Tier Architectures

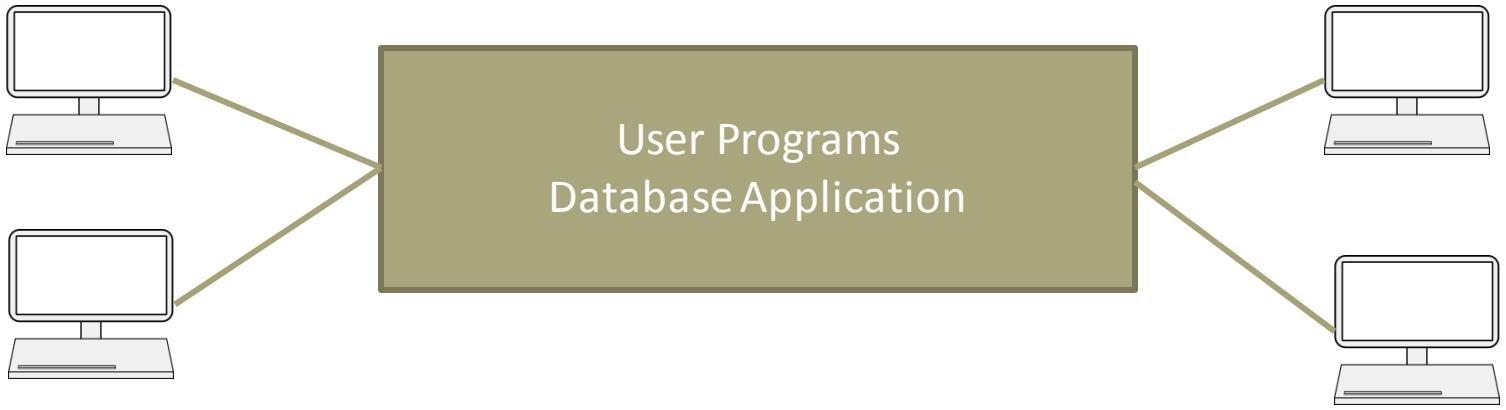
Kathleen Durant

CS 3200

Supporting Architectures for DBMS

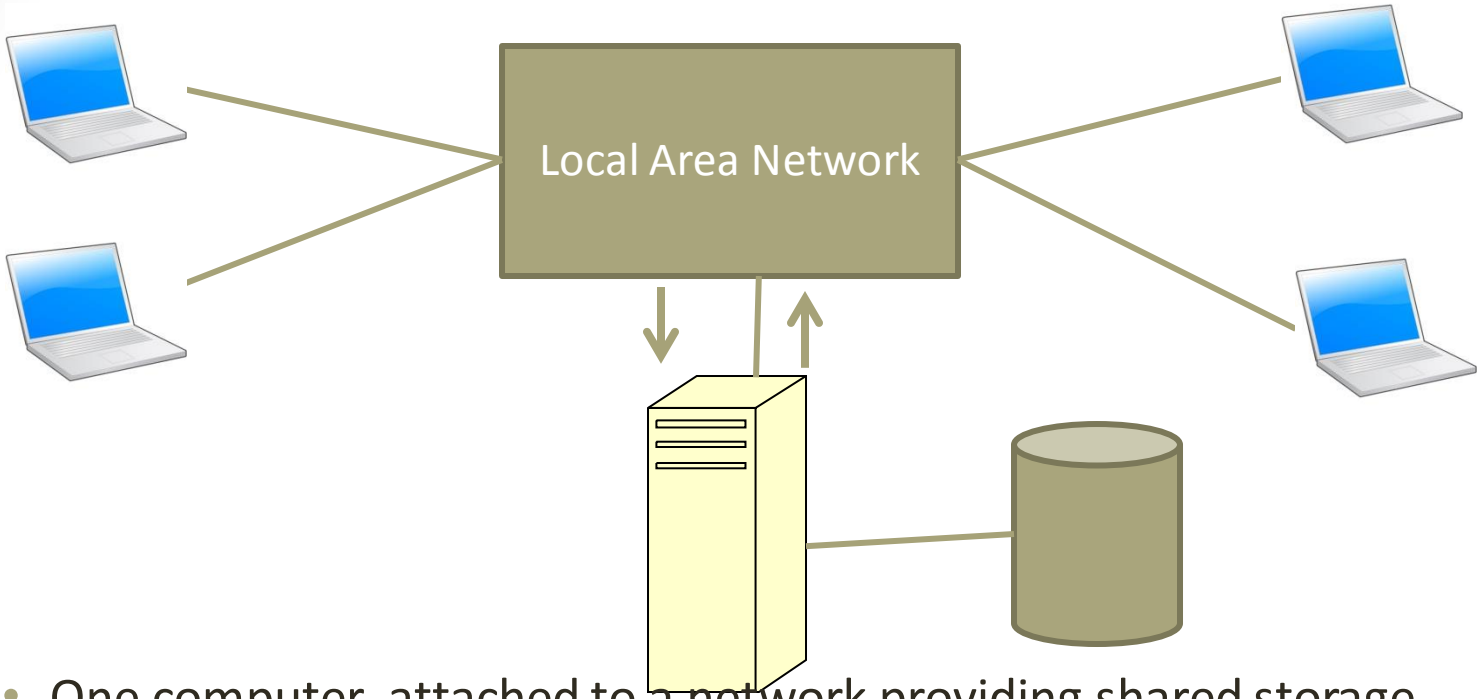
- Over the years there have been many different hardware configurations to support database systems
- Some are outdated others offer different functionality
- The requirements of the enterprise system will determine the appropriate architecture
- Given the increase in the users of database systems (due to the web) most current database solutions involve multiple computers each dedicated to achieve one specific type of task for the database system

