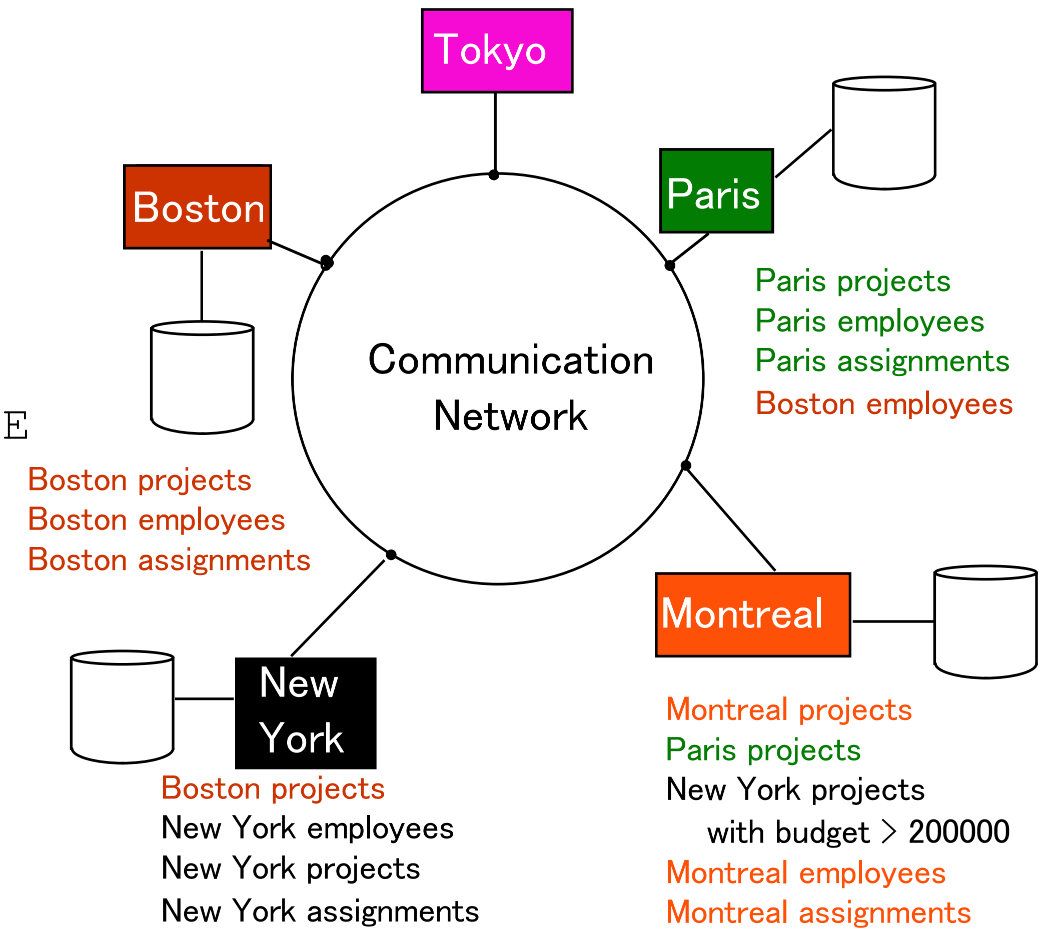
PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

