

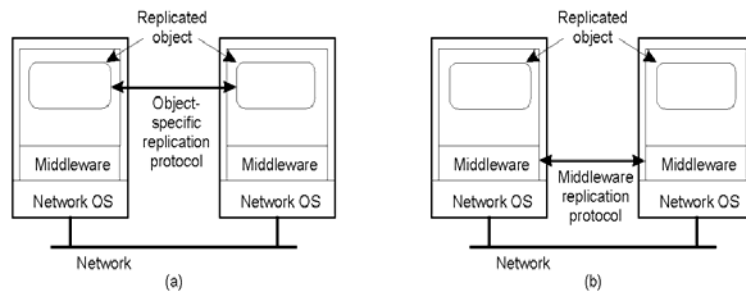
# Consistency and Replication

- Today:
  - Introduction
  - Consistency models
    - Data-centric consistency models
    - Client-centric consistency models
  - Thoughts for the mid-term

## Why replicate?

- Data replication: common technique in distributed systems
- Reliability
  - If one replica is unavailable or crashes, use another
  - Protect against corrupted data
- Performance
  - Scale with size of the distributed system (replicated web servers)
  - Scale in geographically distributed systems (web proxies)
- Key issue: need to maintain *consistency* of replicated data
  - If one copy is modified, others become inconsistent

# Object Replication

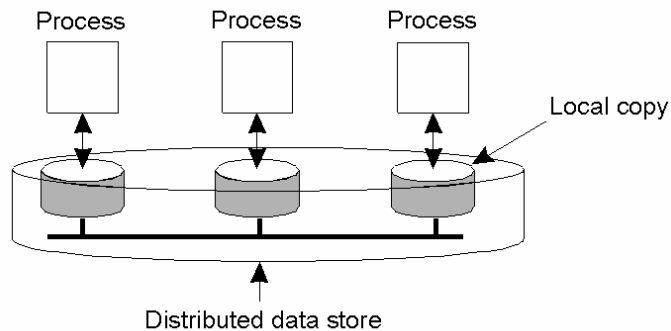


- Approach 1: application is responsible for replication
  - Application needs to handle consistency issues
- Approach 2: system (middleware) handles replication
  - Consistency issues are handled by the middleware
  - Simplifies application development but makes object-specific solutions harder

# Replication and Scaling

- Replication and caching used for system scalability
- Multiple copies:
  - Improves performance by reducing access latency
  - But higher network overheads of maintaining consistency
  - Example: object is replicated  $N$  times
    - Read frequency  $R$ , write frequency  $W$
    - If  $R \ll W$ , high consistency overhead and wasted messages
    - Consistency maintenance is itself an issue
      - What semantics to provide?
      - Tight consistency requires globally synchronized clocks!
- Solution: loosen consistency requirements
  - Variety of consistency semantics possible

# Data-Centric Consistency Models



- Consistency model (aka *consistency semantics*)
  - Contract between processes and the data store
    - If processes obey certain rules, data store will work correctly
  - All models attempt to return the results of the last write for a read operation
    - Differ in how “last” write is determined/defined

# Strict Consistency

- Any read always returns the result of the most recent write
  - Implicitly assumes the presence of a global clock
  - A write is immediately visible to all processes
    - Difficult to achieve in real systems (network delays can be variable)

# Sequential Consistency

- Sequential consistency: weaker than strict consistency
  - Assumes all operations are executed in some sequential order and each process issues operations in program order
    - Any valid interleaving is allowed
    - All agree on the same interleaving
    - Each process preserves its program order
    - Nothing is said about “most recent write”

P1:	W(x)a		
P2:	W(x)b		
P3:		R(x)b	R(x)a
P4:		R(x)b	R(x)a

(a)

P1:	W(x)a		
P2:	W(x)b		
P3:		R(x)b	R(x)a
P4:		R(x)a	R(x)b

(b)

# Linearizability

- Assumes sequential consistency *and*
  - If  $TS(x) < TS(y)$  then  $OP(x)$  should precede  $OP(y)$  in the sequence
  - Stronger than sequential consistency
  - Difference between linearizability and serializability?
    - Granularity: reads/writes versus transactions
- Example:

Process P1	Process P2	Process P3
------------	------------	------------

x = 1;	y = 1;	z = 1;
print ( y, z);	print (x, z);	print (x, y);

## Linearizability Example

- Four valid execution sequences for the processes of the previous slide. The vertical axis is time.

<p>x = 1; print ((y, z); y = 1; print (x, z); z = 1; print (x, y);</p> <p>Prints: 001011</p> <p>Signature: 001011 (a)</p>	<p>x = 1; y = 1; print (x,z); print(y, z); z = 1; print (x, y);</p> <p>Prints: 101011</p> <p>Signature: 101011 (b)</p>	<p>y = 1; z = 1; print (x, y); print (x, z); x = 1; print (y, z);</p> <p>Prints: 010111</p> <p>Signature: 110101 (c)</p>	<p>y = 1; x = 1; z = 1; print (x, z); print (y, z); print (x, y);</p> <p>Prints: 111111</p> <p>Signature: 111111 (d)</p>
---	--	--	--



## Causal consistency

- Causally related writes must be seen by all processes in the same order.
  - Concurrent writes may be seen in different orders on different machines

P1: W(x)a	
P2: R(x)a    W(x)b	
P3: R(x)b    R(x)a	
P4: R(x)a    R(x)b	

(a)

Not permitted

P1: W(x)a	
P2: W(x)b	
P3: R(x)b    R(x)a	
P4: R(x)a    R(x)b	

(b)

Permitted



## Other models

- FIFO consistency: writes from a process are seen by others in the same order. Writes from different processes may be seen in different order (even if causally related)
  - Relaxes causal consistency
  - Simple implementation: tag each write by (Proc ID, seq #)
- Even FIFO consistency may be too strong!
  - Requires all writes from a process be seen in order
- Assume use of critical sections for updates
  - Send final result of critical section everywhere
  - Do not worry about propagating intermediate results
    - Assume presence of synchronization primitives to define semantics



## Other Models

- Weak consistency
  - Accesses to synchronization variables associated with a data store are sequentially consistent
  - No operation on a synchronization variable is allowed to be performed until all previous writes have been completed everywhere
  - No read or write operation on data items are allowed to be performed until all previous operations to synchronization variables have been performed.
- Entry and release consistency
  - Assume shared data are made consistent at entry or exit points of critical sections



## Summary of Data-centric Consistency Models

Consistency	Description
Strict	Absolute time ordering of all shared accesses matters.
Linearizability	All processes must see all shared accesses in the same order. Accesses are furthermore ordered according to a (nonunique) global timestamp
Sequential	All processes see all shared accesses in the same order. Accesses are not ordered in time
Causal	All processes see causally-related shared accesses in the same order.
FIFO	All processes see writes from each other in the order they were used. Writes from different processes may not always be seen in that order

(a)

Consistency	Description
Weak	Shared data can be counted on to be consistent only after a synchronization is done
Release	Shared data are made consistent when a critical region is exited
Entry	Shared data pertaining to a critical region are made consistent when a critical region is entered.

(b)

## Mid-term Exam Comments

- Closed book, closed notes, 90 min
- Lectures 1-13 included on the test
  - Focus on things taught in class (lectures, in-class discussions)
  - Start with lecture notes, read corresponding sections from text
  - Supplementary readings are not included on the test.
- Exam structure: few short answer questions, mix of subjective and “design” questions
- Good luck!