CprE 450/550X
Distributed Systems and Middleware

Consistency & Replication

Yong Guan
3216 Coover
Tel: (515) 294-8378
Email: guan@ee.iastate.edu

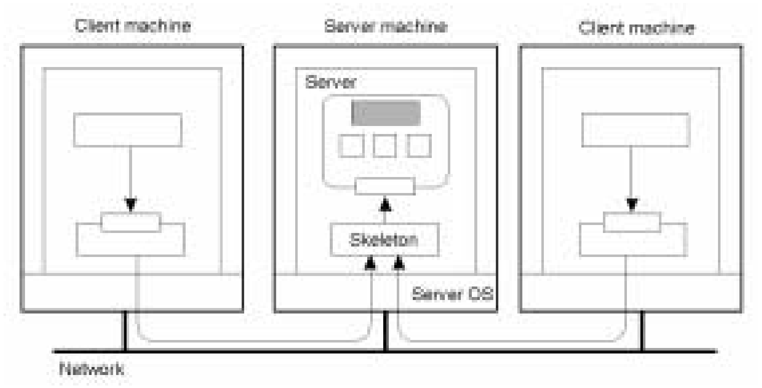
April 8, 2003

Readings for Today's Lecture

- References
  - Chapter 6 of "Distributed Systems: Principles and Paradigms"

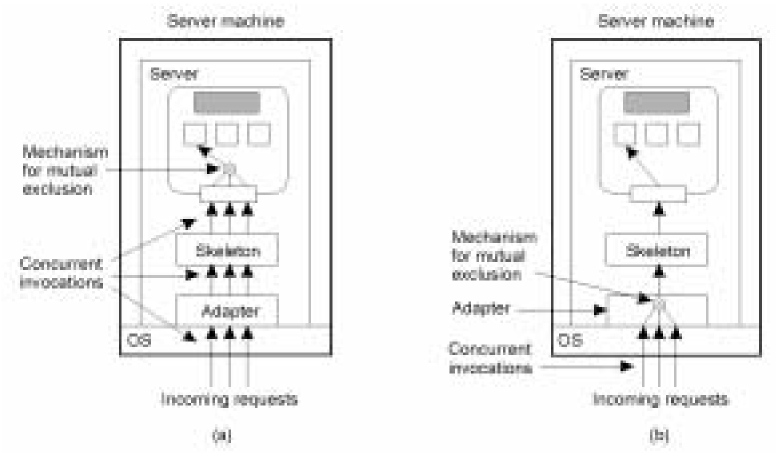
Introduction to Consistency & Replication

Object Replication



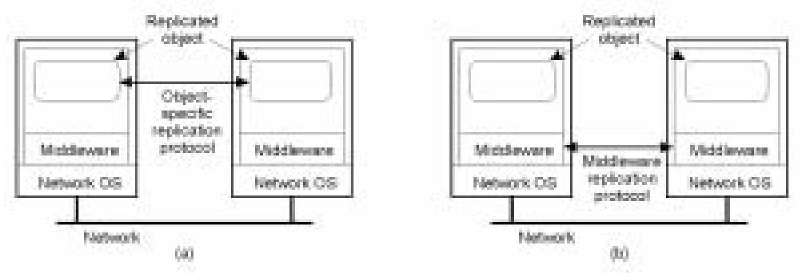
Organization of a distributed remote object shared by two different clients.

Object Replication (2)



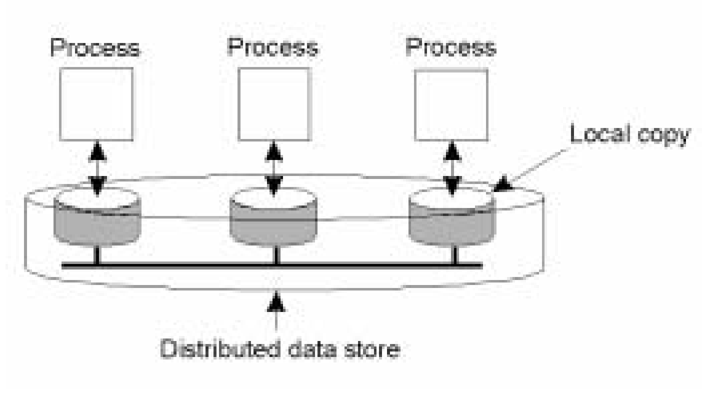
a) A remote object capable of handling concurrent invocations on its own.
b) A remote object for which an object adapter is required to handle concurrent invocations

