Distributed Systems (ICE 601)

Replication & Consistency - Part 2

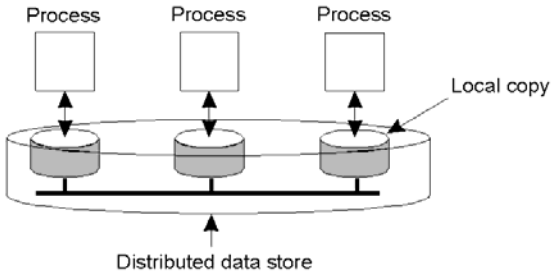
Dongman Lee
ICU

Class Overview

- Introduction
- Replication Model
- Request Ordering
- Consistency Models
- Consistency Protocols
- Case study
  - Transactions with Replicated Data
  - Lazy replication
  - ISIS

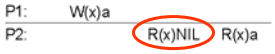
Data-Centric Consistency Models

- Consistency models
  - Definition
    - a contract between processes and the data store
  - Models
    - Strict
    - Sequential
    - Causal
    - FIFO
    - Weak
    - Release
    - Entry



Strict Consistency

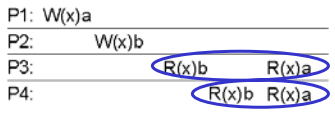
- Description
  - A read always returns the result of the most recent write
    - Existence of absolute global time and strict consistent data store are assumed
- Issues
  - Impossible to achieve absolute global time
  - No guarantee that at most a single operation be performed at a give time interval
    - Multiple operations can be performed simultaneously at a give time interval

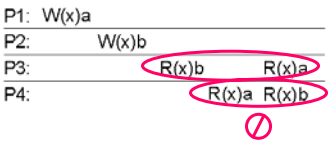


➔ *Linearizability* relaxes the absolute global time by using loosely synchronized clocks

Sequential Consistency

- Description
  - For any execution there is some interleaving of the series of operations issued by all processes that satisfies the followings:
    - SC1: the interleaved sequence of operations is such that if R(x)a occurs in the sequence, then either the last write operation that occurs before it in the interleaved sequence is W(x)a, or no write operation occurs before it and a is the initial value of x – data coherence
    - SC2: the order of operations in the interleaving is consistent with the *program order* in which each process executed them
- Consistency is enforced by the program order and coherence





Sequential Consistency (cont.)

- Four valid execution sequences for three concurrent processes

Process 1	Process 2	Process 3
x = 1;	y = 1;	z = 1;
print (y, z);	print (x,z);	print (x, y);

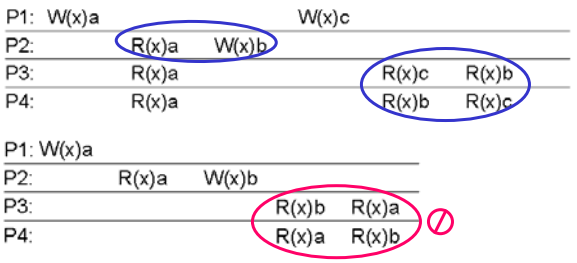
x = 1;	x = 1;	y = 1;	y = 1;
print ((y, z);	y = 1;	z = 1;	x = 1;
y = 1;	print (x, z);	print (x, y);	z = 1;
print (x, z);	print (y, z);	print (x, z);	print (x, z);
z = 1;	z = 1;	x = 1;	print (y, z);
print (x, y);	print (x, y);	print (y, z);	print (x, y);
Prints: 001011	Prints: 101011	Prints: 010111	Prints: 111111
Signature: 001011	Signature: 101011	Signature: 110101	Signature: 111111

Sequential Consistency (cont.)

- Issues
  - Changing the protocol to improve read performance makes write performance worse, and vice versa
    - Based on Lipton and Sanberg's proof: r + w >= t

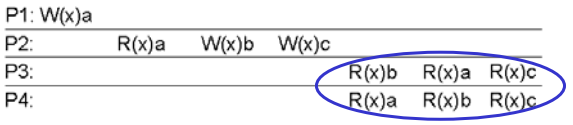
Causal Consistency

- Description
  - Sequential consistency is only applied to causally related operations
    - Writes that are potentially casually related must be seen by all processes in the same order
    - *Concurrent* writes may be seen in a *different order* on different machines
- Consistency is enforced by operation dependency
  - Vector timestamp is a way to support it



FIFO Consistency

- Description
  - Writes done by a single process are seen by all other processes in the order in which they were issued
  - Writes from different processes may be seen in a different order by different processes
- Consistency is enforced by
  - Tag each write operation with a (process, sequence number) pair
  - Perform writes per process in the order of their sequence number



FIFO Consistency (cont.)

