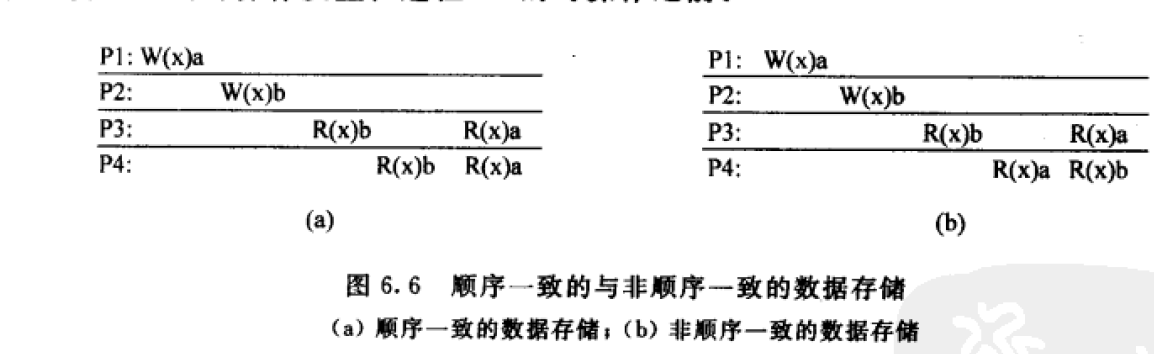
1. **一致性图的判断**

分布式系统的一个重要的问题是：数据的复制。复制的同时要保证各个副本的一致性。

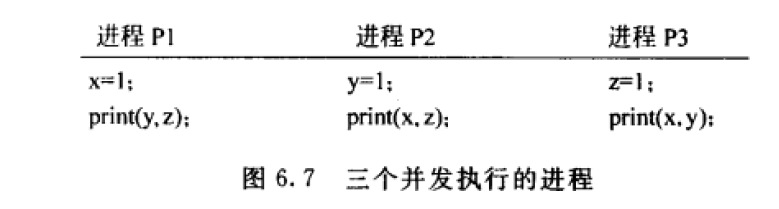
**1．严格的一致性也叫做强一致性**：需要保证数据在更新的同时，所有的副本也同样同时更新，但显然这是不可能实现的。

**2．顺序一致性：**



图（a）表示写的顺序和读的顺序是一致的，因为P3和P4最终读到的x的值都是a，而（b）图中，P3最终独到的是a，而P4最终读到的是b。

也就是说；

，对于这种样式的进程，每个进程的运行相对顺序首先保持不变，同时，对于每个变量，读写的值保持相对有序，不能有p1读到x值为1，随后p2读到x值为0的情况发生。

**3．弱于强一致性但强于顺序一致性的一致性模型---------线性化**

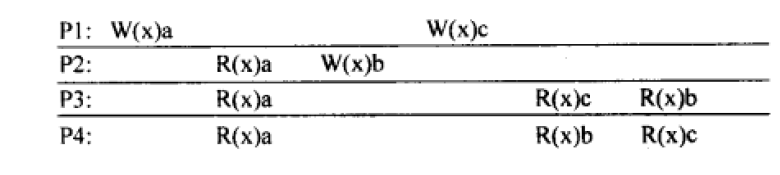
线性化与顺序一致性模型的区别在于：线性化给每个进程定义了一个全局的时间戳，根据时间戳来决定进程的执行顺序。

1. **因果一致性**





**潜在的因果关系：指的是当一个进程对某个变量进行写操作，另一个进程对该变量进行读操作时，进程间的执行顺序就具有潜在的因果关系，此时需要先写，再读，且这个顺序不可改变，但是如果两个进程都对这个变量进行写操作，则认为两个进程可以并发，谁先谁后都可以，同时，其他的进程可以看到不同的并发的写操作的顺序。如下图所示：**