Object Replication (3)



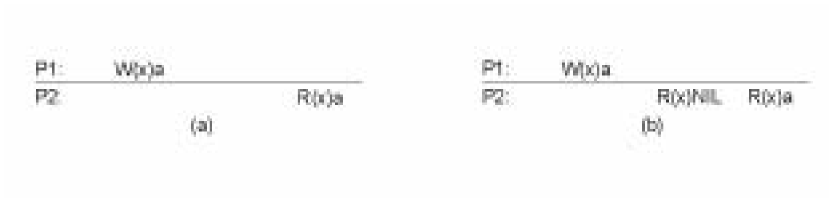
a) A distributed system for replication-aware distributed objects.
b) A distributed system responsible for replica management

Data-Centric Consistency Models



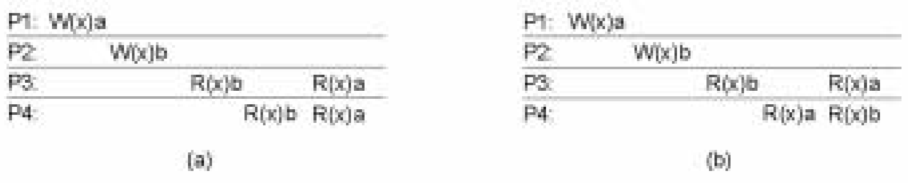
The general organization of a logical data store, physically distributed and replicated across multiple processes.

Strict Consistency

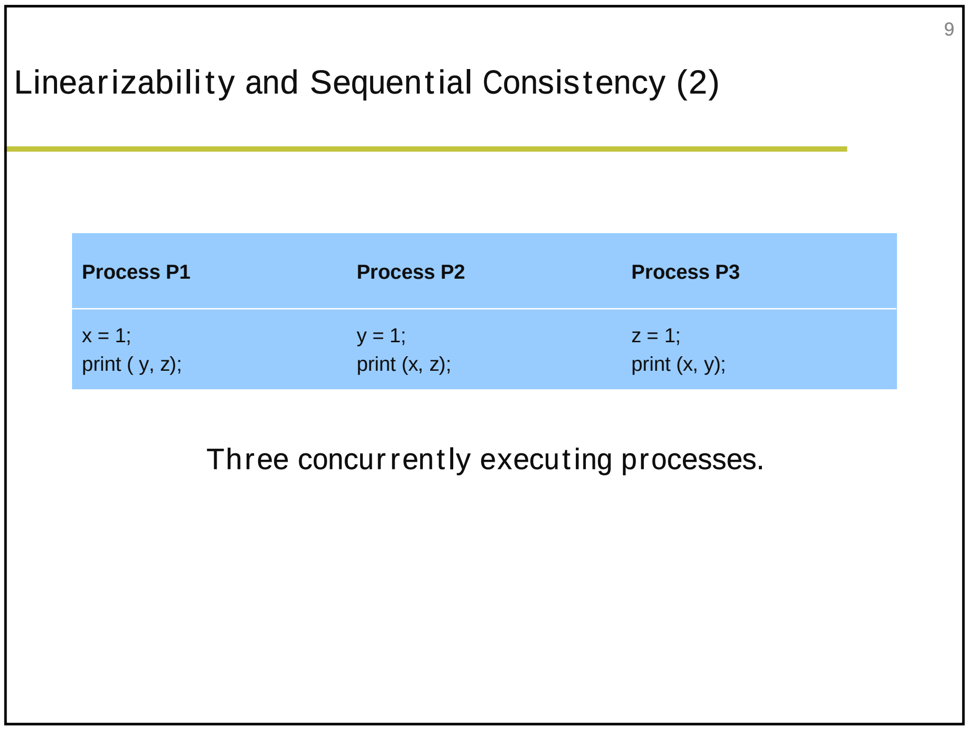


- Behavior of two processes, operating on the same data item.
- A strictly consistent store.
- A store that is not strictly consistent.

Linearizability and Sequential Consistency (1)



a) A sequentially consistent data store.
b) A data store that is not sequentially consistent.

