Distributed Systems
Principles and Paradigms

Maarten van Steen

VU Amsterdam, Dept. Computer Science
Room R4.20, steen@cs.vu.nl

Chapter 07: Consistency & Replication

Version: November 26, 2012

Consistency & replication

- Introduction (what's it all about)
- Data-centric consistency
- Client-centric consistency
- Replica management
- Consistency protocols

Performance and scalability

Main issue

To keep replicas consistent, we generally need to ensure that all conflicting operations are done in the the same order everywhere

Conflicting operations

From the world of transactions:

- Read–write conflict: a read operation and a write operation act concurrently
- Write–write conflict: two concurrent write operations

Issue

Guaranteeing global ordering on conflicting operations may be a costly operation, downgrading scalability Solution: weaken consistency requirements so that hopefully global synchronization can be avoided

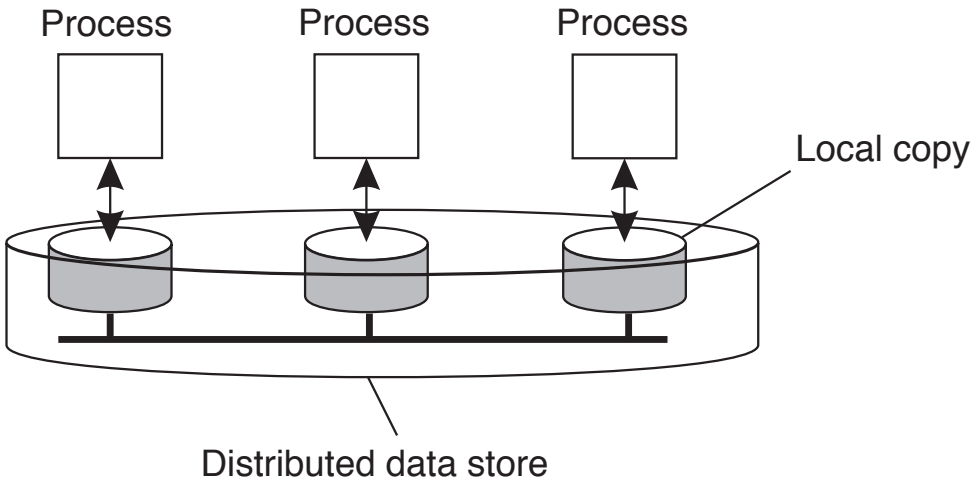
Data-centric consistency models

Consistency model

A contract between a (distributed) data store and processes, in which the data store specifies precisely what the results of read and write operations are in the presence of concurrency.

Essential

A data store is a distributed collection of storages:



