http://www.chinacloud.cn/show.aspx?id=24905&cid=6

事务简介

事务的核心是锁和并发，采用同步控制的方式保证并发的情况下性能尽可能高，且容易理解。这种方式的优势是方便理解;它的劣势是性能比较低。

计算机可以简单的理解为一个标准的打字机，尽管看起来计算机可以并行处理很多事情，但实际上每个CPU单位时间内只能做一件事，要么读取数据、要么计算数据、要么写入数据，所有的任务都可以看成这三件事的集合。计算机的这种特性引出了一个问题：当多个人去读、算、写操作时，如果不加访问控制，系统势必会产生冲突。而事务相当于在读、算、写操作之外增加了同步的模块，进而保证只有一个线程进入事务当中，而其他线程不会进入。

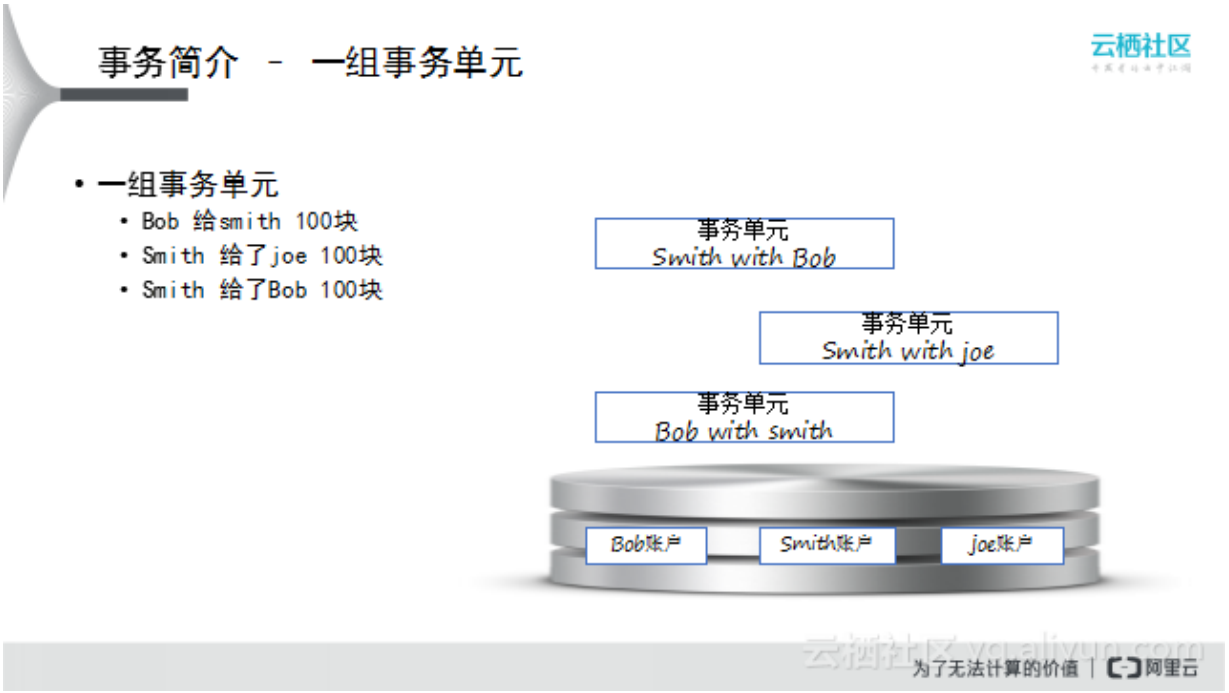
单个事务单元

事务的四大特性分别是：原子型、一致性、隔离性和持久性。其中原子性指的是事务中包含的所有操作要么全做，要么全不做;一致性是指在事务开始以前，数据库处于一致性的状态，事务结束后，数据库也必须处于一致性的状态;隔离性要求系统必须保证事务不受其他并发执行的事务的影响;持久性是指一个事务一旦成功完成，它对数据库的改变必须是永久的，即使是在系统遇到故障的情况下也不会丢失，数据的重要性决定了事务的持久性的重要。

dad1641a2e64880d7ac062eadf9d6adbc6d68308

事务单元是通过Begin-Traction，然后Commit(Begin-Traction、Commit和Rollback之间所有针对数据的写入、读取的操作都应该添加同步访问)，Begin和Commit之间就是一个同步的事务单元。例如，Bob给Smith 100块钱就是一个事务单元，这个过程中有很多步操作，具体如上图所示;但对业务来说，仅是一个转账的操作。