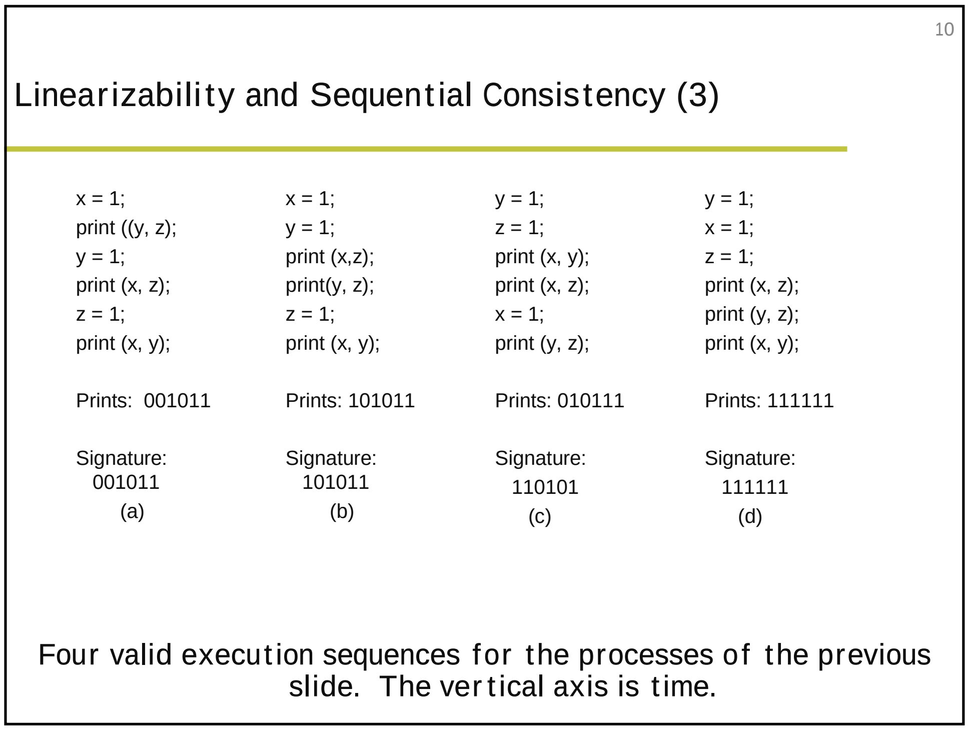


Linearizability and Sequential Consistency (2)

Process P1	**Process P2**	**Process P3**

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

Three concurrently executing processes.



Linearizability and Sequential Consistency (3)

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

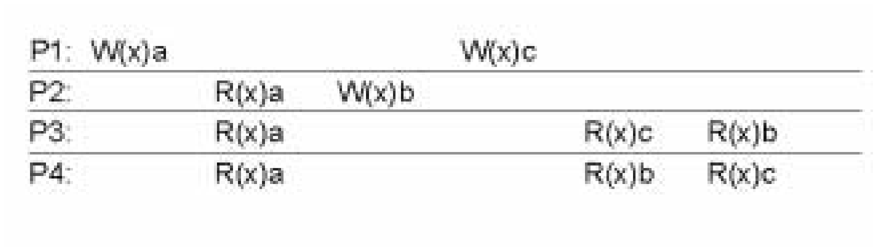
Prints: 001011	Prints: 101011	Prints: 010111	Prints: 111111
Signature: 001011	Signature: 101011	Signature: 110101	Signature: 111111
(a)	(b)	(c)	(d)

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

Casual Consistency (1)

- Necessary condition:
  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.

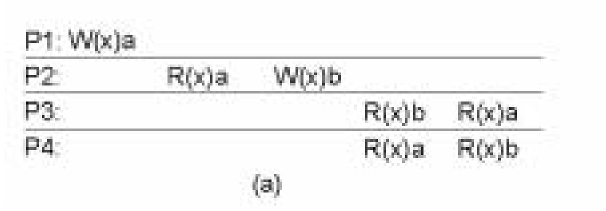
Casual Consistency (2)

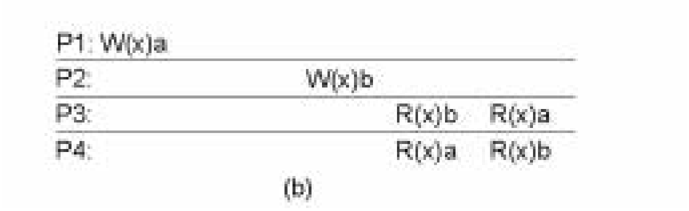


P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b			
P3:		R(x)a			R(x)c	R(x)b
P4:		R(x)a			R(x)b	R(x)c

This sequence is allowed with a casually-consistent store, but not with sequentially or strictly consistent store.

