第七章　一致性和复制

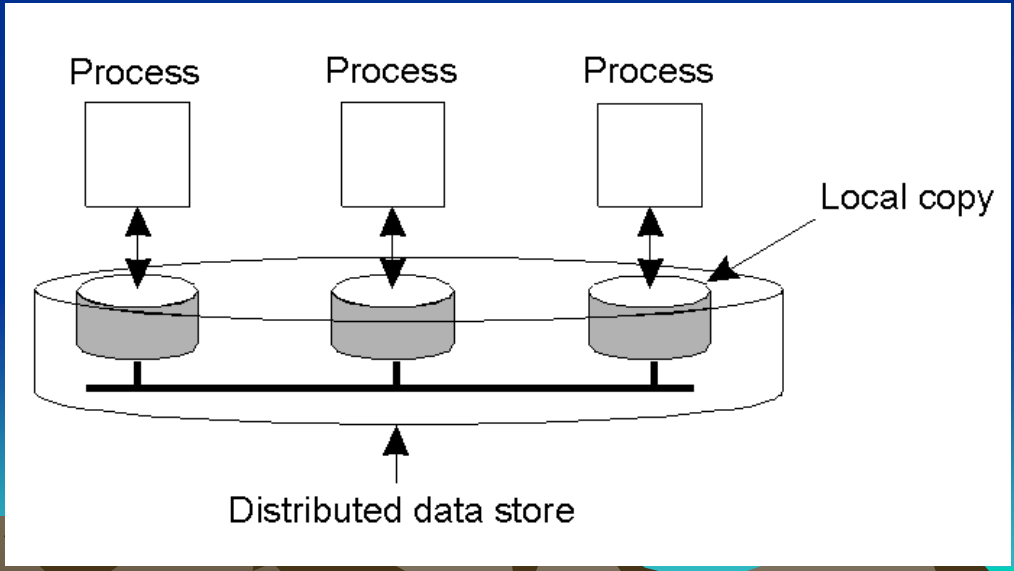
- 概述
- 以数据为中心的一致性模型
- 以客户为中心的一致性模型
- 复制管理
- 一致性协议

概述：复制的目的

- 可靠性
- 性能
  - 服务器数量扩展
  - 地理区域扩展
- 代价：一致性
  - 网络通信开销
  - 强一致性要求的原子操作很难快速完成
- 解决办法：
  - 放宽一致性方面的限制，放宽程度取决于复制数据的访问和更新模式以及数据的用途

以数据为中心的一致性模型

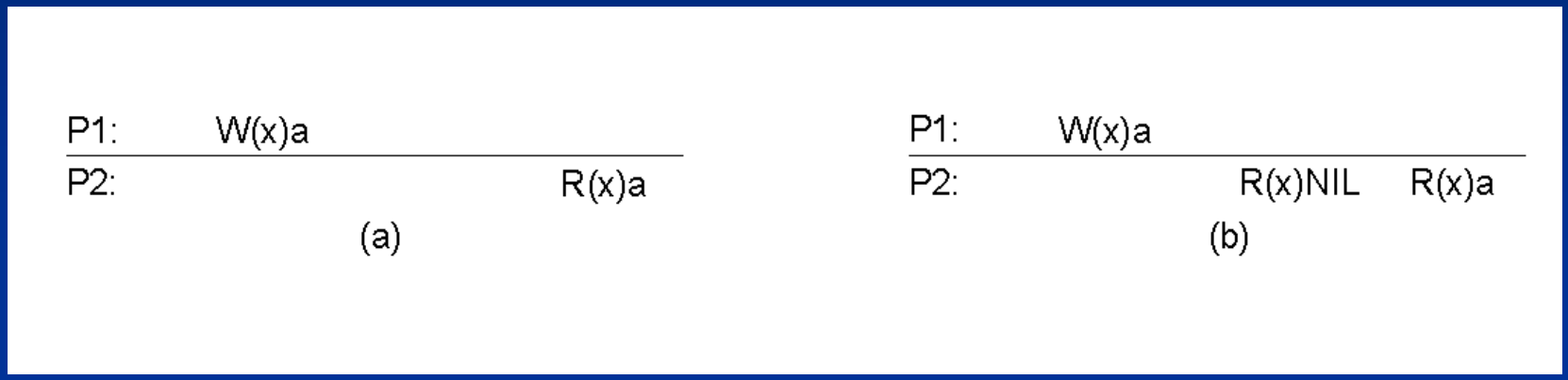
- 讨论共享数据读操作和写操作时的一致性问题
- 一致性模型实质上是进程和数据存储间的约定：如果进程同意遵守某些规则，数据存储将正常进行。正常情况下，进程的读操作应该返回最后一次写操作的结果
- 没有全局时钟，精确定义哪次写操作是最后一次写操作是困难的
- 作为全局时钟的替代，产生了一系列一致性模型，每种模型都有效地限制了一个数据项上执行一次读操作所应返回的值



逻辑数据存储的一般组织，物理上是分布的，并被复制到各个进程

严格一致性
（Strict Consistency）

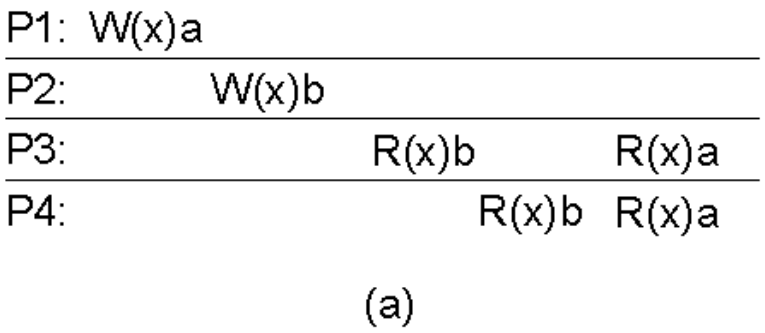
- 任何对数据项X的读操作将返回最近一次对X进行写操作的值
- 对所有进程来说，所有写操作都是瞬间可见的，系统维护着一个绝对的全局时间顺序



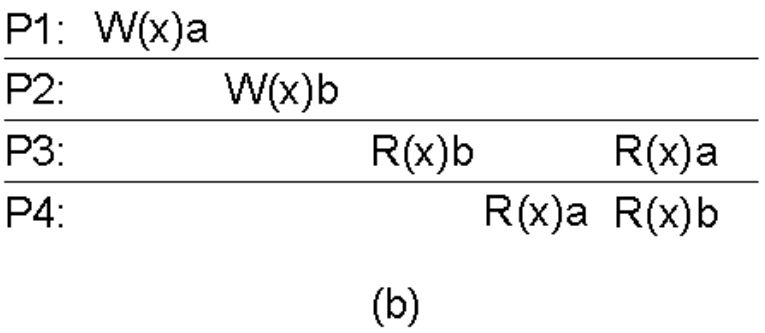
a) 严格的一致性存储
b) 非严格的一致性存储

顺序和线性化一致性
Sequential and Linearizability Consistency (1)