一组事务单元



ad8567bbdabd094fc2bc67faf555592a0cb85364

当三个账户都在进行转账操作时，每个操作都涉及Smith账户，所有的事务都会排队，形成一组事务单元。

事务单元之间的Happen-Before关系中的四种可能性：读写、写读、读读、写写。所有事务之间的关系都可以抽象成这四种之一，来对应现在所有的业务逻辑处理。在此基础之上，需要用最快的速度处理多个事务单元之间的关系，同时还能保障这四种操作的逻辑顺序。

单个事务单元的其他例子

除了转账操作是事务单元外，诸如商品要建立一个基于GMT_Modified的索引、从数据库中读取一行记录、向数据库中写入一行记录，同时更新这行记录的所有索引、删除整张表等都是一个事务单元。

事务单元的实现方式

774487d00e2809953adac13735f55a4371cf664a

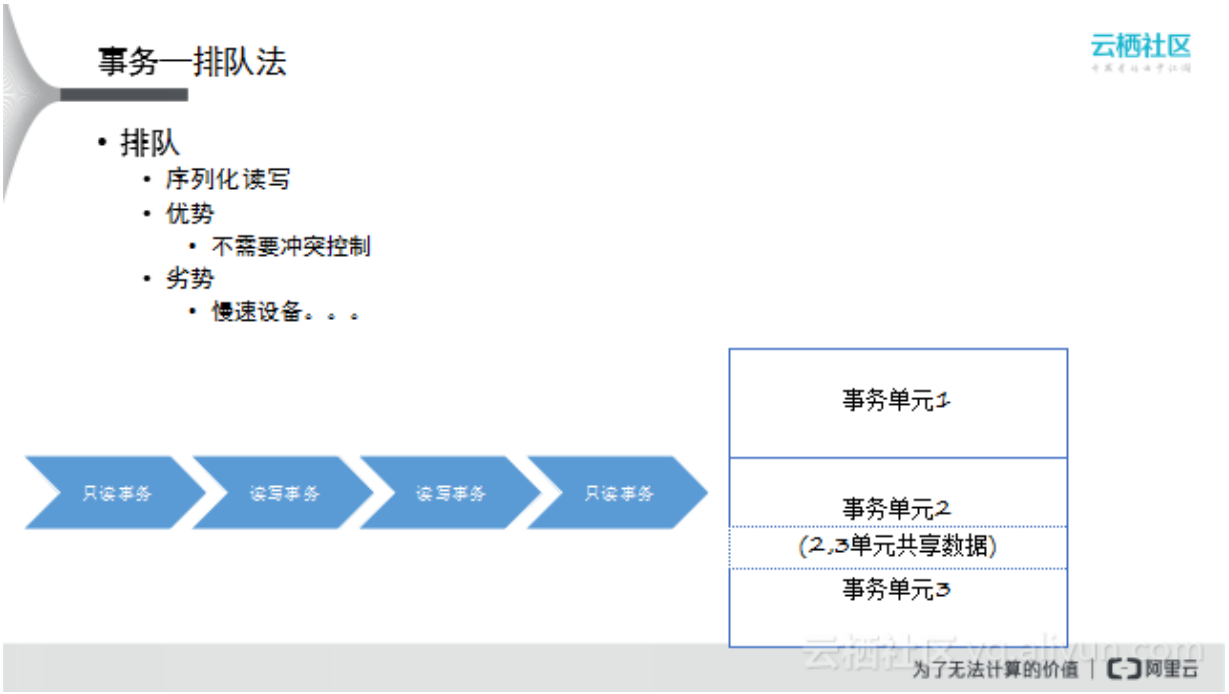
Two Phase Lock(2PL)是数据库中非常重要的一个概念。数据库操作Insert、Update、Delete都是先读再写的操作，例如Insert操作是先读取数据，读取之后判读数据是否存在，如果不存在，则写入该数据，如果数据存在，则返回错误。假设在该场景下没有读操作，只是单纯写入数据，则数据本身并没有事务操作，Delete、Update操作与之类似。数据库利用这些操作的特性，在每一次查询过程中，只要查到数据，就会在该数据上加锁。理论上，所有被读取的数据都已加锁，不会再被其他人读到，也就是说对数据进行的中间操作状态对所有人都不可见，当所有中间状态完成后，提交操作时，解开锁，此时数据对所有系统可见，例如在转账过程中，所有人只能看到两种状态：开始时，A有钱，B没钱;结束时，B有钱，A没钱，而中间A减掉钱，B尚未加上钱的状态被锁隐藏掉了，这个操作就是数据库中处理事务的最标准的方式。如上图所示：事务中的Trx2(JoeLock)与其他事务不相关，因此可以并行执行;Trx1需要Lock两个数据Boblock和Smithlock，而Trx3同样需要Lock这两个数据，因此Trx3必须等待，