Traditional Architecture



- One computer with 1 CPU
  - Multiple dumb terminals
- All processing done on the same physical computer
- Message passing via the operating system protocol
- Tremendous burden on the one computer

File-Server Architecture

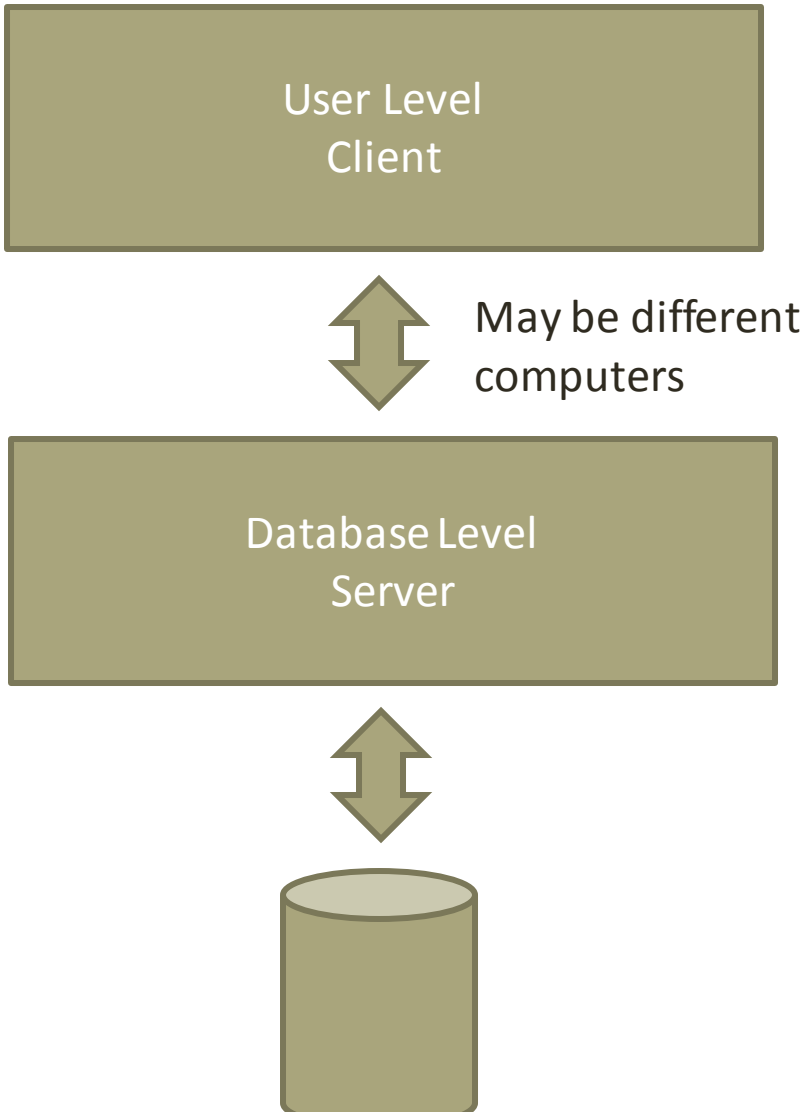


- One computer attached to a network providing shared storage
  - DBMS run on each workstation - data request from workstations
- File system is a shared hard disk drive
- Generates a lot of network traffic
- Tremendous burden on the network
- Concurrency, recovery, and integrity constraints more complex because multiple processes accessing the same file

Client Server Architecture

Requires a resource

Tier 1



Provides a resource

Tier 2

TASKS:
- User interface
- Check user input
- Processes application logic