顺序一致性对存储器的限制比严格一致性要弱一些，要满足以下的条件：
（1） 每个进程的内部操作顺序是确定不变的；
（2） 假如所有的进程都对某一个存储单元执行操作，那么，它们的操作顺序是确定的，即任一进程都可以感知到这些数据同样的操作顺序。



(a)



(b)

a) 顺序一致的数据存储
b) 非顺序一致的数据存储

线性化和顺序一致性 (2)

- 线性化：
  - 弱于严格一致性而强于顺序一致性
  - 根据一系列时钟同步确定序列顺序（利用时间戳）
- 顺序一致性和线性化提供了程序开发人员在并发程序设计中期望的语义：所有写操作都以相同的顺序被每个进程看到。

因果一致性
Casual Consistency (1)

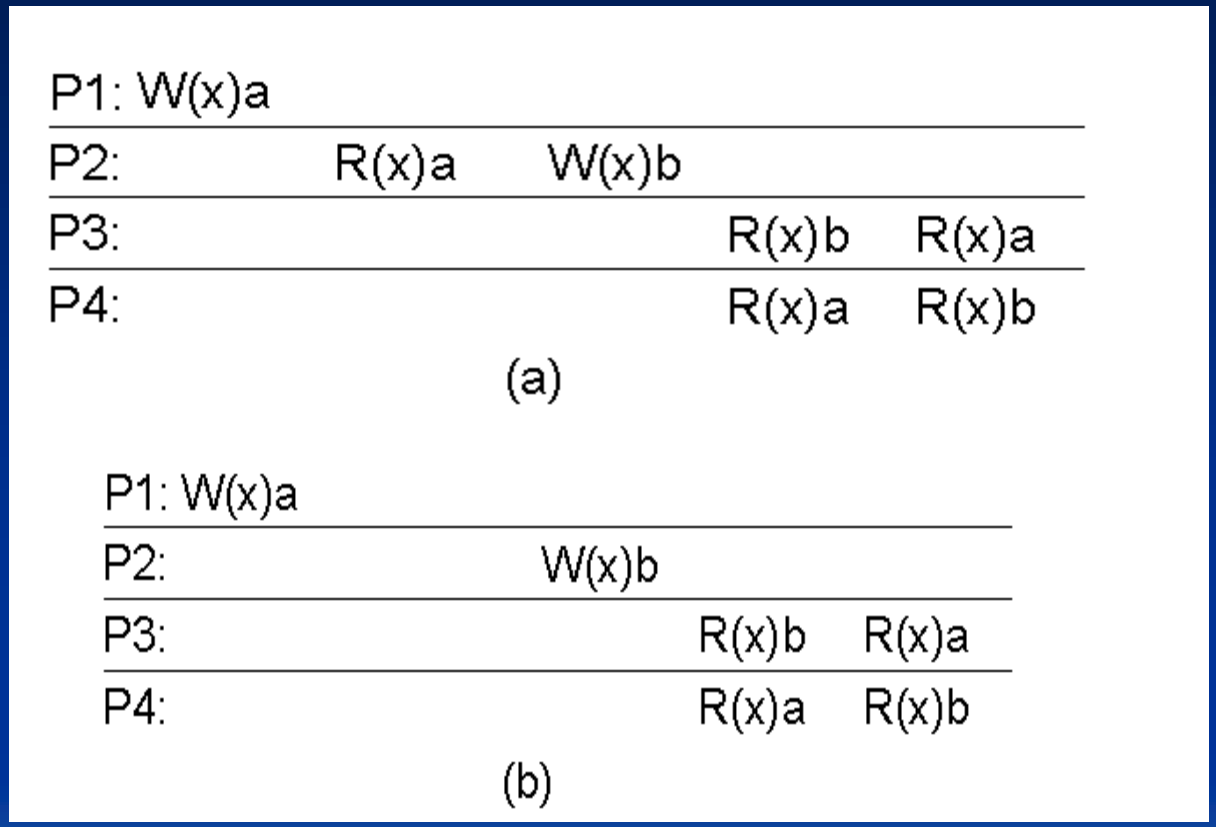
- 所有进程必须以相同的顺序看到具有潜在因果关系的写操作
- 不同机器上的进程可以以不同的顺序看到并发的写操作

因果一致性 (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

因果一致性存储允许的，但顺序和严格一致性存储不允许的顺序

因果一致性 (3)



a) 违背因果一致性的时间存储顺序
b) 符合因果一致性的时间存储顺序

FIFO 一致性
FIFO Consistency (1)

FIFO一致性模型是在因果一致性模型上的进一步弱化，它满足下面的条件：

- 由某一个进程完成的写操作可以被其他所有的进程按照顺序感知到，而从不同进程中来的写操作对不同的进程可以有不同的顺序。

FIFO 一致性 (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

符合FIFO 一致性的时间存储顺序

FIFO 一致性 (3)

与顺序一致性的区别:

- 顺序一致性：尽管语句的执行顺序是非确定的，但所有的进程对顺序达成一致
- FIFO 一致性：各个进程不需要达成一致，不同进程可以以不同的顺序看到

一致性总结

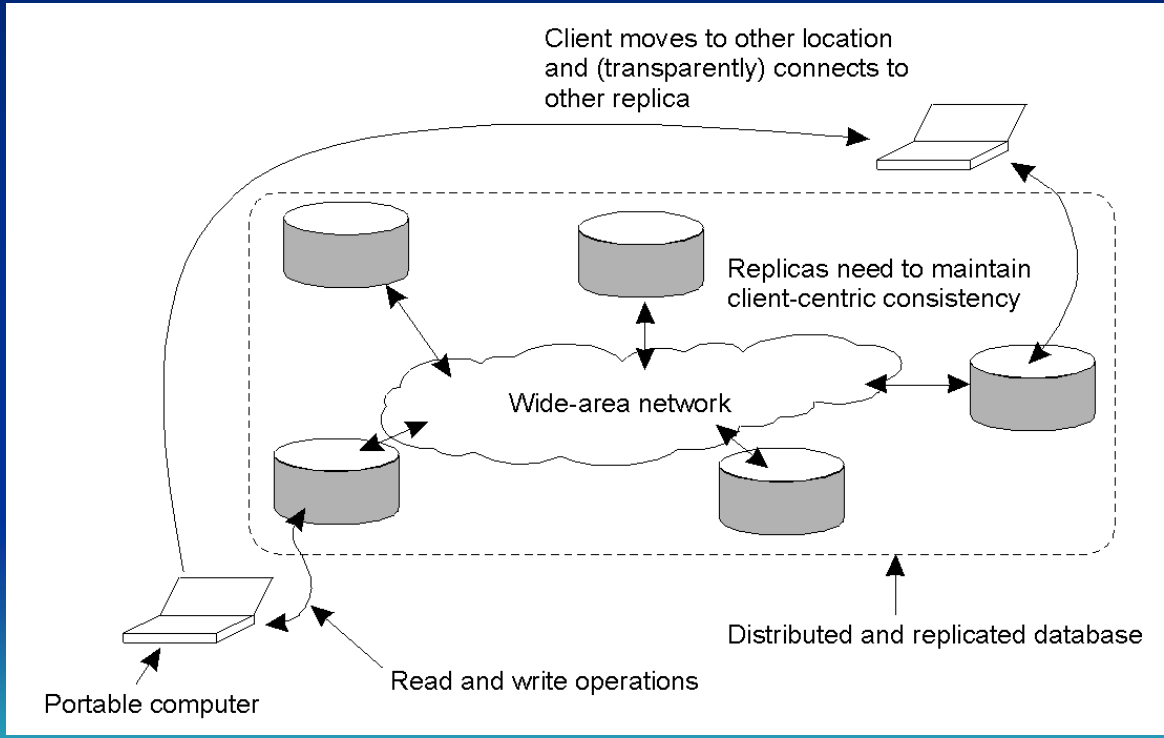
Consistency	Description
严格	所有共享访问按绝对时间排序
线性化	所有进程以相同顺序看到所有的共享访问。而且，访问是根据全局时间戳排序的
顺序	所有进程以相同顺序看到所有的共享访问。访问不是根据时间排序的
因果	所有进程以相同顺序看到因果相关的共享访问。
FIFO	所有进程以不同进程提出写操作的顺序相互看到写操作。来自不同进程的写操作可以不必总是以相同的顺序出现。