Continuous Consistency

Observation

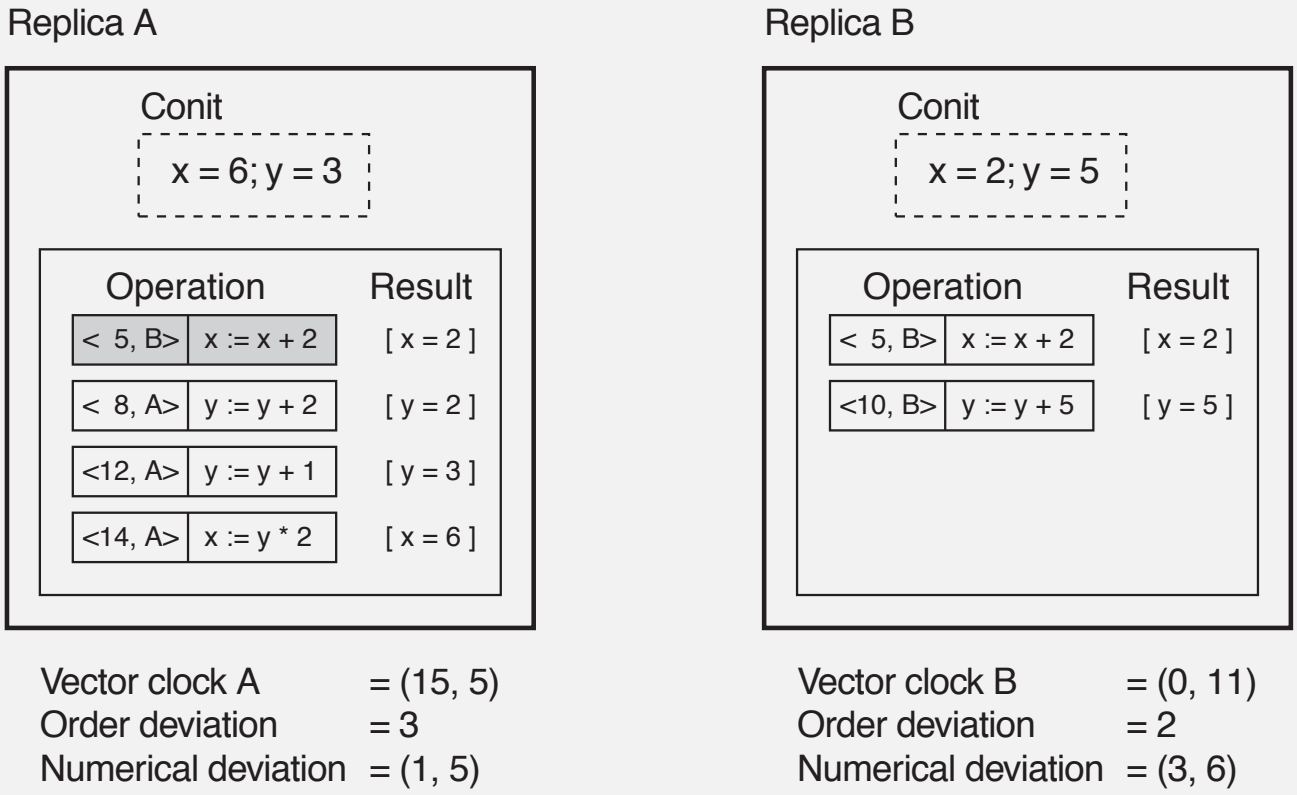
We can actually talk a about a degree of consistency:

- replicas may differ in their numerical value
- replicas may differ in their relative staleness
- there may be differences with respect to (number and order) of performed update operations

Conit

Consistency unit $\Rightarrow$ specifies the data unit over which consistency is to be measured.

