再谈 COW、ROW 快照技术

2017-06-30 22:38 云物互联 阅读(2665) 评论(1) 编辑 收藏 举报

目录

前言

在经过了一段时间的实践之后，再次回顾 COW/ROW 快照技术的实现原理，温故而知新。

快照与备份的区别

传统地，人们一直采用数据复制、备份、恢复等技术来保护重要的数据信息，定期对数据进行备份或复制。由于数据备份过程会影响应用性能，并且非常耗时，因此数据备份通常被安排在系统负载较轻时进行(如夜间)。另外，为了节省存储空间，通常结合全量和增量备份技术。显然，这种数据备份方式存在一个显著的不足，即备份窗口问题。**在数据备份期间，企业业务需要暂时停止对外提供服务。随着企业数据量和数据增长速度的加快，这个窗口可能会要求越来越长，这对于关键性业务系统来说是无法接受的。**诸如银行、电信等机构，信息系统要求 24*7 不间断运行，短时的停机或者少量数据的丢失都会导致巨大的损失。因此，就需要将数据备份窗口尽可能地缩小，甚至缩小为零。数据快照(Snapshot)、持续数据保护(CDP, Continuous Data Protection)等技术，就是为了满足这样的需求而出现的数据保护技术。需要注意的是：目前，随着对信息系统的依赖程度越来越高，即使不是银行、电信这类传统关键行业，在政府、教育、企业也越来越多的系统要求更小的备份窗口和更短的停机时间。**「降低数据保护的代价，提高数据保护过程中的应用感知能力，逐步成为客户的核心诉求」**。

快照技术

快照的定义：关于指定数据集合的一个完全可用拷贝，该拷贝包括相应数据在某个时间点（拷贝开始的时间点）的映像。快照可以是其所表示的数据的一个副本，也可以是数据的一个复制品。从更具体的技术细节来讲，快照是指向保存在存储设备中的数据的引用标记或指针。我们可以这样理解，快照有点像是详细的数据地址目录表，但在计算机中快照被作为完整的数据备份来对待。

快照技术的作用：主要能够进行在线数据恢复，当存储设备发生故障或损坏时能够进行即时的数据恢复，将存储状态恢复到快照时间点的状态。另一个作用是能够为存储用户提供另外一个数据访问的通道，当源数据进行在线应用处理时，用户可以选择访问快照数据，还能够将快照应用到测试等工作。因此，所有存储系统，不论高中低端，只要应用于在线系统，那么快照就成为一个不可或缺的功能。快照在备份、数据保护过程中发挥着越来越大的作用。

快照的优势:

- 快照可以在数秒钟内建立拷贝, 供备份应用使用. 利用快照技术, 配合普通的备份软件是这样实现的:
  - 通过图形的管理界面发出做快照的命令
  - 快照功能自动寻找没有数据改变的时刻进行拷贝，几秒钟之后拷贝生成
    -再使用备份软件对该拷贝进行备份
- 利用快照的镜像可以在数秒钟内把数据恢复到做快照的时间点, 还允许系统管理员选择性地迅速恢复受损或被删文件
- 数据快照的功能还有很多用处, 比如现在需要一份最新的生产数据来做新系统的测试或者提供决策支持和数据分析所用, 而系统又不能停机, 使用磁带备份恢复 一份数据时间又很长. 这样的情况可以利用数据快照的备份功能在任一时间点建立快照拷贝, 利用拷贝的数据进行测试和分析, 不会影响系统的正常使用

快照的分类：

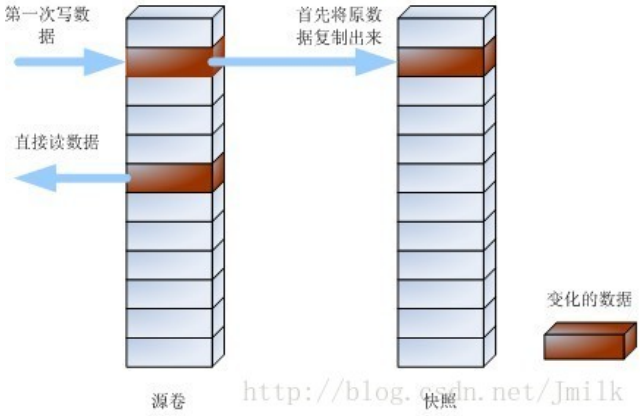
- 全量快照:
  - 镜像分离 (Split Mirror)
- 增量快照
  - 写时拷贝 (Copy-On-Write)

- 写时重定向 (Redirect-On-Write)

