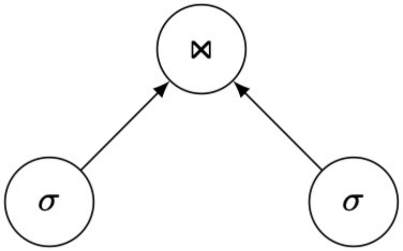
Reliability Through Transactions

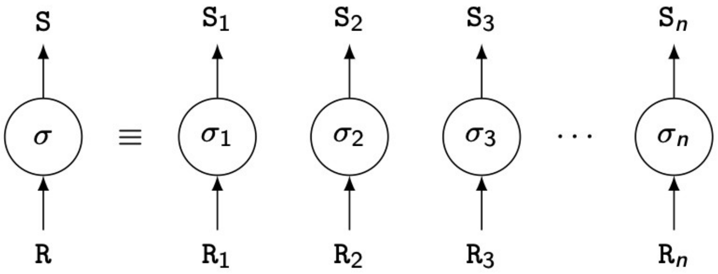
- Replicated components and data should make distributed DBMS more reliable.
- Distributed transactions provide
  - Concurrency transparency
  - Failure atomicity
- Distributed transaction support requires implementation of
  - Distributed concurrency control protocols
  - Commit protocols
- Data replication
  - 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