- Generates DB requests
- Passes results to user

TASKS:
- Checks authorization
- Accepts and process database requests
- Sends data result to client
- Provides concurrent access and recovery

Benefits from Client Server Architecture

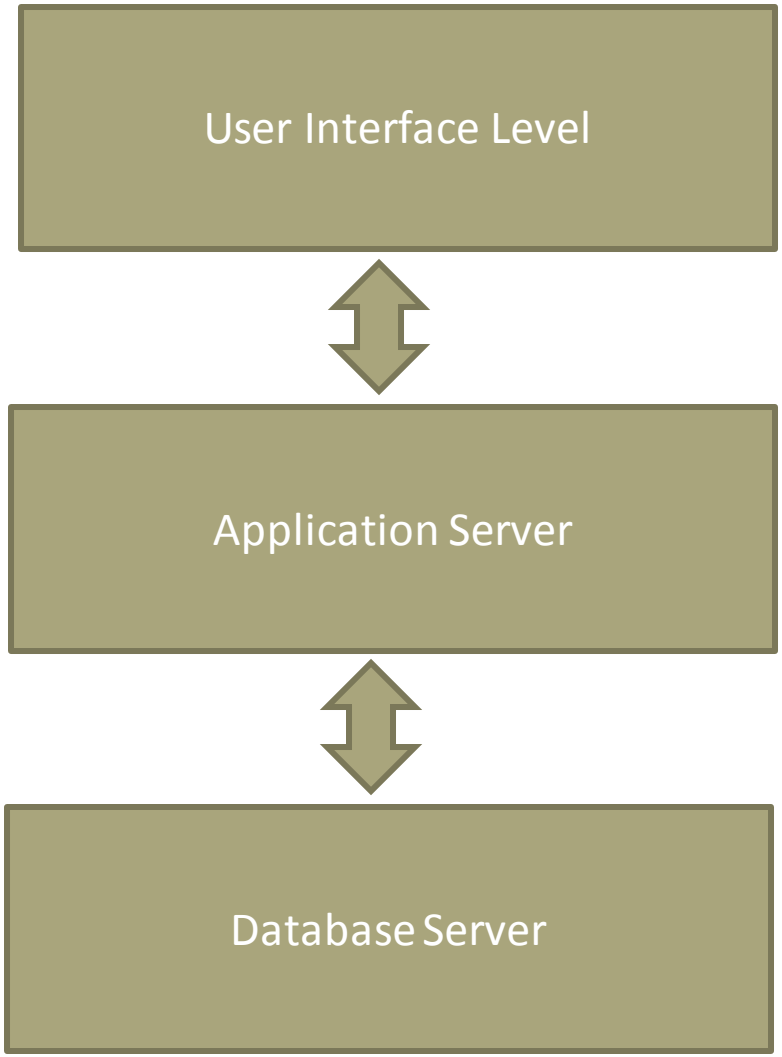
- Provides wider access to existing databases
- Potential increase in performance due to parallelism
  - If solution has separate client and server hardware , then can be processing applications in parallel
  - Dedicated server hardware provides the opportunity to tune the server machine for data processing
- Potential decrease in data storage cost
  - Since just the server needs to store and manage data
- Increased consistency given that the server is the only point of access to the data
- Maps to open systems architecture