以客户为中心的一致性模型
最终一致性（Eventual Consistency）

- 最终一致性：很多情况下系统能容忍相对较高程度的不一致性，共同之处在于：只有一个或少数几个进程执行更新操作，如果较长时间内没有更新操作，那么副本将逐渐成为一致的
  - 数据库系统
  - DNS
  - WWW
- 特点：
  - 只有少数几个进程执行更新操作，只需要处理读写冲突
  - 能容忍相对较高程度的不一致性
  - 实现开销小
- 如果客户总是访问同一副本，最终一致性能工作得很好

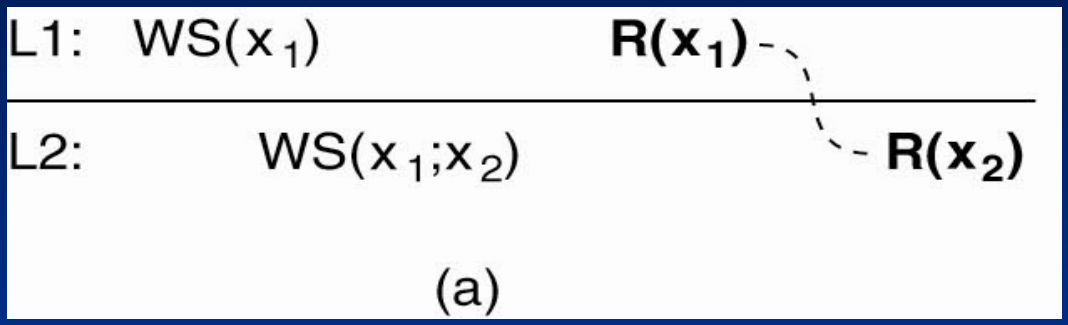
以客户为中心的一致性模型

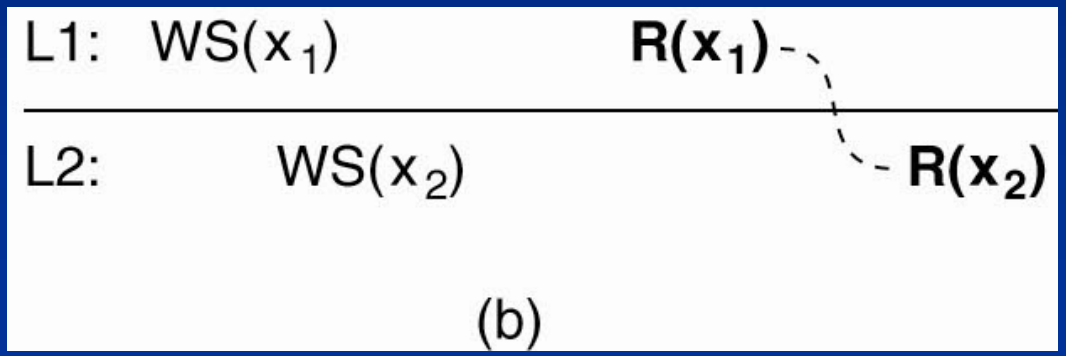
以客户为中心的一致性模型不考虑数据可能被多个用户共享的问题，而是集中考虑一个单独用户应被提供的一致性。



移动用户访问分布式数据库不同副本的原理

单调读（Monotonic Reads）





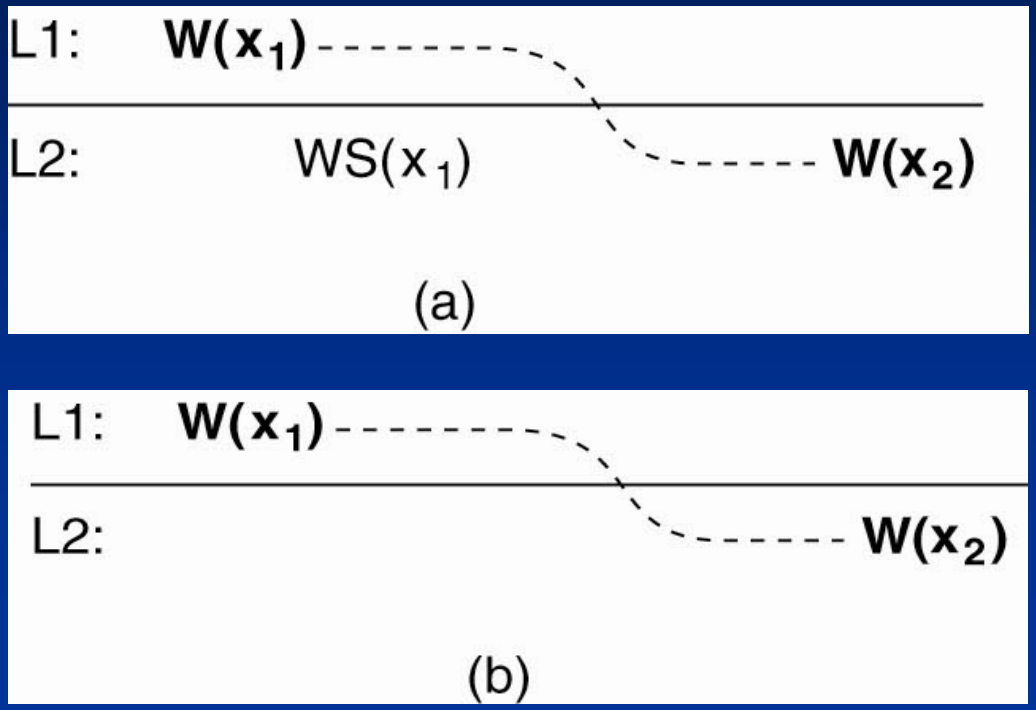
$WS(x_1;x_2)$表示$WS(x_1)$的操作已在L_2更新完毕

- 如果一个进程读取数据项x的值，那么该进程对x执行的任何后续读操作将总是得到第一次读取的那个值或更新的值
- 进程 *P* 对同一数据存储的两个不同本地备份执行的读操作