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

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

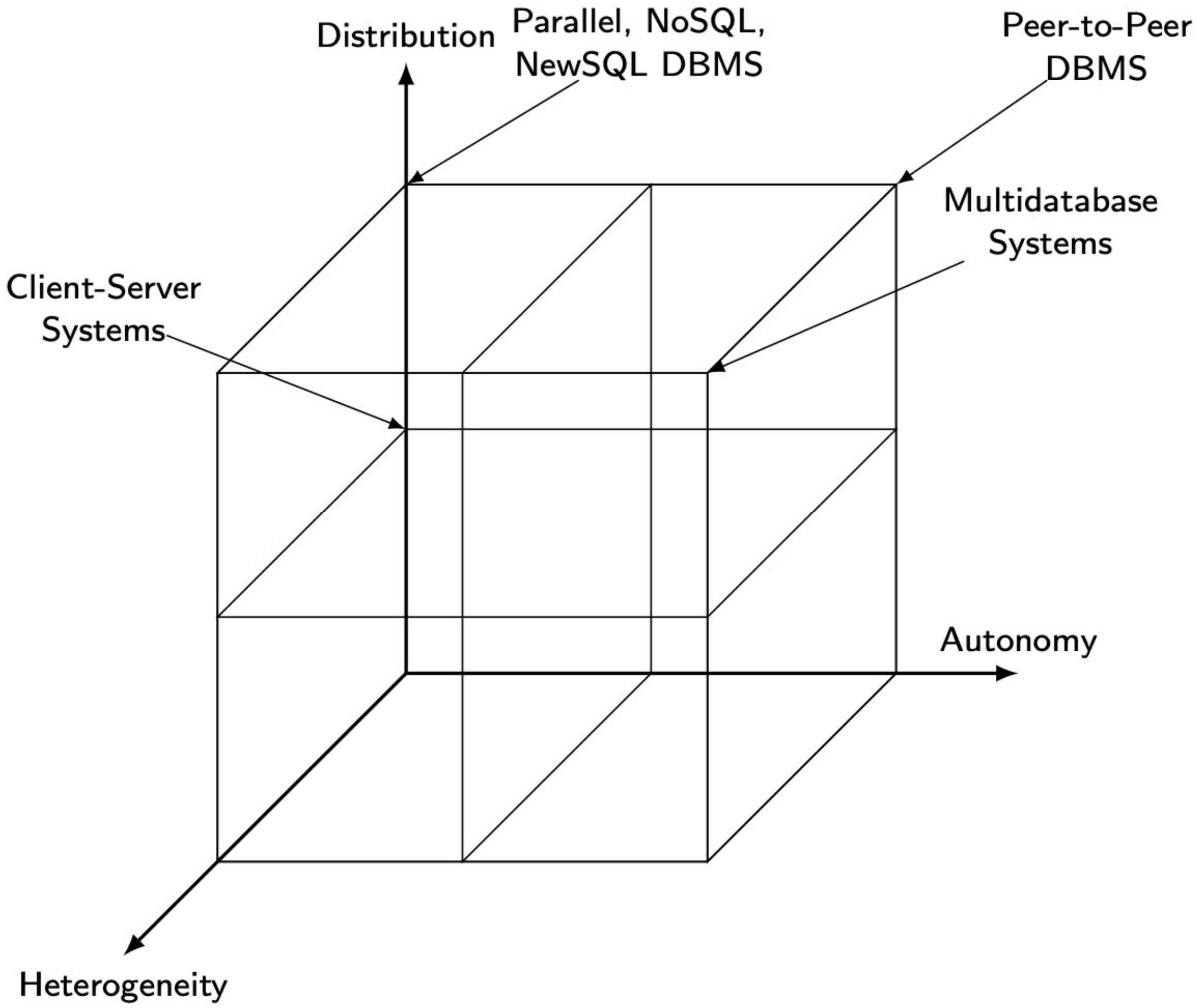
Related Issues

- 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

Outline

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

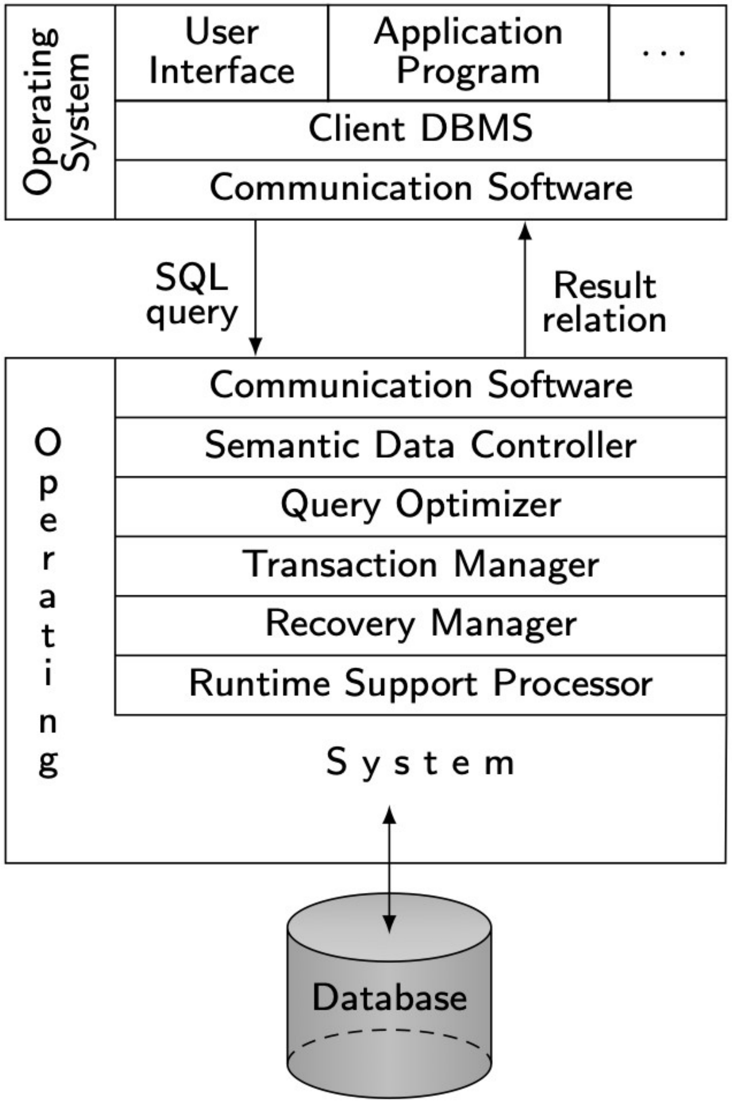
DBMS Implementation Alternatives



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.