PAY

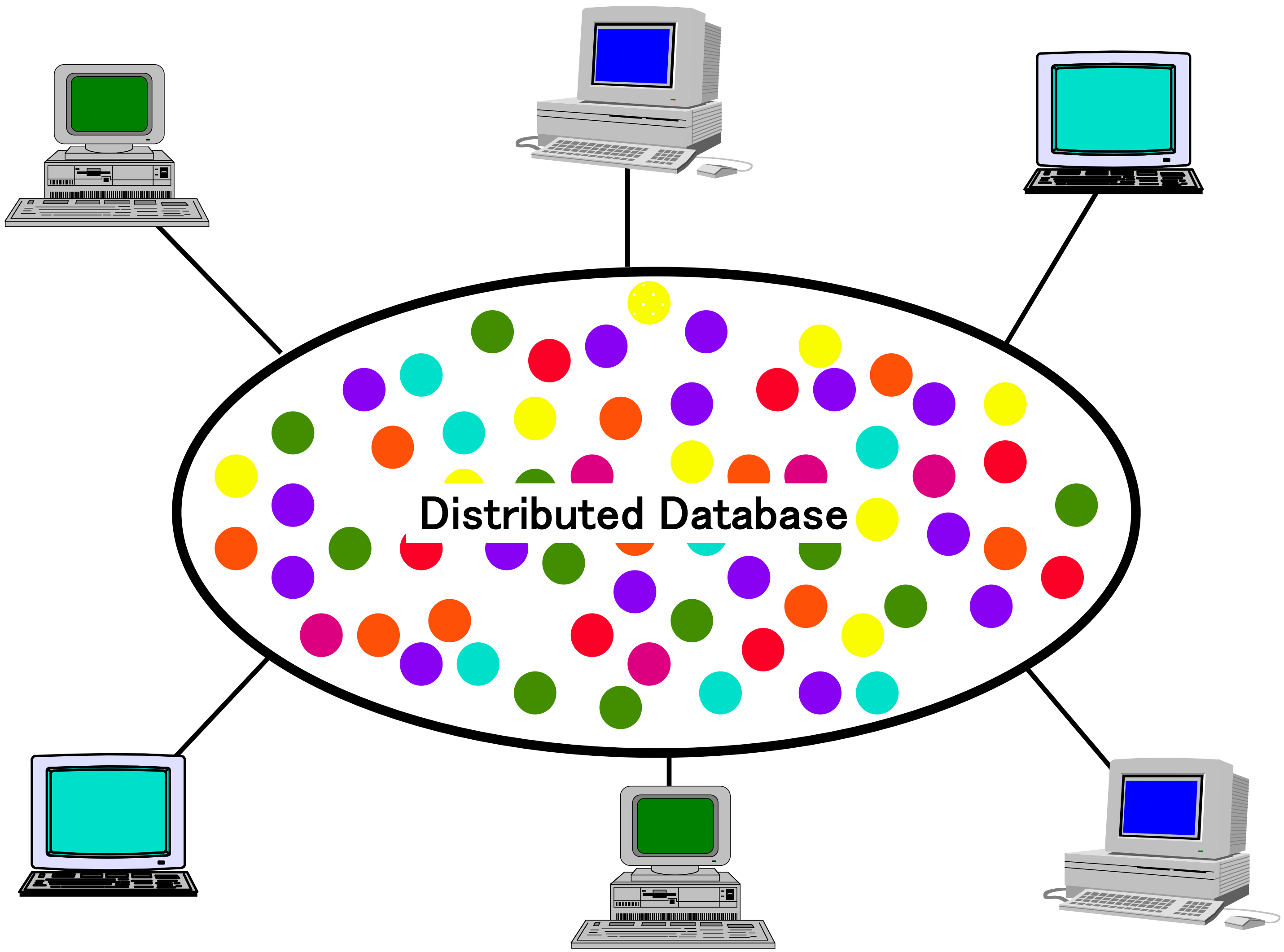
TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Transparent Access

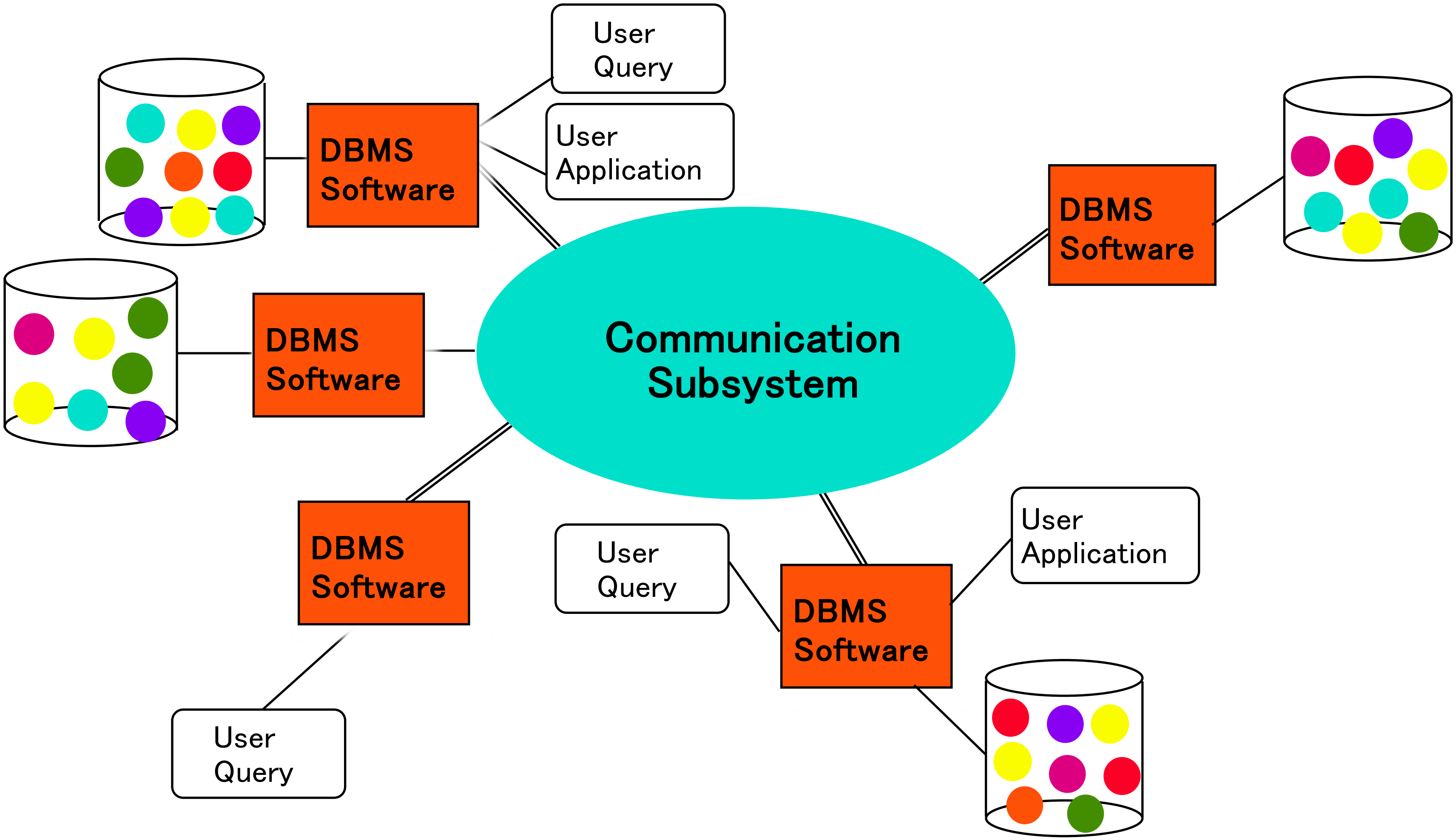
```
SELECT ENAME,SAL
FROM   EMP,ASG,PAY
WHERE  DUR > 12
AND    EMP.ENO = ASG.ENO
AND    PAY.TITLE = EMP.TITLE
```