a)提供单调读一致性的数据存储

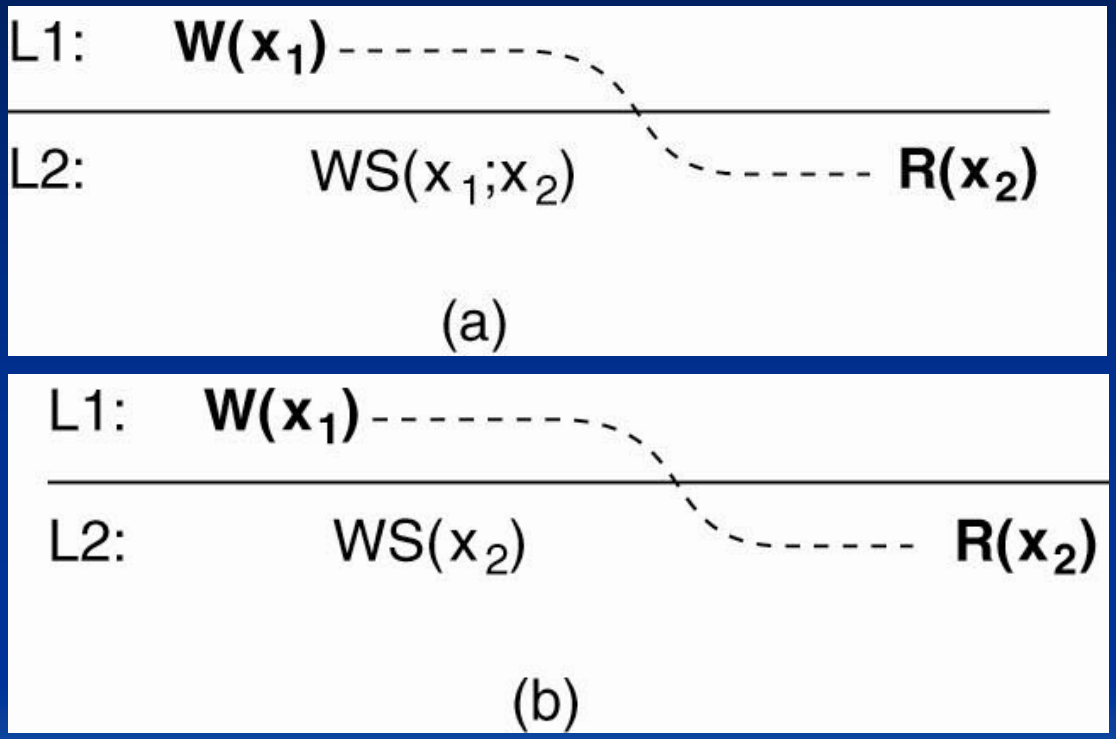
b)不提供单调读一致性的数据存储.

单调写（Monotonic Writes）



- 一个进程对数据项x执行的写操作必须在该进程对x执行的任何后续写操作之前完成
- 进程 *P* 对同一数据存储的两个不同本地备份执行的写操作
  - 提供单调写一致性的数据存储
  - 不提供单调写一致性的数据存储
- 类似以数据为中心的FIFO一致性

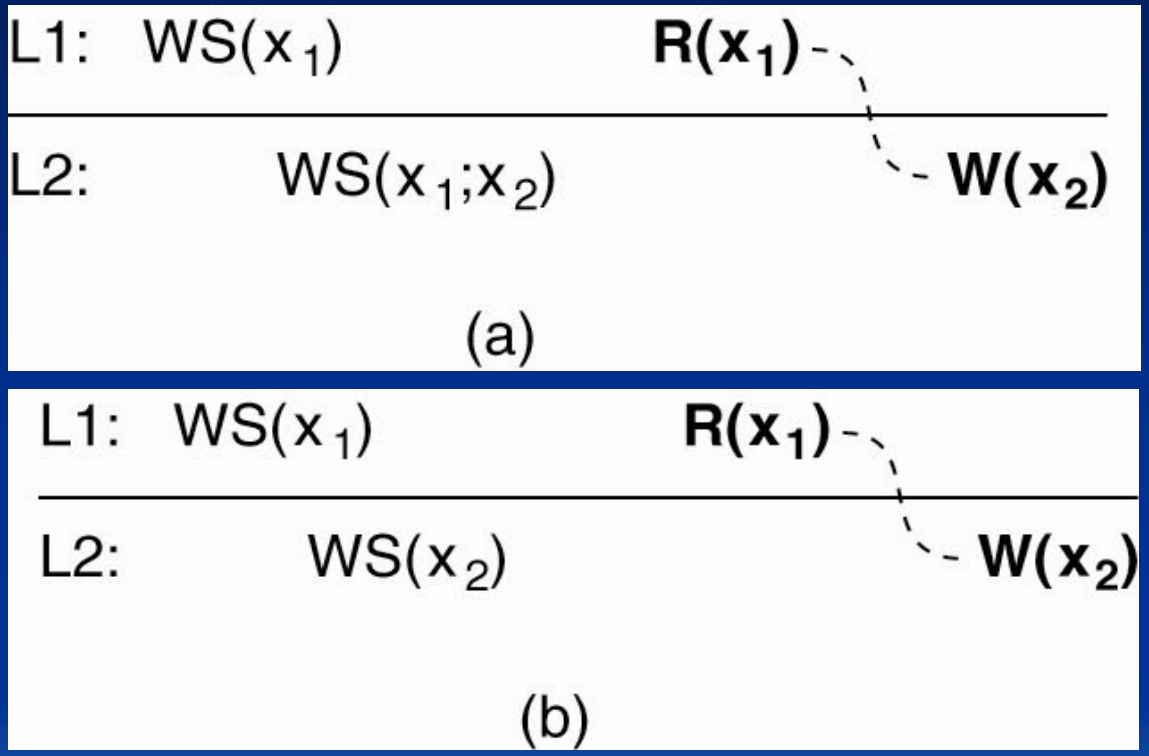
写后读（Read Your Writes）



一个进程对数据项x执行的写操作结果总会被该进程对x执行的任何后续读操作看见

- 提供写后读一致性的数据存储
- 不提供写后读一致性的数据存储

读后写(Writes Follow Reads)