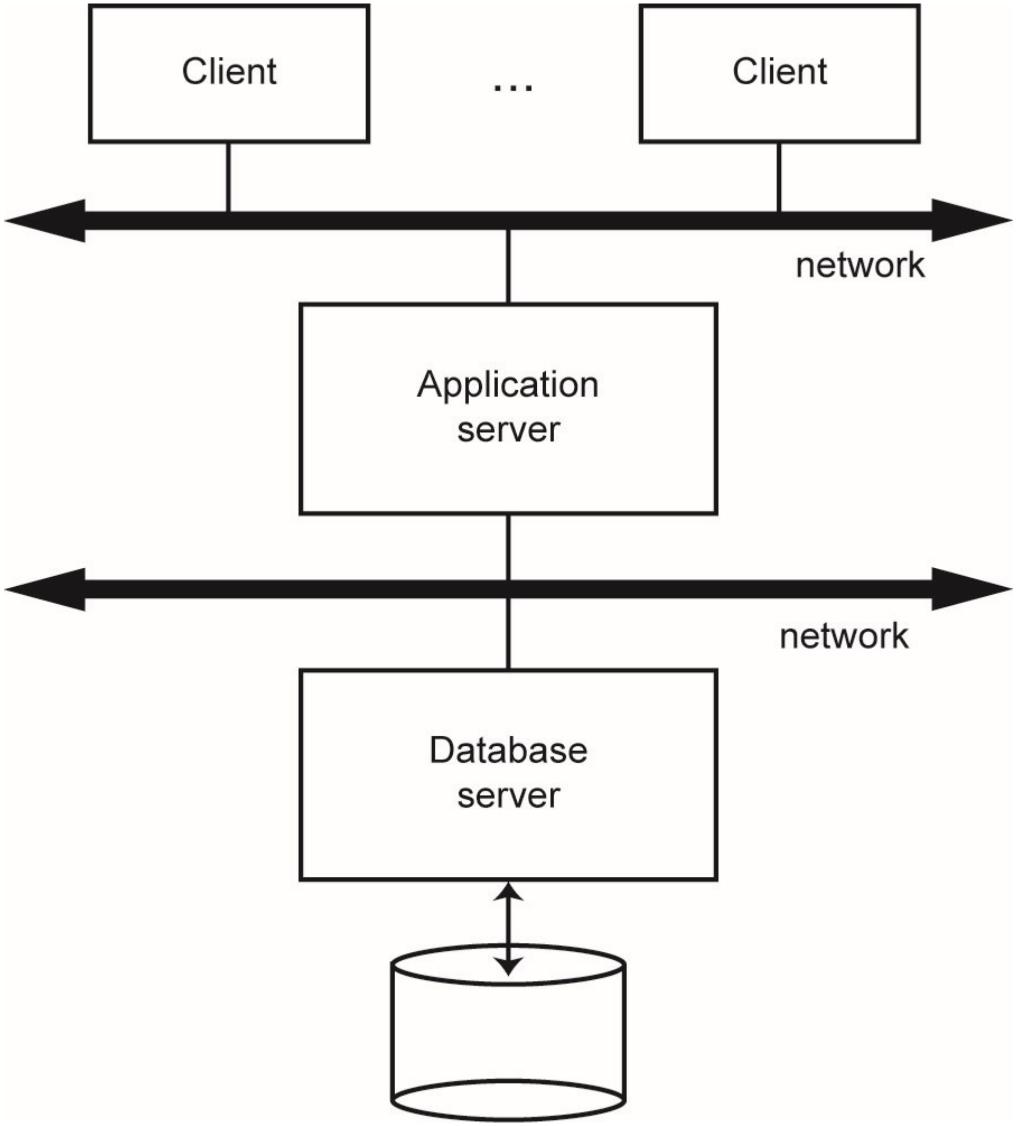
Client/Server Architecture