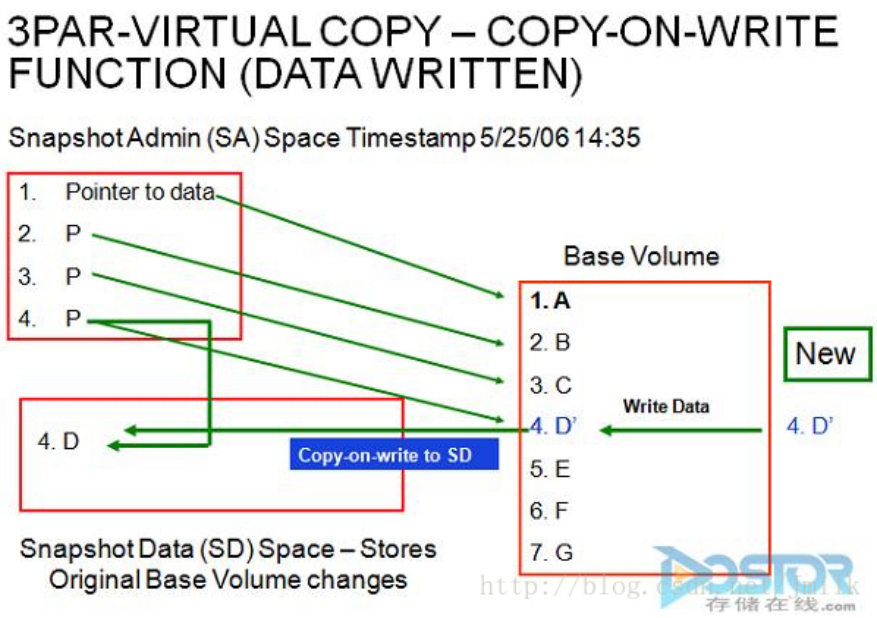
增量快照之 COW

COW(Copy-On-Write)，也被称之为「即写即拷」快照技术或「写时复制」快照技术，这种方式通常也被称为“**元数据(源数据指针表)**”拷贝。顾名思义，如果有人试图改写源数据块上的原始数据，首先将原始数据拷贝到新数据块中，然后再进行改写。当你还原快照需要引用原始数据时，快照软件将原始数据原有的指针映射到新数据块上。

再来深入的看看 COW 的过程，COW 在创建快照时，并不会发生物理的数据拷贝动作，仅是拷贝了原始数据所在的源数据块的物理位置元数据。因此，COW 快照创建非常快，可以瞬间完成。在创建了快照之后，快照软件会监控跟踪原始数据的变化(即对源数据块的写操作)，一旦源数据块中的原始数据被改写，则会将源数据块上的数据拷贝到新数据块中，然后将新数据写入到源数据块中覆盖原始数据。其中所有的源数据块就组成了所谓的源数据卷，而新数据块组成了快照卷。你应该能够看出 COW 有一个很明显的缺点，就是会降低源数据卷的写性能，因为每次改写新数据，实际上都进行了两次写操作。



再再深入的看看 COW 的原理，在创建快照时，会同时创建快照卷，但只需分配相对少量的存储空间，用于保存创建快照后源数据卷中被更新的数据。每个源数据卷都具有一张数据指针表(元数据)，简称源数据指针表，表记录就是指向相应源数据块的地址指针。在创建快照时，存储子系统会建立源数据指针表的一个副本(元数据拷贝)，作为快照卷的数据指针表，简称快照数据指针表。所以，在创建快照之后，这个快照就相当于一个可供上层应用访问的存储逻辑副本，快照卷与源数据卷通过各自的指针表共享同一份物理数据。当源数据卷中任意数据将要被改写时，COW 需要确保对原始数据的拷贝操作发生在原始数据的改写操作之前，并且将原始数据在快照卷中的新地址更新到快照数据指针表记录中，使快照时间点后更新的数据不会出现在快照卷中，快照卷中的数据都必须是快照时间点那一刻的数据，以此保证了快照数据的完整性。



1) 源数据卷包含了原始数据 A~G，源数据指针表的记录在创建快照时均指向原始数据 A~G。
2) 创建快照时，即确立了快照时间点，快照软件会创建快照卷，并且 Copy 源数据指针表为快照卷指针表，此时快照指针表记录同样指向原始数据 A~G。
3) 创建快照之后，首次有新数据更新到源数据卷中，如：新数据 D’ 更新原始数据 D。此时在 D’ 覆盖 D 之前，需要先把 D 拷贝到快照卷中，并且更新快照数据指针表的记录使其指向新的存储地址。最后再将 D’ 写入到 D 原来的位置，源数据指针表不需要更新。