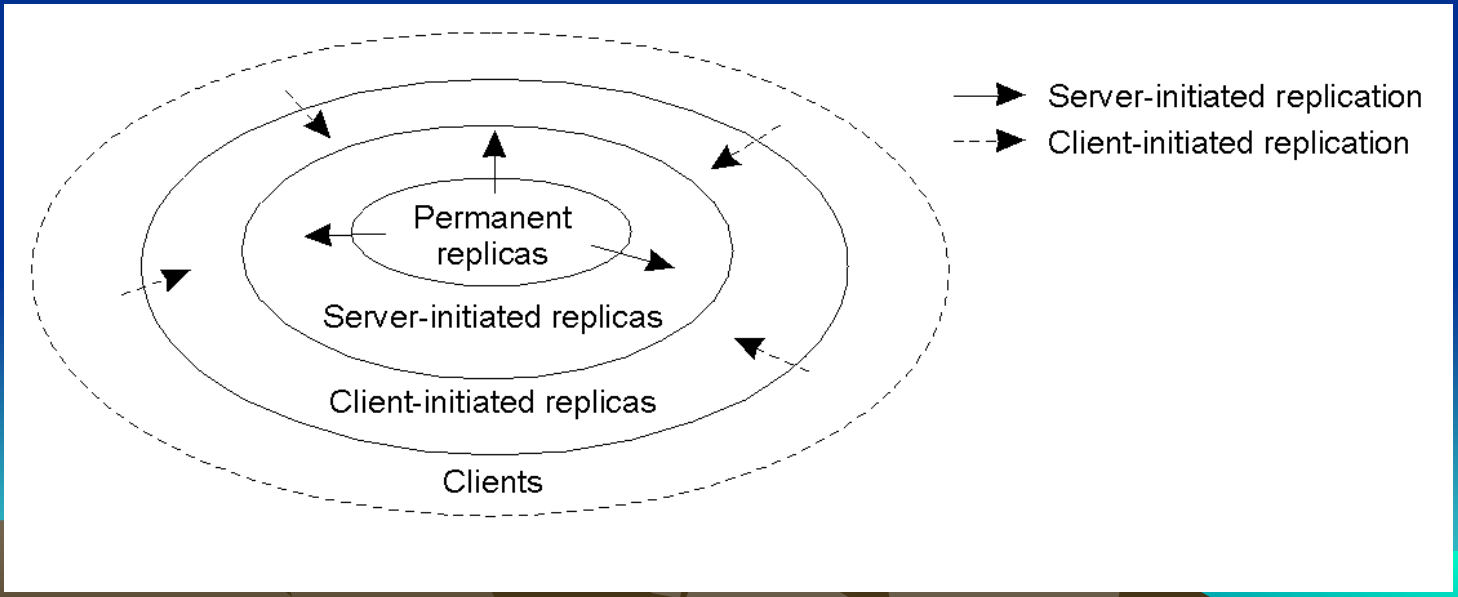
同一个进程对数据项x执行的读操作之后的写操作，保证发生在与x读取值相同或更新的值上

- 提供读后写一致性的数据存储
- 不提供读后写一致性的数据存储

复制管理

副本放置（Replica Placement）

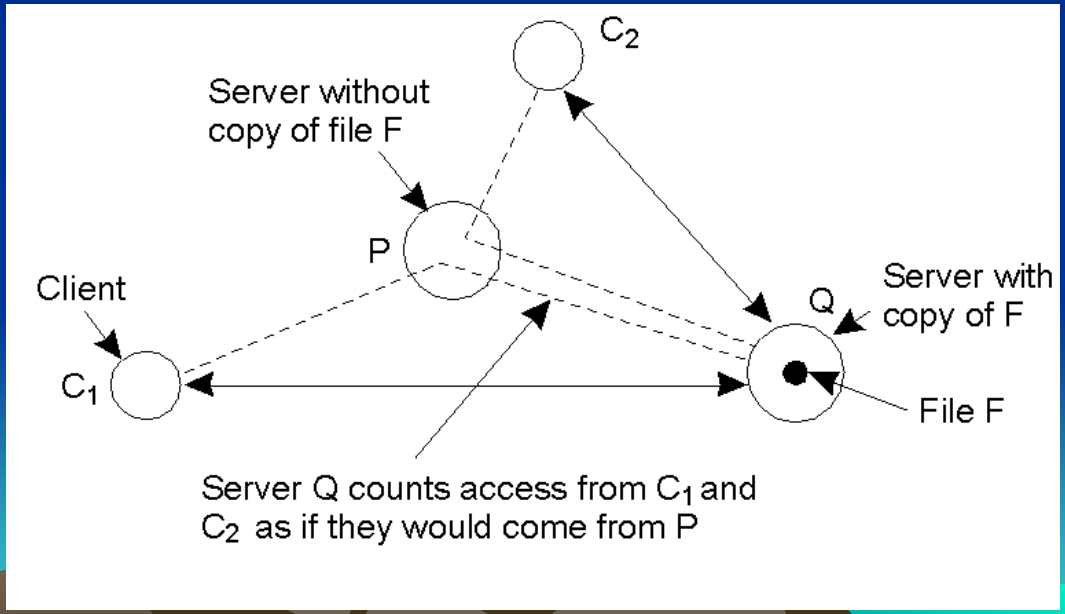
- 分发协议:将数据更新发送给各个副本的方法
- 副本放置：位置、时间和由谁来放置
  - 永久副本：副本的初始集合，数量很少，用作允许被修改以保证一致性的唯一副本
  - 服务器启动的副本:用于在客户附近放置只读备份，从而提高性能
  - 客户启动的副本：客户高速缓存，改善数据的访问时间；对只读数据高效；可以让多个客户共享缓存；效率取决于数据类型



数据存储的不同类型备份

服务器启动的副本
Server-Initiated Replicas

- 服务器启动的副本是为提高性能而存在的数据存储的备份
- 创建或删除副本的确切位置和时间:每台服务器跟踪每个文件的访问计数以及提出这些访问的请求
  - 删除阈值
  - 复制阈值
  - 请求的数量位于两者之间时，当来自P的对F的请求总量超过Q上对F的请求总量的一半时，转移数据到P



来自不同客户的访问请求计数

更新传播

更新传播：更新从一个拷贝传播到其他拷贝

相关设计问题：

- 状态与操作
- 拉协议与推协议
- 单播与多播

状态与操作

实际传播的信息

- 只传播更新的通知：无效化协议
  - 可以指定数据存储的哪些部分被更新了
  - 请求在无效的拷贝上操作时，先更新该拷贝
  - 几乎不占带宽，适合更新操作比读写操作多的场合（两次更新间可能没有读操作）
- 把数据从一个拷贝传送到另一个拷贝
  - 适合读对写的比率相对高的场合：被修改的数据可能在下一个更新前被读取，即更新是有效的可能性较高
  - 可以只传送修改的日志
  - 通过将多个修改压缩到一个消息里来减少通信开销
- 把更新操作传送到其他拷贝：主动复制
  - 假设每个副本由一个进程代表，该进程通过执行操作主动地更新数据：假设操作关联的参数少，传播更新的带宽代价小

拉协议与推协议
Pull versus Push Protocols

- 基于推式协议：基于服务器的协议
  - 永久副本和服务器启动的副本更新
  - 用于副本需要完全相同的时候
  - 适合读对写的比率相对高的场合（更新对读操作有效），高效
- 基于拉式协议：基于客户的协议
  - 一台服务器或客户请求其他服务器向它发送持有的更新
  - 用于客户高速缓存：Web高速缓存，客户轮询服务器