且等待在Boblock上;Joe事务会先结束，Trx3会等到Trx1完成后才会开始。

处理事务的常见方法

处理事务的常见方法有排队法、排他锁、读写锁、MVCC等方式，下面来一一解析。

排队法



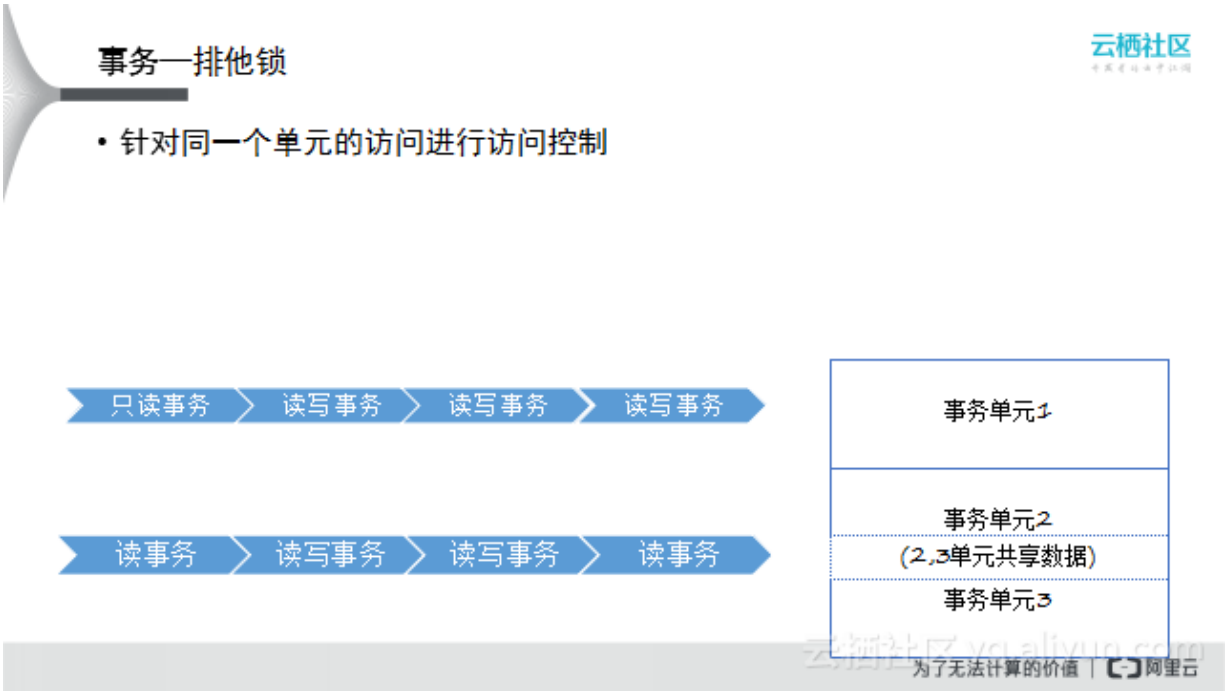
25b87cf81c9a3f036435259ab0e30f91ec2b0191

事务处理中最重要也是最简单的方案是排队法，单线程地处理一堆数据。在Redis中，如果数据全部在内存中，则单线程处理所有Put、Get操作效率最高。这是因为多线程本质是CPU模拟多个线程，这种模拟是以上下文切换为代价，而对于内存的数据库来说，没有上下文切换时效率最高。因此，单个CPU绑定一块内存的数据，针对这块数据做多次读写操作时都是在单个CPU上完成的，单线程处理方式在内存的情况是效率是最优的。