NOTE1：在步骤 3) 中使用了「首次」一词，意思是说当源数据卷中同一位置上的数据被修改了多次也仅仅会在第一次修改时被拷贝，换句话说就是只有原始数据被更新时才会触发拷贝操作，新数据被更新的数据更新并不会影响到快照数据的完整性。所以 COW 偶尔也会被表述为 COFW(Copy-On-First-Write)

NOTE2：源数据指针表至此至终都不会发生变化，所以 COW 对源数据卷的读操作和对源数据卷中单个位置的多次写操作性能都不会有很大的影响。相对的，快照卷数据是非连续的，而且在执行多次快照操作之后，数据会变得非常离散，所以快照卷数据的读写延时较大。

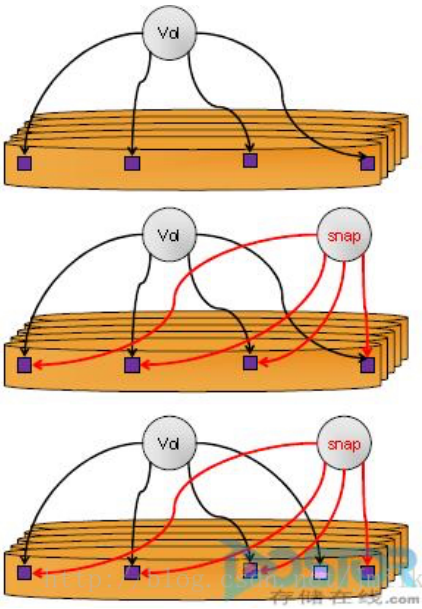
应用场景：这种实现方式在第一次写入某个存储位置时需要完成一个读操作（读原位置的数据），两个写操作（写原位置与写快照空间），如果写入频繁，那么这种方式将非常 消耗IO时间。因此可推断，如果预计某个卷上的I/O多数以读操作为主，写操作较少的场景，这种方式的快照实现技术是一个较理想的选择，因为快照的完成需要较少的时间。除此之外，如果一个应用易出现写入热点，即只针对某个有限范围内的数据进行写操作，那么COW的快照实现方式也是较较理想的选择。因为其数据更改都局限在一个范围内，对同一份数据的多次写操作只会出现一次写时复制操作。但是这种方式的缺点也是非常明显的。如果写操作过于分散且频繁，那么 COW造成的开销则是不可忽略的，有时甚至是无法接受的。因此在应用时，则需要综合评估应用系统的使用场景，以判断这种方式的快照是否适用。

在了解了 COW 的实现原理之后再回头对比一下 COW 与备份之间的区别，COW 技术在创建快照前，并不会占用任何的存储资源，也不会影响系统性能。而且 COW 在使用上非常灵活，能够在任意时间点为任意数据卷创建快照。在快照时间点产生的“备份窗口”的长度与源数据卷的容量成线性比例，一般为几秒钟，对应用影响甚微，并且为快照卷分配的存储空间也大大的减少。拷贝的操作只在源数据卷发生更新时才被触发，因此系统开销很小。但是由于快照卷仅保存了源数据卷被更新的数据，因此快照技术并不能够得到数据的完整物理副本。

增量快照之 row

ROW(Redirect-On-Write)，也被称之为写时重定向。ROW 的实现原理与 COW 非常相似，区别在于「ROW 对原始数据卷的首次写操作，会将新数据重定向到预留的快照卷中」，而非 COW 一般会使用新数据将原始数据覆盖。所以，ROW 快照中的原始数据依旧保留在源数据卷中，并且为了保证快照数据的完整性，在创建快照时，源数据卷状态会由读写变成只读的。如果对一个虚拟机做了多次快照，就产生了一个快照链，虚拟机的磁盘卷始终挂载在快照链的最末端，即虚拟机的写操作全都会落盘到最末端的快照卷中。该特征导致了一个问题，就是如果一共做了 10 次快照，那么在恢复到最新的快照点时，则需要通过合并 10 个快照卷来得到一个完整的最新快照点数据；如果是恢复到第 8 次快找时间点，那么就需要将前 8 次的快照卷合并成为一个完整的快照点数据。从这里可以看出 ROW 的主要缺点是没有一个完整的快照卷，其快照之间的关系是链式的，如果快照层级越多，进行快照恢复时的系统开销会比较大。但 ROW 的优势在于其解决了 COW 快照写两次的问题，所以就写性能而言，ROW 无疑是优于 COW 的。

