Object-Oriented Databases
Course Review

- Exam Information
- Summary
- OODBMS Architectures

Exam

- Session examination
- Oral exam in English
- Duration of 15 minutes
- 5 ECTS

Course Summary

I. Basics of Object-Oriented Databases
   1. Introduction
   2. Object Persistence
   3. db4o

II. Advanced Concepts of Object-Oriented Databases
   4. Standards and Commercial Systems
   5. Storage and Indexing
   6. Version Models

III. Semantic Object Data Management
   7. OM Data Model and OM Data Model Language
   8. Design and Implementation of OMS Avon
   9. Context-Aware Data Management

Persistence Strategies

- Persistence by inheritance
  - persistence capabilities inherited from pre-defined persistent class
  - Versant (C++), Objectivity/DB (C++)
- Persistence by instantiation
  - objects made persistent and get persistence capabilities upon instantiation
  - ObjectStore (C++)
- Persistence by reachability
  - objects made persistent if reachable from other persistent object
  - O_2 (C++/Java), ObjectStore (Java), Versant (Java/Smalltalk), Objectivity/DB (Java/Smalltalk), db4o (Java/.NET), ODMG

OODBMS Architectures

- RDBMS architectures are very similar
  - server centric, index-based, relational algebra execution engine
  - performance and scalability numbers vary by small percentages
- OODBMS architectures vary considerably and exhibit wildly different characteristics
  - performance and scalability numbers may vary by orders of magnitude
- OODBMS architectures and their impact on expectations of early adopters can be seen as **one** of the factors for the, initially, limited success of OODBMS
- It is important to consider application characteristics and understand which OODBMS architecture is best suited

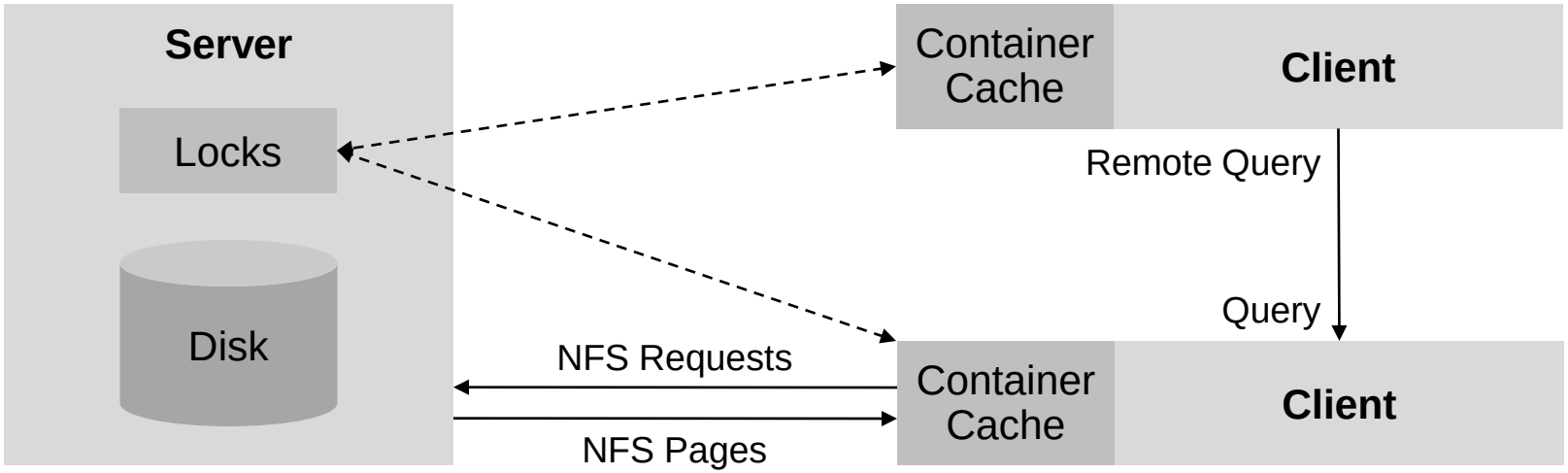
The Major Factors

- Key distinguishing implementation differences lead to vastly different runtime characteristics
- Primary areas impacting on performance include
  - core architecture
  - concurrency model
  - network model
  - query implementation
  - identity management
- Other feature functionality may also have an impact depending on application characteristics