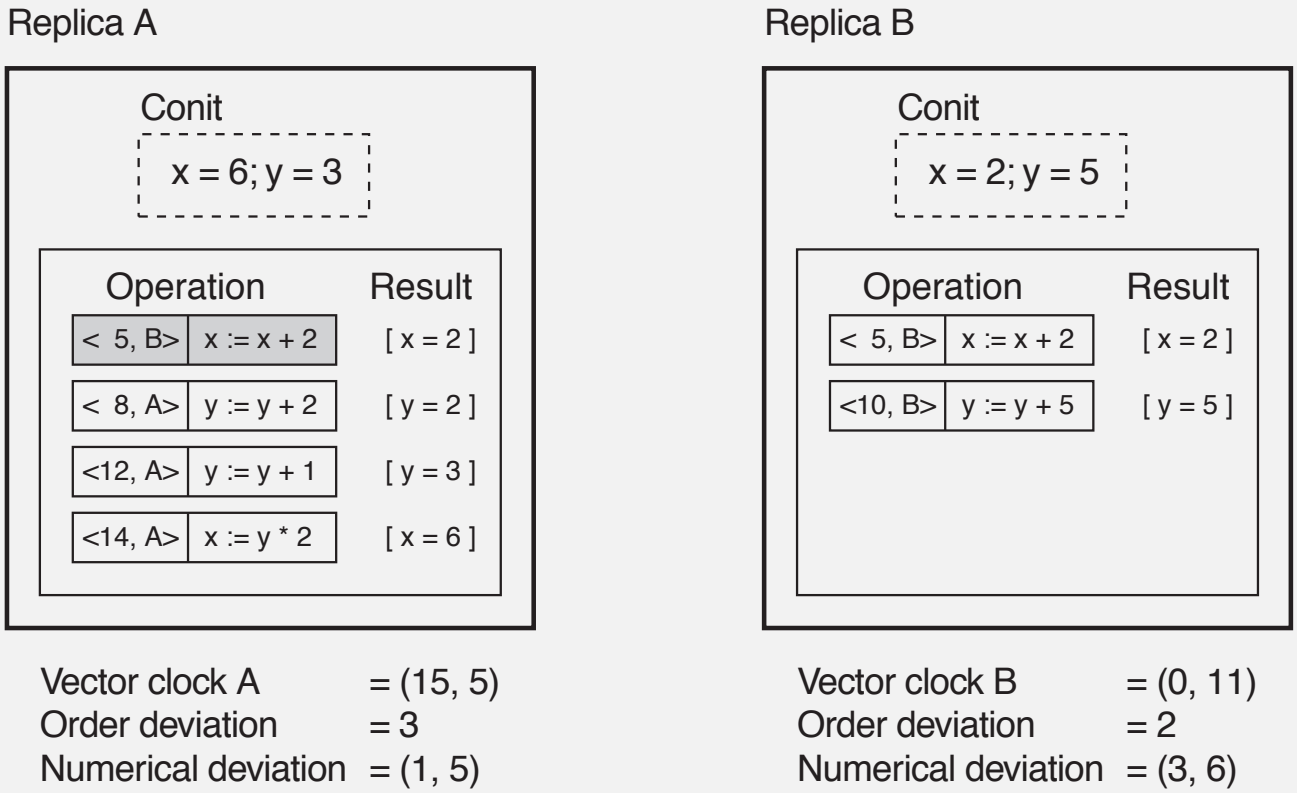
Example: Conit



Conit (contains the variables x and y)

- Each replica has a vector clock: ([known] time @ A, [known] time @ B)
- B sends A operation $[\langle 5, B\rangle: x := x + 2]$; A has made this operation permanent (cannot be rolled back)

Example: Conit



Conit (contains the variables *x* and *y*)

- *A* has three pending operations $\Rightarrow$ order deviation = 3
- *A* has missed one operation from *B*, yielding a max diff of 5 units $\Rightarrow$ (1,5)

Sequential consistency

Definition

The result of any execution is the same as if the operations of all processes were executed in some sequential order, and the operations of each individual process appear in this sequence in the order specified by its program.

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

(a)

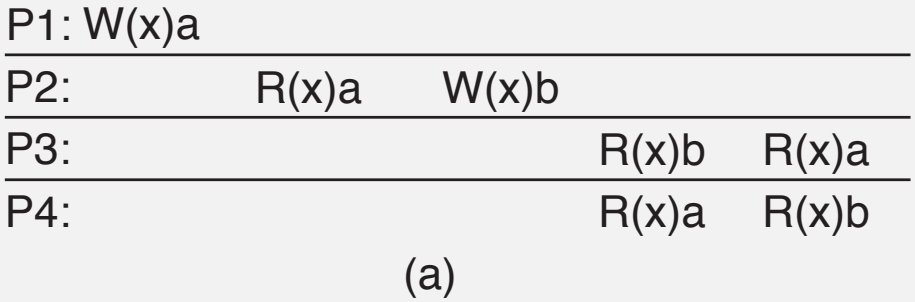
P1:	W(x)a			
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

(b)

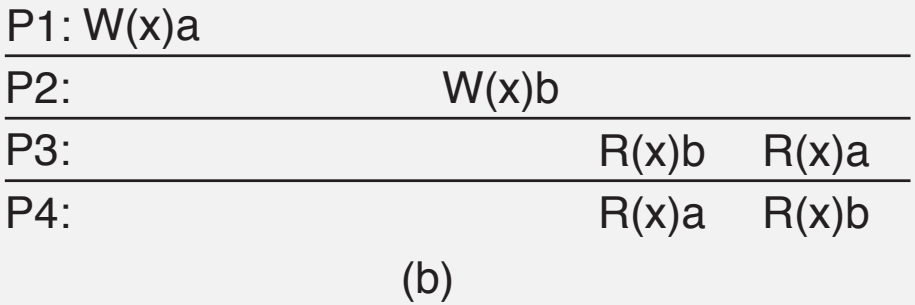
Causal consistency

Definition

Writes that are potentially causally related must be seen by all processes in the same order. Concurrent writes may be seen in a different order by different processes.