问题	基于推式	基于拉式
服务器的状态	客户副本和高速缓存的列表	无
发送的消息	更新（以及以后可能获取的更新）	轮询和更新
客户响应时间	立即（或获取更新的时间）	获取更新的时间

在多客户、单一服务器系统中，基于推式协议与基于拉式协议比较

基于租用的更新传播

- 更新传播的混合形式
- 租用是服务器的承诺，它将在指定的时间内将更新推给客户。
- 租用到期后：
  - 客户被迫轮询服务器以实现更新
  - 客户请求一个新的租期
- 三种类型的租用
  - 根据数据项的年龄：为预期保持不变的数据授予一个长期的租用，可以减少更新消息的数量
  - 根据客户请求高速缓存副本的频率：频率高的客户授予长期的租用，只跟踪对数据感兴趣的客户
  - 基于服务器的状态空间开销：将要过载时，缩短新租用的期限

单播与多播

- 多播：服务器向其他N台服务器发送更新时，底层的网络负责向多个接收者发送一个消息，高效
  - 与推式更新结合更高效
- 单播：服务器向其他N台服务器发送更新时，要发送N个单独的消息
  - 与拉式更新结合更高效：只有一个客户请求更新

Epidemic协议

- 提供最终一致性：保证所有的副本最终是一致的
- 不解决更新冲突，只关心使用最少的消息将更新传播到所有副本
- 一个服务器可以是：
  - 传染性的：持有愿意向其他服务器散布的更新
  - 易感的：尚未更新的服务器
  - 隔离的：已更新的服务器如果不愿意或不能扩散其更新
- 反熵传播模型：服务器P周期的随机选取一台服务器Q交换更新，方式包括：
  - P只把自己的更新推入Q：较差的选择
  - P只从Q拉出新的更新
  - P和Q相互发送更新
  - 可以证明：如果初始只有一台服务器具有传染性，无论采用哪种形式，更新最终将被传播到所有服务器上

gossiping

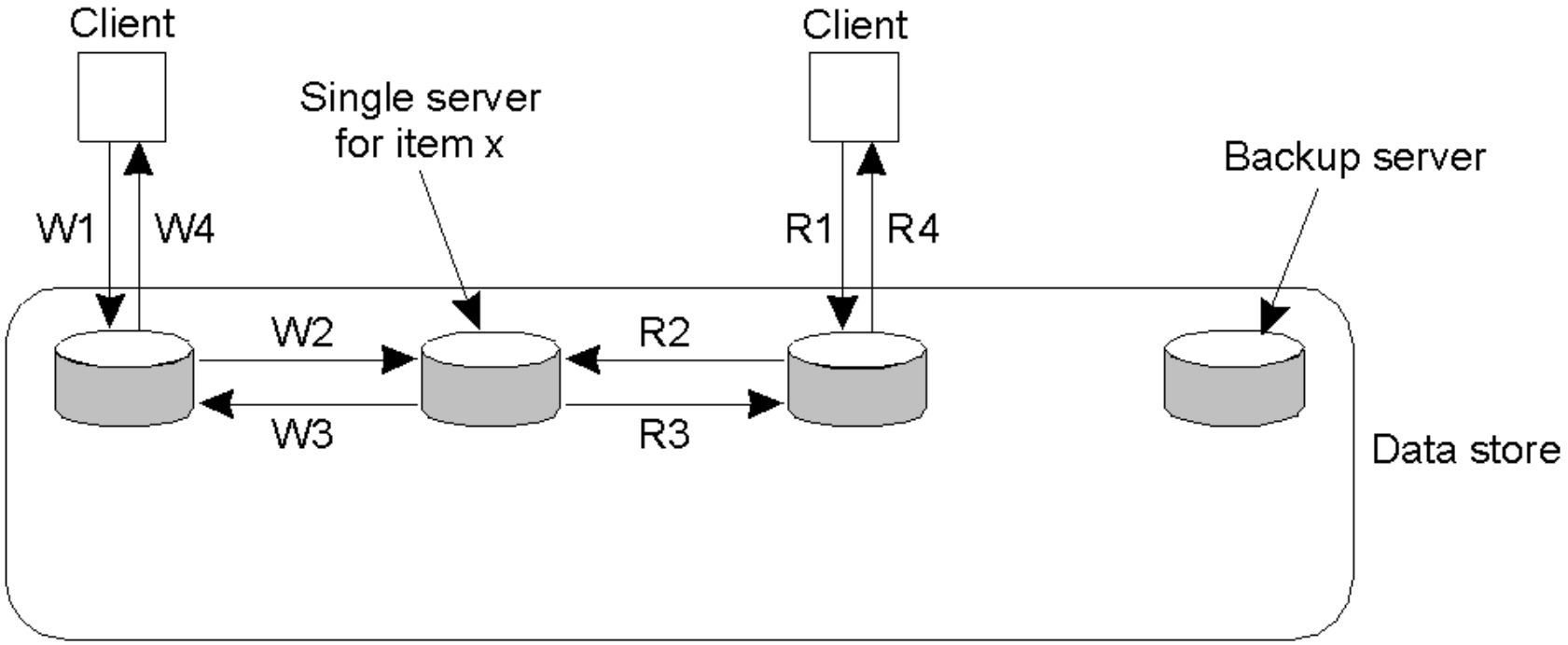
- 思想：
  - 如果服务器P刚刚因为数据项x而被更新，那么它联系任意一个其他服务器Q，并试图将更新推入Q。
  - 如果Q已经被其他服务器更新了， P可能会失去继续扩散的兴趣，变成隔离的（这种可能性是1/k）
- 快速传播更新的方法
- 但不能保证所有的服务器都被更新了
- $s=e^{-(k+1)(1-s)}$,k=3时， s小于2%
- 可以将gossiping和反熵结合

一致性协议

- 一致性协议描述特定的一致性模型的实现
- 基于主备份的协议：每个数据x都有一个关联的主备份，负责协调x的写操作，根据主备份的位置分为：
  - 远程写协议
  - 本地写协议
- 复制的写协议：写操作可以在多个副本上执行
  - 主动复制
  - 基于法定数目的协议

远程写协议
Remote-Write Protocols (1)