Core Architecture

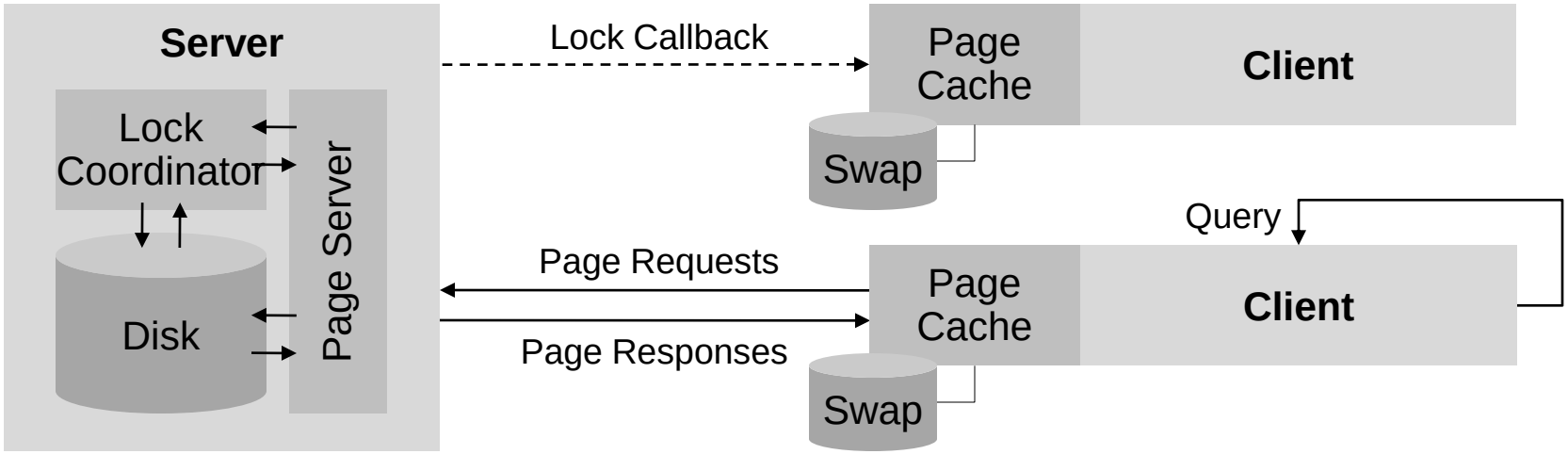
- Core architecture determines the following aspects
  - caching
  - query processing
  - transaction management
  - object life cycle management, i.e. tracking of new, dirty and deleted
- Three architectural variants are popular in OODBMS
  - container-based
  - page-based
  - object-based
- Name of the architecture reflects both
  - unit of transfer in network calls
  - lowest level of locking granularity

