引用计数

Qcow2维护refcount用以管理image中cluster的分配和释放，refcount作用等同于引用计数，代表了指定的cluster的使用状态：

0： 表示空闲

1： 表示已使用

大于等于2：表示已使用并且写访问必须执行COW操作

refcounts通过二级表(类似页表)来进行索引，第一级表称为refcount table,其大小可变、连续、占用多个cluster，其表项中每一个条目为指向第二级表的指针(相对于img file的offset)，每个条目占64bits。

第二级表称为refcount block，每个refcount block占用1个cluster，表中每个条目为2个字节大小的refcount。

引用计数是计算机编程语言中的一种内存管理技术，是指将资源（可以是对象、内存或磁盘空间等等）的被引用次数保存起来，当被引用次数变为零时就将其释放的过程。使用引用计数技术可以实现自动资源管理的目的。同时引用计数还可以指使用引用计数技术回收未使用资源的垃圾回收算法。

当创建一个对象的实例并在堆上申请内存时，对象的引用计数就为1，在其他对象中需要持有这个对象时，就需要把该对象的引用计数加1，需要释放一个对象时，就将该对象的引用计数减1，直至对象的引用计数为0，对象的内存会被立刻释放。

使用这种方式进行内存管理的语言：Objective-C

与跟踪式垃圾回收相比，引用计数的主要优点是可以尽快地回收不再被使用的对象，同时在回收过程中不会导致长时间的停顿，还可以清晰地标明每一个对象的生存周期。

在实时应用或内存受限的系统中，实时响应能力是一项重要指标，而引用计数作为最容易实现的垃圾回收技术之一，很适合于这种情况。引用计数还可以用于管理其他非内存资源，如操作系统对象（经常比内存资源更稀缺）。跟踪式垃圾回收技术用终结器处理此类目标，但延迟回收可能引发其他问题。加权引用计数是适用于分布式系统的派生技术。

在可用内存被活跃对象填满的平台上，跟踪式垃圾回收会被频繁触发，从而降低性能。而引用计数即便在内存濒临耗尽的情况下性能依然有所保障。[1]引用计数还能为其他运行时优化技术提供参考信息，例如对于许多使用不可变对象的系统来说（如函数式编程语言），大量复制对象导致的性能惩罚有时十分严重；在此类系统上一个典型的优化措施是：假如一个对象被创建以后仅使用了一次，且在其不再被引用的同时另一个类似的对象被创建出来（如Javascript中的字符串拼接赋值操作），可以将删除原对象创建新对象的行为变为修改原对象，从而提高效率。引用计数可以为这类优化提供充分的参考信息。

未经优化的引用计数相比跟踪式垃圾回收有两个主要缺点，都需要引入附加机制予以修复：

频繁更新引用计数会降低运行效率。

原始的引用计数无法解决循环引用问题。

另外，如果使用空闲列表分配内存，那么引用计数的空间局域性非常差。仅使用引用计数无法通过移动对象来提高CPU缓存的性能，所以高性能的内存分配器都会同时实现一个跟踪式垃圾回收器以提高性能。许多引用计数实现（比如PHP和Objective-C）的性能不佳都是因为没有实现内存拷贝。