3 Tier Architecture

FIRST TIER TASKS
User interface

SECOND TIER TASKS
Business Logic
Data processing logic

THIRD TIER TASKS
Data validation
Database access



3 Tiered Architecture

DESCRIPTION

- Client is only responsible for the application's user interface
  - Simple error checking on input data
  - Leads to a 'thin client'
- Core business logic of the application resides in its own layer
  - One application server is designed to serve multiple clients
- Addresses the problem of enterprise scalability

BENEFITS

- Less expensive hardware for the clients because client proccesses are thin
- Application maintenance is centralized
- Tier hardware configurations are independent from each other
- Load balancing is easier due to separation of the core business logic from the database functions

Application Server Provides

- Hosts an application programming interface to expose business logic and business processes for use by other applications
- Concurrency for the clients
- Network connection management
- Access to all the database servers
- Database connection pooling and management
- Legacy database support
- Clustering support
- Load balancing
- Failover
- Examples: Oracle Tuxedo Application Server, Java Platform Enterprise Edition, .NET Framework by Microsoft

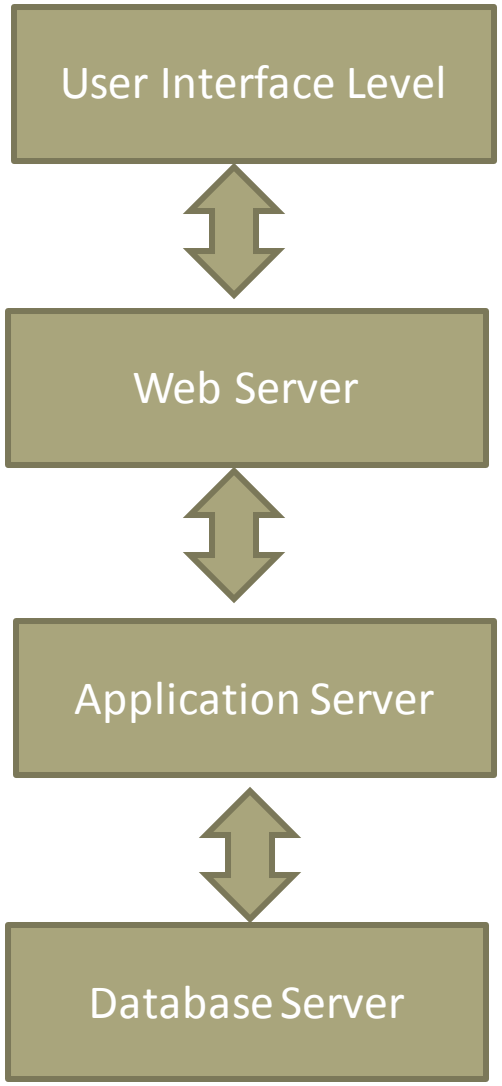
N-Tier Architecture

TIER 1
Client

TIER 2
Web Server

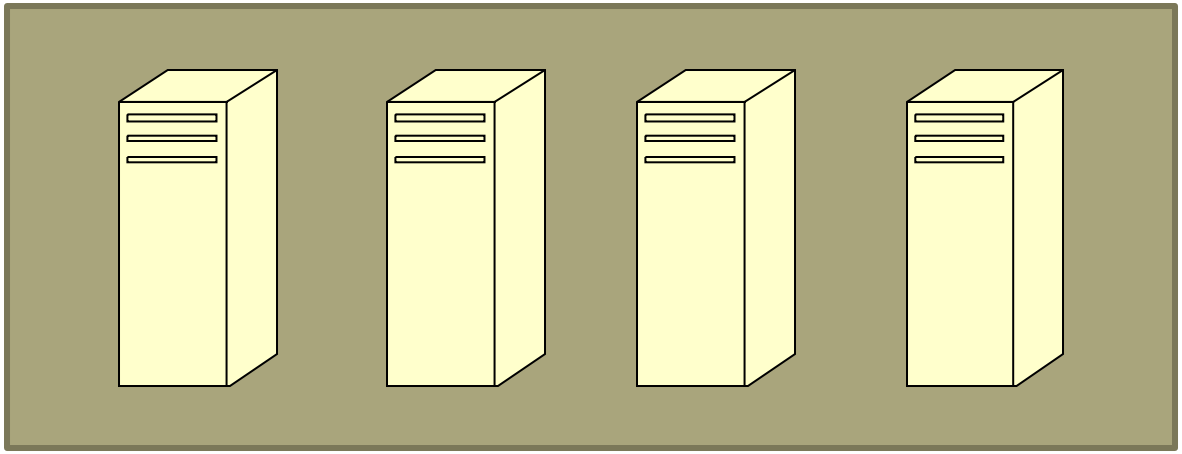
TIER 3
Application Server

TIER 4
Database Server



Appropriate for Large Enterprise Systems

Each tier can be a cluster of computers working together



- A cluster is a logical group of servers running server applications simultaneously and appearing as a single server to the world
- The servers may or may not have communication with their peers in the cluster
- You can dynamically add or remove servers to the cluster, depending upon the load
- Load balancers distribute customer requests among different servers in the cluster
- Scalability is an application's ability to support a growing number of users. It's a measure of a range of factors, including the number of simultaneous users a cluster can support and the time it takes to process a request
- High availability can be defined as redundancy. While handling requests, other servers in the cluster should be able to handle those requests as transparently as possible. A failed server is removed from the cluster as soon as it fails so that future requests are not routed to the failed server.