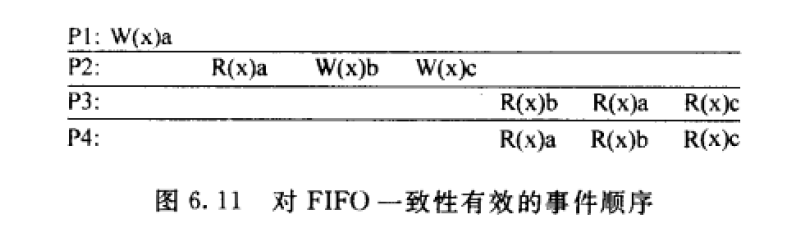


**该图w（x）b和w(x)c表示对x分别写入b和c，但是这两个顺序可以并发的，因此当p3和p4在读取x的值时，可以先读到w(x)c，也可以先读到w(x)b，因此该顺序满足因果一致性。但是对于顺序一致性来说，需要保证每个进程的读写顺序相对一致，即不能有p3最终读到的x为b，而p4读到的最终结果为c，因此不满足顺序一致性，更不满足强一致性。**

1. **FIFO一致性（PRAM一致性）**



**所有由不同进程进行的写操作都是并发的，例如下面的例子：**



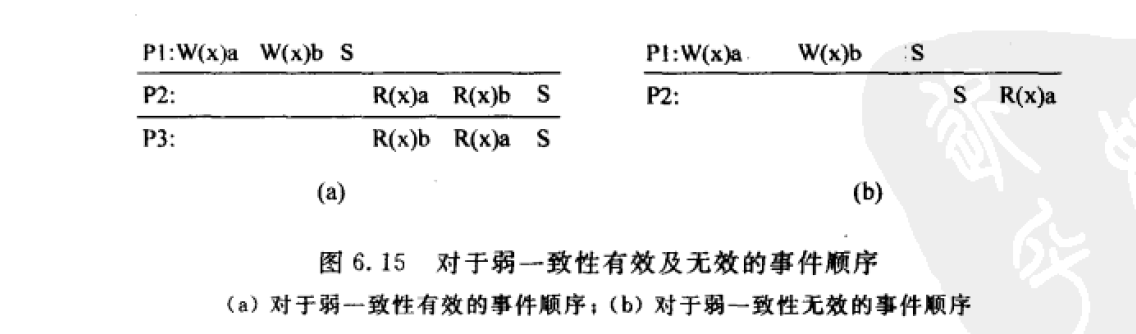
**对于因果一致性的条件：p2的R（x）a和p1的w（x）a是具有因果关系的，p2的w(x)b和w(x)c是同一个进程下的写操作，由于后续的读操作跟前面的写操作构成因果关系，所以需要读到最新的x的值，即要满足因果关系，只能等到w（x）c写完，但是FIFO一致性则不同，它把任何进程的写操作都看作并发，除非是一个进程内的多条连续写操作，这种情况下连续的写操作要保持有序，即不可能出现先读到c后读到b的情况，因此，p2的w(x)b，w(x)c与p1的w（x）a是并发的，所以才会出现p3和p4的情况。**