那么什么时候事务需要用到多线程呢?这个问题的本质取决于下层所使用的存储，如果是内存操作，则可以动态地申请和销毁内存块;而

磁盘的IOPS很低，但吞吐量很高。如果一个场景涉及多次读写操作，单线程可以很高的效率对于内存进行读写操作;但是，由于磁盘的IOPS仅为内存的几千分之一，如果依旧用操作内存的方式操作磁盘，那系统的整体性能将会很低，这意味着必须将大量的读写操作聚合成一个Batch后再提交时才能达到较好的性能。而将大量请求攒到一起的方式一是异步，也就是请求本身和线程不绑定，线程可以不Block(本质来说还是一种多线程的方式)，处理完一个线程后再处理其他线程。这种做法的核心是将大量不同的请求提交到一个Buffer中，再由该Buffer统一读取或者写入磁盘，从而提高效率。在慢速设备中，多线程或异步非常常见，在设计系统时，面对磁盘、网络、SSD等慢速设备必须考虑使用多线程。

排他锁

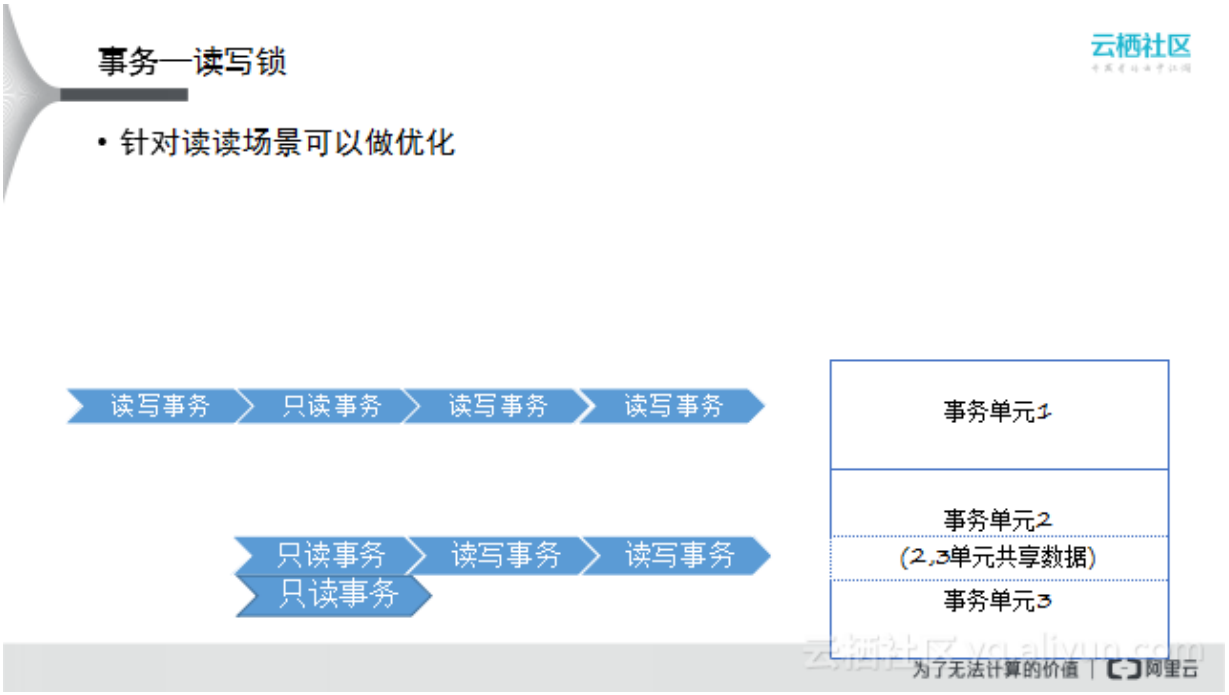


f6565989422f38604d13e0910e09a88fc1cddda2

有些场景不适合用单线程操作，可以利用排他锁的方式来快速隔离并发读写事务。数据库中有一些事务单元是共享的，如图中的事务单元1是共享的，事务单元2/3共享数据;针对事务单元2/3共享数据的