Container-Based Architecture



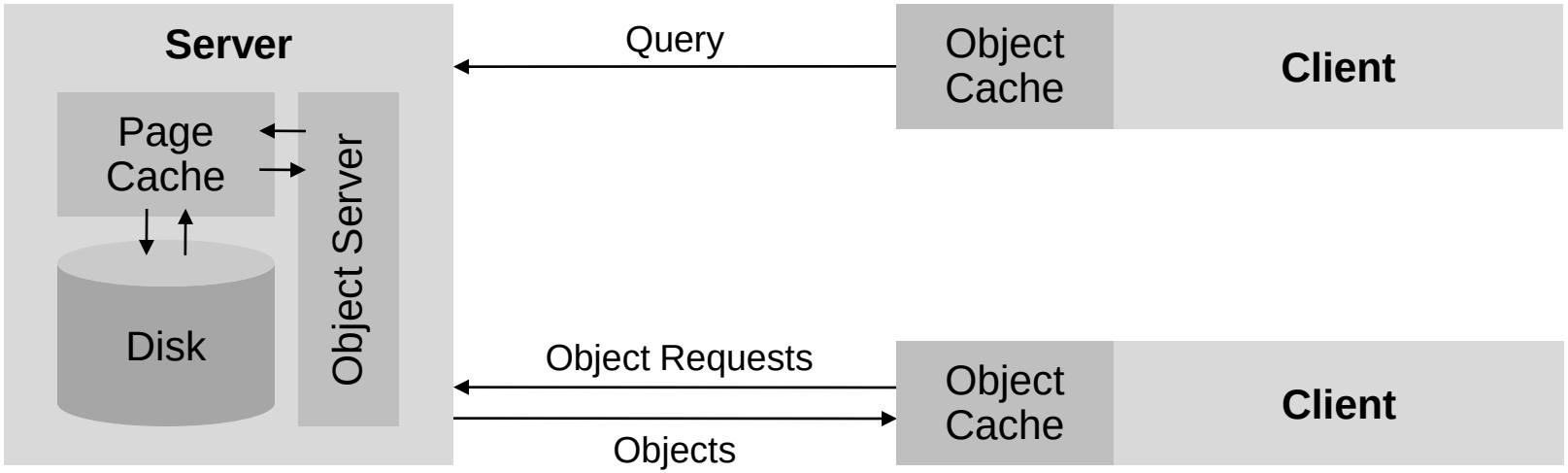
- Ships disk segments (containers) across the network
- Client libraries implement database functionality
  - container caching, query processing, object life cycle management and transactions
- All objects must reside inside a container
- Container model is layered over application domain model

Page-Based Architecture



- Ships pages of disk across the network
  - pages get address translated into virtual memory of an application
- Client libraries implement database functionality
  - container caching, query processing, object life cycle management and transactions
- Typically, object placement strategies are implemented

Object-Based Architecture



- Ships objects across the network
- Caching and behaviour in both client and server
  - **server:** page cache, indexes, locks, queries and transactions
  - **client libraries:** object caching, local locking and object life cycles
- No object placement strategies have to be implemented

Concurrency Model

- The three core architectures have differing concurrency models to provide transaction isolation
  - container concurrency
  - page concurrency
  - object concurrency
- All implementations of these models
  - are tightly coupled with network characteristics
  - can cache locks locally at the client across transaction boundaries
  - are likely to require lock coordination and cache consistency operations for updates
- In container and page-based systems, object placement and locking are tightly coupled
- In object-based systems, these issues are orthogonal

Container Concurrency

- Separate lock server coordinates concurrent access to data in same container
- Locking algorithm
  - clients requests lock before caching a container
  - locks are generally released at transaction boundaries
  - write request will establish a queue if there already read requests
  - subsequent read and write requests inserted in queue
  - after write request has been filled, other requests are filled
  - queue disappears if no further write requests
  - clients caching an updated container must refresh it before read
- As containers hold many objects, possibility of false waits and deadlocks

Page Concurrency

- Page server coordinates concurrent access
  - tracks which applications are accessing each page
  - grants permission to lock pages locally at the client
- Locking algorithm
  - client request lock before caching a page
  - write request causes server to use lock callbacks
  - clients either release lock and give up permission to cache page, or request blocks on each objecting client for a specified timeout
  - clients that have released lock will refresh page on next access
- As locks and permissions are taken out at the page level, false waits and deadlocks can occur