**对于p3来说，首先p2进行到第二步，然后p3读到R(x)b,然后p1，然后p3读到R(x)a,然后p2，然后p3读到R(x)c。**

1. **弱一致性**



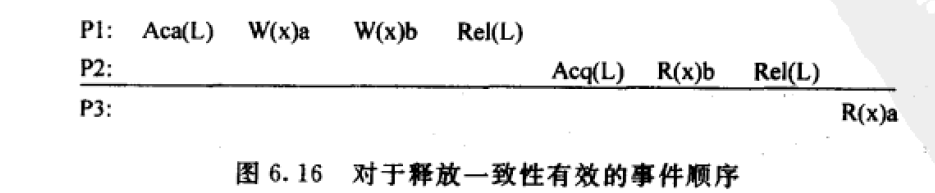




**对于a图，p1同步之后，p2由于没有同步，因此可能读到x为a，也可能为b，p3也是如此，但是对于b图来说，由于p2读取之前先进行了同步，因此p2读到的x的值只能是b，不可能是a。**

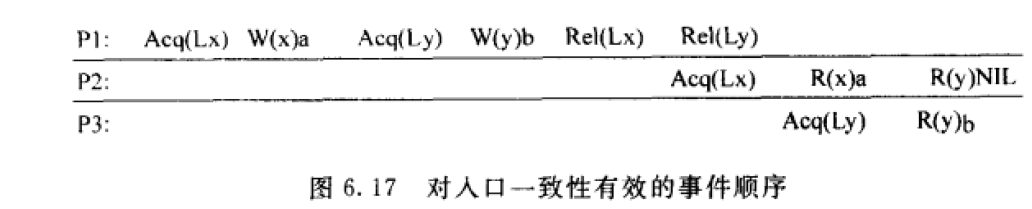
1. **释放一致性**

在弱一致性的基础上，分别出是进入临界区还是离开临界区，即acquire和release操作



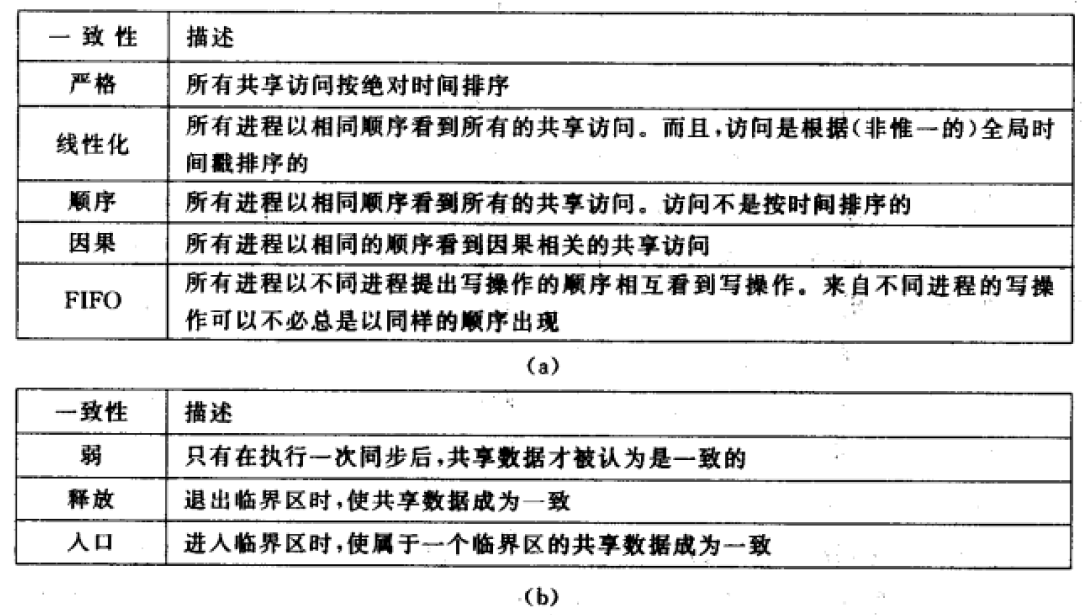
1. **入口一致性**





**入口一致性要求没对一个变量进行操作时，都需要获得对该变量的锁，然后再进行操作，而释放一致性则是根据经验判断那些变量需要上锁。**

**10．总结**



**以客户为中心的一致性模型：**

**单调读：**



**单调写：**



**写后读：**



**读后写：**



事务：

如果要对事务中需要进行的数据项的操作，一种方法是先对待操作的数据项进行锁定，处理完毕之后解锁。

具体：





摘 要: 本报告主要介绍了基于 RMI 机制的分布式时钟系统的设计与实现。该分布式时钟系统能够以客户机/服务器(C/S)的形式在多台机器上实时更新并显示当前的系统时间，显示方式分为数字模式或时钟模式。客户端可以以不同的时间间隔显示时间（运行时刻可以改变）。并利用基于观察者模式(Observer Pattern)的设计模式消除了客户端与服务器端在开发时和运行时的紧耦合性。允许客户端以不同的兴趣注册不同的事件。

关键字: RMI；分布式系统；观察者模式；时钟系统