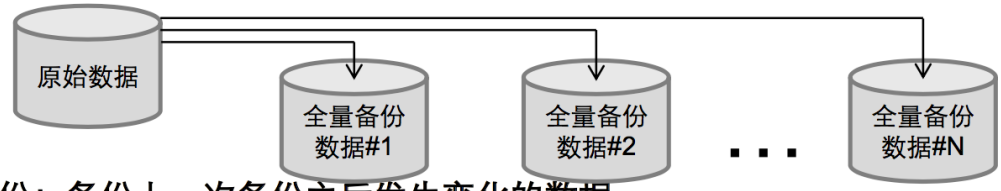
再深入的来看看 ROW 的原理，创建快照时，ROW 也会 Copy 一份源数据指针表作为快照数据指针表，此时两张表的指针记录都相同的。在创建快照之后，也就是在快照时间点之后，发生了写操作，那么新数据会直接被写入到快照卷中，然后再更新源数据指针表的记录，使其指向新数据所在的快照卷地址。可以看出，ROW 与 COW 最大的不同就是：**COW 的快照卷存放的是原始数据，而 ROW 的快照卷存放的是新数据。**因为 ROW 这种设定，所以其多个快照之间的关系必定是链式的，因为最新一次快照的原始数据很可能就存放在了上一次快照时创建的快照卷中。



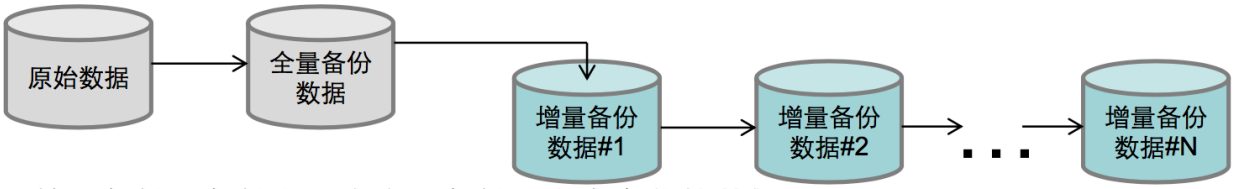
1) 上图中，Vd 是源数据卷的源数据指针表，分别指向 4 个原始数据。
2) 创建快照时，会 Copy 源数据指针表作为快照卷的快照数据指针表 snap，也同样指向 4 个原始数据，并且会分配存储空间作为快照卷。
3) 创建快照后，当有新数据写入时，直接将新数据写入快照卷中，然后修改 Vd 中的记录指向新数据。而 snap 的记录不变。

值得注意的是：ROW 在传统存储场景下最大的问题是对读性能影响比较大。的确，ROW 的写性能基本没有损耗，只是修改指针，实现效率很高。但多次读写操作后，某时刻的源数据卷的数据会变得非常离散(源数据指针表记录都被更新了)，这是 ROW 的连续读写性能就不如 COW 了。所以，ROW 更适合应用到 Write-Intensive(写密集型)的存储系统中。但是，但是，但是，在分布式存储的情况下，ROW 的连续读写的性能却会比 COW 更高。传统存储场景中读写性能的瓶颈一般是在磁盘上，但这种瓶颈在分布式存储场景中是不存在的。用户在业务层看到连续存储，实际上是分布在不同的服务器的不同硬盘中，数据越是分散，系统性能越高。而 ROW 把源数据卷中的原始数据打散之后，对性能反而有好处。**所以现阶段而言，ROW + 分布式存储的快照方式是业界发展的主要方向。**

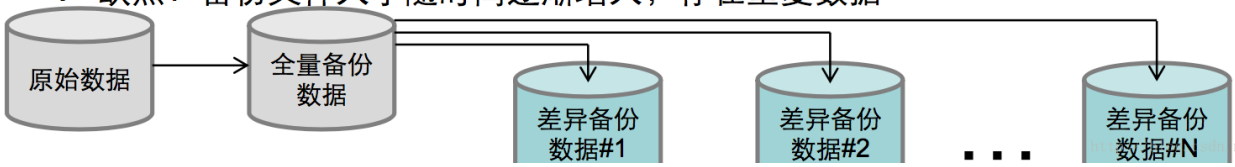
- **全量备份：备份所有数据**
  - → 优点：数据完整，提供最好的数据保护
  - → 缺点：备份数据量大，会造成磁盘的浪费



- **增量备份：备份上一次备份之后发生变化的数据**
  - → 优点：需要备份的数据量小，备份速度快
  - → 缺点：数据恢复复杂，必须具备之前的所有备份数据，恢复时间长



- **差异备份：备份上一次全量备份后发生变化的数据**
  - → 优点：数据恢复时间短
  - → 缺点：备份文件大小随时间逐渐增大，存在重复数据



图中的「增量备份」使用了 ROW 快照技术。