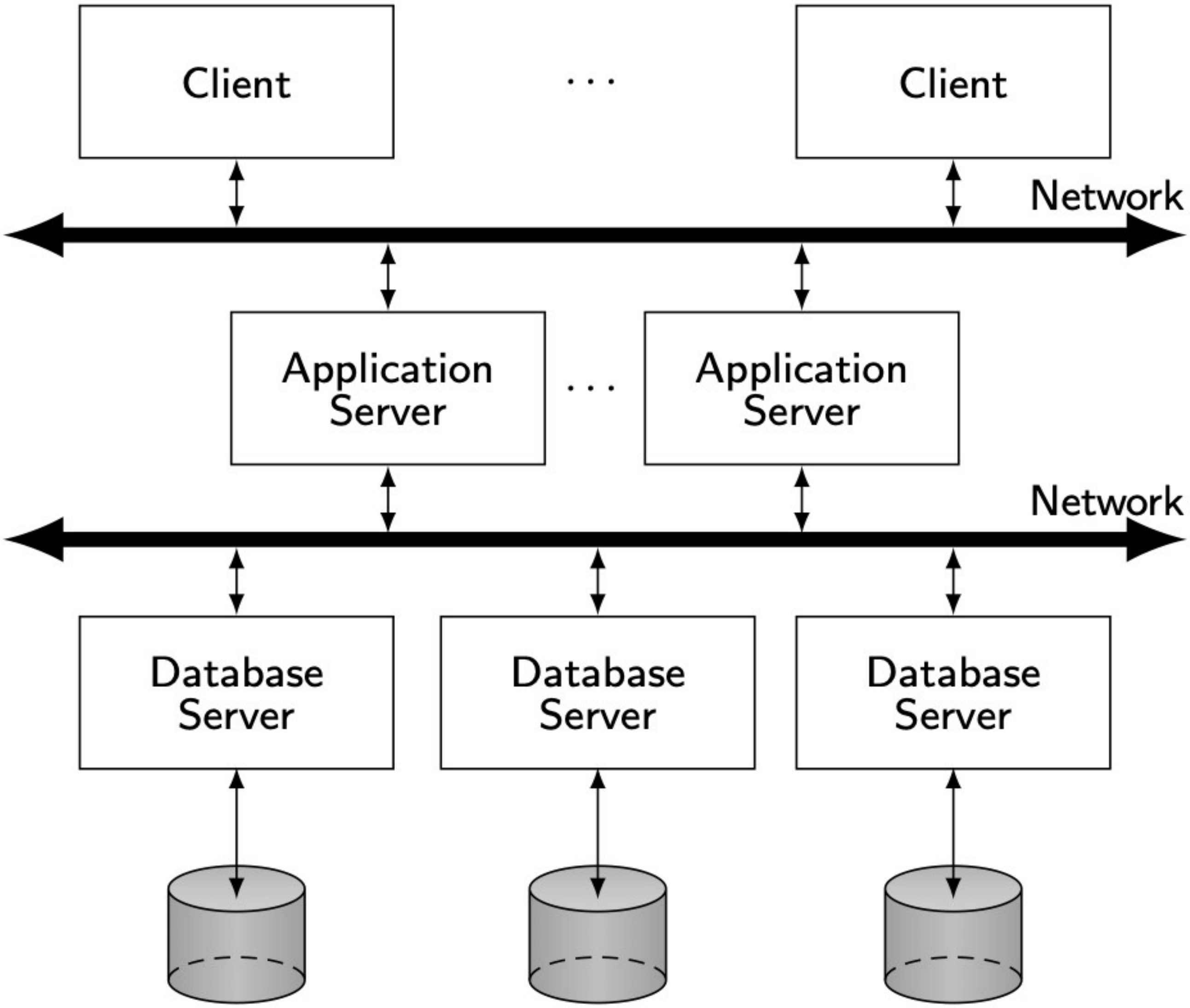
Advantages of Client-Server Architectures