Object Concurrency

- Server maintains queues at object level to control concurrency of access to the same object
- Locking algorithm
  - clients request lock before caching object locally
  - locks are generally released at transaction boundaries
  - write request will establish queue, if read locks already exist
  - all subsequent requests are inserted into queue
  - after write request has been filled, other requests are filled
  - queue disappears if no further write requests
  - clients caching an updated object must refresh it before read
- Since locking is done at object level, no false waits or deadlocks can occur

Network Model

- Core architectures imply three different network models
  - container network model
  - page network model
  - object network model
- In a distributed system, different network models affect
  - bandwidth utilisation
  - performance
- Network model is closely tied to locking model
  - information transfer occurs upon lock, if not already cached locally
  - lowest granularity level of network transfer is equal to lowest granularity level of locking

Network Models

- Container network model
  - object request translated to container request
  - entire container transferred, even if not all objects are accessed
  - locks are place on the whole container
- Page network model
  - object request translated to page request
  - entire page transferred, even if not all objects are accessed
  - locks are placed on the whole page
- Object network model
  - unit of transfer is an object or collection of objects
  - object or collection request may include a depth request
  - locks are placed on one object or all objects within a collection

Query Implementation

- OODBMS focus on supporting seamless navigation between related objects using language constructs
  - native support of navigational access is a key advantage of OODBMS over the RDBMS complex join concept
  - relationships are a static part of the system rather than runtime computed
- Querying data in OODBMS consists of two steps
  - access first level objects of a use case
  - use navigation to access related objects
- Query implementation impacts on
  - where the query execution takes place
  - flexibility of what can be queried
  - indexing capabilities

Container Query

- Query processing at the client
  - client may be another remote client
  - client is, however, a process separate from the NFS page server
  - “opposite” architecture compared to a relational database
- Query processing
  - all objects involved in the query must be identified by the database or container they reside in (which also includes potential indexes) and loaded into the client process for query execution
  - query returns the containers holding the result objects
- From network and locking perspective, result may contain objects that did not actually satisfy the query predicate

Page Query

- Query processing at the client
- Query processing
  - all pages containing objects involved in the query are loaded
  - query returns references to objects that satisfy the query predicate
  - pages containing these objects have, implicitly, already been loaded across the network and translated into the client memory
- Indexed query processing
  - all index pages relevant to the query are loaded
  - query returns references to objects that satisfy the query predicate
  - pages containing these objects may have to be loaded across the network and translated into the client memory
- From network and locking perspective, result may contain objects that did not actually satisfy the query predicate

Object Query

- Query execution engine runs within database server
  - any object is reachable via query, even if it has no relationships
  - indexes for object attributes maintained on server
- Query processing
  - query statement sent to server from client
  - query executed on server using optimiser and indexes
  - result set of objects satisfying the query predicate returned to client
- Only query statement and objects satisfying the query are transferred across the network

Identity Management

- OODBMS use object identity for establish uniqueness and implementing relationships
- Identity management impacts on
  - long-term operational behaviour
  - flexibility, data scalability and schema evolution
- Physical identity
  - unique identifier is dependent on the physical location
  - mutable, reusable, immobile and rigid
  - dereferencing very fast
- Logical identity
  - unique identifier is independent of the physical location
  - immutable, never reused, mobile and flexible
  - logical references need to be translated into physical references

OODBMS Architectures Revealed

- Objectivity/DB
  - container-based architecture
  - physical identity
- ObjectStore Enterprise
  - page-based architecture (queries can be executed on the server)
  - physical identity
- Versant Object Database
  - object-based architecture
  - logical identity
- db4o
  - object-based architecture
  - physical identity

Literature

- Versant Object Database
  - http://www.versant.com/
- Robert Greene: OODBMS Architectures
  - http://www.odbms.org/experts.html#article9
- Adrian Marriott: OODBMS Architectures Revisited
  - http://www.odbms.org/experts.html#article11
- Robert Greene: OODBMS Architectures Defended
  - http://www.odbms.org/experts.html#article12

Object-Oriented Databases
The End