Distributed Database - User View



Distributed DBMS - Reality



Types of Transparency

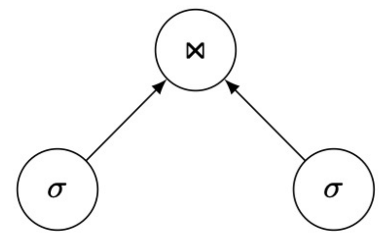
- Data independence
- Network transparency (or distribution transparency)
  - Location transparency
  - Fragmentation transparency
- Fragmentation transparency
- Replication transparency

Reliability Through Transactions

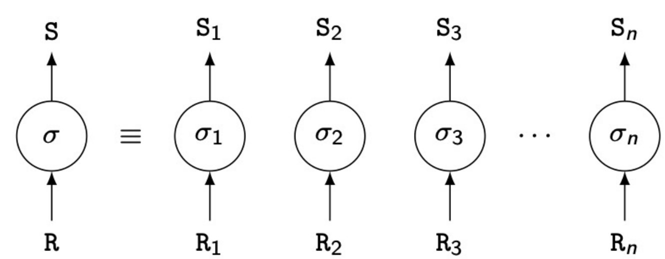
- Replicated components and data should make distributed DBMS more reliable.
- Distributed transactions provide
  - Concurrency transparency
  - Failure atomicity
- Distributed transaction support requires implementation of
  - Distributed concurrency control protocols
  - Commit protocols
- Data replication
  - Great for read-intensive workloads, problematic for updates
  - Replication protocols

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



Scalability

- Issue is database scaling and workload scaling
- Adding processing and storage power
- Scale-out: add more servers
  - Scale-up: increase the capacity of one server → has limits

Outline

- Introduction
  - What is a distributed DBMS
  - History
  - Distributed DBMS promises
  - Design issues
  - Distributed DBMS architecture

© 2020, M.T. Özsu & P. Valduriez

Distributed DBMS Issues

- Distributed database design
  - How to distribute the database
  - Replicated & non-replicated database distribution
  - A related problem in directory management
- Distributed 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

- Distributed concurrency control
  - Synchronization of concurrent accesses
  - Consistency and isolation of transactions' effects
  - Deadlock management
- Reliability
  - How to make the system resilient to failures
  - Atomicity and durability

© 2020, M.T. Özsu & P. Valduriez

Distributed DBMS Issues

- Replication
  - Mutual consistency
  - Freshness of copies
  - Eager vs lazy
  - Centralized vs distributed
- Parallel DBMS
  - Objectives: high scalability and performance
  - Not geo-distributed
  - Cluster computing