- A key difference between sequential and FIFO consistency
  - Sequence consistency works as long as all processes agree the order of execution though it is non deterministic
  - FIFO consistency allows different processes to see the operations in a different order

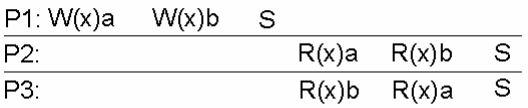
Weak Consistency

- Description
  - Only the final result of data in a critical section is required to all replicas
    - No need to propagate intermediate results to all replicas in order
    - *Synchronization variable* is used for this
- Properties
  - Accesses to synchronization variables associated with a data store are sequentially consistent
    - All processes see all operations on synchronization variables in the same order
  - No operation on a synchronization variable is allowed to be performed until all previous writes have been completed everywhere
    - All writes at local copies are forced to be done
  - No read or write operation on data items are allowed to be performed until all previous operations to synchronization variables have been performed
    - A process can be sure of getting the most recent values

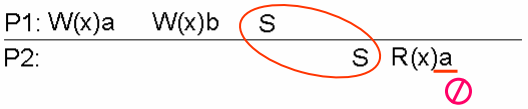
Weak Consistency (cont.)

- Key points
  - ➔ Enforces consistency on a group of operations, not on individual reads and writes
    - useful when most accesses come in cluster
  - ➔ Limit only time when consistency holds, rather than limiting the form of consistency
    - sequential consistency is enforced between groups of operations with weak consistency

Weak Consistency (cont.)



(a)



(b)

a) A valid sequence of events for weak consistency.
b) An invalid sequence for weak consistency.

Release Consistency

- Description
  - It's difficult to tell the difference between entering a critical section and leaving one in weak consistency
    - Two synchronization operations, *acquire* and *release*, instead of one
- Basics
  - Barrier
    - A synchronization mechanism that prevents any process from starting phase n+1 of a program until all processes have finished phase n
    - When a process arrives at a barrier, it must wait until all other processes get there as well
      - departure from barrier on an acquire
      - arrival at barrier on a release

Release Consistency (cont.)

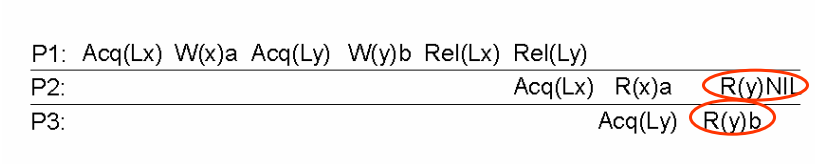
- Distributed data store is release consistent if it obeys the followings
  - Before a read or write operation on shared data is performed, all previous acquires done by the process must have completed successfully.
  - Before a release is allowed to be performed, all previous reads and writes by the process must have completed
  - Accesses to synchronization variables are FIFO consistent (sequential consistency is not required)
- Lazy release consistency
  - Relax the update until necessary

Entry Consistency

- Description
  - Synchronization variables are required to be associated with shared data items, not the entire share data
- Conditions
  - An acquire access of a synchronization variable is not allowed to perform with respect to a process until all updates to the guarded shared data have been performed with respect to that process
    - At acquire, all remote changes to the guarded date must be visible
  - Before an exclusive mode access to a synchronization variable by a process is allowed to perform with respect to that process, no other process may hold the synchronization variable, not even in nonexclusive mode
    - Before updating, a process must enter a critical section
  - After an exclusive mode access to a synchronization variable has been performed, any other process's next nonexclusive mode access to that synchronization variable may not be performed until it has performed with respect to that variable's owner
    - For nonexclusive mode access, a process has to check with the sync owner

Entry Consistency (cont.)

- Example



Summary of 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 (non-unique) 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)

a) Consistency models not using synchronization operations.

b) Models with synchronization operations.