Casual Consistency (3)



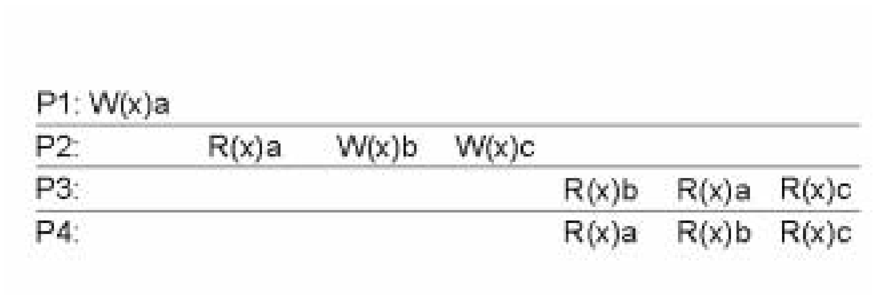


a) A violation of a casually-consistent store.
b) A correct sequence of events in a casually-consistent store.

FIFO Consistency (1)

- Necessary Condition:
  Writes done by a single process are seen by all other processes in the order in which they were issued, but writes from different processes may be seen in a different order by different processes.

FIFO Consistency (2)



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

A valid sequence of events of FIFO consistency

FIFO Consistency (3)

x = 1;	x = 1;	y = 1;
print (y, z);	y = 1;	print (x, z);
y = 1;	**print(x, z);**	z = 1;
print(x, z);	print (y, z);	**print (x, y);**
z = 1;	z = 1;	x = 1;
print (x, y);	print (x, y);	print (y, z);
Prints: 00	Prints: 10	Prints: 01
(a)	(b)	(c)

Statement execution as seen by the three processes from the previous slide. The statements in bold are the ones that generate the output shown.

FIFO Consistency (4)

Process P1	Process P2

x = 1;
if (y == 0) kill (P2);
	y = 1;
	if (x == 0) kill (P1);

Two concurrent processes.

Weak Consistency (1)

- Properties:
- 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.

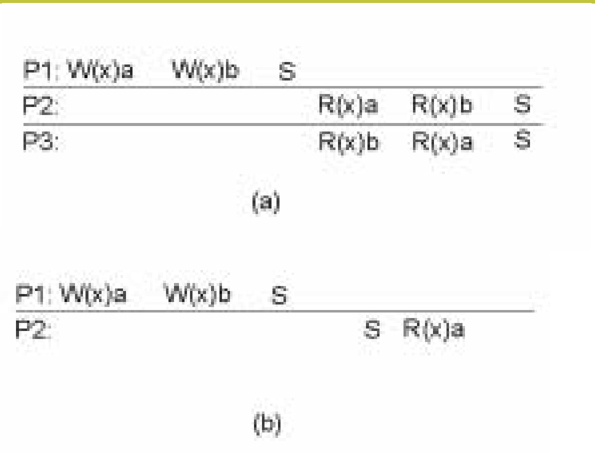
Weak Consistency (2)

```
int a, b, c, d, e, x, y;              /* variables */
int *p, *q;                           /* pointers */
int f( int *p, int *q);               /* function prototype */

a = x * x;                            /* a stored in register */
b = y * y;                            /* b as well */
c = a*a*a + b*b + a * b;              /* used later */
d = a * a * c;                        /* used later */
p = &a;                               /* p gets address of a */
q = &b                                /* q gets address of b */
e = f(p, q)                           /* function call */
```

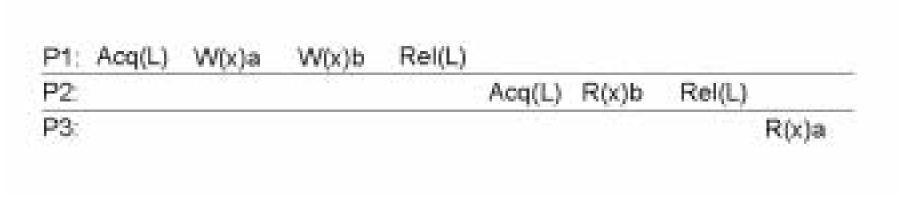
A program fragment in which some variables may be kept in registers.