所有读写Block住，事务单元1单独用一个锁来控制，用这种方式完成系统的访问控制。

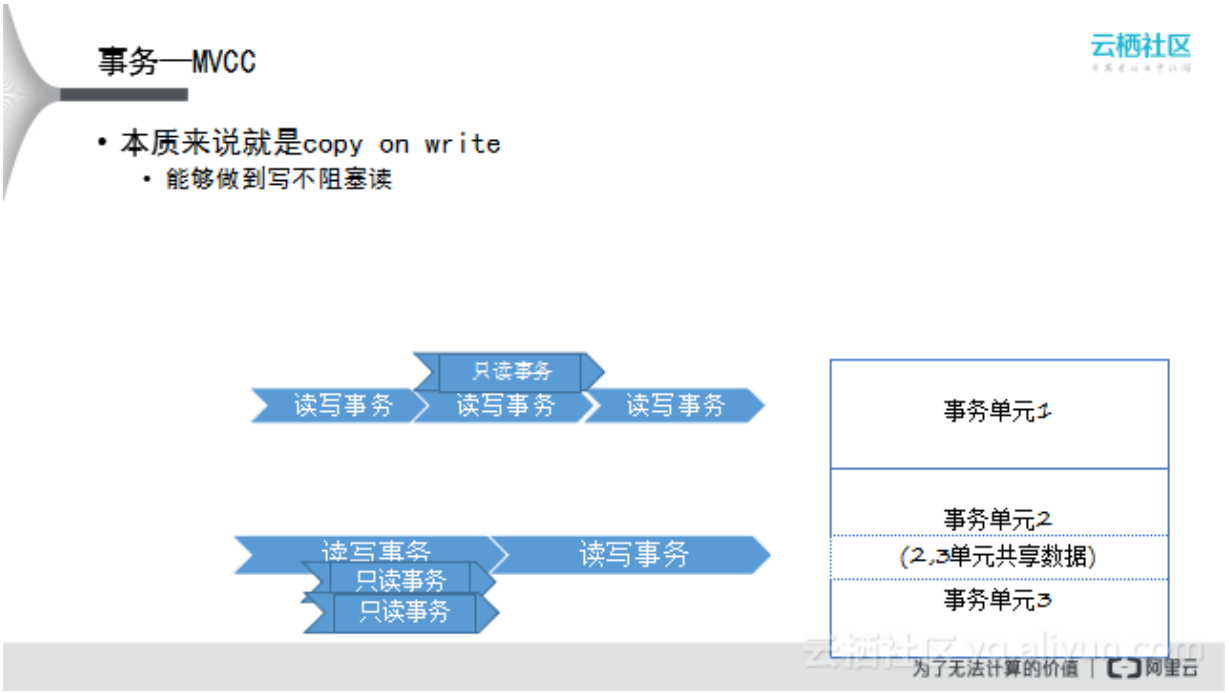
读写锁



46769c09018c7ac199e7520dd5e4735bdf39458c

如果是一个只读的事务，例如只对数据进行查询操作，在该过程中数据一定不被修改，因此多个查询操作可以并行执行，因此一种针对读读场景的优化自然而然产生——读写锁。读写锁的核心是在多次读的操作中，同时允许多个读者来访问共享资源，提高并发性。

MVCC



a1251eb42b581c02f5252a8c6b35ec3c8c969a8d

在最初的数据库事务实现中是不存在MVCC的，它是Oracle在八十年代新加的功能，本质是Copy On Write，也就是每次写都是以重新开始一个新的版本的方式写入数据，因此，数据库中也就包含了之前的所有版本。在数据读的过程中，先申请一个版本号，如果该版本号小于正在写入的版本号，则数据一定可以查询到，无需等到新版本完全写完即可返回查询结果。这种方式可以在读读不阻塞的前提下，实现读写/写读不阻塞，尽可能保证所有的读操作并行，而写操作串行。

事务的调优原则

事务的调优的思路是在不影响业务应用的前提下：

第一，尽可能减少锁的覆盖范围，例如Myisam表锁到Innodb的行锁就是一个减少锁覆盖范围的过程;对于原位锁(排他锁、读写锁等)可变为MVCC多版本(本质仍然是减少锁的范围)。

第二，增加锁上可并行的线程数，例如读锁和写锁的分离，允许并行读取数据。

第三，选择正确锁类型，其中悲观锁适合并发争抢比较严重的场景;乐观锁适合并发争抢不太严重的场景。