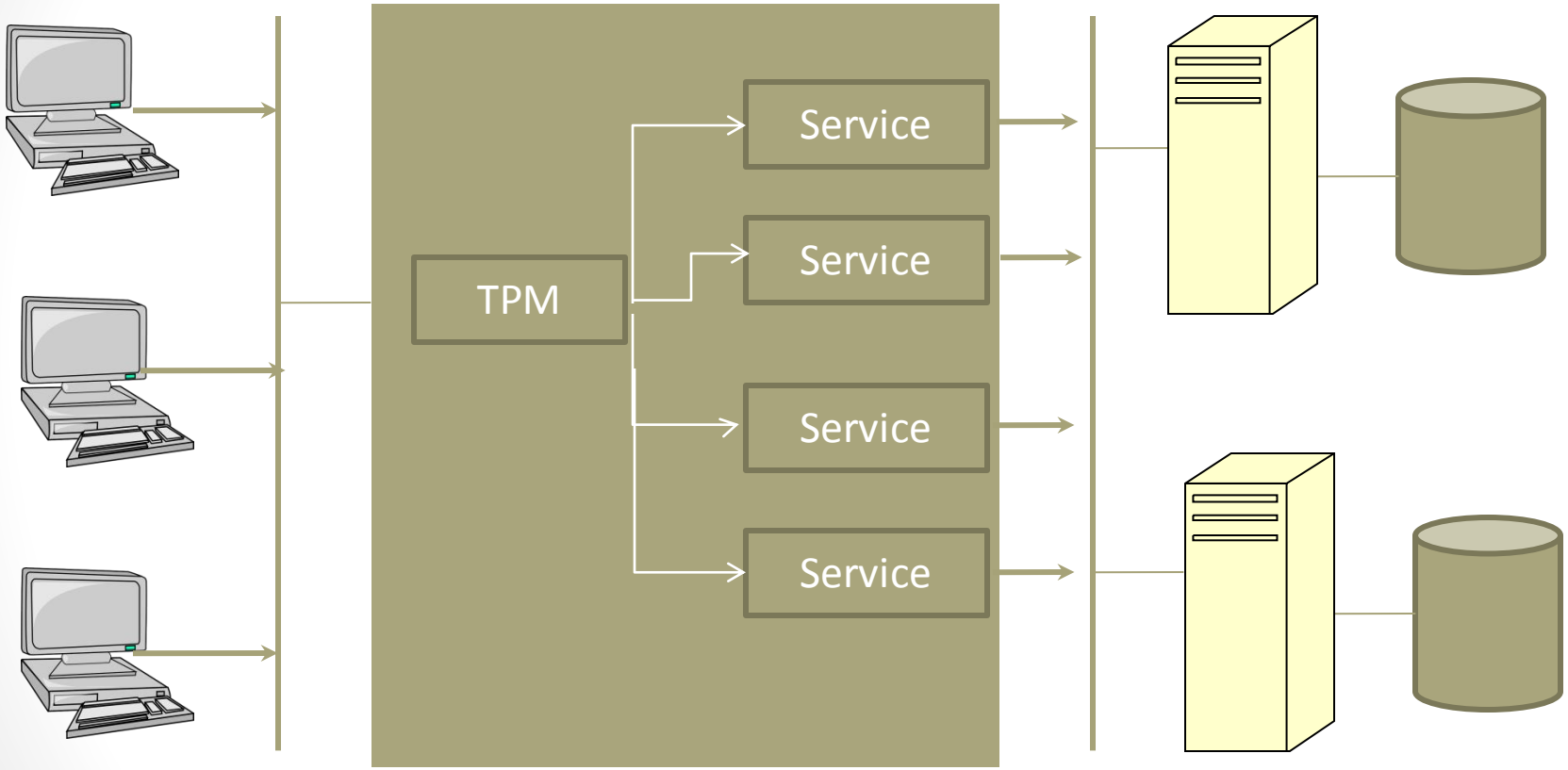
Middleware Software

- Software that mediates with other software
- Allows for communication between disparate applications in a heterogeneous system
- Provides a common interface and hides the complexities of distributed systems
- Necessary due to the number of computer systems and computer applications used to solve a specific problem
- Types of Middleware
  - Synchronous RPC - client blocked while server processes a call
  - Publish/subscribe – asynchronous messaging protocol – subscribers choose the message classes that want information on
  - Message-oriented middleware – resides on both the client and the server , supports asynchronous calls between them
  - Object-request broker manages data exchange between objects
  - **SQL-oriented data access – connects applications to any type of DB across a network, eliminates the need to write code for each specific type of DB**

Transaction Processing Monitor

- TPM is a middleware component that provides a uniform interface to applications developing transactional software
- Used in environments with extremely high volume
- Provides access to the services of a number of resource managers
- Transaction routing - direct specific queries to specific DBMS
- Manages **distributed transactions** on potentially heterogeneous hardware
- Load balancing – direct client requests to specific DBMS that are currently underutilized
- Increased reliability – acts as a transaction manager and maintains the consistency of the database