Weak Consistency (3)



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

Release Consistency (1)



A valid event sequence for release consistency.

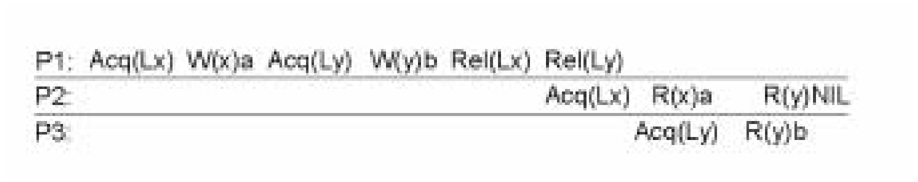
Release Consistency (2)

- Rules:
- 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).

Entry Consistency (1)

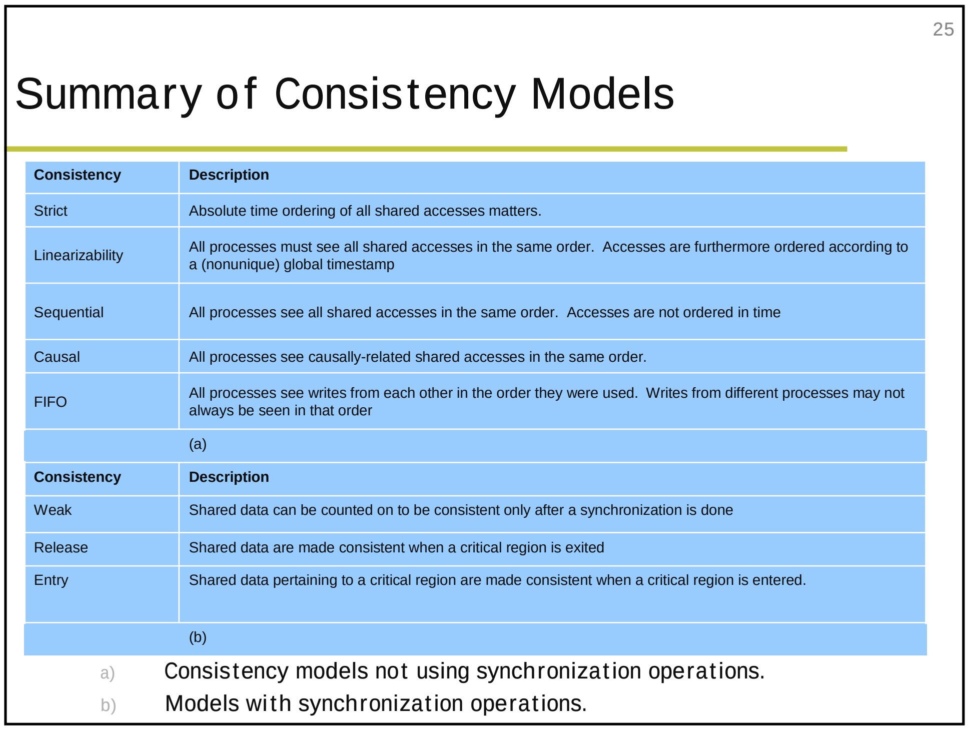
- Conditions:
- An acquire access of a synchronization variable is not allowed to perform with respect to a process until all updates to the guarded shared data have been performed with respect to that process.
- Before an exclusive mode access to a synchronization variable by a process is allowed to perform with respect to that process, no other process may hold the synchronization variable, not even in nonexclusive mode.
- 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.

Entry Consistency (2)



```
P1:  Acq(Lx)  W(x)a  Acq(Ly)  W(y)b  Rel(Lx)  Rel(Ly)
P2:                                           Acq(Lx)  R(x)a     R(y)NIL
P3:                                                 Acq(Ly)  R(y)b
```

- A valid event sequence for entry consistency.



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 (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)

a) Consistency models not using synchronization operations.

b) Models with synchronization operations.