- 基于主备份的远程写协议，所有读操作和写操作都被转发到一个固定的服务器上

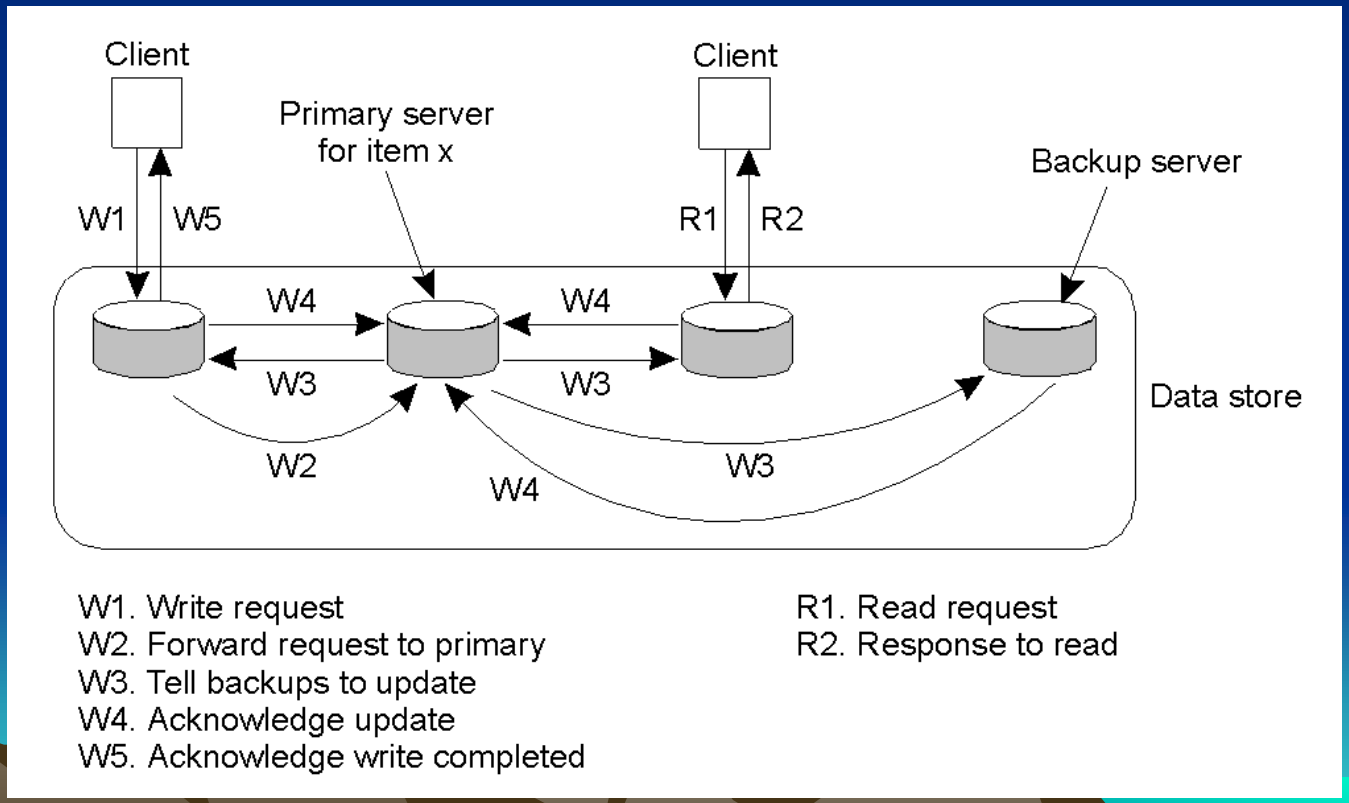


W1. Write request
W2. Forward request to server for x
W3. Acknowledge write completed
W4. Acknowledge write completed

R1. Read request
R2. Forward request to server for x
R3. Return response
R4. Return response

远程写协议
Remote-Write Protocols (2)

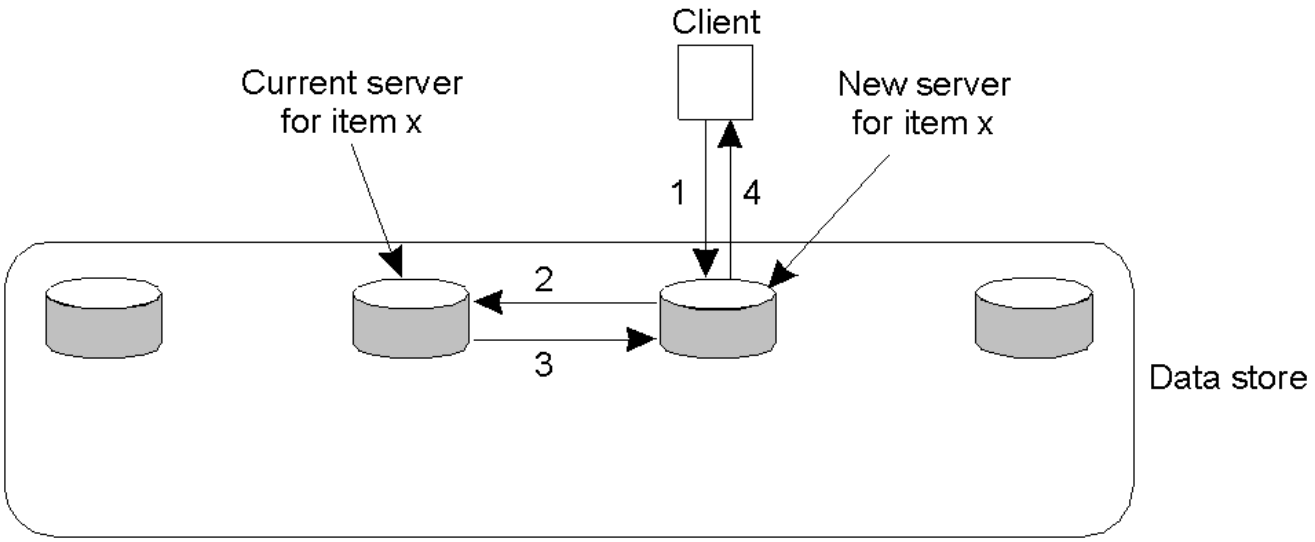
- 主机备份协议：允许进程在本地可用的副本上执行读操作，但必须向一个固定的主拷贝上转发写操作
- 提供实现顺序一致性的直接方法



主机备份协议的原理

本地写协议
Local-Write Protocols (1)