© 2020, M.T. Özsu & P. Valduriez

Related Issues

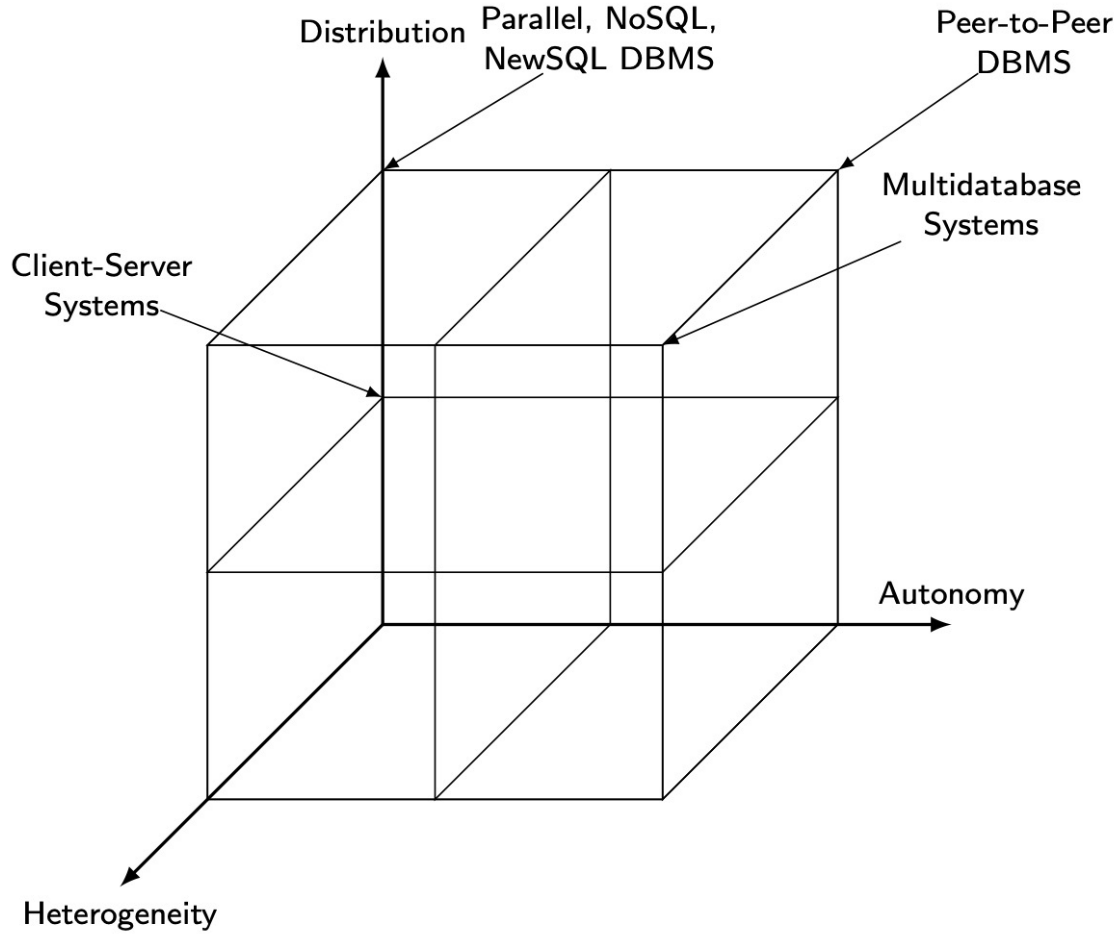
- Alternative distribution approaches
  - Modern P2P
  - World Wide Web (WWW or Web)
- Big data processing
  - 4V: volume, variety, velocity, veracity
  - MapReduce & Spark
  - Stream data
  - Graph analytics
  - NoSQL
  - NewSQL
  - Polystores

© 2020, M.T. Özsu & P. Valduriez

Outline

- Introduction
  - What is a distributed DBMS
  - History
  - Distributed DBMS promises
  - Design issues
  - Distributed DBMS architecture