(a)



(b)

Grouping operations

Definition

- Accesses to synchronization variables are sequentially consistent.
- No access to a synchronization variable is allowed to be performed until all previous writes have completed everywhere.
- No data access is allowed to be performed until all previous accesses to synchronization variables have been performed.

Basic idea

You don't care that reads and writes of a series of operations are immediately known to other processes. You just want the effect of the series itself to be known.

Grouping operations

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			

Observation

Weak consistency implies that we need to lock and unlock data (implicitly or not).

Question

What would be a convenient way of making this consistency more or less transparent to programmers?

Client-centric consistency models

Overview

- System model
- Monotonic reads
- Monotonic writes
- Read-your-writes
- Write-follows-reads

Goal

Show how we can perhaps avoid systemwide consistency, by concentrating on what specific clients want, instead of what should be maintained by servers.

Consistency for mobile users

Example

Consider a distributed database to which you have access through your notebook. Assume your notebook acts as a front end to the database.

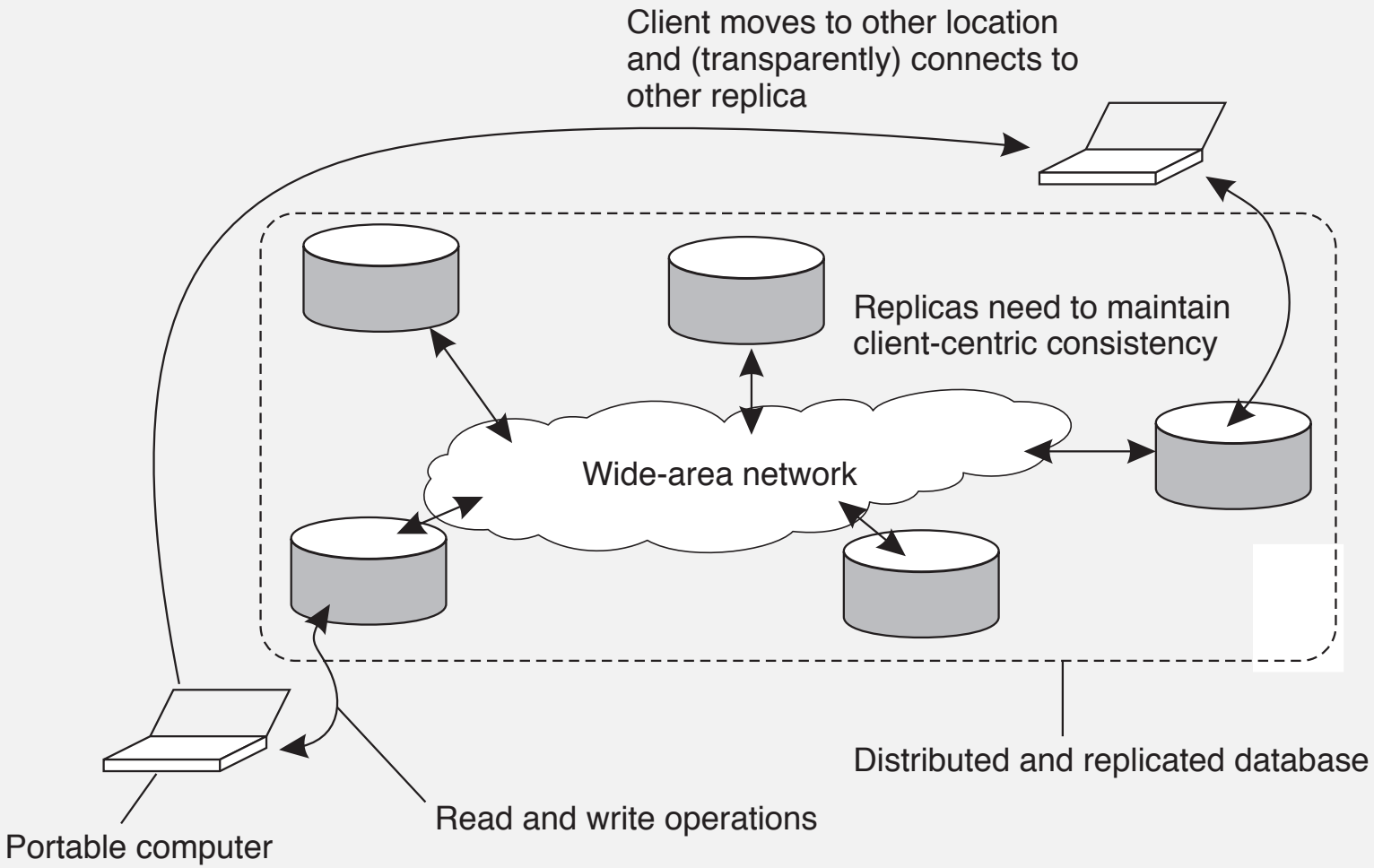
- At location *A* you access the database doing reads and updates.
- At location *B* you continue your work, but unless you access the same server as the one at location *A*, you may detect inconsistencies:
  - your updates at *A* may not have yet been propagated to *B*
  - you may be reading newer entries than the ones available at *A*
  - your updates at *B* may eventually conflict with those at *A*