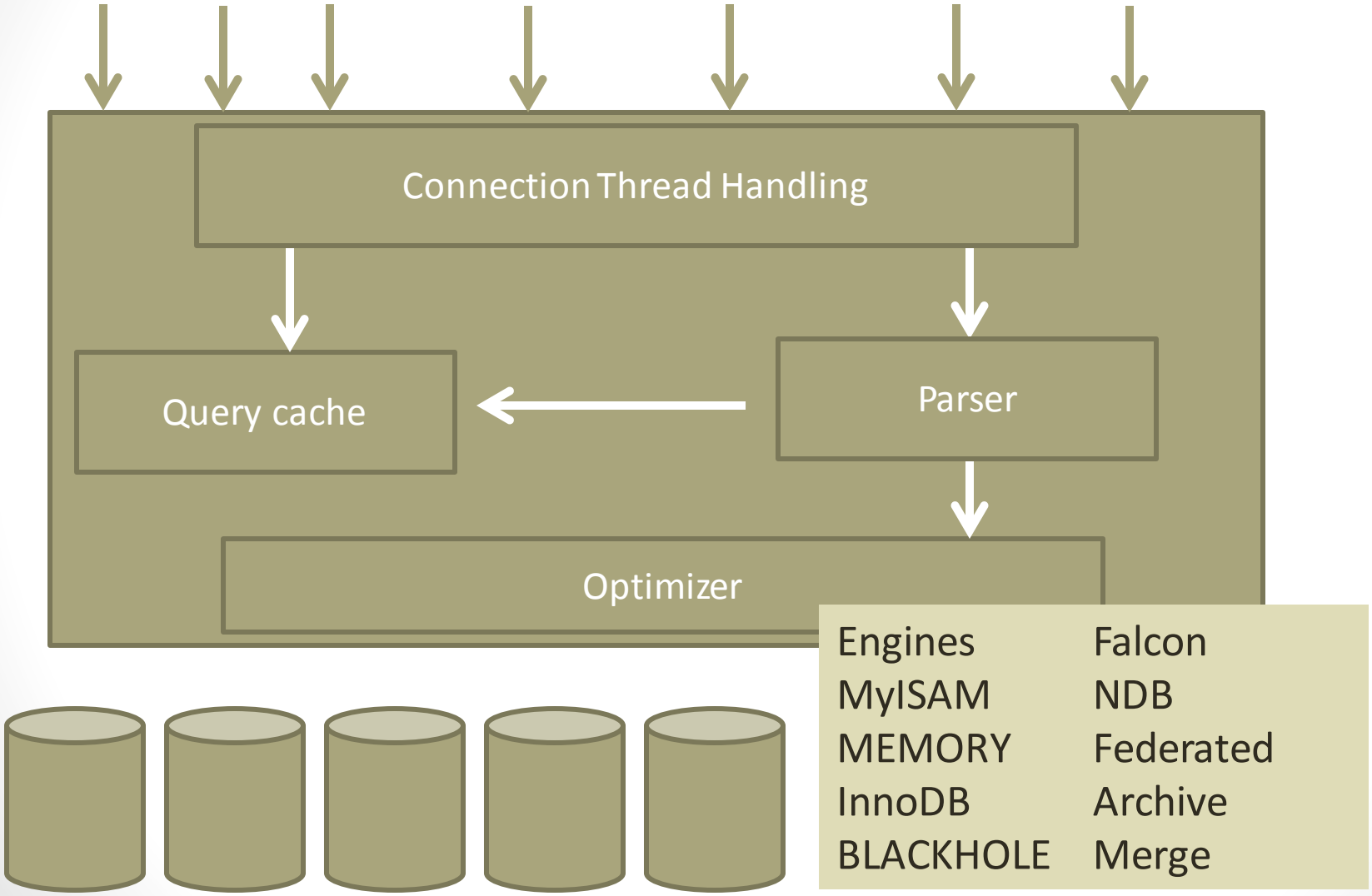
Transaction Processing Monitor in Second Tier



Tiers versus Layers

- *Tier* is a physical structuring mechanism for the system infrastructure
- *Layer* is a logical structuring mechanism for the elements that make up the software solution

MySQL Server Architecture



Query Engine

- Separates query processing and other server tasks from data storage and retrieval
- Plugins at run time
- My SQL lets you choose, on a per-table basis, how your data is stored and what performance, features, and other characteristics you want
- Default query engine is InnoDB

My SQL Engines

- InnoDB - general-purpose storage engine that balances high reliability and high performance, follows ACID , row level locking
- MyISAM – ISAM Btree storage, supports indexes
- Memory - creates special-purpose tables with contents that are stored in memory.
- CSV - stores data in text files using comma-separated values format.
- ARCHIVE - stores large amounts of data without indexes in a very small footprint
- BLACKHOLE - accepts data but throws it away and does not store it
- MERGE – a collection of identical ISAM tables that can be used as one table
- FEDERATED - access data from a remote MySQL database without using replication or cluster technology
- EXAMPLE - stub engine that does nothing
- NDBCluster - high-availability, high-redundancy version of MySQL adapted for the distributed computing environment
  - Types of nodes: Management, data, and SQL nodes
- https://dev.mysql.com/doc/refman/5.6/en/storage-engines.html

Data request flow

- Each client connection gets its own thread within the server process.
- The connection's queries execute within that single thread, which in turn resides on one core or CPU
- The server caches threads, so they don't need to be created and destroyed for each new connection
- MySQL parses queries to create an internal structure (the parse tree), and then applies a variety of optimizations
  - the storage engine does affect how the server optimizes the query
  - The optimizer asks the storage engine about some of its capabilities and the cost of certain operations, and for statistics on the table data
- the server consults the query cache for result sets – if found return the result from the cache as oppose to accessing disk
- Server sends the result set back to the client

MySQL Cluster Architecture