DBMS Implementation Alternatives



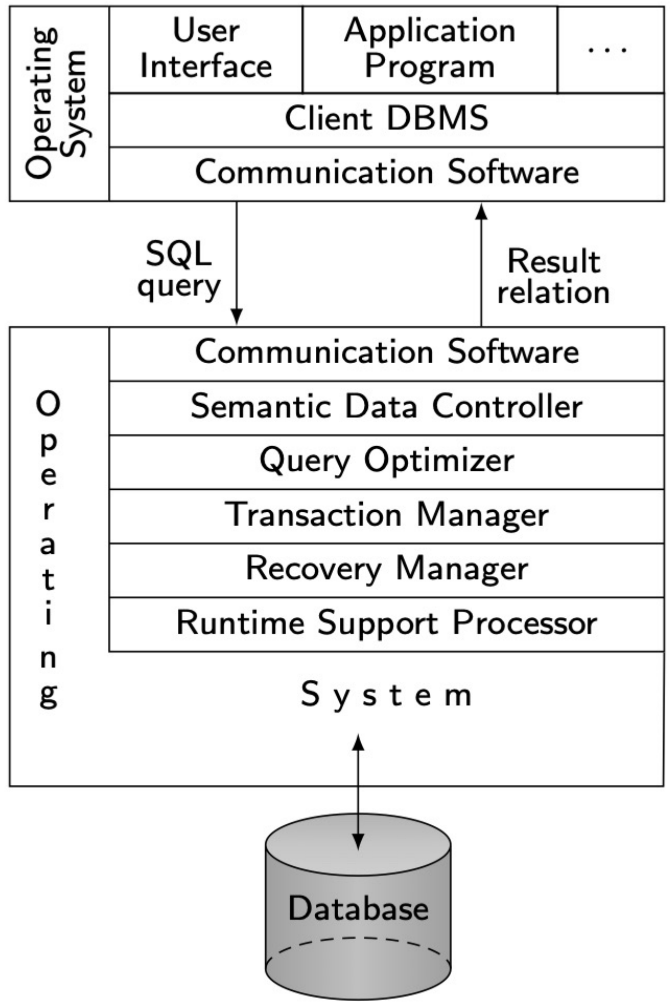
© 2020, M.T. Özsu & P. Valduriez

Dimensions of the Problem

- Distribution
  - Whether the components of the system are located on the same machine or not
- Heterogeneity
  - Various levels (hardware, communications, operating system)
  - DBMS important one
    - data model, query language,transaction management algorithms
- Autonomy
  - Not well understood and most troublesome
  - 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.
    - Execution autonomy: Ability of a component DBMS to execute local operations in any manner it wants to.