- 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

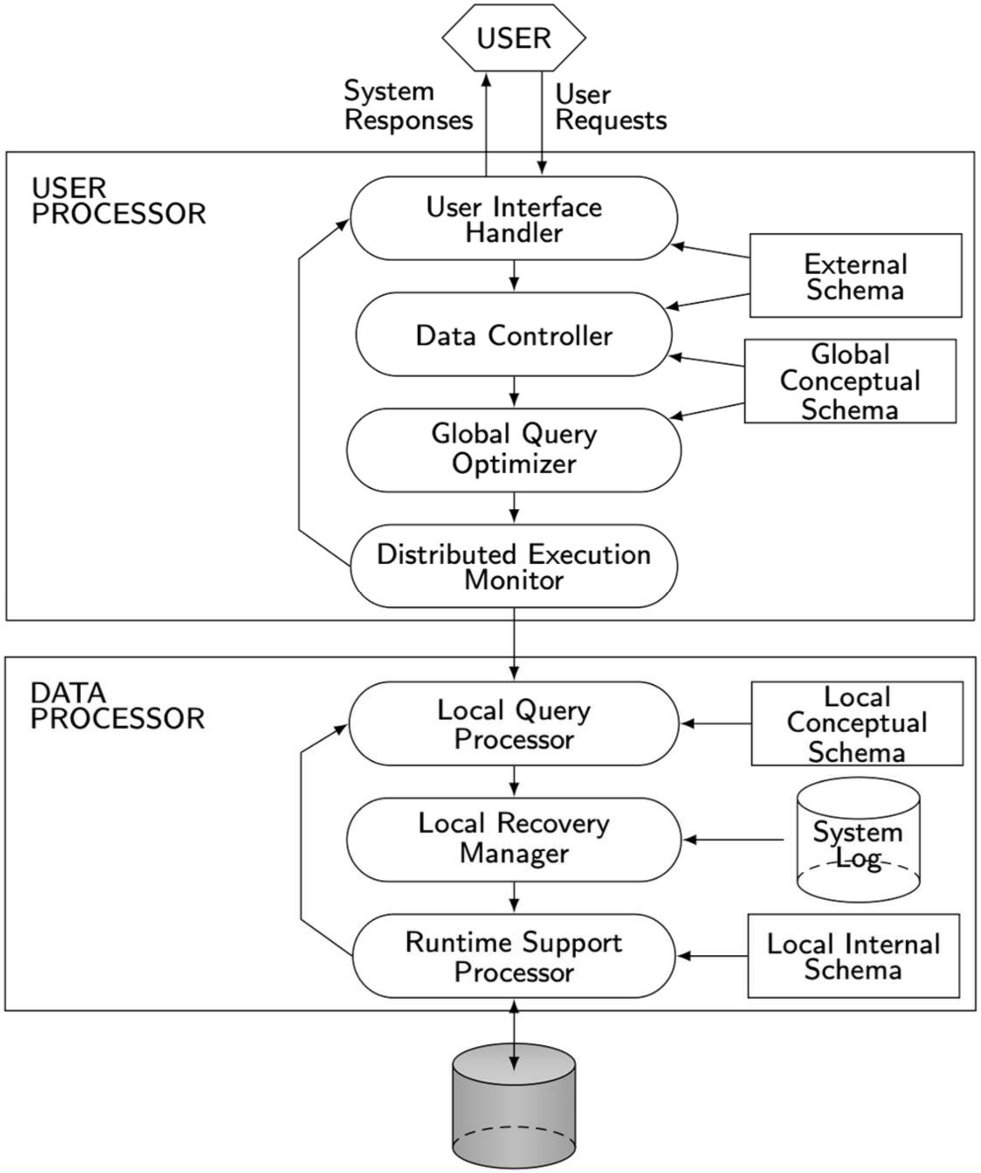
Database Server



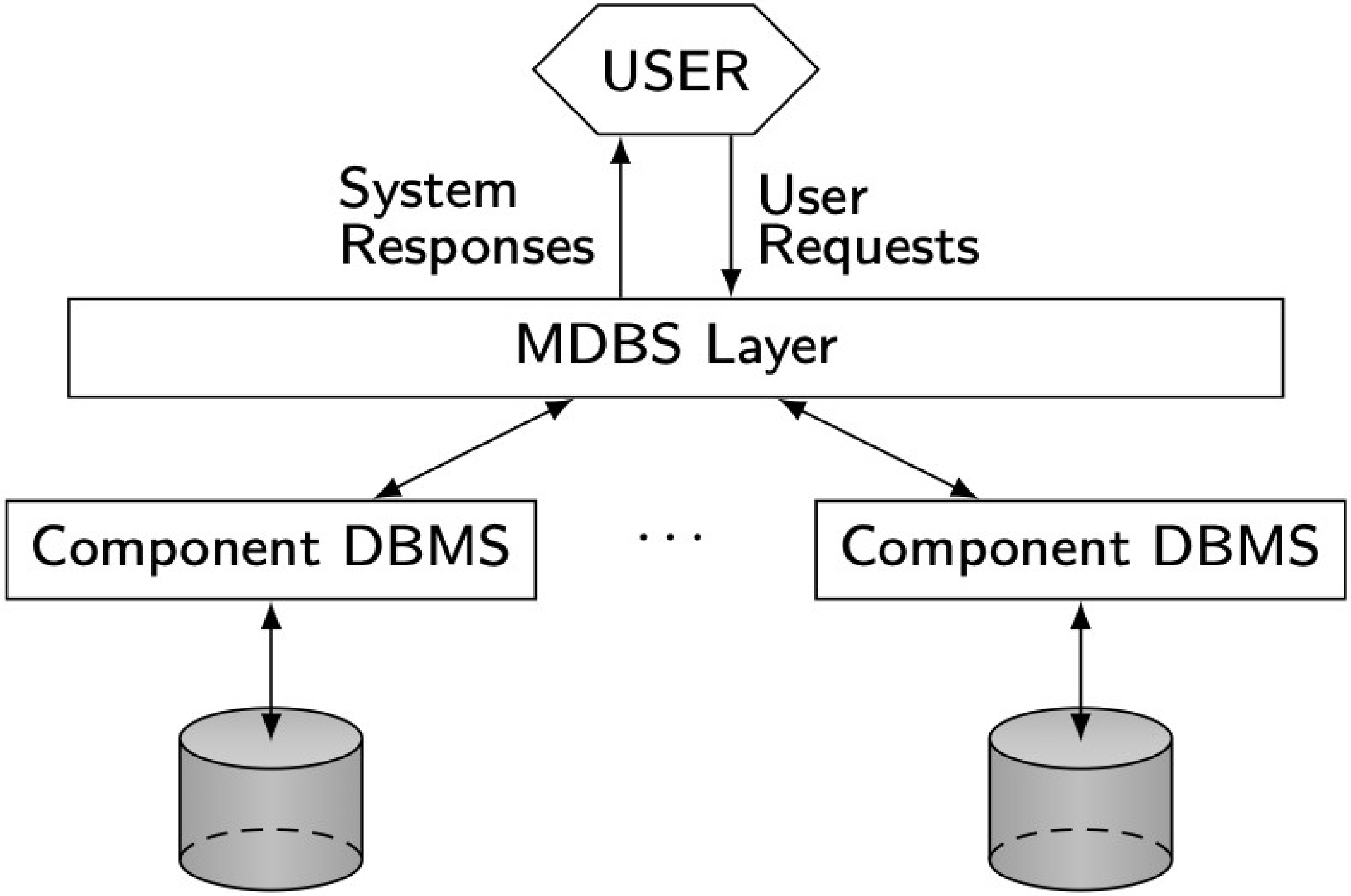