Consistency for mobile users

Note

The only thing you really want is that the entries you updated and/or read at *A*, are in *B* the way you left them in *A*. In that case, the database will appear to be consistent to you.

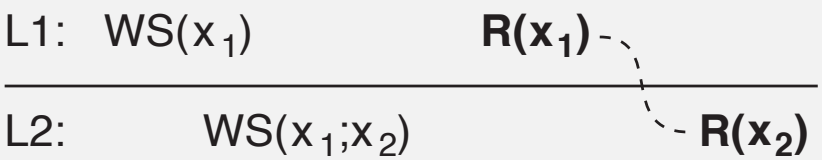
Basic architecture

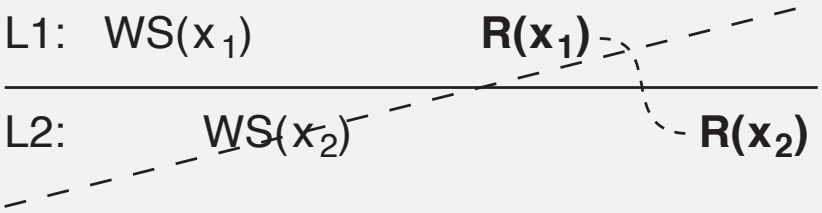


Monotonic reads

Definition

If a process reads the value of a data item x, any successive read operation on x by that process will always return that same or a more recent value.





Client-centric consistency: notation

Notation

- $WS(x_i[t])$ is the set of write operations (at L_i) that lead to version x_i of x (at time t)
- $WS(x_i[t_1]; x_j[t_2])$ indicates that it is known that $WS(x_i[t_1])$ is part of $WS(x_j[t_2])$.
- Note: Parameter t is omitted from figures.

Monotonic reads

Example

Automatically reading your personal calendar updates from different servers. Monotonic Reads guarantees that the user sees all updates, no matter from which server the automatic reading takes place.

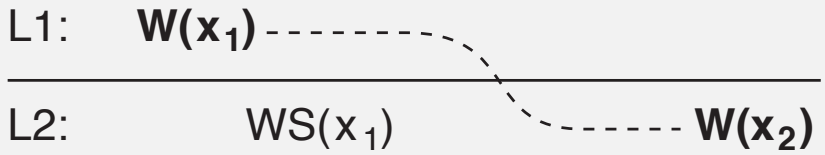
Example

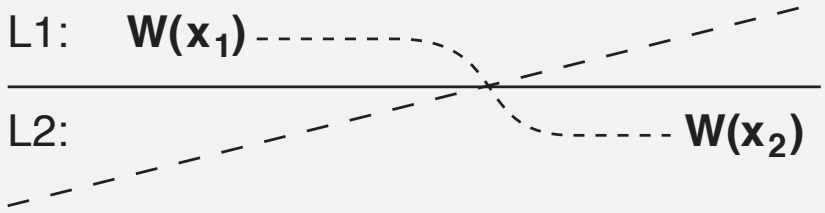
Reading (not modifying) incoming mail while you are on the move. Each time you connect to a different e-mail server, that server fetches (at least) all the updates from the server you previously visited.