© 2020, M.T. Özsu & P. Valduriez

Client/Server Architecture

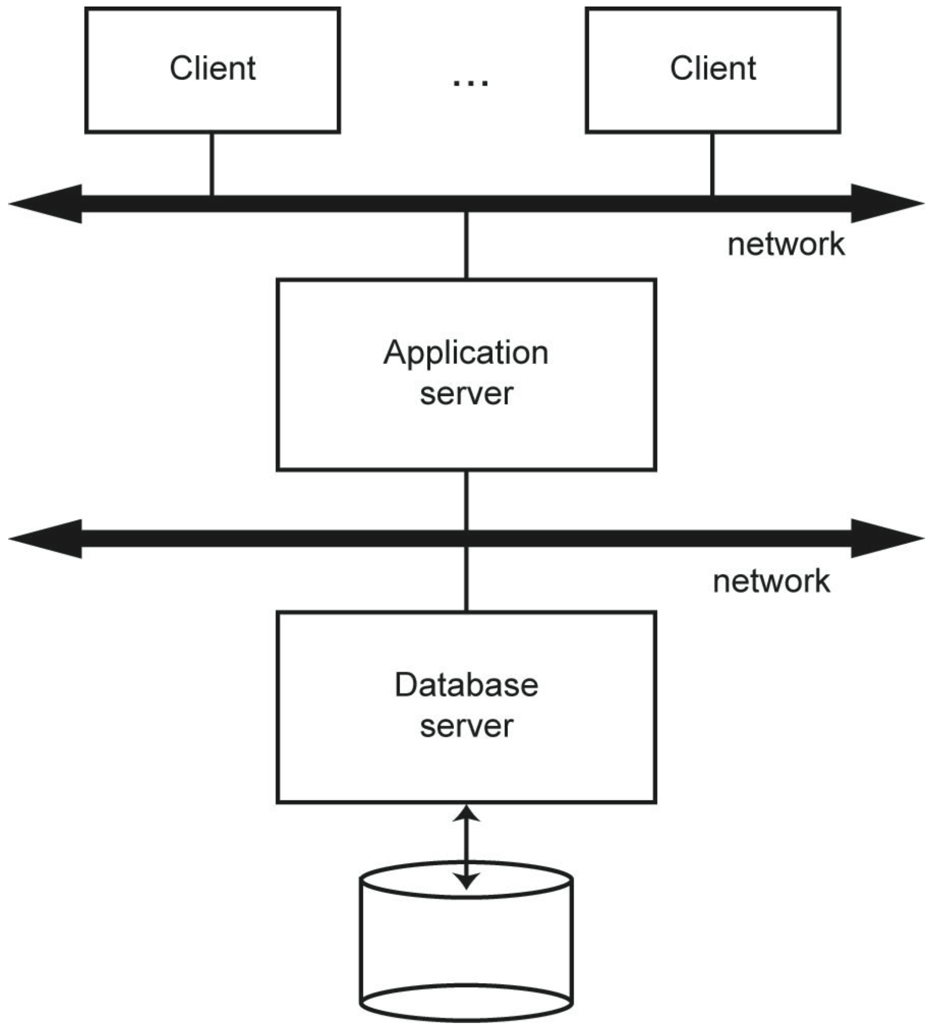


© 2020, M.T. Özsu & P. Valduriez

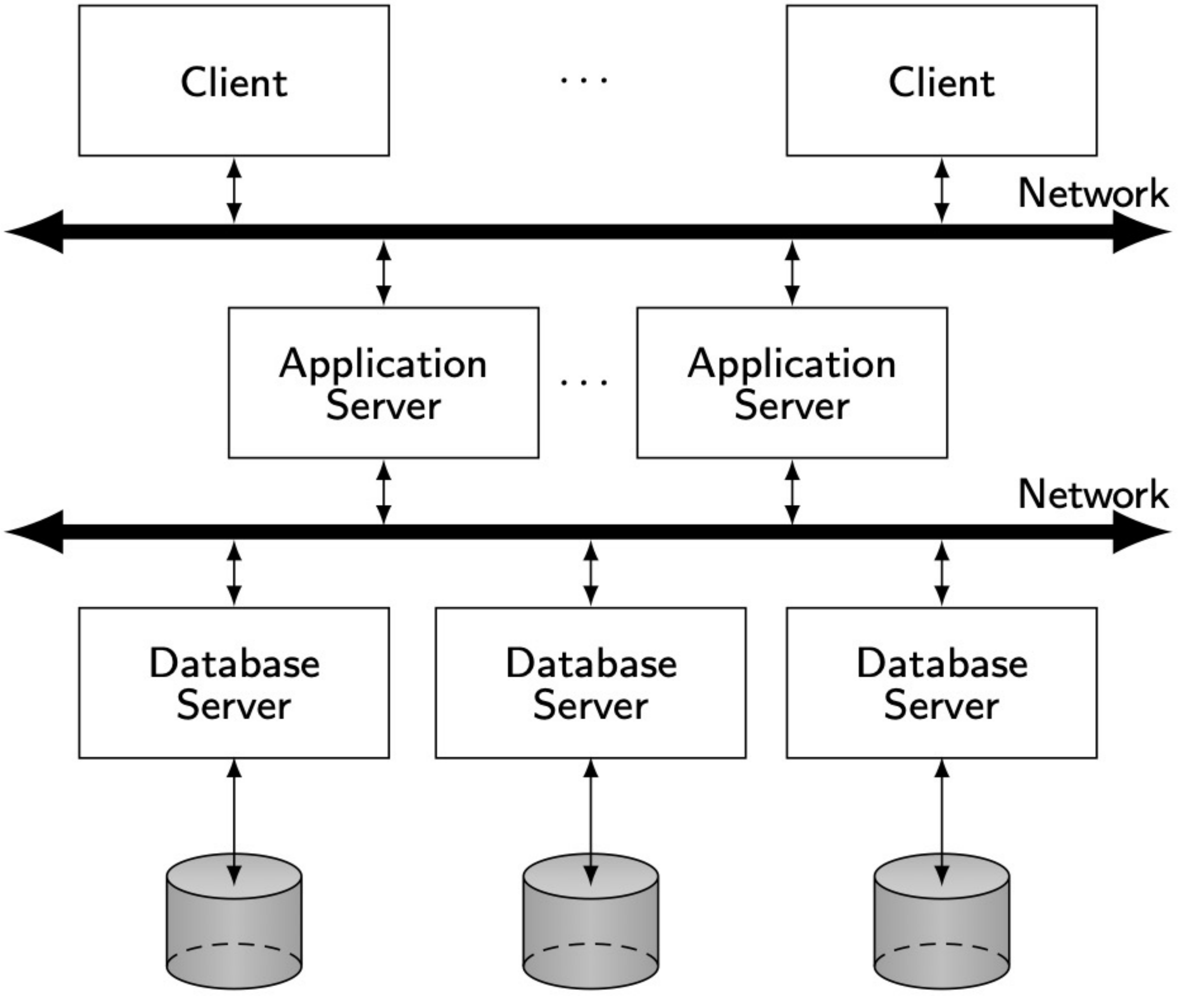
Advantages of Client-Server Architectures

- More efficient division of labor
- Horizontal and vertical scaling of resources
- Better price/performance on client machines
- Ability to use familiar tools on client machines
- Client access to remote data (via standards)
- Full DBMS functionality provided to client workstations
- Overall better system price/performance