Distributed Database Servers



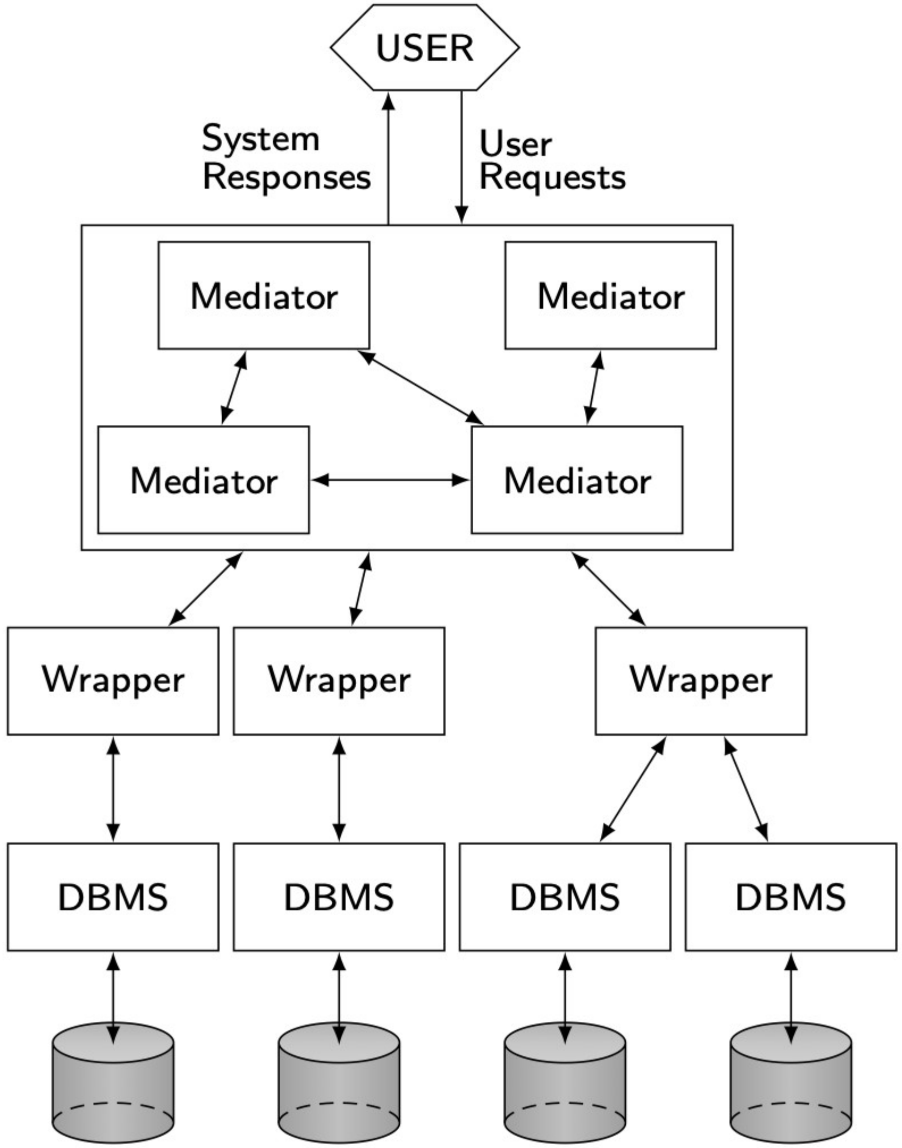
Peer-to-Peer Component Architecture



MDBS Components & Execution