Monotonic writes

Definition

A write operation by a process on a data item x is completed before any successive write operation on x by the same process.





Monotonic writes

Example

Updating a program at server S_2, and ensuring that all components on which compilation and linking depends, are also placed at S_2.

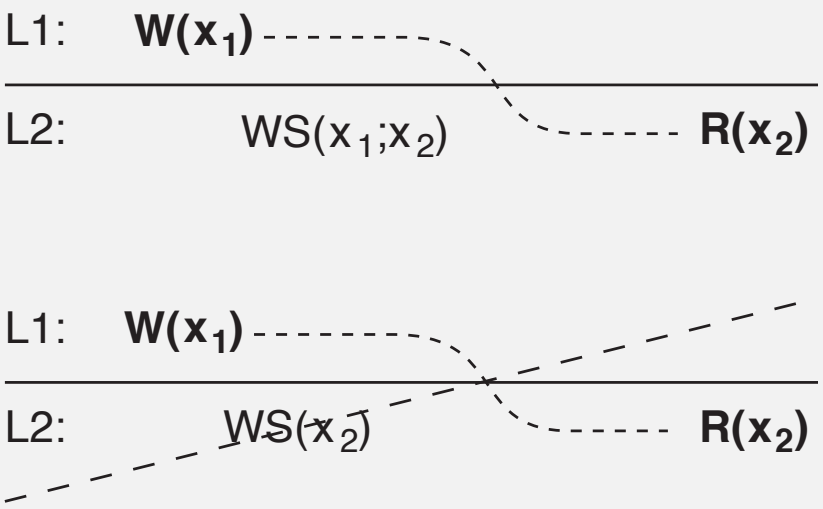
Example

Maintaining versions of replicated files in the correct order everywhere (propagate the previous version to the server where the newest version is installed).

Read your writes

Definition

The effect of a write operation by a process on data item x, will always be seen by a successive read operation on x by the same process.



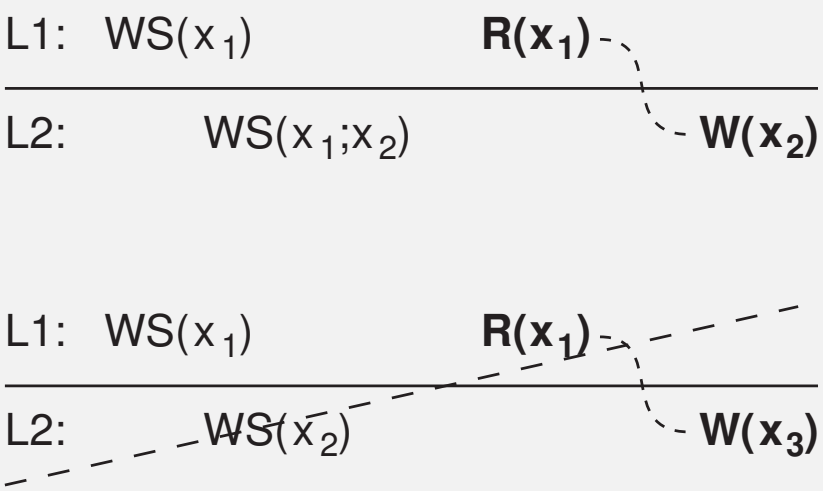
Example

Updating your Web page and guaranteeing that your Web browser shows the newest version instead of its cached copy.

Writes follow reads

Definition

A write operation by a process on a data item x following a previous read operation on x by the same process, is guaranteed to take place on the same or a more recent value of x that was read.



Example

See reactions to posted articles only if you have the original posting (a read "pulls in" the corresponding write operation).