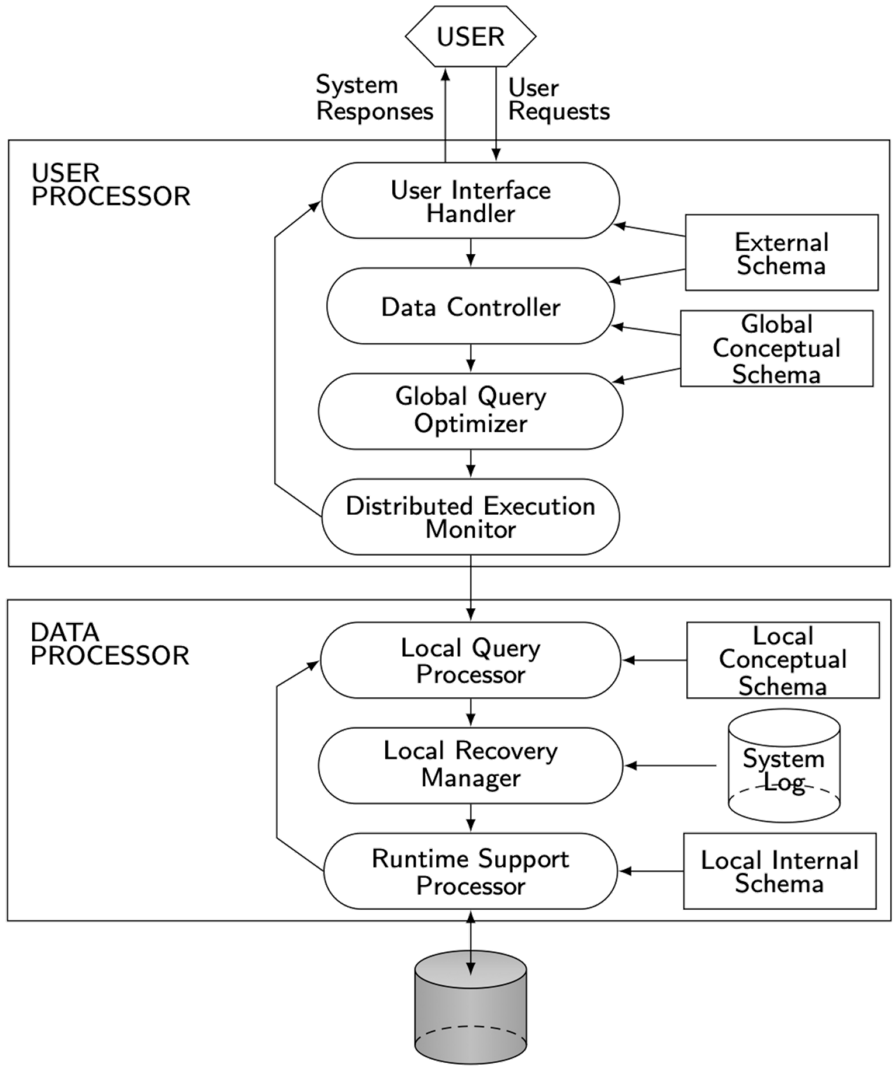
Database Server



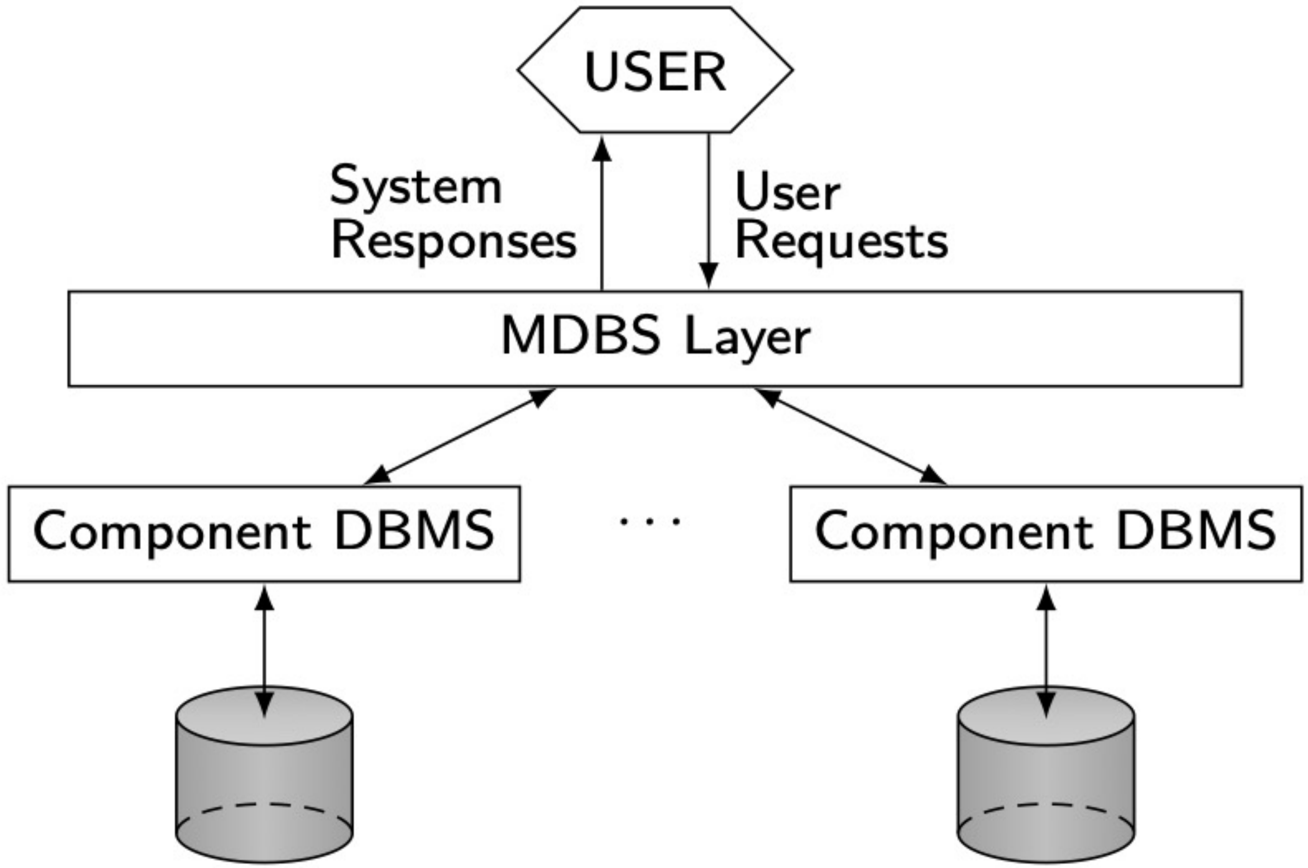
Distributed Database Servers



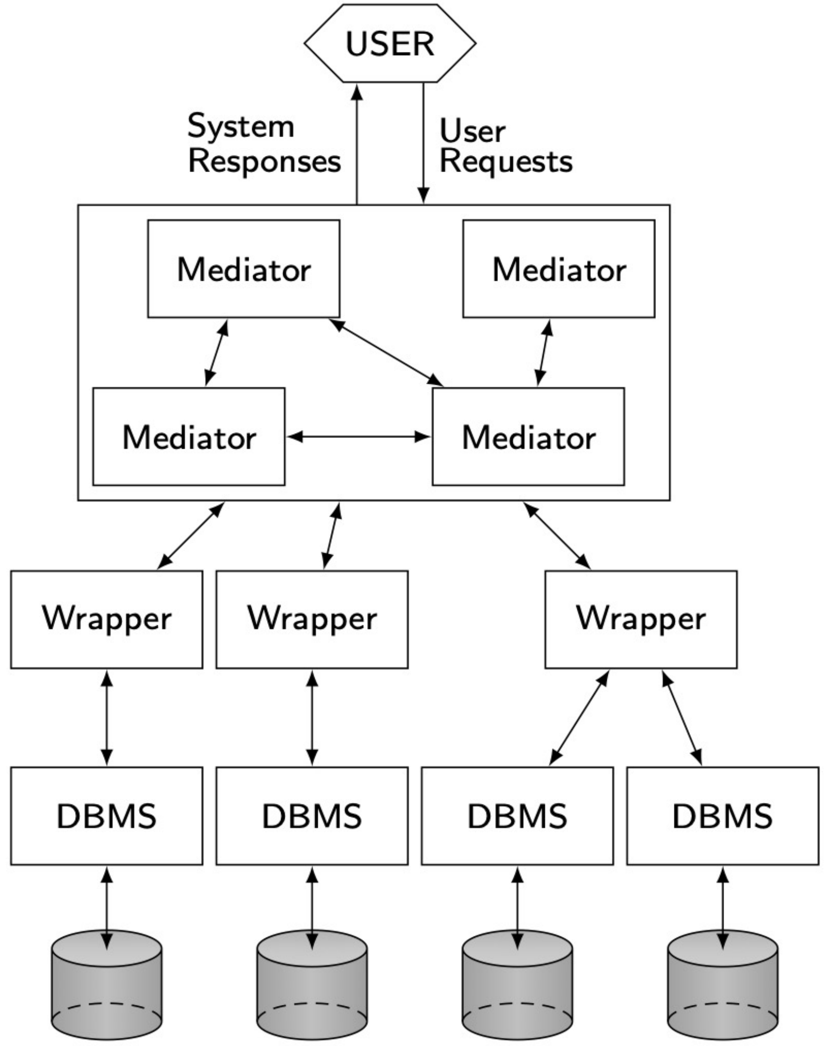
Peer-to-Peer Component Architecture



MDBS Components & Execution



Mediator/Wrapper Architecture



Cloud Computing

On-demand, reliable services provided over the Internet in a cost-efficient manner

- IaaS – Infrastructure-as-a-Service
- PaaS – Platform-as-a-Service
- SaaS – Software-as-a-Service
- DaaS – Database-as-a-Service

Simplified Cloud Architecture