Mediator/Wrapper Architecture

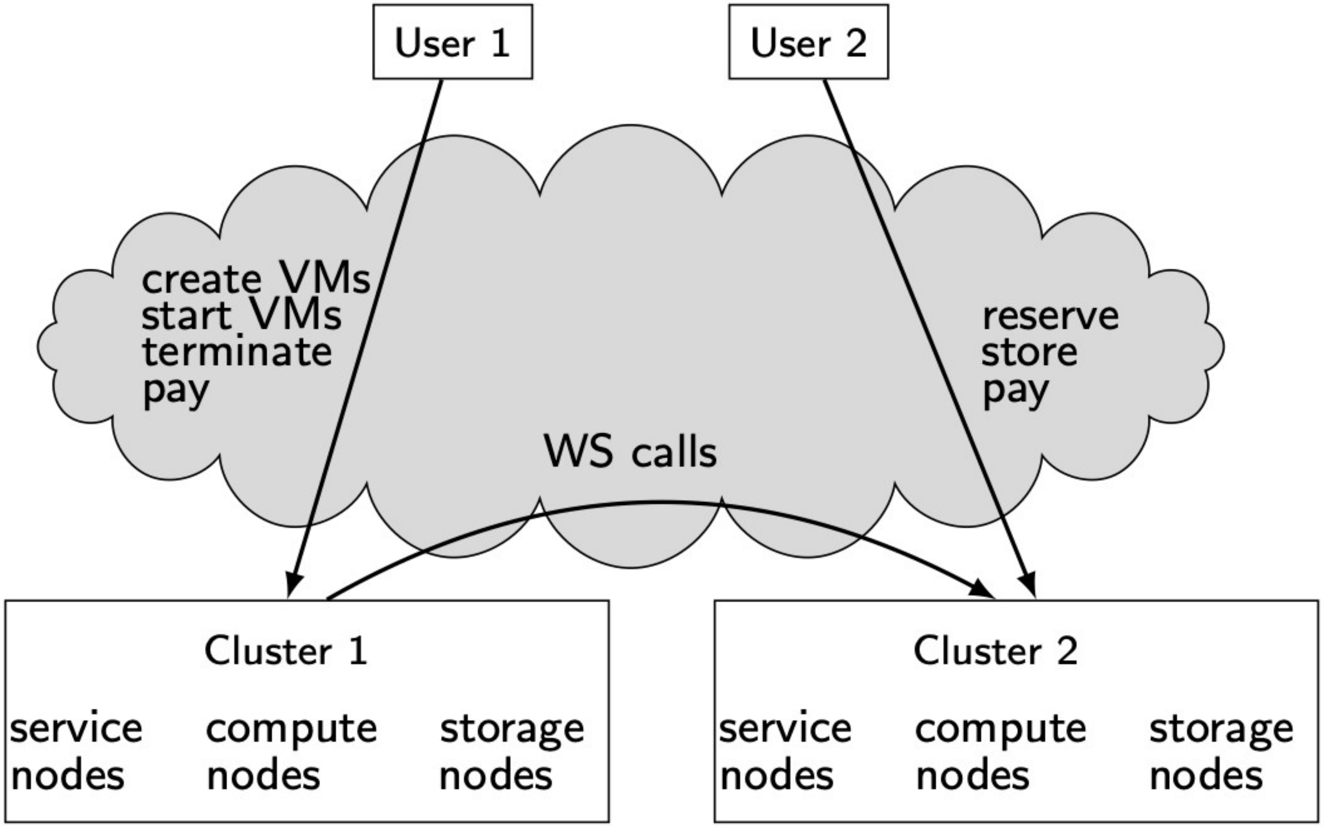


Cloud Computing

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


User 1
User 2
create VMs
start VMs
terminate
pay
reserve
store
pay
WS calls
Cluster 1
service nodes
compute nodes
storage nodes
Cluster 2
service nodes
compute nodes
storage nodes