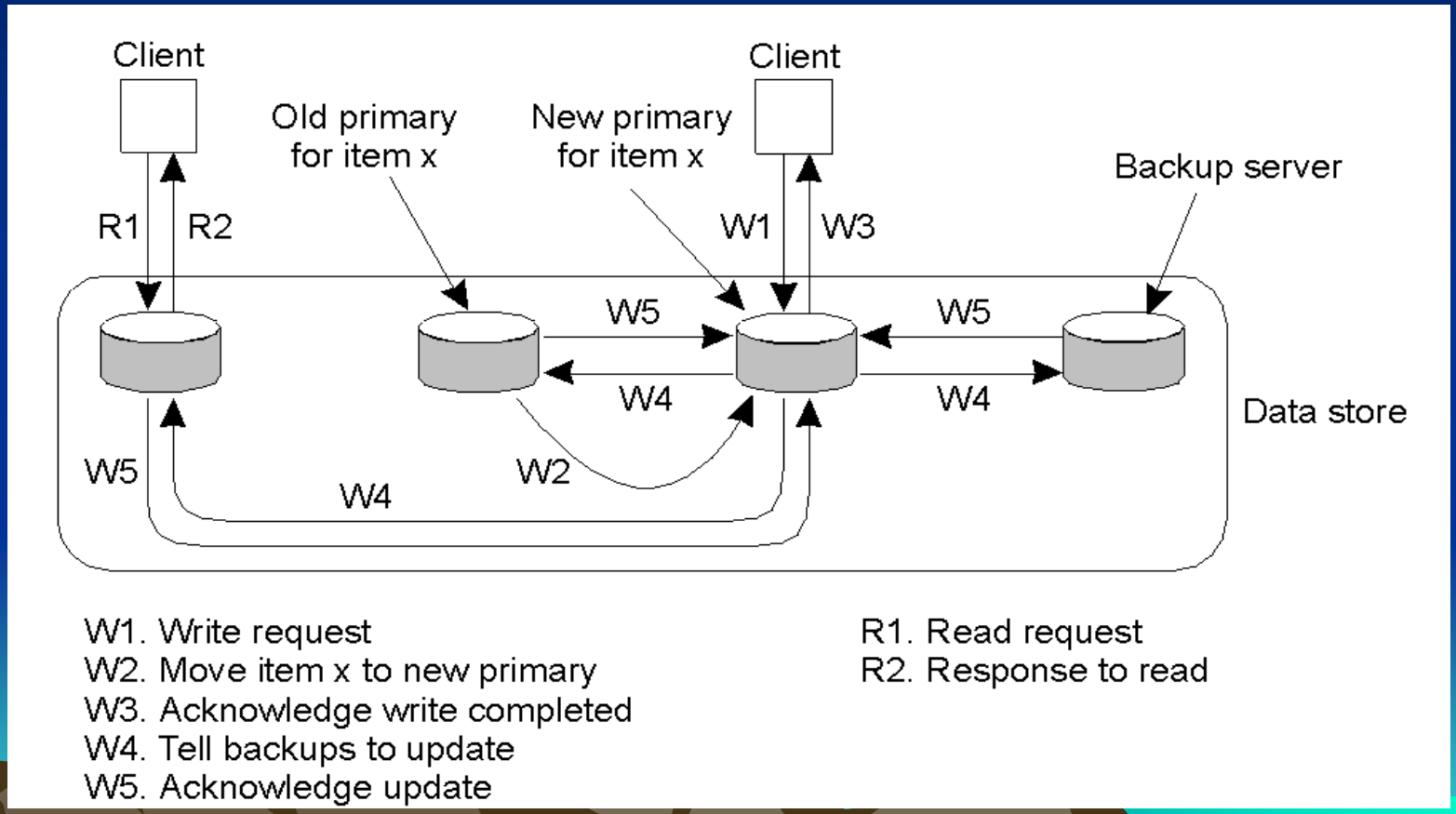
- 基于主备份的本地写协议，其中一个单一的拷贝在多个进程间移动
- 保证一致性
- 需要跟踪数据项的当前位置：广播、转发指针、基于原始位置的方法和层次定位服务



1. Read or write request
2. Forward request to current server for x
3. Move item x to client's server
4. Return result of operation on client's server

本地写协议(2)

- 多个连续的写操作可在本地进行，而读操作的进程还可以访问它们的本地拷贝
- 支持离线操作

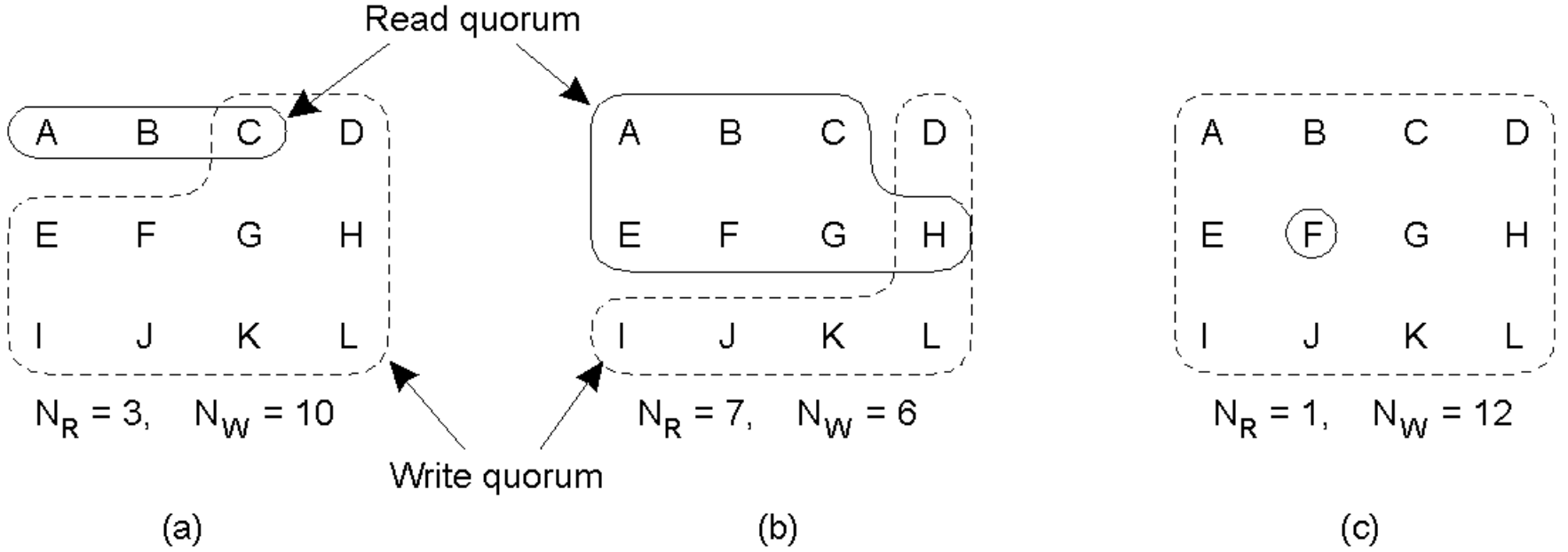


主机备份协议，其中主备份移动到要执行更新的进程那里

复制的写协议--主动复制
Active Replication (1)

- 写操作可以在多个副本上执行，每个副本对应一个进程，该进程执行更新操作；
- 写操作导致更新传播
- 操作需要在各地按相同顺序进行：
  - 时间戳
  - 定序器

基于法定数目的协议
Quorum-Based Protocols



Gifford方法：
客户在读写一个复制的数据时，先向多个服务器提出请求，获得许可；
读团体N_R和写团体N_W, N个副本
$N_R+N_W>N$; $N_W>N/2$
三个实例

a) 读集合和写集合的正确选择
b) 可能导致写写冲突的读集合和写集合
c) 读集合和写集合的正确选择，**ROWA (read one, write all)**

小 结

- 简介
- 以数据为中心的一致性模型
- 以客户为中心的一致性模型
- 分发协议
- 一致性协议

习 题

一个文件被复制在10个服务器上，列出基于法定数目的协议允许的所有读团体和写团体。